Assignment 3（Writeup）

Created By:

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

Feb 16, 2018

COMP 540 – Statistical Machine Learning

Rice University

Problem 1)

a)

$$We\ have\ X \sim Bernoulli(\theta). Thus, P\left(x^{(i)} = 1\right) = \theta, P\left(x^{(i)} = 0\right) = 1 - \theta$$

With i.i.d. assumption, we have $L(D|\theta) = \prod_{i=1}^{m} P\left(x^{(i)}; \theta\right) = \prod_{i=1}^{m}[\theta^{x^{(i)}}(1-\theta)^{1-x^{(i)}}]$

Then $Log\left(L(D|\theta)\right) = \sum_{i=1}^{m}[x^{(i)}\log(\theta) + \log(1-\theta)(1-x^{(i)})]$

Taking derivative of $Log\left(L(D|\theta)\right)$, we get $\sum_{i=1}^{m}[x^{(i)}\frac{1}{\theta} - \frac{1}{1-\theta}(1-x^{(i)})]$, we set it equaling to 0, we get the $\theta_{MLE} = \frac{1}{m}\sum_{i=1}^{m} x^{(i)}$.

Thus, the MLE of θ is the proportion of examples with $x^{(i)} = 1\ in\ D$.

b)

$$Beta(\theta|a,b) \propto P(D|\theta)P(\theta)$$
$$\propto P(D|\theta)P(\theta|a,b)$$
$$\theta_{MAP} = argmax_{\theta}\ P(D|\theta)P(\theta|a,b)$$
$$\mathrm{L} = \log[P(D|\theta)P(\theta|a,b)] = \sum_{i=1}^{m} \log P\left(x^{(i)}|\theta\right) + logP(\theta|a,b)$$
$$= \sum_{i=1}^{m}\left[x^{(i)}\log\theta + \left(1 - x^{(i)}\right)\log(1-\theta)\right] + (a-1)\log\theta + (b-1)\log(1-\theta)$$
$$= (\sum_{i=1}^{m} x^{(i)} + a - 1)\ log\theta + (\sum_{i=1}^{m}(1 - x^{(i)}) + b - 1)\log(1-\theta)$$

Then we take the derivative and set it to 0, we get $\theta_{MAP} = \frac{\sum_{i=1}^{m} x^{(i)} + a - 1}{m + a + b - 2}$

Then set a=b=1, $\theta_{MAP} = \frac{1}{m}\sum_{i=1}^{m} x^{(i)} = \theta_{MLE}$

Problem 2)

a)

$$P(y = 1|x) = g(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}}$$
$$P(y = 0|x) = 1 - g(\theta^T x) = 1 - \frac{1}{1 + e^{-\theta^T x}} = \frac{1}{1 + e^{\theta^T x}}$$

b)

$$y \sim Bernoulli(\gamma)$$
$$x_j|y = 1 \sim N(\mu_j^1, \sigma_j^2)$$

$$x_j|y=0 \sim N(\mu_j^0, \sigma_j^2)$$

$$\mathrm{P(y=1|x)} = \frac{P(x|y=1)P(y=1)}{P(y=1)P(x|y=1)+P(y=0)P(x|y=0)}$$

$$= \frac{\gamma \prod_{j=1}^d N(x_j|\mu_j^1, \sigma_j^2)}{\gamma \prod_{j=1}^d N\left(x_j\middle|\mu_j^1, \sigma_j^2\right) + (1-\gamma)\prod_{j=1}^d N\left(x_j\middle|\mu_j^0, \sigma_j^2\right)}$$

$$\mathrm{P(y=0|x)} = \frac{P(x|y=0)P(y=0)}{P(y=1)P(x|y=1)+P(y=0)P(x|y=0)}$$

$$= \frac{(1-\gamma)\prod_{j=1}^d N\left(x_j\middle|\mu_j^0, \sigma_j^2\right)}{\gamma \prod_{j=1}^d N\left(x_j\middle|\mu_j^1, \sigma_j^2\right) + (1-\gamma)\prod_{j=1}^d N\left(x_j\middle|\mu_j^0, \sigma_j^2\right)}$$

