# Machine Learning Assignment 3

**Shenggan Cheng**
Shanghai Jiao Tong University
chengshenggan@sjtu.edu.cn

## Abstract

In this paper we evaluate the performance of Support Vector Machine (SVM) and Multi-Layer Perceptron (MLP) on different datasets. Then we apply SVM on CIFAR-10 datasets and compared with the state-of-art deep learning method. Finally, we will discuss the strengths and weaknesses of SVM on big data sets.

## 1 Introduction

Support Vector Machine (SVM) is a technique algorithms for pattern recognition which approximately implements *structural risk* minimization. Where some techniques, like Multi-Layer Perceptron (MLP), are based on the minimization of the *empirical risk*, that is the minimization of the number od misclassify points in the training datset, SVM minimize a function which is the sum of two terms. The first term us the empirical risk, the second term controls the structural risk. SVM is attracting many attention beacuse it has a solid mathematical foundation and perform effectively in many different applications. An obviously advantage of SVM is the intuitive interpretation and the understandable theory behind SVM. After training, the separating surface is expressed as a certain linear combination of a given kernel function centered at some of the data points, which named *support vectors*. All the remaining points if the training dataset are effectively discarded and the classification of new points is obtained solely in terms of the support vectors.

## 2 Theory

### 2.1 SVM

Let us briefly review the theory of SVMs.

We assume we are given a set $S$ of $N$ points $x_i \in IR^n (i = 1, 2, ..., N)$. Each point $x_i$ belongs to either of two classes which is $y_i \in \{-1, 1\}$. In the further assmption that the two classes can be linearly separable, the goal is to establish the hyperplane, that divides $S$ leaving all the points of the same class on the same side while maximizing the distance of the closest point. It can be shown that the hyperplane is the solution to the problem:

$$
\begin{aligned}
&Minimize\ \frac{1}{2} w\dot{w} \\
&subject\ to\ y_i(w\dot{x}_i + b) >= 1, i = 1, 2...N
\end{aligned}
$$

where $w$ is the normal of the hyperplane, and $b/w$ the distance of the hyperplane from the origin. If we denote with $\alpha = (\alpha_1, \alpha_2, ..., \alpha_N)$ the $N$ nonnegtive Lagrange multipliers associated with the constraints, the solution to this problem is equivalent to determining the solution to the dual problem.

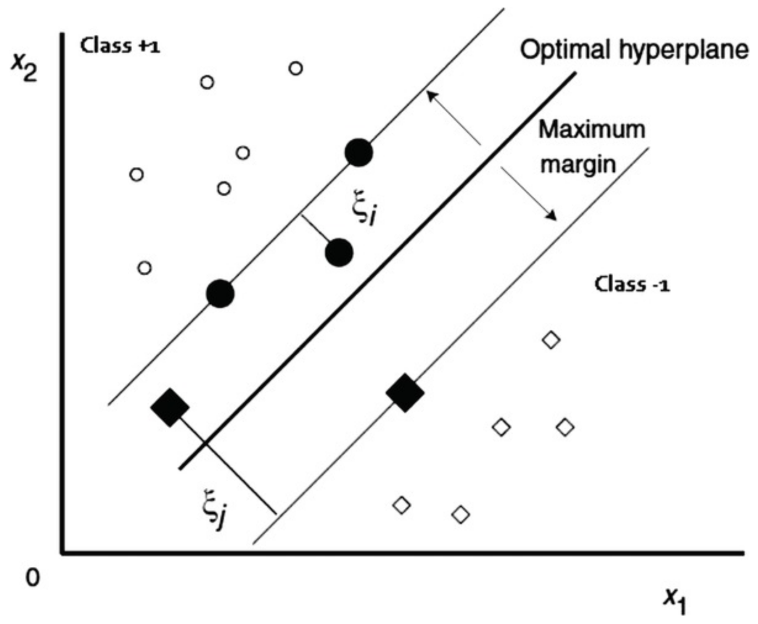

Figure 1: SVM

$$\textit{Minimize } \frac{1}{2}\alpha^T D\alpha + \sum \alpha_i$$
$$\textit{subject to } \sum y_i\alpha_i = 0, \alpha >= 0$$

where the sums are for $i = 1, 2, ..., N$, and $D$ is an $NxN$ matrix such that

$$D_{ij} = y_i y_j x_i \dot{x}_j$$

The solution for $\overline{w}$

$$\overline{w} = \sum_{i=1}^{N} \overline{\alpha}_i y_i x_i$$

The only $\overline{\alpha}_i$ that can be nonzero in 2.1 are those for which the constraints of the first problem are satisfied with the equality sign. Since most of the $\overline{\alpha}_i$ are usually null, the vector $\overline{w}$ is a linear combination of a often relatively small percentage of the points $x_i$. These points are termed *support vectors* because they are the only points of S needed to determine the hyperplane.

