Artificial Intelligence Foundations and Applications

Machine Learning

Sudeshna Sarkar

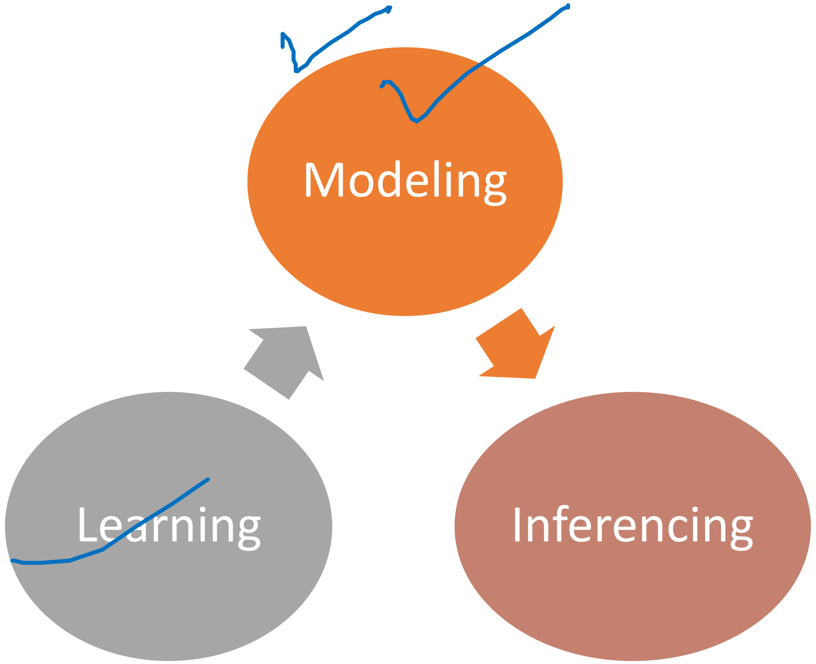
Nov 8 2021

Centre of Excellence in Artificial Intelligence, IIT Kharagpur

Machine Learning : Definition

Learning is the ability to evolve behaviours based on data (experience).

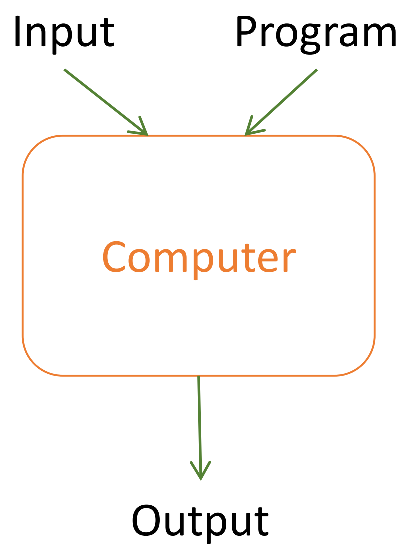
- Learn from data such as build a model from data
- Use the model for prediction, decision making or solving some tasks



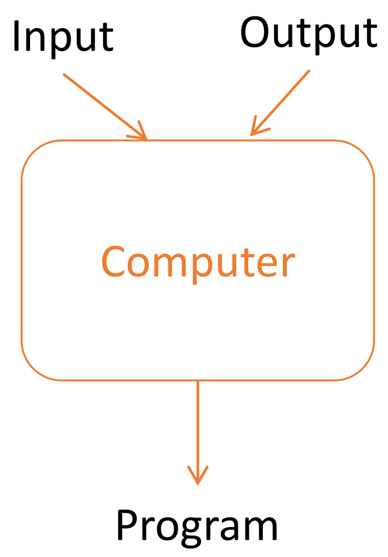
The Machine Learning Solution

- Collect many examples that specify the correct output for a given input
- ML to get the mapping from input to output

Algorithmic solution



Machine learning solution



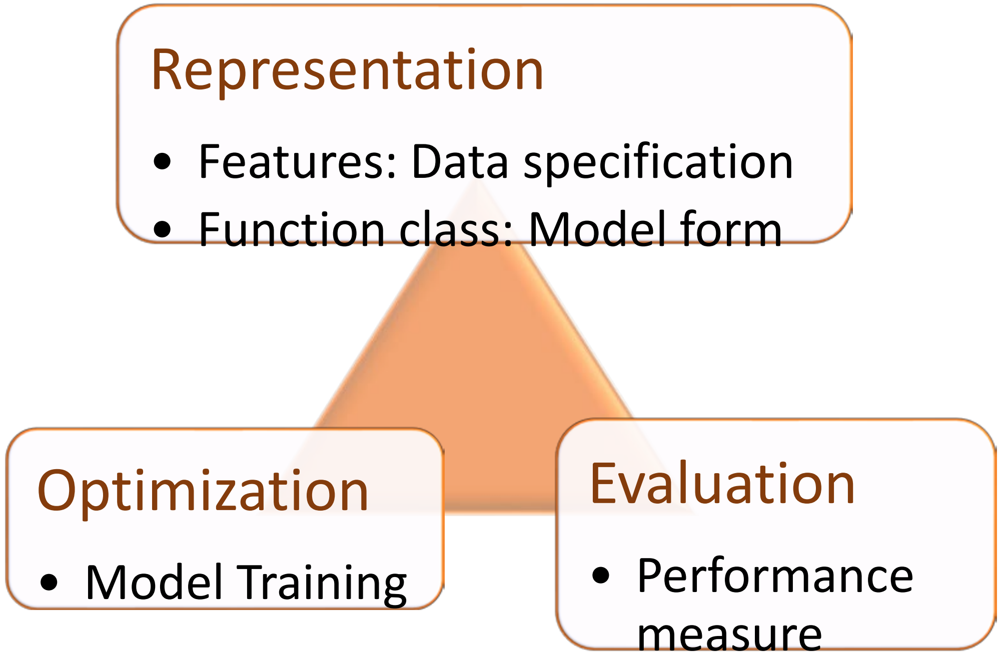
Components of a learning problem

- **Task:** The behaviour or task being improved.
  - For example: classification, acting in an environment
- **Data:** The experiences that are being used to improve performance in the task.
- **Measure of improvement :**
  - For example: increasing accuracy in prediction, improved speed and efficiency

Models for Prediction

Design a Learner

1. Choose the training experience
   - Features
2. Choose the target function f
   - Choose how to represent the target function
3. Choose a learning algorithm to infer the target function



Representation of Data

1. How is the data specified?

   A. Features

   - Feature vector of n features

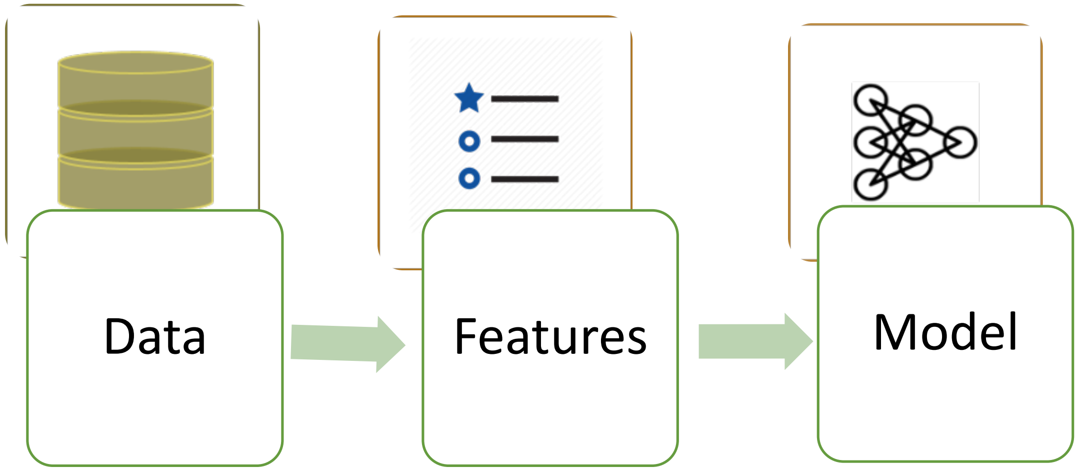
$$\bar{x} = (x_1, x_2, \dots, x_n)$$

   B. Convert input to a vector of basis functions

$$\left(\phi_0(\bar{x}), \phi_1(\bar{x}), \dots, \phi_p(\bar{x})\right)$$

Feature Choice

- Input Data comprise features
  - Structured features (numerical or categorical values)
  - Unstructured (text, speech, image, video, etc)
- Use only relevant features
- Too many features?
  - Select feature subset (reduction)
  - Extract features: Transform features



1. Model Representation

- The richer the representation, the more useful it is for subsequent problem solving.
- The richer the representation, the more difficult it is to learn.

- Linear function
- Decision Tree
- Graphical Model
- Neural Network

$$y = f(\bar{x})$$

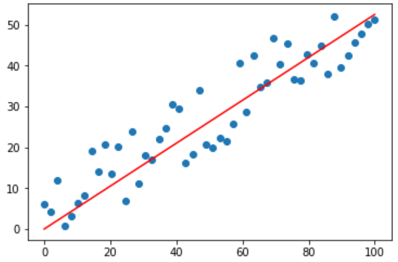
$$y = g(\bar{\phi}(\bar{x}))$$

Model Representation

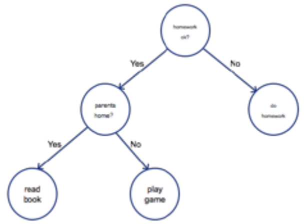
Hypothesis space

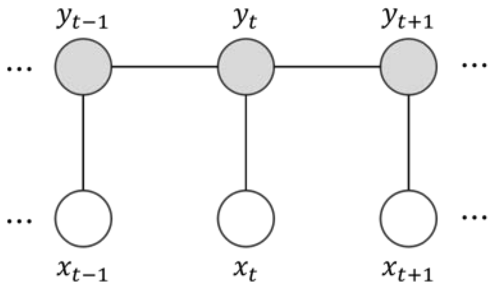
$$y = f(\bar{x})$$

Linear Function	Decision Tree	Graphical Model	Neural Net



$Y = w_0 + w_1 x_1 + \cdots + w_p x_p + \epsilon$





2. Evaluation

1. Accuracy $= \dfrac{\text{\# correctly classified}}{\text{\# all test examples}}$
2. Logarithmic Loss:

$$L_i = -\log(P(Y = y_i | X = x_i))$$

$$L = \sum_{c=1}^{M} y_{oc} \log(p_{oc})$$

3. Mean Squared error

$$MSE = \frac{1}{m} \sum \left(y_{pred} - y_{true}\right)^2$$

3. Optimization

1. Define loss function
2. Optimize loss function

- Stochastic Gradient Descent (Convex functions)
- Combinatorial optimization
  - E.g.: Greedy search
- Constrained optimization
  - E.g.: Linear programming

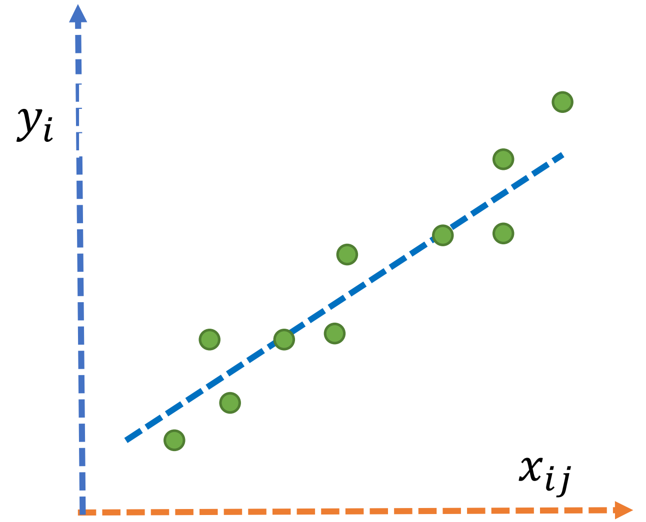
Elements of Optimization

1. Variables
2. Constraints
3. Objective Function

Simple Linear Regression

1. Variables: $w_0, w_1, \dots, w_n$
2. Constraints: none
3. Objective Function: Minimize

$$\sum_{i=1}^{m} \left(y_i\left(w_0 + \sum_{j=1}^{n} w_j x_{ij}\right)\right)^2$$



- m data points, n features
  - x_{ij}: jth attribute of ith instance
  - y_i: output of ith instance

Find coefficients $w_0, w_1, \dots, w_n$ to best fit data

Broad types of machine learning

- **Supervised Learning**
  - Training Data with labels: X,y (pre-classified)
  - Given an observation x, what is the best label for y?
- **Unsupervised learning**
  - Training Data without labels: X
  - Given a set of x's, find hidden structure
- Semi-supervised Learning
  - Training Data + some Labels
- Reinforcement Learning
  - Given: observations and periodic rewards as the agent takes sequential action in an environment
  - Determine optimum policy

Supervised Learning

Given data containing the inputs and outputs:

Training Data:

$\{(\overline{x_1}, y_1), (\overline{x_2}, y_2), \dots, (\overline{x_m}, y_m)\}$

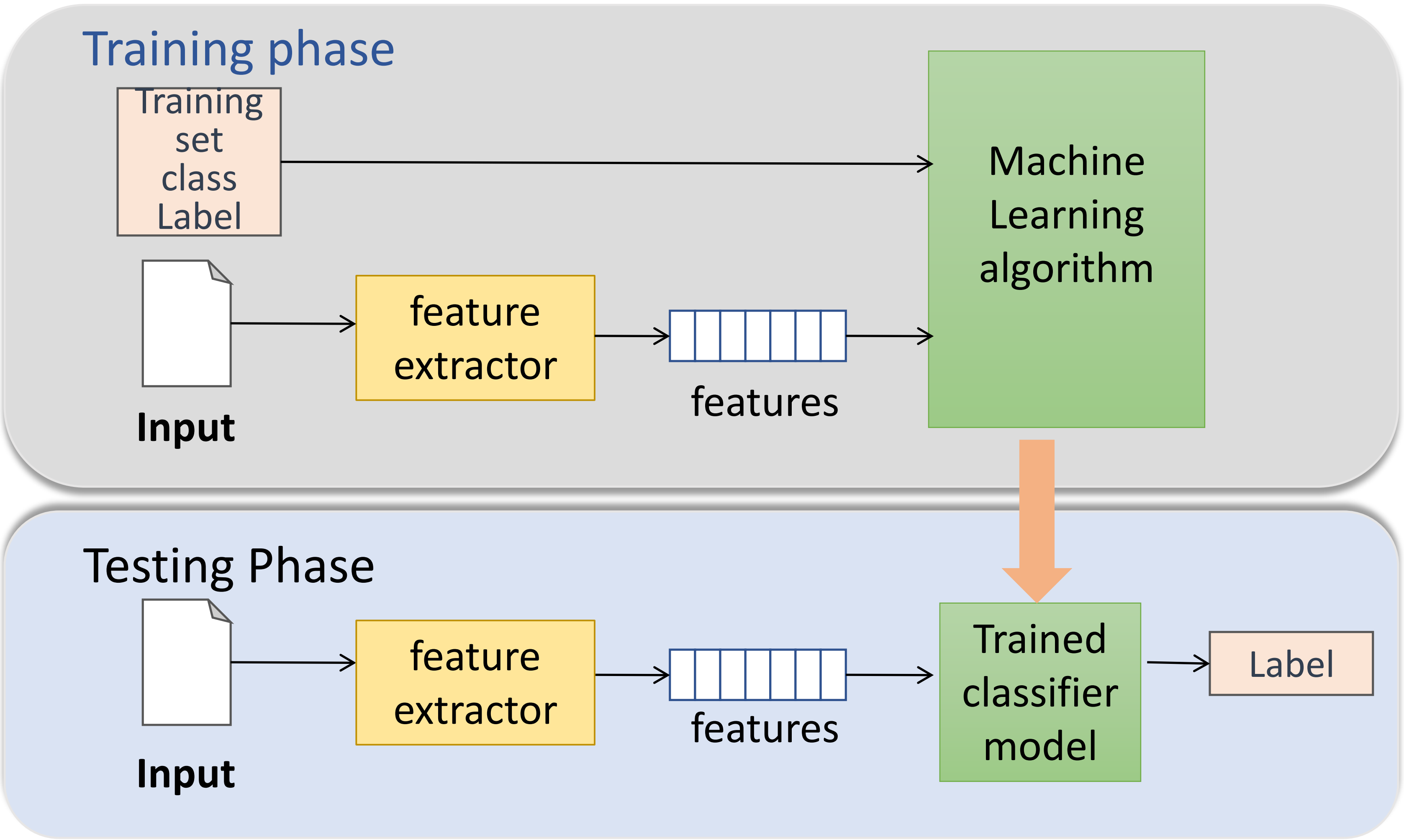
- Learn a function *f*(*x*) to predict *y* given *x*

