Neural Network implementation

Martin Lazo

April 2022

1 Notation

$\odot$	Hadamard (element-wise) product
$\mathbf{v}$	Vector, vectors are denoted in lower case and bold
$\mathbf{M}$	Matrix, matrices are denoted in upper case and bold
$\mathbf{v}_i$	ith element of a vector $\mathbf{v}$
$\mathbf{M}_{i,j}$	Element in the ith row and jth column of a matrix $\mathbf{M}$
$\mathbf{M}_{i,}$	ith row of a matrix $\mathbf{M}$
$\mathbf{M}_{,i}$	ith column of a matrix $\mathbf{M}$
$\mathbf{o}_i$	Vector which preserves only the ith element of a vector as a scalar

$$\mathbf{o}_3\mathbf{v} = \begin{pmatrix} 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \end{pmatrix} = c$$

$\mathbf{O}_i$	Square matrix which preserves only the ith element of a vector

$$\mathbf{O}_3\mathbf{v} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ c \end{pmatrix}$$

Hidden layers are indexed from 0 to n with n being the output layer and 0 being the input layer.

$\mathbf{X}$	Features
$\mathbf{Y}$	Labels
$\mathbf{x}$	Features data point
$\mathbf{y}$	Labels data point
$\mathbf{b}^m$	Bias vector of the mth layer
$\mathbf{W}^m$	Weights matrix of the mth layer
ϕ^m	Activation function of the mth layer, applied element-wise
$\phi^{m\prime}$	Derivative of the activation function of the mth layer

2 Backpropagation calculus

Neural network is a function $\hat{y} : \mathbb{R}^a \to \mathbb{R}^b$ where a and b are positive integers. Loss function ℓ_2 for a set $S = \{(\mathbf{y}_n, \mathbf{x}_n) : 1 \leq n \leq N\}$:

$$\ell_2 = \frac{1}{N} \sum_{\mathbf{y},\mathbf{x} \in S} (\hat{y}(\mathbf{x}) - \mathbf{y})^T (\hat{y}(\mathbf{x}) - \mathbf{y})$$

The aim is to minimize ℓ_2 with respect to weights and biases. This is done by iteratively updating weights using gradient descent:

$$(\mathbf{W}^m_{i,j})_{n+1} = (\mathbf{W}^m_{i,j})_n - \alpha \frac{\partial \ell_2}{\partial (\mathbf{W}^m_{i,j})_n}$$

$$(\mathbf{b}^m_i)_{n+1} = (\mathbf{b}^m_i)_n - \alpha \frac{\partial \ell_2}{\partial (\mathbf{b}^m_i)_n}$$

where α is the learning rate.

$$\frac{\partial \ell_2}{\partial \mathbf{W}^m_{i,j}} = \frac{2}{N} \sum_{\mathbf{y},\mathbf{x} \in S} (\hat{y}(\mathbf{x}) - \mathbf{y})^T \frac{\partial \hat{y}}{\partial \mathbf{W}^m_{i,j}}(\mathbf{x})$$

$$\frac{\partial \ell_2}{\partial \mathbf{b}^m_i} = \frac{2}{N} \sum_{\mathbf{y},\mathbf{x} \in S} (\hat{y}(\mathbf{x}) - \mathbf{y})^T \frac{\partial \hat{y}}{\partial \mathbf{b}^m_i}(\mathbf{x})$$

Neural network output for an input $\mathbf{x}$:

$$\hat{y}(\mathbf{x}) = \phi^n(\mathbf{W}^n \phi^{n-1}(..\phi^0(\mathbf{W}^0 \mathbf{x} + \mathbf{b}^0)..) + \mathbf{b}^n)$$

Partial derivatives w.r.t. weights:

$$\frac{\partial \hat{y}}{\partial \mathbf{W}^m_{i,j}}(\mathbf{x}) = \phi^{n\prime}(..) \odot .. \odot (\mathbf{W}^{m+2}(\phi^{m+1\prime}(..) \odot (\mathbf{W}^{m+1}(\mathbf{O}_i \phi^{m\prime}(..) \mathbf{o}_j \phi^{m-1}(..)))))$$

$$\frac{\partial \hat{y}}{\partial \mathbf{W}^0_{i,j}}(\mathbf{x}) = \phi^{n\prime}(..) \odot .. \odot (\mathbf{W}^2(\phi^{1\prime}(..) \odot (\mathbf{W}^1(\mathbf{O}_i \phi^{0\prime}(..) \mathbf{x}_j))))$$

$$\frac{\partial \hat{y}}{\partial \mathbf{W}^n_{i,j}}(\mathbf{x}) = \mathbf{O}_i \phi^{n\prime}(..) \mathbf{o}_j \phi^{n-1}(..)$$

Partial derivatives w.r.t. biases:

$$\frac{\partial \hat{y}}{\partial \mathbf{b}^m_i}(\mathbf{x}) = \phi^{n\prime}(..) \odot .. \odot (\mathbf{W}^{m+2}(\phi^{m+1\prime}(..) \odot (\mathbf{W}^{m+1}(\mathbf{O}_i \phi^{m\prime}(..)))))$$

$$\frac{\partial \hat{y}}{\partial \mathbf{b}^0_i}(\mathbf{x}) = \phi^{n\prime}(..) \odot .. \odot (\mathbf{W}^2(\phi^{1\prime}(..) \odot (\mathbf{W}^1(\mathbf{O}_i \phi^{0\prime}(..)))))$$

$$\frac{\partial \hat{y}}{\partial \mathbf{b}^n_i}(\mathbf{x}) = \mathbf{O}_i \phi^{n\prime}(..)$$