The problem of classifying a new data point $x$ is now simply solved by looking at the sign of

$$\overline{x}\dot{x} + \overline{b}$$

with $\overline{b}$ obtained from the Khun Tucker Conditions.

If the set $S$ cannot be separated by a hyperplane, the previous analysis can be generalized by introducing $N$ nonnegative variables $\xi = (\xi_1, \xi_2, .., \xi_N)$ such that

$$y_i(w\dot{x}_i + b) >= 1 - \xi_i, i = 1, 2, ..., N$$

The so called generalized hyperplane is then regarded as the solution to

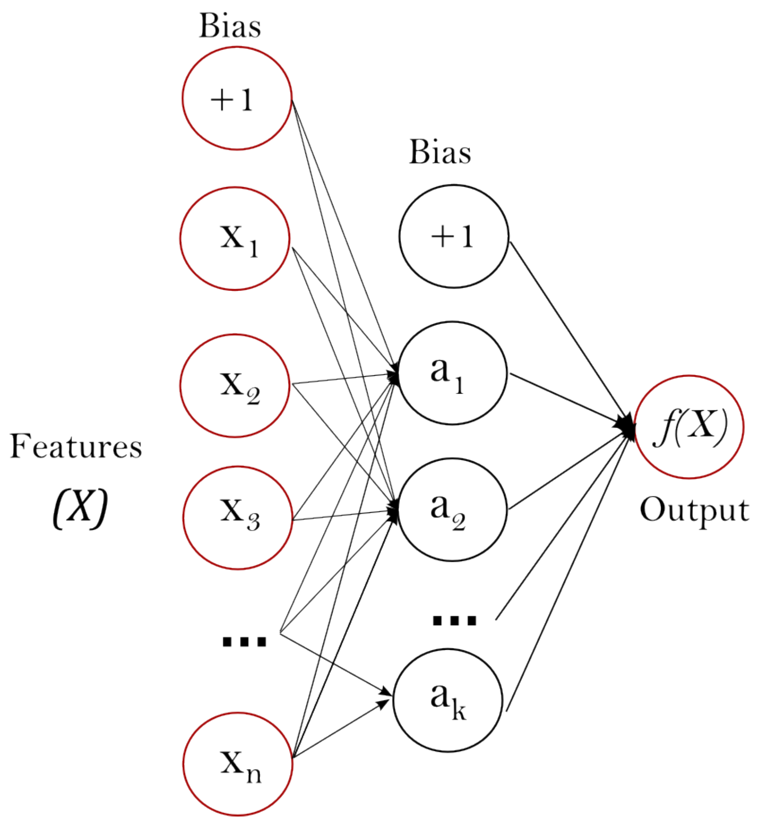

Figure 2: One hidden layer MLP

$$Minimize \ \frac{1}{2} w\dot{w} + C \sum \xi_i$$
$$subject\ to\ y_i(w\dot{x}_i + b) >= 1 - \xi_i, i = 1, 2, .., N, \xi >= 0$$

Similarly to the linearly separable case, the dual formulation requires the solution of a quadratic programming problem with linear constraints. Once again it turns out that the points that satisfy the constraints of above with the equality sign are termed support vectors and are the only points needed to determine the decision surface

The entire construction can also be extended rather naturally to include nonlinear separating surfaces. Each point $x$ in input space is mapped into a point $z = \phi(x)$ of a higher dimensional feature space (possibly of in

nite dimension). The mapping $\phi$ is subject to the condition that the dot product $< \phi(x), \phi(y) >$ in feature space can be rewritten through a kernel function $K = K(x, y)$. Admissible kernel functions, for example, are the polynomial kernel of n-th degree.

$$K(x, y) = (1 + x\dot{y})^n - 1$$

## 2.2 MLP

Multi-layer Perceptron (MLP) is a supervised learning algorithm that learns a function $f(\dot{)} : R^m - > R^0$ by training on a dataset, where $m$ is the number of dimensions for input and $o$ is the number of dimensions for output. Given a set of features $X = x_1, x_2, .., x_m$ and a target $y$, it can learn a non-linear function approximator for either classification. It is different from logistic regression, in that between the input and the output layer, there can be one or more non-linear layers, called hidden layers. Figure 2 shows a one hidden layer MLP with scalar output.

