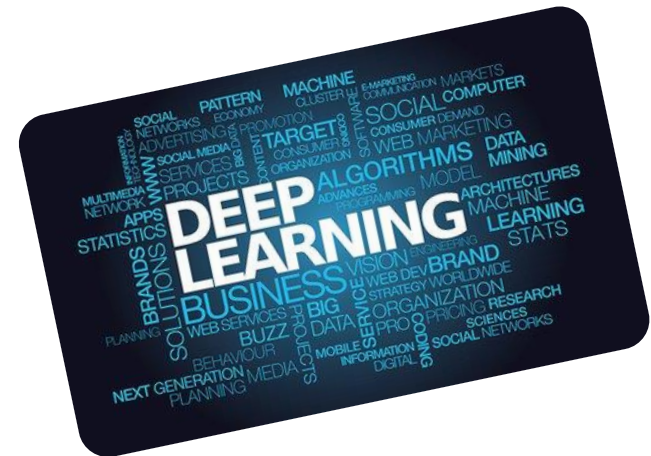


Introduction to Neural Networks And Deep Learning

Zhao Yunkun



AlphaGo Defeats Best Players, But...

I

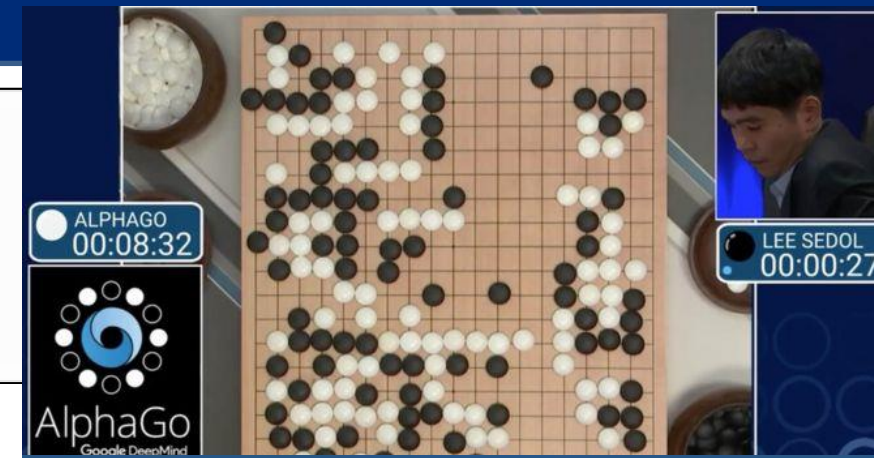
- 2016-03
- AlphaGo(1.0) **4 : 1** Lee Sedol

II

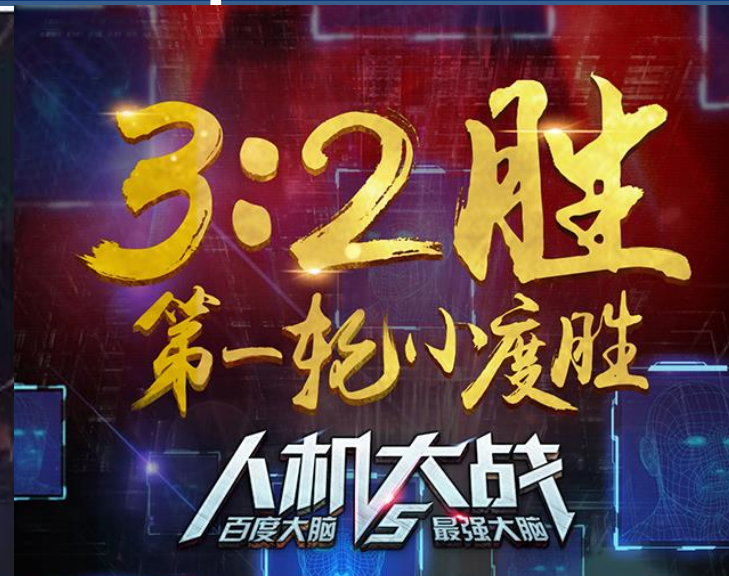
- 2017-05
- AlphaGo(2.0) **3 : 0** Ke Jie

III

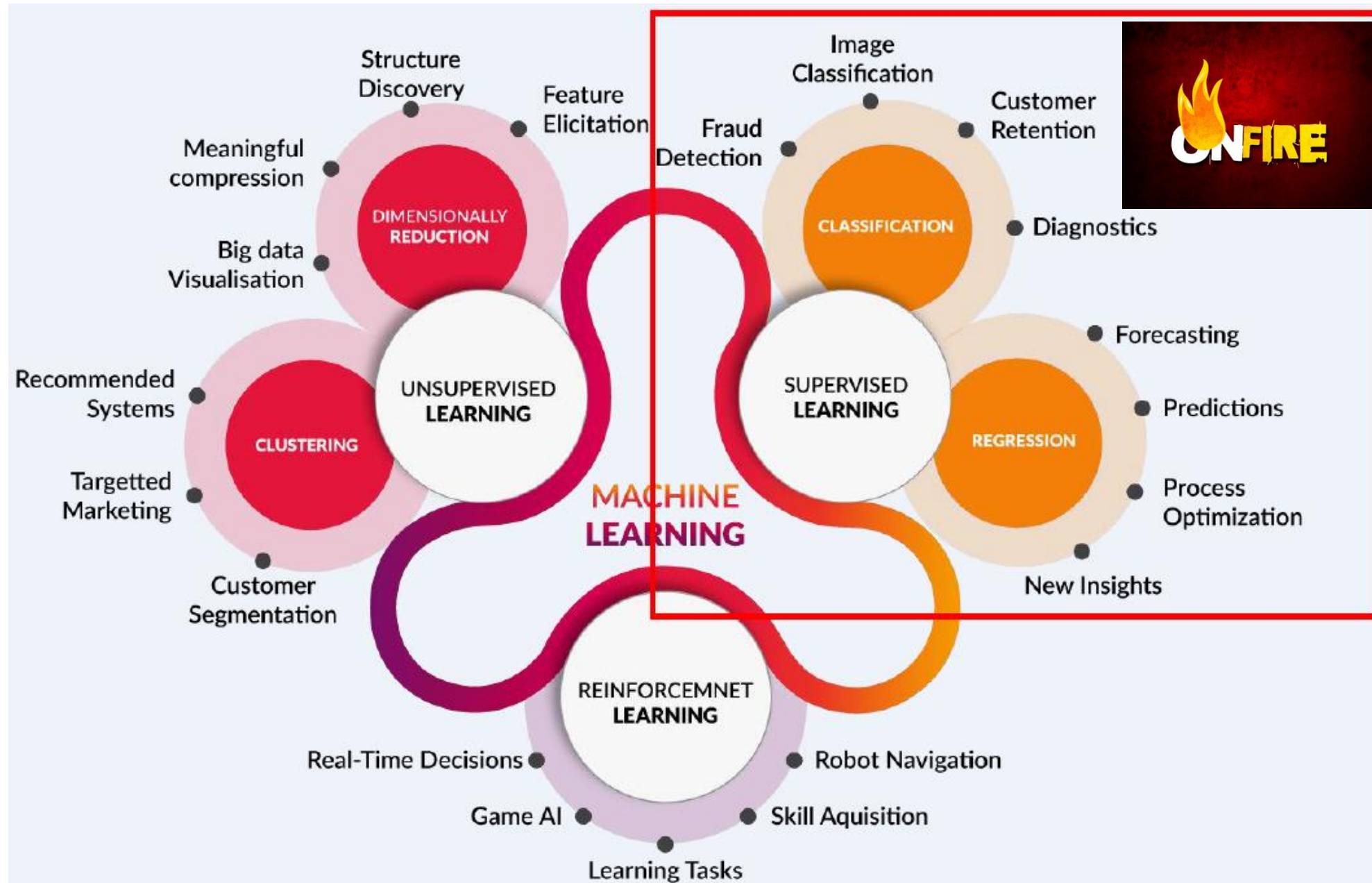
- 2017-10
- AlphaGo(2.0) **0 : 100** AlphaGo Zero



Baidu AI Brain Wins In Face-Recognition Competitions



Machine Learning And Deep Learning



How Does Machine Learns ?

➤ Example: Price Prediction With Regression

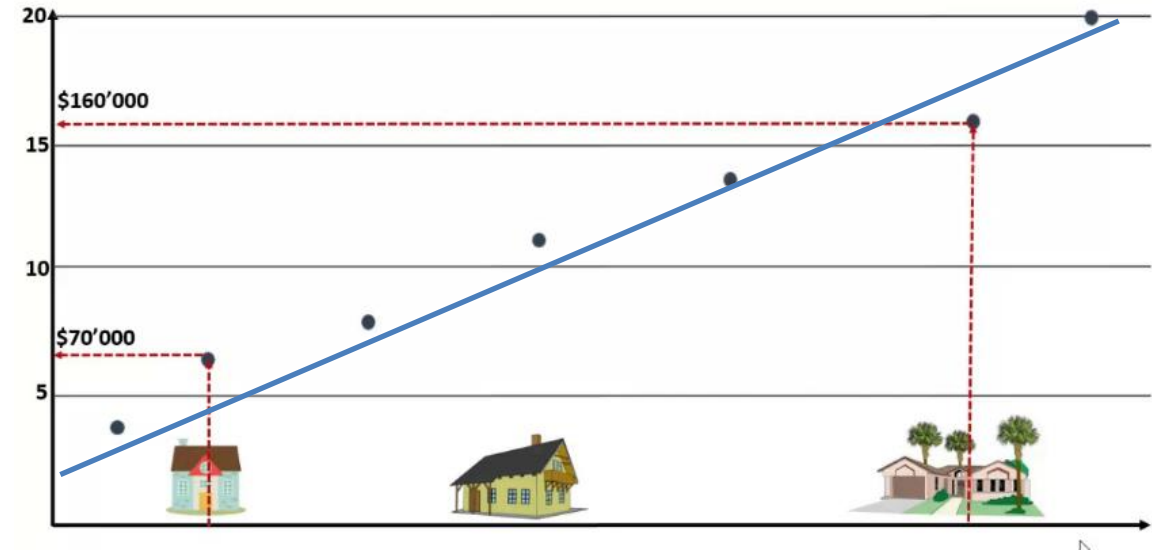
$$\text{Model: } Y = f(X) + \varepsilon = X\beta + \varepsilon$$

$$X = \begin{bmatrix} \text{Size} & \text{Bedroom} & \text{Bathroom} & \dots \\ x_{11} & x_{12} & \dots & x_{1p} \\ x_{21} & x_{22} & \dots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \dots & x_{np} \end{bmatrix}, \beta = \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_p \end{bmatrix}, Y = \begin{bmatrix} \text{Price} \\ y_1 \\ y_2 \\ \vdots \\ y_p \end{bmatrix}$$

$$\min_{\beta} \text{RSS}(\beta) = \varepsilon^T \varepsilon = (Y - X\beta)^T (Y - X\beta)$$

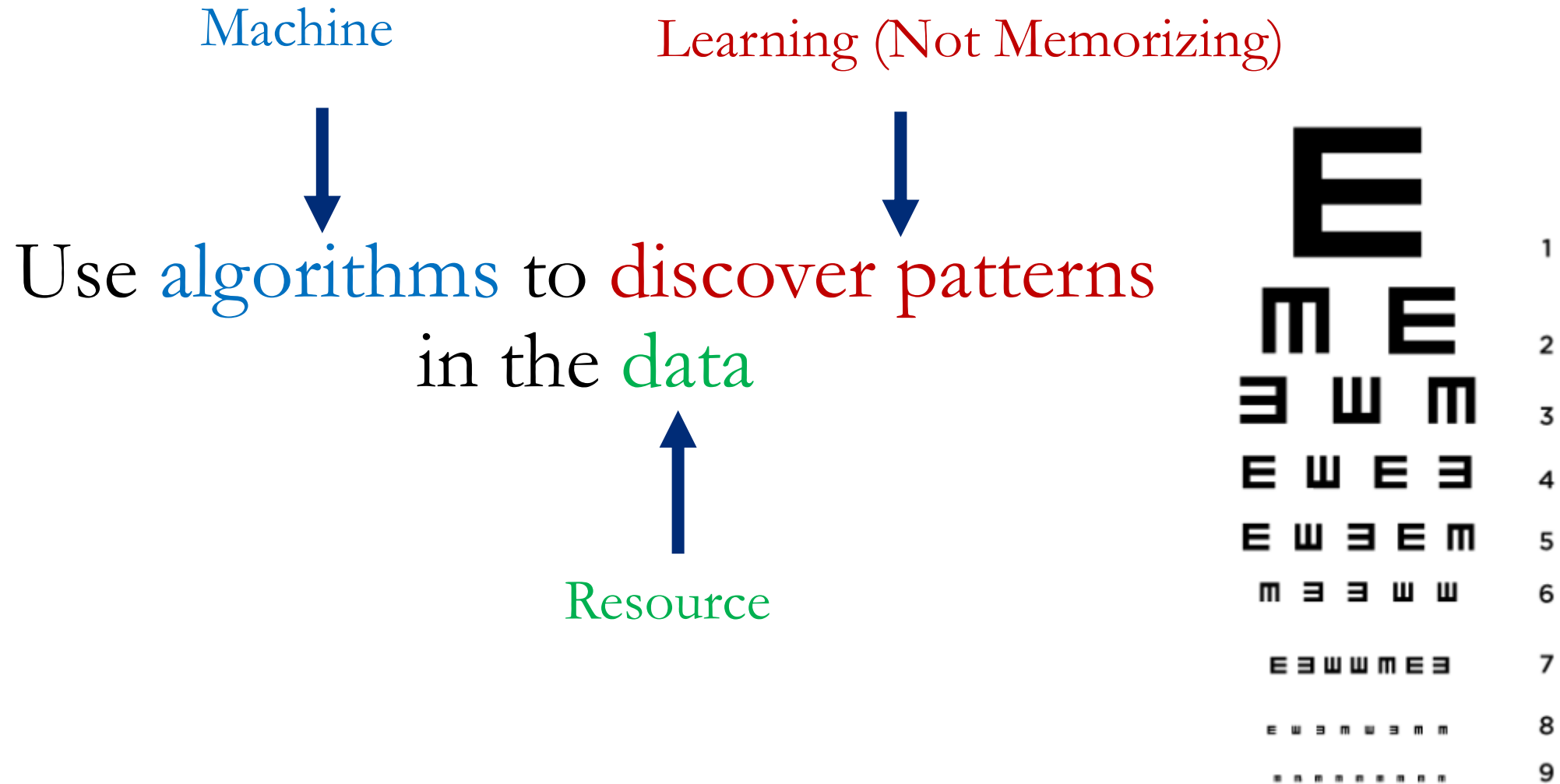
$$\rightarrow \text{f.d.: } -2X^T Y + 2X^T X \beta = 0$$

$$\rightarrow \beta_{OLS} = (X^T X)^{-1} X^T Y$$



ID	Size (m ²)	#Bed room	#Bath room	Price (k)
1	65	2	2	900
2	72	3	2	1200
.....
n	50	1	1	500

How Does Machine Learns ?



