Cost Function Theoretical Questions

Q1: What is the difference between *Cost Function* vs *Gradient Descent*?

Answer

- A **Cost Function** is *something we want to minimize*. For example, our cost function might be the sum of squared errors over the training set.
- **Gradient Descent** is a *method for finding the minimum* of a function of multiple variables.

Source: towardsdatascience.com

Q2: Explain the steps of k-Means Clustering Algorithm

Answer

K-Means clustering intends to partition `n` objects into `k` clusters in which each object belongs to the cluster with the nearest mean. This method produces exactly *k* different clusters of the greatest possible distinction. The best number of clusters *k* leading to the greatest separation (distance) is not known as a priori and must be computed from the data. The objective of K-Means clustering is to minimize total intra-cluster variance, or, the squared error function:

Algorithm:

1. Clusters the data into *k* groups where *k* is predefined.
2. Select *k* points at *random* as cluster centers.
3. Assign objects to their closest cluster center according to the *Euclidean distance* function.
4. Calculate the *centroid* or *mean* of all objects in each cluster.
5. Repeat steps 2, 3 and 4 until the same points are assigned to each cluster in consecutive rounds.

Source: www.saedsayad.com

Q3: Provide an analogy for a *Cost Function* in real life

Answer

For example, you can imagine a four year-old sitting by a fire to keep warm, but

1. Not knowing the danger of fire, she puts her finger into it and gets burned (VERY HOT)
2. The next time she sits by the fire, she doesn't get burned, but she sits too close, gets too hot and has to move away (A BIT HOT)
3. The third time she sits by the fire she finds the distance that keeps her warm without exposing her to any danger (GOOD)

In other words, through experience and feedback (getting burned, then getting too hot) the kid **learns the optimal distance** to sit from the fire. The **heat from the fire depending on distance** in this example acts as a **cost function — it helps the learner to correct / change behaviour to minimize mistakes.**

Source: towardsdatascience.com

Q4: Explain what is *Cost (Loss) Function* in Machine Learning?

Answer

In ML, **Cost Functions** are used to estimate how badly models are performing. It is a function that **measures the performance of a Machine Learning model** for given data. Cost Function quantifies the error between predicted values and expected values and **presents it in the form of a single real number**.

> ***a cost function is a measure of how wrong the model is in terms of its ability to estimate the relationship between X and y.***

The purpose of Cost Function is to be either:

- **Minimized** - then returned value is usually called **cost**, **loss** or **error**. The goal is to find the values of model parameters for which Cost Function return as small number as possible.
- **Maximized** - then the value it yields is named a **reward**. The goal is to find values of model parameters for which returned number is as large as possible.

Source: towardsdatascience.com

Q5: What is the difference between *Objective function*, *Cost function* and *Loss function*

Answer

Long story short, I would say that:

> A loss function **is a part of** a cost function **which is a type of** an objective function.

These are not very strict terms and they are highly related. However:

- **Loss function** is usually a function defined on a data point, prediction and label, and measures the penalty. For example:
  - square loss $l(f(x_i|\theta), y_i) = (f(x_i|\theta) - y_i)^2$, used in linear regression
  - hinge loss $l(f(x_i|\theta), y_i) = \max(0, 1 - f(x_i|\theta)y_i)$, used in SVM
  - 0/1 loss $l(f(x_i|\theta), y_i) = 1 \iff f(x_i|\theta) \neq y_i$, used in theoretical analysis and definition of accuracy
- **Cost function** is usually more general. It might be a sum of loss functions over your training set plus some model complexity penalty (regularization). For example:
  - Mean Squared Error $MSE(\theta) = \frac{1}{N}\sum_{i=1}^{N}(f(x_i|\theta) - y_i)^2$
  - SVM cost function $SVM(\theta) = \|\theta\|^2 + C\sum_{i=1}^{N}\xi_i$ (there are additional constraints connecting ξ_i with C and with training set)
- **Objective function** is the most general term for any function that you optimize during training. For example, a probability of generating training set in maximum likelihood approach is a well defined objective function, but it is not a loss function nor cost function (however you could define an equivalent cost function). For example:
  - MLE is a type of objective function (which you maximize)
  - Divergence between classes can be an objective function but it is barely a cost function, unless you define something artificial, like 1-Divergence, and name it a cost

All that being said, thse terms are **far from strict**, and depending on context, research group, background, can shift and be used in a different meaning. With the main (only?) common thing being "loss" and "cost" functions being something that want wants to minimise, and objective function being something one wants to optimise (which can be both maximisation or minimisation).

Source: stats.stackexchange.com

Q6: Why don't we use *Mean Squared Error* as a cost function in Logistic Regression?

Answer

- In Linear Regression, we used the **Squared Error** mechanism.
- For **Logistic Regression**, such a cost function produces a **non-convex** space with many local minimums, in which it would be very difficult to minimize the cost value and find the global minimum.

Source: towardsdatascience.com

Q7: How would you fix Logistic Regression *Overfitting* problem?

Answer

If **Logistic Regression** model is overfitting that model has high *variance*.

- **Lasso regularization** (L1 Regularization) and
- **Ridge regularization** (L2 Regularization) can be used

For performing the regularization, we will add *regularization terms* to our cost function as shown below:

Cost Function:

$$\sum_{i=1}^{m}(y - y^{(i)})^2$$

Lasso Regularization:

$$\sum_{i=1}^{m}(y - y^{(i)})^2 + \lambda\sum_{j=0}^{p}||\beta_j||$$

Ridge Regularization:

$$\sum_{i=1}^{m}(y - y^{(i)})^2 + \lambda\sum_{j=0}^{p}||\beta_j||^2$$

Source: towardsdatascience.com

Q8: What is the *Hinge Loss* in SVM?

