COMP 540 – Statistical Machine Learning

Rice University

Assignment 4（Writeup）

Created By:

Group: Chengyin Liu(cl93), Ran Jin(Oliver)(rj23)

Problem 1)

a) In general, maximizing the margin for a SVM problem would make a wide boundary of margin and this is optimal since we want to clearly differentiate the classes. On the extreme case, if the margin is minimized, then the points belonging to different classes might be too close that there could exist some misclassifications.

b) Moving points which are not support vectors further away from the decision boundary will not effect the SVM's hinge loss.

The hinge loss function is $J(\theta) = \frac{1}{2m}\sum_{j=0}^{d}\theta_j^2 + \frac{C}{m}\sum_{i=1}^{m}\max(0,1 - y^{(i)}h_\theta(x^{(i)}))$

Since the points that are not support vectors(correctly classified) $\max(0,1 - y^{(i)}h_\theta(x^{(i)}))$ is always 0 no matter how far it is from the boundary. Thus, it will not affect the hinge loss.

Problem 2)

a) $x^{(1)} = 0, x^{(2)} = \sqrt{2}, \Phi(x^{(1)}) = (1,0,0), \Phi(x^{(2)}) = (1,2,2)$

θ *is perpendicular to the devision boundary and* $\Phi(x^{(2)}) - \Phi(x^{(1)})$ is a vector also perpendicular to the boundary. So, $\Phi(x^{(2)}) - \Phi(x^{(1)})$ *is perpendicular to* θ.

vector $\Phi(x^{(2)}) - \Phi(x^{(1)}) = (0,2,2)$ is a vector parallel to the optimal vector.

b) The margin is twice the distance from each support vector to the decision boundary, that is the distance between $\Phi(x^{(2)})$ *and* $\Phi(x^{(1)})$.

$$\frac{\sqrt{0+4+4}}{2} = \sqrt{2}$$

c) Given the margin is $\frac{1}{||\theta||} = \sqrt{2}$,

Then $||\theta|| = \frac{1}{\sqrt{2}}$. Since θ *is parallel to* $(0,2,2)$, then we set $\theta = k(0,2,2)$.

Then $||\theta|| = \sqrt{0 + 4k^2 + 4k^2} = \frac{1}{\sqrt{2}}$.

$k = \frac{1}{4}$, *then we have our* $\theta = (0, \frac{1}{2}, \frac{1}{2})$.

d) $y^{(1)}(\theta^T\Phi(x^{(1)}) + \theta_0) \geq 1,\ y^{(2)}(\theta^T\Phi(x^{(2)}) + \theta_0) \geq 1,$

$-1 * (0 + \theta_0) \geq 1, 1 * (2 + \theta_0) \geq 1,$ *then* $\theta_0 = -1$ (both points are support vectors. So, the inequalities are tight.)

e) $\theta^T \Phi(x) + \theta_0 = 0, \textit{then the decision boundary would be } \frac{x^2}{2} + \frac{x}{\sqrt{2}} - 1 = 0$

Problem 3)

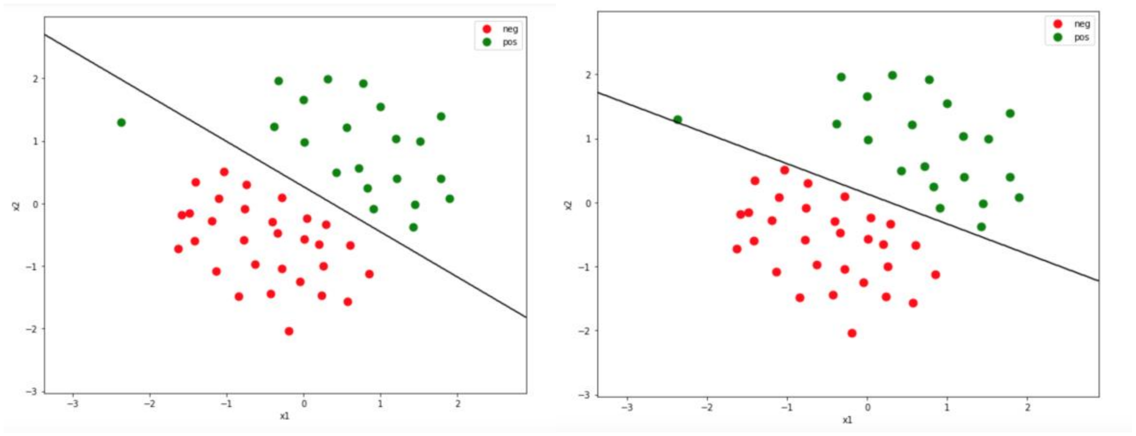
3.1A)

