Dynamic instance-wise decision-making for machine learning

Yasitha Warahena Liyanage
*University at Albany, State University of New York, yasithashehanliyanage@gmail.com*

The University at Albany community has made this article openly available. Please share how this access benefits you.

Follow this and additional works at: [https://scholarsarchive.library.albany.edu/legacy-etd](https://scholarsarchive.library.albany.edu/legacy-etd)

Part of the [Computer Engineering Commons](https://scholarsarchive.library.albany.edu/legacy-etd/3045)

**Recommended Citation**

[https://scholarsarchive.library.albany.edu/legacy-etd/3045](https://scholarsarchive.library.albany.edu/legacy-etd/3045)

This Dissertation is brought to you for free and open access by the The Graduate School at Scholars Archive. It has been accepted for inclusion in Legacy Theses & Dissertations (2009 - 2024) by an authorized administrator of Scholars Archive. Please see [Terms of Use.](https://scholarsarchive.library.albany.edu/legacy-etd/3045) For more information, please contact scholarsarchive@albany.edu.
DYNAMIC INSTANCE–WISE DECISION–MAKING FOR
MACHINE LEARNING

by

Yasitha Warahena Liyanage

A Dissertation
Submitted to the University at Albany, State University of New York
in Partial Fulfillment of
the Requirements for the Degree of
Doctor of Philosophy

College of Engineering and Applied Sciences
Electrical and Computer Engineering
January 2022
To my parents, wife, sister and brother for their love and continuous encouragement.
ABSTRACT

In a typical supervised machine learning setting, the predictions on all test instances are based on a common subset of features discovered during model training. However, using a different subset of features that are most informative for each test instance individually may improve not only the quality of prediction but also the overall interpretability of the model. To this end, in this dissertation, we study the problem of optimizing the trade–off between instance–level sparsity and the quality of prediction using a dynamic instance–wise decision–making approach. Specifically, this approach sequentially reviews features one at a time for each data instance given previously chosen features and stops this process to predict the instance once it determines that including additional features will not improve the quality of final decision. In contrast to most existing work that utilizes a set of features common for all data instances, this method utilizes different features to predict different data instances.

First, for classification problems where features can be assumed independent, an optimization problem is defined in terms of the number of features and the classification rule with the ultimate goal to minimize feature evaluation and misclassification costs. The optimum solution is derived, and its structure is analyzed. Based on the optimum solution and its properties, two new algorithms are designed. The expected number of features needed to achieve a given classification accuracy is also analytically derived. Finally, the performance of the proposed algorithms is illustrated on several public datasets, thus demonstrating their effectiveness and scalability across a broad range of application domains.

Second, the independence assumption is relaxed, since in many applications, ignoring dependencies between features can lead to poor classification performance. A Bayesian network is utilized to model feature dependencies. Using the dependency network, a new method is proposed that sequentially selects the best feature to evaluate at each stage. The optimum number of features to acquire and the optimum classification strategy are derived for each test instance. Theoretical properties of the optimum solution are analyzed, and a new algorithm is proposed that takes advantage of these properties to implement a robust and scalable solution for high dimensional settings. The effectiveness, generalizability, and scalability of the proposed method are illustrated on a variety of real–world datasets from
diverse application domains.

Third, for classification and regression problems, a more general instance–wise decision–making framework is proposed that decides the number of features and the order by which they are evaluated for each data instance. Specifically, the proposed solution dynamically selects the best feature to review at each stage based on the already observed features. Efficient implementations are proposed for both classification and regression settings. The proposed methods’ effectiveness, generalizability, and scalability are illustrated on various real–world datasets from diverse application domains.

Finally, the model–based and post hoc interpretability of the proposed dynamic instance–wise decision–making framework is validated using an example financial credit risk prediction task. Experimental results show that the proposed method can be used in high–stakes applications, where model interpretations are required without sacrificing test accuracy.
ACKNOWLEDGMENT

First, I would like to express my deepest gratitude and sincere appreciation to my advisor, Dr. Daphne–Stavroula Zois, for her guidance, care, patience, and trust throughout my doctoral studies. Without her great guidance and support, this dissertation would not have been possible. She was being a family member to me, encouraging me and giving kind advice to build my professional life and, more importantly, to stay calm during stressful situations. I would also like to extend my sincere gratitude to my co-advisor, Dr. Charalampos Chelmis for his great mentoring and guidance. I am very fortunate to meet them both early in my career. Words are not enough to describe their kindness, encouragement, and support. I would also like to thank my qualifying and dissertation committee members, Dr. Aveek Dutta, Dr. Gary Saulnier, and Dr. Weifu Wang for their valuable time, intriguing questions, and insightful suggestions regarding my thesis and research. Also, I want to thank all the rest of the faculty, colleagues, and staff from the State University of New York at Albany for their great support.

I would like to extend my gratitude to the Credit Systems Research team at Amazon for their great mentoring to build up my technical skills and analytical thinking during my summer internships. Special thanks to Anthony Penta, Yazdani Mohammad, Xinyu Wang, and Flavio Di Palo for their support and guidance.

I would also like to thank all my teachers at Dhammissra College, Sri Lanka, and lectures at the University of Peradeniya, Sri Lanka. I would like to acknowledge the support and guidance of my previous research advisors, Dr. Roshan Godaliyadda, Dr. Parakrama Ekanayake, Dr. Janaka Ekanayake, Dr. Janaka Wijekulasoriya, and Dr. Lilantha Samarayake. Special thanks to Dr. Shirantha Welikala for his great help and kind support during my bachelor’s studies.

And last, but most importantly, I want to thank my dearest family and friends. I want to thank my mother, Damayanthi Randeni, my father, Sisira Liyanage, my wife, MD. Sahani Liyanaarachchi, my sister, MD. Dashini Liyanage, my brother, Eng. Dasitha Liyanage for their love, affection, and continuous encouragement.
# CONTENTS

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ABSTRACT</td>
<td>iii</td>
</tr>
<tr>
<td></td>
<td>ACKNOWLEDGMENT</td>
<td>v</td>
</tr>
<tr>
<td></td>
<td>LIST OF TABLES</td>
<td>xi</td>
</tr>
<tr>
<td></td>
<td>LIST OF FIGURES</td>
<td>xiv</td>
</tr>
<tr>
<td>1.</td>
<td>Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.1</td>
<td>A Motivating Example</td>
<td>2</td>
</tr>
<tr>
<td>1.2</td>
<td>Interpretable Machine Learning</td>
<td>3</td>
</tr>
<tr>
<td>1.3</td>
<td>Key Challenges</td>
<td>3</td>
</tr>
<tr>
<td>1.4</td>
<td>Related Work</td>
<td>4</td>
</tr>
<tr>
<td>1.4.1</td>
<td>Offline Feature Selection Methods</td>
<td>4</td>
</tr>
<tr>
<td>1.4.2</td>
<td>Streaming Feature Selection during Training</td>
<td>4</td>
</tr>
<tr>
<td>1.4.3</td>
<td>Static Instance–wise Feature Selection during Testing</td>
<td>5</td>
</tr>
<tr>
<td>1.4.4</td>
<td>Dynamic Instance–wise Feature Selection during Testing</td>
<td>5</td>
</tr>
<tr>
<td>1.4.5</td>
<td>Interpretable Machine Learning</td>
<td>6</td>
</tr>
<tr>
<td>1.5</td>
<td>Thesis Outline</td>
<td>6</td>
</tr>
<tr>
<td>2.</td>
<td>Dynamic Instance–wise Joint Feature Selection and Classification with Conditionally Independent Features</td>
<td>9</td>
</tr>
<tr>
<td>2.1</td>
<td>Introduction</td>
<td>9</td>
</tr>
<tr>
<td>2.2</td>
<td>Related Work</td>
<td>10</td>
</tr>
<tr>
<td>2.3</td>
<td>Proposed Framework</td>
<td>12</td>
</tr>
<tr>
<td>2.3.1</td>
<td>Problem Formulation</td>
<td>13</td>
</tr>
<tr>
<td>2.3.2</td>
<td>Optimum Solution</td>
<td>14</td>
</tr>
<tr>
<td>2.3.3</td>
<td>Complexity Analysis</td>
<td>16</td>
</tr>
<tr>
<td>Section</td>
<td>Page</td>
<td></td>
</tr>
<tr>
<td>------------------------------------------------------------------------</td>
<td>------</td>
<td></td>
</tr>
<tr>
<td>2.4 Fast Implementation</td>
<td>17</td>
<td></td>
</tr>
<tr>
<td>2.4.1 Stochastic Gradient Algorithm</td>
<td>19</td>
<td></td>
</tr>
<tr>
<td>2.4.2 Complexity Analysis</td>
<td>21</td>
<td></td>
</tr>
<tr>
<td>2.5 Expected Number of Features for Classification</td>
<td>22</td>
<td></td>
</tr>
<tr>
<td>2.6 Experimental Evaluation</td>
<td>22</td>
<td></td>
</tr>
<tr>
<td>2.6.1 Practical Considerations</td>
<td>23</td>
<td></td>
</tr>
<tr>
<td>2.6.2 Effect of Feature Space Quantization</td>
<td>24</td>
<td></td>
</tr>
<tr>
<td>2.6.3 Accuracy as a Function of Average Number of Features</td>
<td>26</td>
<td></td>
</tr>
<tr>
<td>2.6.4 Experiments on Expected Number of Features</td>
<td>27</td>
<td></td>
</tr>
<tr>
<td>2.6.5 Comparison with Baselines</td>
<td>28</td>
<td></td>
</tr>
<tr>
<td>2.6.6 Scalability to High Dimensional Feature Spaces</td>
<td>31</td>
<td></td>
</tr>
<tr>
<td>2.6.7 Statistical Significance</td>
<td>32</td>
<td></td>
</tr>
<tr>
<td>2.6.8 Benefits of Fast Implementation</td>
<td>32</td>
<td></td>
</tr>
<tr>
<td>2.7 Concluding Remarks</td>
<td>33</td>
<td></td>
</tr>
<tr>
<td>2.8 Appendix</td>
<td>34</td>
<td></td>
</tr>
<tr>
<td>2.8.1 Proof of Theorem 2.1</td>
<td>34</td>
<td></td>
</tr>
<tr>
<td>2.8.2 Proof of Lemma 2.1</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>2.8.3 Proof of Lemma 2.2</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>2.8.4 Proof of Theorem 2.2</td>
<td>37</td>
<td></td>
</tr>
<tr>
<td>2.8.5 Proof of Theorem 2.3</td>
<td>38</td>
<td></td>
</tr>
<tr>
<td>3. Dynamic Instance–Wise Classification in Correlated Feature Spaces</td>
<td>44</td>
<td></td>
</tr>
<tr>
<td>3.1 Introduction</td>
<td>44</td>
<td></td>
</tr>
<tr>
<td>3.2 Related Work</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>3.3 Proposed Framework</td>
<td>47</td>
<td></td>
</tr>
<tr>
<td>3.4 Theoretical Results</td>
<td>51</td>
<td></td>
</tr>
<tr>
<td>3.5 Proposed Algorithm</td>
<td>52</td>
<td></td>
</tr>
</tbody>
</table>
5.3 Interpretability .................................................. 104
  5.3.1 Model–Based Interpretability ............................... 105
    5.3.1.1 Sparsity ................................................. 105
    5.3.1.2 Simulatability ......................................... 107
    5.3.1.3 Modularity ............................................. 107
  5.3.2 Post Hoc Interpretability ............................... 108
    5.3.2.1 Dataset–level Interpretations ...................... 109
    5.3.2.2 Prediction–level Interpretations .................. 110
  5.4 Concluding Remarks ........................................ 111

6. Conclusions and Future Work ............................... 113
  6.1 Summary .................................................... 113
  6.2 Future Directions .......................................... 115
# LIST OF TABLES

## 2.1 Datasets used in the experiments.

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Datasets used in the experiments.</td>
</tr>
</tbody>
</table>

23

## 2.2 Expected number of features required to attain the same accuracy achieved by ETANA using alternative feature distributions (Section 2.6.4). “N.A.” denotes inability to achieve the given accuracy even after using all available features.

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.2</td>
<td>Expected number of features required to attain the same accuracy achieved by ETANA using alternative feature distributions (Section 2.6.4). “N.A.” denotes inability to achieve the given accuracy even after using all available features.</td>
</tr>
</tbody>
</table>

26

## 2.3 Comparison of accuracy. The highest accuracy, and the second highest accuracy are bolded and gray–shaded, and gray–shaded, respectively. Cells are marked with ‘- -’ if the corresponding method was unable to generate results within a cutoff time of 12 days.

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.3</td>
<td>Comparison of accuracy. The highest accuracy, and the second highest accuracy are bolded and gray–shaded, and gray–shaded, respectively. Cells are marked with ‘- -’ if the corresponding method was unable to generate results within a cutoff time of 12 days.</td>
</tr>
</tbody>
</table>

28

## 2.4 Comparison of average number of features used. The minimum and the second minimum average number of features used are bolded and gray–shaded, and gray–shaded accordingly. Cells are marked with ‘- -’ if the corresponding method was unable to generate results within a cutoff time of 12 days.

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.4</td>
<td>Comparison of average number of features used. The minimum and the second minimum average number of features used are bolded and gray–shaded, and gray–shaded accordingly. Cells are marked with ‘- -’ if the corresponding method was unable to generate results within a cutoff time of 12 days.</td>
</tr>
</tbody>
</table>

28

## 2.5 Comparison of time (in seconds) required for feature selection (F), classification (C), joint feature selection and classification (F+C), and model training (T). The minimum and the second minimum F+C times are bolded and gray–shaded, and gray–shaded accordingly. Cells are marked with ‘- -’ if the corresponding method was unable to generate results within a cutoff time of 12 days.

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.5</td>
<td>Comparison of time (in seconds) required for feature selection (F), classification (C), joint feature selection and classification (F+C), and model training (T). The minimum and the second minimum F+C times are bolded and gray–shaded, and gray–shaded accordingly. Cells are marked with ‘- -’ if the corresponding method was unable to generate results within a cutoff time of 12 days.</td>
</tr>
</tbody>
</table>

29

## 3.1 Explanation of main symbols used in this chapter.

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Explanation of main symbols used in this chapter.</td>
</tr>
</tbody>
</table>

47

## 3.2 Datasets used in the experiments.

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.2</td>
<td>Datasets used in the experiments.</td>
</tr>
</tbody>
</table>

55

## 3.3 Accuracy and average number of features used by IFC\(^2\)F for different feature evaluation cost values using Lung2, Dexter, Madelon and MLL datasets.

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.3</td>
<td>Accuracy and average number of features used by IFC(^2)F for different feature evaluation cost values using Lung2, Dexter, Madelon and MLL datasets.</td>
</tr>
</tbody>
</table>

56

## 3.4 Accuracy and average number of features used by IFC\(^2\)F for different dependency structures using Lung2, Dexter, Madelon and MLL datasets.

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.4</td>
<td>Accuracy and average number of features used by IFC(^2)F for different dependency structures using Lung2, Dexter, Madelon and MLL datasets.</td>
</tr>
</tbody>
</table>

56

## 3.5 Computational complexity of baselines. Parameters \(K\) and \(N\) denote number of features and instances, accordingly.

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.5</td>
<td>Computational complexity of baselines. Parameters (K) and (N) denote number of features and instances, accordingly.</td>
</tr>
</tbody>
</table>

57

## 3.6 Comparison of accuracy. The highest accuracy, and the second highest accuracy are bolded and gray–shaded, and gray–shaded, respectively. Cells are marked with ‘- -’ if the corresponding method was unable to generate results within a cutoff time of 12 days.

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.6</td>
<td>Comparison of accuracy. The highest accuracy, and the second highest accuracy are bolded and gray–shaded, and gray–shaded, respectively. Cells are marked with ‘- -’ if the corresponding method was unable to generate results within a cutoff time of 12 days.</td>
</tr>
</tbody>
</table>

58
3.7 Comparison of average number of features used. The minimum and the second minimum average number of features used are bolded and gray–shaded, and gray–shaded accordingly. Cells are marked with ‘- -’ if the corresponding method was unable to generate results within a cutoff time of 12 days.  

3.8 Comparison of time (in seconds) required for feature selection (F), classification (C), joint feature selection and classification (F+C), model training (T) and pre-processing (P). The minimum and the second minimum F+C times are bolded and gray–shaded, and gray–shaded accordingly. Cells are marked with ‘- -’ if the corresponding method was unable to generate results within a cutoff time of 12 days.  

3.9 Words (features) picked by IFC\textsuperscript{2}F are highlighted in yellow. The true/predicted label is given at the end of each review. The second column reports features selected for each review in ascending order (Y–axis) versus feature value (X–axis).  

4.1 Datasets used for Classification.  

4.2 Accuracy and average number of features used by IFCO for four feature evaluation cost $e$ values using Lung2, Dexter, Madelon and MLL datasets.  

4.3 Comparison of accuracy. The highest accuracy, and the second highest accuracy are bolded and gray–shaded, and gray–shaded, respectively. Cells are marked with ‘- -’ if the corresponding method was unable to generate results within a cutoff time of 12 days.  

4.4 Comparison of average number of features used. The minimum and the second minimum average number of features used are bolded and gray–shaded, and gray–shaded accordingly. Cells are marked with ‘- -’ if the corresponding method was unable to generate results within a cutoff time of 12 days.  

4.5 Comparison of time (in seconds) required for feature selection (F), classification (C), joint feature selection and classification (F+C), model training (T) and pre-processing (P). The minimum and the second minimum F+C times are bolded and gray–shaded, and gray–shaded accordingly. Cells are marked with ‘- -’ if the corresponding method was unable to generate results within a cutoff time of 12 days.  

4.6 Words (features) picked by IFCO are highlighted in yellow. The true/predicted label is given at the end of each review. The second column reports features selected in ascending order (Y–axis) versus feature value (X–axis).  

4.7 Datasets used for Regression.
4.8 Comparison of Regression Methods. Minimum and the second minimum values are bolded and gray–shaded, and gray–shaded accordingly. 93

5.1 Credit risk dataset features. 105

5.2 Accuracy (± standard deviation) and average number of features (Feat.) used in the credit risk dataset. 111
LIST OF FIGURES

1.1 Using a \( K \times D \) matrix, (a) existing machine learning models utilize standard feature selection (FS) approaches to extract a set \( \hat{K} \ll K \) of features. This fixed subset is then used to predict all data instances during online decision-making. In contrast, dynamic instance-wise decision-making utilizes different features to predict different data instances online using a model learned offline from all features. ................................................................. 2

2.1 (a) Illustration of Lemma 2.1 for \( L = 3 \) of classes and misclassification costs \( Q_1 = [0, 1, 1], Q_2 = [1, 0, 1] \) and \( Q_3 = [1, 1, 0] \); Illustration of Theorem 2.2: (b) at stage 4 and (c) stage 5, using MLL dataset with misclassification costs \( Q_1 = [0, 1, 1], Q_2 = [1, 0, 1] \) and \( Q_3 = [1, 1, 0] \). The MLL dataset contains 72 samples, each of which comprises 5848 gene expression values belonging to one of 3 diagnostic classes [69]. In both (b) and (c), the blue region corresponds to continuation to the next stage, while the red region corresponds to stopping. ............................................................. 18

2.2 Variation of (a) accuracy, (b) average number of features, and (c) training time (sec) as a function of the number \( V \in \{2, 3, 5, 10, 20, 30, 40, 50, 100\} \) of bins using Lung Cancer, Dexter, Madelon and MLL datasets. ................................. 24

2.3 Distribution of number of features used under 4 different orderings (proposed (c.f. Section 2.6.1), worst (inverse of proposed), two random orderings), and associated accuracy. ............................................................ 25

2.4 Comparison of (a) model training time (seconds), and (b) accuracy achieved by F–ETANA and ETANA using Dexter (2 classes), MLL (3 classes), Lung2 (5 classes), and CAR datasets (11 classes). .................................................. 32

3.1 Sample Bayesian Network of features in \( F \) and the class variable in \( C \). ................ 48

3.2 Graphical illustration of the proposed framework. \( MI(X; C) \) denotes mutual information between variables \( X \) and \( C \). .............................................................. 51

3.3 Variation of (a) accuracy, (b) average number of features, and (c) training time (sec) as a function of the number \( V \in \{2, 4, 6, 8, 12, 20, 30, 40, 50, 100\} \) of bins using Lung2, Dexter, Madelon and MLL datasets. .................................................. 55

3.4 Evolution of posterior probability distribution, i.e., \( P(\text{positive}|F_1, \ldots, F_k) \) (in blue) and \( P(\text{negative}|F_1, \ldots, F_k) \) (in orange) for 4 IMDB reviews in Table 3.9. 62
3.5 Variation of (a) accuracy, and (b) average number of features as the percentage of missing features increases from 0.01% to 50% across datasets. 64

4.1 Variation of (a) accuracy, and (b) average number of features as a function of the number $V \in \{2, 4, 6, 8, 12, 20, 30, 40, 50, 100\}$ of bins using Lung2, Dexter, Madelon and MLL datasets. 84

4.2 Variation of the posterior distributions, i.e., $P(\text{positive}|F_1, \ldots, F_k)$ (in blue) and $P(\text{negative}|F_1, \ldots, F_k)$ (in orange) for four IMDB reviews in Table 4.6. 91

4.3 Variation of (a) accuracy, and (b) average number of features as the percentage of missing features increases from 1% to 50% using Lung Cancer, Dexter, Madelon and MLL datasets. 92

4.4 Performance comparison of IFRO with best performing Baseline (i.e., Tree) on (a) Boston Houses, and (b) Computer Activity datasets (using first 48 test instances – horizontal axis). 93

4.5 Features picked by IFRO (X–axis) on 4 test instances of the Boston Houses dataset. Y–axis: house price $\times$ $1000$. Note that all 4 instances start with the mean of the prior house price distribution. \textbf{LSTAT}: percentage lower status of the population, \textbf{INDUS}: proportion of non-retail business acres per town, \textbf{AGE}: proportion of owner-occupied units built prior to 1940, \textbf{CHAS}: Charles River dummy variable (= 1 if tract bounds river; 0 otherwise), \textbf{PTRATIO}: pupil–teacher ratio by town, \textbf{B} = 1000 × (B_k − 0.63)^2, where $B_k$ is the proportion of blacks by town, \textbf{RM}: average number of rooms per dwelling, \textbf{TAX}: full–value property–tax rate per $10,000$, \textbf{ZN}: Proportion of residential land zoned for lots over 25,000 sq.ft., \textbf{RAD}: index of accessibility to radial highways, \textbf{DIS}: Weighted distances to five Boston employment centers. 94

4.6 Variation of (a) mean absolute error, and (b) average number of features as the percentage of missing features increases from 0.01% to 50%. 95

5.1 Number of features used to classify ten random test instances in the credit risk dataset. Range corresponds to the 95% confidence interval. 106

5.2 First four stages of the IFCO’s decision making process for the credit risk dataset. 108

5.3 Partial dependence plots of features (a) savings account status, $PD(F_6)$, and (b) age, $PD(F_{13})$, using IFCO. savings account status $\in \{0, 1, \ldots, 4\}$ and age is in years. The dark blue line represents the mean, while the light blue region represents the 95% confidence interval. 109
5.4 Normalized feature importance for credit risk dataset.  

5.5 Variation of $P(\text{credit risk} = \text{high}|F_1, \ldots, F_k)$ (in orange) and $P(\text{credit risk} = \text{low}|F_1, \ldots, F_k)$ (in blue) for 4 test instances in the credit risk dataset.
CHAPTER 1

Introduction

Machine learning has revolutionized the world by automating the process of inferring useful patterns hidden in data. The majority of machine learning applications is supervised because some data instances and their actual labels are available in advance to train a robust framework to infer labels of unseen data instances. Fig. 1.1(a) shows the standard supervised machine learning setting. Specifically, the model is provided with $D$ training data instances along with the actual labels. Each data instance is represented using a set of $K$ features. The goal is to predict the labels of the test set. Typically, a feature selector is utilized before model training to select a subset of $\hat{K}$ informative features from the initial set of potentially redundant features to maximize accuracy, speed up the training process, and improve interpretability. Most of the existing work on feature selection [23, 29] extracts a subset of discriminative features that can globally describe the data well, where the same feature subset is used to predict all test instances during online decision-making.

However, a feature is typically acquired at a cost, relating to the time and effort spent in generating it and its discriminative power. For instance, in healthcare, specific tasks may be very informative (e.g., magnetic resonance imaging (MRI)) during the diagnosis process but can be intrusive. In ubiquitous computing, limited energy resources can prevent the continuous collection of high-fidelity data, or users may not wish to reveal answers to specific questions [17]. Similarly, in the domain of robotics, an autonomous vehicle can control the view of its environment (e.g., change position, modify sensors parameters) to inspect and classify an object of interest [28]. In this context, it may be essential to select which sensors to use or what kind of measurements to take while at the same time ensuring that an object in the field of view can be accurately classified. Such cases give rise to the fascinating phenomenon, where the complete set of features can be accessed during the training process, while during the testing process, a complete set of features for all data instances cannot be obtained (i.e., features are different for each instance by choice or due to constraints). Therefore, there is a need to devise a framework that dynamically selects features and performs decision-making individually for each data instance during the testing
Figure 1.1: Using a $K \times D$ matrix, (a) existing machine learning models utilize standard feature selection (FS) approaches to extract a set $\hat{K} << K$ of features. This fixed subset is then used to predict all data instances during online decision–making. In contrast, dynamic instance–wise decision–making utilizes different features to predict different data instances online using a model learned offline from all features.

time to balance the associated cost–accuracy trade–off (see Fig. 1.1(b)).

1.1 A Motivating Example

A wide range of applications, including but not limited to healthcare and robotics, demand practical solutions that can perform feature selection and decision–making jointly in a dynamic setting individually for each data instance. For example, consider a scenario
where a doctor is called to provide a medical diagnosis to a patient. The doctor’s diagnosis may often be time–critical (e.g., in emergencies) and depend on numerous costly medical tests (out of which features are to be extracted), some of which cost thousands of dollars [7]. At the same time, different sets of tests may be appropriate for different patients (i.e., data instances). For example, [34] has shown that relevant features for predicting heart failure may differ across patient subgroups. Considering dependencies between medical tests (and corresponding features) is equally crucial [73], since individual features may seem irrelevant with the target variable when examined independently, but when combined, may improve prediction accuracy and enhance the interpretability of the final decision. At the same time, the order by which tests are conducted for each patient affects the accumulated cost and the final decision [49].

1.2 Interpretable Machine Learning

Recent advances in machine learning set paths to complex function approximators that can achieve high performance in many domains (e.g., computer vision [26], language modeling [55]). However, humans are often reluctant to deploy such complex models in practice, particularly in health care, criminal justice, and financial markets, since they do not have formal justifications about what the model is doing and why it outputs specific decisions [3, 57]. Inherently interpretable machine learning models can be used to discover relevant knowledge about domain relationships in data, debug or justify the model and its outputs, and control and improve the model [46, 45]. Using a sparse set of features to classify data instances is essential for model interpretability [45] since we can explicitly observe which features contribute to each model output. However, optimizing the trade–off between sparsity and prediction accuracy remains a crucial challenge in the field of interpretable machine learning [58].

1.3 Key Challenges

In the context of dynamic instance–wise decision making, the following key challenges remain to be addressed:

1. How to devise a scalable framework to perform dynamic feature selection during the
testing stage?

2. How to handle correlated features?

3. How to select the optimum order to evaluate features?

4. How to handle both discrete and continuous target variables (i.e., classification and regression problems)?

5. How to utilize dynamic instance–wise decision–making for model interpretations?

This thesis considers various aspects related to these challenges and devises appropriate solutions (see Section 1.5 for a detailed description).

1.4 Related Work

In this section, the most relevant prior work is summarized.

1.4.1 Offline Feature Selection Methods

The majority of the state–of–the–art feature selection methods [23] belong to this category, where the goal is to select a global subset of features representing an entire dataset assuming all the features are available in advance. Incorporating L1–norm prior as a regularizer to a standard empirical risk minimizer is the most popular method [23] due to its applicability in both classification and regression problems. However, in real–world applications, where features are not freely available to acquire and feature extraction may be time–consuming, e.g., medical tests in disease diagnosis, batch processing is impractical and unfeasible. Further, selecting the most relevant subset of features, when features arrive sequentially over time, constitutes a more challenging but practical problem to explore than batch processing methods [29].

1.4.2 Streaming Feature Selection during Training

Streaming feature selection methods [53, 74, 66, 71, 75, 29, 76] select a global subset of relevant features to represent an entire dataset when features arrive sequentially during model training. Existing work on this area is roughly divided into two directions depending
on the availability of prior information about the feature space \([74, 66, 71]\) or not \([75]\). In general, various threshold–based approaches have been proposed, where a newly arriving feature is selected if a constraint is satisfied (e.g., predefined threshold \([53]\), dynamically varying threshold \([74]\), conditional independence via \(G^2\)–test \([66]\) or Fisher’s \(Z\)–test \([71]\)). Only a handful of these methods (e.g., \([74]\)) are designed to handle both classification and regression tasks.

### 1.4.3 Static Instance–wise Feature Selection during Testing

Recent studies have shown that the relative importance of each feature may vary across data instances \([9]\). For example, relevant features for predicting heart failure may differ across patient subgroups \([34]\). At the same time, as complex models of machine learning become prevalent, the need to interpret their decisions becomes critical. Hence, instance–wise feature selection \([9, 70, 68]\) tries to identify a few relevant features that explain/predict the output of a machine learning model during testing. These methods can handle both regression and classification tasks \([9, 70, 68]\). However, these methods require revealing all feature values of a test instance to perform feature selection. Further, such methods do not scale for large feature spaces, since the search space grows exponentially with the number of available features.

### 1.4.4 Dynamic Instance–wise Feature Selection during Testing

The work proposed in this thesis belong to this category. In this regime, features arrive sequentially, one at a time during testing, and the goal is to jointly select informative features and predict the label of each data instance. Classification with costly features \([16, 31, 32]\) considers costs associated with feature evaluation and misclassification, and the goal is to limit the number of features used for classification per data instance during testing. These methods formulate the problem using a deterministic Markov decision process (MDP) and solve it using a linear \([16]\) or a non–linear approximation \([31, 32]\) of the associated Q–function. MDP’s state space grows exponentially with the dimension of the feature space, making these methods impractical for high–dimensional settings. Further, these methods are limited to classification tasks.
1.4.5 Interpretable Machine Learning

Inherently interpretable machine learning models can be used to discover relevant knowledge about domain relationships in data, debug or justify the model and its outputs, and control and improve the model [46, 45]. Examples include the generalized additive models (GAMs) [25] and the decision tree. GAMs combine single–feature models through a linear function, identifying the contribution of individual features to the model output. Common GAMs include the logistic regression and the explainable boosting machines, which use linear and boosted decision tree shape functions [42].

Using a sparse set of features to classify data instances is essential for model interpretability [45], since we can explicitly observe which features contribute to each model output. However, in standard settings, sparsity is achieved globally by incorporating a regularizer to the model parameters (e.g., GAMs with L1–norm regularizer), where the same subset of features is used to classify all test instances. In contrast, the decision tree achieves instance–level sparsity by evaluating features along different decision paths, using different features to classify different test instances. Nonetheless, it uses a greedy approach to build the tree structure, where locally optimal splits are obtained at every tree node [19], hence, using more features than necessary.

1.5 Thesis Outline

This Chapter introduced the problem of dynamic instance–wise decision–making, highlighted the critical challenges associated with it, and discussed the most relevant prior work from feature selection, instance–wise decision making, and interpretable machine learning.

Chapter 2 presents the problem of dynamic instance–wise classification for conditionally independent features. We define an optimization problem that simultaneously minimizes the number of features evaluated and the cost associated with the classification decision. We derive the optimum solution using dynamic programming under the assumption that feature distributions are conditionally independent given the target variable. Next, we exploit the structure of the optimum solution and use its properties to design an efficient implementation, which offers near–optimum performance with significant complexity reduction. Further, we analytically derive the expected number of features needed to achieve a given classification
accuracy. Extensive experimental evaluation using several publicly available datasets shows the superiority of the proposed approach in terms of classification accuracy, the average number of features used per data instance, and time required for joint feature selection and classification compared to the state–of–the–art.

Chapter 3 introduces the problem of dynamic instance–wise classification for correlated features. We consider the same optimization problem defined in Chapter 2 that simultaneously minimizes the number of features evaluated and the cost associated with the classification decision. To accommodate correlated features, we model their statistical dependencies using a Bayesian network. We present a novel feature ordering such that each selected feature contains the maximum possible new information about the target variable compared to the already evaluated feature set. We prove that the functions related to the optimum solution (derived using dynamic programming) are concave, continuous, and piecewise linear on the domain of a sufficient statistic. Using these properties, we propose an efficient implementation of the optimum solution. We illustrate the effectiveness and scalability of the proposed method on various real–world datasets.

Chapter 4 addresses the more general problem of predicting both discrete and continuous target variables (i.e., both classification and regression problems) and determining the order by which features must be reviewed. Specifically, we define an optimization problem in terms of the feature ordering, the number of features, and the prediction rule to minimize the average costs associated with the prediction rule and the feature evaluations. We derive the optimum feature ordering, the number of features, and the prediction strategy that needs to be adopted for each data instance separately for classification and regression settings. We propose efficient implementations for classification and regression settings by analyzing the structure of the functions related to the optimum solution and approximating the target variable using a Gaussian mixture model, respectively. We evaluate the proposed methods’ effectiveness, generalizability, and scalability using several publicly available datasets.

Chapter 5 justifies the model–based and post hoc interpretability of the dynamic instance–wise feature selection framework proposed in Chapter 4. Using an example from credit risk prediction, we show the applicability of such a framework in high-stakes applications, where model interpretations are required without sacrificing test accuracy.

Finally, Chapter 6 summarizes the contributions of this Thesis and discusses potential
directions for future research.
CHAPTER 2
Dynamic Instance–wise Joint Feature Selection and Classification
with Conditionally Independent Features

2.1 Introduction

This chapter studies the problem of dynamic instance–wise joint feature selection and classification in the context of supervised machine learning. Specifically, features are examined sequentially one at a time during testing, and the goal is to identify for each data instance individually, which features to use, and how to accurately classify that instance using the selected features. The proposed solution sequentially reviews features and classifies a data instance once it determines that including additional features cannot improve the classification quality further. Furthermore, it utilizes different features to classify different data instances. The proposed solution assumes that the order by which features are reviewed is fixed and common for all data instances, and the features are conditionally independent given the class variable.

In summary, the contributions of this chapter are as follows:

i. The optimum solution (i.e., the feature at which the review process stops, and the associated classification rule) is mathematically derived for each data instance individually.

ii. It is shown that the optimum solution enjoys a number of important mathematical properties.

iii. Two approaches are proposed, one that implements the optimum solution and a fast implementation, which utilizes a stochastic gradient algorithm to estimate the optimal linear thresholds that describe the threshold structure of the optimum solution.

iv. The expected number of features needed to achieve a given classification accuracy is analytically derived.
v. The effectiveness, generalizability, and scalability of the proposed methods is demonstrated on eleven publicly available datasets.

All proofs are included in the Appendix. For reproducibility purposes, the source code of the proposed methods is available at: https://github.com/IDIASLab/ETANA.

2.2 Related Work

In this section, the most relevant prior work on feature selection and classification is summarized.

Traditional feature selection [23] assumes that all candidate features are available during training in a supervised classification setting. This assumption, however, does not often hold in real-world applications, where some features may be expensive to generate in advance. To this end, online feature selection has received considerable attention (for comprehensive surveys, refer to [36, 29, 5]). The majority of such frameworks assumes that features arrive one at a time or in batches during the training process, while all training instances are available before this process starts. Their goal is to choose a subset of features from a larger set of potentially redundant features, while not having access to the full feature space in advance. Thus, various threshold-based approaches have been proposed, where a newly arriving feature is selected if a constraint is satisfied (e.g., predefined threshold [53], dynamically varying threshold [74], conditional independence via $G^2$-test [66] or Fisher’s $Z$-test [71]). Since such methods typically require prior information about the feature space, rough set theory based methods [75, 76] that do not require such knowledge have been recently explored. In [75], features are selected if they exhibit high correlation with the class variable and low correlation with already selected features, while the boundary region of the decision is kept as small as possible. [76] adopts the same approach, but also considers the relevant size of the neighborhood of each data instance.

It is important to distinguish online feature selection from the problem studied in this dissertation. In those works, features arrive sequentially during the training process, and the same set of selected features is used for classifying all instances during the testing process. This is in sharp contrast from the setting herein, where the full feature space is accessible during the training process, and a different subset of features is dynamically selected and
used for classifying each instance during the testing process.

Online learning methods assume that training instances arrive sequentially in a supervised classification setting, and are used to iteratively update the function of a classifier [27]. This is in contrast to batch learning techniques, where a collection of training data is used to train a classifier offline, without further updates once training is complete. To address real-world applications that involve high-dimensional data, sparse online learning has also been explored, where the goal is to learn a sparse linear classifier from a sequence of high-dimensional training instances (see [65, 67] and references there in). In [65], sparsity regularization and truncation in conjunction with exploration–exploitation are employed in the case where access to the full feature space is allowed/limited. To enable scaling in larger dimensions, [67] adopts a second–order online learning approach that assumes that the weights of the linear classifier follow a Gaussian distribution. Methods like the above typically require the user to define the number of non-zero elements (i.e., features) to be used by the model. As a result, the resulting sparse representation of the feature space is global to the entire dataset, which implies that the same fixed set of features is used during the testing process. This is different from the work presented herein, where all data instances can be accessed during the training process, and the aim is to devise an algorithm that can dynamically select and use for classification different features for each instance during the testing process.

To facilitate the ease of interpretation [46] of popular but complex machine learning models, instance-wise feature selection [9, 70, 68] has recently gained attention. Motivated by the fact that relevant features may differ across data instances (e.g., heart failure prediction [34]), this line of work aims to identify a small number of relevant features that explain the output of a model during testing. In [9], a feature selector method is proposed based on mutual information principles and a single neural network to identify the top k features that explain a pretrained model. [70] adopts an actor–critic architecture with three neural networks bypassing the need for backpropagation and achieves better predictive performance. To avoid the high computational cost of the above methods, [68] limits the number of possible relevant feature subsets to K, modeling such constraint by a mixture of K deep neural networks and using the sensitivity’s magnitude of the model to select the most relevant features. All above methods require access to all feature values of a test instance before
selecting the appropriate features, and do not scale for large feature spaces. Instead, in the proposed approach, features are examined sequentially one at a time during testing and the goal is to jointly select features and classify each data instance. In addition, the number of features used for each instance is optimally derived by the proposed framework and not arbitrarily set by the user a priori. Finally, contrary to existing work, the methods proposed in this chapter scale to large feature spaces with more than 1 million features.

