COMP540-HW3-Writeup

Yunyi Lin&Pei Zeng

2/13/2018

1 MAP and MLE parameter estimation

a)

$\mathcal{D} = \{x^{(i)} | 1 \leq i \leq m\}$ where $x^{(i)}$ drawn from a Bernoulli distribution with parameter θ. Moreover, $x^{(i)} = 1$ represents *heads* and $x^{(i)} = 0$ represents *tails*.

Then the likelihood function of Bernoullli trials is as below,

$\mathcal{L}(\theta|x) = \prod_{i=1}^{m} \theta^{x^{(i)}}(1-\theta)^{x^{(i)}}$

Then the log likelihood funciton is as below,

$ln\mathcal{L} = \sum_{i=1}^{m} x^{(i)} ln\theta + \sum_{i=1}^{m}(1-x^{(i)})ln(1-\theta)$

Take the partial derivative of the log likelihood function wrt θ:

$\frac{\partial ln\mathcal{L}}{\partial \theta} = \frac{\sum_{i=1}^{m} x^{(i)}}{\theta} - \frac{\sum_{i=1}^{m}(1-x^{(i)})}{1-\theta}$

Set $frac\partial ln\mathcal{L}\partial\theta = 0$, thus we can get $\hat{\theta} = \frac{\sum_{i=1}^{m} x^{(i)}}{m}$.

b)

According to the MAP method, $P(\theta|x) \propto P(x|\theta)P(\theta)$

Moreover, since we have a beta prior distribution on θ: $Beta(\theta|a,b) \propto \theta^{a-1}(1-\theta)^{b-1}$

Thus we can get $P(\theta|x) \propto \theta^{\sum_{i=1}^{m} x^{(i)}+a-1}(1-\theta)^{\sum_{i=1}^{m}(1-x^{(i)})+b-1}$

Take log of both sides, $lnP(\theta|x) \propto \sum_{i=1}^{m} x^{(i)} ln\theta + \sum_{i=1}^{m}(1-x^{(i)})ln(1-\theta)$

Take the partial derivative of the right side wrt θ and set it equal to zero, we will get:

$\frac{\sum_{i=1}^{m} x^{(i)}+a-1}{\theta} - \frac{\sum_{i=1}^{m}(1-x^{(i)})+b-1}{1-\theta} = 0$

Thus we got $\hat{\theta} = \frac{\sum_{i=1}^{m} x^{(i)}+a-1}{m+a+b-2}$ and we can see that under a uniform prior (Beta distribution with a = b = 1), the MAP and MLE estimates of θ are equal.

2 Logistic regression and Gaussian Naive Bayes

Consider a binary classification problem with dataset $\mathcal{D} = \{(x^{(i)}, y^{(i)}) | 1 \leq i \leq m; x^{(i)} \in \mathfrak{R}^d, y^{(i)} \in \{0,1\}\}$

a)

For logistic regression, the posterior probability for each class 1 is:

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

For logistic regression, the posterior probability for each class 0 is:

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

b)

Derive the posterior probabilities for class 1:

According to the Bayes Rule we know that

$P(y=1|x) = P(y=1|x_{1,\ldots,x_d}) = \frac{P(x_{1,\ldots,x_d}|y=1)P(y=1)}{P(x_{1,\ldots,x_d})}$, where $P(x) = P(x_{1,\ldots,x_d}|y=1)P(y=1) + P(x_{1,\ldots,x_d}|y=0)P(y=0)$

We know that $y \sim Bernoulli(\gamma), x_j|y=0 \sim N(\mu_j^0, \sigma_j^2)$ and $x_j|y=1 \sim N(\mu_j^1, \sigma_j^2)$.

