2018

Methods for Online Feature Selection for Classification Problems

Alaleh Razmjoo
University of Central Florida

METHODS FOR ONLINE FEATURE SELECTION FOR CLASSIFICATION PROBLEMS

by

ALALEH RAZMJOO
M.Sc. Semnan University, 2012
B.Sc. Damghan University, 2009

A dissertation submitted in partial fulfilment of the requirements
for the degree of Doctor of Philosophy
in the Department of Industrial Engineering and Management Systems
in the College of Engineering and Computer Science
at the University of Central Florida
Orlando, Florida

Summer Term
2018

Major Professor: Qipeng Phil Zheng

ABSTRACT

Online learning is a growing branch of machine learning which allows all traditional data mining techniques to be applied on an online stream of data in real-time. In this dissertation, we present three efficient algorithms for feature ranking in online classification problems. Each of the methods are tailored to work well with different types of classification tasks and have different advantages. The reason for this variety of algorithms is that like other machine learning solutions, there is usually no algorithm which works well for all types of tasks. The first method, is an online sensitivity based feature ranking (SFR) which is updated incrementally, and is designed for classification tasks with continuous features. We take advantage of the concept of global sensitivity and rank features based on their impact on the outcome of the classification model. In the feature selection part, we use a two-stage filtering method in order to first eliminate highly correlated and redundant features and then eliminate irrelevant features in the second stage. One important advantage of our algorithm is its generality, which means the method works for correlated feature spaces without preprocessing. It can be implemented along with any single-pass online classification method with separating hyperplane such as SVMs. In the second method, with help of probability theory we propose an algorithm which measures the importance of the features by observing the changes in label prediction in case of feature substitution. A non-parametric version of the proposed method is presented to eliminate the distribution type assumptions. These methods are application to all data types including mixed feature spaces. At last, we present a class-based feature importance ranking method which evaluates the importance of each feature for each class, these sub-rankings are further exploited to train an ensemble of classifiers. The proposed methods will be thoroughly tested using benchmark datasets and the results will be discussed in the last chapter.

To my mother, who has been with me through highs and lows.

ACKNOWLEDGMENTS

I am using this opportunity to express my sincere gratuity to everyone who helped to reach my goals during my course of study from the beginning to this point.

I am thankful for the Dr. Waldemar Karwowski who put his trust in me and inspired me to explore new opportunities during my course of study at University of Central Florida.

I am also very much grateful for the guidance and help from my advisers, Dr. Qipeng Zheng, and Dr. Petros Xanthapolous. Their knowledge, positivity and forward thinking helped me to achieve my research goals.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: INTRODUCTION

Classification in Machine Learning

Machine learning has become a game changer in the way decision systems work in virtually every scientific discipline. Today, machine learning methods are used not only across the tech industry but in finance, medical sciences, psychology, engineering and so forth (Deo, 2015; Michalski et al., 2013; Bose and Mahapatra, 2001). Generally speaking machine learning algorithms could be categorized as unsupervised and supervised learning methods. Both of these categories could be used for pattern recognition and eventually prediction of future occurrences of a dependent variable. Unsupervised methods are applicable to clustering tasks where no target label is available to identify different categories in a corpus of training data. The data could be a bunch of images or online newspaper articles. The unsupervised learning methods such as k-means clustering could be then used to categorize the data into similar groups. One application of these algorithms are the recommender systems (Bobadilla et al., 2013).

On the other hand supervised learning methods are commonly used for labeled or annotated data to extract patterns or train predictive models. One major application of supervised methods is the use of classification models for predictive analysis. A classification or regression model is in fact a function of independent variables which are called *features* that is used to predict a dependent variable called *target*. The basic concept of a regression predictive model has been first introduced more that two centries ago Legendre (1805), however, its application to machine learning methods and computer-based pattern recognition is relatively new. The classification problem is defined as building a mathematical model to predict the target category of a occurrence based on the its corresponding observed features. The classification model is *trained* using available data and is tested to measure its prediction power or *accuracy*. The output of the model is translated to

the target label using an activation function such as sigmiod function. There are many different approaches to define and train a classifier. Some of the commonly used algorithms are listed below (Alpaydin, 2010).

- Fisher's linear discriminant: works by creating one or more linear combinations of predictors to classify two or more classes (McLachlan, 2004).
- Logistic regression: the binary logistic model is used to estimate the probability of a binary response based on one or more features used as predictors (Cox, 1958; Walker and Duncan, 1967).
- Naive Bayes classifier: these family of probabilistic models are based on applying Bayes theorem with strong independence assumptions between the features (Russell and Norvig, 2016).
- Perceptron: a classification algorithm that makes its predictions based on a linear predictor function combining a set of weights with the feature vector (Aizerman et al., 1964).
- Support vector machines: fitting a separating hyperplane by maximizing the separating margin of the classes (Suykens and Vandewalle, 1999; Joachims, 1998).
- k-nearest neighbor: categorize based on the classes of the similar datapoints based on a majority vote approach (Larose, 2005; Keller et al., 1985; Dudani, 1976).
- Decision trees: Classification by branching the data based on some type of information gain criteria (Quinlan, 1986; Kohavi and Quinlan, 2002; Biggs et al., 1991).
- Artificial Neural Network (ANN): a neural network is a layered architecture of nodes that perform arithmetic operations on the output of the previous layer. Different layers may perform different kinds of transformations on their inputs. The output of a ANN is used to predict the class of the target using an activation function (Zhang, 2000; Fausett, 1994).

Ensemble learning

Ensemble methods in machine learning are an alternative approach for model training. In this approach, instead of training a single classifier, a number of them are trained under specified circumstances. Empirical research show that an ensemble of classifiers improve the prediction power of the model in comparison with a single classifier. Beside improvement of the model accuracy, ensemble learning helps to reduce the effect of overfitting by training multiple classifiers at a time. One approach to train the ensemble is the use of a bagging strategy. In this way, using bootstrapping each classifier is trained using a subset of the training set. To deal with imbalanced datasets, these subsets could be deliberately balanced to improve the performance of the individual weak learners (Breiman, 1996; Prasad et al., 2006; Breiman, 2001).

In AdaBoost-based methods, the output of weak learners are combined into a weighted sum that represents the final output of the boosted classifier (Breiman et al., 1998; Hastie et al., 2009).

Online learning

With the growing trend in online data generation in many areas of science and technology there is a need for machine learning algorithms which could be trained incrementally and with minimum required computational capacities. To meet this need, online learning methods are presented to handle streams of data. The current major challenges with Big data streams are their volume, velocity and volatility. The first challenge is the volume of the data streams that arrives to the system along the time axis. In many cases this number grows possibly to infinity or to an unknown system termination point. When the data arrives at a high speed, velocity becomes a challenge because the data needs to processed at a high speed. Therefore, methods that requires high computational power seem hard to apply. At last, volatility comes from the nature of the dynamic data and dis-

tributions that change with time and in respond to internal or external factors. In developing and applying any online learning solution to real world problems these three challenge areas should be taken into consideration. Moreover, these types of algorithms are a proper alternative for very large data mining tasks where there is limited RAM. In these situations incremental algorithms become handy.

Examples of online learning applications includes: mining the data from sensors that record data in real-time and usually at a high speed such as traffic control sensors (Cohen et al., 2008), activity trackers such as bio medical devices and sport devices (Anguita et al., 2013), and streams of data produced on the World Wide Web (Gaber et al., 2005; Liu et al., 2015), to mention but a few.

One of the challenges in online learning is the problem of concept drift which is referred to the situations where the distribution of classification variables changes over time (Sayed-Mouchaweh, 2016; Hoffman et al., 2013). There are a great deal of examples for concept drift in online machine learning, in particular, in costumer behavior analysis tasks, such as changes in customer preferences and changes in potential costumer's characteristics in online marketing (Tsymbal, 2004). Therefore, in the development of online learning methods, it is important to include possible remedies to capture these changes.

There are also several software packages to facilitate development of online learning applications including MOA (Bifet et al., 2010) and Spark streaming (Zaharia et al., 2013; Bifet et al., 2010).

Preprocessing in online setting

Real world data is not always collected prepared for the machine learning models to work with. The collective steps to prepare the data to be used in machine learning models is called *preprocessing*. Common preprocessing steps for classification tasks are listed as following.

- Labeling missing values and type matching: at the vary early stage in a machine learning

pipeline, incorrect inputs and missing values need to be identified and labeled as NaN.

- OneHotencoding and/or feature binarization: in presence of categorical data, an ordered variable may be mistakenly taken as a continuous one in the training process. Therefore, One Hot endconding is used to produce binary features that implies existence of a category or lack of it by zero and ones. Binazization, on the other hand, is used to break a continuous feature into a categorical one.

- Imputation or removal of missing values: after identifying the missing values, imputation is used to fill in the gaps by commonly using mean, mode, or median of the variable.

- Scaling: difference in scales of the features may deteriorate the power of a model, therefore continuous features are usually scaled to a pre-defined range before the training process takes place.

- Feature engineering/feature selection: the last step in the preprocessing stage is the feature selection or introducing new features by manipulation of already existing ones. This steps is investigated in details in the rest of this dissertation.

Preprocessing in online learning has been the subject of many researches in the recent years. The challenges for preprocessing of the data streams boils down to the ever changing statistical characters of a streaming variable. Therefore, conventional methods for imputation, scaling and filter based feature selection or feature engineering are not applicable. Zliobaite and Gabrys (2014) have proposed a four sections reference framework for adaptive preprocessing of data streams. The sections are the categorization of preprocessing techniques, characterization of changing environment, characterization of adaptive learning process and the scenarios for interaction between adaptive predictors and adaptive preprocessing in a changing environment. Unlike offline machine learning model training, where preprocessing is done once with all the data present, the online

preprocessing shall be an automated procedure that is adaptive to the changes of the underlying environment (Krempl et al., 2014). The challenges in online preprocessing of data streams could be listed as follows:

- Incomplete input in the training or missing feature values.
- Imbalanced target appearance that happens when the class observations are not balanced.
- Handling Delayed Information that happens where the true label of some training instances does not arrive immediately therefore some type of delayed learning must take place.
- Concept drift: due to its importance, this subject will be discussed in details in future sections.

After dealing with data cleaning and preparation, it is time to decide what type of feature manipulation approach is appropriate for the problem. Feature manipulation can be roughly put in two categories of feature extraction and feature selection, which will be later discussed in details.

Dissertation Goals and Structure

As previously discussed, feature important ranking and selection plays an important role in machine learning applications. Also, with the growing number of devices that generate high volume of data, methods to capture the pattern of the data streams in real-time could help with future development of machine learning based applications and analysis. Currently there are a limited number of online feature importance ranking and feature selection algorithms which work with high reliability.

Unlike batch methods, conventional wrapper methods are not applicable to online applications because saving and retrieving the data is not computationally wise where there are high speed, high volume and high velocity data being generated. To address this issue, in this dissertation, three feature importance ranking and feature selection methods are proposed. Each of the developed algorithms have their own advantages and can be used in different learning environments which is discussed in each chapter, separately. The proposed methods could be used in both online and offline settings. Because of their efficiency and speed compared to batch feature selection methods, the proposed methods could be used for offline applications as well.

In Chapter 2, a detailed literature review of the topic online learning and feature selection in general machine learning applications is presented. Also, the current methodologies in online feature selection are discussed.

In Chapter 3, we present a fast and efficient online sensitivity based feature ranking method (SFR) which is updated incrementally. We take advantage of the concept of global sensitivity and rank features based on their impact on the outcome of the classification model. In the feature selection part, we use a two-stage filtering method in order to first eliminate highly correlated and redundant features and then eliminate irrelevant features in the second stage. One important advantage of our algorithm is its generality, which means the method works for correlated feature spaces without preprocessing. It can be implemented along with any single-pass online classification method with separating hyperplane such as SVMs.

In Chapter 4, we develop two methods for incremental ranking of features in classification tasks. Our ranking strategies are based on measuring the sensitivity of classification outcome with respect to individual features. The two methods work for different types of classification environments with discrete, continuous and mixed feature types with minimum prior assumptions. The second method which is a modification of the original method is designed to handle concept drift while

avoiding cumbersome computations.

In Chapter 5, we propose a class-based feature importance ranking method for online learning problems. The class-based feature ranking is technically producing multiple rankings of the features that will be eventually used for feature selection on an ensemble of classifiers. For each class of target, there will be one classifier present in the ensemble. The final output of the classifier is then calculated as the output of the ensemble. The feature ranking is based on the class-based incremental entropy measurements of the individual feature values. For continuous features, variance is used as a substituting measure for differential entropy.

All the methods are rigorously tested using benchmark datasets. Finally, a conclusion of the dissertation and possible future research directions are presented in Chapter 6.

CHAPTER 2: LITERATURE REVIEW

Incremental learning

In this chapter, we present a literature study on the applications of online learning as well as theoretical methods to train and test streams of data. Also, we present an overview and comparison of available feature selection methods from the literature. Finally, we discuss different approaches to online feature selection and the literature gap in handling different tasks where incremental feature importance ranking becomes necessary.

Applications of Online Learning

With the growing speed in producing streams of data in the recent decades and with the advent of Internet of Things (IoT), there are many research directions to capture the potential of data streams by applying real-time machine learning methodologies. Some areas where stream mining plays an essential role are the following:

- Online marketing data: Application of machine learning on online user data has lead to invention of many novel online marketing methods that have been used by large companies for monetization of user data streams by performing targeted advertisement (Jokinen et al., 2008; Deaton and Gabriel, 2001; Lazarus et al., 2000). In this context, click-through rate (CTR) is an important factor in measuring the effectiveness/attractiveness of online ads. Graepel et al. (2010) propose a Bayesian CTR prediction algorithm applied to the data of Sponsored Search in Microsoft search engine, Binge. Ngai et al. (2009) reviews the recent research papers concerning data analysis of Customer Identification, Customer Attraction,

Customer Retention and Customer Development. Many of the proposed approaches could be performed on online data of company customers or, in general, Web users.

- Stock market data: Abraham et al. (2001) suggests a neural network for one-day-ahead stock forecasting and a neuro-fuzzy system for analyzing the trend of the predicted stock values. Lunga and Marwala (2006) study the predictability of financial market moves with Learn++ by forecasting the daily movement direction of the Dow Jones. Bollen et al. (2011) has taken advantage of the Twitter text data for sentiment analysis and prediction of stock market trend.

- Internet traffic data: Roughan et al. (2004) propose a Statistical signature-based approach to IP traffic Classification. Tian et al. (2009) investigate the dynamic attributes of real-world traffic by considering concept drift from different angels i.e. overall traffic level and application level. Also, a SVM-based method for Internet traffic classification is proposed by Yuan et al. (2010).

- Health data: Banaee et al. (2013) provides a review of of the latest methods and algorithms used to analyze data from wearable sensors used for physiological monitoring of vital signs in health-care services. They investigate the commonly used data mining tools applied to problems such as anomaly detection, prediction and decision making when considering in particular continuous time series measurements. Wu et al. (2007) uses a naive Bayes classifier with Gaussian clustering to model patient motion with consideration of their medical condition, using streams of data generated by a wearable sensor system.

- Fraud detection: Ma et al. (2009) investigated online learning methods for detecting malicious websites, for example websites performing scam activities, using lexical and host-based features of the associated URLs. Halawa et al. (2017) develop an early warning system for the detection of suspicious account activity with the goal of quick identification and

remediation of compromised accounts.

- Text categorization: The automated categorization (or classification) of texts into predefined categories is an important problem in online learning. Sebastiani (2002) survey the challenges in document representation, classifier construction, and classifier evaluation. Ye et al. (2009) compared three supervised machine learning algorithms of naive Bayes, SVM and the character based N-gram model for sentiment classification of the reviews on travel blogs for several popular travel destinations around the world.

- Other applications: Huang and Xiao (2018) investigated different methodologies used for analyzing mobile user network log data for traffic sensing. Bruzzone and Prieto (1999) presents a new classification method for analysis of remote-sensing images. Their proposed classifier is based on Radial Basis Function (RBF) neural networks and utilizes an incremental training approach. Kim et al. (2000) proposed a real-time classification methodology for petroleum products based upon the near-infrared (NIR) spectra. The proposed real-time classifier (RTC) is designed based on the combination of principal component analysis (PCA) and a Bayesian classifier. Kargupta et al. (2004) proposed a mobile distributed data stream mining system for real-time vehicle monitoring.

Stochastic Gradient Descent and its extensions

Stochastic gradient descent (SGD) is a popular optimization approach due to its unique and simple iterative procedure which is easy to implement in many supervised and unsupervised learning methods. The stochastic process of SGD depends on random selection of examples at each iteration. In online learning setting, however, datapoints come in a continuing stream. Considering memory limitations, computational capabilities and time constraints, SGD can be used as an optimization tool for many data mining tasks. When we are talking about SGD, the first characteristic

that comes into mind is its simplicity compared to other analytical methods. In a general data mining problem we seek to minimize the cost function $Q(z; w) = l(fw(x); y)$ as the average loss over sample set. The simplicity of SGD comes from the idea of using a single example to update the weights w at each iteration. While in gradient descent algorithm, at each iteration an average of subgradient values on all sample data points is needed to update current values of w. Even though this may not seem to be substantial for small datasets, it can be computationally inefficient for large datasets. In addition to simplicity, SGD requires less memory compared to gradient descent since at each iteration we only need to have access to one or a few datapoints. This will even reduce networking costs between computation units and database servers. This attribute even becomes more important in online learning settings when sometimes datapoints arrive at a fast speed and in high volumes. The use of SGD makes it possible to discard datapoints once they are used in the updating process, hence, a huge amount of memory space can be saved. SGD plays an important role in many online data mining tasks as an incremental update approach where batch datasets do not exist. Despite advantages of SGD, it is not always a good idea to use it for optimization purposes due to its shortcomings in special types of data mining classes. Before deciding on using SGD, it is important to consider that similar to other approximation methods, SGD is sensitive to noises that are introduced by random selection. Experimental results show that shuffling data at each iteration could mitigate unwanted patterns in the data. As suggested by Bottou (2012), a wise selection of step size can also reduce the effects of noises, however, the final decision is still based on trial and error. One more case where SGD cannot work properly is when the classes of dataset are imbalanced. In this case, if random selection turns out to be excessively in favor of one class, the estimations will be biased toward the majority class. Classification of imbalanced datasets has always been challenging. The problem is even worse in online setting since we have no control over the stream of incoming datapoints. However there are some attempts to reduce the effects of imbalanced data in online learning. Nguyen et al. (2011) have proposed an online procedure to maintain class balance by discarding some datapoints of the majority class on a random basis. As

they suggest this method works well for Big data, where discarding a number of instances may not hurt too much. Even though they applied their method on Bayesian learning procedure, it can be also extended to SGD. Another related work is by Wang and Yao (2012) where they suggested a strategy to overcome imbalanced learning by buffering stream of data and randomly picking same size mini-batches to make the updates. Both these methods seem to work well with Big data streams where data points are abundant. However, in real-world problems sometimes data is not that cheap and we may not wish to ignore many of them. Take traffic sensor data, for instance, where traffic information such as volume, average speed, and congestion is recorded in 10 minute time intervals. Using these data to detect accident patterns is an imbalanced data mining task, since incident cases are very rare compared to non-incident cases. Using the available methods, we need to discard many of data points to control the balance which is not desired. Hence, one research opportunity is to develop an efficient online algorithm that can handle imbalanced datasets without a need for buffering or data selection. Weighted relaxed support vector machines (WRSVM) is a recently presented classification method which aims to handle imbalanced datasets by assigning proportional weights in objective function (Şeref et al., 2017). WRSVM can be categorized as a cost sensitive SVM and has some advantages compared to previously developed cost sensitive SVMs. For the case of online imbalanced datasets, WRSVM seems to be an appropriate classification method which provides the opportunity of updating weights with stream of data. In other word, by using WRSVM in an online setting we will be able to proportionally decrease effects of the majority class's datapoints in favor of the minority class without a need to discard the majority class's data points. Putting together all benefits and drawbacks of SGD, and considering experimental results, we can conclude that SGD may not be a good optimization tool for small datasets in batch data mining setting. In particular, when there is no limitation on optimization time, storage memory and computational cost, analytic methods (if possible) are preferred to incremental, and approximate methods such as gradient descent as well as SGD. In addition, setting hyper-parameters such as the number of iterations and regularization parameters introduces more

complexity to the process that could be avoided by applying other methods. On the other hand, SGD is a powerful tool when it comes to Big data and online learning or a combination of both, where advantages of SGD perfectly outweigh its disadvantages. One class of web data mining tasks where SGD plays an important role are recommender systems. Recommender systems can be found in almost every online shopping application, where costumers are offered a set of products that seems to match their preferences. Taking Netflix as an example where the recommendations are based on ratings that are previously given by costumers and paring them with movies in a utility matrix. However, when the number of costumers and products grows, handling utility matrix as a whole becomes cumbersome. Therefore, SGD can be used as an alternative method where we look at a fraction of utility matrix when seeking to minimize the total error Leskovec et al. (2014). Since SVMs have been proven to be powerful in batch settings, Kivinen et al. (2010) took one step further and examined SVM in online learning setting. They proposed an incremental updating approach for SVM that can be also applied with kernel functions so can be extended to nonlinear classification tasks. One notable remark of their work is that they could theoretically prove what convergence rate and error bounds of their method are. To tackle convergence problems and decrease the effect of noise caused by single example iterations, Shalev-Shwartz et al. (2011); Wang et al. (2010) introduced a simple stochastic sub-gradient descent called Pegasos. In Pegasos essentially the same idea of SGD is adopted with the difference that at each iteration a small number of instances are used to improve parameters, in hope of reducing the noise of a single example. Pegasos can be basically used on any classification task with SVM where batch gradient descent is not efficient. However, its application in online learning problems, especially where buffering of mini-batches is possible, is significant. Besides presenting their algorithm, the authors have also made some suggestions that may further improve the results of Pegasos, such as sparse feature selection and gradient-projection approach. Moreover, the results show that in practice Pegasos yields significant results even in the presence of kernels. LASVM is another algorithm that is designed to solve SVM in an online setting (Bordes et al., 2005). The difference between

