CS 481

Artificial Intelligence Language Understanding

March 9, 2023

Announcements / Reminders

- Enjoy your Spring Break!

- Please follow the Week 09 To Do List instructions
- Written Assignment #03 is due on Sunday 03/26/23 at 11:59 PM CST
- Programming Assignment #02 is due on Sunday 04/02/23 at 11:59 PM CST

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

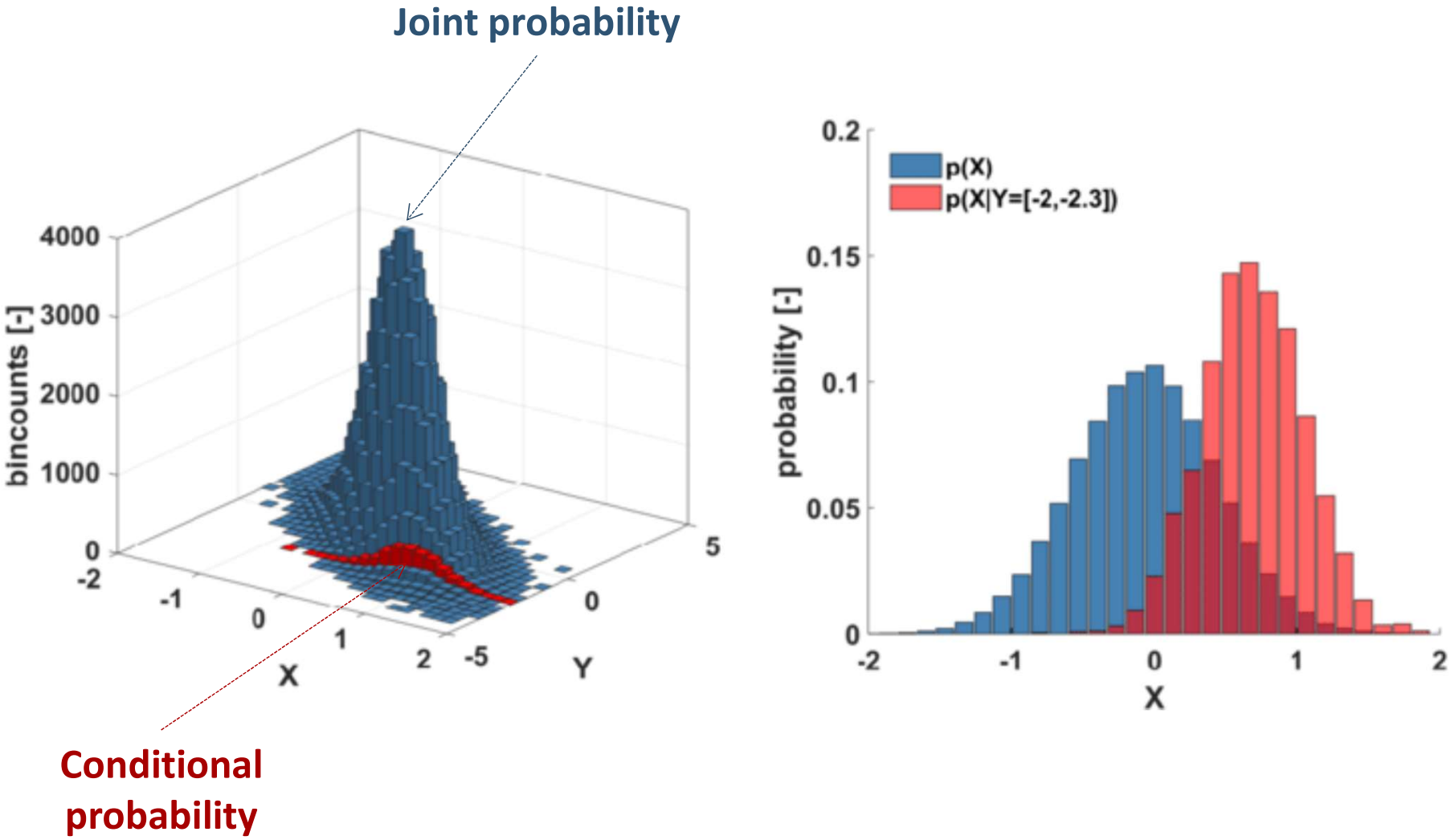
Plan for Today

- **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.

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:

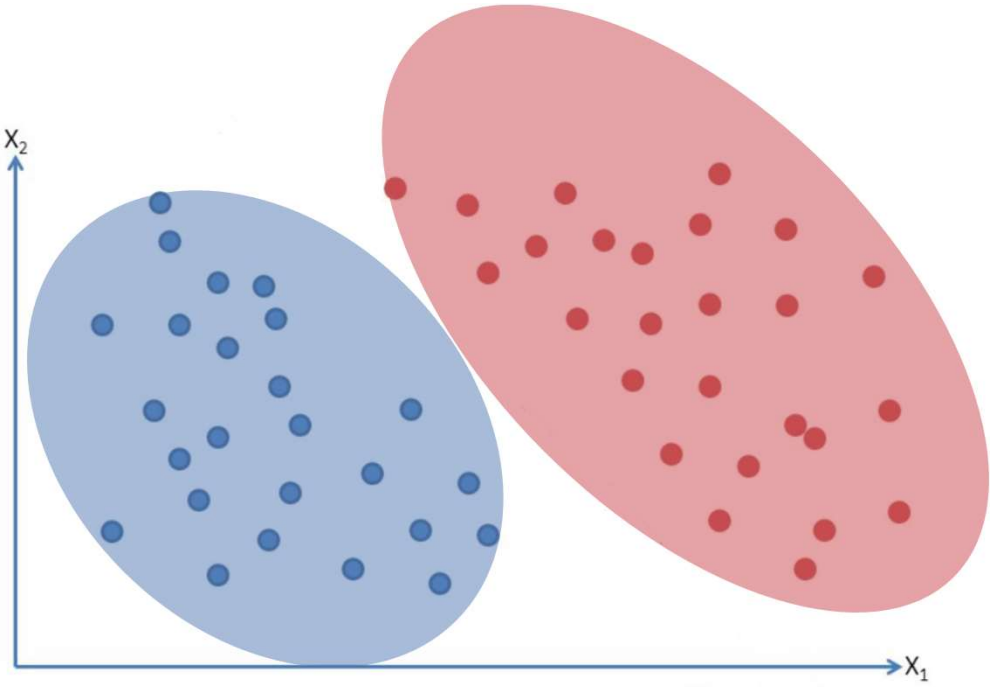
- a document $\boldsymbol{x}$
- a fixed set of classes $Y = \{y_1, y_2, \ldots, y_J\}$
- a training set of N hand-labeled documents $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)$

Output:

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

Generative vs Discriminative Models

Generative

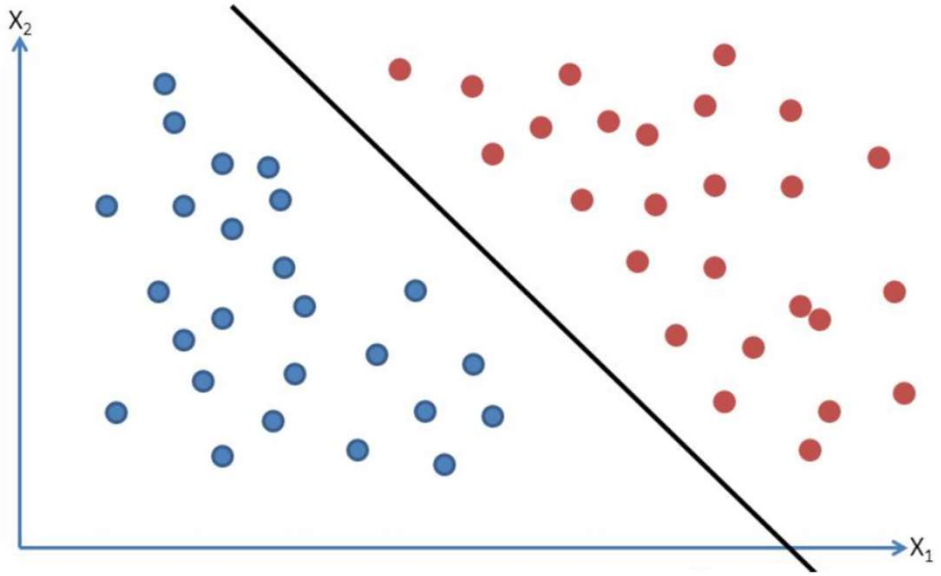


Generative model models actual distributions for EACH CLASS / LABEL / TAG

to

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

Discriminative



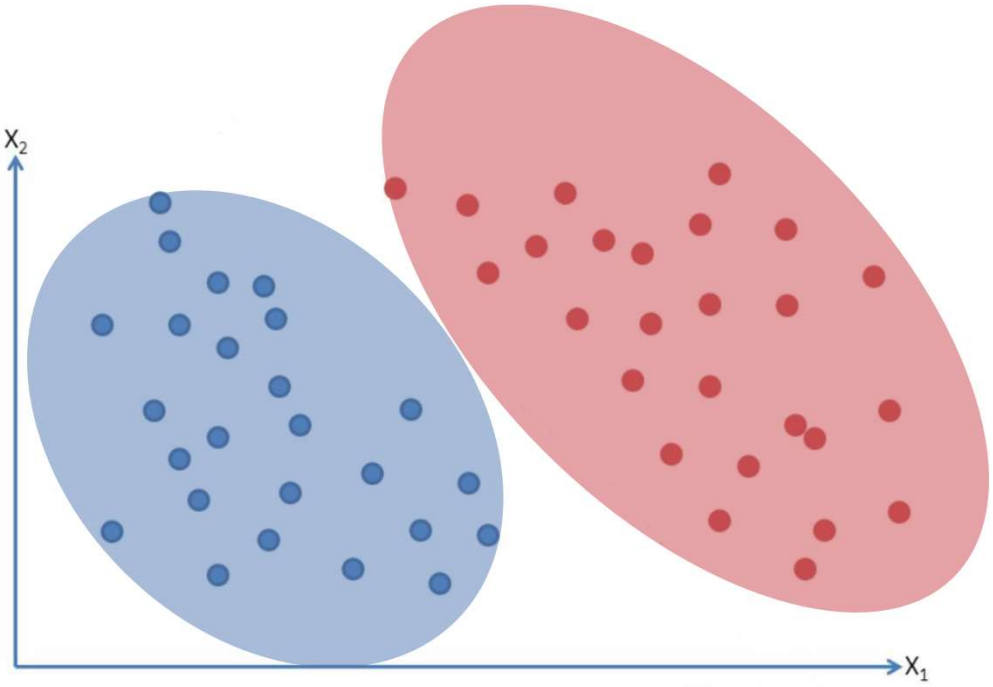
Discriminative model models the decision boundary between CLASSES / LABELS / TAGS

to

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

Generative vs Discriminative Models

Generative

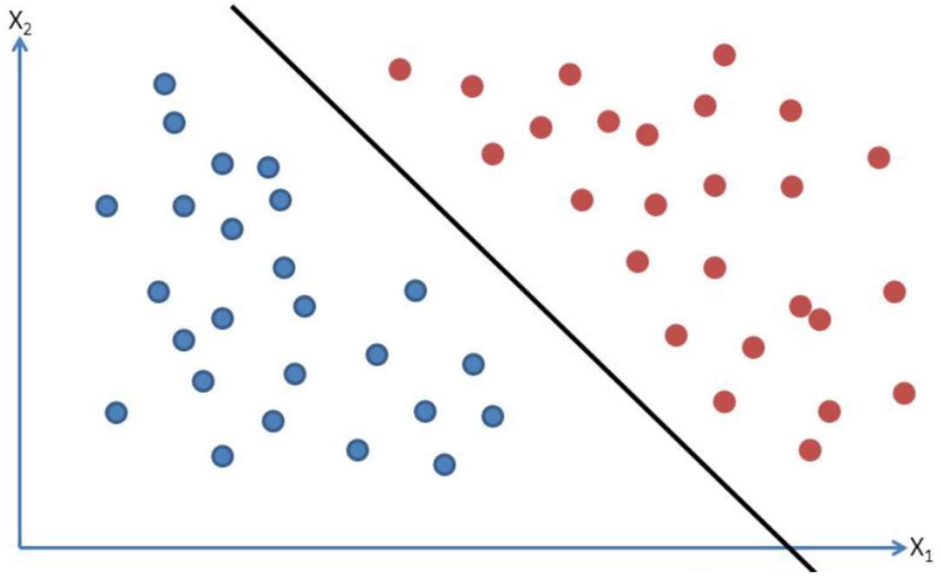


Generative model uses training data to learn P(sample, class) **joint probabilities**

and then

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

Discriminative



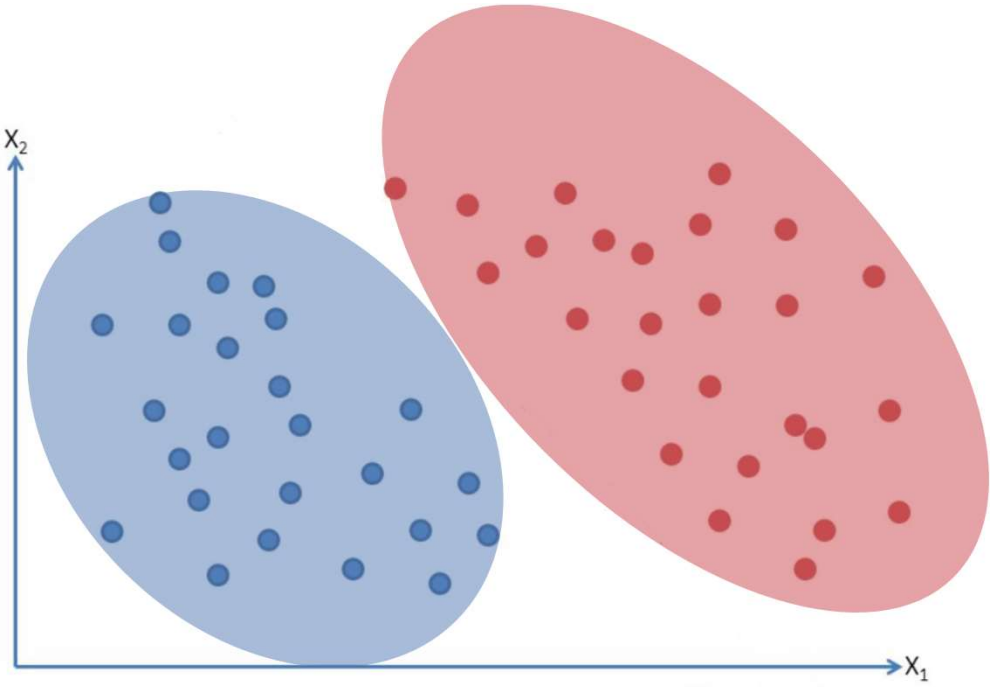
Discriminative model uses training data to learn P(class | sample) **conditional probability**

and then

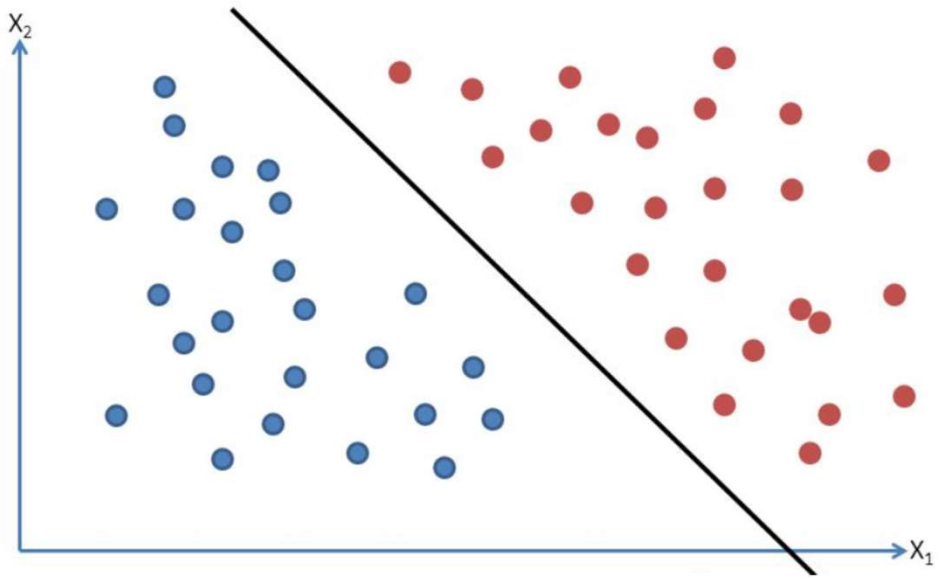
uses it to make a prediction

Generative vs Discriminative Models

Generative



Discriminative



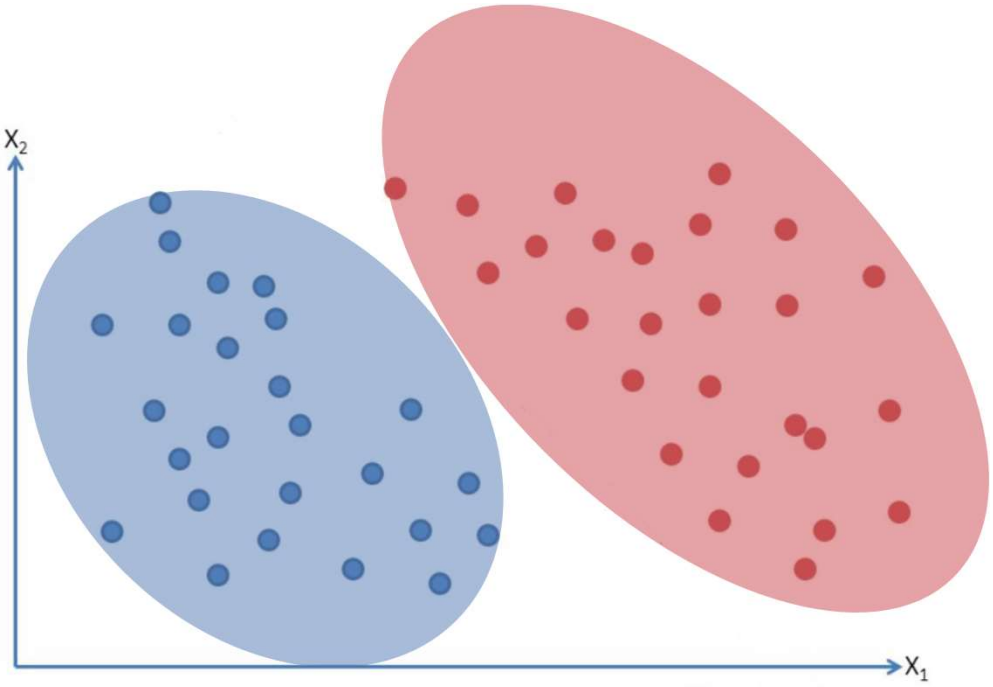
Generative model learns "details" about the class (its features):

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

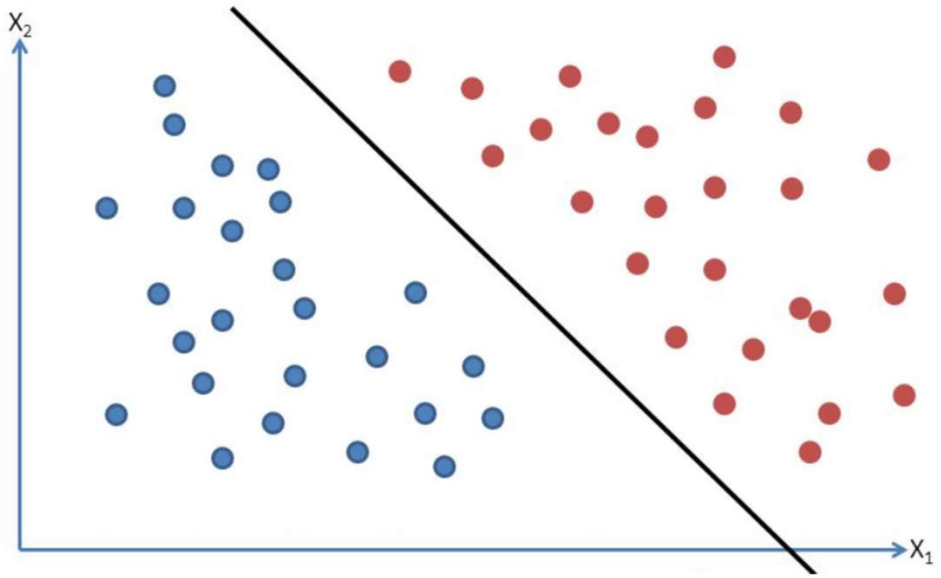
Discriminative model learns where the boundary between classes is and ignores the "details"

Generative vs Discriminative Classifier

Generative



Discriminative



Generative classifiers:

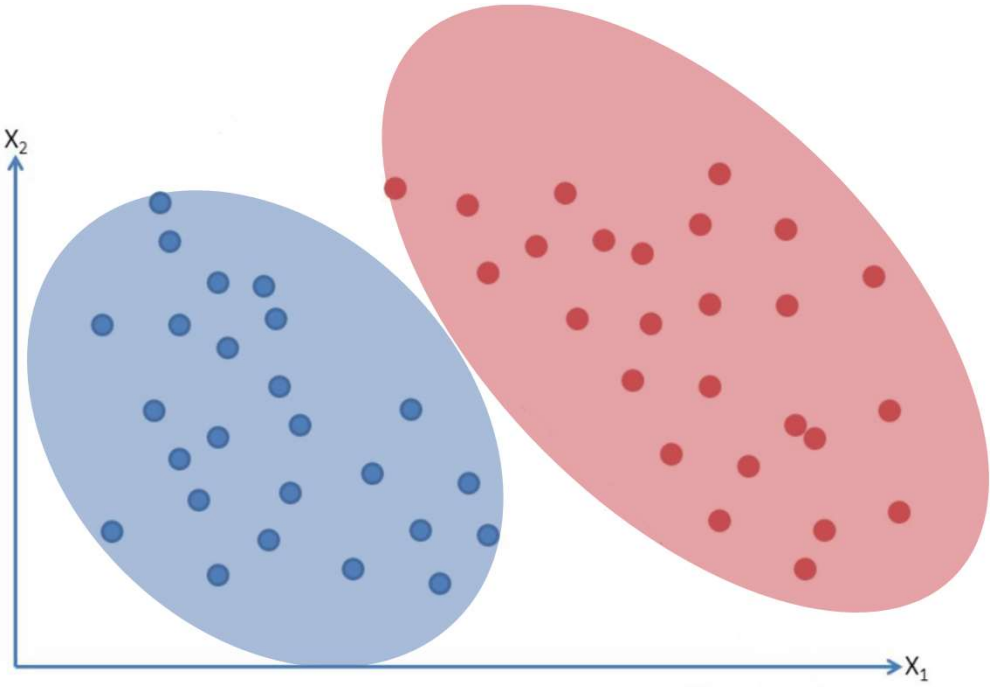
- **Assume some form of** P(class), P(sample | class)
- **Estimate** P(class), P(sample | class) **using training data**
- **Use Bayes Theorem to calculate** P(class | sample)

Discriminative classifiers:

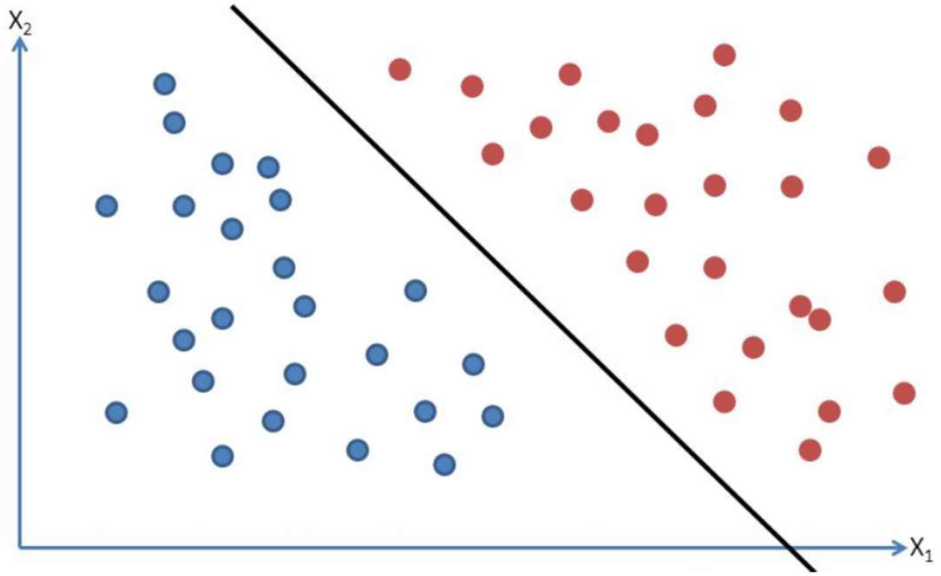
- **Assume some form of** P(class | sample)
- **Estimate** P(class | sample) **using training data**

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

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

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

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

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

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

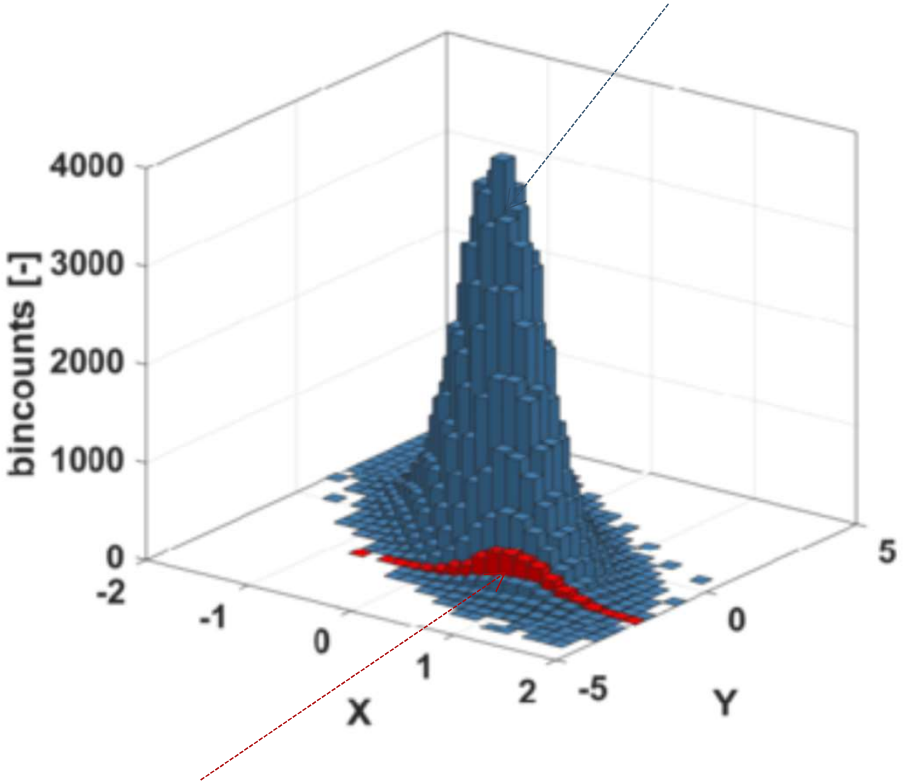
Logistic Regression will learn THIS directly

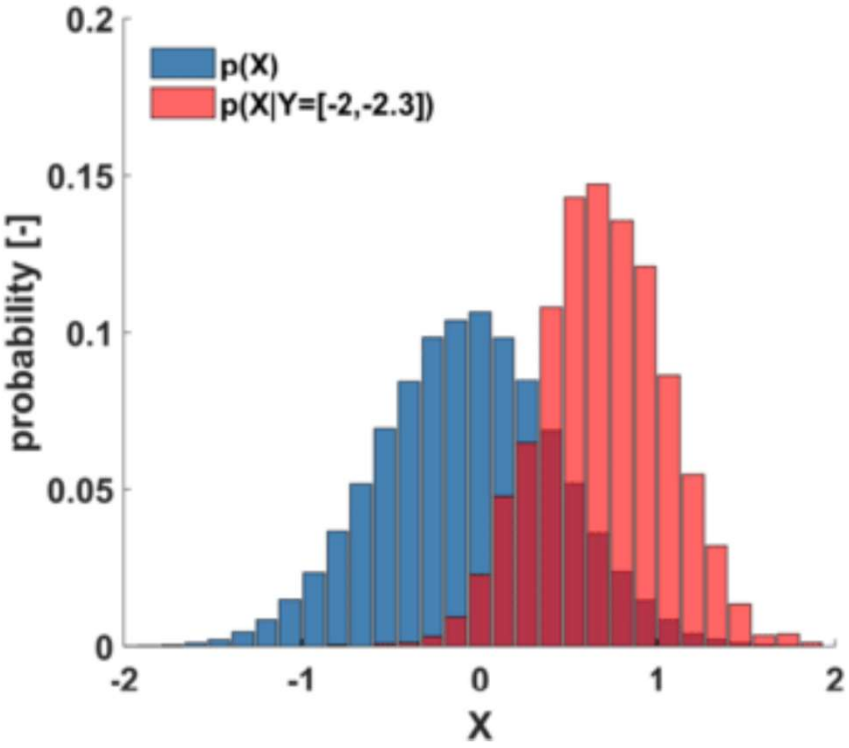
$\mathbf{x} = x_1, x_2, \ldots, x_N$, **so:**

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

Joint vs Conditional Probability

Naive Bayes learned THIS...



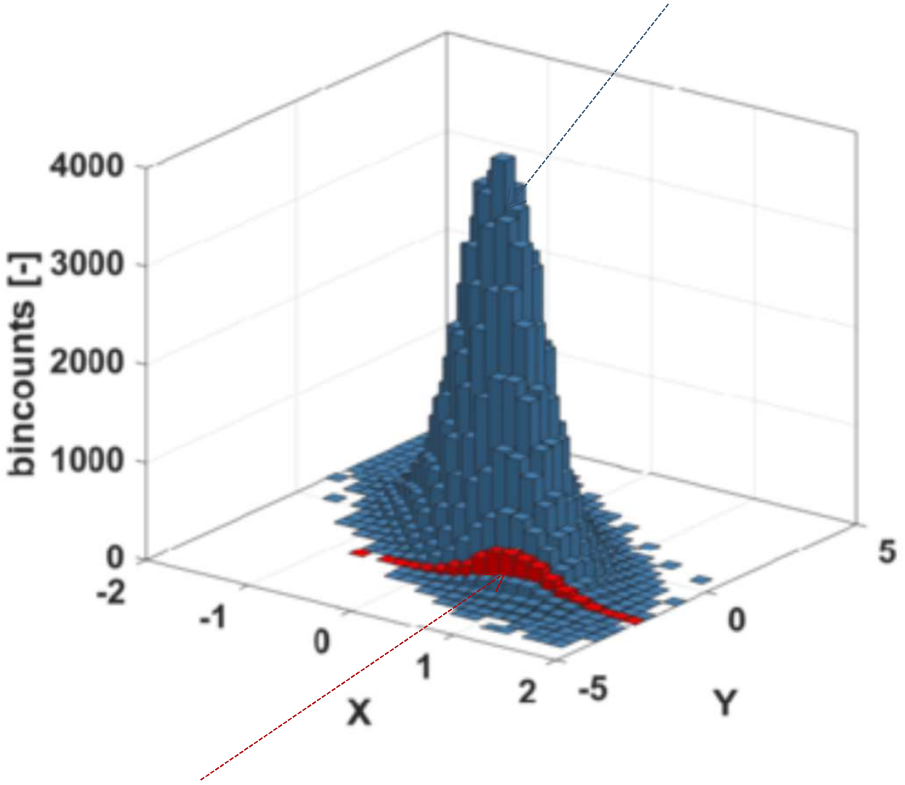


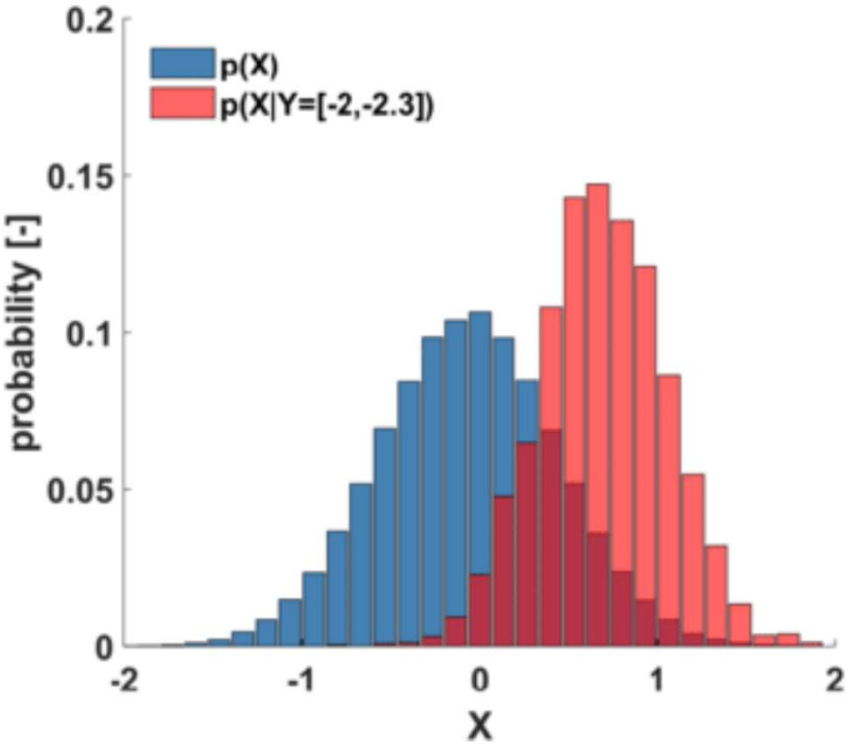
... to be able to estimate THAT

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...

