Assignment 1

Juliana De La Vega Fernandez - MScAC

October 14, 2016

1 LOGISTIC REGRESSION

1.1 DEFINE A LOSS FUNCTION TO BE THE NEGATIVE LOG POSTERIOR OVER THE WEIGHTS. WRITE THE LOSS FUNCTION AS SIMPLIFIED AS POSSIBLE. ASSUME THE DATA IS I.I.D.

Step 1. Calculate the Loss Function without regularization.
The examples in the training set are described by:
$D = \{(x^{(1)}, \quad t^{(1)}), \quad ... \quad , (x^{(N)}, \quad t^{(N)})\}$

Where the input is a D-dimensional vector:
$x^{(i)} = \left(x_1^{(i)}, \quad x_2^{(i)}, \quad ... \quad , x_D^{(i)}\right)^T$

And the targets are binary:
$t^{(i)} \epsilon \quad \{0,1\}$

The model takes the form of a logistic function, described as:
$p\left(t = 1 | x^{(i)}, \quad w, \quad w_0\right) \quad = \quad \sigma\left(w^T x^{(i)} + w_0\right) = \frac{1}{1+exp\left(-\sum_{d=1}^{D} w_d^T x_d^{(i)} - w_0\right)}$

The likelihood function is defined as:
$p\left(t^{(1)}, t^{(2)}, ..., t^{(N)} | x^{(1)}, x^{(2)}, ..., x^{(N)}, w, w_0\right)$

Assuming that the training examples are I.I.D., the likelihood function is expressed as:
$L(w) = \prod_{i=1}^{N} p\big(C = 1 | x^{(i)}\big)^{t^{(i)}} \cdot \big(1 - p\big(C = 1 | x^{(i)}\big)\big)^{1-t^{(i)}}$

The loss function is defined as the negative log posterior of the likelihood:
$l_{log}(w) = -\sum_{i=1}^{N} t^{(i)} \cdot \log\left(p\left(C=1|x^{(i)}\right)\right) - \sum_{i=1}^{N}\left(1-t^{(i)}\right)\cdot\log\left(1-p\left(C=1|x^{(i)}\right)\right)$

And replacing each probability for class 1 and class 2:
$l_{log}(w) = -\sum_{i=1}^{N} t^{(i)} \cdot \log\left(\frac{1}{1+exp\left(-\sum_{d=1}^{D} w_d^T x_d^{(i)} - w_0\right)}\right) - \sum_{i=1}^{N}\left(1-t^{(i)}\right)\cdot\log\left(1-\frac{1}{1+exp\left(-\sum_{d=1}^{D} w_d^T x_d^{(i)} - w_0\right)}\right)$

$l_{log}(w) = -\sum_{i=1}^{N} t^{(i)} \cdot \left(-\quad \log\left(1+exp\left(-\sum_{d=1}^{D} w_d^T x_d^{(i)} - w_0\right)\right)\right) - \sum_{i=1}^{N}\left(1-t^{(i)}\right)\cdot\log\left(\frac{exp\left(-\sum_{d=1}^{D} w_d^T x_d^{(i)} - w_0\right)}{1+exp\left(-\sum_{d=1}^{D} w_d^T x_d^{(i)} - w_0\right)}\right)$

$l_{log}(w) = -\sum_{i=1}^{N} t^{(i)} \cdot \left(-\quad \log\left(1+exp\left(-\sum_{d=1}^{D} w_d^T x_d^{(i)} - w_0\right)\right)\right)$
$-\sum_{i=1}^{N}\left(1-t^{(i)}\right)\cdot\left(\log\left(exp\left(-\sum_{d=1}^{D} w_d^T x_d^{(i)} - w_0\right)\right) - \log\left(1+exp\left(-\sum_{d=1}^{D} w_d^T x_d^{(i)} - w_0\right)\right)\right)$

$l_{log}(w) = -\sum_{i=1}^{N} t^{(i)} \cdot \left(-\quad \log\left(1+exp\left(-\sum_{d=1}^{D} w_d^T x_d^{(i)} - w_0\right)\right)\right) - \sum_{i=1}^{N}\left(1-t^{(i)}\right)\cdot\left(-\sum_{d=1}^{D} w_d^T x_d^{(i)} - w_0\right)$
$-\quad \sum_{i=1}^{N}\left(-t^{(i)}\right)\cdot\left(-\log\left(1+exp\left(-\sum_{d=1}^{D} w_d^T x_d^{(i)} - w_0\right)\right)\right) \quad - \quad \sum_{i=1}^{N}\left(-\log\left(1+exp\left(-\sum_{d=1}^{D} w_d^T x_d^{(i)} - w_0\right)\right)\right)$

