DATA SCIENCE & MACHINE LEARNING (part - 2)

techworldthink • March 06, 2022

4. Explain how to choose the value of k in k-NN algorithm.

K-Nearest Neighbors is the supervised machine learning algorithm used for classification and regression. It manipulates the training data and classifies the new test data based on distance metrics. It finds the k-nearest neighbors to the test data, and then classification is performed by the majority of class labels.

Selecting the optimal K value to achieve the maximum accuracy of the model is always challenging for a data scientist.

- There are no pre-defined statistical methods to find the most favorable value of K.
- Initialize a random K value and start computing.
- Choosing a small value of K leads to unstable decision boundaries.
- The substantial K value is better for classification as it leads to smoothening the decision boundaries.
- Derive a plot between error rate and K denoting values in a defined range. Then choose the K value as having a minimum error rate.

In KNN, finding the value of k is not easy. A small value of k means that noise will have a higher influence on the result and a large value make it computationally expensive.

There is no straightforward method to calculate the value of K in KNN. You have to play around with different values to choose the optimal value of K. Choosing a right value of K is a process called Hyperparameter Tuning.

The value of optimum K totally depends on the dataset that you are using. The best value of K for KNN is highly data-dependent. In different scenarios, the optimum K may vary. It is more or less hit and trail method.

You need to maintain a balance while choosing the value of K in KNN. K should not be too small or too large. A small value of K means that noise will have a higher influence on the result.

Larger the value of K, higher is the accuracy. If K is too large, you are under-fitting your model. In this case, the error will go up again. So, at the same time you also need to prevent your model from under-fitting. Your model should retain generalization capabilities otherwise there are fair chances that your model may perform well in the training data but drastically fail in the real data. Larger K will also increase the computational expense of the algorithm.

There is no one proper method of estimation of K value in KNN. No method is the rule of thumb but you should try considering following suggestions:

1. Square Root Method: Take square root of the number of samples in the training dataset.

2. Cross Validation Method: We should also use cross validation to find out the optimal value of K in KNN. Start with K=1, run cross validation (5 to 10 fold), measure the accuracy and keep repeating till the results become consistent.

K=1, 2, 3... As K increases, the error usually goes down, then stabilizes, and then raises again. Pick the optimum K at the beginning of the stable zone. This is also called **Elbow Method.**

3. Domain Knowledge also plays a vital role while choosing the optimum value of K.

4. K should be an **odd number**.

5. Explain entropy and information gain.

8.5 Entropy

The degree to which a subset of examples contains only a single class is known as *purity*, and any subset composed of only a single class is called a *pure* class. Informally, entropy[3] is a measure of "impurity" in a dataset. Sets with high entropy are very diverse and provide little information about other items that may also belong in the set, as there is no apparent commonality.

Entropy is measured in bits. If there are only two possible classes, entropy values can range from 0 to 1. For n classes, entropy ranges from 0 to $\log_2(n)$. In each case, the minimum value indicates that the sample is completely homogeneous, while the maximum value indicates that the data are as diverse as possible, and no group has even a small plurality.

8.5.1 Definition

Consider a segment S of a dataset having c number of class labels. Let p_i be the proportion of examples in S having the i th class label. The entropy of S is defined as

$$\text{Entropy}\,(S) = \sum_{i=1}^{c} -p_i \log_2(p_i).$$

8.5.2 Examples

Let "xxx" be some class label. We denote by p_{xxx} the proportion of examples with class label "xxx".

1. **Entropy of data in Table 8.1**

   Let S be the data in Table 8.1. The class labels are "amphi", "bird", "fish", "mammal" and "reptile". In S we have the following numbers.

Number of examples with class label "amphi"	= 3
Number of examples with class label "bird"	= 2
Number of examples with class label "fish"	= 2
Number of examples with class label "mammal"	= 2
Number of examples with class label "reptile"	= 1
Total number of examples	= 10

   Therefore, we have:

$$\text{Entropy}\,(S) = \sum_{\text{for all classes "xxx"}} -p_{\text{xxx}} \log_2(p_{\text{xxx}})$$

[4]Plot created using R language.

$$\begin{aligned}
&= -p_{\text{amphi}} \log_2(p_{\text{amphi}}) - p_{\text{bird}} \log_2(p_{\text{bird}}) \\
&\quad - p_{\text{fish}} \log_2(p_{\text{fish}}) - p_{\text{mammal}} \log_2(p_{\text{mammal}}) \\
&\quad - p_{\text{reptile}} \log_2(p_{\text{reptile}}) \\
&= -(3/10)\log_2(3/10) - (2/10)\log_2(2/10) \\
&\quad - (2/10)\log_2(2/10) - (2/10)\log_2(2/10) \\
&\quad - (1/10)\log_2(1/10) \\
&= 2.2464
\end{aligned}$$

