Dropout-based Support Vector Regularization

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Bachelor’s thesis

Degree Programme in Automation Engineering

Valkeakoski, Autumn 2016

Dat Tran Thanh

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Degree Programme in Automation Engineering
Valkeakoski

Author Dat Tran Thanh **Year** 2016

Subject **Dropout-based Support Vector Regularization**

ABSTRACT

In this thesis we consider a new regularization technique that exploits the probabilistic Dropout scheme at the sample level. The new regularization approach is incorporated into the Maximum Margin Classification (MMC) framework resulting in a new variant of the Support Vector Machine classifier. We show here that the added regularizer comes with a geometrical interpretation related to the selection of support vectors. In addition, we illustrate that the new formulation is consistent with the guarantee provided in the Statistical Learning Theory. Experimental results from several classification problems show better generalization performance achieved by adding the new regularization as compared to the standard approach.

Keywords Support Vector Machine, Regularization, Dropout, Kernel Methods.

Pages 28 pages

CONTENTS

1 INTRODUCTION 1

2 BASIC CONCEPTS 3

2.1 Learning from Examples 3
2.2 Linear models 6
2.3 Kernel trick 12

3 RELATED WORK 14

3.1 Dropout-SVM 15

4 PROPOSED METHOD 16

4.1 The proposed DropSVM classifier 17
4.2 Discussion 19

5 EXPERIMENTS 22

5.1 Experiment settings 22
5.2 Experimental Result 24

6 CONCLUSION 25

REFERENCES 27

ACKNOWLEDGEMENT

I would like to express my utmost gratitude towards Dr. Alexandros Iosifidis who is currently working at Signal Processing Department in Tampere University Of Technology. Under his supervision, I was brought closer to the research-related activities and more thorough understanding of many state-of-the-art results in machine learning. In addition, I am also thankful for Mr. Raine Lehto and Mrs. Niina Valtaranta for their fruitful feedback.

Finally, I am forever grateful to my parents, who have constantly supported me not only in this thesis work but also throughout the whole academic pursuit in Finland.

Tampere, January 2017

Dat Tran Thanh

1 INTRODUCTION

This thesis discusses machine learning, particularly, the regularization techniques in linear models. Machine learning is a subcategory of Artificial Intelligence (AI) where the objective is to solve complex problems by learning from examples. Machine learning aims at modelling the relationships between objects through observations rather by constructing an algorithm for each specific problem. Many successful products in the industry were created through the contribution of a machine learning solution, e.g. object recognition, autonomous navigation, recommender system, stock prediction.

Regularization is an important aspect in many machine learning models. Since regularization has direct bearings in the generalization performance of a model, this field of research has continuously attracted several work. The Dropout technique was originally proposed by Hinton (2012) for training neural networks as a mean of regularization. Due to the large number of parameters in neural networks, such learning models have a strong capability to fit the data well, which in turn leads to the problem of overfitting on the training data. The main idea of Dropout is to randomly drop a subset of hidden units during updating the parameters of the network on each back-propagation iteration. Such process has been shown to prevent the parameters of the network from over co-adaptation. The same idea has found its resemblance in linear models as part of the so-called feature noising methods. By augmenting the finite training data with its artificially corrupted versions, according to a specific distribution, the linear models are expected to be less susceptible to noise and hence, increase performance over an unseen test sample.

The feature noising framework in linear models consists of two steps: complementing the original data by its infinite noisy versions and minimizing the average or the expectation of the corresponding loss function of the model under the given corrupting distribution. One way to create corrupted data is to randomly omit some of the dimensions of the original data, hence the Dropout procedure is considered as part of the feature noising framework. There are several works existing in the literature regarding different types of loss functions, e.g. van der Maaten (2013), Wager (2013), Wang (2013). While expectation of quadratic loss and exponential loss under corrupting distribution can be computed analytically (van der Maaten, 2013), in case of logistic loss or log-loss in generalized linear model, the expectation is only approximated through Taylor series expansion (van der Maaten, 2013; Wager, 2013).

For the Support Vector Machine (SVM) model, the non-differentiability of hinge loss hinders the calculation of the loss function as well. Under the data augmentation framework, previous attempts to regularize the SVM

predictor includes explicit corruption of training data as a mean of creating new data such as the work of Burges and Schölkopf (1997) or analysing the worst-case scenarios of feature deletion in the test set (Dekel, 2008; Globerson and Roweis, 2006; Smola, 2008). In these proposals, high computational cost is induced for explicit corruption or the setting is unlikely to take place in a practical situation in the analysis of worst-case scenarios. The direct implementation of Dropout training technique in SVM predictor was proposed by Chen (2014) in which a variational upper bound of the expected hinge loss was derived.

In general, previous work employing Dropout either in a neural network or under feature noising framework aim at limiting the co-operation between data dimensions, which can be understood as feature regularization. That is, all the above-described statistical methods exploit the Dropout approach in terms of regularizing the training data dimensions. Inspired by the Dropout idea, in this paper, we propose a novel approach that can be used in regularizing non-linear models (and in particular kernel-based models) by exploiting Dropout at the sample level. We use as a special case the widely used Maximum Margin Classification framework and in particular the SVM classifier. We will show that the motivation to regularizing its model by randomly dropping some of the training samples is highly intuitive. In addition, the intuition is justified by experimental results on standard classification problems. The contributions of this thesis are as follows:

- A novel formulation exploiting the sample Dropout procedure for MMC that is able to regularize the maximum margin decision boundary. The formulation results in an elegant solution which in turn allows the utilization of existing efficient SVM solvers without any modification.

- Qualitative discussion on the effect of the proposed regularization framework in the case of SVM training.

The thesis is organized as follows. Chapter 2 presents some preliminaries in machine learning. The related work of Chen (2014) with the implicit Dropout training for SVM is presented in Chapter 3. In Chapter 4, we describe the new formulation of SVM that constrains the decision boundary through Dropout, additionally we present the interpretation of the proposed method geometrically as well as under the regularization theory. The performance of our formulation is compared against that of the standard SVM classifier in publicly available classification problems in Chapter 5. In Chapter 6, conclusions are drawn from previous chapters.

2 BASIC CONCEPTS

This chapter introduces the basic tools and notations in machine learning as well as an important result from the Statistical Learning Theory. The existing literature in the field that relates to the proposed method in the later chapters is reviewed.

2.1 Learning from Examples

The paradigm in machine learning that we discuss here is called *Learning from Examples*. The main idea is that given some observations of the relations between objects, can we build the learner that could infer the general relationship between the objects by examining the observations? The learning problem is hence an inductive inference, in which incomplete information of a phenomenon is used to model the generating rule behind the phenomenon. Mathematically, these rules are described by functions which represent a mapping from input observations to output observations. Based on the type of the output observations, the learning task can be categorized as classification or regression. As the name suggests, classification is the task of classifying input objects into one of the finitely many classes, while regression refers to the task of assigning a continuous value to each of the inputs. In either case, the learning example has the form of an input-output pair. The aim of the learning process is to use the learned function to make inference of the output from an unseen input, i.e. the input which is not used during the learning process. Therefore, the effectiveness of the learning task is measured based on the inference made from unseen observations, the closer the inferred output to the true output, the more effective the learned function.

From the above presented mathematical point of view, machine learning is deemed largely as function fitting. It is interesting to note that not all functions that describe perfectly the mapping of the finite learning examples truly model the underlying rules that regulate the phenomenon. However, mathematical formulation allows the utilization of existing tools and methods to analyse the conditions under which the learning task is effective. The field of statistical learning theory investigates and gives insights into the learning problems that can be used to derive a successful learning algorithms. Moreover, from these insights, guarantees of the effectiveness of certain learning algorithms can be made.

To be concrete, we introduce some notions and concepts. Denote the known observations or the training data as the set $\mathcal{Z} = \{(x_i, y_i) \in \mathcal{X} \times \mathcal{Y} | i = 1, \dots, N\}$. As mentioned previously, if $\mathcal{Y} = \mathbb{R}$ we call the learning problem regression while if $\mathcal{Y}$ is the finite set, e.g. $\mathcal{Y} = \{-1,1\}$, we call the learning problem classification, specifically in case of two-class classification problem, it is also called binary classification problem.

The problem of learning hence aims at finding the function $f: \mathcal{X} \rightarrow \mathcal{Y}$ given only the set $\mathcal{Z}$ with the requirement that given any $x_{test} \in \mathcal{X}$, f is able to predict $y_{predict} = f(x_{test})$ that is as close as possible to y_{test} which is the true value of the pair (x_{test}, y_{test}). In order to find f, certain assumptions must be made. Under the statistical learning theory, it is assumed that there exists an unknown probability distribution $P(X, Y)$ defined over $\mathcal{X} \times \mathcal{Y}$ that truly describes the generating process of the observations. In addition, a loss function is needed in order to measure the amount of loss or penalty associated with the choice of f:

$$\ell(f(\boldsymbol{x}), y) \tag{1}$$

$\ell(f(\boldsymbol{x}), y)$ is the quantity that expresses how much risk we incur when choosing f. A straight forward choice of f could be to assign $\ell(f(\boldsymbol{x}), y) = 1$ if the prediction is wrong and $\ell(f(\boldsymbol{x}), y) = 0$ if the prediction is correct.

