Binary Classification

Shusen Wang

Vector and Matrix Derivatives

Derivative of Vector w.r.t. Scalar

- The derivative of a vector $\mathbf{y} \in \mathbb{R}^n$ w.r.t. a scalar $x \in \mathbb{R}$:

$$\frac{\partial\,\mathbf{y}}{\partial\,x} = \begin{bmatrix} \frac{\partial y_1}{\partial x} \\ \frac{\partial y_2}{\partial x} \\ \vdots \\ \frac{\partial y_n}{\partial x} \end{bmatrix}$$

- Example:

$$\mathbf{y} = \begin{bmatrix} 3x^2 \\ x+1 \\ \log x \\ e^x \end{bmatrix}, \qquad \frac{\partial\,\mathbf{y}}{\partial\,x} = \begin{bmatrix} 6x \\ 1 \\ 1/x \\ e^x \end{bmatrix}$$

Derivative of Scalar w.r.t. Vector

- The derivative of a scalar $y \in \mathbb{R}$ w.r.t. a vector $\mathbf{x} \in \mathbb{R}^m$:

$$\frac{\partial\, y}{\partial\, \mathbf{x}} = \begin{bmatrix} \frac{\partial y}{\partial x_1} \\ \frac{\partial y}{\partial x_2} \\ \vdots \\ \frac{\partial y}{\partial x_m} \end{bmatrix}$$

- Example 1:

$$y = \|\mathbf{x}\|_2^2 = \sum_{i=1}^{m} x_i^2, \qquad \frac{\partial\, y}{\partial\, \mathbf{x}} = 2\mathbf{x}.$$

Derivative of Scalar w.r.t. Vector

- The derivative of a scalar $y \in \mathbb{R}$ w.r.t. a vector $\mathbf{x} \in \mathbb{R}^m$:

$$\frac{\partial\, y}{\partial\, \mathbf{x}} = \begin{bmatrix} \frac{\partial y}{\partial x_1} \\ \frac{\partial y}{\partial x_2} \\ \vdots \\ \frac{\partial y}{\partial x_m} \end{bmatrix}$$

- Example 2:

$$y = \mathbf{x}^T \mathbf{z} = \sum_{i=1}^{m} x_i z_i, \qquad \frac{\partial\, y}{\partial\, \mathbf{x}} = \mathbf{z}.$$

Derivative of Scalar w.r.t. Vector

- The derivative of a scalar $y \in \mathbb{R}$ w.r.t. a vector $\mathbf{x} \in \mathbb{R}^m$:

$$\frac{\partial y}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial y}{\partial x_1} \\ \frac{\partial y}{\partial x_2} \\ \vdots \\ \frac{\partial y}{\partial x_m} \end{bmatrix}$$

- Example 3:

$$y = \sum_{i=1}^{m} \log(1 + e^{-x_i}), \qquad \frac{\partial y}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial \log(1+e^{-x_1})}{\partial x_1} \\ \vdots \\ \frac{\partial \log(1+e^{-x_m})}{\partial x_m} \end{bmatrix} = \begin{bmatrix} -\frac{1}{1+e^{x_1}} \\ \vdots \\ -\frac{1}{1+e^{x_m}} \end{bmatrix}$$

Derivative of Vector w.r.t. Vector

- The derivative of a vector $\mathbf{y} \in \mathbb{R}^n$ w.r.t. a vector $\mathbf{x} \in \mathbb{R}^m$:

$$\frac{\partial \mathbf{y}}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \frac{\partial y_2}{\partial x_1} & \cdots & \frac{\partial y_n}{\partial x_1} \\ \frac{\partial y_1}{\partial x_2} & \frac{\partial y_2}{\partial x_2} & \cdots & \frac{\partial y_n}{\partial x_2} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial y_1}{\partial x_m} & \frac{\partial y_2}{\partial x_m} & \cdots & \frac{\partial y_n}{\partial x_m} \end{bmatrix}$$

$m \times n$ matrix

The (i, j)-th entry is $\frac{\partial y_j}{\partial x_i}$

- Example 1:

$$\frac{\partial \mathbf{x}}{\partial \mathbf{x}} = \underbrace{\begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}}_{m \times m}$$

Derivative of Vector w.r.t. Vector

- The derivative of a vector $\mathbf{y} \in \mathbb{R}^n$ w.r.t. a vector $\mathbf{x} \in \mathbb{R}^m$:

$$\frac{\partial \mathbf{y}}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \frac{\partial y_2}{\partial x_1} & \cdots & \frac{\partial y_n}{\partial x_1} \\ \frac{\partial y_1}{\partial x_2} & \frac{\partial y_2}{\partial x_2} & \cdots & \frac{\partial y_n}{\partial x_2} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial y_1}{\partial x_m} & \frac{\partial y_2}{\partial x_m} & \cdots & \frac{\partial y_n}{\partial x_m} \end{bmatrix}$$

$m \times n$ matrix

- Example 2:

$$\mathbf{y} = \begin{bmatrix} a_1 x_1^2 \\ a_2 x_2^2 \\ \vdots \\ a_m x_m^2 \end{bmatrix} \in \mathbb{R}^m, \qquad \frac{\partial \mathbf{y}}{\partial \mathbf{x}} = \underbrace{\begin{bmatrix} 2a_1 x_1 & 0 & \cdots & 0 \\ 0 & 2a_2 x_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 2a_m x_m \end{bmatrix}}_{m \times m}$$

Derivative of Vector w.r.t. Vector

- The derivative of a vector $\mathbf{y} \in \mathbb{R}^n$ w.r.t. a vector $\mathbf{x} \in \mathbb{R}^m$:

$$\frac{\partial \mathbf{y}}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \frac{\partial y_2}{\partial x_1} & \cdots & \frac{\partial y_n}{\partial x_1} \\ \frac{\partial y_1}{\partial x_2} & \frac{\partial y_2}{\partial x_2} & \cdots & \frac{\partial y_n}{\partial x_2} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial y_1}{\partial x_m} & \frac{\partial y_2}{\partial x_m} & \cdots & \frac{\partial y_n}{\partial x_m} \end{bmatrix}$$

$m \times n$ matrix

- Example 3:

$$\mathbf{A} \in \mathbb{R}^{n\times m}, \qquad \mathbf{y} = \mathbf{A}\mathbf{x} \in \mathbb{R}^n, \qquad \frac{\partial \mathbf{y}}{\partial \mathbf{x}} = \mathbf{A}^T \in \mathbb{R}^{m\times n}$$

Chain Rule

- Let $\mathbf{z} \in \mathbb{R}^{n_z}$ be a function of $\mathbf{y} \in \mathbb{R}^{n_y}$ and $\mathbf{y}$ be a function of $\mathbf{x} \in \mathbb{R}^{n_x}$.

$$\underbrace{\frac{d\mathbf{z}}{d\mathbf{x}}}_{n_x \times n_z} = \underbrace{\frac{d\mathbf{y}}{d\mathbf{x}}}_{n_x \times n_y} \underbrace{\frac{d\mathbf{z}}{d\mathbf{y}}}_{n_y \times n_z}$$

Derivative of Scalar w.r.t. Matrix

- The derivative of a scalar $y \in \mathbb{R}$ w.r.t. a matrix $\mathbf{Z} \in \mathbb{R}^{p \times q}$:
    1. Vectorization: $\mathbf{x} = \mathrm{vec}(\mathbf{Z}) \in \mathbb{R}^{pq \times 1}$.
    2. Compute $\frac{\partial\, y}{\partial\, \mathbf{x}} \in \mathbb{R}^{pq \times 1}$.
    3. Reshape the resulting $pq \times 1$ vector to $p \times q$ matrix.

Derivative of Vector w.r.t. Matrix

- The derivative of a vector $\mathbf{y} \in \mathbb{R}^n$ w.r.t. a matrix $\mathbf{Z} \in \mathbb{R}^{p \times q}$:
  1. Vectorization: $\mathbf{x} = \mathrm{vec}(\mathbf{Z}) \in \mathbb{R}^{pq \times 1}$.
  2. Compute $\frac{\partial \mathbf{y}}{\partial \mathbf{x}} \in \mathbb{R}^{pq \times n}$.
  3. Reshape the resulting $pq \times n$ matrix to $p \times q \times n$ tensor.

Binary Classification

Tasks

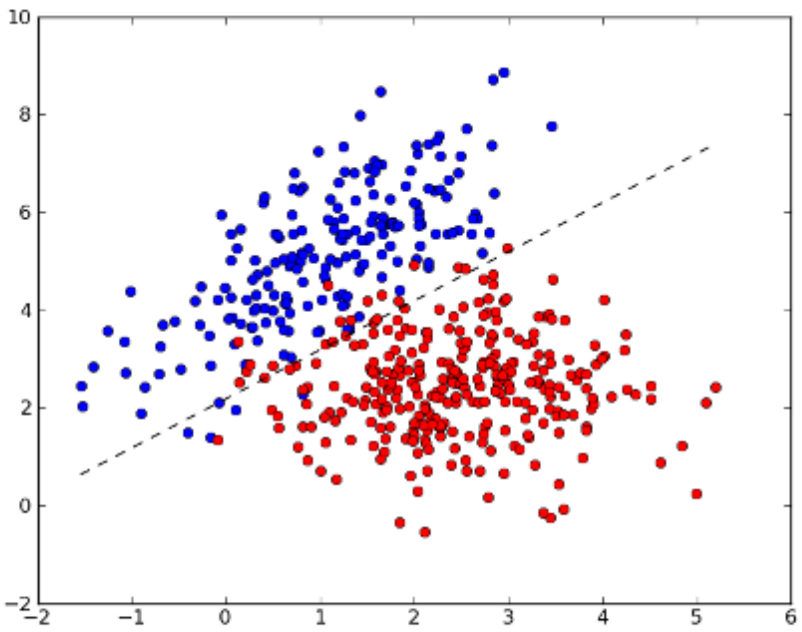
Methods

Algorithms

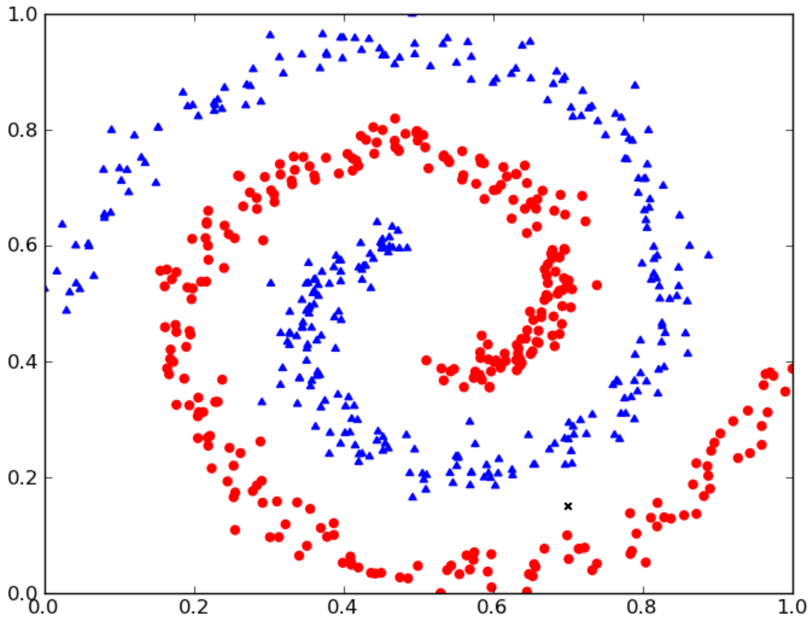
Binary Classification

Input: feature vectors $\mathbf{x}_1, \cdots, \mathbf{x}_n \in \mathbb{R}^d$ and labels $y_1, \cdots, y_n \in \{-1, +1\}$.

Output: a function $f: \mathbb{R}^d \mapsto \{-1, +1\}$.