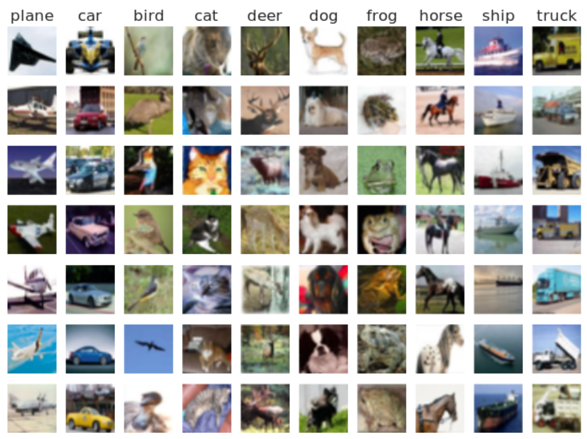

Figure 3: Overview of CIFAR-10 Dataset

The leftmost layer, known as the input layer, consists of a set of neurons $\{x_i|x_1, x_2, .., x_m\}$ representing the input features. Each neuron in the hidden layer transforms the values from the previous layer with a weighted linear summation $w_1x_1 + w_2x_2 + ... + w_mx_m$ followed by a non-linear activation function $g(\dot{}) : R- > R-$ like the hyperbolic tan function. The output layer receives the values from the last hidden layer and transforms them into output values.

## 3 Data Description

In this report, we use three different dataset for two experiments. In the first, we choose two datasets from `https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/` which are *phishing* and *svmguide1*. *phishing* and *svmguide1* is used for experiments one. And for experiments two, we use CIFAR-10 from `http://www.cs.utoronto.ca/~kriz/cifar.html`.

### 3.1 phishing and svmguide1

This two datasets are from LIBSVM Data which contains many classification, regression, multi-label and string data sets stored in LIBSVM format. Many are from UCI, Statlog, StatLib and other collections.

Phishing dataset is from UCI/Phishing Websites. All features are categorical. Is uses binary encoding to generate feature vectors. Each feature vector is normalized to maintain unit-length. And it contain 2 classes and 11,055 samples with 68 features.

Svmguide1 dataset is from CWH03a. Original data is an astroparticle application from Jan Conrad of Uppsala University, Sweden. And it contain 2 classes and 3089(Training)/4000(Test) samples with 4 features.

### 3.2 CIFAR-10

The CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images.

The dataset is divided into five training batches and one test batch, each with 10000 images. The test batch contains exactly 1000 randomly-selected images from each class. The training batches contain the remaining images in random order, but some training batches may contain more images from one class than another. Between them, the training batches contain exactly 5000 images from each class.

Here are the classes in the dataset, as well as 10 random images from each:

Computer algorithms for recognizing objects in photos often learn by example. CIFAR-10 is a set of images that can be used to teach a computer how to recognize objects. Since the images in CIFAR-10 are low-resolution (32x32), this dataset can allow researchers to quickly try different algorithms to see what works. Various kinds of convolutional neural networks tend to be the best at recognizing the images in CIFAR-10.

## 4 Experiment 1

In this experiment, we will use *phishing* and *svmguide1* as our dataset to evaluate the performance of SVM and MLP. And we will analysis the two algorithms' performance in this two dataset.

### 4.1 Detail of Experiment

#### 4.1.1 Tools

We choose *scikit-learn* as our tools to implements the experiments. *Scikit-learn* is a very popular machine learning tools for data mining and data analysis. For loading the dataset in *libsvm* format, *scikit-learn* provides the API named *load_svmlight_file* which can load the dataset in only one line. For SVM and MLP, we will use *sklearn.svm.SVC* and *sklearn.neural_network.MLPClassifier* as our model.

#### 4.1.2 GridSearchCV

In order to find the best experimental settings, We will try different settings including the size of the training set, different dimensions of datasets and hyperparameters. We will use *GridSearchCV* provided by *scikit-learn*. *GridSearchCV* will exhaustively search over specified parameter values for an estimator.

Hyperparameters in SVM include Penalty parameter $C$, kernel and gamma which represent the kernel coefficient. Hyperparameters in MLP include hidden_layer_sizes, learning_rate, solver and alpha(L2 penalty parameter).