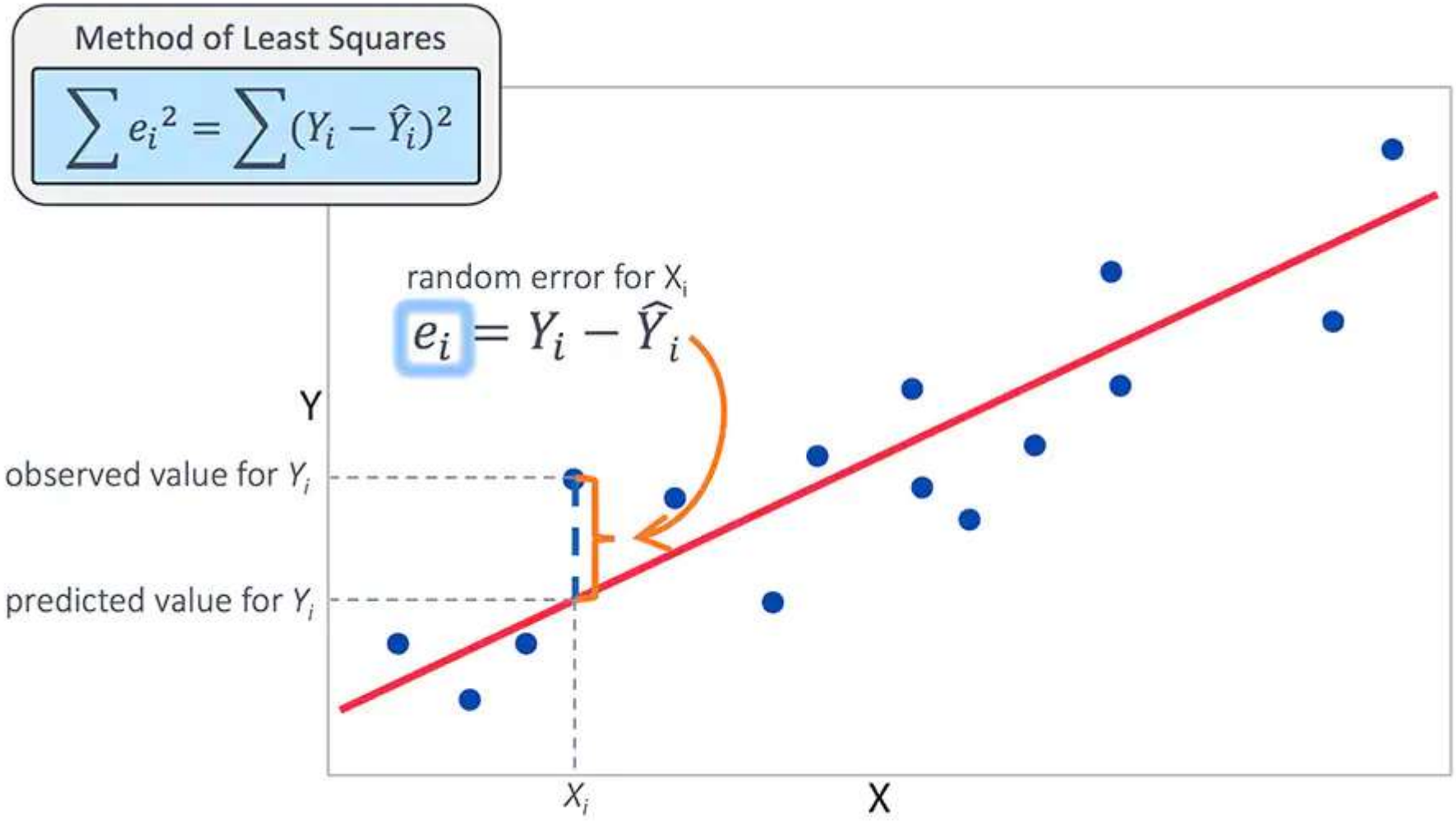




... and learns THAT instead

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

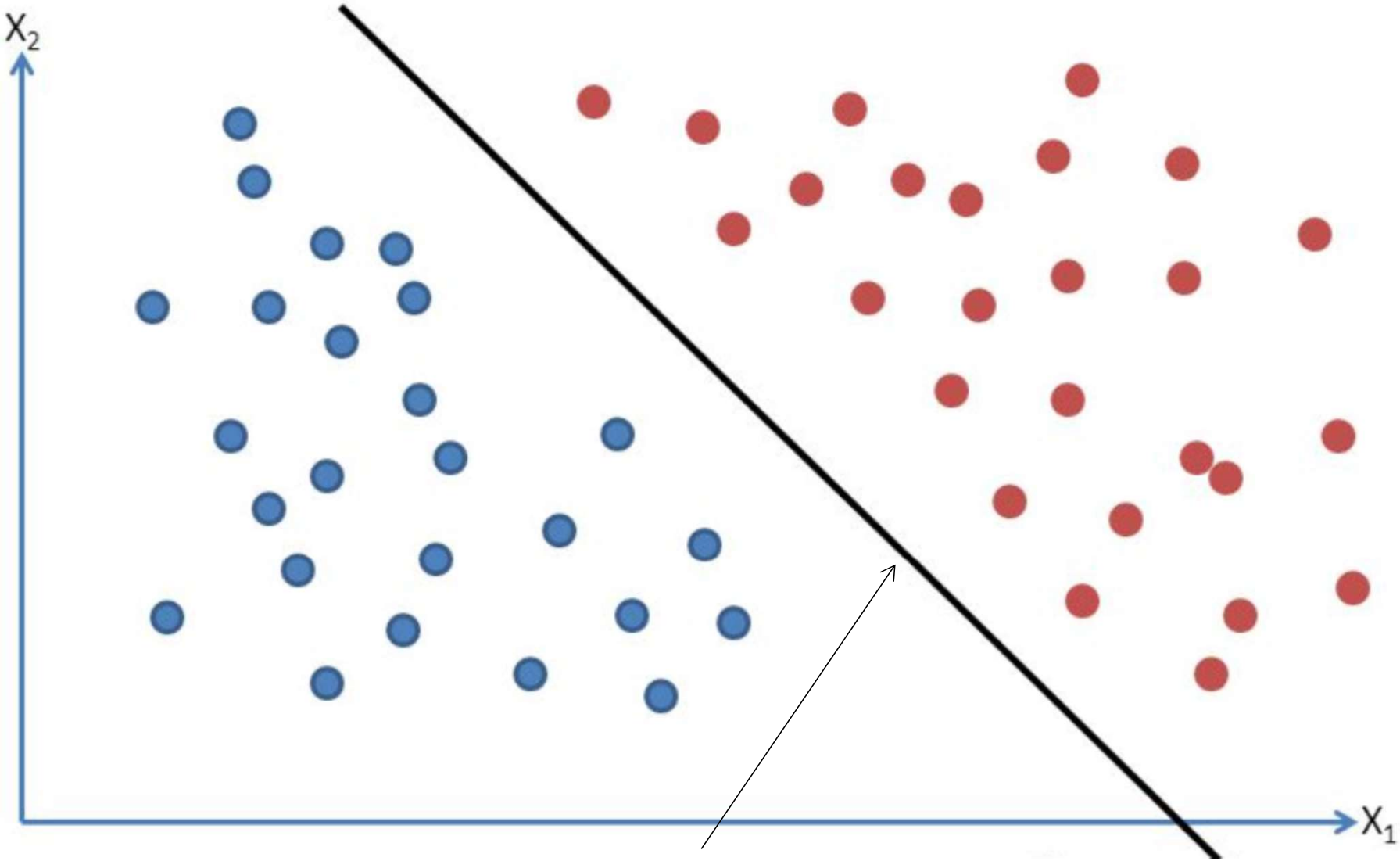
Linear Regression Using Least-Squares



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

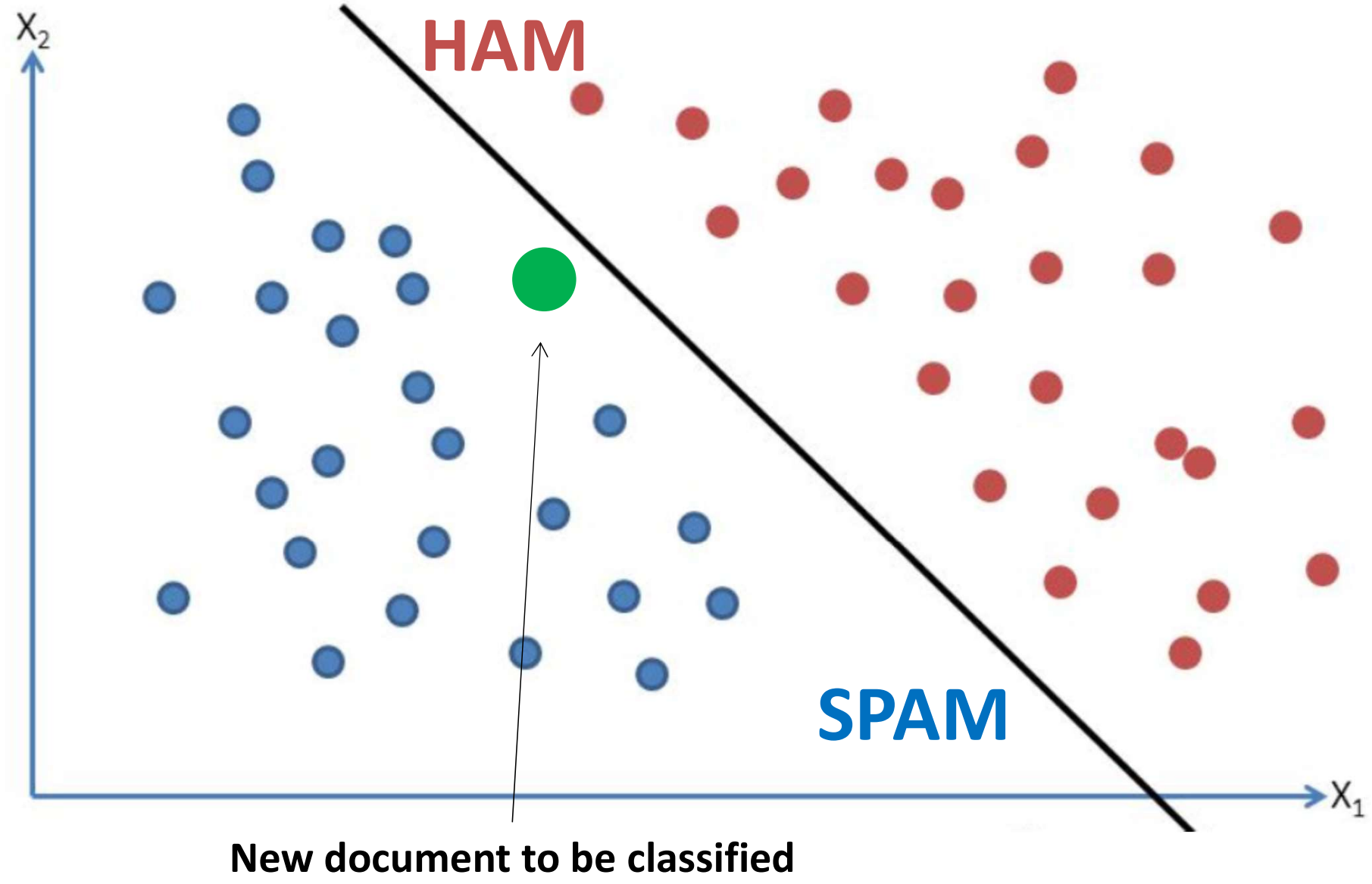
Source: https://www.jmp.com/en_us/statistics-knowledge-portal/what-is-multiple-regression/fitting-multiple-regression-model.html

Linear Separator

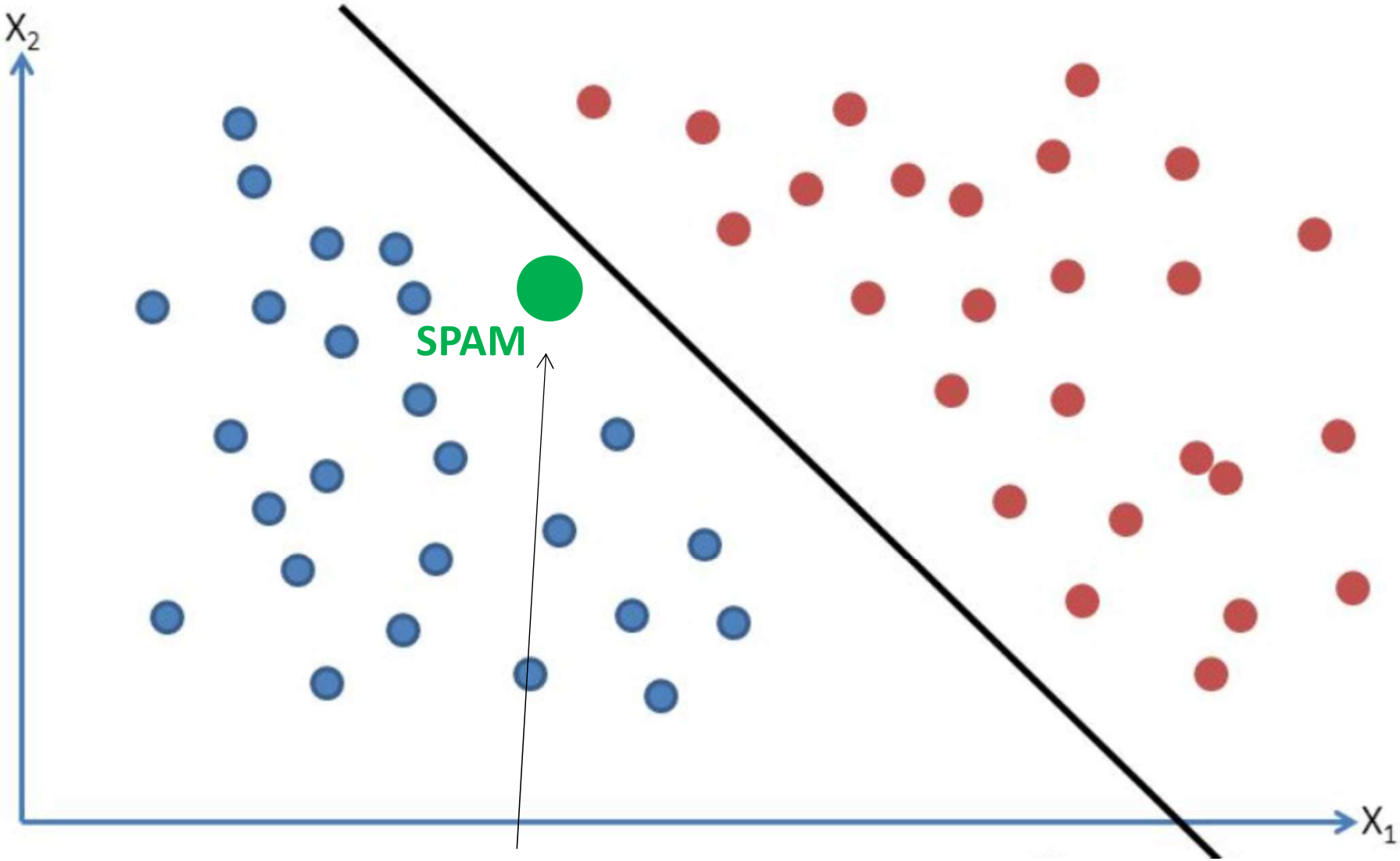


Line in 2D, plane in 3D, hyperplane in ND

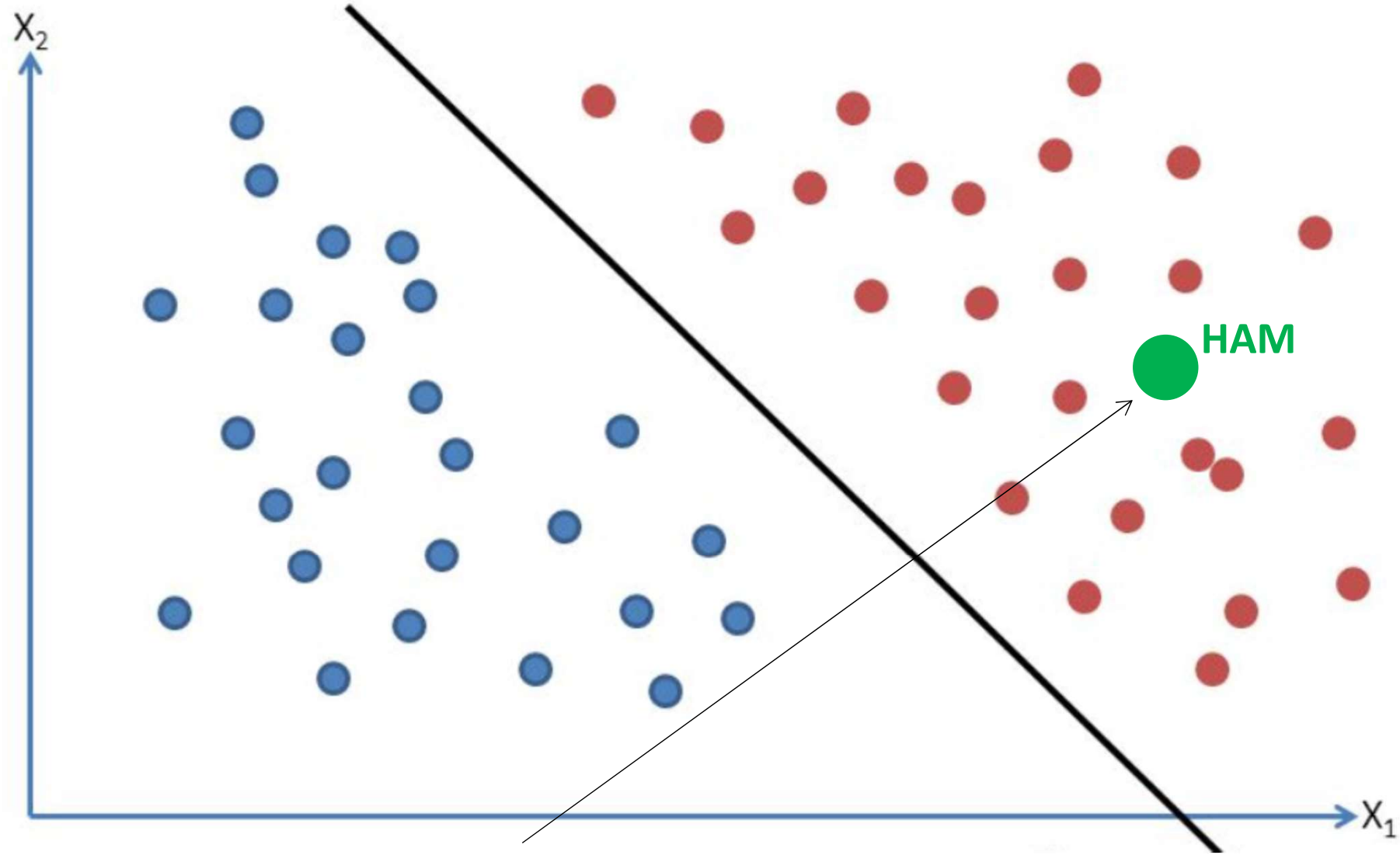
Text Classification: Separator



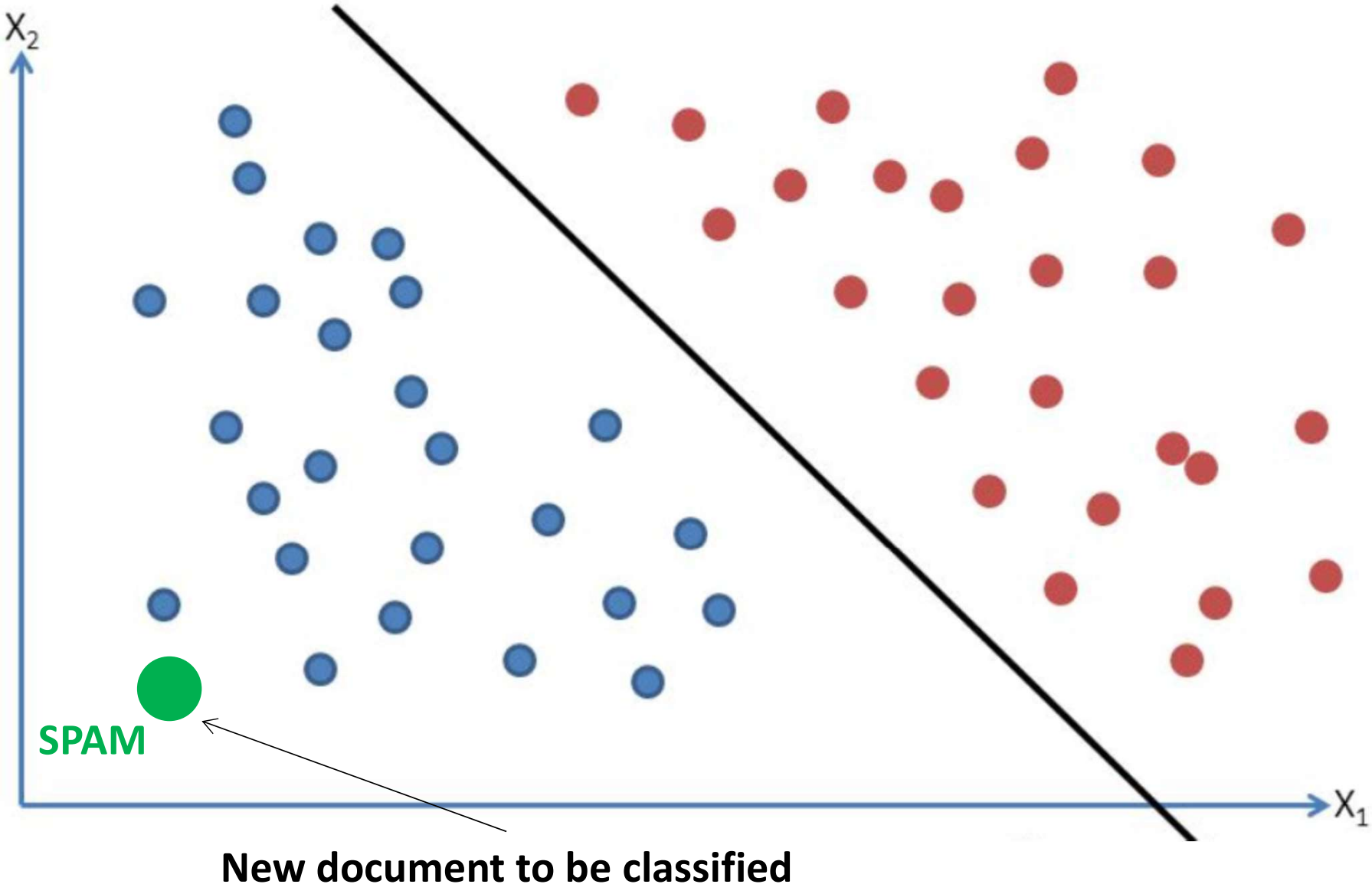
Text Classification: Separator



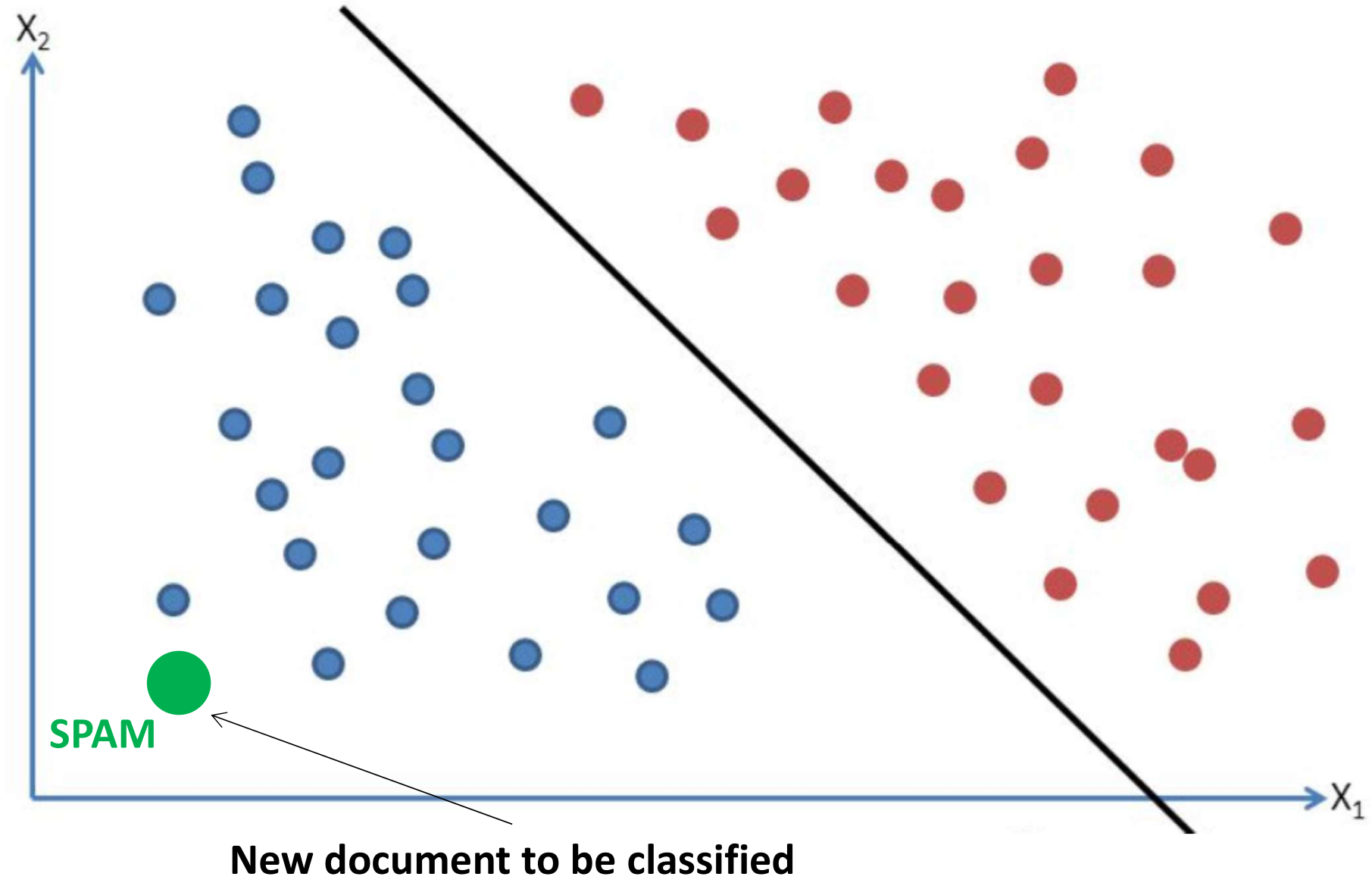
Text Classification: Separator



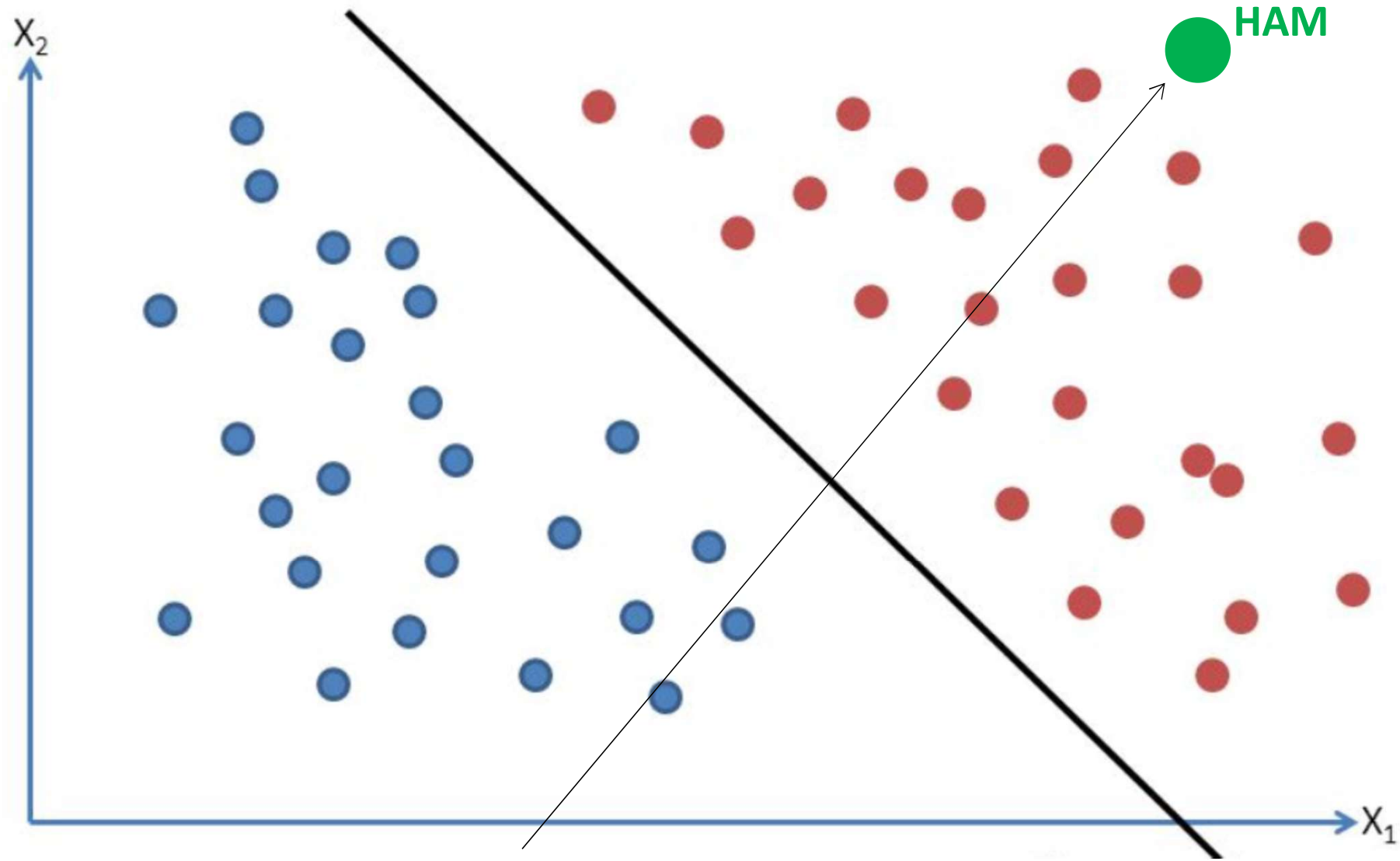
Text Classification: Separator



Text Classification: Separator



Text Classification: Separator



Vector Dot / Scalar Product

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

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

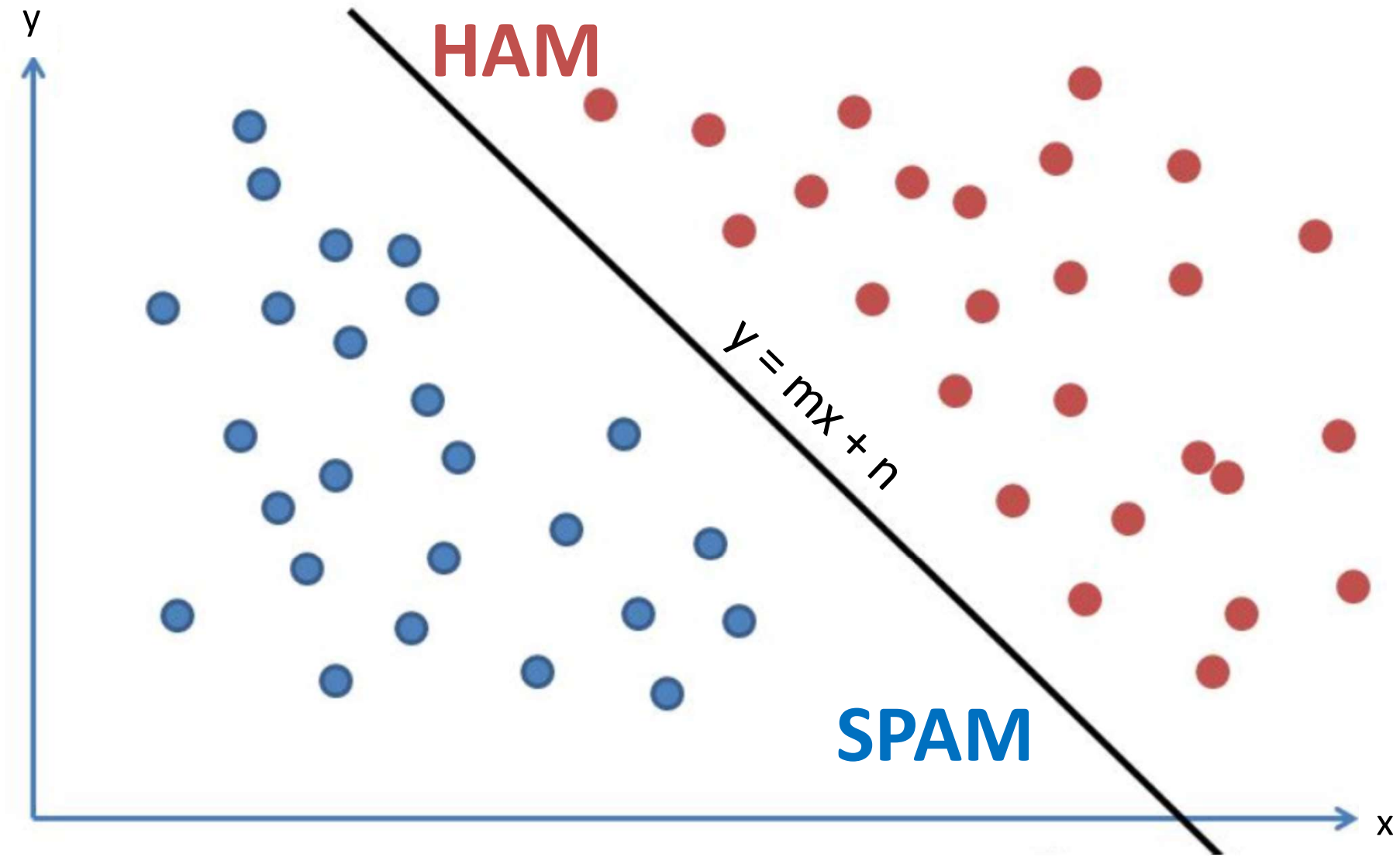
their vector dot/scalar product is:

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

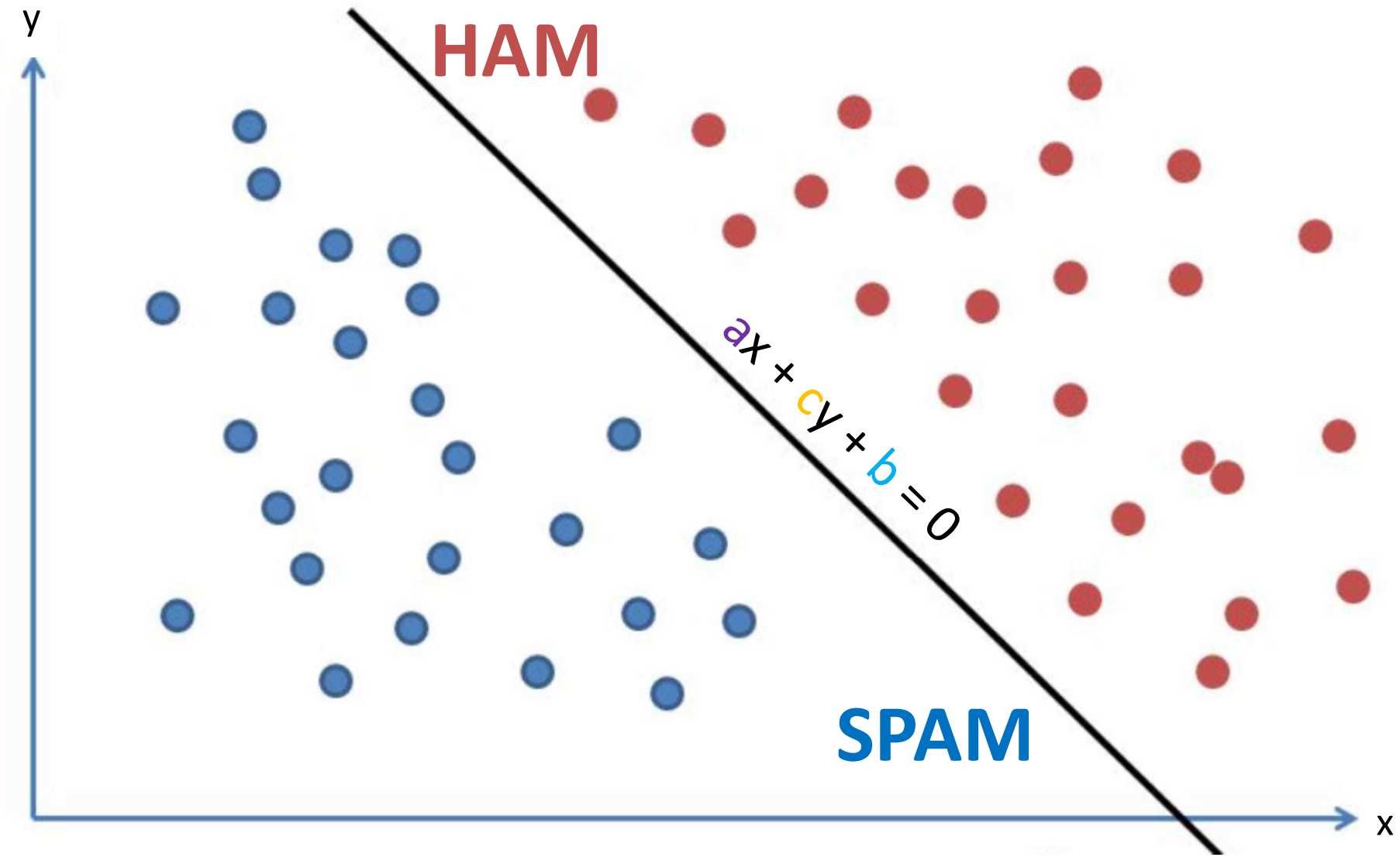
Using matrix representation:

