Introduction to Statistical Machine Learning with Applications in Econometrics

Deep Learning

Instructor: Ma, Jun

Renmin University of China

November 22, 2022

Deep learning

- The neural network model became popular in the 1980s.
- Re-emerged around 2010 as Deep Learning.
- Part of success due to vast improvements in computing power, larger training sets, and software.
- Response Y and p different predictors $X = \left(X_1, X_2, ..., X_p\right)^\top$. We are interested in estimating $f(x) = \mathrm{E}[Y \mid X = x]$.
- Our training data consist of $\{(X_1, Y_1), (X_2, Y_2), ..., (X_n, Y_n)\}$, where $X_i = \left(X_{1,i}, X_{2,i}, ..., X_{p,i}\right)^\top$.
- The neural network model is a nonlinear model: $f(X) \approx m(X, \beta)$ for some optimal coefficients

  $$\beta = \underset{\theta}{\operatorname{argmin}} \mathrm{E}\left[(Y - m(X, \theta))^2\right] = \underset{\theta}{\operatorname{argmin}} \mathrm{E}\left[(f(X) - m(X, \theta))^2\right]$$

  to be estimated.
  - $m(X, \beta)$ is nonlinear in parameters.
  - Much more computational burden.

Single layer neural network

- Let $\theta = (b, w)$, $b = (b_0, b_1, ..., b_K)$ and

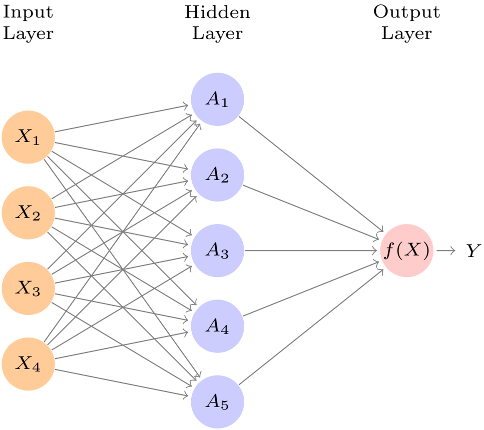
  $$w = \left(w_{10}, w_{11}, ..., w_{1p}, w_{20}, w_{21}, ..., w_{2p}, ..., w_{K0}, w_{K1}, ..., w_{Kp}\right).$$

- The single layer neural network model:

$$\begin{aligned} m(X, \theta) &= b_0 + \sum_{k=1}^{K} b_k h_k(X) \\ &= b_0 + \sum_{k=1}^{K} b_k g\left(w_{k0} + \sum_{j=1}^{p} w_{kj} X_j\right). \end{aligned}$$

- The model is fit by nonlinear least squares:

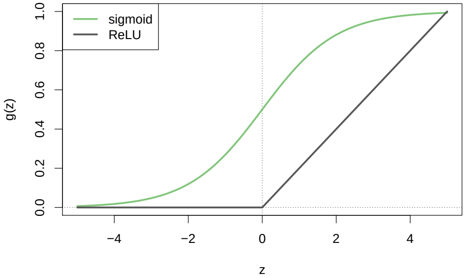
$$\min_{a} \sum_{i=1}^{n} \left(Y_i - m(X_i, \theta)\right)^2.$$



ISL Figure 10.1

- $A_k = h_k(X) = g\left(w_{k0} + \sum_{j=1}^{p} w_{kj} X_j\right)$ are called the activations in the hidden layer. These are analogous to neurons in a human brain.
- g is called the activation function.

Nonlinear activation



ISL Figure 10.2

- Popular are the sigmoid and rectified linear.
- Having a nonlinear activation function allows the model to capture complex nonlinearities and interaction effects.
  - E.g.,
$$\frac{1}{4}(X_1 + X_2)^2 - \frac{1}{4}(X_1 - X_2)^2 = X_1 X_2.$$
  - Sum of two nonlinear transformations of linear functions can give us an interaction.