The learning strategy is then to select f^* that results in the lowest overall risk, i.e. the expected risk:

$$f^* = \arg\min_f E_P\big(\ell(f(\boldsymbol{x}), y)\big) = \arg\min_f \mathcal{R}(f) \tag{2}$$

Where in (2), E_P denotes the expectation with respect to the joint distribution $P(X, Y)$. The function f^* is our ideal function and often called the target function. In practice, the joint probability distribution $P(X, Y)$ is unknown and only a part of it, the set $\mathcal{Z}$ is available. In order to build a learner from the limited amount of known data, an induction principle is needed to approximate the expected risk. The so-called Empirical Risk Minimization (ERM) induction principle was developed by Vapnik (1998) to exactly do this. The data set $\mathcal{Z}$ is used to build a stochastic approximation of the expected risk in (2), which is called empirical risk:

$$\mathcal{R}(f; \mathcal{Z}) = \frac{1}{N} \sum_{i=1}^{N} \ell(f(x_i), y_i) \tag{3}$$

Consequently, the learning strategy is to learn the function that minimizes the empirical risk in (3):

$$\tilde{f} = \arg\min_f \mathcal{R}(f; \mathcal{Z}) \tag{4}$$

Up until now, we have not discussed any assumption on the function space $\mathcal{F}$ in which we are searching f and how effective $\tilde{f}$ is in estimating y_{test} given x_{test}. Pure minimization of (4) can be problematic since we could simply build $\tilde{f}$ that "remembers" all pair values in $\mathcal{Z}$ which results in near zero empirical risk although not being able to make any prediction on x_{test}. The situation is referred to as overfitting in which the minimum of empirical risk is very small or but the expected risk is large. The quantity

that we are truly interested in, however, is the expected risk. In order to have a control or guarantee over the effectiveness of choosing f, Statistical Learning Theory studies the probabilistic bounds on the gap between the empirical and expected risk. The number of training examples N and the complexity of $\mathcal{F}$ measured by the so-called capacity h are involved in the bounds. Since there exists several methods to measure the capacity of the function space $\mathcal{F}$ such as covering numbers, annealed entropy, VC dimension (Vapnik and Chervonenkis, 1971) or the scale sensitive versions of it (Kearns and Shapire, 1994; Alon et al., 1993), the specific versions of the bounds depend on h. However, it could be generally described with the following form (Evgeniou, 2002): with probability of at least η:

$$\mathcal{R}(f) < \mathcal{R}(f; \mathcal{Z}) + \varphi\left(\sqrt{\frac{h}{N}}, \eta\right) \tag{5}$$

with φ is an increasing function of $\frac{h}{N}$ and η.

The result in (5) provides us a guarantee and a guiding principle in choosing the function space $\mathcal{F}$ in which we perform the empirical risk minimizer. We can see that if the capacity h is too large, the distance between $\mathcal{R}(f; \mathcal{Z})$ and $\mathcal{R}(f)$ can be very large, hence overfitting occurs. In order to prevent overfitting, one could consider reducing the complexity when the hypothesis space $\mathcal{F}$ is large. This can be done by incorporating the complexity of the hypothesis space $\mathcal{F}$ into the minimization problem in (4), finding $\tilde{f}$ with the best trade-off between empirical risk and complexity. With this insight, the learning problem is now modified again and becomes:

$$\min_{f \in \mathcal{F}} \mathcal{R}(f; \mathcal{Z}) + \lambda \Omega(f) \tag{6}$$

Where $\Omega(f)$ is the term that controls the complexity of the hypothesis space $\mathcal{F}$ and λ is a non-negative parameter that controls the trade-off between minimizing $\mathcal{R}(f; \mathcal{Z})$ and $\Omega(f)$.

The minimization problem in (6) can also be interpreted under the Regularization Theory in which $\Omega(f)$ is called the regularizer that imposes a certain constraint on the class of hypothesis function. The introduction of a regularizer into the empirical risk minimization can be explained by the fact that solely minimizing $\mathcal{R}(f; \mathcal{Z})$ in (4) can lead to an ill-posed problem. The inclusion of a regularizer in many cases ensures a well-posed problem.

Lately, we have presented the problem of learning from examples in the general form of the optimization problem in (6). In fact, optimization problem (6) is central in the supervised-learning setting. Based on different assumptions incorporated into the learning process, we have different classes of learning models built from (6). Specifically (6) presents us with

the choice of hypothesis space $\mathcal{F}$, the choice of loss function $\ell(f(x), y)$ and the choice of $\Omega(f)$ incorporated. The next section discusses some of these selections.

2.2 Linear models

The class of linear models assumes that there exists a linear relationship between the label y and the dimensions of input data x. Specifically, denote D the dimension of x, i.e. $x \in R^D$, the function that models the phenomenon between x and y is assumed to have the form:

$$y = f(x) = w^T x + b \tag{7}$$

with $w \in R^D$ also known as the weight vector and $b \in R$ is the offset from the origin to the hyperplane f. There are many reasons one might select linear estimator to model the relationship between known data pairs. The selection is highly problem dependent. From the arguments of previous section, a direct suggestion of using linear model might come from the situation where the amount of data N is not so large. From the bound in (5), choosing a class of functions that has high capacity h will likely lead to overfitting. In that case, a linear model probably works well.

For regression problem, y_{test} is simply calculated by $f(x_{test})$. On the other hand, classification task needs more treatment to build the learned model. Since we assume the mapping is linear, i.e. the function f is a hyperplane in the input space R^D, f devides R^D into two halves. Thus f inherently bears the characteristic of a binary classifier. The separating hyperplane between the two class is $f(x) = 0$, therefore classification decision is made by taking the sign of $f(x_{test})$, i.e. x_{test} belongs to positive class if $f(x_{test}) > 0$ and vice versa. Figure 1 illustrates a binary classifier.

In order to treat classification problem with k classes ($k > 2$), one might employ one-vs.-rest or one-vs.one strategy to break down the multiple-class classification problem into many binary classification problems. One-vs.-rest involves training k binary classifiers for k classes: for the i-th binary classifier f_k, the samples that belong to the i-th class are considered positive samples while the rest are considered negative samples. The classification decision is made by selecting the class that maximizes $f_k(x_{test})$. In one-vs.-one strategy, $k(k-1)/2$ binary classifiers are trained with each classifier trained by the samples from a pair of classes. The classification decision is made by evaluating all $k(k-1)/2$ classifiers, the class that receives maximum number of votes from all classifiers is selected as the label of the unseen example.

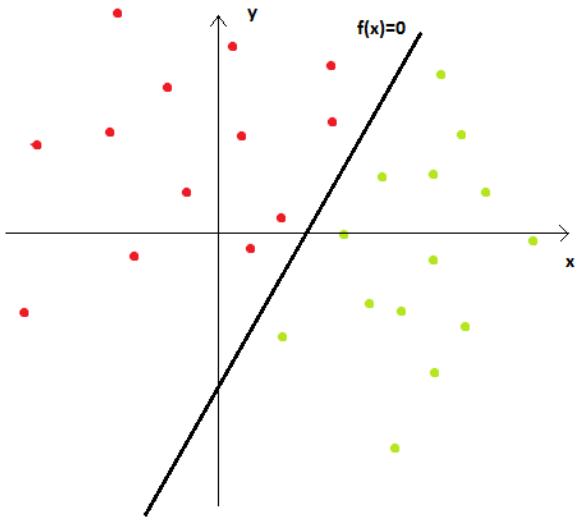

Figure 1. An example of a binary classifier in 2 dimensions.

After defining the hypothesis space $\mathcal{F}$, there are two decisions left in order to construct the learning problem: the selection of the loss function and the selection of the regularizer. We will discuss two types of loss function that are prevalent choices which lead to popular linear models: the least square estimator and the Support Vector Machine (SVM) estimator. In addition, for a moment, we will not consider adding a regularizer in our learning problem to illustrate its necessity. The learning problem is therefore considered as the optimization problem in (4).

