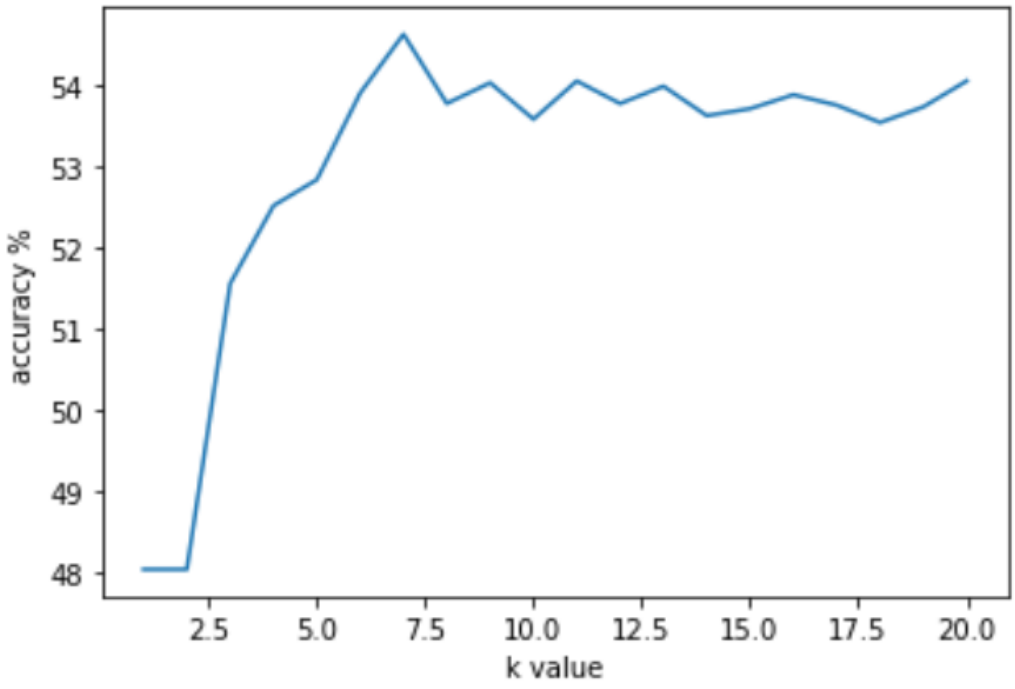
For **question 1**, I implemented K-Nearest Neighbors using this pseudocode from: “Slides adapted from Dr. Carlotta Domeniconi's Pattern Recognition at George Mason University.”

- Learning algorithm:
  - Store training examples
- Prediction algorithm: Look at the K most similar training examples (nearest neighbors).
  - Classification: assign the majority class label (majority voting) of these k neighbors
  - Regression: assign average response of these k neighbors

Results:




```
best k-value=7
with avg acc=54.617172
         getting the distances took 38.293190 seconds?
with k=7 testing set with random_state=0 had acc=0.575758
```

For **question 4**, I implemented K-Means using this pseudocode from: "Slides by Derek Hoiem and Kristen Grauman"

1. Randomly initialize the cluster centers, $c_1, \ldots, c_K$
2. Given cluster centers, determine points in each cluster

- For each point p, find the closest c_i. Put p into cluster i

3. Given points in each cluster, solve for c_i

- Set c_i to be the mean of points in cluster i

4. If c_i have changed, repeat Step 2

Results:

the answer to 4a is [5.171, 3.171]

the answer to 4b is [5.300, 4.000]

the answer to 4c is [6.200, 3.025]

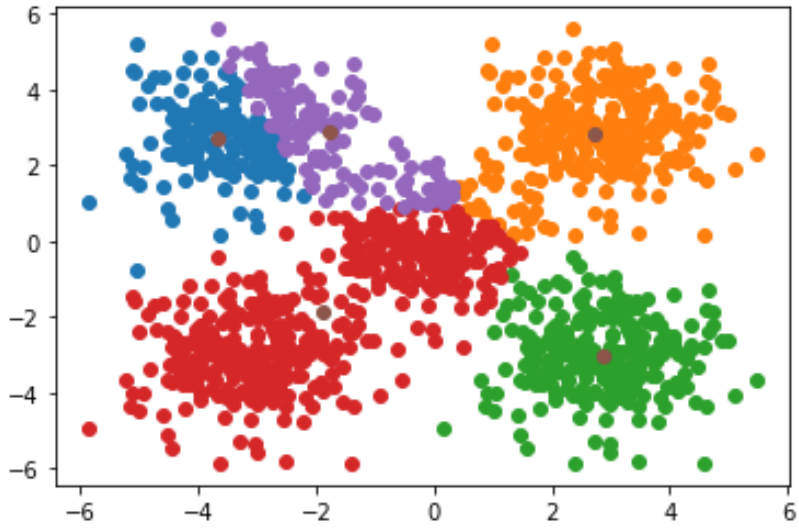
the answer to 4d is 3 iterations because there had to be an additional iteration to confirm convergence