The work herein is closely related to classification with costly features [16, 31, 32], whose goal is to limit the number of features used for classification per data instance during the testing process without requiring a fixed order for feature selection. In [16], a Markov decision process formulation is proposed and solved via linearly approximated Q-learning. Inspired by the recent advances in neural networks, [31] replaces the linear approximation in [16] with various state-of-the-art deep reinforcement learning techniques, and allows the use of an external high-performance classifier trained separately with all features to improve performance. Finally, [32] considers a different formulation of the same problem with a per-sample budget constraint. Both average and hard budget variations are considered under complete/missing feature sets, the solutions of which are acquired using deep reinforcement learning. The work herein is conceptually different from [16, 31, 32] in that the instance-wise joint feature selection and classification problem is explicitly addressed by defining an optimization problem with respect to each data instance, not the whole training dataset. Even though [16, 31, 32] employ sequential decision-making formalisms, the proposed modeling choices differ significantly, making it possible to perform instance-wise joint feature selection and classification on large-scale datasets. Additionally, the optimum solution is analytically derived, its properties are analyzed and the expected number of features needed to achieve a given classification accuracy is theoretically characterized.

2.3 Proposed Framework

This section formulates the optimization problem, and provides the optimum solution along with the associated computational complexity.
2.3.1 Problem Formulation

Consider a set $S$ of data instances, with each data instance $s \in S$ being described using an assignment of values $f = \{f_1, f_2, \ldots, f_K\}$ to a set $F = \{F_1, F_2, \ldots, F_K\}$ of $K$ features. Each data instance $s$ is drawn from some probability distribution $P(F = f)$ over the feature space. Further, each instance $s$ may belong to one of $L$ classes, with corresponding a priori probability $P(C = c_i) = p_i$ for each assignment $c_i, i = 1, 2, \ldots, L$, of the class variable $C$. Moreover, coefficients $e_k > 0, k = 1, 2, \ldots, K$, represent the cost of evaluating features $F_k$, respectively, and coefficients $Q_{ij} \geq 0, i, j \in \{1, \ldots, L\}$, denote the misclassification cost of selecting class $c_j$ when class $c_i$ is true.

To select one out of $L$ possible classes for each data instance $s$, the proposed approach evaluates features sequentially, where at each step it has to decide between stopping and continuing the feature evaluation process based on the accumulated information thus far and the cost of evaluating the remaining features. Herein, a pair of random variables $(R, D_R)$ is introduced. $R \in \{0, 1, \ldots, K\}$ (referred to as stopping time [60] in decision theory) denotes the feature at which the framework stops the evaluation process and assigns $s$ to a specific class, and $D_R \in \{1, \ldots, L\}$, which depends on $R$, denotes the possibility to select among the $L$ classes. The event $\{R = k\}$ depends only on the feature set $\{F_1, F_2, \ldots, F_k\}$, whereas the event $\{D_R = j\}$ represents choosing class $c_j$ based on information accumulated up to feature $R$. The goal is to select random variables $R$ and $D_R$ by solving the following optimization problem:

$$
\min_{R, D_R} J(R, D_R),
$$

where the cost function is defined as:

$$
J(R, D_R) \triangleq \mathbb{E}\left\{ \sum_{k=1}^{R} e_k \right\} + \sum_{j=1}^{L} \sum_{i=1}^{L} Q_{ij} P(D_R = j, C = c_i),
$$

in which the first term denotes the cost of evaluating features, and the second term penalizes misclassification errors. To avoid the computational complexity associated with considering all possible feature orderings, it is assumed that the ordering by which features are reviewed is fixed and common for all data instances (see Section 2.6.1 for a heuristic ordering).
2.3.2 Optimum Solution

The optimization problem in Eq. (2.1) is intractable for general dependencies between features. To simplify the derivation of the proposed framework, we assume that features are conditionally independent given the class variable. This leads to an approach that achieves a good balance between accuracy and number of features used per instance, as indicated by our experimental evaluation results.

To solve the optimization problem defined in Eq. (2.1), the \textit{a posteriori probability} vector $\pi_k \triangleq [\pi_1^k, \pi_2^k, \ldots, \pi_L^k]^T$ is introduced, where the $k$th feature is evaluated to generate outcome $f_k$, and $\pi_i^k \triangleq P(c_i|F_1, \ldots, F_k)$. To simplify the notation, $P(c_i|F_1, \ldots, F_k)$ is hereafter used in lieu of $P(C = c_i|F_1 = f_1, \ldots, F_k = f_k)$. The posterior probability vector $\pi_k$ can be recursively updated via Bayes’ rule as:

$$
\pi_k = \frac{\text{diag}(\Delta_k(F_k))\pi_{k-1}}{\Delta_k^T(F_k)\pi_{k-1}},
$$

(2.3)

where $\Delta_k(F_k) \triangleq [P(F_k|c_1), P(F_k|c_2), \ldots, P(F_k|c_L)]^T$, $\text{diag}(A)$ denotes a diagonal matrix with diagonal elements being the elements in vector $A$, and $\pi_0 \triangleq [p_1, p_2, \ldots, p_L]^T$.

Next, the probability $P(D_R = j, C = c_i)$ is simplified by exploiting the definition of the \textit{a posteriori} probability $\pi_i^R$. Specifically, based on the fact that $x_R = \sum_{k=0}^K x_k \mathbb{1}_{\{R = k\}}$ for any sequence of random variables $\{x_k\}$, where $\mathbb{1}_A$ is the indicator function for event $A$ (i.e., $\mathbb{1}_A = 1$ when $A$ occurs, and $\mathbb{1}_A = 0$ otherwise), the probability $P(D_R = j, c_i)$ can be written as follows:

$$
P(D_R = j, c_i) = \mathbb{E}\{\pi_i^R \mathbb{1}_{\{D_R = j\}}\}.
$$

(2.4)

Using Eq. (2.4), the average cost in Eq. (2.2) can be written compactly as:

$$
J(R, D_R) = \mathbb{E}\left\{\sum_{k=1}^R e_k + \sum_{j=1}^L \left(\sum_{i=1}^L Q_{ij} \pi_i^R\right) \mathbb{1}_{\{D_R = j\}}\right\},
$$

(2.5)

which in turn can be rewritten as follows:

$$
J(R, D_R) = \mathbb{E}\left\{\sum_{k=1}^R e_k + \sum_{j=1}^L Q_j^T \pi_R \mathbb{1}_{\{D_R = j\}}\right\},
$$

(2.6)
where $Q_j \triangleq [Q_{1,j}, Q_{2,j}, \ldots, Q_{L,j}]$.

To obtain the optimum stopping time $R$, the optimum classification rule $D_R$ for any given $R$ must first be obtained. To this end, a lower bound (independent of $D_R$) must be acquired for the second term inside the expectation in Eq. (2.6), which is the part that depends on $D_R$. At any stopping time $R$, the optimum decision $D_R$ takes only 1 out of $L$ possibilities such that the misclassification cost is minimum, and thus, $\sum_{j=1}^{L} 1_{\{D_R=j\}} = 1$. Furthermore, $Q_j^T \pi_R \geq g(\pi_R), \forall j \in \{1, 2, \ldots, L\}$, where $g(\pi_R) \triangleq \min_{1 \leq j \leq L} [Q_j^T \pi_R]$. Since $1_{\{D_n=j\}}$ is non-negative, it can be shown that:

$$\sum_{j=1}^{L} (Q_j^T \pi_R) 1_{\{D_R=j\}} \geq g(\pi_R) \sum_{j=1}^{L} 1_{\{D_R=j\}} = g(\pi_R).$$

The lower bound $g(\pi_R)$ is independent of the decision $D_R$. Furthermore, such lower bound can be achieved only by the optimum classification rule given by:

$$D_R^{optimum} = \arg \min_{1 \leq j \leq L} [Q_j^T \pi_R],$$

(2.8)

which is therefore the optimum decision for a given stopping time $R$. From Eq. (2.8), it is shown that:

$$J(R, D_R) \geq J(R, D_R^{optimum}),$$

where

$$J(R, D_R^{optimum}) = \min_{D_R} J(R, D_R).$$

(2.9)

Thus, the cost function in Eq. (2.6) can be reduced to one which depends only on the stopping time $R$ as follows:

$$\tilde{J}(R) = \mathbb{E}\left\{ \sum_{k=1}^{R} e_k + g(\pi_R) \right\}.$$

(2.10)

To optimize the cost function in Eq. (2.10) with respect to $R$, the following optimization problem needs to be solved:

$$\min_{R \geq 0} \tilde{J}(R) = \min_{R \geq 0} \mathbb{E}\left\{ \sum_{k=1}^{R} e_k + g(\pi_R) \right\}.$$
Since \( R \in \{0, 1, \ldots, K\} \), the optimum strategy consists of a maximum of \( K + 1 \) stages, where the optimum solution must minimize the corresponding average cost going from stages 0 to \( K \). The solution can be obtained using dynamic programming [8].

**Theorem 2.1.** For \( k = K - 1, \ldots, 0 \), function \( \bar{J}_k(\pi_k) \) is related to \( \bar{J}_{k+1}(\pi_{k+1}) \) through the equation:

\[
\bar{J}_k(\pi_k) = \min \left[ g(\pi_k), \bar{A}_k(\pi_k) \right],
\] (2.12)

where \( \bar{A}_k(\pi_k) = \epsilon_{k+1} + \sum_{F_{k+1}} \Delta_{k+1}^T(F_{k+1}) \pi_k \times J_{k+1} \left( \frac{\text{diag} \left( \Delta_{k+1}(F_{k+1}) \pi_k \right)}{\Delta_{k+1}^T(F_{k+1}) \pi_k} \right) \), with \( J_K(\pi_K) = g(\pi_K) \). Therefore, the optimum feature selection strategy is \( \{F_{i_1}, F_{i_2}, \ldots, F_{R_{\text{optimum}}}\} \), where \( R_{\text{optimum}} \) is equal to the first \( k < K \) for which \( g(\pi_k) \leq \bar{A}_k(\pi_k) \), or \( R_{\text{optimum}} = K \) if there are no more features to be evaluated.

The optimum stopping strategy derived from Eq. (2.12) has a very intuitive structure. Specifically, it stops at stage \( k \) when the cost of stopping (the first expression in the minimization) is no greater than the expected cost of continuing given all information accumulated at the current stage \( k \) (the second expression in the minimization). Equivalently, at each stage \( k \), the proposed method faces two options given \( \pi_k \): (i) stop evaluating features and select optimally between the \( L \) classes, or (ii) continue with the next feature. The cost of stopping is \( g(\pi_k) \), whereas the cost of continuing is \( \bar{A}_k(\pi_k) \).

Based on Eqs. (2.3), (2.8) and Theorem 2.1, a dynamic instance–wise feature selection and classification Algorithm (ETANA), is presented. Initially, the posterior probability vector \( \pi_0 \) is set to \( [p_1, p_2, \ldots, p_L] \), and the two terms in Eq. (2.12) are compared. If the first term is less than or equal to the second term, ETANA classifies the instance under examination to the appropriate class, based on the optimum rule in Eq. (2.8). Otherwise, the first feature is evaluated. ETANA repeats these steps until either it decides to classify the instance using \( < K \) features, or using all \( K \) features.

### 2.3.3 Complexity Analysis

During training, Eq. (2.12) is recursively solved offline. This is achieved by quantizing the interval \([0, 1]\) over \( L \) values such that \( \sum_{i=1}^L \pi_k^i = 1 \) to generate different possible vectors
π_k. Then, a K × d matrix is computed, where each row contains values of the function \( \tilde{A}_k(\pi_k) \), \( k = 0, 1, \ldots, K - 1 \), evaluated using Theorem 2.1 for all possible d vectors of π_k using backward iteration. The complexity of computing a single value of function \( \tilde{A}_k(\pi_k) \), \( k \in \{0, \ldots, K-1\} \) is \( O(LV) \), since there is a sum of V terms, where each term in this summation requires to compute a dot product between a pair of L–dimensional vectors. Here, V is the number of bins considered when quantizing the feature space (see Section 2.6.1 for details). Since there are \( K \times d \) elements in the table, the overall complexity is \( O(KdLV) \). However, d is \( O(\eta^{-1}) \), where 1/\( \eta \) is the arithmetic precision of every element π_i, \( i \in \{1, \ldots, L\} \) in the probability vector. For example, for \( \eta = 10 \), there are \( O(10^{L-1}) \) different L–dimensional probability vectors with each element having an arithmetic precision up to 0.1. Computing \( g(\pi_k) \) for all possible \( \pi_k \) vectors is \( O(L^2 \eta^{L-1}) \). Thus, the time complexity of training ETANA is \( O(KVL \eta^{L-1} + L^2 \eta^{L-1}) \), since at most K features must be reviewed in the worst case.

During testing, ETANA uses the \( K \times d \) matrix generated during training to perform dynamic joint feature selection and classification. The computational complexity of this process is analyzed next. The complexity of acquiring a new feature is \( O(1) \), while updating \( \pi_k \) using Eq. (2.3) is \( O(L) \), since there is a dot product between a pair of L–dimensional vectors. Selecting between continuing or stopping the feature evaluation process by comparing the two terms, \( g(\pi_k) \) and \( \tilde{A}_k(\pi_k) \), in Eq. (2.12) is \( O(1) \). Once the process stops, determining the optimum classification decision using Eq. (2.8) is \( O(L^2) \). Hence, ETANA can jointly select features and classify an instance in \( O(KL + L^2) \).

2.4 Fast Implementation

To overcome the exponential time complexity of training ETANA (see Section 2.3.3), this section presents a Fast version of ETANA (F–ETANA) that exploits structural properties of the optimum classification rule in Eq. (2.8) and the optimum stopping strategy in Eq. (2.12).

Consider a general form of the function \( g(\pi_R) \) used to derive the optimum classification rule in Eq. (2.8) as follows:

\[
g(\varpi) \triangleq \min_{1 \leq j \leq L} \left[ Q_j^T \varpi \right], \ var \in [0, 1]^L,
\]  

(2.13)
Figure 2.1: (a) Illustration of Lemma 2.1 for $L = 3$ of classes and misclassification costs $Q_1 = [0, 1, 1], Q_2 = [1, 0, 1]$ and $Q_3 = [1, 1, 0]$; Illustration of Theorem 2.2: (b) at stage 4 and (c) stage 5, using MLL dataset with misclassification costs $Q_1 = [0, 1, 1], Q_2 = [1, 0, 1]$ and $Q_3 = [1, 1, 0]$. The MLL dataset contains 72 samples, each of which comprises 5848 gene expression values belonging to one of 3 diagnostic classes [69]. In both (b) and (c), the blue region corresponds to continuation to the next stage, while the red region corresponds to stopping.

where $\varpi = [\omega_1, \ldots, \omega_L]^T$, such that $\omega_i \geq 0, \sum_{i=1}^L \omega_i = 1$. Here, the domain of $g(\varpi)$ is the probability space of $\varpi$, which is a $L - 1$ dimensional unit simplex. Function $g(\varpi)$ has some interesting properties as described in Lemma 2.1.

**Lemma 2.1.** Function $g(\varpi)$ is concave, continuous, and piecewise linear and consist of at most $L$ hyperplanes.

Fig. 2.1(a) shows a visualization of Lemma 2.1, when $L = 3$, so that the domain of $g(\varpi)$ is a 2-dimensional unit simplex (i.e., an equilateral triangle). Next, the general form of the optimum stopping strategy in Eq. (2.12) is considered:

$$\bar{J}_k(\varpi) = \min [g(\varpi), \bar{A}_k(\varpi)],$$

(2.14)

where $\bar{A}_k(\varpi) = \epsilon_{k+1} + \sum_{F_{k+1}} \Delta_{k+1}^T(F_{k+1})\varpi \times \bar{J}_{k+1} \left( \frac{\text{diag}(\Delta_{k+1}(F_{k+1})) \varpi}{\Delta_{k+1}(F_{k+1})\varpi} \right)$. Lemma 2.2 summarizes the key properties enjoyed by this function.

**Lemma 2.2.** The functions $\bar{J}_k(\varpi), k = 0, \ldots, K - 1$, are concave, continuous, and piecewise linear.
The fact that $g(\varpi)$ and $J_k(\varpi)$ are concave and piecewise linear allows for their compact representation. Recall that according to Theorem 2.1, the method stops at stage $k$ whenever $g(\varpi) \leq \bar{A}_k(\varpi)$, where $\bar{A}_k(\varpi)$ is the optimum cost-to-go at stage $k$. In particular, to decide between continuing and stopping, it is sufficient to keep track of the thresholds at the intersections of $g(\varpi)$ with every $\bar{A}_k(\varpi)$ as stated in Theorem 2.2 below.

**Theorem 2.2.** At every stage $k$, there exists at most $L$ threshold curves that separate the unit simplex into regions which alternatively switch between continuation to the next stage and stopping. In particular, the region starting from every corner of the $L-1$ dimensional unit simplex always corresponds to stopping the feature evaluation process.

Theorem 2.2 suggests that the region where the a posteriori probability vector $\pi$ falls into will help decide between continuing to the next stage or stopping. This provides an alternative fast implementation of the optimum solution using thresholds. Figs. 2.1(b) and (c) show a visualization of Theorem 2.2; both sub-figures (b) and (c) contain maximum number of threshold curves (i.e., 3 since $L = 3$).

### 2.4.1 Stochastic Gradient Algorithm

In this section, a stochastic gradient algorithm is proposed to estimate the threshold curves described in Theorem 2.2. For ease of implementation, the approximation is limited to linear threshold curves of the form given in Eq. (2.15).

Let $\theta_{\tilde{D}} \triangleq [\theta_{\tilde{D}}^1, \theta_{\tilde{D}}^2, \ldots, \theta_{\tilde{D}}^L]$ denote the parameters of a linear hyperplane, where $\tilde{R}$ is the number of features evaluated so far, and $\tilde{D}_{\tilde{R}} = j, \tilde{R} \in \{0, 1, \ldots, K\}, j \in \{1, \ldots L\}$ represents a decision choice. Then, the decision $Z_{\theta}$ to “stop” or “continue” at each stage $\tilde{R}$ under the decision choice $\tilde{D}_{\tilde{R}} = j$, as function of $\varpi$, is defined as follows:

$$Z_{\theta_{\tilde{D}}} (\varpi) = \begin{cases} \text{stop,} & \text{if} \quad \theta_{\tilde{D}}^T \varpi \leq 0 \\ \text{continue,} & \text{otherwise} \end{cases} \quad (2.15)$$

Decision $Z_{\theta_{\tilde{D}}}$ is indexed by $\theta_{\tilde{D}}$ to show the explicit dependency of the parameters on the decision, where $\theta_{\tilde{D}} \triangleq [\theta_{\tilde{D}_0}, \theta_{\tilde{D}_1}, \ldots, \theta_{\tilde{D}_{K-1}}] \in \mathbb{R}^{K \times L}, \tilde{D}_{\tilde{R}} = j, \forall \tilde{R}$, is the concatenation of $\theta_{\tilde{D}_{\tilde{R}}}$ vectors, one for each stage $\tilde{R}$. Now, recall the cost function in Eq. (2.10). Since the goal
is to find linear thresholds for each decision choice $\tilde{D}_{\tilde{R}} = j$ independently, a modified version of the cost function in Eq. (2.10) is used:

$$
\tilde{H}(\theta_{\tilde{D}}) = \mathbb{E}_{x_{\tilde{D}}} \left\{ \sum_{k=1}^{\tilde{R}} e_k + Q_{\tilde{D},\tilde{R}} \pi_{\tilde{R}} \right\}.
$$

(2.16)

Algorithm 1: Stochastic Gradient Algorithm for Estimating Optimal Linear Thresholds

**Require:** Initial parameters $\theta_{\tilde{D},0}$

**Output:** Optimal parameters $\theta_{\tilde{D},opt}$

1: **for** iterations $t = 0, 1, 2, \ldots$ **do**
2: Evaluate $\tilde{H}(\theta_{\tilde{D},t} + \beta_t \xi_t)$ and $\tilde{H}(\theta_{\tilde{D},t} - \beta_t \xi_t)$ using Function 2 based on Eq. (2.16)
3: Estimate $\hat{\nabla}_{\theta_{\tilde{D}}} \tilde{H}(\theta_{\tilde{D},t})$ using Eq. (2.17)
4: Update $\theta_{\tilde{D},t}$ to $\theta_{\tilde{D},t+1}$ using Eq. (2.18)
5: Stop if $||\hat{\nabla}_{\theta_{\tilde{D}}} \tilde{H}(\theta_{\tilde{D},t})||_2 \leq \rho$ or maximum number $t_{max}$ of iterations reached
6: **end for**
7: **return** $\theta_{\tilde{D},opt}$

Algorithm 1 generates a sequence of estimates $\theta_{\tilde{D},t}$ by computing the gradient $\nabla_{\theta_{\tilde{D}}} \tilde{H}(\theta)$. Here, $\theta_{\tilde{D},t}$ denotes the estimate of $\theta_{\tilde{D}}$ at iteration $t$. Although evaluating the gradient in closed form is intractable due to the non-linear dependency of $\tilde{H}(\theta_{\tilde{D}})$ and $\theta_{\tilde{D}}$, estimate $\hat{\nabla}_{\theta_{\tilde{D}}} \tilde{H}(\theta_{\tilde{D}})$ can be computed using a simulation-based gradient estimator. Among the several simulation-based gradient estimators in the literature [54], the simultaneous perturbation stochastic approximation (SPSA) algorithm [62] is used herein for simplicity. SPSA estimates the gradient at each iteration $t$ using a finite difference method and a random direction $\xi_t$, as follows:

$$
\hat{\nabla}_{\theta_{\tilde{D}}} \tilde{H}(\theta_{\tilde{D},t}) = \frac{\tilde{H}(\theta_{\tilde{D},t} + \beta_t \xi_t) - \tilde{H}(\theta_{\tilde{D},t} - \beta_t \xi_t)}{2\beta_t} \xi_t,
$$

(2.17)

where $\xi^i_t = \begin{cases} -1, & \text{with probability 0.5} \\ +1, & \text{with probability 0.5} \end{cases}$.

Using the gradient estimate in Eq. (2.17), parameter $\theta_{\tilde{D},t}$ is updated as follows:

$$
\theta_{\tilde{D},t+1} = \theta_{\tilde{D},t} - a_t \hat{\nabla}_{\theta_{\tilde{D}}} \tilde{H}(\theta_{\tilde{D},t}),
$$

(2.18)
where \( a_t \) and \( \beta_t \) are typically chosen as in [62]:

\[
\begin{align*}
    a_t &= \varepsilon (t + 1 + \varsigma)^{-\kappa}, \quad 0.5 < \kappa \leq 1, \quad \varepsilon, \varsigma > 0 \\
    \beta_t &= \mu (t + 1)^{-\upsilon}, \quad 0.5 < \upsilon \leq 1, \quad \mu > 0.
\end{align*}
\] (2.19)

Algorithm 1 is guaranteed to converge to a local minimum with probability one [62]. The following stopping criteria are adopted: \( ||\hat{\nabla} \theta \tilde{H}^{\theta_{\tilde{D},t}}||_2 \leq \rho \), or algorithm stops when it reaches a user-defined maximum number of iterations. Finally, \( \tilde{H}(.) \) in Eq. (2.16) is estimated using Function 2.

**Function 2** \( \tilde{H}(.) \)

**Require:** parameter \( \theta_{\tilde{D}} \) and \( \pi_0 \)

**Output:** \( \tilde{H}(\theta_{\tilde{D}}) \)

**Initialization:** \( k = 0 \) and \( \tilde{H} = 0 \)

1. while \( \theta_{\tilde{D},t}^T \pi_k \geq 0 \) do
2. \( k = k + 1 \)
3. Obtain a new feature \( F_k \)
4. Update \( \pi_k \) using Eq. (2.3)
5. \( \tilde{H} = \tilde{H} + e_k \)
6. end while
7. \( \tilde{H} = \tilde{H} + Q_{\tilde{D},t}^T \pi_k \)
8. return \( \tilde{H} \)

### 2.4.2 Complexity Analysis

During training, Algorithm 1 is used for determining the optimum linear thresholds. In Algorithm 1, i) evaluating \( \tilde{H}(\theta_{\tilde{D},t} + \beta_t \xi) \) and \( \tilde{H}(\theta_{\tilde{D},t} - \beta_t \xi) \) using Function 2 is \( \mathcal{O}(KL) \), ii) computing \( \hat{\nabla} \theta_{\tilde{D}} \tilde{H}(\theta_{\tilde{D},t}) \) using Eq. (2.17) is \( \mathcal{O}(KL) \), iii) updating \( \theta_{\tilde{D},t} \) to \( \theta_{\tilde{D},t+1} \) using Eq. (2.18) is \( \mathcal{O}(KL) \), and iv) computing \( ||\hat{\nabla} \theta_{\tilde{D}} \tilde{H}(\theta_{\tilde{D},t})||_2 \) is \( \mathcal{O}(KL) \). Thus, the complexity of Algorithm 1 is \( \mathcal{O}(KL t_{max}) \), where \( t_{max} \) is a user-defined threshold on the maximum number of iterations allowed. Since Algorithm 1 must be run for every decision choice \( \tilde{D} \), the computational complexity of training F–ETANA is \( \mathcal{O}(KL^2 t_{max}) \).

During testing, F–ETANA acquires a new feature in \( \mathcal{O}(1) \), and updates \( \pi_k \) using Eq. (2.3) in \( \mathcal{O}(L) \). Selecting between continuing or stopping the feature evaluation process based on \( \pi_k \) and the \( L \) linear thresholds requires \( \mathcal{O}(L^2) \). Once the process stops, determining
the optimum classification decision using the \( L \) linear thresholds is \( O(L^2) \). Hence, F–ETANA can jointly select features and classify an instance in \( O(KL^2) \).

### 2.5 Expected Number of Features for Classification

In this section, the expected number \( R \) of features to be evaluated for a given classification accuracy is analytically derived. Although independent and identically distributed features are assumed, and the focus is on one–versus–the–rest classification [6], useful insights into the interplay between feature selection and classification can be gained.

**Theorem 2.3.** For independent and identically distributed feature distributions, achieving error probability \( P_e \equiv P(C = c_i | c_i) \), such that \( P_e \leq \gamma \), with \( Q_{ij} = 1, \forall i \neq j, Q_{ii} = 0, i, j \in \{1, 2, \ldots, L\} \) and \( e_k = e, \forall k \), requires:

\[
\mathbb{E}\{R|c_i\} = \frac{(1 - 2\gamma) \log \frac{1-e}{e} + \log \frac{P(\tilde{c}_i)}{P(c_i)}}{\sum_{v=1}^{V} \theta_v \log \frac{\theta_v}{\bar{\theta}_v}},
\]

(2.20)

features, where \( \{\tilde{c}_i\} \triangleq \{\cup_{j=1,j\neq i}^{L} c_j\} \), and \( \theta_v, \bar{\theta}_v \) are the parameters of the feature distributions.

Examining the expression in Eq. (2.20) reveals that there is an inherent trade–off between the error probability bound \( \gamma \) and the expected number of features needed to achieve such a bound. Additionally, the term in the denominator corresponds to the Kullback–Leibler (KL) divergence [12] between distributions \( P(F|c_i) \) and \( P(F|\tilde{c}_i) \). Intuitively, since KL divergence measures the distance between these two distributions, if this distance is large, less features are on average needed to achieve a given accuracy.

### 2.6 Experimental Evaluation

In this section, an extensive set of experiments is conducted to evaluate the performance of ETANA and F–ETANA using 11 benchmark datasets: 6 DNA Microarray Datasets (Lung Cancer, Lung2, MLL, Car, Leukemia, Prostate) [69], 4 NIPS feature selection challenge datasets (Dexter, Madelon, Dorothea, Spambase) [1], and 1 high dimensional dataset (News20) [2]. Table 2.1 summarizes these datasets. For Madelon, MLL, Dexter and Dorothea datasets, the originally provided training and validation sets are used, while for the remain-
Table 2.1: Datasets used in the experiments.

<table>
<thead>
<tr>
<th>Dataset</th>
<th># Instances</th>
<th># Features</th>
<th># classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Madelon</td>
<td>2,000</td>
<td>500</td>
<td>2</td>
</tr>
<tr>
<td>Lung Cancer</td>
<td>181</td>
<td>12,533</td>
<td>2</td>
</tr>
<tr>
<td>MLL</td>
<td>72</td>
<td>5,848</td>
<td>3</td>
</tr>
<tr>
<td>Dexter</td>
<td>300</td>
<td>20,000</td>
<td>2</td>
</tr>
<tr>
<td>Car</td>
<td>174</td>
<td>9,182</td>
<td>11</td>
</tr>
<tr>
<td>Lung2</td>
<td>203</td>
<td>3,312</td>
<td>5</td>
</tr>
<tr>
<td>Leukemia</td>
<td>72</td>
<td>7,129</td>
<td>2</td>
</tr>
<tr>
<td>Prostate</td>
<td>102</td>
<td>6,033</td>
<td>2</td>
</tr>
<tr>
<td>Spambase</td>
<td>4601</td>
<td>57</td>
<td>2</td>
</tr>
<tr>
<td>Dorothea</td>
<td>800</td>
<td>100,000</td>
<td>2</td>
</tr>
<tr>
<td>News20</td>
<td>19,996</td>
<td>1,355,191</td>
<td>2</td>
</tr>
</tbody>
</table>

ing datasets, five-fold cross validated results are reported. All experiments are conducted on a commodity PC with Intel(R) Core(TM) i7-7700 @3.60 GHz CPU, 16 GB memory, and running Windows 10 Pro, 64 bit operating system.

2.6.1 Practical Considerations

A smoothed maximum likelihood estimator is used to estimate \( p(F_k|c_i) \), \( k = 1, \ldots, K \), \( i = 1, \ldots, L \), after quantizing the feature space. Specifically, \( \hat{p}(F_k|c_i) = \frac{S_{k,i} + 1}{S_i + V} \), where \( S_{k,i} \) denotes the number of instances that satisfy \( F_k = f_k \) and belong to class \( c_i \), \( S_i \) denotes the total number of instances belonging to class \( c_i \), and \( V \) is the number of bins considered. The effect of the number \( V \) of bins on the performance of ETANA is studied in Section 2.6.2. The a priori probabilities are estimated as \( P(c_i) = \frac{S_i}{\sum_{i=1}^{L} S_i} \), \( i = 1, \ldots, L \).

Feature ordering is crucial for early stopping, as different features can hinder or facilitate the quick identification of the class of which an instance may belong to. Consider an example of classifying fruits as either ‘Apple’ or ‘Orange’ using two features \( F_1 \) and \( F_2 \), where \( F_1 \) is the fruit color, and \( F_2 \) is the fruit weight. Intuitively, \( F_1 \) can potentially simplify the classification process as compared to \( F_2 \). As a result, if feature \( F_2 \) was to be examined first, it would be very probable for feature \( F_1 \) to be examined as well to improve the chances of accurate classification. Instead, if \( F_1 \) was to be evaluated first, a decision could be made using one feature only. To avoid the computational complexity of evaluating all \( K! \) possible feature orderings, features are sorted in increasing order of the sum of type I and II errors.
(considering the true class as the positive class and all the rest classes as a single negative class), scaled by the cost coefficient of the \( n \)th feature to promote low cost features that at the same time are expected to result in few errors. Type I and II errors are computed using the confusion matrix, since all features are available during training for ETANA and F–ETANA. For multiclass classification datasets, we treat each class in turn as the positive class and treat the rest as the negative class. The resulting global ordering is generated once during training. During testing, features are examined sequentially one at a time based on this global ordering, and the proposed algorithms dynamically decide when to stop the feature evaluation process for each data instance individually. Finally, for ETANA, \( \eta = 10 \), and for F–ETANA, \( \rho = 10^{-5} \), and \( t_{\text{max}} = 10^5 \) are set as the stopping criteria in Algorithm 1.

### 2.6.2 Effect of Feature Space Quantization

In Section 2.6.1, the conditional probabilities of features given the class were estimated using a data binning technique (i.e., \( \hat{p}(F_k|c_i) = \frac{s_{k,i} + 1}{s_{i} + V} \)). In this subsection, the effect of the number \( V \) of bins on ETANA is analyzed using four relatively small datasets (i.e., Lung, Dexter, Madelon, MLL).

In Fig. 2.2, the variation in accuracy, the average number of features used for classification, and the training time are illustrated as a function of \( V \). ETANA’s accuracy and the average number of features used for classification is relatively robust to the number of bins except for Dexter dataset (see Fig. 2.2(a) and Fig. 2.2(b)). In the case of Dexter dataset,
increasing the number of bins from 2 to 20 results in a significant improvement in accuracy which suggests that increasing the resolution of the feature space to a high value helps to accommodate data sparsity in sparse datasets such as Dexter. On the other hand, the linear relationship between training time and the number of bins (see Fig. 2.2(c)) is due to Eq. (2.12). In the rest of the experiments, to reduce training time, $V$ is set to a low value (i.e., 4), except for sparse datasets (e.g., Dexter, Dorothea and News20), where $V$ is set to a slightly higher value (i.e., 12).

Figure 2.3: Distribution of number of features used under 4 different orderings (proposed (c.f. Section 2.6.1), worst (inverse of proposed), two random orderings), and associated accuracy.
Table 2.2: Expected number of features required to attain the same accuracy achieved by ETANA using alternative feature distributions (Section 2.6.4). “N.A.” denotes inability to achieve the given accuracy even after using all available features.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Accuracy</th>
<th>Avg. # Feat. (ETANA)</th>
<th>Exp. # Feat. (Best)</th>
<th>Exp. # Feat. (Average)</th>
<th>Exp. # Feat. (Worst)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Madelon</td>
<td>0.6217</td>
<td>4.09</td>
<td>9.60</td>
<td>539.3</td>
<td>N.A.</td>
</tr>
<tr>
<td>Lung Cancer</td>
<td>0.9890</td>
<td>2.03</td>
<td>1.66</td>
<td>17.83</td>
<td>12,400</td>
</tr>
<tr>
<td>MLL</td>
<td>1.00</td>
<td>5.07</td>
<td>4.52</td>
<td>32.21</td>
<td>373.6</td>
</tr>
<tr>
<td>Dexter</td>
<td>0.8133</td>
<td>12.80</td>
<td>5.80</td>
<td>1085.7</td>
<td>N.A.</td>
</tr>
<tr>
<td>Car</td>
<td>0.8097</td>
<td>12.90</td>
<td>10.77</td>
<td>16.20</td>
<td>48.82</td>
</tr>
<tr>
<td>Lung2</td>
<td>0.8820</td>
<td>15.59</td>
<td>7.83</td>
<td>21.21</td>
<td>127.7</td>
</tr>
<tr>
<td>Leukemia</td>
<td>0.9571</td>
<td>2.08</td>
<td>2.06</td>
<td>27.63</td>
<td>4863.8</td>
</tr>
<tr>
<td>Prostate</td>
<td>0.9310</td>
<td>3.34</td>
<td>2.95</td>
<td>28.34</td>
<td>693.0</td>
</tr>
<tr>
<td>Spambase</td>
<td>0.8467</td>
<td>7.47</td>
<td>5.22</td>
<td>24.99</td>
<td>1497.0</td>
</tr>
<tr>
<td>Dorothea</td>
<td>0.9400</td>
<td>2.89</td>
<td>2.49</td>
<td>24.81</td>
<td>48.09</td>
</tr>
</tbody>
</table>

2.6.3 Accuracy as a Function of Average Number of Features

To study the behavior of ETANA for varying values of feature evaluation cost $e$, when all features incur same cost (i.e., $e_k = e$), accuracy is measured for constant misclassification costs (i.e., $Q_{i,j} = 1\forall i \neq j, Q_{i,i} = 0, i,j \in \{1, \ldots, L\}$) and $e = \{0.1, 0.08, 0.06, 0.04, 0.02, 0.01, 0.001, 0\}$. Different $e$ values result in different number of features and levels of accuracy. Intuitively, using a small subset of the feature set leads to low accuracy, whereas performance improves as the average number of features used increases. Although the value of $e$ has not been analytically optimized, $e = 0.01$ was found to yield the best accuracy using the smallest average number of features for the majority of the datasets considered, often outperforming the state–of–the–art. Thus, from here onwards, unless specified, results are reported for this value.

In Fig. 2.3, the distribution of number of features used by ETANA to reach a classification decision is illustrated for 4 different orderings. The accuracy achieved in each case is also provided. The proposed ordering consistently achieves the best accuracy with the least number of features used. Different orderings result in different trade–offs between accuracy and number of features used. This illustrates the importance of ordering features by their informativeness during training.
2.6.4 Experiments on Expected Number of Features

To better understand the practical implications of the theoretical result in Section 2.5, $E\{R|c_i\}$ is computed for each $c_i$, $i \in \{1, .., L\}$, using Eq. (2.20) for each dataset. The expected number $E\{R\}$ of features is also computed as follows:

$$E\{R\} = \sum_{i=1}^{L} E\{R|c_i\} P(c_i). \quad (2.21)$$

In the interest of computing the expected number of features required to obtain the same accuracy achieved by ETANA, $\gamma$ is set to $(1-\text{accuracy achieved by ETANA})$. Further, three alternative feature distributions are considered: i) best: the distribution of the first feature in the ordering proposed in Section 2.6.1, ii) worst: the distribution of the last feature in the ordering proposed in Section 2.6.1, and iii) average: the “average” feature distribution found by approximating the denominator term in Eq. (2.20) using the average KL divergence among all feature distributions as follows:

$$\sum_{v=1}^{V} \theta_v \log \frac{\theta_v}{\bar{\theta}_v} \approx \frac{1}{K} \sum_{k=1}^{K} \sum_{v=1}^{V} \theta_v^k \log \frac{\theta_v^k}{\bar{\theta}_v^k}, \quad (2.22)$$

where $\theta_v^k = P(F_k = v|c_i)$, and $\bar{\theta}_v^k = P(F_k = v|\bar{c}_i)$. Finally, $V$ and $e$ are set based on the discussions in Section 2.6.2 and Section 2.6.3, respectively.