The result is indeed as what the question stated

```
J =  1.0  grad =  [-0.12956186 -0.00167647]
```

3.1B)

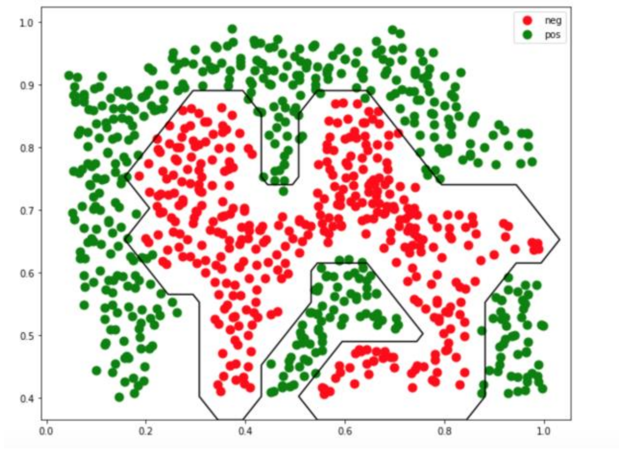
The left one is C=1, and the right one is C=100.



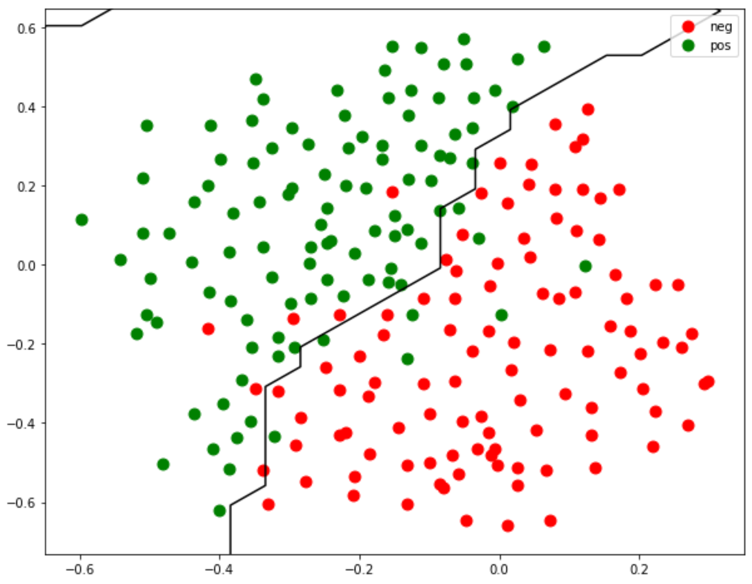
As we see in the left one(C=1), there exists some misclassification, but the right one(C=100) correctly classifies every example.

3.1C)

```
Guassian kernel value (should be around 0.324652) =  0.324652467358
```



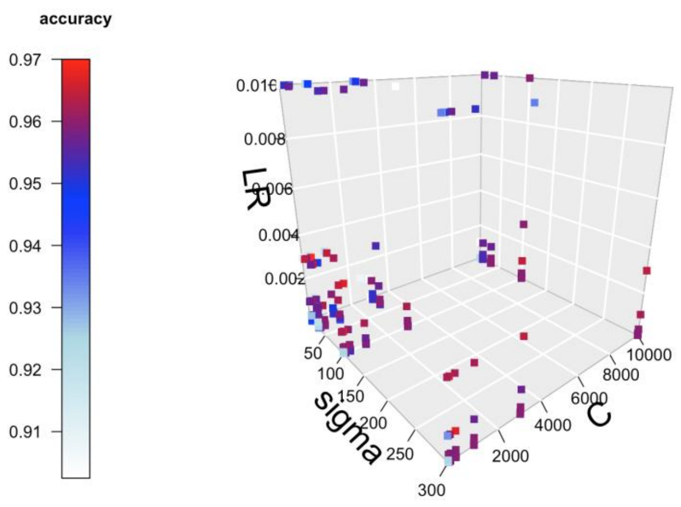
3.2)
Best C, sigma pair is (0.300000, 0.100000) with an error of 0.040000.



3.3)
We separated 4000 training images into 3600 for training set, and 400 for validation. We use Gaussian kernel since the feature vectors are in a high dimensional space in which Euclidean distance is a reasonable measurement, and it gives a pretty high accuracy as we tried iterations = 200 in the beginning. Although the span of each dimension of the feature vector is the same in this problem, we do scale the data matrix, using *StandardScaler()* to make the distances computed after Gaussian kernel more smoother.

