CS 481

Artificial Intelligence Language Understanding

February 28, 2023

Announcements / Reminders

- Please follow the Week 07 To Do List instructions
- Written Assignment #02 due on Thursday (03/02/23) at 11:59 PM CST

- Exam dates:
  - Midterm: 03/02/2023 during Thursday lecture time
  - Final: 04/27/2023 during Thursday lecture time

Plan for Today

- Naïve Bayes as a language model
- Text classification
  - other types of classifiers
  - logistic regression

NB Sentiment Analysis: Example

Training set

$\mathbf{x}_1$	just plain boring	y_1= -
$\mathbf{x}_2$	entirely predictable and lacks energy	y_2= -
$\mathbf{x}_3$	no surprises and very few laughs	y_3= -
$\mathbf{x}_4$	very powerful	y_4= +
$\mathbf{x}_5$	the most fun film of the summer	y_5= +

Test set

$\mathbf{x}_6$	predictable ~~with~~ no fun	y_6=???

unknown word (not in V) → ignore

Model

+

$$P(y = +) = \frac{2}{5}$$

$$P(x_i = predictable \mid y = +) = \frac{1}{29}$$

$$P(x_i = no \mid y = +) = \frac{1}{29}$$

$$P(x_i = fun \mid y = +) = \frac{2}{29}$$

-

$$P(y = -) = \frac{3}{5}$$

$$P(x_i = predictable \mid y = -) = \frac{2}{34}$$

$$P(x_i = no \mid y = -) = \frac{2}{34}$$

$$P(x_i = fun \mid y = -) = \frac{1}{34}$$

$\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \ldots, \mathbf{x}_{N-2}, \mathbf{x}_{N-1}, \mathbf{x}_N$ - feature vectors (in **bold**) | $y_1, y_2, y_3, \ldots, y_{N-2}, y_{N-1}, y_N$ - labels

Naive Bayes: Language Model

$$y_{MAP} \propto \underset{y \in Y}{argmax} \left(\boldsymbol{P}(y) * \prod_{i=1}^{N} \boldsymbol{P}(x_i \mid y) \right)$$

If we use all words to create a model, Naive Bayes model resembles a (+, -) language model

Model pos	
0.1	I
0.1	love
0.01	this
0.05	fun
0.1	film

Model neg	
0.2	I
0.001	love
0.01	this
0.005	fun
0.1	film

I	love	this	fun	film
0.1	0.1	0.01	0.05	0.1
0.2	0.001	0.01	0.005	0.1

P(s|pos) > P(s|neg)

Naive Bayes: Language Model

$$y_{MAP} \propto \underset{y \in Y}{argmax} \left(\boldsymbol{P}(y) * \prod_{i=1}^{N} \boldsymbol{P}(x_i \mid y) \right)$$

With (+, -) language models we can GENERATE new sentences (samples) that belong to (+, -) classes.

Model pos	
0.1	I
0.1	love
0.01	this
0.05	fun
0.1	film

Model neg	
0.2	I
0.001	love
0.01	this
0.005	fun
0.1	film

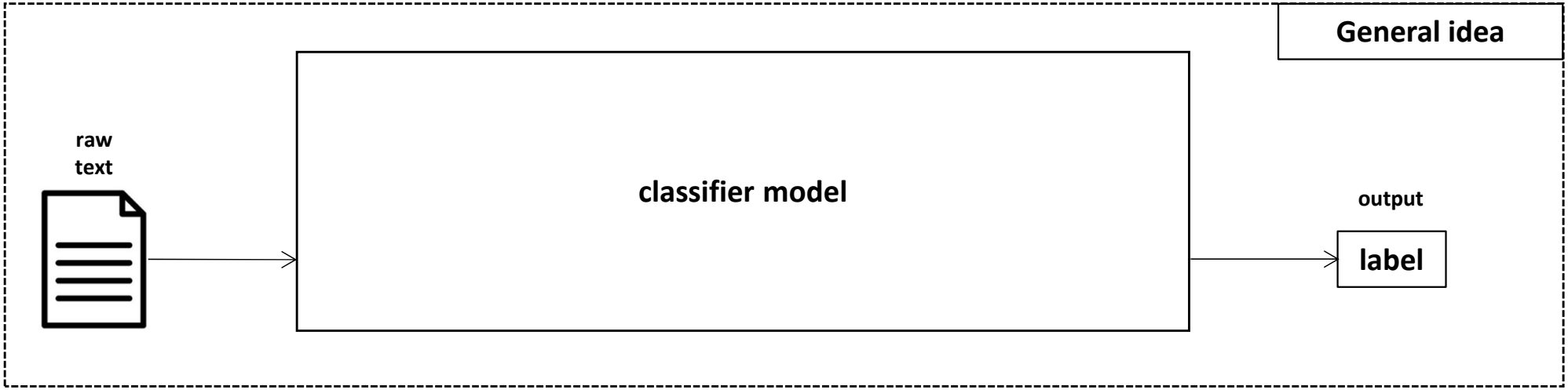
I	love	this	fun	film
0.1	0.1	0.01	0.05	0.1
0.2	0.001	0.01	0.005	0.1

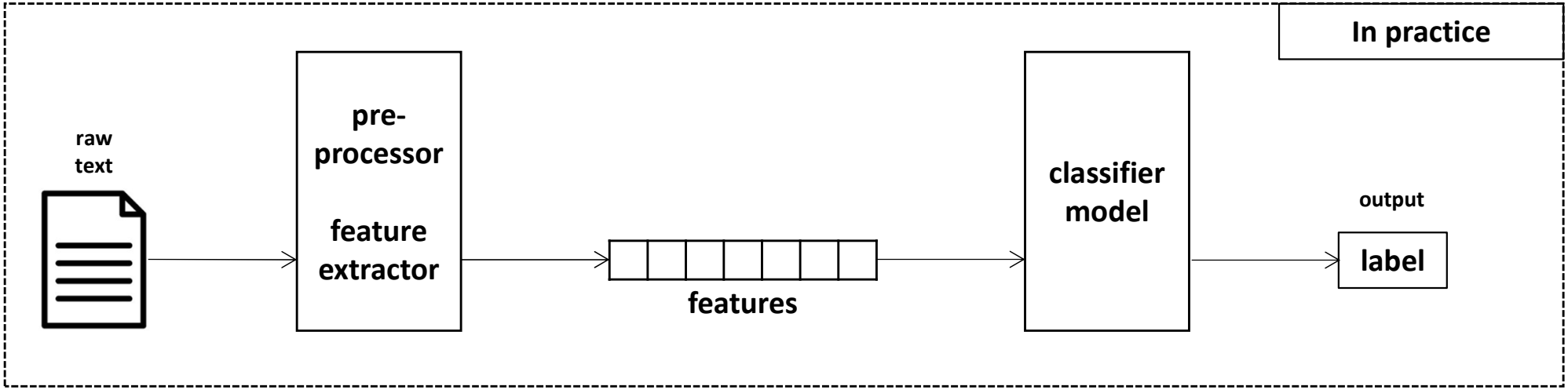
P(s|pos) > P(s|neg)

Text Classification: Supervised ML

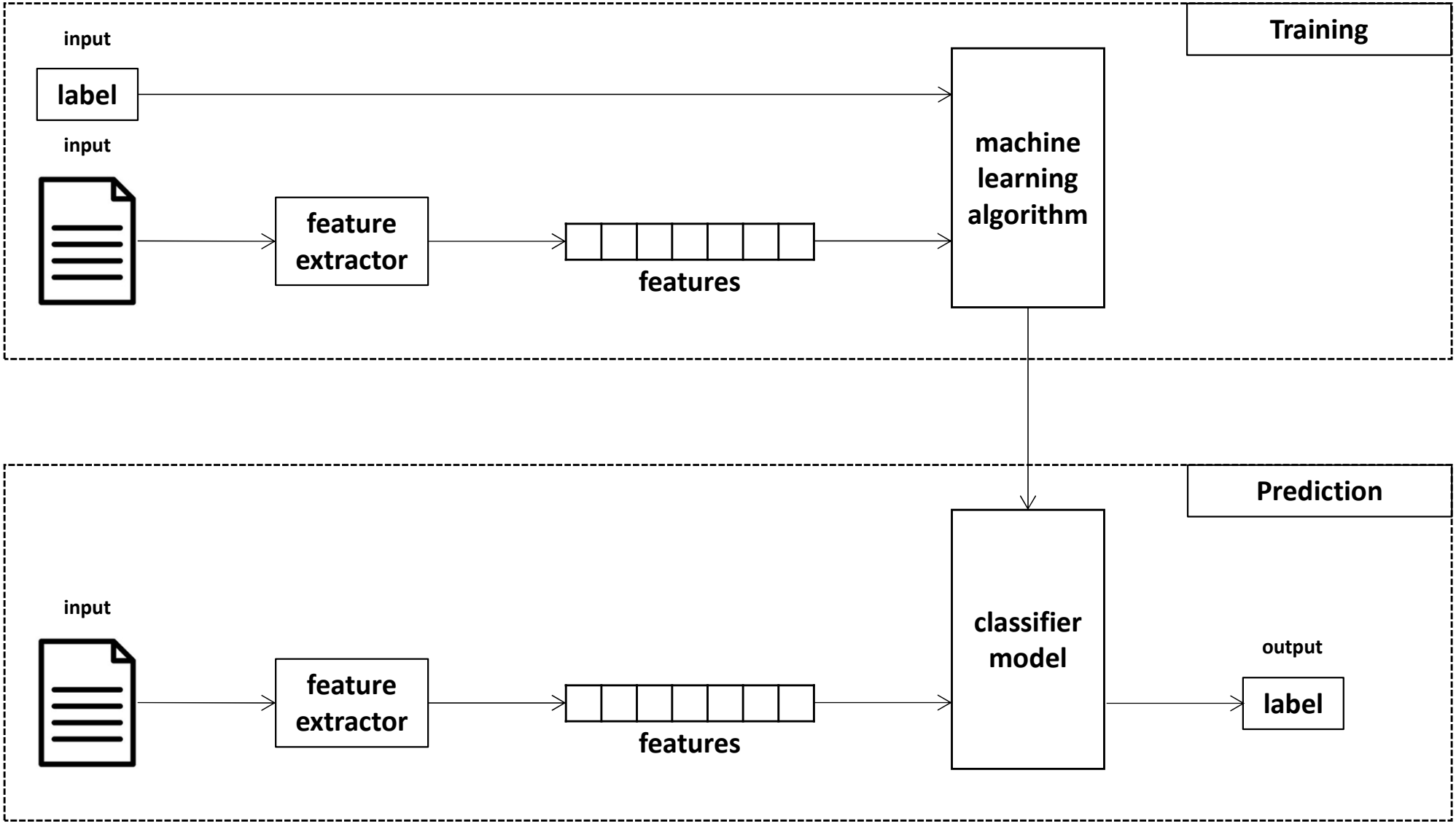
- Various Machine Learning supervised learning classifier approaches can be employed:
  - Naïve Bayes
  - Logistic regression
  - Neural networks
  - k-Nearest Neighbors
  - etc.

Text Classification: the Idea



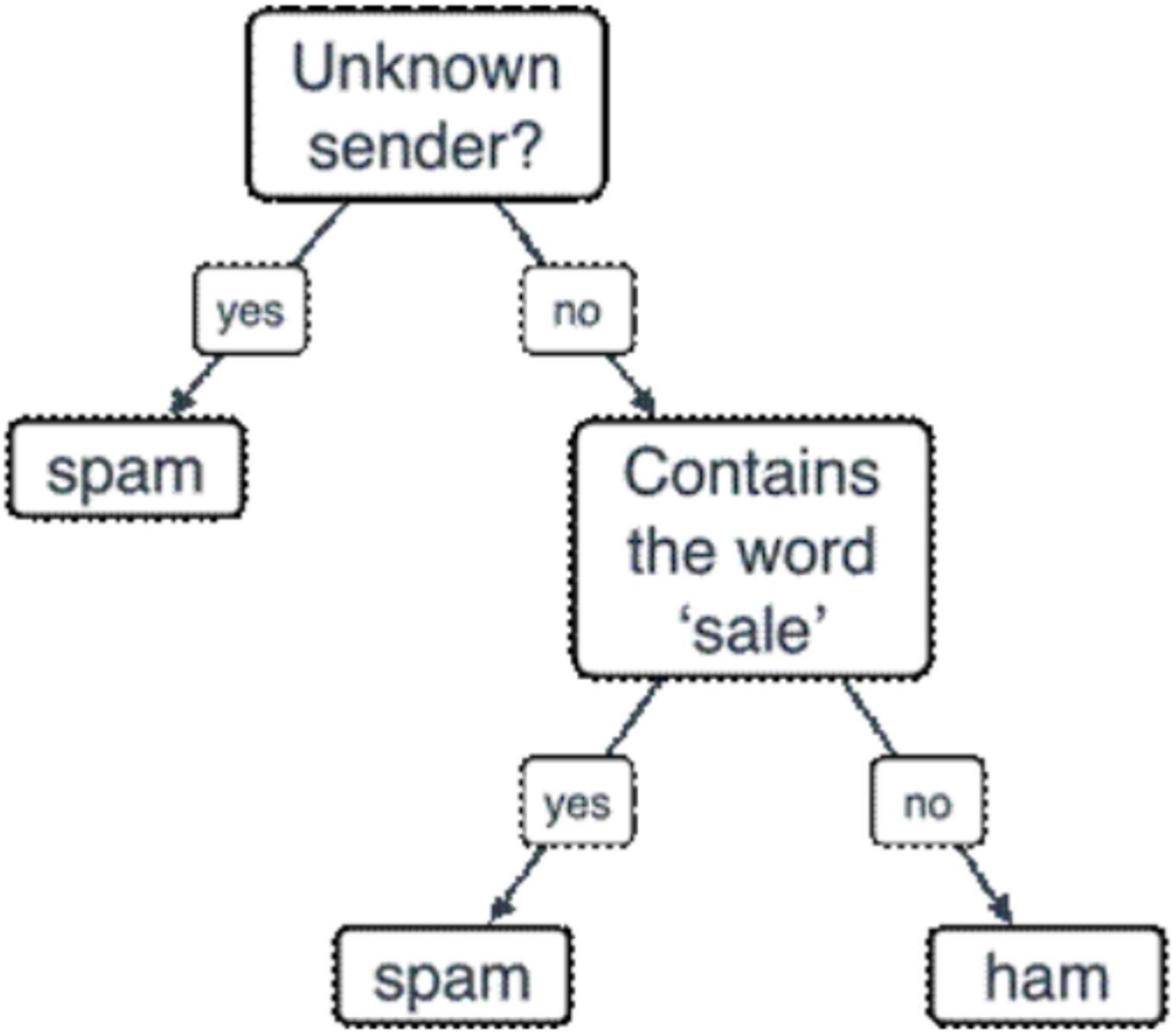


Supervised Learning with ML



Classification with Decision Trees

Decision Tree: Spam Filter



Classification with Linear and Nonlinear “Separators”

What is Regression?

Definition:

A technique for **estimating the relationship between a dependent variable** ("outcome") **and one or more independent variables** ("predictors" or "**features**"). The most common form is **linear regression**, in which one **finds the line** (or a more complex linear combination) **that most closely fits the data** (for example using the least-squares method).

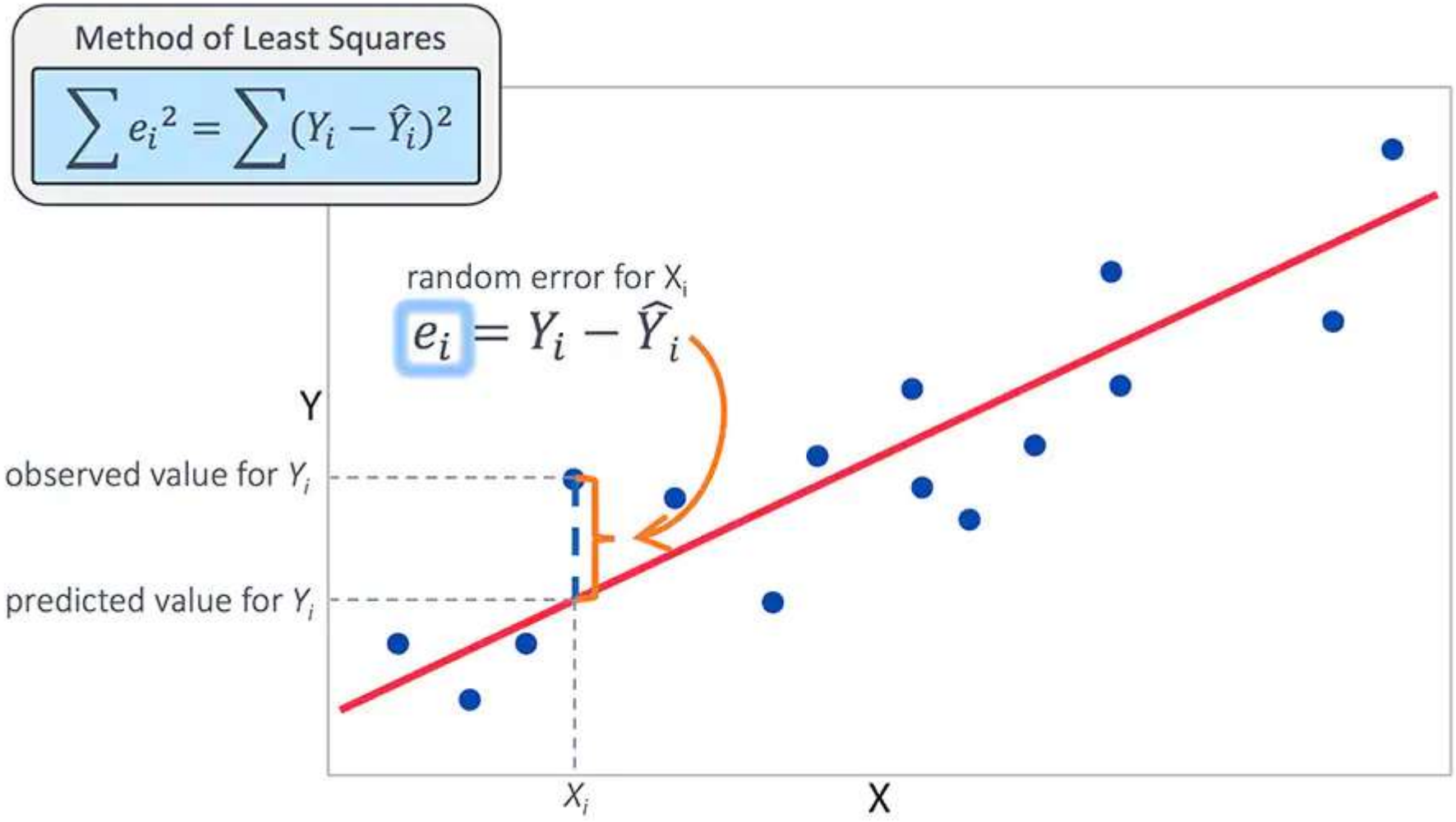
Source: https://en.wikipedia.org/wiki/Regression_analysis

Origins of ‘Regression’ Term

Source: https://en.wikipedia.org/wiki/Francis_Galton

Sir Francis Galton, an English polymath studied, among other things, heredity in humans. In one experiment he compared children height to their parent heights. He observed that children heights **regressed** towards the average height of an adult.

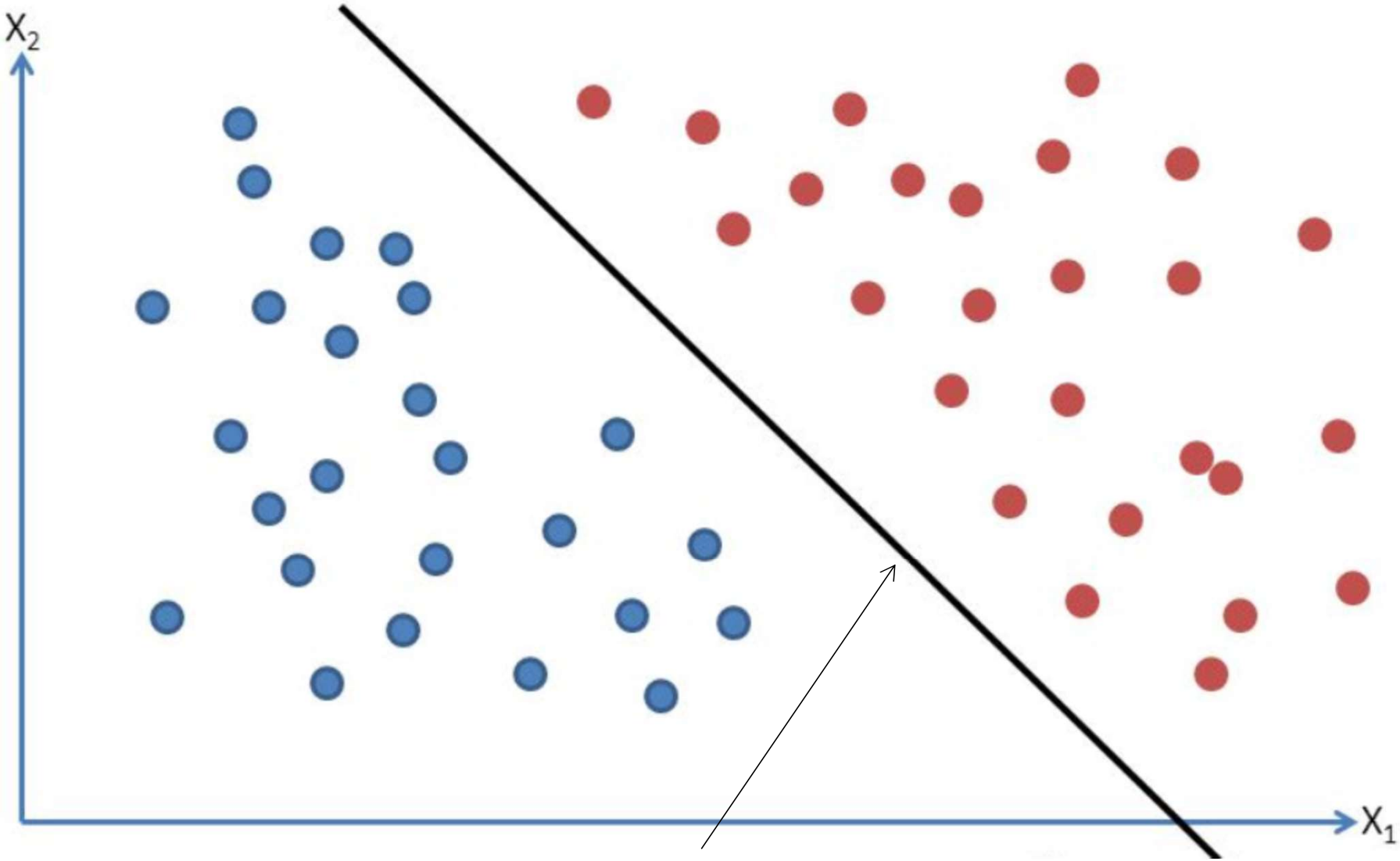
Linear Regression Using Least-Squares



The goal is to find the line y = ax + b that minimizes the amount of error.

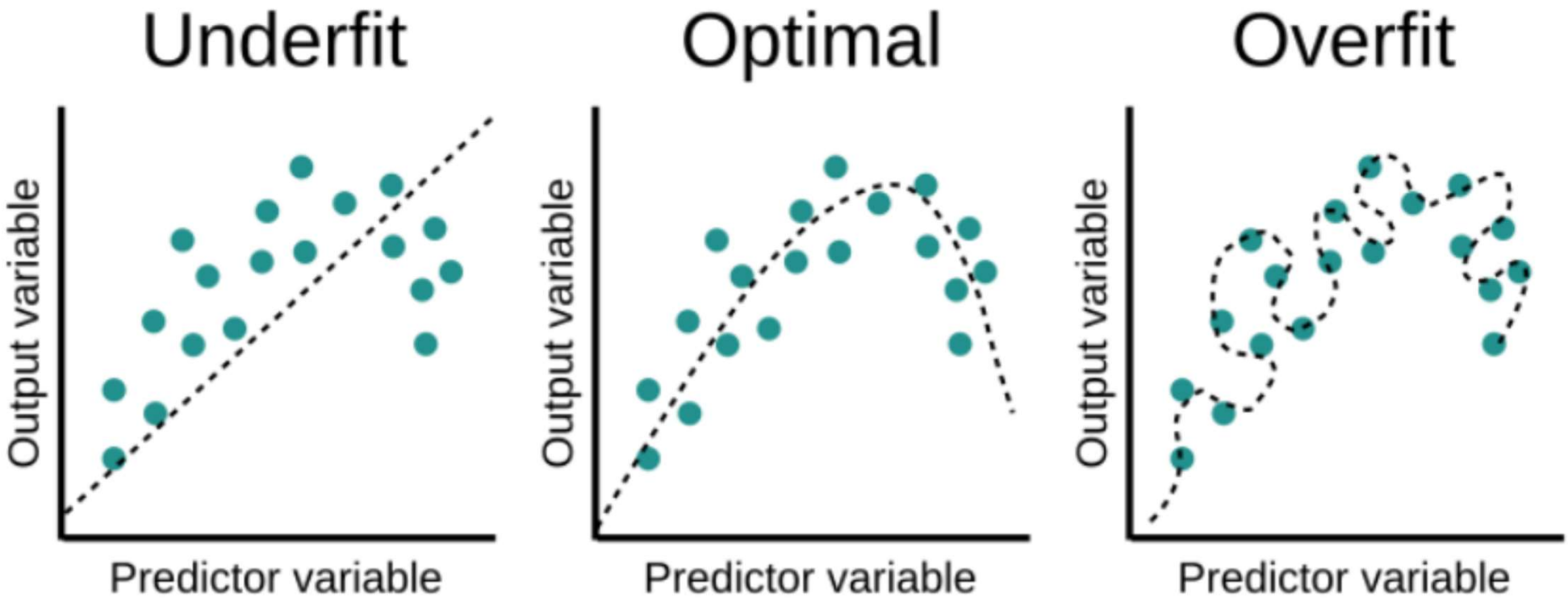
Source: https://www.jmp.com/en_us/statistics-knowledge-portal/what-is-multiple-regression/fitting-multiple-regression-model.html

Linear Separator



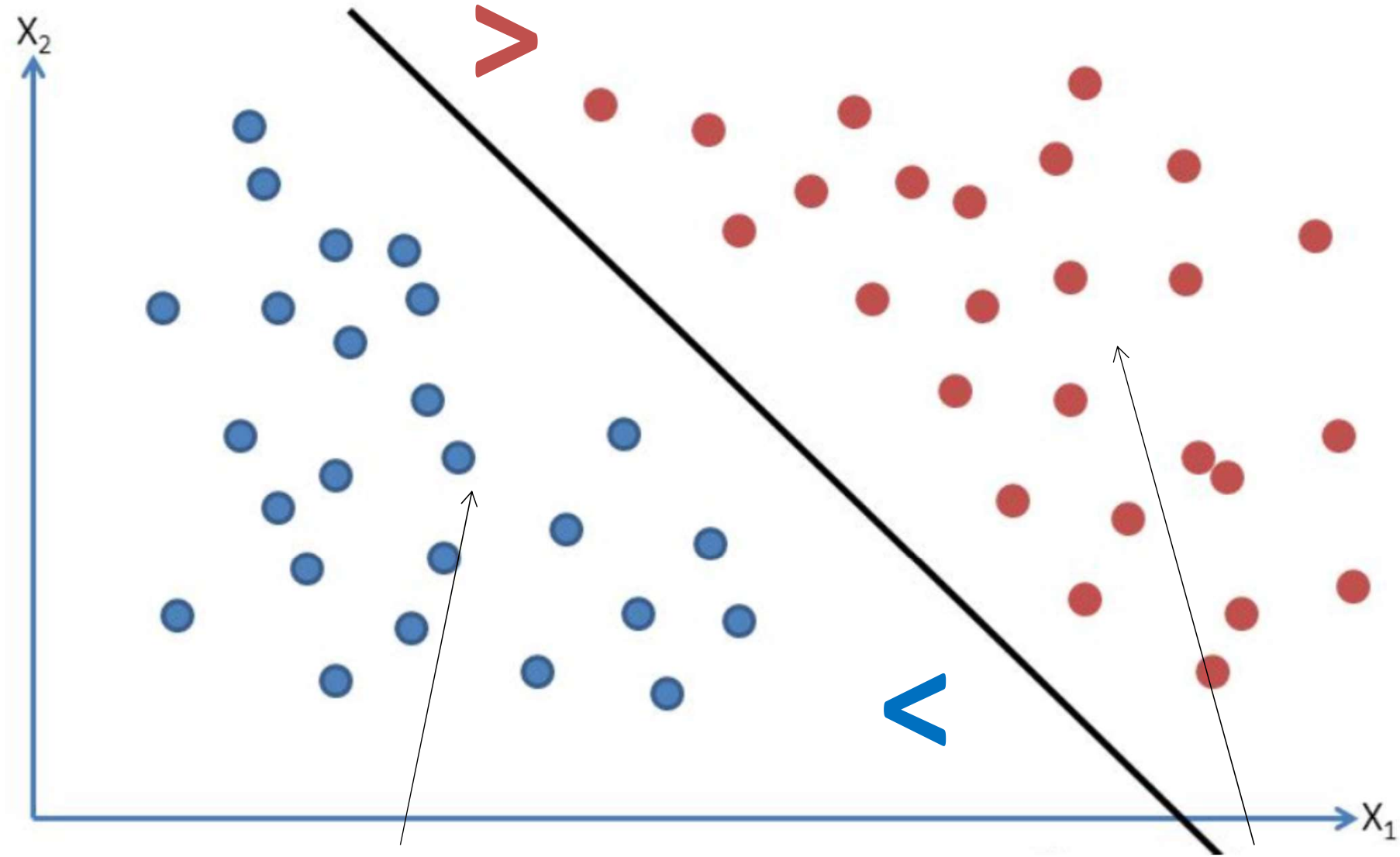
Line in 2D, plane in 3D, hyperplane in ND

Regression: Underfitting / Overfitting



Source: https://livebook.manning.com/book/machine-learning-for-mortals-mere-and-otherwise/chapter-9/v-4/29

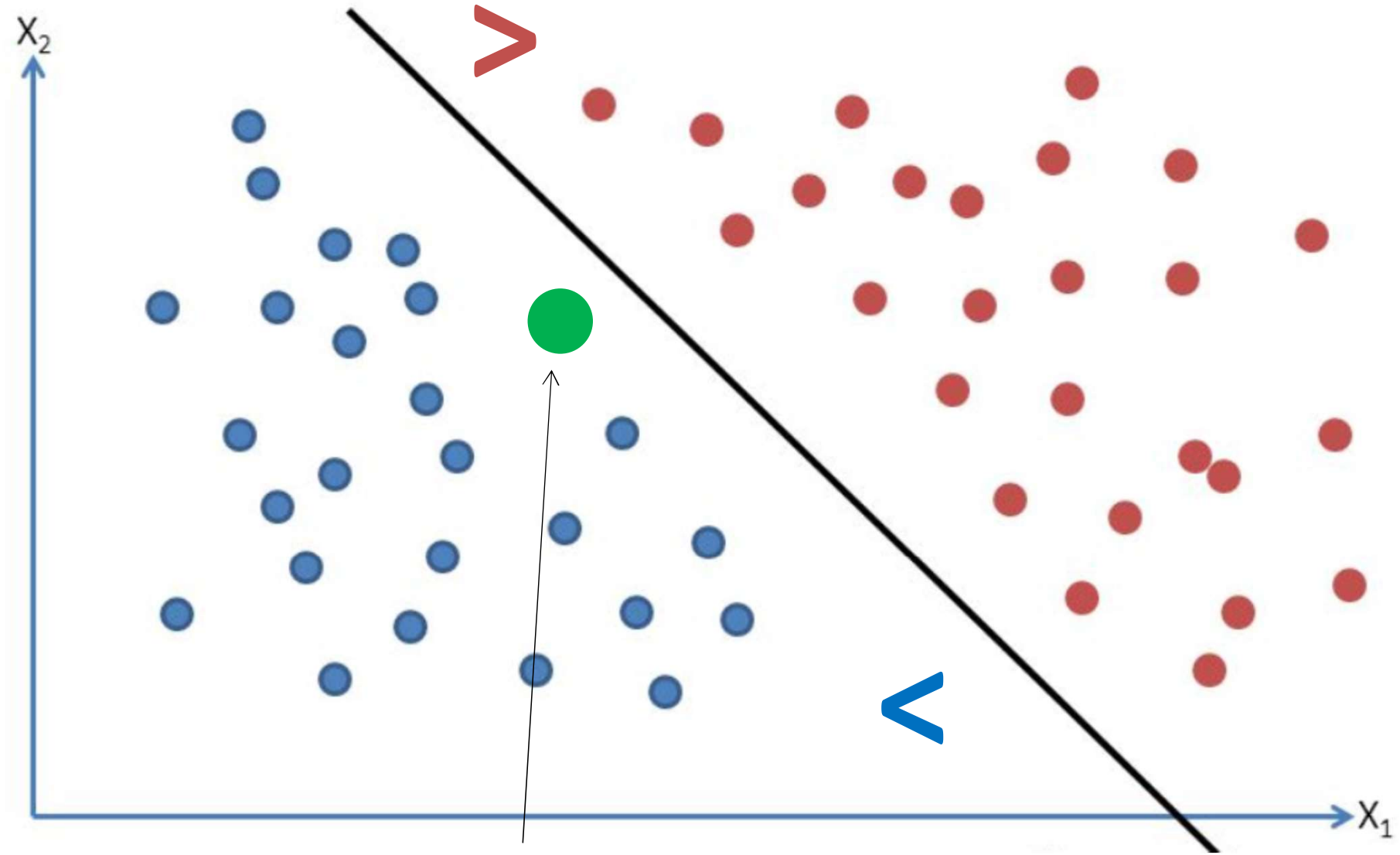
Linear Separator



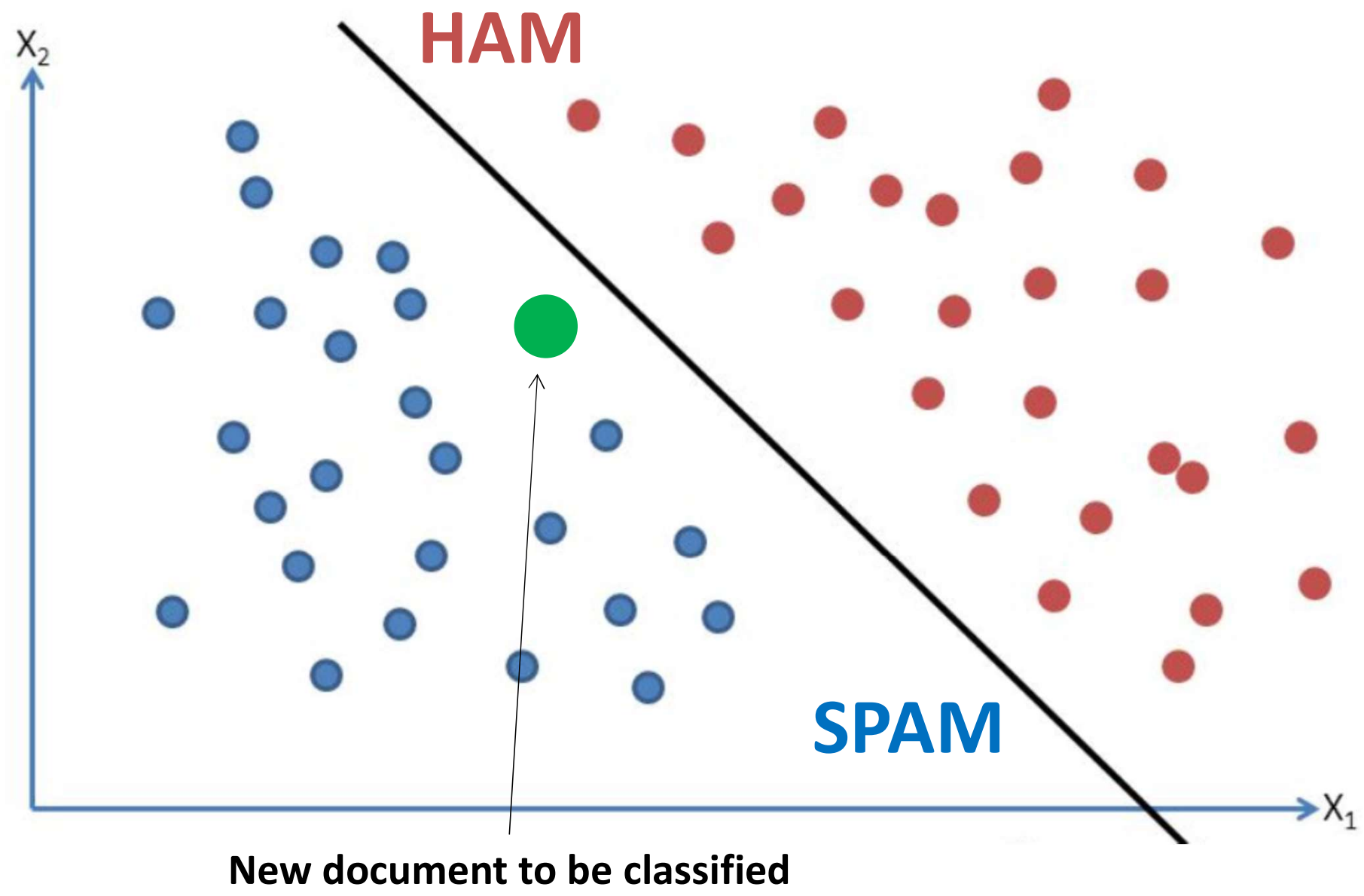
"Below" the separator

"Above" the separator

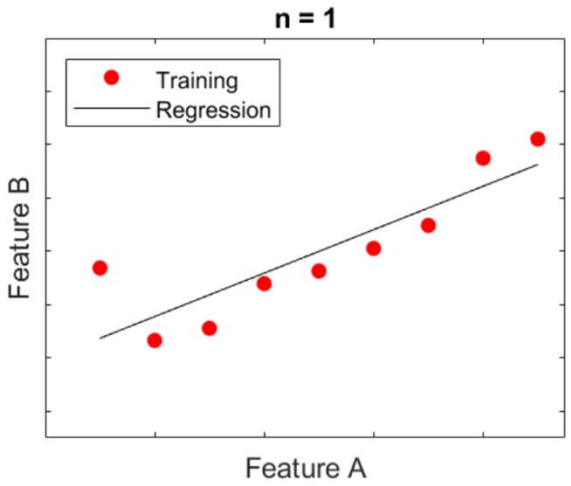
Text Classification: Linear Separator



Text Classification: Linear Separator



Univariate Linear Regression



Real function: $y = w_1 * x + w_0$

Hypothesis / model: $h_{\boldsymbol{w}}(x) = w_1 * x + w_0$

where $\boldsymbol{w} = \langle w_0, w_1 \rangle$ **are coefficients / weights.**

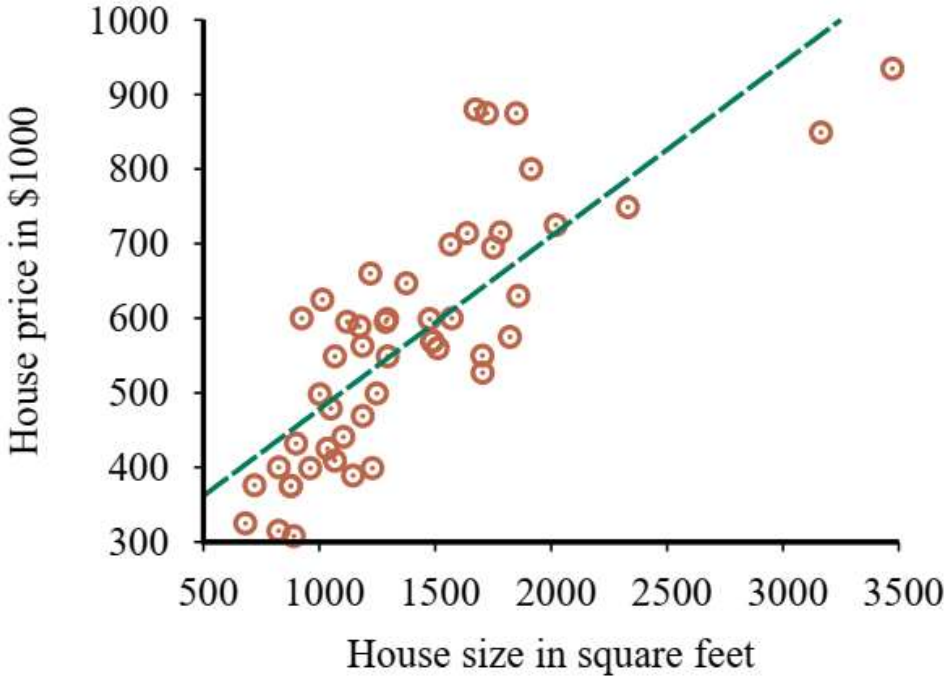
Squared-error loss function:

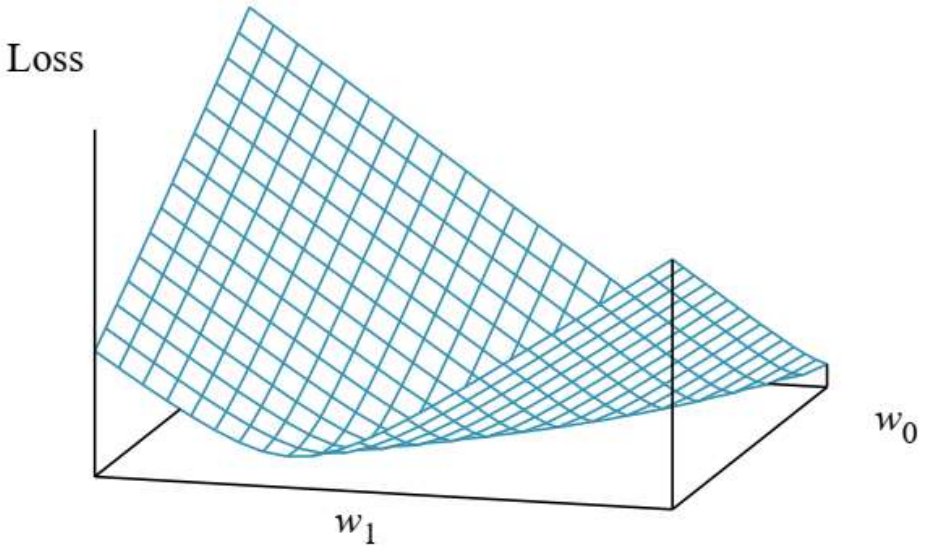
$$Loss(h_{\boldsymbol{w}}) = \sum_{j=1}^{N} \left(y_j - h_{\boldsymbol{w}}(x_j)\right)^2 = \sum_{j=1}^{N} \left(y_j - (w_1 * x_j + w_0)\right)^2$$

We want to find $\boldsymbol{w}^* = \underset{\boldsymbol{w}}{argmin}\, Loss(h_{\boldsymbol{w}})$:

