Support Vector Machines: Theoretical Summary Experiments

Joseph Hilland, *Student, UNM ECE*

***Abstract*—This document describes introductory theory behind concepts associated with machine learning such as risk, complexity and features in the context of Support Vector Machines (SVM). Topics introduced and analyzed using experiments in this paper include Vapnik-Chervonenkis (VC) dimension and SVM.**

***Index Terms*—Support Vector Machines, Vapnik-Chervonenkis dimension, Overfitting, Statistical Theory, Complexity.**

I. Introduction

THIS paper will attempt to introduce the concepts of SVMs, a type of supervised learning algorithm that is utilized for classification types within machine learning. The paper will introduce practical applications as well as demonstrate via experiments the various concepts that have been introduced.

The theoretical and experiment portions of this paper will only briefly cover the key concepts of SVMS. These concepts include but are not limited to types of Risk, complexity, VC theory and dimension, SVM Criteria, dual solution of the SVM and support vectors. The experiments introduced in this paper have been performed in python. The python build system is poetry, which will not be introduced in this paper. And finally, the scikit-learn python module was utilized in the SVM experiments.

The fundamental idea behind SVMs is best explained with pictures. Figure 1 shows a linear data set of two classes that can be linearly separated with a straight line. Figure 2 represents data of non-linear classification. This will require modifications to an SVM, and therefore will be out of the context of this paper. Instead, we will only focus on the theory and experiments of linear datasets for classification with SVM.

SVM algorithms goal is to find a hyperplane that creates a boundary between the types of data samples. In 2-dimensional space, this optimal hyperplane is merely a linear line separating out the data points. Support vectors, support hyperplanes and margins are all utilized and related to the concept of the optimal hyperplane. SVMs only solve binary problems, as multidimensional classification is out of the scope of this paper. Therefore, the SVM algorithms, theory and examples in this paper will only cover two classes of data.

SVMs can be utilized for both classification and regression, this paper will primarily investigate the classification capabilities of SVMs. SVMs work best when the dataset being classified is small and complex. This data must be linearly separable, otherwise an optimal hyperplane could not be utilized for separating the data into classes for classification.

Manuscript received April 19, 2021; revised August 16, 2021.

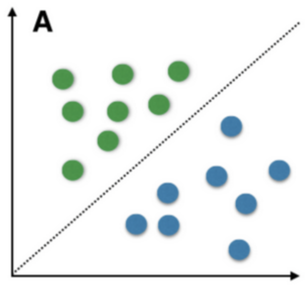


Fig. 1. Linear dataset, no SVM modifications requried

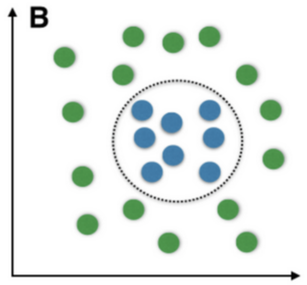


Fig. 2. Non-linear dataset, more SVM modifications required

II. Thoery

A. Risk: Empirical, Actual and Structural

In the context of machine learning, empirical risk and risk refer to the measure of how well the algorithm model has fit the input training dataset. That is, the average loss that is incurred by a model on a training dataset. Structural risk within the context of support vector machines refers to the fitting of the training data versus keeping the model simple to generalize data that is yet to be seen. This ability for the model to prevent overfitting is crucial. Actual risk is determined by computing the probability of error during a test. Actual risk is determined by utilizing the empirical and structural risk.

Using these concepts of risk within the context of machine learning, one can determine how well a model performs. This includes determining if the model is too complex, in which case the model can underfit or overfit. Empirical risk is defined as the average loss of all the training samples.

B. Complexity and Overfitting

Complexity and Overfitting are closely related within the concepts of machine learning. Complexity refers to the sophistication of the model or algorithm. The more parameters within the model, or the more relationships it can represent between input and output then the more complex it is. Overfitting can occur within an SVM if a model is trained too well with training data. This can occur by capturing noise and classifying it even though it does not represent the true pattern. A model trained too well, that is too complex may memorize the training data rather than learn from the patterns. You can usually identify such models based on low empirical risk but high test errors.

C. VC Theory and Dimension

The VC dimension is used to measure the complexity of a model in the context of SVMs for machine learning. It was developed by Vladimir Vapnik and Alexey Chervonenkis during 1960-1990s. The VC dimension is determined after studying the capabilities of a model class with respect to shattering sets of points.

