COMP540-HW4-Yunyi Lin&Pei Zeng

Yunyi Lin&Pei Zeng

3/5/2018

1 Intuitions about support vector machines

1.1

Since the bigger margin will give a clearer division of points. In general cases, we can get the most certain classification for every point (best performance) by maximizing the margin. Moreover, it's a noisy process when we generated the data, if the boundary was set to be close to one of our training points of some class, it's very possible that a point of the same class will be generated across the boundary, resulting in an incorrect classification. Therefore, it makes sense to make the boundary as far away from our training points as possible.

1.2

No. According to the definition of hinge loss function $J(\theta, \theta_0) = C\sum_{i=1}^{m} \max(0,1 - y^{(i)}h_\theta(x^{(i)}))$, the vectors that are not support vectors will cause right hand max function be negative. Moving the point further away from the decision boundary only change its values but the symbol will stay negative and the function will still return zero. So the SVM's hinge loss won't be effected.

2 Fitting an SVM classifier by hand

2.1

Because the feature vector is $\phi(x) = (1, \sqrt{2}x, x^2)$, then after mapping, the two data points would be

$$\phi(x^{(1)}) = (1,0,0),\ \ \phi(x^{(2)}) = (1,2,2)$$

The θ vector is perpendicular to decision boundary, which can be (0, 2, 2).

2.2
Since both points are support vectors, in this model, the distance to boundary is half of distance between these two points, which is

$$\frac{1}{2}\sqrt{(1-1)^2+(2-0)^2+(2-0)^2}=\sqrt{2}$$

2.3
Since the margin is equal to $\frac{1}{\|\theta\|}$we can set $\frac{1}{\|\theta\|}=\sqrt{2}$ then solve for $\|\theta\| = \frac{\sqrt{2}}{2}$

Thus $(\theta_1^2+\theta_2^2+\theta_3^2)=\frac{1}{2}$, combined with what we got from 2.1 we know that $\theta_1 = 0 \; and \; \theta_2 = \theta_3$

Therefore, we can slove the problem above and have $\theta = (0, \frac{1}{2}, \frac{1}{2})$

2.4
Solving for θ_0 based on the two inequalities we have:

$$y^{(1)}(\theta^T\phi(x^{(1)})+\theta_0)\geq 1$$

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

By plugging the data and θ calculated before we got:

$$-\left(0*1+\frac{1}{2}*0+\frac{1}{2}*0+\theta_0\right)\geq 1,\;\; \theta_0\leq -1$$

$$\left(0*1+\frac{1}{2}*2+\frac{1}{2}*2+\theta_0\right)\geq 1,\;\; \theta_0\geq -1$$

Therefore, we get $\theta_0 = -1$

2.5
The decision boundary is $(\theta_0+\theta^T(\phi(x)))=0$, which in this problem is $-1+\frac{x}{\sqrt{2}}+\frac{x^2}{2}=0$.

3 Support vector machines for binary classification

3.1: Support vector machines

3.1A: The hinge loss function and gradient (5 points)

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

According to the output, the cost J is 1.0 and the gradient of the loss function with respect to an all-zeros θ vector is $[-0.12956186 - 0.00167647]^T$.

3.1B Example dataset 1: impact of varying C (2 points)