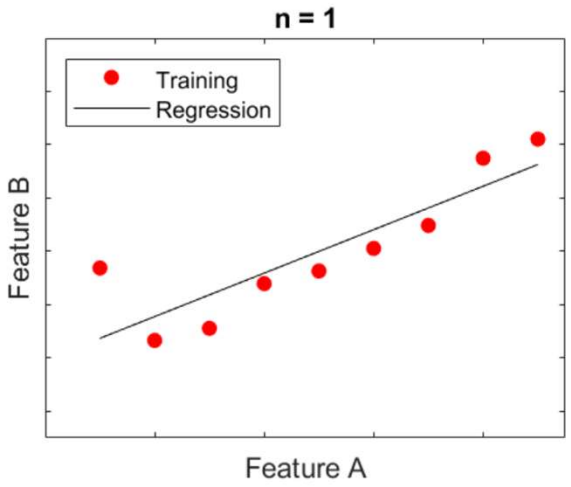
solve $\frac{\partial Loss(h_{\boldsymbol{w}})}{\partial w_0} = 0$, $\frac{\partial Loss(h_{\boldsymbol{w}})}{\partial w_1} = 0$

Weight / Parameter Space





Gradient Descent



Given a squared-error loss function:

$$Loss(h_{\boldsymbol{w}}) = \sum_{j=1}^{N} \left(y_j - h_{\boldsymbol{w}}(x_j)\right)^2 = \sum_{j=1}^{N} \left(y_j - (w_1 * x_j + w_0)\right)^2$$

We want to find $\boldsymbol{w}^* = \underset{\boldsymbol{w}}{argmin}\, Loss(h_{\boldsymbol{w}})$:

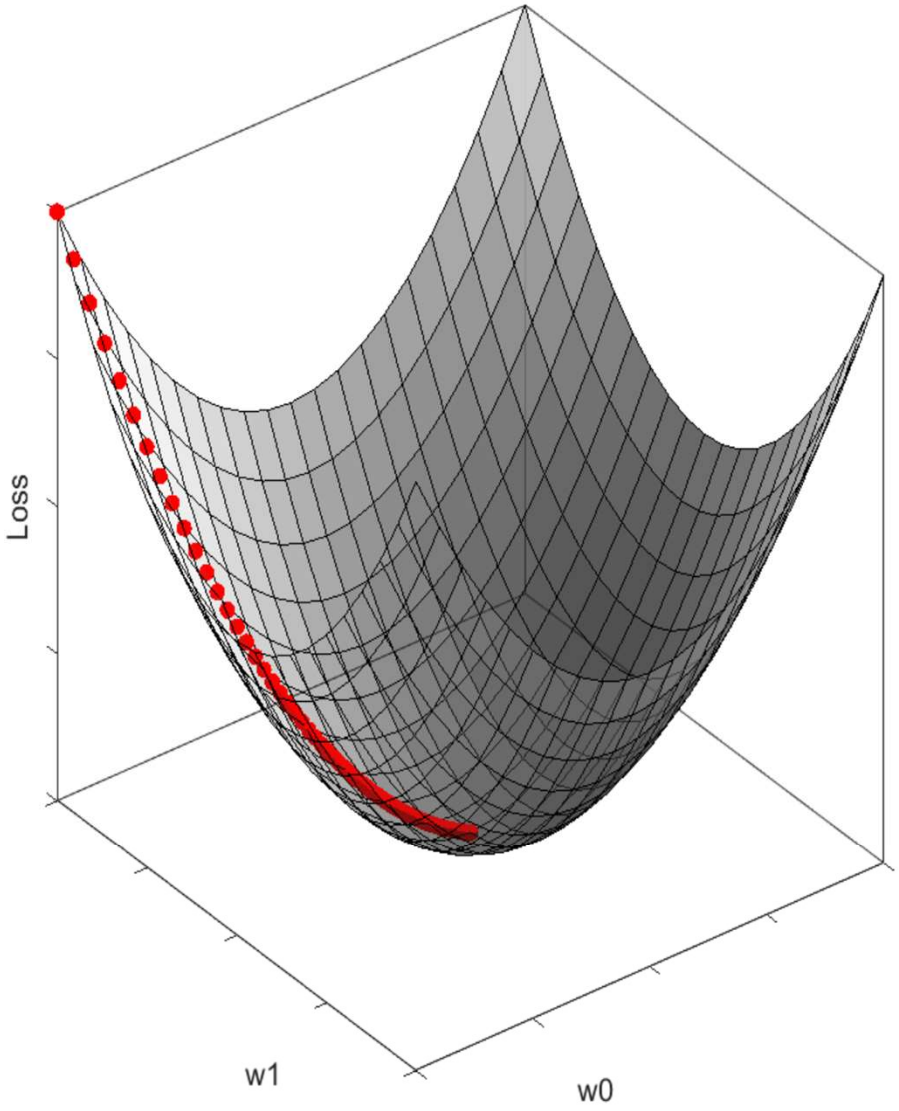
solve $\frac{\partial Loss(h_{\boldsymbol{w}})}{\partial w_0} = 0\,, \frac{\partial Loss(h_{\boldsymbol{w}})}{\partial w_1} = 0$

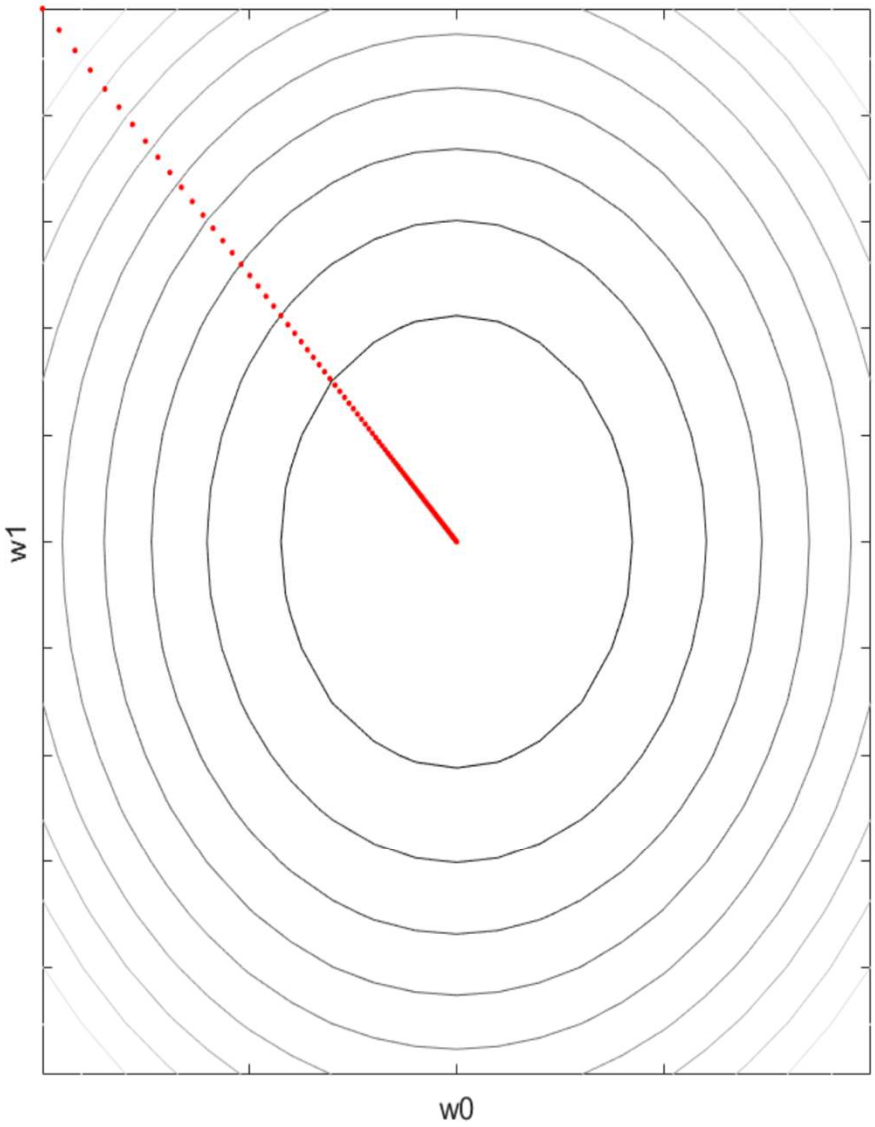
Sometimes these will be hard (or impossible) to solve. Gradient descent technique can be instead:

w ← any point in the parameter space
while not converged **do**
for each w_i **in w do** $w_i \leftarrow w_i - \alpha * \frac{\partial Loss(\boldsymbol{w})}{\partial\, w_i}$

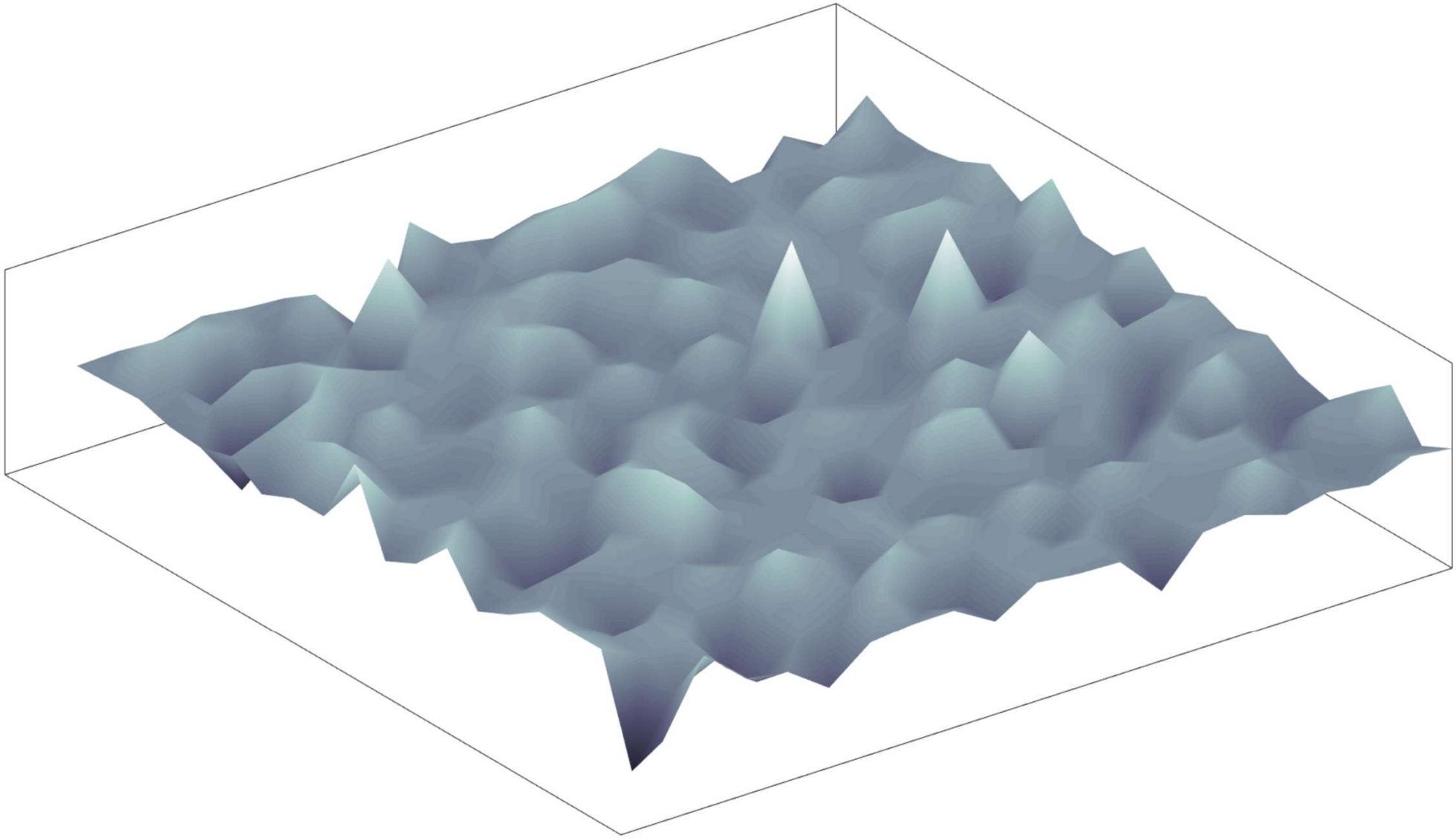
α **- step size / learning rate**

Gradient Descent



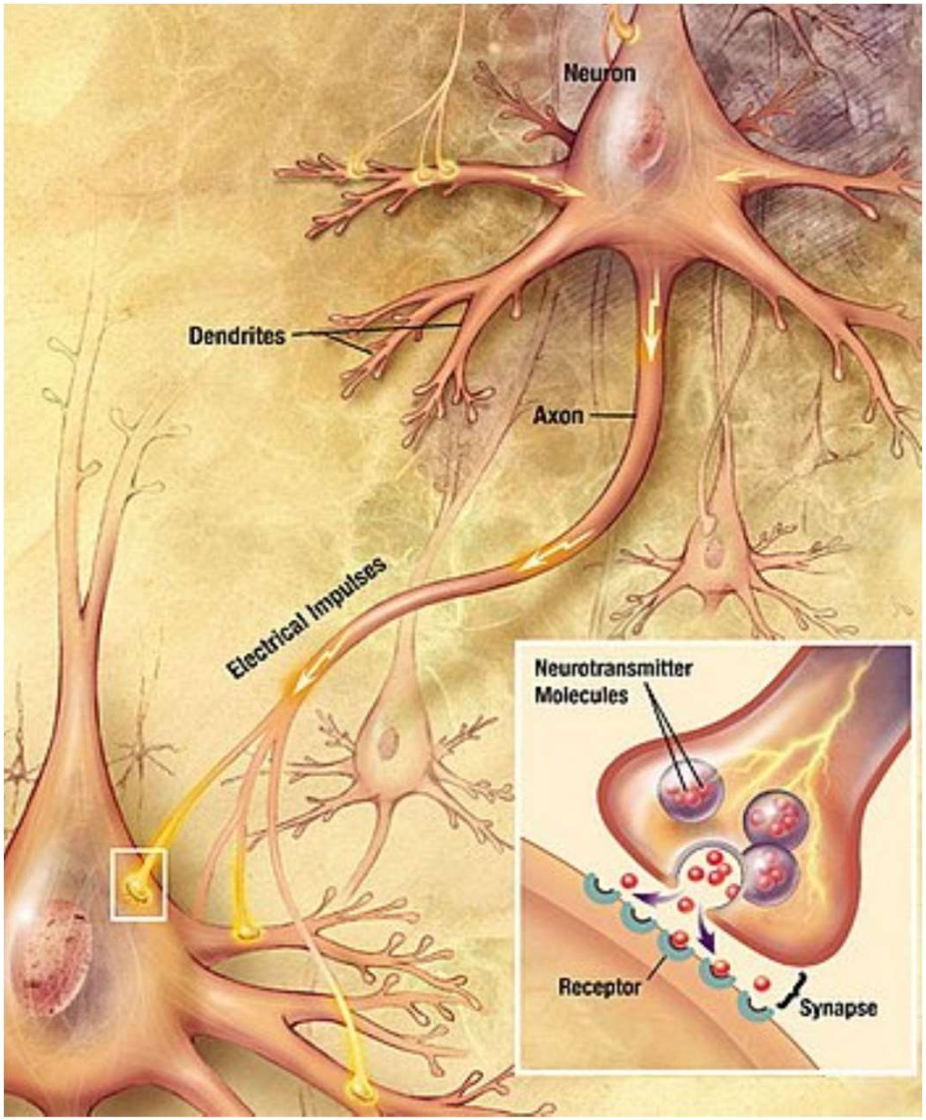


Challenging Parameter Space



Classification with Perceptrons

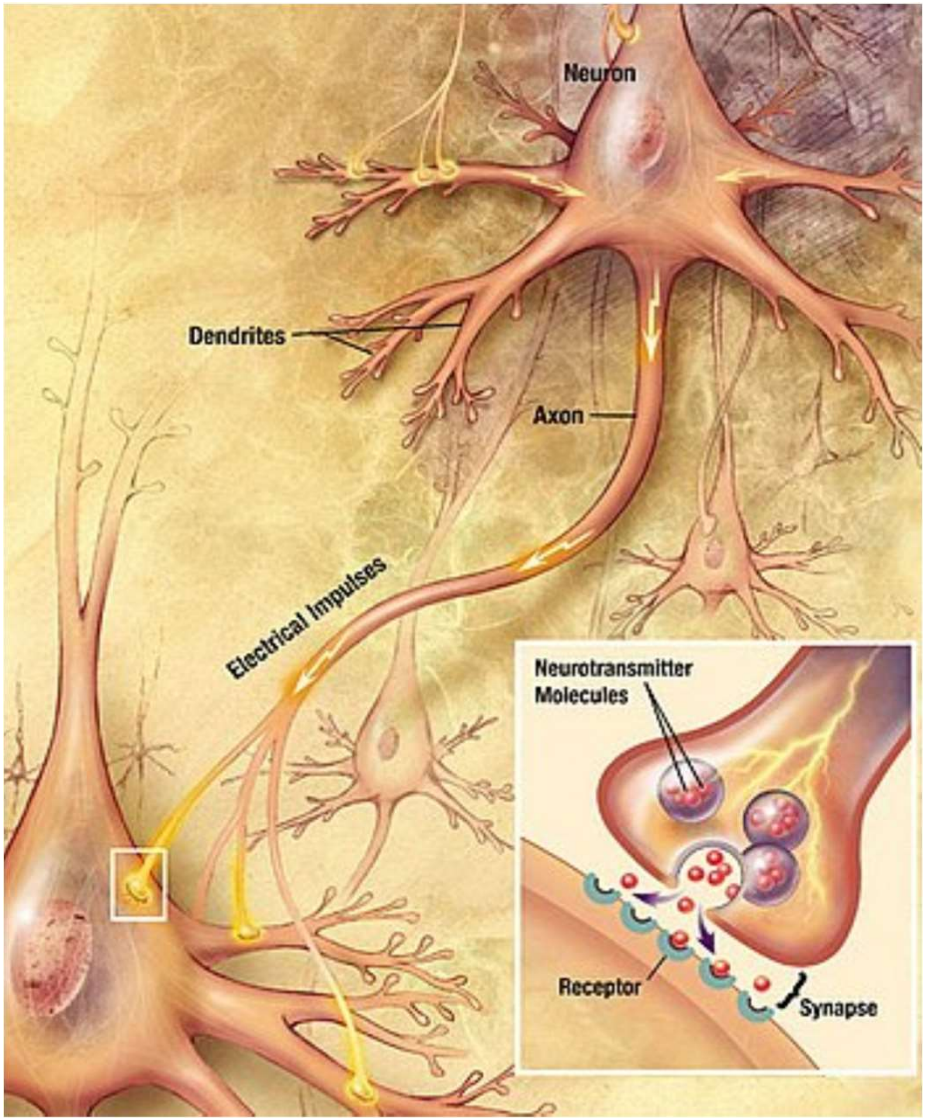
A Biological Neuron



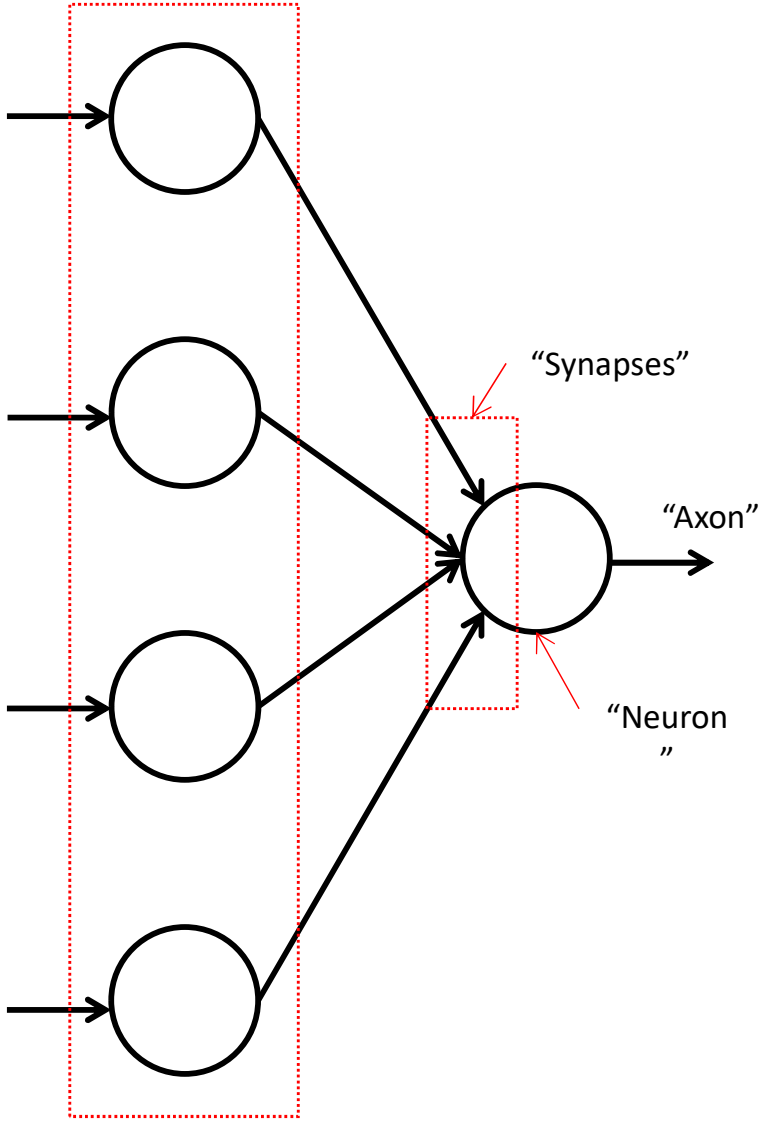
Source: https://en.wikipedia.org/wiki/Neuron

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



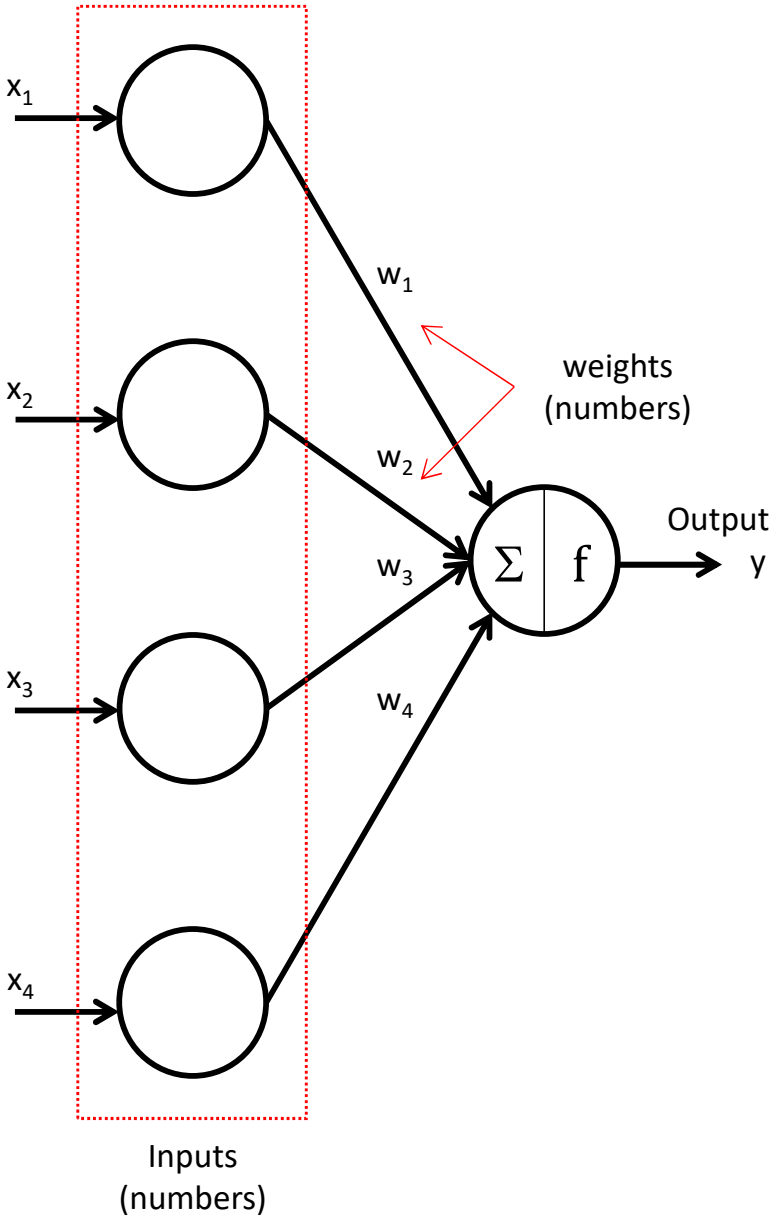
Source: https://en.wikipedia.org/wiki/Neuron



Artificial Neuron (Perceptron)

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

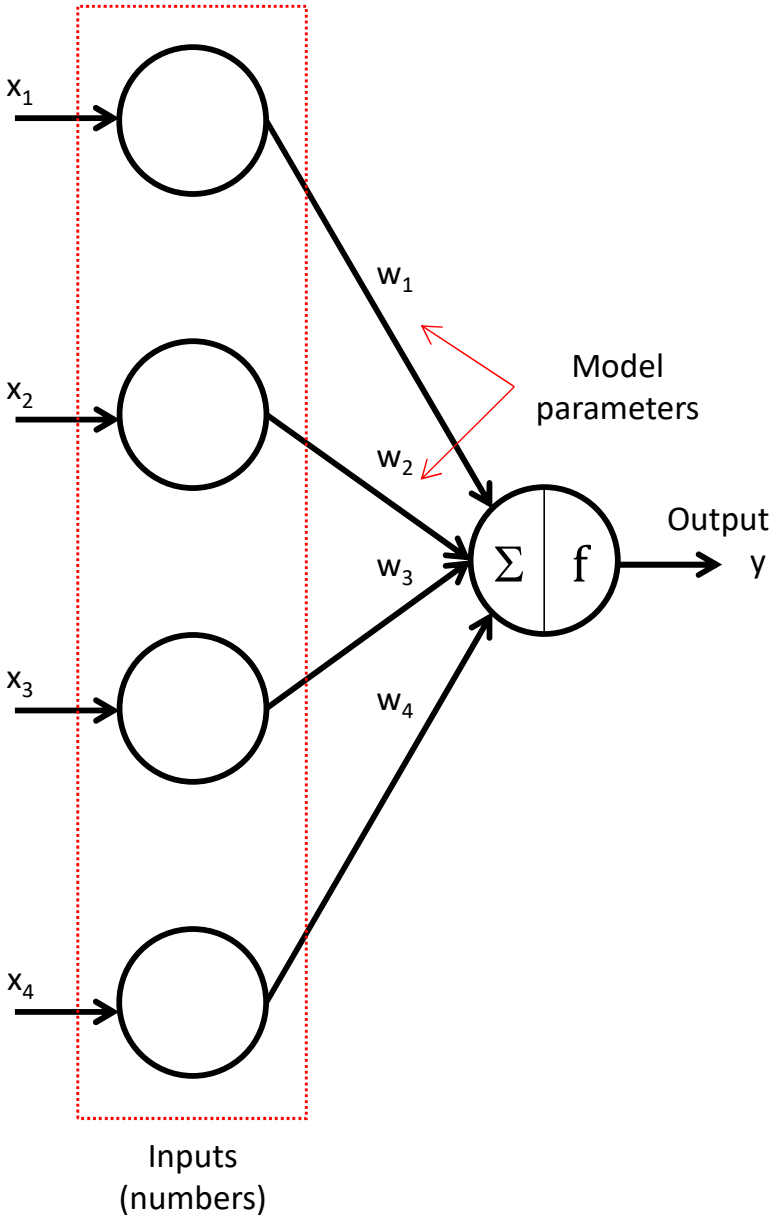
- inputs x_i - numerical values representing information
- weights w_i - numerical values representing how "important" corresponding input is
- weighted sum: $\Sigma\ w_i * x_i$
- activation function f that decides if the neuron "fires"



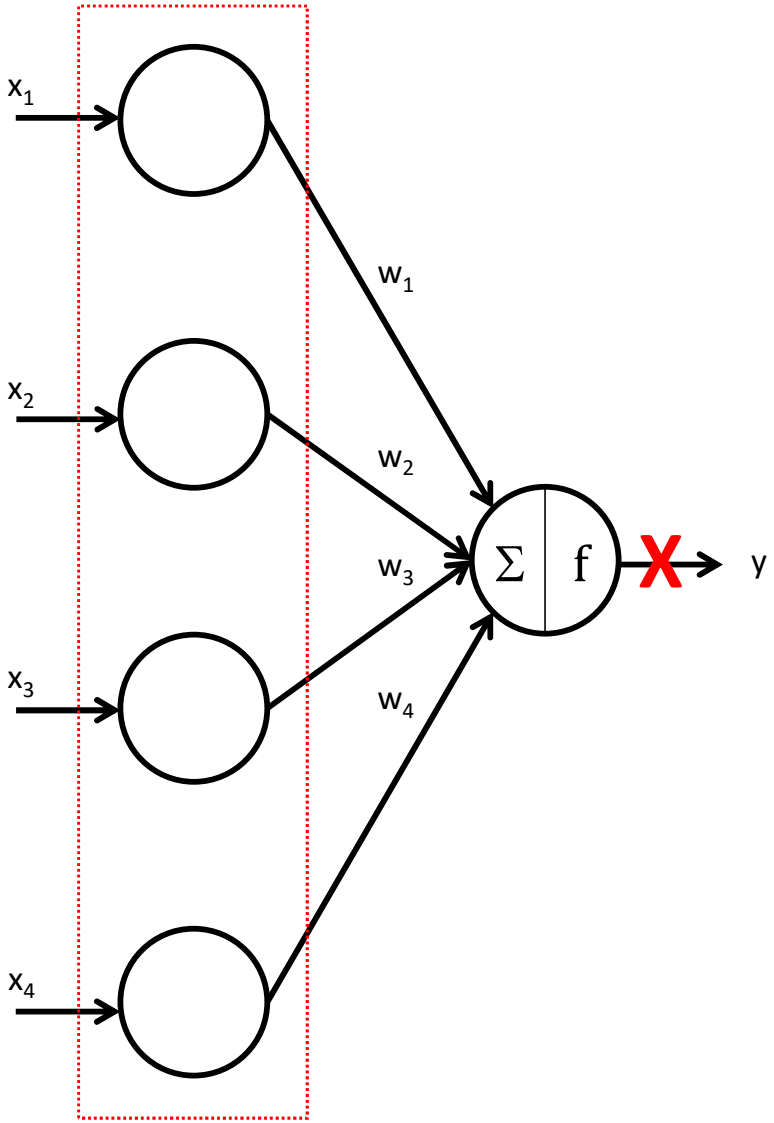
Artificial Neuron (Perceptron)

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

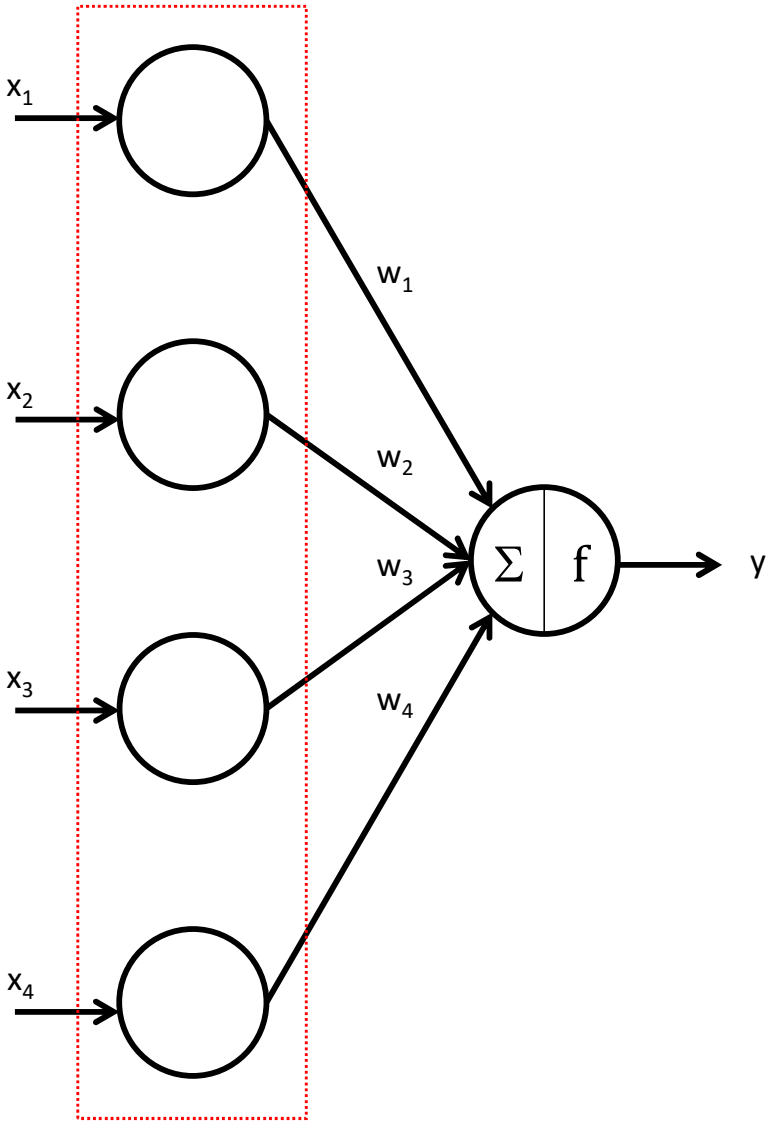
- inputs x_i - numerical values representing information
- weights w_i - numerical values representing how "important" corresponding input is
- weighted sum: $\Sigma\ w_i * x_i$
- activation function f that decides if the neuron "fires"



Artificial Neuron (Perceptron)