Table 2.2 illustrates the theoretical results corresponding to each alternative feature distribution described above separately, along with actual accuracy and the average number of features used by ETANA for each dataset. Intuitively, it is expected that less features must be evaluated compared to the actual number of features used by ETANA to achieve the same accuracy if all features are identically distributed with the best feature distribution. This behavior can be observed in all datasets except Madelon. This implies that ETANA achieves the same performance as if using the best feature distribution, hence, enabling us to predict in advance the required number of feature evaluations to achieve a specific accuracy. On the other hand, if all features are identically distributed with the worst feature distribution, it is expected that more features need to be evaluated. In the worst scenario, no matter how many “noisy” features are used, good accuracy may not be achievable.
Table 2.3: Comparison of accuracy. The highest accuracy, and the second highest accuracy are bolded and gray–shaded, and gray–shaded, respectively. Cells are marked with ‘−−’ if the corresponding method was unable to generate results within a cutoff time of 12 days.

### Table 2.3: Comparison of accuracy

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Madelon</td>
<td>0.6217</td>
<td>0.5180</td>
<td>0.5117</td>
<td>0.5117</td>
<td>0.5817</td>
<td>0.5417</td>
<td>0.5817</td>
<td>0.6050</td>
</tr>
<tr>
<td>Lung Cancer</td>
<td>0.9890</td>
<td>0.9890</td>
<td>0.9835</td>
<td>0.9797</td>
<td>0.9890</td>
<td>0.9890</td>
<td>0.9724</td>
<td>0.9613</td>
</tr>
<tr>
<td>MLL</td>
<td>1.00</td>
<td>0.9467</td>
<td>0.9600</td>
<td>0.9067</td>
<td>0.8667</td>
<td>0.8000</td>
<td>0.8000</td>
<td>0.9333</td>
</tr>
<tr>
<td>Dexter</td>
<td>0.8133</td>
<td>0.7967</td>
<td><strong>0.8527</strong></td>
<td>0.7375</td>
<td>0.7300</td>
<td>0.7800</td>
<td>0.7967</td>
<td>0.5990</td>
</tr>
<tr>
<td>Car</td>
<td>0.8097</td>
<td><strong>0.8274</strong></td>
<td>0.5973</td>
<td>0.7929</td>
<td>0.7982</td>
<td>0.6082</td>
<td>0.5575</td>
<td>0.6429</td>
</tr>
<tr>
<td>Lung2</td>
<td>0.8820</td>
<td>0.8918</td>
<td><strong>0.9117</strong></td>
<td>0.8717</td>
<td>0.8187</td>
<td>0.8420</td>
<td>0.8471</td>
<td>0.8820</td>
</tr>
<tr>
<td>Leukemia</td>
<td><strong>0.9571</strong></td>
<td><strong>0.9571</strong></td>
<td>0.9438</td>
<td>0.7914</td>
<td>0.9295</td>
<td>0.9295</td>
<td>0.8867</td>
<td>0.8324</td>
</tr>
<tr>
<td>Prostate</td>
<td><strong>0.9310</strong></td>
<td>0.9010</td>
<td>0.9210</td>
<td>0.8148</td>
<td>0.8910</td>
<td>0.8633</td>
<td>0.8833</td>
<td>0.9014</td>
</tr>
<tr>
<td>Spambase</td>
<td><strong>0.8467</strong></td>
<td>0.5109</td>
<td>0.7870</td>
<td><strong>0.8598</strong></td>
<td>0.8241</td>
<td>0.8011</td>
<td>0.8011</td>
<td>0.8014</td>
</tr>
<tr>
<td>Dorothea</td>
<td><strong>0.9400</strong></td>
<td>0.7114</td>
<td><strong>0.9314</strong></td>
<td>0.9314</td>
<td>0.9114</td>
<td><strong>0.9429</strong></td>
<td>0.9000</td>
<td>0.6457</td>
</tr>
<tr>
<td>News20</td>
<td>0.7352</td>
<td>0.6346</td>
<td><strong>0.7846</strong></td>
<td><strong>0.7846</strong></td>
<td><strong>0.7846</strong></td>
<td><strong>0.7846</strong></td>
<td><strong>0.7846</strong></td>
<td><strong>0.7846</strong></td>
</tr>
<tr>
<td>Avg. rank</td>
<td>1.86</td>
<td>3.77</td>
<td>4.09</td>
<td>5.64</td>
<td>4.00</td>
<td>5.32</td>
<td>6.00</td>
<td>5.32</td>
</tr>
</tbody>
</table>

Table 2.4: Comparison of average number of features used. The minimum and the second minimum average number of features used are bolded and gray–shaded, and gray–shaded accordingly. Cells are marked with ‘−−’ if the corresponding method was unable to generate results within a cutoff time of 12 days.

### Table 2.4: Comparison of average number of features used

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Madelon</td>
<td>4.09</td>
<td>55.48</td>
<td><strong>2.00</strong></td>
<td><strong>2.00</strong></td>
<td>3.00</td>
<td>3.00</td>
<td>3.00</td>
<td>4.00</td>
</tr>
<tr>
<td>Lung Cancer</td>
<td><strong>2.03</strong></td>
<td>6.56</td>
<td>37.20</td>
<td>8.40</td>
<td>52.00</td>
<td>6.80</td>
<td>4.00</td>
<td>4.60</td>
</tr>
<tr>
<td>MLL</td>
<td>5.07</td>
<td>14.69</td>
<td>11.00</td>
<td>12.00</td>
<td>28.00</td>
<td>5.00</td>
<td><strong>3.00</strong></td>
<td>7.00</td>
</tr>
<tr>
<td>Dexter</td>
<td>12.80</td>
<td>243.4</td>
<td>10.00</td>
<td>104.0</td>
<td>21.00</td>
<td>9.00</td>
<td>6.00</td>
<td><strong>1.00</strong></td>
</tr>
<tr>
<td>Car</td>
<td>12.90</td>
<td>340.20</td>
<td>6.80</td>
<td>36.00</td>
<td>41.40</td>
<td>8.40</td>
<td><strong>5.20</strong></td>
<td>24.40</td>
</tr>
<tr>
<td>Lung2</td>
<td>15.59</td>
<td>27.91</td>
<td>16.20</td>
<td>18.00</td>
<td>28.20</td>
<td>9.40</td>
<td><strong>5.80</strong></td>
<td>34.40</td>
</tr>
<tr>
<td>Leukemia</td>
<td><strong>2.08</strong></td>
<td>9.53</td>
<td>4.40</td>
<td>13.40</td>
<td>21.00</td>
<td>4.60</td>
<td>2.20</td>
<td>3.20</td>
</tr>
<tr>
<td>Prostate</td>
<td>3.34</td>
<td>10.39</td>
<td>5.80</td>
<td>40.20</td>
<td>14.00</td>
<td>3.80</td>
<td><strong>1.60</strong></td>
<td>7.00</td>
</tr>
<tr>
<td>Spambase</td>
<td><strong>7.47</strong></td>
<td>56.00</td>
<td>7.60</td>
<td>42.20</td>
<td>24.60</td>
<td>33.80</td>
<td><strong>33.80</strong></td>
<td>42.60</td>
</tr>
<tr>
<td>Dorothea</td>
<td><strong>2.89</strong></td>
<td>8.10</td>
<td>17.40</td>
<td>34.00</td>
<td>52.00</td>
<td>24.00</td>
<td>3.00</td>
<td>113.0</td>
</tr>
<tr>
<td>News20</td>
<td><strong>81.70</strong></td>
<td>4000.6</td>
<td><strong>241.8</strong></td>
<td><strong>241.8</strong></td>
<td><strong>241.8</strong></td>
<td><strong>241.8</strong></td>
<td><strong>241.8</strong></td>
<td><strong>241.8</strong></td>
</tr>
<tr>
<td>Avg. rank</td>
<td>2.64</td>
<td>6.09</td>
<td>3.95</td>
<td>5.95</td>
<td>6.00</td>
<td>3.80</td>
<td>2.41</td>
<td>5.09</td>
</tr>
</tbody>
</table>

2.6.5 Comparison with Baselines

Next, ETANA and F–ETANA are compared with 6 state–of–the–art methods: OFS–Density [75], OFS–A3M [76], SAOLA [71], OSFS [66], Fast–OSFS [66], and Alpha–Investing [74]. All such methods select a fixed common subset of features during training, that is then used during testing to classify all instances. In summary, a feature is selected if it satisfies an appropriately defined criterion (e.g., belongs in the approximated Markov blanket of the class variable [71, 66], p–statistic is greater than a dynamically varying threshold [74]), or such that the boundary region of the decision is maintained as little as possible [75, 76]. The main reason for comparing with such methods is that they have been shown to outperform standard feature selection algorithms while scaling well in high dimensional settings.

For a fair comparison, all baselines use a k–nearest neighbors (k–NN) classifier with
Table 2.5: Comparison of time (in seconds) required for feature selection (F), classification (C), joint feature selection and classification (F+C), and model training (T). The minimum and the second minimum F+C times are bolded and gray-shaded, and gray-shaded accordingly. Cells are marked with ‘- -’ if the corresponding method was unable to generate results within a cutoff time of 12 days.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Time</th>
<th>ETANA</th>
<th>F-ETANA</th>
<th>Time</th>
<th>OFS-Density</th>
<th>OFS-A3M</th>
<th>SAOLA</th>
<th>Fast–OSFS</th>
<th>OSFS</th>
<th>Alpha–Investing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Madelon</td>
<td>F+C</td>
<td>0.097</td>
<td>0.428</td>
<td>C</td>
<td>0.011</td>
<td>0.010</td>
<td>0.010</td>
<td>0.011</td>
<td></td>
<td>0.003</td>
</tr>
<tr>
<td></td>
<td>T</td>
<td>0.247</td>
<td>0.748</td>
<td></td>
<td>0.074</td>
<td>0.075</td>
<td>0.075</td>
<td>0.073</td>
<td></td>
<td>0.073</td>
</tr>
<tr>
<td>Lung Cancer</td>
<td>F+C</td>
<td>0.003</td>
<td>0.003</td>
<td>C</td>
<td>0.011</td>
<td>0.012</td>
<td>0.013</td>
<td>0.010</td>
<td>0.010</td>
<td>0.010</td>
</tr>
<tr>
<td></td>
<td>T</td>
<td>0.111</td>
<td>0.430</td>
<td></td>
<td>0.071</td>
<td>0.089</td>
<td>0.068</td>
<td>0.070</td>
<td>0.072</td>
<td>0.072</td>
</tr>
<tr>
<td>MLL</td>
<td>F+C</td>
<td>0.003</td>
<td>0.003</td>
<td>C</td>
<td>0.004</td>
<td>0.011</td>
<td>0.013</td>
<td>0.010</td>
<td>0.010</td>
<td>0.010</td>
</tr>
<tr>
<td></td>
<td>T</td>
<td>0.743</td>
<td>3.193</td>
<td></td>
<td>0.068</td>
<td>0.071</td>
<td>0.069</td>
<td>0.071</td>
<td>0.073</td>
<td>0.073</td>
</tr>
<tr>
<td>Dexter</td>
<td>F+C</td>
<td>0.125</td>
<td>0.681</td>
<td>C</td>
<td>0.048</td>
<td>0.180</td>
<td>0.060</td>
<td>0.038</td>
<td>0.033</td>
<td>0.024</td>
</tr>
<tr>
<td></td>
<td>T</td>
<td>0.218</td>
<td>2.106</td>
<td></td>
<td>0.090</td>
<td>0.073</td>
<td>0.069</td>
<td>0.067</td>
<td>0.063</td>
<td>0.089</td>
</tr>
<tr>
<td>Car</td>
<td>F+C</td>
<td>0.044</td>
<td>0.395</td>
<td>C</td>
<td>0.011</td>
<td>0.013</td>
<td>0.013</td>
<td>0.010</td>
<td>0.010</td>
<td>0.013</td>
</tr>
<tr>
<td></td>
<td>T</td>
<td>0.359</td>
<td>37.21</td>
<td></td>
<td>0.070</td>
<td>0.066</td>
<td>0.067</td>
<td>0.070</td>
<td>0.073</td>
<td>0.073</td>
</tr>
<tr>
<td>Lung2</td>
<td>F+C</td>
<td>0.052</td>
<td>0.021</td>
<td>C</td>
<td>0.012</td>
<td>0.014</td>
<td>0.013</td>
<td>0.012</td>
<td>0.010</td>
<td>0.010</td>
</tr>
<tr>
<td></td>
<td>T</td>
<td>0.782</td>
<td>6.901</td>
<td></td>
<td>0.070</td>
<td>0.067</td>
<td>0.066</td>
<td>0.072</td>
<td>0.070</td>
<td>0.070</td>
</tr>
<tr>
<td>Leukemia</td>
<td>F+C</td>
<td>0.001</td>
<td>0.002</td>
<td>C</td>
<td>0.337</td>
<td>0.011</td>
<td>0.013</td>
<td>0.010</td>
<td>0.010</td>
<td>0.011</td>
</tr>
<tr>
<td></td>
<td>T</td>
<td>0.043</td>
<td>2.162</td>
<td></td>
<td>0.270</td>
<td>0.069</td>
<td>0.068</td>
<td>0.068</td>
<td>0.070</td>
<td>0.072</td>
</tr>
<tr>
<td>Prostate</td>
<td>F+C</td>
<td>0.002</td>
<td>0.002</td>
<td>C</td>
<td>0.011</td>
<td>0.012</td>
<td>0.012</td>
<td>0.010</td>
<td>0.010</td>
<td>0.010</td>
</tr>
<tr>
<td></td>
<td>T</td>
<td>0.788</td>
<td>1.620</td>
<td></td>
<td>0.070</td>
<td>0.067</td>
<td>0.067</td>
<td>0.069</td>
<td>0.071</td>
<td>0.072</td>
</tr>
<tr>
<td>Spambase</td>
<td>F+C</td>
<td>0.237</td>
<td>0.482</td>
<td>C</td>
<td>0.016</td>
<td>0.027</td>
<td>0.031</td>
<td>0.033</td>
<td>0.033</td>
<td>0.003</td>
</tr>
<tr>
<td></td>
<td>T</td>
<td>0.285</td>
<td>0.012</td>
<td></td>
<td>0.086</td>
<td>0.006</td>
<td>0.087</td>
<td>0.067</td>
<td>0.065</td>
<td>0.069</td>
</tr>
<tr>
<td>Dorothea</td>
<td>F+C</td>
<td>0.035</td>
<td>0.033</td>
<td>C</td>
<td>0.054</td>
<td>0.110</td>
<td>0.102</td>
<td>0.074</td>
<td>0.028</td>
<td>0.016</td>
</tr>
<tr>
<td></td>
<td>T</td>
<td>0.246</td>
<td>7.208</td>
<td></td>
<td>0.000</td>
<td>0.178</td>
<td>0.069</td>
<td>0.067</td>
<td>0.070</td>
<td>0.067</td>
</tr>
<tr>
<td>News20</td>
<td>F+C</td>
<td>117.6</td>
<td>346.47</td>
<td>C</td>
<td>0.000</td>
<td>0.000</td>
<td>0.144</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td></td>
<td>T</td>
<td>3076</td>
<td>429.7</td>
<td></td>
<td>0.000</td>
<td>0.000</td>
<td>0.106</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>Avg. rank</td>
<td></td>
<td>1.86</td>
<td>2.23</td>
<td></td>
<td>6.36</td>
<td>7.73</td>
<td>3.91</td>
<td>4.27</td>
<td>6.27</td>
<td>3.36</td>
</tr>
</tbody>
</table>

Three neighbors to evaluate a selected feature subset, since it has been shown to outperform support vector machine (SVM), classification and regression tree (CART), and J48 classifiers on the datasets used in [75, 71]. At the same time, parameter α used by SAOLA, OSFS, and Fast–OSFS is set to 0.01, which has been shown to produce the best performance [71, 66]. The code for all baselines is either publicly available or has been provided by their authors. The same training and testing datasets are used by all methods. All baselines consider features sequentially during training in order to derive a common subset of features. Such feature subset is revealed simultaneously during testing to classify all instances. Instead, all features are available during training for ETANA and F–ETANA, while during testing, features are examined sequentially one at a time for each data instance. Finally, the same
metrics (i.e., accuracy, number of features used, time) used by the baselines are adopted. Next, observations drawn from Tables 2–4 are summarized below.

**Madelon:** ETANA achieves the highest accuracy using ∼4 features on average. In fact, improves accuracy by 3% over Alpha–Investing, which has the highest accuracy among all the baselines. Alpha–investing, however, is faster.

**Lung Cancer:** ETANA, F–ETANA, SAOLA and Fast–OSFS achieve the highest accuracy, but SAOLA and Fast–OSFS require 25.6, and 33 times more features respectively, compared to ETANA. Further, ETANA is much faster compared to SAOLA and Fast–OSFS.

**MLL:** ETANA achieves 100% accuracy using only 5.07 features on average. This corresponds to an improvement of 4.1% in accuracy with 53.9% less of features used compared to OFS–Density, which achieves the highest accuracy among all the baselines. At the same time, ETANA is much faster compared to OFS–Density.

**Dexter:** OFS–Density achieves the highest accuracy, resulting however in a significant slowdown compared to ETANA.

**Car:** F–ETANA achieves the highest accuracy, however, ETANA is a close second while using only 12.90 features on average. This corresponds to an improvement of 1.4% and 68.85% in accuracy and average number of features used respectively, while at the same time, leads to a faster runtime compared to SAOLA, the best performing baseline.

**Lung2:** OFS–Density achieves the highest accuracy with 3.9% more features than ETANA, and much slower runtime.

**Leukemia:** ETANA achieves the highest accuracy using the minimum number of features on average, and is the fastest in joint feature selection and classification. In fact, this corresponds to an improvement of 1.2% in accuracy with 52.7% less features compared to OFS–Density, which achieves the highest accuracy among all the baselines.

**Prostate:** ETANA achieves the highest accuracy using 3.34 features on average. In fact, ETANA achieves an improvement of 1.0% in accuracy with 42.4% less features compared to OFS–Density, which achieves the highest accuracy among all the baselines. Further, ETANA is much faster in joint feature selection and classification compared to OFS–Density.

**Spambase:** OFS–A3M achieves the highest accuracy (1.5% better than ETANA), but requires 5.6 times more features compared to ETANA, and much slower runtime.

**Dorothea:** Fast–OSFS achieves the highest accuracy, however, ETANA is a close second with only 0.3% difference in accuracy, and requires 8.3 times more features compared to
ETANA. Further, ETANA is much faster in joint feature selection and classification compared to Fast-OSFS.

Several observations can be drawn from the above results. First, among all baselines, OFS-Density achieves the highest accuracy in most datasets. The proposed methods outperform (or are closely second to) OFS-Density with respect to accuracy. Second, OSFS consistently selects the least number of features among all baselines. In comparison, ETANA either selects less or approximately similar number of features. The first result supports the importance of evaluating different features for different data instances to achieve superior overall accuracy. The second result indicates the need for larger number and/or different subsets of features for some instances (as compared to the average in the case of ETANA, or least in the case of OSFS), so as to reduce misclassification. Specifically, easy to classify data instances require few features as opposed to more challenging data instances that require more features to be accurately classified by ETANA. The behavior of F-ETANA is analyzed in detail in Section 2.6.8. Third, the runtime of ETANA and F-ETANA is consistently lower during testing than the baselines. This is a direct result of evaluating a small number of features for most of the data instances, and only explore additional features for harder to classify instances.

### 2.6.6 Scalability to High Dimensional Feature Spaces

This subsection discusses the performance of algorithms, ETANA and F-ETANA, and the baselines on the News20 dataset. Experiments on this dataset are conducted using the high performance computing cluster provided by the Information Technology Services at the University of Albany, SUNY. Specifically, a single node with 20 Intel(R) Xeon(R) E5-2680 v4 @2.40GHz CPUs with 256 GB memory was used. Except for SAOLA, the rest of the baselines were unable to generate results within a cutoff time of 12 days. Although SAOLA achieves the highest accuracy, it requires $\sim200\%$ more features and is $\sim20$ times slower compared to ETANA (see the second last row in Tables 2.3, 2.4 and 2.5). This experiment provides further supporting evidence for the benefit of dynamically selecting features and classifying each data instance individually. Specifically, fast and accurate classification decision for each data instance can be obtained using a small percentage of all features on average (i.e., $\sim0.006\%$ and $\sim0.295\%$ for ETANA and F-ETANA respectively). In addition, the proposed
methods scale to large dimensional feature spaces (i.e., more than 1.3 million features).

2.6.7 Statistical Significance

To validate the statistical significance of the results presented in Sections 2.6.5 and 2.6.6, a Friedman test is conducted. This constitutes a well-known method to compare the performance of several algorithms across multiple datasets [14]. The average ranking (avg. rank) of each method is provided in the last row in Tables 2.3–2.5. The \( p \)-values of the Friedman test on classification accuracy, the average number of features used and time required for joint feature selection and classification are \( 4.53 \times 10^{-4}, 1.62 \times 10^{-5} \) and \( 8.38 \times 10^{-18} \), respectively. Thus, there is a significant difference [14] in the performance of both ETANA and F–ETANA, and the baselines.

2.6.8 Benefits of Fast Implementation

Thus far it has been shown that ETANA performs quite well in terms of accuracy, number of features, and time required for joint feature selection and classification. The limitation of ETANA is in its training time (see Table 2.5), due to the construction of a
\((K + 1) \times d\) matrix which grows exponentially with the number of classes (see Section 2.3.3). F–ETANA drastically reduces the time required for model training as compared to ETANA without significant drop in accuracy (see Fig. 2.4). On the other hand, F–ETANA requires more features per data instance compared to ETANA (see Table 2.4).

The above observations indicate a trade–off between reducing training time and increasing the average number of features used for instance–wise classification. This is a result of the approximations adopted by F–ETANA, which depending on the structure of the feature space can lead to inaccurate estimates of the partition boundaries (see Fig. 2.1). Specifically, when the red regions corresponding to “stopping” are shrunk towards the corners of the simplex, more features are required for the proposed optimization framework to reach either one of those regions.

2.7 Concluding Remarks

In this chapter, a framework is presented for dynamic instance–wise joint feature selection and classification during testing. In an effort to minimize the number of feature evaluations per data instance for fast and accurate classification, an appropriate optimization problem was defined and solved, while useful mathematical properties of the optimum solution were derived. The proposed algorithms, ETANA and F–ETANA, perform well on a variety of public datasets, with comparable and often superior performance compared to prior work. Furthermore, F–ETANA resulted in a drastic reduction in model training time compared to ETANA.

Note that, we assume that features are conditionally independent given the class variable to simplify the derivation of the proposed framework. However, in many domains, features exhibits correlations. Further, individual features may seem irrelevant with the class when examined independently, but when combined, may improve prediction accuracy and enhance the interpretability of the final decision. We will formulate an extension that accommodates theses issues in the next chapter.
2.8 Appendix

2.8.1 Proof of Theorem 2.1

At the end of the $K$th stage, assuming that all the features have been examined, the only remaining expected cost is the optimum misclassification cost of selecting among $L$ decision choices at stage $k = K$, which is $\bar{J}_K(\pi_K) = g(\pi_K)$ (see Eq. 2.8).

Then, consider any intermediate stage $k = 0, 1, \ldots, K - 1$. Being at stage $k$, with the available information $\pi_k$, the optimum strategy has to choose between, either to terminate the feature evaluation process and incur cost $g(\pi_k)$, which is the optimum misclassification cost of selecting among $L$ decision choices (see Eq. (2.8)), or continue and incur cost of $e_{k+1}$ to evaluate feature $F_{k+1}$ and an additional cost $\bar{J}_{k+1}(\pi_{k+1})$ to continue optimally. Thus, the total cost of continuing optimally (referred to as optimum cost–to–go [8]) is $e_{k+1} + \bar{J}_{k+1}(\pi_{k+1})$.

It is important to note that at stage $k$, the outcome of examining feature $F_{k+1}$ is not known. Thus, the expected optimum cost–to–go, which is equal to $e_{k+1} + \mathbb{E}\{\bar{J}_{k+1}(\pi_{k+1})|\pi_k\}$, must be considered. Using Eq. (2.3) to express $\pi_{k+1}$ in terms of $\pi_k$, and by the definition of the expectation operator (i.e., if a random variable $Y$ has set $D$ of possible values and probability mass function $P(Y)$, then the expected value $\mathbb{E}\{H(Y)\}$ of any function $H(Y)$ equals to $\sum_{Y \in D} P(Y)H(Y)$), the optimum cost–to–go $\bar{A}_k(\pi_k)$ is:

$$\bar{A}_k(\pi_k) \triangleq e_{k+1} + \mathbb{E}\{\bar{J}_{k+1}(\pi_{k+1})|\pi_k\}$$
$$= e_{k+1} + \sum_{F_{k+1}} P(F_{k+1}|F_1, F_2, \ldots, F_k)\bar{J}_{k+1} \left( \frac{\text{diag} \left( \Delta_{k+1}(F_{k+1}) \right) \pi_k}{\Delta_{k+1}(F_{k+1}) \pi_k} \right).$$  \hspace{1cm} (2.23)

Using Bayes’ rule and the law of total probability, the term $P(F_{k+1}|F_1, F_2, \ldots, F_k)$ is simplified as follows:

$$P(F_{k+1}|F_1, \ldots, F_k) = \frac{P(F_1, \ldots, F_{k+1})}{P(F_1, \ldots, F_k)}$$
$$= \frac{\sum_{j=1}^L P(F_1, \ldots, F_{k+1}, c_j)}{\sum_{j=1}^L P(F_1, \ldots, F_k, c_j)}$$
$$= \frac{\sum_{j=1}^L P(F_1, \ldots, F_{k+1}|c_j)P(c_j)}{\sum_{j=1}^L P(F_1, \ldots, F_k|c_j)P(c_j)}. \hspace{1cm} (2.24)$$

Note that Eq. (2.24) can be further simplified by exploiting the fact that the random variables
$F_k$ are independent under each class $c_j$:

$$P(F_{k+1}|F_1, \ldots, F_k) = \frac{\sum_{j=1}^{L} P(c_j) \prod_{n=1}^{k+1} P(F_n|c_j)}{\sum_{j=1}^{L} P(c_j) \prod_{n=1}^{k} P(F_n|c_j)} = \frac{\sum_{j=1}^{L} \left( p_j \prod_{n=1}^{k} P(F_n|c_j) \right) P(F_{k+1}|c_j)}{\sum_{j=1}^{L} p_j \prod_{n=1}^{k} P(F_n|c_j)}. \quad (2.25)$$

Using the definition $\pi_k^j$, Eq. (2.25) can be simplified as follows:

$$P(F_{k+1}|F_1, \ldots, F_k) = \sum_{j=1}^{L} \pi_k^j P(F_{k+1}|c_j) = \Delta_{k+1}^T(F_{k+1}) \pi_k. \quad (2.26)$$

Finally, substituting Eq. (2.26) in Eq. (2.23), results in:

$$\bar{\mathcal{A}}_k(\pi_k) = e_{k+1} + \sum_{F_{k+1}} \Delta_{k+1}^T(F_{k+1}) \pi_k \bar{J}_{k+1}\left(\frac{\text{diag}(\Delta_{k+1}(F_{k+1})) \pi_k}{\Delta_{k+1}^T(F_{k+1}) \pi_k}\right). \quad (2.27)$$

### 2.8.2 Proof of Lemma 2.1

Consider the definition of $g(\varpi)$:

$$g(\varpi) \triangleq \min_{1 \leq j \leq L} \left[ Q_j^T \varpi \right], \varpi \in [0, 1]^L.$$

The term $Q_j^T \varpi$ is linear with respect to $\varpi$, and since the minimum of linear functions is a concave, piecewise linear function, $g(\varpi)$ is shown to be a concave, piecewise linear function as well. Concavity also assures the continuity of this function. Finally, minimization over finite $L$ hyperplanes guarantees that the function $g(\varpi)$ is made up of at most $L$ hyperplanes.

### 2.8.3 Proof of Lemma 2.2

First, let us consider the function $\bar{J}_{K-1}(\varpi)$ given by:

$$\bar{J}_{K-1}(\varpi) = \min \left[ g(\varpi), e_K + \sum_{F_K} \Delta_K^T(F_K) \varpi \bar{J}_K \left( \frac{\text{diag}(\Delta_K(F_K)) \varpi}{\Delta_K^T(F_K) \varpi} \right) \right]. \quad (2.28)$$
Using the fact that $\bar{J}_K(\pi_K) = g(\pi_K)$, Eq. (2.28) can be rewritten as follows:

$$J_{K-1}(\varpi) = \min \left[ g(\varpi), e_K + \sum_{F_K} \Delta^T_K(F_K) \varpi g \left( \frac{\operatorname{diag} (\Delta_K(F_K)) \varpi}{\Delta^T_K(F_K) \varpi} \right) \right].$$  (2.29)

Starting from the following function inside the summation of Eq. (2.29):

$$Q(\varpi) \triangleq \Delta^T_K(F_K) \varpi \min_{1 \leq j \leq L} \left[ Q^j \operatorname{diag} (\Delta_K(F_K)) \varpi \right],$$  (2.30)

and using the definition of $g(\varpi)$ in Eq. (2.13), it can be shown:

$$Q(\varpi) = \Delta^T_K(F_K) \varpi \min_{1 \leq j \leq L} \left[ Q^j \operatorname{diag} (\Delta_K(F_K)) \varpi \right]$$
$$= \min_{1 \leq j \leq L} \left[ \Delta^T_K(F_K) \varpi Q^j \operatorname{diag} (\Delta_K(F_K)) \varpi \right]$$
$$= \min_{1 \leq j \leq L} \left[ Q^j \operatorname{diag} (\Delta_K(F_K)) \varpi \right].$$  (2.31)

Note that the term $M^T_j \operatorname{diag} (\Delta_K(F_K)) \varpi$ is linear with respect to $\varpi$. Using the fact that the minimum of linear functions is a concave, piecewise linear function, implies that $Q(\varpi)$ is a concave, piecewise linear function. Furthermore, it is known that: i) the non–negative sum of concave/piecewise linear functions is also a concave/piecewise linear function, and ii) the minimum of two concave/piecewise linear functions is also a concave/piecewise linear function. Based on these two facts, and the fact that $\Delta_K(F_K)$ is a probability vector which is non–negative, the function $\bar{J}_{K-1}(\varpi)$ in Eq. (2.28) is concave and piecewise linear. Concavity also assures the continuity of this function.

Then, consider the function $\bar{J}_{K-2}(\varpi)$ given by:

$$\bar{J}_{K-2}(\varpi) = \min \left[ g(\varpi), e_{K-1} + \sum_{F_{K-1}} \Delta^T_{K-1}(F_{K-1}) \varpi \bar{J}_{K-1} \left( \frac{\operatorname{diag} (\Delta_{K-1}(F_{K-1})) \varpi}{\Delta^T_{K-1}(F_{K-1}) \varpi} \right) \right].$$  (2.32)

It has already been shown that the functions $\bar{J}_{K-1}(\varpi)$ and $g(\varpi)$ are concave and piecewise linear. Using the facts that i) the non–negative sum of concave/piecewise linear functions is also a concave/piecewise linear function, and ii) the minimum of two concave/piecewise linear functions is also a concave/piecewise linear function, it is shown that the function
\( \tilde{J}_{K-2}(\varpi) \) is also concave and piecewise linear. Concavity also assures the continuity of this function. Using similar arguments, the concavity, the continuity and the piecewise linearity of functions \( \tilde{J}_k(\varpi), n = 0, \ldots, K-3 \), can also be guaranteed.

### 2.8.4 Proof of Theorem 2.2

The proof starts by showing that all \( L \) corners of the \( L - 1 \) dimensional unit simplex always correspond to stopping irrespective of the stage. In other words, when \( \varpi = u_i \), \( g(\varpi) < \tilde{A}_k(\varpi) \), for all \( k = 0, \ldots, K-1 \), where \( u_i \) denotes the column vector with a 1 in the \( i \)th coordinate and 0’s elsewhere. At stage \( k = K-1 \), the following holds:

\[
\tilde{A}_{K-1}(\varpi) = e_K + \sum_{F_K} \Delta_T^T(F_K) u_i \tilde{J}_{K-1} \left( \frac{\text{diag}(\Delta_K(F_K)) u_i}{\Delta_T^T(F_K) u_i} \right) \\
= e_K + \sum_{F_K} \Delta_T^T(F_K) u_i g \left( \frac{\text{diag}(\Delta_K(F_K)) u_i}{\Delta_T^T(F_K) u_i} \right) \\
= e_K + \sum_{F_K} P(F_K|c_i) g \left( \frac{P(F_K|c_i) u_i}{P(F_K|c_i)} \right) \\
= e_K + g(u_i) \sum_{F_K} P(F_K|c_i) \\
= e_K + g(u_i) \\
> g(u_i), \tag{2.33}
\]

where the last inequality holds since \( e_K > 0 \). From Eq. (2.28), \( \tilde{J}_{K-1}(u_i) = g(u_i) \). Then, for \( k = K-2 \), the following holds:

\[
\tilde{A}_{K-2}(\varpi) = e_{K-1} + \sum_{F_{K-1}} \Delta_T^T(F_{K-1}) u_i \tilde{J}_{K-1} \left( \frac{\text{diag}(\Delta_{K-1}(F_{K-1})) u_i}{\Delta_T^T(F_{K-1}) u_i} \right) \\
= e_{K-1} + \sum_{F_{K-1}} P(F_{K-1}|c_i) \tilde{J}_{K-1} \left( \frac{P(F_{K-1}|c_i) u_i}{P(F_{K-1}|c_i)} \right) \\
= e_{K-1} + \tilde{J}_{K-1}(u_i) \sum_{F_{K-1}} P(F_{K-1}|c_i) \\
= e_{K-1} + g(u_i) \sum_{F_{K-1}} P(F_{K-1}|c_i) \\
= e_{K-1} + g(u_i) > g(u_i), \tag{2.34}
\]
where the last inequality holds since $e_{K-1} > 0$. Using similar arguments, the latter result can be proven for all $k = 0, \ldots, K - 3$. The rest of the proof is very intuitive. Using the facts: (i) the functions $\bar{A}_k(\varpi)$ are concave (see the proof of Lemma 2.2), and (ii) the simplex corners always correspond to stopping, it is shown that the hyperplanes of $g(\varpi)$ connected to the $L$ corners of the unit simplex can have only one intersection with each $\bar{A}_k(\varpi)$. Finally, since $g(\varpi)$ is made up of at most $L$ hyperplanes (see Lemma 2.1), at every stage $k$, there are at most $L$ threshold curves which split up the probability space of $\varpi$ (i.e., the $L - 1$ dimensional unit simplex) into areas that correspond to either continuing or stopping.

2.8.5 Proof of Theorem 2.3

Using Bayes’ rule and the law of total probability, the a posteriori probability $\pi_R^i = P(c_i|F_1, \ldots, F_R)$ of class $c_i, i \in \{1, \ldots, L\}$ at the stopping feature $R$ is:

$$\pi_R^i = P(c_i|F_1, \ldots, F_R)$$

$$= \frac{P(F_1, \ldots, F_R|c_i)P(c_i)}{P(F_1, \ldots, F_R)}$$

$$= \frac{P(F_1, \ldots, F_R, c_i) + P(F_1, \ldots, F_R, \bar{c}_i)}{P(F_1, \ldots, F_R|c_i)P(c_i)}$$

$$= \frac{P(F_1, \ldots, F_R|c_i)P(c_i)}{P(F_1, \ldots, F_R|c_i)P(c_i) + P(F_1, \ldots, F_R|\bar{c}_i)P(\bar{c}_i)}$$

$$= \frac{1}{1 - \pi_R^i}$$

$$= \frac{P(F_1, \ldots, F_R|\bar{c}_i)P(\bar{c}_i)}{P(F_1, \ldots, F_R|c_i)P(c_i)}$$

$$= \left(1 - \frac{\pi_R^i}{\pi_R^i}\right) \frac{P(c_i)}{P(\bar{c}_i)}.$$  \hspace{1cm} (2.35)
where \( \{ \bar{c}_i \} \triangleq \{ \bigcup_{j=1, j \neq i}^{L} c_j \} \). Under the i.i.d. assumption on feature distributions, Eq. (2.35) becomes:

\[
\left( \frac{1 - \pi_R^i}{\pi_R^i} \right) \frac{P(c_i)}{P(\bar{c}_i)} = \prod_{k=1}^{R} \frac{P(F_k|c_i)}{P(F_k|\bar{c}_i)}
\]

\[
\log \left( \frac{1 - \pi_R^i}{\pi_R^i} \right) \frac{P(c_i)}{P(\bar{c}_i)} = \log \left( \prod_{k=1}^{R} \frac{P(F_k|c_i)}{P(F_k|\bar{c}_i)} \right)
\]

\[
\log(1 - \pi_R^i) - \log(\pi_R^i) + \log \left( \frac{P(c_i)}{P(\bar{c}_i)} \right) = \sum_{k=1}^{R} \log \left( \frac{P(F_k|c_i)}{P(F_k|\bar{c}_i)} \right)
\]

\[
\log(1 - \pi_R^i) - \log(\pi_R^i) + \log \left( \frac{P(c_i)}{P(\bar{c}_i)} \right) = \sum_{k=1}^{R} \mathcal{X}_k,
\]

(2.36)

where \( \mathcal{X}_k \triangleq \log \left( \frac{P(F_k|c_i)}{P(F_k|\bar{c}_i)} \right) \).