$l_{log}(w) = -\sum_{i=1}^{N}\left(1-t^{(i)}\right)\cdot\left(-\sum_{d=1}^{D} w_d^T x_d^{(i)} - w_0\right) \quad - \quad \sum_{i=1}^{N}\left(-\log\left(1+exp\left(-\sum_{d=1}^{D} w_d^T x_d^{(i)} - w_0\right)\right)\right)$

$l_{log}(w) = \sum_{i=1}^{N}\left(1-t^{(i)}\right)\cdot\left(\sum_{d=1}^{D} w_d^T x_d^{(i)} - w_0\right) \quad + \quad \sum_{i=1}^{N}\left(\log\left(1+exp\left(-\sum_{d=1}^{D} w_d^T x_d^{(i)} - w_0\right)\right)\right)$

$l_{log}(w) = \sum_{i=1}^{N}\left(1-t^{(i)}\right)\cdot\left(\sum_{d=1}^{D} w_d^T x_d^{(i)} - w_0\right) \quad + \quad \sum_{i=1}^{N}\left(\log\left(1+exp\left(-\sum_{d=1}^{D} w_d^T x_d^{(i)} - w_0\right)\right)\right)$

Step 2. Calculate the Loss Function with regularization.
A Gaussian prior is placed on the weights such that:
$p(w) = N\left(w|0,\alpha^{-1}I\right) = \left(\frac{\alpha}{2\pi}\right)^{\frac{D}{2}} exp\left(-\frac{\alpha}{2}w^T w\right)$

This term is added to the log posterior calculated in Step 1, obtaining the loss function:

$l_{log}(w) = \sum_{i=1}^{N}\left(1-t^{(i)}\right)\cdot\left(\sum_{d=1}^{D} w_d^T x_d^{(i)} - w_0\right) \quad + \quad \sum_{i=1}^{N}\left(\log\left(1+exp\left(-\sum_{d=1}^{D} w_d^T x_d^{(i)} - w_0\right)\right)\right)$
$+\left(\frac{\alpha}{2\pi}\right)^{\frac{D}{2}} exp\left(-\frac{\alpha}{2}w^T w\right)$

1.2 Derive what the gradient of the loss is with respecto to w_j and w_0.

Step 3. Derive the loss without regularization with respect to w_j.

$\frac{\partial loss}{\partial w_j} = \sum_{i}^{N} \quad = \quad {}_1\left(1-t^{(i)}\right)\cdot x_j^{(i)} \quad - \quad \sum_{i}^{N} \quad = \quad {}_1\left(\frac{1}{1+exp\left(-w_j x_j^{(i)} + w_0\right)}\cdot exp\left(-w_j x_j^{(i)} + w_0\right)\cdot x_j^{(i)}\right)$

$\frac{\partial loss}{\partial w_j} = \sum_{i}^{N} \quad = \quad {}_1\, x_j^{(i)}\cdot\left(1-t^{(i)}\right) - \quad \left(\frac{exp\left(-w_j x_j^{(i)} + w_0\right)}{1+exp\left(-w_j x_j^{(i)} + w_0\right)}\right)$

$$\frac{\partial loss}{\partial w_j} = \sum_i^N \; = \; {}_1 x_j^{(i)} \cdot \left(1 - t^{(i)}\right) - \quad p\left(C = 0 | x^{(i)}, w_j, w_o\right)$$

Step 4. Derive the loss without regularization with respect to w_0.

$$\frac{\partial loss}{\partial w_0} = \sum_i^N \; = \; {}_1 \left(1 - t^{(i)}\right) \cdot (-1) \quad - \quad \sum_i^N \; = \; {}_1 \left(\frac{1}{1 + exp\left(-w_j x_j^{(i)} + w_0\right)} \cdot exp\left(-w_j x_j^{(i)} + w_0\right) \cdot (-1) \right)$$

$$\frac{\partial loss}{\partial w_0} = -\sum_i^N \; = \; {}_1 \left(1 - t^{(i)}\right) \quad + \quad \sum_i^N \; = \; {}_1 \left(\frac{exp\left(-w_j x_j^{(i)} + w_0\right)}{1 + exp\left(-w_j x_j^{(i)} + w_0\right)} \right)$$