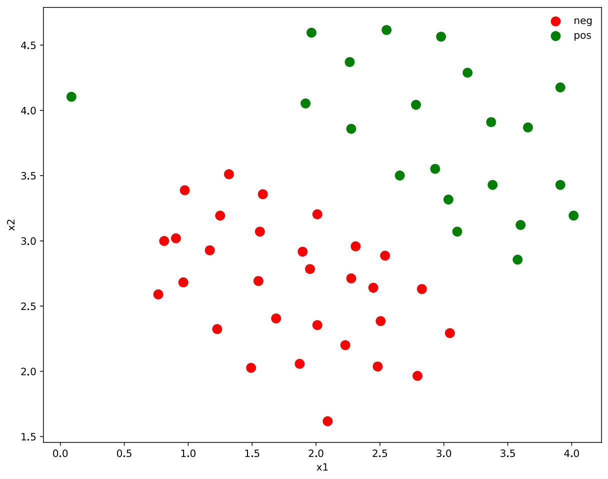


Figure 1: Example dataset1

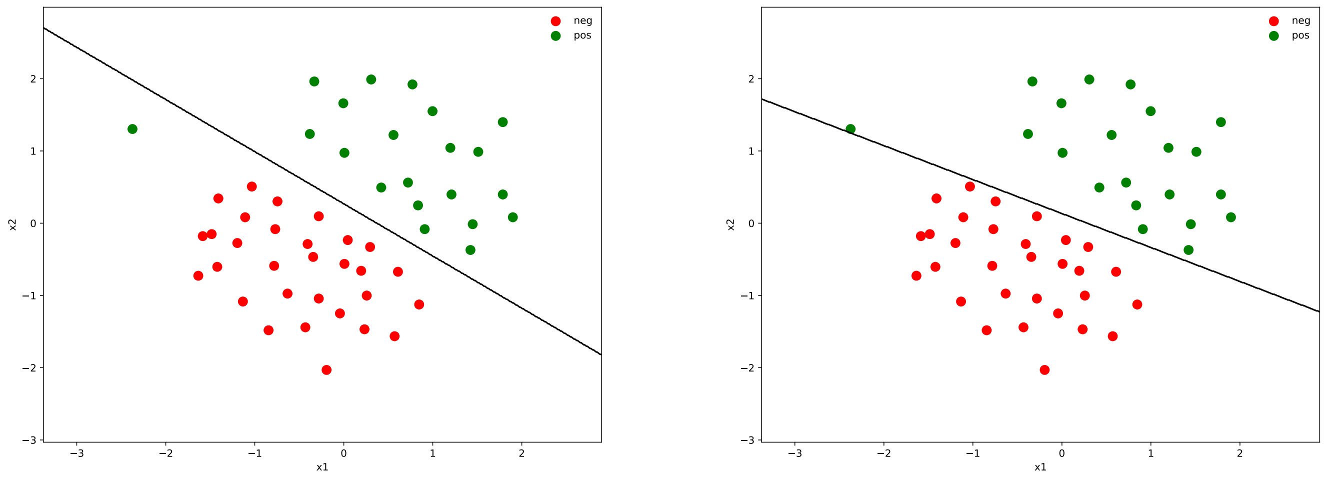


Figure 2: (left) SVM decision boundary with C = 1 for Example dataset 1; (right) SVM decision boundary with C = 100 for Example dataset 1

We can see that, when C = 100, the SVM classifies every single example correctly, but has a decision boundary that does not appear to be a natural fit for the data.

SVM with Gaussian kernels

3.1C Gaussian kernel (3 points)

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

From the output we confirmed that the Guassian kernel value to be 0.324652467358

Example dataset 2: learning non-linear boundaries

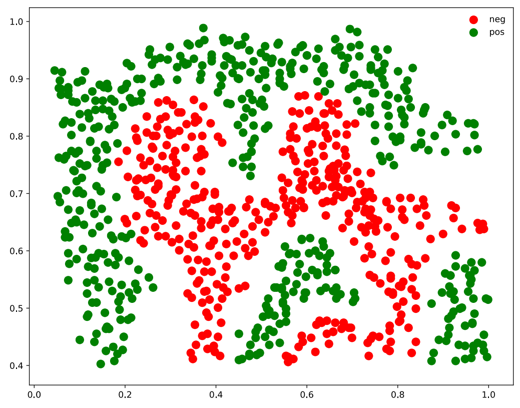


Figure 4 : Example dataset 2

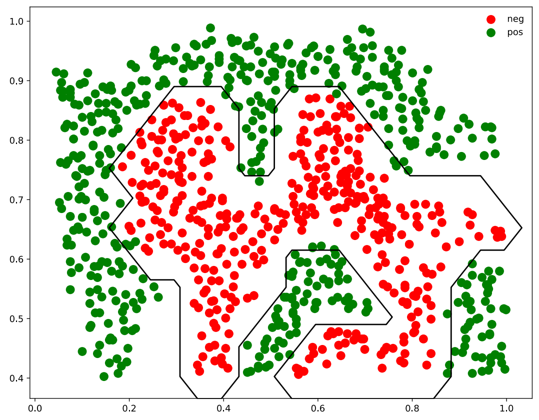


Figure 5: SVM gaussian kernel decision boundary for Example dataset 2, C = 1 and σ = 0.02