- *Least Square Estimator*

With the square loss, $\mathcal{R}(f;\mathcal{Z})$ is defined as

$$\mathcal{R}(f;\mathcal{Z}) = \frac{1}{N}\sum_{i=1}^{N}\left(y_i - f(x_i)\right)^2 \tag{8}$$

The constructed estimator is then called least square estimator and (4) becomes

$$\min_{w,b} \frac{1}{N}\sum_{i=1}^{N}(y_i - w^T x_i - b)^2 \tag{9}$$

To simplify the notion, we denote $\theta = \begin{bmatrix} w \\ b \end{bmatrix} \in R^{D+1}$ and $x_i' = \begin{bmatrix} x_i \\ 1 \end{bmatrix} \in R^{D+1}, i = 1, \dots, N$. In addition, denote the appended training data by $X' = [x_1', \dots, x_N'] \in R^{D\times N}$ and the label vector by $\boldsymbol{y} = [y_1, \dots, y_N]^T \in R^N$. Consequently, (9) becomes:

$$\min_{\theta} J = \min_{\theta} \frac{1}{N}\sum_{i=1}^{N}(y_i - \theta^T x_i')^2 \tag{10}$$

The closed-form solution of (10) can be easily found by solving for the stationary point of J:

$$\frac{\partial J}{\partial \theta} = -\frac{2}{N}\sum_{i=1}^{N} x_i'(y_i - \theta^T x_i') \tag{11}$$

Setting $\partial J/\partial\theta = 0$ results in:

$$\sum_{i=1}^{N} x_i' \, {x_i'}^T \theta = \sum_{i=1}^{N} x_i' y_i$$

Or

$$X'{X'}^T\theta = X'\boldsymbol{y} \tag{12}$$

The system of linear equations in (12) is problematic if $X'{X'}^T$ is singular which either leads to no solution of θ or infinitely many solutions of θ. (12) is thus ill-posed. Following the regularization framework, the least square problem can be transformed to a well-posed problem by adding the square norm constraint on the weight, i.e. $\Omega(f) = \|\theta\|_2^2$ and we have the regularized least square version as:

$$\min_{\theta} J = \min_{\theta} \frac{1}{2N}\sum_{i=1}^{N}(y_i - \theta^T x_i')^2 + \frac{\lambda}{2}\theta^T\theta \tag{13}$$

Where $\lambda > 0$ is the parameter that controls the trade-off between minimizing the empirical loss and the norm penalty. Here the coefficient $\frac{1}{2}$ is added to both terms in (13) to simplify the calculation.

Solving for the stationary point of (13) results in:

$$(X'{X'}^T + \lambda I)\theta = X'\boldsymbol{y} \tag{14}$$

With I is the identity matrix of appropriate size. Since $\lambda > 0$, the matrix $X'{X'}^T + \lambda I$ is guaranteed to be non-singular, hence unique solution of θ exists in (14). In the following part, the contribution of the norm regularizer $\|\theta\|_2^2$ is further justified with another type of loss function, the hinge loss.

- *Support Vector Machine (SVM)*

The SVM was originally proposed by Cortes and Vapnik (1995) for binary classification problem, i.e. $y \in \{-1,1\}$. SVM classifier conforms to the above presented learning framework in which the same square norm regularizer $\|w\|_2^2$ is used in conjunction with the hinge loss to tighten the probabilistic bound (5):

$$\mathcal{R}(f;\mathcal{Z}) = \sum_{i=1}^{N} \max(0, 1 - y_i f(x_i)) \tag{15}$$

The SVM learning problem therefore has the form:

$$\min_{w,b} c \sum_{i=1}^{N} \max(0,1 - y_i(w^T x_i + b)) + \frac{1}{2}\|w\|_2^2 \qquad (16)$$

In (16), the parameter c is added to the loss term for simpler manipulation of the problem. It should be noted that $c \propto \frac{1}{\lambda}$, with λ defined in (6). Both parameters are used to control the trade-off between minimizing loss and penalty.

As can be seen from Figure 2, the "kink" in the max function leads to the non-differentiability of the objective in (16).

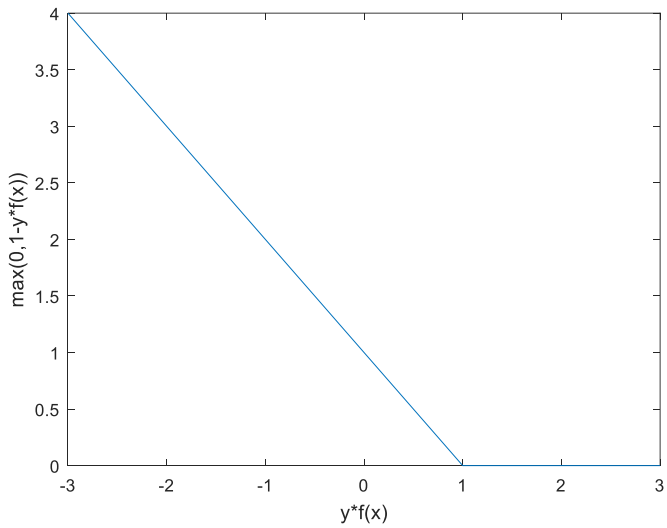

Figure 2. Hinge loss plot

In order to utilize calculus tools, the hinge loss is re-written using slack variables ξ_i:

$$\min_{w,b} c \sum_{i=1}^{N} \xi_i + \frac{1}{2}\|w\|_2^2 \qquad (17)$$

Subject to $\quad y_i(w^T x_i + b) \geq 1 - \xi_i, \forall i = 1, \dots, N$
$\quad\quad\quad\quad \xi_i \geq 0, i = 1, \dots, N$

The slack variables can be interpreted as the deviation of the predicted value $f(x_i)$ from the ideal or target value y_i. Particularly, the learned function $f(x_i)$ is expected to have similar sign as y_i, i.e. $y_i f(x_i)$ should be non-negative. By minimizing ξ_i, the lower bound of $y_i f(x_i)$ is maximized, forcing y_i and $f(x_i)$ to have the same sign. The ideal value of $f(x_i)$ is 1 or -1 when x_i belongs to positive or negative class respectively, corresponding to $\xi_i = 0$, which is the constrained minimum value.

Based on the Karush-Kuhn-Tucker (KKT) theorem (Fletcher, 1981), solving the optimization problem of (17) is equivalent to solving the following dual problem:

$$\max_{\boldsymbol{\alpha}} \sum_i^N \alpha_i - \frac{1}{2}\sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y_i y_j \boldsymbol{x}_i^T \boldsymbol{x}_j \tag{18}$$

Subject to $\sum_{i=1}^N y_i \alpha_i = 0;$
$0 \leq \alpha_i \leq c,\ i = 1, \dots, N$

Where $\boldsymbol{\alpha} = [\alpha_1, \dots, \alpha_N]^T$ is the Lagrange multipliers. Having determined the optimal $\boldsymbol{\alpha}^*$ in (18), the optimal weight vector w^* and the intercept term b^* can be computed as:

$$w^* = \sum_{i=1}^N a_i^* y_i x_i \tag{19}$$

$$\mathrm{b}^* = 1 - w^{*T} x_i \ \text{ for } y_i = 1 \tag{20}$$

Alternatively, the decision function f can be expressed in terms of the Lagrange multipliers $\boldsymbol{\alpha}$ as:

$$f(x) = \sum_{i=1}^N \alpha_i^* x_i^T x + b^* \tag{21}$$

Comparing the primal and the dual problem, both optimization problems are in quadratic convex form, hence global solutions exist. However, the dual problem is simpler since the slack variables together with its Lagrange multipliers do not appear in the dual form, resulting in quadratic optimization problem with single variable $\boldsymbol{\alpha}$ and linear constraints. More importantly, the dual problem and its resulting decision boundary are cast entirely in terms of the dot product of the training data, i.e. $x_i^T x_i$, which allows the utilization of the so-called "kernel trick" presented in the following section.

We have seen so far how $\|w\|_2$ is incorporated into the learning objective to transform an ill-posed problem to a well-posed one as well as to tighten the probabilistic bound between the expected loss and empirical loss in (5). For the class of linear models, $\|w\|_2$ has an interesting geometric interpretation for the SVM classifier. We can see that the quantity $f(x)$ represents the algebraic distance between x and the decision boundary. Denote x_p the projection of x onto the hyperplane $f(x) = 0$, we have

$$x = x_p + r\frac{w}{\|w\|_2} \tag{22}$$

We can see that r is positive when x belongs to the positive side of $f(x) = 0$ and vice versa. Because $f(x_p) = 0$:

$$f(x) = r\|w\|_2$$

Or

$$r = \frac{f(x)}{\|w\|_2} \tag{23}$$

Now suppose the training data is linearly separable, i.e. there exists a hyperplane that perfectly separates the two classes, the optimal function f should produce the value $f(x_i)$ having the same sign as the target label y_i, i.e.

$$\begin{aligned} y_i f(x_i) > 0, \forall\, y_i = 1 \\ y_i f(x_i) < 0, \forall y_i = -1 \end{aligned} \tag{24}$$

Since $f(x)$ and $cf(x)$ define the same decision boundary, we can fix the constraint in (24) as

$$\begin{aligned} y_i f(x_i) \geq 1, \forall\, y_i = 1 \\ y_i f(x_i) \leq -1, \forall\, y_i = -1 \end{aligned} \tag{25}$$

Denote x_+^* and x_-^* the closest samples from the positive class and negative class to the optimal hyperplane satisfying (25), i.e.

$$\begin{aligned} f(x_+^*) = 1 \\ f(x_-^*) = -1 \end{aligned} \tag{26}$$

Combining (23) and (26), it is clear that the margin or the sum of geometric distance between x_+^* and x_-^* to the decision boundary is:

$$d = \frac{2}{\|w\|_2} \tag{27}$$

This results shows that the inclusion of $\|w\|_2$ into hinge loss minimization problem of SVM classifier aims at maximizing the margin d. The linear separability assumption of the data might not be true, hence the constraint in (25) can be loosened by adding the slack variables as in (17). Figure 3 shows an example in which a large margin solution is better although the constraints in (25) are violated. The samples that satisfy (26) are called support vectors since they have direct effect in the selection of optimal hyperplane.