Machine Learning: Formalization

Input: $X \in \mathbb{R}^d$ (Housing information, e.g., size, #room, etc.)

Output: Regression: $y \in (-\infty, \infty)$ (e.g., Price)
Classification: $y \in (0, 1)$

Target Function: $f : X \rightarrow y$ (Ideal price prediction formula)
 f is unknown

Data: $D = \{(x_1, y_1), \dots, (x_N, y_N)\}$ (Historical records)

Approximation Function (or Model $\hat{f} \in F$): $\hat{f} : X \rightarrow y$ (Min. in-sample/train error)

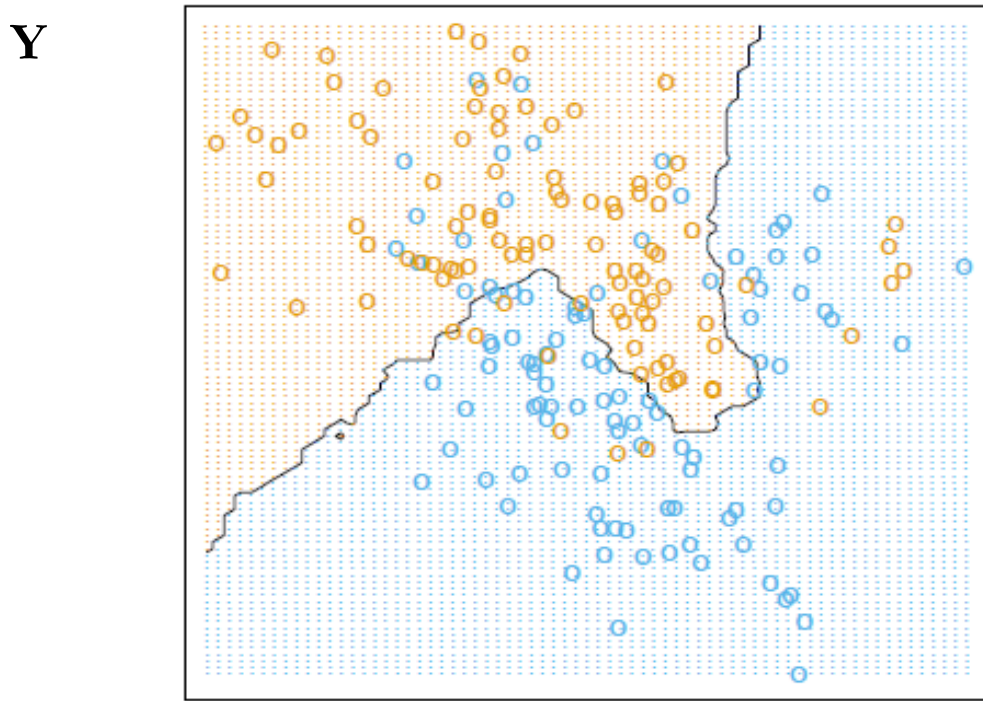
where in-sample error is $\frac{1}{N} \sum_{n=1}^N \text{Loss}(\hat{f}(x_n) \neq f(x_n))$

Goal of Machine Learning: $\min_{\hat{f}} E[\text{Loss}(\hat{f}(x_{\text{new}}) \neq f(x_{\text{new}}))]$ (Min. out-of-sample/test error)

Machine Learning

- Example: Simple Classification With Traditional Machine Learning Methods on Nonlinear-Pattern Data

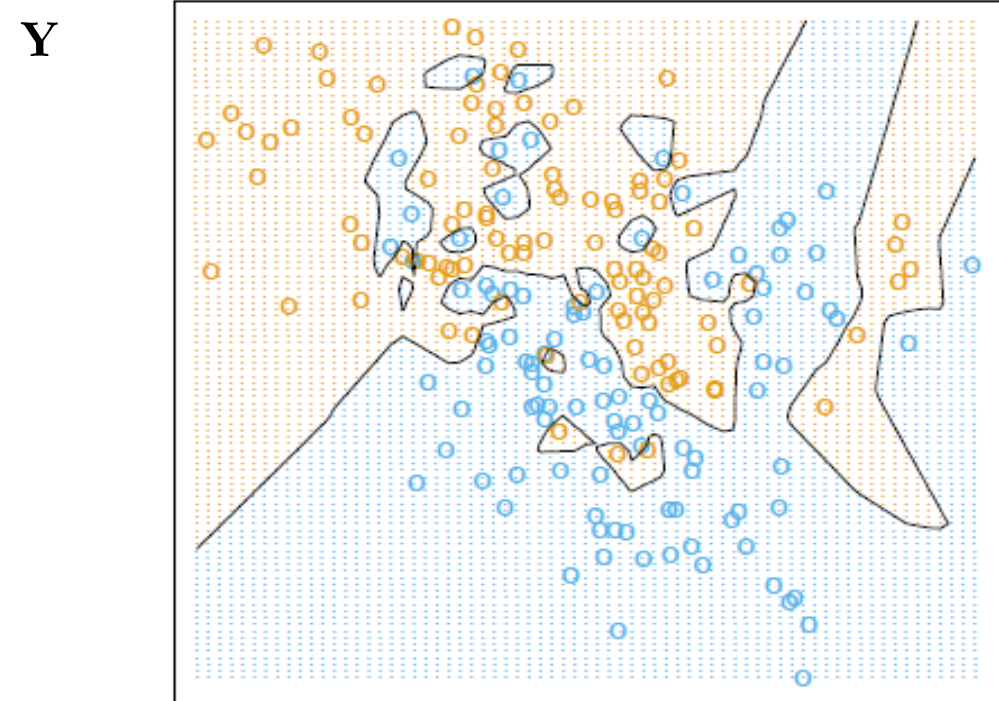
Underfit: Bias



From ESL

X

Overfit: Variance



From ESL

X

Traditional Machine Learning

➤ Generalization Error Bound on Test Data (i.e., Out-of-sample):

$$\Pr\left\{\left|\frac{1}{N}\sum_{n=1}^N \text{Loss}(\hat{f}(x_n) \neq f(x_n)) - E[\text{Loss}(\hat{f}(x_{\text{new}}) \neq f(x_{\text{new}}))]\right| > \varepsilon\right\}$$

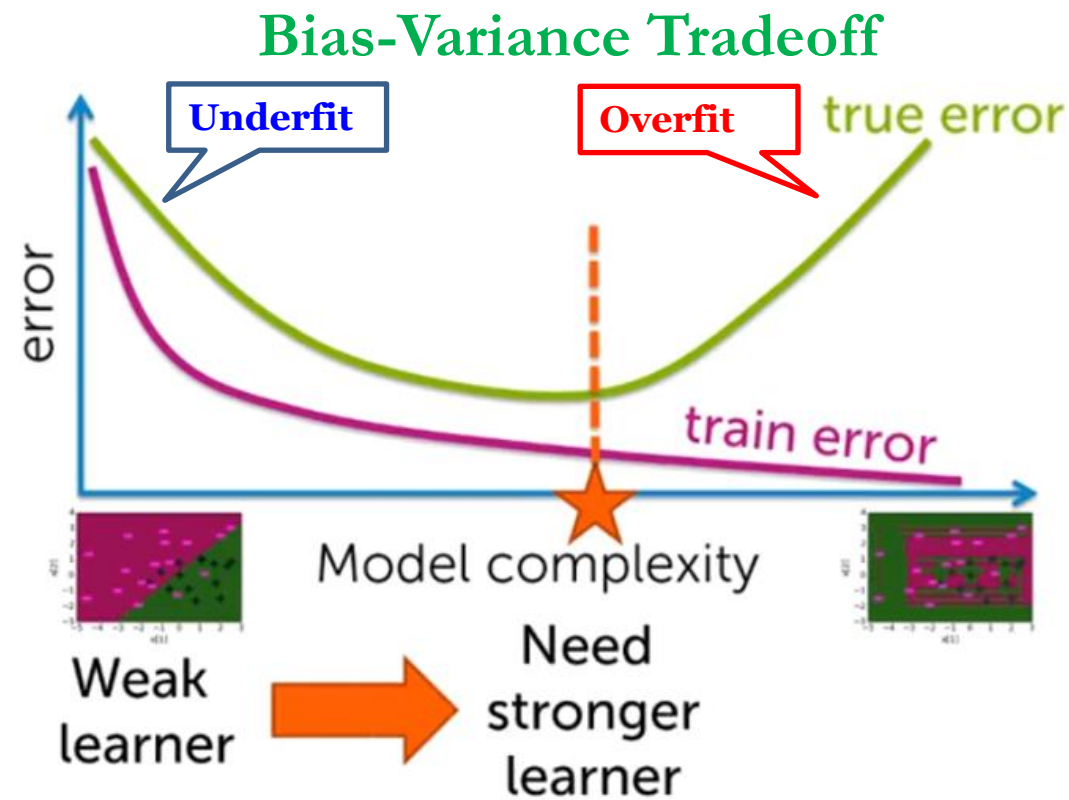
$$= \Pr\{|\text{InSampleError} - \text{OutOfSampleError}| > \varepsilon\}$$

$$\leq \frac{2M}{e^{2N\varepsilon^2}}$$

- M: reflects the complexity of model
- N: number of training samples
- ε : How much you can tolerate your
- In-sample error to be different from
- Out-of-sample error

♥ \hat{f} : InSampleError ≈ 0

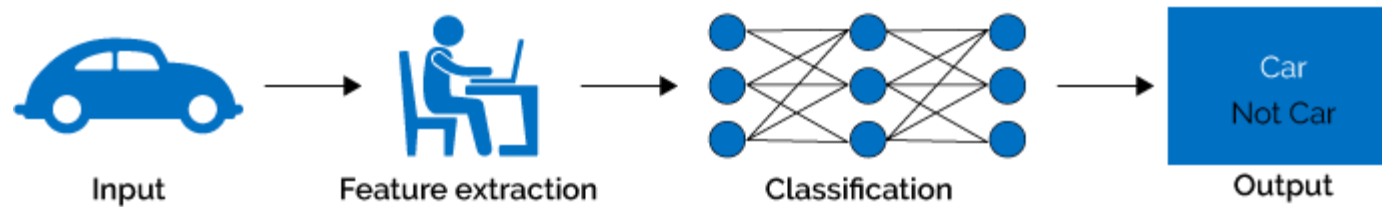
and InSampleError \approx OutOfSampleError



$$E[(y - \hat{f}(x))^2] = (\text{Bias}[\hat{f}(x)])^2 + \text{Var}[\hat{f}(x)] + \sigma^2$$

Traditional Machine Learning

Machine Learning



➤ Human brain is so great in learning, why not making machine learning

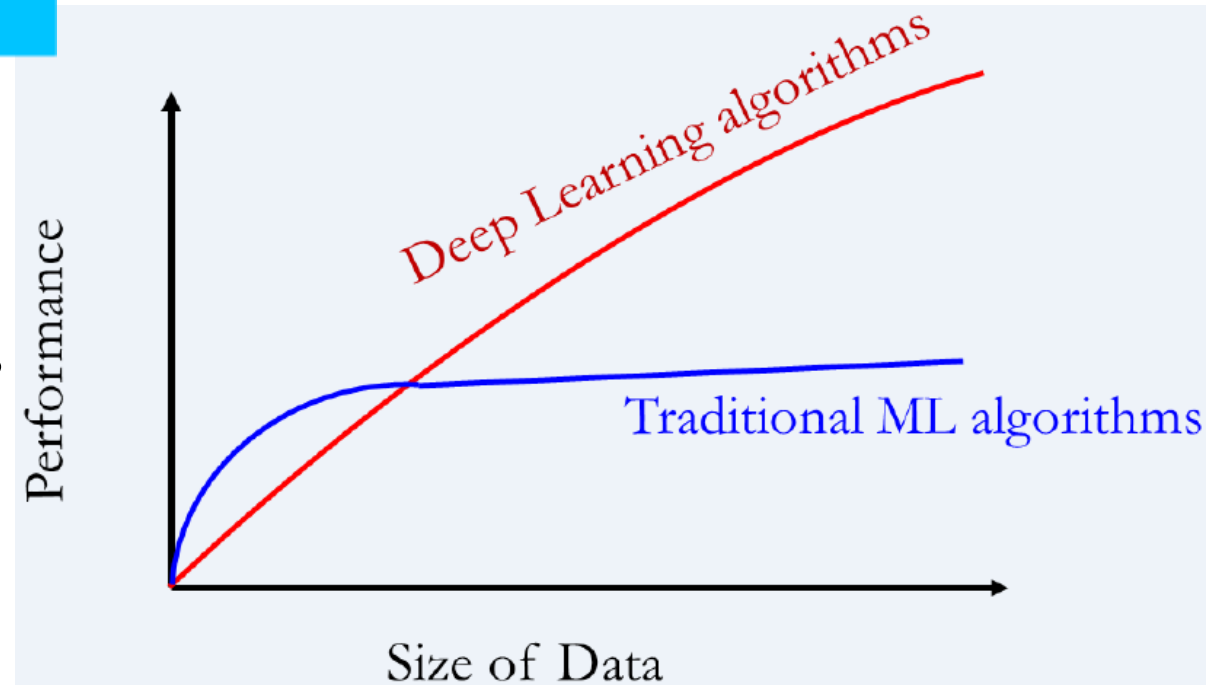
mimic human train ?