where $N\left(x_j\middle|\mu_j^1, \sigma_j^2\right) = \frac{1}{\sqrt{2\pi}\sigma_j}\exp(-\frac{(x_j-\mu_j^1)^2}{2\sigma_j^2}), N\left(x_j\middle|\mu_j^0, \sigma_j^2\right) = \frac{1}{\sqrt{2\pi}\sigma_j}\exp(-\frac{(x_j-\mu_j^0)^2}{2\sigma_j^2})$

c)

Uniform priors means that we have $\gamma = 0.5$, then we have the posterior for Gaussian NB:

$$\mathrm{P(y=1|x)} = \frac{\gamma \prod_{j=1}^d N(x_j|\mu_j^1, \sigma_j^2)}{\gamma \prod_{j=1}^d N\left(x_j\middle|\mu_j^1, \sigma_j^2\right) + (1-\gamma)\prod_{j=1}^d N\left(x_j\middle|\mu_j^0, \sigma_j^2\right)}$$

$$= \frac{(\sqrt{2\pi})^{-d}\prod_{j=1}^d \sigma_j^{-1}\exp(-0.5\sum_{i=1}^d \frac{(x_j-\mu_j^1)^2}{\sigma_j^2}))}{(\sqrt{2\pi})^{-d}\prod_{j=1}^d \sigma_j^{-1}\exp(-0.5\sum_{i=1}^d \frac{(x_j-\mu_j^1)^2}{\sigma_j^2})) + (\sqrt{2\pi})^{-d}\prod_{j=1}^d \sigma_j^{-1}\exp(-0.5\sum_{i=1}^d \frac{(x_j-\mu_j^0)^2}{\sigma_j^2}))}$$

$$= \frac{\exp(-0.5\sum_{i=1}^d \frac{(x_j-\mu_j^1)^2}{\sigma_j^2})}{\exp\left(-0.5\sum_{i=1}^d \frac{(x_j-\mu_j^1)^2}{\sigma_j^2}\right) + \exp(-0.5\sum_{i=1}^d \frac{(x_j-\mu_j^0)^2}{\sigma_j^2})}$$

$$= \frac{1}{1+\exp(-0.5\sum_{i=1}^d \frac{(x_j-\mu_j^0)^2}{\sigma_j^2} + 0.5\sum_{i=1}^d \frac{(x_j-\mu_j^1)^2}{\sigma_j^2})}$$

$$= \frac{1}{1+\exp(\sum_{i=1}^d \frac{-(\mu_j^1-\mu_j^0)^2}{\sigma_j^2} + 0.5\sum_{i=1}^d \frac{(\mu_j^1)^2-(\mu_j^0)^2}{\sigma_j^2})}$$

The above is for logistic regression.

$$\mathrm{P(y=1|x)} = \frac{1}{1+e^{-\theta^T x}} = \frac{1}{1+\exp(-\theta_0 - \sum_{i=1}^d \theta_j x_j)}$$

Now we let $\theta_0 = 0.5\sum_{i=1}^{d}\frac{(\mu_j^1)^2-(\mu_j^0)^2}{\sigma_j^2}$, then $\theta_j = \frac{\mu_j^1-\mu_j^0}{\sigma_j^2}$ where j= 1, ….. , d

Thus, we see that the Gaussian NB can be written as

$\mathrm{P}(\mathrm{y}=1|\mathrm{x}) = \frac{1}{1+\exp(-(\theta_0+\sum_{j=1}^{d}\theta_j x_j))} = \frac{1}{1+e^{-\theta^T x}}$ which is equivalent to $\mathrm{P}(\mathrm{y}=1|\mathrm{x})$ for logistic regression with parameterization.

