Winter Session (2024-2025) Machine Learning

Day 1: Main notes

1. Linear Regression

a) Where can you use linear Regression in training data ?

The dependant variable (target) should have a linear relationship with the independant variables (features)

This linear relationship can be verified:

- Visualization : Plot the features against the target to check if the relationship appears linear.

- Correlation: Use "Pearson Correlation" to see if features are strongly linearly correlated with the target.

b) How the relationship between the features and the target values are represented ?

The linear relationship is represented by the following equation:

$$y = w_1 x + w_0$$

where:

y : target value

x : feature

w_0 : intercept

w_1 : slope

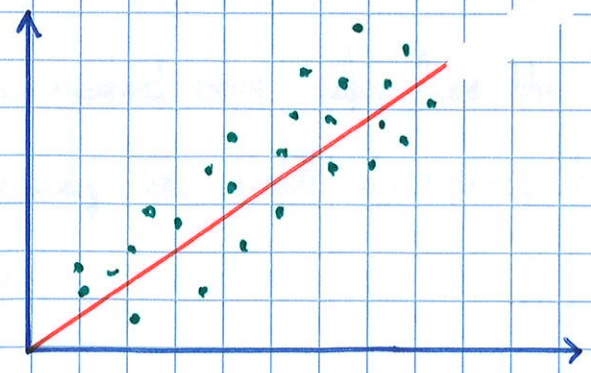
c) What we aim to learn ?

We aim to predict w_0 and w_1 using the data that we have.

"w" in on represent the weight; since it represent the effect of feature on the target value.

Remark : "w" represents the influence of the feature (x) on the target value (y). By adjusting the weight we can control how strongly (x) impacts (y)

d) How the linear regression can be visualized ?



- Scatter point (data or trained data)
- line representing relationship bet. x and y

How the training of data performs to predict the target value ?

- It works on adjusting (w_0) and (w_1) ; and the process continues until the loss is minimized (which is the difference between the predicted value and the true value).

f) What are the loss used when dealing with linear regression ?

A common loss function is " Mean Square Error (MSE)"

$$l(w_0; w_1) = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

Also ; it can be written as follows:

$$l(w_0; w_1) = \sum_{i=1}^{n} (y_i - (w_1 x_i + w_0))^2$$

Minimizing the squared error helps find the line that best fits the data by reducing the overall discrepancy between predicted and and actual values

g) How the squared error loss function can be theoretically minimized ?

It can be theoretically minimized when the partial derivatives with respect to w_0 and w_1 are zero.

$$\frac{\partial}{\partial w_0} \sum_{i=1}^{N} (y_i - (w_1 x_i + w_0))^2 = 0$$

$$\frac{\partial}{\partial w_1} \sum_{i=1}^{N} (y_i - (w_1 x_i + w_0))^2 = 0$$

- This has unique solution:

$$w_0 = \frac{1}{N} \sum_{i=1}^{N} y_i - \frac{w_1}{N} \sum_{i=1}^{N} x_i$$

$$w_1 = \frac{\frac{1}{N} \left(\sum_{i=1}^{N} x_i y_i \right) - \bar{x}\bar{y}}{\frac{1}{N} \left(\sum_{i=1}^{N} x_i^2 \right) - \bar{x}^2}$$

Remark: When building a machine learning model, the goal is to teach the model to recognize patterns in the data so that it can make accurate predictions on new, unseen data.

achieve this ; we divide the data into two parts :

Training set : The model learns patterns, relationships or trends from this data

Testing set : This is the portion of the data reserved to evaluate the model.

b) Why we are using Machine Learning to find w_0 and w_1 ? Why we don't solve the partial derivative equations when they are equal to zero to find optimal w_0 and w_1 instead ?

In more complex cases, finding the minimum by solving partial derivative in either :

- Mathematically intractable : the equations might not have a closed form equations.
- Computationally expensive : The number of variables or the complexity of the functions makes solving analytically unreasonable.