Step 5. Include the regularization. Derive $p(w)$ with respect to w_j and add it to the partial derivative of the loss with respect to w_j.

$$\frac{\partial loss}{\partial w_j} \quad = \quad -\alpha w_j + \sum_{i=1}^N x_j^{(i)} \cdot \left(1 - t^{(i)}\right) - \left(\frac{exp\left(-w_j x_j^{(i)} + w_0\right)}{1 + exp\left(-w_j x_j^{(i)} + w_0\right)} \right)$$

$$\frac{\partial loss}{\partial w_j} \quad = \quad -\alpha w_j + \sum_{i=1}^N x_j^{(i)} \cdot \left(1 - t^{(i)}\right) - \quad p\left(C = 0 | x_j^{(i)}, w_j, w_o\right)$$

Step 6. Include the regularization. Derive $p(w)$ with respect to w_0 and add it to the partial derivative of the loss with respect to w_0.

$$\frac{\partial loss}{\partial w_0} = \sum_{i=1}^N \left(1 - t^{(i)}\right) \cdot (-1) - \sum_{i=1}^N \left(\frac{1}{1 + exp\left(-w_j x_j^{(i)} + w_0\right)} \cdot exp\left(-w_j x_j^{(i)} + w_0\right) \cdot (-1) \right)$$

$$\frac{\partial loss}{\partial w_0} = -\sum_{i=1}^N \left(1 - t^{(i)}\right) + \sum_{i=1}^N \left(\frac{exp\left(-w_j x_j^{(i)} + w_0\right)}{1 + exp\left(-w_j x_j^{(i)} + w_0\right)} \right)$$

1.3 Write a pseudocode for gradient descent using those gradients.

for i=1 to N **do**

Update:

$$\mathbf{w} \leftarrow \mathbf{w} + \lambda \alpha \mathbf{w} - \lambda x^{(i)} \cdot \left(1 - t^{(i)}\right) + \left(\lambda \frac{exp(-\mathbf{w} x^{(i)} + w_0)}{1 + exp(-\mathbf{w} x^{(i)} + w_0)} \right)$$

end for

2 Decision Trees

2.1 Construct a decision tree for the binary classification of customers of the resaurant "Mama's Pasta" into 'Satisfied' or 'Unsatisfied'

Step 1. Calculate the Information Gain for each criterion in order to select an attribute on which to perform the first split.

In this case, IG_{OC_1} stands for the information gain for the decision criteria 'overcooked' at the first split. Similarly, IG_{WT_1} stands for the information gain for the decision criteria 'waiting time' at the first split. Lastly, IG_{R_1} stands for the information gain for the decision criteria 'rude waiter' at the first split.

$$IG_{OC\_1} = 1 - \left[\tfrac{3}{5}H\left(Y|yes\right) + \tfrac{2}{5}H\left(Y|no\right)\right]$$

$$IG_{OC\_1} = 1 - \left[\tfrac{3}{5}\cdot\left(-\tfrac{1}{3}\log_2\tfrac{1}{3} - \tfrac{2}{3}\log_2\tfrac{2}{3}\right) + \tfrac{2}{5}\cdot\left(-\tfrac{2}{2}\log_2\tfrac{2}{2} - \tfrac{0}{2}\log_2\tfrac{0}{2}\right)\right]$$

$$IG_{OC\_1} = 1 - \left[\tfrac{3}{5}\cdot\left(-\tfrac{1}{3}\log_2\tfrac{1}{3} - \tfrac{2}{3}\log_2\tfrac{2}{3}\right)\right]$$

$$IG_{OC\_1} \approx 0.449022$$

$$IG_{WT\_1} = 1 - \left[\tfrac{3}{5}H\left(Y|long\right) + \tfrac{2}{5}H\left(Y|short\right)\right]$$

$$IG_{WT\_1} = 1 - \left[\tfrac{3}{5}\cdot\left(-\tfrac{2}{3}\log_2\tfrac{2}{3} - \tfrac{1}{3}\log_2\tfrac{1}{3}\right) + \tfrac{2}{5}\cdot\left(-\tfrac{1}{2}\log_2\tfrac{1}{2} - \tfrac{1}{2}\log_2\tfrac{1}{2}\right)\right]$$