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

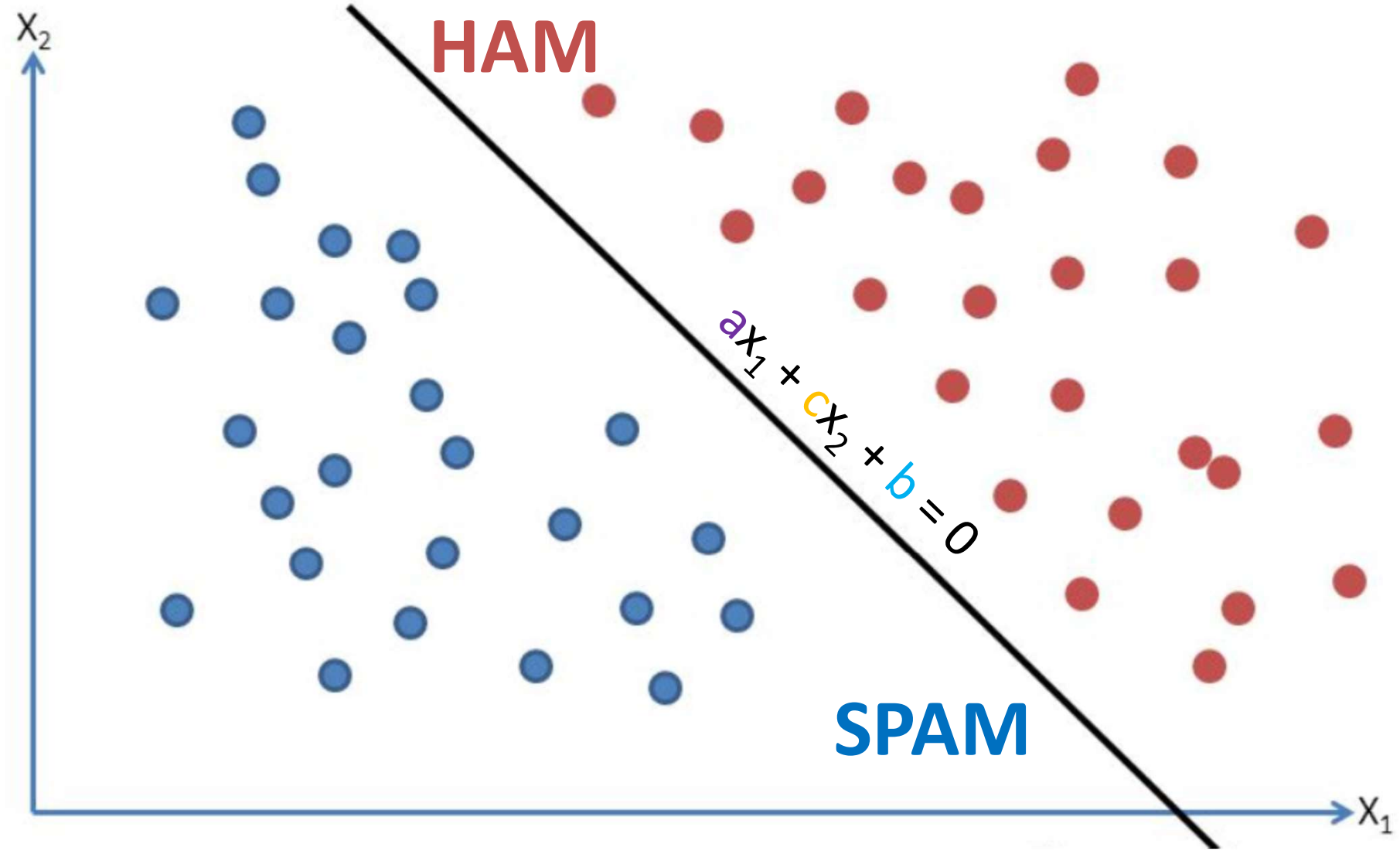
Text Classification: Separator



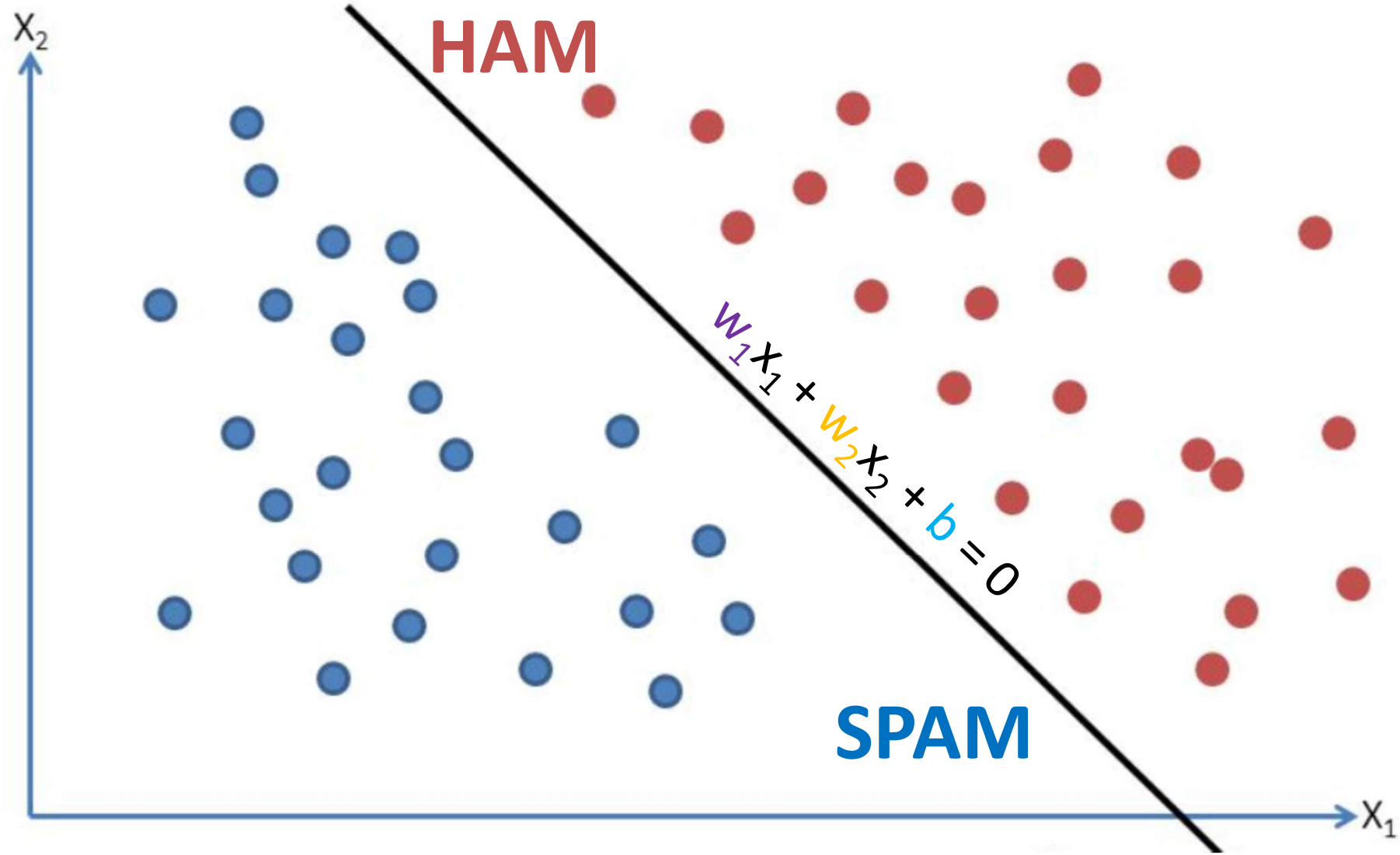
Text Classification: Separator



Text Classification: Separator



Text Classification: Separator



Vector Dot / Scalar Product

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

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

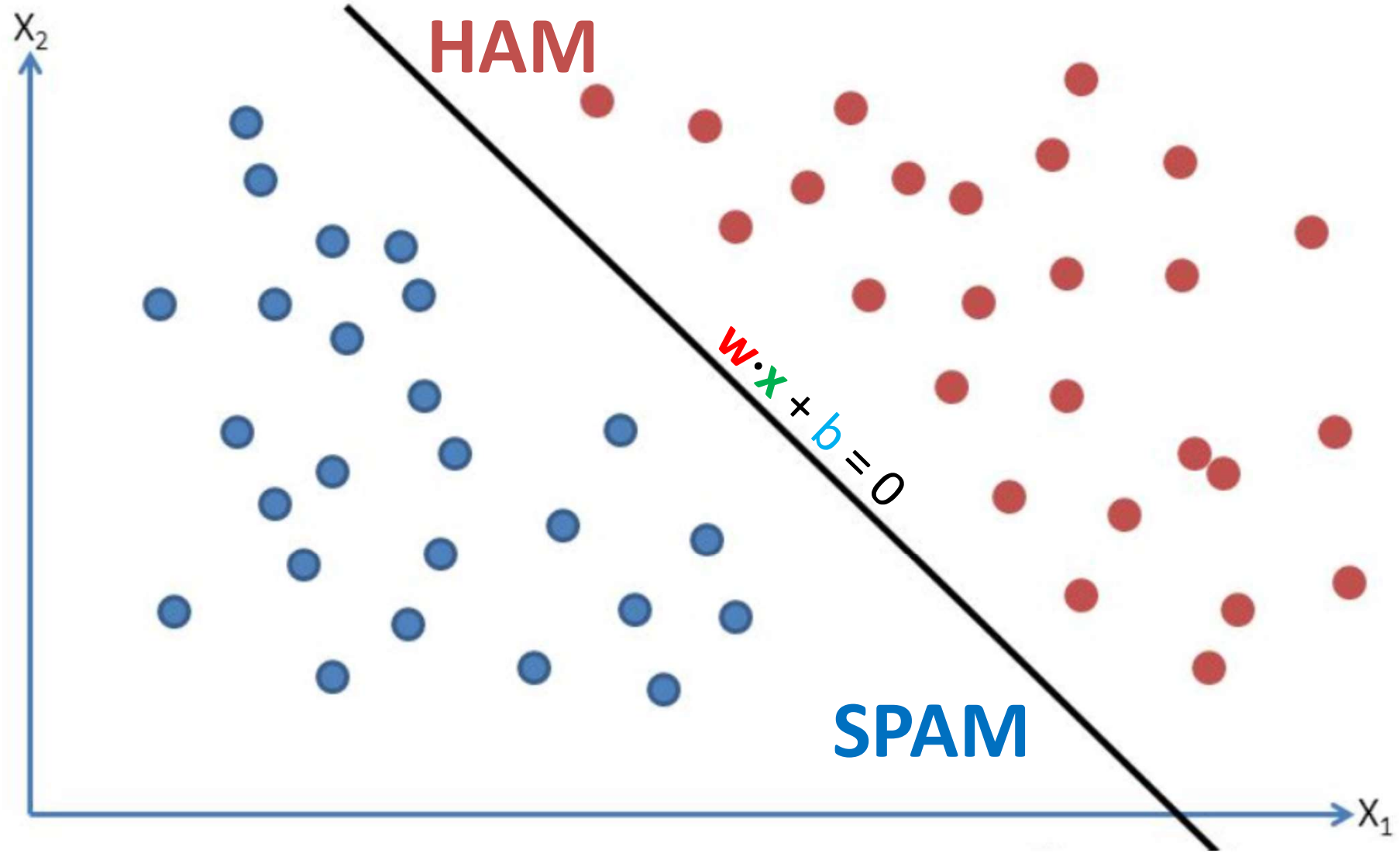
their vector dot/scalar product is:

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

Using matrix representation:

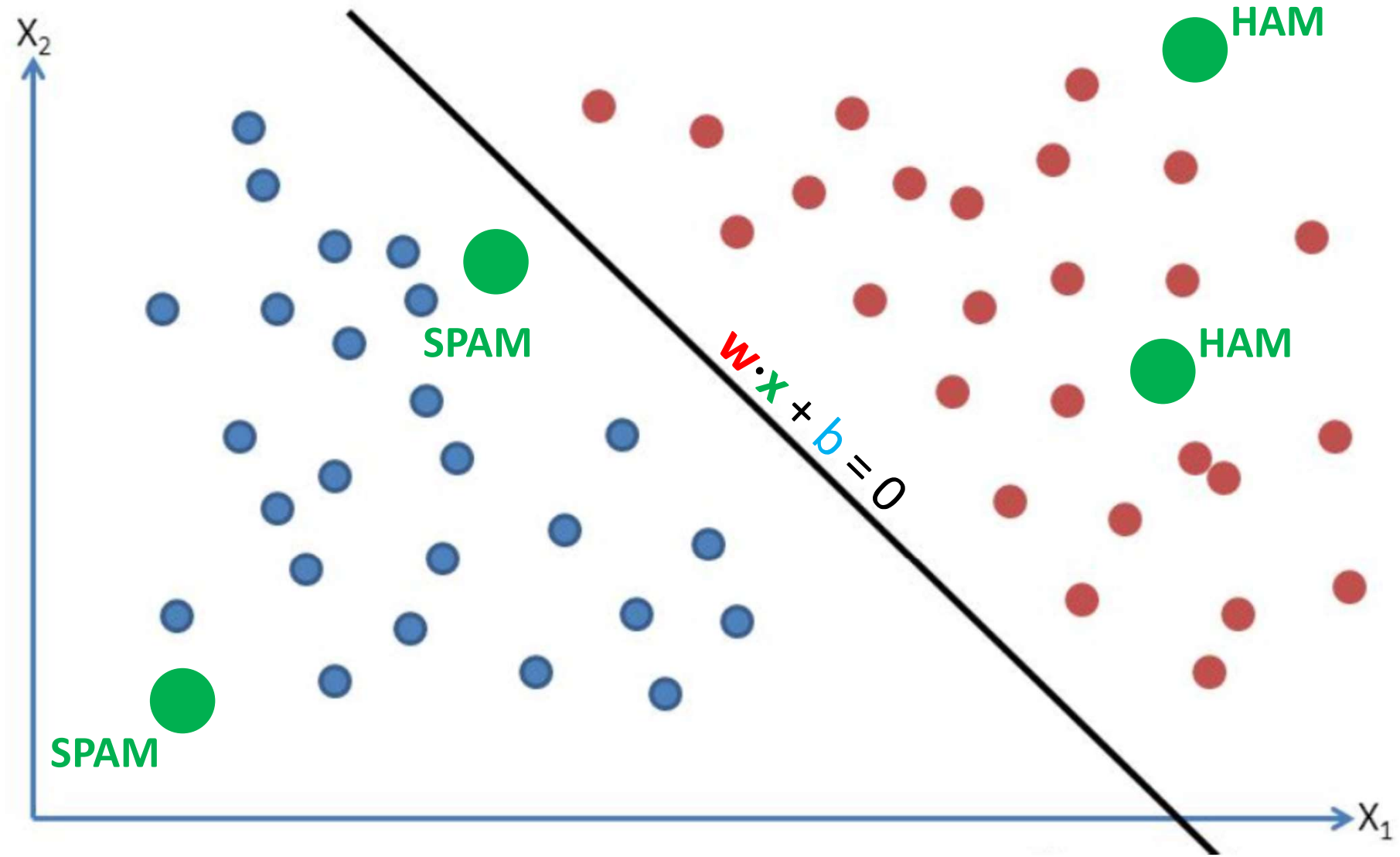
$$\boldsymbol{w} \cdot \boldsymbol{x} = \boldsymbol{w}\boldsymbol{x}^{\boldsymbol{T}} = [w_1 \quad w_2 \quad \cdots \quad w_N] \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix}$$

Text Classification: Separator

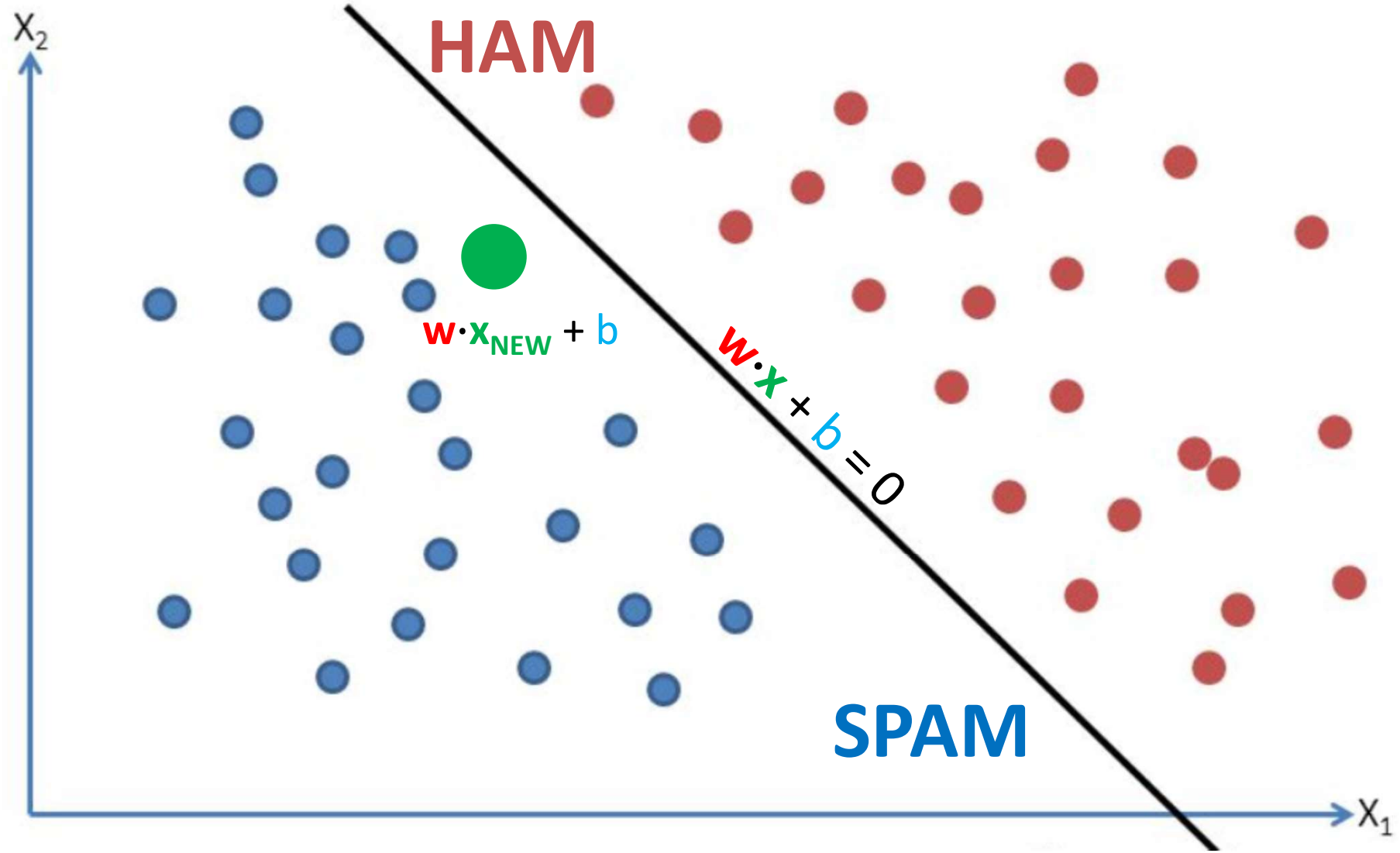


where: $\boldsymbol{w} = [w_1, w_2]$ and $\boldsymbol{x} = [x_1, x_2]$

Text Classification: Separator

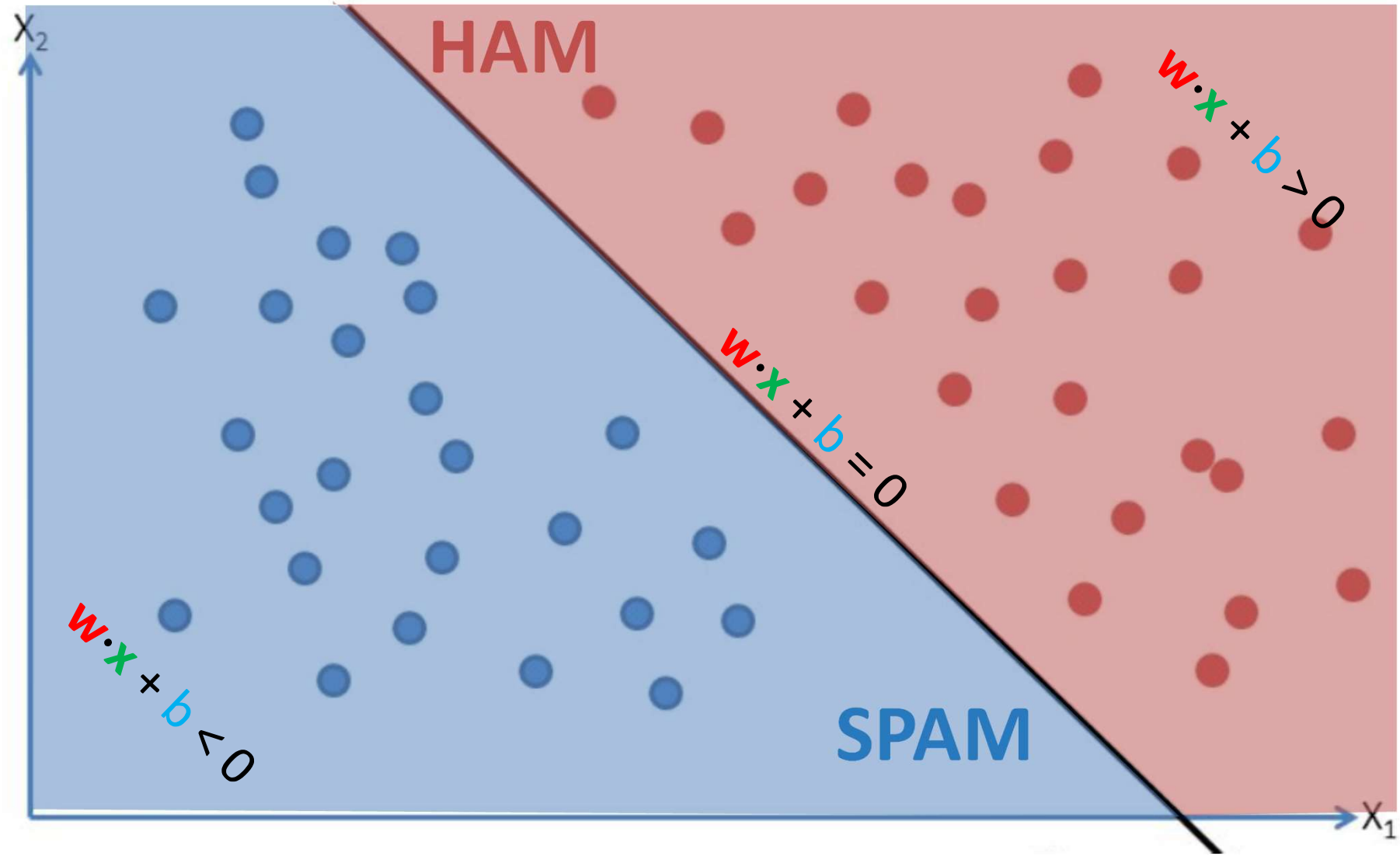


Text Classification: Separator



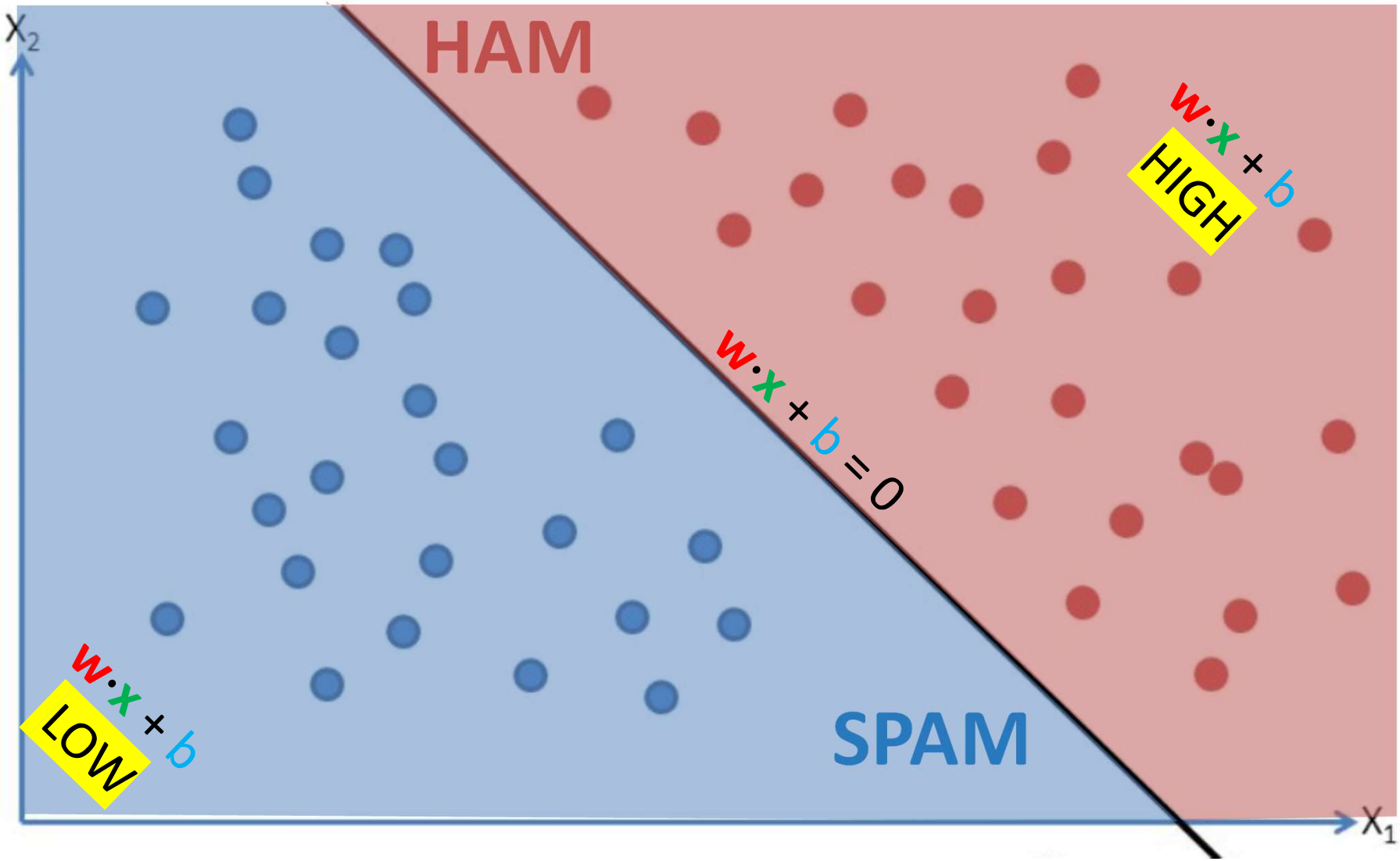
where: $w = [w_1, w_2]$ and $x = [x_1, x_2]$

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.

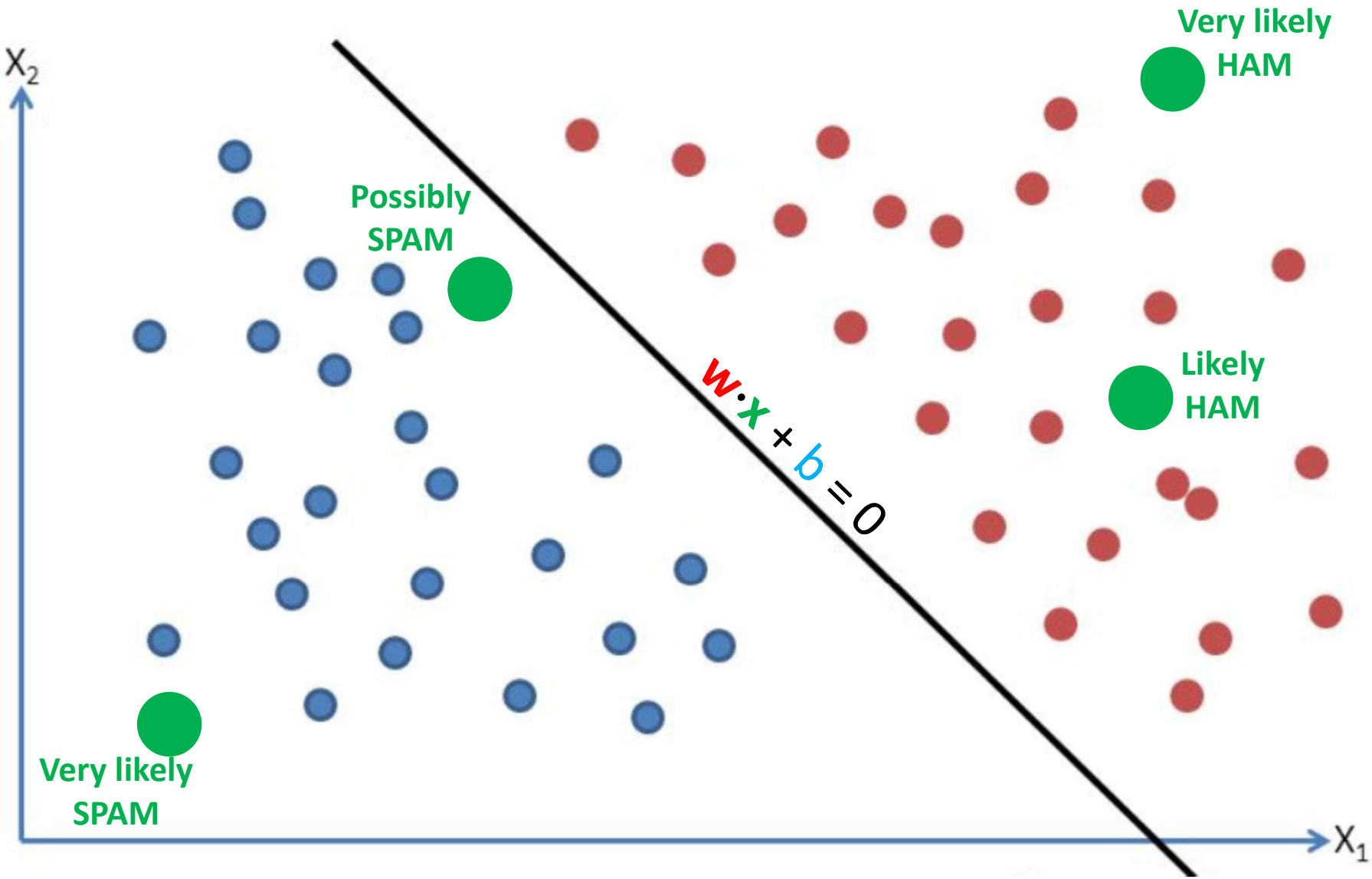
We'd like a principled classifier that gives us a probability, just like Naive Bayes did

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



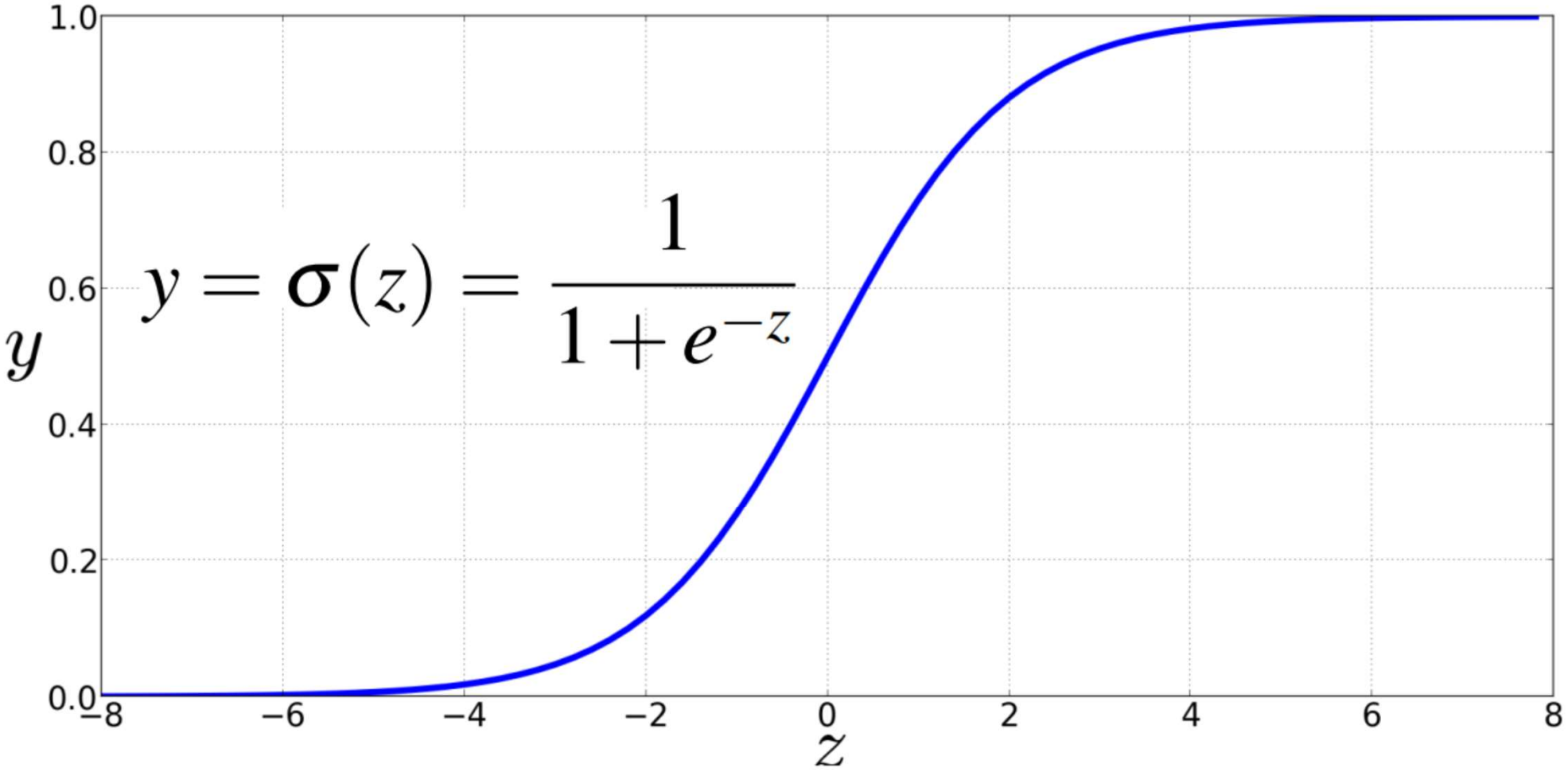
Logistic Regression: The Idea

- 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...

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

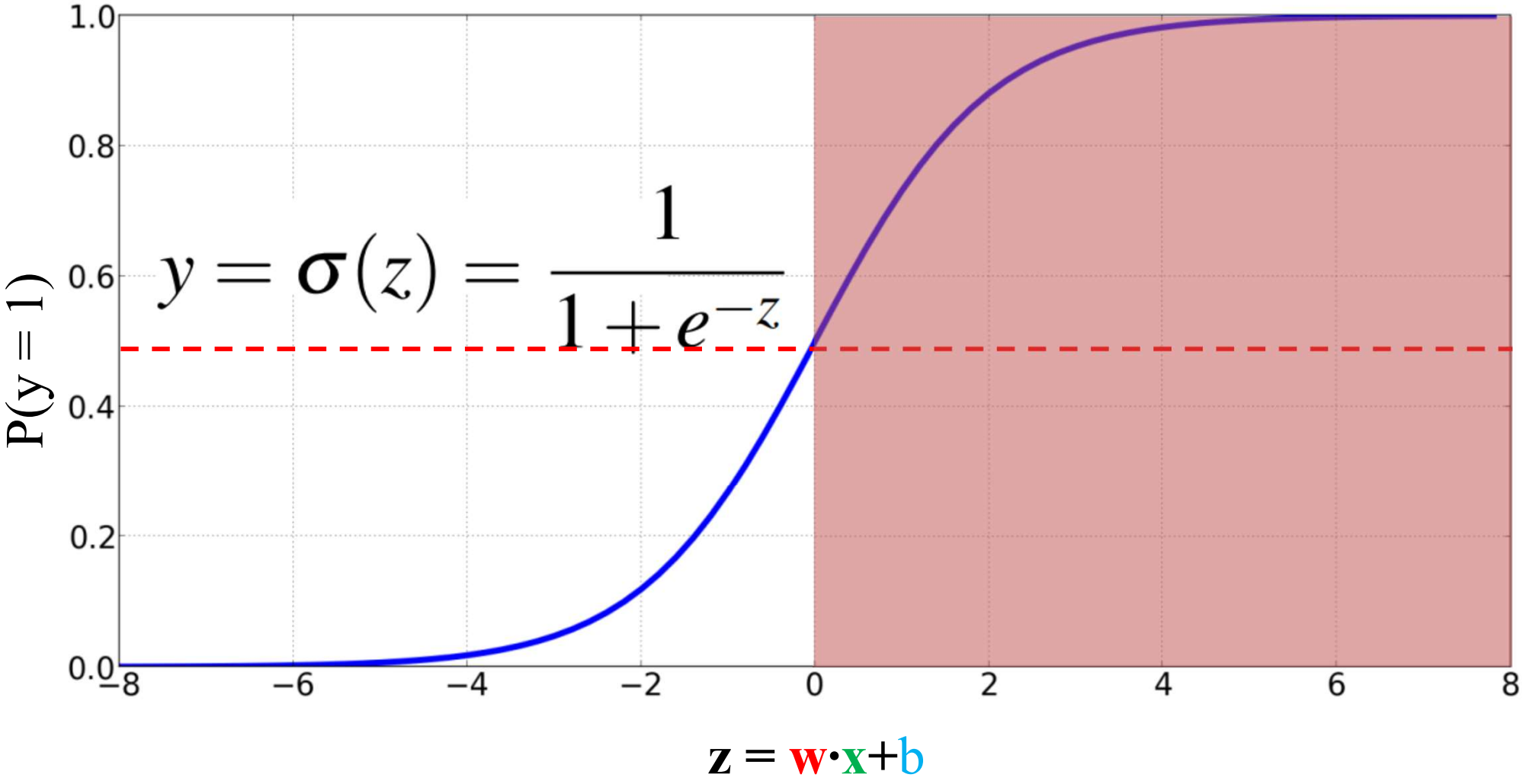
Probabilities → Classification

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

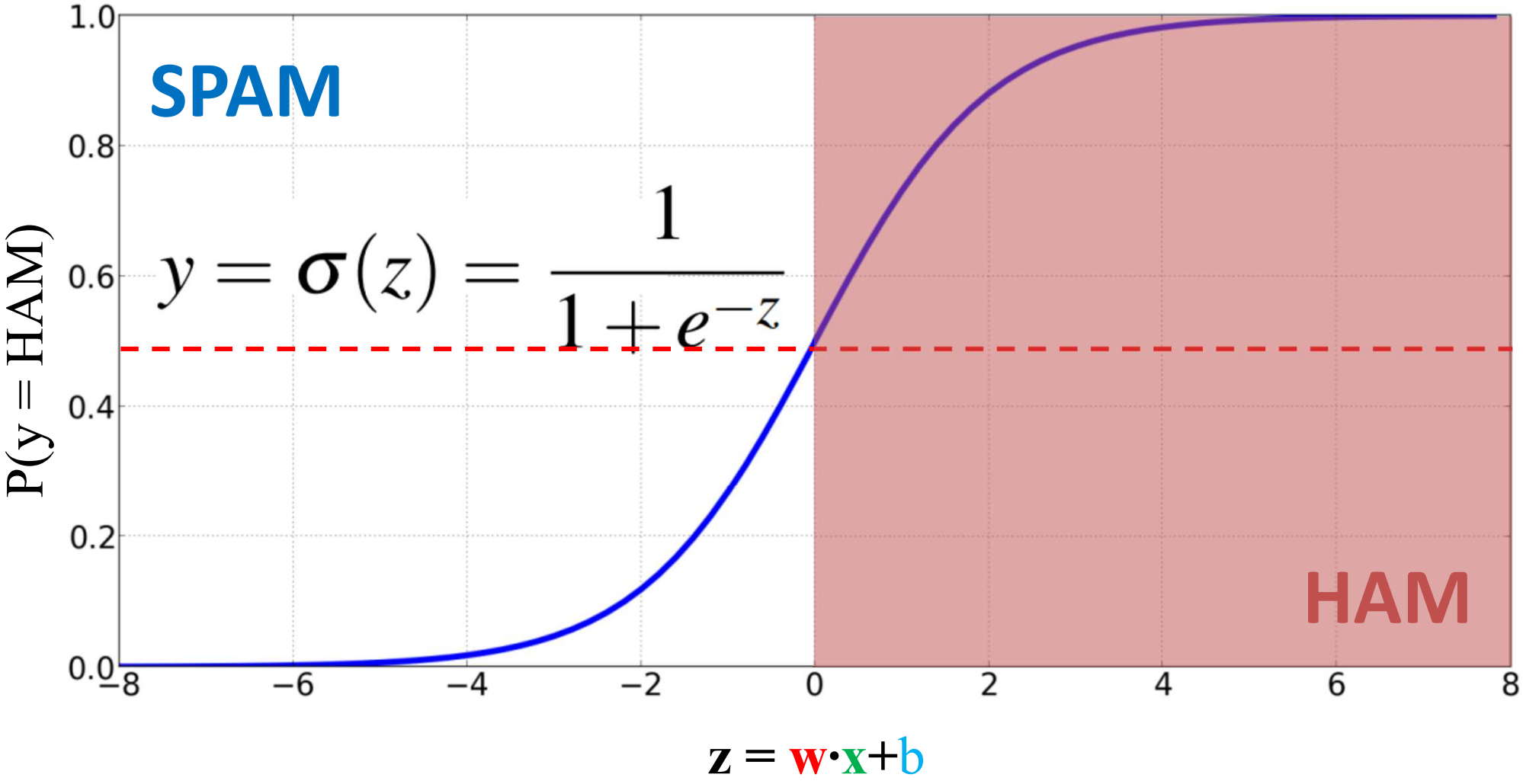
$$\hat{y} = \begin{cases} 1 & \text{if } P(y=1|x) > 0.5 \\ 0 & \text{otherwise} \end{cases}$$