Information Gain:

- Information gain is the measurement of changes in entropy after the segmentation of a dataset based on an attribute.
- It calculates how much information a feature provides us about a class.
- According to the value of information gain, we split the node and build the decision tree.
- A decision tree algorithm always tries to maximize the value of information gain, and a node/attribute having the highest information gain is split first. It can be calculated using the below formula:

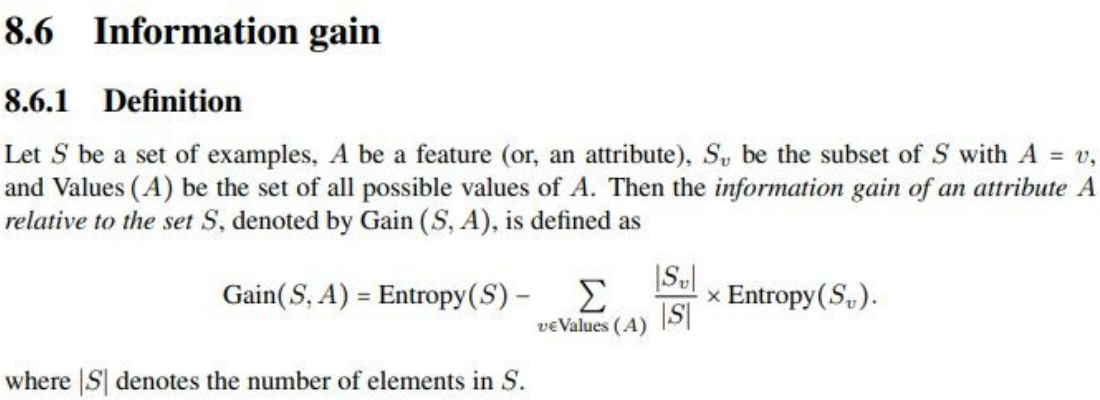
Information Gain= Entropy(S)- [(Weighted Avg) *Entropy(each feature)

Entropy: Entropy is a metric to measure the impurity in a given attribute. It specifies randomness in data. Entropy can be calculated as:

```
Entropy(s)= -P(yes)log2 P(yes)- P(no) log2 P(no)
```

Where,

- S= Total number of samples
- P(yes)= probability of yes
- P(no)= probability of no



8.6 Information gain

8.6.1 Definition

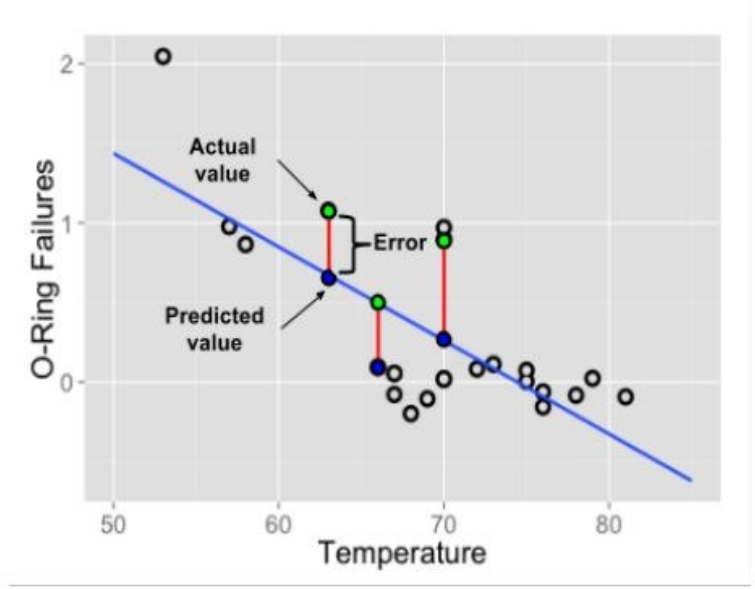
Let S be a set of examples, A be a feature (or, an attribute), S_v be the subset of S with $A = v$, and Values (A) be the set of all possible values of A. Then the *information gain of an attribute A relative to the set S*, denoted by Gain (S, A), is defined as

$$\text{Gain}(S, A) = \text{Entropy}(S) - \sum_{v \in \text{Values}(A)} \frac{|S_v|}{|S|} \times \text{Entropy}(S_v).$$

where $|S|$ denotes the number of elements in S.

