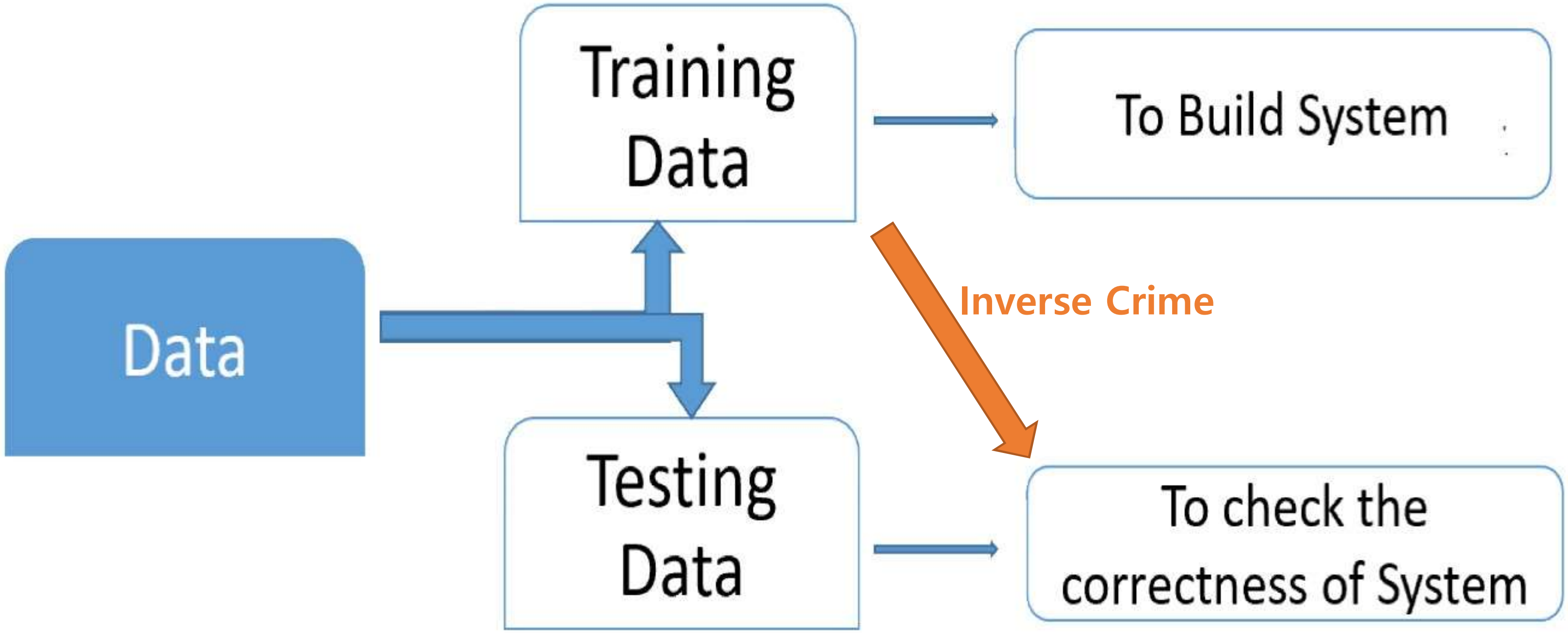
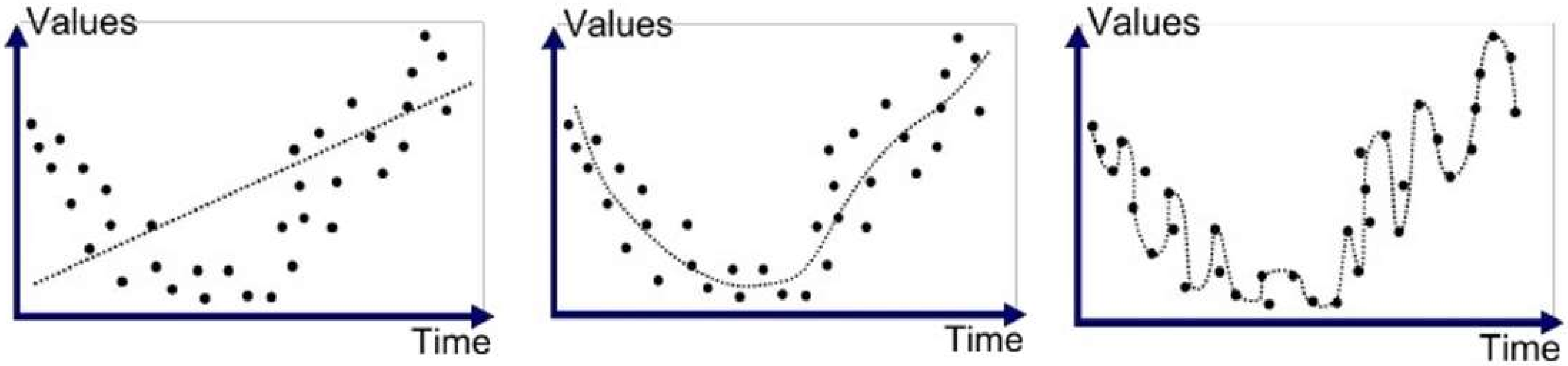
MDL Ch. 1 Handout

Inverse Crime



Overfitting (1D)



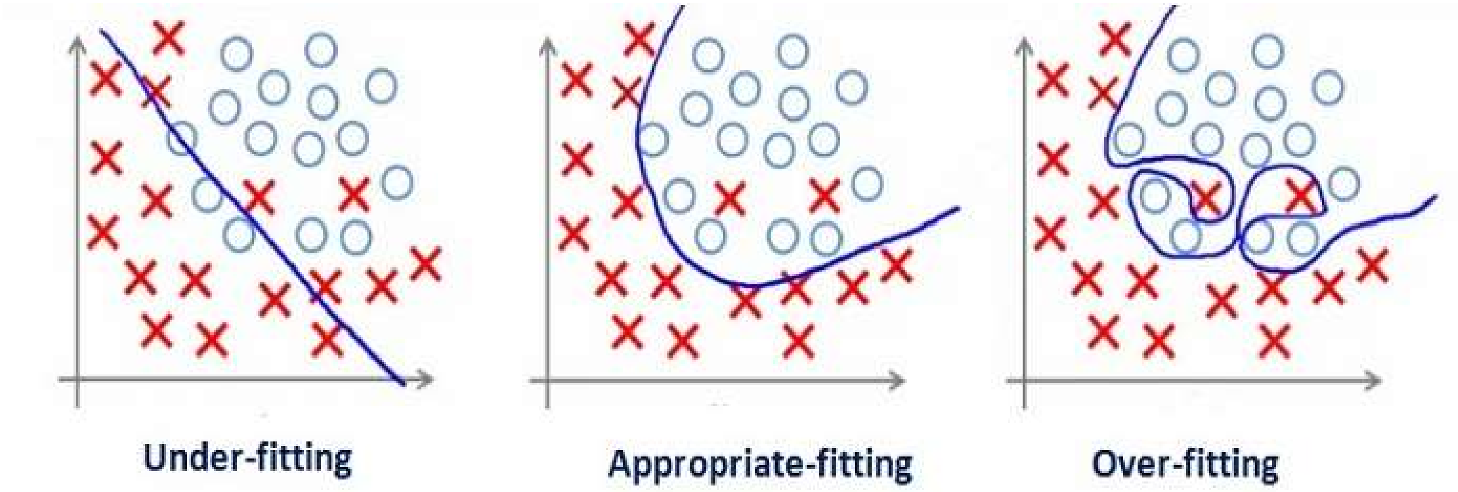
Underfitted

Good Fit/Robust

Overfitted

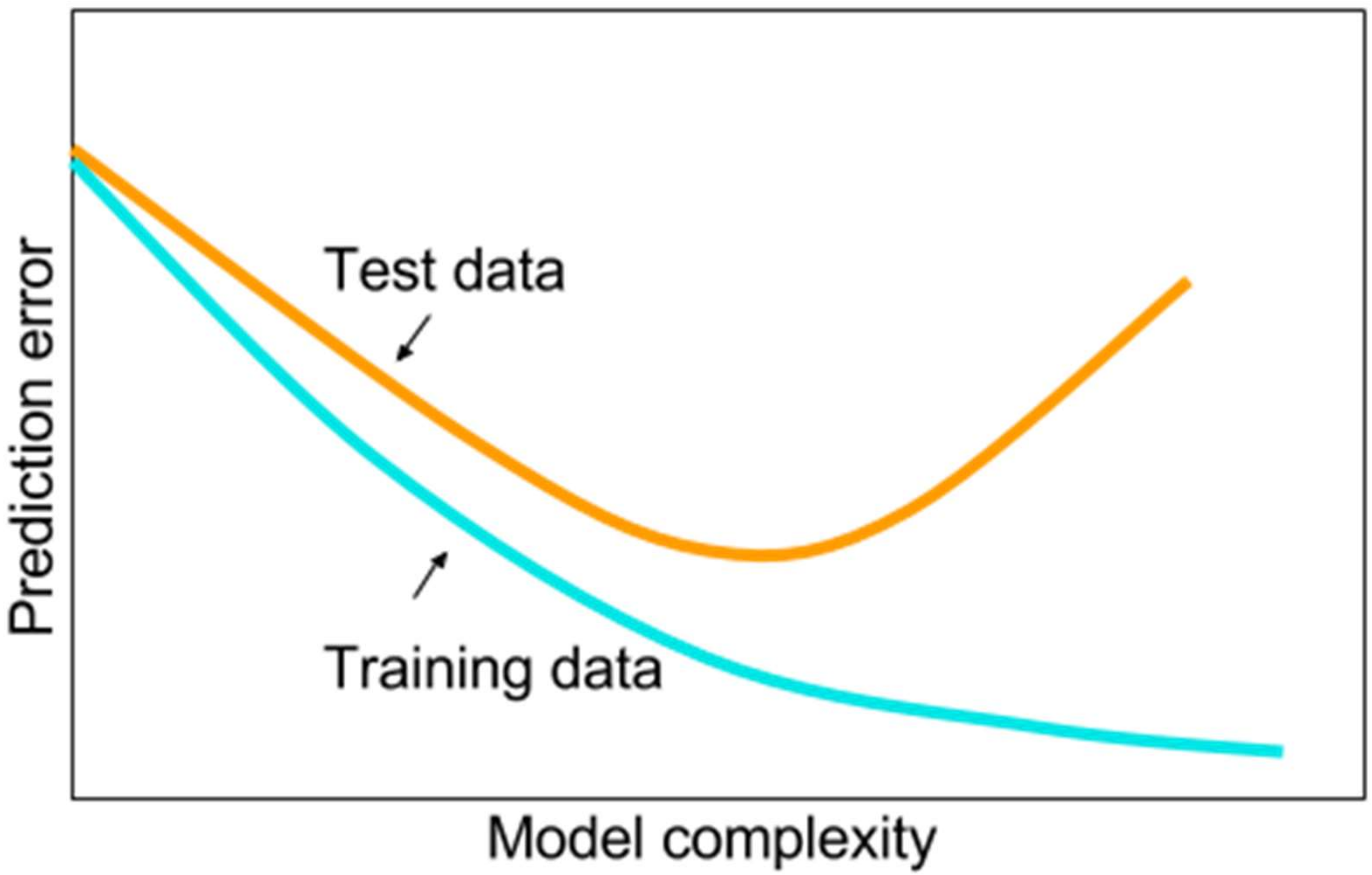
Overfitting (2D)

