Neural Networks for Multi-Class Classification

Joey Shi

July 2020

1 Abstract

In this paper, we will be formalizing the process of training a neural network classification model. We will be covering the softmax loss function, computing the gradient of the network, and gradient descent.

2 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 $y_i \in \{1, 2, \ldots, k_l\}$ is a class label.
3. $h(z)$ is a non-linear function. For our purposes, we will say it 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.

3 Maximum Likelihood Estimate

Consider the following probability distribution:

$$p(y_i \mid D, x_i) = \frac{\exp(z_{i,y_i}^{(L)})}{\sum_{c=1}^{k} \exp(z_{i,c}^{(L)})}, \qquad [z_i^{(L)} \text{ is defined in section 3}]$$

$p(y_i \mid D, x_i)$ is the probability that our model predicts y_i given example x_i and model parameters D. We will develop a classification model with the above probability mass function. We can optimize the model by finding the parameters D that maximizes $p(y \mid D, X)$. Assuming that each example is independent, we can equivalently minimize

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

4 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

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

5 Gradients of the Objective Function

Theorem 1. *Let $A^{(l)}$ be a matrix such that each row is an activation vector $a_i^{(l)}$ and let $Z^{(l)}$ be defined similarly. Define a new matrix R by the recursion relation $R^{(l-1)} = R^{(l)}W^{(l)} \circ h'\left(Z^{(l-1)}\right)$ with a base case of $R^{(L)}$, where $r_{ic}^{(l)} = p(y_i = c \mid D, x_i) - I(y_i = c)$. Then the derivatives of the weights and biases of our network are given by*

$$\frac{\partial f}{\partial W^{(l)}} = \left(R^{(l)}\right)^T A^{(l-1)} \qquad \frac{\partial f}{\partial b^{(l)}} = \sum_{i=1}^{n} r_i^{(l)}$$

Example:

$$\begin{aligned} \frac{\partial f}{\partial w_{c,j}^{(L)}} &= \sum_{i=1}^{n} \left[-a_{i,j}^{(L-1)} I(y_i = c) + p(y_i = c \mid D, x_i) a_{i,j}^{(L-1)} \right] \\ &= \sum_{i=1}^{n} [p(y_i = c \mid D, x_i) - I(y_i = c)] a_{i,j}^{(L-1)} \\ &= \sum_{i=1}^{n} r_{i,c}^{(L)} a_{i,j}^{(L-1)} \\ \frac{\partial f}{\partial W^{(L)}} &= \left(R^{(L)}\right)^T A^{(L-1)} \\ \frac{\partial f}{\partial b^{(L)}} &= \sum_{i=1}^{n} r_i \end{aligned}$$

6 Stochastic Gradient Descent Algorithm

Let $f(D)$ be a multivariable loss function for a model. 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, sample $\lfloor n/B \rfloor$ training examples without replacement and compute the function gradient $f(D^t)$ from the batch. Then,

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

We will use the parameters D^E to make predictions for the model.