Everything up to today is about linear

$x$, $\phi(w)$

$w^T x$, $w^T \phi(x)$

# Neural Networks

# Neural Networks

- Origins: Algorithms that try to mimic the brain.
- Widely used in 80s and early 90s; popularity diminished in late 90s.
- Recent resurgence from 10s: state-of-the-art techniques for many applications:
  - Computer Vision 2012 AlexNet
  - Natural language processing
  - Speech recognition
  - Decision-making / control problems (AlphaGo, Dota, robots)
- Limited theory:
  - Non-convexity SGD can optimize well
  - Model are complex but generalization error is small

# Neural Networks

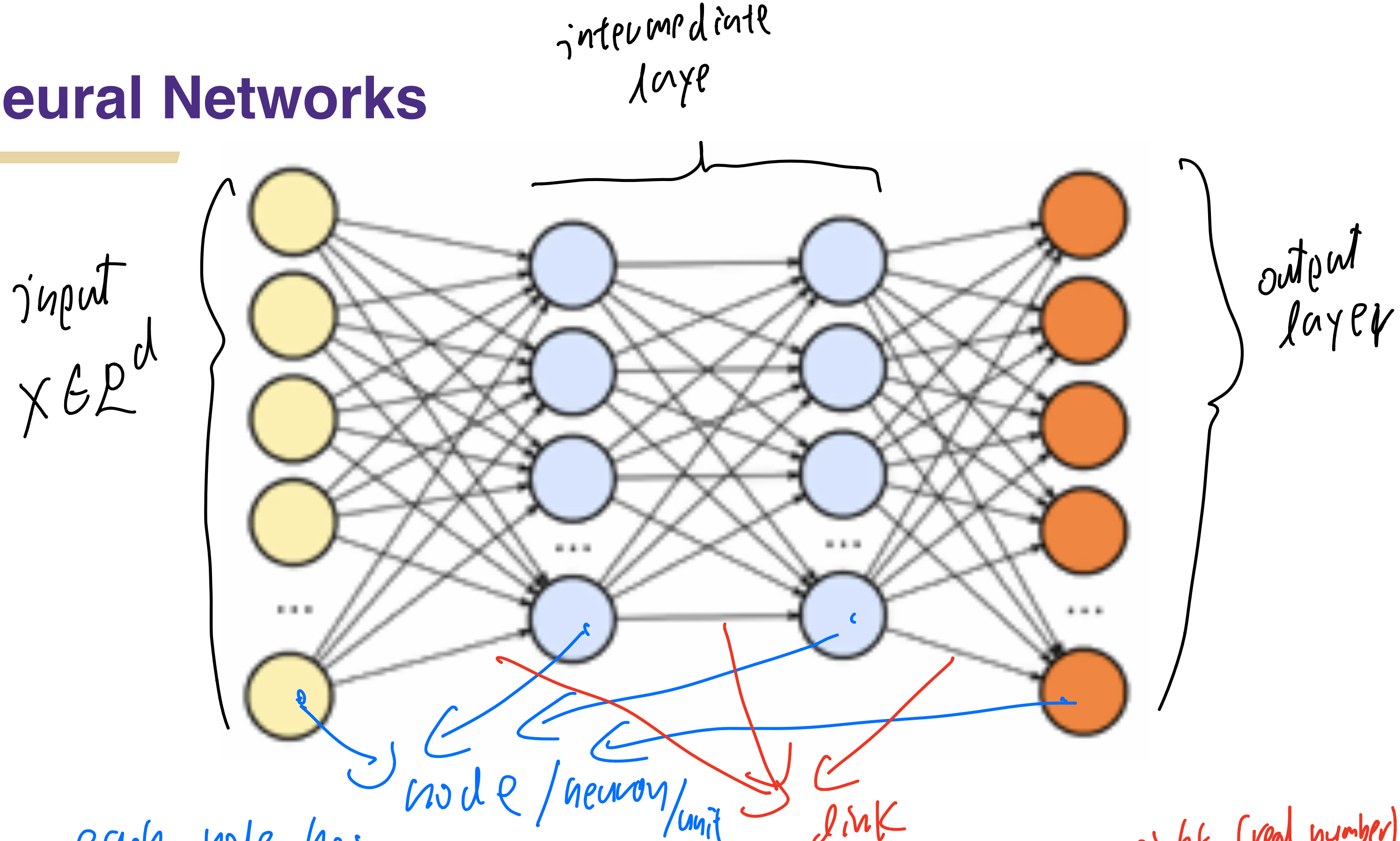
This week:

1. Definitions of neural networks

2. Training neural networks:
   1. Algorithm: back propagation
   2. Putting it to work

3. Neural network architecture design:
   1. Convolutional neural network → Computer vision

# Neural Networks

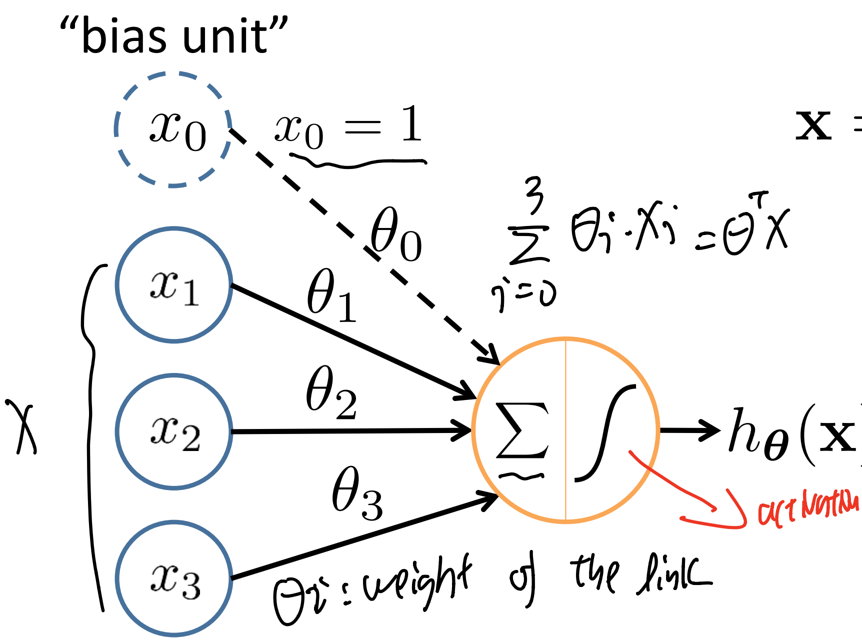



each link has weight (real number)

each node has
1) input
2) activation function
3) output

d=3

# Single Node




$$\mathbf{x} = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \qquad \boldsymbol{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix}$$