Feature Space

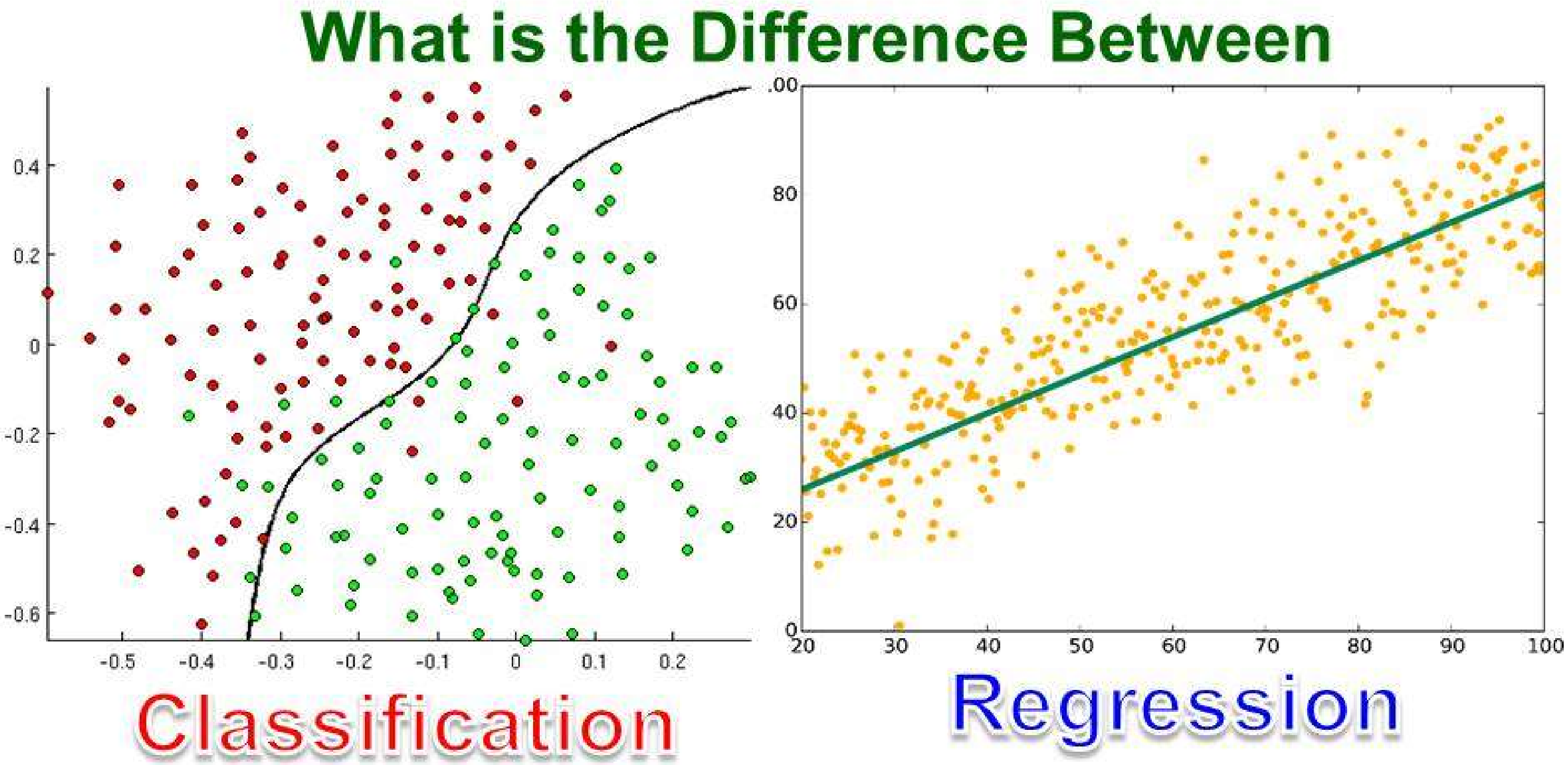


(too simple to explain the variance)

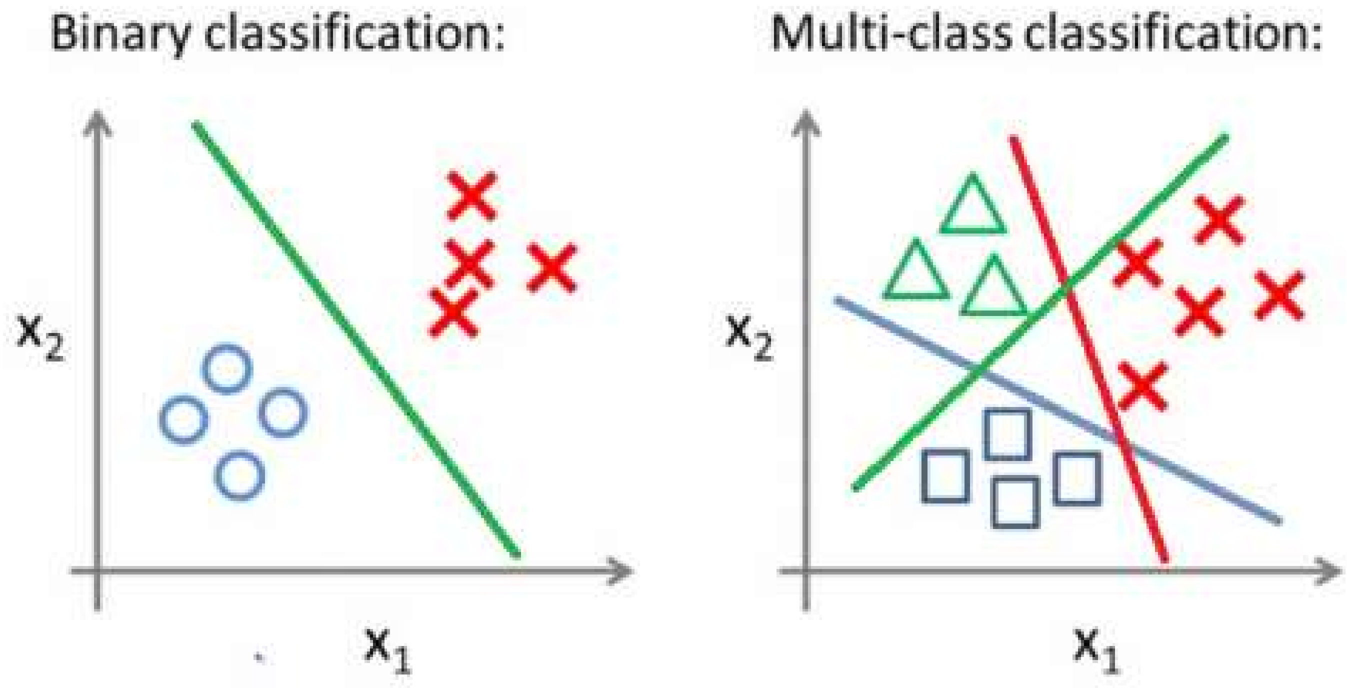
(forcefitting -- too good to be true)


Test data
Training data
Prediction error
Model complexity

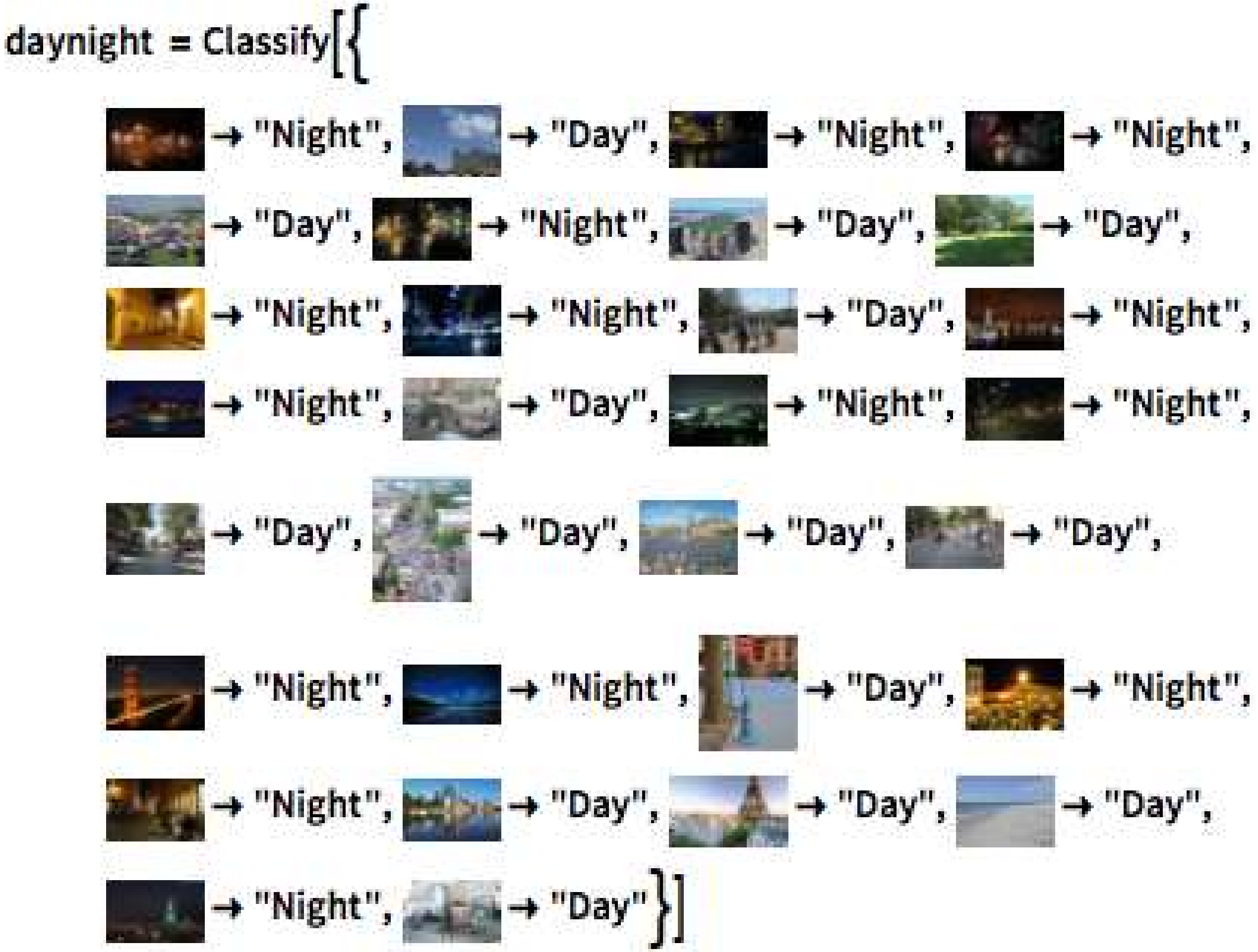
Classification vs. Regression



What is Classification?

