GR5260
Programming for Quantitative & Computational Finance

Instructor: Ka Yi Ng

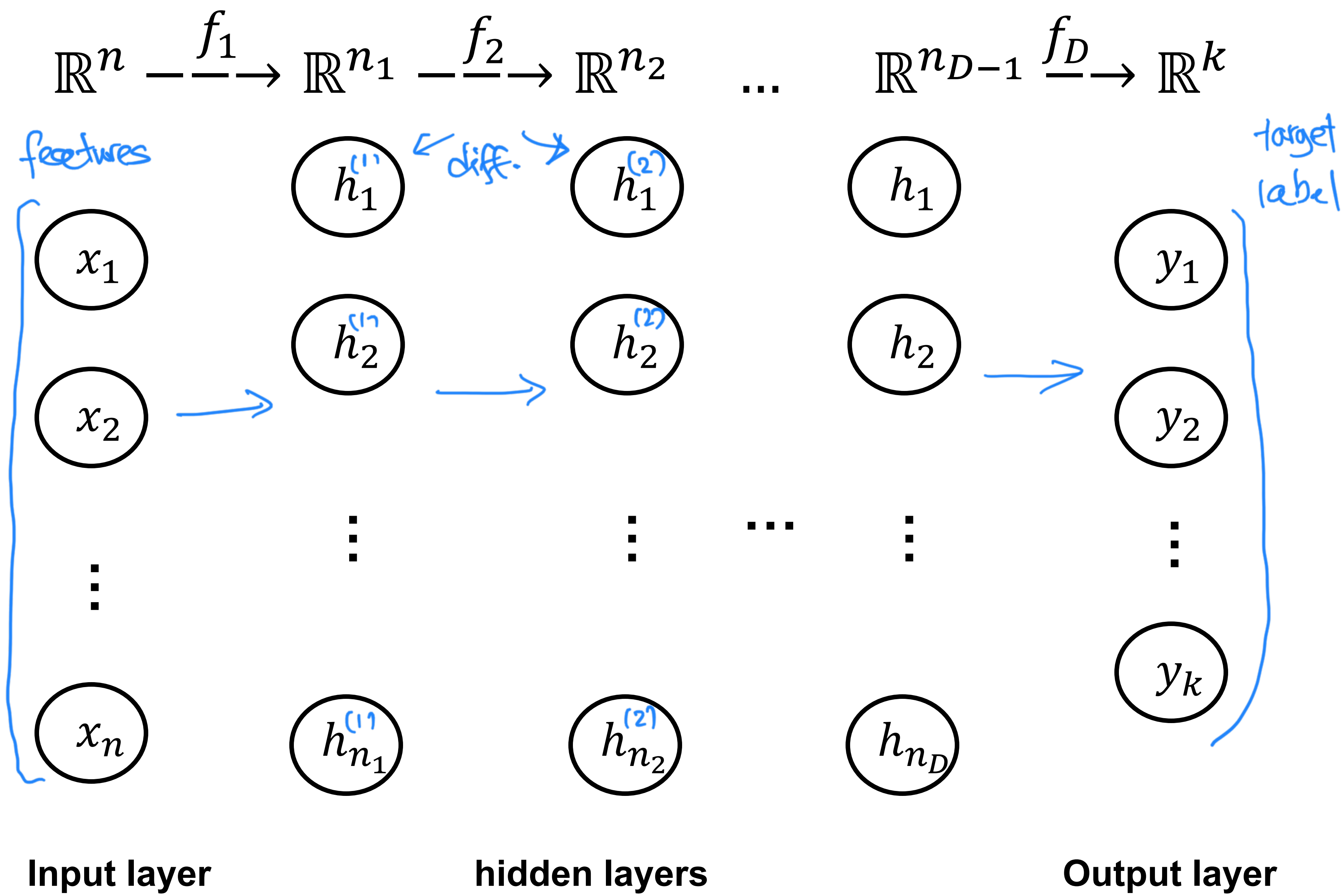
April 28, 2023

Neural networks

Feedforward Neural Networks

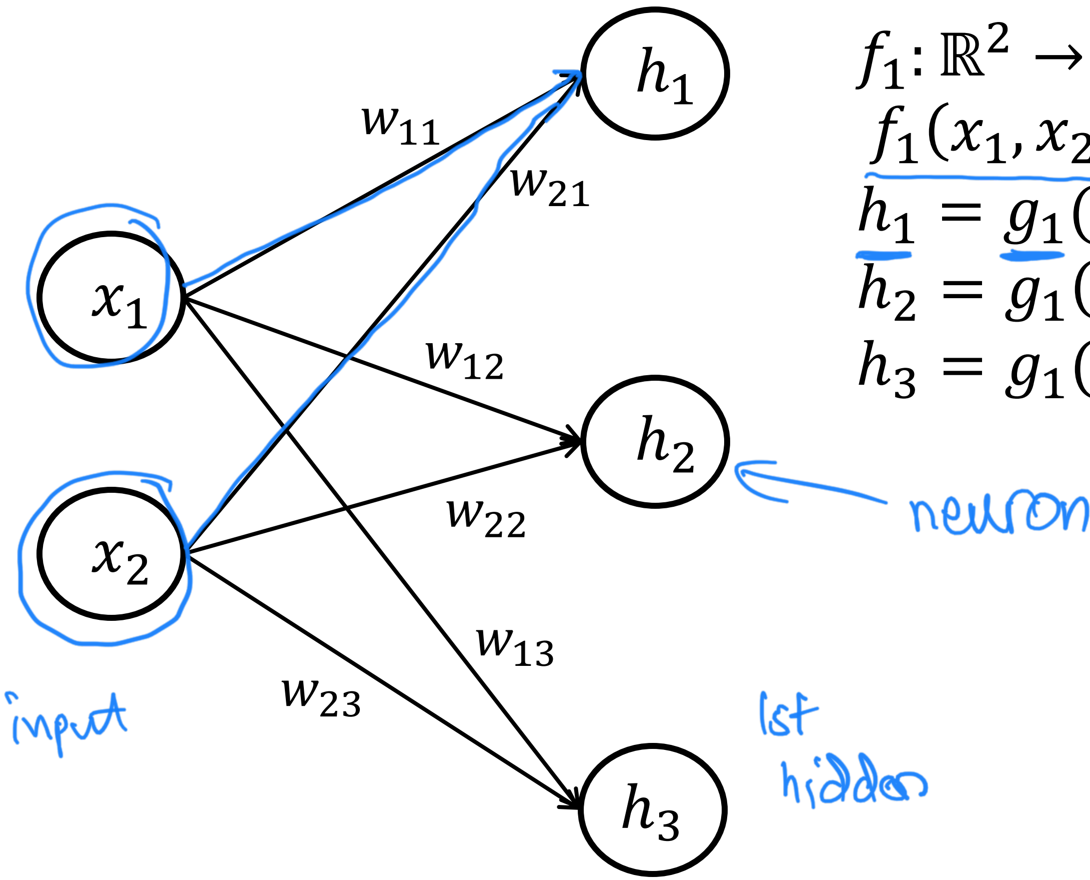
- Estimate the input-output mapping $f: \mathbb{R}^n \to \mathbb{R}^k$ using a parametrized function $\hat{f}: \mathbb{R}^n \to \mathbb{R}^k$ of the following form:
- $\hat{f} = f_D \circ f_{D-1} \circ \cdots \circ f_2 \circ f_1$
  where $f_1: \mathbb{R}^n \to \mathbb{R}^{n_1}, f_2: \mathbb{R}^{n_1} \to \mathbb{R}^{n_2}$, ... ,
  $f_{D-1}: \mathbb{R}^{n_{D-2}} \to \mathbb{R}^{n_{D-1}},\ f_D: \mathbb{R}^{n_{D-1}} \to \mathbb{R}^k$
- $f_i(\boldsymbol{x}) = (g_i(W_{i1}\boldsymbol{x} + b_{i1}), g_i(W_{i2}\boldsymbol{x} + b_{i2}), \dots, g_i(W_{in_i}\boldsymbol{x} + b_{in_i}))$
- W_{ij} is a weight vector of same dimension as $\boldsymbol{x}$
- b_{ij} is a scalar, called bias
- $g_i: \mathbb{R} \to \mathbb{R}$ is an activation function (eg. $max(0, z)$, $tanh(z)$)

