Data-driven Kernels for Support Vector Machines

by

Xin Yao

A research paper
presented to the University of Waterloo
in partial fulfillment of the
requirement for the degree of
Master of Mathematics
in
Computational Mathematics

Supervisor: Prof. Yuying Li

Waterloo, Ontario, Canada, 2013

Abstract

Kernel functions can map data points to a non-linear feature space implicitly, and thus significantly enhance the effectiveness of some linear models like SVMs. However, in current literature, there is a lack of a proper kernel that can deal with categorical features naturally.

In this paper, we present a novel kernel for SVM classification based on data-driven similarity measures to compute the similarity between two samples. It then followed by the ensemble selection method to combine all similarity measures. Our kernels can make full use of information from both labeled and unlabeled data, which leads to a semi-supervised approach. Experiment results show that our approaches largely improve the performance of SVMs on all metrics we used. Especially, in the unbalanced class case, keeping other performance measures on an excellent level, two methods can enhance the prediction accuracy on the minority class significantly.

Acknowledgements

I would like to thank all the people who have made this work possible.

Table of Contents

List of Tables **vii**

List of Figures **viii**

1 Introduction **1**

2 Support Vector Machines **3**

2.1 Support Vector Machine Formulations 3

2.1.1 Hard-Margin Support Vector Machines 3

2.1.2 Soft-Margin Support Vector Machines 7

2.2 Kernel Trick 11

3 Methodology **14**

3.1 Similarity Measures 14

3.2 Ensemble Selection Method 17

3.3 Approach 20

4 Numerical Results **21**

4.1 Data Preprocessing and Cleaning 22

4.2 Feature Selection 23

4.3 Results of SVMs with the RBF kernel 24

4.4 Results of SVMs with the Lin Similarity Measure 25

4.5 Results of the Ensemble Selection Method 27
4.6 Comparison of Three Methods 29

5 Conclusion **32**

References **32**

List of Tables

3.1 Similarity Measure for Categorical Features 16

4.1 Frequency of labels of churn, appetency and up-selling 22
4.2 Top 5 feature importance scorse from random forests 24
4.3 Performance of RBF kernel SVMs. 26
4.4 Performance of SVMs with Lin similarity measure. 27
4.5 Performance of the Ensemble Selection Method. 29
4.6 Best performance of each method 31

List of Figures

2.1 Two separating hyperplane with different margins. 5
2.2 Inseparable case in a two-dimensional space 8
2.3 The idea of non-linear SVM. 12

4.1 ROC and Precision-Recall Curves of SVMs with Lin Similarity Measure . . 25
4.2 ROC and Precision-Recall Curves of Ensemble Selection Method 28
4.3 ROC and Precision-Recall Curves of the Ensemble Selection Method . . . 30

Chapter 1

Introduction

Over the past decade, Support Vector Machine (SVM) has become a significant tool for solving classification problems in statistical learning and data mining. A natural way of solving classification problems is seeking a hyperplane to separate the classes. With perfectly separable classes, there exists infinite possible separating hyperplanes. SVM is based on the principle of finding the "best" hyperplane which is the one that is farthest away from the classes. This principle is called maximal margin principle. The hyperplane can be obtained by solving a convex quadratic optimization problem. However, in the real world, most problems are not perfectly separable. Hence, a nonlinear transformation needs to be done to separate classes effectively in another high dimensional space. Through kernel functions, we can map the input features to this nonlinear feature space, without computing the mapping explicitly. In this space, SVM finds a linear separating hyperplane with the maximal margin. An important advantage of the kernel-based SVM is that the mapping can be done using the "kernel trick". The trick computes this transformation implicitly and efficiently. Another advantage is that using ℓ_2 penalty, SVM successfully stabilizes the solution and overcomes over-fitting especially in high-dimensional problems. Consequently, choosing a proper kernel function is central for the effectiveness of a SVM model.

In previous studies, many kernel functions have been proposed to capture properties of data. Radial basis function (RBF) kernels are the most common choice and has been used extensively in the machine learning field. Using RBF kernels means choosing radial basis function networks [9] as the prior hypothesis. Diffusion kernels [17] are widely used to analyze the network data. They can be used to unfold correlated structures between nodes of networks in an Euclidean space. As natural language text mining becomes increasingly important, by assigning to each pair of elements (strings) an "inner product" in a feature space, string kernels [19] are employed especially for text classification tasks.

Although many kernel functions have been developed for different types of data, there is a lack of a proper kernel to deal with categorical data naturally. Categorical data is one kind of data that each value can take one of a limited and usually fixed number of possible values. Categorical data widely exists in many data mining problems, for example, customer behavior data mining tasks, when predicting customer loyalty or managing customer relationship in modern marketing strategies. Problems, containing categorical features with many different possible values, a large number of missing values, and unbalanced class proportions, attract more and more attention in data mining field these days [2, 13].

In this research project, we explore a novel kernel based on data-driven similarity measures which are used to compute the similarity between categorical data instances. Our motivation comes from two facts: kernel functions can be viewed as a similarity measure between two input vectors, and using information of both labeled and unlabeled sets may significantly improve performance. The latter approach usually is called semi-supervised learning in the literature. Moreover, in order to combine different similarity measures, the ensemble selection [7] approach is used to optimize the overall performance of diverse similarity kernel-based SVMs. Under the SVM framework, we compare the performance of our kernel with the standard RBF kernel and other leading classification models on a customer loyalty prediction problem.

The rest of the paper is organized as follows: Chapter 2 provides some background knowledge on SVM and kernel trick. Chapter 3 describes our novel kernel with similarity measures and the ensemble selection method. Chapter 4 compares performance of our approach with common used RBF kernels on customer loyalty data. Chapter 5 summarizes the paper with further discussion.

Chapter 2

Support Vector Machines

In this chapter, we discuss SVM for the two-class classification problem. Firstly, we discuss hard-margin SVMs, which assumes that the training data points are linearly separable in the input space. Although it cannot be used in many real world situations, it is the easiest model to understand and it forms the foundation of more complex SVMs. Secondly, we extend hard-margin SVMs to the soft-margin SVMs which is for the case when training data points are not linearly separable. Then we introduce the "kernel trick" to do classification in the nonlinear feature space to enhance separability.

2.1 Support Vector Machine Formulations

2.1.1 Hard-Margin Support Vector Machines

The discussion follows the formulation of [22] and [24]. In a two-class classification problem, we have M m-dimensional training data points $\mathbf{x}_i$ $(i = 1, \ldots, M)$ and the associated class labels be $y_i \in \{-1, +1\}$. SVMs seek an optimal hyperplane to separate the two classes.

A hyperplane in $\mathbb{R}^m$ can be described with a linear equation with the following form for some nonzero vector $\mathbf{w} \in \mathbb{R}^m$ and $b \in \mathbb{R}$

$$\mathbf{w}^\top \mathbf{x} + b = 0. \tag{2.1}$$

The hyperplane

$$y_i \left(\mathbf{w}^\top \mathbf{x}_i + b\right) \geq c, \quad \text{for } i = 1, \ldots, M, \quad \text{and} \quad c > 0, \tag{2.2}$$

forms a separating hyperplane that separates $\{\mathbf{x}_i : y_i = 1\}$ and $\{\mathbf{x}_i : y_i = -1\}$.

A hyperplane can be arbitrarily rescaled, for example,

$$\mathbf{w}^\top \mathbf{x} + b = 0 \text{ is equaivalent to } s\left(\mathbf{w}^\top \mathbf{x} + b\right) = 0,\ \forall s \neq 0. \tag{2.3}$$

In particular, the hyperplane (2.2) can be reparameterized as

$$y_i\left(\mathbf{w}^\top \mathbf{x}_i + b\right) \geq 1, \quad \text{for } i = 1, \ldots, M. \tag{2.4}$$

A separating hyperplane satisfying condition (2.4) is called a canonical separating hyperplane.

This way, we can have a decision function

$$\mathbf{w}^\top \mathbf{x}_i + b \begin{cases} > 0 & \text{for} \quad y_i = 1, \\ < 0 & \text{for} \quad y_i = -1. \end{cases} \tag{2.5}$$

Since the two classes are linearly separable, there exists an infinite number of separating hyperplanes. The distance between the separating hyperplane and the training data point nearest to the hyperplane is called the margin. Figure 2.1 shows two hyperplanes that satisfy (2.4) but with different margins. Intuitively, the hyperplane with a larger margin has a higher generalization ability. SVMs are based on the notion of seeking the hyperplane with the maximum margin, which is called the optimal separating hyperplane. Mathematically, the margin can be formulated as

$$\text{margin} = 2 \times \min\left\{y_i d_i,\ i = 1, \ldots, M\right\}, \tag{2.6}$$

where d_i is the signed distance between instance $\mathbf{x}_i$ and the hyperplane.

