Machine Learning

Abdelhak Mahmoudi
abdelhak.mahmoudi@um5.ac.ma

INPT- 2020

Content

1. The Big Picture
2. Supervised Learning
   - Linear Regression, Logistic Regression, Support Vector Machines, Trees, Random Forests, Boosting, Artificial Neural Networks
3. Unsupervised Learning
   - Principal Component Analysis, K-means, Mean Shift

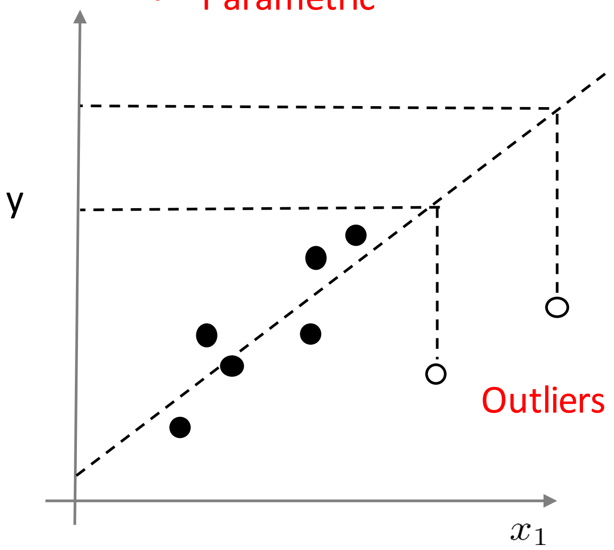
Supervised Learning

- Linear Regression
- Logistic Regression
- Support Vector Machines
- **Trees**
- Random Forests
- Boosting
- Artificial Neural Networks

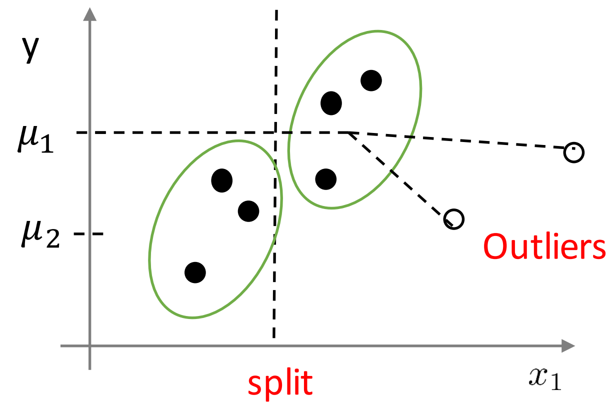
Classification and Regression Trees (CART)

Regression

Linear regression
- Linear models
- Parametric



Regression trees
- Non Linear model
- Non-Parametric

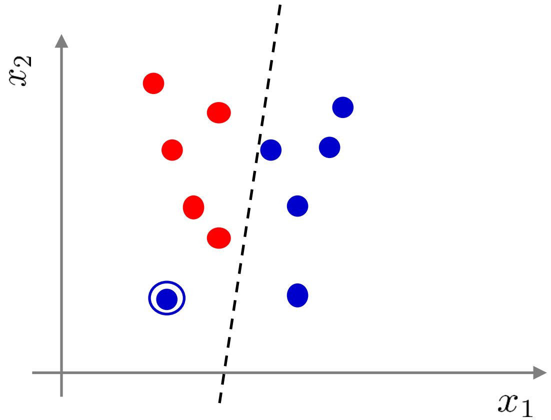


Classification and Regression Trees (CART)

Classification

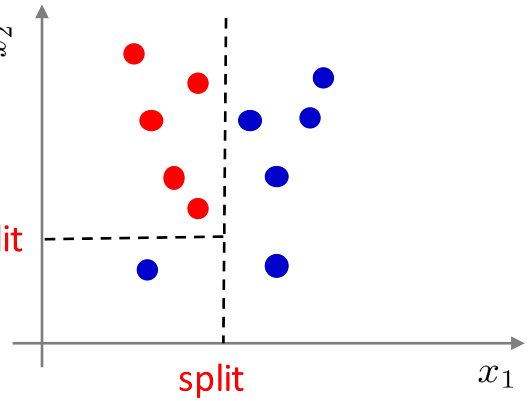
Logistic regression

- Linear models
- Parametric



Classification trees

- Non Linear model
- Non-Parametric



Classification Trees (aka Decision Trees)

Example of Restaurant Data

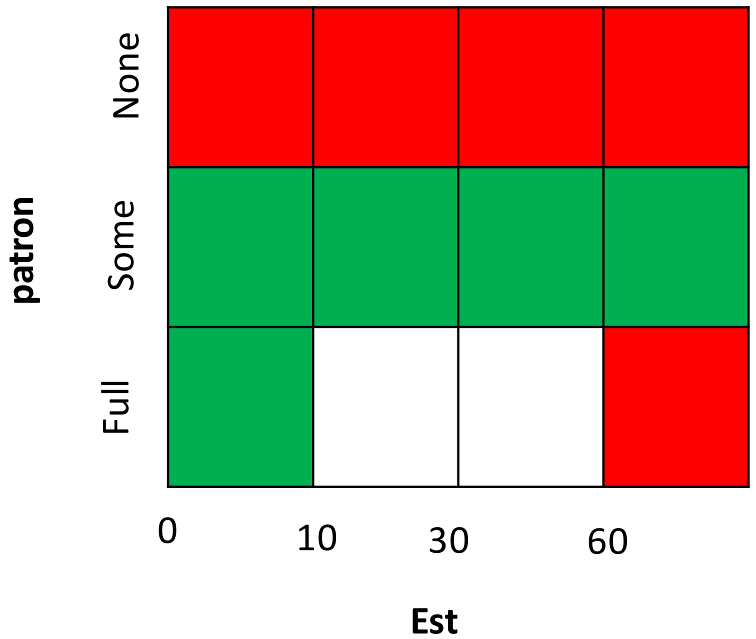
X	F										Y
Client	**Alt**	**Tea**	**Fri**	**Hun**	**Patron**	**Price**	**Rain**	**Res**	**Type**	**Est**	**Wait**
1	T	F	F	T	Some	$$$	F	T	Moroccan	0-10	T
2	T	F	F	T	Full	$	F	F	Chinese	30-60	F
3	F	T	F	F	Some	$	F	F	Burger	0-10	T
4	T	F	T	T	Full	$	F	F	Chinese	10-30	T
5	T	F	T	F	Full	$$$	F	T	Moroccan	>60	F
6	F	T	F	T	Some	$$	T	T	Italian	0-10	T
7	F	T	F	F	None	$	T	F	Burger	0-10	F
8	F	F	F	T	Some	$$	T	T	Chinese	0-10	T
9	F	T	T	F	Full	$	T	F	Burger	>60	F
10	T	T	T	T	Full	$$$	F	T	Italian	10-30	F
11	F	F	F	F	None	$	F	F	Chinese	0-10	F
12	T	T	T	T	Full	$	F	F	Burger	30-60	T

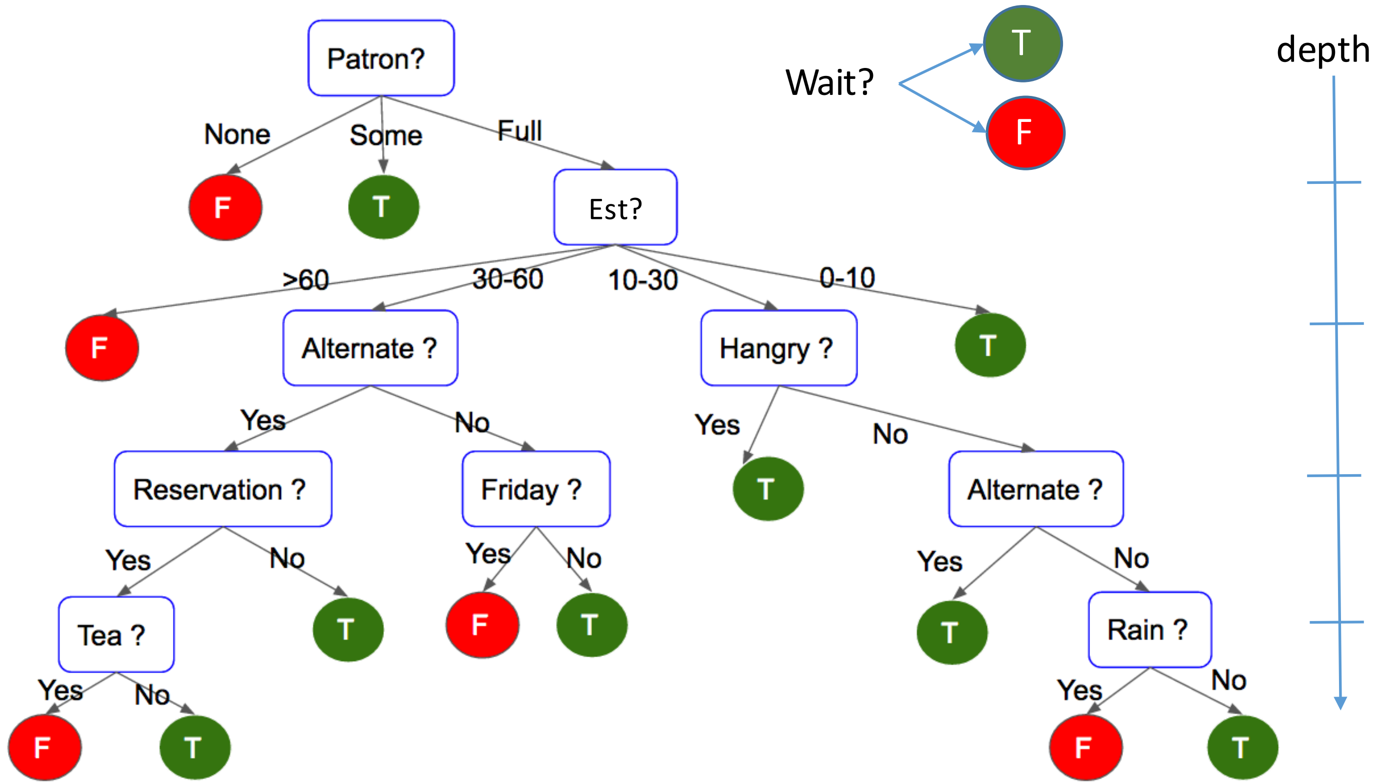
- Alt: is there any other alternative?
- Fri: is it Friday?
- Hun: is the client hungry?
- Patron: how many people are in the restaurant?
- Res: Restaurant
- Est: wait estimate

Most of the features are
Discrete (=categorical, =qualitative)

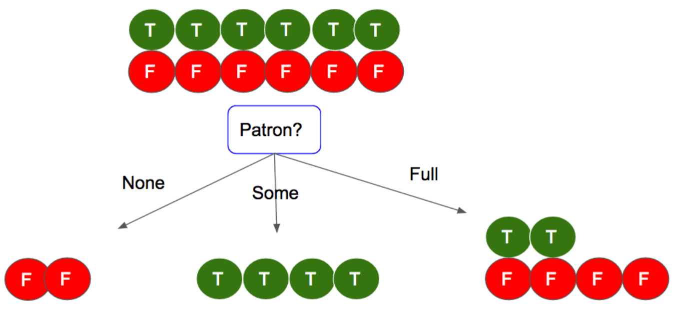
Classification Trees (aka Decision Trees)

Pick one feature at a time, and make a binary decision

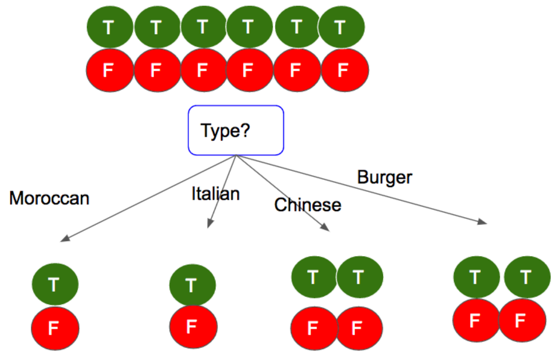




DT: Which feature to split with first?



More informative
Less impurity



Less informative
More impurity

Split with the feature F that maximizes the Information Gain

$$\underset{\text{Information Gain}}{I(F)} = \underset{\text{Parent entropy}}{H(S)} - \underset{\text{Expected entropy}}{EH(F)}$$

S: subset = {p positives and n negatives}
F: Feature

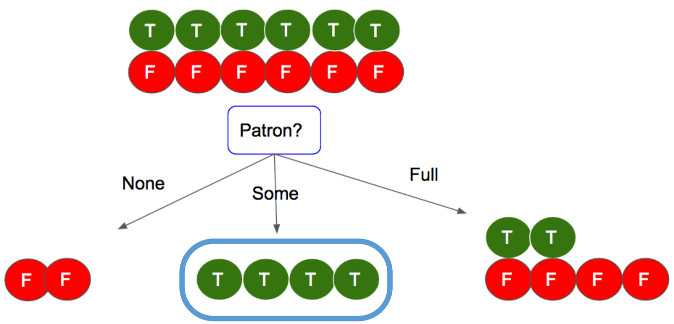
Entropy

$$H(S) = -\sum_{c} p_c \log_2(p_c)$$

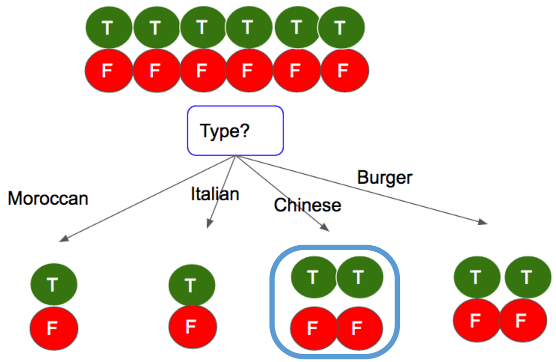
Python: np.log2()

p_c : probability of examples in class c
S : subset of data examples

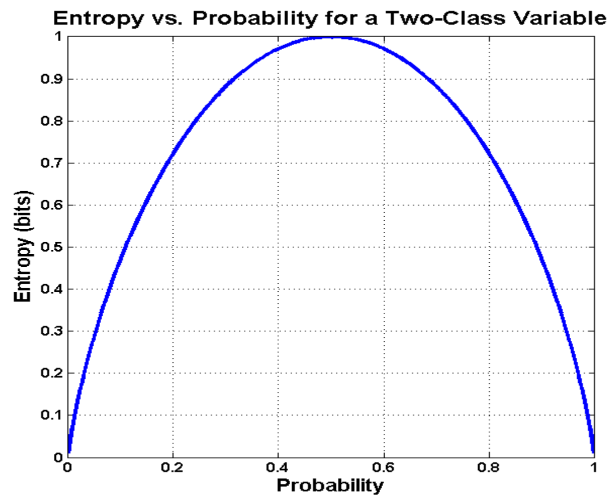
Interpretation: Measure of the impurity in a subset of examples



All examples in the same class
Entropy = 0



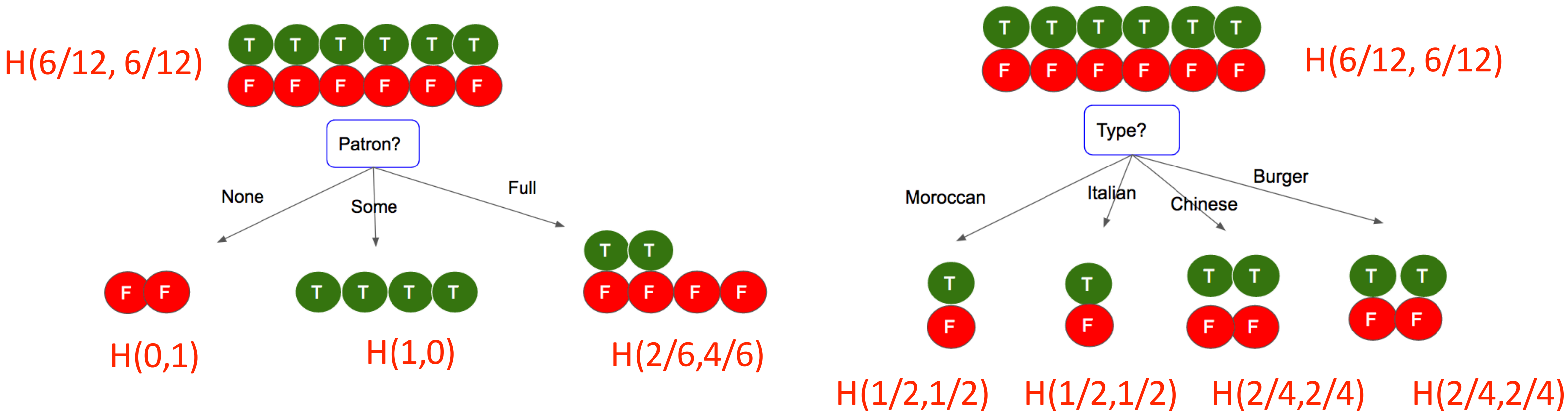
All examples evenly split between classes
Entropy = 1



