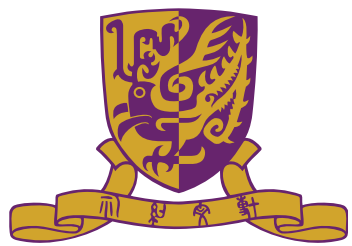
DDA5001 Machine Learning

Neural Networks (Part I): Model

Xiao Li

School of Data Science
The Chinese University of Hong Kong, Shenzhen



Neural Network Model

Training Two Layer Neural Networks

The Linear Models We Studied and Limitations

We have mainly studied linear regression and linear classification.

- Least squares, perception, logistic regression, and SVM. They are all linear supervised machine learning models.
- Linear model does not perform well on highly non-linearly separable / regressible data.

Many challenging objectives consist of non-linear structures. One way to deal with non-linear case is through nonlinear transformation of data, making the data to be linearly separable in higher dimension.

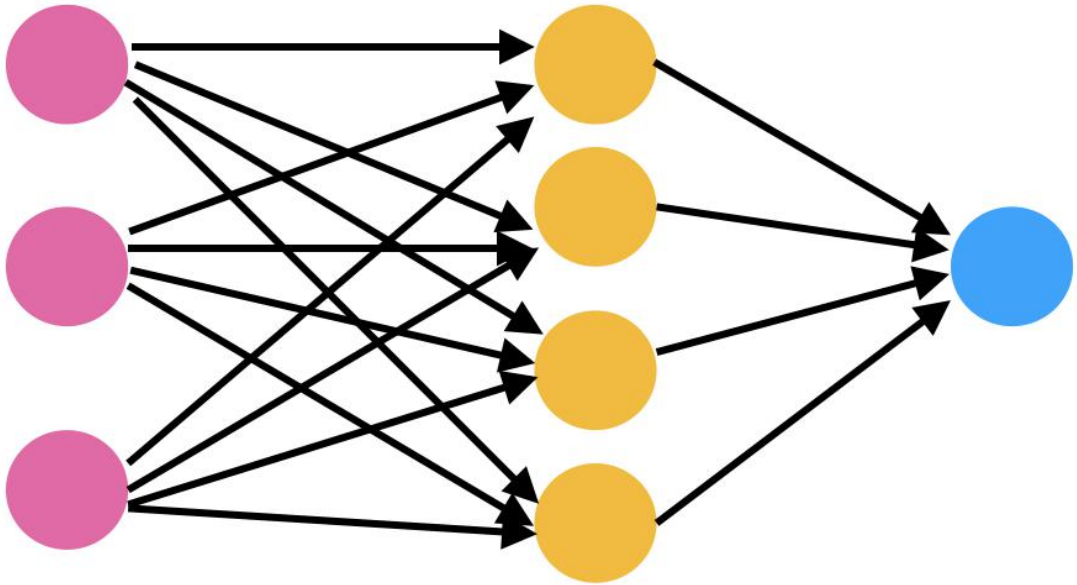
- Kernel method. However, this method needs to choose the right κ.

Another way is to impose non-linearity in our model $f_{\boldsymbol{\theta}}(\boldsymbol{x})$ (nonlinear in $\boldsymbol{\theta}$), leading to nonlinear supervised learning model.

- Note that supervised learning model is to use $f_{\boldsymbol{\theta}}(\boldsymbol{x})$ to approximate the underlying (nonlinear) pattern g.
- One most noticeable model is neural networks due to its universal approximation power.

Structure of Neural Network

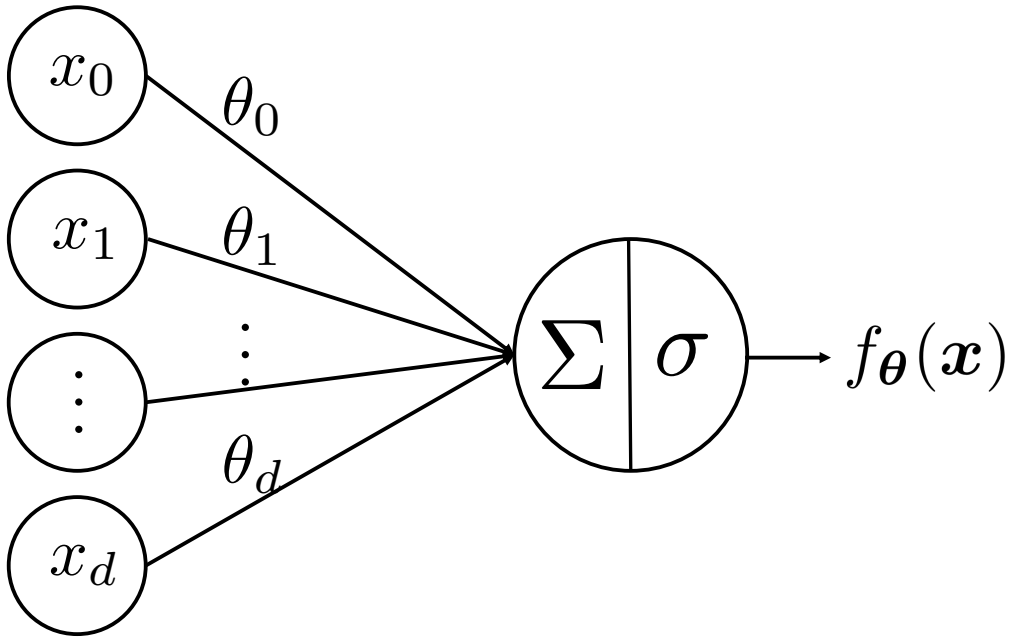
Neural network structure:



- Originally conceived as models for the brain.
  - Nodes are neurons.
  - Edges are synapses.

Let's start with the simplest single layer neural net.

Single Layer Neural Network

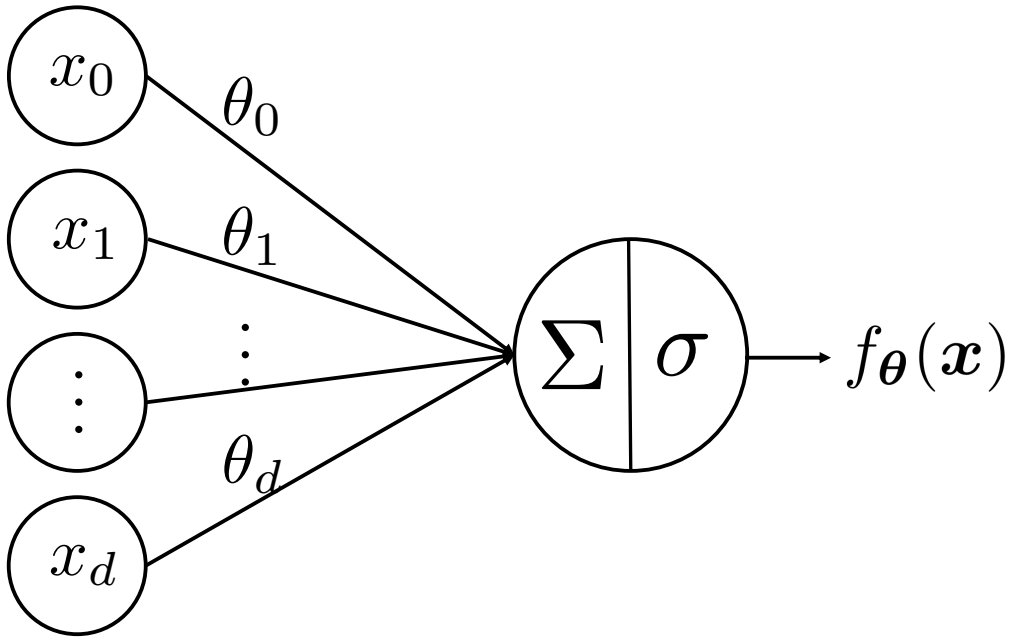


- Sample $\boldsymbol{x} = (x_0, \ldots, x_d)$ with $x_0 = 1$.
- Parameters $\boldsymbol{\theta} = (\theta_0, \ldots, \theta_d)$, called weights.
- σ is called activation function.

This neural network (NN) structure means

$$f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \sigma\left(\sum_{i=0}^{d} \theta_i x_i\right) = \sigma(\boldsymbol{\theta}^\top \boldsymbol{x}).$$

Neural Network As A Generalized Linear Model



This single layer neural network covers linear models.