It can be shown that d_i is equal to [14]

$$d_i = \frac{1}{||\mathbf{w}||}\left(\mathbf{w}^\top \mathbf{x}_i + b\right). \tag{2.7}$$

Then, equations (2.4) , (2.7) and (2.6) together imply that the margin of a canonical separating hyperplane is equal to

$$2 \times \min\left\{y_i d_i\right\} = \frac{2}{||\mathbf{w}||}. \tag{2.8}$$

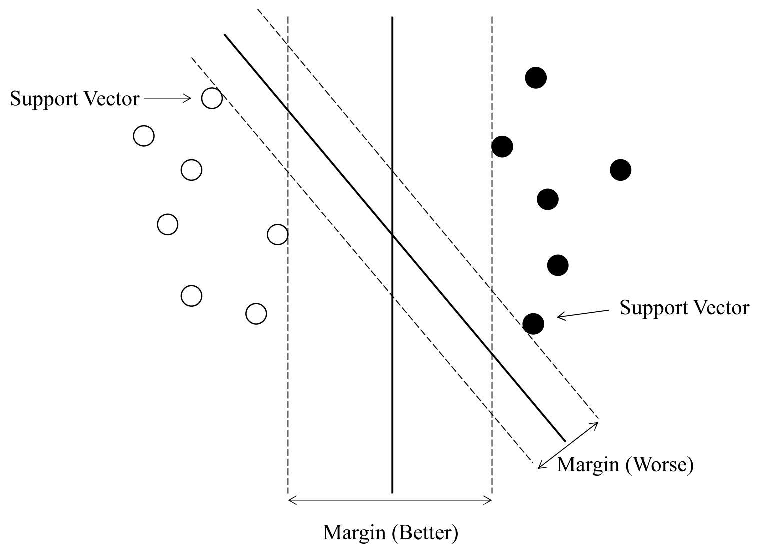

Figure 2.1: Two separating hyperplane with different margins. Adapted from [24]

Therefore, the optimal separating hyperplane can be obtained by solving the following optimization problem for $\mathbf{w}$ and b:

$$\min \ \frac{1}{2}||\mathbf{w}||^2 \tag{2.9}$$

$$\text{s.t.} \ \ y_i \left(\mathbf{w}^\top \mathbf{x}_i + b\right) \geq 1, \quad \text{for } i = 1, \ldots, M. \tag{2.10}$$

Problem (2.9) is a convex quadratic programming problem. The assumption of linear separability means that there exist $\mathbf{w}$ and b that satisfy (2.10). Because the optimization problem has a convex quadratic objective function with linear inequality constraints, even if the solutions are not unique, a local minimizer is always a global minimizer. This is one of the advantages of support vector machines.

This constrained optimization problem can be solved by introducing Lagrange multipliers $\alpha_i \geq 0$ and a Lagrangian

$$L(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\mathbf{w}^\top \mathbf{w} - \sum_{i=1}^{M} \alpha_i \left(y_i \left(\mathbf{w}^\top \mathbf{x}_i + b\right) - 1\right), \tag{2.11}$$

where $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_M)^\top$. The Lagrangian L has to be minimized with respect to the primal variables $\mathbf{w}$ and b, and maximized with respect to the dual variables α_i, i.e. a saddle point has to be found.

The solution satisfies the following Karush-Kuhn-Tucker (KKT) conditions:

$$\begin{aligned}
\frac{\partial}{\partial \mathbf{w}} L(\mathbf{w}, b, \boldsymbol{\alpha}) &= \mathbf{0}, && (2.12)\\
\frac{\partial}{\partial b} L(\mathbf{w}, b, \boldsymbol{\alpha}) &= 0, && (2.13)\\
y_i\left(\mathbf{w}^\top \mathbf{x}_i + b\right) &\geq 1 \quad \text{for } i = 1, \ldots, M, && (2.14)\\
\alpha_i\left(y_i\left(\mathbf{w}^\top \mathbf{x}_i + b\right) - 1\right) &= 0 \quad \text{for } i = 1, \ldots, M, && (2.15)\\
\alpha_i &\geq 0 \quad \text{for } i = 1, \ldots, M. && (2.16)
\end{aligned}$$

Specifically, equations (2.15) which link inequality constraints and their associated Lagrange multipliers are called KKT complementarity conditions.

Equation (2.15) implies that either $\alpha_i = 0$, or $\alpha_i \neq 0$ and $y_i\left(\mathbf{w}^\top \mathbf{x}_i + b\right) = 1$, must be satisfied. The training data points, whose $\alpha_i \neq 0$, are called Support Vectors. Support vectors lie on the margin (Figure 2.1). All remaining samples do not show up in the final decision function: their constraints are inactive at the solution. This nicely captures the intuition of the problem that the hyperplane is determined by the points closest to it.

Using (2.11), (2.12) and (2.13) are reduced to

$$\mathbf{w} = \sum_{i=1}^{M} \alpha_i y_i \mathbf{x}_i \tag{2.17}$$

and

$$\sum_{i=1}^{M} \alpha_i y_i = 0. \tag{2.18}$$

By substituting (2.17) and (2.18) into (2.11), one eliminates the primal variables and arrives at the dual problem:

$$\max \sum_{i=1}^{M} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{M} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^\top \mathbf{x}_j \tag{2.19}$$

$$\text{s.t.} \quad \sum_{i=1}^{M} \alpha_i y_i = 0, \quad \text{and} \quad \alpha_i \geq 0, \;\; i = 1, \ldots, M. \tag{2.20}$$

The formulated support vector machine is called the hard-margin support vector machine. Because

$$\frac{1}{2}\sum_{i,j=1}^{M}\alpha_i\alpha_j y_i y_j \mathbf{x}_i^\top \mathbf{x}_j = \frac{1}{2}\left(\sum_{i=1}^{M}\alpha_i y_i \mathbf{x}_i\right)^\top \left(\sum_{i=1}^{M}\alpha_i y_i \mathbf{x}_i\right) \geq 0,\ \forall \alpha_i \tag{2.21}$$

maximizing (2.19) under the constraints (2.20) is a concave quadratic programming problem. If a solution exists, specially, if the classification problem is linearly separable, the global optimal solution α_i $(i = 1, \ldots, M)$ exists. For convex quadratic programming, the values of the primal and dual objective functions coincide at the optimal solutions if they exist, which is called zero duality gap.

Data that are associated with $\alpha_i \neq 0$ are support vectors for class with 1 or -1. Then from (2.5) and (2.17) the decision function is given by

$$D(\mathbf{x}) = \sum_{i \in S}\alpha_i y_i \mathbf{x}_i^\top \mathbf{x} + b, \tag{2.22}$$

where S is the set of support vectors, and from the KKT conditions, b is given by

$$b = y_i - \mathbf{w}^\top \mathbf{x}_i, \quad \text{for } i \in S. \tag{2.23}$$

Then unknown data sample $\mathbf{x}$ is classified into

$$\begin{cases} y = +1 & \text{if} \quad D(\mathbf{x}) > 0, \\ y = -1 & \text{if} \quad D(\mathbf{x}) < 0. \end{cases} \tag{2.24}$$

If $D(\mathbf{x}) = 0$, $\mathbf{x}$ is on the boundary and thus is unclassifiable. When the training data are separable, the region $\{\mathbf{x} \mid -1 < D(\mathbf{x}) < 1\}$ is the generalization region.

2.1.2 Soft-Margin Support Vector Machines

In the hard-margin SVM, we assume that the training data are linearly separable. But in practice, a separating hyperplane may not exist, e.g. if a high noise level causes a large overlap of the classes. As a result, there is no feasible solution to the hard-margin SVM so that it becomes unsolvable. Here we extend the hard-margin SVM so that it is applicable to an inseparable case.

