

CS 481

Artificial Intelligence Language Understanding

March 7, 2023

Announcements / Reminders

- Please follow the Week 08 To Do List instructions
- Exam dates:
 - Final: 04/27/2023 during Thursday lecture time

Plan for Today

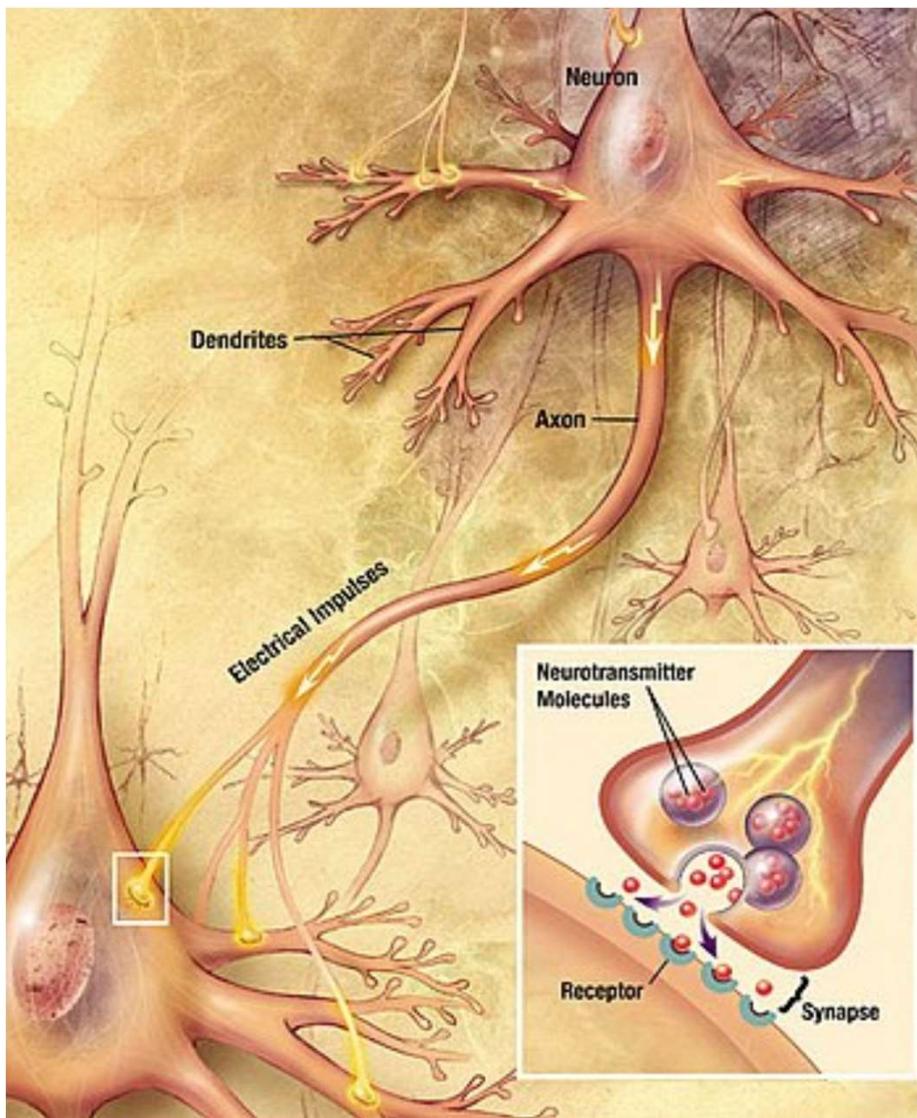
- Text classification
 - other types of classifiers
 - logistic regression

Text Classification: Supervised ML

- Various Machine Learning supervised learning classifier approaches can be employed:
 - Naïve Bayes
 - Logistic regression
 - Neural networks
 - k-Nearest Neighbors
 - etc.

Classification with Perceptrons

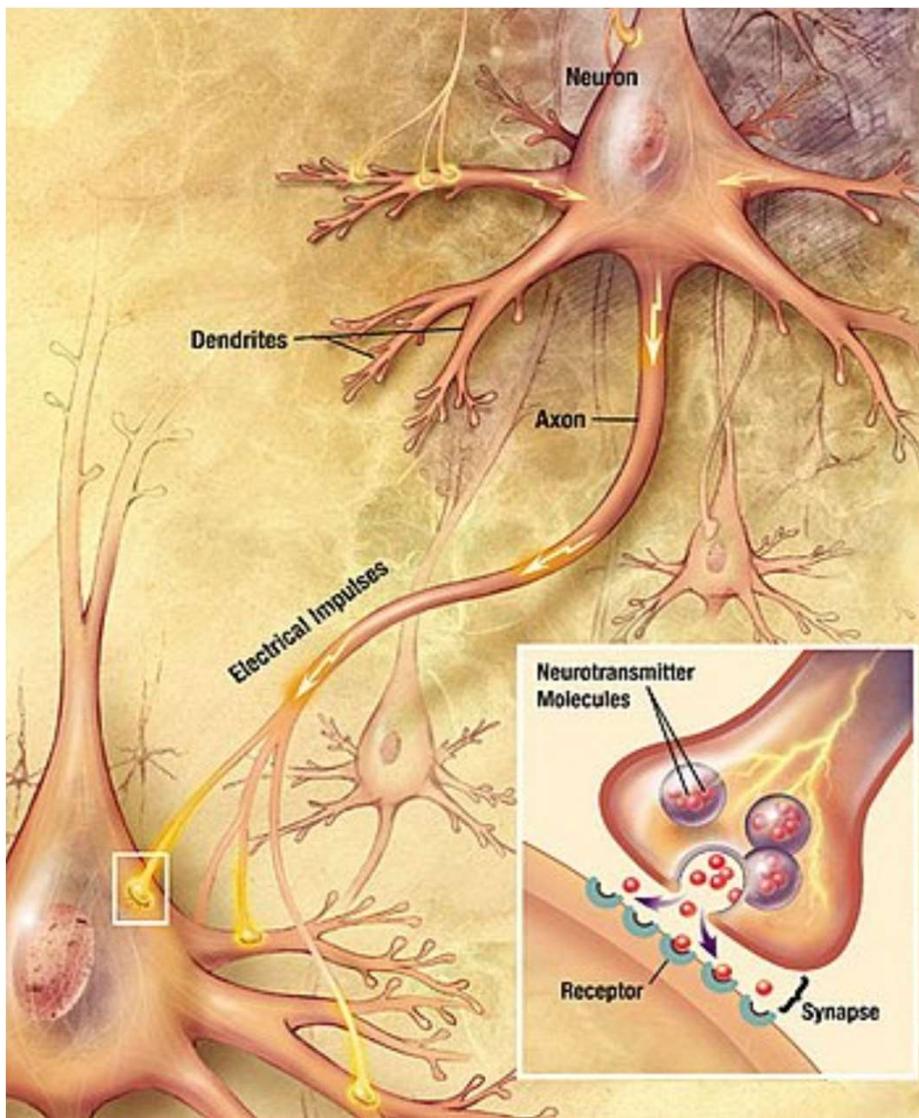
A Biological Neuron



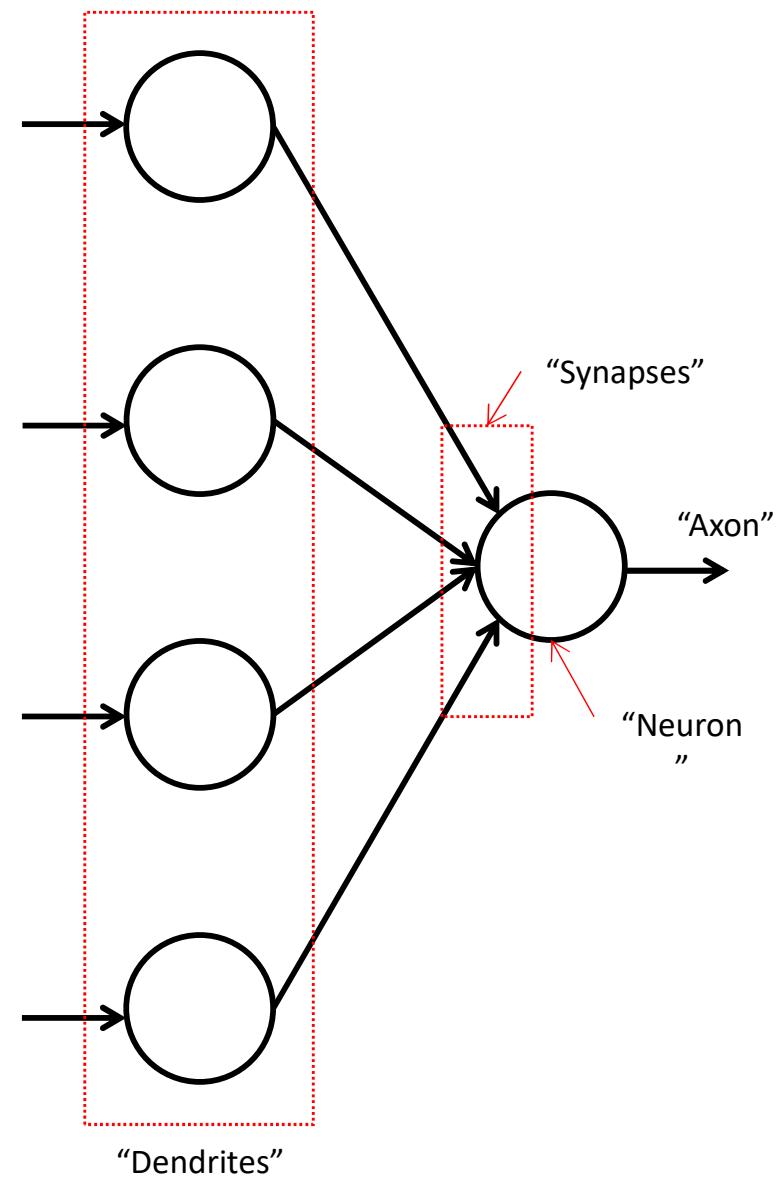
Source: <https://en.wikipedia.org/wiki/Neuron>

A **neuron** or nerve cell is an electrically excitable cell that **communicates with other cells via specialized connections called synapses**. Most **neurons receive signals via the dendrites and soma** and **send out signals down the axon**. At the majority of synapses, signals cross from the axon of one neuron to a dendrite of another.

Biological vs. Artificial Neuron



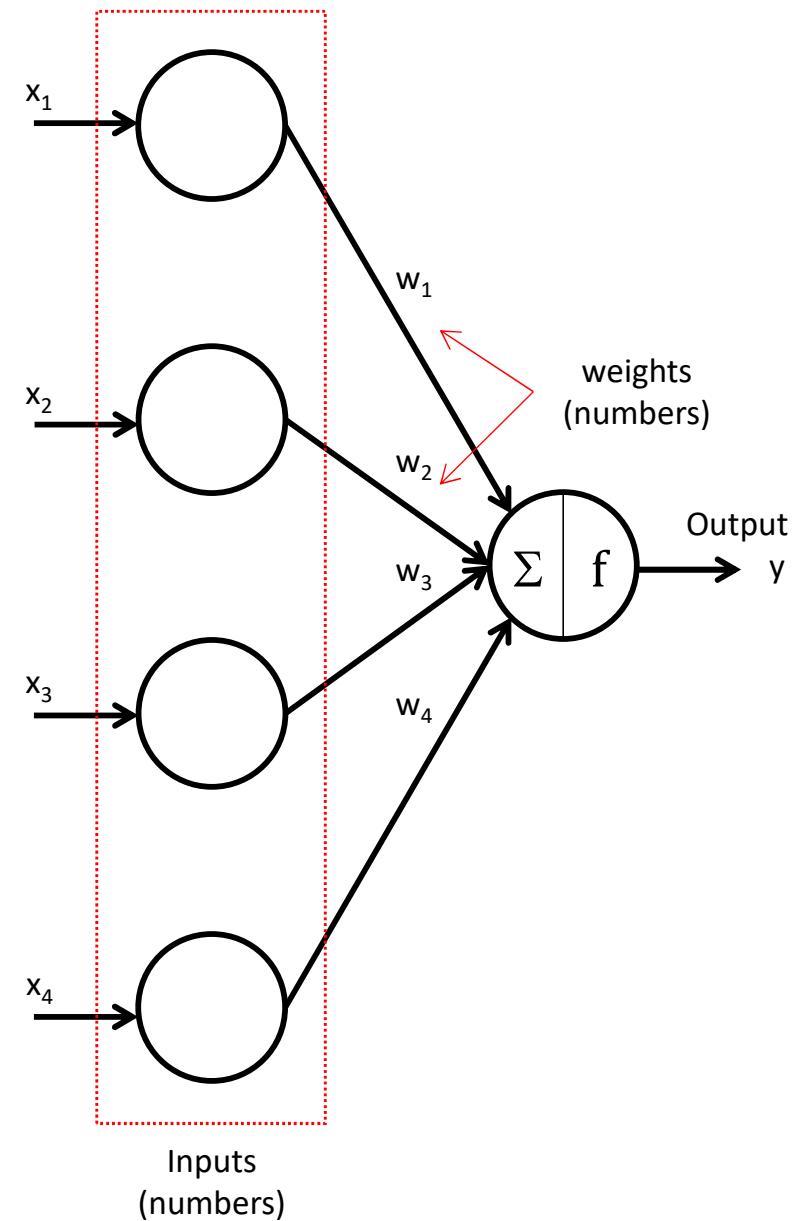
Source: <https://en.wikipedia.org/wiki/Neuron>



Artificial Neuron (Perceptron)

A (single-layer) **perceptron** is a model of a biological neuron. It is made of the following components:

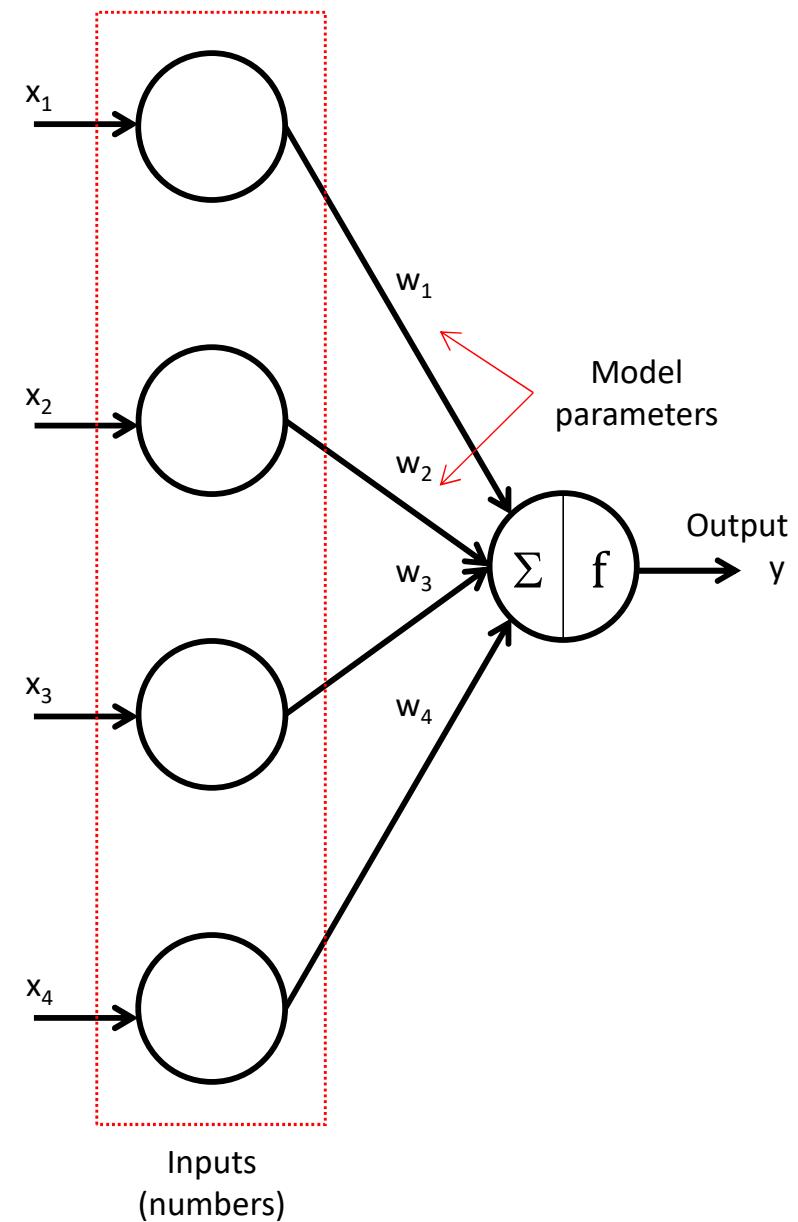
- inputs x_i - numerical values representing information
- weights w_i - numerical values representing how “important” corresponding input is
- weighted sum: $\sum w_i * x_i$
- activation function f that decides if the neuron “fires”



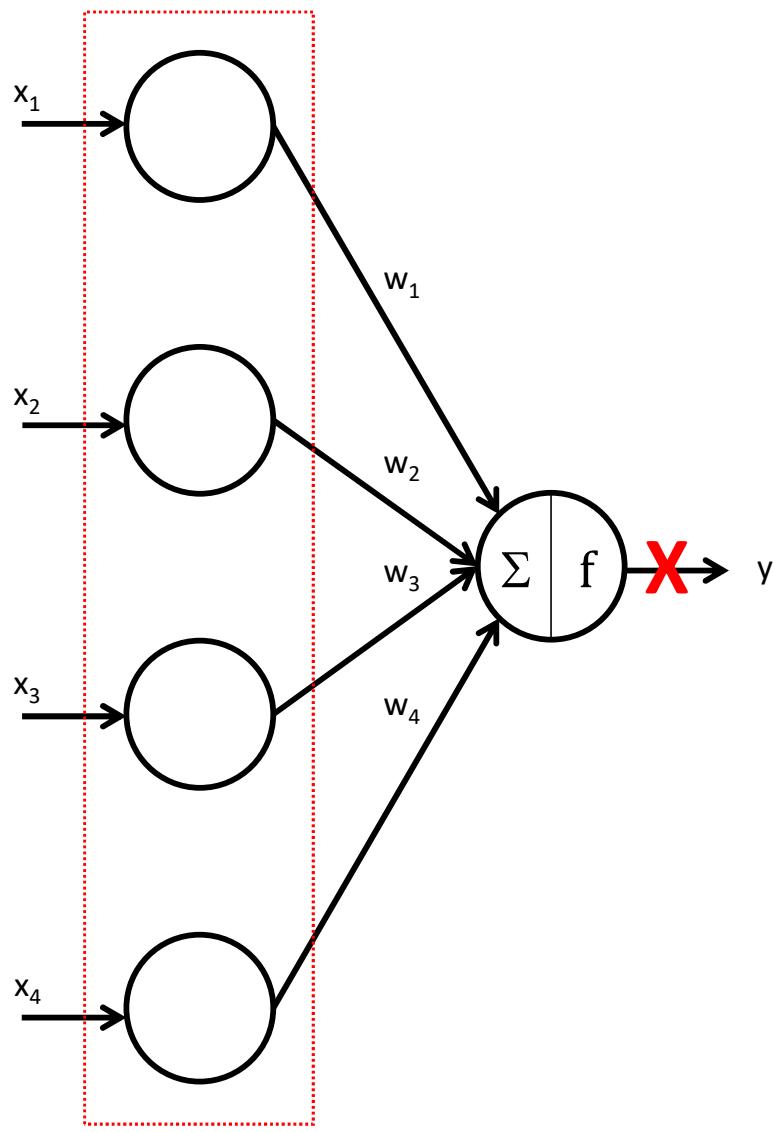
Artificial Neuron (Perceptron)

A (single-layer) **perceptron** is a model of a biological neuron. It is made of the following components:

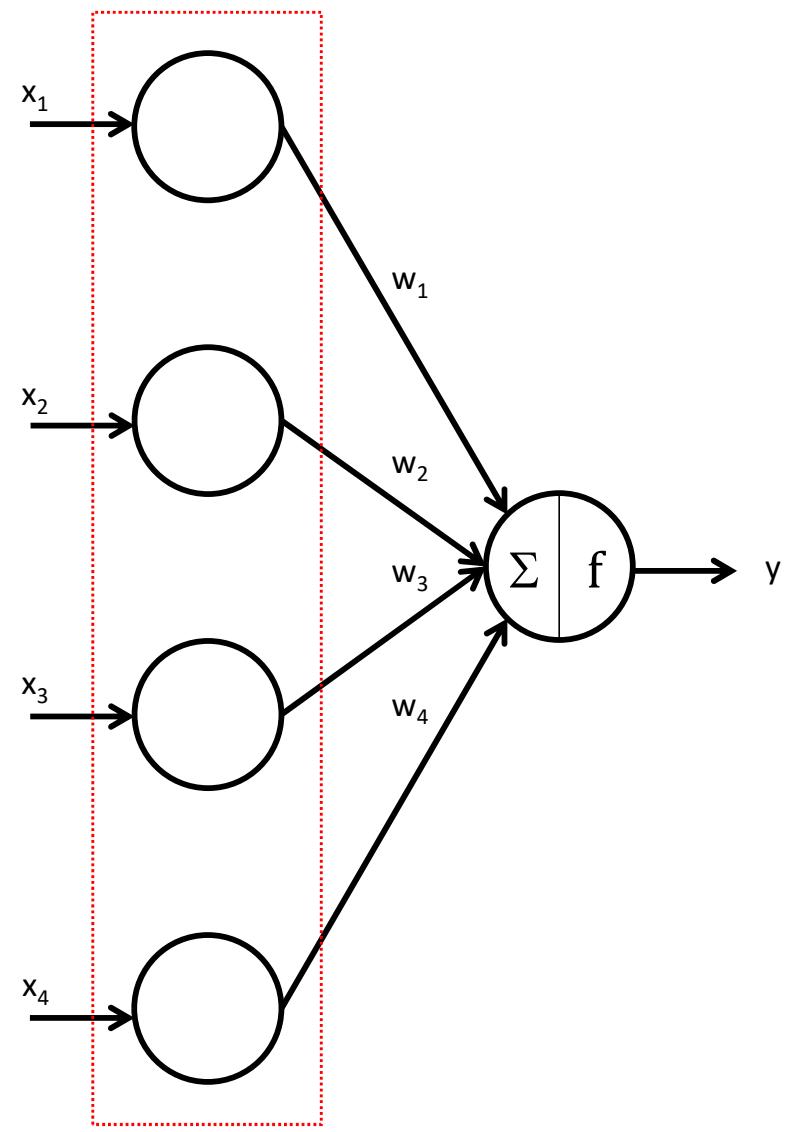
- inputs x_i - numerical values representing information
- weights w_i - numerical values representing how “important” corresponding input is
- weighted sum: $\sum w_i * x_i$
- activation function f that decides if the neuron “fires”



Artificial Neuron (Perceptron)

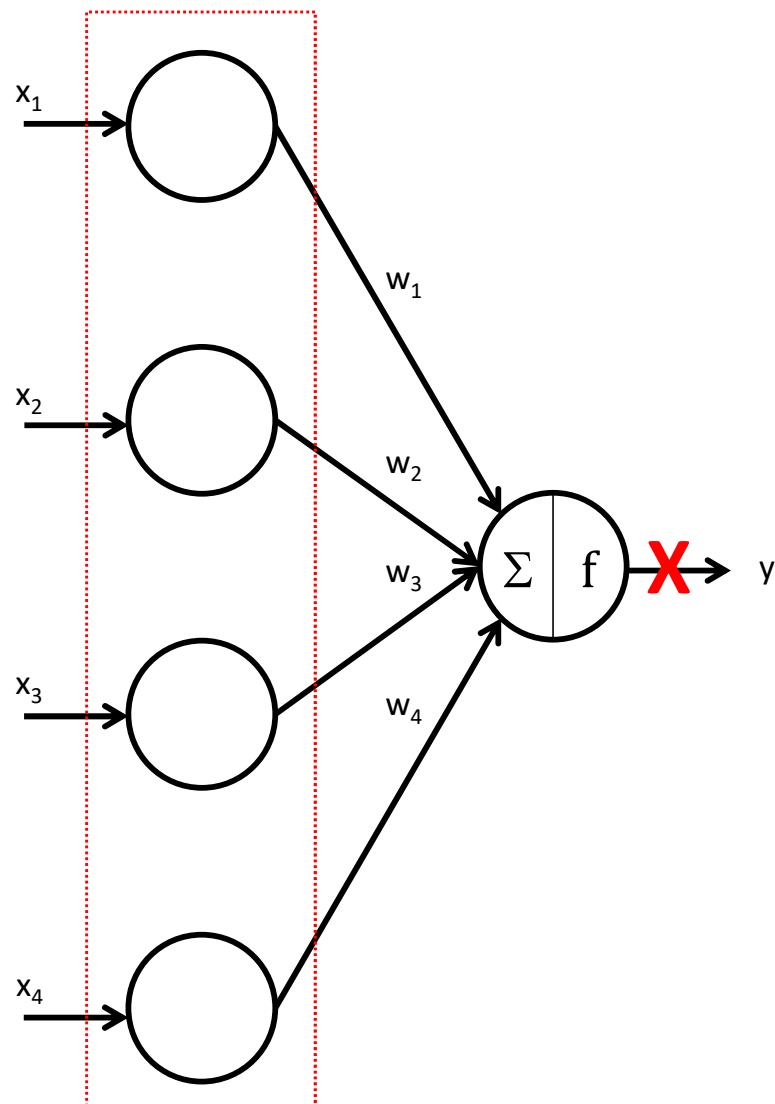


$\sum w_i * x_i < 0 \rightarrow f = 0 \rightarrow \text{DON'T "fire"}$

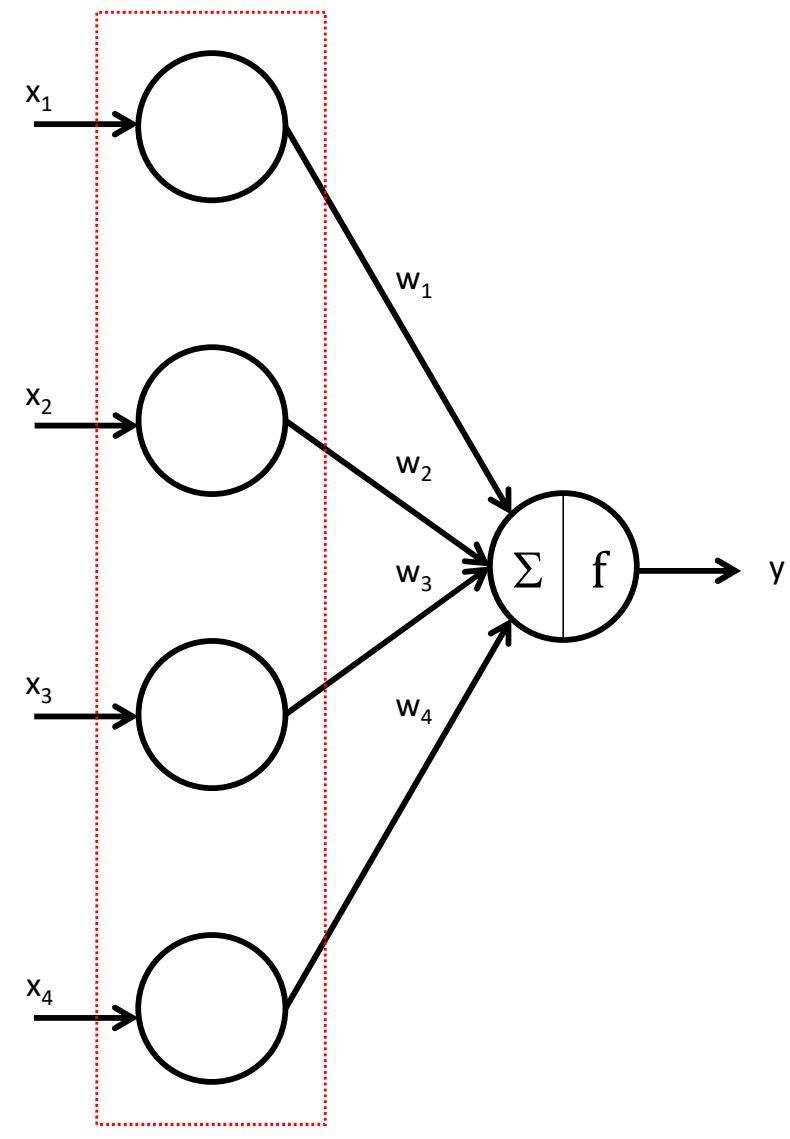


$\sum w_i * x_i \geq 0 \rightarrow f = 1 \rightarrow \text{"fire"}$

Single-layer Perceptron as a Classifier

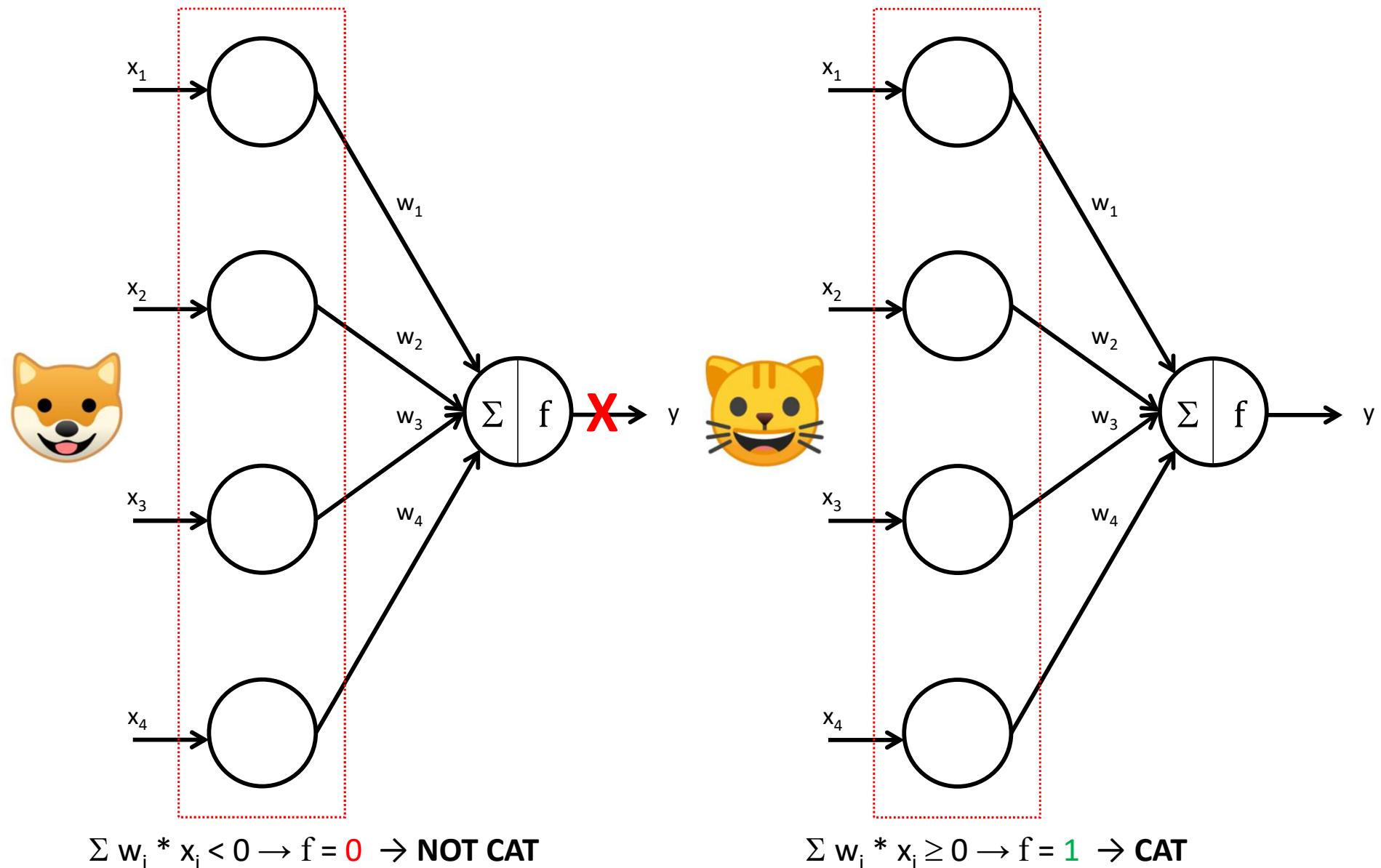


$\sum w_i * x_i < 0 \rightarrow f = 0 \rightarrow \text{NO}$

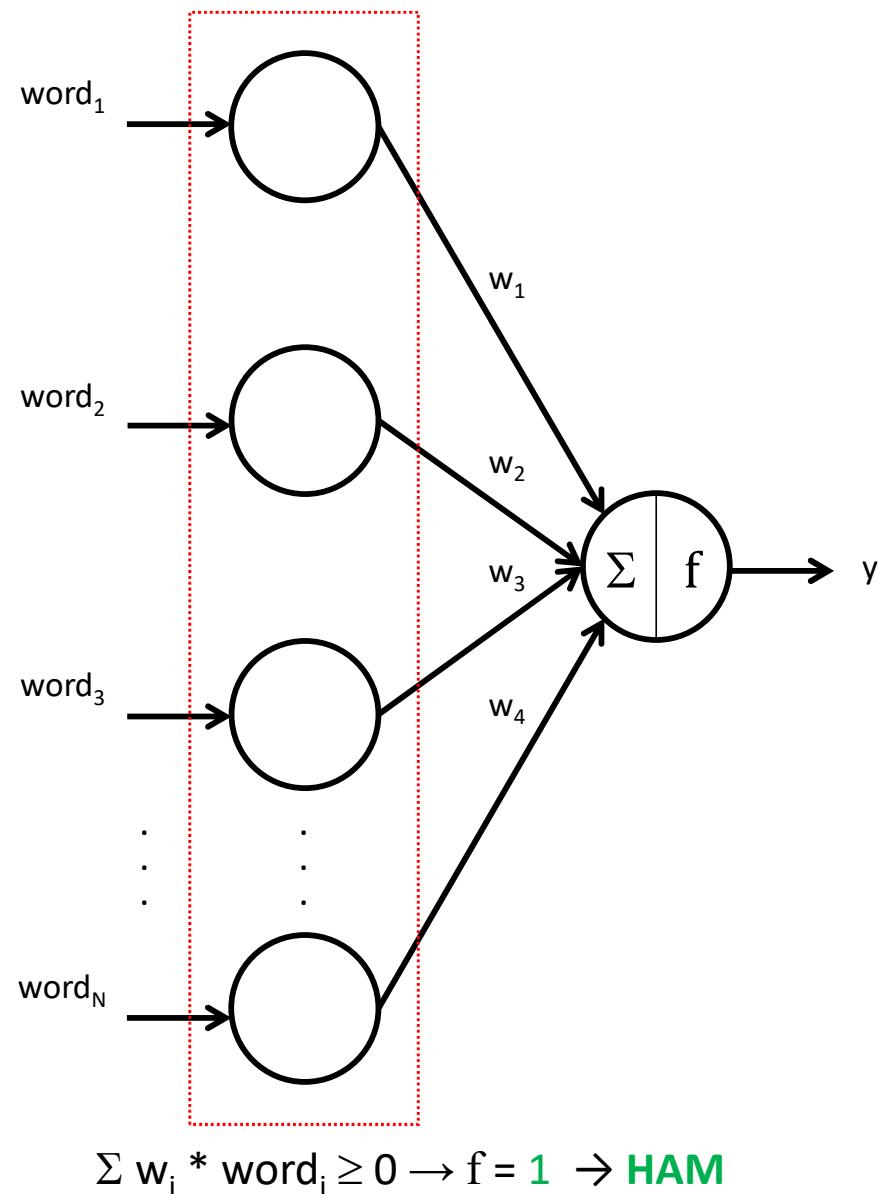
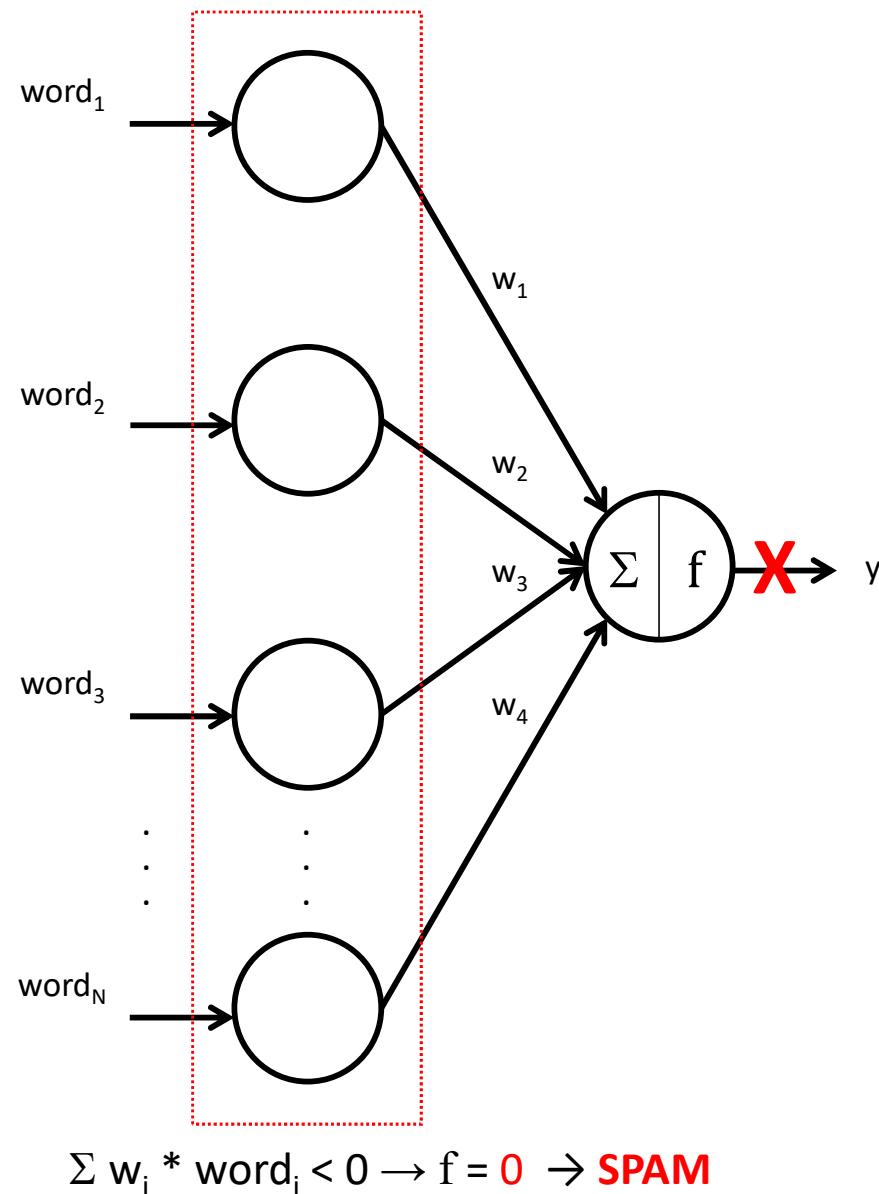


