10707 Homework 1

David S. Hippocampus*
Department of Computer Science
Cranberry-Lemon University
Pittsburgh, PA 15213
hippo@cs.cranberry-lemon.edu

Problem 1 (6 pts)

This question will test your general understanding of overfitting as they relate to model complexity and training set size. Consider a continuous domain and a smooth joint distribution over inputs and outputs, so that no test or training case is ever duplicated exactly.

- For a fixed training set size, sketch a graph of the typical behavior of training error rate (y-axis) versus model complexity (x-axis). Add to this graph a curve showing the typical behavior of the corresponding test error rate versus model complexity, on the same axes. (Assume that we have an infinite test set drawn independently from the same joint distribution as the training set). Mark a vertical line showing where you think the most complex model your data supports is; chose your horizontal range so that this line is neither on the extreme left nor on the extreme right. Indicate on your vertical axes where zero error is and draw your graphs with increasing error upwards and increasing complexity rightwards.

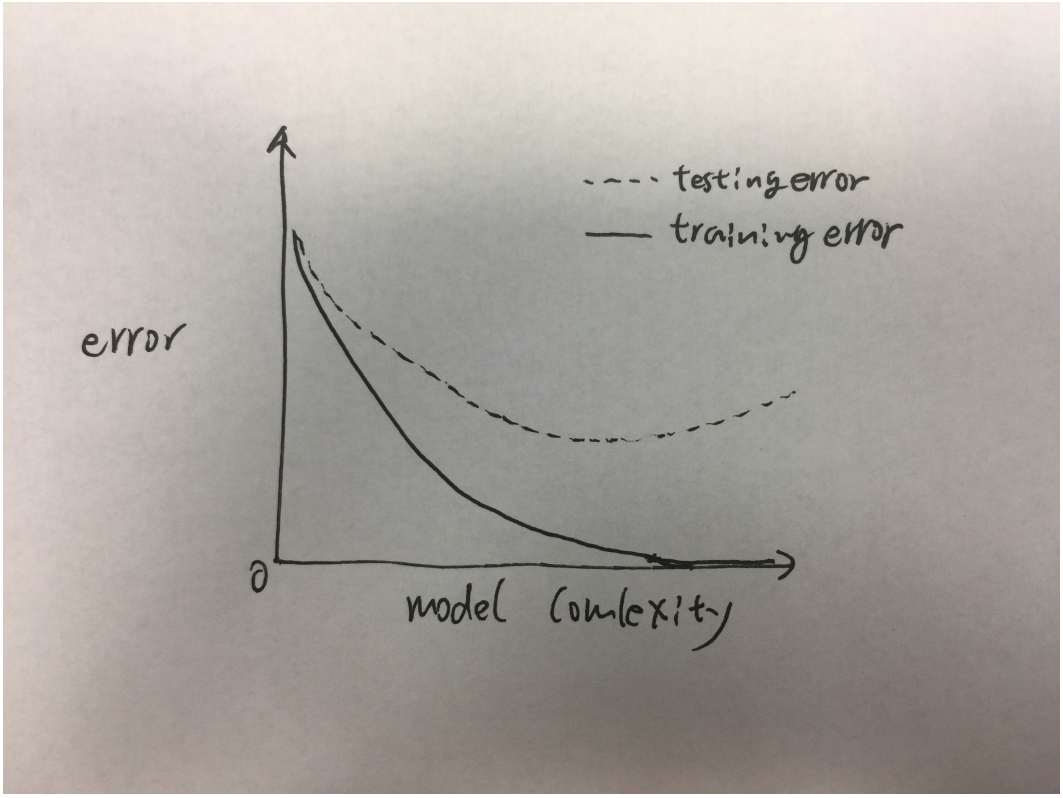
- For a fixed model complexity, sketch a graph of the typical behavior of training error rate (y-axis) versus training set size (x-axis). Add to this graph a curve showing the typical behavior of test error rate versus training set size, on the same axes (again on an iid infinite test set).

- One of the commonly used regularization methods in neural networks is *early stopping*. Argue qualitatively why (or why not) early stopping is a reasonable regularization metric.

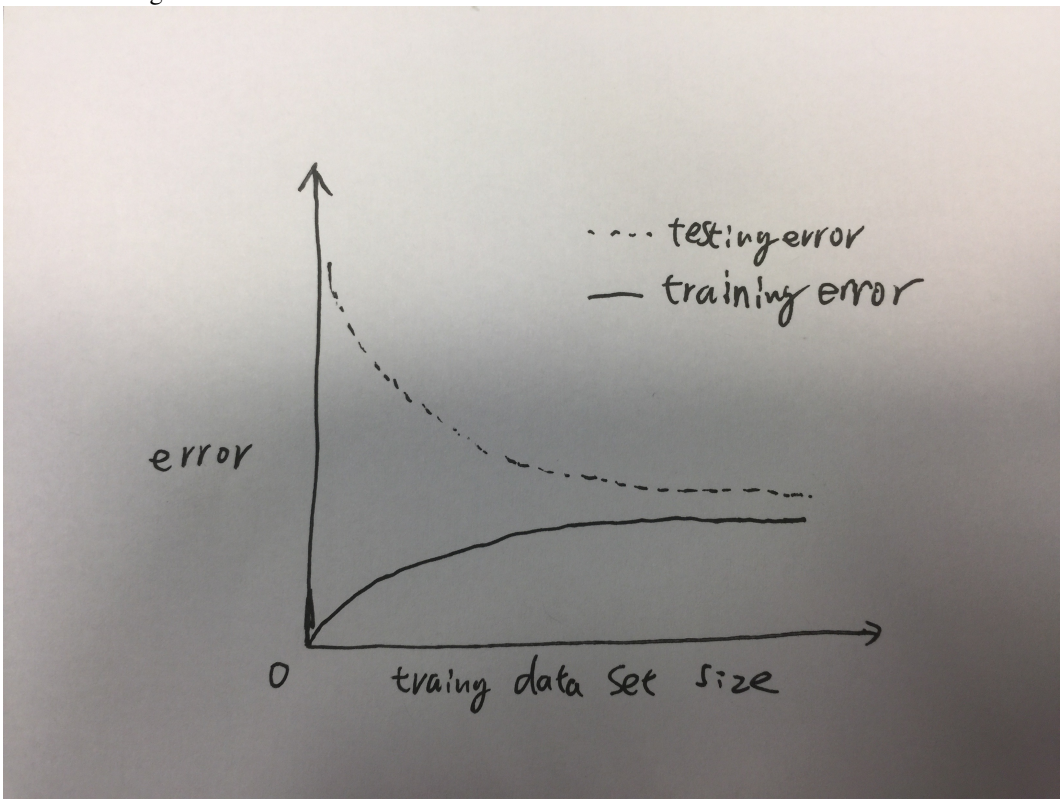
Your answer here

- Training error decreases with model complexity, testing error decreases at first, however it eventually increases due to over fit.

*Use footnote for providing further information about author (webpage, alternative address)—*not* for acknowledging funding agencies.



- Training error increase with the size of training data set. Testing error decreases with the size of training data set



- Early stopping is a reasonable regularization metric because when the training error stops improving, the testing error in the validation set may start to increase due to over fitting. Early stop prevents the model from over fitting.

Problem 2 (8 pts)

Consider N training points (x_i, y_i) drawn i.i.d from a distribution that is characterized as:

$$x_i \sim h(x) \tag{1}$$
$$y_i = f(x_i) + \epsilon_i \tag{2}$$
$$\epsilon_i \sim (0, \sigma^2) \tag{3}$$

where f is the regression function. An estimator for this data, *linear* in y_i is given by

$$\hat{f}(x^*) = \sum_{i=1}^{N} l_i(x^*; \mathcal{X}) y_i, \tag{4}$$

where $l_i(x^*; \mathcal{X})$ depends on the entire training sequence of x_i (denoted by $\mathcal{X}$) but do not depend on y_i. Show that k-nearest-neighbor regression and linear regression are members of this class of estimators. What would be the $l_i(x^*; \mathcal{X})$ in both these regressions (knn and linear)?

Your answer here

In linear regression the estimator is given as:

$\hat{f}(x^*) = (x^*)^T \beta$, and $\beta = (X^T X)^{-1} X^T y$.

