

## ABSTRACT

HOANG, PHUONG. Supervised Learning in Baseball Pitch Prediction and Hepatitis C Diagnosis. (Under the direction of Hien T. Tran.)

Machine learning is so ubiquitous nowadays that one probably uses it multiple times during the day without realizing it. For example, it is used in web search engines to improve efficiency, by email providers to identify junk emails, and in voice recognition, among others. Machine learning is a powerful tool that can be used to analyze large amount of data to make actionable predictions. Since machine learning uses algorithms that iterate on data, the quality and quantity of training data are important factors for accurate predictions. In particular, the data available for baseball pitch prediction is huge, millions of observations (pitches) each containing more than fifty features. However, the prediction task restricts researchers to working only with the less than ideal features that were measured before the target pitch is thrown. In addition, the presence of noise in pitch type labels makes it even harder to train classifiers. Meanwhile, the dataset for Hepatitis C is fairly small with less than two hundreds observations and 20 features. This disadvantage prevents researchers from removing observations with low quality when building reliable diagnosis models. Hence, prediction problems in the presence of missing features are pervasive in machine learning. This thesis focuses on a number of classification methods and other machine learning tools, and tailor them to address the above issues specifically.

First, in the pitch prediction problem, unlike the current method which suggests a static feature selection algorithm for each pitcher, we propose a novel dynamic feature selection procedure that is shown to be more adaptive for each pitcher in each count. The tradeoff is that the size of training data is reduced dramatically with pitcher-count data segmentation. Thus, we propose a simple heuristic approach for constructing and selecting features to include during training that are shown to surpass this tradeoff, which in turn yields considerable improvement in prediction accuracy.

In the second part of the thesis, we propose a new learning algorithm for Hepatitis C diagnosis that addresses the important issue of class imbalance. Most existing learning algorithms simply ignore the presence of class imbalance due to the lucrative high accuracy that can be easily attained. The current method suggests combining over-sampling (minority class) and weighted cost in Support Vector Machine. Through our research study, however, we were able to show that doing both is unnecessary. We choose only to employ the later

but add the parameter optimization procedure to improve classification performance. Our experimental results show that our proposed method is more accurate and reliable than the existing learning methods.

# Supervised Learning in Baseball Pitch Prediction and Hepatitis C Diagnosis

by  
Phuong Hoang

A dissertation submitted to the Graduate Faculty of  
North Carolina State University  
in partial fulfillment of the  
requirements for the Degree of  
Doctor of Philosophy

Applied Mathematics

Raleigh, North Carolina

2015

APPROVED BY:

---

Carl Meyer

---

Negash Medhin

---

Ernie Stitzinger

---

Hien T. Tran  
Chair of Advisory Committee

## **DEDICATION**

This thesis is dedicated to my parents, my wife, my son and Odisg.

## **BIOGRAPHY**

Phuong Hoang was born in Saigon, Vietnam on September 3, 1985. He went to the United States at the age 17 spending a year as an exchange student in Hartford high school, Michigan. In 2005, he started his undergraduate studies at University of Bridgeport, Connecticut in Business Administration before seeing the light and switching to mathematics in the second year. He later transferred to North Carolina State University and earned a B.S. in applied mathematics with financial mathematics concentration in May 2010. During his undergraduate studies, he interned at Sony Ericsson as a software tester on the GPS team. After graduation, he continued on at NC State for graduate school in applied mathematics. During his doctoral studies, he served as a REU graduate assistant for three summers. In his spare time, he manages of an e-commerce store that sells and trades textbooks and household electronics. He is also an avid photographer and has volunteered to be a cameraman on a few research workshops.

## ACKNOWLEDGEMENTS

First and foremost, I would like to thank my advisor Dr. Hien Tran for his guidance, encouragement, and support during the past 10 years. I deeply appreciate his great patience and the freedom he gave me to pursue new ideas, many of which were made into this thesis. His enthusiasm, persistence and expertise on research guided me throughout my doctoral study, from brainstorming ideas to doing experiments and writing technical papers, among many other things.

I would also like to thank my thesis committee members, Dr. Carl Meyer, Dr. Negash Medhin, and Dr. Ernie Stitzinger for their valuable feedbacks on this thesis. Especially, Dr. Meyer's fruitful lectures play a big role in keeping my love with applied mathematics.

Special thanks go to my undergraduate advisors Dr. Jeff Scroggs and Dr. Sandra Paur. I consider successful completion of Dr. Paur's analysis courses to be my best achievement in undergraduate study. I also owe to her for guiding me to graduate school and making sure it happens. I would also like to acknowledge Dr. H. T. Banks who provided me with a research fellowship during the third year of my Ph.D. program.

Graduate school would not have been the same with my colleagues Mark Hunnell, Rohit Sivaparasad, Hansi Jiang, Glenn Sidle and George Lankford. Besides our many fun social activities, they were always a useful sounding board when tackling theoretical questions. I wish to thank my former REU students, Michael Hamilton, Joseph Murray and Shar Shuai. Mentoring those who are smarter than I am has helped me in many ways, especially in finding better solution for a "solved" problem by looking at them from a different angle. Thanks also go to the research mentors from MIT Lincoln Laboratory Dr. Lori Layne and Dr. David Padgett for whom I was lucky enough to work with for two summers.

There are many friends who have helped me immeasurably throughout my student life. A few deserve special recognition. Nhat-Minh Phan who helped me in scrapping data that I used in this thesis and Nam Tran was always there for me, whether helping me financially during the rainy day or "shipping" his wife to Raleigh to help my wife and babysit my son when I were away for conferences. Particular thanks also go to Quoc Nguyen and Thanh Nguyen for putting up with me throughout the years. Also I would like to thank Dung Tran and Sombat Southivorarat for being excellent readers of this thesis.

My biggest thanks go to my loving and supportive parents, my wife and my son. This thesis is dedicated to them. Thank you for sharing this journey with me.

# TABLE OF CONTENTS

<b>LIST OF TABLES</b> . . . . .	<b>vii</b>
<b>LIST OF FIGURES</b> . . . . .	<b>viii</b>
<b>Chapter 1 Introduction</b> . . . . .	<b>1</b>
1.1 Statement of the Problems . . . . .	1
1.2 Dissertation Outline . . . . .	2
1.3 Summary of Contributions . . . . .	3
<b>Chapter 2 Classification</b> . . . . .	<b>4</b>
2.1 $k$ -Nearest Neighbors . . . . .	7
2.2 Linear Discriminant Analysis . . . . .	10
2.3 Support Vector Machines . . . . .	13
2.3.1 Linear Separable Case . . . . .	13
2.3.2 Nonseparable Case - Soft Margin SVM . . . . .	18
2.3.3 Nonlinearly Separable Case - Kernel trick . . . . .	22
<b>Chapter 3 Overfitting</b> . . . . .	<b>28</b>
3.1 Regularization . . . . .	30
3.2 Validation . . . . .	33
3.2.1 Hold-Out Validation . . . . .	34
3.2.2 Cross Validation . . . . .	35
<b>Chapter 4 Baseball Pitch Prediction</b> . . . . .	<b>37</b>
4.1 PITCHf/x Data . . . . .	38
4.2 Related Work . . . . .	38
4.3 Our Model . . . . .	41
4.3.1 Dynamic Feature Selection Approach . . . . .	41
4.3.2 Model Implementation . . . . .	43
4.3.3 ROC Curves . . . . .	44
4.3.4 Hypothesis Testing . . . . .	46
4.3.5 Classification . . . . .	47
4.4 Results Analysis . . . . .	47
4.4.1 Overall Results . . . . .	47
4.4.2 By Count Analysis . . . . .	49
4.4.3 By Pitcher Analysis . . . . .	49
4.4.4 By Noise Level . . . . .	52
<b>Chapter 5 Medical Diagnosis</b> . . . . .	<b>54</b>
5.1 Previous Work . . . . .	55

5.2	Class Imbalance and Related Work . . . . .	57
5.3	Model Implementation . . . . .	60
5.3.1	Data Preprocessing . . . . .	60
5.3.2	Cost Sensitive SVM . . . . .	63
5.3.3	Parameters Optimization . . . . .	64
5.3.4	Evaluation Metrics . . . . .	66
5.4	Results Analysis . . . . .	66
<b>Chapter 6</b>	<b>Conclusion and Future Work . . . . .</b>	<b>72</b>
6.1	Pitch Prediction . . . . .	72
6.2	Hepatitis Diagnosis . . . . .	74
	<b>BIBLIOGRAPHY . . . . .</b>	<b>76</b>
	<b>APPENDICES . . . . .</b>	<b>82</b>
Appendix A	Baseball Pitch Prediction . . . . .	83
A.1	Features in Groups . . . . .	83
A.2	Baseball Glossary and Info . . . . .	88
A.3	Software . . . . .	89
Appendix B	Hepatitis C Diagnosis . . . . .	90
B.1	Definitions of Attributes . . . . .	90



## LIST OF TABLES

Table 2.1	Frequently Used Notation, adapted from [1, 12, 37, 57] . . . . .	6
Table 2.2	Accuracy and speed comparison of $k$ -NN method using different metrics	8
Table 4.1	List of original attributes selected for pitch prediction . . . . .	39
Table 4.2	Data for each pitcher. . . . .	50
Table 4.3	Prediction accuracy comparison (percents). Symbols: $k$ -Nearest Neighbors ( $k$ -NN), Support Vector Machine with linear kernel (SVM-L), Support Vector Machine with Gaussian kernel (SVM-G), Naive Guess (NG).	50
Table 4.4	CPU Times (seconds) . . . . .	51
Table 4.5	Best Improvement over Naive Guess (percents) . . . . .	51
Table 4.6	Prediction results by Type Confidence levels (TC). Symbol: $k$ -Nearest Neighbors ( $k$ -NN), Support Vector Machine with linear kernel (SVM-L), and with Gaussian kernel (SVM-G), Linear Discriminant Analysis (LDA). . . . .	52
Table 5.1	Hepatitis C classification accuracies comparison among recent studies.	56
Table 5.2	Breast cancer classification accuracies comparison among recent studies. . . . .	57
Table 5.3	Hepatitis C data: description of attributes. See Appendix B.1 for the definitions of some of the attributes. . . . .	62
Table 5.4	The table shows the sensitivity and specificity comparison between algorithms: Support Vector Machines (SVM), Under-sampling (US), Different Error Costs (DEC), SMOTE with Different Costs (SDC). . . . .	69
Table 5.5	The table shows the G-Mean comparison between algorithms: Support Vector Machines (SVM), Under-sampling (US), Different Error Costs (DEC), SMOTE with Different Costs (SDC). . . . .	69

## LIST OF FIGURES

Figure 2.1	Basic setup of the learning problem [1] . . . . .	5
Figure 2.2	An example of $k$ -NN; using the 7-NN rule, the unknown data point in red is classified to the black class. Out of the seven nearest neighbors, five are of black class and two are of white class (the dashed circle denotes the region that contains the 7 nearest neighbors of the unknown data point). . . . .	7
Figure 2.3	An example of binary classification problem. Data from class 1 (blue) favor the $y$ -axis while data from class 2 (red) spread out the along the North East direction. The unknown data (black) is to be classified with $k$ -NN. . . . .	9
Figure 2.4	An example of LDA. Two one-dimensional density functions are shown. The dashed vertical line represents the Bayes decision boundary. The solid vertical line represents the LDA decision boundary estimated from training data. The source code used to make this figure is adapted from [58], used under CC0 1.0, via Wikimedia Commons. . . . .	12
Figure 2.5	An example of a linearly separable two-class problem with SVM. The source code used to make this figure is adapted from [51]. . . . .	14
Figure 2.6	An example of a linearly nonseparable two-class problem with SVM. The incorrectly classified data points are enclosed in blue circle. The source code used to make this figure is adapted from [51]. . . . .	19
Figure 2.7	Kernels are used for mapping a non-linearly separable problem into a higher dimension linearly separable problem. The source code used to make this figure is adapted from [50]. . . . .	22
Figure 2.8	An example of kernel trick. . . . .	23
Figure 3.1	Example showing overfitting of a classifier by Chabacano, used under CC BY-SA, via Wikimedia Commons [17]. The green curve separates the blue and the red dots perfectly in this (training) data set, hence it has lower $E_{\text{in}}$ than that of the black curve. However, it also models the noise at the boundary in addition to model the underlying trend. Hence, it will more likely perform poorly on a new data set from the same population, has higher $E_{\text{out}}$ . The green curve is an example of an overfitted classifier. . . . .	29
Figure 3.2	Hinge loss function. Note that the loss is asymmetric: incorrect classification, $y_i f(x_i) < 0$ , is linearly increasing loss as $y_i f(x_i)$ decreases and correct classification with $y_i f(x_i) \geq 1$ is always zero loss. . . . .	32
Figure 4.1	Distribution of Group 1 features (1-13) via feature selection method of Miguel Batista (2008-2009). . . . .	43

Figure 4.2	Schematic diagram of the proposed adaptive feature selection. . . . .	44
Figure 4.3	ROC curve. In this example, three features are measured with the ROC curve. The blue curve represents the best feature among the three since it has the highest AUC. The diagonal line represents random guessing. The area between a ROC curve and the diagonal line quantifies how much better that feature is at distinguishing the two classes compared to random guessing. . . . .	45
Figure 4.4	Hypothesis testing for feature selection. The black bars are the number of features returned by the ROC curve test, the blue are the features considered optimal for $\alpha = .01$ , the red are for $\alpha = 0.05$ . . . . .	46
Figure 4.5	Prediction accuracy comparison (percents). Symbol: $k$ -Nearest Neighbors ( $k$ -NN), Support Vector Machine with linear kernel (SVM-L), and with Gaussian kernel (SVM-G), Linear Discriminant Analysis (LDA), Prediction Tree (PT). . . . .	48
Figure 4.6	Prediction accuracy by count. . . . .	49
Figure 5.1	Hepatitis C classification comparison with 10-fold CV over $N = 500$ repetitions. . . . .	58
Figure 5.2	Schematic diagram of the proposed diagnosis classification . . . . .	61
Figure 5.3	Parameter Optimization via Grid Search on Hepatitis C (10-CV). . . . .	65
Figure 5.4	G-Mean comparison on Hepatitis C dataset with 10-fold CV over $N = 500$ repetitions. . . . .	67
Figure 5.5	ROC-AUC comparison on Hepatitis C dataset with 10-fold CV over $N = 500$ repetitions. . . . .	68
Figure 5.6	Feature Selection comparison with 10-fold CV over $N = 500$ repetitions. Number of missing values associated with features 19, 16, 20 are 67, 29, and 20 respectively (see Table 5.3). . . . .	70
Figure 5.7	The scatter plots of the reduced feature subsets by LFDA. . . . .	71
Figure 6.1	Relationship among three main components of this learning model. . . . .	74

# CHAPTER

# 1

# INTRODUCTION

## 1.1 Statement of the Problems

In this thesis, we address some challenges in supervised learning when dealing with both large and small datasets in the context of baseball pitch prediction and medical diagnosis domains. The purpose of this dissertation is to address some major issues that come up with each problem. For pitch prediction, the challenge is finding better alternatives for pitch prediction when only pre-pitch information is available in training. Furthermore, pitchers tend to develop similar pitching patterns at the same pitch count. So instead of having one learning model per pitcher, having a separate model for each pitcher-count pair may adapt to the changes in game situation better. This further data segmentation indeed divides the number of observations available for training classifier into smaller sets. This brings us back to the previous challenge, how to find features with higher predictive strength to compensate for the potential lack of training data. We propose in this thesis an

approach that will overcome this shortcoming. More specifically, from the raw data, we generate synthetic features that replicate human (pitchers) decision making, then group them in their similarity and perform two filtering steps to include only a set of optimal features that are then used for training classifiers. In validation, we applied our final model to other datasets that were unseen by the machine to ensure its effectiveness in predicting future data.

For Hepatitis C diagnosis, researchers have been developing methods for disease classification where nearly perfect (above 95%) accuracy were reached. Many of these studies however omit the presence of class imbalance where accuracy alone is a poor evaluation metric. Some studies use validation method, but report only the best result, another misleading measurement of classification performance with particularly small number of training observations. We propose a method that (1) provides a simpler solution that outperforms the current methods and (2) employs parameters optimization and cross validation within metrics designated for treating class imbalance.

The essence of our methods is to attain the best fit of the (in-sample) data at hand and perform reliably with the out-of-sample data once they encountered (i.e., minimize overfitting possibilities). To achieve this goal, we carefully examine theoretical foundation of overfitting, modify our classifier appropriately and provide as much theoretical justification to the observed heuristic as possible.

## **1.2 Dissertation Outline**

The thesis begins, in Chapter 2, with an introduction of the supervised learning and three classification techniques used later in this study. Chapter 3 follows with the discussion of the overfitting problem in supervised learning, as well as two useful tools to address the overfitting problem. In Chapter 4 we propose a pitch prediction model, built on PITCHf/x data of MLB seasons 2008-2012. We also compare the performance of various classification methods mentioned and employ cross validation techniques that previously introduced in Chapter 2 and 3. Chapter 5 focuses on the medical diagnosis applications in which we reconfigure the Support Vector Machine classification with weighted cost and employ parameters optimization and conduct a wide range of validation techniques to address

the overfitting and to enhance the performance overall. Chapter 6 draws conclusions and suggests areas of potential interest for future work.