Problem 3)

a)

If we reject, the loss will be

$$L_\gamma = \sum_{k=1}^{C} L(\alpha_{C+1}|y=k)P(y=k|x) = \lambda_\gamma \sum_{k=1}^{C} P(y=k|x) = \lambda_\gamma$$

If we decide that y=j, then the loss will be

$$L_j = \sum_{k=1}^{C} L(\alpha_j|y=k)P(y=k|x) = 0 * P(y=j|x)\sum_{k\neq j}^{C} \lambda_S P(y=k|x)$$

$$= \lambda_S[1-P(y=j|x)]$$

as j is the most probable class.

If $\mathrm{P}(\mathrm{y}=\mathrm{j}|\mathrm{x}) \geq 1-\frac{\lambda_\gamma}{\lambda_S}$,

$$L_j = \lambda_S[1-P(y=j|x)]$$
$$\leq \lambda_S[1-(1-\frac{\lambda_\gamma}{\lambda_S})]$$
$$\leq \lambda_\gamma = L_\gamma$$

So, the minimum risk is obtained if we decide that y=j since it is smaller than the loss of the one from the rejection, and it is also smaller than if we decide that y=k where k≠j. On the other hand, if $\mathrm{P}(\mathrm{y}=\mathrm{j}|\mathrm{x}) \leq 1-\frac{\lambda_\gamma}{\lambda_S}$, we have $L_j \leq L_\gamma$, then we would decide to reject.

b)

$If\ \frac{\lambda_\gamma}{\lambda_S} = 0$, then $1-\frac{\lambda_\gamma}{\lambda_S} = 1$.

Since $\mathrm{P}(\mathrm{y}=\mathrm{j}|\mathrm{x}) \leq 1$, then we always reject.

$If\ \frac{\lambda_\gamma}{\lambda_S} = 1$, then $1-\frac{\lambda_\gamma}{\lambda_S} = 0$.

Since $\mathrm{P}(\mathrm{y}=\mathrm{j}|\mathrm{x}) \geq 0$, then we never reject.

As $\frac{\lambda_\gamma}{\lambda_S}$ increases form 0 to 1, the class tends to be less likely to reject. Since only for class j where $P(y = j|x) \leq 1 - \frac{\lambda_\gamma}{\lambda_S}$, we decide that y = j, the value of $\frac{\lambda_\gamma}{\lambda_S}$ performs as a threshold to control how high the probability $P(y = j|x)$ should be in order to decide y = j in our classifier.

Problem 4) The Euclidean distance between new vector x and the other point $x^{(i)}$ in D is $dis(x, x^{(i)}) = \sqrt{\sum_{j=1}^{d}(x_j - x_j^{(i)})^2}$

We consider $dis^2(x, x^{(i)}) = \sum_{j=1}^{d}(x_j - x_j^{(i)})^2$

As dot products $= x^T x - 2x^T x^{(i)} + (x^{(i)})^T x^{(i)}$

Kernelize it $= k(x, x) - 2k(x, x^{(i)}) + k(x^{(i)}, x^{(i)})$

Specifically we can use Gaussian kernel, which satisfies Mercer condition because Gram matrix with elements $k(x, x') = \exp(-\frac{||x-x'||^2}{2\sigma^2})$ is positive definite.

Then the classification rule shall be based on

$$dis^2(x, x^{(i)}) = exp\left(-\frac{||x - x||^2}{2\sigma^2}\right) - 2\exp\left(-\frac{||x - x^{(i)}||^2}{2\sigma^2}\right)$$

$$+ \exp\left(-\frac{||x^{(i)} - x^{(i)}||^2}{2\sigma^2}\right) = 2 - \exp\left(-\frac{||x - x^{(i)}||^2}{2\sigma^2}\right)$$

Problem 5)

a)

$$k(x, x') = ck_1(x, x') \; where \; k_1 \; is \; a \; valid \; kernel$$

The Gram matrix of $k_1 = K_1$ which is positive definite.