$$IG_{WT\_1} = 1 - \left[\tfrac{3}{5}\cdot\left(-\tfrac{2}{3}\log_2\tfrac{2}{3} - \tfrac{1}{3}\log_2\tfrac{1}{3}\right) + \tfrac{2}{5}\right]$$

$$IG_{WT\_1} \approx 0.0490225$$

$$IG_{R\_1} = 1 - \left[\tfrac{4}{5}H\left(Y|yes\right) + \tfrac{1}{5}H\left(Y|no\right)\right]$$

$$IG_{R\_1} = 1 - \left[\tfrac{4}{5}\cdot\left(-\tfrac{2}{4}\log_2\tfrac{2}{4} - \tfrac{2}{4}\log_2\tfrac{2}{4}\right) + \tfrac{1}{5}\cdot\left(-\tfrac{1}{1}\log_2\tfrac{1}{1} - 0\right)\right]$$

$$IG_{R\_1} = 1 - \left[\tfrac{4}{5}\cdot\left(-\tfrac{2}{4}\log_2\tfrac{2}{4} - \tfrac{2}{4}\log_2\tfrac{2}{4}\right)\right]$$

$$IG_{R\_1} = 0.2$$

Obtaining for the first split: $IG_{OC\_1} > IG_{R\_1} > IG_{WT\_1}$. This would indicate that the first decision criteria is 'overcooked'. It is certain that if the binary answer to 'overcooked pasta' is 'no', the customer will be satisfied.

Step 2. Calculate the information gain for the remaining decision criteria.

In this case, IG_{WT_2} stands for the information gain for the decision criteria 'waiting time' at the second split. Lastly, IG_{R_2} stands for the information gain for the decision criteria 'rude waiter' at the second split.

$$IG_{WT\_2} = 1 - \left[\tfrac{2}{3}H\left(Y|long\right) + \tfrac{1}{3}H(Y|short)\right]$$

$$IG_{WT\_2} = 1 - \left[\tfrac{2}{3}\cdot\left(-\tfrac{1}{2}\log_2\tfrac{1}{2} - \tfrac{1}{2}\log_2\tfrac{1}{2}\right) + \tfrac{1}{3}\cdot(0)\right]$$

$$IG_{WT\_2} = 1 - \left[\tfrac{2}{3}\cdot\left(-\tfrac{1}{2}\log_2\tfrac{1}{2} - \tfrac{1}{2}\log_2\tfrac{1}{2}\right)\right]$$

$$IG_{WT\_2} \approx 0.333333$$

$$IG_{R\_2} = 1 - \left[\tfrac{2}{3}H\left(Y|yes\right) + \tfrac{1}{3}H(Y|no)\right]$$
Âă

$$IG_{R\_2} = 1 - \left[\tfrac{2}{3}\cdot\left(0 - \tfrac{2}{2}\log_2\tfrac{2}{2}\right) + \tfrac{1}{3}\cdot\left(-\tfrac{1}{1}\log_2\tfrac{1}{1} - 0\right)\right]$$

$$IG_{R\_2} = 1 - [0]$$
Âă

$IG_{R\_2} = 1$
Âă

For the second split, $IG_{R\_2} > IG_{WT\_2}$. This means that in the case that the decision criteria 'overcooked pasta' is 'yes', the next decisive criterion will be 'rude waiter'. If the binary response corresponds to 'no', then the classification of the customer will be 'satisfied', otherwise the customer will be 'unsatisfied'. This leaves out the decision criteria 'waiting time', it is not necessary for the classification of the customer into the 'satisfied' or 'unsatisfied' categories.

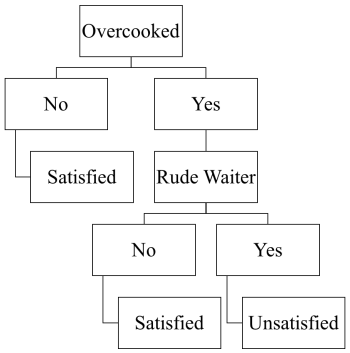


Figure 2.1: Decision tree for "Mama's Pasta" customer satisfaction classification.

2.2 USING THE DECISION TREE, PREDICT WHETHER EACH OF THE TEST USERS WILL BE SATISFIED OR NOT AFTER THEIR VISIT TO "MAMA'S PASTA".