Where 0.5 is the decision boundary

Logistic Regression Classifier



Logistic Regression Classifier



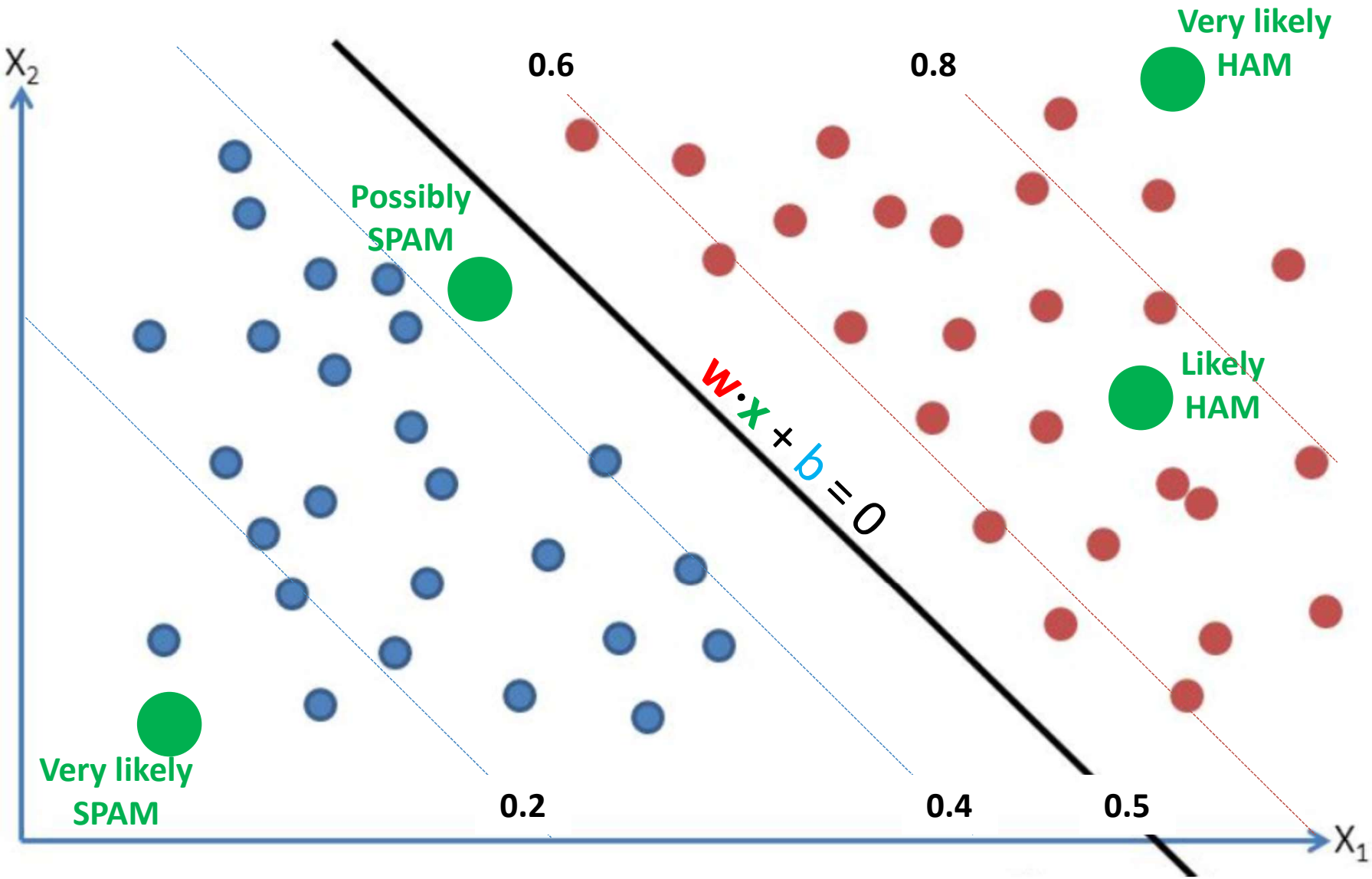
Probabilities → Classification

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

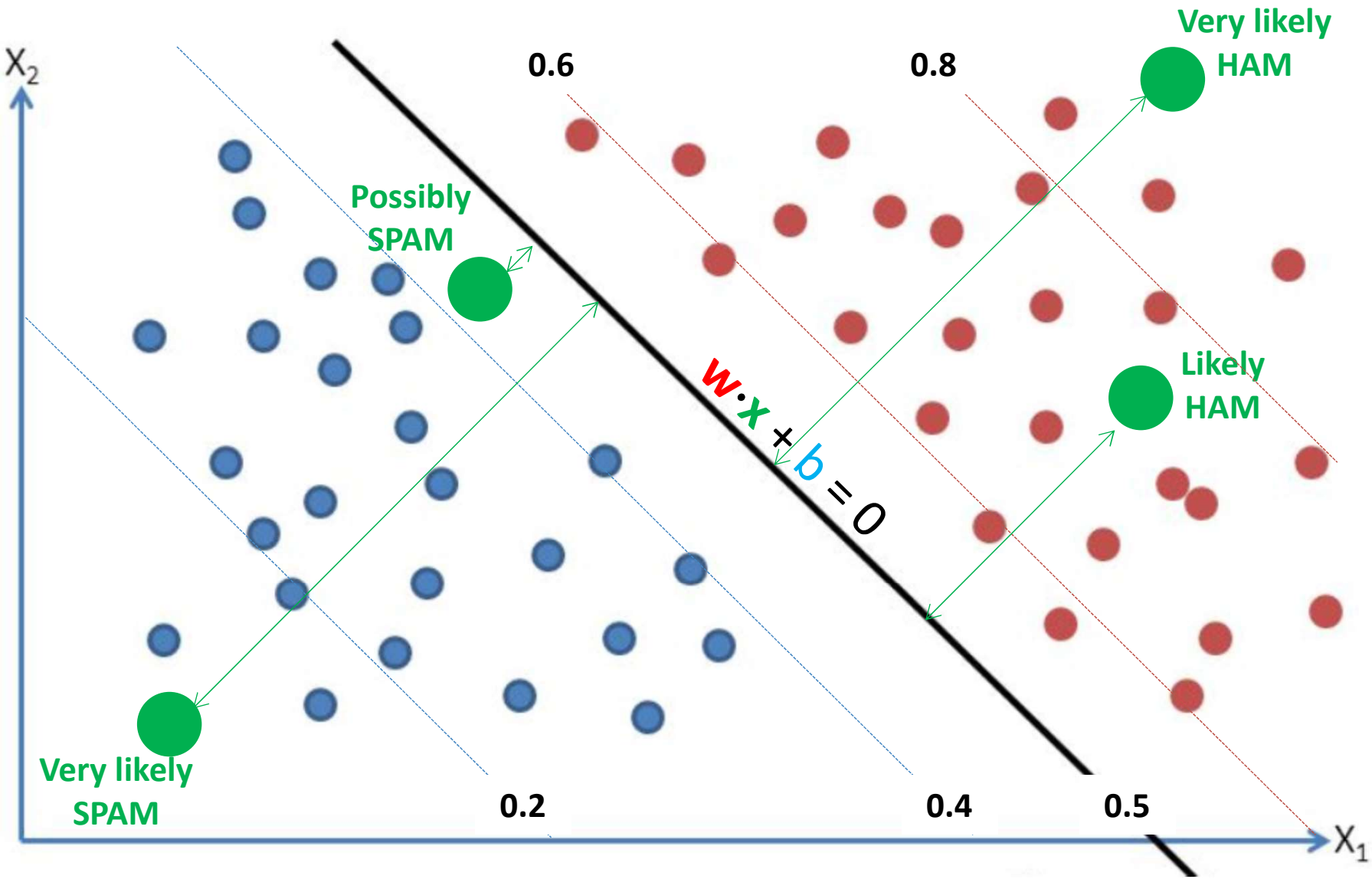
$$\hat{y} = \begin{cases} 1 & \text{if } P(y=1|x) > 0.5 \quad \text{if } w \cdot x + b > 0 \\ 0 & \text{otherwise} \quad \text{if } w \cdot x + b \leq 0 \end{cases}$$

Where 0.5 is the decision boundary

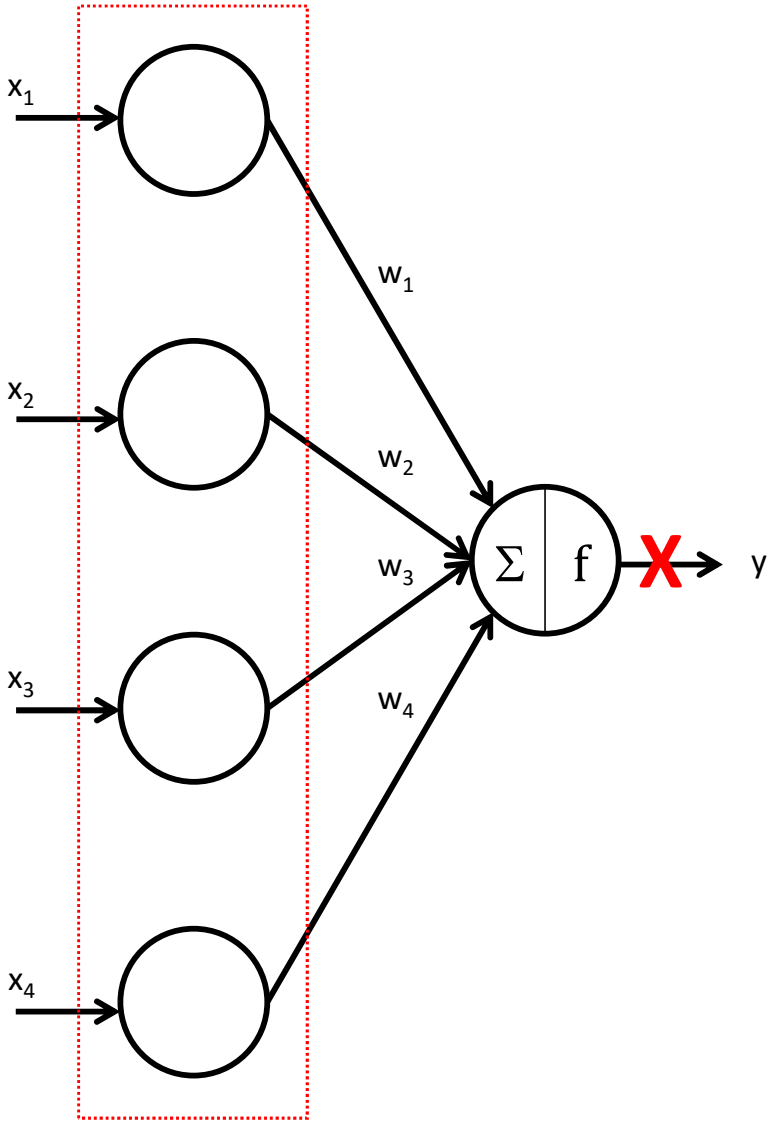
Text Classification: Separator



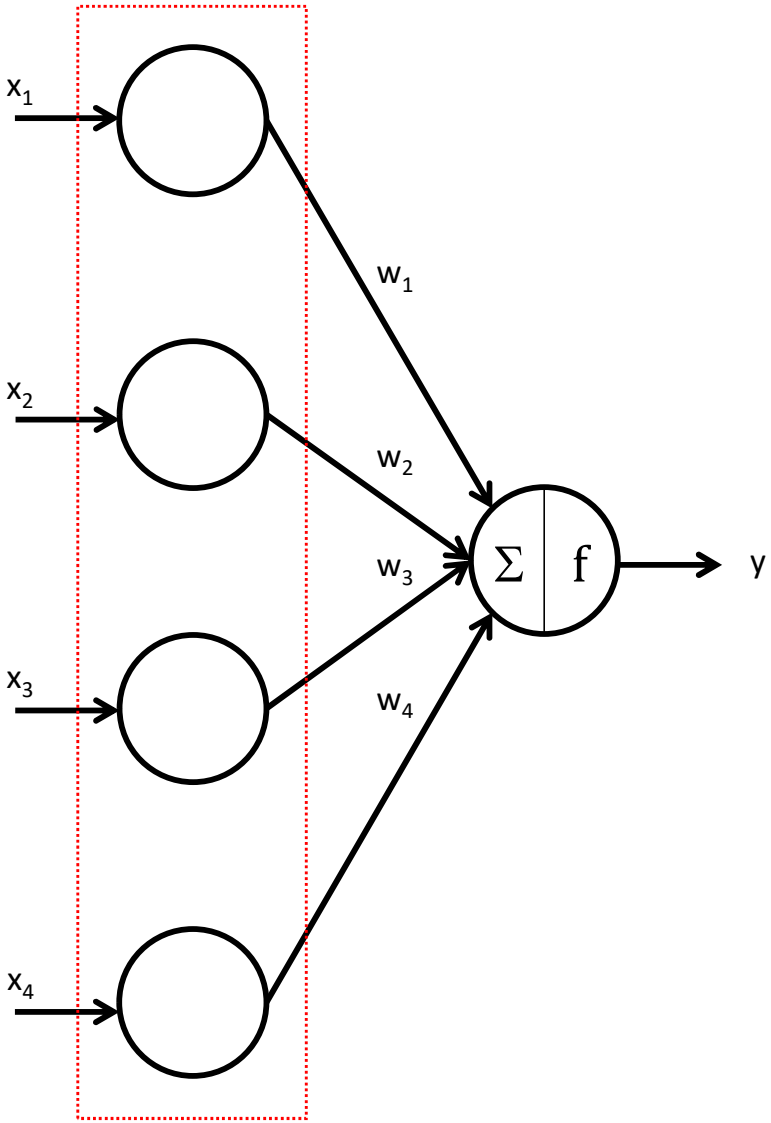
Text Classification: Separator



Does This Resemble Anything?

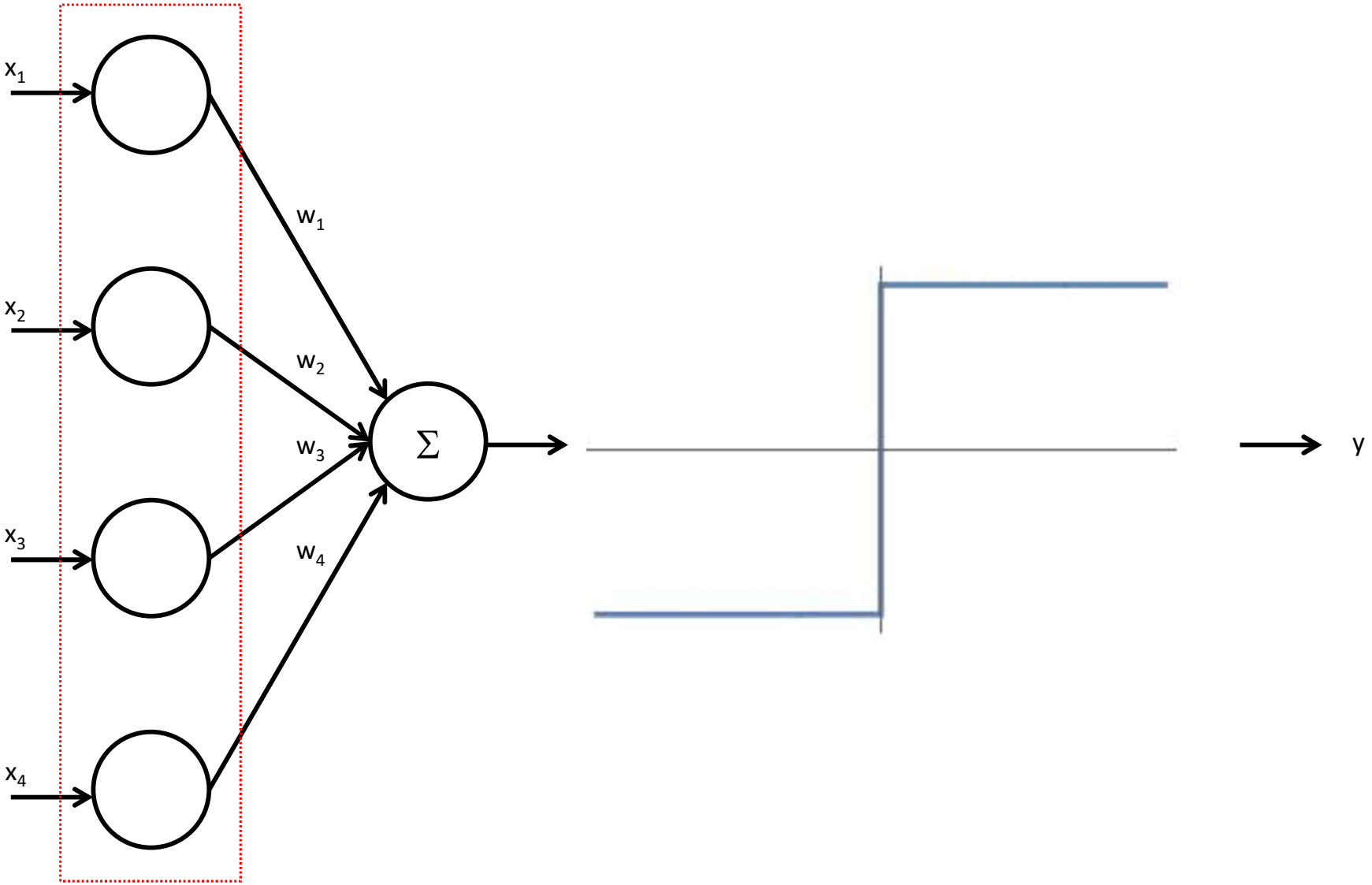


$\Sigma\, w_i * x_i < 0 \rightarrow f = 0 \rightarrow$ **NO**



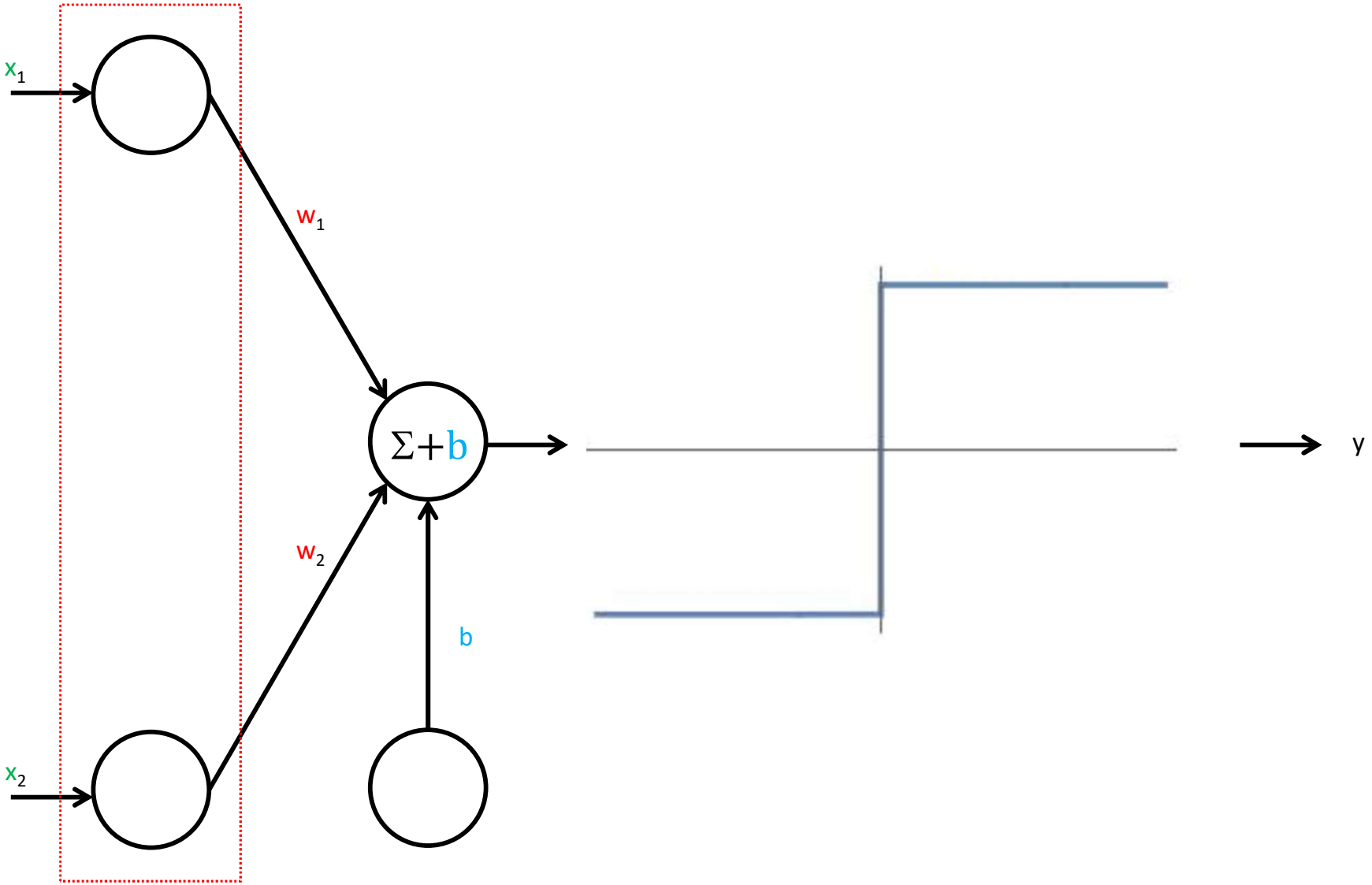
$\Sigma\, w_i * x_i \geq 0 \rightarrow f = 1 \rightarrow$ **YES**

Step Activation Function



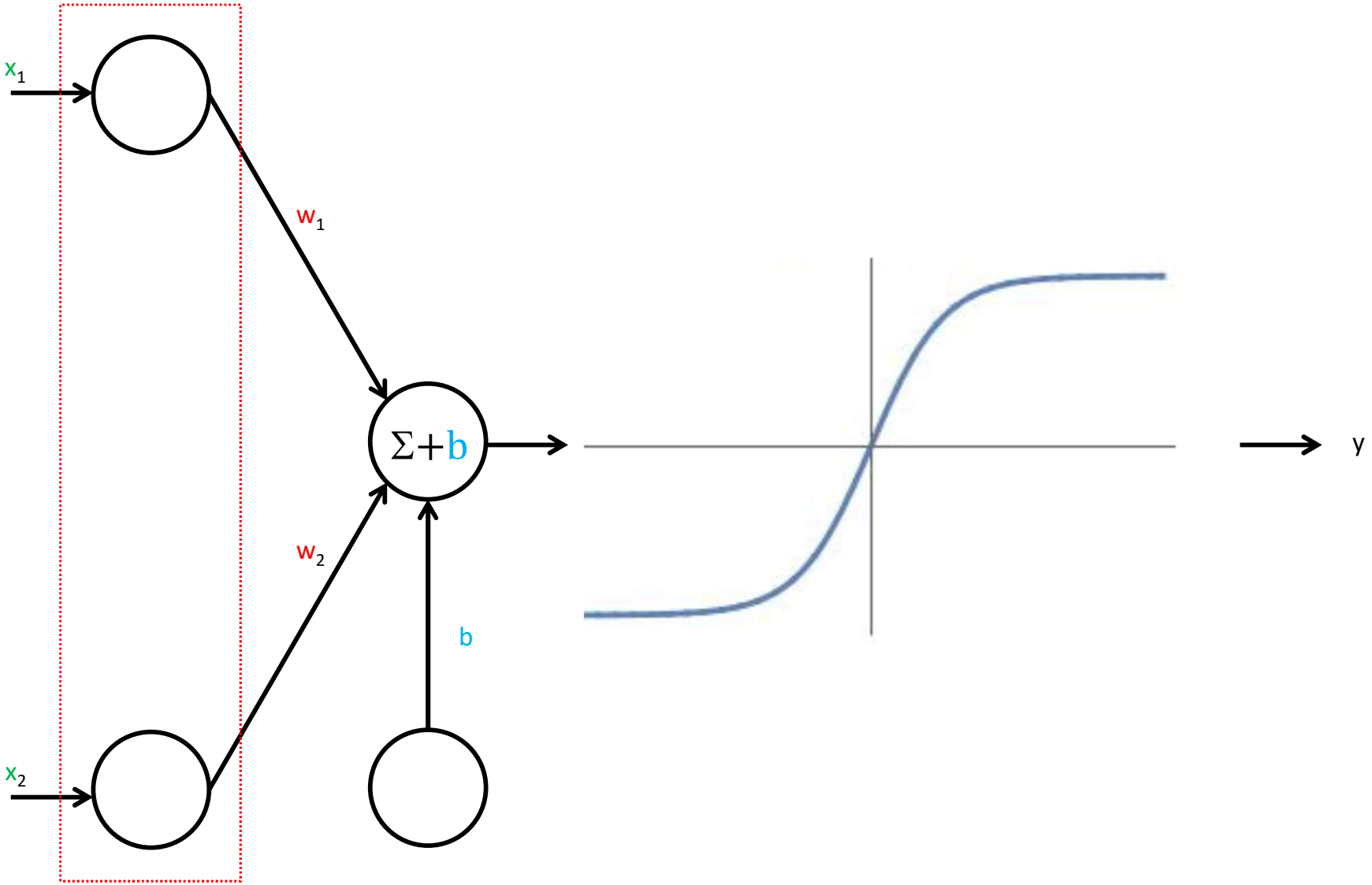
$\Sigma\ w_i * x_i \rightarrow f$

Step Activation Function



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

Sigmoid Activation Function



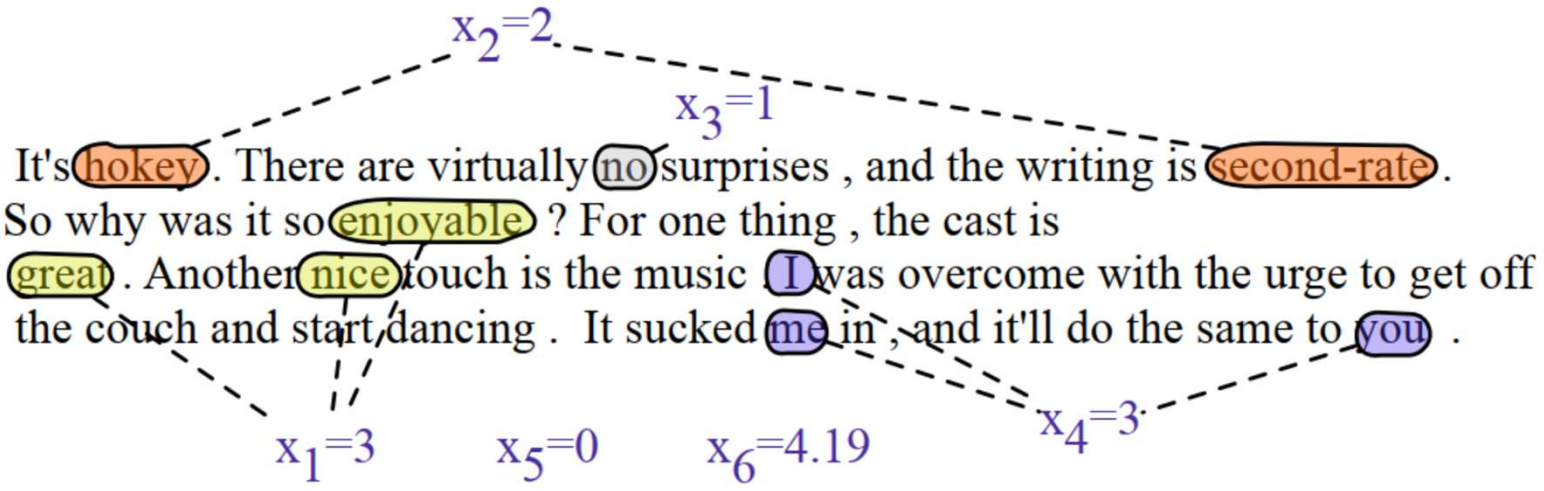
$\Sigma\ w_i * x_i + b \rightarrow f \rightarrow y$

Sentiment Analysis: Example

Sample document:

It's hokey . There are virtually no surprises , and the writing is second-rate . So why was it so enjoyable ? For one thing , the cast is great . Another nice touch is the music . I was overcome with the urge to get off the couch and start dancing . It sucked me in , and it'll do the same to you .

Sentiment Analysis: Example



Feature vector:

Var	Definition	Value
x_1	count(positive lexicon) $\in$ doc)	3
x_2	count(negative lexicon) $\in$ doc)	2
x_3	$\begin{cases} 1 & \text{if "no"} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$	1
x_4	count(1st and 2nd pronouns $\in$ doc)	3
x_5	$\begin{cases} 1 & \text{if "!"} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$	0
x_6	log(word count of doc)	$\ln(66) = 4.19$

Sentiment Analysis: Example

Feature vector x:

Var	Definition	Value
x_1	count(positive lexicon) $\in$ doc)	3
x_2	count(negative lexicon) $\in$ doc)	2
x_3	$\begin{cases} 1 & \text{if "no"} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$	1
x_4	count(1st and 2nd pronouns $\in$ doc)	3
x_5	$\begin{cases} 1 & \text{if "!"} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$	0
x_6	log(word count of doc)	$\ln(66) = 4.19$

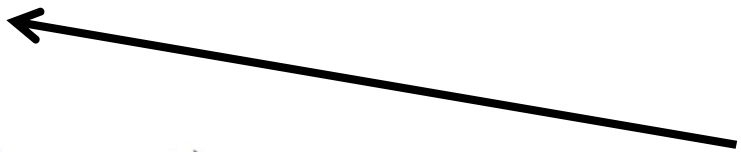
What is w·x + b ???

Suppose: w = [2.5, 5.0, 1.2, 0.5, 2.0, 0.7]
and b = 0.1

Sentiment Analysis: Example

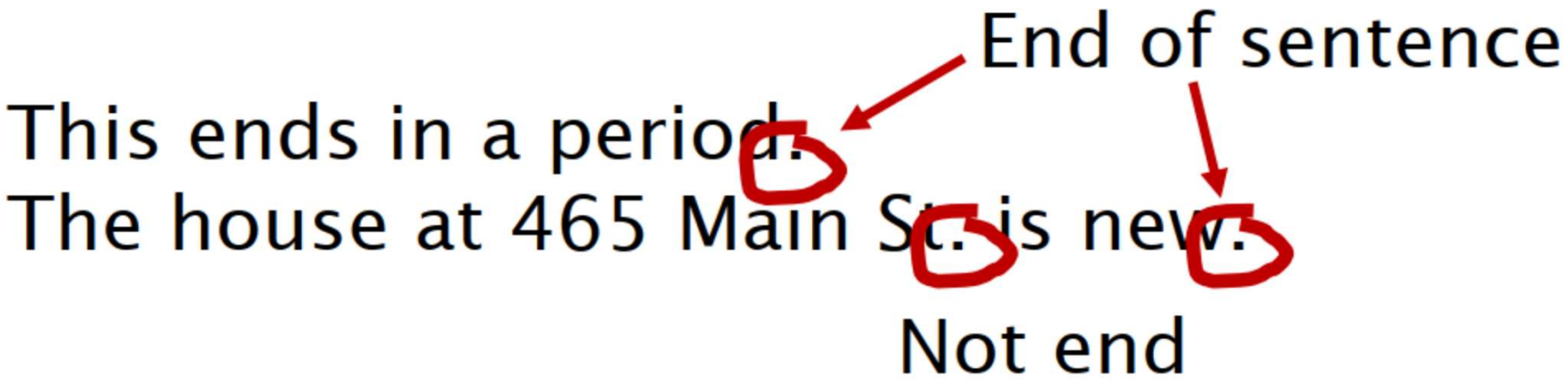
$$
\begin{aligned}
p(+|x) = P(Y=1|x) &= \sigma(w \cdot x + b) \\
&= \sigma([2.5, -5.0, -1.2, 0.5, 2.0, 0.7] \cdot [3, 2, 1, 3, 0, 4.19] + 0.1) \\
&= \sigma(.833) \\
&= 0.70
\end{aligned}
$$

$$
\begin{aligned}
p(-|x) = P(Y=0|x) &= 1 - \sigma(w \cdot x + b) \\
&= 0.30
\end{aligned}
$$



Positive sentiment

Period Disambiguation: Example



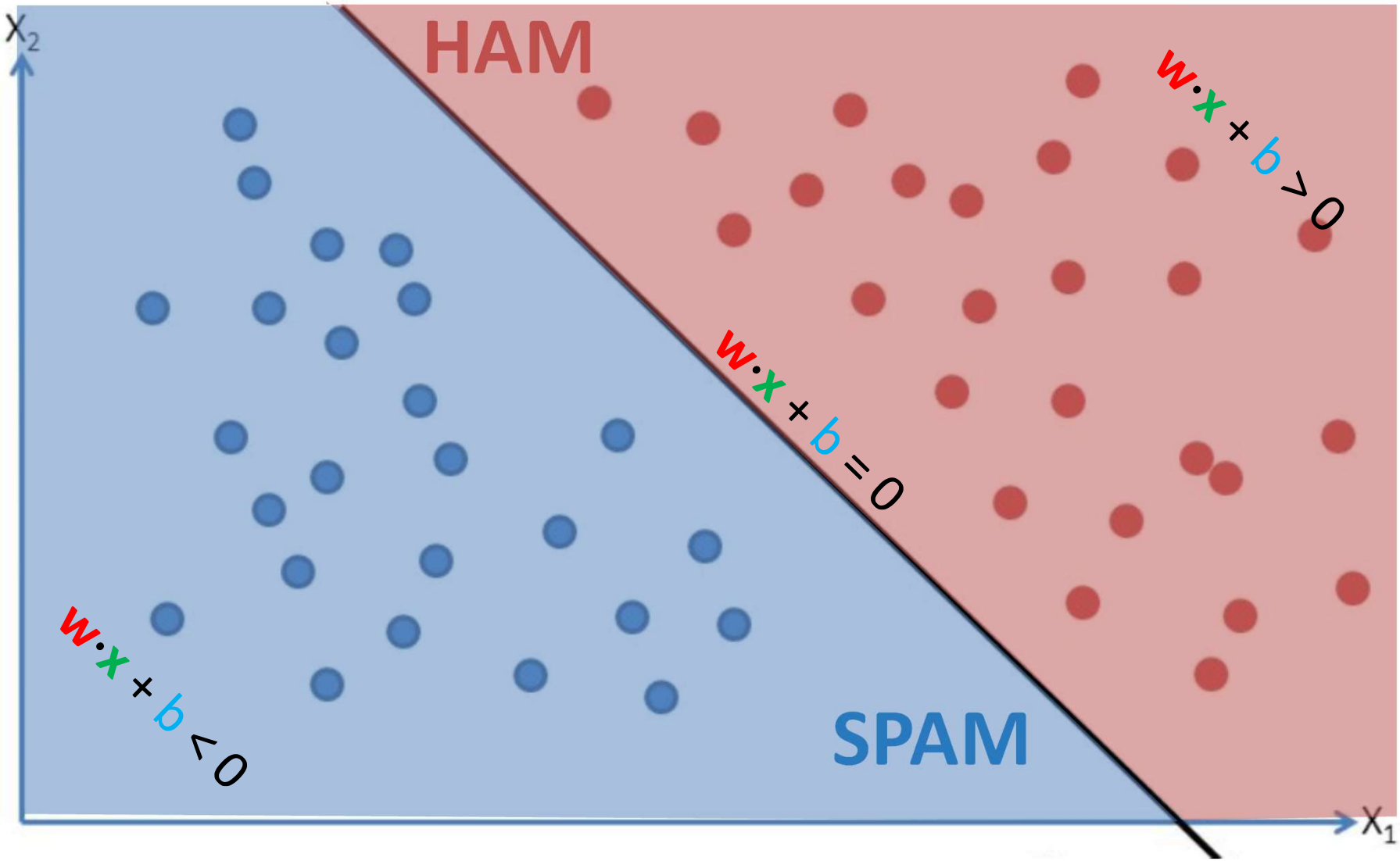
Feature vector x

$$x_1 = \begin{cases} 1 & \text{if } \text{“}Case(w_i) = \text{Lower”} \\ 0 & \text{otherwise} \end{cases}$$

$$x_2 = \begin{cases} 1 & \text{if } \text{“}w_i \in \text{AcronymDict”} \\ 0 & \text{otherwise} \end{cases}$$

$$x_3 = \begin{cases} 1 & \text{if } \text{“}w_i = \text{St. \& } Case(w_{i-1}) = \text{Cap”} \\ 0 & \text{otherwise} \end{cases}$$

How Do We Get w and b?