Therefore $\hat{f}(x^*) = \sum_{i=1}^{N} ((x^*)^T (X^T X)^{-1} X^T)_i y_i$

$l_i(x^*; \mathcal{X}) = ((x^*)^T (X^T X)^{-1} X^T)_i$

In KNN estimator is given as: $f(x^*) = \sum_{i=1}^{N} \frac{y_i}{k} 1_{x i \in N_k((x^*))}$

where $N_k(x_0)$is the k nearest neighbour of x_0.

$l_i(x^*; \mathcal{X}) = \frac{1}{k} 1_{x i \in N_k((x^*))}$

Problem 3 (6 pts)

The form of Bernoulli distribution, given by:

$$\text{Bern}(x|\mu) = \mu^x (1-\mu)^{1-x},$$

is not symmetric between the two values of $x \in \{0, 1\}$. Often, it will be convenient to use an equivalent formulation for which $x \in \{-1, 1\}$, in which case the distribution can be written as:

$$p(x|\mu) = \left(\frac{1-\mu}{2}\right)^{(1-x)/2} \left(\frac{1+\mu}{2}\right)^{(1+x)/2},$$

where $\mu \in [-1, 1]$. Show that this new distribution is normalized and compute its mean, variance, and entropy.

Your answer here

$p(x = 1|\mu) = \frac{1+\mu}{2}$

$p(x = -1|\mu) = \frac{1+\mu}{2}$

E[x] = 1*p(x=1 | μ) $- 1 * p(x = -1|\mu) = \mu$

var(x) = $\sum_{i=1}^{N} p_i x_i^2 - \mu^2 = 1 - \mu^2$

entropy=-$\sum_{i=1}^{N} (p_i log(p_i)) = -(\frac{1-\mu}{2} log(\frac{1-\mu}{2}) + \frac{1+\mu}{2} log(\frac{1+\mu}{2}))$

Problem 4 (12 pts)

Consider a binary classification problem in which the target values are $t \in \{0, 1\}$, with a neural network output $y(x, w)$ that represents $p(t = 1|x; w)$, and suppose that there is a probability ϵ that the class label on a training data point has been incorrectly set. Assuming independent and identically distributed data, write down the error function corresponding to the negative log likelihood. What is the error function when $\epsilon = 0$? Note that this error function makes the model robust to incorrectly labelled data, in contrast to the usual error function.

Your answer here

when $\epsilon = 0$ error function is :

$E = -\sum_{.}^{N}\{t_n ln(y_n) + (1 - t_n)ln(1 - y_n)\}$

when $\epsilon! = 0$ error function is :

$E = -\sum_{n=1}^{N}\{t_n ln(y_n(1 - \epsilon) + \epsilon(1 - y_n)) + (1 - t_n)ln(1 - (y_n(1 - \epsilon) + \epsilon(1 - y_n)))\}$.

Problem 5 (8 pts)

Consider a two-layer network function in which the hidden-unit nonlinear activation functions are given by logistic sigmoid functions of the form:

$$\sigma(a) = \frac{1}{1 + \exp(-a)} \tag{5}$$

Show that there exists an equivalent network, which computes exactly the same function, but with hidden unit activation functions given by $\tanh(a)$ where the tanh function is defined by:

$$\tanh(a) = \frac{e^a - e^{-a}}{e^a + e^{-a}} \tag{6}$$

Your answer here

$\tanh(a) + 1 = \frac{e^a - e^{-a}}{e^a + e^{-a}} + \frac{e^a + e^{-a}}{e^a + e^{-a}} = \frac{2e^a}{e^a + e^{-a}} = \frac{2}{1 + e^{-2a}} = 2\sigma(2a)$

Therefore $\sigma(a) = \frac{\tanh(\frac{a}{2})}{2} + \frac{1}{2}$.

That is to say if we perform some scaling of 1/2 to the input and scaling of 1/2 to the output and apply a bias of 1/2 we can get back the same result, which means that these 2 activation functions are essentially the same.

Problem 6 (60 pts)

For this question you will write your own implementation of backpropagation algorithm for training your own neural network. Please do not use any toolboxes. We recommend that you use MATLAB or Python, but you are welcome to use any other programming language if you wish.

The goal is to label images of 10 handwritten digits of "zero", "one",..., "nine". The images are 28 by 28 in size (MNIST dataset), which we will be represented as a vector x of dimension 784 by listing all the pixel values in raster scan order. The labels t are 0,1,2,...,9 corresponding to 10 classes as written in the image. There are 3000 training cases, containing 300 examples of each of 10 classes, 1000 validation (100 examples of each of 10 classes), and 3000 test cases (300 examples of each of 10 classes). they can be found in the file digitstrain.txt, digitsvalid.txt and digitstest.txt: http://www.cs.cmu.edu/∼rsalakhu/10707/assignments.html

Format of the data: digitstrain.txt contains 3000 lines. Each line contains 785 numbers (comma delimited): the first 784 real-valued numbers correspond to the 784 pixel values, and the last number denotes the class label: 0 corresponds to digit 0, 1 corresponds to digit 1, etc. digitsvalid.txt and

digitstest.txt contain 1000 and 3000 lines and use the same format as above. As a warm up question, load the data and plot a few examples. Decide if the pixels were scanned out in row-major or column-major order.

Backpropagation Algorithm

Implement backpropagation algorithm with sigmoid activation function in a single-layer neural network. The output layer should be a softmax output over 10 classes corresponding to 10 classes of handwritten digits. Your backprop code should minimize the cross-entropy entropy function for multi-class classification problem.

a) Basic generalization [5 points]

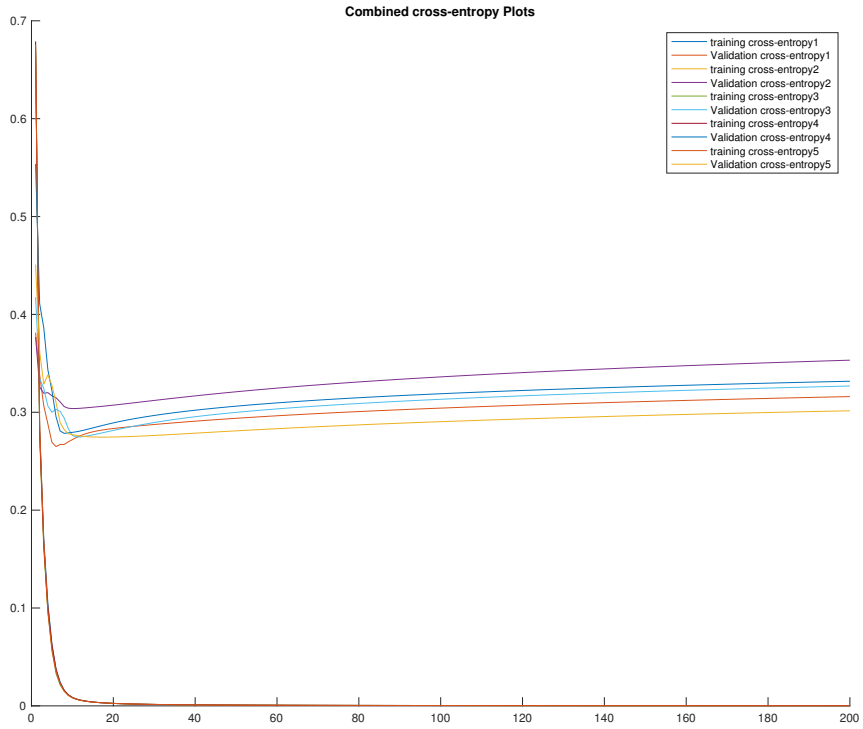
Train a single layer neural network with 100 hidden units (with architecture: $784 \rightarrow 100 \rightarrow 10$). You should use initialization scheme discussed in class as well as choose a reasonable learning rate (e.g. 0.1). Train the network repeatedly (more than 5 times) using different random seeds, so that each time, you start with a slightly different initialization of the weights. Run the optimization for at least 200 epochs each time. If you observe underfitting, continue training the network for more epochs until you start seeing overfitting.

Plot the average training cross-entropy error (sum of the cross-entropy error terms over the training dataset divided by the total number of training example) on the y-axis vs. the epoch number (x-axis). On the same figure, plot the average validation cross-entropy error function.

Examine the plots of training error and validation error (generalization). How does the network's performance differ on the training set versus the validation set during learning? Use the plot of error curves (training and validation) to support your argument.