Deep Learning



➤ Deep Learning doesn't do different things, **it does things differently**

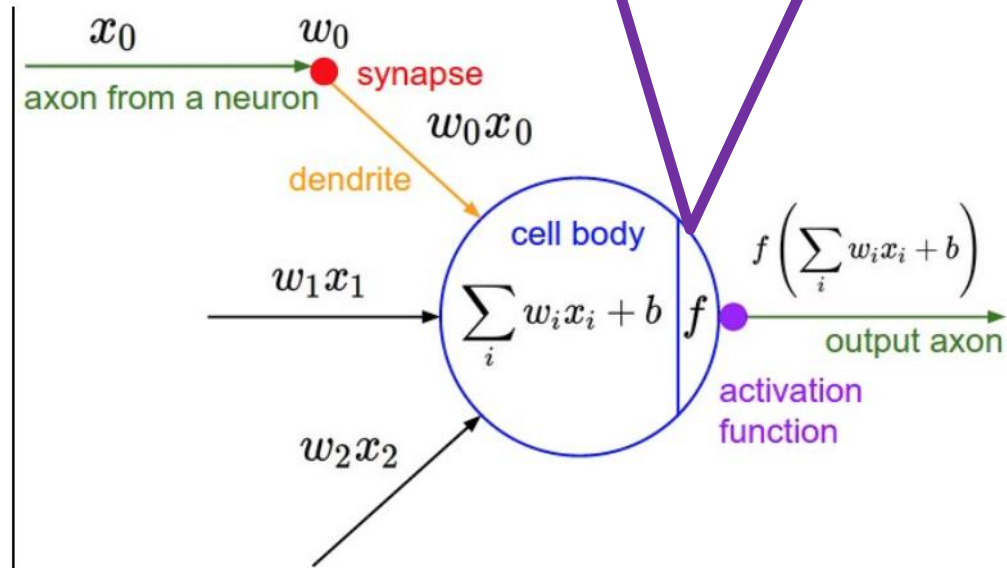
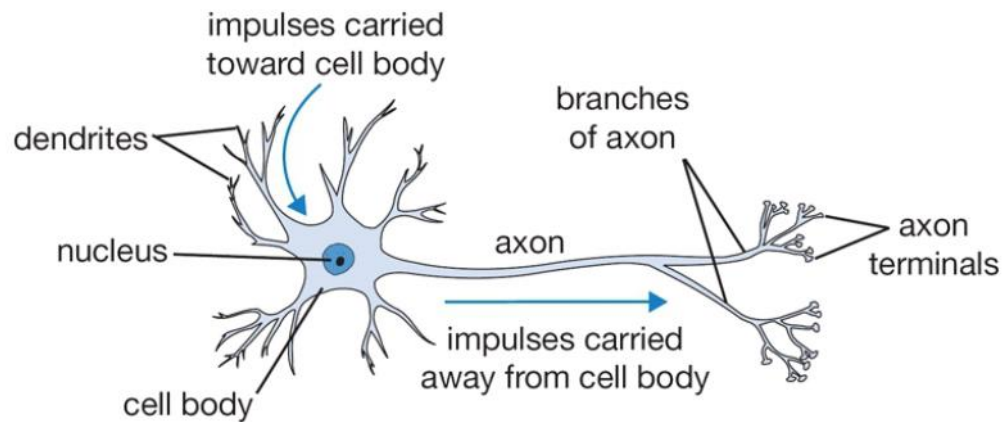
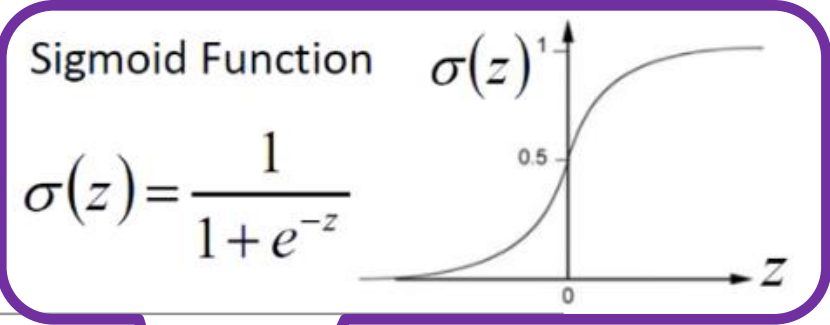
➤ Deep Learning is self-adaptive



Deep Learning: Basics

Deep Learning

- How to Make Machine Learning Mimic Human Brain ?
- Biological View
 - Neuron (or Perceptron): Information Reception, Process and Transfer
 - Hierarchical Structure: Multi-Layer and Multi-Level Inputs
 - **Activation Function (Nonlinear):** Nonlinear Reaction to Inputs

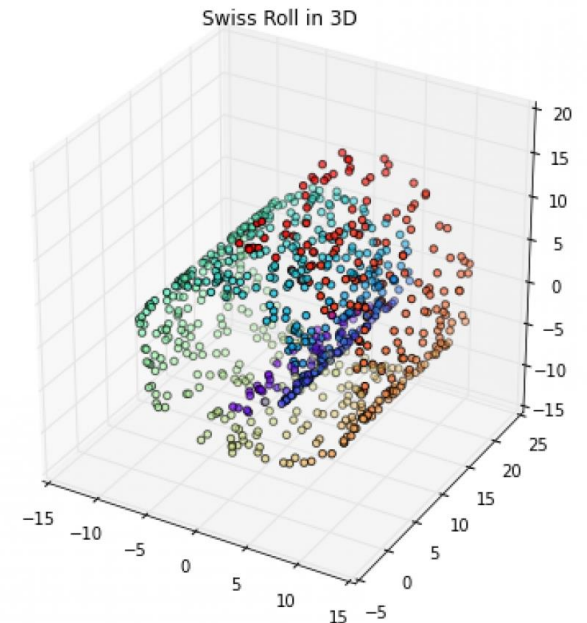
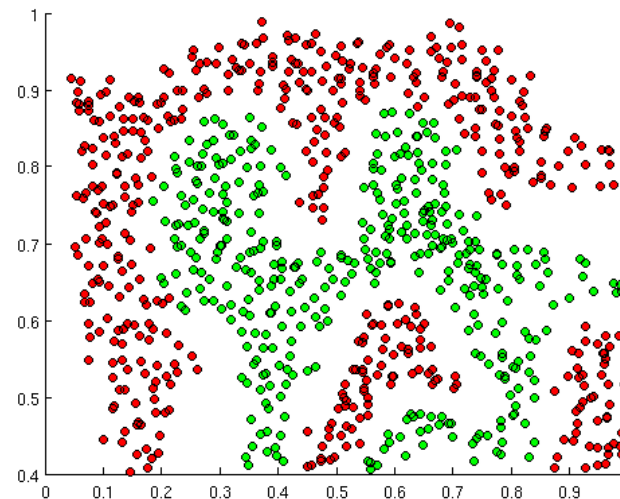
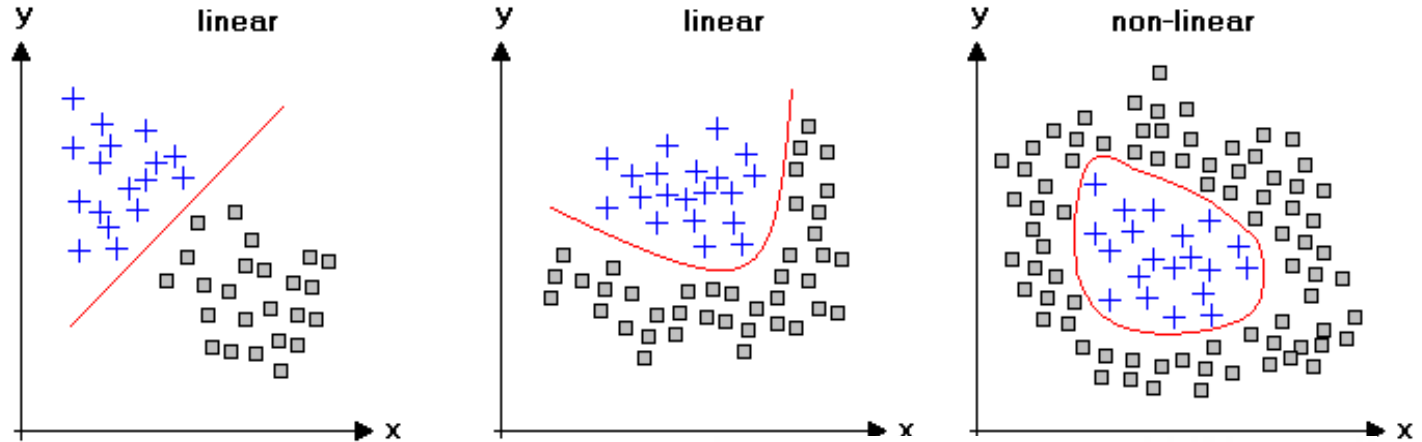


A cartoon drawing of a biological neuron (left) and its mathematical model (right).

Neural Networks: Activation Function

- How do neural networks learn nonlinear data pattern better ?
- Nonlinear Activation Function

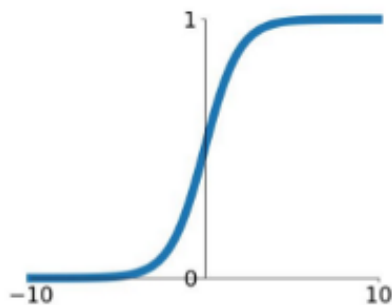
Nonlinear Activation functions are really important for NN to learn and make sense of something really **Complicated** and **Non-linear complex functional mappings between the inputs and outputs**. They introduce **non-linear properties** to our NN.



Activation Functions

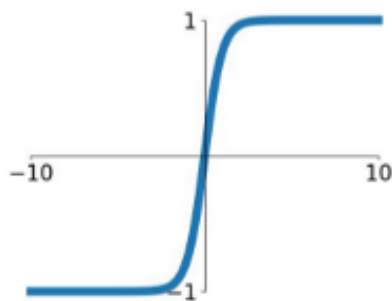
Sigmoid

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



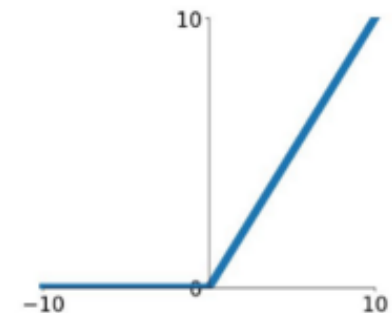
tanh

$$\tanh(x)$$



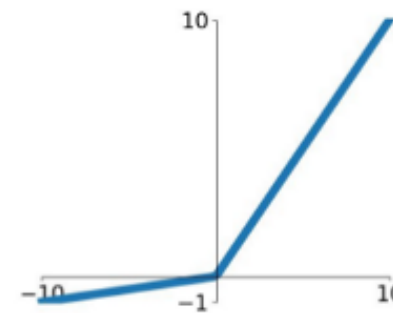
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