6. Explain the Ordinary Least Square method in regression.

In order to determine the optimal estimates of α and β, an estimation method known as ordinary least squares (OLS) was used. In OLS regression, the slope and intercept are chosen such that they minimize the sum of the squared errors, that is, the vertical distance between the predicted y value and the actual y value. These errors are known as residuals,



In mathematical terms, the goal of OLS regression can be expressed as the task of minimizing the following equation:

$$\sum (y_i - \hat{y}_i)^2 = \sum e_i^2$$

In plain language, this equation defines e (the error) as the difference between the actual y value and the predicted y value. The error values are squared and summed across all points in the data.

The method of **least squares** is a standard approach in regression analysis to approximate the solution of overdetermined systems (sets of equations in which there are more equations than unknowns) by minimizing the sum of the squares of the residuals (a residual being: the difference between an observed value, and the fitted value provided by a model) made in the results of each individual equation.

7. Define activation function. Give two examples.

In an artificial neural network, the function which takes the incoming signals as input and produces the output signal is known as the activation function.

Simply put, an activation function is **a function that is added into an artificial neural network in order to help the network learn complex patterns in the data**. When comparing with a neuron-based model that is in our brains, the activation function is at the end deciding what is to be fired to the next neuron.

1. Threshold activation function

The *threshold activation function* is defined by

$$f(x) = \begin{cases} 1 & \text{if } x > 0 \\ -1 & \text{if } x \leq 0 \end{cases}$$

The graph of this function is shown in Figure 9.5.

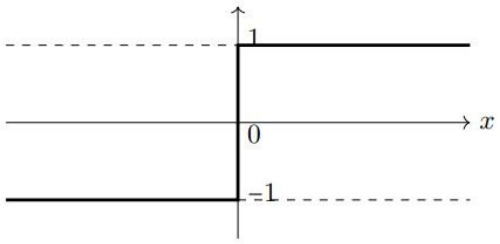


Figure 9.5: Threshold activation function

2. Unit step functions

Sometimes, the threshold activation function is also defined as a unit step function in which case it is called a *unit-step activation function*. This is defined as follows:

$$f(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases}$$

The graph of this function is shown in Figure 9.6.

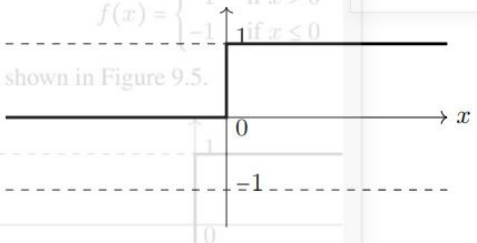


Figure 9.6: Unit step activation function

3. Sigmoid activation function (logistic function)

One of the must commonly used activation functions is the sigmoid activation function. It is defined as follows:

$$f(x) = \frac{1}{1+e^{-x}}$$

The graph of the function is shown in Figure 9.7.

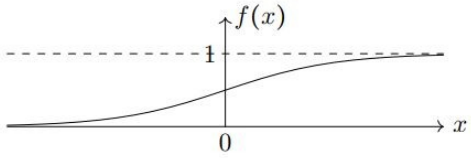


Figure 9.7: The sigmoid activation function

Sigmoid function is known as the logistic function which helps to normalize the output of any input in the range between 0 to 1. The main purpose of the activation function is to maintain the output or predicted value in the particular range, which makes the good efficiency and accuracy of the model.

Range: 0 to 1

Here Y can be anything for a neuron between range -infinity to +infinity. So, we have to bound our output to get the desired prediction or generalized results.

The major drawback of the sigmoid activation function is to create a vanishing gradient problem.

- This is the Non zero Centered Activation Function
- The model Learning rate is slow
- Create a Vanishing gradient problem.

4. Linear activation function

The linear activation function is defined by

$$F(x) = mx + c.$$

This defines a straight line in the xy-plane.

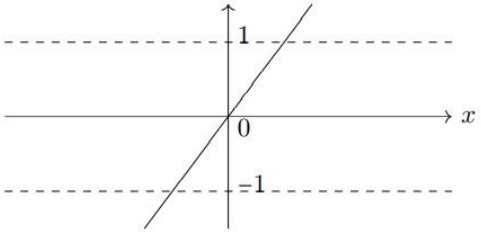


Figure 9.8: Linear activation function

7. Hyperbolic tangential activation function

This is defined by

$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}.$$

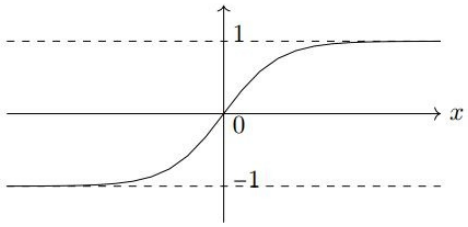


Figure 9.11: Hyperbolic tangent activation function