LASVM and previously developed algorithm is that it tries to optimize dual function in two direction, named PROCESS and REPROCESS. The algorithm works with updating coefficients based on their importance to the model. By this means, if an incoming data-point is misclassified with the current parameters it will be definitely used in the updating stage, otherwise it might be discarded. LARANK is an extension of the later algorithm which aims to optimize SVM for multi-class problems (Bordes et al., 2007). Early online SVM algorithms were focused on point loss functions such as hinge loss (as is the case in Pegasos algorithm). However, there were not any online methods for optimization of non-decomposable loss functions such as Precision@k which are useful when dealing with imbalanced and more complex datasets. Kar et al. (2014) developed an online framework to efficiently handle these loss functions in an online setting. However, their model is still an extension to previous online SVM methods. Finally, Bottou (2010) has conducted a detailed study on convergence of SGD and possible remedies to speed up the convergence for large scale learning. As asserted, some modifications of original SGD, like second order SGD, and averaged SGD have better performance for very large datasets: Second order SGD multiplies the gradients by a positive definite matrix Γ_t approaching the inverse of the Hessian :

$$w_{t+1} = w_t - \gamma_t \Gamma_t \nabla_w Q(z_t, w_t) \tag{2.1}$$

The averaged stochastic gradient descent (ASGD) algorithm (Polyak and Juditsky, 1992) performs SGD and computes the average as follows:

$$w_{t+1} = w_t - \gamma_t \nabla_w Q(z_t, w_t) \tag{2.2}$$

$$\bar{w}_{t+1} = \frac{t}{t+1}\bar{w}_t - \frac{t}{t+1} w_{t+1} \tag{2.3}$$

The details on the convergence of the aforementioned methods could be found in the same reference.

Online Passive Aggressive

Online Passive aggressive algorithms are an alternative approach to online optimization. The main focus of online PA methods is on sequential learning of binary classification models. The procedure is similar to SGD learning method, in which a newly arrived datapoint is tested by the current model. If the prediction is incorrect a penalty occurs, then, the data and the true label is used to update the model coefficients for better future performance. Considering the *hinge loss* as the default loss function, the update algorithm is *passive* when there was no error made.

The three variations of the passive online PA updates are as following (Crammer et al., 2006):

PA:

$$w_{t+1} = w_t + \tau_t y_t x_t \quad where \quad \tau_t = \frac{\ell_t}{\|x_t\|^2} \tag{2.4}$$

PA-I

$$w_{t+1} = w_t + \tau_t y_t x_t \quad where \quad \tau_t = min\{C, \frac{\ell_t}{\|x_t\|^2}\} \tag{2.5}$$

PA-II

$$w_{t+1} = w_t + \tau_t y_t x_t \quad where \quad \tau_t = \frac{\ell_t}{\|x_t\|^2 + \frac{1}{2C}} \tag{2.6}$$

where w_t is the tth step coefficients with initialization from zero, and ℓ_t is the loss at tth step. C is a positive real number which is used a a degree for aggressiveness. Simply put, the update rule of PAs is to correct the error on the currently seen example by projecting w_t onto the half-space of vectors which attain a hinge-loss of zero on the current example. Crammer et al. (2006) also propose some modifications of the online PA for regression model and multi-class classifi-

cation problems. He and Wang (2012) have compared different versions of online PA and SGD using benchmark datasets. Wang and Vucetic (2010) proposed a modified version of PA algorithm which maintains only a fixed number of support vectors, by introducing an additional constraint to the original PA optimization problem. They succeeded to obtain a closed-form solution for the support vector removal and coefficients update. Jorge and Paredes (2018) have taken the basic idea of PA updates to the loss functions with nonlinear projections of the feature space. Using nonlinear projections of the data helps to produce new features and supposedly increases the power of the classification model. Their results show significant improvement compared to the original online PA methods.

Wang et al. (2013a) have investigated the fuzzy weighted loss functions incorporated with online PA optimization. The performance of the fuzzy edition of online PA is significantly better for smaller datasets. They also investigated the computational complexity of their proposed version of PA which appears to be almost the same as original online PA.

Concept Drift

As previously discussed, one of the challenges in data stream mining is that time dependent data is prone to change with regard to environmental factors that could be known or unknown. In the literature this phenomena is called *concept drift*. As an example seasonal changes and customer trend shifts may lead to concept drift in the data. There are several types of concept drift that may happen in practice. Identification of the type of the concept drift and proposing remedies to deal with them is the subject area of many recent papers in the online machine learning field (de Barros and de Carvalho Santos, 2018). General categorization of concept drift types is based on the speed at which the underlying statistics of the data changes over time. Sudden changes in the data is called a *shift*. Detailed analysis of different drift types could be found in (Gama et al., 2004;

Gonçalves Jr et al., 2014; GonçAlves Jr and De Barros, 2013). Let Y be the dependent variable and X be the independent feature, then concept drift can affect the posterior $P(Y|X = x)$, the conditional feature $P(X|Y = y)$, the probability distribution of the feature $P(X = x)$, and the class prior distribution $P(Y = y)$ (Sethi and Kantardzic, 2017).

One way to detect concept drift is sequential analysis on the stream of data which is basically monitoring the sequence of performance metrics such as accuracy, F-measure, and geometric accuracy. To decide on what performance measure to observe it based on the type of the problem on hand. Decreasing accuracy may be a good signal for occurrence of concept drift (Page, 1954; de Lima Cabral and de Barros, 2018). Window based distribution monitoring methodologies are based on distribution estimations of sliding windows of the data Bifet and Gavalda (2007). Finally, statistical process control based methods could be used for analyzing the error rates of data chunks to detect changes in the stream of the data Haussler and Warmuth (1993).

Feature selection

Feature selection is the task of identifying important features for a classification problem and removing irrelevant variables from the model (Dash and Liu, 1997). Ideally, feature selection should not reduce the accuracy of the model while reducing computations and storage needs.

There is a wide variety of methods to tackle this problem appropriate for different types of data. For example, biclustering algorithm is used to find related causes to known conditions in gene studies (Kluger et al., 2003; Desai et al., 2012). Biclustering as a general scheme for unsupervised feature ranking is discussed by Huang et al. (2011). Cantú-Paz et al. (2004) provided an application of feature ranking for identifying useful variables in “interesting” states on fusion physics data

.

Some feature selection methods work with calculation of rankings for the features and then fea-

ture reduction is done based on the ranks. Apparently, those highest ranked features are meant to remain in the classification model. The advantage of these methods is the insights which could be gained from the rankings itself. For instance, feature rankings of stock trend prediction models could be used by investment firms to increase their return on investment (Nair et al., 2010; Lin et al., 2013). On the other hand, some feature reduction approaches do not return a ranking but just reduce the dimensionality (Wold et al., 1987; Bi et al., 2003).

Filter methods

Filter methods work by independently calculating the rankings of features. Therefore, the classification method of interest does not affect the rankings. This is normally done by measuring the relationship between the features and the target variable. Among the filter methods those multivariate schemes take into account the inter-relationships of the features, so that they can handle redundancy. Relief is an example for this category with a ranking measure based on mutual information calculation (Robnik-Šikonja and Kononenko, 2003) and correlation-based feature selection methods (Yu and Liu, 2003). Seref et al. (2018) used normalized mutual information (NMI) scores for feature selection in clustering applications. As an application in medical diagnosis, Fan and Chaovalitwongse (2010) presented a feature subset selection based on inter-class and intra-class distances. On the other hand Fisher score is calculated based on the assumption that high quality features have similar values for samples from the same class (Duda et al., 1973). For the categorical feature spaces, the correlation between the features and the target could be measured using Chi2 metric and they will be ranked accordingly. The feature with highest chi2 value will be ranked as the most important (Liu and Setiono, 1995).

Wrapper methods

The second category of feature selection methods are the wrapper methods. The wrappers work by incorporating the classifier into the feature selection strategy and by searching between the results of the classifier among different subset of features. These type of algorithms are criticized for using a brute force approach which is computationally expensive. In particular, for large scale datasets searching among possible subsets to find a good set of features is exhaustive. Greedy search methods are categorized as backward or forward feature elimination tasks. Kohavi and John (1997) have reviewed the earlier feature subset selection algorithms such as best-first, branch and bound and genetic algorithms. Recursive Feature Elimination (RFE) is a popular wrapper method which uses a recursive evaluation procedure to remove the unwanted features from the model (Guyon et al., 2002). RFE has been implemented in many widely used machine learning packages. A sequential backward selection, using the number of errors in a validation subset as the measure to decide which feature to remove in each iteration is proposed by Maldonado and Weber (2009). Their classifier of interest is SVM with kernel functions. Zhu et al. (2007) has proposed a hybrid feature subset selection approach that adds or deletes a feature from a candidate feature subset (the wrapper part) based on the univariate feature ranking information (the filter part). Kabir et al. (2010) have invented a novel wrapper method with a base neural network classifier and a built-in mechanism to update the architecture of the classifier. They use correlation information of the features to encourage the search strategy for selecting less correlated (distinct) features if they enhance accuracy of the classifier. Such an encouragement will reduce redundancy in the selected subset.

Embedded methods

Unlike other two methods which may or may not be dependent on the classifier, embedded methods work by simultaneously performing the feature selection and model training. Compared with filter methods, embedded algorithms could be as computationally efficient Guyon and Elisseeff (2003). Le Thi and Nguyen (2017) have developed a framework of embedded feature selection methods for multi-class SVM. Another example of embedded methods are the algorithms which perform pruning on the features such as ID3, and C4.5 (Quinlan, 1986, 2014).

Feature selection vs feature extraction

So far, we discussed multiple approaches for feature selection in conventional machine learning applications. Feature reduction is generally performed to reduce the size of the model as well as to improve the performance by removing irrelevant factors from the model. Feature selection methods work by directly eliminating the unwanted features and presenting a final sub-collection of initial features. However, in many practical applications, feature selection on its own is not enough to improve the performance of the models. Therefore, another category of methods, called feature extractions come into play. Feature extraction contains a broad selection of methods that practically invent new features from initial set of features. Multi-linear subspace learning is the mapping from a high-dimensional vector space to a set of lower dimensional vector spaces (Vasilescu and Terzopoulos, 2003). Principal component analysis (PCA) is an example of multi-linear subspace learning algorithms that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components (Hotelling, 1933).

Class-based feature selection

In feature selection methods, normally the power of each feature to distinguish the labels is identified by the whole training data. In practice, however, there might be some features which are good in distinguishing between some labels and not useful for others. For instance, in identifying potential costumers for a product in online marketing, the class of customers all has high incomes, however, the income level among non-customers varies significantly.

There are several approaches to take advantage of class-specific feature structures for feature selection in the literature. Even though the idea is not still used and implemented in online learning, there are several papers which have presented algorithms for class-specific feature selection in batch learning setup. Majority of these methods focus on ensembles of classifiers in which each classifier is trained using a subset of features important to each class. In these types of class-based feature selection algorithms, the underlying feature selection or ranking is typically one of the commonly used feature selection methods. For instance, Zhou and Wang (2006) adopted the RELIEF weight measure and class separability for the feature selection part of their method. In another paper, Nina and Wang (2008) used PCA as a base method for their feature selection. de Lannoy et al. (2011) have applied Jeffries–Matusita (JM) distance for class-dependent feature subset selection and then have trained multiple Bayesian classifiers to form their ensemble of classifiers.

Another concept which is used for class-dependent feature selection is separability of features values in each class. Suppose d_{kn}^{in} is the average distance between the patterns within class k for feature n, and d_{kn}^{out} is the average distance between the patterns in class k and those not in class k for feature n. Let $r_{kn} = d_{kn}^{out}/d_{kn}^{in}$, and call it the separability index. The ratio r_{kn} is used to rank order the features of class k where a higher ratio brings a higher rank. This method is applied to both batch and online learning (Roy, 2015; Roy et al., 2013). Even though this method may work for continuous feature spaces but it is not a universal solution for mixed environments.

Online feature selection

Unlike batch learning, the problem of feature selection for online learning has gained much less attention in the literature. The reason is that many conventional methods which are applicable to batch learning could not be used in online learning. For instance, the feature subset selection using RFE needs the presence of all training set, while the online feature selection or feature importance ranking should be done in real-time. Common algorithms for online feature selection is the addaptation of avalaible batch feature selection strategies where the importance of a feature can be learned incrementally. Statistical correlational measures are among such rankings Ramírez-Gallego et al. (2017).

Streaming features

More precisely, there are different types of feature selection in the context of online learning. In one scenario, the feature space itself is evolving and new features emerge as the learning process continues. A good example of this problem maybe the text mining applications in which new features are the new words which appear in new training data. Therefore, the role of feature selection is identifying useful and relevant features to the model as they arrive. In the literature this type of online feature selection is called *streaming feature selection* (Perkins and Theiler, 2003). Wang et al. (2015) attempted to measure the correlation of the new features as they arrive in clusters.

Fixed sized feature space

The second type of online feature selection which is the focus of this work, is based on the assumption that the set of features are known and fixed from the beginning. Therefore, the feature selection task is to identify unimportant features as the training input gets to the model one by one. The use of $l1$ norm for feature reduction has been proposed by (Wang et al., 2014). Even though this method is useful in feature reduction by producing sparse models, it does not calculate the rankings of the features. In real world applications feature ranking could reveal important information about the influential factors on the classification target.

To the knowledge of the authors, there are currently few methods for feature importance ranking and feature selection in online setup. Carvalho and Cohen (2006) have proposed an embedded feature reduction strategy which works with *Modified Balance Winnow* classifier, the disadvantage of this method is that it cannot be independently used with any classifier. Katakis et al. (2005) proposed an online feature subset selection method for text classification, though useful for text mining tasks, this method is not applicable to other online classification problems. Bolon-Canedo et al. (2016) have developed an online learning pipeline in which the features are ranked by updating the value of Chi2 equation. This parameter basically measures the strength of correlation between two categorical variables (Liu and Setiono, 1995). As a mean to handle continuous features they suggested the use of incremental k-means discretization method. The disadvantage of this method is that discretization of continuous features can lead to loss of information. Razmjoo et al. (2017) have proposed a new feature importance measure based on the sensitivity analysis of the classification output due to changes in individual features. Despite its practical value for problems with continuous features, it cannot be directly applied to problems involving both continuous and categorical data.

Finally, Nguyen et al. (2012) suggested the use of ensemble learning for online feature selection in which several classifiers are trained with different feature subsets, however, the main issue could be

the speed and efficiency of such ensemble online methods and thus scalability may not be possible.

CHAPTER 3: FEATURE IMPORTANCE RANKING BASED ON SENSITIVITY ANALYSIS

In this chapter we propose the steps to develop a feature importance ranking based on the strength of sensitivity of the target variable to the changes to the individual features. The proposed method is applicable to continuous feature spaces or ordered features where increasing or decreasing the value of a feature practically makes sense. The major advantage of this method is the built-in mechanism we use to reduce the redundancy in terms of correlated features.

Feature importance ranking in online setting

In order to distinguish between the degree of importance for each feature, we would need to know the relationship between features and the class label, as well as relationship between features themselves. We start developing our method first by assuming that all features are mutually independent, and their influence on class labels is unknown at the beginning. We further extend our method to more complicated feature spaces with correlated features involved.

Feature importance measure

When the underlying characteristics of feature space is unknown, it is not possible to determine the relationship between a specific feature value and a class label. However, the importance of a feature can still be naturally described by its impact on the outcome of classification model by measuring sensitivity of outcome due to changes in features' values. Sensitivity analysis for feature importance ranking is a common tool to understand relevance of variables in statistical models (Iooss and Lemaître, 2015). This way, a feature is assumed to be irrelevant or redundant if perturb-

ing the feature's value does not normally change the result of the classification model. Therefore, we would like to know how much on average, results are sensitive to the changes of independent features. This sensitivity measure will allow us to rank features' importance in an incremental manner as described in the rest of this section.

Since we assume that features are mutually independent, we can conclude that changing one feature's value does not affect others. Here, our focus is on online classification process where an estimation of the parameters of the separating hyperplane with maximum margin are updated as the datapoints arrive into the system.

Let $x = (x_1, ..., x_d)$ be the newly arrived instance, $w = (w_1, ..., w_d)$ coefficients and c the intercept, be the classification parameters for the separating hyperplane. We decide which class x belongs to as follows:

step 1: Calculate:

$$a = w_1.x_1 + w_2.x_2 + ... + w_d.x_d + c \tag{3.1}$$

step 2: The class label is predicted as $\mathrm{sign}(a)$.

Note that we can conveniently remove the intercept parameter c from the classification model by adding a constant feature as $x_{d+1} = 1$. Therefore, for ease of conversation we will not mention c in the rest of this chapter.

Having this in mind, we add a perturbation $x_j \cdot \epsilon$ to the value of jth feature to change the class prediction.

The minimal effort to change the sign would require the following equation:

$$w_1 \cdot x_1 + w_2 \cdot x_2 + ... + w_j \cdot x_j + w_j \cdot x_j \cdot \epsilon + ... + w_d \cdot x_d = 0 \tag{3.2}$$

Therefore, $\epsilon = a/(w_j \cdot x_j)$.

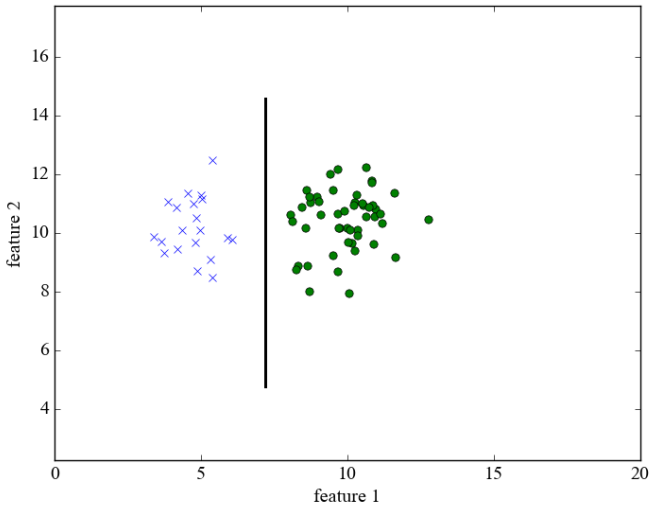
As an instance let for a given x and w, and for the j^{th} feature we obtain $|\epsilon| = 0.2$. This result can be interpreted as more than 20% change in the value of j^{th} feature will change the predicted class label. From an intuitive perspective, there should be an inverse relationship between the value of $|\epsilon|$ and importance of a feature. This means if classification model is on average insensitive to the changes in value of a specific feature, that feature is redundant or irrelevant. Therefore, we developed the following measure as the average importance of a feature.