Explanation of Gradient Descent ? (to find w_0 and w_1)

Step 0 : Initialize starting points (w_0 and w_1)

- Randomly pick initial values for the weights
- These values might not be mean the optimal solution ; but they provide starting points

Step 1: Compute the Gradient descent or an estimate of the Gradient descent

For the univariate case:

- $$\frac{\partial}{\partial w_0} \sum_{i=1}^{n} (y_i - (w_1 x_i + w_0))^2 = -2 \sum_{i=1}^{n} (y_i - (w_1 x_i + w_0))$$
- $$\frac{\partial}{\partial w_1} \sum_{i=1}^{n} (y_i - (w_1 x_i + w_0))^2 = -2 \sum_{i=1}^{n} (y_i - (w_1 x_i + w_0)) x_i$$

Step 2: Update the weights w_0 ; w_1

The Gradient descent is to find the values of w_0 and w_1 that minimizes the loss function.

Minimizing the loss means reducing the error between the predicted values and the actual values.

The sign of the Gradient (positive or negative) tells us whether to increase or decrease the weight.

- if the gradient is positive; the weight should decrease to reduce the loss.
- if the gradient is negative; the weight should increase to reduce the loss.

Magnitude of the change:

- A large gradient means the current weight is far from the optimal value; so it requires a large update
- A small gradient means the weight is close to the optimal value, so only adjustment is needed.

How the gradient is updated ?

The weight is updated by updating the learning rate (η)

$$w_0 \leftarrow w_0 - \eta \frac{\partial L}{\partial w_0}$$

$$w_1 \leftarrow w_1 - \eta \frac{\partial L}{\partial w_1}$$

Step 3: Repeat Step 2 until the loss function will become within the accepted threshold

2. Linear Classification

Linear Classification is when a line acts as a boundary between two different categories.

A decision Boundary is defined as below:

- Linearly separable data can be divided using a linear decision boundary:

$$x_2 - (w_1 x_1 + w_0) = 0$$

- The points, which we want to classify with value 1, are above this line. They are points for which:

$$x_2 - (w_1 x_1 + w_0) > 0$$

- while the points which we want to classify with value 0 are

$$x_2 - (w_1 x_1 + w_0) < 0$$

Remark: In the example that we are dealing with the goal is to find a relationship between the features.

To do this, we can deal with "Stochastic Gradient Descent".

The Stochastic Gradient Descent can learn the full relationship in a dataset, even when processing small batches. The key idea is that in that SGD uses random samples to approximate the full gradient.

Remark: The advantage of Stochastic Gradient Descent is that smaller batches allow faster updates which can speed up learning.

Let's explain the steps of Stochastic Gradient Descent:

1. start with initial weights
2. divide the data into small batches of equal sizes
3. Iterative Process:
   a. Select a small batch
   b. Compute predictions

c. Calculate loss / error

d. Compute Gradient

e. Update weight ; meaning update learning rate

f. Repeat for the next batch. The updated weights are used for the next batch in the iteration.

Why I am assuming on the magnitude of error is large in the positive direction; I need to select w in the opposite direction to decrease it ?

For a linear regression model;

$$\hat{y} = w_1 x_1 + w_0$$

as $w_1 \uparrow$ $\hat{y} \uparrow \rightarrow$ error $\uparrow$

$\leftarrow$

this is the reason behind decreasing w_1 when updating it when it has high magnitude in the $\rightarrow$ direction.

Returning back to the linear classification; there is a specific function you can use it for SDG.

Which variable should be response with rest being features ?

To be able to answer this ; you should conduct some tests :

Correlation tests :

- Pearson Correlation (for continious variables)
  As you know ; it measures the relationship (linear) between two variables

- Spearman Rank Correlation (for non linear data)
  it measures monotonic relationship between variables

- Chi - Square test
  Tests the independance between two categorical variables

Thus ; these tests can be used to be able to answer the features that affect the target value.