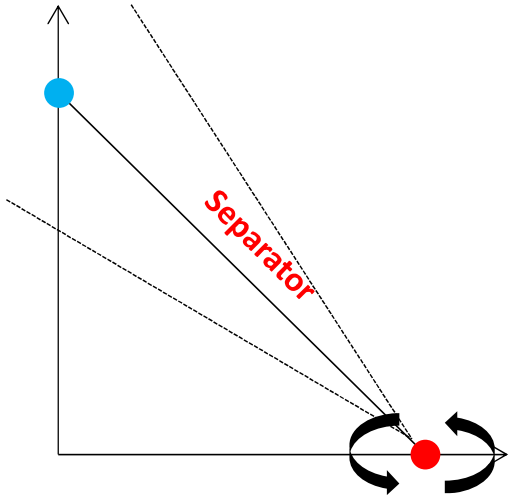


where: $\boldsymbol{w} = [w_1, w_2]$ **and** $\boldsymbol{x} = [x_1, x_2]$

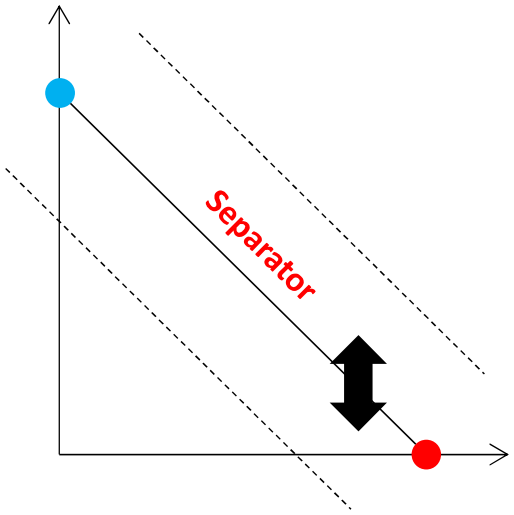
Updating Weights/Bias: The Idea

$$w_{BEFORE} \cdot x + b_{BEFORE} = 0$$

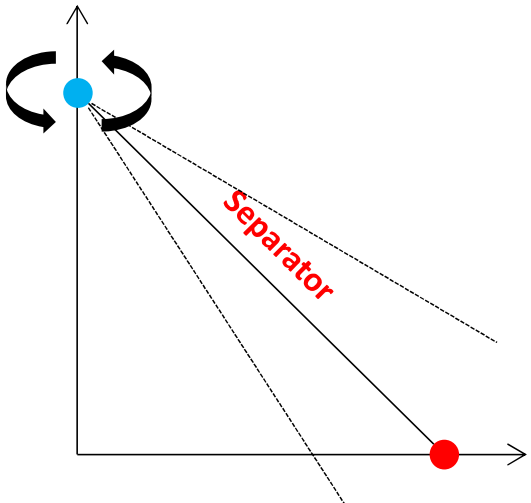
+ some combination of



rotation



translation

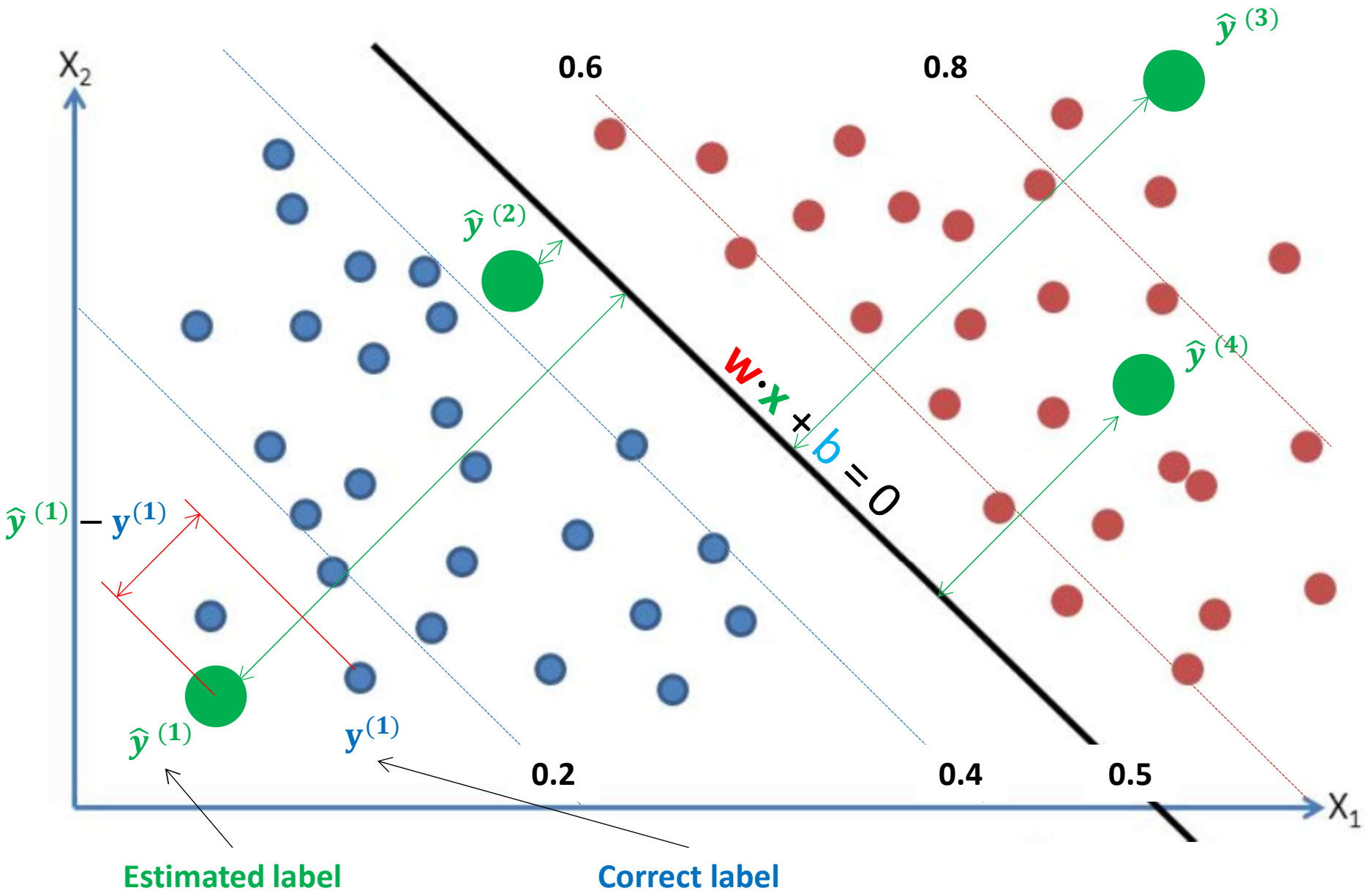


rotation

=

$$w_{AFTER} \cdot x + b_{AFTER} = 0$$

Classification Error: Class Estimates



Minimize Classification Error: the Idea

Supervised classification:

- We know the correct label y (either 0 or 1) for each **x**.
- But what the classifier produces is an estimate, $\hat{y}$

We want to set **w** and b to minimize the **distance** between our estimate $\hat{y}^{(i)}$ and the true $y^{(i)}$.

How?

- We need a distance estimator: a **loss function** or a **cost function**
- We need an optimization algorithm to update **w** and b to minimize the loss.

Minimize Error: Components

A Loss Function:

- Cross-Entropy Loss

Optimization Algortithm:

- Stochastic Gradient Descent

Classification Error: Distance

We want to know how far is the classifier output:

$$\hat{y} = \sigma(w \cdot x + b)$$

from the true output:

$$y \qquad [= \text{either } 0 \text{ or } 1]$$

We'll call this difference:

$L(\hat{y}, y)$

how much $\hat{y}$ (estimate) differs from the true y

Cross-Entropy Loss: Intuition

It is a case of conditional maximum likelihood estimation

We choose the parameters w, b that maximize

- **the log probability**
- **of the true *y* labels in the training data**
- **given the observations x**

Cross-Entropy Loss: Calculation

Goal: **maximize** probability of the correct label P(*y*|x)

Since there are only 2 discrete outcomes (0 or 1) we can express the probability P(*y*|x) from our classifier (the thing we want to maximize) as

$$p(y|x) = \hat{y}^{y}(1-\hat{y})^{1-y}$$

noting that:

if y = 1, this simplifies to $\hat{y}$

if y = 0, this simplifies to 1- $\hat{y}$

Cross-Entropy Loss: Calculation

Goal: **maximize** probability of the correct label P(*y*|x)

To make things simpler, we can take a logarithm of

$$p(y|x) = \hat{y}^y (1-\hat{y})^{1-y}$$

to get:

$$\begin{aligned} \log p(y|x) &= \log \left[\hat{y}^y (1-\hat{y})^{1-y}\right] \\ &= y \log \hat{y} + (1-y) \log(1-\hat{y}) \end{aligned}$$

It works, because: Whatever values maximize log(P(*y*|x))
will also maximize P(*y*|x)

Cross-Entropy Loss: Calculation

Goal: **maximize** probability of the correct label P(*y*|x)

$$\begin{aligned}\log p(y|x) &= \log\left[\hat{y}^y(1-\hat{y})^{1-y}\right] \\ &= y\log\hat{y}+(1-y)\log(1-\hat{y})\end{aligned}$$

Goal: **minimize** error / cross-entropy loss

$$L_{\text{CE}}(\hat{y},y) = -\log p(y|x) = -[y\log\hat{y}+(1-y)\log(1-\hat{y})]$$

which yields:

$$L_{\text{CE}}(\hat{y},y) = -[y\log\sigma(w\cdot x+b)+(1-y)\log(1-\sigma(w\cdot x+b))]$$

Minimizing Loss

The loss function is parameterized by weights

$$\theta = (\mathbf{w}, b)$$

Let's represent $\hat{y}$ as $f(\mathbf{x}; \theta)$ to make the dependence on θ more obvious

We want the weights that minimize the loss, averaged over all examples:

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}} \frac{1}{m} \sum_{i=1}^{m} L_{\mathrm{CE}}(f(x^{(i)}; \theta), y^{(i)})$$

Minimizing Loss

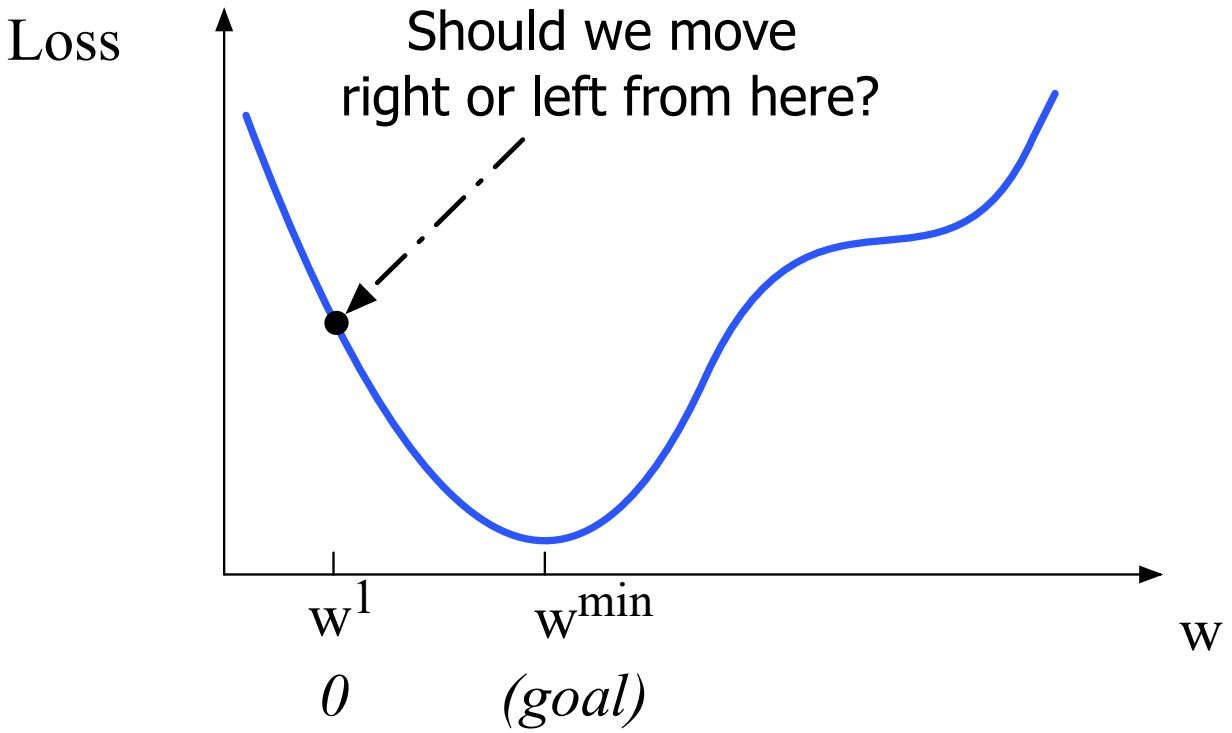
For logistic regression, loss function is convex

- **a convex function has just one minimum**
- **gradient descent starting from any point is guaranteed to find the minimum**

Minimizing Loss

Q: Given current **w**, should we make it bigger or smaller?

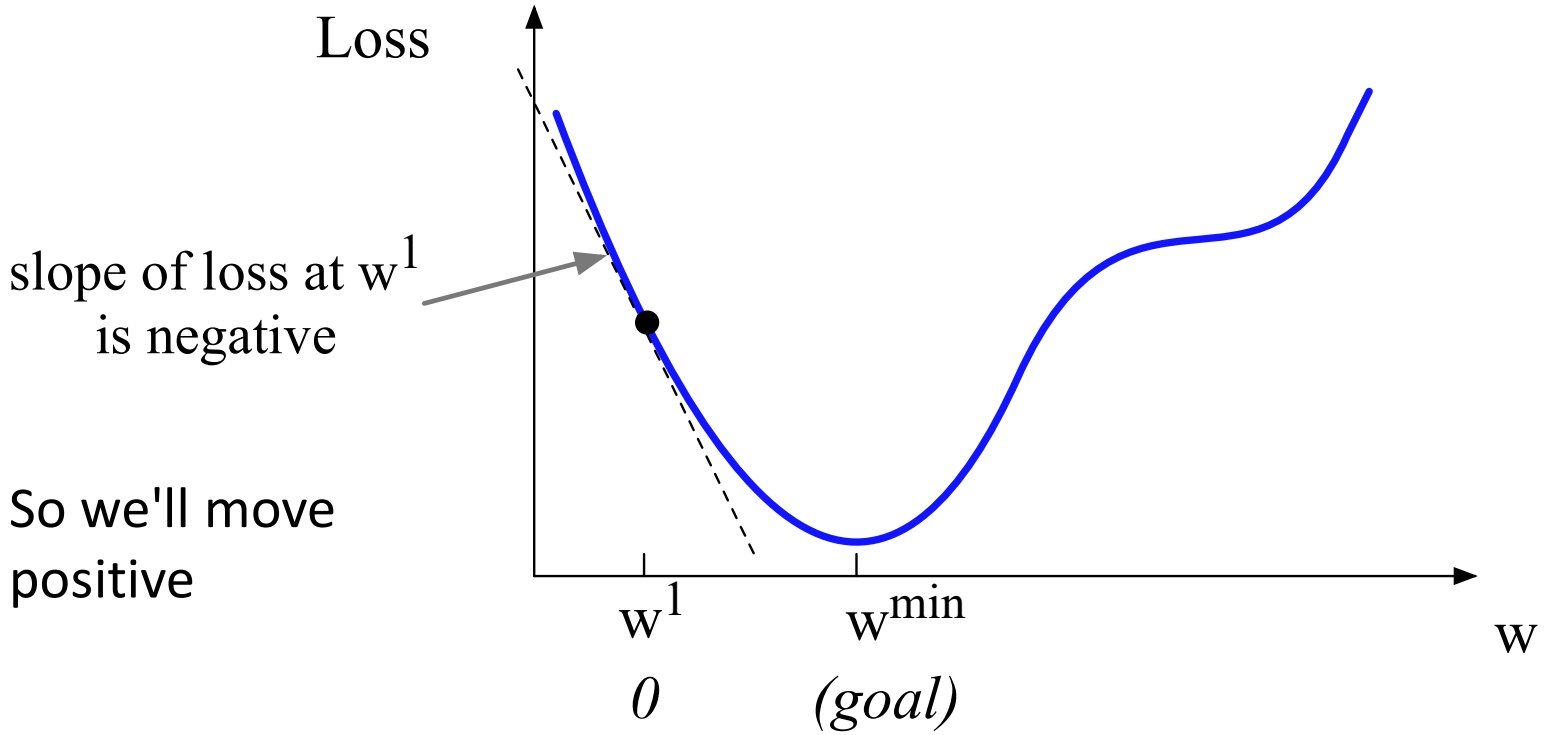
A: Move **w** in the reverse direction from the slope of the loss function



Minimizing Loss

Q: Given current **w**, should we make it bigger or smaller?

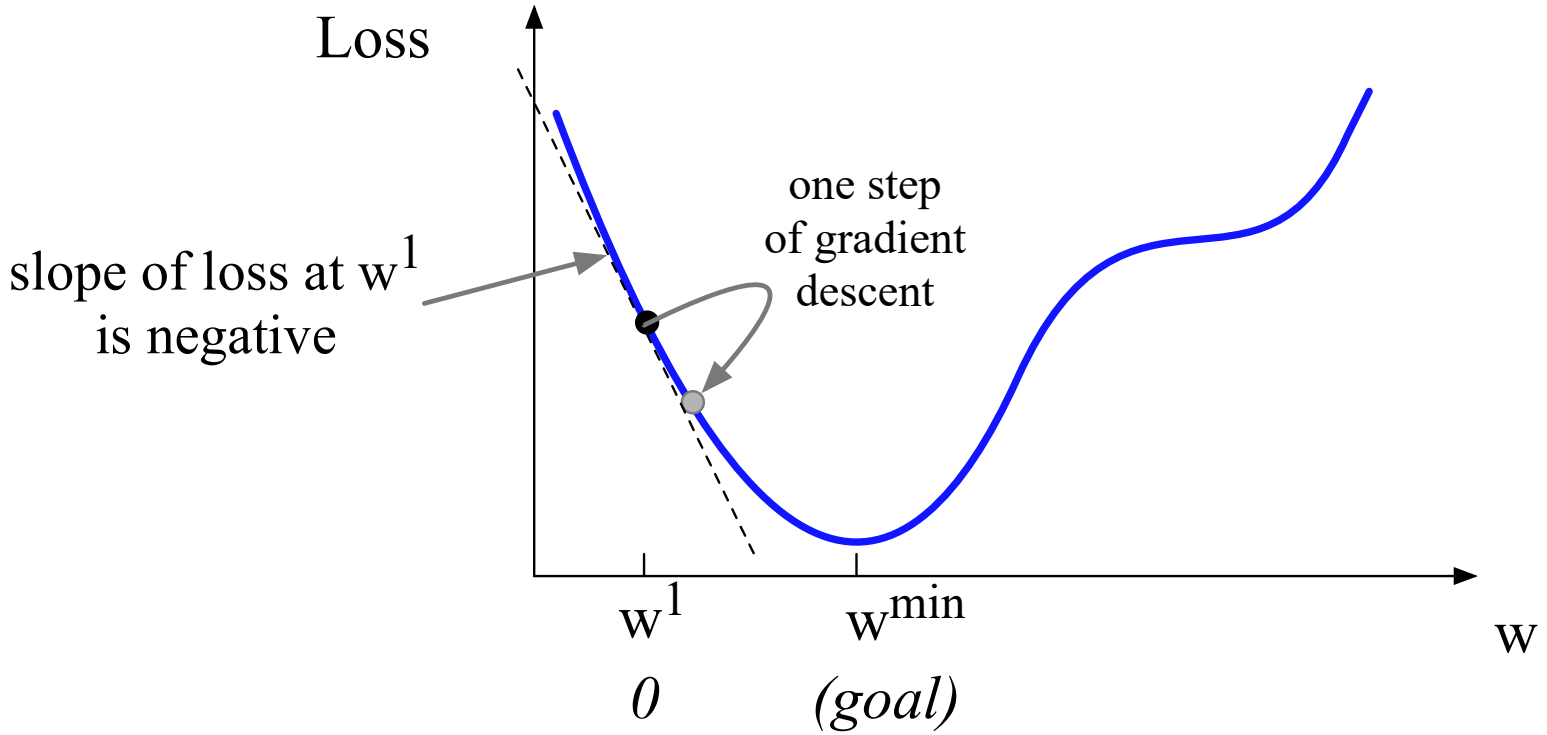
A: Move **w** in the reverse direction from the slope of the loss function



Minimizing Loss

Q: Given current **w**, should we make it bigger or smaller?

A: Move **w** in the reverse direction from the slope of the loss function



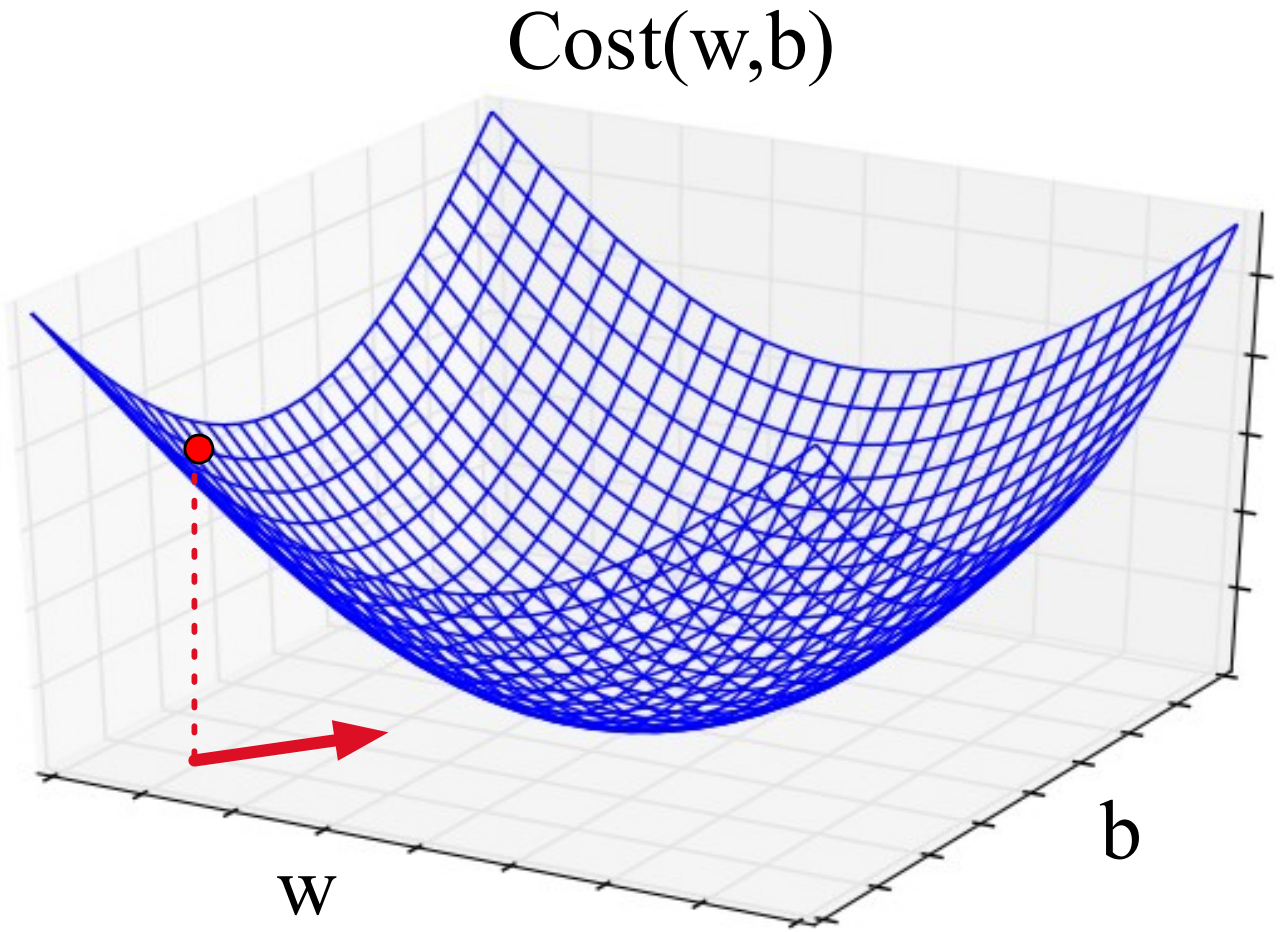
Gradients and Gradient Descent

The **gradient** of a function of many variables is a vector pointing in the direction of the greatest increase in a function.

Gradient Descent: Find the gradient of the loss function at the current point and move in the **opposite** direction.

Gradients: Visualized

Visualizing the gradient vector at the **red** point



Gradients and Learning Rate

- The value of the gradient (slope in our example) $\frac{d}{dw} L(f(x;w), y)$ weighted by a **learning rate** η

- Higher learning rate means move **w** faster

$$w^{t+1} = w^t - \eta \frac{d}{dw} L(f(x;w), y)$$

Gradients and Weights/Bias Update

Let's represent $\hat{y}$ as f (**x**; θ) to make the dependence on θ more obvious:

$$\nabla_\theta L(f(x;\theta),y)) = \begin{bmatrix} \frac{\partial}{\partial w_1} L(f(x;\theta),y) \\ \frac{\partial}{\partial w_2} L(f(x;\theta),y) \\ \vdots \\ \frac{\partial}{\partial w_n} L(f(x;\theta),y) \end{bmatrix}$$

and the final equation is for updating θ:

$$\theta_{t+1} = \theta_t - \eta \nabla L(f(x;\theta),y)$$

Stochastic Gradient Descent

function STOCHASTIC GRADIENT DESCENT($L()$, $f()$, x, y) **returns** θ

where: L is the loss function
f is a function parameterized by θ
x is the set of training inputs $x^{(1)}$, $x^{(2)}$, ..., $x^{(m)}$
y is the set of training outputs (labels) $y^{(1)}$, $y^{(2)}$, ..., $y^{(m)}$

$\theta \leftarrow 0$

repeat til done

For each training tuple $(x^{(i)}, y^{(i)})$ (in random order)

1. Optional (for reporting):
   Compute $\hat{y}^{(i)} = f(x^{(i)}; \theta)$ # What is our estimated output $\hat{y}$?
   Compute the loss $L(\hat{y}^{(i)}, y^{(i)})$ # How far off is $\hat{y}^{(i)}$) from the true output $y^{(i)}$?
2. $g \leftarrow \nabla_\theta L(f(x^{(i)}; \theta), y^{(i)})$ # How should we move θ to maximize loss?
3. $\theta \leftarrow \theta - \eta\, g$ # Go the other way instead

return θ

Hyperparameters

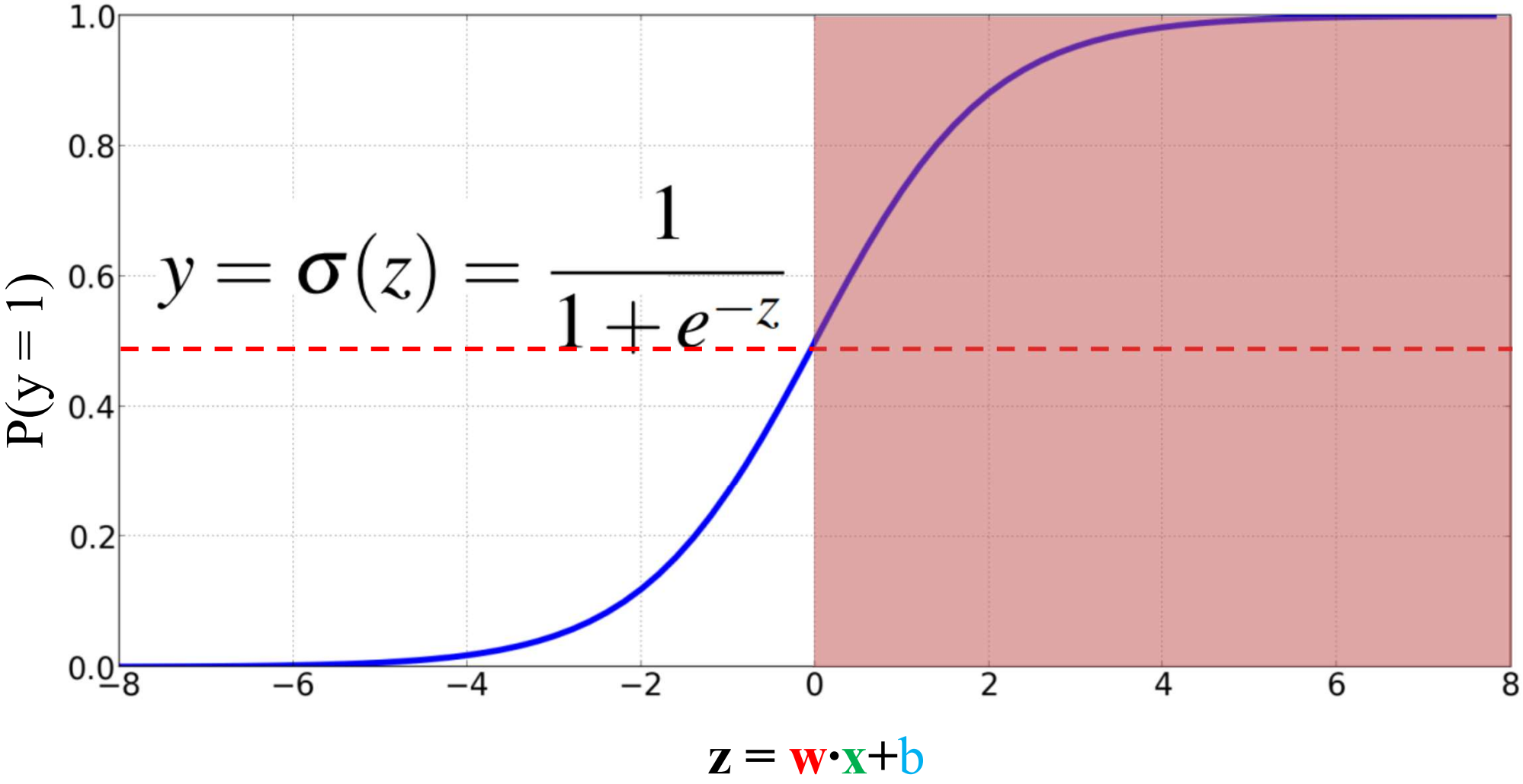
The learning rate η is a **hyperparameter**

- too high: the learner will take big steps and overshoot
- too low: the learner will take too long

Hyperparameters:

- Briefly, a special kind of parameter for an ML model
- Instead of being learned by algorithm from supervision (like regular parameters), they are chosen by algorithm designer.

Logistic Regression Classifier



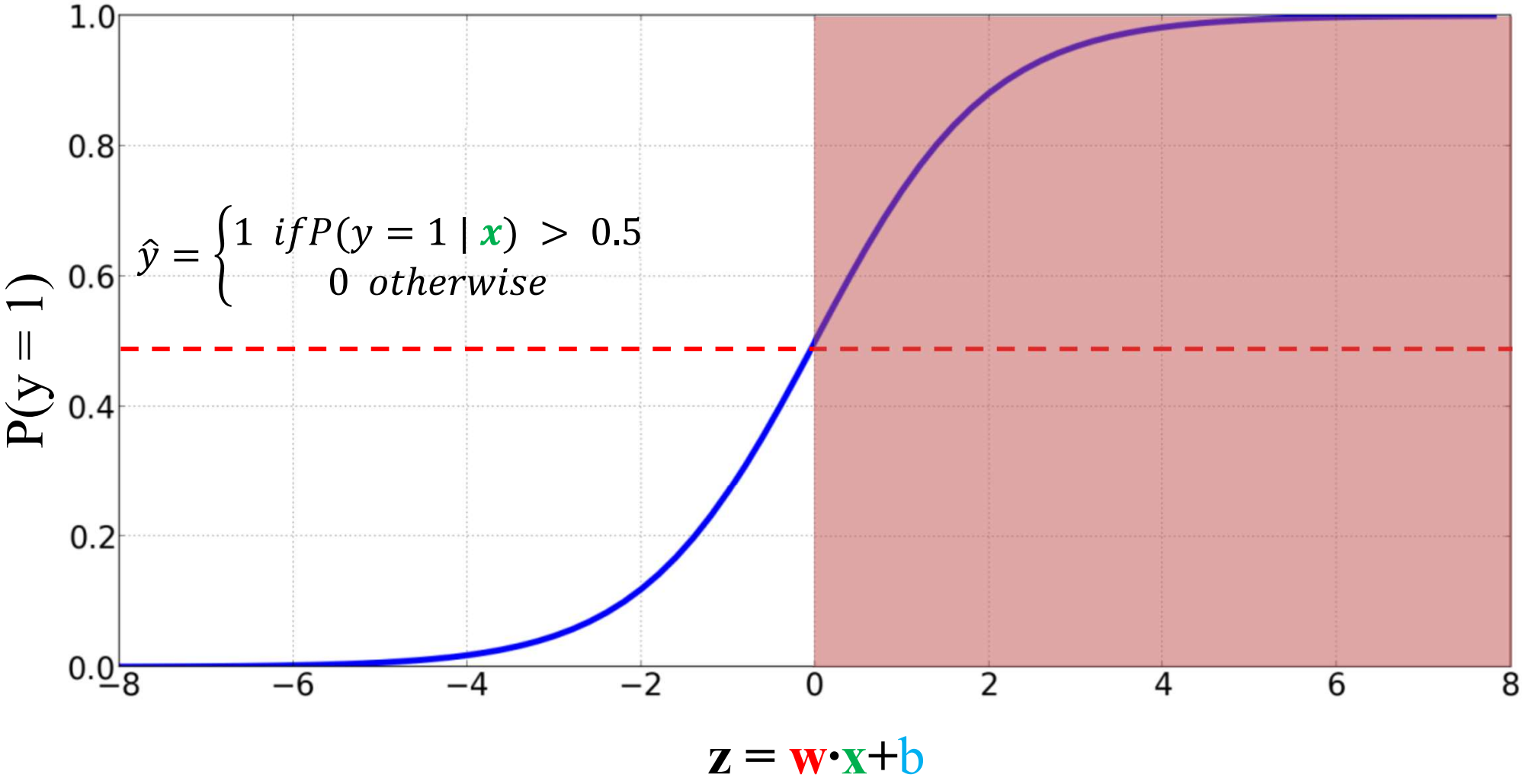
Probabilities → Classification

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

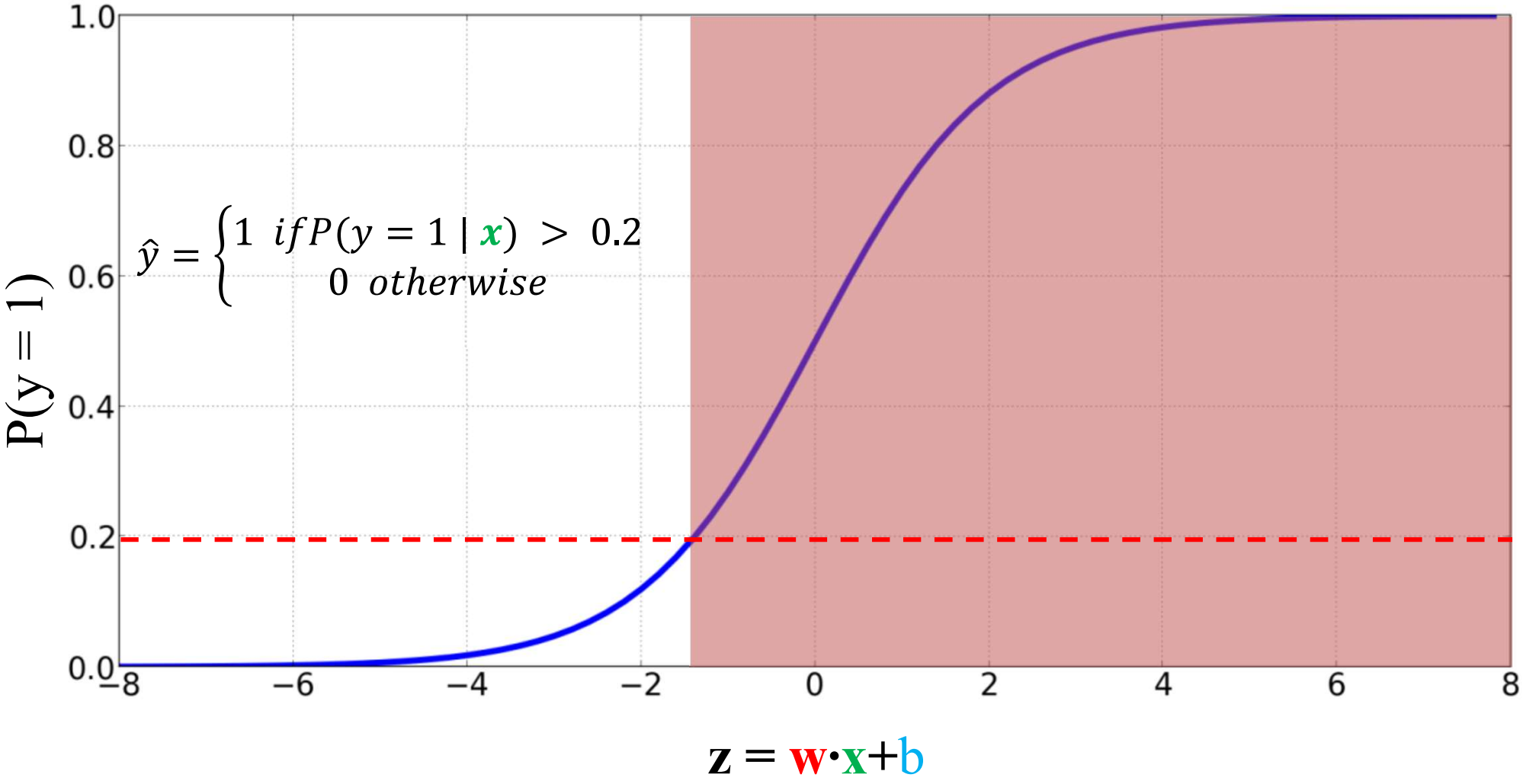
$$\hat{y} = \begin{cases} 1 & \text{if } P(y=1|x) > 0.5 \\ 0 & \text{otherwise} \end{cases}$$