Definition Let $w^t = (w_1^t, w_2^t, ..., w_d^t)$ be the t^{th} classification parameters, and $x^t = (x_1^t, x_2^t, ..., x_d^t)$ be the t^{th} instance which arrives into the system. The importance factor of j^{th} feature is defined as following:

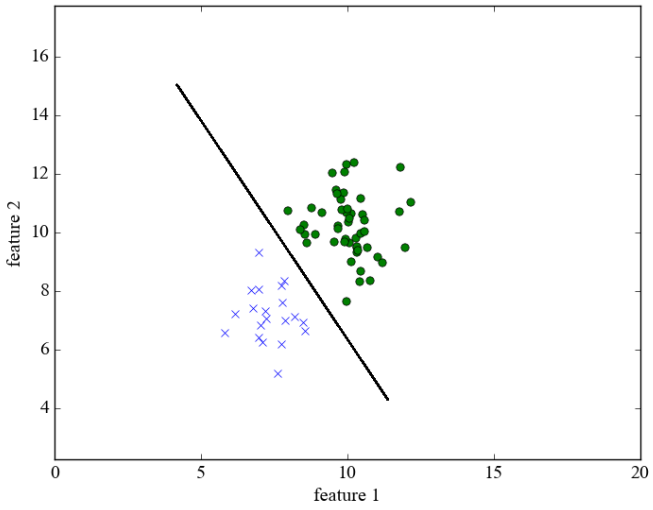
$$e_j^t = \begin{cases} \frac{1}{t}[(t-1)e_j^{t-1} + \frac{1}{1+\alpha \cdot |\frac{a^t}{w_j^{t-1} \cdot x_j^t}|}] & \text{if } w_j^{t-1} \neq 0 \\ \frac{t-1}{t} e_j^{t-1} & \text{if } w_j^{t-1} = 0 \end{cases} \tag{3.3}$$

where $a^t = {w^{t-1}}^T \cdot x^t$ and α is a real positive number that serves as a tuning parameter, by default α is set to 1. The initial values for e_j^0's are set to be zero.

To understand the concept intuitively, we provided a simple two dimensional classification task in Figure 3.1. As can be seen in (a) feature 2 is redundant using the proposed importance measure, because each instance can be highly disturbed across vertical axis and still does not fall into the other class. In subfigure (b), however, both features are equally important and removing one will result in high classification error. In this case the average distance of instances according to both feature axes is almost the same. The variable e_j^t in Equation 3.3 acquires a value between 0 and 1 at all times which indicates the importance of a feature.



(a) Feature 2 is redundant



(b) Both features are important

Figure 3.1: Classification of two normally distributed class instances

Feature importance ranking for correlated feature spaces

One common approach to deal with real valued correlated features is to calculate the correlation matrix for the feature space. In the case of online learning, correlation matrix can be updated along with classification parameters, incrementally.

As explained earlier, our approach is to rank features according to their average effect on the predicted label. We would like to know for a given instance, say x, how much disturbance in the value of a specific feature is needed for the x to be predicted in a different class. Low sensitivity means that feature is not important in this case. This approach can be as well applied to correlated feature spaces. However, we should be cautious that changing one feature's value of x, will change the values of highly correlated features. Hence, ϵ perturbation in the value of i^{th} feature will result in a variation of $b \cdot \epsilon$ on the value of the j^{th} feature, where b is the regression slope of regression line between the two features.

Before proceeding to the details of the proposed feature importance measure, as our necessary tools, here we briefly review the calculation of the correlation matrix as well as the regression slope matrix in an incremental manner according to (Dowdy et al., 2011). Let $X \in \mathbb{R}^n$ and $Y \in \mathbb{R}^n$ be two different features values for all n instances, an efficient way to calculate the regression slope is by using the formulas developed when updating correlation matrix as following:

$$S_{XX} = \sum_i (x_i^2) - \frac{(\sum_i x_i)^2}{n}$$
$$S_{XY} = \sum_i (x_i \cdot y_i) - \frac{(\sum_i x_i)(\sum_i y_i)}{n}$$

The correlation coefficient, r is:

$$R^2 = \frac{S_{XY}^2}{S_{XX}S_{XY}}$$
$$r = \sqrt{R^2}$$

Then, the regression slope of the regression line between X and Y will be:

$$b = \frac{S_{XY}}{S_{XX}}$$

Note that values of S_{XY} and S_{XX} can be simply modified to be updated as the data points arrive into the system. During implementation for a classification task with d features there are two $d \times d$ matrices to be defined, one for saving and updating correlation coefficients and the other for regression slopes.

Now, we have the material needed to define our feature importance measure as following.

Definition Let $w^t = (w_1^t, w_2^t, ..., w_d^t)$ be the t^{th} classification parameter, and $x^t = (x_1^t, x_2^t, ..., x_d^t)$ be the t^{th} instance that has arrived into the system. The importance factor of j^{th} feature is described as following:

$$e_j^t = \begin{cases} \frac{1}{t}[(t-1)e_j^{t-1} + \frac{1}{1+\alpha\cdot|\frac{a^t}{\sum_{k\in A} w_k^{t-1}\cdot b_{kj}^t \cdot x_j^t}|}] & \text{if } w_j^{t-1} \neq 0 \\ (\frac{t-1}{t})\cdot e_j^{t-1} & \text{if } w_j^{t-1} = 0 \end{cases} \tag{3.4}$$

where $a^t = {w^{t-1}}^T \cdot x^t$ and α is a real positive number that serves as a tuning parameter. b_{kj}^t is the slope of regression line for k^{th} and j^{th} feature at time t. Finally, A is the set of indexes that

are correlated to j^{th} feature (we may assume two features to be correlated where absolute value of their correlation is greater than a set minimum). $e_j^0 = 0$ for all $j = 1, 2, ..., d$.

In the case that there is no significant correlation between features, Equation (3.4) will be equivalent to Equation (3.3). The ranking algorithm can be implemented along with the online learner of interest.

Feature selection

Feature importance ranking itself provides useful information about input data. For instance, in financial planning, feature ranking reveals influential factors in fluctuations of stocks' price in real time. Besides, online feature importance ranking can be used as a guide to feature selection. There are several ways to select important features at each time step. The easiest way to handle this, is to specify a fixed number for the number of features to be selected. However, this may lead to removing important features if the number of truly important features exceed that limit.

Hereby, we define a two stage filter method to eliminate redundant and unimportant features appropriately. In the first stage, we eliminate redundant features through monitoring the correlation matrix. For highly correlated features, those with smaller importance value are set to be removed. After elimination of redundant features, we define a minimum importance threshold for remaining ones to be included in the subset of selected features. As an instance, we assume a feature to be unimportant if on average disturbing its value by 100% does not change the class prediction. Therefore, any feature with sensitivity coefficient of $e_j^t = \dfrac{1}{1+\alpha}$ at t^{th} step is considered as unimportant and its corresponding coefficient in the classification model, i.e., w_j^t is set to zero. In this way, unwanted features are no longer present in the predictive model. In this regard, we introduce a tolerance level of λ to be set for the purpose of feature selection. Therefore, if e_j^t is less than

$\frac{1}{1+\alpha.\lambda}$ feature j is regarded as unimportant. In practice, we understand that it may happen that none of features meet minimum importance criteria in the second stage. To tackle this problem, we set a minimum and maximum percentage of total features to be selected at the feature selection step. One important point to note is that features are no longer present in the model at the moment they are removed. Therefore, it is important to execute this step while there are sufficient datapoints into the system and correlations are statistically meaningful. The decision on when to remove redundant features from the model is to be made based on the domain and storage/computational limitations. A summary of necessary steps are shown in Algorithm 1.

Algorithm 1: Online feature ranking for correlated feature spaces

Input: d = # of features, α sensitivity tuning parameter (default $\alpha = 1$), minimum threshold of correlation, min # of selected features, max # of selected features, λ, classification update function.

Initialization: $w = 0$ *where* $w \in R^d$.

for *t=1,2,...* **do**

- receive instance x^t
- predict label $y^t = \text{sign}(w^{t-1}.x^t)$
- update correlation matrix
- update regression slope matrix
- update feature's importance ranking, e.g., e_j^t for $j = 1, 2, ..., d$
- feature selection (optional):
  - step 1: eliminate highly correlated features
  - step 2: eliminate irrelevant features
- make prediction
- receive correct label $y^t \in \{-1, 1\}$
- update classification parameters $w^t \leftarrow w^{t-1}$.

end

Output: feature importance rankings, w.

While learning a classification model in online fashion, one issue that may arise is the fact that the underlying characteristics of feature space may change over time. In other words, a model that works for older data points may not work for new data as properly. In the literature this behavior of data streams is called *concept drift*. There are many researches devoted to the detection and handling of concept drift (Sayed-Mouchaweh, 2016). In the case of online feature importance ranking, the problem may arise when the importance of features change during time in an unpredictable manner. For instance, in a stock market, the important factors on increase or decrease of companies stocks' value can easily change over time such that one unimportant feature today, becomes influential in near future or vice versa. One way to handle this problem is the use of a time decay function in the learning process as suggested by Wang et al. (2013b). Therefore, by applying a time decay function in place of averaging over all sensitivities in equation (3.4), it can be rewritten as (3.5).

$$e_j^t = \begin{cases} \beta \cdot e_j^{t-1} + (1-\beta) \cdot \frac{1}{1+\alpha \cdot |\frac{a^t}{\sum_{k \in A} w_k^{t-1} \cdot b_{kj}^t \cdot x_j^t}|} & \text{if } w_j^{t-1} \neq 0 \\ \beta \cdot e_j^{t-1} & \text{if } w_j^{t-1} = 0 \end{cases} \tag{3.5}$$

where β, $0 < \beta < 1$ indicates how much focus is given to sensitivity of last datapoint comparing to older instances. In order to increase the effect of later datapoints, $\beta < (t-1)/t$ should hold. Therefore, as the number of instances grows, the sensitivity of older datapoints become less influential. For $\beta = (t-1)/t$, Equation (3.4) and (3.5) are equivalent. Whether to use Equation 3.4 or 3.5 is to be made based on characteristics of the learning environment.

Experimental results

To thoroughly test the practical values of the proposed method, we perform three sets of experiments. The experimental tests will include: comparison with batch learning methods, evaluation of the feature ranking measure in online mode, and evaluation of the proposed feature selection method as described in Section 3.

Experiments setting

We conduct our experiments on publicly available datasets of different sizes Chang and Lin (2011); Lichman (2013) focusing on those datasets with non-categorical feature spaces. The information on testing datasets can be found in Table 3.1.

Table 3.1: Datasets' information

Dataset	# instances	# features	% of majority class
Pima	768	8	65%
Ionosphere	351	34	64%
wdbc	569	30	63%
spambase	4601	57	61%
ijcnn1	49990	22	90%
svmguide3	1243	22	76%

Comparison with batch learning methods

In the first set of tests we evaluate the ranking method in comparison with two popular feature ranking methods: ExtraTree (Geurts et al., 2006) and RFE (Guyon et al., 2002). To do the comparison, first feature ranking is done using the whole training set, then training process takes place

Table 3.2: Two tailed paired T-test is performed on the accuracies, SFR vs. other methods. The paired values of (p-value, t-stat) are shown in the table. As highlighted boldface, where $p - value/2 < 0.05$, and $t - stat > 0$, we could conclude SFR has significantly performed better.

	25% of features		50% of features		75% of features	
Dataset	vs. ExtraTree	vs. RFE	vs. ExtraTree	vs. RFE	vs. ExtraTree	vs. RFE
Pima	(0.31, 1.03)	(0.18,1.36)	(0.67, 0.43)	(0.9, 0.13)	(0.11, 1.63)	(0.32, -1.0)
Ionosphere	**(0.00, 10.59)**	**(0.00, 8.42)**	**(0.00, 6.8)**	**(0.00, 8.17)**	**(0.00, 4.436)**	**(0.00, 4.53)**
wdbc	**(0.00, 3.86)**	**(0.00, 3.75)**	**(0.00, 4.86)**	**(0.036, 2.197)**	**(0.00, 3.01)**	(0.68, -0.41)
spambase	**(0.026, 2.34)**	**(0.017, 2.51)**	(0.00, -6.77)	(0.52,0.66)	(0.00, -8.07)	**(0.01, 2.54)**
svmguide3	(0.109, 1.65)	**(0.04, 2.11)**	**(0.00, 5.1)**	**(0.00, 4.95)**	**(0.00, 4.74)**	**(0.00, 5.28)**
ijcnn1	**(0.00, 15.8)**	**(0.00, 15.8)**	**(0.00, 43.2)**	**(0.00, 43.2)**	**(0.00, 39.76)**	**(0.00, 39.76)**

with the unwanted features deleted according to this ranking. It is worth noticing that unlike RFE and ExtraTree, feature ranking for SFR are computed incrementally. Comparison graphs show the accuracy vs. number of selected features. For this experiment, 20% of data points are taken for testing purposes. To eliminate the factor of chance, we run the test on randomly shuffled datasets 30 times. Therefore, the graphs are average accuracy on all results (Figure 3.2). After the feature rankings are obtained, we used single-pass stochastic gradient descent with fixed stepsize of 0.1 with tuning parameter $\alpha = 1$ to train the classification model. The greater value for α means bigger tolerance which leads to smaller sensitivity.

More detailed statistical tests were performed on the performance of the three methods. Significantly better results are highlighted in Table 3.2.

Stability

One important factor that affects the results of online learners is their stability to the order of instances. Kalousis et al. (2007) suggested to use Spearmans' coefficient to see how much on average the results of a ranking measure are similar when the testing and training data are randomly

Table 3.3: Stabilities of tested methods

	SFR		RFE		ExtraTree	
Dataset	stability	avg. std	stability	avg. std	stability	avg. std
Pima	0.9289	0.489	0.9639	0.3348	0.9371	0.5137
Ionosphere	0.909	2.7439	0.7150	4.56	0.5244	7.6485
wdbc	0.9236	1.9552	0.9087	2.0991	0.6974	4.5246
spambase	0.9274	3.7017	0.9488	3.0547	0.9157	4.3514
svmguide3	0.9439	1.2923	0.9117	1.572	0.9080	1.6769
ijcnn1	0.9891	0.5539	0.9764	0.798	0.9637	0.966

selected as following:

$$S_R(r, r') = 1 - \sum_i \frac{(r_i - r'_i)^2}{m(m^2 - 1)} \tag{3.6}$$

where m is the number of features, and r, r' are two different rankings. Also, r_i and r'_i are the ranks of feature i in rankings r and r', respectively.

In order to compare the stability of SFR with the two other methods, we repeated the test 30 times. The average stabilities are recorded in Table 3.3. In this table, the reported stability is the average of two by two similarities between rankings for the 30 repetitions. Moreover, average standard deviation (avg std) is the standard deviation for rankings obtained for each feature in the 30 repetitions averaged for all features in the dataset. Apparently lower standard deviation shows the ranking of the features were close and the rankings are stable.

Experiments in online mode

Since, SFR is developed as an online feature ranking method, we ideally need to compare its merits in an online fashion. The goal of these experiments is to see how feature reduction based on online ranking methods would affect the prediction power of the model in time. For the sake

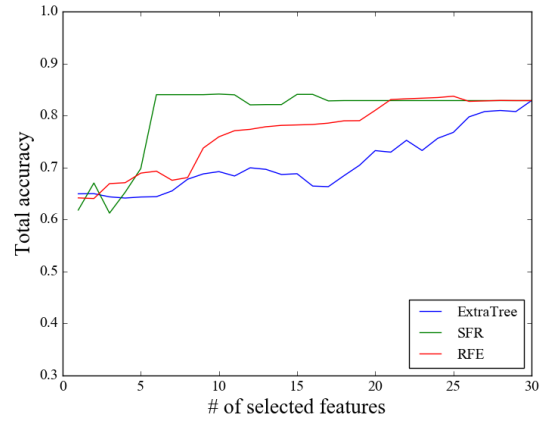
of comparison, we chose the ranking measure presented by (Bolon-Canedo et al., 2016). In their methods, authors basically implemented the Chi2 correlation measure in a incremental form. Since Chi2 is originally for categorical variables, they suggested the use of online k-means discretizer to transform continuous features to categories. In our experiments we use $k = 10$ as the number of categories in our implementation. Moreover, we added random feature elimination as an online feature selection method to see how would other methods compete against it as suggested in Wu et al. (2013). The results are shown in Figure 3.3. All the feature ranking methods are being updated incrementally, and at specific timepoints, a quarter of features which are deemed unimportant according to each rankings are dropped from the model. After each instance arrives in to the system accuracy is updated as:

$$accuracy = 1 - \frac{wrong}{total} \tag{3.7}$$

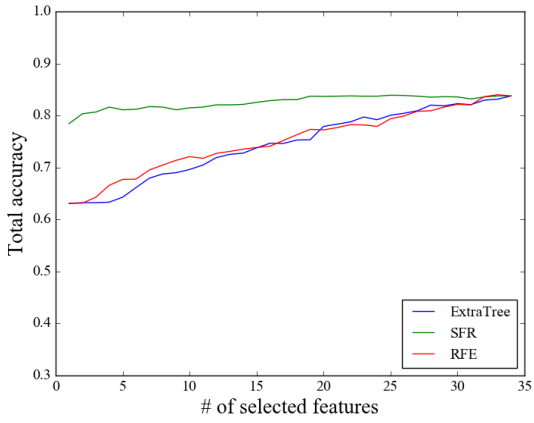
Each time the prediction model makes a mistake, one is to be added to *wrong*.

Evaluation of feature selection method

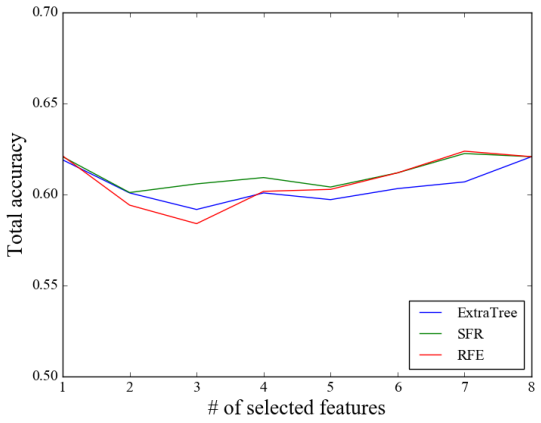
In the last set of tests, we evaluate efficiency of feature selection methods proposed in Section 3 for two different values of λ. The bar graphs which show the accuracy of classification model with and without feature selection are presented in Figure 3.4. At last, the total number of nonzero coefficients in the classification model after feature selection are shown in Table 3.4. As it can be seen from bar graphs there is very little difference between results for tests with $\lambda = 0.5$ and $\lambda = 1$. The minimum and maximum percentage of used features is 25% and 75%, respectively. Also, the minimum threshold for redundancy correlation is set to $|r| > 0.8$.