At this point, Wald’s Identity for a sequence of i.i.d. random variables \( \mathcal{X}_1, \ldots, \mathcal{X}_R \) [64] states that:

\[
\mathbb{E} \left\{ \sum_{k=1}^{R} \mathcal{X}_k \right\} = \mathbb{E} \left\{ \mathbb{E} \left\{ \sum_{k=1}^{R} \mathcal{X}_k | R \right\} \right\} = \mathbb{E} \{ R \} \mathbb{E} \{ \mathcal{X}_1 \}.
\]

(2.37)

Taking the expectation operator on both sides in Eq. (2.36) and applying Wald’s Identity, it can be shown that:

\[
\mathbb{E} \left\{ \log(1 - \pi_R^i) - \log(\pi_R^i) + \log \left( \frac{P(c_i)}{P(\bar{c}_i)} \right) \right\} = \mathbb{E} \left\{ \sum_{k=1}^{R} \mathcal{X}_k \right\}
\]

\[
\mathbb{E} \{ \log(1 - \pi_R^i) \} - \mathbb{E} \{ \log(\pi_R^i) \} + \log \left( \frac{P(c_i)}{P(\bar{c}_i)} \right) = \mathbb{E} \left\{ \sum_{k=1}^{R} \mathcal{X}_k \right\}
\]

\[
= \mathbb{E} \{ R \} \mathbb{E} \{ \mathcal{X}_1 \}.
\]

(2.38)

Utilizing the conditional independence assumption [64], the following expression can be derived:

\[
\mathbb{E} \{ R | c_i \} = \frac{\mathbb{E} \{ \log(1 - \pi_R^i) | c_i \} - \mathbb{E} \{ \log(\pi_R^i) | c_i \} + \log \left( \frac{P(c_i)}{P(\bar{c}_i)} \right)}{\mathbb{E} \{ \mathcal{X}_1 | c_i \}}.
\]

(2.39)
Utilizing the fact that the optimum feature selection strategy compares $\pi_k$ with specific threshold values to decide between stopping or continuing the feature selection process (Theorem 2.2) enables the computation of $\mathbb{E}\{\log(\pi^i_R) | c_i\}$ and $\mathbb{E}\{\log(1 - \pi^i_R) | c_i\}$. To estimate the number of features evaluated until stopping, the aforementioned thresholds are approximated as shown below.

First, recall function $g(\pi_k)$, which represents the optimum cost of stopping at stage $k$:

$$g(\pi_k) = \min_{1 \leq j \leq L} \left[ M_j^T \pi_k \right], \pi_k \in [0, 1]^L. \quad (2.40)$$

Without loss of generality, consider constant misclassification costs, i.e., $M_{ij} = 1, \forall i \neq j, M_{ii} = 0, i, j \in \{1, 2, \ldots, L\}$. Substituting these values in Eq. (2.40), it is shown that:

$$g(\pi_k) \triangleq \min_{1 \leq j \leq L} \sum_{l=1, l\neq j}^{L} \pi^l_k = \min_{1 \leq j \leq L} \left[ 1 - \pi^j_k \right]. \quad (2.41)$$

Under the one–vs–rest approximation, i.e., classify $c_i$ versus $\overline{c}_i$, the modified $\tilde{g}(\pi_k)$ is given by:

$$\tilde{g}(\pi_k) \triangleq \min \left[ 1 - \pi^i_k, 1 - \sum_{j=1, j \neq i}^{L} \pi^j_k \right] = \min \left[ 1 - \pi^i_k, \pi^i_k \right]. \quad (2.42)$$

The associated modified cost $\tilde{J}_k(\pi_k)$ is:

$$\tilde{J}_k(\pi_k) = \min \left[ \tilde{g}(\pi_k), \tilde{A}_k(\pi_k) \right], \quad (2.43)$$

where

$$\tilde{A}_k(\pi_k) = c_{k+1} + \sum_{F_{k+1}} \Delta^T_{k+1}(F_{k+1}) \pi_k \tilde{J}_{k+1} \left( \frac{\text{diag} \left( \Delta_{k+1}(F_{k+1}) \pi_k \right)}{\Delta^T_{k+1}(F_{k+1}) \pi_k} \right). \quad (2.44)$$

For $k = K - 1$ in Eq. (2.43), we have:

$$\tilde{J}_{K-1}(\pi_{K-1}) = \min \left[ \tilde{g}(\pi_{K-1}), \tilde{A}_{K-1}(\pi_{K-1}) \right], \quad (2.45)$$
where

\[ \hat{A}_{K-1}(\pi_{K-1}) = e_K + \sum_{F_K} \Delta^T_K(F_K)\pi_{K-1} \hat{J}_K \left( \frac{\text{diag}(\Delta_K(F_K))\pi_{K-1}}{\Delta^T_K(F_K)\pi_{K-1}} \right). \]  

(2.46)

Using the facts that:

\[ \hat{J}_K(\pi_K) = \hat{g}(\pi_K) \]  

(2.47)

\[ e_K > 0 \]  

(2.48)

\[ P(F_K|c_j) \geq 0, \forall j, \]  

(2.49)

\[ \hat{A}_{K-1}(\pi_{K-1}) \] can be simplified as follows:

\begin{align*}
\hat{A}_{K-1}(\pi_{K-1}) &= e_K + \sum_{F_K} \Delta^T_K(F_K)\pi_{K-1} \times \hat{g} \left( \frac{\text{diag}(\Delta_K(F_K))\pi_{K-1}}{\Delta^T_K(F_K)\pi_{K-1}} \right) \\
&= e_K + \sum_{F_K} \Delta^T_K(F_K)\pi_{K-1} \min \left[ 1 - \frac{P(F_K|c_i)\pi_{K-1}^i}{\Delta^T_K(F_K)\pi_{K-1}}, \frac{P(F_K|c_i)\pi_{K-1}^i}{\Delta^T_K(F_K)\pi_{K-1}} \right] \\
&= e_K + \sum_{F_K} \min \left[ \sum_{j=1, j\neq i}^L P(F_K|c_j)\pi_{K-1}^j, P(F_K|c_i)\pi_{K-1}^i \right] \\
&\geq e_K, \quad (2.50)
\end{align*}

where the minimum value is achieved at \( \pi_{K-1}^i \in \{0, 1\} \). At this point, the feature evaluation process at stage \( K - 1 \) stops when the following condition is satisfied:

\[ \hat{g}(\pi_{K-1}) \leq \hat{A}_{K-1}(\pi_{K-1}). \]  

(2.51)

Exploiting the lower bound for \( \hat{A}_{K-1}(\pi_{K-1}) \) in Eq. (2.50), it is shown that if \( \hat{g}(\pi_{K-1}) \leq e_K \), the feature evaluation process stops at stage \( K - 1 \) with probability 1. More precisely, based on the aforementioned observation and the result in Eq. (2.41), it is straightforward to show that, if \( \pi_{K-1}^i \geq (1 - e_K) \), feature evaluation process stops and \( c_i \) is selected with probability 1. On the other hand, if \( \pi_{K-1}^i \leq e_K \), the feature evaluation process stops and \( c_i \) is selected with probability 1. Note that the result in Eq. (2.50) is valid for every \( k \). Thus, using similar arguments it is shown that, at any stage \( k \), if \( \pi_k^i \geq (1 - e_{k+1}) \), the feature evaluation process stops and \( c_i \) is selected with probability 1.
stops and $c_i$ is selected with probability 1. On the other hand, if $\pi_k^i \leq e_{k+1}$, the feature evaluation process stops and $\tau_i$ is selected with probability 1.

Now, consider the initial problem of finding $\mathbb{E}\{\log(\pi_R^i) | c_i\}$ and $\mathbb{E}\{\log(1 - \pi_R^i) | c_i\}$ in Eq. (2.39). Assuming the feature evaluation process terminated with $\pi_R^i$ hitting, equaling, either $(1 - e_{R+1})$ or $e_{R+1}$\[^4\], it is shown that:

$$
\log(\pi_R^i | c_i) = \begin{cases} 
\log e_{R+1} & \text{with probability } \gamma \\
\log(1 - e_{R+1}) & \text{with probability } 1 - \gamma,
\end{cases} \quad (2.52)
$$

where $\gamma$ is a user–selected bound on the error probability $P_e \triangleq P(C = \tau_i | c_i)$, such that $P_e \leq \gamma$. Thus:

$$
\mathbb{E}\{\log(\pi_R^i) | c_i\} = \gamma \log e_{R+1} + (1 - \gamma) \log(1 - e_{R+1}). \quad (2.53)
$$

Taking the expectation operator on both sides in Eq. (2.53) results in:

$$
\mathbb{E}\{\log(\pi_R^i) | c_i\} = \gamma \mathbb{E}\{\log e_{R+1}\} + (1 - \gamma)\mathbb{E}\{\log(1 - e_{R+1})\}. \quad (2.54)
$$

Note that, using similar arguments, it is shown that:

$$
\mathbb{E}\{\log(1 - \pi_R^i) | c_i\} = \gamma \mathbb{E}\{\log(1 - e_{R+1})\} + (1 - \gamma)\mathbb{E}\{\log e_{R+1}\}. \quad (2.55)
$$

The term $\mathbb{E}\{\log(1 - \pi_R^i) | c_i\} - \mathbb{E}\{\log(\pi_R^i) | c_i\}$ can be simplified as follows:

$$
\mathbb{E}\{\log(1 - \pi_R^i) | c_i\} - \mathbb{E}\{\log(\pi_R^i) | c_i\} \\
= \gamma \mathbb{E}\{\log(1 - e_{R+1})\} + (1 - \gamma)\mathbb{E}\{\log e_{R+1}\} - \gamma \mathbb{E}\{\log e_{R+1}\} - (1 - \gamma)\mathbb{E}\{\log(1 - e_{R+1})\} \\
= (\mathbb{E}\{\log e_{R+1}\} - \mathbb{E}\{\log(1 - e_{R+1})\}) - 2\gamma(\mathbb{E}\{\log e_{R+1}\} - \mathbb{E}\{\log(1 - e_{R+1})\}) \\
= (1 - 2\gamma)\mathbb{E}\{\log \frac{e_{R+1}}{1 - e_{R+1}}\}. \quad (2.56)
$$

To compute $\mathbb{E}\{R | c_i\}$ using Eq. (2.39), the i.i.d. assumption on feature distributions is employed so that $\mathbb{E}\{X_k | c_i\}$ can be determined. Specifically, it is assumed that all features
have identical categorical distributions \([47]\), i.e., \(P(F_k) \triangleq P(F), \forall k \in \{1, \ldots, K\}\) given by:

\[
P(F = v|c_i) = \theta_v, \text{ and } P(F = v|\bar{c}_i) = \bar{\theta}_v, v \in 1, \ldots, V, \tag{2.57}
\]

such that \(\sum_{v=1}^{V} \theta_v = 1\), and \(\sum_{v=1}^{V} \bar{\theta}_v = 1\), where \(V\) is the total number of unique assignments for \(F\). Using this distribution, \(\mathbb{E}\{X_1|c_i\}\), where \(X_1 = \log \left( \frac{P(F_1|c_i)}{P(F_1|\bar{c}_i)} \right)\), is computed as:

\[
X_1|c_i = \begin{cases} 
\log \frac{\theta_1}{\bar{\theta}_1} & \text{with probability } \theta_1 \\
\vdots & \\
\log \frac{\bar{\theta}_V}{\theta_V} & \text{with probability } \theta_V,
\end{cases} \tag{2.58}
\]

\[
\mathbb{E}\{X_1|c_i\} = \sum_{v=1}^{V} \theta_v \log \frac{\bar{\theta}_v}{\theta_v}. \tag{2.59}
\]

Finally, substituting Eq. (2.59) and Eq. (2.56) in Eq. (2.39) results in:

\[
\mathbb{E}\{R|c_i\} = \frac{(1 - 2\gamma) \mathbb{E}\left\{ \log \frac{e_{R+1}}{1-e_{R+1}} \right\} + \log \frac{P(c_i)}{P(\bar{c}_i)}}{\sum_{v=1}^{V} \theta_v \log \frac{\bar{\theta}_v}{\theta_v}}. \tag{2.60}
\]

When the cost of using each feature is the same, i.e., \(e_k = e, \forall k\), the above expression becomes:

\[
\mathbb{E}\{R|c_i\} = \frac{(1 - 2\gamma) \log \frac{e}{1-e} + \log \frac{P(c_i)}{P(\bar{c}_i)}}{\sum_{v=1}^{V} \theta_v \log \frac{\bar{\theta}_v}{\theta_v}}. \tag{2.61}
\]

Multiplying both denominator and numerator by \(-1\) results in:

\[
\mathbb{E}\{R|c_i\} = \frac{(1 - 2\gamma) \log \frac{1-e}{e} + \log \frac{P(\bar{c}_i)}{P(c_i)}}{\sum_{v=1}^{V} \theta_v \log \frac{\bar{\theta}_v}{\theta_v}}. \tag{2.62}
\]
CHAPTER 3
Dynamic Instance–Wise Classification in Correlated Feature Spaces

3.1 Introduction

This chapter extends the work from the previous chapter, which formalizes the dynamic instance–wise classification problem with the simplifying assumption that features are conditionally independent given the class variable. Specifically, herein features ensure consistency: correlation versus dependency are efficiently and effectively modeled using a Bayesian network. In general, handling feature dependencies can lead to computationally intractable solutions [45]. Therefore, a novel feature ordering is presented, such that each selected feature contains the maximum possible new information about the class variable with respect to the already evaluated feature set. The optimum solution is also derived for this more general formulation, and critical structural properties are analyzed that facilitate the design of a scalable method. The proposed method is evaluated and compared with prior work on a variety of real–world datasets.

The unique contributions of this chapter are summarized as follows:

i. The optimum stopping feature (i.e., the feature at which the sequential evaluation process terminates) and the optimum classification strategy are mathematically derived for each data instance individually without imposing any assumptions on feature dependencies.

ii. The structure of the optimum solution is theoretically analyzed.

iii. An efficient implementation of the optimum solution is introduced that considers feature correlations to guide the feature selection process.

iv. The effectiveness, generalizability, and scalability of the proposed method are evaluated using eleven publicly available datasets.
All proofs are included in the Appendix. To facilitate reproducibility, the source code of the proposed methods is available at: https://github.com/IDIASLab/IFC2F.

3.2 Related Work

In this section, the most relevant prior work on feature selection and classification is summarized.

To accommodate large or unknown feature spaces during model training, streaming feature selection methods [29] are designed to handle features arriving sequentially over time. Existing work on this area can be roughly categorized into two directions depending on the availability of prior information about the feature space [53, 74, 66, 71] or not [75, 76]. In general, various threshold–based approaches have been proposed, where a newly arriving feature is selected if a constraint is satisfied (e.g., predefined threshold [53], dynamically varying threshold [74], conditional independence via $G^2$–test [66] or Fisher’s $Z$–test [71]). In [75], features are selected if they exhibit high correlation with the class variable and low correlation with already selected features, while the boundary region of the decision is kept as small as possible. [76] extends [75] by considering the size of the neighborhood of each data instance. Motivated by real–world applications, where training instances arrive sequentially or access to the full training dataset is not feasible, [67] jointly trains a linear model and acquires a sparse representation of the feature space global to the entire dataset during the training process. All the above works use the same fixed set of features to classify all instances during testing. In sharp contrast to the above lines of work, in the proposed setting, both the training instances and the full feature space is available during the training process, and the goal is to dynamically select and use for classification a different subset of features for each instance during the testing process. As a result, each testing instance is classified using different variable features. This instance–wise property is demonstrated in Section 3.6.7.

Recent studies have shown that relevant features may differ across data instances, for example, in predicting heart failure for different patients [34]. At the same time, as complex machine learning models become more prevalent, the need to interpret their results becomes critical. Hence, instance–wise feature selection [9, 70, 68] tries to identify a small number of relevant features that explain/predict the output of a machine learning model during
testing. In [9], a single neural network is learned to identify the top $k$ features that explain a pretrained model based on mutual information. [70] adopts an actor–critic architecture with three neural networks, bypassing the need for backpropagation to outperform [9]. To avoid the high computational cost of the above methods, [68] limits the number of possible relevant feature subsets to $K$, and models this constraint with a mixture of $K$ deep neural networks. The sensitivity magnitude of the model is then used to select the most relevant features. These methods work in a **static** setting, since all feature values of a test instance must be first revealed. Such methods do not scale for large feature spaces, since the search space grows exponentially with the number of features. The method proposed herein can be used for model interpretability, and as such is related to these methods. However, unlike that line of work, the proposed method is **dynamic**, in the sense that features arrive sequentially one at a time during testing and the goal is to **jointly** select features and classify each data instance in this regime. Additionally, the number of features used for each instance is neither fixed nor predefined; it is **optimally** derived by the proposed framework. Finally, the proposed method scales well with large feature spaces being able to accommodate more than 1 million features.

Similar to this work, classification with costly features [16, 32] considers costs associated with feature evaluation and misclassification and the goal is to limit the number of features used for classification per data instance during testing. Such methods, however, define the problem **globally** with respect to the training dataset, namely, by introducing a penalization term to limit the number of features used for classification in a standard empirical loss minimization problem. In that sense, even though such approaches end up sequentially evaluating different features per instance before they classify it, the resulting classification function is **globally** learned with respect to the **dataset**. This modified problem is shown to be equivalent to a deterministic Markov decision process (MDP) formulation and solved by a linear [16] or non–linear approximation [32] of the associated Q–function. The size of the state space of the MDP grows exponentially with the dimension of the feature space, making these methods impractical for high–dimensional settings. The proposed approach in this work is conceptually different from these methods in that the problem of joint feature selection and classification is defined and solved **individually** with respect to each **data instance**. In addition, the optimum classification strategy and the optimum number of features to be used for classification for each data instance individually are mathematically
Table 3.1: Explanation of main symbols used in this chapter.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S$</td>
<td>collection of data instances</td>
</tr>
<tr>
<td>$s$</td>
<td>data instance $s \in S$</td>
</tr>
<tr>
<td>$F$</td>
<td>Feature set</td>
</tr>
<tr>
<td>$K$</td>
<td># of features</td>
</tr>
<tr>
<td>$F_k$</td>
<td>$k$th feature $F_k \in F$ $(1 \leq k \leq K)$</td>
</tr>
<tr>
<td>$f_k$</td>
<td>value of $k$th feature $F_k$ $(1 \leq k \leq K)$</td>
</tr>
<tr>
<td>$L$</td>
<td># of classes</td>
</tr>
<tr>
<td>$C$</td>
<td>class variable</td>
</tr>
<tr>
<td>$c_i$</td>
<td>class assignment to class $C$</td>
</tr>
<tr>
<td>$p_i$</td>
<td>prior probability of class $i$</td>
</tr>
<tr>
<td>$c_k$</td>
<td>Cost coefficient of $k$th feature $F_k$</td>
</tr>
<tr>
<td>$Q_{ij}$</td>
<td>Missclassification cost for classes $c_j$ and $c_j$</td>
</tr>
</tbody>
</table>

derived in the generic case of correlated features. Last but not least, key properties of the optimum solution are uncovered, thus enabling the proposed method to scale to large feature spaces.

### 3.3 Proposed Framework

In this section, the task of instance–wise supervised multi–class classification in correlated feature spaces is posed as a sequential decision process for which the optimum solution is derived. Specifically, the goal is to learn to sequentially choose a subset of features, relative to each test instance, using which each particular instance is to be assigned to one of $L$ classes. Table 3.1 summarizes the notation used hereafter.

Dependencies between features and the class variable (see Fig. 3.1) are modeled using a Bayesian network $\mathcal{B} \triangleq \{G, \Theta\}$ [50], which can be learned using methods such as [48, 35]. Specifically, a directed acyclic graph $G = (V, E)$, where $V = F \cup \{C\}$ is the set of features in $F$ augmented with variable $C$, and $E$ is the set of edges denoting correlations\(^1\) among the nodes in $V$, is given, and the set $\Theta$ of conditional probability distributions for $G$ is learned during training.

The goal is to leverage feature dependencies to train an instance–wise classifier that

\(^1\) Correlation is measured using mutual information, which quantifies the “distance” from independence between a pair of random variables [12].
allows the number of features used for classification to vary relative to each instance, so as to optimize the trade–off between accuracy and sparsity at the individual instance level. Specifically, in order to select one out of $L$ possible classes for each instance $s$, the proposed approach evaluates features sequentially by choosing the features that are: (i) highly correlated with the class variable, and (ii) conditionally independent with the already evaluated feature set. At each step, the proposed approach considers the cost of examining the remaining features to decide between continuing the process or if enough information is available for a classification decision to be reached. Herein, two random variables $R$ and $D_R$ are defined.

**Definition 3.1.** Random variable $R \in \{0, \ldots, K\}$ denotes the number of feature evaluations before the framework terminates. The event $\{R = k\}$ represents that the framework stops after evaluating $k$ features.

**Definition 3.2.** Random variable $D_R \in \{1, \ldots, L\}$ denotes the assigned value for class variable $C$ based on the information accumulated up to feature $F_R$. The event $\{D_{(R=k)} = i\}$ represents assignment of class $c_i$ using features $\{f_1, f_2, \ldots, f_k\}$.

The optimum value $R^*$ and the best class assignment $D_{R^*}^*$ for each data instance $s \in S$ are obtained by minimizing:

$$J(R, D_R) = \mathbb{E}\left\{ \sum_{k=1}^{R} e_k \right\} + \sum_{j=1}^{L} \sum_{i=1}^{L} Q_{ij} P(D_R = j, C = c_i),$$

(3.1)

where $e_k > 0$ is the feature evaluation cost representing the time and effort required to evaluate feature $F_k$, $Q_{ij} \geq 0$ is the misclassification cost of assigning class $c_j$ when class $c_i$ is true, and $P(D_R = j, C = c_i)$ is the joint probability of assigning class $c_j$ when class $c_i$ is true. Specifically, $\mathbb{E}\{\sum_{k=1}^{R} e_k\}$ is the expected cost accrued due to feature evaluations, and
the double summation corresponds to expected cost associated with the classification rule $D_R$. Thus, the optimization problem is equivalent to finding $R^*, D_R^*$, such that:

$$
\text{minimize } J(R, D_R).
$$

(3.2)

To solve Eq. (3.2), first, a subset of highly correlated features with the class variable $C$ is identified. These features are sufficient for accurately inferring $C$’s value. This is achieved by finding the Markov blanket $M_C$ of $C$ in $G$. $B_C$ denotes the induced subgraph of $G$ with nodes $V_{B_C} = M_C \cup \{C\}$. Features in $M_C$ are then sequentially evaluated so that at each step, the feature in the subset of currently unselected features, that provides the maximum additional information about $C$ with respect to the already evaluated feature set is selected. This is achieved using Eq. (3.3):

$$
\pi_k = \frac{\text{diag}\left(\Delta_k(F_k|F_1, \ldots, F_{k-1}, C)\right)\pi_{k-1}}{\Delta_k^T(F_k|F_1, \ldots, F_{k-1}, C)\pi_{k-1}},
$$

(3.3)

where $\Delta_k(F_k|F_1, \ldots, F_{k-1}, C) = [P(F_k|F_1, \ldots, F_{k-1}, c_1), \ldots, P(F_k|F_1, \ldots, F_{k-1}, c_L)]^T$ can be computed using exact inference algorithms (e.g., [35]), $\text{diag}(A)$ denotes a diagonal matrix with diagonal elements being the elements in vector $A$, $\pi_0 = [p_1, p_2, \ldots, p_L]^T$, and $\pi_k \triangleq [\pi^1_k, \pi^2_k, \ldots, \pi^L_k]^T$ is the \textit{a posteriori probability} vector with $\pi^i_k = P(c_i|F_1, \ldots, F_k)$. In lieu of $P(C = c_i|F_1 = f_1, \ldots, F_k = f_k)$, $P(c_i|F_1, \ldots, F_k)$ is used hereafter to improve readability.

For illustration purposes, suppose that $F_1$ is the feature most correlated with $C$. Let $\tilde{B}_1 \subseteq \{M_C \cup C\}$ denote the subset of $F_1$’s Markov blanket in $B_C$. $F_1$ is conditionally independent to all $F_k \in \{M_C - \tilde{B}_1\}$ given $\tilde{B}_1$. Therefore, the next feature to be evaluated should come from $\{M_C - \tilde{B}_1\}$. It follows, that the kth feature to be evaluated must belong to $\{M_C - \bigcup_{i=1}^{k-1} \tilde{B}_i\}$.

Next, $P(D_R = j, C = c_i)$ can be simplified as $P(D_R = j, c_i) = \mathbb{E}\{\pi^i_R 1_{\{D_R = j\}}\}$ by exploiting the fact that $x_R = \sum_{k=0}^K x_k 1_{\{R = k\}}$ for any sequence of random variables $\{x_k\}$,
where $A = 1$ when $A$ occurs, and $A = 0$ otherwise. Thus:

$$J(R, D_R) = \mathbb{E}\left\{ \sum_{k=1}^{R} e_k + \sum_{j=1}^{L} Q_j^T \pi_R \mathbb{I}_{\{D_R = j\}} \right\}, \quad (3.5)$$

where $Q_j \triangleq [Q_{1,j}, Q_{2,j}, \ldots, Q_{L,j}]^T$.

At this point, the optimum classification strategy $D_R^*$ can be obtained for any given $R$ by noting that

$$\sum_{j=1}^{L} Q_j^T \pi_R \mathbb{I}_{\{D_R = j\}} \geq g(\pi_R),$$

where $g(\pi_R) \triangleq \min_{1 \leq j \leq L} [Q_j^T \pi_R]$. Therefore, the optimum classification strategy $D_R^*$ for any given $R$ is:

$$D_R^* = \arg \min_{1 \leq j \leq L} [Q_j^T \pi_R], \quad (3.6)$$

which assigns the given data instance to the class yielding the minimum misclassification cost. This suggests that $J(R, D_R) \geq \min_{D_R} J(R, D_R)$, which in turn implies that the cost function in Eq. (3.5) can be reduced to:

$$\tilde{J}(R) = \mathbb{E}\left\{ \sum_{k=1}^{R} e_k + g(\pi_R) \right\}. \quad (3.7)$$

Note that Eq. (3.7) can be minimized with respect to $R \in \{0, \ldots, K\}$ in at most $K + 1$ stages using dynamic programming [8], as shown in Theorem 3.1.

**Theorem 3.1.** For $k = K - 1, \ldots, 0$, the function $J_k(\pi_k)$ is related to $\tilde{J}_{k+1}(\pi_{k+1})$ as follows:

$$\tilde{J}_k(\pi_k) = \min [g(\pi_k), A_k(\pi_k)], \quad (3.8)$$

where $A_k(\pi_k) \triangleq e_{k+1} + \sum_{F_{k+1}} \Delta_{k+1}^T (F_{k+1} | F_1, \ldots, F_k, C) \pi_k \tilde{J}_{k+1} \left( \frac{\text{diag} \left( \Delta_{k+1}^T (F_{k+1} | F_1, \ldots, F_k, C) \pi_k \right) \pi_k}{\Delta_{k+1}^T (F_{k+1} | F_1, \ldots, F_k, C) \pi_k} \right)$, and $\tilde{J}_K(\pi_K) = g(\pi_K)$. The optimum feature selection strategy is $\{F_1, F_2, \ldots, F_R^*\}$, where $R^*$ is equal to the first $k < K$ for which $g(\pi_k) \leq A_k(\pi_k)$, or $R^* = K$ if there are no more features to be evaluated.

Fig. 3.2 summarizes the process of classifying a data instance during testing. Initially, $k = 0$ and $\pi_0 = [p_1, \ldots, p_L]$. At each stage $k$, the proposed framework compares the cost $g(\pi_k)$ of stopping to the expected cost $A_k(\pi_k)$ of continuing. If $g(\pi_k) \leq A_k(\pi_k)$, the framework stops evaluating features and classifies the instance using Eq. (3.6). Otherwise, it evaluates
the next best feature $F_{k+1} = \arg\max_{X \in \{M_{c} - \bigcup_{i=1}^{k} \tilde{B}_{i}\}} MI(X; C)$ from the Bayesian network, and updates $\pi_{k}$ using Eq. (3.3). Details about the computation of mutual information, $MI(X; C)$, are provided in Section 3.5.1. These steps are repeated until the data instance is classified using a subset of or all $K$ features.

### 3.4 Theoretical Results

In this section, important properties of the optimum classification strategy $D_{R}^{*}$ in Eq. (3.6) and the optimum feature selection strategy in Eq. (3.8) are analytically derived.

Consider a general form of the function $g(\pi_{R})$ used to derive the optimum classification strategy in Eq. (3.6) given by $g(\omega) \triangleq \min_{1 \leq j \leq L} [Q_{j}^{T} \omega]$, $\omega \in [0, 1]^{L}$, where $\omega = [\omega_{1}, \ldots, \omega_{L}]^{T}$, such that $\omega_{i} \geq 0$, $\sum_{i=1}^{L} \omega_{i} = 1$. Here, the domain of $g(\omega)$ is the probability space of $\omega$, which is a $L - 1$ dimensional unit simplex. Function $g(\omega)$ has some interesting properties as described in Lemma 3.1.

**Lemma 3.1.** The function $g(\omega)$ is concave, continuous, and piecewise linear. In particular, $g(\omega)$ consists of at most $L$ hyperplanes, represented by the set $\{Q_{j}^{T}\}_{j=1}^{L}$ of $L$ vectors.

Next, consider the general form of the function $A_{k}(\pi_{R})$ in Eq. (3.8) given by $A_{k}(\omega) = \epsilon_{k+1} + \sum F_{k+1} \Delta(\omega) \bar{J}_{k+1} \left( \frac{\text{diag} \left( \frac{\Delta(F_{k+1} | F_{1}, \ldots, F_{K}, C)}{\Delta(F_{k+1} | F_{1}, \ldots, F_{K}, C)} \omega \right)}{\Delta(F_{k+1} | F_{1}, \ldots, F_{K}, C) \omega} \right)$. Lemma 3.2 summarizes the key properties of this function.
Lemma 3.2. The functions $A_k(\varpi), k = 0, \ldots, K-1$, are concave, continuous, and piecewise linear.

The properties of functions $g(\varpi)$ and $A_k(\varpi)$ stated in Lemmas 3.1 and 3.2, respectively, allow for a parsimonious representation of the function related to the optimum feature selection strategy in Eq. (3.8) as stated in Theorem 3.2.

Theorem 3.2. At every stage $k \in \{0, \ldots, K\}$, there exists a set $\{\alpha_i^k\}, \alpha_i^k \in \mathbb{R}^{1 \times L}$, of vectors such that $\bar{J}_k(\varpi) = \min_i [\alpha_i^k \varpi]$ with $\{\alpha_K^i\} \triangleq \{Q_j^T\}_{j=0}^L$.

The above properties can be used to derive an efficient algorithmic implementation of the optimum solution.

3.5 Proposed Algorithm

Theorem 3.2, Lemma 3.1 and the fact that $\bar{J}_k(\varpi) = \min [g(\varpi), A_k(\varpi)], k \in \{0, \ldots, K-1\}$, allow for an efficient implementation of the optimum feature selection strategy in Theorem 3.1. Specifically, the decision to stop or continue the feature evaluation process depends only on the vector $\alpha_T^k = \arg \min_{\alpha_i^k} [\alpha_i^k \varpi]$, such that if $\alpha_T^k \in \{Q_j^T\}_{j=0}^L$, the feature evaluation process stops, otherwise, the next feature is to be evaluated. This is based on the fact that if $\arg \min_{\alpha_i^k} [\alpha_i^k \varpi] \in \{Q_j^T\}_{j=0}^L$, it implies that $g(\varpi) \leq A_k(\varpi)$ due to the following two reasons: i) the set $\{Q_j^T\}_{j=0}^L$ of $L$ vectors represents the $L$ hyperplanes of $g(\varpi)$ (see Lemma 3.1), and ii) $\bar{J}_k(\varpi) = \min_i [\alpha_i^k \varpi] = \min [g(\varpi), A_k(\varpi)]$. Based on this fact, a dynamic Instance–wise joint Feature selection and Classification algorithm for Correlated Features (IFC$^2$F), is presented. Initially, $\varpi$ is set to $\pi_0$, and $\alpha_T^0 = \arg \min_{\alpha_i^0} [\alpha_i^0 \varpi]$ is computed. If $\alpha_T^0 \in \{Q_j^T\}_{j=0}^L$, IFC$^2$F classifies the instance under examination to the appropriate class, based on the optimum classification strategy in Eq. (3.6). Otherwise, the first feature is evaluated. IFC$^2$F repeats these steps until either it decides to classify the instance using $< K$ features, or using all $K$ features. Algorithm 3 describes these steps in detail. The input vector sets $\{\alpha_i^k\}$ can be computed using a standard point–based value iteration algorithm [33] during training. For simplicity, the Perseus algorithm [61] is used, among the several point–based value iteration algorithms in the literature [59]. Specifically, a fixed number $\beta$ (e.g., $\sim 100$ [61]) of reachable $\varpi$ vectors from each stage, marginal probability tables $\Delta(F_k|F_1, \ldots, F_{k-1}, C)$, misclassifi-
Algorithm 3 IFC²F

**Input:** Vector sets \{α_{i0}, \ldots, α_{iK-1}\}, and misclassification costs \(Q_{ij}, i, j \in \{1, \ldots, L\}\)

**Output:** Classification decision \(D\) of the instance under examination, number \(R\) of features used

Initialize \(\varpi = \pi_0\)

for \(k = 0\) to \(K\) do

  if \(k = K\) or \(\arg\min \alpha_k \varpi \in \{Q^T_j\}_{j=0}^L\) then
    Break
  else
    Obtain next feature value \(f_{k+1}\)
    Update \(\varpi\) using Eq. (3.3)
  end if
end for

Return: \(D = \arg\min_{1 \leq j \leq L} [Q^T_j \varpi]; R = k\)

...cation costs \(Q_{ij}\) and feature evaluation costs \(e_k\) were provided to the Perseus algorithm to obtain \(\{\alpha_k^i\}, k \in \{0, \ldots, K - 1\}\).

### 3.5.1 Practical Considerations

The adjusted mutual information (AMI)\(^2\) between each feature and the class variable is computed, and \(M_c\) is acquired by removing low correlation features based on a threshold \(\eta\) on AMI. Specifically, \(\eta\) is initialized to 1, and is iteratively halved until the number of filtered features is greater than zero. For simplicity, the Bayesian Network is assumed to exhibit a tree structure rooted at the class variable (experiments are performed on three alternative network structures in Section 3.6.3). Such networks can be efficiently constructed (e.g., by building the maximum spanning tree [21]) using pairwise conditional mutual information. Conditional probability tables (i.e., \(P(F_k | \Pi_{F_k})\), where \(\Pi_{F_k}\) denotes the set of parents of \(F_k\)), are estimated using a smoothed maximum likelihood estimator. Specifically, \(\hat{P}(F_a = f_a | F_b = f_b, C = c_i) = \frac{N_{a,b,i}+1}{N_{a,i}^b+V}\), where \(N_{a,b,i}\) denotes the number of samples that satisfy \(F_a = f_a\) and \(F_b = f_b\), and belong to class \(c_i\), \(N_{b,i}\) denotes the number of samples that satisfy \(F_b = f_b\), and belong to class \(c_i\), and \(V\) is the number of quantization levels considered. The \textit{a priori} probabilities are estimated as \(P(c_i) = \frac{N_i}{\sum_{i=1}^L N_i}, i = 1, \ldots, L\) where \(N_i\) is the number of instances that belong to class \(c_i\). To reduce both memory requirements and the number of computations when...

---

\(^2\)The maximum of this value represents perfect correlation between the variables, while a value around zero represents independence [52].
storing and computing marginal probability tables $\Delta(F_k|F_1, \ldots, F_{k-1}, C)$, the dependency of each variable is limited to be second–order, such that the only dependency for $F_k$ other than $C$, is its first ancestor in the set $\{F_1, \ldots, F_{k-1}\}$, except for $F_1$ which has $C$ as its only dependent.

### 3.5.2 Complexity Analysis

**Preprocessing stage:** This stage consists of three steps. First, extracting the highly correlated feature set based on a threshold on mutual information is $O(K \log K)$. Second, learning a tree–based Bayesian network with corresponding conditional probability distributions (CPDs) is $O(K^2 + KL)$ [21]. Third, computing marginal probability tables from CPDs is $O(KL)$. Thus, the complexity of the preprocessing stage is $O(K^2 + KL)$.

**Training stage:** In this stage, the optimum $\{\alpha^i_k\}$ vectors are determined. Computing the optimum $\alpha^i_k$ vectors for all $K$ stages by considering a fixed set of belief points from each stage using Perseus algorithm is $O(KLV)$ [61]. Thus, model training is $O(KLV)$.

**Testing stage:** The computational complexity of computing the minimum $\alpha^i_k \varpi$ among the set $\{\alpha^i_k\}$ is $O(L)$, since: i) the set $\{\alpha^i_k\}$ computed using the Perseus algorithm contains at most a constant number of vectors [61], and ii) the dot product between a pair of $[0,1]^L$ vectors require $2L – 1$ computations. The complexity of obtaining a new feature is $O(1)$, while updating $\varpi$ using Eq. (3.3) is $O(L)$, since a dot product between a pair of $L$–dimensional vectors must be computed. Hence, IFC$^2$F can classify an instance in $O(KL)$.

### 3.6 Experimental Evaluation

In this section, an extensive set of experiments is conducted to evaluate the performance of IFC$^2$F using 11 benchmark datasets: 6 DNA Microarray Datasets (Lung Cancer, Lung2, MLL, Car, Leukemia, Prostate) [69], 4 NIPS feature selection challenge datasets (Dexter, Madelon, Dorothea, Spambase) [1], and 1 high dimensional dataset (News20) [2]. Table 3.2 summarizes these datasets. For Madelon, MLL, Dexter and Dorothea, the originally provided training and validation sets are used, while for the remaining datasets, five–fold cross validated results are reported. All experiments are conducted on an iMac with Quad–Core Intel Core i7 @3.30 GHz CPU, 16 GB memory, and macOS Catalina.
Table 3.2: Datasets used in the experiments.

<table>
<thead>
<tr>
<th>Dataset</th>
<th># Instances</th>
<th># Features</th>
<th># Classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Madelon</td>
<td>2,000</td>
<td>500</td>
<td>2</td>
</tr>
<tr>
<td>Lung Cancer</td>
<td>181</td>
<td>12,533</td>
<td>2</td>
</tr>
<tr>
<td>MLL</td>
<td>72</td>
<td>5,848</td>
<td>3</td>
</tr>
<tr>
<td>Dexter</td>
<td>300</td>
<td>20,000</td>
<td>2</td>
</tr>
<tr>
<td>Car</td>
<td>174</td>
<td>9,182</td>
<td>11</td>
</tr>
<tr>
<td>Lung2</td>
<td>203</td>
<td>3,312</td>
<td>5</td>
</tr>
<tr>
<td>Leukemia</td>
<td>72</td>
<td>7,129</td>
<td>2</td>
</tr>
<tr>
<td>Prostate</td>
<td>102</td>
<td>6,033</td>
<td>2</td>
</tr>
<tr>
<td>Spambase</td>
<td>4601</td>
<td>57</td>
<td>2</td>
</tr>
<tr>
<td>Dorothea</td>
<td>800</td>
<td>100,000</td>
<td>2</td>
</tr>
<tr>
<td>News20</td>
<td>19,996</td>
<td>1,355,191</td>
<td>2</td>
</tr>
</tbody>
</table>

Figure 3.3: Variation of (a) accuracy, (b) average number of features, and (c) training time (sec) as a function of the number $V \in \{2, 4, 6, 8, 12, 20, 30, 40, 50, 100\}$ of bins using Lung2, Dexter, Madelon and MLL datasets.