Thus,$P(y=1) = \gamma$ and $P(y=0) = 1-\gamma$

$P(x_{1,\ldots,x_d}|y=0;\mu_j^0,\sigma_j^2) = \prod_{j=1}^d \frac{1}{\sqrt{2\pi\sigma_j}} exp(-\frac{1}{2}(\frac{x_j-\mu_j^1}{\sigma_j})^2)$

$P(x_{1,\ldots,x_d}|y=1;\mu_j^1,\sigma_j^2) = \prod_{j=1}^d \frac{1}{\sqrt{2\pi\sigma_j}} exp(-\frac{1}{2}(\frac{x_j-\mu_j^0}{\sigma_j})^2)$

Therefore,

$$P(y=1|x) = \frac{P(x_{1,\ldots,x_d}|y=1)P(y=1)}{P(x_{1,\ldots,x_d})} = \frac{\gamma\prod_{j=1}^d \frac{1}{\sqrt{2\pi\sigma_j}} exp(-\frac{1}{2}(\frac{x_j-\mu_j^1}{\sigma_j})^2)}{\gamma\prod_{j=1}^d \frac{1}{\sqrt{2\pi\sigma_j}} exp(-\frac{1}{2}(\frac{x_j-\mu_j^1}{\sigma_j})^2) + (1-\gamma)\prod_{j=1}^d \frac{1}{\sqrt{2\pi\sigma_j}} exp(-\frac{1}{2}(\frac{x_j-\mu_j^0}{\sigma_j})^2)}$$

Using the same method we can get the posterior probabilities for class 0:

$$P(y=0|x) = \frac{P(x_{1,\ldots,x_d}|y=0)P(y=0)}{P(x_{1,\ldots,x_d})} = \frac{(1-\gamma)\prod_{j=1}^d \frac{1}{\sqrt{2\pi\sigma_j}} exp(-\frac{1}{2}(\frac{x_j-\mu_j^0}{\sigma_j})^2)}{\gamma\prod_{j=1}^d \frac{1}{\sqrt{2\pi\sigma_j}} exp(-\frac{1}{2}(\frac{x_j-\mu_j^1}{\sigma_j})^2) + (1-\gamma)\prod_{j=1}^d \frac{1}{\sqrt{2\pi\sigma_j}} exp(-\frac{1}{2}(\frac{x_j-\mu_j^0}{\sigma_j})^2)}$$

c)

Assuming that class 1 and class 0 are equally likely, which is $\gamma = 1-\gamma = 0.5$

Then,

$$\begin{aligned} P(y=1|x) &= \frac{1}{1+\frac{(1-\gamma)\prod_{j=1}^d \frac{1}{\sqrt{2\pi\sigma_j}} exp(-\frac{1}{2}(\frac{x_j-\mu_j^0}{\sigma_j})^2)}{\gamma\prod_{j=1}^d \frac{1}{\sqrt{2\pi\sigma_j}} exp(-\frac{1}{2}(\frac{x_j-\mu_j^1}{\sigma_j})^2)}} \\ &= \frac{1}{1+exp(\frac{\sum_{j=1}^d (x_j^2 - 2\mu_j^1 x_j + (\mu_j^1)^2 - x_j^2 + 2\mu_j^0 x_j - (\mu_j^0)^2)}{2\sigma_j^2})} \\ &= \frac{1}{1+exp(\frac{d((\mu_j^1)^2-(\mu_j^0)^2)}{2\sigma_j^2} + \frac{(\mu_j^0-\mu_j^1)}{\sigma_j^2}\sum_{j=1}^d x_j)} \end{aligned}$$

Thus we can set $\theta_0 = \frac{d((\mu_j^0)^2-(\mu_j^1)^2)}{2\sigma_j^2}$ and $\theta_j = \frac{(\mu_j^1-\mu_j^0)}{\sigma_j^2}$,j=1,. . . ,d

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

3 Reject option in classifiers

a)

If point x, j_max is the class j which has the maximum posterior probability, then at point x the risk it is misclassified is $\lambda_s[1 - P(y = j_{max}|x)]$ and the risk it is rejected is λ_r.

On the other hand, if point x, j is not the class which has the maximum posterior probability, then at point x the risk it is misclassified is $\lambda_s[1 - P(y = j|x)] \geq \lambda_s[1 - P(y = j_{max}|x)]$, which shows that we should always take the maximum posterior probability.

So we would pick the minimum of the two risks above: $\lambda_s[1 - P(y = j_{max}|x)]$,$\lambda_r$.