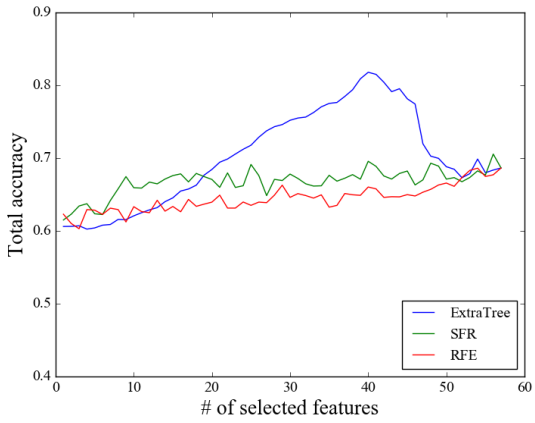
(a) wdbc dataset



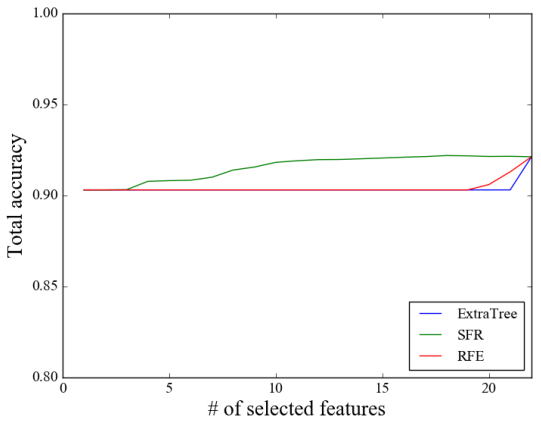
(b) Ionosphere dataset



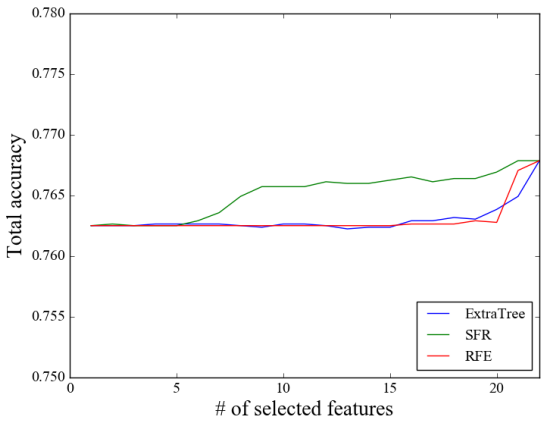
(c) Pima datast



(d) spambase dataset

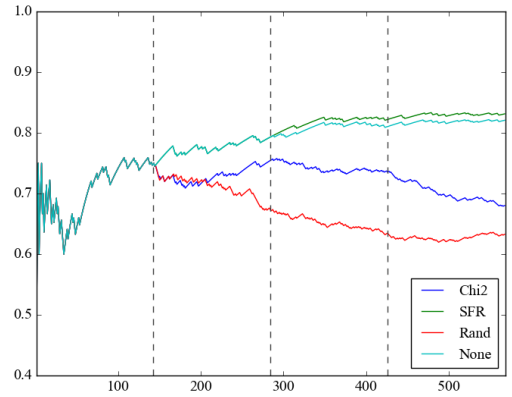


(e) ijcnn1 dataset

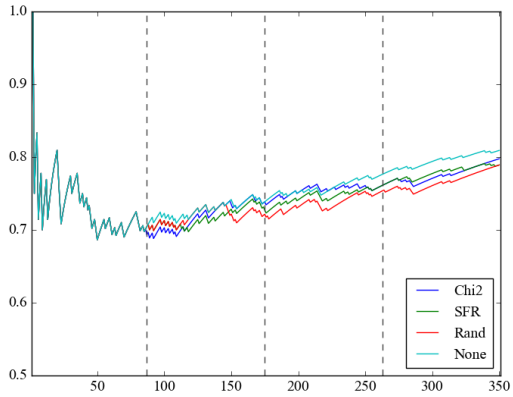


(f) svmguide3 dataset

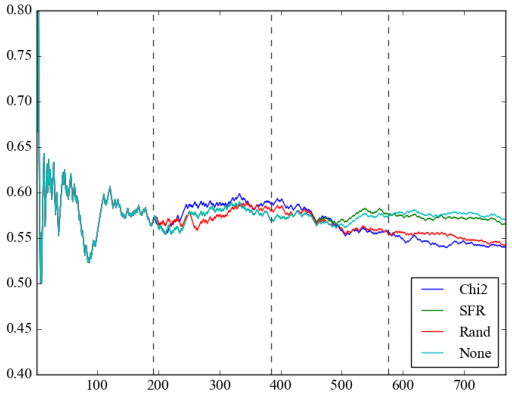
Figure 3.2: Comparison of average performance



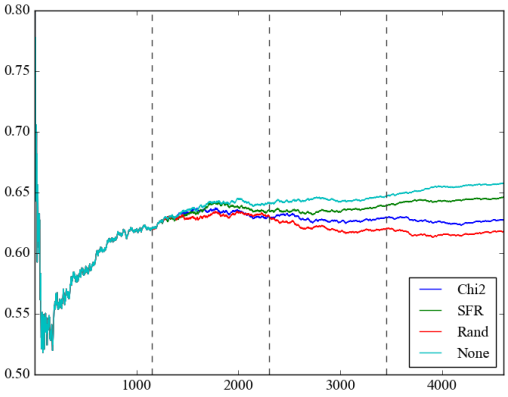
(a) wdbc dataset



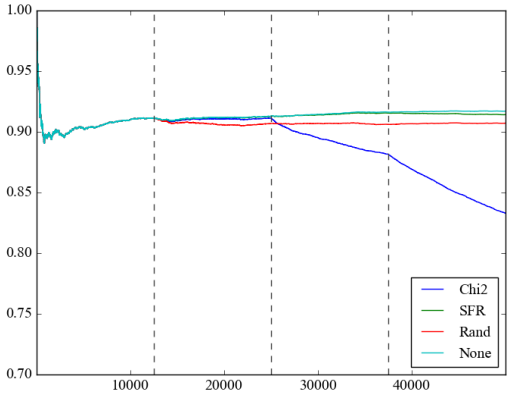
(b) Ionosphere dataset



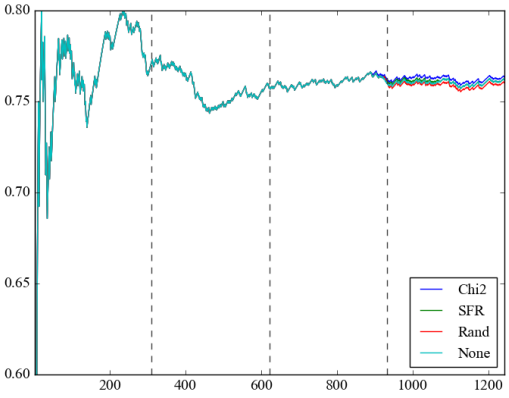
(c) Pima datast



(d) spambase dataset

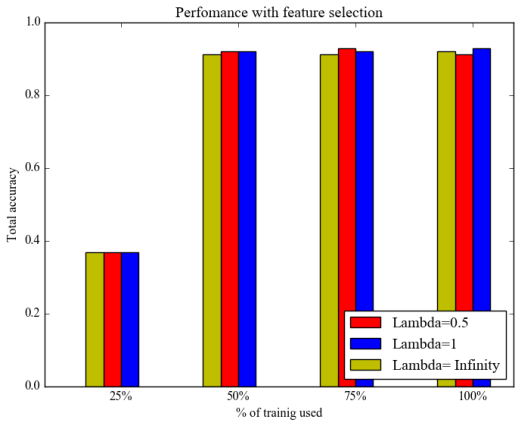


(e) ijcnn1 dataset

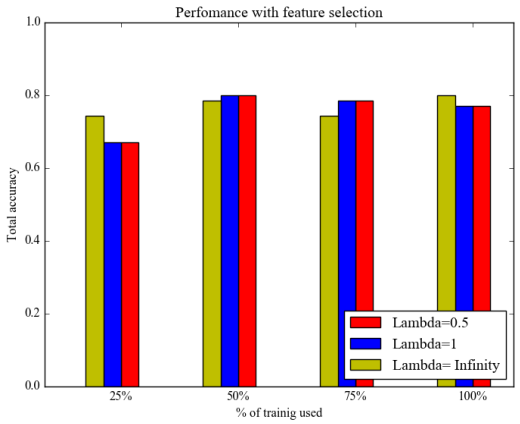


(f) svmguide3 dataset

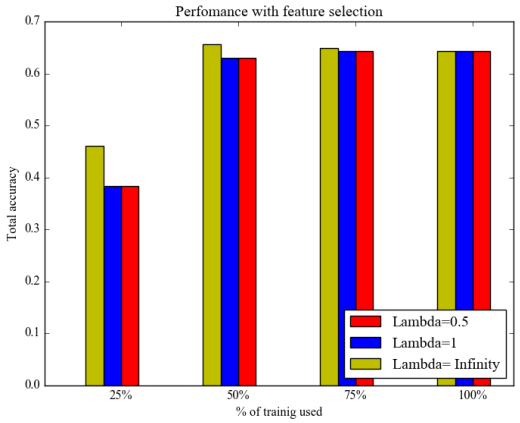
Figure 3.3: Comparison of classification accuracy in online mode. Dashed lines indicate the time-points where feature reduction has happened.



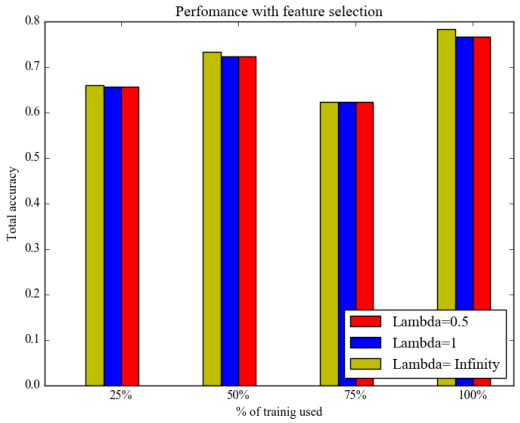
(a) wdbc dataset



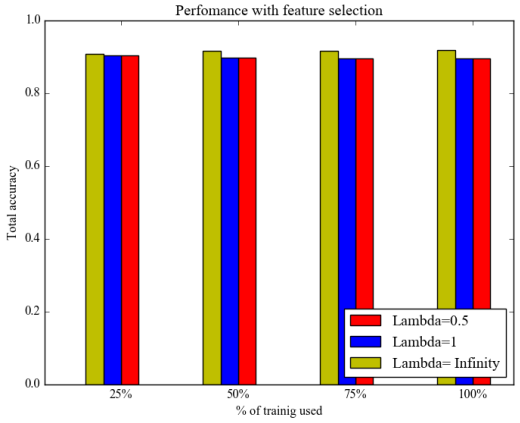
(b) Ionosphere dataset



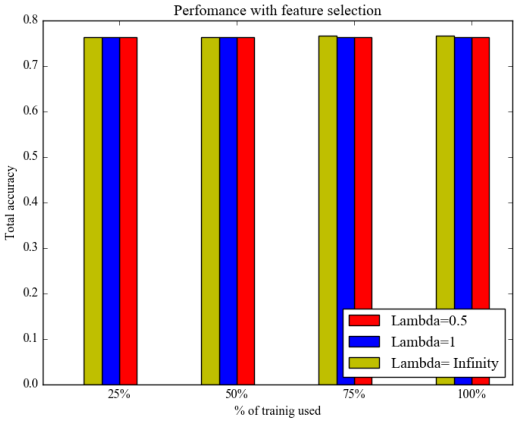
(c) Pima datast



(d) spambase dataset



(e) ijcnn1 dataset



(f) svmguide3 dataset

Figure 3.4: Performance with feature selection

Table 3.4: Total number of nonzero coefficients after feature selection including the extra constant feature.

Dataset	$\lambda = 0.5$				$\lambda = 1$				Total Features
	25%	50%	75%	100%	25%	50%	75%	100%	
Pima	4	4	4	4	4	4	4	4	8
Ionosphere	9	9	9	9	9	9	9	9	34
wdbc	8	8	8	8	9	11	12	12	30
spambase	15	15	15	15	15	15	15	15	57
svmguide3	6	6	6	6	6	6	6	6	22
ijcnn1	6	6	6	6	6	6	6	6	22

Discussion and concluding remarks

In this chapter we presented a new approach to the online feature ranking by introducing an efficient way to rank features in an incremental manner. We conducted experiments to compare the method's efficiency in both batch and online mode. The results will confirm the superiority of our method in comparison with ExtraTree and RFE (Pedregosa et al., 2011) (See Figure 3.2). Based on this we can draw the conclusion that our method can be used as a feature ranking method for feature selection in batch learning classification problems.

Furthermore, the results of stability test show that SFR is highly stable for all six testing datasets and the produced ranks are similar in different repetitions (see Table 3.3). This is a significant achievement, because many feature selection methods such as $L1$ based feature reduction algorithms often suffer from low stability (Kamkar et al., 2015).

The second set of experimental results is an evidence that SFR is successful is eliminating the unimportant features, because the accuracy of the model after feature reduction remains close to the accuracy of the model with no feature selection. The main difference between SFR and online Chi2 is the ability of SFR to take care of correlated features, unlike the latter which assumes all the features are two by two independent. Detailed information on how much correlation in testing

datasets exists based on the dataset correlation matrix is shown in Table 3.5. For instance, SFR works well on wdbc dataset, which has relatively high amount of correlated features. On the other hand, there are no correlated features in the dataset ijcnn1, but SFR reaches better results. One possible cause could be the discretization process in online Chi2, which might have caused information loss. Accuracy of the models for Ionosphere and svmguide3 are almost the same for all feature selection methods. It could be due to reason that features have similar prediction power and the selection method does not affect the results.

In the last set of experiments, we test the performance of the proposed feature selection method. For some datasets such as Svmguide3 and wdbc feature selection actually improves the classification results while for ijcnn1 dataset the results were slightly lower with feature selection. For the datasets, spambase, ijcnn1, svmguide3 and Ionosphere only the minimum required features are left untouched while for others number of selected features fluctuates in their defined bound. Considering that only a fraction of features are selected at each step, we can say that feature selection yields satisfying results.

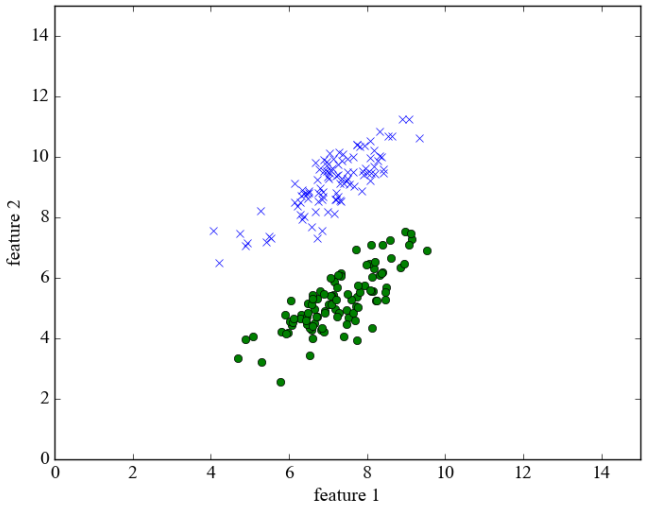
Even though the proposed experiments provided satisfactory results for the proposed method, there are some exceptions where our method might not properly rank features. In the feature selection stage we assumed that one of the two highly correlated features is generally redundant and can be removed accordingly, however, this may not always be true. A two-dimensional classification task, where both features are highly correlated, yet both are vital to the model is shown in Figure 3.5, subfigure (b). Removing each will result in high classification error. Meanwhile, two highly correlated figures where one is truly redundant are shown in subfigure (a). Therefore, if the storage is not a huge problem, we suggest the algorithm for uncorrelated features to be used to avoid these exceptions. Moreover, in the data selection step we assigned lower and upper limits for the number of feature to be selected. These limits can be assigned according to the size of feature set and limitations on storage/computational capacities. In real world applications, stream learning is usually implemented using distributed systems such as MapReduce or Spark in Hadoop ecosystem.

Our method which includes updating correlation matrix can be easily modified to fit in distributed learning environments (Bengfort and Kim, 2016).

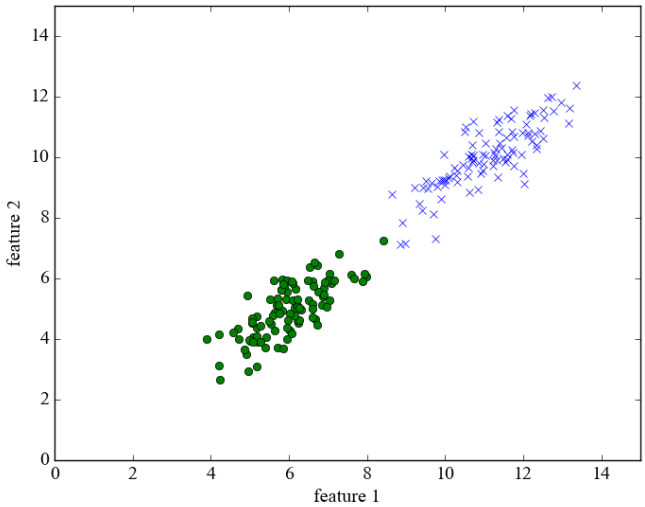
At last, we need to highlight the point that the proposed method best works for continuous feature spaces. Future works can be done to further extend this methodology on categorical and/or mixed feature spaces to embrace more diverse classification tasks. Moreover, since the current implementation is based on the distance from separating hyperplane, it could not be directly applied to multiclass problems. An extention of the method for multiclass classifications could be addressed in future works using SVM-like methods for multiclass problems (Hsu and Lin, 2002).

Table 3.5: Percentage of correlated features in each dataset.

Dataset	$0 < \|corr\| < 0.5$	$0.5 \leq \|corr\| < 0.7$	$0.7 \leq \|corr\|$
Pima	96.42	3.57	0
Ionosphere	84.84	8.73	0.53
wdbc	66.67	17.24	16.09
spambase	98.06	1.5	0.44
ijcnn1	100	0	0
svmguide3	79.22	5.19	6.49



(a) Both features are important



(b) One feature is redundant

Figure 3.5: Synthetic datapoints are generated from Gaussian distribution with covariance of 0.8 between the two features.

CHAPTER 4: FEATURE IMPORTANCE RANKING FOR MIXED ONLINE ENVIRONMENTS

In this chapter, we present a new online feature importance ranking approach based on sensitivity analysis of the classification output. The difference between this method and the one proposed in the previous chapter are:

1. Previous method could work only with continuous features, however, datasets with mixed of integer, categorical and continuous values are highly common in real world applications.
2. Unlike the previous method, the new one does not depend on the type of the classifier. In the other words, it could be applied along with a decision tree, SVM, Neural Network, etc.
3. The previous method is primarily developed for binary classification tasks, but the new one could also work with multi-class classification problems.

Problem description

The first step to understand the importance of the features in a classification task is to officially describe the concept of relevance in ML prediction. We build our method based on the following definition of redundancy. Let X and Y be two random variables, then X is considered irrelevant to the output Y if 4.1 holds.

$$P(Y = y|X = x) = P(Y = y) \tag{4.1}$$

The next step is to develop a methodology to measure the relevance with regard to this criteria. One way to achieve this goal is to observe the behavior of the assumably dependent variable Y to the changes incurred on input variable X. Since the measurement of posterior conditional probabilities of the target variable directly is not possible, Shen et al. (2008) proposed a feature importance ranking based on the earlier definition of redundancy for batch training of a SVM classification model. They attempt to measure the impact of a feature by permutation of its values across all training set. It could be shown that this permutation process causes the elimination of a feature's impact on the outcome and makes it deliberately irrelevant to the target. They have presented a few versions of this basic algorithm in which all the feature vectors are fixed except for one which is being shuffled or removed. They observe the changes in the outcome of the classification twice, once with the correct data and the second time with the disturbance. If a feature is important this difference should be relatively large.

The intuition behind this approach is similar to the effective sensitivity analysis method called one-factor-at-a-time in which the effects of an input is observed by measuring the changes in the output by fixing all factors but one (Czitrom, 1999; Saltelli and Annoni, 2010). Moreover the similar idea is used for Monte Carlo simulation applications in which the relationship between input variability and model's output variability is studied (Thomopoulos, 2012).

Before proceeding to the feature ranking algorithm, we review some definitions in online classification which will be later used in the chapter.

Definition Loss function $\ell(w)$ is the total loss suffered from wrong predictions.

Definition Online classification model $f(x) = w^T \cdot x$ is the predictive model which is to be updated each time a new instance arrives.

Definition Decision function $\varphi(f(x))$ gives the predicted label based on the predictive model $f(x)$.

In the following we state an intuitive lemma for conditional probabilities and then we take advantage of it to propose our online algorithm.

Lemma 4.0.1 *Let $X_1, X_2, ..., X_d$ be independent random variables with corresponding distribution function of $f_i(x)$ for $i = 1, ..., d$. Also, let Y be a discrete random variable. If X_j and Y are independent, then:*

$$P(Y = y|(x_1, x_2, ..., x_j, ..., x_d)) = P(Y = y|(x_1, x_2, ..., x'_j, ..., x_d)) \tag{4.2}$$

where x'_j is randomly drawn from $f_j(x)$ distribution.

Proof:

By contradiction let assume:

$$P(Y = y|(x_1, x_2, ..., x_j, ..., x_d)) \neq P(Y = y|(x_1, x_2, ..., x'_j, ..., x_d)) \tag{4.3}$$

Since $X_1, X_2, ..., X_d$ are two by two independent then we have:

$$\begin{aligned} &P(Y = y|(x_1, x_2, ..., x_{j-1}, x_{j+1}, ..., x_d)) \cdot P((Y = y|x_j) \neq \\ &P(Y = y|(x_1, x_2, ..., x_{j-1}, x_{j+1}, ..., x_d)) \cdot P((Y = y|x'_j) \end{aligned} \tag{4.4}$$

Therefore the following should hold:

$$P((Y = y|x_j) \neq P((Y = y|x'_j) \tag{4.5}$$

the last equation is in contradiction to the independence assumption therefore 4.2 should hold.

