Binary Classification

- We will focus on the binary classification problem to build up our intuition about machine learning.
- Binary classification is closely related to binary hypothesis testing, except that we are not provided with conditional probability models for the observation.
- Instead, we have a dataset consisting of labeled examples.
- While we can use these examples to estimate the probability models, our goal is to design a decision rule to classify new examples.
  - → Ex: Design a classifier for cat and dog images.
  - → Ex: Classify MRI tumor images as malignant or benign.
  - → Ex: Automatically sort harvested fruit into quality bins.

- **Binary Classification Framework:**

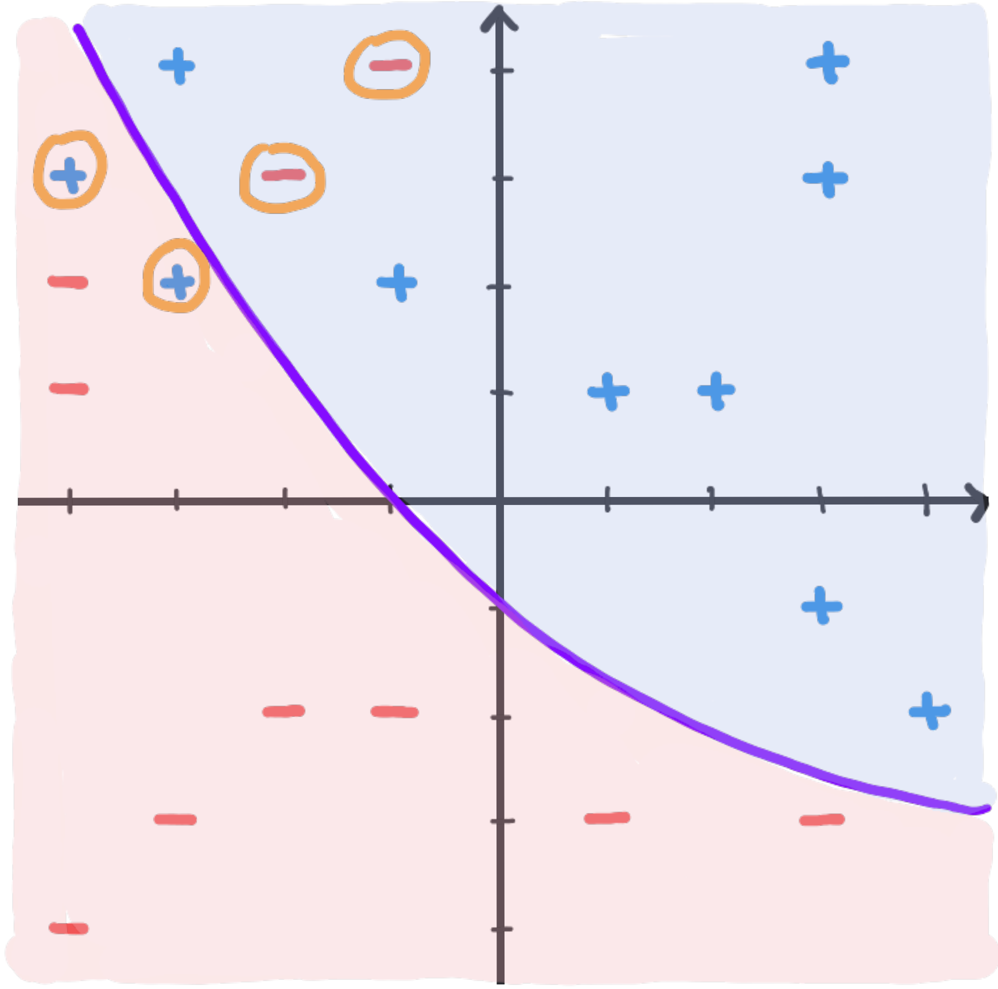
→ We have a dataset consisting of a collection of labeled examples of the form $(\underline{x}_i, y_i)$ for $i = 1, \dots, n$ where

* $\underline{x}_i = \begin{bmatrix} x_{i,1} \\ \vdots \\ x_{i,d} \end{bmatrix}$ is the d-dimensional feature vector

* $y_i \in \{+1, -1\}$ is the label

→ Our task is to design a classifier (or decision rule) $D(\underline{x})$ that outputs either $+1$ or -1 for any possible input vector.

→ We measure performance using the error rate, which is the fraction of misclassified points.