$\sum w_i * x_i \geq 0 \rightarrow f = 1 \rightarrow \text{YES}$

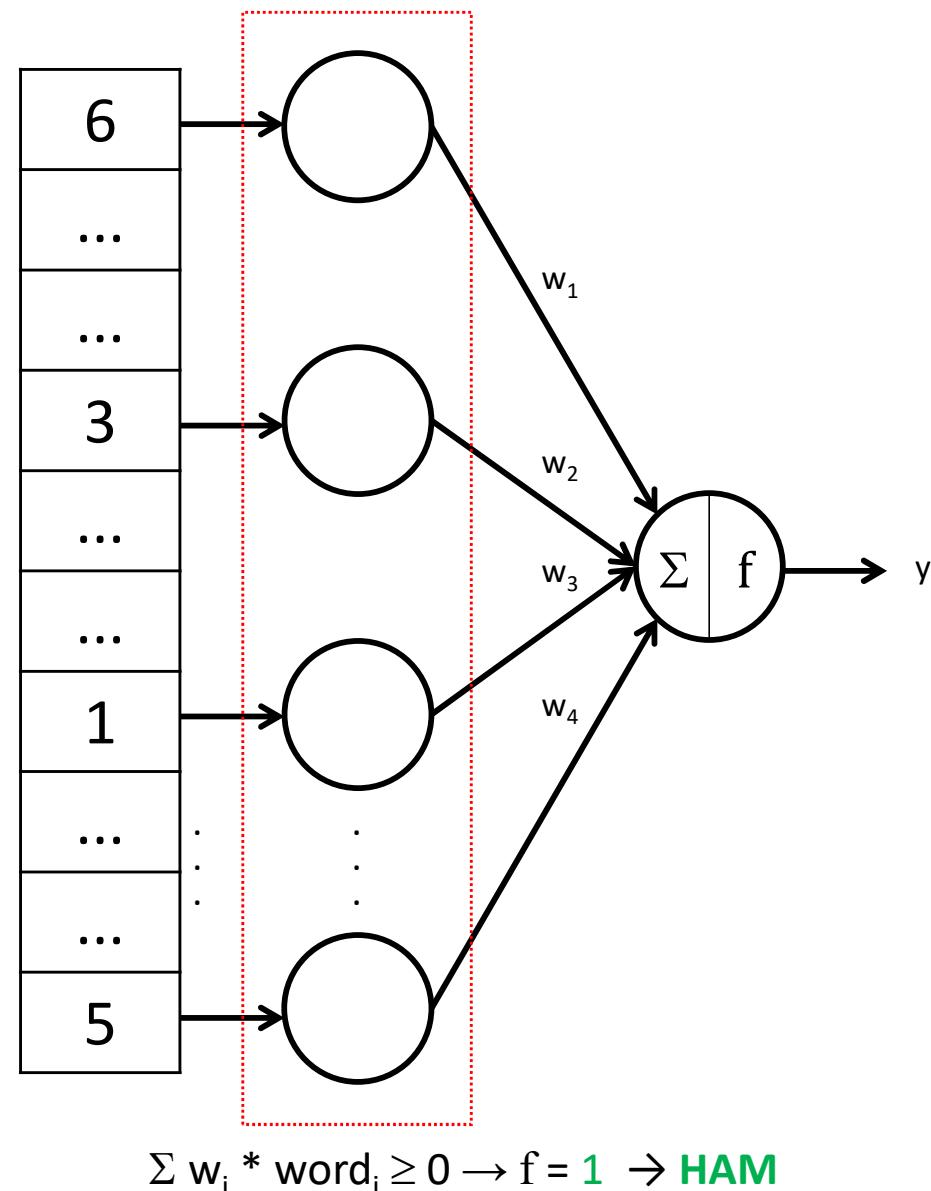
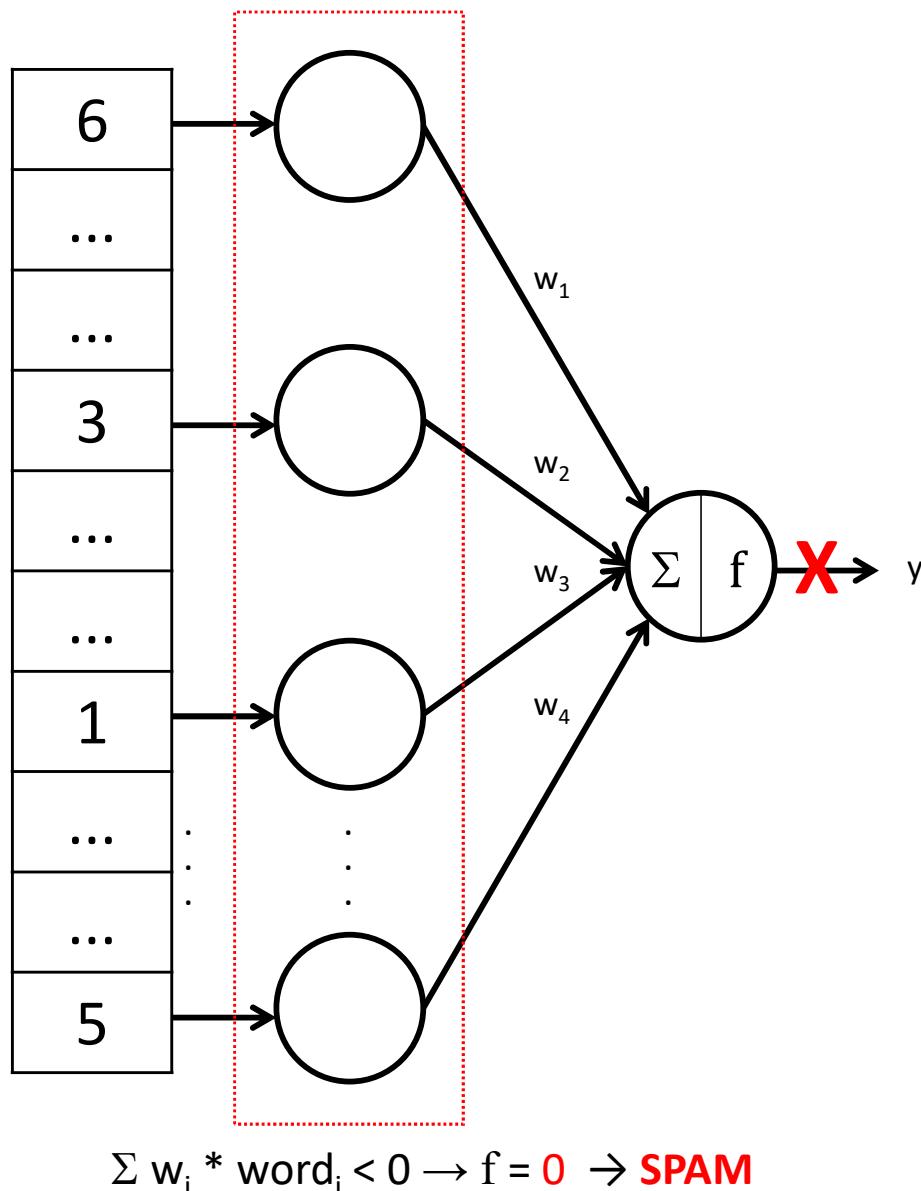
Single-layer Perceptron as a Classifier



Single-layer Perceptron as a Classifier

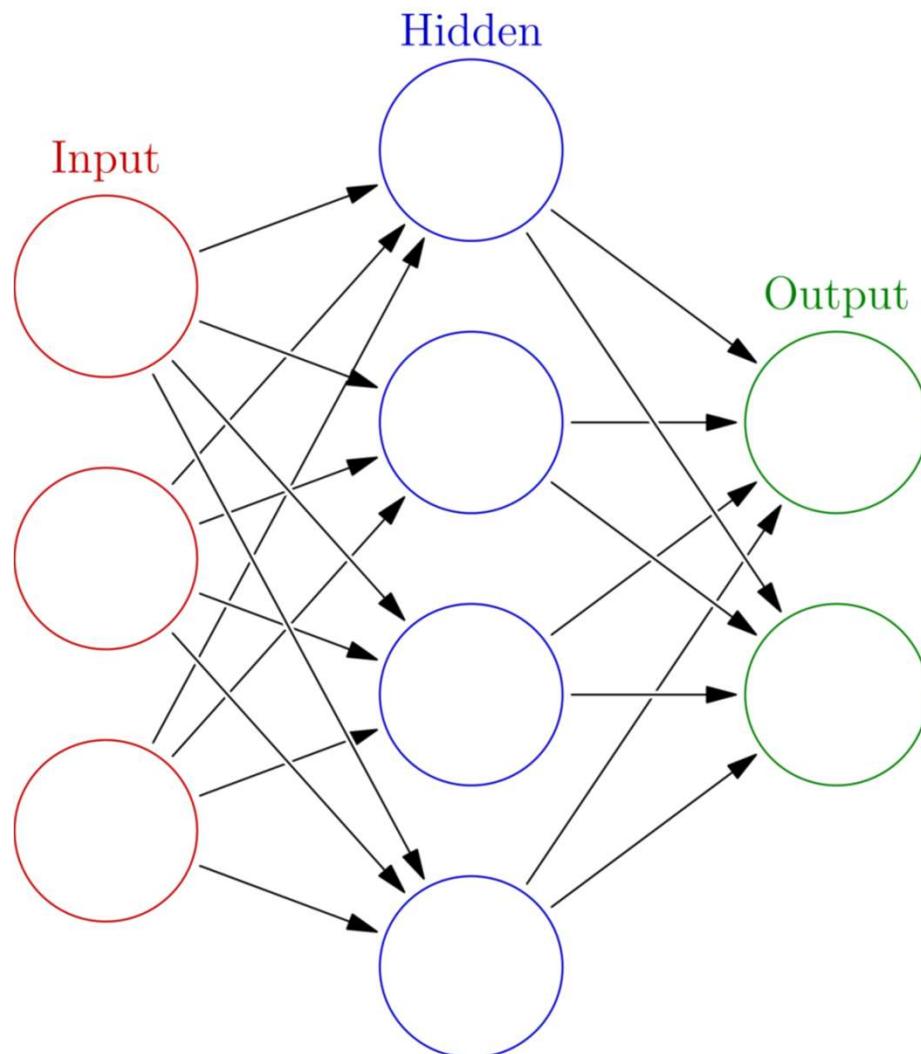


Single-layer Perceptron as a Classifier



Classification with Artificial Neural Networks

Artificial Neural Network (1943)



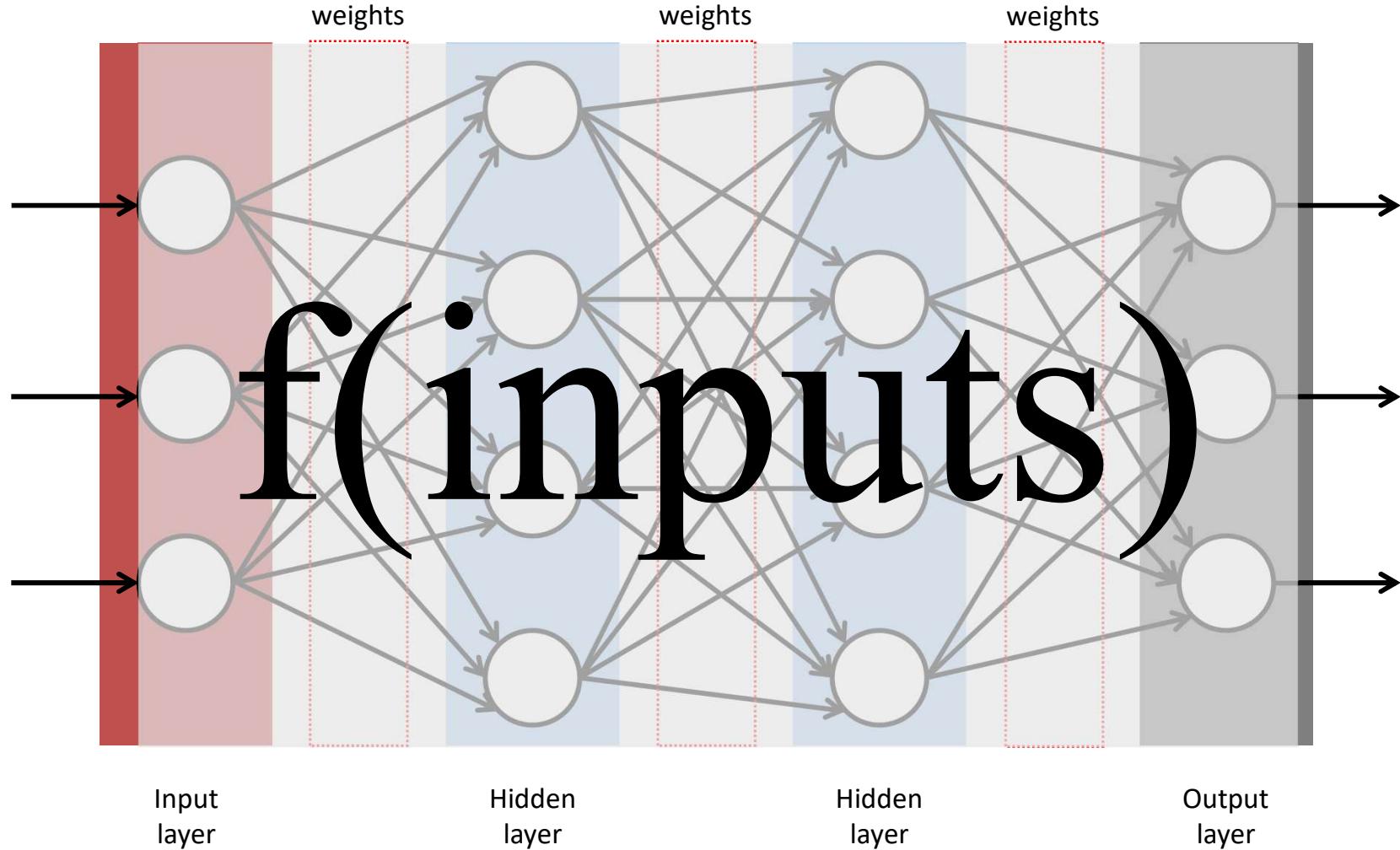
First computational models of an **Artificial Neural Network** (loosely inspired by biological neural networks) were proposed by Warren McCulloch and Walter Pitts in 1943. Their ideas are a **key component of modern day machine and deep learning**.

Source:

https://en.wikipedia.org/wiki/Artificial_neural_network

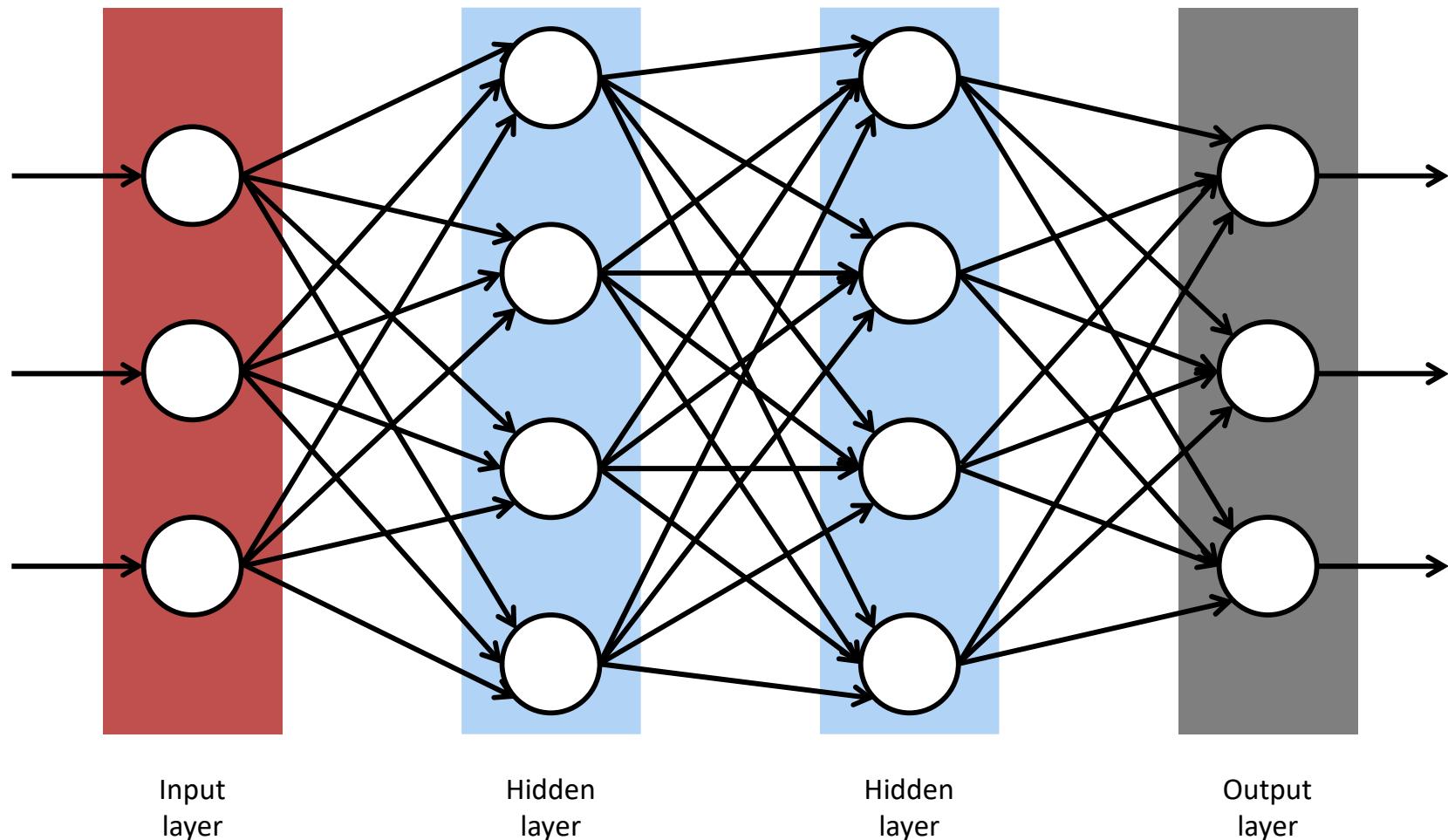
ANN as a Complex Function

In ANNs **hypotheses take form of complex algebraic circuits** with tunable connection strengths (weights).



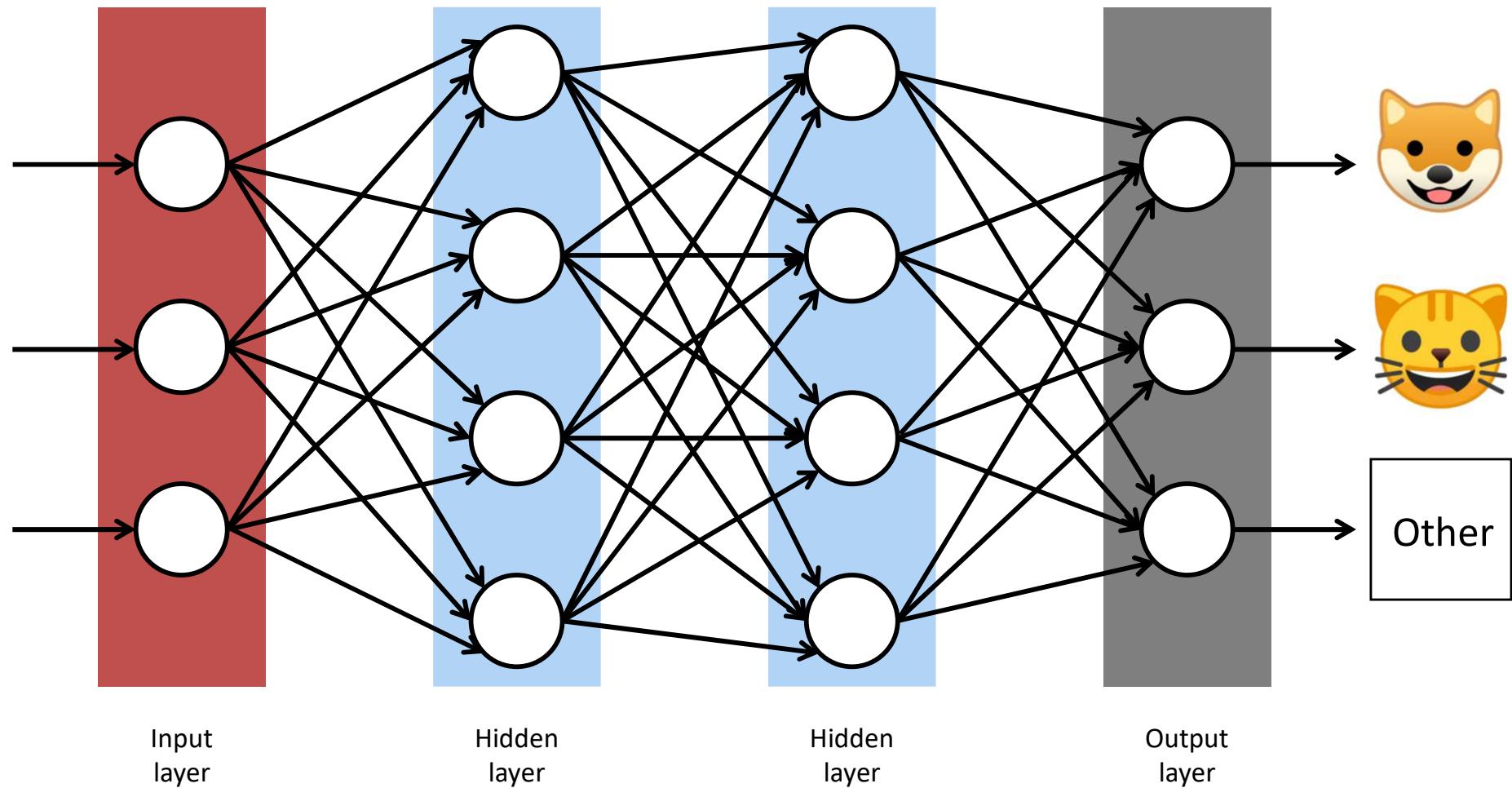
Artificial Neural Network (ANN)

An artificial neural network is made of **multiple artificial neuron layers**.

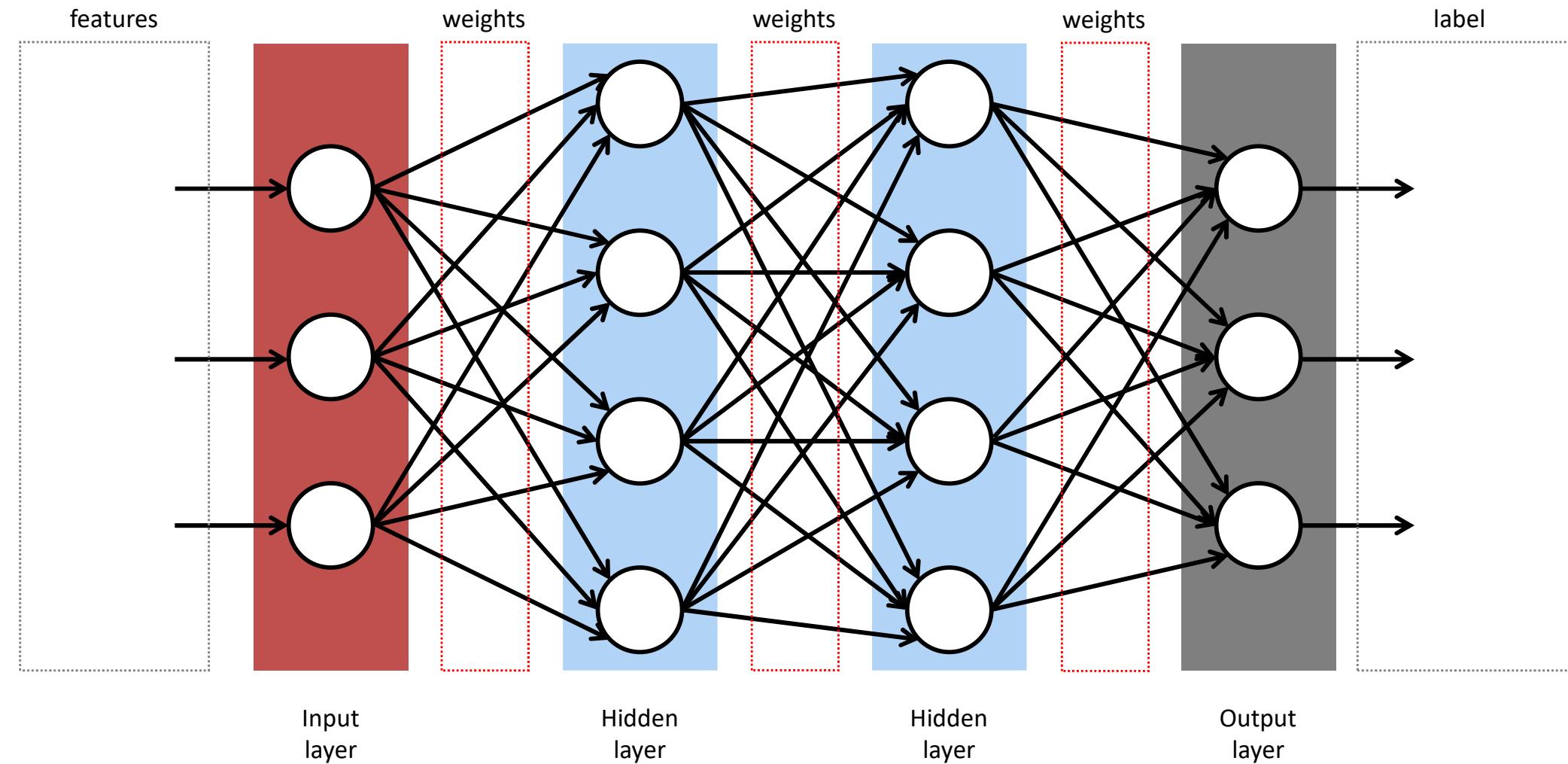


ANN as an Image Classifier

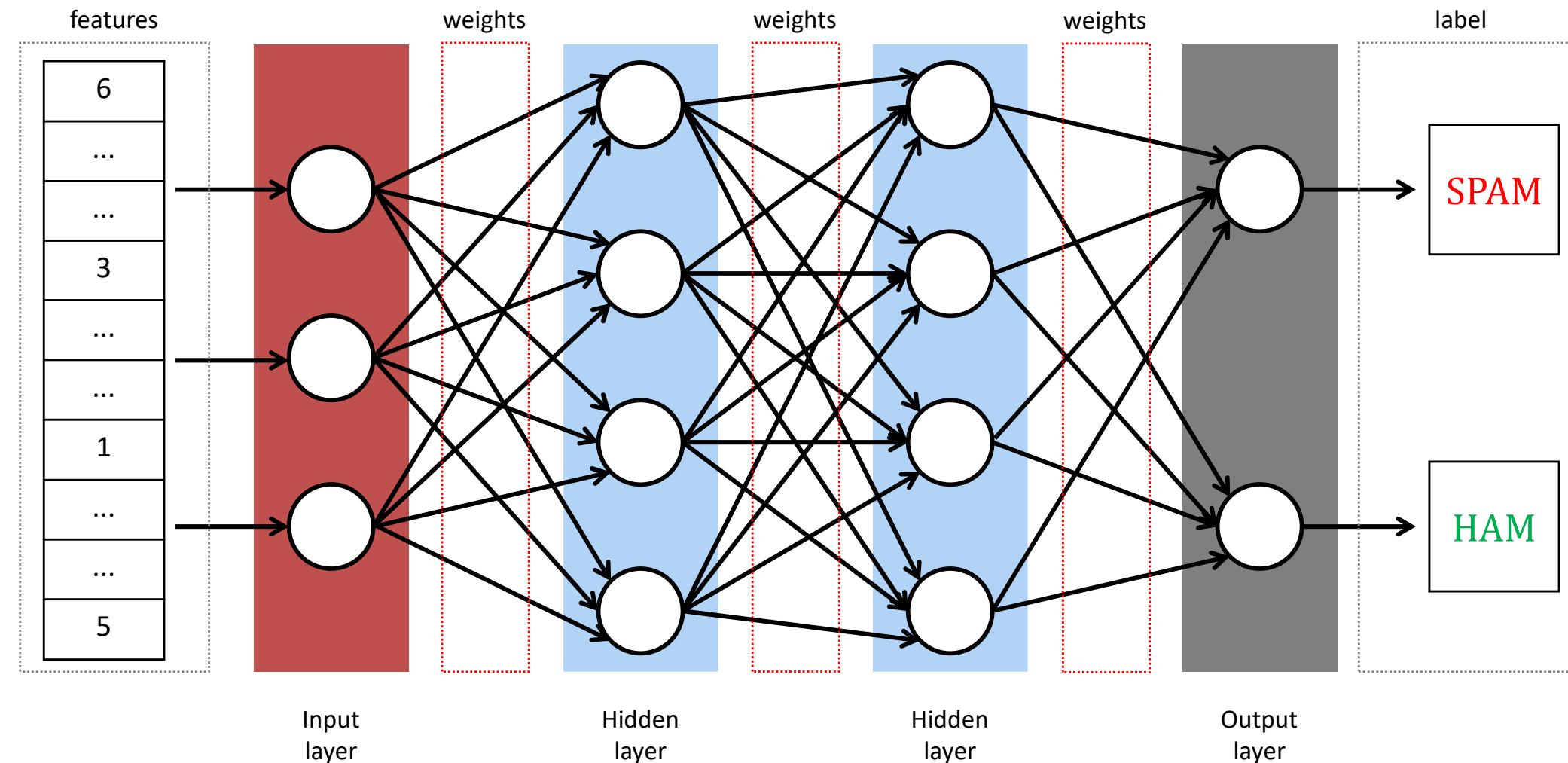
An artificial neural network can be used as a **classifier** as well.



ANN as a General Classifier

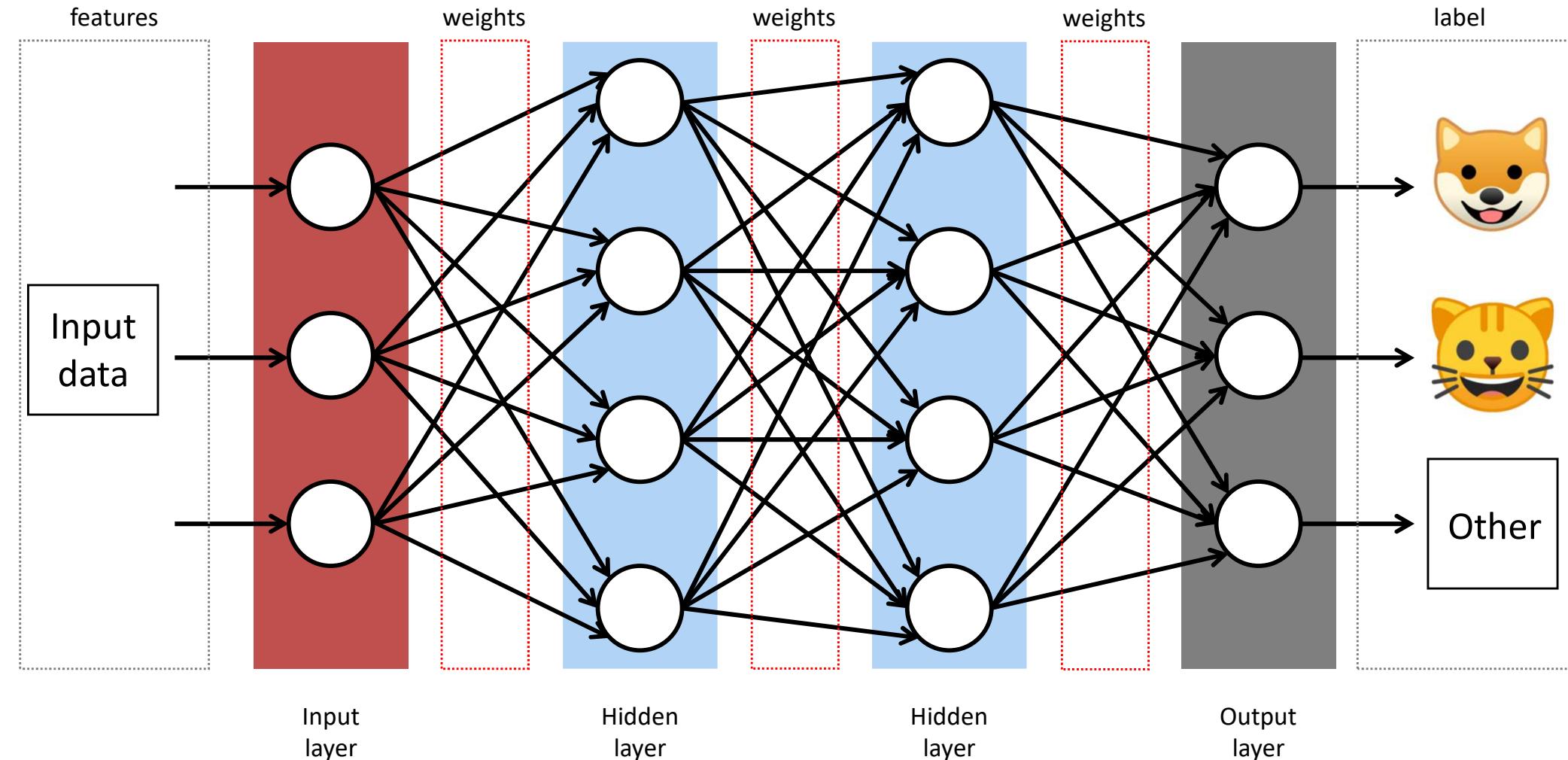


ANN as a Text Classifier



ANN: Supervised Learning

In order to work properly a classifier **needs to be trained** first with **labeled data**.



Training will adjust all the weights within this artificial neural network.

Training Data: Features + Labels

Typically input data will be represented by a **limited set of features**.



