Solution to Homework 4

Shoeb Mohammed and Zhuo Chen

March 21, 2016

1 Kernelizing the perceptron

$$\mathcal{D} = \{x^{(i)} | 1 \leq i \leq m \, y^{(i)} \in \{-1, 1\}\}$$

$$\theta^{(i)} \leftarrow \theta^{(i-1)} - [h_{\theta^{(i-1)}}(x^{(i)}) - y^{(i)}]x^{(i)} \tag{1}$$

1.1

From equation 1, it is clear that θ is a linear combination of vectors $x^{(i)}$. Thus, θ can be implicitly represented by the weights α_i in

$$\theta = \sum_{i=1}^{m} \alpha_i \phi(x^{(i)}) \tag{2}$$

The α_i are dual variables and initialized to zero.

1.2

From equation 2 it follows that

$$\begin{aligned} h_{\theta^{(i)}}(\phi(x^{(i+1)})) &= sign\left(\theta^{(i)^T}\phi(x^{(i+1)})\right) \\ &= sign\left(\sum_{j=1}^{m} \alpha_j \phi(x^{(j)})^T \phi(x^{(i+1)})\right) \\ &= sign\left(\sum_{j=1}^{m} \alpha_j K(x^{(j)}, x^{(i+1)})\right) \end{aligned} \tag{3}$$

In equation 3, during training, $\alpha_j = 0$ for $j > i$.

1.3

For a new training example, we use the update rule

$$\begin{aligned}\alpha_{i+1} &\leftarrow h_{\theta^{(i)}}(\phi(x^{(i+1)})) - y^{(i)} \\ &\leftarrow sign\left(\sum_{j=1}^{m} \alpha_j K(x^{(j)}, x^{(i+1)})\right) - y^{(i)}\end{aligned} \tag{4}$$

In equation 4, during training, $\alpha_j = 0$ for $j > i$.

2 Fitting an SVM classifier by hand

$$\mathcal{D} = \{(0, -1), (\sqrt{2}, +1)\}$$
$$\phi(x) = (0, \sqrt{2}x, x^2)$$

We fit a maximum margin classifier for $\mathcal{D}$ and features $\phi(x)$.

2.1

The vector v along the line joining the two points is parallel to optimal vector θ.

$$v = (0, 2, 2)$$

2.2

The value of the margin is $2\sqrt{2}$.

2.3

$$\theta = (0, 1, 1)$$

2.4

$$\theta_0 = 2$$

2.5

The equation for decision boundary is

$$\theta^T \phi(x) = 2$$

3 Support vector machines for binary classification

3.1 Support vector machines

3.1.1 The hinge loss function and gradient

3.1.2 Example dataset 1: impact of varying C

- Checked loss function and gradient with the values in the homework.
- Verified decision boundary with C=1 and C=100. It matches the boundary in homework figure 3.

