METL TP 1

Tientso Ning

Softmax

1. Clearly redefine the softmax function:

softmax(X) = $\frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}}$, where $x \in X$

2. Show that the softmax is invariant to constant offsets.

If we examine:

$$\text{softmax}(x) = \text{softmax}(x + c)$$

$$e^{x_i + c} = e^{x_i} e^{c}$$

$$\sum_{j=1}^{n} e^{x_j + c} = e^{c} \sum_{j=1}^{n} e^{x_j}$$

We notice that the value ec our additionally added constant value, will cancel out, showing that our function is invariant to this offset.

3. Please see code.

Gradient, Sigmoid, Forward, Backward

1. Define the sigmoid function and derive the gradients.

We define the sigmoid function as such:

$\sigma(X) = \frac{1}{1+e^{-x_i}}$, where $x \in X, X = \{x_1, x_2, \ldots, x_n\}$

And derive the gradients:

$$\begin{aligned}
\frac{d}{dx}\sigma(x) &= \frac{d}{dx}\left[\frac{1}{1+e^{-x}}\right] \\
&= \frac{d}{dx}\left(1+\mathrm{e}^{-x}\right)^{-1} \\
&= -\left(1+e^{-x}\right)^{-2}\left(-e^{-x}\right) \\
&= \frac{e^{-x}}{\left(1+e^{-x}\right)^{2}} \\
&= \frac{1}{1+e^{-x}} \cdot \frac{e^{-x}}{1+e^{-x}} \\
&= \frac{1}{1+e^{-x}} \cdot \frac{(1+e^{-x})-1}{1+e^{-x}} \\
&= \frac{1}{1+e^{-x}} \cdot \left(\frac{1+e^{-x}}{1+e^{-x}} - \frac{1}{1+e^{-x}}\right) \\
&= \frac{1}{1+e^{-x}} \cdot \left(1 - \frac{1}{1+e^{-x}}\right) \\
&= \sigma(x) \cdot (1-\sigma(x))
\end{aligned}$$

2. Derive the gradient to the inputs of the softmax function when cross-entropy loss is used.

We know that since we have a one-hot vector representation of the input vector, we can essentially evaluate loss as:

$-\log(q_y) = -\log\left(\frac{e^{x_y}}{\sum_j e^{x_j}}\right) = -x_y + \log\sum_j e^{x_j}$

Since the values where we have zero will cause the entropy calculation to zero out (due to the product with zero) and we only have to evaluate the second part with the log. We insert the softmax function, and apply the rule of log division, and rearrange and agree signs to get the representation ready for gradient calculation.

Then we can calculate the gradient for each xi:

$Loss = \nabla_{x_i} \log\sum_j e^{x_j} - \nabla_{x_i} x_y = \frac{1}{\sum_j e^{x_j}} \nabla_{x_i} \sum_j e^{x_j} - \nabla_{x_i} x_y = \frac{e^{x_i}}{\sum_j e^{x_j}} - \nabla_{x_i} x_y = q_i - \nabla_{x_i} x_y = q_i - 1(y = 1if y = i, elsey = 0)$

3. Derive the gradients when we have a one-hidden-layer neural network (perceptron).

We can calculate the gradient as such: - ...

4. How many parameters are there in this neural network?

Since there are H hidden units, we can consider the parameters to be the weights of that layer of the neural network, and say that there are H parameters.

5. - 7. Code Section

Please check the code for implementation for this section.