Table 2.1: Decision tree predictions

Person ID	**Overcooked Pasta**	**Waiting time**	**Rude waiter**	**Satisfied?**
6	No	Short	No	Yes
7	Yes	Long	Yes	No
8	Yes	Short	No	Yes

For person 6, the fact that the pasta was not overcooked, immediately suggests that he is satisfied according to the decision tree. For person 7, the pasta was overcooked, this means that the decision criteria 'rude waiter' must be checked. As the waiter was rude, and the waiting time does not seem relevant in this decision tree, the predicted outcome is that person 7 will be unsatisfied. Lastly, for person 8, the pasta was overcooked, and the waiter was not rude. This would suggest that this person is most likely to be satisfied after leaving the restaurant.

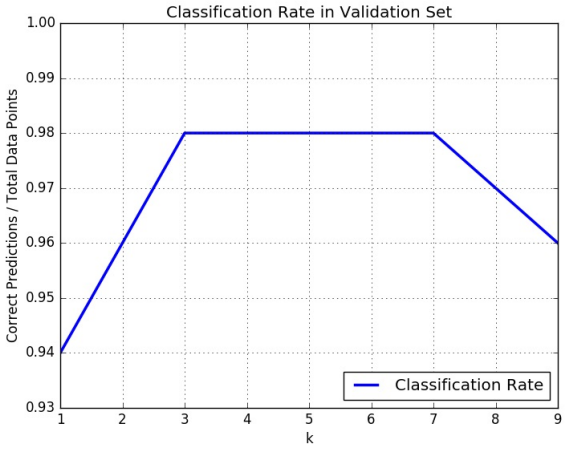


Figure 3.1: Classification rate of KNN on validation set vs k neighbors.

3 Logistic Regression vs. KNN

3.1 k-Nearest Neighbors

3.1.1 Comment on the performance of the classifier and argue which value of k you would choose. What is the classification rate on test set of your chosen value k^*? Compute the rate for k^*+2 and k^*-2 on test set.

The implementation of KNN in file `A1_KNN_performance.py` uses the function `run_knn` in order to calculate the classification rate of each image as a 2 or an 8. In order to generate the plot, `run_knn` was looped changing the value of k. The classification rate was calculated by dividing the total number of correct classifications over the complete set.

Figure 3.1 shows the result of `A1_KNN_performance.py` over the validation set. In general, the results produced have a classification minima of 94%, and a maxima of 98%. There are three values of k which generate the maxima, [3, 5, 7]. It is possible that k equal to [1, 9] will not perform adequately. For $k = 1$, the data point that is being evaluated will only choose the label of the closest training example. This will ignore the general distribution of the data, leading to a classification that is overattentive to detail.

On the other hand, $k = 9$ will not be a good choice in this training set because it is choosing the nine closest training examples and averaging over them in order to set the label for the data point in question. If the data set in training was larger, it is possible that $k = 9$ would be a good choice. However, the data set that is being used is small. Therefore, choosing the nearest nine training example may lead to a confusing label.

The highest performing k are [3, 5, 7]. I will choose $k = 5$ to avoid having the problem of picking too many training examples and have a relaxed classifier, or too few examples creating a

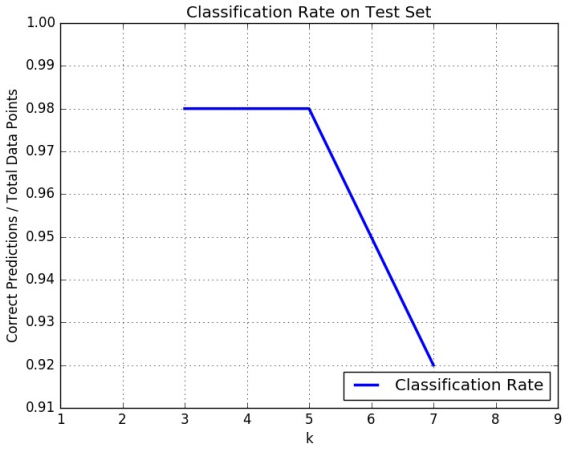


Figure 3.2: Classification rate of KNN on test set vs k neighbors.

short-sighted classifier.

3.1.2 Does the performance for these values of k correspond to the validation performance?

Figure 3.2 shows the result obtained when evaluating $k = [3,5,9]$ on the test set. For $k = 35$ the result for the classification rate is 98%, corresponding to the same value obtained during the validation stage. However, for $k = 7$ the classification rate drops to 92%. This might be an indicator that the data on the test set is a little differently distributed from the data in the training set. It can also mean that for this data set, choosing seven nearest neighbors leads to selecting members of more than one class.