Linear Classification



Nonlinear Classification

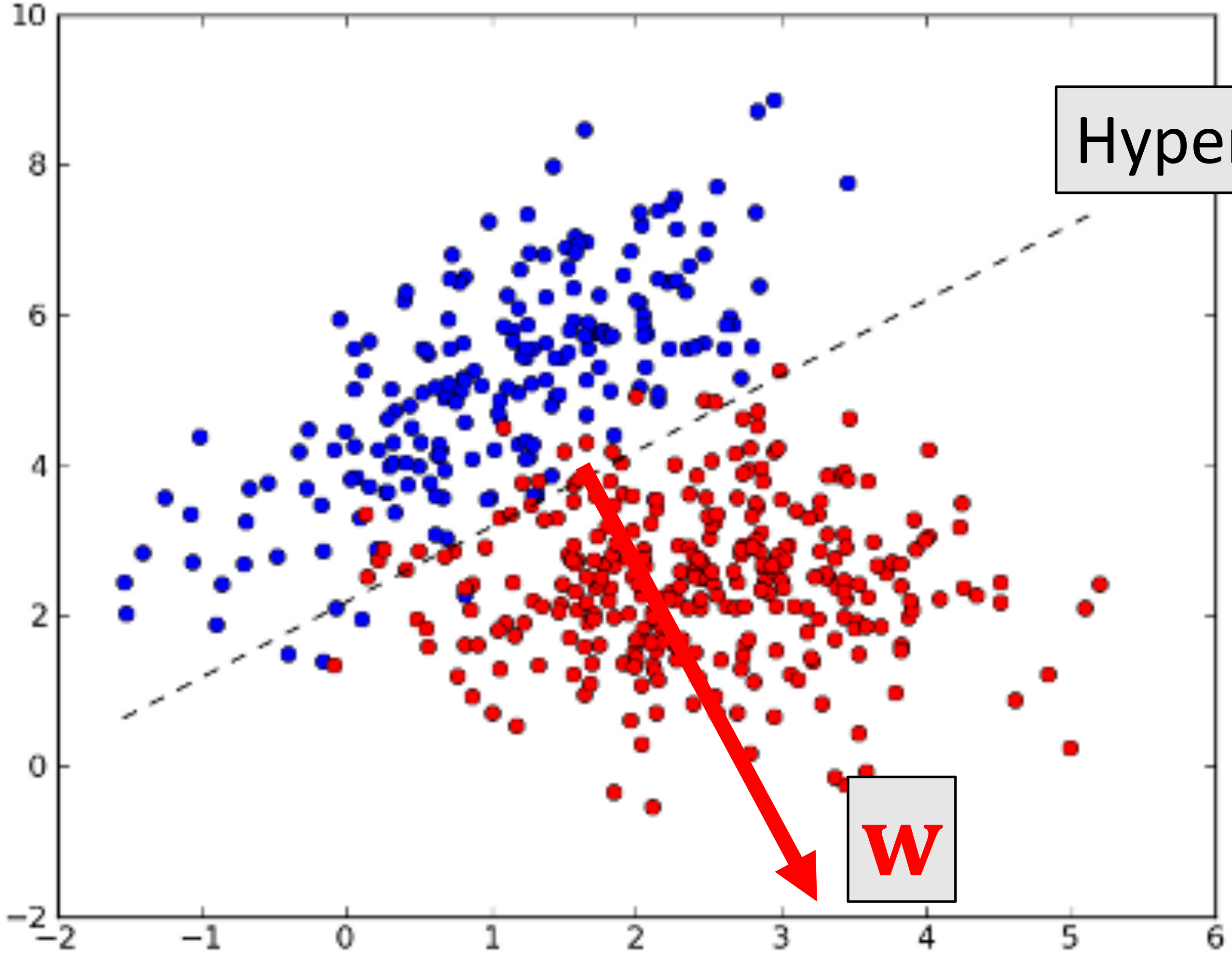
Logistic Regression (Linear Classification)

Tasks

Methods

Algorithms

Linear Classifier



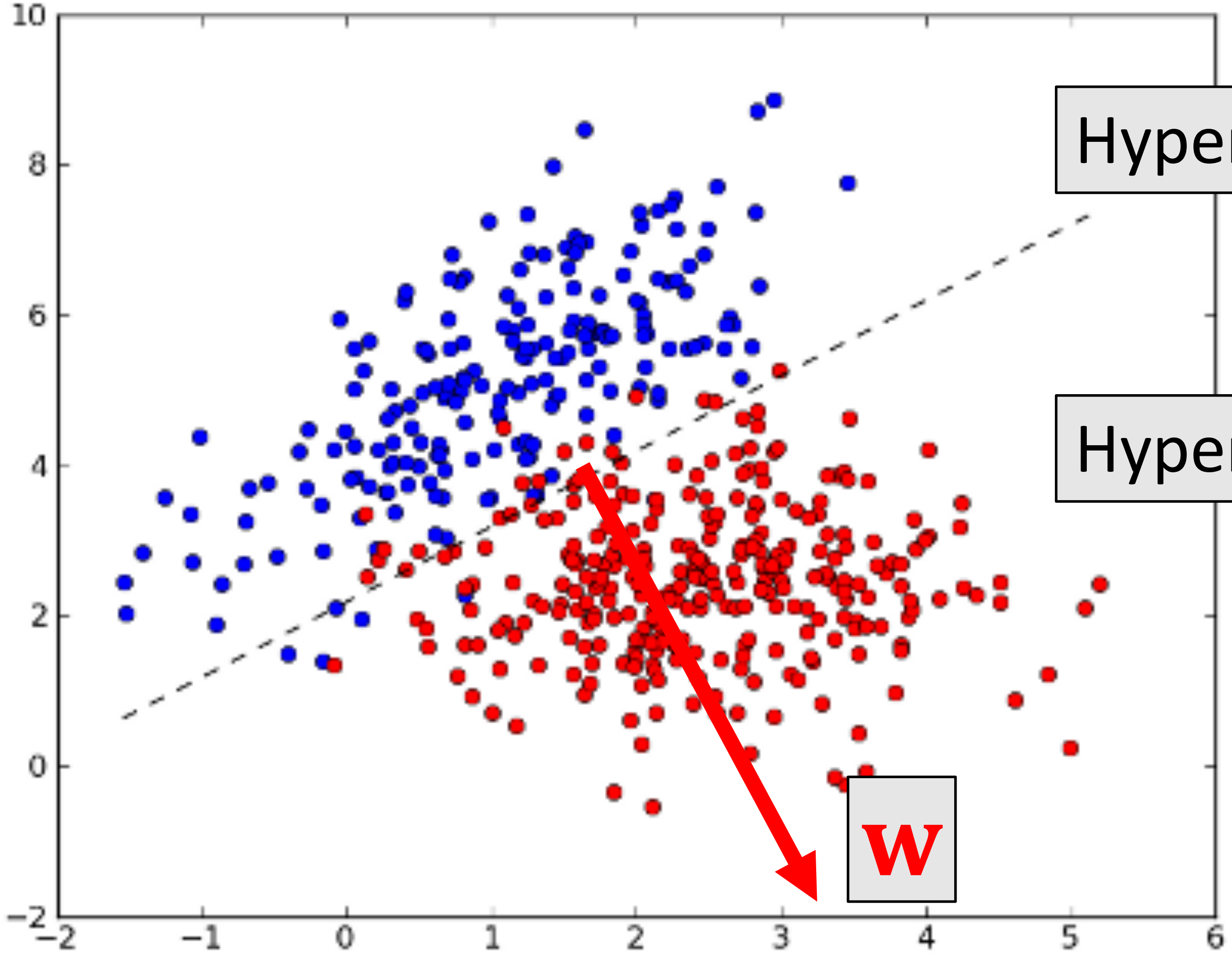
Hyperplane: $\mathbf{w}^T\mathbf{x} + b = 0$

Define $\bar{\mathbf{x}}_j = [\mathbf{x}_j; 1] \in \mathbb{R}^{d+1}$

Define $\bar{\mathbf{w}} = [\mathbf{w}, b] \in \mathbb{R}^{d+1}$

➔ $\mathbf{x}_j^T\mathbf{w} + b = \bar{\mathbf{x}}_j^T\bar{\mathbf{w}}$

Linear Classifier

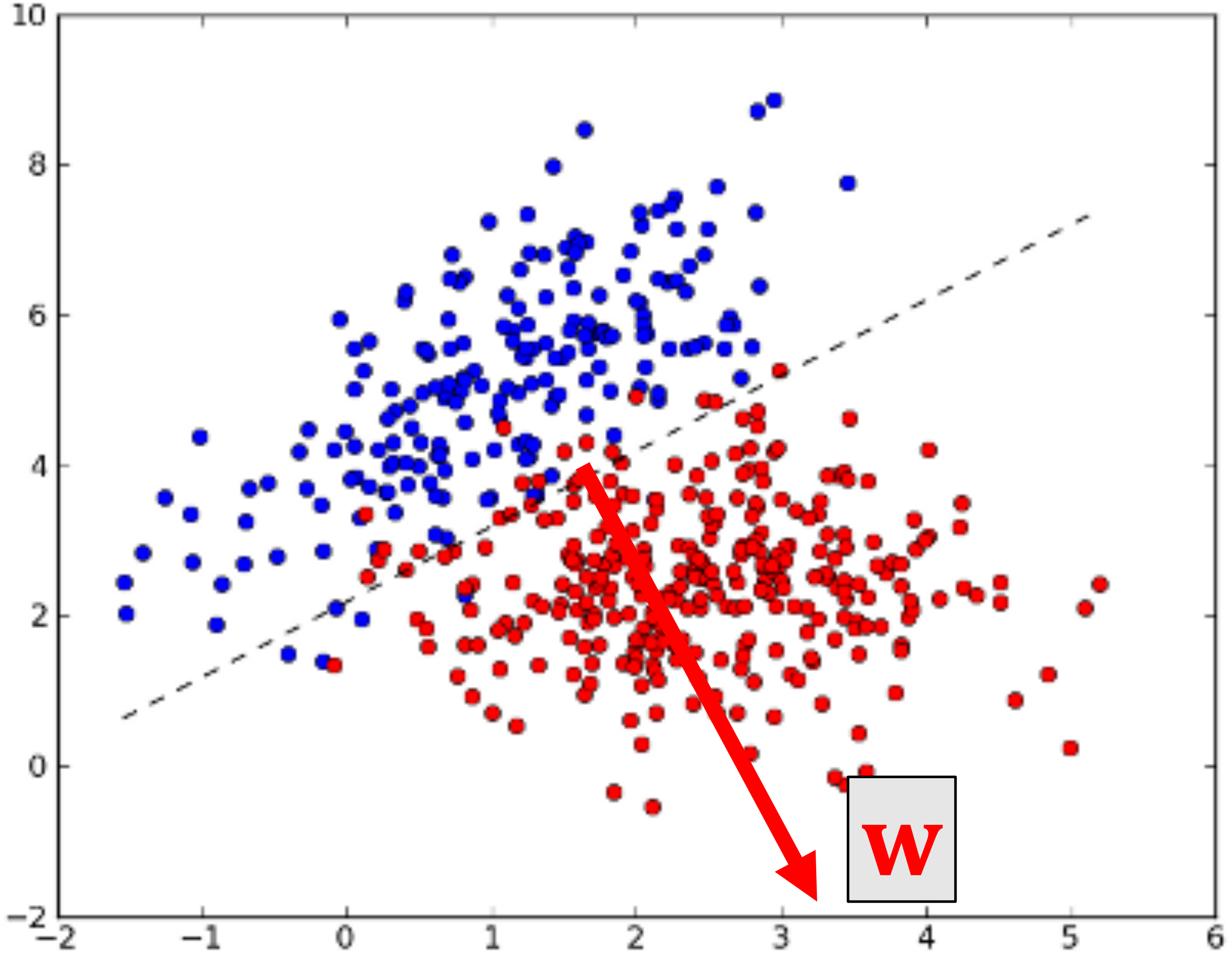


Hyperplane: $\mathbf{w}^T \mathbf{x} + b = 0$

Hyperplane: $\bar{\mathbf{w}}^T \bar{\mathbf{x}} = 0$

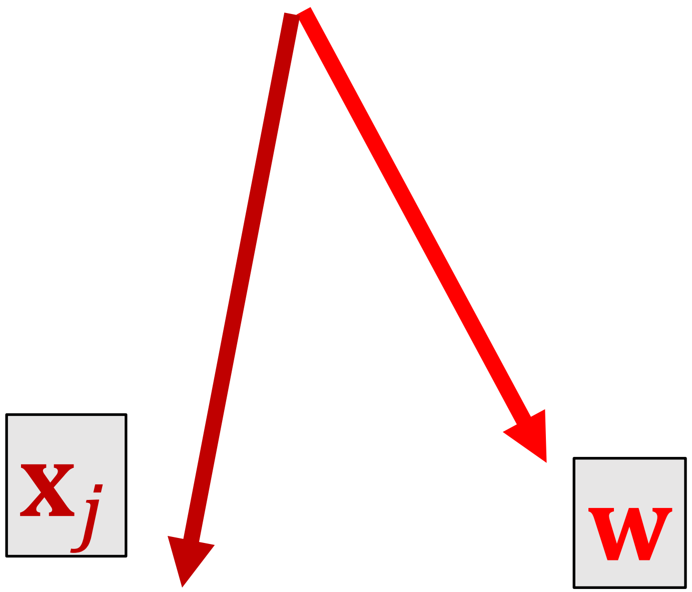
(The intercept is removed.)

