**Loss-Functions for Machine Learning Algorithms**

* In supervised machine learning algorithms, we want to minimize the error (coming form a loss function) for each training example during the learning process. This is done using some optimization strategies like gradient descent.
* Loss functions provide metrics to compare different ML approaches
* A loss function maps decisions to their associated costs.
* Importantly, the choice of loss function is directly related to the activation function used in the output layer of your neural network. These two design elements are connected!

**Maximum Likelihood**

* Maximum likelihood estimation (MLE), is a framework for inference for finding the best statistical estimates of parameters from historical training data.
* MLE can be defined as a method for estimating parameters (such as the mean or variance ) from sample data such that the probability (likelihood) of obtaining the observed data is maximized.
* A common modeling problem involves how to estimate a joint probability distribution for a dataset.
* The joint probability distribution can be restated as the multiplication of the conditional probability for observing each example given the distribution parameters.
* Density estimation involves selecting a probability distribution function and the parameters of that distribution that best explain the joint probability distribution of the observed data (X).
* In MLE, we wish to maximize the probability of observing the data from the joint probability distribution given a specific probability distribution and its parameters
* One way to interpret MLE is to view it as minimizing the dissimilarity between the empirical distribution […] defined by the training set and the model distribution, with the degree of dissimilarity between the two measured by the KL divergence. […] Minimizing this KL divergence corresponds exactly to minimizing the cross-entropy between the distributions.
* Almost universally, deep learning neural networks are trained under the framework of MLE using cross-entropy as the loss function, due to reasons of consistency and efficiency
  + Consistency: As the number of training examples approaches infinity, the maximum likelihood estimate of a parameter converges to the true value of the parameter.
  + Efficiency: A way to measure how close we are to the true parameter is by the expected mean squared error,over m training samples. That parametric mean squared error decreases as m increases, and for m large, the Cramér-Rao lower bound shows that no consistent estimator has a lower mean squared error than the maximum likelihood estimator.

The following steps describe each of the following loss functions below:

1. Write the expression for our predictor function, f(X) and identify the parameters that we need to find
2. Identify the loss to use for each training example
3. Find the expression for the Cost Function – the average loss on all examples
4. Find the gradient of the Cost Function with respect to each unknown parameter
5. Decide on the learning rate and run the weight update rule for a fixed number of iterations

|  |  |
| --- | --- |
| **Regression Loss Functions** | |
| **Squared Error Loss (L2 Loss)**   * MSE loss function is a positive quadratic function and has a global minimum. Hence, it is always guaranteed that Gradient Descent will converge (if it converges at all) to the global minimum. * The MSE loss function penalizes the model for making large errors by squaring them. Therefore, it should not be used if our data is prone to many outliers. * Under the framework of maximum likelihood estimation and assuming a Gaussian distribution for the target variable, mean squared error can be considered the cross-entropy between the distribution of the model predictions and the distribution of the target variable. For example, mean squared error is the cross-entropy between the empirical distribution and a Gaussian model. |  |
| **Absolute Error Loss (L1 Loss)**   * The MAE cost is more robust to outliers as compared to MSE |  |
| **Huber Loss**   * The Huber loss combines the best properties of MSE and MAE. * Huber loss is more robust to outliers than MSE. It is used in Robust Regression, M-estimation and Additive Modelling. A variant of Huber Loss is also used in classification. |  |

|  |  |
| --- | --- |
| **Binary Classification Loss Functions** | |
| **Binary Cross Entropy Loss**   * Entropy indicates disorder or uncertainty. It is measured for a random variable X with probability distribution p(X): * *A greater value of entropy for a probability distribution indicates a greater uncertainty in the distribution. Likewise, a smaller value indicates a more certain distribution.* * This makes binary cross-entropy suitable as a loss function – you want to minimize its value. * Under the framework maximum likelihood, the error between two probability distributions is measured using cross-entropy. * In the training dataset, the probability of an example belonging to a given class would be 1 or 0 * Therefore, under maximum likelihood estimation, we would seek a set of model weights that minimize the difference between the model’s predicted probability distribution given the dataset and the distribution of probabilities in the training dataset * Cross-entropy for a binary problem is calculated as the average cross entropy across all examples. | S = Entropy Level; (-) before sum yields positive value    Figure 1: L = Log-Loss    Sigmoid function is used to calculate p |
| **Hinge Loss**   * Primarily used with Support Vector Machine (SVM) Classifiers with class labels -1 and 1 * Hinge Loss not only penalizes the wrong predictions but also the right predictions that are not confident. | L = Hinge Loss |

|  |  |
| --- | --- |
| **Multi-Class Classification Loss Functions** | |
| **Multi-Class Cross Entropy Loss**   * Generalization of the Binary Cross Entropy loss. * “Softmax is implemented through a neural network layer just before the output layer. The Softmax layer must have the same number of nodes as the output layer.” | The loss for input vector X\_i and the corresponding one-hot encoded target vector Y\_i |
| **KL-Divergence**   * The Kullback-Liebler Divergence is a measure of how a probability distribution differs from another distribution. A KL-divergence of zero indicates that the distributions are identical. * The divergence function is not symmetric.. This is why KL-Divergence cannot be used as a distance metric. * We want to approximate the true probability distribution P of our target variables with respect to the input features, given some approximate distribution Q. Since KL-Divergence is not symmetric, we can do this in two ways: * *KL-Divergence is used more commonly to approximate complex functions than in multi-class classification. We come across KL-Divergence frequently while playing with deep-generative models like Variational Autoencoders (VAEs).* | 1. used in Supervised learning; 2. used in Reinforcement Learning     KL-Divergence is functionally similar to multi-class cross-entropy and is also called relative entropy of P with respect to Q |