So the minmum riks is obtained if we dicide $\lambda_s[1 - P(y = j|x)] \leq \lambda_r$ which is $P(y = j|x) \leq 1 - \frac{\lambda_r}{\lambda_s}$, otherwise we decide to reject.

b)

i)

If $\lambda_r = 0$, then the probability we decide to reject is $P(y = j|x) \geq 1$ so in this case we will definitely reject.

ii)

If $\lambda_r = 1$, then the probability we decide to reject is $P(y = j|x) \leq 0$ so in this case we will never reject.

iii)

Moreover, as $\frac{\lambda_r}{\lambda_s}$ inscrease the class tends to be less likely to be rejected.

4 Kernelizing k-nearest neighbors

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

We can consider the distance function as dot prodcuts and then kernelize it:

$d(x, x^{(i)}) = \sqrt{\sum_{j=1}^{d}(x_j - x_j^{(i)})^2} = \sqrt{x^T x - 2x^T x^{(i)} + (x^{(i)})^T x^{(i)}} = \sqrt{k(x, x) - 2k(x, x^{(i)}) + k(x^{(i)}, x^{(i)})}$

Specifically we choose to use Gaussian kernel here since it satisfies the Mercer's Theorem that the Gram matrix with elements $k(x, x') = exp(-\frac{||x-x'||}{2\sigma^2})$ is positive definite.

5 Constructing kernels

a)

$k(x, x') = ck_1(x, x')$, where $c \geq 0$

Since $k_1(x, x')$ is a valid kernel, we know that the Gram matrix K_1 is positive semi-definite that $\forall a \in \mathbb{R}^n, a^T K_1 a \geq 0$

Moreover,we got $K = cK_1$ then $\forall a \in \mathbb{R}^n, a^T K a = ca^T K_1 a \geq 0$

Therefore, the Gram matrix K is also a positive semi-definite, which means $k(x, x')$ is also a valid kernel.

b)

$k(x, x') = f(x)k_1(x, x')f(x')$ where $f(x)$ is any function on $x \in \mathbb{R}^d$ and $f : \mathcal{X} \to \mathbb{R}$ which means $f(x)$ is a scalar.

Since $k_1(x, x')$ is valid, we know that $k_1(x, x') = \phi_1(x)^T \phi_1(x')$

$k(x, x') = f(x)k_1(x, x')f(x') = f(x)\phi_1(x)^T \phi_1(x')f(x')$ let $\phi : x \to f(x)\phi_1(x)$

Then we can see $k(x, x') = [f(x)\phi_1(x)^T][\phi_1(x')f(x')] = \phi(x)^T \phi(x')$, therefore $k(x, x')$ is valid.

c)

$k(x, x') = k_1(x, x') + k_2(x, x')$

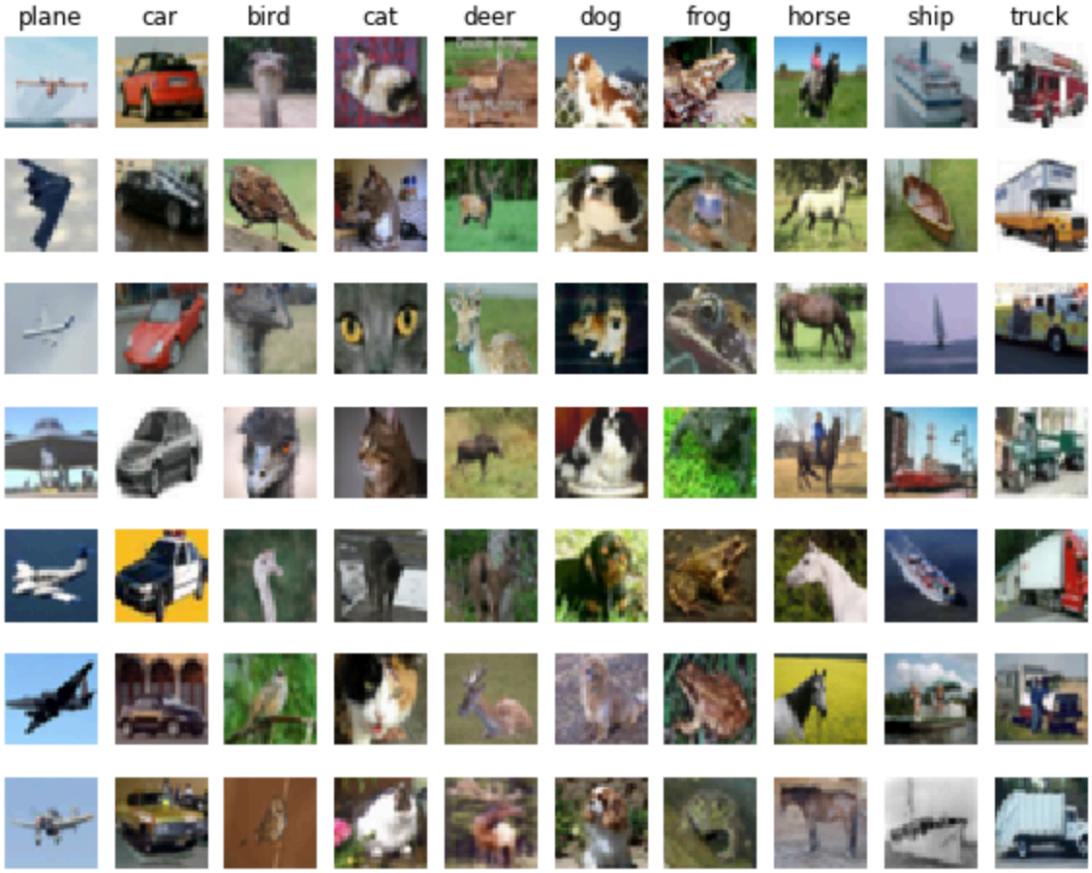
Since $k_i(x, x')i = 1, 2$ are valid kernels, we know that the Gram matrix K_i is positive semi-definite that $\forall a \in \mathbb{R}^n, a^T K_i a \geq 0$