Where 0.5 is the decision boundary

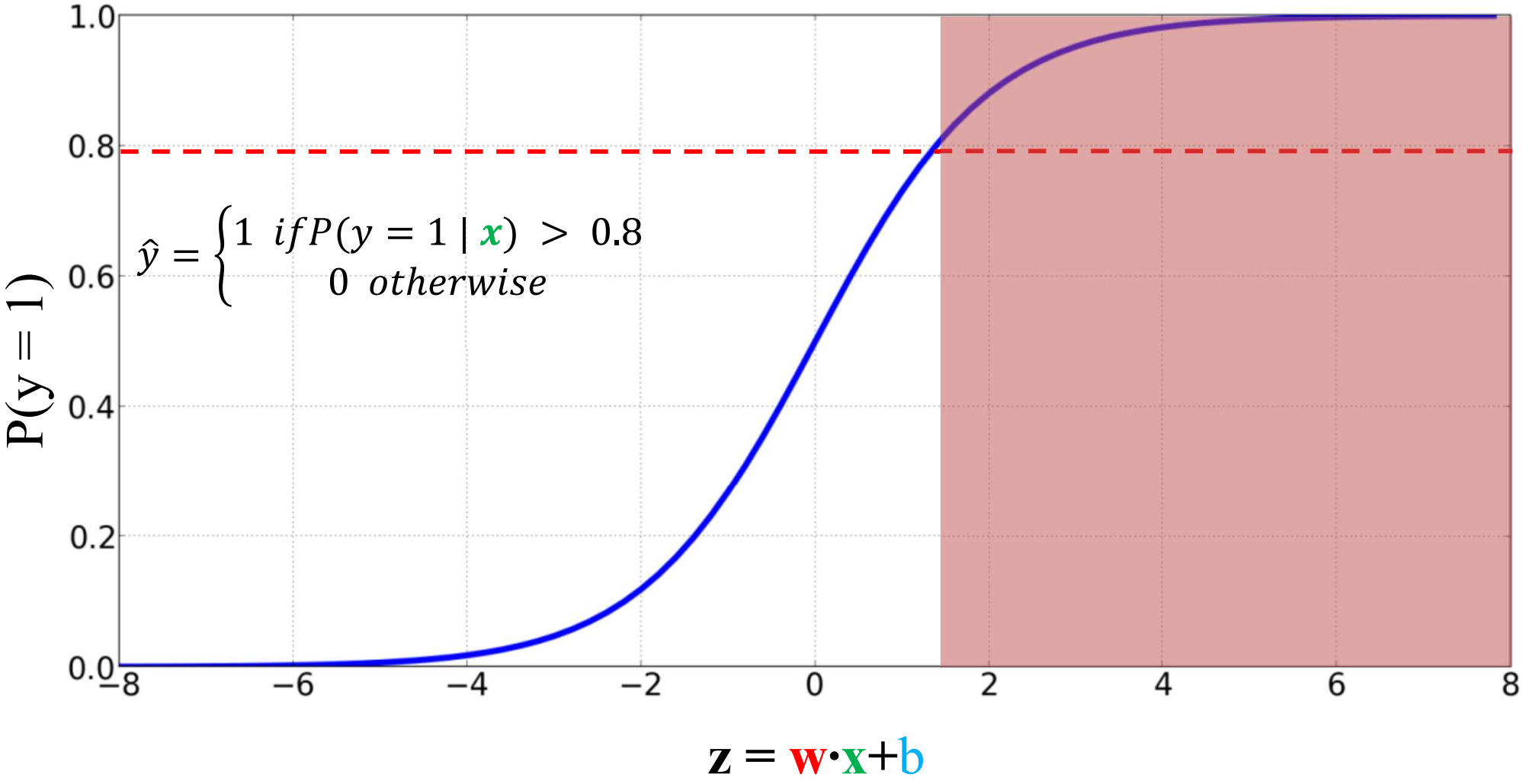
Logistic Regression: Threshold 0.5



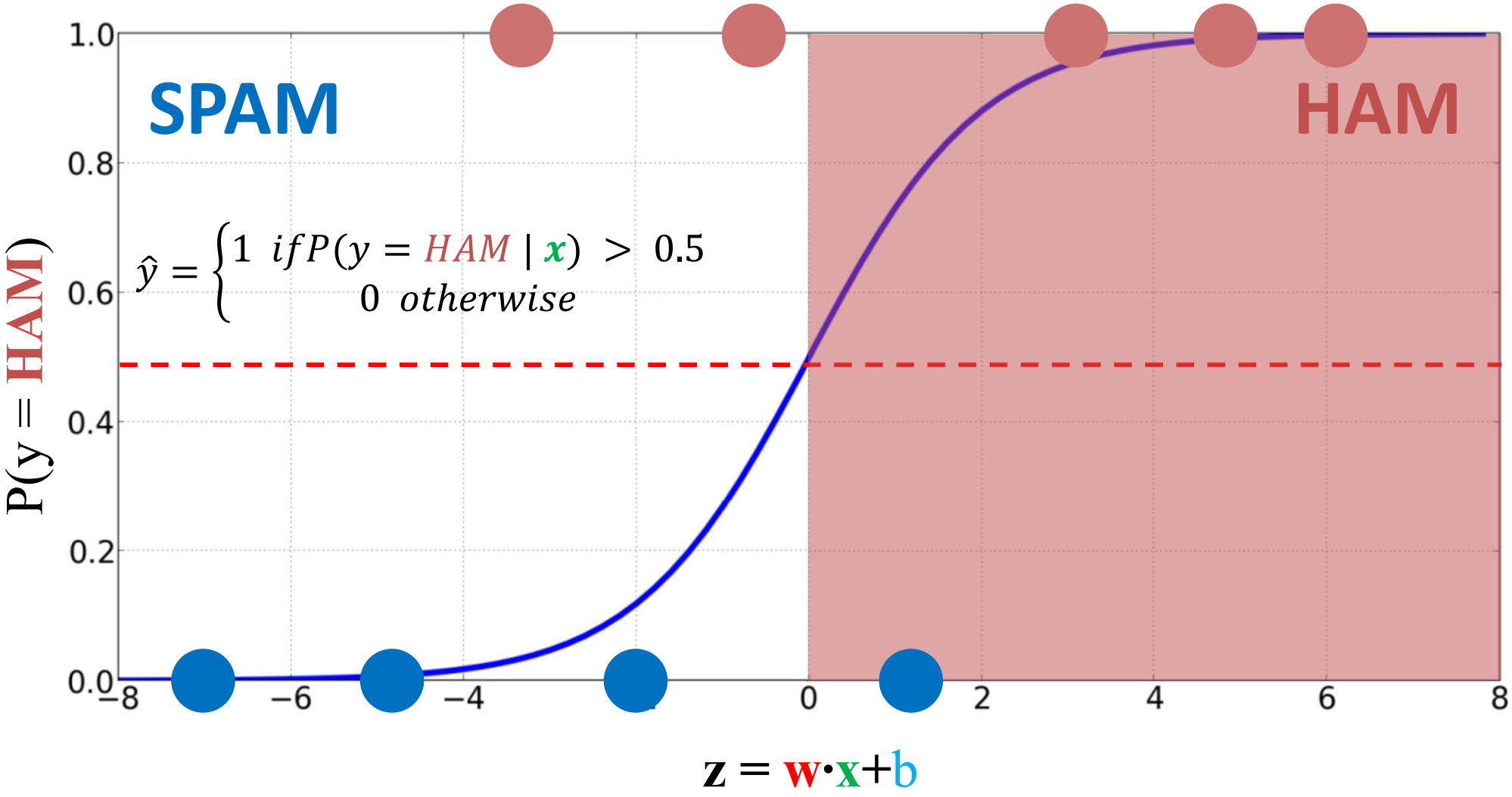
Logistic Regression: Threshold 0.2



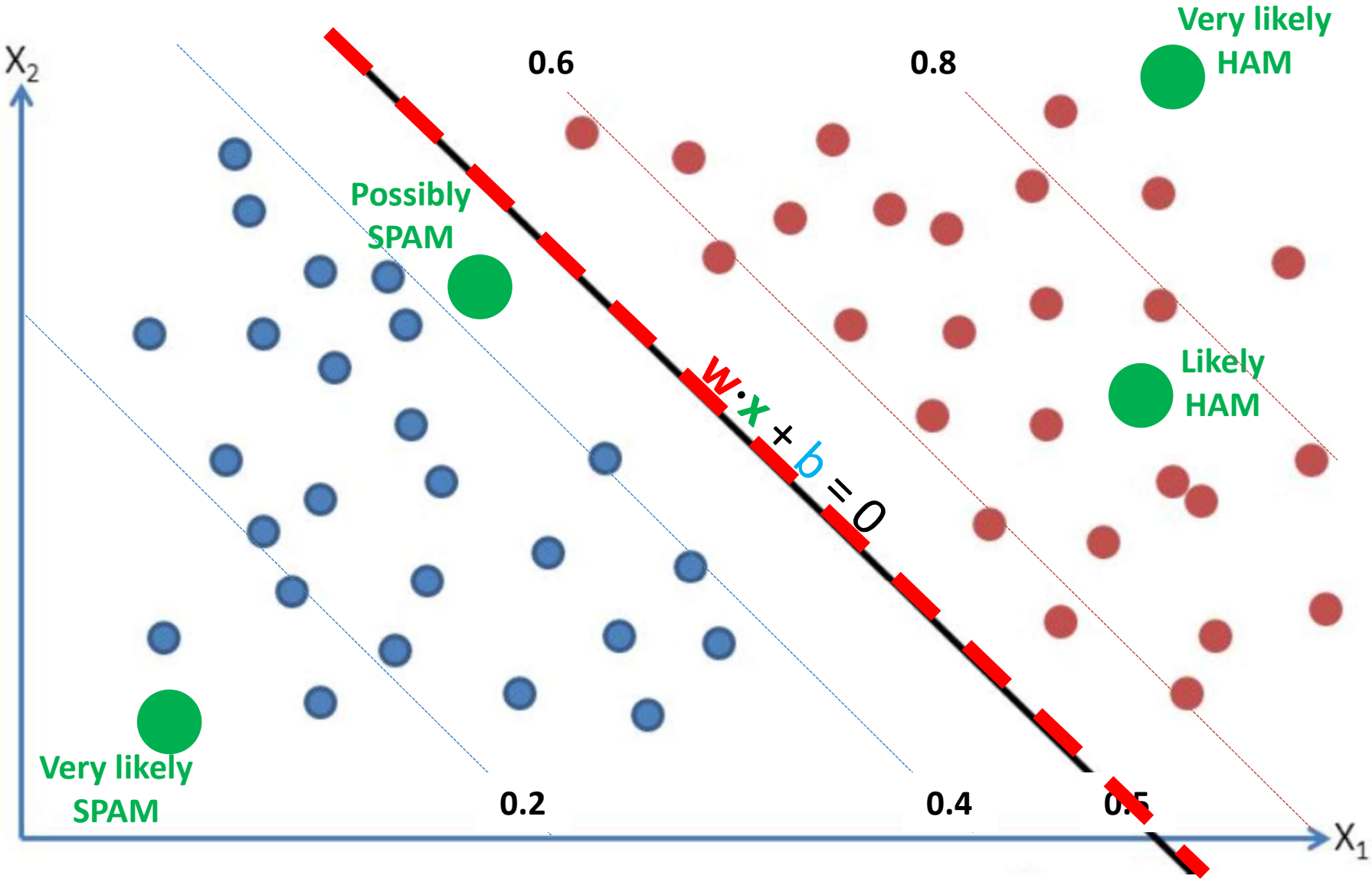
Logistic Regression: Threshold 0.8



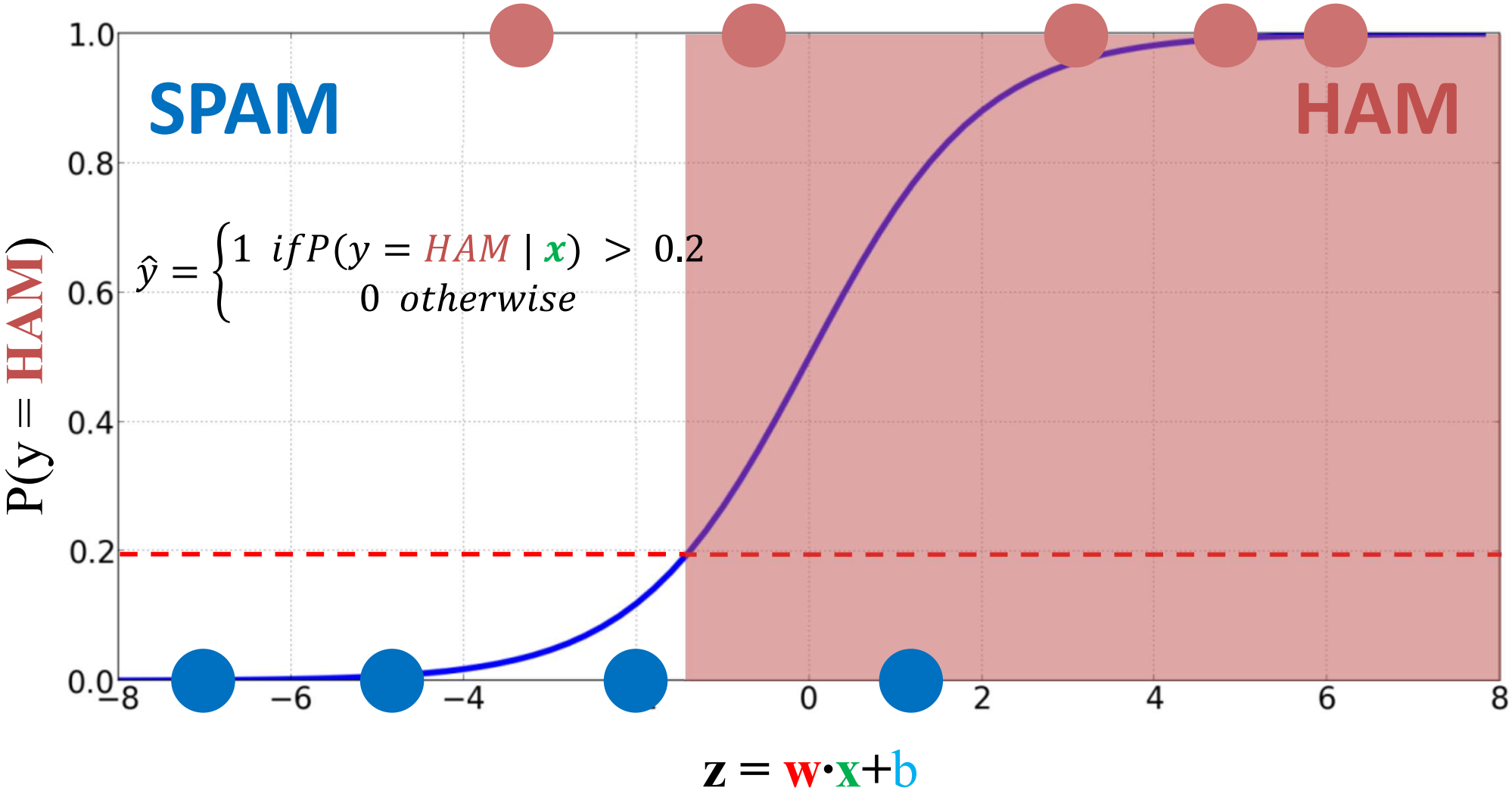
Logistic Regression: Threshold 0.5



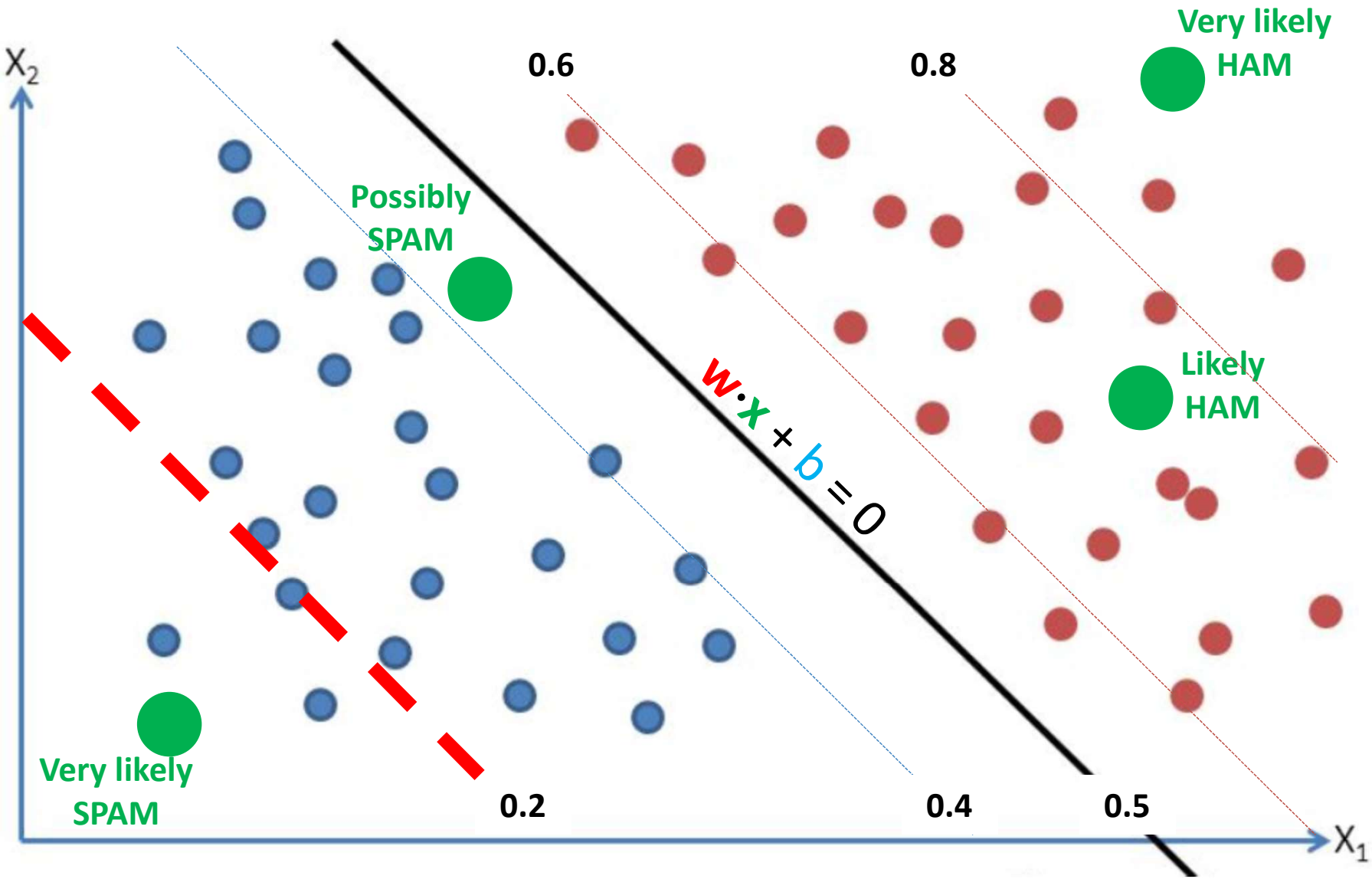
Text Classification: Separator



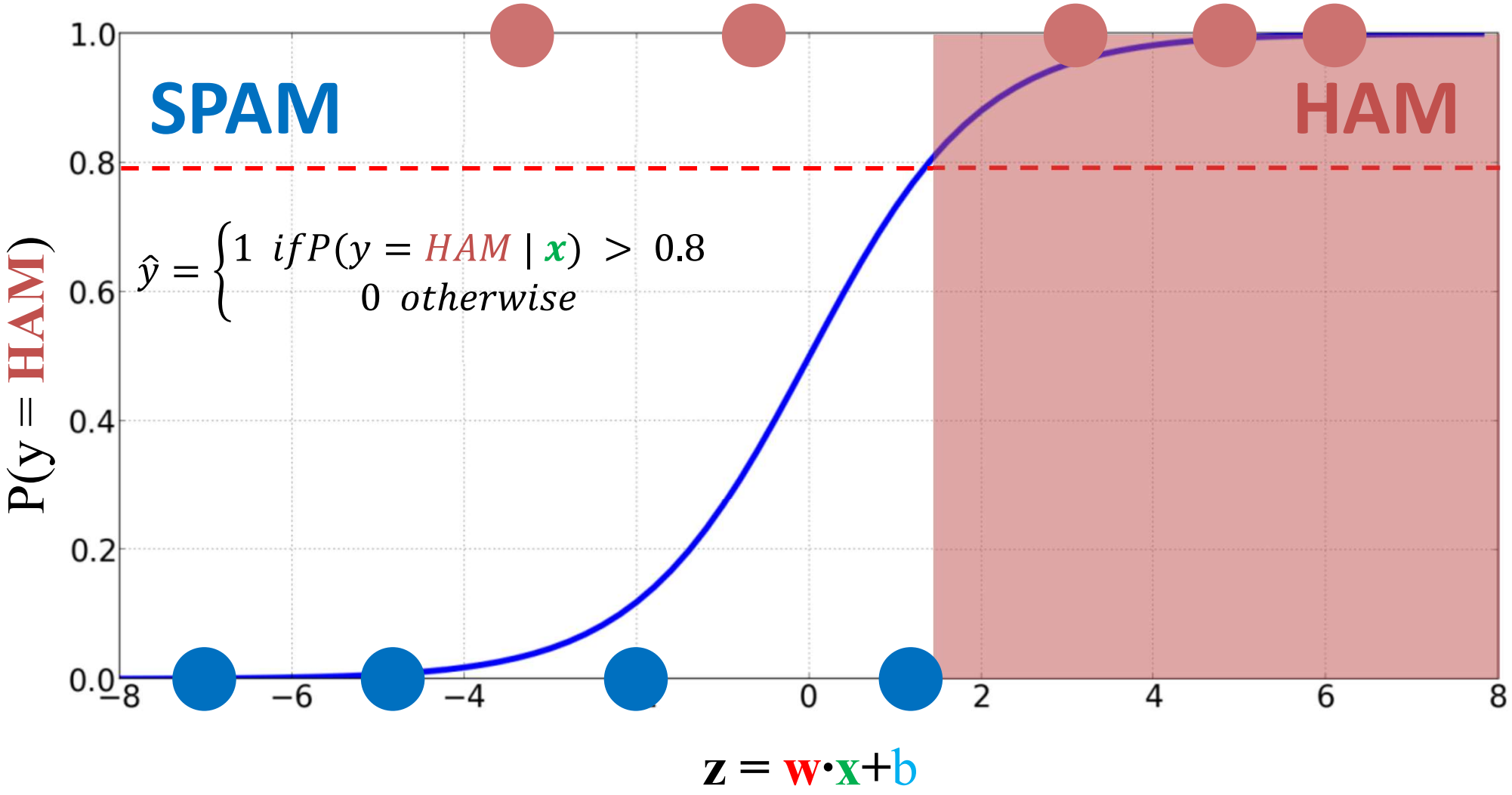
Logistic Regression: Threshold 0.2



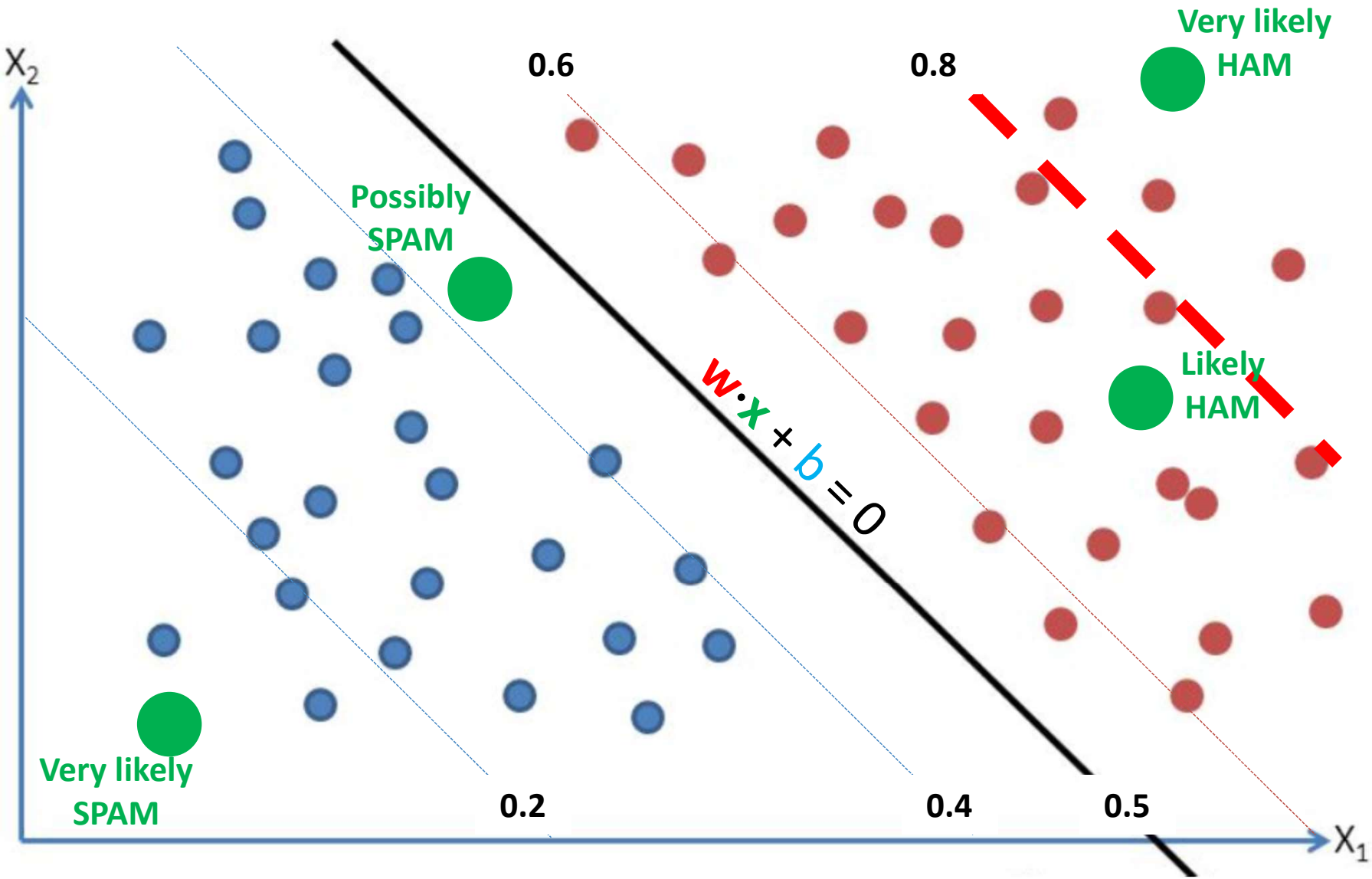
Text Classification: Separator



Logistic Regression: Threshold 0.8



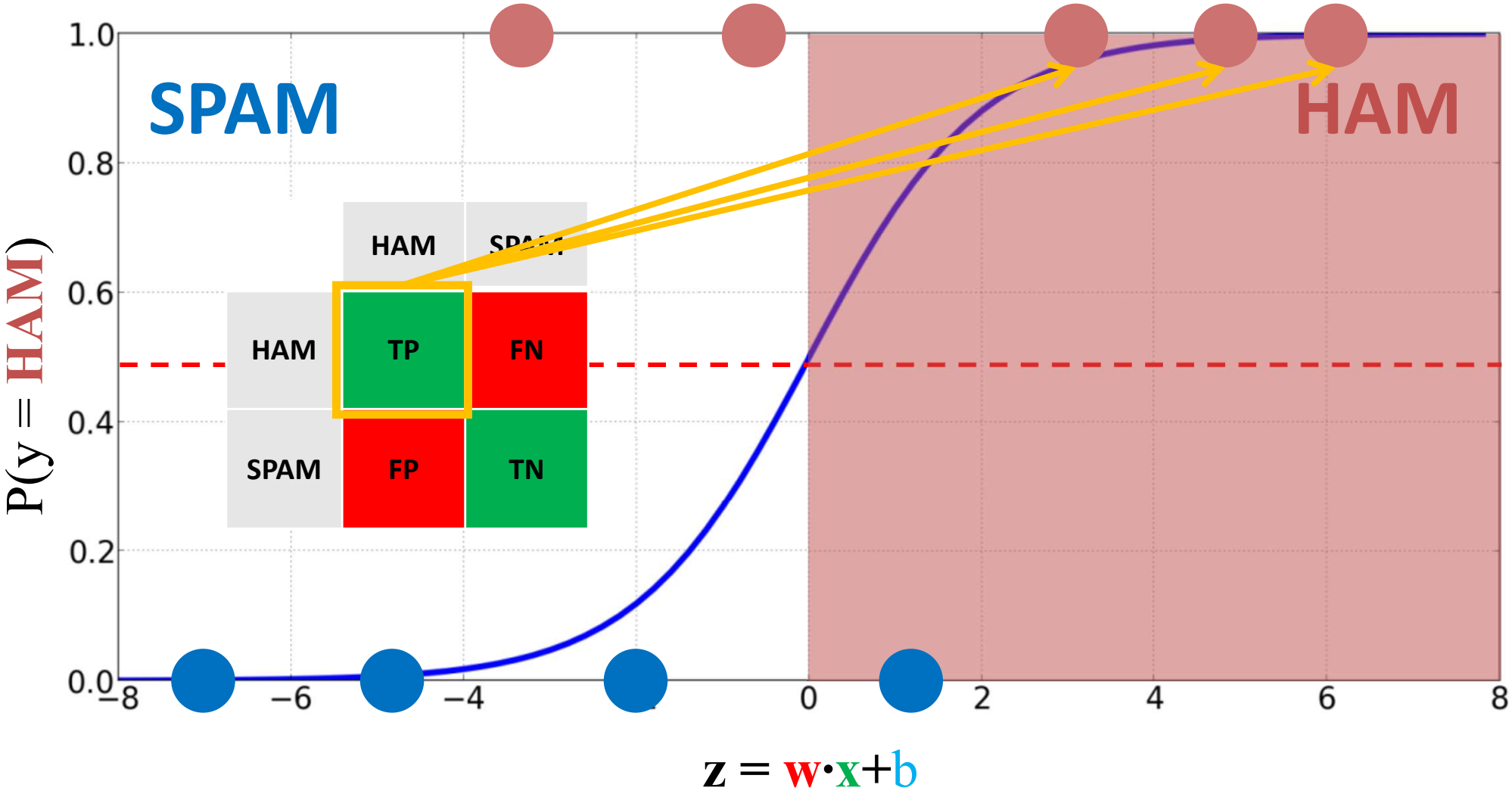
Text Classification: Separator



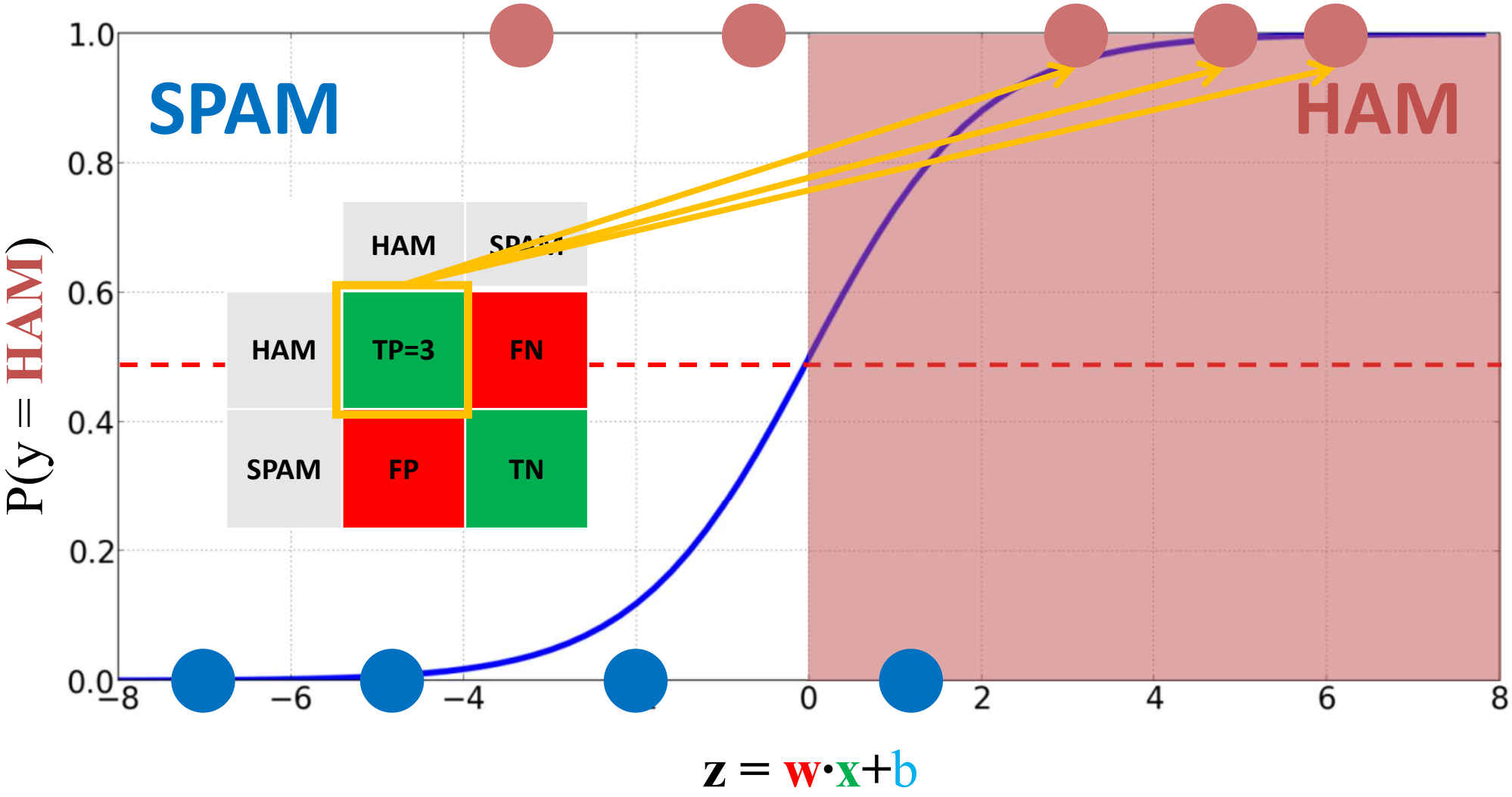
Classifier Evaluation: Confusion Matrix

		Predicted class		
		Positive	Negative	
Actual class	Positive	True Positive (TP)	False Negative (FN) Type II Error	Sensitivity (Recall) $\frac{TP}{TP+FN}$
	Negative	False Positive (FP) Type I Error	True Negative (TN)	Specificity $\frac{TN}{TN+FP}$
		Precision $\frac{TP}{TP+FP}$	Negative Predictive Value $\frac{TN}{TN+FN}$	Accuracy $\frac{TP+TN}{TP+TN+FP+FN}$

