Homework 3

CS420-Machine learning, Shikui Tu, Spring 2018

* Name:Zhiwen Qiang Student ID:515030910367 Email: qlightman@163.com

1 SVM vs. Neural Networks

1.1 Dataset

Table 1: Dataset Description

Name	Class	training size	testing size	feature
a3a	2	3185	29376	123
w1a	2	2477	47272	300
svmguide1	2	3089	4000	4
pendigits	10	7494	3498	16

1.2 SVM results

1.2.1 Setting

In this section, the four dataset I use and the code are in the zip file, opening one of the dataset folder, execute the **script.sh**, then you can see the reuslts.

1.2.2 Apply SVM with or without scaling

It is very important to scale the dataset before applying SVM. There are mainly two advantages for that:

- It can avoid attributes in greater numeric ranges dominating those in smaller numeric ranges.
- It can avoid numerical difficulties during the calculation.

Here, I compare the results of libsvm on **svmguide1** dataset and **pendigits** dataset. The result is shown in **Fig 1.1**.It is obvious to see that the results of applying scaling is much better. So except otherwise explained, the dataset is scaled before applying to any algorithms in the following sections.

1.2.3 Apply SVM with different kernels

I tested the performace of SVM by using four basic kernels:

- linear: $K(x_i, x_j) = x_i^T x_j$
- polynomial: $K(x_i, x_j) = (\gamma x_i^T x_J + r)^d, \gamma > 0$

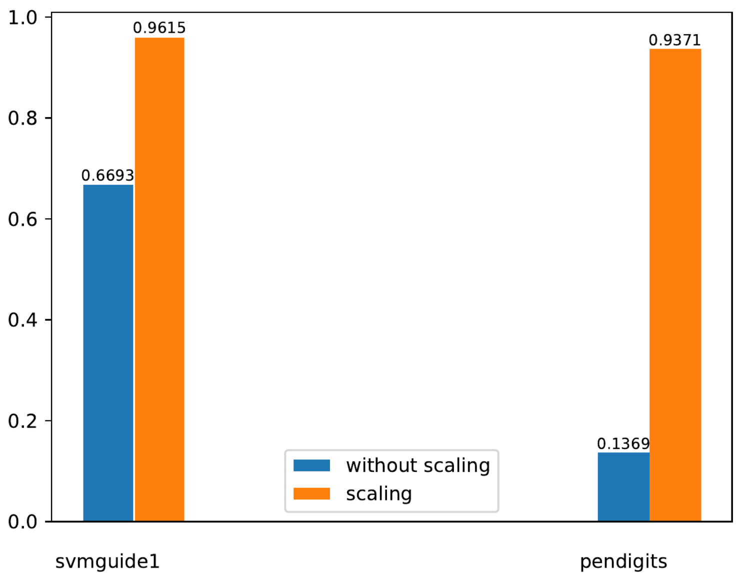

Figure 1.1: Results of SVM with scaling/non scaling

- radial basis function(RBF): $K(x_i, x_j) = exp(-\gamma||x_i - x_j||^2), \gamma > 0$
- sigmoid: $K(x_i, x_j) = \tanh(\gamma x_i^T x_j + r)$

Here, γ, r and d are kernel parameters. In general, the RBF kernel is a reasonable first chice for the following advantages:

- It can handle the case when the relation between class labels and attributes is nonlinear.
- The linear kernel is a special case of RBF kernel.
- The number of hyperparameters for RBF kernel is less compared with polynomial kernel.
- The RBF kernel has fewer numerical difficulties.

Here, I use the **grid.py** presented by **LIBSVM** to implement a 'grid-search' on C and γ on RBF kernel using cross-validation, which tries different pairs of (C, γ) values and the one with the best cross-validation accuracy is picked. The results of applying 'grid-search' on the four dataset is shown in **Fig 1.2**.

The results of applying SVM with different kernels on the four dataset is shown in **Fig 1.3**. We can see that in most cases the results obtianed by RBF kernel with grid search is the best.

