1 Logistic Regression (40 points)

1.1 (10 points) **Bayes' Rule**

Use Bayes' rile to show that $p(y = 1|\boldsymbol{x})$ takes the form of a logistic function:

$$p(y = 1|\boldsymbol{x}) = \sigma(w^T x + b) = \frac{1}{1 + \exp(-\sum_{i=1}^{D} w_i x_i - b)}$$

Derive expressions for the weight $\boldsymbol{w} = (w_1, \dots, w_D)^T$ and the bias b in terms of the parameters of the class likelihoods and priors (i.e., $\mu_{i0}, \mu_{i1}, \sigma_i \; and \; \alpha$).

Proof:

$$p(y = 1|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|y = 1)p(y = 1)}{p(\boldsymbol{x}|y = 1)p(y = 1) + p(\boldsymbol{x}|y = 0)p(y = 0)}$$

$$= \frac{1}{1 + \frac{p(\boldsymbol{x}|y = 0)p(y = 0)}{p(\boldsymbol{x}|y = 1)p(y = 1)}} = \frac{1}{1 + \exp\left(\ln\left(\frac{p(\boldsymbol{x}|y = 0)p(y = 0)}{p(\boldsymbol{x}|y = 1)p(y = 1)}\right)\right)}$$

$Accroding\ to\ the\ conditional\ independence\ assumption, we\ have$:

$$\frac{1}{1 + \exp\left(\ln(\frac{p(y = 0)}{p(y = 1)}) + \ln\left(\frac{p(\boldsymbol{x}|y = 0)}{p(\boldsymbol{x}|y = 1)}\right)\right)}$$

$$= \frac{1}{1 + \exp\left(\ln(\frac{p(y = 0)}{p(y = 1)}) + \sum_{i=1}^{D} \ln\left(\frac{p(x_i|y = 0)}{p(x_i|y = 1)}\right)\right)}$$

$After\ we\ fit\ the\ models\ to\ Gaussian, \sum_{i=1}^{D} \ln\left(\frac{p(x_i|y = 0)}{p(x_i|y = 1)}\right) =$

$$= \sum_{i=1}^{D} \ln\left(\frac{\frac{\exp(\frac{-(x_i - \mu_{i0})^2}{2\sigma_i^2})}{\sqrt{2\pi\sigma_i^2}}}{\frac{\exp(\frac{-(x_i - \mu_{i1})^2}{2\sigma_i^2})}{\sqrt{2\pi\sigma_i^2}}}\right) = \sum_{i=1}^{D} \ln\left(\exp(\frac{(x_i - \mu_{i1})^2}{2\sigma_i^2} - \frac{(x_i - \mu_{i0})^2}{2\sigma_i^2})\right)$$

$$= \sum_{i=1}^{D} \ln\left(\exp(\frac{(x_i - \mu_{i1})^2 - (x_i - \mu_{i0})^2}{2\sigma_i^2})\right)$$

$$= \sum_{i=1}^{D} \ln\left(\exp(\frac{(\mu_{i0} - \mu_{i1})(2x_i - (\mu_{i0} + \mu_{i1}))}{2\sigma_i^2})\right)$$

$$= \sum_{i=1}^{D} \ln\left(\exp\left(\frac{(\mu_{i0}-\mu_{i1})\big(2x_i-(\mu_{i0}+\mu_{i1})\big)}{2\sigma_i^2}\right)\right) =$$

$$= \sum_{i=1}^{D} \ln\left(\exp\left(\frac{\big(\mu_{i0}^2-\mu_{i1}^2\big)+2x_i(\mu_{i0}-\mu_{i1})}{2\sigma_i^2}\right)\right)$$

$$= \sum_{i=1}^{D} \left(\frac{\mu_{i0}^2-\mu_{i1}^2}{2\sigma_i^2}+\frac{\mu_{i0}-\mu_{i1}}{\sigma_i^2}x_i\right)$$

$$p(y=1|\boldsymbol{x}) = \frac{1}{1+\exp\left(\ln\left(\frac{p(y=0)}{p(y=1)}\right)+\sum_{i=1}^{D}\ln\left(\frac{p(x_i|y=0)}{p(x_i|y=1)}\right)\right)}$$

