MACHINE LEARNING HOMEWORK 02

BY OURU AUGUSTINE

March 13, 2024

1 Qn 1) The table below provides a training data set containing six observations, three predictors, and one qualitative response variable.

Obs	X1	X2	X3	Y	d
1	0	3	0	Red	3
2	2	0	0	Red	2
3	0	1	3	Red	3.1
4	0	1	2	Green	2.2
5	-1	0	1	Green	1.4
6	1	1	1	Red	1.7

Suppose we wish to use this data set to make a prediction for Y when X1=X2=X3=0 using K-nearest neighbors.

1.1 a)Compute the Euclidean distance between each observation and the test point, X1=X2=X3=0.

ANSWER: For Obs1(Red): $d = \sqrt{(0-0)^2 + (3-0)^2 + (0-0)^2} = 3$ In the same way

For Obs2(Red):$d = \sqrt{2^2 + 0^2 + 0^2} = 2$,Obs3(Red):$d = \sqrt{0^2 + 1^2 + 3^2} = 3.1622$, Obs4(Green):$d = \sqrt{1^2 + 2^2} = 2.236$, Obs5(Green) :$d = \sqrt{(-1)^2 + 0^2 + 1^2} = 1.4142$,Obs6(Red): $d = \sqrt{1^2 + 1^2 + 1^2}$ =1.732

1.2 b)What is our prediction with K = 1? Why?

With K = 1 (one neighbor), we consider the data point with the shortest distance to the test point. Here, the closest point is (-1, 0, 1, Green) with a distance of 1.4141. Therefore, our prediction for Y with K = 1 is Green. This is because the closest neighbor has the label "Green."

1.3 c)What is our prediction with K = 3? Why?

With K = 3 (three neighbors), we consider the three data points with the shortest distances:

Obs5(Green) with d=1.4

Obs6(Red) with d= 1.7

Obs2(Red) with d= 2

Now, we look at the majority class among these three neighbors. Where Red has the highest count

Therefore, our prediction for Y with K = 3 is Red. In this case, increasing K from 1 to 3 does change the prediction

1.4 d)If the Bayes decision boundary in this problem is highly non-linear, then would we expect the best value for K to be large or small? Why?

If the Bayes decision boundary (the line separating the classes) in this problem is highly non-linear, then a smaller value of K might be preferable. Here's why:

● Smaller K: With a smaller K, we focus on the closest neighbors. In a non-linear scenario, these neighbors are more likely to be on the same side of the true decision boundary as the test point.

● Larger K: With a larger K, we consider more distant neighbors. In a non-linear case, these distant neighbors could be on the opposite side of the decision boundary, potentially leading to a less accurate prediction. However, the optimal value of K depends on the specific dataset and the complexity of the decision boundary. It's often recommended to try different K values and choose the one that leads to the best performance on a validation set

2 Qn2)Carefully explain the differences between the KNN classifier and the KNN regression methods.

Both K-Nearest Neighbors (KNN) classifier and KNN regression are machine learning algorithms that utilize the concept of "similarity" to make predictions. However, they differ in their target variable and the nature of the prediction:

KNN Classifier:

● Target Variable: Categorical (discrete)

● Prediction: Class label membership In KNN classification, the algorithm predicts the class label (e.g., Red, Green) for a new data point based on the majority vote of its K nearest neighbors in the training data.

● Steps:

1. Calculate the distance (usually Euclidean) between the new data point and all points in the training set.

2. Find the K closest neighbors based on the calculated distances.

3. Assign the new data point the most frequent class label among its K nearest neighbors

KNN Regression:

● Target Variable: Continuous (numerical)

● Prediction: Numerical value In KNN regression, the algorithm predicts the numerical value for a new data point based on the average (or other aggregation methods) of the values of its K nearest neighbors in the training data.

● Steps: 1.

Calculate the distance between the new data point and all points in the training set.

2. Find the K closest neighbors based on the calculated distances.

3. Predict the value for the new data point by averaging (or using other methods like weighted average) the values of its K nearest neighbors.

Key Differences:

Feature	KNN Classifier	KNN Re
Target Variable	Categorical (Class labels)	Continuous
Prediction	Class label membership	Numeric
Aggregation Method	Majority vote for the most frequent class label	Average (or other methods) of t
Decision Boundary	Learns a discrete boundary between classes	Doesn't explicitly learn a boun

Example: Imagine predicting the type of fruit (apple, orange) based on its size and weight (KNN classifier) or predicting the price of a house based on its size and location (KNN regression).

In conclusion:

● KNN classifiers are useful for predicting discrete categories, while KNN regression is suitable for predicting continuous values.

● They both leverage the "wisdom of the crowd" principle by looking at the closest neighbors to make predictions, but the way they aggregate information from neighbors differs based on the nature of the target variable.

