Data Science 2

Loss Functions

Ondřej Týbl

Charles University, Prague

Outline

Maximum Likelihood Estimation

Definition

Given a network architecture (i.e., the computation graph with specified activation functions), we aim to find a set of weights W^j and biases w_0^j to represent the underlying data.

Putting all the weights and biases together into a single parameter vector:

$$\theta = \left(\text{vector}(W^1), w_0^1, \text{vector}(W^2), w_0^2, \dots\right) \in \Theta,$$

where Θ denotes the parameter space and vector(A) denotes the vectorization of matrix A. We shall write $f_\theta(x) \in \mathbb{R}^n$ for the output of the network given the input $x \in \mathbb{R}^m$.

Definition

We identify the network output $f_\theta(x) \in \mathbb{R}^n$ with a probability density $p_\theta(\cdot|x)$ in the label space $\mathbb{R}^n$, e.g. in the case of a continuous labels we shall put

$$p_\theta(\cdot|x) = \mathcal{N}\left(f_\theta(x), \sigma^2 I_{n\times n}\right)$$

for some unknown $\sigma^2 > 0$.

By means of Maximum Likelihood Estimation we find $\hat{\theta} \in \Theta$ so that $p_{\hat{\theta}}(\cdot|x)$ is *close* to the true label distribution given input $x \in \mathbb{R}^m$.

Definition

Definition (Maximum Likelihood Estimation for Neural Networks)[1]

Let $x_1, \ldots, x_N \in \mathbb{R}^m$ be training examples of input data with labels drawn independently from some (unknown) conditional probability density, that is we have $y_j \sim p_{model}\left(\cdot|x_j\right), j = 1, \ldots N$. The *Maximum Likelihood Estimation* of $\theta \in \Theta$ is given by

$$\hat{\theta} = \arg\max_{\theta \in \Theta} \prod_{j=1}^{N} p_\theta\left(y_j|x_j\right)$$

[1] We use a simplified formulation where the empirical distribution for x coincides with the true one. See Sec.2.2 in [1] for the general case.

Definition

Assume there exists unique $\theta_{TRUE} \in \Theta$ so that the true conditional distribution $p_{model}(\cdot|x)$ is equal to $p_{\theta_{TRUE}}(\cdot|x)$.

The theory gives us important asymptotical properties with growing sample size $N \to \infty$

- *Consistency*: The estimator converges to $\theta_{TRUE} \in \Theta$ in probability,
- *Efficiency*: The estimator achieves the Cramér–Rao lower bound, i.e. no consistent estimator has lower asymptotic mean squared error.

Regression task

In the regression task, with continuous labels $y_1, \ldots, y_N \in \mathbb{R}^n$, we represent (as above) the distribution $p_\theta(\cdot|x) = \mathcal{N}\left(f_\theta(x), \sigma^2 I_{n\times n}\right)$, for some unknown $\sigma^2 > 0$. Then we have

$$
\begin{aligned}
\hat{\theta} &= \arg\max_{\theta\in\Theta} \prod_{j=1}^{N} p_\theta\left(y_j|x_j\right) \\
&= \arg\min_{\theta\in\Theta} -\sum_{j=1}^{N} \log\left(p_\theta\left(y_j|x_j\right)\right) \\
&= \arg\min_{\theta\in\Theta} -\sum_{j=1}^{N} \log\left(\frac{1}{\sqrt{2\pi\sigma^{2n}}} \exp\left[-\frac{\left(y_j - f_\theta\left(x_j\right)\right)^T\left(y_j - f_\theta\left(x_j\right)\right)}{2\sigma^2}\right]\right) \\
&= \arg\min_{\theta\in\Theta} \sum_{j=1}^{N} \left[\frac{1}{2\sigma^2}\left(y_j - f_\theta\left(x_j\right)\right)^T\left(y_j - f_\theta\left(x_j\right)\right)\right] + const \\
&= \arg\min_{\theta\in\Theta} \frac{1}{N}\sum_{j=1}^{N} \left[\left(y_j - f_\theta\left(x_j\right)\right)^T\left(y_j - f_\theta\left(x_j\right)\right)\right] + const
\end{aligned}
$$

Regression task

Thus, Maximum Likelihood Estimation in the regression task is equivalent to minimization of the mean squared error loss function

$$\mathcal{L}(\theta) = \frac{1}{N} \sum_{j=1}^{N} \left(y_j - f_\theta\left(x_j\right)\right)^T \left(y_j - f_\theta\left(x_j\right)\right)$$

over the training data.

Regression task

Consider a multilayer perceptron with single matrix multiplication

$$f(x) = \varphi\left(Wx + w_0\right),$$

where φ is the identity, $W \in \mathbb{R}^m$ and $w_0 \in \mathbb{R}$.

From the correspondence of mean squared error minimization and Maximum Likelihood Estimation, the problem of the W, w_0 estimation is equivalent to the problem of least-squares estimation in the linear regression model

$$y_j = Wx_j + w_0 + \epsilon_j,$$

where the error terms $\epsilon_j \sim \mathcal{N}\left(0, \sigma^2\right)$ are independent.