$$= \frac{1}{1+\exp\left(\ln\left(\frac{1-\alpha}{\alpha}\right)+\sum_{i=1}^{D}\left(\frac{\mu_{i0}^2-\mu_{i1}^2}{2\sigma_i^2}+\frac{\mu_{i0}-\mu_{i1}}{\sigma_i^2}x_i\right)\right)}$$

$$= \frac{1}{1+\exp\left(\sum_{i=1}^{D}\left(\frac{\mu_{i0}-\mu_{i1}}{\sigma_i^2}\right)x_i+\left(\ln\left(\frac{1-\alpha}{\alpha}\right)+\sum_{i=1}^{D}\left(\frac{\mu_{i0}^2-\mu_{i1}^2}{2\sigma_i^2}\right)\right)\right)}$$

This means we can write GBC in the form of logistic regression:

$$p(y=1|\boldsymbol{x}) = \sigma(w^T x+b) = \frac{1}{1+exp\left(-\sum_{i=1}^{D} w_i x_i - b\right)}$$

$$with\ w_i = -\frac{\mu_{i0}-\mu_{i1}}{\sigma_i^2} = \frac{\mu_{i1}-\mu_{i0}}{\sigma_i^2}$$

$$and\ b = -\left(\ln\left(\frac{1-\alpha}{\alpha}\right)-\sum_{i=1}^{D}\left(\frac{\mu_{i0}^2-\mu_{i1}^2}{\sigma_i^2}\right)\right) = \ln\left(\frac{\alpha}{1-\alpha}\right)+\sum_{i=1}^{D}\left(\frac{\mu_{i0}^2-\mu_{i1}^2}{\sigma_i^2}\right).$$

1.2 (15 points) **Maximum Likelihood Estimation**

Now, suppose you are given training set $D = \{(\boldsymbol{x}^{(1)}, y^{(1)}), .., (\boldsymbol{x}^{(n)}, y^{(n)})\}$. Consider a binary logistic regression classifier of the same form as before:

$$p\left(y = 1 \middle| \boldsymbol{x}^{(n)}, \boldsymbol{w}, b\right) = \sigma(w^T x + b) = \frac{1}{1 + \exp(-\sum_{i=1}^{D} w_i x_i - b)}$$

Derive an expression for $E(\boldsymbol{w}, b)$, the negative log-likelihood of $y^{(1)}, \ldots, y^{(n)}$, given $\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(n)}$ and the model parameters, under the i.i.d assumption. Then derive expressions for the derivatives of E with respect to each of the model parameters.

Proof:

$$Accroding\ to\ \text{i. i. d assumption:}$$

$$E(\boldsymbol{w}, b) = -\ln\left(\prod_{i=1}^{N} p\left(y^{(i)} = 1 \middle| x^{(i)}\right)^{y^{(i)}} p\left(y^{(i)} = 0 \middle| x^{(i)}\right)^{1-y^{(i)}}\right)$$

$$= -\sum_{i=1}^{N} t^{(i)} \ln p\left(y^{(i)} = 1 \middle| x^{(i)}\right) - \sum_{i=1}^{N} \left(1 - y^{(i)}\right) \ln p\left(y^{(i)} = 0 \middle| x^{(i)}\right)$$

$$= \sum_{i=1}^{N} y^{(i)} \ln(1 + \exp(-z^{(i)})) + \sum_{i=1}^{N} \left(1 - y^{(i)}\right) \ln(1 + \exp(-z^{(i)}))$$

$$+ \sum_{i=1}^{N} \left(1 - y^{(i)}\right) z^{(i)}, with\ z = \boldsymbol{w}x + b$$

$$= \sum_{i=1}^{N} \ln(1 + \exp(-\boldsymbol{w}x^{(i)} - b)) + \sum_{i=1}^{N} \left(1 - y^{(i)}\right)\left(\boldsymbol{w}x^{(i)} + b\right)$$

$$The\ derivatives\ of\ E\ with\ respect\ to\ each\ of\ the\ model\ parameters:$$