## 1.3 Summary of Contributions

This section summarizes the contributions of this study and refers to the specific sections of the thesis where they can be found:

1. We design and implement a novel feature selection approach for baseball pitch prediction (section 4.3). Our model is shown to adapt well to the different pitchers (section 4.4.3) and different count situation (section 4.4.2) even at the time they change their pitch type. Overall, our experimental results show that our model (1) can achieve higher accuracy (up to 10 %) than the existing method under hold-out validation (introduced in section 3.2.1) and (2) shown to be stable with low variances in cross validation results (less than 2% difference).
2. We propose a new disease diagnosis system that is able to combat overfitting caused by unbalanced datasets. Our model adapts the Support Vector Machine (SVM) method using modified cost function that is sensitive to class imbalance (section 5.3.2). We implement cross validation within the grid search algorithm to determine the best choices of parameters  $C$  and  $\gamma$  of the SVM with Gaussian kernel (section 5.3.3). For each classifier, we make 500 simulation runs to measure and compare consistency in term of variances.

## CHAPTER

# 2

## CLASSIFICATION

Classification is the process of taking an unlabeled data observation and using some rule or decision-making process to assign a label to it. Given  $\mathcal{X}$  is the input space and  $\mathcal{Y}$  is the output space (often called labels),  $\mathcal{D}$  denotes the data set of input-output examples  $(x_1, y_1), \dots, (x_N, y_N)$  where  $y_i = f(x_i)$  for  $i = 1, \dots, N$ . The process of classification is to find an algorithm or strategy that uses the data set  $\mathcal{D}$  to find a function  $g$  from the hypothesis set  $H$  that best approximates the ideal function  $f : \mathcal{X} \rightarrow \mathcal{Y}$  [1].

To support this concept, in [1], the authors present the credit card approval example. The goal here is for the bank to use historical records of previous customers to figure out a good formula for credit approval. In this case, as illustrated in Figure 2.1,  $x_i$  is the customer information that is used to make a credit decision,  $y_i$  is a Yes/No decision,  $f$  is the ideal formula for credit approval and data set  $\mathcal{D}$  contains all input-output examples corresponding to previous customers and the credit decision for them in hindsight. Once we find  $g$  ( $f$  remains unknown) that best matches  $f$  on the training data, we apply  $g$  to classify new credit card customer with the hope that it would still match  $f$  on (future)

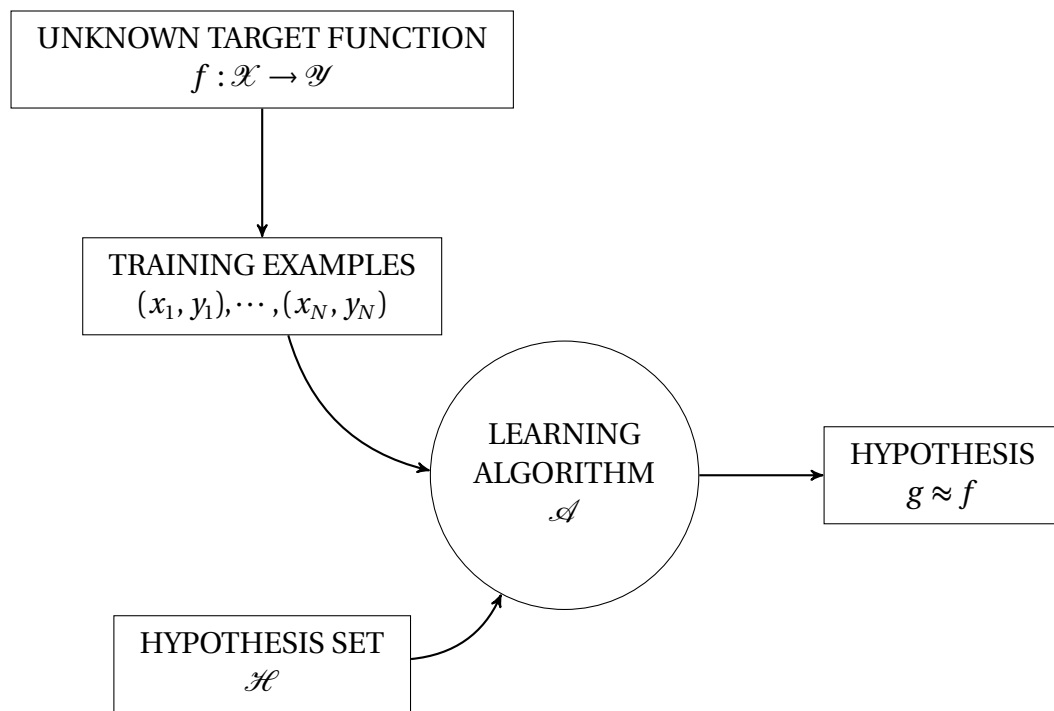


Figure 2.1: Basic setup of the learning problem [1]



unlabeled data. The notation used in this chapter and the next one mainly follow that of [1], some frequently used notation are listed in Table 2.1.

Table 2.1: Frequently Used Notation, adapted from [1, 12, 37, 57]

$\lambda$	regularization parameter
$\Omega$	penalty for model complexity; either a bound on generalization error, or a regularization term
$\Phi$	feature transform, $z = \Phi(x)$
$\phi$	a coordinate in feature transform $\Phi$ , $z = \phi_i(x)$
$C$	bound on the size of weights in a soft order constraint
$d$	dimensionality of the input space $\mathcal{X} = \mathbb{R}^d$ or $\mathcal{X} = \{1\} \times \mathbb{R}^d$
$\mathcal{D}$	data set $\mathcal{D} = \{(x_1, y_1), \dots, (x_N, y_N)\}$ often the training set, but sometimes split into training and validation/test sets
$\mathcal{D}_{\text{train}}$	subset of $\mathcal{D}$ used for training when a validation or test set is used
$E_{\text{in}}, E_{\text{in}}(h)$	in-sample error (training error) for hypothesis $h$
$E_{\text{cv}}$	cross validation error
$E_{\text{out}}, E_{\text{out}}(h)$	out-of-sample error for hypothesis $h$
$E_{\text{val}}$	validation error
$E_{\text{test}}$	test error
$f$	target function, $f : \mathcal{X} \rightarrow \mathcal{Y}$
$g$	final hypothesis $g \in \mathcal{H}$ selected by the learning algorithm; $g : \mathcal{X} \rightarrow \mathcal{Y}$
$g^{(\mathcal{D})}$	final hypothesis when the training set is $\mathcal{D}$
$h$	a hypothesis $h \in H$ ; $h : \mathcal{X} \rightarrow \mathcal{Y}$
$\mathcal{H}$	hypothesis set
$\mathcal{X}$	input space whose elements are $x \in \mathcal{X}$
$\mathcal{Y}$	output space whose elements are $y \in \mathcal{Y}$

The classification methods used in this study are the  $k$ -nearest neighbors ( $k$ -NN), the Linear Discriminant Analysis (LDA) and the Support Vector Machine (SVM). A brief description of  $k$ -NN and LDA is provided in the next two sections. Meanwhile SVM is chosen to be the main classifier of our study and we will discuss it in more details in the third section.

## 2.1 $k$ -Nearest Neighbors

The  $k$ -nearest neighbors algorithm ( $k$ -NN) classifies an unlabeled point based on the closest  $k$  training points in the multidimensional feature space; each of the  $k$  neighbors has a class label and the label of a given point is determined by the majority vote of the class labels of its  $k$ -nearest neighbors, see [57]. An example of  $k$ -NN is presented below in Figure 2.2.

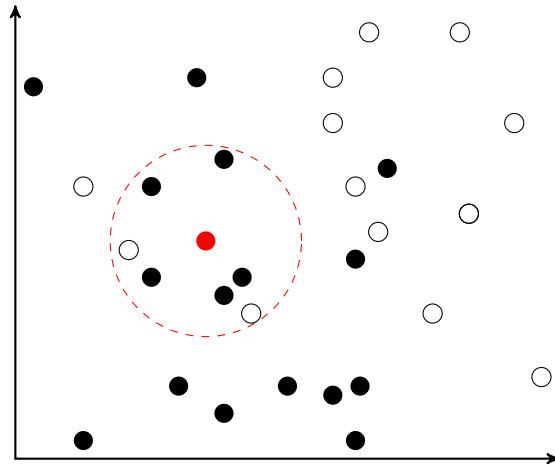


Figure 2.2: An example of  $k$ -NN; using the 7-NN rule, the unknown data point in red is classified to the black class. Out of the seven nearest neighbors, five are of black class and two are of white class (the dashed circle denotes the region that contains the 7 nearest neighbors of the unknown data point).

1. A value for  $k$  is defined by the user.
2. The distance between the unclassified point and each class-labeled point is calculated.
3. The  $k$  nearest neighbors are chosen based on that distance.
4. The class label of the unclassified point is defined as the majority of the class labels of the  $k$  nearest neighbors.

This method is customizable—the user can use different values of  $k$  and optionally pick different distance metrics. The standard choice of metric is the Euclidean distance (2.1),

$$d_{\text{Eucl}}(x, y) = \sum_{i=1}^l (x_i - y_i)^2. \quad (2.1)$$

Mahalanobis distance (2.2) takes into account the correlations of the data,

$$d_{\text{Mahal}}^i(x, y) = \sqrt{(x - y)^T \Sigma^{-1} (x - y)}, \quad (2.2)$$

where  $\Sigma$  is the covariance [9, 57]. It is clear that if the covariance matrix  $\Sigma$  is the identity matrix, then Mahalanobis distance is the same as the Euclidean distance. Another common metric often used with  $k$ -NN is the Manhattan distance (2.3), also known as the city-block distance or  $L_1$  norm,

$$d_{\text{Man}}(x, y) = \sum_{i=1}^l |x_i - y_i|. \quad (2.3)$$

An example of a binary class problem using  $k$ -NN classifier is presented below. In this example, data from both classes are generated from the Gaussian distribution with different means and covariances, as illustrated in Figure 2.3. We use the  $k$ -NN algorithm with three metrics above to classify the unknown data set (black). The unknown set distributes similar to class 2 but its center is closer to that of class 1. As shown in Table 2.2, Mahalanobis outperforms the other two types of distance in this case because it pays more attention to the covariances of each class. However the drawback is the time complexity due to matrix multiplication.

Table 2.2: Accuracy and speed comparison of  $k$ -NN method using different metrics

Methods	Accuracy	CPU time
k-NN	97.80	0.039014s
k-NN Man	97.90	0.039674s
k-NN Mahal	98.40	0.174085s

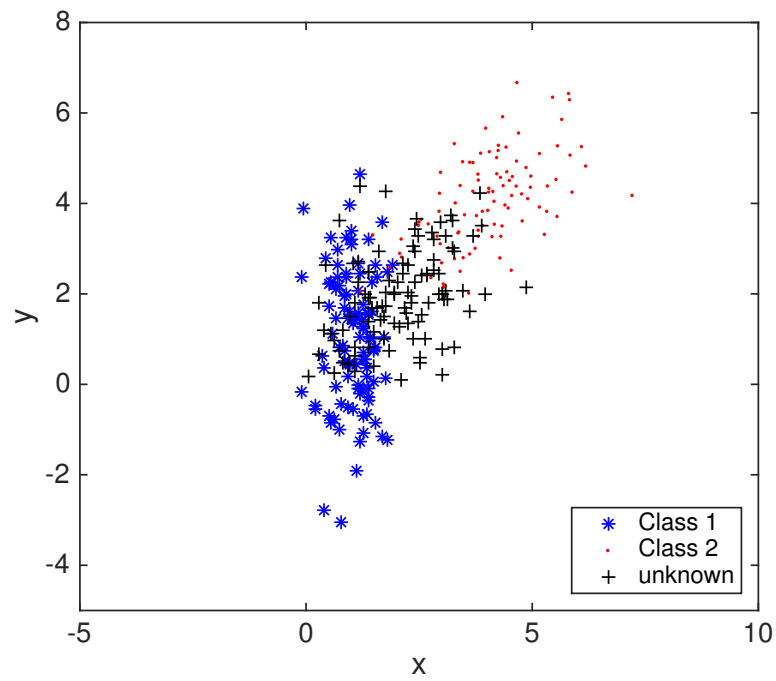


Figure 2.3: An example of binary classification problem. Data from class 1 (blue) favor the  $y$ -axis while data from class 2 (red) spread out the along the North East direction. The unknown data (black) is to be classified with  $k$ -NN.

## 2.2 Linear Discriminant Analysis

The Linear Discriminant Analysis (LDA) classifier assumes that the observations within each class  $k$  are generated from a Gaussian (or normal) distribution with a class-specific mean vector  $\mu_k$ 's and a common variance  $\sigma_k^2$ 's. Estimates for these parameters are substituted into the Bayes classifier results.

Assume that we have only one feature (or predictor) in a  $K$ -class classification problem. Then the Bayes' theorem states that

$$\Pr(Y = k|X = x) = \frac{\pi_k f_k(x)}{\sum_{l=1}^K \pi_l f_l(x)}, \quad (2.4)$$

where  $\pi_k$  represent the prior probability that a randomly chosen observation is associated with the  $k$ -th class,  $f_k(X) \equiv \Pr(X = x|Y = k)$  denotes the density function of  $X$  for an observation that comes from  $k$ -th class, and  $\Pr(Y = k|X)$ , abbreviated as  $p_k(X)$ , refers to the posterior probability that an observation  $X = x$  belongs to the  $k$ -th class. If we can compute all the terms for (2.4), we would then easily classify an observation to the class for which  $p_k(X)$  is largest.

Since  $f_k(x)$  is Gaussian, the normal density takes the form

$$f_k(x) = \frac{1}{\sqrt{2\pi}\sigma_k} \exp\left(-\frac{1}{2\sigma_k^2}(x - \mu_k)^2\right). \quad (2.5)$$

Substituting (2.5) into (2.4) under assumption that  $\sigma_k^2 \equiv \sigma^2$ , we obtain

$$p_k(x) = \frac{\pi_k \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(x - \mu_k)^2\right)}{\sum_{l=1}^K \pi_l \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(x - \mu_l)^2\right)} \quad (2.6)$$

$$= \frac{\pi_k \exp\left(-\frac{1}{2\sigma^2}(x - \mu_k)^2\right)}{\sum_{l=1}^K \pi_l \exp\left(-\frac{1}{2\sigma^2}(x - \mu_l)^2\right)}. \quad (2.7)$$

To classify an observation  $X = x$ , we need to see which of the  $p_k(x)$  is largest. Taking the

log of (2.7), we obtain

$$\log(p_k(x)) = \log\left(\pi_k \exp\left(-\frac{1}{2\sigma^2}(x - \mu_k)^2\right)\right) - \underbrace{\log\left(\sum_{l=1}^K \pi_l \exp\left(-\frac{1}{2\sigma^2}(x - \mu_l)^2\right)\right)}_{\text{this term does not depend on } k}. \quad (2.8)$$

Notice that  $\log(p_k(X))$  depends only on the first term of the right hand side. Let's call this term  $\eta_k(x)$ , we have

$$\eta_k(x) = \log\left(\pi_k \exp\left(-\frac{1}{2\sigma^2}(x - \mu_k)^2\right)\right) \quad (2.9)$$

$$= \log(\pi_k) - \frac{1}{2\sigma^2}(x - \mu_k)^2 \quad (2.10)$$

$$= \log(\pi_k) - \underbrace{\frac{x^2}{2\sigma^2}}_{\text{this term does not depend on } k} + x \cdot \frac{\mu_k}{\sigma^2} - \frac{\mu_k^2}{2\sigma^2}. \quad (2.11)$$

Therefore, classifying an observation  $X = x$  is equivalent to assigning it to the class which

$$\delta_k(x) = x \cdot \frac{\mu_k}{\sigma^2} - \frac{\mu_k^2}{2\sigma^2} + \log(\pi_k) \quad (2.12)$$

is largest, where  $\delta_k(x)$  denotes the class discriminant function.

An example is shown in Figure 2.4, adapted from [33]. The two normal density functions are shown,  $f_1(x)$  and  $f_2(x)$ , represent two distinct classes. The mean and variance parameters for the two density functions are  $\mu_1 = 40$ ,  $\mu_2 = 80$ , and  $\sigma_1^2 = \sigma_2^2 = 100$ . Because the two densities overlap, there is some uncertainty about the class to which a given observation  $X = x$  belongs. Under the assumption that an observation is equally likely come from either class,  $\pi_1 = \pi_2 = 0.5$ , then we can compute the Bayes classifier since  $X$  is drawn from a Gaussian distribution within each class, and all parameters involved are known. In practice, even if we are certain on the assumption that  $X$  is drawn from a Gaussian distribution within each class, we still have to estimate the parameters  $\mu_k$ ,  $\pi_k$ , and  $\sigma^2$ . This is where the LDA method comes into play, it specifically provides the estimates for  $\mu_k$ ,  $\pi_k$ , and  $\sigma^2$ , and hence, an approximation of Bayes classifier. Since these estimates depend largely on the training data that may or may not be the a good representation of each class, the LDA

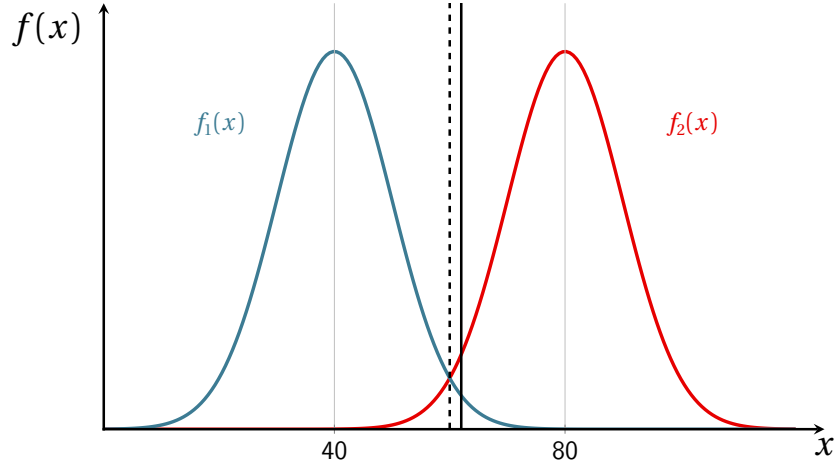


Figure 2.4: An example of LDA. Two one-dimensional density functions are shown. The dashed vertical line represents the Bayes decision boundary. The solid vertical line represents the LDA decision boundary estimated from training data. The source code used to make this figure is adapted from [58], used under CC0 1.0, via Wikimedia Commons.

decision boundary can be different from the Bayes decision boundary, as illustrated in Figure 2.4.