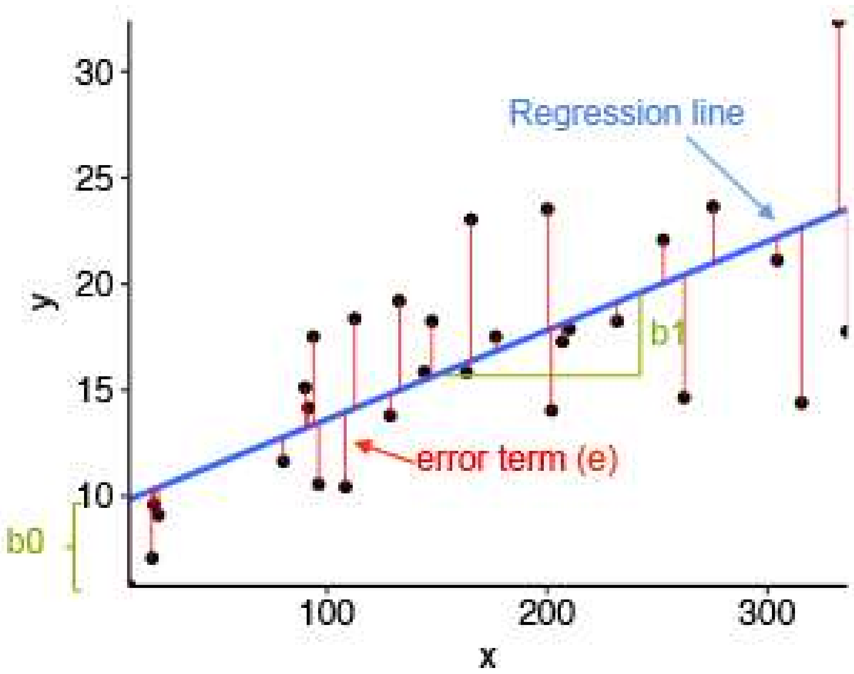


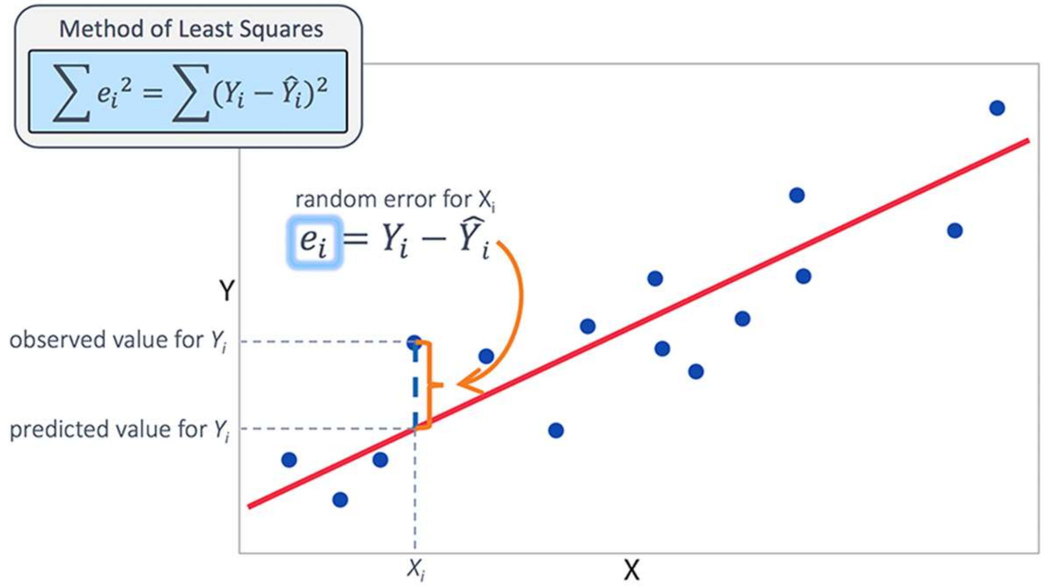
Classification (Ex.)



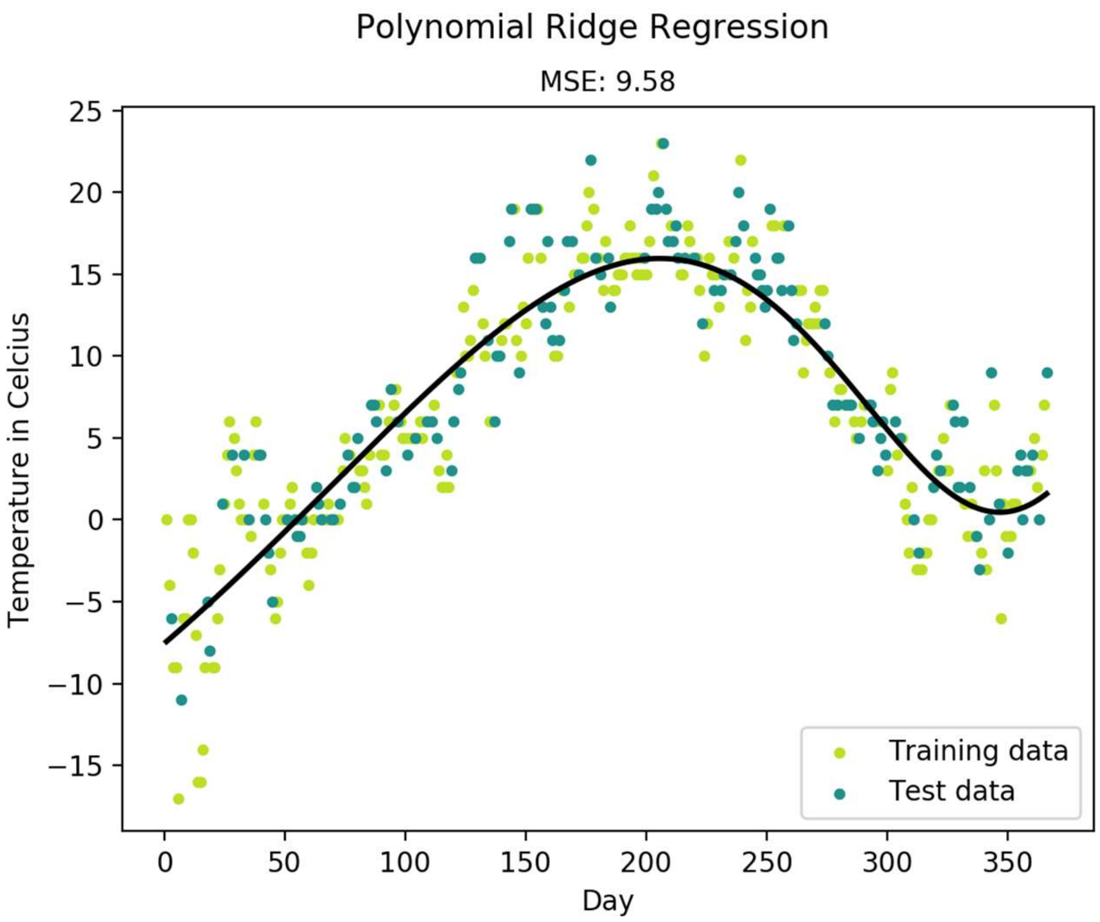
Classification (Ex.)

What is Regression?