Entropy (binary classification)

$$H\left(\frac{p}{p+n},\frac{n}{p+n}\right) = -\frac{p}{p+n}\log(\frac{p}{p+n}) - \frac{n}{p+n}\log(\frac{n}{p+n})$$

p: positive
n: negative



Expected Entropy

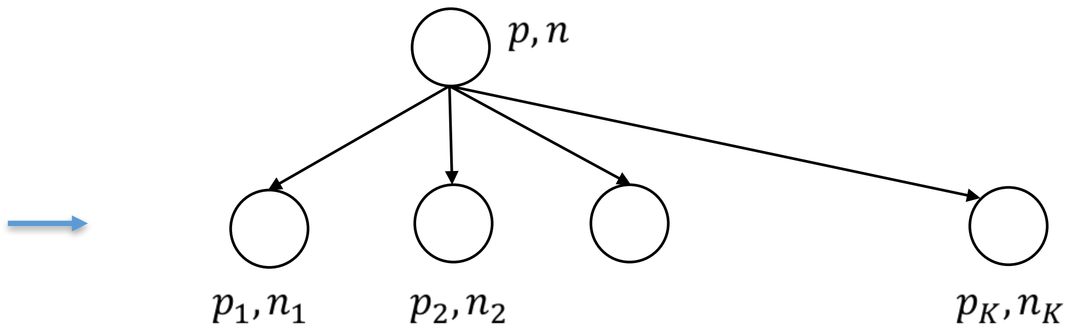
$$EH(F) = \sum_{i=1}^{K} \frac{p_i + n_i}{p + n} H\left(\frac{p_i}{p_i + n_i}, \frac{n_i}{p_i + n_i}\right)$$

K = number of splits (regions) with Feature F

= number of children nodes

Expectation Entropy = weighted average of children entropy

children

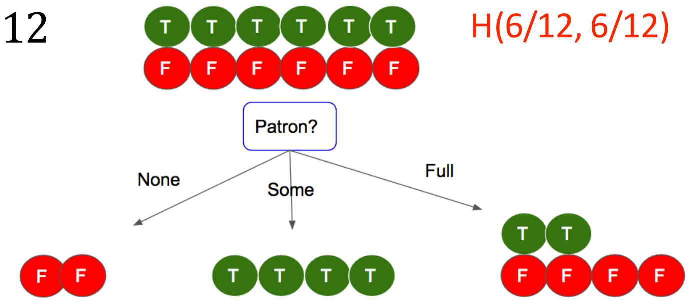


Expected Entropy (Example)

$$EH(F) = \sum_{i=1}^{K} \frac{p_i + n_i}{p + n} H\left(\frac{p_i}{p_i + n_i}, \frac{n_i}{p_i + n_i}\right)$$

$p + n = 12$

H(6/12, 6/12)



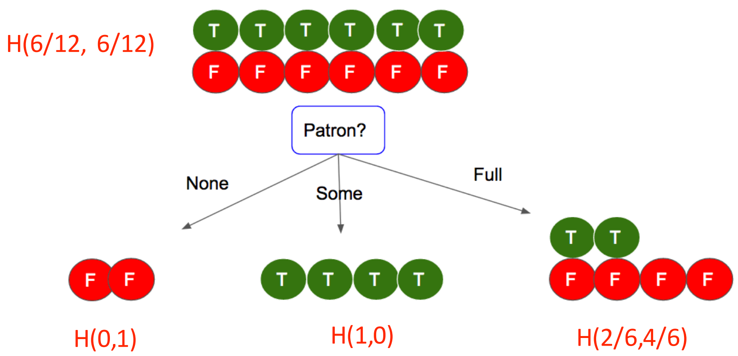
$K = 3$

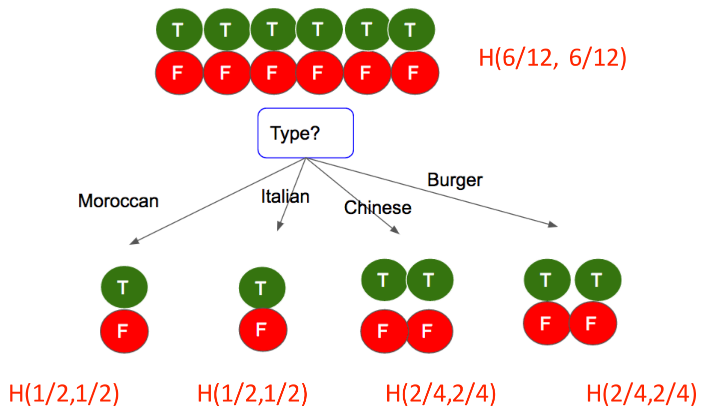
H(0,1) = 0   H(1,0) = 0   H(2/6,4/6) = .92

$p_i + n_i$ 2   4   6

Information Gain

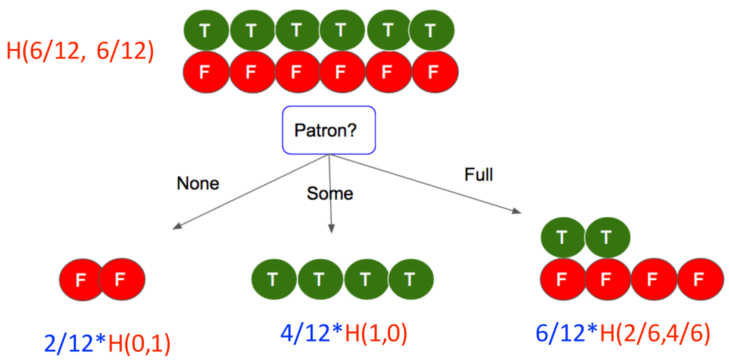
$$I(F) = H\left(\frac{p}{p+n}, \frac{n}{p+n}\right) - EH(F)$$

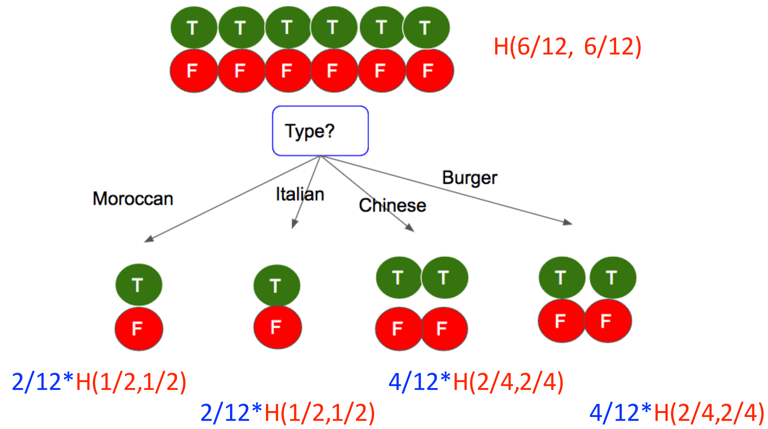




Information Gain

$$I(F) = H\left(\frac{p}{p+n}, \frac{n}{p+n}\right) - EH(F)$$





I(Patron) = 0.54

I(Type) = 0.0

Other impurity measures

- p_c: probability of examples in class c
- S : subset of data examples

- CART algorithm uses the Entropy

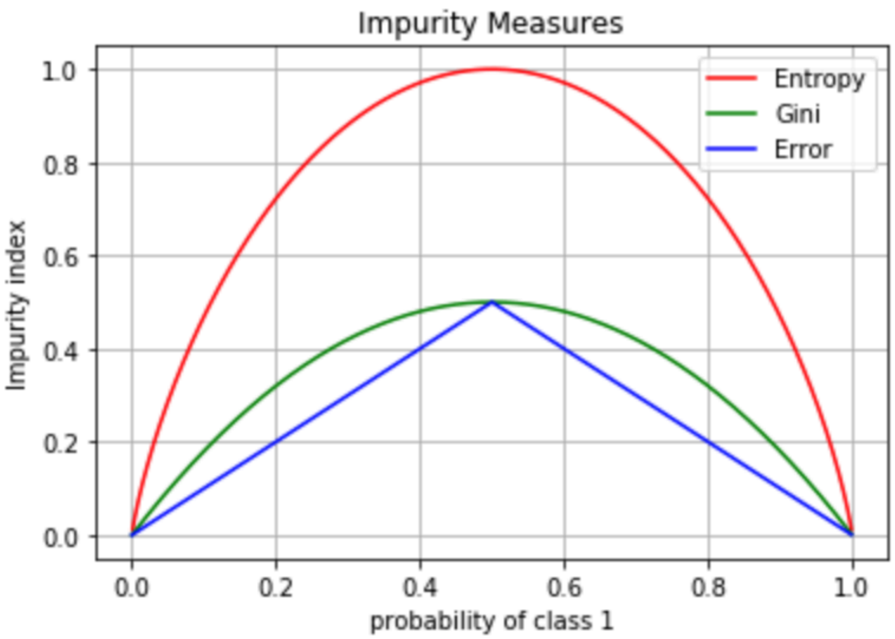
$$H(S) = -\sum_c p_c \log_2(p_c)$$

- Iterative Dichotomiser ID3 and C4.5 algorithms use Gini Index

$$G(S) = 1 - \sum_c p_c^2$$

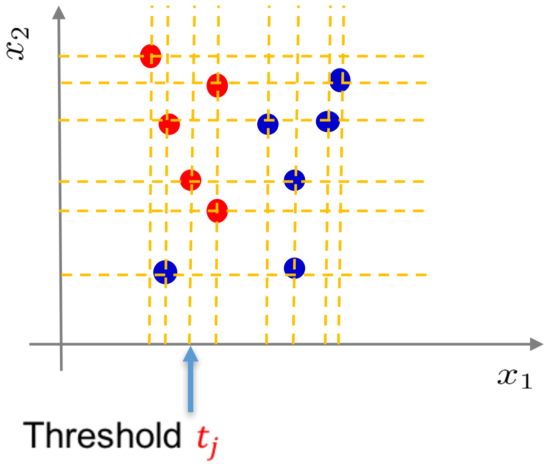
- One could also use the Class Error

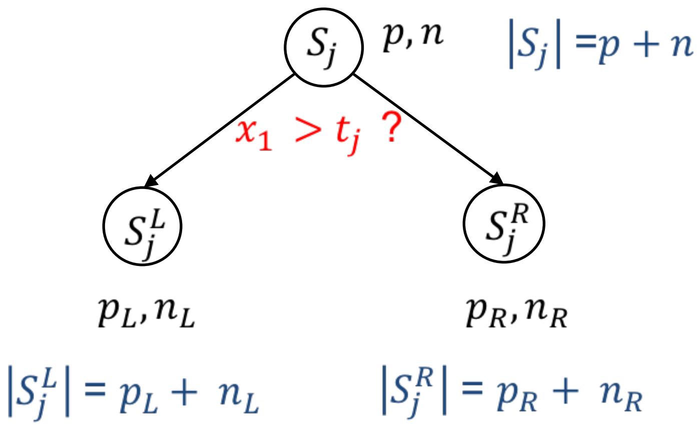
$$E(S) = 1 - \max_c(p_c)$$



What if a feature is continuous (quantitative)?

$$I_j = H(S_j) - \sum_{i \epsilon \{L,R\}} \frac{|S_j^i|}{|S_j|} H(S_j^i)$$





Note: Doing so, trees are almost Binary! Even if a feature is categorical (qualitative)

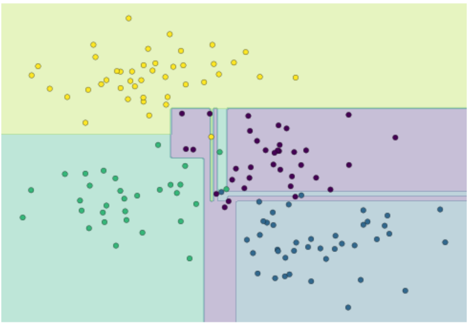
Decision Trees

- **Advantages**
  - Easy to interpret
  - Deals with non linearity
  - Handle qualitative features without the need to create fictive ones (one hot vector)
  - Provide most important features (in terms of information gain)

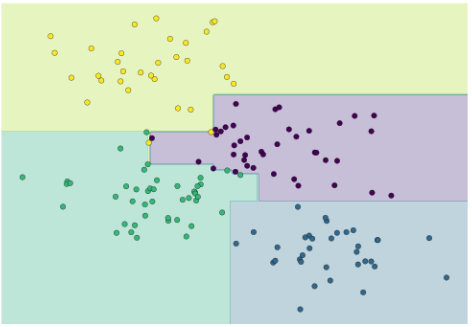
Decision Trees

- **Disadvantages**
  - Trees leads to overfitting (high variance): little change in little number of examples affect the whole tree.

DT on Data1



DT on Data2 = half of Data1

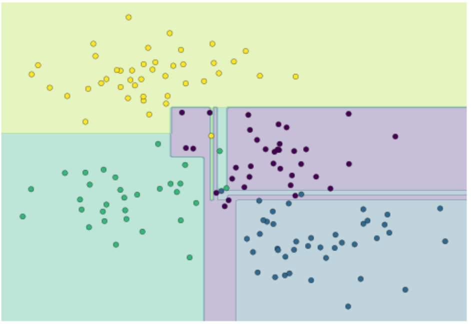


Decision Trees

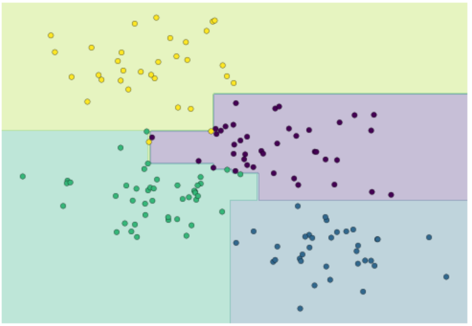
- **Disadvantages**
  - Trees leads to overfitting (high variance): little change in little number of examples affect the whole tree.

- **Solution: Ensemble Methods**
  - Bagging: Random Forest (Leo Breiman 2001) consists of combining multiple independent weak trees to reduce variance.
  - Boosting to reduce bias

DT on Data1



DT on Data2 = half of Data1

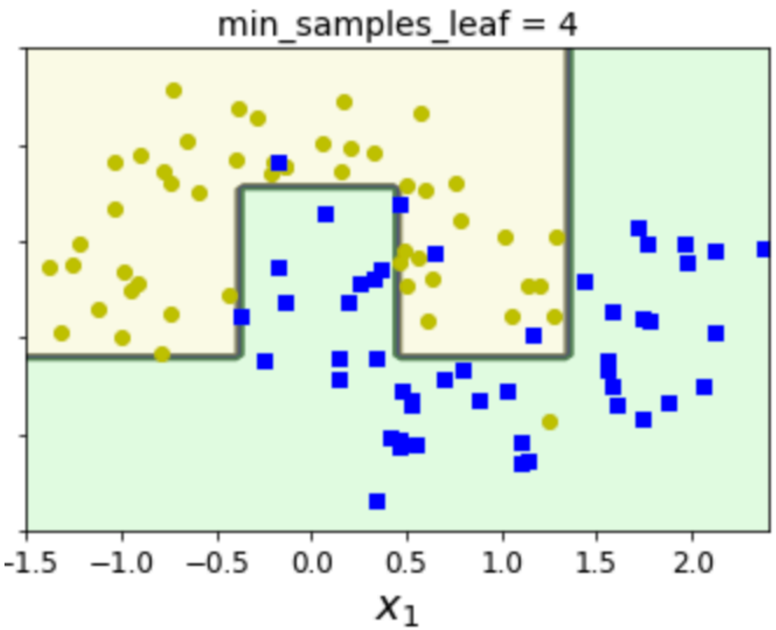


A tree alone will overfit.
However, it is clear that in some places, the two trees together produce consistent results

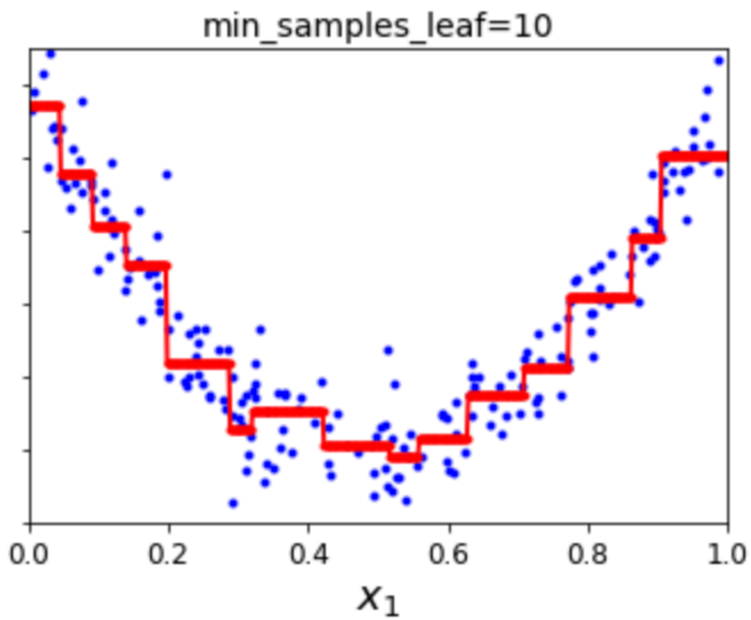
This idea comes from Bootstrapping (Brad Efron 1979):
Given a set of m independent observations o_1.., o_m, each with variance σ^2, the variance of the mean o of the observations is given by σ^2/m.