Based on the idea of irrelevance in lemma 4.0.1 we present two algorithms for online feature importance ranking built on sensitivity measurement of the target variable where the challenge is

that the process should be incrementally executable.

Parametric method (Param)

Due to the volume and speed of data generation, a common practice in online learning is that the datapoints which have arrived to the system are assumed not to be saved or retrieved for future use (Crammer et al., 2006). Therefore the shuffling approach as presented for batch learning tasks as proposed in (Shen et al., 2008) is not usable.

Problem setting:

In our first algorithm we try to quantify the sensitivity of the classification output to the changes to the individual features in an incremental manner. The main idea to this approach is that according to lemma 4.0.1 the changes to the value of an irrelevant feature does not affect the output, at least in majority of times. The first step to establish our method is to generate ”realistic” random values for each feature to substitute in our classification model. Even though the exact distribution of the underlying feature may not be known, in practice many real-world continuous features follow a normal distribution. So, we could generate random values to substitute the current value of a feature and record the changes in the prediction based on the normality assumption of the features. On the other hand, we approximate the distribution of categorical features by generalized Bernoulli distribution. Therefore, by updating the probabilities of each category of a discrete/categorical feature, we would have its approximate distribution.

Goal of algorithm

Doing the aforementioned procedure incrementally and updating the sensitivity metric, we are able to rank the features. This ranking could be further used for feature reduction as explained in future sections.

One notable advantage of this approach is that it could work along with any type of classification method such as decision trees, neural networks, logistic regression and SVM. Regardless of the defined loss function, the optimization of interest could be any incremental method such as SGD, and Passive Aggressive (Crammer et al., 2006). The implementation details are discussed in Section 4.

Details of the parametric algorithm

As summarized in Algorithm 2, let X_j be the jth feature which is continuous and has normal distribution of $f_j(x) \sim N(\mu_j, \sigma_j)$. We could estimate and update the value of σ_j, μ_j as the variance and mean of its distribution in an incremental manner and as the new values arrive to the system. To observe the sensitivity of the model to the changes of X_j, we randomly generate x_j^{t*} with respect to distribution $f_j(x)$. The incremental calculations of distribution parameters according to (Finch, 2009) could be obtained as follows:

$$\mu_j^t = \frac{(t-1)\mu_j^{t-1} + x_j^t}{t} \tag{4.6}$$

$$\sigma_j^t = \left[\frac{(t-1)(\sigma_j^{(t-1)} + \mu_j^{(t-1)^2}) + x_j^{t^2}}{t} - (\mu_j^t)^2\right]^{1/2} \tag{4.7}$$

where μ_j^{t-1} and σ_j^{t-1} are the current mean and standard deviation of jth feature, respectively. Also, x_j^i is the newly arrived value of X_j, and t is total of number of instances used for training so far. The case for categorical features is more straightforward, because this type of variables supposedly follow a generalized Bernoulli distribution. For a given categorical feature X_j, the probability of category C_j^i, for $i = 1, ..., k^j$, where k^j is number of categories for jth feature k^j is determined by:

$$P(X_j = C_j^i) = \frac{number\ of\ observations\ with\ X_j = C_j^i}{total\ number\ of\ observations} \tag{4.8}$$

In implementation, we define two class of features, i.e. categorical and continuous. For each type of features the distribution parameters are updated accordingly. There are two loops in the Algorithm 2 where the first one is the training process that continues as the datapoints arrive to the system and the inside loop is used for sensitivity measurement updates.

Algorithm 2: Parametric feature ranking

Input: d = # of features

Initialization: *set $F = f_1, ..., f_d$, where $f_i = f(x_i)$ is the distribution function for the ith feature, $S = s_1, ..., s_d$ where s_i is the sensitivity parameter of ith feature, $R \in \mathbb{R}^d$ is the ranking list*

for *t=1,2,...* **do**
- receive instance x^t
- predict label as $\varphi(f(x^t))$, where $\varphi(f(x^t))$ is the predicting function criteria
- get the true label
- suffer an error penalty if the prediction is wrong
- **for** *j=1,2,...,d* **do**
  - update the feature's distribution statistics, e.g. f_j,
  - set $x_i^{t\prime} = x_i^t$ for i=1,...,j-1,j+1,...,d
  - set $x_j^{t\prime} = x_j^{t*}$ where x_j^{t*} is randomly drawn from distribution f_j
  - **if** $\varphi(f(x^t)) \neq \varphi(f(x^{t\prime}))$ **then**
    - $s_j = s_j + 1$
  - **end**
- **end**
- update R, the feature with lowest sensitivity is assigned the worst feature
- feature selection(optional)
- update classifier parameters

end

Output: feature importance rankings, classifier's parameters.

Non-parametric method (NoParam)

In an attempt to improve the earlier sensitivity measure algorithm, in this section we present a non-parametric modification of Param, called NoParam to address the following challenges. First, at the current implementation Param is based on the assumption that the continuous features have a normal distribution with unknown parameters. Also discrete features are assumed to have categorical distribution. Even though this assumption holds for many classification tasks involving natural variables, it may not always be true. In the other word, assuming a specific family of distributions may not always be possible.

Second, in Param we assumed that the underlying distribution is always the same and only the parameters needed to be updated. However, this assumption may not be true when *concept drift* happens (Tsymbal, 2004). An illustrative example of concept drift in which the x axis is the time and the variable Y is a continuous feature's value along the time axis is presented in Figure 4.1. The value of Y has different normal distribution statistics which change over time in three steps with an increase in the mean values. Therefore, assuming the same distribution for the feature all along the time axis is not correct.

Details of the Non-parametric algorithm

In order to contain the problem of concept drift we modify Algorithm 2, such that instead of estimating statistics for presumed distributions of the features, we make a sliding window with a constant length, say m as the representative values of a feature. Each time a new instance arrives into the model this representative window would be updated. In this way, the representative window always keeps the last m values of a feature. The rest of the feature ranking algorithm is

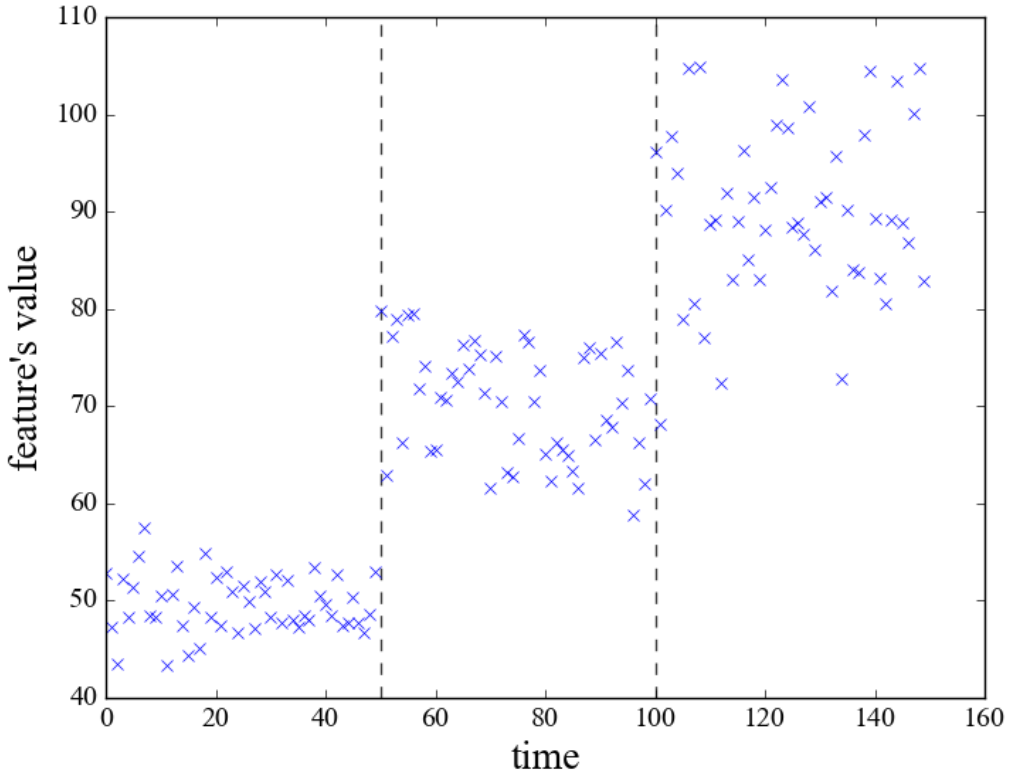


Figure 4.1: An example of a feature with changing distribution

similar to Algorithm 2, the only difference is that we draw the random values of a feature from the representative window instead of a probability distribution. The details of the NoParam algorithm is shown in Algorithm 3.

Feature selection

After obtaining the feature rankings, the actual feature selection needs to be done. Unlike batch feature selection where feature reduction is performed once and permanently, in online learning there are various ways to perform feature reduction. Since there is no agreed upon approach for feature selection in online setting, hereby we discuss two possible strategies, namely, permanent

Algorithm 3: Non-parametric feature ranking

Input: d = # of features

Initialization: $W \in \mathbb{R}^{d \times m}$ *where* d *is the number of features and* m *is the size of representative window,* $S = s_1, ..., s_d$ *where* s_i *is the sensitivity parameter of* i*th feature,* $R \in \mathbb{R}^d$ *is the ranking list*

for *t=1,2,...* **do**
- receive instance x^t
- predict label as $\varphi(f(x^t))$, where $\varphi(f(x^t))$ is the predicting function criteria
- get the true label
- suffer an error penalty if the prediction is wrong
- **for** *j=1,2,...,d* **do**
  - update the feature's distribution statistics, e.g. f_j,
  - set $x_i^t = x_i^{t\prime}$ set $x_i^{t\prime} = x_i^t$ for i=1,...,j-1,j+1,...,d
  - set $x_j^{t\prime} = x_j^{t*}$ where x_j^{t*} is randomly drawn from the feature's representative window, i.e. $\{w_{j1}, ..., w_{jm}\}$
  - **if** $\varphi(f(x^t)) \neq \varphi(f(x^{t\prime}))$ **then**
    - $s_j = s_j + 1$
  - **end**
  - update W
- **end**
- feature selection (optional)
- update classifier parameters

end

Output: feature importance rankings, classifier's parameters.

feature selection and dynamic feature selection.

Permanent feature selection

One way to reduce the number of features in online learning is to permanently remove those features deemed unimportant from the classification model. In this approach, a classification model is trained using all the available features, for an appropriate number of iterations. When there is sufficient evidence that rankings are statistically reliable, a portion of all the features are marked as unwanted and are permanently removed from the model. Consequently, in future iterations, the acquisition of such unwanted features will not be necessary anymore. Since the feature reduction

is done in multiple steps/timepoints the model will better adapt to the changes and the classification power, e.g. accuracy of the model will not drop significantly. Moreover, this method is easy to execute and it will not put any extra computational burden to the system. In implementation the feature reduction intervals are predefined and at each time a small portion of features is to be dropped from the model.

Dynamic feature selection

Another approach for online feature selection is to dynamically remove unimportant features from the model at each iteration. In this method the rankings are used to identify the important features, afterwards, unwanted features are removed from the model, and the reduced model is used for training and testing. The issue for dynamic feature selection is on how to determine the set of features to remain in the model. Bolon-Canedo et al. (2016) has proposed the use of thresholding in the ranking measure used in their model, i.e. the χ^2 value for each feature. In practice, however, it is important to keep the calculations as simple as possible. Because, in classification problems with high number of features the dynamic feature selection itself can slow down the training process. Therefore, we propose our dynamic feature selection method as a simple, yet intuitively efficient two step process. In the first step a minimum and maximum number of features to remain in the model is to be determined according to the size of the model and computational capabilities. Then before acquiring a new instance, the features with sensitivity of higher than $1 - (1/d)$ will be temporarily removed from the model. The sensitivity value for removal criteria comes from the fact that an increase in the number of features will usually cause less output sensitivity per feature. This process should be done any time an instance arrives to the system. The drawback of this approach is that it is computationally more expensive than permanent and incremental feature removal, and the sudden drop of large number of features may deter the efficiency of the classifier.

Empirical results

To evaluate the merits of the proposed methods, Param, and NoParam, we perform a number of experiments on benchmark datasets. To demonstrate generalizability, we chose benchmark datasets from various domains, including finance, chemistry, and web mining (Lichman, 2013). Moreover, we tried to select datasets with different feature spaces, including, all continuous and mixed features. More details about the the datasets could be found in Table 4.1.

Experimental setting

In order to compare the proposed algorithms with current available algorithms for online feature ranking, we perform the real-time/incremental training process on the datasets. In the experiments we use the multi-class SVM as the classification model and the incremental optimization is done by SGD with fixed stepsize of 0.1 which is implemented in sklearn package (Pedregosa et al., 2011).

In the first set of experiments, we compare our methods with online Chi2 as proposed in (Bolon-Canedo et al., 2016). Moreover, we add the random feature elimination as a second comparison method to see how our methods are better than randomly selecting features. Lastly, the graph for no-feature-selection is presented for sake of comparison. The accuracy of the method in real-time processing where feature selection is done in the permanent and incremental manner as discussed in Section 4 is presented in Figure 4.2. The feature removal intervals are each set after a quarter of training data has arrived. At each elimination timepoint a quarter of features are removed from the model according to the rankings. Each experiment is performed on 50 shuffles of the data and the presented graphs are the average accuracies over the repetitions. We conclude the method for which the model would have a better recovery from the feature reduction is a better ranking

approach. Two tailed paired T-test is performed on the accuracies, Param, and NoParam vs. other methods at the last iteration's obtained accuracies. The paired values of (p-value, t-stat) are shown in the Table 4.2. Where $p - value/2 < 0.05$, and $t - stat > 0$ we could conclude our method has significantly performed better.

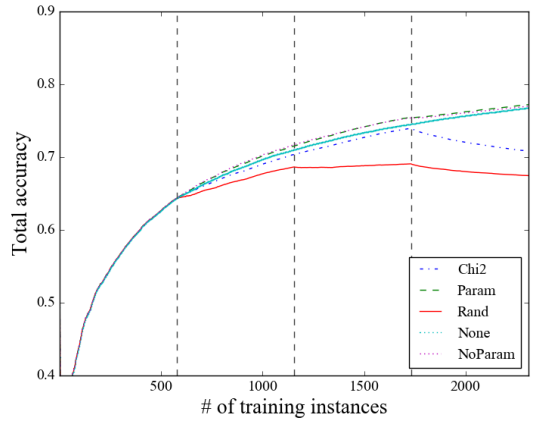
In the second set of experiments, we implemented the dynamic feature selection approach as stated in Section 4 using Param as the feature importance ranking method. The minimum and maximum number of features to be removed in the dynamic setting is set by 25% and 75% of the total features, respectively. The results are presented in Figure 4.3.

Table 4.1: Datasets' information

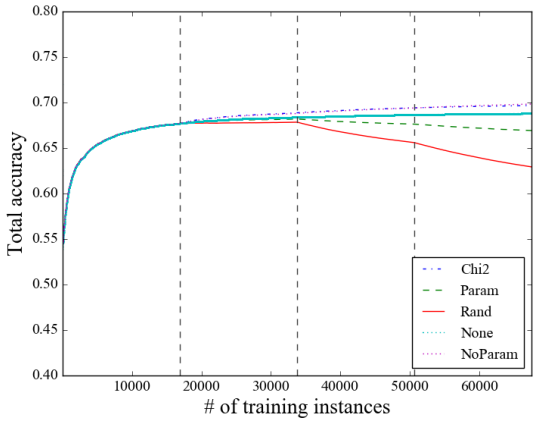
Dataset	number of instances	number of features	type of features	number of classes
Segment	2310	19	continuous	7
connect4	67557	126	categorical	3
Australian	690	14	mixed	2
phishing	2456	30	categorical	2
biodegradation	1055	41	mixed	2
mushrooms	8124	112	categorical	2

Discussion

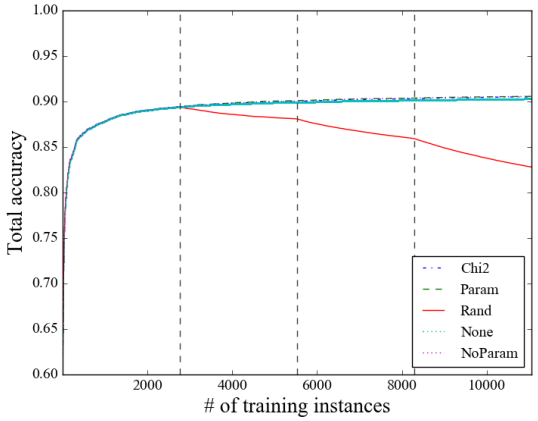
Comparison experimental results show sustained accuracy of the classifier as incremental feature reduction takes place (see Figure 4.2). As the results confirms, in majority of the test datasets, Param and NoParam perform significantly better than the other methods. More specifically, we would expect our methods work better than online Chi2 in continuous and mixed classification environments which is the case for Segment, Biodegrdation. In these cases, lower accuracy for online Chi2 might be due to the discretization procedure which might have caused the loss of in-



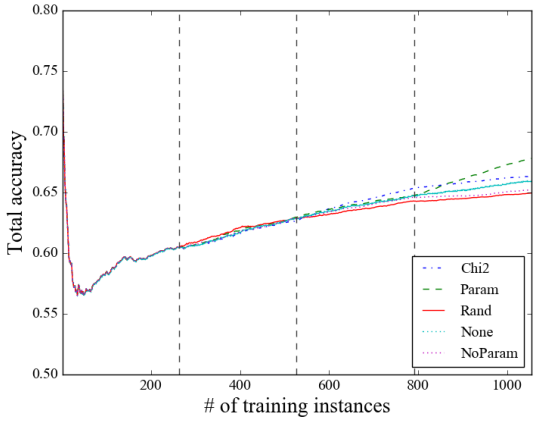
(a) Segment dataset



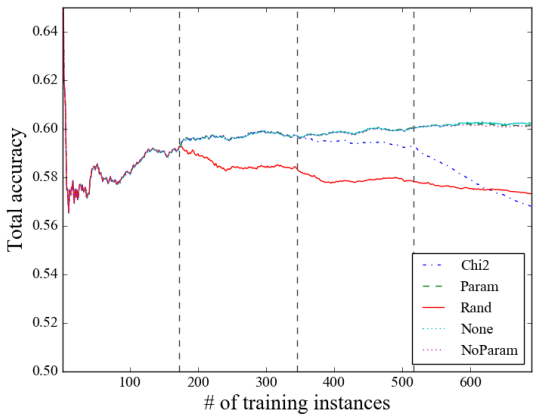
(b) connect4 dataset



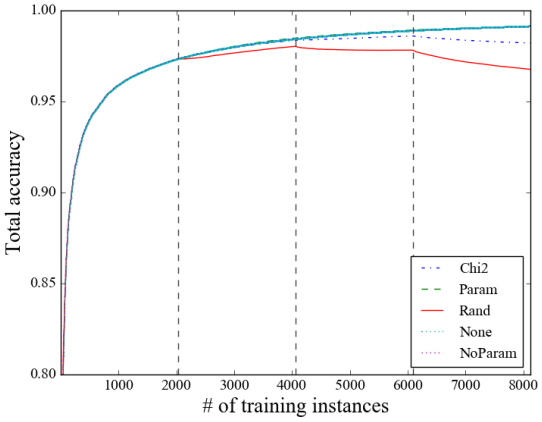
(c) Phishing dataset



(d) Biodegradation dataset

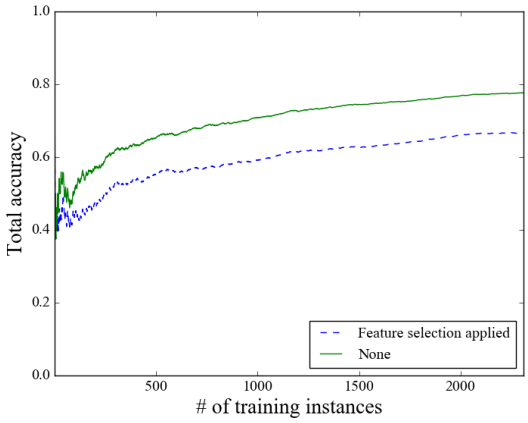


(e) Australian dataset

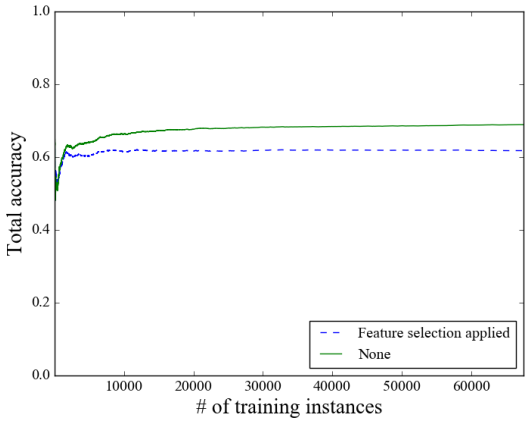


(f) Mushrooms dataset

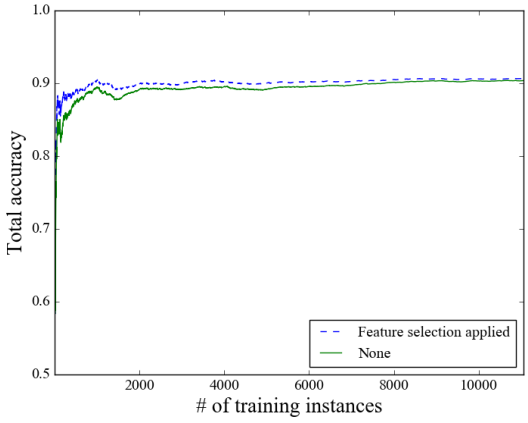
Figure 4.2: Real time accuracy changes with permanent feature reduction