Features:
Wheels: 4
Weight: 8 tons
Passengers: 1

Label:
Truck



Features:
Wheels: 6
Weight: 8 tons
Passengers: 1

Label:
Truck



Features:
Wheels: 4
Weight: 1 ton
Passengers: 4

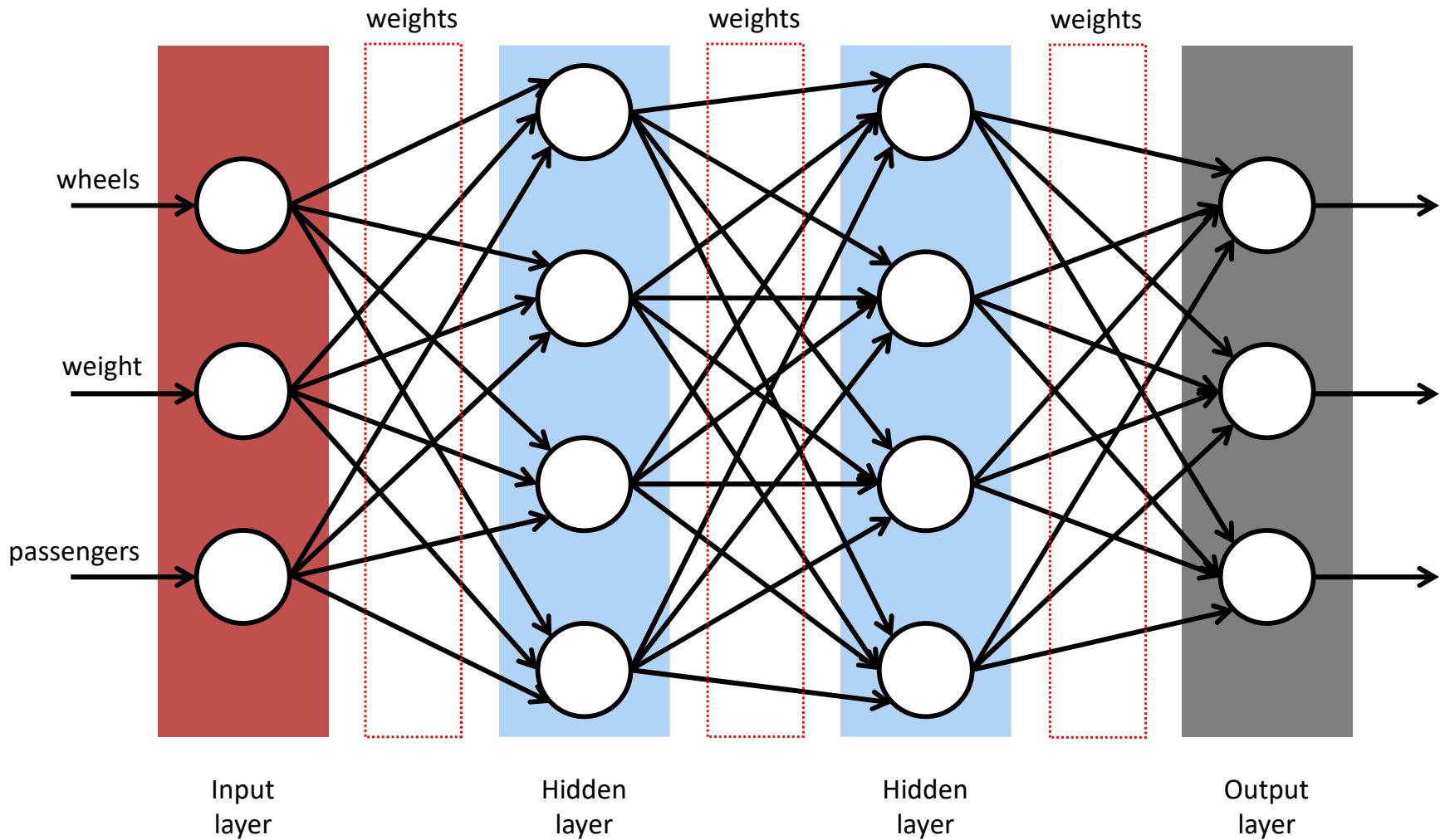
Label:
Car



Features:
Wheels: 4
Weight: 2 tons
Passengers: 4

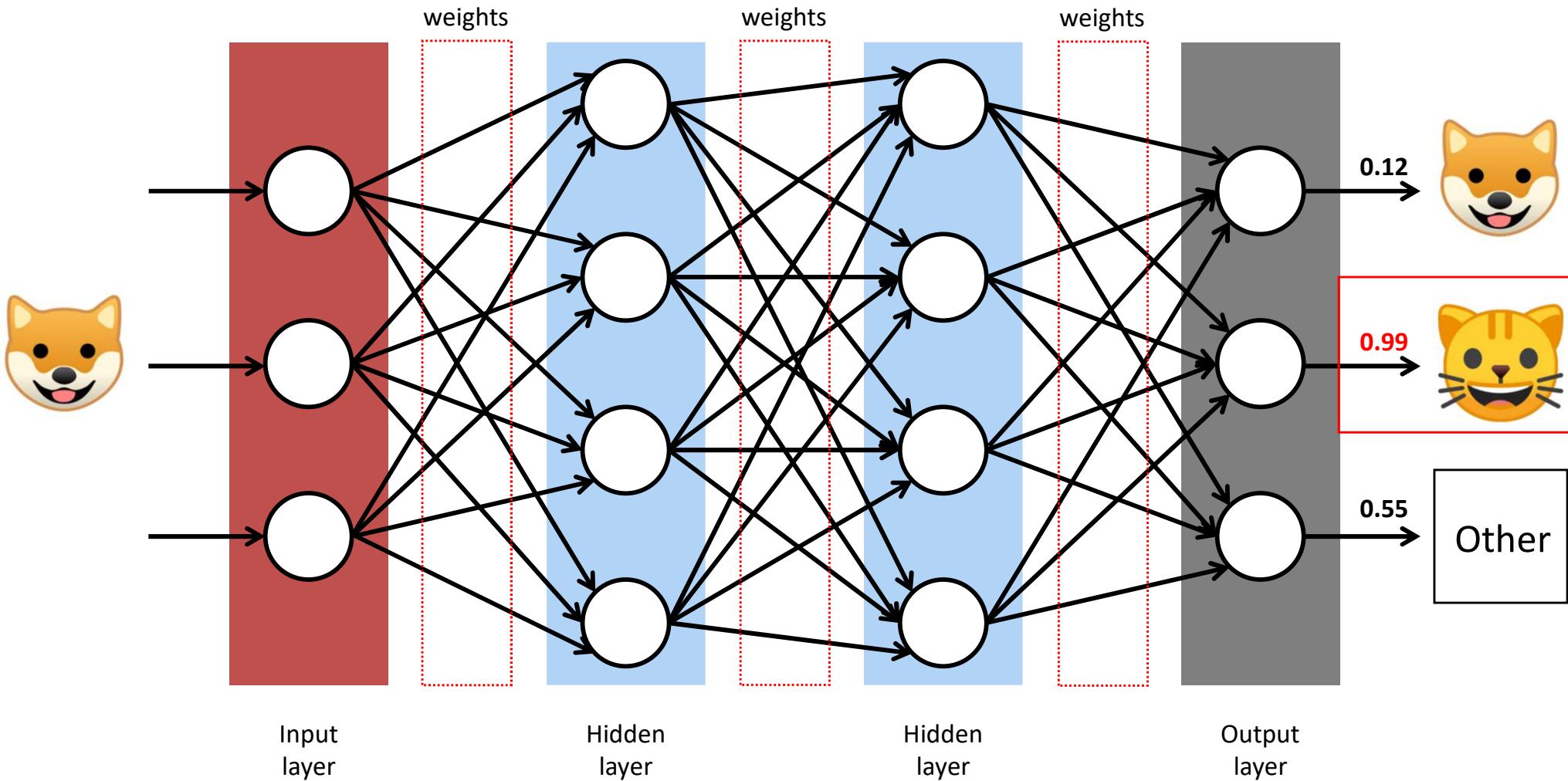
Label:
Car

ANN: Supervised Learning



ANN: Supervised Learning

An **untrained classifier** will **NOT** label input data correctly.

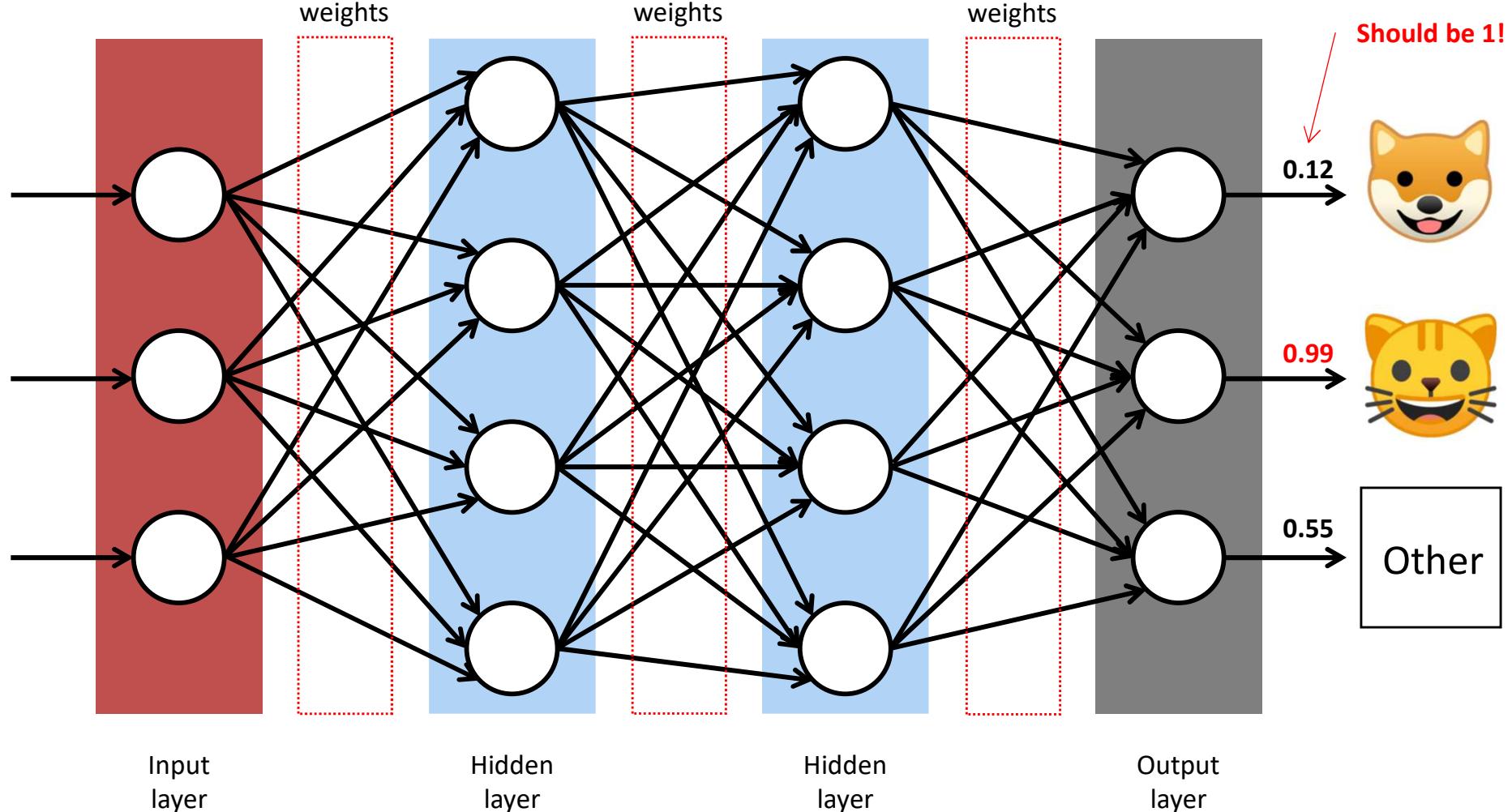


ANN: Training

Given: input d 

and it's corresponding **expected** label: DOG calculate

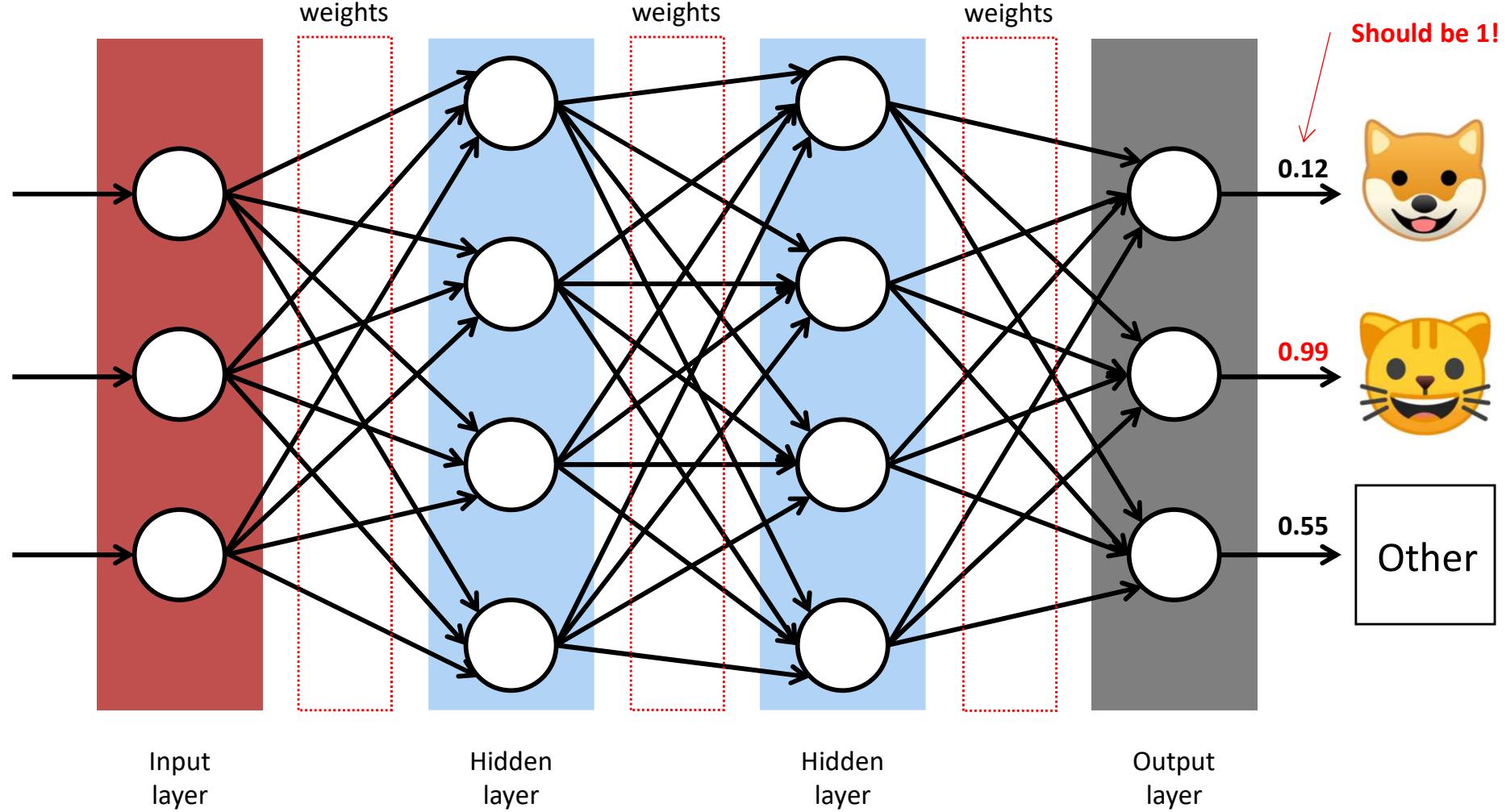
"error".



"Error" = 0.88. Go back and **adjust all the weights** to ensure it is lower next time.

ANN: Training

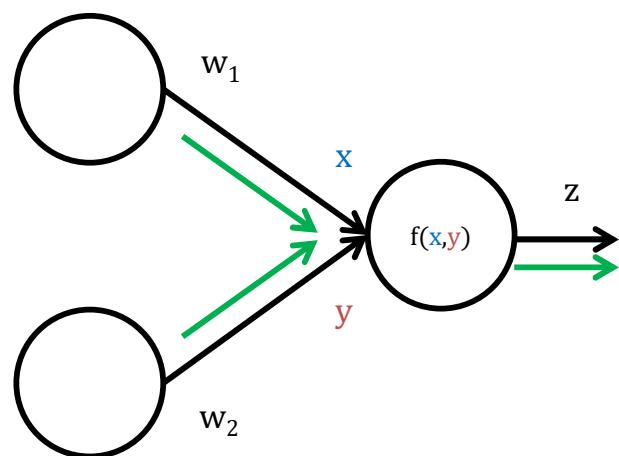
Show data / label pair:  / DOG.



← **Correct all the weights.** Repeat many times.

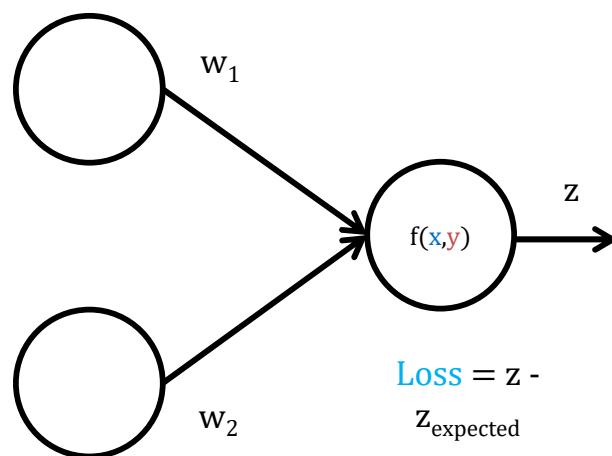
Back-propagation

Feed forward



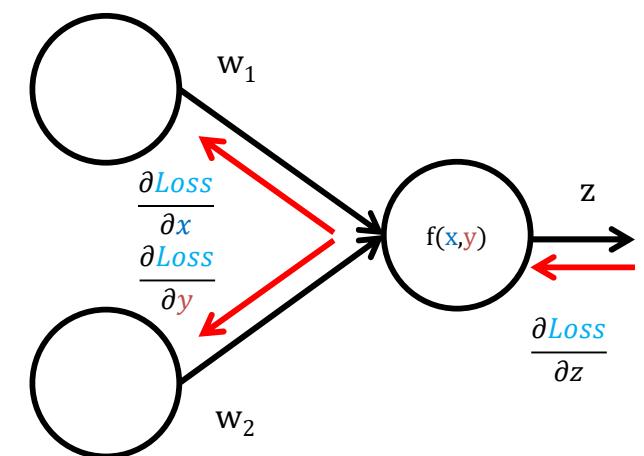
Feed a **labeled sample** through the network

Evaluate Loss



How “incorrect” is the result compare to the label?

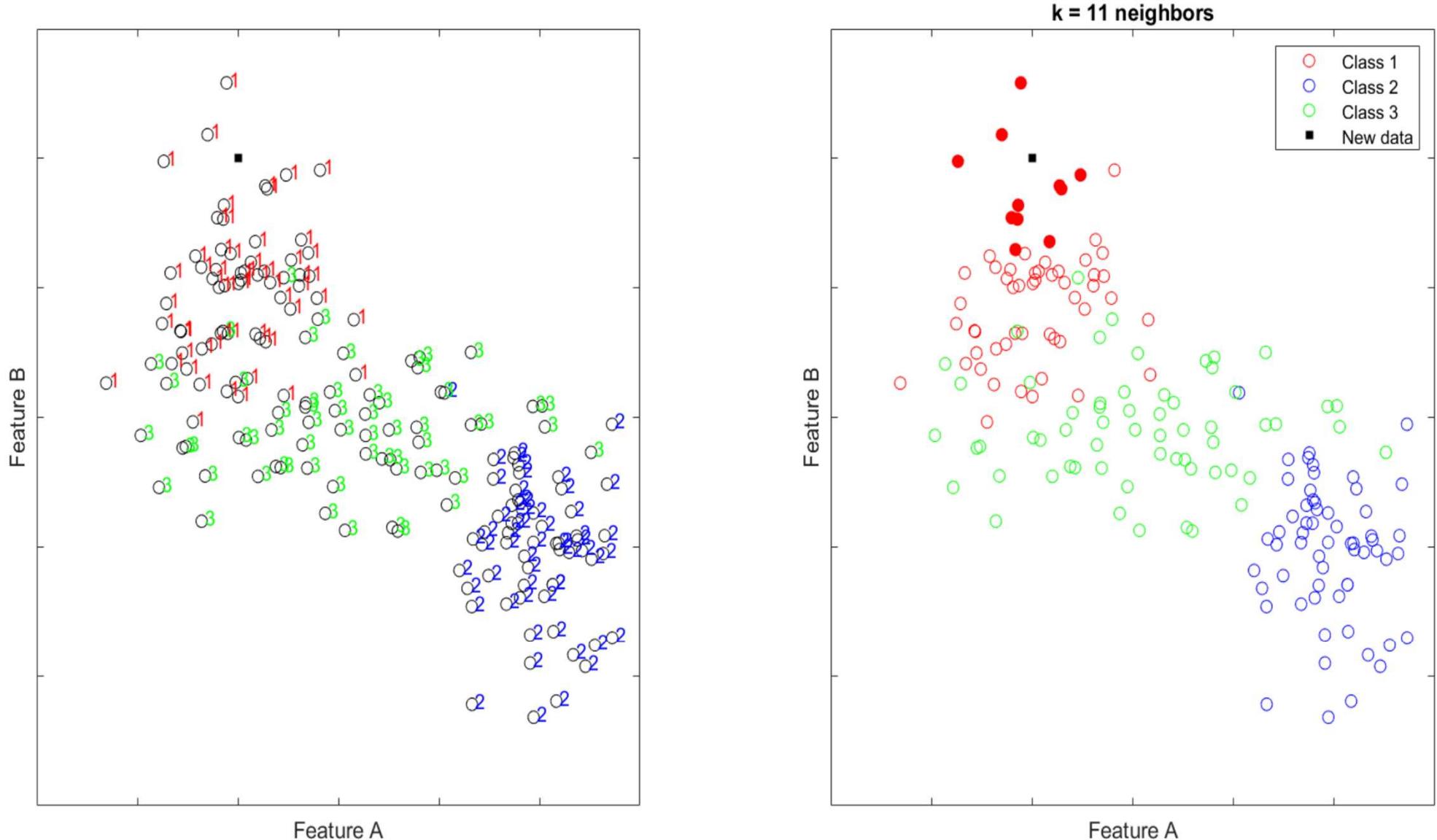
Back-propagation



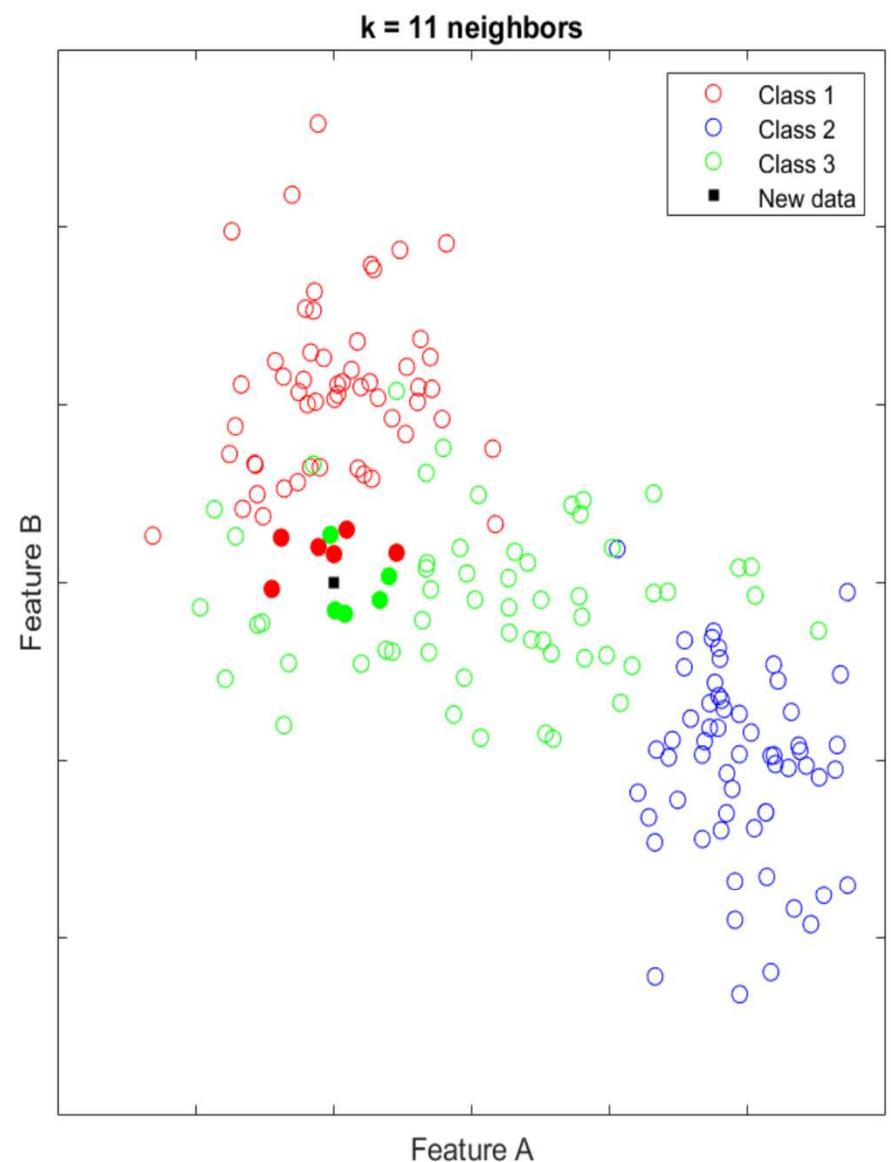
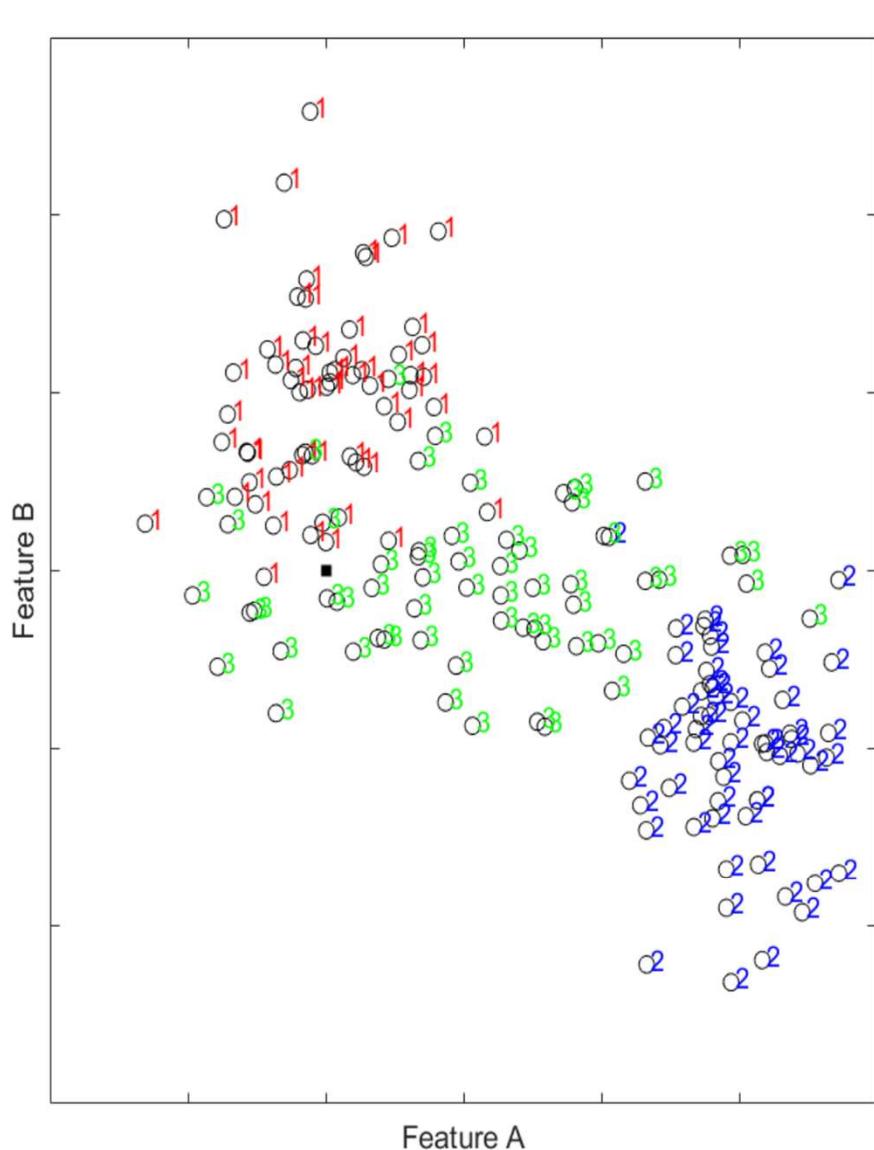
Update weights
(use **Gradient Descent**)

Classification with k-Nearest Neighbors

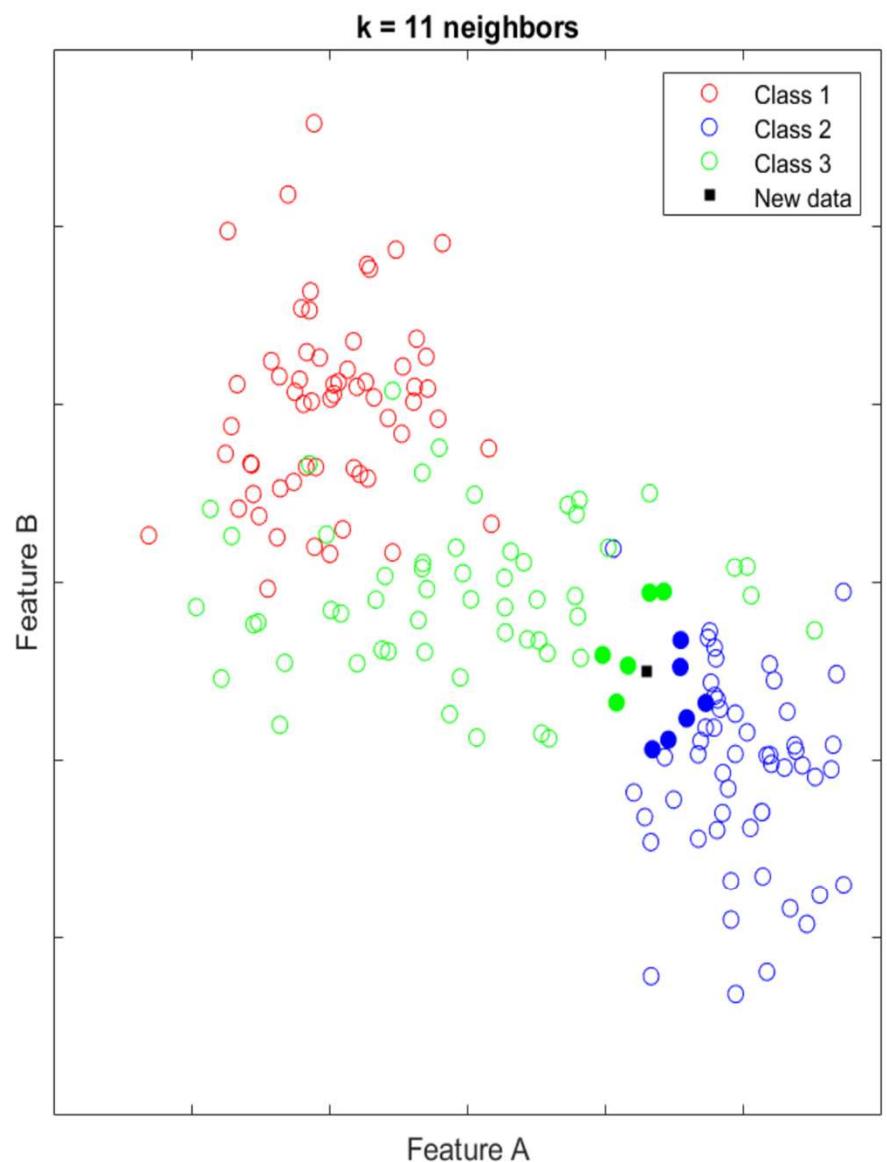
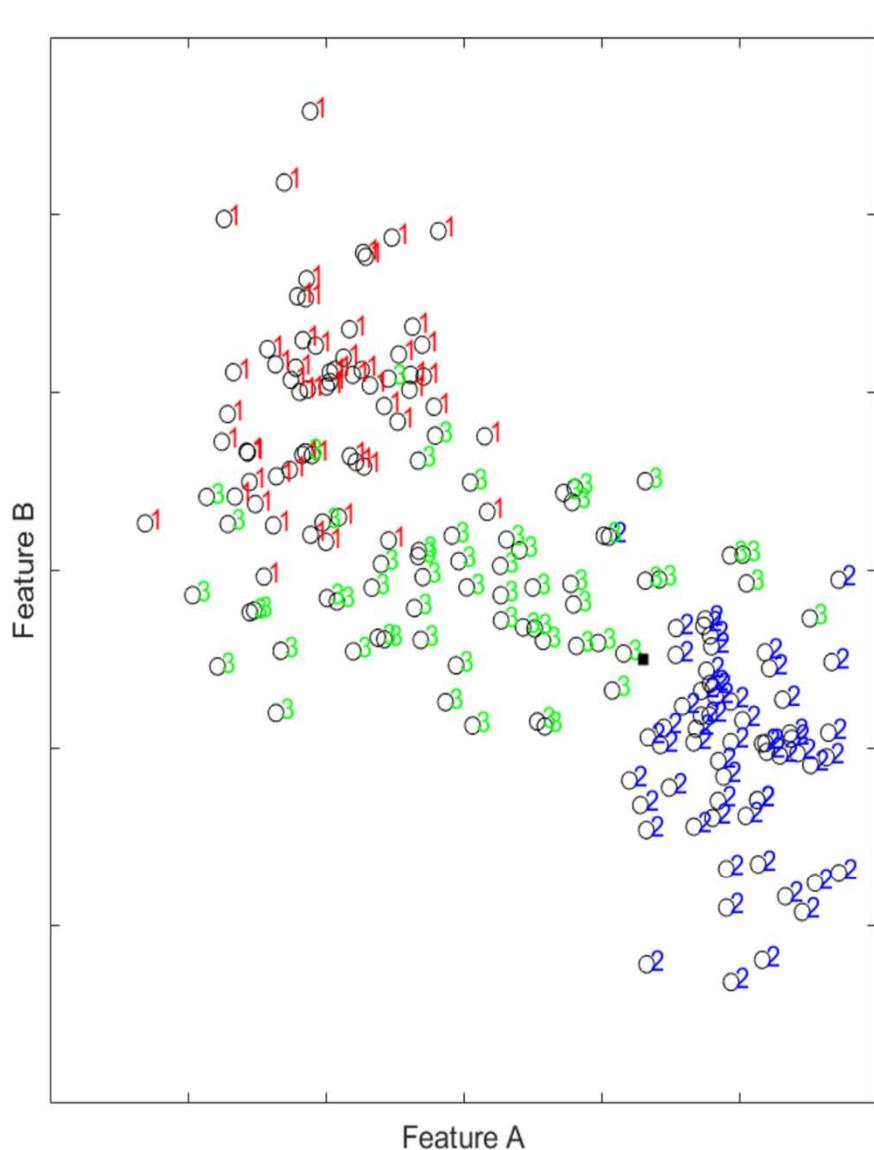
$k = 11$ Nearest Neighbors



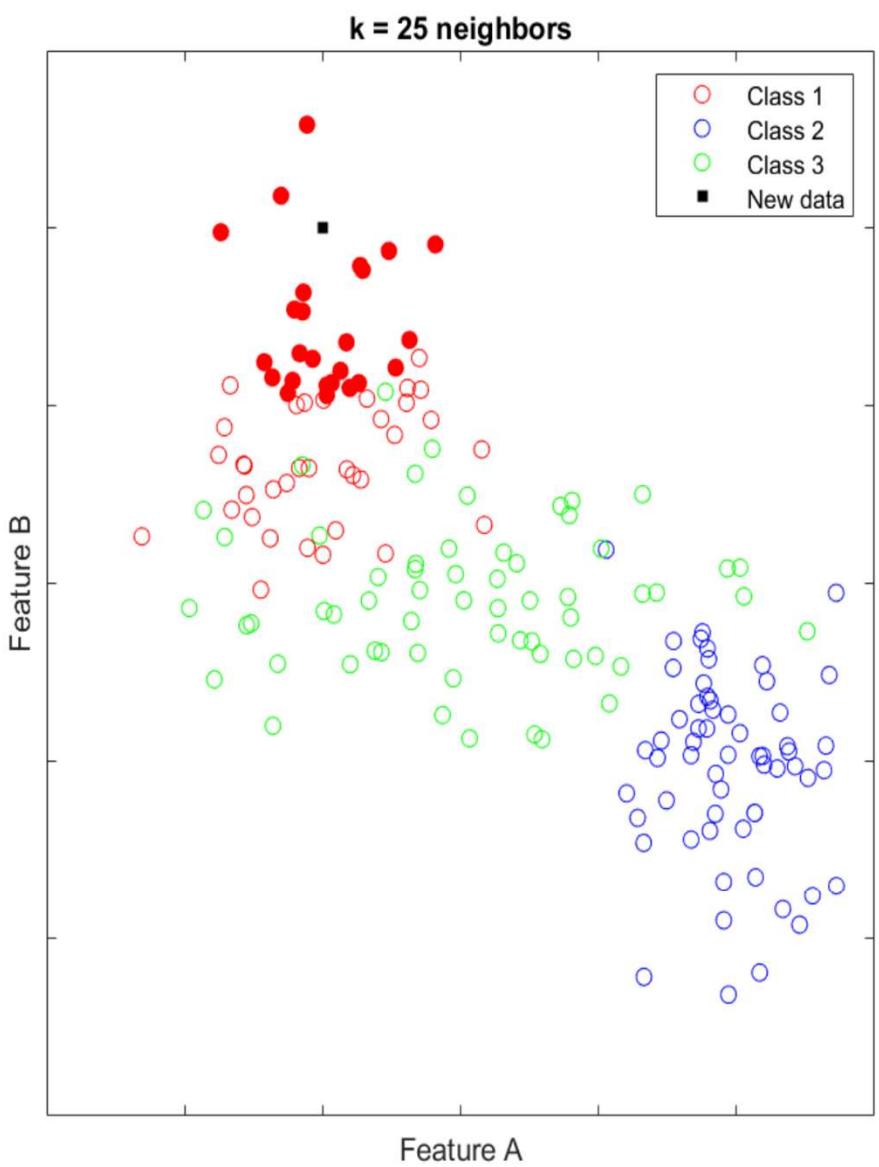
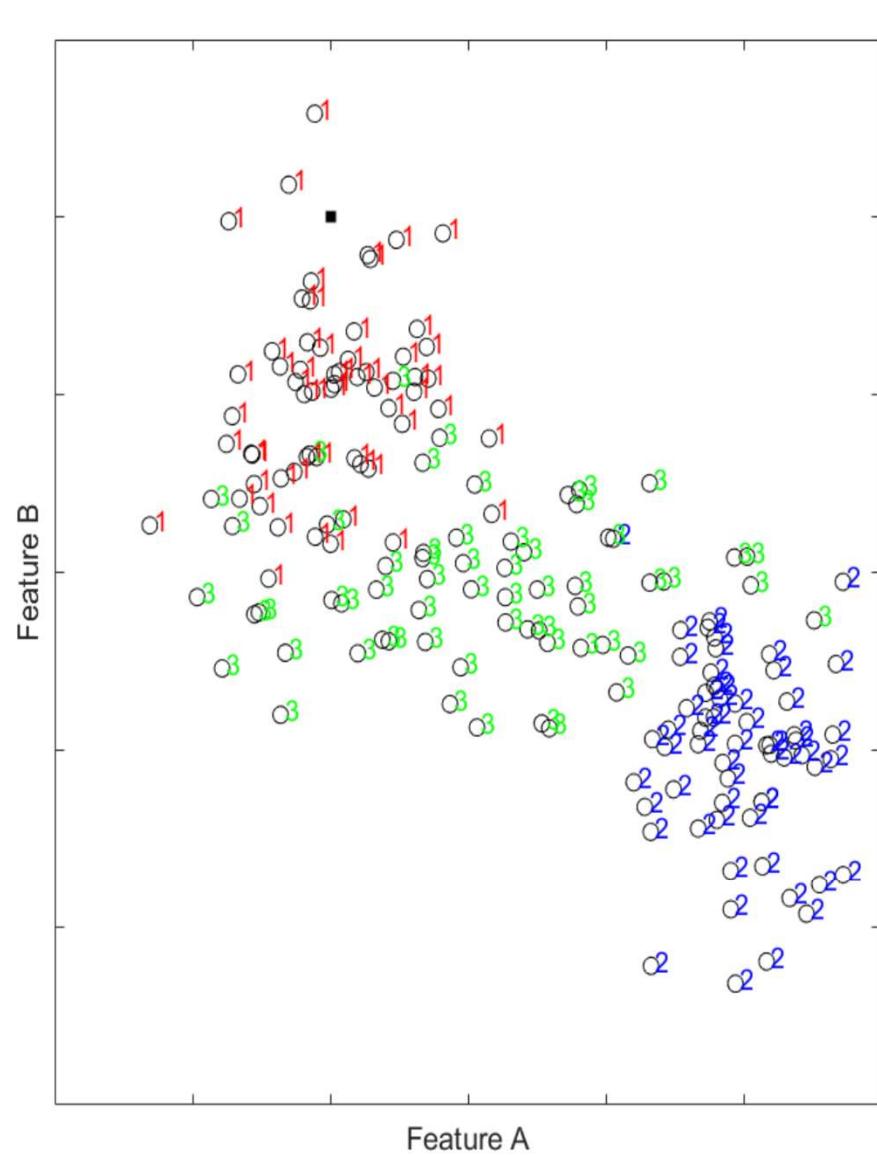
$k = 11$ Nearest Neighbors