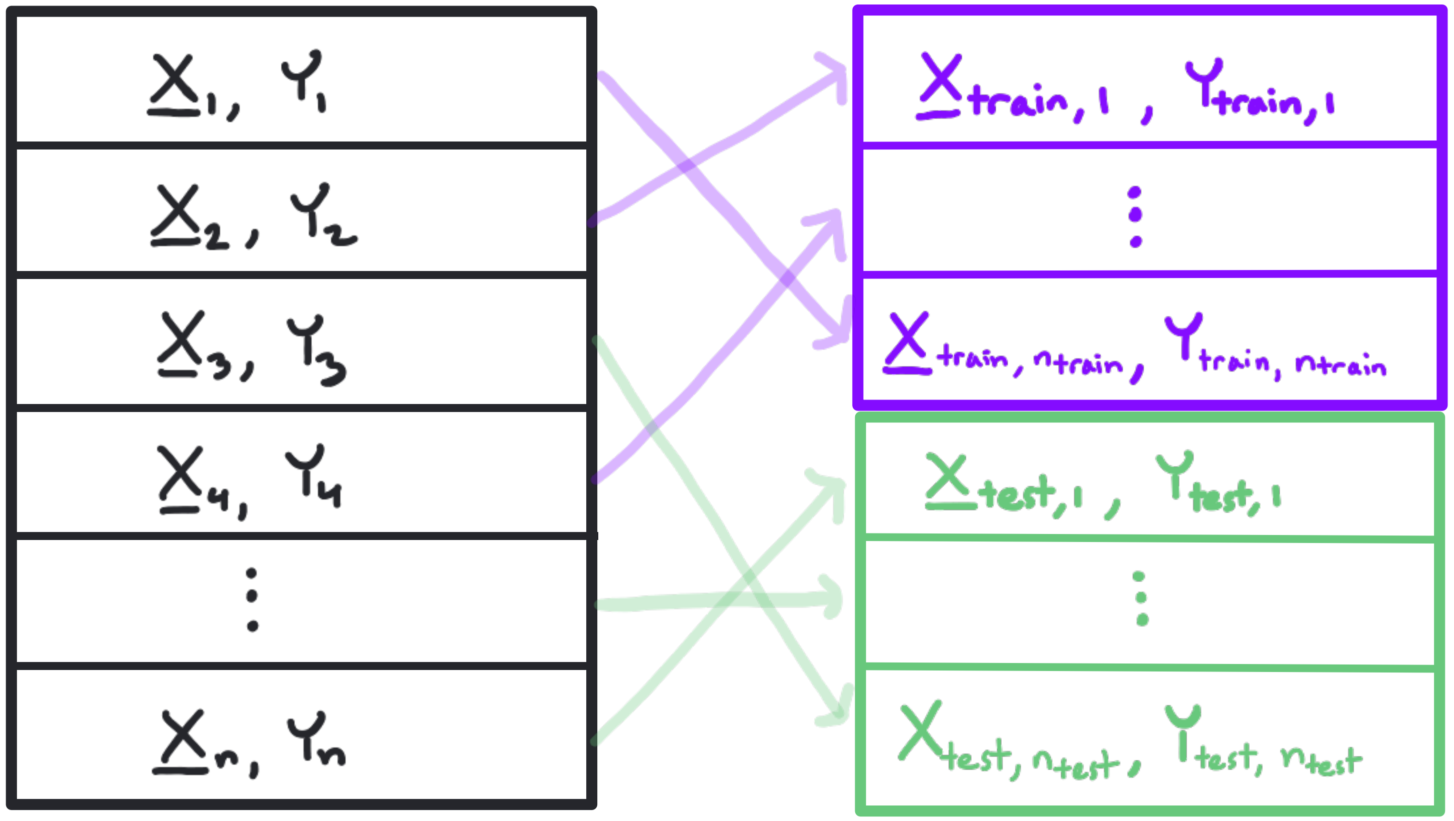
2-dimensional example dataset

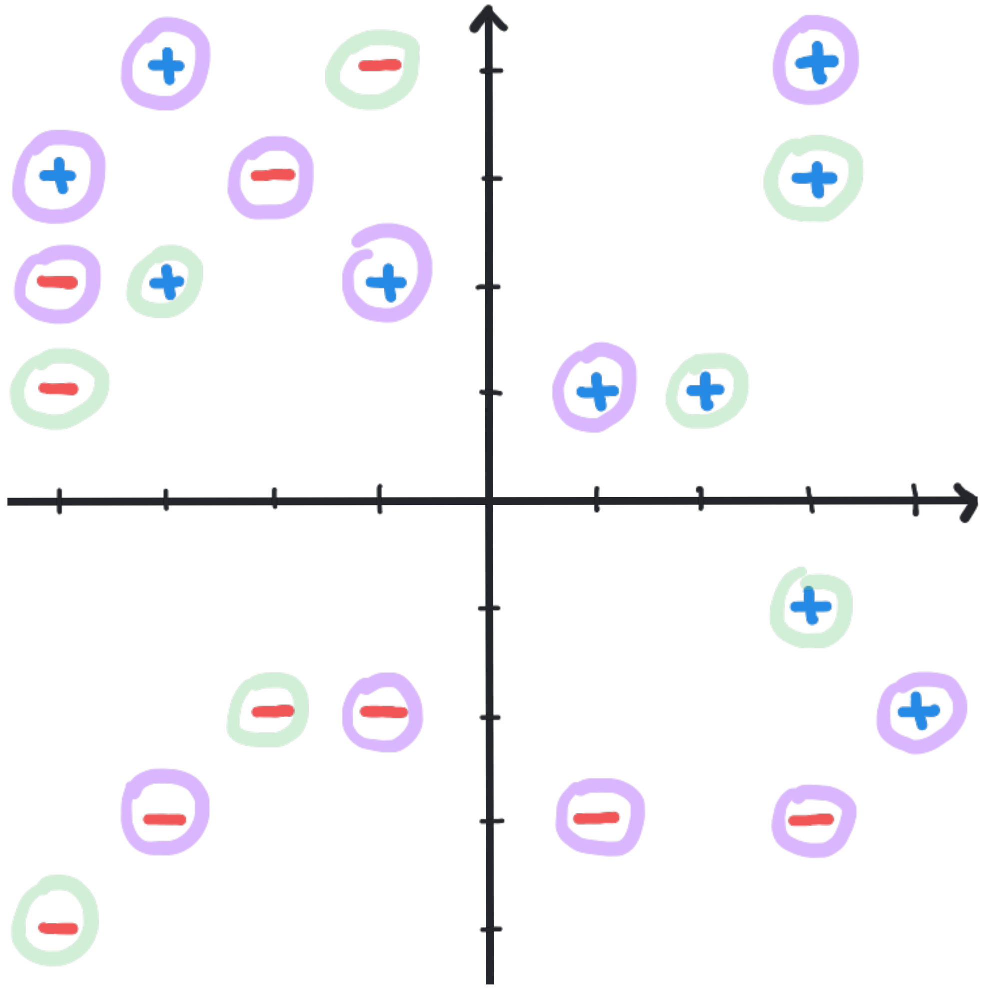
examples $n = 20$

misclassified $= 4$

Error Rate $= \frac{4}{20} = 20\%$

- Recall that we need to split our dataset into training data and test data.





→ The training data $(\underline{X}_{train,i}, Y_{train,i})$, $i = 1, \ldots, n_{train}$ is used to design a good decision rule and tune its parameters.

→ The test data $(\underline{X}_{test,i}, Y_{test,i})$, $i = 1, \ldots, n_{test}$ is used to estimate the error rate and should not be used for parameter tuning.

→ Note that $n_{train} + n_{test} = n$.

- It will be helpful to define a function to count errors:

$$g_{error}(D(\underline{x}_i), y_i) = \begin{cases} 1, & D(\underline{x}_i) \neq y_i \\ 0, & D(\underline{x}_i) = y_i \end{cases} \quad \begin{matrix} \underline{x}_i \text{ misclassified} \\ \underline{x}_i \text{ correctly classified} \end{matrix}$$