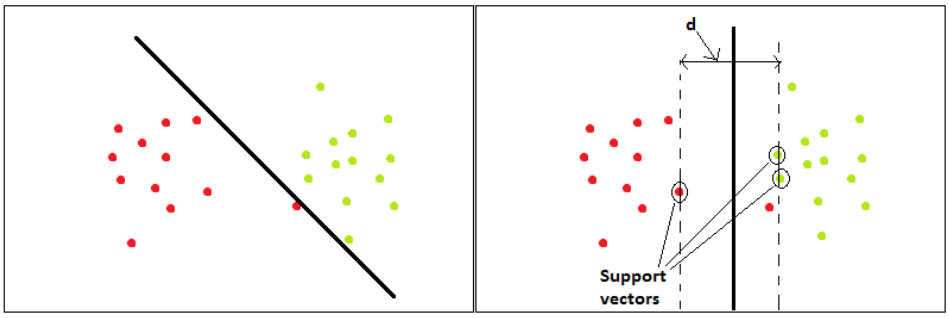

Figure 3. On the left: the hyperplane perfectly separates training samples. On the right: SVM optimal hyperplane with one Red sample misclassified. Although having one training sample misclassified, SVM solution separates the clusters better.

2.3 Kernel trick

The linear model is simple to interpret and it is assumed that the data in the input space can be separated by a hyperplane in case of the classification problems. In fact, this assumption is naïve in practice. For example, the simple XOR problem cannot be solved by a hyperplane in the input space as illustrated in Figure 4.

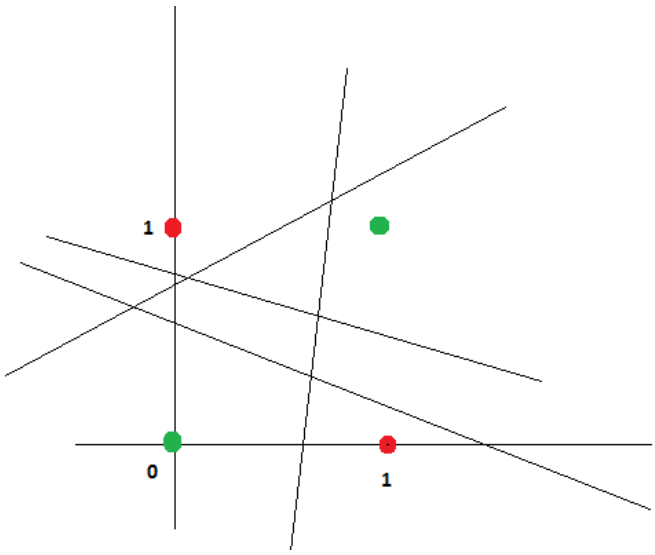

Figure 4. XOR problem: two class Red and Green cannot be separated by any line.

The success of SVM can be attributed to the application of Mercer's Theorem (Boser, 1992) and Reproducing Kernel Hilbert Spaces (RKHS) which allows the computation of linear decision functions in a feature space which is nonlinearly connected to the input space without explicit mapping. The idea of the kernel trick is to first map the input data into a feature space $\mathcal{F}$ via the nonlinear function ϕ chosen a priori:

$$\begin{aligned} &\phi: R^D \to \mathcal{F} \\ &x \mapsto \phi(x) \end{aligned} \tag{28}$$

The learned function is then formulated in $\mathcal{F}$, i.e. $\phi(x_1), \dots, \phi(x_N)$ is the set of the training data. Instead of explicitly mapping the input x_i through $\phi(x_i)$, the kernel trick aims at formulating the learning problem only in terms of the dot products which could be efficiently evaluated by a dot-product kernel function $k(.,.)$. Specifically, the kernel trick utilizes $k(.,.)$ which has the following property:

$$k(x_i, x_j) = \phi(x_i)^T \phi(x_j) \tag{29}$$

That is: $\phi(x_i)^T \phi(x_j)$ is evaluated by $k(x_i, x_j)$. The question is, given a kernel function $k(.,.)$, how can one know the existence of the feature map

ϕ that satisfies (29). The answer lies in the positive-definiteness of the kernel function.

Denote $\boldsymbol{K}$ the kernel matrix of the training data whose element $\boldsymbol{K}_{ij} = k(x_i, x_j);\ i, j = 1, \dots, N$. $\boldsymbol{K}$ is said to be positive-definite if:

$$u^T \boldsymbol{K} u > 0, \forall u \in R^N \tag{30}$$

For a kernel function $k(.,.)$ and its kernel matrix $\boldsymbol{K}$ to be positive-definite, it is necessary that $k(.,.)$ is symmetric, i.e. $k(x_i, x_j) = k(x_j, x_i)$, and $k(x, x) \geq 0$.

It is proofed that for any positive definite kernel $k(.,.)$, there exists a mapping ϕ into $\mathcal{F}$ such that (29) is satisfied. Interested readers could refer to e.g. (Schölkopf and Smola, 2002) for more details.

It is now obvious to see how the kernel trick is applied to SVM classifier. In the feature space $\mathcal{F}$, the dual problem in (18) becomes:

$$\max_{\boldsymbol{\alpha}} \sum_i^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y_i y_j \phi(x_i)^T \phi(x_j) \tag{31}$$

Subject to $\sum_{i=1}^N y_i \alpha_i = 0$
$0 \leq \alpha_i \leq c,\ i = 1, \dots, N$

The dot product $\phi(x_i)^T\, \phi(x_j)$ is then evaluated by $k(x_i, x_j)$, (31) becomes:

$$\max_{\boldsymbol{\alpha}} \sum_i^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y_i y_j k(x_i, x_j) \tag{32}$$

Or

$$\max_{\boldsymbol{\alpha}} \mathbf{1}^T \boldsymbol{\alpha} - \frac{1}{2} (\boldsymbol{\alpha} * \boldsymbol{y})^T \boldsymbol{K} (\boldsymbol{\alpha} * \boldsymbol{y}) \tag{33}$$

Subject to $\boldsymbol{\alpha}^T \boldsymbol{y} = 0,\ \mathbf{0} \leq \boldsymbol{\alpha} \leq \boldsymbol{c},\ i = 1, \dots, N.$

Where $\mathbf{1}$ denotes the vector of ones of appropriate size and $*$ denotes the element-wise product operator.

Consequently, the decision function is:

$$f(x) = \sum_{i=1}^N \alpha_i^* k(x_i, x) + b^* \tag{34}$$

It is however, less obvious to see how the kernel trick can be applied to the least square classifier. Denote $\boldsymbol{\Phi}' = [\phi(x_1{}'), \dots, \phi(x_N{}')]$ and hence $\boldsymbol{K}' = \boldsymbol{\Phi}'^T \boldsymbol{\Phi}'$, the regularized least square solution in $\mathcal{F}$ becomes:

$$\theta = (\boldsymbol{\Phi}' \boldsymbol{\Phi}'^T + \lambda I)^{-1} \boldsymbol{\Phi}' \boldsymbol{y} \tag{35}$$

In order express θ in terms of the kernel matrix $\boldsymbol{K}' = \boldsymbol{\Phi}'^T \boldsymbol{\Phi}'$, we have to resort to the identity:

$$(P^{-1} + B^T R^{-1} B)^{-1} B^T R^{-1} = P B^T (B P B^T + R)^{-1} \quad (36)$$

And (35) can be rewritten as:

$$\theta = \mathbf{\Phi}'(\mathbf{\Phi}'^T \mathbf{\Phi}' + \lambda I)^{-1} \boldsymbol{y} = \mathbf{\Phi}'(\boldsymbol{K}' + \lambda I)^{-1} \boldsymbol{y} \quad (37)$$

Consequently, the decision function becomes:

$$f(x) = \theta^T \phi(x) = y(\boldsymbol{K}' + \lambda I)^{-1} k(x) \quad (38)$$

Where in (38) the kernel vector $k(x)$ is defined as $k(x) = [k(x_1, x), \dots, k(x_N, x)]^T$.

It can be seen from (33) and (38) that the application of the kernel trick to the least square model and SVM model avoids the explicit computation of the mapping ϕ. This is beneficial since the explicit transformation of x_i to $\phi(x_i)$ incurs high computational cost, in some cases even prohibitive. In addition, the dimension of $\mathcal{F}$ is expected to be much larger than the original input space so that the training data $\phi(x_i)$ is linearly separable with higher probability (Cover, 1987). Therefore, the explicit calculation of the dot product in $\mathcal{F}$ requires more computations compared to the original input space.

