# Report for Assignment 2

Zou Yanyan

March 22, 2016

## 1

Show the distribution of three datasets in Figure 1, 2, 3 respectively. The cross-shaped points represent "+1", and the yellow circle points represent "-1".

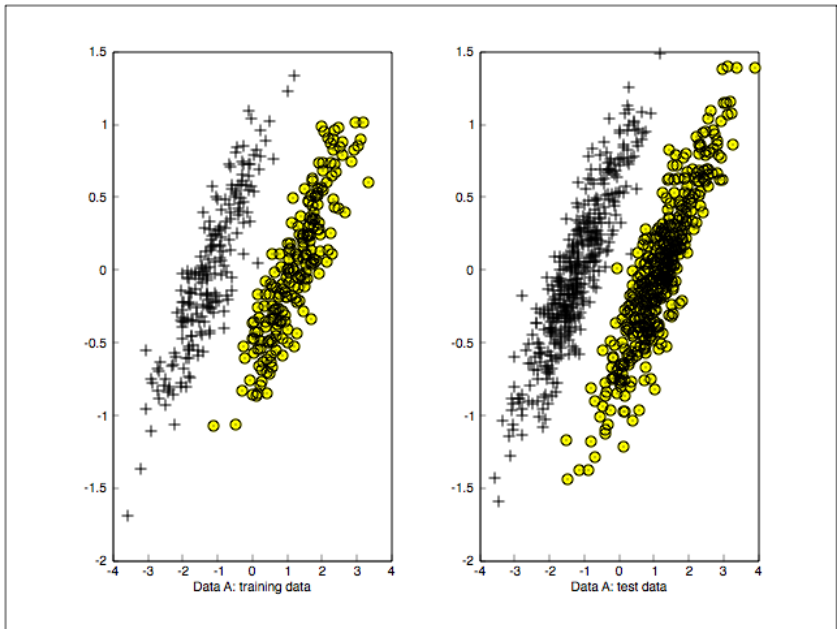


Figure 1. Data A

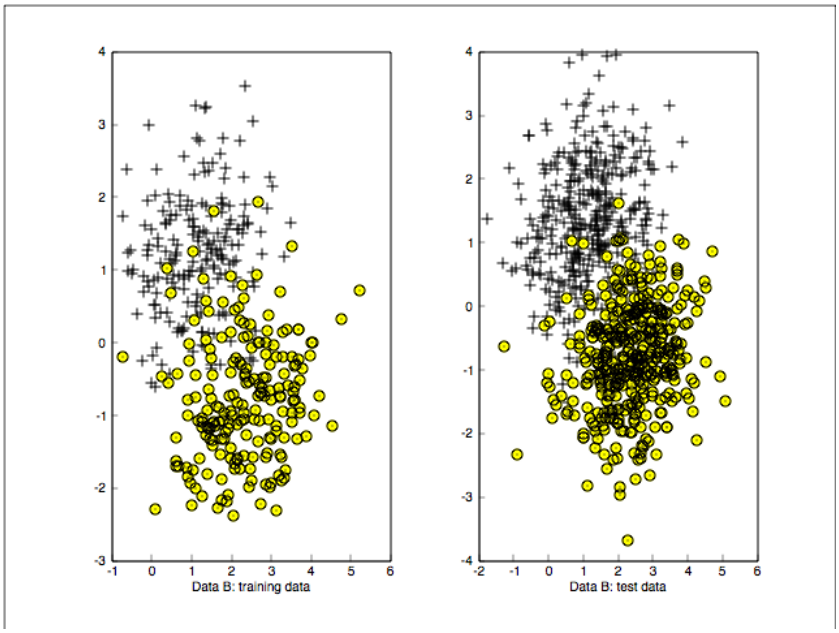


Figure 2. Data B

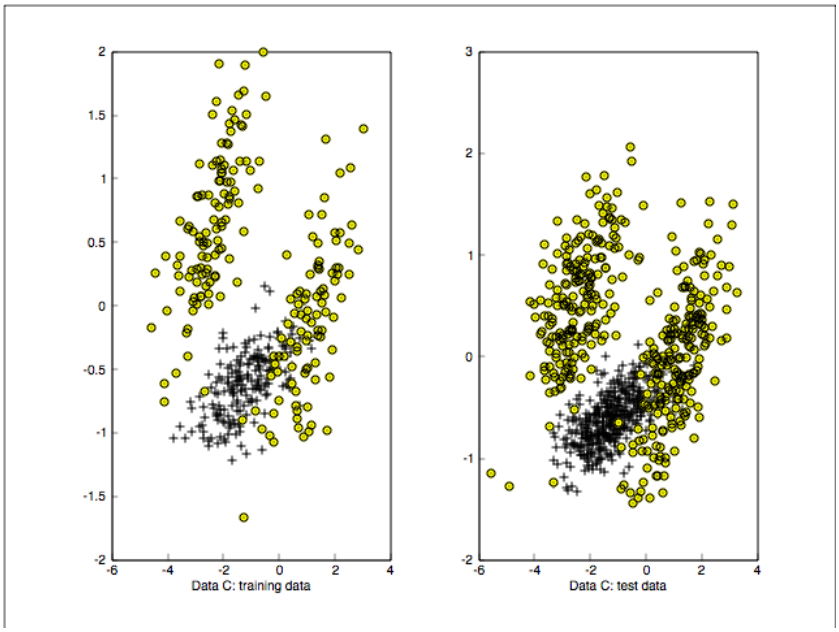


Figure 3. Data C

As we can see from Table 1, the experiment results from primal form and dual form respectively are almost the same. Hence, the implementation is correct.

| | A_train | A_test | B_train | B_test | C_train | C_test |
|---|---|---|---|---|---|---|
| primal | 0.9975 | 0.9963 | 0.9125 | 0.92 | 0.8575 | 0.835 |
| dual | 0.9975 | 0.9963 | 0.9125 | 0.92 | 0.8575 | 0.835 |

Table 1. linear SVM with $C = 1.0$

The accuracies of training and test data A are the highest, since the data A is linearly separable where it is to be separated, almost hundred percent. While, the data B and C are slightly linearly separable, so the accuracies are much lower. Especially, the data C is skewed with the lowest accuracies.

## 2

Let us consider the data A first. To make figures easier to be observed, we consider the $log_{10}C$ rather than $C$ as $C$ varies from 0.0001 to 1000. Also, the margin is inversely proportional to the module of $w$, where $w$ is a 2D vector as well as the parameter of linear SVM, so we can only consider the inverse module of $w$ as the margin.

| C | A_train | A_test | support vectors | margin |
|---|---|---|---|---|
| 0.0001 | 0.9525 | 0.9125 | 400 | 40.303 |
| 0.001 | 0.955 | 0.91625 | 372 | 4.8067 |
| 0.01 | 0.9875 | 0.98 | 174 | 1.8959 |
| 0.1 | 0.9975 | 0.99625 | 54 | 1.0133 |
| 1 | 0.9975 | 0.99625 | 14 | 0.6095 |
| 10 | 0.9975 | 0.9975 | 4 | 0.3832 |
| 100 | 1 | 0.99625 | 1 | 0.173 |
| 1000 | 1 | 0.99625 | 0 | 0.173 |