$\bar{X}$	Y
$\overline{x_1}$	y_1
$\overline{x_2}$	y_2
…	..
$\overline{x_m}$	y_m

Model → $f_\theta(\bar{x}) \rightarrow y$

Parameters

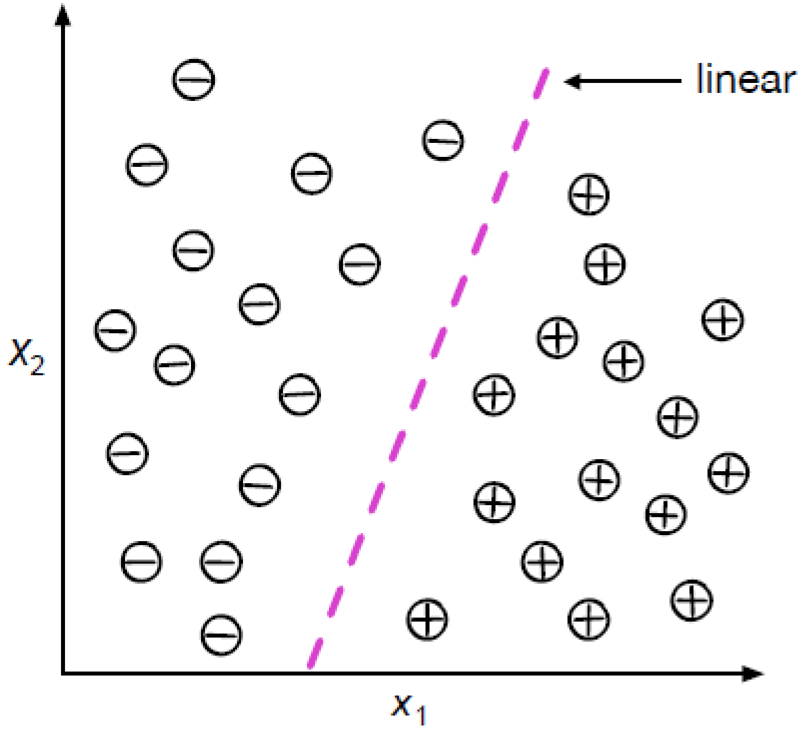
- Training: Learn the model from the Training Data
- Given Test instance $\overline{x'}$, predict $y' = f_\theta(\overline{x'})$


Training phase
Training set class Label
Machine Learning algorithm
feature extractor
features
Input
Testing Phase
feature extractor
features
Trained classifier model
Label
Input

Supervised Learning

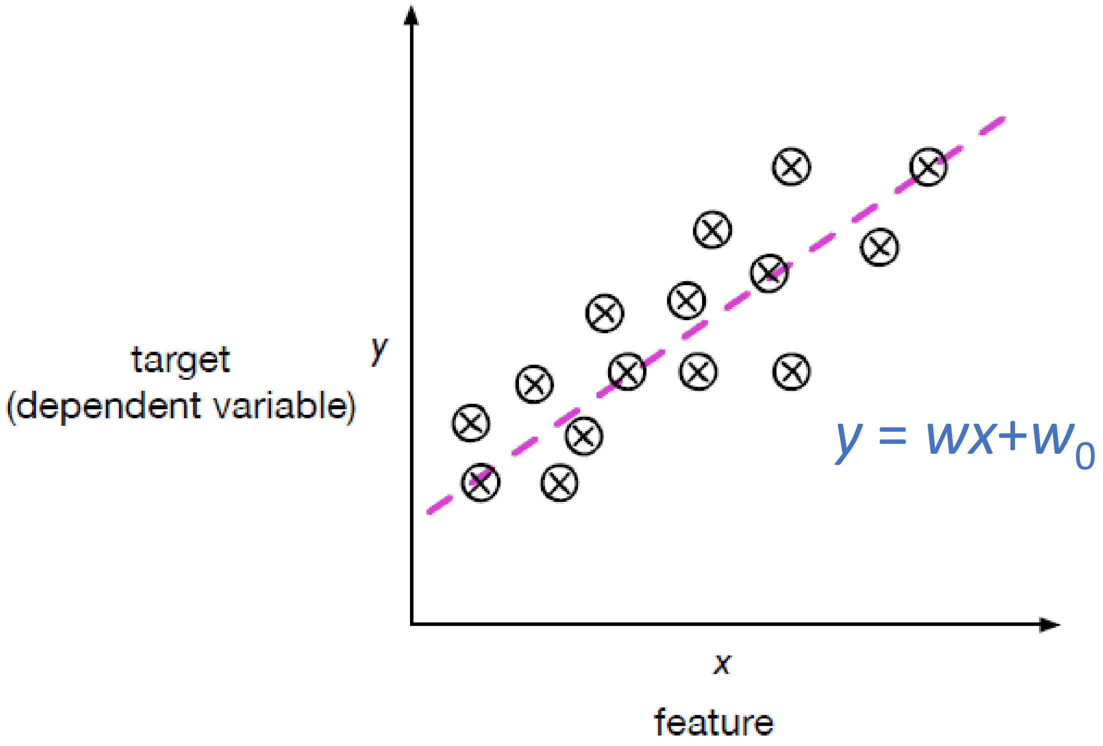
Classification

Y is categorical/ discrete



Regression

Y is numeric / continuous



Example Tasks

Classification

- Object identification from images
- Defect classification
- Credit card transaction fraud or not

Regression

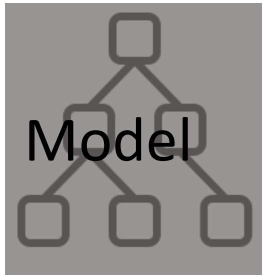
- House price prediction
- Remaining Useful Life Prediction
- Probability of developing cracks
- Demand forecasting

Structured Prediction

- Machine translation: English sentence → Japanese sentence
- Dialogue: conversational history → next utterance
- Image captioning: image → sentence describing image
- Image segmentation: image → segmentation

Supervised Learning
Classification Example

Training
Samples



Train a model to minimize loss

Test
Instances

Probabilistic Classification

X1	X2	X3	X4	Category
				Type 1
				Type 3
				Type 1
				Type 2

Predict a probability distribution over the set of classes **Pr (Y|X)**

X1	X2	X3	X4	Type 1	Type 2	Type 3
				0.4	0.15	0.45
				0.2	0.7	0.1
				0.1	0.2	0.7
				0.5	0.1	0.4

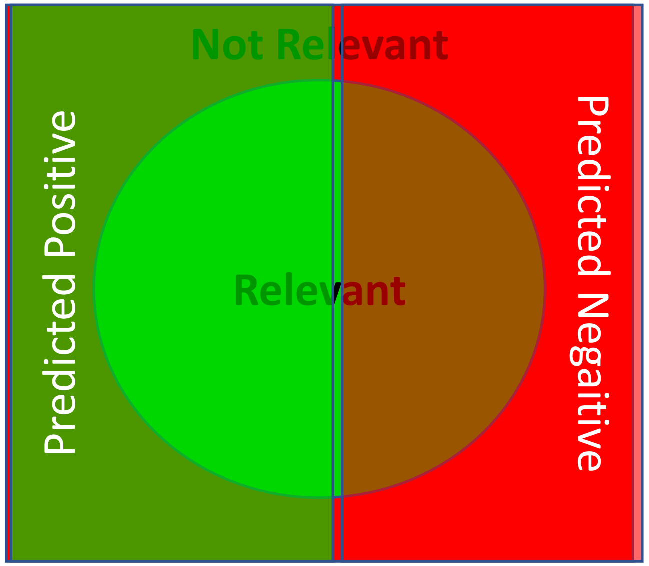
Evaluation for Classification problems

$$\textbf{Accuracy} = \frac{\#\text{ correctly classified}}{\#\text{ all test examples}}$$

$$= \frac{\#\text{predicted true } pos + \#\text{predicted true } neg}{\#\text{all test examples}}$$

$$\textbf{Precision} = \frac{\#\text{ predicted true } pos}{\#\text{ predicted pos}}$$

$$\textbf{Recall} = \frac{\#\text{ predicted true } pos}{\#\text{ True } pos}$$



		True Class		
		Pos	Neg	
Predicted Class	Pos	TP	FP	
	Neg	FN	TN	

Loss Function Classification problems

Loss indicates how bad the model's prediction is.

1. Fraction of Misclassifications

$$Error = \sum_{i=1}^{m} \frac{I(y_i \neq \hat{y_i})}{m}$$

2. Logarithmic Loss: Maximize the log likelihood. For a loss function, minimize the negative log likelihood of the correct class:

$$L_i = -\log(P(Y = y_i | X = x_i))$$

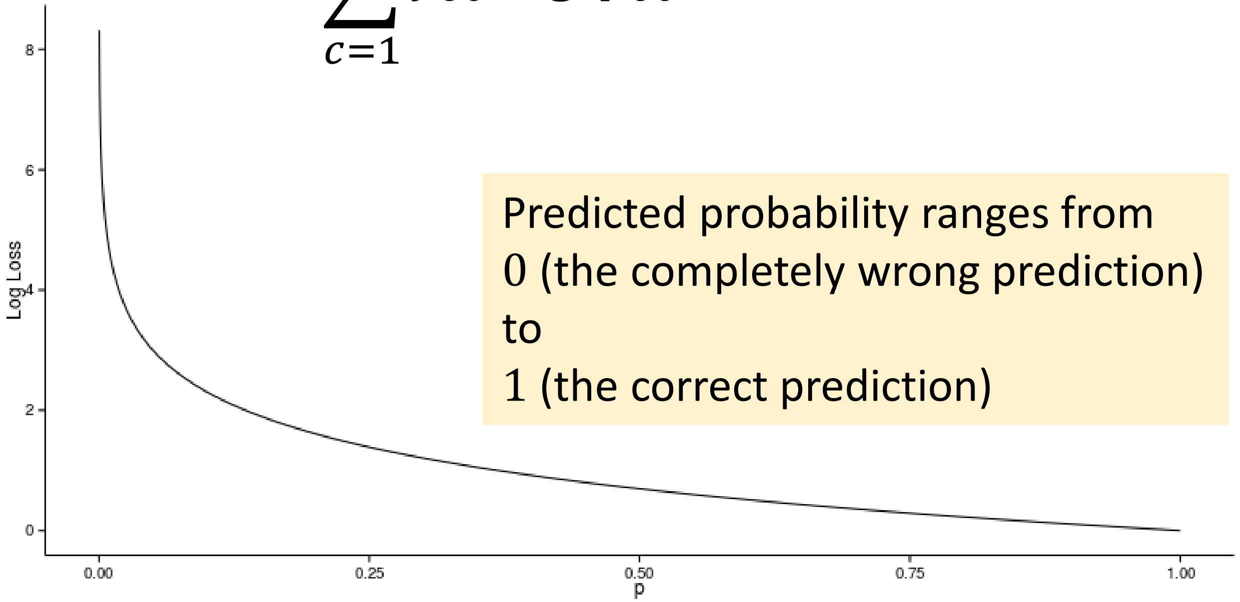
Logarithmic Loss Function

Logarithmic Loss:

$$L_i = -\log(P(Y = y_i | X = x_i))$$

$$L = \sum_{c=1}^{M} y_{oc} \log(p_{oc})$$

M - number of classes
y - binary indicator (0 or 1) if class label ***c*** is the correct classification for observation ***o***
p - predicted probability observation ***o*** is of class ***c***



Predicted probability ranges from
0 (the completely wrong prediction)
to
1 (the correct prediction)

2. Evaluation for regression problem

Mean Squared error

$$MSE = \frac{1}{m}\sum_{i=1}^{m}(\widehat{y_i} - y_i)^2$$

Predicted value

True value

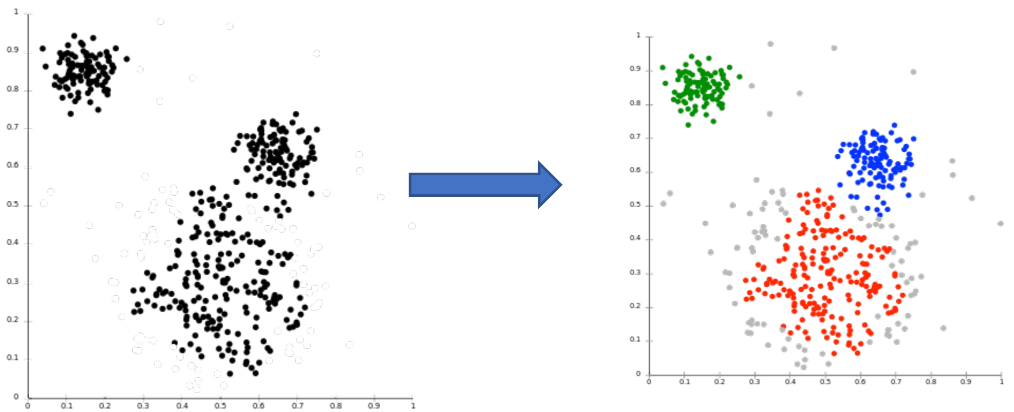
Unsupervised Learning (Clustering)

Given $\{\overline{x_1}, \overline{x_2}, \ldots \overline{x_m}, \}$ without labels

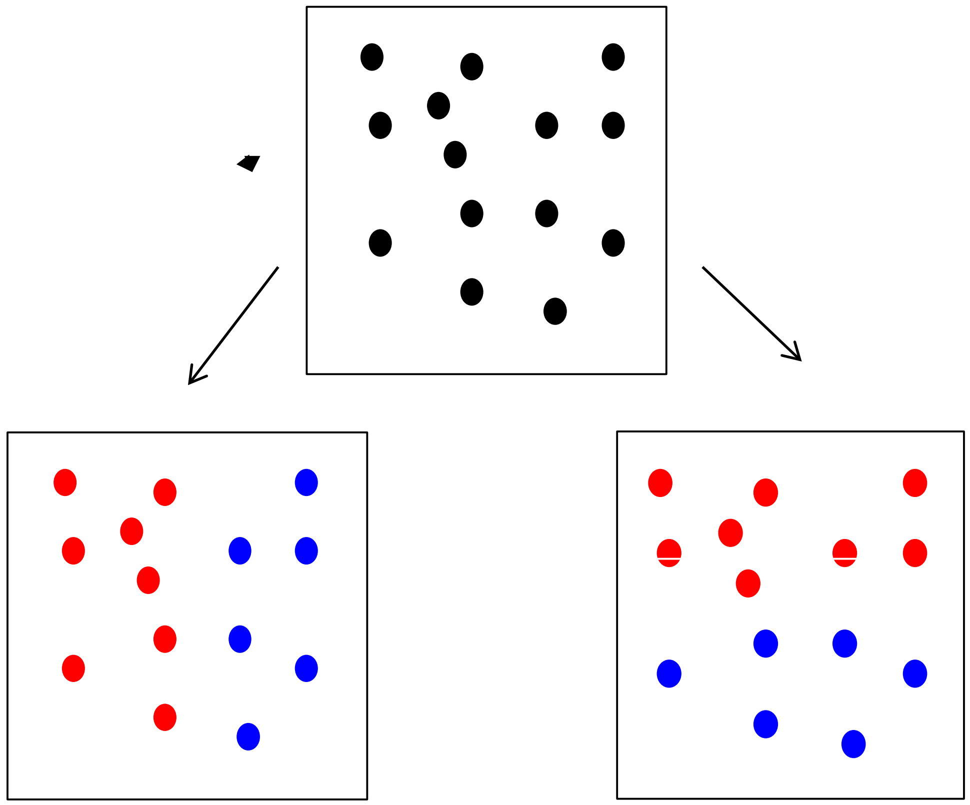
Find hidden structure in the data

- Clustering
- Dimensionality Reduction

Clustering: Grouping similar objects



Clustering Problems



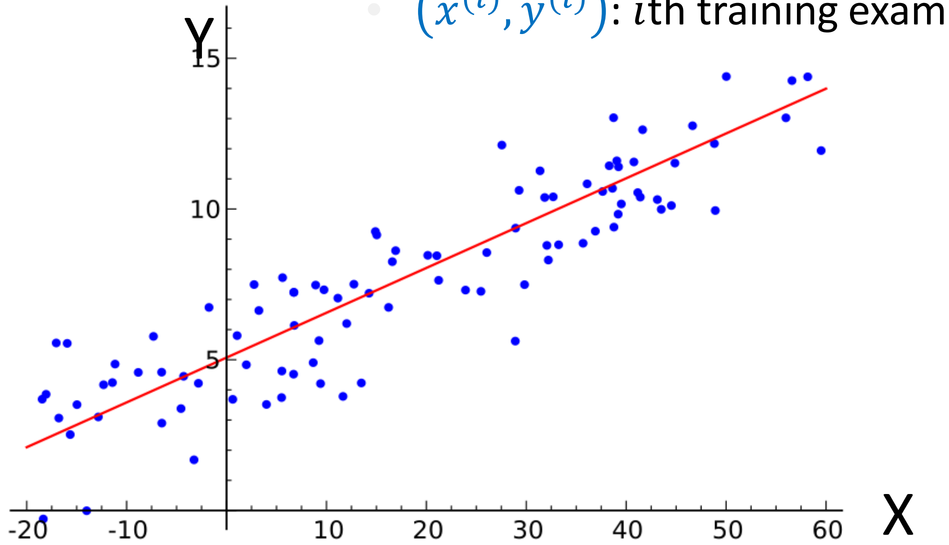
How to evaluate clustering?