Feedforward Neural Networks



Feedforward Neural Networks

$$\boxed{f_i(\boldsymbol{x}) = (g_i(W_{i1}\boldsymbol{x} + b_{i1}), g_i(W_{i2}\boldsymbol{x} + b_{i2}), \dots, g_i(W_{in_i}\boldsymbol{x} + b_{in_i}))}$$



$f_1: \mathbb{R}^2 \to \boxed{\mathbb{R}^3}$

$f_1(x_1, x_2) = (h_1, h_2, h_3)$

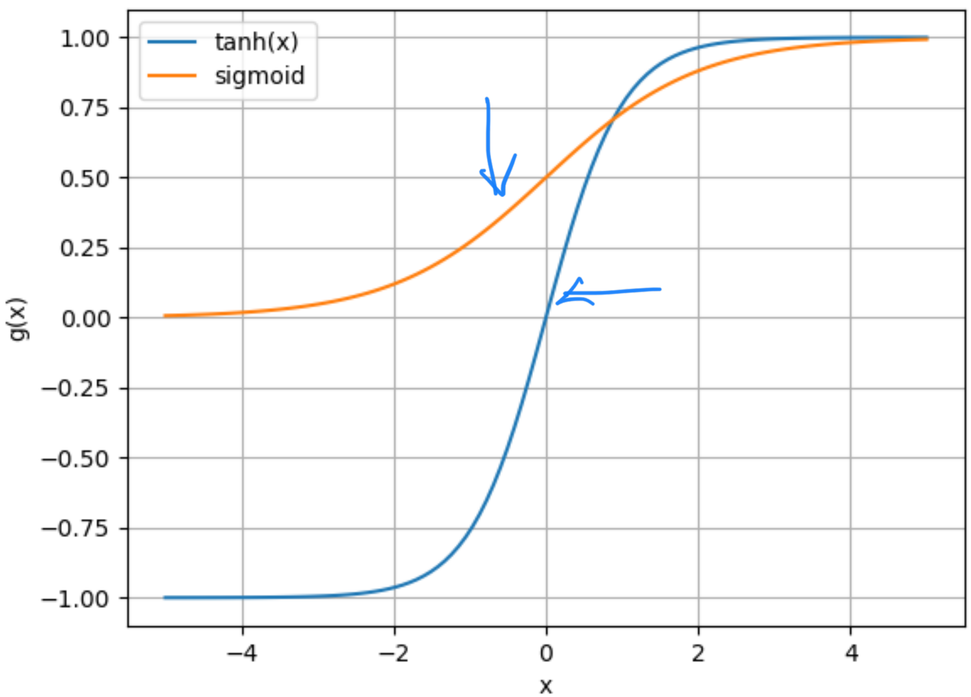
$h_1 = g_1(w_{11}x_1 + w_{21}x_2 + b_{11})$