Table 2. Performance of data A with different C values

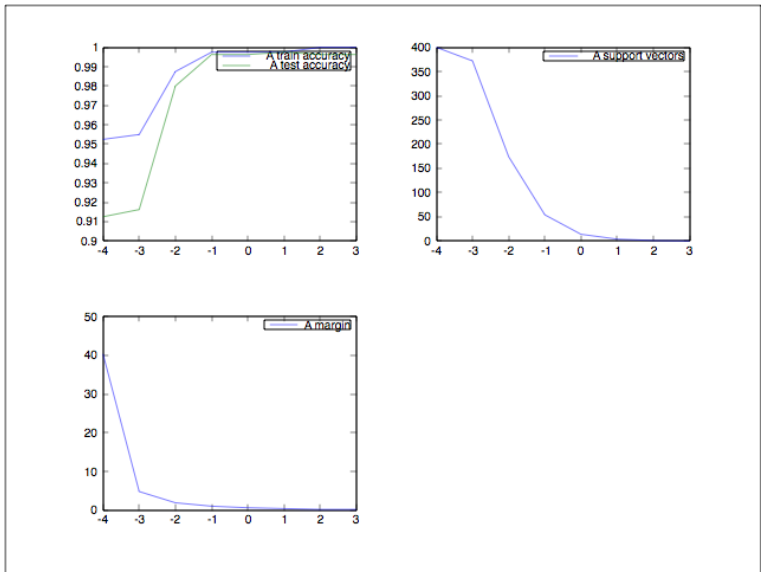


Figure 4. Performances of data A with different C values

Considering the data A, as C increases, the accuracies of training and test data increases, the number of support vectors decrease, and the margin decreases.

Now, we consider the data B.

| C | B_train | B_test | support vectors | margin |
|---|---|---|---|---|
| 0.0001 | 0.9125 | 0.91875 | 400 | 43.986 |
| 0.001 | 0.92 | 0.91625 | 382 | 4.9294 |
| 0.01 | 0.9175 | 0.92 | 185 | 2.2255 |
| 0.1 | 0.915 | 0.92375 | 100 | 1.4223 |
| 1 | 0.9125 | 0.92 | 80 | 1.2509 |
| 10 | 0.9125 | 0.92125 | 77 | 1.2351 |
| 100 | 0.9125 | 0.92125 | 78 | 1.2241 |
| 1000 | 0.9125 | 0.92125 | 77 | 1.2241 |

Table 3. Performance of data B with different C values

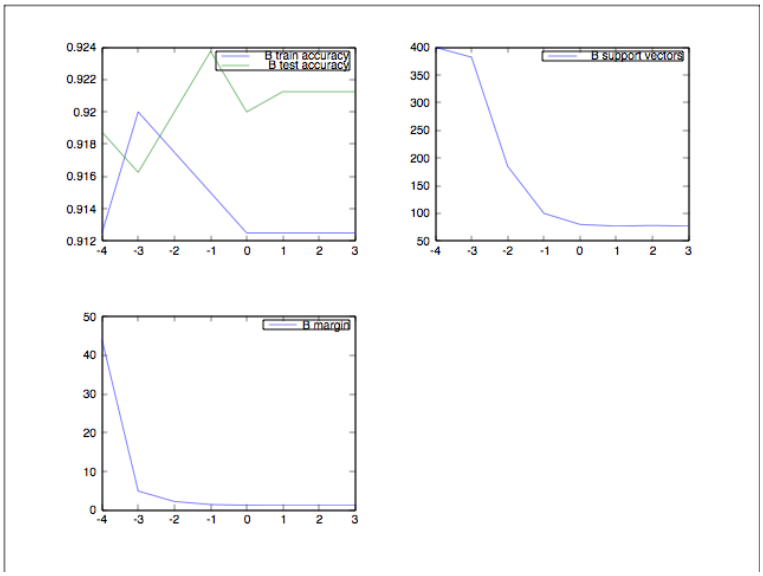


Figure 5. Performances of data B with different C values

As we can see, with C increasing, the number of support vectors and the margin both decrease, which the accuracies of training and test data do not change a lot.

Let us see what happen in data C.

| C | C_train | C_test | support vectors | margin |
|---|---|---|---|---|
| 0.0001 | 0.7625 | 0.78 | 400 | 106.08 |
| 0.001 | 0.76 | 0.78125 | 400 | 10.608 |
| 0.01 | 0.83 | 0.82 | 332 | 1.9501 |
| 0.1 | 0.855 | 0.83625 | 196 | 0.9544 |
| 1 | 0.8575 | 0.835 | 151 | 0.7256 |
| 10 | 0.8575 | 0.83625 | 143 | 0.689 |
| 100 | 0.8575 | 0.83625 | 143 | 0.689 |
| 1000 | 0.8575 | 0.83625 | 142 | 0.689 |