To allow for the possibility of data points violating (2.10), we introduce slack variables

$$\xi_i \geq 0,\ i = 1, \ldots, M \tag{2.25}$$

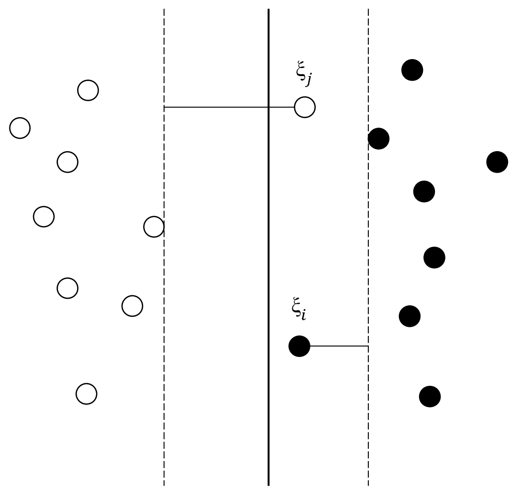

Figure 2.2: Inseparable case in a two-dimensional space

along with relaxed constraints

$$y_i \left(\mathbf{w}^\top \mathbf{x}_i + b\right) \geq 1 - \xi_i,\ i = 1, \ldots, M. \tag{2.26}$$

With the slack variable ξ_i, feasible solutions always exist. For the training data $\mathbf{x}_i$, if $0 < \xi_i < 1$ (ξ_i in Figure 2.2), the data does not have the maximum margin but are still correctly classified. But if $\xi_i \geq 1$ (ξ_j in Figure 2.2) the data is misclassified by the optimal hyperplane.

In order to find the hyperplane which generalizes well, both the margin and the number of training errors need to be controlled. Thus, we consider the following optimization problem:

$$\begin{aligned} \min\ \frac{1}{2}||\mathbf{w}||^2 \quad &+ \quad C\sum_{i=1}^{M}\xi_i && (2.27)\\ \text{s.t. } y_i\left(\mathbf{w}^\top \mathbf{x}_i + b\right) \quad &\geq \quad 1 - \xi_i,\ \ i = 1, \ldots, M, && (2.28)\\ \xi_i \quad &\geq \quad 0, \qquad i = 1, \ldots, M. \end{aligned}$$

where C is the margin parameter that determines the trade-off between the maximization of the margin and the minimization of the classification error. A larger C penalizes more on classification errors, consequently, leading to a lower classification error but a smaller margin. As C decreases, we will have a larger margin but a higher classification error. Because of slack variables, we call the SVM the soft-margin-SVM.

Similar to the linearly separable case, introducing the nonnegative Lagrange multipliers α_i and β_i, we obtain

$$\begin{aligned} L\left(\mathbf{w},\, b,\, \boldsymbol{\xi},\, \boldsymbol{\alpha},\, \boldsymbol{\beta}\right) &= \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{M}\xi_i - \sum_{i=1}^{M}\beta_i\xi_i \\ &\quad -\sum_{i=1}^{M}\alpha_i\left(y_i\left(\mathbf{w}^\top\mathbf{x}_i + b\right) - 1 + \xi_i\right), \end{aligned} \tag{2.29}$$

where $\boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_M)^\top$, $\boldsymbol{\xi} = (\xi_1, \dots, \xi_M)^\top$, and $\boldsymbol{\beta} = (\beta_1, \dots, \beta_M)^\top$.

For the optimal solution, the following KKT conditions are satisfied:

$$\frac{\partial L\left(\mathbf{w},\, b,\, \boldsymbol{\xi},\, \boldsymbol{\alpha},\, \boldsymbol{\beta}\right)}{\partial \mathbf{w}} = \mathbf{0}, \tag{2.30}$$

$$\frac{\partial L\left(\mathbf{w},\, b,\, \boldsymbol{\xi},\, \boldsymbol{\alpha},\, \boldsymbol{\beta}\right)}{\partial b} = 0, \tag{2.31}$$

$$\frac{\partial L\left(\mathbf{w},\, b,\, \boldsymbol{\xi},\, \boldsymbol{\alpha},\, \boldsymbol{\beta}\right)}{\partial \boldsymbol{\xi}} = \mathbf{0}. \tag{2.32}$$

$$\alpha_i\left(y_i\left(\mathbf{w}^\top\mathbf{x}_i + b\right) - 1 + \xi_i\right) = 0 \quad \text{for} \quad i = 1, \dots, M, \tag{2.33}$$

$$y_i\left(\mathbf{w}^\top\mathbf{x}_i + b\right) \geq 1 - \xi_i \quad \text{for} \quad i = 1, \dots, M, \tag{2.34}$$

$$\beta_i\xi_i = 0 \quad \text{for} \quad i = 1, \dots, M, \tag{2.35}$$

$$\alpha_i \geq 0,\ \beta_i \geq 0,\ \xi_i \geq 0 \quad \text{for} \quad i = 1, \dots, M. \tag{2.36}$$

Using (2.29), we reduce (2.30) to (2.32), respectively, to

$$\mathbf{w} = \sum_{i=1}^{M}\alpha_i y_i \mathbf{x}_i, \tag{2.37}$$

$$\sum_{i=1}^{M}\alpha_i y_i = 0, \tag{2.38}$$

$$\alpha_i + \beta_i = C \quad \text{for} \ i = 1, \dots, M. \tag{2.39}$$

Thus substituting (2.37) to (2.39) into (2.29), we obtain the following dual problem:

$$\min \sum_{i=1}^{M} \alpha_i \quad - \quad \frac{1}{2} \sum_{i,\, j=1}^{M} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^\top \mathbf{x}_j \tag{2.40}$$

$$\text{s.t.} \sum_{i=1}^{M} \alpha_i y_i \quad = \quad 0 \quad \text{for} \ \ i = 1, \dots, M, \tag{2.41}$$

$$C \geq \alpha_i \quad \geq \quad 0 \quad \text{for} \ \ i = 1, \dots, M. \tag{2.42}$$

The only difference between soft-margin SVMs and hard-margin SVMs is the upper bound C on the Lagrange multipliers α_i. The inequality constraints in (2.42) are called box constraints.

Especially, (2.33) and (2.35) are called KKT complementarity conditions. From these, there are three cases for α_i:

1. $\alpha_i = 0$. Then $\xi_i = 0$. Thus, $\mathbf{x}_i$ is correctly classified.
2. $0 < \alpha_i < C$. Then $y_i \left(\mathbf{w}^\top \mathbf{x}_i + b\right) - 1 + \xi_i = 0$ and $\xi_i = 0$. Therefore, $y_i \left(\mathbf{w}^\top \mathbf{x}_i + b\right) = 1$ and $\mathbf{x}_i$ is a support vector. Especially, we call the support vector with $C > \alpha_i > 0$ an unbounded support vector.
3. $\alpha_i = C$. Then $y_i \left(\mathbf{w}^\top \mathbf{x}_i + b\right) - 1 + \xi_i = 0$ and $\xi_i \geq 0$. Thus $\mathbf{x}_i$ is a support vector. We call the support vector with $\alpha_i = C$ a bounded support vector. If $0 \leq \xi_i \leq 1$, $\mathbf{x}_i$ is correctly classified, and if $\xi_i \geq 1$, $\mathbf{x}_i$ is misclassified.

The decision function is the same as that of the hard-margin SVM and is given by

$$D(\mathbf{x}) = \sum_{i \in S} \alpha_i y_i \mathbf{x}_i^\top \mathbf{x} + b, \tag{2.43}$$

where S is the set of support vectors.

Then unknown data sample is classified into

$$\begin{cases} y = +1 & \text{if} \ \ D(\mathbf{x}) > 0, \\ y = -1 & \text{if} \ \ D(\mathbf{x}) < 0. \end{cases} \tag{2.44}$$

If $D(\mathbf{x}) = 0$, $\mathbf{x}$ is on the boundary and thus is unclassifiable. When there is no bounded support vector, the region $\{\mathbf{x} \,|\, -1 < D(\mathbf{x}) < 1\}$ is the generalization region, which is the same as the hard-margin SVM.

2.2 Kernel Trick

SVMs construct an optimal hyperplane with the largest margin in the input space. The limited classification power of linear models was highlighted in the 1960s by Minsky and Papert [20]. Using a linear model means having a hypothesis that data points are linearly or almost linearly separable. However, many real-world applications require more complex hypotheses than that of linear models. On the other hand, the optimal hyperplane is a linear combination of the input features. But labels usually cannot be separated by a simple linear combination of the input features. Consequently, more complex feature mappings are required to be exploited.