Classification And Regression Trees

Classification (Decision) Trees



Regression Trees



Supervised Learning

- Linear Regression
- Logistic Regression
- Support Vector Machines
- Trees (Decision and Regression)
- **Random Forests**
- Boosting
- Artificial Neural Networks

Random Forest

Bagging

Bootstrap Aggregating

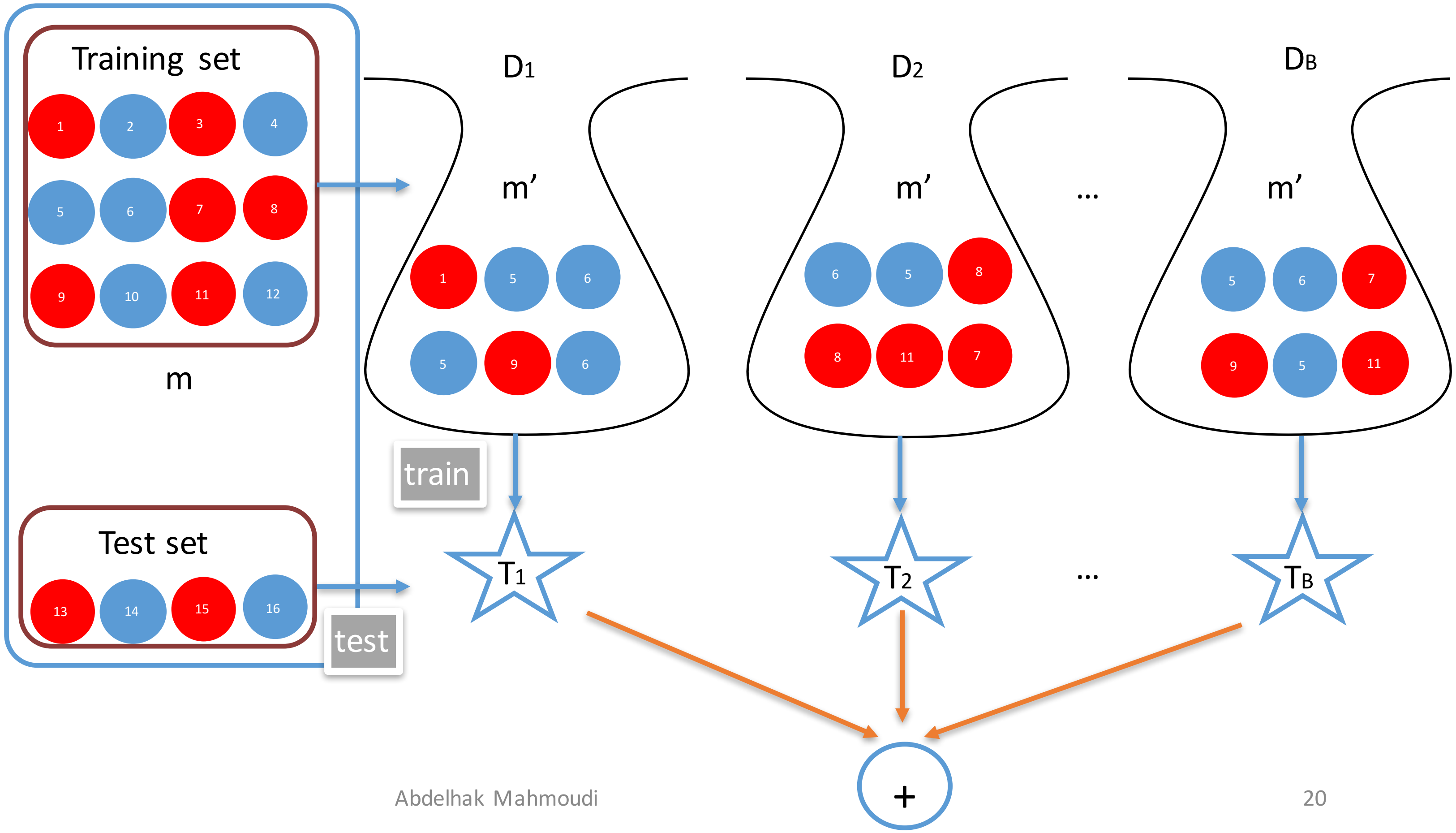
Training

Pick m’ examples with replacement and train B trees

Testing

Regression: mean errors of all the B trees

Classification: vote

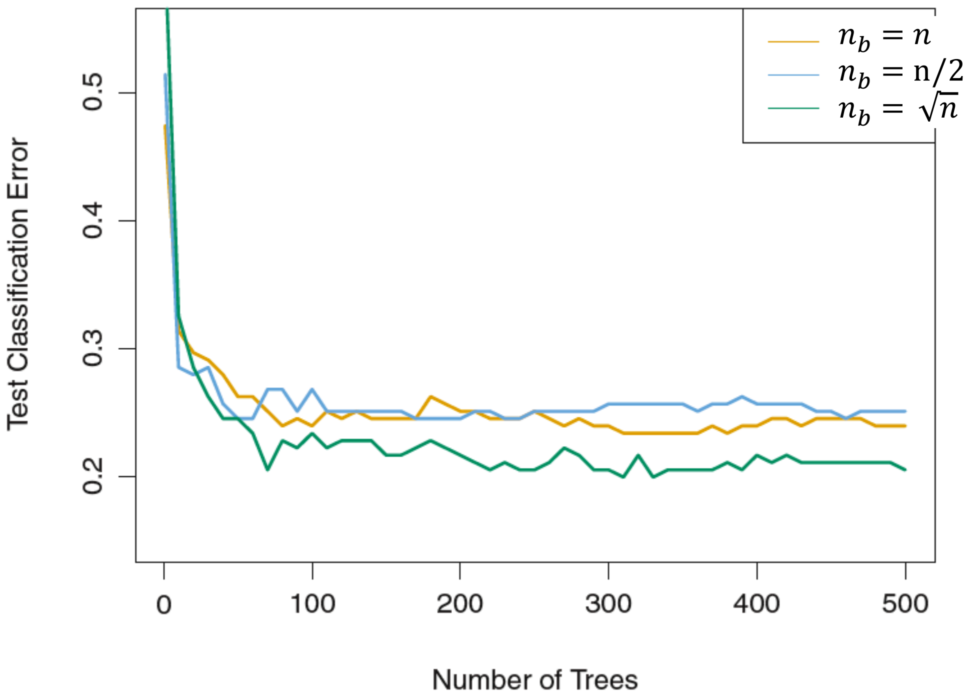


Random Forest

- **Problem:** Bagged trees will look quite similar to each other, so averaging them will not led to much reduction of variance!

- **Solution:** Random Forest constructs multiple trees where each tree uses n_b random features from the the n initial features (generally $n_b = \sqrt{n}$)

- $n_b = n$-> Bagging case



Random Forest

- Both training and prediction are very fast, because of the simplicity of the underlying decision trees.
- Tasks can be straightforwardly parallelized, because the individual trees are entirely independent entities.
- The multiple trees allow for a probabilistic classification: a majority vote among estimators gives an estimate of the probability
- RF is a Nonparametric model, extremely flexible, and can thus perform well on tasks that are under-fit by other models.

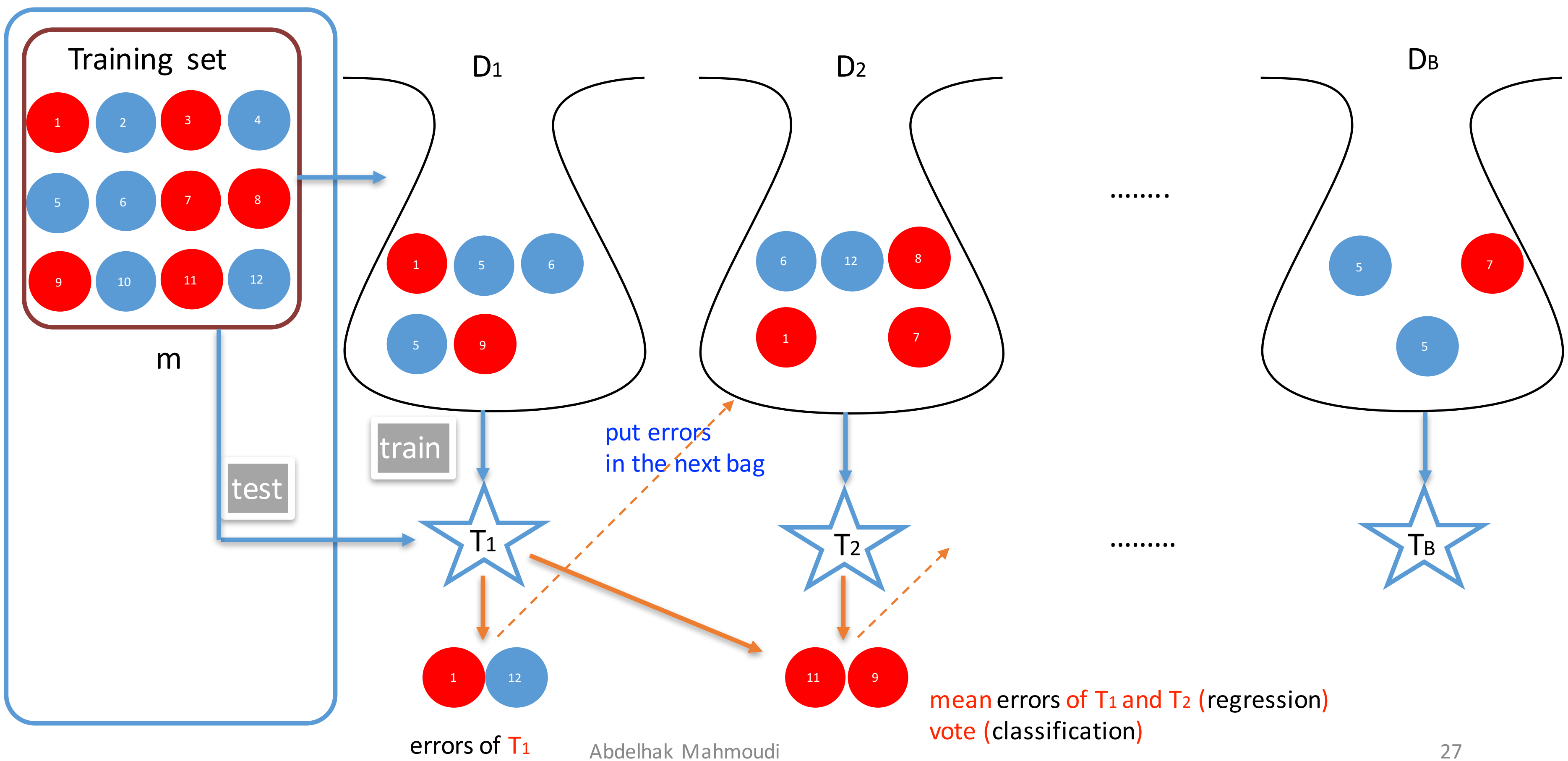
Random Forest

- Hyper-Parameters Tuning
  - d: Depth of the trees
  - B: number of Bags

Supervised Learning

- Linear Regression
- Logistic Regression
- Support Vector Machines
- Trees (Decision and Regression)
- Random Forests
- **Boosting**
- Artificial Neural Networks

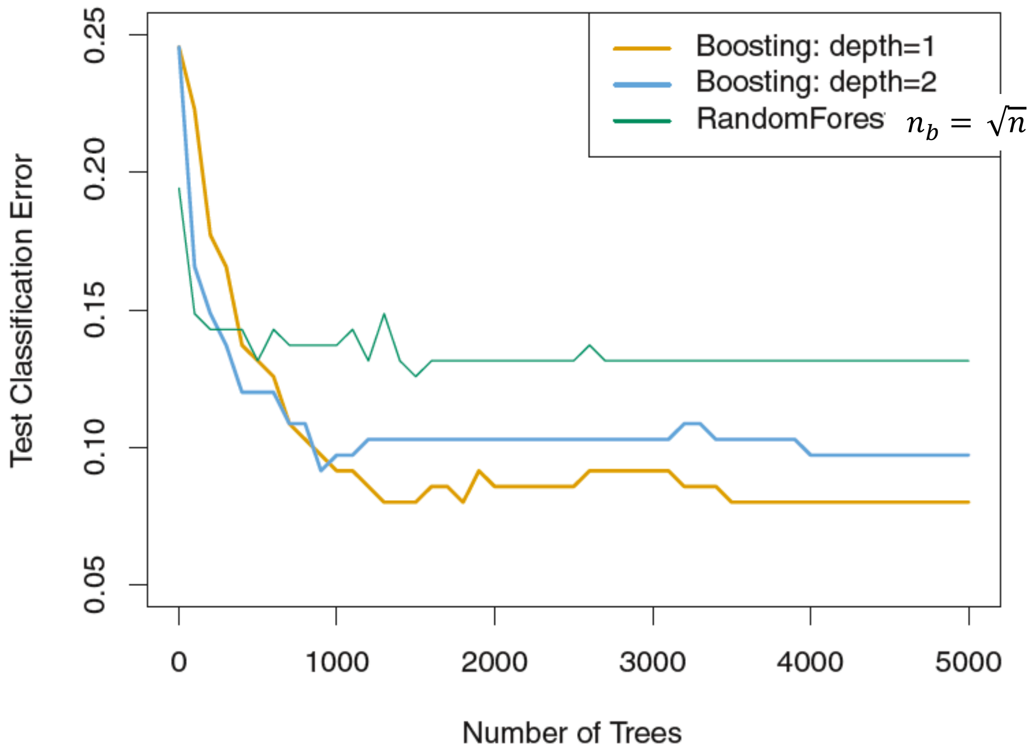
Boosting



Training set
1 2 3 4
5 6 7 8
9 10 11 12
m
test
train
D1
D2
DB
1 5 6
5 9
6 12 8
1 7
5 7
5
....
T1
T2
TB
....
put errors
in the next bag
1 12
11 9
errors of T1
mean errors of T1 and T2 (regression)
vote (classification)

Boosting

- Outperforms RF
- Smaller Trees (depth = 1) are sufficient because the growth of a particular tree takes into account preceding trees.
- Smaller trees can aid in interpretability.
- Boosting (Freund & Schapire 1990)
- Adaboost (Adaptive Boosting), 1996

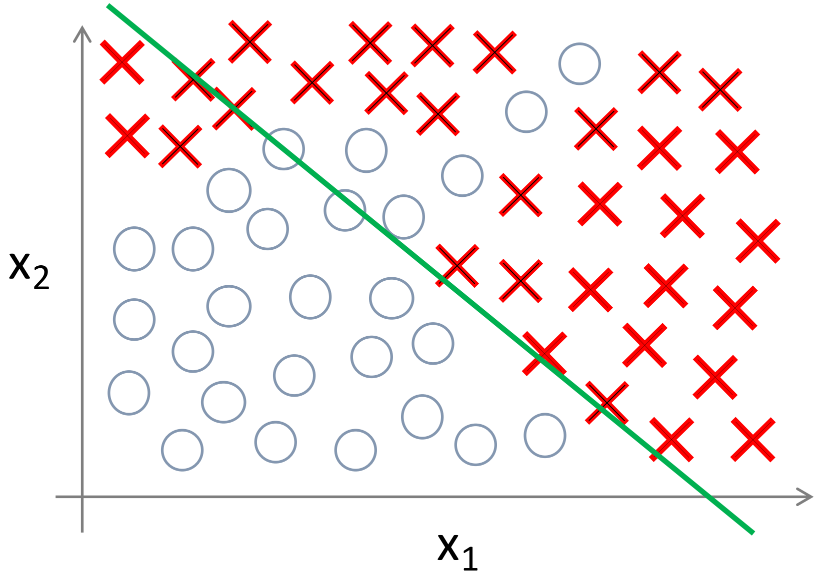


Supervised Learning

- Linear Regression
- Logistic Regression
- Support Vector Machines
- Trees (Decision and Regression)
- Random Forests
- Boosting
- **Artificial Neural Networks**

Non Linear Classification

A linear model will certainly underfit !