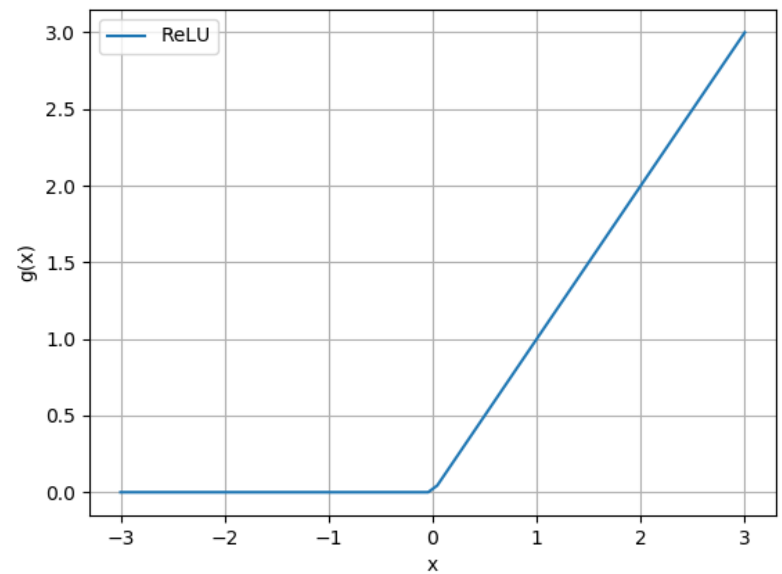
$h_2 = g_1(w_{12}x_1 + w_{22}x_2 + b_{12})$

$h_3 = g_1(w_{13}x_1 + w_{23}x_2 + b_{13})$

Activation functions

- Rectified linear unit (ReLU): $g(z) = max(0, z)$
- Hyperbolic tangent $g(z) = tanh(z) = \frac{\exp(z) - \exp(-z)}{\exp(z) + \exp(-z)}$
- Logistic sigmoid $g(z) = \frac{1}{1 + \exp(-z)}$





More examples in https://en.wikipedia.org/wiki/Activation_function

Feedforward Neural Networks

- Hyper-parameters:
  - L: depth of neural network
  - $n_1, \dots, n_{L-1}$: width of the hidden layers
  - $g_1, \dots, g_L$: activation functions
- Model parameters of $\hat{f}(\boldsymbol{x};\ \{W_{ij}, b_{ij}\}_{i,j})$: W_{ij}'s, b_i's
- Training set: $(\boldsymbol{x}^{(1)}, \boldsymbol{y}^{(1)}), \dots, (\boldsymbol{x}^{(m)}, \boldsymbol{y}^{(m)})$

Goal: solve for W_{ij}'s and b_{ij}'s that minimize the average cost function $J(\boldsymbol{w})$ over the training set

$$J(\boldsymbol{w}) = \frac{1}{m}\sum_{i=1}^{m} L(\hat{f}(x^{(i)}), y^{(i)})$$

  - Stochastic Gradient Descent or Adam is used
  - Weight initialization: Glorot uniform $w_{ij} \sim Unif(-a, a)$
  - Back propagation to compute gradients

Universal Approximation

- Let $\mathcal{N}$ be a feedforward network with one hidden layer and linear output units (ie. $\hat{f} = f_2 \circ f_1$ and f_2 is linear)
- (Hornik 1991) *If the activation function is continuous, bounded and non-constant, $\mathcal{N}$ can approximate <u>any continuous</u> function on a compact subset $\boldsymbol{X} \subset \mathbb{R}^n$ to any degree of accuracy provided that sufficiently many hidden units are available.*
- *Activation function example:* $tanh(z)$
- (Leshno 1993) *$\mathcal{N}$ with a locally bounded piecewise continuous activation can approximate <u>any continuous</u> function to any degree of accuracy if and only if the activation function is not a polynomial*.
- *Activation function example:* $max(0, z)$