The LDA method approximates the Bayes classifier by substituting the following estimates for  $\pi_k$ ,  $\mu_k$  and  $\sigma^2$  into (2.12),

$$\begin{aligned}\hat{\pi}_k &= \frac{n_k}{n}, \\ \hat{\mu}_k &= \frac{1}{n_k} \sum_{i: y_i=k} x_i, \\ \hat{\sigma}^2 &= \frac{1}{n-K} \sum_{k=1}^K \sum_{i: y_i=k} (x_i - \hat{\mu}_k)^2,\end{aligned}$$

where  $n$  is the total number of training observations and  $n_k$  is the number of training observations in  $k$ -th class. After the LDA procedure, (2.12) becomes

$$\hat{\delta}_k(x) = x \cdot \frac{\hat{\mu}_k}{\hat{\sigma}^2} - \frac{\hat{\mu}_k^2}{2\hat{\sigma}^2} + \log(\hat{\pi}_k). \quad (2.13)$$

The LDA classifier can be extended to multiple predictors. To do this, we assume that  $X = (X_1, X_2, \dots, X_p)$  is drawn from a multivariate Gaussian distribution with a class-specific mean vector and common covariance matrix, the corresponding equations (2.5) and (2.12) are

$$f(x) = \frac{1}{(2\pi)^{p/2} |\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x - \mu)^T \Sigma^{-1} (x - \mu)\right), \quad (2.14)$$

and

$$\delta_k(x) = x^T \Sigma^{-1} \mu_k - \frac{1}{2} \mu_k^T \Sigma^{-1} \mu_k + \log(\pi_k). \quad (2.15)$$

The formulas for estimating the unknown parameters  $\pi_k$ ,  $\mu_k$ , and  $\Sigma$  are similar to the one-dimensional case. To assign an observation  $X = x$ , the LDA uses these estimates in (2.15) and assigns the class label for which discrimination function  $\hat{\delta}_k(x)$  is largest. The word *linear* in the classifier's name comes from the fact that these discrimination functions are linear functions of  $x$ . Unlike the  $k$ -NN, where no assumptions are made about the shape of the decision boundary, the LDA produces linear decision boundaries. See [33] for more details.

## 2.3 Support Vector Machines

### 2.3.1 Linear Separable Case

Support Vector Machine is a linear classification tool that simultaneously optimizes prediction accuracy and avoids overfitting (Chapter 3). The algorithm is dependent on the notion of margin.

For a two-class classification problem, the separating hyperplane

$$u(x) = w \cdot x + b \quad (2.16)$$

is not unique as it may be biased towards one class<sup>1</sup>. The goal is to maximize the distance between two classes, and effectively drawing the decision boundary that maximizes this margin. Intuitively, this separation is achieved by the hyperplane that has the largest distance to the nearest training data point (the so-called *support vectors*) of either class.

---

<sup>1</sup>the dot symbol in (2.16) denotes the inner product



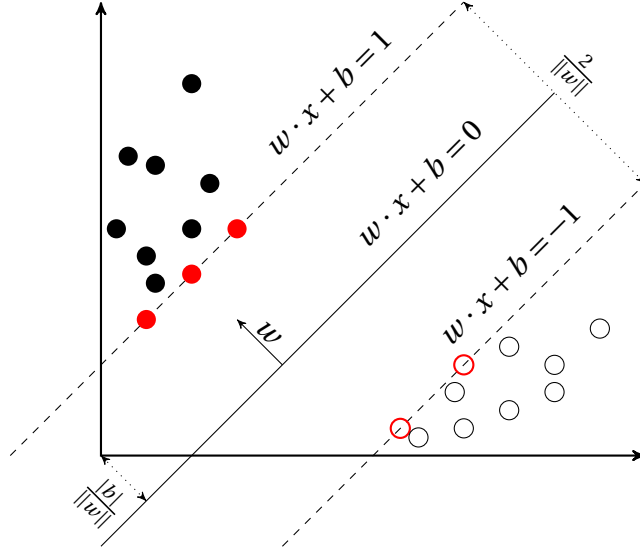


Figure 2.5: An example of a linearly separable two-class problem with SVM. The source code used to make this figure is adapted from [51].

An example of linearly separable SVM for binary classification is presented in Figure 2.5.

Let  $\{x_i, y_i\}$ ,  $i = 1, 2, \dots, l$ ,  $x_i \in \mathbb{R}^d$  be the input-output samples of the training set,  $D$ . Suppose we have some separating hyperplane  $H$ , a point  $x$  that lies on  $H$  needs to satisfy  $w \cdot x + b = 0$ , where  $w$  is the weight vector that is normal to  $H$ . Then all observations from the training data need to satisfy the constraints

$$x_i \cdot w + b \geq +1 \quad \text{if } y_i = +1, \quad (2.17)$$

$$x_i \cdot w + b \leq -1 \quad \text{if } y_i = -1, \quad (2.18)$$

which can be simplified to

$$y_i(x_i \cdot w + b) \geq 1, \quad i = 1, 2. \quad (2.19)$$

We define the following hyperplanes,

$$\begin{aligned} H_1 : x_i \cdot w + b &= +1, \\ H_2 : x_i \cdot w + b &= -1, \end{aligned} \tag{2.20}$$

such that the points on the planes  $H_1$  and  $H_2$  are the support vectors. Note that  $H_1$  and  $H_2$  have same normal vector hence they are parallel to each other. In addition, no training points should lie between them. Now, from (2.20) the distances from the origin to  $H_1$  and  $H_2$  are  $\frac{|1-b|}{\|w\|}$  and  $\frac{|-1-b|}{\|w\|}$ , respectively. Hence, the margin between  $H_1$  and  $H_2$  is  $\frac{2}{\|w\|}$ . In order to maximize the margin between  $H_1$  and  $H_2$ ,  $\|w\|$  must be minimized. Combining this with the constraint (2.19), we have an optimization problem

$$\min \quad \frac{1}{2} \|w\|^2, \tag{2.21}$$

$$\text{subject to} \quad y_i(x_i \cdot w + b) \geq 1, \quad i = 1, 2, \dots, l. \tag{2.22}$$

Since the norm  $\|w\|$  involves a square root, which makes optimization difficult, we have replaced  $\|w\|$  with  $\frac{\|w\|^2}{2}$ . We now have a quadratic program that can be solved using Lagrange multipliers, as suggested in [14, 57, 60]. The Lagrangian for this problem is

$$L = \frac{1}{2} \|w\|^2 - \sum_{i=1}^l \alpha_i [y_i(x_i \cdot w + b) - 1], \tag{2.23}$$

where  $\alpha_i$  denotes the Lagrangian multipliers. To minimize  $L$ , the following Karush-Kuhn-Tucker (KKT) conditions [57] must be satisfied

$$L_w = 0, \tag{2.24}$$

$$L_b = 0, \tag{2.25}$$

$$\alpha_i \geq 0, \quad i = 1, 2, \dots, l, \tag{2.26}$$

$$\alpha_i [y_i(x_i \cdot w + b) - 1] = 0, \quad i = 1, 2, \dots, l. \tag{2.27}$$

Combining these with equation (2.23) we have

$$w = \sum_{i=1}^l \alpha_i y_i x_i, \quad (2.28)$$

$$\sum_{i=1}^l \alpha_i y_i = 0. \quad (2.29)$$

We now consider the Lagrangian duality problem which is called the Wolfe dual representation form. Following [14, 57], we have,

$$\max \quad \frac{1}{2} \|w\|^2 - \sum_{i=1}^l \alpha_i [y_i (x_i \cdot w + b) - 1], \quad (2.30)$$

$$\text{subject to} \quad w = \sum_{i=1}^l \alpha_i y_i x_i, \quad (2.31)$$

$$\sum_{i=1}^l \alpha_i y_i = 0, \quad (2.32)$$

$$\alpha_i \geq 0. \quad (2.33)$$

Substituting (2.31) in (2.30) yields

$$\frac{1}{2} \left( \sum_{i=1}^l \alpha_i y_i x_i \right) \cdot \left( \sum_{j=1}^l \alpha_j y_j x_j \right) - \left( \left( \sum_{i=1}^l \alpha_i y_i x_i \right) \cdot \left( \sum_{j=1}^l \alpha_j y_j x_j \right) + b \sum_{i=1}^l \alpha_i y_i - \sum_{i=1}^l \alpha_i \right). \quad (2.34)$$

Applying (2.32) and rearranging the terms, we have

$$\sum_{i=1}^l \alpha_i - \frac{1}{2} \left( \sum_{i=1}^l \alpha_i y_i x_i \right) \cdot \left( \sum_{j=1}^l \alpha_j y_j x_j \right). \quad (2.35)$$

The Lagrangian problem becomes

$$\max \sum_{i=1}^l \alpha_i - \frac{1}{2} \sum_{i,j=1}^l \alpha_i \alpha_j y_i y_j x_i \cdot x_j, \quad (2.36)$$

$$\text{subject to } \sum_{i=1}^l \alpha_i y_i = 0, \quad (2.37)$$

$$\alpha_i \geq 0. \quad (2.38)$$

If  $\alpha_i > 0$ , the corresponding data point is the support vector and the solution for the optimal separating hyperplane is

$$w = \sum_{i=1}^n \alpha_i y_i x_i, \quad (2.39)$$

where  $n \leq l$  is the number of support vectors. Once we determine  $w$  and  $b$ , from equation (2.27), the optimal linear discriminant function is

$$g(x) = \text{sgn}(w \cdot x + b) \quad (2.40)$$

$$= \text{sgn} \left( \sum_{i=1}^l \alpha_i y_i x_i \cdot x + b \right). \quad (2.41)$$

Finally, from equation (2.27), for any nonzero  $\alpha_m$  which is associated with some support

vector  $x_m$  and label  $y_m$ , we compute  $b$  as followed

$$\alpha_m[y_m(x_m \cdot w + b) - 1] = 0, \quad (2.42)$$

$$y_m(x_m \cdot w + b) - 1 = 0, \quad (2.43)$$

$$y_m x_m \cdot w + y_m b - 1 = 0, \quad (2.44)$$

$$b = \frac{1}{y_m}(1 - y_m x_m \cdot w), \quad (2.45)$$

$$= \frac{1}{y_m} - x_m \cdot w, \quad (2.46)$$

$$= \frac{1}{y_m} - \sum_{i=1}^l \alpha_i y_i x_i \cdot x_m, \quad (2.47)$$

$$= y_m - \sum_{i=1}^l \alpha_i y_i x_i \cdot x_m. \quad (2.48)$$

Notice that equation (2.48) holds because  $y_m = \pm 1$ .

### 2.3.2 Nonseparable Case - Soft Margin SVM

The above setup only works for separable data. For a nonseparable two-class problem, we cannot draw a separating hyperplane with associated hyperplanes  $H_1$  and  $H_2$  such that there is no data point lying between them. To address this issue, recall that  $H_1$  and  $H_2$  have the form

$$x_i \cdot w + b = \pm 1 \quad (2.49)$$

and the margin is the distance between them. Any training data point,  $x_i$  (with associated class label  $y_i$ ) in the training set must belong to one of the following three cases (see Figure 2.6),

- $x_i$  lies outside the margin and correctly classified, so  $x$  satisfies the inequality constraints in (2.22), i.e.,  $y_i(x_i \cdot w + b) \geq 1$ ,
- $x_i$  lies between the margin and correctly classified, so  $0 \leq y_i(x_i \cdot w + b) < 1$ ,
- $x_i$  lies between the margin and incorrectly classified, so  $y_i(x_i \cdot w + b) < 0$ .

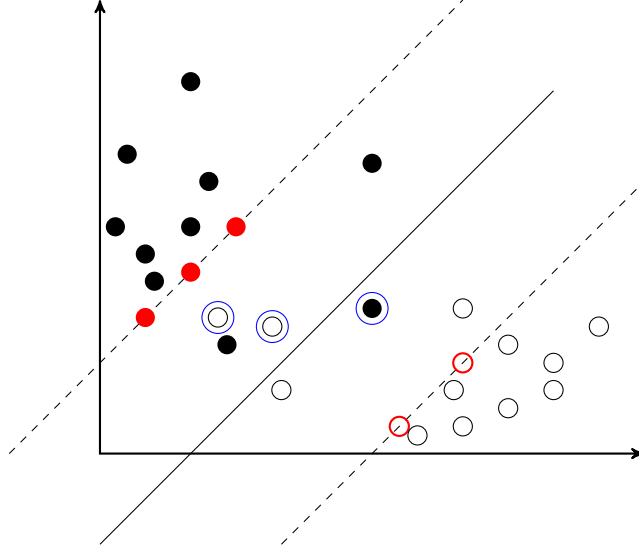


Figure 2.6: An example of a linearly nonseparable two-class problem with SVM. The incorrectly classified data points are enclosed in blue circle. The source code used to make this figure is adapted from [51].

By introducing a slack variable  $\xi_i$ , we account all the above three cases in a single constraint

$$y_i(x \cdot w + b) \geq 1 - \xi_i. \quad (2.50)$$

That is, the first, second, and third case correspond to  $\xi_i = 0$ ,  $0 < \xi_i \leq 1$ , and  $\xi_i > 1$  respectively. The constraints (2.17) and (2.51) become

$$x_i \cdot w + b \geq +1 - \xi_i \quad \text{if } y_i = +1, \quad (2.51)$$

$$x_i \cdot w + b \leq -1 + \xi_i \quad \text{if } y_i = -1. \quad (2.52)$$

The new optimization problem is

$$\begin{aligned}
& \min \quad \frac{1}{2} \|w\|^2 + C \sum_{i=1}^l \xi_i, \\
& \text{subject to} \quad y_i(x_i w + b) \geq 1 - \xi_i, \quad i = 1, 2, \dots, l, \\
& \quad \quad \quad \xi_i \geq 0, \quad i = 1, 2, \dots, l,
\end{aligned} \tag{2.53}$$

where  $C$  is a parameter that controls the trade-off between the two main goals: maximizing margin and having fewer number of misclassification. This is still a convex optimization problem, hence we proceed with the Lagrange method as before [57]. The Lagrangian for this new problem is

$$L = \frac{1}{2} \|w\|^2 + C \sum_{i=1}^l \xi_i - \sum_{i=1}^l \mu_i \xi_i - \sum_{i=1}^l \alpha_i [y_i(x_i \cdot w + b) - 1 + \xi_i], \tag{2.54}$$

with the corresponding KKT conditions

$$L_w = 0 \quad \text{or} \quad w = \sum_{i=1}^l \alpha_i y_i x_i, \tag{2.55}$$

$$L_b = 0 \quad \text{or} \quad \sum_{i=1}^l \alpha_i y_i = 0, \tag{2.56}$$

$$L_{\xi_i} = 0 \quad \text{or} \quad C - \mu_i - \alpha_i = 0, \quad i = 1, 2, \dots, l, \tag{2.57}$$

$$\alpha_i [y_i(x_i \cdot w + b) - 1 + \xi_i] = 0, \quad i = 1, 2, \dots, l, \tag{2.58}$$

$$\mu_i \xi_i = 0, \quad i = 1, 2, \dots, l, \tag{2.59}$$

$$\alpha_i \geq 0, \quad i = 1, 2, \dots, l, \tag{2.60}$$

$$\mu_i \geq 0, \quad i = 1, 2, \dots, l, \tag{2.61}$$

and the associated Wolfe dual representation

$$\max \quad \frac{1}{2} \|w\|^2 + C \sum_{i=1}^l \xi_i - \sum_{i=1}^l \mu_i \xi_i - \sum_{i=1}^l \alpha_i [y_i(x_i \cdot w + b) - 1 + \xi_i], \quad (2.62)$$

$$\text{subject to} \quad w = \sum_{i=1}^l \alpha_i y_i x_i, \quad (2.63)$$

$$\sum_{i=1}^l \alpha_i y_i = 0, \quad (2.64)$$

$$C - \mu_i - \alpha_i = 0, \quad i = 1, 2, \dots, l, \quad (2.65)$$

$$\alpha_i \geq 0, \quad \mu_i \geq 0, \quad i = 1, 2, \dots, l. \quad (2.66)$$

By substituting the equality constraints (2.63) and (2.64) into the Lagrangian (2.62), the optimization problem becomes

$$\max \quad \sum_{i=1}^l \alpha_i - \frac{1}{2} \sum_{i,j=1}^l \alpha_i \alpha_j y_i y_j x_i \cdot x_j, \quad (2.67)$$

$$\text{subject to} \quad \sum_{i=1}^l \alpha_i y_i = 0, \quad (2.68)$$

$$0 \leq \alpha_i \leq C, \quad i = 1, 2, \dots, l. \quad (2.69)$$

As before, we use KKT conditions, equations (2.58) and (2.59) to solve for  $b$ . Combining equation (2.57),  $C - \alpha_i - \mu_i = 0$  and equation (2.59),  $\mu_i \xi_i = 0$  in show that  $\xi_i = 0$  if  $\alpha_i < C$ . Therefore we simply take any point that satisfies  $0 < \alpha_i < C$  and  $\xi_i = 0$ , and using equation (2.58) to compute  $b$ . With some algebra, we should obtain the solution for  $b$  that is identical to equation (2.48) from the separable case in the previous section. Once  $\alpha$ ,  $w$ , and  $b$  are determined, we obtain the optimal decision function

$$g(x) = \text{sgn} \left( \sum_{i=1}^l \alpha_i y_i x_i \cdot x + b \right), \quad (2.70)$$

$$\text{where} \quad b = y_m - \sum_{i=1}^l \alpha_i y_i x_i \cdot x_m. \quad (2.71)$$



This decision function is identical to equation (2.41) of the separable case with the only exception that the Lagrange multipliers  $\alpha_i$  are now bounded above by  $C$ , as seen in (2.69) [14].