3.2 Example dataset 3: selecting hyper parameters for SVMs (5 points)

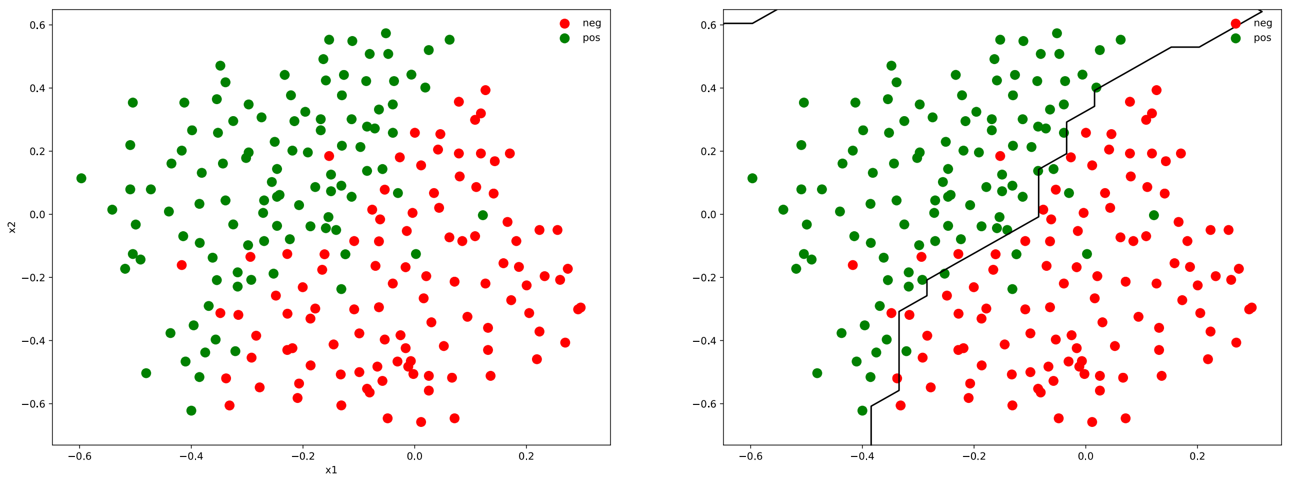


Figure 6:* （left） *Example dataset 3 ; (right) SVM gaussian kernel decision boundary for Example dataset 3 with the best hyper parameters.

```
best C =  0.3 best sigma =  0.1 accuracy = 0.96
```

The best hyper parameters we obtained is C = 0.3 and σ = 0.1, which gave us an accuracy around 96%.

3.3 Spam Classification with SVMs (10 points)

We separated 4000 original training set into 3200 training set and 800 validation set. Moreover, we think the Linear kernel is best for this problem since the data is almost 0s and 1s so we use Linear kernel for this problem.