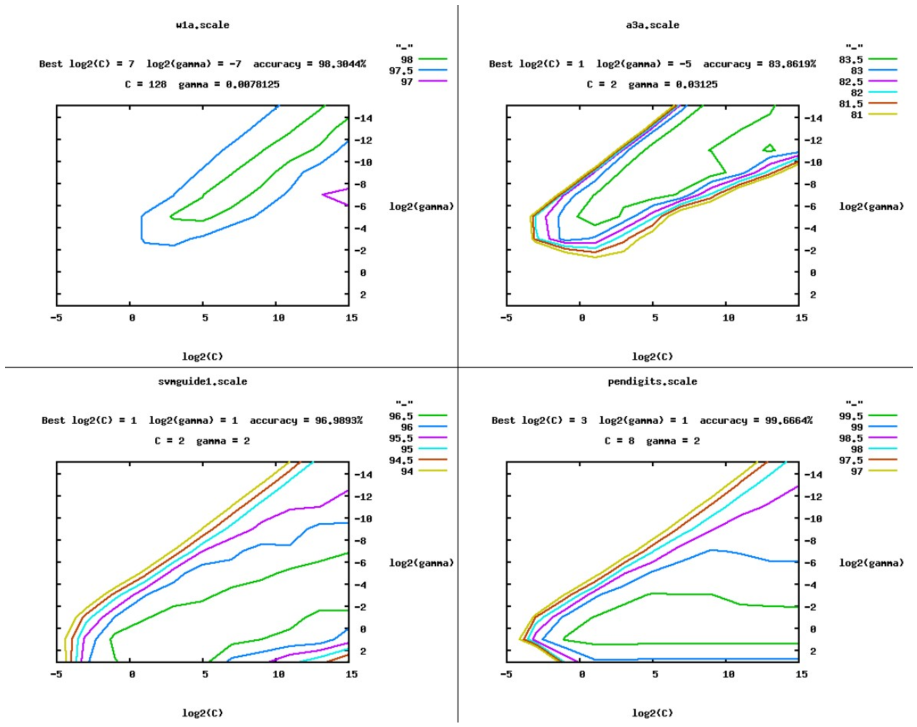

Figure 1.2: Applying grid search on four dataset

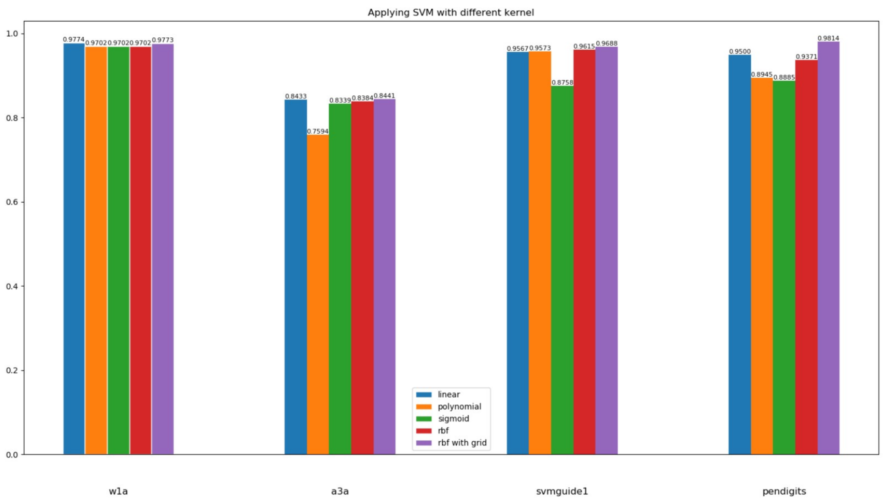

Figure 1.3: Results of SVM with different kernel

1.3 MLP results

1.3.1 Setting

In this part, the code **MLP.py**, **main.py** are in the zip file, the **MLP.py** file are used to transform the file format, execute the **main.py** file, you can see the results.

1.3.2 Apply MLP with different solvers

The sklearn provide three solvers for weight optimization.

- **lbfgs** is an optimizer in the family of quasi-Newton methods.
- **sgd** refers to stochastic gradient descent.
- **adam** refers to a stochastic gradient-based optimizer

The results of applying SVM with different solvers on four dataset is shown in **Fig 1.4**. We can see that the **lbfgs** and **adam** performs similar, while **sgd**'s performance can vary greatly depends on the dataset.