Linear Classifier

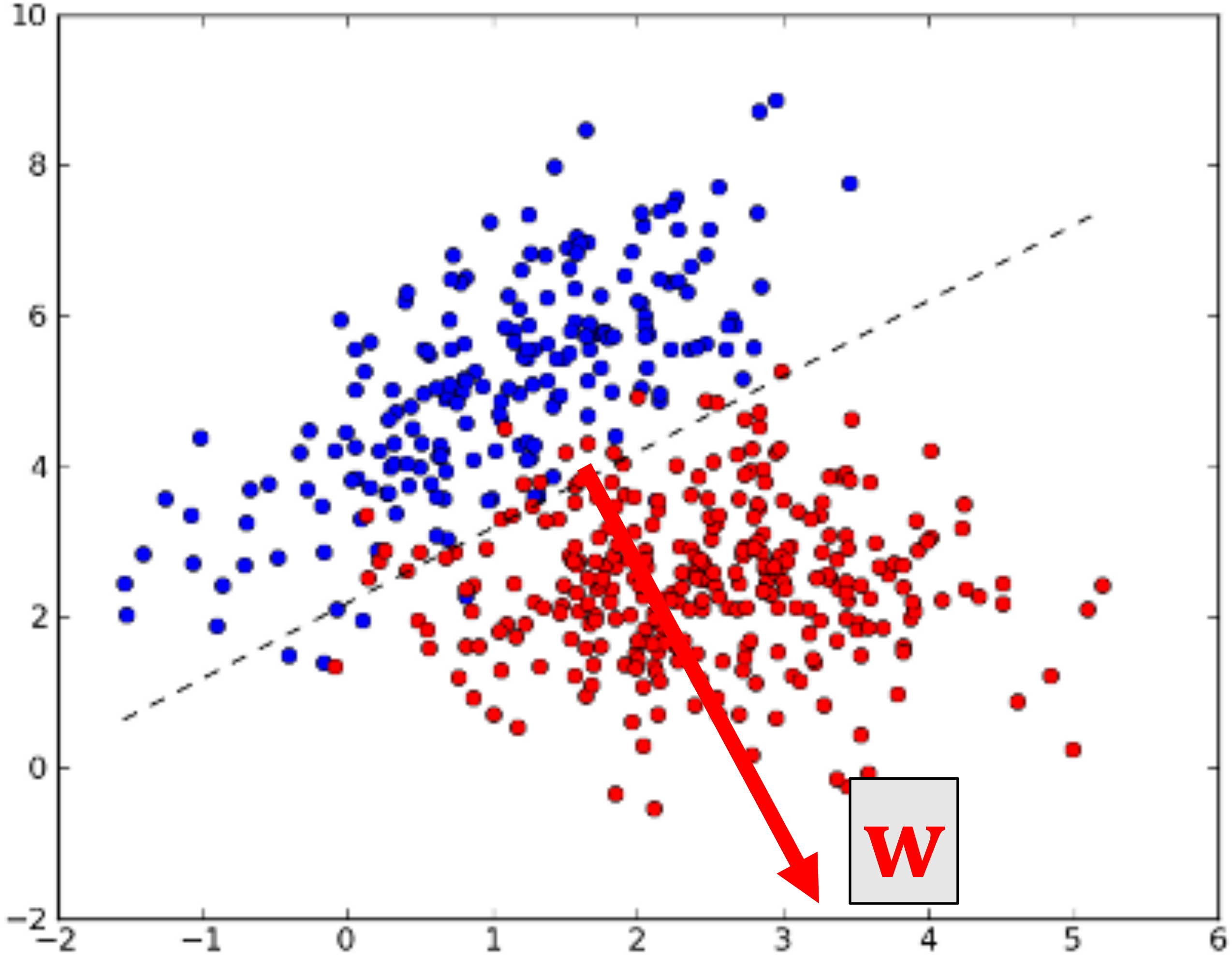


Learn a vector $\mathbf{w}$ such that

- If $y_j = +1$, then $\mathbf{w}^T \mathbf{x}_j > 0$.

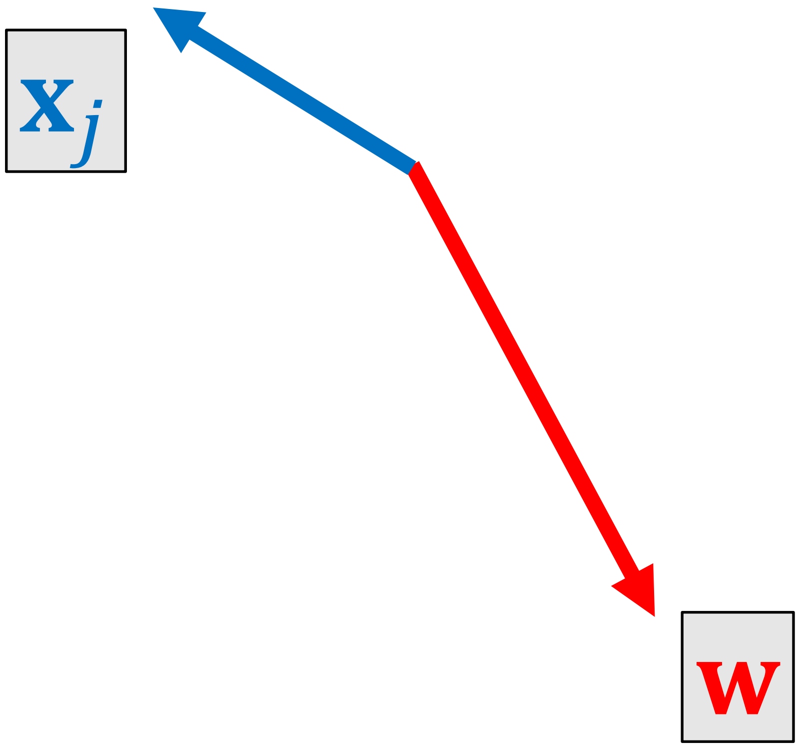


Linear Classifier

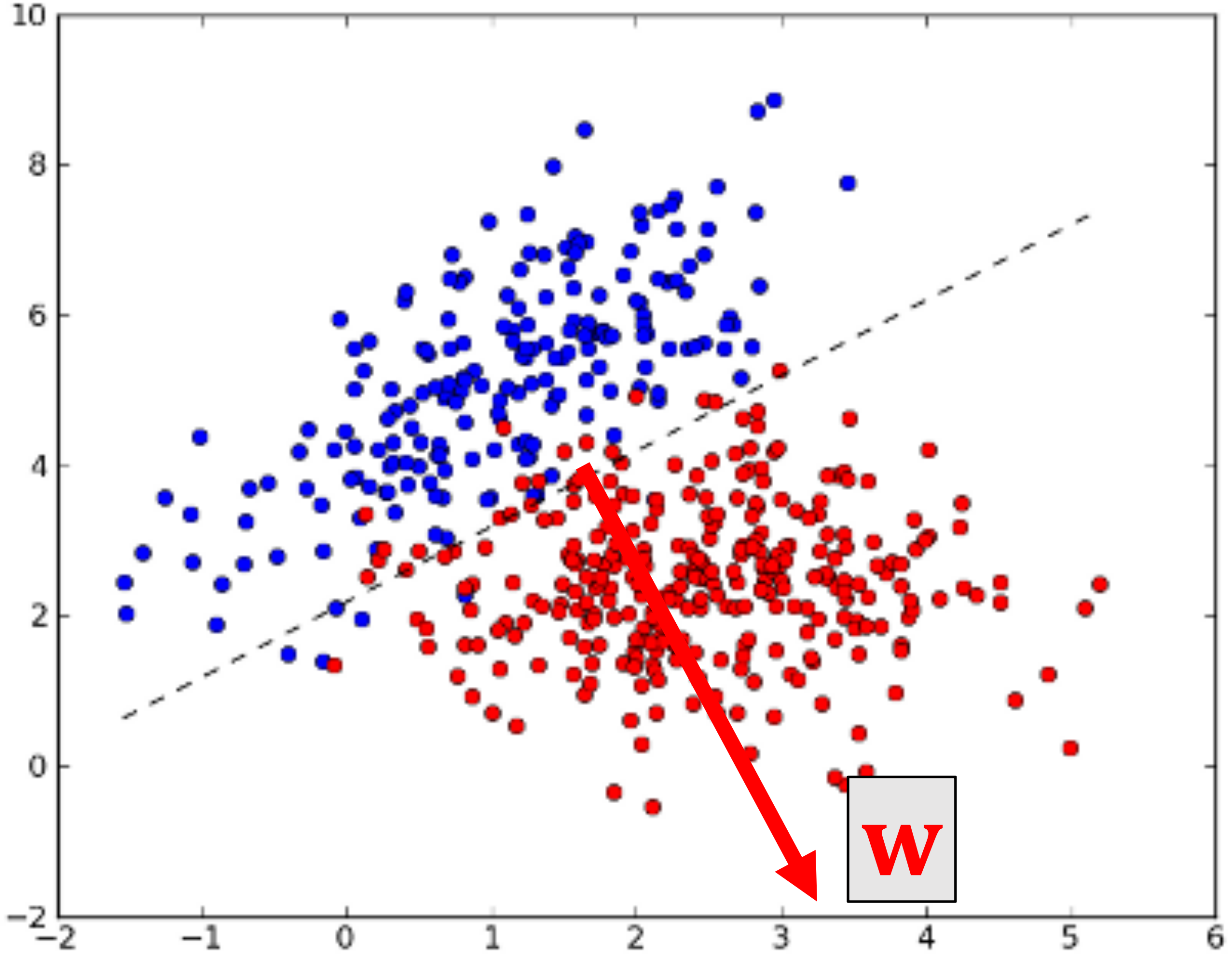


Learn a vector $\mathbf{w}$ such that

- If $y_j = +1$, then $\mathbf{w}^T \mathbf{x}_j > 0$.
- If $y_j = -1$, then $\mathbf{w}^T \mathbf{x}_j < 0$.



Linear Classifier



Learn a vector $\mathbf{w}$ such that

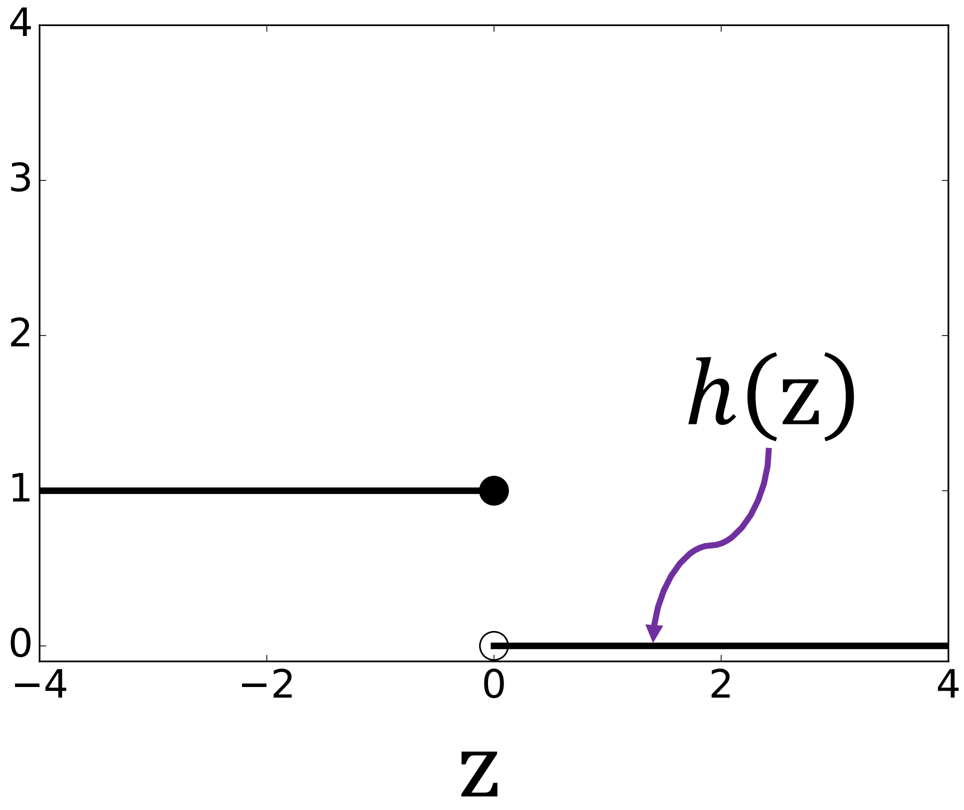
- If $y_j = +1$, then $\mathbf{w}^T \mathbf{x}_j > 0$.
- If $y_j = -1$, then $\mathbf{w}^T \mathbf{x}_j < 0$.

