Assignment 4

Aidan Fischer

I pledge my honor that I have abided by the Stevens Honor System

Submission Instructions You shall submit a zip file named Assignment4_LastName_FirstName.zip which contains:

- python files (.ipynb or .py) including all the code, comments, plots, and result analysis. You need to provide detailed comments in English.
- pdf file including explanations and answers for non-coding questions.

1. **Nearest Neighbors** (10 points) Implement a basic k-NN model on the yeast dataset. The task is to predict the compartment in a cell that a yeast protein will localize to based on properties of its sequence. Apply cross-validation and report your best performance.

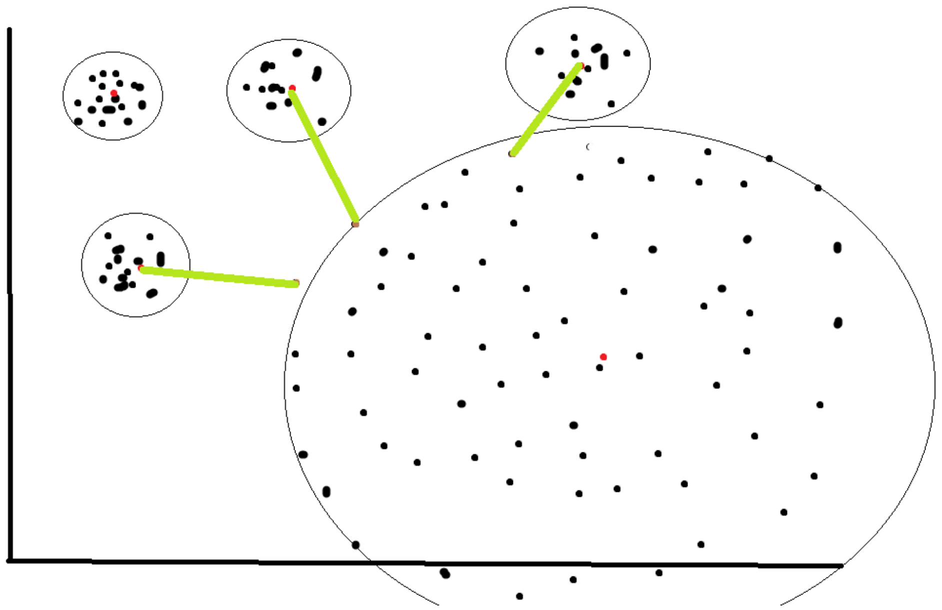
   Best model K: 6

   Best model error = 0.3422818791946309

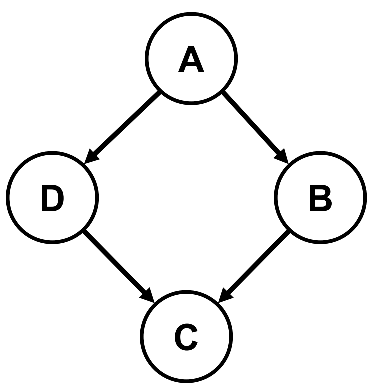
2. **Clustering** (5 points) Suppose we clustered a set of N data points using two different clustering algorithms: k-means and Gaussian mixtures. In both cases we obtained 5 clusters and in both cases the centers of the clusters are exactly the same. Can a few (say 3) points that are assigned to different clusters in the kmeans solution be assigned to the same cluster in the Gaussian mixture solution? If no, explain. If so, sketch an example or explain in 1-2 sentences.

   Yes.

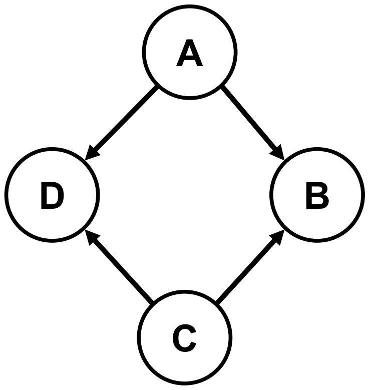
   In mixture of gaussians, say four of the clusters are near each other, but variance is low (the size of the cluster is "small", i.e. on a 2d plane the gaussian is tight). The 5th is further away from the first 4 but variance is high (this gaussian component is large on a 2d plane, the datapoints in it are spread out). Under this model, 3 points chosen in different kmeans selections may all be in the large gaussian component, because some points in the large gaussian component will be closer to the means of the other 4 components than the mean of itself. I decided to also include a picture, though it isn't very pretty.