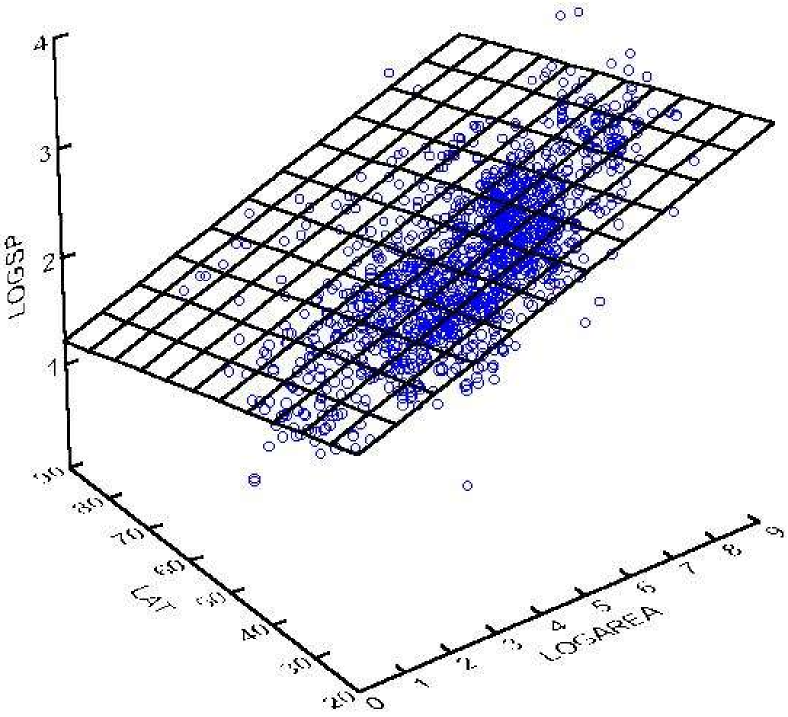


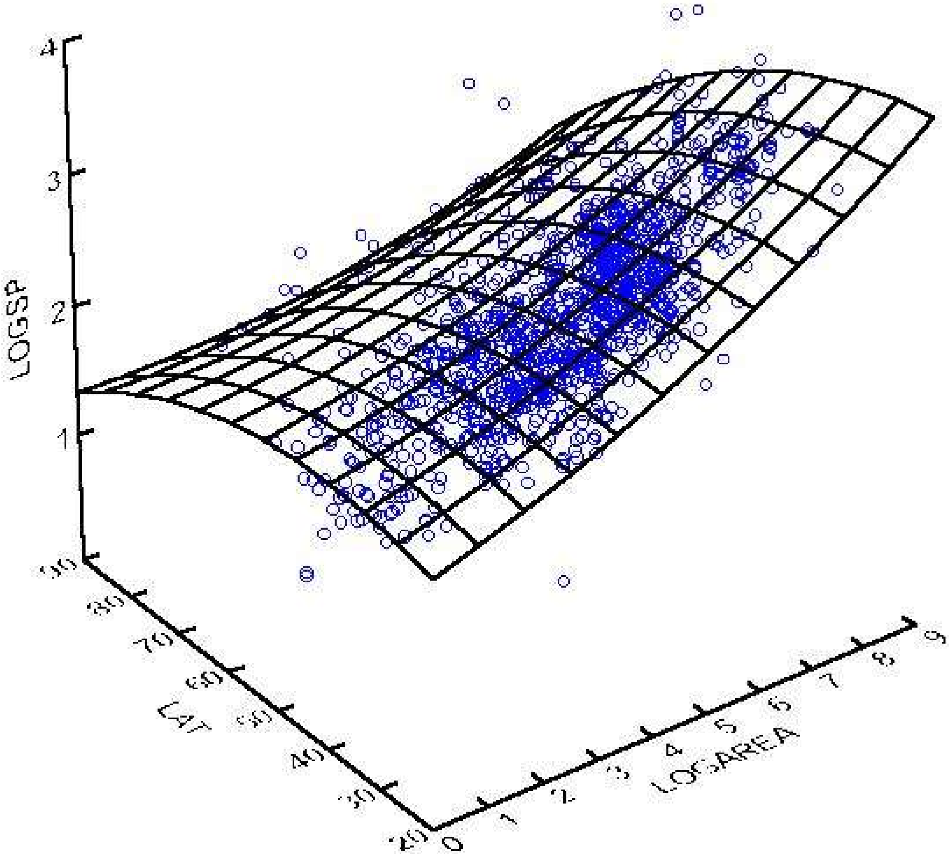


Regression


Polynomial Ridge Regression
MSE: 9.58
Temperature in Celcius
Day
Training data
Test data
25
20
15
10
5
0
−5
−10
−15
0
50
100
150
200
250
300
350

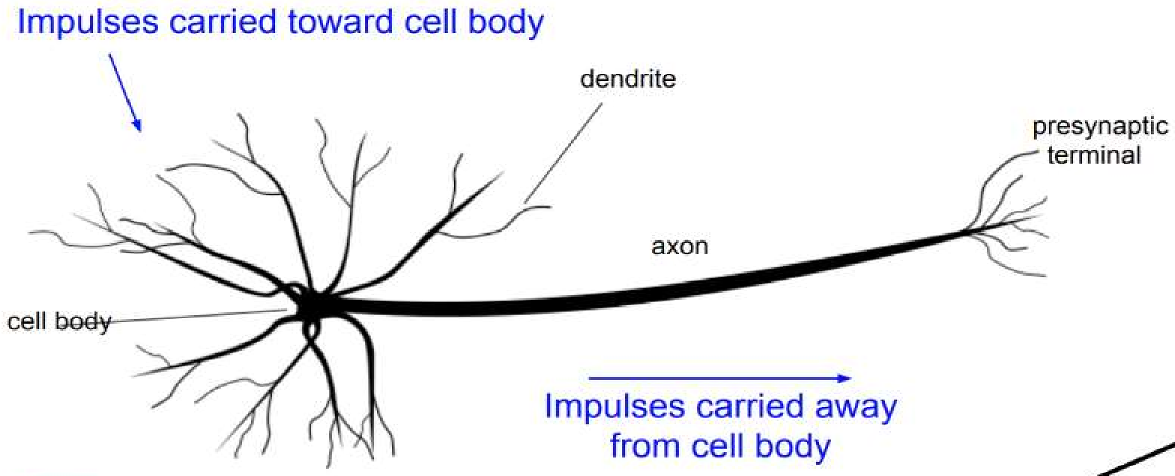
Multi-dimensional Regression



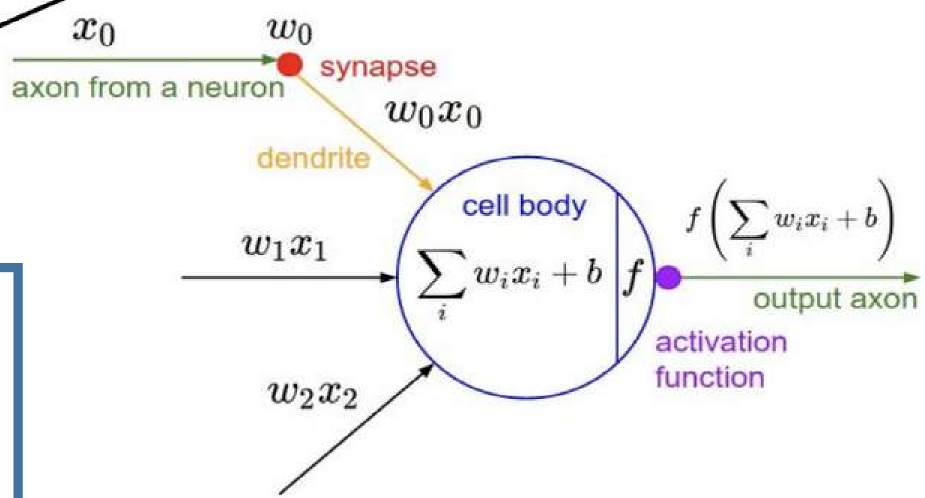


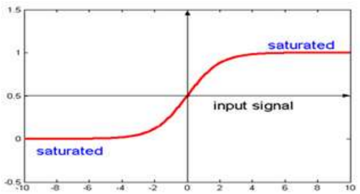
MDL Ch. 2 Handout

Biological Neuron vs. Artificial Neuron



The Keys = Weights !!



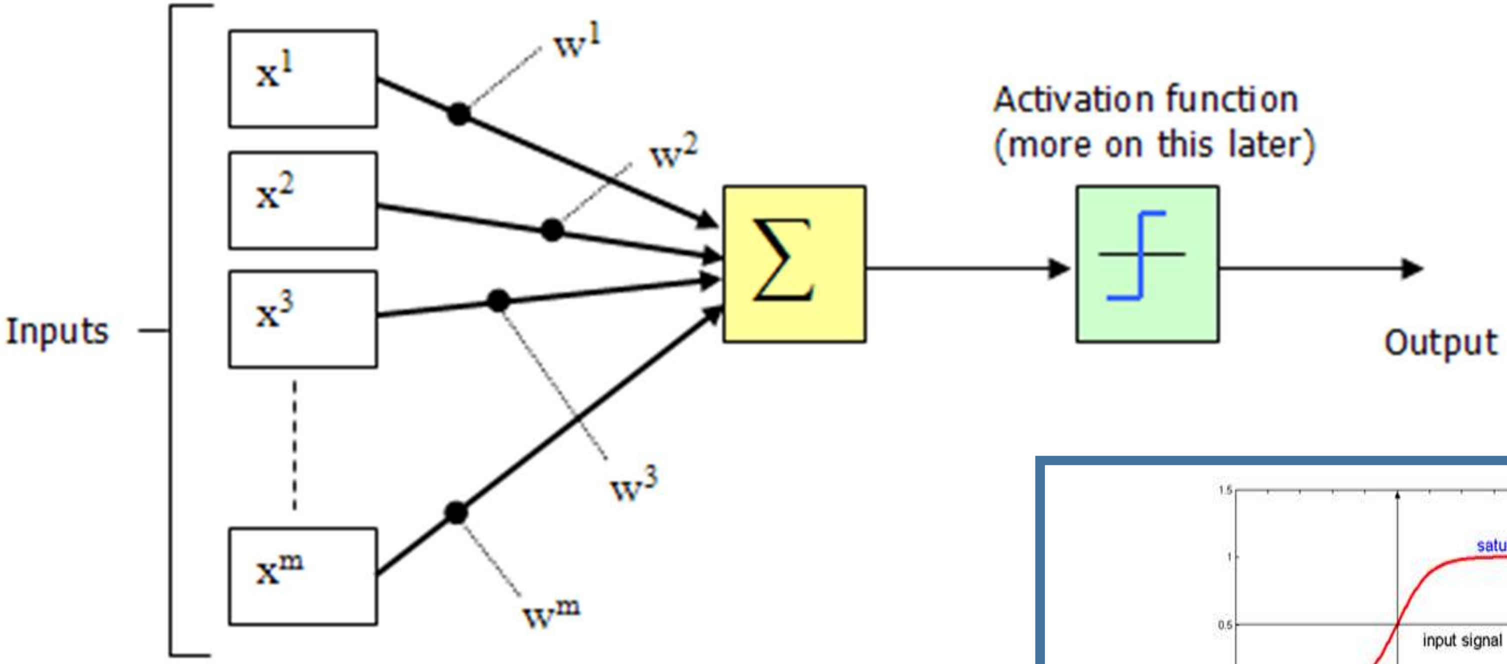


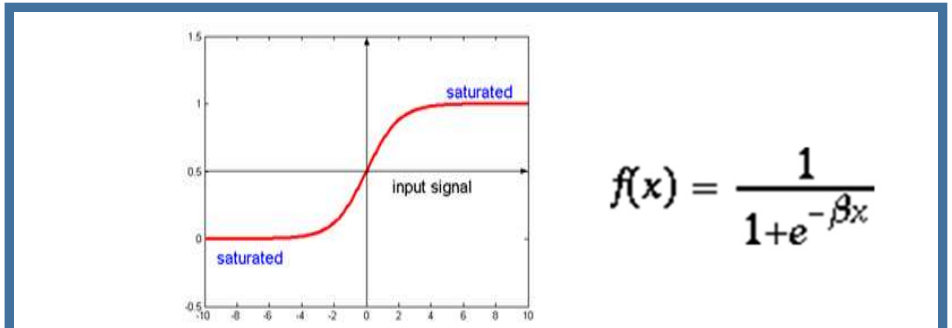
$$f(x) = \frac{1}{1+e^{-\beta x}}$$

Sigmoid Activation Function

Artificial Neural Network (ANN)

The Keys = Weights !!!