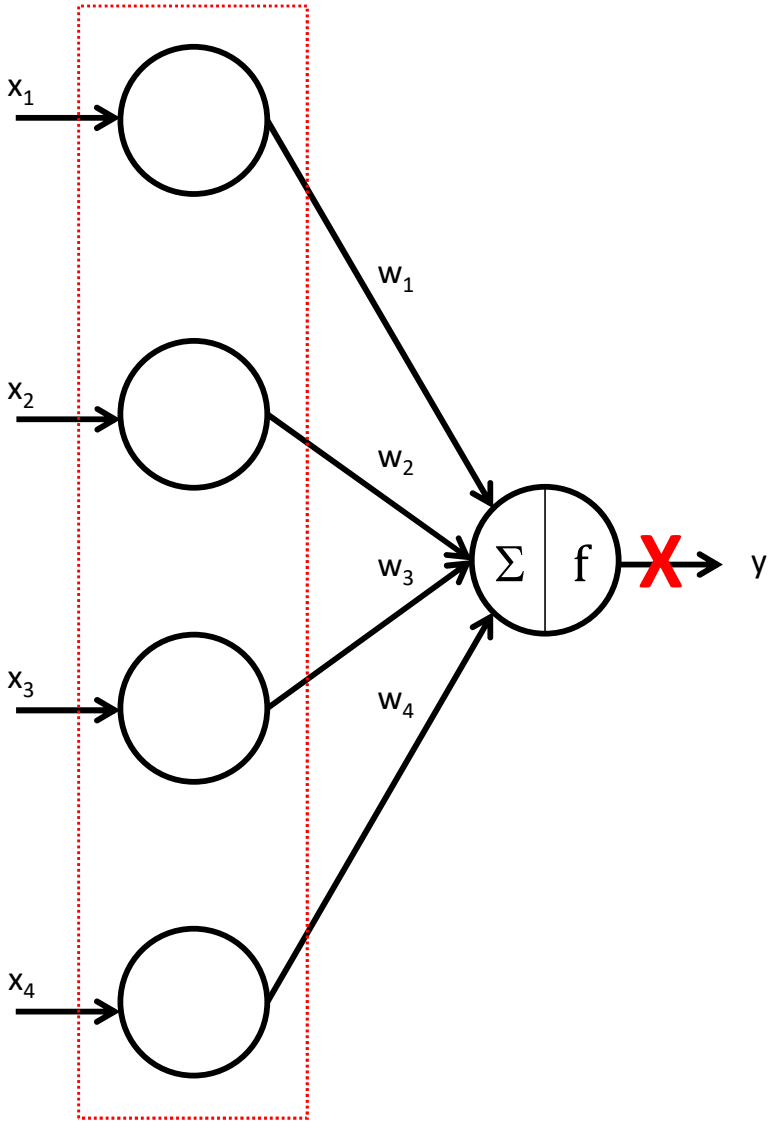


$\Sigma\ w_i * x_i < 0 \rightarrow f = 0$ → **DON'T "fire"**

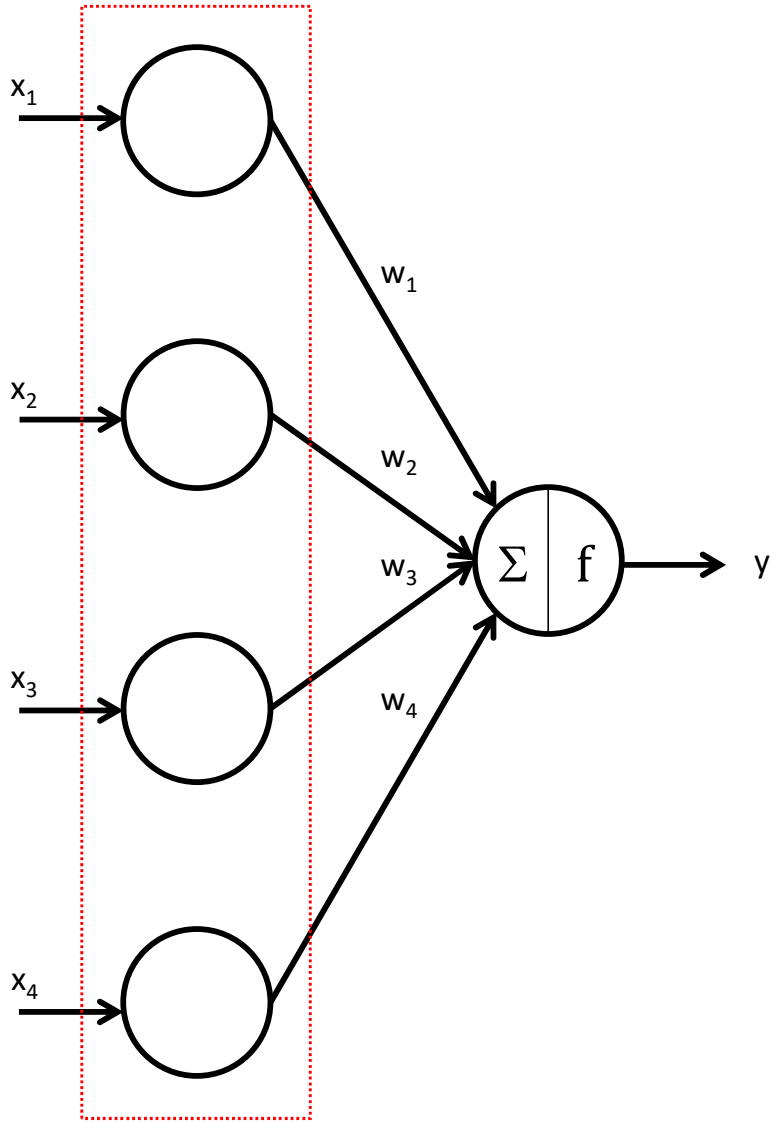


$\Sigma\ w_i * x_i \geq 0 \rightarrow f = 1$ → **"fire"**

Single-layer Perceptron as a Classifier

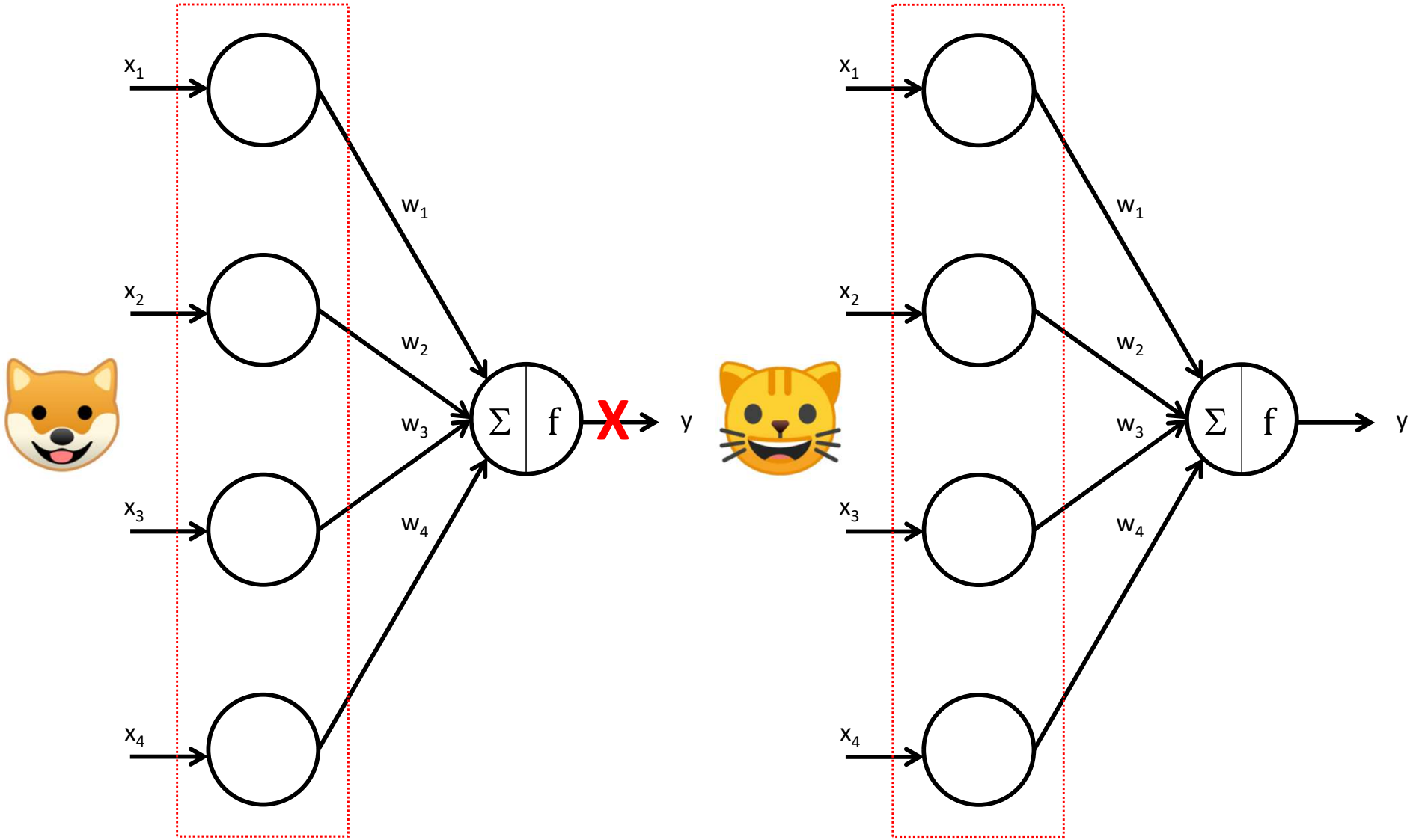


$\Sigma\, w_i * x_i < 0 \rightarrow f = 0 \rightarrow$ **NO**



$\Sigma\, w_i * x_i \geq 0 \rightarrow f = 1 \rightarrow$ **YES**

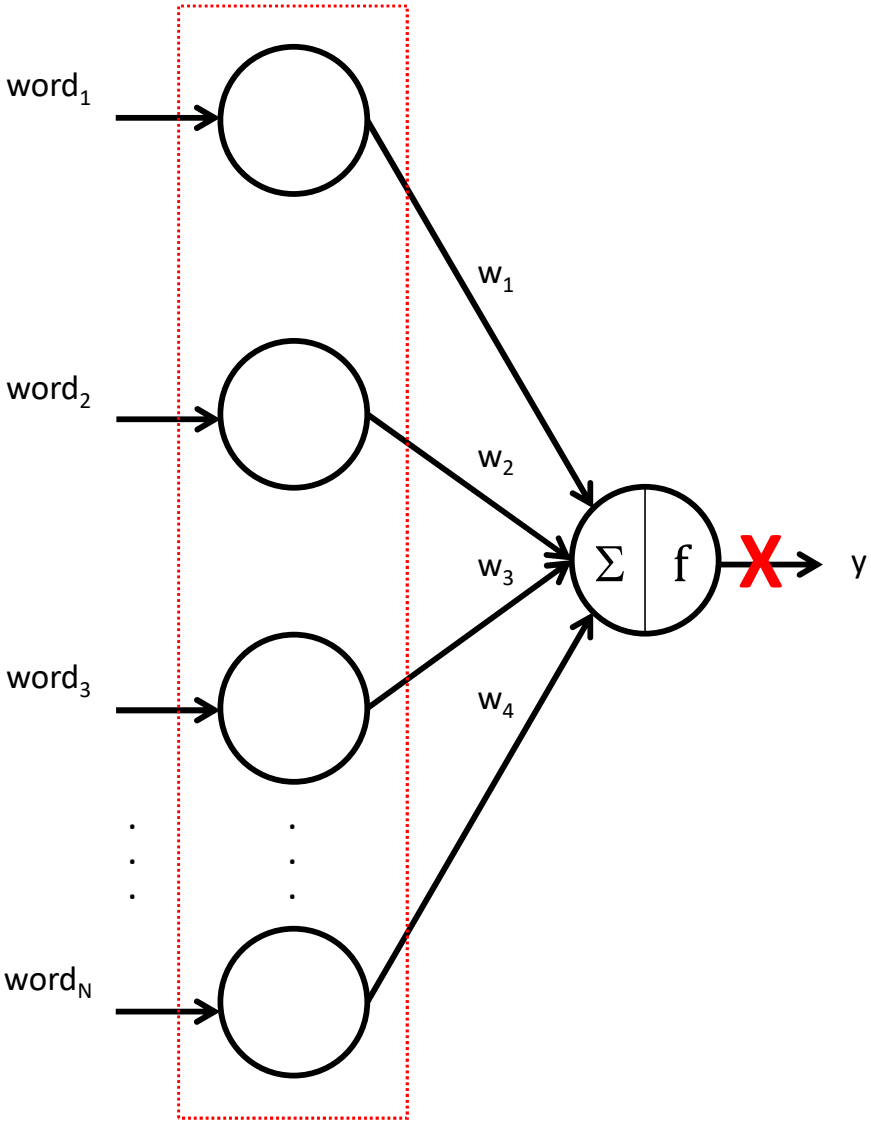
Single-layer Perceptron as a Classifier



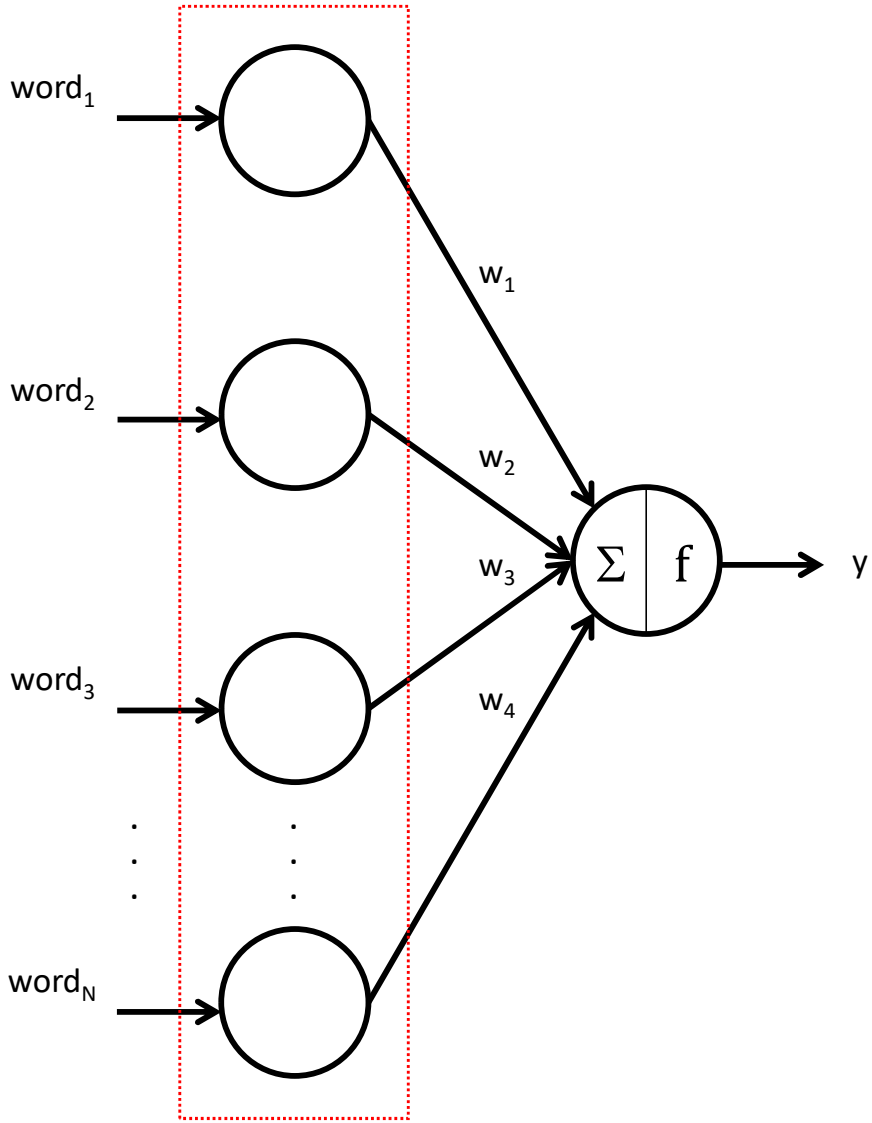
$\Sigma\, w_i * x_i < 0 \rightarrow f = 0$ → **NOT CAT**

$\Sigma\, w_i * x_i \geq 0 \rightarrow f = 1$ → **CAT**

Single-layer Perceptron as a Classifier

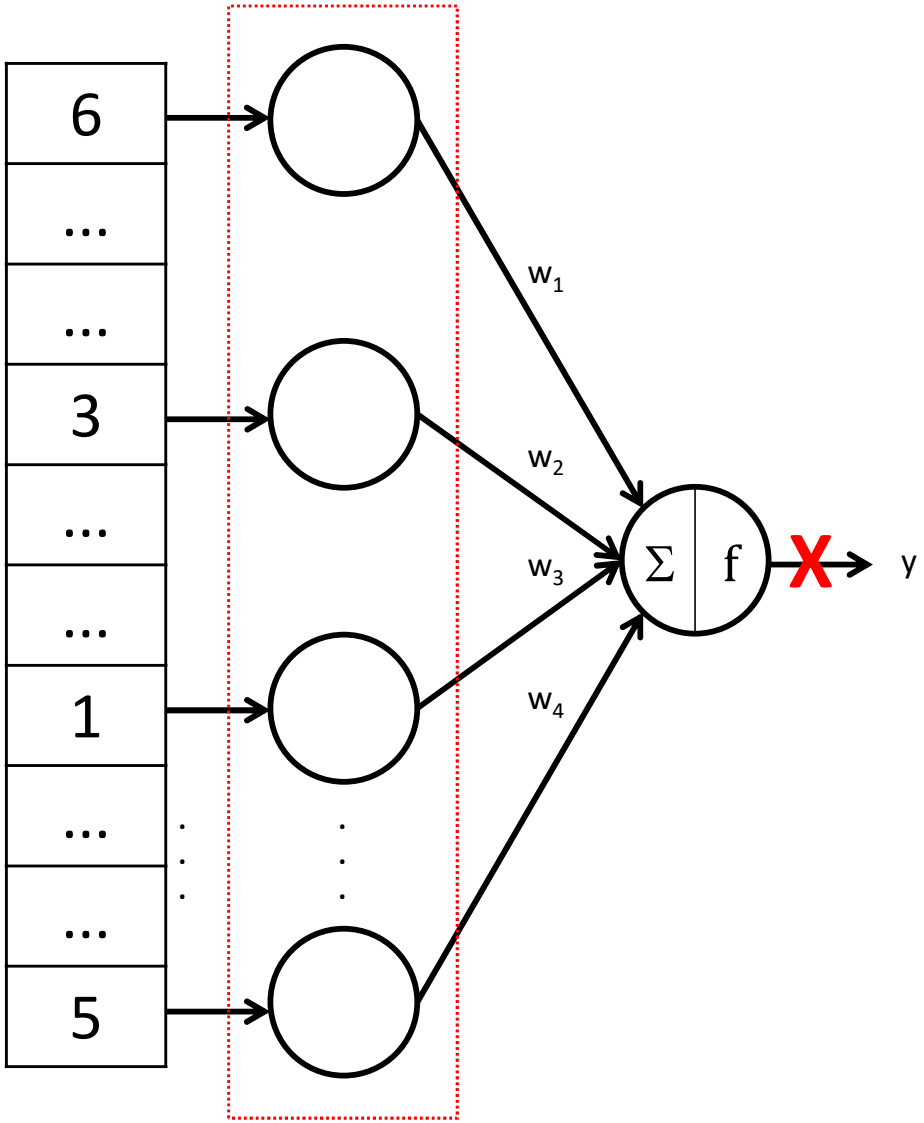


$\Sigma\, w_i * word_i < 0 \rightarrow f = 0 \rightarrow$ **SPAM**

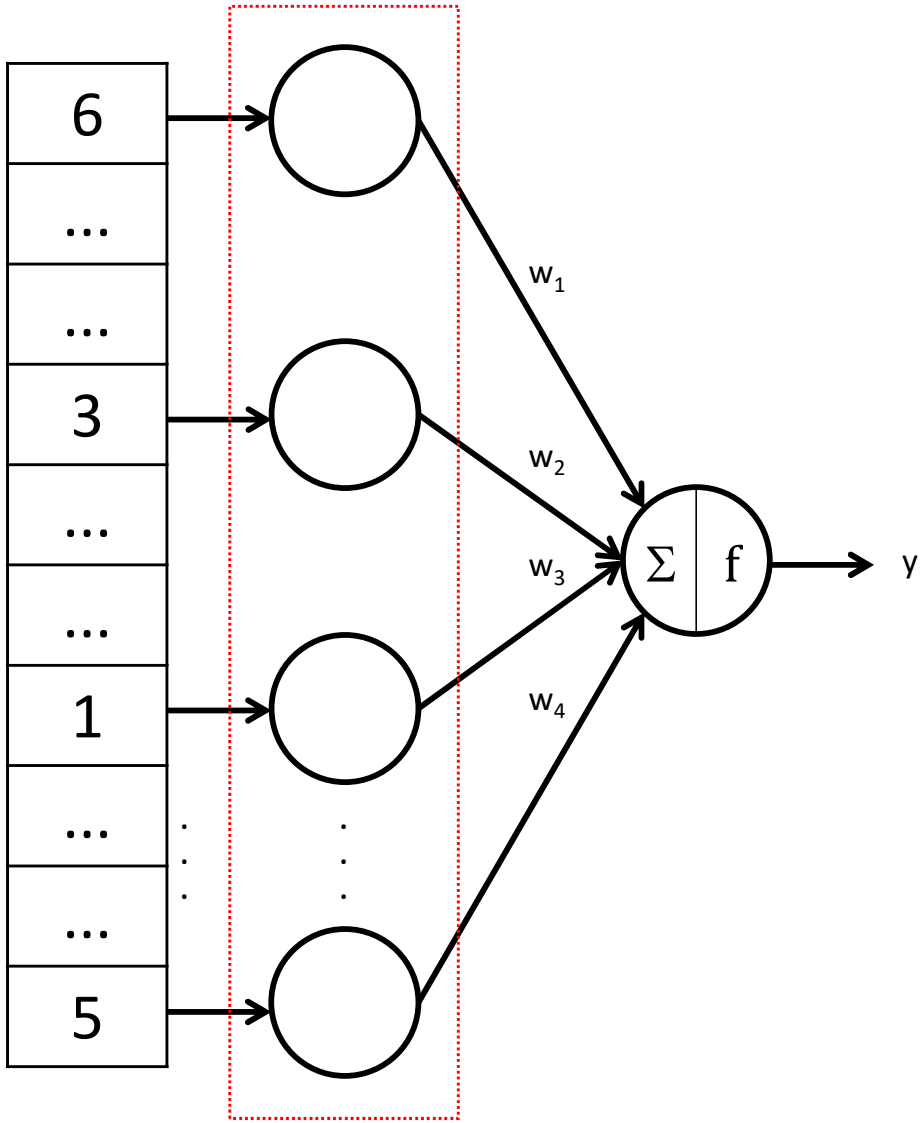


$\Sigma\, w_i * word_i \geq 0 \rightarrow f = 1 \rightarrow$ **HAM**

Single-layer Perceptron as a Classifier



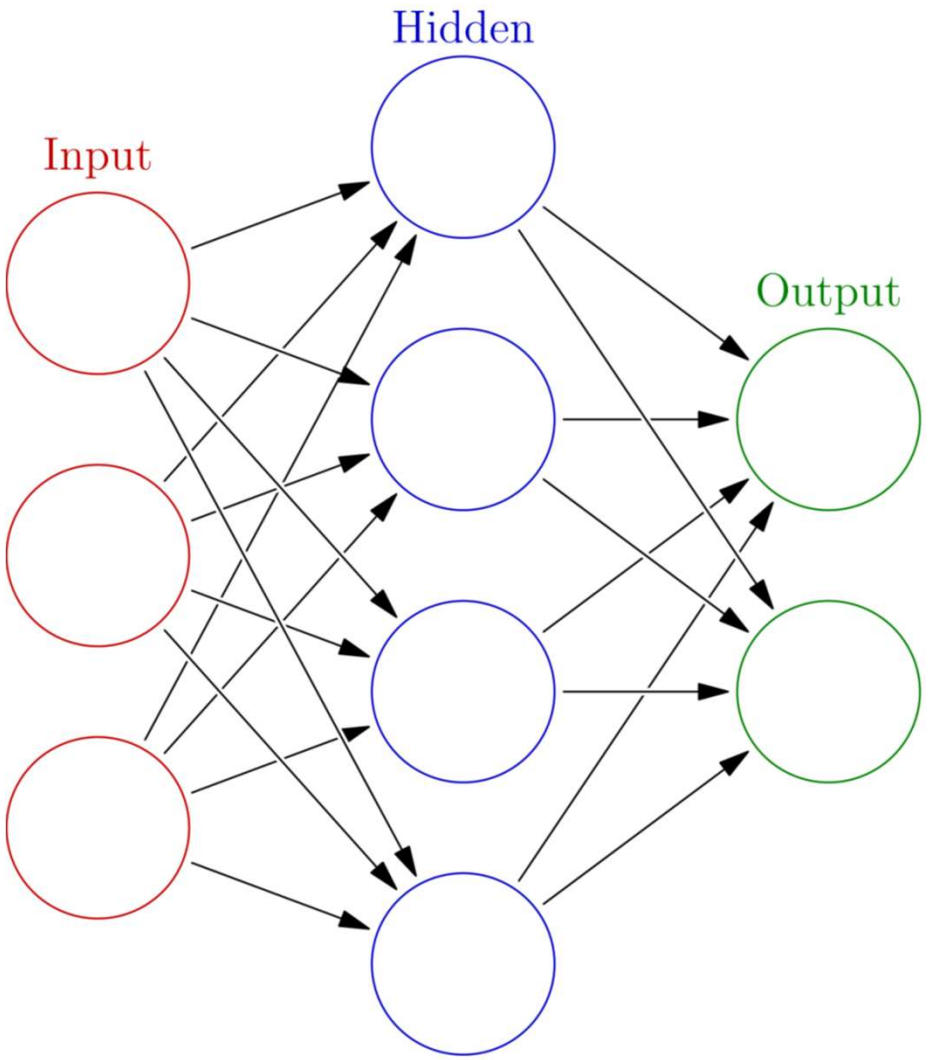
$\Sigma\ w_i * word_i < 0 \rightarrow f = 0 \rightarrow$ **SPAM**



$\Sigma\ w_i * word_i \geq 0 \rightarrow f = 1 \rightarrow$ **HAM**

Classification with Artificial Neural Networks

Artificial Neural Network (1943)

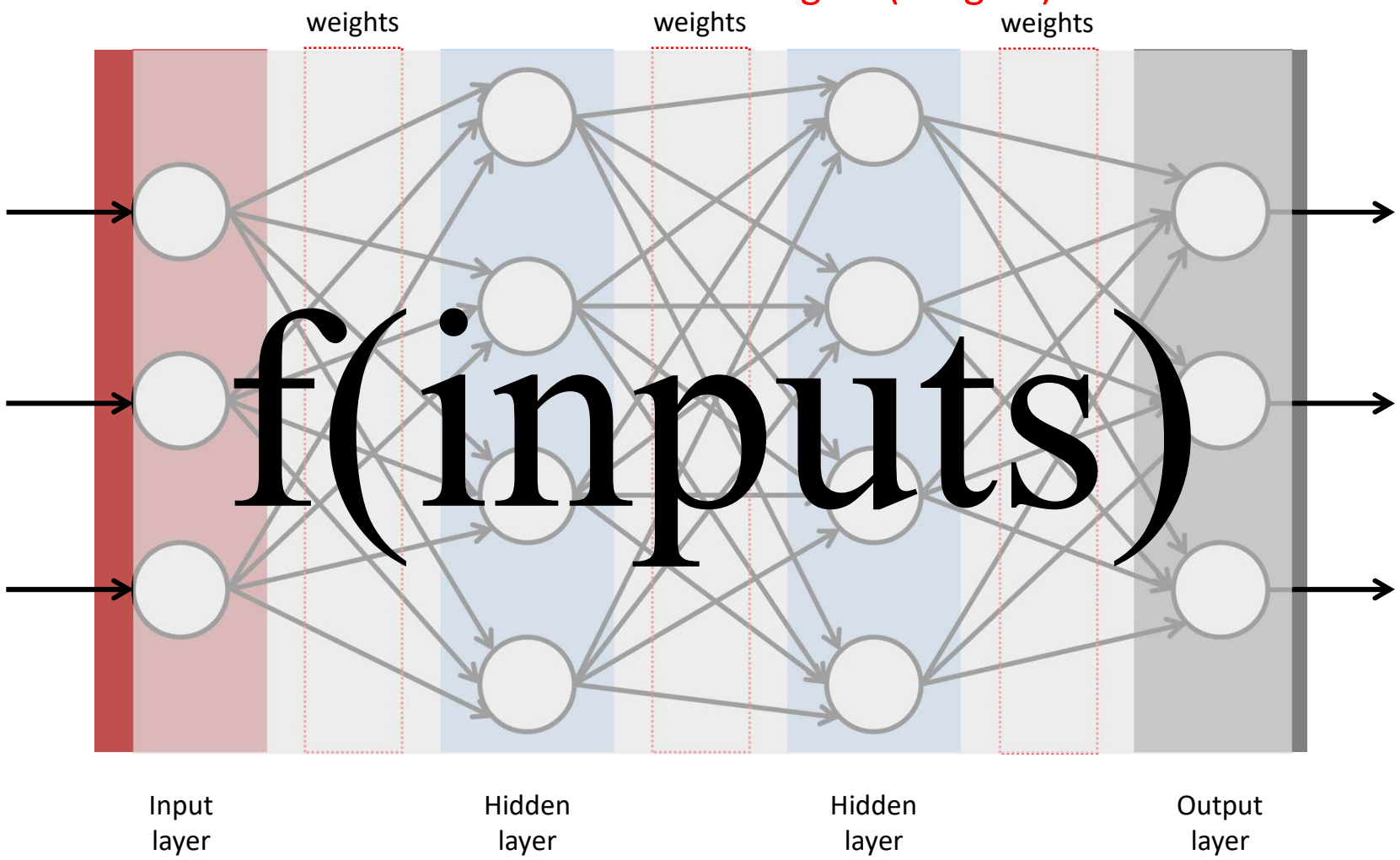


Source: https://en.wikipedia.org/wiki/Artificial_neural_network

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**.

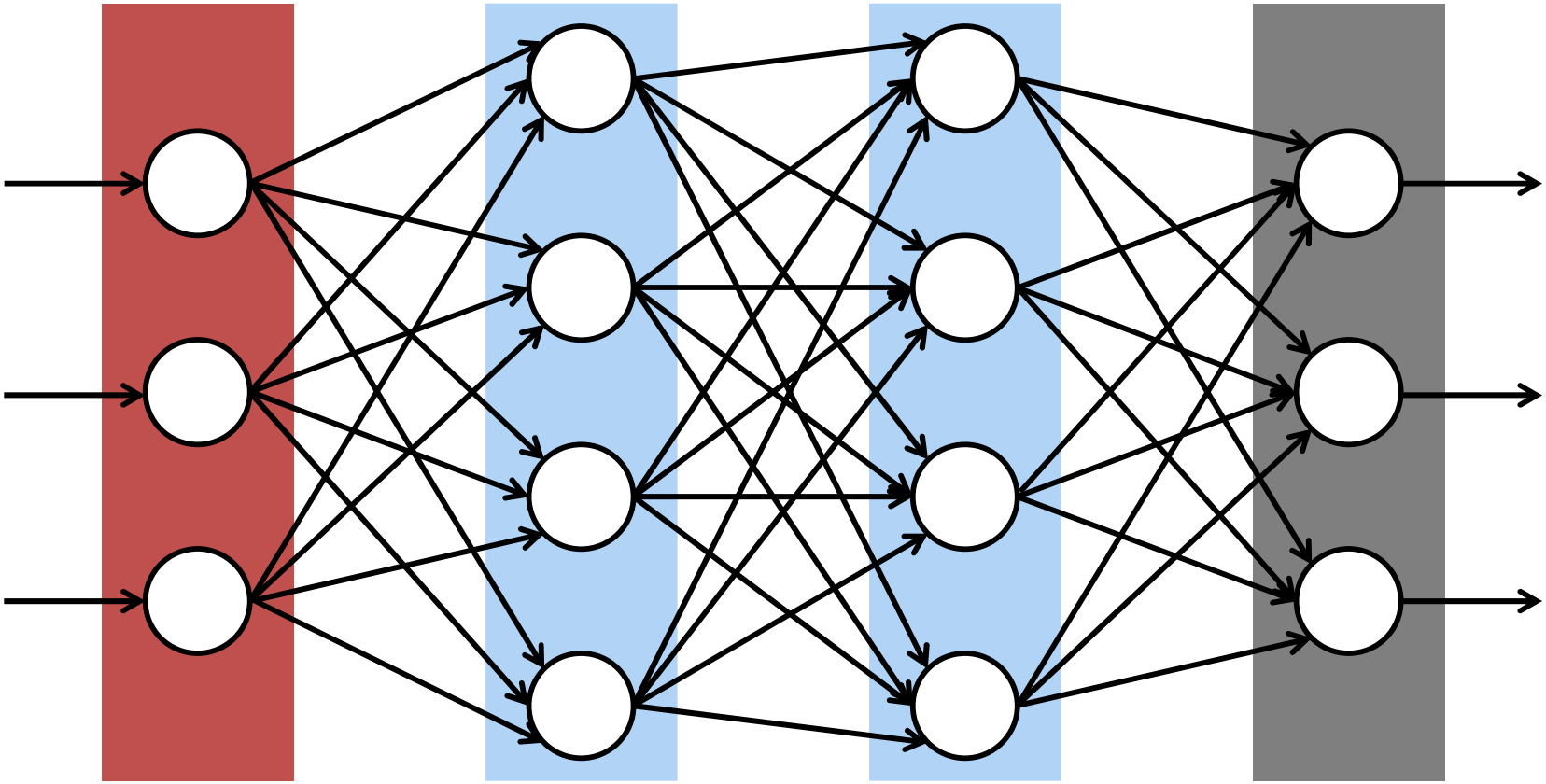
ANN as a Complex Function

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



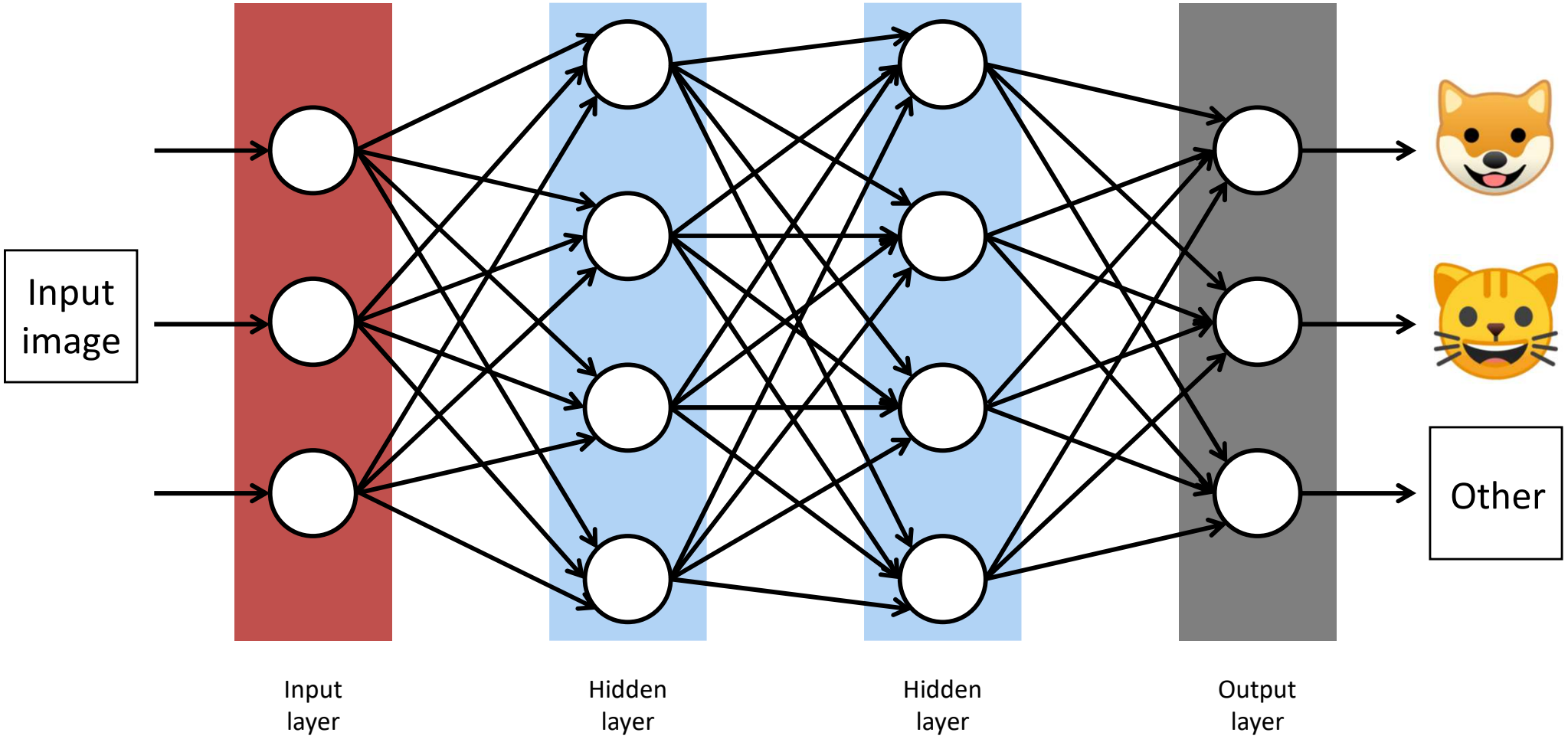
Artificial Neural Network (ANN)

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

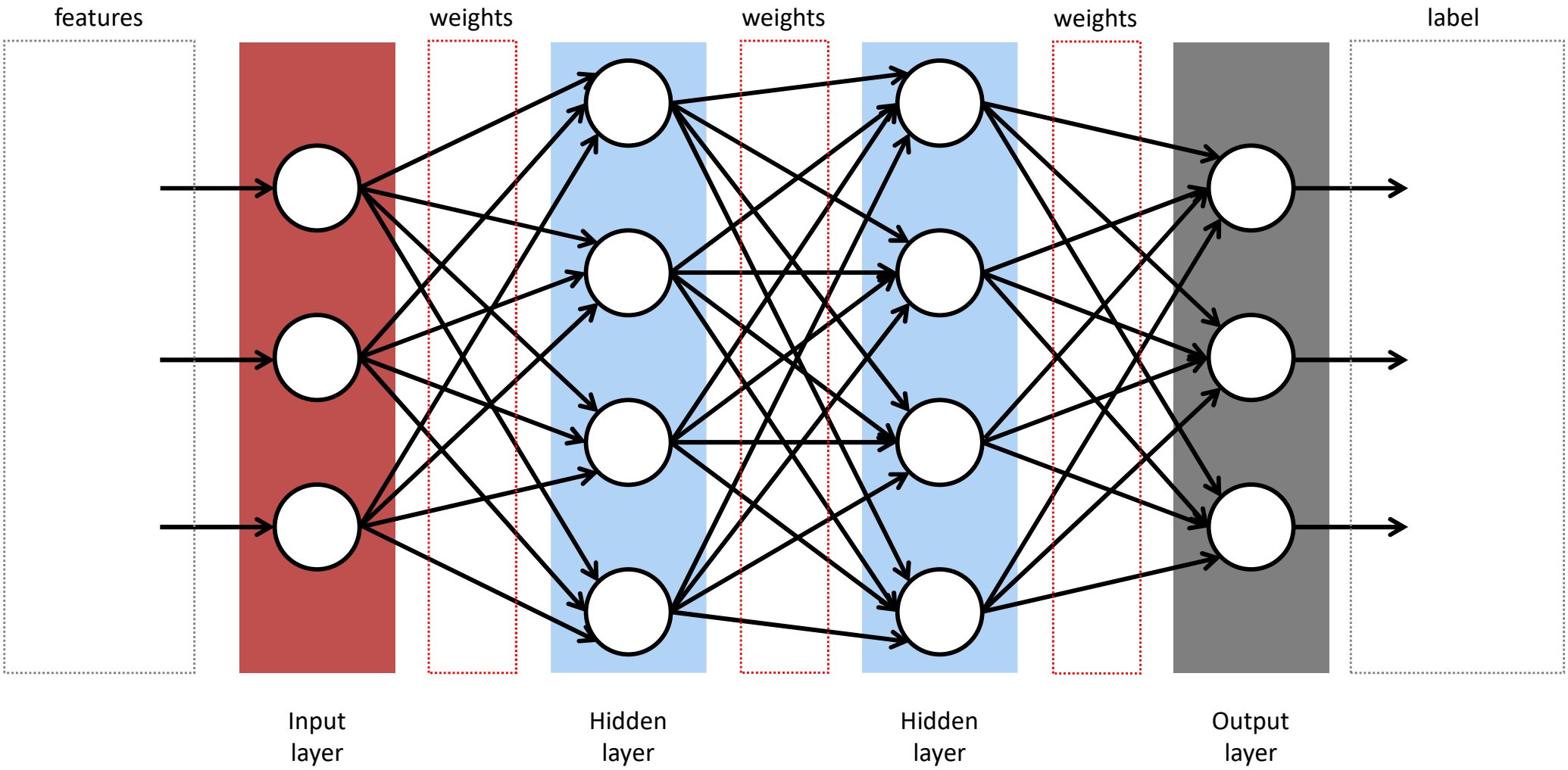


ANN as an Image Classifier

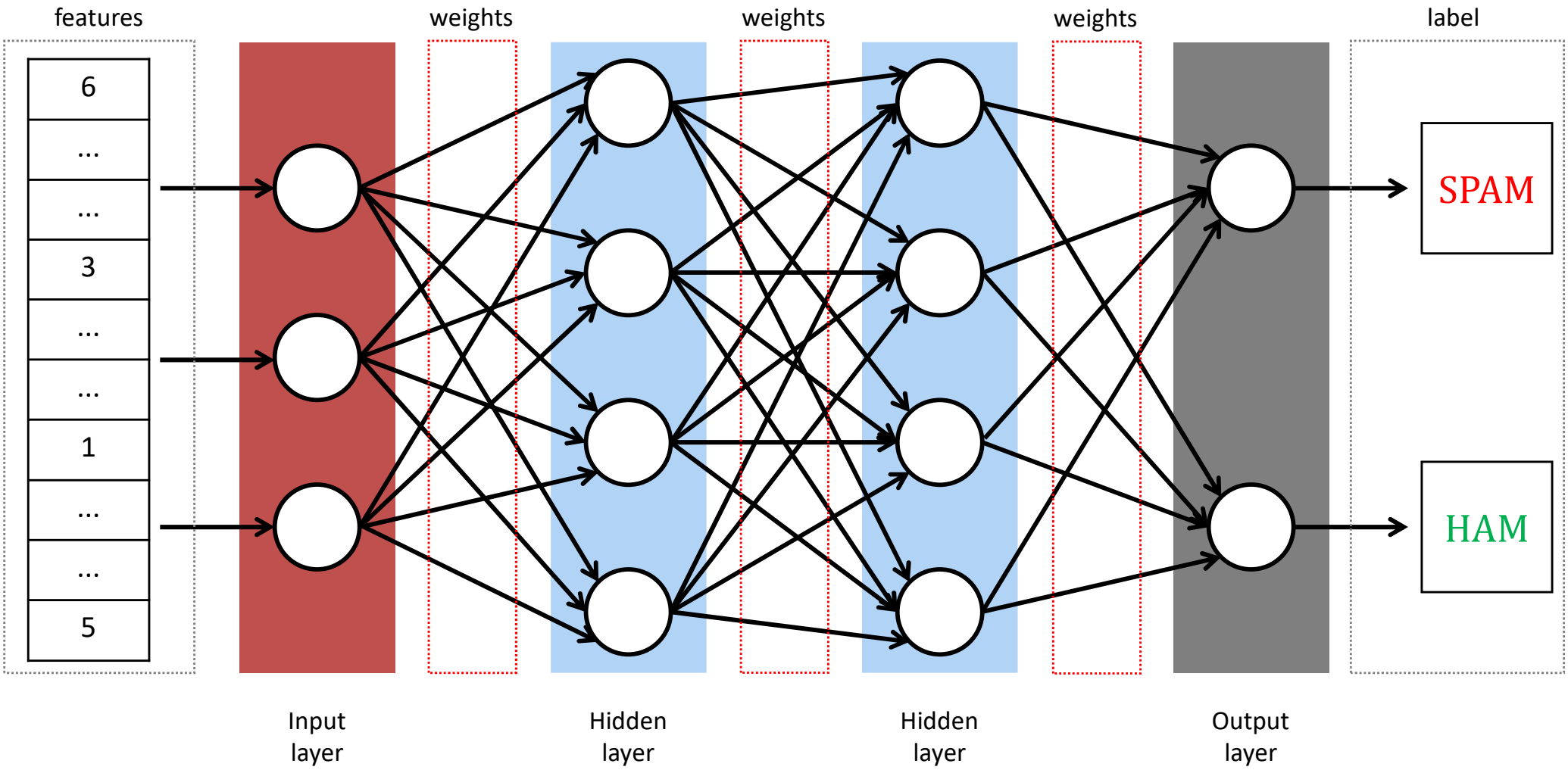
An artificial neural network can be used as a **classifier** as well.



ANN as a General Classifier

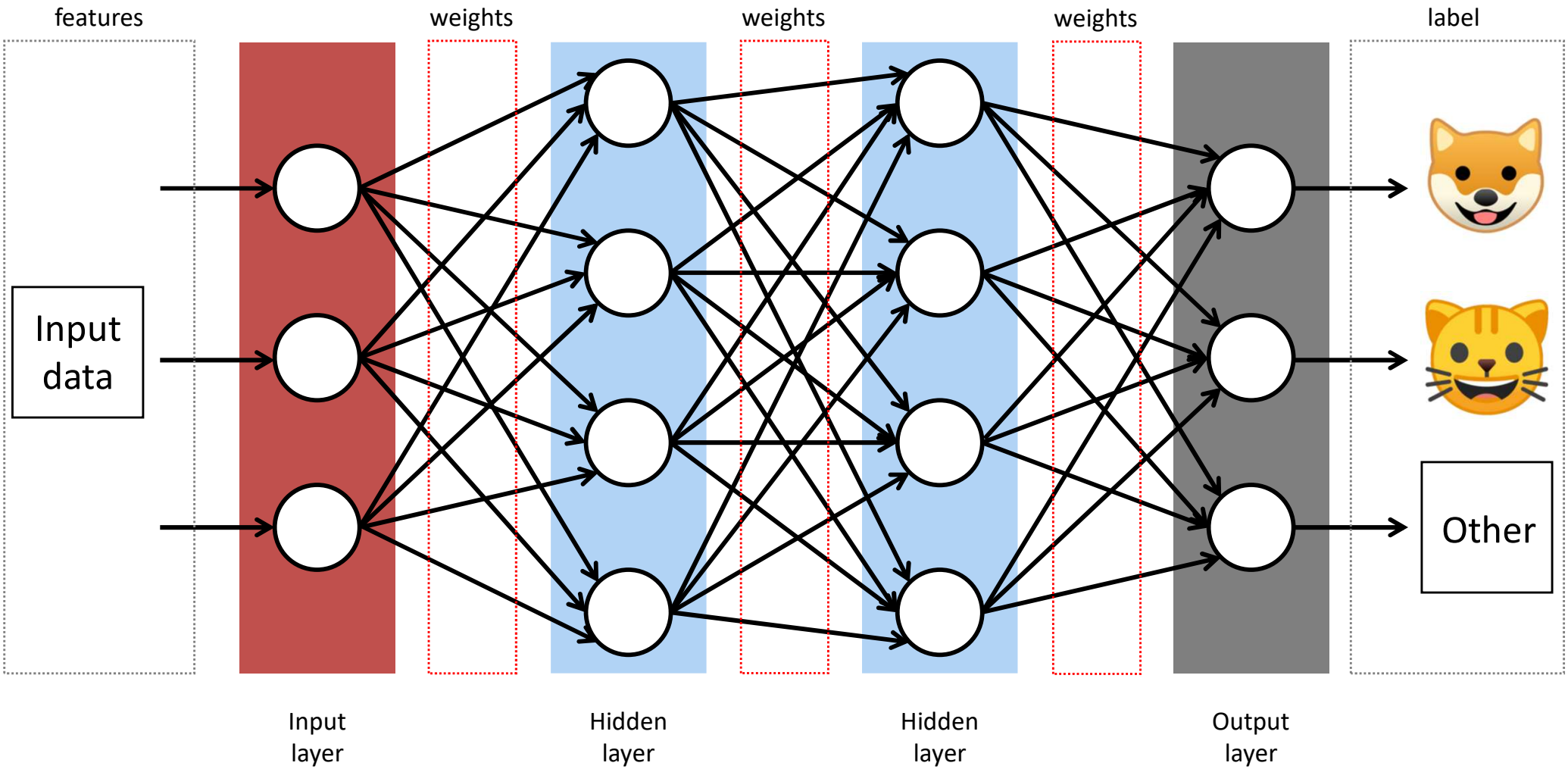


ANN as a Text Classifier



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**.