   The circles represent gaussian mixture model components, the black dots are data points, the red dots are the cluster means that are shared between the gaussian mixture model and kmeans clustering. The brown dots represent the points chosen in this problem. They are all assigned to the same gaussian mixture class. However, each is assigned to a different k-means cluster because they are closer to other means.



(a)



(b)

3. **Bayesian Networks** (10 points) Do the following statements hold in each of the above networks ? Please explain your reasoning

   - $A \perp C | B, D$
   - $B \perp D | A, C$

   - $A \perp C | B, D$ in network (a). There are two paths from A to C, one through B and one through D. Both nodes B and D have arrows meet head-to-tail, and since B and D are in the given set, i.e. in the set C, A is marginally independent to C given B, D
   - $A \not\perp C | B, D$ in network (b). At nodes B and D, the arrows meet head-to-head, but both B and D are in the set C, so A and C are marginally dependent given B and D
   - $B \not\perp D | A, C$ in network (a). A meets tail-to-tail, and is in the set C, so this path is blocked, but C meets head-to-head, and is also in the set C, so this path isn't blocked. B and D are marginally dependent given A, C
   - $B \perp D | A, C$ in network (b). Both A and C are in set C and meet tail-to-tail on the paths from B to D, so both paths are blocked and B and D are marginally independent given A and C

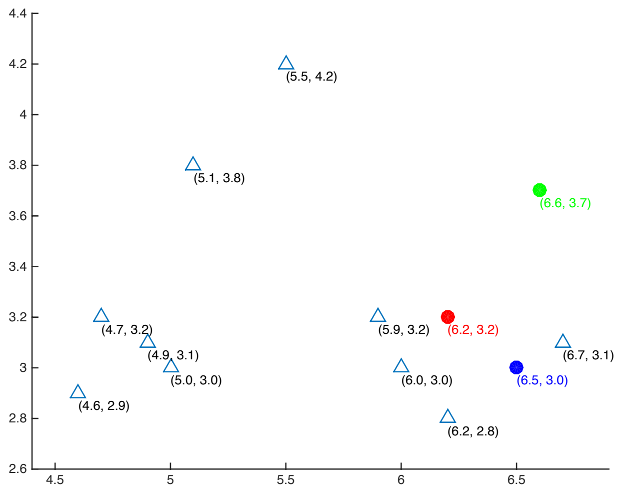


Figure 1: Scatter plot of datasets and the initialized centers of 3 clusters

4. **K-means** (30 points) Given the matrix X whose rows represent different data points, you are asked to perform a k-means clustering on this dataset using the Euclidean distance as the distance function. Here k is chosen as 3. The Euclidean distance d between a vector x and a vector y both in $\mathcal{R}^d$ is defined as $d(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^{d}(x_i - y_i)^2}$. All data in X were plotted in Figure 1. The centers of 3 clusters were initialized as $\mu_1 = (6.2, 3.2)(\text{red}), \mu_2 = (6.6, 3.7)(\text{green}), \mu_3 = (6.5, 3.0)(\text{blue})$.

   Implemented K-Means in a python script included in homework submission, so work is not shown here. Full output of clustering algorithm is in the file KMeans.txt in the program folder

   (a) What's the center of the first cluster (red) after one iteration? (Answer in the format of $[x_1, x_2]$, round your results to three decimal places)
   (5.171, 3.171)

   (b) What's the center of the second cluster (green) after two iteration?
   (5.3, 4)

   (c) What's the center of the third cluster (blue) when the clustering converges?
   (6.2, 3.025)

   (d) How many iterations are required for the clusters to converge?
   2. The third iteration just confirms the convergence.

