Notes

Sebastian Rietsch

May 2, 2019

Some notes about Machine Learning concepts.

Contents

1 Maximum Likelihood Estimation (MLE)[1] [2]

Maximum likelihood estimation is a method that determines values for the parameters of a model. The parameter values are found such that they maximise the likelihood that the process described by the model produced the data that were actually observed.

We first have to decide which model we think best describes the process of generating the data.

Then what we want to calculate is the total probability of observing all of the data, i.e. the joint probability distribution of all observed data points. To do this we would need to calculate some conditional probabilities, which can get very difficult. So it is here that we will make our first assumption. *The assumption is that each data point is generated independently of the others.* This assumption makes the maths much easier. If the events (i.e. the process that generates the data) are independent, then the total probability of observing all of data is the product of observing each data point individually (i.e. the product of the marginal probabilities).

$$f(x_1, \ldots, x_n; \theta) = \prod_{i=1}^{n} f(x_i; \theta)$$

$$L(\theta) = \prod_{i=1}^{n} f_\theta(x_i)$$

We now search for the parameters θ that maximize the Likelihood-function L, i.e. $\theta_{ML} = \arg\max_{\theta \in \Theta} L(\theta)$. We can do this by differentiation. All we have to do is find the derivative of the function.

The above expression for the total probability is actually quite a pain to differentiate, so it is almost always simplified by taking the natural logarithm of the expression. This is absolutely fine because the natural logarithm is a monotonically increasing function. This means that if the value on the x-axis increases, the value on the y-axis also increases. This is important because it ensures that the maximum value of the log of the probability occurs at the same point as the original probability function.

$$log(L(\theta)) = log(\prod_{i=1}^{n} f_\theta(x_i)) = \sum_{i=1}^{n} log(f_\theta(x_i))$$

2 Binary cross entropy [3]

$$H_p(q) = -\frac{1}{N} \sum_{i=1}^{N} y_i \cdot log(p(y_i)) + (1 - y_i) \cdot (1 - p(y_i))$$

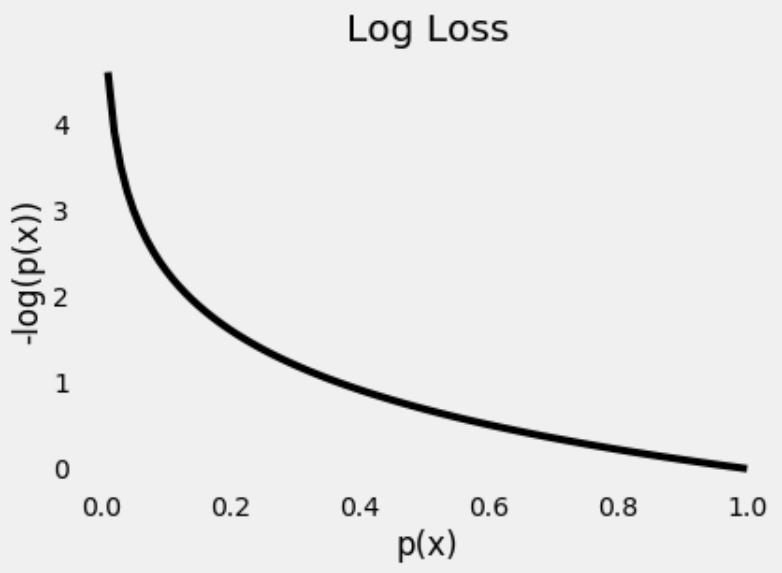

Figure 1: Negative log loss

where y_i is the true class label and $p(y_i)$ is the predicted probability of x_i coming from the positive class $y = 1$. Basically: **sum of negative logs of predicted true class probabilities** (weighted by number of samples). Look at figure 1 for negative log loss.