3.2 Logistic Regression

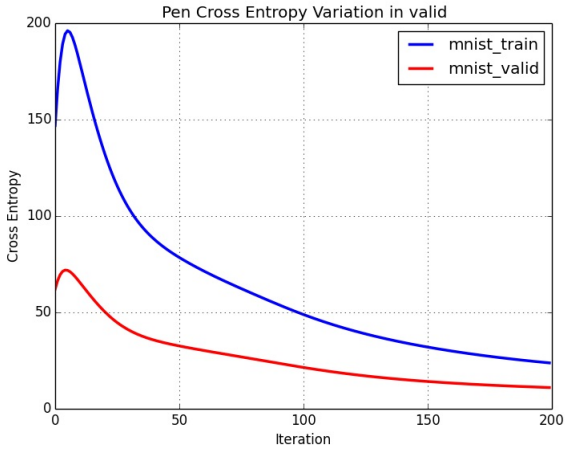
3.2.1 Report which hyperparameters settings worked the best and the final cross entropy and classification error on the training, validation and test sets.

The hyperparameters I found worked best were:

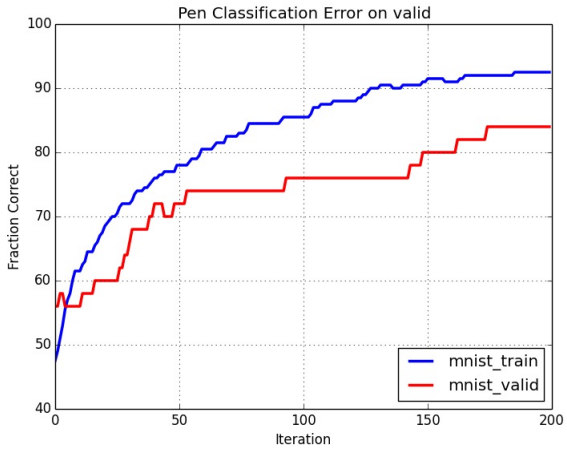
- learning rate: 0.09
- num_iterations: 200

With those parameters, the results obtained were the following:
The results for the validation and the test set against the training set can be viewed in figures 3.3 and 3.4.

Set	Cross Entropy	Classification Error
mnist_train	12.437593	4.50
mnist_valid	15.998257	22.00
mnist_train	14.715843	7.50
mnist_test	11.355649	22.00

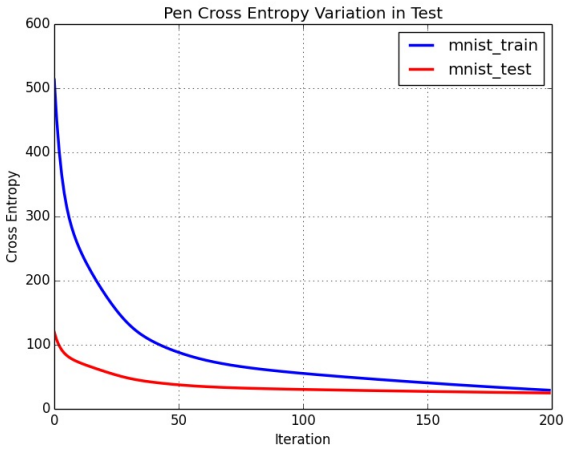


V 3 2 1.jpg V 3 2 1.jpg

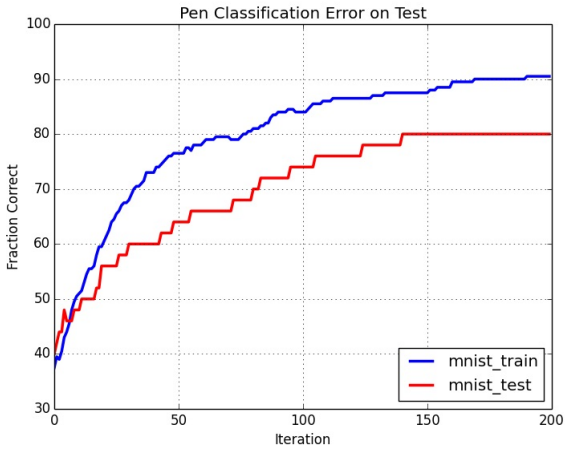


V 3 2 1.jpg V 3 2 1.jpg

Figure 3.3: Results for the hyperparameters using mnist_train and mnist_valid. Top displays cross entropy variation in each iteration. Bottom shows the classification rate in each iteration.