### 2.3.3 Nonlinearly Separable Case - Kernel trick

In reality, the two classes cannot be linearly separated. Fortunately, there is a simple method that makes the linear SVM work well with non-linear case. The idea relies on the kernel trick that allows us to map the original input space to a higher-dimensional features space where the training set can be linearly separable [18, 57]. Figure 2.7 illustrates the linear separability of the kernel trick.

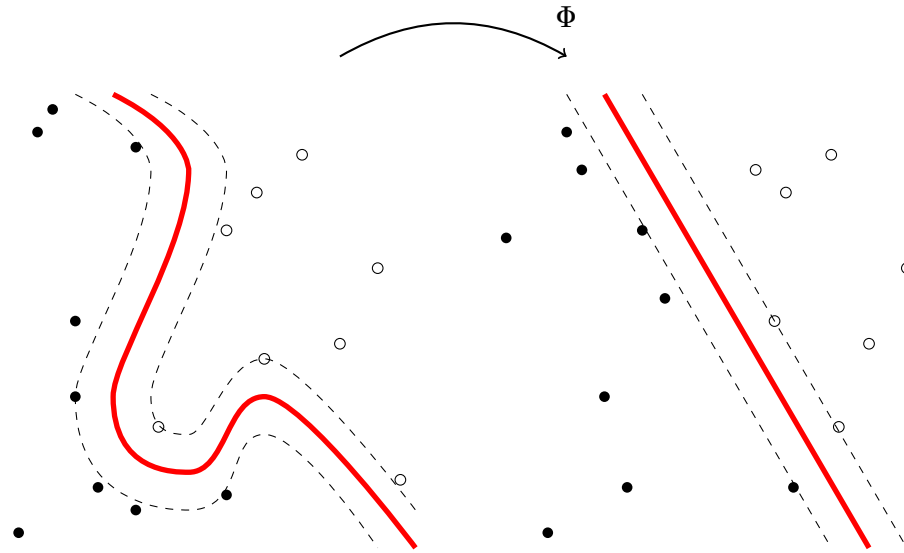


Figure 2.7: Kernels are used for mapping a non-linearly separable problem into a higher dimension linearly separable problem. The source code used to make this figure is adapted from [50].

To see how does mapping to a higher dimensional space provide linear separability, let's consider an example shown in Figure 2.8. Data points from the red class and the black class in Figure 2.8a cannot simply linearly separated in the original space  $\mathcal{X} = \mathbb{R}^2$ . Under

the mapping

$$\Phi: \mathbb{R}^2 \rightarrow \mathbb{R}^3, \quad \Phi(x) = \begin{pmatrix} x_1 \\ x_2 \\ x_1 x_2 \end{pmatrix}, \quad (2.72)$$

the two classes can be linearly separated by the  $x_1 x_2$  - hyperplane, as shown in Figure 2.8b.

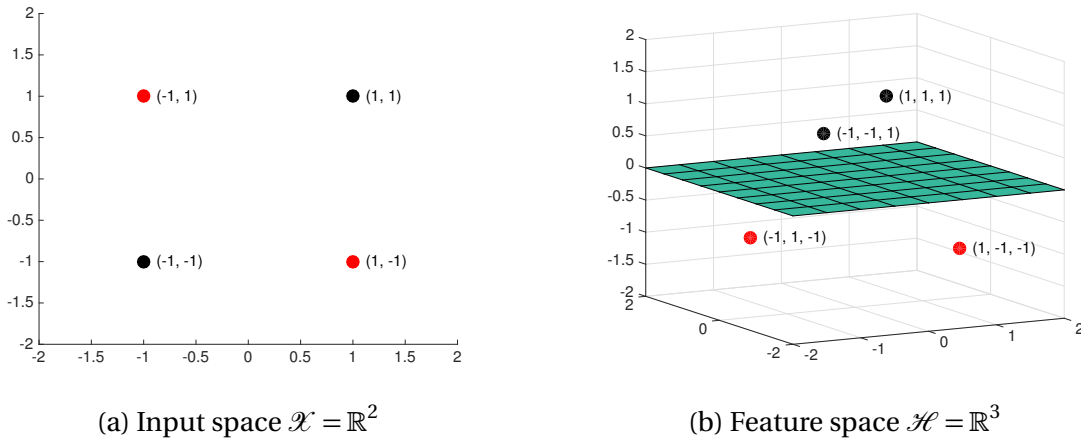


Figure 2.8: An example of kernel trick.

The most widely used kernels in practice are:

$$\text{Linear kernel} \quad K(x_i, x_j) = x_i \cdot x_j, \quad (2.73)$$

$$\text{Polynomial kernel} \quad K(x_i, x_j) = (r + \gamma x_i \cdot x_j)^p, \quad (2.74)$$

$$\text{Gaussian kernel} \quad K(x_i, x_j) = \exp(-\gamma \|x_i - x_j\|^2), \quad (2.75)$$

$$\text{Sigmoid kernel} \quad K(x_i, x_j) = \tanh(r + \gamma x_i \cdot x_j). \quad (2.76)$$

If we have an learning algorithm where examples (training data) appear only in the inner products, we can freely replace the inner product with a different one, so called kernel, where the kernel happens to be an inner product in some feature space. Remember the soft margin SVM in (section 2.3.2) in which examples enter the training algorithm in the

form of inner product, via equations (2.67) - (2.69),

$$\begin{aligned}
& \max \quad \sum_{i=1}^l \alpha_i - \frac{1}{2} \sum_{i,j=1}^l \alpha_i \alpha_j y_i y_j \underbrace{x_i \cdot x_j}_{\text{inner product}}, \\
& \text{subject to} \quad \sum_{i=1}^l \alpha_i y_i = 0, \\
& \quad \quad \quad 0 \leq \alpha_i \leq C.
\end{aligned}$$

Suppose we map the data to some higher dimensional Euclidean space  $\mathcal{H}$ , using a mapping  $\Phi$  such that

$$\Phi: \mathbb{R}^d \rightarrow \mathcal{H}, \quad (2.77)$$

then the SVM algorithm depends only on the data through the inner product in  $\mathcal{H}$  by  $\Phi(x_i) \cdot \Phi(x_j)$ . Moreover, if there exists a kernel function  $K$  such that

$$K(x_i, x_j) = \Phi(x_i) \cdot \Phi(x_j), \quad (2.78)$$

then replacing the inner product by the kernel  $K(x_i, x_j)$ , the Wolfe dual representation becomes

$$\max \quad \sum_{i=1}^l \alpha_i - \frac{1}{2} \sum_{i,j=1}^l \alpha_i \alpha_j y_i y_j \underbrace{K(x_i, x_j)}_{\text{kernel}}, \quad (2.79)$$

$$\text{subject to} \quad \sum_{i=1}^l \alpha_i y_i = 0, \quad (2.80)$$

$$0 \leq \alpha_i \leq C, \quad i = 1, 2, \dots, l \quad (2.81)$$

and the solution has form

$$g(x) = \text{sgn} \left( \sum_{i=1}^l \alpha_i y_i K(x_i, x) + b \right), \quad (2.82)$$

$$\text{where } b = y_m - \sum_{i=1}^l \alpha_i y_i K(x_i, x_m), \quad (2.83)$$

for some support vector  $x_m$  with label  $y_m$ , associated with nonzero Lagrangian multiplier  $\alpha_m$  and  $\xi_m = 0$ . It should be noted that the entire learning SVM algorithm does not require either the higher dimensional space  $\mathcal{H}$  or the mapping  $\Phi$  explicitly but the solution to the optimization problem is still a simple linear combination. That is the beauty of kernel trick.

For a given kernel, not only that computing the associated  $\Phi$  and  $\mathcal{H}$  is irrelevant, both  $\Phi$  and  $\mathcal{H}$  can also be non-unique. For example, suppose that  $x \in \mathbb{R}^2$ , we choose a simple polynomial kernel  $K(x_i, x_j) = (x_i \cdot x_j)^2$  corresponding to (2.74) with  $r = 0$ ,  $p = 2$ , and  $\gamma = 1$ . We can find the mapping  $\Phi$

$$\Phi : \mathbb{R}^2 \rightarrow \mathcal{H}, \quad (2.84)$$

$$\langle \Phi(x), \Phi(y) \rangle = (x \cdot y)^2. \quad (2.85)$$

Here, it's easy to show that all the mappings  $\Phi_1$ ,  $\Phi_2$  and  $\Phi_3$  and the associated feature spaces  $\mathcal{H}_1 = \mathcal{H}_2 = \mathbb{R}^3$ , and  $\mathcal{H}_3 = \mathbb{R}^4$  satisfy condition (2.85). That is, neither the mapping  $\Phi$  nor the space  $\mathcal{H}$  is unique for this given kernel [14].

$$\Phi_1(x) = \begin{pmatrix} x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \end{pmatrix}, \quad \Phi_2(x) = \frac{1}{\sqrt{2}} \begin{pmatrix} (x_1^2 - x_2^2) \\ 2x_1x_2 \\ (x_1 + x_2)^2 \end{pmatrix}, \quad \Phi_3(x) = \begin{pmatrix} x_1^2 \\ x_1x_2 \\ x_1x_2 \\ x_2^2 \end{pmatrix}.$$

It also should be noted that the Gaussian kernel (2.75) can be infinite dimensional. To show this, we need to find a corresponding  $\Phi \in \mathbb{R}^\infty$  for kernel (2.75). Without loss of generality, assume  $\gamma > 0$  and  $x \in \mathbb{R}$ . Expanding the exponential term as the Taylor series,

we have

$$\begin{aligned}
K(x_i, x_j) &= \exp(-\gamma \|x_i - x_j\|^2) \\
&= \exp(-\gamma(x_i - x_j)^2) \\
&= \exp(-\gamma x_i^2 + 2\gamma x_i x_j - \gamma x_j^2) \\
&= \exp(-\gamma x_i^2) \exp(-\gamma x_j^2) \exp(2\gamma x_i x_j) \\
&= \exp(-\gamma x_i^2) \exp(-\gamma x_j^2) \left( \sum_{k=0}^{\infty} \frac{(2\gamma x_i x_j)^k}{k!} \right) \\
&= \sum_{k=0}^{\infty} \left( \exp(-\gamma x_i^2) \sqrt{\frac{(2\gamma)^k}{k!}} x_i^k \exp(-\gamma x_j^2) \sqrt{\frac{(2\gamma)^k}{k!}} x_j^k \right) \\
&= \Phi(x_i) \cdot \Phi(x_j),
\end{aligned}$$

where

$$\Phi(x) = \exp(-\gamma x^2) \left( 1, \sqrt{\frac{2\gamma}{1!}} x, \sqrt{\frac{(2\gamma)^2}{2!}} x^2, \sqrt{\frac{(2\gamma)^3}{3!}} x^3, \dots \right)^T.$$

That is,  $\Phi$  maps  $x \in \mathbb{R}^1$  to  $\mathbb{R}^\infty$ .

In order to determine which kernels does there exist a pair  $(\Phi, \mathcal{H})$  with the desired properties, we rely on the Mercer's condition. In general, any positive semi-definite function that satisfies the Mercer's condition (see Theorem<sup>2</sup> 2.3.1 below) can be used as a kernel function [12, 18, 37, 57].

**Theorem 2.3.1** (Mercer's Theorem [22, 59]) *There exists a mapping  $\Phi$  and an expansion*

$$K(x, y) = \sum_i \Phi(x)_i \Phi(y)_i \tag{2.86}$$

---

<sup>2</sup>Proof of Mercer's theorem is presented in [22, 59].

if and only if, for any  $g(x)$  such that

$$\int g(x)^2 dx \text{ is finite} \quad (2.87)$$

then

$$\int K(x, y)g(x)g(y) dx dy \geq 0. \quad (2.88)$$

Let's apply the above theorem to check if the the dot product  $K(x, y) = (x \cdot y)^p$  is a valid kernel. We must show that for any  $g(x)$  satisfying (2.87),

$$\int \left( \sum_{i=1}^d x_i y_i \right)^p g(x)g(y) dx dy \geq 0. \quad (2.89)$$

As suggested in [14], a typical term in the expansion of  $\left( \sum_{i=1}^d x_i y_i \right)^p$  contributes a term of the form

$$\frac{p!}{r_1!r_2!\cdots(p-r_1-r_2\cdots)!} \int x_1^{r_1} x_2^{r_2} \cdots y_1^{r_1} y_2^{r_2} \cdots g(x)g(y) dx dy, \quad (2.90)$$

to the left hand side of (2.88). This term is equivalent to

$$\frac{p!}{r_1!r_2!\cdots(p-r_1-r_2\cdots)!} \left( \int x_1^{r_1} x_2^{r_2} \cdots g(x) dx \right)^2 \geq 0. \quad (2.91)$$

Kernel construction is an active field of machine learning research [1, 14, 57], however, it does not belong to the scope of this study. We employ only the existing ones such as the linear kernel (SVM-L) and Gaussian kernel (SVM-G) in our experiments with SVM.

## CHAPTER

### 3

# OVERFITTING

Overfitting is the phenomenon where a hypothesis with lower in-sample error  $E_{\text{in}}$  yields a higher out-of-sample error  $E_{\text{out}}$ . In this case,  $E_{\text{in}}$  is no longer useful in choosing the hypothesis that best represents the target function. Often it is the case when the model is more complex than necessary, as Abu-Mostafa (2012) put, "it uses additional degrees of freedom to fit idiosyncrasies (noise), yielding a final hypothesis that is inferior" [1]. An example of overfitting is illustrated in Figure 3.1. Overfitting can also occurs when a hypothesis that is far simpler than the target function, hence it is referred as underfitting<sup>1</sup>.

Overfitting mainly depends on the three parameters: the noise  $\sigma^2$ , the target function complexity  $Q_f$ , and the number of training data points  $N$ . As  $\sigma^2$  increases, more stochastic noise<sup>2</sup> is added to the data. Meanwhile, as  $Q_f$  increases, we add more deterministic noise<sup>3</sup> to

---

<sup>1</sup>We will examine this special case in detail in chapter 5

<sup>2</sup>When learning from data, there are random fluctuations and/or measurements errors in the data which cannot be modeled. This phenomenon is referred to the stochastic noise.