Key Idea:

Encourage $y_j \mathbf{w}^T \mathbf{x}_j$ to be positive

Directly Minimize the Classification Error?

Minimize $\sum_j h\left(y_j \mathbf{w}^T \mathbf{x}_j\right)$, where $h(\mathrm{z}) = \begin{cases} 1, & \text{if } z < 0; \\ 0, & \text{if } z \geq 0. \end{cases}$



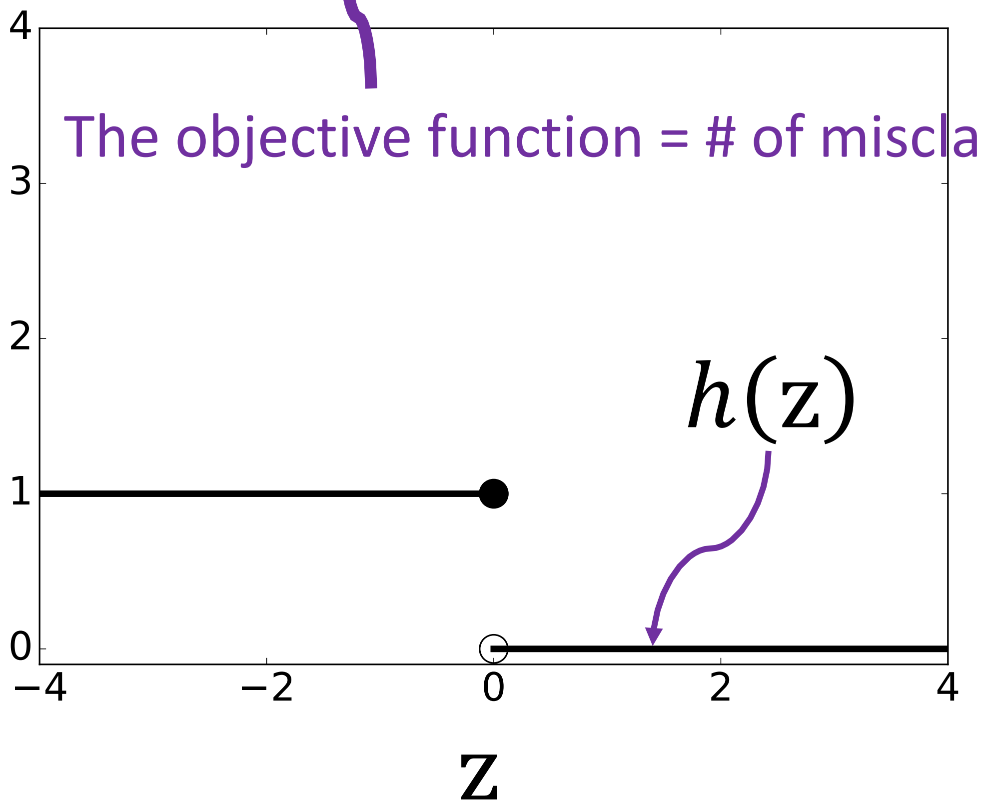
Key Idea:

Encourage $y_j \mathbf{w}^T \mathbf{x}_j$ to be positive

Directly Minimize the Classification Error?

Minimize $\sum_j h\left(y_j \mathbf{w}^T \mathbf{x}_j\right)$, where $h(\mathrm{z}) = \begin{cases} 1, & \text{if } z < 0; \\ 0, & \text{if } z \geq 0. \end{cases}$

The objective function = # of misclassified training samples



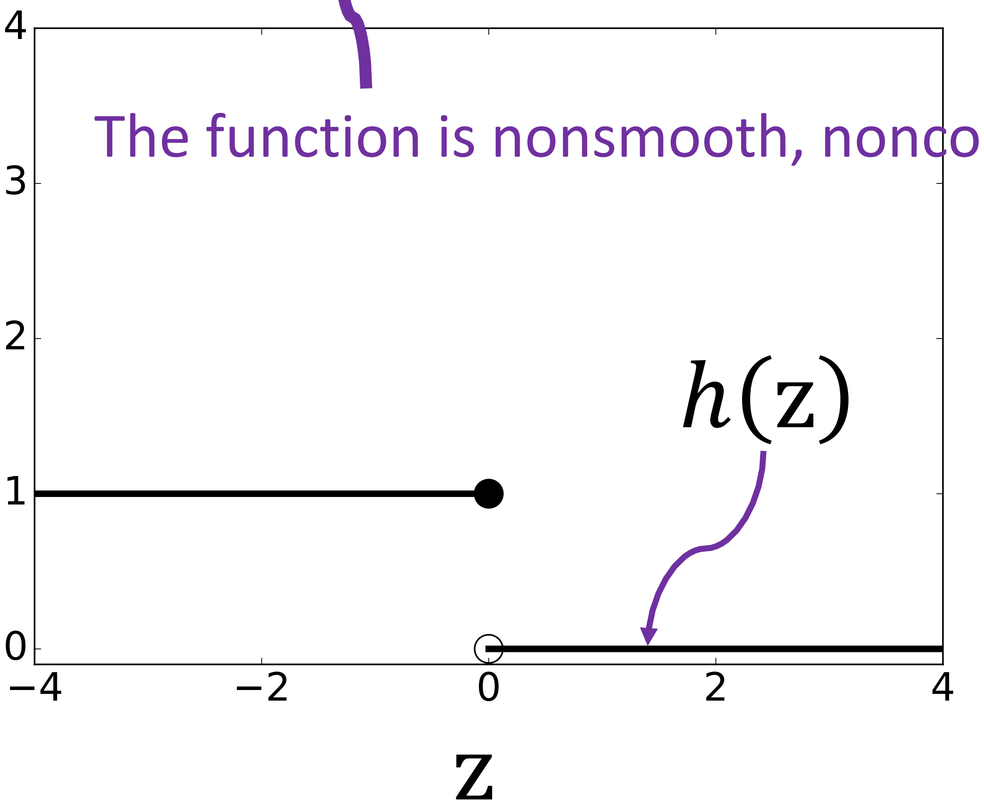
Key Idea:

Encourage $y_j \mathbf{w}^T \mathbf{x}_j$ to be positive

Directly Minimize the Classification Error?

Minimize $\sum_j h\left(y_j \mathbf{w}^T \mathbf{x}_j\right)$, where $h(z) = \begin{cases} 1, & \text{if } z < 0; \\ 0, & \text{if } z \geq 0. \end{cases}$

The function is nonsmooth, nonconvex, and hard to optimize.

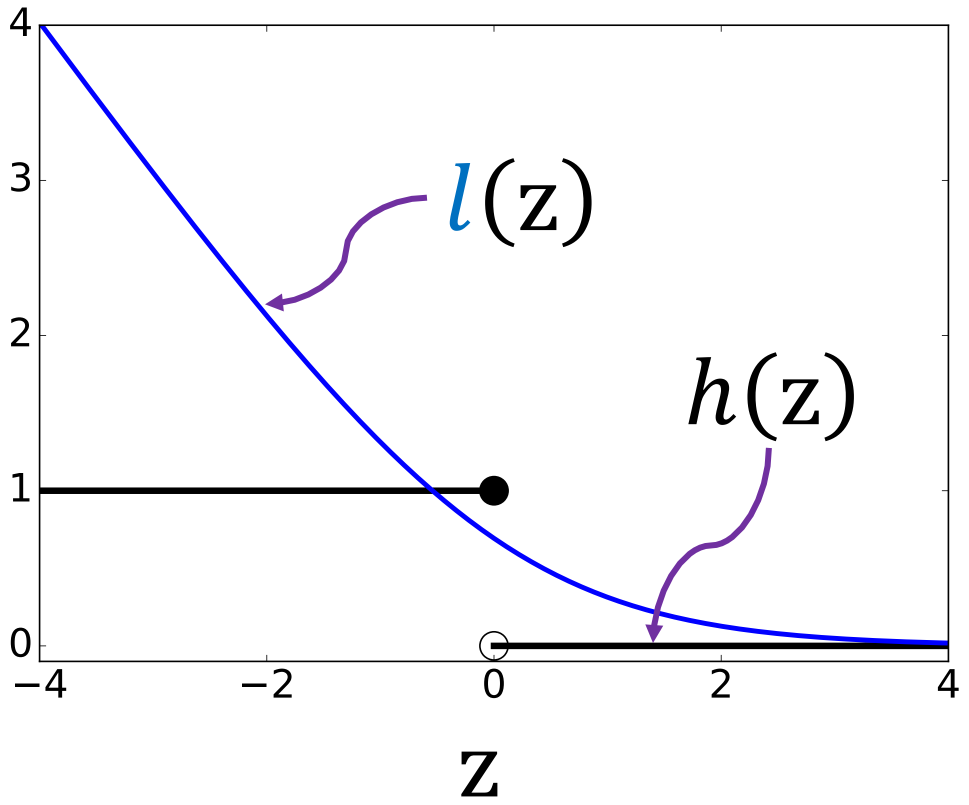


Key Idea:

Encourage $y_j \mathbf{w}^T \mathbf{x}_j$ to be positive

Logistic Regression

Minimize $\sum_j l(y_j \mathbf{w}^T \mathbf{x}_j)$, where $l(\mathrm{z}) = \log(1 + e^{-z})$.



Key Idea:

Encourage $y_j \mathbf{w}^T \mathbf{x}_j$ to be positive

Logistic Regression

Tasks

Methods

Algorithms

Logistic Regression

Logistic regression: $\min_{\mathbf{w}} \frac{1}{n}\sum_{j=1}^{n} l\left(y_j \mathbf{w}^T \mathbf{x}_j\right)$, where $l(\mathrm{z}) = \log(1 + e^{-z})$.

Tasks	Methods	Algorithms
Binary Classification	Logistic Regression	Gradient Descent (GD)
Multi-Class Classification	SVM	Accelerated GD
	Neural Networks	Stochastic GD

Logistic Regression

Logistic regression: $\min_{\mathbf{w}} \sum_{j=1}^{n} l\left(y_j \mathbf{w}^T \mathbf{x}_j\right)$, where $l(\mathrm{z}) = \log(1 + e^{-z})$.

Gradient at $\mathbf{w}_t$: $\mathbf{g}_t = \sum_{j=1}^{n} \frac{-y_j \, \mathbf{x}_j}{1+\exp(y_j \mathbf{w}_t^T \mathbf{x}_j)}$.

- $\frac{\partial \, l(z)}{\partial \, z} = \frac{-e^{-z}}{1+e^{-z}} = -\frac{1}{1+e^{z}}$.

- Chain rule:
  - Define $z_j = y_j \mathbf{w}^T \mathbf{x}_j$
  - $\frac{\partial \, l\left(y_j \mathbf{w}^T \mathbf{x}_j\right)}{\partial \, \mathbf{w}} = \frac{\partial z_j}{\partial \, \mathbf{w}} \cdot \frac{\partial \, l(z_j)}{\partial z_j} = \left(y_j \mathbf{x}_j\right)\left(-\frac{1}{1+e^{z_j}}\right) = -\frac{y_j \, \mathbf{x}_j}{1+\exp(-y_j \mathbf{w}_t^T \mathbf{x}_j)}$

Logistic Regression

Logistic regression: $\min_{\mathbf{w}} \sum_{j=1}^{n} l\left(y_j \mathbf{w}^T \mathbf{x}_j\right)$, where $l(\mathrm{z}) = \log(1 + e^{-z})$.

Gradient at $\mathbf{w}_t$: $\mathbf{g}_t = \sum_{j=1}^{n} \frac{-y_j \, \mathbf{x}_j}{1+\exp(y_j \mathbf{w}_t^T \mathbf{x}_j)}$.