T 3 2 1.jpg T 3 2 1.jpg



T 3 2 1.jpg T 3 2 1.jpg

Figure 3.4: Results for the hyperparameters using mnist_train and mnist_test. Top displays cross entropy variation in each iteration. Bottom shows the classification rate in each iteration.

3.2.2 Report how the loss changes during training.Do the results change over time?

During training, cross entropy decreases its value, usually at an accelerated pace during the first iterations, and afterwards in smaller steps. For all graphs in figure 3.5 it is possible to see how most of the time, the same pattern is followed by the training set and the validation set. This shows that the data within each set is distributed in a similar manner. It is also possible to see that for the complete training set, the curve for validation and training is much more similar. This is a direct effect of having more data, a better regression may be calculated which fits the new data.

For various iterations of the code, it is possible to notice how the results vary within a certain range. This is attributed to the variation in the weights, they are randomized every time, and also in the variation of the fit of the distribution that is achieved in each repetition. Yet the curve is still similar in shape after each repetition.

3.3 Regularized logistic regression

3.3.1 Report the best hyperparameters and corresponding test set error

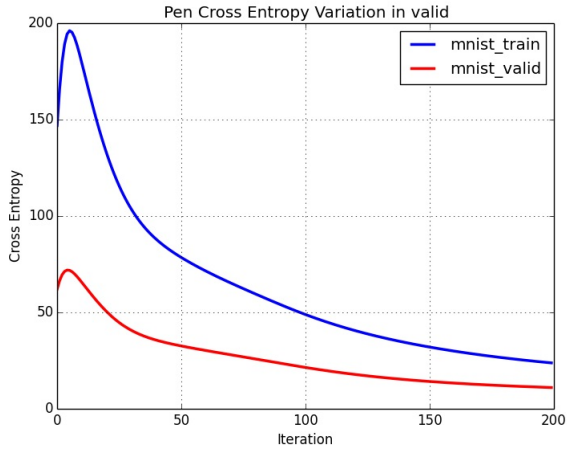
The hyperparameters I found worked best for the mnist_train_small were:

- learning rate: 0.01
- num_iterations: 50
- weight decay: 1

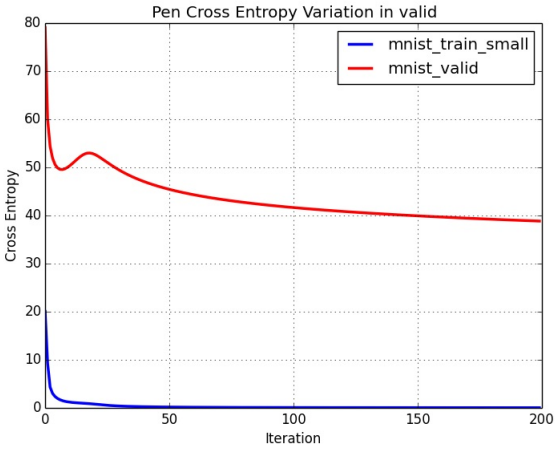
and for the mnist_train set:

- learning rate: 0.5
- num_iterations: 200
- weight decay: 0.01

Table 3.1 portrays the cross entropy and errors reported in the validation and test set after running the script using the hyperparameters previously discussed for the mnist_train set. Figure 3.6 displays the results for the cross entropy variation in the validation and test sets, and compares the results obtained for the variation in the cross entropy in the larger data set (mnist_train) versus the smaller data set (mnist_train_small).



V 3 2 1.jpg V 3 2 1.jpg



small.jpg small.jpg

Figure 3.5: Results for the cross entropy variation using mnist_train and mnist_valid (Top), and mnist_train_small and mnist_valid (bottom).

Table 3.1: Cross entropy and classification error on regularized logistic regression.

