MACHINE LEANING- WORKSHEET – 3

1. Question -1 - Answer - **D**
2. Question -2 - Answer - **D**
3. Question -3 - Answer - **C**
4. Question -4 - Answer - **B**
5. Question -5 - Answer **- D**
6. Question -6 - Answer - **C**
7. Question -7 - Answer - **D**
8. Question -8 - Answer - **A**
9. Question -9 - Answer – **D**
10. Question -10 - Answer - **B**
11. Question -11 - Answer – **A**
12. Question -12 - Answer – **B**

13. Benefits of Clustering :

Clustering Intelligence Servers provides the following benefits:

- Increased resource availability: If one Intelligence Server in a cluster fails, the other Intelligence Servers in the cluster can pick up the workload. This prevents the loss of valuable time and information if a server fails.

- Strategic resource usage: You can distribute projects across nodes in whatever configuration you prefer. This reduces overhead because not all machines need to be running all projects, and allows you to use your resources flexibly.

- Increased performance: Multiple machines provide greater processing power.
- Greater scalability: As your user base grows and report complexity increases, your resources can grow.

- Simplified management: Clustering simplifies the management of large or rapidly growing systems.

14. Clustering is an unsupervised machine learning methodology that aims to partition data into distinct groups, or clusters. There are a few different forms including hierarchical, density, and similarity based. Each has a few different algorithms associated with it as well. One of the hardest parts of any machine learning algorithm is feature engineering, which can especially be difficult with clustering as there is no easy way to figure out what best segments your data into separate but similar groups.

The guiding principle of similarity based clustering is that similar objects are within the same cluster and dissimilar objects are in different clusters. This is not different than the goal of most conventional clustering algorithms. With similarity based clustering, a measure must be given to determine how similar two objects are. This similarity measure is based off distance, and different distance metrics can be employed, but the similarity measure usually results in a value in [0,1] with 0 having no similarity and 1 being identical. To measure feature weight importance, we will have to use a weighted Euclidean distance function. The similarity measure is defined in the following:

β here is a value that we will actually have to solve for, (w) represents the distance weight matrix, and d represents the pair wise distances between all objects. To solve for β, we have to use the assumption that if using the standard weights (all 1's), our similarity matrix would uniformly distributed between [0,1] resulting in a mean of .5. So to find β, we solve the equation:

$$\frac{2}{n(n-1)} \sum_{j<i} \frac{1}{1+\beta * d_{ij}} = 0.5$$

If using a weighted euclidean distance, it is possible to use this similarity matrix to identify what features introduce more noise and which ones are important to clustering. The ultimate goal is to minimize the "fuzziness" of the similarity matrix, trying to move everything in the middle (ie .5) to either 1 or 0. For this purpose we use the loss metric:

$$E(w) = \frac{2}{N(N-1)} \sum_{q<p} \frac{1}{2} \left(\rho_{pq}^{(w)} \left(1 - \rho_{pq}^{(1)} \right) + \rho_{pq}^{(1)} \left(1 - \rho_{pq}^{(w)} \right) \right)$$

Here (1) represents the base weights (all 1's), and ρ represents the resulting fuzzy partition matrix that is a product of the weights used in the euclidean distance function between points p and q.

We can then attempt to use Gradient Descent on this loss function to try and minimize it with respect to the similarity matrix. Gradient Descent is one of the most common optimization algorithms in machine learning that is used to find best parameters of a given function by using the function gradient, a combination of the partial derivatives. By taking steps proportional to the negative of the gradient, we can try to find the local minimum of the function. We will continually update the weights until either our maximum number of iterations has been met, or the function converges. So the gradient descent will be of our loss function with a partial derivative in respect to the weights. We will update the weights every iteration with respect to the gradient and learning rate.

$$\triangle w_j = -\eta \frac{\partial E(w)}{\partial w_j},$$

Where n is the learning rate defined. n is a very important parameter, as something too small will require too much computation, while too big and the function may never converge.