- We have shown: $\frac{\partial \, l(y_j \mathbf{w}^T \mathbf{x}_j)}{\partial \, \mathbf{w}} = -\frac{y_j \, \mathbf{x}_j}{1+\exp(-y_j \mathbf{w}_t^T \mathbf{x}_j)}$

- Objective function: $f(\mathbf{w}) = \sum_j l\left(y_j \mathbf{w}^T \mathbf{x}_j\right)$.

- $\frac{\partial f(\mathbf{w})}{\partial \, \mathbf{w}} = \sum_j \frac{\partial \, l(y_j \mathbf{w}^T \mathbf{x}_j)}{\partial \, \mathbf{w}} = -\sum_j \frac{y_j \, \mathbf{x}_j}{1+\exp(-y_j \mathbf{w}_t^T \mathbf{x}_j)}$.

Logistic Regression

Logistic regression: $\min_{\mathbf{w}} \sum_{j=1}^{n} l\left(y_j \mathbf{w}^T \mathbf{x}_j\right)$, where $l(z) = \log(1 + e^{-z})$.

Gradient at $\mathbf{w}_t$: $\mathbf{g}_t = \sum_{j=1}^{n} \frac{-y_j \mathbf{x}_j}{1+\exp(y_j \mathbf{w}_t^T \mathbf{x}_j)}$.

GD repeat:

1. Compute gradient: $\mathbf{g}_t$
2. Update: $\mathbf{w}_{t+1} = \mathbf{w}_t - \alpha\, \mathbf{g}_t$

Tune the step size (learning rate) α

Algorithms

- Gradient Descent (GD)
- Accelerated GD
- Stochastic GD

Logistic Regression

Logistic regression: $\min_{\mathbf{w}} \sum_{j=1}^{n} l\left(y_j \mathbf{w}^T \mathbf{x}_j\right)$, where $l(\mathrm{z}) = \log(1 + e^{-z})$.

Gradient at $\mathbf{w}_t$: $\mathbf{g}_t = \sum_{j=1}^{n} \frac{-y_j \mathbf{x}_j}{1+\exp(y_j \mathbf{w}_t^T \mathbf{x}_j)}$.

AGD repeat:

1. Compute gradient: $\mathbf{g}_t$
2. Update momentum: $\mathbf{v}_{t+1} = \beta \mathbf{v}_t + \mathbf{g}_t$
3. Update: $\mathbf{w}_{t+1} = \mathbf{w}_t - \alpha \, \mathbf{v}_{t+1}$

Tune α and β $(0 \leq \beta < 1)$

Algorithms

Gradient Descent (GD)

Accelerated GD

Stochastic GD

Logistic Regression

Logistic regression: $\min_{\mathbf{w}} \sum_{j=1}^{n} l\left(y_j \mathbf{w}^T \mathbf{x}_j\right)$, where $l(\mathrm{z}) = \log(1 + e^{-z})$.

Gradient at $\mathbf{w}_t$: $\mathbf{g}_t = \frac{1}{n} \sum_{j=1}^{n} \tilde{\mathbf{g}}_{t,j}$, where $\tilde{\mathbf{g}}_{t,j} = \frac{-n y_j \mathbf{x}_j}{1+\exp(y_j \mathbf{w}_t^T \mathbf{x}_j)}$.

Per-iteration time complexity is $O(nd)$

- $O(d)$ time for computing $\mathbf{w}_t^T \mathbf{x}_j$
- $O(d)$ time for computing $\tilde{\mathbf{g}}_{t,j}$
- $O(nd)$ time for computing $\mathbf{g}_t = \frac{1}{n} \sum_j \tilde{\mathbf{g}}_{t,j}$

Algorithms

Gradient Descent (GD)

Accelerated GD

Stochastic GD

Logistic Regression

Logistic regression: $\min_{\mathbf{w}} \sum_{j=1}^{n} l\left(y_j \mathbf{w}^T \mathbf{x}_j\right)$, where $l(z) = \log(1 + e^{-z})$.

Gradient at $\mathbf{w}_t$: $\mathbf{g}_t = \frac{1}{n} \sum_{j=1}^{n} \tilde{\mathbf{g}}_{t,j}$, where $\tilde{\mathbf{g}}_{t,j} = \frac{-n y_j \mathbf{x}_j}{1+\exp(y_j \mathbf{w}_t^T \mathbf{x}_j)}$.

The stochastic gradient is close to the full gradient:

$$\mathbf{g}_t = \mathbb{E}_j\left[\tilde{\mathbf{g}}_{t,j}\right],$$

where j is randomly sampled from $\{1, \cdots, n\}$.

Algorithms

- Gradient Descent (GD)
- Accelerated GD
- Stochastic GD

Logistic Regression

Logistic regression: $\min_{\mathbf{w}} \sum_{j=1}^{n} l\left(y_j \mathbf{w}^T \mathbf{x}_j\right)$, where $l(z) = \log(1 + e^{-z})$.

Gradient at $\mathbf{w}_t$: $\mathbf{g}_t = \frac{1}{n} \sum_{j=1}^{n} \tilde{\mathbf{g}}_{t,j}$, where $\tilde{\mathbf{g}}_{t,j} = \frac{-n y_j \mathbf{x}_j}{1+\exp(y_j \mathbf{w}_t^T \mathbf{x}_j)}$.

SGD repeats

1. Randomly draw j from $\{1, 2, \cdots, n\}$.
2. Compute the stochastic gradient $\tilde{\mathbf{g}}_{t,j}$.
3. Update: $\mathbf{w}_{t+1} = \mathbf{w}_t - \alpha\, \tilde{\mathbf{g}}_{t,j}$.

Per-iteration time complexity is $O(d)$.

Algorithms

Gradient Descent (GD)

Accelerated GD

Stochastic GD

Logistic Regression

Logistic regression: $\min_{\mathbf{w}} \sum_{j=1}^{n} l\left(y_j \mathbf{w}^T \mathbf{x}_j\right)$, where $l(z) = \log(1 + e^{-z})$.

Gradient at $\mathbf{w}_t$: $\mathbf{g}_t = \frac{1}{n} \sum_{j=1}^{n} \tilde{\mathbf{g}}_{t,j}$, where $\tilde{\mathbf{g}}_{t,j} = \frac{-n y_j \mathbf{x}_j}{1+\exp(y_j \mathbf{w}_t^T \mathbf{x}_j)}$.

Accelerated SGD repeats

1. Randomly draw j from $\{1, 2, \cdots, n\}$.
2. Compute the stochastic gradient $\tilde{\mathbf{g}}_{t,j}$.
3. Update momentum: $\mathbf{v}_{t+1} = \beta \mathbf{v}_t + \tilde{\mathbf{g}}_{t,j}$.
4. Update: $\mathbf{w}_{t+1} = \mathbf{w}_t - \alpha \mathbf{v}_{t+1}$.

Algorithms

Gradient Descent (GD)

Accelerated GD

Stochastic GD

Logistic Regression

Logistic regression: $\min_{\mathbf{w}} \sum_{j=1}^{n} l\left(y_j \mathbf{w}^T \mathbf{x}_j\right)$, where $l(\mathrm{z}) = \log(1+e^{-z})$.

Gradient at $\mathbf{w}_t$: $\mathbf{g}_t = \frac{1}{n}\sum_{j=1}^{n} \tilde{\mathbf{g}}_{t,j}$, where $\tilde{\mathbf{g}}_{t,j} = \frac{-n y_j \mathbf{x}_j}{1+\exp(y_j \mathbf{w}_t^T \mathbf{x}_j)}$.

Output of SGD:

- Option 1: output the last iteration $\mathbf{w}_{t+1}$
- Option 2: output the average of $\mathbf{w}$ produced by the last tens of iteration.

Algorithms

Gradient Descent (GD)

Accelerated GD

Stochastic GD

Training and Prediction

- Training:

$$\mathbf{w}^{\star} = \underset{\mathbf{w}}{\operatorname{argmin}} \sum_j l\left(y_j \mathbf{w}^T \mathbf{x}_j\right), \text{ where } l(\mathrm{z}) = \log(1 + e^{-z}).$$

- For a test feature vector $\mathbf{x}' \in \mathbb{R}^d$, make prediction by

$$\operatorname{sign}\left(\mathbf{x}'^T \mathbf{w}^{\star}\right).$$

Summary

- Logistic regression model for *linear binary* classification.

$$\mathbf{w}^\star = \underset{\mathbf{w}}{\operatorname{argmin}} \sum_j l\left(y_j \mathbf{w}^T \mathbf{x}_j\right), \text{ where } l(z) = \log(1 + e^{-z}).$$

- Compute the gradient using vector derivatives and the chain rule.
- Gradient-based algorithms: GD, AGD, SGD, etc.
- Make prediction using $\operatorname{sign}\left(\mathbf{x}'^T \mathbf{w}^\star\right)$.