Set	**Cross Entropy**	**Classification Error**
mnist_train	2.217406	0
mnist_valid	7.170649	10.00
mnist_train	4.441760	0.5
mnist_test	7.171276	4

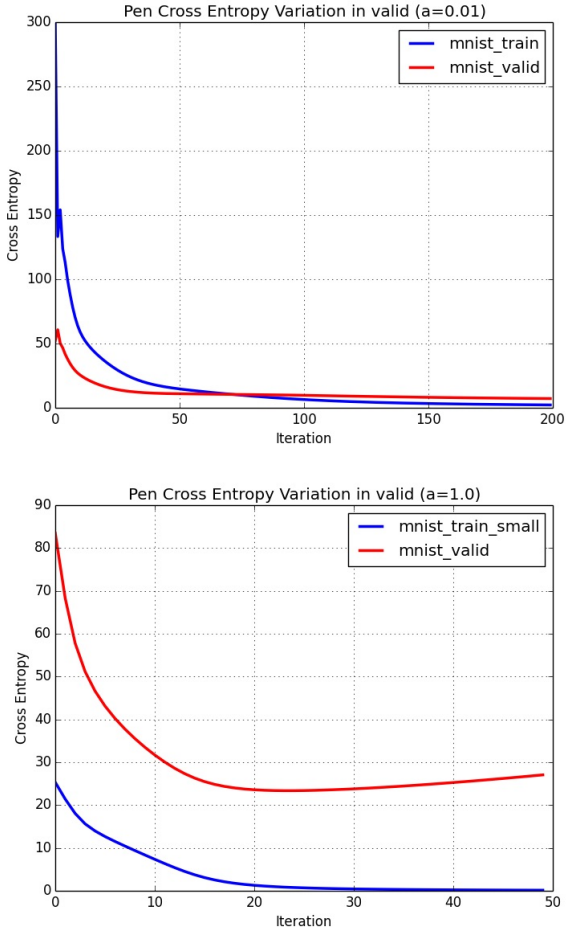


Figure 3.6: Results for the cross entropy variation using mnist_train and mnist_valid (Top), and mnist_train_small and mnist_valid (bottom).

3.3.2 Given other hyperparameters fized, how does the loss and classification error change when alpha is increased

When alpha is smaller, the cross entropy for the training set is smaller. Once an increase in alpha has occurred, the cross entropy for the training set increases, and a corresponding decrease in the cross entropy in the validation set is observed. The fraction of correct classifications also shows improvement. On the second increase of alpha, the cross entropy for the validation set increases greatly, and the fraction of correct classification experiences a decrease. At the third increase in the alpha hyperparameter, the cross entropy for the training set decreases while it increases for the validation. The fraction of correct classifications continues to decrease.

The general pattern observed is that it goes up and then down. I believe it behaves this way because it is finding a balance on how to change the weights. Specifically, how much to change them with respect to how they are at present time. It gives a value to the amount of change the learning rate may want to exert over the weights. For the larger dataset, I found that an alpha of 0.01 was the best choice, which is corroborated during the alpha increasing experiment.

3.3.3 Logistic regression with regularizer vs logistic regression

Comparing figures 3.5 (results for both train sets and the validation set on the logistic regression) and 3.6 (results for both train sets and the validation set on the regularized logistic regression). For the mnist_train_small data set, the regularized logistic regression performs better, reducing the cross entropy in 20 iterations. For the non regularized case, it would take the algorithm, about 100 iterations to achieve great change in the cross entropy.Even after the 100 iterations, the cross entropy would still be higher than that achieved by the regularized algorithm.

For the larger data set, mnist_train, the regularized algorithm achieves a greater reduction in the cross entropy than the non regularized algorithm for any given period of time. This would all point towards the fact that having a regularized algorithm is better for this problem because it allows to control how much the variable will change, how much it will be penalized for changing. This way it wont change always in the same amount, but verify how much it needs to change in order to achieve its goal.

3.3.4 Logistic regression with regularizer vs logistic regression vs kNNs

All though for kNNs the cross entropy is not available, it has the best results in the classification rate, achieving a classification error of 2% in the test set. For the regularized logistic regression, the classification error is 4%, and 22% classification error in the logistic regression. With this information, the worst performer is the logistic regression. It is followed by the regularized logistic regression, and finally kNNs. I would choose one parameter over another depending on how the data is distributed, how the decision boundary would be, and

the amount of data I have. I believe that it is easier to overfit data using kNN if the amount of data points are scarce.

The pros they have are: - non parametric
- could classify more than two classes

For the case of the kNN, the decision boundary is very flexible, however, it needs more memory in order to store the distribution of its space and to be able to calculate the distance to the neighbors.

The logistic regression offers the possibility of controlling certain hyperparameters that allow the model to learn at a faster rate. The con (to this pro) is that sometimes the changes made may produce very different results, they are not linear.