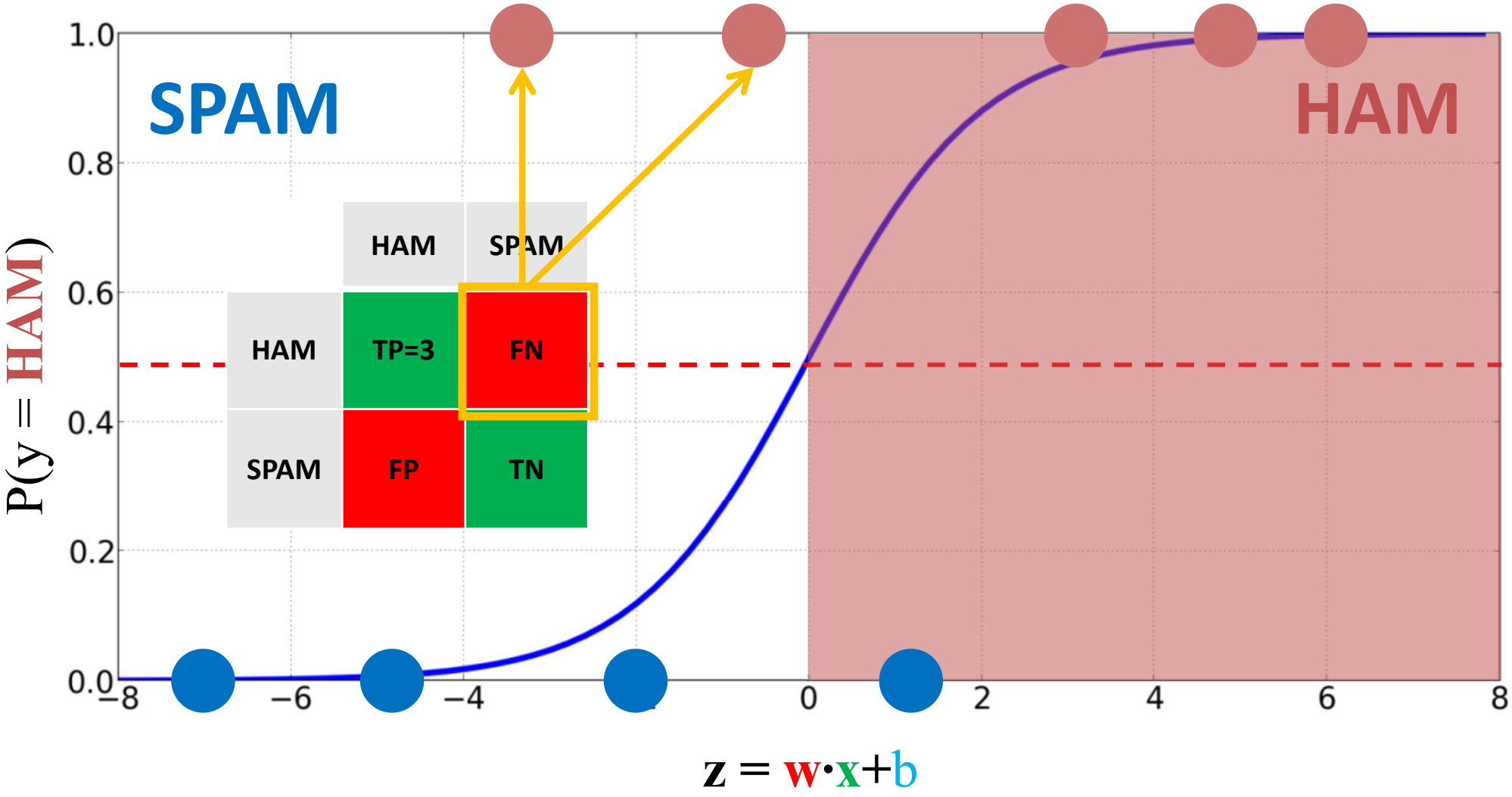
Logistic Regression: Threshold 0.5



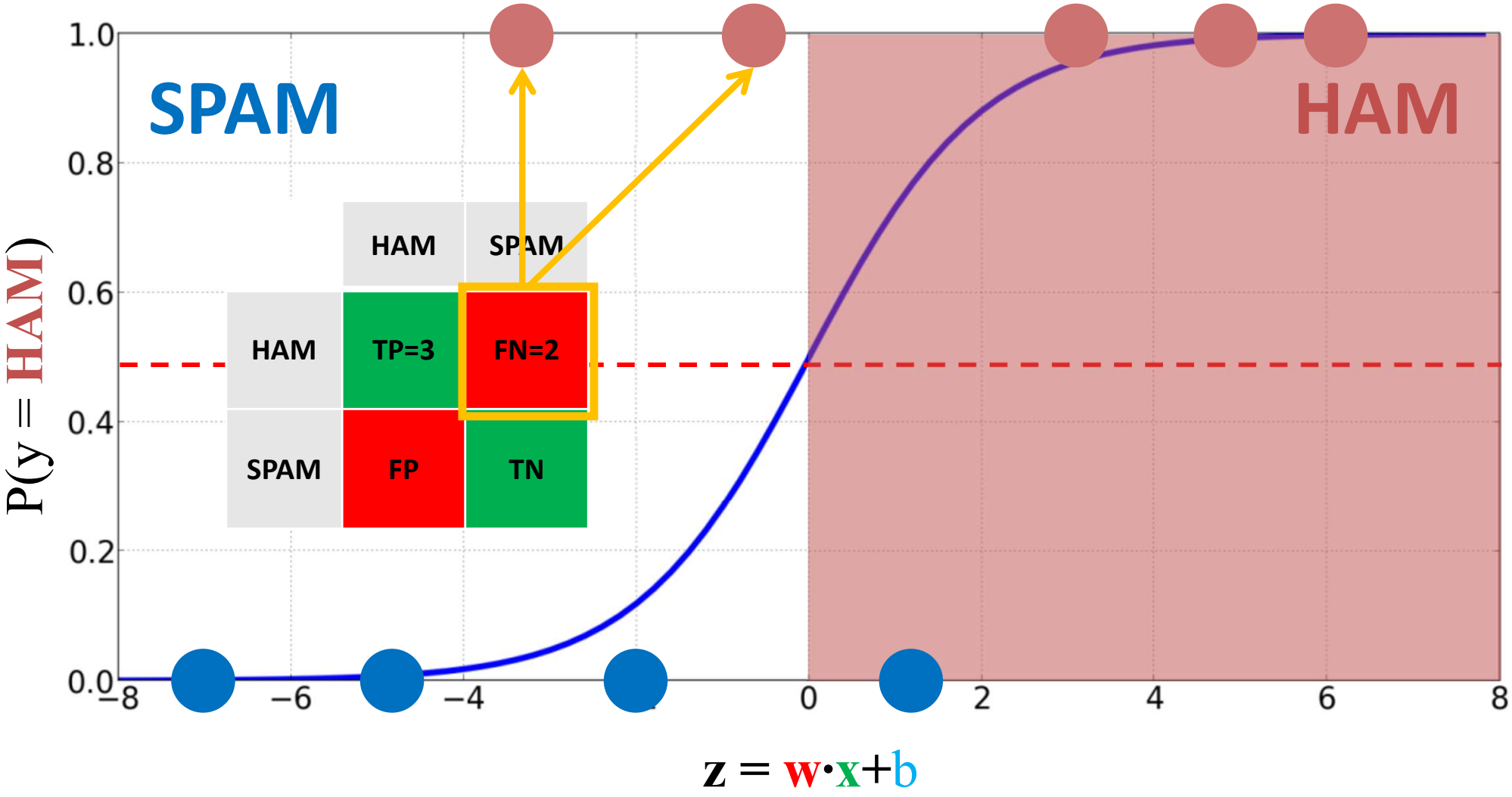
Logistic Regression: Threshold 0.5



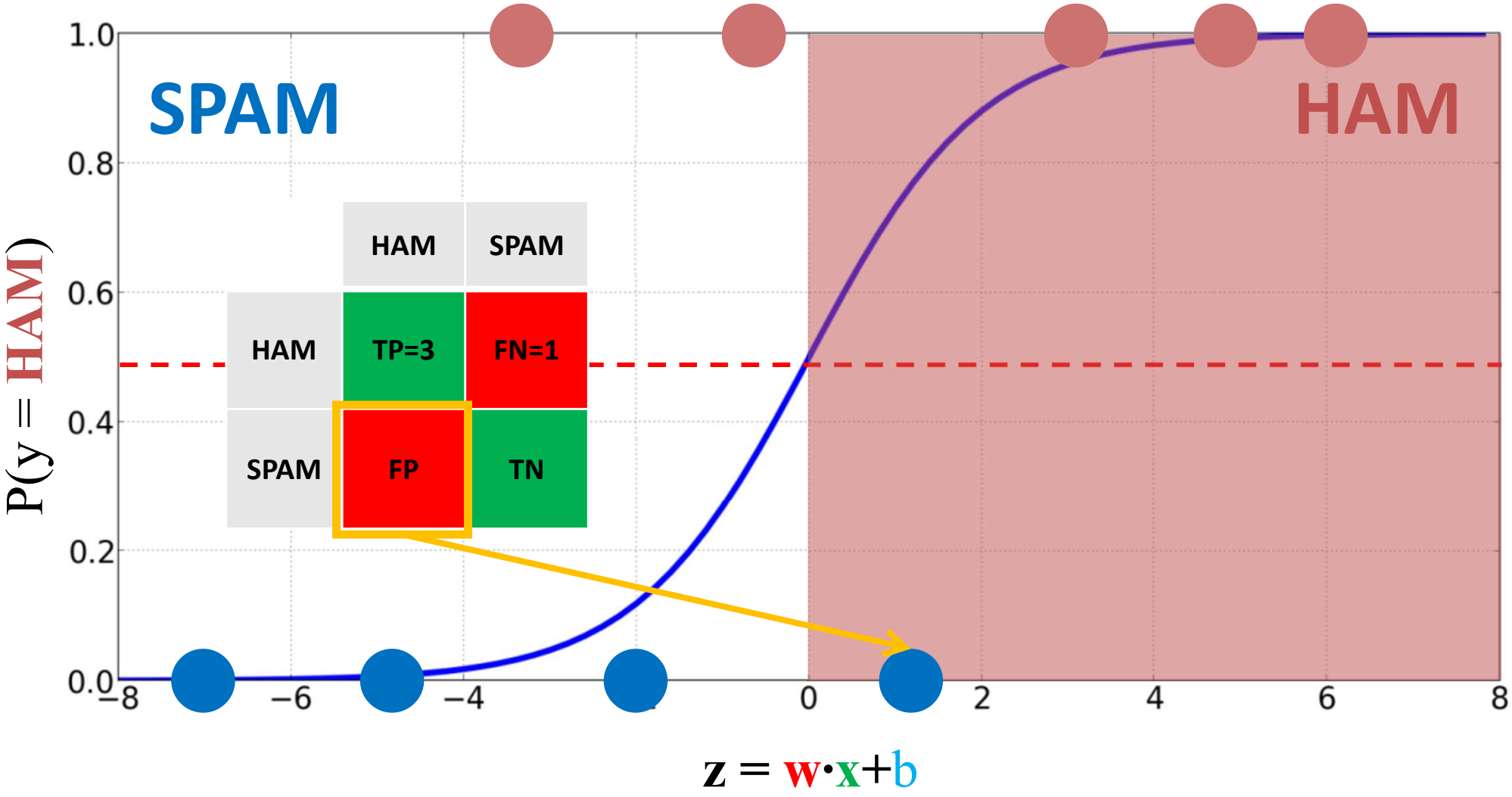
Logistic Regression: Threshold 0.5



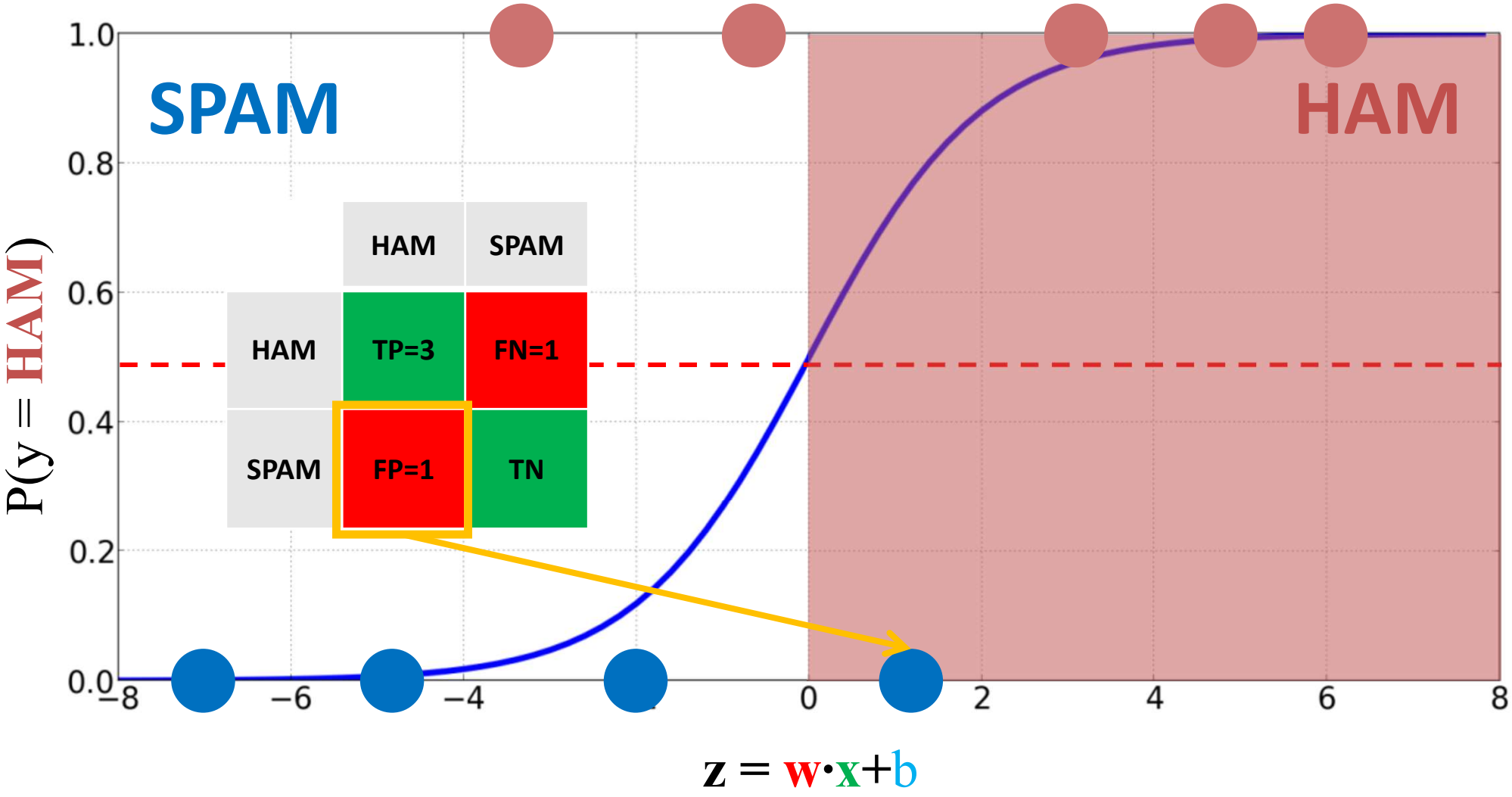
Logistic Regression: Threshold 0.5



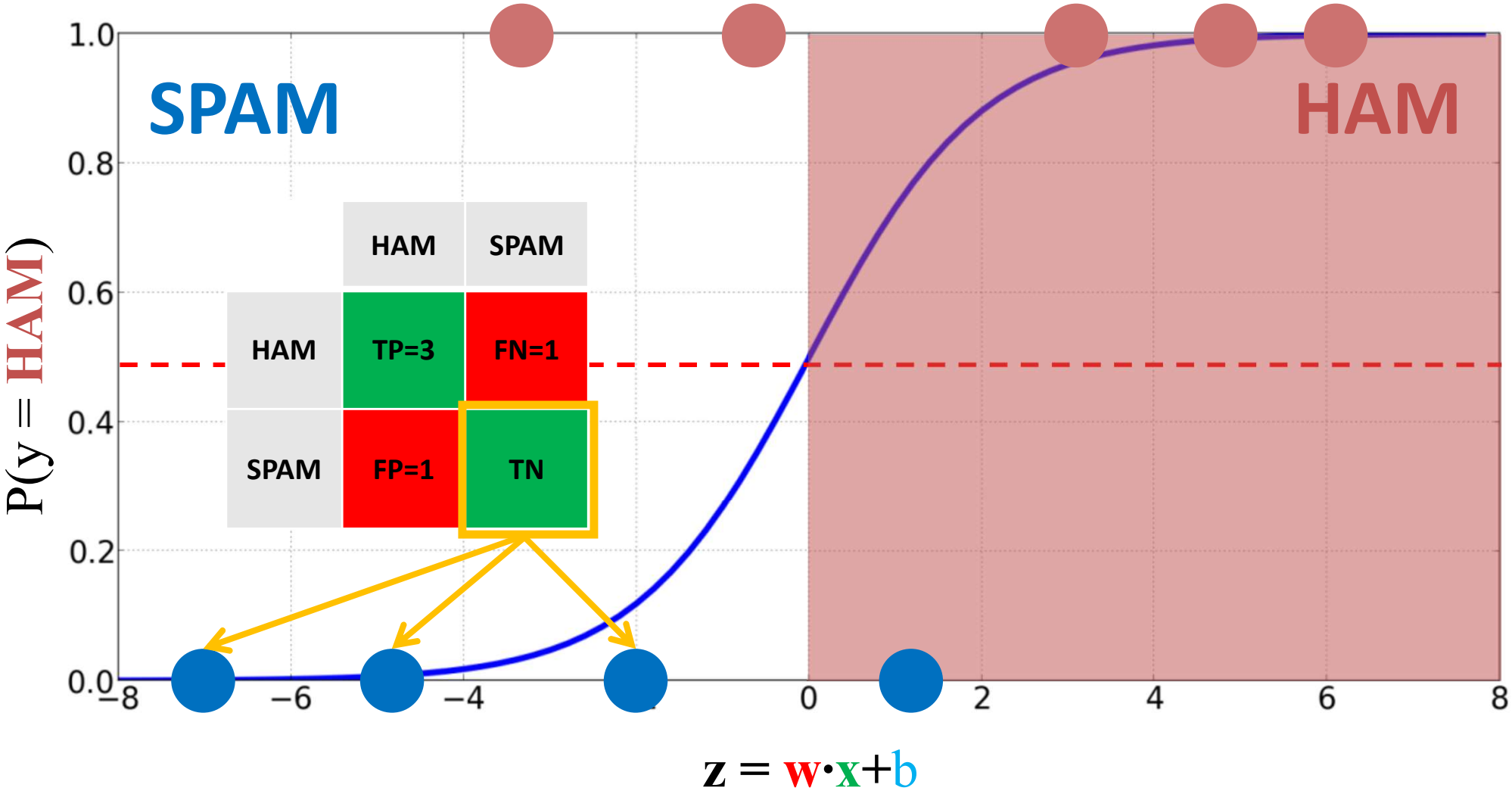
Logistic Regression: Threshold 0.5



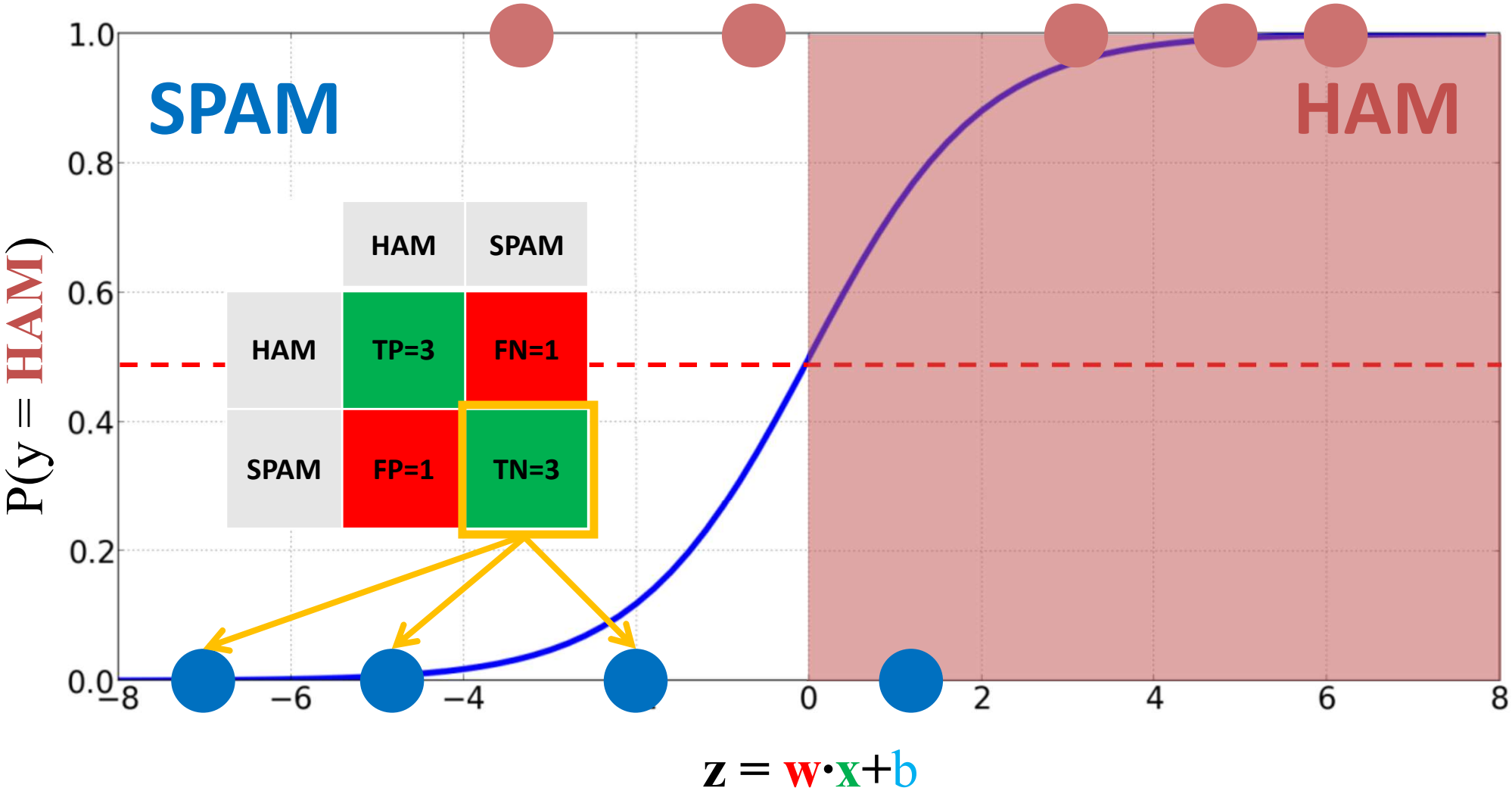
Logistic Regression: Threshold 0.5



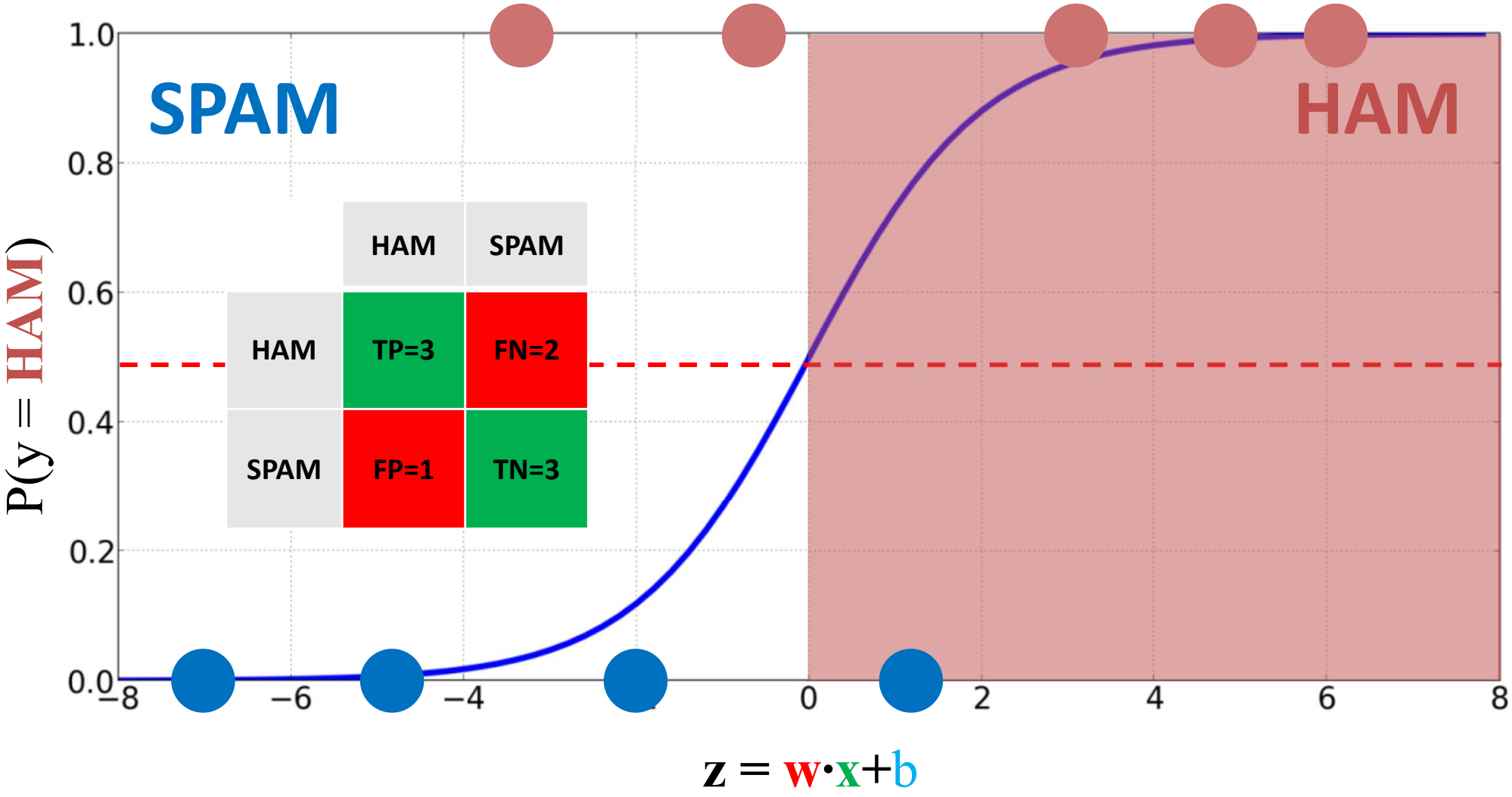
Logistic Regression: Threshold 0.5



Logistic Regression: Threshold 0.5



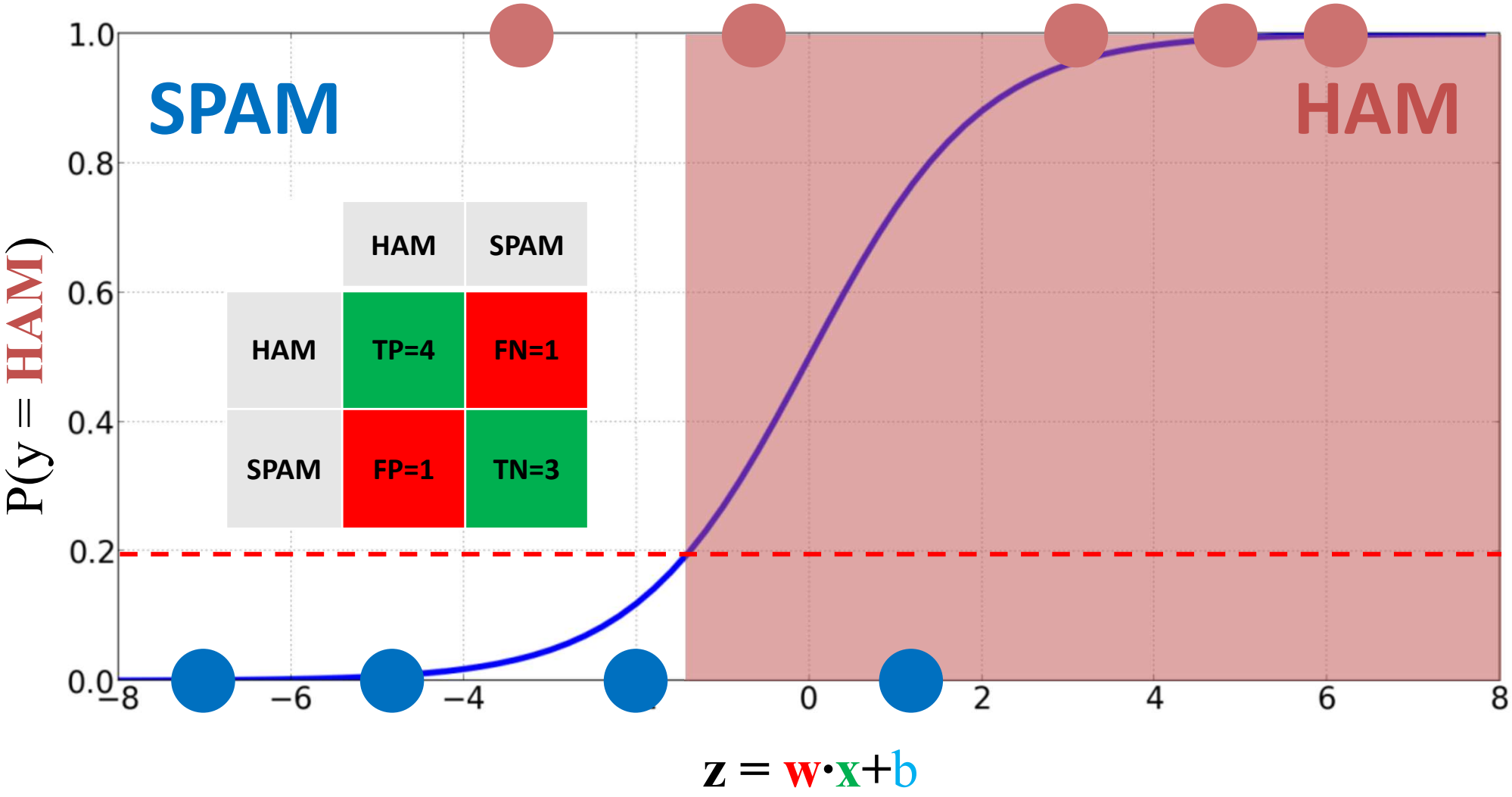
Logistic Regression: Threshold 0.5



Confusion Matrix: Threshold 0.5

		Predicted class: HAM	Predicted class: SPAM	
Actual class	HAM	True Positive (TP) 3	False Negative (FN) 2	Sensitivity (Recall) $\frac{TP}{TP+FN} = \frac{3}{5}$
	SPAM	False Positive (FP) 1	True Negative (TN) 3	Specificity $\frac{TN}{TN+FP} = \frac{3}{4}$
		Precision $\frac{TP}{TP+FP} = \frac{3}{4}$	Negative Predictive Value $\frac{TN}{TN+FN} = \frac{3}{5}$	Accuracy $\frac{TP+TN}{TP+TN+FP+FN} = \frac{6}{9}$

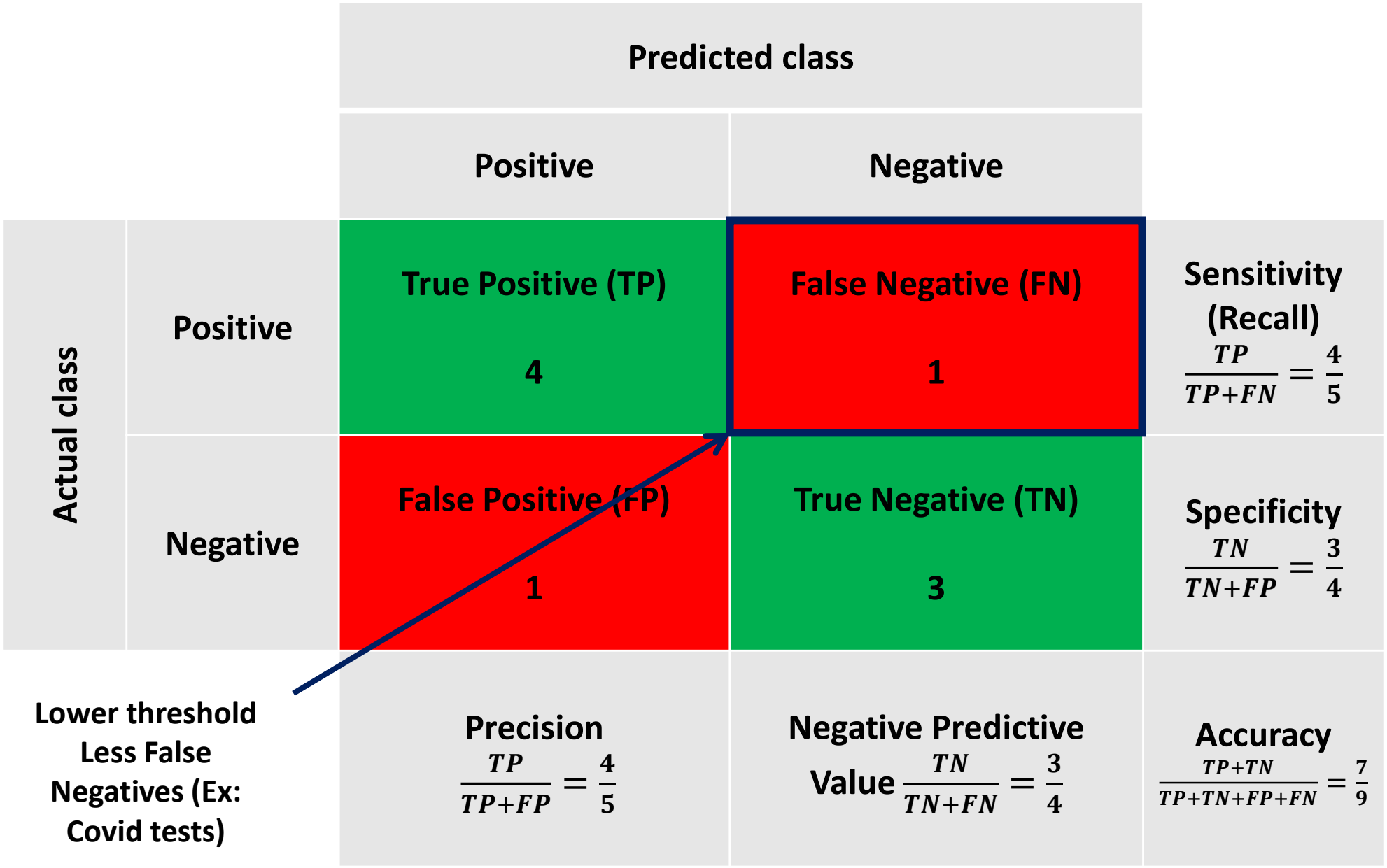
Logistic Regression: Threshold 0.2



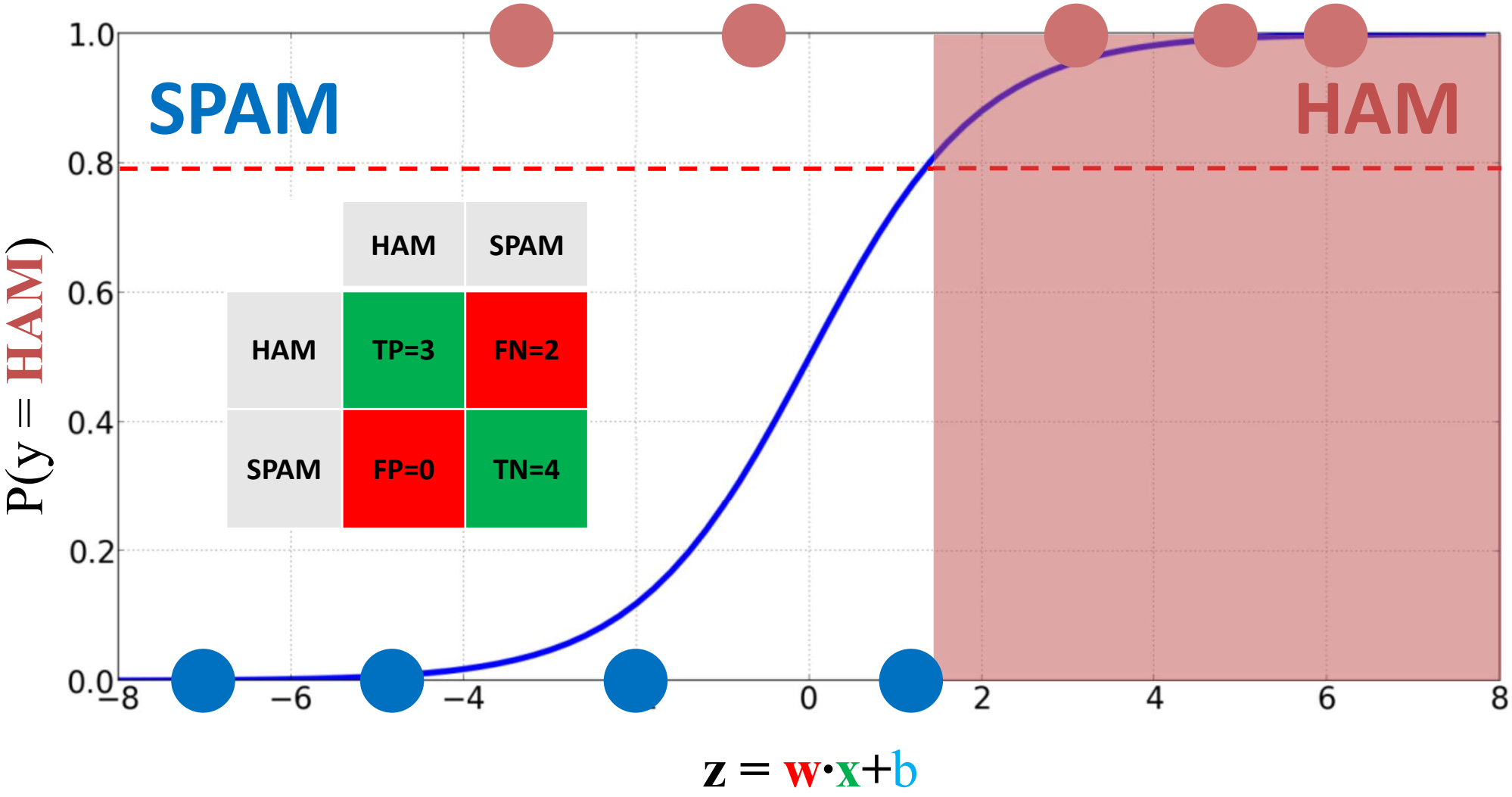
Confusion Matrix: Threshold 0.2

		Predicted class: HAM	Predicted class: SPAM	
Actual class	HAM	True Positive (TP) 4	False Negative (FN) 1	Sensitivity (Recall) $\frac{TP}{TP+FN} = \frac{4}{5}$
	SPAM	False Positive (FP) 1	True Negative (TN) 3	Specificity $\frac{TN}{TN+FP} = \frac{3}{4}$
		Precision $\frac{TP}{TP+FP} = \frac{4}{5}$	Negative Predictive Value $\frac{TN}{TN+FN} = \frac{3}{4}$	Accuracy $\frac{TP+TN}{TP+TN+FP+FN} = \frac{7}{9}$

Confusion Matrix: Threshold 0.2



Logistic Regression: Threshold 0.8



Confusion Matrix: Threshold 0.8

		Predicted class: HAM	Predicted class: SPAM	
Actual class	HAM	True Positive (TP) 3	False Negative (FN) 2	Sensitivity (Recall) $\frac{TP}{TP+FN} = \frac{3}{5}$
	SPAM	False Positive (FP) 0	True Negative (TN) 4	Specificity $\frac{TN}{TN+FP} = \frac{4}{4}$
		Precision $\frac{TP}{TP+FP} = \frac{3}{3}$	Negative Predictive Value $\frac{TN}{TN+FN} = \frac{4}{6}$	Accuracy $\frac{TP+TN}{TP+TN+FP+FN} = \frac{7}{9}$

Confusion Matrix: Threshold 0.8

		Predicted class: Positive	Predicted class: Negative	
Actual class	Positive	True Positive (TP) 3	False Negative (FN) 2	Sensitivity (Recall) $\frac{TP}{TP+FN} = \frac{3}{5}$
	Negative	False Positive (FP) 0	True Negative (TN) 4	Specificity $\frac{TN}{TN+FP} = \frac{4}{4}$
Higher threshold Less False Positives (Ex: crime convictions)		Precision $\frac{TP}{TP+FP} = \frac{3}{3}$	Negative Predictive Value $\frac{TN}{TN+FN} = \frac{4}{6}$	Accuracy $\frac{TP+TN}{TP+TN+FP+FN} = \frac{7}{9}$

Classifier Evaluation: Confusion Matrix

		Predicted class: Positive	Predicted class: Negative	
Actual class	Positive	True Positive (TP)	False Negative (FN) Type II Error	Sensitivity (Recall) $\frac{TP}{TP+FN}$
	Negative	False Positive (FP) Type I Error	True Negative (TN)	Specificity $\frac{TN}{TN+FP}$
		Precision $\frac{TP}{TP+FP}$	Negative Predictive Value $\frac{TN}{TN+FN}$	Accuracy $\frac{TP+TN}{TP+TN+FP+FN}$

Classifier Performance Metrics

- **Precision and recall provide two ways to summarize the errors made for the positive class (FP, FN).**

- **F-measure provides a single score that summarizes the precision and recall.**

- **Accuracy summarizes the correct predictions for both positive and negative classes.**

Receiver Operating Characteristic

Threshold: 0.5

	HAM	SPAM
HAM	TP=3	FN=2
SPAM	FP=1	TN=3

$$TPR = \frac{TP}{TP + FN} = \frac{3}{5}$$

$$FPR = \frac{FP}{FP + TN} = \frac{1}{4}$$

Threshold: 0.2

	HAM	SPAM
HAM	TP=4	FN=1
SPAM	FP=1	TN=3

$$TPR = \frac{TP}{TP + FN} = \frac{4}{5}$$

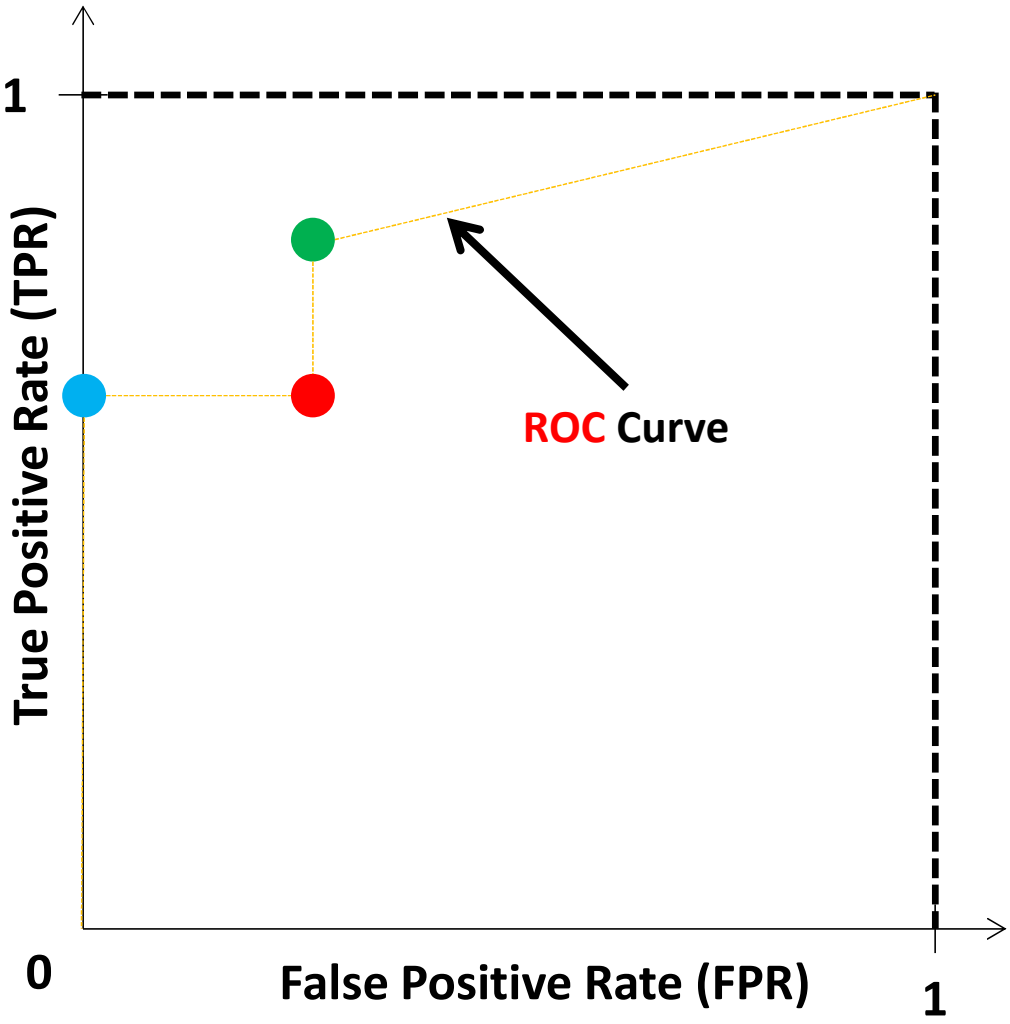
$$FPR = \frac{FP}{FP + TN} = \frac{1}{4}$$