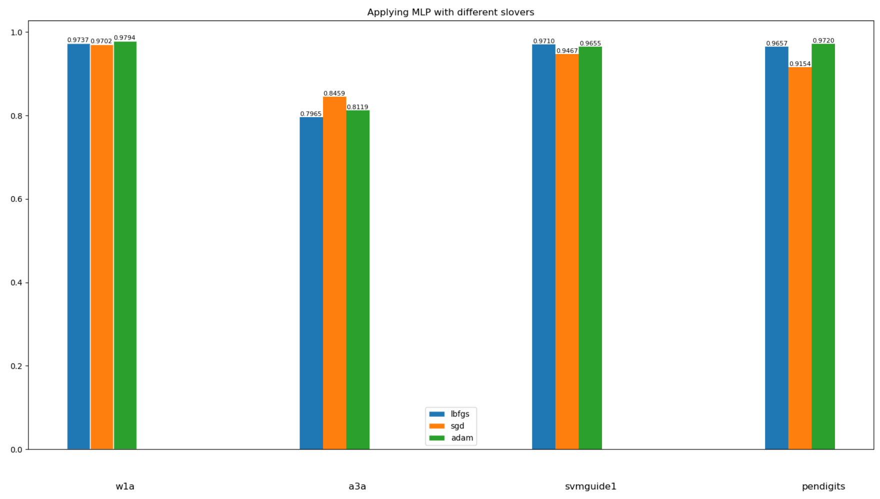

Figure 1.4: Applying MLP with different solvers

1.3.3 Apply MLP with different sample size

Here I use four different sample size (0.2,0.5,0.8,1) of the training set to test the result of MLP algorithm. The result is shown in **Fig 1.5**.

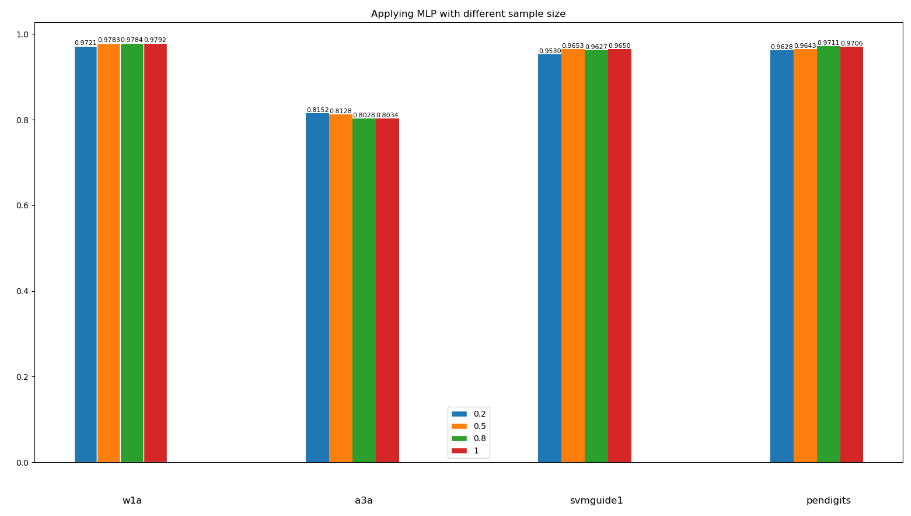

Figure 1.5: Apply MLP with different sample size

1.3.4 Apply MLP with different hidden layer sizes

Here I use four different hidden layer size ((200,100),(100,80),(50,40),(5,2)) to test the result of MLP algorithm. The result is shown in**Fig 1.6**.

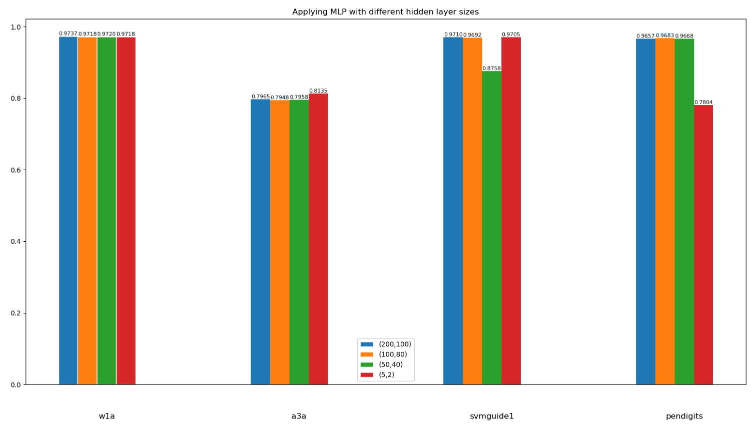

Figure 1.6: Applying MLP with different hidden layer sizes

1.4 Comparsion

By comparing the best result obtained by SVM and MLP, we can see that for small scale dataset, their performace are similar. As is shown in **Fig 1.7**.