- The training error rate is the fraction of misclassified training examples:

$$Err_{train} = \frac{\#\text{ misclassified training examples}}{\#\text{ total training examples}}$$

$$= \frac{1}{n_{train}} \sum_{i=1}^{n_{train}} g_{error}(D(\underline{X}_{train,i}), Y_{train,i})$$

- The test error rate is the fraction of misclassified test examples:

$$Err_{test} = \frac{\#\text{ misclassified test examples}}{\#\text{ total test examples}}$$

$$= \frac{1}{n_{test}} \sum_{i=1}^{n_{test}} g_{error}(D(\underline{X}_{test,i}), Y_{test,i})$$

- Remember we can get to 0% training error rate by memorizing dataset but this may not lead to a low test error rate.

- <u>Nearest Neighbor Classifier:</u>

→ Basic Idea: Find the training example that is closest to the input vector, and output its label as the guess.

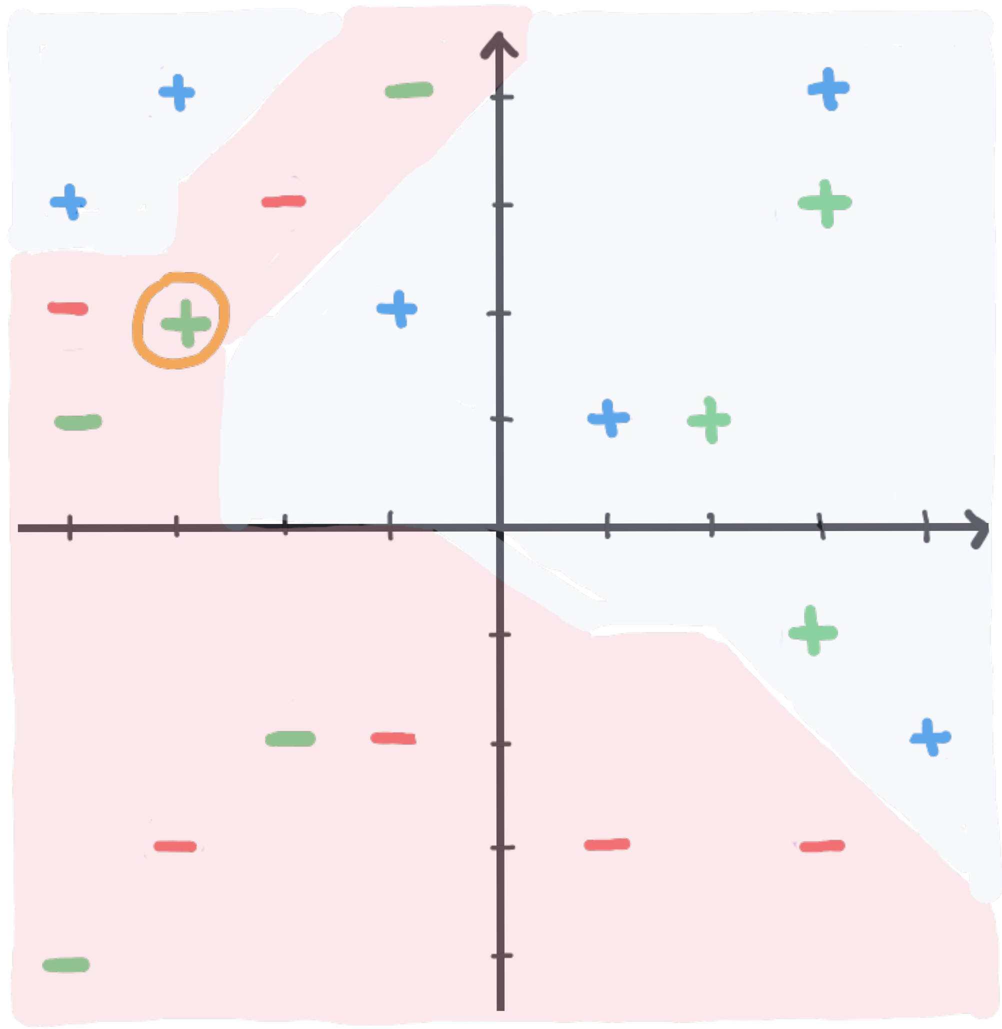
$$D_{NN}(\underline{x}) = Y_{train,\, i_{closest}}$$