3 RELATED WORK

Previous work on learning linear models exploiting feature noising as a regularizer can be divided into two strategies: explicit data corruption and implicit data corruption. The former line of research includes virtual SVM (Burges and Schölkopf, 1997), adversarial worst case training (Globerson and Roweis, 2006; Dekel and Shamir, 2008; Teo, 2008). Under explicit corruption, the training data is corrupted multiple times M with respect to a corrupting distribution $p(\tilde{\boldsymbol{x}}_n | \boldsymbol{x}_n)$. This approach results in an enriched training set $\{\tilde{\boldsymbol{x}}_{nm}, y_n\},\ n = 1, \dots, N,\ m = 1, \dots, M$. The standard classifier is then trained by minimizing the average loss over the augmented set. In addition to the inelegance of the method, it clearly incurs high computational cost in many practical problems when $M \to \infty$.

The latter approach is more elegant by considering the expectation of the corrupted data. Specifically, the empirical loss in the standard models is replaced by its expectation with respect to the corrupting distribution. This approach includes the work of (Wager, Wang and Liang, 2013), (van der Maaten, 2013), (Wang and Manning, 2013). While the expectation of quadratic loss and exponential loss have a close-form expression, the expected logistic loss or the loss in Generalized Linear Model (GLM) does not have a close-form expression, leading to the approximation strategy

by using second-order Taylor expansion (Wager, Wang and Liang, 2013). Similarly, the non-differentiability of the hinge loss in SVM model prohibits a close-form expression. Chen (2014) derived a variational upper-bound on the expected hinge loss of the SVM classifier and proposed an Iteratively Re-weighted Least Square (IRLS) algorithms to minimize the upper-bound called Dropout-SVM. This section, we present the work of Chen as a basis to compare with our proposed method since both methods incorporate Dropout training into SVM.

3.1 Dropout-SVM

Under implicit corruption, there are two assumptions related to the corrupting model. It is assumed that the corrupting distributions are independent and unbiased. Particularly, denote $\tilde{x}$ the corrupted version of input feature x, we have:

$$p(\tilde{x}|x) = \prod_{d=1}^{D} p(\tilde{x}_d|x_d; n_d) \tag{39}$$

$$E_p[\tilde{x}|x] = x \tag{40}$$

Where in (39) n_d denotes the natural parameter of the exponential distribution family and $E_p[.]$ denotes the expectation taken over p.
The objective of SVM classifier under implicit corruption is defined:

$$\min_{\theta} 2c\mathcal{L}_h(\theta, \mathcal{Z}) + \|w\|_2^2 \tag{41}$$

Where $\mathcal{L}_h(\theta, \mathcal{Z}) = \sum_{i=1}^{N} \mathrm{E}_p[\max(0,1 - y_i(w^T\tilde{x}_i + b)]$ is the expected hinge loss with respect to $p(\tilde{x}|x)$. A multiplication factor of 2 is added to $\mathcal{L}_h(\theta, \mathcal{Z})$ to simplify the calculation.

Directly tackle (41) is intractable since there exists no close-form of $\mathcal{L}_h(\theta, \mathcal{Z})$. In Dropout-SVM, $\mathcal{L}_h(\theta, \mathcal{Z})$ is replaced by its upper-bound and the objective of Dropout-SVM is to minimize the upper-bound with respect to $p(\tilde{x}|x)$. Exploiting the pseudo-likelihood expression of the response variable and Jensen's inequality, a variational upper-bound of $\mathcal{L}_h(\theta, \mathcal{Z})$ was derived:

$$\mathcal{L}_h\big(\theta, q(\boldsymbol{v})\big) = \sum_1^N \left\{\frac{1}{2}\mathrm{E_q}[\log(v_n)] + \mathrm{E}_q\left[\frac{1}{2v_n}\mathrm{E}_p(v_n + cl_n)^2\right]\right\} - H(\boldsymbol{v}) + c' \tag{42}$$

Where $H(\boldsymbol{v})$ is the entropy of the variational distribution $q(\boldsymbol{v}) = \prod_n q(v_n)$, $\boldsymbol{v} \triangleq \{v_n\}_{n=1}^N$, c' is a constant and $l_n = 1 - y_n(w^T x_n + b)$.

The objective of Dropout-SVM is hence reformulated as:

$$\min_{\boldsymbol{\theta}, q(\boldsymbol{v}) \in \mathcal{P}} \|w\|_2^2 + \mathcal{L}_h(\theta, q(\boldsymbol{v})) \tag{43}$$

With $\mathcal{P}$ denotes the simplex space of normalized distributions.

The upper-bound in (42) includes the term $\mathrm{E}_p(v_n + cl_n)^2$ which is the expectation of a quadratic loss and can be analytically computed given $q(\boldsymbol{v})$. Dropout-SVM exploits an iterative two-step approach which resembles the procedure of an EM algorithm:

For $q(v)$: this steps minimize (43) with respect to the variational distribution $q(\boldsymbol{v})$. Denote $\mathcal{GIG}(x; p, a, b) \propto x^{p-1} \exp\left(-\frac{1}{2}\left(\frac{b}{x} + ax\right)\right)$ a generalized inverse Gaussian distribution, the optimal $q(\boldsymbol{v})$ is given as:

$$q(\boldsymbol{v}) \propto \prod_n \mathcal{GIG}\left(v_n; \frac{1}{2}; 1, c^2 \mathrm{E}_p[l_n^2]\right) \tag{44}$$

With the second order expectation of the sample loss calculated by $\mathrm{E}_p[l_n^2] = w^T\left(\mathrm{E}_p[\tilde{x}_n]\mathrm{E}_p[\tilde{x}_n]^T + V_p[\tilde{x}_n]\right)w - 2y_n w^T \mathrm{E}_p[\tilde{x}_n] + 1.\mathrm{V_p}[\tilde{x}_n]$ is a $D \times D$ diagonal matrix with variance of $\tilde{x}_{nd}$ in the d-th diagonal element. With the unbiased assumption of the corrupting distribution, $\mathrm{E}_p[\tilde{\boldsymbol{x}}_n] = x_n$ as mentioned in (40). The remaining question is how the variance $\mathrm{V_p}[\tilde{x}_n]$ is calculated. This question was addressed by the work of van der Maater (2013) for the class of exponential functions.

For θ: this step minimizes (43) with respect to w when fixing $q(\boldsymbol{v})$. The objective (43), after discarding irrelevant terms, has the form of a re-weighted quadratic loss. Denote $k_n = \frac{1}{c\sqrt{\mathrm{E}_p[l_n^2]}}$ the modified weights and $y_n^h = \left(1 + \frac{1}{ck_n}\right) y_n$ is the re-weighted label. The closed-form solution of w is then given by:

$$w = \left(\frac{2}{c^2} I + \sum_{n=1}^{N} k_n \mathrm{E}_p[\tilde{\boldsymbol{x}}_n \tilde{\boldsymbol{x}}_n^T]\right)^{-1} \left(\sum_{n=1}^{N} k_n y_n^h \mathrm{E}_p[\tilde{\boldsymbol{x}}_n]\right) \tag{45}$$

Dropout-SVM alternates between the above two steps to calculate $q(v)$ and θ until a stopping criterion is met. It should be noted that Dropout-SVM presented above is a linear model in the input space. A nonlinear extension using the idea from representation learning was also proposed in the work of Chen. The variational upper-bound of the expected hinge loss has a similar form as in (42) in the representation space with additional transform coefficient variables. For more details of the extension, interested readers should refer to (Chen, 2014).

4 PROPOSED METHOD

This chapter starts by the formulation of the new regularization scheme which exploits Dropout technique on the sample level during training, called DropSVM. The proposed method is then interpreted under the statistical learning framework and regularization theory with the motivation behind the new formulation. In addition, comparison are made between DropSVM and Dropout-SVM and the novelty in the proposed method is pointed out.

4.1 The proposed DropSVM classifier

We consider the general case of SVM classifier under the kernel formulation in which linear SVM is a special case using the linear kernel function:

$$k(x_i, x_j) = x_i^T x_j \tag{46}$$

The training data is assumed to be mapped to an arbitrary-dimensional Hilbert space $\mathcal{F}$ through the nonlinear mapping:

$$\phi(\cdot): x_i \in R^D \mapsto \phi(x_i) \in \mathcal{F} \tag{47}$$

The dimension of $\mathcal{F}$ could be finite or infinite. For example, dimension of the space induced by the class of polynomial kernel is finite while for RBF kernel is infinite. Based on the Representer Theorem (Schölkopf and Smola, 2002), we can represent the weight w of the decision boundary in $\mathcal{F}$ in terms of the linear combination of the training data in $\mathcal{F}$. Denotes $\mathbf{\Phi} = [\phi(\boldsymbol{x}_1), \dots, \phi(\boldsymbol{x}_N)]$ and $\boldsymbol{K} = \mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi}$ then:

$$w = \sum_{i=1}^{N} \gamma_i \phi(x_i) = \mathbf{\Phi}\boldsymbol{\gamma} \tag{48}$$

Where $\boldsymbol{\gamma} = [\gamma_1, \dots, \gamma_N] \in R^N$ is the vector of the combination coefficients to reconstruct w in terms of $\phi(x_i), i = 1, \dots, N$.