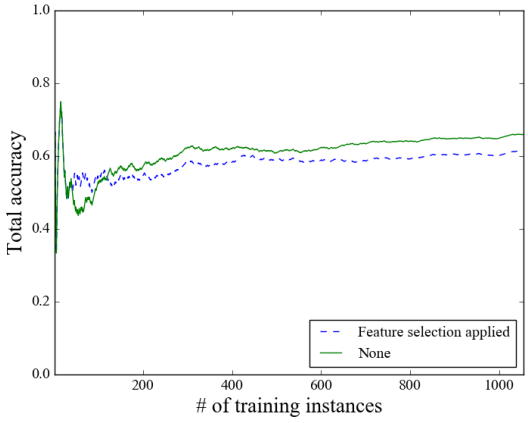
(a) Segment dataset



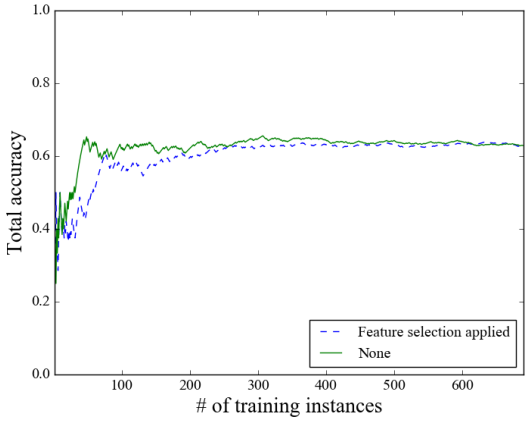
(b) connect4 dataset



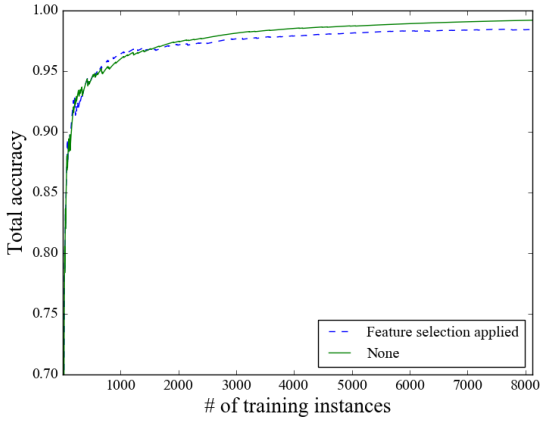
(c) Phishing dataset



(d) Biodegradation dataset



(e) Australian dataset



(f) Mushrooms dataset

Figure 4.3: Real time accuracy changes with dynamic feature selection

Table 4.2: Statistical analysis results

	Param			NoParam		
Dataset	vs. Chi2	vs. Rand	vs. None	vs. Chi2	vs. Rand	vs. None
Segment	(32.23, 0.00)	(14.70, 0.00)	(5.07, 0.00)	(33.29, 0.00)	(14.32, 0.00)	(2.22, 0.03)
connect4	(-87.79, 0.00)	(6.74, 0.00)	(-63.30, 0.00)	(9.25, 0.00)	(11.7, 0.00)	(51.75, 0.00)
Australian	(10.68, 0.00)	(6.57, 0.00)	(-2.11, 0.04)	(10.87, 0.00)	(6.68, 0.00)	(-2.23, 0.3)
phishing	(1.71, 0.09)	(12.55, 0.00)	(11.01, 0.00)	(1.52, 0.13)	(12.6, 0.00)	(11.63, 0.00)
biodegradation	(6.73, 0.00)	(6.49, 0.00)	(8.91, 0.00)	(-5.18, 0.00)	(6.49, 0.00)	(8.91, 0.00)
mushroom	(56.64, 0.00)	(14.32, 0.00)	(-0.04, 0.97)	(58.63, 0.00)	(14.09, 0.00)	(-5.35, 0.00)

formation. Moreover, unlike online Chi2, Param and Noparam are dependent to the classifier.

In the second set of experiments, the dynamic feature selection approach has been used to compare the accuracy of the classifier before and after feature selection. The results show that dynamic feature selection can deteriorate the efficiency of the classifier (see Figure 4.3). This can be seen in Segment, connect4 and Biodegradation datasets. This results might be caused by the instantaneous nature of dynamic feature selection, in which the classifier does not have time to adapt to the feature reduction. Therefore, considering both results, we would suggest the use of incremental and permanent feature selection in the real world applications. However, in case of concept drift and for the applications where the relationships between features and the classifier's outcome changes during the time, the use of dynamic feature reductions is recommended. Because a feature could be important at some time and become irrelevant in a future time. Even though, for our experimental purposes we could not access any benchmark online dataset with such characteristics to run comparisons.

Conclusion

In this chapter we presented two algorithms for online feature importance ranking based on sensitivity analysis of the classifier's output. The proposed methods are useful for a wide variety of classification problems without a need for prepossessing or discretization of the input variables, which makes the method superior to the comparable algorithms. Moreover, we discussed two major strategy for feature selection based on the obtained rankings. By performing experiments on the benchmark datasets we were able to compare the different feature selection methods in practice. Experimental results show that permanent and incremental feature reduction are in general better in terms of sustaining the quality of the predictions. Moreover, this approach does not add extra computational burden on the system. Even though the experimental results are promising, there are some limitations to the algorithms. In particular, the proposed sensitivity analysis does not take into account the interaction and correlation between the features. Therefore, when the features are highly correlated and specially in continuous feature spaces where direct measurement of the correlation matrix is possible, we suggest the use of SFR as proposed by (Razmjoo et al., 2017). The use of online algorithms is vital for many Big data applications, therefore, future works could be done on expansion of feature selection methods for other types of machine learning tasks such as online clustering and online network analysis.

CHAPTER 5: ONLINE CLASS-BASED FEATURE SELECTION

Preliminaries

In this chapter, we propose an ensemble-based approach to online classification and feature selection. In order to develop our online feature selection algorithm, we first need to review the procedure of online training of a classifier. In an online learning process, data is produced by an stochastic process, where observation x_t is the realization at tth timestamp. The goal of online data mining is to say something meaningful about the process. Broadly speaking, a classifier is said to be trained in an online manner, when the datapoints become available one by one, usually in a high speed, and the training model parameters are updated with an incremental optimization method such as Stochastic Gradient descent (SGD) (Bottou, 2012), ADAM (Kingma and Ba, 2014), Passive Aggressive (PA) (Crammer et al., 2006), to mention but a few.

Motivation for class-based approach

The motivation for the use of class-based feature selection and ensemble learning comes from the fact that in some practical cases assuming a unique distribution for an independent variable or a feature, as we will refer to it in the rest of the chapter, may not be true. A basic illustration of a classification task with two features and the dependent variable shown as labels on the points is shown in Figure 5.1. In this dataset, $X_1^1 \sim N(4,1)$ and $X_2^1 \sim U(1,6)$, also $X_1^2 \sim N(5,1)$ and $X_2^2 \sim U(1,1.4)$. This situation may happen due to statistical differences of the two populations of different classes in a classification domain.

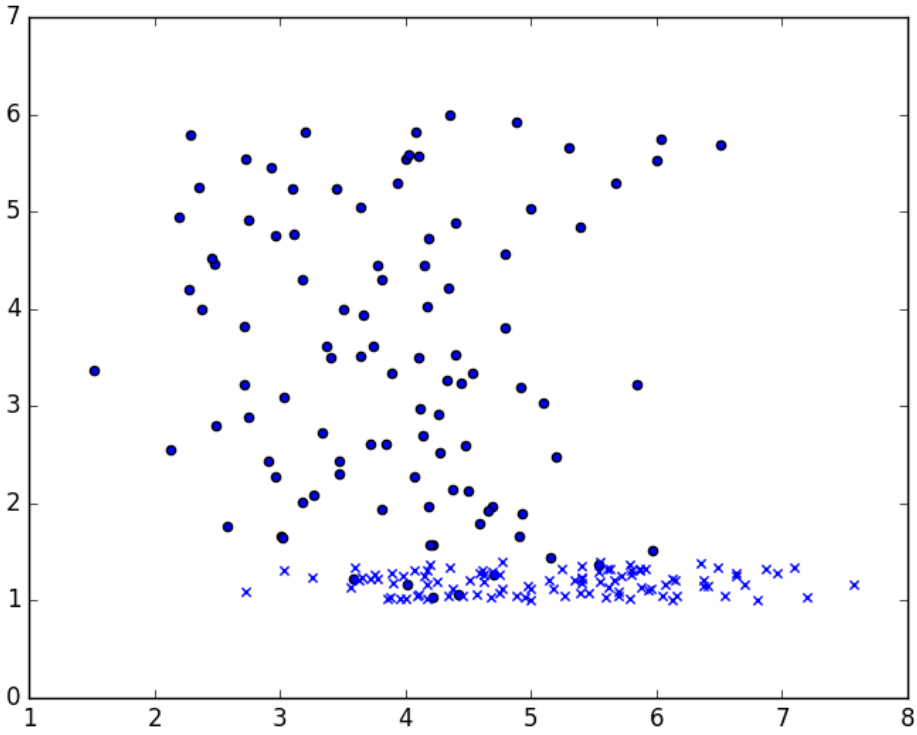


Figure 5.1: A two class classification problem, with varying underlying distribution of features

Ensemble learning

To deal with the different distributions for the distinct classes of data, we propose the use of ensemble learning. The main idea for ensemble of classifiers is to train multiple models concurrently and use voting methods to obtain a unique prediction. In our application, we aim to train k distinct classifiers where k is the total number of target classes. Therefore, for a binary classification problem, we would train two classifiers. Moreover, our strategy is based on class-based feature selection. A schema of how the the ensemble learning and class-based feature selection will be incorporated in our application is presented in Figure 5.2.

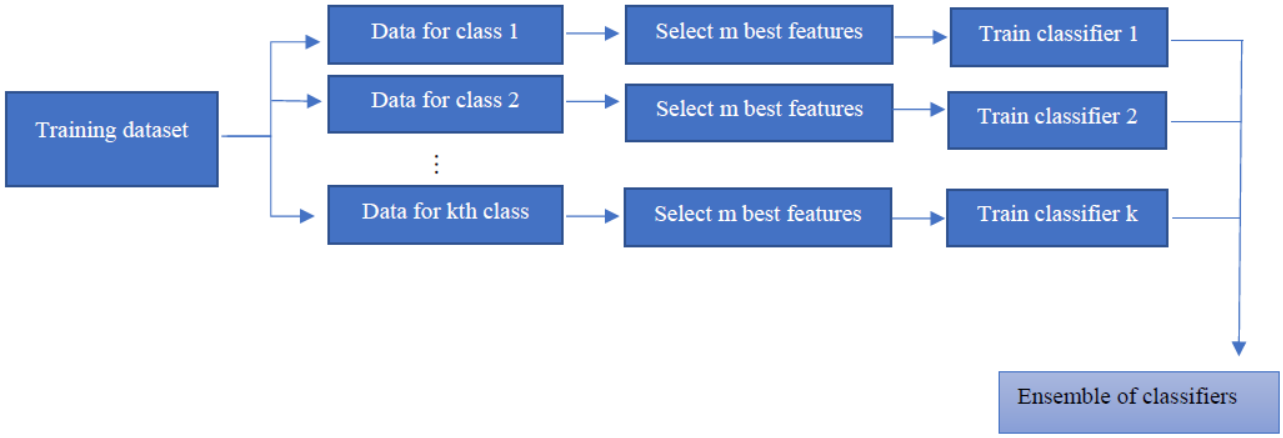


Figure 5.2: Schematic algorithm for class-based feature selection

In order to measure the goodness of features we take advantage of information theory regarding the definition of entropy in the values of a feature in a specific class. Intuitively, the features with less entropy and those whose value is consistent among a specific class is considered to be a good feature in distinguishing that class members. The definition of Shannon entropy (Cover and Thomas, 2012) for categorical features is presented in Equation 5.1.

$$E(X) = \sum_{i=1}^{l} P(x_i)I(x_i) = -\sum_{i=1}^{l} P(x_i)logP(x_i) \tag{5.1}$$

where X is the feature vector and x_i, for $i = 1, ..., l$ are the different values/categories of X. $P(x_j)$ is the probability of ith value happening and can be updated incrementally.

Application of Shannon entropy as described in Equation 5.1 is presented in Equation 5.2.

$$E(X|Y = y_j) = p(Y = y_i)H(X|Y = y_j) = -\sum_{i=1}^{t} p(x_i|y_j)logp(x_i|y_j) \tag{5.2}$$

This conditional definition of entropy is useful in class-based feature importance measure, because

it enables us to measure the entropy of a discrete feature for data of different classes.

Algorithm for class-based feature ranking

In many classification tasks, we deal with different types of features such as binary, categorical, and continuous features. Since the classification methods do not handle the categorical data directly, usually, some encoding methods such as Onehotencoder is applied to the data in preprocessing steps. In case of online training, this encoding should be done in real time. Beside this initial transformation, some algorithms only accept discrete variables, therefore, some type of discritization process takes place to transform the continuous features into discrete ones. In our method, however, we try to avoid discretization because it naturally leads to loss of information.

Let $U_d^k = \{x_d^t | y_t = k, for\ t = 1, ..., T\}$ for $k = 1, ..., K$ be the segmentation of the training data into different classes for dth feature, for $d = 1, 2, ..., D$. In this definition D is the total number of features and T is the size of the training set. Of course T may get as large as the stream of data let. Also, for any given d, $\sum_{k=1}^{K} |U_d^k| = t$ holds.

We initialize $R \in \Re^{K \times D}$ as the ranking matrix, for which each row of the matrix represents the feature rankings of each classifier. Therefore, for a binary classification, $k = 2$. In our implementation the features are ranked from the least important to the most important one.

We start by defining a measure for goodness of categorical features and how to incrementally update measure this value.

The class-based entropy of a categorical feature, shown by $E(U_d^k)$ is calculated based on Shannon entropy, as presented in Equation 5.2. In this definition, a feature with the least relative entropy is the most important among all features.

Since the proposed definition of entropy is originally applicable to discrete random variables, we

consider the alternative version of entropy measurement for a continuous variable X as following:

$$h(X) = -\int_x p(x)log(p(x))dx \tag{5.3}$$

Even though the direct use of differential entropy 5.3 is not trivial, the entropy of random variables with known distributions has been investigated in the literature (Lazo and Rathie, 1978). For example, this value for uniform distribution with parameters a and b as lower and upper bounds is $ln(b-a)$, the differential entropy of normal distribution with parameter μ and σ is calculated as $ln(\sigma\sqrt{2\pi e})$.

As it could be seen from these results, there is a direct relationship between standard deviation and differential entropy, which means higher variance leads to higher entropy for a continuous random variable. Therefore, considering that there is uncertainty about the type of underlying distribution in a classification problem, it is rather safe to use the standard deviation of a continuous variable in lieu of its entropy.

Now, let divide the features into categorical and continuous indexes as following:

Cat : $\{i|X_i\ is\ a\ categorical\ feature\}$

Con : $\{i|X_i\ is\ a\ continuous\ feature\}$

Moreover, for $d \in Cat$, $E(X_d) = -\sum_{i=1}^{c_d} p(x_d^i)logp(x_d^i)$ where c_d is the number of unique values of dth feature. Also, $E(X_d) \approx s_d$ for $d \in Con$, for which s_d is the standard deviation of X_d. different cases, for a arbitrary feature X_d, and class k there are different cases for relative class-based entropy:

1. $E(U_d^k)$ and $E(X_d)$ are both relatively small values.
2. $E(U_d^k)$ is a large value but $E(X_d)$ is relatively small.
3. $E(U_d^k)$ and $E(X_d)$ are both relatively large values.

4. $E(U_d^k)$ and $E(X_d)$ are both relatively large values.

Therefore by normalization $V_k(X_d) = E(U_d^k)/E(X_d)$. We opt to use $V_k(X_d)$ as a sensible variable to measure the goodness of a feature. Features with smallest $V_k(X_d)$ are recognized as the important ones for the dth classifier.

Revisiting the illustrative example from earlier this chapter, using the proposed algorithm, we obtain the accuracies, $C = 0.85$ of the general classifier (i. e. no feature selection is applied) , $C1 = 0.725$ feature 1 is important (class 0) , $C2 = 0.925$ feature 2 is important (class 1) , $C - classbased = 0.925$. In this example the application of class-based feature selection and ensemble learning has improved the accuracy (Figure 5.1).

Ensemble

One of the most important steps in ensemble learning is to decide on how to incorporate the results of the multiple classifiers into one final prediction to present to the decision maker. The implementation of this step depends on multiple characteristics of the classification tasks. First, for varying type of individual outputs of single classifiers we could plan differently. There are many suggestions on this matter in the literature that could be summarized as following:

- Majority vote: in this method the result is defined as the highest voted outcome.
- Highest probability schema: this approach is used when the probability of outcomes of the classifications are calculated. Then, the the outcome with highest confidence is chosen as the output of the ensemble.
- Average probability schema: in this method the average probability for each outcome across the single classifiers is calculated. The output is the outcome with highest average probability.

In practice, when base classifiers return the class probability all the three methods maybe used for aggregation of the individual predictions. On the other hand, if the probability of outcomes are unknown, the majority vote is used.

Experimental setting

In this section, we examine the practical value of the proposed method by performing experiments on public benchmark datasets. In our experiments we have selected multiple datasets from UCI repository (Lichman, 2013). Datasets are purposefully chosen from different domains to be a good representative of real-world applications. The datasets include a variety of features such as continuous, categorical, and mixed feature spaces. Also, both binary and multi-class classification problems will be investigated. The details on the experimental datasets could be found in Table 5.1.

The classification method could be selected regardless of the feature selection method. For our application, we used the logistic regression classifier with constant learning rate of 0.1. To imitate the online learning environment, SGD is used for the incremental optimization of the classifiers. Moreover, the highest probability schema is applied to aggregate the outcome of the ensemble of classifiers.