Our algorithm goes like this, basically we start from a small iteration number, and set 7 values for sigma, C, and learning rates. Specifically, we have
sigma values = [0.3,1,3,10,30,100,300],
C values = [10,30,100,300,1000,3000,10000],
learning rates = [1e-5,3e-5,1e-4,3e-4,1e-3,3e-3,1e-2].

The following is our graph picking the sigma, C, and learning rate combination. And here we only show the points with accuracy above 0.9. Best sigma, C, lr are (10.000000, 100.000000, 0.003000) with an accuracy on the validation set of 0.975000.



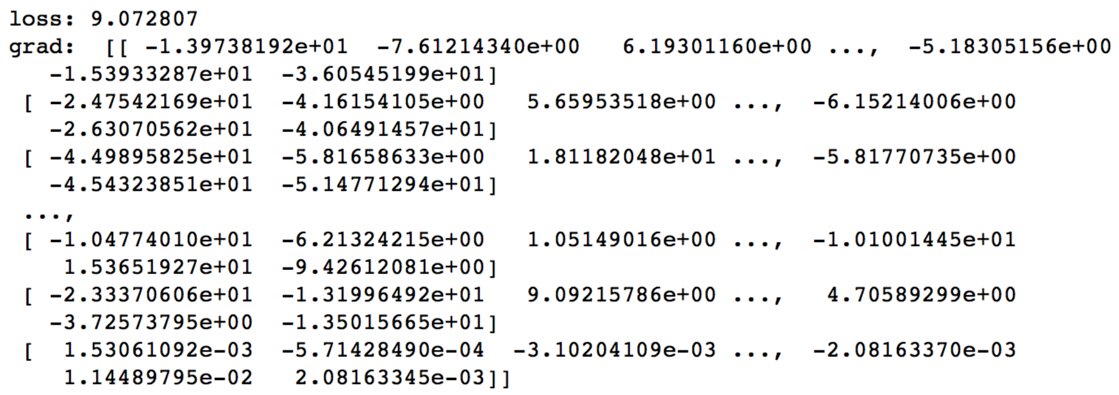
We then run several numbers of iterations with the best combination of other parameters. Based on the loss converging speed, we set iteration number as 30000 when we train the whole training set. The final accuracies we get on training set and testing set are:
Accuracy on the training set: 0.99925
Accuracy on the testing set: 0.988

And the top 15 indicative words for spam emails we got are as follows:

```
###### top 15 words ######
remot
clearli
otherwis
mondai
info
wife
with
dollarac
doesn
gt
human
mark
militari
similar
hot############
```

Problem 4)

A)

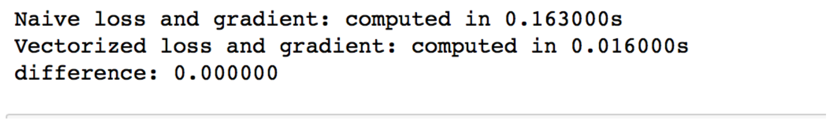


```
loss: 9.072807
grad:  [[ -1.39738192e+01  -7.61214340e+00   6.19301160e+00 ...,  -5.18305156e+00
   -1.53933287e+01  -3.60545199e+01]
 [ -2.47542169e+01  -4.16154105e+00   5.65953518e+00 ...,  -6.15214006e+00
   -2.63070562e+01  -4.06491457e+01]
 [ -4.49895825e+01  -5.81658633e+00   1.81182048e+01 ...,  -5.81770735e+00
   -4.54323851e+01  -5.14771294e+01]
 ...,
 [ -1.04774010e+01  -6.21324215e+00   1.05149016e+00 ...,  -1.01001445e+01
    1.53651927e+01  -9.42612081e+00]
 [ -2.33370606e+01  -1.31996492e+01   9.09215786e+00 ...,   4.70589299e+00
   -3.72573795e+00  -1.35015665e+01]
 [  1.53061092e-03  -5.71428490e-04  -3.10204109e-03 ...,  -2.08163370e-03
    1.14489795e-02   2.08163345e-03]]
```

➢ It is possible that once in a while a dimension in the gradient check will not match exactly. What could such a discrepancy be caused by? Is it a reason for concern? Hint: the SVM loss function is not, strictly speaking, differentiable.

The SVM loss function is not differentiable at all points. Specifically the max function is not differentiable at some points. The function takes effect which makes the slight mismatch. Thus, the numerical gradient will fail to check at these points. However, such discrepancy should not be a concern since there are only few of those points which should not significantly influence results.

