NEURAL NETWORK

Project 1 – SVM

Arunava Manna

A0232541R

Arunava.manna@u.nus.edu

EE5904

Project 1: SVM for Classification of Spam Email Messages

Contents

Data Pre - Processing

Real world data contains many inconsistencies resulted from human error, formatting error which can lead to error in model prediction. Since model learns a specific pattern of the data and assumes other data will follow the same pattern. For that reason, I have used the below approach to standardize both the train and test data before use it further.

$$New\ data_{train} = \frac{(data_{train} - mean(data_{train}))}{standard\ deviation(data_{train})}$$

$$New\ data_{test} = \frac{(data_{test} - mean(data_{train}))}{standard\ deviation(data_{train})}$$

Non-linear kernels show degradation in the performance if the data is not standardized. But if the data is sparse or data has more zero elements then the standardization can not be used. For these types of data min-max normalization must be used. If standardization is applied on the sparse data, then the sparse pattern will be destroyed.

$$New\ data_{train} = \frac{(data_{train} - max(data_{train}))}{max(data_{train}) - min(data_{train})}$$

$$New\ data_{train} = \frac{(data_{test} - max(data_{train}))}{max(data_{train}) - min(data_{train})}$$

We will review results using both standardization and min-max normalization.

Mercer's condition check

For training set S = $\{(x_i, d_i)\}$, i = 1,2, …, N

$$\text{Gram Matrix: K} = \left(\begin{bmatrix} K(x_1,x_1) & \cdots & K(x_1,x_N) \\ \vdots & \ddots & \vdots \\ K(x_N,x_1) & \cdots & K(x_N,x_N) \end{bmatrix}\right) \epsilon\ R^{N\times N}$$

Is positive semi-definite that means the eigenvalues are nonnegative. Then a function can be used as kernel.

For every function for example, the linear and polynomial hard margin function or the polynomial soft margin, the mercer condition validation is important for checking the availability of the kernel. Initially, the threshold was decided as 1e-4, but later the threshold was changed based on the experiment.

Quadprog Toolbox

For solving quadratic objective functions with linear constraint, MATLAB function "*quadprog*" has been used to find a minimum for a problem such as

$$\min_{x} \frac{1}{2} x^T H x + f^{T_x} \text{ such that } \begin{cases} A \cdot x \le b \\ Aeq \cdot x = beq \\ lb \le x \le ub \end{cases}$$

Where *H, A*, and *Aeq* are matrices, and *f, b, beq, lb, ub*, and *x* are vectors.

And Finally, x = *quadprog*(*H, f, A, b, Aeq, beq, lb, ub, x0, options*)

Task 1

- For hard margin $0 \leq \alpha_i$

$c = +\infty$ (in theory)

And $0 \leq \alpha_i \leq c$

For soft margin $0 \leq \alpha_i \leq c$

C = 0.1, 0.6, 1.1, 2.1

A hard margin SVM with the linear kernel

Kernel function: $k(x_1, x_2) = x_1^T x_2$

Dual problem of hard margin linear kernel SVM is shown as

$$\text{Maximize: } Q(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j d_1 d_j k(x_1, x_2)$$

Subject to

1. $\sum_{i=1}^{N} \alpha_i d_i = 0$

2. $\alpha_i \geq 0$

Hence, the parameters of the *QUADPROG* function are,

$H_{ij} = d_i d_j k(x_i, x_j) = d_1 d_j x_i^T x_j$

$f = -ones(2000,1)$

Aeq = trainLabel'

Beq = 0

lb = *zeros*(2000, 1)

ub = *ones*(2000, 1) * C

x0 = []

options =optimset('LargeScale', 'Maxiter', 1000)

$C = 10^6$

Hard Margin SVM with a polynomial kernel

Kernel function: $k(x_i, x_j) = (x_i^T x_j + 1)^p$ where p = 1,2,3,4,5

Dual problem of hard margin polynomial SVM is shown below:

$$Q(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j d_1 d_j k(x_1, x_2)$$

Subject to 1. $\sum_{i=1}^{N} \alpha_i d_i = 0$

2. $\alpha_i \geq 0$

Hence, the parameters of the *quadprog* function are,

$H_{ij} = d_i d_j k(x_i, x_j) = d_1 d_j (x_i^T x_j + 1)^p$

$f = -ones(2000,1)$

Aeq = trainLabel'