Threshold: 0.8

	HAM	SPAM
HAM	TP=3	FN=2
SPAM	FP=0	TN=4

$$TPR = \frac{TP}{TP + FN} = \frac{3}{5}$$

$$FPR = \frac{FP}{FP + TN} = \frac{0}{4}$$



TPR: Sensitivity | FPR: 1 - Specificity

ROC Area Under the Curve

Threshold: 0.5

	HAM	SPAM
HAM	TP=3	FN=2
SPAM	FP=1	TN=3

$$TPR = \frac{TP}{TP + FN} = \frac{3}{5}$$

$$FPR = \frac{FP}{FP + TN} = \frac{1}{4}$$

Threshold: 0.2

	HAM	SPAM
HAM	TP=4	FN=1
SPAM	FP=1	TN=3

$$TPR = \frac{TP}{TP + FN} = \frac{4}{5}$$

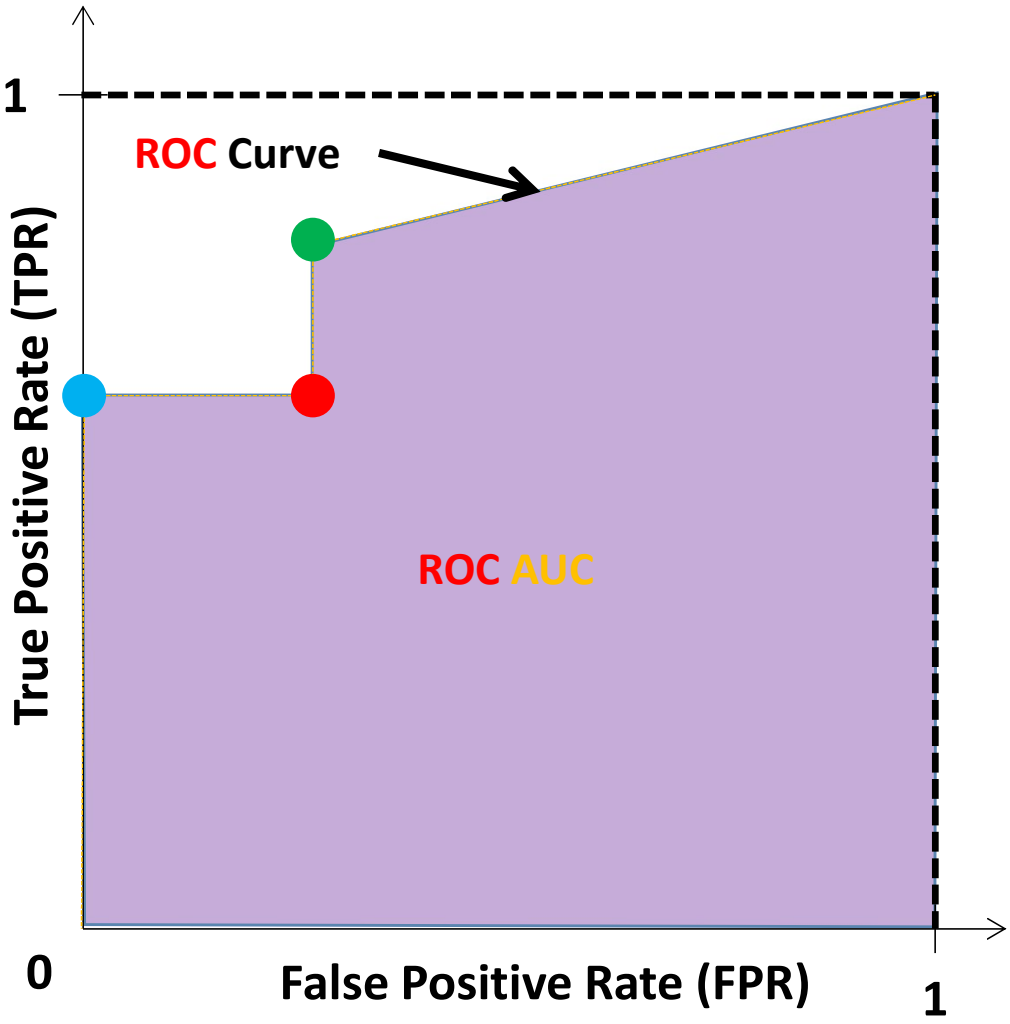
$$FPR = \frac{FP}{FP + TN} = \frac{1}{4}$$

Threshold: 0.8

	HAM	SPAM
HAM	TP=3	FN=2
SPAM	FP=0	TN=4

$$TPR = \frac{TP}{TP + FN} = \frac{3}{5}$$

$$FPR = \frac{FP}{FP + TN} = \frac{0}{4}$$



TPR: Sensitivity | FPR: 1 - Specificity

Receiver Operating Characteristic

Threshold: 0.5

	HAM	SPAM
HAM	TP=3	FN=2
SPAM	FP=1	TN=3

$$TPR = \frac{TP}{TP + FN} = \frac{3}{5}$$

$$FPR = \frac{FP}{FP + TN} = \frac{1}{4}$$

Threshold: 0.2

	HAM	SPAM
HAM	TP=4	FN=1
SPAM	FP=1	TN=3

$$TPR = \frac{TP}{TP + FN} = \frac{4}{5}$$

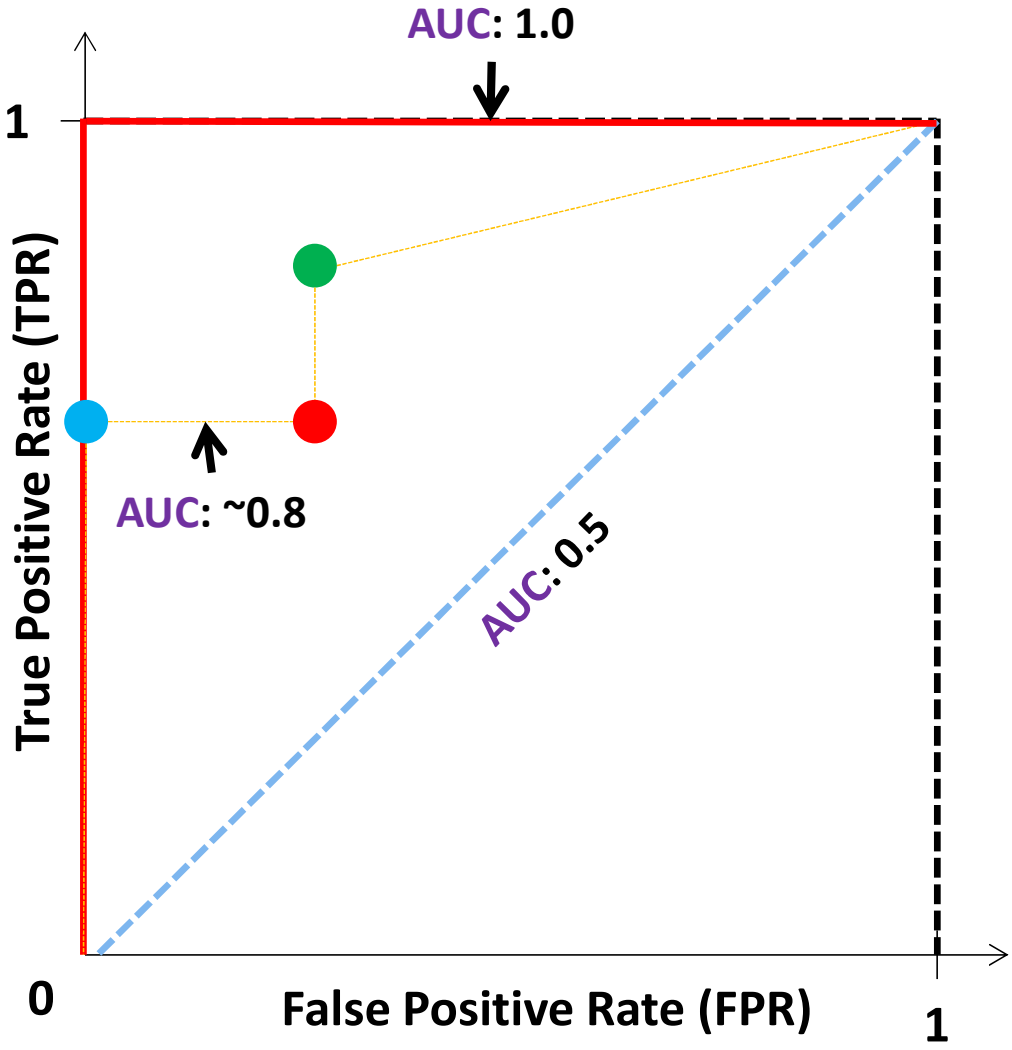
$$FPR = \frac{FP}{FP + TN} = \frac{1}{4}$$

Threshold: 0.8

	HAM	SPAM
HAM	TP=3	FN=2
SPAM	FP=0	TN=4

$$TPR = \frac{TP}{TP + FN} = \frac{3}{5}$$

$$FPR = \frac{FP}{FP + TN} = \frac{0}{4}$$



TPR: Sensitivity | FPR: 1 - Specificity

Receiver Operating Characteristic

Threshold: 0.5

	HAM	SPAM
HAM	TP=3	FN=2
SPAM	FP=1	TN=3

$$TPR = \frac{TP}{TP + FN} = \frac{3}{5}$$

$$FPR = \frac{FP}{FP + TN} = \frac{1}{4}$$

Threshold: 0.2

	HAM	SPAM
HAM	TP=4	FN=1
SPAM	FP=1	TN=3

$$TPR = \frac{TP}{TP + FN} = \frac{4}{5}$$

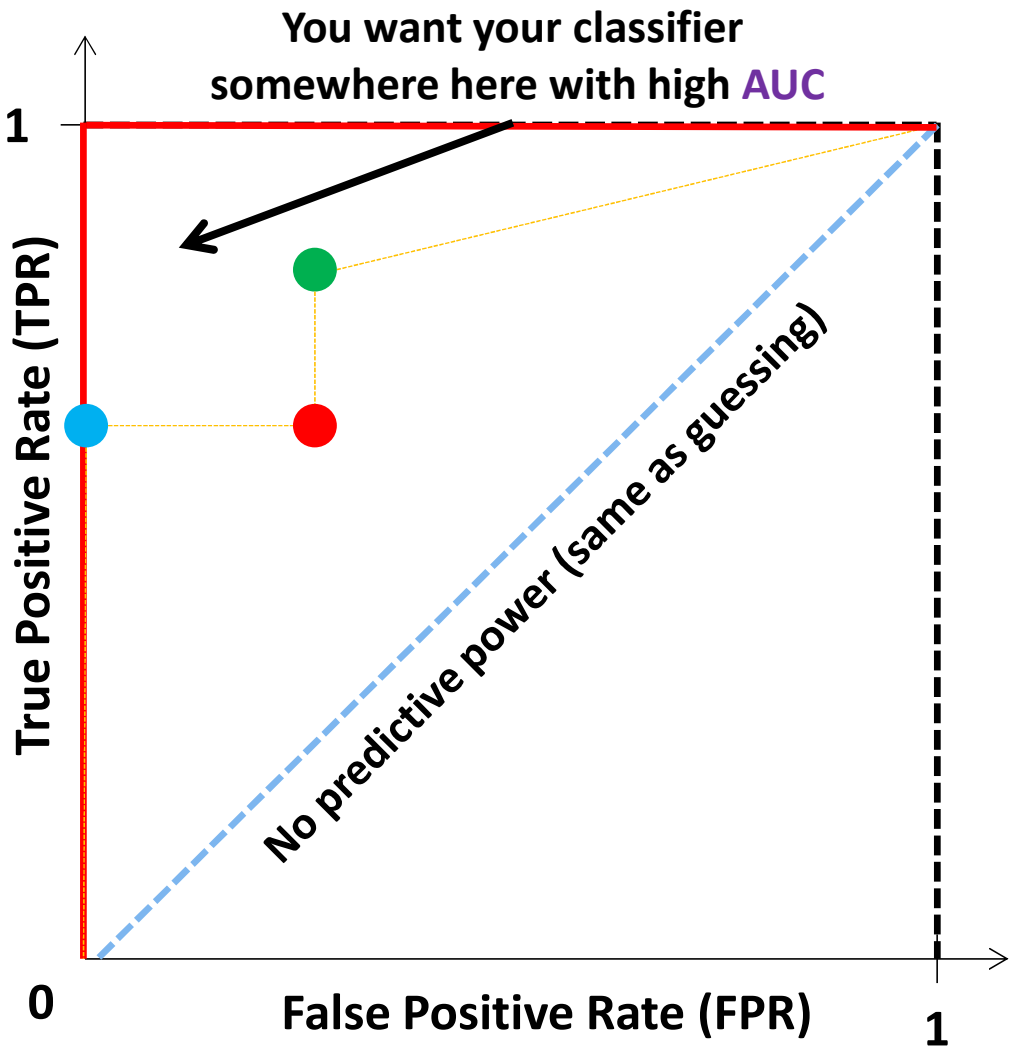
$$FPR = \frac{FP}{FP + TN} = \frac{1}{4}$$

Threshold: 0.8

	HAM	SPAM
HAM	TP=3	FN=2
SPAM	FP=0	TN=4

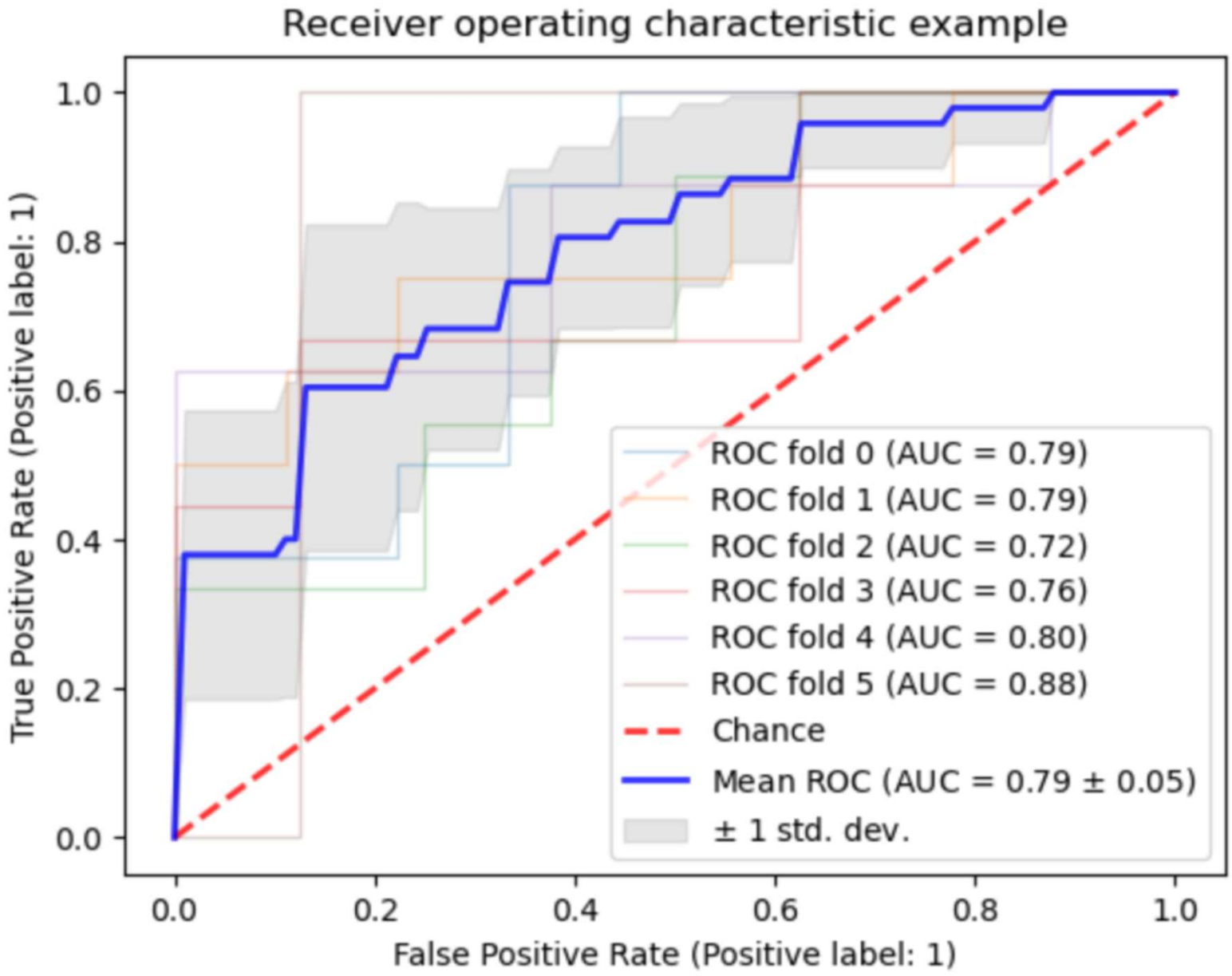
$$TPR = \frac{TP}{TP + FN} = \frac{3}{5}$$

$$FPR = \frac{FP}{FP + TN} = \frac{0}{4}$$

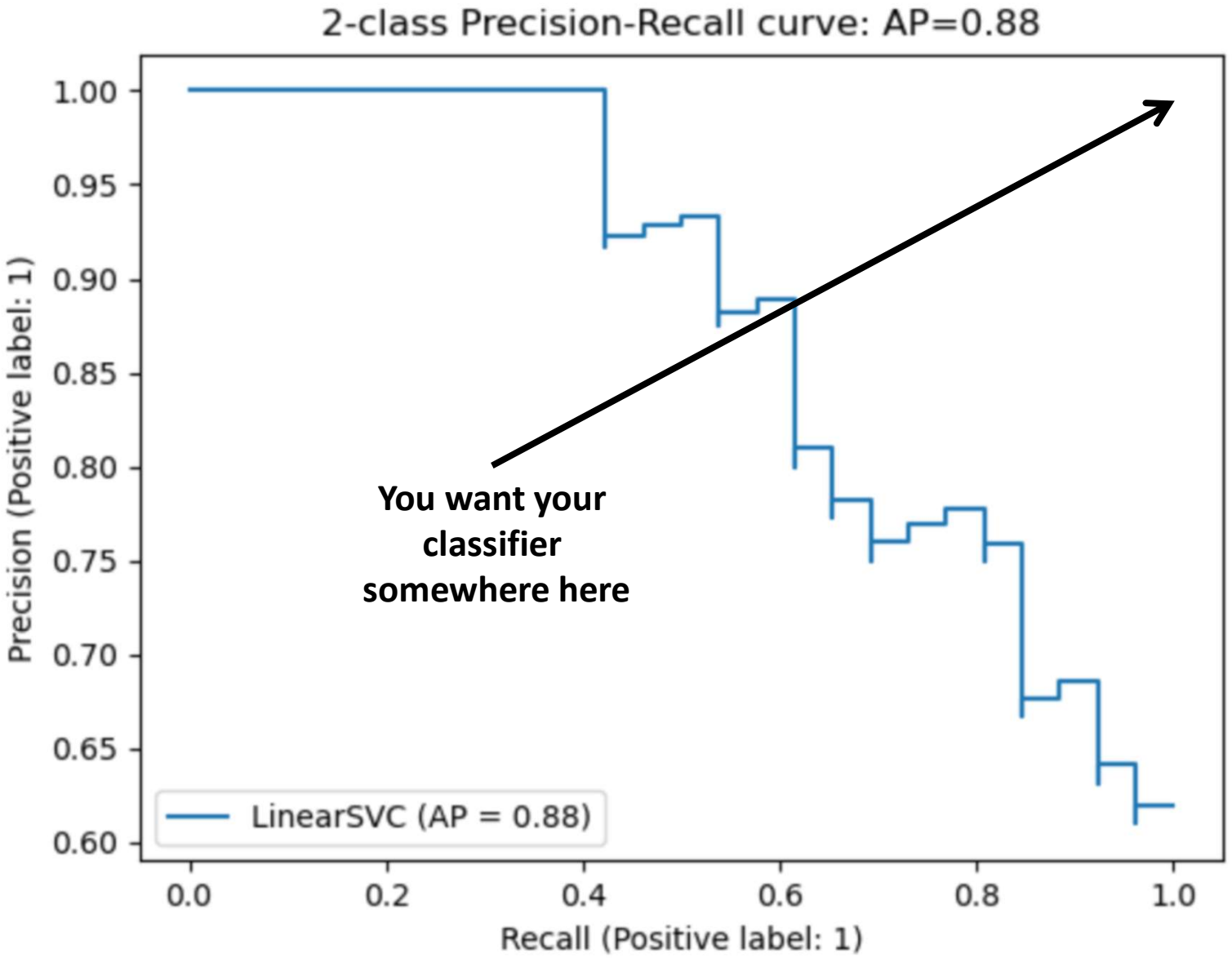


TPR: Sensitivity | FPR: 1 - Specificity

Receiver Operating Characteristic



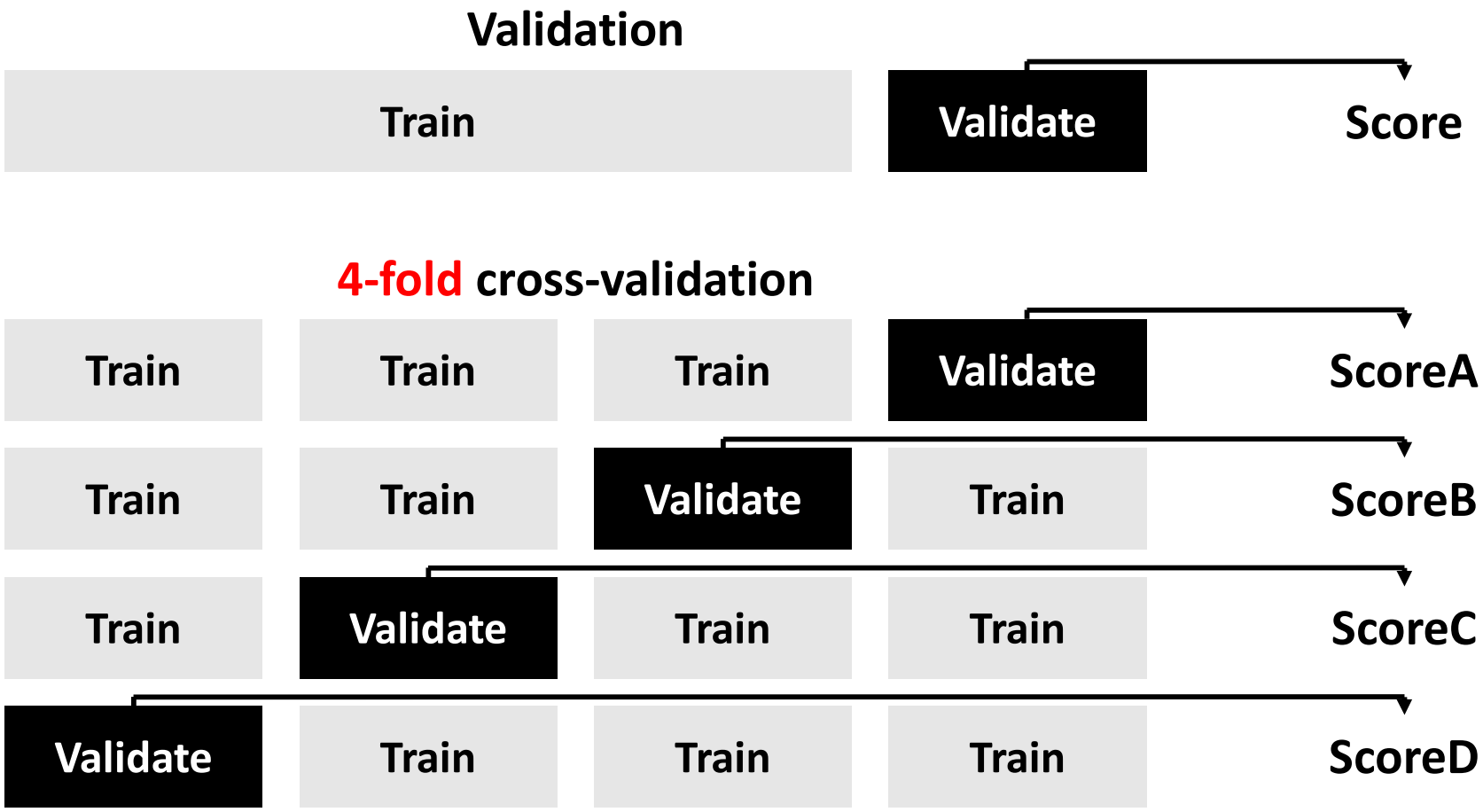
Precision - Recall Curve



ROC vs. Precision-Recall Curves

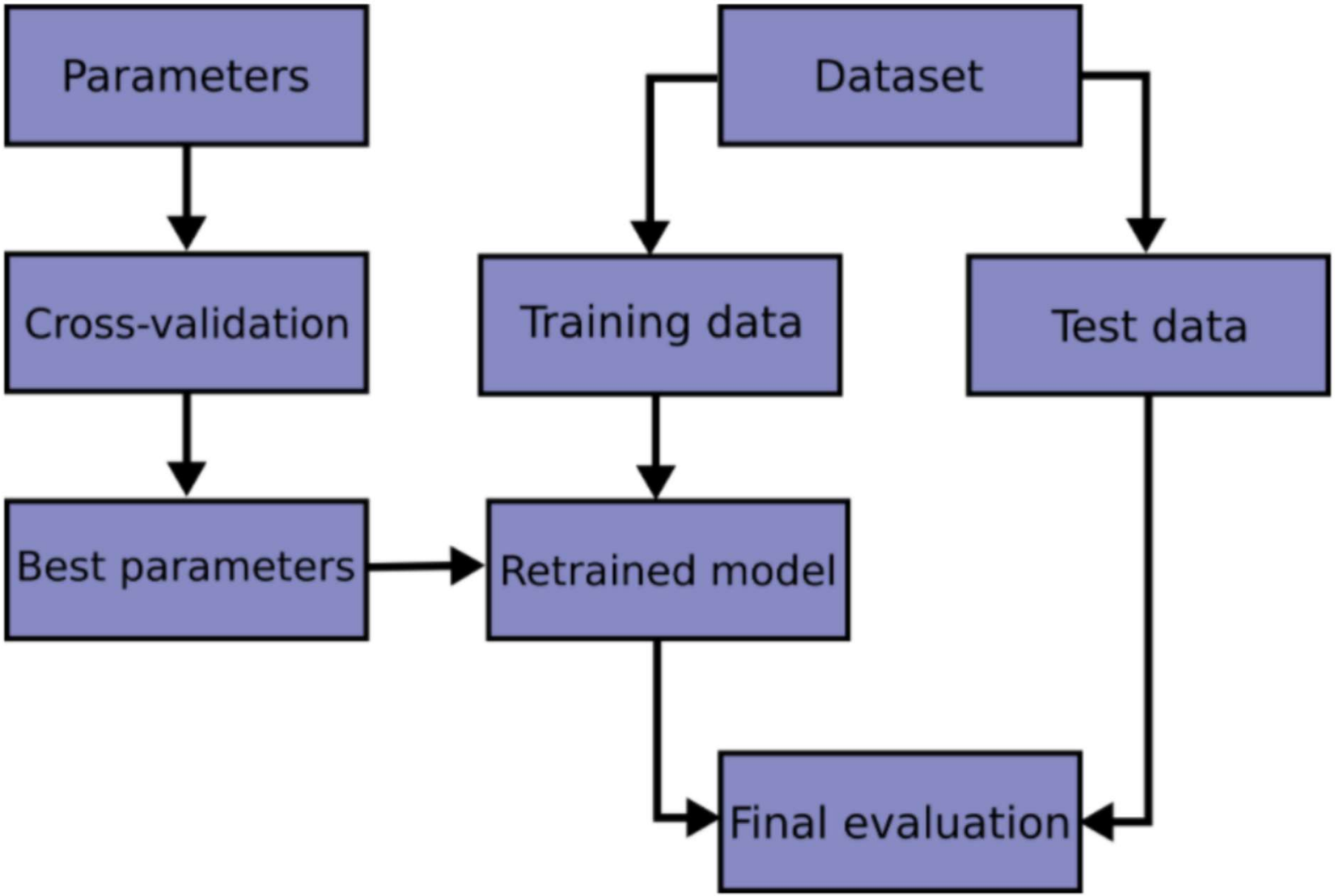
- Both summarize model performance using different probability thresholds
- ROC curves should be used when there are roughly equal numbers of observations for each class
- Precision-Recall curves should be used when there is a moderate to large class imbalance (when we are interested in the positive class and there's only a few positive samples)

K-Fold Cross-Validation

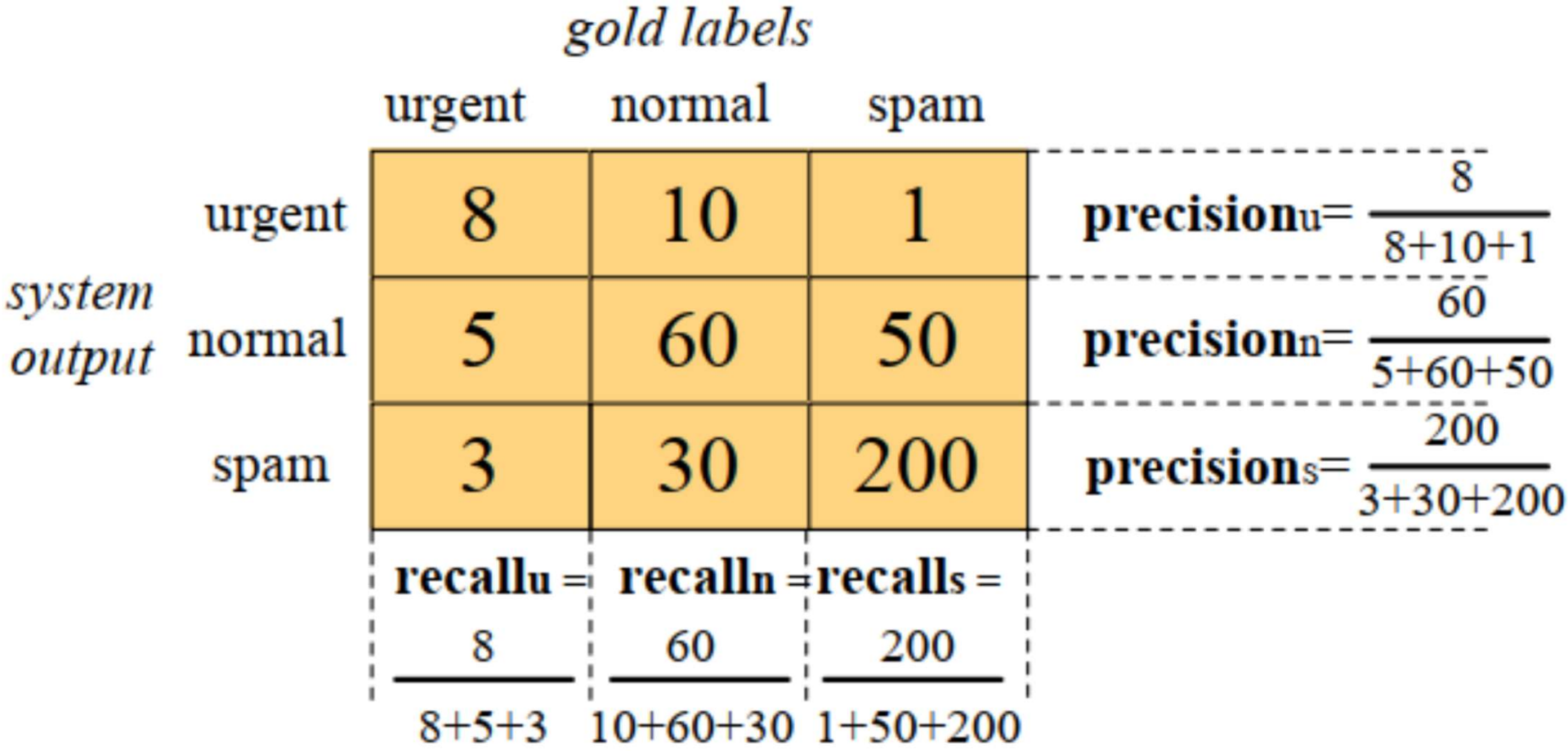


$$\text{Score} = \frac{\text{ScoreA} + \text{ScoreB} + \text{ScoreC} + \text{ScoreD}}{4}$$

Parameter Tuning



3-class Confusion Matrix



Macroaveraging and Microaveraging

Macroaveraging:

- **compute the performance for each class, and then average over classes**

Microaveraging:

- **collect decisions for all classes into one confusion matrix**
- **compute precision and recall from that table.**

Macroaveraging and Microaveraging

Class 1: Urgent

	true urgent	true not
system urgent	8	11
system not	8	340

$$\text{precision} = \frac{8}{8+11} = .42$$

Class 2: Normal

	true normal	true not
system normal	60	55
system not	40	212

$$\text{precision} = \frac{60}{60+55} = .52$$

Class 3: Spam

	true spam	true not
system spam	200	33
system not	51	83

$$\text{precision} = \frac{200}{200+33} = .86$$

Pooled

	true yes	true no
system yes	268	99
system no	99	635

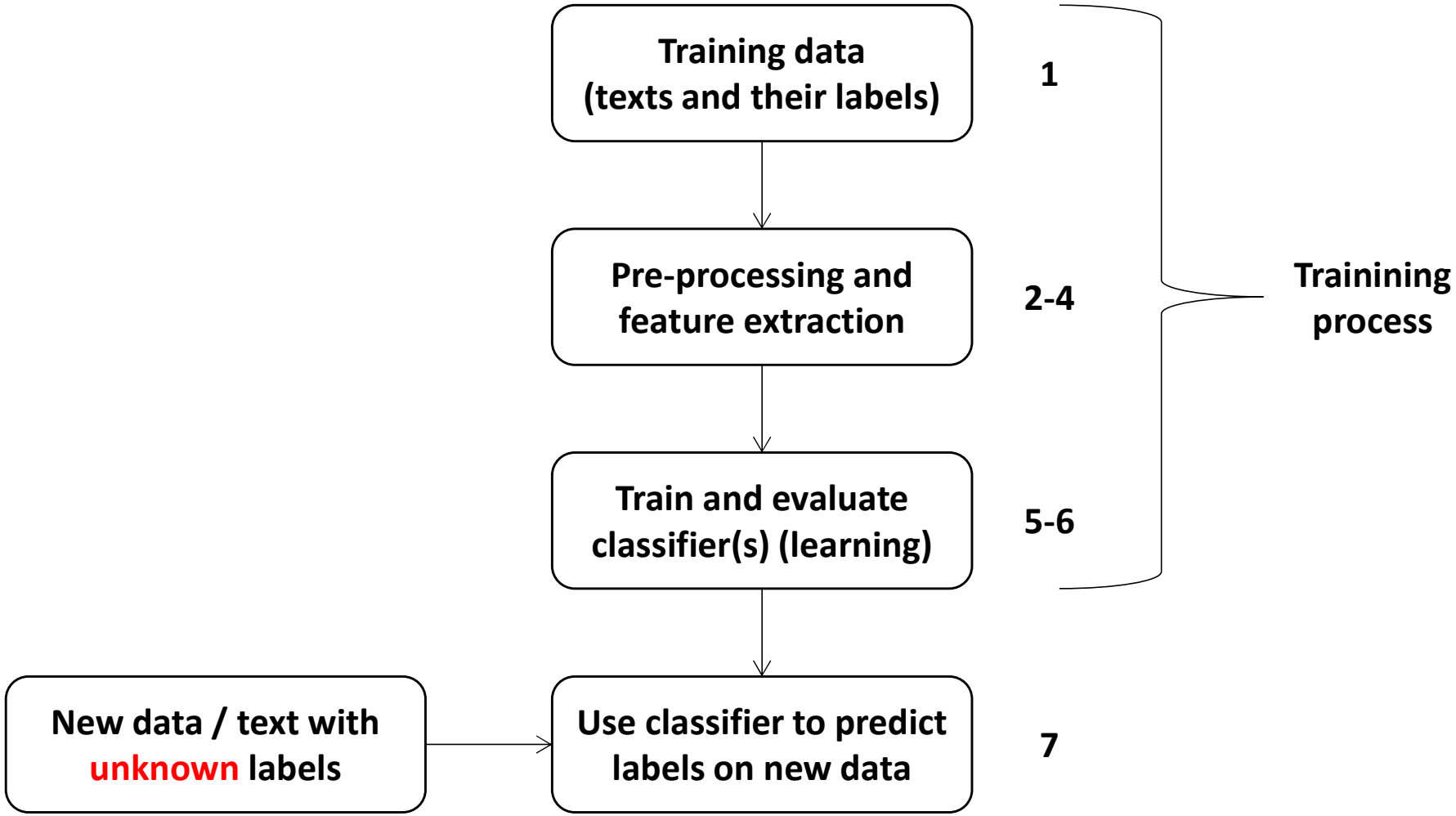
$$\text{microaverage precision} = \frac{268}{268+99} = \mathbf{.73}$$

$$\text{macroaverage precision} = \frac{.42+.52+.86}{3} = \mathbf{.60}$$

Text Classification System Pipeline

1. Obtain / collect / create labeled data set suitable for the task
2. Split the data set into:
   - two (training and test sets) parts OR
   - three (training, validation, and test sets) parts
3. Choose evaluation metric
4. Transform raw text into feature vectors:
   - bag of words
   - other types
5. Using feature vectors and labels from the training set, train the classifier / create a model
6. Using evaluation metric from (3) benchmark the classifier / model performance using the test set
7. Deploy the classifier / model to serve a real world application and monitor its performance

Text Classification System Pipeline



Poor Classifier Performance: Reasons

1. With all possible features extracted, we ended up with a sparse feature vector (some features are too rare and end up being noise) → makes training hard
2. Few (~20%) relevant samples compared to non-relevant (~80%) samples in the data set → skews learning towards non-relevant data
3. Need better learning algorithm
4. Need better pre-processing / feature extraction
5. Classifier parameters / hyperparameters need tuning