- Internal Evaluation:
  - Intra-cluster distances are minimized
  - Inter-cluster distances are maximized
- External Evaluation

Linear Models

Linear Regression

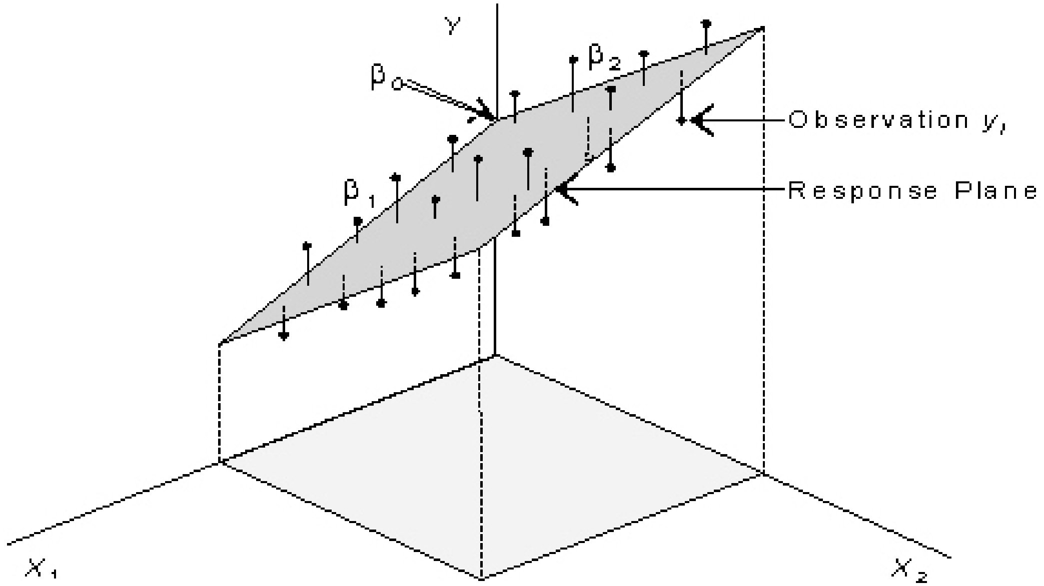
- n features
- m training examples
- $(x^{(i)}, y^{(i)})$: ith training example



Understand the relationship between dependent variable x and explanatory variable y

Objective: predict y from x

Linear Model: $\boxed{Y = \theta X + \theta_0}$

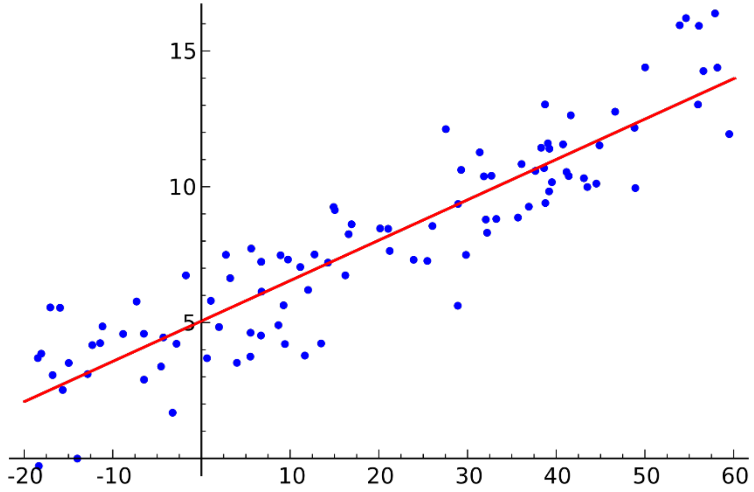


Linear hypothesis function: Intuition

$$h_\theta(\mathrm{x}) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$$

Two equivalent questions:

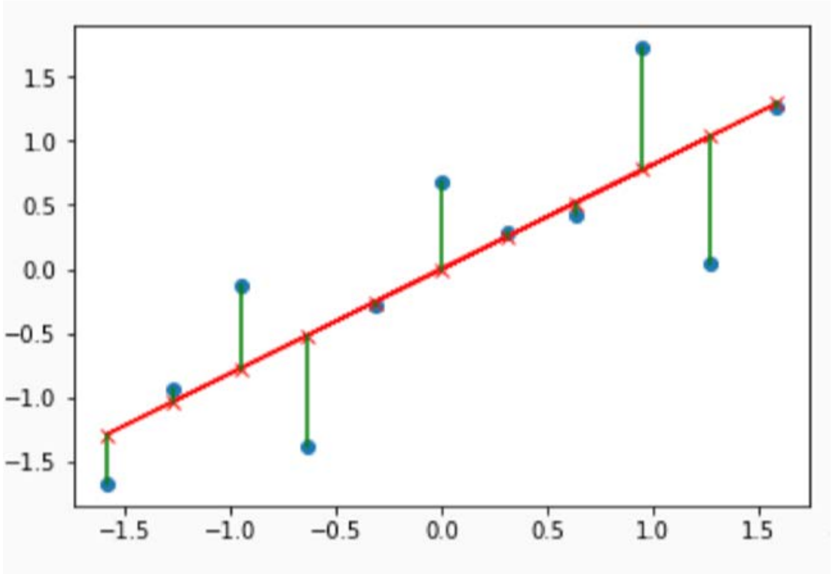
1. Which is the best straight line to fit the data?
2. How to learn the values of the parameters θ_i ?



Cost function

$e^{(i)} = \widehat{y^{(i)}} - y^{(i)} = h_\theta(x^{(i)}) - y^{(i)}$:
prediction error for ith training example

$$J(\bar{\theta}) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$



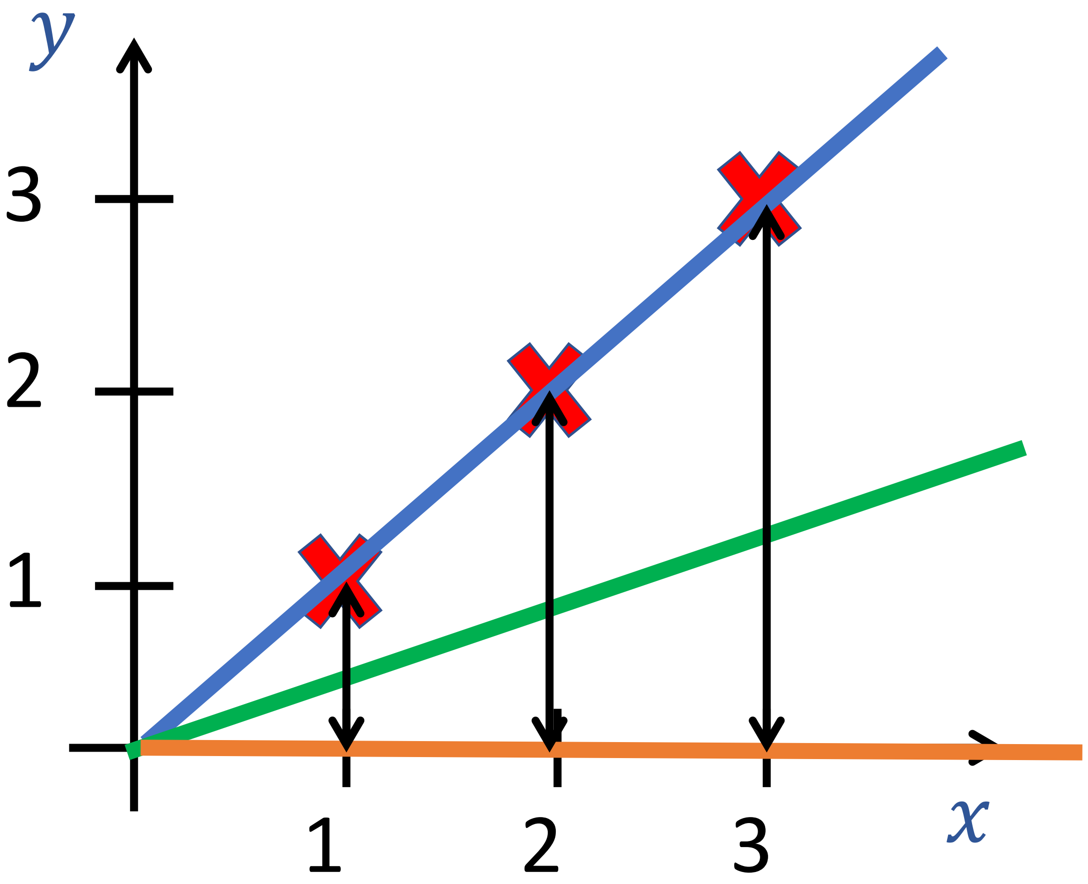
$e^{(i)} = h_\theta(x^{(i)}) - \widehat{y^{(i)}}$

Choose parameters $\bar{\theta}$ so that

$J(\bar{\theta})$ is minimized

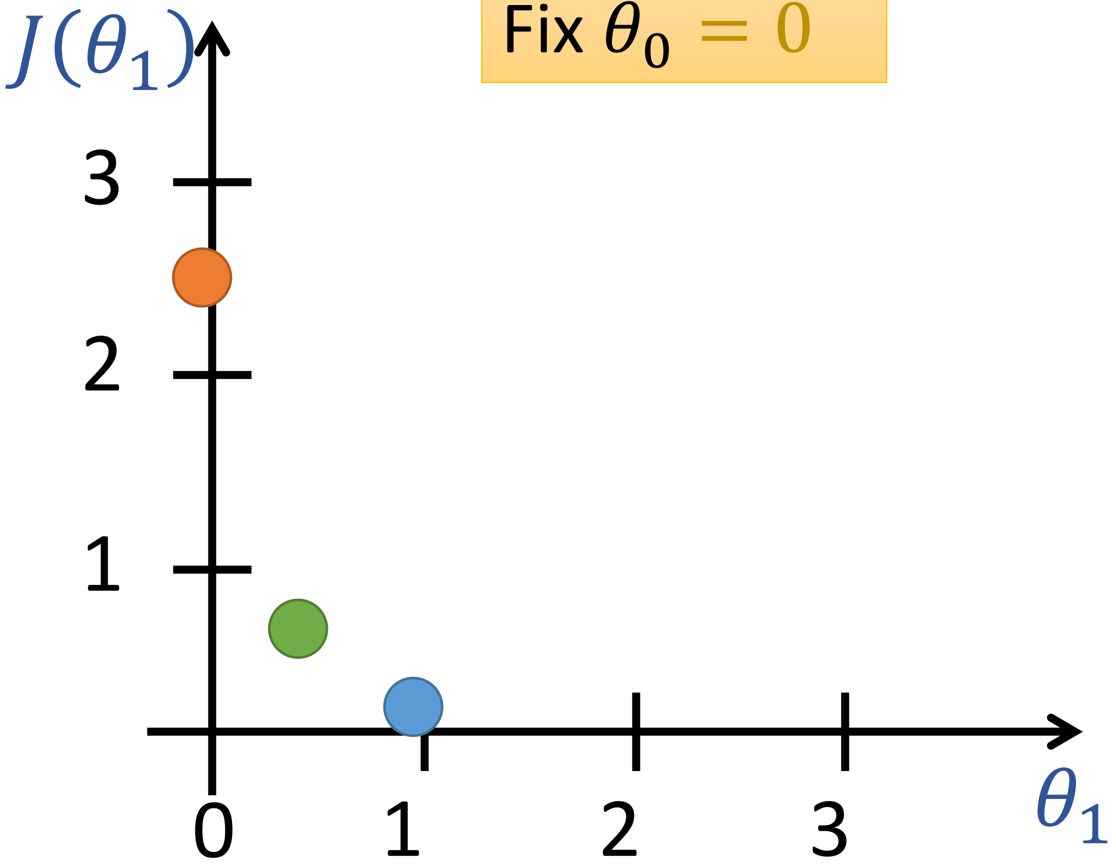
$$\underset{\bar{\theta}}{\text{minimize}}\, J(\bar{\theta})$$

$h_\theta(x)$: function of x for fixed θ

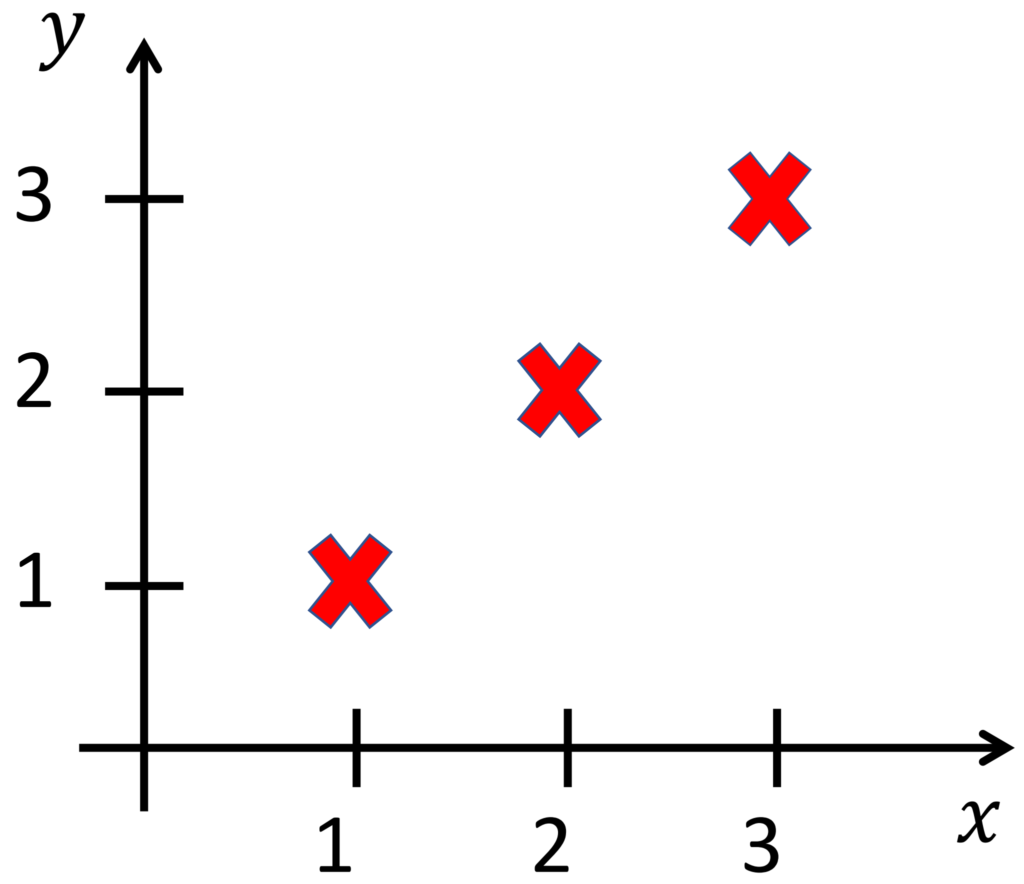


$J(\theta_1)$: function of θ_1

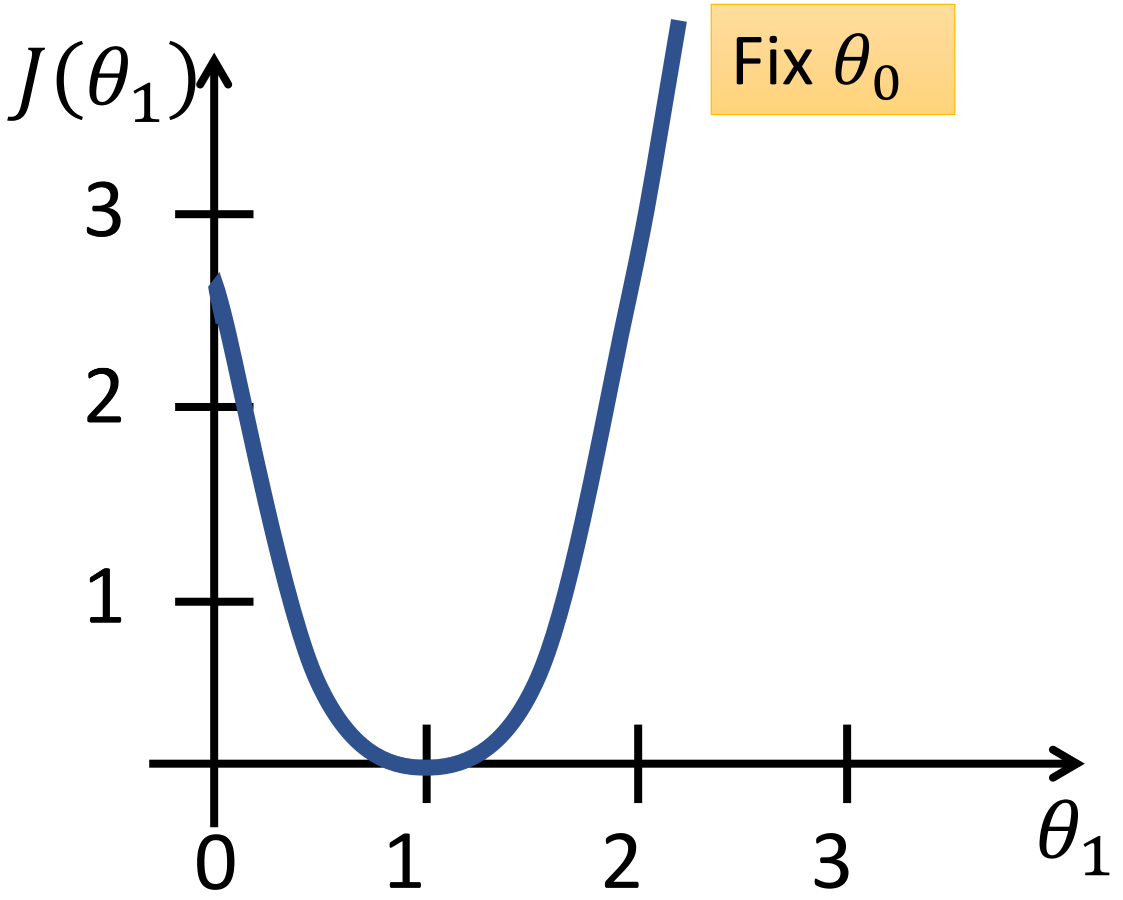
Fix $\theta_0 = 0$



$h_\theta(x)$, function of x for fixed θ



$J(\theta_1)$, function of θ_0, θ_1

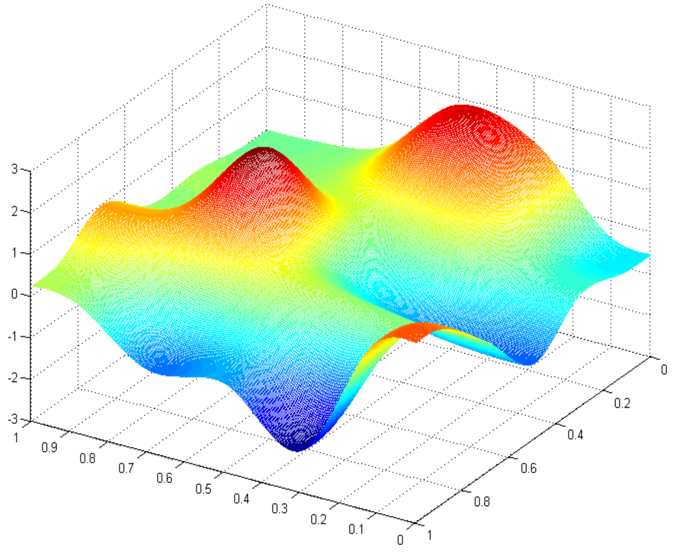


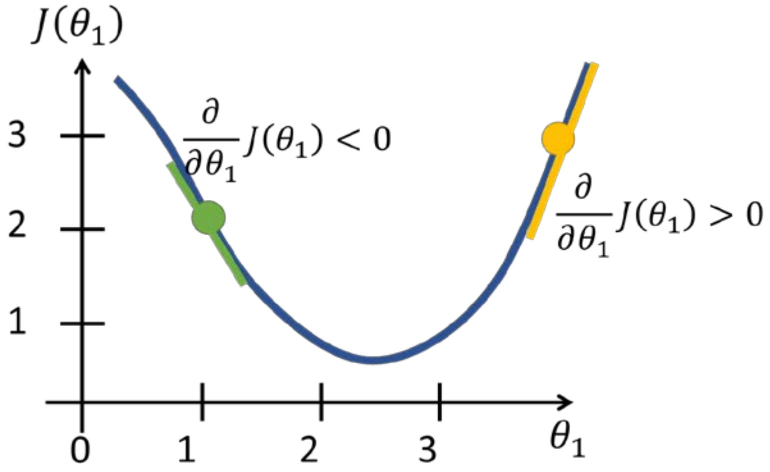
Minimizing cost function & Gradient Descent