Your answer here

Training set cross entropy error goes close to zero quickly after approximately 10 epochs. However, the cross-entropy error for the validations sets remained staidly close to 0.3, and is increasing through out the training.

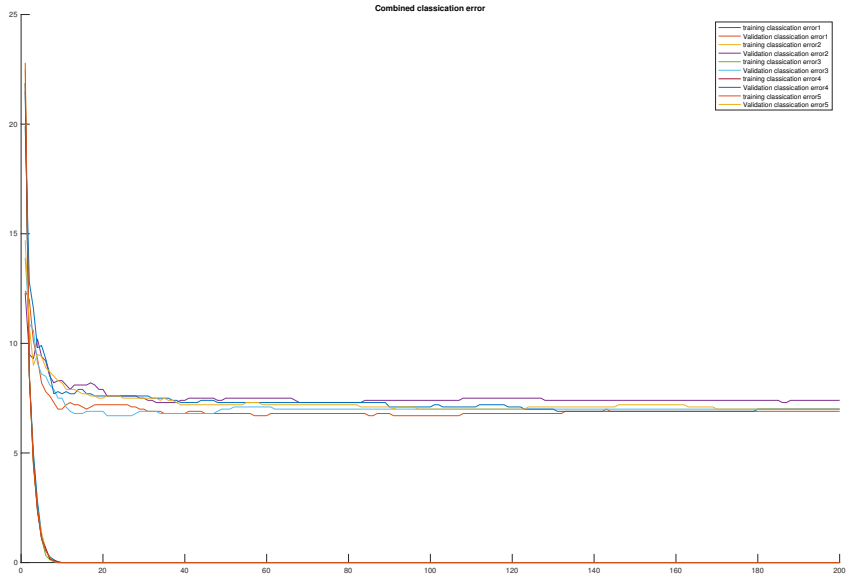


b) Classification error [5 points]

You should implement an alternative performance measure to the cross entropy, the mean classification error. You can consider the output correct if the correct label is given a higher probability than the incorrect label. You should then count up the total number of examples that are classified incorrectly (divided by the total number of examples) according to this criterion for training and validation respectively, and maintain this statistic at the end of each epoch. Plot the classification error (in percentage) vs. number of epochs, for both training and validation. Do you observe a different behavior compared to the behavior of the cross-entropy error function?

Your answer here

Generally the behaviors of classification error and cross-entropy are very similar with the training set goes to zero quickly and the validation set stabled at some level.

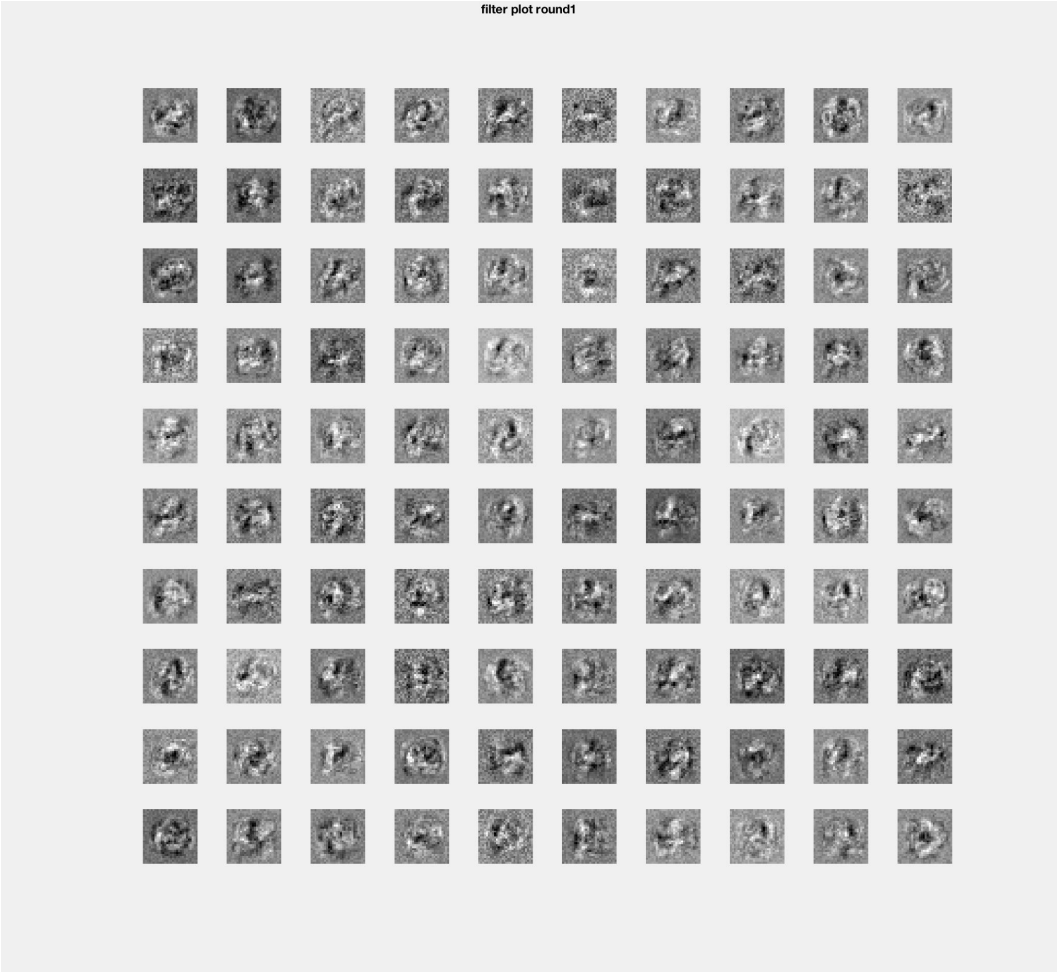


c) Visualizing Parameters [5 points]

Visualize your best results of the learned W as 100 28x28 images (plot all filters as one image, as we have seen in class). Do the learned features exhibit any structure?

Your answer here

In round 1 of 5, I used rng(1) seed in Matlab and obtained the best performance, and here is a plot. The learned features have some rounded structure which may be exacting certain features from the raw digits as digits genernaly appears in this round region.