Equation 1 defines the empirical risk with the VC Dimension:

$$R_{emp}(\alpha) = \frac{1}{2N} \sum_{n=1}^{N} \big|\, y - f(\mathrm{x}, \alpha) \,\big| \tag{1}$$

where $f()$ is defined so that the loss function $\big|\, y - f(\mathrm{x}, \alpha) \,\big|$ can only take the values of 0 or 1. Then, with the probability of $1 - \eta$, the following bound holds:

$$R(\alpha) \le R_{emp}(\alpha) + \sqrt{\frac{h(\log(2N/h)+1) - \log(\eta/4)}{N}} \tag{2}$$

This is a bound on the risk with probability defined as $1 - \eta$. It is neither gauranteed nor dependent on the prability distribution. The right side can be easily computed if we are provided the knowedge of h. That is, where the second term of the right side is the structural risk (R_s). The inductive principle of structural risk minimization will then consist of choosing a machine where the dimension h is small, so that the bound on the risk is minimized.

D. SVM Criterion and Support Vectors

Criteria used within SVMs revolves around the optimal hyperplane. This hyperplane separates the classes or predicts the numerical values based on the training datasets. The hyperplane attempts to separate data points into two separate classes. For construction of the optimal hyperplane, two support hyperplanes are first identified. These support hyperplanes lie on the most extreme points between the two classes. The margin is the distance between these two support hyperplanes.

Support vectors are datapoints near hyperplanes that satisfy certain conditions. This is seen in support-vector machines. If a datapoint is well within the boundary (support hyperplane), the penalizing factor ξ_n is 0. Otherwise, if the datapoint is on the other side, this factor ξ_n is equal to its distance between the datapoint and the support hyperplane, i.e., $\xi_n \ge 0$ and $\alpha_n = C$. If a sample is on the margin, $0 < \alpha_n = C$. Finally, if a sample is outside the margin, $\xi_n = 0$ and $\alpha_n = 0$.

E. SVM Dual Solution, Results

For the dual solution, the empirical risk and structural risk needs to be minimized. This is done through te margin maximization which can be seen below:

$$\text{minimize } L_p(\mathrm{w}, \xi_n) = \frac{1}{2}\|\mathrm{w}\|^2 + \ C\sum_{n=1}^{N} \xi_n \tag{3}$$

$$\text{subject to } \begin{cases} y_n\left(\mathrm{w}^\top \mathrm{x}_n + b\right) > 1 - \xi_n \\ \xi_n \ge 0 \end{cases}$$

C is a free parameter, p denotes the primal and En is the distance where a datapoint exceeds the support hyperplane towards the other class.

The primal problem for an SVM is to find the optimal hyperplane. This hyperplane maximizes the margin between the classes, without losing the proper classification of the training data. We need to use Lagrange multipliers to change the constrained problem into an unconstrained one. Since there are 2N constraints, we need 1N multipliers.

Solving for w will result in the Karush-Kuhn-Tucker condition that creates a point over the domain of the criteria.

III. Experiments

A. Graphical Representation of an SVM for Classification

The experiment conducted involved creating an SVM for classification, and then plotting the SVM features for viewability and analysis. Python was the programming language chosen for these experiments

First step was to utilize python to generate 2D samples in four Gaussian clusters. These clusters were centered around the points listed in the assignment section above. To do this the numpy and matplotlib modules within python were utilized. Figure 3 shows the basic plot of the clusters, two colors were used to visualize the classes from one another.

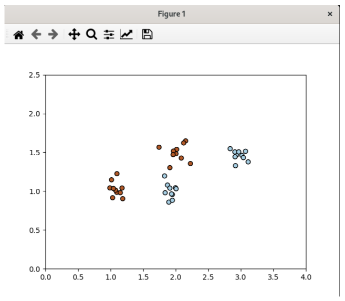
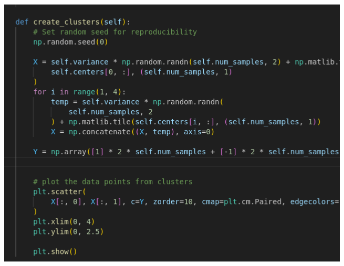


Fig. 3. Plotting Clusters

Figure 4 is the python code that was used to generate and plot these points, for reproducability, the points do not change on every run of the script.

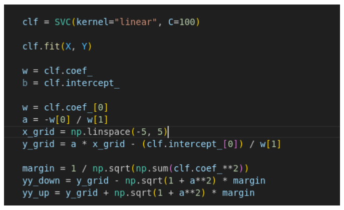
Next step was to train the Linear SVM. Scikit-learn module within python was used. This module from python is highly documented and supported, therefore it was chosen over libsvm. The code for training the SVM can be seen in figure 5.