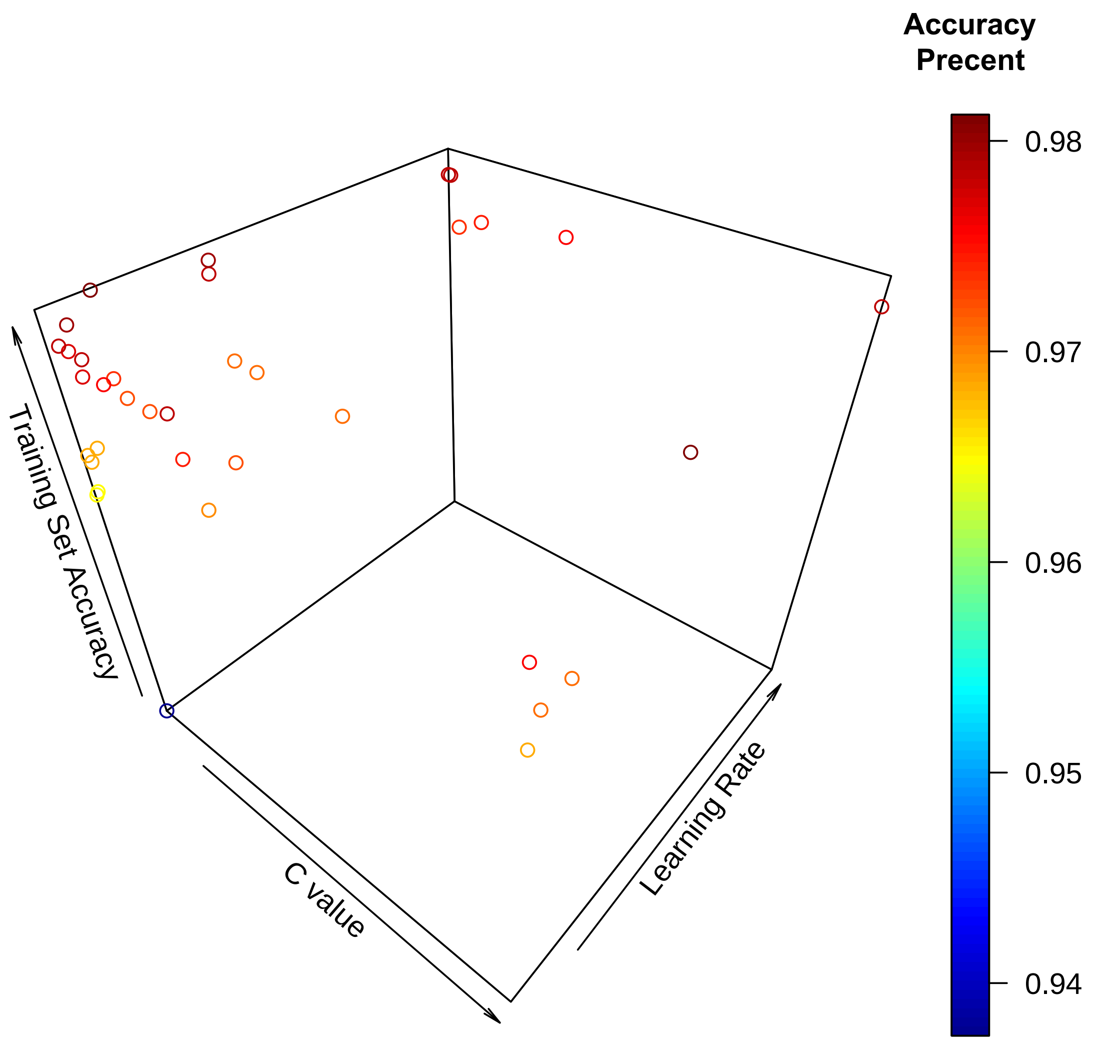
Our algorithm goes like this, basically we set 6 values C and learning rates and using the 10,000 with batch size = 500.

Specifically, we have:

C values = [0.1, 0.3, 1.0, 3.0, 10.0, 30.0],

learning rates = [1e-3, 3e-3, 1e-2, 3e-2, 1e-1, 3e-1]

The following is our graph on how to pick the C value and learning rate and our output for the best hypermeters combination.



```
For linear kernel:
the best acc =  0.98125
the best C =  0.3
the best learning rate =  0.03
the best iteration =  10000
Accuracy of model on test data =  0.987
```

We determine that (C = 0.3 ,learning rate= 0.03) is the best combination under 10,000 iterations. Our best classifier gets a training accuracy of about 98.1% and a test accuracy of about 98.7%.