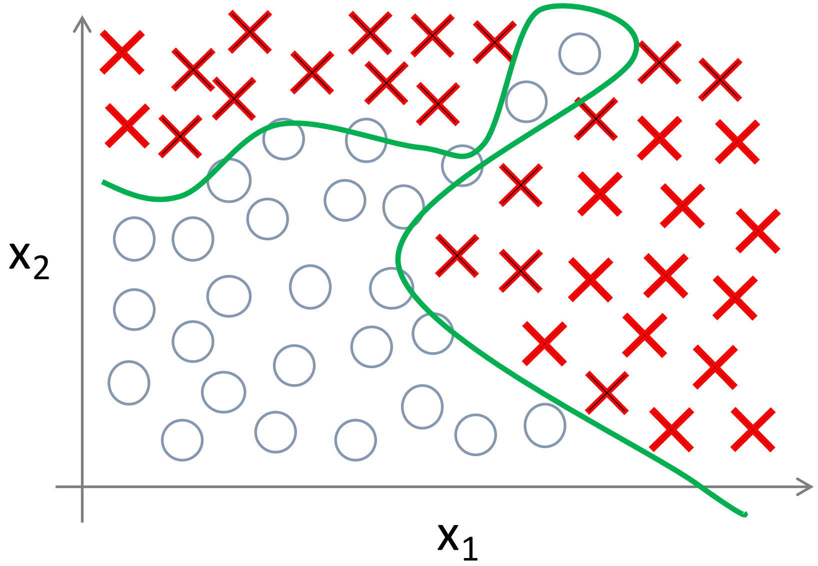


$$g(w_0 + w_1 x_1 + w_2 x_2)$$

Non Linear Classification

A non linear model in a high dimensional space may be a solution, but…

… what if we have many many features?!



$$g(w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_{12} x_1^2 + w_{13} x_2^2 \ldots + w_{25} x_1^d + w_{26} x_2^d)$$

Non Linear Classification: Computer Vision

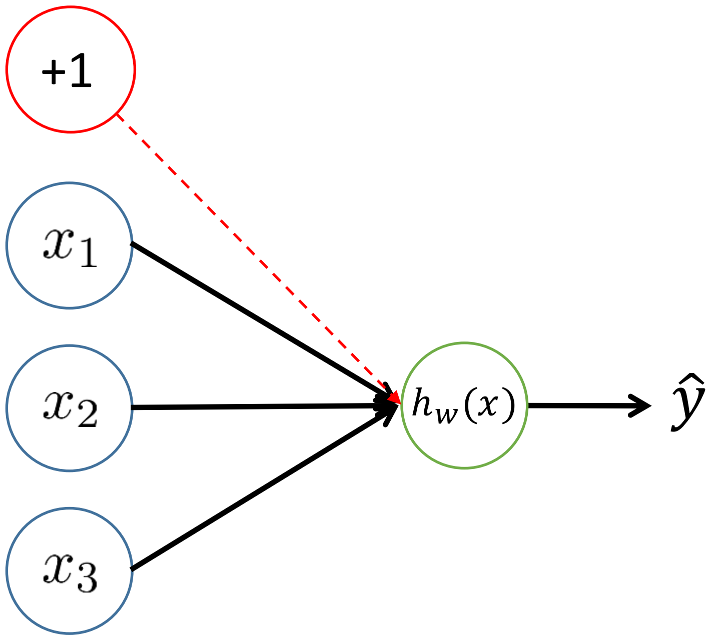
A non linear model in a high dimensional space may be a solution, but…

… what if we have many many features?!

… like in computer vision

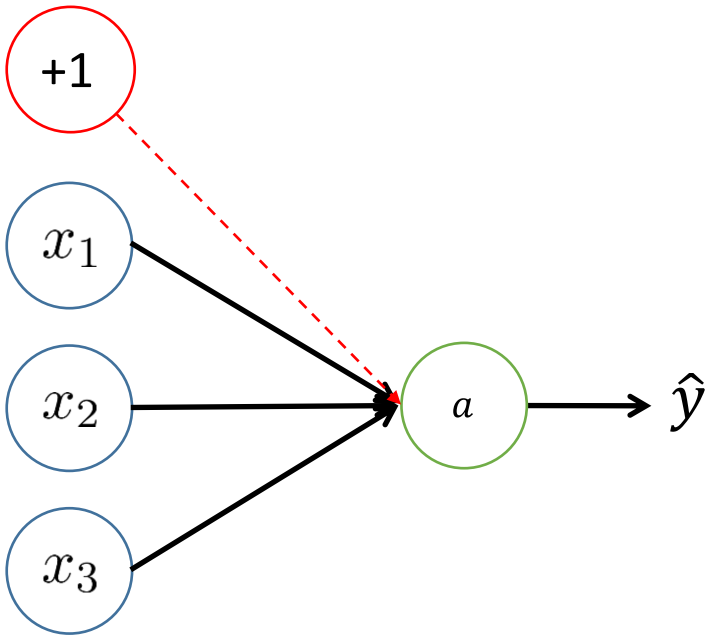
194	210	201	212	199	213	215	195	178	158	182	209
180	189	190	221	209	205	191	167	147	115	129	163
114	126	140	188	176	165	152	140	170	106	78	88
87	103	115	154	143	142	149	153	173	101	57	57
102	112	106	131	122	138	152	147	128	84	58	66
94	95	79	104	105	124	129	113	107	87	69	67
68	71	69	98	89	92	98	95	89	88	76	67
41	56	68	99	63	45	60	82	58	76	75	65
20	43	69	75	56	41	51	73	55	70	63	44
50	50	57	69	75	75	73	74	53	68	59	37
72	59	53	66	84	92	84	74	57	72	63	42
67	61	58	65	75	78	76	73	59	75	69	50

ANN Model Representation



$$x = \begin{bmatrix} \overset{=1}{x_0} \\ x_1 \\ x_2 \\ x_3 \\ \vdots \end{bmatrix} \quad w = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \\ \vdots \end{bmatrix} \qquad \hat{y} = h_w(x) = g(w^T x)$$

ANN Model Representation



$$x = \begin{bmatrix} \overset{=1}{x_0} \\ x_1 \\ x_2 \\ x_3 \\ \vdots \end{bmatrix} \qquad w = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \\ \vdots \end{bmatrix}$$

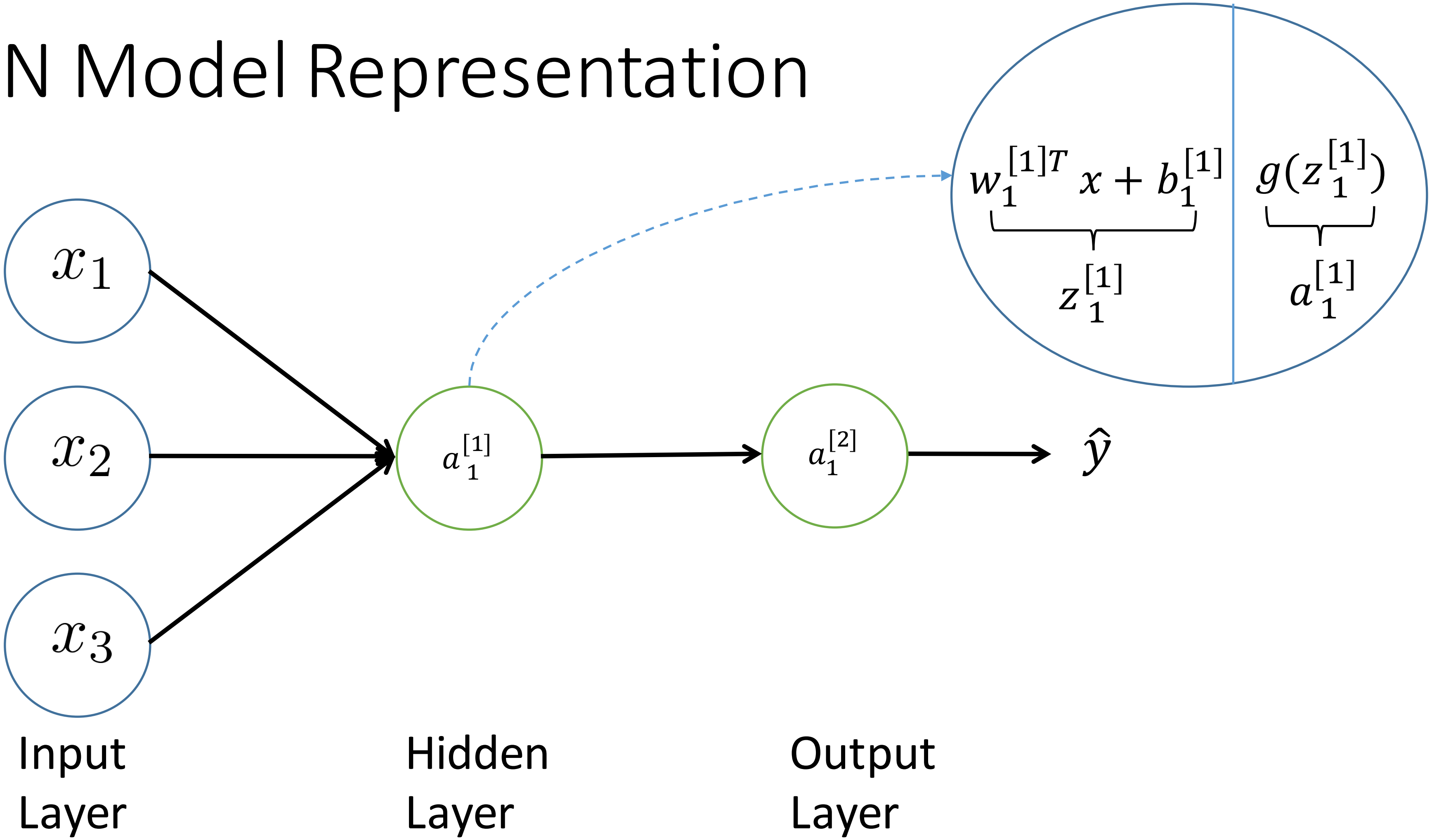
$$\hat{y} = h_w(x) = g(w^T x)$$

Change notation

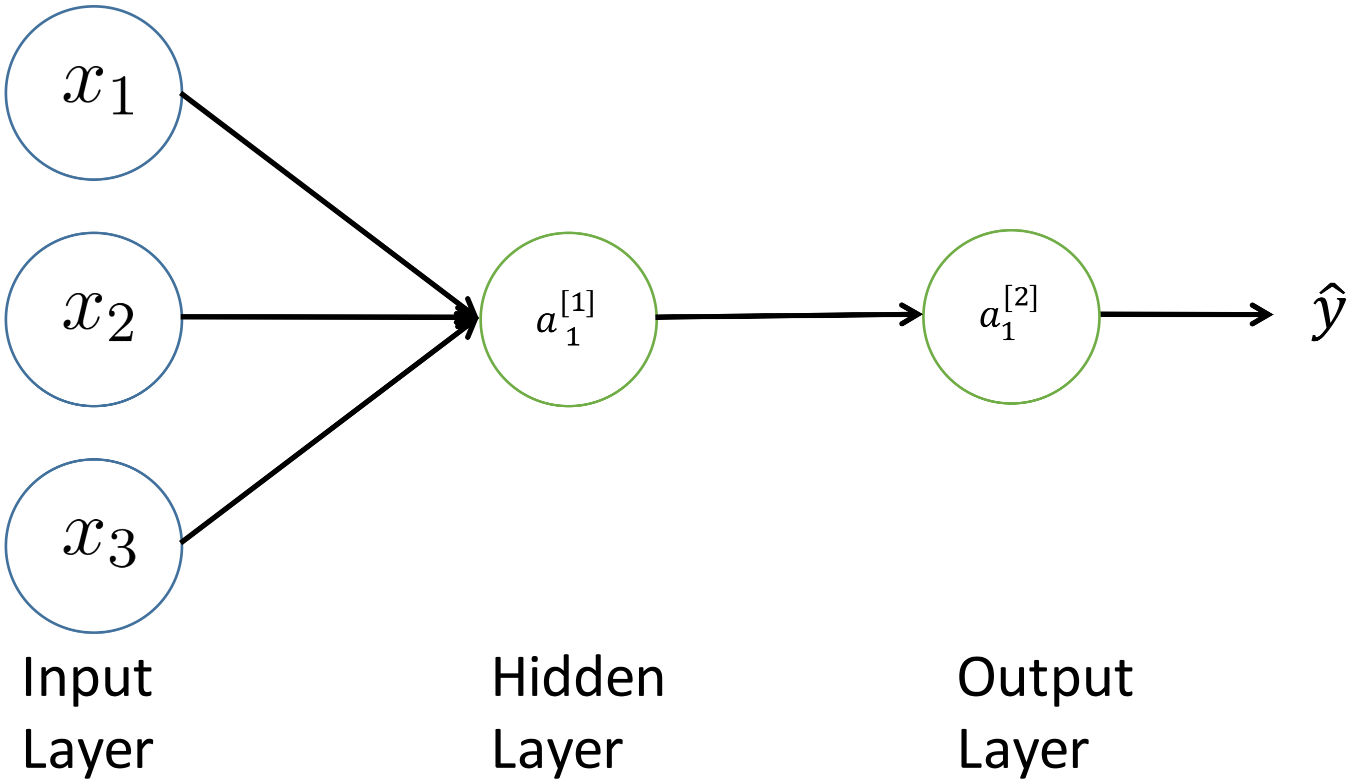
$$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \end{bmatrix} \qquad w = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ \vdots \end{bmatrix}$$