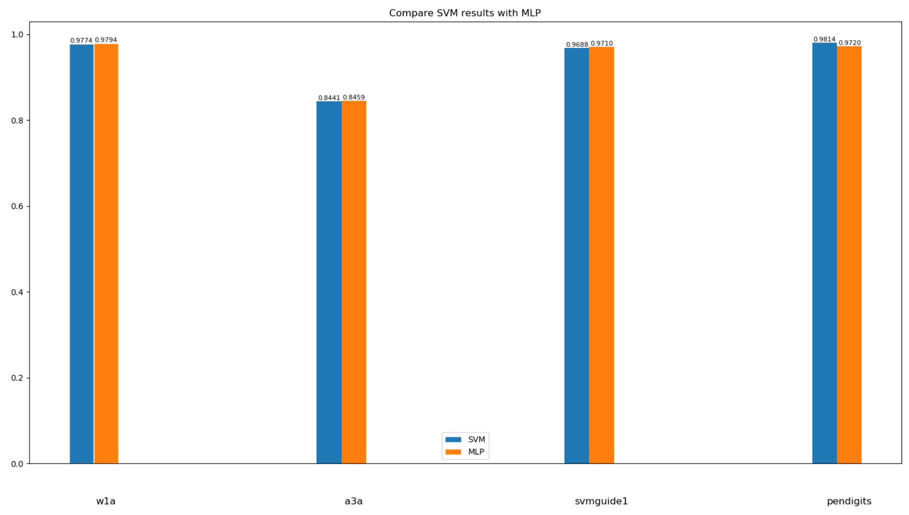

Figure 1.7: Compare SVM results with MLP

2 Bonus

2.1 Settings

Here the code **SVM.py** is in the zip file, I didn't include the dataset since it's too large. Putting the CIFAR10 dataset in the right path, execute the **SVM.py** file, you can see the results.

2.2 Results

Here, I select **CIFAR10** dataset, which consists of 32*32 natural image dataset with 10 categories. **Fig 2.1** are the classes in the dataset, as well as 10 random images from each. Overall there are 50000 images with 5000 images for each class. I applied SVM with RBF kernel on this dataset and the results are shown in **Table 2**. We can see that the results is very poor, the overall accuracy is 9.73 %. It seems that the SVM just randomly pick a category for each testing image. While according to the dataset website, they are 18% test error without data augmentation by using convolutional neural network.

2.3 Strengths and weaknesses of SVM on big data sets

2.3.1 Weaknesses

- Most of the kernel matrix K is a $n * n$ matrix where n is the number of the data, which means storing the kernel matrix requires memory that scales quadratically with the size of the data. Also, the training time for traditional SVM algorithms also scales superlinearly with the number of

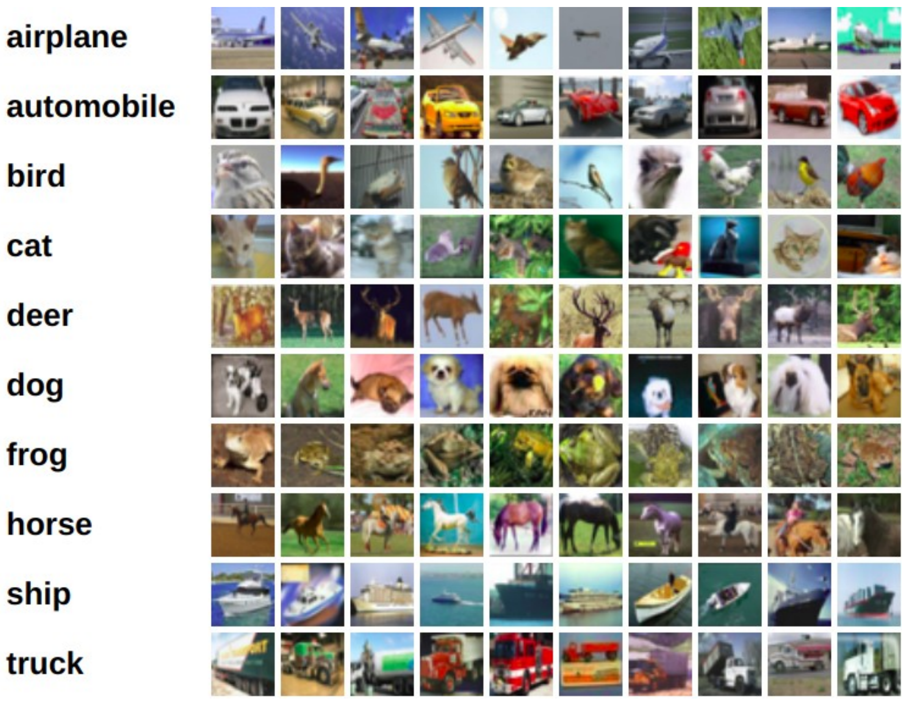

Figure 2.1: Overview of the CIFAR10 dataset

Table 2: Results of applying SVM on CIFAR10

features	details	accuracy
category 0	13 of 984 test values correct.	0.0132113821138
category 1	34 of 1007 test values correct.	0.0337636544191
category 2	8 of 1010 test values correct	0.00792079207921
category 3	14 of 995 test values correct.	0.0140703517588
category 4	1 of 1010 test values correct.	0.000990099009901
category 5	16 of 988 test values correct.	0.0161943319838
category 6	701 of 1008 test values correct.	0.695436507937
category 7	18 of 1026 test values correct.	0.0175438596491
category 8	14 of 987 test values correct.	0.0141843971631
category 9	154 of 985 test values correct.	0.156345177665
overall	973 of 10000 test values correct.	0.0973

data points. So these SVM algorithms aren't feasible since they require too much memories and are kind of slow.