#### 4.1.3 Metric

*Accuracy_score* and *confusion_matrix* is two import metric function in *scikit-learn*. We will use this two function as our metric.

*Accuracy_score* compute accuracy classification score. In multilabel classification, this function computes subset accuracy: the set of labels predicted for a sample must exactly match the corresponding set of labels in *y_true*.

*Confusion_matrix* compute confusion matrix to evaluate the accuracy of a classification. By definition a confusion matrix $C$ is such that $C_{i,j}$ is equal to the number of observations known to be in group $i$ but predicted to be in group $j$. Thus in binary classification, the count of true negatives is $C_{0,0}$, false negatives is $C_{1,0}$, true positives is $C_{1,1}$ and false positives is $C_{0,1}$.

### 4.2 Results

#### 4.2.1 Size of Training set and Data Dimension

First we will design some experiments to demostrate the importance of dataset size and data dimension. Below is the result of the experiments. In these experiments, we will freeze the hyperparameters with the default value and vary the size of training set and data dimension.

In *phishing* and *svmguide1* dataset, we use 10%, 50%, and 100% of the whole training set, and use 50%, 100% of the feature dimensions. The results are shown in 1 and 2. So we can find out that the more data and dimension of dataset, the higher performance will be get. So in all experiment below will use the full datasets for training and testing.

Table 1: Different Size of Training Set

| | Traing Set | SVM | MLP |
|---|---|---|---|
| Phishing | 10% | 0.896 | 0.9272 |
| | 50% | 0.9159 | 0.9353 |
| | 100% | 0.9181 | 0.9462 |
| svmguid1 | 10% | 0.6667 | 0.7184 |
| | 50% | 0.7201 | 0.9158 |
| | 100% | 0.775 | 0.9515 |

Table 2: Different Dimension of Data

| | Feature Dimension | SVM | MLP |
|---|---|---|---|
| Phishing | 50% | 0.8982 | 0.9408 |
| | 100% | 0.9181 | 0.9461 |

#### 4.2.2 Hyperparameters

We will use *GridSearchCV* provided by *scikit-learn*. *GridSearchCV* will exhaustively search over specified parameter values for an estimator. And *GridSearchCV* will return the best setting of hyperparameters. The parameter list provided is shown in 3 and 4. And the results of *GridSearchCV* is shown in 6, 7 and 8.

### 4.3 Conclusion

So we can get the results of SVM and MLP. In *phishing*, SVM gets 94.39% accuracy while MLP gets 96.16% accrucy. In *svmguid1*, SVM gets 77.51% accuracy while MLP gets 96.12% accrucy. SVM gets poor performance in *svmguid1*, so we belive that we must do some preprocess with dataset. With normalized dataset, SVM can get 96.15%. So we can conclude that the preprocess of dataset is very important.

## 5 Experiment 2

In this experiment, we will use CIFAR-10 as our dataset to evaluate the performance od SVM. And because the sample of CIFAR-10 is image, so we will flat the RGB value to get the feature. So feature shape is ($32 \times 32 \times 3 = 3072$), this is too large as feature for SVM. We will give two approach to reduce the dimension of feature in below section. And we will compare with the experiment without dimensional reduction.

### 5.1 Detail of Experiment

#### 5.1.1 PCA

Principal component analysis (PCA), is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components. If there are $n$ n observations with $p$ variables, then the number of distinct principal components is $min(n-1, p)$. This transformation is defined in such a way that the first principal component has the largest possible variance (that is, accounts for as much of the variability in the data as possible), and each succeeding component in turn has the highest variance possible under the constraint that it is orthogonal to the preceding components. The resulting vectors are an uncorrelated orthogonal basis set. PCA is sensitive to the relative scaling of the original variables.

Table 3: The Parameters of SVM

| | |
|---|---|
| kernel | 'linear', 'rbf', 'poly' |
| C | 1, 10, 100, 1000" |
| gamma | 1e-3, 1e-4, 'auto' |

Table 4: The Parameters of MLP

| | |
|---|---|
| solver | 'lbfgs', 'sgd', 'adam' |
| hidden_layer_sizes | (100,), (10,20), (200,200), (100,200,100) |
| alpha | 1e-3 1e-4 |
| learning_rate | 'constant', 'invscaling', 'adaptive' |

Table 5: The Results of SVM on phishing

| | | |
|---|---|---|
| best_para | C: 1000 gamma: auto kernel: rbf | |
| accrucy | 0.9439 | |
| Confusion Matrix | 913 | 57 |
| | 67 | 1174 |