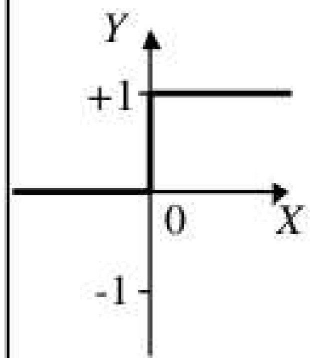
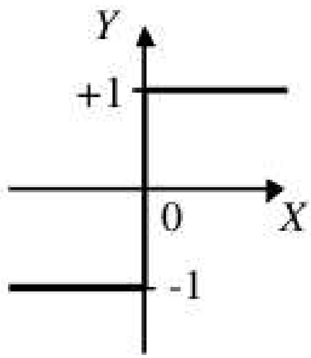
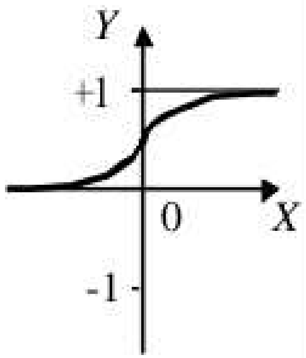
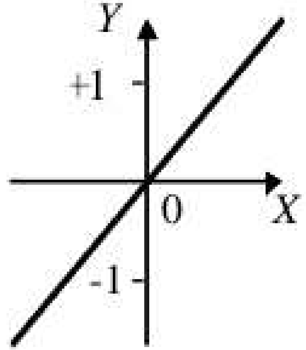




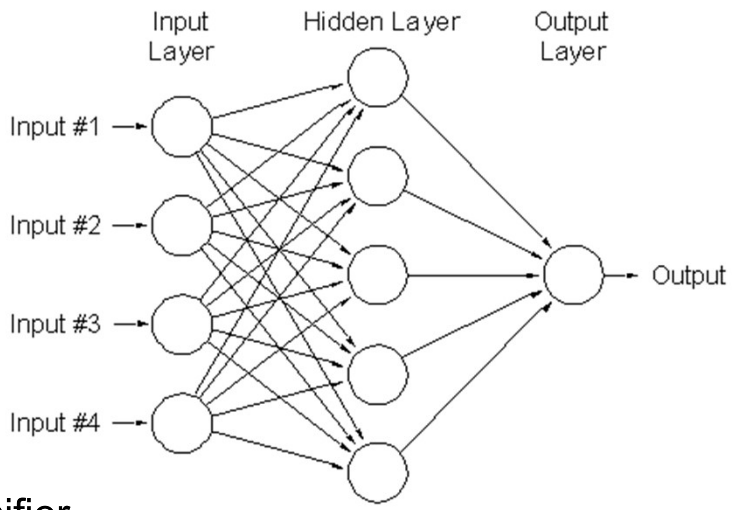
$$f(x) = \frac{1}{1+e^{-\beta x}}$$

Sigmoid Activation Function

Some Activation functions of a neuron

Step function	*Sign function*	*Sigmoid function*	*Linear function*
			
$Y^{step} = \begin{cases} 1, \text{ if } X \geq 0 \\ 0, \text{ if } X < 0 \end{cases}$	$Y^{sign} = \begin{cases} +1, \text{if } X \geq 0 \\ -1, \text{if } X < 0 \end{cases}$	$Y^{sigmoid} = \frac{1}{1+e^{-X}}$	$Y^{linear} = X$

Multi-Layer Neural Networks



- Nonlinear classifier
- **Training:** **find network weights w** to minimize the error between tru e training labels y_i and estimated labels $f_{\mathbf{w}}(\mathbf{x}_i)$:

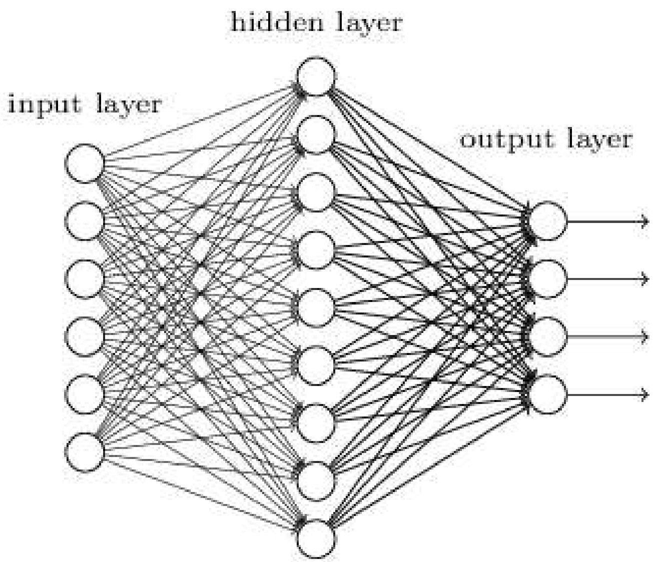
$$E(\mathbf{w}) = \sum_{i=1}^{N} \left(y_i - f_{\mathbf{w}}(\mathbf{x}_i)\right)^2$$

- Minimization can be done by gradient descent provided *f* is differentiable
  - This training method is called **back-propagation**

Shallow Network vs. Deep Network

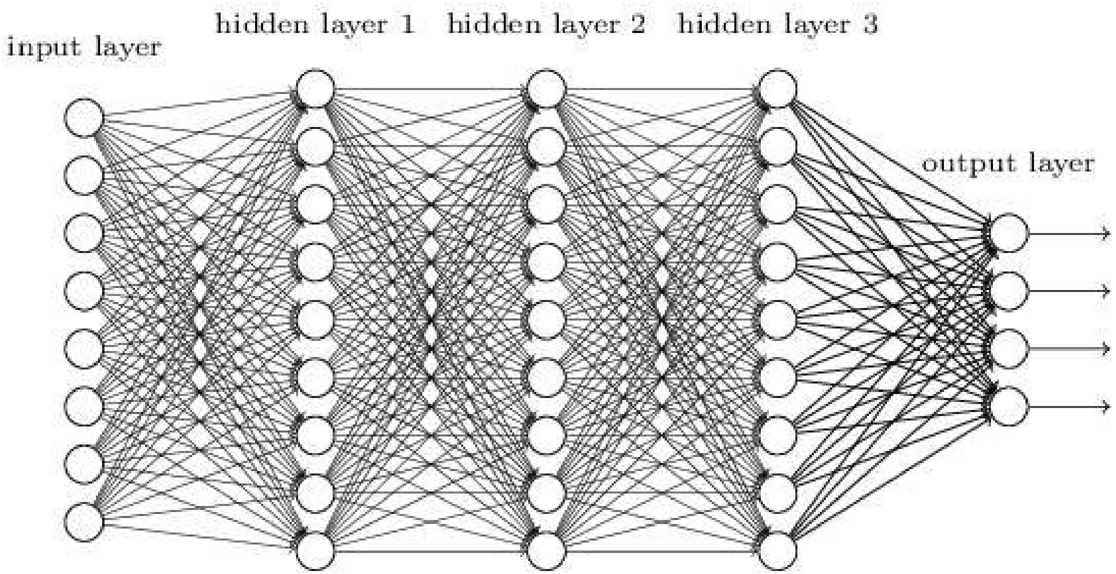
20 Years

"Non-deep" feedforward neural network



# of Hidden Layer <= 1 (i.e., shallow network)

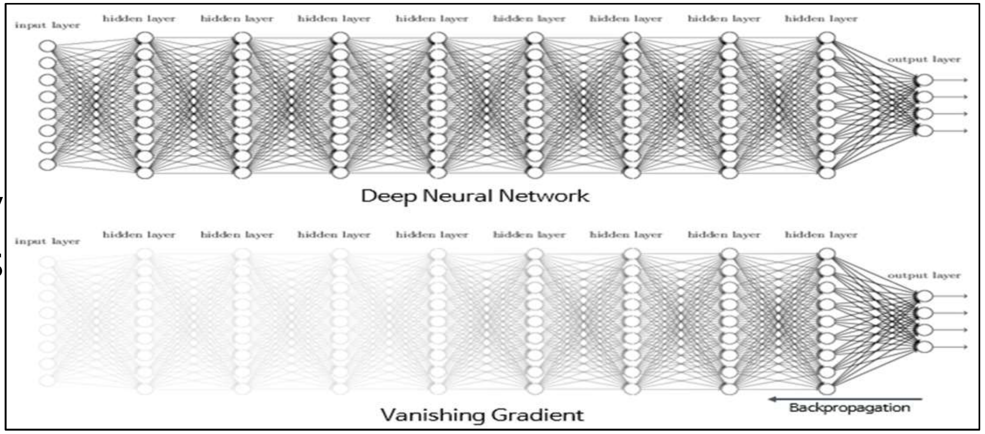
Deep neural network



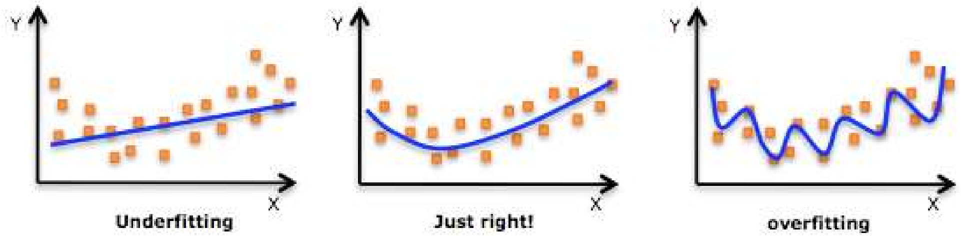
# of Hidden Layer >= 2 (i.e., deep network)

Difficulties of Training Deep Neural Network (or Multi Layer NN)

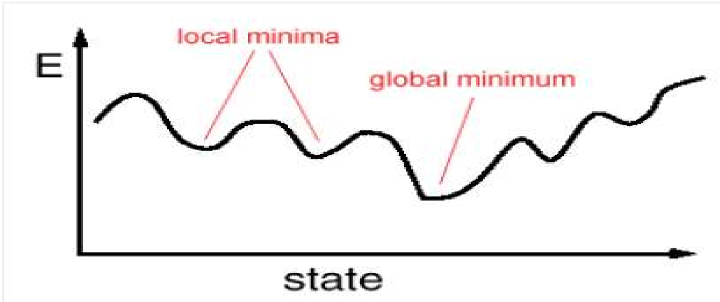
- **Vanishing Gradient in Backpropagation**
  - Problem with nonlinear activation function
  - Gradient (error signal) decreases exponentially with the number of layers and the front layers train very slowly.



- **Over-fitting (No Generalization)**
  - Given limited amounts of labeled data, training via backpropagation does not work well

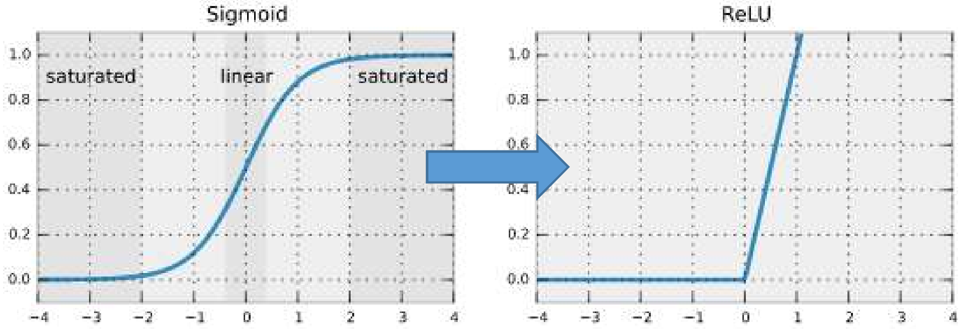


- **Local Minima**
  - Difficulty in optimization

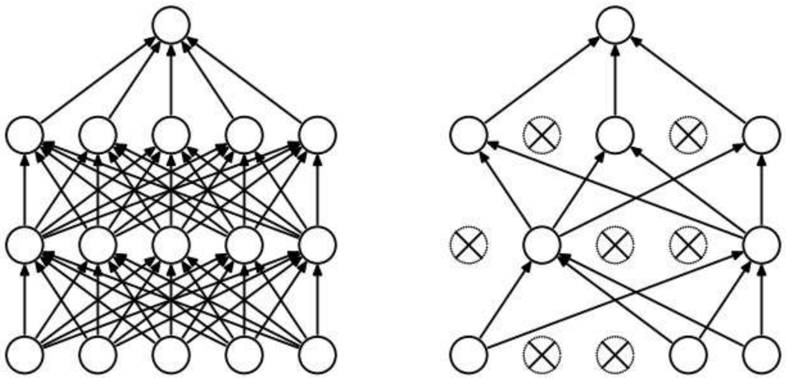


New Solutions for Deep Neural Network

- **Vanishing Gradient**
  - Solved by a new non-linear activation function: Rectified Linear Unit (ReLU) in 2010 & 2011



- **Over-fitting**
  - Solved by new regularization methods: dropout (Hinton et al., 2012) etc.