5. **Expectation Maximization (EM)** (25 points) In this question you will implement the EM algorithm for Gaussian Mixture Models. A good read on gaussian mixture EM can be found at this link. A sample dataset for this problem can be downloaded in canvas files. For this problem:

   - n is the number of training points
   - f is the number of features
   - k is the number of gaussians
   - X is an $n \times f$ matrix of training data
   - w is an $n \times k$ matrix of membership weights. $w(i, j)$ is the probability that x_i was generated by gaussian j
   - π is a $k \times 1$ vector of mixture weights (gaussian prior probabilities). π_i is the prior probability that any point belongs to cluster i
   - μ is a $k \times f$ matrix containing the means of each gaussian
   - Σ is an $f \times f \times k$ tensor of covariance matrices. $\Sigma(:, :, i)$ is the covariance of gaussian i

   (a) **Expectation**: Complete the function $[w] = \text{Expectation}(X, k, \pi, \mu, \Sigma)$. This function takes in a set of parameters of a gaussian mixture model, and outputs the membership weights of each data point

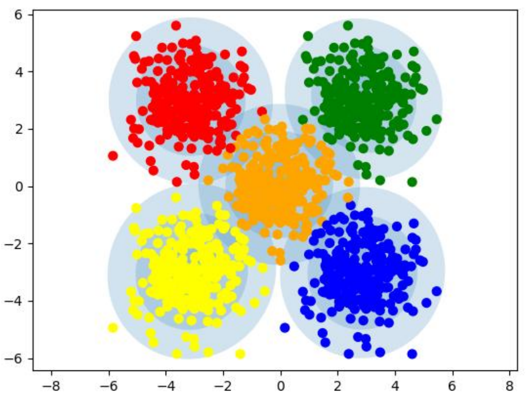
   (b) **Maximization of Means**: Complete the function $[\mu] = \text{MaximizeMean}(X, k, w)$. This function takes in the training data along with the membership weights, and calculates the new maximum likelihood mean for each gaussian.

   (c) **Maximization of Covariances**: Complete the function $[\Sigma] = \text{MaximizeCovariance}(X, k, w, \mu)$. This function takes in the training data along with membership weights and means for each gaussian, and calculates the new maximum likelihood covariance for each gaussian

   (d) **Maximization of Mixture Weights** : Complete the function $[\pi] = \text{MaximizeMixtures}(k, w)$. This function takes in the membership weights, and calculates the new maximum likelihood mixture weight for each gaussian.

   (e) **EM**: Put everything together and implement the function $[\pi, \mu, \Sigma] = \text{EM}(X, k, \pi_0, \mu_0, \Sigma_0, \text{nIter})$. This function runs the EM algorithm for nIter steps and returns the parameters of the underlying GMM. Note: Since this code will call your other functions, make sure that they are correct first. A good way to test your EM function offline is to check that the log likelihood, $\log P(X|\pi, \mu, \Sigma)$ is increasing for each iteration of EM.

   Below is a visualization of the resulting model generated by my program.