Table 4. Data C performance with different C values

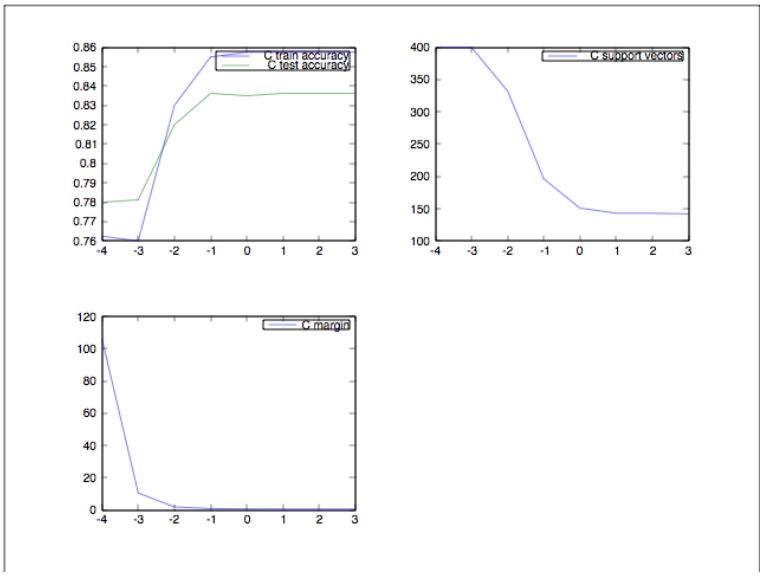


Figure 6. Performances of data C with different C values

Similarly, the number of support vectors and the margin decrease as C increases, while the accuracies are improved in general.

The parameter $C = \frac{1}{\lambda}$ is to balance how much favor increasing the margin over satisfying the classification constrains. Smaller values of $C$ will push the margin boundaries and potentially the decision boundary past the examples. Larger values of $C$ will correspond to the small margin, since we prefer to satisfy the classification constrains, and that may be the reason why the number of support vectors decreases while the accuracies increase.

## 3

As we discussed before, The parameter $C = \frac{1}{\lambda}$ is to balance how much favor increasing the margin over satisfying the classification constrains. The optimal value of $C$ depends on the datasets and the conditions we need to satisfy. In general, we can select $C$ which correspond to the most generalized model.

According to the question 2, it is obvious that we can not optimize the value of $C$ by maximizing the margin on the training set, since the maximum margin on the training set does not always correspond to the well-generalized model in test set. Moreover, our goal is to obtain good performance in yet unseen data, i.e. test set.

One possible way to select a good value of $C$ is to use iterative method. That is like what we implement in question 2. We have no idea what the best value of $C$ is, hence, we try almost each possible value and select the best one which corresponds to the best performance in test set. The optimal value of $C$ may be not unique. The table 3 shows

some optimal values of $C$ which correspond to well-generalized model in test set.

| C | A_train | A_test |
|---|---|---|
| 10 | 0.9975 | 0.9975 |
| C | B_train | B_test |
| 0.1 | 0.915 | 0.92375 |
| C | C_train | C_test |
| 0.1 | 0.855 | 0.83625 |

Table 5. Performances of 3 dataset with optimal $C$ values

**4**

Extending the previous implementation of dual form, we introduce the Gaussian (RBF) kernel, $K(x, x') = exp(-\frac{||x-x'||^2}{2\sigma^2})$, or $K(x, x') = exp(\gamma||x - x'||^2)$, where $\gamma = -\frac{1}{2\sigma^2}$. Here, we introduce to the second one.

| Gaussian kernel | | | |
|---|---|---|---|
| | Train | Test | gamma |
| A | 0.9975 | 0.99625 | 5 |
| B | 0.915 | 0.92125 | 0.5 |
| C | 0.9375 | 0.93 | 5 |