Fitting the model: gradient descent

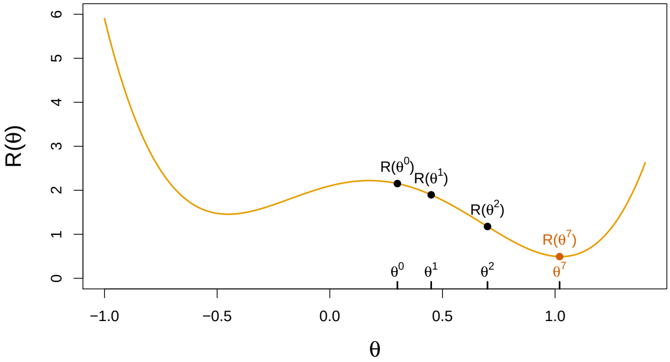
- The minimization problem:

$$\min_{b,w} \frac{1}{2} \sum_{i=1}^{n} \left\{ Y_i - \left(b_0 + \sum_{k=1}^{K} b_k g \left(w_{k0} + \sum_{j=1}^{p} w_{kj} X_{j,i} \right) \right) \right\}^2$$

  is non-convex. It may have multiple local minima.
- Denote

$$R(\theta) = \frac{1}{2} \sum_{i=1}^{n} \left(Y_i - m(X_i, \theta)\right)^2 .$$

- We apply the gradient descent method.
  - Start with an initial guess: θ^0 for the true minimizer;
  - Find a vector δ that reflects a small change such that $\theta^{t+1} = \theta^t + \delta$ reduces the objective: $R\left(\theta^{t+1}\right) < R\left(\theta^t\right)$;
  - We pick $\delta = -\rho \nabla R\left(\theta^t\right)$, where $\nabla R\left(\theta^t\right) = \left.\frac{\partial R(\theta)}{\partial \theta}\right|_{\theta=\theta^t}$ is the gradient and $\rho > 0$ is the learning rate;
  - The algorithm returns $\left(\theta^t, R\left(\theta^t\right)\right)$ as the minimizer and minimum whenever $\left|R\left(\theta^{t+1}\right) - R\left(\theta^t\right)\right| < \varepsilon$, where $\varepsilon > 0$ is the tolerance.



ISL Figure 10.17

- The gradient $\nabla R(\theta)$ gives the direction in θ-space in which $R(\theta)$ increases most rapidly: for $\|\theta' - \theta\| = 1$,

$$R(\theta') - R(\theta) \approx \nabla R(\theta)^\top (\theta' - \theta)$$

and

$$|R(\theta') - R(\theta)| \leq \|\nabla R(\theta)\|.$$

The upper bound is attained when $\theta' - \theta = \nabla R(\theta) / \|\nabla R(\theta)\|$.

Stochastic gradient descent

- Denote

$$R_i(\theta) = \frac{1}{2}\left\{Y_i - \left(b_0 + \sum_{k=1}^{K} b_k g\left(w_{k0} + \sum_{j=1}^{p} w_{kj} X_{j,i}\right)\right)\right\}^2.$$

  Then

$$\frac{\partial R(\theta)}{\partial \theta} = \sum_{i=1}^{n} \frac{\partial R_i(\theta)}{\partial \theta}.$$

- When n is large, instead of summing over all n observations, we can sample a small fraction of them each time we compute a gradient step.
- This process is known as stochastic gradient descent.

Multilayer neural networks

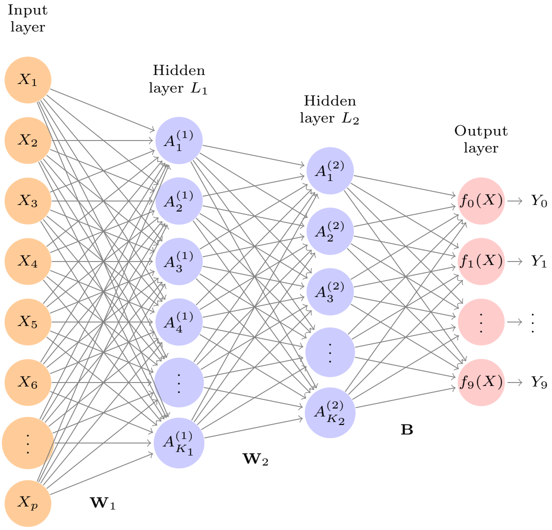
- In theory a single hidden layer with a large number of units has the ability to approximate most functions (White, 1992).
- A multiple hidden layer neural network model is called deep learning.
- The first hidden layer has hidden activations