Next, the lagrange multipliers and bias were extracted from the SVM model. These computed the primal parameters which



```
def create_clusters(self):
    # Set random seed for reproducibility
    np.random.seed(0)

    X = self.variance * np.random.randn(self.num_samples, 2) + np.matlib.
        self.centers[0, :], (self.num_samples, 1)
    )
    for i in range(1, 4):
        temp = self.variance * np.random.randn(
            self.num_samples, 2
        ) + np.matlib.tile(self.centers[i, :], (self.num_samples, 1))
        X = np.concatenate((X, temp), axis=0)

    Y = np.array([1] * 2 * self.num_samples + [-1] * 2 * self.num_samples


    # plot the data points from clusters
    plt.scatter(
        X[:, 0], X[:, 1], c=Y, zorder=10, cmap=plt.cm.Paired, edgecolors=
    )
    plt.xlim(0, 4)
    plt.ylim(0, 2.5)

    plt.show()
```

Fig. 4. Cluster creation code



```
clf = SVC(kernel="linear", C=100)

clf.fit(X, Y)

w = clf.coef_
b = clf.intercept_

w = clf.coef_[0]
a = -w[0] / w[1]
x_grid = np.linspace(-5, 5)
y_grid = a * x_grid - (clf.intercept_[0]) / w[1]

margin = 1 / np.sqrt(np.sum(clf.coef_**2))
yy_down = y_grid - np.sqrt(1 + a**2) * margin
yy_up = y_grid + np.sqrt(1 + a**2) * margin
```

Fig. 5. SVM Training Code, Scikit-learn

led to the separated visualized lines. These are the margin and support vectors. Figure 6 shows the support vectors along with the hyperplanes and optimal hyperplane. The support hyperplanes are dashed lines. Support vectors can be seen with a black enlarged circle around their color classifier.

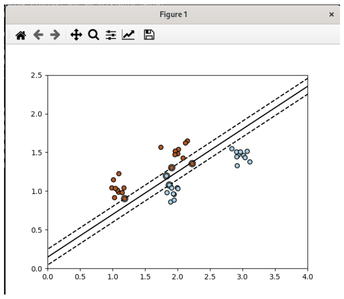


Fig. 6. Trained SVM with Support Vectors

B. Graphical representation of Risk

This exercise is meant to draw a plot of the actual, empirical and structural risk. These graphs were previously shown in the theoretical section. The given script will be utilized to generate a set of binary labeled data in space of 10 dimensions that can be classified with a linear classifier in this space. The plot below depicts the results of the linear classifier.

From figure 7, the green line represents the Empirical risk, the blue line represents the Actual risk followed by the Structural risk in red.

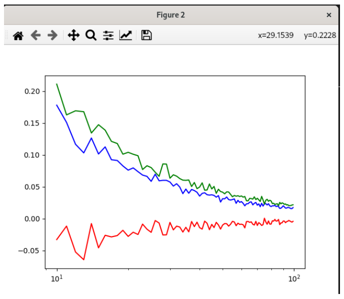


Fig. 7. Empirical, Actual, Structural Risk

IV. Discussion

This paper was written to describe the fundamental concepts behind risk, complexity, VC dimension, SVM features in the context of machine learning. The theory was outlined so that it could be utilized when conducting and analyzing the experiments. The experiments were conducted in python, which is highly documented and supported with respect to SVM classification. It is a highly portable and powerful computer programming language.

V. Acknowledgements

The author would like to thank the University of New Mexico for the ability to conduct research on SVMs in the context of machine learning.

References

[1] *H. Wang, and D. Hu*. Comparison of SVM and LS-SVM for regression. [2005]. Available: https://doi.org/10.1109/icnnb.2005.1614615

[2] *Evgeny Byvatov, Uli Fechner, Jens Sadowski, and Gisbert Schneider*. Comparison of Support Vector Machine and Artificial Neural Network Systems for Drug/Nondrug Classification. [2003]. Available: https://pubs.acs.org/doi/abs/10.1021/ci0341161

[3] *Achmad Widodo, and Bo-Suk Yang*. Support vector machine in machine condition monitoring and fault diagnosis. [2006]. Available: https://www.sciencedirect.com/science/article/abs/pii/S0888327007000027

[4] *Andreas Christmann, and Ingo Steinwart* .Support Vector Machines. [2008]. Available: https://link.springer.com/book/10.1007/978-0-387-77242-4