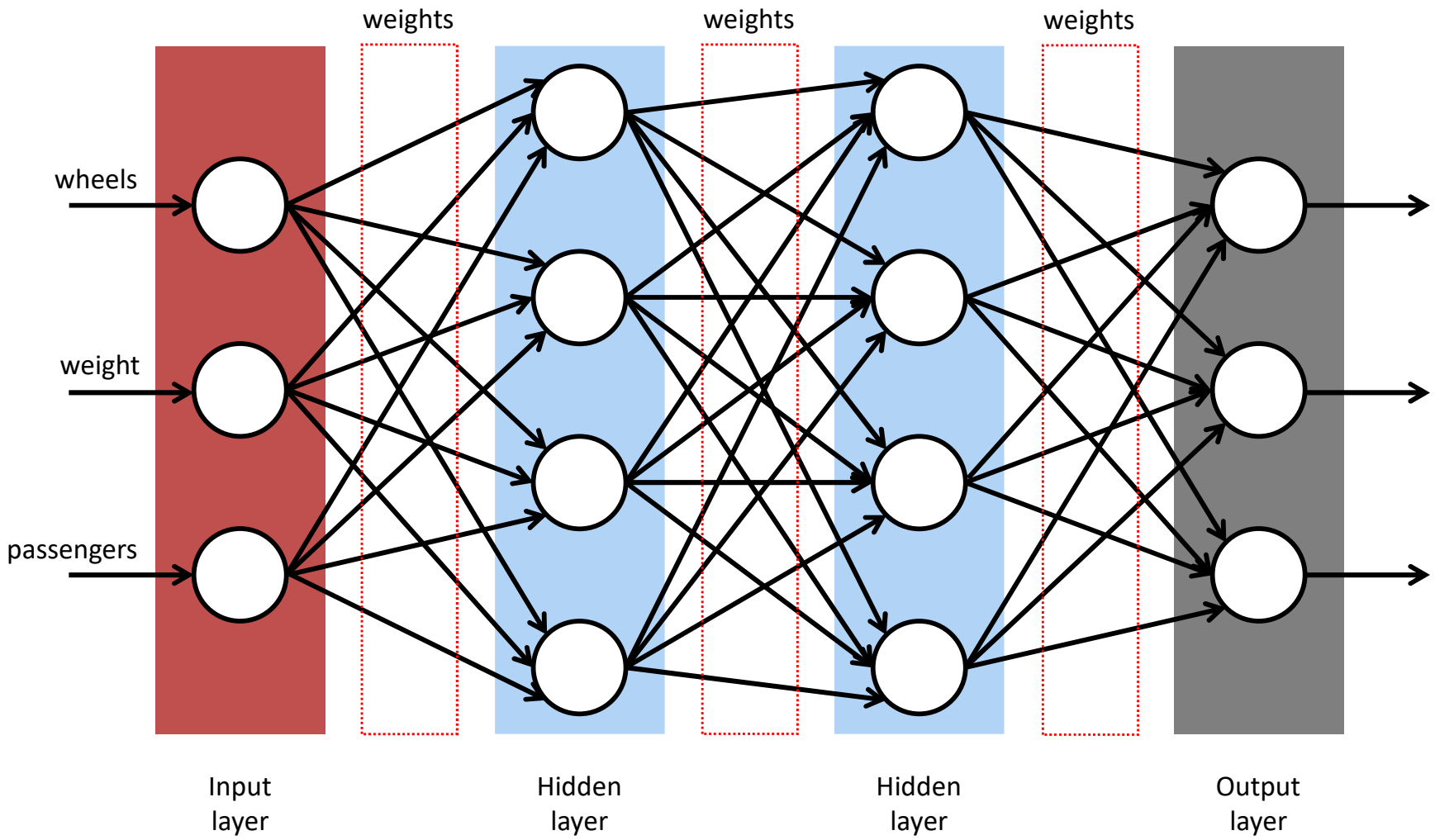
Features:	Label:
Wheels: 4 Weight: 8 tons Passengers: 1	Truck

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

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

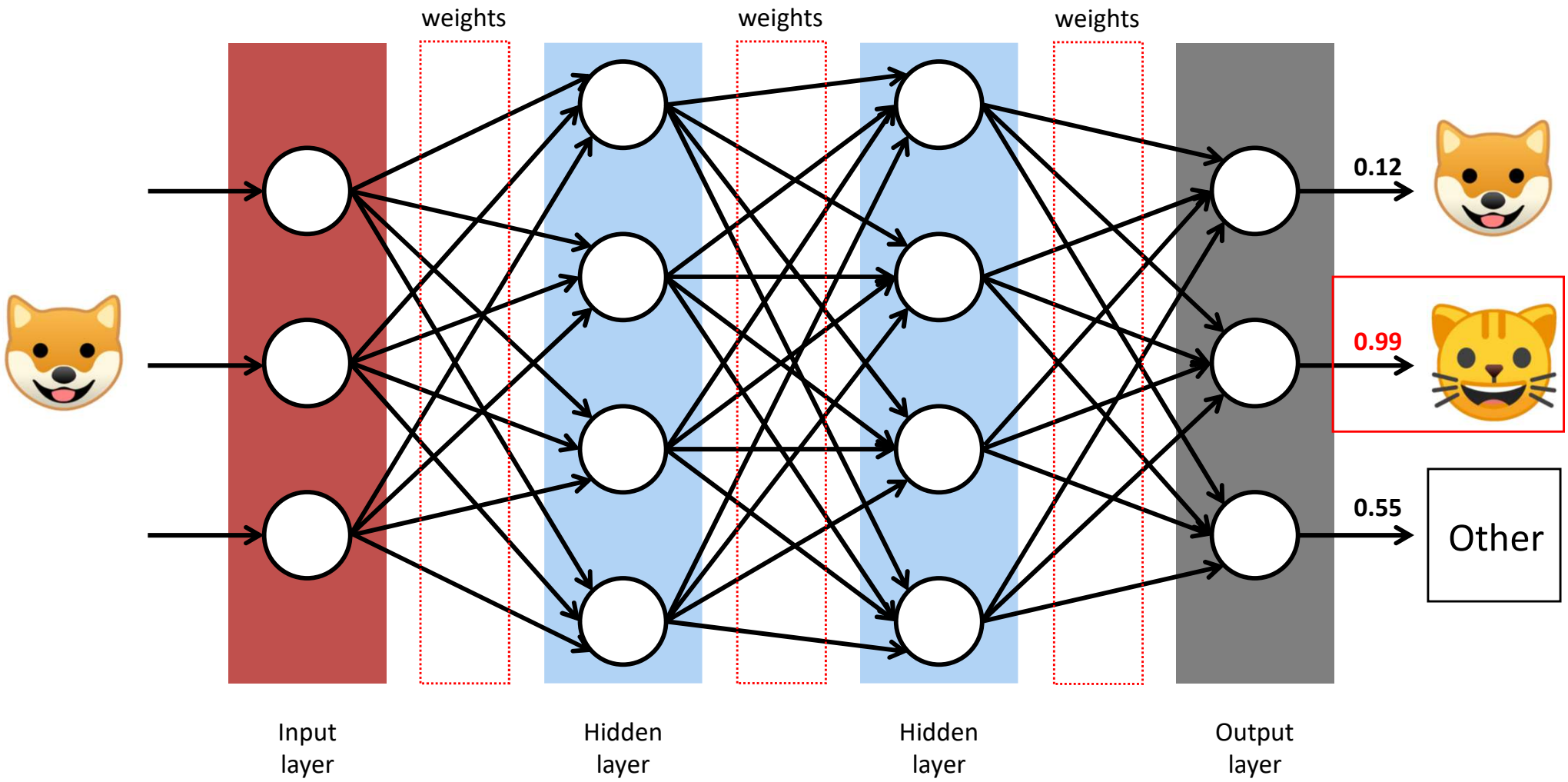
Features:	Label:
Wheels: 4 Weight: 2 tons Passengers: 4	Car

ANN: Supervised Learning



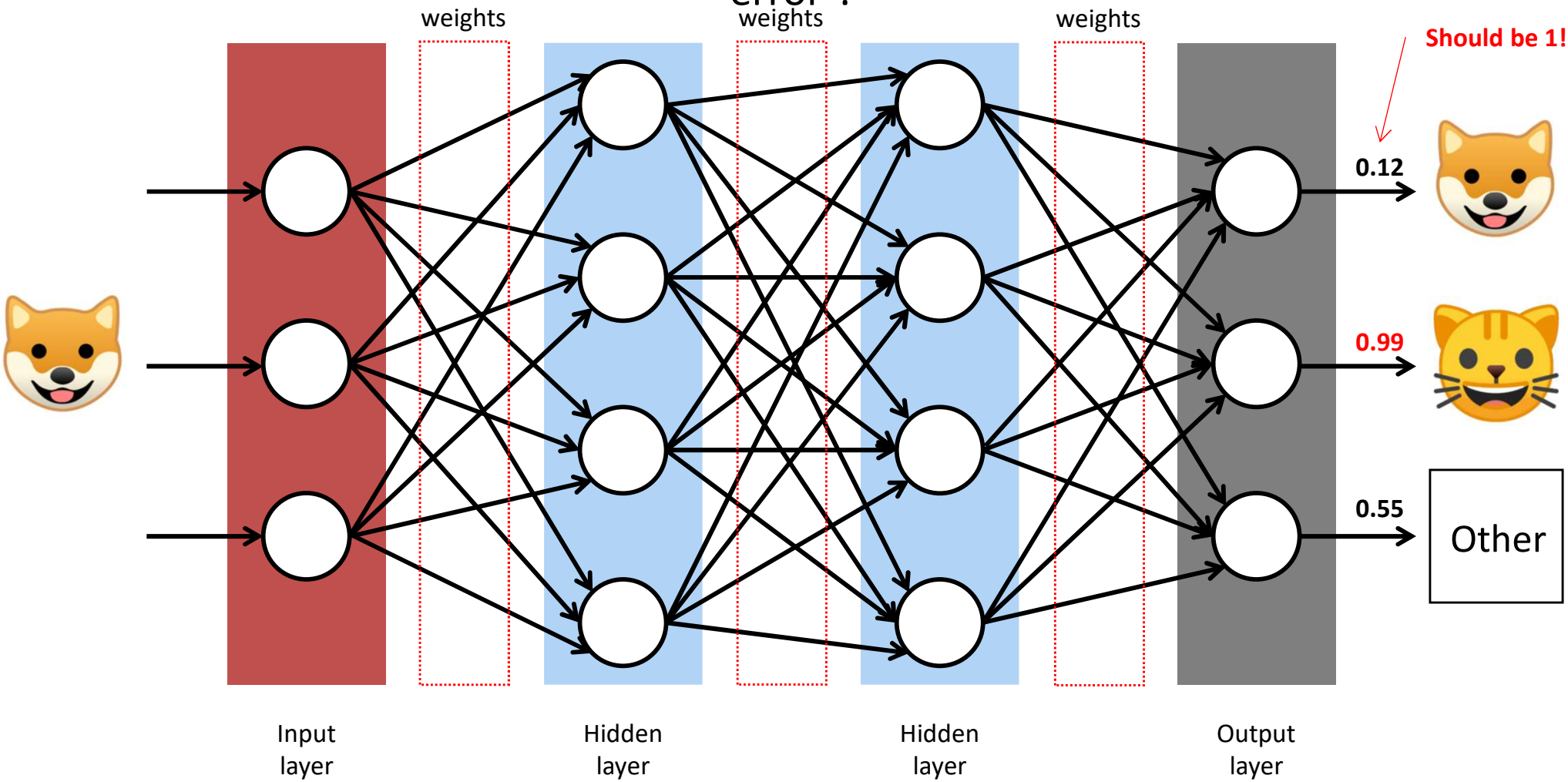
ANN: Supervised Learning

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



ANN: Training

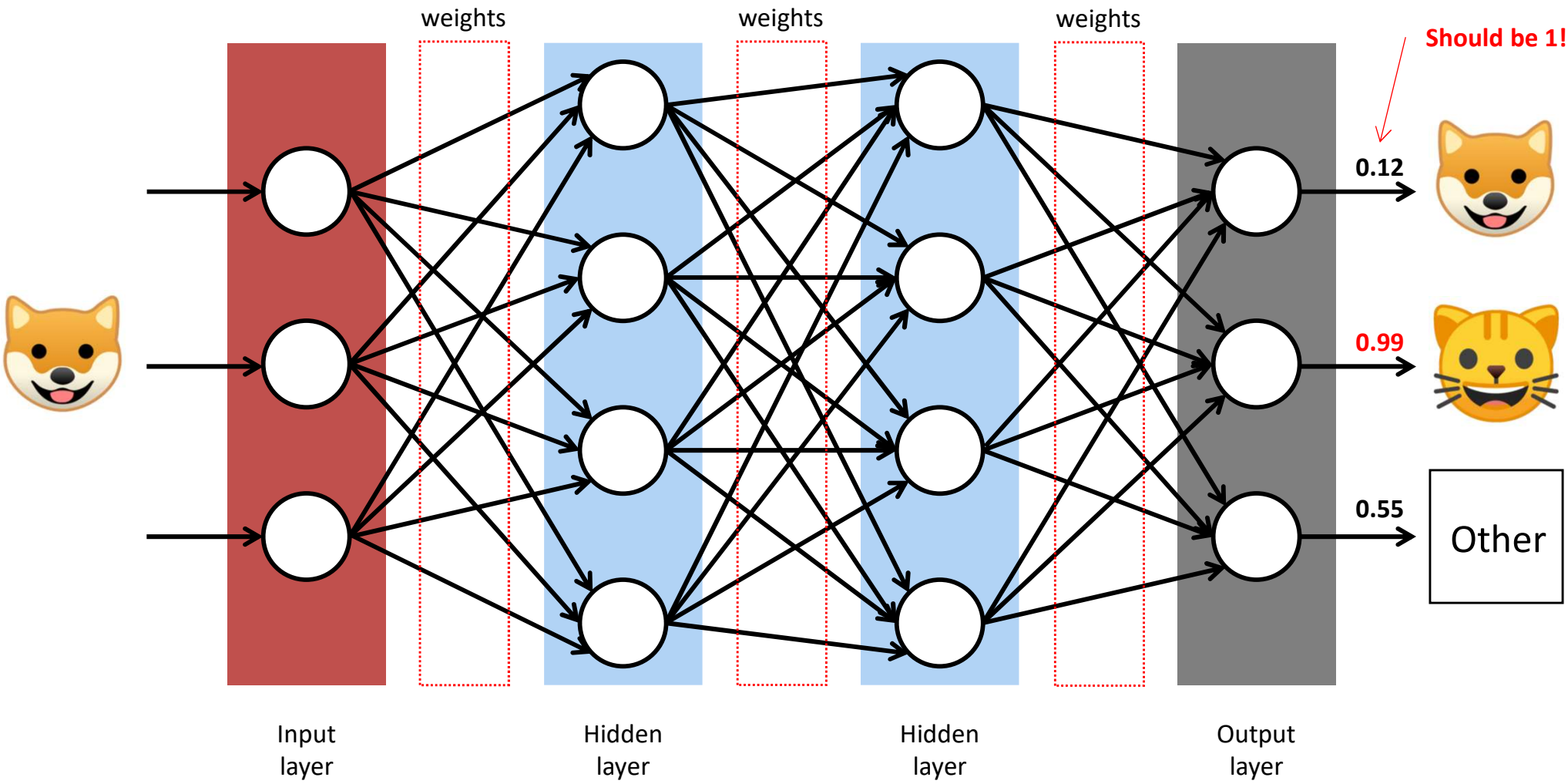
Given: input d  and it's corresponding **expected** label: DOG calculate "error".



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

ANN: Training

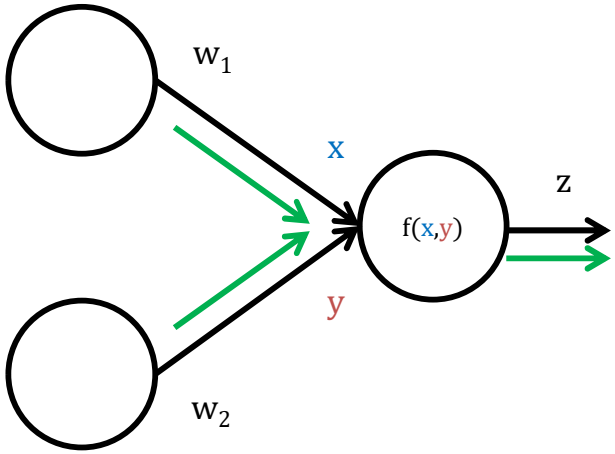
Show data / label pair:  / DOG.



Correct all the weights. Repeat many times.

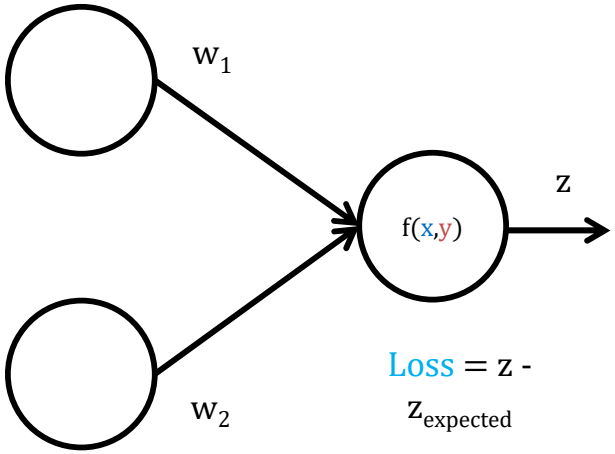
Back-propagation

Feed forward



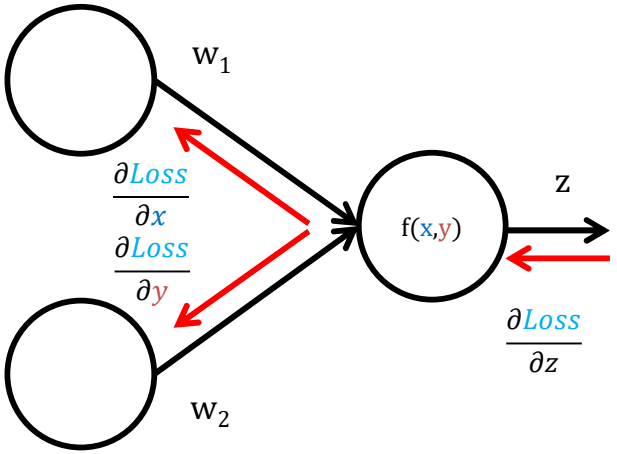
Feed a **labeled sample** through the network

Evaluate Loss



How "incorrect" is the result compare to the label?

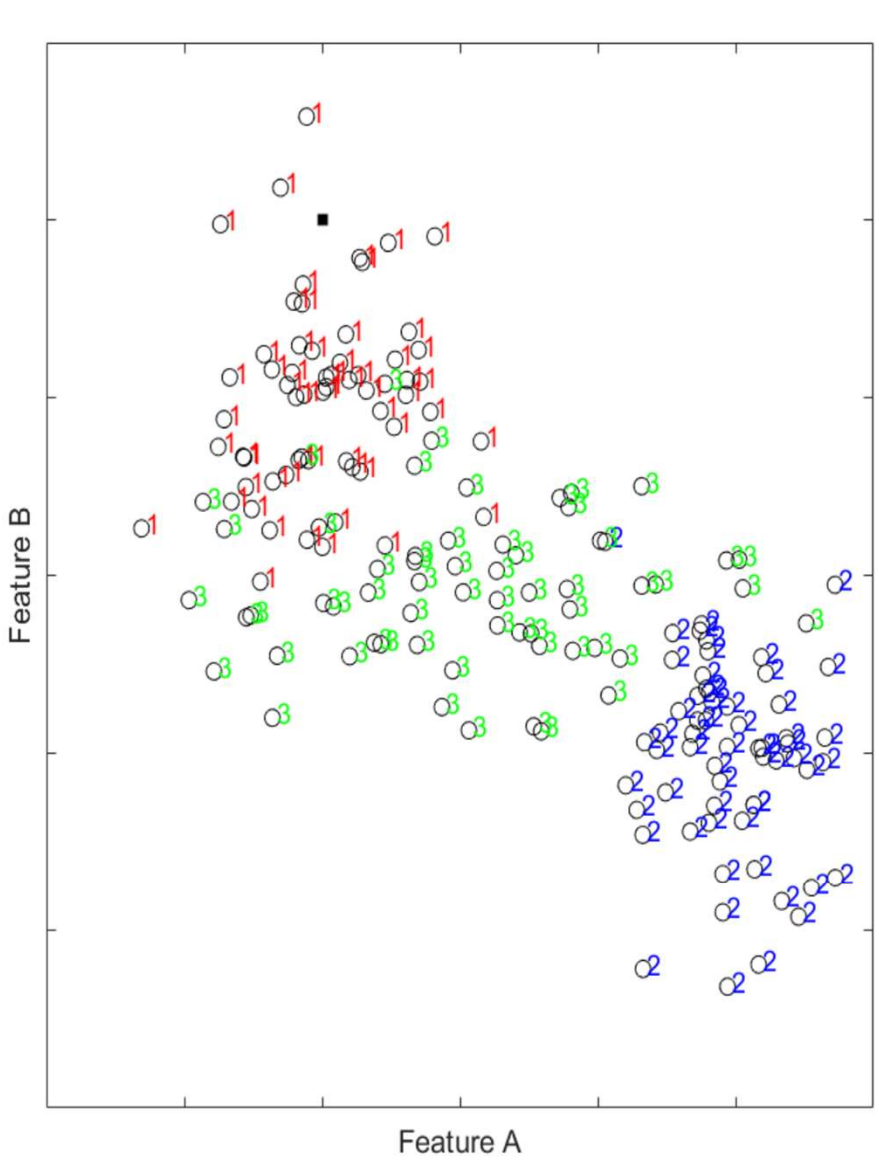
Back-propagation

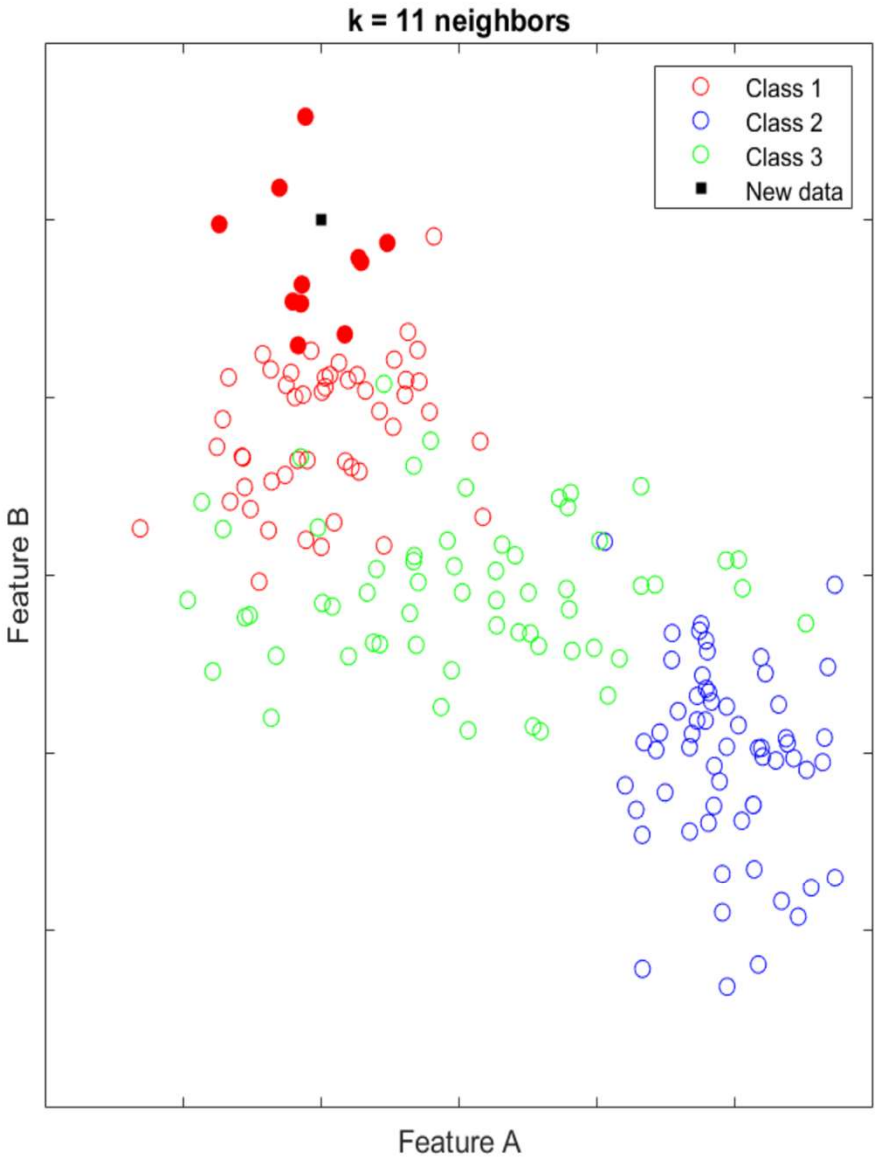


Update weights (use **Gradient Descent**)