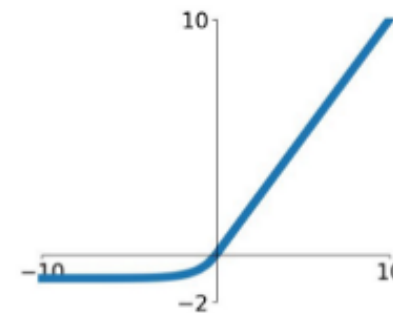


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

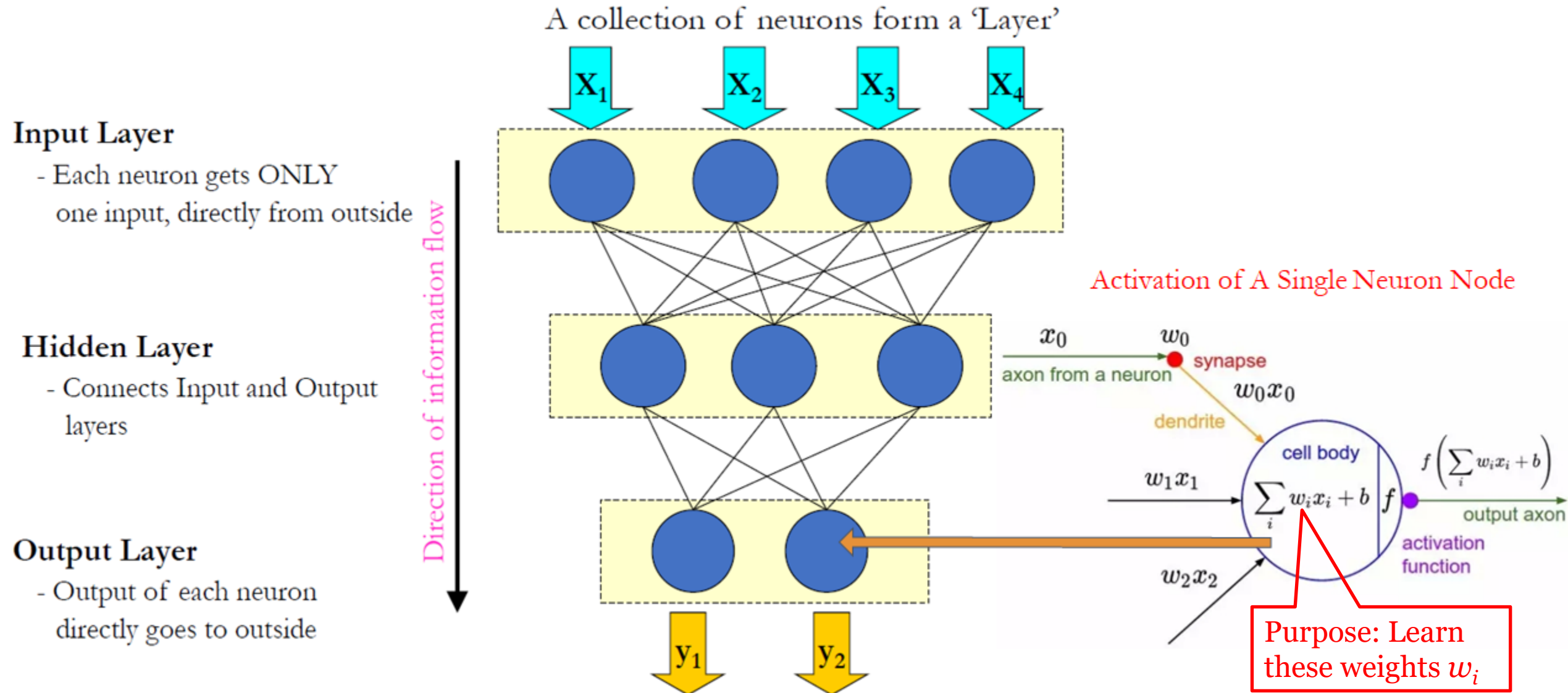
ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



Neural Networks: Structure

- Structure: Interconnecting Multi-Layers of Neurons/Perceptrons

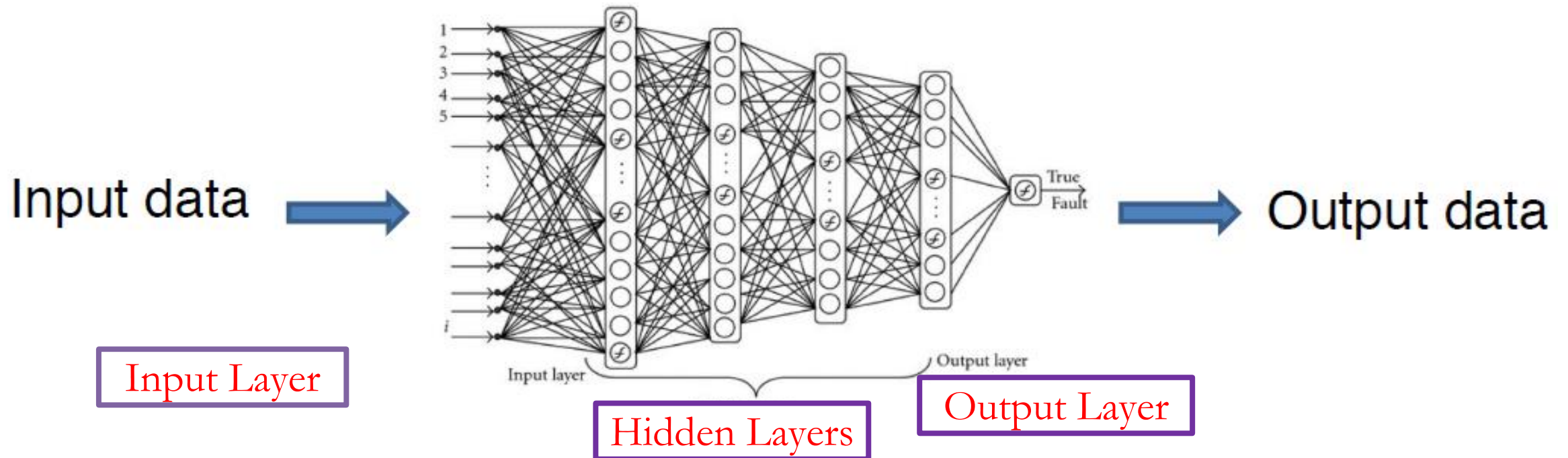


Neural Networks: Deep Neural Networks

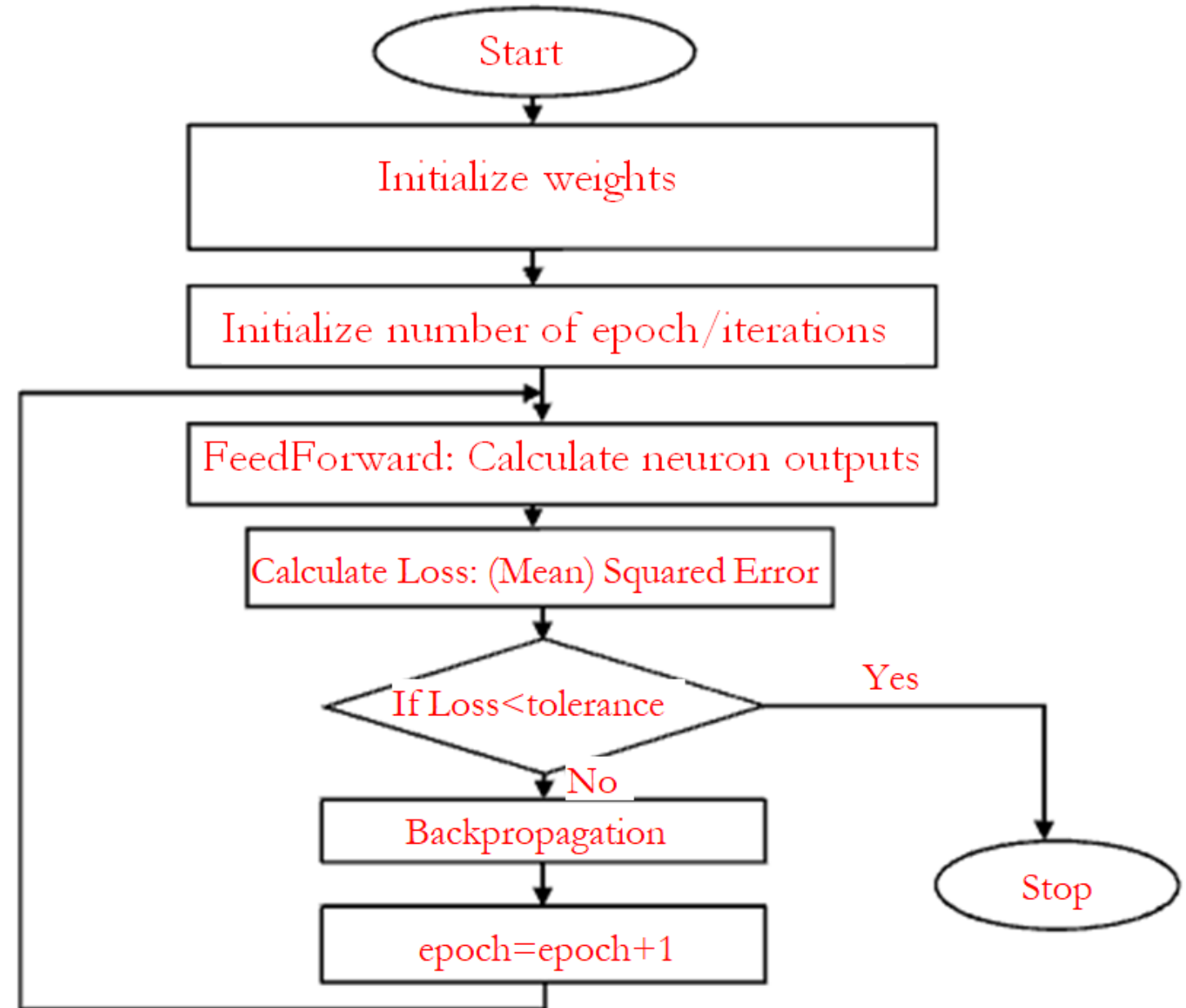
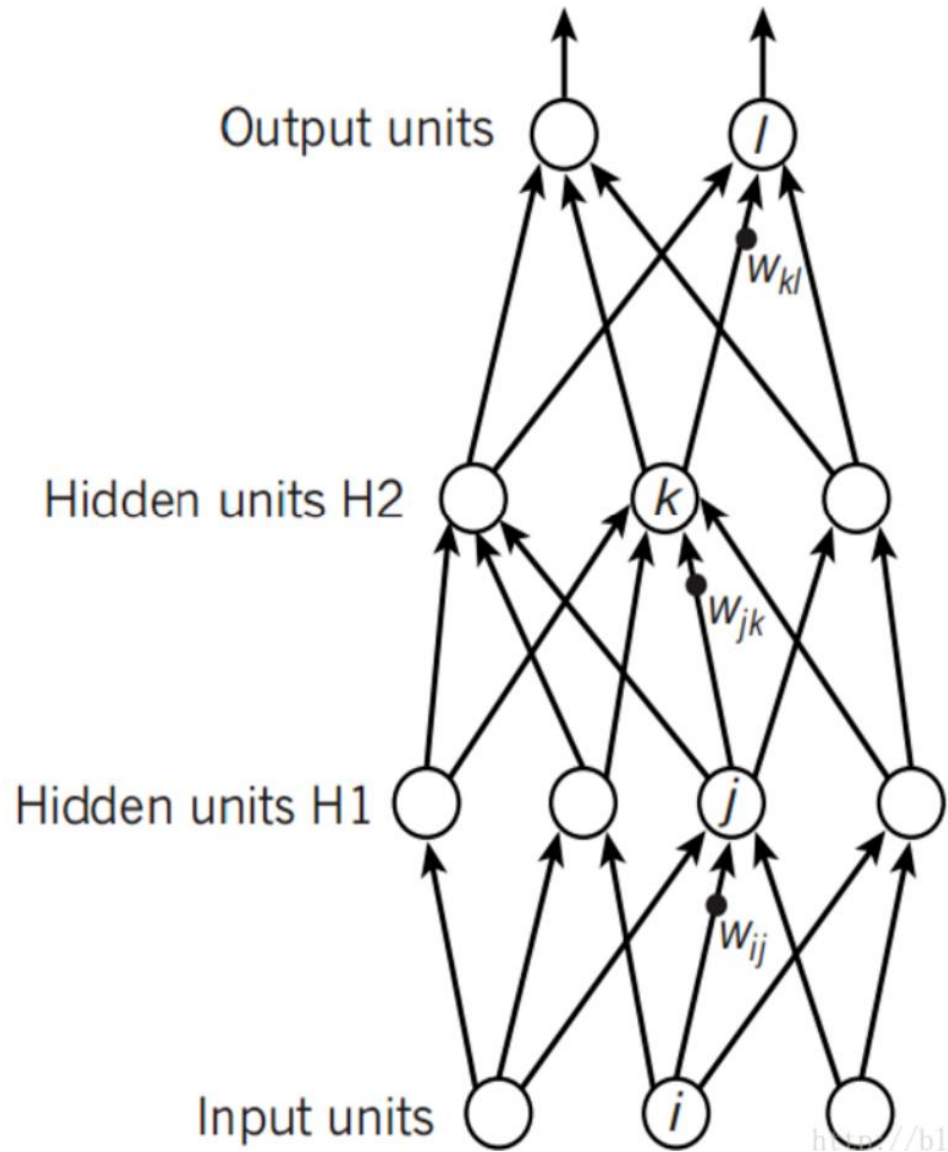
➤ Deep Neural Networks

- Structure: Interconnecting Multi-Layers of Neurons/Perceptrons
- Multiple hidden layers