### 3.6.1 Effect of Feature Space Quantization

In Section 3.5.1, the feature space was quantized to estimate the conditional probability tables. Herein, the effect of the number $V$ of quantization levels on IFC$^2$F is analyzed using four datasets (Lung2, Dexter, Madelon, MLL) (see Fig. 3.3). It is observed that increasing the number of bins results in a significant drop in accuracy for the MLL dataset, while for the Dexter dataset, increasing the number of bins from 2 to 20 results in an improvement in accuracy. All the datasets show a reduction in the number of features used as $V$ is increased. These observations suggest that increasing the resolution of the feature space to a very high value can cause overfitting. However, at the same time, it can help to accommodate data sparsity in sparse datasets (e.g., Dexter). On the other hand, the linear relationship between training time and the number of bins $V$ in Fig. 3.3(c) validates the $O(KLV)$ complexity.
Table 3.3: Accuracy and average number of features used by IFC²F for different feature evaluation cost values using Lung2, Dexter, Madelon and MLL datasets.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>$e = 0.1$</th>
<th>$e = 0.01$</th>
<th>$e = 0.001$</th>
<th>$e = 0.0001$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Accuracy</td>
<td>Avg. # Features</td>
<td>Accuracy</td>
<td>Avg. # Features</td>
</tr>
<tr>
<td>Madelon</td>
<td>0.6217</td>
<td>1.00</td>
<td>0.6700</td>
<td>5.03</td>
</tr>
<tr>
<td>Dexter</td>
<td>0.6467</td>
<td>1.00</td>
<td>0.8533</td>
<td>10.95</td>
</tr>
<tr>
<td>MLL</td>
<td>0.9333</td>
<td>2.20</td>
<td>1.00</td>
<td>3.40</td>
</tr>
<tr>
<td>Lung2</td>
<td>0.6846</td>
<td>0.00</td>
<td>0.8573</td>
<td>11.30</td>
</tr>
</tbody>
</table>

Table 3.4: Accuracy and average number of features used by IFC²F for different dependency structures using Lung2, Dexter, Madelon and MLL datasets.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Tree</th>
<th>Line</th>
<th>Random</th>
<th>Clique</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Accuracy</td>
<td>Avg. # Features</td>
<td>Accuracy</td>
<td>Avg. # Features</td>
</tr>
<tr>
<td>Madelon</td>
<td>0.6700</td>
<td>5.03</td>
<td>0.6150</td>
<td>4.73</td>
</tr>
<tr>
<td>Dexter</td>
<td>0.8533</td>
<td>10.95</td>
<td>0.7833</td>
<td>6.20</td>
</tr>
<tr>
<td>MLL</td>
<td>1.00</td>
<td>3.40</td>
<td>1.00</td>
<td>4.80</td>
</tr>
<tr>
<td>Lung2</td>
<td>0.8573</td>
<td>11.30</td>
<td>0.8672</td>
<td>12.57</td>
</tr>
</tbody>
</table>

of IFC²F’s training stage (see Section 3.5.2). In the rest of the experiments, to eliminate overfitting, V is set to a moderate value (i.e., 4), except for sparse datasets (i.e., Dexter, Dorothea, Spambase and News20), where V is set to a slightly higher value (i.e., 10).

3.6.2 Effect of Feature Evaluation Cost

To study the behavior of IFC²F for varying values of feature evaluation cost $e = \{0.1, 0.01, 0.001, 0.0001\}$, when all features incur the same cost (i.e., $e_k = e, \forall k$), the accuracy and the average number of features used for classification for constant misclassification costs (i.e., $Q_{ij} = 1\forall i \neq j, Q_{ii} = 0, i, j \in \{1, \ldots, L\}$) are measured (see Table 3.3). Different $e$ values result in different number of features used and levels of accuracy. Intuitively, using a small portion of the total feature set leads to low accuracy, whereas when the average number of features used increases, the performance improves dramatically. From here onwards, unless specified, results are reported for $e = 0.01$, since according to Table 3.3, IFC²F achieves the best tradeoff between accuracy and the average number of features used for this value.
3.6.3 Effect of Bayesian Network Structure

In this subsection, the behavior of IFC$^2$F is analyzed for different dependency structures by considering three alternatives in addition to the tree–based structure (“Tree”) introduced in Section 3.5.1. Initially, features are reverse sorted (highest value first) with respect to mutual information with the class variable and the following dependency structures are considered: i) “Line”: a directed line graph having edges pointing outward starting from the first feature in the ordering; ii) “Random”: a random directed acyclic graph; and iii) “Clique”: a complete directed graph. Note that for all of these dependency structures, the class variable is considered as a common parent connected to all feature nodes. Table 3.4 provides a comparison among these four dependency structures. The “Tree” structure achieves the best trade–off between accuracy and the number of features used, and additionally seems to better approximate the true structure in three out of four datasets. However, it is possible that alternative and/or simpler structures may be more appropriate in certain scenarios, as indicated by the case of Lung2, where the random structure achieves higher overall accuracy (at the expense of average number of features used). Since the true dependency structure between features is unknown, a definitive conclusion about the optimality of the “Tree” structure cannot be reached. This forms an interesting direction to be explored in future work. Henceforth, the “Tree” structure is considered in the rest of the experimental analysis.

Table 3.5: Computational complexity of baselines. Parameters $K$ and $N$ denote number of features and instances, accordingly.

<table>
<thead>
<tr>
<th>Method</th>
<th>Computational Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>OFS–Density [75]</td>
<td>$O(K^2N^2\log N)$</td>
</tr>
<tr>
<td>OFS–A3M [76]</td>
<td>$O(K^2N^2\log N)$</td>
</tr>
<tr>
<td>SAOLA [71]</td>
<td>$O(K^2)$</td>
</tr>
<tr>
<td>OSFS [66]</td>
<td>$O(K^2KK^K)$</td>
</tr>
<tr>
<td>Fast–OSFS[66]</td>
<td>$O(KK^K)$</td>
</tr>
<tr>
<td>Alpha–Investing [74]</td>
<td>$O(K^3)$</td>
</tr>
</tbody>
</table>

3.6.4 Comparison with Baselines

In this subsection, IFC$^2$F is compared with i) 2 dynamic feature selection methods: ETANA [39], F–ETANA [39], and ii) 6 streaming feature selection methods: OFS–Density [75], OFS–A3M [76], SAOLA [71], OSFS [66], Fast–OSFS[66], and Alpha–Investing [74].
The main reason for comparing with such methods is two-fold. First, both these methods are designed to handle sequentially arriving features during model training and select a global common subset of features that is used to classify all instances during testing. Table 3.5 summarizes the computational complexity of the baselines, as reported by their authors. The complexity of the proposed approach is discussed in Section 3.5.2. The main reason for comparing with such methods is two-fold. First, both these methods and the proposed algorithm are sequential (i.e., examine one feature at a time). Second, these baselines have been shown to outperform standard feature selection algorithms and
Table 3.8: Comparison of time (in seconds) required for feature selection (F), classification (C), joint feature selection and classification (F+C), model training (T) and preprocessing (P). The minimum and the second minimum F+C times are bolded and gray–shaded, and gray–shaded accordingly. Cells are marked with ‘*’ if the corresponding method was unable to generate results within a cutoff time of 12 days.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Madelon</td>
<td>F+</td>
<td>C</td>
<td>0.070</td>
<td>0.097</td>
<td>0.428</td>
<td>F</td>
<td>129.8</td>
<td>151.1</td>
<td>0.048</td>
<td>0.076</td>
<td>0.080</td>
<td><strong>0.033</strong></td>
</tr>
<tr>
<td></td>
<td>T</td>
<td>0.155</td>
<td>0.247</td>
<td>0.748</td>
<td></td>
<td>T</td>
<td>0.011</td>
<td>0.010</td>
<td>0.010</td>
<td>0.011</td>
<td>0.011</td>
<td><strong>0.010</strong></td>
</tr>
<tr>
<td></td>
<td>P</td>
<td>35.58</td>
<td>0.142</td>
<td>0.131</td>
<td></td>
<td></td>
<td>0.074</td>
<td>0.075</td>
<td>0.075</td>
<td>0.073</td>
<td>0.073</td>
<td>0.076</td>
</tr>
<tr>
<td>Lung Cancer</td>
<td>F+</td>
<td>C</td>
<td><strong>0.001</strong></td>
<td>0.003</td>
<td>0.003</td>
<td>F</td>
<td>12.80</td>
<td>40.43</td>
<td>2.465</td>
<td>1.279</td>
<td>18.23</td>
<td>0.459</td>
</tr>
<tr>
<td></td>
<td>T</td>
<td>0.213</td>
<td>11.79</td>
<td>1.349</td>
<td></td>
<td>T</td>
<td>0.011</td>
<td>0.012</td>
<td>0.013</td>
<td>0.010</td>
<td>0.010</td>
<td><strong>0.010</strong></td>
</tr>
<tr>
<td></td>
<td>P</td>
<td>146.0</td>
<td>1.806</td>
<td>1.796</td>
<td></td>
<td></td>
<td>0.071</td>
<td>0.089</td>
<td>0.068</td>
<td>0.068</td>
<td>0.070</td>
<td>0.072</td>
</tr>
<tr>
<td>MLL</td>
<td>F+</td>
<td>C</td>
<td><strong>0.001</strong></td>
<td>0.003</td>
<td>0.003</td>
<td>F</td>
<td>1.468</td>
<td>5.543</td>
<td>1.513</td>
<td>0.564</td>
<td>4.679</td>
<td><strong>0.154</strong></td>
</tr>
<tr>
<td></td>
<td>T</td>
<td>0.448</td>
<td>24.63</td>
<td>3.193</td>
<td></td>
<td>T</td>
<td>0.008</td>
<td>0.071</td>
<td>0.069</td>
<td>0.071</td>
<td>0.071</td>
<td>0.073</td>
</tr>
<tr>
<td></td>
<td>P</td>
<td>291.2</td>
<td>0.133</td>
<td>0.133</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dexter</td>
<td>F+</td>
<td>C</td>
<td><strong>0.054</strong></td>
<td>0.125</td>
<td>0.681</td>
<td>F</td>
<td>77.88</td>
<td>4854.3</td>
<td>0.747</td>
<td>1.087</td>
<td>2.509</td>
<td>12.98</td>
</tr>
<tr>
<td></td>
<td>T</td>
<td>0.290</td>
<td>22.18</td>
<td>2.106</td>
<td></td>
<td>T</td>
<td>0.090</td>
<td>0.073</td>
<td>0.069</td>
<td>0.007</td>
<td>0.063</td>
<td>0.089</td>
</tr>
<tr>
<td></td>
<td>P</td>
<td>35.91</td>
<td>0.133</td>
<td>0.135</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Car</td>
<td>F+</td>
<td>C</td>
<td><strong>0.029</strong></td>
<td>0.044</td>
<td>0.395</td>
<td>F</td>
<td>8.236</td>
<td>40.55</td>
<td>1.155</td>
<td>0.999</td>
<td>13.40</td>
<td>0.710</td>
</tr>
<tr>
<td></td>
<td>T</td>
<td>4.950</td>
<td>20.95</td>
<td>37.21</td>
<td></td>
<td>T</td>
<td>0.011</td>
<td>0.013</td>
<td>0.013</td>
<td>0.010</td>
<td>0.010</td>
<td>0.013</td>
</tr>
<tr>
<td></td>
<td>P</td>
<td>462.5</td>
<td>361.6</td>
<td>4.157</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lung2</td>
<td>F+</td>
<td>C</td>
<td><strong>0.008</strong></td>
<td>0.052</td>
<td>0.021</td>
<td>F</td>
<td>4.437</td>
<td>16.52</td>
<td>0.702</td>
<td>0.840</td>
<td>14.54</td>
<td>0.366</td>
</tr>
<tr>
<td></td>
<td>T</td>
<td>0.120</td>
<td>782.4</td>
<td>6.901</td>
<td></td>
<td>T</td>
<td>0.012</td>
<td>0.013</td>
<td>0.013</td>
<td>0.012</td>
<td>0.010</td>
<td><strong>0.013</strong></td>
</tr>
<tr>
<td></td>
<td>P</td>
<td>28.80</td>
<td>0.905</td>
<td>0.798</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Leukemia</td>
<td>F+</td>
<td>C</td>
<td><strong>0.001</strong></td>
<td>0.001</td>
<td>0.002</td>
<td>F</td>
<td>1.793</td>
<td>5.553</td>
<td>0.649</td>
<td>0.433</td>
<td>1.079</td>
<td>0.182</td>
</tr>
<tr>
<td></td>
<td>T</td>
<td>0.103</td>
<td>8.043</td>
<td>2.162</td>
<td></td>
<td>T</td>
<td>0.037</td>
<td>0.011</td>
<td>0.019</td>
<td>0.010</td>
<td>0.010</td>
<td><strong>0.011</strong></td>
</tr>
<tr>
<td></td>
<td>P</td>
<td>56.92</td>
<td>1.000</td>
<td>1.002</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Prostate</td>
<td>F+</td>
<td>C</td>
<td><strong>0.001</strong></td>
<td>0.002</td>
<td>0.002</td>
<td>F</td>
<td>2.116</td>
<td>6.010</td>
<td>0.487</td>
<td>0.362</td>
<td>0.748</td>
<td>0.167</td>
</tr>
<tr>
<td></td>
<td>T</td>
<td>0.183</td>
<td>7.088</td>
<td>1.629</td>
<td></td>
<td>T</td>
<td>0.011</td>
<td>0.012</td>
<td>0.012</td>
<td>0.010</td>
<td>0.010</td>
<td><strong>0.010</strong></td>
</tr>
<tr>
<td></td>
<td>P</td>
<td>84.57</td>
<td>0.822</td>
<td>0.813</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Spambase</td>
<td>F+</td>
<td>C</td>
<td><strong>0.066</strong></td>
<td>0.237</td>
<td>0.482</td>
<td>F</td>
<td>53.58</td>
<td>65.03</td>
<td>0.083</td>
<td>38.79</td>
<td>387.6</td>
<td><strong>0.049</strong></td>
</tr>
<tr>
<td></td>
<td>T</td>
<td>0.039</td>
<td>0.285</td>
<td>0.012</td>
<td></td>
<td>T</td>
<td>0.016</td>
<td>0.027</td>
<td>0.031</td>
<td>0.033</td>
<td>0.033</td>
<td><strong>0.038</strong></td>
</tr>
<tr>
<td></td>
<td>P</td>
<td>2.321</td>
<td>0.0283</td>
<td>0.0296</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dorothea</td>
<td>F+</td>
<td>C</td>
<td><strong>0.013</strong></td>
<td>0.035</td>
<td>0.033</td>
<td>F</td>
<td>2152.7</td>
<td>17410</td>
<td>16.99</td>
<td>53.52</td>
<td>518.5</td>
<td>217.1</td>
</tr>
<tr>
<td></td>
<td>T</td>
<td>0.213</td>
<td>204.6</td>
<td>7.208</td>
<td></td>
<td>T</td>
<td>0.051</td>
<td>0.113</td>
<td>0.102</td>
<td>0.074</td>
<td>0.027</td>
<td><strong>0.316</strong></td>
</tr>
<tr>
<td></td>
<td>P</td>
<td>743.3</td>
<td>15.06</td>
<td>15.08</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>News20</td>
<td>F+</td>
<td>C</td>
<td><strong>9.190</strong></td>
<td>117.61</td>
<td>346.47</td>
<td>F</td>
<td>--</td>
<td>--</td>
<td>144.4</td>
<td>--</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td></td>
<td>T</td>
<td>0.564</td>
<td>307.61</td>
<td>429.7</td>
<td></td>
<td>T</td>
<td>--</td>
<td>--</td>
<td>0.106</td>
<td>--</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td></td>
<td>P</td>
<td>2064.4</td>
<td>1335.2</td>
<td>1331.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Avg. rank</td>
<td></td>
<td>1.23</td>
<td>2.82</td>
<td>3.23</td>
<td></td>
<td>7.36</td>
<td>8.73</td>
<td>4.82</td>
<td>5.27</td>
<td>7.27</td>
<td>4.27</td>
<td></td>
</tr>
</tbody>
</table>

scale well in high dimensional settings. Similar to IFC²F, ETANA and F–ETANA assume all features are available during training, while during testing, features arrive sequentially one at a time for each data instance. However, ETANA and F–ETANA assume features are conditionally independent given the class variable.
For a fair comparison, all streaming feature selection methods use a $k$-nearest neighbors ($k$–NN) classifier with three neighbors to evaluate a selected feature subset, since it has been shown to outperform support vector machine (SVM), classification and regression tree (CART), and J48 classifiers on the datasets used in [75, 71]. At the same time, parameter $\alpha$ used by SAOLA, OSFS, and Fast–OSFS is set to 0.01, which has been shown to produce the best performance [71, 66]. The code for all baselines is either publicly available or has been provided by their authors. The same training and testing datasets are used by all methods. Finally, the same metrics (i.e., accuracy, number of features used, time) used by the baselines are adopted. Observations from Tables 3.6, 3.7, 3.8 are summarized next.

**Madelon:** IFC$^2$F achieves the highest accuracy. In fact, this corresponds to an improvement of 7.8% in accuracy with being 27.8% faster in joint feature selection and classification compared to ETANA, which has the second highest accuracy. ETANA, however, requires 18.7% less features on average compared to IFC$^2$F.

**Lung Cancer:** ETANA, F–ETANA, SAOLA, and Fast–OSFS achieve the highest accuracy, but require 50.4% to 3.75 $\times$ 10$^3$% more features on average and are 200% to 1.82 $\times$ 10$^6$% slower in joint feature selection and classification for a difference of 1.7% in accuracy compared to IFC$^2$F.

**MLL:** Both IFC$^2$F and ETANA achieve 100% accuracy. However, IFC$^2$F requires 32.9% less features on average and is 200% faster in joint feature selection and classification compared to ETANA.

**Dexter:** IFC$^2$F achieves the highest accuracy and is the fastest in joint feature selection and classification.

**Car:** F–ETANA achieves the highest accuracy (10.7% improvement), but requires 1.31 $\times$ 10$^3$% more features on average and is 1.26 $\times$ 10$^3$% slower in joint feature selection and classification compared to IFC$^2$F.

**Lung2:** OFS–Density achieves the highest accuracy, but requires 43.4% more features on average and is 5.55 $\times$ 10$^4$% slower in joint feature selection and classification compared to IFC$^2$F for a difference of 5.4% in accuracy.

**Leukemia:** IFC$^2$F, ETANA and F–ETANA achieve the highest accuracy. However, IFC$^2$F requires 8.7% and 80.1% less features on average compared to ETANA and F–ETANA, respectively.

**Prostate:** ETANA achieves the highest accuracy, but is 50% slower in joint feature selection
and classification compared to IFC$^2$F.

**Spambase:** OFS–A3M achieves the highest accuracy, but requires $\sim$ 9 times more features for a difference of 4.9% in accuracy and is much slower compared to IFC$^2$F.

**Dorothea:** IFC$^2$F and Fast–OSFS achieve the highest accuracy. However, Fast–OSFS requires $\sim$ 10 times more features on average and is much slower compared to IFC$^2$F.

Several observations can be drawn from the above results. For the majority of datasets, ETANA achieves the highest accuracy while IFC$^2$F is competitive achieving higher or closely second performance compared to ETANA. This result demonstrates the fact that for some datasets, the assumption of features being conditionally independent given the class variable (used in ETANA) is more appropriate than assuming that features are dependent (used in IFC$^2$F). On the other hand, IFC$^2$F requires less number of features on average than ETANA, while OSFS consistently selects the least number of features among all baselines. This observation suggests that considering feature dependencies helps to get rid of redundant features. Further, IFC$^2$F is the fastest algorithm to perform joint feature selection and classification in all the datasets compared to the baselines. This is due to the fact that it uses less number of features on average per data instance. Specifically, easy to classify data instances require few features as opposed to more challenging data instances that require more features to be accurately classified by IFC$^2$F.

### 3.6.5 Performance Assessment on a High Dimensional Dataset

In this subsection, the performance of IFC$^2$F and the baselines is discussed within the context of the News20 dataset. Except for IFC$^2$F, ETANA, F–ETANA and SAOLA, the rest of the methods were unable to generate results within a cutoff time of 12 days. Although SAOLA achieves the highest accuracy, it requires $\sim$6 times more features and is $\sim$160 times slower in joint feature selection and classification for a mere improvement of 4.6% in accuracy compared to IFC$^2$F (second last row in Tables 3.6, 3.7, 3.8). This experiment demonstrates the ability of IFC$^2$F to scale for more than 1.3 million features, and provides further supporting evidence for the advantage of dynamically selecting features and classifying each data instance individually.
3.6.6 Statistical Significance

To validate the statistical significance of the results presented in Sections 3.6.4 and 3.6.5, a Friedman test, which constitutes a well–known method to compare the performance of several algorithms across multiple datasets [14], is conducted. The average ranking (avg. rank) of each method is given in the last row in Tables 3.6–3.8. The $p$–values of the Friedman test on classification accuracy, the average number of features used and time required for joint feature selection and classification are $1.69 \times 10^{-4}$, $8.07 \times 10^{-7}$ and $4.55 \times 10^{-24}$, respectively. Thus, there is a significant difference [14] in the performance of IFC$^2$F and the baselines.
Table 3.9: Words (features) picked by IFC²F are highlighted in yellow. The true/predicted label is given at the end of each review. The second column reports features selected for each review in ascending order (Y-axis) versus feature value (X-axis).

<table>
<thead>
<tr>
<th>IMDB Review Text (True Label, Predicted Label)</th>
<th>'great'</th>
<th>'bad'</th>
</tr>
</thead>
<tbody>
<tr>
<td>I work at a movie theater and every Thursday night we have an employee screening of one movie that comes out the next day. Today it was The Guardian. I saw the trailers and the ads and never expected much from it, and in no way really did i anticipate seeing this movie. Well turns out this movie was a lot more than I would have thought. It was a great story first of all. Ashton Kutcher and Kevin Costner did amazing acting work in this film. Being a big fan of That 70’s Show I always found it hard thinking of Kutcher as anyone but Kelso despite the great acting he did in The Butterfly Effect, but after seeing this movie I think I might be able to finally look at him as a serious actor. It was also a great tribute to the unsung heroes of the U.S. Coast Guard. (positive, positive)</td>
<td><img src="image1" alt="Graph" /></td>
<td><img src="image2" alt="Graph" /></td>
</tr>
<tr>
<td>I saw this only because my 10-yr-old was bored. He and his friend hated it but of course liked being at the movies. This is the first time I’ve strongly disagreed with Ebert in many years. There is not a single thing to recommend this film. Willis is good, as always. But the story stinks, is unbelievable, there is no real story, no action, no interesting cinematic sequences, no surprises, and worst of all, the child star is A thoroughly repulsive slug guaranteed to turn off any parent who does not have a dweeby fat slob for a kid. By all means stay away and spare your child - unless you want to punish him or her. There is no excuse for such lousy directing or writing and one hopes these filmmakers will suffer accordingly. (negative, negative)</td>
<td><img src="image3" alt="Graph" /></td>
<td><img src="image4" alt="Graph" /></td>
</tr>
<tr>
<td>I felt compelled to comment on this film because it’s listed as the fourth lowest-rated sci-film of all time on the IMDb. WHAT!?!? Sure, this movie is crappy, but it’s HILARIOUS! It’s not awful on an Ed Wood level, it’s more surreal and uneven. There are some classic moments in the film. The brain surgery is gross and great and even nuttier when you consider that the film was rated PG! Gor chasing after his dolly before getting battery acid dumped on his face- “Mine! Gimmee!” Zandor Vorkoff’s speeches at the beginning of the film- ”Before Amir, Kali was but another weak nation struggling to break free from centuries of stagnant feudalism!” Angelo Rossito also has some great lines. ”No, Gor! No!” ”You want these keys, don’t you, my pretties?” It is absolutely wrong that this is the 4th lowest-rated sci-film on the IMDb because it is ENTERTAINING. No matter how bad a film is, if it still manages to be weird, quirky, unsettling, or entertaining, it has merit and doesn’t deserve to be dumped on and dismissed. I won’t defend most of Al Adamson’s films, but this one, along with Dracula VS. FRANKENSTEIN and BLOOD OF GHOSTLY HORROR, are entertaining enough to make up for their awfulness. (positive, negative)</td>
<td><img src="image5" alt="Graph" /></td>
<td><img src="image6" alt="Graph" /></td>
</tr>
<tr>
<td>Jack Lemmon was one of our great actors. His performances in Days Of Wine And Roses, The Apartment, Some Like It Hot, Missing (to name the first ones that come to mind) were all worthy of Best Actor nomination. His only win was for Save The Tiger, and that’s a shame. He gets melancholy down to a science, but never brings it into balance with the driver in his character. He actually did a similar character much better toward the end of his career in the one-note Glengarry Glen Ross. As for the movie, wonderful supporting work by Jack Gilford as Lemmon’s partner and Thayer David as an arsonist, go for naught because the rest of the script is a muddled jumble of cliched vignettes, angst, neurotic nostalgia, and pointless moralizing. Worth seeing once as a time capsule into 1970’s style experimental direction by Avildsen. (negative, positive)</td>
<td><img src="image7" alt="Graph" /></td>
<td><img src="image8" alt="Graph" /></td>
</tr>
</tbody>
</table>
Figure 3.5: Variation of (a) accuracy, and (b) average number of features as the percentage of missing features increases from 0.01% to 50% across datasets.

3.6.7 Demonstration of Instance–wise Feature Selection

Table 3.9 demonstrates the instance–wise nature of IFC²F using 4 illustrative examples from the IMDB movie reviews dataset (50,000 instances, 89,523 features, 2 classes) [44]. The IMDB dataset is selected because the raw text of reviews is available and can be directly used to interpret the classification rationale. The training and validation sets with bag–of–words features are used as provided. The Markov blanket based feature ordering is \{‘bad’, ‘great’, ‘no’, ‘best’, ‘even’, ‘plot’, ‘nothing’, ‘love’, ‘don’’t’, ‘waste’, \ldots\}. Figure 3.4 illustrates the evolution of the posterior probability distribution \(\pi_k\) as more and more features are evaluated, until the stopping condition \(g(\pi_k) \leq A_k(\pi_k)\) is satisfied. At that time, the instance is assigned to the class with the maximum posterior probability; this is a direct result of using constant misclassification costs, i.e., \(Q_{01} = Q_{10} = 1, Q_{00} = Q_{11} = 0\) (see Eq. (3.6)). Observe that the proposed framework evaluates more features to predict challenging reviews such as (c) and (d) compared to easy and straightforward reviews such as (a) and (b). In summary, IFC²F selects different features for different data instances in a dynamic setting and assigns the class label based on the observed features.
3.6.8 Robustness to Missing Features

In this subsection, the ability of IFC\textsuperscript{2}F to handle missing features is evaluated. Specifically, \( x\% \) of features are randomly removed from each test instance and the posterior probability \( \pi_k \) is kept unchanged if a feature is missing. \( x\% \) is increased from 0.01\% to 50\%, and the effect on the accuracy and the average number of features used for classification is noted (see Fig 3.5). Evidently, IFC\textsuperscript{2}F is robust for up to 10\% missing features, beyond which value, the posterior probability \( \pi_k \) may no longer represent the underlying true class value. Thus, IFC\textsuperscript{2}F can identify informative features to make accurate predictions when a small subset of features is missing.

3.7 Concluding Remarks

In this chapter, a framework to perform dynamic instance–wise joint feature selection and classification with correlated features is proposed. Specifically, feature dependencies are modeled using a Bayesian network. Based on the learned dependency network, a method is proposed to select the most informative features sequentially and reach a classification decision for each instance individually. The effectiveness and scalability of the proposed method are illustrated on various real–world datasets. The proposed method robustly performs well on all of them, with comparable and often superior performance compared to the prior art.

The proposed method selects the most informative features from the dependency graph utilizing the proposed Markov blanket–based feature ordering. However, this dependency graph is learned offline during training. Hence the ordering in which features are selected is common for all test instances. Further, we assume that the class variable is categorical, thus limited to classification problems. The next chapter addresses these limitations by deriving the optimum instance–wise feature ordering and extending the framework to predict both categorical and real–valued class variables (i.e., both classification and regression problems).
3.8 Appendix

3.8.1 Proof of Theorem 3.1

At the end of the $K$th stage, assuming that all the features have been examined, the only remaining expected cost is the optimum misclassification cost of selecting among $L$ classes, which is $ar{J}_K(\pi_k) = g(\pi_k)$.

Then, consider any intermediate stage $k = 0, 1, \ldots, K - 1$. Being at stage $k$, with available information $\pi_k$, the optimum strategy has to choose between, either to terminate and incur cost $g(\pi_k)$, which is the optimum misclassification cost of selecting among $L$ classes, or continue with the next feature $F_{k+1}$, and incur cost $e_{k+1}$ and an additional cost $\bar{J}_{k+1}(\pi_{k+1})$ to continue optimally at stage $k + 1$. Thus, the total cost of continuing optimally is $e_{k+1} + \bar{J}_{k+1}(\pi_{k+1})$. However, at stage $k$, the assignment $f_{k+1}$ of the next feature $F_{k+1}$ is not known. Thus, the expected optimum cost-to-go, which is equal to $e_{k+1} + \mathbb{E}\{\bar{J}_{k+1}(\pi_{k+1})|\pi_k\}$, needs to be considered. Using Bayes’ rule to express $\pi_{k+1}$ in terms of $\pi_k$, and by the definition of the expectation operator, the optimum cost-to-go $A_k(\pi_k)$ takes the following form:

$$A_k(\pi_k) \triangleq e_{k+1} + \mathbb{E}\{\bar{J}_{k+1}(\pi_{k+1})|\pi_k\}$$

$$= e_{k+1} + \sum_{F_{k+1}} P(F_{k+1}|F_1, \ldots, F_k) \bar{J}_{k+1} \left( \frac{\text{diag} \left( \Delta(F_{k+1}|F_1, \ldots, F_k, C) \right) \pi_k}{\Delta^T(F_{k+1}|F_1, \ldots, F_k, C) \pi_k} \right). \quad (3.9)$$

Next, the term $P(F_{k+1}|F_1, F_2, \ldots, F_k)$ must be simplified. Specifically, using the Bayes’ rule and the law of total probability, it can be shown that:

$$P(F_{k+1}|F_1, F_2, \ldots, F_k) = \frac{P(F_1, F_2, \ldots, F_{k+1})}{P(F_1, F_2, \ldots, F_k)}$$

$$= \frac{\sum_{j=1}^{L} P(F_1, \ldots, F_k C_j)}{\sum_{j=1}^{L} P(F_1, \ldots, F_k C_j)}$$

$$= \frac{\sum_{j=1}^{L} P(F_1, \ldots, F_{k+1} C_j) P(C_j)}{\sum_{j=1}^{L} P(F_1, \ldots, F_k C_j) P(C_j)}. \quad (3.10)$$
Using the chain rule, Eq. (3.10) can be simplified as follows:

\[
P(F_{k+1}|F_1, F_2, \ldots, F_k) = \frac{\sum_{j=1}^{L} P(F_1, \ldots, F_k|C_j)P(F_{k+1}|F_1, \ldots, F_k, C_j)P(C_j)}{\sum_{j=1}^{L} P(F_1, \ldots, F_k|C_j)P(C_j)}
\]

\[= \sum_{j=1}^{L} \frac{P(F_1, \ldots, F_k|C_j)P(C_j)}{\sum_{j=1}^{L} P(F_1, \ldots, F_k|C_j)P(C_j)} P(F_{k+1}|F_1, \ldots, F_k, C_j)
\]

\[= \sum_{j=1}^{L} \pi_j^k P(F_{k+1}|F_1, \ldots, F_k, C_j)
\]

\[= \Delta^T(F_{k+1}|F_1, \ldots, F_k, C)\pi_k.
\] (3.11)

Finally, substituting Eq. (3.11) in Eq. (3.9), the desired result can be acquired:

\[
\mathcal{A}_k(\pi_{k+1}) = e_{k+1} + \sum_{F_{k+1}} \Delta^T(F_{k+1}|F_1, \ldots, F_k, C)\pi_k \bar{J}_{k+1} \left( \frac{\text{diag}(\Delta(F_{k+1}|F_1, \ldots, F_k, C))\pi_k}{\Delta^T(F_{k+1}|F_1, \ldots, F_k, C)\pi_k} \right),
\] (3.12)

which completes the proof.

### 3.8.2 Proof of Lemma 3.1

Consider the definition of \(g(\varpi)\):

\[
g(\varpi) \triangleq \min_{1 \leq j \leq L} [Q_j^T \varpi], \varpi \in [0, 1]^L.
\]

The term \(Q_j^T \varpi\) is linear with respect to \(\varpi\), and since the minimum of linear functions is a concave, piecewise linear function, \(g(\varpi)\) is a concave, piecewise linear function as well. Concavity also assures the continuity of this function. Minimization over finite \(L\) hyperplanes guarantees that the function \(g(\varpi)\) is made up of at most \(L\) hyperplanes. Hence the set \(\{Q_j^T\}_{j=1}^L\) of \(L\) vectors represents those \(L\) hyperplanes.
First, consider the function $A_{K-1}(\varpi)$ given by:

$$A_{K-1}(\varpi) = e_K + \sum_{F_K} \Delta^T(F_K|F_1, \ldots, F_{K-1}, C) \varpi \bar{J}_K \left( \frac{\text{diag} \left( \Delta(F_K|F_1, \ldots, F_{K-1}, C) \right) \varpi}{\Delta^T(F_K|F_1, \ldots, F_{K-1}, C) \varpi} \right).$$

(3.13)

Using the fact that $\bar{J}_K(\pi_K) = g(\pi_K)$, Eq. (3.13) can be rewritten as follows:

$$A_{K-1}(\varpi) = e_K + \sum_{F_K} \Delta^T(F_K|F_1, \ldots, F_{K-1}, C) \varpi \times g \left( \frac{\text{diag} \left( \Delta(F_K|F_1, \ldots, F_{K-1}, C) \right) \varpi}{\Delta^T(F_K|F_1, \ldots, F_{K-1}, C) \varpi} \right).$$

(3.14)

Using the definition of $g(\varpi)$, Eq. (3.14) can be rewritten as follows:

$$A_{K-1}(\varpi) = e_K + \sum_{F_K} \Delta^T(F_K|F_1, \ldots, F_{K-1}, C) \varpi \min_{1 \leq j \leq L} \left[ Q_j^T \text{diag} \left( \Delta(F_K|F_1, \ldots, F_{K-1}, C) \right) \varpi \right].$$

(3.15)

Using the facts that $Q_j$ and $\Delta(F_K|F_1, \ldots, F_{K-1}, C)$ are non-negative vectors, Eq. (3.15) can be simplified as follows:

$$A_{K-1}(\varpi) = e_K + \sum_{F_K} \min_{1 \leq j \leq L} \left[ Q_j^T \text{diag} \left( \Delta(F_K|F_1, \ldots, F_{K-1}, C) \right) \varpi \right].$$

(3.16)

Note that the term $Q_j^T \text{diag} \left( \Delta(F_K|F_1, \ldots, F_{K-1}, C) \right) \varpi$ is linear with respect to $\varpi$. Using the facts that i) $e_K > 0$, ii) the minimum of linear functions is a concave, piecewise linear function, and iii) the non-negative sum of concave/piecewise linear functions is also a concave/piecewise linear function, implies that $A_{K-1}(\varpi)$ is a concave, piecewise linear function. Concavity also assures the continuity of this function.

Then, consider the function $A_{K-2}(\varpi)$ given by:

$$A_{K-2}(\varpi) = e_{K-1} + \sum_{F_{K-1}} \Delta^T(F_{K-1}|F_1, \ldots, F_{K-2}, C) \varpi \bar{J}_{K-1} \left( \frac{\text{diag} \left( \Delta(F_{K-1}|F_1, \ldots, F_{K-2}, C) \right) \varpi}{\Delta^T(F_{K-1}|F_1, \ldots, F_{K-2}, C) \varpi} \right).$$

(3.17)
Note that $\bar{J}_{K-1}(\bar{\omega}) = \min [g(\bar{\omega}), A_{K-1}(\bar{\omega})]$ (see Theorem 3.1). Using the facts that i) $g(\bar{\omega})$ is a concave, piecewise linear function, ii) $A_{K-1}(\bar{\omega})$ is a concave, piecewise linear function, and iii) the minimum of two concave/piecewise linear functions is also a concave/piecewise linear function, implies that $J_{K-1}(\bar{\omega})$ is also concave and piecewise linear. Furthermore, the non–negative sum of concave/piecewise linear functions is also a concave/piecewise linear function. Based on this fact and the facts that $e_{K-1} > 0$, and $\Delta(F_{K-1}|F_1, \ldots, F_{K-2}, C)$ is a non–negative vector, the function $A_{K-2}(\bar{\omega})$ is concave and piecewise linear. Concavity also assures the continuity of this function. Using similar arguments, the concavity, the continuity and the piecewise linearity of functions $A_k(\bar{\omega}), k = 0, \ldots, K - 3,$ can also be guaranteed.