(a) Standard Neural Net
(b) After applying dropout.

- **Local minima**
  - Solved by high-dimensional non-convex optimization: local minima are all similar
  - Local minima are good and close to global minima

Delta Rule : Perceptron Rule

Perceptron Training Rule

- Weights modified for each example
- Update Rule:

$$w_i \leftarrow w_i + \Delta w_i$$

where

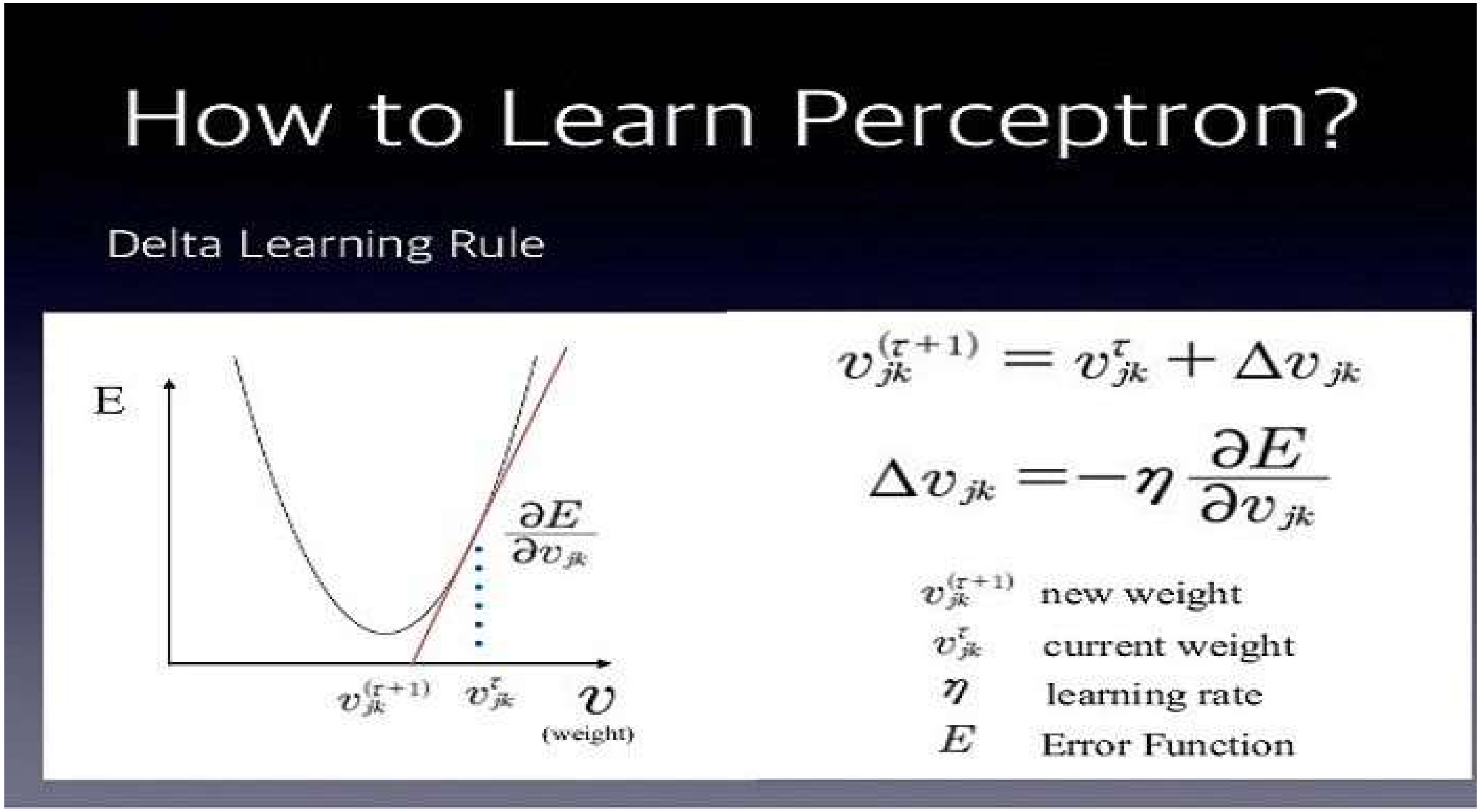
$$\Delta w_i = \eta(t - o)x_i$$

learning rate — target value — perceptron output — input value

In textbook

$$w_{ij} \leftarrow w_{ij} + \alpha e_i x_j \qquad \text{(Equation 2.2)}$$

Gradient Descent in Delta Rule



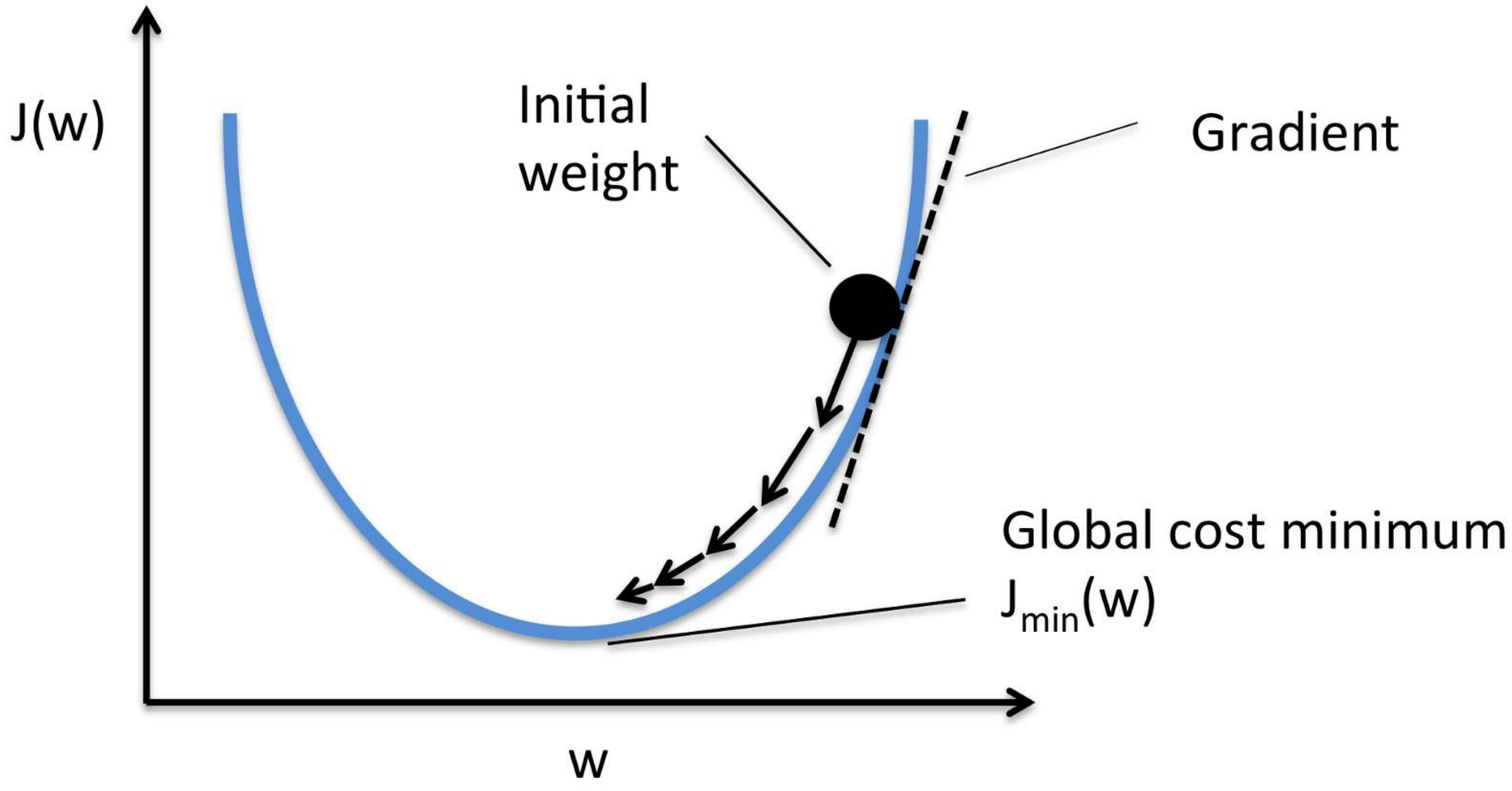
$\nu = \omega$

$$E_p = \frac{1}{2}\sum_n (t_{j_n} - a_{j_n})^2$$

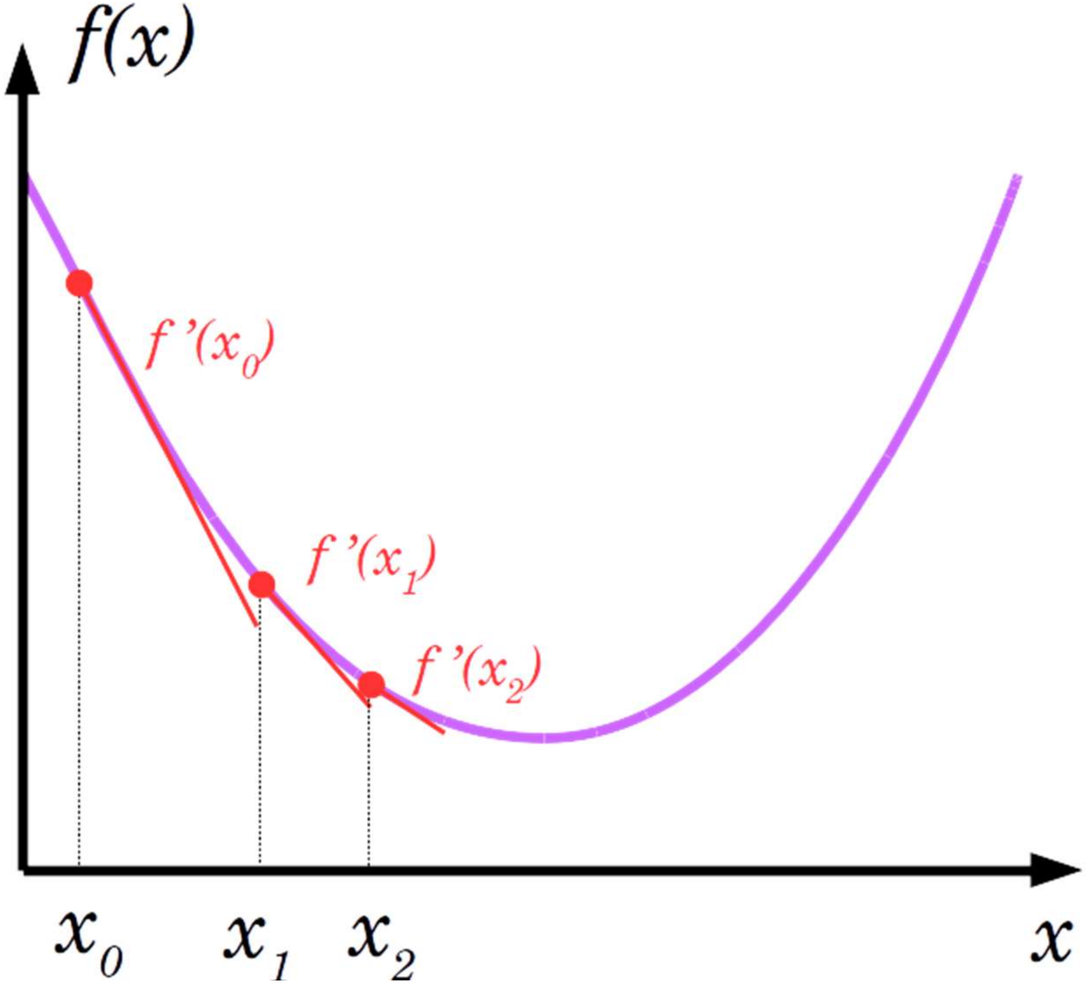
t = target
a = net output = f(ν)

https://medium.com/@neuralnets/delta-learning-rule-gradient-descent-neural-networks-f880c168a804

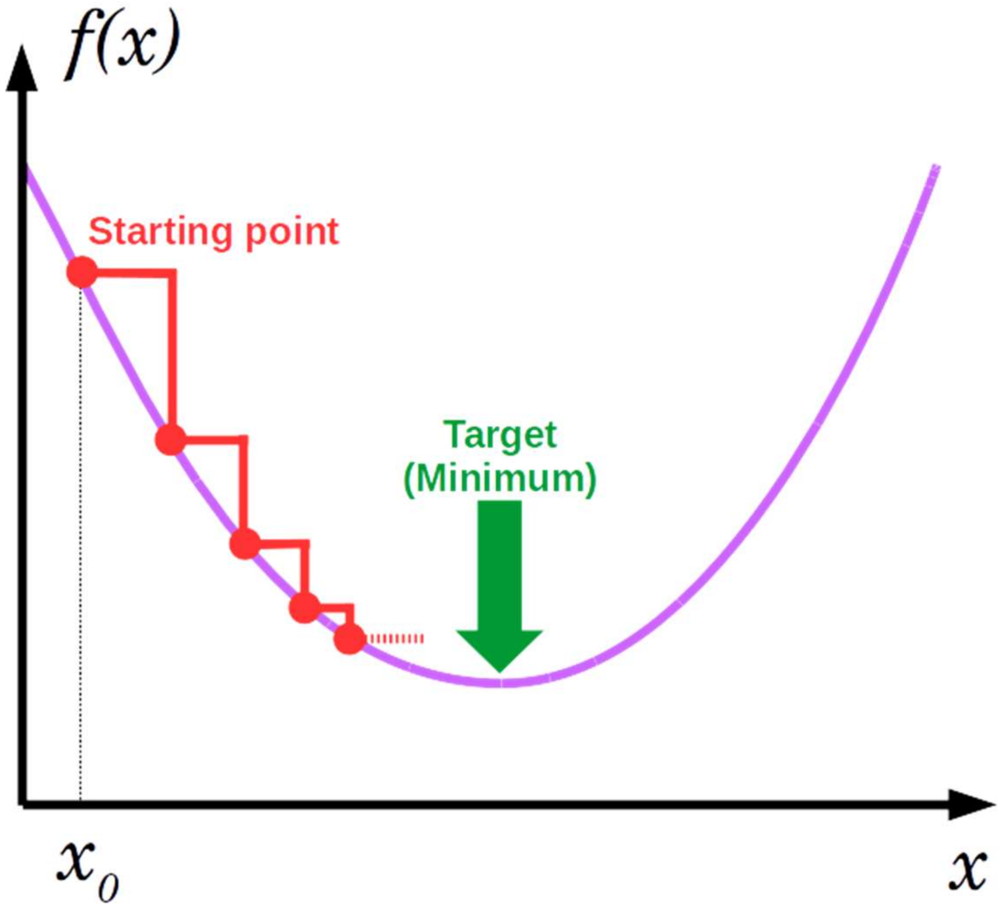
Gradient Descent (1)



Gradient Descent (2)