<sup>3</sup>For a given learning problem, there is a best approximation to the target function, the part of the target function outside this best fit acts like noise in the data. This model's inability to approximate  $f$  is referred to

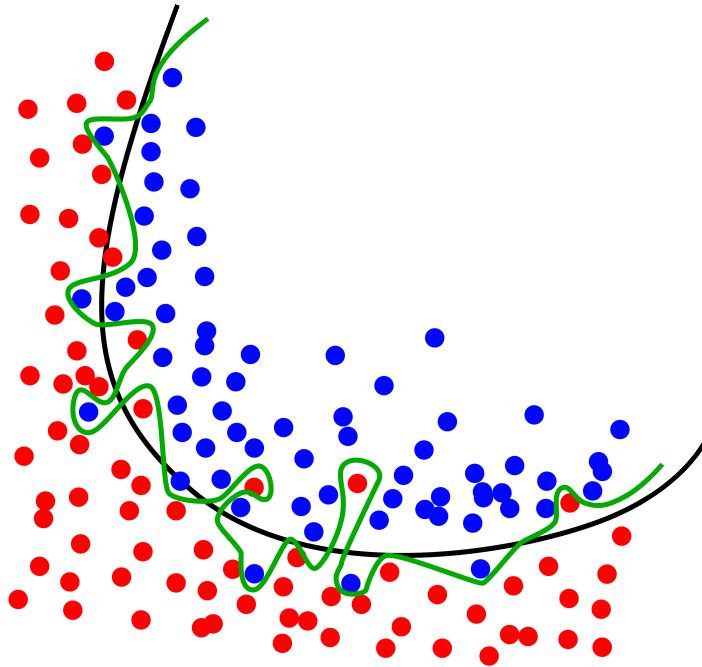


Figure 3.1: Example showing overfitting of a classifier by Chabacano, used under CC BY-SA, via Wikimedia Commons [17]. The green curve separates the blue and the red dots perfectly in this (training) data set, hence it has lower  $E_{\text{in}}$  than that of the black curve. However, it also models the noise at the boundary in addition to model the underlying trend. Hence, it will more likely perform poorly on a new data set from the same population, has higher  $E_{\text{out}}$ . The green curve is an example of an overfitted classifier.



the data. As the number of data points  $N$  increases, noise level  $\sigma^2$  drops and less overfitting occurs. It is important to realize that both stochastic and deterministic noises cannot be modeled and cannot be distinguished [1]. To understand how these two types of noise affect the model performance, one could use the bias-variance decomposition of error (loss) function:

$$E_{\mathcal{D}}[E_{\text{out}}] = \sigma^2 + \text{bias} + \text{var}. \quad (3.1)$$

In equation (3.1),  $\sigma^2$  and **bias** are the direct impact of the stochastic noise and deterministic noise, respectively. The interesting **var** term is indirectly impacted by both types of the noise, through the hypothesis set  $\mathcal{H}$ . Because **var** is controlled by the size of  $\mathcal{H}$ , it decreases as the number of data points  $N$  increases. Moreover, if we make  $\mathcal{H}$  more complex, we will decrease **bias** in the expense of increasing **var**. In practice, the later usually dominates, so overfitting occurs not because of the direct impact on noise, but mainly because of the indirect impact on variance. In this chapter, we mainly adapt the context of Chapter 4 in [1] which gives a throughout discussion to the topic of overfitting and the methodology to prevent overfitting.

### 3.1 Regularization

Regularization is used to prevent overfitting by explicitly controlling the model complexity. To achieve this goal, an additional parameter  $\Omega(h)$  is introduced to account for model complexity of an individual hypothesis  $h$ . As shown in the error function (3.2), instead of minimizing  $E_{\text{in}}(h)$  alone, one would minimize both  $E_{\text{in}}(h)$  and  $\Omega(h)$ . By doing this, the learning algorithm is constrained to not only fitting the data well but also using a simpler hypothesis hence improves generalization [1, 12].

$$E_{\text{out}}(h) \leq E_{\text{in}}(h) + \Omega(\mathcal{H}) \quad \text{for all } h \in \mathcal{H}. \quad (3.2)$$

Several regularization techniques have been presented in the machine learning literature. Some notable ones are weight decay [1], L1 and L2 regularization in regression [48], and Tikhonov regularization [12]. We emphasize on the last technique, most commonly

---

the deterministic noise.

used for ill-posed problems and widely adapted in statistics and machine learning. For example, the author of [11] proves that training with noise is equivalent to solving a Tikhonov regularization with Neural Network classifier. We will also see how it can be viewed as a soft constraint SVM problem.

In the simplest case, given a mapping  $A : X \rightarrow Y$ , to obtain a regularized solution to  $Ax = y$ , we seek for  $x$  that fits  $y$  in the least squares sense, but penalize solutions of large norm and solve the optimization problem

$$\begin{aligned} x_\lambda &= \arg \min \|Ax - y\|_Y^2 + \lambda \|x\|_X^2, \\ &= (A^*A + \lambda I)^{-1} A^* y, \end{aligned} \quad (3.3)$$

where  $\lambda > 0$  is called the regularization parameter.

Back to the supervised learning regime, given  $n$  input-output examples  $(x_1, y_1), \dots, (x_n, y_n)$  with  $x_i \in \mathbb{R}^d$  and  $y_i \in \{-1, 1\}$  for all  $i$ , we want to choose a classifier function  $f : \mathcal{X} \rightarrow \mathcal{Y}$  that fits the data but not too complex. Tikhonov regularization (3.3) suggests such function as follows:

$$\min_{f \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^n V(y_i, f(x_i)) + \lambda \|f\|_{\mathcal{H}}^2, \quad (3.4)$$

where  $V : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}$  is the loss function,  $\|\cdot\|_{\mathcal{H}}$  is the norm on the hypothesis space of functions  $\mathcal{H}$ , and  $\lambda \in \mathbb{R}$  is the regularization parameter [55]. The most intuitive loss function is indeed the missclassification loss function, i.e, the 0-1 loss, which gives 0 if we have classified correctly,  $f(x_i)$  and  $y_i$  have the same sign, and 1 if incorrectly classified. This forms a step function

$$V(y_i, f(x_i)) = \begin{cases} 1 & \text{for } y_i f(x_i) < 0, \\ 0 & \text{for } y_i f(x_i) \geq 0. \end{cases} \quad (3.5)$$

The major drawback is that the step function (3.5) is non convex (also undefined at  $x = 0$ ), which gain difficulties in the optimization problem [1, 55, 57]. To address this shortcoming, one can use the *hinge loss* function. In particular, the *hinge loss* function assigns a positive loss to *correct* classification in such  $0 < y_i f(x_i) < 1$ , as shown in Figure 3.2. The goal is to classify most input  $x_i$  with at least a value of  $y_i f(x_i) \geq 1$ . It's analogous

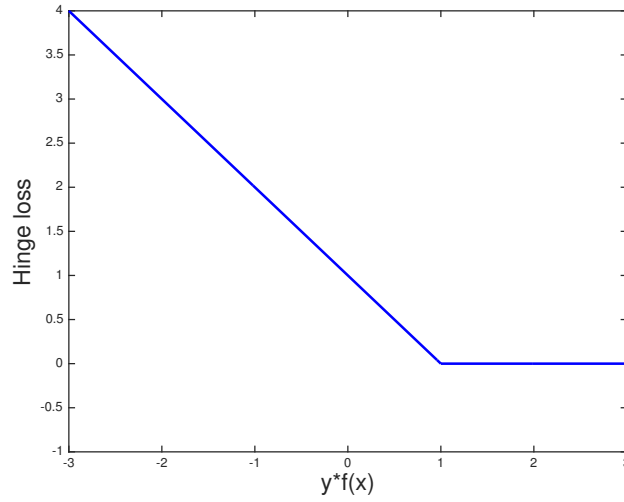


Figure 3.2: Hinge loss function. Note that the loss is asymmetric: incorrect classification,  $y_i f(x_i) < 0$ , is linearly increasing loss as  $y_i f(x_i)$  decreases and correct classification with  $y_i f(x_i) \geq 1$  is always zero loss.

to the idea of SVM when we want to classifies most of  $x_i$  outside of the separating margin. Now, given the hinge-loss function, Tikhonov regularization becomes

$$\min_{f \in H} \frac{1}{n} \sum_{i=1}^n (1 - y_i f(x_i))_+ + \lambda \|f\|_H^2, \quad (3.6)$$

where  $(u)_+ = \max\{u, 0\}$ . Multiplying (3.6) by  $\frac{1}{2\lambda}$  and choosing  $C = \frac{1}{2n\lambda}$  yield

$$\min_{f \in H} \frac{1}{2} \|f\|_H^2 + C \sum_{i=1}^n (1 - y_i f(x_i))_+. \quad (3.7)$$

Note that the hinge loss is not differentiable at  $y_i f(x_i) = 1$ . To address this issue, let's introduce slack variables  $\xi_i$  such that for each point in the training set,  $\xi_i$  replace  $(1 - y_i f(x_i))_+$ .

For each  $\xi_i$ , we require  $\xi_i \geq (1 - y_i f(x_i))_+$ . The Tikhonov regularization problem becomes

$$\begin{aligned} \min_{f \in H} \quad & \frac{1}{2} \|f\|_H^2 + C \sum_{i=1}^n \xi_i, \\ \text{subject to} \quad & y_i f(x_i) \geq 1 - \xi_i, \quad i = 1, \dots, n, \\ & \xi_i \geq 0, \end{aligned} \tag{3.8}$$

which is equivalent to the SVM problem (2.53) in Section 2.3. For a more comprehensive explanation of regularization perspective on SVM, see [11, 14, 55, 57].

## 3.2 Validation

In the previous section, we show that regularization can combat overfitting by control model complexity. Validation on the other hand estimates the out-of-sample error directly, as illustrated in (3.9). Note that both techniques are often used together in a learning problem as an attempt to minimizing  $E_{\text{out}}$  rather than just  $E_{\text{in}}$ .

$$\begin{aligned} E_{\text{out}}(h) &= E_{\text{in}}(h) + \underbrace{\text{overfit penalty}}_{\text{regularization estimates this quantity}}, \\ \underbrace{E_{\text{out}}(h)}_{\text{validation estimates this quantity}} &= E_{\text{in}}(h) + \text{overfit penalty}. \end{aligned} \tag{3.9}$$

The process of validation requires the presence of the validation set. But first let's define the test set. The test set is a subset of input space  $\mathcal{D}$  that is removed from the data set. The test set is absolutely not involved in the learning process, hence unlike  $E_{\text{in}}$ ,  $E_{\text{test}}$  is an unbiased estimation of  $E_{\text{out}}$ . The validation set is constructed similarly, it is also a subset of  $\mathcal{D}$  that is not used directly in the training process. However, the validation error,  $E_{\text{val}}$ , is used to help us make certain choices in choosing parameters of a classifier or feature selection. This indirectly impacts the learning process rendering the validation set no longer qualified to be a test set. Nevertheless,  $E_{\text{val}}$  is still a better choice than  $E_{\text{in}}$  in term of bias for model evaluation.

### 3.2.1 Hold-Out Validation

Hold-out method involves randomly partition the dataset  $\mathcal{D}$  of size  $N$  into two parts: a training set  $\mathcal{D}_{\text{train}}$  of size  $(N - K)$  and a validation set (or hold out-set)  $\mathcal{D}_{\text{val}}$  of size  $K$ . The learning model is trained on the  $\mathcal{D}_{\text{train}}$  to obtain a final hypothesis  $g^-$  that is later used to predict the responses for the observations in the  $\mathcal{D}_{\text{val}}$ <sup>4</sup>. The validation error for  $g^-$  is computed using the validation set  $\mathcal{D}_{\text{val}}$  as follows,

$$E_{\text{val}} = \frac{1}{K} \sum_{x_n \in \mathcal{D}_{\text{val}}} e(g^-(x_n), y_n), \quad (3.10)$$

where  $e(g^-(x), y)$  denotes the point-wise error function,

$$e(g^-(x), y) = \begin{cases} 0 & \text{for } g^-(x) = y, \\ 1 & \text{for otherwise.} \end{cases} \quad (3.11)$$

It is important to realize that the validation error rate  $E_{\text{val}}$  gives an unbiased estimate of  $E_{\text{out}}$  because the hypothesis  $g^-$  was formed independently of the data point in  $\mathcal{D}_{\text{val}}$ . In fact, taking expectation of  $E_{\text{val}}$  with respect to  $\mathcal{D}_{\text{val}}$ , results in

$$\mathbb{E}_{\mathcal{D}_{\text{val}}} [E_{\text{val}}(g^-)] = \frac{1}{K} \sum_{x_n \in \mathcal{D}_{\text{val}}} \mathbb{E}_{\mathcal{D}_{\text{val}}} [e(g^-(x_n), y_n)], \quad (3.12)$$

$$= \frac{1}{K} \sum_{x_n \in \mathcal{D}_{\text{val}}} \mathbb{E}_{x_n} [e(g^-(x_n), y_n)], \quad (3.13)$$

$$= \frac{1}{K} \sum_{x_n \in \mathcal{D}_{\text{val}}} E_{\text{out}}(g), \quad (3.14)$$

$$= E_{\text{out}}(g^-). \quad (3.15)$$

The first equality in (3.12) uses linearity of expectation and the second equality in (3.13) is true because  $e(g^-(x_n), y_n)$  depends only on  $x_n$ . Hold-out method is a very simple strategy but it has two drawbacks: (1) the test error  $E_{\text{out}}$  can be greatly varying depending on which

---

<sup>4</sup>The minus subscript in  $g^-$  indicates that some data points have been removed from the training (for validation purpose). At the end  $g$  is the final hypothesis that is trained with all data points in  $\mathcal{D}_{\text{train}}$ .

observations are included in the training and validation set; and (2) only a subset of the available observations are used for training and validating the model, so we can potentially lose valuable information especially with smaller datasets [37].

### 3.2.2 Cross Validation

In the validation process, finding  $K$ , the size of the validation set is not easy. The dilemma we encounter is described in (3.16). In the first approximation, we want  $K$  to be small, to minimize the difference between  $E_{\text{out}}(g)$  and  $E_{\text{out}}(g^-)$ . However, in the second approximation, larger  $K$  results in less variance between  $E_{\text{out}}(g^-)$  and  $E_{\text{val}}(g^-)$ .

$$E_{\text{out}}(g) \underset{\text{small } K}{\approx} E_{\text{out}}(g^-) \underset{\text{large } K}{\approx} E_{\text{val}}(g^-) \quad (3.16)$$

This leads us to a refinement of the hold-out approach, called cross validation (CV), a common strategy for model selection that gained widespread application due to its simplicity and universality.

The popularity of the cross validation technique is mostly due to the universality of the data splitting heuristics [8]. In the basic approach, called  $k$ -fold cross-validation, the original sample is randomly partitioned into  $k$  equal size subsamples. Commonly,  $k = 10$  is considered standard CV, another choice when  $k = N$  so  $K = 1$  is referred as *leave-one-out* approach. Of the  $k$  subsamples, a single subsample is retained as the validation data for testing the model, and the remaining  $k - 1$  subsamples are used as training data. The cross-validation process is then repeated  $k$  times (the folds), with each of the  $k$  subsamples used exactly once as the validation data. The  $k$  results from the folds can then be averaged (or otherwise combined) to produce a single estimation of the true error rate. That is,

$$E_{\text{cv}}(g^-) = \frac{1}{k} \sum_{n=1}^k E_{\text{val}}(g_n^-). \quad (3.17)$$

The advantage of this method is that all observations are used for both training and validation, and each observation is used for validation exactly once. Not only does  $k$ -fold CV give a nearly unbiased estimate of the generalized (out-of-sample) error rate, it also reduces the variability in the estimation, hence considered better than the hold-out method

in term of avoiding overfitting [37]. However it often produce unpredictable high variability on small dataset. Efron, Bradley (1983) shows that applying randomized bootstrap (a nonparametric maximum likelihood estimation) can enhance the stability of regular CV [27]. Furthermore, Andrew Ng (1996) points out that overfitting can occur even with large dataset that is partially corrupted by noise. The author shows that it can be overcome by selecting the hypothesis with a higher CV error, over others with lower CV errors and propose an algorithm (LOOCVCV<sup>5</sup>) to perform this task.

---

<sup>5</sup>LOOCVCV: Leave-one-out Cross-Validated Cross Validation

## CHAPTER

# 4

## BASEBALL PITCH PREDICTION

Baseball, one of the most popular sports in the world, has a uniquely discrete gameplay structure that naturally allows fans and observers to record information about the game in progress, resulting in a wealth of data that is available for analysis. Major League Baseball (MLB), the professional baseball league in the US and Canada, uses a system known as PITCHf/x to record information about every individual pitch that is thrown in league play. We apply several machine learning classification methods to this data to classify pitches by type (fastball or nonfastball). We then extend the classification to prediction by restricting our analysis to pre-pitch features. By performing significant feature analysis and introducing a dynamic approach for feature selection, moderate improvement over published results is achieved.



## 4.1 PITCHf/x Data

In this study, a database created by Sportvision’s PITCHf/x pitch tracking system that characterizes each pitch with approximately 50 features in which 18 features from the raw data are used directly (see Table 4.1). Additional features are derived from the raw data that we believe to be more relevant to pitch prediction. The motivation for our approach is that prediction, unlike classification, relies on pre-delivery game information. For example, post-delivery features such as speed and curve angle, which can be used to determine whether or not it was a fastball, are not available pre-pitch. So for prediction, we use information from prior pitches in similar game plan situation to judge which pitch can be expected. Some of the adaptive features include the percentage of fastballs thrown in the previous inning, the velocity of the previous pitch, strike result percentage of the previous pitch, and current game count. Overall, we developed a set of 77 features and arranged them into 6 groups of similarity. Group 1 contains features that describe general information of the current game situation such as inning, number of outs, number of base runners, etc. Group 2 features focus on pitch type tendency of the pitcher, such as percentage of fastballs thrown in the previous inning, in previous game, or lifetime percentage of fastballs, etc. Group 3 features aggregate pitch velocity information from the past, while group 4 concerns the location of previous pitches as they cross the home plate. Group 5 consists of features that illustrate different ways to measure *strike result percentage* in a given situation while group 6 does the same for *ball-strike combo* from the similar count in the past. For a full list of features used in each of the 6 groups, see Appendix A.1. In addition, a glossary of baseball terminologies is provided in Appendix A.2.