2.1 Diving deeper

Entropy is a measure of uncertainty associated with a given distribution $q(y)$.

$$H(q) = -\sum_{c=1}^{C} q(y_c) \cdot log(q(y_c))$$

Example with two classes (and *log* base 2):

- All points from one class: $H(q) = -1 \cdot log(1) = 0 \Rightarrow$ no uncertainty
- 50:50 distribution: $H(q) = -log(0.5) = 1 \Rightarrow$ maximum uncertainty

3 Generative Adversarial Networks (GANs) [4]

3.1 How do GANs work?

The basic idea of GANs: set up game between two players. One of them is called **generator** and creates samples that are intended to come from the same distribution as the training data. The other player is the **discriminator** who examines samples to determine whether they are real or fake.

Formally, GANs are a structured probabilistic model containing latent variables z and observed variables x.

The two players in the game are represented by two functions, each of which is differentiable both with respect to its inputs and with respect to its parameters.

- Discriminator: D takes x as input and uses $\theta^{(D)}$ as parameters
- Generator: G takes z as input and uses $\theta^{(G)}$ as parameters

Both players have cost functions that are defined in terms of both players parameters.

- Discriminator: wishes to minimize $J^{(D)}(\theta^{(D)}, \theta^{(G)})$ while controlling only $\theta^{(D)}$
- Generator: wishes to minimize $J^{(G)}(\theta^{(D)}, \theta^{(G)})$ while controlling only $\theta^{(G)}$

Because each player's cost depends on the other players parameters, but each player cannot control the other players parameters, this scenario is most straightforward to describe as a game rather than as an optimization problem. . The solution to an optimization problem is a (local) minimum, a point in parameter space where all neighboring points have greater or equal cost. The solution to a game is a Nash equilibrium. . Here, we use the terminology of local differential Nash equilibria (Ratliff et al., 2013). In this context, a Nash equilibrium is a tuple $(\theta^{(D)}, \theta^{(G)})$ that is a local minimum of $J^{(D)}$ w.r.t $\theta^{(D)}$ and a local minimum of $J^{(G)}$ w.r.t $\theta^{(G)}$.

The training process. The training process consists of simultaneous SGD. On each step, two minibatches are sampled: a minibatch of x values from the dataset and a minibatch of z values drawn from the models prior over latent variables. Then two gradient steps are made simultaneously: one updating $\theta^{(D)}$ to reduce $J^{(D)}$ and one updating $\theta^{(G)}$ to reduce $J^{(G)}$. In both cases, it is possible to use the gradient-based optimization algorithm of your choice. Adam (Kingma and Ba, 2014) is usually a good choice.

3.2 Cost functions

Several different cost functions may be used within the GANs framework.

3.2.1 The discriminators cost, $J^{(D)}$

The cost used for the discriminator is:

$$J^{(D)}(\theta^{(D)}, \theta^{(G)}) = -\frac{1}{2}\mathbb{E}_{x\sim p_{data}} logD(x) - \frac{1}{2}\mathbb{E}_z log(1 - D(G(z))).$$

This is just the standard cross-entropy cost that is minimized when training a standard binary classifier with a sigmoid output. The only difference is that the classifier is trained on two minibatches of data; one coming from the dataset, where the label is 1 for all examples, and one coming from the generator, where the label is 0 for all examples.

References

[1] Maximum likelihood methode. https://de.wikipedia.org/wiki/Maximum-Likelihood-Methode. Accessed: 02.05.2019.

[2] Probability concepts explained maximum likelihood estimation. https://towardsdatascience.com/probability-concepts-explained-maximum-likelihood-estimation-c7b4342fdbb1. Accessed: 02.05.2019.

[3] Understanding binary cross entropy log loss a visual explanation. https://towardsdatascience.com/understanding-binary-cross-entropy-log-loss-a-visual-explanation-a3ac6025181a. Accessed: 02.05.2019.

[4] I. Goodfellow. Nips 2016 tutorial: Generative adversarial networks, 2016.