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

  for $k = 1, ..., K_1$.
- The second hidden layer treats the activations $\left\{A_k^{(1)} : k = 1, ..., K_1\right\}$ of the first hidden layer as inputs and computes new activations

$$A_\ell^{(2)} = h_\ell^{(2)}(X) = g\left(w_{\ell 0}^{(2)} + \sum_{k=1}^{K_1} w_{\ell k}^{(2)} A_k^{(1)}\right)$$

  for $\ell = 1, ..., K_2$.



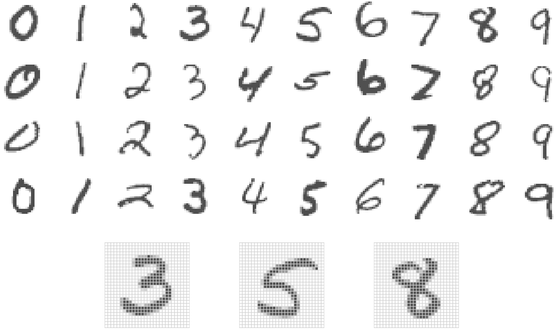
ISL Figure 10.4

- A two-layer model:

$$
\begin{aligned}
m(X,\theta) &= b_0 + \sum_{\ell=1}^{K_2} b_\ell h_\ell^{(2)}(X) \\
&= b_0 + \sum_{\ell=1}^{K_2} b_\ell g\left(w_{\ell 0}^{(2)} + \sum_{k=1}^{K_1} w_{\ell k}^{(2)} h_k^{(1)}(X) \right).
\end{aligned}
$$

- $\mathbf{W}_1$: weights that feed from the input layer to the first hidden layer, $K_1 \times (p+1)$.
- $\mathbf{W}_2$: weights that feed from the first hidden layer to the second hidden layer, $(K_1+1) \times K_2$.
- $\mathbf{B}$: $10 \times (K_2+1)$.

Example: MNIST Digits



ISL Figure 10.3

- Each grayscale image has 28×28 pixels ($p = 784$), each of which is an eight-bit number ($X_j \in \{0, 1, ..., 255\}, \forall j = 1, ..., p$).
- 60000 training images, 10000 test images.
- Labels are the digit class 0-9.
- Goal: build a classifier to predict the image class.
- Use a two-layer model with $K_1 = 256$ and $K_2 = 128$.

- $(Y_0, Y_1, ..., Y_9)$: the vector of 10 dummy variables (class labels).
- The training data: $\left(Y_{t,i}, X_i\right) : t = 0, 1, ..., 9, i = 1, ..., 6000$.
- Denote

$$Z_t = b_{t0} + \sum_{\ell=1}^{K_2} b_{t\ell} h_\ell^{(2)} (X)$$

for $t = 0, 1, ..., 9$.
- The model for approximating $\Pr[Y_t = 1 \mid X]$:

$$m_t (X) = \frac{e^{Z_t}}{\sum_{\ell=0}^{9} e^{Z_\ell}},$$

for $t = 0, 1, ..., 9$.
- The model estimates a probability for each of the 10 classes. The classifier then assigns the image to the class with the highest probability.
- We look for coefficient estimates that minimize the negative multinomial log-likelihood:

$$-\sum_{i=1}^{n} \sum_{t=0}^{9} Y_{t,i} \log (m_t (X_i)) .$$

- Adding the number of coefficients in $(\mathbf{W}_1, \mathbf{W}_2, \mathbf{B})$, we get 235146 weights/model parameters.
- To avoid overfitting, some regularization is needed.
  - Dropout regularization: randomly remove units with some probability at each gradient descent update.
  - Similar to randomly omitting variables when growing trees in random forests.
- Best reported deep learning test error rates are around 0.5%. Human error rate is around 0.2%.

Method	Test Error
Neural Network + Ridge Regularization	2.3%
Neural Network + Dropout Regularization	1.8%
Multinomial Logistic Regression	7.2%
Linear Discriminant Analysis	12.7%