Comparison

In the first set of tests, we aim to compare the proposed feature selection methods with state-of-the-art feature selection method proposed by Razmjoo et al. (2017) and random feature elimination as suggested in (Wang et al., 2014). Moreover we add the learning accuracy for the classifier with

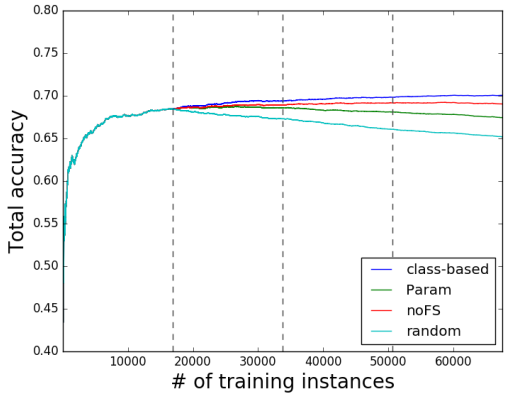
Table 5.1: Datasets' information

Dataset	number of instances	number of features	type of features	number of classes
Segment	2310	19	continuous	7
connect4	67557	126	categorical	3
Australian	690	14	mixed	2
phishing	2456	30	categorical	2
German	1000	20	mixed	2
krkp	3196	36	categorical	2

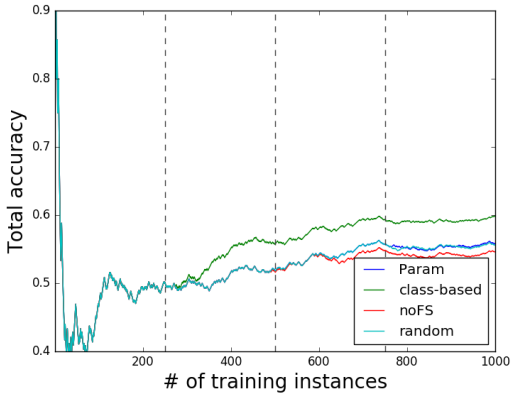
Table 5.2: Final accuracy with feature reduction applied

Dataset	No feature reduction	30% feature reduction	60% feature reduction
Segment	0.78918	0.75974	0.68961
connect4	0.69042	0.64779	0.41797
Australian	0.63333	0.58986	0.56667
phishing	0.89796	0.88973	0.86549
German	0.57300	0.59900	0.58800
krkp	0.91145	0.87891	0.87860

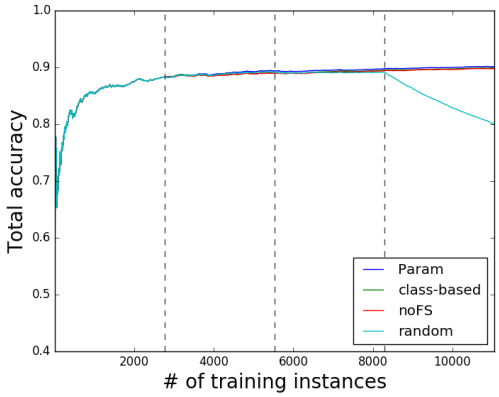
the full set of features present (noFS). In the proposed setting, the accuracy of the learning process is depicted in Figure 5.3. The feature selection is done be removing portions of features as pre-specified times. More specifically, a quarter of features are removed from the model after a quarter of training data has been used for training and feature importance calculation. The removal time stamps are also shown in the graphs. After the last feature removal step, only 25% of features are remained in the model.



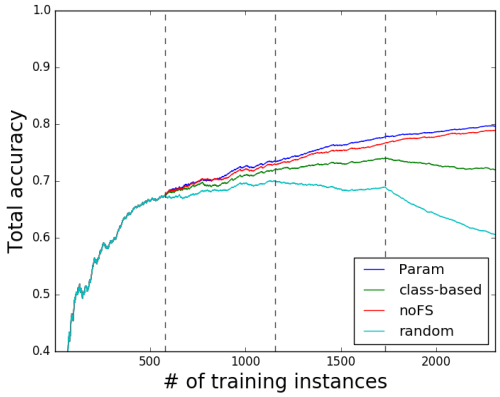
(a) connect4 dataset



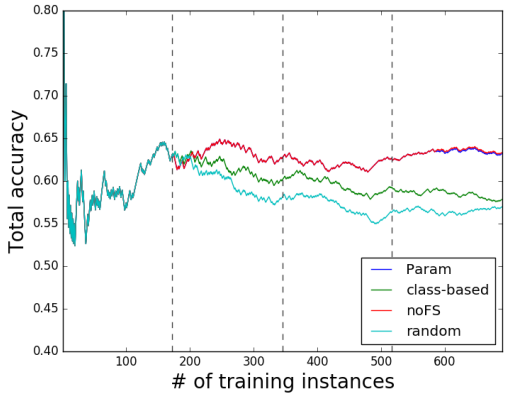
(b) German dataset



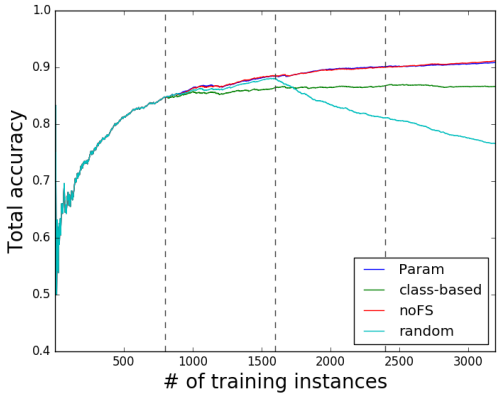
(c) phishing datast



(d) segment dataset



(e) Australian dataset

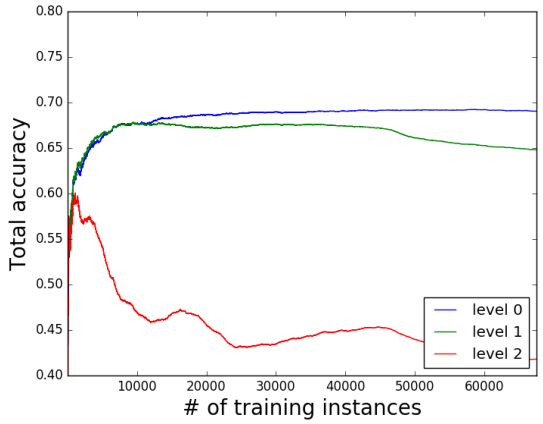


(f) krkp dataset

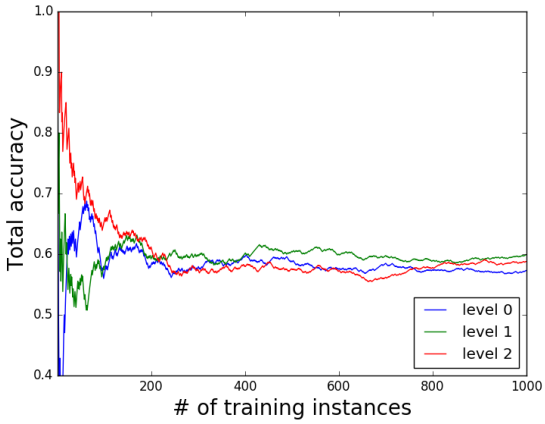
Figure 5.3: Comparison of realtime accuracy

Dynamic feature selection

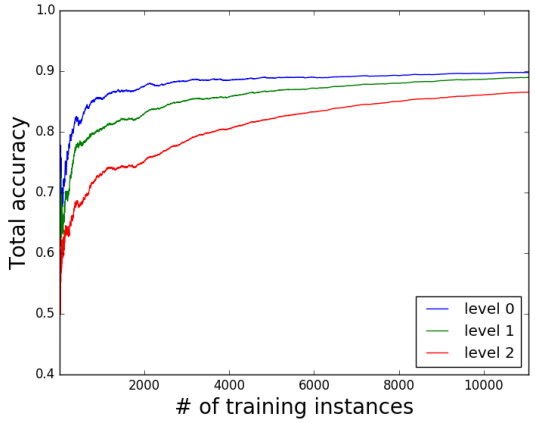
Since the proposed methods are ultimately designed to work in online learning enviroments, it is important to test their performance in the dynamic feature selection situations. As discussed in details by Razmjoo et al. (2017); Bolon-Canedo et al. (2016), feature selection in online setting could be done in a dynamic fashion. This means that at each training iteration, only a subset of features' parameters are updated. This approach is similar to the dropout method in neural network training, however, this time the removal is done as a feature selection strategy and to reduce the size of the model. The same reduced model would be used for the testing at the current timestamp. Of course, the removal of the features is based on their ranks obtained from the current feature rankings. One possible way to decide on how many of the features to be removed at each time stamp is basically selecting the k best features to remain in the model. In our experiments, for sake of comparison, we perform three levels of real-time dynamic feature reduction. The 0th level is the original model without feature reduction, the 1st level is the model with instant removal of 30% of features, and level 2 is the accuracy with dynamic removal of 60% of features.



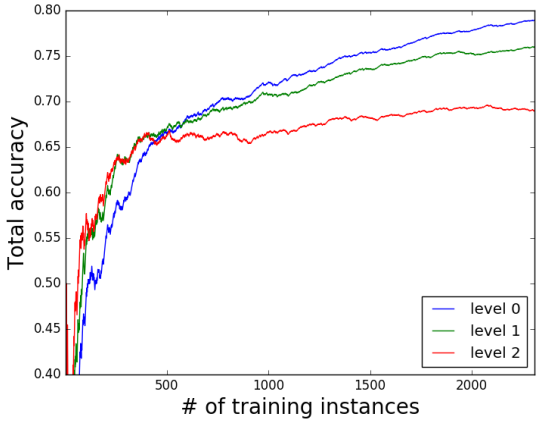
(a) connect4 dataset



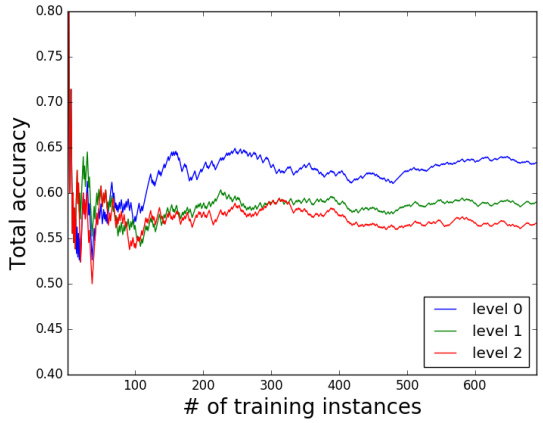
(b) German dataset



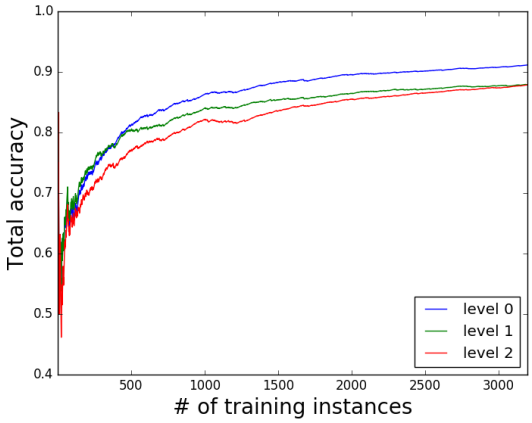
(c) phishing datast



(d) segment dataset



(e) Australian dataset



(f) krkp dataset

Figure 5.4: Comparison of realtime accuracy, with feature reduction applied

Results and Discussion

The results of the first set of experiments show that permanently reducing the model may be helpful to improve the accuracy. The real-time accuracy of the datasets German and connect-4 show that class-based feature importance ranking and feature selection results in better accuracy compares to Param and not applying any feature selection method. However, krkp, Australian and Segment dataset, the performance of Param is better, even when compared with the full model (noFS). Hence, we conclude that none of the methods guaranties the best performance on all the datasets.

The second set of experiments involving dynamic feature reduction results show that in general sudden removal of a large portion of features does decrease the accuracy of the model. However, the results vary from one dataset to another. In all the datasets except for German dynamic feature reduction deteriorates the results. 30% removal of model features seems to moderately affect the accuracy of the model in majority of the datasets. Another approach to reduce the size of the model could be a mix of permanent and dynamic feature selection. This could be done by permanently removing small portions of the features that are deemed irrelevant the the feature selection method and also dynamically removing a small portion of features at each step for both training and testing. The accuracies of the models at the final stage are reported in Table 5.2.

CHAPTER 6: CONCLUSION AND RECOMMENDATIONS

From an industrial engineer's perspective, modern machine learning methods could be used as substitute tools for optimization of very large datasets. Current volumes of data being produced in the industry make the use of closed form solutions rather impossible. Therefore, stochastic optimization methods would become better alternative for large scale predictive model training. Applications of machine learning methods in solving classical industrial engineering problems have been growing along several directions in the recent years.

Tiwari et al. (2018) have surveyed the applications of Big data solutions to supply chain management. According to this study, machine learning methods are used for predictive analysis in supply chain such as demand prediction, investment decision-making, and transportation management to mention but a few (Souza, 2014; Carbonneau et al., 2008). On the other hand, data mining and operations research share many optimization methods to solve real-world problem. Olafsson et al. (2008) have discussed in details the theoretical aspects of optimization methods used for machine learning problems and their relationship to conventional operations research methods. As an example, they discuss the implications of data mining methods to management of electronic services. Data mining methods have been applied to other areas of industrial engineering research too. For example, Agha and Alnahhal (2012) have proposed a predictive model to for body measurement for ergonomic design purposes.

Roughly speaking, research in machine learning field is performed in two directions: the first direction is the theory and algorithm development, and the other one is applying the currently available methodologies to extract meaningful patterns or perform analysis such as building predictive models using real-world data.

One of the current challenges of machine learning methods is developing new algorithms to learn from streams of data in real-time. Our contribution is to propose some new feature selection algo-

rithms for incremental ranking of the features in classification predictive models.

In business problem solving, the importance of feature ranking lies in the fact that in some applications merely knowing what are the factors influencing a dependent outcome helps in future plannings of the company. In scientific research, on the other hand, identifying contributing factors to the dependent output helps to better understand the cause and effects in the motivating research. Some of the well-known statistical approaches for feature/factor significance hypotheses testing are done by building regression or ANOVA models. However, these type of analysis makes some assumption about data which may not be essentially true. Also, the independence of variables is crucial for the hypotheses tests to be reliable.

In this dissertation, we discussed three methods that could be used along with online classifiers for obtaining the ranks of the features as well as performing feature reduction. In the first method, we presented a ranking method for classification tasks with continuous features (Razmjoo et al., 2017). The most important advantage of this method is the consideration for correlated features and the embedded mechanism to deal with this problem.

In the second method, we proposed two incremental feature ranking methods to measure the relevance of the features based on their impact on the target value (Alaleh Razmjoo, 2018). The proposed methods work independently from the classifier of interest which makes them versatile in sense of being applicable to classification models such as Artificial Neural Networks, SVMs, logistic regression, etc. More importantly, to use the proposed method, pre-processing is not a prerequisite which gives the method more practical value in the context of online learning. As discussed previously, current feature importance rankings do not directly offer a remedy to the situations where concept drift is present. For instance Chi2 assumes the same correlation for features and the output which is to be updated incrementally (Bolon-Canedo et al., 2016). In the same context, the method proposed by Razmjoo et al. (2017) assumes the same distribution when it updates the correlation between features without considering changes in the underlying feature space. In the second proposed method (NoParam) we use the sliding window concept to make sure only the

recently seen values of feature are used to update the feature importance measure.

Finally, we proposed a class-based feature ranking method to extract as much the insight as possible from the features. Even though these series of algorithms could be used on any online classification tasks, it is worth mentioning that for very sparse problems, the use of $l1$ norm as a feature reduction approach might be a computationally better choice. A good example could be the problem of document classification, where the number of features is extremely large. In another example, in the applications with very large number of features where deep learning models are being used importance ranking may become an extra computational burden. This is because the features are not being modeled directly but as combinations in the nodes. Also, deep learning models with a very large training set are deemed to work by figuring out the optimal subset of features during the training internally.

In our research we investigated two methods for feature reduction in real-time learning. However, the further experimental studies on different rules for dynamic feature selection as an standard method for feature selection in online learning could help with better results from online feature importance ranking methods.

Future research based on the online feature selection methods described in this dissertation could be pursued in two directions: theoretical development of online learning algorithms for Big data applications and industry research on the application of the proposed methods on streams of data in commercial environments.

With the growing focus of the industry on cloud computing and parallel computation on cluster of computers, the future theoretical research on the online algorithms could be focused on modification of the proposed methods or other state-of-the-art feature engineering algorithms to be applicable in parallel computing environments. On the other hand, the empirical research could be done using real-world streams of data to implement real-time training and testing systems and to integrate them with current semi-offline systems. Semi-offline machine learning models are those which are trained with batch learning methods and are used/tested in an online environment.

Therefore, the models are prone to loss of efficiency in presence of concept drift.

LIST OF REFERENCES

Abraham, A., Nath, B., and Mahanti, P. K. (2001). Hybrid intelligent systems for stock market analysis. In *International Conference on Computational Science*, pages 337–345. Springer.

Agha, S. R. and Alnahhal, M. J. (2012). Neural network and multiple linear regression to predict school children dimensions for ergonomic school furniture design. *Applied ergonomics*, 43(6):979–984.

Aizerman, A., Braverman, E., and Rozoner, L. (1964). Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control*, 25:821–837.

Alaleh Razmjoo, Petros Xanthopoulos, Q. P. Z. (2018). Feature importance ranking for classification in mixed online environments. *Annals of Operations Research*. In press, doi: 10.1007/s10479-018-2972-2.

Alpaydin, E. (2010). *Introduction to Machine Learning*. The MIT Press, 2nd edition.

Anguita, D., Ghio, A., Oneto, L., Parra Perez, X., and Reyes Ortiz, J. L. (2013). A public domain dataset for human activity recognition using smartphones. In *Proceedings of the 21th International European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, pages 437–442.

Banaee, H., Ahmed, M. U., and Loutfi, A. (2013). Data mining for wearable sensors in health monitoring systems: a review of recent trends and challenges. *Sensors*, 13(12):17472–17500.

Bengfort, B. and Kim, J. (2016). *Data Analytics with Hadoop*. O'Reilly Media, Incorporated.

Bi, J., Bennett, K., Embrechts, M., Breneman, C., and Song, M. (2003). Dimensionality reduction via sparse support vector machines. *Journal of Machine Learning Research*, 3(Mar):1229–1243.

Bifet, A. and Gavalda, R. (2007). Learning from time-changing data with adaptive windowing. In *Proceedings of the 2007 SIAM international conference on data mining*, pages 443–448. SIAM.

Bifet, A., Holmes, G., Kirkby, R., and Pfahringer, B. (2010). Moa: Massive online analysis. *Journal of Machine Learning Research*, 11(May):1601–1604.

Biggs, D., De Ville, B., and Suen, E. (1991). A method of choosing multiway partitions for classification and decision trees. *Journal of Applied Statistics*, 18(1):49–62.

Bobadilla, J., Ortega, F., Hernando, A., and Gutiérrez, A. (2013). Recommender systems survey. *Knowledge-based systems*, 46:109–132.

Bollen, J., Mao, H., and Zeng, X. (2011). Twitter mood predicts the stock market. *Journal of computational science*, 2(1):1–8.

Bolon-Canedo, V., Fernández-Francos, D., Peteiro-Barral, D., Alonso-Betanzos, A., Guijarro-Berdiñas, B., and Sánchez-Maroño, N. (2016). A unified pipeline for online feature selection and classification. *Expert Systems with Applications*, 55:532–545.

Bordes, A., Bottou, L., Gallinari, P., and Weston, J. (2007). Solving multiclass support vector machines with larank. In *Proceedings of the 24th international conference on Machine learning*, pages 89–96. ACM.

Bordes, A., Ertekin, S., Weston, J., and Bottou, L. (2005). Fast kernel classifiers with online and active learning. *Journal of Machine Learning Research*, 6(Sep):1579–1619.

Bose, I. and Mahapatra, R. K. (2001). Business data mining—a machine learning perspective. *Information & management*, 39(3):211–225.

Bottou, L. (2010). Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer.

Bottou, L. (2012). Stochastic gradient descent tricks. In *Neural networks: Tricks of the trade*, pages 421–436. Springer.

Breiman, L. (1996). Bagging predictors. *Machine learning*, 24(2):123–140.

Breiman, L. (2001). Random forests. *Machine learning*, 45(1):5–32.

Breiman, L. et al. (1998). Arcing classifier (with discussion and a rejoinder by the author). *The annals of statistics*, 26(3):801–849.

Bruzzone, L. and Prieto, D. F. (1999). An incremental-learning neural network for the classification of remote-sensing images. *Pattern Recognition Letters*, 20(11-13):1241–1248.

Cantú-Paz, E., Newsam, S., and Kamath, C. (2004). Feature selection in scientific applications. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 788–793. ACM.

Carbonneau, R., Laframboise, K., and Vahidov, R. (2008). Application of machine learning techniques for supply chain demand forecasting. *European Journal of Operational Research*, 184(3):1140–1154.

Carvalho, V. R. and Cohen, W. W. (2006). Single-pass online learning: Performance, voting schemes and online feature selection. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 548–553. ACM.