3 Qn3)We cover the SNE, symmetric SNE and t-SNE in the class.

3.1 a) What does the SNE algorithm try to minimize?

Answer:

The SNE(Stochastic Neighbor Embedding) algorithm aims to minimize the divergence between the probability distributions of pairwise similarities in the high-dimensional space and the pairwise similarities in the low-dimensional space.

Specifically, it tries to preserve the local structure of data points by minimizing the Kullback-Leibler divergence between the high-dimensional similarity matrix (P) and the low-dimensional similarity matrix (Q).

3.2 (b) Calculate the gradient of the SNE cost function with respect to y_i.

The gradient of the SNE cost function with respect to the low-dimensional embedding coordinates (y_i) is given by:

$\frac{\partial C}{\partial y_i} = 2\sum_j \Big((p_{j|i} - q_{j|i}) + (p_{i|j} - q_{i|j})\Big)(y_i - y_j)$

where:

$(p_{j|i})$ represents the probability that data point (x_j) is a neighbor of (x_i) in the high-dimensional space.

$(q_{j|i})$ represents the probability that (y_j) is a neighbor of (y_i) in the low-dimensional space.

The summation is over all data points except (i).

3.3 (c) Calculate the gradient of the symmetric SNE cost function with respect to y_i.

In symmetric SNE, the cost function is symmetrized, resulting in a simpler gradient.

The gradient of the symmetric SNE cost function is similar to the SNE gradient but without the asymmetry in probabilities:

$\frac{\partial C}{\partial y_i} = 4\sum_j (p_{j|i} - q_{j|i})(y_i - y_j)$

3.4 (d) Calculate the gradient of the t-SNE cost function with respect to y_i

t-SNE builds upon SNE by addressing the crowding problem and improving visualization.

It uses a Student-t distribution instead of a Gaussian to compute similarities in the low-dimensional space.

The gradient of the t-SNE cost function is given by: $\frac{\partial C}{\partial y_i} = 4\sum_j (p_{j|i} - q_{j|i})(y_i - y_j)(1 + |y_i - y_j|^2)^{-1}$

4 Qn 4) Consider the fitted values that result from performing linear regression without an intercept. In this setting, the i-th fitted value takes the form [$\hat{y}_i = x_i\hat{\beta}$,] where [$\hat{\beta} = (\Sigma_{i=1}^{n} x_i y_i) / (\Sigma_{i'=1}^{n} x_{i'}^2)$.] Show that we can write [$\hat{y}i = \Sigma\, i' = 1^n a_{i'} y_{i'}$.] What is ($a_{i'}$)?

Note: We interpret this result by saying that the fitted values from linear regression are linear combinations of the response values.

ANSWER

Breaking down the expression for the fitted values in linear regression without an intercept.

Given the linear regression model without an intercept: $\hat{y}_i = x_i\hat{\beta}$

where:

($\hat{y}_i$) represents the i-th fitted value. (x_i) is the predictor variable for the i-th observation.

($\hat{\beta}$) is the estimated coefficient.

We have the expression for ($\hat{\beta}$): $\hat{\beta} = \frac{\sum_{i=1}^{n} x_i y_i}{\sum_{i'=1}^{n} x_{i'}^2}$

Now, let's express the fitted value ($\hat{y}i$) in terms of the response values (yi'):

[$\hat{y}_i = x_i\hat{\beta}$]

Substitute the expression for ($\hat{\beta}$): $\hat{y}i = x_i \left(\frac{\sum i=1^n x_i y_i}{\sum_{i'=1}^{n} x_{i'}^2} \right)$

Rearrange the terms: [$\hat{y}i = \frac{\sum i=1^n x_i y_i}{\sum_{i'=1}^{n} x_{i'}^2} \cdot x_i$]

Now, let's express the numerator as a summation: $\hat{y}i = \frac{\sum i'=1^n a_{i'} y_{i'}}{\sum_{i'=1}^{n} x_{i'}^2} \cdot x_i$

where: $a_{i'} = \frac{x_{i'}}{\sum_{i'=1}^{n} x_{i'}^2}$

Therefore, we can write: $\hat{y}i = \sum i' = 1^n a_{i'} y_{i'}$

In summary, the fitted values from linear regression without an intercept are indeed linear combinations of the response values, with the coefficients ($a_{i'}$) given by the expression above.

These coefficients depend on the predictor variables and their squared sums.

5 Qn 5 The last question is about the gradient descent of logistic regression. In the gitlab repository, there is a folder code including the gradient descent algorithm for linear regression. The task here is to write the gradient descent algorithm for the logistic regression. Requirement: the function is called LogisticRegression and is implemented in a class with the following structure.