One solution to this problem is projecting the data into a higher dimensional space (feature space) by a non-linear mapping to increase the classification power of the linear model as shown by Figure 2.3. This is equivalent to selecting a set of non-linear combinations of features and rewriting the data in the non-linear combinations. As a result, we can build the non-linear model in two steps: firstly, a non-linear mapping transforms the data into a feature space F, and then a linear model is used to do classification in the feature space. In this case, the decision function (2.43) is given by

$$D(\mathbf{x}) = \sum_{i \in S} \alpha_i y_i \langle \boldsymbol{\phi}(\mathbf{x}_i), \boldsymbol{\phi}(\mathbf{x}) \rangle + b, \tag{2.45}$$

where $\boldsymbol{\phi} : X \to F$ is a non-linear map from the input space to some feature space and $\langle \cdot, \cdot \rangle$ denotes the inner product.

Kernel

The kernel trick is a way of computing the inner product $\langle \boldsymbol{\phi}(\mathbf{x}_i), \boldsymbol{\phi}(\mathbf{x}) \rangle$ in the feature space directly as a function of the original input data points, making it possible to combine the two steps of building non-linear modes into one. The function is called a kernel function or a kernel.

Definition. A kernel is a function K, such that for all $\mathbf{x}, \mathbf{z} \in X$,

$$K(\mathbf{x}, \mathbf{z}) = \langle \boldsymbol{\phi}(\mathbf{x}), \boldsymbol{\phi}(\mathbf{z}) \rangle, \tag{2.46}$$

where $\boldsymbol{\phi}$ is a mapping from X to a (inner product) feature space F.

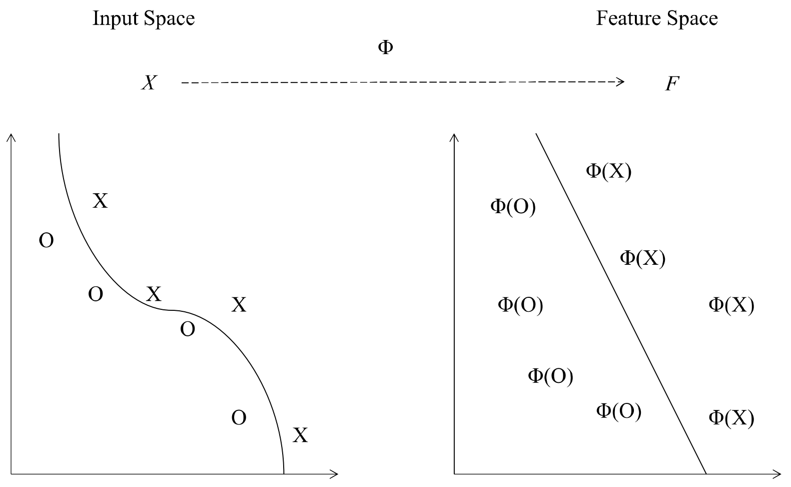

Figure 2.3: The idea of non-linear SVM: map the data points into a higher dimensional space via function Φ, then construct a hyperplane in feature space. This results in a nonlinear decision boundary in the input space. Adapted from [21].

Example. Radial basis function (RBF) kernel can be written as

$$K(\mathbf{x},\, \mathbf{z}) = e^{-\gamma||\mathbf{x}-\mathbf{z}||^2}, \tag{2.47}$$

where γ is known as the width scalar. In [23], Steinwart et al. argue that RBF kernel maps data points into a infinite-dimensional space. But when computing the inner product in this infinite-dimensional space by the RBF kernel, we don't need to know the information of the corresponding feature mapping ϕ.

Using the kernel function, the decision function (2.43) can be rewritten as

$$D(\mathbf{x}) = \sum_{i \in S} \alpha_i y_i K\left(\mathbf{x}_i,\, \mathbf{x}\right) + b. \tag{2.48}$$

Functions of Kernel

One important fact of the kernelized dual form (equation (2.48)) is that the dimension of feature space does not effect the computation. Because kernel functions make it possible to map the data implicitly into a feature and train the linear model in that space, one does not represent the feature vectors explicitly. The number of operations required to compute the inner product by evaluating kernel functions is not proportional to the number of dimensions of the feature space. For example, although the RBF kernel maps data points into a infinite-dimensional space, the number of operations required to evaluate the kernel function is not infinity. In this way, without knowing the underlying feature mapping, we can still compute inner product in the feature space by kernel functions and be able to learn the linear model in the feature space.

In addition to being regarded as a shortcut of computing inner product in the feature space, kernels can also be viewed as a function to measure similarity between two vectors.

Provided that two vectors $\mathbf{x}$ and $\mathbf{x}'$ are normalized to length 1, the inner product of two vectors can be interpreted as computing the cosine of the angle between these two vectors. The resulting cosine ranges from -1 to 1. A value of -1 means that two vectors are exactly opposite and 1 means that they are exactly the same. The zero value usually indicates independence of the two vectors and in-between values indicates intermediate similarity. Thus, the inner product of two normalized vectors can be viewed as a similarity of these two vectors.

Embedding data into a feature space provides the freedom to choose the mapping $\boldsymbol{\phi}$. The inner product (similarity) is not restricted to the input space, but also can be used in the feature space. Kernel functions offer the possibility of computing the mapping and inner product implicitly. At the same time, kernel functions can be viewed as a similarity of two vectors. Moreover, it enable us to design a large variety of similarity measures instead of figuring out an interpretable geometric space.

Chapter 3

Methodology

Now, we discuss two methodologies to improve the performance of SVMs on categorical data. Firstly, different data-driven similarity measures are used to capture the relationship between two samples, and used as the kernel matrix in SVMs. Unlike standard kernels like the RBF kernel that does not depend on test data, these data-driven similarities can use the information in both the training and test data. As a result, they can extract more information from the underlying distribution of the dataset. Then, in order to combine benefits of each similarity measure, an ensemble selection method is used to construct an ensemble from a library of SVM models with different similarity measures. The ensemble selection method allows models to be optimized to different performance evaluation methods.

3.1 Similarity Measures

The use of kernels is an attractive computational shortcut. Although it is possible to create a feature space and work out an inner product in that space, in practice, the approach taken is to define a kernel function directly. Consequently, both the computation of inner product and design of a feature space can be avoided.

There are several ways to design a kernel by defining the kernel function directly. In [21], Schlkopf et el. show that one can make a kernel by taking linear combination of other kernels. Another way of making a kernel comes from designing an inner product. However, all these methods share the same problem that the information from the test data is ignored. As a consequence, if the training data is biased, i.e. training samples cannot represent the population well, for example, when labels are unbalanced or training set is small, the prediction error can be large.

To solve this problem, we can design a new kind of kernels which can make full use of information in the whole dataset. Since a kernel function can be viewed as a similarity between two vectors, a more intuitive way of designing a kernel may be using a data-driven similarity measure. Such data-driven similarity measures take into account the frequency distribution of different feature values in a dataset to define a similarity measure between two categorical feature values. As a result, even though the training set may be unbalanced or small, when computing the frequency, these data-driven similarity based kernels can reduce the bias by involving information in both training and test data. In the literature, the notion of using information of both labeled (training) and unlabeled (test) data is called semi-supervised learning.

The following discussion follows the formulation of [3]. For the sake of notation, consider a categorical dataset D containing N samples and d features. Define A_k as the k^{th} feature and it takes n_k values in the dataset. Then we can define the following notation:

- $f_k(x)$: The number of times feature A_k takes the value x in the dataset D.
- $\hat{p}_k(x)$: The sample probability of feature A_k to take the value x in the dataset D. The sample probability is given by
$$\hat{p}_k(x) = \frac{f_k(x)}{N}. \tag{3.1}$$
- $p_k^2(x)$: Another probability estimation [12] of feature A_k to take the value x in the given dataset D is given by
$$p_k^2(x) = \frac{f_k(x)\,(f_k(x)-1)}{N(N-1)}. \tag{3.2}$$

Usually, the similarity measure assigns a similarity value between two samples X and Y belonging to the dataset D as follows:

$$S(X,\,Y) = \sum_{k=1}^{d} w_k S_k(X_k,\,Y_k), \tag{3.3}$$

where $S_k(X_k,\,Y_k)$ is the feature similarity between two values for one categorical feature A_k and X_k and Y_k are the k^{th} values of sample X and Y respectively. The quantity w_k denotes the weight assigned to the feature A_k.