FNN: Regression

- Input layer: feature vector $\boldsymbol{x} \in \mathbb{R}^n$
- Output layer: label value y, $g_D = \text{identity}$
- Hyper-parameters:
  - D: depth of neural network
  - $n_1, \ldots, n_{D-1}$: width of the hidden layers
  - $g_1, \ldots, g_{D-1}$: activation functions
- Cost function: $L(y, \hat{y}) = \|y - \hat{y}\|^2$
- Training set: $\left(\boldsymbol{x}^{(\mathbf{1})}, \boldsymbol{y}^{(\mathbf{1})}\right), \ldots, \left(\boldsymbol{x}^{(\boldsymbol{m})}, \boldsymbol{y}^{(\boldsymbol{m})}\right)$
- Goal: solve for W_{ij}'s and b_i's that minimize the average cost function $J(\boldsymbol{w})$ over the training set

$$J(\boldsymbol{w}) = \frac{1}{m} \sum_{i=1}^{m} (\hat{f}\left(x^{(i)}\right) - y^{(i)})^2$$

FNN: Binary classification

- Feature space: $X \subset \mathbb{R}^n$, output: $Y = \{0, 1\}$
- Training set: $(\boldsymbol{x}^{(\mathbf{1})}, \boldsymbol{y}^{(\mathbf{1})}), \dots, (\boldsymbol{x}^{(\boldsymbol{m})}, \boldsymbol{y}^{(\boldsymbol{m})})$
- Cost function: take a probabilistic viewpoint
- What is this likelihood $P(\boldsymbol{y}^{(\mathbf{1})}, \dots, \boldsymbol{y}^{(\boldsymbol{m})} | \boldsymbol{x}^{(\mathbf{1})}, \dots, \boldsymbol{x}^{(\boldsymbol{m})})$?
- Consider $P(y|\boldsymbol{x})$ conditional probability of y given $\boldsymbol{x}$
- Assume that it follows a Bernoulli distribution with parameter $p(\boldsymbol{x})$
- $P(y = 1|\boldsymbol{x}) = p(\boldsymbol{x})$, $P(y = 0|\boldsymbol{x}) = 1 - p(\boldsymbol{x})$
- $P(\boldsymbol{y}^{(\mathbf{1})}, \dots, \boldsymbol{y}^{(\boldsymbol{m})} | \boldsymbol{x}^{(\mathbf{1})}, \dots, \boldsymbol{x}^{(\boldsymbol{m})})$ can be expressed as:
- $\prod_{i=1}^{m} P(y = \boldsymbol{y}^{(\boldsymbol{i})} | \boldsymbol{x}^{(\boldsymbol{i})}) = \prod_{i=1}^{m} p(\boldsymbol{x})^{\boldsymbol{y}^{(\boldsymbol{i})}} (1 - p(\boldsymbol{x}))^{\mathbf{1} - \boldsymbol{y}^{(\boldsymbol{i})}}$
- Find $p(\boldsymbol{x})$ that maximizes the likelihood of observing the training dataset