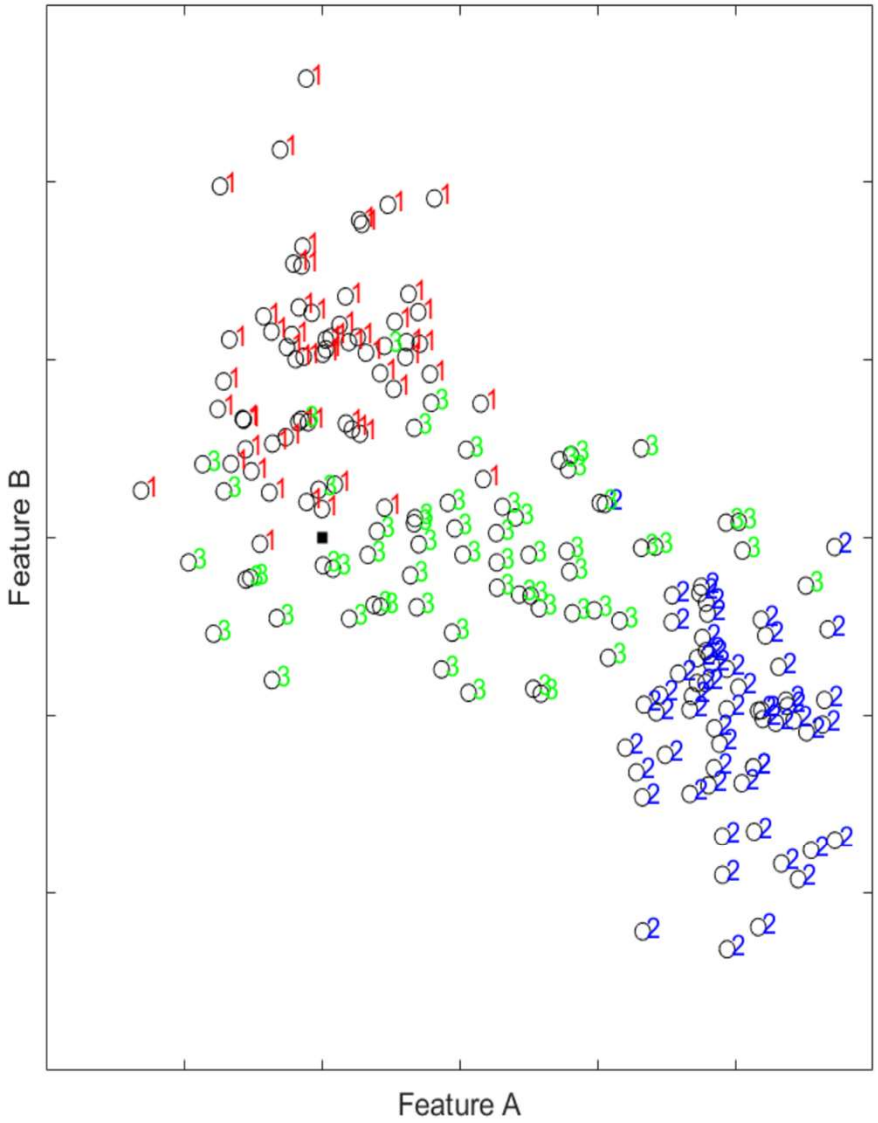
Classification with k-Nearest Neighbors

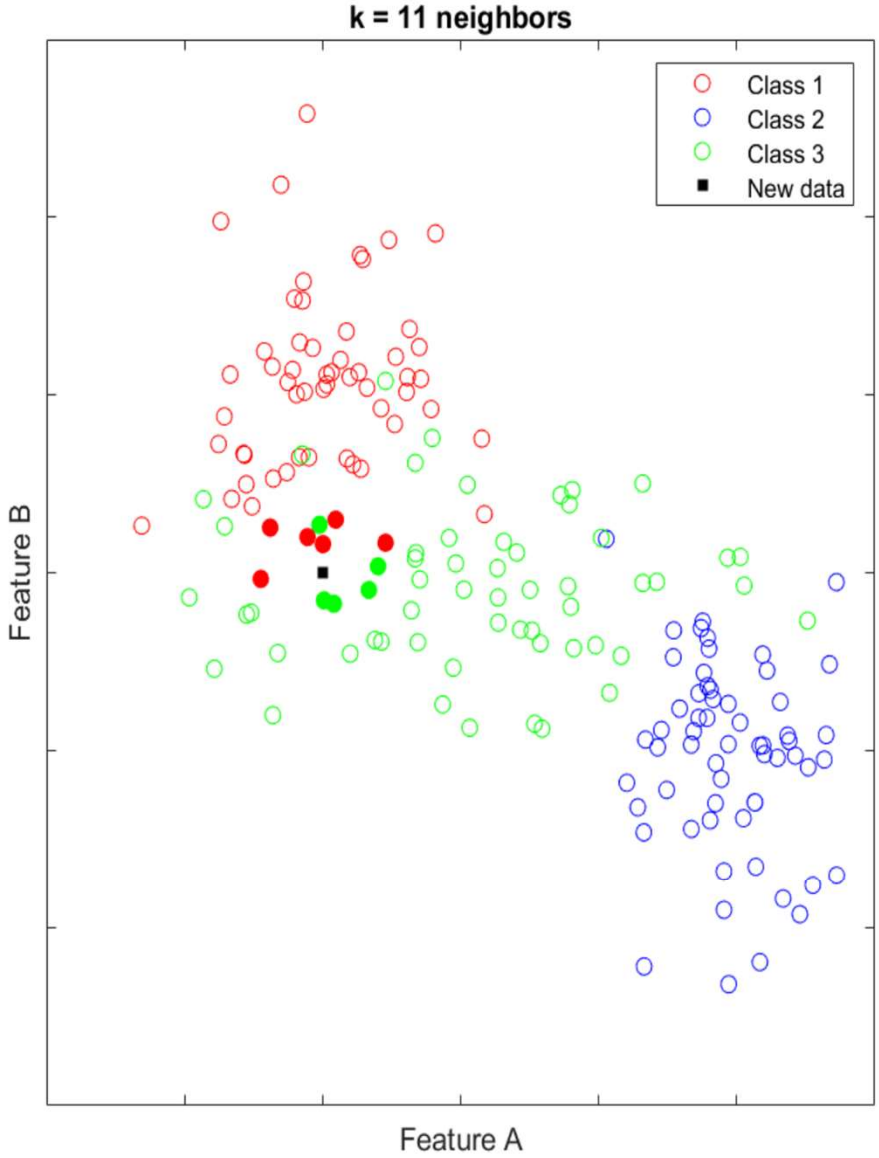
k = 11 Nearest Neighbors



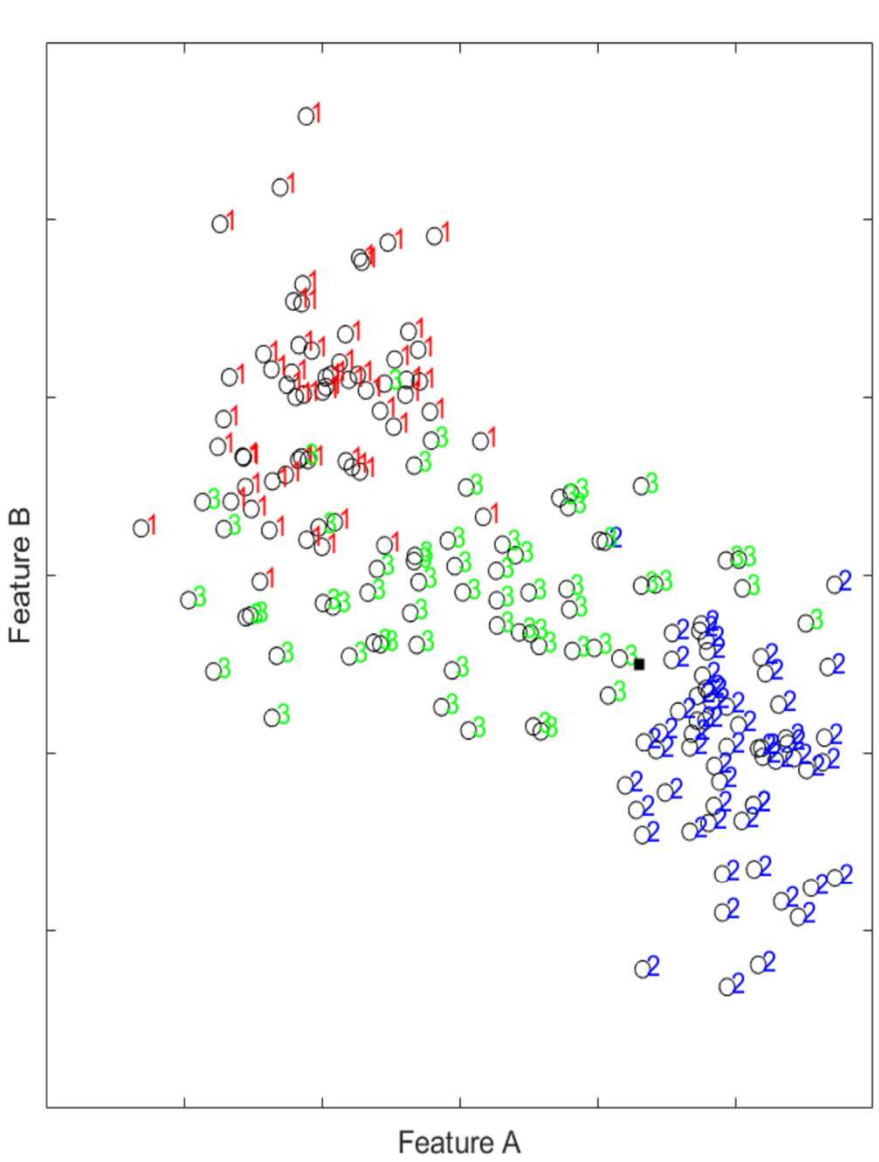


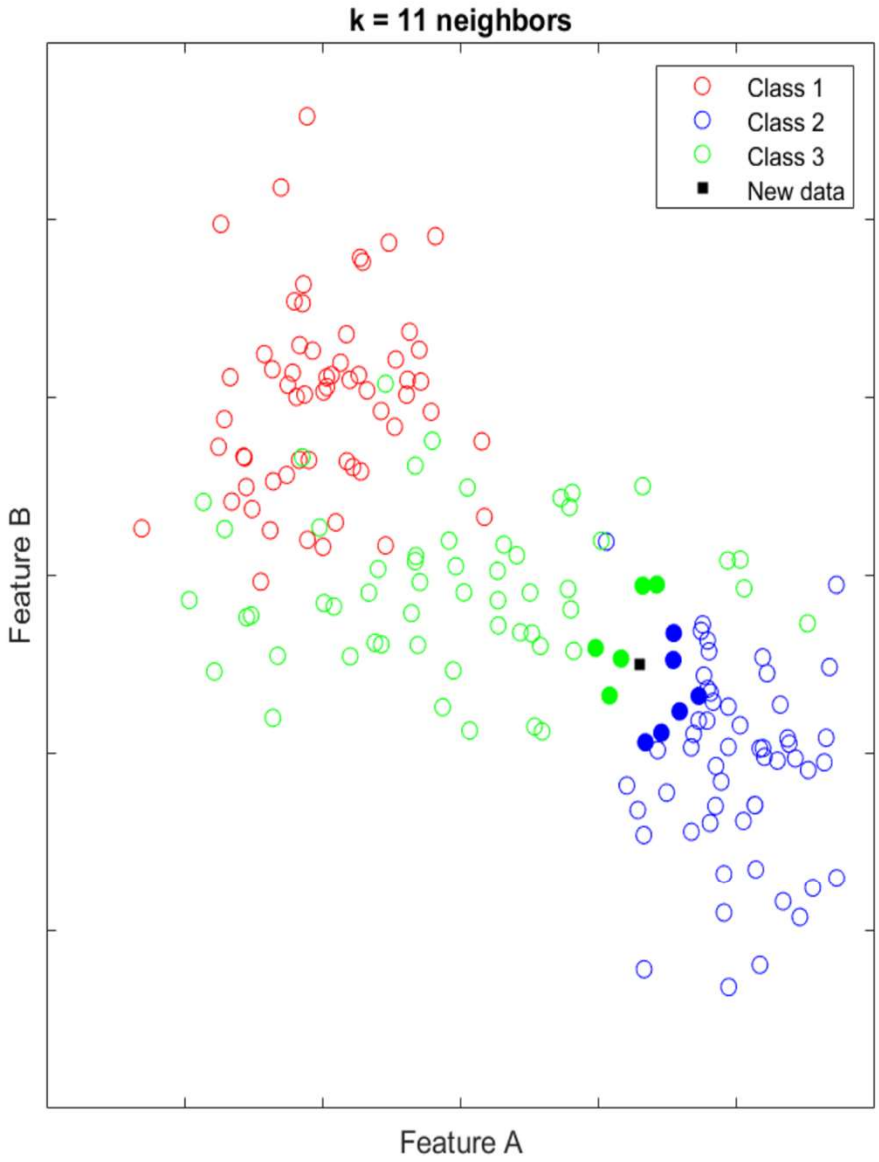
k = 11 Nearest Neighbors



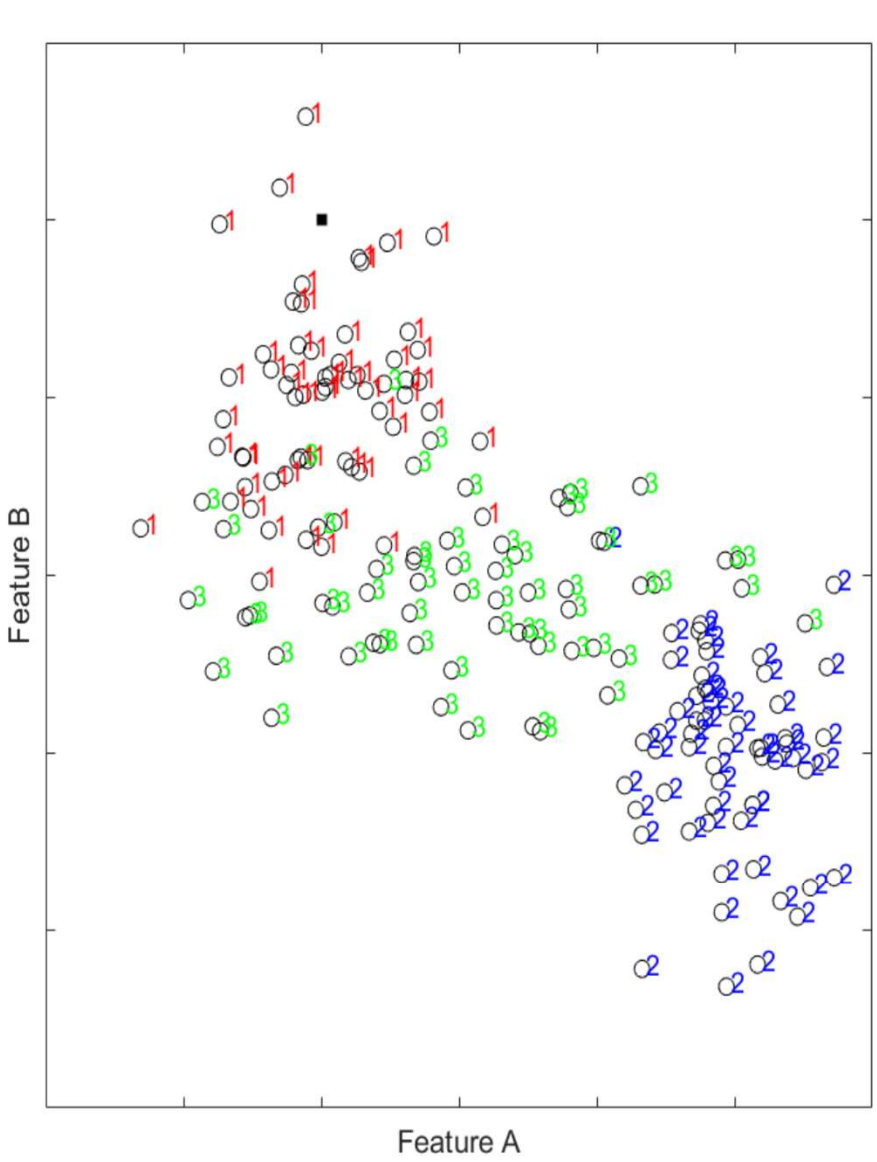


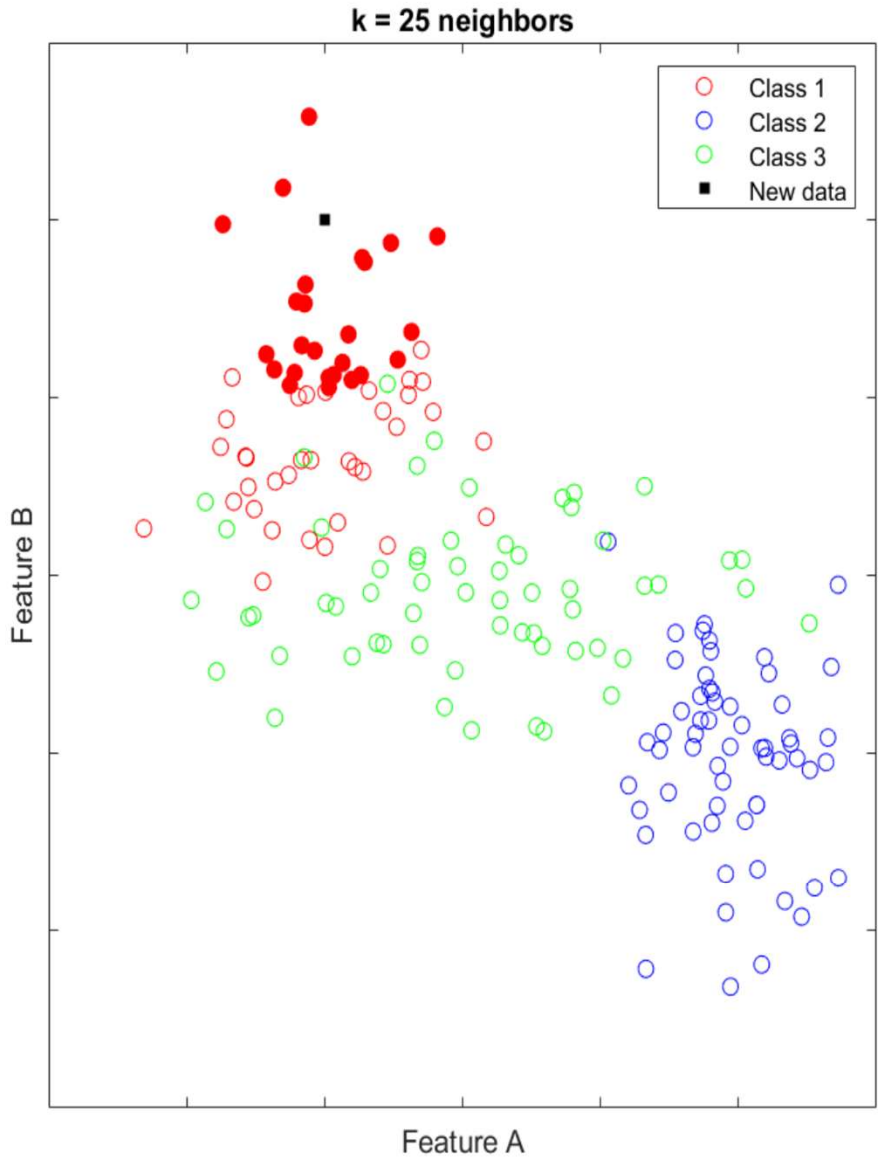
k = 11 Nearest Neighbors



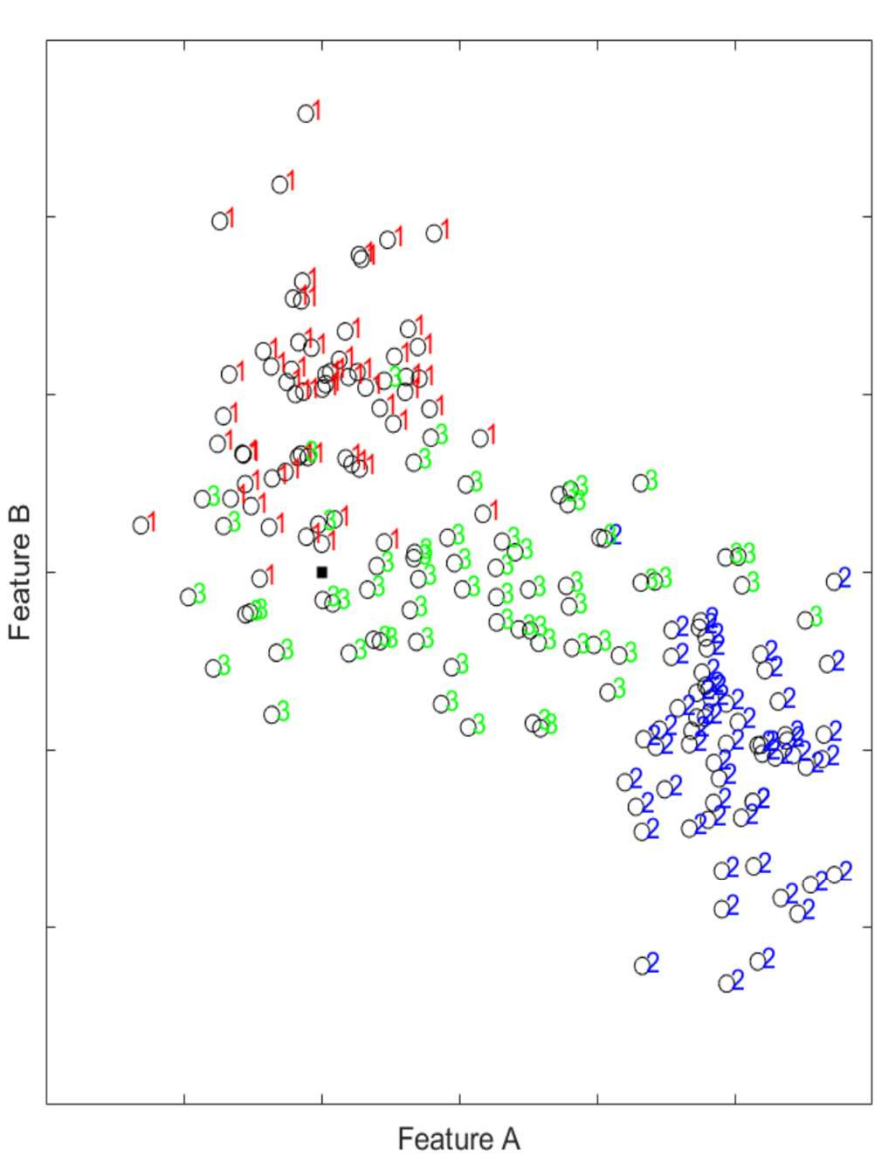


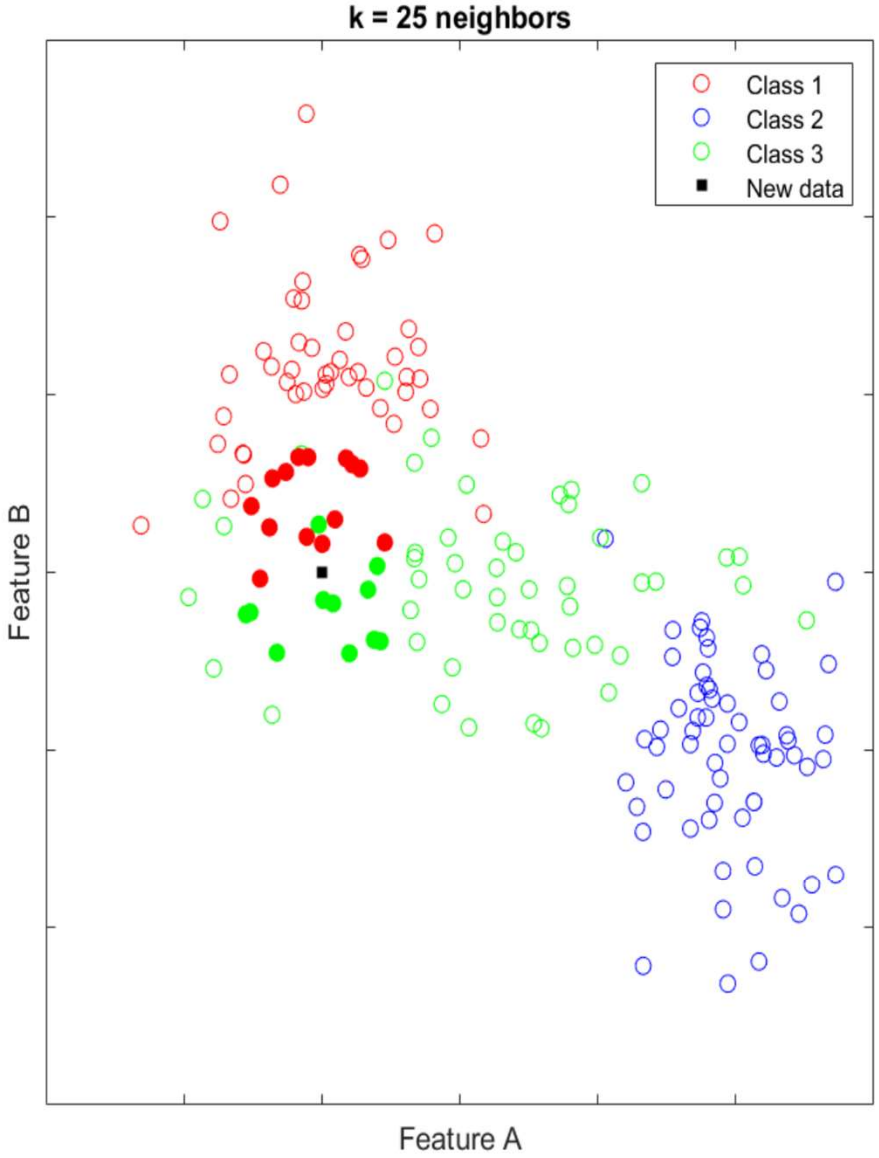
k = 25 Nearest Neighbors



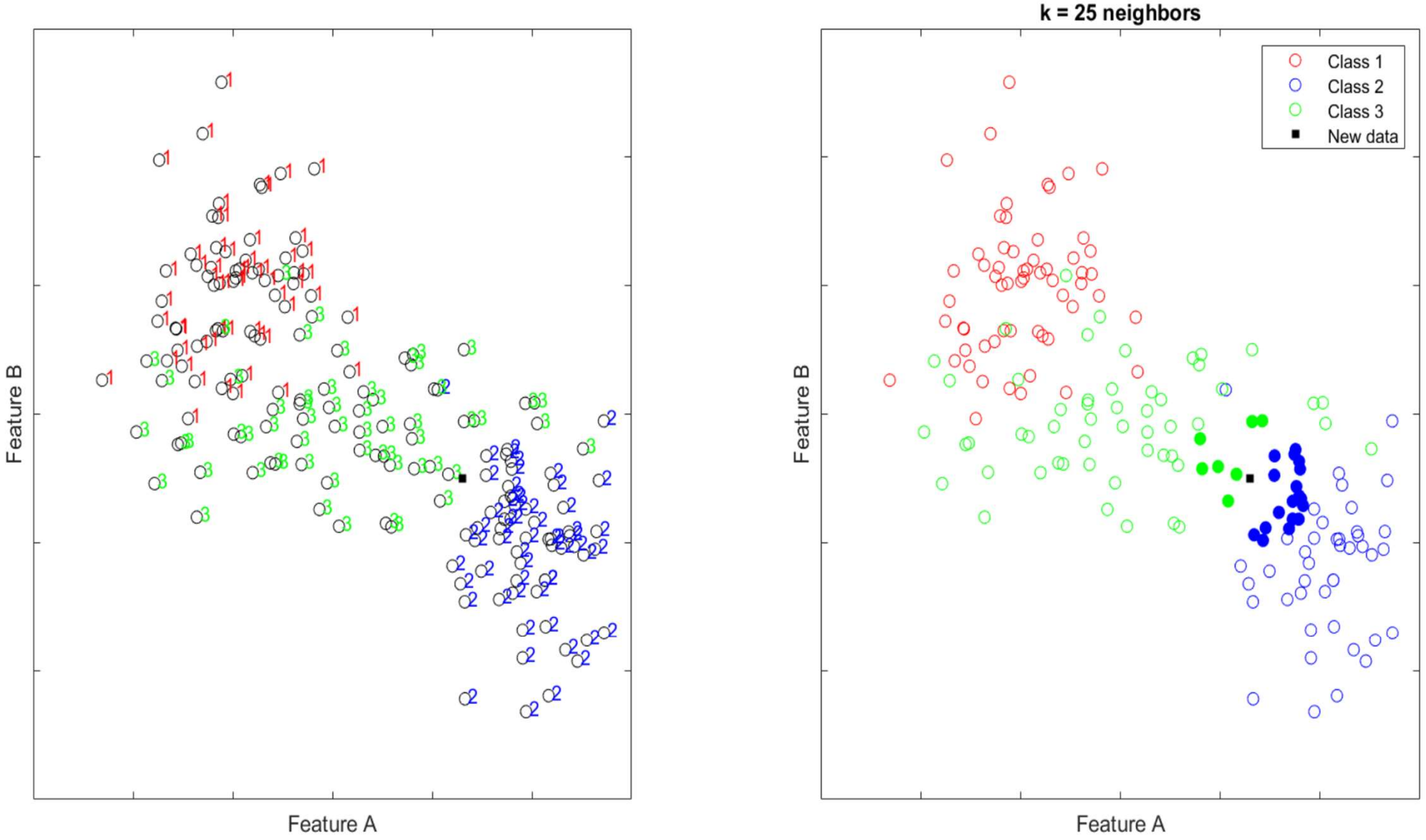


k = 25 Nearest Neighbors

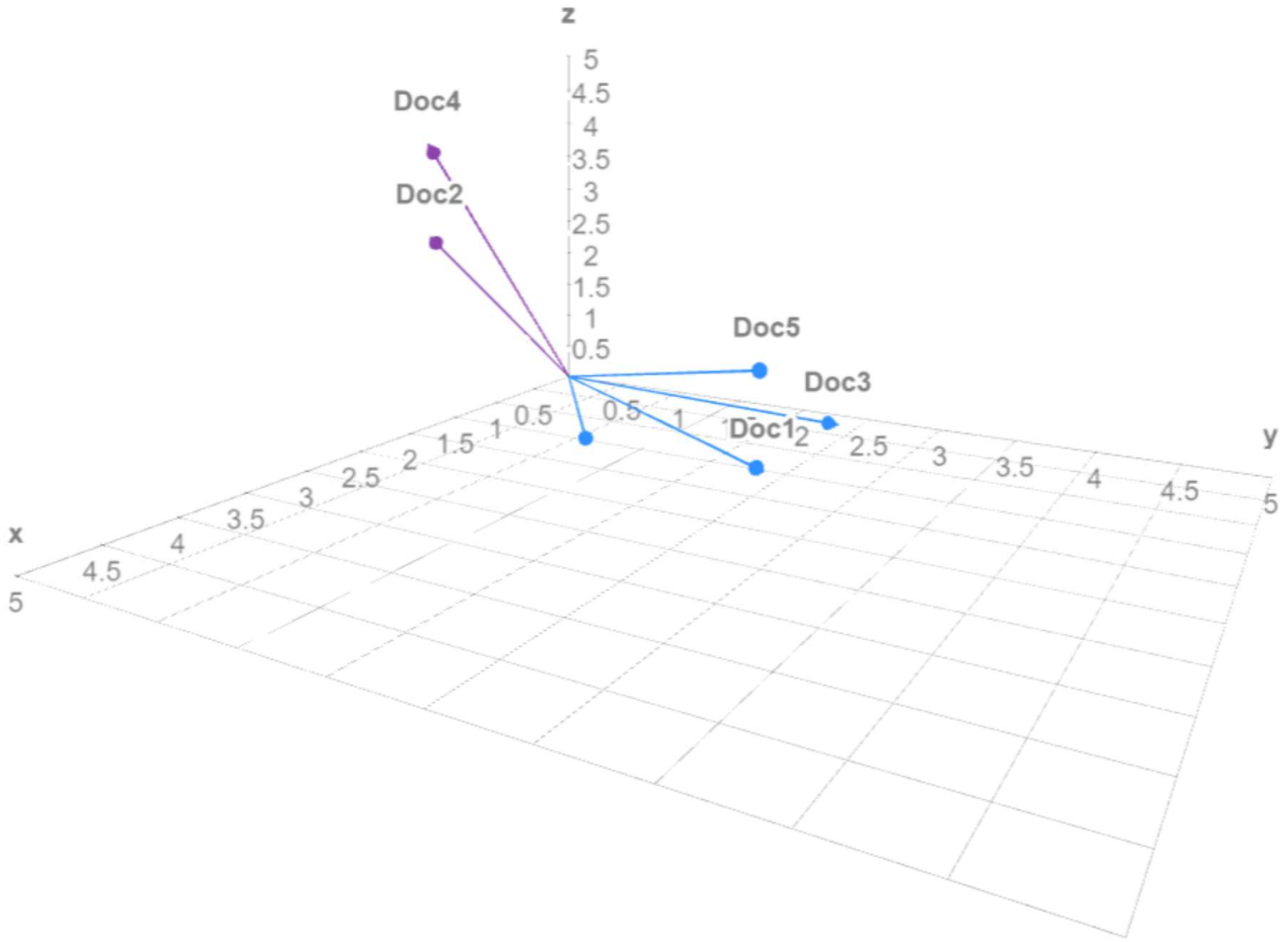




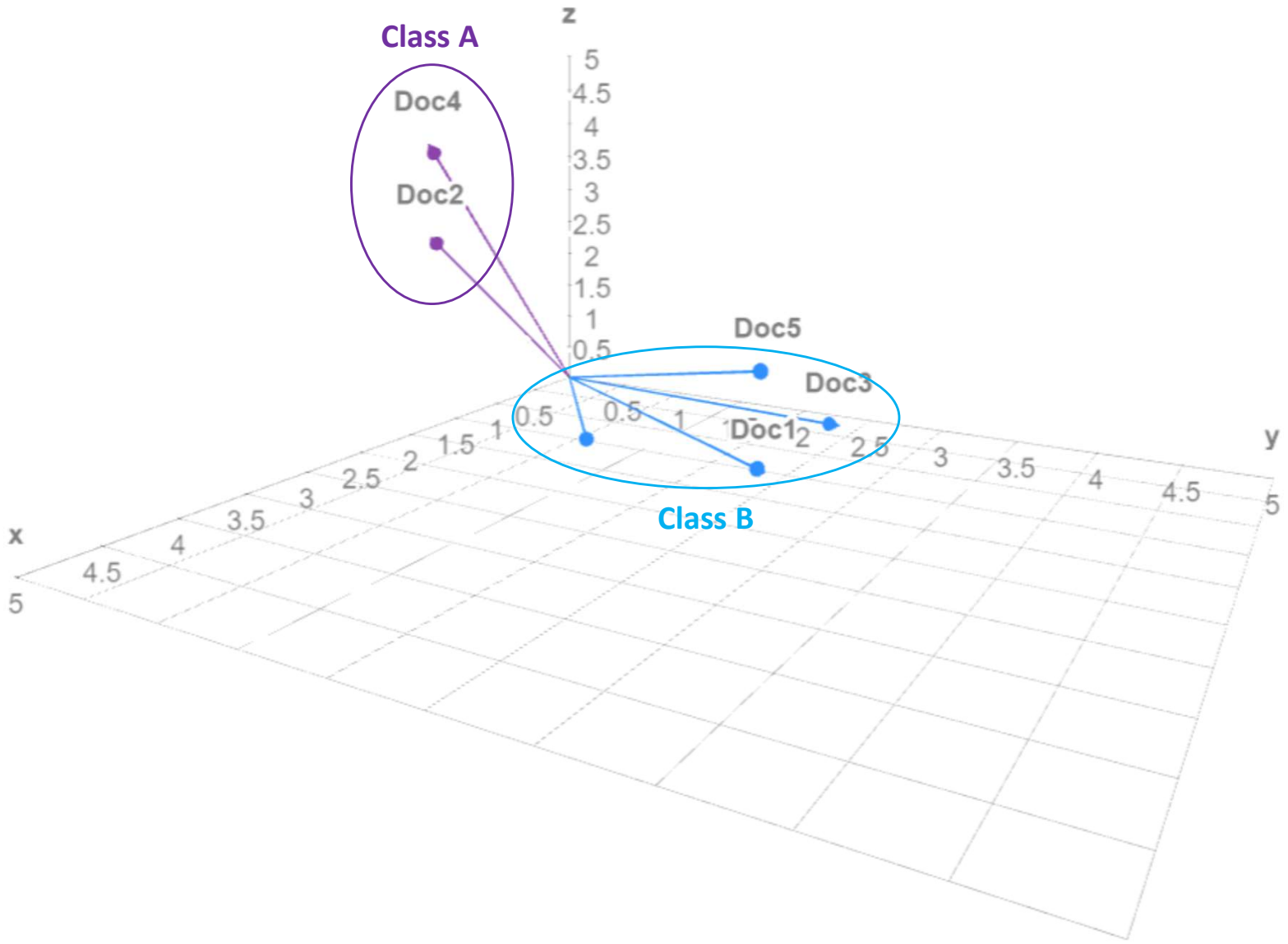
k = 25 Nearest Neighbors



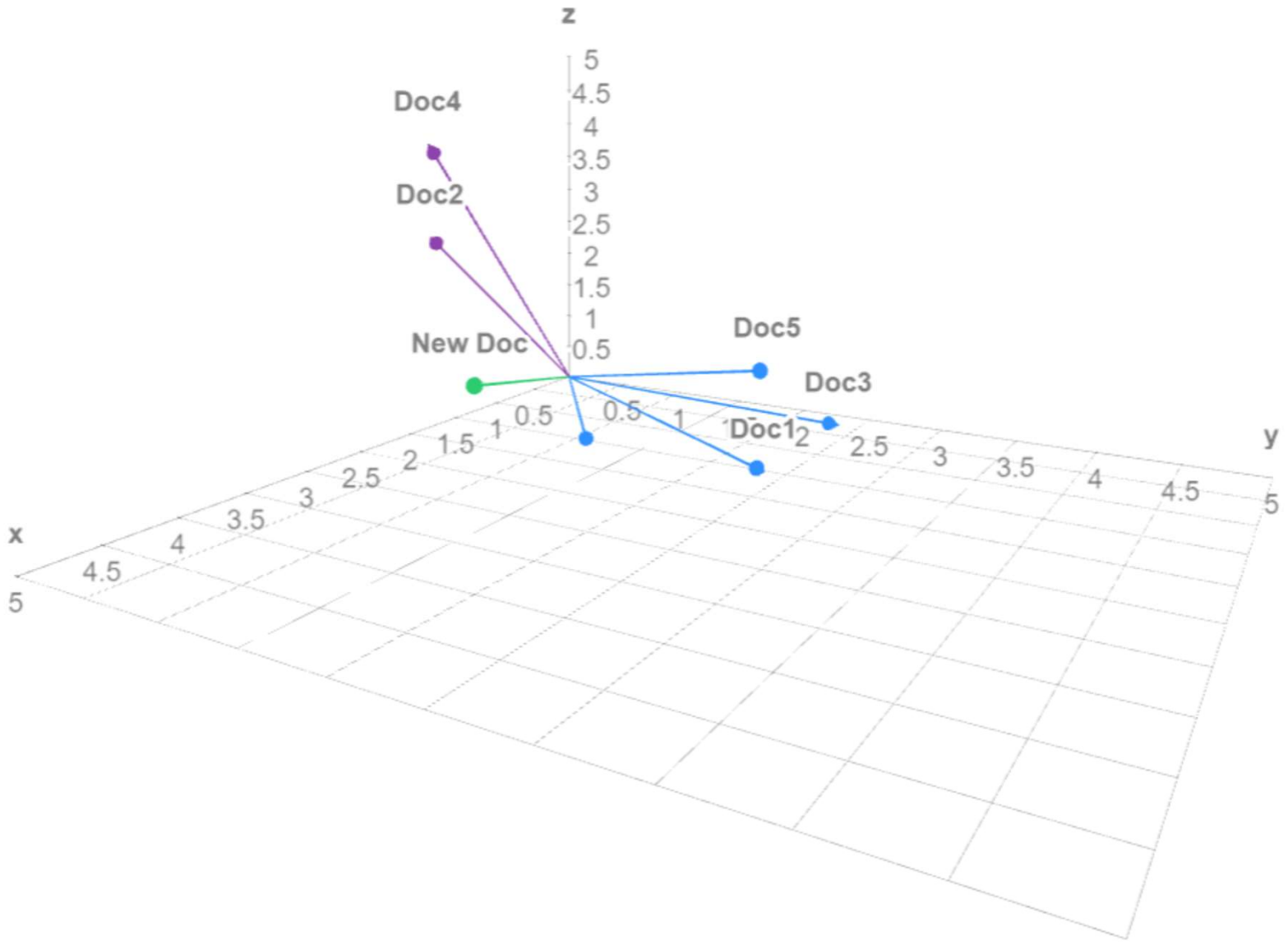
kNN: Text Classification



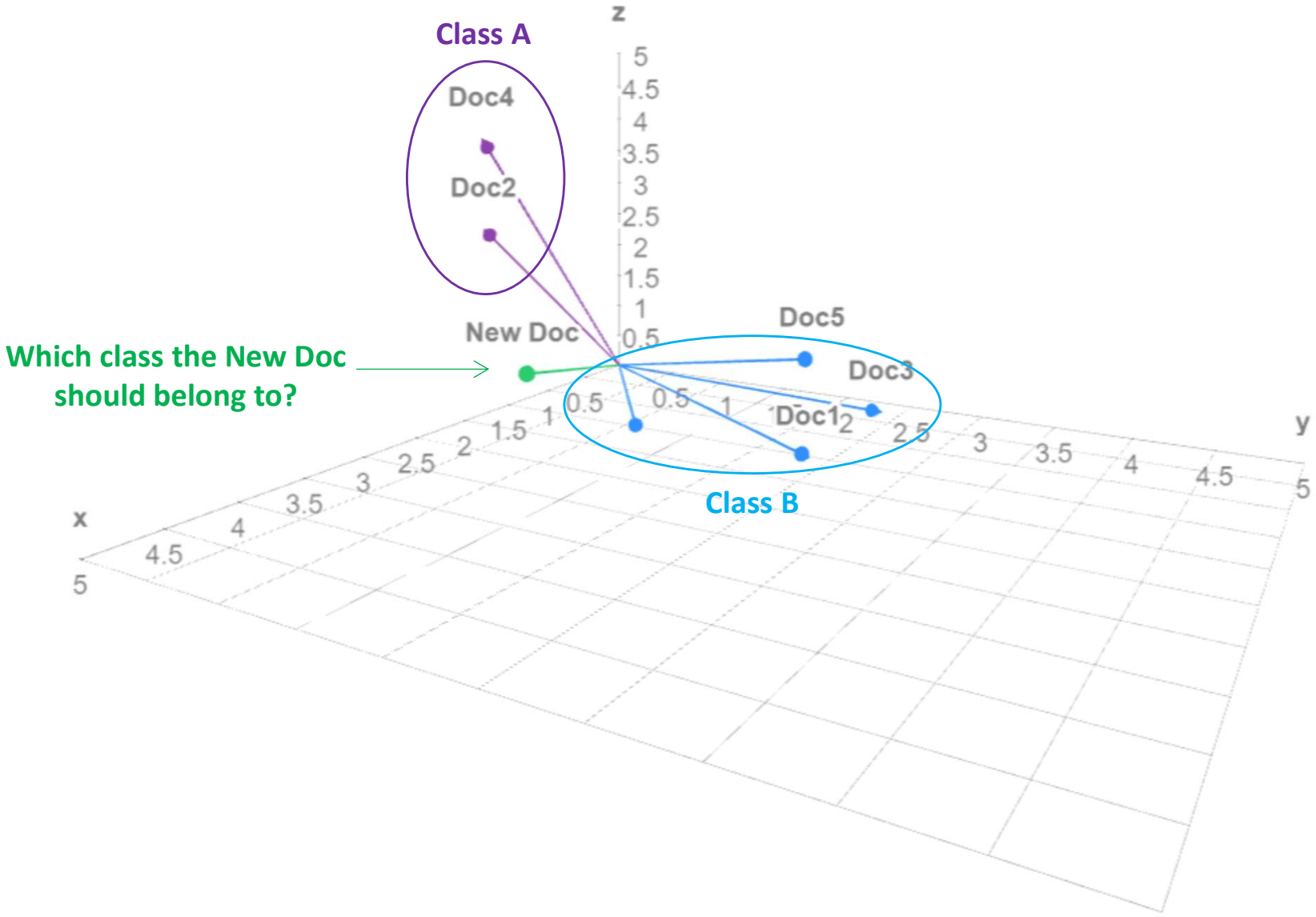
kNN: Text Classification



kNN: Text Classification

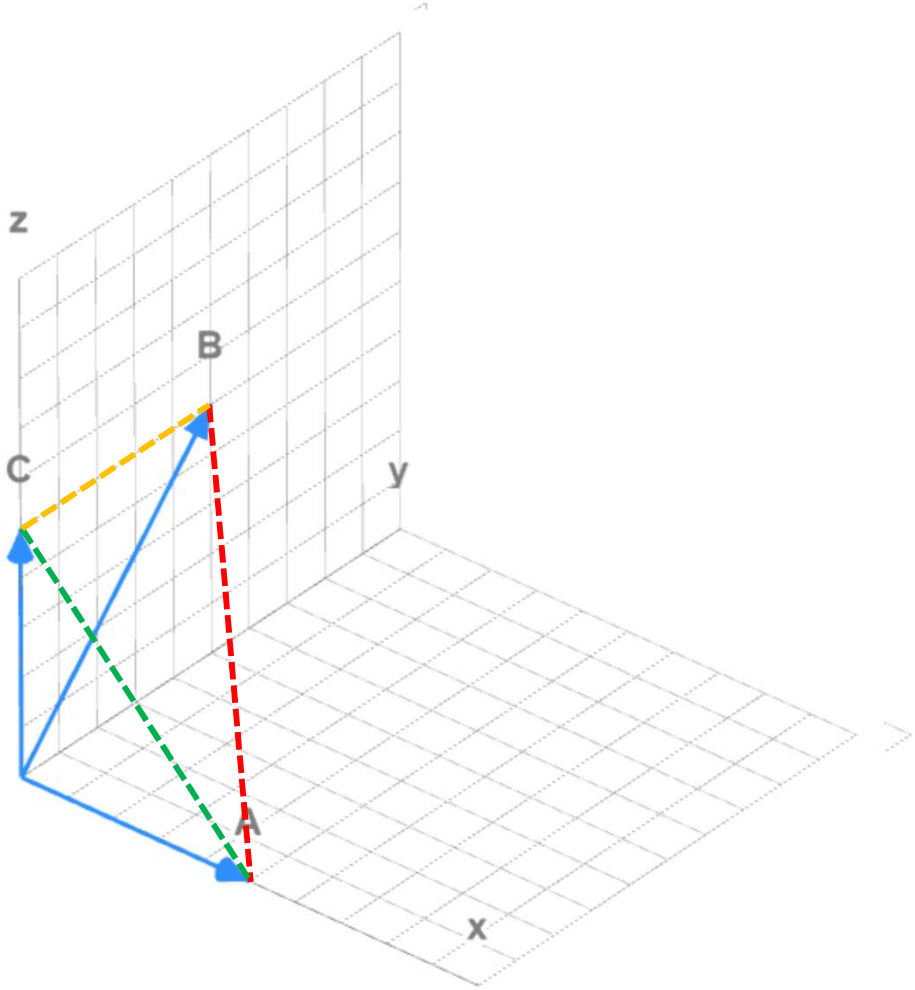


kNN: Text Classification



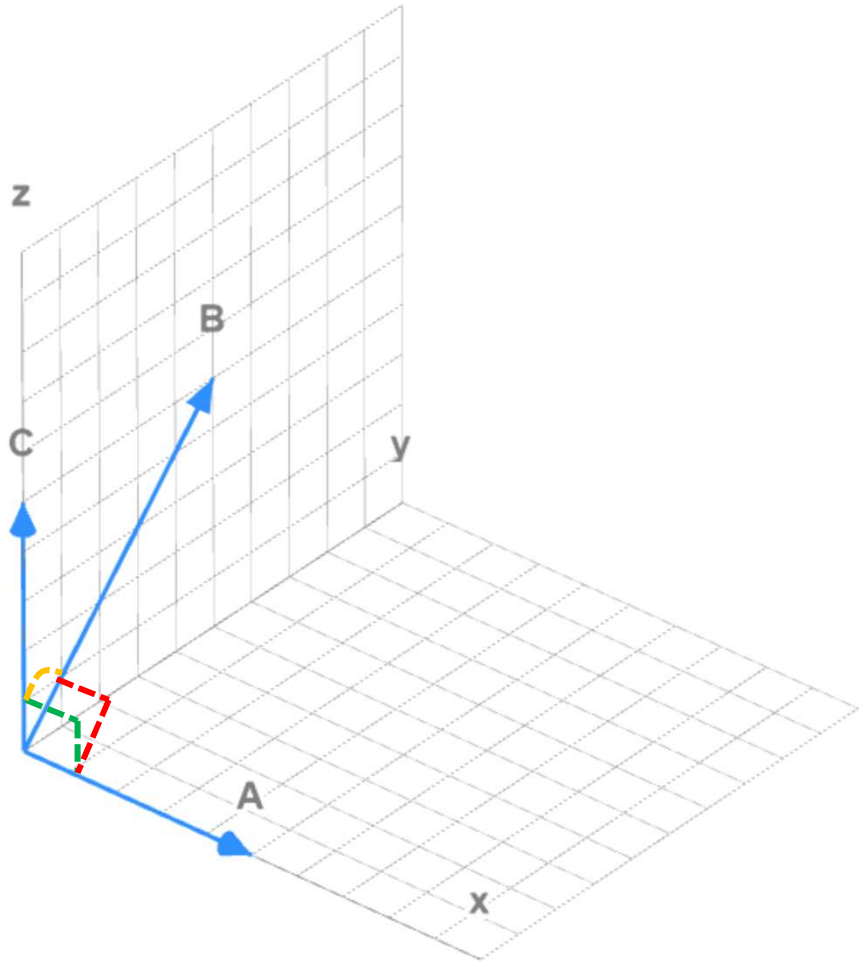
Distance Measures

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

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

$$CosineSimilarity(cat, dog) = \frac{v_{cat} \cdot v_{dog}}{\|v_{cat}\|\|v_{dog}\|} = 0$$

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

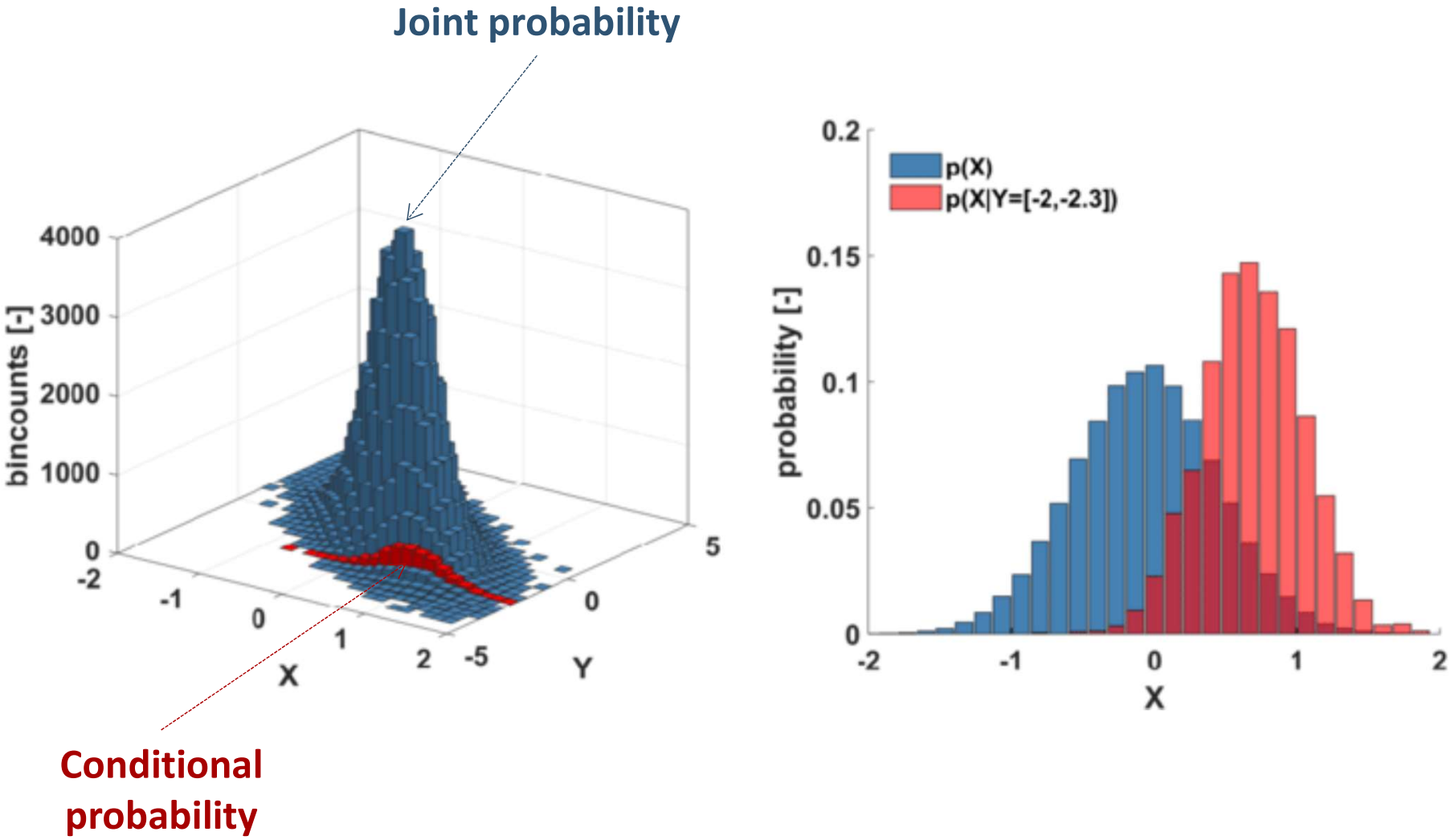
$$CosineSimilarity(D_1, D_2) = \frac{D_1 \cdot D_2}{\|D_1\|\|D_2\|} = \frac{1}{\sqrt{2}\sqrt{2}} = \frac{1}{2}$$

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



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:

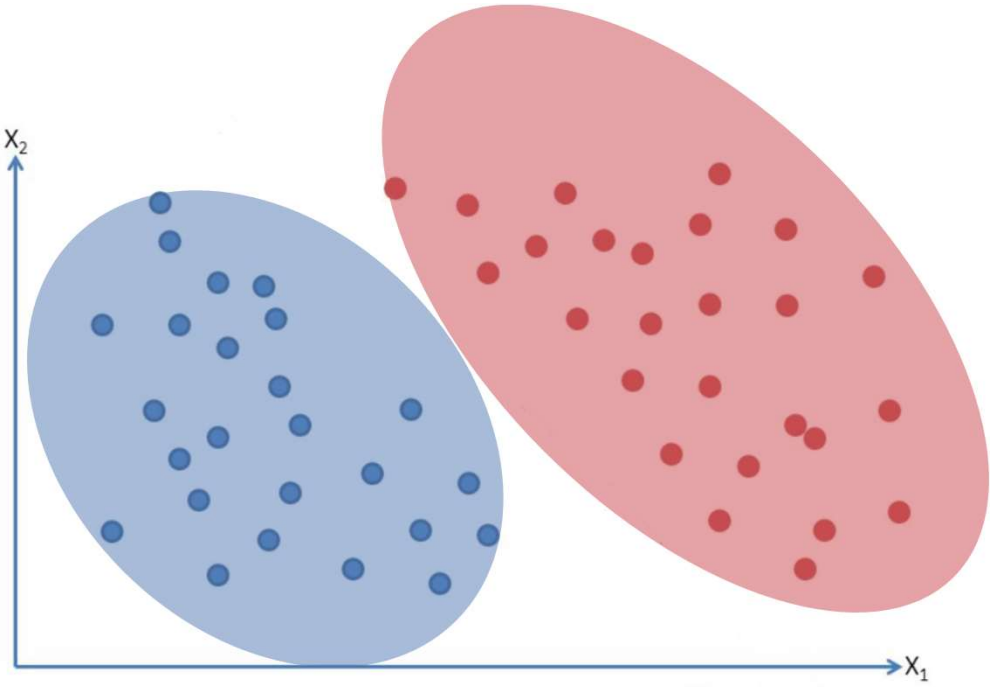
- a document $\boldsymbol{x}$
- a fixed set of classes $Y = \{y_1, y_2, \ldots, y_J\}$
- a training set of N hand-labeled documents $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)$

Output:

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

Generative vs Discriminative Models

Generative

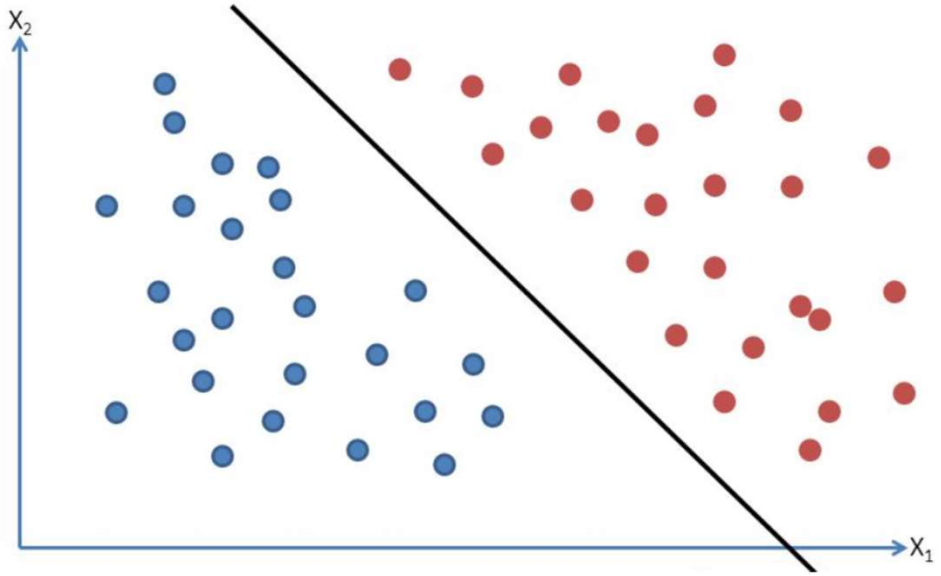


Generative model models actual distributions for EACH CLASS / LABEL / TAG

to

make a P(class | sample) **prediction**

Discriminative



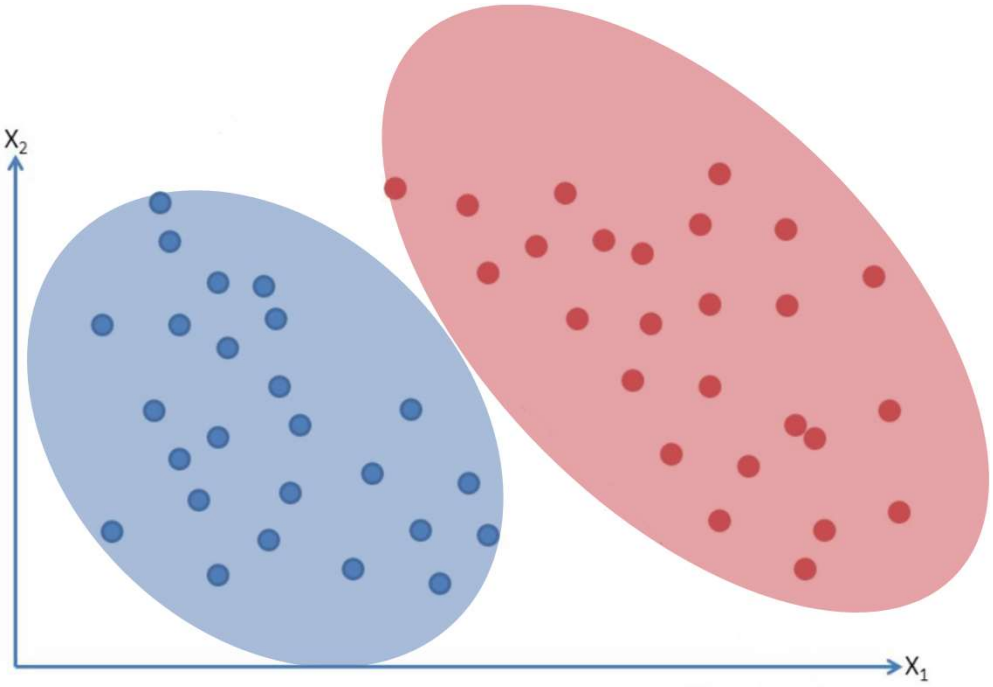
Discriminative model models the decision boundary between CLASSES / LABELS / TAGS

to

make a P(class | sample) **prediction**

Generative vs Discriminative Models

Generative

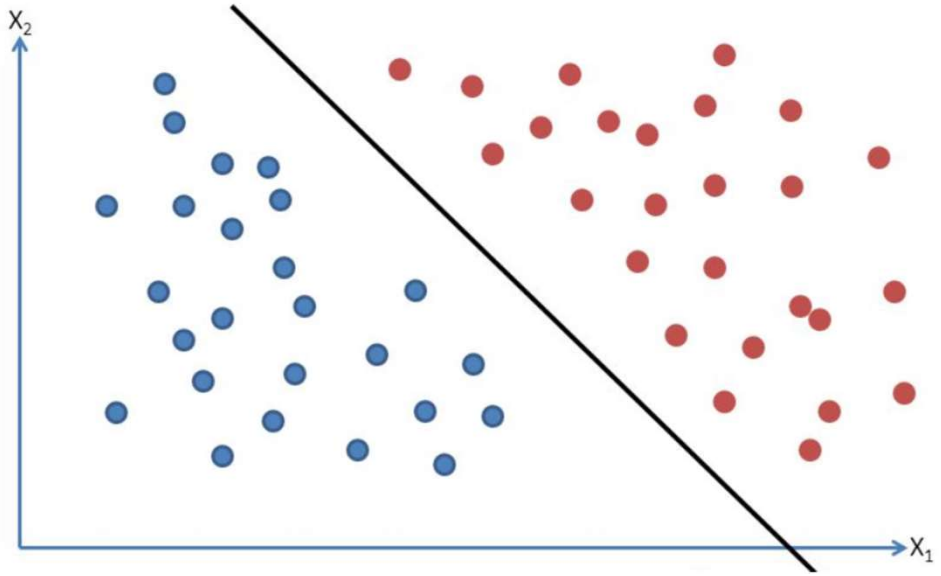


Generative model uses training data to learn P(sample, class) **joint probabilities**

and then

uses Bayes Theorem to get the P(class | sample) **prediction**

Discriminative



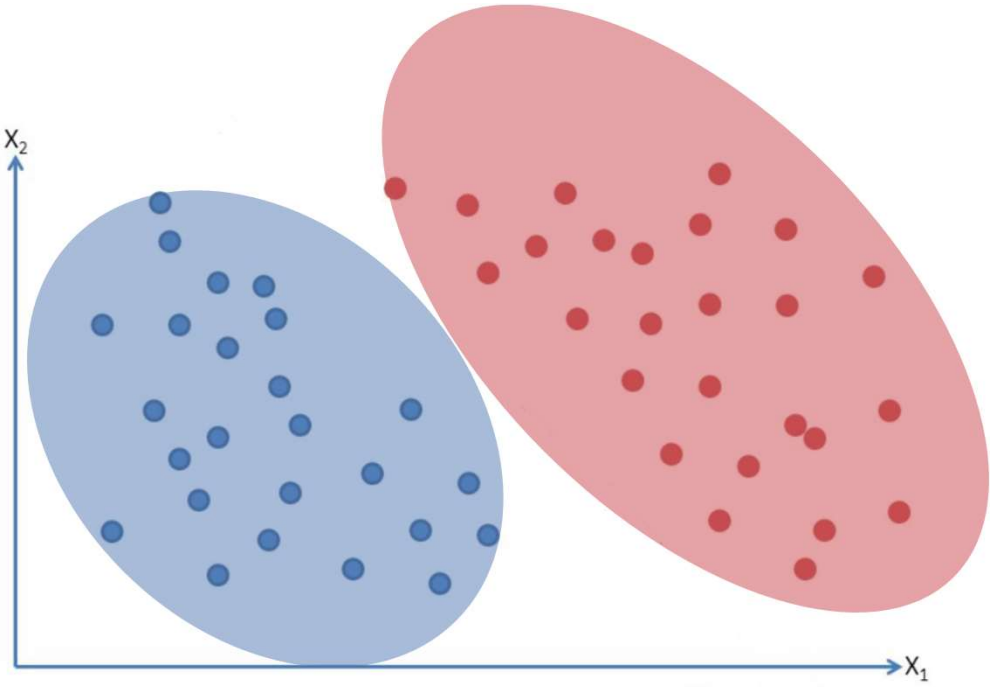
Discriminative model uses training data to learn P(class | sample) **conditional probability**

and then

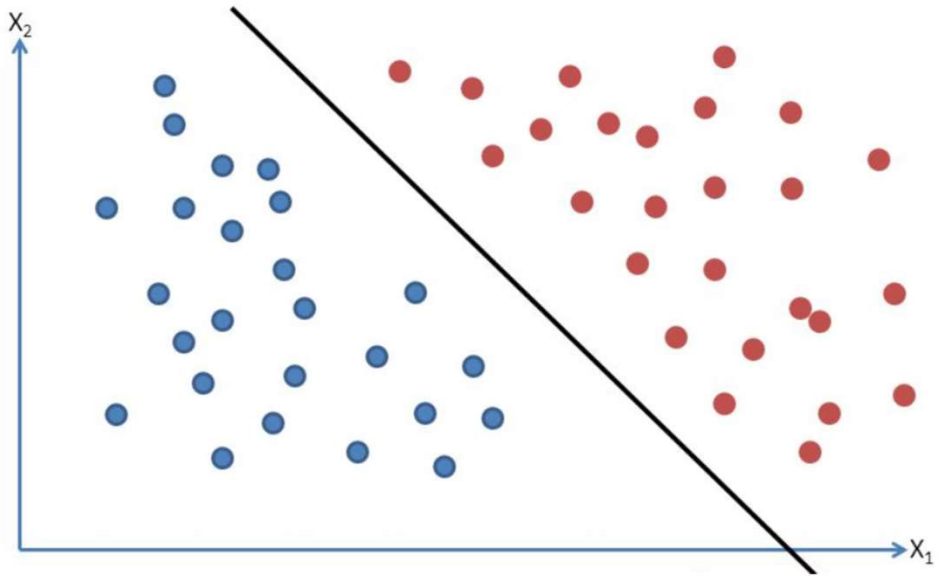
uses it to make a prediction

Generative vs Discriminative Models

Generative



Discriminative



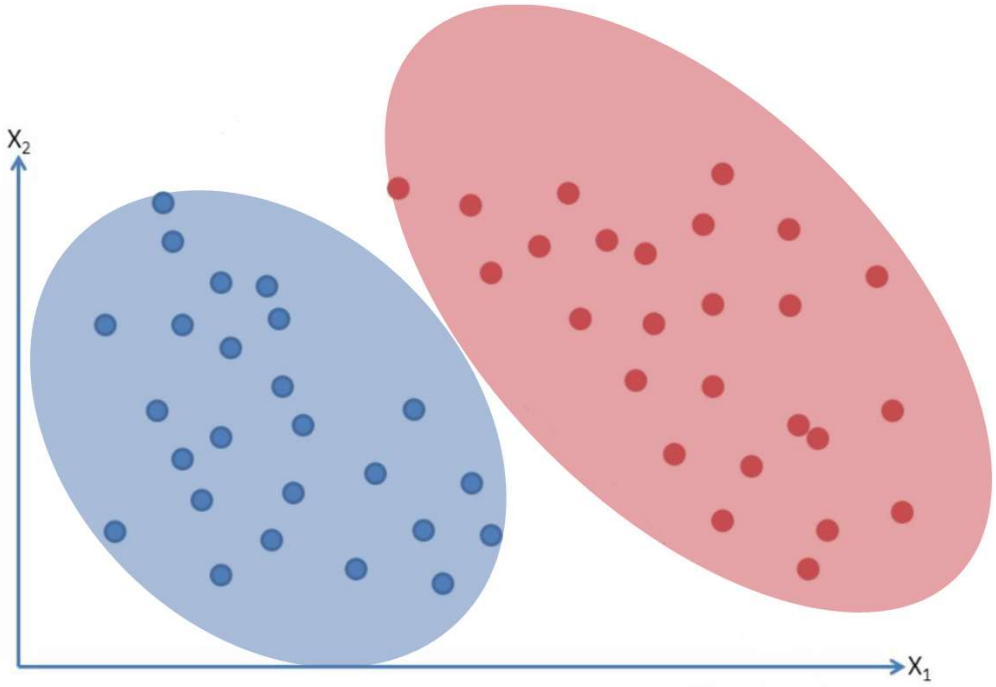
Generative model learns "details" about the class (its features):

