



STATS / DATA SCI 315

Softmax Derivatives
Information theory basics

Softmax and Derivatives



Recall squared loss case

$$\begin{aligned}\ell(y, \hat{y}) &= \frac{1}{2}(y - \hat{y})^2 \\ \hat{y} &= \mathbf{w}^\top \mathbf{x}\end{aligned}$$

$$\begin{aligned}\partial_{\hat{y}} \ell(y, \hat{y}) &= (\hat{y} - y) \\ \partial_{\mathbf{w}} \hat{y} &= \mathbf{x}\end{aligned}$$



Applying chain rule

$$\begin{aligned}\partial_{\mathbf{w}}\ell(y, \hat{y}) &= \mathbf{x}(\hat{y} - y) \\ &= \mathbf{x}(\mathbf{w}^{\top} \mathbf{x} - y)\end{aligned}$$



Cross-entropy in terms of the o's

$$\begin{aligned}l(\mathbf{y}, \hat{\mathbf{y}}) &= - \sum_{j=1}^q y_j \log \frac{\exp(o_j)}{\sum_{k=1}^q \exp(o_k)} \\&= \sum_{j=1}^q y_j \log \sum_{k=1}^q \exp(o_k) - \sum_{j=1}^q y_j o_j \\&= \log \sum_{k=1}^q \exp(o_k) - \sum_{j=1}^q y_j o_j.\end{aligned}$$



Gradient of loss w.r.t. \mathbf{o}

$$\partial_{o_j} l(\mathbf{y}, \hat{\mathbf{y}}) = \frac{\exp(o_j)}{\sum_{k=1}^q \exp(o_k)} - y_j = \text{softmax}(\mathbf{o})_j - y_j.$$



Gradient of loss w.r.t. weights

- Note that weights are in a $q \times d$ matrix \mathbf{W}
- Let \mathbf{w}_j^T be the j th row of \mathbf{W}
- Since $\mathbf{o} = \mathbf{W} \mathbf{x}$, we have $o_j = \mathbf{w}_j^T \mathbf{x}$, only \mathbf{w}_j affects o_j

$$\begin{aligned}\frac{\partial \ell}{\partial \mathbf{w}_j} &= \frac{\partial o_j}{\partial \mathbf{w}_j} \times \frac{\partial \ell}{\partial o_j} \\ &= \mathbf{x}(\hat{y}_j - y_j)\end{aligned}$$



Cross-entropy also works with soft observed labels

- So far we've assumed hard labels in the data but soft labels for our model output
- However, labels themselves can be soft
- Cross-entropy loss continues to make sense with soft observed labels too
- Interpretation: expected loss under hard labels sampled from the observed soft label

$$l(\mathbf{y}, \hat{\mathbf{y}}) = - \sum_{j=1}^q y_j \log \hat{y}_j.$$

Information Theory Basics



Entropy

- Suppose I need to encode an “alphabet” where symbol j occurs with prob $P(j)$
- The encoding needs to be in binary (0s and 1s)
- Intuitively, a good encoding with assign shorter codes to frequent symbols
- Turns out the optimal encoding needs these many bits/symbol on average :

$$-\sum_j P(j) \log_2 P(j)$$



Entropy

- If we use “nats” instead of bits, we get **entropy** expressed in natural logs
- 1 nats is approx. 1.44 bits
- The optimal encoding of symbol j will use approx. $\log(1/P(j))$ nats

$$H[P] = \sum_j -P(j) \log P(j).$$



Cross-entropy

- What is the true distribution was P but we think it is Q
- We will assign an encoding with length $-\log Q(j)$ to symbol j
- So our expected code length will be

$$H(P, Q) = - \sum_j P(j) \log Q(j)$$



KL divergence or relative entropy

- Note that if Q is not the same as P , we expect some overhead
- That is $H(P, Q) > H(P)$
- KL divergence, aka **relative entropy**, measures this excess

$$KL(P||Q) = H(P, Q) - H(P) = \sum_j P(j) \log \frac{P(j)}{Q(j)}$$