$$z = w^T x + \overset{\boxed{= w_0}}{b},$$
$$\hat{y} = a = g(z)$$

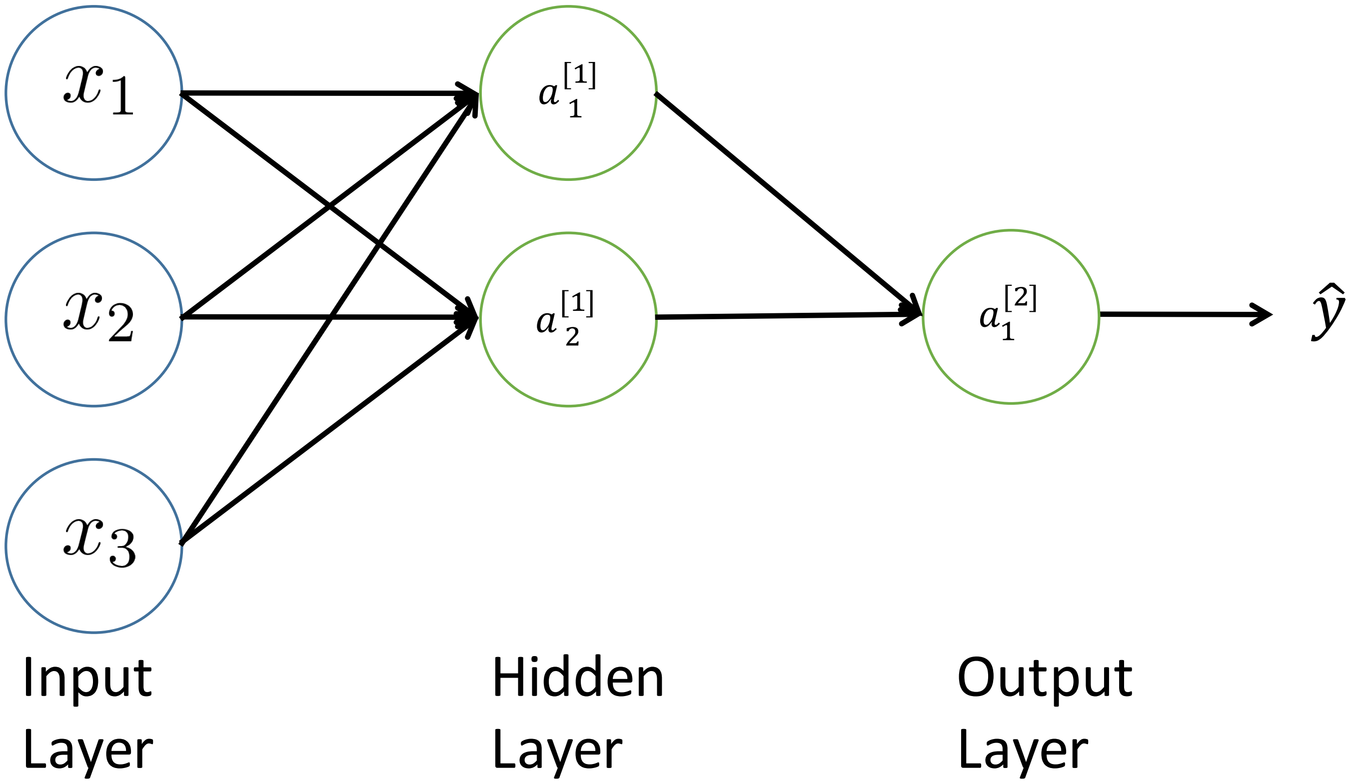
ANN Model Representation



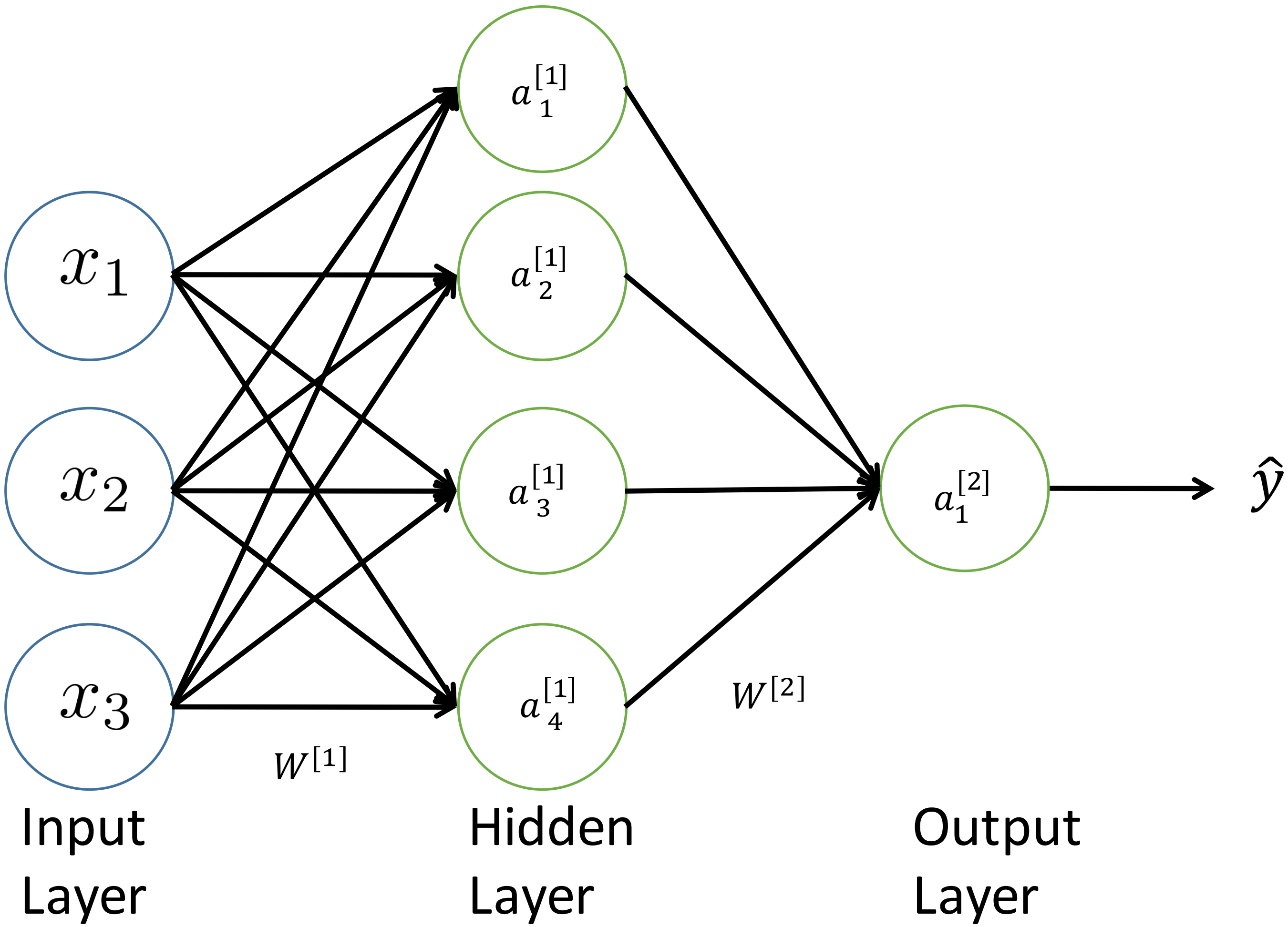
ANN Model Representation



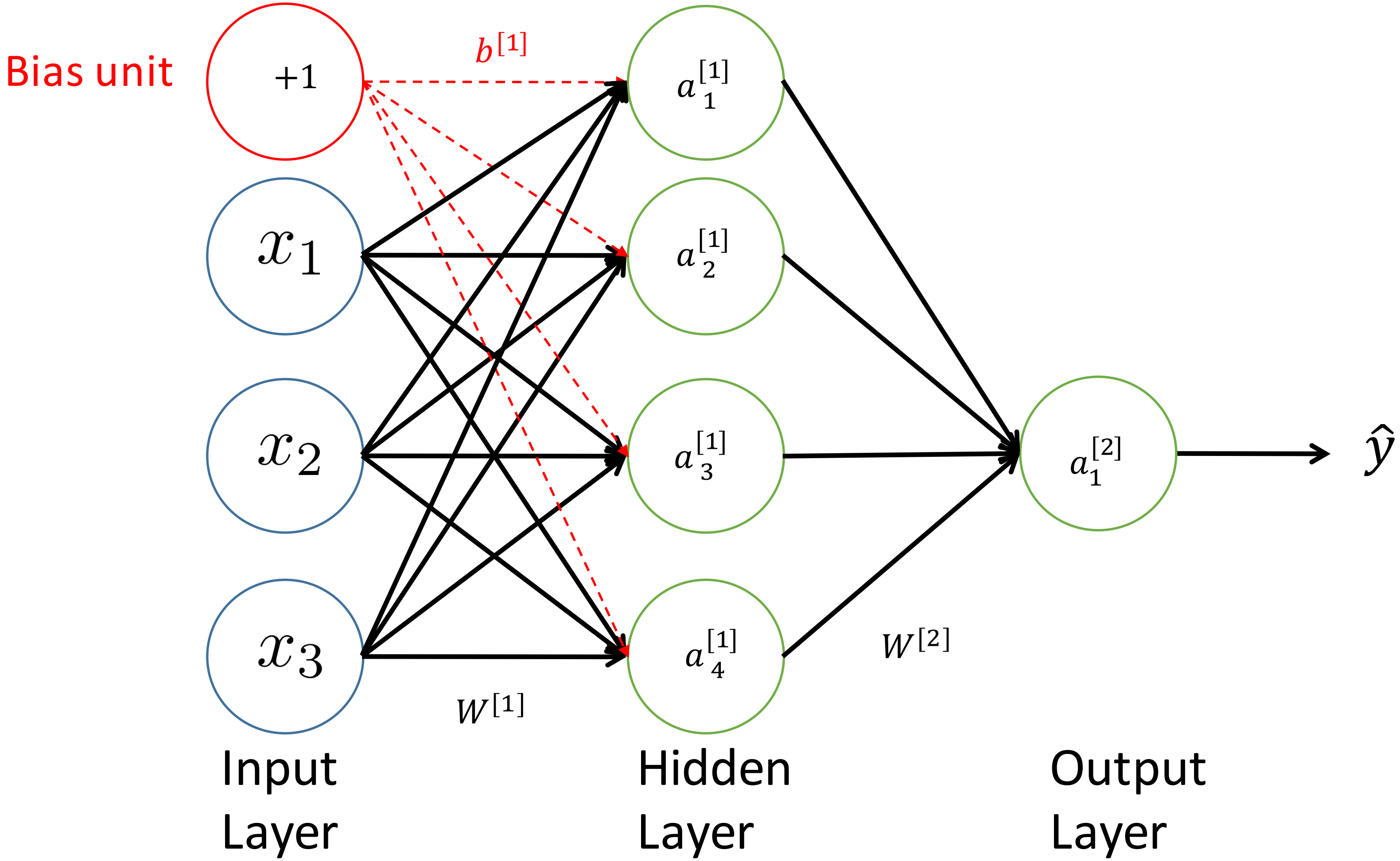
ANN Model Representation



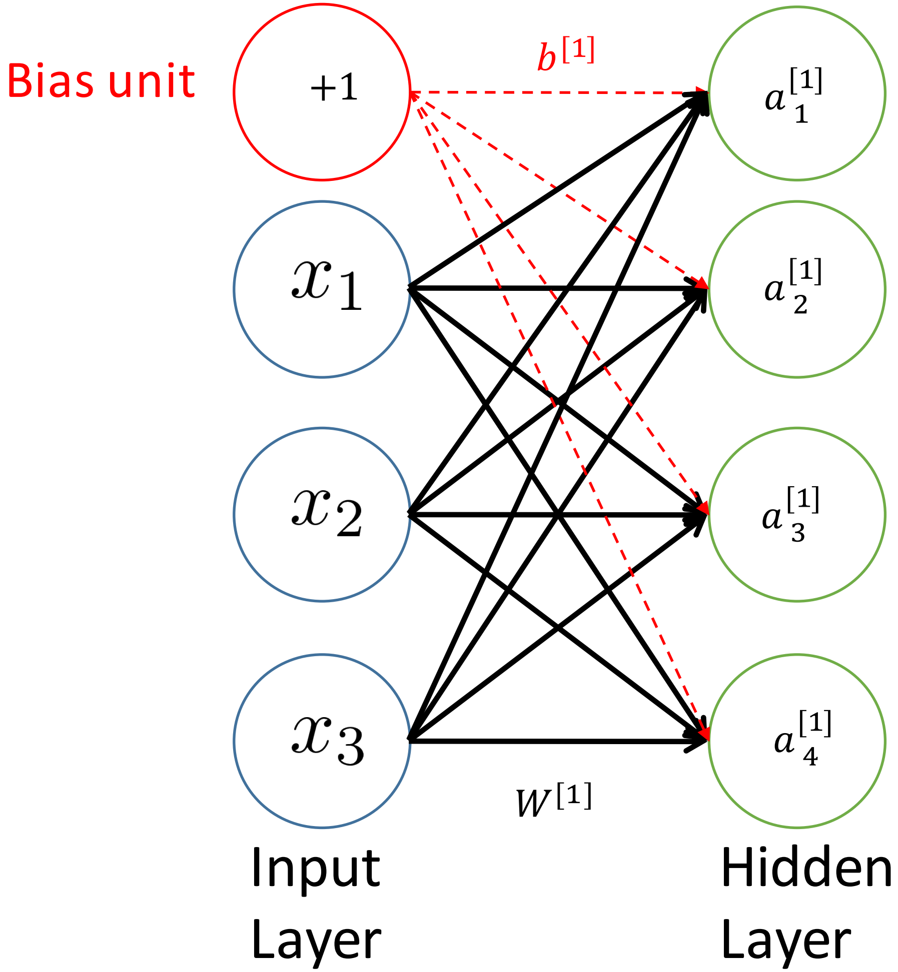
ANN Model Representation



ANN Model Representation

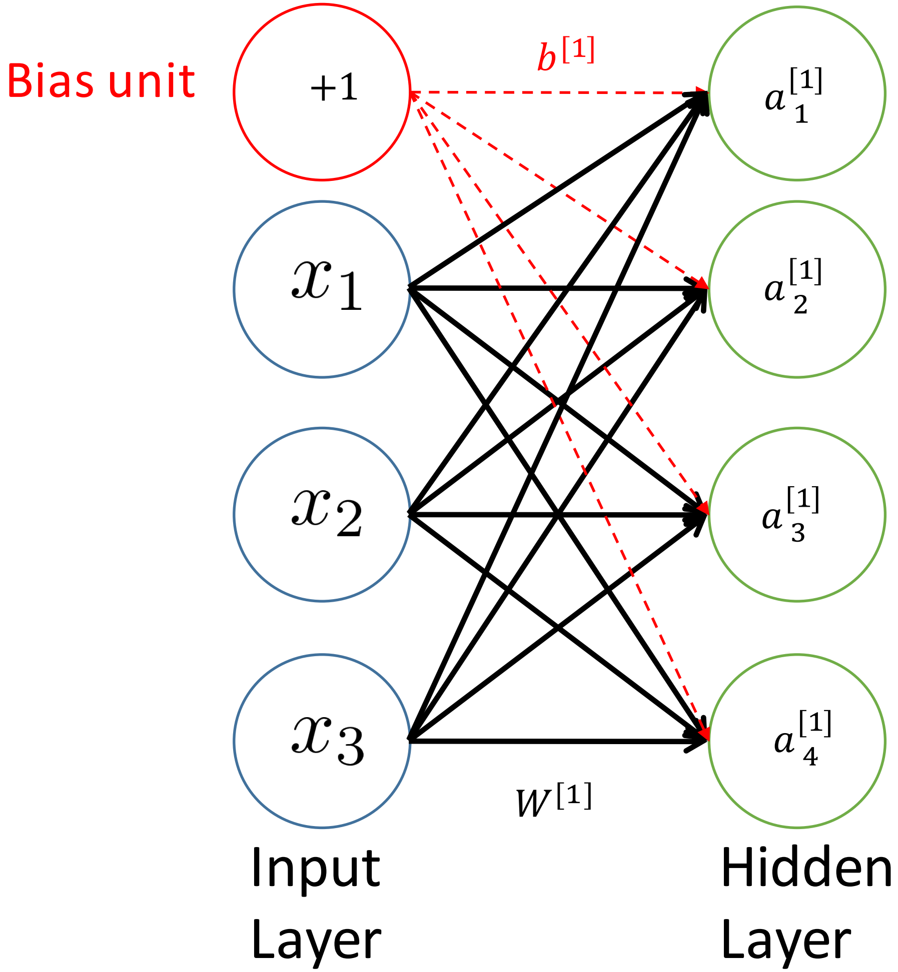


ANN Model Representation



$$z_1^{[1]} = w_1^{[1]T} x + b_1^{[1]},\ a_1^{[1]} = g(z_1^{[1]})$$
$$z_2^{[1]} = w_2^{[1]T} x + b_2^{[1]},\ a_2^{[1]} = g(z_2^{[1]})$$
$$z_3^{[1]} = w_3^{[1]T} x + b_3^{[1]},\ a_3^{[1]} = g(z_3^{[1]})$$
$$z_4^{[1]} = w_4^{[1]T} x + b_4^{[1]},\ a_4^{[1]} = g(z_4^{[1]})$$

ANN Model Representation

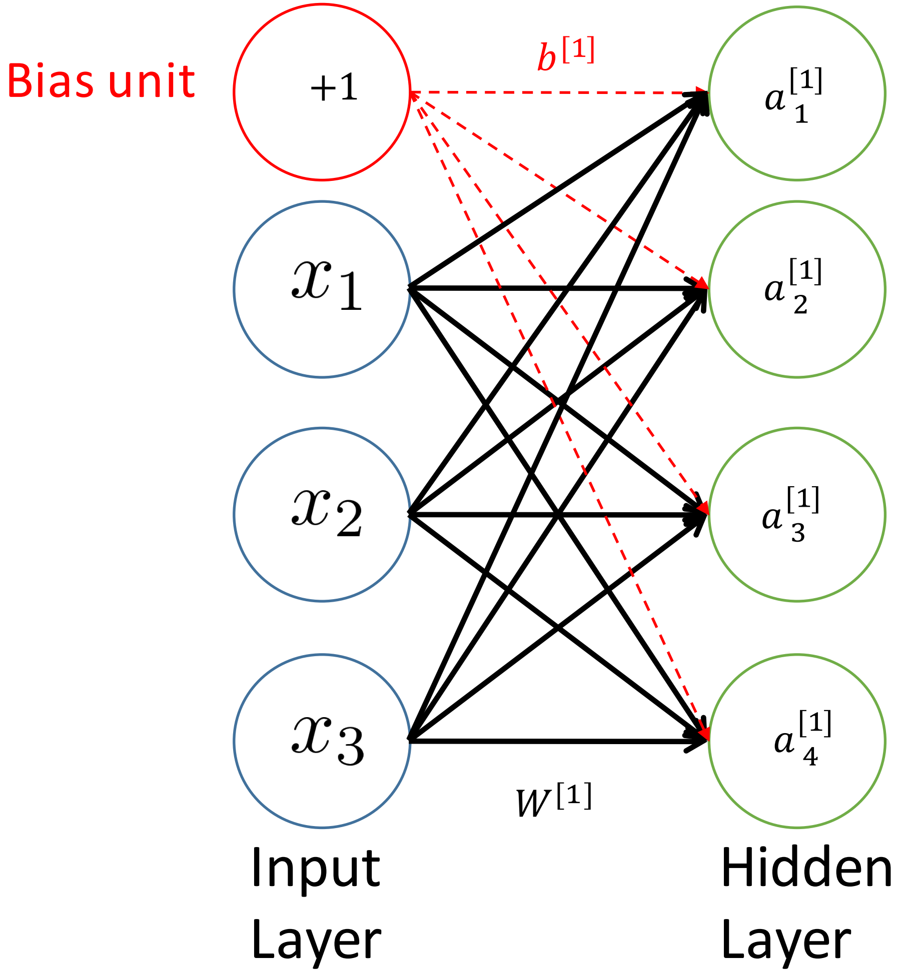


$$
\begin{array}{cccc}
z^{[1]} & W^{[1]} & b^{[1]} & a^{[1]}
\end{array}
$$

$$
\begin{aligned}
z_1^{[1]} &= w_1^{[1]T} x + b_1^{[1]}, \quad a_1^{[1]} = g(z_1^{[1]}) \\
z_2^{[1]} &= w_2^{[1]T} x + b_2^{[1]}, \quad a_2^{[1]} = g(z_2^{[1]}) \\
z_3^{[1]} &= w_3^{[1]T} x + b_3^{[1]}, \quad a_3^{[1]} = g(z_3^{[1]}) \\
z_4^{[1]} &= w_4^{[1]T} x + b_4^{[1]}, \quad a_4^{[1]} = g(z_4^{[1]})
\end{aligned}
$$

$$
W^{[1]} = \begin{bmatrix} w_1^{[1]T} \\ w_2^{[1]T} \\ w_3^{[1]T} \\ w_4^{[1]T} \end{bmatrix} = \begin{bmatrix} w_{11}^{[1]} & w_{12}^{[1]} & w_{13}^{[1]} \\ w_{21}^{[1]} & w_{22}^{[1]} & w_{23}^{[1]} \\ w_{31}^{[1]} & w_{32}^{[1]} & w_{33}^{[1]} \\ w_{41}^{[1]} & w_{42}^{[1]} & w_{43}^{[1]} \end{bmatrix}
$$