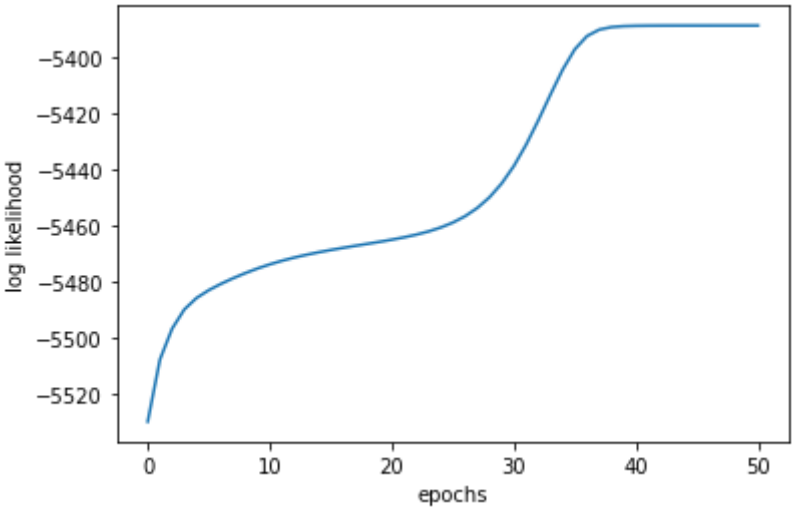
For **question 5**, I implemented mixture of gaussians using the expectation minimization algorithm; here is the pseudocode:

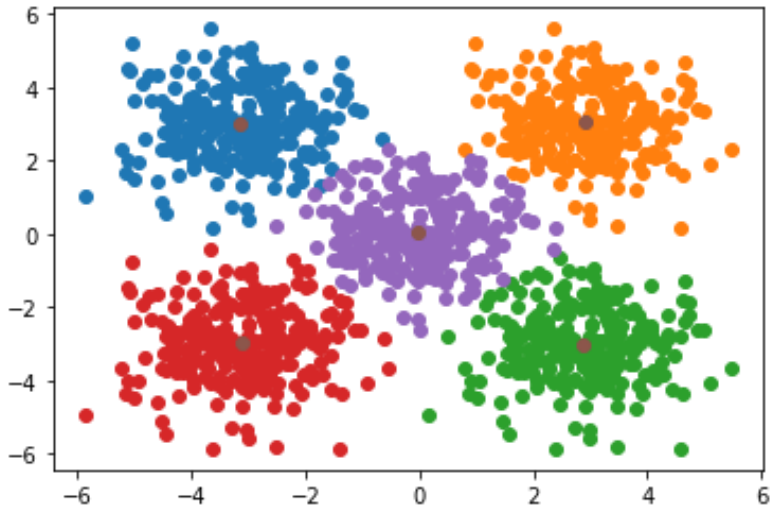
1. Set initial parameters.
2. For nIter iterations:
    a. Update membership weights with the E-Step function
    b. Update parameters with the first part of the M-Step functions
    c. Calculate and store the loglikelihood with the last M-Step function
3. Use the membership weights to update the gaussians.