Moreover,we got $K = K_1 + K_2$ then $\forall a \in \mathbb{R}^n, a^T K a = a^T (K_1 + K_2)a = a^T K_1 a + a^T K_1 a \geq 0$

Therefore, the Gram matrix K is also a positive semi-definite, which means $k(x, x')$ is also a valid kernel.

6 One vs all logistic regression (15 points)

Download the data



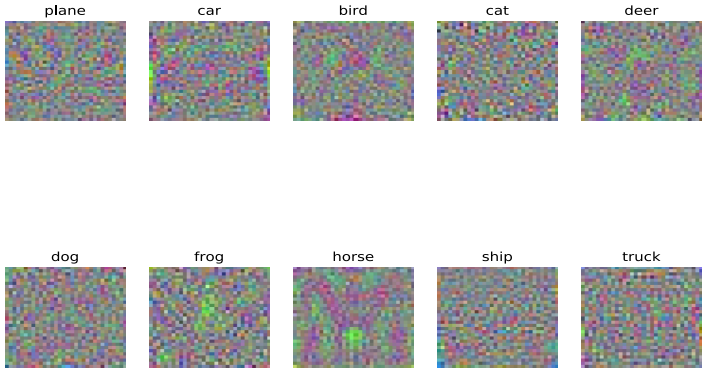

```
Training data shape:  (49000, 3072)
Validation data shape:  (1000, 3072)
Test data shape:  (10000, 3072)
Training data shape with bias term:  (49000, 3073)
Validation data shape with bias term:  (1000, 3073)
Test data shape with bias term:  (10000, 3073)
```

Problem 6.1. Implementing a one-vs-all classifier for the CIFAR-10 dataset
Problem 6.2: Predicting with a one-vs-all classifier (5 points)
Implementing a one_vs_all classifier for CIFAR-10

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]]

Visualizing the learned one-vs-all classifier



Comparing your OVA classifier with sklearn's classifier
Comparing your 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]]

Visualizing the sklearn OVA classifier

Comment on the difference between my ova classifier with sklearn's classifier:
Running my OVA classifier and sklearn's classifier, we get exactly the same test set accuracy: 0.36200. The confusion matrixes of these two classifiers are also same. But when visualizing plot by these two classifiers, the contour of animals' images of my OVA classifier is clearer than images of sklearn's classifier.

7 Softmax regression (45 points)

Problem 7.1: Implementing the loss function for softmax regression (naive version) (5 points)

Implementing softmax regression for the CIFAR-10 dataset

Load the CIFAR-10 dataset

Implementing the loss function for softmax regression (naive version)

loss: (should be close to 2.38): 2.26697618699

We should expect to see a value of about $-\log_e(0.1)$ since the numerator is the exponential of theta in kth category, and the denominator is sum of all K categories, so the ratio will approximately be the one divided by 10, which equal to 0.1.

