Cross entropy

Akash Garg

Version 1.0 - October 6, 2020

Image classification has become the hot new thing. Coming from a typical graphics background, the typical problem setup is generally, setup an objective function and then solve for the unknowns. What should this objective function be?

In the case of image classification we have we have some a set of binary values (or delta functions) that look like this: $p = [0, ...1, ...0]$, where the value for the i'th value is 1 when the input matches the i'th label.

The "function" we seek to minimize of course is just neural network itself and the parameters are the weights.

What should the objective function be? Typically we'd use something like $L = ||f(x) - p(x)||^2$; but that doesn't work very well for classification problems (I need to followup on this and explore why). But from the literature it looks like it doesn't.

To map our function values $q = f(x)$ to a binary output p we can treat them as probability distributions. Then the objective-function we week to minimize is the cross-entropy loss term:

$$H(p, q) = -\sum_i p(i) \log(q(i))$$

Intuitively this measures how well a predicted probability distribution (denoted by q) matches the true distribution (denoted by p).

Coming from graphics these aren't the kinds of "loss" terms I'm familiar with. It isn't convex, symmetric and doesn't follow the triangle-inequality. I find it both surprising and interesting that one is able to solve large optimization problems with this setup.

Where does this term come from?

The full definition:

$$H(p, q) = H(p) + D_{KL}(p||q)$$

where,

$$H(p) = -\sum_i p(i) \log(p(i))$$

and

$$D_{KL} = \sum_i p(i) \log(\frac{p(i)}{q(i)})$$

$H(p)$ is Shannon's entropy and is a measure of entropy which measures uncertainty in a set of probabilities. The key point about Shannon's entropy is it is maximized when events are of uniform probability. It effectively tells us how unpredictable the outcome will be if you are trying to guess which event will happen. When events are certain, the entropy is zero. An illustrative example is a coin toss, where $x = 1$ for heads and $x = 0$ for tails. If p(1) = 1, then $H = -p(0)\log(p(0)) - p(1)\log(p(1)) = -0\log(0) - 1\log(1) = 0$. H is maximized when $p(1) = p(0) = \frac{1}{2}$.

D_{KL} is the Kullback-Leibler divergence, which I don't quite understand but the intuitive explanation is that it is a measure of how different one probability distribution is from the other. It's used to quantify how much information is lost when you approximate one distribution with another. If q is close to p then D_{KL} is small.

Because the entropy term $H(p)$ doesn't depend on the predicted distribution q, minimizing cross-entropy $H(p, q)$ is equivalent to minimizing D_{KL}.

Because the true distribution p is encoded like $p = [0, ...1, ..., 0]$ where the distribution takes on a value of 1 for label y_i, and the entropy $H(p)$ is zero for a delta distribution function (no uncertainty), the cross-entropy term simply reduces to minimizing D_{KL}.

Note however, that there is no restriction on q currently that makes it behave like a true probability distribution (e.g., the values don't sum to 1). To alleviate this, the popular choice for the predicted distribution is q is to apply the softmax function:

$$q = \frac{e^f_{y_i}}{\sum_j e^f_j}$$

which squashes a vector of values to be between zero and one. This formulation treats both p and q as probability distributions that we want to match and the cross-entropy objective wants the predicted distribution to have all of its mass on the correct answer.

$$P(y_i|x_i; W) = \frac{e^f_{y_i}}{\sum_j e^f_j}$$

can be interpreted as the (normalized) probability assigned to the correct label y_i given the image x_i parameterized by W. To see this, remember that the Softmax classifier interprets the scores inside the output vector f as the unnormalized log probabilities. Exponentiating these quantities therefore gives the (unnormalized) probabilities, and the division performs the normalization so that the probabilities sum to one.

From the probabilistic view, we want to solve q such that it is the Maximum Likelihood Estimation (MLE). A likelihood estimation is given by maximizing log likelihood:

$$\prod_i P(y_i|x_i; W)$$

In practice, we want to take a log of the above for numerical stability so we don't have to multiply many small numbers, this becomes:

$$\sum_i \log(P(y_i|x_i; W))$$

Since we know that log of values smaller than one yield negative values, we want to maximize this function, i.e., values closer to probability of 1. To frame this as a minimization problem, we can simply negate the

above to get:

$$-\sum_i \log(P(y_i|x_i;W))$$

Turns out this is equivalent to the cross-entropy term. So minimizing the cross entropy loss function is effectively minimizing the negative log likelihood or the maximum likelihood estimation.

MLE however doesn't have any prior knowledge of the expected distribution of the parameters and thus may overfit. To prevent this a regularization term is added such that the new loss function we try to minimize is:

$$L_i = -\log(P(y_i|x_i;W)g(W)) = -\log P(y_i|x_i;W) + \log(g(W))$$

If we want W to be normally distributed around zero (assume that g is Gaussian), then $\log(g(W)) \approx ||W||^2$. Such a regularization term also penalizes W to take on large values.

Further reading: [1, 2, 3].

References

[1] *Linear Classification*. URL: https://cs231n.github.io/linear-classify/.

[2] *MAP estimation as regularisation of MLE*. URL: https://stats.stackexchange.com/questions/367485/map-estimation-as-regularisation-of-mle.

[3] Chris Said. *Things that confused me about cross-entropy*. URL: https://chris-said.io/2020/12/26/two-things-that-confused-me-about-cross-entropy/.

Revision	Date	Description
1.0	October 6, 2020	Creation