FNN: Binary classification

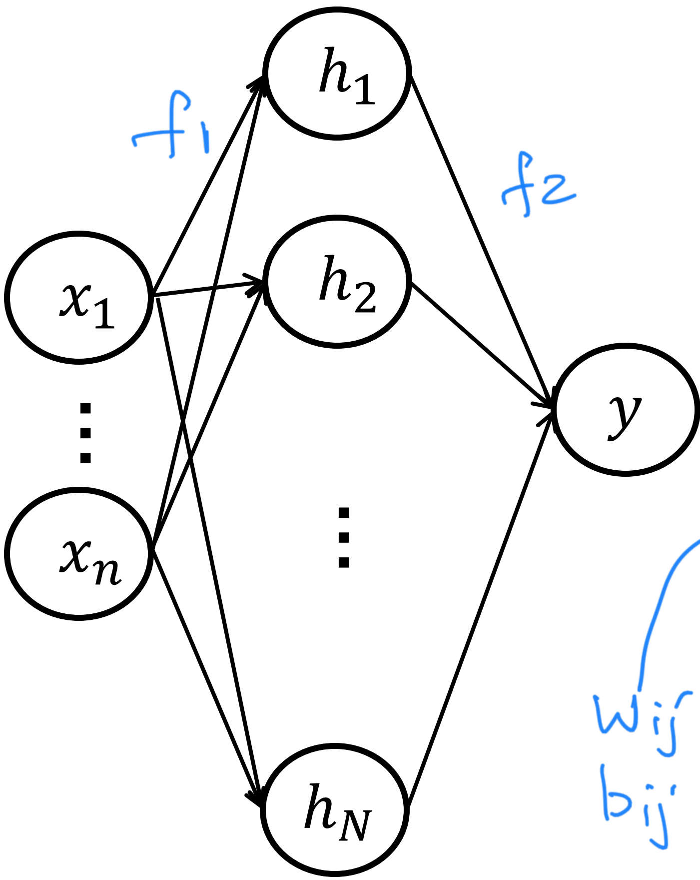
NN

- Write $p(\boldsymbol{x})$ as a parametrized function $p(\boldsymbol{x};\boldsymbol{\theta})$
- Likelihood function $h(\boldsymbol{\theta}) = \prod_{i=1}^{m} p(\boldsymbol{x};\boldsymbol{\theta})^{\boldsymbol{y}^{(i)}}(1-p(\boldsymbol{x};\boldsymbol{\theta}))^{\mathbf{1}-\boldsymbol{y}^{(i)}}$
- Goal: Find $\widehat{\boldsymbol{\theta}}$ that maximizes the likelihood function
- Equivalently: Find $\widehat{\boldsymbol{\theta}}$ that minimizes the negative log-likelihood function $J(\boldsymbol{\theta}) = -\log(h(\boldsymbol{\theta}))$
- $J(\boldsymbol{\theta}) = -\frac{1}{m}\sum_{i=1}^{m} \underbrace{\boldsymbol{y}^{(i)}\log(p(\boldsymbol{x};\boldsymbol{\theta})) + (1-\boldsymbol{y}^{(i)})\log(1-p(\boldsymbol{x};\boldsymbol{\theta}))}_{\text{Cost function}}$

- Use feedforward neural network $\hat{f}(\boldsymbol{x};\boldsymbol{\theta})$ to approximate $p(\boldsymbol{x};\boldsymbol{\theta})$ where $\boldsymbol{\theta}$ is the set of W_{ij}'s and b_{ij}'s in the network

FNN: Binary classification

$$\boxed{\hat{f}: \mathbb{R}^n \to \mathbb{R} \text{ with } \hat{f} = f_2 \circ f_1}$$



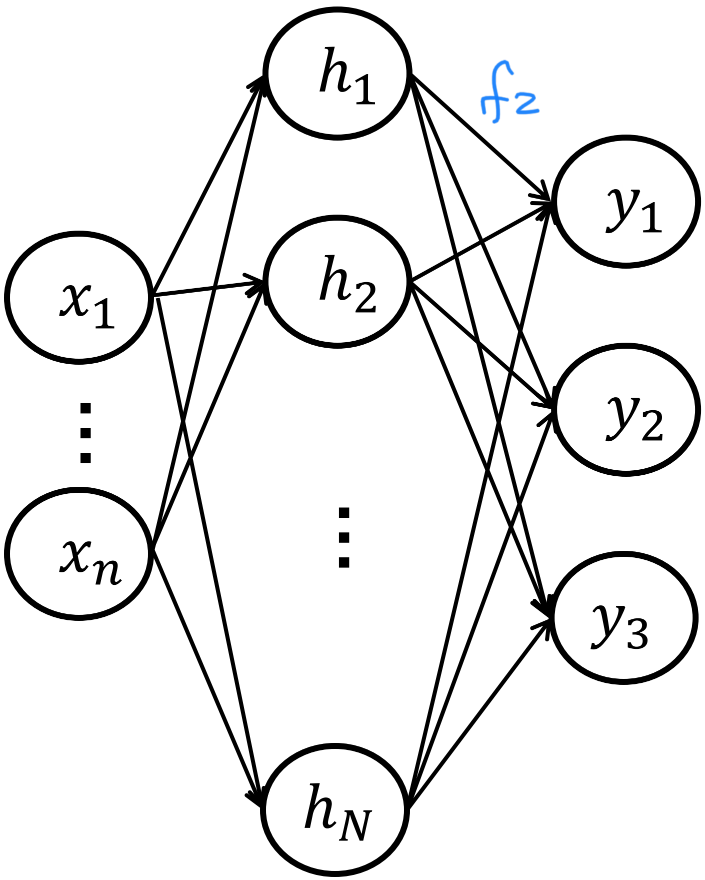
$f_1: \mathbb{R}^n \to \mathbb{R}^N$ with $f_1(\boldsymbol{x}) = (h_1, \ldots, h_N)$
and $h_i = g(W_{1i}\boldsymbol{x} + b_1)$
g is an activation function (eg. ReLU)