- **includes words "*rolex*" AND "*replica*"**
- **includes word "*buy*"**
- **has spelling mistakes**
- **etc.**

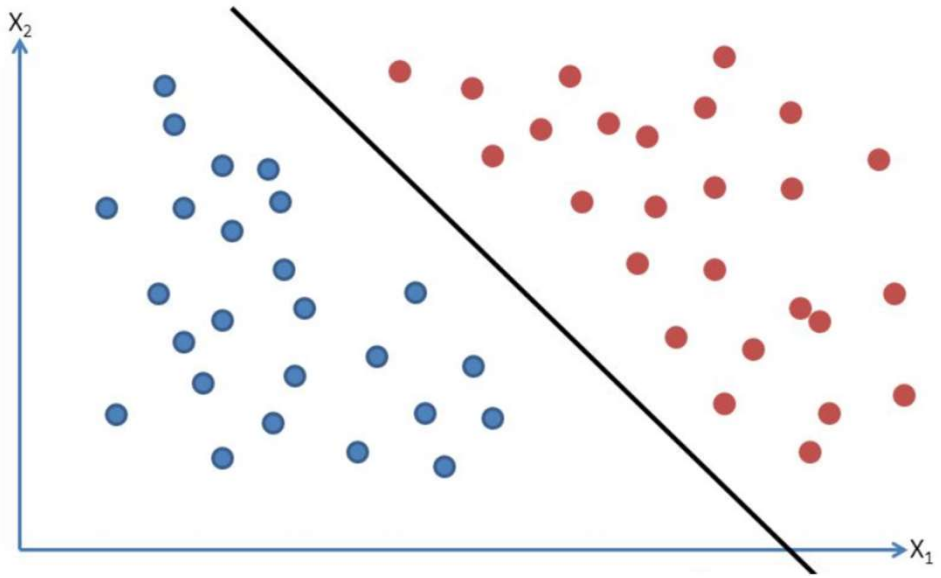
Discriminative model learns where the boundary between classes is and ignores the "details"

Generative vs Discriminative Classifier

Generative



Discriminative



Generative classifiers:

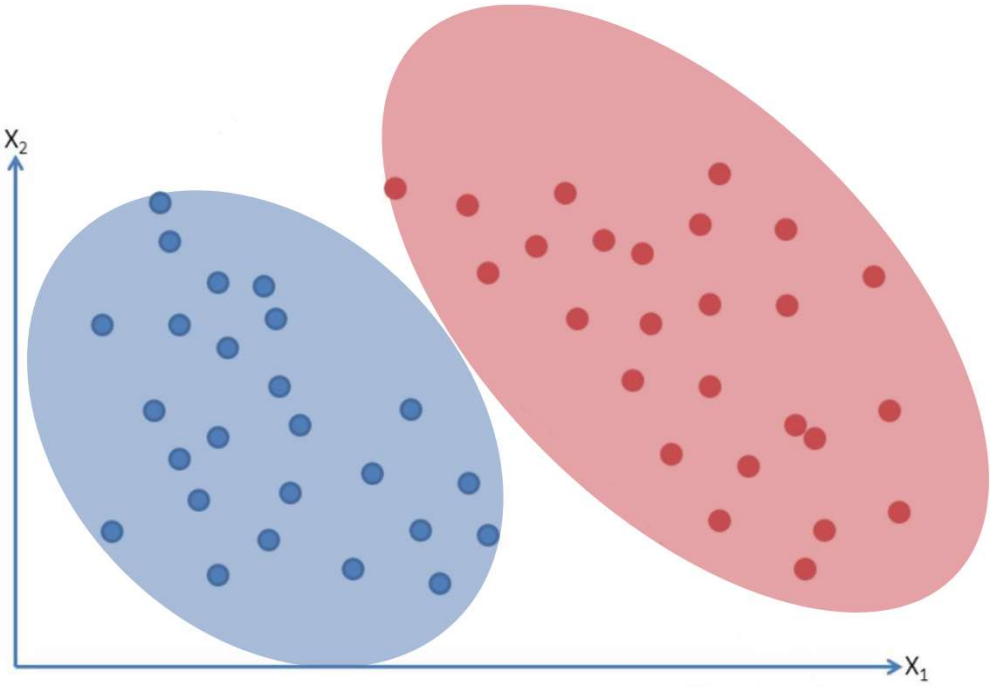
- **Assume some form of** P(class), P(sample | class)
- **Estimate** P(class), P(sample | class) **using training data**
- **Use Bayes Theorem to calculate** P(class | sample)

Discriminative classifiers:

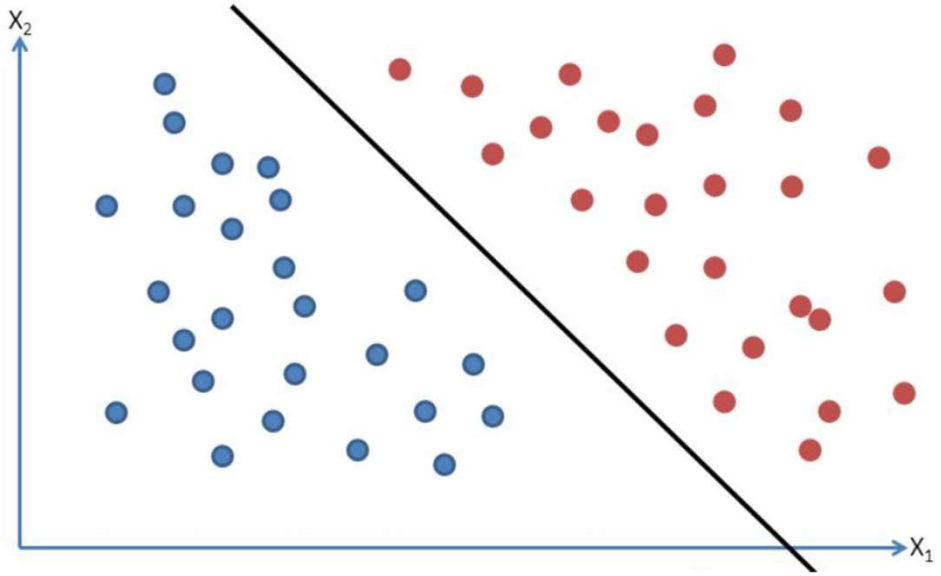
- **Assume some form of** P(class | sample)
- **Estimate** P(class | sample) **using training data**

Generative vs Discriminative Classifier

Generative



Discriminative



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

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

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

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

Classification: Conditional Probability

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

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

$$P(y \mid x_1 \wedge x_2 \wedge \ldots \wedge x_N) = \frac{P(x_1 \wedge x_2 \wedge \ldots \wedge x_N \mid y) * P(y)}{P(x_1 \wedge x_2 \wedge \ldots \wedge x_N)}$$

Conditional Probability (Product Rule)

$$\boldsymbol{P}(x_1 \wedge x_2 \wedge \ldots \wedge x_N \mid y) * \boldsymbol{P}(y) = \boldsymbol{P}(x_1 \wedge x_2 \wedge \ldots \wedge x_N \wedge y)$$

so:

$$\boldsymbol{P}(x_1 \wedge x_2 \wedge \ldots \wedge x_N \wedge y) = \boldsymbol{P}(x_1 \wedge x_2 \wedge \ldots \wedge x_N \mid y) * \boldsymbol{P}(y)$$

Classification: Conditional Probability

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

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

$$P(y \mid x_1 \wedge x_2 \wedge \ldots \wedge x_N) = \frac{P(x_1 \wedge x_2 \wedge \ldots \wedge x_N \mid y) * P(y)}{P(x_1 \wedge x_2 \wedge \ldots \wedge x_N)}$$

$$P(y \mid x_1 \wedge x_2 \wedge \ldots \wedge x_N) = \frac{P(x_1 \wedge x_2 \wedge \ldots \wedge x_N \wedge y)}{P(x_1 \wedge x_2 \wedge \ldots \wedge x_N)}$$

Classification: Conditional Probability

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

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

$$P(y \mid x_1 \wedge x_2 \wedge \ldots \wedge x_N) = \frac{P(x_1 \wedge x_2 \wedge \ldots \wedge x_N \mid y) * P(y)}{P(x_1 \wedge x_2 \wedge \ldots \wedge x_N)}$$

Conditional probability

Joint probability

$$P(y \mid x_1 \wedge x_2 \wedge \ldots \wedge x_N) = \frac{P(x_1 \wedge x_2 \wedge \ldots \wedge x_N \wedge y)}{P(x_1 \wedge x_2 \wedge \ldots \wedge x_N)}$$

Classification: Conditional Probability

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

With Naive Bayes we went after THIS to get THAT …

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

$$P(y \mid x_1 \wedge x_2 \wedge \ldots \wedge x_N) = \frac{P(x_1 \wedge x_2 \wedge \ldots \wedge x_N \wedge y)}{P(x_1 \wedge x_2 \wedge \ldots \wedge x_N)}$$

… while making some really Naive assumptions along the way

Classification: Conditional Probability

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

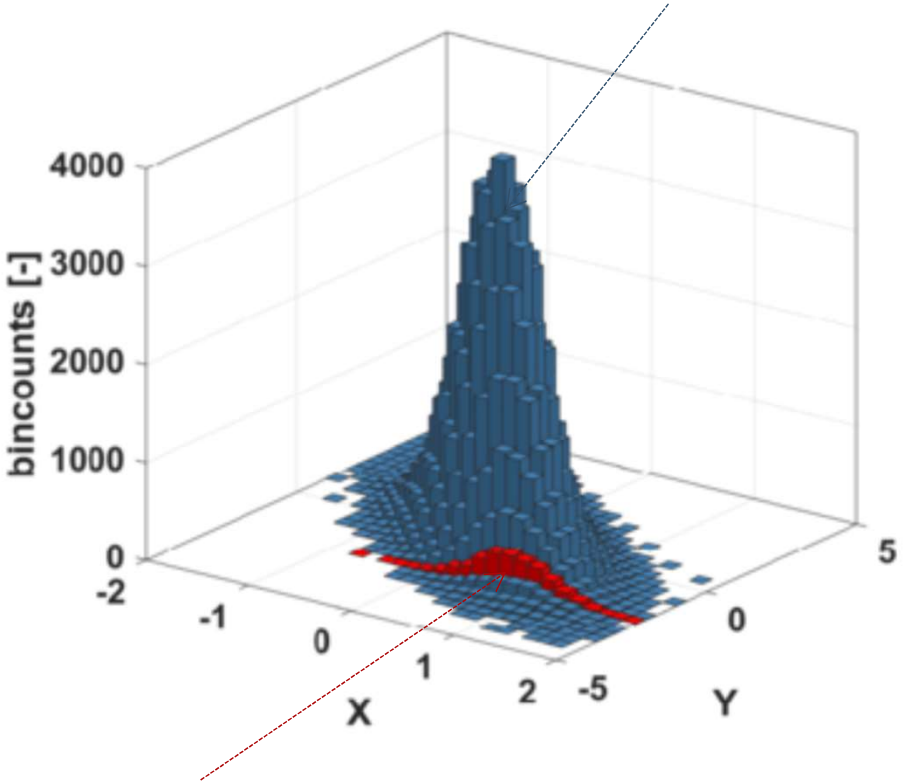
Logistic Regression will learn THIS directly

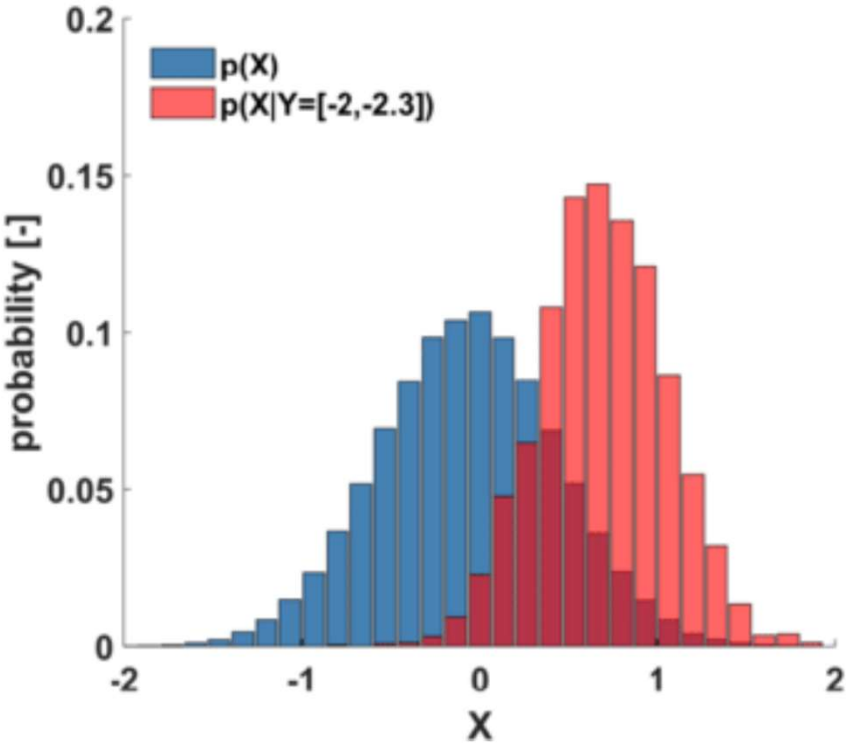
$\mathbf{x} = x_1, x_2, \ldots, x_N$, **so:**

$$P(y \mid x_1 \wedge x_2 \wedge \ldots \wedge x_N) = \frac{P(x_1 \wedge x_2 \wedge \ldots \wedge x_N \wedge y)}{P(x_1 \wedge x_2 \wedge \ldots \wedge x_N)}$$

Joint vs Conditional Probability

Naive Bayes learned THIS...



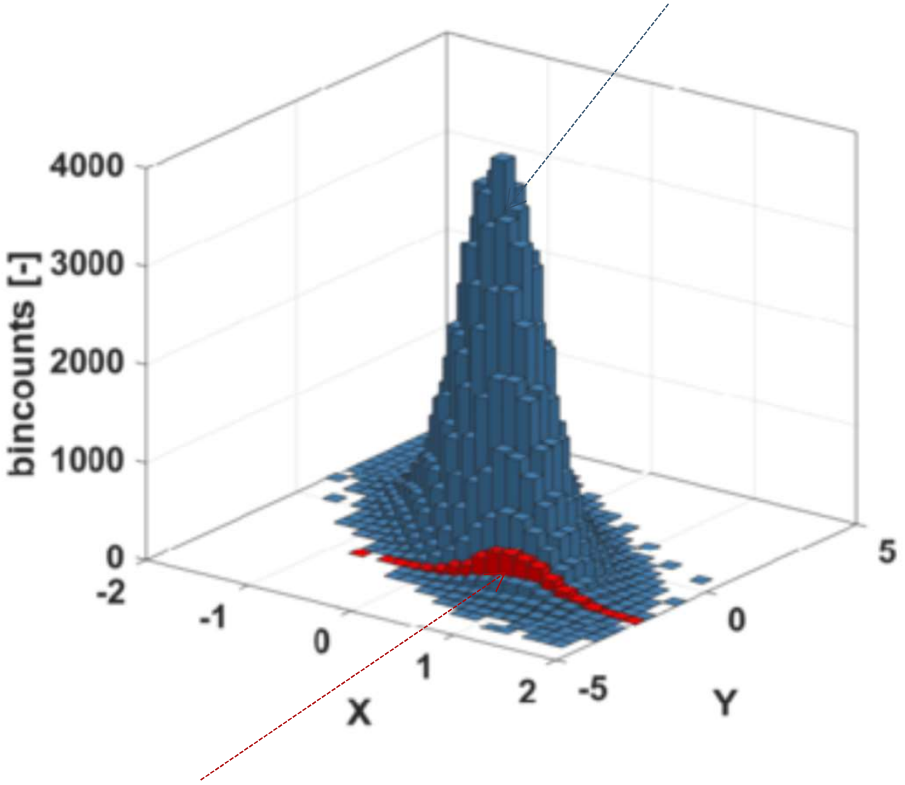


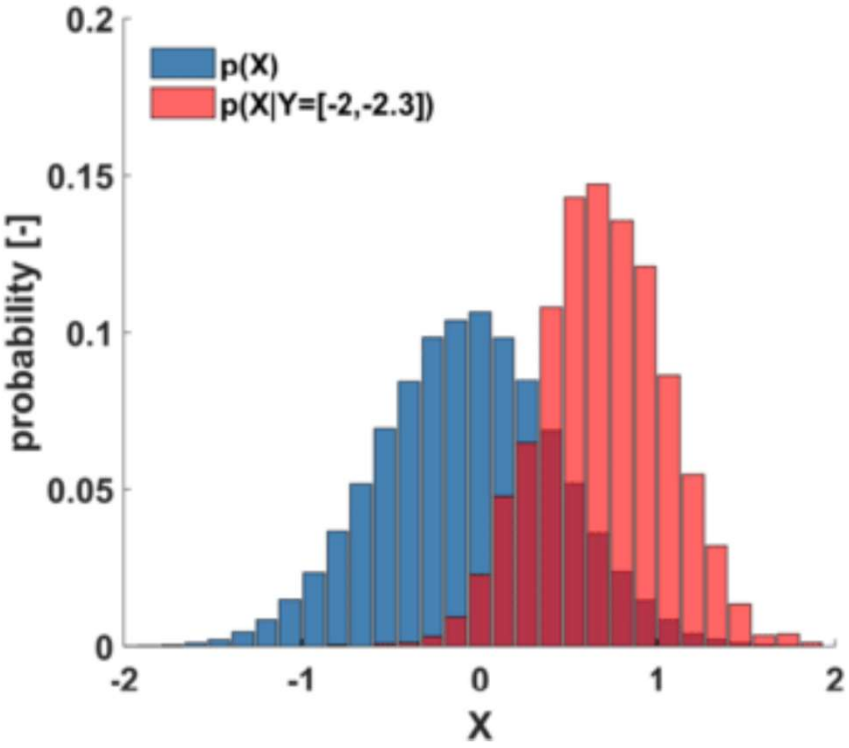
... to be able to estimate THAT

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...

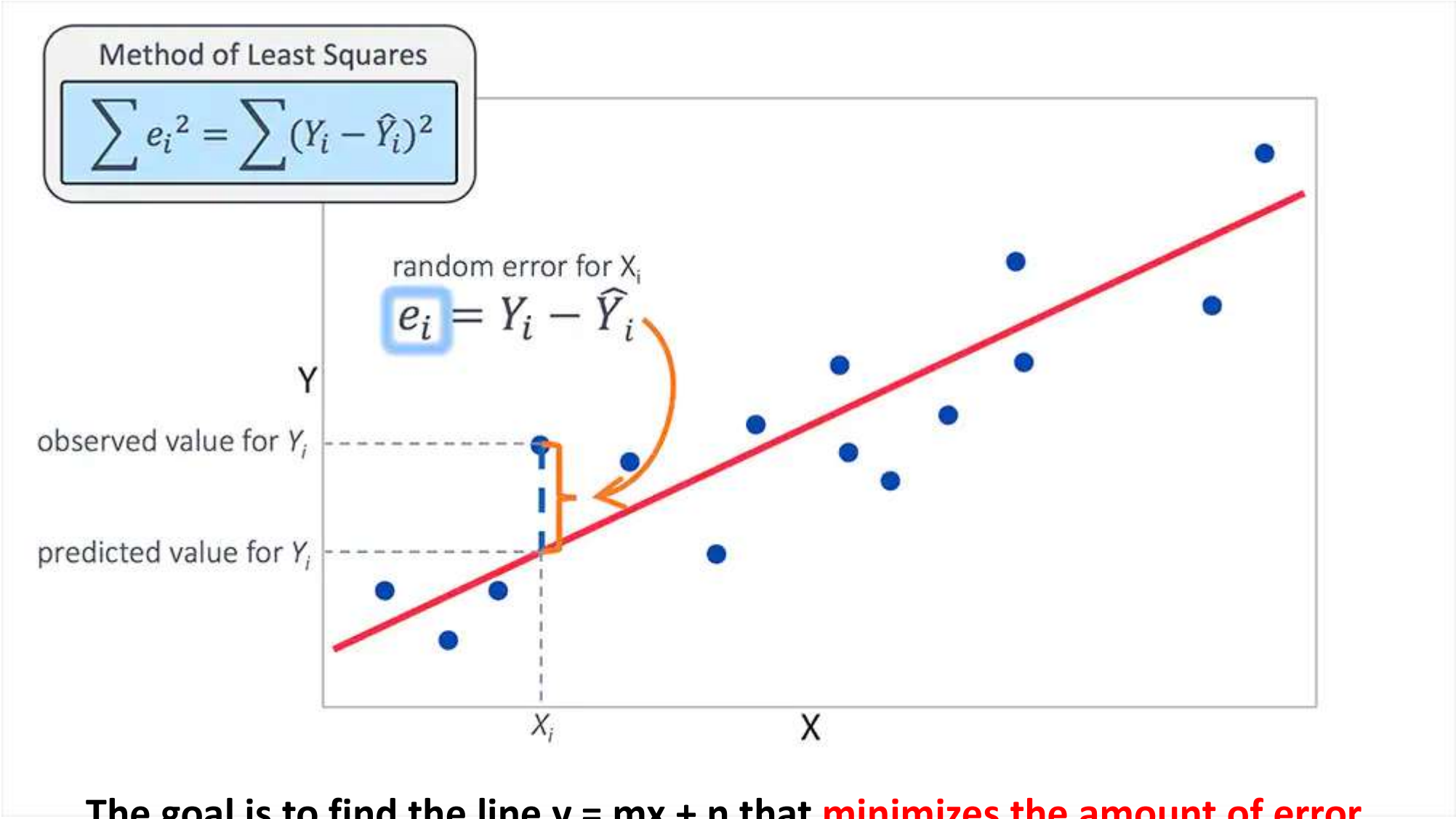




... and learns THAT instead

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

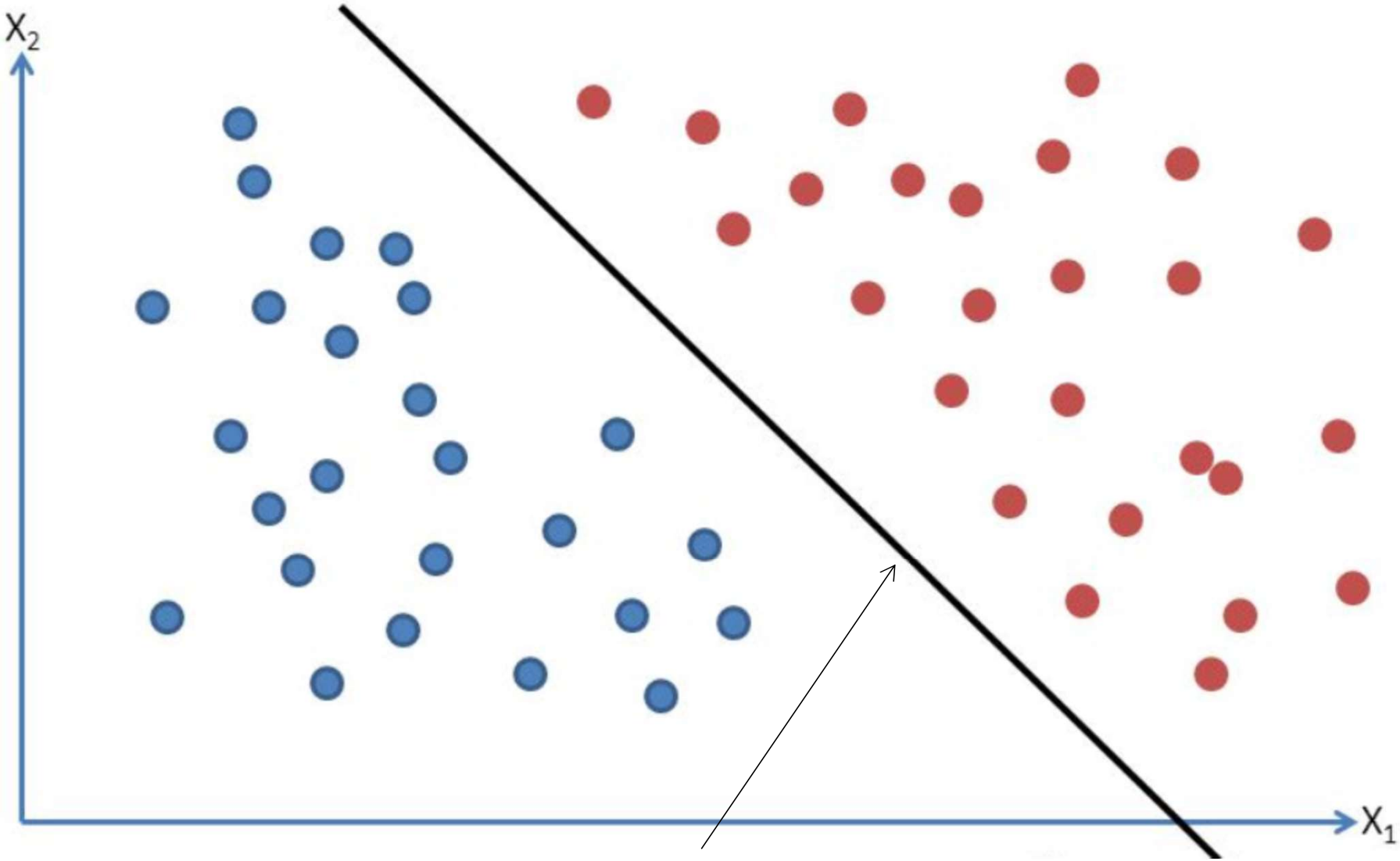
Linear Regression Using Least-Squares



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

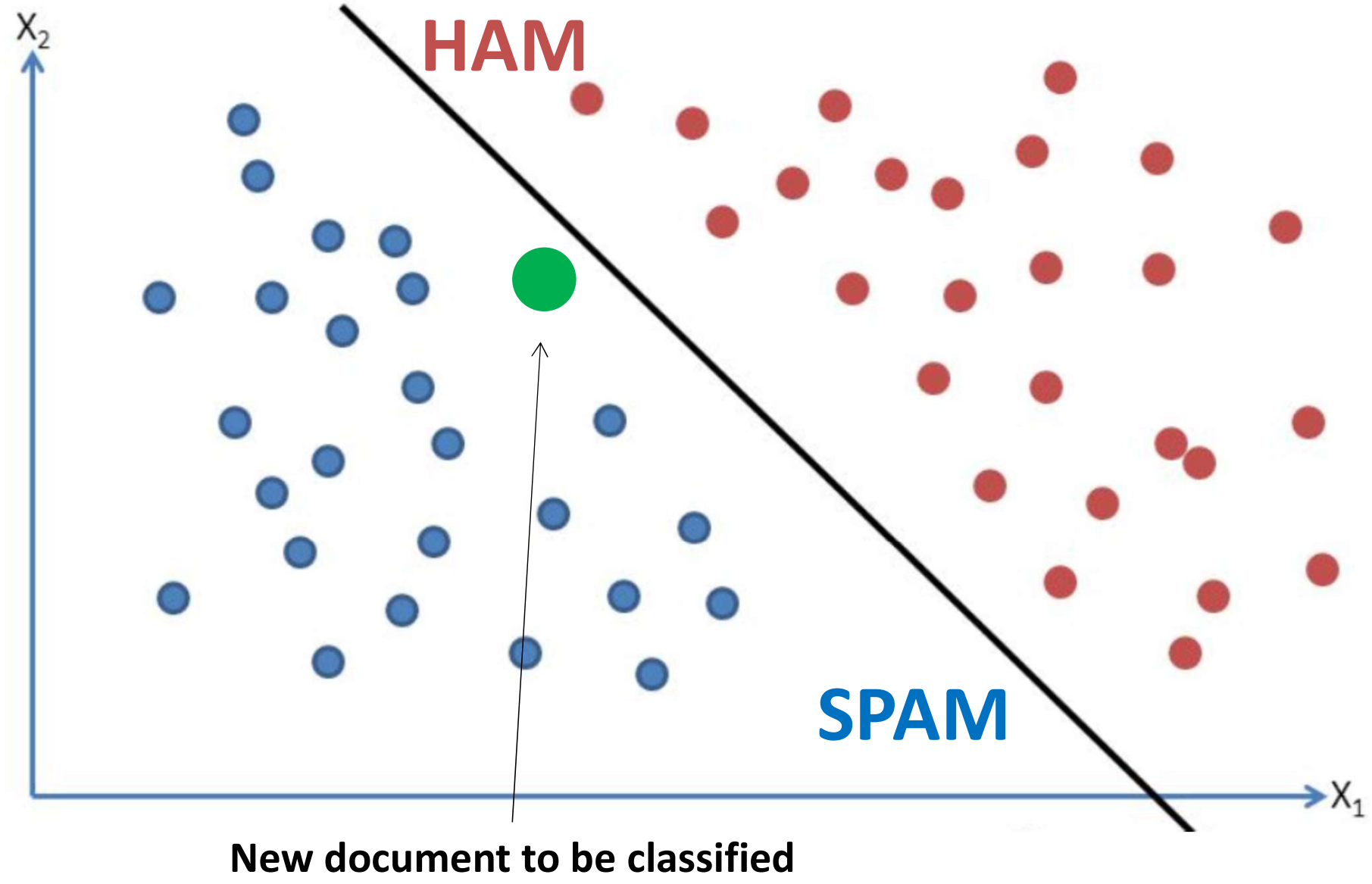
Source: https://www.jmp.com/en_us/statistics-knowledge-portal/what-is-multiple-regression/fitting-multiple-regression-model.html