where $i_{closest}$ is the index of the closest training example

$$\| \underline{x} - X_{train,\, i_{closest}} \| = \min_{i=1,\dots,n_{train}} \| \underline{x} - X_{train,\, i} \|$$

$$\| \underline{x} \| = \sqrt{\sum_{j=1}^{d} x_j^2} \text{ is the length of } \underline{x}$$

→ We can also consider a k-Nearest Neighbor Classifier that finds the k training examples closest to the input vector and outputs the majority vote amongst their labels as its guess.



$n_{train} = 12$

\# Training Errors = 0

Training Error Rate = 0%

$n_{test} = 8$

\# Test Errors = 1

Test Error Rate = $\frac{1}{8}$ = 12.5%

- **Closest Average Classifier:**

→ Basic Idea: Compute the conditional sample mean vector of the training data given $Y=+1$ and $Y=-1$. Find the closest mean vector to the input vector and output its label.

$n_{train,+}$ = # training examples with $Y_i = +1$

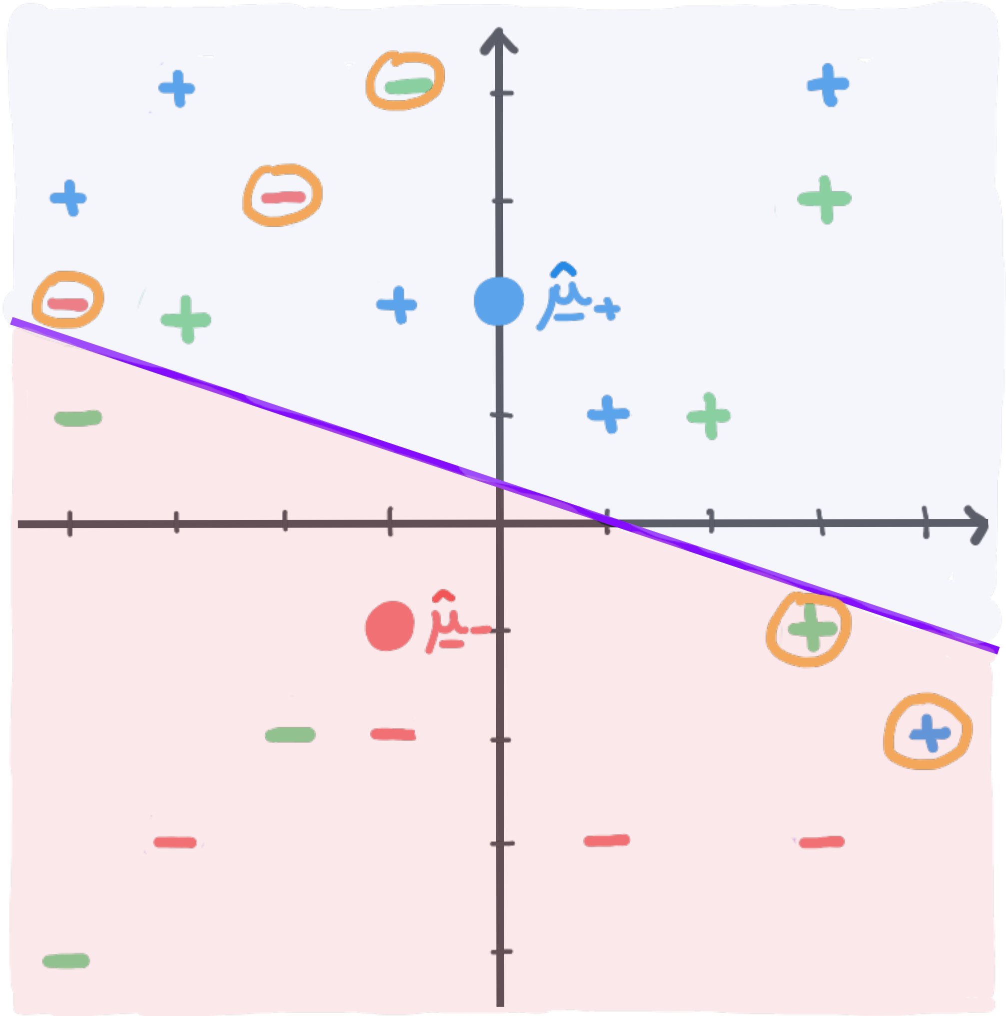
$n_{train,-}$ = # training examples with $Y_i = -1$

$$L_+ = \{ i \in \{1, \dots, n_{train}\} : Y_{train,i} = +1 \}$$

$$L_- = \{ i \in \{1, \dots, n_{train}\} : Y_{train,i} = -1 \}$$

$$\hat{\underline{\mu}}_+ = \frac{1}{n_{train,+}} \sum_{i \in L_+} \underline{X}_{train,i} \qquad \hat{\underline{\mu}}_- = \frac{1}{n_{train,-}} \sum_{i \in L_-} \underline{X}_{train,i}$$

$$D_{avg}(\underline{x}) = \begin{cases} +1 & \|\underline{x} - \hat{\underline{\mu}}_+\| \le \|\underline{x} - \hat{\underline{\mu}}_-\| \\ -1 & \|\underline{x} - \hat{\underline{\mu}}_-\| > \|\underline{x} - \hat{\underline{\mu}}_-\| \end{cases}$$