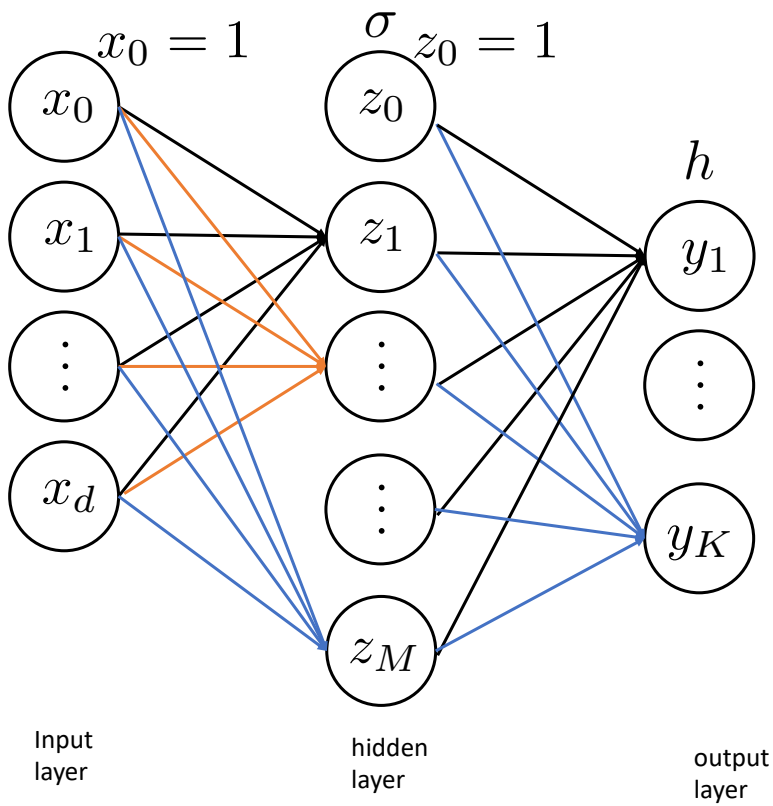
- Vanilla linear regression model if $\sigma(t) = t$.
- Logistic regression (binary case) if

$$\sigma(t) = \frac{1}{1 + e^{-y \cdot t}}$$

More generally, neural network can have more than one layer.

One Hidden Layer (Two-Layer) Neural Network

We generalize neural network to two-layer:



Model:

$$z_m = \sigma(\boldsymbol{v}_m^\top \boldsymbol{x}), \quad m = 1, \ldots, M$$
$$y_k = h(\boldsymbol{w}_k^\top \boldsymbol{z}), \quad k = 1, \ldots, K$$

One Hidden Layer Neural Network

Mathematically, this one hidden layer neural network is presented as

$$z_m = \sigma(\boldsymbol{v}_m^\top \boldsymbol{x}), \quad m = 1, \ldots, M$$
$$y_k = h(\boldsymbol{w}_k^\top \boldsymbol{z}), \quad k = 1, \ldots, K$$

The main idea: Use the output of M linear models + possibly nonlinear transformation σ as the input to another linear model.

- σ is fixed activation function, h depends on tasks.
- What are the unknown parameters?

$$\boldsymbol{v}_m, \quad \boldsymbol{w}_k, \qquad m = 1, \ldots, M \quad \text{and} \quad k = 1, \ldots, K$$

- What are the dimension of $\boldsymbol{v}_m$ and $\boldsymbol{w}_k$?
- How many extra parameters compared to vanilla linear model?

Neural Network Model in Matrix Form

$$z_m = \sigma(\boldsymbol{v}_m^\top \boldsymbol{x}), \quad m = 1, \ldots, M.$$
$$y_k = h(\boldsymbol{w}_k^\top \boldsymbol{z}), \quad k = 1, \ldots, K.$$

Let us omit the bias term for simplicity and without loss of generality. Letting $\boldsymbol{V} \in \mathbb{R}^{M \times d}$ denote the matrix having rows $\boldsymbol{v}_m^\top$ and $\boldsymbol{W} \in \mathbb{R}^{K \times M}$ denote the matrix having rows $\boldsymbol{w}_k^\top$, we can write this neural net as

$$\boldsymbol{z} = \sigma(\boldsymbol{V}\boldsymbol{x}).$$
$$\boldsymbol{y} = h(\boldsymbol{W}\boldsymbol{z}).$$

One hidden layer NN model can be written compactly

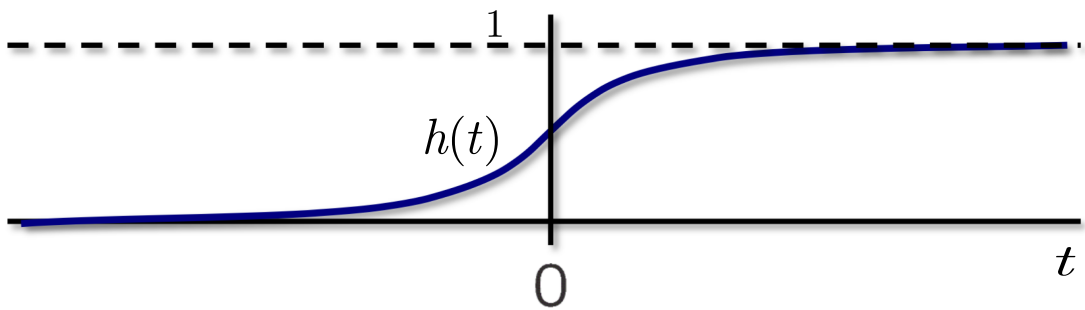
$$\boxed{\boldsymbol{y} = f_{\boldsymbol{\theta}}(\boldsymbol{x}) = h(\boldsymbol{W}\sigma(\boldsymbol{V}\boldsymbol{x})).}$$