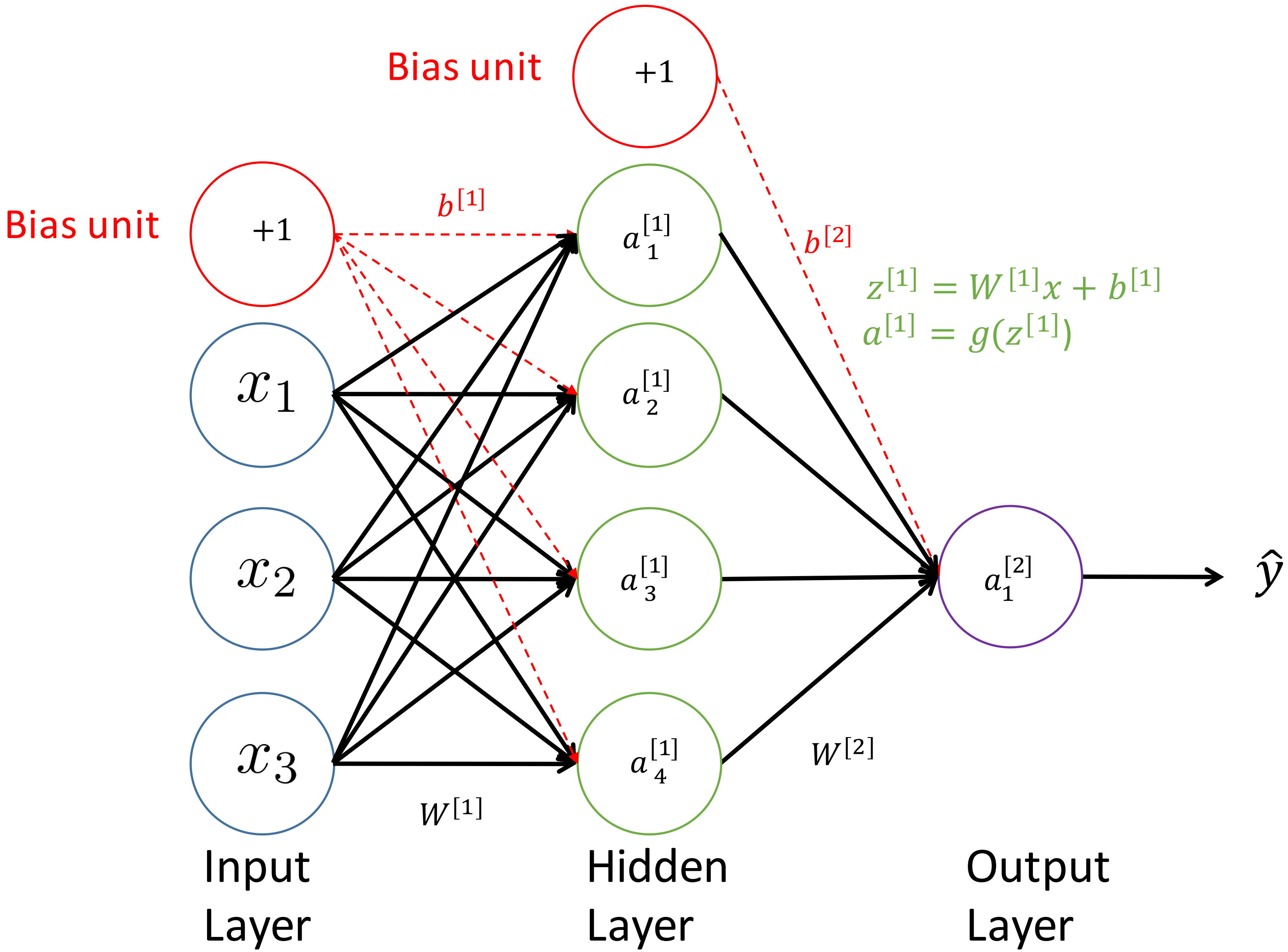
ANN Model Representation



$$\begin{array}{cccccc}
z^{[1]} & & W^{[1]} & & b^{[1]} & a^{[1]} \\
z_1^{[1]} & = & w_1^{[1]T} & x + & b_1^{[1]}, & a_1^{[1]} = g(z_1^{[1]}) \\
z_2^{[1]} & = & w_2^{[1]T} & x + & b_2^{[1]}, & a_2^{[1]} = g(z_2^{[1]}) \\
z_3^{[1]} & = & w_3^{[1]T} & x + & b_3^{[1]}, & a_3^{[1]} = g(z_3^{[1]}) \\
z_4^{[1]} & = & w_4^{[1]T} & x + & b_4^{[1]}, & a_4^{[1]} = g(z_4^{[1]})
\end{array}$$

$$W^{[1]} = \begin{bmatrix} w_1^{[1]T} \\ w_2^{[1]T} \\ w_3^{[1]T} \\ w_4^{[1]T} \end{bmatrix} = \begin{bmatrix} w_{11}^{[1]} & w_{12}^{[1]} & w_{13}^{[1]} \\ w_{21}^{[1]} & w_{22}^{[1]} & w_{23}^{[1]} \\ w_{31}^{[1]} & w_{32}^{[1]} & w_{33}^{[1]} \\ w_{41}^{[1]} & w_{42}^{[1]} & w_{43}^{[1]} \end{bmatrix}$$

$$z^{[1]} = W^{[1]}x + b^{[1]}$$
$$a^{[1]} = g(z^{[1]})$$



$$z^{[1]} = W^{[1]}x + b^{[1]}$$
$$a^{[1]} = g(z^{[1]})$$

$$z^{[2]} = W^{[2]}a^{[1]} + b^{[2]}$$
$$\hat{y} = a^{[2]} = g(z^{[2]})$$

ANN Model Representation: Vectorization

$$z^{[1]} = W^{[1]}x + b^{[1]}$$
$$a^{[1]} = g(z^{[1]})$$

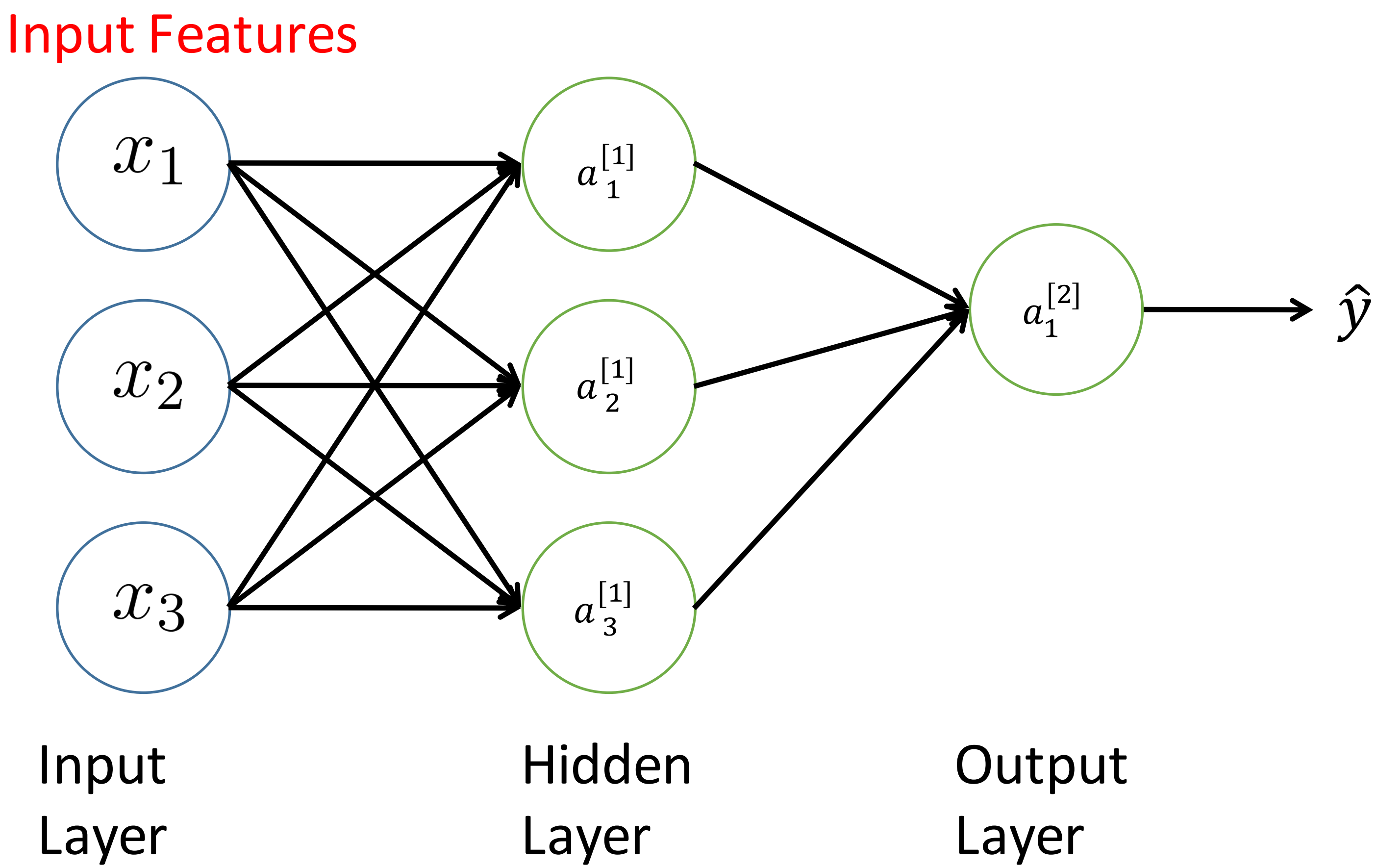
$$z^{[2]} = W^{[2]}a^{[1]} + b^{[2]}$$
$$\hat{y} = a^{[2]} = g(z^{[2]})$$

For one example

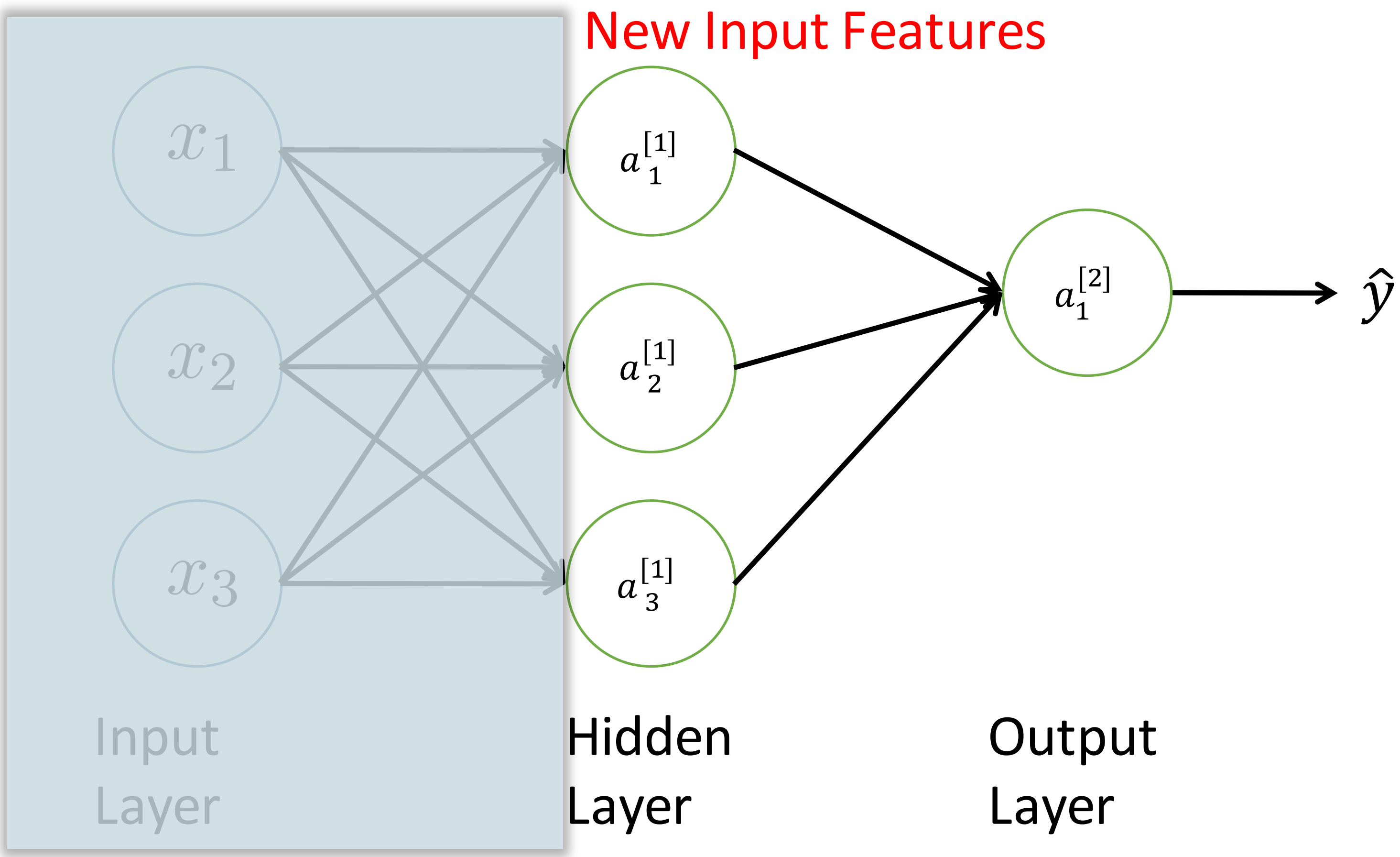
$$Z^{[1]} = W^{[1]}X + b^{[1]}$$
$$A^{[1]} = g(Z^{[1]})$$
$$Z^{[2]} = W^{[2]}A^{[1]} + b^{[2]}$$
$$A^{[2]} = g(Z^{[2]})$$

For All Examples (matrix notation)