## 4.2 Related Work

One area of statistical analysis of baseball that has gained attention in the last decade is pitch analysis. Studying pitch characteristics allows baseball teams to develop more successful pitching routines and batting strategies. To aid this study, baseball pitch data produced by the PITCHf/x system is now widely available for both public and private use. PITCHf/x is a pitch tracking system that allows measurements to be recorded and associated

Table 4.1: List of original attributes selected for pitch prediction

<b>PITCHf/x Variables</b>	<b>Description</b>
<i>atbat_num</i>	number of pitches recorded against the specific batter he is facing
<i>outs</i>	number of outs during the at bat
<i>batter</i>	batter's unique identification number
<i>pitcher</i>	pitcher's unique identification number
<i>stand</i>	dominant hand of batter; left/right
<i>p_throws</i>	pitching hand of pitcher; left/right
<i>des</i>	outcome of one pitch from pitcher's perspective; ball/strike/foul/in play, etc.
<i>event</i>	outcome of at bat from batter's perspective; ground out/double/single/walk, etc
<i>pitch_type</i>	classification of pitch type; FF = Four-seam Fastball, SL = Slider, etc.
<i>sv_id</i>	date/time stamp of the pitch; YYMMDD_hhmmss
<i>start_speed</i>	pitch speed, miles per hour
<i>px</i>	horizontal distance of the pitch from the home plate
<i>pz</i>	vertical distance, of the pitch from the home plate
<i>on_first</i>	binary column; display 1 if runner on first, 0 otherwise
<i>on_second</i>	binary column; display 1 if runner on second, 0 otherwise
<i>on_third</i>	binary column; display 1 if runner on third, 0 otherwise
<i>type_confidence</i>	likelihood of the pitch type being correctly classified
<i>ball_strike</i>	display either ball or strike

with every pitch thrown in Major League Baseball (MLB) games. The system, which was installed in every MLB stadium circa 2006, records useful information for every pitch. Some measurements such as the initial velocity, plate velocity, release point, spin angle and spin rate are useful to characterize the pitch type (e.g., fastball, curveball, changeup, knuckleball). The pitch type is determined by PITCHf/x using a proprietary classification algorithm. Because of the large number of MLB games in a season (2430) and the high number of pitches thrown in a game (an average of 146 pitches per team), PITCHf/x provides a rich data set on which to train and evaluate methodologies for pitch classification and prediction. Pitch analysis can either be performed using the measurements provided by PITCHf/x in their raw form or by using data derived features. Each of the recorded pitches is classified by the PITCHf/x proprietary algorithm and provided with measurement data. For the purposes of analysis, the proprietary PITCHf/x classification algorithm is assumed to generally represent truth. For example, in [9] and [10], several classification algorithms including Support Vector Machine (SVM) and Bayesian classifiers were used to classify pitch types based on features derived from PITCHf/x data. The authors evaluated classification algorithms both on accuracy, as compared to the truth classes provided by PITCHf/x, and speed. In addition, Linear Discrimination Analysis (LDA) and Principal Component Analysis (PCA) were used to evaluate feature dimension reduction useful for classification. The pitch classification was evaluated using a set of pitchers' data from the 2011 MLB regular season.

Another important area of ongoing research is pitch prediction, which could have significant real-world applications and potentially provides MLB managers with the statistical competitive edge in simulation using in training and coaching. One example of research on this topic is the work by [31], who use a Linear Support Vector Machine (SVM-L) to perform binary (*fastball* vs. *nonfastball*) classification on pitches of unknown type. The SVM-L model is trained on PITCHf/x data from pitches thrown in 2008 and tested on data from 2009. Across all pitchers, an average prediction accuracy of roughly 70 percent is obtained, though pitcher specific accuracies vary.

Other related pitch data analysis in the literature, such as [65], which examines the pitching strategy of major league pitchers, specifically determining whether or not they (from a game theoretic approach) implement optimally mixed strategies for handling batters. The paper concludes that pitchers do mix optimally with respect to the *pitch*

variable and behave rationally relative to the result of any given pitch.

Our study provides a machine learning approach to pitch prediction, using a binary classification method to predict pitch type, defined as *fastball* vs. *nonfastball*. A distinct difference in our approach is the introduction of an adaptive strategy to feature selection to mimic portions of pitchers' behavior. This allows the machine learning algorithm to select different sets of features in different situations to train the classifiers. The features used contain not only original features, but also hybrid features that are created to better resemble the way pitchers seem to process data. Additionally, cross validation is implemented to detect and avoid overfitting in our predictions. Overall, the prediction accuracy is improved by approximately 8% from results published in [31]. A report of our initial effort in this study can be found in [32].

## 4.3 Our Model

### 4.3.1 Dynamic Feature Selection Approach

A key difference between our approach and former research of [31] is the feature selection methodology. Rather than using a static set of features, an adaptive set is used for each pitcher/count pair. This allows the algorithm to adapt to achieve the best prediction performance result possible on each pitcher/count pair of data.

In baseball there are a number of factors that influence a pitcher's decision, either consciously or unconsciously. For example, one pitcher may not like to throw curveballs during the daytime because the increased visibility makes them easier to spot; however, another pitcher may not make his pitching decisions based on the time of the game. In order to maximize accuracy of a prediction model, one must try to accommodate each of these factors. For example, a pitcher may have particularly good control of a certain pitch and thus favors that pitch, but how can one create a feature to represent its favorability? A potential approach is to create a feature that measures the pitcher's success with a pitch since the beginning of the season, or the previous game, or even the previous batter faced. Which features would best capture the true effect of his preference for that pitch? The answer is that each of these approaches may be best in different situations, so they all must be considered for best accuracy. Pitchers have different dominant pitches, strategies

and experiences; in order to maximize accuracy, our model must be adaptable to various pitching situations. It is noted that in feature space, pitches of same pitch type from different pitchers look different, so classifiers must be trained on each pitcher separately.

Of course, simply adding many features to our model is not necessarily favorable due to the curse of dimensionality. In addition, some features might not be relevant to pitch prediction. Our approach is to change the problem of predicting a pitch into predicting a pitch for each pitcher in a given count. Count, which gives the number of balls and strikes in an at-bat situation, has a significant effect on the pitcher/batter relationship. For example, study by [35] showed that average slugging percentage (a weighted measure of on-base frequency of a batter) is significantly lower in counts that favor the pitcher; however, for neutral counts or counts that favor the batter, there is no significant difference in average slugging percentage. In addition, [31] concluded that pitchers are much more predictable when there are more balls than strikes. These studies show that count is an important factor in making pitch predictions. In order to maximize accuracy, we take an additional step by choosing a most relevant pool of features from the entire available set for each pitcher/count pair. This allows us to maintain our adaptive strategy while controlling dimensionality.

An example of this strategy is shown in Figure 4.1, where we present the selection of predictive features from group 1 for the MLB pitcher Miguel Batista using the data from the 2008 and 2009 seasons when he played for the Seattle Mariners. This figure is segmented by pitching scenarios including batter-favored count, neutral count, and pitcher-favored count. Note that the batter-favored and pitcher-favored counts tend to result in dissimilar feature sets being selected, whereas the neutral counts tend to result in a mixture thereof. More specifically, feature number 6 (lifetime percentage of fastballs thrown) is highly indicative of the *status quo* or standard pitching, it is selected for almost every count except the batter-favored counts (i.e., 3-0, 3-1, 2-0, 1-0), where it is selected only once out of 4 counts (pitch count, 1-0). This shows that our feature selection model adapts to the game situation. The pitcher behaves nominally until he is in an unfavorable count, when he often changes his behavior and a priori value of fastball is no longer selected to use in prediction. We also apply this analysis to the other 5 groups of features. It is noted that each count situation yields a different set of features. There are certainly overlapping features, but there is no unique set of features that are chosen in all counts.

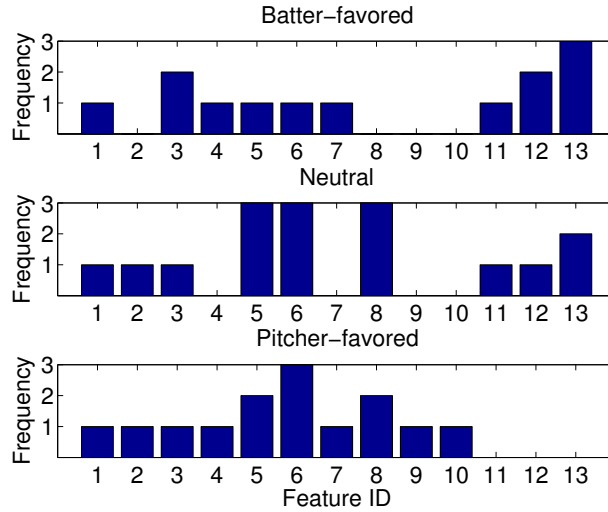


Figure 4.1: Distribution of Group 1 features (1-13) via feature selection method of Miguel Batista (2008-2009).

### 4.3.2 Model Implementation

As discussed above, our feature selection algorithm is adaptive; that is, we seek a good set of features for each pitcher/count situation. The implementation of this adaptive strategy mainly consists of the following three steps. These steps in our adaptive feature selection approach are summarized and depicted in Figure 4.2.

1. Select a subset of features (18) from the raw data and create additional features (59) from the data that are deemed more relevant to pitch prediction. This set of 77 features is further divided into 6 groups of similar features. The number of features from each group varies from 6 to 22 (see the full list in the Appendix A.1).
2. Compute the receiver operating characteristic (ROC) curve and the corresponding area under curve (AUC) for each group of features, then select the most useful features for pitch prediction. In practice, selecting only the best feature provides worse prediction than selecting the best two or three. Hence, at this stage, the size of each group is reduced from 6-22 features to 1-10.
3. Remove all redundant features from our final set. From our grouping, features are

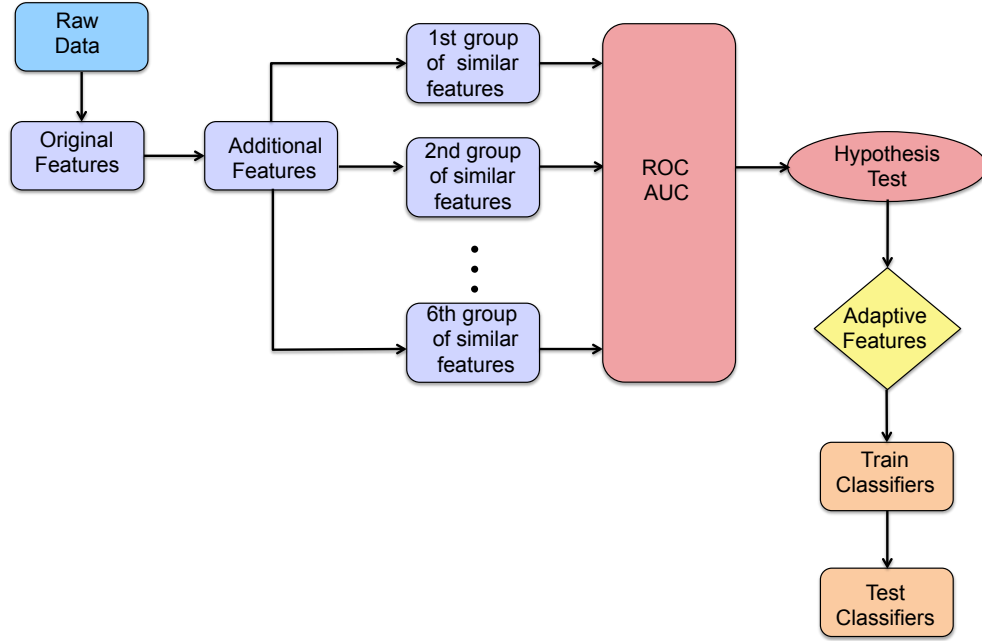


Figure 4.2: Schematic diagram of the proposed adaptive feature selection.

taken based on their relative strength. There is the possibility that a group of features may have poor predictive power. In those instances, we want to prune them from our feature set before we begin to predict. The resulting set of features is pruned by conducting a hypothesis testing to measure the significance of each feature at the  $\alpha = .01$  level.

### 4.3.3 ROC Curves

Receiver operating characteristic (ROC) curves are two-dimensional graphs that are commonly used to visualize and select classifiers based on their performance [30]. They have been used in many applications including signal detection theory [28] and diagnostic tools in clinical medicine [56, 72]. It is noted that a common method to determine the performance of classifiers is to calculate the area under the ROC curve, often denoted by AUC [13]. An example of a ROC curve using data from PITCHf/x pitch tracking system is depicted in Figure 4.3.

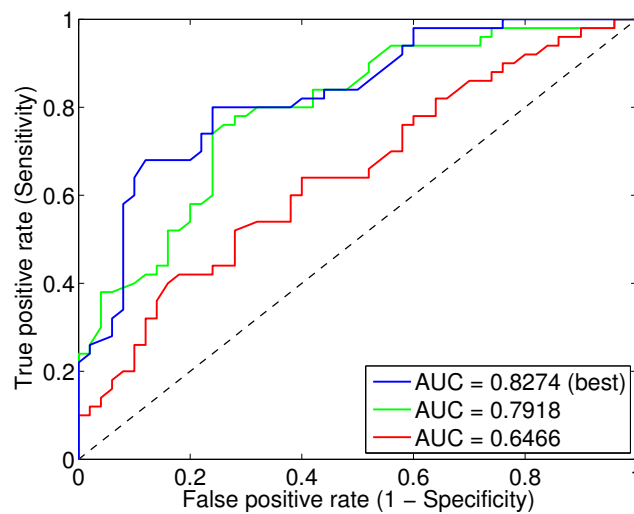


Figure 4.3: ROC curve. In this example, three features are measured with the ROC curve. The blue curve represents the best feature among the three since it has the highest AUC. The diagonal line represents random guessing. The area between a ROC curve and the diagonal line quantifies how much better that feature is at distinguishing the two classes compared to random guessing.



### 4.3.4 Hypothesis Testing

The ability of a feature to distinguish between two classes can be verified by using a hypothesis test. Given any feature  $f$ , we compare  $\mu_1$  and  $\mu_2$ , the mean values of  $f$  in Class 1 (*fastballs*) and Class 2 (*nonfastballs*), respectively. Then we consider

$$H_0 : \mu_1 = \mu_2$$

$$H_A : \mu_1 \neq \mu_2$$

and conduct a hypothesis test based on the student's  $t$  distribution. We compare the  $p$ -value of the test against a significance level of  $\alpha = .01$ . When the  $p$ -value is less than  $\alpha$ , we

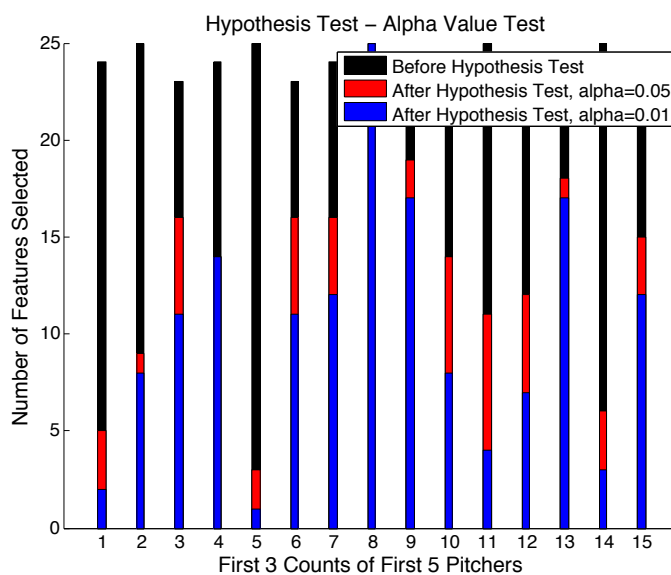


Figure 4.4: Hypothesis testing for feature selection. The black bars are the number of features returned by the ROC curve test, the blue are the features considered optimal for  $\alpha = .01$ , the red are for  $\alpha = 0.05$ .

reject the null hypothesis and conclude that the studied feature means are different for each class, meaning that the feature is significant in separating the classes. In that sense,

this test allows us to remove features which have insignificant separation power.

Figure 4.4 depicts the hypothesis testing step for the first 15 pitcher-count pairs in our dataset. The number of features rise in the optimal set when we increase  $\alpha = .01$  to  $\alpha = .05$  as expected. In our experiment, the accuracy, however, is lowered than the stricter significant level. It is also important to notice that the number of the optimal features is varying throughout different pitcher-count pairs.

### 4.3.5 Classification

Classification is the process of taking an unlabeled data observation and using some rule or decision making process to assign a label to it as presented earlier in Chapter 2. Within the scope of this study, classification represents determining the type of a pitch, i.e., given a pitch  $x$  with characteristics  $x_i$ , determine the pitch type  $x$  is. Several classification can be used to accomplish this task. The methods used in this study are the  $k$ -nearest neighbor ( $k$ -NN), the Linear Discriminant Analysis (LDA) and the Support Vector Machine (SVM), see chapter 2 for a discussion of each method.

## 4.4 Results Analysis

To form a baseline for our prediction results, the prediction model is compared against the naive guess model. The naive guess simply returns the most frequent pitch type thrown by each pitcher, calculated from the training set, see [31]. The improvement factor  $I$  is calculated as follows:

$$I = \frac{A_1 - A_0}{A_0} \times 100 \quad (4.1)$$

where  $A_0$  and  $A_1$  denotes the accuracies of naive guess and our model, respectively.