Problem 7.2: Implementing the gradient of loss function for softmax regression (naive version) (5 points)

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

numerical: 1.219896 analytic: 1.219896, relative error: 9.607892e-09
numerical: 0.412196 analytic: 0.412196, relative error: 7.575155e-08
numerical: 0.121181 analytic: 0.121181, relative error: 1.035536e-07
numerical: -0.544416 analytic: -0.544416, relative error: 1.309612e-07

numerical: 0.526976 analytic: 0.526976, relative error: 1.123345e-07
numerical: -0.417867 analytic: -0.417867, relative error: 8.238003e-08
numerical: 0.912383 analytic: 0.912383, relative error: 3.395058e-08
numerical: 1.382606 analytic: 1.382606, relative error: 3.140028e-09
numerical: -2.399155 analytic: -2.399155, relative error: 3.383841e-08
*numerical: 1.027961 analytic: 1.027961, relative error: 1.496*442e-08
naive loss: 2.266976e+00 computed in 16.693439s

Problem 7.3: Implementing the loss function for softmax regression (vectorized version) (10 points)
Problem 7.4: Implementing the gradient of loss for softmax regression (vectorized version) (5 points)

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

vectorized loss: 2.266976e+00 computed in 0.430081s
Loss difference: 0.000000
Gradient difference: 0.000000

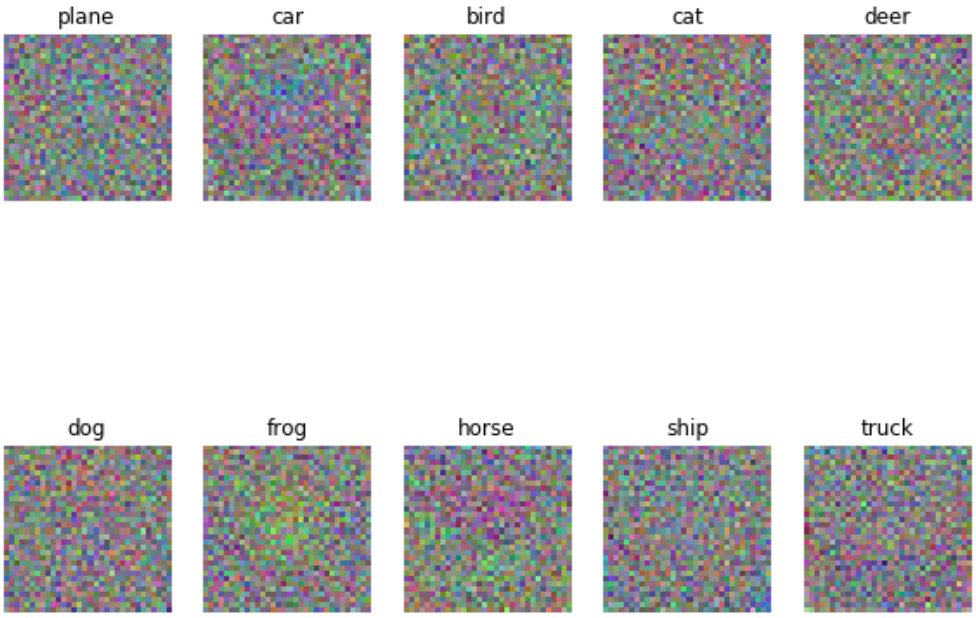
Problem 7.5: Implementing mini-batch gradient descent (5 points)
Implementing mini-batch gradient descent
Problem 7.6: Using a validation set to select regularization lambda and learning rate for gradient descent (5 points)

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

best validation accuracy achieved during cross-validation: 0.418000

Problem 7.7: Training a softmax classifier with the best hyperparameters (5 points)