$f_2: \mathbb{R}^N \to \mathbb{R}$ with
$f_2(\boldsymbol{x}) = \sigma(W_2\boldsymbol{x} + b_2)$
$\sigma(z) = \frac{1}{1+\exp(-z)}$ logistic sigmoid

$\hat{f}(\boldsymbol{x}; \boldsymbol{\theta})$ is an estimate of $p(\boldsymbol{x}; \boldsymbol{\theta})$
Prediction: $\vec{x}$
W_{ij} if $\hat{f}(\boldsymbol{x}) \geq 0.5, \hat{y} = 1$; otherwise, $\hat{y} = 0$
b_{ij}

FNN: K-class classification

$$\boxed{\hat{f}: \mathbb{R}^n \to \mathbb{R}^K,\ \hat{f} = f_2 \circ f_1}$$



$f_1: \mathbb{R}^n \to \mathbb{R}^N$ with $f_1(\boldsymbol{x}) = (h_1, \ldots, h_N)$
and $h_i = g(W_{1i}\boldsymbol{x} + b_1)$
g is an activation function (eg. ReLU)

$f_2: \mathbb{R}^N \to \mathbb{R}^3$ with $f_2(\boldsymbol{h}) = s(W_2\boldsymbol{h} + b_2)$
s: $\mathbb{R}^3 \to \mathbb{R}^3$ softmax function
$s(z_1, z_2, z_3) = (s_1, s_2, s_3)$ where