So, $\forall \, u \neq 0, u^T K_1 u > 0$.

We have assumption c>0, then the Gram matrix of k would be $k = k_1$, and $\forall \, u \neq 0$, $u^T K_1 u = cu^T K_1 u > 0$

So, K is also positive definite, which means that the new kernel $k(x, x^{(i)})$ is also valid.

b)

$$k\left(x, x^{(i)}\right) = f(x)k_1(x, x')f(x'), k_1 \; is \; valid$$

$$\text{Also } k_1(x, x') = \phi_1(x)^T \phi_1(x')$$
$$k(x, x') = f(x)k_1(x, x')f(x') = f(x)\phi_1(x)^T \phi_1(x')f(x')$$

Do feature map,
$\phi: x \to f(x)\phi_1(x)$
where $\phi(x) = f(x)\phi_1(x), k(x, x') = \phi(x)^T \phi(x')$, which implies $k(x, x') \; is \; valid.$

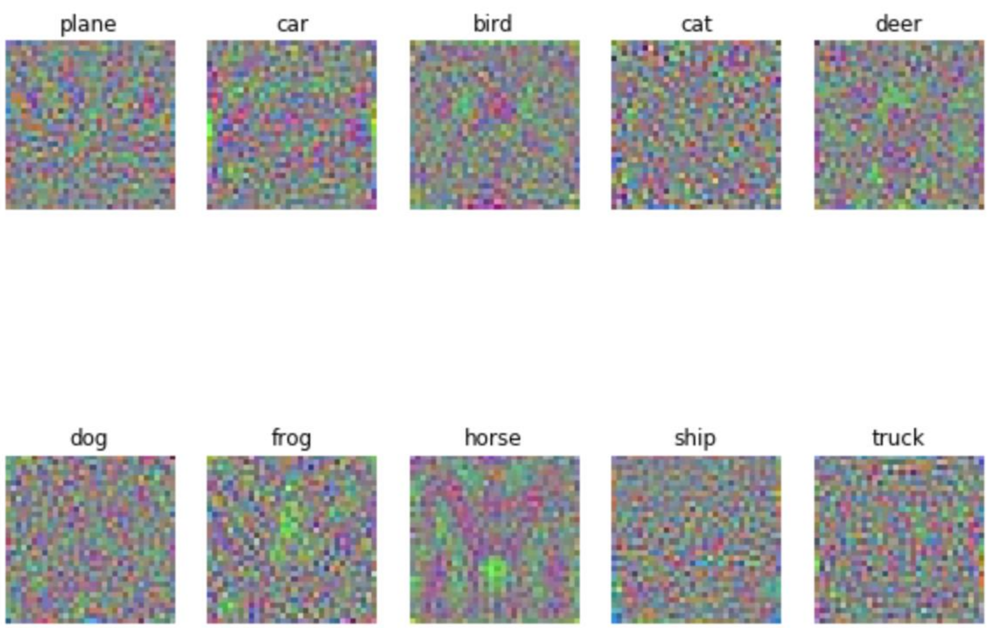
c)
$k(x, x') = k_1(x, x') + k_2(x, x'), \; where \; k_1 \; and \; k_2 \; are \; valid.$
Gram matrix of $k_1, k_2 \; are \; K_1, K_2 \;, respectively.$
$K_1, K_2 \; are \; positive \; definite$
So
$\forall \; u \neq 0, u^T K_1 u > 0, u^T K_2 u > 0, Gram \; matrix \; of \; k(x, x') \; is \; K = K_1 + K_2,$
$\forall \; u, u^T K_1 u = u^T (K_1 + K_2) u = u^T K_1 u + u^T K_2 u > 0$