$n_{train} = 12$

Training Errors = 3

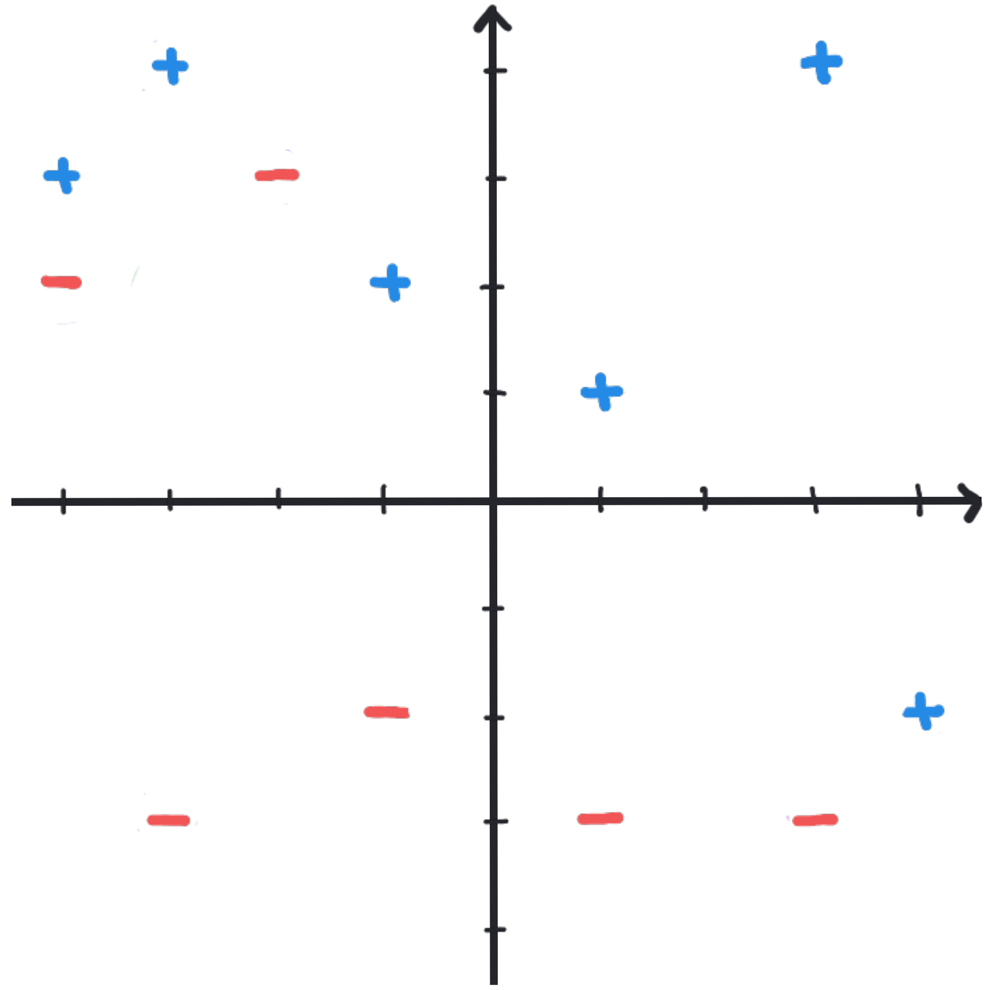
Training Error Rate $= \frac{3}{12} = 25\%$

$n_{test} = 8$

Test Errors = 2

Test Error Rate $= \frac{2}{8} = 25\%$

- **Linear Discriminant Analysis:**



→ Basic Idea: Assume that, given the label, the input vector is Gaussian with mean vector μ_+ or μ_- and the same covariance matrix Σ. Estimate these parameters from the data and apply the ML rule.

→ Define $n_{train,+}$, $n_{train,-}$, L_+, L_-, $\hat{\mu}_+$, $\hat{\mu}_-$ as on previous slide.

$$\hat{\Sigma}_+ = \frac{1}{n_{train,+} - 1} \sum_{i \in L_+} (\underline{x}_{train,i} - \hat{\underline{\mu}}_+)(\underline{x}_{train,i} - \hat{\underline{\mu}}_+)^T$$

$$\hat{\Sigma}_- = \frac{1}{n_{train,-} - 1} \sum_{i \in L_-} (\underline{x}_{train,i} - \hat{\underline{\mu}}_-)(\underline{x}_{train,i} - \hat{\underline{\mu}}_-)^T$$

$$\hat{\Sigma} = \frac{1}{n_{train} - 2} \left((n_{train,+} - 1)\, \hat{\Sigma}_+ + (n_{train,-} - 1)\, \hat{\Sigma}_- \right)$$