Table 3.1 summarizes the mathematical formulas for the similarities used in this paper. Adapted from Table 2 in [3], Table 3.1 only lists similarity measures used in this work.

Measure	$S_k\left(X_k,\, Y_k\right)$	$w_k,\, k=1,\ldots,d$
Overlap	$= \begin{cases} 1 & \text{if } X_k = Y_k \\ 0 & \text{otherwise} \end{cases}$	$\frac{1}{d}$
IOF	$= \begin{cases} 1 & \text{if } X_k = Y_k \\ \frac{1}{1+\log f_k(X_k) \log f_k(Y_k)} & \text{otherwise} \end{cases}$	$\frac{1}{d}$
OF	$= \begin{cases} 1 & \text{if } X_k = Y_k \\ \frac{1}{1+\log \frac{N}{f_k(X_k)} \log \frac{N}{f_k(Y_k)}} & \text{otherwise} \end{cases}$	$\frac{1}{d}$
Lin	$= \begin{cases} 2\log \hat{p}_k(X_k) & \text{if } X_k = Y_k \\ 2\log(\hat{p}_k(X_k) + \hat{p}_k(Y_k)) & \text{otherwise} \end{cases}$	$\frac{1}{\sum_{i=1}^{d} \log \hat{p}_i(X_i) + \log \hat{p}_i(Y_i)}$
Goodall1	$= \begin{cases} 1 - \sum\limits_{q\in Q} p_k^2(q) & \text{if } X_k = Y_k \\ 0 & \text{otherwise} \end{cases}$	$\frac{1}{d}$
Goodall2	$= \begin{cases} 1 - \sum\limits_{q\in Q} p_k^2(q) & \text{if } X_k = Y_k \\ 0 & \text{otherwise} \end{cases}$	$\frac{1}{d}$
Goodall3	$= \begin{cases} 1 - p_k^2(X_k) & \text{if } X_k = Y_k \\ 0 & \text{otherwise} \end{cases}$	$\frac{1}{d}$
Goodall4	$= \begin{cases} p_k^2(X_k) & \text{if } X_k = Y_k \\ 0 & \text{otherwise} \end{cases}$	$\frac{1}{d}$

Table 3.1: Similarity Measure for Categorical Features. For measure Goodall1, $\{Q \subseteq A_k : \forall q \in Q,\, p_k(q) \leq p_k(X_k)\}$. For measure Goodall2, $\{Q \subseteq A_k : \forall q \in Q,\, p_k(q) \geq p_k(X_k)\}$. Adapted from [3].

Characteristics of similarity measures

According to the quantities being used, similarity measures listed in Table 3.1 may be classified into four groups.

Overlap is the simplest measure listed in the table. It is developed based on the idea of computing proportion of matching features in two samples. S_k is assigned to a value of 1 if the k^{th} feature of two samples are the same and a value of 0 if they are different. In addition, it equally weights matches and mismatches.

Originally used for the information retrieval in documents, *OF* and *IOF* involve the frequency of values in a feature. The main difference of these two measures is giving different similarities to mismatches. The *IOF* measure assigns lower similarity to mismatches on more frequent values and higher similarity to mismatches on less frequent values, while the *IF* measure assigns the opposite weights.

Lin is a special measure because it not only uses the probability estimation $\hat{p}_k$, but also follows the information-theoretic framework proposed by Lin in [18]. In terms of weights, Lin measure gives more weights to matches on frequent values and low weights to mismatches on infrequent values. Moreover, results in [18] show that Lin measure has an excellent performance on measuring similarity of categorical samples. As a result, in the experiments, the Lin measure will be used as the example of data-driven similarity measures.

Because of using another probability estimation $p_k^2(x)$, *Goodall1*, *Goodall2*, *Goodall3*, and *Goodall4* are clustered into one group. The first difference among these four measures is that *Goodall1* and *Goodall2* use the cumulative probability of other values in the feature either more or less frequent than than the current showing up one. Regardless of other values, *Goodall3* and *Goodall4* just involve the probability of the common values that two samples share. Secondly, in terms of weighting, *Goodall1* and *Goodall3* assigns higher similarity if the matching values are infrequent. While *Goodall2* and *Goodall4* assign the opposite similarity.

3.2 Ensemble Selection Method

An ensemble constructs a collection of models (base models) whose predictions are combined by weighted average or majority vote. Mathematically, it can be formulated as

$$F(y|x) = \sum_{m \in M} w_m f_m(y|x), \tag{3.4}$$

where M is the index set of base models which are chosen, w_m are the weights for the corresponding base models f_m and F is the final ensemble. A large number of research has shown that compared with the single model, ensemble methods can increases performance significantly (e.g. [5, 6, 11]). However, these usual ensemble methods either simply average all the base models without any selection, or collect a single kind of base model like tree model.

Ensemble selection [7] was proposed as an approach of building ensembles by selecting from large collection of diverse models. Compared with other ensemble methods, the ensemble selection method benefits from the ability of using many more base models and includes a selection process. Especially, ensemble selection's ability to optimize to any performance metric is an attractive capability of the method that is useful in domains which use non-traditional performance evaluation measures like AUC and AP [10].

The ensemble selection method as shown in Algorithm 3.1 is a two-step procedure. Before the procedure, the whole dataset is equally divided into three subsets. One subset is used for training base models; another one is for validation and the other one is for testing the ensemble performance. The two-step procedure begins by training models using as many type of models and tuning parameters as can be applied to the problem. Little attempt is made to optimize the performance of single models in the first step; all models no matter what their performance, are candidates in the model library for further selection. It is then followed by selecting a subset of models from the model library that yield the best performance on the performance metric. The selecting step can be done by using a forward stepwise model selection, i.e., at each step selecting the base model in the library that maximizes the performance of the ensemble if added. The performance of adding a potential model to the ensemble is evaluated by combining the prediction of selected base models and the potential model on the validation set.

Preventing overfitting

The forward stepwise selection may encounter a problem that the ensemble performs much better on the training set than on the test set, namely overfitting, in the selection procedure. The reason of overfitting in the ensemble selection is that by greedily selecting the best base model at each step, the ensemble yields a extremely good performance on the validation set. Under this circumstance, the selection stops when the ensemble is small.

Caruana et al. suggest the sorted ensemble initialization approach [7] to address the overfitting problem. Specifically, instead of starting with an empty ensemble, sort the models in the library by their performance on the validation set, and put the best N

Algorithm 3.1 Ensemble Selection Algorithm

```
1. Start with the empty ensemble.
2. Repeat, until no model can improve the ensemble's performance:
    (a) Choose the model $i$ in the library that maximizes the ensemble performance to
        the metric after being added.
    (b) Set
            $M \leftarrow M \cup i,$
            $F(y|x) = \sum_{m \in M} \frac{1}{|M|} f_m(y|x).$
    Break
3. Output an ensemble selection model
            $F(y|x) = \sum_{m \in M} \frac{1}{|M|} f_m(y|x).$   (3.5)
```

models in the ensemble. N is chosen by the size of model library. As a result, the ensemble will not contain only those models that overfit the training set.

3.3 Approach

Based on the discussion of similarity measures and ensemble selection method, we propose two approaches to improve the performance SVM model on the categorical data.

Firstly, in order to deal with categorical data naturally, we use the data-driven similarity measures listed in Table 3.1 as the kernel function in SVM models. Since data-driven similarity measures use information from both training and test set, this approach can give a better estimation of the similarity between samples.

Secondly, in order to combine benefits of those similarity measures, we use ensemble selection technique to build up an ensemble of SVM models with different similarity measures. In doing so, we take advantage of data-driven similarity measures and the ensemble method.

Chapter 4

Numerical Results

In this chapter, we present numerical results of the two approaches discussed in Chapter 3. These two methods are applied to the dataset from KDD Cup 2009[1], provided by a French Telecom company. The task of the competition is to predict the customer behavior from the customer data. The competition has two datasets and each contains three labels — churn (switch providers), appetency (buy new products or services) and up-selling (buy upgrades or add-ons), which indicate three different behavior of customers. In our experiments, we focus on the small dataset with the label up-selling, which is also called slow challenge in the competition.