- $\boldsymbol{\theta} := (\boldsymbol{V}, \boldsymbol{W})$ contains the network parameters, called weights.
- The input of the next layer is the output of the previous layer ($\boldsymbol{z} = \sigma(\boldsymbol{V}\boldsymbol{x})$).

The Ingredients in Typical Neural Networks

- A historically common choice for σ is the logistic / sigmoid function:

$$\sigma(t) = \frac{1}{1 + e^{-t}}.$$



- A popular activation function design based on sigmoid is SiLU (Sigmoid Linear Unit):

$$\sigma(t) = \frac{t}{1 + e^{-t}}.$$

- Another popular choice of σ is the ReLU (rectified linear unit):

$$\sigma(t) = \max(0, t).$$

The Ingredients in Typical Neural Networks

The choice of h depends somewhat on the application.

- Regression, we usually use

$$h(t) = t.$$

- Logistic regression for binary classification, where $K = 1, y = \{+1, -1\}$

$$h(t) = \frac{1}{1 + e^{-y \cdot t}}.$$

  One can also use SVM model here.

- Multiclass classification

$$h(t_k) = \frac{e^{t_k}}{\sum_{j=1}^{K} e^{t_j}}, \quad t_k = \boldsymbol{w}_k^\top \boldsymbol{z},$$

  which is to let the last layer to be a multi-class logistic regression classifier.

$\rightsquigarrow$ h is to impose certain linear model ($\boldsymbol{z}$ as input) at the end of NN.

The Representation Power of Neural Networks

One natural question would be why consider neural network model for nonlinear case.

- Supervised learning is to approximate the underlying pattern g. The target function g is nonlinear in general.
- The following theorem shows a universal approximation power of just one hidden layer neural networks, which shows that neural networks can represent any regular function, and thus can approximate any underlying (possibly nonlinear) g.

Theorem: Universal approximation Power

Let g be a continuous function on a bounded subset $\mathcal{K}$ of d-dimensional space. Then, there exists a one hidden layer neural network $f_{\boldsymbol{\theta}}$ with a finite number of hidden neurons that approximates g arbitrarily well. Namely, for all sample $\boldsymbol{x} \in \mathcal{K}$, we have $|f_{\boldsymbol{\theta}}(\boldsymbol{x}) - g(\boldsymbol{x})| < \varepsilon$ for every $\varepsilon > 0$.

Remarks

- Like kernel methods, neural networks fit a linear model in a nonlinear feature space.
- Unlike kernel methods, these nonlinear features are learned — i.e., output of the hidden layer.
- Interpretation of NN: One can think neural network model as extracting features by nonlinear network and finally put the extracted feature z into the last layer for regression or classification. So, the last layer can be viewed as LS, LR, SVM, etc.
  $\rightsquigarrow$ This is also true for deep learning, i.e., deep neural network model (later).
- We will see that the training of neural network model involves nonconvex optimization because of the involved nonlinear structure of $f_{\boldsymbol{\theta}}$.

Training Two Layer Neural Networks

Training Neural Networks

- Training data: $(\boldsymbol{x}_1, \boldsymbol{y}_1), \ldots, (\boldsymbol{x}_n, \boldsymbol{y}_n)$ with $\boldsymbol{x}_i \in \mathbb{R}^d$ and $\boldsymbol{y}_i \in \mathbb{R}^K$
- Neural network model

$$f_{\boldsymbol{\theta}}(\boldsymbol{x}) = h(\boldsymbol{W}\sigma(\boldsymbol{V}\boldsymbol{x})).$$

- We aim to learn the weight parameters $\boldsymbol{\theta} = (\boldsymbol{V}, \boldsymbol{W})$ such that

$$\boldsymbol{y}_i \leftarrow f_{\boldsymbol{\theta}}(\boldsymbol{x}_i).$$

  Thus, this is a supervised learning problem.