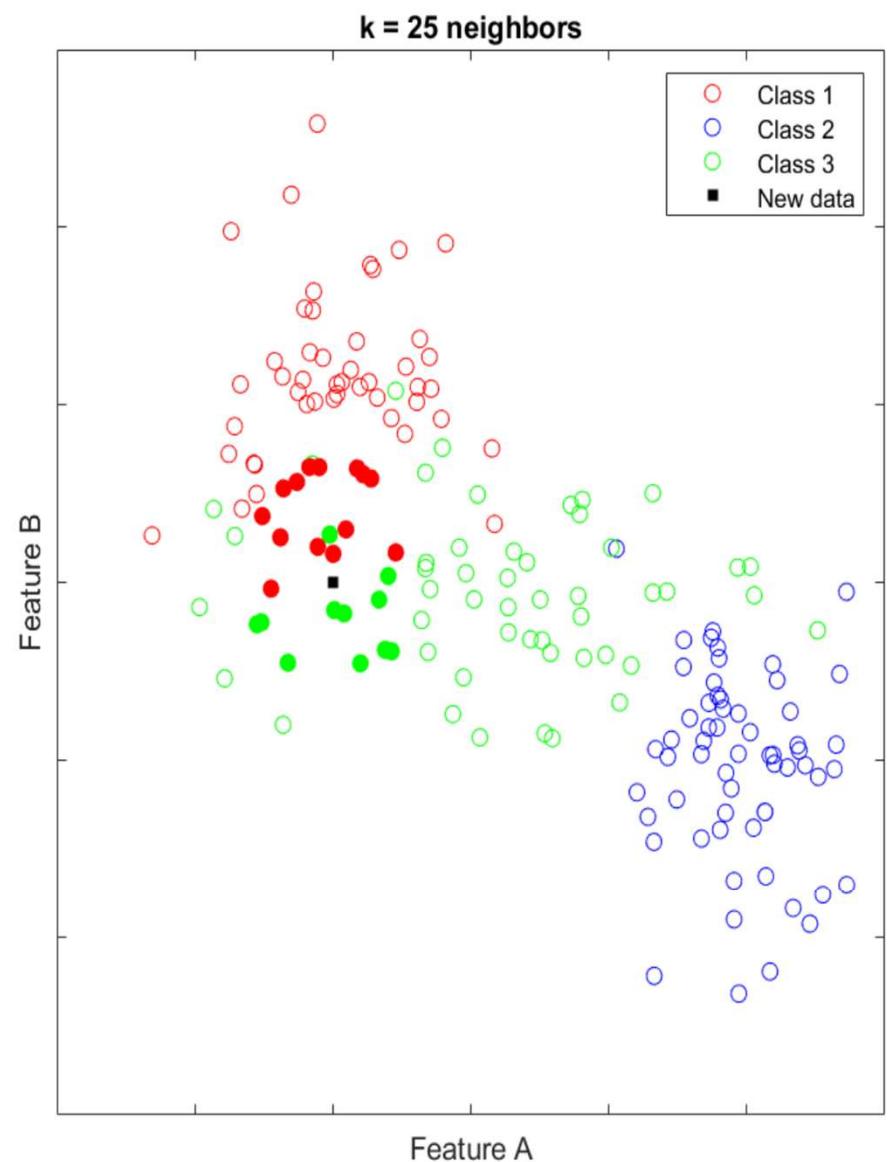
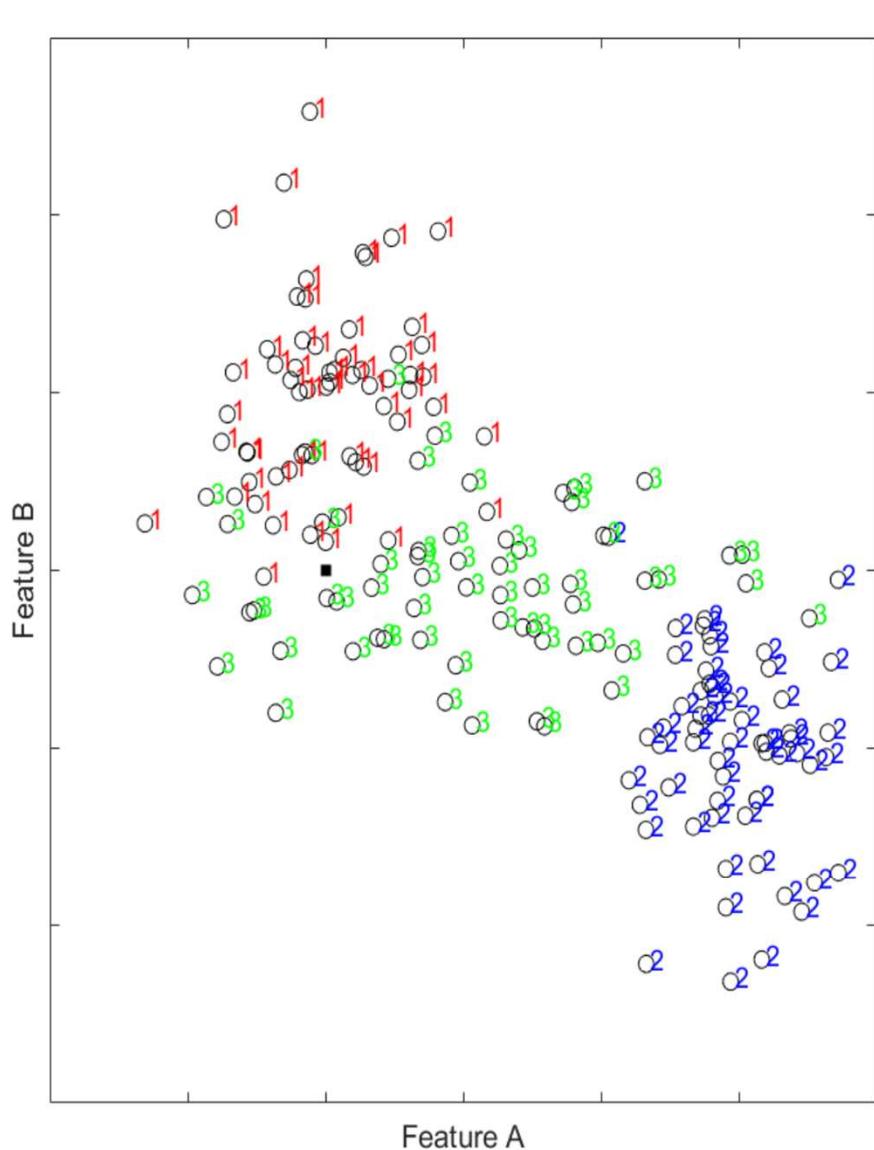
$k = 11$ Nearest Neighbors



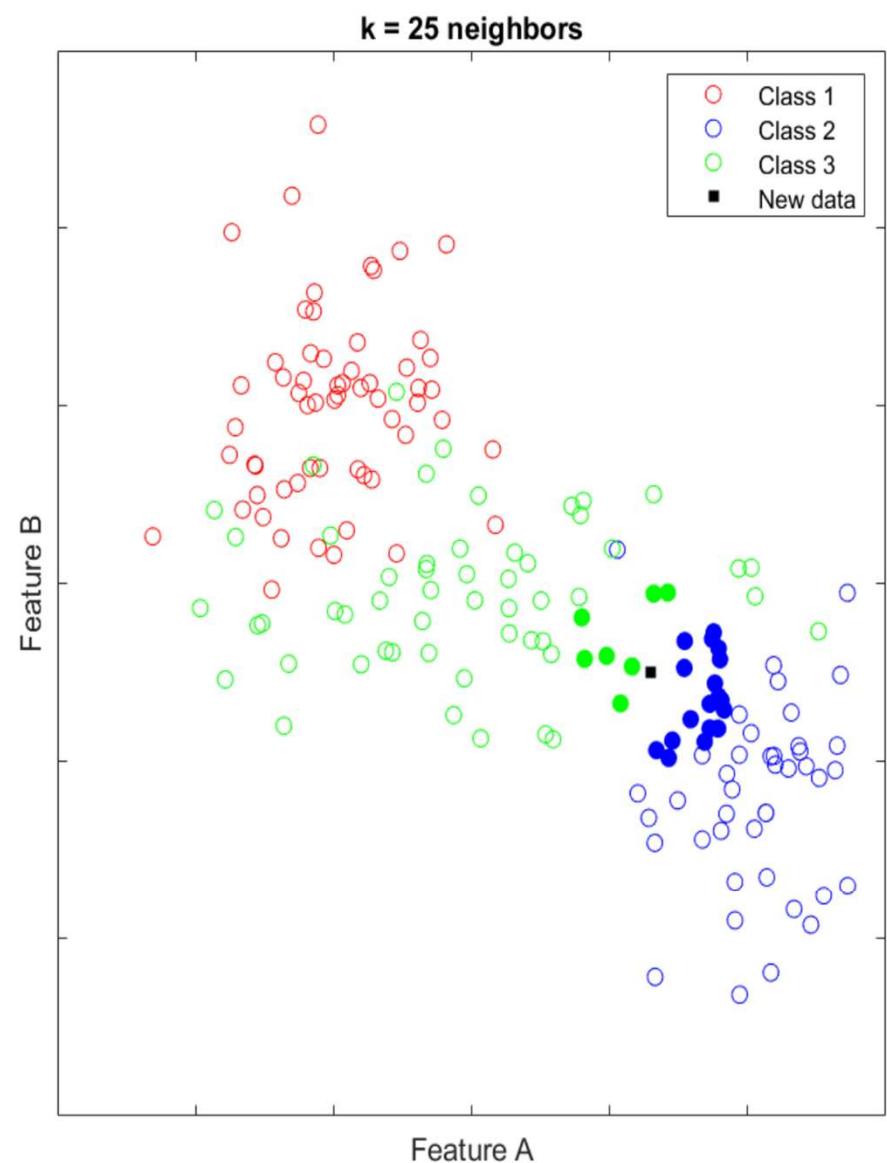
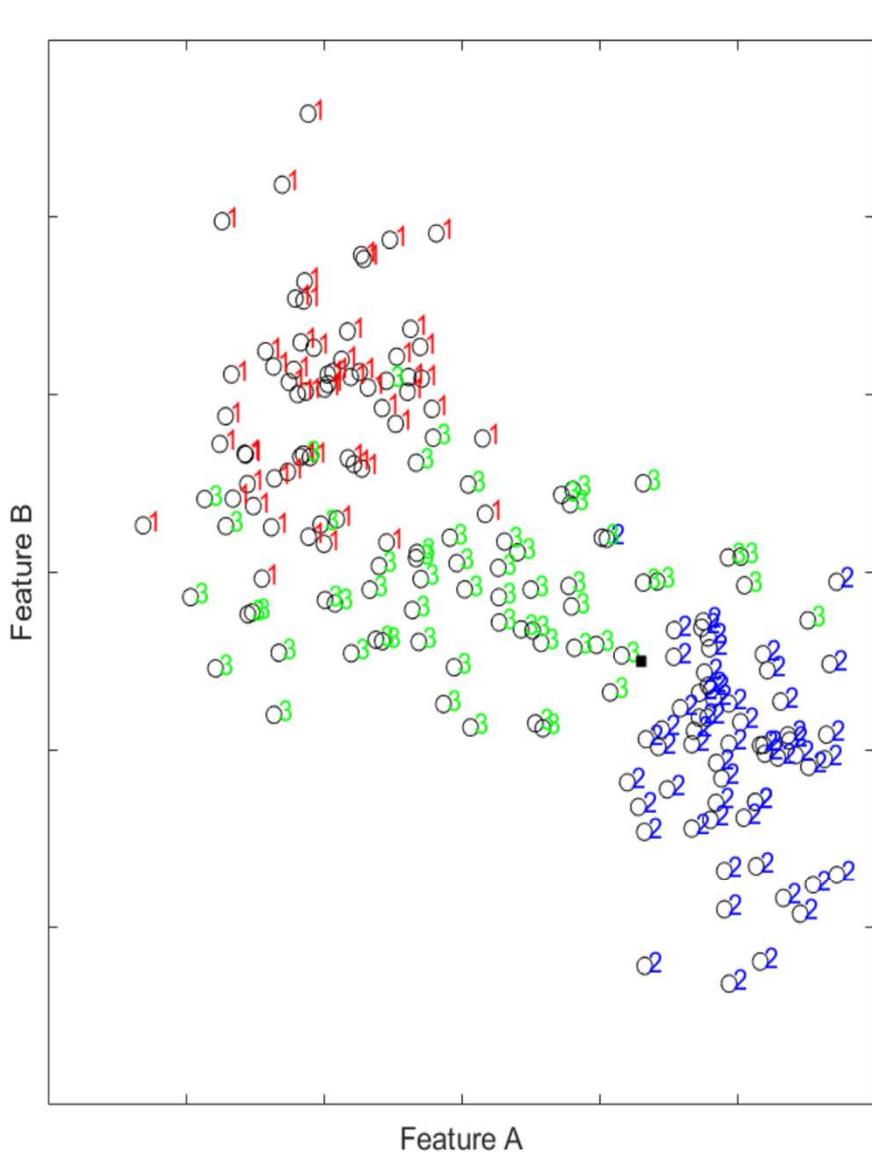
$k = 25$ Nearest Neighbors



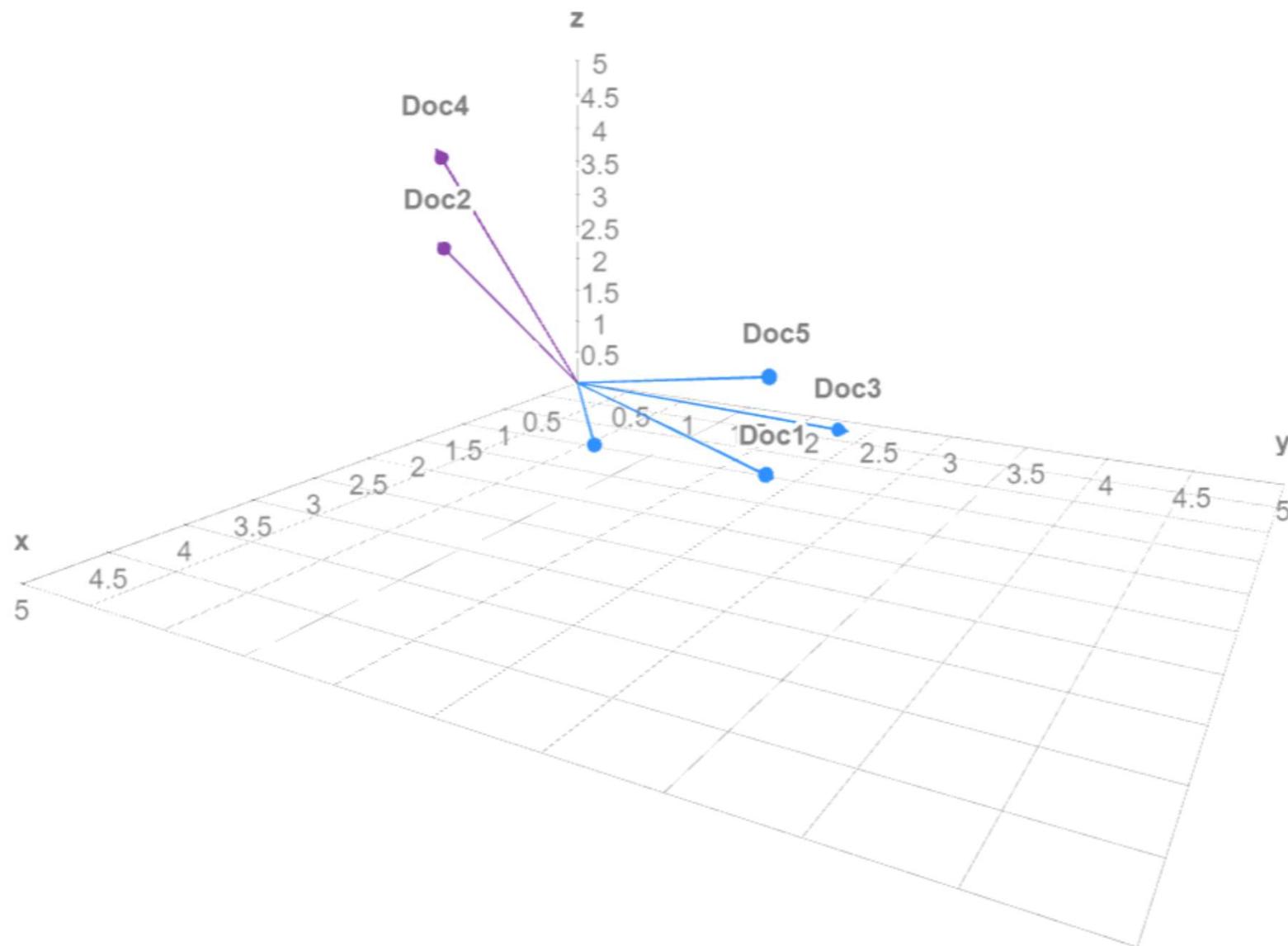
$k = 25$ Nearest Neighbors



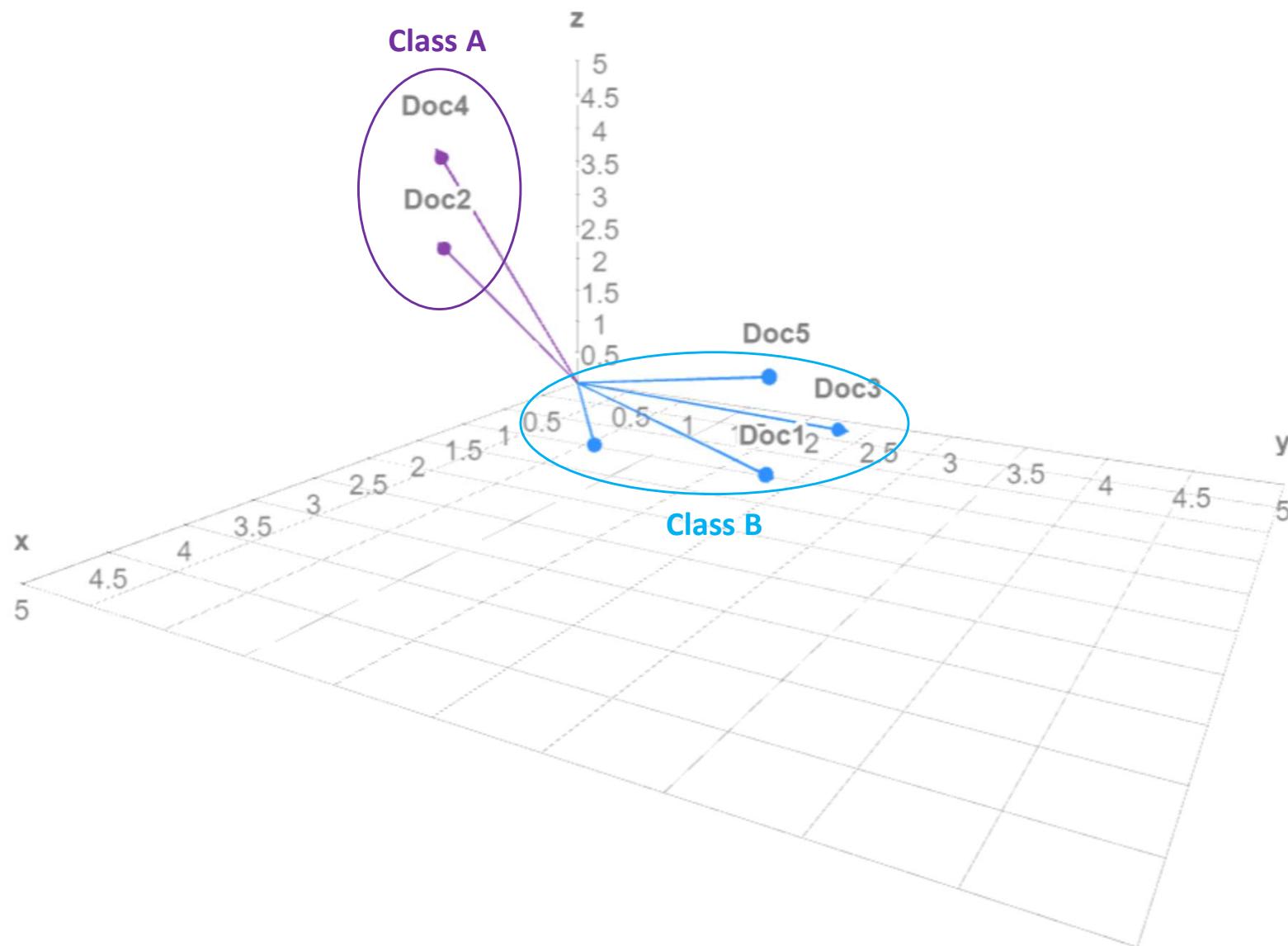
$k = 25$ Nearest Neighbors



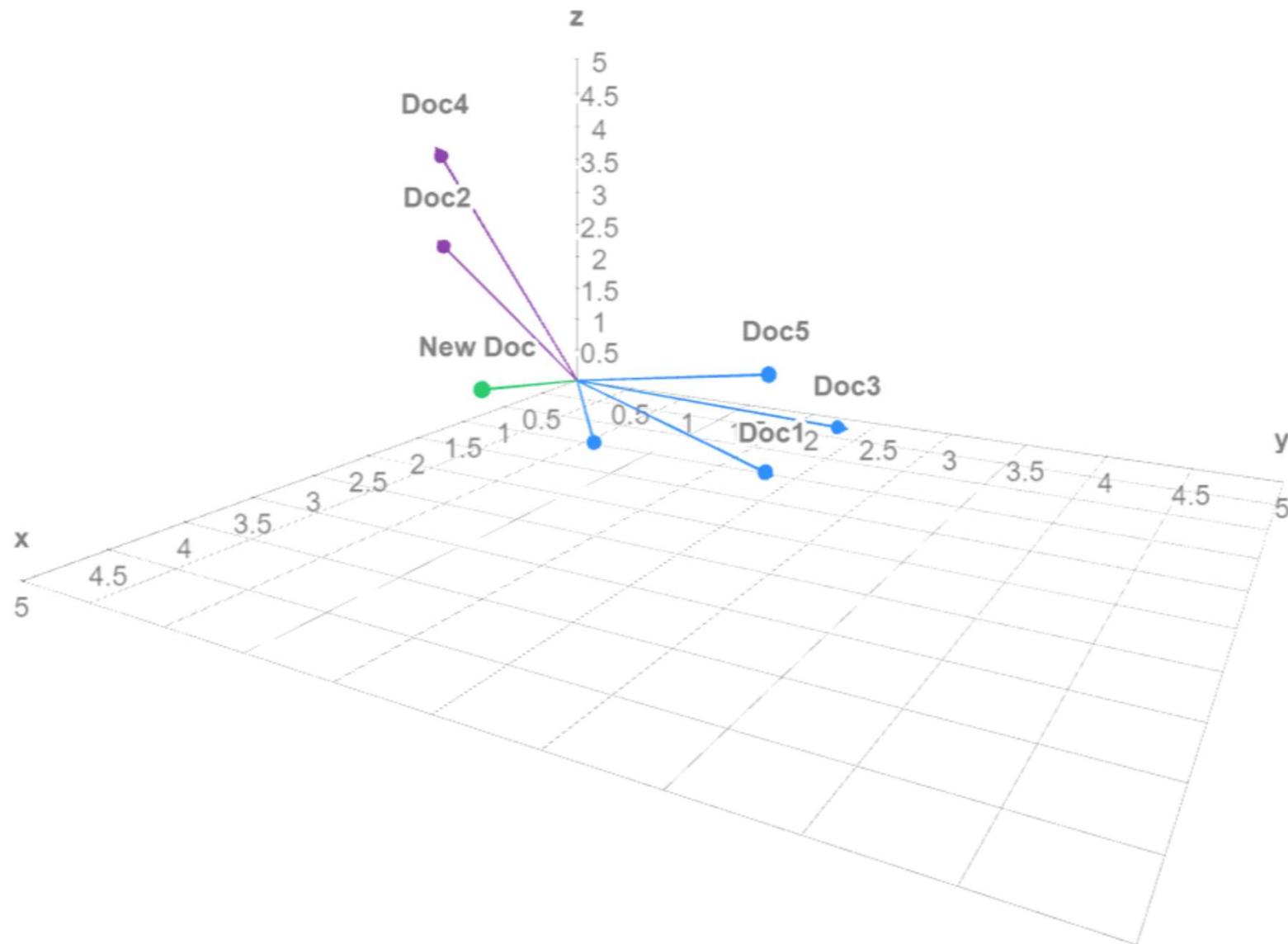
kNN: Text Classification



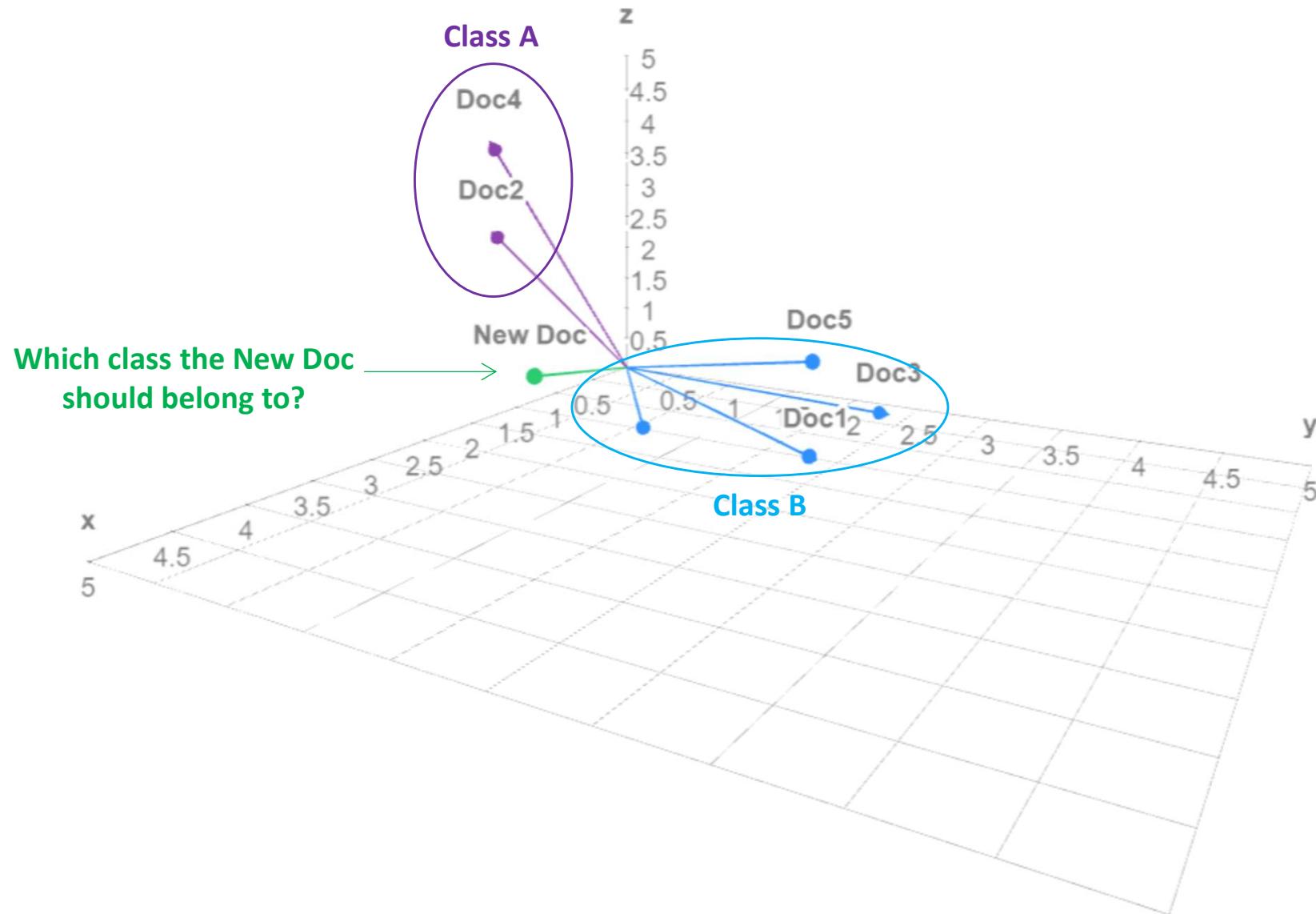
kNN: Text Classification



kNN: Text Classification

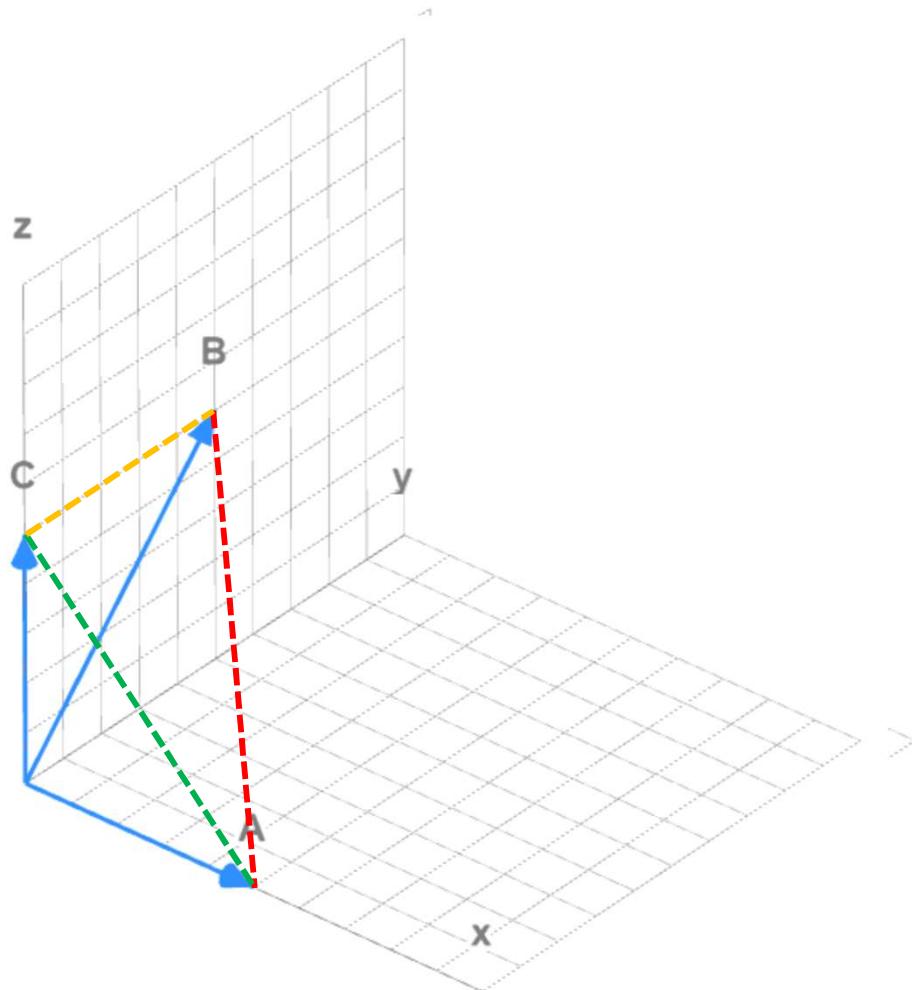


kNN: Text Classification



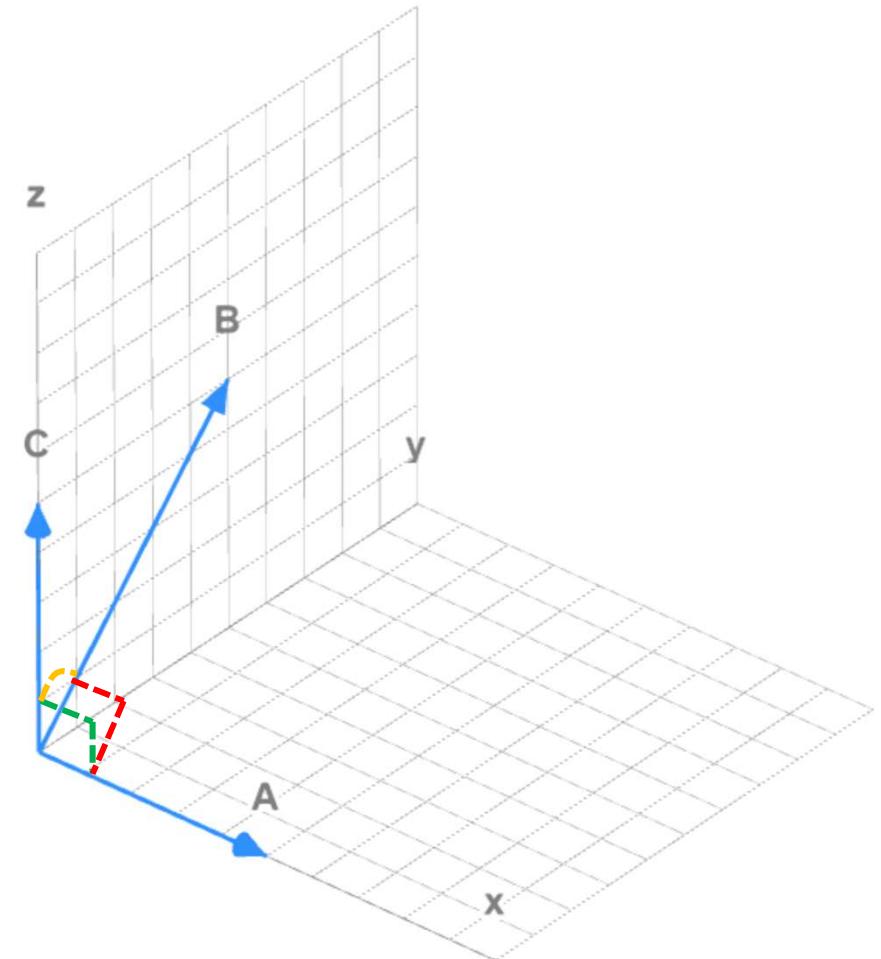
Distance Measures

Euclidean distance



$$D(x, y) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2}$$

Cosine similarity



$$D(x, y) = \cos(\theta) = \frac{x \cdot y}{\|x\| \|y\|}$$

Word Vectors: Potential Problems

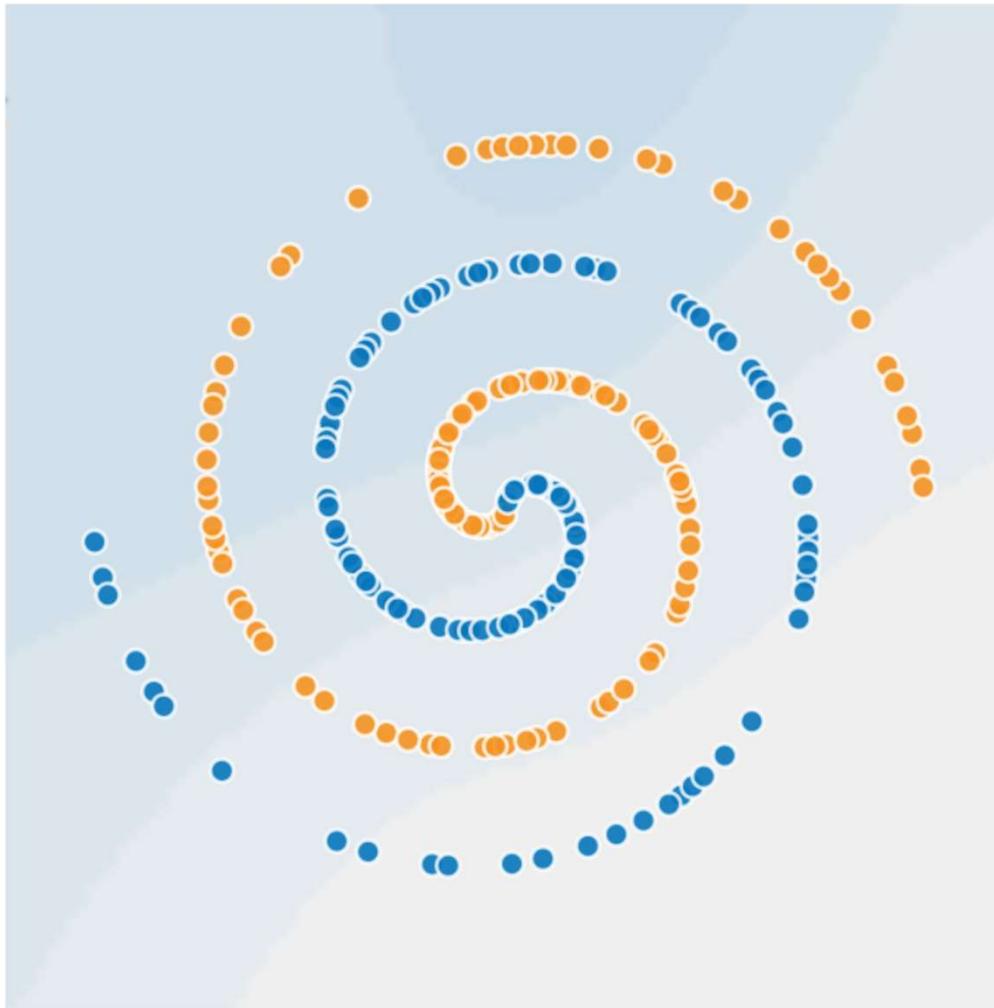
$$\text{cat} = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}, \text{dog} = \begin{bmatrix} 0 \\ 1 \\ \vdots \\ 0 \\ 0 \end{bmatrix}, \text{mouse} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \\ 0 \end{bmatrix}, \text{bird} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}$$

$$\text{CosineSimilarity}(\text{cat}, \text{dog}) = \frac{\mathbf{v}_{\text{cat}} \cdot \mathbf{v}_{\text{dog}}}{\|\mathbf{v}_{\text{cat}}\| \|\mathbf{v}_{\text{dog}}\|} = 0$$

$$D_1 = \text{"the cat and the mouse"} = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 1 \\ 0 \end{bmatrix}, D_2 = \text{"the cat and the bird"} = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}$$

$$\text{CosineSimilarity}(D_1, D_2) = \frac{D_1 \cdot D_2}{\|D_1\| \|D_2\|} = \frac{1}{\sqrt{2}\sqrt{2}} = \frac{1}{2}$$

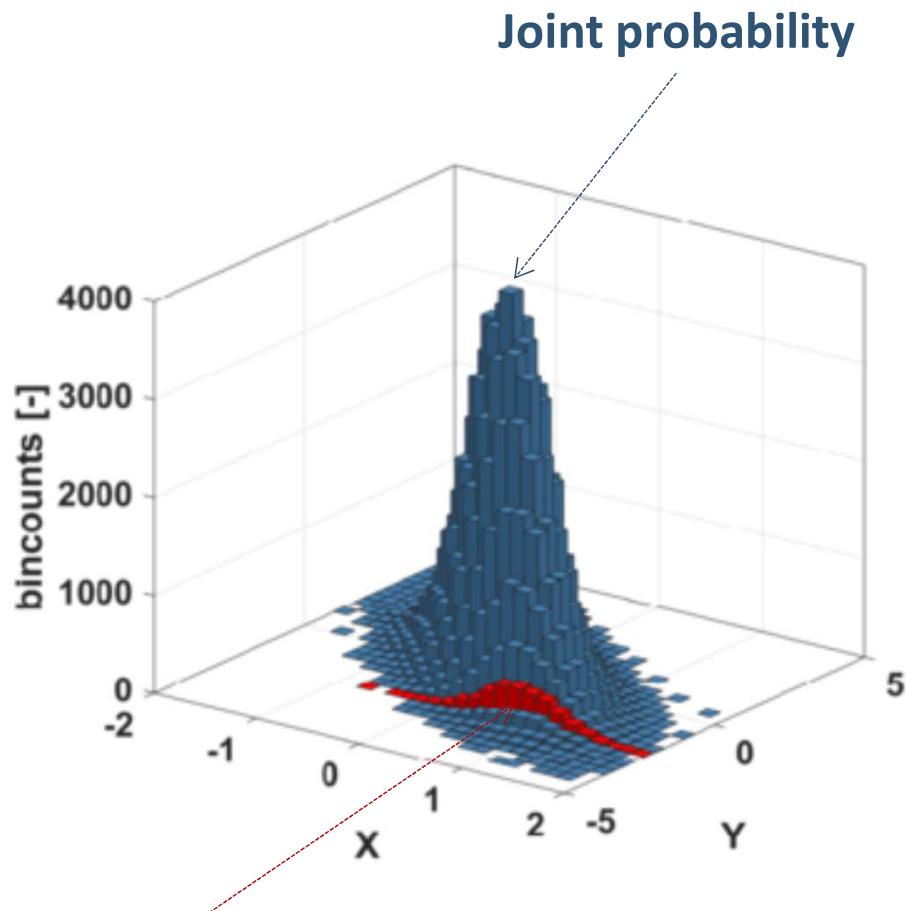
How Would kNN Do Here?