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



softmax on raw pixels final test set accuracy: 0.396200
[[466 53 47 26 11 27 31 49 199 91]
[58 484 31 30 17 37 54 35 79 175]
[98 61 245 79 94 79 182 72 63 27]
[36 68 89 219 44 182 172 51 62 77]
[56 33 147 44 251 87 211 105 32 34]
[46 57 95 133 64 334 107 64 68 32]
[10 45 64 96 81 78 520 41 23 42]
[55 50 63 46 79 73 65 420 49 100]
[168 75 15 20 5 41 19 19 518 120]
[64 180 16 19 11 26 45 47 87 505]]

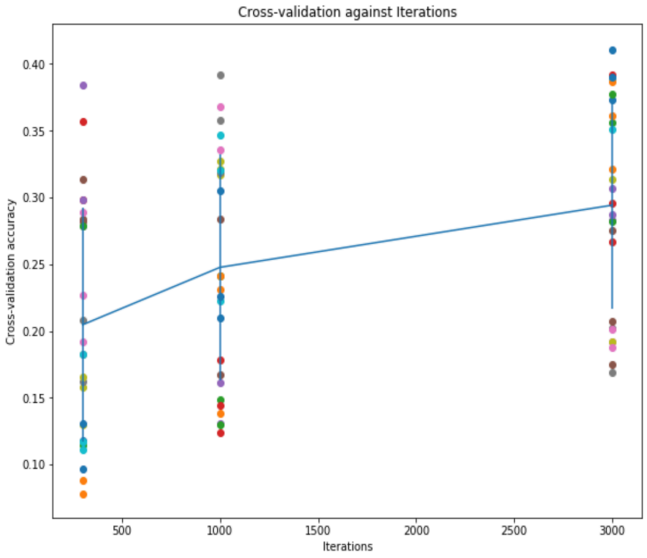
Comment:
Focusing on error made by classifier, we find the confusion categories will happen correspondingly. In another word, one category ‘A’ is easily recognized as another ‘B’ means ‘B’ will also easily be recognized as ‘A’. As we can see in confusion matrix, we can see small animals are easy to be misrecognized, for example, bird, cat, deer, dog and frog are easy to be

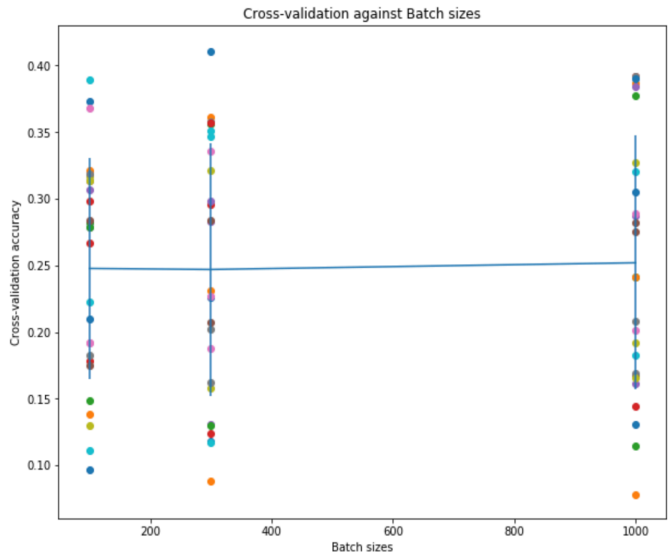
recognized as others. Other sets of categories easy to be mixed, like car and truck, plane and ship, truck and ship, horse and truck, might because they are similar in shape or size.

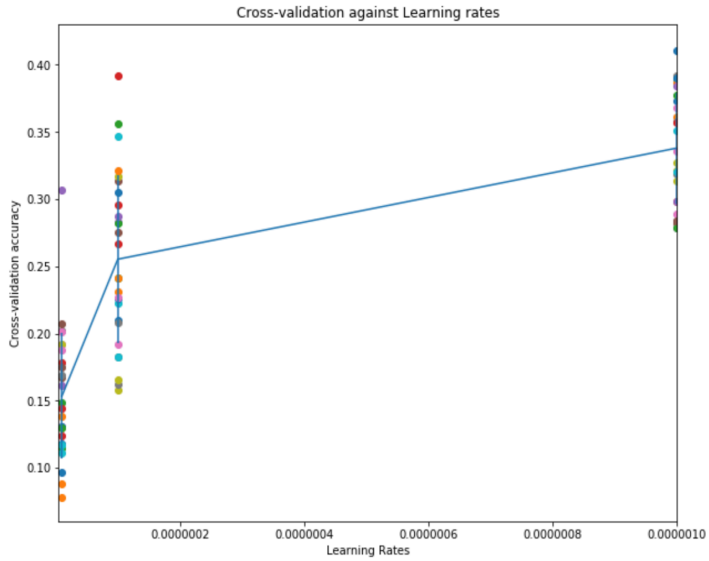
Extra credit: Problem 7.8: Experimenting with other hyper parameters and optimization method (10 points)