We introduce the Dropout version of w denoted as $\widetilde{w}_{ji}$:

$$\widetilde{w}_{ji} = \left(\mathbf{\Phi}(\mathbf{1}\, \boldsymbol{m}_{ji}^T)\right) \boldsymbol{\gamma} \quad , i = 1, \dots, N; j = 1, \dots, E \tag{49}$$

With $m_{ji} \in R^N$ is a binary mask vector in epoch j used to classify sample i. Each element of m_{ji} is equal to 1 with a probability of p and equal to 0 with a probability $(1 - p)$. $\mathbf{1}$ is a vector of ones. We can interpret $\widetilde{w}_{ji}$ as follows: while w is constructed by the linear combination of $\phi(x_i)$ with coefficient $\boldsymbol{\gamma}$, $\widetilde{w}_{ji}$ is constructed by using the same combination coefficients $\boldsymbol{\gamma}$ but with some basis $\phi(x_i)$ omitted or dropped out by the probability p. In other words, the original w is learned by the contribution of every training sample $\phi(x_i)$ in the kernel subspace and the Dropout version $\widetilde{w}_{ji}$ is created with some training samples not taken into account.

The modified SVM optimization problem is the following:

$$\min_{w,b} \frac{1}{2} w^T w + c \sum_{i=1}^{N} \xi_i + q \frac{1}{E} \sum_{j=1}^{E} \sum_{i=1}^{N} \left(w^T \phi(x_i) - \widetilde{w}_{ji}^T \phi(x_i) \right)^2 \quad (50)$$

subject to $\quad y_i(w^T \phi(x_i) + b) \geq 1 - \xi_i, \;\; i = 1, \dots, N$
$\quad \xi_i \geq 0, i = 1, \dots, N$

In (50), the last term is added to the standard SVM minimization problem with q is the parameter to control the amount of regularization this new term contributes to the learned decision function and E denotes the number of epochs used in training. The learned weight w is expected to behave in a similar manner as in standard SVM with another constraint: the output produced by w should be as close as possible to the output produced by $\widetilde{w}_{ji}$. The motivation behind this constraint will be discussed later in this section. We now proceed to derive the dual problem of (50) by substituting $w = \boldsymbol{\Phi\gamma}$ and $\widetilde{w}_{ji} = \left(\boldsymbol{\Phi}(\mathbf{1}\, \boldsymbol{m}_{ji}^T)\right) \boldsymbol{\gamma}$:

$$\min_{w,b} \frac{1}{2} \boldsymbol{\gamma}^T \boldsymbol{K} \boldsymbol{\gamma} + c \sum_{i=1}^{N} \xi_i + q \frac{1}{E} \sum_{j=1}^{E} \sum_{i=1}^{N} \left(\boldsymbol{\gamma}^T \boldsymbol{\Phi}^{\mathrm{T}} \phi(x_i) - \boldsymbol{\gamma}^T \left(\boldsymbol{\Phi}(\mathbf{1}\, \boldsymbol{m}_{ji}^T) \right)^T \phi(x_i) \right)^2 \quad (51)$$

Subject to $\quad y_i(\gamma^T \boldsymbol{\Phi}^{\mathrm{T}} \phi(x_i) + b) \geq 1 - \xi_i, \;\; i = 1, \dots, N$
$\quad \xi_i \geq 0, i = 1, \dots, N$

To simplify the expression, we further denote the kernel vector of the i-th sample by $\boldsymbol{k}_i = \boldsymbol{\Phi}^{\boldsymbol{T}} \phi(\boldsymbol{x}_i)$ and the Dropout version of $\boldsymbol{k}_i$ in epoch j by $\widetilde{\boldsymbol{k}}_i^{(j)} = \left(\boldsymbol{\Phi}(\mathbf{1}\, \boldsymbol{m}_{ji}^T)\right)^T \phi(\boldsymbol{x}_i)$, (51) becomes:

$$\min_{w,b} \frac{1}{2} \boldsymbol{\gamma}^T \boldsymbol{K} \boldsymbol{\gamma} + c \sum_{i=1}^{N} \xi_i + q \frac{1}{E} \sum_{j=1}^{E} \sum_{i=1}^{N} \left\| \boldsymbol{\gamma}^T \widehat{\boldsymbol{k}}_i^{(j)} \right\|_2^2$$

Or

$$\min_{w,b} \frac{1}{2} \boldsymbol{\gamma}^T \boldsymbol{K} \boldsymbol{\gamma} + c \sum_{i=1}^{N} \xi_i + q \frac{1}{E} \sum_{j=1}^{E} \boldsymbol{\gamma}^T \widehat{\boldsymbol{K}}^{(j)} \widehat{\boldsymbol{K}}^{(j)^T} \boldsymbol{\gamma} \quad (52)$$

Subject to $\quad y_i(\gamma^T \boldsymbol{k}_i + b) \geq 1 - \xi_i, i = 1, \dots, N$
$\quad \xi_i \geq 0, i = 1, \dots, N$

Where $\widehat{\boldsymbol{K}}^{(j)} = \left[\widehat{\boldsymbol{k}}_1^{(j)}, \dots, \widehat{\boldsymbol{k}}_N^{(j)} \right]$ and $\widehat{\boldsymbol{k}}_i^{(j)} = \boldsymbol{k}_i - \widetilde{\boldsymbol{k}}_i^{(j)}$. We can simply consider $\widehat{\boldsymbol{k}}_i$ as equal to $\boldsymbol{k}_i$ with some of its elements set to zero with probability $(1 - p)$.

Let $\alpha_i, \beta_i, i = 1, \dots, N$ be the Lagrange multipliers, the Lagrangian function of (52) is:

$$J(\gamma, b, \xi, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \frac{1}{2} \boldsymbol{\gamma}^T \boldsymbol{K} \boldsymbol{\gamma} + c \sum_{i=1}^{N} \xi_i + q \frac{1}{E} \sum_{j=1}^{E} \boldsymbol{\gamma}^T \widehat{\boldsymbol{K}}^{(j)} \widehat{\boldsymbol{K}}^{(j)^T} \boldsymbol{\gamma} - \sum_{i=1}^{N} \alpha_i [y_i(\gamma^T k_i + b) - 1 + \xi_i] - \sum_{i=1}^{N} \beta_i \xi_i \quad (53)$$

Solving for the stationary condition we have the following results:

$$\frac{\partial J}{\partial \gamma} = 0 \;\Rightarrow \gamma = \left[\boldsymbol{K} + \frac{2q}{E}\sum_{j=1}^{E} \widehat{\boldsymbol{K}}^{(j)}\widehat{\boldsymbol{K}}^{(j)^T}\right]^{-1} \sum_{i=1}^{N} \alpha_i y_i k_i \tag{54}$$

$$\frac{\partial J}{\partial b} = 0 \;\Rightarrow \sum_{i=1}^{N} \alpha_i y_i = 0 \tag{55}$$

$$\frac{\partial J}{\partial \xi_i} = 0 \;\Rightarrow \alpha_i + \beta_i = c, \forall i = 1, \dots, N \tag{56}$$

Substitute (4.9), (4.10) and (4.11) into (4.8) we get the dual problem:

$$\max_{\boldsymbol{\alpha}} \mathbf{1}^T \boldsymbol{\alpha} - \frac{1}{2}(\boldsymbol{\alpha} * \boldsymbol{y})^T \overline{\boldsymbol{K}}(\boldsymbol{\alpha} * \boldsymbol{y}) \tag{57}$$

Subject to $\boldsymbol{\alpha}^T \boldsymbol{y} = 0$

$\mathbf{0} \leq \boldsymbol{\alpha} \leq \boldsymbol{c}, i = 1, \dots, N$

With $\overline{\boldsymbol{K}}$ defined as $\overline{\boldsymbol{K}} = \boldsymbol{K}\left[\boldsymbol{K} + \frac{2q}{E}\sum_{j=1}^{E} \widehat{\boldsymbol{K}}^{(j)}\widehat{\boldsymbol{K}}^{(j)^T}\right]^{-1} \boldsymbol{K}$.

Comparing (57) and (33), it is clear that the two dual problems are exactly the same, except for the different kernel matrices used. We can consider that the adoption of the proposed regularization presented above is equivalent to solving the original SVM in a transformed kernel space. That is the solution to the proposed optimization problem not only possesses an elegant expression but also enables an efficient implementation through the existing libraries without any modification due to its resemblance to the standard SVM. For example, LIBSVM (Chang and Lin, 2011) has the option to train SVM with any given kernel matrix.