Answer

When we talk about loss function, what we really mean is a training objective that we want to minimize.

- ***Hinge loss*** is a loss function used for training *classifiers*. Hinge loss is most notably used for *soft-margin* SVMs. The hinge loss penalizes the SVM model for inaccurate predictions (*misclassifications*).

- Hinge loss $l(y, \hat{y})$ is defined to be $\max(0, 1 - y\hat{y})$ where $y \in \{1, -1\}$.

Let's try to understand $y\hat{y}$.

- If $y = 1$, we would want $\hat{y}$ to be as **positive** as possible, in particular, if $\hat{y} \geq 1$, we are happy and the hinge loss would be evaluated to zero. If $y < 1$, we would want to penalize our prediction.

- On the other hand, if $y = -1$, we would want $\hat{y}$ to be as **negative** as possible, in particular, if $\hat{y} \leq -1$, we are happy and the hinge loss would be evaluated to zero. If $y > -1$, we would want to penalize our prediction.

These two conditions can be combined compactly, if the model is doing well, we would want $y\hat{y} \geq 1$ and we want to penalize our model otherwise.

Source: math.stackexchange.com

Q9: What type of *Cost Functions* do *Greedy Splitting* use?

Answer

- For **regression** predictive modeling problems the cost function used is the *sum squared* error across all training samples. It is shown below:

$$RSS = \sum_{i=1}^{n}(y_i - f(x_i))^2$$

  where,

  - y_i is the real output.
  - $f(x_i)$ is the predicted output.

- For **classification** problems the cost function used is the *Gini index* function which provides an indication of how *pure* the leaf nodes are (how mixed the training data assigned to each node is). It is shown below:

$$G(p) = \sum_{k=1}^{K} p_k(1 - p_k)$$

  where, p_k is the probability of the item being in the category `k`.

Source: machinelearningmastery.com

Q10: How would you choose the *Loss Function* for a Deep Learning model?

Answer

- **Binary targets:** In this case, the observed value is drawn from `-1` to `1`. The loss function for this case is shown as follows:

$$L = log(1 + exp(-y.\hat{y}))$$

  Where, $\hat{y}$ is the predicted output. y is the observed output.

  This type of loss function implements a machine learning method known as *logistic regression*.

- **Categorical targets:** If `y` are the probabilities of `k` classes, and `r` th class is the ground truth class, the loss function for a single instance is defined as follows:

$$L = -log(\hat{y}_r)$$

  This type of loss function implements *multinomial logistic regression*, and it is called **cross-entropy loss**.

- **Binary logistic regression** is similar to multinomial logistic regression with the value of `k` set to `2`.

Source: Neural Networks and Deep Learning: A Textbook by Charu C. Aggarwal

Q11: What is the *Objective Function* of *k-Means*?

Answer

- The objective of K-Means clustering is to minimize total intra-cluster variance, or, *distance function*. The **objective function** of *k-means* depends on the proximities of the data points to the *cluster centroids*. It is shown below:

where π_k is the weight of `x`, n_k is the number of data objects assigned to cluster C_k, $m_k = \sum_{x \in C_k} \frac{\pi_x x}{n_k}$ is the centroid of cluster C_k. `K` is the number of clusters set by the user. The function `dist` computes the distance between object `x` and centroid $m_k, 1 \leq k \leq K$.

- While the selection of distance function is optional, the **squared Euclidean distance**, i.e. $|x - m|^2$ has been most widely used in both research and practice.

Source: Advances in K-means Clustering by Junjie Wu

Q12: What *Distance Function* do you use for *Quantitative Data*?

Answer

- The most common distance function for quantitative data is the L_p-norm. The L_p-norm between two data points $\bar{X} = (x_1...x_d)$ and $\bar{Y} = (y_1...y_d)$ is defined as follows:

$$Dist(\bar{X}, \bar{Y}) = (\sum_{i=1}^{d}|x_i - y_i|^p)^{1/p}$$

- Two special cases of the L_p-norm are the **Euclidean** (`p = 2`) and the **Manhattan** (`p = 1`) metrics. These special cases derive their intuition from spatial applications where they have clear physical interpretability.
- The **Euclidean distance** is the straight-line distance between two data points.
- The **Manhattan distance** is the *city block* driving distance in a region in which the streets are arranged as a rectangular grid.

Source: www.amazon.com

Q13: What are some necessary Mathematical Properties a Loss Function needs to have in Gradient-Based Optimization?

Answer

- In general, the loss needs to be differentiable, with some caveats. \ Consider a neural network with parameters $\theta \in R^d$, which is usually a vector of weights and biases. The gradient descent algorithm seeks to find parameters θ_{min} which minimize the loss function:

$$L : R^d \to R$$

Gradient descent is performed by the update rule:

$$\theta_n \leftarrow \theta_{n-1} - \gamma\nabla L(\theta_{n-1})$$

yielding new parameters θ_n which should give a smaller loss $L(\theta_n)$. The quantity γ is the learning rate. \ The gradient descent rule requires the gradient $\nabla L(\theta_{n-1})$ to be defined, so the loss function must be differentiable. \ In most texts on calculus or mathematical analysis you'll find the result that if a function is differentiable at a point `x`, it is also continuous at `x`. Obviously, there is no hope that we could perform this procedure without knowing the gradient. \ In principle, differentiability is sufficient to run gradient descent. That said, unless `L` is convex, gradient descent offers no guarantees of convergence to a global minimizer. In practice, neural network loss functions are rarely convex anyway.

- An unfortunate technicality that has to be mentioned is that strictly speaking if you use the **ReLU** activation function, the neural network function `f` becomes non-differentiable.

Source: ai.stackexchange.com