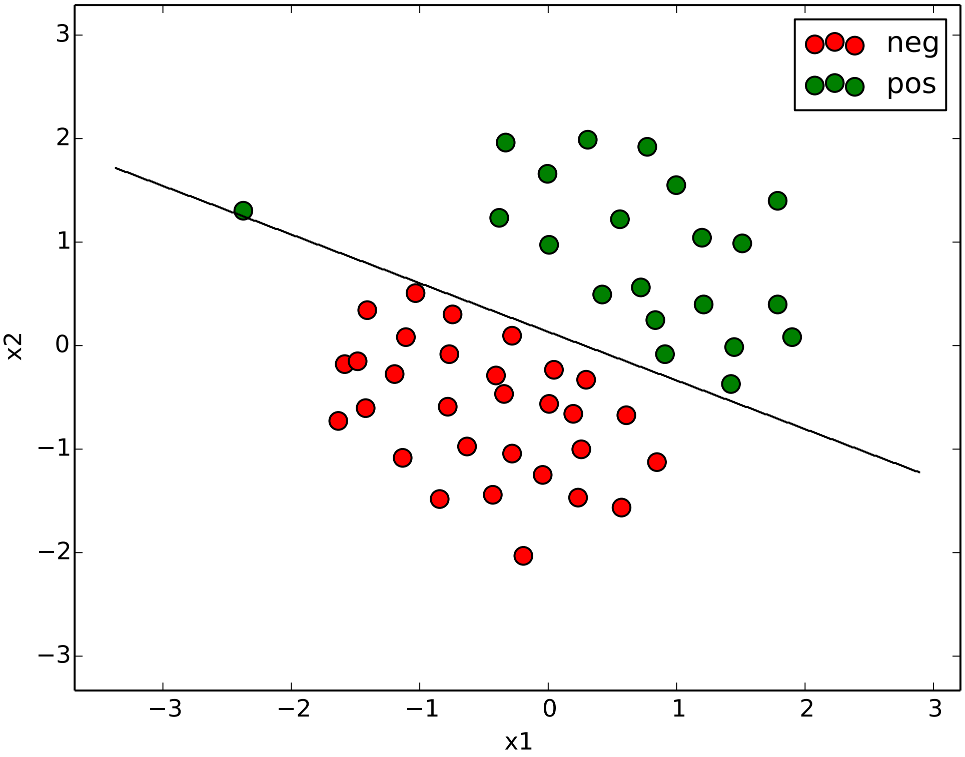

Figure 1: decision boundary with C=100

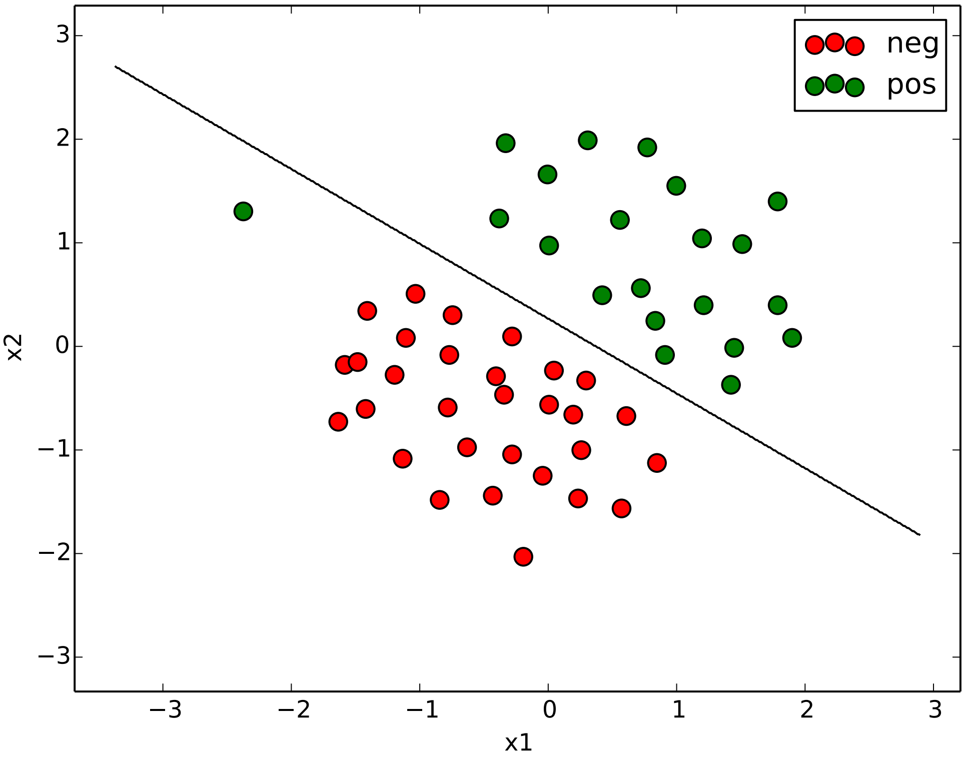

Figure 2: decision boundary with C=1

3.1.3 Gaussian kernel

- Implemented the Gaussian kernel.
- Verified it works correct.
- The decision boundary using Gaussian kernel is plotted in figure 3