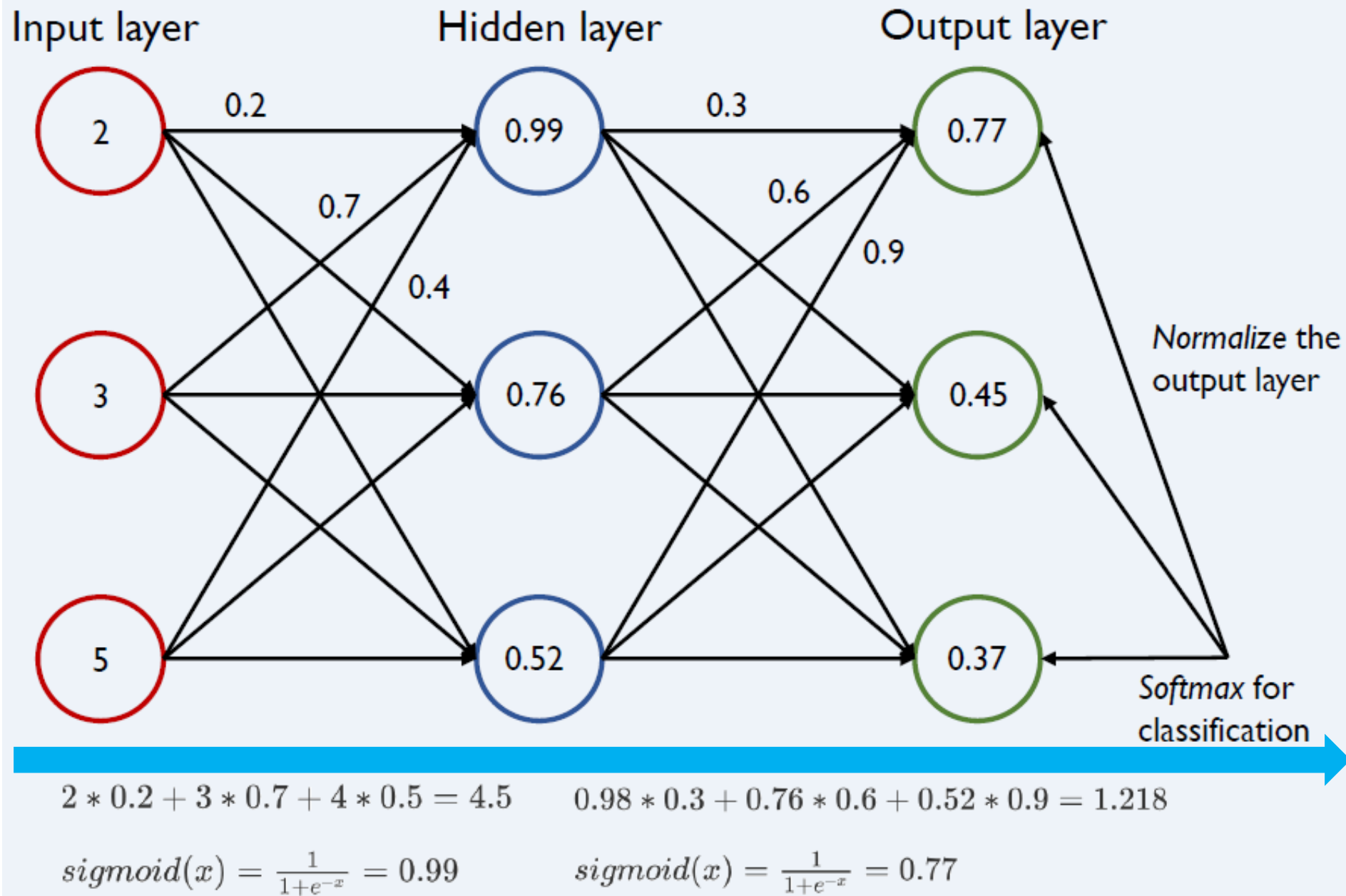
NN (perceptron) consists of three layers:



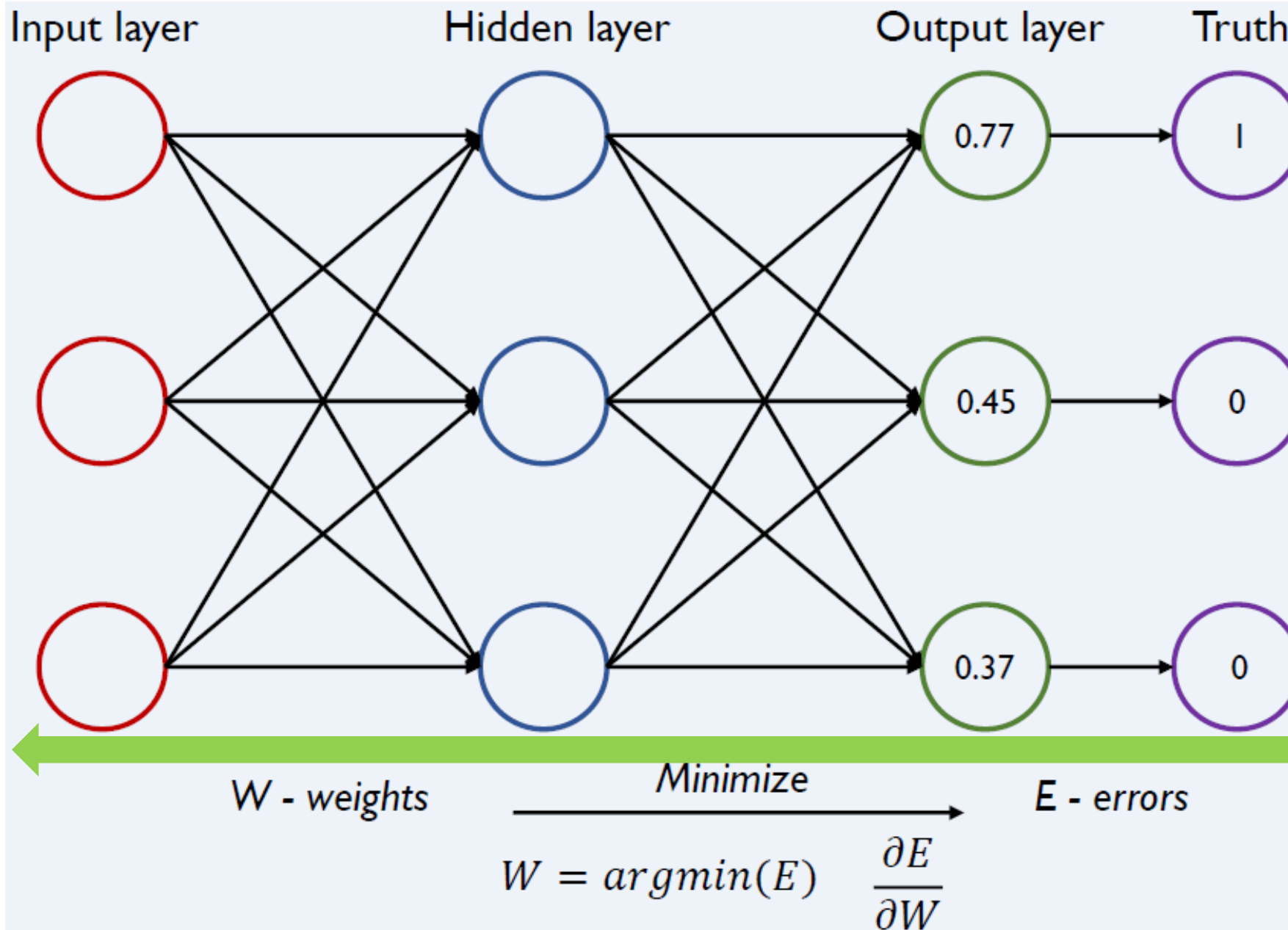
Neural Networks: Training Procedures



Neural Networks: FeedForward



Neural Networks: BackPropagation



Q: Why do we need backpropagation?

- Reduce overall/total errors after updating weights
- Go back to update weights/parameters (**Gradient Descent**)

Neural Networks: BackPropagation

➤ Gradient Descent

Update Rule:

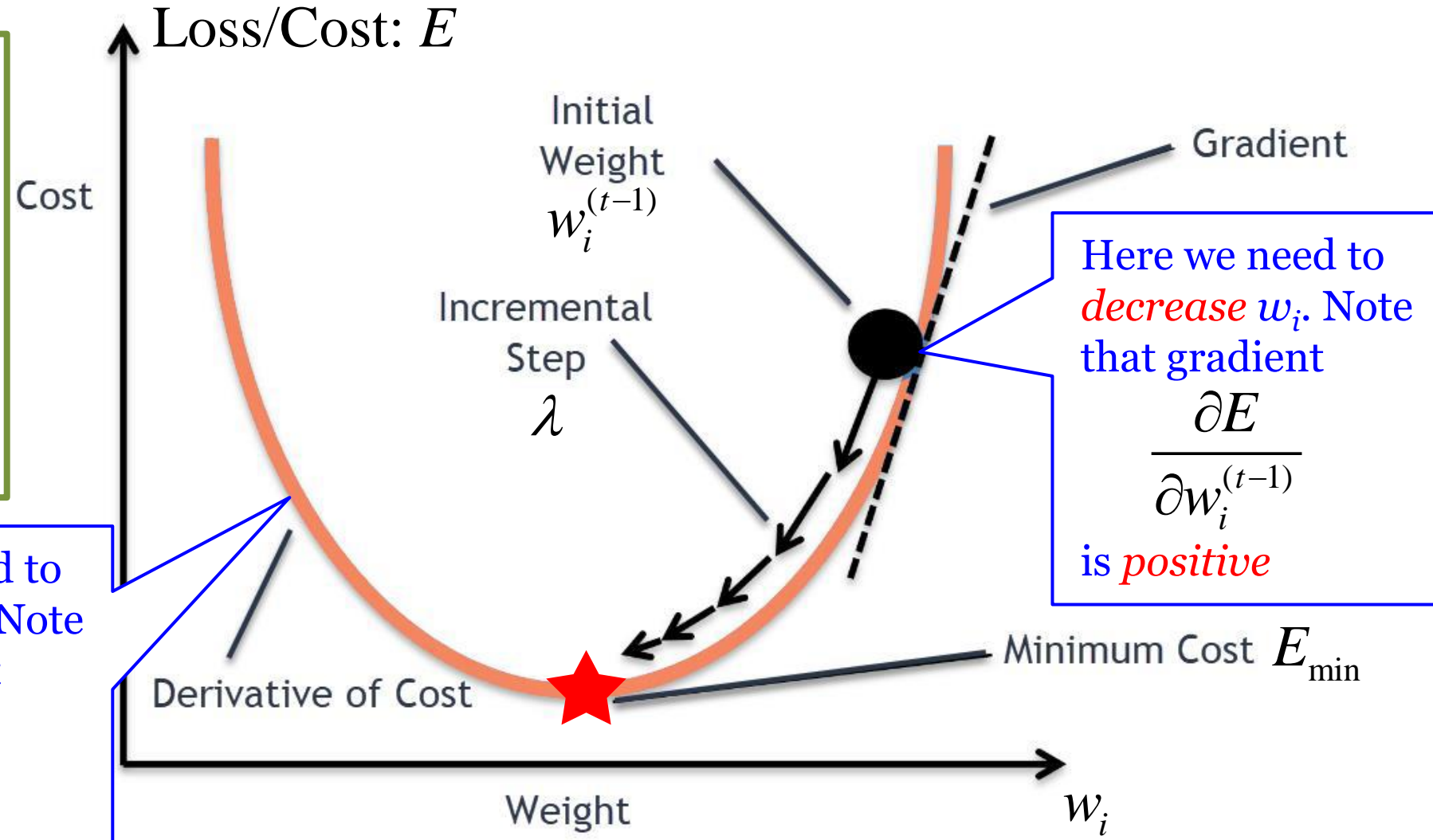
Move in the direction opposite to the gradient direction, by a step with rate λ :

$$w_i^{(t)} \leftarrow w_i^{(t-1)} - \lambda \times \frac{\partial E}{\partial w_i^{(t-1)}}$$