Minimizing function $J(\theta_0, \theta_1)$

Outline:

- Start with some θ_0, θ_1
- Keep changing θ_0, θ_1 to reduce $J(\theta_0, \theta_1)$
- until we end up at a minimum





$$\theta_1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_1)$$

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right) x_j^{(i)}$$

Computing partial derivatives

Repeat until convergence{

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\bar{\theta})$$

$$\frac{\partial}{\partial \theta_j} J(\bar{\theta}) = \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right) x_j^{(i)}$$

Equivalently

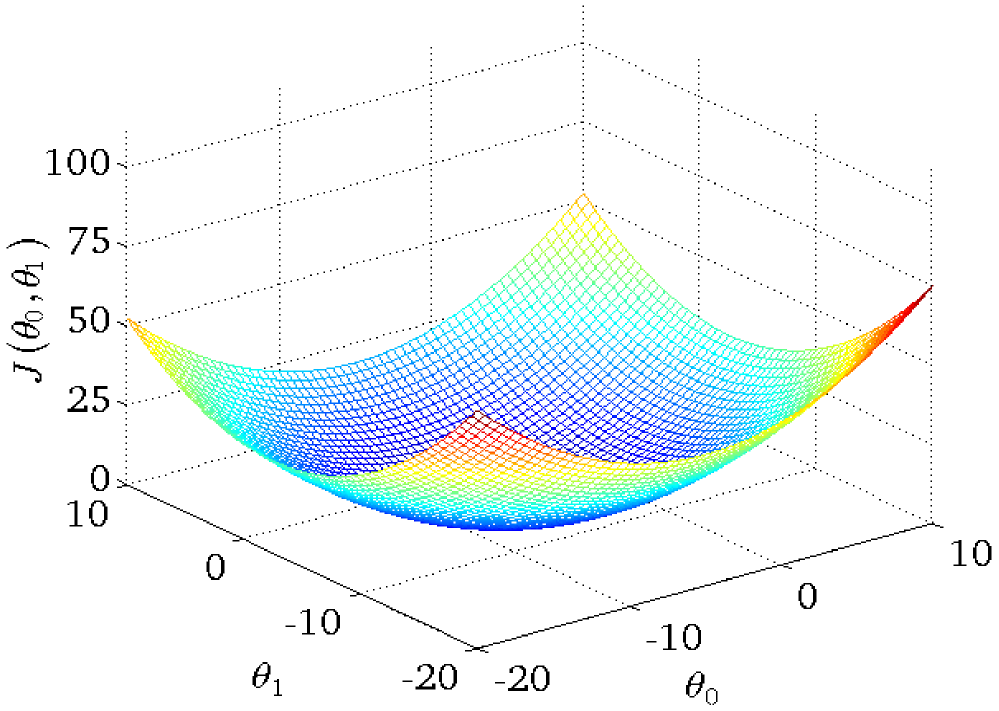
$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right) x_j^{(i)}$$

}

Convergence

- The cost function in linear regression is always a convex function – always has a single global minimum

- So, gradient descent will always converge



Batch gradient descent

"Batch": Each step of gradient descent uses all the training examples

Repeat until convergence{

m: Number of training examples

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{\boxed{m}} \left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)$$

$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{\boxed{m}} \left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right) x^{(i)}$$

}

Logistic Regression

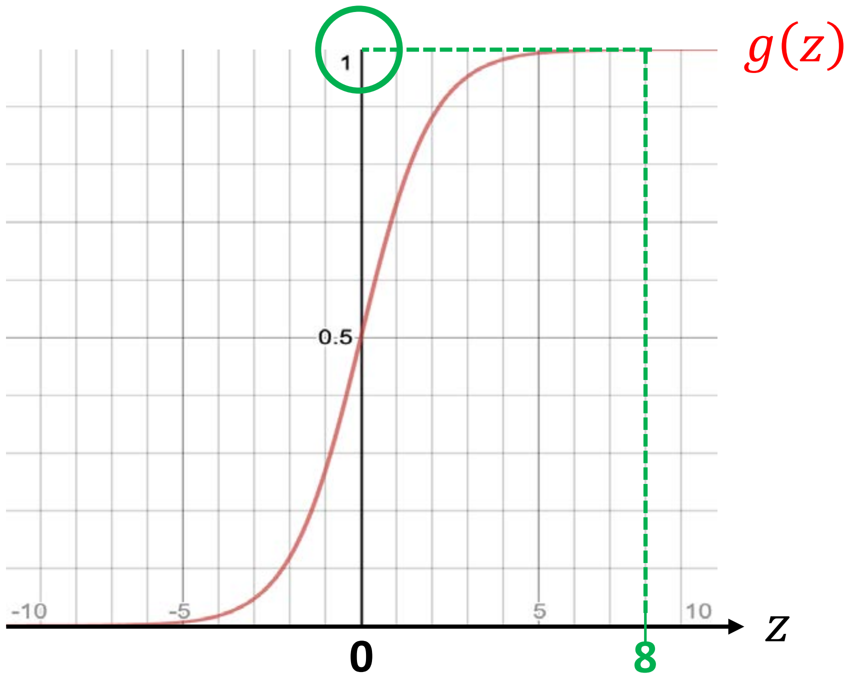
for Classification

Regression vs. Classification

- How can we make the possible outputs of $\boldsymbol{h_\theta(x)} = \boldsymbol{\theta^T x}$ discrete-valued (as opposed to real-valued)?
  - By using an ***activation function*** (e.g., ***sigmoid or logistic function***)

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

$z \in \mathbb{R}, but$
$g(z) \in [0,1]$



Assume a labeled example ($\boldsymbol{x}$, y):

If y = **1**, we want $g(z) \approx 1$ (i.e., we want a correct prediction)

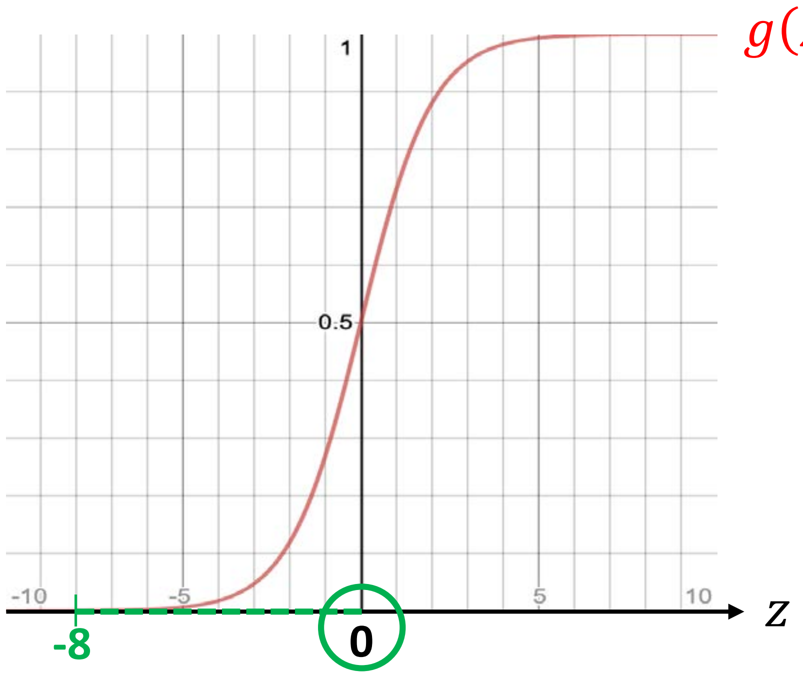
For this to happen, $\boldsymbol{z} \gg \boldsymbol{0}$

Regression vs. Classification

- How can we make the possible outputs of $\boldsymbol{h_\theta(x) = \theta^T x}$ discrete-valued (as opposed to real-valued)?
  - By using an ***activation function*** (e.g., ***sigmoid or logistic function***)

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

$z \in \mathbb{R}, but$
$g(z) \in [0,1]$



Assume a labeled example ($\boldsymbol{x}$, y):

If y = **0**, we want $g(z) \approx 0$ (i.e., we want a correct prediction)

For this to happen, $\boldsymbol{z \ll 0}$

Regression vs. Classification

- How can we make the possible outputs of $\boldsymbol{h_\theta(x)} = \boldsymbol{\theta^T x}$ discrete-valued (as opposed to real-valued)?
  - By using an ***activation function*** (e.g., ***sigmoid or logistic function***)

$$\boldsymbol{x} = [\boldsymbol{x_0}, \ldots, \boldsymbol{x_m}] \longrightarrow g(\underbrace{\theta^T x}_{\boldsymbol{z}}) \longrightarrow \underbrace{\boldsymbol{h_\theta(x)}} \in [0,1]$$

We can apply *thresholding*--
if $\boldsymbol{h_\theta(x)}$ < 0.5, predict 0;
otherwise, predict 1

Alternative Interpretation: $h_\theta(x)$ = estimated probability that $y = 1$ on input x

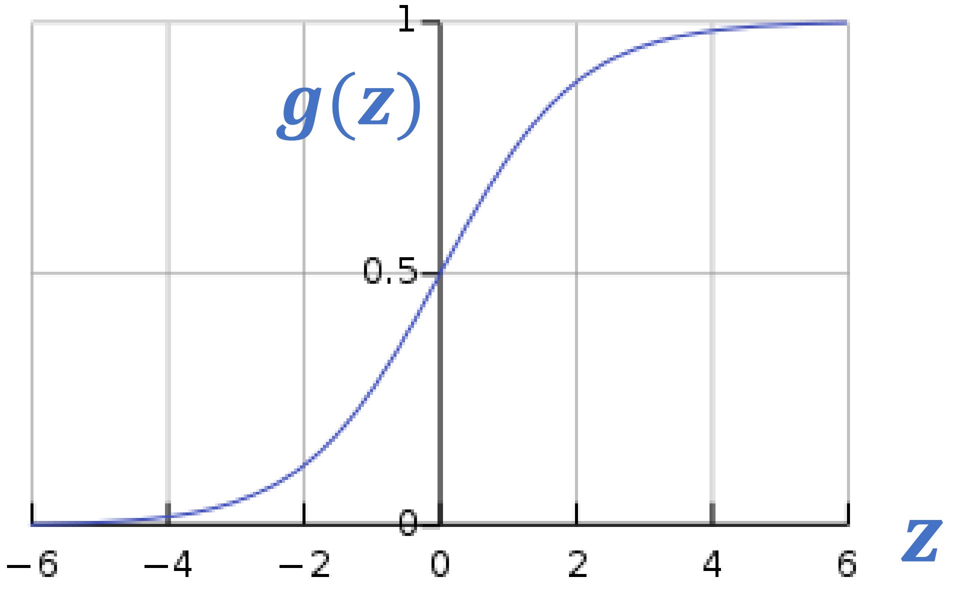
Hypothesis representation

Want $0 \leq h_\theta(x) \leq 1$

$$\boxed{h_\theta(x) = \frac{1}{1 + e^{-\theta^\top x}}}$$

$h_\theta(x) = g(\theta^\top x)$,

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

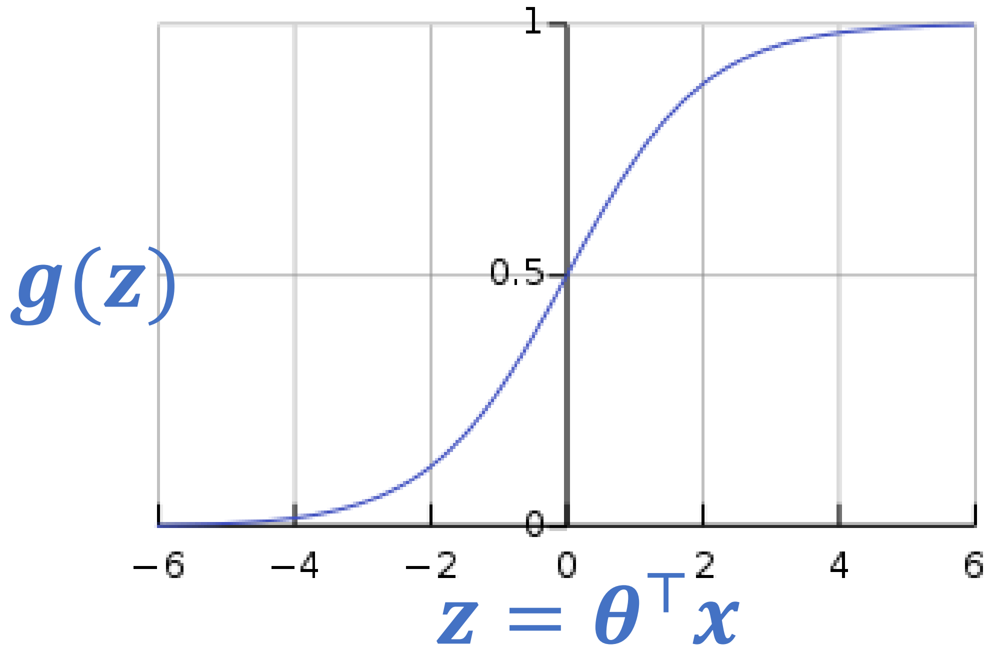


- Sigmoid function
- Logistic function

Logistic regression

$$h_\theta(x) = g(\theta^\top x)$$

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



Suppose predict "y = 1" if $h_\theta(x) \geq 0.5$

$$z = \theta^\top x \geq 0$$

predict "y = 0" if $h_\theta(x) < 0.5$

$$z = \theta^\top x < 0$$

Cost function for Logistic Regression

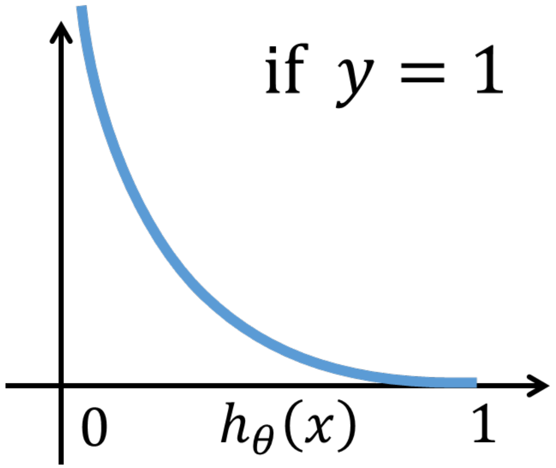
Linear Regression

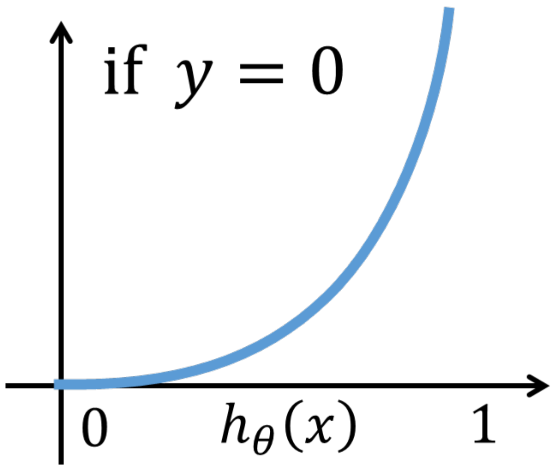
$$\text{Cost}(h_\theta(x), y) = \frac{1}{2}(h_\theta(x) - y)^2$$

Logistic Regression

$$\text{Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$

$$= -y\ \log(h_\theta(x)) - (1 - y)\log(1 - h_\theta(x))$$





Logistic regression

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} \mathrm{Cost}(h_\theta(x^{(i)}), y^{(i)})$$

$$= -\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)} \log\left(h_\theta\left(x^{(i)}\right)\right) + (1 - y^{(i)}) \log\left(1 - h_\theta\left(x^{(i)}\right)\right)\right]$$

Learning: fit parameter θ

$$\min_\theta J(\theta)$$

Prediction: given new x

Output $h_\theta(x) = \frac{1}{1+e^{-\theta^\top x}}$

Gradient descent

$$J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)} \log\left(h_\theta\left(x^{(i)}\right)\right) + (1 - y^{(i)}) \log\left(1 - h_\theta\left(x^{(i)}\right)\right)\right]$$

Goal: $\min_{\theta} J(\theta)$

Good news: Convex function!