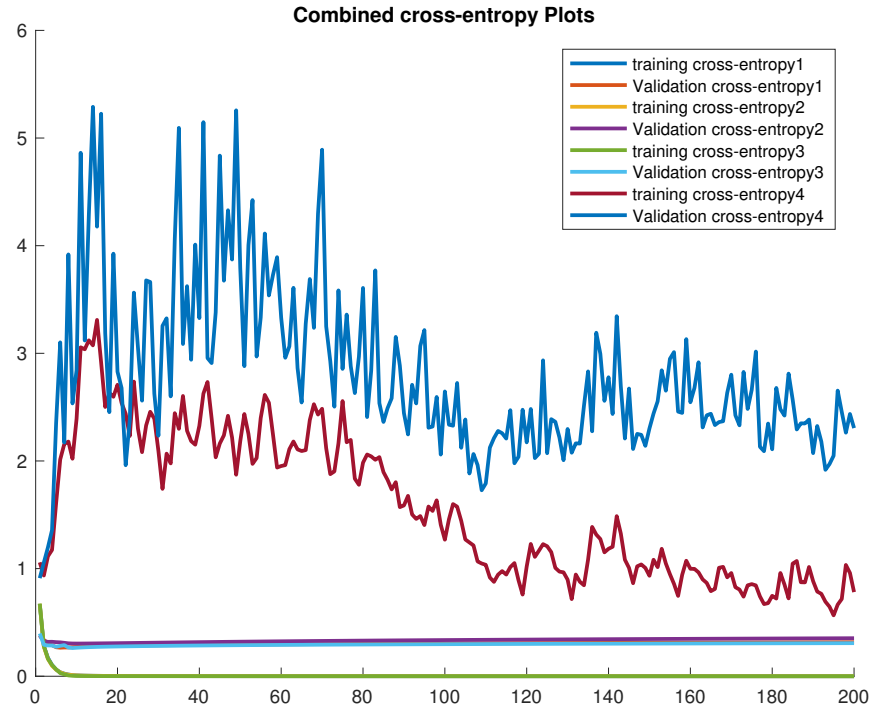


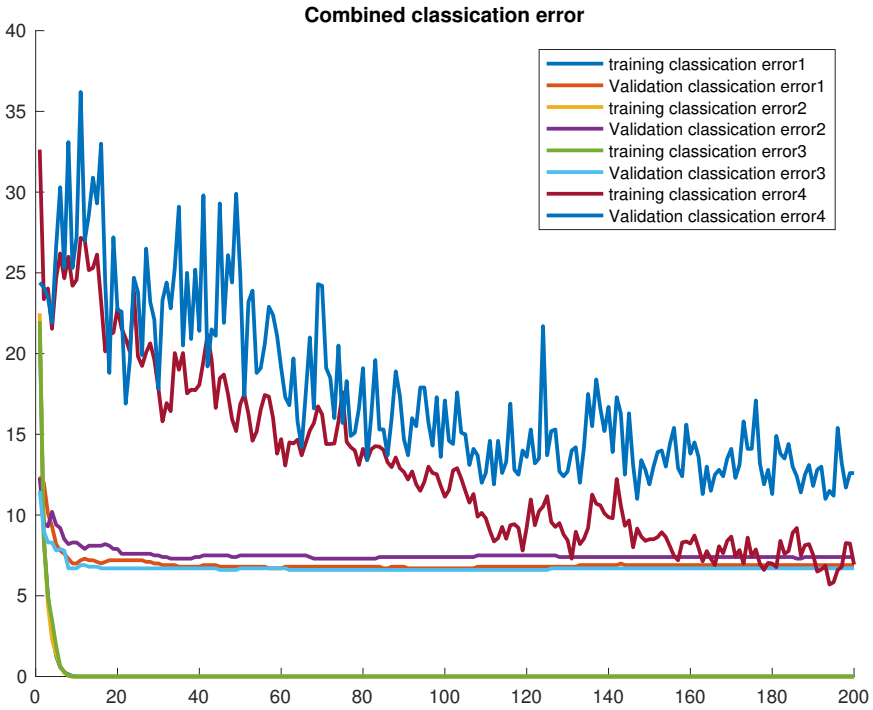
d) Learning rate [5 points]

Try different values of the learning rate ϵ. You should start with a learning rate of 0.1. You should then reduce it to .01, and increase it to 0.2 and 0.5. What happens to the convergence properties of the algorithm (looking at both average cross entropy and % Incorrect)? Try momentum of $\{0.0, 0.5, 0.9\}$. How does momentum affect convergence rate? How would you choose the best value of these parameters?

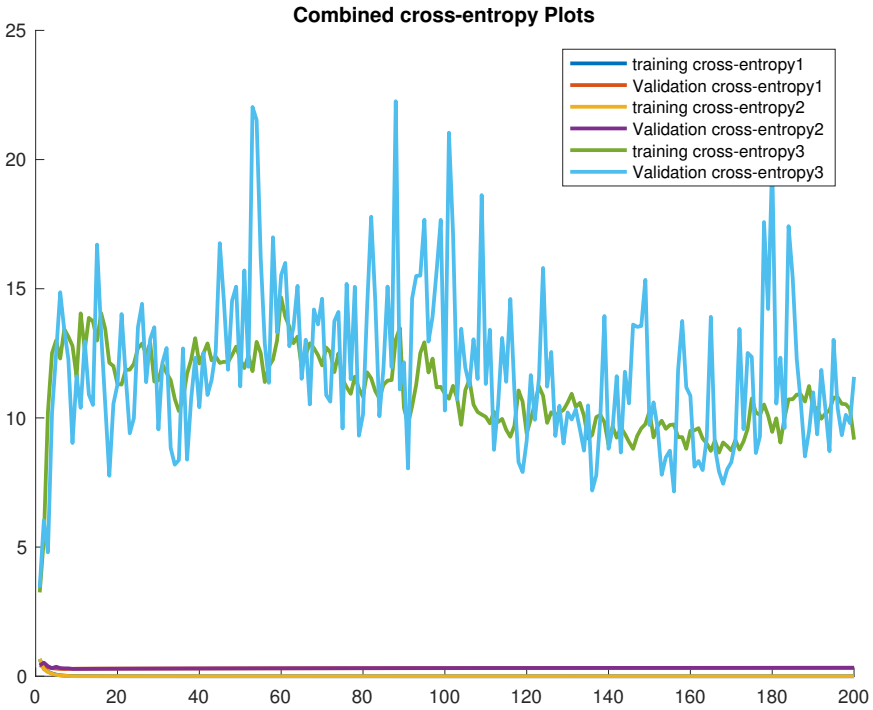
Your answer here

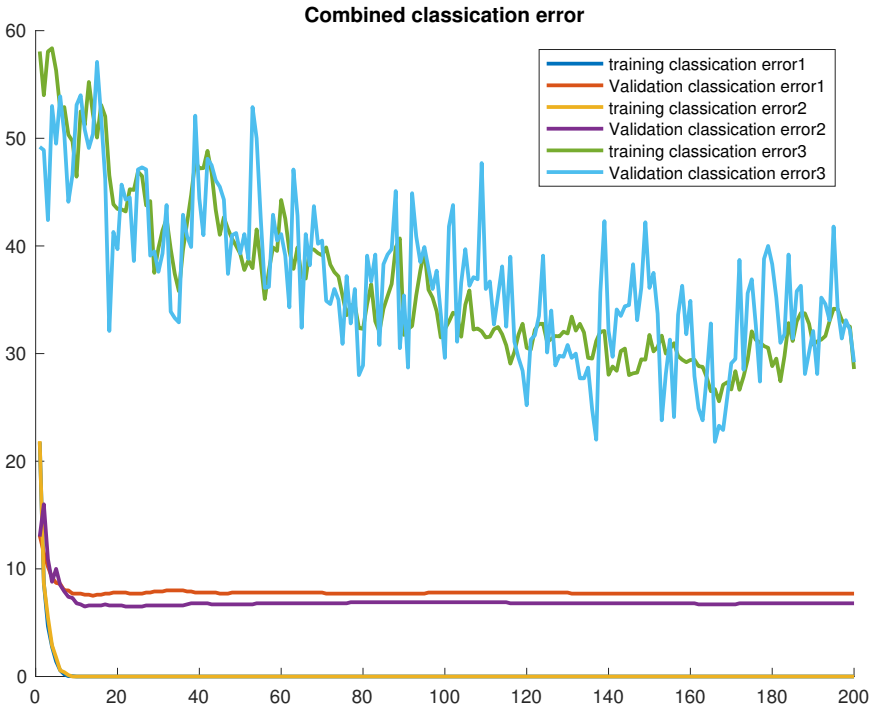
The the number groups 1,2,3 and 4 co-respond to learning rate 0.001,0.1,0.2,and 0.5. The algorithm was not able to converge with learning rate of 0.5, and other learning rate exhibits similar convergence property. They all converged before 20 epochs.





The the number groups 1,2,and 3 co-respond to momentum 0.0, 0.5, and 0.9. The algorithm was not able to converge with momentum of 0.9. The group with momentum of 0.5 shows the best validation accuracy.


Combined cross-entropy Plots
training cross-entropy1
Validation cross-entropy1
training cross-entropy2
Validation cross-entropy2
training cross-entropy3
Validation cross-entropy3
25
20
15
10
5
0
0
20
40
60
80
100
120
140
160
180
200


Combined classication error
training classication error1
Validation classication error1
training classication error2
Validation classication error2
training classication error3
Validation classication error3
60
50
40
30
20
10
0
0
20
40
60
80
100
120
140
160
180
200

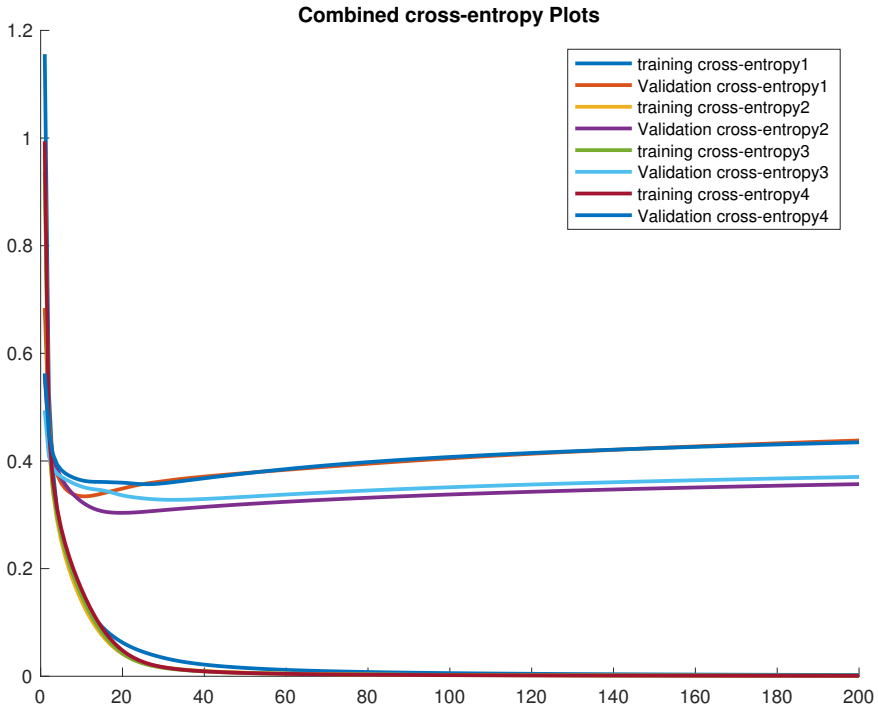
e) Number of hidden units [5 points]