3.8.4 Proof of Theorem 3.2

At the final stage, i.e., $k = K$, $J_K(\bar{\omega}) = g(\bar{\omega}) = \min_{1 \leq j \leq L} [Q_j^T \bar{\omega}]$. Hence, $\{\alpha_k^i\} \triangleq \{Q_j^T\}_{j=0}^L$. The rest of the proof is very intuitive. Using the facts that i) $g(\bar{\omega})$ and $A_k(\bar{\omega})$ are concave and piecewise linear with respect to $\omega$, ii) $J_k(\bar{\omega}) = \min [g(\bar{\omega}), A_k(\bar{\omega})], k \in \{0, \ldots, K - 1\}$ (see Theorem 3.1), and iii) the minimum of two concave/piecewise linear functions is also a concave/piecewise linear function, implies that the function $\bar{J}_k(\bar{\omega})$ is also concave and piecewise linear. Finally, since $J_k(\bar{\omega})$ is a concave and piecewise linear function defined on a probability space, it is noted that $\bar{J}_k(\bar{\omega}) = \min_i [\alpha_k^i \bar{\omega}]$, where the set $\{\alpha_k^i\}$ of vectors represents its linear pieces.
CHAPTER 4
Optimum Feature Ordering for Dynamic Instance–wise Decision–Making

4.1 Introduction

In the previous chapters, we studied the problem of instance–wise dynamic feature selection and classification for independent and correlated features, respectively. The goal was to determine the number of features needed to predict a discrete class variable when the order by which the features are reviewed is fixed and common for all instances. Herein, we remove these restrictive assumptions and consider the more general problem of predicting both discrete and continuous target variables (i.e., both classification and regression problems) and determining the order by which features must be reviewed. Thus, we derive the optimum feature ordering, the number of features, and the prediction strategy that needs to be adopted for each data instance individually when features sequentially arrive one at a time during testing. Further, we propose efficient implementations for both classification and regression problems. Experimental results show that both the order by which our framework reviews features and the number of features used for different test instances vary. In addition, less number of features on average is needed to achieve comparable performance with the state–of–the–art.

In summary, the contributions of this chapter are as follows:

i. An optimization problem is defined in terms of the feature ordering, the number of features, and the prediction rule to minimize the average costs associated with the final prediction and the feature evaluations.

ii. The optimum feature ordering, the number of features, and the prediction strategy that needs to be adopted for each data instance individually are derived separately for classification and regression settings.

iii. Efficient implementations for classification and regression problems are proposed by
analyzing the structure of the functions related to the optimum solution and approxi-
mating the target variable using a Gaussian mixture model.

iv. The proposed method’s effectiveness, generalizability, and scalability are evaluated
using several publicly available datasets.

All proofs are included in the Appendix. For reproducibility purposes, the source code of
the proposed methods is available at: https://github.com/IDIASLab/IFCO.

4.2 Related Work

In this section, the most relevant prior work on feature selection and instance–wise
decision–making are summarized.

The majority of the state–of–the–art feature selection methods [23] are offline because
they select a global subset of features representing an entire dataset assuming all the features
are available in advance. Incorporating L1–norm prior as a regularizer to a standard empir-
ical risk minimizer is the most popular method due to its applicability in both classification
and regression problems. However, in real–world applications where features are not freely
available to acquire and incur much processing time for feature extraction, e.g., medical
tests, batch processing is impractical and infeasible. Instead, streaming feature selection
methods [53, 74, 66, 71, 75, 29, 76] select a global subset of relevant features to represent
an entire dataset when features arrive sequentially during model training. Existing work on
this area divides roughly into two directions depending on the availability of prior informa-
tion about the feature space [74, 66, 71] or not [75]. In general, various threshold–based
approaches have been proposed, where a newly arriving feature is selected if a constraint
is satisfied (e.g., predefined threshold [53], dynamically varying threshold [74], conditional
independence via $G^2$–test [66] or Fisher’s Z–test [71]). Only a handful of these methods
(e.g., [74]) are designed to handle both classification and regression tasks.

Recent studies have shown that the relative importance of each feature may vary across
data instances [9]. For example, relevant features for predicting heart failure may differ
across patient subgroups [34]. At the same time, as complex models of machine learning
become prevalent, the need to interpret their decisions becomes critical. Herein, instance–
wise feature selection methods [9, 70, 68] identify a few relevant features that explain/predict
the output of a machine learning model during testing. These methods can handle both regression and classification tasks, but require revealing all feature values of a test instance to perform feature selection. Further, such methods do not scale for large feature spaces since the search space grows exponentially with the number of features. Dynamic instance-wise feature selection methods [16, 31, 32], instead sequentially select the relevant features and predict each data instance during testing. Classification with costly features [16, 31, 32] considers costs associated with feature evaluation and misclassification, and the goal is to limit the number of features used for classification per data instance during testing. These methods formulate the problem utilizing a deterministic Markov decision process (MDP) and solve it using a linear [16] or a non-linear approximation [31, 32] of the associated Q–function. MDP’s state space grows exponentially with the dimension of the feature space, making these methods impractical for high-dimensional settings. Further, these methods are also limited to classification tasks.

### 4.3 Problem Description

In a standard supervised machine learning setting, the problem of interest is accurately predicting a real valued target variable \( C \) by evaluating the best feature subset per data instance from a set \( F \in \{F_1, \ldots, F_K\} \) of \( K \) features. Each feature \( F_k \in F \) is evaluated at a cost \( e(F_k) \). At each time, one of the following two actions are executed: i) **stop** and optimally predict the current data instance based on the reviewed features, or ii) **continue** and select the next feature from the remaining set of available features. Herein, three random variables, \( \sigma, \sigma(R), D_{\sigma(R)} \), are defined as follows.

**Definition 4.1.** Random variable \( \sigma \) denotes a feature ordering, a permutation of the features in the set \( F \). For instance, if \( K = 3 \), \( \sigma \in \{(F_1, F_2, F_3), (F_1, F_3, F_2), (F_2, F_1, F_3), (F_2, F_3, F_1), (F_3, F_1, F_2), (F_3, F_2, F_1)\} \).

**Definition 4.2.** Random variable \( \sigma(R) \) represents the number of features evaluated before the framework terminates, assuming feature ordering \( \sigma \). The event \( \{\sigma(R = k)\} \) associated with \( \sigma(R) \) means that the framework stops after evaluating \( k \) features considering feature ordering \( \sigma \). For instance, if \( K = 3 \), \( \sigma = (F_3, F_1, F_2) \), then \( \sigma(R = 2) \) denotes that the framework stops after evaluating feature \( F_1 \) to make a prediction.
Definition 4.3. Random variable $D_{\sigma(R)}$ denotes the prediction rule of the target variable $C$. Specifically, the event $\{D_{\sigma(R=k)} = \hat{c}\}$ associated with $D_{\sigma(R)}$ represents predicting $\hat{c}$ for target variable $C$ using information $\{f_{\sigma(1)}, f_{\sigma(2)}, \ldots, f_{\sigma(k)}\}$. For instance, if $K = 3$, $\sigma = (F_3, F_1, F_2)$, then the event $\{D_{\sigma(R=2)} = \hat{c}\}$ represents predicting $\hat{c}$ as the target variable $C$ based on the feature assignments $\{f_3, f_1\}$.

The optimization function in Eq. (4.1) is proposed to find the optimum feature ordering $\sigma^*$, the optimum number $\sigma^*(R^*)$ of features and the best prediction $D^*_{\sigma^*(R^*)}$ for each data instance.

$$\underset{\sigma, \sigma(R), D_{\sigma(R)}}{\text{minimize}} \ J(\sigma, \sigma(R), D_{\sigma(R)}),$$

$$J(\sigma, \sigma(R), D_{\sigma(R)}) = \mathbb{E}\left\{\sum_{k=1}^{R} e(F_{\sigma(k)})\right\} + L(D_{\sigma(R)}),$$

(4.1)

where $\mathbb{E}\left\{\sum_{k=1}^{R} e(F_{\sigma(k)})\right\}$ is the expected cost accrued due to feature evaluations, and function $L(D_{\sigma(R)})$ represents the average cost associated with the prediction rule $D_{\sigma(R)}$. Equivalently, the first term represents the cost of reviewing features, and the second term penalizes the error of the prediction rule.

4.4 Classification

In this section, $C$ is considered to be a discrete variable with prior probability distribution $P(C = c_i) = p_i, i = 1, \ldots, L$. The following cost function $L(D_{\sigma(R)})^3$ is considered:

$$L(D_{\sigma(R)}) = \sum_{j=1}^{L} \sum_{i=1}^{L} Q_{ij} P(D_{\sigma(R)} = j, C = c_i),$$

(4.2)

where $Q_{ij} \geq 0, i, j \in \{1, \ldots, L\}$ are misclassification cost coefficients representing the cost of predicting $c_j$ when the true target is $c_i$, and $P(D_{\sigma(R)} = j, C = c_i)$ is the joint probability of predicting target $c_j$ when target $c_i$ is true. Using the fact that $x_R = \sum_{k=0}^{K} x_k 1_{\{R=k\}}$ for any sequence of random variables $\{x_k\}$, where $1_A$ is the indicator function for event $A$, the

---

3For ease of derivation, the cost function in Eq. (4.2) is used. However, other cost functions typically adopted for classification can be used instead.
be the remaining set of features. Theorem 4.1 summarizes the optimum solution found by

\[ P(D_{\sigma(R)} = j, C = c_i) = \mathbb{E} \left\{ \pi_{\sigma(R)}^i \mathbb{1}_{\{D_{\sigma(R)} = j\}} \right\}, \tag{4.3} \]

where \( \pi_{\sigma(k)}^i \triangleq P(C = c_i | F_{\sigma(1)}, \ldots, F_{\sigma(k)}) \) is the posterior probability of \( \{C = c_i\} \) given the accumulated information until \( k \)th feature for ordering \( \sigma \). Using the result in Eq. (4.3), the cost function \( \mathcal{L}(D_{\sigma(R)}) \) in Eq. (4.2) takes the following form:

\[ \mathcal{L}(D_{\sigma(R)}) = \mathbb{E} \left\{ \sum_{j=1}^{L} Q_j^T \pi_{\sigma(R)} \mathbb{1}_{\{D_{\sigma(R)} = j\}} \right\}, \tag{4.4} \]

where \( Q_j \triangleq [Q_{1,j}, Q_{2,j}, \ldots, Q_{L,j}]^T \) and \( \pi_{\sigma(k)} \triangleq [\pi_{\sigma(k)}^1, \pi_{\sigma(k)}^2, \ldots, \pi_{\sigma(k)}^L]^T \). The posterior probability vector \( \pi_{\sigma(k)} \in [0, 1]^L \) can be updated recursively via Bayes’ rule as follows:

\[ \pi_{\sigma(k)} = \frac{\text{diag} \left( \Delta(F_{\sigma(k)} | F_{\sigma(1)}, \ldots, F_{\sigma(k-1)}, C) \right) \pi_{\sigma(k-1)}}{\Delta^T(F_{\sigma(k)} | F_{\sigma(1)}, \ldots, F_{\sigma(k-1)}, C) \pi_{\sigma(k-1)}}, \tag{4.5} \]

where \( \Delta(F_{\sigma(k)} | F_{\sigma(1)}, \ldots, F_{\sigma(k-1)}, C) \triangleq [P(F_{\sigma(k)} | F_{\sigma(1)}, \ldots, F_{\sigma(k-1)}, c_1), \ldots, P(F_{\sigma(k)} | F_{\sigma(1)}, \ldots, F_{\sigma(k-1)}, c_L)]^T \), \( \text{diag}(A) \) represents a diagonal matrix with elements of vector \( A \), and \( \pi_{\sigma(0)} \triangleq [\pi_{1}, \pi_{2}, \ldots, \pi_{L}]^T \). Note that \( \sum_{j=1}^{L} Q_j^T \pi_{\sigma(R)} \mathbb{1}_{\{D_{\sigma(R)} = j\}} \geq g(\pi_{\sigma(R)}) \), where \( g(\pi_{\sigma(R)}) \triangleq \min_{1 \leq j < L} \{Q_j^T \pi_{\sigma(R)}\} \). Thus, the optimum classification strategy \( D_{\sigma(R)}^* \) for any number \( \sigma(R) \) and ordering \( \sigma \), is:

\[ D_{\sigma(R)}^* = \arg \min_{1 \leq j < L} \{Q_j^T \pi_{\sigma(R)}\}. \tag{4.6} \]

Since \( J(\sigma, \sigma(R), D_{\sigma(R)}) \geq J(\sigma, \sigma(R), D_{\sigma(R)}^*) \), where \( J(\sigma, \sigma(R), D_{\sigma(R)}^*) = \min_{D_{\sigma(R)}} [J(\sigma, \sigma(R), D_{\sigma(R)})] \), the optimization function in Eq. (4.1) can be rewritten to depend only on feature ordering \( \sigma \) and the number \( \sigma(R) \) of features as follows:

\[ \tilde{J}(\sigma, \sigma(R)) = \mathbb{E} \left\{ \sum_{k=1}^{R} c(F_{\sigma(k)}) + g(\pi_{\sigma(R)}) \right\}. \tag{4.7} \]

Finally, the optimization problem \( \min_{\sigma, \sigma(R)} [\tilde{J}(\sigma, \sigma(R))] \) is solved to obtain both the optimum ordering \( \sigma^* \) and the optimum number \( \sigma^*(R^*) \) of features simultaneously. At stage \( k \), let \( N_k = \{F_{\sigma^*(1)}, \ldots, F_{\sigma^*(k)}\}, 0 \leq k \leq K \), be the \( k \) features reviewed thus far, and \( Z_k = F - N_k \) be the remaining set of features. Theorem 4.1 summarizes the optimum solution found by
Theorem 4.1. For \( k = K - 1, \ldots, 0 \), the function \( \hat{J}_k(\pi_{\sigma^*(k)}) \) is related to \( \hat{J}_{k+1}(\pi_{\sigma^*(k+1)}) \) through the equation:

\[
\hat{J}_k(\pi_{\sigma^*(k)}) = \min \{ g(\pi_{\sigma^*(k)}), \hat{A}_k(\pi_{\sigma^*(k)}) \} \quad \text{with} \\
\hat{J}_K(\pi_{\sigma^*(K)}) = g(\pi_{\sigma^*(K)}), \tag{4.8}
\]

where \( \hat{A}_k(\pi_{\sigma^*(k)}) \triangleq \min_{F_{k+1} \in Z_k} \left[ e(F_{k+1}) + \sum_{F_{k+1}} \Delta^T(F_{k+1}|F_{\sigma^*(1)}, \ldots, F_{\sigma^*(k)}, C)\pi_{\sigma^*(k)} \right] \).

The optimum number \( \sigma^*(R^*) \) of features is equal to the first \( k < K \) for which \( g(\pi_{\sigma^*(k)}) \leq \hat{A}_k(\pi_{\sigma^*(k)}) \), or \( \sigma^*(R^* = K) \) if there are no more features to be reviewed. Equivalently, the feature review process stops at stage \( k \) if the cost of stopping \( g(\pi_{\sigma^*(k)}) \) is no greater than the minimum expected cost of continuing \( \hat{A}_k(\pi_{\sigma^*(k)}) \), given all information accumulated up to stage \( k \). Further, the optimum feature \( F_{\sigma^*(k+1)} \) is obtained from the set \( Z_k \) of remaining features such that the total cost until termination is minimized. Hence, the optimum feature ordering \( \sigma^* \) derived from Theorem 4.1 is not a global ordering common for all data instances but rather varies based on the assigned feature values \( \{f_1, \ldots, f_K\} \) for each data instance.

4.4.1 Theoretical Results

In this section, properties of the optimum solution in Theorem 4.1 are analyzed. Consider a general form of the function \( g(\pi_{\sigma^*(k)}) \), used to derive the optimum classification strategy in Eq. (4.6), given by \( g(\varpi) \triangleq \min_{1 \leq j \leq L} [Q^T_j \varpi], \varpi \in [0, 1]^L \), where \( \varpi \triangleq [\omega_1, \ldots, \omega_L]^T \), such that \( \omega_i \in [0, 1], \sum_{i=1}^L \omega_i = 1 \). Function \( g(\varpi) \) has unique properties, as described in Lemma 4.1.

Lemma 4.1. Function \( g(\varpi) \) is continuous, concave, and piecewise linear, and consists of at most \( L \) hyperplanes represented by the set \( \{Q^T_j\}_{j=1}^L \) of \( L \) vectors.
Next, consider the general form of the function $\hat{A}_k(\pi^{*}(k))$ in Eq. (4.8) given by:

$$
\hat{A}_k(\omega) = \min_{F_{k+1} \in Z_k} \left[ e(F_{k+1}) + \sum_{F_{k+1}} \Delta^T(F_{k+1}|F_{\sigma^*(1)}, \ldots, F_{\sigma^*(k)}, C) \right. \\
\times \left. \omega \tilde{J}_{k+1}\left( \frac{\text{diag} \left( \Delta(F_{k+1}|F_{\sigma^*(1)}, \ldots, F_{\sigma^*(k)}, C) \right) \omega}{\Delta^T(F_{k+1}|F_{\sigma^*(1)}, \ldots, F_{\sigma^*(k)}, C) \omega} \right) \right].
$$

(4.9)

Lemma 4.2 summarizes the key properties of this function.

**Lemma 4.2.** The functions $\hat{A}_k(\omega), k = 0, \ldots, K - 1,$ are continuous, concave, and piecewise linear. In particular, $\hat{A}_k(\omega)$ takes the form $\hat{A}_k(\omega) \triangleq \min_{F_{k+1} \in Z_k} [\beta_k^{F_{k+1}} \omega]$, where $F_{\sigma^*(k+1)} \triangleq \arg\min_{F_{k+1} \in Z_k} [\beta_k^{F_{k+1}} \omega]$, and $\beta_k^{F_{k+1}} \in \mathbb{R}^{1 \times L}$.

Using the properties of functions $g(\omega)$ and $\hat{A}_k(\omega)$ stated in Lemmas 4.1 and 4.2, respectively, the function related to the optimum feature selection strategy in Eq. (4.8) can be represented compactly as shown in Theorem 4.2.

**Theorem 4.2.** At every stage $k \in \{0, \ldots, K\}$, there exists a finite set $\{\alpha_k^i\}, \alpha_k^i \in \mathbb{R}^{1 \times L}$, of vectors such that $\tilde{J}_k(\omega) = \min \{\alpha_k^i \omega\}$, where $\{\alpha_k^i\} \triangleq \{\beta_k^{F_{\sigma^*(k+1)}} \cup \{Q_j^T\}_{j=1}^L\}, k \in \{0, \ldots, K - 1\}$ with $\{\alpha_k^i\} \triangleq \{Q_j^T\}_{j=1}^L$.

Note that, functions $\tilde{J}_k(\omega), k = 0, \ldots, K,$ decompose into linear hyperplanes represented by the vectors in the set $\{\alpha_k^i\}$. Hence, the finite set $\{\alpha_k^i\}$ can be used as a compact representation of $\tilde{J}_k(\omega)$ instead of computing $\tilde{J}_k(\omega)$ at every realization of $\omega$. Section 4.4.2 uses this property to derive an efficient algorithmic implementation of the optimum solution.

### 4.4.2 IFCO Algorithm

Lemma 4.1, Lemma 4.2 and Theorem 4.2 allow for a fast implementation of the optimum feature selection strategy in Theorem 4.1. Specifically, the decision to stop or continue the feature review process depends only on $\alpha^*_k = \arg\min_{\alpha_k^i} [\alpha_k^i \omega]$, such that if $\alpha^*_k \in \{Q_j^T\}_{j=1}^L$, the feature review process stops. Otherwise, the feature review process continues with feature $F_{\sigma^*(k+1)}$ associated with $\alpha^*_k \in \{\beta_k^{F_{\sigma^*(k+1)}}\}$. This is due to $\tilde{J}_k(\omega) = \min \{g(\omega), \hat{A}_k(\omega)\}$. Namely, if $\alpha^*_k \in \{Q_j^T\}_{j=1}^L$, Lemma 4.1 implies that $g(\omega) \leq \hat{A}_k(\omega)$. On the other hand, if $\alpha^*_k \in \{\beta_k^{F_{\sigma^*(k+1)}}\}$, Lemma 4.2 implies that $g(\omega) > \hat{A}_k(\omega)$. Based on the above, IFCO, an algorithm for Instance–wise Feature selection and Classification with optimum feature
Ordering is presented. Initially, $\varpi$ is set to $\pi_0$, and $\alpha^*_0 = \arg \min_{\alpha^*_0} [\alpha^*_0 \varpi]$ is computed. If $\alpha^*_0 \in \{Q^T_j\}_{j=1}^L$, IFCO classifies the instance under examination using Eq. (4.6). Otherwise, IFCO reviews feature $F_{\sigma^*(1)}$ associated with $\alpha^*_0 \in \{\beta^*_0 F_{\sigma^*(1)}\}$. IFCO repeats these steps until either it decides to classify the instance using $< K$ features, or exhausts all features. Algorithm 4 describes these steps in detail. The input vector sets $\{\alpha^i_k\}$ can be computed using a point–based value iteration algorithm [33] during training. For simplicity, the Perseus algorithm [61] is used, among the several point–based value iteration algorithms in the literature [59]. The Perseus algorithm takes, a fixed number $\zeta$ (e.g., $\sim 100$ [61]) of reachable $\varpi$ vectors from each stage, conditional probability distributions $P(F_k|C_i)$, misclassification costs $Q_{ij}$, and feature review costs $e(F_k)$, as inputs to generate $\{\alpha^i_k\}, k = 0, \ldots, K - 1$.

4.4.2.1 Practical Considerations

A smoothed maximum likelihood estimator is employed to estimate the conditional probability tables, i.e., $P(F_k|C)$, after quantizing the feature space. Specifically, $\hat{P}(F_k = f_k|C = c_i) = \frac{S_{k,i} + 1}{S_i + V}$, where $S_{k,i}$ denotes the number of training instances that satisfy $F_k = f_k$ and $C = c_i$, $S_i$ represents the total number of training instances that satisfy $C = c_i$, and $V$ is the number of quantization levels considered. The prior probabilities are estimated as $\hat{P}(C = c_i) = N_i / \sum_{i=1}^L N_i, i = 1, \ldots, L$.

4.4.2.2 Complexity Analysis

Preprocessing stage: This stage consists of two steps. First, computing conditional probability tables, i.e., $P(F_k|C), k = 1, \ldots, K, i = 1, \ldots, L$, is $O(KL)$. Second, generating number $\zeta$ of reachable $\varpi \in [0, 1]^L$ vectors from all $K$ stages is $O(\zeta KL)$. Thus, the complexity of the preprocessing stage is $O(\zeta KL)$. Training stage: In this stage, the optimum $\{\alpha^i_k\}$ vectors are determined. Computing the optimum $\alpha_k^*$ vectors for all $K$ stages by considering number $\zeta$ of belief points from each stage using Perseus algorithm is $O(\zeta K^2L)$. Thus, model training is $O(\zeta K^2L)$. Testing stage: The computational complexity of computing $\alpha_k^*$ is $O(\zeta L)$, since i) the set $\{\alpha^i_k\}$ contains at most number $\zeta$ of vectors, and ii) the dot product between a pair of $[0, 1]^L$ vectors require $2L - 1$ computations. The complexity of obtaining a new feature is $O(1)$, and updating $\varpi$ using Eq. (4.5) is $O(L)$, since a dot product between a pair of $L$–dimensional vectors must be computed. Hence, IFCO can classify an
Algorithm 4 IFCO

**Input:** Vector sets \( \{ \beta^F_0 \}^{(1)}, \ldots, \{ \beta^F_{K-1} \}^{(K)} \), and misclassification costs \( Q_{ij}, i,j \in \{1,\ldots,L\} \)  

**Output:** Classification decision \( D \) of the instance under examination, number \( R \) of features used

Initialize \( \varpi = \pi_{\sigma^*(0)} \)

for \( k = 0 \) to \( K - 1 \) do

\[ \alpha^*_k = \arg \min_{\alpha_k} [\alpha_k \varpi] \]

if \( \alpha^*_k \in \{ Q_j^{T} \}_{j=1}^L \) then

Break

else

\( \alpha^*_k \in \{ \beta^F_{\sigma^*(k+1)} \} \)

Obtain next feature value \( f_{\sigma^*(k+1)} \)

Update \( \varpi \) using Eq. (4.5)

end if

end for

Return: \( D = \arg \min_{1\leq j \leq L} [Q_j^T \varpi], R = k \)

instance in \( O(\zeta KL) \).

### 4.5 Regression

In this section, the target variable \( \mathcal{C} \) is considered to be a continuous variable with prior probability distribution \( P(c) \), such that \( \int_{\mathcal{C}} P(c) dc = 1 \). The mean square error\(^4\) is considered as the cost function \( \mathcal{L}(D_{\sigma(R)}) \).

\[
\mathcal{L}(D_{\sigma(R)}) = \lambda \mathbb{E} \left\{ \left( D_{\sigma(R)} - C \right)^2 \right\},
\]

(4.10)

where \( \lambda \) is a parameter that controls the trade off between cost of reviewing features and the quality of the prediction in the corresponding optimization function in Eq. (4.1). Let \( \Phi_{\sigma(k)}(c) = P(c|F_{\sigma(1)}, \ldots, F_{\sigma(k)}) \) denote the posterior probability distribution of the target variable \( \mathcal{C} \) given the accumulated information up until the \( k \)th feature assuming ordering \( \sigma \).

The posterior probability distribution \( \Phi_{\sigma(k)}(c) \) can be recursively updated via Bayes’ rule as follows:

\[
\Phi_{\sigma(k+1)}(c) = \frac{P(F_{\sigma(k+1)}|F_{\sigma(1)}, \ldots, F_{\sigma(k)}, c) \Phi_{\sigma(k)}(c)}{\int_{\mathcal{C}} P(F_{\sigma(k+1)}|F_{\sigma(1)}, \ldots, F_{\sigma(k)}, c) \Phi_{\sigma(k)}(c) dc}.
\]

(4.11)

\(^4\)For ease of derivation, the mean squared error is used. However, other cost functions typically adopted for regression can be used instead.
In order to obtain the optimum prediction strategy $D^*_\sigma(R)$ for any given number $\sigma(R)$ of features and ordering $\sigma$, the derivative of the cost function $\mathcal{L}(D^*_\sigma(R))$ in Eq. (4.10) with respect to $D^*_\sigma(R)$ is computed as follows:

$$
\frac{d}{dD^*_\sigma(R)} \mathbb{E} \left\{ (D^*_\sigma(R) - C)^2 | \sigma(R), \sigma \right\} = \frac{d}{dD^*_\sigma(R)} \int_\mathcal{C} P(c|F_{\sigma(1)} \ldots, F_{\sigma(R)}) (D^*_\sigma(R) - c)^2 dc
$$

$$
= \int_\mathcal{C} P(c|F_{\sigma(1)} \ldots, F_{\sigma(R)}) \times 2(D^*_\sigma(R) - c) dc
$$

$$
= \int_\mathcal{C} \Phi_{\sigma(R)}(c) \times 2(D^*_\sigma(R) - c) dc. \quad (4.12)
$$

The optimum prediction strategy $D^*_\sigma(R)$ can be obtained by setting the derivative in Eq. (4.12) to zero, which leads to:

$$
D^*_\sigma(R) = \int_\mathcal{C} \Phi_{\sigma(R)}(c) dc. \quad (4.13)
$$

Since the corresponding $J(\sigma, \sigma(R), D^*_\sigma(R)) \geq J(\sigma, \sigma(R), D^*_\sigma(R))$, the cost function in Eq. (4.10) can be reduced to one which depends only on the feature ordering $\sigma$ and the number $\sigma(R)$ of features as follows:

$$
\tilde{J}(\sigma, \sigma(R)) = \mathbb{E} \left\{ \sum_{k=1}^R e(F_{\sigma(k)}) + g(\Phi_{\sigma(R)}(c)) \right\}, \quad (4.14)
$$

where $g(\Phi_{\sigma(R)}(c)) = \lambda \mathbb{E}\{ (D^*_\sigma(R) - C)^2 \}$. Similar to Theorem 4.1, both the optimum ordering $\sigma^*$ and the optimum number $\sigma^*(R^*)$ of features can be obtained simultaneously using dynamic programming [8] as stated in Theorem 4.3.

**Theorem 4.3.** For $k = K-1, \ldots, 0$, the function $\tilde{J}_k(\Phi_{\sigma^*(k)}(c))$ is related to $\tilde{J}_{k+1}(\Phi_{\sigma^*(k+1)}(c))$ through the equation:

$$
\tilde{J}_k(\Phi_{\sigma^*(k)}(c)) = \min \left[ g(\Phi_{\sigma^*(k)}(c)), \tilde{A}_k(\Phi_{\sigma^*(k)}(c)) \right] \text{ with } \tilde{A}_k(\Phi_{\sigma^*(k)}(c)) = g(\Phi_{\sigma^*(k)}(c)), \quad (4.15)
$$

where $\tilde{A}_k(\Phi_{\sigma^*(k)}(c)) \triangleq \min_{F_{k+1}} \left[ e(F_{k+1}) + \int_{F_{k+1}} \int_\mathcal{C} P(f_{k+1}|F_{\sigma^*(1)}, \ldots, F_{\sigma^*(k)}, c) \Phi_{\sigma^*(k)}(c) dc \right] dF_{k+1}$. 

79
The optimum number $\sigma^*(R^*)$ of features is equal to the first $k < K$ features for which $g(\Phi_{\sigma^*(k)}(c)) \leq \tilde{A}(\Phi_{\sigma^*(k)}(c))$, or $\sigma^*(R^* = K)$ if there are no more features to be reviewed.

4.5.1 Efficient Implementation

Computing the optimum ordering $\sigma^*$ and the optimum number $\sigma^*(R^*)$ of features using the equations in Theorem 4.3 is computationally intensive due to the double integration. Herein, the prior probability density $P(c)$ is approximated using a Gaussian mixture model as follows: $P(c) \approx \sum_{i=1}^{L} P(M = m_i)P(c|M = m_i)$, where $M \in \{m_1, \ldots, m_L\}$ is the latent variable representing the mixture component for $C$, $P(M = m_i) = \eta_i$ is the weight of mixture $m_i$, $P(c|M = m_i)$ denotes a Gaussian distribution $\mathcal{N}(\mu_i, \nu_i^2)$, and $L$ is the number of mixture components. A Gaussian mixture model constitutes a universal approximator of densities, in the sense that any smooth density can be approximated with any specific nonzero amount of error by a Gaussian mixture model with enough components [22]. Under the Gaussian mixture model, the target variable $C$ is assumed to be fully described given the mixing variable $M$; equivalently, $C$ is independent of any other variable given $M$. Using the above fact, the posterior probability distribution $\Phi_{\sigma(k)}(c)$ can be approximated as follows:

$$\Phi_{\sigma(k)}(c) \approx \sum_{i=1}^{L} P(c|m_i, F_{\sigma(1)}, \ldots, F_{\sigma(k)})P(m_i|F_{\sigma(1)}, \ldots, F_{\sigma(k)})$$

$$= \sum_{i=1}^{L} P(c|m_i)P(m_i|F_{\sigma(1)}, \ldots, F_{\sigma(k)}) = C_M^T M_{\sigma(k)},$$

(4.16)

where $C_M \triangleq [P(c|m_1), \ldots, P(c|m_L)]^T$, and $M_{\sigma(k)} \triangleq [P(m_1|F_{\sigma(1)}, \ldots, F_{\sigma(k)}), \ldots, P(m_L|F_{\sigma(1)}, \ldots, F_{\sigma(k)})]^T$. Note that, $M_{\sigma(k)}$ can be recursively updates via Bayes’ rule as follows:

$$M_{\sigma(k)} = \frac{\text{diag} \left( \Delta(F_{\sigma(k)}|F_{\sigma(1)}, \ldots, F_{\sigma(k-1)}, M) \right) M_{\sigma(k-1)}}{\Delta^T(F_{\sigma(k)}|F_{\sigma(1)}, \ldots, F_{\sigma(k-1)}, M) M_{\sigma(k-1)}}$$

(4.17)

where $\Delta(F_{\sigma(k)}|F_{\sigma(1)}, \ldots, F_{\sigma(k-1)}, M) \triangleq [P(F_{\sigma(k)}|F_{\sigma(1)}, \ldots, F_{\sigma(k-1)}, m_1), \ldots, P(F_{\sigma(k)}|F_{\sigma(1)}, \ldots, F_{\sigma(k-1)}, m_L)]^T$, and $M_{\sigma(0)} \triangleq [\eta_1, \eta_2, \ldots, \eta_L]^T \triangleq \eta$. Further, the optimum prediction strategy
$D_{\sigma(R)}^*\text{ can be approximated using } M_{\sigma(k)} \text{ as follows:}$

$$D_{\sigma(R)}^* \approx \int_{\mathcal{C}} \frac{C^T_{M} M_{\sigma(k)} c dc}{c} = \int_{\mathcal{C}} \frac{\sum_{i=1}^{L} c P(c|m_i) P(m_i|F_{\sigma(1)}, \ldots, F_{\sigma(k)}) dc}{c}$$

$$= \sum_{i=1}^{L} P(m_i|F_{\sigma(1)}, \ldots, F_{\sigma(k)}) \int_{\mathcal{C}} c P(c|m_i) dc$$

$$= \sum_{i=1}^{L} P(m_i|F_{\sigma(1)}, \ldots, F_{\sigma(k)}) \mu_i = \mu^T M_{\sigma(k)}, \quad (4.18)$$

where $\mu = [\mu_1, \ldots, \mu_L]^T$. Furthermore, Eq. 4.15 in Theorem 4.3 takes the following form:

$$\widehat{J}_k(M_{\sigma^*(k)}) \approx \min \left[ g(M_{\sigma^*(k)}), \widehat{A}_k(M_{\sigma^*(k)}) \right]$$

$$\widehat{A}_k(M_{\sigma^*(k)}) \approx \min_{F_{k+1} \in Z_k} \left[ B_{k+1}^{F_k}(M_{\sigma^*(k)}) \right], \quad (4.19)$$

where $B_{k+1}^{F_k}(M_{\sigma^*(k)}) = e(F_{k+1}) + \int_{F_{k+1} \in Z_k} \Delta^T(f_{k+1}|F_{\sigma^*(1)}, \ldots, F_{\sigma^*(k)}, M) M_{\sigma^*(k)}$ and $g(M_{\sigma^*(k)})$ takes the following form:

$$g(M_{\sigma^*(k)}) = \lambda \left( \int_{\mathcal{C}} \frac{\sum_{i=1}^{L} P(c|m_i) P(m_i|F_{\sigma^*(1)}, \ldots, F_{\sigma^*(R)}) (\mu^T M_{\sigma^*(k)} - c)^2 dc}{c} \right)$$

$$= \lambda \left( (\mu^T M_{\sigma^*(k)})^2 \sum_{i=1}^{L} P(m_i|F_{\sigma^*(1)}, \ldots, F_{\sigma(R)}) \int_{\mathcal{C}} c P(c|m_i) dc \right.$$

$$- 2\mu^T M_{\sigma^*(k)} \sum_{i=1}^{L} P(m_i|F_{\sigma^*(1)}, \ldots, F_{\sigma(R)}) \int_{\mathcal{C}} c P(c|m_i) dc$$

$$+ \sum_{i=1}^{L} P(m_i|F_{\sigma^*(1)}, \ldots, F_{\sigma(R)}) \int_{\mathcal{C}} c^2 P(c|m_i) dc \right)$$

$$= \lambda \left( (\mu^T M_{\sigma^*(k)})^2 - 2\mu^T M_{\sigma^*(k)} \sum_{i=1}^{L} P(m_i|F_{\sigma^*(1)}, \ldots, F_{\sigma(R)}) \mu_i \right.$$

$$+ \sum_{i=1}^{L} P(m_i|F_{\sigma^*(1)}, \ldots, F_{\sigma(R)}) (\mu_i^2 + \nu_i^2) \right)$$

$$= \lambda \left( \mu^2 + \nu^2 - (\mu^T M_{\sigma^*(k)})^2 \right), \quad (4.20)$$

where $\mu^2 = [\mu_1^2, \ldots, \mu_L^2]^T$ and $\nu^2 = [\nu_1^2, \ldots, \nu_L^2]^T$. 

81
Algorithm 5 IFRO

Input: Functions \(B^*_0(.)\), \(\ldots\), \(B^*_{K-1}(.)\), vectors \(\eta, \mu, \nu^2\) and trade–off parameter \(\lambda\)

Output: Predicted mean \(D\) and variance \(Var\) of the instance under examination, and the number \(R\) of features used

Initialize \(M_{\sigma^*}(0) = \eta\), \(Z_0 = \{F_1, \ldots, F_K\}\)

for \(k = 0\) to \(K - 1\) do

Compute cost of stopping \(g(M_{\sigma^*}(k))\) using Eq. (4.20)

Initialize cost of continuing \(\hat{A} = \infty\) and \(F_{\sigma^*(k+1)} = \emptyset\)

for \(F_{k+1}\) in \(Z_k\) do

if \(\hat{A} > B^{F_{k+1}}_k(M_{\sigma^*}(k))\) then

\(\hat{A} = B^{F_{k+1}}_k(M_{\sigma^*}(k))\)

\(F_{\sigma^*(k+1)} = F_{k+1}\)

end if

end for

if \(g(M_{\sigma^*}(k)) \leq \hat{A}\) then

Break

else

Obtain next feature value \(f_{\sigma^*(k+1)}\)

Update \(M_{\sigma^*}(k)\) using Eq. (4.17)

Remove \(F_{\sigma^*(k+1)}\) from \(Z_K\)

end if

end for

Return: \(D = \mu^T M_{\sigma^*(k)}, Var = g(M_{\sigma^*(k)})/\lambda, R = k\)

4.5.2 IFRO Algorithm

In this section, IFRO, an algorithm for Instance–wise Feature selection and Regression with optimum feature Ordering is presented. Initially, \(M_{\sigma^*}(0)\) and \(Z_0\) is set to \(\eta\) and \(\{F_1, \ldots, F_K\}\), respectively. If \(g(M_{\sigma^*}(0)) \leq \min_{F_1 \in Z_0} \left[ B^{F_1}_k(M_{\sigma^*}(0)) \right] \), IFRO predicts the mean \(D = \mu^T M_{\sigma^*}(0)\) and the variance \(Var = \mu^2 M_{\sigma^*}(0) + \nu^2 M_{\sigma^*}(0) - (\mu^T M_{\sigma^*}(0))^2 = g(M_{\sigma^*}(0))/\lambda\) of the instance under examination. Otherwise, feature \(F_{\sigma^*(1)} = \arg\min_{F_1 \in Z_0} \left[ B^{F_1}_k(M_{\sigma^*}(0)) \right] \) is reviewed. IFRO repeats these steps until either it decides to predicts the instance using \(< K\) features, or exhausts all features. Algorithm 5 describes this process in detail.

4.5.2.1 Practical Considerations

The functions \(B^{F_{k+1}}_k(.)\), \(k = 0, \ldots, K - 1\), in Eq. (4.19) are approximated using value iteration over a fixed number \(\zeta\) (e.g. 1000) of reachable \(M_{\sigma(k)}\) vectors from each stage. Expectation maximization algorithm [52] is employed to fit a Gaussian mixture model to
Table 4.1: Datasets used for Classification.