Results:

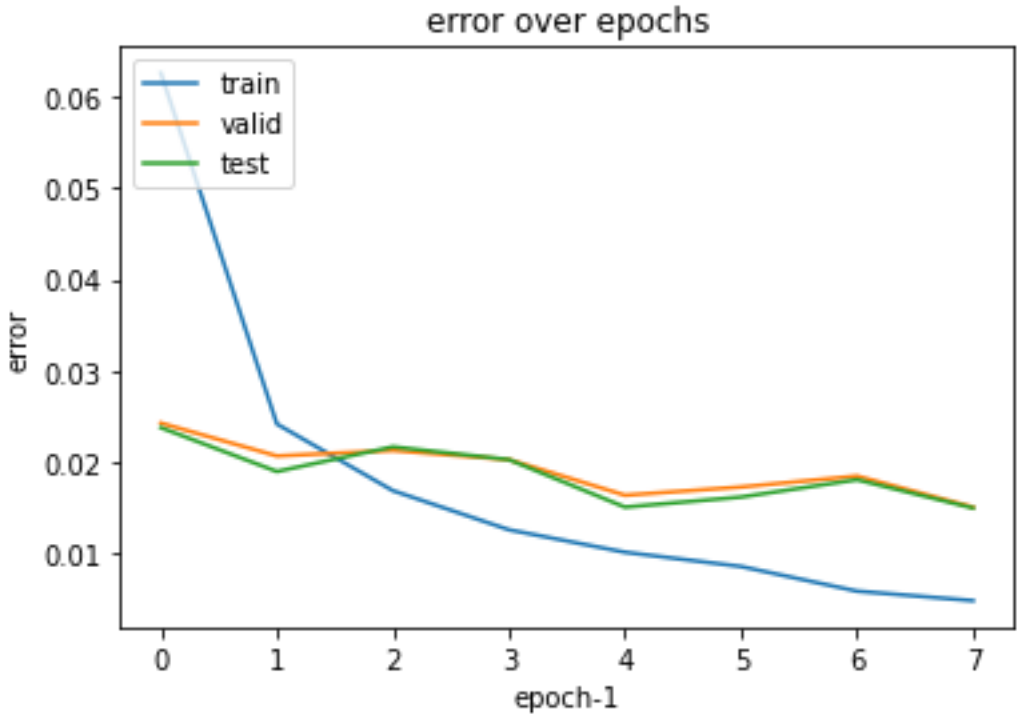
Using random.seed(11) And random_state=88


6
4
2
0
−2
−4
−6
−6
−4
−2
0
2
4
6


−5400
−5420
−5440
−5460
−5480
−5500
−5520
log likelihood
0
10
20
30
40
50
epochs


6
4
2
0
−2
−4
−6
−6
−4
−2
0
2
4
6

For **question 6**, I applied a Convolutional Neural Network on the mnist dataset with Keras.

- Choose the proper activation and loss function.
    - I chose relu as my activation function and crossentropy as my 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 Iter size, number of Iters etc.
    - best validation scores
    - 0.9848999977111816
    - best test scores
    - 0.9849999999999078
    - The lower the Iter size or number of Iters the greater the accuracy is because as mentioned later overfitting is an issue to apply a solution of this nature to a problem of this difficulty/complexity level.
- Apply early stopping using the validation set to avoid overfitting.
    - I set my tolerance equal to 3 epochs.
- Give a brief description of your observations.
    - Convolutional neural networks are incredibly powerful. Overfitting is a problem because after only a small number of epochs the model's accuracy is already 99% so any more training will do more harm than good.
- Does pooling make the model more or less sensitive to small changes in the input images? Why?
    - Minor rotations to the images did not cause a great change in accuracy with pooling.
    - Minor translations also did not cause a great change in accuracy with pooling.
    - The reason for this is: pooling reduces the sample feature maps by summarizing the presence of features in patches and because these changes don't create any similarity

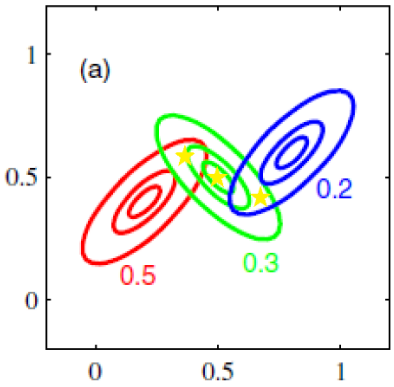
regarding the features of differently classified datapoints there won't be increased sensitivity.