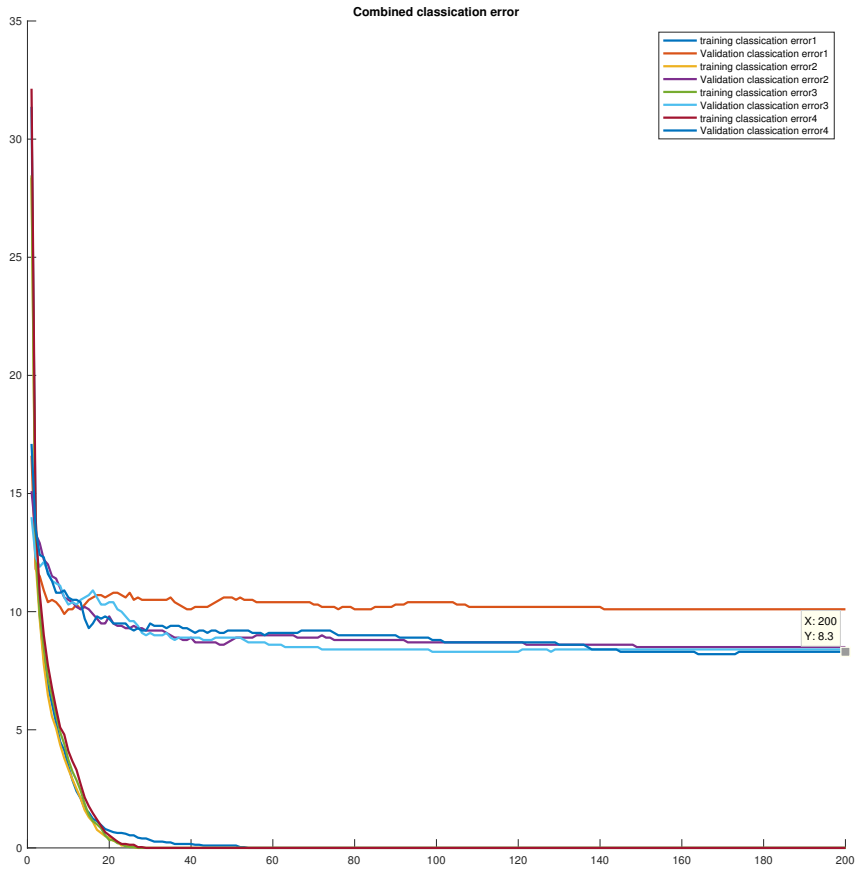
Set the learning rate ϵ to $.01$, momentum to 0.5 and try different numbers of hidden units on this problem. You should try training a network with 20, 100, 200, and 500 hidden units. Describe the effect of this modification on the convergence properties, and the generalization of the network.

Your answer here

The the number groups 1,2,3 and 4 co-respond to number of hidden units of 20, 100, 200, and 500.As can be seen from the cross-entropy and classification error graph, the group with 20 hidden units converged last, this may be caused by the over simplification of the model so it encountered difficulties to converge during training. The rest of the models have similar convergence properties.

With 20 hidden units, the model is the least generalized among all 4, with a classification error of more than 10%. The rest of the models shows similar validation error, so that means that increasing number of hidden units beyond 100 does not really help the training or accuracy.





f) Best performing single-layer network [10 points]

Cross-validation: Explore various model hyper-parameters, including

- learning rates
- momentum
- number of hidden units in each layer
- number of epochs (early stopping)
- L_2 regularization (weight decay)

to achieve the best validation accuracy. Briefly describe your findings.

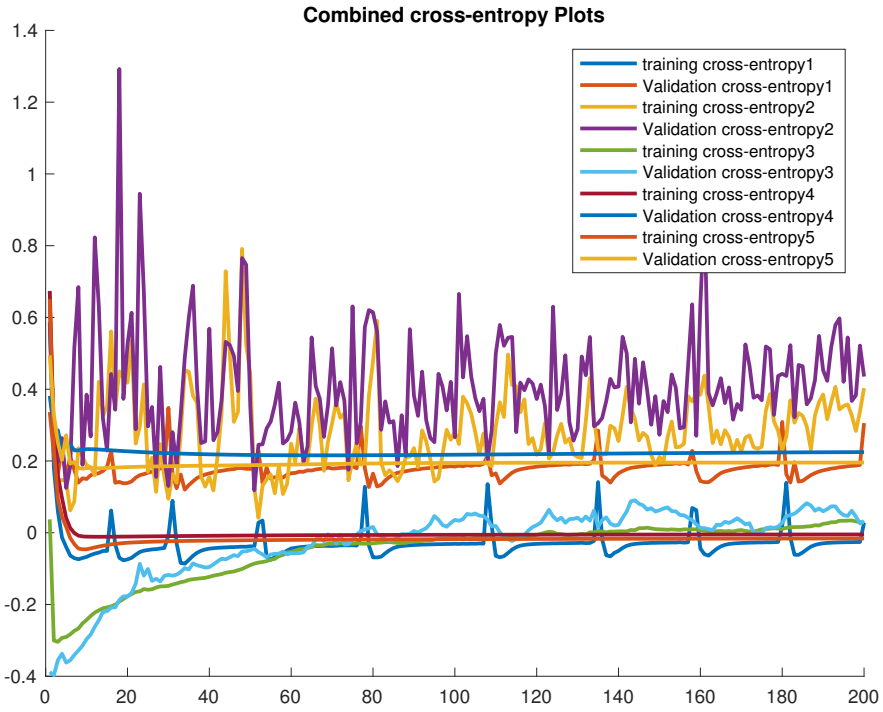
Given the best found values, report the final performance of your 1-layer neural network (both average cross entropy and % Incorrect) on the training, validation, and test sets. Visualize your best results of the learned W as 28x28 images (plot all filters as one image, as we have seen in class).

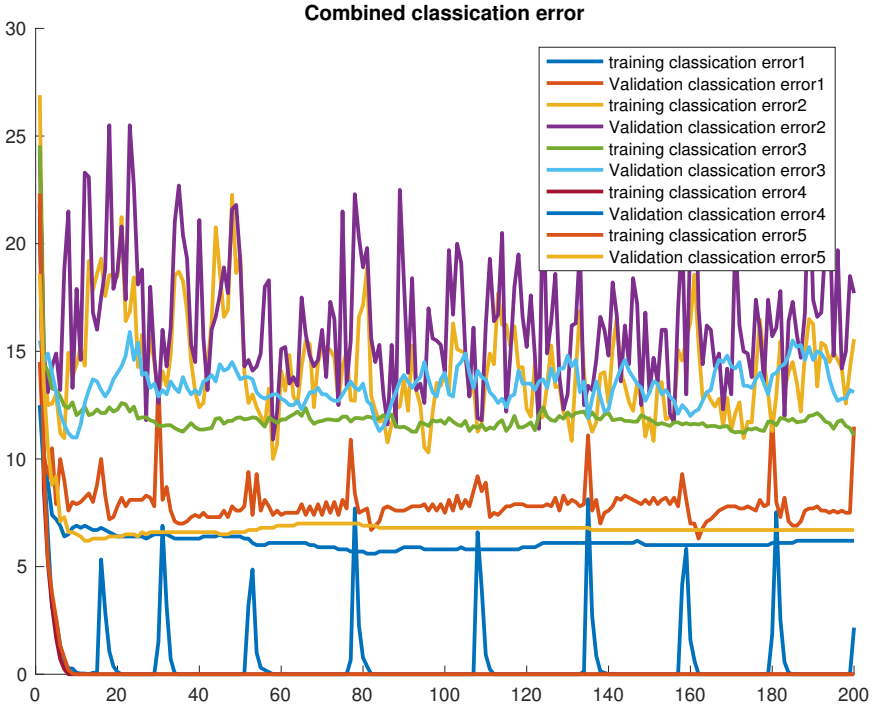
Your answer here

From the result of previous questions, I chose learning rate = 0.1 or 0.2, momentum =0.5, number of hidden units = 100 to find the best result, and from the scale of the weights which is around e-10, I tried with L2 regularization with 0.0001,0.00001,0.00005, here is a list of combinations I tried.