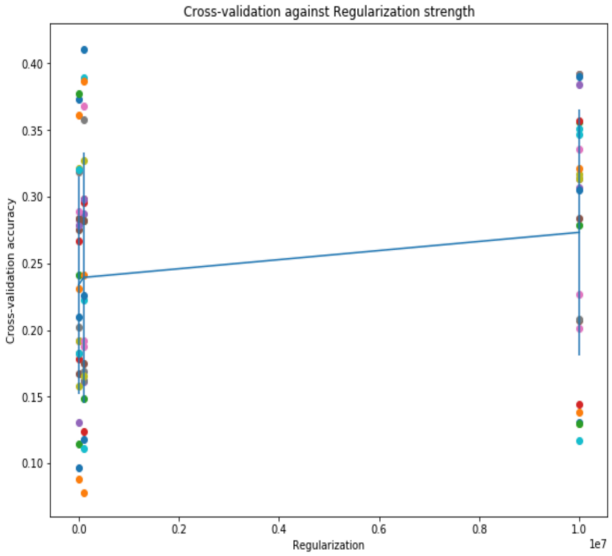
Extra credit: Experimenting with other hyper parameters and optimization method

Produce plots that show the variation of test set accuracy as a function of batch size/number of iterations.









Softmax on raw pixels final test set accuracy: 0.396200.
iter 3000 size 300 lr 5.000000e-06 reg 1.000000e+05 train accuracy: 0.412878 val accuracy: 0.419000
iter 3000 size 300 lr 5.000000e-06 reg 1.000000e+06 train accuracy: 0.398776 val accuracy: 0.400000
iter 3000 size 300 lr 5.000000e-06 reg 1.000000e+07 train accuracy: 0.336469 val accuracy: 0.346000
iter 3000 size 300 lr 5.000000e-06 reg 1.000000e+05 train accuracy: 0.424735 val accuracy: 0.414000
iter 3000 size 300 lr 5.000000e-06 reg 1.000000e+06 train accuracy: 0.375694 val accuracy: 0.372000
iter 3000 size 300 lr 5.000000e-06 reg 1.000000e+07 train accuracy: 0.289204 val accuracy: 0.312000
iter 3000 size 300 lr 5.000000e-06 reg 1.000000e+05 train accuracy: 0.277918 val accuracy: 0.261000

iter 3000 size 300 lr 5.000000e-06 reg 1.000000e+06 train accuracy: 0.225980 val accuracy: 0.225000
iter 3000 size 300 lr 5.000000e-06 reg 1.000000e+07 train accuracy: 0.142653 val accuracy: 0.148000

Best validation accuracy achieved during cross-validation is 0.419000 by iter 3000 size 300 lr 5.000000e-06 reg 1.000000e+07.

Softmax on raw pixels final test set accuracy: 0.393300.

What is the best batch size/number of iterations/learning rate/regularization strength combination for this problem? What is the best test set accuracy that can be achieved by this combination?

Answer:
The best batch size for this question is *300*, best number of iterations is *3000*, the best learning rate is *5.000000e-06* and the best regularization strength is *1.000000e+07.*
Moreover, the best test set accuracy that can be achieved by this combination is: *0.393300.*

Problem 7.9. Comparing OVA binary logistic regression with softmax regression (5 points)

Correct recognized number of two methods										
	plane	car	bird	cat	deer	dog	frog	horse	ship	truck
OVA	465	465	193	161	234	272	457	405	547	421
Softmax	466	484	245	219	251	334	520	420	518	505

The test accuracy by OVA is 0.362000, while test accuracy by Softmax 0.396200
Comment:
Their performance is close but still has 3% difference. The softmax classifier is better. So I will choose softmax classifier method for the CIFAR-10 classification problem.