Problem 6) OVA logistic regression

1. Implementing a OVA classifier for the CIFAR-10 dataset

```
one_vs_all on raw pixels final test set accuracy: 0.362000
[[465  59  21  24  19  35  26  60 201  90]
 [ 67 465  18  35  23  31  44  50  94 173]
 [123  65 193  77  96  89 151  89  69  48]
 [ 66  86  78 161  49 193 171  51  62  83]
 [ 65  38 103  64 234  90 194 128  36  48]
 [ 48  63  81 126  81 272 114  88  72  55]
 [ 31  53  67 102  85  78 457  52  29  46]
 [ 53  62  51  46  69  84  66 405  49 115]
 [143  78   8  25   9  34  22  20 547 114]
 [ 59 208  14  22  23  29  60  56 108 421]]
```

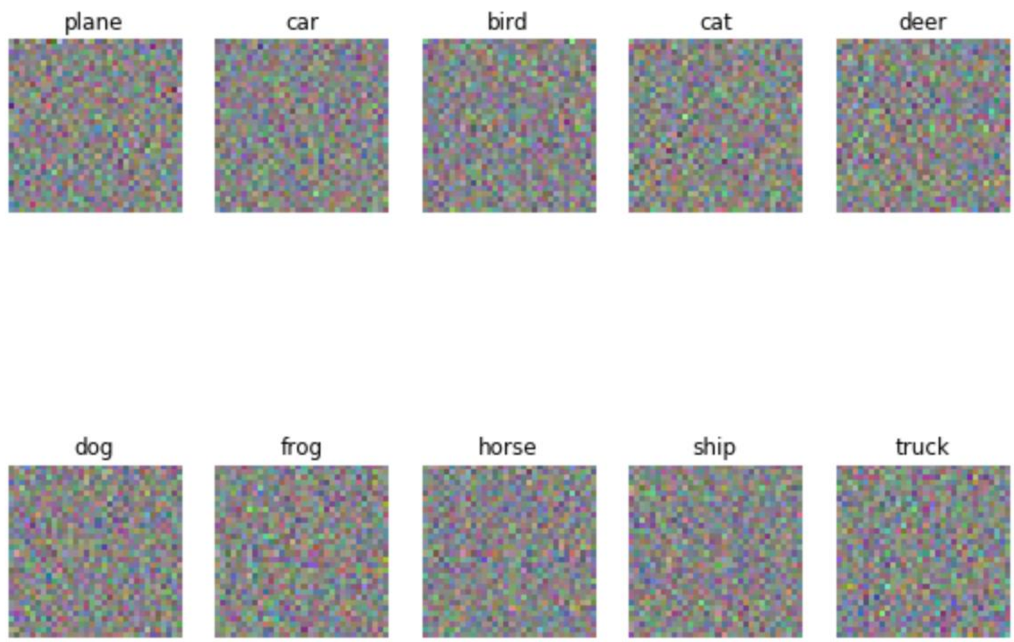
2. Visualizing the learned one-vs-all classifier



3. Comparing our functions with sklearn's

```
one_vs_all on raw pixels final test set accuracy (sklearn): 0.362000
[[465  59  21  24  19  35  26  60 201  90]
 [ 67 465  18  35  23  31  44  50  94 173]
 [123  65 193  77  96  89 151  89  69  48]
 [ 66  86  78 161  49 193 171  51  62  83]
 [ 65  38 103  64 234  90 194 128  36  48]
 [ 48  63  81 126  81 272 114  88  72  55]
 [ 31  53  67 102  85  78 457  52  29  46]
 [ 53  62  51  46  69  84  66 405  49 115]
 [143  78   8  25   9  34  22  20 547 114]
 [ 59 208  14  22  23  29  60  56 108 421]]
```

4. Visualizing the sklearn OVA classifier



The accuracies appear to be the same for our result and from the sklearn OVA classifier. However, our result visualization seems to be more patterned than the visualization using the sklearn OVA classifier. (i.e. the horse looks more like a horse)