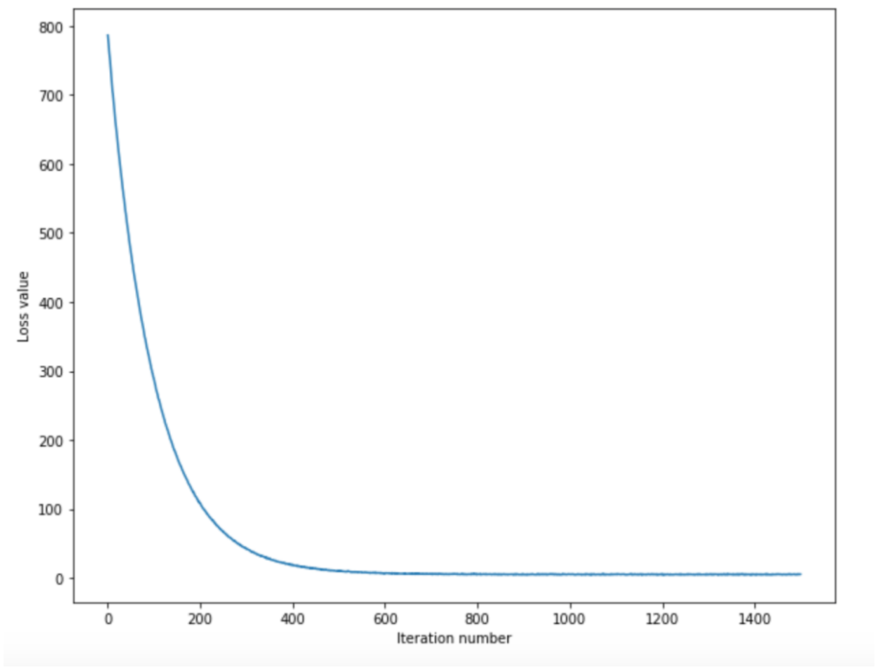
B)



```
Naive loss and gradient: computed in 0.163000s
Vectorized loss and gradient: computed in 0.016000s
difference: 0.000000
```

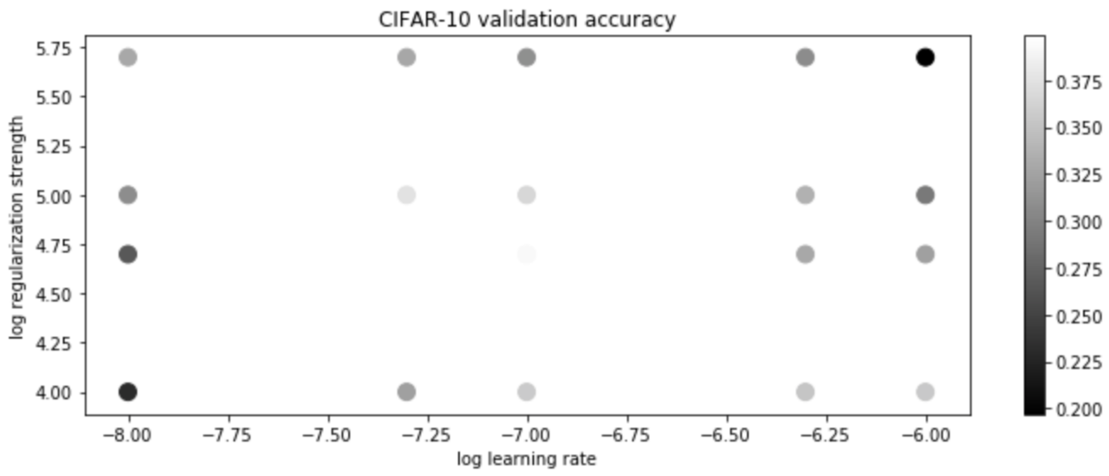
From above, we see that the computing speed of vectorized loss and gradient is 10 times faster than the speed of naive loss and gradient. Also, the difference between the results of each version is very close to zero. To train a SVM with mini-batch gradient descent (1500 iterations), it took around 11.5 seconds.