The small dataset consists of 50000 samples and 230 features, out of which 40 are categorical. Almost all the features, including 190 continuous features have missing values. There is no description of what each feature means. Table 4.1 presents the frequency of three labels, from which we can observe that the label is highly unbalanced. In our experiments, the model performance evaluation is scored based on Accuracy, Area Under the ROC curve (AUC), and Average Precision (AP) [10] by a 3-fold cross-validation approach. In n-fold cross-validation, first the training set is divided into n subsets of equal size. Sequentially, one set is tested using the model trained on the remaining $n-1$ subsets. As a result, each sample of the whole dataset is trained and predicted once and the cross-validation performance measure is the average performance on the n subsets.

[1]http://www.kdd.org/kdd-cup-2009-customer-relationship-prediction

Label	Frequency of -1 : 1	Ratio
churn	46328 : 3672	12 : 1
appetency	49110 : 899	55 : 1
up-selling	46318 : 3682	12 : 1

Table 4.1: Frequency of labels of churn, appetency and up-selling

4.1 Data Preprocessing and Cleaning

Since the raw dataset posts several challenges, e.g., many missing values, a mixture of categorical and continuous features and categorical features with a large number of possible values, before experiments, we need to preprocess the raw data.

Missing Values

There are about 66% values missing in the whole dataset, which is a problem for SVM models. After two empty categorical features are deleted, missing categorical values are less of a problem since they can be treated as a standalone value in each feature. But missing values for continuous data are more concerning. In terms of the high missing rate, first we delete features with a missing rate above 95%, which contains very little information for prediction. After this step, 43 out of 190 continuous features remain and the missing rate is reduced to 15%. Then we follow a standard approach of imputing missing values by its median of the corresponding feature.

Discretization

During data cleaning process, one important observation is that many continuous features are more like "categorical" features since they contain only a limited number of discrete values. In addition, similarity measures are suitable only for categorical features. Inspired by the observation, we encode the 20 most common values for each continuous feature and record all other values as a separate value. After this step, all the continuous features become categorical features. Another benefit is that outliners are smoothed out implicitly during the process. Note that since the RBF kernel for SVMs can deal with continuous features, the process is only done for similarity measure based kernel SVMs.

Another problem for SVMs with the RBF kernel is that it cannot handle categorical features directly. Moreover, there are a large amount of possible values for some categorical features and many values just appear once in some features. The way of encoding a categorical feature for RBF kernels is generating indicator variables for each different values that the feature can take. In order to avoid an explosion in the number of features in SVMs with the RBF kernel and better estimate the frequency of each possible value for similarity measures, rather than all the values, we limit the encoded number of possible values to 20 most common values in each feature.

4.2 Feature Selection

After preprocessing and cleaning, we are left with 121 features which is still a large number. Among these features, there may be some features which contain very little information for prediction. This kind of features which are irrelevant for prediction usually are called redundant features. Moreover, these redundant features may not only cost computational time but also increase prediction error. Before training SVM models, it is necessary to remove redundant features by feature selection.

A naturally way of feature selection is giving each feature a score to indicate its importance. In the literature, tree method [4] is proposed for this usage. But [14] points out that tree method is unstable: small variations in the training set can result in large different trees and predictions for the same text set.

In [6], Breiman proposed the random forests approach to solve the classification problem and also measure the importance of features. Random forests are a collection of tree models such that each tree depends on the values of random vectors sampled independently from dataset. For each tree in the forests, random forests adopt a bootstrap sample from the set of samples and a random subset of features from the feature set. By adding the randomness, random forests reduce the correlation between individual trees and thus reduce the variance of prediction. At the same time, the tree construction process can be considered as a type of variable selection and the information measure [4] (usually accuracy or Gini index) reduction due to the fact that a split on a specific feature could indicate the relative importance of the variable in the tree model. By averaging trees in the forests, random forests stabilize the feature importance of a single tree, which leads to a better prediction performance and feature importance score. In this paper, we will use the feature importance score generated by random forests to select features and compare the performance of our SVM models with this leading approach.

In order to compute the feature importance scores, the random forest is trained on the whole dataset. To save space, Table 4.2 lists the top 5 scores of features under two

Feature	MeanDecreaseAccuracy		Feature	MeanDecreaseGini
Var126	91.00		Var126	5220
Var28	26.37		Var28	1090
Var226	11.07		Var211	900
Var210	10.60		Var226	474
Var218	10.18		Var206	239

Table 4.2: Top 5 feature importance scorse from random forests

different information measures. First and third columns are names of features and second and fourth columns are scores of using accuracy and Gini index. Table 4.2 also shows that these two rankings share 4 out of 5 features. Thus the rankings of features are consistent under two information measures.

In our experiments, we select top 40 features under each information measure and take union of two selected sets to obtain 51 features out of 121 features.

4.3 Results of SVMs with the RBF kernel

As suggested by Hsu et el. in [16], in general, the RBF kernel is a reasonable first choice and it nonlinearly maps samples into a higher dimensional space. Thus, our experiments start with the most frequently used kernel, RBF kernel, for SVM (we refer to it as RBF kernel SVM), whose performance will be considered as the benchmark of subsequent experiments.

The RBF kernel takes the form

$$K(\mathbf{x}_i, \mathbf{x}_j) = e^{-\gamma||\mathbf{x}_i - \mathbf{x}_j||^2}, \tag{4.1}$$

where x_i and x_j are two sample vectors. In the categorical data, they are indicator variables encoded from the original data. Hence, in a SVM using RBF kernel, the margin parameter C in the SVM model and the scale parameter γ in the RBF kernel, are two tuning parameters to be decided. Consequently, some model selection (parameter search) must be done to identify good (C, γ) so that the model can have a good performance. This process can be done by a "grid-search" on C and γ using cross-validation. In [16], an exponentially growing sequence (for example, $\gamma = 2^{-10}, 2^{-8}, \ldots, 2^{3}$) of tuning parameters is recommended as a practical method. Also, Zhu (2008 [24]) discusses the SVM model's sensitivity to its tuning parameters and points out that the performance of SVMs using RBF kernel functions is very sensitive to the parameter γ while not as sensitive to the

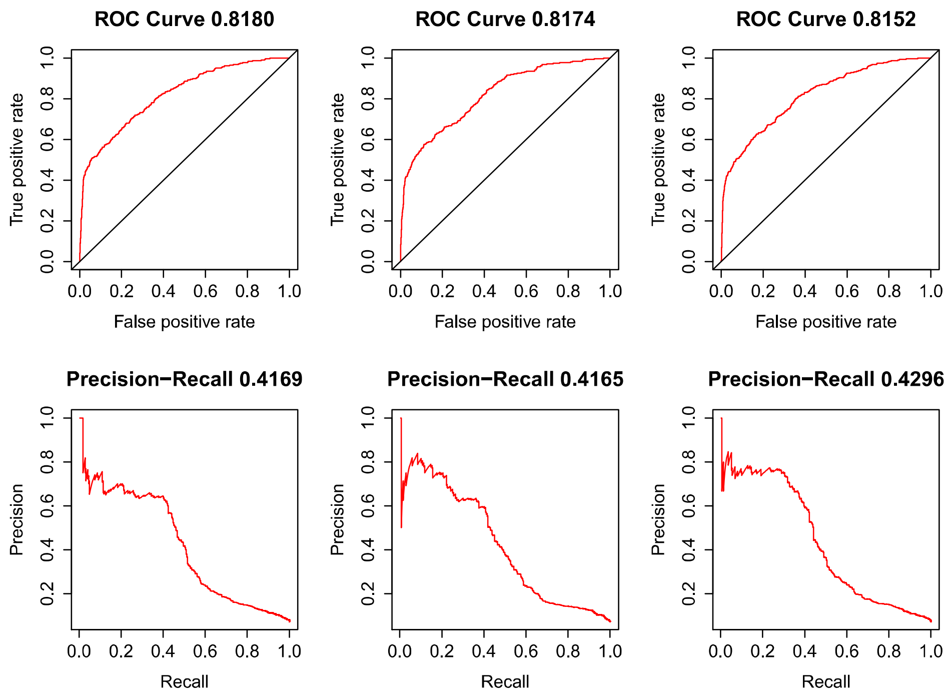

Figure 4.1: ROC and Precision-Recall Curves of SVMs with Lin Similarity Measure

parameter C. In order to save computational time, we may use a fine grid-search on γ while a coarse grid-search on C.

The results of grid-search on (C, γ) as well as the cross-validation performance are displayed in Table 4.3. Note that TotalAccuracy refers to the average accuracy rate of the cross-validation , PosAccuracy is the average accuracy rate of the cross-validation on the class with label 1, the bold font row indicates the best performance in terms of AUC and red entries are the best performance of each measure.