Bad news: No analytical solution

Repeat {

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

}

(Simultaneously update all θ_j)

$$\boxed{\frac{\partial}{\partial \theta_j} J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right) x_j^{(i)}}$$

Gradient descent

$$J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)} \log\left(h_\theta(x^{(i)})\right) + (1 - y^{(i)}) \log\left(1 - h_\theta(x^{(i)})\right)\right]$$

Goal: $\min_\theta J(\theta)$

Repeat {

(Simultaneously update all θ_j)

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right) x_j^{(i)}$$

}

Gradient descent for Linear Regression

Repeat {

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right) x_j^{(i)}$$

}

$$\boxed{h_\theta(x) = \theta^\top x}$$

Gradient descent for Logistic Regression

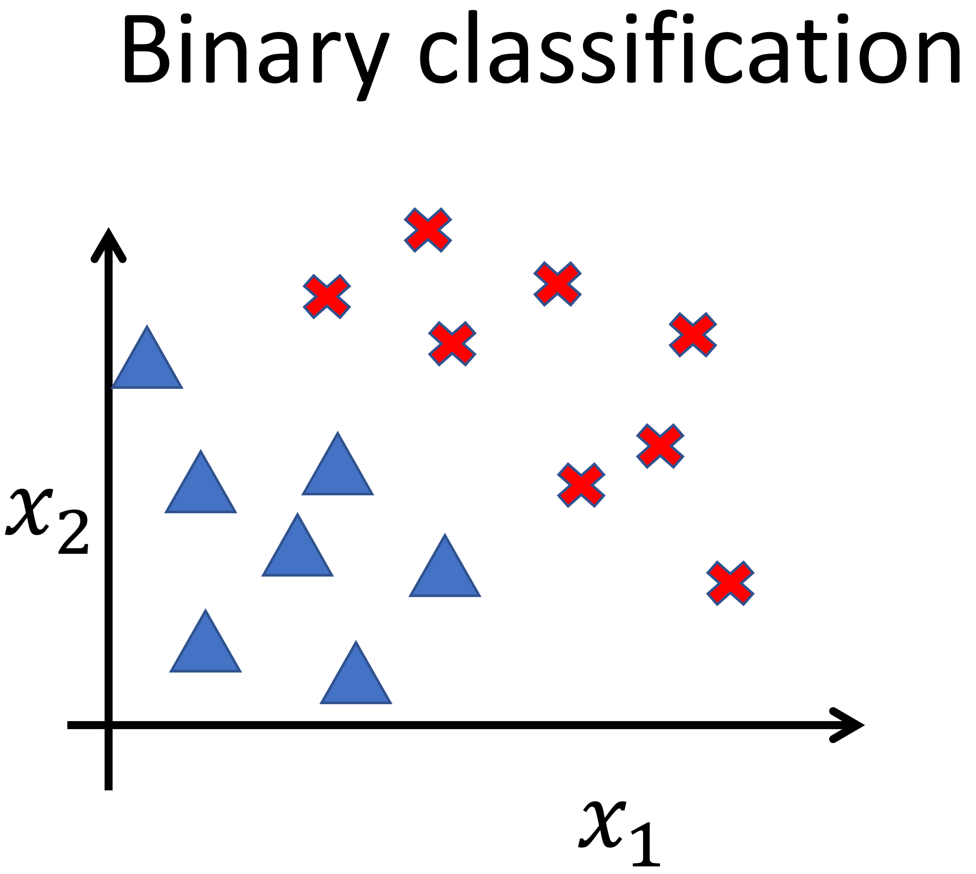
Repeat {

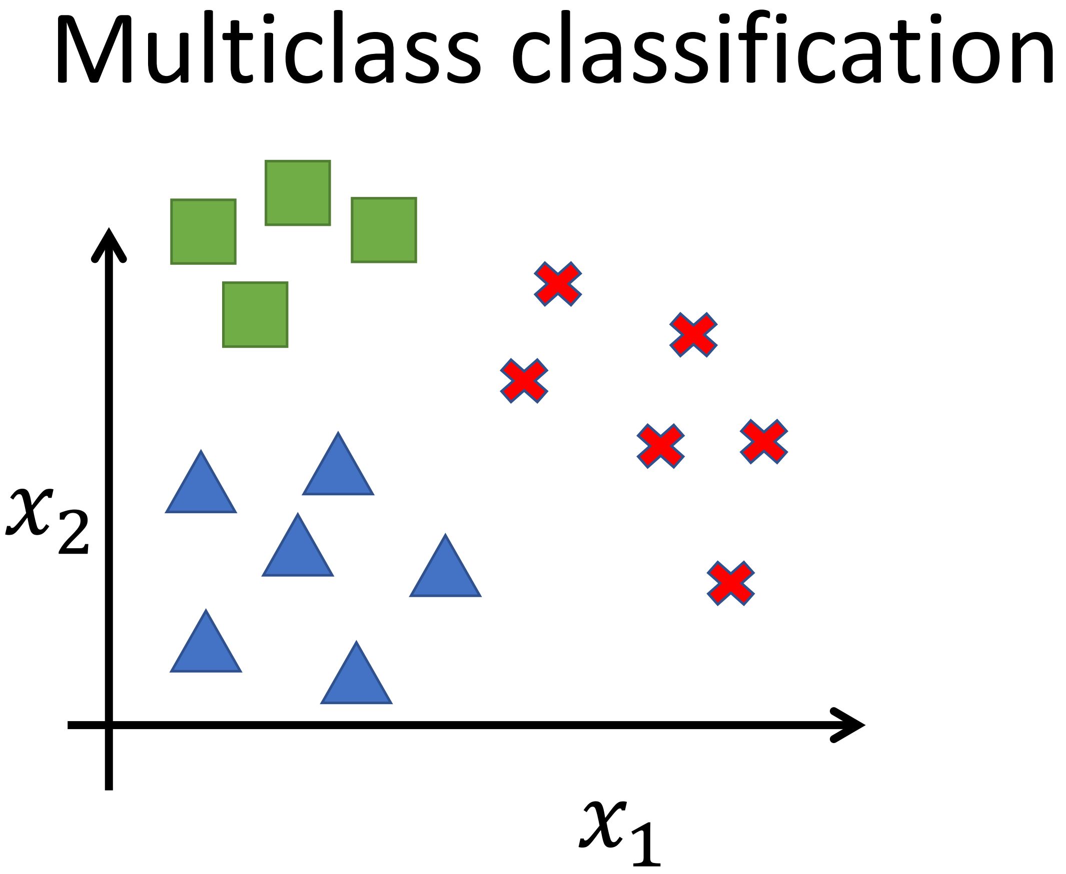
$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right) x_j^{(i)}$$

}

$$\boxed{h_\theta(x) = \frac{1}{1 + e^{-\theta^\top x}}}$$

Multiclass classification





Multi-class Classification

- Binary classification: $h_\theta(x) = \theta^T x$
- Multi-class Classification: y can take on K different values $y^{(i)} \in \{1, 2, \ldots, K\}$
- $h_\theta(x)$ estimates the probability of belonging to each class

$$P(y = k|x, \theta) \propto \exp\left(\theta_k^T x\right)$$

$$\theta = \begin{bmatrix} \vdots & \vdots & \vdots \\ \theta_1 & \theta_2 & \theta_k \\ \vdots & \vdots & \vdots \end{bmatrix}$$

$$P(y = k|x, \theta) = \frac{\exp\left(\theta_k^T x\right)}{\sum_{j=1}^{K} \exp\left(\theta_j^T x\right)}$$

Multi-class Classification Cost Function

$$P(y = k | x, \theta) = \frac{\exp\left(\theta_k^T x\right)}{\sum_{j=1}^{K} \exp\left(\theta_j^T x\right)}$$

$$J(\theta) = -\left[\sum_{i=1}^{m} \sum_{j=1}^{K} 1\{y^{(i)} = k\} \log \frac{\exp\left(\theta_k^T x^{(i)}\right)}{\sum_{j=1}^{K} \exp\left(\theta_j^T x^{(i)}\right)}\right]$$

Machine Learning– Part 2

Neural Networks

Biological Neural Network

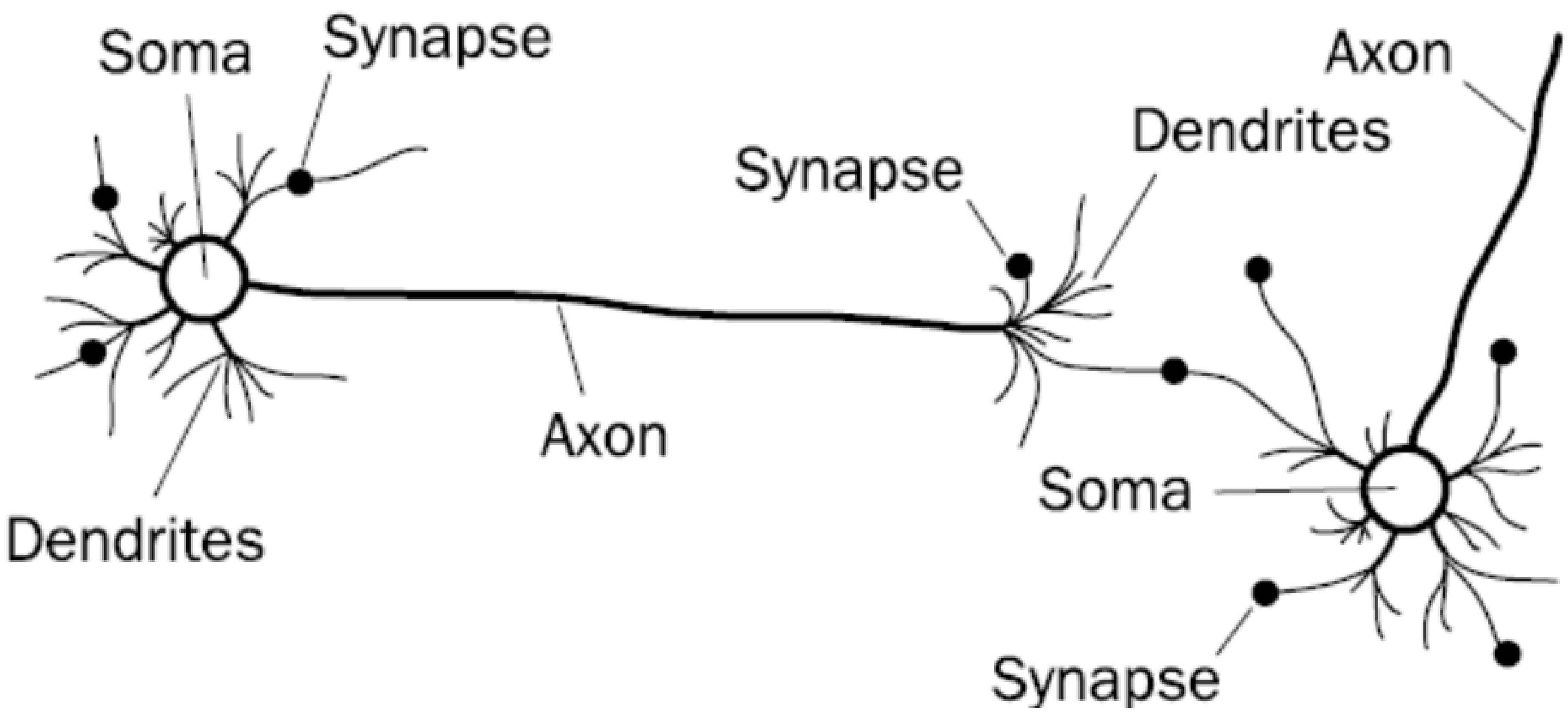
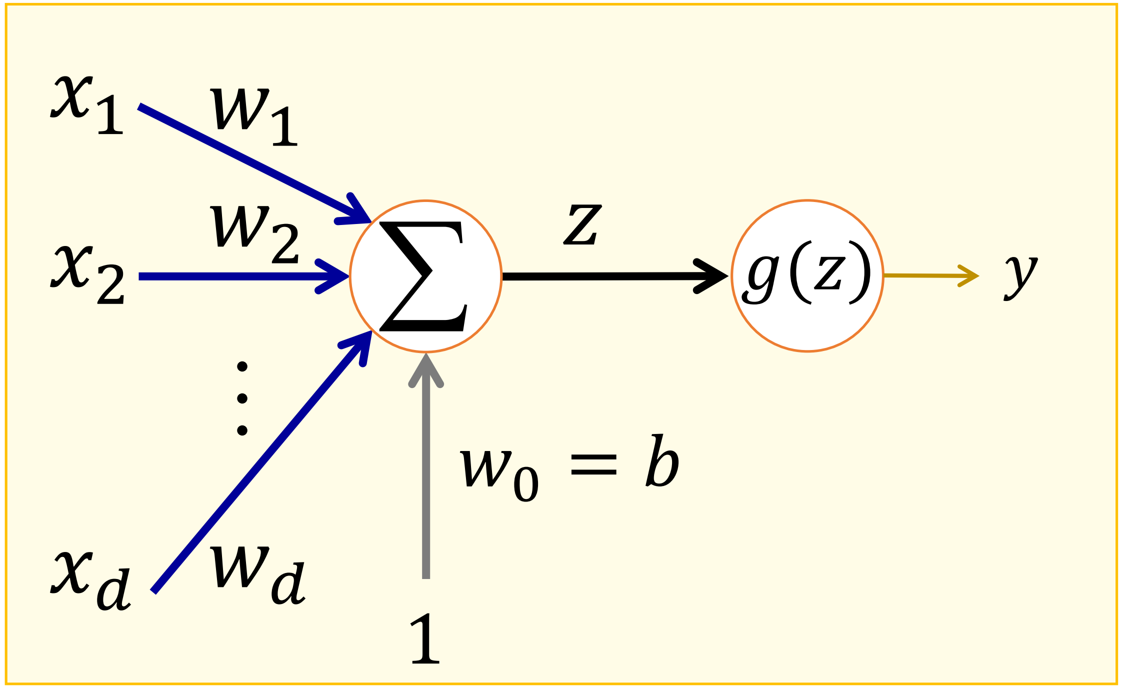


Image courtesy: F. A. Makinde et. al., “Prediction of crude oil viscosity using feed-forward back-propagation neural network (FFBPNN)”." Petroleum & Coal 2012

Artificial Neuron



Terminologies

$\boldsymbol{x}$**:** input, $\boldsymbol{w}$**:** weights, $\boldsymbol{b}$**:** bias

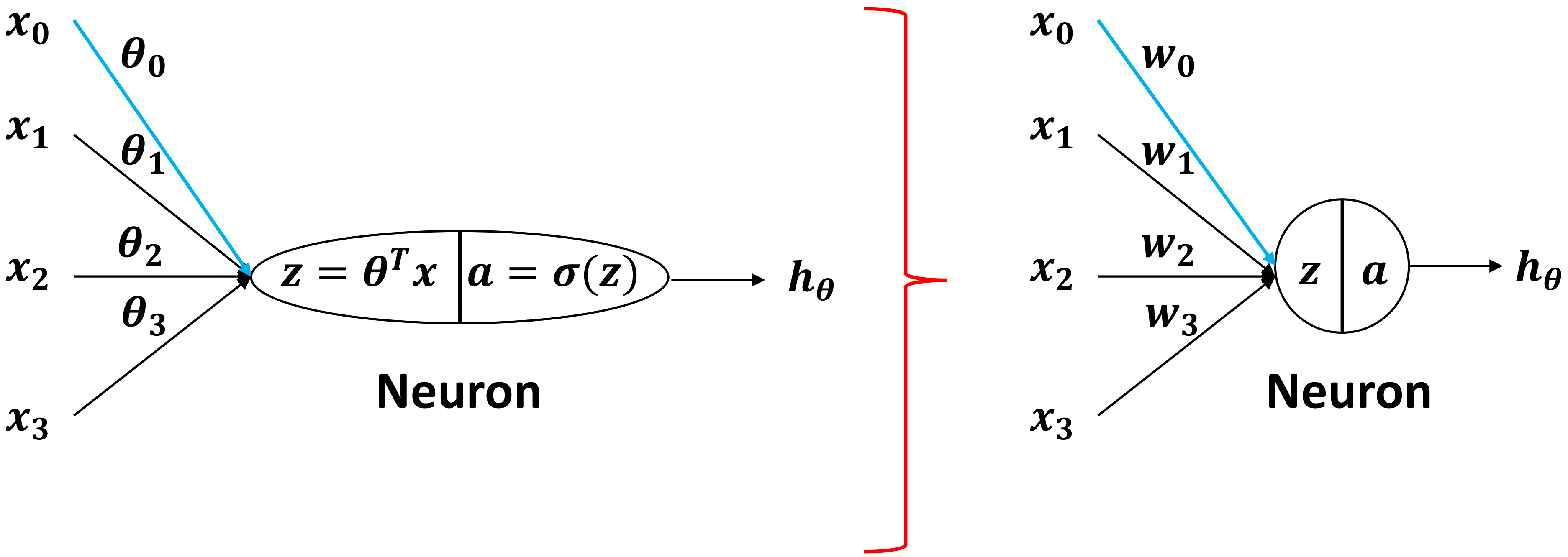
z: pre-activation (input activation)

g: activation function

y: activation for output units

Towards Neural Networks

- Logistic regression is a **neural network** with only 1 neuron



Output Units: Linear

$$\hat{y} = w^T a + b$$

Used to produce the mean of a conditional Gaussian distribution:

$$p(\mathbf{y} \mid \mathbf{x}) = N(\mathbf{y}; \hat{\mathbf{y}}, \sigma)$$

Maximizing log-likelihood $\Longrightarrow$ Minimizing squared error

Output Units: Sigmoid

$$\hat{y} = \sigma(w^T a + b)$$

$$J(\theta) = -\log p(y|x)$$
$$= -\log \sigma\big((2y-1)(\boldsymbol{w}^T \boldsymbol{a} + b)\big)$$