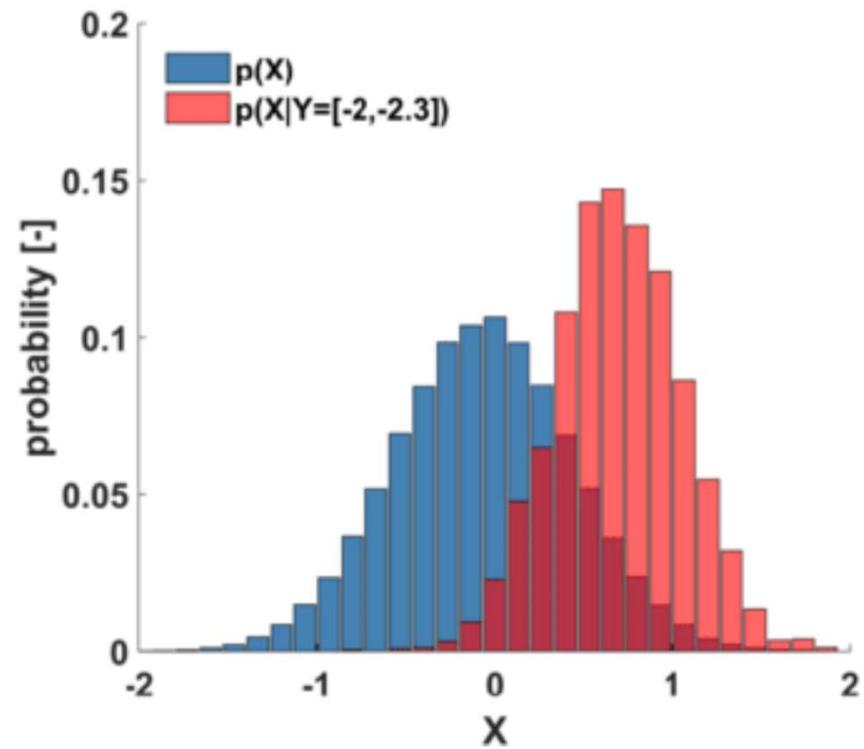
Text Classification: Supervised ML

- Various Machine Learning supervised learning classifier approaches can be employed:
 - Naïve Bayes
 - Logistic regression
 - Neural networks
 - k-Nearest Neighbors
 - etc.

Joint vs Conditional Probability



Conditional
probability



Source: https://www.researchgate.net/figure/Illustration-of-a-bivariate-conditional-probability-distribution-as-a-simple-data-based_fig1_330092567

Classification: Key Question

Given a document (email, tweet, etc.):



which category / class does it belong to?

Text Classification: Supervised ML

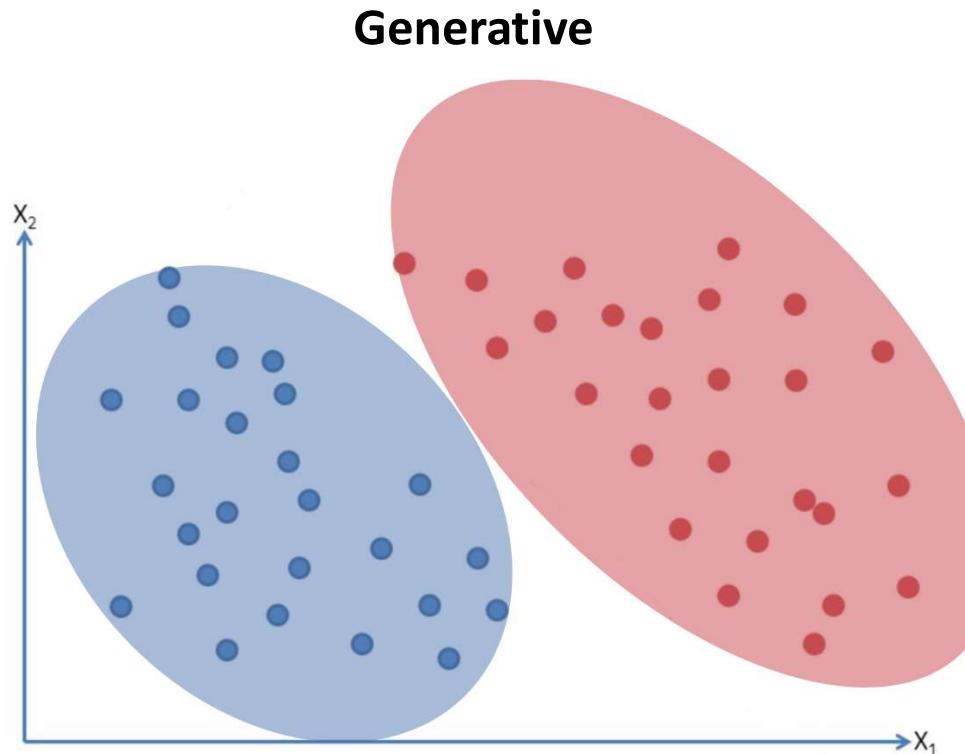
Input:

- a document \mathbf{x}
- a fixed set of classes $Y = \{y_1, y_2, \dots, y_J\}$
- a training set of N hand-labeled documents $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)$

Output:

- $h: \mathbf{x} \rightarrow y$ ($y = h(\mathbf{x})$) $\rightarrow [P(y | \mathbf{x})]$

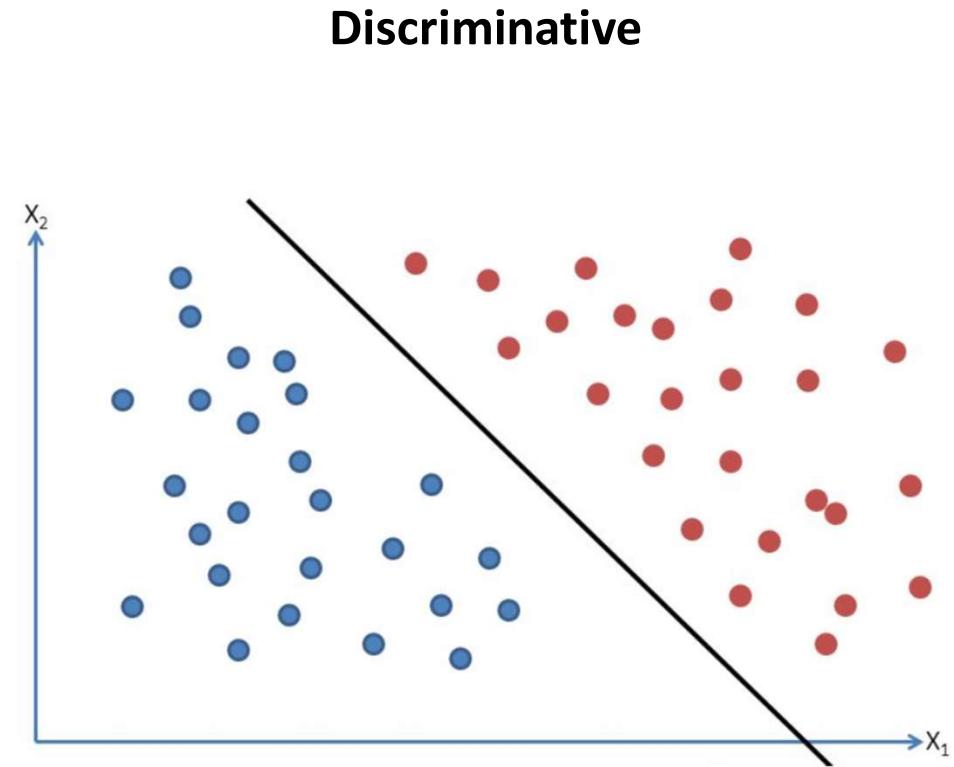
Generative vs Discriminative Models



Generative model models **actual distributions** for EACH CLASS / LABEL / TAG

to

make a $P(\text{class} | \text{sample})$ prediction

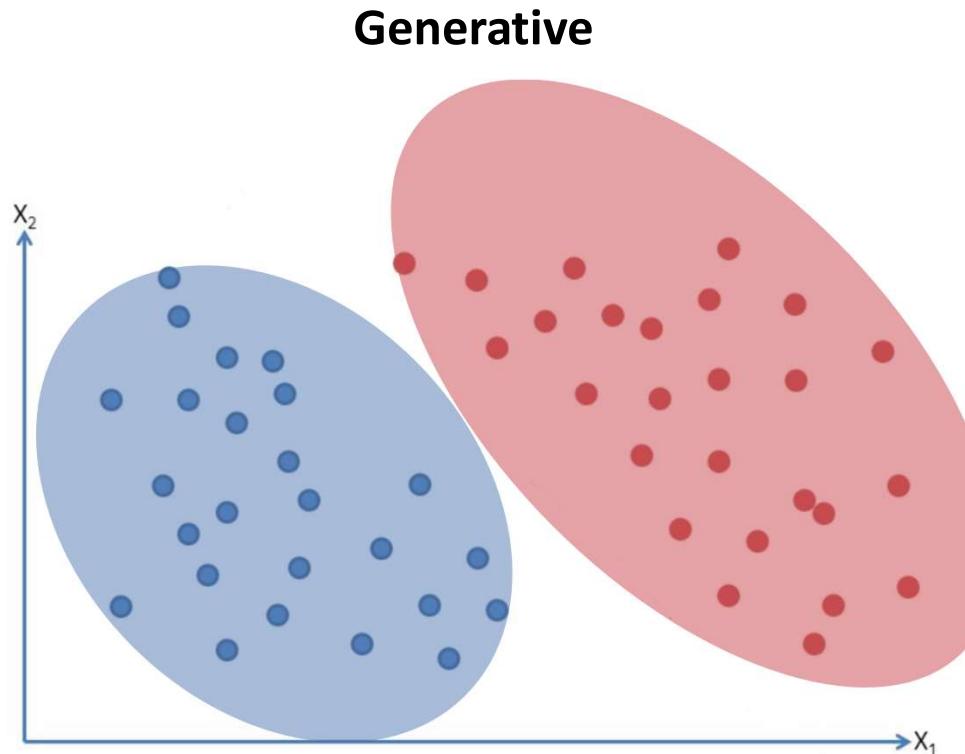


Discriminative model models **the decision boundary** between CLASSES / LABELS / TAGS

to

make a $P(\text{class} | \text{sample})$ prediction

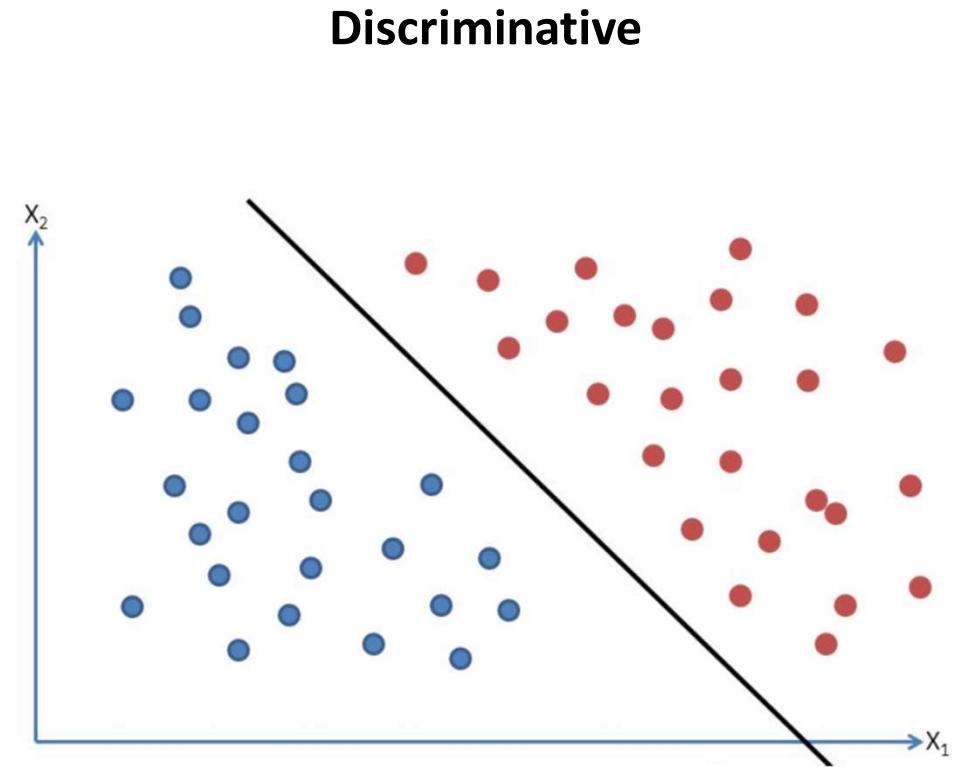
Generative vs Discriminative Models



Generative model uses training data to learn $P(\text{sample}, \text{class})$ **joint probabilities**

and then

uses Bayes Theorem to get the $P(\text{class} | \text{sample})$ prediction

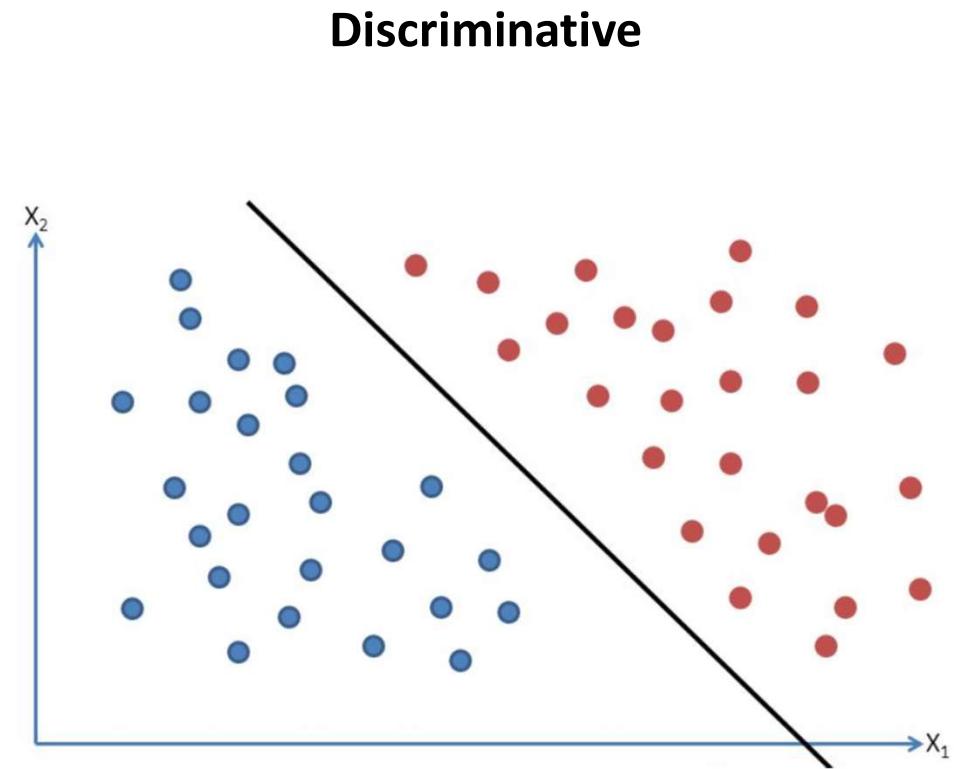
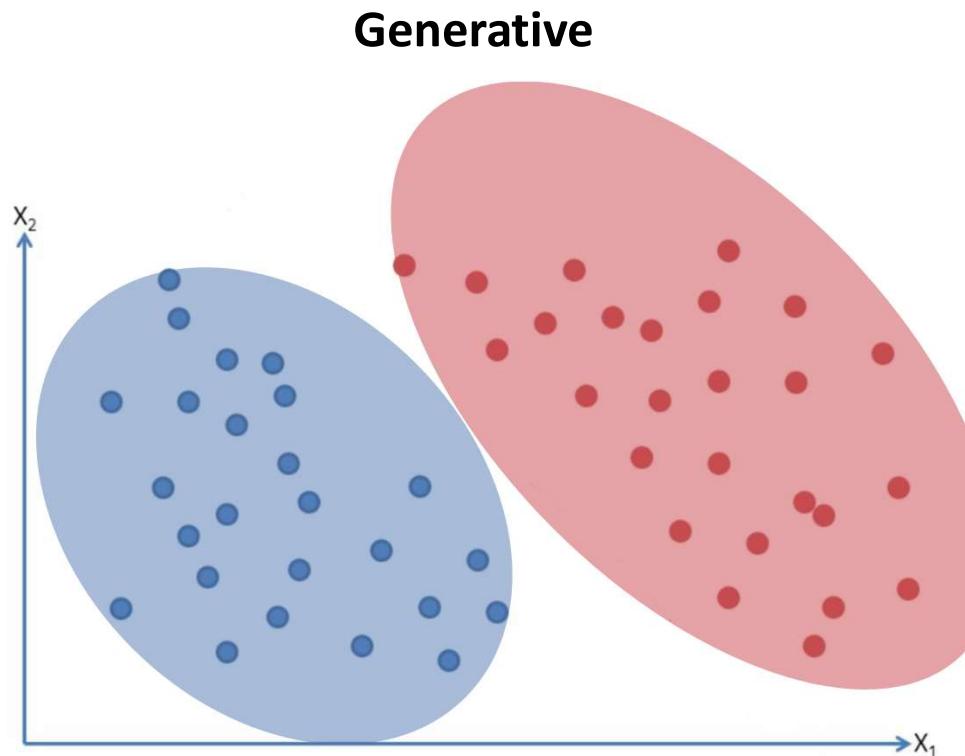


Discriminative model uses training data to learn $P(\text{class} | \text{sample})$ **conditional probability**

and then

uses it to make a prediction

Generative vs Discriminative Models

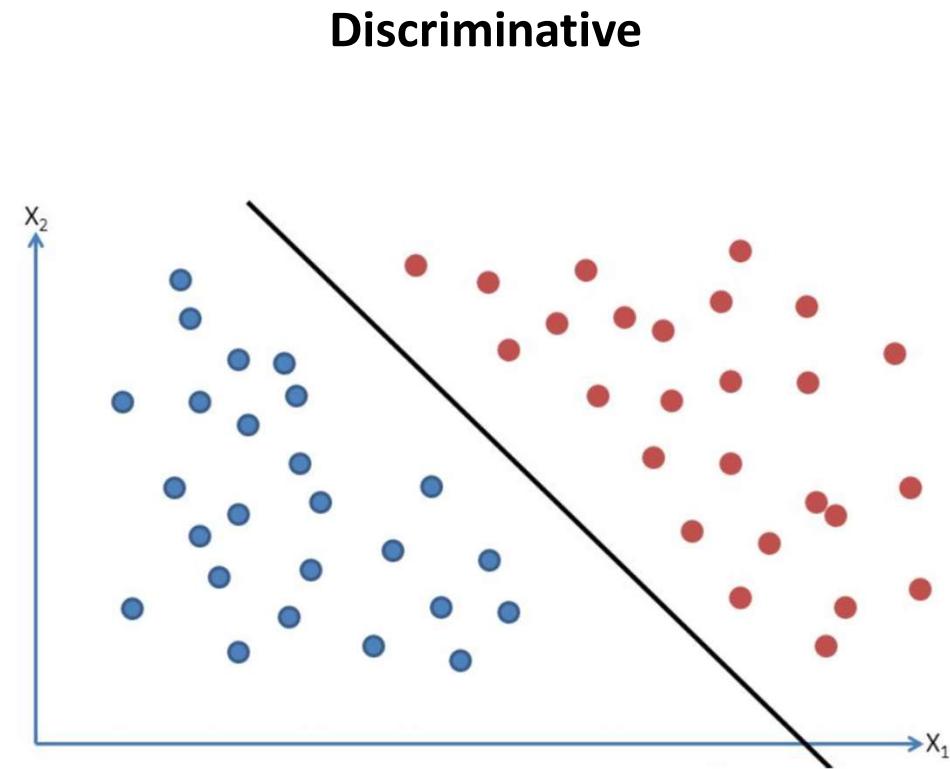
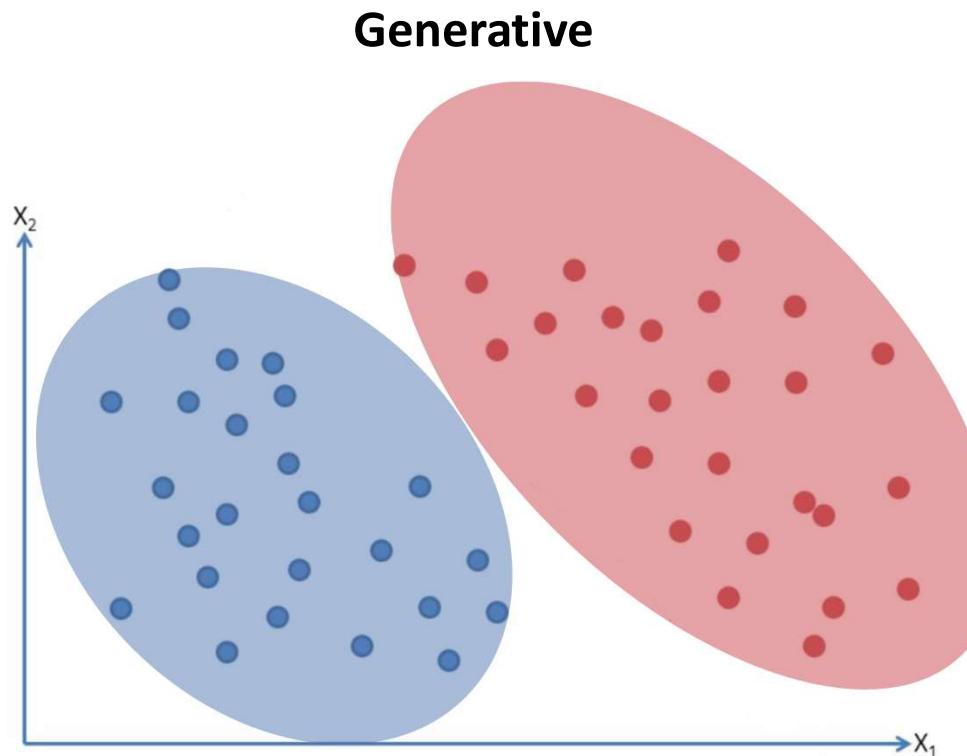


Generative model **learns “details” about the class** (its features):

- includes words “rolex” AND “replica”
- includes word “buy”
- has spelling mistakes
- etc.

Discriminative model **learns where the boundary between classes** is and ignores the “details”

Generative vs Discriminative Classifier



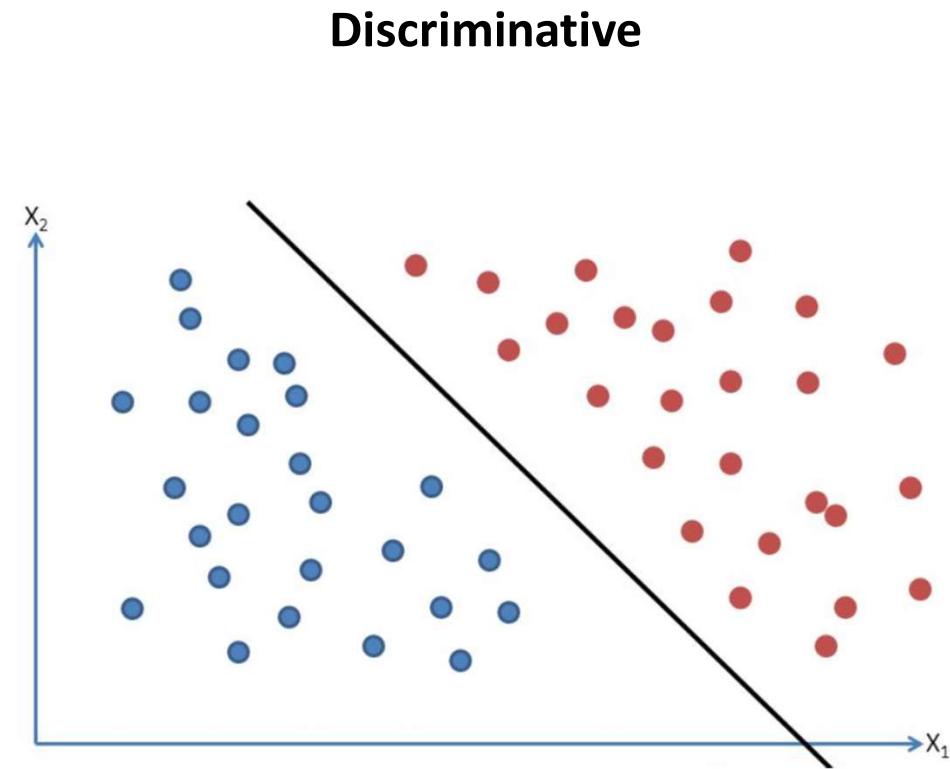
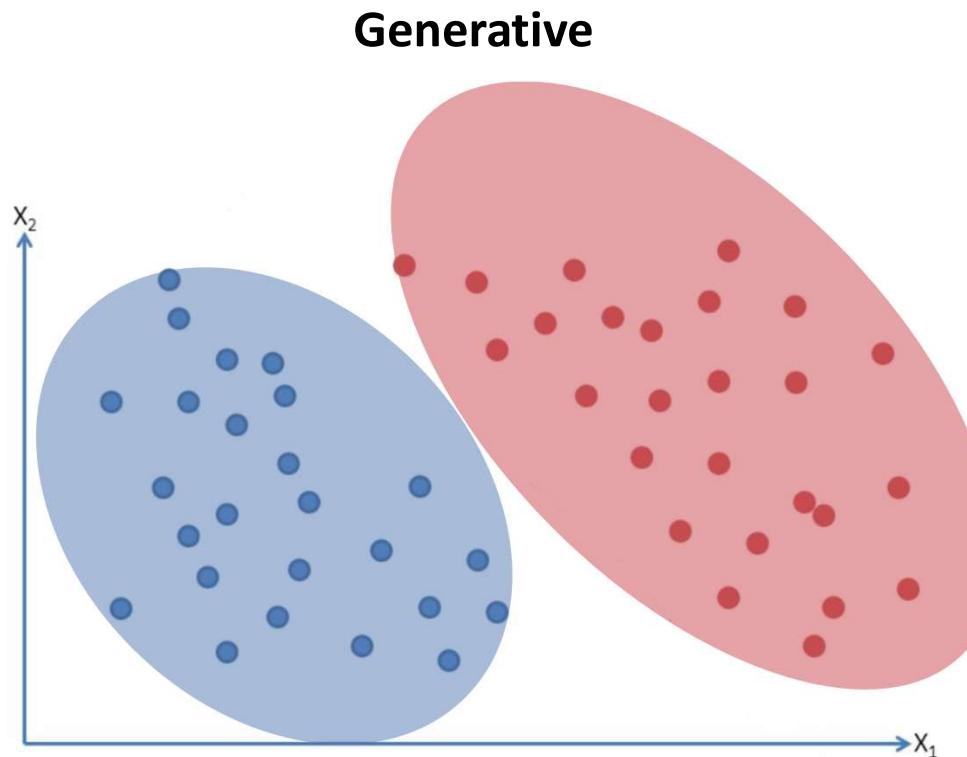
Generative classifiers:

- **Assume some form of $P(\text{class})$, $P(\text{sample} \mid \text{class})$**
- **Estimate $P(\text{class})$, $P(\text{sample} \mid \text{class})$ using training data**
- **Use Bayes Theorem to calculate $P(\text{class} \mid \text{sample})$**

Discriminative classifiers:

- **Assume some form of $P(\text{class} \mid \text{sample})$**
- **Estimate $P(\text{class} \mid \text{sample})$ using training data**

Generative vs Discriminative Classifier



Generative classifiers:

- **Naive Bayes**
- Bayesian networks
- Markov random fields
- Hidden Markov Models (HMM)

Discriminative classifiers:

- **Logistic regression**
- Support Vector Machines
- Traditional neural networks
- k-Nearest Neighbors
- Conditional Random Fields (CRF)s

Classifier

$$y_{MAP} = \underset{y \in Y}{argmax} (P(y | \mathbf{x})) = \underset{y \in Y}{argmax} \left(\frac{P(\mathbf{x} | y) * P(y)}{P(\mathbf{x})} \right)$$

$$\mathbf{x} = x_1, x_2, \dots, x_N$$

MAP: Maximum a posteriori (corresponds to the most likely class).

Classification: Conditional Probability

$$P(y | x) = \frac{P(x | y) * P(y)}{P(x)}$$

$\mathbf{x} = x_1, x_2, \dots, x_N$, so:

$$P(y | x_1 \wedge x_2 \wedge \dots \wedge x_N) = \frac{P(x_1 \wedge x_2 \wedge \dots \wedge x_N | y) * P(y)}{P(x_1 \wedge x_2 \wedge \dots \wedge x_N)}$$

Conditional Probability (Product Rule)

$$P(x_1 \wedge x_2 \wedge \dots \wedge x_N \mid y) * P(y) = P(x_1 \wedge x_2 \wedge \dots \wedge x_N \wedge y)$$

so:

$$P(x_1 \wedge x_2 \wedge \dots \wedge x_N \wedge y) = P(x_1 \wedge x_2 \wedge \dots \wedge x_N \mid y) * P(y)$$

Classification: Conditional Probability

$$P(y | x) = \frac{P(x | y) * P(y)}{P(x)}$$

$\mathbf{x} = x_1, x_2, \dots, x_N$, so:

$$P(y | x_1 \wedge x_2 \wedge \dots \wedge x_N) = \frac{P(x_1 \wedge x_2 \wedge \dots \wedge x_N | y) * P(y)}{P(x_1 \wedge x_2 \wedge \dots \wedge x_N)}$$

$$P(y | x_1 \wedge x_2 \wedge \dots \wedge x_N) = \frac{P(x_1 \wedge x_2 \wedge \dots \wedge x_N \wedge y)}{P(x_1 \wedge x_2 \wedge \dots \wedge x_N)}$$

Classification: Conditional Probability

$$P(y | x) = \frac{P(x | y) * P(y)}{P(x)}$$

$x = x_1, x_2, \dots, x_N$, so:

$$P(y | x_1 \wedge x_2 \wedge \dots \wedge x_N) = \frac{P(x_1 \wedge x_2 \wedge \dots \wedge x_N | y) * P(y)}{P(x_1 \wedge x_2 \wedge \dots \wedge x_N)}$$

Conditional probability

Joint probability

$$P(y | x_1 \wedge x_2 \wedge \dots \wedge x_N) = \frac{P(x_1 \wedge x_2 \wedge \dots \wedge x_N \wedge y)}{P(x_1 \wedge x_2 \wedge \dots \wedge x_N)}$$

Classification: Conditional Probability

$$P(y | x) = \frac{P(x | y) * P(y)}{P(x)}$$

With Naive Bayes
we went after THIS
to get THAT ...