### 4.4.1 Overall Results

We apply prediction techniques to all pitchers who threw at least 750 pitches in both 2008 and 2009. After performing feature selection on each data subset, each classifier (see Appendix A.3) is trained on data from 2008 and tested on data from 2009. The average classification accuracy for each classifier is computed for test points with a type confidence

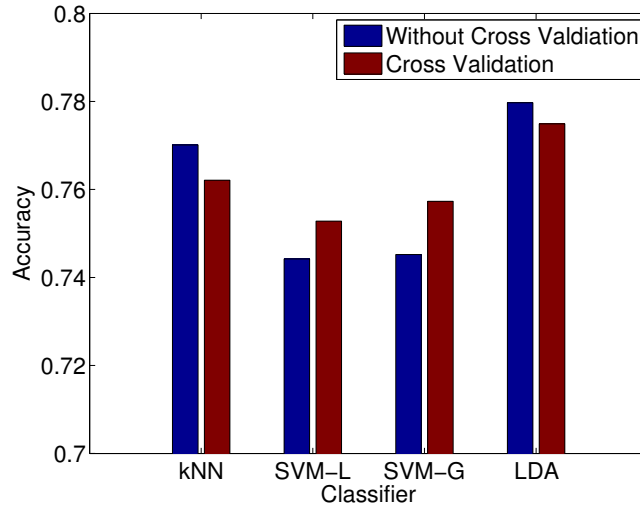


Figure 4.5: Prediction accuracy comparison (percents). Symbol:  $k$ -Nearest Neighbors ( $k$ -NN), Support Vector Machine with linear kernel (SVM-L), and with Gaussian kernel (SVM-G), Linear Discriminant Analysis (LDA), Prediction Tree (PT).

of at least 90%. Type confidence, one of the original features from raw data (see Table 4.1), estimates the accuracy of the class labeling by PITCHf/x, that is, how confidence can we assure that a pitch label (pitch type) is correctly determined.

Figure 4.5 depicts the average accuracies among all applicable pitches in 2009 season. LDA outperforms the other methods at 78% on average accuracy,  $k$ -NN comes closely in second at 77%. Compared to the naive model's prediction accuracy, our model yields a 21% improvement. In previous work, the average prediction accuracy of the 2009 season is 70% with 18% improvement [31]. It should be noted that previous work uses the linear SVM classifier and considers 359 pitchers who threw at least 300 pitches in both the 2008 and 2009 seasons.

In addition, with cross validation implemented, our accuracies remain stable within  $\pm 2\%$  of the original results. This serves as an important confirmation that our results are reliable and our methods would perform well when applied to a newly introduced data set.

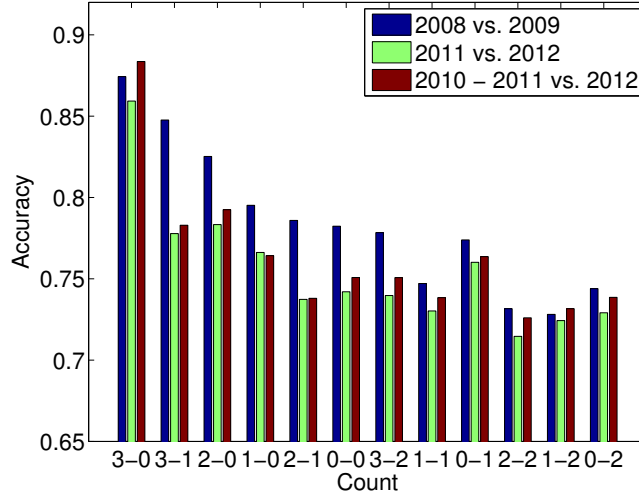


Figure 4.6: Prediction accuracy by count.

#### 4.4.2 By Count Analysis

Figure 4.6 depicts average prediction accuracy per each count situation. Accuracy is significantly higher in batter favored counts and approximately equal in neutral and pitcher favored counts. Prediction is best at the 3-0 count (89%) and worst at the 1-2 and 2-2 counts (73%). In addition, we also calculate prediction accuracy for the 2012 season, using training data from the 2011 season or from both the 2010 and 2011 seasons, shown in Figure 4.6. Even though the size of training data is double in the latter case, we only gain very minimal prediction accuracies at every count situation.

#### 4.4.3 By Pitcher Analysis

We also selected eight pitchers from the 2008 and 2009 MLB regular seasons to examine in details with SVM and  $k$ -NN classifiers.

Table 4.2 describes the training and testing sets. Data from 2008 season were used for training and data from 2009 were used for testing. Table 4.3 depicts the prediction accuracy among the eight pitchers as compared across SVM and  $k$ -NN classifiers as well as naive guess. On average, 79.76% of pitches are correctly predicted by SVM-L and SVM-G classifiers

Table 4.2: Data for each pitcher.

Pitcher	Training Size	Test Size
Fuentes	919	798
Madson	975	958
Meche	2821	1822
Park	1309	1178
Rivera	797	850
Vaquez	2412	2721
Wakefield	2110	1573
Weathers	943	813

Table 4.3: Prediction accuracy comparison (percents). Symbols:  $k$ -Nearest Neighbors ( $k$ -NN), Support Vector Machine with linear kernel (SVM-L), Support Vector Machine with Gaussian kernel (SVM-G), Naive Guess (NG).

Pitcher	$k$ -NN	SVM-L	SVM-G	NG
Fuentes	80.15	78.38	76.74	71.05
Madson	81.85	77.23	79.38	25.56
Meche	72.73	74.83	74.17	50.77
Park	70.31	72.40	71.88	52.60
Rivera	93.51	89.44	90.14	89.63
Vaquez	72.50	72.50	73.05	51.20
Wakefield	100.00	95.50	96.33	100.00
Weathers	76.01	77.76	76.38	35.55
Average	80.88	79.76	79.76	66.04

while  $k$ -NN perform slightly better, at 80.88% accurate. Furthermore,  $k$ -NN is also a better choice in term of computational speed, as noted in Table 4.4. Table 4.5 presents the best

Table 4.4: CPU Times (seconds)

Pitcher	$k$ -NN	SVM-L	SVM-G
Fuentes	0.3459	0.7357	0.3952
Madson	0.3479	0.5840	0.4076
Meche	0.3927	1.2616	0.7270
Park	0.3566	0.7322	0.4591
Rivera	0.4245	0.6441	0.4594
Vaquez	0.4137	1.1282	0.7248
Wakefield	0.4060	0.3057	0.5267
Weathers	0.3480	0.5315	0.3641
Average	0.3794	0.7408	0.5799

Table 4.5: Best Improvement over Naive Guess (percents)

Pitcher	Improvement	Classifier
Fuentes	12.81	$k$ -NN
Madson	22.01	$k$ -NN
Meche	47.39	SVM-L
Park	37.62	SVM-L
Rivera	0.04	$k$ -NN
Vaquez	42.68	SVM-G
Wakefield	0.00	$k$ -NN
Weathers	118.73	SVM-L

classifier that yields highest improvement over Naive guess. In fact, different classifiers perform differently among pitchers.

#### 4.4.4 By Noise Level

PITCHf/x provide a type confidence value associated with every pitch being recorded, which indicates the likelihood of a pitch type being correctly classified. Intuitively, it measures the quality of our data and one can think of instances with low type confidence as having high level of noise. Of course we want to build our prediction algorithm in the noiseless environment. However, as shown in Table 4.6, higher the type confidence cut off thresholds reduce the sizes of testing sets. In fact, majority of test points have an 80% or higher type confidence (lower noise). There is not a significant reduction in test sizes from the

Table 4.6: Prediction results by Type Confidence levels (TC). Symbol:  $k$ -Nearest Neighbors ( $k$ -NN), Support Vector Machine with linear kernel (SVM-L), and with Gaussian kernel (SVM-G), Linear Discriminant Analysis (LDA).

TC (%)	50	60	70	80	90	95	99
Test size	355,755	344,300	332,238	312,574	196,853	24,412	7,150
$k$ -NN	75.94	75.58	75.88	75.72	77.01	84.49	84.63
SVM-L	73.99	73.99	73.99	74.05	74.42	77.00	73.94
SVM-G	74.02	74.03	74.07	74.16	74.52	76.02	72.07
LDA	77.19	77.17	77.13	77.13	77.97	83.19	81.76

50% level (355,755) to the 80% level (312,574), hence prediction performances from all methods remain stable throughout these intervals. Only when the cut-off threshold of type confidence is raised to 90% level, we can notice the reduction in test sizes and the increasing in average prediction accuracies among all methods. We obtain even higher average prediction accuracies at the 95% level where LDA is 83% and  $k$ -NN is 84% accurate. However, there are only 24,412 pitches at this level, less than 7% of all pitches from original test set of about 360,000 pitches, at the 0% level. Hence, we choose the 90% level to be the most reasonable choice to cut off, which contains more than 50% of the original test points. Notice that even at a low type confidence level of 50%, LDA still outperforms other methods, resulting in 78% accuracy. Unlike LDA and  $k$ -NN, SVM based classifiers show only minimal improvement when noisy data are removed. Originally, SVM classifier was built to reduce the impact of noise and to enhance generality in classification (we have shown that the

SVM has Tikhonov regularization embedded in section 3.1 ). Furthermore, only the support vectors (a subset of training data) are used in the final decision function while the rest of the training data are not considered (section 2.3). It would be very likely that those data points with low level of type confidence are not selected to be support vectors. Thus, SVM results are consistency among different levels of type confidence threshold cut-off. This is indeed the motivation for us to focus on implement SVM based classifier for our medical diagnosis work where the datasets are much smaller and the presence of noise is not negligible.



## CHAPTER

# 6

## CONCLUSION AND FUTURE WORK

### 6.1 Pitch Prediction

In baseball pitch prediction, strong predictive features such as speed and spin rate cannot be used. This issue forces us to find an innovative way to enhance the quality of the available feature set. Our research hence revolves around the feature selection step. We highlight some key contributions below.

- Originally, our approach developed from consideration of the factors that affect pitching decisions. For example, the pitcher/batter handedness matchup is often mentioned by sports experts as an effect [31, 35], and was originally included in our model. However, it was discovered that implementing segmentation of data based on handedness has essentially no effect on the prediction results. Thus, handedness is no longer implemented as a further splitting criterion of the model, but this component remains a considered feature. In general, unnecessary data segmentations have a

negative impact solely because they reduce the size of training and testing data for the classifiers to work with.

- The most notable is our dynamic feature selection approach which widely varies the set of features used in each pitcher-count situation. Features that yield strong prediction in some situations fail to provide any benefit in others. In fact, it is interesting to note that in the 2008 vs. 2009 prediction scheme, every feature is used in at least one situation and no feature is used in every situation. It is also interesting to note that the LDA, the most successful classification algorithm of this model is supported by our feature selection technique. In general, as a Bayesian classifier, the LDA relies on a feature independence assumption, which is realistically not satisfied. Our model survives this assumption; even though the features within each of the 6 groups are highly dependent, the final features that are chosen are highly independent.
- The model represents a significant improvement over simple guessing. It is a useful tool for batting coaches, batters, and others who wish to understand the potential pitching implications of a given game scenario. For example, batters could theoretically use this model to increase batting average, assuming that knowledge about a pitch's type makes it easier to hit. The model, for example, is especially useful in certain intense game scenarios and achieves accuracy as high as 90 percent. It is in these game environments that batters can most effectively use this model to translate knowledge into hits.

Looking forward, much can be done to improve the model.

- First, new features would be helpful. There is much game information that we did not include in our model, such as batting averages, slugging percentage per batter, stadium location, weather, and others, which could help improve the prediction accuracy of the model.
- Another potential modification is extension to multi-class classification. Currently, our model makes a binary decision and decides if the next pitch will be a fastball or not. It does not determine what kind of fastball the pitch may be. However, this task is much more difficult and would almost certainly result in a decrease in accuracy.

- Further, prediction is not limited to only the pitch type. Possible prediction objects can be pitch thrown location (specific quadrant), or pitch landing location, if for a given situation the hit occur. That information could be useful to prepare the corresponding defensive player for the impending flight of the ball.

## 6.2 Hepatitis Diagnosis

Our model focuses on combining three key techniques to address imbalance class: weighted cost Support Vector Machine, parameter optimization, and balance metric evaluation. These three techniques complement each other as shown in Figure 6.1. They have been shown to be efficient for providing an accurate and reliable diagnosis for Hepatitis C.

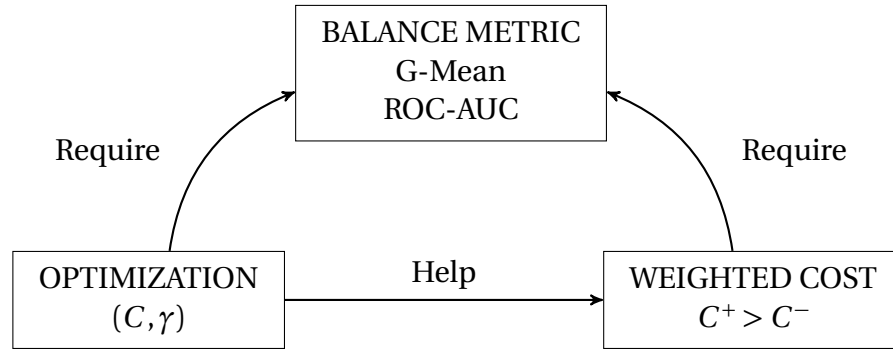


Figure 6.1: Relationship among three main components of this learning model.

We make the following specific contributions in an attempt to design and implement novel techniques to handle class imbalance:

- First, we provide empirical justification showing the existence of overfitting in previous works. For example, 15-NN is 80% accurate, but simply classifies every instances as negative. Although high accuracy can be easily attained, the poor performance in detecting positive instances remains unsolved. Such results cannot be used for disease diagnosis purpose.
- Second, we investigate methodologies that have been successfully addressed class

imbalance. We design our model by combining the weighted cost SVM with the parameters optimization via Grid Search. We also replace accuracy by other metrics such as G-Mean and ROC-AUC that are more appropriate for model evaluation of unbalanced datasets.

- Finally, learning from small datasets can yield inconsistent results even with cross validation. In response, we perform 500 simulation runs on our model and use mean value and 95% confidential interval. This is an attempt to ensure that the result comparison with the benchmark study is accurate and reliable. Our experiments show that this model can effectively use the small and unbalanced medical data to obtain trustworthy diagnosis.

Going forward, we discuss several future research directions as follows.

- Feature selection is a critical step in constructing classifier, it is in fact the center of our the contribution for the baseball pitch prediction. Our attempt of feature selection is applying LFDA to reduce the original 19 features to 2 features before feeding in the classification algorithm. While lower the classification result a little, it greatly enhances consistency with noticeable low variance. To enhance classification accuracy, one potential approach is to determine the right input-output features (currently is 19-2) for LFDA. This is however a tough task against the "Curse of Dimensionality" <sup>1</sup>.
- There does not exist a set rule to pre-determine which kernel is best for a given dataset [1, 6]. In our experiment, we select linear kernel and Gaussian kernel because they have shown good results and run typically fast. However, authors of [23] introduce the notion of kernel alignment which indicates the degree of agreement between a kernel and a given learning task. This notion can potentially be used for kernel selection. Specifically for unbalanced datasets, the authors of [69] proposed the conformal transformation kernels. It is interesting to see whether or not this new class-boundary alignment kernel would interfere with the weighted cost SVM if they coexist in the same learning algorithm.

---

<sup>1</sup>It is common knowledge in machine learning that increasing the number of features increases the accuracy up to a certain point. Thereafter, it degrades the performance. This phenomenon is referred as the "Curse of Dimensionality"[6].

## BIBLIOGRAPHY

- [1] Abu-Mostafa, Y. S. et al. *Learning From Data*. AMLBook, 2012.
- [2] Adeli, M. & Zarabadipour, H. "Automatic disease diagnosis systems using pattern recognition based genetic algorithm and neural networks". *Int. J. Phys. Sci.* v6 i25 (2011), pp. 6076–6081.
- [3] Adeli, M. et al. "New hybrid hepatitis diagnosis system based on Genetic algorithm and adaptive network fuzzy inference system". *Electrical Engineering (ICEE), 2013 21st Iranian Conference on*. IEEE. 2013, pp. 1–6.
- [4] Afif, M. H. et al. "SS-SVM (3SVM): a new classification method for hepatitis disease diagnosis". *Int. J. Adv. Comput. Sci. Appl* 4 (2013).
- [5] Akay, M. F. "Support vector machines combined with feature selection for breast cancer diagnosis". *Expert systems with applications* 36.2 (2009), pp. 3240–3247.
- [6] Akbani, R. *Defending against malicious nodes in closed MANETs through packet authentication and a hybrid trust management system*. The University of Texas at San Antonio, 2009.
- [7] Akbani, R. et al. "Applying support vector machines to imbalanced datasets". *Machine Learning: ECML 2004*. Springer, 2004, pp. 39–50.
- [8] Arlot, S. "A survey of cross-validation procedures for model selection". *Statistics Surveys* 4 (2010), pp. 40–79.
- [9] Attarian, A. et al. "A Comparison of Feature Selection and Classification Algorithms in Identifying Baseball Pitches". *Int. MultiConference of Engineers and Computer Scientists 2013, Lecture Notes in Engineering and Computer Science*. Newswood Limited, 2013, pp. 263–268.
- [10] Attarian, A. et al. "Baseball Pitch Classification: A Bayesian Method and Dimension Reduction Investigation". *IAENG Transactions on Engineering Sciences* (2014), pp. 392–399.
- [11] Bishop, C. M. "Training with noise is equivalent to Tikhonov regularization". *Neural computation* 7.1 (1995), pp. 108–116.
- [12] Bishop, C. M. *Pattern recognition and machine learning*. Springer, 2006.