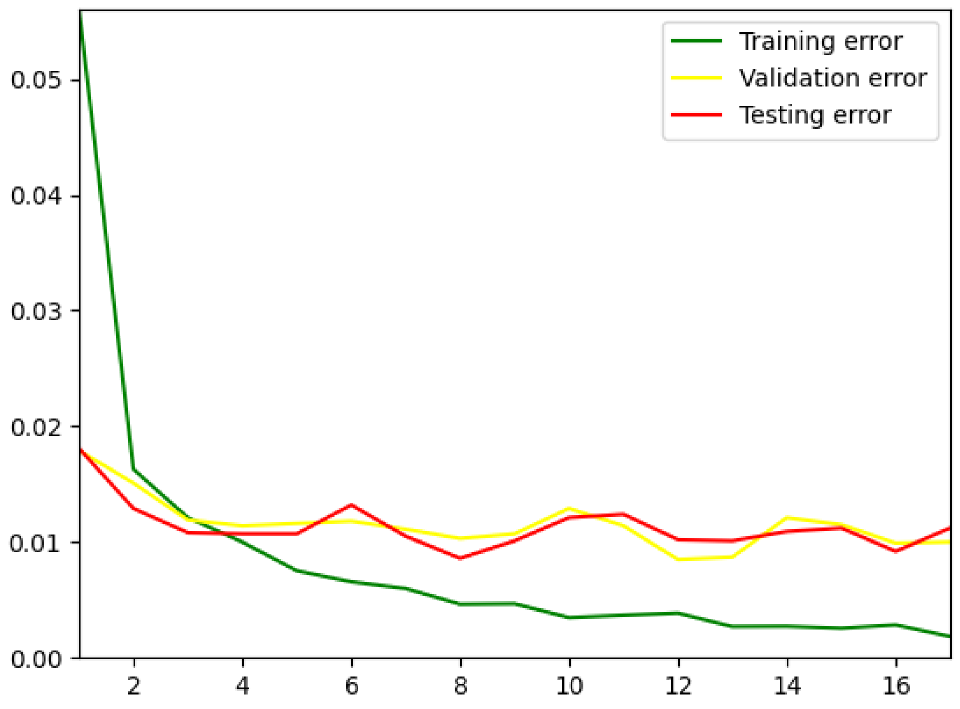


6. **Convolutional Neural Networks** (20 points) Develop a Convolutional Neural Network (CNN) model to predict a handwritten digit images into 0 to 9 (You can use Keras or other packages). The pickled file represents a tuple of 3 lists: the training set, the validation set and the testing set. Each of the three lists is a pair formed from a list of images and a list of class labels for each of the images. An image is represented as numpy 1-dimensional array of 784 (28 x 28) float values between 0 and 1 (0 stands for black, 1 for white). The labels are numbers between 0 and 9 indicating which digit the image represents. The code block below shows how to load the dataset.

```
import pickle, gzip, numpy

# Load the dataset
f = gzip.open('mnist.pkl.gz', 'rb')
u = pickle._Unpickler(f)
u.encoding = 'latin1'
train_set, valid_set, test_set = u.load()
f.close()
```

- Choose the proper activation and loss function.
  ReLU activation function, Sparse Categorical Cross Entropy loss function
- Plot the train, validation, and test errors as a function of the epochs.



- Report the best accuracy on the validation and test data sets. Discuss the parameter choices such as the filter size, number of filters etc.
  Note: The best accuracies are overall and include the extra epochs I run as part of my soft-early-stop strategy (explained below). Best epoch is the epoch chosen as the best model during training.
  Best Train Accuracy: 0.9981600046157837

Best Valid Accuracy: 0.9915000200271606

Best Test Accuracy: 0.9914000034332275

Best epoch: 12

I chose the filter sizes for the convolutional layers to go from 7 to 5 to 3, hoping that different sized feature pools would be obtained. However, pooling layers remained all at a size of 2, to halve the size of each dimension each time.

- Apply early stopping using the validation set to avoid overfitting.

  I applied a soft-early-stop algorithm. I ran epochs until there was no significant improvement to the validation accuracy after 4 further training epochs. Once no further improvement is detected, the best model is chosen as the last epoch that gave a significant improvement to the validation accuracy. Significant improvement is defined by comparing any improvement against a tolerance. Any improvement less than this tolerance is counted as a non-improvement epoch.

- Give a brief description of your observations.

  Initial few epochs of training result in the greatest improvement to accuracy, then we quickly encounter diminishing returns. Also, convolutional neural networks are very effective at this task. We quickly reach very small train errors, and validation and test errors settle around 1%.

- Does pooling make the model more or less sensitive to small changes in the input images? Why? By small changes, we mean moving the input images to the left or right, rotating them slightly etc.

  Less sensitive. Pooling causes small changes to be cobbled together into one node (a movement of one pixel will cause no change after a 2x2 max pooling operation).