INFO 7375 - Neural Networks & AI

Homework to Chapter - 5

Submitted By:

Abdul Haseeb Khan
NUID: 002844724
khan.abdulh@northeastern.edu

Why is a hidden layer needed?

A hidden layer is needed because the Perceptron provides only binary linear separation and requires categories to be linearly separable, so adding an intermediate layer introduces nonlinearity that improves classification capabilities when patterns are more complex or linearly inseparable. Neural networks also need multiple layers to learn complex representations, with each layer extracting increasingly abstract features that capture intricate patterns and relationships that a single linear layer cannot model.

The Perceptron augmented with one or more hidden layers is referred to as a neural network, emphasizing that the added layer changes the model's expressive power beyond linear separation.

Describe in detail forward propagation and back propagation calculations for a neural network with one hidden layer.

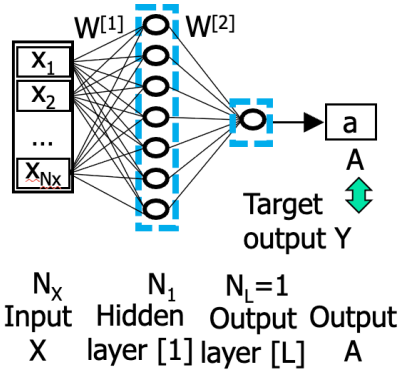
Notations:

N_x = Number of Neurons in Input Layer
N_1 = Number of Neurons in First Layer or Hidden Layer
N_L = Number of Neurons in Last Layer
A = Output
Y = Target Output
$W^{[1]}$ = Weight Matrix received by First Layer
$W^{[L]}$ = Weight Matrix received by Last Layer
J = Loss Function/ Cost Function

A network with one hidden layer uses matrices W of size $N_1 \times N_x$ and W of size $N_2 \times N_1$, where N_1 is the size of the hidden layer, N_x is the input dimension, and $N_2 = 1$ for a single output neuron, with biases typically the same for each neuron in a given layer. The weight indices $w_{kj}^{[s]}$ denote the transmission weight in layer s from neuron j in the sending layer s−1 to neuron k in the receiving layer s, and with only one neuron in the output layer the first index in $W^{[L]}$ can be omitted by writing $w_k^{[L]}$ instead of $w_{1k}^{[L]}$. Parameters W, b, W, and b stay the same for all training samples, while X(p), Z(p), A(p), z(p), a(p) ≡ a(p), and y(p) vary with the sample index p.

Forward Propagation

For one hidden layer, the forward pass is $Z = W^{[1]}X + E_{N_1} b^{[1]} E_M^T$ and $A^{[1]} = f^{[1]}(Z^{[1]})$, followed by $Z = W^{[L]}A^{[1]} + b^{[L]}E_M^T$ and $\mathrm{A} \equiv A^{[L]} = f^{[L]}(Z^{[L]})$. With a single output neuron $N_L = 1$ the vector E_{N_L} can be omitted in all expressions at the output layer, which simplifies Z to the scalar-bias broadcast form.



In classification networks, the hidden layer activation is usually f(z) = ReLU(z) and the output layer uses f(z) = σ(z), where ReLU is f(z) = max(z, 0) and the sigmoid is a smooth differentiable function convenient for the output layer in logistic regression.

For a set of M samples in matrix form, $Z^{[1]} = W^{[1]}X + E_{N_1} b^{[1]} E_M^T$, $A^{[1]} = f^{[1]}(Z^{[1]})$, $Z^{[L]} = W^{[L]}A^{[1]} + b^{[L]}E_M^T$ and $\mathrm{A} \equiv A^{[L]} = f^{[L]}(Z^{[L]})$ hold with columns corresponding to samples p = 1,...,M, and A is the calculated output of the network. The flow by layer can be summarized as input X to hidden aggregated signal Z, to hidden output A, then to output aggregated signal Z, and finally to output $A \equiv A$, with the table listing $Z = W^{[1]}X + E_{N_1} b^{[1]} E_M^T$, $\mathrm{A} = \mathrm{ReLU}(Z^{[1]})$, $Z = W^{[L]}A^{[1]} + b^{[L]}E_M^T$, and $\mathrm{A} = \sigma(Z^{[L]})$.

	First (hidden) layer	Second (output) layer
Input	X	$A^{[1]}$
Aggregated signal	$Z^{[1]} = W^{[1]}X + E_{N_1} b^{[1]} E_M^T$	$Z^{[L]} = W^{[L]}A^{[1]} + b^{[L]}E_M^T$
Output	$A^{[1]} = f^{[1]}(Z^{[1]}) = \mathrm{ReLU}(Z^{[1]})$	$A^{[L]} = f^{[L]}(Z^{[L]}) = \sigma(Z^{[L]})$

Loss function ($J(A, Y)$)

The cross-entropy cost function used is $J = -\frac{1}{M}\left(Y \ln A^T + (E_M^T - Y)\ln(E_M^T - A)^T\right)$, which is convex in w and b and has a single global minimum, making it suitable for gradient descent optimization. Backpropagation is described as an algorithm for supervised learning

that calculates the gradient of the loss with respect to weights and bias using gradient descent with this cross-entropy log loss function.

Backpropagation

At the output layer, the error signal is $\delta Z = A - Y$, and gradients are $\delta W^{[2]} = \delta Z^{[2]} A^{[1]T}$ and $\delta b^{[2]} = \delta Z^{[2]} E_M$ with sample-averaging, matching the schematic derivation. With sigmoid activation at the output, $f(z) = \sigma(z)$ with $\frac{\partial \sigma(z)}{\partial z} = \sigma(z)(1 - \sigma(z))$ is noted in the layer 2 derivation, though $\delta Z = A - Y$ encapsulates the derivative for cross-entropy with sigmoid.

The hidden-layer error is obtained by the chain rule as $\delta Z^{[1]} = W^{[2]T} \delta Z^{[2]}$, and for ReLU this derivative is 1 for $z \geq 0$ and 0 for $z < 0$, so the element wise product gates error flow where Z is positive. The hidden-layer gradients are $\delta W^{[1]} = \delta Z^{[1]} X^T$ and $\delta b^{[1]} = E_{N_1}^T \delta Z^{[1]} E_M$ in the schematic expression, corresponding to sample-averaged weight and bias updates.