4.2 Discussion

In the previous section, we posed a new optimization problem for the SVM classifier without discussing its implication and motivation. This section presents the intuition and meaning of Dropout technique on the sample level as a new regularizer to the classic SVM. In addition, the novelty of the proposed method is pointed out through its fundamental difference from the previously proposed ones.

- *Intuition and motivation*

As mentioned earlier, nonlinear SVM through the kernel trick provides the classifier an ability to fit training data well, however it might lead to the situation of overfitting. The solution boundary of the standard SVM is governed completely by the support vectors that lie at the margin. Given the hypothesis that some of the support vectors contaminated by noise during measuring process, the decision boundary learned through

maximum margin classifier can be consequently affected, resulting in high classification error during test phase. This motivates us to train SVM by randomly dropping out some of the samples. However, in case the dropped out support vectors are indeed close to their expected values, i.e. close to noise-free states, they contain important separating information, hence should be kept.

To address this motivation, one might propose to train M SVM classifiers with M training data versions from which some samples are dropped out and the decision is constructed by averaging out decision of M classifiers. This approach resembles the explicit feature corruption mentioned previously, hence incurs high computational cost and might be prohibitive in some applications. Our proposed method in fact addresses this motivation. We could view $\widetilde{w}_{ji}$ as the SVM weight learned when dropping out some samples in epoch j and we would like our model to learn w that produces the output as close as possible to the output produced by $\widetilde{w}_{ji}$ for any epoch $j = 1, \dots, E$. This objective is done by incorporating the total square output difference, $\sum_{j=1}^{E}\sum_{i=1}^{N}\left(w^T\phi(x_i) - \widetilde{w}_{ji}^T\phi(x_i)\right)^2$, into the minimization process. In epoch j, if the dropped out samples do not contain any support vectors, the decision boundary does not change, otherwise, we minimize the differences. Since we still keep minimizing $\|w\|_2^2$ together with the hinge loss, the maximum margin characteristic of the classifier is still retained. That is, in any case of the above hypothesis, our proposed model still retains the separating information while discarding noisy critical samples.

- *Implication under statistical learning theory*

In addition to the geometric interpretation based on the support vectors, DropSVM conforms to the statistical learning theory. In order to see this, it should be emphasized that the target of the learning problem is to construct a learned function that can generalize well on the test set. In other words, the learned function should minimize the expectation of the loss functional. The bound presented in (5) gives a formal probabilistic guarantee of the generalization performance in terms of the empirical loss, the number of training samples and the complexity of the hypothesis space.

Since the hypothesis space is the space of linear functions, standard SVM tightens the bound in (5) by penalizing on the deviation of f through $l2$-norm of the weight vector. In addition to the penalty put on the weight vector w, our proposed method also penalizes on the deviation of the output produced by w from its Dropout versions. In other words, additional restriction is put on the complexity of the class of linear functions, i.e. w. This is expected to further reduce the gap between expected loss and empirical loss in (5), which is empirically proved through the set of experiments.

- *Versatility*

The reasoning above shows that the proposed regularization term $\frac{1}{E}\sum_{j=1}^{E}\sum_{i=1}^{N}\left(w^T\phi(x_i) - \widetilde{w}_{ji}^T\phi(x_i)\right)^2$ is suitable to control the complexity of the class of linear functions. This leads to the question: can we formulate the learning problem using this proposed regularizer with other type of loss function? The answer is yes!

For example, the new regularizer can be incorporated into the square loss using the representation $w = \boldsymbol{\Phi\gamma}$ as follows:

$$\min_{\boldsymbol{\gamma}} \frac{1}{2}\sum_{i=1}^{N}(y_i - \boldsymbol{\gamma}^T k_i)^2 + q\frac{1}{E}\sum_{j=1}^{E}\boldsymbol{\gamma}^T\widehat{\boldsymbol{K}}^{(j)}\widehat{\boldsymbol{K}}^{(j)^T}\boldsymbol{\gamma} \qquad (58)$$

Solving for the stationary condition, the solution of (58) can be derived:

$$\boldsymbol{\gamma} = \left(\boldsymbol{K}\boldsymbol{K}^T + \frac{2q}{E}\sum_{j=1}^{E}\widehat{\boldsymbol{K}}^{(j)}\widehat{\boldsymbol{K}}^{(j)^T}\right)^{-1}\boldsymbol{K}\boldsymbol{y} \qquad (59)$$

- *DropSVM vs DropoutSVM*

The similar characteristic between our proposed DropSVM and DropoutSVM is that the probabilistic Dropout technique is employed. However, the fundamental difference is that DropSVM employs a Dropout scheme on the training sample level while in Dropout-SVM a random subset of features are dropped during training. The significance of our method can be justified under both geometrical interpretation and statistical learning framework. While feature noising applied to other linear models whose expected empirical risk has an explicit form of penalty term that regularizes the complexity of the estimator, the upper-bound in (42) is difficult to interpret under the statistical learning framework. For example, in (Wager, Wang and Liang, 2013) the generalized linear model has the expected empirical loss of the form $\sum_{i=1}^{N} l_{x_i,y_i}(w) + R(w)$ with $\sum_{i=1}^{N} l_{x_i,y_i}(w)$ is the empirical loss under uncorrupted data, $R(w)$ is thus seen as the regularizer of model's complexity incorporated into the learning problem.

Regarding the solution of the two methods, DropSVM is a quadratic optimization problem with linear constraint, hence global solution exists. Moreover, our additional regularizing term leads to exactly the same dual form as standard SVM with only different kernel matrix. This has the benefit of existing efficient SVM implementation such as the SMO algorithm implemented in LIBSVM. Contrarily, the solution of Dropout-SVM provides no guarantee of a global solution.

Finally, DropSVM is formulated under Representer Theorem, hence readily possesses the nonlinearity extension without explicit nonlinear transformation. On the contrary, nonlinear extension of Dropout-SVM requires approximation step and the modified upper bound is optimized with an additional variable, i.e. the transformation coefficients. That adds up computations cost to achieve the nonlinearity.

5 EXPERIMENTS

5.1 Experiment settings

In order to evaluate the proposed method, two set of experiments are conducted with the candidate classifiers: our proposed DropSVM and standard SVM classifier. The first set of experiments consists of twelve standard classification problems extracted from machine learning repository of University of California Irvine (UCI) (Lichman, 2013). The data set size ranges from as small as 182 samples up to 1000 samples. A summary of each data set is presented in Table 1.

In order to make maximal use of the available data, we employ the cross-validation procedure by randomly dividing the data into five sets in a stratified manner. In each cross-validation round, four out of five sets are used to train the candidate classifiers while the left-over set is used as a test set. Due to the random nature of the method, the experiments are conducted five times for each dataset and the average classification rate is measured over all five experiments and presented as the final result. Regarding the kernel used, both DropSVM and SVM are trained using the RBF kernel function, whose parameter σ is set equal to the mean Euclidean distance between training vectors, which is a popular scaling factor. The regularization parameters, i.e. c for the standard SVM and c, q, p for DropSVM are selected following a grid search strategy using the ranges $c = 10^{\{-3,\dots,3\}}$, $q = 10^{\{-3,\dots,3\}}$ and $p = \{0.1, 0.2, \dots, 0.9\}$. For the proposed regularizer, a variety of number of epochs are tested $E = \{10, 25, 50, 100\}$.

Table 1. UCI dataset information

Data set	# Samples	# Dimensions	# Classes
Australian	690	14	2
Column2c	310	6	2
Column3c	310	6	3
German	1000	24	2
Glass	214	9	6
Heart	270	13	2
Indians	768	8	2
Ionosphere	351	34	2
Relax	182	12	2
Spect	267	22	2
Spectf	267	44	2
Syn. Con.	600	60	6

The second set of experiments are conducted to test human action recognition performance. Three active datasets including Hollywood2, the Olympic sports and Hollywood3D are introduced to gauge the performance.

In Hollywood2 (Marszalek, Laptev and Schmid, 2009), there are 1707 videos illustrating 12 human actions which had been collected from 69 different Hollywood movies. The standard training-testing split provided by the database are used (823 videos for training and 884 videos for testing). It should be noted that the training set and the test set come from different movies. Since a video potentially contains more than one action, we evaluate the classifiers by calculating the mean average precision over all the classes (mAP).

Similar settings are applied to the Olympic sports dataset (Niebles, Chen and Fei Fei, 2010) and the Hollywood3D dataset (Hadfield and Bowden, 2013). That is, the standard training-testing split provided by the database is used and mAP is calculated. The Olympic sports dataset consists of 649 training videos and 134 testing videos depicting 16 sports activities. The Hollywood3D consists of 951 video pairs of left and right channel) extracted from a set of Hollywood movies. This dataset has 13 action labels and a “no action” label.