$x = x_1, x_2, \dots, x_N$, SO:

$$P(y | x_1 \wedge x_2 \wedge \dots \wedge x_N) = \frac{P(x_1 \wedge x_2 \wedge \dots \wedge x_N \wedge y)}{P(x_1 \wedge x_2 \wedge \dots \wedge x_N)}$$

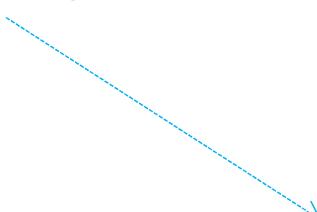
... while making some really
Naive assumptions along the
way

Classification: Conditional Probability

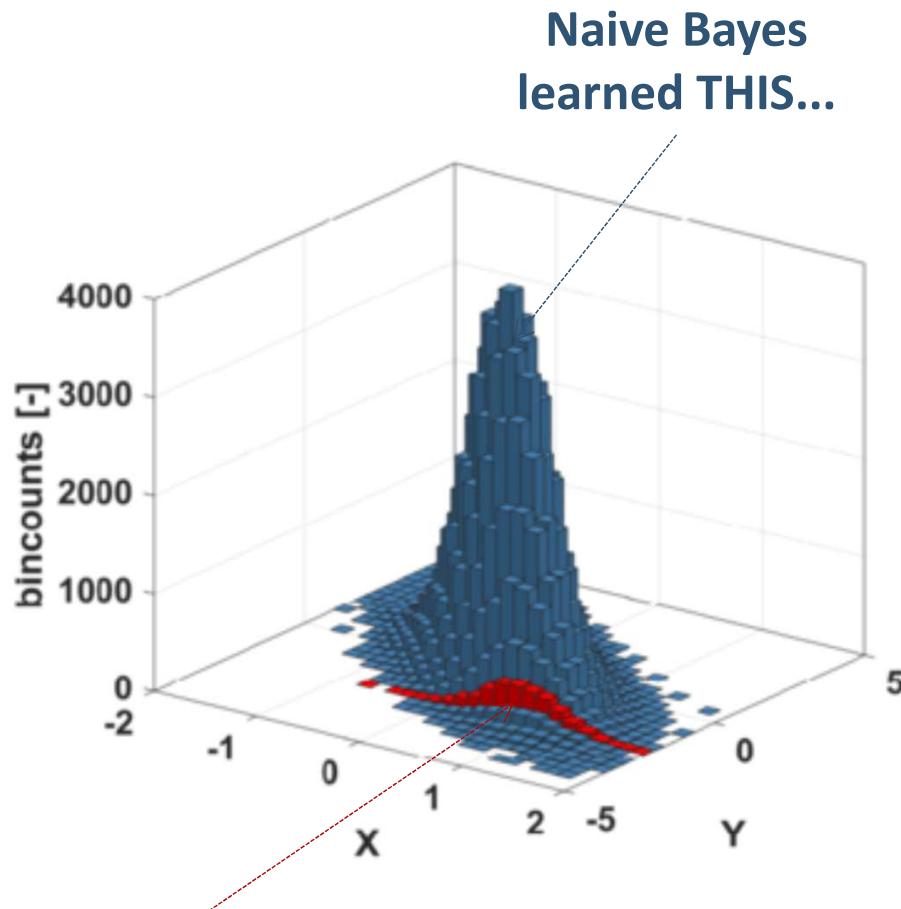
$$P(y | x) = \frac{P(x | y) * P(y)}{P(x)}$$

Logistic Regression
will learn **THIS**
directly

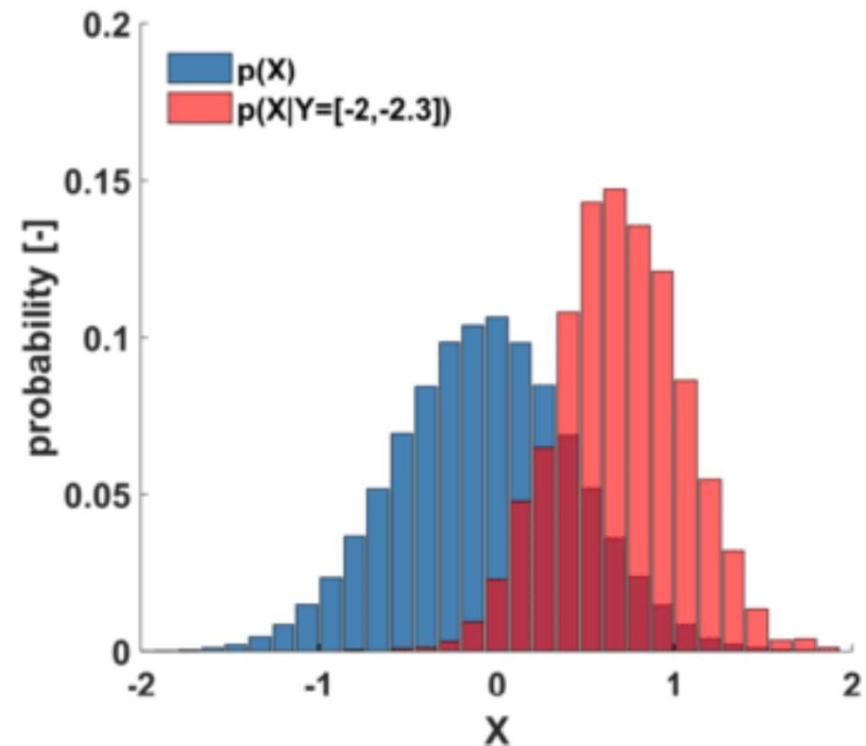
$\mathbf{x} = x_1, x_2, \dots, x_N$, so:


$$P(y | x_1 \wedge x_2 \wedge \dots \wedge x_N) = \frac{P(x_1 \wedge x_2 \wedge \dots \wedge x_N \wedge y)}{P(x_1 \wedge x_2 \wedge \dots \wedge x_N)}$$

Joint vs Conditional Probability



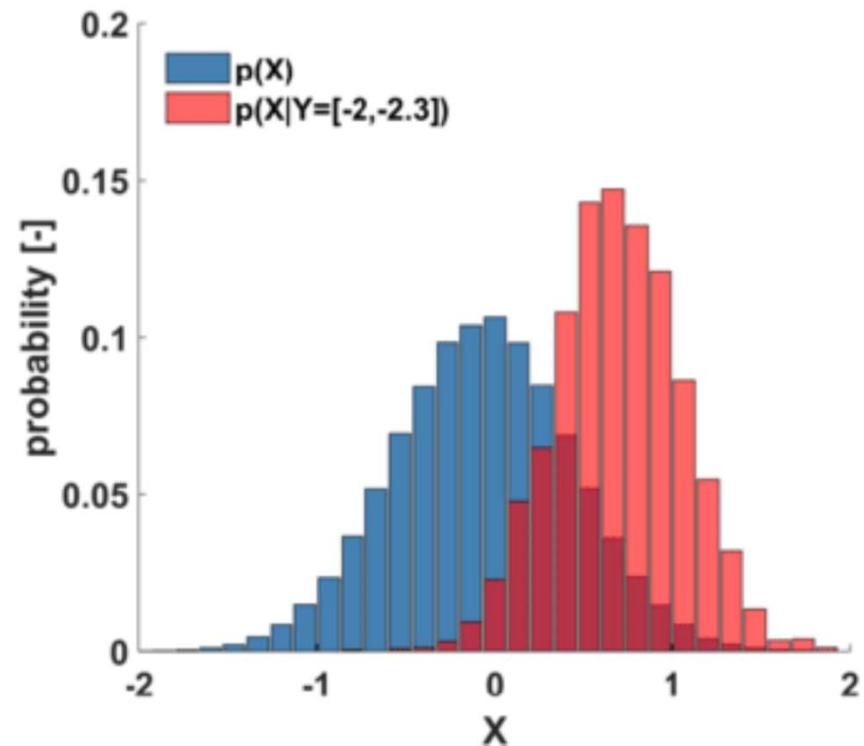
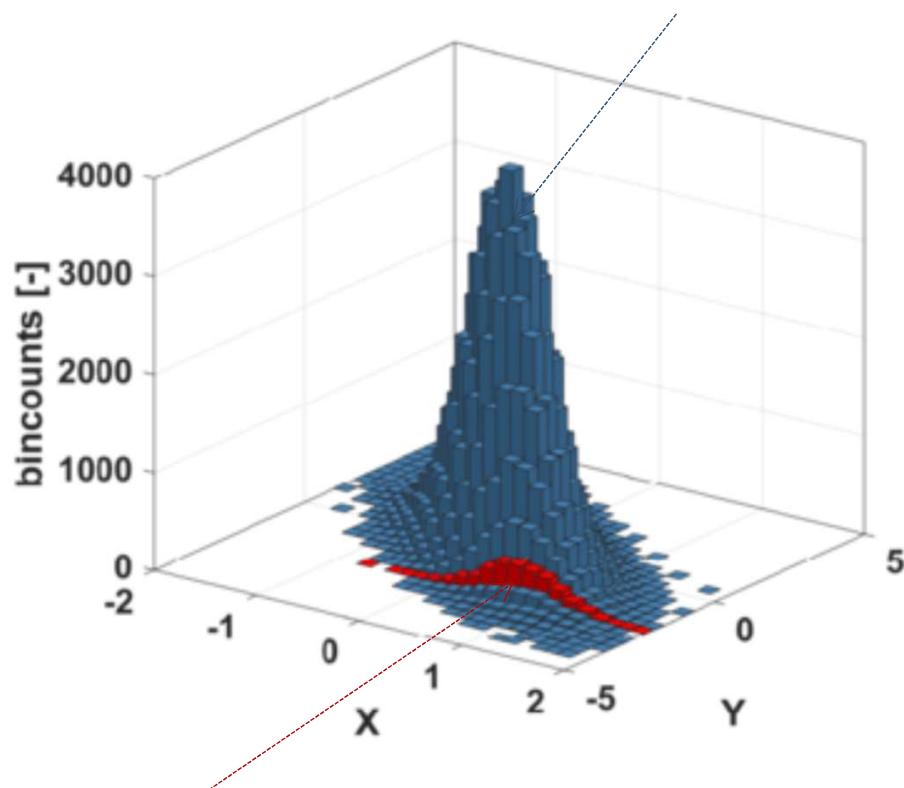
... to be able to
estimate THAT



Source: https://www.researchgate.net/figure/Illustration-of-a-bivariate-conditional-probability-distribution-as-a-simple-data-based_fig1_330092567

Joint vs Conditional Probability

Logistic regression
ignores THIS...



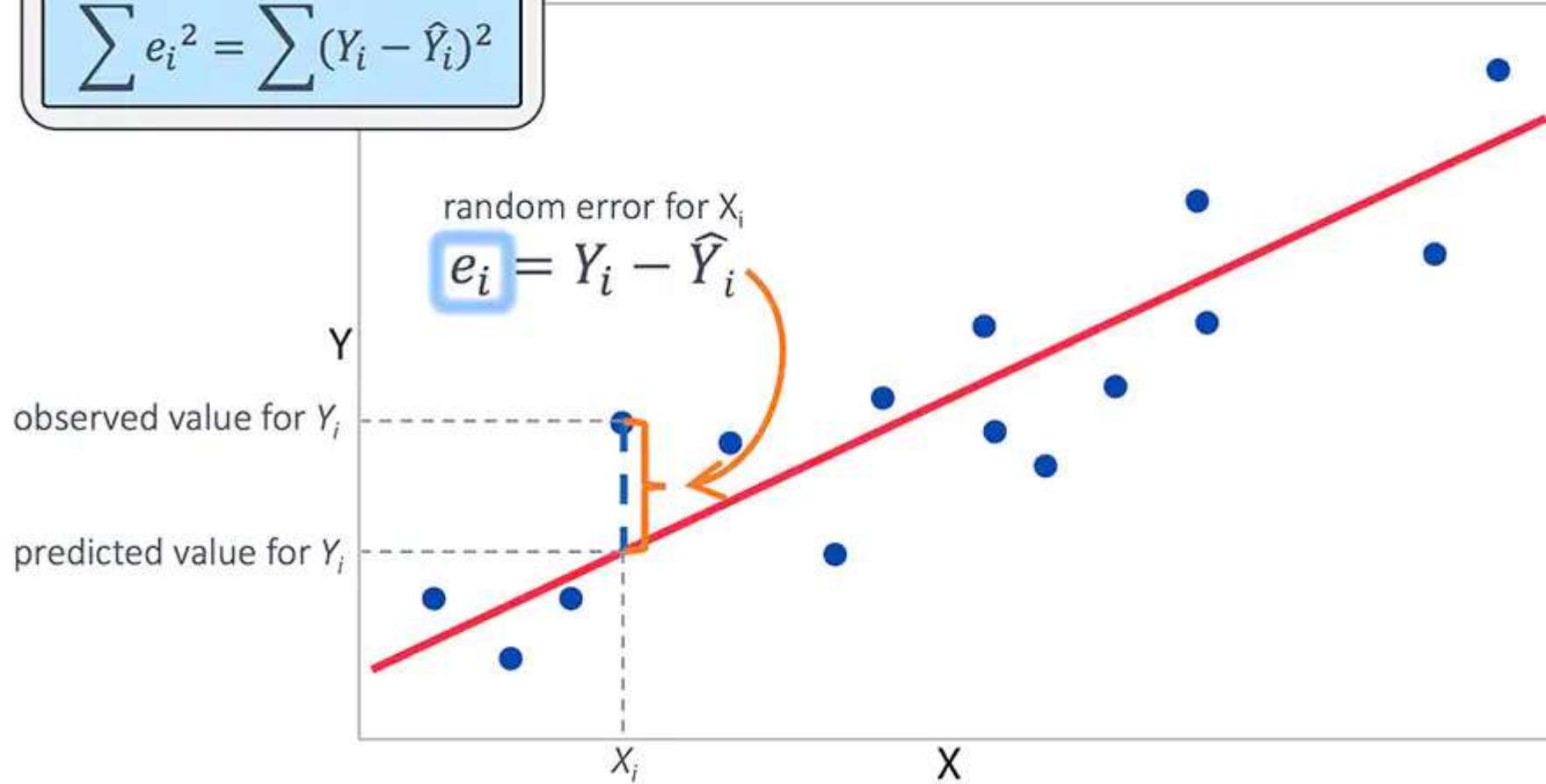
... and learns THAT
instead

Source: https://www.researchgate.net/figure/Illustration-of-a-bivariate-conditional-probability-distribution-as-a-simple-data-based_fig1_330092567

Linear Regression Using Least-Squares

Method of Least Squares

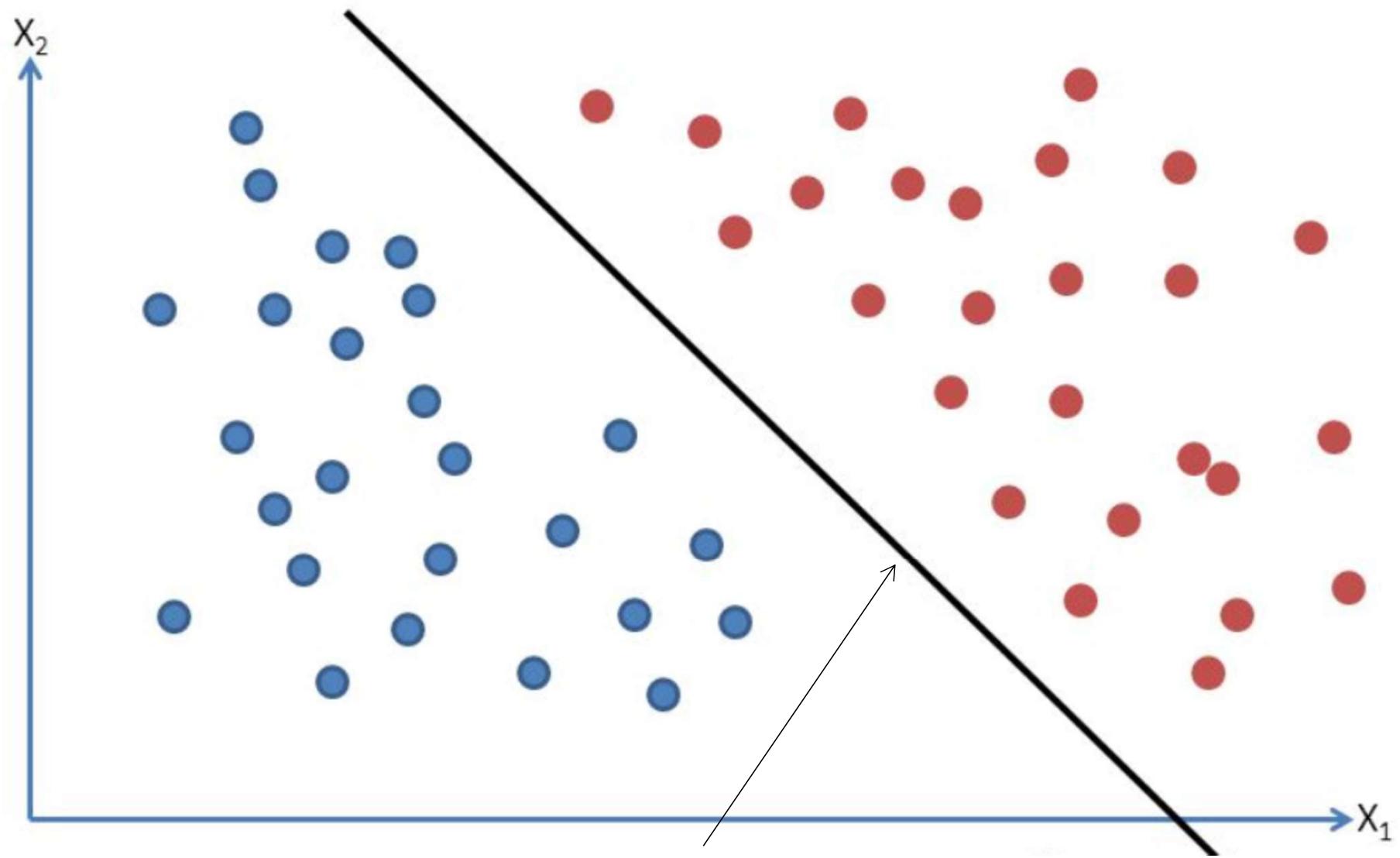
$$\sum e_i^2 = \sum (Y_i - \hat{Y}_i)^2$$



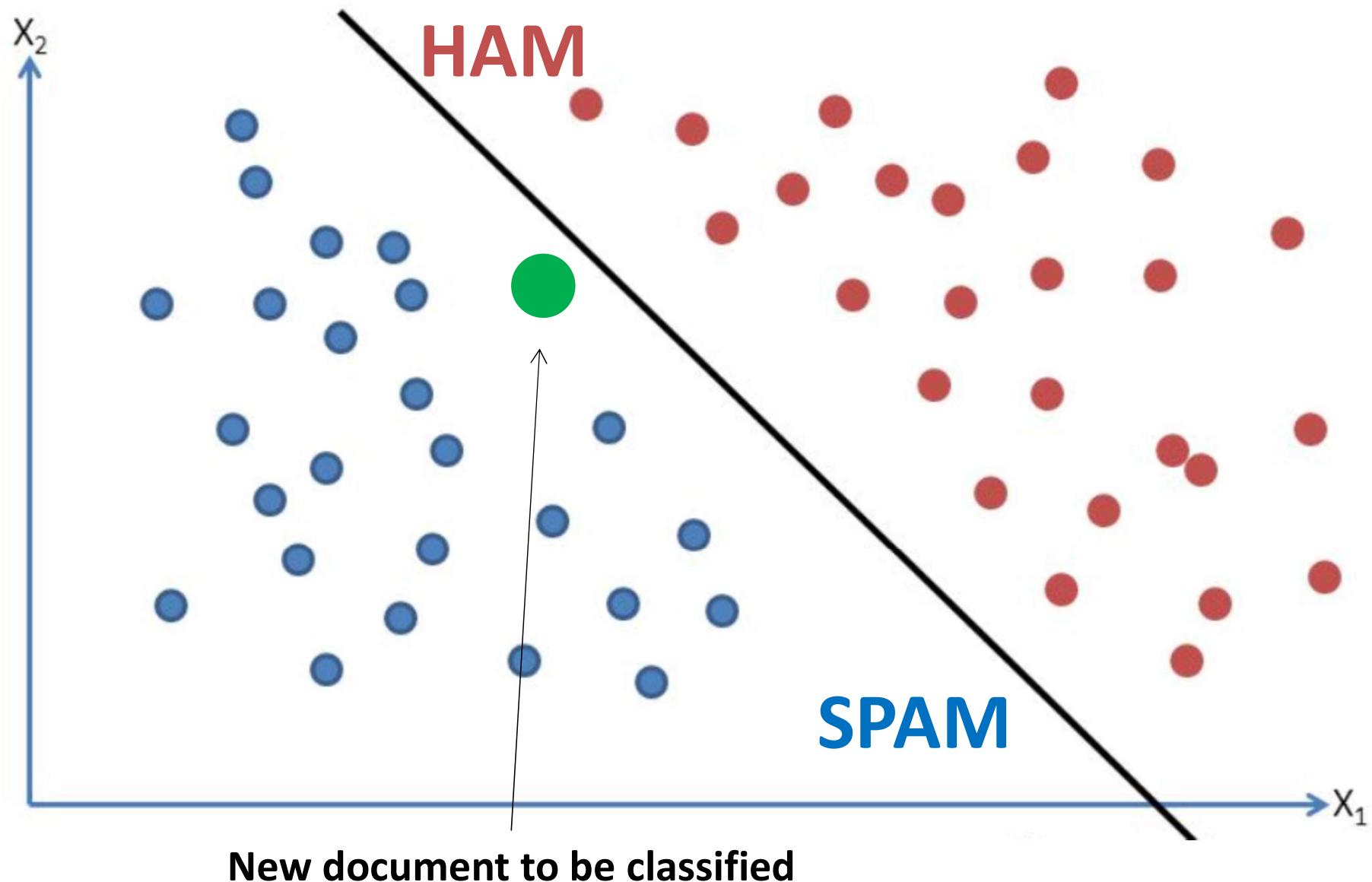
The goal is to find the line $y = mx + n$ that **minimizes the amount of error**.

Source: https://www.jmp.com/en_us/statistics-knowledge-portal/what-is-multiple-regression/fitting-multiple-regression-model.html

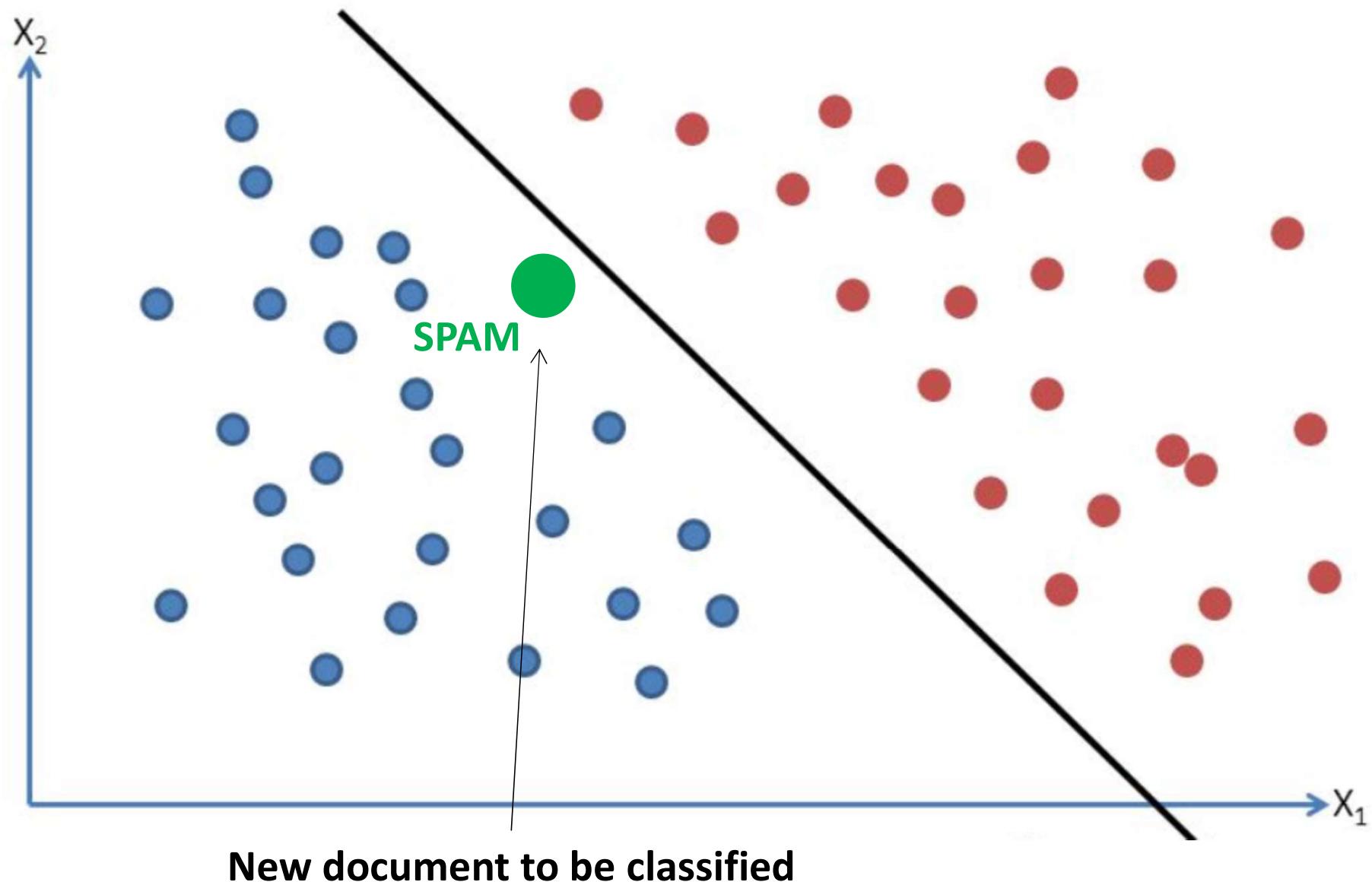
Linear Separator



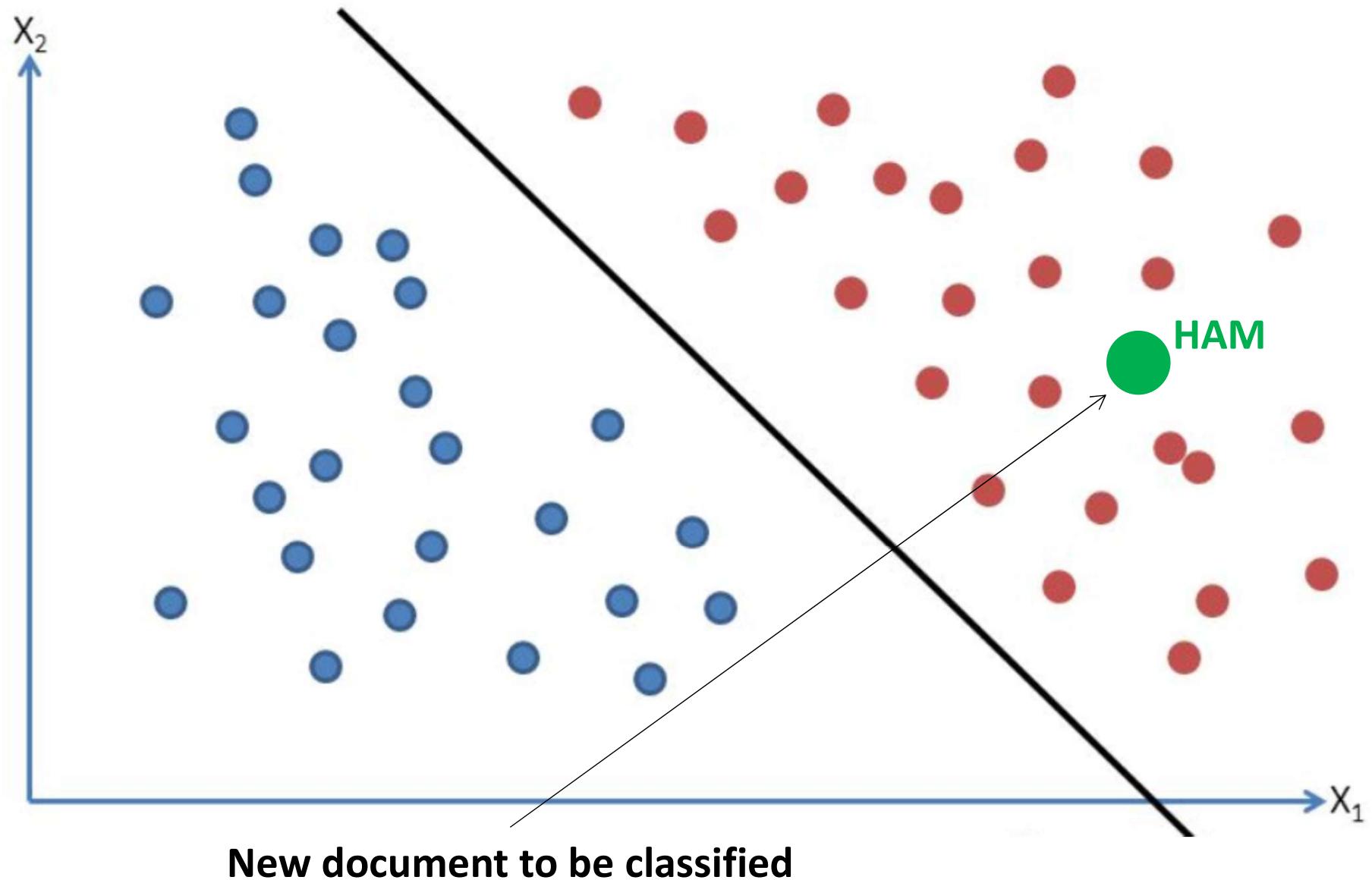
Text Classification: Separator



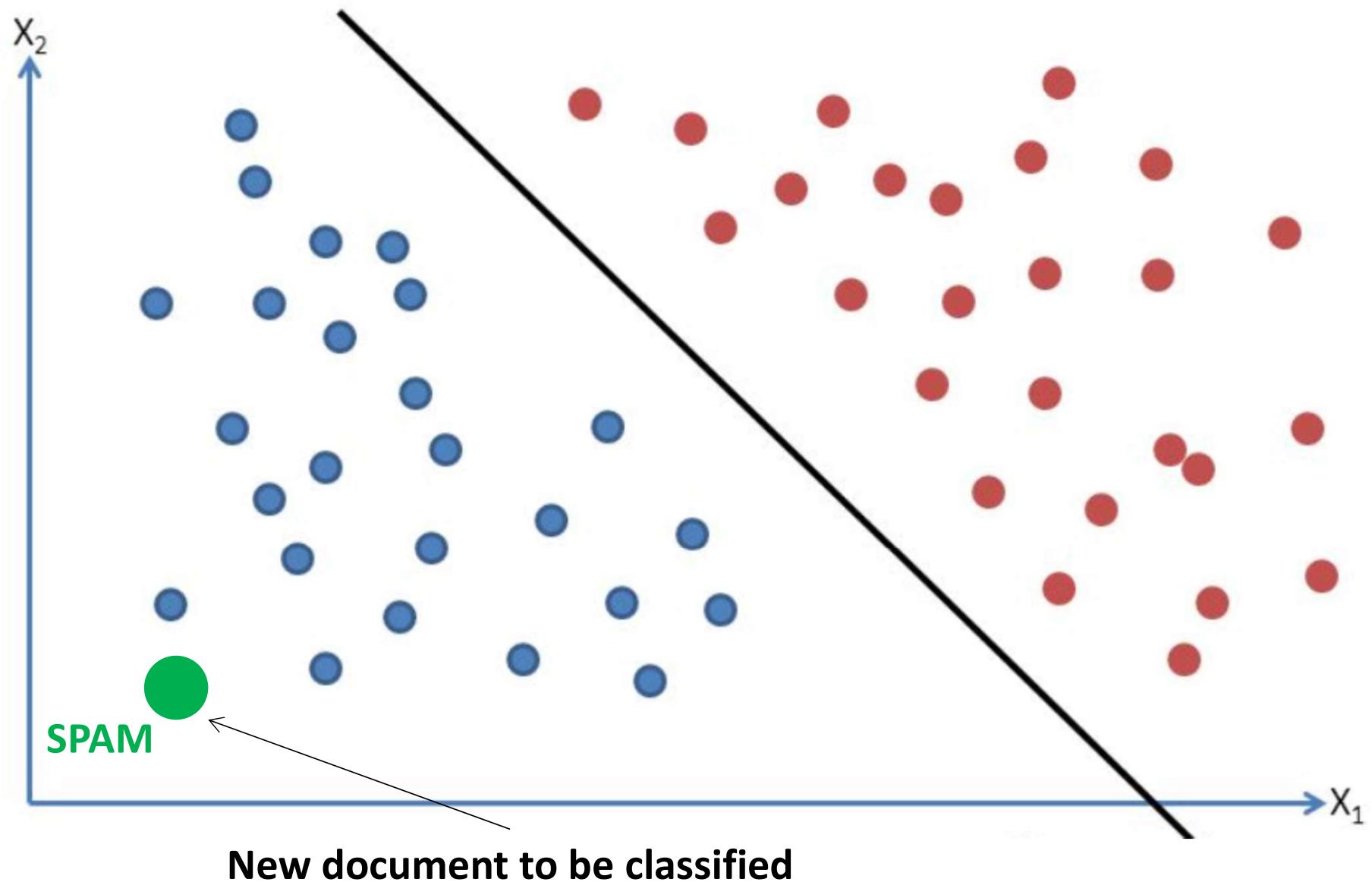
Text Classification: Separator



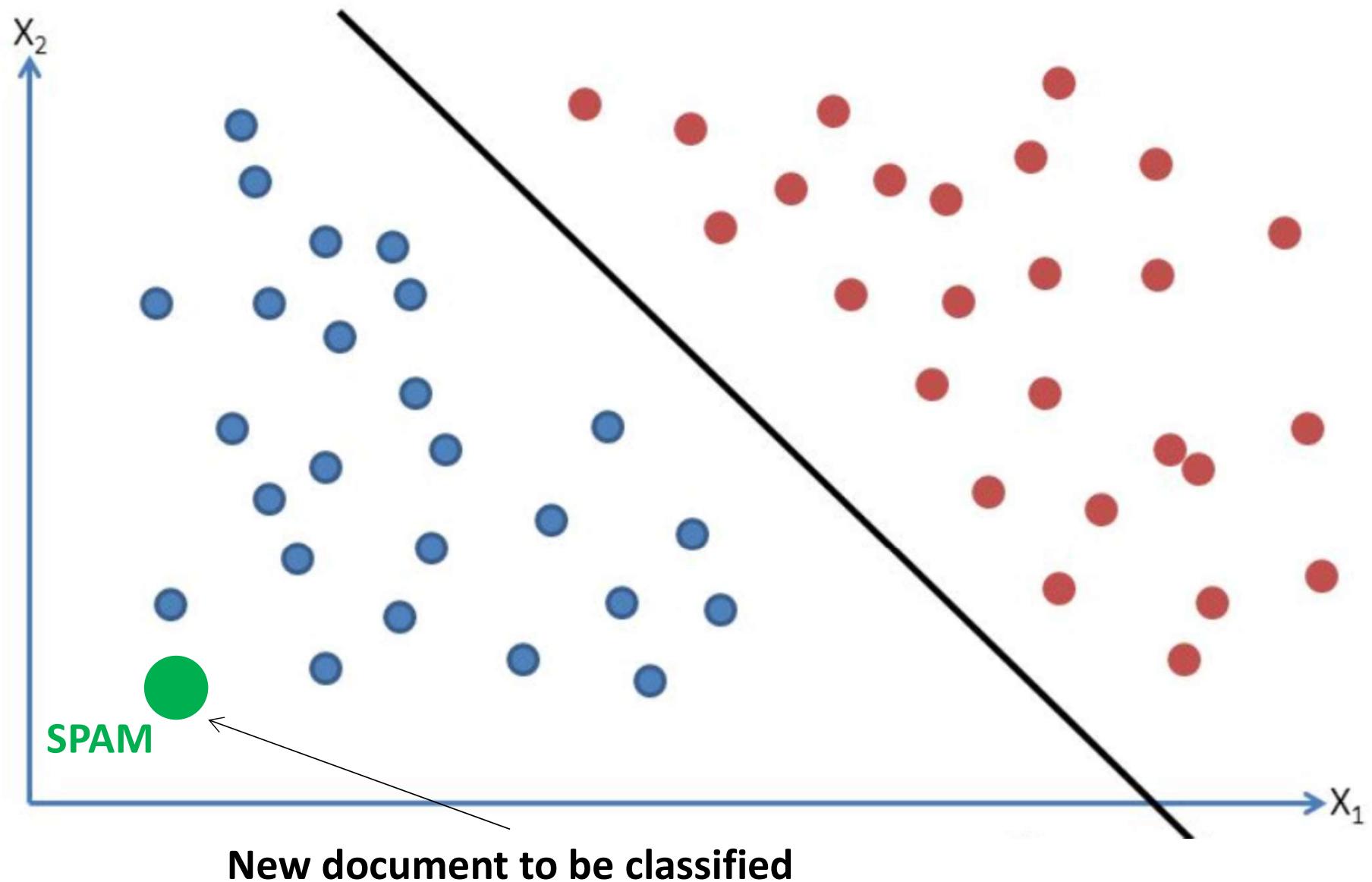
Text Classification: Separator



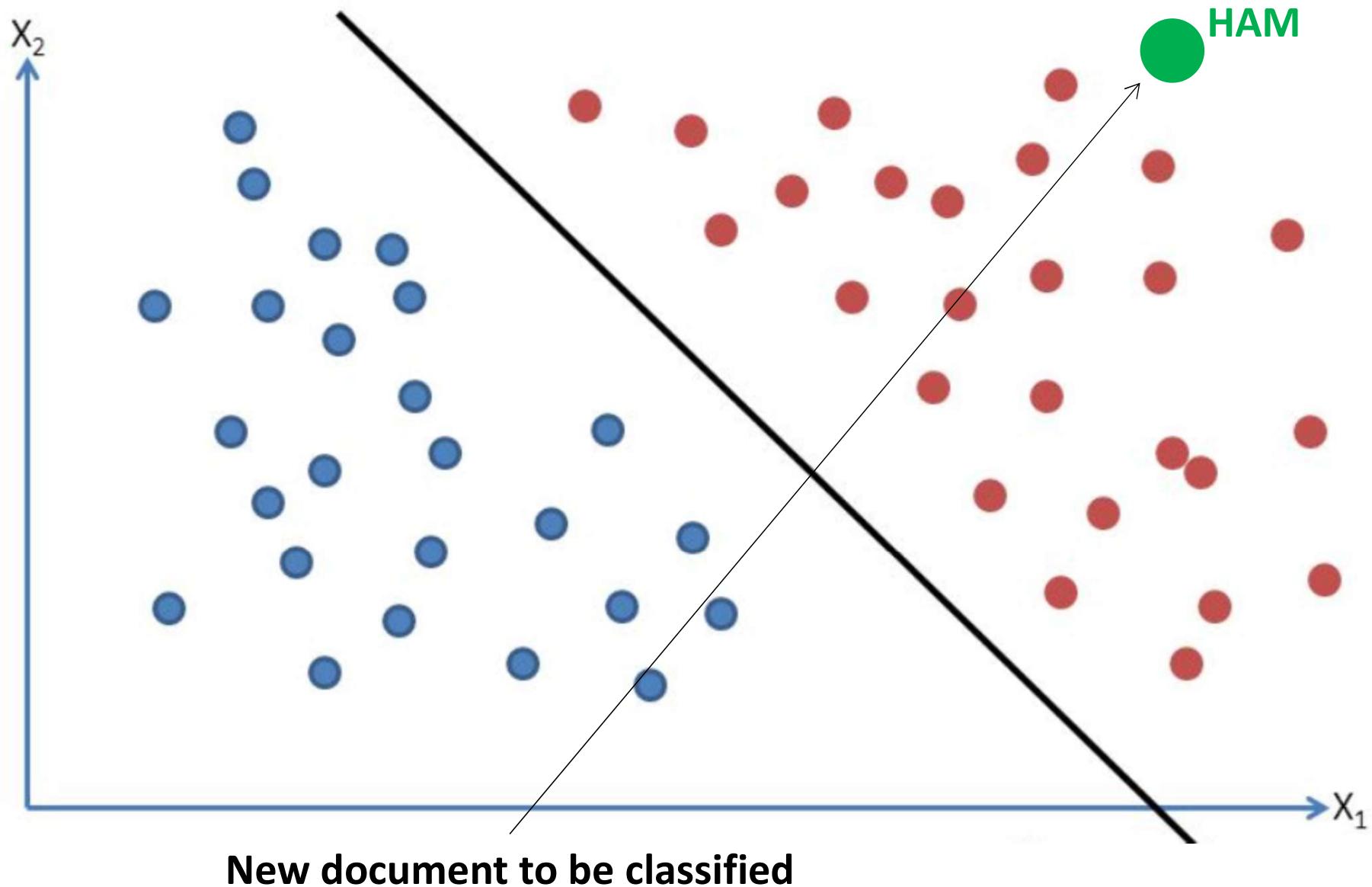
Text Classification: Separator



Text Classification: Separator



Text Classification: Separator



Vector Dot / Scalar Product

Given two vectors \mathbf{a} and \mathbf{b} (N - vector space dimension):

$$\mathbf{a} = [a_1, a_2, \dots, a_N] \text{ and } \mathbf{b} = [b_1, b_2, \dots, b_N]$$