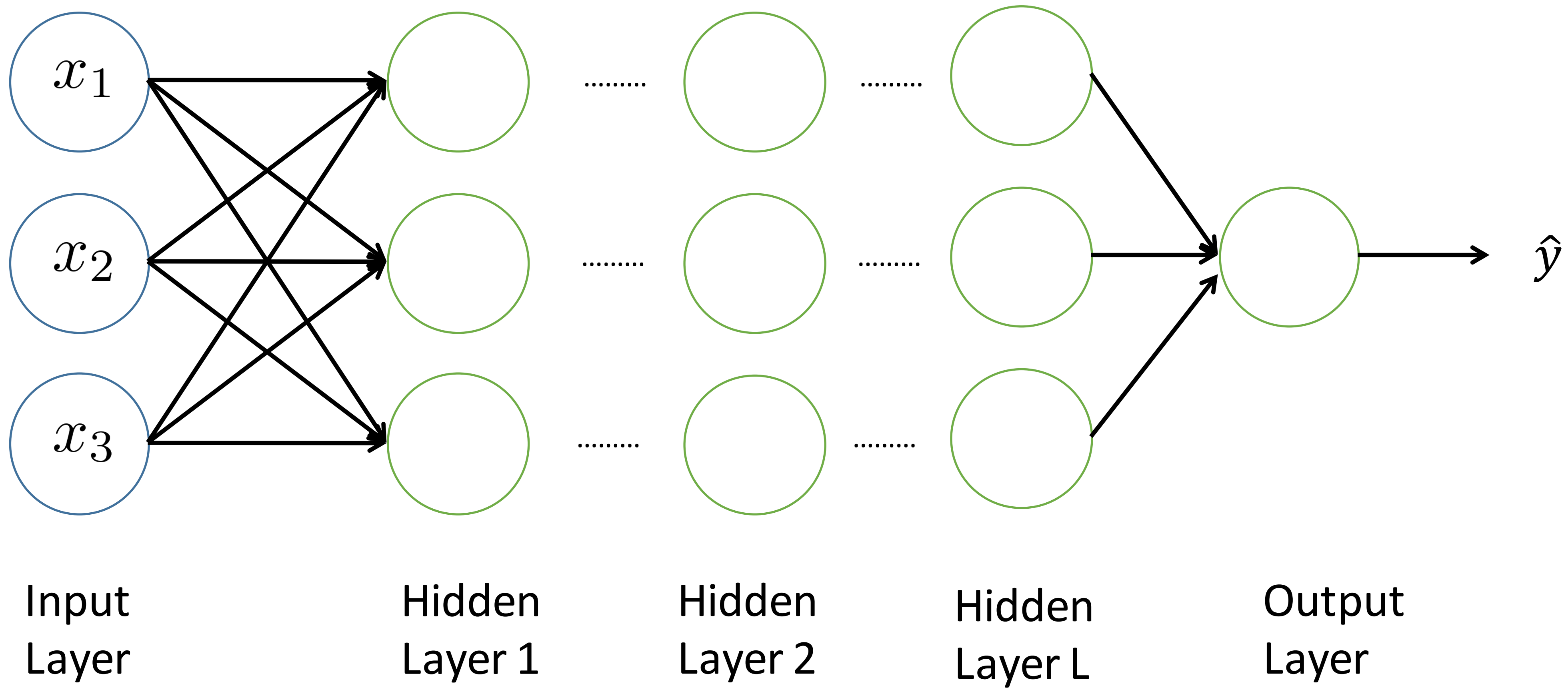
ANN Representation: Learning its own features



ANN Representation: Learning its own features



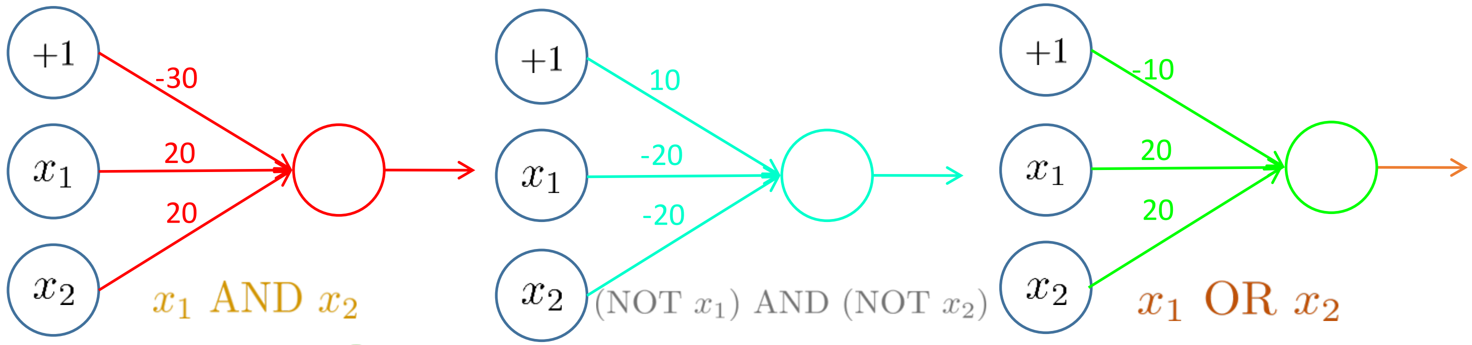
ANN Representation: Deep NN

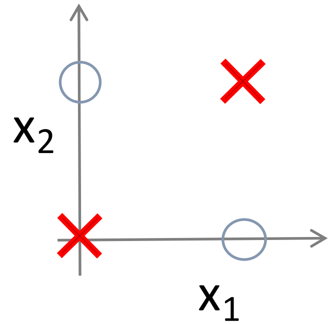


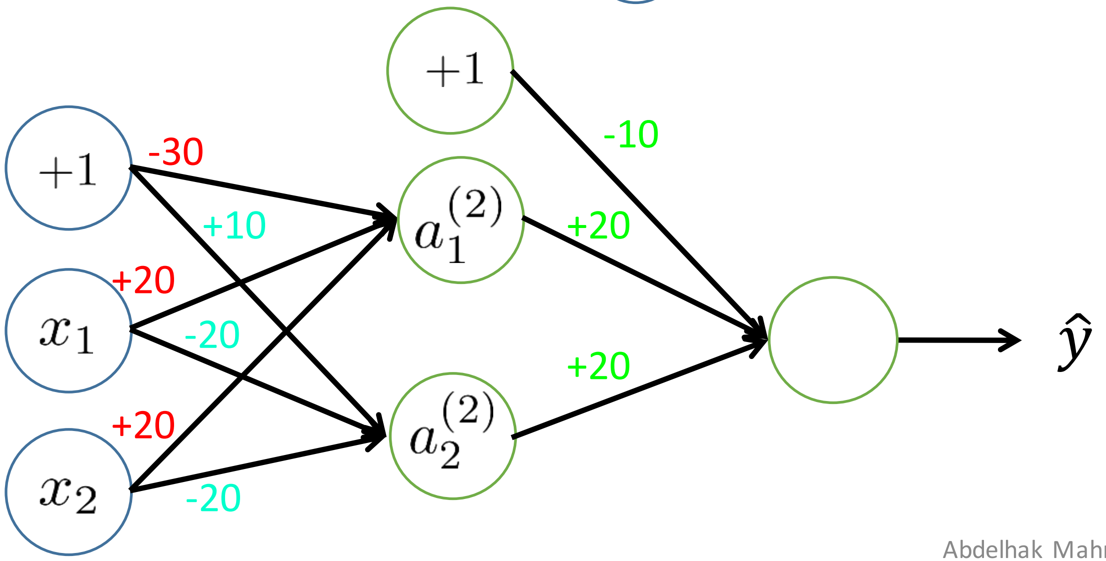
Why ANN?

- Features Generation
  - Learn Features by it self
- Data non linearly separable
  - Learn complex non linear functions
- Deals with Unstructured data
  - Convolutional Neural Networks (Vision)
  - Recurrent Neural Networks (Sequence)
  - Generative Adversarial Networks (Generate data)

Simple Example

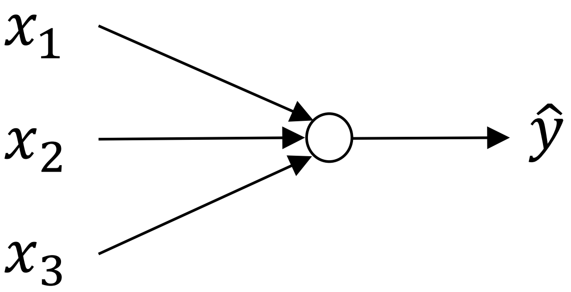






x_1	x_2	$a_1^{(2)}$	$a_2^{(2)}$	$\hat{y}$
0	0	0	1	1
0	1	0	0	0
1	0	0	0	0
1	1	1	0	1

ANN Learning



Remember Logistic Regression

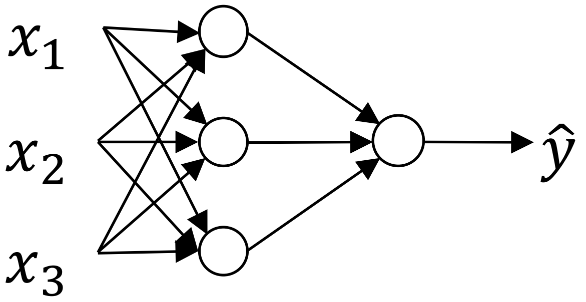
$$x \longrightarrow \boxed{z = w^T x + b} \longrightarrow \boxed{a = g(z)} \longrightarrow \boxed{\mathcal{L}(a, y)}$$

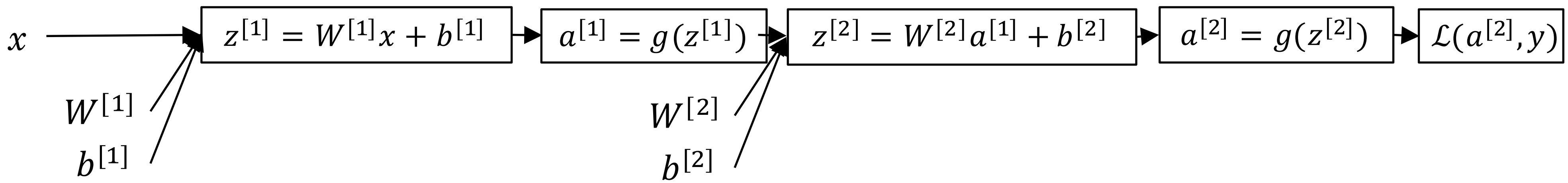
dw
db $\quad dz \quad da$

Notation: $dt = \dfrac{\partial \mathcal{L}}{\partial t}$

Cost Function $\quad \mathcal{L}(a, y) = -\frac{1}{m}[\sum_{i=1}^{m} y^{(i)} \log\, a^{(i)} + (1 - y^{(i)}) \log{(1 - a^{(i)})}]$

ANN Learning: Forward Propagation

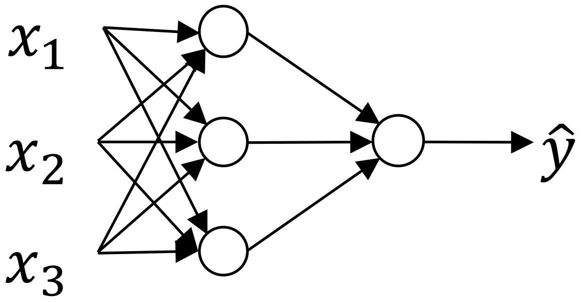




Cost Function

$$\mathcal{L}\left(a^{[2]}, y\right) = -\frac{1}{m}\sum_{i=0}^{m}\left(y^{(i)}\log\left(a^{[2](i)}\right) + (1-y^{(i)})\log\left(1-a^{[2](i)}\right)\right)$$

ANN Learning: Backward Propagation



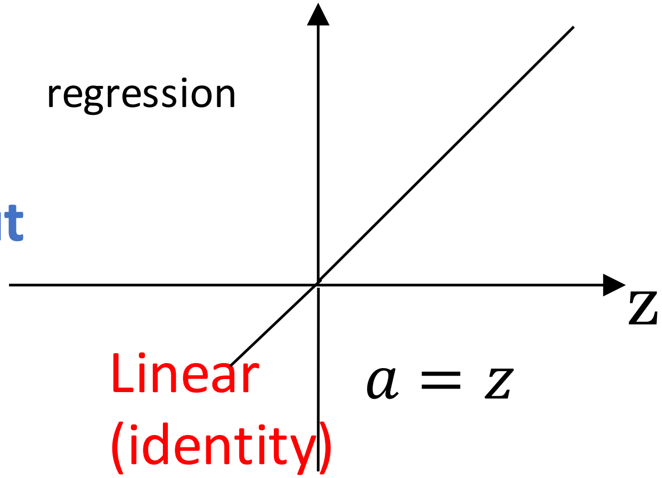
$$x \rightarrow \boxed{z^{[1]} = W^{[1]}x + b^{[1]}} \rightarrow \boxed{a^{[1]} = g(z^{[1]})} \rightarrow \boxed{z^{[2]} = W^{[2]}a^{[1]} + b^{[2]}} \rightarrow \boxed{a^{[2]} = g(z^{[2]})} \rightarrow \boxed{\mathcal{L}(a^{[2]}, y)}$$

$$\begin{matrix} dW^{[1]} \\ db^{[1]} \end{matrix} \leftarrow dz^{[1]} \leftarrow \begin{matrix} da^{[1]} \\ dW^{[2]} \\ db^{[2]} \end{matrix} \leftarrow dz^{[2]} \leftarrow da^{[2]}$$