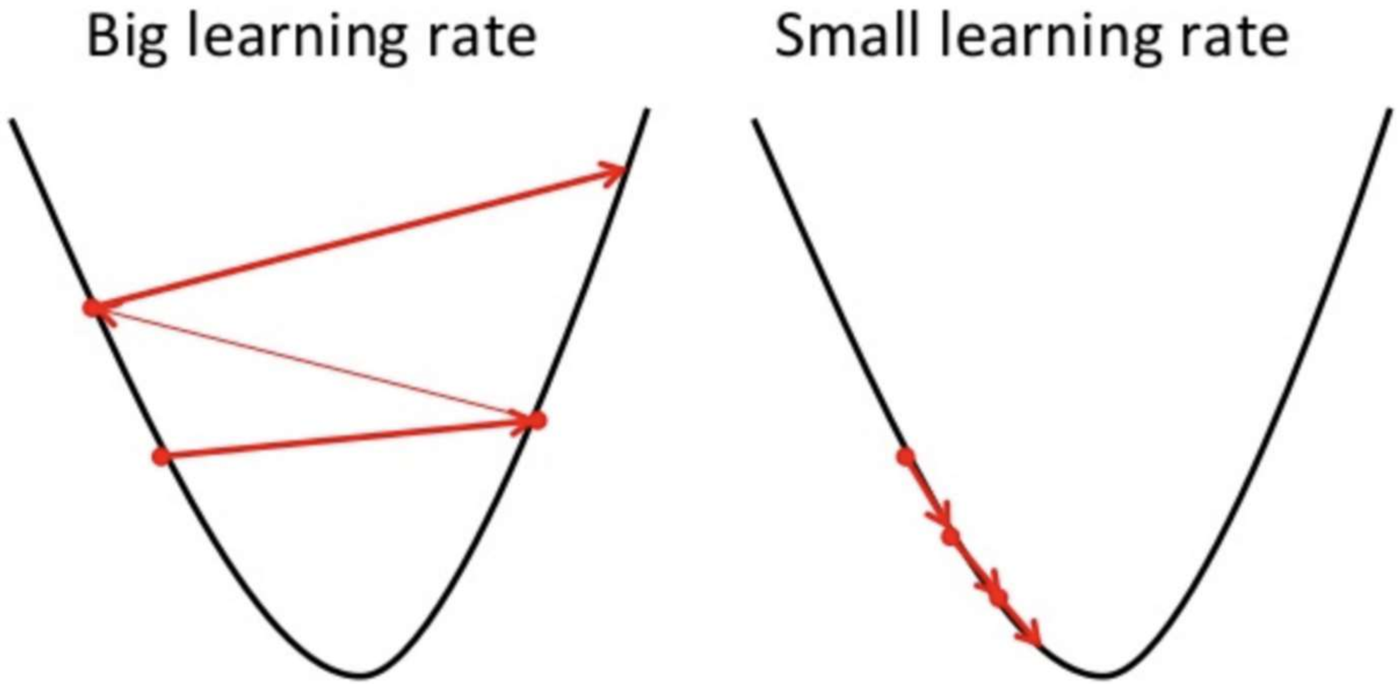


Stepping (Learning Rate) in Gradient Descent

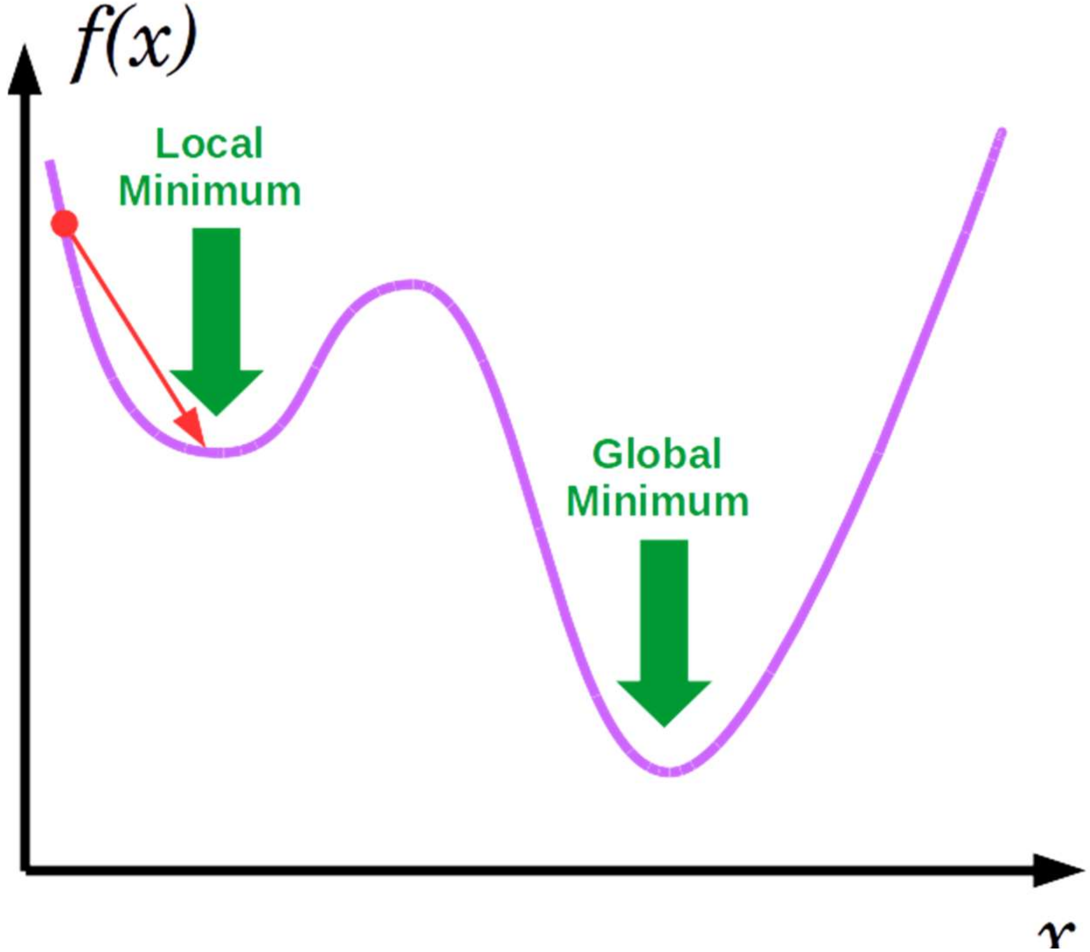


https://www.neural-networks.io/en/single-layer/gradient-descent.php

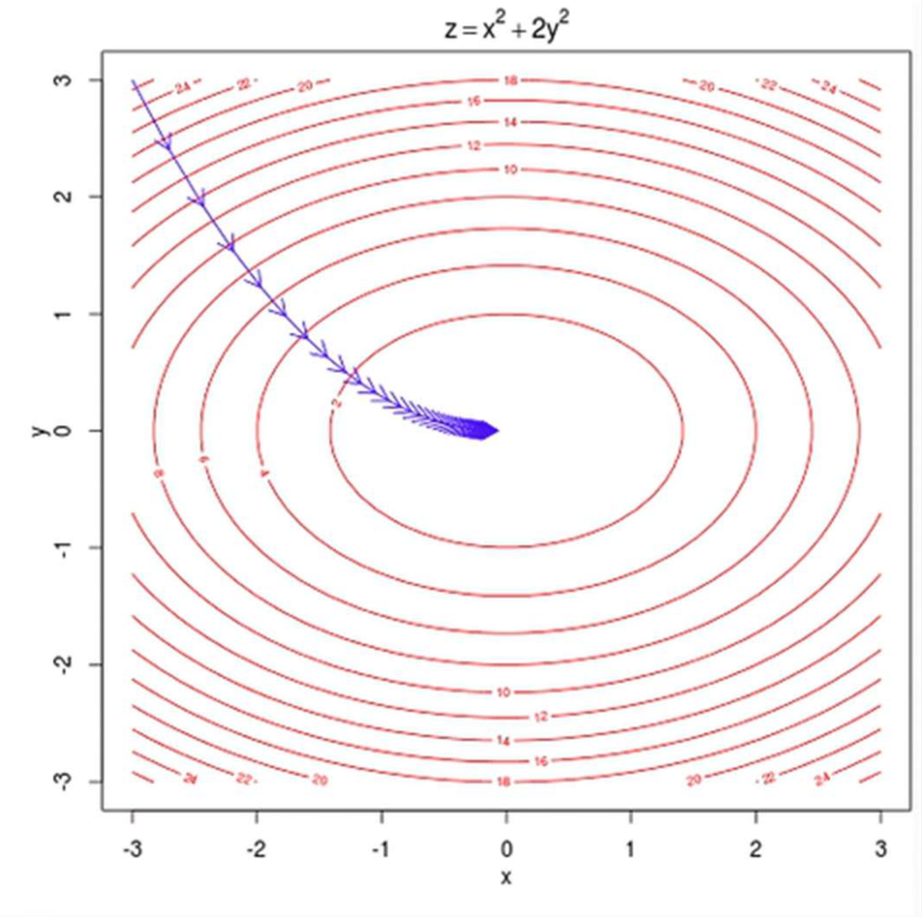
Effect of Learning Rate or Step Size



Effect of Local Minima



Multidimensional Optimization



Generalized Delta Rule

Why activation function then?

- If neural networks had no activation functions, they would fail to learn the complex non-linear patterns that exist in real-world data. Activation functions enable neural networks to learn these non-linear relationships by introducing non-linear behaviors through activation functions.
- Checkout https://www.datacamp.com/tutorial/introduction-to-activation-functions-in-neural-networks

$$w_{ij} \leftarrow w_{ij} + \alpha \delta_i x_j \qquad \text{(Equation 2.3)}$$

$$\delta_i = \varphi'(v_i) e_i \qquad \text{(Equation 2.4)}$$