<table>
<thead>
<tr>
<th>Dataset</th>
<th># Instances</th>
<th># Features</th>
<th># Classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Madelon</td>
<td>2,000</td>
<td>500</td>
<td>2</td>
</tr>
<tr>
<td>Lung Cancer</td>
<td>181</td>
<td>12,533</td>
<td>2</td>
</tr>
<tr>
<td>MLL</td>
<td>72</td>
<td>5,848</td>
<td>3</td>
</tr>
<tr>
<td>Dexter</td>
<td>300</td>
<td>20,000</td>
<td>2</td>
</tr>
<tr>
<td>Car</td>
<td>174</td>
<td>9,182</td>
<td>11</td>
</tr>
<tr>
<td>Lung2</td>
<td>203</td>
<td>3,312</td>
<td>5</td>
</tr>
<tr>
<td>Leukemia</td>
<td>72</td>
<td>7,129</td>
<td>2</td>
</tr>
<tr>
<td>Prostate</td>
<td>102</td>
<td>6,033</td>
<td>2</td>
</tr>
<tr>
<td>Spambase</td>
<td>4601</td>
<td>57</td>
<td>2</td>
</tr>
<tr>
<td>Dorothea</td>
<td>800</td>
<td>100,000</td>
<td>2</td>
</tr>
<tr>
<td>News20</td>
<td>19,996</td>
<td>1,355,191</td>
<td>2</td>
</tr>
</tbody>
</table>

the target variable \( C \) and the conditional probability densities, i.e., \( P(F_k|M), k = 1, \ldots, K \), with \( L \) and \( L_F \), mixing components, respectively.

4.5.2.2 Complexity Analysis

**Training stage:** Approximating the functions \( B^{F_k+1}(.), k = 0, \ldots, K - 1 \), using value iteration over number \( \zeta \) of reachable \( M_{\sigma(k)} \in [0, 1]^L \) vectors from each stage is \( \mathcal{O}(\zeta K^2 LL_F) \).

**Testing stage:** The computational complexity of computing the cost of stopping \( g(M_{\sigma*(k)}) \) using Eq. (4.20) is \( \mathcal{O}(L) \). The complexity of computing the cost of continuing \( \hat{A} \) is \( \mathcal{O}(\zeta KL) \), since approximating the function value \( B^{F_k+1}(M_{\sigma(k)}) \) for unseen \( M_{\sigma(k)} \) during training require to search over a set of at most \( \zeta \) function points. The complexity of obtaining a new feature is \( \mathcal{O}(1) \), updating \( M_{\sigma*(k)} \) using Eq. (4.17) is \( \mathcal{O}(L) \), and removing a feature from the set \( Z_k \) is \( \mathcal{O}(1) \). Hence, IFRO can predict an instance in \( \mathcal{O}(\zeta K^2 L) \).

4.6 Experimental Evaluation

In this section, both classification and regression algorithms’ performance is validated using real–world datasets. All experiments are conducted on an iMac with Quad–Core Intel Core i7 @3.30 GHz CPU, 16 GB memory, and macOS Catalina.
Figure 4.1: Variation of (a) accuracy, and (b) average number of features as a function of the number $V \in \{2, 4, 6, 8, 12, 20, 30, 40, 50, 100\}$ of bins using Lung2, Dexter, Madelon and MLL datasets.

### 4.6.1 Classification

IFCO’s performance is tested using 11 benchmark datasets: 6 DNA Microarray Datasets (Lung Cancer, Lung2, MLL, Car, Leukemia, Prostate) [69], 4 NIPS feature selection challenge datasets (Dexter, Madelon, Dorothea, Spambase) [1], and one high dimensional dataset (News20) [2] (see Table 4.1). The originally provided training and validation sets are used for Madelon, MLL, Dexter, and Dorothea datasets, while five–fold cross–validated results are reported for the remaining datasets.

#### 4.6.1.1 Effect of Feature Space Quantization

Recall, in Section 4.4.2.1, the conditional probability tables $P(C|F_k), \forall k$, are estimated by uniformly quantizing the feature space of $F_k$ into $V$ quantization levels. This section analyses the effect of $V$ on IFCO using four datasets (Lung2, Dexter, Madelon, MLL) (see Fig. 4.1). Observe that IFCO is robust up to $V = 50$, while the performance degrades for $V>50$ for all the datasets. On the other hand, in the Dexter dataset, the performance increases when $V$ increases from 2 to 10. These observations suggest that increasing the resolution of the feature space to a very high value can cause overfitting. At the same time, it can help to accommodate data sparsity in sparse datasets (e.g., Dexter). Thus, $V$ is set
Table 4.2: Accuracy and average number of features used by IFCO for four feature evaluation cost $\epsilon$ values using Lung2, Dexter, Madelon and MLL datasets.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>$\epsilon = 0.1$</th>
<th>$\epsilon = 0.01$</th>
<th>$\epsilon = 0.001$</th>
<th>$\epsilon = 0.0001$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Accuracy</td>
<td>Avg. # Features</td>
<td>Accuracy</td>
<td>Avg. # Features</td>
</tr>
<tr>
<td>Madelon</td>
<td>0.6183</td>
<td>1.00</td>
<td>0.6233</td>
<td>2.02</td>
</tr>
<tr>
<td>Dexter</td>
<td>0.7133</td>
<td>1.00</td>
<td>0.8367</td>
<td>5.14</td>
</tr>
<tr>
<td>MLL</td>
<td>0.9333</td>
<td>1.73</td>
<td>1.00</td>
<td>3.27</td>
</tr>
<tr>
<td>Lung2</td>
<td>0.8771</td>
<td>1.54</td>
<td>0.9412</td>
<td>4.51</td>
</tr>
</tbody>
</table>

Table 4.3: Comparison of accuracy. The highest accuracy, and the second highest accuracy are bolded and gray–shaded, and gray–shaded, respectively. Cells are marked with ‘- -’ if the corresponding method was unable to generate results within a cutoff time of 12 days.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Madelon</td>
<td>0.6233</td>
<td>0.6700</td>
<td>0.6217</td>
<td>0.5180</td>
<td>0.5117</td>
<td>0.5117</td>
<td>0.5817</td>
<td>0.5417</td>
<td>0.5817</td>
<td>0.6050</td>
</tr>
<tr>
<td>Lung2</td>
<td>0.9890</td>
<td>0.9724</td>
<td>0.9890</td>
<td>0.9890</td>
<td>0.9835</td>
<td>0.9779</td>
<td>0.9890</td>
<td>0.9890</td>
<td>0.9724</td>
<td>0.9613</td>
</tr>
<tr>
<td>MLL</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>0.9467</td>
<td>0.9600</td>
<td>0.9067</td>
<td>0.8667</td>
<td>0.8000</td>
<td>0.8000</td>
<td>0.9333</td>
</tr>
<tr>
<td>Dexter</td>
<td>0.8367</td>
<td>0.8533</td>
<td>0.8133</td>
<td>0.7967</td>
<td>0.8527</td>
<td>0.7375</td>
<td>0.7300</td>
<td>0.7900</td>
<td>0.7900</td>
<td>0.5000</td>
</tr>
<tr>
<td>Car</td>
<td>0.8620</td>
<td>0.7471</td>
<td>0.8097</td>
<td>0.8274</td>
<td>0.5973</td>
<td>0.7929</td>
<td>0.7982</td>
<td>0.6082</td>
<td>0.5575</td>
<td>0.6429</td>
</tr>
<tr>
<td>Lung2</td>
<td>0.9412</td>
<td>0.8573</td>
<td>0.8820</td>
<td>0.8918</td>
<td>0.9117</td>
<td>0.8717</td>
<td>0.8817</td>
<td>0.8420</td>
<td>0.8471</td>
<td>0.8820</td>
</tr>
<tr>
<td>Leukemia</td>
<td>0.9429</td>
<td>0.9571</td>
<td>0.9571</td>
<td>0.9571</td>
<td>0.9438</td>
<td>0.7914</td>
<td>0.9295</td>
<td>0.9295</td>
<td>0.8867</td>
<td>0.8324</td>
</tr>
<tr>
<td>Prostate</td>
<td>0.9410</td>
<td>0.9005</td>
<td>0.9310</td>
<td>0.9010</td>
<td>0.9210</td>
<td>0.8148</td>
<td>0.8910</td>
<td>0.8633</td>
<td>0.8833</td>
<td>0.9014</td>
</tr>
<tr>
<td>Spambase</td>
<td>0.8170</td>
<td>0.8104</td>
<td>0.8467</td>
<td>0.5109</td>
<td>0.7870</td>
<td>0.8598</td>
<td>0.8241</td>
<td>0.8011</td>
<td>0.8011</td>
<td>0.8074</td>
</tr>
<tr>
<td>Dorothea</td>
<td>0.9400</td>
<td>0.9429</td>
<td>0.9400</td>
<td>0.7714</td>
<td>0.9314</td>
<td>0.9314</td>
<td>0.9114</td>
<td>0.9429</td>
<td>0.9000</td>
<td>0.6457</td>
</tr>
<tr>
<td>News20</td>
<td>0.7575</td>
<td>0.7503</td>
<td>0.7352</td>
<td>0.6346</td>
<td>5.64</td>
<td>7.18</td>
<td>5.32</td>
<td>7.05</td>
<td>7.95</td>
<td>7.14</td>
</tr>
</tbody>
</table>

| Avg. rank | 2.50  | 4.00  | 3.00  | 5.23  | 5.64  | 7.18  | 5.32  | 7.05  | 7.95  | 7.14  |

to a moderate value (i.e., 4), except for sparse datasets (i.e., Dexter, Dorothea, Spambase, and News20), where $V$ is set to a slightly higher value (i.e., 10).

4.6.1.2 Accuracy as a Function of Average Number of Features

Table 4.2 illustrates the effect of using constant feature evaluation costs (i.e., $e(F_k) = e, \forall k$) on IFCO for constant misclassification cost coefficients (i.e., $Q_{ij} = 1 \forall i \neq j, Q_{ii} = 0, i, j \in \{1, \ldots, L\}$). When feature evaluation cost $e$ is set to a higher value, i.e., $e = 0.1$, the model underfits due to using a small portion of the complete feature set. On the other hand, when feature evaluation cost is set to a lesser value, i.e., $e = 0.0001$, the model overfits due to introducing noisy features into the model. Thus, feature evaluation cost is set to a moderate value, $e = 0.01$, to obtain a generalized performance.
Table 4.4: Comparison of average number of features used. The minimum and the second minimum average number of features used are bolded and gray–shaded, and gray–shaded accordingly. Cells are marked with ‘- -’ if the corresponding method was unable to generate results within a cutoff time of 12 days.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Madelon</td>
<td>2.02</td>
<td>5.03</td>
<td>4.09</td>
<td>55.48</td>
<td><strong>2.00</strong></td>
<td><strong>2.00</strong></td>
<td>3.00</td>
<td>3.00</td>
<td>5.00</td>
<td>4.00</td>
</tr>
<tr>
<td>Lung Cancer</td>
<td>1.50</td>
<td><strong>3.35</strong></td>
<td>2.03</td>
<td>6.56</td>
<td>37.20</td>
<td>8.40</td>
<td>52.00</td>
<td>6.80</td>
<td>4.00</td>
<td>4.60</td>
</tr>
<tr>
<td>MLL</td>
<td>3.27</td>
<td>3.40</td>
<td>5.07</td>
<td>14.69</td>
<td>11.00</td>
<td>12.00</td>
<td>28.00</td>
<td>5.00</td>
<td><strong>3.00</strong></td>
<td>7.00</td>
</tr>
<tr>
<td>Dexter</td>
<td>5.14</td>
<td>10.95</td>
<td>12.80</td>
<td>243.4</td>
<td>10.00</td>
<td>104.0</td>
<td>21.00</td>
<td>9.00</td>
<td>6.00</td>
<td><strong>1.00</strong></td>
</tr>
<tr>
<td>Car</td>
<td>9.82</td>
<td>24.20</td>
<td>12.90</td>
<td>540.20</td>
<td>6.80</td>
<td>36.00</td>
<td>41.40</td>
<td>8.40</td>
<td>5.20</td>
<td>24.40</td>
</tr>
<tr>
<td>Lung2</td>
<td><strong>4.51</strong></td>
<td>11.30</td>
<td>15.59</td>
<td>27.91</td>
<td>16.20</td>
<td>18.00</td>
<td>28.20</td>
<td>9.40</td>
<td>5.80</td>
<td>34.40</td>
</tr>
<tr>
<td>Leukemia</td>
<td><strong>1.51</strong></td>
<td>1.90</td>
<td>2.08</td>
<td>9.53</td>
<td>4.40</td>
<td>13.40</td>
<td>21.60</td>
<td>4.60</td>
<td>2.20</td>
<td>3.20</td>
</tr>
<tr>
<td>Prostate</td>
<td>3.26</td>
<td>4.08</td>
<td>3.34</td>
<td>10.39</td>
<td>5.80</td>
<td>40.20</td>
<td>14.00</td>
<td>3.80</td>
<td><strong>1.60</strong></td>
<td>7.00</td>
</tr>
<tr>
<td>Spambase</td>
<td><strong>3.90</strong></td>
<td>4.72</td>
<td>7.47</td>
<td>56.00</td>
<td>7.60</td>
<td>42.20</td>
<td>24.60</td>
<td>33.80</td>
<td>33.80</td>
<td>42.60</td>
</tr>
<tr>
<td>Dorothea</td>
<td><strong>2.08</strong></td>
<td>2.29</td>
<td>2.89</td>
<td>8.10</td>
<td>17.40</td>
<td>34.00</td>
<td>32.00</td>
<td>24.00</td>
<td>3.00</td>
<td>113.0</td>
</tr>
<tr>
<td>News20</td>
<td><strong>25.18</strong></td>
<td>43.68</td>
<td>81.70</td>
<td>4000.6</td>
<td><strong>---</strong></td>
<td><strong>---</strong></td>
<td><strong>241.8</strong></td>
<td><strong>---</strong></td>
<td><strong>---</strong></td>
<td><strong>---</strong></td>
</tr>
<tr>
<td>Avg. rank</td>
<td><strong>1.82</strong></td>
<td><strong>3.82</strong></td>
<td><strong>4.36</strong></td>
<td><strong>8.09</strong></td>
<td><strong>5.50</strong></td>
<td><strong>7.77</strong></td>
<td><strong>7.91</strong></td>
<td><strong>5.32</strong></td>
<td><strong>5.39</strong></td>
<td><strong>6.82</strong></td>
</tr>
</tbody>
</table>

4.6.1.3 Comparison with Baselines

IFCO is compared with i) 3 dynamic instance–wise feature selection methods: IFC2F [41], ETANA [40], F–ETANA [40], and ii) 6 streaming feature selection methods: OFS–Density [75], OFS–A3M [76], SAOLA [71], OSFS [66], Fast–OSFS [66], and Alpha–Investing [74]. Streaming feature selection methods [75, 76, 71, 66, 74] select a feature if it satisfies an appropriately defined criterion (e.g., belongs in the approximated Markov blanket of the target variable [71, 66], p–statistic is greater than a dynamically varying threshold [74]), or such that the boundary region of the decision is maintained as little as possible [75, 76]. These methods are designed to handle sequentially arriving features during model training and select a common global subset of features that is used to classify all instances during testing. The main reason for comparing with such methods is that they have been shown to outperform standard feature selection algorithms and scale well in high dimensional settings.

IFCO works similar to IFC2F, ETANA, and F–ETANA, where it assumes all features are available during training, while during testing, features arrive sequentially one at a time for each data instance. However, in IFCO, the feature ordering is different for different data instances, while IFC2F, ETANA, and F–ETANA consider a fixed ordering for all the data instances.

For a fair comparison, all streaming feature selection methods use a K–NN classifier with three neighbors to evaluate a selected feature subset since it has been shown to outperform SVM, CART, and J48 classifiers on the datasets used in [75, 71]. At the same time,
Table 4.5: Comparison of time (in seconds) required for feature selection (F), classification (C), joint feature selection and classification (F+C), model training (T) and preprocessing (P). The minimum and the second minimum F+C times are bolded and gray–shaded, and gray–shaded accordingly. Cells are marked with ‘−−’ if the corresponding method was unable to generate results within a cutoff time of 12 days.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Time required</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>F</td>
</tr>
<tr>
<td>Madelon</td>
<td>2.903</td>
</tr>
<tr>
<td>Lung Cancer</td>
<td>0.007</td>
</tr>
<tr>
<td>MLL</td>
<td>0.010</td>
</tr>
<tr>
<td>Dexter</td>
<td>0.704</td>
</tr>
<tr>
<td>Car</td>
<td>0.049</td>
</tr>
<tr>
<td>Lung2</td>
<td>0.020</td>
</tr>
<tr>
<td>Leukemia</td>
<td>0.006</td>
</tr>
<tr>
<td>Prostate</td>
<td>0.034</td>
</tr>
<tr>
<td>Spambase</td>
<td>0.461</td>
</tr>
<tr>
<td>Dorothea</td>
<td>0.238</td>
</tr>
<tr>
<td>News20</td>
<td>67.01</td>
</tr>
<tr>
<td>Avg. rank</td>
<td>4.00</td>
</tr>
</tbody>
</table>
able or has been provided by their authors. All methods use the same training and testing datasets. Finally, the same metrics (i.e., accuracy, number of features used, time) used by the baselines are adopted (see Tables 4.3, 4.4, and 4.5).

**Madelon**: IFCO achieves the second–highest accuracy, using only 2.02 features on average. IFC
$^2$F achieves the highest accuracy but requires 150% more features compared to IFCO.

**Lung Cancer**: IFCO, ETANA, F–ETANA, SAOLA, and Fast–OSFS achieves the highest accuracy. Among these highest accuracy achieving methods, IFCO uses the minimum number of features on average.

**MLL**: IFCO, IFC
$^2$F, and ETANA achieve 100% accuracy. However, IFC
$^2$F and ETANA require 4.0% and 55.0%, respectively, more features on average than IFCO.

**Dexter**: IFC
$^2$F achieves the highest accuracy but requires 130% more features on average for a difference of 1.9% in accuracy compared to IFCO.

**Car**: IFCO achieves the highest accuracy (4.2% improvement) and uses 97.1% fewer features on average than F–ETANA, which achieves the second–highest accuracy.

**Lung2**: IFCO achieves the highest accuracy and requires the minimum number of features on average.

**Leukemia**: IFC
$^2$F, ETANA, and F–ETANA achieve the highest accuracy but requires 26.7%, 38.7%, and 535%, respectively, more features on average for a slight difference of 1.4% in accuracy compared to IFCO.

**Prostate**: IFCO achieves the highest accuracy using only 3.26 features on average. This corresponds to an improvement of 1.1% accuracy with 2.3% fewer features on average than ETANA, the second–highest accuracy achieved method.

**Spambase**: OFS–A3M achieves the highest accuracy but requires 982% more features for a difference of 4.3% in accuracy compared to IFCO.

**Dorothea**: IFCO achieves the second–highest accuracy (only 0.3% difference compared to the highest accuracy) using the minimum number of features on average.

Note that, IFCO achieves the highest accuracy using the minimum number of features on average compared to all the methods for most of the datasets. This is because IFCO sequentially selects the most appropriate feature subset that improves the classification accuracy per data instance. Specifically, IFCO optimizes the trade–off between the feature
evaluation cost and the misclassification cost per data instance concerning all the possible feature orderings, thus evaluating fewer features on average without sacrificing classification accuracy. However, IFCO is slower in joint feature selection and classification compared to both IFC²F and ETANA.

4.6.1.4 Performance Assessment on a High Dimensional Dataset

This section demonstrates the ability of IFCO to scale for more than 1.3 million features using the News20 dataset (second last row in Tables 4.3, 4.4, 4.5). Except for IFCO, IFC²F, ETANA, F-ETANA, and SAOLA, the rest of the methods could not generate results within a cutoff time of 12 days. IFCO achieves the second–highest accuracy using the minimum number of features on average. This performance corresponds to a difference of 2.7% accuracy with 89.6% fewer features than SAOLA, the highest accuracy achieved baseline. Further, IFCO is 22 times faster in joint feature selection and classification compared to SAOLA.

4.6.1.5 Statistical Significance

Friedman test, a standard method to compare the performance of different approaches across multiple datasets [14], is conducted to test for the statistical significance of the results presented in Sections 4.6.1.3 and 4.6.1.4. Each method’s average ranking (Avg. rank) is reported in the last row in Tables 4.3–4.5. The p-values of the Friedman test on classification accuracy, the average number of features used, and time required for joint feature selection and classification are $9.34 \times 10^{-7}$, $2.73 \times 10^{-10}$, and $7.46 \times 10^{-26}$, respectively. Thus, there is a significant difference [14] in the performance of IFCO and the baselines.

4.6.1.6 Demonstration of Instance–wise Feature Selection

The instance–wise nature of IFCO is demonstrated in Table 4.6 and Figure 4.2 using four examples from the IMDB movie reviews dataset (50,000 instances, 89,523 features, two classes) [44]. The originally provided training and validation sets with the bag–of–words features are used. In Figure 4.2, the variation of the posterior distribution $\pi_{\sigma^*(k)}$ is demonstrated until IFCO stops. Once it stops, the instance is assigned to the class with the maximum posterior probability due to constant misclassification costs, i.e., $Q_{01} = Q_{10} = 1, Q_{00} = Q_{11} = 0$ (see Eq. (4.6)). Observe that the proposed framework evaluates
Table 4.6: Words (features) picked by IFCO are highlighted in yellow. The true/predicted label is given at the end of each review. The second column reports features selected in ascending order (Y–axis) versus feature value (X–axis).

<table>
<thead>
<tr>
<th>IMDB Review Text (True Label, Predicted Label)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>(a) Everywhere I hear that people are calling this show <em>bad</em> because its premise is too far fetched....well maybe a do-good cab driver in Philly is pushing it a little (at least the cab drivers ive met), but that’s what makes the show <em>great</em>. The fact that this show is a little off the reality track is an issue but its still enjoyable and fun. Its highly watchable and even though u know Mike wins out in the end he never wins in life. David Morse is a <em>great</em> actor and does a <em>great</em> job in the title role. His supporting cast is great and i must say the location of the show is especially <em>great</em>! All in all I watch this show not because im looking for a good dose of reality or a show with lots of action, I watch this show because its got <em>great</em> acting, a good premise, and a <em>great</em> story-line every week. It’s also a plus when i can pick out the landmarks he drives by, or know what intersection he’s at. I Love this show and I love Philly!! Give this show a shot! <em>(positive, positive)</em></td>
<td></td>
</tr>
<tr>
<td>(b) One of the <em>worst</em> things a film studio can do is exploit the tragedies of others, commercializing a ‘shock’ or ‘gore’ factor in order to sell tickets to be able to buy their Birch a new diamond necklace. Another <em>worst</em> thing is to totally misrepresent the true facts of an incredible saga by fabricating events, dialog and images to the director’s own liking. Lastly, one of the <em>worst</em> things a film studio can do is to use bottom-of-the-barrel actors and shoot it all on a sound stage that was rented for fifty cents a day. All three of these travesties the makers of this film are guilty of. This is, hands-down, the <em>worst</em> movie I have ever seen, and I’ve seen thousands. A score of ‘1’ is too good for this waste of celluloid. Not only should the filmmakers be ashamed for making it, they should be ashamed for negatively exploiting the heroes of this story, which are the people who experienced this tragedy firsthand, both the living and the dead. <em>(negative, negative)</em></td>
<td></td>
</tr>
<tr>
<td>(c) This is not a good movie at all. I cannot believe that after fifty years, this movie gets the National Award when there have been such gems from Marathi cinema that have been so systematically ignored. . . . There are just so many things that are wrong and lacking in this movie that it amazing it even got considered for the National Award. That this movie is awarded as the best movie to come out this year goes to show the biased judgment of people who hold the reins of Indian cinema and the diminutive understanding of the people who blindly appreciate this movie. The topic chosen is <em>great</em>. It is important that such movies be made but only by people who are able to handle them. . . . It is amazing that such a young boy could give such a respectable performance. He put many of today’s actors to shame. Unintentionally maybe, but he brought his own innocence to his character and that made it a memorable performance. . . . Lack of research and not of funds, is what makes ‘Shwaas’ such a <em>bad</em> movie. <em>(negative, positive)</em></td>
<td></td>
</tr>
<tr>
<td>(d) OK, I am a sucker. I loved it. I had no expectations and had them all fulfilled. It was a <em>terrible</em> movie. I loved it. I have managed to wear out a DVD from over use. No one can understand my obsession. I can’t either, to tell the truth. For those who have seen the movie this will come as no surprise, but I asked the clerk at the video store if I could buy a copy and I could because there were two in stock and only one had been check out and over half of the time it had been to me. Now, the movie is <em>terrible</em>. The special effects are <em>terrible</em>. The acting is <em>terrible</em>, but I loved it. The actors are silly, the plot silly, the goofs numerous–like being able to see through the monsters, The &quot;arachnids&quot; looked like they were made out of plastic garbage bags (maybe they were), There was light underground, TNT wasn’t deafening, etc... You must really love B-Movies to get any enjoyment out of this...alcohol helps enormously for others. <em>(positive, negative)</em></td>
<td></td>
</tr>
</tbody>
</table>
Figure 4.2: Variation of the posterior distributions, i.e., $P(\text{positive} | F_1, \ldots, F_k)$ (in blue) and $P(\text{negative} | F_1, \ldots, F_k)$ (in orange) for four IMDB reviews in Table 4.6.

more features to predict challenging reviews such as (c) and (d) compared to easy and straightforward reviews such as (a) and (b). In summary, IFCO selects different features for different data instances in a dynamic setting and predicts the target label based on the observed features.

4.6.1.7 Robustness to Missing Features

This section demonstrates the robustness of IFCO to missing features during testing. Specifically, a random subset of size $x\%$ is removed from each test instance, and the posterior probability $\pi_{\sigma^*(k)}$ is kept unchanged if a missing feature is encountered. Fig. 4.3 shows the accuracy variation and the average number of features used when $x$ increases from 1\% to 50\%. Observe that IFCO is robust up to 30\% of missing features. This illustrates the ability of IFCO to identify the most informative features even when some crucial features are missing.
Figure 4.3: Variation of (a) accuracy, and (b) average number of features as the percentage of missing features increases from 1% to 50% using Lung Cancer, Dexter, Madelon and MLL datasets.

Table 4.7: Datasets used for Regression.

<table>
<thead>
<tr>
<th>Dataset</th>
<th># Instances</th>
<th># Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boston Houses</td>
<td>506</td>
<td>13</td>
</tr>
<tr>
<td>Computer Activity</td>
<td>8192</td>
<td>21</td>
</tr>
<tr>
<td>Diabetes</td>
<td>442</td>
<td>10</td>
</tr>
<tr>
<td>Student Performance</td>
<td>649</td>
<td>32</td>
</tr>
</tbody>
</table>

4.6.2 Regression

The performance of IFRO is analyzed using four datasets: i) Boston Houses [24], ii) Computer Activity [13], iii) Diabetes [18], and iv) Student Performance [11] (see Table 4.7). 70% of the data instances in each dataset are used for training, while the remaining 30% are used for testing. The numbers $L$ and $L_F$ of mixing components in the Gaussian mixture models of $P(C)$ and $P(F_k|C)$, respectively, are set to 10.

4.6.2.1 Comparison with Baselines

IFRO's performance is compared with two state–of–the–art regression methods: i) linear regression, ii) support vector regression with Gaussian kernel (SVR), two offline feature selection methods: i) linear model trained with L–1 prior as a regularizer (LASSO), ii) least
angle regression (LAR) [18], one streaming feature selection method: Alpha–Investing [74], and Decision tree regression (Trees). Standard implementations from the python sklearn package [52] are considered. It is important to note that all the above state–of–the–art regression methods predict a point estimate, whereas IFRO predicts a Gaussian distribution, i.e., $\Phi_{\sigma^*(R)}(c)$, for each test instance (see Figure 4.4). To compute mean absolute error (MAE) and mean squared error (MSE) in Table 4.8, $\Phi_{\sigma^*(R)}(c)$’s mean is considered. The results in Table 4.8 are summarized below.

**Boston:** IFRO achieves the minimum MAE and MSE using only 3.64 features on average.
Figure 4.5: Features picked by IFRO (X–axis) on 4 test instances of the Boston Houses dataset. Y–axis: house price × $1000. Note that all 4 instances start with the mean of the prior house price distribution. **LSTAT**: percentage lower status of the population, **INDUS**: proportion of non-retail business acres per town, **AGE**: proportion of owner-occupied units built prior to 1940, **CHAS**: Charles River dummy variable (= 1 if tract bounds river; 0 otherwise), **PTRATIO**: pupil–teacher ratio by town, **B** = 1000×(B_k − 0.63)^2, where B_k is the proportion of blacks by town, **RM**: average number of rooms per dwelling, **TAX**: full–value property–tax rate per $10,000, **ZN**: Proportion of residential land zoned for lots over 25,000 sq.ft., **RAD**: index of accessibility to radial highways, **DIS**: Weighted distances to five Boston employment centers.

**Computer Activity**: Tree achieves the minimum MAE and MSE, but uses 145.7% more features to improve 15.6% and 17.5% in MAE and MSE, respectively, compared to IFRO. **Diabetes**: IFRO achieves the minimum MAE using 3.74 features on average. Linear regression achieves the minimum MSE (1.9% better) but uses 167.4% more features than IFRO. **Student Performance**: IFRO achieves the minimum MSE using the minimum number of features. LASSO achieves the minimum MAE (0.4% better) but requires 3.6% more features.

IFRO achieves the minimum MAE and MSE using fewer features on average than the baselines for most of the datasets. This is because IFRO uses the most relevant feature subset for each individual data instance to make an accurate prediction. Further, IFRO provides uncertainty over the prediction.

### 4.6.2.2 Demonstration of Instance–wise Feature Selection

In Figure 4.5, the instance–wise nature of IFRO is demonstrated using four test instances from the Boston Houses dataset. Here, four subfigures represent example instances from the following four scenarios (**true, predicted**): (a) predicts accurately in the upper range (50.0, 48.8), (b) predicts accurately in the lower range (15.6, 15.4), (c) overestimates (16.5, 27.4), and (d) underestimates (27.5, 15.4), the actual house price. Observe
that IFRO uses different subsets of features for different data instances to make a prediction.

4.6.2.3 Robustness to Missing Features

Fig. 4.6 demonstrates the robustness of IFRO to missing features during joint feature selection and regression using the same procedure described in Section 4.6.1.7. Similar to IFCO, this illustrates the ability of IFRO to identify the most informative features even when some crucial features are missing.

4.7 Concluding Remarks

In this chapter, a framework to perform dynamic instance–wise feature selection that selects both the order and the number of features for each data instance individually and works in both classification and regression settings is proposed. For each data instance individually, the optimum feature ordering, the optimum number of features, and the optimum prediction rule for both classification and regression problems are derived. IFCO and IFRO are proposed for classification and regression problems, respectively, that implement the corresponding optimum solution. The effectiveness of IFCO and IFRO is illustrated on various real–world datasets. The proposed methods robustly perform well on all of them,
with comparable and often superior performance than the prior art.

In the final chapter, we propose this work as an interpretable machine learning framework.

4.8 Appendix

4.8.1 Proof of Theorem 4.1

At the final stage $k = K$, assuming that all $K$ features have been evaluated, the only remaining expected cost is the optimum misclassification cost, i.e., $\hat{J}_K(\pi_{\sigma^*(K)}) = g(\pi_{\sigma^*(K)})$. At any intermediate stage $k = 0, 1, \ldots, K - 1$, the optimum decision maker can either incur $g(\pi_{\sigma^*(k)})$, which is the optimum cost of stopping the feature evaluation process, or continue with the next optimum feature $F_{\sigma^*(k+1)}$ and incur $e(F_{\sigma^*(k+1)}) + \hat{J}_{k+1}(\pi_{\sigma^*(k+1)})$, which corresponds to the cost of evaluating $F_{\sigma^*(k+1)}$ plus the cost of continuing optimally at stage $k+1$. However, at stage $k$, the optimum feature $F_{\sigma^*(k+1)}$ to evaluate and its assignment $f_{\sigma^*(k+1)}$ are not known. For any $F_{k+1}$, the expected optimum cost-to-go is $e(F_{k+1}) + \mathbb{E}\{\hat{J}_{k+1}(\pi_{k+1})|\pi_{\sigma^*(k)}\}$, where $\pi^1_{k+1} \triangleq P(C = c_i|F_{\sigma^*(1)}, \ldots, F_{\sigma(k)}, F_{k+1})$. The next optimum feature $F_{\sigma^*(k+1)}$ must be selected from the set $Z_k$ of remaining features, such that the total cost until termination is minimized. In other words, the optimum cost-to-go $\hat{A}_k(\pi_{\sigma^*(k)})$ is:

$$\hat{A}_k(\pi_{\sigma^*(k)}) \triangleq \min_{F_{k+1} \in Z_k} \left[ e(F_{k+1}) + \mathbb{E}\{\hat{J}_{k+1}(\pi_{k+1})|\pi_{\sigma^*(k)}\} \right].$$ (4.21)

Using Bayes’ rule to express $\pi_{k+1}$ using $\pi_{\sigma^*(k)}$, and expanding the expectation operator, $\hat{A}_k(\pi_{\sigma^*(k)})$ takes the form:

$$\hat{A}_k(\pi_{\sigma^*(k)}) = \min_{F_{k+1} \in Z_k} \left[ e(F_{k+1}) + \sum_{F_{k+1}} P(F_{k+1}|F_{\sigma^*(1)}, \ldots, F_{\sigma^*(k)}) \right. \right.$$

$$\times \hat{J}_{k+1} \left( \frac{\text{diag} \left( \Delta(F_{k+1}|F_{\sigma^*(1)}, \ldots, F_{\sigma^*(k)}, C) \pi_{\sigma^*(k)} \right)}{\Delta^T(F_{k+1}|F_{\sigma^*(1)}, \ldots, F_{\sigma^*(k)}, C) \pi_{\sigma^*(k)}} \right).$$ (4.22)
Note that the probability \( P(F_{k+1}|F_{\sigma^*(1)}, \ldots, F_{\sigma^*(k)}) \) can be simplified as follows:

\[
P(F_{k+1}|F_{\sigma^*(1)}, \ldots, F_{\sigma^*(k)}) = \frac{P(F_{k+1}, F_{\sigma^*(1)}, \ldots, F_{\sigma^*(k)})}{P(F_{\sigma^*(1)}, \ldots, F_{\sigma^*(k)})} \\
= \sum_{i=1}^{L} \left[ P(F_{k+1}|F_{\sigma^*(1)}, \ldots, F_{\sigma^*(k)}, C = c_i) \frac{P(C = c_i)}{P(F_{\sigma^*(1)}, \ldots, F_{\sigma^*(k)})} \right] \\
= \sum_{i=1}^{L} \left[ P(F_{k+1}|F_{\sigma^*(1)}, \ldots, F_{\sigma^*(k)}, C = c_i) \frac{P(C = c_i)}{P(F_{\sigma^*(1)}, \ldots, F_{\sigma^*(k)})} \right] \\
= \sum_{i=1}^{L} P(f_{k+1}|F_{\sigma^*(1)}, \ldots, F_{\sigma^*(k)}, c) \pi_{\sigma^*(k)} \\
= \Delta^T(F_{k+1}|F_{\sigma^*(1)}, \ldots, F_{\sigma^*(k)}, C) \pi_{\sigma^*(k)}. \quad (4.23)
\]

The desired result can be obtained by substituting the result in Eq. (4.23) to Eq. (4.22) as follows:

\[
\hat{A}_k(\pi_{\sigma^*(k)}) = \min_{F_{k+1} \in \mathbb{Z}_k} \left[ e(F_{k+1}) + \sum_{F_{k+1}} \Delta^T(F_{k+1}|F_{\sigma^*(1)}, \ldots, F_{\sigma^*(k)}, C) \pi_{\sigma^*(k)} \right] \\
\times \hat{J}_{k+1}\left( \frac{\text{diag} \left( \Delta^T(F_{k+1}|F_{\sigma^*(1)}, \ldots, F_{\sigma^*(k)}, C) \pi_{\sigma^*(k)} \right)}{\Delta^T(F_{k+1}|F_{\sigma^*(1)}, \ldots, F_{\sigma^*(k)}, C) \pi_{\sigma^*(k)}} \right), \quad (4.24)
\]

which completes the proof.

4.8.2 Proof of Lemma 4.1

Let us consider the definition of \( g(\varpi) \):

\[
g(\varpi) \triangleq \min_{1 \leq j \leq L} \left[ Q_j^T \varpi \right], \varpi \in [0, 1]^L.
\]
The term $Q^T_j \varpi$ is linear with respect to $\varpi$. Since the minimum of linear functions is a concave, piecewise linear function, $g(\varpi)$ is a concave, piecewise linear function. Concavity also assures the continuity of this function. Minimization over finite $L$ guarantees that the function $g(\varpi)$ is made up of at most $L$ hyperplanes represented by the vectors in the set $\{Q^T_j\}_{j=1}^L$.