ANN Learning: Activation Functions

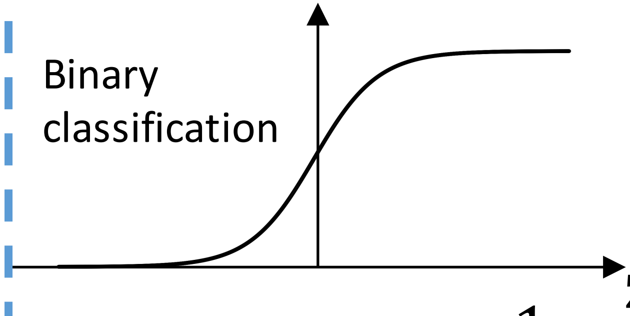
Output Layer

regression



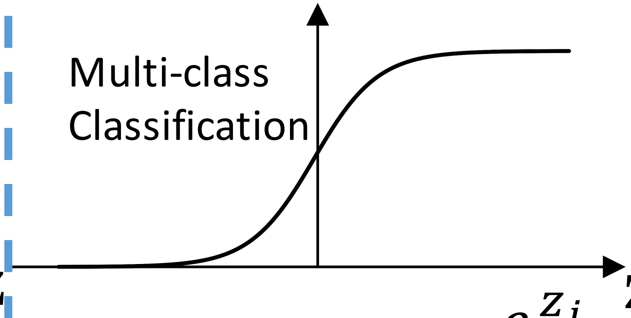
Linear (identity) $a = z$

Binary classification



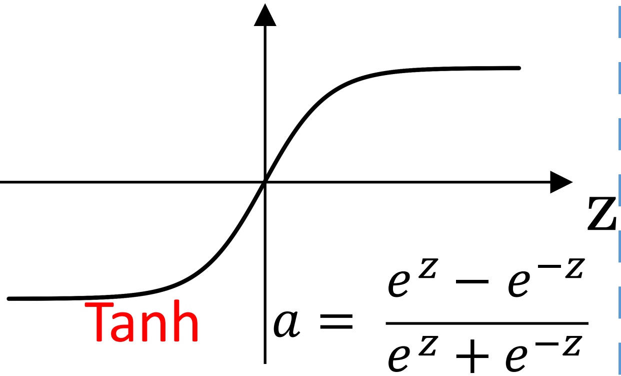
Sigmoid $a = \frac{1}{1 + e^{-z}}$

Multi-class Classification

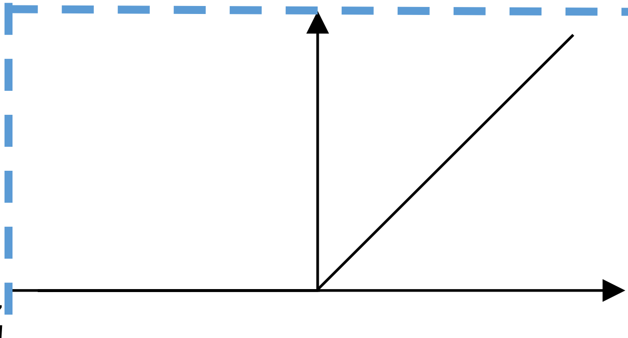


Softmax $a_j = \frac{e^{z_j}}{\sum_{j=1}^{K} e^{z_j}}$

Hidden Layers

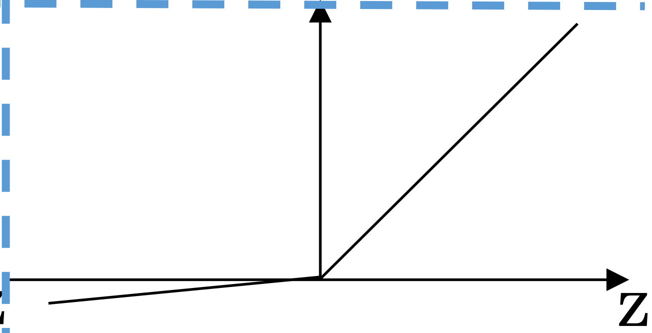


Tanh $a = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$



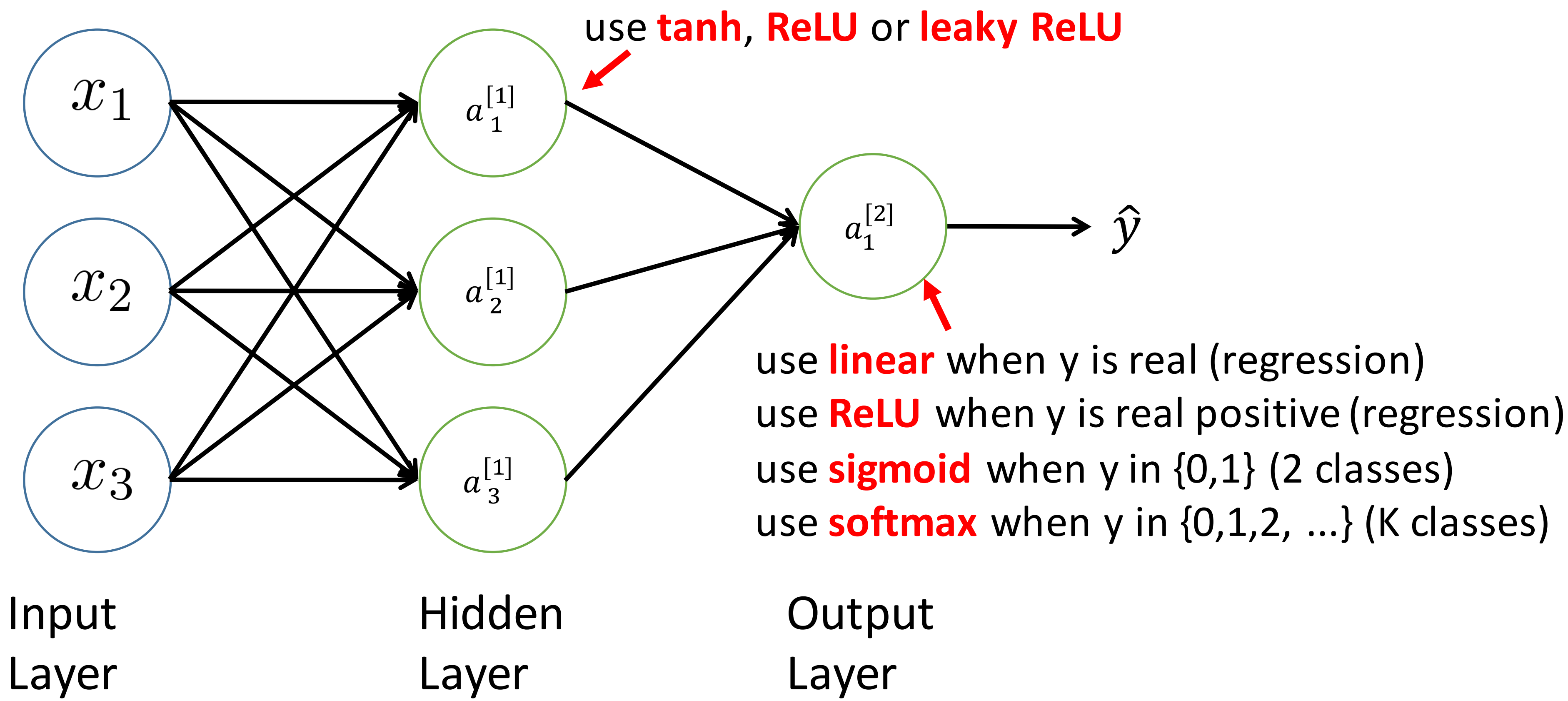
ReLU $a = \max(0, z)$

Rectified Linear Unit



Leaky ReLU $a = \max(0.01z, z)$

ANN Learning : Activation Functions



ANN Learning : Backward Propagation

f ∘ g	(f′ ∘ g) × g′
f(g(x))	f′(g(x))g′(x)
$\frac{dy}{dx} = \frac{dy}{du}\frac{du}{dx}$	

$W^{[1]}$, $b^{[1]}$ $W^{[2]}$, $b^{[2]}$

$$x \rightarrow \boxed{z^{[1]} = W^{[1]}x + b^{[1]}} \rightarrow \boxed{a^{[1]} = g(z^{[1]})} \rightarrow \boxed{z^{[2]} = W^{[2]}a^{[1]} + b^{[2]}} \rightarrow \boxed{a^{[2]} = g(z^{[2]})} \rightarrow \boxed{\mathcal{L}(a^{[2]}, y)}$$

$$dW^{[1]}, db^{[1]} \leftarrow dz^{[1]} \leftarrow da^{[1]} \leftarrow dz^{[2]} \leftarrow da^{[2]}$$

$dW^{[2]}$, $db^{[2]}$

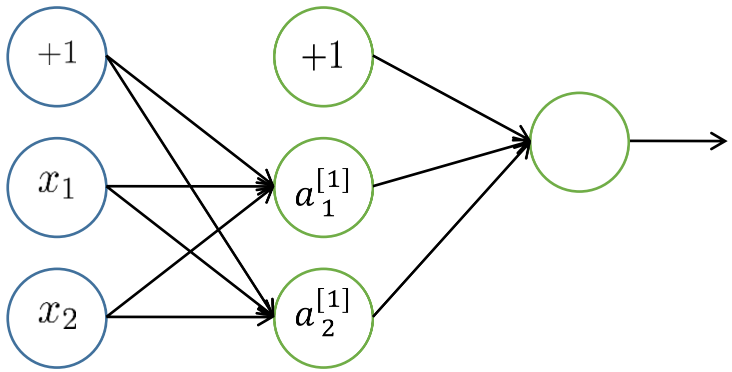
For one example

$$dz^{[2]} = a^{[2]} - y$$
$$dW^{[2]} = dz^{[2]}a^{[1]^T}$$
$$db^{[2]} = dz^{[2]}$$
$$dz^{[1]} = W^{[2]T}dz^{[2]} * g^{[1]\prime}(z^{[1]})$$
$$dW^{[1]} = dz^{[1]}x^T$$
$$db^{[1]} = dz^{[1]}$$

For All Examples (or a batch of examples)

$$dZ^{[2]} = A^{[2]} - Y$$
$$dW^{[2]} = \frac{1}{m}dZ^{[2]}A^{[1]^T}$$
$$db^{[2]} = \frac{1}{m}np.sum(dZ^{[2]}, axis = 1, keepdims = True)$$
$$dZ^{[1]} = W^{[2]T}dZ^{[2]} * g^{[1]\prime}(Z^{[1]})$$
$$dW^{[1]} = \frac{1}{m}dZ^{[1]}X^T$$
$$db^{[1]} = \frac{1}{m}np.sum(dZ^{[1]}, axis = 1, keepdims = True)$$

ANN Learning: Random Initialization

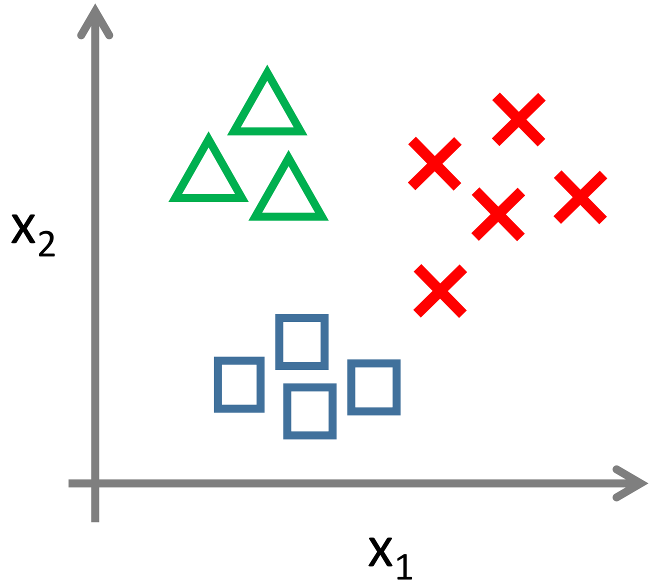


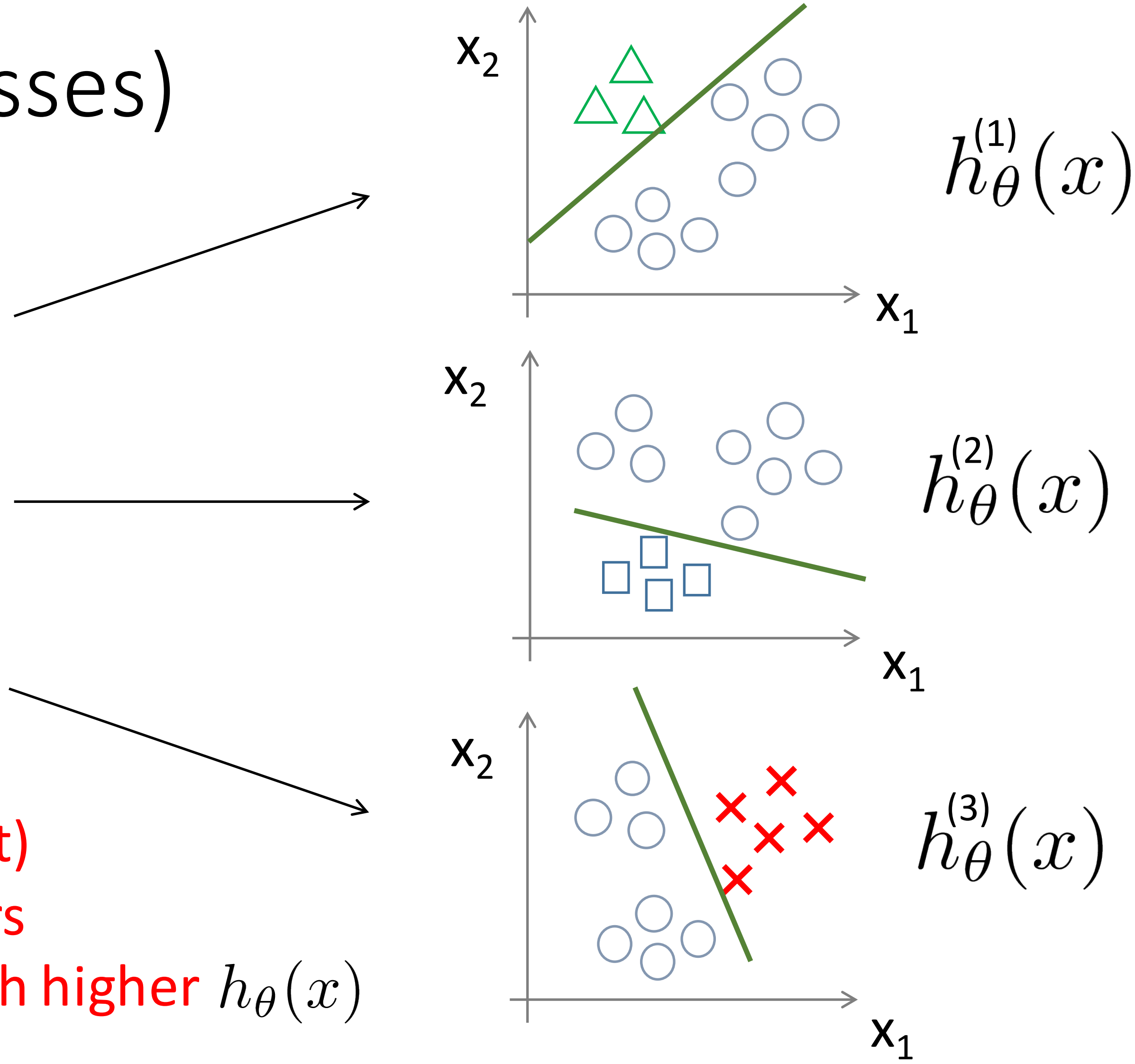
If $W^{[1]}$ and $b^{[1]}$ are initialized with zeros, after each update, parameters corresponding to inputs going into each of the two hidden units are identical, which results in $a_1^{[1]} = a_2^{[1]}$

ANN Implementation

1. Pick a network architecture
   - No. of input units: Dimension of features
   - No. output units: Number of classes
   - No. hidden layers
   - No. hidden units: (have same no. in every layer, the more the better)
2. Randomly initialize weights
3. Repeat (chose a number of iterations)
   1. Implement forward propagation
   2. Compute cost function
   3. Implement backward propagation to compute partial derivatives
   4. Update the W and b: $W^{[l]} =: W^{[l]} - \alpha \, dW^{[l]}$ $\quad b^{[l]} =: b^{[l]} - \alpha \, db^{[l]}$

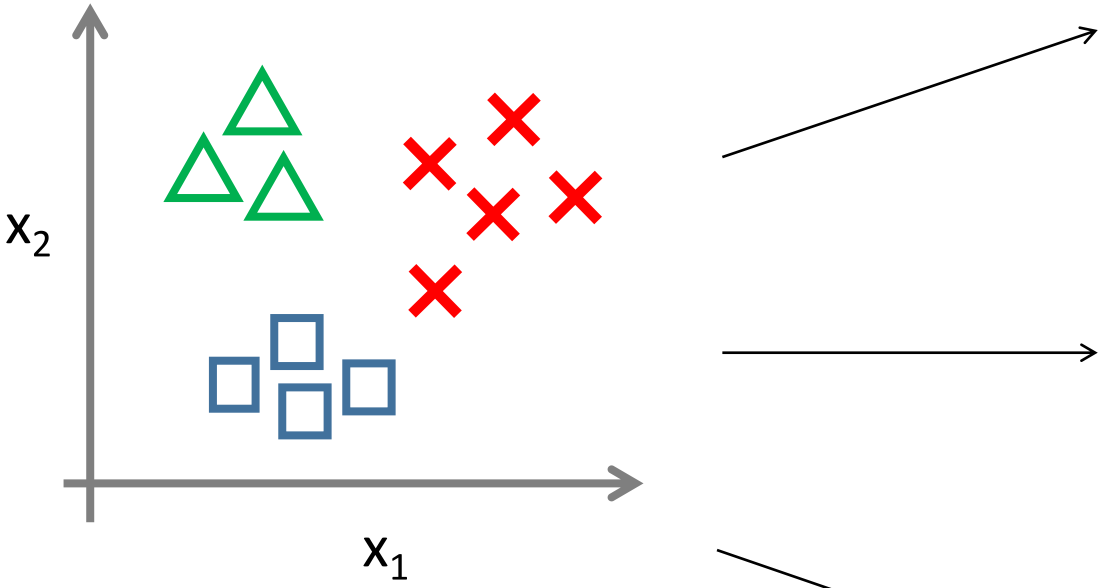
Multi-Class (N-classes)

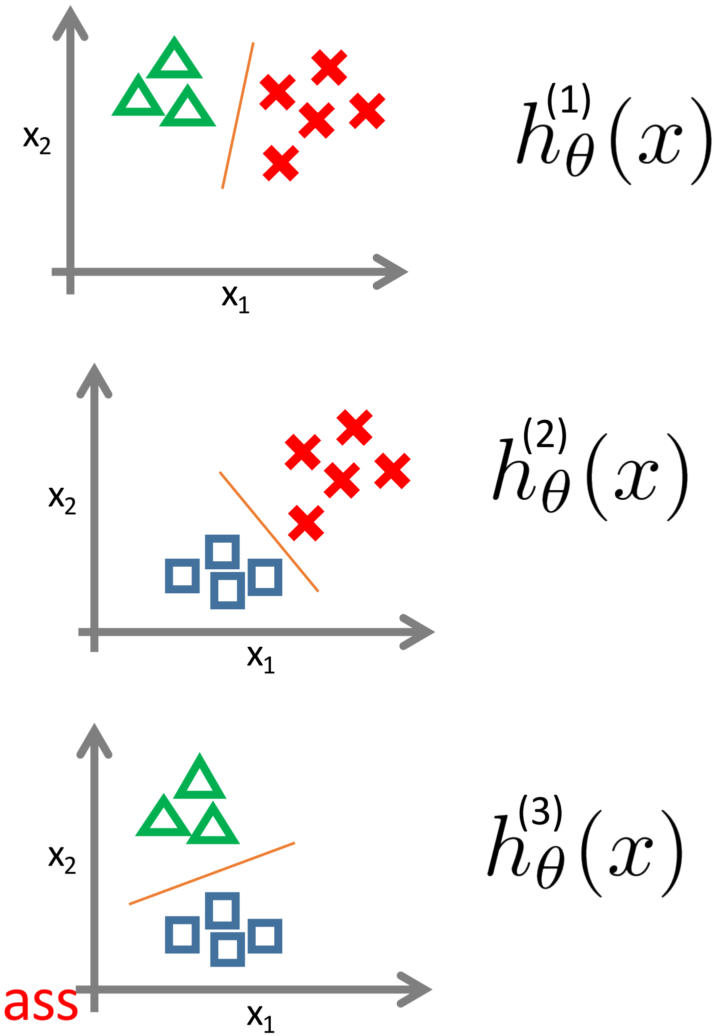




- One-vs-All (One-vs-Rest)
- Train **N** binary classifiers
- Classify to the class with higher $h_\theta(x)$

Multi-Class (N-classes)





- One-vs-One
- Train **N–(N–1)/2** binary Classifiers
- Classify to the most frequently assigned class