where

e_i = The error of the output node i

v_i = The weighted sum of the output node i

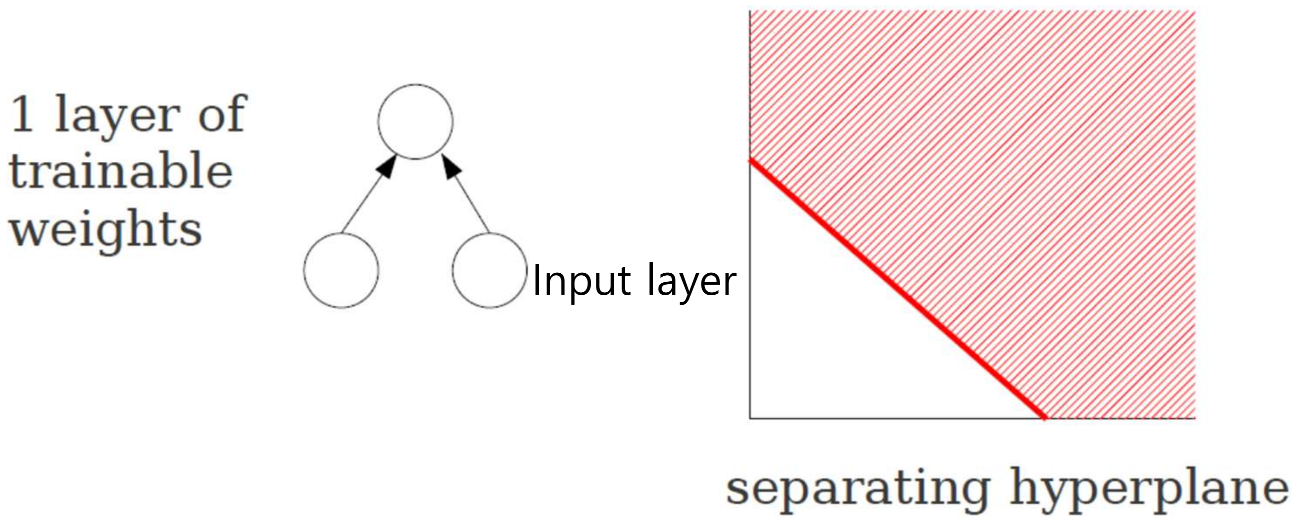
φ' = The derivative of the activation function φ of the output node i

e from the cost function.
If activation fun. is x, then its derivative is 1 (check Equation 2.2)

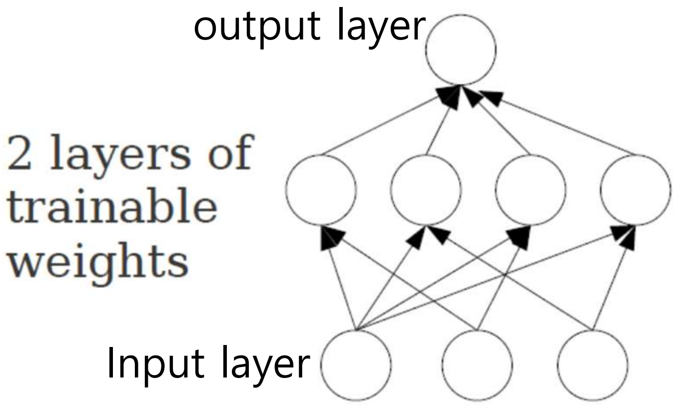
Single vs. Multi Layer Network: XOR Problem

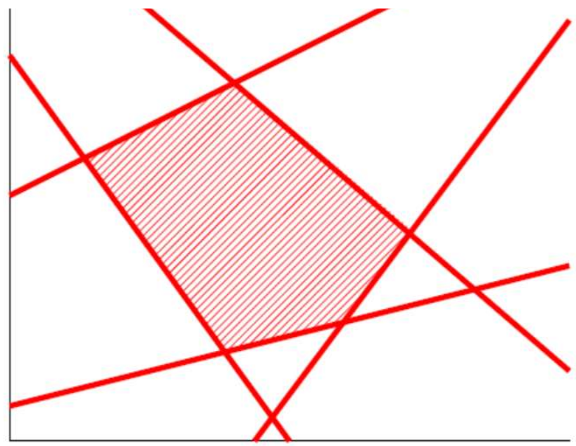
Multi Layer Network:
XOR Problem

Single Layer NN and Decision Boundary



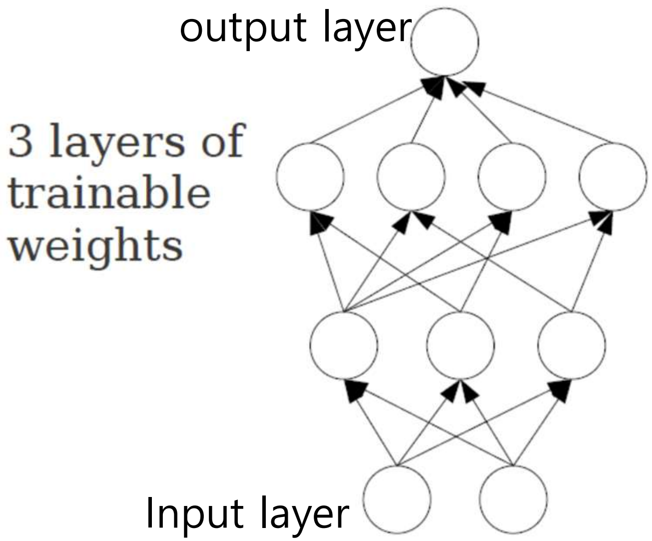
2 Layer NN and Decision Boundary

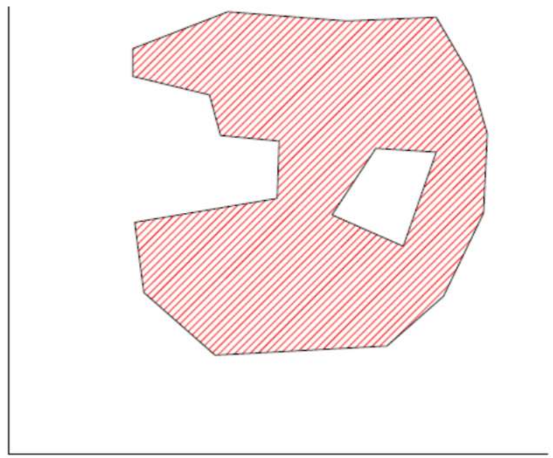




convex polygon region

3 Layer NN and Decision Boundary





composition of polygons:
convex regions

Solving XOR with 2 Layer NN

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

Two solutions:

$$x_1 \overline{x}_2 \vee \overline{x}_1 x_2$$

$$(x_1 \vee x_2) \wedge \overline{x_1 \wedge x_2}$$