4.4 Results of SVMs with the Lin Similarity Measure

Results (Table 4.4 and Figure 4.1) in this section demonstrate the performance of SVMs with the Lin similarity measure. Parameters of grid search and numerical results of cross-validation can be found in Table 4.4. The top row of Figure 4.1 displays 3 plots: ROC curves of cross-validation on 3 folds. The bottom row displays Precision-Recall curves of

AUC	TotalAccuracy	PosAccuracy	AP	C	γ
0.6646	0.9262	0.0022	0.1425	1	2^{0}
0.7223	0.9301	0.1022	0.2590	1	2^{-2}
0.7552	0.9338	0.2011	0.3457	1	2^{-4}
0.7737	0.9344	0.2293	0.3571	1	2^{-6}
0.7772	0.9268	0.0838	0.3008	1	2^{-8}
0.7782	0.9230	0.0098	0.2959	1	2^{-9}
0.7793	**0.9247**	**0.0054**	**0.2889**	**1**	$\mathbf{2^{-10}}$
0.7751	0.9257	0.0011	0.2823	1	2^{-11}
0.6640	0.9262	0.0022	0.1423	0.1	2^{0}
0.7221	0.9298	0.0935	0.2579	0.1	2^{-2}
0.7549	0.9334	0.2011	0.3443	0.1	2^{-4}
0.7737	0.9344	0.2250	0.3572	0.1	2^{-6}
0.7741	0.9254	0.0577	0.2956	0.1	2^{-8}
0.7759	0.9234	0.0218	0.2853	0.1	2^{-9}
0.7709	0.9250	0.0044	0.2562	0.1	2^{-10}
0.7668	0.9256	0.0022	0.2227	0.1	2^{-11}
0.6583	0.9262	0.0022	0.1379	0.01	2^{0}
0.7207	0.9299	0.0924	0.2552	0.01	2^{-2}
0.7550	0.9338	0.2087	0.3449	0.01	2^{-4}
0.7726	0.9340	0.2283	0.3577	0.01	2^{-6}
0.7705	0.9250	0.0294	0.2747	0.01	2^{-8}
0.7624	0.9249	0.0011	0.2310	0.01	2^{-9}
0.7584	0.9256	0.0000	0.2018	0.01	2^{-10}
0.7497	0.9261	0.0000	0.2050	0.01	2^{-11}

Table 4.3: Performance of RBF kernel SVMs. The bold font row indicates the best performance in terms of AUC and red entries are the best performance under each measure.

AUC	TotalAccuracy	PosAccuracy	AP	C
0.4999	0.9264	0	-	2^{-7}
0.5000	0.9264	0	-	$2^{-7.5}$
0.8057	0.9405	0.4011	0.4213	2^{-8}
0.8042	0.9405	0.3924	0.4181	$2^{-8.5}$
0.8067	0.9397	0.3772	0.4155	2^{-9}
0.8079	0.9398	0.3609	0.4223	$2^{-9.5}$
0.8119	0.9393	0.3370	0.4180	2^{-10}
0.8169	**0.9370**	**0.2848**	**0.4210**	$\mathbf{2^{-10.5}}$
0.8141	0.9357	0.2468	0.4139	2^{-11}
0.8124	0.9350	0.2152	0.4109	$2^{-11.5}$
0.8039	0.9310	0.1121	0.3944	2^{-12}

Table 4.4: Performance of SVMs with Lin similarity measure. The bold font row indicates the best performance in terms of AUC and red entries are the best performance under each measure.

3 folds. And the title of each plot gives information of AUC or AP based on the curve.

In the table, AUC is the area under the ROC curve shown in Figure 4.1. Since it is a portion of the area of the unit square, its value is always between 0 and 1. A random guessing produces the diagonal line between (0, 0) and (1, 1) on the ROC curve, which gives 0.5, the worst AUC. A value of 1 gives the perfect AUC. AP is the area under the precision-recall curve shown in the second row of Figure 4.1. Although the best AP is 1 and the worst is 0, in practice, it takes value between 0.2 to 0.5.

In an ROC curve, the Y axis is the true positive rate which can be viewed as the benefit and X axis is the false positive rate which can be viewed as the cost of a classification model. We prefer a higher benefit with lower cost, which means a higher ROC curve when the false positive rate is low. For the precision-recall curve, we expect the curve dropping from 1 to 0 as slow as possible to cover a larger area bounded by the X axis and the curve.

4.5 Results of the Ensemble Selection Method

Unlike single SVM models, which can only be optimized to one performance evaluation measure, the ensemble selection method is allowed to be optimized to arbitrary performance metrics. Similarity measures listed in Table 3.1 with parameters in Table 4.4 are used to train the base SVM models of the model library. Then the ensemble is built via the

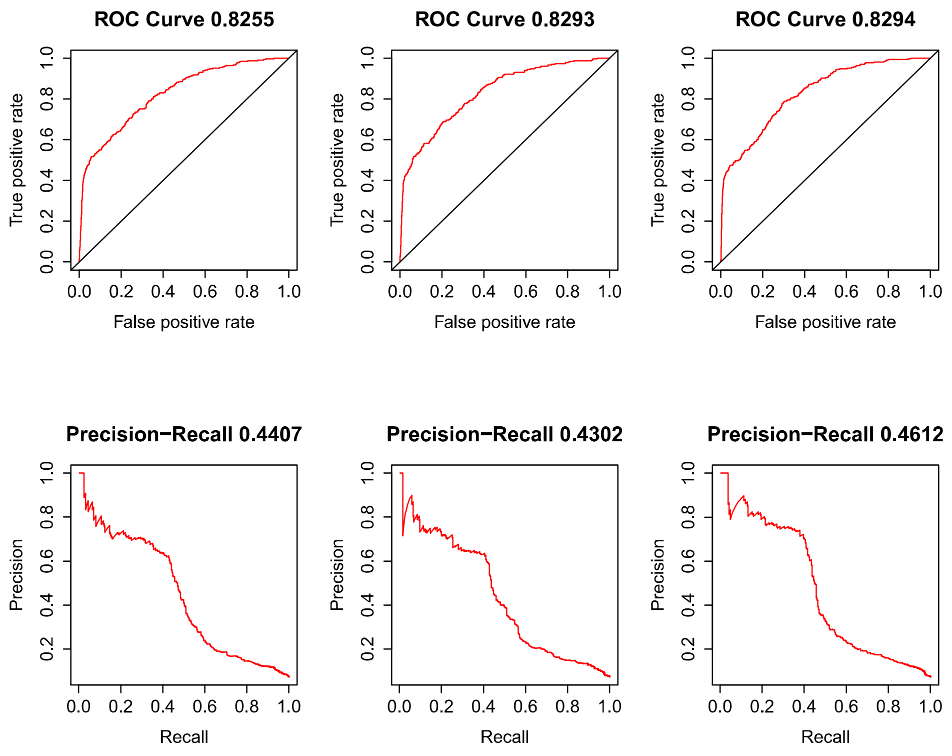

Figure 4.2: ROC and Precision-Recall Curves of Ensemble Selection Method

Ensemble Selection on AUC					
	AUC	TotalAccuracy	PosAccuracy	Precision	#model
Fold1	0.82552	0.8977	0.5229	0.4132	5
Fold2	0.8293	0.7798	0.6895	0.4186	8
Fold3	0.8294	0.7186	0.7565	0.4388	7
Avg	**0.8281**	0.7987	0.6563	0.4235	6.7
Ensemble Selection on Precision					
	AUC	TotalAccuracy	PosAccuracy	Precision	#model
Fold1	0.8185	0.9013	0.5098	0.4407	5
Fold2	0.8229	0.5460	0.8954	0.4302	10
Fold3	0.8241	0.6380	0.8247	0.4613	7
Avg	0.8218	0.6951	0.7433	**0.4440**	7.3

Table 4.5: Performance of the Ensemble Selection Method. The bold font row indicates the best performance in terms of AUC.

ensemble selection method described in Section 3.2. During the ensemble selection process, the 3 fold cross-validation is adopted. The difference between the 3 fold cross-validation used here and the previous description is that instead of using one for training and the rest for test, here one fold is used for training base models; another one is used as a validation set for selection and the other one is for test. The overall performance is computed by averaging the performance on all three folds.