Conditional PDFs of $\underline{X}$ given $Y = +1$ and -1

$$\hat{f}_{\underline{X}|Y}(\underline{x} \mid y = +1) = \frac{1}{\sqrt{(2\pi)^d \det(\hat{\Sigma})}} \exp\left(-\frac{1}{2} (\underline{x} - \hat{\underline{\mu}}_+)^T \hat{\Sigma}^{-1} (\underline{x} - \hat{\underline{\mu}}_+) \right)$$

$$\hat{f}_{\underline{X}|Y}(\underline{x} \mid y = -1) = \frac{1}{\sqrt{(2\pi)^d \det(\hat{\Sigma})}} \exp\left(-\frac{1}{2} (\underline{x} - \hat{\underline{\mu}}_-)^T \hat{\Sigma}^{-1} (\underline{x} - \hat{\underline{\mu}}_-) \right)$$

- **Linear Discriminant Analysis:**

$$\hat{f}_{\underline{x}|Y}(\underline{x}\,|\,y=+1) = \frac{1}{\sqrt{(2\pi)^d \det(\hat{\Sigma})}} \exp\left(-\frac{1}{2}(\underline{x}-\hat{\underline{\mu}}_+)^T \hat{\Sigma}^{-1}(\underline{x}-\hat{\underline{\mu}}_+)\right)$$

$$\hat{f}_{\underline{x}|Y}(\underline{x}\,|\,y=-1) = \frac{1}{\sqrt{(2\pi)^d \det(\hat{\Sigma})}} \exp\left(-\frac{1}{2}(\underline{x}-\hat{\underline{\mu}}_-)^T \hat{\Sigma}^{-1}(\underline{x}-\hat{\underline{\mu}}_-)\right)$$

ML Rule: Guess $+1$ if $\hat{f}_{\underline{x}|Y}(\underline{x}\,|\,y=+1) \geq \hat{f}_{\underline{x}|Y}(\underline{x}\,|\,y=-1)$ or, equivalently,

$$(\underline{x}-\hat{\underline{\mu}}_+)^T \hat{\Sigma}^{-1}(\underline{x}-\hat{\underline{\mu}}_+) \leq (\underline{x}-\hat{\underline{\mu}}_-)^T \hat{\Sigma}^{-1}(\underline{x}-\hat{\underline{\mu}}_-)$$

$$\cancel{\underline{x}^T \hat{\Sigma}^{-1}\underline{x}} - 2\hat{\underline{\mu}}_+^T \hat{\Sigma}^{-1}\underline{x} + \hat{\underline{\mu}}_+^T \hat{\Sigma}^{-1}\hat{\underline{\mu}}_+ \leq \cancel{\underline{x}^T \hat{\Sigma}^{-1}\underline{x}} - 2\hat{\underline{\mu}}_-^T \hat{\Sigma}^{-1}\underline{x} + \hat{\underline{\mu}}_-^T \hat{\Sigma}^{-1}\hat{\underline{\mu}}_-$$

$$\hat{\underline{\mu}}_+^T \Sigma^{-1}\hat{\underline{\mu}}_+ - \hat{\underline{\mu}}_-^T \hat{\Sigma}^{-1}\hat{\underline{\mu}}_- \leq 2(\hat{\underline{\mu}}_+ - \hat{\underline{\mu}}_-)^T \hat{\Sigma}^{-1}\underline{x}$$

$$c \leq \underline{b}^T \underline{x}$$ **Linear Classifier!**

$$D_{LDA}(\underline{x}) = \begin{cases} +1 & 2(\hat{\underline{\mu}}_+ - \hat{\underline{\mu}}_-)^T \hat{\Sigma}^{-1}\underline{x} \geq \hat{\underline{\mu}}_+^T \hat{\Sigma}^{-1}\hat{\underline{\mu}}_+ - \hat{\underline{\mu}}_- \hat{\Sigma}^{-1}\hat{\underline{\mu}}_- \\ -1 & \text{otherwise} \end{cases}$$

→ Closest average classifier can be viewed as a special case with the assumption $\Sigma = I$, the identity matrix.