Intermezzo: Information Theory

Entropy

Definition (Self-Information)

For an event with probability p, we define the amount *self-information* (or amount surprise) as

$$I(p) = -\log p$$

We motivate the definition by the requirements

- an almost sure event has no self-information as when it occurs no additional information is brought,
- less likely events bring more self-information as when they occur we learn more about the system,
- self-information is additive on the system of independent events (for algebraic simplicity).

Entropy

Definition (Entropy)

For a discrete random variable X with a distribution p_X we define its *entropy* as the expected self-information

$$H(X) = \mathbb{E}\left[I(p_X)\right] = \mathbb{E}\left[-\log(p_X(X))\right],$$

The entropy of a distribution is the entropy of a random variable following this distribution.

If p_X is supported on a countable set $\mathcal{X}$ we have

$$H(X) = -\sum_{x \in \mathcal{X}} p_X(x) \log p_X(x),$$

where $0 \cdot \log 0$ is defined via the limit $0 \log 0 = \lim_{p \to 0+}(p \log p) = 0$.

Entropy

Among the discrete random variables on $\{1, \ldots, N\}$

- the one-hot distribution, which is equal to one for some $j \in \{1, \ldots N\}$ and zero else elsewhere, has the lowest entropy and is equal to zero,
- the uniform distribution has the highest entropy and is equal to

$$-\sum_{j=1}^{N} \frac{1}{N} \log \frac{1}{N} = \log N$$

Entropy

Entropy can be also defined for continuous random variables and a well-known fact is that among the distributions with fixed finite variance, normal distribution is the one with the highest entropy.

As a measure of a deviation from the mean, variance could also be used to measure the amount of *surprise* or *self-information*. However, entropy, unlike variance does not depend on the actual values but only on the probabilities which is often desirable[2].

[2] A nice account on the applications may be found in [2]

Definition (Cross-Entropy)

For two discrete probability distributions P and Q, we define the *cross-entropy* as the expected self-information of P given Q

$$H(P, Q) = \mathbb{E}_P [I(Q)] = \mathbb{E}[-\log(Q(X))],$$

where $X \sim P$.

If P, Q are supported on a countable set $\mathcal{X}$ we have

$$H(P, Q) = -\sum_{x \in \mathcal{X}} P(x) \log Q(x),$$

again with the convention $0 \log 0 = 0$.

Cross-Entropy

H is not symmetric.

In $H(P, Q)$ the distribution P plays the role a of a prior.

Kullback-Leibler Divergence

Theorem (Gibbs' inequality, proof on p.56 in [2])

For two discrete probability distributions P and Q, we have

$$H(P, Q) \geq H(P)$$

with equality if and only if $P = Q$.

Using Gibb's inequality we can standardize the cross-entropy to be zero for identical distributions.

Definition (Kullback-Leibler Divergence)

For two discrete probability distributions P and Q, we define the *Kullback-Leibler Divergence*

$$D_{KL}(P \| Q) = H(P, Q) - H(P) = \mathbb{E}\left[\log \frac{P(X)}{Q(X)}\right]$$

where $X \sim P$.

Kullback-Leibler Divergence

Compares amount of surprise of Q given P to the amount of surprise in P itself and thus is a good candidate for measuring distances between distributions.

However, D_{KL} is not symmetric (as already H is not) and thus it is not a proper distance.

We could symmetrize the divergence

$$D_{KL}^{S}(P, Q) = D_{KL}(P, Q) + D_{KL}(Q, P)$$

but it is not desirable as can be seen from the example below.

Kullback-Leibler Divergence

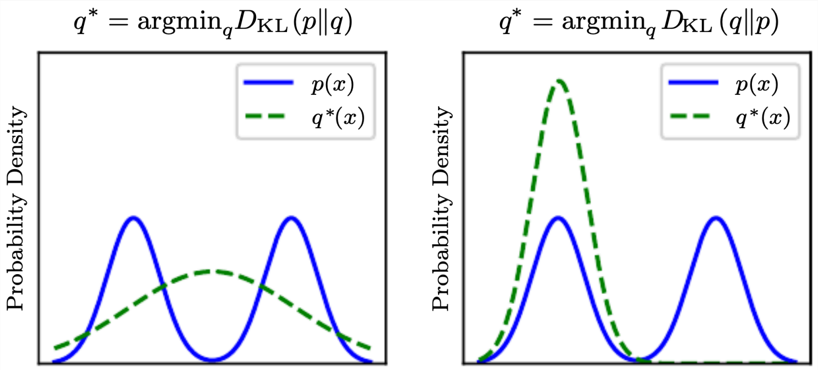

Figure 1: Comparison of what happens to the divergence minimization problem when the two distributions are swapped. Here p denotes the target distribution, q denotes the approximation distribution when chosen from a *set of distributions with one peak*.

Kullback-Leibler Divergence

From the above example we see that symmetricity is not a desirable property and we will minimize D_{KL} directly when training the network.