Linear Separator

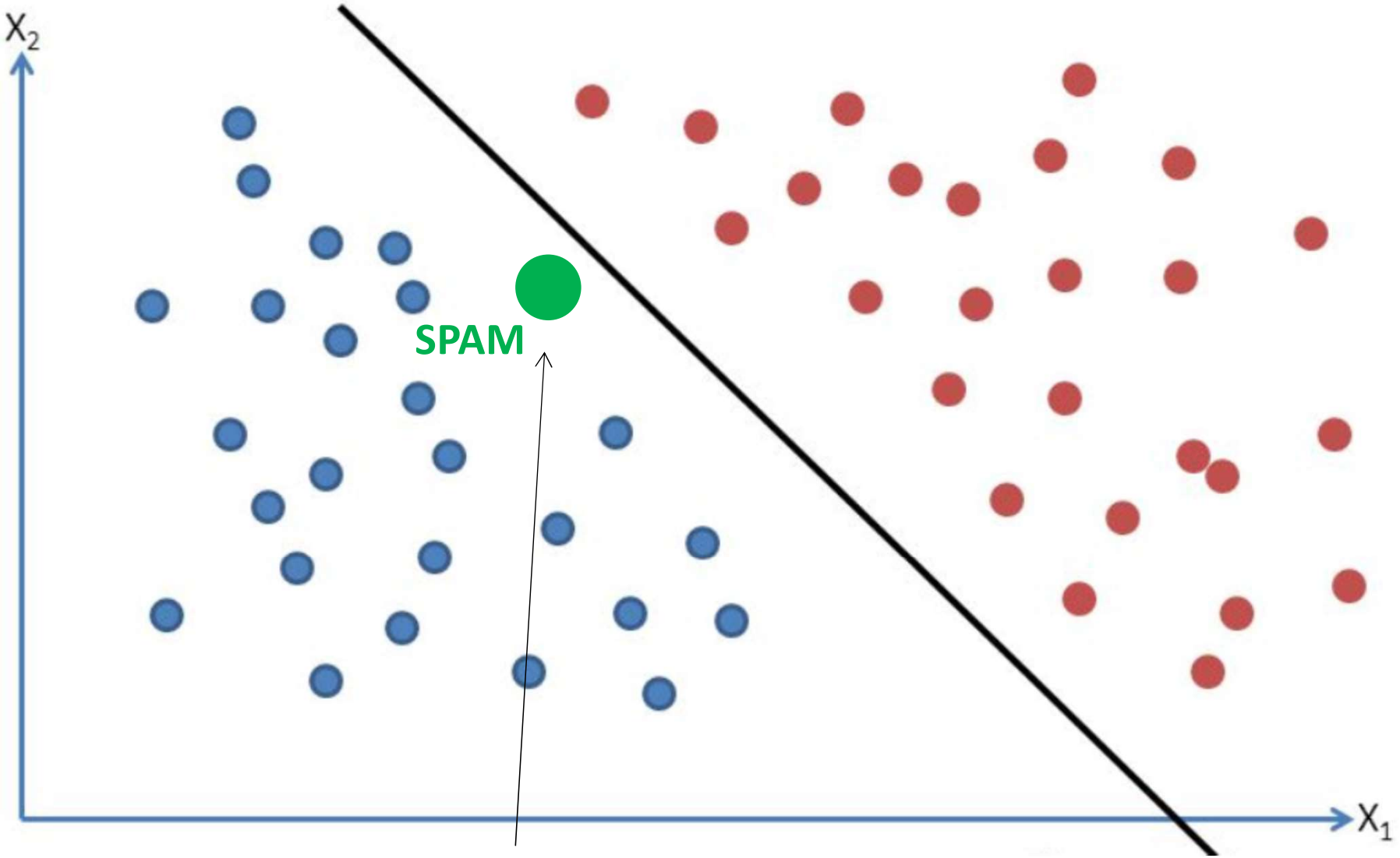


Line in 2D, plane in 3D, hyperplane in ND

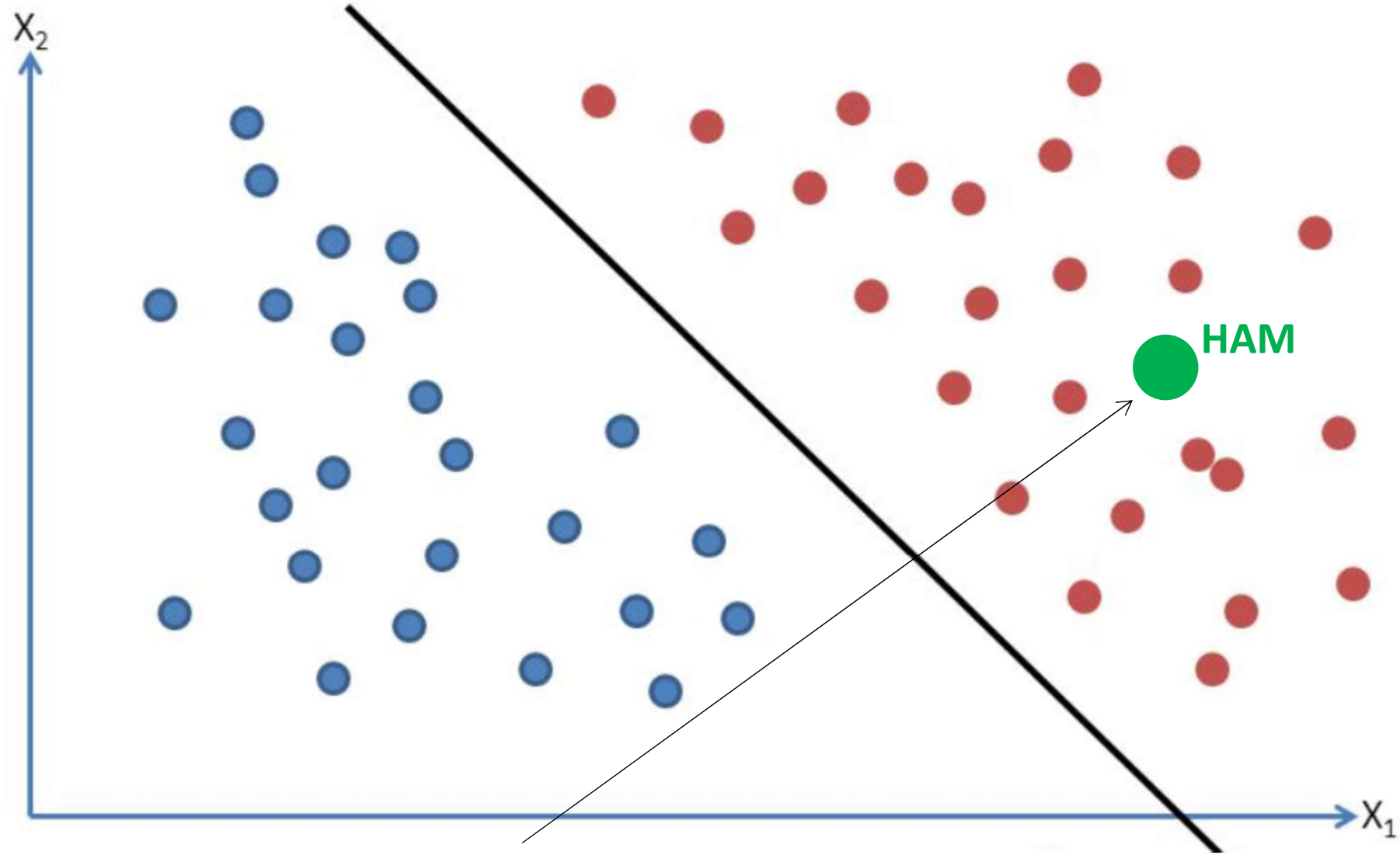
Text Classification: Separator



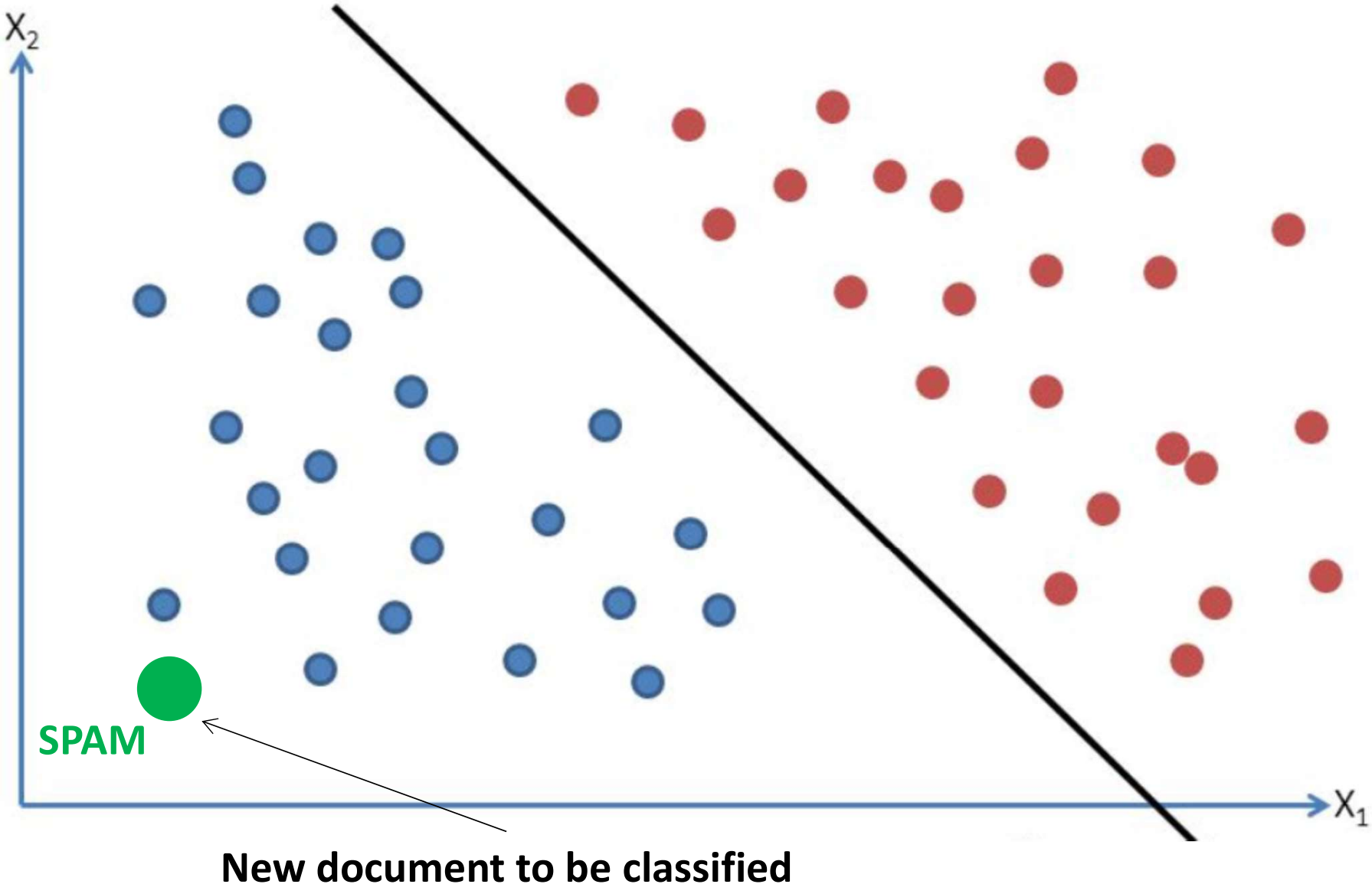
Text Classification: Separator



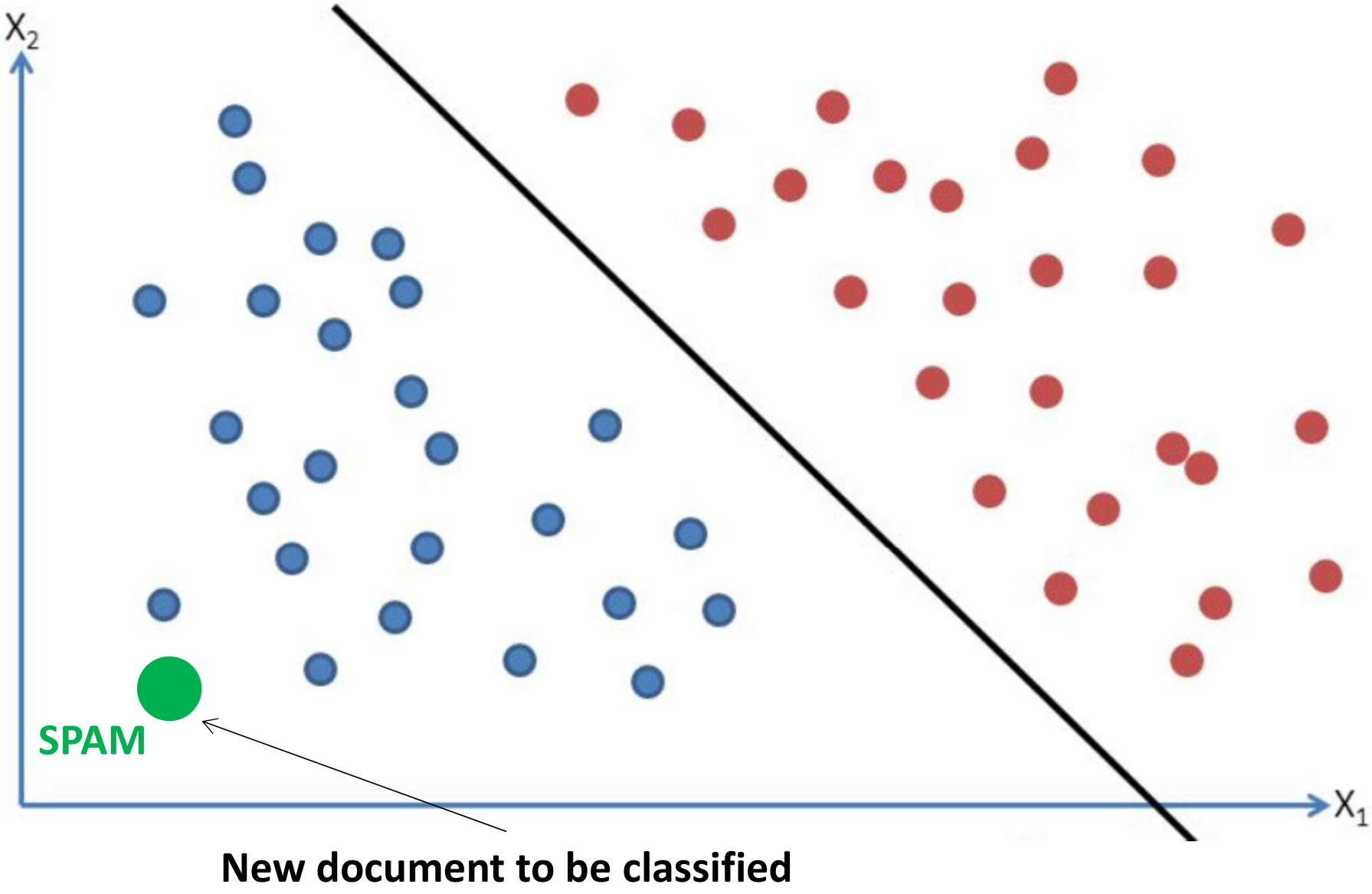
Text Classification: Separator



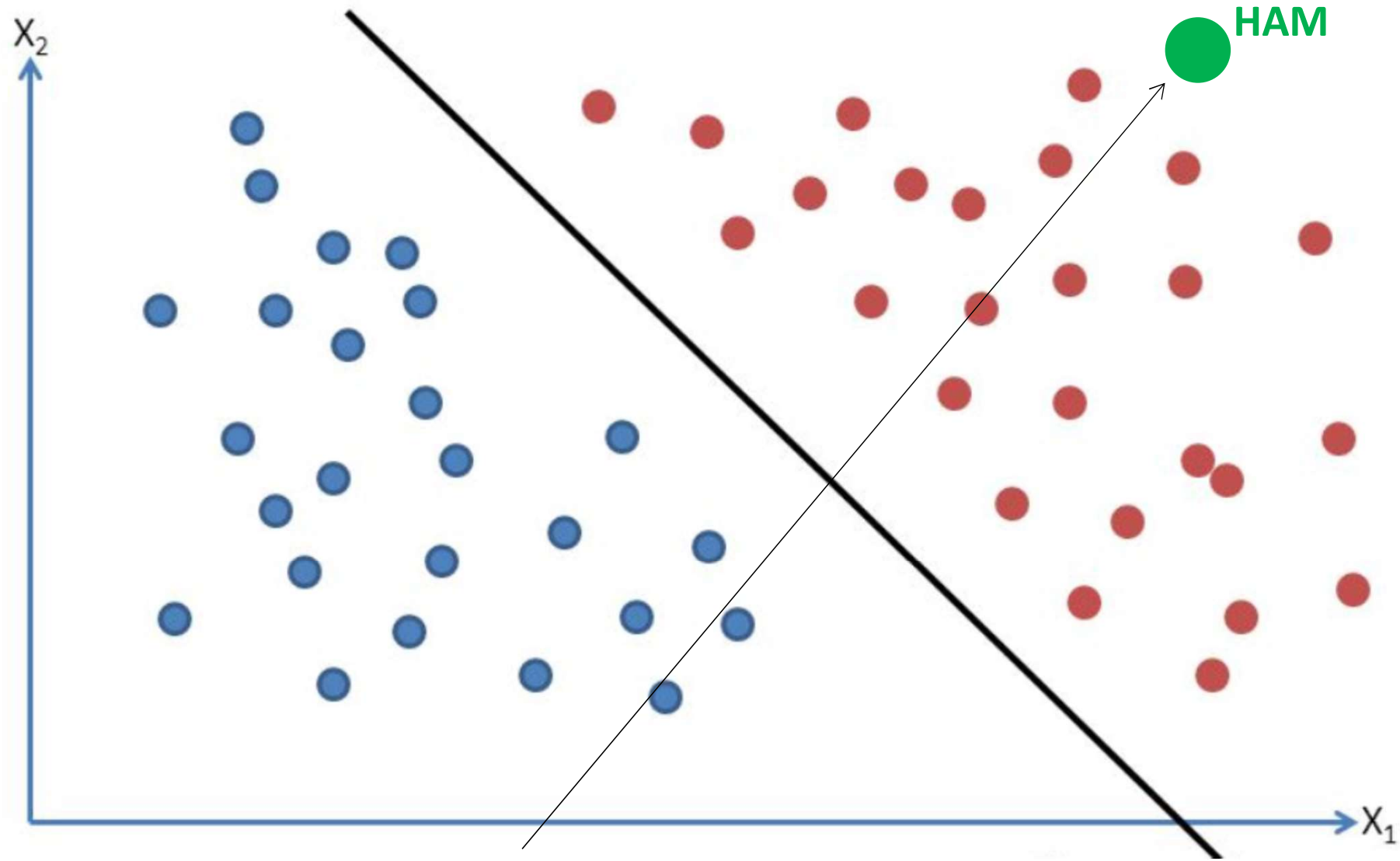
Text Classification: Separator



Text Classification: Separator



Text Classification: Separator



Vector Dot / Scalar Product

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

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

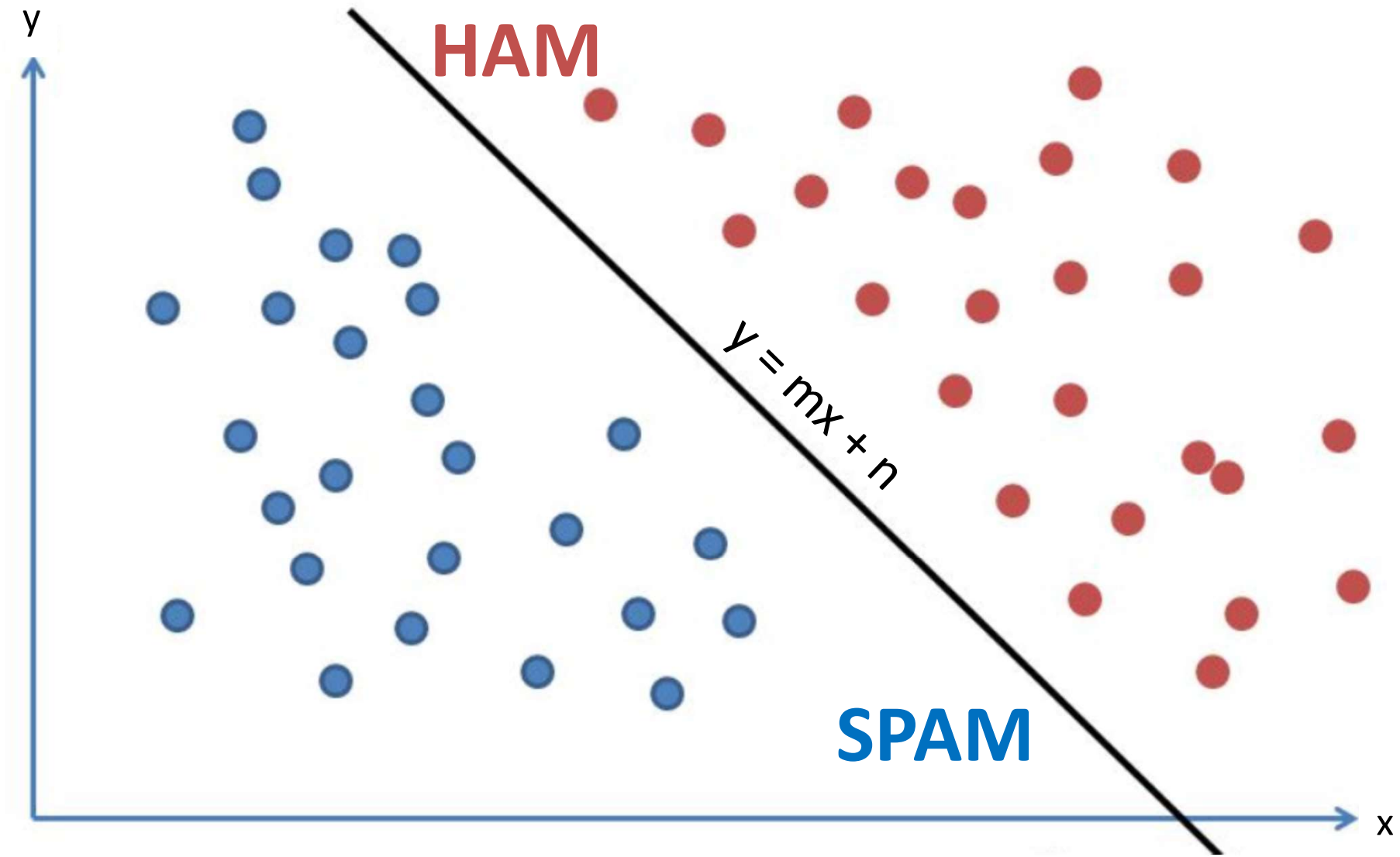
their vector dot/scalar product is:

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

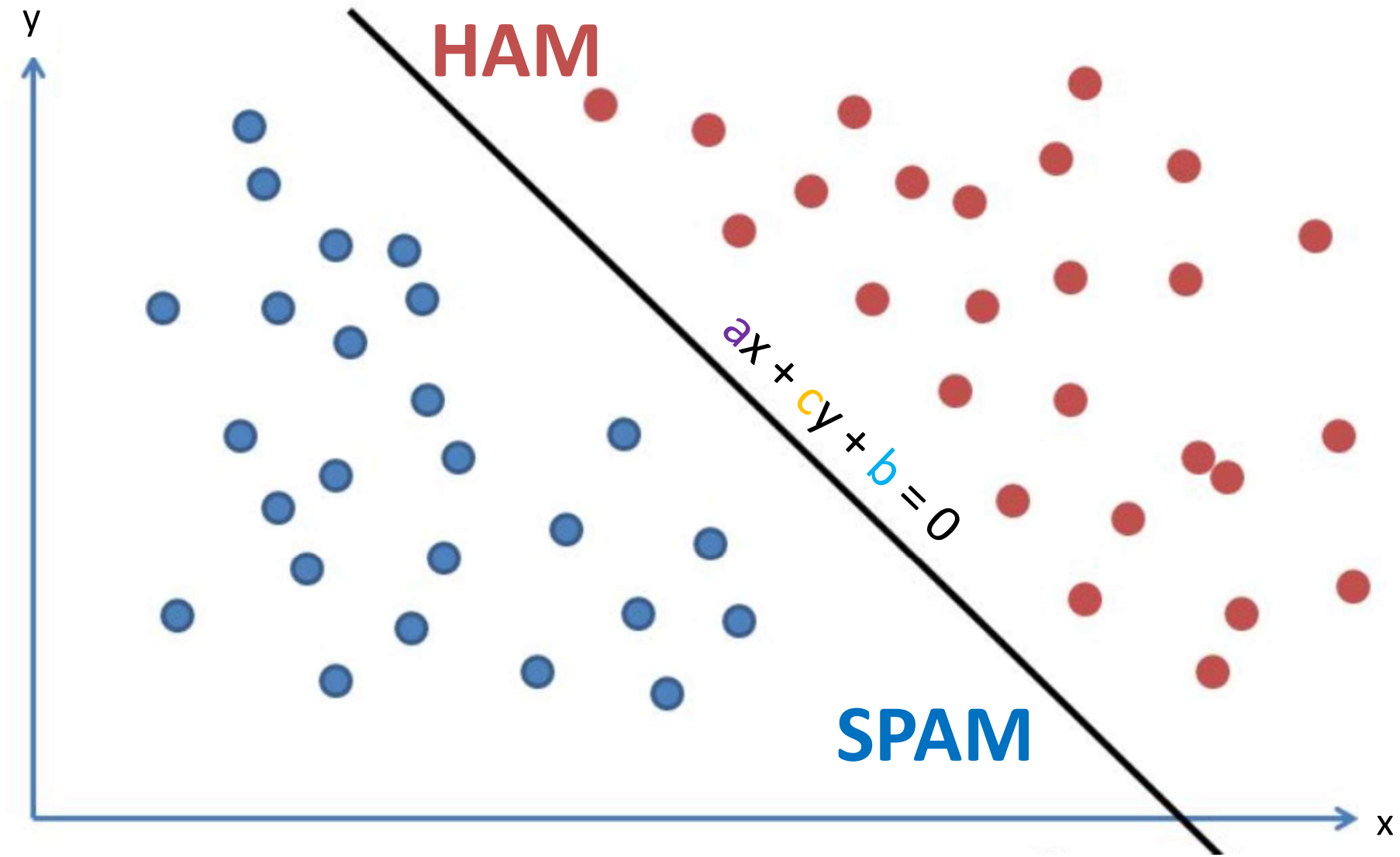
Using matrix representation:

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

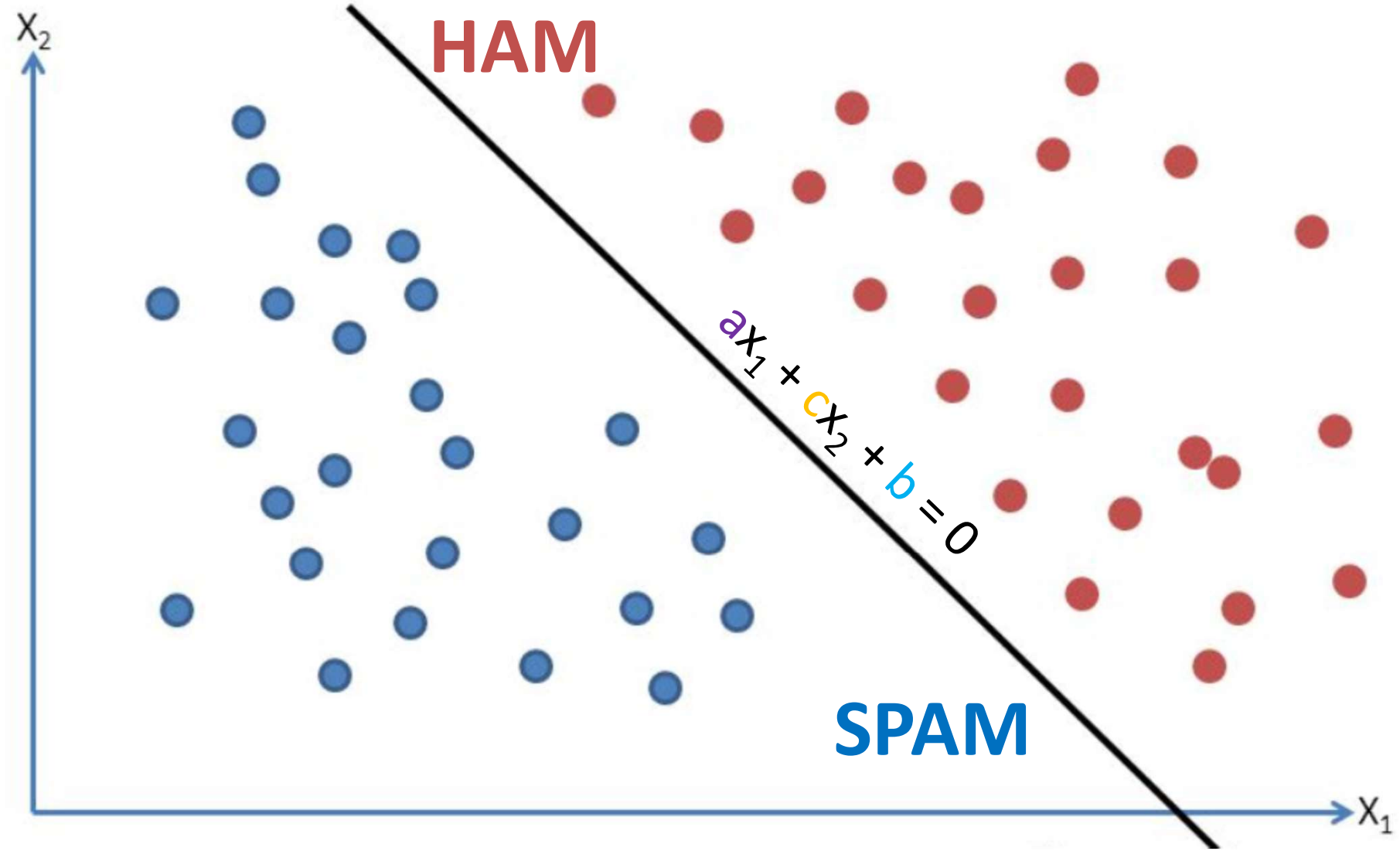
Text Classification: Separator



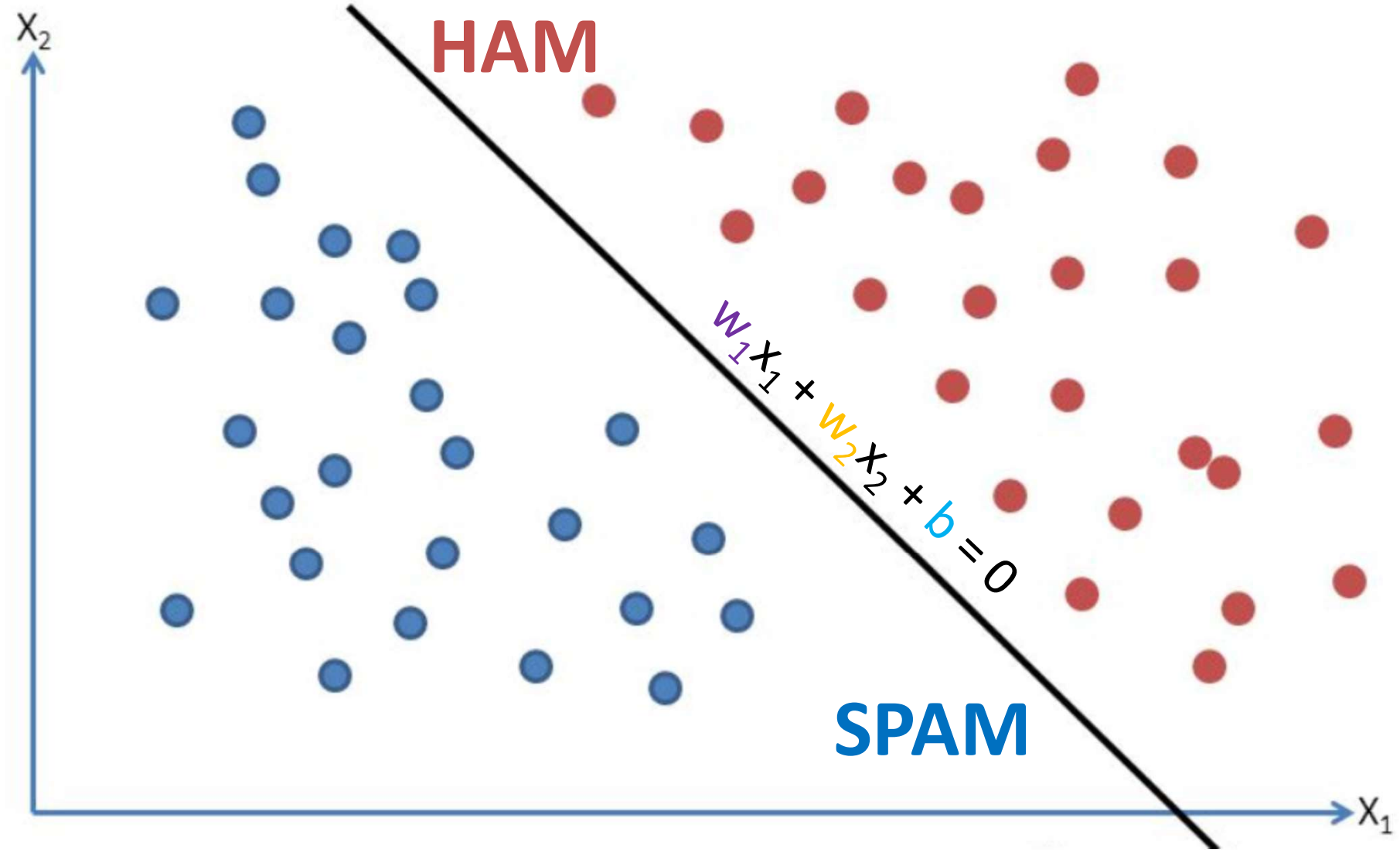
Text Classification: Separator



Text Classification: Separator



Text Classification: Separator



Vector Dot / Scalar Product

Given two vectors $\boldsymbol{w}$ and $\boldsymbol{x}$ (N - vector space dimension):

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

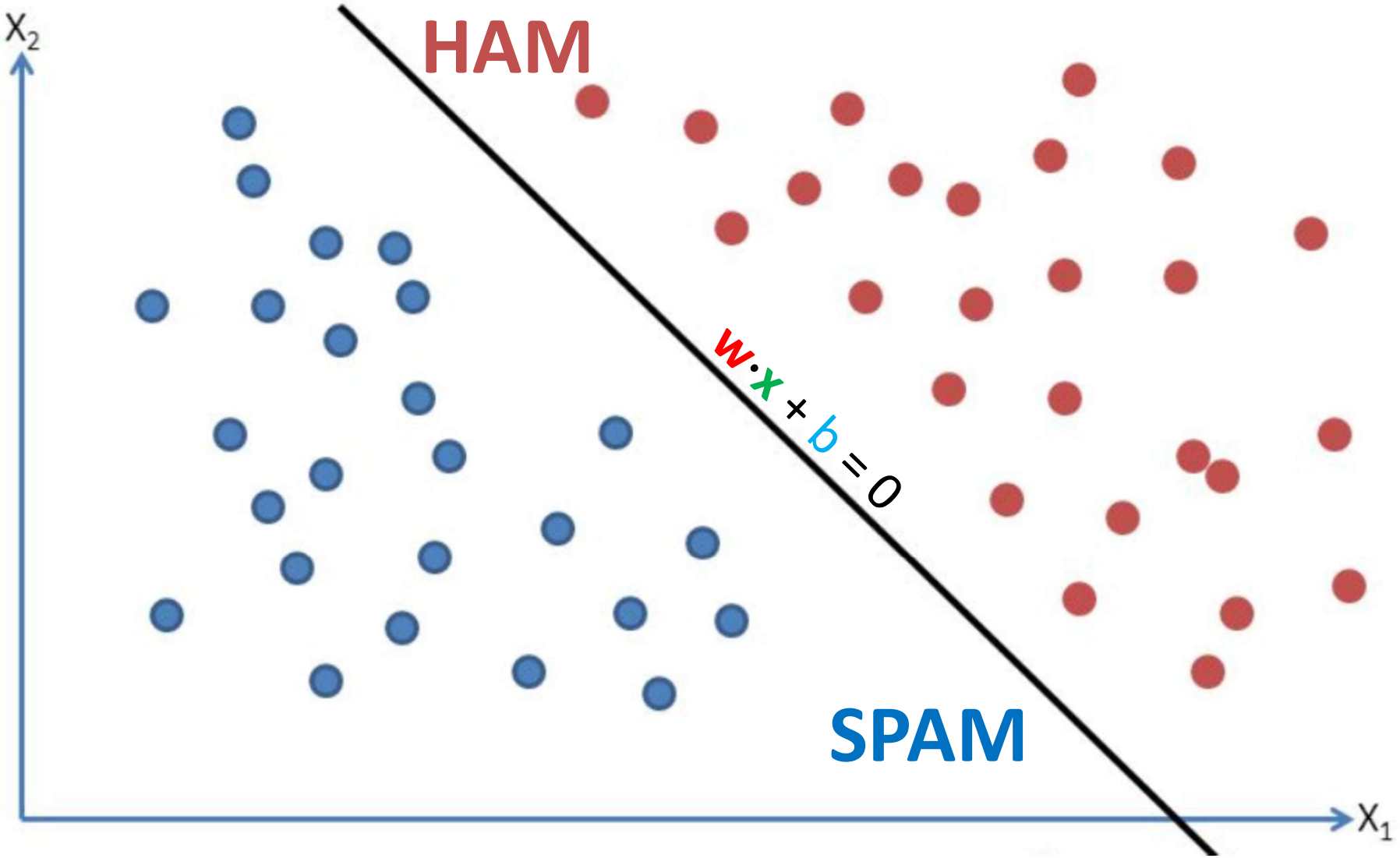
their vector dot/scalar product is:

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

Using matrix representation:

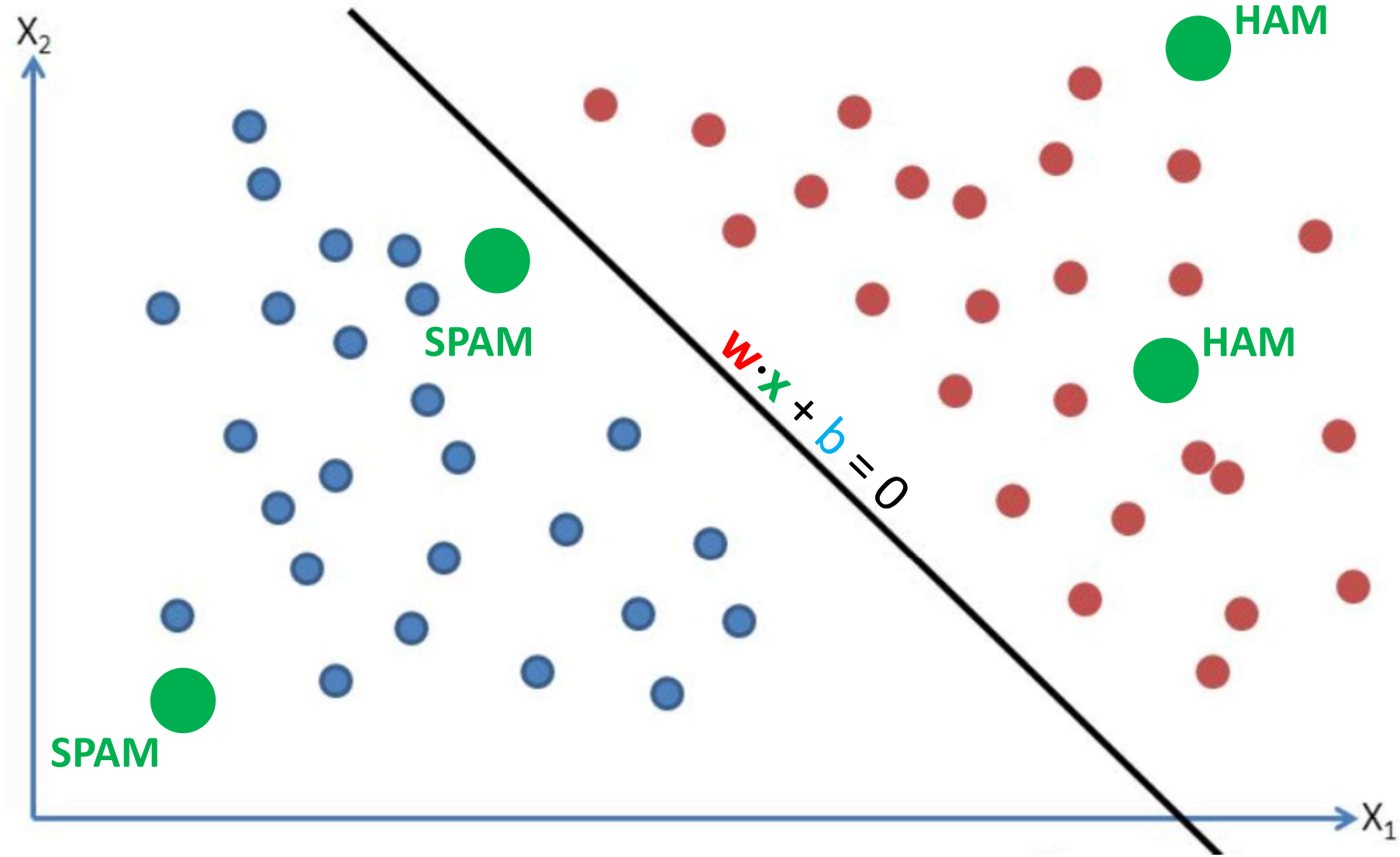
$$\boldsymbol{w} \cdot \boldsymbol{x} = \boldsymbol{w}\boldsymbol{x}^T = [w_1 \quad w_2 \quad \cdots \quad w_N] \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix}$$

Text Classification: Separator

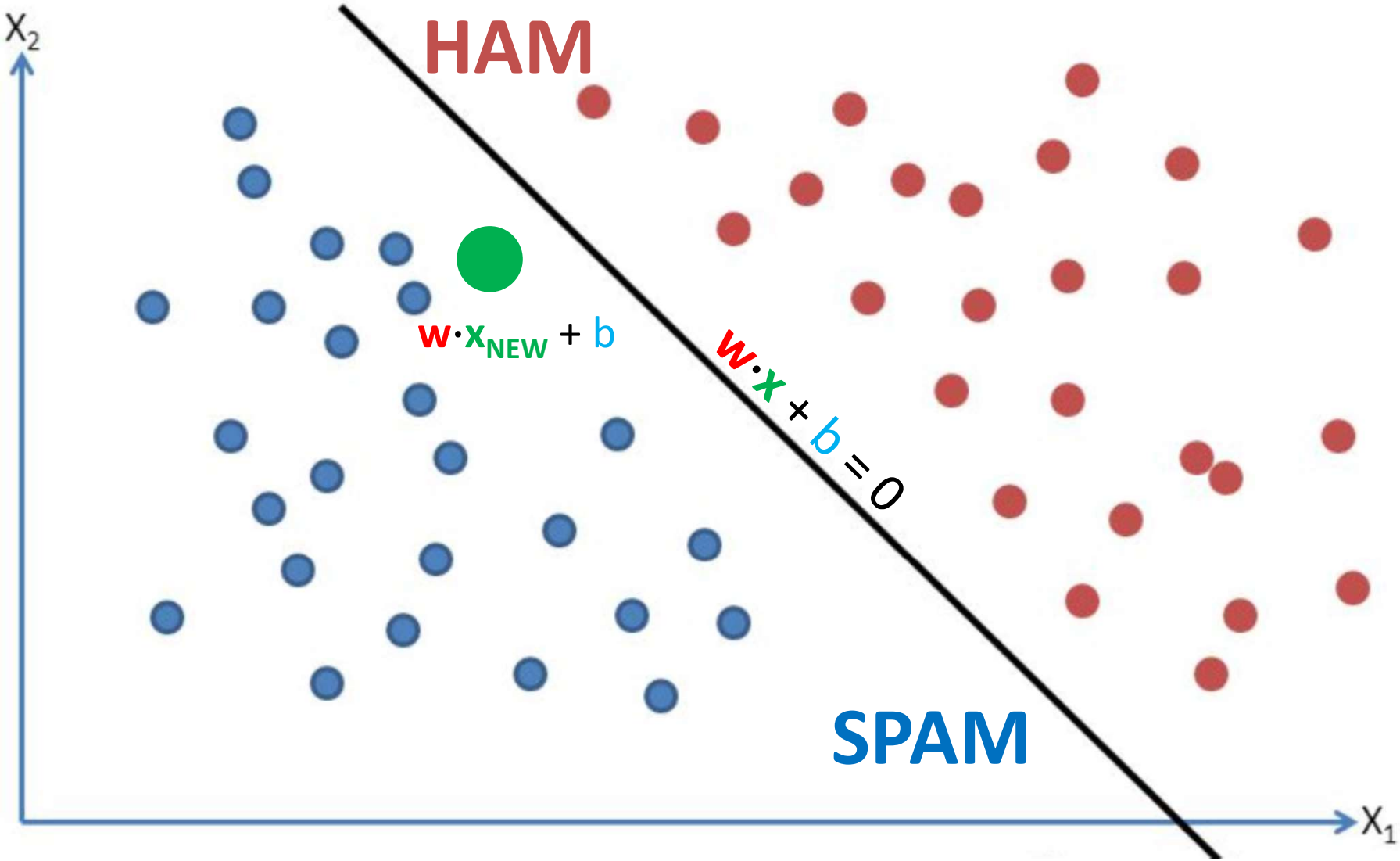


where: $\boldsymbol{w} = [w_1, w_2]$ and $\boldsymbol{x} = [x_1, x_2]$

Text Classification: Separator

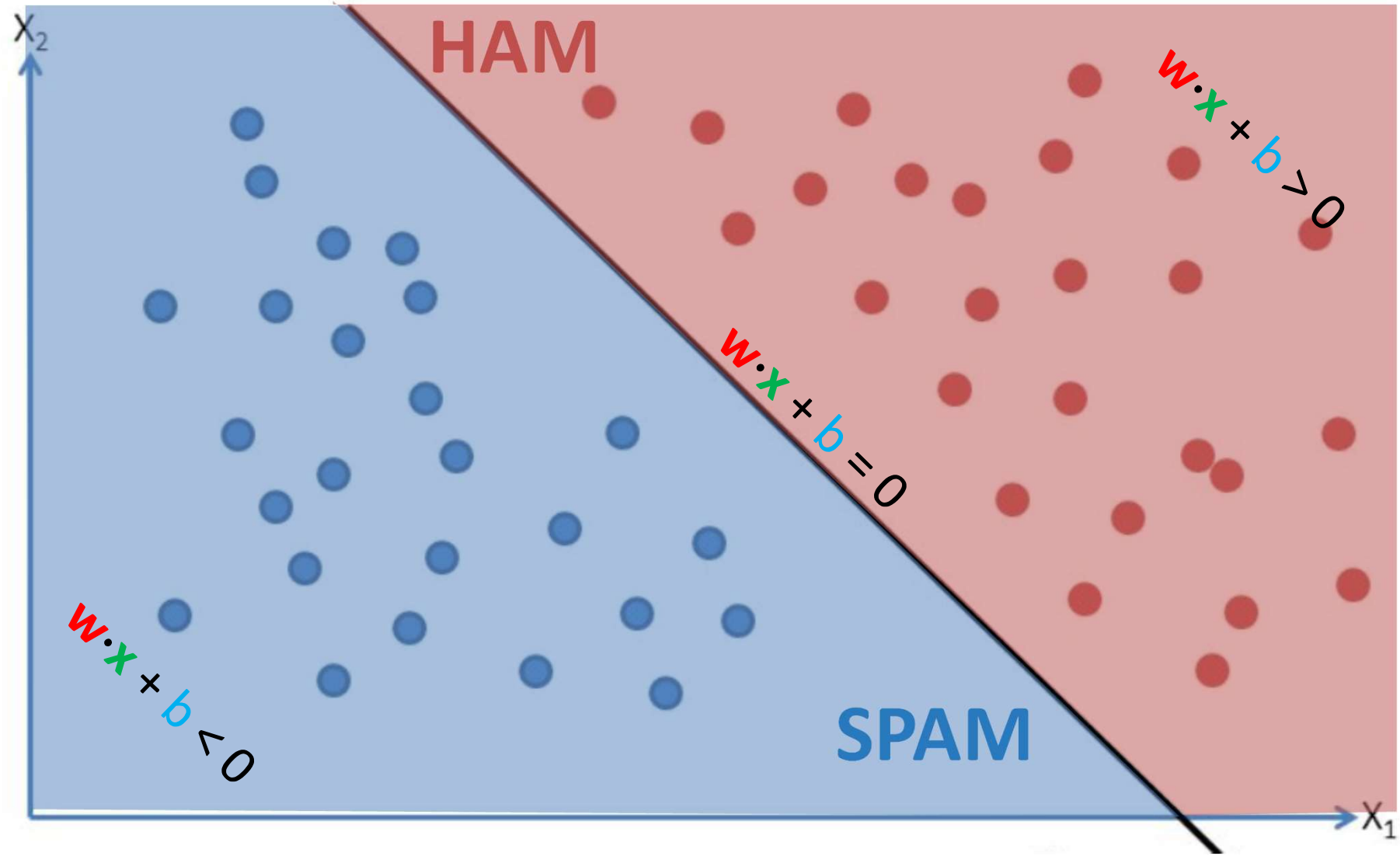


Text Classification: Separator



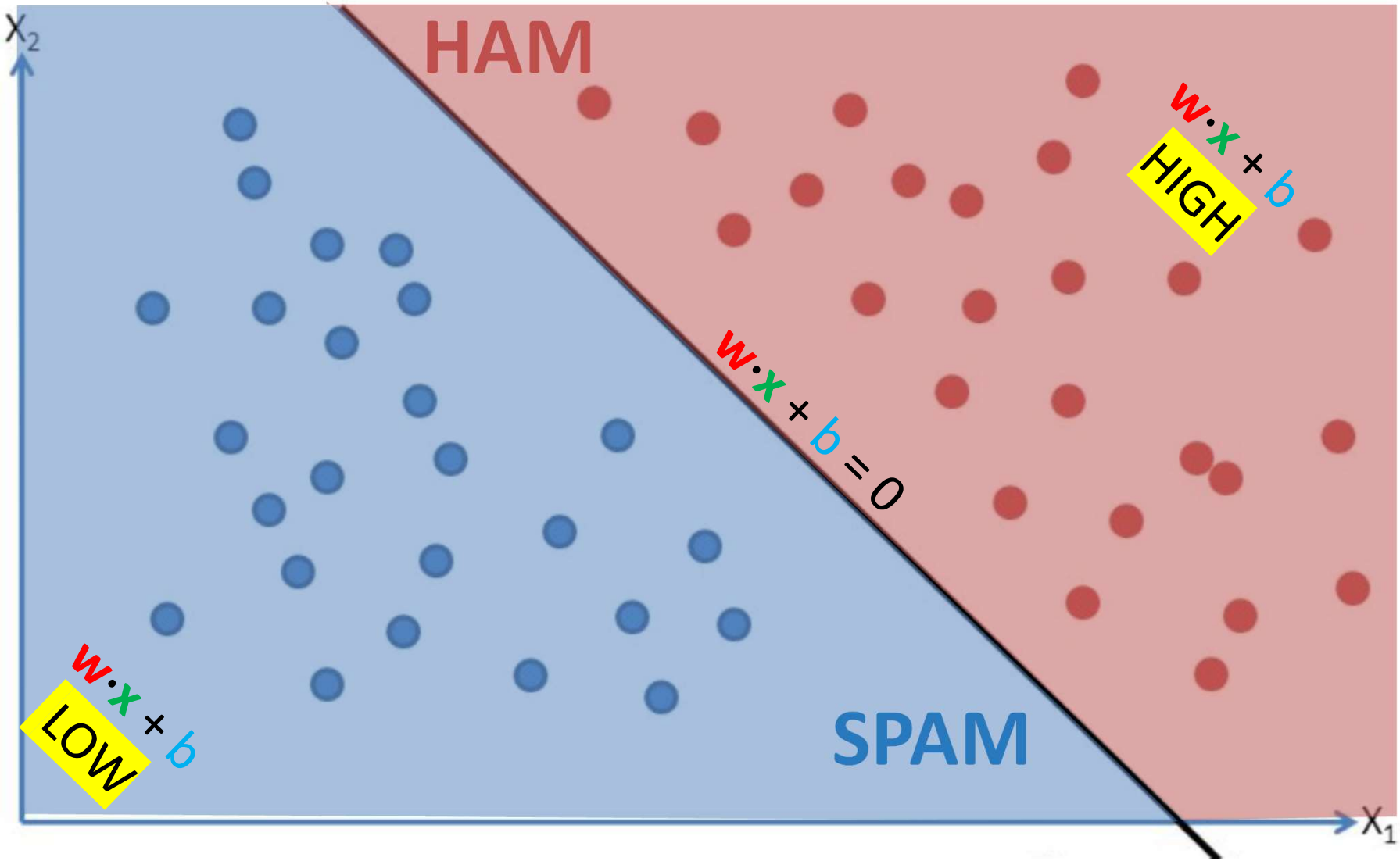
where: $w = [w_1, w_2]$ and $x = [x_1, x_2]$

Text Classification: Separator



where: $\boldsymbol{w} = [w_1, w_2]$ and $\boldsymbol{x} = [x_1, x_2]$

Text Classification: Separator



where: $\boldsymbol{w} = [w_1, w_2]$ and $\boldsymbol{x} = [x_1, x_2]$

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

$$[x_1, x_2, \ldots, x_N]$$

We want to compute a predicted class

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

Logistic Regression Classification

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

$$\mathbf{x} = [x_1, x_2, \ldots, x_N]$$

Weights vector $\mathbf{w}$:

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

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.

