Chapter 03 – Model Training | Loss function

Author: Gianmarco Scarano

gianmarcoscarano@gmail.com

1. Definition

A loss function $L(\phi)$ is a function which returns a single number that describes the mismatch between the model predictions $f[x_i, \phi]$ and their corresponding ground-truth outputs y_i. So, during training we seek that parameter $\tilde{\phi}$ that minimizes the loss and hence maps the training inputs to the outputs as closely as possible:

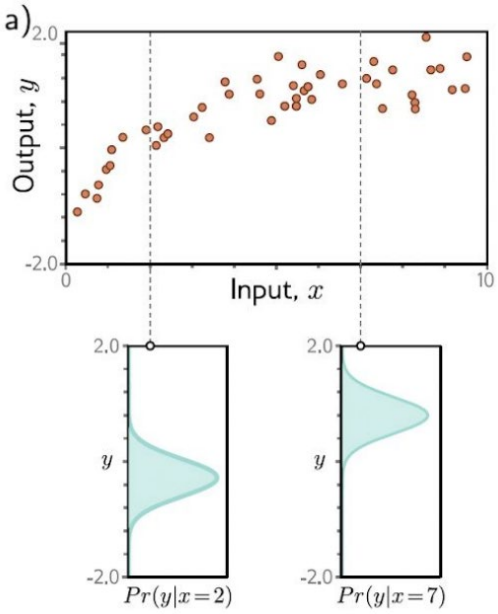
$$\tilde{\phi} = \underset{\phi}{\text{argmin}}\, \text{L}[\phi]$$

2. Maximum Likelihood

In Maximum likelihood, we model our loss function as a conditional probability distribution over the outputs with the following formula:

$$y = f[x, \phi] \implies Pr\{y|x; \phi\}$$

So, for example, in a regression task we might have to predict a real-valued output y from the input x based on training data $\{x_i, y_i\}$. For each input value x, the ML model predicts a distribution $Pr(y|x)$ over the output y (right image).



In other words, this prediction is done using different distribution parameters $\theta_i = f[x_i, \phi]$

and each observed training output y_i should have high probability under its corresponding distribution, such that we are going to choose the model parameters ϕ that maximizes the combined probability across all *i*-training examples. This is done through the argmax of the likelihood.

$$\underset{\phi}{\text{argmax}} \left[\prod_{i=1}^{I} Pr(\mathbf{y}_i|\mathbf{f}[\mathbf{x}_i, \boldsymbol{\phi}]) \right]$$

The likelihood is the product of the probabilities of each y_i given x_i (each pair contained in our dataset) which translates in computing the probabilities of y_i given the model parameters. I'm taking the maximum because I want to maximize the probaility of recovering the data in our dataset by using my $f(\cdot)$ function (which most of the times is a Neural Network).

MLE always asks for assumptions, such that the observations (x_i, y_i) must be independent and identically distributed.

$$Pr(\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_I|\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_I) = \prod_{i=1}^{I} Pr(\mathbf{y}_i|\mathbf{x}_i)$$

2.1 Log Likelihood & Negative Log Likelihood

Log likelihood adds a simple log function before the product of each Probability (2nd formula above) in order to have a sum appear (which is an easier mathematical operation than product).

$$\begin{aligned} &= \underset{\phi}{\text{argmax}} \left[\log \left[\prod_{i=1}^{I} Pr(\mathbf{y}_i|\mathbf{f}[\mathbf{x}_i, \phi]) \right] \right] \\ &= \underset{\phi}{\text{argmax}} \left[\sum_{i=1}^{I} \log \left[Pr(\mathbf{y}_i|\mathbf{f}[\mathbf{x}_i, \phi]) \right] \right] \end{aligned}$$

However, we see that fitting a model means minimizing a loss function. To convert the maximum log-likelihood criterion to a minimization problem, we multiply the whole quantity by (-1), which indeed returns us the negative log-likelihood criterion as we can see on the right.

$$\begin{aligned} \hat{\phi} &= \underset{\phi}{\text{argmin}} \left[\ominus \sum_{i=1}^{I} \log \left[Pr(\mathbf{y}_i|\mathbf{f}[\mathbf{x}_i, \phi]) \right] \right] \\ &= \underset{\phi}{\text{argmin}} \left[\mathrm{L}[\phi] \right], \end{aligned}$$

In the end, then, for inference works, we simply determine a probability distribution over y. But often, we want a point estimate rather than a distribution, so we return the maximum of the distribution in terms of the parameters θ predicted by the model:

$$\hat{\mathbf{y}} = \underset{\mathbf{y}}{\text{argmax}} \left[Pr(\mathbf{y}|\mathbf{f}[\mathbf{x}, \hat{\phi}]) \right]$$

In other words, given a new input, we take the y which maximizes the output of our probability.

A usual way of seeing things during a loss is that we first learn the parameters of our Neural Network and then we compute the probability of the output, given those parameters.