Beq = 0

lb = *zeros*(2000, 1)

ub = *ones*(2000, 1) * C

x0 = []

options =optimset('LargeScale', 'Maxiter', 1000)

C = 10^6

Soft margin SVM with a polynomial kernel

Kernel Function: $k(x_i, x_j) = (x_i^T x_j + 1)^p$ where p = 1,2,3,4,5

Dual problem of hard margin polynomial SVM is shown below:

$$Q(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j d_1 d_j k(x_1, x_2)$$

Subject to 1. $\sum_{i=1}^{N} \alpha_i d_i = 0$

2. $C \geq \alpha_i \geq 0$

Hence, the parameters of the *quadprog* function are,

$$H_{ij} = d_i d_j k(x_i, x_j) = d_1 d_j (x_i^T x_j + 1)^p$$

$$f = -ones(2000,1)$$

Aeq = trainLabel’

Beq = 0

lb = $zeros$(2000, 1)

ub = $ones$(2000, 1) * C

x0 = []

options =optimset(‘LargeScale’, ‘Maxiter’, 1000)

C = [0.1, 0.6, 1.1, 2.1]

Task 2

Calculation of W and b

$$w_0 = \sum_{i=1}^{N} \alpha_{0,i} d_i x_i$$

$b_0 = \dfrac{\sum_{i=1}^{m} \frac{1}{d_i} - w_0^T x_i}{m}$ where m is the total number of x_i with 0 < $\alpha_i \leq C$

The bias value is the mean of all the biases from the support vectors.

Deciding the test image label

$$g(x_{test}) = \sum_{i=1}^{N} \alpha_{0,i} d_i k(x_i, x_{test}) + b_0$$

if $g(x_{test})$ > 0, then the label of x_{test} = +1, otherwise x_{test} =-1

for checking the mercer condition, the below tolerance level has been used to consider more flexibility for checking if the positiveness of the eigen values.

```
tolerance = length(eigenvalues)*eps(max(eigenvalues))
```

✓ Motivation of selecting the above tolerance:
As default, 0 is considered to be the decision tolerance whether a matrix is positive definite or not. At the beginning, the mercer condition was failing for most of the p values with the 0 tolerance since the eigen values are of very small magnitude. Hence, the MathWorks website shows the above way of deciding custom tolerance which solves the problem of considering small eigen values. In this way the value of tolerance will change depending on the polynomial.
It can be noticed that the data is 77% sparse and the SVM result from both normalization and standardization can be observed below.

Table 1: With standardization:

Type Of SVM	Training Accuracy				Test Accuracy			
Hard Margin with Linear Kernel	93.55				93.8151			
Hard Margin with polynomial kernel	P = 2	P = 3	P = 4	P = 5	P = 2	P = 3	P = 4	P = 5
	99.8	100	75.05	77.95	86.6849	86.7839	72.3958	78.5807
Soft Margin with polynomial kernel	C = 0.1	C = 0.6	C = 1.1	C = 2.1	C = 0.1	C = 0.6	C = 1.1	C = 2.1
P=1	93	93.25	93.35	93.35	93.6198	93.4896	93.6849	93.6849
P=2	98.7	99.2	99.35	99.5	90.2344	88.4115	88.0208	88.0208
P=3	98.3	98.4	99.3	99.5	89.5182	88.9974	89.1276	88.8021
P=4	88.75	87.95	88.55	89.9	79.6224	80.0781	79.6875	81.3151
P=5	68.25	65.15	65.8	67	68.1641	63.9974	64.6484	65.7552

Table 2: With Normalization

Type Of SVM	Training Accuracy				Test Accuracy			
Hard Margin with Linear Kernel	93.55				93.8151			
Hard Margin with polynomial kernel	P = 2	P = 3	P = 4	P = 5	P = 2	P = 3	P = 4	P = 5
	99.6	99.75	99.85	99.9	89.7786	89.0625	88.1510	88.4766
Soft Margin with polynomial kernel	C = 0.1	C = 0.6	C = 1.1	C = 2.1	C = 0.1	C = 0.6	C = 1.1	C = 2.1
P=1	88.1	91.05	91.8	92.2	90.1042	92.0573	92.5781	92.5781
P=2	89.85	92.3	93	93.45	91.1042	92.0573	93.2292	93.6198
P=3	91	93.1	93.9	94.4	91.8620	93.0990	93.5547	93.8151
P=4	91.2	93.75	94.5	94.75	91.6667	93.2943	93.4896	93.2292
P=5	91.75	94.45	94.85	95.4	91.9922	93.4245	93.2292	92.5781

Explanation

The above experiments are done with the threshold value of 1e-6.