- Before applying SVM to a real problem, you have to choose the right kernel, which means you have to provide the true structure of the data as an input, while other algorithms, like neural networks or random-forests, try to automatically find the structure. Also, finding the structure of big data set is difficult in many cases. Of course people can use some well-difined kernels but tune the parameters for these kernels and the C parameter can be time consuming.

2.3.2 Strengths

- We can perform kernel approximation uses the Nystrom approximation (Williams and Seeger 2001). This is a way to approximate the eigenvalues/eigenvectors of a large matrix using a smaller submatrix. So that the size of kernel matrix can be greatly reduced.
- We can approximate the optimization problem with a set of smaller subproblems. For example, using stochastic gradient descent on the primal problem. So that the training time can be greatly reduced.
- There are many SVM applications which can handle big data sets like **LaSVM** which uses online approximation to approximate SVM solver. As is shown in **Fig 2.2**, LASVM requires considerably less memory than a regular SVM solver. This becomes a considerable speed advantage for large training sets.

3 Reference

LIBSVM. Chih-Chung Chang and Chih-Jen Lin (https://www.csie.ntu.edu.tw/ cjlin/libsvm/)
Williams and Seeger (2001). Using the Nystroem method to speed up kernel machines.
CIFAR10: (http://www.cs.utoronto.ca/ kriz/cifar.html)
LaSVM.leon.bottou.org (http://leon.bottou.org/projects/lasvm)

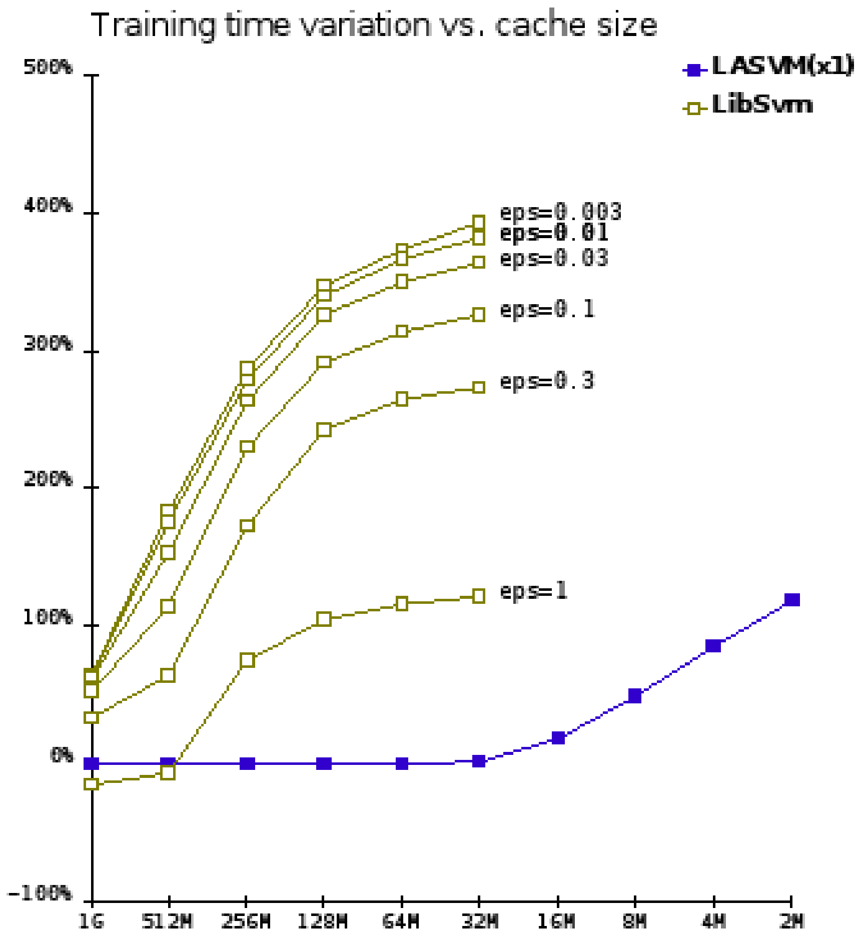

Figure 2.2: Compare LaSVM with LibSVM