$$\frac{\sigma E(\boldsymbol{w}, b)}{\sigma w_j} = \sum_{i=1}^{N} \left(1 - y^{(i)}\right) x_j^{(i)} - \sum_{i=1}^{N} x_j^{(i)} \frac{\exp(-\boldsymbol{w}x^{(i)} - b)}{1 + \exp(-\boldsymbol{w}x^{(i)} - b)}$$

$$\frac{\sigma E(\boldsymbol{w}, b)}{\sigma b} = \sum_{i=1}^{N} \left(1 - y^{(i)}\right) - \sum_{i=1}^{N} \frac{\exp(-\boldsymbol{w}x^{(i)} - b)}{1 + \exp(-\boldsymbol{w}x^{(i)} - b)}$$

1.3 (15 points) **L2 Regularization**

Now assume that a Gaussian prior is placed on each element of **w**, and b such that $p(w_i) = N\left(w_i \middle| 0, {}^{1}/_{\lambda}\right) and\ p(b) = N\left(b \middle| 0, {}^{1}/_{\lambda}\right)$. Derive an expression that is proportional to $p(\boldsymbol{w}, b|D)$, the posterior distribution of **w** and b, based on this prior and the likelihood defined above. The expression you derive must contain all terms that depend on **w** and b.

Define $L\ (\boldsymbol{w}, b)$ to be the negative logarithm of the expression you derive. Show that $L\ (\boldsymbol{w}, b)$ takes the following form:

$$L(\boldsymbol{w}, b) = E(\boldsymbol{w}, b) + \frac{\lambda}{2}\sum_{i=1}^{D} w_i^2 + C(\lambda)$$

where $C(\lambda)$ is a term that depends on λ but not on either **w** or b. What are the derivatives of L with respect to each of the model parameters?

Proof:

$$p(\boldsymbol{w}, b|D) \propto p(D|\boldsymbol{w}, b)p(\boldsymbol{w}, b)$$

$$Accroding\ to\ the\ conditional\ independence\ assumption,$$

$$p(\boldsymbol{w}, b) = \prod_{i=1}^{J} N\left(w_i \middle| 0, {}^{1}/_{\lambda}\right) N\left(b \middle| 0, {}^{1}/_{\lambda}\right)$$

$$p(D|\boldsymbol{w}, b) = -\sum_{i=1}^{N} \ln(1 + \exp(-\boldsymbol{w}x^{(i)} - b)) - \sum_{i=1}^{N} (1 - y^{(i)})(\boldsymbol{w}x^{(i)} + b)$$

$$p(\boldsymbol{w}, b|D)$$

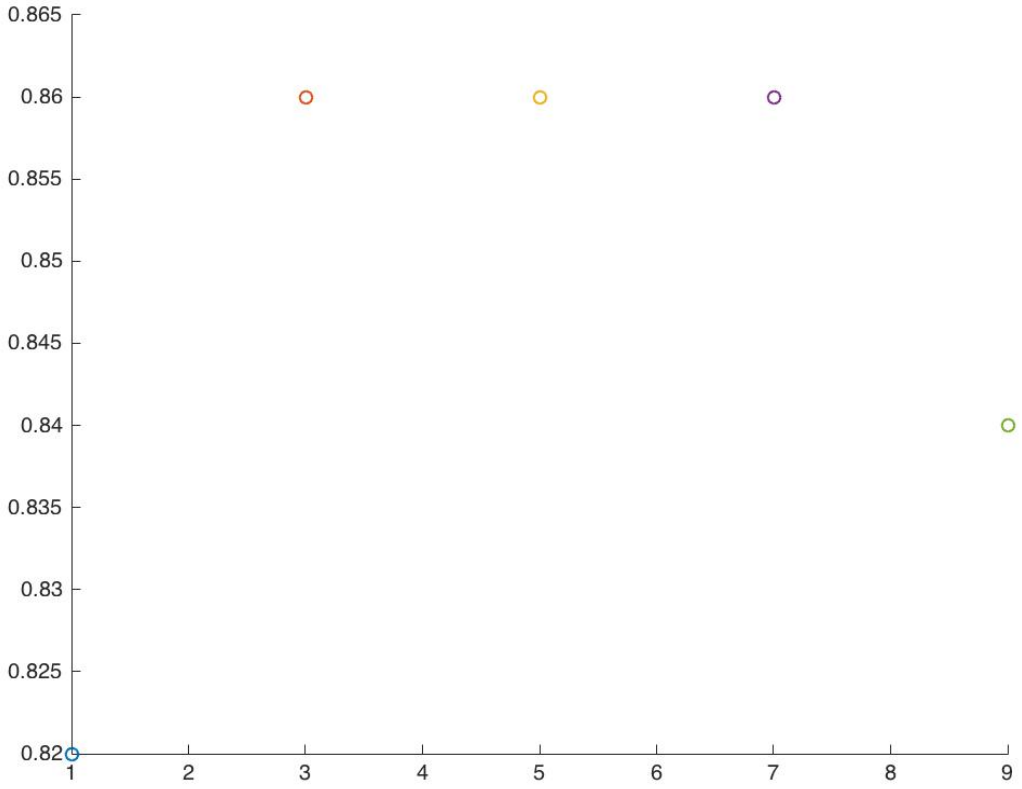
$$\propto \left(\prod_{i=1}^{N} \frac{1}{1 + \exp(-\boldsymbol{w}x^{(i)} - b)}^{y^{(i)}} \frac{\exp(-\boldsymbol{w}x^{(i)} - b)}{1 + \exp(-\boldsymbol{w}x^{(i)} - b)}^{1-y^{(i)}}\right) \prod_{i=1}^{D} N\left(w_i \middle| 0, {}^{1}/_{\lambda}\right)$$