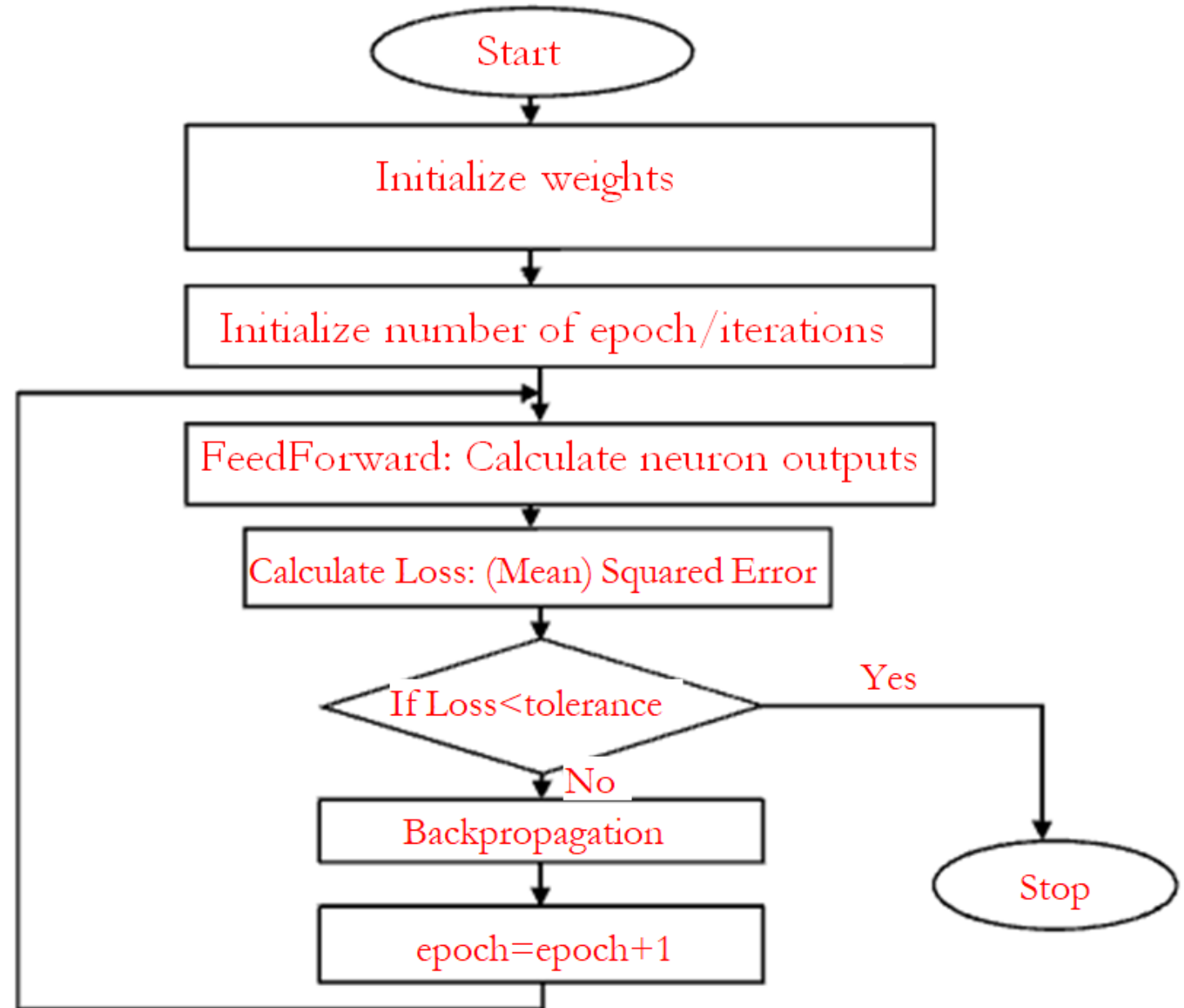
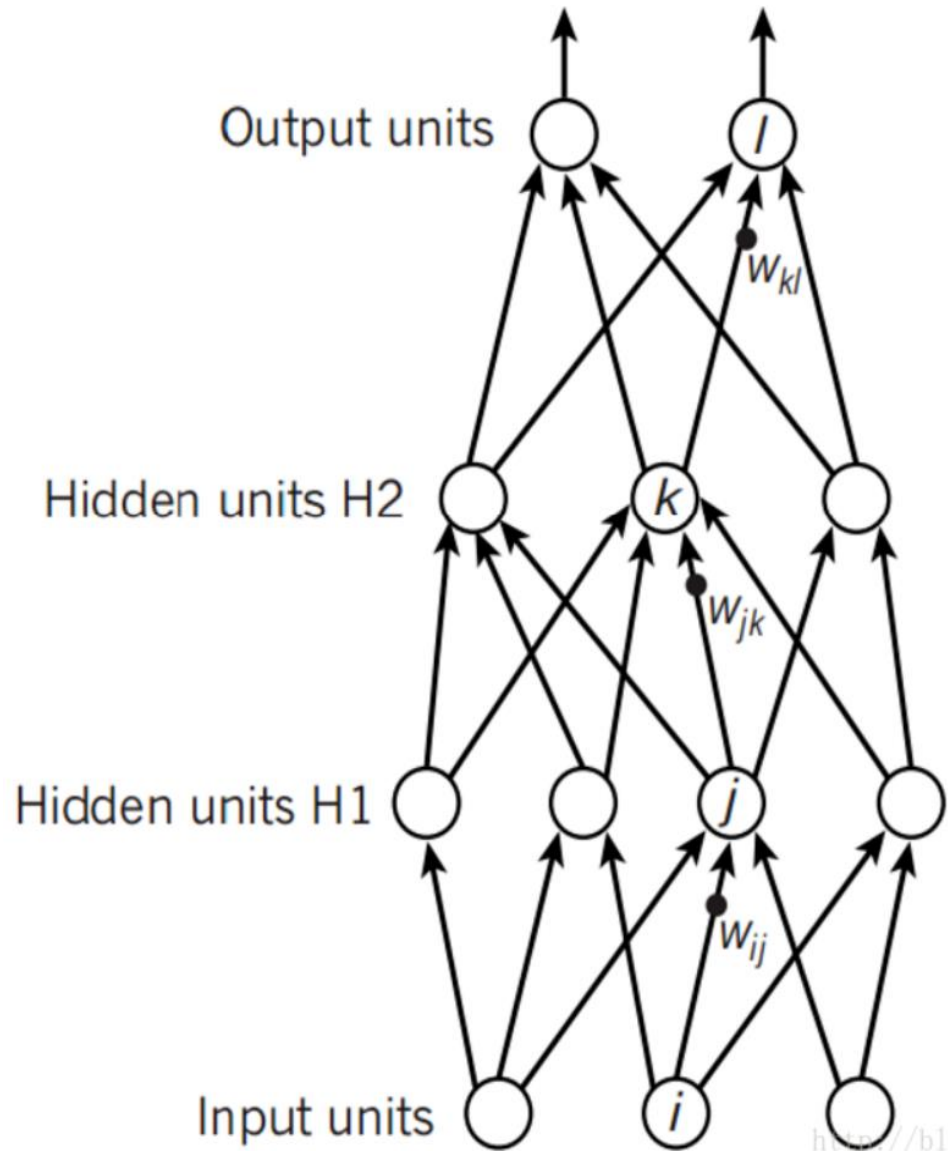
Here we need to *increase* w_i . Note that gradient

$$\frac{\partial E}{\partial w_i^{(t-1)}}$$

is *negative*

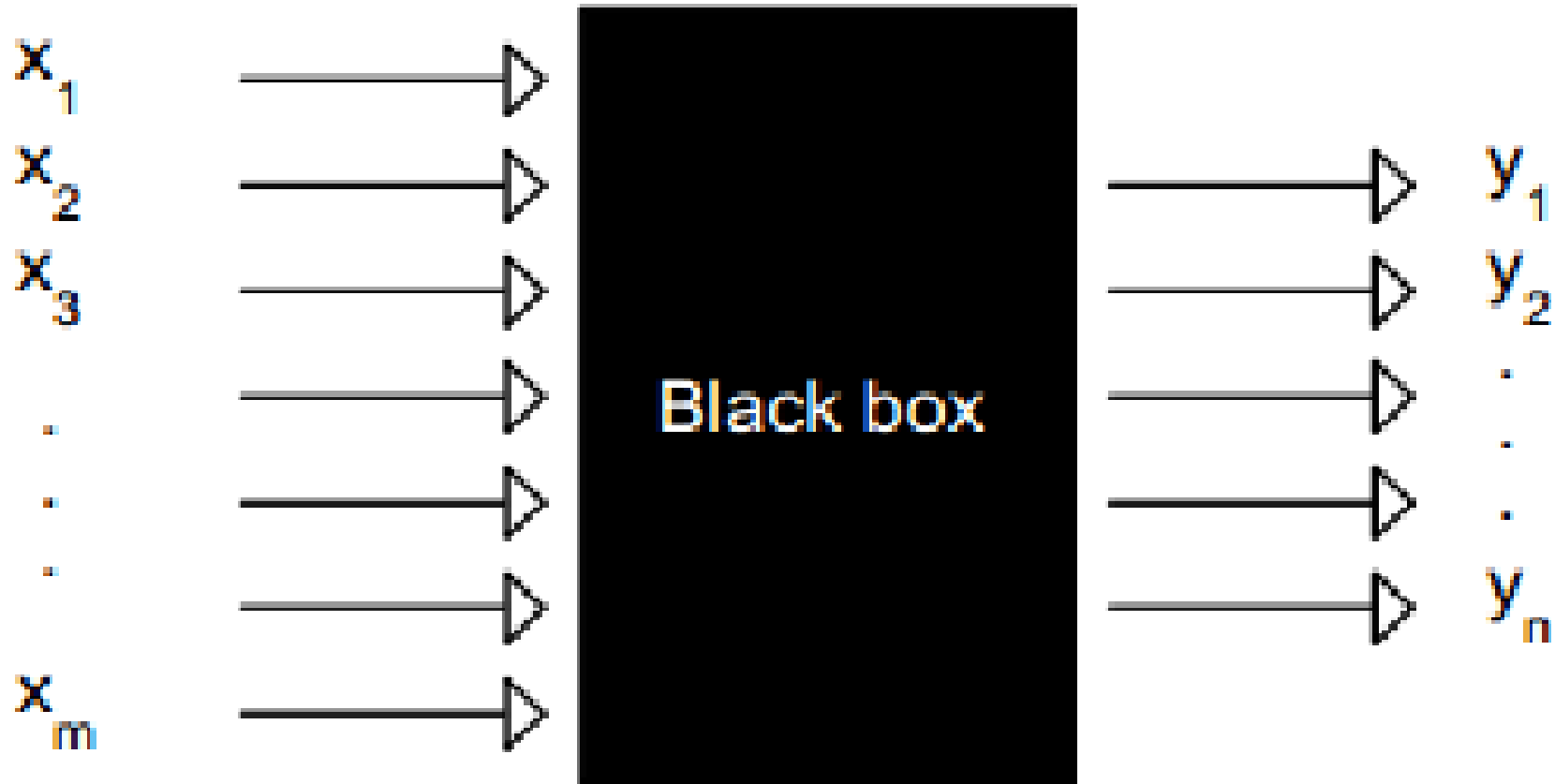


Neural Networks: Summary



Neural Networks: Limitations

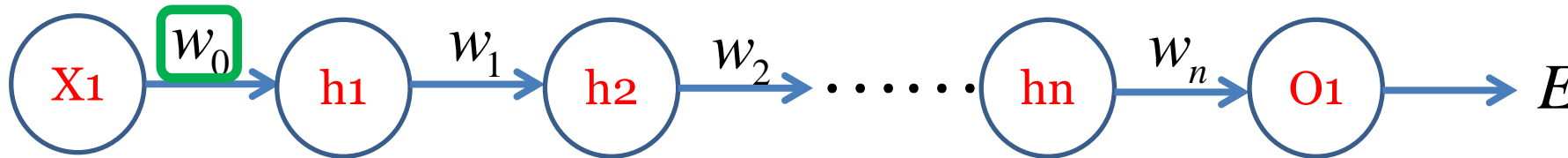
Neural Networks: Limitation I



- Good at making prediction, but bad at making interpretation
- What happens in the “Black Box”? The effectiveness of features ?

Neural Networks: Limitation II

- Gradient Vanishing & Gradient Explosion Update: $w_i^{(t)} \leftarrow w_i^{(t-1)} - \lambda \times \frac{\partial E}{\partial w_i^{(t-1)}}$



➤ Derivative Chain Rule

$$\frac{\partial E}{\partial w_0} = \frac{\partial E}{\partial O_{1_out}} \times \frac{\partial O_{1_out}}{\partial O_{1_in}} \times \frac{\partial O_{1_in}}{\partial h_{n_out}} \times \frac{\partial h_{n_out}}{\partial h_{n_in}} \times \dots \times \frac{\partial h_{1_out}}{\partial h_{1_in}} \times \frac{\partial h_{1_in}}{\partial w_0}$$

- Calculating gradient of weights: $= \frac{\partial E}{\partial O_{1_out}} \times f'(O_{1_in}) \times w_n \times f'(h_{n_in}) \times w_{n-1} \times \dots \times f'(h_{1_in}) \times x_1$

- If we choose activation functions carelessly,

- If gradients are lower than 1 \rightarrow Gradient Vanishing : $\frac{\partial Sigmoid(x)}{\partial x} \in (0,1)$ so $\frac{\partial E}{\partial w_0} \rightarrow 0$