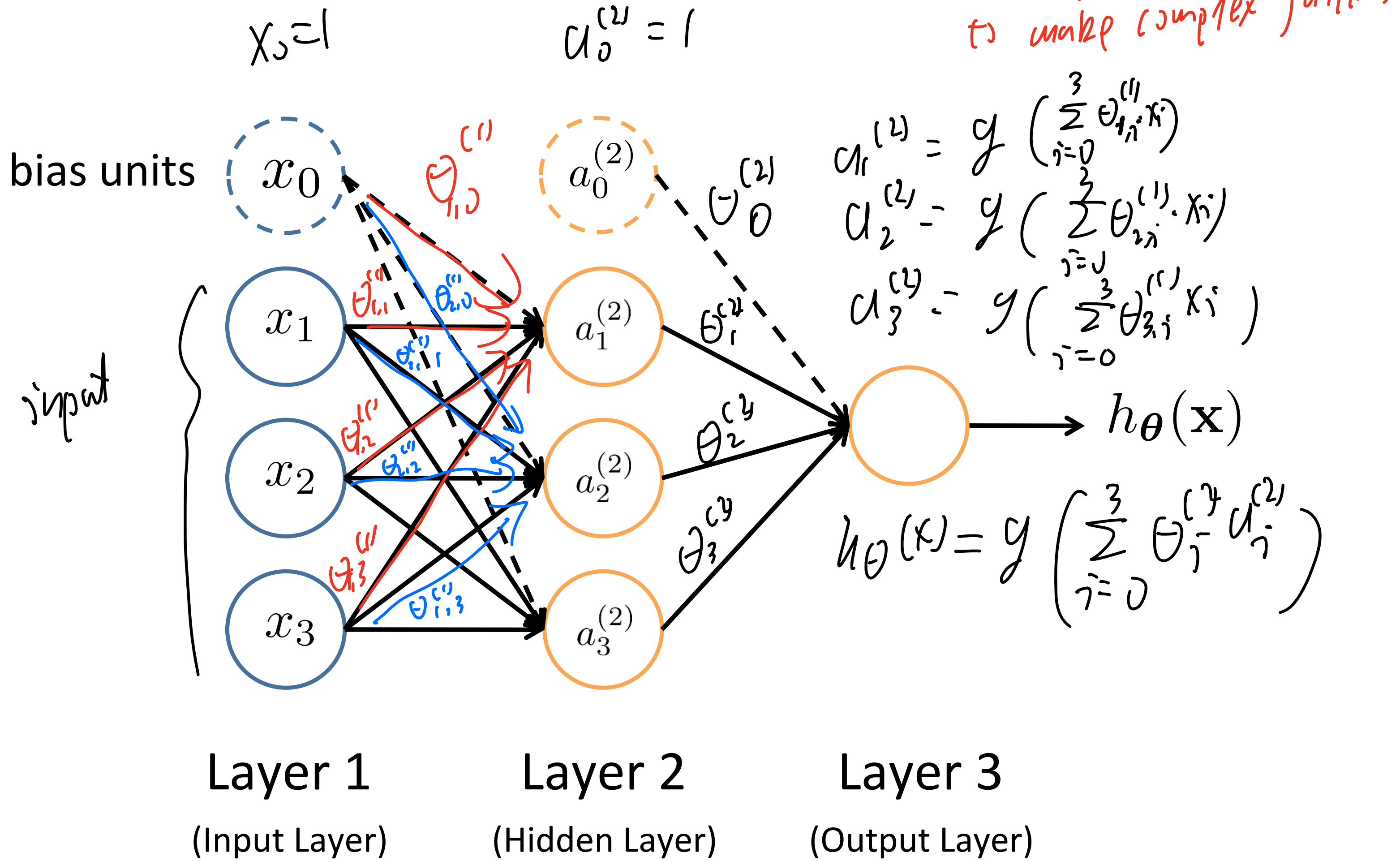
$$h_{\boldsymbol{\theta}}(\mathbf{x}) = g\left(\boldsymbol{\theta}^{\mathsf{T}}\mathbf{x}\right) = \frac{1}{1 + e^{-\boldsymbol{\theta}^{\mathsf{T}}\mathbf{x}}}$$