4 Support vector machines for multi-class classification

4A: Loss and gradient function for multi-class SVM – naive version (5 points)

```
loss: 8.839014
grad:  [[ -1.18305831e+01  -7.44894086e+00   4.23530792e+00 ...,  -6.67931080e+00
   -1.32013891e+01  -3.62710131e+01]
 [ -2.22176940e+01  -3.71548700e+00   3.54038588e+00 ...,  -7.91691110e+00
   -2.37573875e+01  -4.11718038e+01]
 [ -4.21031607e+01  -5.06578466e+00   1.51590063e+01 ...,  -8.02772137e+00
   -4.25758117e+01  -5.21712285e+01]
 ...,
 [ -7.48837628e+00  -6.52358876e+00  -3.28540500e+00 ...,  -9.97843722e+00
    1.60267192e+01  -8.75489228e+00]
 [ -2.01141746e+01  -1.33457092e+01   3.83403120e+00 ...,   4.05828493e+00
   -2.92979473e+00  -1.25889414e+01]
 [  2.12244898e-02   2.38775412e-03  -2.24489804e-02 ...,  -1.83673662e-03
    1.23061226e-02  -6.02040821e-03]]
```

Question: 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.

Answer: The discrepancy is caused by the SVM loss function is not differentiable at some points. But it should not be a concern since compare the large compute times, these few points that cannot be differentiated won't have a significant influence on results

4B: Loss and gradient function for multi-class SVM – vectorized version (10 points)

```
iteration 0 / 1500: loss 790.816246
iteration 100 / 1500: loss 287.769245
iteration 200 / 1500: loss 107.786691
iteration 300 / 1500: loss 43.062693
iteration 400 / 1500: loss 18.780533
iteration 500 / 1500: loss 10.617934
iteration 600 / 1500: loss 7.093434
iteration 700 / 1500: loss 5.843449
iteration 800 / 1500: loss 5.546236
iteration 900 / 1500: loss 5.800153
iteration 1000 / 1500: loss 5.732845
iteration 1100 / 1500: loss 5.445107
iteration 1200 / 1500: loss 5.299464
iteration 1300 / 1500: loss 4.757723
iteration 1400 / 1500: loss 5.499645
That took 4.534814s
```

A cell in the multiclass svm.ipynb notebook sets up a mul- ticlass svm instance and trains it with a specific learning rate and regularization strength for 1500 iterations. And it took us 4.534815 to train an SVM on this data set.

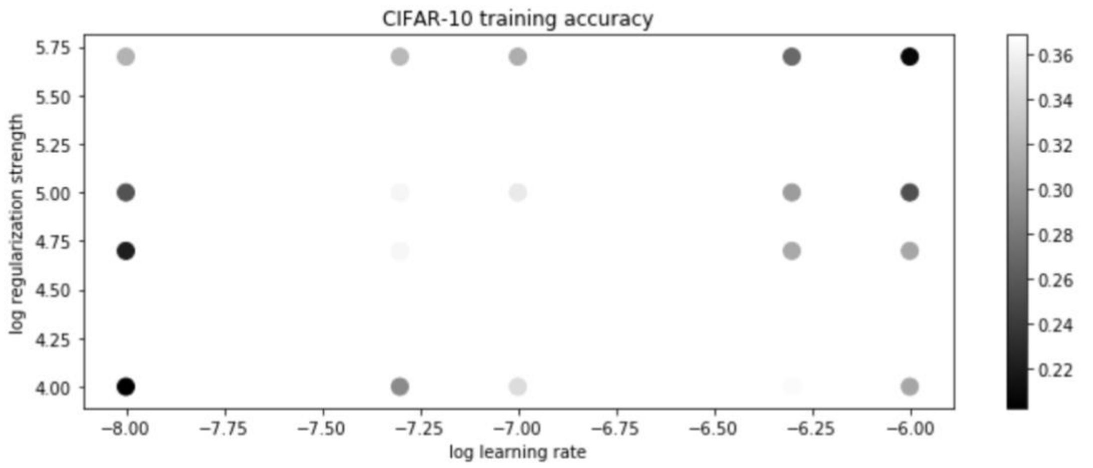
4C: Prediction function for multi-class SVM (5 points)