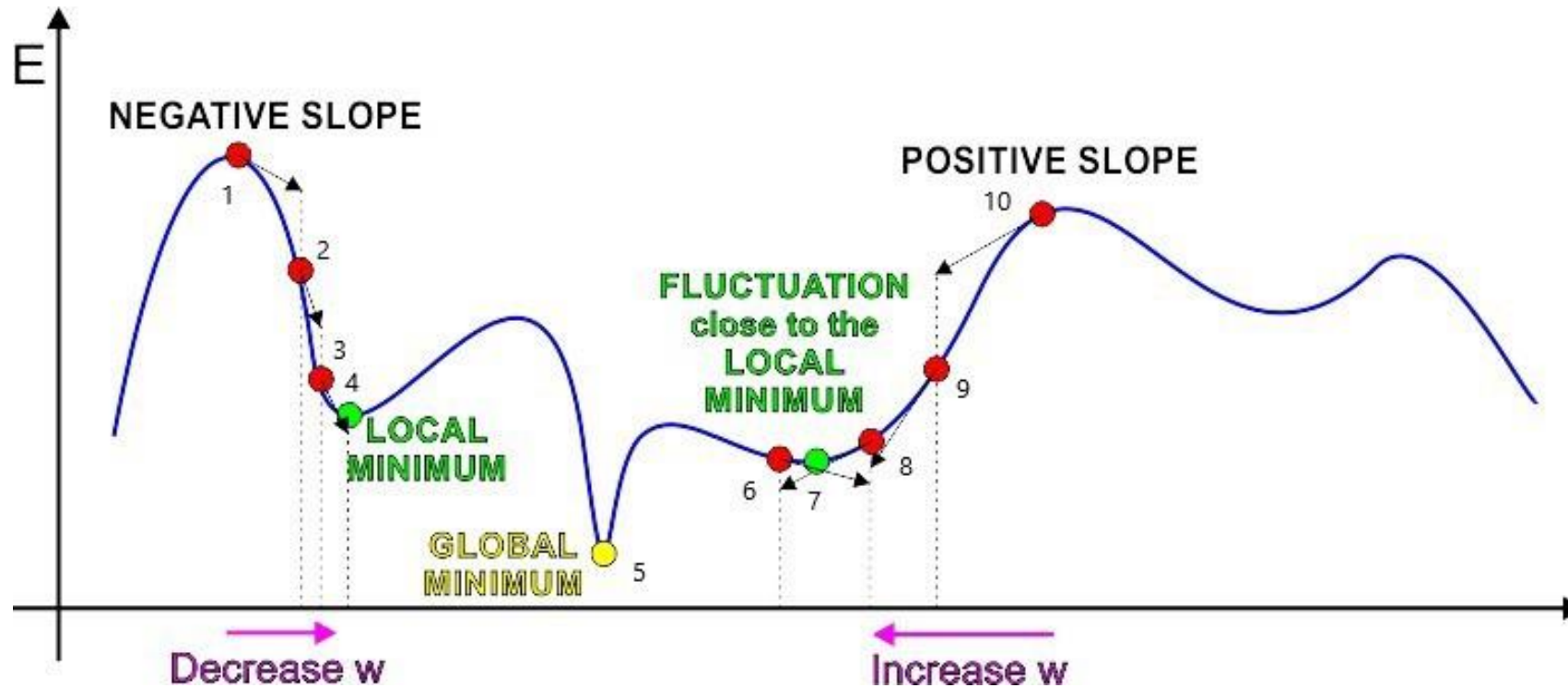
- Solutions: Choose activation functions carefully

- If gradients are greater than 1 \rightarrow Gradient Explosion : $\frac{\partial E}{\partial w_0} \rightarrow \infty$

- Solutions: Gradient Clipping; Regularization on weights

Neural Networks: Limitation III

➤ Optimization With Gradient Descent:



Remember:

Gradient Descent is a greedy algorithm

➤ Solutions:

- Revised GD: Stochastic Gradient Descent (SGD), etc.
- Alternative optimization methods: Simulated Annealing in Boltzmann Machine

Neural Networks: Limitation IV

- Generalization Error Bound on Test Data (i.e., Out-of-sample):

$$\Pr\left\{\left|\frac{1}{N}\sum_{n=1}^N \text{Loss}(\hat{f}(x_n) \neq f(x_n)) - E[\text{Loss}(\hat{f}(x_{\text{new}}) \neq f(x_{\text{new}}))]\right| > \varepsilon\right\}$$

$$= \Pr\{|\text{InSampleError} - \text{OutOfSampleError}| > \varepsilon\}$$

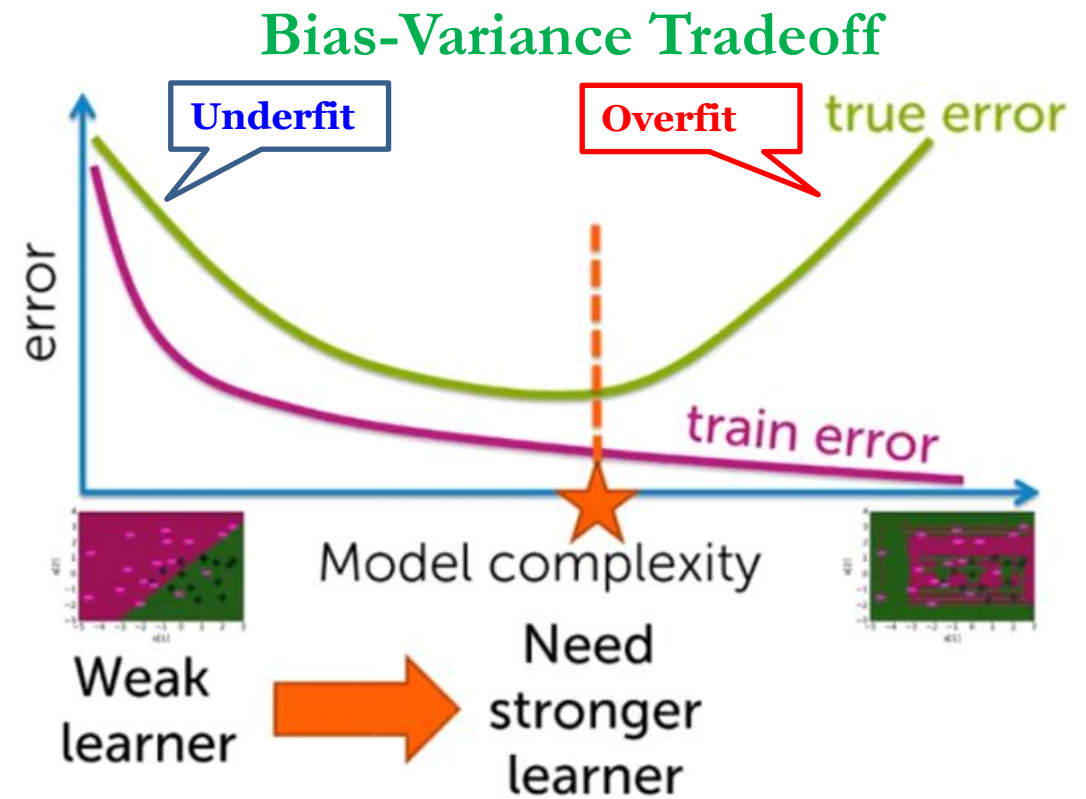
$$\leq \frac{2M}{e^{2N\varepsilon^2}}$$

- ♥ \hat{f} : InSampleError ≈ 0
and InSampleError \approx OutOfSampleError

- Neural Networks may still overfit !

- General Solutions:

- Regularization on weights
- Batch normalization
- Dropout neurons randomly in each layer

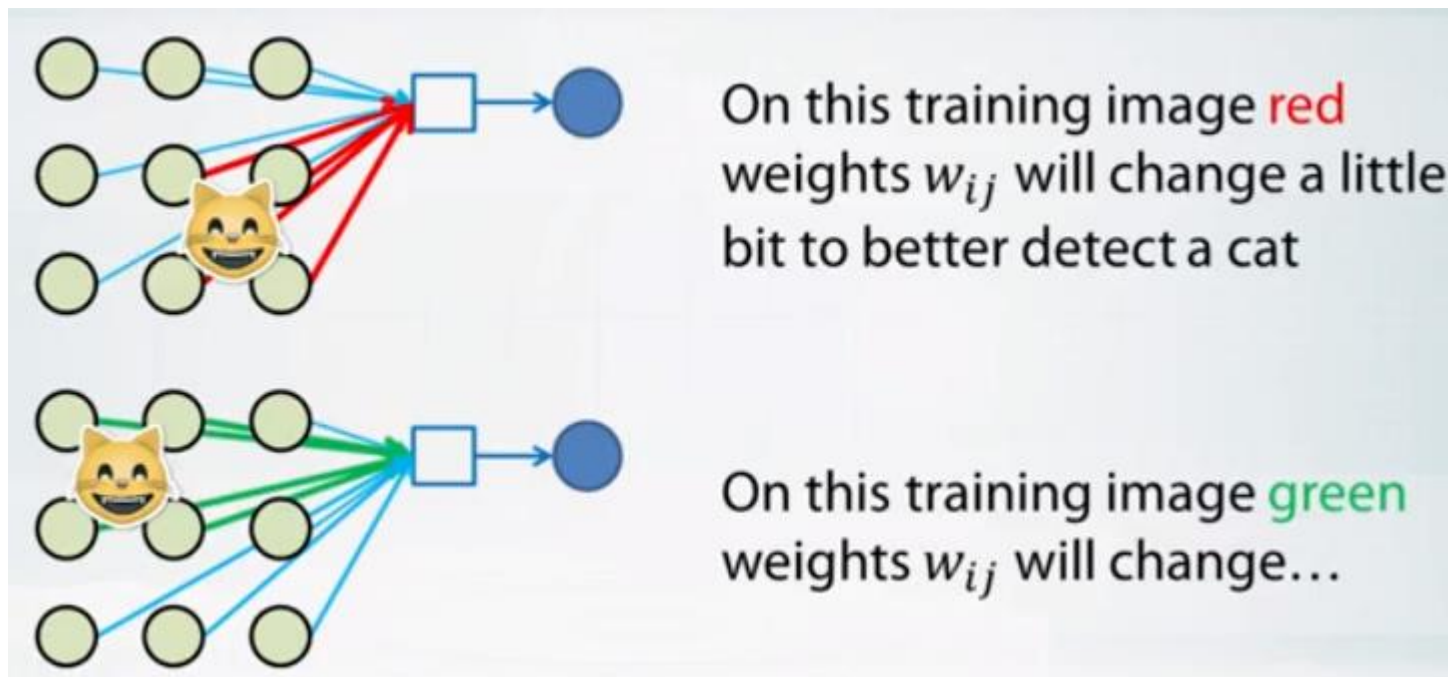


“No Free Lunch”

Neural Networks: Limitation IV

➤ Image Classification:

- What if we use a normal full-connected neural network to do classification?
- We split the whole image into multiple pixels. Each pixel (a value to represent darkness or brightness of color: 0 to 255) is one feature.
- What is wrong here ?

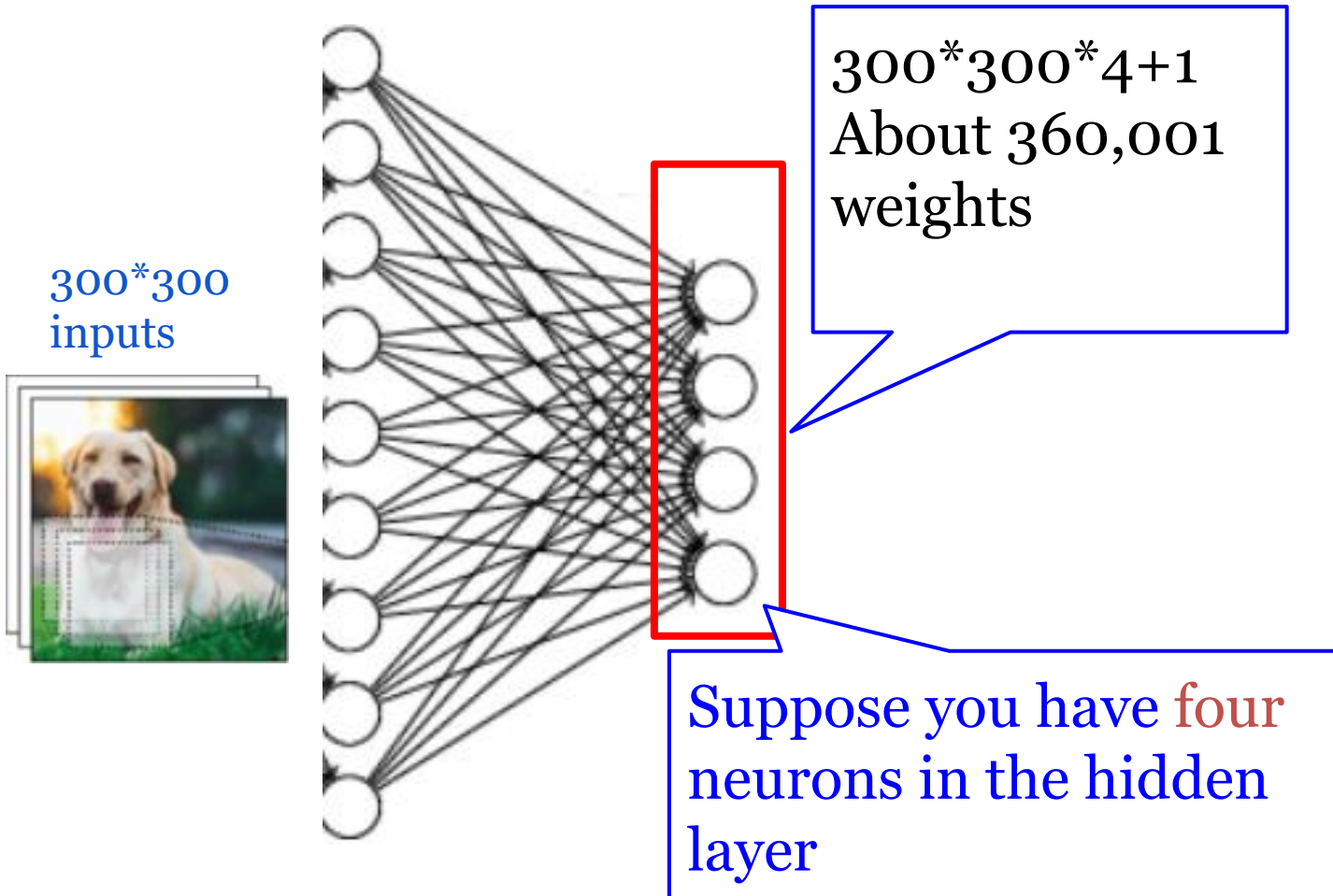


- We don't fully utilize the training data
- What if in test data, the cat is in other areas (e.g., in the centre of the image)?