### 4.8.3 Proof of Lemma 4.2

First, consider the function $\hat{A}_{K-1}(\varpi)$ given by:

$$
\hat{A}_{K-1}(\varpi) = \min_{F_K} \left[ e(F_K) + \sum_{F_K} \Delta^T(F_K|F_{\sigma^*(1)}, \ldots, F_{\sigma^*(K-1)}, C) \varpi \right.
\times \hat{J}_K \left( \frac{\text{diag} \left( \Delta(F_K|F_{\sigma^*(1)}, \ldots, F_{\sigma^*(K-1)}, C) \right) \varpi}{\Delta^T(F_K|F_{\sigma^*(1)}, \ldots, F_{\sigma^*(K-1)}, C) \varpi} \right].
$$

(4.25)

Using the fact that $\hat{J}_K(\varpi) = g(\varpi)$, Eq. (4.25) can be rewritten as follows:

$$
\hat{A}_{K-1}(\varpi) = \min_{F_K} \left[ e(F_K) + \sum_{F_K} \Delta^T(F_K|F_{\sigma^*(1)}, \ldots, F_{\sigma^*(K-1)}, C) \varpi \right.
\times g \left( \frac{\text{diag} \left( \Delta(F_K|F_{\sigma^*(1)}, \ldots, F_{\sigma^*(K-1)}, C) \right) \varpi}{\Delta^T(F_K|F_{\sigma^*(1)}, \ldots, F_{\sigma^*(K-1)}, C) \varpi} \right).
$$

(4.26)

Using the definition of $g(\varpi)$, Eq. (4.26) can be rewritten as follows:

$$
\hat{A}_{K-1}(\varpi) = \min_{F_K} \left[ e(F_K) + \sum_{F_K} \Delta^T(F_K|F_{\sigma^*(1)}, \ldots, F_{\sigma^*(K-1)}, C) \varpi \right.
\times \min_j \left[ Q^T_j \text{diag} \left( \Delta(F_K|F_{\sigma^*(1)}, \ldots, F_{\sigma^*(K-1)}, C) \right) \varpi \right].
$$

(4.27)

Using the facts that $Q_j$ and $\Delta(F_K|F_{\sigma^*(1)}, \ldots, F_{\sigma^*(K-1)}, C)$ are non-negative vectors, Eq. (4.27) can be simplified as follows:

$$
\hat{A}_{K-1}(\varpi) = \min_{F_K} \left[ e(F_K) + \sum_{F_K} \Delta^T(F_K|F_{\sigma^*(1)}, \ldots, F_{\sigma^*(K-1)}, C) \varpi \right.
\times \min_j \left[ Q^T_j \text{diag} \left( \Delta(F_K|F_{\sigma^*(1)}, \ldots, F_{\sigma^*(K-1)}, C) \right) \varpi \right].
$$

(4.28)

Note that the term $Q^T_j \text{diag}(\Delta(F_K|F_{\sigma^*(1)}, \ldots, F_{\sigma^*(K-1)}, C)) \varpi$ is linear with respect to $\varpi$. Using the facts that i) $e(F_K) > 0$, ii) the minimum of linear functions is a concave, piecewise
linear function, and iii) the non-negative sum of concave/piecewise linear functions is also a concave/piecewise linear function, implies that \( \hat{A}_{K-1}(\varpi) \) is a concave, piecewise linear function. Concavity also assures the continuity.

Then, consider the function \( \hat{A}_{K-2}(\varpi) \) given by:

\[
\hat{A}_{K-2}(\varpi) = \min_{F_{K-1}} \left[ e(F_{K-1}) + \sum_{F_{K-1}} \Delta^T(F_{K-1}|F_{\sigma^*(1)}, \ldots, F_{\sigma^*(K-2)}, \mathcal{C}) \varpi \right] \times \hat{J}_{K-1} \left( \frac{\Delta^T(F_{K-1}|F_{\sigma^*(1)}, \ldots, F_{\sigma^*(K-2)}, \mathcal{C}) \varpi}{\Delta^T(F_{K-1}|F_{\sigma^*(1)}, \ldots, F_{\sigma^*(K-2)}, \mathcal{C}) \varpi} \right).
\]

(4.29)

Note that \( \hat{J}_{K-1}(\varpi) = \min[g(\varpi), \hat{A}_{K-1}(\varpi)] \) (see Theorem 1). Using the facts that i) \( g(\varpi) \) is a concave, piecewise linear function, ii) \( \hat{A}_{K-1}(\varpi) \) is a concave, piecewise linear function, and iii) the minimum of two concave/piecewise linear functions is also a concave/piecewise linear function, implies that \( \hat{J}_{K-1}(\varpi) \) is also concave and piecewise linear. Recall that the non-negative sum of concave/piecewise linear functions is also a concave/piecewise linear function. Based on this fact and that \( e(F_{K-1}) > 0 \), and \( \Delta(F_{K-1}|F_{\sigma^*(1)}, \ldots, F_{\sigma^*(K-2)}, \mathcal{C}) \) is a non-negative vector, implies that the function \( \hat{A}_{K-2}(\varpi) \) is concave and piecewise linear. Concavity also assures the continuity of this function. Using similar arguments, the concavity, continuity and piecewise linearity of functions \( \hat{A}_k(\varpi), k = 0, \ldots, K - 3 \), can also be guaranteed. Thus, \( \hat{A}_k(\varpi) \) takes the form \( \hat{A}_k(\varpi) \triangleq \min_{F_{k+1} \in \mathcal{Z}_k} [\beta_{F_{k+1}}^k \varpi], k = 0, \ldots, K - 1 \), where the set \( \{\beta_{F_{k+1}}^k\}_{F_{k+1} \in \mathcal{Z}_k} \in \mathbb{R}^{1 \times L} \) of vectors represents its linear pieces. Finally, the feature \( F_{k+1} \) at which \( \hat{A}_k(\varpi) \) achieves its minimum is given by \( F_{\sigma^*(k+1)} \triangleq \arg \min_{F_{k+1} \in \mathcal{Z}_k} [\beta_{F_{k+1}}^k \varpi] \).

### 4.8.4 Proof of Theorem 4.2

At the final stage \( k = K \), \( \hat{J}_K(\varpi) = g(\varpi) = \min_j [Q_j^T \varpi] \). Hence, \( \{\alpha^i_K\} \triangleq \{Q_j^T\}_{j=0}^L \). Using the facts that i) \( g(\varpi) \) and \( \hat{A}_k(\varpi) \) are concave and piecewise linear with respect to \( \varpi \), ii) \( \hat{J}_k(\varpi) = \min[g(\varpi), \hat{A}_k(\varpi)], k \in \{0, \ldots, K - 1\} \) (see Theorem 4.1), and iii) the minimum of two concave/piecewise linear functions is also a concave/piecewise linear function, implies that the function \( \hat{J}_k(\varpi) \) is concave and piecewise linear. Since \( g(\varpi) = \min_j [Q_j^T \varpi] \) (see Lemma 4.1) and \( \hat{A}_k(\varpi) = \beta_{F_{\sigma^*(k+1)}}^k \varpi \) (see Lemma 4.2), \( J_k(\varpi) \) takes the form \( J_k(\varpi) = \min_i \alpha^i_k \varpi \), where the set \( \{\alpha^i_k\} \triangleq \{[\beta_{F_{\sigma^*(k+1)}}^k] \cup \{Q_j^T\}_{j=1}^L\}, k \in \{0, \ldots, K - 1\} \) of vectors represents its linear pieces.
4.8.5 Proof of Theorem 4.3

At the final stage \( k = K \), the only remaining expected cost is the optimum misclassification cost, which is \( \hat{J}_K(\Phi_{\sigma^*(K)}(c)) = g(\Phi_{\sigma^*(K)}(c)) \). At any intermediate stage \( k = 0, 1, \ldots, K-1 \), the decision maker can either incur \( g(\Phi_{\sigma^*(k)}(c)) \) and stop the feature evaluation process, or continue with the next optimum feature \( F_{\sigma^*(k+1)} \) and incur cost \( c(F_{\sigma^*(k+1)}) + \hat{J}_{k+1}(\Phi_{\sigma^*(k+1)}(c)) \), which corresponds to the cost of evaluating feature \( F_{\sigma^*(k+1)} \) and an additional cost of continuing optimally at stage \( k + 1 \). However, at stage \( k \), the optimum feature \( F_{\sigma^*(k+1)} \) to evaluate and its assignment \( f_{\sigma^*(k+1)} \) are not known. For any \( F_{k+1} \), the expected optimum cost-to-go is \( e(F_{k+1}) + E\{\hat{J}_{k+1}(\Phi_{k+1}(c))|\Phi_{\sigma^*(k)}(c)\} \), where \( \Phi_{k+1}(c) = P(c|F_{\sigma^*(1)}, \ldots, F_{\sigma^*(k)}, F_{k+1}) \). The next optimum feature \( F_{\sigma^*(k+1)} \) must be selected from the set \( Z_k \) of remaining features, such that the total cost until termination is minimized. In other words, the optimum cost-to-go \( \hat{A}_k(\Phi_{\sigma^*(k)}(c)) \) is:

\[
\hat{A}_k(\Phi_{\sigma^*(k)}(c)) = \min_{F_{k+1}} \left[ e(F_{k+1}) + E\{\hat{J}_{k+1}(\Phi_{k+1}(c))|\Phi_{\sigma^*(k)}(c)\} \right].
\]

Using Bayes’ rule to express \( \Phi_{k+1} \) in terms of \( \Phi_{\sigma^*(k)} \), and by the definition of the expectation operator:

\[
\begin{aligned}
\hat{A}_k(\Phi_{\sigma^*(k)}(c)) &= \min_{F_{k+1}} \left[ e(F_{k+1}) + \int_{F_{k+1}} P(f_{k+1}|F_{\sigma^*(1)}, \ldots, F_{\sigma^*(k)}) \right.
\times \hat{J}_{k+1} \left( \frac{P(f_{k+1}|F_{\sigma^*(1)}, \ldots, F_{\sigma^*(k)}, c)\Phi_{\sigma^*(k)}(c)}{\int_{c} P(f_{k+1}|F_{\sigma^*(1)}, \ldots, F_{\sigma^*(k)}, c)\Phi_{\sigma^*(k)}(c) dc} \right) df_{k+1} \bigg].
\end{aligned}
\]

(4.30)

Note that the probability \( P(f_{k+1}|F_{\sigma^*(1)}, \ldots, F_{\sigma^*(k)}) \) can be simplified as follows:

\[
P(f_{k+1}|F_{\sigma^*(1)}, \ldots, F_{\sigma^*(k)}) = \frac{P(f_{k+1}, F_{\sigma^*(1)}, \ldots, F_{\sigma^*(k)})}{P(F_{\sigma^*(1)}, \ldots, F_{\sigma^*(k)})}
= \int_{c} P(f_{k+1}, F_{\sigma^*(1)}, \ldots, F_{\sigma^*(k)}, c)P(c) dc 
P(F_{\sigma^*(1)}, \ldots, F_{\sigma^*(k)})
= \int_{c} P(f_{k+1}|F_{\sigma^*(1)}, \ldots, F_{\sigma^*(k)}, c)P(F_{\sigma^*(1)}, \ldots, F_{\sigma^*(k)}) dc 
P(F_{\sigma^*(1)}, \ldots, F_{\sigma^*(k)})
= \int_{c} P(f_{k+1}|F_{\sigma^*(1), \ldots, F_{\sigma^*(k)}}, c) \frac{P(c, F_{\sigma^*(1)}, \ldots, F_{\sigma^*(k)})}{P(F_{\sigma^*(1)}, \ldots, F_{\sigma^*(k)})} dc
= \int_{c} P(f_{k+1}|F_{\sigma^*(1), \ldots, F_{\sigma^*(k)}}, c)\Phi_{\sigma^*(k)}(c) dc.
\]

(4.31)
The desired result can be obtained by substituting the result in Eq. (4.31) to Eq. (4.30) as follows:

\[
\hat{A}_k(\Phi_{\sigma^*(k)}(c)) = \min_{F_{k+1} \in Z_k} \left[ e(F_{k+1}) + \int_{F_{k+1}} \int_C P(f_{k+1}|F_{\sigma^*(1)}, \ldots, F_{\sigma^*(k)}, c)\Phi_{\sigma^*(k)}(c)dc \right. \\
\times \left. \hat{J}_{k+1} \left( \frac{P(f_{k+1}|F_{\sigma^*(1)}, \ldots, F_{\sigma^*(k)}, c)\Phi_{\sigma^*(k)}(c)}{\int_C P(f_{k+1}|F_{\sigma^*(1)}, \ldots, F_{\sigma^*(k)}, c)\Phi_{\sigma^*(k)}(c)dc} \right) df_{k+1} \right], 
\]

(4.32)

which completes the proof.
CHAPTER 5
Dynamic Instance–Wise Feature Selection for Model Interpretation

“Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead” – Cynthia Rudin (2021 Squirrel AI Award Winner)

5.1 Introduction

Recent advances in machine learning set paths to complex function approximators that can achieve high performance in many domains [26, 55]. However, humans are often reluctant to deploy such complex models in practice, particularly in health care, criminal justice, and financial markets, since they do not have formal justifications about what the model is doing and why it outputs specific classification decisions [3, 57]. Due to this reason, methods that explain black–box models have been proposed [43, 56]. Nevertheless, these explainable methods are often not reliable and can be misleading [57].

Inherently interpretable machine learning models can be used to discover relevant knowledge about domain relationships in data, debug or justify the model and its outputs, and control and improve the model [46, 45]. Examples include the generalized additive models (GAMs) [25] and the decision tree. GAMs combine single–feature models through a linear function, identifying the contribution of individual features to the model output. Common GAMs include the logistic regression and the explainable boosting machines, which use linear and boosted decision tree shape functions [42].

Using a sparse set of features to classify data instances is essential for model interpretability [45], since we can explicitly observe which features contribute to each model output. However, in standard settings, sparsity is achieved globally by incorporating a regularizer to the model parameters (e.g., GAMs with L1–norm regularizer), where the same subset of features is used to classify all test instances. In contrast, the decision tree achieves instance–level sparsity by evaluating features along different decision paths, using different
features to classify different test instances. Nonetheless, it uses a greedy approach to build
the tree structure, where locally optimal splits are obtained at every tree node [19], hence,
using more features than necessary.

Chapter 4 proposed an algorithm for Instance–wise Feature selection and Classification
with optimum feature Ordering (IFCO), which dynamically selects both the order and the
number of features to classify each data instance individually when features sequentially
arrive one at a time during testing. Herein, the model–based and post hoc interpretability
of IFCO is justified. Experimental results on a credit risk classification task show that IFCO
can be used in high–stakes applications, where model interpretations are required without
sacrificing test accuracy.

5.2 Background

In chapter 4, we proposed to solve the following optimization problem:

\[
\min_{\sigma,\sigma(R),D_{\sigma(R)}} \mathbb{E} \left\{ \sum_{k=1}^{R} e(F_{\sigma(k)}) + \mathcal{L}(D_{\sigma(R)}) \right\}, 
\]  

(5.1)

where \( \sigma \) denotes a permutation of the features, \( \sigma(R) \) denotes the number of features evaluated before the framework terminates assuming feature ordering \( \sigma \), and \( D_{\sigma(R)} \) denotes the classification rule used. The term \( e(F_k) > 0, k \in \{1, \ldots, K\} \), denotes the cost of acquiring feature \( F_k \), and \( \mathcal{L}(D_{\sigma(R)}) = \sum_{j=1}^{L} \sum_{i=1}^{L} Q_{ij} P(D_{\sigma(R)} = j, \mathcal{C} = c_i) \) is the cost associated with the classification rule \( D_{\sigma(R)} \), where \( c_i \) denotes an assignment to class variable \( \mathcal{C} \), and \( Q_{ij} \geq 0 \)
represents the cost of selecting class \( c_j \) when the true class is \( c_i \), \( i, j \in \{1, \ldots, L\} \). The optimum classification strategy \( D^{*}_{\sigma(R)} \) for any number \( \sigma(R) \) and ordering \( \sigma \) was shown to be:

\[
D^{*}_{\sigma(R)} = \arg \min_{1 \leq j \leq L} \left[ Q_{j}^T \pi_{\sigma(R)} \right],
\]  

(5.2)

where \( Q_j \triangleq [Q_{1,j}, Q_{2,j}, \ldots, Q_{L,j}]^T \), \( \pi_{\sigma(k)} \triangleq [\pi^1_{\sigma(k)}, \pi^2_{\sigma(k)}, \ldots, \pi^L_{\sigma(k)}]^T \), and \( \pi^i_{\sigma(k)} \triangleq P(\mathcal{C} = c_i | F_{\sigma(1)}, \ldots, F_{\sigma(k)}) \). The posterior probability vector \( \pi_{\sigma(k)} \in [0,1]^L \) is updated recursively
via Bayes’ rule as follows:

$$\pi_{\sigma(k)} = \frac{\text{diag} \left( \Delta \left( F_{\sigma(k)} \| F_{\sigma(1)}, \ldots, F_{\sigma(k-1)}, C \right) \right) \pi_{\sigma(k-1)}}{\Delta^T \left( F_{\sigma(k)} \| F_{\sigma(1)}, \ldots, F_{\sigma(k-1)}, C \right) \pi_{\sigma(k-1)}},$$  \hspace{1cm} (5.3)

where $\Delta \left( F_{\sigma(k)} \| F_{\sigma(1)}, \ldots, F_{\sigma(k-1)}, C \right) \triangleq \left[ P(F_{\sigma(k)} \| F_{\sigma(1)}, \ldots, F_{\sigma(k-1)}, c_1), \ldots, P(F_{\sigma(k)} \| F_{\sigma(1)}, \ldots, F_{\sigma(k-1)}, c_L) \right]^T$, diag$(A)$ represents a diagonal matrix with elements of vector $A$, $\pi_{\sigma(0)} \triangleq [p_1, p_2, \ldots, p_L]^T$, and $p_i = P(C = c_i)$. The optimum ordering $\sigma^*$ and the optimum number $\sigma^*(R^*)$ of features was then derived using the following dynamic programming equations:

$$\widehat{J}_k(\pi_{\sigma^*(k)}) = \min \left[ g(\pi_{\sigma^*(k)}), \widehat{A}_k(\pi_{\sigma^*(k)}) \right], k = 0, \ldots, K - 1,$$

$$\widehat{J}_K(\pi_{\sigma^*(K)}) = g(\pi_{\sigma^*(K)}),$$  \hspace{1cm} (5.4)

where $\widehat{A}_k(\pi_{\sigma^*(k)}) \triangleq \min_{F_k \in Z_k} \left[ e(F_{k+1}) + \sum_{F_k} \Delta^T \left( F_{\sigma^*(1)} \| F_{\sigma^*(k)}, C \right) \pi_{\sigma^*(k)} \widehat{J}_{k+1}(\pi_{\sigma^*(k+1)}) \right]$, $g(\pi_{\sigma^*(R)}) \triangleq \min_j Q_j^T \pi_{\sigma^*(R)}$, and $Z_k$ is the set of remaining features at stage $k$. The optimum number $\sigma^*(R^*)$ of features is equal to the first $k < K$ features for which $g(\pi_{\sigma^*(k)}) \leq \widehat{A}_k(\pi_{\sigma^*(k)})$, or $\sigma^*(R^* = K)$ if there are no more features to be evaluated.

Finally, we proposed IFCO using the fact that $g(\pi_{\sigma^*(k)})$ and $\widehat{A}_k(\pi_{\sigma^*(k)})$ are continuous, concave, and piece–wise linear functions. To classify a test instance, IFCO starts by setting $k = 0$ and assigning the prior distribution of the class variable $C$ to the posterior probability $\pi_{\sigma^*(0)}$. If $\widehat{J}_0(\pi_{\sigma^*(0)})$ belongs to a hyperplane of $g(\pi_{\sigma^*(0)})$, IFCO stops and classifies the instance using Eq. (5.2). Otherwise, it evaluates the feature associated with the hyperplane of $\widehat{A}_0(\pi_{\sigma^*(0)})$, increments $k$ by 1, and updates the posterior probability $\pi_{\sigma^*(0)}$ using Eq. (5.3). IFCO repeats these steps until it stops or there are no more features available (see Chapter 4 for more details).

### 5.3 Interpretability

This section justifies the model–based and post hoc interpretability of IFCO. The former demonstrates the glass–box nature of IFCO, such that a human can understand how it behaves and which factors influence its decision–making process. The latter analyzes the relationships that IFCO has learned from a given dataset [46].

We consider logistic regression (LR) with L1–norm regularizer, and decision tree (DT)
Table 5.1: Credit risk dataset features.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>F₁</td>
<td>Checking account status</td>
<td>F₁₁</td>
<td>Present residence</td>
</tr>
<tr>
<td>F₂</td>
<td>Duration in months</td>
<td>F₁₂</td>
<td>Property</td>
</tr>
<tr>
<td>F₃</td>
<td>Credit history</td>
<td>F₁₃</td>
<td>Age in years</td>
</tr>
<tr>
<td>F₄</td>
<td>Purpose of the credit</td>
<td>F₁₄</td>
<td>Other installment plans</td>
</tr>
<tr>
<td>F₅</td>
<td>Credit amount</td>
<td>F₁₅</td>
<td>Housing</td>
</tr>
<tr>
<td>F₆</td>
<td>Savings account status</td>
<td>F₁₆</td>
<td>Existing credits</td>
</tr>
<tr>
<td>F₇</td>
<td>Present employment (years)</td>
<td>F₁₇</td>
<td>Job</td>
</tr>
<tr>
<td>F₈</td>
<td>Installment rate</td>
<td>F₁₈</td>
<td>Number of dependents</td>
</tr>
<tr>
<td>F₉</td>
<td>Personal status</td>
<td>F₁₉</td>
<td>Telephone</td>
</tr>
<tr>
<td>F₁₀</td>
<td>Other debtors</td>
<td>F₂₀</td>
<td>Foreign worker</td>
</tr>
</tbody>
</table>

As baselines, since they are well-studied machine learning models that are interpretable on a modular level [45]. For demonstration purposes, we use the German credit risk dataset [15]. The goal is to classify people as high or low credit risk based on a set of 20 features (see Table 5.1).

5.3.1 Model–Based Interpretability

Model–based interpretability considers three criteria: sparsity, simulatability, and modularity [46]. Sparsity refers to using a sparse set of features to classify each data instance. Simulatability represents the ability to simulate and reason about the entire decision–making process. Modularity denotes the ability to interpret the meaningful portions of the decision–making process independently.

5.3.1.1 Sparsity

IFCO imposes instance–level sparsity by utilizing the feature evaluation cost, i.e., \( \sum_{k=1}^{R} e(F_{\sigma(k)}) \), in the optimization function in Eq. (5.1). Specifically, evaluating features in different orderings \( \sigma \) and terminating at different number \( \sigma(R) \) of features results in different accumulated costs. By penalizing these accumulated feature evaluation costs, IFCO optimizes the number of features used to classify individual data instances. It also uses a varying number of features to classify different data instances.

Similar to IFCO, DT uses a varying number of features to classify different data instances by evaluating features along different decision paths in the tree. In contrast, LR
imposes global sparsity by utilizing the L1–norm penalty on the model parameters in the optimization function. Global sparsity degrades the model’s interpretability since it uses the same subset of features to classify all test instances. However, before interpreting a sparse solution, the stability of the sparsity should be validated [46]. In particular, if the sparsity varies drastically due to a small perturbation in the training dataset, the resulting interpretations are meaningless. Herein, we train 100 instances of each machine learning model using 100 bootstraps of the training data and observe the variation of the number of features used to classify a fixed set of test instances (see Fig. 5.1). Observe that the variation in the number of features used to classify the same test instance is less than one feature with 95% confidence. Therefore, the instance–level sparsity of IFCO and DT and the global sparsity of LR are stable to small perturbations in the training dataset.

Figure 5.1: Number of features used to classify ten random test instances in the credit risk dataset. Range corresponds to the 95% confidence interval.
5.3.1.2 Simulatability

A human can simulate and reason about IFCO’s entire decision–making process. The functions $\hat{J}_k(\pi_{\sigma^*(k)}), k = 0, \ldots, K,$ in Eq. (5.4) can be decomposed into linear hyperplanes because $g(\pi_{\sigma^*(k)})$ and $\hat{A}_k(\pi_{\sigma^*(k)})$ are continuous, concave, and piece–wise linear [38]. Hence, there are unique regions in the domain of $\hat{J}_k(\pi_{\sigma^*(k)}),$ i.e., the posterior probability space generated by $\pi_{\sigma^*(k)},$ that determine whether to continue and evaluate a specific feature or stop the decision–making process. Fig. 5.2 illustrates this decision–making process for the credit risk dataset. At stage $k = 0,$ IFCO starts by assigning the prior probability $P(credit\ risk = high)$ to the posterior probability $\pi_{\sigma^*(0)}.$ If this probability falls in the blue region, IFCO evaluates $F_1.$ If it falls in the orange region, IFCO evaluates $F_6.$ Otherwise, IFCO stops and reaches a decision, i.e., the person is classified as low or high credit risk if this probability falls in the green or red region, respectively. If IFCO decides to evaluate a feature, it updates the posterior probability $\pi_{\sigma^*(0)}$ using Eq. (5.3) and continues the decision–making process similarly in the next stage, as shown in Fig. 5.2.

DT is also a simulatable model due to its hierarchical decision–making process [46]. Each node compares the feature value with a fixed threshold and decides whether to follow the left or right branch. Final decisions are at the leaves. However, DT uses a greedy approach to learn the tree structure, where locally optimal splits are obtained at every tree node [19]. In contrast, IFCO optimizes the instance–wise feature ordering, hence uses fewer features on average to classify data instances compared to DT (see Table 5.2). In LR, a human only needs to compute the dot product between the feature vector and the corresponding weight vector to obtain a classification decision. Each weight is proportionate to the effect of the corresponding feature on the class variable when the rest features are kept unchanged.

5.3.1.3 Modularity

IFCO enforces modularity by employing a sequential decision–making process based on a sufficient statistic, i.e., the posterior probability vector $\pi_{\sigma^*(k)},$ that is recursively updated as seen in Eq. (5.3). At each stage $k,$ the only information required for this update is the observation vector $\Delta(F_{\sigma^*(k)}|F_{\sigma^*(1)}, \ldots, F_{\sigma^*(k-1)}, C) = [P(F_{\sigma^*(k)}|F_{\sigma^*(1)}, \ldots, F_{\sigma^*(k-1)}, c_1), \ldots, P(F_{\sigma^*(k)}|F_{\sigma^*(1)}, \ldots, F_{\sigma^*(k-1)}, c_L)]^T.$ In other words, the posterior probability decomposes into the probability of each feature given the already evaluated features and the class vari-
Figure 5.2: First four stages of the IFCO’s decision making process for the credit risk dataset.

Probabilistic models can enforce modularity by specifying a conditional independence structure, making it easier to reason about different parts of a model independently [46]. IFCO adopts such an assumption, which simplifies the observation vector to \( \Delta(F_{\sigma^*(k)}|C) = [P(F_{\sigma^*(k)}|c_1), \ldots, P(F_{\sigma^*(k)}|c_L)]^T \). This assumption helps to decompose the posterior probability into simple and meaningful portions in terms of the probability of each feature given the class variable. It also speeds up computations and enables prediction-level interpretations.

LR inherits modularity by having a decision function based on an affine transformation of the input feature space [25], while each node in DT can be viewed as a modular block that contributes to the final classification decision.

5.3.2 Post Hoc Interpretability

This section analyzes the dataset- and prediction-level information learned by IFCO about the credit risk dataset.
Figure 5.3: Partial dependence plots of features (a) savings account status, $PD(F_6)$, and (b) age, $PD(F_{13})$, using IFCO. savings account status $\in \{0, 1, \ldots, 4\}$ and age is in years. The dark blue line represents the mean, while the light blue region represents the 95% confidence interval.

5.3.2.1 Dataset–level Interpretations

**Partial Dependence**: Partial dependence captures the marginal effects of an individual feature on the output of a machine learning model [20]. Specifically, for feature $F_i$, the partial dependence function is approximated by $PD(F_i) \approx \frac{1}{N} \sum_{n=1}^{N} \hat{f}(F_i, \bar{F}^{(n)}_i)$, where $\hat{f}$ is the model output, $\bar{F}^{(n)}_i$ is the $n$th training instance without feature $F_i$, and $N$ is the total number of training instances. Fig. 5.3 shows the partial dependence functions of two features on the probability of credit risk being high. Observe that the probability of credit risk being high decreases as the feature savings account status increases. Feature savings account status = 4 represents “no known savings accounts” [15], which increases credit risk. In contrast, feature age does not seem to affect credit risk.

**Feature Importance**: IFCO computes feature importance by averaging the number of times each feature contributes to a particular classification decision. Fig. 5.4 shows the feature importance for each feature in the credit risk dataset. Note that both IFCO and LR choose checking account status and credit history as the top two important features. On the other hand, DT selects credit amount as the most important feature, which ranks fourth according to IFCO.
Figure 5.4: Normalized feature importance for credit risk dataset.

Accuracy Stability: Test accuracy should be stable for any perturbations in the training data [45]. Herein, we compute out–of–sample accuracy using 10–fold cross–validation (see Table 5.2). In addition to interpretable machine learning models, we also consider a black–box model, i.e., gradient boosted trees (XGB) [10], which provides an upper bound on the achievable accuracy. The deviation in the out–of–sample accuracy achieved by IFCO is ±0.04 over the 10–folds. Hence, the accuracy is stable to any perturbations in the data. Note that XGB achieves the highest accuracy (i.e., only 0.1% better), but requires 3.4 times more features than IFCO.

5.3.2.2 Prediction–level Interpretations

This section analyzes the ability of IFCO to provide prediction–level interpretations using 4 test instances of the credit risk dataset. Recall that IFCO assigns the data instance to the class with the highest posterior probability (see Chapter 4). Fig. 5.5(a) represents a correctly predicted low credit risk instance, where IFCO evaluates features $F_1$, $F_3$, and
Table 5.2: Accuracy (± standard deviation) and average number of features (Feat.) used in the credit risk dataset.

<table>
<thead>
<tr>
<th>Method</th>
<th>Accuracy</th>
<th>Feat.</th>
</tr>
</thead>
<tbody>
<tr>
<td>IFCO</td>
<td>0.754±0.040</td>
<td>5.85</td>
</tr>
<tr>
<td>LR</td>
<td>0.740±0.034</td>
<td>14.0</td>
</tr>
<tr>
<td>DT</td>
<td>0.702±0.044</td>
<td>6.78</td>
</tr>
<tr>
<td>XGB</td>
<td>0.755±0.037</td>
<td>19.9</td>
</tr>
</tbody>
</table>

$F_6$, representing that the person has a bad checking account status, a good credit history, and a good savings account status, respectively. Note that having a bad checking account status is discounted by having a good credit history and a good savings account status. Fig. 5.5(c) represents a correctly predicted high credit risk instance, where IFCO evaluates features $F_1$, $F_3$, $F_2$, and $F_{12}$, representing that the person has a bad checking account status, a bad credit history, a credit history of 36 months, and no known property, respectively. Fig. 5.5(b) represents a high credit risk instance incorrectly predicted as low, where IFCO evaluates features $F_1$, $F_3$, $F_6$, $F_2$, $F_5$, $F_{20}$, and $F_{12}$, representing that the person has a bad checking account status, a good credit history, a bad savings account status, a credit history of 12 months, an existing credit amount of 1056, is a foreign worker, and owns real state property, respectively. Having a bad checking and savings account status is discounted by having a good credit history with low existing credit amount and owning real state property, hence incorrectly predicting the person to be low credit risk. Fig. 5.5(d) represents a low credit risk instance predicted as high, where IFCO evaluates features $F_1$, $F_3$, $F_2$, and $F_5$, representing that the person has a bad checking account status, a moderate credit history, a credit history of 39 months, and an existing credit amount of 11,760, respectively. The high credit amount within a short credit length and the bad checking account status justify IFCO’s high–risk prediction.

5.4 Concluding Remarks

This chapter discusses the glass–box nature of IFCO, an algorithm that dynamically selects both the order and the number of features to individually classify each data instance when features sequentially arrive one at a time during model testing. Model–based interpretability is demonstrated using instance–level sparsity, simulatability, and modularity. Dataset– and prediction–level interpretations are analyzed during the post hoc stage.
Experimental results validate that IFCO outperforms state–of–the–art interpretable models while applicable in high-stakes applications, where model interpretations are necessary.
CHAPTER 6
Conclusions and Future Work

In this Chapter, we summarize the work presented in this thesis and outline various future directions.

6.1 Summary

We started this dissertation by formally introducing the problem of dynamic instance-wise decision-making and highlighting the key challenges associated with it. In brief, the problem is to optimize the trade-off between data instance-level sparsity and the quality of prediction in a dynamic setting to reduce the total cost associated with evaluating costly features and improve the accuracy and interpretability of the final decisions.

In Chapter 2, we considered a simplified version of this problem, assuming features are conditionally independent given the target variable, and evaluated in a fixed ordering common to all the test instances to predict a discrete target variable. We derived the optimum solution using dynamic programming. This solution has an intuitive structure, where at each stage, the proposed method faces two options given the posterior probability of the target variable: i. stop evaluating features and select optimally between the potential targets, or ii. continue with the next feature. We proposed ETANA, an algorithm that implements the brute-force solution. We showed that the optimum solution has interesting mathematical properties that split the posterior probability space into regions that determine whether to continue or stop the feature evaluation process. We proposed F-ETANA, a fast implementation of the optimum solution that approximates these splits using linear hyperplanes. This efficient implementation scales up to millions of features without sacrificing accuracy. Further, we derived the expected number of features required to achieve a given classification accuracy. This result is beneficial to approximate the budget in advance for applications involving high-cost features (e.g., medical diagnosis). The proposed algorithms, ETANA and F-ETANA, perform well on various public datasets, with comparable and often superior performance compared to prior work.
Next, in Chapter 3, we relaxed the conditional independence assumption by modeling the feature dependencies using a Bayesian network. We proposed a feature ordering such that each selected feature contains the maximum possible new information about the target variable compared to the already evaluated feature set. We derived the optimum solution using dynamic programming. We showed that the functions related to the cost of stopping and continuing are concave, continuous, and piecewise linear. We proposed IFC\(^2\)F, an algorithm that implements the optimum solution efficiently by approximating these linear pieces using a modified point–based value iteration algorithm. We illustrated the effectiveness and scalability of IFC\(^2\)F on various real–world datasets. IFC\(^2\)F robustly performs well on all of them, with comparable and often superior performance compared to the prior art.

Chapter 4 considered the more general problem of predicting both discrete and continuous target variables (i.e., both classification and regression problems) and determining the order by which features must be reviewed. In the classification setting, we considered a cost function in terms of fixed misclassification costs. We proved that the functions related to the optimum solution are concave, continuous, and piecewise linear. We presented IFCO, a routine that implements the optimum solution utilizing these structural properties. In the regression setting, we considered the mean squared error cost function. We modeled the continuous target variable using a Gaussian mixture model to avoid the overhead in computing integrals in the optimum solution. We presented IFRO, a routine that implements the optimum solution utilizing the Gaussian mixture assumption. We illustrated the effectiveness of IFCO and IFRO on various real–world datasets. IFCO and IFRO robustly perform well on all of them, with comparable and often superior performance than the prior art.

Finally, in Chapter 5, we justified the glass–box nature of IFCO. Specifically, we discussed IFCO’s model–based and post hoc interpretability. Model–based interpretability is discussed in terms of sparsity, simulatablity, and modularity. During the post hoc stage, we showed dataset– and prediction–level interpretations using an example credit risk prediction problem. Experimental results showed IFCO’s applicability in high–stakes applications, where model interpretations are required without sacrificing accuracy.
6.2 Future Directions

As discussed in previous chapters, dynamic instance–wise decision–making is essential to automate tasks from many domains, such as health care, criminal justice, financial markets, and disaster prediction. Still, there exist challenges associated with this problem that are potentially interesting future research directions.

Similar to features, training data instances are not freely available in many domains. For example, in credit risk prediction problems, acquiring individuals’ credit data from sources such as Experian is expensive [30]. In such applications, we can not guarantee that the training set is available at once before model training. Retraining the model whenever new training instances arrive is not feasible. Online learning methods provide a partial solution to such problems assuming that training instances come sequentially in a supervised classification setting and are used to update the function of a classifier iteratively [27]. Selecting the optimum training instances to learn a generalized model efficiently and combining it with a dynamic instance–wise decision–making framework can address a broad spectrum of applications.

Throughout this dissertation, we considered a single–label prediction problem, where we assume a feature vector is associated with a single target variable. Multi–label classification instead assumes an instance may be associated with multiple labels [63]. Multi–label context is getting much attention due to its applicability in various domains, including text classification [37] and health care [72]. For instance, in medical diagnosis, a patient may suffer from multiple diseases (i.e., multiple labels associated with a single data instance), and the doctor’s goal is to identify those diseases simultaneously by conducting an optimum set of medical tests (i.e., features) considering the cost of medical tests and the time–sensitivity of decisions. The most common approach to solve such problems is to consider labels separately and model them as separate classification tasks [63]. This approach has several drawbacks; it is computationally inefficient, does not capture dependencies among labels, and, more importantly, is not cost–effective when features incur high costs (e.g., medical domain). Therefore, extending dynamic instance–wise decision–making framework in multi–label settings is an important research direction.

The proposed dynamic instance–wise decision–making framework requires labeled data to train. However, labeled data instances are often challenging, expensive, or time–consuming
to obtain since it requires slow human annotation and expensive laboratory experiments [77]. These restrictions with labeling often result in a scarce of labeled data and a surplus of unlabeled data. Semi-supervised machine learning methods address this problem by combining a large amount of unlabeled data with the available labeled data to learn robust classifiers. The most popular approach is to use generative mixture models with an expectation-maximization algorithm to identify mixture components using unlabeled data and fully determine the mixture distribution using labeled data [77]. Such an approach can be deployed in a dynamic instance-wise decision-making framework to handle semi-supervised problems (recall, in Chapter 4, we modeled the target variable using a Gaussian mixture model).

Causal reasoning is necessary for machine learning models to achieve the highest level of interpretability [51]. In Chapter 5, we focused on statistical interpretability using observational data. Causal interpretability instead aims to provide causal interventional and counterfactual explanations based on imaginary scenarios not explicitly visible on observational data. In causal interventional studies, we force treatment variables to predefined values to observe the effect on the target variable. For instance, consider a set of treatments conducted by a doctor to diagnose a patient. In such cases, given the cost of medical treatments and the patient’s condition, the doctor requires to conduct the treatments in an optimal way to recover the patient’s health condition as quickly as possible. On the other hand, counterfactual interpretability investigates for minimal changes required to revert the outcome of a machine learning model. For example, what are the minimal changes required on the medical test report that would have detected a cardiovascular disease of a healthy patient? In this thesis, we showed that dynamic instance-wise decision-making is essential for such high-stakes decisions. Therefore, bridging the gap between dynamic instance-wise decision-making and casual interpretability is the next-generation machine learning to achieve human-level intelligence.
BIBLIOGRAPHY


[34] David P Kao, James D Lewsey, Inder S Anand, Barry M Massie, Michael R Zile, Peter E Carson, Robert S McKelvie, Michel Komajda, John JV McMurray, and JoAnn


[67] Yue Wu, Steven CH Hoi, Tao Mei, and Nenghai Yu, *Large-scale online feature selection for ultra-high dimensional sparse data*, ACM Transactions on Knowledge Discovery from Data (TKDD) **11** (2017), no. 4, 48.