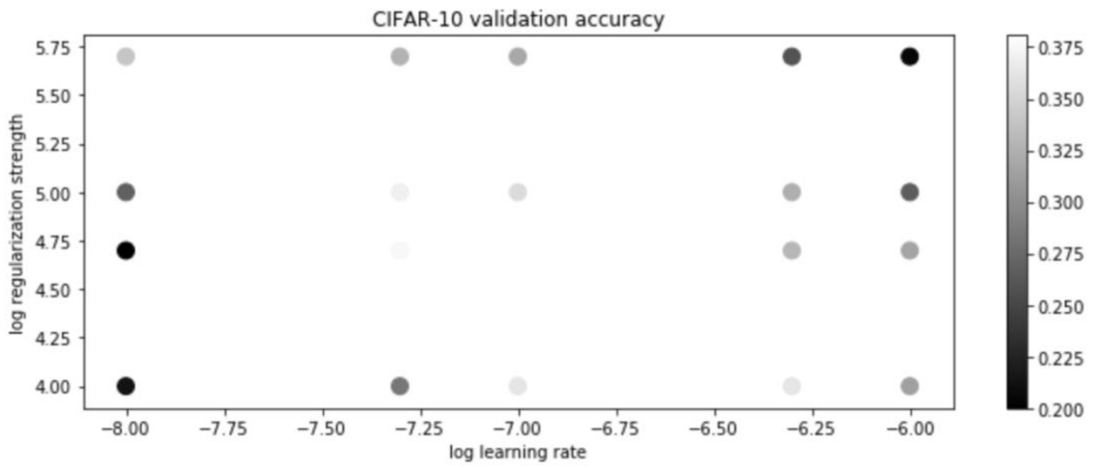
```
training accuracy: 0.363163
validation accuracy: 0.382000
```

A cell in the multiclass svm.ipynb notebook evaluated the performance of your SVM learned in the previous step on both the training and validation set. You got about 36.3% accuracy on training set and 38.2% accuracy on validation set.

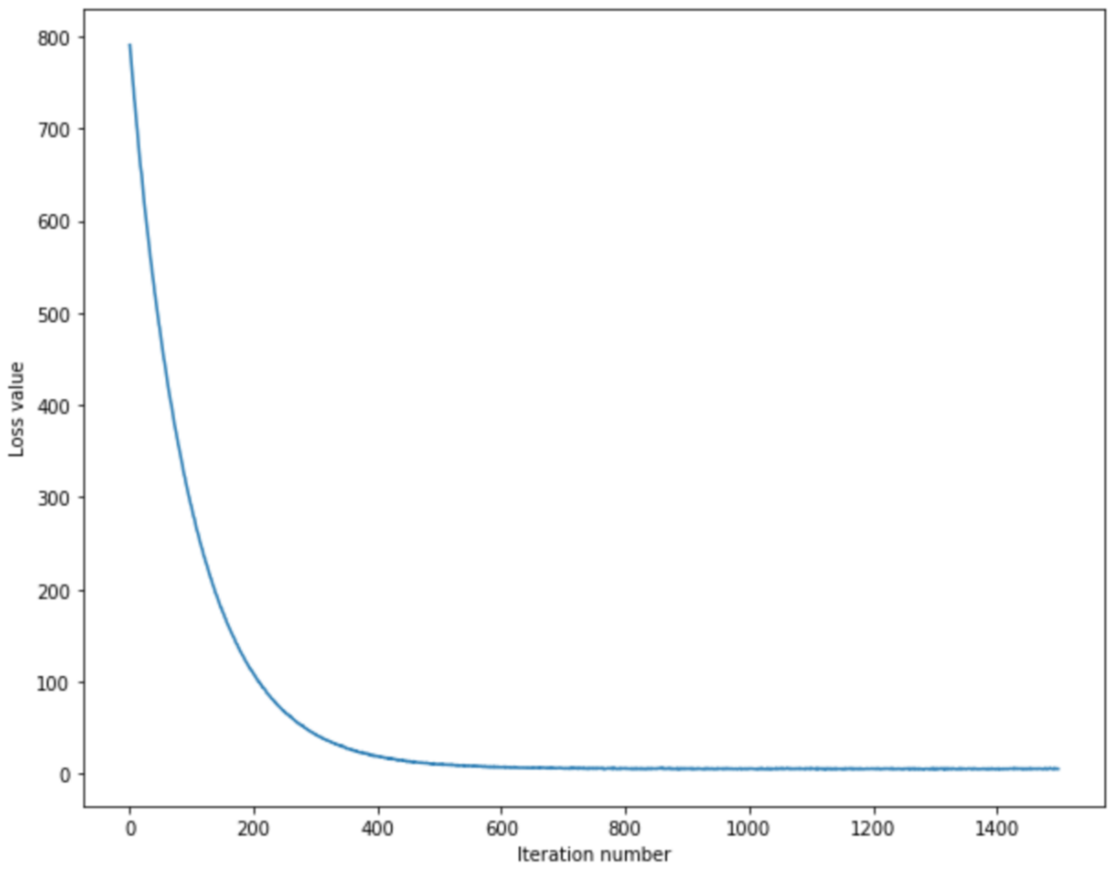
4D: Tuning hyper parameters for training a multi-class SVM (10 points)

```
the best learning rate =  1e-07
the best reg =  50000.0
best validation accuracy achieved during cross-validation: 0.381000
```