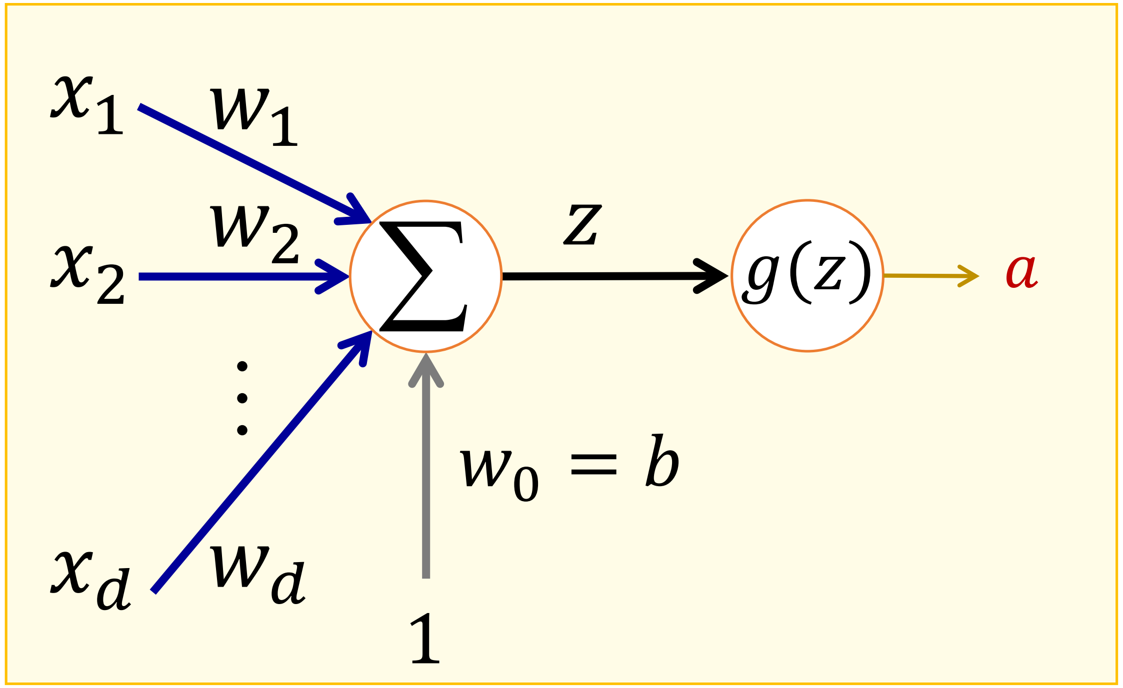
Output Softmax Units

Need to produce a vector $\hat{y}$ with $\hat{y}_i = p(y = i|x)$

$$\mathrm{softmax}(z)_i = \frac{\exp(z_i)}{\sum_j \exp(z_j)}$$

$$\log \mathrm{softmax}(z)_i = z_i - \log \sum_j \exp(z_j)$$

Artificial Neuron – hidden unit

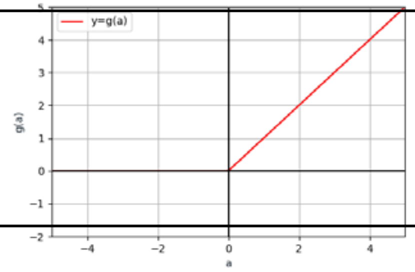


$\boldsymbol{w} = [w_1\ w_2 \ldots w_d]^T$ and $\boldsymbol{x} = [x_1\ x_2 \ldots x_d]^T$

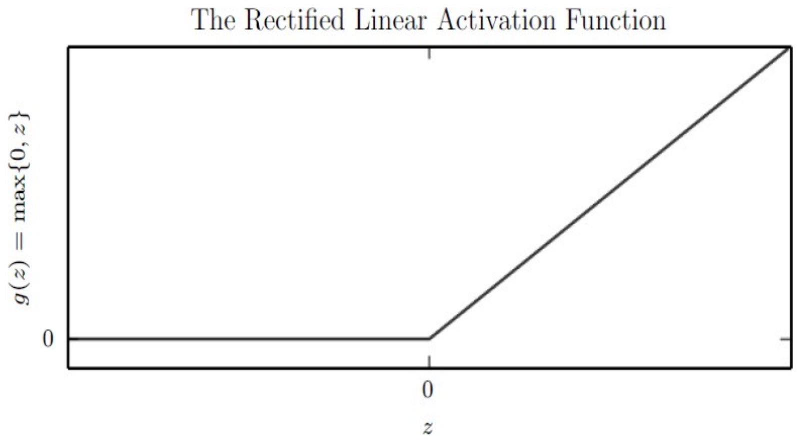
$$\boldsymbol{z} = b + \sum_{i=1}^{d} w_i x_i = [\boldsymbol{w}^T b]\begin{bmatrix}\boldsymbol{x}\\ 1\end{bmatrix}$$

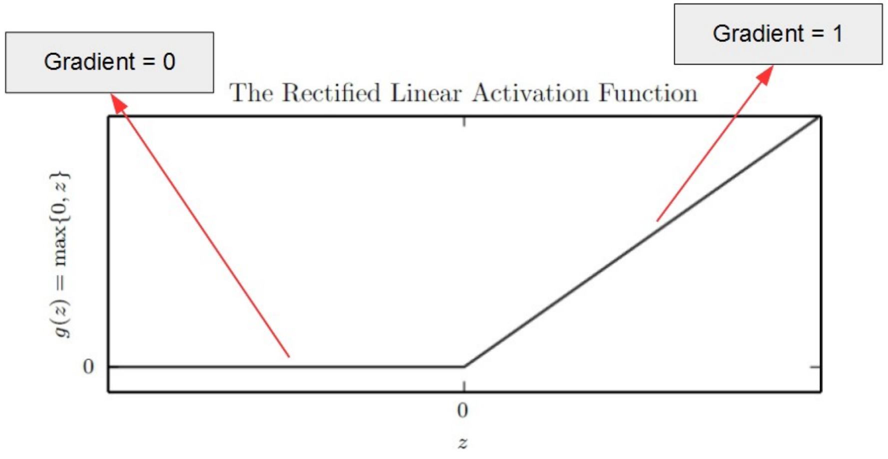
$$\boldsymbol{a} = g(z)$$

Activation Functions for Hidden Nodes

Name	Function	Gradient	Graph
Sigmoid	$\sigma(z) = \dfrac{1}{1+\exp(-z)}$	$g'(z)$ $= g(z)(1-g(z))$	
Tanh	$\tanh(z) = \dfrac{\exp(z)-\exp(-z)}{\exp(z)+\exp(-z)}$	$g'^{(z)} = 1 - g^2(z)$	
ReLU	$g(z) = \max(0, z)$	$g'(z) = \begin{cases} 1, & z \geq 0 \\ 0, & z < 0 \end{cases}$	
softplus	$g(z) = \ln(1+e^z)$		

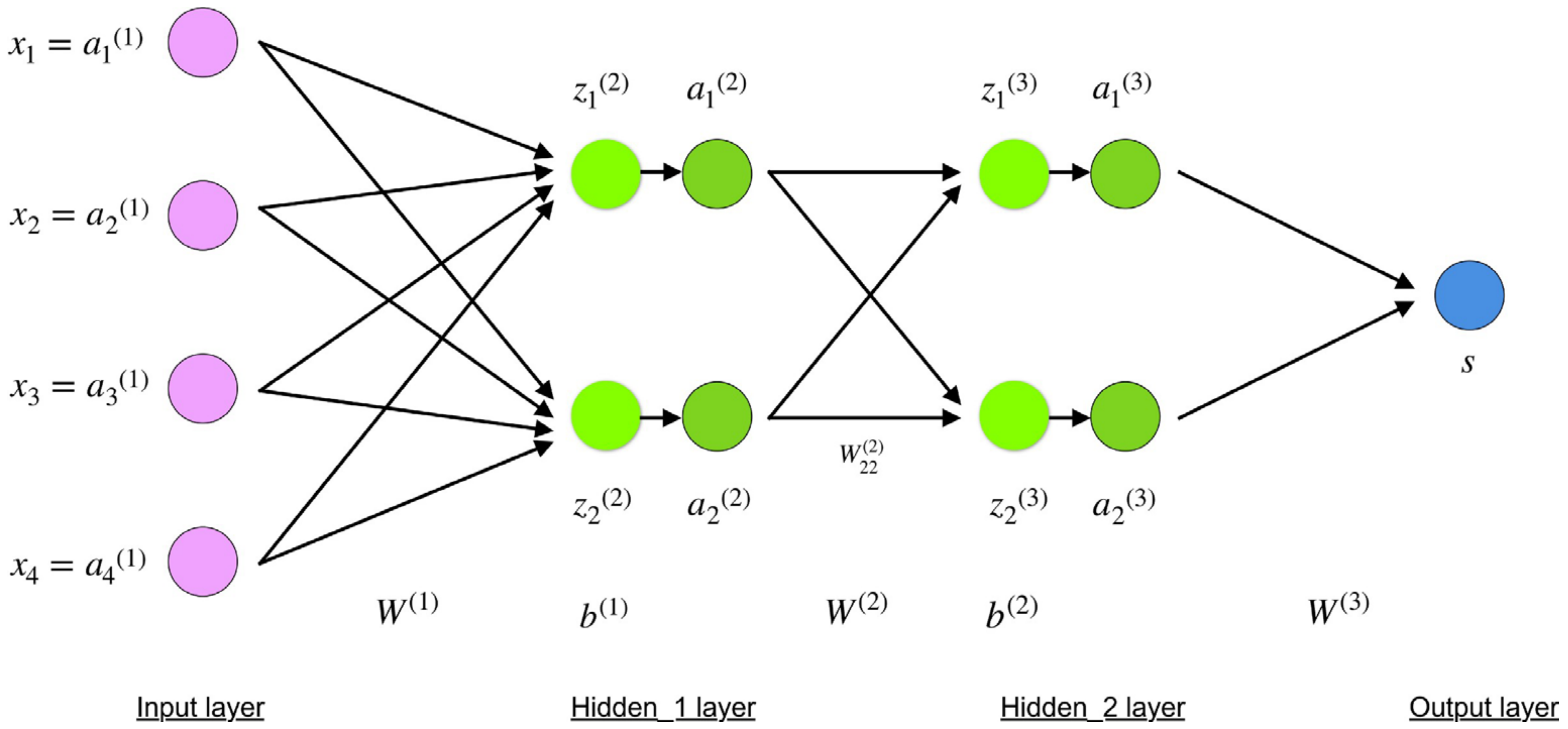
Rectified Linear Units





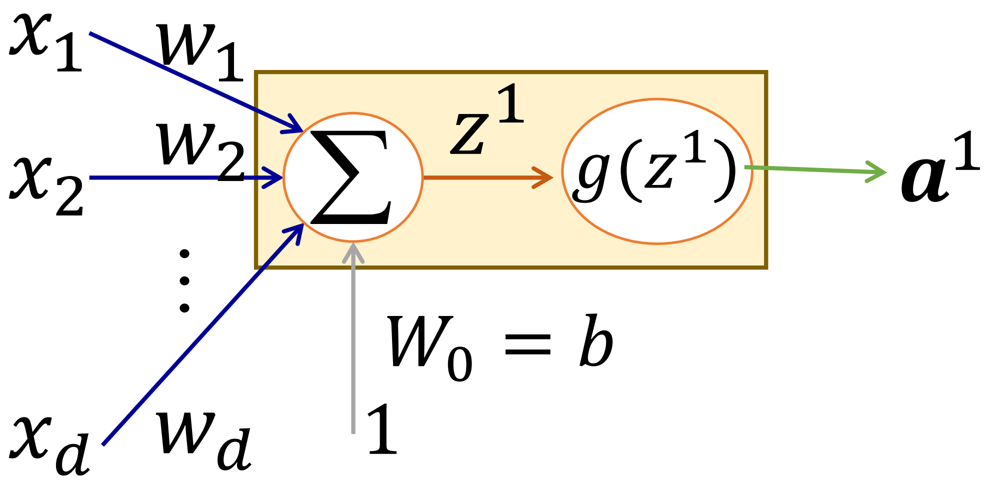
- Activation function: $g(z) = \max\{0, z\}$
  - Gives large and *consistent* gradients (does not saturate) when active
  - Efficient to optimize, converges much faster than sigmoid or tanh

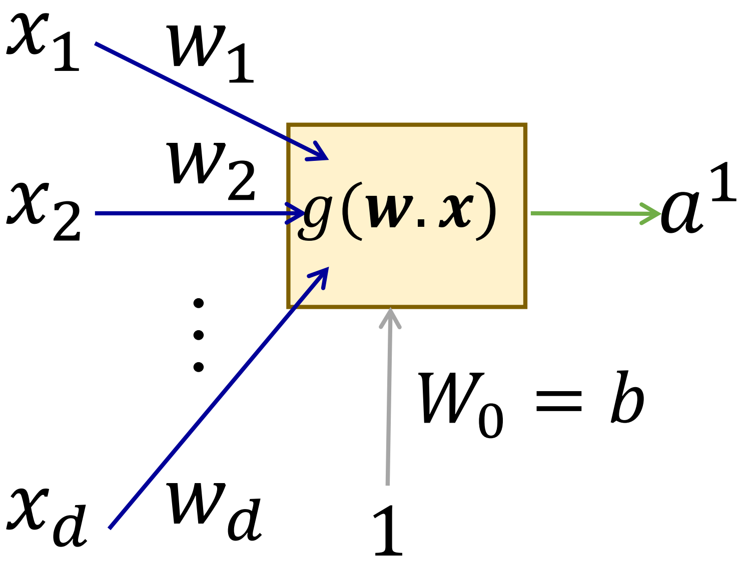
Multilayer Neural Network



https://towardsdatascience.com/understanding-backpropagation-algorithm-7bb3aa2f95fd

Basic Neural Units



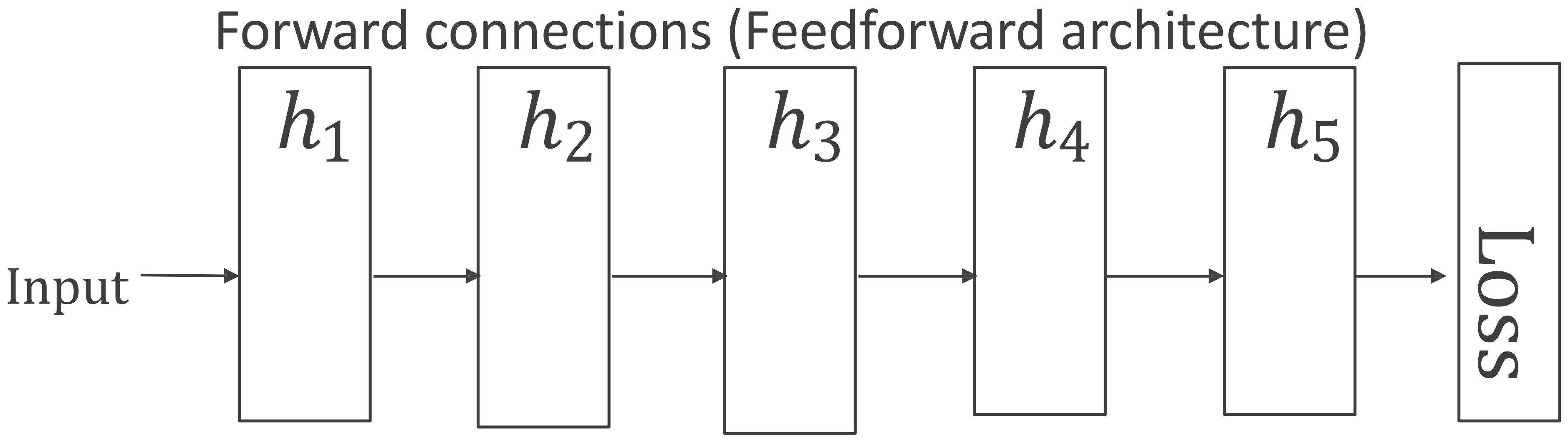


$$z_1^1 = b_1^1 + \sum_{i=1}^{d} w_{1,i}^1 \, x_i$$

$$a_1^1 = g(z_1^1)$$

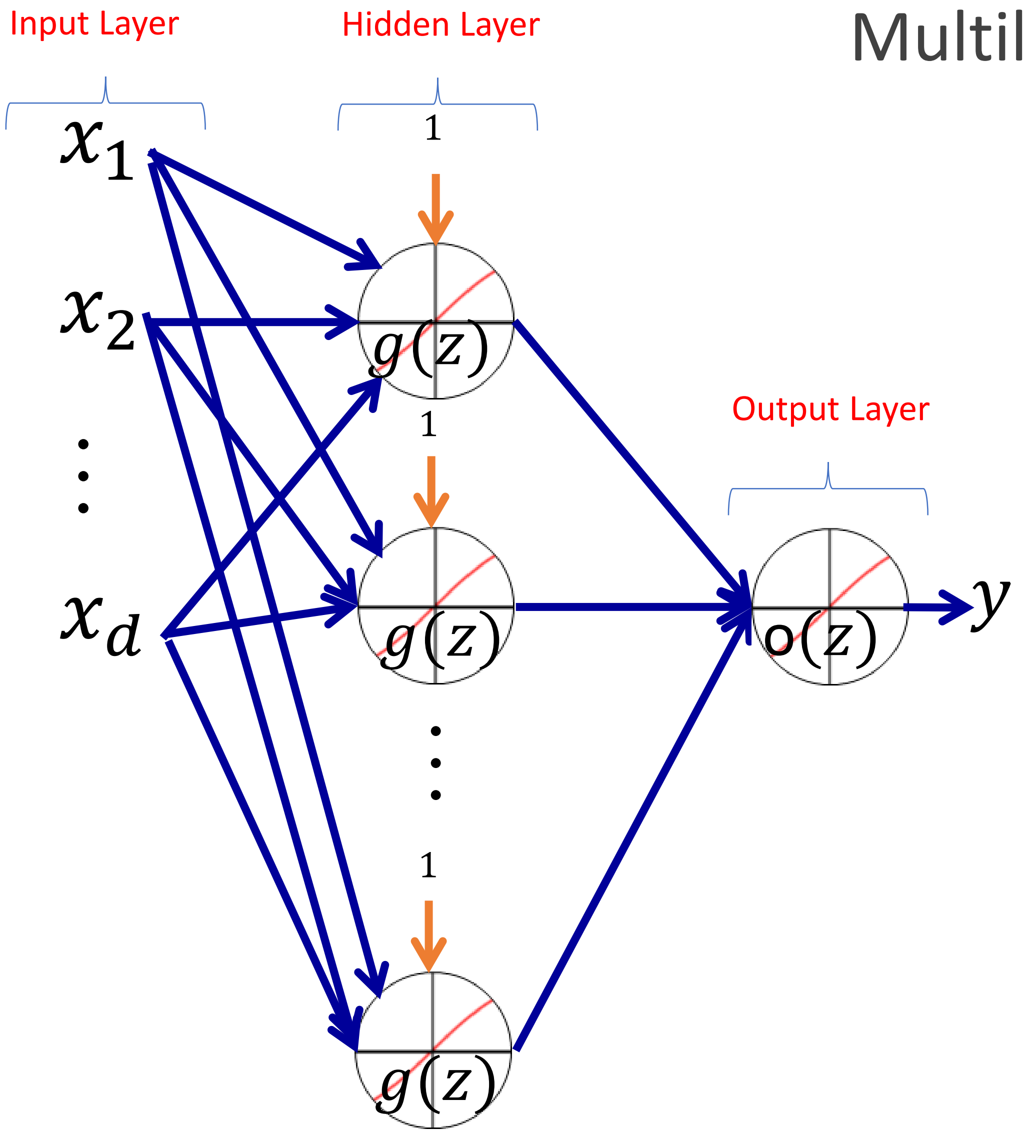
Neural networks in blocks

We can visualize $a_L = h_L \circ h_{L-1} \circ \cdots \circ h_1(x)$ as a cascade of blocks.