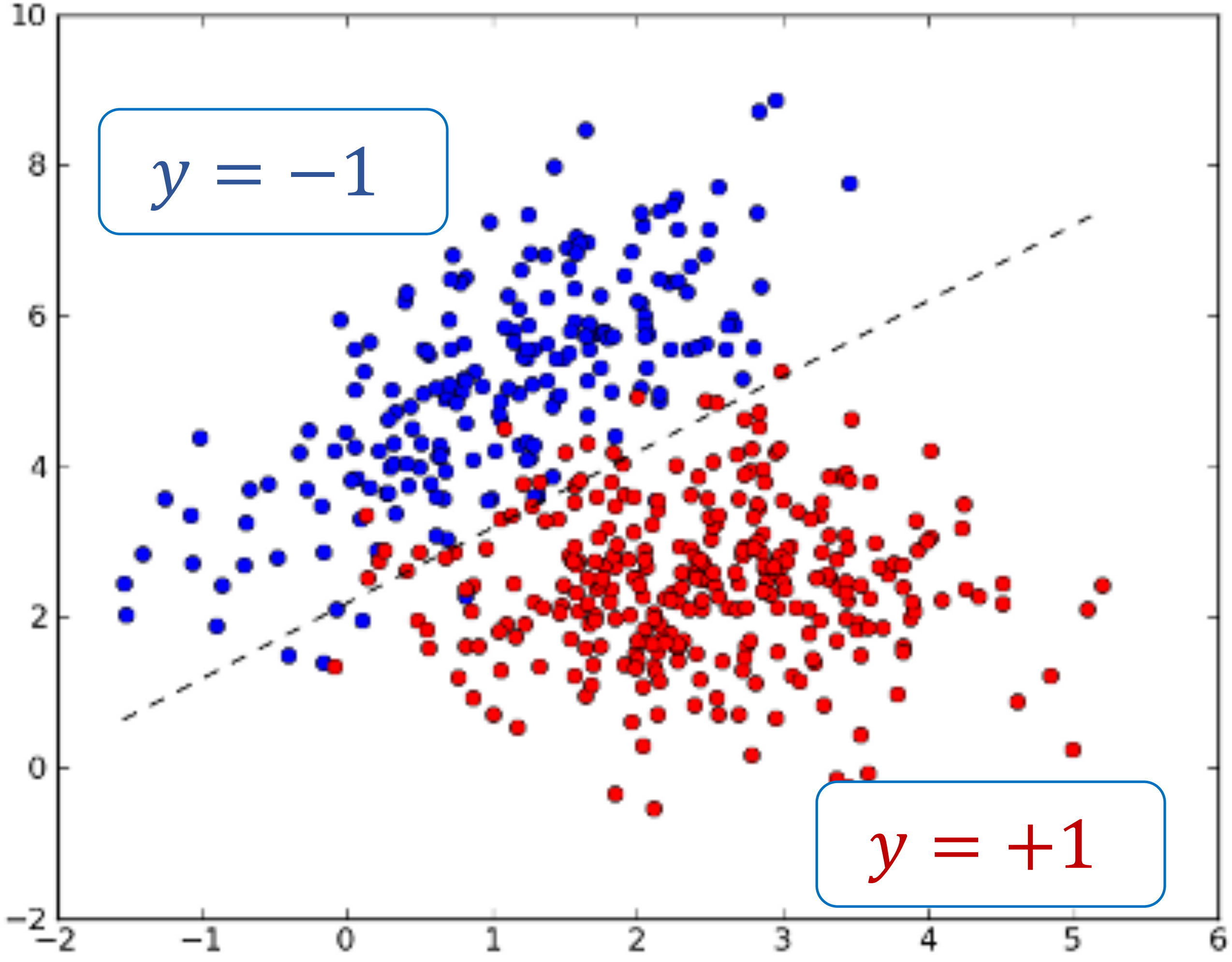
Evaluate Binary Classification

Evaluate Binary Classification

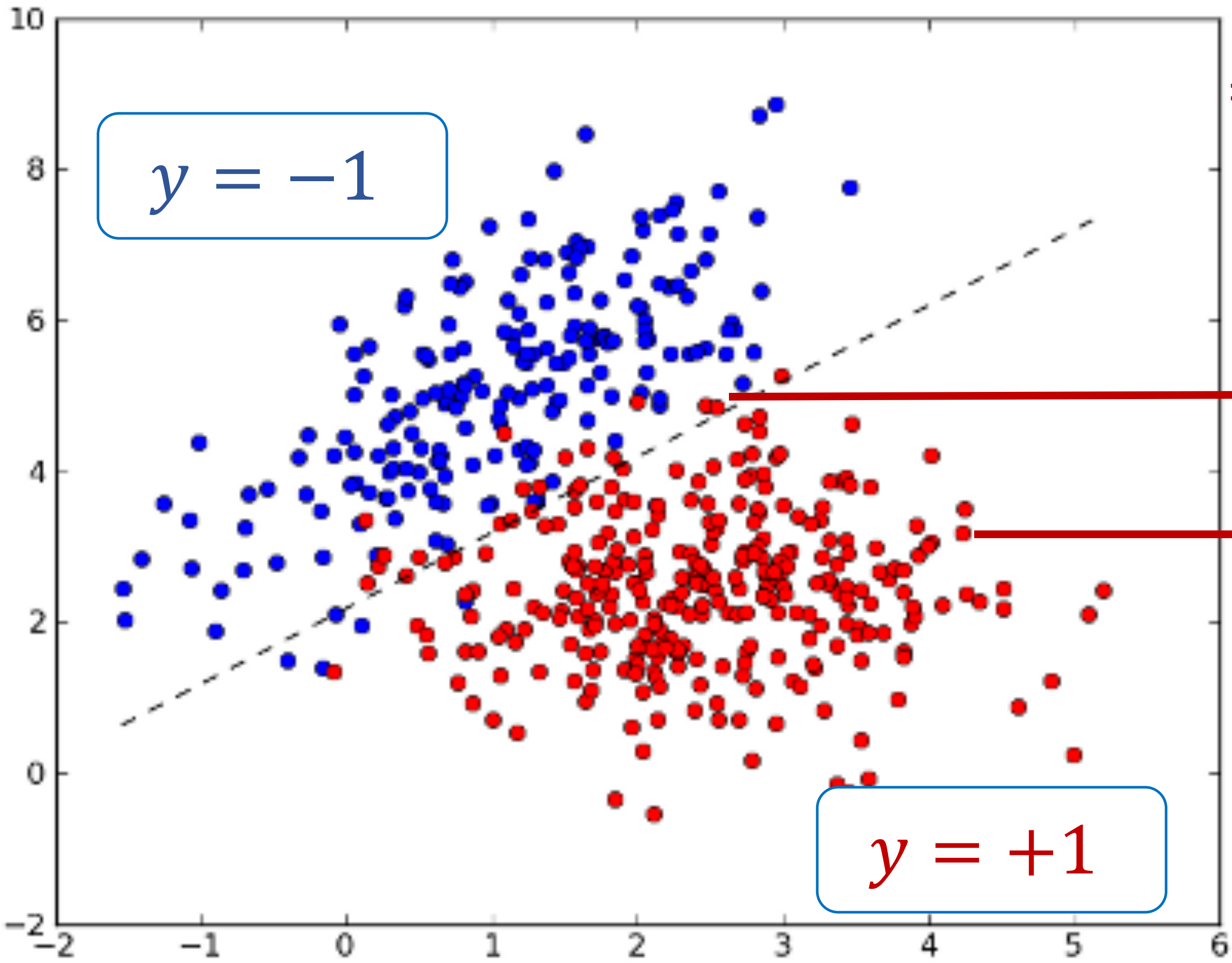
- Error Rate $= \dfrac{\#\ \text{Classification Errors}}{\#\ \text{Samples}}$
- Accuracy = 1 - Error Rate

Error rate and **Accuracy** are not meaningful in class-imbalance problems.

Evaluate Binary Classification



Evaluate Binary Classification



True Positive Rate (TPR)

$$= \frac{\#\mathrm{TP}}{\#\mathrm{TP} + \#\mathrm{FN}}$$

False Negative (FN)

True Positive (TP)

Evaluate Binary Classification

False Positive Rate (FPR)

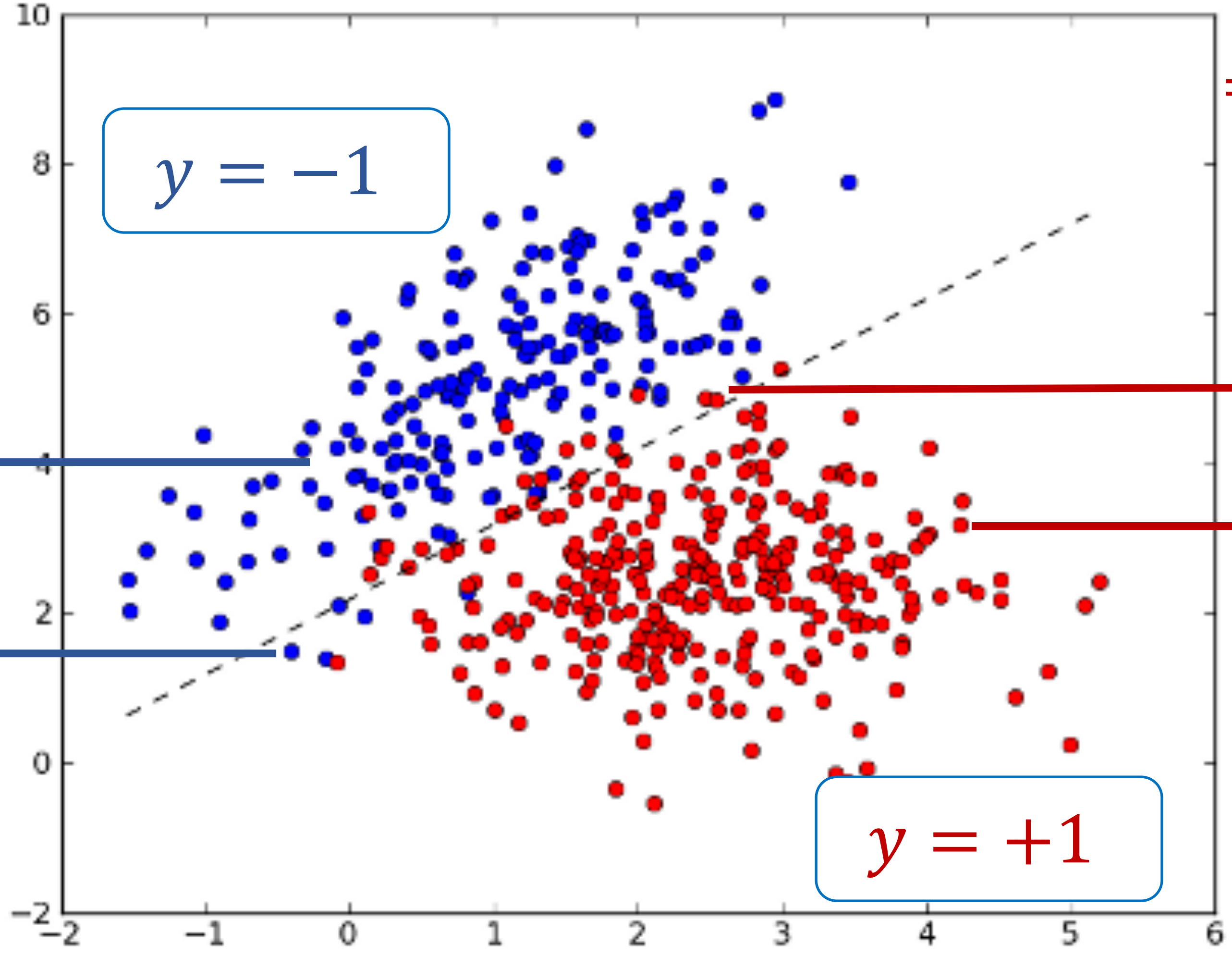
$$= \frac{\#\mathrm{FP}}{\#\mathrm{FP} + \#\mathrm{TN}}$$

True Positive Rate (TPR)

$$= \frac{\#\mathrm{TP}}{\#\mathrm{TP} + \#\mathrm{FN}}$$

True Negative (TN)

False Positive (FP)



False Negative (FN)

True Positive (TP)

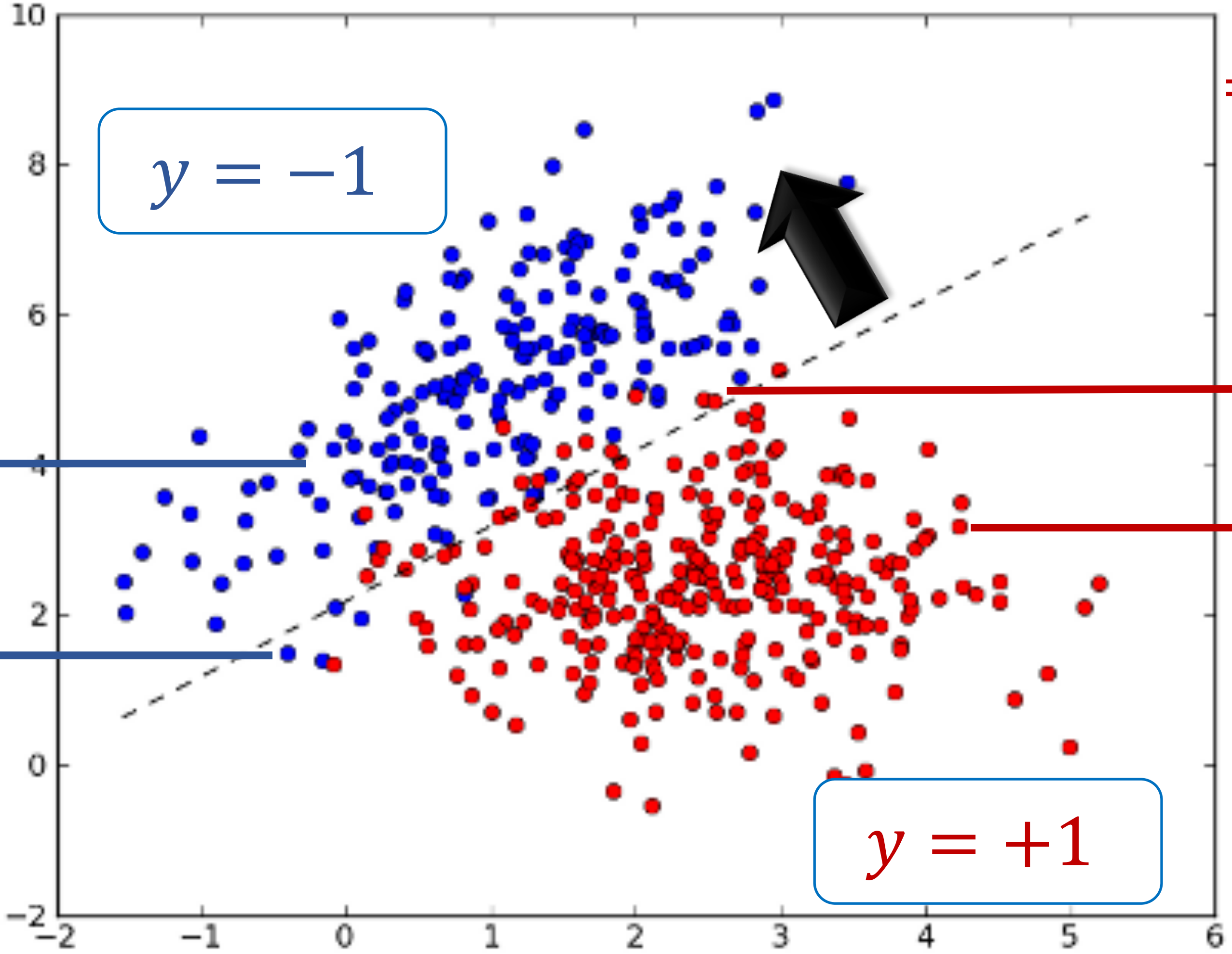
Evaluate Binary Classification

False Positive Rate (FPR)

$$= \frac{\#FP}{\#FP + \#TN} \nearrow$$

True Positive Rate (TPR)

$$= \frac{\#TP}{\#TP + \#FN} \nearrow$$



True Negative (TN)