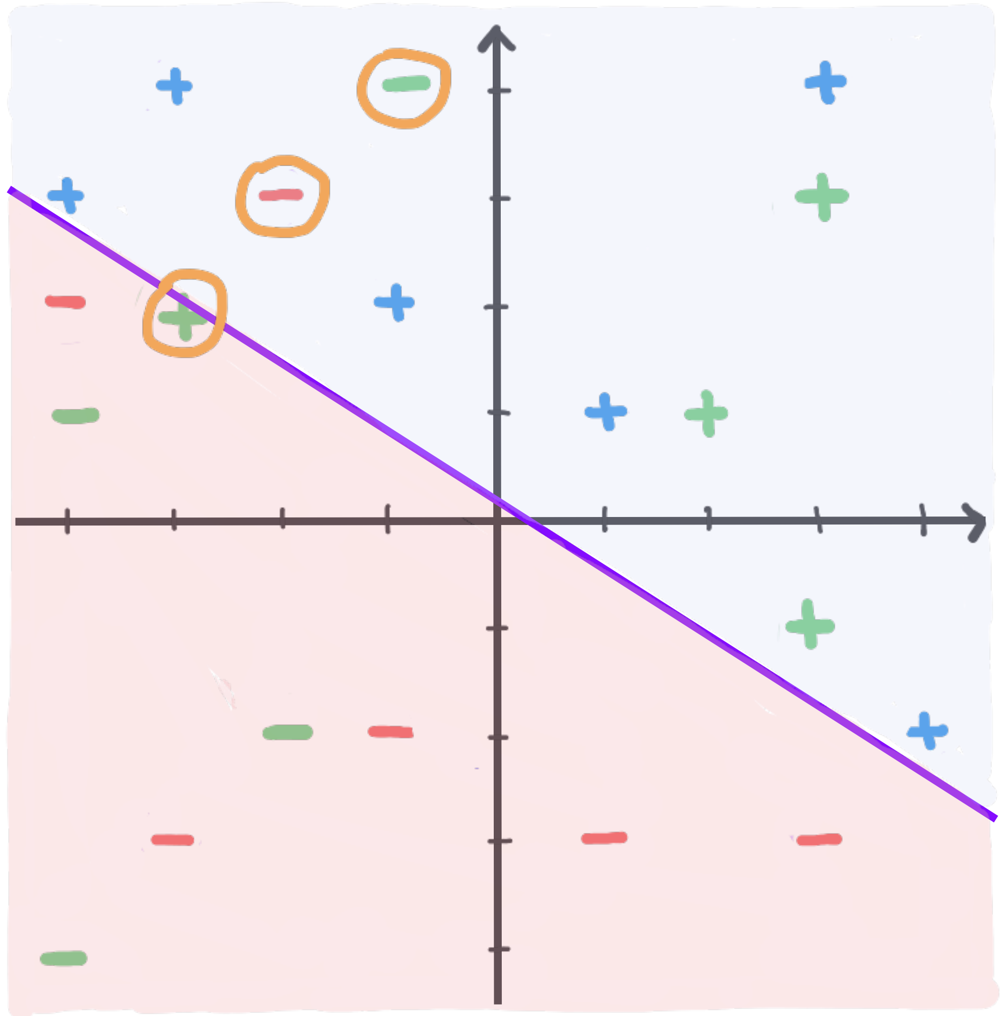
- **Linear Discriminant Analysis:**

→ Basic Idea: Assume that, given the label, the input vector is Gaussian with mean vector $\underline{\mu}_+$ or $\underline{\mu}_-$ and the same covariance matrix Σ. Estimate these parameters from the data and apply the ML rule.

$$D_{LDA}(\underline{x}) = \begin{cases} +1 & 2(\hat{\underline{\mu}}_+ - \hat{\underline{\mu}}_-)^T \hat{\Sigma}^{-1} \underline{x} \geq \hat{\underline{\mu}}_+^T \hat{\Sigma}^{-1} \hat{\underline{\mu}}_+ - \hat{\underline{\mu}}_- \hat{\Sigma}^{-1} \hat{\underline{\mu}}_- \\ -1 & \text{otherwise} \end{cases}$$

→ This is a linear classifier.

→ We could also assume the covariance matrices are different. This is called Quadratic Discriminant Analysis.