Neural Networks: Limitation IV

➤ Image Classification:

Fully Connected Normal NN



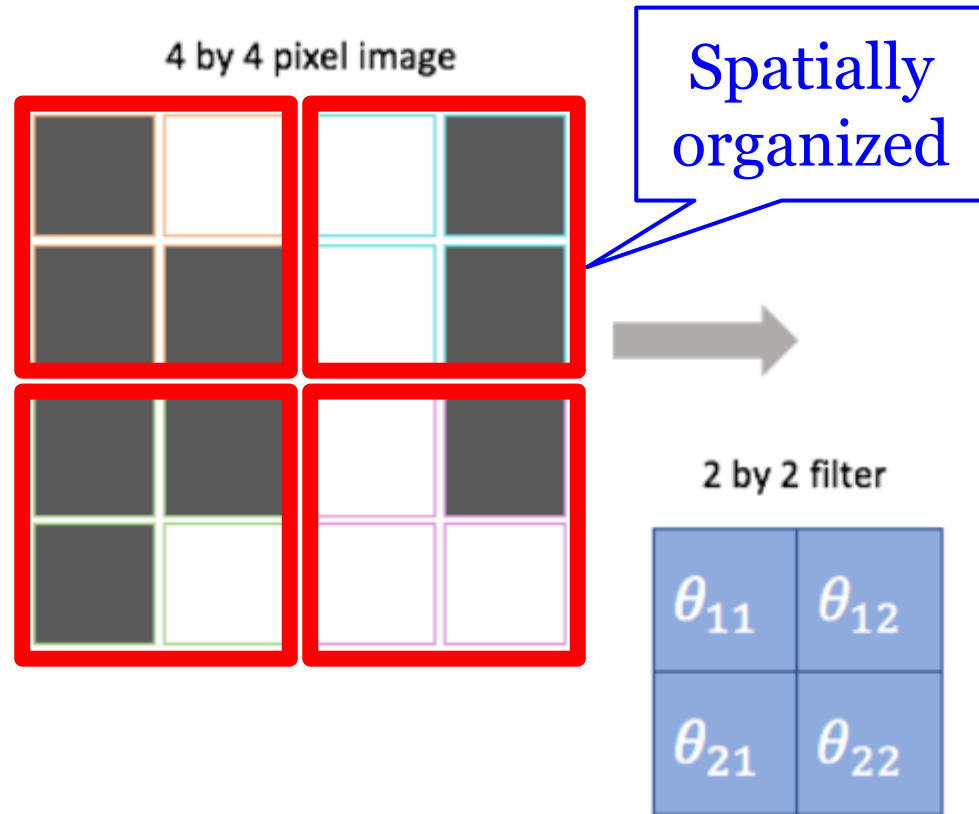
Fully-Connected Neural Network in Computer Vision:

- Slow
- Overfit

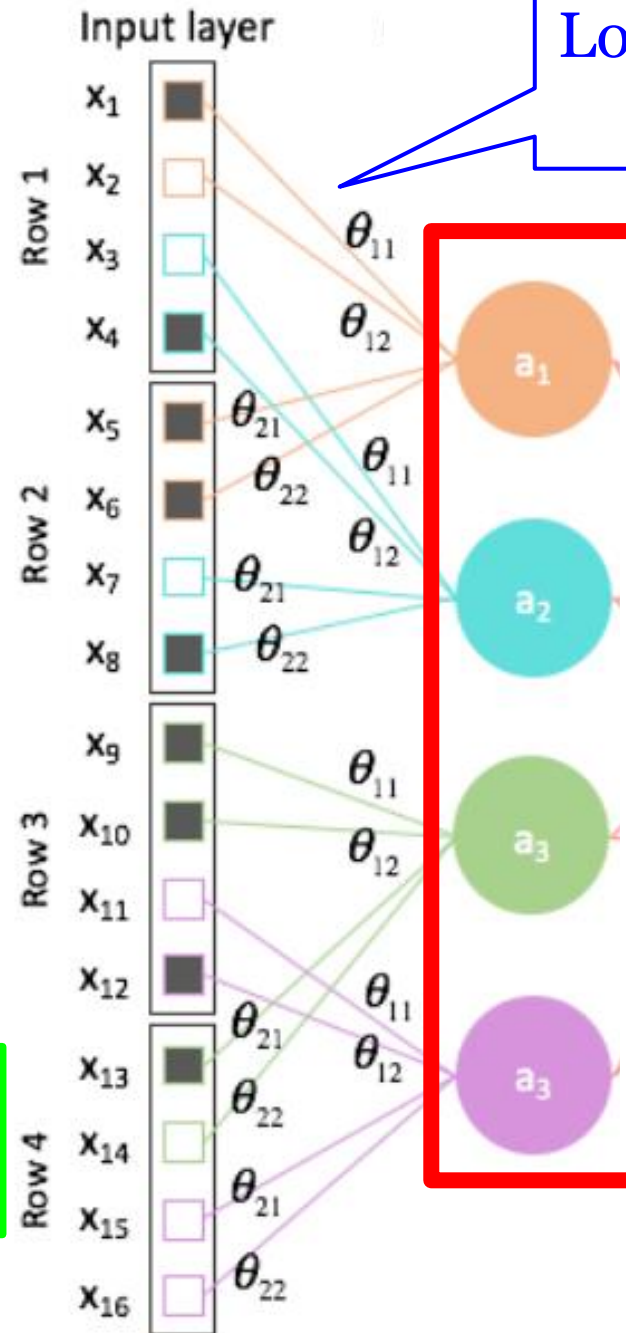


Neural Networks: CNN

➤ Convolutional Neural Networks:



Convolution Layer (Filter/Kernel):
Suppose Slide Stride=2



Local/Sparse Connectivity:
Sharing weights

Because interesting features (edges) can happen at anywhere in the image.



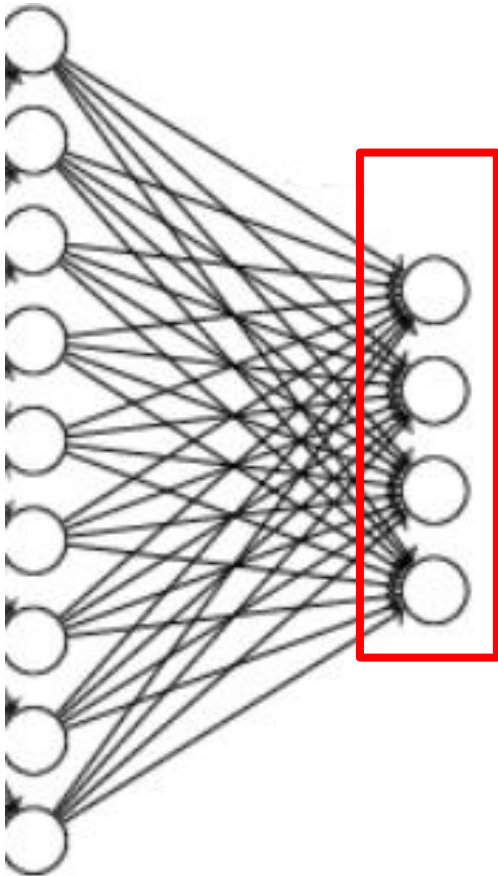
Feature Map:
with new “pixels”

Neural Networks: CNN

➤ Convolutional Neural Networks

Fully Connected Normal NN

300*300
inputs

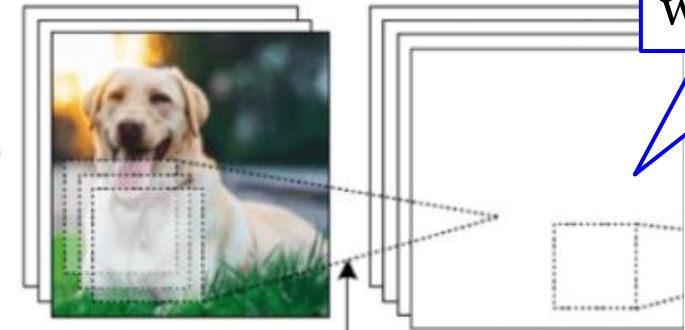


$300*300*4+1$
About 360,001
weights

Suppose you
have **four**
neurons in the
hidden layer

300*300
inputs

CNN



Convolution

Pool

$(5*5+1)*4$
Only 104
weights

Suppose you have
four $5*5$ kernels in
the convolution layer

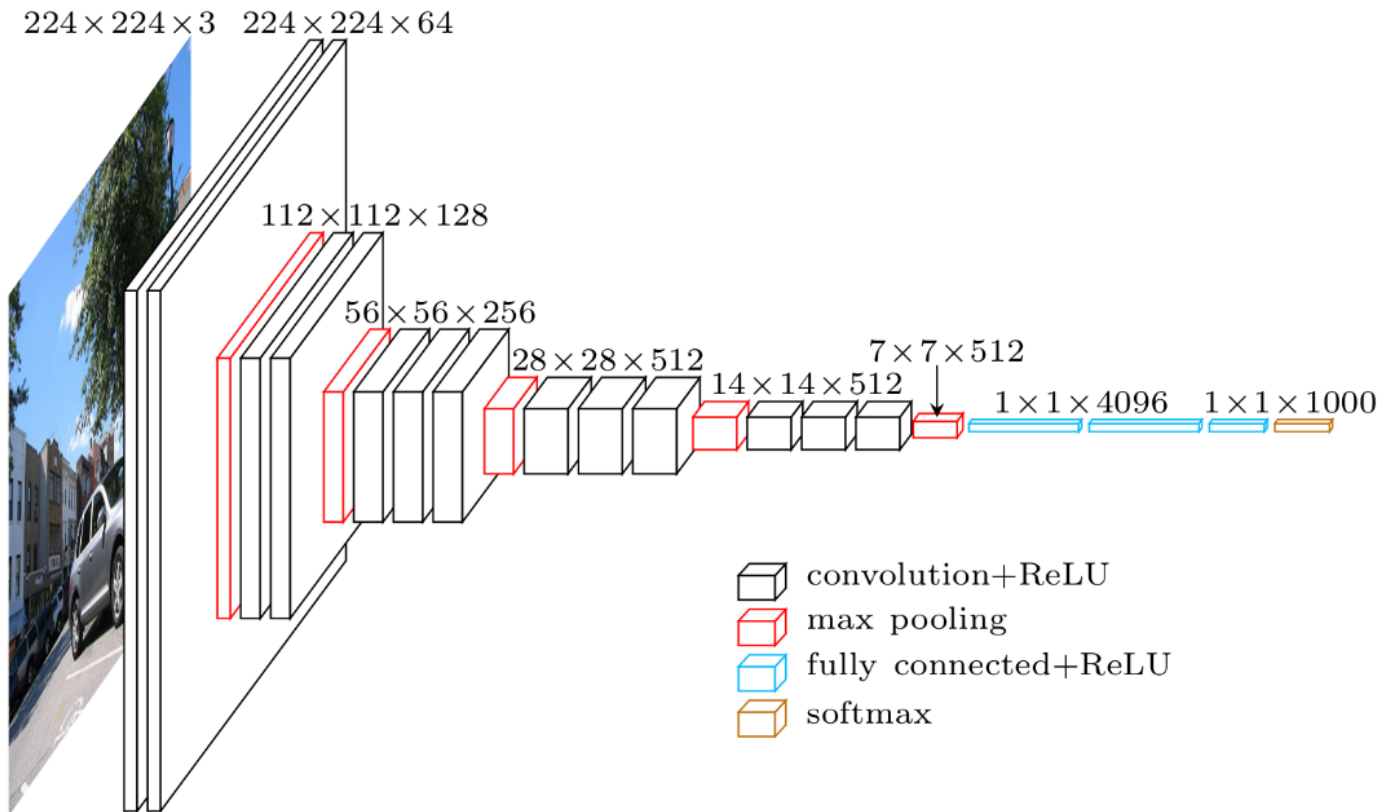
- Recurrent Neural Networks (RNN)
 - Time-dependent Neuron States
 - Long-Short-Term-Memory (LSTM)
- Restricted Boltzmann Machine (RBM)
- Deep Belief Network (DBN)
- Auto-Encoder
- More...

Neural Networks: Applications in IS Research

Neural Networks: Applications

➤ Academic Research

- Shunyuan Zhang, Dokyun Lee, Param Vir Singh, Kannan Srinivasan. *How Much is an Image Worth? Airbnb Property Demand Analytics Leveraging A Scalable Image Classification Algorithm*. Working Paper. (https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2976021)



- Question: What is the effect of joining Airbnb photography program (i.e., verified photos)?
- Training: Label image quality on Amazon Mechanical Turk
- CNN (VGG-16): Image quality and (12) interpretable image features

Neural Networks: Applications

➤ Computer Vision

- Image Classification, Face Recognition, Object Detection
- Video Mining

➤ Natural Language Processing

- Textual Feature Extraction (e.g., Content features, sentiment/semantic features, etc.)

➤ Time Series Forecasting

➤ Speech Recognition

➤ More...

Remember:
Earn money and Buy GPUs

In God we trust, all others bring data

-----William Edwards Deming (1900-1993)

Thank You !