False Positive (FP)

False Negative (FN)

True Positive (TP)

Evaluate Binary Classification

False Positive Rate (FPR)

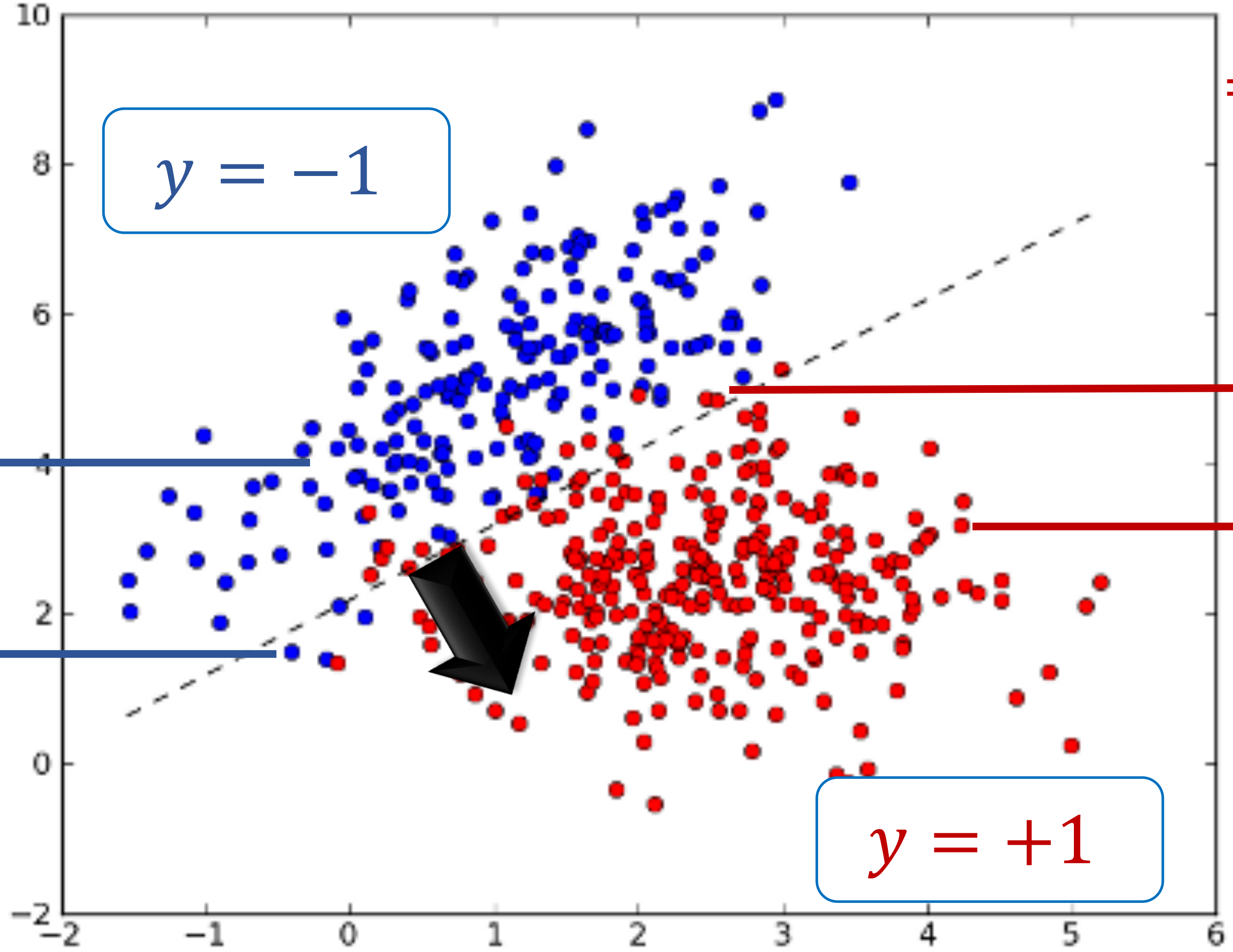
$$= \frac{\#\mathrm{FP}}{\#\mathrm{FP} + \#\mathrm{TN}}$$

True Positive Rate (TPR)

$$= \frac{\#\mathrm{TP}}{\#\mathrm{TP} + \#\mathrm{FN}}$$

True Negative (TN)

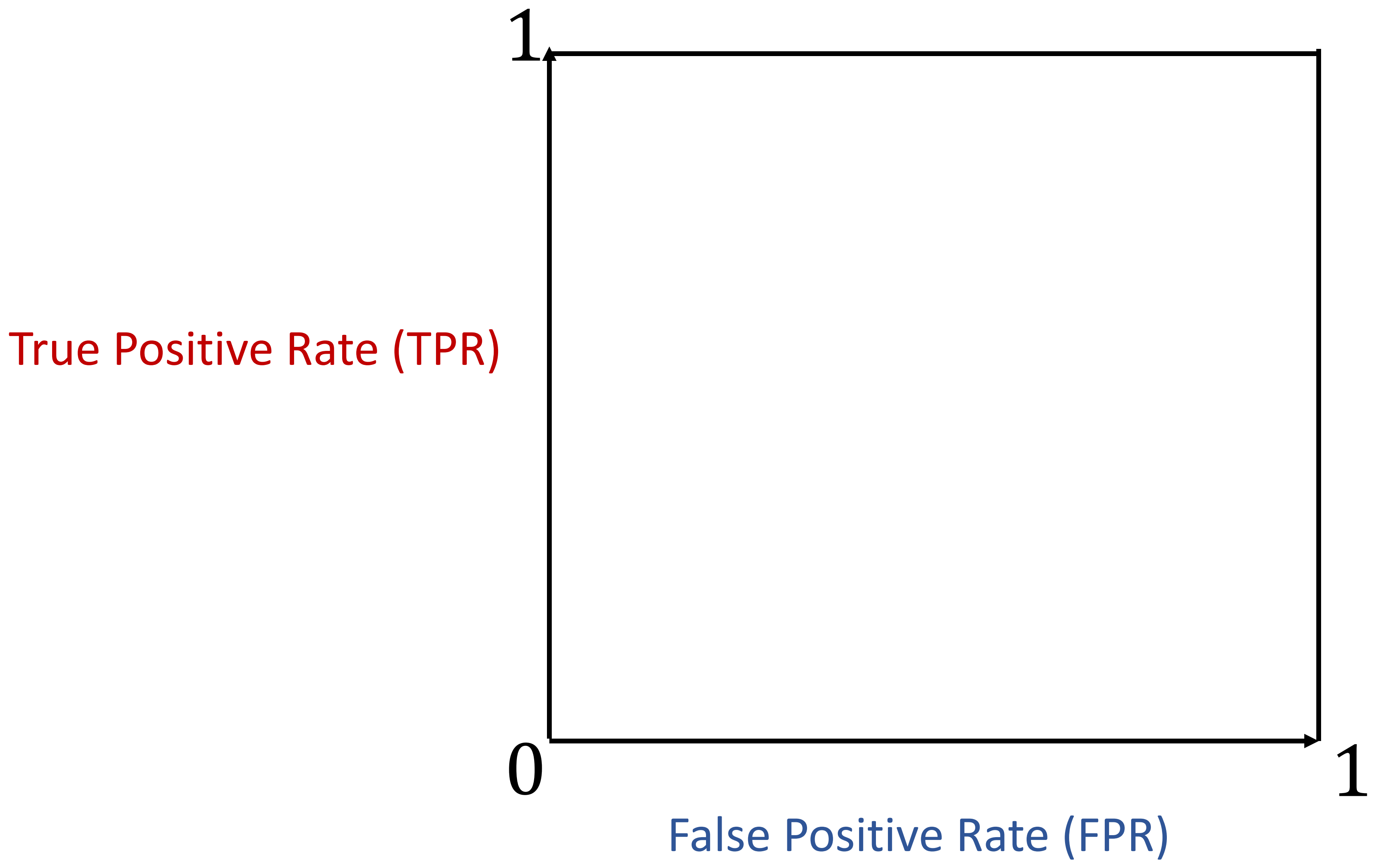
False Positive (FP)



False Negative (FN)

True Positive (TP)

