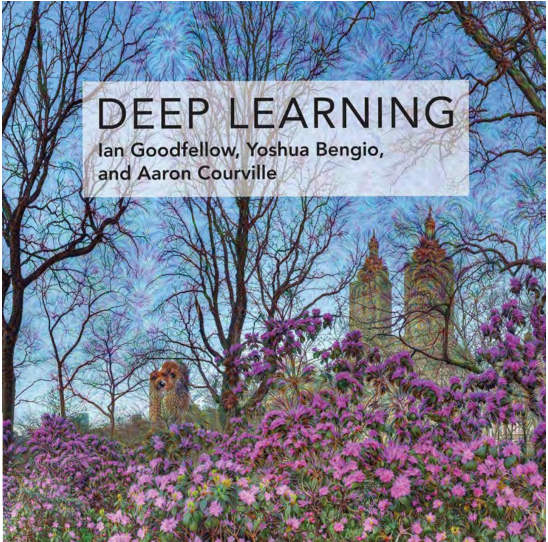
DSO530 Statistical Learning Methods

Lecture 10 : Neural Networks (Optional)

Dr. Xin Tong
Department of Data Sciences and Operations
Marshall School of Business
University of Southern California
xint@marshall.usc.edu

Deep Learning

- Image Classification
- Video Classification
- Natural Language Processing (NLP)



Single Layer Neural Networks

- Using $X = (X_1, X_2, \cdots, X_p)$ to predict Y (Numerical)
- Two steps

1. The K activation $A_k, k = 1, \cdots, K$ in the **hidden layer** are functions of the **input features**: $X_1, \cdots, X_p$.

$$A_k = h_k(X) = g(w_{k0} + \sum_{j=1}^{p} w_{kj} X_j)$$

   Here: $w_{k0}, k = 1, \cdots, K$ and $w_{kj}, k = 1, \cdots, K, j = 1, \cdots, p$ are called **weights**, to be estimated from the training data. $g(\cdot)$ is the so-called **activation function** which is specified in advance.

2. Then, these K activations from the **hidden layer** are fed into the **output layer**, resulting in

$$f(X) = \beta_0 + \sum_{k=1}^{K} \beta_k A_k.$$

Single Layer Neural Networks

$$
\begin{aligned}
\hat{Y} = f(X) &= \beta_0 + \sum_{k=1}^{K} \beta_k h_k(X) && (1) \\
&= \beta_0 + \sum_{k=1}^{K} \beta_k g(w_{k0} + \sum_{j=1}^{p} w_{kj} X_j) && (2)
\end{aligned}
$$

Single Layer Neural Networks

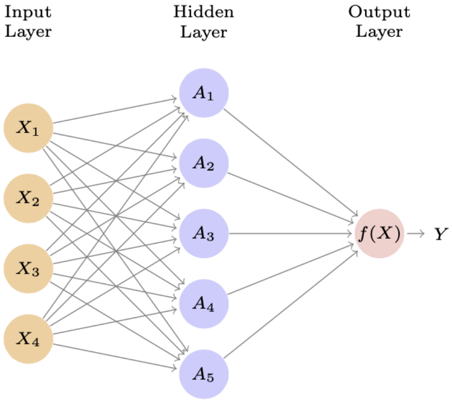


FIGURE 10.1. *Neural network with a single hidden layer. The hidden layer computes activations $A_k = h_k(X)$ that are nonlinear transformations of linear combinations of the inputs $X_1, X_2, \ldots, X_p$. Hence these A_k are not directly observed. The functions $h_k(\cdot)$ are not fixed in advance, but are learned during the training of the network. The output layer is a linear model that uses these activations A_k as inputs, resulting in a function $f(X)$.*

Activation Function

- **Sigmoid** activation function:

$$g(z) = \frac{e^z}{1 + e^z}.$$

- **ReLU** (rectified linear unit) activation function:

$$g(z) = (z)_+ = \begin{cases} 0 \text{ if } z < 0, \\ z \text{ otherwise.} \end{cases}$$

Fitting a Neural Network

- For numerical response, typically squared-error loss is used, so we choose the minimize the RSS

$$RSS = \sum_{i=1}^{n} [y_i - f(x_i)]^2,$$

where

$$f(x_i) = \beta_0 + \sum_{k=1}^{K} \beta_k g(w_{k0} + \sum_{j=1}^{p} w_{kj} x_{ij}) \tag{3}$$

- Denote all parameters as θ, then we have

$$R(\theta) = \sum_{i=1}^{n} [y_i - f_\theta(x_i)]^2.$$

Fitting a Neural Network (cont):

1. Get an initial estimate θ as θ_0, and set $m = 0$.
2. Iterate until $R(\theta)$ fails to decrease:
   a. Compute the gradient at θ^m:

   $$\nabla R(\theta^m) = \frac{\partial R(\theta)}{\partial \theta}|_{\theta=\theta_m}.$$

   b. Move θ a little in the opposite direction:

   $$\theta^{m+1} \leftarrow \theta^m - \rho \nabla R(\theta^m).$$

- Here, ρ is the **learning rate**.