$$L(\boldsymbol{w}, b) = -\ln(expression) = E(\boldsymbol{w}, b) - \sum_{i=1}^{D} \frac{-(w_i)^2}{2\left({}^{1}/_{\lambda}\right)} + (D)\ln\sqrt{2\pi\left({}^{1}/_{\lambda}\right)}$$

$$= E(\boldsymbol{w}, b) + \frac{\lambda}{2}\sum_{i=1}^{D} w_i^2 + +C(\lambda)$$

2 Digit Classification (60 points)

2.1(10 points) k-**Nearest Neighbors (***code in knn_script.m***)**



k	Classification rate of validation data
1	82%
3	86%
5	86%
7	86%
9	84%

It shows that it has best performance when k = 3,5,7.
Also to reduce the influence of the noise we should use larger k which is 7 in this case.
The classification rate of test data

k	Classification rate of test data
5	94%
7	94%
9	88%

The test performance corresponds to the validation performance, because k-NN is a non-parametric model, the only hyper parameter is k. Therefore, the validation and testing set should have similar response for finding k.

2.2(10 points) **Logistic regression (***code in logistic.m logistic_template.m evaluate.m logistic_predict.m***)**

	`Training_data`	`Training_data_small`
`Learning_rate`	`0.1`	`0.1`
`Num_iteration`	`80`	`240`
`Initial_weights`	`All zeros`	`All zeros`

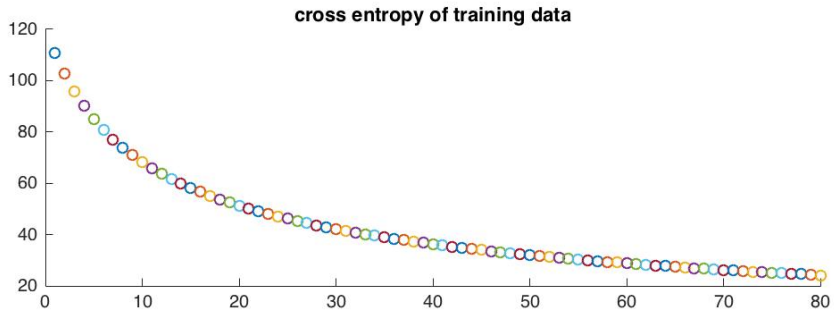
Result of training the model trained by train_data:

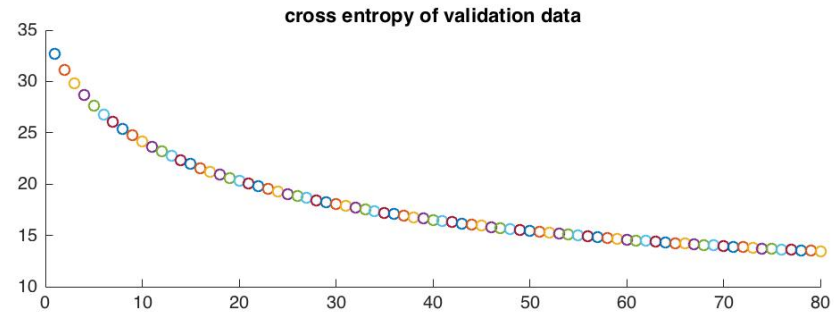
	Cross Entropy	Classification Error
Training	**24.147825**	**0%**
Validation	**13.438001**	**10%**
Test	**12.2186**	**8%**

Result of training the model trained by train_data_small:

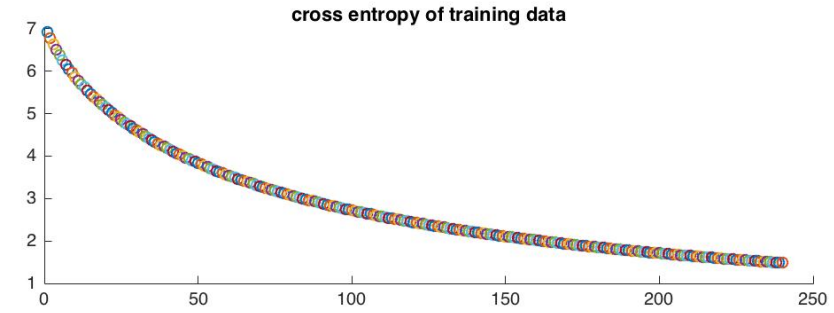
	Cross Entropy	Classification Error
Training	**1.483817**	**0%**
Validation	**33.535344**	**38%**
Test	**27.0655**	**24%**

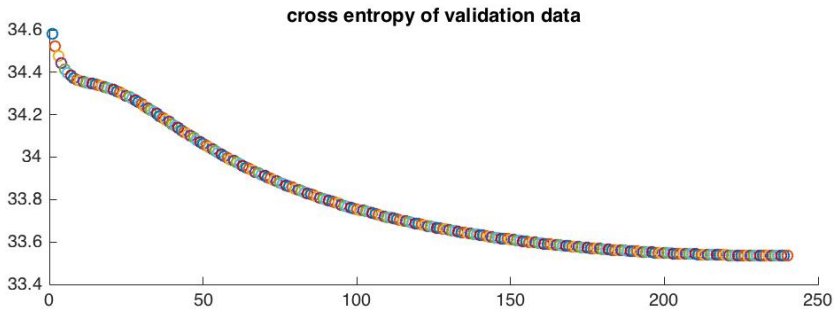
Cross entropy of model trained by train_data:





Cross entropy of model trained by train_data_small:

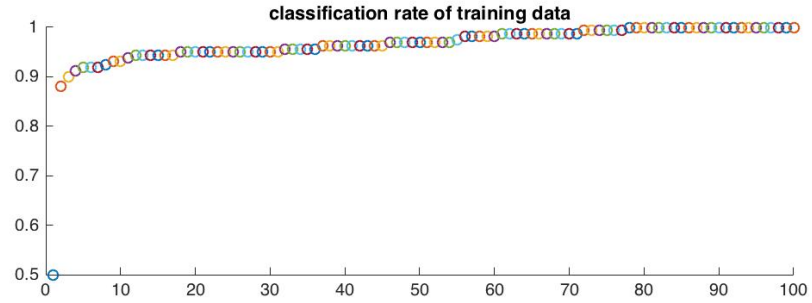


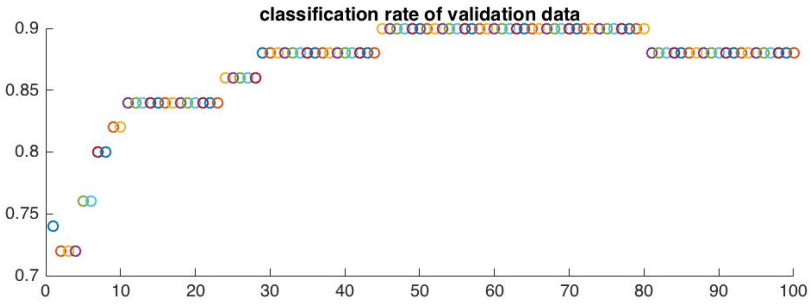


Since both validation and training cross entropy keep reducing. 0.1 is a pretty good learning rate as far as I concerned. To avoid over fitting, initialization weights start from 0 should be a good choice, and it actually works pretty good though. To

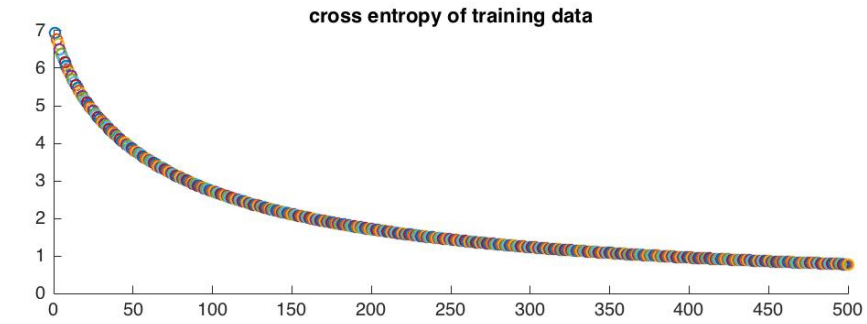
figure out the number of iterations, we need to find a point where cross entropy or the misclassification error of validation set goes up and stop at this point.

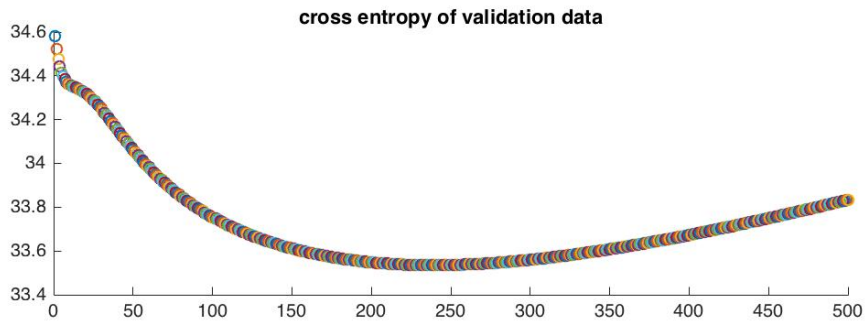
Classification rate of model trained by train_data:





Cross entropy of model trained by train_data_small:

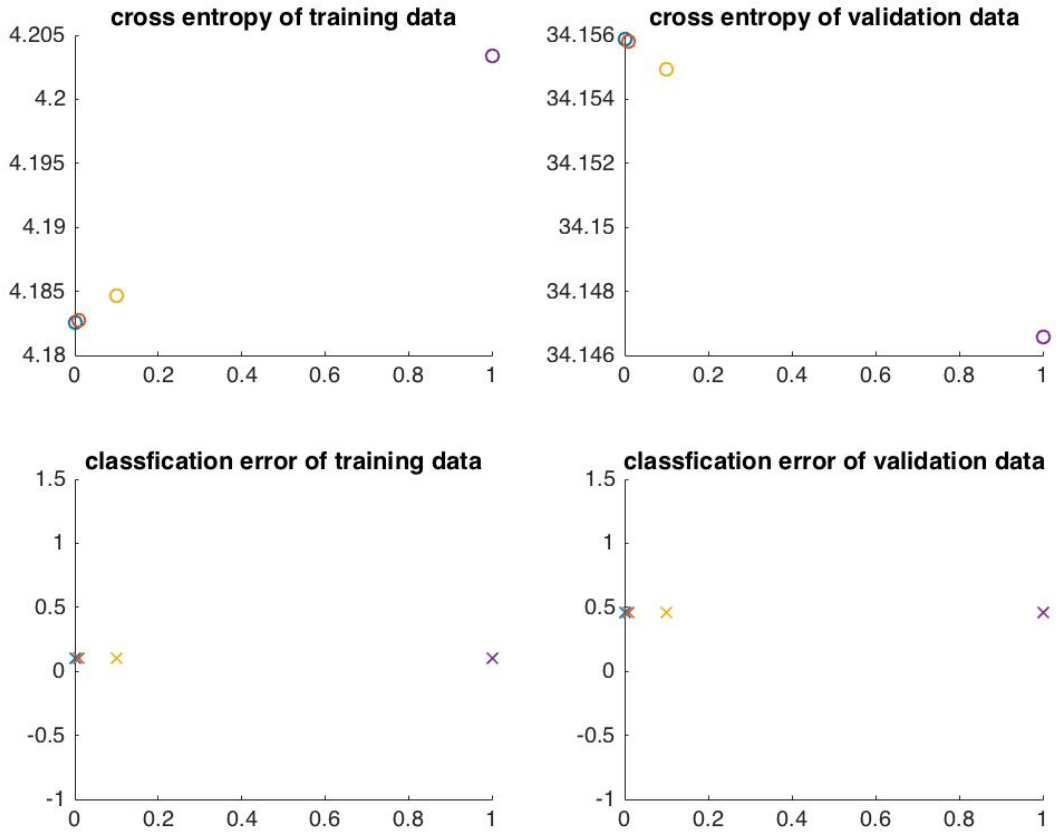




The classification rate of the validation set decrease when the number of iterations reaches 80 (train_data), and the cross entropy of the validation set increase when the number of iterations reaches 240(train_data_small).

2.3(10 points) **Penalized logistic regression (***code in logistic.m logistic_template.m evaluate.m logistic_predict.m***)**

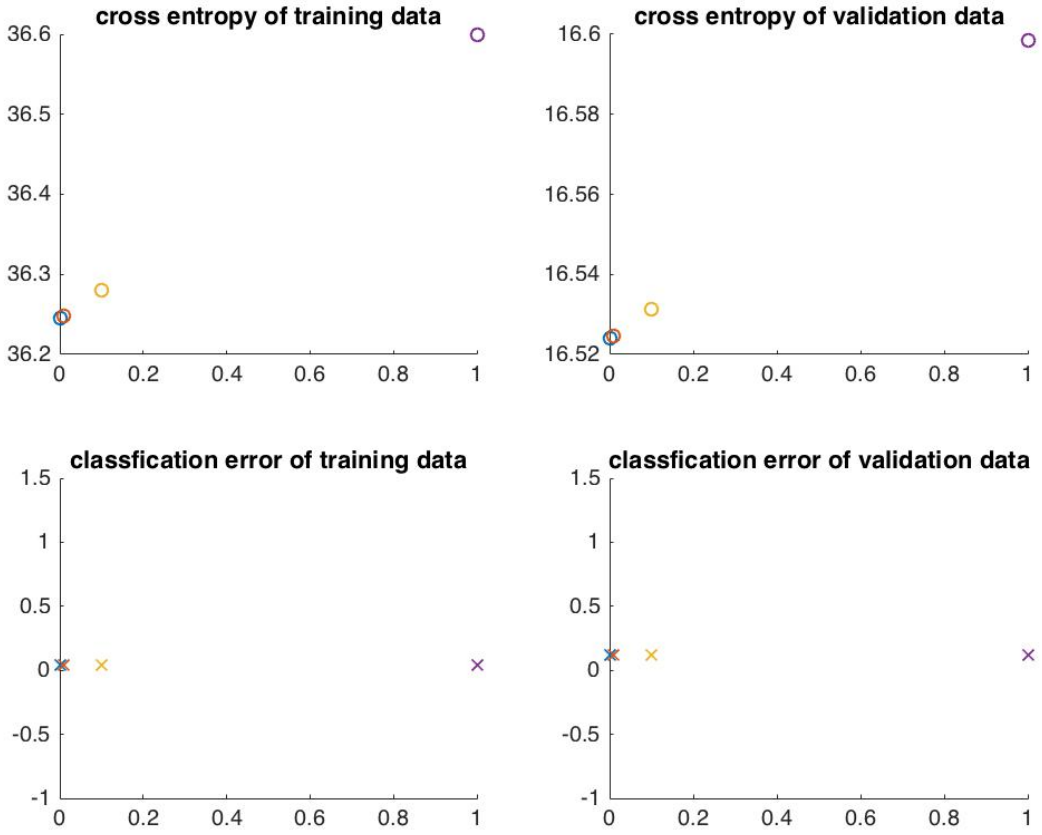
Cross entropy and classification error of model trained by train_data_small:



In this case, we see that the cross entropy goes down in validation set and goes up in training set. It should have some thing to do with the size of our training set which is too small in this case means that it is really easy to over fit the training data. We conclude that lambda equals to 1 is the best choice in this case.

Final test error:
Cross entropy: 11.6114
Classification error: 8%

Cross entropy and classification error of model trained by train_data:



Here, we see that both cross entropy in validation set and training set goes up. Training the model using a large size training set could reduce the chance to over fit. In conclusion, pick a smaller lambda is a good choice which is 0.001 in this case.

Final test error:
Cross entropy: 27.0335
Classification error: 24%

In both cases, the classification error is close to each other. I think the reason for this is because the penalization on parameter doesn't have a huge influence on the final predicted possibility. And the classification rate is more robust to small change of predicted possibility.

Result of training the model trained by train_data:

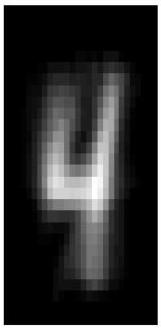
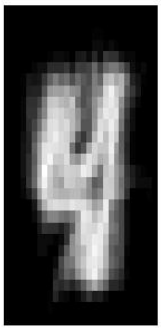
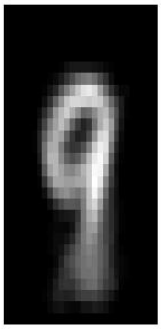
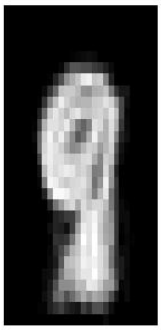
	Cross Entropy	Classification Error
Penalized	**11.6114**	**8%**
Not penalized	**12.2186**	**8%**

Result of training the model trained by train_data:

	Cross Entropy	Classification Error
Penalized	**27.0335**	**8%**
Not penalized	**27.0655**	**8%**

After doing comparison, we found that the penalized logistic regression tends to perform better for validation set whereas logistic regression has better performance for training set. The reason for this is because when penalizing the regression, we have two targets to achieve (1) make training error small (2) make parameter small. This is a tradeoff between the complexity and the generalization, which means we will have a worse performance for the training set and a better performance for the validation set.

2.4(15 points) **Naive Bayes (***code in run_nb.m***)**
Training accuracy: 86.25%
Test accuracy: 80%

Mean: Variance:

Comment:
Means visualized like a standard 4 and 9.
Two classes don't share the covariance, which means the decision boundary should be nonlinear. Conflicts with the Naïve Bayes assumption, which may result in poor result.

2.5(15 points) **Compare k-NN, Logistic regression, and Naïve Bayes**

	k-NN	**Logistic regression**	**Penalized logistic regression**	**Naïve Bayes**
Classification rate	94%	92%	92%	80%

For this problem, *k*-NN is the most successful in predicting the classes of the test data. This may have to do with the fact that it makes the fewest assumptions. Naïve Bayes classification method is based on the assumption that the conditional distribution fits Gaussian, which may be inappropriate for this problem. And also data may not be i.i.d which can make this classifier total wrong. The logistic regression is also pretty good because it makes few assumptions to the data, the reason for not as good as k-NN could due to the nonlinearity in true decision boundary. The penalized logistic regression converges slower compared to logistic regression but are more generalized.

As for the performance, k-NN is very fast because it doesn't have a training process but takes huge amount of memory to store the data. Naïve Bayes is fast in this case because the likelihood function of Gaussian has a closed form solution thus there is no need to optimize (i.e. training). Logistic regression is much slower compared to the other two due to gradient descent (i.e. optimization).