round	learning rate	regularization λ	momentum
1	0.1	0.0001	0.5
2	0.2	0.0001	0.5
3	0.1	0.001	0.5
4	0.1	0.00001	0.5
5	0.1	0.00005	0.5

As can be seen from the below cross entropy and classification error graph, round 4 which is the combination with learning rate 0.1, momentum 0.5 and L2 regularization parameter $\lambda = 0.00001$ shows the best performance. For $\lambda > 0.00005$, which is the case of round 1 to 3, the model failed to converge. All of the modeled have been trained with 200 epochs, for round 1 ,2, 3, it is obvious that they will not converge, so there is no point of running more epochs. For round 4 and 5, the cross-entropy and classification error stabled after approximately 80 epochs. Although there were no sign of over fitting I would suggest the training to stop earlier just to save time.

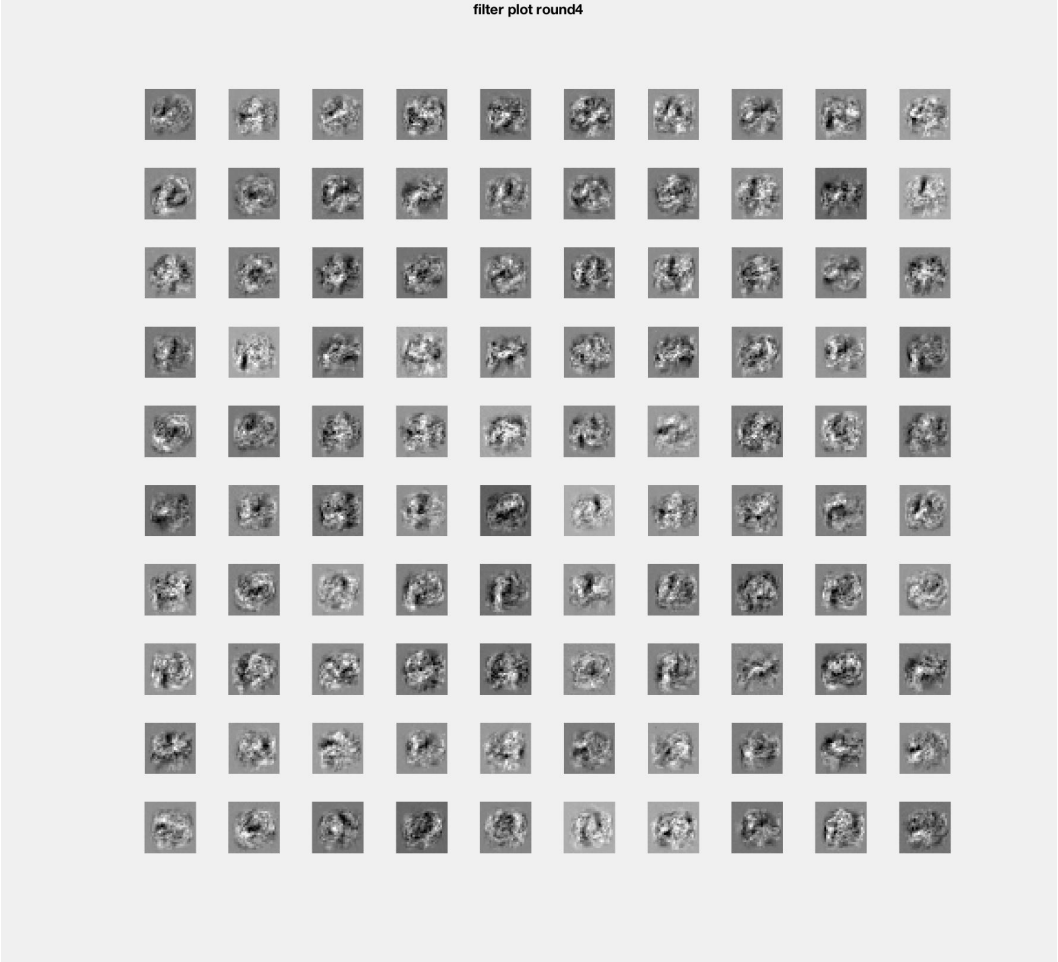




The final performance can be found from the table below

Data Set	Cross-entropy	Classification Error
training	-0.005	0%
validation	0.2247	6.2%
testing	0.2958	7.9667%

Visualization of the learned W:



g) Extension to multiple layers [10 points]

Implement a 2-layer neural network, starting with a simple architecture containing 100 hidden units in each layer (with architecture: $784 \rightarrow 100 \rightarrow 100 \rightarrow 10$).

Cross-validation: Explore various model hyper-parameters, including learning rates, momentum, number of hidden units in each layer, number of epochs, and weight decay to achieve the best validation accuracy. Briefly describe your findings.

Given the best found values, report the final performance of your 2-layer neural network (both average cross entropy and % Incorrect) on the training, validation, and test sets. Visualize your best results of the learned 1st-layer W as 28x28 images (plot all filters as one image, as we have seen in class). How do these filters compare to the ones you obtained when training a single-layer network?

Does 1-layer network outperform a 2-layer model in term of generalization capabilities?

Your answer here

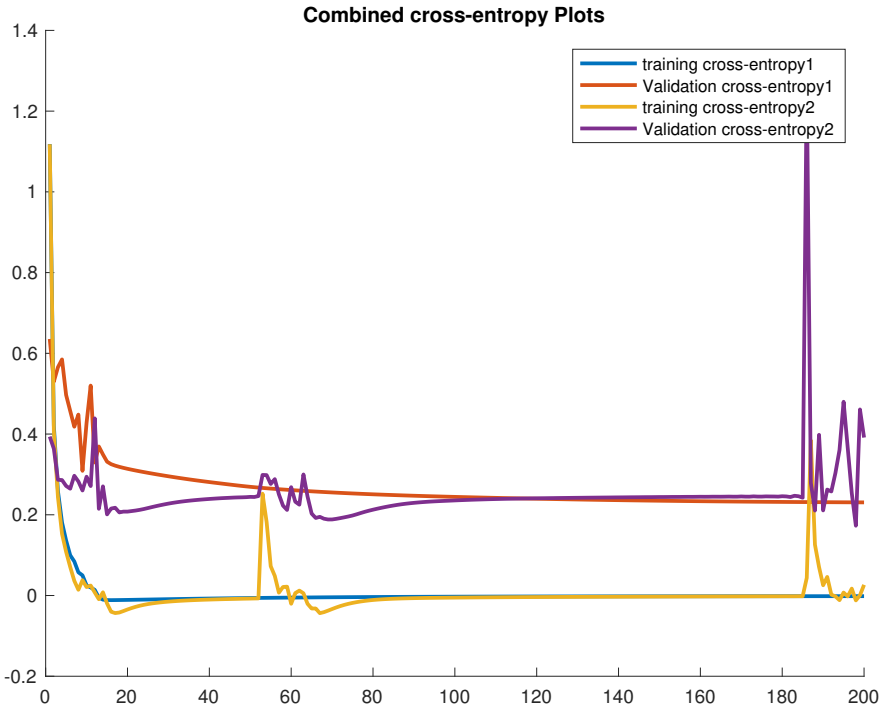
I have tried turning learning rate, momentum and number of hidden units individually and found that the combination of learning rate = 0.1, momentum = 0.5, and number of hidden units = 100 is the most suitable one. Increasing the number of hidden units does not improve the result significantly, but increased training time, therefore the number of hidden units are set to 100.