- We can quantify this approximation by choosing a loss function which we will seek to minimize by picking $\boldsymbol{\theta}$ appropriately

The Learning Problem for Training Neural Networks

Regression: $h(t) = t$ and use squared ℓ_2 loss, resulting in

- $K = 1$

$$\min_{\boldsymbol{\theta}} \left\{ \mathcal{L}(\boldsymbol{\theta}) = \frac{1}{n} \sum_{i=1}^{n} (y_i - f_{\boldsymbol{\theta}}(\boldsymbol{x}_i))^2 \right\}$$

- $K > 1$

$$\min_{\boldsymbol{\theta}} \left\{ \mathcal{L}(\boldsymbol{\theta}) = \frac{1}{n} \sum_{i=1}^{n} \|\boldsymbol{y}_i - f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)\|_2^2 \right\}$$

The Learning Problem for Training Neural Networks

Classification (Binary): $K = 1, y = \{+1, -1\}$ and h is logistic function, and use logistic loss, resulting in

$$\min_{\boldsymbol{\theta}} \left\{ \mathcal{L}(\boldsymbol{\theta}) = -\frac{1}{n} \sum_{i=1}^{n} \log(f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)) \right\}$$

Classification (Multi-class): $K > 1$, $\boldsymbol{y} = (0, \ldots, 1, \ldots, 0)$, h is softmax and use multiple-class logistic loss, resulting in

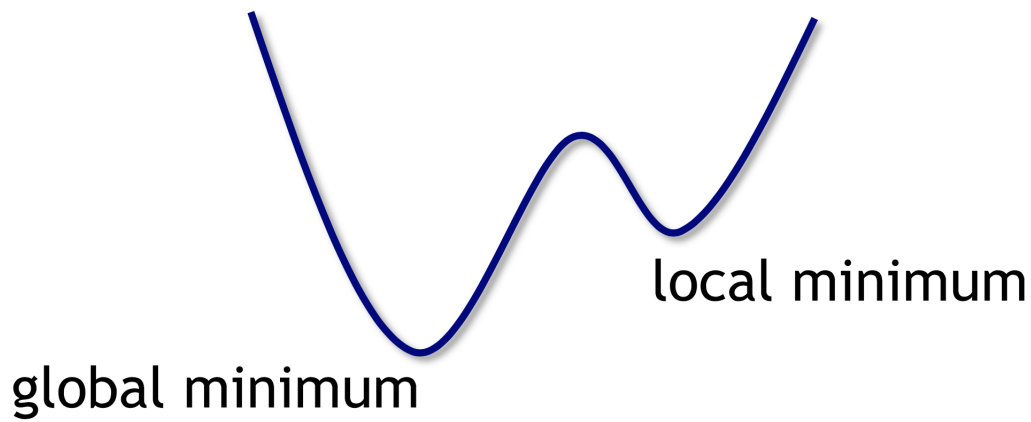
$$\min_{\boldsymbol{\theta}} \left\{ \mathcal{L}(\boldsymbol{\theta}) = -\frac{1}{n} \sum_{i=1}^{n} \boldsymbol{y}_i^\top \log(f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)) \right\}$$

Training Neural Networks is Nonconvex Optimization

For example, regression training problem can be written as:

$$\min_{\boldsymbol{\theta}=(\boldsymbol{V},\boldsymbol{W})} \left\{ \mathcal{L}(\boldsymbol{\theta}) = \frac{1}{n} \sum_{i=1}^{n} \|\boldsymbol{y}_i - h(\boldsymbol{W}\sigma(\boldsymbol{V}\boldsymbol{x}_i))\|^2 \right\}.$$

This is a highly nonconvex optimization problem.



Training Neural Networks

We put an abstract form of the former learning problems:

$$\min_{\boldsymbol{\theta}} \left\{ \mathcal{L}(\boldsymbol{\theta}) = \frac{1}{n} \sum_{i=1}^{n} \ell_i(\boldsymbol{\theta}) \right\}$$

For different applications (regression or classification), ℓ_i has its own form.

- One can apply a gradient-based training algorithm.
- One needs to compute the gradient

$$\nabla \mathcal{L}(\boldsymbol{\theta}) = \frac{1}{n} \sum_{i=1}^{n} \nabla \ell_i(\boldsymbol{\theta}).$$

- Apply gradient-based training algorithm:

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \mu_k \nabla \mathcal{L}(\boldsymbol{\theta}_k).$$

However... The gradient is not easy to compute and GD training algorithm might be too optimistic.

Next: Training Algorithm and its Ingredients

Next lectures will dive into the depth of neural network training:

- How to compute the gradient?
  $\rightsquigarrow$ The well-known backpropagation (BP).
- In contemporary applications, n is so large. Applying GD is not feasible.
  $\rightsquigarrow$ Stochastic gradient descent, Adagrad, and Adam family.