3. K_Nearest Neighbors (KNN) regression

K_Nearest Neighbors (KNN) regression is a simple machine learning algorithm used to predict a target variable based on K nearest neighbors in the dataset.

a) What are the steps needed to implement the K-Nearest Neighbors (KNN) regression ?

Step 1: Calculate distance between the target value and each data in the dataset

KNN uses a distance (e.g., Euclidean distance) to find closest neighbors. The formula for Euclidean distance is:

$$\text{Distance} = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2}$$

Step 2: Choose the Closest Neighbors based on the distance

Step 3: Predict the target value

The target value can be predicted by setting or determining K and the metric distance.

1. How many neighbors (K) should we use for regression?
2. Of the K nearest neighbors, should ones further away be treated the same as those nearby?

   uniform: all neighbors equal
   distance: weight neighbors by inverse of distance from data point

3. How do we measure distance between data points to calculate "nearness"? Let d be a vector differences between a test sample and training sample, one element for each feature:

   l_1: L1 norm (Manhattan distance) $\sum_{i=1}^{n} |d_i|$

   l_2: L2 norm (Euclidean distance) $\sqrt{\sum_{i=1}^{n} d_i^2}$

Step 4: Validate your choices

- Evaluate the model's performance (e.g using Square Mean Error or R^2) after trying different combinations of these parameters.
- Use a validation set or cross-validation to test how well the model generalizes to new data.

What is R^2 test and how can you use it to evaluate the model ?

The coefficient of determination is calculated using the following formula :

$$R^2 = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y})^2}$$

where :

- y_i : True target value
- $\hat{y}_i$: Predicted value from the model
- $\bar{y}$: Mean of the true target values

How can you interpret the result of R^2 ?

- if $R^2 = 1$:

it means that it is a perfect prediction ; meaning that the residual error which is :

$\sum_i (y_i - \hat{y})^2$ are zero ;

meaning predictions perfectly match true values.

- if $R^2 < 0$; it means that the model performs worse than the baseline model (predicting the mean).

- if $R^2 = 0$; it means that the model does no better than simply predicting the mean of the target variable $\bar{y}$ for all inputs

4. Explanation of Feature Selection:

The values of the features should be correlated with the response value.

Including many formative features makes our model more complex, less interpretable ; more computationally costly to use and maintain.

For this purpose ; we should conduct some correlation tests to assess the relationship between the features and the target value.

- Features that are strongly correlated with the response is useful for prediction

- Features that are weakly correlated with the response is useless for predictions and should be removed.

5. Feature Normalization

Feature Normalization is fix the features such that they are scaled similarly ; with similar distribution and ranges.

a) Why feature Normalization is important ?

Feature in a dataset can have very different range of values.
In KNN ; distance calculations rely on all features values where features with larger ranges will dominate the distance metric.
The imbalance causes the model to prioritize some features over others ; even if those features are not necessarily more important.

b) What could be the solution to solve this problem ?

Normalization rescales feature values so they have similar ranges and distributions ; ensuring that :

- No single feature disproportionality influences distance calculations.
- The model uses all the data effectively.

Normalization is commonly done using one of the following methods :

Min Max Scaling :

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}}$$

Z-score Standarization:

$$X' = \frac{X - \mu}{\sigma}$$

with

μ : Mean of the feature

σ : Standard deviation of the feature.

Remark: the Z-score standarization is less sensitive to outliers since it centers data based on the mean. It works well with unbounded features.

6. K-Nearest Neighbors Classification

KNN classification is a machine learning method used to assign labels (or classes) to data points based on their similarity to other points.

a) What is KNN classification specifically ?

Let's consider the following example:

- The wine dataset contains scores that represents the quality of the wine ; typically on scale (0 to 10)
- Instead of predicting exact score, we want to classify wines into broader categories such as :

  poor : from 0 to 3

  average : from 4 to 6

  excellent : from 7 to 10

b) What is the reason behind this classification ?

- Simplification
- Practicality