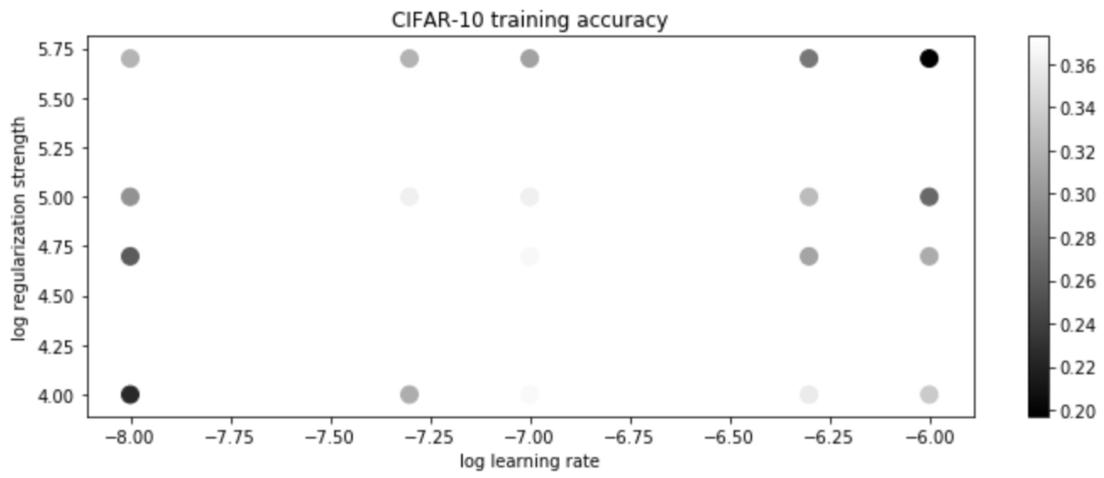
We can confirm the results as iterations go up, the loss goes down and finally converge.



C)
training accuracy: 0.367367
validation accuracy: 0.372000

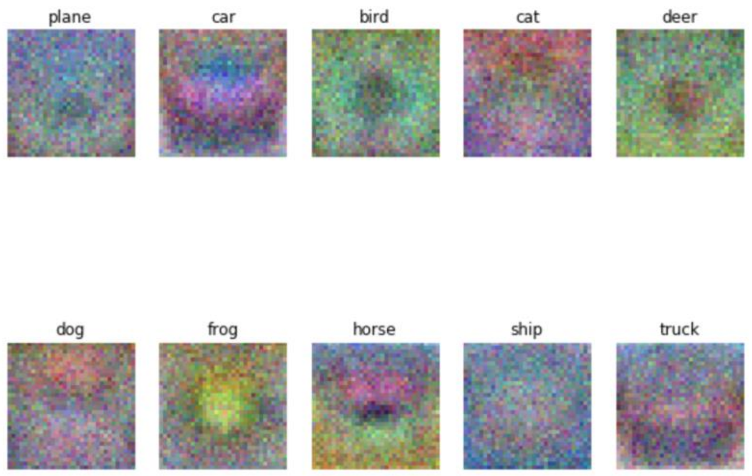
D)
best validation accuracy achieved:
validation accuracy= 0.399000, learning rate= 5.000000e-08, Regularization= 5.000000e+04, train accuracy=0.372776, iterations=1500.

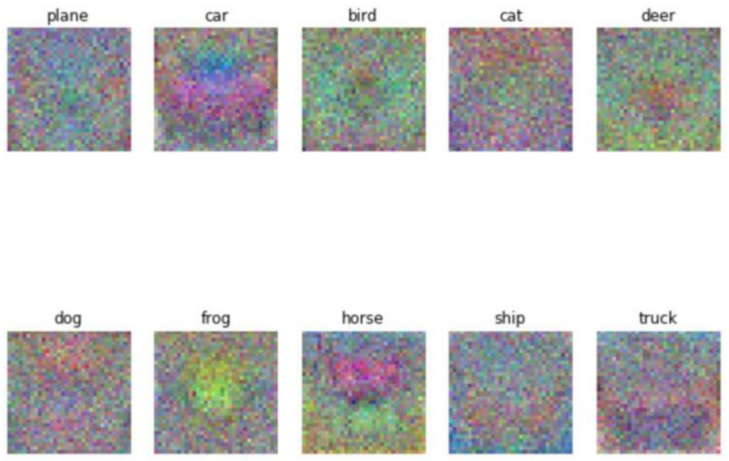




And linear SVM on raw pixels final test set accuracy is: 0.371700

E) The first one is from hw4(SVM), and the second one is from hw3(Softmax). By looking at them visually. There is no much differentiation.





For this problem, Multi-class SVM takes longer time to train but doesn't achieve higher performance on classifying the test set than softmax regression.

From hw3, we obtained that (batch size=1200, number of iteration= 2000, learning rate=1e-6, regularization parameter= 500000) is the best combination for this problem and the best test set accuracy = 0.419.

From this time(hw4), we obtained that (batch size=400, number of iteration= 1500, learning rate=5e-8, regularization parameter= 5e+4) is the best combination for this problem and the best test set accuracy = 0.3717

The hyper parameter selections shown above are different as softmax regression is discrimitive. And for this problem, softmax regression overall show a higher test set accuracy and faster computation speed.