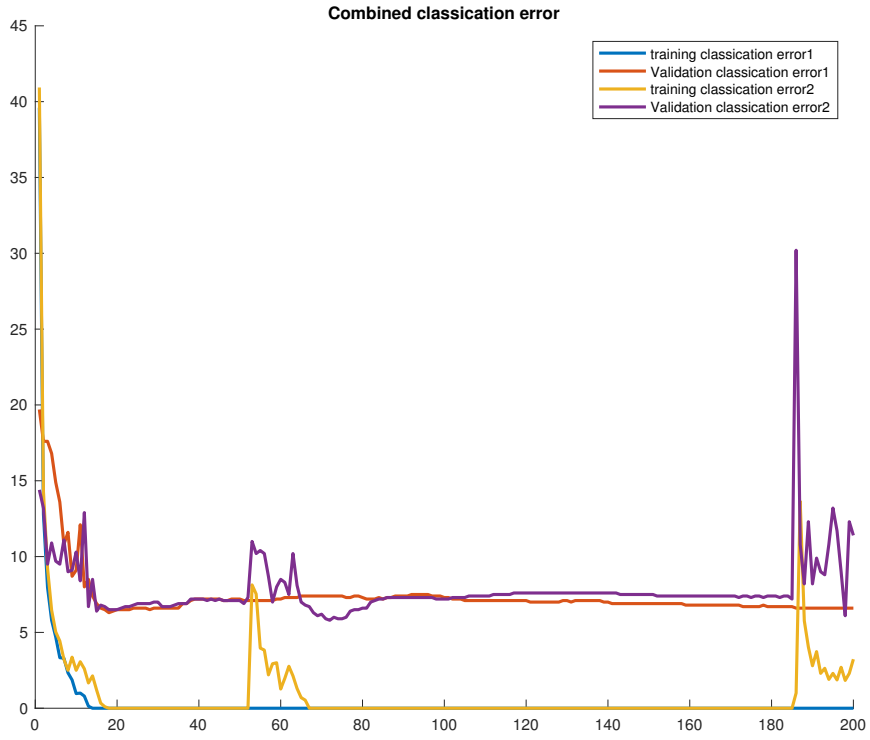
Here is a table of hyper-parameters used.

round	learning rate	regularization λ	momentum	No. of hidden units
1	0.1	0.00001	0.5	100
2	0.1	0.00005	0.5	100

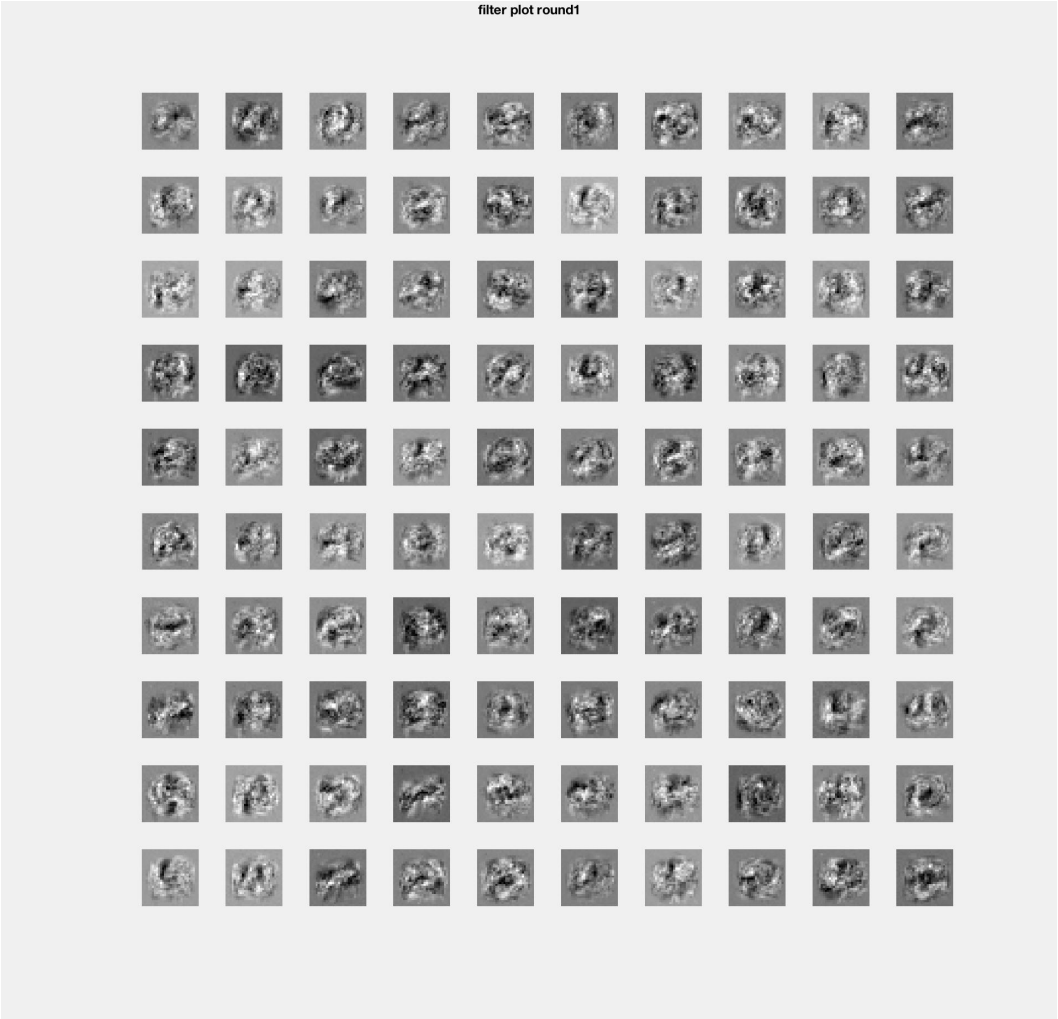
As can be seen from the cross-entropy and classification error graph, round 2 with $\lambda = 0.00005$ failed to converge, and round 1 with $\lambda = 0.00001$ gives similar result to the model with one hidden layer. The final performance of this model is shown in the table below. If we compare this result with that of 1 layer ANN, we can see that there is no improvement after adding another hidden layer. This may be due to the fact that 1 hidden layer ANN already has the ability to classify digits with a reasonable performance, adding another layer just adds complexity to the model and thus may not help.

Data Set	Cross-entropy	Classification Error
training	-0.0001594	0%
validation	0.2307	6.6%
testing	0.3142	7.6333%





Visualization of the filters:



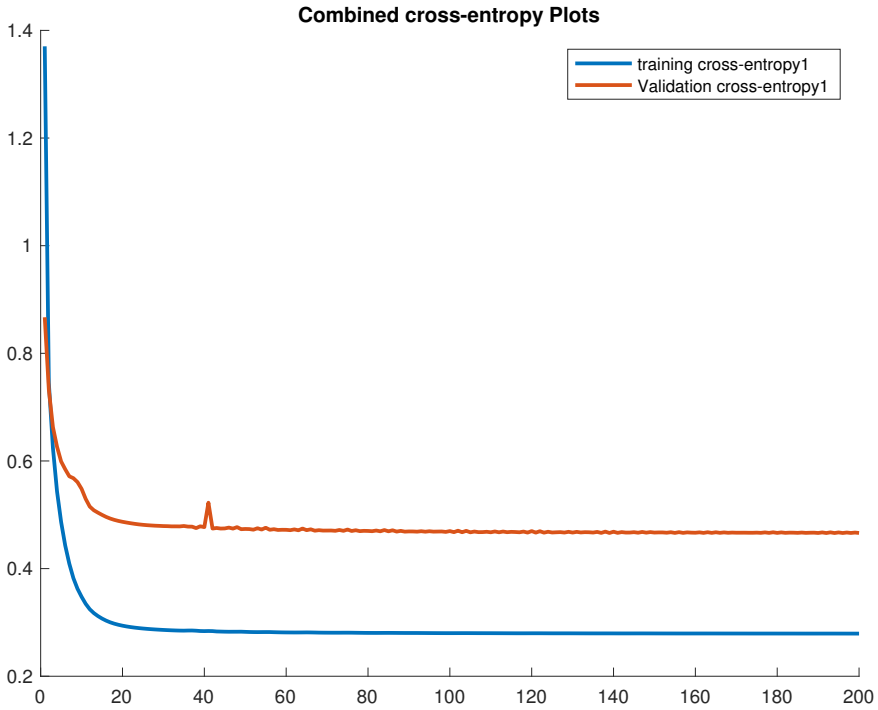
The filters for 2-layer network looks similar to the filter of 1-layer network. Visually, they all shows some round shape.