Multilayer Neural Network

Handwritten digits classification:

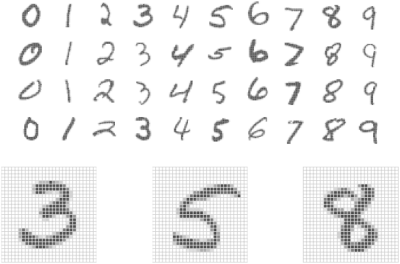


FIGURE 10.3. *Examples of handwritten digits from the* MNIST *corpus. Each grayscale image has* 28×28 *pixels, each of which is an eight-bit number (0–255) which represents how dark that pixel is. The first 3, 5, and 8 are enlarged to show their 784 individual pixel values.*

- X: $28 \times 28 = 784$ pixels
- Y: digits 0 - 9 (10 classes) $->$ Convert to 10 dummy variables $Y_0, Y_1, \cdots, Y_9$

Architecture of Multilayer Neural Network

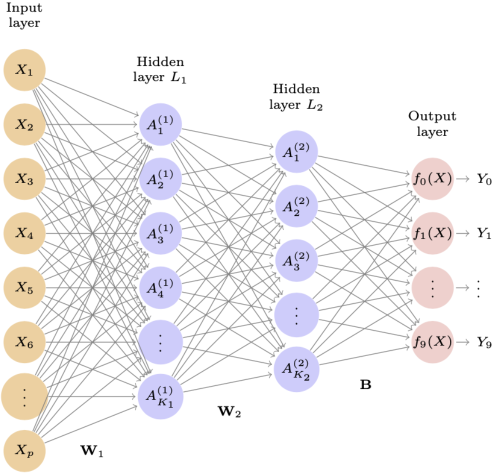


FIGURE 10.4. *Neural network diagram with two hidden layers and multiple outputs, suitable for the* MNIST *handwritten-digit problem. The input layer has $p = 784$ units, the two hidden layers $K_1 = 256$ and $K_2 = 128$ units respectively, and the output layer 10 units. Along with intercepts (referred to as* biases *in the deep-learning community) this network has 235,146 parameters (referred to as* weights*).*

Formulation

- First hidden layer (for $k = 1, \cdots, K_1$, here, $K_1 = 256$)

$$A_k^{(1)} = h_k^{(1)}(X) = g(w_{k0}^{(1)} + \sum_{j=1}^{p} w_{kj}^{(1)} X_j).$$

- Second hidden layer (for $l = 1, \cdots, K_2$, here, $K_2 = 128$)

$$A_l^{(2)} = h_l^{(2)}(X) = g(w_{l0}^{(2)} + \sum_{j=1}^{K_1} w_{lj}^{(2)} A_k^{(1)}).$$

- Output layer (for $m = 0, 1, \cdots, 9$)

$$Z_m = \beta_{m0} + \sum_{l=1}^{K_2} \beta_{ml} A_l^{(2)}.$$

$$f_m(x) = \frac{e^{Z_m}}{\sum_{l=0}^{9} e^{Z_l}}.$$

Number of Parameters:

- First hidden layer: $256 \times (784 + 1) = 200960$
- Second hidden layer: $128 \times (256 + 1) = 32896$
- Final layer: $10 \times (128 + 1) = 1290$
- Total number of parameters: $200960 + 32896 + 1290 = 235146$

Activation Function and Loss Function

- **softmax** activation function:

$$f_m(x) = P(Y = m|X = x) = \frac{e^{Z_m}}{\sum_{l=0}^{9} e^{Z_l}}.$$

- Loss Function: negative multinomial log-likelihood (cross-entropy)

$$-\sum_{i=1}^{n} \sum_{m=0}^{9} y_{im} \log(f_m(x_i)).$$

Other Deep Learning Structures

- Convolutional neural networks (CNN)
- Recurrent neural networks (RNN)