- [13] Bradley, A. "The use of the area under the ROC curve in the evaluation of machine learning algorithms". *Pattern Recogn.* **30.7** (1997), pp. 1145–1159.
- [14] Burges, C. J. "A tutorial on support vector machines for pattern recognition". *Data mining and knowledge discovery* **2.2** (1998), pp. 121–167.
- [15] Cawley, G. C. & Talbot, N. L. "Manipulation of prior probabilities in support vector classification". *Neural Networks, 2001. Proceedings. IJCNN'01. International Joint Conference on*. Vol. 4. IEEE. 2001, pp. 2433–2438.
- [16] Central, H. *What is Ascites?* URL: <http://www.hepatitiscentral.com/hcv/whatis/ascites/>.
- [17] Chabacano. *Diagram showing overfitting of a classifier*. 2008. URL: <https://commons.wikimedia.org/wiki/File:Overfitting.svg>.
- [18] Chang, C.-C. & Lin, C.-J. "LIBSVM: A library for support vector machines". *ACM Transactions on Intelligent Systems and Technology (TIST)* **2.3** (2011), p. 27.
- [19] Chang, C.-C. & Lin, C.-J. "LIBSVM: A library for support vector machines". *ACM Transactions on Intelligent Systems and Technology* **2** (3 2011). Software available at <http://www.csie.ntu.edu.tw/~cjlin/libsvm>, 27:1–27:27.
- [20] Chawla, N. V. et al. "SMOTE: synthetic minority over-sampling technique". *Journal of artificial intelligence research* (2002), pp. 321–357.
- [21] Chen, H.-L. et al. "A new hybrid method based on local fisher discriminant analysis and support vector machines for hepatitis disease diagnosis". *Expert Systems with Applications* **38.9** (2011), pp. 11796–11803.
- [22] Courant, R. & Hilbert, D. *Methods of mathematical physics*. Vol. 1. CUP Archive, 1966.
- [23] CRISTIANINI, N. "On kernel-target alignment". *Advances in Neural Information Processing Systems* (2002).
- [24] Delen, D. et al. "Predicting breast cancer survivability: a comparison of three data mining methods". *Artificial intelligence in medicine* **34.2** (2005), pp. 113–127.
- [25] Drummond, C. & Holte, R. C. "Exploiting the cost (in) sensitivity of decision tree splitting criteria". *ICML*. 2000, pp. 239–246.
- [26] Duch, W. et al. "Minimal distance neural methods" (1998).

- [27] Efron, B. "Estimating the error rate of a prediction rule: improvement on cross-validation". *Journal of the American Statistical Association* **78.382** (1983), pp. 316–331.
- [28] Egan, J. *Signal detection theory and ROC analysis*. New York: Cognition and Perception. Academic Press, 1975.
- [29] Elkan, C. "The foundations of cost-sensitive learning". *International joint conference on artificial intelligence*. Vol. 17. 1. Citeseer. 2001, pp. 973–978.
- [30] Fawcett, T. "An introduction to ROC analysis". *Pattern recognition letters* **27.8** (2006), pp. 861–874.
- [31] Ganeshapillai, G. & Guttag, J. "Predicting the Next Pitch". *MIT Sloan Sports Analytics Conference*. 2012.
- [32] Hamilton, M. et al. "Applying Machine Learning Techniques to Baseball Pitch Prediction". *3rd Int. Conf. on Pattern Recognition Applications and Methods*. SciTePress, 2014, pp. 520–527.
- [33] Hastie, T. & Tibshirani, R. *The elements of statistical learning*. Springer, 2009.
- [34] He, H., Garcia, E., et al. "Learning from imbalanced data". *Knowledge and Data Engineering, IEEE Transactions on* **21.9** (2009), pp. 1263–1284.
- [35] Hopkins, T. & Magel, R. "Slugging Percentage in Differing Baseball Counts". *Journal of Quantitative Analysis in Sports* **4.2** (2008), p. 1136.
- [36] Hsu, C.-W. et al. *A practical guide to support vector classification*. 2003.
- [37] James, G. et al. *An introduction to statistical learning*. Springer, 2013.
- [38] Japkowicz, N. "The class imbalance problem: Significance and strategies". *Proc. of the Int. Conf. on Artificial Intelligence*. Citeseer. 2000.
- [39] Jerez-Aragónés, J. M. et al. "A combined neural network and decision trees model for prognosis of breast cancer relapse". *Artificial intelligence in medicine* **27.1** (2003), pp. 45–63.
- [40] Jilani, T. A. et al. "PCA-ANN for classification of Hepatitis-C patients". *International Journal of Computer Applications (0975–8887)* **14.7** (2011).

- [41] Karabatak, M. & Ince, M. C. "An expert system for detection of breast cancer based on association rules and neural network". *Expert Systems with Applications* **36.2** (2009), pp. 3465–3469.
- [42] Kubat, M., Matwin, S., et al. "Addressing the curse of imbalanced training sets: one-sided selection". *ICML*. Vol. 97. Nashville, USA. 1997, pp. 179–186.
- [43] Kubat, M. et al. "Learning when negative examples abound". *Machine learning: ECML-97*. Springer, 1997, pp. 146–153.
- [44] Lichman, M. *UCI Machine Learning Repository*. 2013.
- [45] Lin, Y. et al. "Support vector machines for classification in nonstandard situations". *Machine learning* **46.1-3** (2002), pp. 191–202.
- [46] Marckmann, G. "Recommendations for the ethical development and use of medical decision-support systems." *MedGenMed: Medscape general medicine* **3.3** (2001), pp. 5–5.
- [47] Neshat, M. & Yaghobi, M. "Designing a fuzzy expert system of diagnosing the hepatitis B intensity rate and comparing it with adaptive neural network fuzzy system". *Proceedings of the World Congress on Engineering and Computer Science*. Vol. 2. 2009, pp. 797–802.
- [48] Ng, A. Y. "Feature selection, L 1 vs. L 2 regularization, and rotational invariance". *Proceedings of the twenty-first international conference on Machine learning*. ACM. 2004, p. 78.
- [49] Nugroho, A. S. et al. "A solution for imbalanced training sets problem by combnet-ii and its application on fog forecasting". *IEICE TRANSACTIONS on Information and Systems* **85.7** (2002), pp. 1165–1174.
- [50] Peng, Y. *Tikz example-Kernel trick*. 2013. URL: <http://blog.pengyifan.com/tikz-example-kernel-trick/>.
- [51] Peng, Y. *Tikz example-SVM trained with samples from two classes*. 2013. URL: <http://blog.pengyifan.com/tikz-example-svm-trained-with-samples-from-two-classes/>.
- [52] Polat, K. & Güneş, S. "Breast cancer diagnosis using least square support vector machine". *Digital Signal Processing* **17.4** (2007), pp. 694–701.



- [53] Provost, F. "Machine learning from imbalanced data sets 101". *Proceedings of the AAAI 2000 workshop on imbalanced data sets*. 2000, pp. 1–3.
- [54] Şahan, S. et al. "A new hybrid method based on fuzzy-artificial immune system and k-nn algorithm for breast cancer diagnosis". *Computers in Biology and Medicine* **37.3** (2007), pp. 415–423.
- [55] Smola, A. J. et al. "The connection between regularization operators and support vector kernels". *Neural networks* **11.4** (1998), pp. 637–649.
- [56] Swets, J. et al. "Better decisions through science". *Scientific American* **283** (2000), pp. 82–87.
- [57] Theodoridis, S. & Koutroumbas, K. *Pattern recognition*. 4th. Burlington, Mass.: Academic Press, 2009.
- [58] Thoma, M. *Normal distribution for a discriminant analysis*. 2014. URL: <https://commons.wikimedia.org/wiki/File:Lda-gauss-1.svg#globalusage>.
- [59] Vapnik, V. *The nature of statistical learning theory*. Springer Science & Business Media, 2013.
- [60] Vapnik, V. N. "An overview of statistical learning theory". *Neural Networks, IEEE Transactions on* **10.5** (1999), pp. 988–999.
- [61] Veropoulos, K. et al. "Controlling the sensitivity of support vector machines". *Proceedings of the international joint conference on AI*. 1999, pp. 55–60.
- [62] Veterans Affairs, U. D. of. *Viral Hepatitis: Commonly used terms in cirrhosis*. URL: <http://www.hepatitis.va.gov/patient/complications/cirrhosis/terms.asp>.
- [63] Wang, B. X. & Japkowicz, N. "Boosting support vector machines for imbalanced data sets". *Knowledge and Information Systems* **25.1** (2010), pp. 1–20.
- [64] Wang, S.-j. et al. "Empirical analysis of support vector machine ensemble classifiers". *Expert Systems with Applications* **36.3** (2009), pp. 6466–6476.
- [65] Weinstein-Gould, J. "Keeping the Hitter Off Balance: Mixed Strategies in Baseball". *Journal of Quantitative Analysis in Sports* **5.2** (2009), p. 1173.

- [66] West, D. & West, V. "Model selection for a medical diagnostic decision support system: a breast cancer detection case". *Artificial Intelligence in medicine* **20.3** (2000), pp. 183–204.
- [67] West, D. et al. "Ensemble strategies for a medical diagnostic decision support system: A breast cancer diagnosis application". *European Journal of Operational Research* **162.2** (2005), pp. 532–551.
- [68] Wikipedia. *Wikipedia Glossary of Baseball*. 2013. URL: <http://en.wikipedia.org/wiki/Glossary-of-baseball>.
- [69] Wu, G. & Chang, E. Y. "Class-boundary alignment for imbalanced dataset learning". *ICML 2003 workshop on learning from imbalanced data sets II, Washington, DC*. 2003, pp. 49–56.
- [70] Xue, J.-H. & Hall, P. "Why Does Rebalancing Class-unbalanced Data Improve AUC for Linear Discriminant Analysis?" *Pattern Analysis and Machine Intelligence, IEEE Transactions on* **37.5** (2015), pp. 1109–1112.
- [71] Yasin, H. et al. "Hepatitis-C classification using data mining techniques". *International Journal of Computer Applications* **24.3** (2011), pp. 1–6.
- [72] Zweig, M. H. & Campbell, G. "Receiver-operating characteristic ROC plots: A fundamental evaluation tool in clinical medicine". *Clin. Chem.* **39.4** (1993), pp. 561–577.

## **APPENDICES**

## APPENDIX

### A

# BASEBALL PITCH PREDICTION

## A.1 Features in Groups

From the original 18 features given in Table 4.1, we generated a total of 77 features and arranged them into 6 groups as follows:

### Group 1: General information of game situation

1. Inning
2. Time (day/afternoon/night)
3. Number of outs
4. Last at bat events
5. Pitcher vs. batter specific: fastball or nonfastball on previous pitch

6. Pitcher vs. batter specific: lifetime percentage of fastballs
7. Pitcher vs. batter specific: previous pitch's events
8. Numeric score of previous at bat event
9. Player on first base (true/false)
10. Player on second base (true/false)
11. Player on third base (true/false)
12. Number of base runners
13. Weighted base score

## **Group 2: Pitch type tendency from the past**

1. Percentage of fastball thrown in previous inning
2. Percentage of fastball thrown in previous game
3. Percentage of fastballs thrown in previous at bat
4. Lifetime percentage of fastballs thrown to a specific batter over all at bats
5. Percentage of fastballs over previous 5 pitches
6. Percentage of fastballs over previous 10 pitches
7. Percentage of fastballs over previous 15 pitches
8. Percentage of fastballs over previous 20 pitches
9. Previous pitch in specific count: pitch type
10. Previous pitch in specific count: fastball or nonfastball
11. Previous 2 pitches in specific count: fastball/nonfastball combo

12. Previous 3 pitches in specific count: fastball/nonfastball combo
13. Previous pitch: pitch type
14. Previous pitch: fastball or nonfastball
15. Previous 2 pitches: fastball/nonfastball combo
16. Inning
17. Player on first base (true/false)
18. Percentage of fastballs over previous 10 pitches thrown to a specific batter
19. Percentage of fastballs over previous 15 pitches thrown to a specific batter
20. Previous 5 pitches in specific count: percentage of fastballs
21. Previous 10 pitches in specific count: percentage of fastballs
22. Previous 15 pitches in specific count: percentage of fastballs

### **Group 3: Pitch velocity from the past**

1. Previous pitch: velocity
2. Previous 2 pitches: velocity average
3. Previous 3 pitches: velocity average
4. Previous pitch in specific count: velocity
5. Previous 2 pitches in specific count: velocity average
6. Previous 3 pitches in specific count: velocity average

#### **Group 4: Pitch location at home plate**

1. Previous pitch: horizontal position
2. Previous pitch: vertical position
3. Previous 2 pitches: horizontal position average
4. Previous 2 pitches: vertical position average
5. Previous 3 pitches: horizontal position average
6. Previous 3 pitches: vertical position average
7. Previous pitch: zone (Cartesian quadrant)
8. Previous 2 pitches: zone (Cartesian quadrant) average
9. Previous 3 pitches: zone (Cartesian quadrant) average
10. Previous pitch in specific count: horizontal position
11. Previous pitch in specific count: vertical position
12. Previous 2 pitches in specific count: horizontal position average
13. Previous 2 pitches in specific count: vertical position average
14. Previous 3 pitches in specific count: horizontal position average
15. Previous 3 pitches in specific count: vertical position average
16. Previous pitch in specific count: zone (Cartesian quadrant)
17. Previous 2 pitches in specific count: zone (Cartesian quadrant) average
18. Previous 3 pitches in specific count: zone (Cartesian quadrant) average

## **Group 5: Strike-result-percentage**

Strike-result percentage (SRP): a metric we created that measures the percentage of strikes from all pitches in the given situation.

1. SRP of fastball thrown in the previous inning
2. SRP of fastball thrown in the previous game
3. SRP of fastball thrown in the previous 5 pitches
4. SRP of fastball thrown in the previous 10 pitches
5. SRP of fastball thrown in the previous 15 pitches
6. SRP of fastball thrown in previous 5 pitches thrown to the currently facing batter
7. SRP of nonfastball thrown in the previous inning
8. SRP of nonfastball thrown in the previous game
9. SRP of nonfastball thrown in the previous 5 pitches
10. SRP of nonfastball thrown in the previous 10 pitches
11. SRP of nonfastball thrown in the previous 15 pitches
12. SRP of nonfastball thrown in previous 5 pitches thrown to the currently facing batter

## **Group 6: Ball-strike combo from the similar count in the past**

1. Previous pitch: ball or strike (boolean)
2. Previous 2 pitches: ball/strike combo
3. Previous 3 pitches: ball/strike combo
4. Previous pitch in specific count: ball or strike
5. Previous 2 pitches in specific count: ball/strike combo
6. Previous 3 pitches in specific count: ball/strike combo



## A.2 Baseball Glossary and Info

Most of these definitions are obtained directly from [68].

1. **Strike Zone:** A box over the home plate which defines the boundaries through which a pitch must pass in order to count as a strike when the batter does not swing. A pitch that does not cross the plate through the strike zone is a ball.
2. **Strike:** When a batter swings at a pitch but fails to hit it, when a batter does not swing at a pitch that is thrown within the strike zone, when the ball is hit foul and the strike count is less than 2 (a batter cannot strike out on a foul ball, however he can fly out), when a ball is bunted foul, regardless of the strike count, when the ball touches the batter as he swings at it, when the ball touches the batter in the strike zone, or when the ball is a foul tip. Three strikes and the batter is said to have struck out.
3. **Ball:** When the batter does not swing at the pitch and the pitch is outside the strike zone. If the batter accrues four balls in an at bat he gets a walk, a free pass to first base.
4. **Hit-By-Pitch:** When the pitch hits the batters body. The batter gets a free pass to first base, similar to a walk.
5. **Hit:** When the batter makes contact with the pitch and successfully reaches first, second or third base. Types of hits include single (batter ends at first base), doubles (batter ends at second base), triple (batter ends at third base) and home-run.
6. **Out:** When a batter or base runner cannot, for whatever reason, advance to the next base. Examples include striking out (batter can not advance to first), grounding out, popping out and lining out.
7. **Count:** Is the number of balls and strikes during an at bat. There are 12 possible counts spanning every combination of 0-3 balls (4 balls is a walk) and 0-2 strikes (3 strikes is a strikeout).

8. Run: When a base runner crosses home plate. This is a point for that player's team. The outcome of a baseball game is determined by which team has more runs at the end of nine Innings.
9. Inning: One of nine periods of playtime in a standard game.
10. Slugging Percentage: A measure of hitter power. Defined as the average number of bases the hitter earns per at bat.
11. Batting Average: The percentage of time the batter earns a hit.
12. At Bat : A series of pitches thrown by the pitcher to one hitter resulting in a hit, a walk, or an out.

### **A.3 Software**

All data processing and classification tasks were performed off-line using commercial software package MATLAB (R2014a), The MathWorks Inc., Natick, MA.

1.  $k$ -NN(s, c):  $k$ -nearest-neighbors algorithm (MATLAB: *knnsearch*) with standardized Euclidean metric/Manhattan metric
2. SVM (L, G): Support Vector Machine with linear kernel and with Gaussian kernel
3. LDA: Linear discriminant analysis (MATLAB: *classify*) with linear and quadratic types.