The Basics of Neural Networks

Greg Strabel

March 6, 2023

1 Assumptions

A neural network is a directed graph $G = (V, A)$ composed of nodes, V, and arrows, A. Assume that the nodes of the network have been indexed by a subscript $k \in \{1, ..., K\}$ and that the output of the k^{th} neuron is $o_k = \sigma_k(\psi_k)$ where $\sigma_k : \mathbb{R} \to \mathbb{R}$ is an activation function and

$$\psi_k = \sum_{l:(l,k)\in A} w_{kl} o_l + \beta_k \tag{1}$$

where $w_{kl} \in \mathbb{R}$ is the weight applied to the input $o_l \in \mathbb{R}$ and $\beta_k \in \mathbb{R}$ is a bias term. If the k^{th} neuron is in the first layer, then the o_l are just the inputs to the neural network.

2 Back-propogation

Neural networks are typically trained using an algorithm called back-propogation. Given a loss function $L : \mathbf{Z} \times \mathbf{Y} \to \mathbb{R}$, where $\mathbf{Z}$ is the co-domain of the network, the chain rule of differentiation implies:

$$\frac{\partial L}{\partial w_{kj}} = \frac{\partial L}{\partial \psi_k}\frac{\partial \psi_k}{\partial w_{kj}} = \lambda_k o_j \tag{2}$$

and

$$\frac{\partial L}{\partial \beta_k} = \frac{\partial L}{\partial \psi_k} = \lambda_k \tag{3}$$

where

$$\lambda_k = \frac{\partial L}{\partial \psi_k} = \sum_{l:(k,l)\in A} \frac{\partial L}{\partial \psi_l}\frac{\partial \psi_l}{\partial o_k}\frac{\partial o_k}{\partial \psi_k} = \sum_{l:(k,l)\in A} \lambda_l w_{lk} \sigma_k'(\psi_k) \tag{4}$$

This is the back-propagation equation which we can use to work backwards through the network to update parameters. Suppose we have a set of data $\{(x_i, y_i)\}_{i=1...m}$. Starting at terminal node l, we have:

$$\lambda_{li} = \frac{\partial L_i}{\partial \psi_{li}} = \frac{\partial L_i}{\partial z_{li}}\sigma_l'(\psi_{li}) \tag{5}$$

which suggests the updates:

$$w_{kl} \leftarrow w_{kl} - \eta \sum_{i=1}^{m} \frac{\partial L_i}{\partial z_{li}} \sigma_l' \left(\psi_{li}\right) o_{li} = w_{kl} - \eta \sum_{i=1}^{m} \lambda_{li} o_{li} \tag{6}$$

and

$$\beta_l \leftarrow \beta_l - \eta \sum_{i=1}^{m} \frac{\partial L_i}{\partial z_{li}} \sigma_l' \left(\psi_{li}\right) = \beta_l - \eta \sum_{i=1}^{m} \lambda_{li} \tag{7}$$

where η is the learning rate. Proceed recursively with the updates:

$$w_{kj} \leftarrow w_{kj} - \eta \sum_{i=1}^{m} \frac{\partial L_i}{\partial w_{kj}} = w_{kj} - \eta \sum_{i=1}^{m} \lambda_{ki} o_{ji} \tag{8}$$

and

$$\beta_k \leftarrow \beta_k - \eta \sum_{i=1}^{m} \frac{\partial L_i}{\partial \beta_k} = \beta_k - \eta \sum_{i=1}^{m} \lambda_{ki} \tag{9}$$

using equation (4) to produce the summands.

3 Dropout

Dropout is a method for preventing overfitting. For each mini-batch during training, rather than using the linear predictor

$$\psi_k = \sum_{l:(l,k)\in A} w_{kl} o_l + \beta_k \tag{10}$$

in the k^{th} neuron we instead use

$$\psi_k = \sum_{l:(l,k)\in A} w_{kl} r_{kl} o_l + \beta_k \tag{11}$$

where $r_{kl} \sim Bernoulli(p_{kl})$ and $p_{kl} \in [0, 1]$. This has the effect of changing the chain rule to:

$$\frac{\partial L}{\partial w_{kj}} = \frac{\partial L}{\partial \psi_k} \frac{\partial \psi_k}{\partial w_{kj}} = \lambda_k o_j r_{kj} \tag{12}$$

where now

$$\lambda_k = \frac{\partial L}{\partial \psi_k} = \sum_{l:(k,l)\in A} \frac{\partial L}{\partial \psi_l} \frac{\partial \psi_l}{\partial o_k} \frac{\partial o_k}{\partial \psi_k} = \sum_{l:(k,l)\in A} \lambda_l w_{lk} r_{lk} \sigma_k' \left(\psi_k\right) \tag{13}$$