In order to extract meaningful training input from video data, we use the video representation proposed in (Wang and Schmid, 2013) including HOG, HOF, MBHx, MBHy and trajectory descriptors. Bag-of-Words (BoW) representation is exploited for each descriptor. The descriptors extracted from each video is consequently encoded using 4000 codewords, the encoding scheme is similar to those presented in (Wang and Schmid, 2013). Regarding the kernel type used in human action recognition task,

we use the RBF-χ^2 kernel which outperforms other types when BoW representation is employed (Rozenfeld, 2008). The regularization settings are similar to the ones applied to UCI datasets, i.e. $c = 10^{\{-3,\dots,3\}}$, $q = 10^{\{-3,\dots,3\}}$ and $p = \{0.1, 0.2, \dots, 0.9\}$.

As mentioned earlier, both DropSVM and standard SVM are trained using LIBSVM implementation (Chang and Lin, 2011).

5.2 Experimental Result

Table 2 illustrates the performance of each algorithm in the standard UCI datasets and human action recognition experiments. In addition, we include the average percentage of the training samples used as support vectors (SVs) for each algorithm and the Dropout percentage p for each dataset.

It is clear that the exploitation of the proposed regularization form improves the classification results in all data sets. One interesting observation is that the percentage of Dropout value varies for each data set to gain the best regularizing performance. While some datasets need to drop a small number of training samples, some datasets gain the best performance with 70% to 90% Dropout such as Australian, Column2c, Indians, Ionosphere, Spectf. In addition, in most cases, the number of support vectors used in DropSVM increases significantly. This further illustrates that the decision boundary governed by more training samples yields better results when being appropriately controlled.

Table 2. Experiment results for UCI datasets

Data set	SVM	DropSVM
Australian	86.64 (SVs 34%)	**86.99** (p=0.8, SVs 80%)
Column2c	85.68 (SVs 16%)	**86.39** (p=0.7, SVs 60%)
Column3c	85.74 (SVs 12%)	**86.39** (p=0.6, SVs 10%)
German	76.48 (SVs 25%)	**76.78** (p=0.6, SVs 80%)
Glass	69.07 (SVs 12%)	**70.28** (p=0.2, SVs 30%)
Heart	84.67 (SVs 21%)	**85.19** (p=0.5, SVs 60%)
Indians	77.58 (SVs 28%)	**77.60** (p=0.9, SVs 50%)
Ionosphere	94.53 (SVs 9%)	**94.87** (p=0.9, SVs 60%)
Relax	71.43 (SVs 36%)	**71.76** (p=0.3, SVs 10%)
Spect	82.32 (SVs 19%)	**84.12** (p=0.5, SVs 40%)
Spectf	79.93 (SVs 21%)	**81.12** (p=0.8, SVs 70%)
Syn. Con.	98.30 (SVs 4%)	**98.77** (p=0.9, SVs 90%)
Hollywood2	61.41 (SVs 53%)	**62.31** (p=0.4, SVs 82%)
Hollywood3D	29.45 (SVs 17%)	**30.52** (p=0.8, SVs 32%)
Olympics	82.77 (SVs 30%)	**83.99** (p=0.7, SVs 54%)

Moreover, we speculate that the variation in the Dropout percentage can be explained based on the percentage of the training data used as support vectors in the standard SVM model as follows: for the standard SVM classifier with high percentage of support vectors, the amount of Dropout percentage and epochs needed in order to have some support vectors dropped is small and vice versa. To illustrate this reasoning, we train standard SVM classifier with the regularization parameter $c = c_{best}$ which achieves the best result for DropSVM model on each data set. The percent of support vectors (SVs) in standard SVM model, the dropout percentage (p) and the number of epoch (E) that achieves the best result with $c = c_{best}$ for DropSVM classifier are presented in Table 3. To verify the reasoning, the correlation coefficient between SVs, the percent of support vector in standard SVM model, and the product pE, which represents the chance that a support vector is dropped in DropSVM, is calculated. This correlation value is equal to -0.613, which verifies the speculation to some extent.

Table 3. Support vectors (SVs) in SVM and Dropout settings in DropSVM

Data set	SVs (%)	p (%)	E
Australian	34	80	10
Column2c	16	60	10
Column3c	12	10	25
German	25	80	25
Glass	12	30	50
Heart	21	60	25
Indians	28	50	50
Ionosphere	09	60	100
Relax	36	10	10
Spect	19	40	100
Spectf	21	70	10
Syn. Con.	04	90	100

6 CONCLUSION

In this thesis, we have proposed a new regularization formulation for the support vector classifier. Based on the insight given by Representer Theorem that the weight vector w lies in the subspace spanned by the training samples, the new formulation incorporates probabilistic Dropout

scheme at the sample level. As pointed out in the discussion chapter, the new method is supported by a clear motivation and geometric interpretation of the SVM classifier and conforms to the importance result derived in statistical learning theory. In addition to an elegant solution and guarantee on global optimality, the new regularizer can be incorporated into other linear models as suggested in this thesis.

The experiments conducted on a variety of classification problems showed consistent improvement of the new regularizer over the classic SVM. Based on the support vector information and the Dropout settings that gained the best accuracy, we pointed out a relationship between these information: the more support vectors used in SVM, the lower amount of Dropout needed to have some support vectors dropped in order to improve accuracy.

Further work utilizing the new regularizer can be conducted to other linear models such as spectral regression and linear discriminant analysis. We believe it is an interesting path to be pursued in the future.

REFERENCES

Hinton, G., Srivastava, N., Krizhevsky, A., Sutsekever, I., Salakhutdinov, R. (2011). *Improving neural networks by preventing co-adaptation of feature detectors*. Retrieved from https://arxiv.org/abs/1207.0580

van der Maaten, L., Chen M., Tyree S., and Weinberger K. (2013). *Learning with marginalized corrupted features*. International Conference in Machine Learning

Wager S., Wang S., and Liang P. (2013). *Dropout training as adaptive regularization*. Neural Information Processing Systems.

Wang S., Wang M., Wager S., Liang P., and Manning C. (2013). *Feature noising for log-linear structured prediction*. Empirical Methods on Natural Language Processing.

Burges C., Scholkopf ,B. (1997). *Improving the accuracy and speed of support vector machines*. Neural Information Processing Systems.

Dekel O., Shamir O., (2008). *Learning to classify with missing and corrupted features*. International Conference in Machine Learning.

Globerson A., Roweis S., (2006). *Nightmare at test time: Robust learning by feature deletion*. International Conference in Machine Learning.

Teo C., Globerson A., Roweis S., Smola A. (2008) *Convex learning with invariances*. Neural Information Processing Systems.

Chen N., Zhu J., Chen J., Zhang B., (2014). *Dropout training for support vector machines*. Twenty-Eighth AAAI Conference on Artificial Intelligence.

Vapnik V. (1995). The nature of statistical learning theory. Springer-Verlag.

Muller K., Mika S., Ratsch G., Tsuda K., Scholkopf B., (2001). *An introduction to kernel-based learning algorithms*. IEEE Transactions on Neural Networks 12, 181–201.

Scholkpf B., Smola A., (2001). *Learning with Kernels*. MIT Press.

Chang C., Lin C., (2011) LIBSVM: A library for support vector machines. *ACM Transaction Intelligent Systems Technology.*

Hastie T., Tibshirani R., Friedman J., (2001) *The Elements of Statistical Learning; Data Mining, Inference and Prediction*. Springer Verlag, New York.

Evgeniou T, (2002). Computational Statistics & Data Analysis (38).

Lichman M., (2013). *UCI Machine Learning Repository Irvine*, CA: *University of California, School of Information and Computer Science*. Retrieved from: http://archive.ics.uci.edu/ml

Vapnik, V., Chervonenkis, A., (1971). *On the Uniform Convergence of Relative Frequencies of events to their probabilities*. Theory Probability Application 17 (2), 264 – 280.

Kearns, M., Shapire, R., (1994). *Efficient distribution-free learning of probabilisitic concepts*. Journal of Computer and System Sciece 48 (3), 464 – 497.

Marszalek, M., Laptev, I., Schmid, C., (2009). *Actions in context. Computer Vision and Pattern Recognition* , 2929–2936.

Niebles, J., Chend, C., Fei-Fei, L., (2010). *Modeling temporal structure of decomposable mition segemnts for activity classification*. European Conference on Computer Vision , 392–405.

Hadfield, S., Bowden, R., (2013). *Hollywood 3D: Recognizing actions in 3D natural scenes*. Computer Vision and Pattern Recognition , 3398–3405.

Wang, H., Schmid, (2013). *Action recognition with improved trajectories. International Conference on Computer Vision* , 3551–3558.

Laptev, I., Marszalek, M., Schmid, C., Rozenfeld, B., (2008). *Learning realistic human actions from movies*. Computer Vision and Pattern Recognition, 18.

Fletcher R., (1981). *Practical Methods of Optimization: Volume 2 Constrained Optimization*. New York: Wiley.

Evgeniou, T., Pontil, M., Poggio, T., (1999). *A unified framework for Regularization Networks and Support Vector Machines*. A.I. Memo No. 1654, Artificial Intelligence Laboratory, Massachusetts Institute of Technology.