7) Softmax regression

1. Implementing the loss function for softmax regression (naive version)

loss: (should be close to 2.38): 2.34867271843

➢ Why are we expected to see a value of about $-log_e(0.1)? Why?$

Training images have an uniform distribution on all the K labels. (i.e. each genre is around 10% of the whole training sets)

For $\sum_{k=1}^{K} I\{y^{(i)} = k\} log \frac{\exp(\theta^{(k)^T} x^{(i)})}{\sum_{j=1}^{K}(\theta^{(k)^T} x^{(j)})}$. We see that only when k=$y^{(i)}$ will make it equal to 1 and the rest are all equaling to 0. Then the probability $\frac{\exp(\theta^{(k)^T} x^{(i)})}{\sum_{j=1}^{K}(\theta^{(k)^T} x^{(j)})}$ will be about or a little above 10%. Thus,

$$J(\theta) = \frac{-1}{m}\sum_{i=1}^{m} log \frac{\exp(\theta^{(k)^T} x^{(i)})}{\sum_{j=1}^{K}(\theta^{(k)^T} x^{(j)})} \approx \frac{-1}{m}\sum_{i=1}^{m} \log(0.1) = -log_e(0.1)$$

2. Implementing the gradient of loss function for softmax regression (naive version)

```
numerical: 0.274933 analytic: 0.274933, relative error: 1.005208e-07
numerical: -0.340997 analytic: -0.340997, relative error: 1.137087e-07
numerical: 0.382215 analytic: 0.382215, relative error: 1.917162e-07
numerical: -0.086363 analytic: -0.086363, relative error: 5.178196e-08
numerical: 2.140955 analytic: 2.140955, relative error: 1.612604e-08
numerical: 0.327284 analytic: 0.327284, relative error: 5.657240e-08
numerical: -2.407549 analytic: -2.407549, relative error: 9.270840e-09
numerical: -3.203358 analytic: -3.203358, relative error: 1.631323e-09
numerical: 1.578628 analytic: 1.578628, relative error: 2.358069e-08
numerical: 1.098961 analytic: 1.098960, relative error: 5.507342e-08
naive loss: 2.348673e+00 computed in 28.301643s
```

3. Implementing the loss function and its gradient for softmax regression (vectorized version)

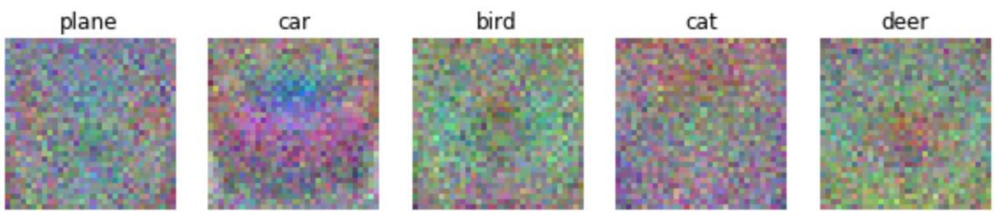
```
vectorized loss: 2.348673e+00 computed in 0.394729s
Loss difference: 0.000000
Gradient difference: 0.000000
```

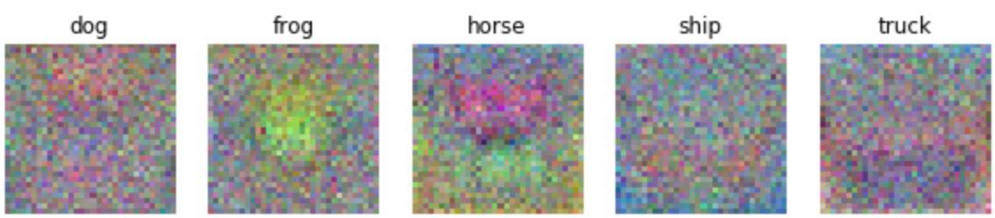
4. Using a validation set to select regularization lambda and learning rate for gradient descent