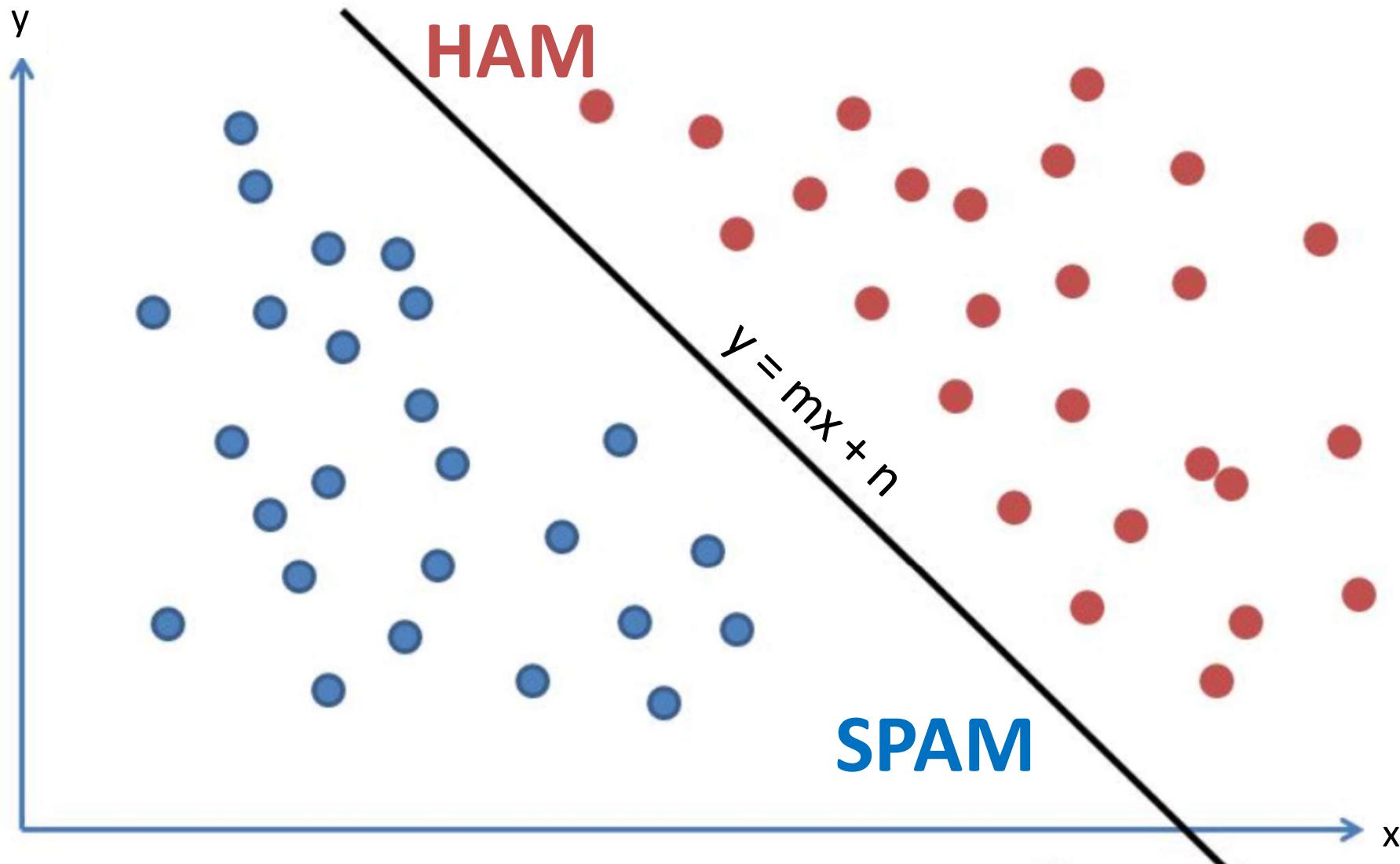
their vector dot/scalar product is:

$$\mathbf{a} \cdot \mathbf{b} = \sum_{i=1}^N a_i * b_i = a_1 * b_1 + a_2 * b_2 + \dots + a_N * b_N$$

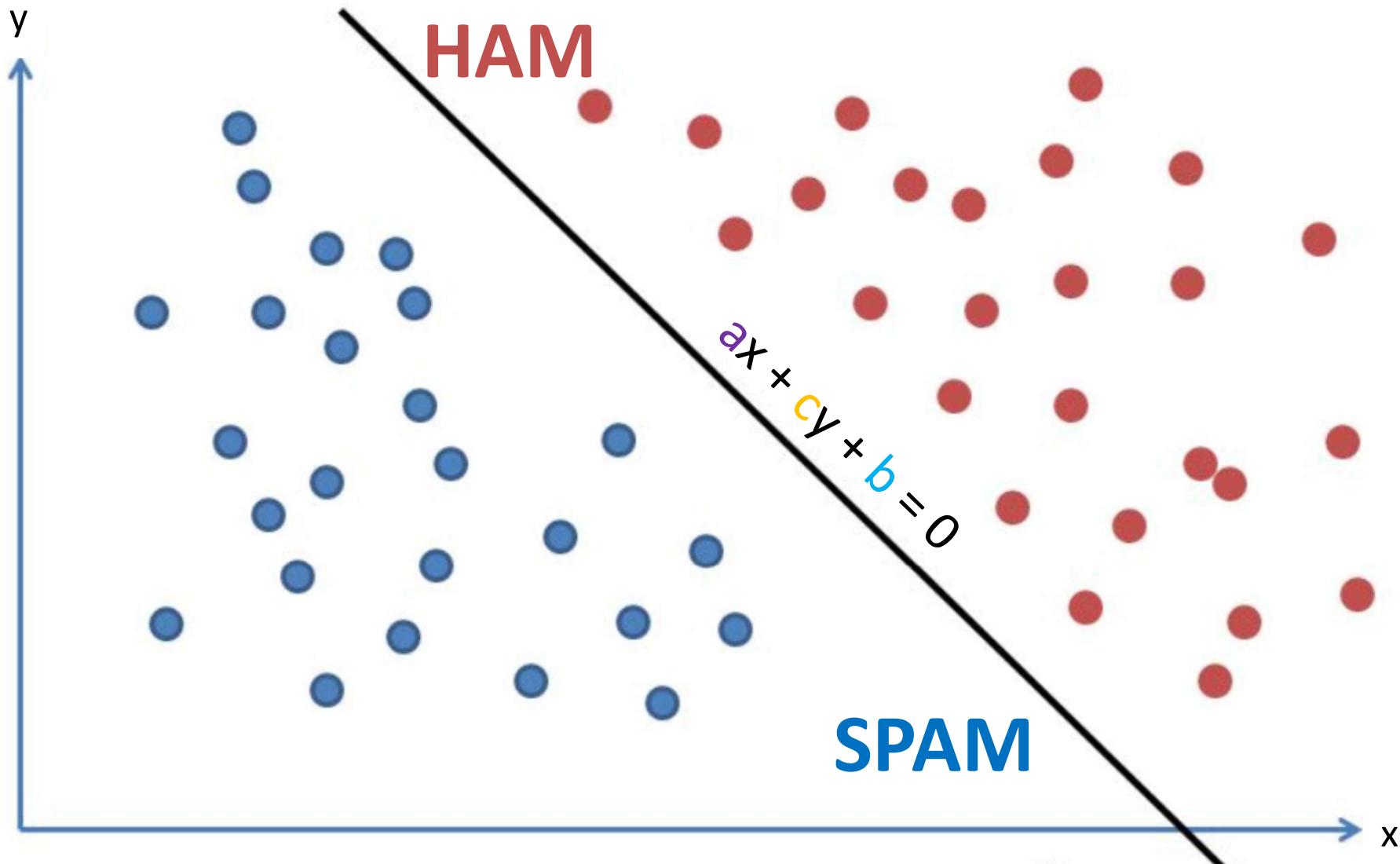
Using matrix representation:

$$\mathbf{a} \cdot \mathbf{b} = \mathbf{ab}^T = [a_1 \quad a_2 \quad \cdots \quad a_N] \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_N \end{bmatrix}$$

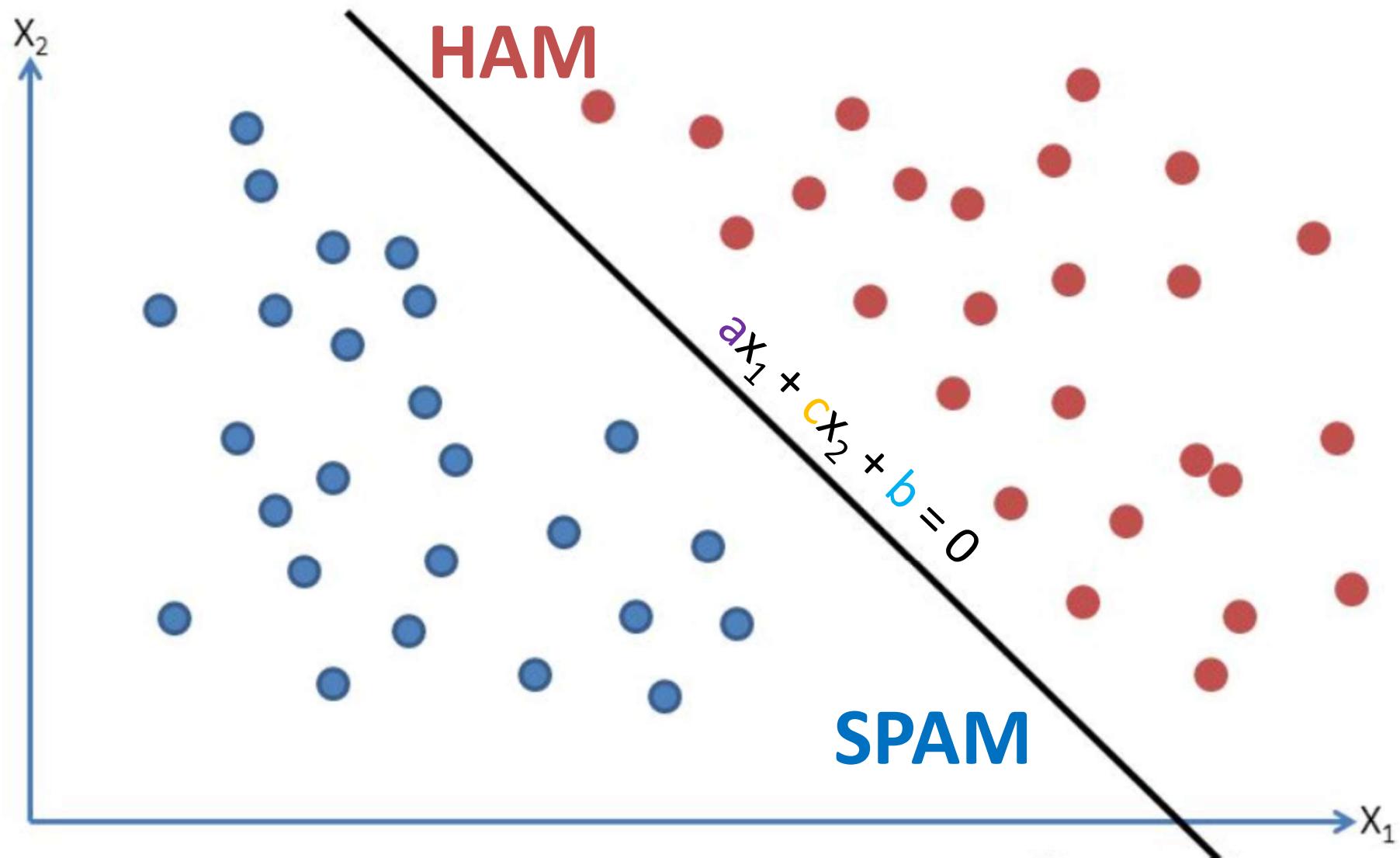
Text Classification: Separator



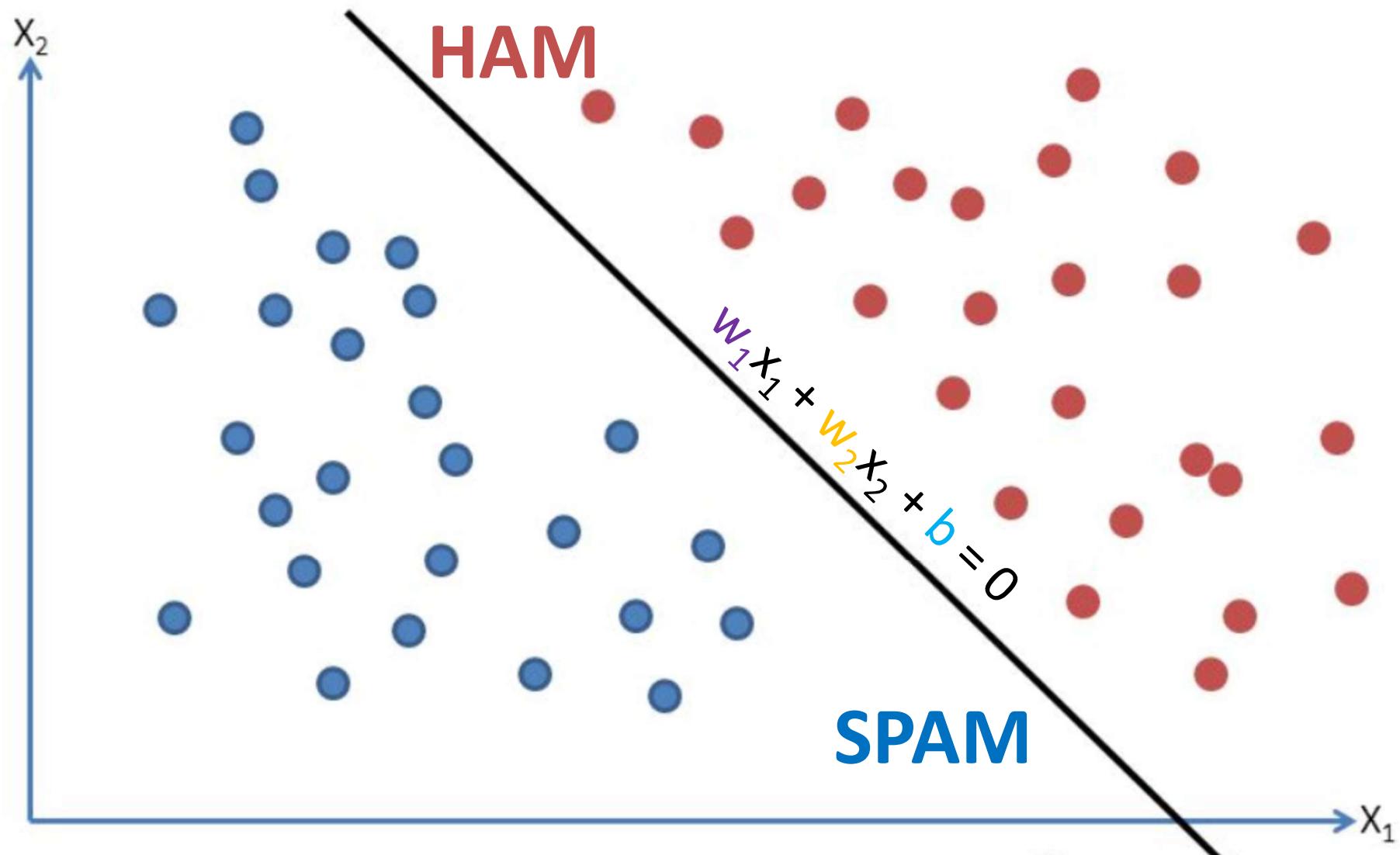
Text Classification: Separator



Text Classification: Separator



Text Classification: Separator



Vector Dot / Scalar Product

Given two vectors w and x (N - vector space dimension):

$$w = [w_1, w_2, \dots, w_N] \text{ and } x = [x_1, x_2, \dots, x_N]$$

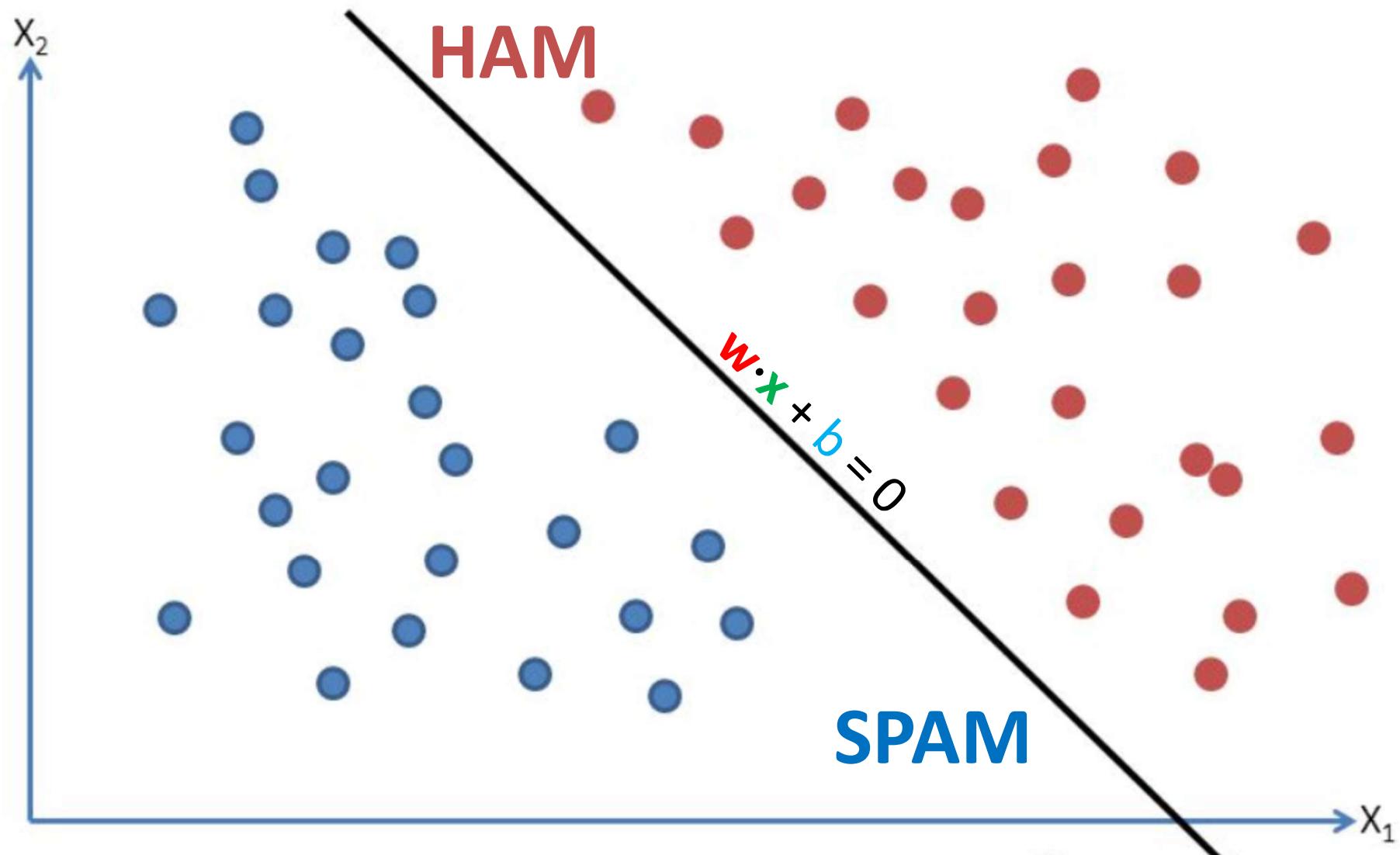
their vector dot/scalar product is:

$$w \cdot x = \sum_{i=1}^N w_i * x_i = w_1 * x_1 + w_2 * x_2 + \dots + w_N * x_N$$

Using matrix representation:

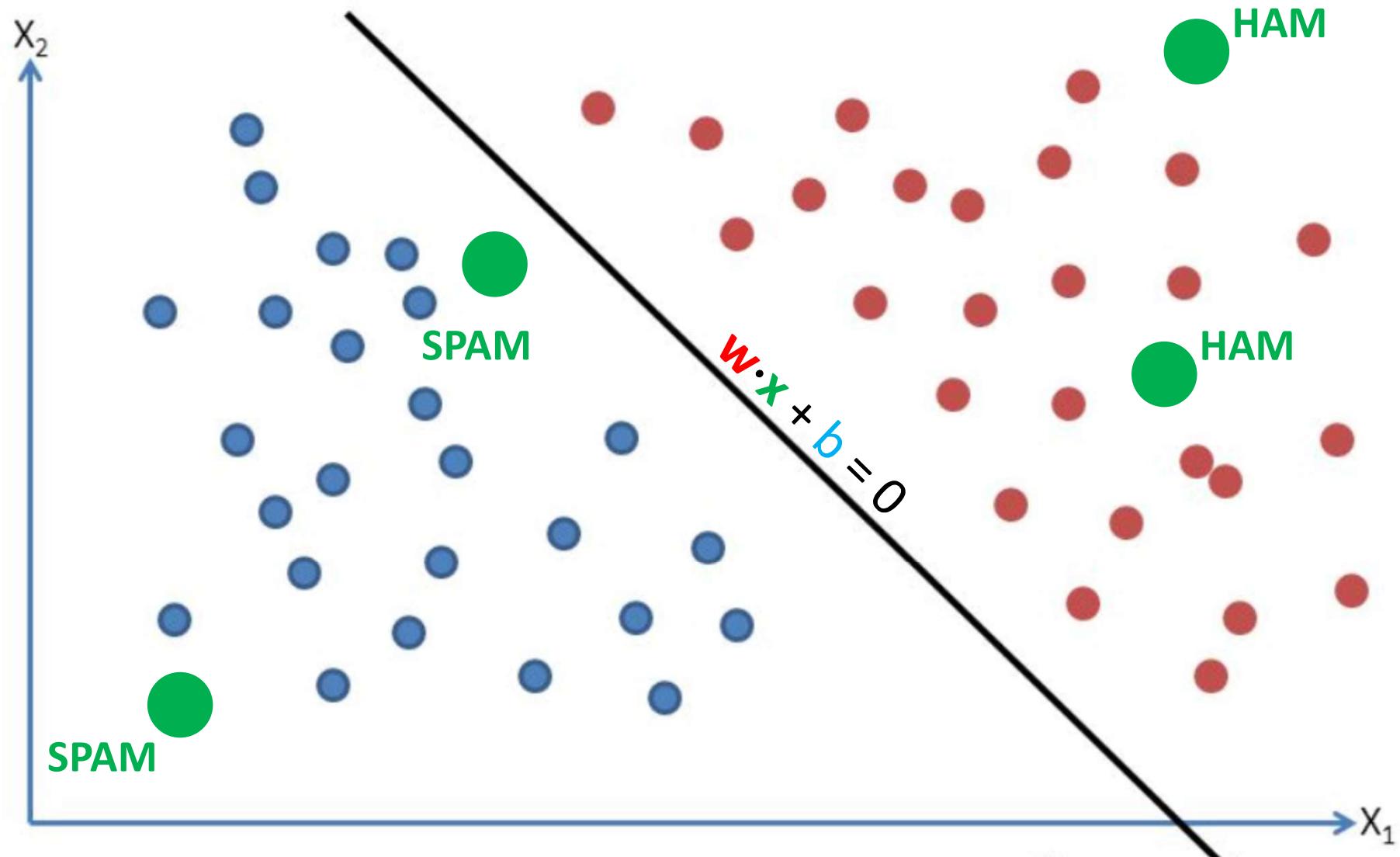
$$w \cdot x = wx^T = [w_1 \quad w_2 \quad \dots \quad w_N] \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix}$$

Text Classification: Separator

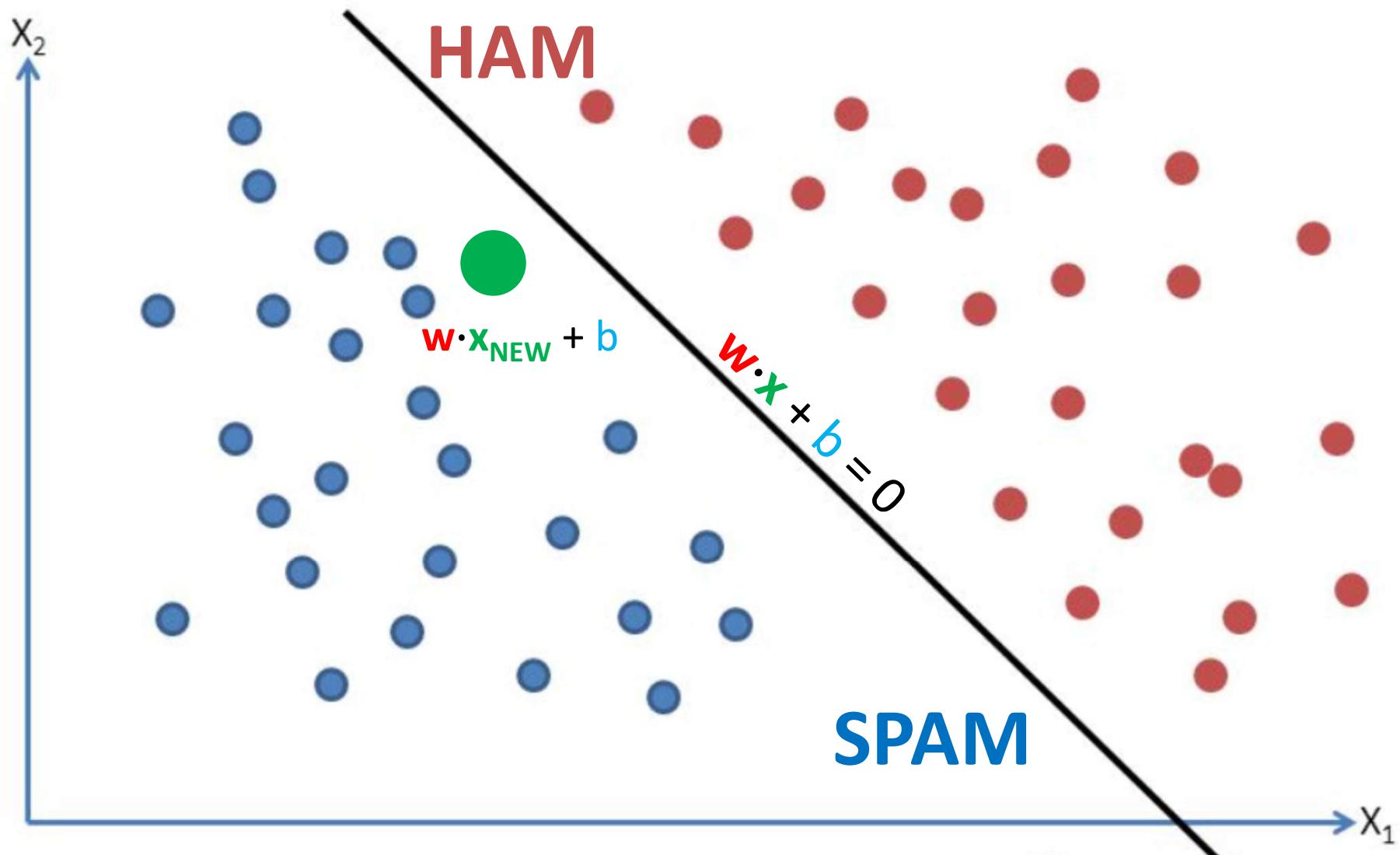


where: $w = [w_1, w_2]$ and $x = [x_1, x_2]$

Text Classification: Separator

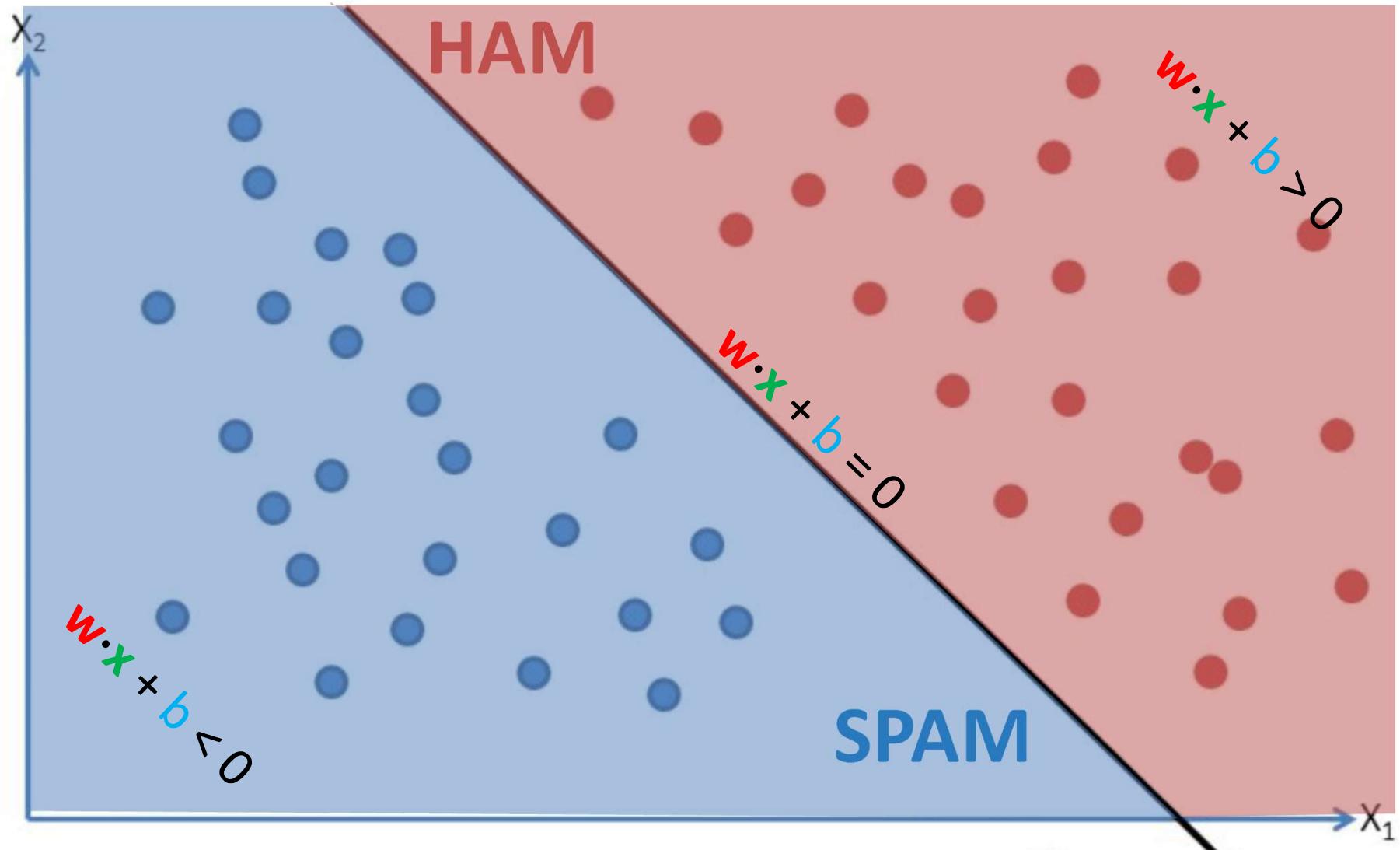


Text Classification: Separator



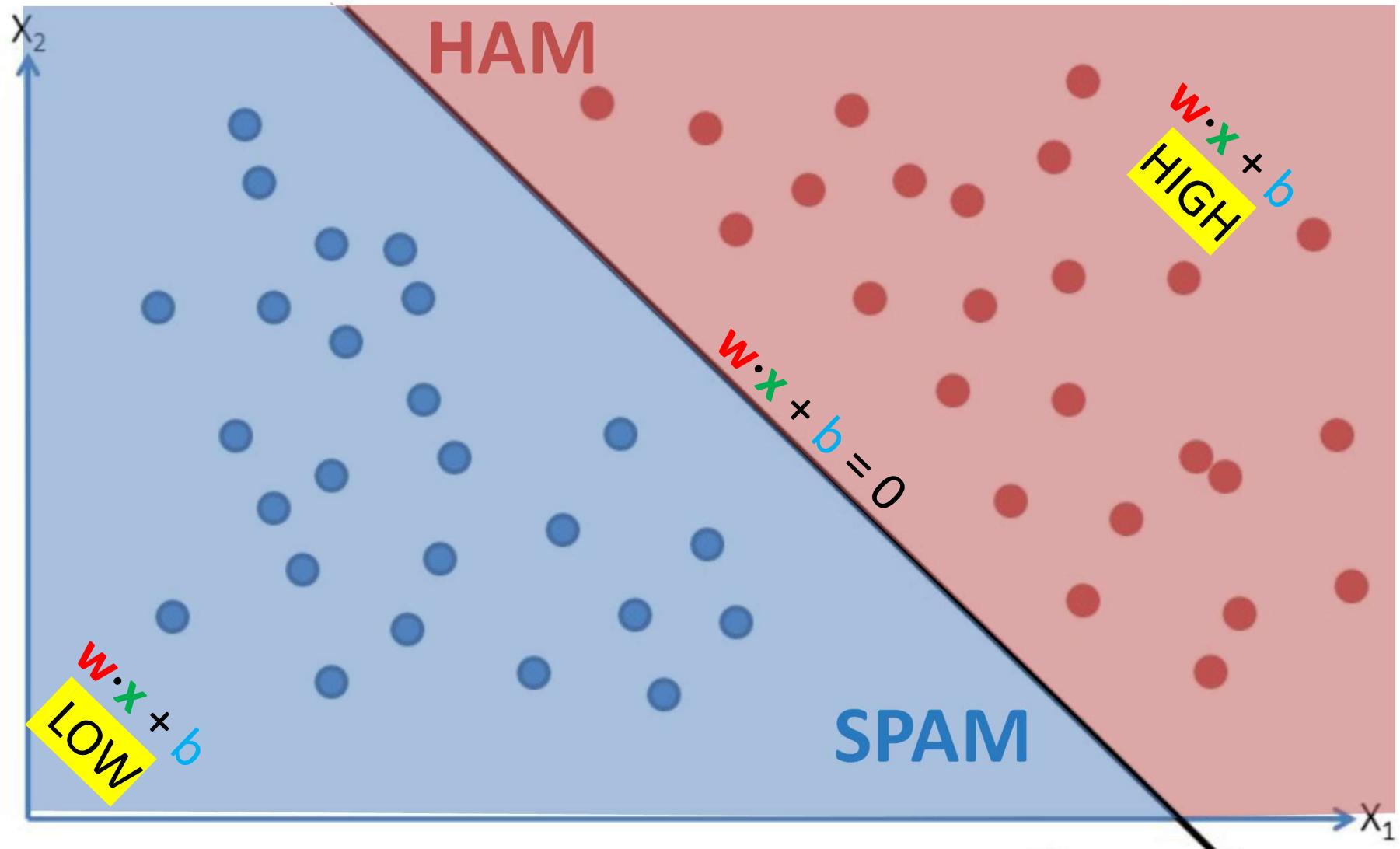
where: $w = [w_1, w_2]$ and $x = [x_1, x_2]$

Text Classification: Separator



where: $w = [w_1, w_2]$ and $x = [x_1, x_2]$

Text Classification: Separator



where: $w = [w_1, w_2]$ and $x = [x_1, x_2]$

Binary Classification: Refresher

Given a series of input/output pairs:

$$(\mathbf{x}^{(i)}, y^{(i)})$$

For each observation / sample $\mathbf{x}^{(i)}$

Where $\mathbf{x}^{(i)}$ is represented by a feature vector

$$[\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N]$$

We want to compute a predicted class

$$\hat{y}^{(i)} \in \{0, 1\}$$

Logistic Regression Classification

Input observation / sample vector $\mathbf{x}^{(i)}$:

$$\mathbf{x} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N]$$

Weights vector \mathbf{w} :

$$\mathbf{w} = [w_1, w_2, \dots, w_N]$$

Output:

$$y' \in \{0,1\}$$

Logistic Regression: Initial Approach

For each feature x_i , corresponding weight w_i tells us the importance of x_i

Sum up all the weighted features and the bias

$$z = \left(\sum_{i=1}^N w_i x_i \right) + b$$
$$z = w \cdot x + b$$

If this sum is HIGH, we say $y = 1$; if LOW, then $y = 0$

We Want a Probabilistic Classifier

We need to formalize **HIGH** and **LOW**.