If you can think of it in terms of a 3D graph, it would be like stretching or shrinking each axis, in a way that would put our points into tighter groups, that are further away from each other. We are not actually changing the locations of the data, we are solely transforming how we measure the distances that drive our similarity metrics.

Here is a created example where I introduce 3 clusters with separate centroids on the first two variables, but introduce a third noise variable that would make the clustering more difficult. These are colored by the actual cluster labels given when

the data is created. When eliminating the third noise variable, we can see it would be much easier to identify clusters.

Clustering is an unsupervised machine learning methodology that aims to partition data into distinct groups, or clusters. There are a few different forms including hierarchical, density, and similarity based. Each have a few different algorithms associated with it as well. One of the hardest parts of any machine learning algorithm is feature engineering, which can especially be difficult with clustering as there is no easy way to figure out what best segments your data into separate but similar groups.

The guiding principle of similarity based clustering is that similar objects are within the same cluster and dissimilar objects are in different clusters. This is not different than the goal of most conventional clustering algorithms. With similarity based clustering, a measure must be given to determine how similar two objects are. This similarity measure is based off distance, and different distance metrics can be employed, but the similarity measure usually results in a value in [0,1] with 0 having no similarity and 1 being identical. To measure feature weight importance, we will have to use a weighted euclidean distance function. The similarity measure is defined in the following:

$$\rho_{ij}^{(w)} = \frac{1}{1 + \beta * d_{ij}^{(w)}}$$

β here is a value that we will actually have to solve for, (w) represents the distance weight matrix, and d represents the pairwise distances between all objects. To solve for β, we have to use the assumption that if using the standard weights(all

1's), our similarity matrix would uniformly distributed between [0,1] resulting in a mean of .5. So to find β, we solve the equation:

$$\frac{2}{n(n-1)} \sum_{j<i} \frac{1}{1+\beta * d_{ij}} = 0.5$$

If using a weighted euclidean distance, it is possible to use this similarity matrix to identify what features introduce more noise and which ones are important to clustering. The ultimate goal is to minimize the “fuzziness” of the similarity matrix, trying to move everything in the middle (ie .5) to either 1 or 0. For this purpose we use the loss metric:

$$E(w) = \frac{2}{N(N-1)} \sum_{q<p} \frac{1}{2} \left(\rho_{pq}^{(w)} \left(1 - \rho_{pq}^{(1)} \right) + \rho_{pq}^{(1)} \left(1 - \rho_{pq}^{(w)} \right) \right)$$

Here (1) represents the base weights (all 1's), and ρ represents the resulting fuzzy partition matrix that is a product of the weights used in the euclidean distance function between points p and q.

We can then attempt to use Gradient Descent on this loss function to try and minimize it with respect to the similarity matrix. Gradient Descent is one of the most common optimization algorithms in machine learning that is used to find best parameters of a given function by using the function gradient, a combination of the partial derivatives. By taking steps proportional to the negative of the gradient, we can try to find the local minimum of the function. We will continually update the weights until either our maximum number of iterations has been met,

or the function converges. So the gradient descent will be of our loss function with a partial derivative in respect to the weights. We will update the weights every iteration with respect to the gradient and learning rate.

$$\triangle w_j = -\eta \frac{\partial E(w)}{\partial w_j},$$

Where n is the learning rate defined. n is a very important parameter, as something too small will require too much computation, while too big and the function may never converge.

If you can think of it in terms of a 3D graph, it would be like stretching or shrinking each axis, in a way that would put our points into tighter groups, that are further away from each other. We are not actually changing the locations of the data, we are solely transforming how we measure the distances that drive our similarity metrics.

Here is a created example where I introduce 3 clusters with separate centroids on the first two variables, but introduce a third noise variable that would make the clustering more difficult. These are colored by the actual cluster labels given when the data is created. When eliminating the third noise variable, we can see it would be much easier to identify clusters.