LogSumExp Derivation

Benjamin Striner

January 27, 2018

1 LogSumExp Trick

Calculating $\log \sum_i e^{x_i}$ is frequent in machine learning and is referred to as LogSumExp. A common trick makes this function numerically stable.

$$\log \sum_i e^{x_i - C} = \log \sum_i \frac{e^{x_i}}{e^C} \tag{1}$$

$$= \log \frac{\sum_i e^{x_i}}{e^C} \tag{2}$$

$$= \log[\sum_i e^{x_i}] - \log e^C \tag{3}$$

$$\log \sum_i e^{x_i - C} = \log[\sum_i e^{x_i}] - C \tag{4}$$

$$\log[\sum_i e^{x_i}] = \log[\sum_i e^{x_i - C}] + C \tag{5}$$

We typically select $C = \max_j x_j$. That means the largest exponent we calculate is e^0, so our exponents never overflow.

2 Softmax

Typical softmax formulation is $f(x)_i = \frac{e^{x_i}}{\sum_j e^{x_j}}$.

3 Cross-entropy

Typical cross-entropy formulation is $L(p, q) = -\sum_i p_i \log q_i$.

4 Cross-entropy of Softmax

The cross-entropy of a softmax is therefore $L(p, f(x)) = -\sum_i p_i \log \frac{e^{x_i}}{\sum_j e^{x_j}}$. This calculation can be stabilized using the LogSumExp trick.

$$L(p, f(x)) = -\sum_i p_i \log \frac{e^{x_i}}{\sum_j e^{x_j}} \tag{6}$$

$$= -\sum_i p_i [\log(e^{x_i}) - \log(\sum_j e^{x_j})] \tag{7}$$

$$L(p, f(x)) = -\sum_i p_i [x_i - LogSumExp_j(x_j)] \tag{8}$$

5 Conclusion

You normally won't have to do the math yourself. Pytorch loss functions "BCEWithLogitsLoss" and "CrossEntropyLoss" will perform these calculations for you.

The important thing to remember is to not include the softmax or sigmoid output in your network when using these loss functions. The softmax or sigmoid are already included in the loss function and you don't want to accidentally apply them twice.

You can build a network without using the trick, but you may or may not end up getting NaN errors.