Maximum Likelihood Estimation Revisited

Classification task

In the classification task[3], with categorical labels $y_1, \ldots, y_N \in \{0, 1\}$, we represent the distribution using the sigmoid function

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

$$p_\theta(\{1\}|x) = \sigma(f_\theta(x)) = \frac{1}{1 + e^{-f_\theta(x)}}$$

$$p_\theta(\{0\}|x) = 1 - \sigma(f_\theta(x)) = \frac{1}{1 + e^{f_\theta(x)}} = \sigma(-f_\theta(x)).$$

For the label distribution we have

$$p_{model}\left(\{1\}|x_j\right) = \mathbb{1}_{[y_j=1]}$$

$$p_{model}\left(\{0\}|x_j\right) = \mathbb{1}_{[y_j\neq 1]}$$

[3] We use two classes for simplicity, but the following can be adjusted to multiple-classes problem.

Classification task

Then we have

$$\begin{aligned}
\hat{\theta} &= \arg\max_{\theta\in\Theta} \prod_{j=1}^{N} p_\theta\left(y_j|x_j\right) \\
&= \arg\min - \sum_{j=1}^{N} \log\left(p_\theta\left(y_j|x_j\right)\right) \\
&= -\arg\min \sum_{j=1}^{N} \left[1_{[y_j=1]} \log\left(\sigma\left(f_\theta\left(x\right)\right)\right) + 1_{[y_j=0]} \log\left(\sigma\left(-f_\theta\left(x\right)\right)\right)\right] \\
&= -\arg\min \sum_{j=1}^{N} \left[p_{model}\left(\{1\}|x_j\right) \log\left(p_\theta\left(\{1\}|x\right)\right)\right. \\
&\qquad\qquad \left. + p_{model}\left(\{0\}|x_j\right) \log\left(p_\theta\left(\{0\}|x\right)\right)\right] \\
&= \arg\min_{\theta\in\Theta} \sum_{j=1}^{N} H\left(p_{model}\left(\cdot|x_j\right), p_\theta\left(\cdot|x_j\right)\right)
\end{aligned}$$

Classification Task

Thus, Maximum Likelihood Estimation in the classification task is equivalent to minimizing the cross-entropy loss function:

$$\frac{1}{N}\sum_{j=1}^{N} H\left(p_{\text{model}}\left(\cdot \mid x_j\right), p_\theta\left(\cdot \mid x_j\right)\right)$$

over the training data. Since $H\left(p_{\text{model}}(\cdot \mid x)\right)$ does not depend on θ, this is also equivalent to minimizing the Kullback-Leibler divergence:

$$\begin{aligned}
&\arg\min_{\theta\in\Theta} \frac{1}{N}\sum_{j=1}^{N} H\left(p_{\text{model}}\left(\cdot \mid x_j\right), p_\theta\left(\cdot \mid x_j\right)\right) \\
&\qquad = \arg\min_{\theta\in\Theta} \frac{1}{N}\sum_{j=1}^{N} D_{\text{KL}}\left(p_{\text{model}}\left(\cdot \mid x_j\right) \parallel p_\theta\left(\cdot \mid x_j\right)\right)
\end{aligned}$$

Consider a multilayer perceptron with single matrix multiplication

$$f(x) = \varphi\left(Wx + w_0\right),$$

where $\varphi = \sigma$ is the sigmoid function, $W \in \mathbb{R}^m$ and $w_0 \in \mathbb{R}$.

From the correspondence of the cross-entropy minimization and Maximum Likelihood Estimation, the problem of the W, w_0 estimation is equivalent to the problem of logistic regression[4] with the model

$$p\left(x\right) = \frac{1}{1 + e^{-\left(Wx + w_0\right)}}.$$

[4]Recall the logistic loss from the previous lectures which is the same as our cross-entropy loss

Classification task

In the case of a multi-class classification, we represent the labels $y_1, \ldots, y_N \in \{1, \ldots, K\}$ for some $K \in \mathbb{N}$ and use a network architecture so that $f_\theta(x) \in \mathbb{R}^K$ represents the logits.

Instead of applying the sigmoid function σ we create an output vector that represents a probability distribution:

$$p_\theta(x) = \operatorname{softmax}(f_\theta(x)),$$

where

$$\operatorname{softmax}(v) = \left(\frac{e^{v_1}}{\sum_{j=1}^m e^{v_j}}, \ldots, \frac{e^{v_m}}{\sum_{j=1}^m e^{v_j}} \right), \quad v \in \mathbb{R}^m$$

so that the image of softmax has always positive values that sum to one.

Classification task

Similar correspondence to cross-entropy minimization is then obtained. Another good reason to use softmax is also the simple form of the derivatives with respect to the network output $f_\theta(x)$ which will be important in the training procedure.

Zapomněnka[5]: A nice motivation for the choice of the softmax function for classification task is explained here: link.[6]

[5] to-be-forgotten

[6] Loosely speaking, softmax function should be used to model probabilities, if we want to obtain a distribution with maximum entropy under some constraints.

C. M. Bishop.
Neural networks for pattern recognition.
Oxford university press, 1995.

D. J. MacKay.
Information theory, inference and learning algorithms.
Cambridge university press, 2003.