Clearly if $r_{kj} = 0$ then $\frac{\partial L}{\partial w_{kj}} = 0$ and so w_{kj} is not updated for this mini-batch. At test time, rather than use (10), we use:

$$\psi_k = \sum_{l:(l,k)\in A} w_{kl} p_{kl} o_l + \beta_k \tag{14}$$

4 Batch Normalization

One complication that arises in training deep neural networks is the fact that the distribution of the inputs to each layer change during training as parameters from previous layers are updated. This change is known as internal covariate shift. Ioffe and Szegedy (2015) introduced Batch Normalization to normalize layers' inputs during mini-batch training. There is more than one way to normalize; in the discussion below I deviate slightly from Ioffe and Szegedy (2015). Suppose we have a mini-batch $\{x_i, y_i\}_{i=1\ldots m}$. During the forward pass, rather that using

$$o_{ki} = \sigma_k \left(\psi_{ki}\right) \tag{15}$$

we use

$$o_{ki} = \sigma_k \left(\widehat{\psi}_{ki}\right) \tag{16}$$

where

$$\widehat{\psi}_{ki} = \gamma_k \frac{\psi_{ki} - \widehat{\mu}_k}{\sqrt{\widehat{\nu}_k^2 + \epsilon}} + \theta_k \tag{17}$$

$$\widehat{\mu}_k = \frac{1}{m} \sum_{j=1}^{m} \psi_{kj} \tag{18}$$

$$\widehat{\nu}_k^2 = \frac{1}{m} \sum_{j=1}^{m} \left(\psi_{kj} - \widehat{\mu}_k\right)^2 \tag{19}$$

and $\gamma_k, \delta_k \in \mathbb{R}$ are additional parameters to learn. During the backwards pass we treat $\widehat{\mu}_k$ and $\widehat{\nu}_k^2$ as fixed. Batch Normalization introduces a new term to the equations of back-propogation:

$$\frac{\partial L_i}{\partial \widehat{\psi}_{ki}} = \sum_{l:(k,l)\in A} \frac{\partial L_i}{\partial \psi_{li}} \frac{\partial \psi_{li}}{\partial o_{ki}} \frac{\partial o_{ki}}{\partial \widehat{\psi}_{ki}} = \sum_{l:(k,l)\in A} \lambda_{li} w_{lk} \sigma_k' \left(\widehat{\psi}_{ki}\right) \tag{20}$$

so that

$$\lambda_{ki} = \frac{\partial L_i}{\partial \psi_{ki}} = \frac{\partial L_i}{\partial \widehat{\psi}_{ki}} \frac{\partial \widehat{\psi}_{ki}}{\partial \psi_{ki}} = \sum_{l:(k,l)\in A} \lambda_{li} w_{lk} \sigma_k' \left(\widehat{\psi}_{ki}\right) \frac{\gamma_k}{\sqrt{\widehat{\nu}_k^2 + \epsilon}} \tag{21}$$

The parameter updates for w_{kj} and β_k remain the same as those in (8) and (9) except that now we use (21) to calculate λ_{ki}. Additionally, we need to update the parameters γ_k and θ_k. The chain rule gives us:

$$\frac{\partial L_i}{\partial \gamma_k} = \frac{\partial L_i}{\partial \widehat{\psi}_{ki}} \frac{\partial \widehat{\psi}_{ki}}{\partial \gamma_k} = \sum_{l:(k,l)\in A} \lambda_{li} w_{lk} \sigma_k' \left(\widehat{\psi}_{ki}\right) \frac{\psi_{ki} - \widehat{\mu}_k}{\sqrt{\widehat{\nu}_k^2 + \epsilon}} \tag{22}$$

and

$$\frac{\partial L_i}{\partial \theta_k} = \frac{\partial L_i}{\partial \widehat{\psi}_{ki}} \frac{\partial \widehat{\psi}_{ki}}{\partial \theta_k} = \sum_{l:(k,l)\in A} \lambda_{li} w_{lk} \sigma_k' \left(\widehat{\psi}_{ki} \right) \tag{23}$$

so that updates take the form:

$$\gamma_k \leftarrow \gamma_k - \eta \sum_{i=1}^{m} \sum_{l:(k,l)\in A} \lambda_{li} w_{lk} \sigma_k' \left(\widehat{\psi}_{ki} \right) \frac{\psi_{ki} - \widehat{\mu}_k}{\sqrt{\widehat{\nu}_k^2 + \epsilon}} \tag{24}$$

and

$$\theta_k \leftarrow \theta_k - \eta \sum_{i=1}^{m} \sum_{l:(k,l)\in A} \lambda_{li} w_{lk} \sigma_k' \left(\widehat{\psi}_{ki} \right) \tag{25}$$

5 Variations of Stochastic Gradient Descent

5.1 Implicit SGD

5.2 Momentum

For the first mini-batch, we use the updates in (3) and (8). This results in a change to the parameters of Δw and $\Delta \beta$. Starting with the second mini-batch, we now follow the updates:

$$\Delta w_{kj} \leftarrow \alpha \Delta w_{kj} - \eta \frac{\partial L}{\partial w_{kj}} \qquad \textit{and} \qquad \Delta \beta_k \leftarrow \alpha \Delta \beta_k - \eta \frac{\partial L}{\partial \beta_k} \tag{26}$$

followed by

$$w \leftarrow w + \Delta w \qquad \textit{and} \qquad \beta \leftarrow \beta + \Delta \beta \tag{27}$$