The best regularization strength we chose is 5e+04 with the best learning rate 1e-07 and the best validation accuracy achieved during cross-validation is 38.1%.

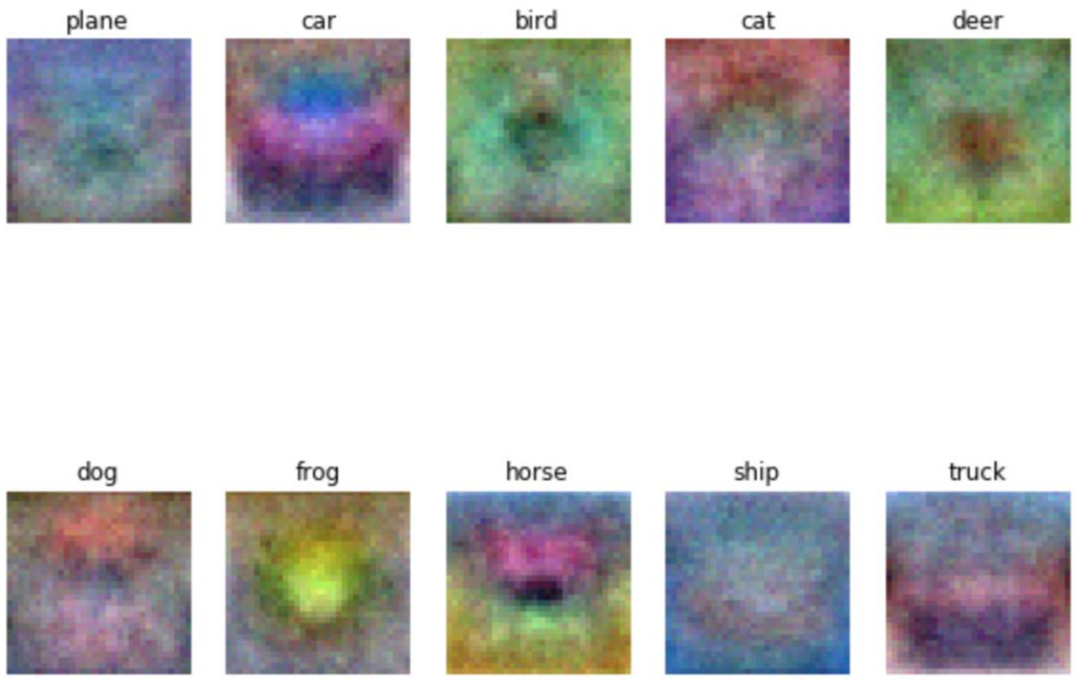


From the plot above, we can see that as iterations go up, the loss goes down and finally converges.

```
linear SVM on raw pixels final test set accuracy: 0.365400
```