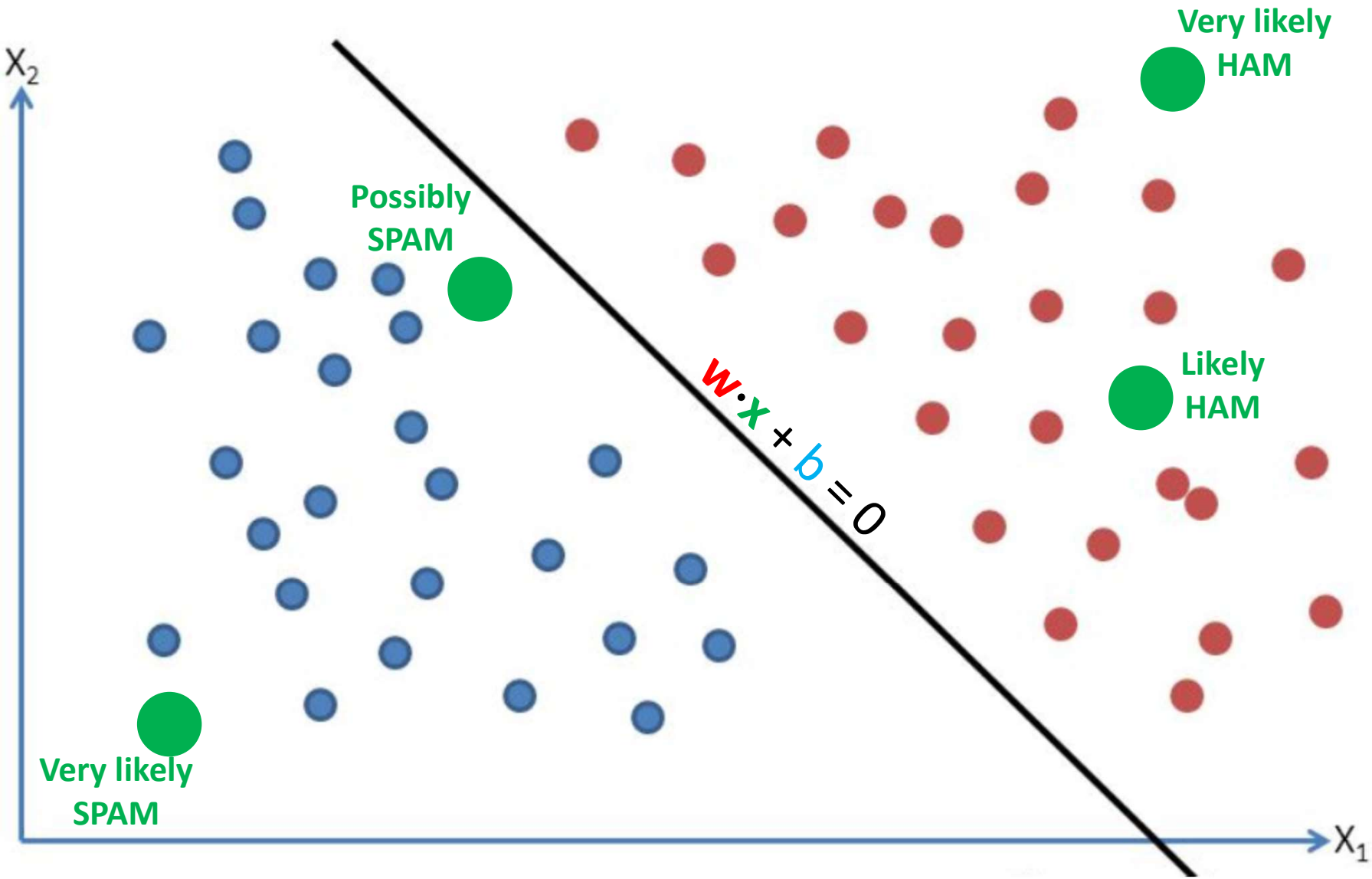
We'd like a principled classifier that gives us a probability, just like Naive Bayes did

We want a model that can tell us:

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

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

Text Classification: Separator



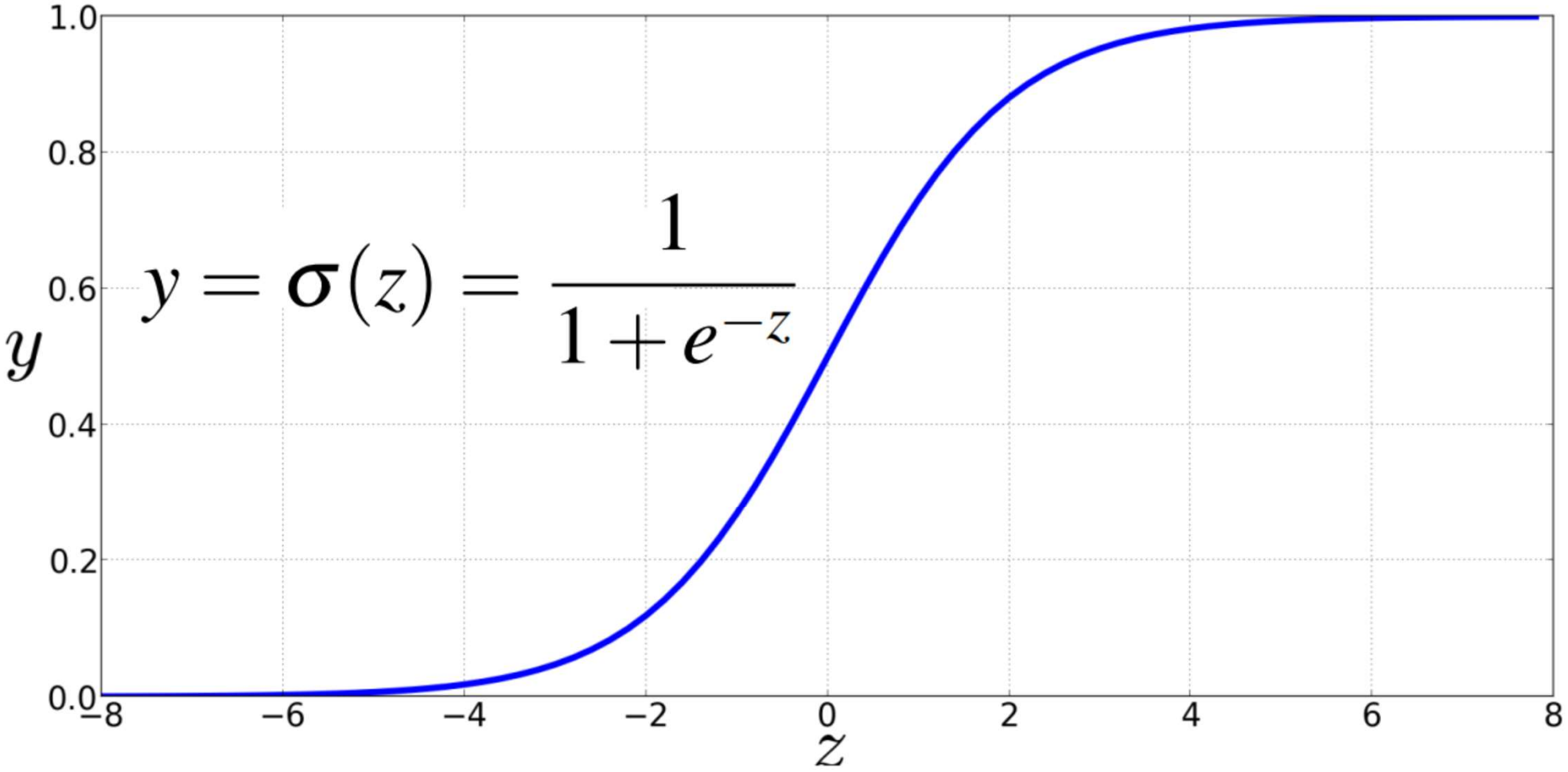
Logistic Regression: The Idea

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

$$\sigma(\mathbf{w} \cdot \mathbf{x} + b)$$

- 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...

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

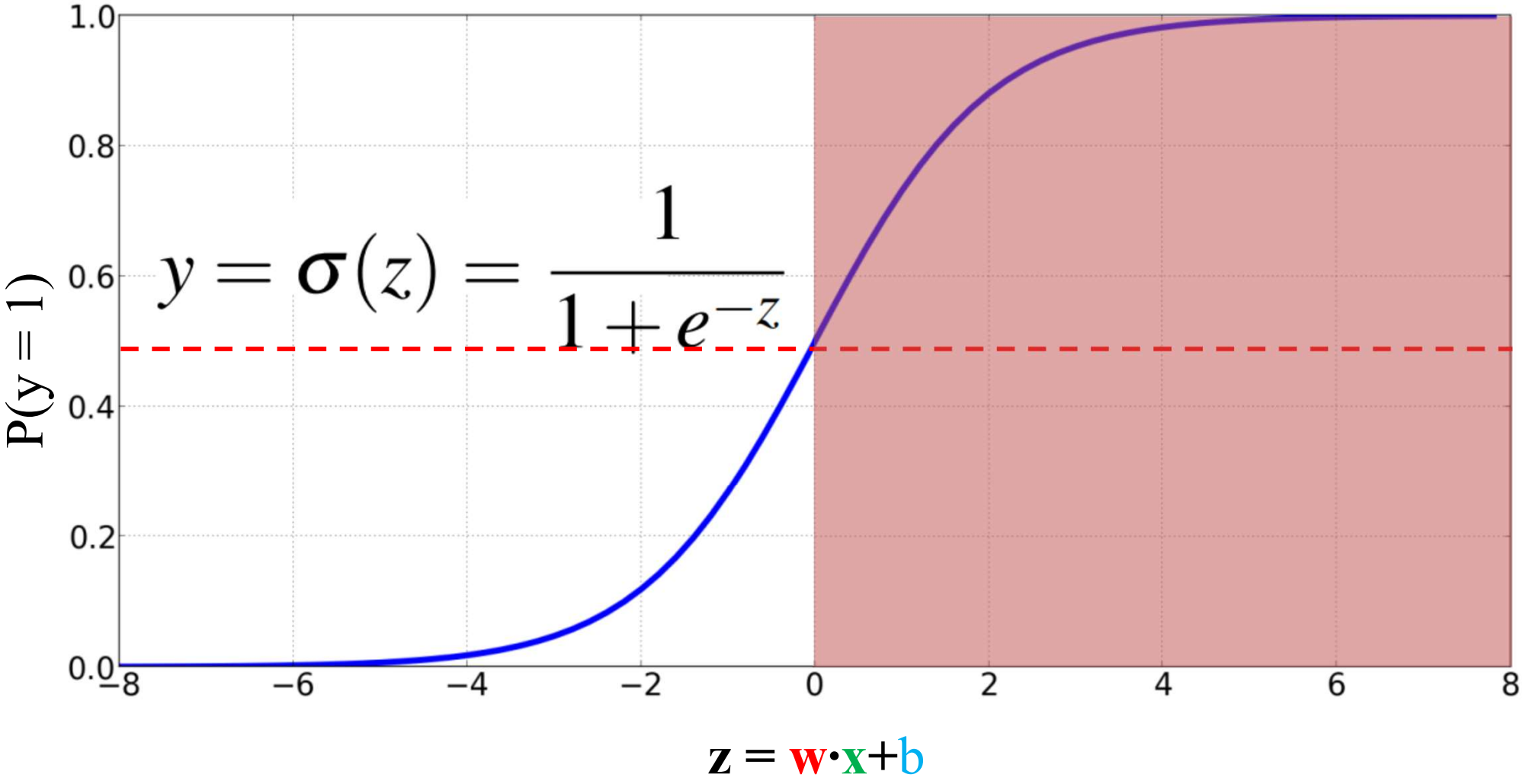
Probabilities → Classification

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

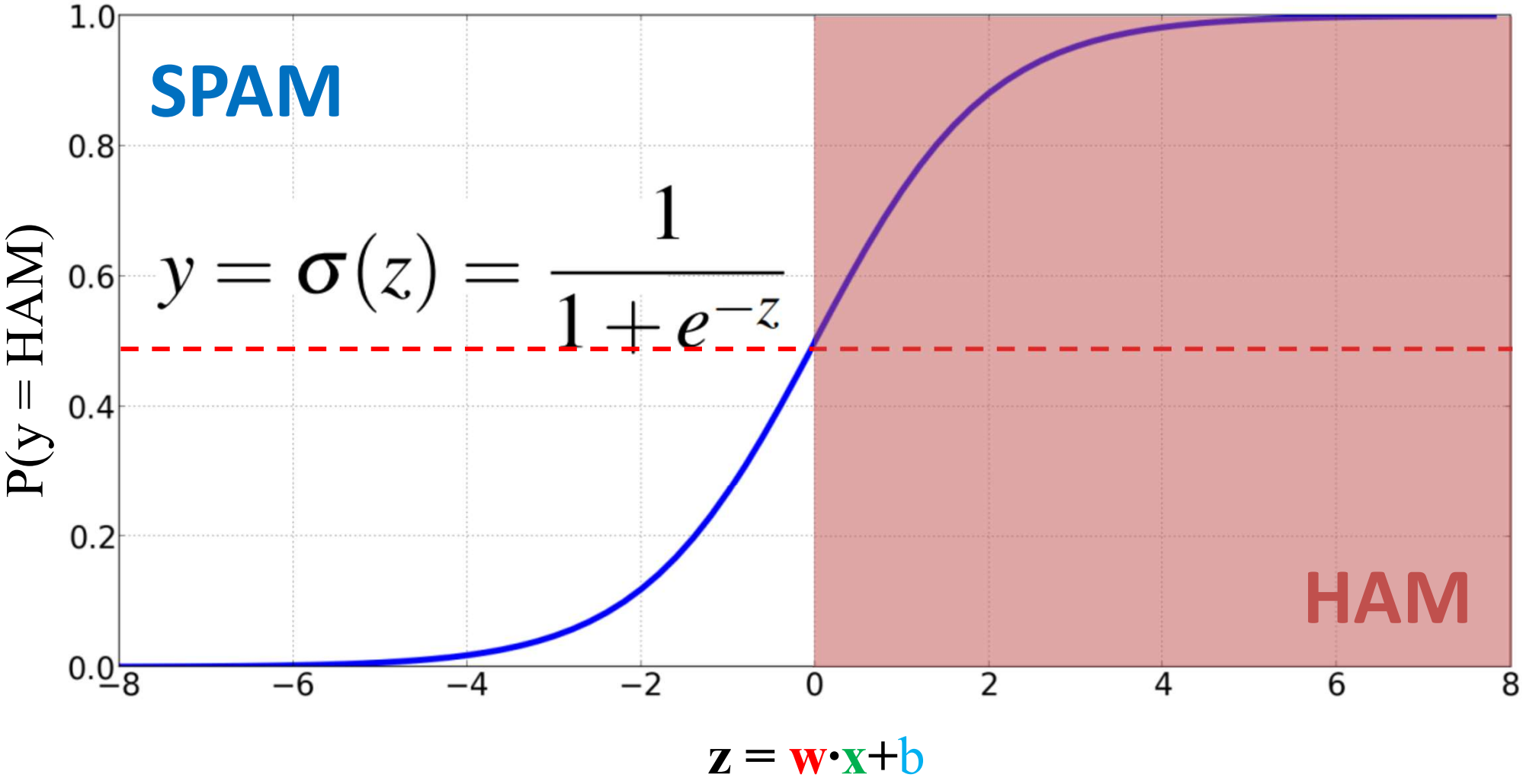
$$\hat{y} = \begin{cases} 1 & \text{if } P(y=1|x) > 0.5 \\ 0 & \text{otherwise} \end{cases}$$

Where 0.5 is the decision boundary

Logistic Regression Classifier



Logistic Regression Classifier



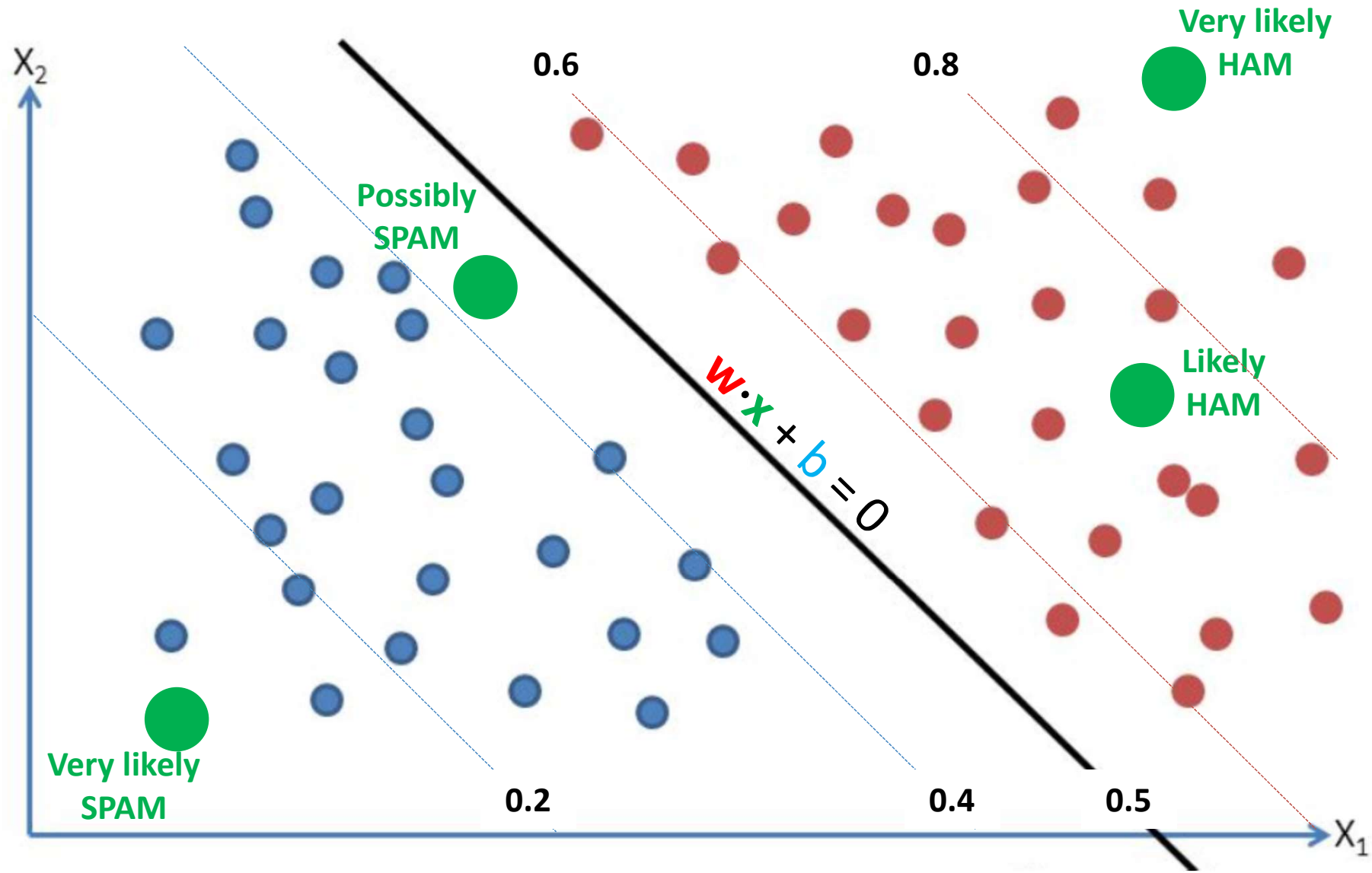
Probabilities → Classification

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

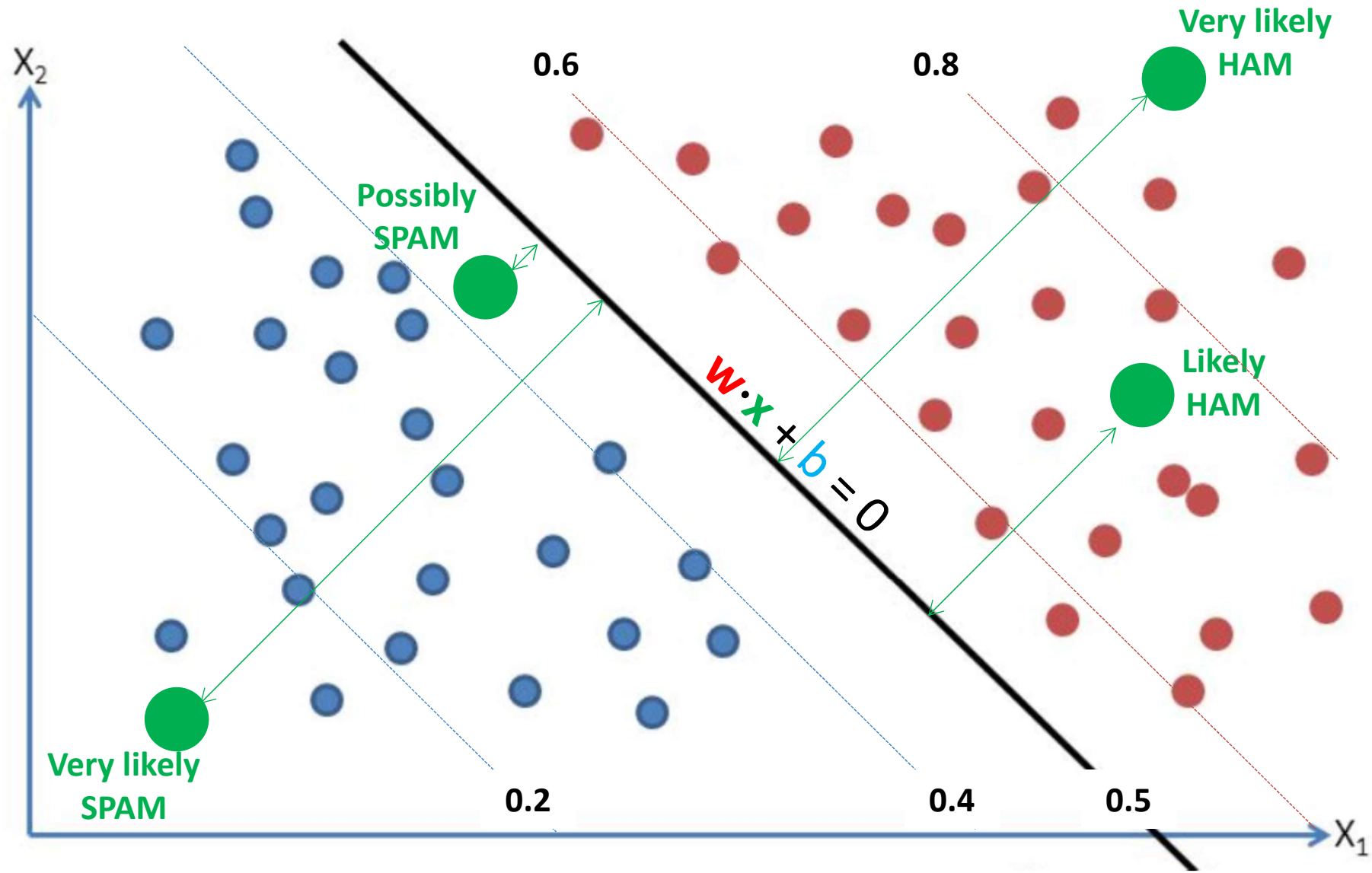
$$\hat{y} = \begin{cases} 1 & \text{if } P(y=1|x) > 0.5 \quad \text{if } w \cdot x + b > 0 \\ 0 & \text{otherwise} \quad \text{if } w \cdot x + b \leq 0 \end{cases}$$

Where 0.5 is the decision boundary

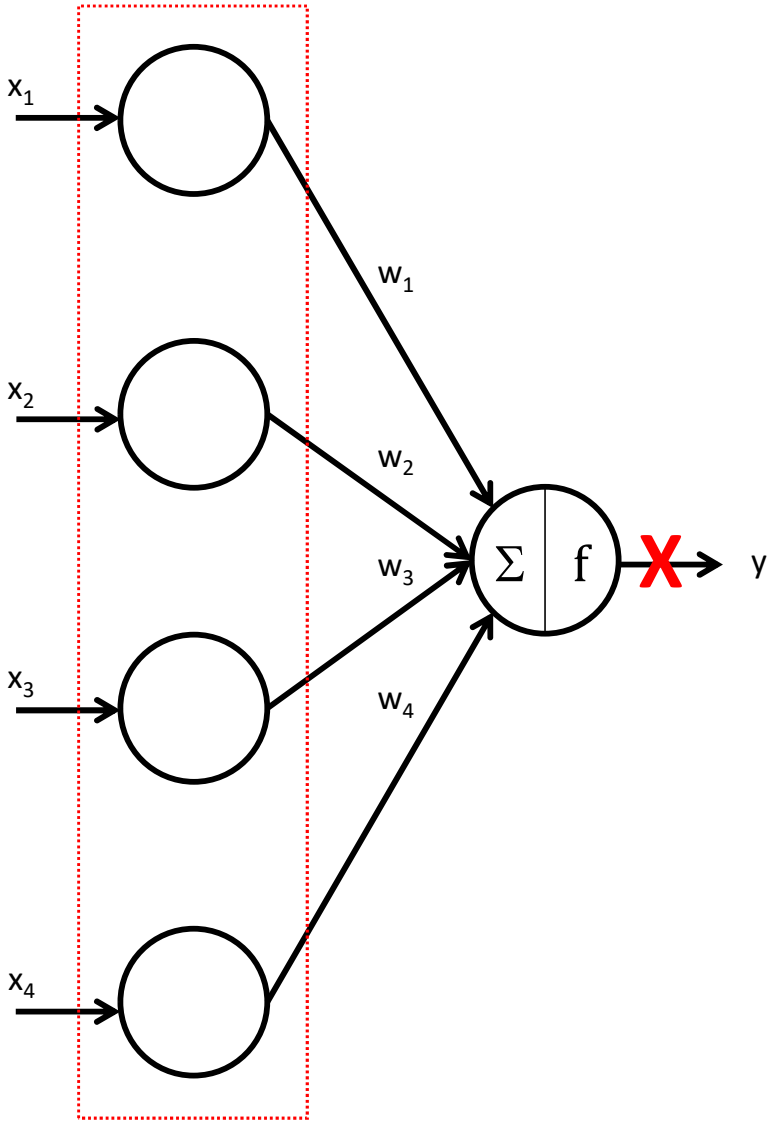
Text Classification: Separator



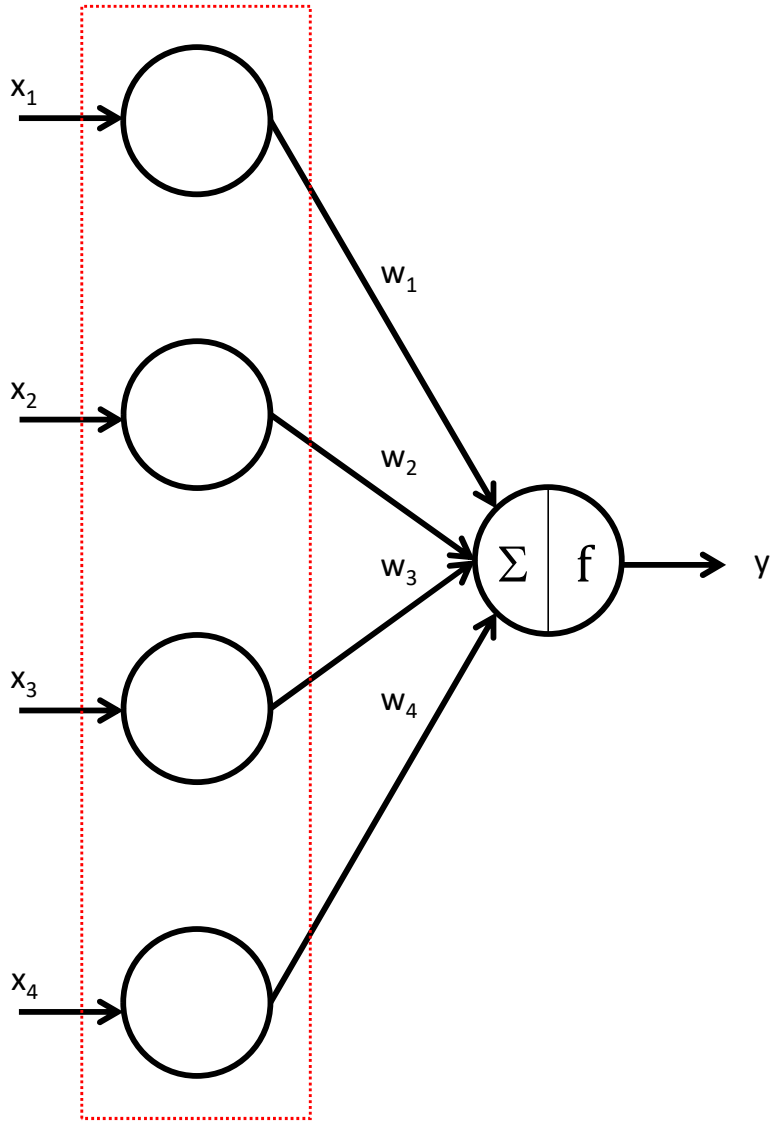
Text Classification: Separator



Does This Resemble Anything?

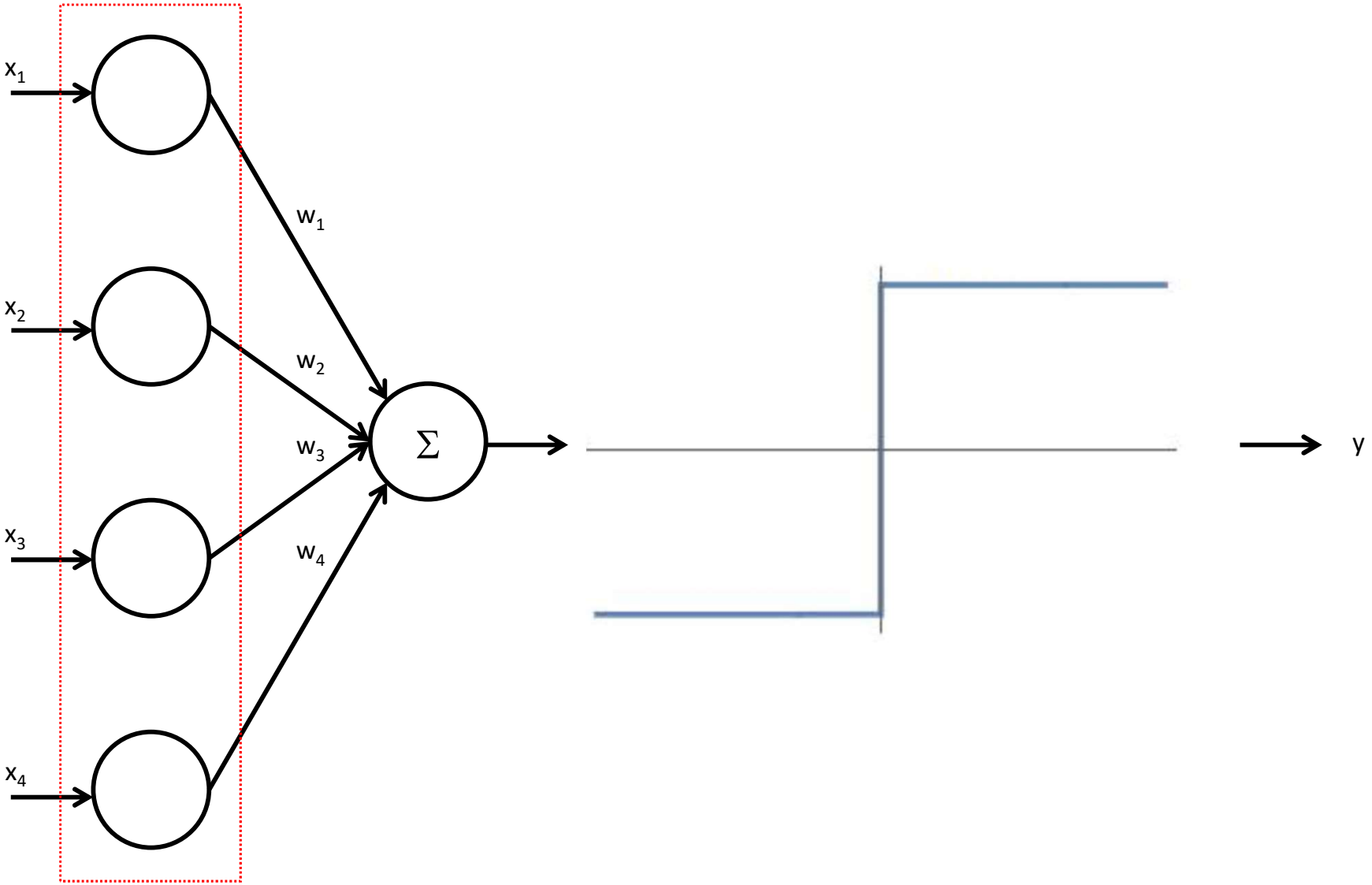


$\Sigma\ w_i * x_i < 0 \rightarrow f = 0 \rightarrow$ **NO**



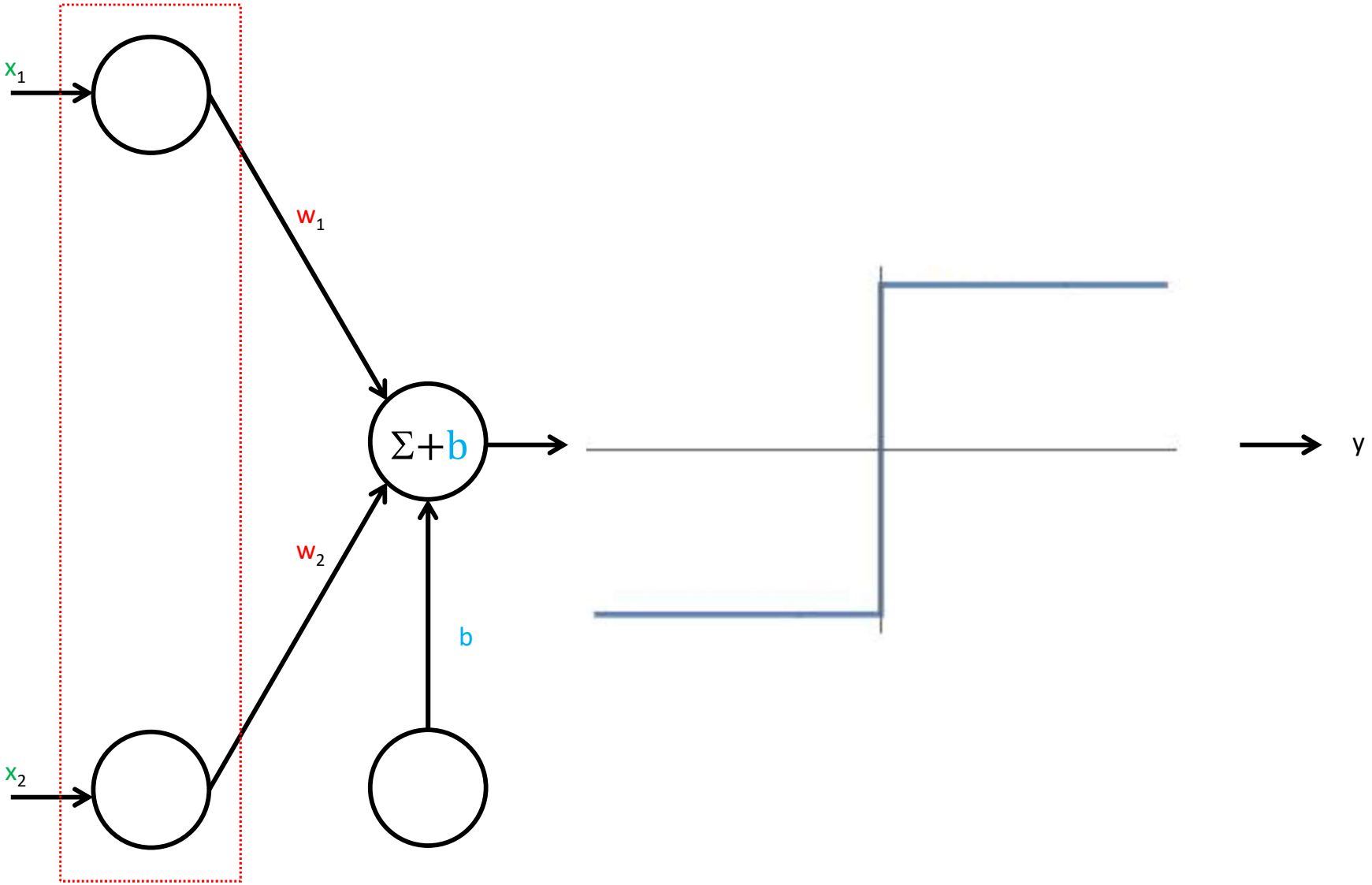
$\Sigma\ w_i * x_i \geq 0 \rightarrow f = 1 \rightarrow$ **YES**

Step Activation Function



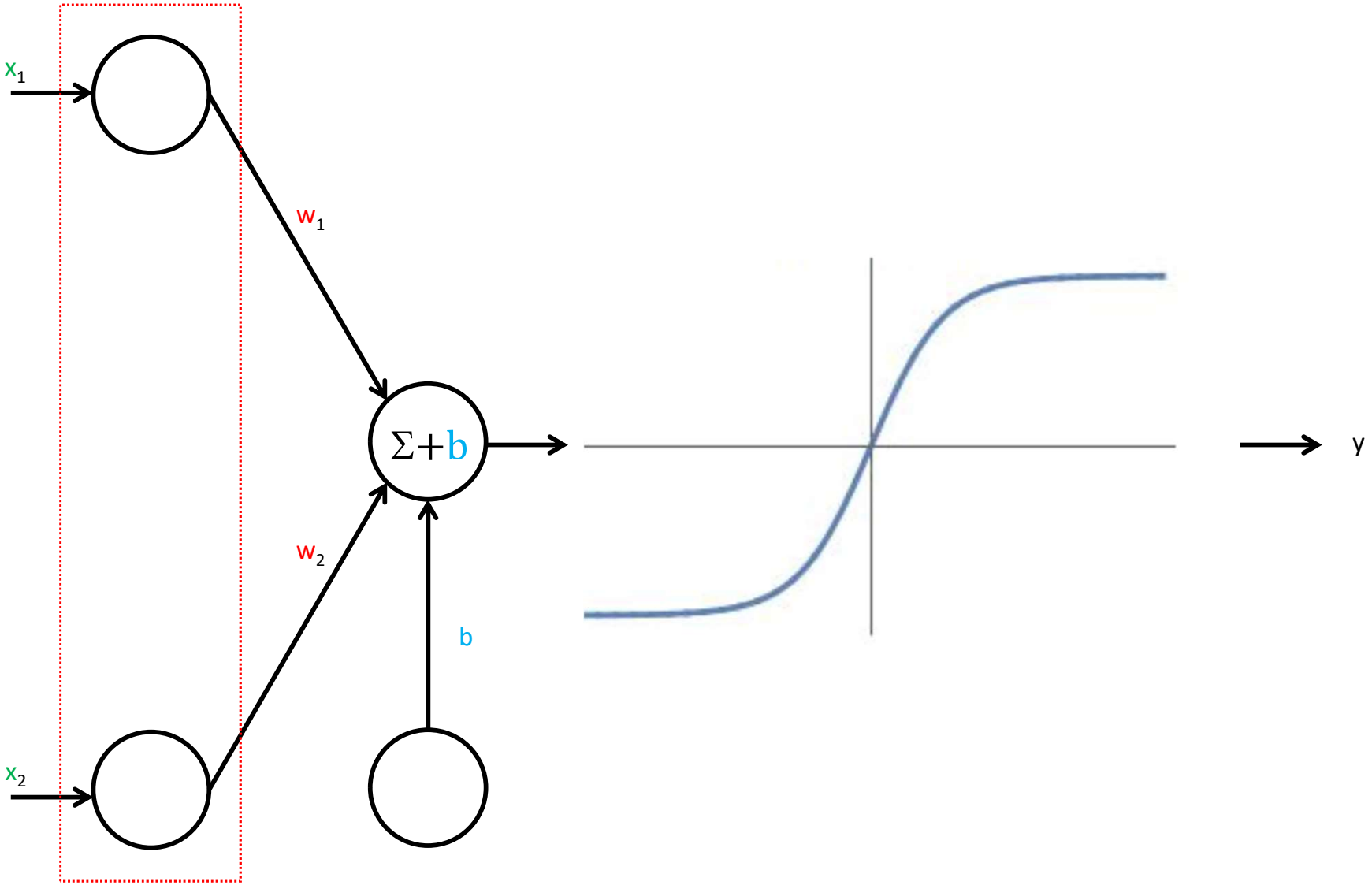
$\Sigma\ w_i * x_i \rightarrow f$

Step Activation Function



$\Sigma\ w_i * x_i + b \rightarrow f \rightarrow y$

Sigmoid Activation Function



$\Sigma\ w_i * x_i + b \rightarrow f \rightarrow y$