Neural Networks for Multi-Class Classification

Joey Shi

July 2020

1 Notation

1. X is a $n \times d$ feature matrix of real numbers, such that each row $x_i \in \mathbb{R}^d$ is an example.
2. y is an n-dimensional label vector, such that each $y_i \in \{1, 2, \ldots, k\}$ is a class label.
3. $h(z)$ is an activation function.
4. $W^{(l)}$ is a $k^{(l)} \times k^{(l-1)}$ weight matrix.
5. $b^{(l)}$ is a $k^{(l)}$-dimensional bias vector.

2 Maximum Likelihood Estimate

$$p(y_i \mid D, x_i) = \frac{\exp(z_{i,y_i}^{(L)})}{\sum_{c=1}^{k} \exp(z_{i,c}^{(L)})}$$

$p(y_i \mid D, x_i)$ is the probability that our model predicts y_i given example x_i and model parameters D. Assuming that our model has the above probability mass function, we optimize the model by finding the parameters D that maximizes $p(y_i \mid D, x_i)$. This is equivalent to finding the D that minimizes $-\log(p(y_i = c \mid D, x_i))$.

$$-\log(p(y_i = c \mid D, x_i)) = -z_{i,y_i}^{(L)} + \log\left(\sum_{c=1}^{k} \exp(z_{i,c}^{(L)})\right) \qquad \text{[Softmax Loss]}$$

3 Objective Function

Consider a feed-forward neural network. For an input layer defined by x_i, our hidden layers are defined by the activation vectors $a_i^{(l)}$ and our output layer is defined by $z_i^{(L)}$.

$$\begin{aligned} a_i^{(0)} &= x_i & z_i^{(1)} &= W^{(1)} x_i + b^{(1)} \\ a_i^{(1)} &= h(z^{(1)}) & z_i^{(2)} &= W^{(2)} a_i^{(1)} + b^{(2)} \\ &\vdots & &\vdots \\ a_i^{(L-1)} &= h(z^{(L-1)}) & z_i^{(L)} &= W^{(L)} a_i^{(L-1)} + b^{(L)} \end{aligned}$$

If the network is trained, given the example x_i, it would predict the class label $\hat{y}_i = \arg\max_c z_{i,c}^{(L)}$. Using softmax loss, our objective function becomes

$$f(W^{(1)}, b^{(1)}, \ldots, W^{(L)}, b^{(L)}) = \sum_{i=1}^{n} \left[-z_{i,y_i}^{(L)} + \log\left(\sum_{c=1}^{k} \exp\left(z_{i,c}^{(L)}\right)\right)\right]$$

4 Gradients

$$
\begin{aligned}
\frac{\partial f}{\partial w_{c,j}^{(L)}} &= \sum_{i=1}^{n} \left[-a_{i,j}^{(L-1)} I(y_i = c) + p(y_i = c \mid D, x_i) a_{i,j}^{(L-1)} \right] \\
&= \sum_{i=1}^{n} [p(y_i = c \mid D, x_i) - I(y_i = c)] a_{i,j}^{(L-1)} \\
&= \sum_{i=1}^{n} r_{i,c} a_{i,j}^{(L-1)} \\
&= \langle r^c, (a^j)^{(L-1)} \rangle \\
\frac{\partial f}{\partial W^{(L)}} &= R^T A^{(L-1)} \\
\frac{\partial f}{\partial W^{(L-1)}} &= \left[R W^{(L)} \circ h' \left(Z^{(L-1)} \right) \right]^T A^{(L-1)} \\
\frac{\partial f}{\partial b^{(L)}} &= \sum_{i=1}^{n} r_i \\
\frac{\partial f}{\partial b^{(L-1)}} &= \sum_{i=1}^{n} \left[R W^{(L)} \circ h' \left(Z^{(L-1)} \right) \right]_i^T
\end{aligned}
$$

5 Stochastic Gradient Descent Algorithm

Let $f(D)$ be a multivariable function. Given E epoches, B number of batches, and a learning rate α^t, we proceed with minimizing $f(D)$. Randomly initialize D^0. For each epoch, for each batch, compute the function gradient $f(D^t)$ from the batch. Then,

$$D^{t+1} = D^t - \alpha^t \nabla f(D^t)$$