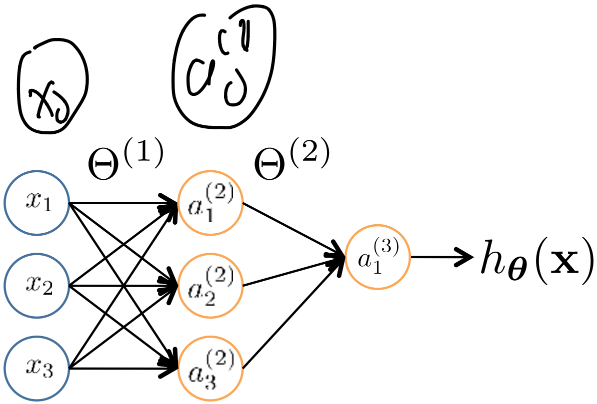
Binary Logistic Regression

Sigmoid (logistic) activation function: $g(z) = \frac{1}{1 + e^{-z}}$

other: linear $g(z) = z$
ReLU $g(z) = \max\{0, z\}$

# Neural Network






$a_i^{(j)}$ = "activation" of unit $i$ in layer $j$

$\Theta^{(j)}$ = weight matrix stores parameters from layer $j$ to layer $j$ + 1

$$a_1^{(2)} = g(\Theta_{10}^{(1)} x_0 + \Theta_{11}^{(1)} x_1 + \Theta_{12}^{(1)} x_2 + \Theta_{13}^{(1)} x_3)$$
$$a_2^{(2)} = g(\Theta_{20}^{(1)} x_0 + \Theta_{21}^{(1)} x_1 + \Theta_{22}^{(1)} x_2 + \Theta_{23}^{(1)} x_3)$$
$$a_3^{(2)} = g(\Theta_{30}^{(1)} x_0 + \Theta_{31}^{(1)} x_1 + \Theta_{32}^{(1)} x_2 + \Theta_{33}^{(1)} x_3)$$
$$h_\Theta(x) = a_1^{(3)} = g(\Theta_{10}^{(2)} a_0^{(2)} + \Theta_{11}^{(2)} a_1^{(2)} + \Theta_{12}^{(2)} a_2^{(2)} + \Theta_{13}^{(2)} a_3^{(2)})$$

If network has $s_j$ units in layer $j$ *and* $s_{j+1}$ units in layer $j$+1, then $\Theta^{(j)}$ has dimension $s_{j+1} \times (s_j+1)$ .

$$\Theta^{(1)} \in \mathbb{R}^{3\times 4} \qquad \Theta^{(2)} \in \mathbb{R}^{1\times 4}$$

# Multi-layer Neural Network - Binary Classification

L: depth

$$a^{(1)} = x$$

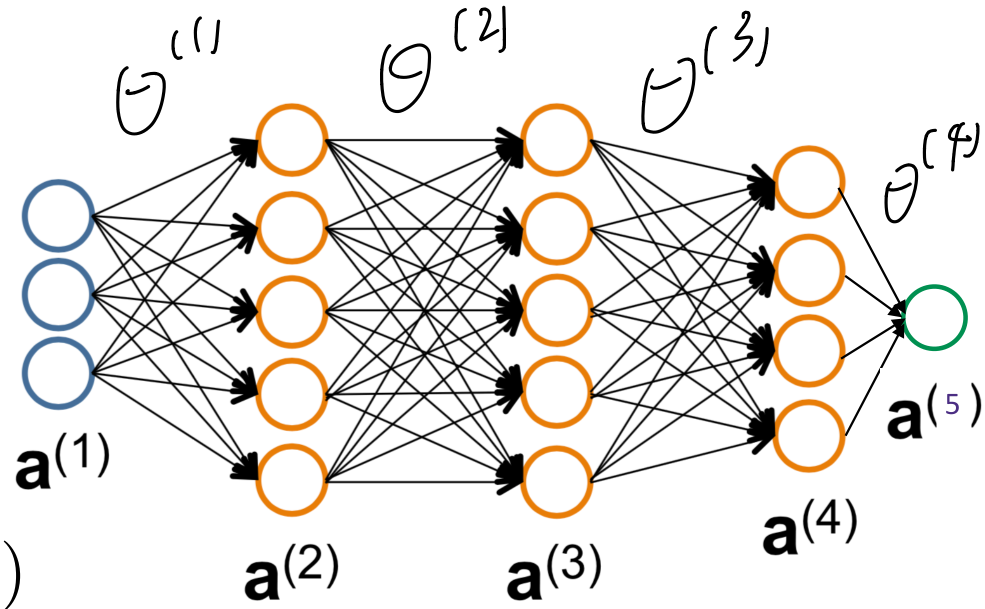
$$a^{(2)} = g(\Theta^{(1)} a^{(1)})$$

$$\vdots$$

$$a^{(l+1)} = g(\Theta^{(l)} a^{(l)})$$

$$\vdots$$

$$\hat{y} = g(\Theta^{(L)} a^{(L)})$$




y: label

$$L(y, \hat{y}) = y \log(\hat{y}) + (1-y)\log(1-\hat{y})$$

ŷ: our prediction ∈ [0,1]