Recipe for a Loss

1. Choose a suitable probability distribution Pr(y|θ) defined over the domain of the predictions y with distribution parameters θ.
2. Set the machine learning model f[x, ϕ] to predict one or more of these parameters, so θ = f[x, ϕ] and Pr(y|θ) = Pr(y|f[x, ϕ]).
3. To train the model, find the network parameters $\hat{\phi}$ that minimize the negative log-likelihood loss function over the training dataset pairs $\{x_i, y_i\}$:

$$\hat{\phi} = \underset{\phi}{\text{argmin}} \left[L[\phi] \right] = \underset{\phi}{\text{argmin}} \left[-\sum_{i=1}^{I} \log \left[Pr(\mathbf{y}_i|\mathbf{f}[\mathbf{x}_i, \phi]) \right] \right]$$

4. To perform inference for a new test example x, return either the full distribution Pr(y|f[x,$\hat{\phi}$|) or the maximum of this distribution.

In the example given on the book, for a univariate regression problem we aim at finding parameters which compute the mean μ, such that $\mu = f[x, \phi]$. By simple calculations and by following the recipe for losses, we substitute μ with $f[x, \phi]$ and by taking out all the rest (meaning that we remove things which do not involve μ, we are left with the following formula which is the **Mean Squared Error** Loss):

$$Pr(y|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{(y-\mu)^2}{2\sigma^2}\right]$$

pdf
Probability Density Function

$$\begin{aligned}
\hat{\phi} &= \underset{\phi}{\operatorname{argmin}} \left[-\sum_{i=1}^{I} \log \left[\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{(y_i - \mathrm{f}[\mathbf{x}_i, \phi])^2}{2\sigma^2} \right] \right] \right] \\
&= \underset{\phi}{\operatorname{argmin}} \left[-\sum_{i=1}^{I} \log \left[\frac{1}{\sqrt{2\pi\sigma^2}} \right] - \frac{(y_i - \mathrm{f}[\mathbf{x}_i, \phi])^2}{2\sigma^2} \right] \\
&= \underset{\phi}{\operatorname{argmin}} \left[-\sum_{i=1}^{I} -\frac{(y_i - \mathrm{f}[\mathbf{x}_i, \phi])^2}{2\sigma^2} \right] \\
&= \underset{\phi}{\operatorname{argmin}} \left[\sum_{i=1}^{I} (y_i - \mathrm{f}[\mathbf{x}_i, \phi])^2 \right]
\end{aligned}$$

MSELoss

2.2 Binary Cross Entropy and Multiclass Classification

For another example, such as the Binary Classification, we might want to use the Binary Cross Entropy Loss.

In BC, we want to predict if a value x is of class (for example) 0 or 1. A suitable choice for the probability distribution is the Bernoulli distribution which is defined in the domain $\{0, 1\}$ (through a parameter λ).

$$\Pr(y \mid \lambda) = (1 - \lambda)^{1-y} \cdot \lambda^{y}$$

We set our ML model to predict this value, but since we are not guaranteed that the network will output the value between 0 and 1, we clip it using a logistic sigmoid.

So, in the end, the parameter that we want to learn (λ) ends up being: $sig[f[x, \varphi]]$

- The likelihood is, therefore:

$$Pr(y|\mathbf{x}) = (1 - \mathrm{sig}[\mathrm{f}[\mathbf{x}, \phi]])^{1-y} \cdot \mathrm{sig}[\mathrm{f}[\mathbf{x}, \phi]]^{y}$$

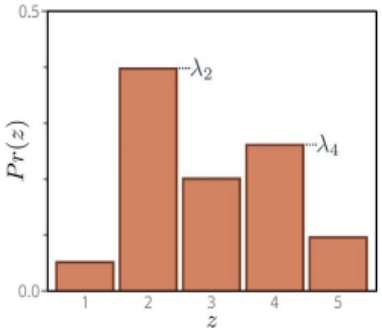
- And the Negative Log Likelihood is:

$$L[\phi] = \sum_{i=1}^{I} -(1 - y_i) \log\left[1 - \mathrm{sig}[\mathrm{f}[\mathbf{x}_i, \phi]]\right] - y_i \log\left[\mathrm{sig}[\mathrm{f}[\mathbf{x}_i, \phi]]\right]$$

For multiclass classification, instead of mapping a value to a class 0 or 1, we want to map it to > 2 classes (so y can be 1, 2, 3, etc..).

The methodology is almost the same as the BC problem, so we want to choose an appropriate representation of the distribution, in this case: Categorical distribution.

The parameters are constrained to take values between zero and one, and they must collectively sum to one to ensure a valid probability distribution.



Due to the fact that we still have the network problem, we use the non-linear SoftMax function which clips our values of the classes between 0 and 1 (having the sum up to 1). So, in the end:

- Therefore the likelihood is given by:

$$Pr(y = k|\mathbf{x}) = \text{softmax}_k\Big[\mathbf{f}[\mathbf{x}, \boldsymbol{\phi}]\Big]$$

- And the NLL is given by:

$$L[\boldsymbol{\phi}] = -\sum_{i=1}^{I} \log\Big[\text{softmax}_{y_i}\Big[\mathbf{f}[\mathbf{x}_i, \boldsymbol{\phi}]\Big]\Big]$$

Remember, the *negative log-likelihood criterion* (which maximizes the data likelihood) and the *cross-entropy criterion* (which minimizes the distance between the model and the data distribution) are equivalent.