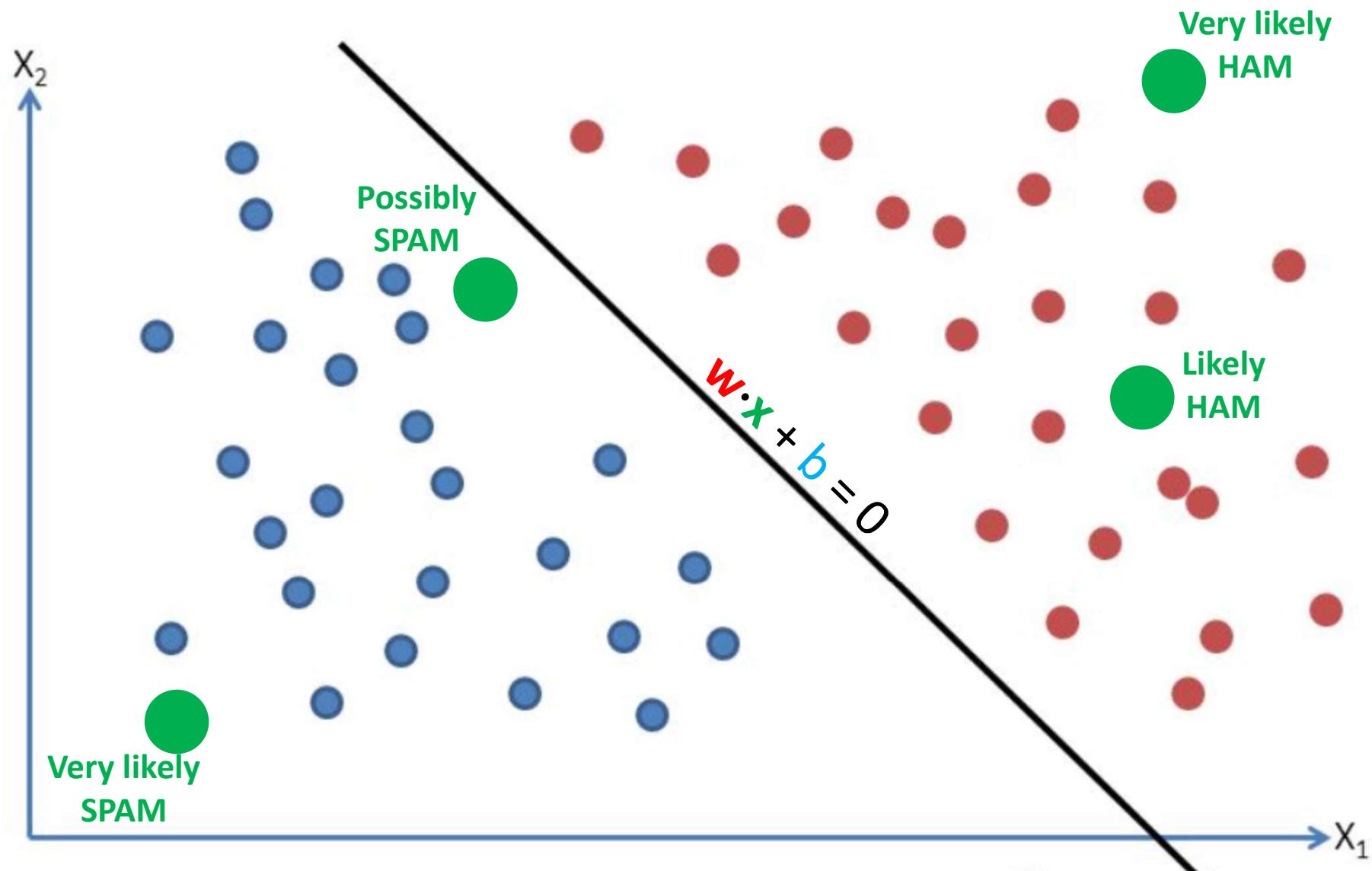
We'd like a principled classifier that gives us a probability, just like Naive Bayes did

We want a model that can tell us:

$$P(y = 1 \mid \mathbf{x}; \mathbf{w}) \text{ or } P(y = \text{SPAM} \mid \mathbf{x}; \mathbf{w})$$

$$P(y = 0 \mid \mathbf{x}; \mathbf{w}) \text{ or } P(y = \text{HAM} \mid \mathbf{x}; \mathbf{w})$$

Text Classification: Separator



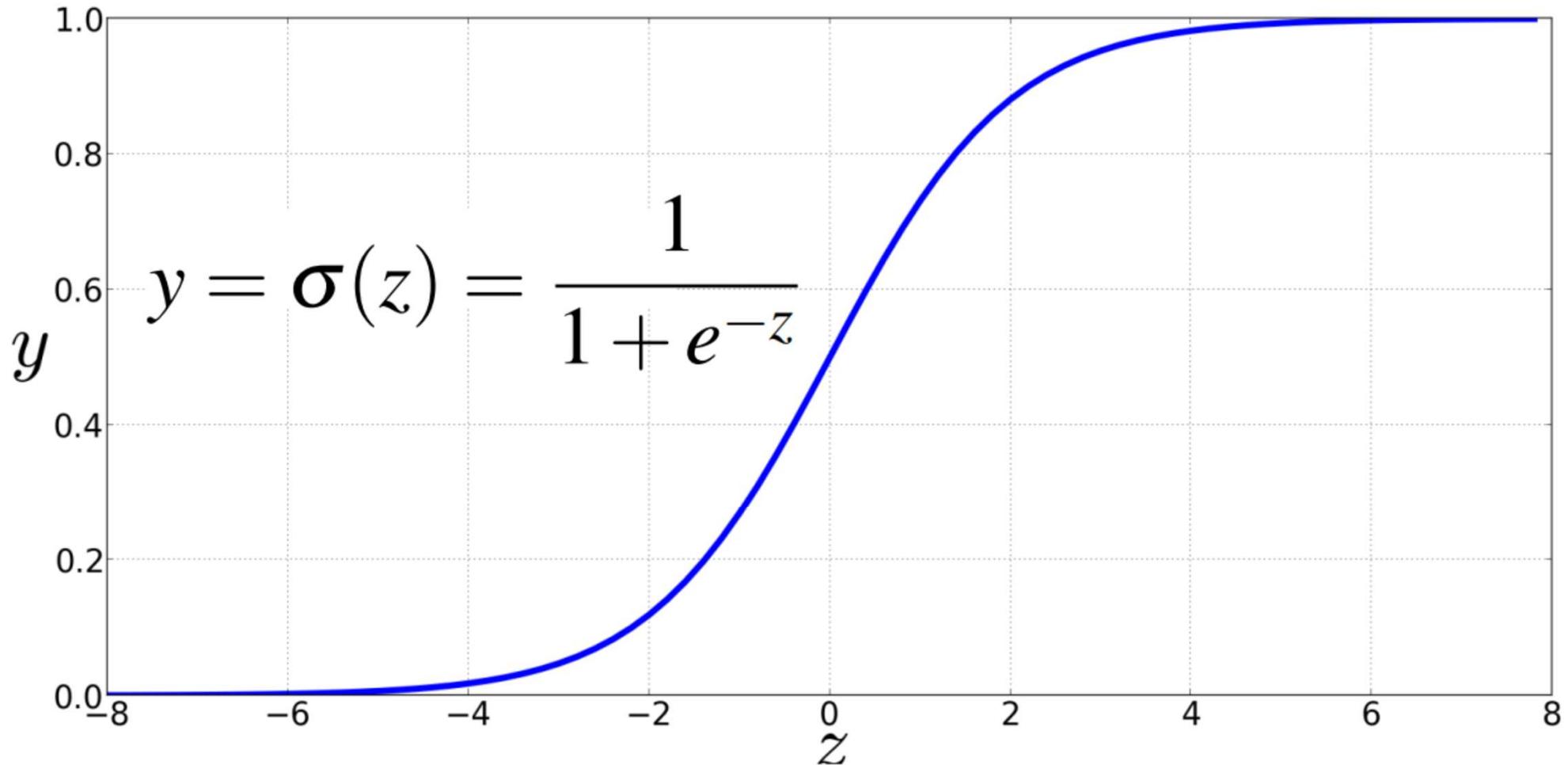
Logistic Regression: The Idea

- Compute $w \cdot x + b$ for observation / sample x
- Pass it through the sigmoid function:

$$\sigma(w \cdot x + b)$$

- Treat the result as probability

Sigmoid / Logistic Function



Calculating Probabilities with Sigmoid

$$\begin{aligned} P(y = 1) &= \sigma(w \cdot x + b) \\ &= \frac{1}{1 + \exp(-(w \cdot x + b))} \end{aligned}$$

$$\begin{aligned} P(y = 0) &= 1 - \sigma(w \cdot x + b) \\ &= 1 - \frac{1}{1 + \exp(-(w \cdot x + b))} \\ &= \frac{\exp(-(w \cdot x + b))}{1 + \exp(-(w \cdot x + b))} \end{aligned}$$

By The Way...

$$\begin{aligned} P(y=0) &= 1 - \sigma(w \cdot x + b) & = \sigma(-(w \cdot x + b)) \\ &= 1 - \frac{1}{1 + \exp(-(w \cdot x + b))} & \text{Because} \\ &= \frac{\exp(-(w \cdot x + b))}{1 + \exp(-(w \cdot x + b))} & 1 - \sigma(x) = \sigma(-x) \end{aligned}$$

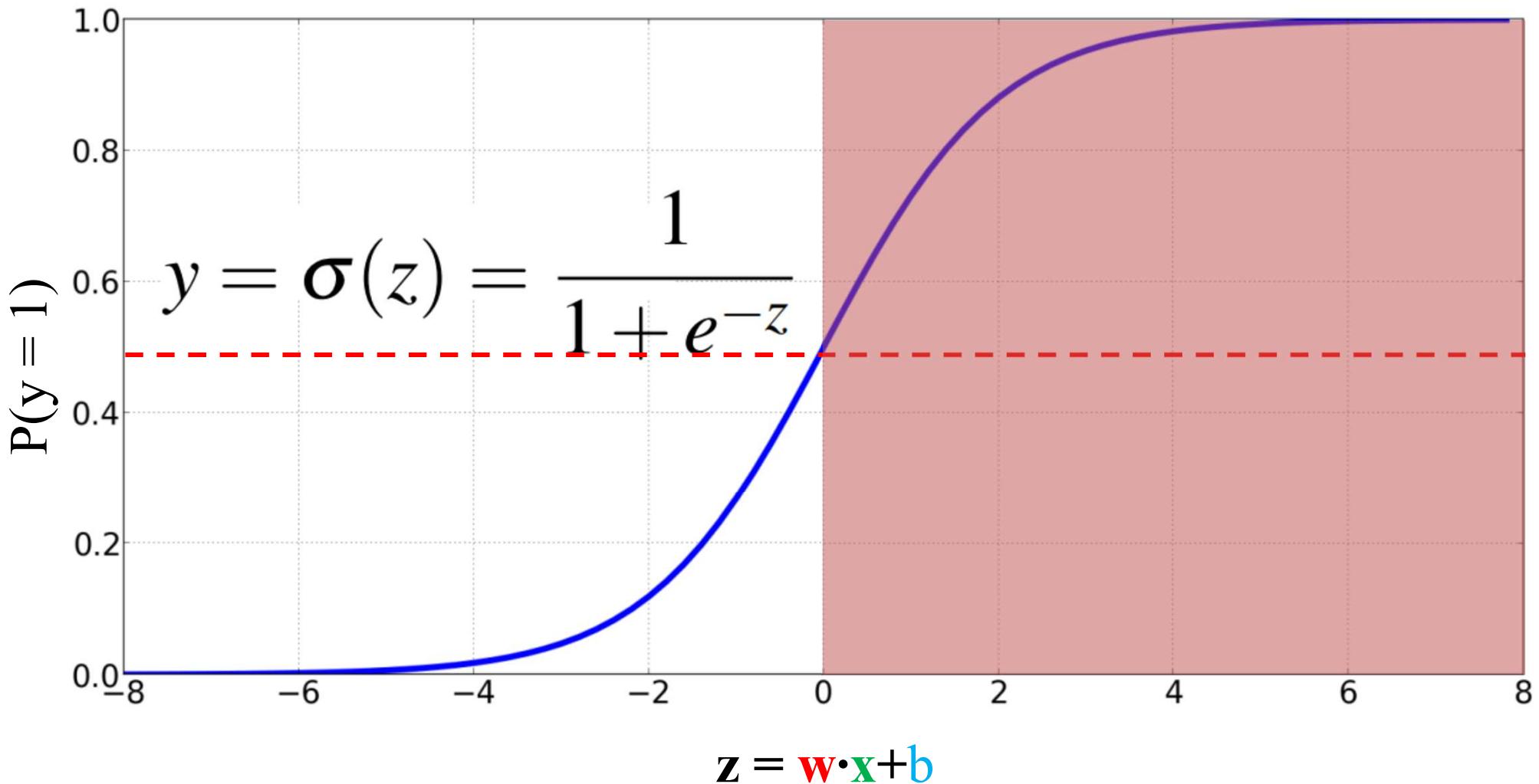
Probabilities → Classification

Once we know the probability, we can use it to classify:

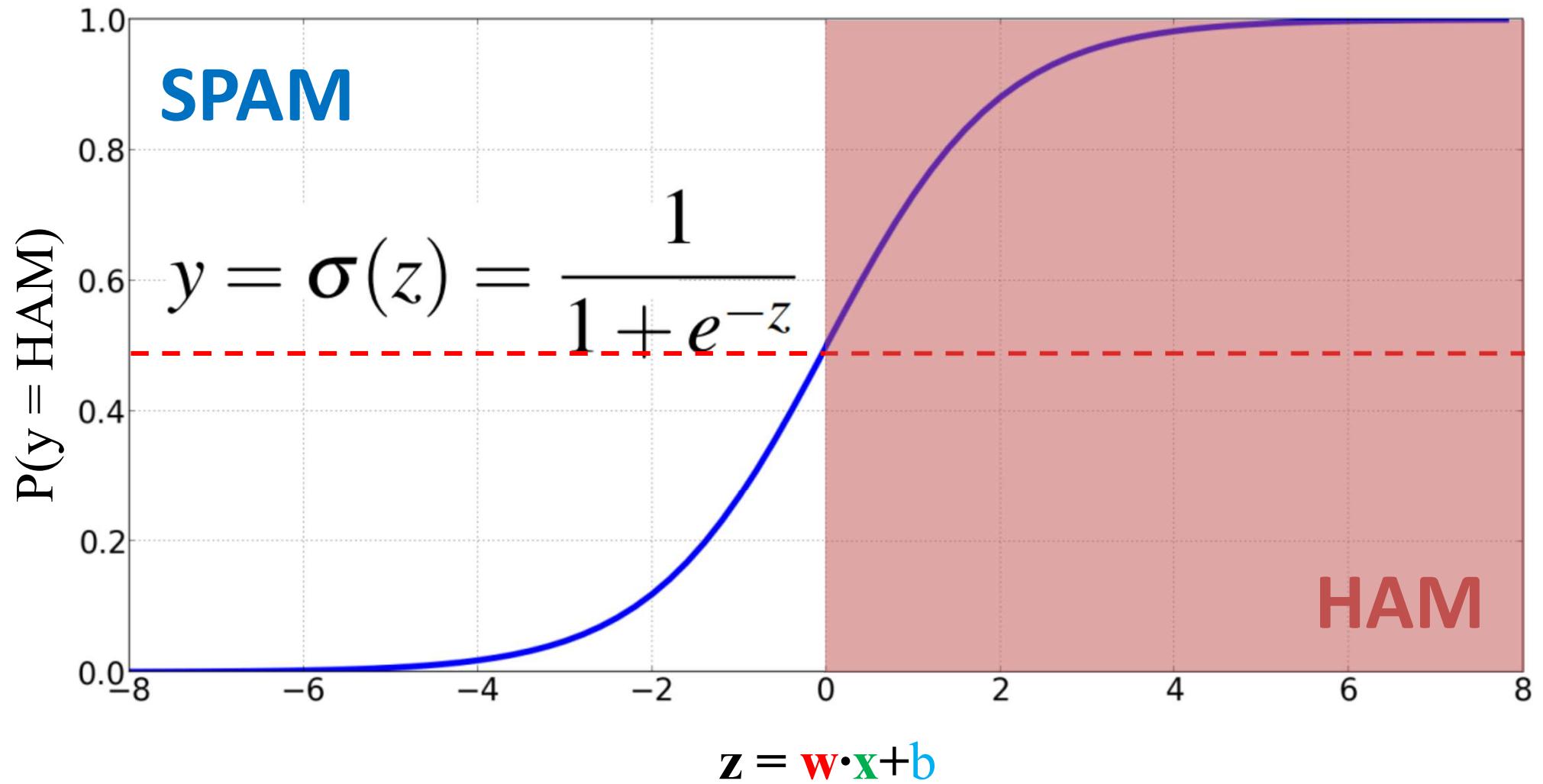
$$\hat{y} = \begin{cases} 1 & \text{if } P(y=1|x) > 0.5 \\ 0 & \text{otherwise} \end{cases}$$

Where 0.5 is the **decision boundary**

Logistic Regression Classifier



Logistic Regression Classifier



Probabilities → Classification

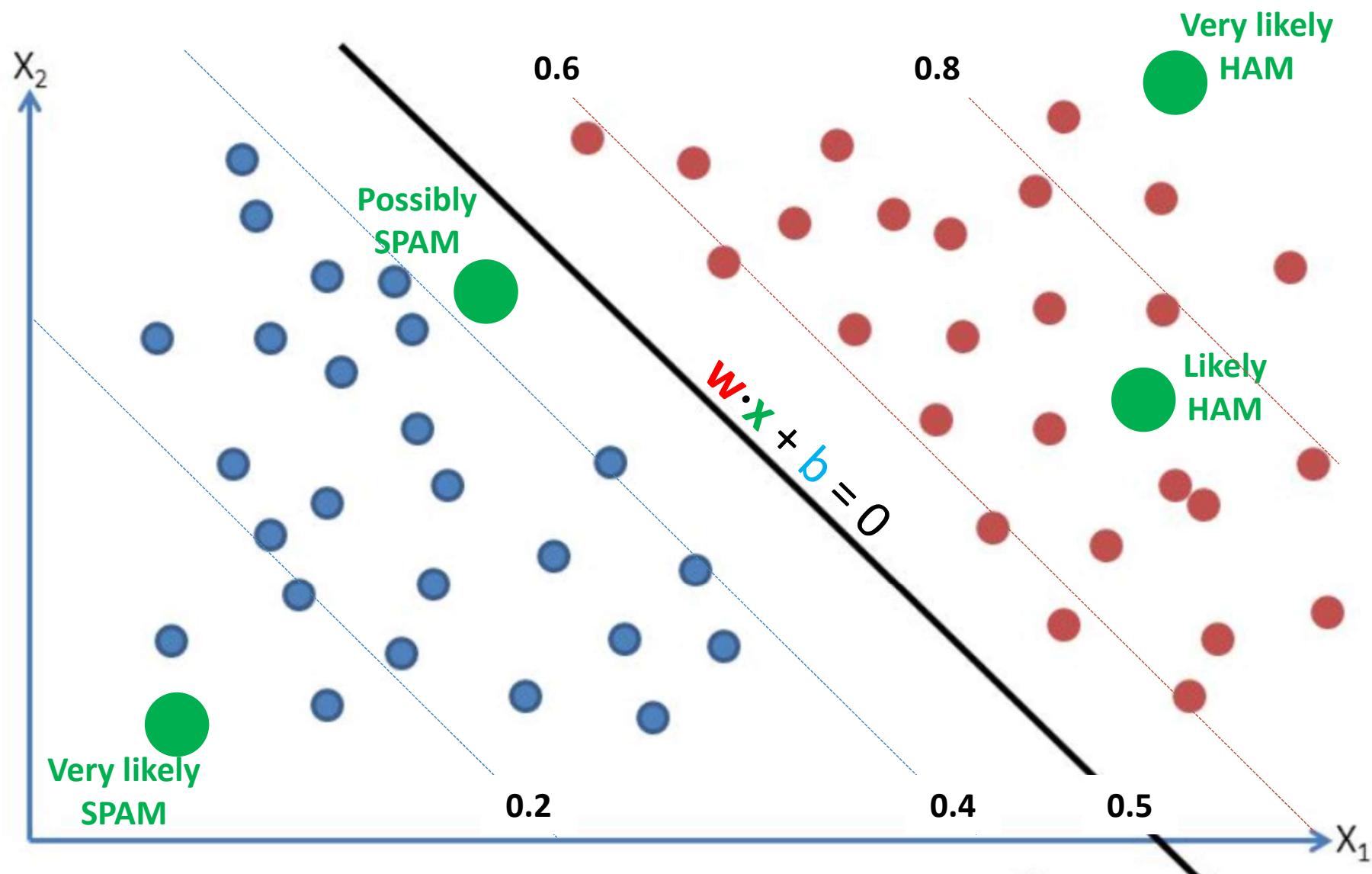
Once we know the probability, we can use it to classify:

$$\hat{y} = \begin{cases} 1 & \text{if } P(y=1|x) > 0.5 \\ 0 & \text{otherwise} \end{cases}$$

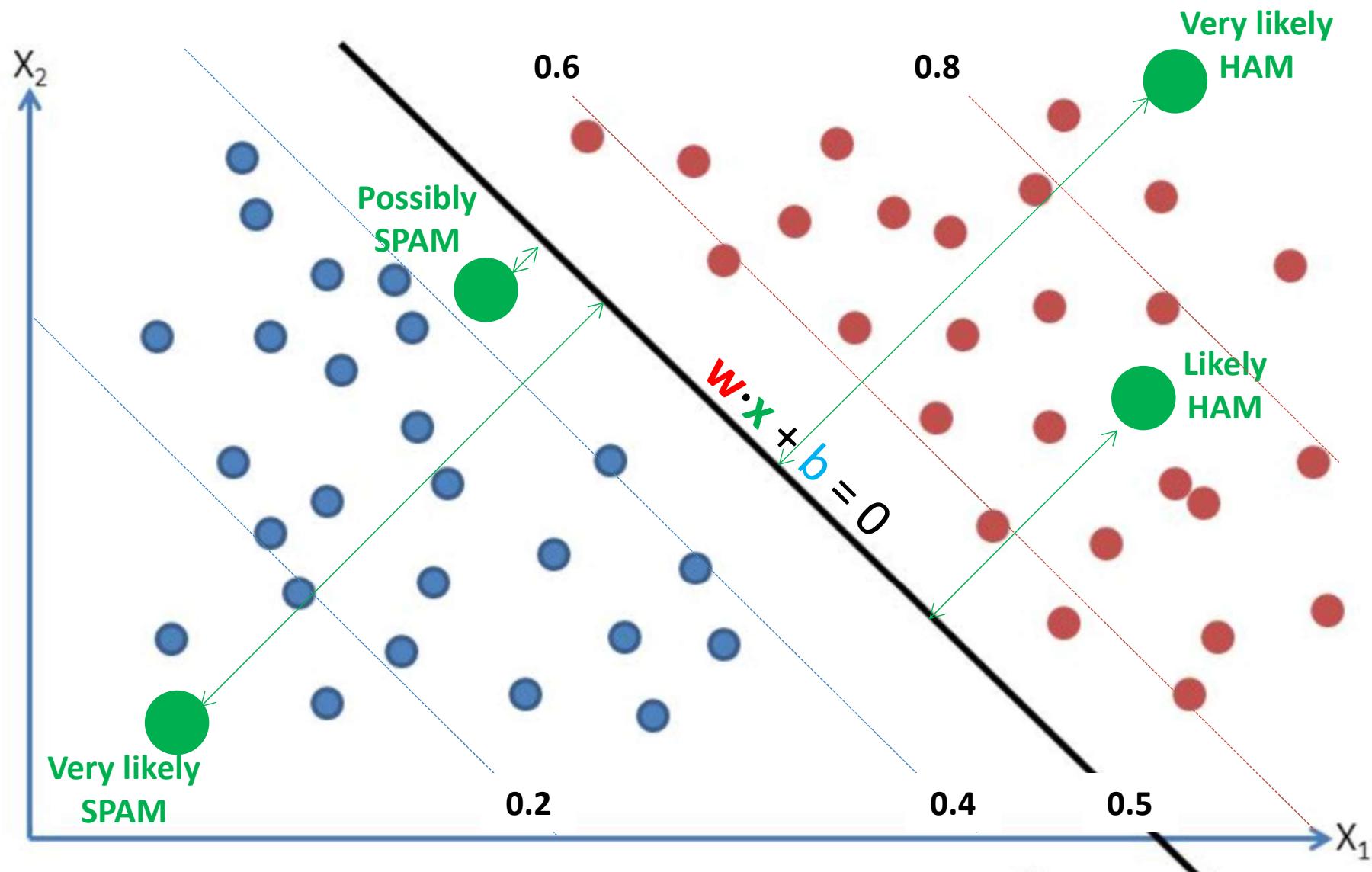
if $w \cdot x + b > 0$
if $w \cdot x + b \leq 0$

Where 0.5 is the **decision boundary**

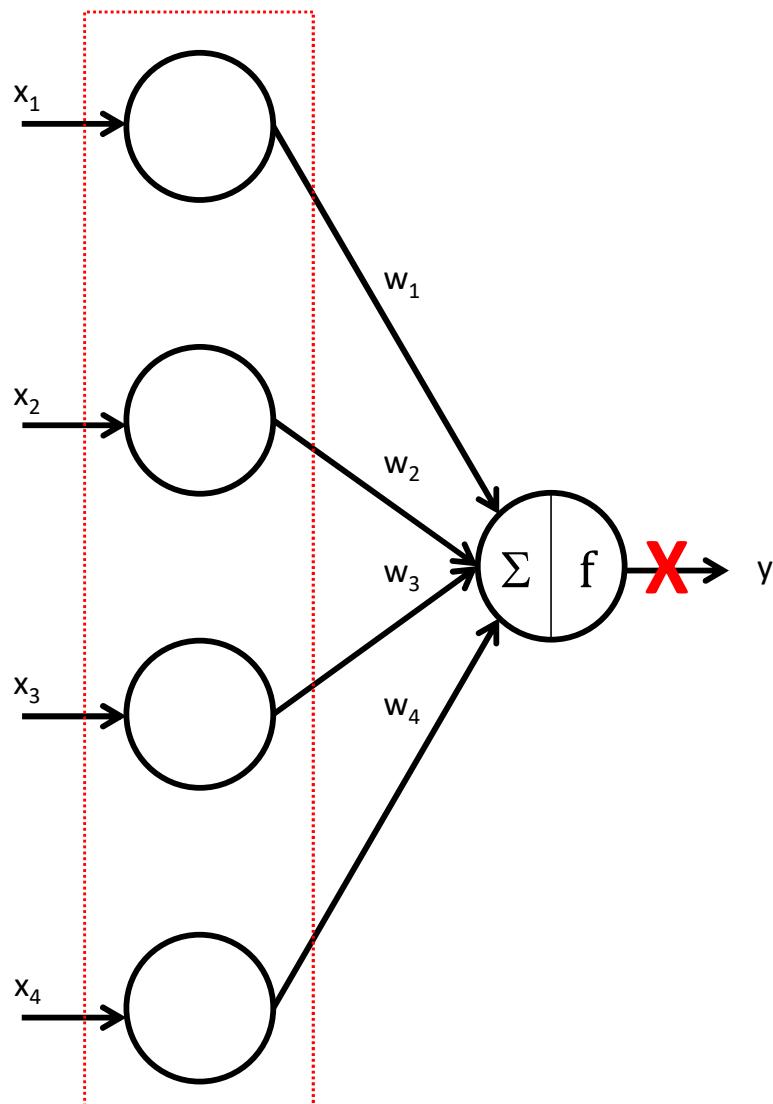
Text Classification: Separator



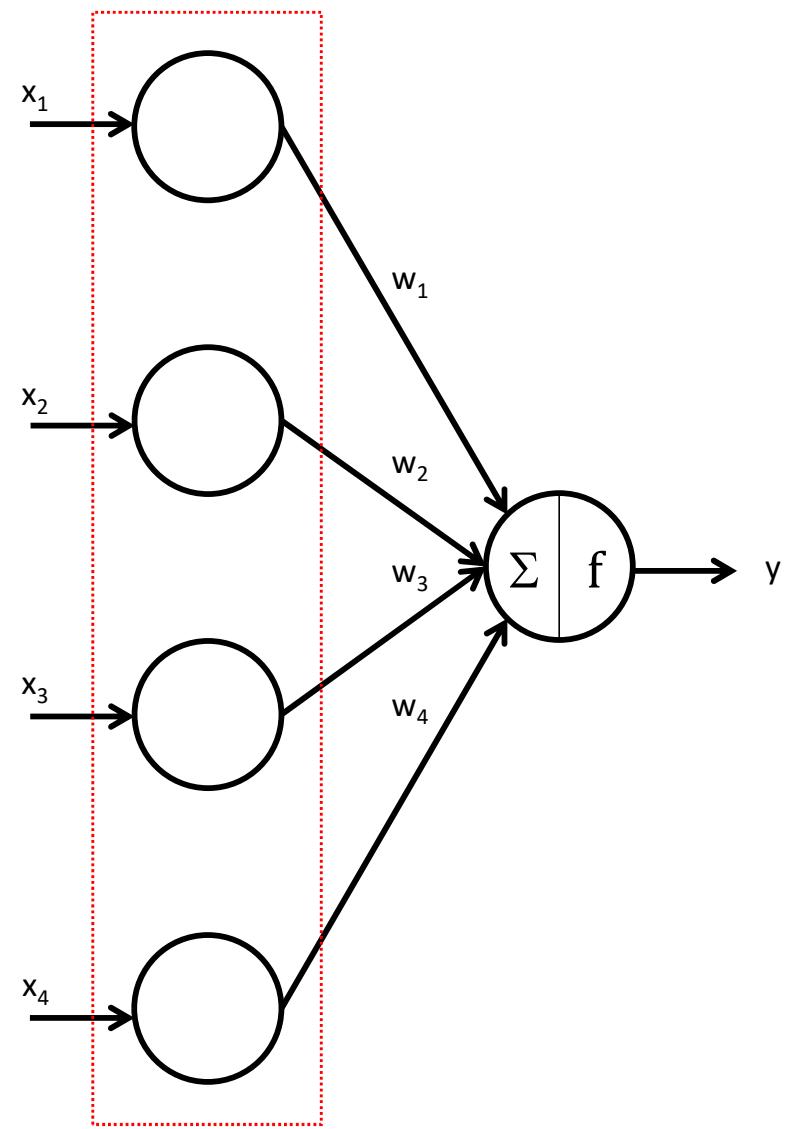
Text Classification: Separator



Does This Resemble Anything?

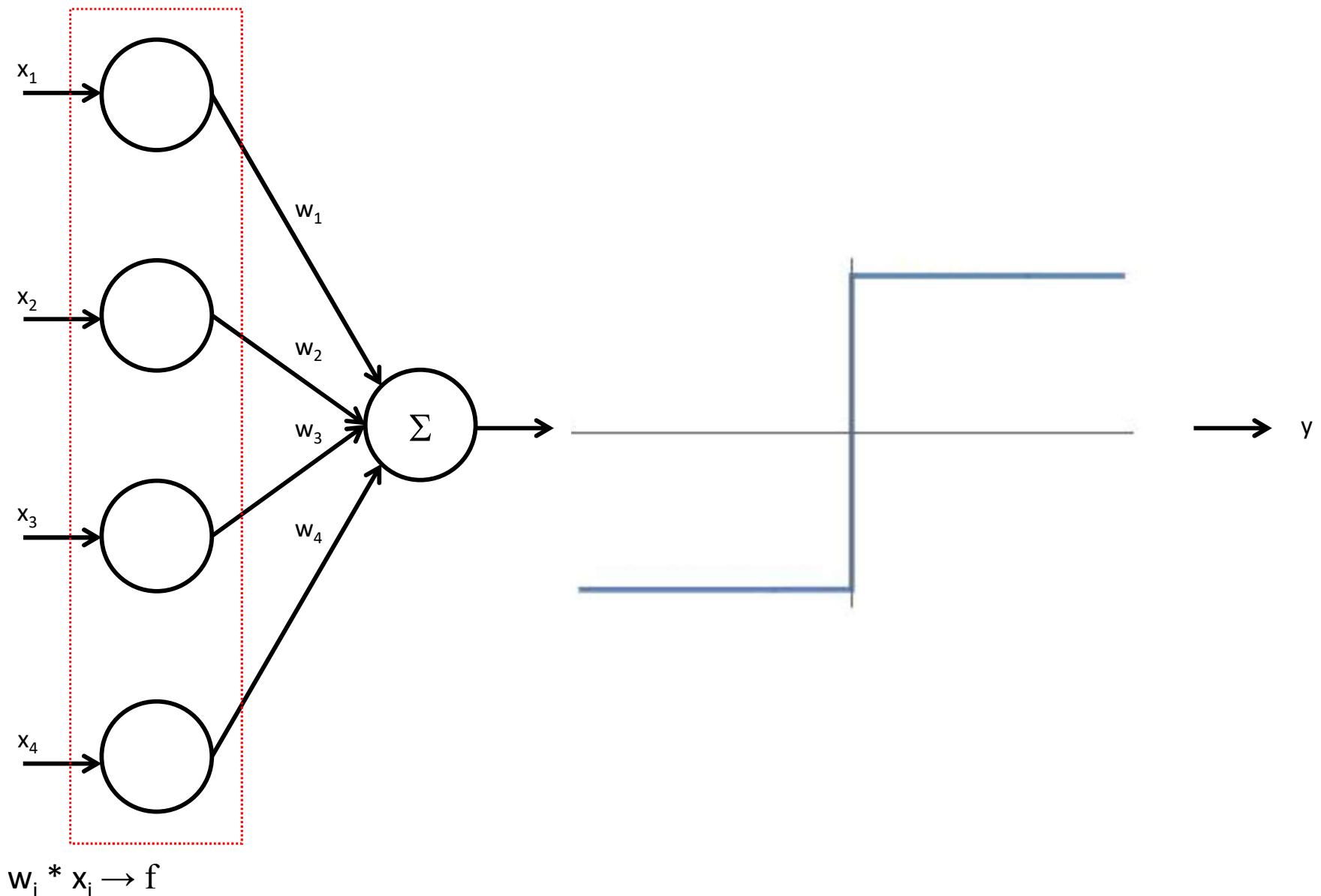


$\sum w_i * x_i < 0 \rightarrow f = 0 \rightarrow \text{NO}$

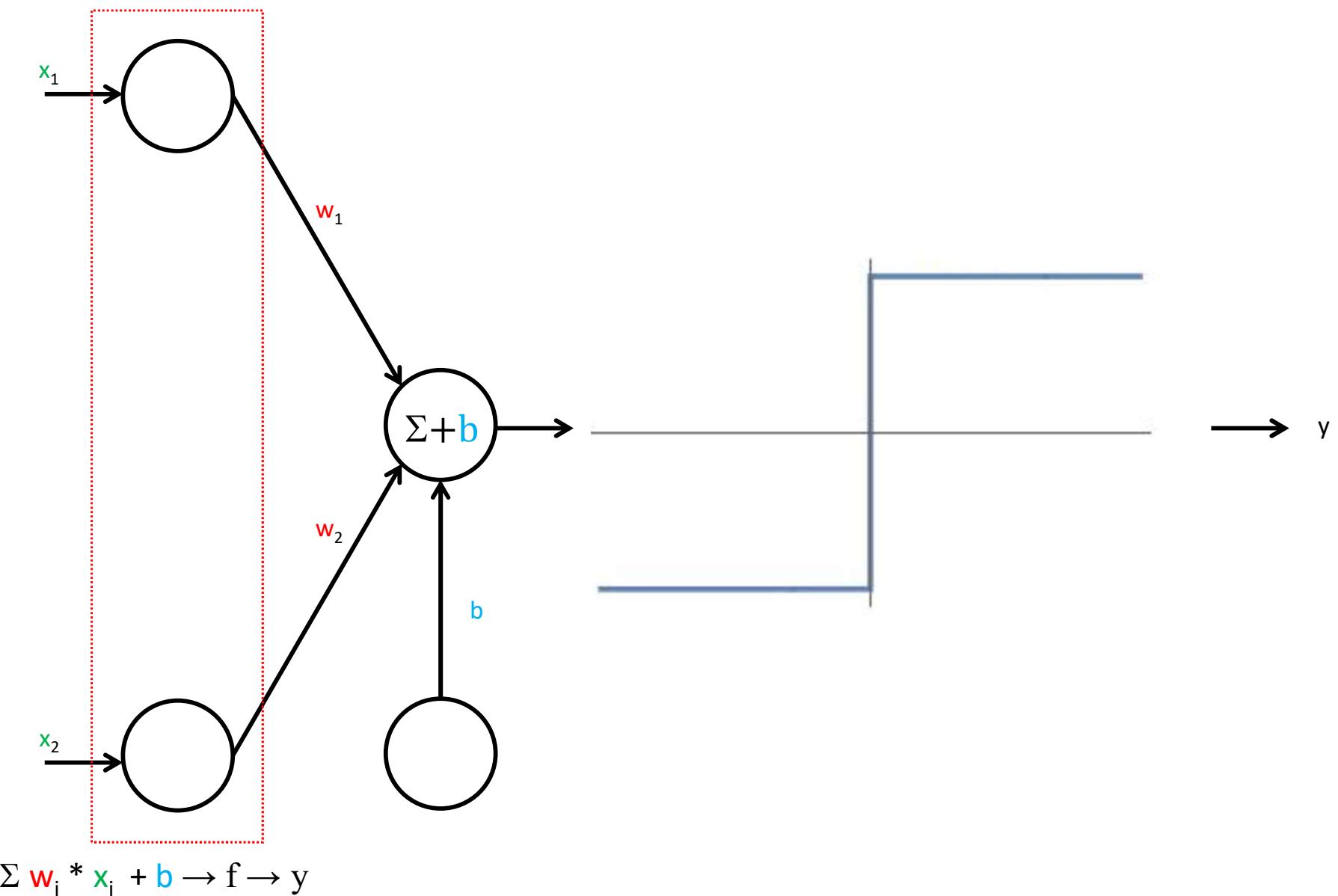


$\sum w_i * x_i \geq 0 \rightarrow f = 1 \rightarrow \text{YES}$

Step Activation Function



Step Activation Function



Sigmoid Activation Function