- ✓ the linear kernel displays that the training accuracy and the test accuracy has very slight difference and the error is very low in the classification. The data can be assumed to be linearly sparable which is why this level of accuracy can be achieved with a linear kernel. The hard margin also expresses that the hyperplane is created at such position which has very less misclassification points in between hyperplane and the support vectors.
- ✓ Polynomial soft and hard margin has some common characteristics. Table 1 illustrates that training accuracy is getting higher from polynomial degree 2 to degree 3 but it is decreasing after that. For example, in hard polynomial the accuracy is 100% at p=3 but the kernel could only achieve 78%

accuracy with p=5. The reason of this can be introduction of too much nonlinearity in the kernel. Since the data is linearly separable so there is more misclassification when more nonlinearity is considered.

- ✓ Moreover, the model is overfitting with higher values of C in soft polynomial kernel. As the value of C gets higher (i.e., the cost of misclassification is going up), the model will try to make the margins very thin. This will adjust the training accuracy, but the test points are still not following that adjustment. Hence, the model will perform poorly in that case.

As a conclusion, for this type of data the linear kernel with hard margin is performing better. The polynomial kernels with low p values such as p=2 for hard margin and p=1 for soft margin is good enough.

After doing the normalization, it can be observed that the accuracy has improved significantly.

- ✓ There is no change in the performance of the linear kernel with the hard margin.
- ✓ The accuracy has been improved with the increase in the complexity of the polynomial kernel with the hard margin which can be a reason of nonlinearity in the data. With the increase in the nonlinearity in the data is getting projected onto a higher dimensional plane. So the nonlinear higher dimensional plane is able to separate them significantly from what we had in the standardized data.
- ✓ Soft margin is allowing more misclassification by lowering the C value which is why the accuracy is high for C value and low for C=0.1. Similar to the hard margin, the accuracy is going up with the increase in the degree of the polynomial.The polynomial kernels are following the cover's theorem perfectly.
- ✓ However, with the increase In the C value, the test accuracy went down a little bit for higher p values. It can be because of the higher cost incurred on the nonlinear kernel.

As a conclusion, after doing the standardization, the performance of all the kernels have been improved.

Task 3

There are many different types of non-linear kernels. Among them I tried a simple function for my experiment, which is radial basis kernel with hard margin.

$$k(x, y) = \exp\left(-\frac{\|x_i - x_j\|}{10^2}\right)$$

Threshold = 1e-4

The performance from the above function can be shown below:

Table 2

Training Accuracy	Test Accuracy
100	94.53

The above value changes when the value of constant c or the cost of misclassification is changed. The only reason can be because of the highest value of the tangent function which is changing with the change in the constant value. That’s why only this particular type of kernel is providing some good result.

I also tried with the hyperbolic tangent function hard margin, but most of time it was not fulfilling the mercer condition and if the mercer condition is fulfilled then the accuracy is not great compared to other functions.

Although the RBF function performs very well in the training set and the test set, but we know that the hard margin that performs 100% in the training set is not the desired one for classification problem because that will result in more total error from both training and test data. Whereas with soft margin even though there are some misclassifications in both training and testing, but that is the optimal one. So, I would like to propose the polynomial kernel with degree 5 and cost of misclassification as 1.1 for the evaluation set.

Appendix: MATLAB Folders:

m_files: contains all the .m files to run the codes

for different functions there are different .m files and all can be used by SVM_main.m file.

After running the file, it will ask to choose a specific function and then a threshold value.

Which will return the test and train accuracy of the given train and test data.

For inserting the evaluation data, open the dataloader.m file and change the test.mat to the desired file name.