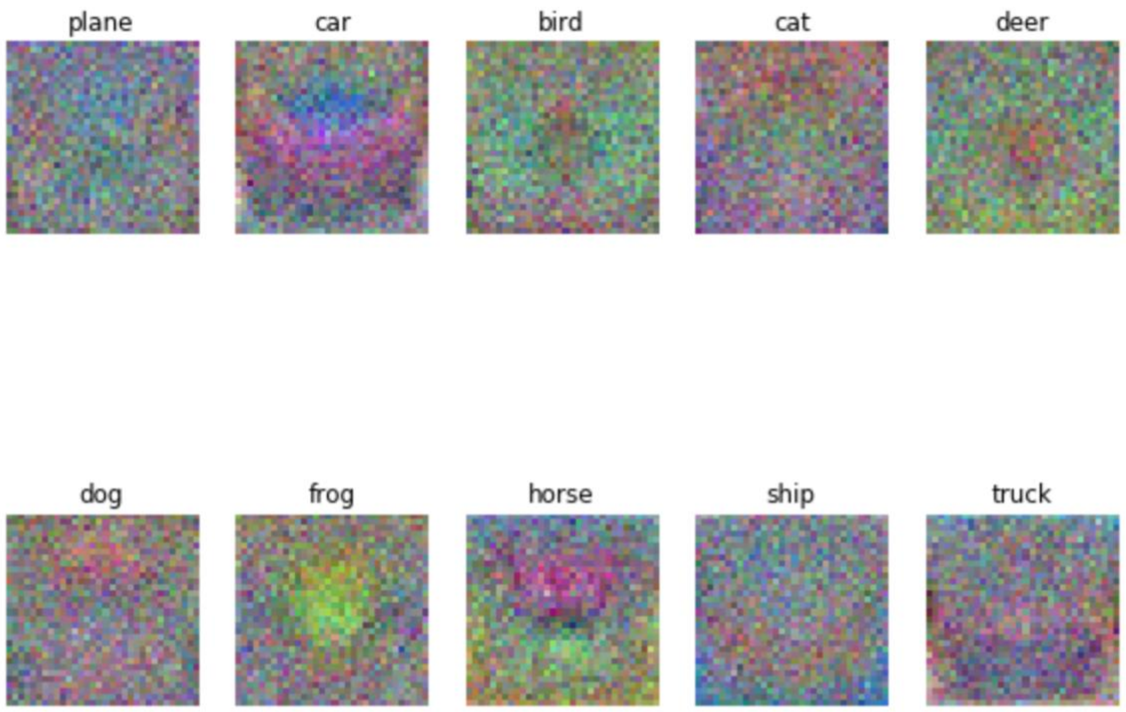
Our linear SVM on raw pixels final test set accuracy is around 36.5%.

4E: Comparing the performance of multi-class SVM and softmax regression (5 points)



multi-class SVM:

As we can see from above, the best regularization strength we chose is 5e+04 with the best learning rate 1e-07 and the best validation accuracy achieved during cross-validation is 38.1%. Moreover, Our linear SVM on raw pixels final test set accuracy is around 36.5%.



softmax regression:

```
lr 5.000000e-07 reg 5.000000e+05 train accuracy: 0.410837 val accuracy:
0.412000
best validation accuracy achieved during cross-validation: 0.412000
softmax on raw pixels final test set accuracy: 0.403200
```

Based on the output, we can see the best regularization strength we chose is 5e+05 with the best learning rate 5e-07 and the best validation accuracy achieved during cross-validation is 41.2%. Moreover, softmax on raw pixels final test set accuracy is around 40.3%.

Discussion:

Multi-class SVM takes longer time to train but doesn't achieve higher performance on classifying the test set than Softmax regression. Overall, Softmax regression show a higher test set accuracy and faster computation speed and I think Softmax regression might be a better method considing this problem than SVM.