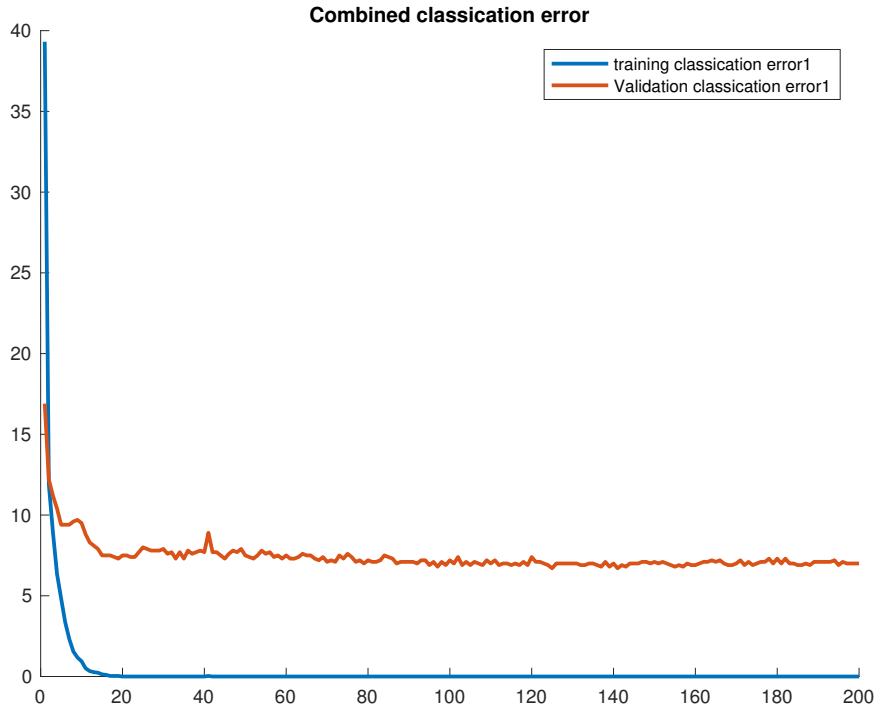
h) Batch Normalization [10 points]

For your two-layer network, implement batch normalization for a batch size of 32. Describe how batch norm helps (or doesn't help) in terms of speed and accuracy (train and validation).

Your answer here

After implementing the batch normalization, it can be seen from the cross-entropy and classification error that it does not help in terms of training speed or accuracy. The batch normalization group's training error goes to zero at round 20 epochs, this is similar to the group without batch normalization. The final validation set accuracy stays at 7.00% which is close to the 6.6% obtained from the group without BN. I believe this set of input data the model may have chosen some γ and β that actually unlearned the BN as the input data may already be very closed to normalized.


Combined cross-entropy Plots
training cross-entropy1
Validation cross-entropy1
1.4
1.2
1
0.8
0.6
0.4
0.2
0
20
40
60
80
100
120
140
160
180
200


Combined classication error
training classication error1
Validation classication error1
40
35
30
25
20
15
10
5
0
0
20
40
60
80
100
120
140
160
180
200

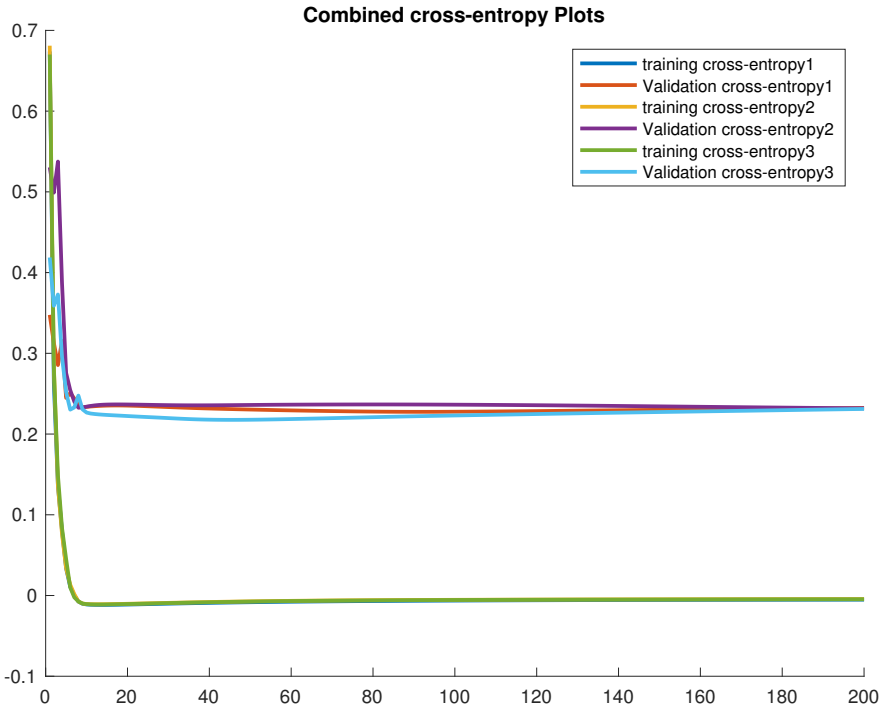
i) Different Activation Functions [5 points]

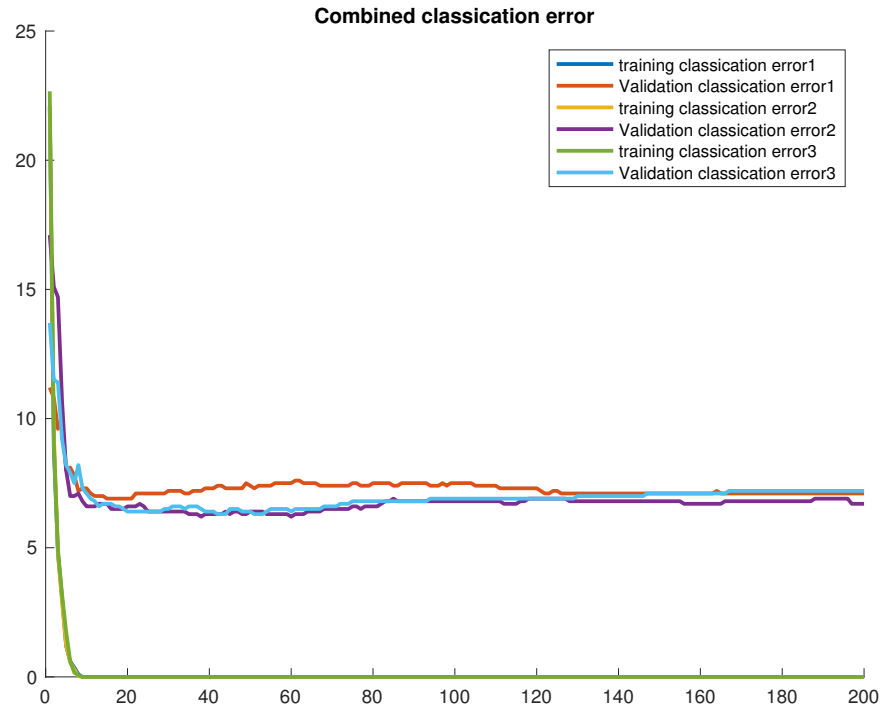
Now, change the activation functions to ReLU and tanh instead of the original sigmoid. Do you see any difference in terms of performance or accuracy ? Report your findings.

Your answer here

I tried ReLU, tanh, and sigmoid as activation function in 3 different rounds receptively. The corss-entropy and classification error are shown below. All of them are able to converge, with the ReLu around and sigmoid round having very close validation errors. tanh round shows slightly better validation error. Overall, the accuracy is very similar.

round	activation function
1	ReLU
2	tanh
3	sigmoid


Combined cross-entropy Plots
training cross-entropy1
Validation cross-entropy1
training cross-entropy2
Validation cross-entropy2
training cross-entropy3
Validation cross-entropy3
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
-0.1
0
20
40
60
80
100
120
140
160
180
200


Combined classication error
training classication error1
Validation classication error1
training classication error2
Validation classication error2
training classication error3
Validation classication error3
25
20
15
10
5
0
0
20
40
60
80
100
120
140
160
180
200