Table 4.5 gives the information of optimizing the ensemble selection method on AUC and precision, respectively. The column under #model in the table indicates the number of models in each ensemble on each fold and Avg gives the average performance over 3 folds. Again, bold entries show the best average performance in terms of AUC and Precision.

Figure 4.2 shows ROC curves and Precision-Recall curves of two ensemble selection methods. Top 3 plots display ROC curves on 3 folds of the ensemble selection method with AUC optimized over AUC. And Bottom 3 plots display Precision-Recall curves on 3 folds of the ensemble selection method with AP optimized.

4.6 Comparison of Three Methods

In this section, we compare SVMs with the Lin similarity and the ensemble selection method with standard SVM models with the RBF kernel. Table 4.6 summaries the best performance of all methods on each performance measure.

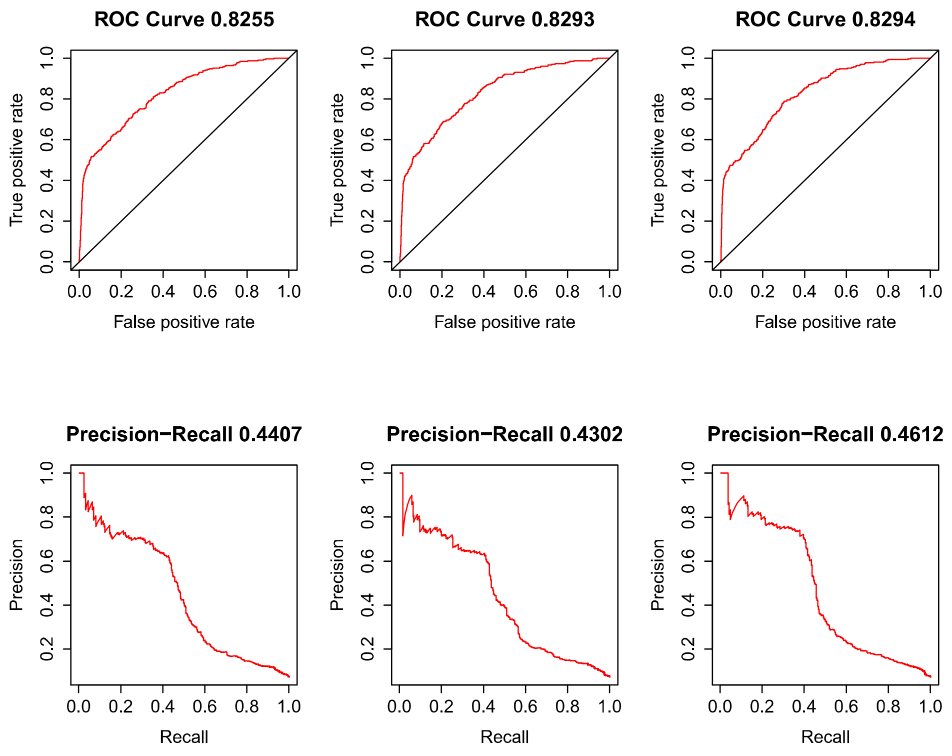

Figure 4.3: ROC and Precision-Recall Curves of the Ensemble Selection Method

Method	AUC	TotalAccuracy	PosAccuracy	AP
RBF kernel	0.7793	0.9338	0.2293	0.3571
Lin kernel	0.8169	0.9405	0.4021	0.4222
EnsembleAUC	0.8281	0.7987	0.6563	0.4235
EnsembleAP	0.8218	0.6951	0.7433	0.44402
RandomForest	0.8380	0.8272	0.6022	0.4303

Table 4.6: Best performance of RBF kernel, Lin similarity measure kernel SVMs, ensemble selection method and random forest

As mentioned before, the performance of the RBF kernel may be considered as benchmark of the performance of kernels. Compared with the RBF kernel, the kernel with Lin similarity measure yields a better performance on all performance measures listed in Table 4.3 – 4.6. Especially, keeping the same total accuracy level, the kernel with Lin similarity measure dramatically improves accuracy (75%) on the minority class and thus performs well on both AUC and AP.

With the benefit of optimizing performance to arbitrary performance metric, ensemble methods not only consistently outperform the SVM with RBF kernel on AUC and AP, but also significantly improve accuracy on the minority class even compared with the Lin similarity kernel SVM. In addition, because of a diverse collection of similarity measures, the ensemble selection approach performs slightly better than random forests in terms of accuracy of the minority class.

Chapter 5

Conclusion

This paper has presented a novel use of kernels based on data-driven similarity measures to dealing with categorical features for SVMs. In order to combine benefits of similarity measures, we have applied the ensemble selection method to build up an ensemble to optimize over several performance metrics.

Unlike standard kernels like RBF, when computing similarity measure between samples, our kernel is more closed to a semi-supervised approach that involves the information from both labeled and unlabeled data. The ensemble selection method uses the forward stepwise approach to select base models for the metrics that we wish to optimize.

Compared with the benchmark — RBF kernel, the results have shown that our approaches are able to improve the performance on all the metrics we used. Especially, keeping other performance measures at the same high level, ensemble selection method can significantly increase prediction accuracy on the minority class which is much more difficult to be predicted when classes are unbalanced.

Possible extension includes developing new similarity measures and introducing different weight to each selected feature.

References

[1] Shigeo Abe. *Support vector machines for pattern classification.* Springer, 2010.

[2] Michael J Berry and Gordon S Linoff. *Data mining techniques: for marketing, sales, and customer relationship management.* Wiley. com, 2004.

[3] Shyam Boriah, Varun Chandola, and Vipin Kumar. Similarity measures for categorical data: A comparative evaluation. *red*, 30(2):3, 2008.

[4] Leo Breiman. *Classification and regression trees.* CRC press, 1993.

[5] Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

[6] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[7] Rich Caruana, Alexandru Niculescu-Mizil, Geoff Crew, and Alex Ksikes. Ensemble selection from libraries of models. In *Proceedings of the twenty-first international conference on Machine learning*, page 18. ACM, 2004.

[8] Chih-Chung Chang and Chih-Jen Lin. Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27, 2011.

[9] Sheng Chen, CFN Cowan, and PM Grant. Orthogonal least squares learning algorithm for radial basis function networks. *Neural Networks, IEEE Transactions on*, 2(2):302–309, 1991.

[10] Tom Fawcett. An introduction to roc analysis. *Pattern recognition letters*, 27(8):861–874, 2006.

[11] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). *The annals of statistics*, 28(2):337–407, 2000.

[12] David W Goodall. A new similarity index based on probability. *Biometrics*, pages 882–907, 1966.

[13] Jiawei Han, Micheline Kamber, and Jian Pei. *Data mining: concepts and techniques.* Morgan kaufmann, 2006.

[14] Trevor Hastie, Robert Tibshirani, Jerome Friedman, and James Franklin. The elements of statistical learning: data mining, inference and prediction. *The Mathematical Intelligencer*, 27(2):83–85, 2005.

[15] Douglas M Hawkins. The problem of overfitting. *Journal of chemical information and computer sciences*, 44(1):1–12, 2004.

[16] Chih-Wei Hsu, Chih-Chung Chang, Chih-Jen Lin, et al. A practical guide to support vector classification, 2003.

[17] Risi Imre Kondor and John Lafferty. Diffusion kernels on graphs and other discrete input spaces. In *ICML*, volume 2, pages 315–322, 2002.

[18] Dekang Lin. An information-theoretic definition of similarity. In *ICML*, volume 98, pages 296–304, 1998.

[19] Huma Lodhi, Craig Saunders, John Shawe-Taylor, Nello Cristianini, and Chris Watkins. Text classification using string kernels. *The Journal of Machine Learning Research*, 2:419–444, 2002.

[20] Marvin Minsky and Papert Seymour. Perceptrons. 1969.

[21] Bernhard Schlkopf and Alexander J Smola. Learning with kernels: support vector machines, regularization, optimization, and beyond. *Cambridge: TheMITPress*, 2002.

[22] Bernhard Schölkopf, Christopher JC Burges, and Alexander J Smola. *Advances in kernel methods: support vector learning.* The MIT press, 1999.

[23] Ingo Steinwart, Don Hush, and Clint Scovel. An explicit description of the reproducing kernel hilbert spaces of gaussian rbf kernels. *Information Theory, IEEE Transactions on*, 52(10):4635–4643, 2006.

[24] Mu Zhu. Kernels and ensembles. *The American Statistician*, 62(2), 2008.