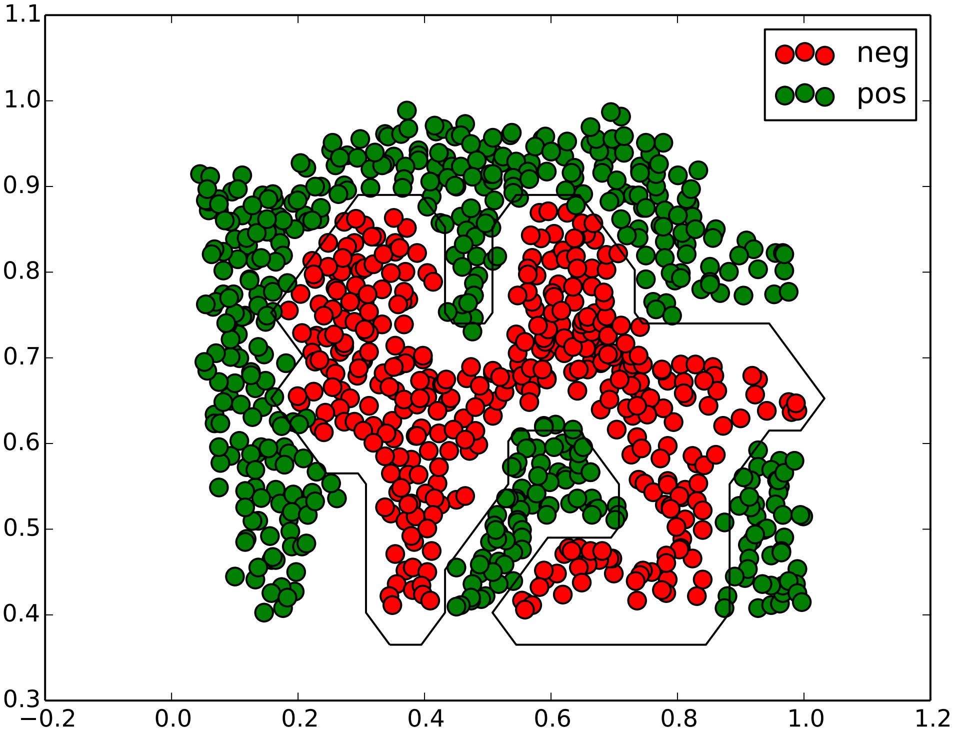

Figure 3: Learning non-linear boundary using a Gaussian kernel

3.2 Example dataset 3: selecting hyper parameters for SVMs

Searching over $C, \sigma \in \{0.01, 0.03, 0.1, 0.3, 1, 3, 10, 30\}$ we get following best hyper parameters. The decision boundary is plotted in figure 4

```
*****Problem 3.2******
Best C = 1.000000e-01 Best sigma = 1.000000e-01
with validation accuracy = 9.600000e-01
```

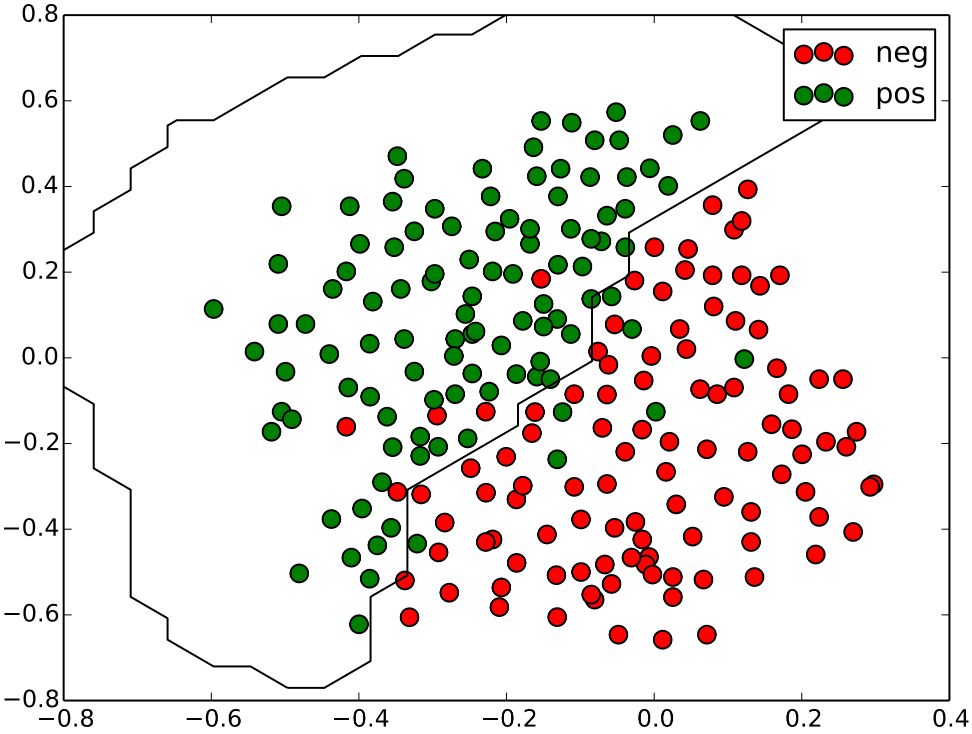

Figure 4: decision boundary with best $C = 0.1$ and best $\sigma = 0.1$

3.3 Spam Classification with SVMs

To find best C, learning rate and number of iterations, we devided the training set(4000) to training set(3200) and validation set(800), and did some experiments.