best validation accuracy achieved during cross-validation: 0.405000

5. Evaluating the best softmax classifier on the test set and visualizing the coefficients

```
softmax on raw pixels final test set accuracy: 0.399900
[[432  48  44  27  15  26  28  35 253  92]
 [ 50 471  19  37  23  39  48  37 108 168]
 [ 91  55 234  77 114  86 180  60  75  28]
 [ 42  65  89 246  46 173 149  53  67  70]
 [ 59  33 130  61 274  73 194 100  43  33]
 [ 41  42  88 146  63 332 108  65  82  33]
 [ 16  47  61  95  78  66 533  27  30  47]
 [ 49  47  67  52  82  69  71 391  57 115]
 [111  69  10  20   7  42  16  14 581 130]
 [ 46 172  14  27  13  16  47  42 118 505]]
```





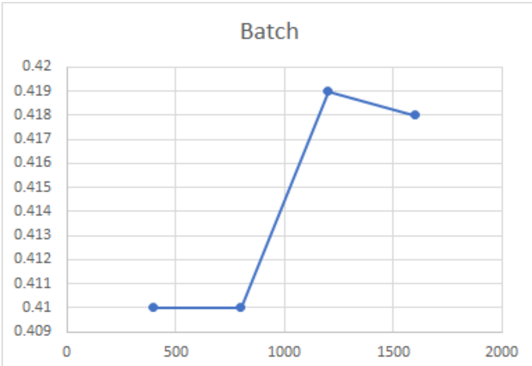
➢ Interpret the visualized coefficients in the light of the errors made by the classifier

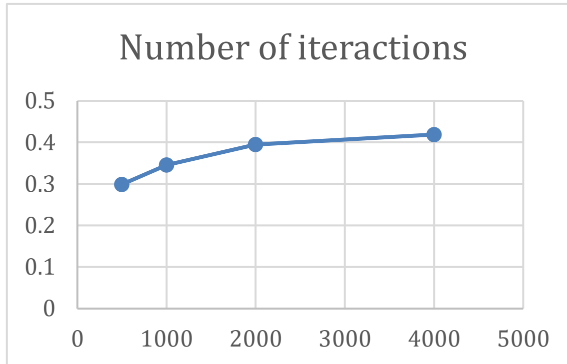
From the confusion matrix, the 432 in the (1,1) position of the matrix and 253 in the (1,9) position of the matrix would mean that 432 images that were correctly identified as planes. However, 253 that that are actually planes are erroneously identified as ships. This is an implication that ship and planes should have some similar image in our visualization. We can validate it by looking at the images in the output for ship and plane are looking quite similar on some extent. Also, we see that 581 in the (9,9) position of the matrix seems to be the highest in the diagonal which means that ship images would most likely be the easiest to be differentiated from others.

6. Extra Credit. Experimenting with other hyper parameters and optimization

method

The accuracy as a function of batch size/number of iterations:





(batch size=1200, number of iteration= 4000, learning rate=1e-6, regularization parameter= 500000) is the best combination for this problem and the best test set accuracy = 0.419.

7. Comparing OVA binary logistic regression with softmax regression

	Plane	Car	Bird	Cat	Deer	Dog	Frog	Horse	Ship	Truck
OVA	465	465	193	161	234	272	457	405	547	421
softmax	432	471	234	246	274	332	533	391	581	505

The table basically takes the value from the confusion matrices of OVA and softmax of their diagonal values that are the true identifications of the categories.

We see that for each method regarding each category, they tend to better than one another or sometimes the other way around. However, one thing that is similar is that their difference never exceeds 100 which means that accuracies do not differ significantly since the two methods are similar since they both train 10 different parameter vectors, one for each digit. And the difference cause would be that instead of training a model to find a single parameter vector each time like OVA, softmax only trains a parameter matrix once. After all, the overall accuracies for softmax regression method seem to have higher accuracies than the ones from OVA by looking at the table above. Thus, I would recommend softmax.