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

Binary Logistic Regression

# Multi-layer Neural Network - Binary Classification

if $\sigma(z) = z$

$\hat{y} = g(\Theta^{(4)} a^{(4)}) = g(\Theta^{(4)} \Theta^{(3)} \Theta^{(2)} \Theta^{(1)} x)$ → linear

we want non-linear

$\hat{y} = g(\Theta^{(4)} a^{(4)})$

$$a^{(1)} = x$$

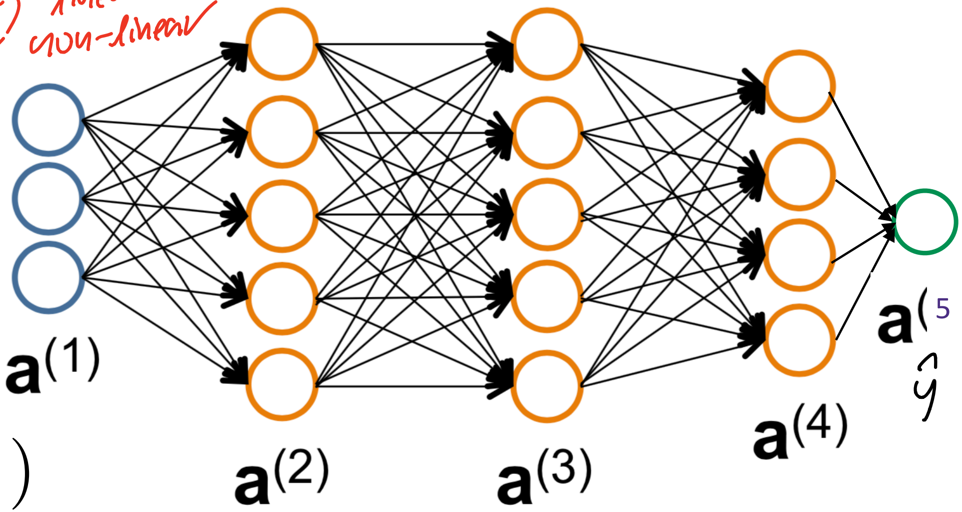
$$a^{(2)} = \sigma(\Theta^{(1)} a^{(1)})$$

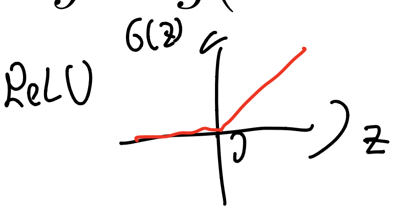
$$\vdots$$

$$a^{(l+1)} = \sigma(\Theta^{(l)} a^{(l)})$$

$$\vdots$$

$$\hat{y} = g(\Theta^{(L)} a^{(L)})$$




$\hat{y} = h_\Theta(x)$

$$L(y, \hat{y}) = y \log(\hat{y}) + (1 - y)\log(1 - \hat{y})$$

$L(y, \hat{y})$ = a function of $\Theta$: non-convex

$$\sigma(z) = \max\{0, z\} \quad g(z) = \frac{1}{1 + e^{-z}}$$

Binary Logistic Regression

# Multiple Output Units: One-vs-Rest

K: # of classes

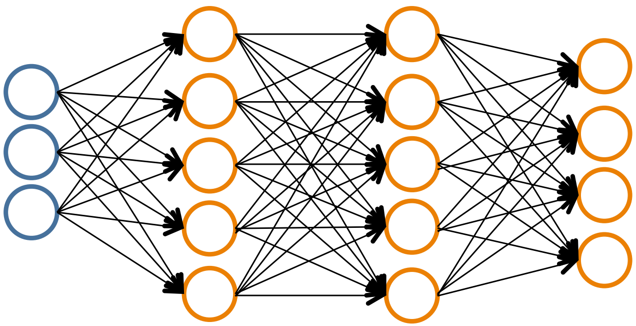
Pedestrian

Car

Motorcycle

Truck

$y = \begin{pmatrix} 0 \\ \vdots \\ 1 \\ 0 \\ 0 \end{pmatrix}$



loss = cross-entropy

$\ell(h_\Theta(x), y) = \sum_{k=1}^{K} -\log\left[[h_\Theta(x)]_k\right] \cdot y_k$