Table 6. Performances of Gaussian Kernel with optimal parameters

We also try to use Polynomial kernel function, $K(x, x') = (ax^T x' + c)^d$, where $a, c, d$ are parameters. The table 5 shows the experiments results with the optimal parameters. Different datasets correspond to different parameters with the best performance even they apply the same kernel function.

| Polynomial Kernel | | | | | |
|---|---|---|---|---|---|
| | Train | Test | a | c | d |
| A | 0.9975 | 0.99625 | 1.138 | 3 | 2 |
| B | 0.9125 | 0.92 | 0.89 | 0 | 1 |
| C | 0.93 | 0.92125 | 1.138 | 3 | 2 |

Table 7. Performances of Polynomial Kernel with optimal parameters

According to those experiment results, we can see that performances of all three data set are improved, especially in data C. Therefore, applying proper kernel function to solve non-linearly separable data is a good idea.

**5**

There are many ways to set the value of learning rate. In this case, I apply learning rate as the inverse value of the number of iteration times, that is $learning\_rate = \frac{1}{iteration+1}$. After introducing different values of $\lambda$, we get different performances. The experiment results of three datasets are shown in Table 8, 9, 10 respectively. Compared to the performance of linear SVM, the accuracies on both training and test set are similar with some $\lambda$. However, if we try to set $\lambda$ as larger values, the performance is poor, even causing Runtime warning with NAN value of objective function. The parameter $\lambda$ is also to balance how much we favor to keep the parameters small, i.e. minimizing the squared norm, or to fit to the training data, i.e. minimizing the empirical risk. When we select the optimal $\lambda$, we need to consider this trade-off.

| λ | A_train | A_test | A_loss |
|---|---|---|---|
| 0.0001 | 0.9975 | 0.99625 | 0.645349335 |
| 0.001 | 0.9975 | 0.99625 | 0.64356307 |
| 0.01 | 0.99 | 0.9875 | 0.345752105 |
| 0.1 | 0.9725 | 0.95625 | 0.336513073 |
| 1 | 0.96 | 0.92625 | 0.469421968 |
| 2 | 0.9525 | 0.91375 | 0.426619534 |
| 5 | 0.0975 | 0.1125 | INF |
| 6 | 0.0475 | 0.085 | INF |
| 10 | 0.5 | 0.5 | NAN |
| 100 | 0.5 | 0.5 | NAN |
| 1000 | 0.5 | 0.5 | NAN |

Table 8. Performances on data A

| λ | B_train | B_test | B_loss |
|---|---|---|---|
| 0.0001 | 0.915 | 0.92 | 0.024095742 |
| 0.001 | 0.9125 | 0.91875 | 0.049066596 |
| 0.01 | 0.915 | 0.9225 | 0.10648338 |
| 0.1 | 0.9125 | 0.91875 | 0.263182416 |
| 1 | 0.915 | 0.92 | 0.379484206 |
| 2 | 0.915 | 0.91625 | 0.440519094 |
| 5 | 0.3175 | 0.3125 | INF |
| 6 | 0.53 | 0.535 | INF |
| 10 | 0.5 | 0.5 | NAN |
| 100 | 0.5 | 0.5 | NAN |
| 1000 | 0.5 | 0.5 | NAN |

Table 9. Performances on data B

| λ | C_train | C_test | C_loss |
|---|---|---|---|
| 0.0001 | 0.83 | 0.8325 | 0.063276604 |
| 0.001 | 0.835 | 0.8325 | 0.070700773 |
| 0.01 | 0.8325 | 0.83375 | 0.122671118 |
| 0.1 | 0.835 | 0.83375 | 0.317587575 |
| 1 | 0.735 | 0.76625 | 0.380761066 |
| 2 | 0.6375 | 0.7025 | 0.38234422 |
| 5 | 0.385 | 0.31625 | INF |
| 6 | 0.38 | 0.31625 | INF |
| 10 | 0.5 | 0.5 | NAN |
| 100 | 0.5 | 0.5 | NAN |
| 1000 | 0.5 | 0.5 | NAN |

Table 10. Performances on data C