Chang, C.-C. and Lin, C.-J. (2011). Libsvm: a library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)*, 2(3):27.

Cohen, L., Avrahami-Bakish, G., Last, M., Kandel, A., and Kipersztok, O. (2008). Real-time data mining of non-stationary data streams from sensor networks. *Information Fusion*, 9(3):344–353.

Cover, T. M. and Thomas, J. A. (2012). *Elements of information theory*. John Wiley & Sons.

Cox, D. R. (1958). The regression analysis of binary sequences. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 215–242.

Crammer, K., Dekel, O., Keshet, J., Shalev-Shwartz, S., and Singer, Y. (2006). Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7(Mar):551–585.

Czitrom, V. (1999). One-factor-at-a-time versus designed experiments. *The American Statistician*, 53(2):126–131.

Dash, M. and Liu, H. (1997). Feature selection for classification. *Intelligent data analysis*, 1(1-4):131–156.

de Barros, R. S. M. and de Carvalho Santos, S. G. T. (2018). A large-scale comparison of concept drift detectors. *Information Sciences*.

de Lannoy, G., François, D., and Verleysen, M. (2011). Class-specific feature selection for one-against-all multiclass svms. In *ESANN 2011, 19th European Symposium on Artificial Neural Networks, Bruges, Belgium, April 27-29, 2011, Proceedings*.

de Lima Cabral, D. R. and de Barros, R. S. M. (2018). Concept drift detection based on fisher's exact test. *Information Sciences*, 442:220–234.

Deaton, D. W. and Gabriel, R. G. (2001). Method and system for generating incentives based on substantially real-time product purchase information. US Patent 6,292,786.

Deo, R. C. (2015). Machine learning in medicine. *Circulation*, 132(20):1920–1930.

Desai, B., Andhale, P., Rege, M., and Yu, Q. (2012). *Biclustering and Feature Selection Techniques in Bioinformatics*, pages 280–287. Springer Berlin Heidelberg, Berlin, Heidelberg.

Dowdy, S., Wearden, S., and Chilko, D. (2011). *Statistics for research*, volume 512. John Wiley & Sons.

Duda, R. O., Hart, P. E., and Stork, D. G. (1973). *Pattern classification*. Wiley, New York.

Dudani, S. A. (1976). The distance-weighted k-nearest-neighbor rule. *IEEE Transactions on Systems, Man, and Cybernetics*, 4(SMC-6):325–327.

Fan, Y.-J. and Chaovalitwongse, W. A. (2010). Optimizing feature selection to improve medical diagnosis. *Annals of Operations Research*, 174(1):169–183.

Fausett, L. (1994). *Fundamentals of neural networks: architectures, algorithms, and applications*. Prentice-Hall, Inc.

Finch, T. (2009). Incremental calculation of weighted mean and variance.

Gaber, M. M., Zaslavsky, A., and Krishnaswamy, S. (2005). Mining data streams: a review. *ACM Sigmod Record*, 34(2):18–26.

Gama, J., Medas, P., Castillo, G., and Rodrigues, P. (2004). Learning with drift detection. In *Brazilian symposium on artificial intelligence*, pages 286–295. Springer.

Geurts, P., Ernst, D., and Wehenkel, L. (2006). Extremely randomized trees. *Machine learning*, 63(1):3–42.

GonçAlves Jr, P. M. and De Barros, R. S. M. (2013). Rcd: A recurring concept drift framework. *Pattern Recognition Letters*, 34(9):1018–1025.

Gonçalves Jr, P. M., de Carvalho Santos, S. G., Barros, R. S., and Vieira, D. C. (2014). A comparative study on concept drift detectors. *Expert Systems with Applications*, 41(18):8144–8156.

Graepel, T., Candela, J. Q., Borchert, T., and Herbrich, R. (2010). Web-scale bayesian click-through rate prediction for sponsored search advertising in microsoft's bing search engine. In *Proceedings of the 27th International Conference on Machine Learning ICML 2010, Invited Applications Track (unreviewed, to appear)*. Omnipress.

Guyon, I. and Elisseeff, A. (2003). An introduction to variable and feature selection. *J. Mach. Learn. Res.*, 3:1157–1182.

Guyon, I., Weston, J., Barnhill, S., and Vapnik, V. (2002). Gene selection for cancer classification using support vector machines. *Machine learning*, 46(1-3):389–422.

Halawa, H., Ripeanu, M., Beznosov, K., Coskun, B., and Liu, M. (2017). An early warning system for suspicious accounts. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pages 51–52. ACM.

Hastie, T., Rosset, S., Zhu, J., and Zou, H. (2009). Multi-class adaboost. *Statistics and its Interface*, 2(3):349–360.

Haussler, D. and Warmuth, M. (1993). The probably approximately correct (pac) and other learning models. In *Foundations of Knowledge Acquisition*, pages 291–312. Springer.

He, Z. and Wang, H. (2012). A comparison and improvement of online learning algorithms for sequence labeling. *Proceedings of COLING 2012*, pages 1147–1162.

Hoffman, J., Rodner, E., Donahue, J., Darrell, T., and Saenko, K. (2013). Efficient learning of domain-invariant image representations. *arXiv preprint arXiv:1301.3224*.

Hotelling, H. (1933). Analysis of a complex of statistical variables into principal components. *Journal of educational psychology*, 24(6):417.

Hsu, C.-W. and Lin, C.-J. (2002). A comparison of methods for multiclass support vector machines. *IEEE transactions on Neural Networks*, 13(2):415–425.

Huang, J. and Xiao, M. (2018). State of the art on road traffic sensing and learning based on mobile user network log data. *Neurocomputing*, 278:110–118.

Huang, Q., Tao, D., Li, X., Jin, L., and Wei, G. (2011). Exploiting local coherent patterns for unsupervised feature ranking. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 41(6):1471–1482.

Iooss, B. and Lemaître, P. (2015). *A Review on Global Sensitivity Analysis Methods*, pages 101–122. Springer US, Boston, MA.

Joachims, T. (1998). Text categorization with support vector machines: Learning with many relevant features. In *European conference on machine learning*, pages 137–142. Springer.

Jokinen, J. V., Blants, L., Pitkänen, R., Pienimäki, S., Mattila, J., and Suomela, R. (2008). Real-time wireless e-coupon (promotion) definition based on available segment. US Patent 7,343,317.

Jorge, J. and Paredes, R. (2018). Passive-aggressive online learning with nonlinear embeddings. *Pattern Recognition*, 79:162–171.

Kabir, M. M., Islam, M. M., and Murase, K. (2010). A new wrapper feature selection approach using neural network. *Neurocomputing*, 73(16-18):3273–3283.

Kalousis, A., Prados, J., and Hilario, M. (2007). Stability of feature selection algorithms: a study on high-dimensional spaces. *Knowledge and information systems*, 12(1):95–116.

Kamkar, I., Gupta, S. K., Phung, D., and Venkatesh, S. (2015). Stable feature selection with support vector machines. In *Australasian Joint Conference on Artificial Intelligence*, pages 298–308. Springer.

Kar, P., Narasimhan, H., and Jain, P. (2014). Online and stochastic gradient methods for non-decomposable loss functions. In *Advances in Neural Information Processing Systems*, pages 694–702.

Kargupta, H., Bhargava, R., Liu, K., Powers, M., Blair, P., Bushra, S., Dull, J., Sarkar, K., Klein, M., Vasa, M., et al. (2004). Vedas: A mobile and distributed data stream mining system for

real-time vehicle monitoring. In *Proceedings of the 2004 SIAM International Conference on Data Mining*, pages 300–311. SIAM.

Katakis, I., Tsoumakas, G., and Vlahavas, I. (2005). On the utility of incremental feature selection for the classification of textual data streams. *Advances in Informatics*, pages 338–348.

Keller, J. M., Gray, M. R., and Givens, J. A. (1985). A fuzzy k-nearest neighbor algorithm. *IEEE transactions on systems, man, and cybernetics*, SMC-15(4):580–585.

Kim, M., Lee, Y.-H., and Han, C. (2000). Real-time classification of petroleum products using near-infrared spectra. *Computers & Chemical Engineering*, 24(2-7):513–517.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kivinen, J., Smola, A. J., and Williamson, R. C. (2010). Online learning with kernels. *IEEE Transactions on Signal Processing*, 100(10):1–12.

Kluger, Y., Basri, R., Chang, J. T., and Gerstein, M. (2003). Spectral biclustering of microarray data: coclustering genes and conditions. *Genome research*, 13(4):703–716.

Kohavi, R. and John, G. H. (1997). Wrappers for feature subset selection. *Artificial intelligence*, 97(1-2):273–324.

Kohavi, R. and Quinlan, J. R. (2002). Data mining tasks and methods: Classification: decision-tree discovery. In *Handbook of data mining and knowledge discovery*, pages 267–276. Oxford University Press, Inc.

Krempl, G., Žliobaite, I., Brzeziński, D., Hüllermeier, E., Last, M., Lemaire, V., Noack, T., Shaker, A., Sievi, S., Spiliopoulou, M., et al. (2014). Open challenges for data stream mining research. *ACM SIGKDD explorations newsletter*, 16(1):1–10.

Larose, D. T. (2005). k-nearest neighbor algorithm. *Discovering knowledge in data: An introduction to data mining*, pages 90–106.

Lazarus, M. A., Caid, W. R., Pugh, R. S., Kindig, B. D., Russell, G. S., Brown, K. B., Dunning, T. E., and Carleton, J. L. (2000). System and method for optimal adaptive matching of users to most relevant entity and information in real-time. US Patent 6,134,532.

Lazo, A. V. and Rathie, P. (1978). On the entropy of continuous probability distributions (corresp.). *IEEE Transactions on Information Theory*, 24(1):120–122.

Le Thi, H. A. and Nguyen, M. C. (2017). Dca based algorithms for feature selection in multi-class support vector machine. *Annals of Operations Research*, 249(1-2):273–300.

Legendre, A. (1805). *Nouvelles méthodes pour la détermination des orbites des comètes*. Nineteenth Century Collections Online (NCCO): Science, Technology, and Medicine: 1780-1925. F. Didot.

Leskovec, J., Rajaraman, A., and Ullman, J. D. (2014). *Mining of massive datasets*. Cambridge university press.

Lichman, M. (2013). UCI machine learning repository.

Lin, Y., Guo, H., and Hu, J. (2013). An svm-based approach for stock market trend prediction. In *Neural Networks (IJCNN), The 2013 International Joint Conference on*, pages 1–7. IEEE.

Liu, H. and Setiono, R. (1995). Chi2: Feature selection and discretization of numeric attributes. In *ICTAI*, pages 388–391.

Liu, Y., Li, H., Peng, G., Lv, B., and Zhang, C. (2015). Online purchaser segmentation and promotion strategy selection: evidence from chinese e-commerce market. *Annals of Operations Research*, 233(1):263–279.

Lunga, D. and Marwala, T. (2006). Online forecasting of stock market movement direction using the improved incremental algorithm. In *International Conference on Neural Information Processing*, pages 440–449. Springer.

Ma, J., Saul, L. K., Savage, S., and Voelker, G. M. (2009). Identifying suspicious urls: an application of large-scale online learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 681–688. ACM.

Maldonado, S. and Weber, R. (2009). A wrapper method for feature selection using support vector machines. *Information Sciences*, 179(13):2208–2217.

McLachlan, G. (2004). *Discriminant analysis and statistical pattern recognition*, volume 544. John Wiley & Sons.

Michalski, R. S., Carbonell, J. G., and Mitchell, T. M. (2013). *Machine learning: An artificial intelligence approach*. Springer Science & Business Media.

Nair, B. B., Mohandas, V., and Sakthivel, N. (2010). A decision tree-rough set hybrid system for stock market trend prediction. *International Journal of Computer Applications*, 6(9):1–6.

Ngai, E. W., Xiu, L., and Chau, D. C. (2009). Application of data mining techniques in customer relationship management: A literature review and classification. *Expert systems with applications*, 36(2):2592–2602.

Nguyen, H.-L., Woon, Y.-K., Ng, W.-K., and Wan, L. (2012). Heterogeneous ensemble for feature drifts in data streams. *Advances in Knowledge Discovery and Data Mining*, pages 1–12.

Nguyen, H. M., Cooper, E. W., and Kamei, K. (2011). Online learning from imbalanced data streams. In *Soft Computing and Pattern Recognition (SoCPaR), 2011 International Conference of*, pages 347–352. IEEE.

Nina, Z. and Wang, L. (2008). Class-dependent feature selection for face recognition. In *International Conference on Neural Information Processing*, pages 551–558. Springer.

Olafsson, S., Li, X., and Wu, S. (2008). Operations research and data mining. *European Journal of Operational Research*, 187(3):1429–1448.

Page, E. S. (1954). Continuous inspection schemes. *Biometrika*, 41(1/2):100–115.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Perkins, S. and Theiler, J. (2003). Online feature selection using grafting. In *ICML*, pages 592–599.

Polyak, B. T. and Juditsky, A. B. (1992). Acceleration of stochastic approximation by averaging. *SIAM Journal on Control and Optimization*, 30(4):838–855.

Prasad, A. M., Iverson, L. R., and Liaw, A. (2006). Newer classification and regression tree techniques: bagging and random forests for ecological prediction. *Ecosystems*, 9(2):181–199.

Quinlan, J. R. (1986). Induction of decision trees. *Machine learning*, 1(1):81–106.

Quinlan, J. R. (2014). *C4. 5: programs for machine learning*. Elsevier.

Ramírez-Gallego, S., Krawczyk, B., García, S., Woźniak, M., and Herrera, F. (2017). A survey on data preprocessing for data stream mining: current status and future directions. *Neurocomputing*, 239:39–57.

Razmjoo, A., Xanthopoulos, P., and Zheng, Q. P. (2017). Online feature importance ranking based on sensitivity analysis. *Expert Systems with Applications*, 85:397–406.

Robnik-Šikonja, M. and Kononenko, I. (2003). Theoretical and empirical analysis of relieff and rrelieff. *Machine learning*, 53(1-2):23–69.

Roughan, M., Sen, S., Spatscheck, O., and Duffield, N. (2004). Class-of-service mapping for qos: a statistical signature-based approach to ip traffic classification. In *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, pages 135–148. ACM.

Roy, A. (2015). Automated online feature selection and learning from high-dimensional streaming data using an ensemble of kohonen neurons. In *Neural Networks (IJCNN), 2015 International Joint Conference on*, pages 1–8. IEEE.

Roy, A., Mackin, P. D., and Mukhopadhyay, S. (2013). Methods for pattern selection, class-specific feature selection and classification for automated learning. *Neural Networks*, 41:113–129.

Russell, S. J. and Norvig, P. (2016). *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited,.

Saltelli, A. and Annoni, P. (2010). How to avoid a perfunctory sensitivity analysis. *Environmental Modelling & Software*, 25(12):1508–1517.

Sayed-Mouchaweh, M. (2016). *Learning from Data Streams in Dynamic Environments*. Springer.

Sebastiani, F. (2002). Machine learning in automated text categorization. *ACM computing surveys (CSUR)*, 34(1):1–47.

Seref, O., Fan, Y.-J., Borenstein, E., and Chaovalitwongse, W. A. (2018). Information-theoretic feature selection with discrete k-median clustering. *Annals of Operations Research*, 263(1-2):93–118.

Şeref, O., Razzaghi, T., and Xanthopoulos, P. (2017). Weighted relaxed support vector machines. *Annals of Operations Research*, 249(1-2):235–271.

Sethi, T. S. and Kantardzic, M. (2017). On the reliable detection of concept drift from streaming unlabeled data. *Expert Systems with Applications*, 82:77–99.

Shalev-Shwartz, S., Singer, Y., Srebro, N., and Cotter, A. (2011). Pegasos: Primal estimated sub-gradient solver for svm. *Mathematical programming*, 127(1):3–30.

Shen, K.-Q., Ong, C.-J., Li, X.-P., and Wilder-Smith, E. P. (2008). Feature selection via sensitivity analysis of svm probabilistic outputs. *Machine Learning*, 70(1):1–20.

Souza, G. C. (2014). Supply chain analytics. *Business Horizons*, 57(5):595–605.

Suykens, J. A. and Vandewalle, J. (1999). Least squares support vector machine classifiers. *Neural processing letters*, 9(3):293–300.

Thomopoulos, N. T. (2012). *Essentials of Monte Carlo simulation: Statistical methods for building simulation models*. Springer Science & Business Media.

Tian, X., Sun, Q., Huang, X., and Ma, Y. (2009). A dynamic online traffic classification methodology based on data stream mining. In *Computer Science and Information Engineering, 2009 WRI World Congress on*, volume 1, pages 298–302. IEEE.

Tiwari, S., Wee, H., and Daryanto, Y. (2018). Big data analytics in supply chain management between 2010 and 2016: Insights to industries. *Computers & Industrial Engineering*, 115:319–330.

Tsymbal, A. (2004). The problem of concept drift: definitions and related work. *Computer Science Department, Trinity College Dublin*.

Vasilescu, M. A. O. and Terzopoulos, D. (2003). Multilinear subspace analysis of image ensembles. In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, volume 2, pages II–93. IEEE.

Walker, S. H. and Duncan, D. B. (1967). Estimation of the probability of an event as a function of several independent variables. *Biometrika*, 54(1-2):167–179.

Wang, J., Wang, M., Li, P., Liu, L., Zhao, Z., Hu, X., and Wu, X. (2015). Online feature selection with group structure analysis. *IEEE Transactions on Knowledge and Data Engineering*, 27(11):3029–3041.

Wang, J., Zhao, P., Hoi, S. C., and Jin, R. (2014). Online feature selection and its applications. *IEEE Transactions on Knowledge and Data Engineering*, 26(3):698–710.

Wang, L., Ji, H.-B., and Jin, Y. (2013a). Fuzzy passive–aggressive classification: A robust and efficient algorithm for online classification problems. *Information Sciences*, 220:46–63.

Wang, S., Minku, L. L., and Yao, X. (2013b). A learning framework for online class imbalance learning. In *Computational Intelligence and Ensemble Learning (CIEL), 2013 IEEE Symposium on*, pages 36–45. IEEE.

Wang, S. and Yao, X. (2012). Multiclass imbalance problems: Analysis and potential solutions. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 42(4):1119–1130.

Wang, Z., Crammer, K., and Vucetic, S. (2010). Multi-class pegasos on a budget. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 1143–1150. Citeseer.

Wang, Z. and Vucetic, S. (2010). Online passive-aggressive algorithms on a budget. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 908–915.

Wold, S., Esbensen, K., and Geladi, P. (1987). Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52.

Wu, W. H., Bui, A. A., Batalin, M. A., Liu, D., and Kaiser, W. J. (2007). Incremental diagnosis method for intelligent wearable sensor systems. *IEEE Transactions on Information Technology in Biomedicine*, 11(5):553–562.

Wu, X., Yu, K., Ding, W., Wang, H., and Zhu, X. (2013). Online feature selection with streaming features. *IEEE transactions on pattern analysis and machine intelligence*, 35(5):1178–1192.

Ye, Q., Zhang, Z., and Law, R. (2009). Sentiment classification of online reviews to travel destinations by supervised machine learning approaches. *Expert systems with applications*, 36(3):6527–6535.

Yu, L. and Liu, H. (2003). Feature selection for high-dimensional data: A fast correlation-based filter solution. In *Proceedings of the 20th international conference on machine learning (ICML-03)*, pages 856–863.

Yuan, R., Li, Z., Guan, X., and Xu, L. (2010). An svm-based machine learning method for accurate internet traffic classification. *Information Systems Frontiers*, 12(2):149–156.

Zaharia, M., Das, T., Li, H., Hunter, T., Shenker, S., and Stoica, I. (2013). Discretized streams: Fault-tolerant streaming computation at scale. In *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles*, pages 423–438. ACM.

Zhang, G. P. (2000). Neural networks for classification: a survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 30(4):451–462.

Zhou, N. and Wang, L. (2006). A novel support vector machine with class-dependent features for biomedical data. In *Systems, Man and Cybernetics, 2006. SMC'06. IEEE International Conference on*, volume 2, pages 1666–1670. IEEE.

Zhu, Z., Ong, Y.-S., and Dash, M. (2007). Wrapper–filter feature selection algorithm using a

memetic framework. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 37(1):70–76.

Zliobaite, I. and Gabrys, B. (2014). Adaptive preprocessing for streaming data. *IEEE transactions on knowledge and data Engineering*, 26(2):309–321.