$$s_i = s_i(\boldsymbol{z}) = \frac{\exp(z_i)}{\sum_{j=1}^{3} \exp(z_j)}$$

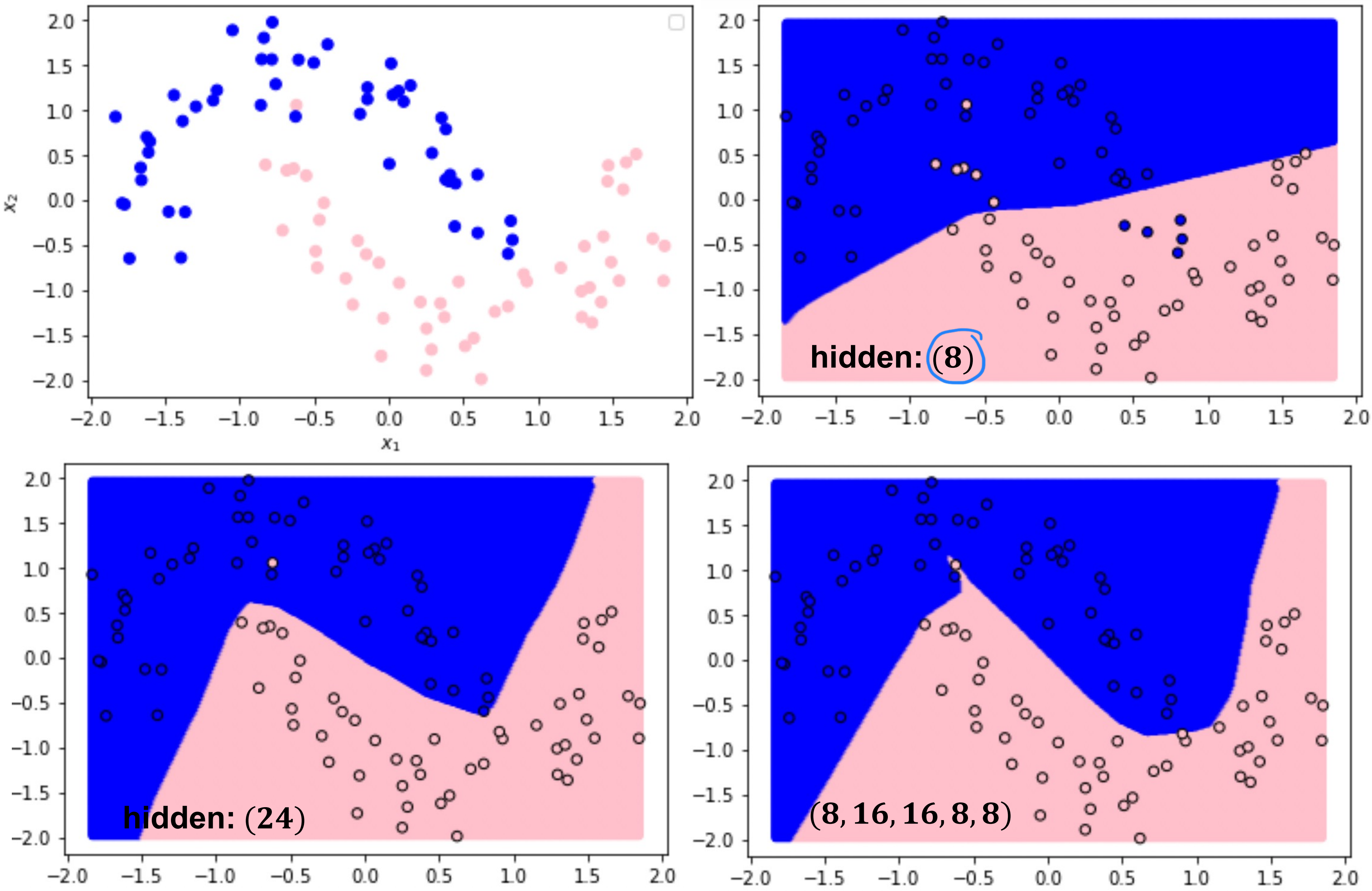
$\hat{f}(\boldsymbol{x}; \boldsymbol{\theta})$: estimate of $p_i(\boldsymbol{x}; \boldsymbol{\theta})'s$
ie. estimate of $P(y = i\,|\boldsymbol{x})$

Cost function:
- negative log-likelihood

Prediction:
- class with highest predicted probability

MLP: width & depth

ReLU



Time series forecasting

- Classical linear time series models AR(p)
- $X(t) = \phi_0 + \phi_1 X(t-1) + \cdots + \phi_p X(t-p) + \varepsilon$
- Classical ML models and ensembles
  - Features: lagged returns
- Recurrent neural networks (RNN)
  - Hidden state $h(t)$: information up to time t
  - $h(t) = f(X(t), h(t-1))$
  - Recurrent layer as a parametrization of function f
  - $h(t) = g(\boxed{X(t)W_{xh} + h(t-1)W_{hh} + b_h})$
  - W_{xh} and W_{hh} are weight matrices, b_h bias vector
  - g: an activation function
  - Output layer: $y = s(h(t)W + b)$

    logistic function
- LSTM networks
  - Different parametrization of f