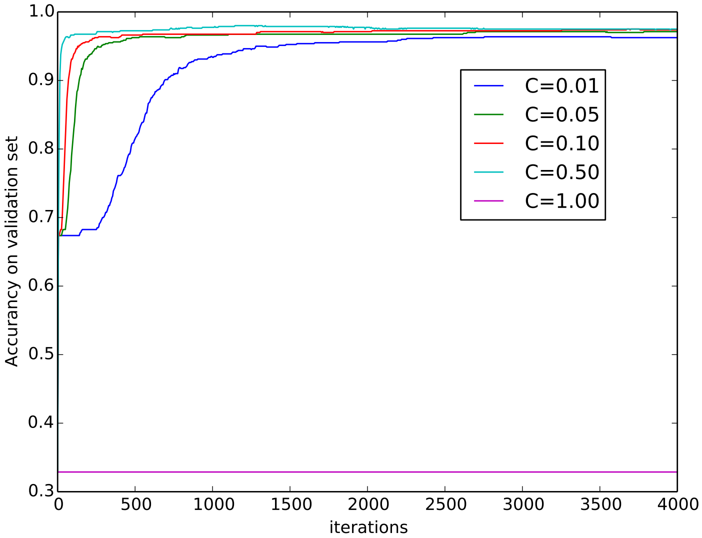

Figure 5: How the performance on validation set changes with iteration. Here we set learning rate to 0.1 and vary C. We can see that the performance of c=0.05,0.1 and 0.5 are similar, while when C=1 the model is overfitted.

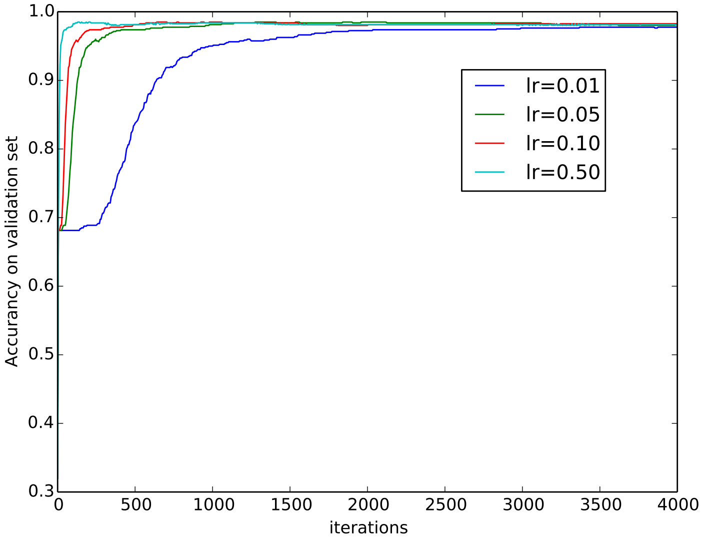

Figure 6: Set C to 0.1 and vary learning rate. According to this figure I think if we set learning rate to 0.1, 2000 iterations will be enough.

Set C to 0.1 and learning rate to 0.1, with 2000 iterations, we will get:

```
Accuracy of model on training data is:  0.98025
Accuracy of model on test data is:  0.981
Top 15 predictors of spam are:
click
remov
our
here
your
pleas
basenumb
guarante
will
you
email
nbsp
free
offer
hour
```

Then we test the RBF-kernalized condition. (kernal matrix is generated using sklearn) We tried different sigma, but the performance of all sigma is much worse than unkernalized model(accurancy on validation set is less than 0.8). I think it's because the weight of each word is different, but when the data is kernalized, this difference doesn't exist. Additionally, kernalized model is unsuffcient to give top15 important words. So in this problem, the data should not be kernalized.