The activation functions must be **1st-order differentiable (almost) everywhere**

Multilayer Neural Network

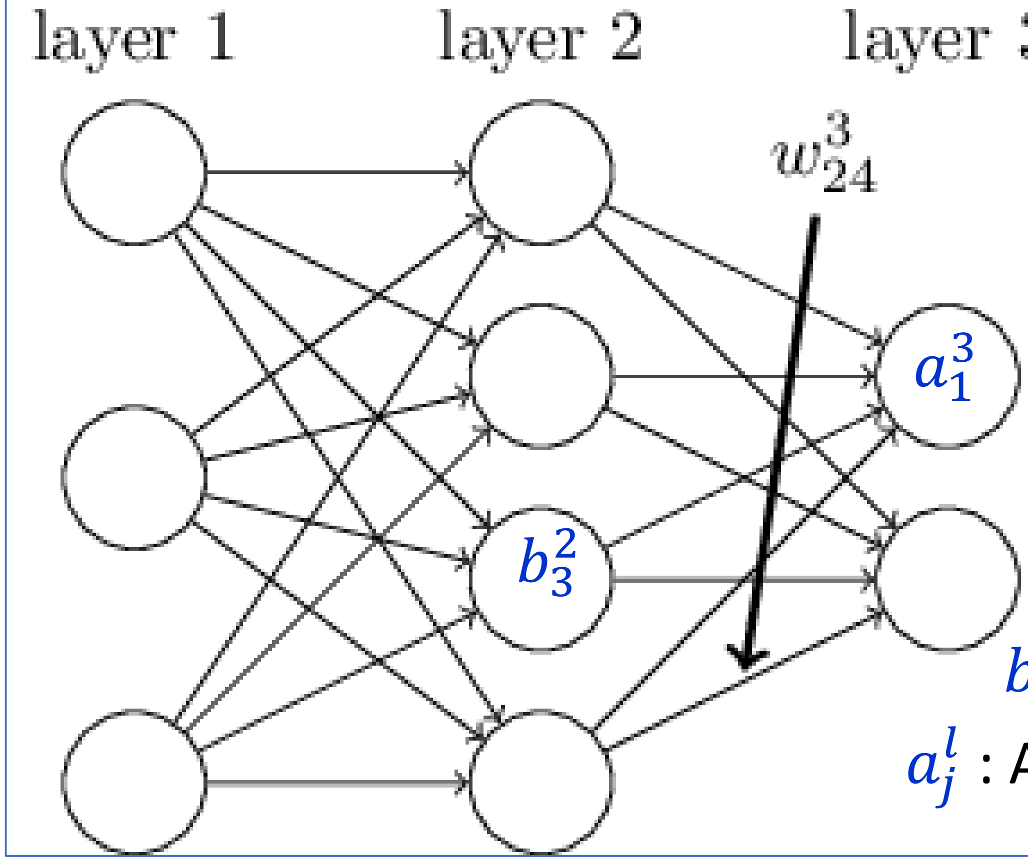


Intuition:

Hidden Layer: Extracts better representation of the input data

Output layer: Does the classification

Notations



w_{jk}^l is the weight from the k^{th} neuron in the $(l-1)^{\text{th}}$ layer to the j^{th} neuron in the l^{th} layer

b_j^l : Bias for jth neuron in lth layer.

a_j^l : Activation of the jth neuron in the lth layer.

$$a_j^l = g\left(\sum_k w_{jk}^l\, a_k^{l-1} + b_j^l\right)$$

Vectorized form: $a^l = g\left(w^l a^{l-1} + b^l\right)$

$$z^l = w^l a^{l-1} + b^l$$

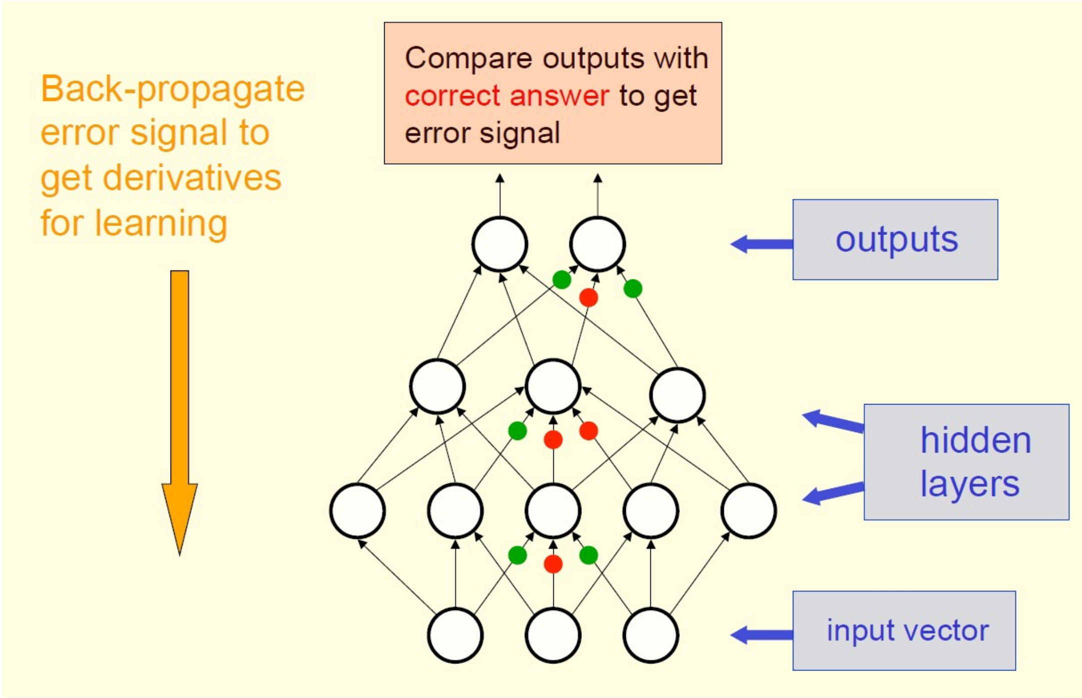
$$a^l = g\left(z^l\right)$$

Backpropagation

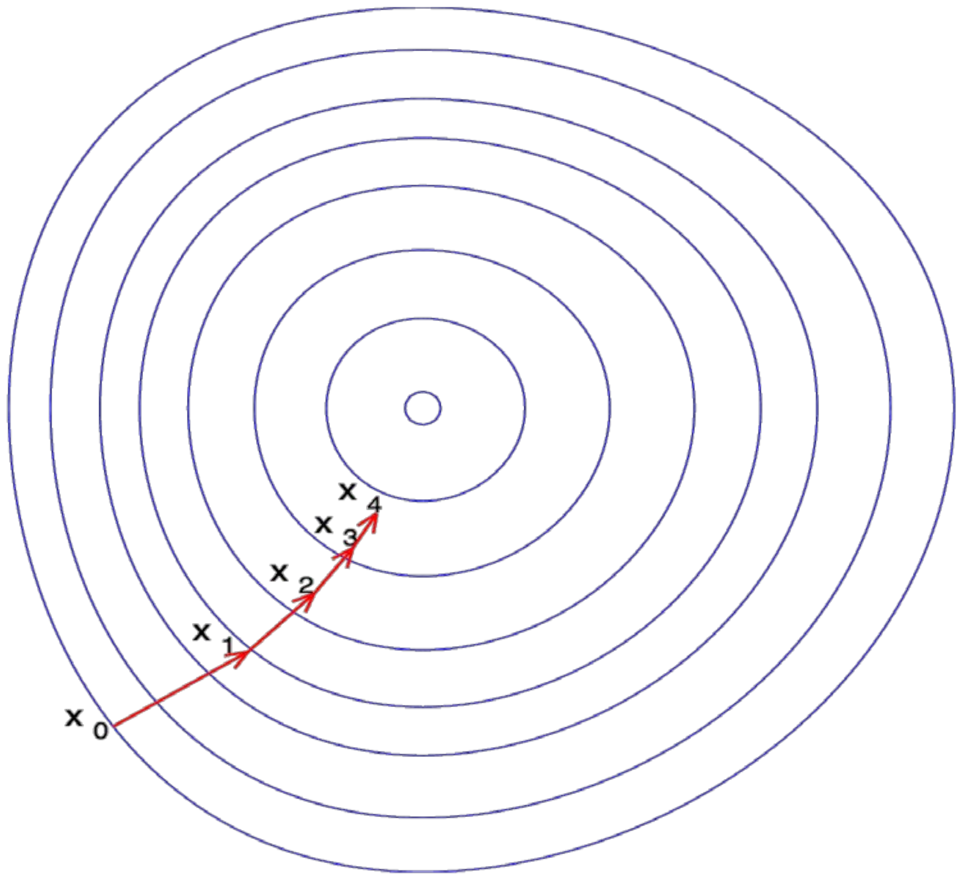
- Feedforward Propagation: Accept input $x^{(i)}$, pass through intermediate stages and obtain output $\hat{y}^{(i)}$
- During Training: Compute scalar cost J(θ)

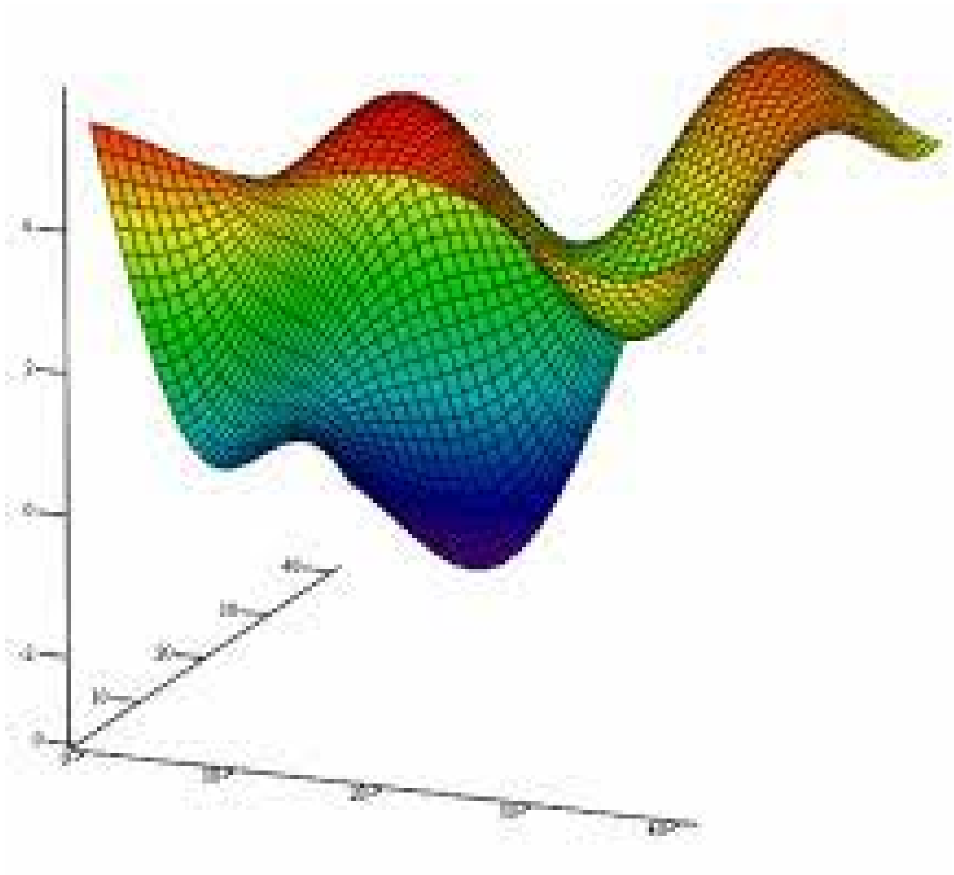
$$J(\theta) = \sum_i L\left(NN\left(x^{(i)}; \theta\right), y^{(i)}\right)$$

- Backpropagation allows information to flow backwards from cost to compute the gradient



Gradient Descent



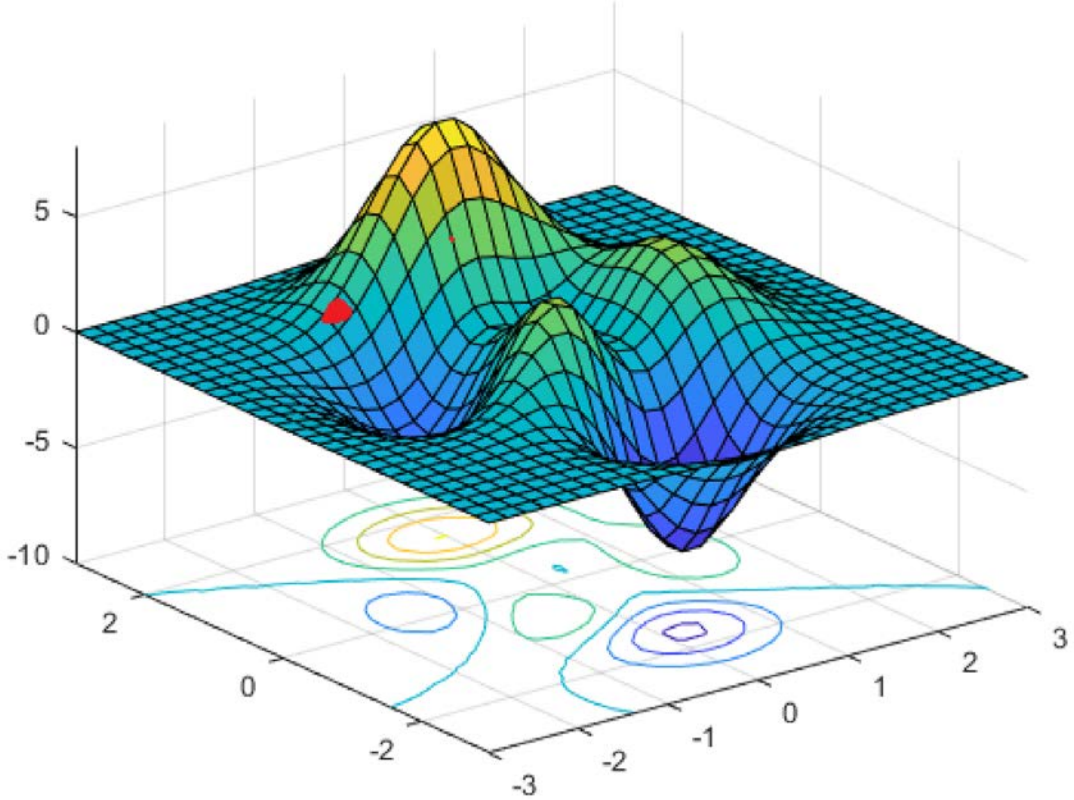


Stochastic gradient descent

Sample rather than computing the full sum

While not converged:

- **Select a single datapoint in order from the data**
- **Compute gradient with just one point**
- **Update parameters**

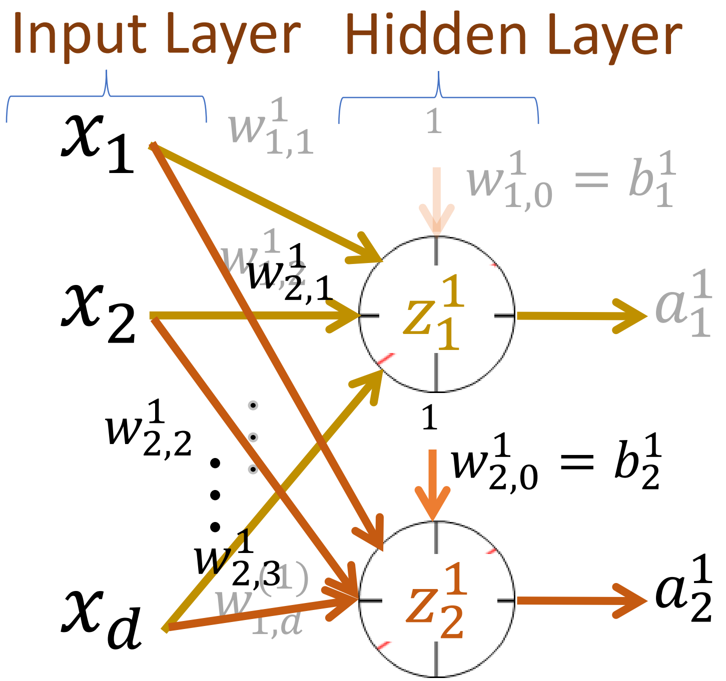


Mini-batch SGD:

While not converged:

- **Select a random batch of datapoints**
- **Compute gradient with the batch**
- **Update parameters**

Multilayer Neural Network



$z^1_1 = b^1_1 + \sum_{i=1}^{d} w^1_{1,i}\, x_i \qquad a^1_1 = g(z^1_1)$

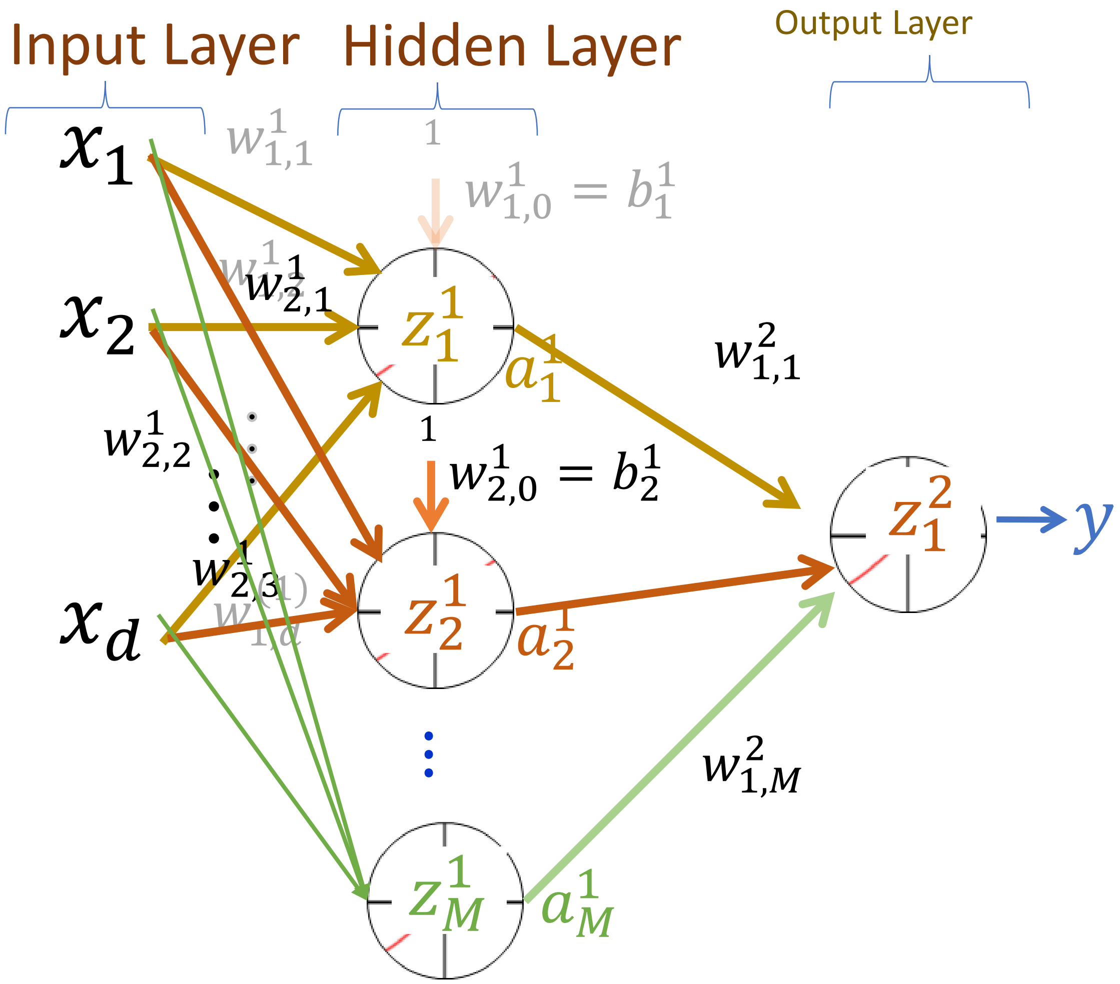
$z^1_2 = b^1_2 + \sum_{i=1}^{d} w^1_{2,i}\, x_i \qquad a^1_2 = g(z^1_2)$

.. … …

$z^1_m = b^1_m + \sum_{i=1}^{d} w^1_{m,i}\, x_i \qquad a^1_m = g(z^1_m)$

$$a^{(0)} = x$$
$$z^{(1)} = \boldsymbol{w}^{(1)}\boldsymbol{a}^{(0)}$$
$$a^{(1)} = g\left(\boldsymbol{z}^{(1)}\right)$$

Multilayer Neural Network



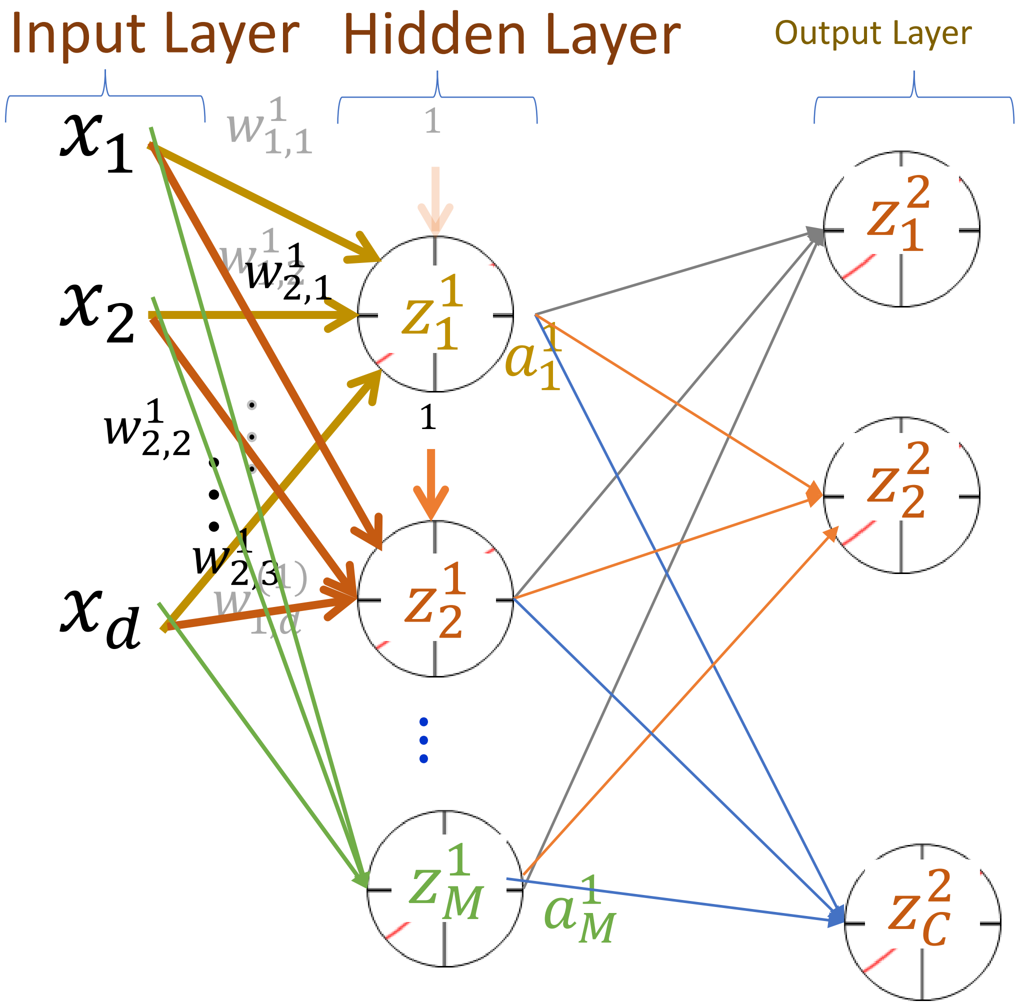
Output Layer Activation

$$y_1 = o(\boldsymbol{z_1^2})$$

output

- Sigmoid for 2-class classification
- Softmax for multi-class classification
- Linear for regression

Multilayer Neural Network

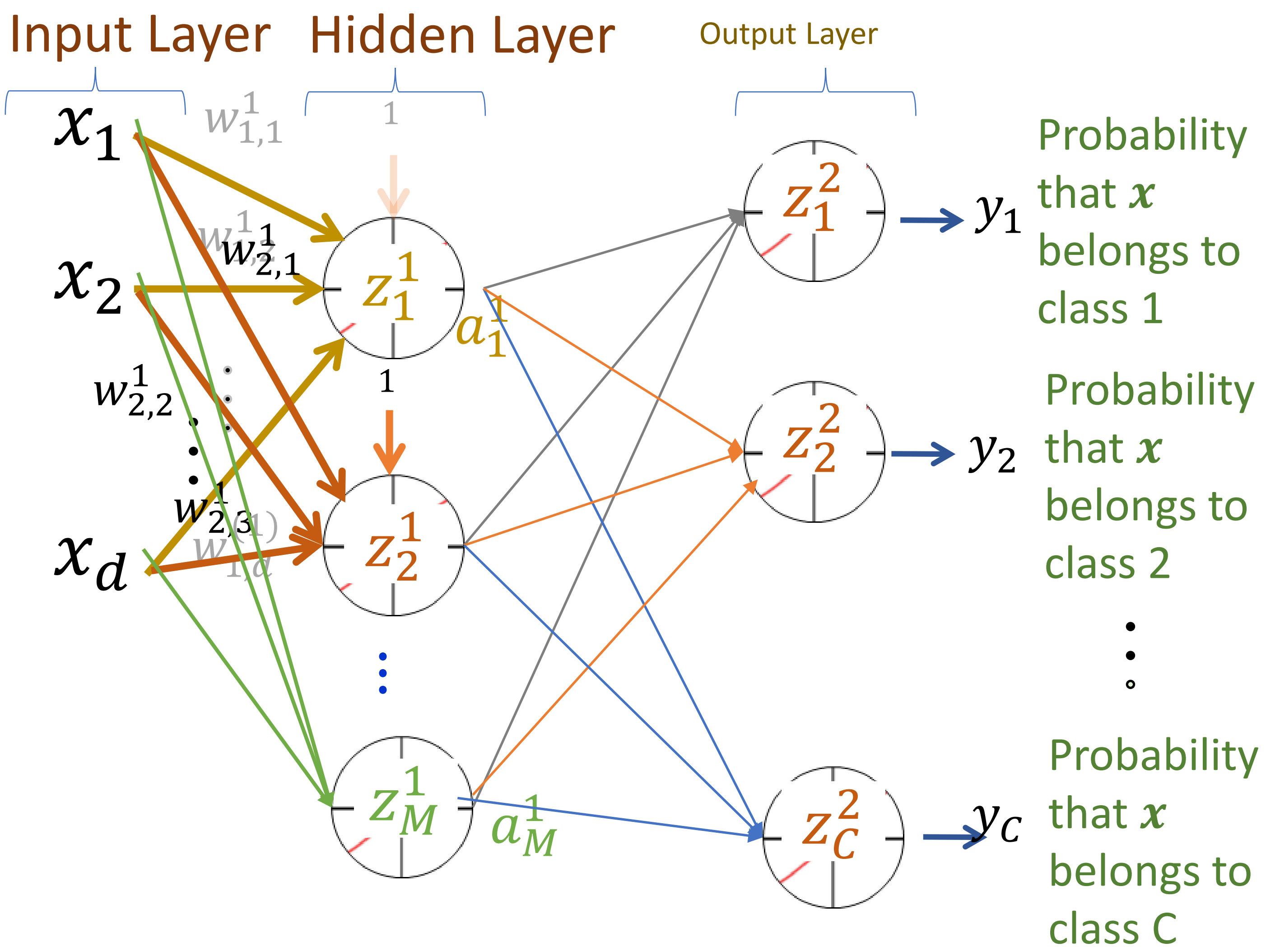


$$\rightarrow y_1 = o_1(z_1^2) = \frac{\exp(z_1^2)}{\sum_c \exp(z_c^2)}$$

$$\rightarrow y_2 = o_2(z_2^2) = \frac{\exp(z_2^2)}{\sum_c \exp(z_c^2)}$$

$$\rightarrow y_C = o_C(z_C^2) = \frac{\exp(z_C^2)}{\sum_c \exp(z_c^2)}$$

Training a Neural Network – Loss Function



Aim to maximize the probability corresponding to the correct class for any example $\boldsymbol{x}$

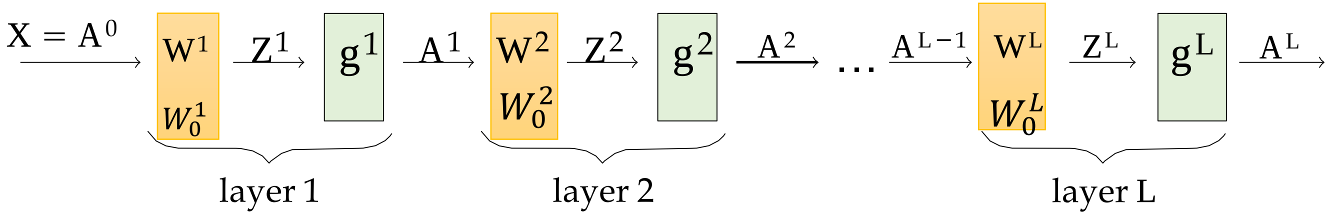
$$\max \boldsymbol{y}_c$$
$$\equiv \max\,(log\, \boldsymbol{y}_c)$$
$$\equiv \min\,(-log\, \boldsymbol{y}_c)$$

Can be equivalently expressed as

$$-\textstyle\sum_i \prod_{i=c} \log(y_i)$$

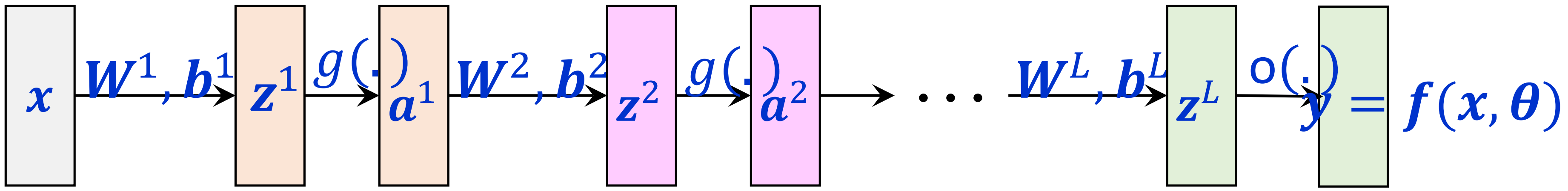
known as <u>cross-entropy loss</u>

Multi layered network



Forward Pass in a Nutshell

$\boldsymbol{\theta}$ is the collection of all learnable parameters i.e., all $\boldsymbol{W}$ and $\boldsymbol{b}$



Hidden layer pre-activation:

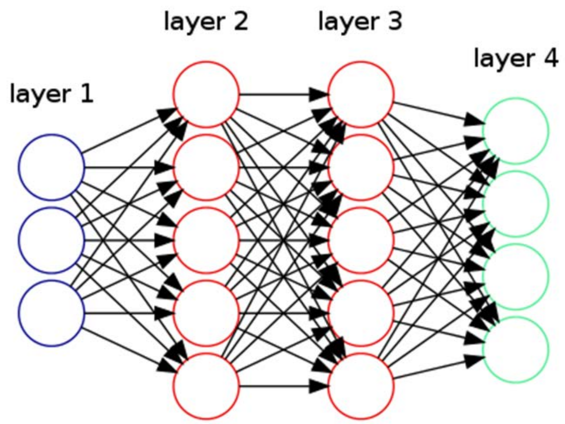
For $l = 1, \ldots, L$; $\boldsymbol{z}^{(l)} = \boldsymbol{W}^{(l)}\boldsymbol{a}^{(l-1)} + \boldsymbol{b}^{(l)}$

Hidden layer activation:

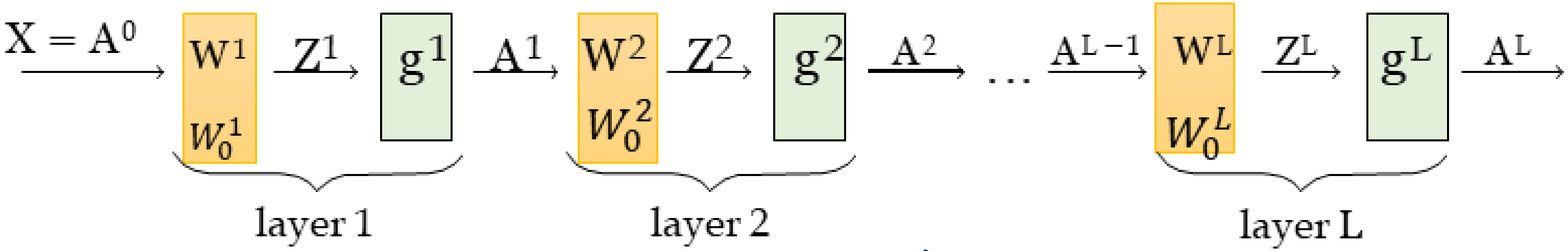
For $l = 1, \ldots, L-1$; $\boldsymbol{a}^{(l)} = g(\boldsymbol{z}^{(l)})$

Output layer activation:

For $l = L$; $\boldsymbol{y} = \boldsymbol{a}^{(L)} = o(\boldsymbol{z}^{(L)}) = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\theta})$



Error back-propagation



- We will train neural networks using gradient descent methods.
- To do SGD for a training example (x, y), we need to compute

$$\nabla_W Loss(NN(x;W),y)$$

where W represents all weights W^l, W_0^l in all the layers $l = (1, \dots, L)$.

$$\frac{\partial Loss}{\partial W^L} = \underbrace{\frac{\partial Loss}{\partial A^L}}_{\text{Depends on Loss function}} \cdot \underbrace{\frac{\partial A^L}{\partial Z^L}}_{g^{l'}} \cdot \underbrace{\frac{\partial Z^L}{\partial W^L}}_{A^{L-1}}$$