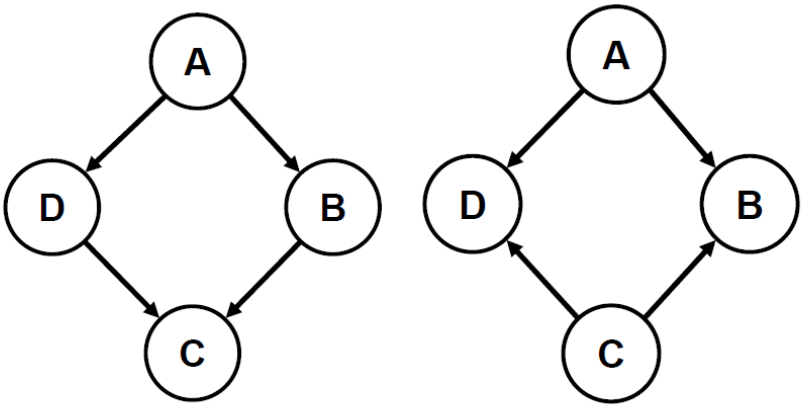
2. Clustering (5 points) Suppose we clustered a set of N data points using two different clustering algorithms: k-means and Gaussian mixtures. In both cases we obtained 5 clusters and in both cases the centers of the clusters are exactly the same. Can a few (say 3) points that are assigned to different clusters in the k-means 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, k-means uses a generative approach towards assignment whereas Gaussian mixture clustering gives soft (probabilistic) assignment to each data point.

Cluster centers can match for both, but if Gaussian mixture components have large variances/components are spread around their center, points on the edges between clusters may be given different assignments in the Gaussian mixture solution.

Consider the graph below. The 3 yellow stars all exist in the green cluster but 1 is also in the blue and another in the red cluster as well. Because of this overlap k means can place them in different clusters and they can still be in the same cluster by gaussian mixture.





(a) (b)

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

$A \perp C | B, D$

$B \perp D | A, C$

Look at all possible paths from A to B, any such path is said to be blocked (A and B are conditional independent given C) if it includes a node such that either

1. the arrows on the path meet either head-to-tail or tail-to-tail at the node, and the node is in the set C
2. the arrows meet head-to-head at the node, and neither the node, nor any of its descendants, is in the set C.

$A \perp C | B, D$

for graph (a) : Yes, they are cond. independent

$p(A \perp C|B) = \frac{p(A)p(B|A)p(C|B)}{p(B)} = \frac{p(A|B)p(C \perp B)}{p(B)} = p(A|B)p(C|B)$

$p(A \perp C|D) = p(A \perp C|D) = \frac{p(A)p(D|A)p(C|D)}{p(D)} = \frac{p(A|D)p(C \perp D)}{p(D)} = p(A|D)p(C|D)$

both nodes B and D follow the 1st and 2nd rules for D-separation,

as both are the central nodes in head-to-tail paths so all paths from A to C block.

$A \perp C | B, D$

for graph (b) : No, they are cond. dependent

$p(A \perp C|B) \propto p(B|A, C)p(A)p(C)$

$p(A \perp C|D) \propto p(D|A, C)p(A)p(C)$

for both node B and D the 1st and 2nd rules for D-separation are violated,

as both nodes are where the arrows meet in a head-to-head path.

$B \perp D | A, C$

for graph (a) : No, they are cond. dependent

$p(B \perp D|A) = \frac{p(B,D,A)}{P(A)} = \frac{p(B|A)p(D|A)P(A)}{P(A)} = p(B|A)p(D|A)$

$p(B \perp D|C) \propto p(C|B, D)p(B)p(D)$

The arrows meet at Node A in a tail-to-tail path. This means the path $_D \swarrow^A \searrow_B$ is blocked.

However the path $^D \searrow_C \swarrow^B$ does not satisfy as it is head-to-head this means that path is active. Because all paths between D and B must be blocked and we have found an active one they can't be cond. independent.

$B \perp D | A, C$

for graph (b) : Yes, they are cond. independent

$p(B \perp D|A) = \frac{p(B,D,A)}{P(A)} = \frac{p(B|A)p(D|A)P(A)}{P(A)} = p(B|A)p(D|A)$

$p(B \perp D|C) = \frac{p(B,D,C)}{P(C)} = \frac{p(B|C)p(D|C)P(C)}{P(C)} = p(B|C)p(D|C)$

The arrows meet at Node A in a tail-to-tail path. This means the path $_D \swarrow^A \searrow_B$ is blocked.

The arrows meet at Node C in a tail-to-tail path. This means the path $_D \swarrow^C \searrow_B$ is blocked.

All paths are blocked so it is D-separable and cond. independent.