Receiver Operating Characteristic (ROC) Curve

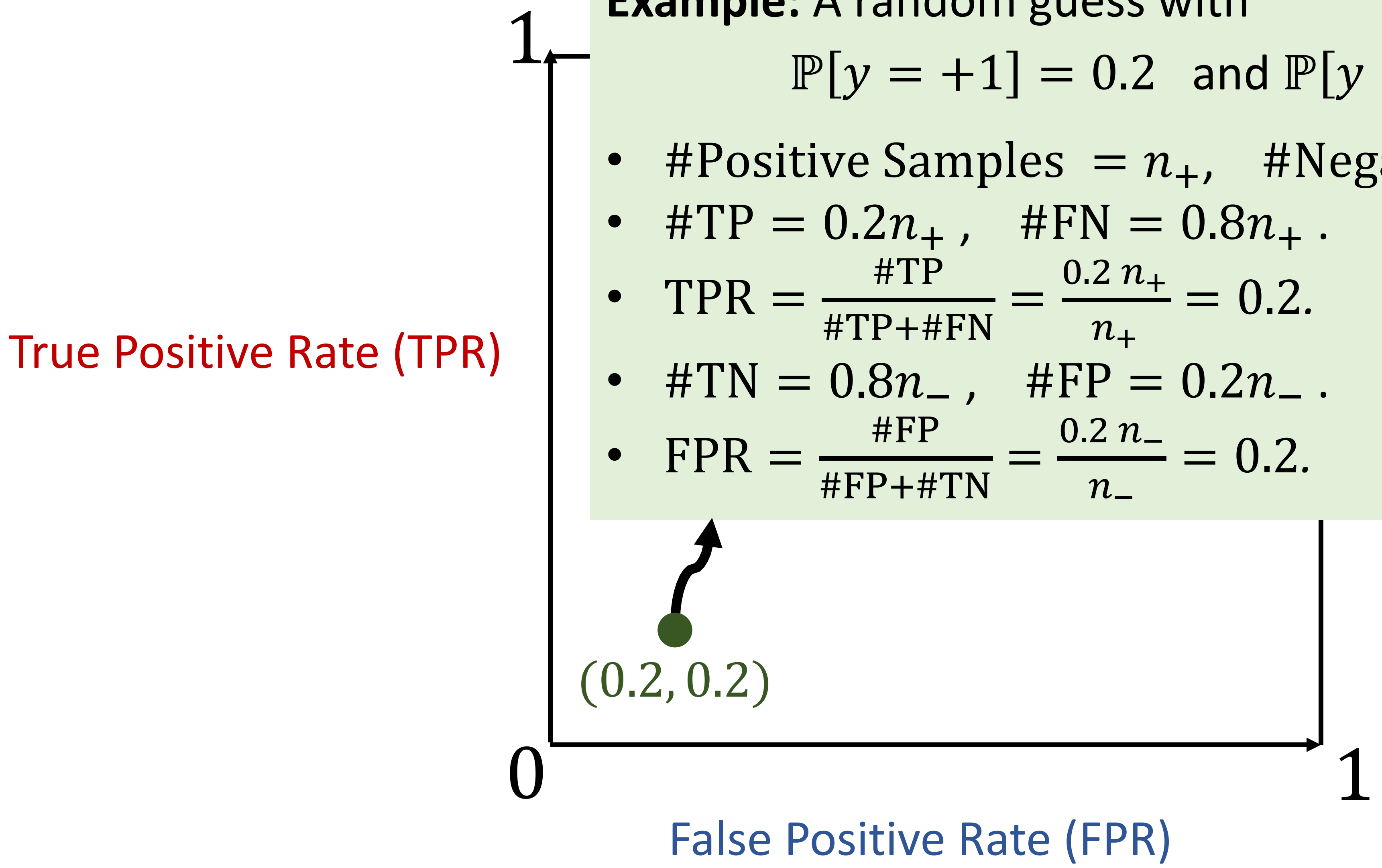


Receiver Operating Characteristic (ROC) Curve

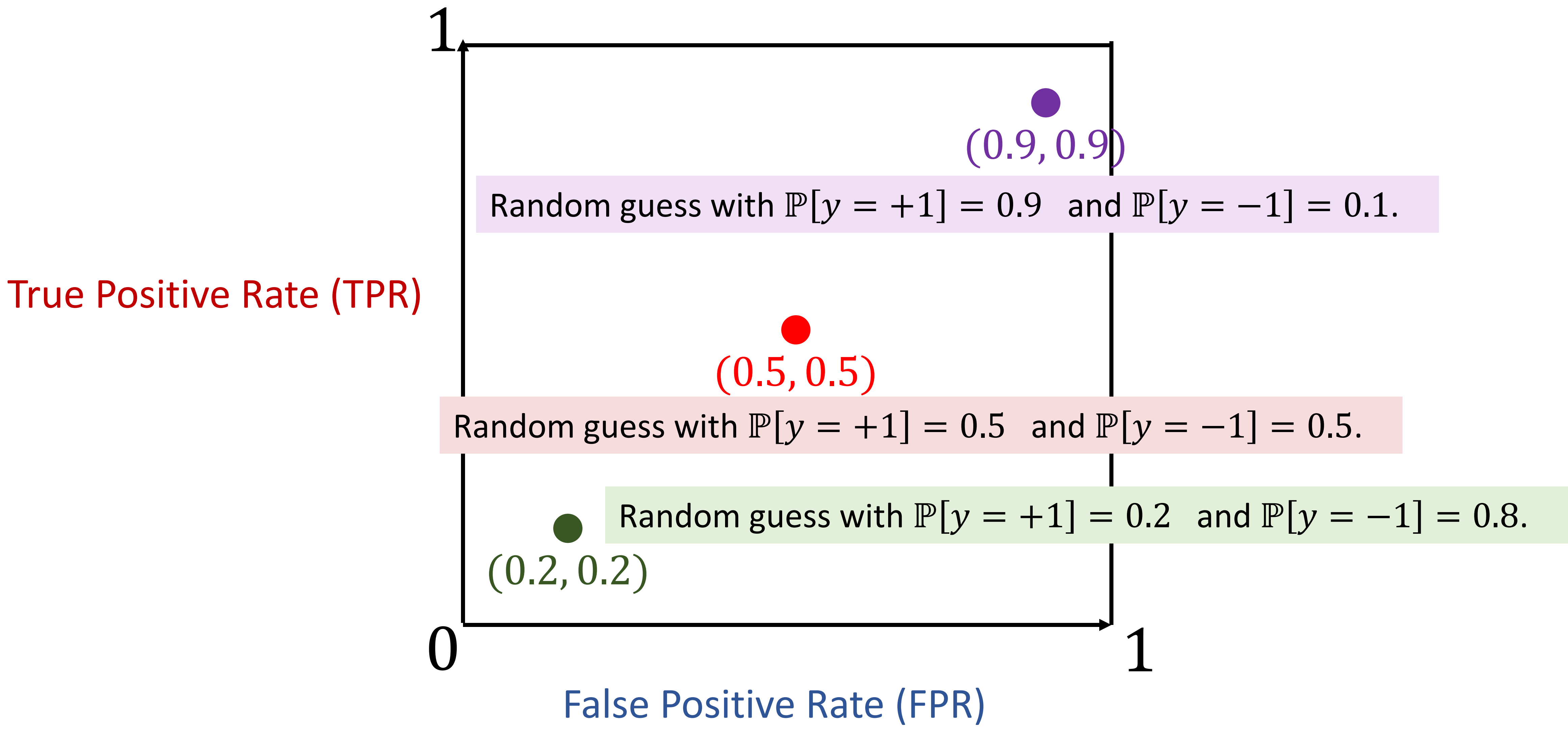
Example: A random guess with

$$\mathbb{P}[y = +1] = 0.2 \quad \text{and } \mathbb{P}[y = -1] = 0.8.$$

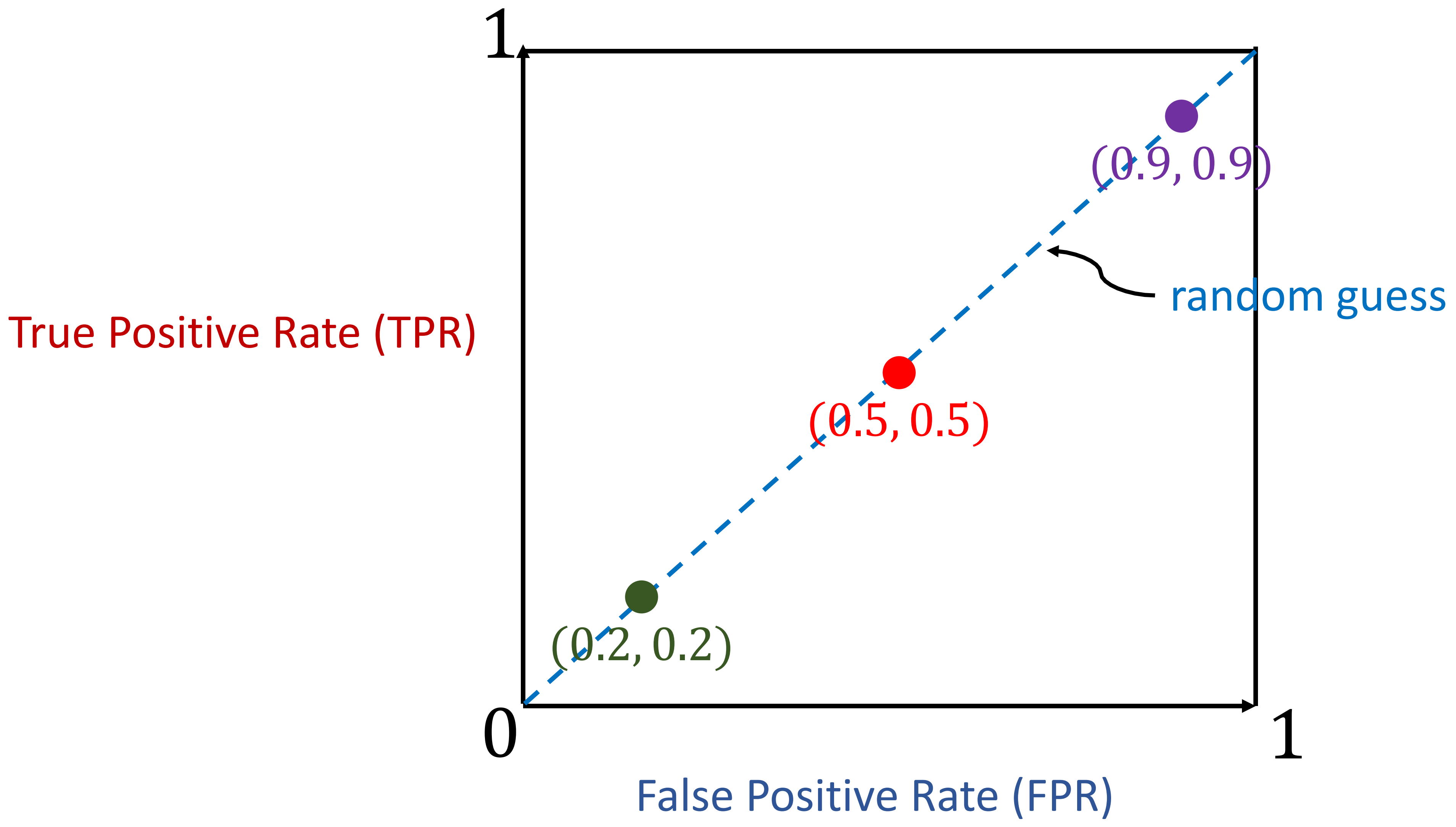
- #Positive Samples $= n_+$, #Negative $Samples$ $= n_-$.
- $\#\text{TP} = 0.2 n_+$, $\#\text{FN} = 0.8 n_+$.
- $\text{TPR} = \frac{\#\text{TP}}{\#\text{TP} + \#\text{FN}} = \frac{0.2\, n_+}{n_+} = 0.2.$
- $\#\text{TN} = 0.8 n_-$, $\#\text{FP} = 0.2 n_-$.
- $\text{FPR} = \frac{\#\text{FP}}{\#\text{FP} + \#\text{TN}} = \frac{0.2\, n_-}{n_-} = 0.2.$



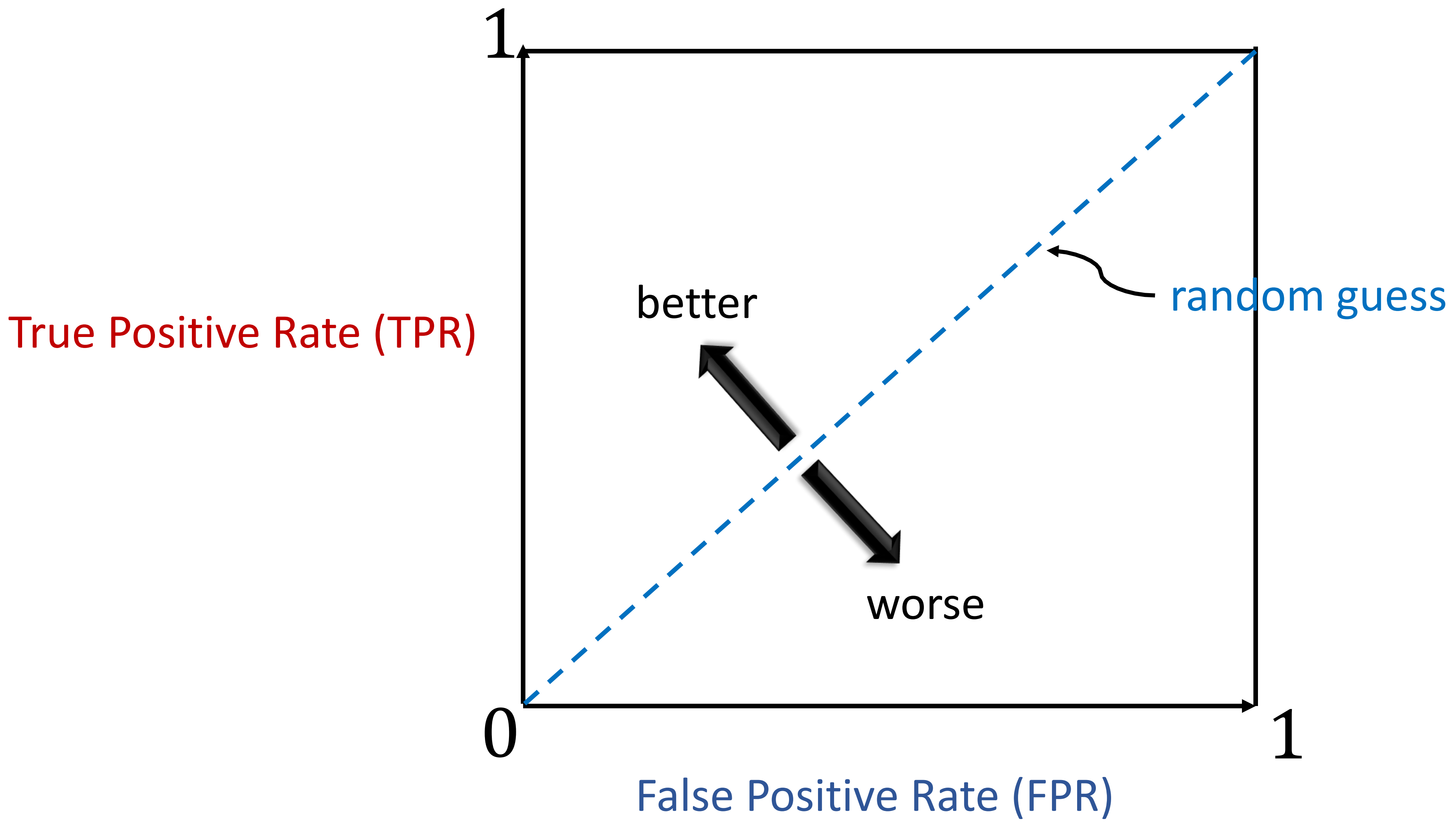
Receiver Operating Characteristic (ROC) Curve



Receiver Operating Characteristic (ROC) Curve



Receiver Operating Characteristic (ROC) Curve



Receiver Operating Characteristic (ROC) Curve