$$h_\Theta(\mathbf{x}) \in \mathbb{R}^K$$

only 1 non-zero entry in y

$[h_\Theta(x)]_k \in [0,1]$

$\sum_{k=1}^{K} [h_\Theta(x)]_k = 1$

Multi-class Logistic Regression

We want:

$h_\Theta(\mathbf{x}) \approx \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$ when pedestrian

$h_\Theta(\mathbf{x}) \approx \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$ when car

$h_\Theta(\mathbf{x}) \approx \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$ when motorcycle

$h_\Theta(\mathbf{x}) \approx \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$ when truck

# Multi-layer Neural Network - Regression

Θs are initialized randomly

$$a^{(1)} = x$$

$$a^{(2)} = \sigma(\Theta^{(1)} a^{(1)})$$

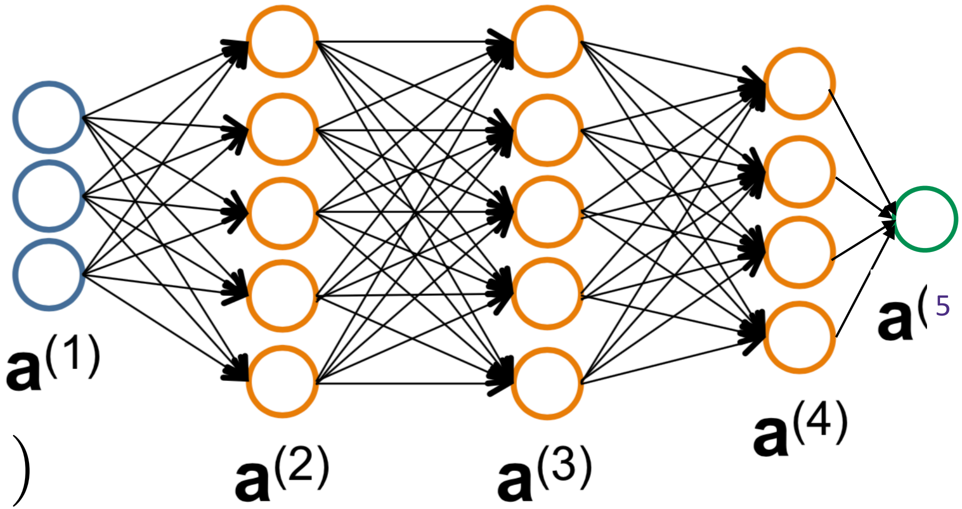
$$\vdots$$

$$a^{(l+1)} = \sigma(\Theta^{(l)} a^{(l)})$$

$$\vdots$$

$$\widehat{y} = \Theta^{(L)} a^{(L)}$$

no g, just linear




$$L(y, \widehat{y}) = (y - \widehat{y})^2$$

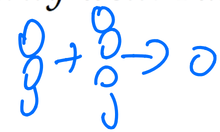
$$\sigma(z) = \max\{0, z\}$$

Regression

# Neural Networks are arbitrary function approximators

**Theorem 10** (Two-Layer Networks are Universal Function Approximators). *Let F be a continuous function on a bounded subset of D-dimensional space. Then there exists a two-layer neural network $\hat{F}$ with a finite number of hidden units that approximate F arbitrarily well. Namely, for all $\mathbf{x}$ in the domain of F, $|F(\mathbf{x}) - \hat{F}(\mathbf{x})| < \epsilon$.*

exp # of neuron



Cybenko, Hornik (theorem reproduced from CIML, Ch. 10)

This theorem cannot explain why DL is so powerful

· RBF Kernel : $w^T\phi(x)$ is also a universal

· DL has other properties (unknown)

· Neural Tangent Kernel