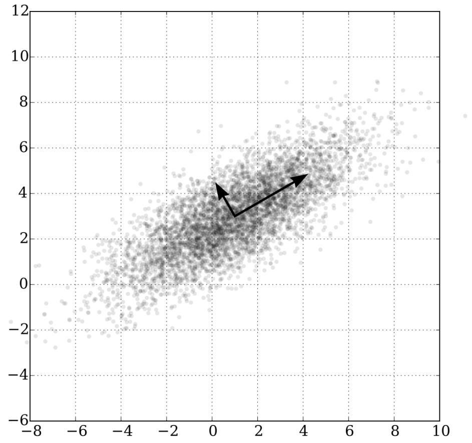

Figure 4: Overview of PCA

And we will use *sklearn.decomposition.PCA* as our implements of PCA. The API*sklearn.decomposition.PCA* uses *n_components* to control number of components to keep.

### 5.1.2 HOG

The histogram of oriented gradients (HOG) is a feature descriptor used in computer vision and image processing for the purpose of object detection. The technique counts occurrences of gradient orientation in localized portions of an image. This method is similar to that of edge orientation histograms, scale-invariant feature transform descriptors, and shape contexts, but differs in that it is computed on a dense grid of uniformly spaced cells and uses overlapping local contrast normalization for improved accuracy.

And we will use *skimage.feature.hog* to compute the HOG feature of every image in CIFAR-10. And put the HOG feature as the input of SVM classifier.

Table 6: The Results of SVM on svmguide1

| | | |
|---|---|---|
| best_para | C: 1 gamma: auto kernel: rbf | |
| accrucy | 0.7751 | |
| Confusion Matrix | 978 | 2 |
| | 137 | 401 |

Table 7: The Results of MLP on phishing

| | | |
|---|---|---|
| best_para | alpha: 0.0001, hidden_layer_sizes: (100, 200, 100), learning_rate: ’adaptive’, solver: ’adam’ | |
| accrucy | 0.9616 | |
| Confusion Matrix | 960 | 65 |
| | 20 | 1166 |

Table 8: The Results of MLP on svmguide1

| | | |
|---|---|---|
| best_para | alpha: 0.001, hidden_layer_sizes: (100,), learning_rate: ’constant’, solver: ’lbfgs’ | |
| accrucy | 0.9612 | |
| Confusion Matrix | 200 | 9 |
| | 15 | 394 |

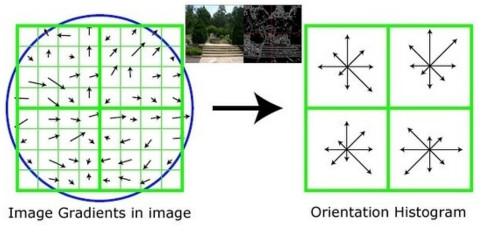

Figure 5: Overview of HOG

Table 9: Results on CIFAR-10 Dataset

| Model | SVM with PCA | SVM with HOG | NiN(2014) | VGG(2014) | ResNet(2015) |
|---|---|---|---|---|---|
| Acc | 52.47 | 46.55 | 91.2 | 92.4 | 93.57 |

### 5.2 Results

According to `http://rodrigob.github.io/are_we_there_yet/build/classification_datasets_results.html` we can find the performance of some state-of-art models. We can find out almost models are CNN-based Deep Learning method. And our SVM results and some selected deep learning benchmark are shown in 9.

### 5.3 Conclusion

There are four main advantages: Firstly it has a regularisation parameter, which makes the user think about avoiding over-fitting. Secondly it uses the kernel trick, so you can build in expert knowledge about the problem via engineering the kernel. Thirdly an SVM is defined by a convex optimisation problem (no local minima) for which there are efficient methods (e.g. SMO). Lastly, it is an approximation to a bound on the test error rate, and there is a substantial body of theory behind it which suggests it should be a good idea.

The disadvantages are that the theory only really covers the determination of the parameters for a given value of the regularisation and kernel parameters and choice of kernel. In a way the SVM moves the problem of over-fitting from optimising the parameters to model selection. Sadly kernel models can be quite sensitive to over-fitting the model selection criterion.

Within these series of experiments, we can find that the SVM is very slow on the big dataset, and the performance of SVM on big dataset is much worse than the deep learning methods. But we can do some dimensional reduction to imporve the performance of SVM on big dataset.