$n_{train} = 12$

\# Training Errors = 1

Training Error Rate $= \frac{1}{12} = 8.3\%$

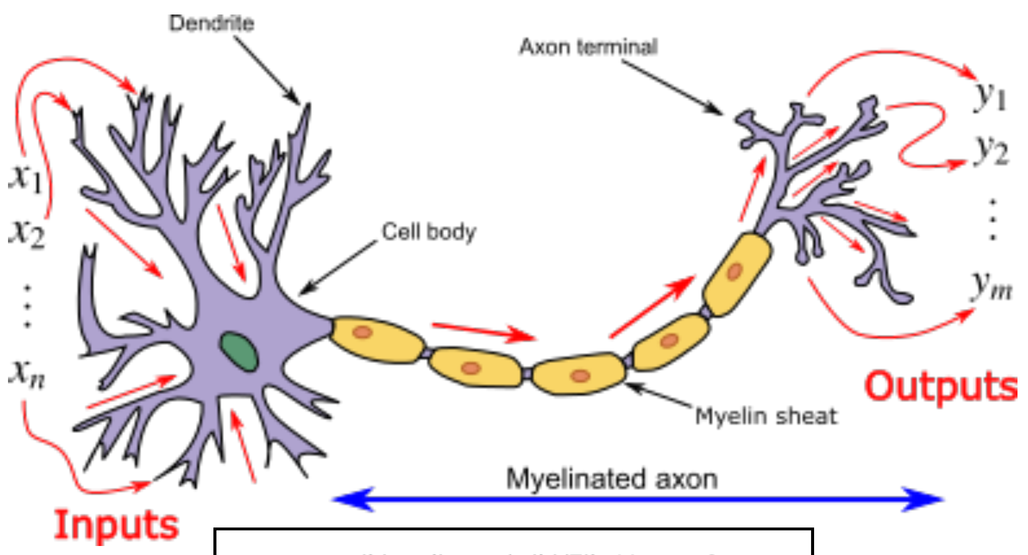
$n_{test} = 8$

\# Test Errors = 2

Test Error Rate $= \frac{2}{8} = 25\%$

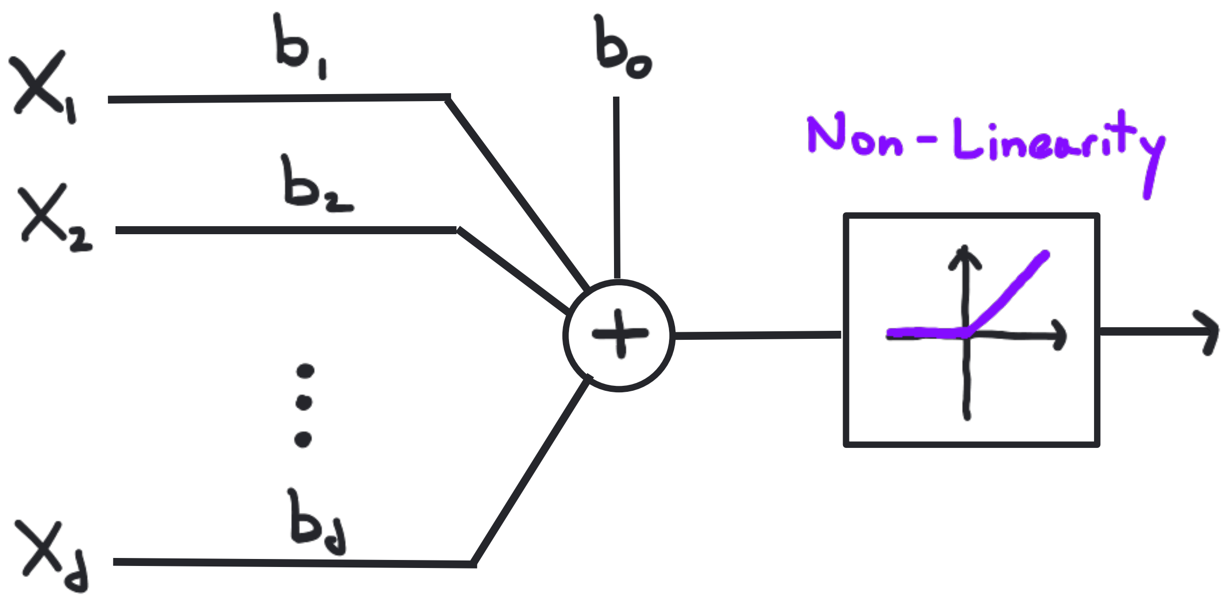
- Modern approaches to binary classification rely heavily on large-scale optimization techniques to carefully select the parameters of decision rules drawn from much richer families of functions.

- For instance, neural networks use gradient descent to set the weights of the individual units, which are loosely based on biological neurons.

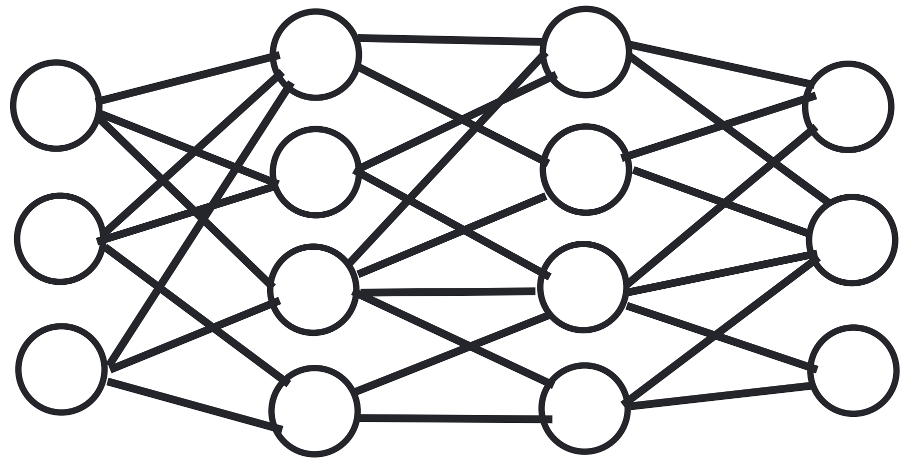


commons.wikimedia.org/wiki/File:Neuron3.png

Author: Prof. Loc Vu-Quoc

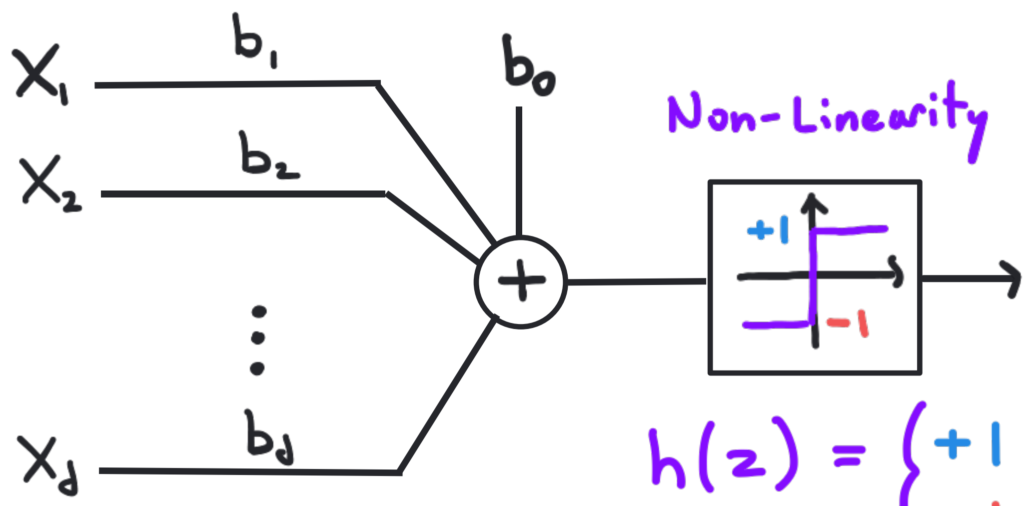


- Deep learning uses many layers of these units to tackle massive datasets. (Beyond our scope.)

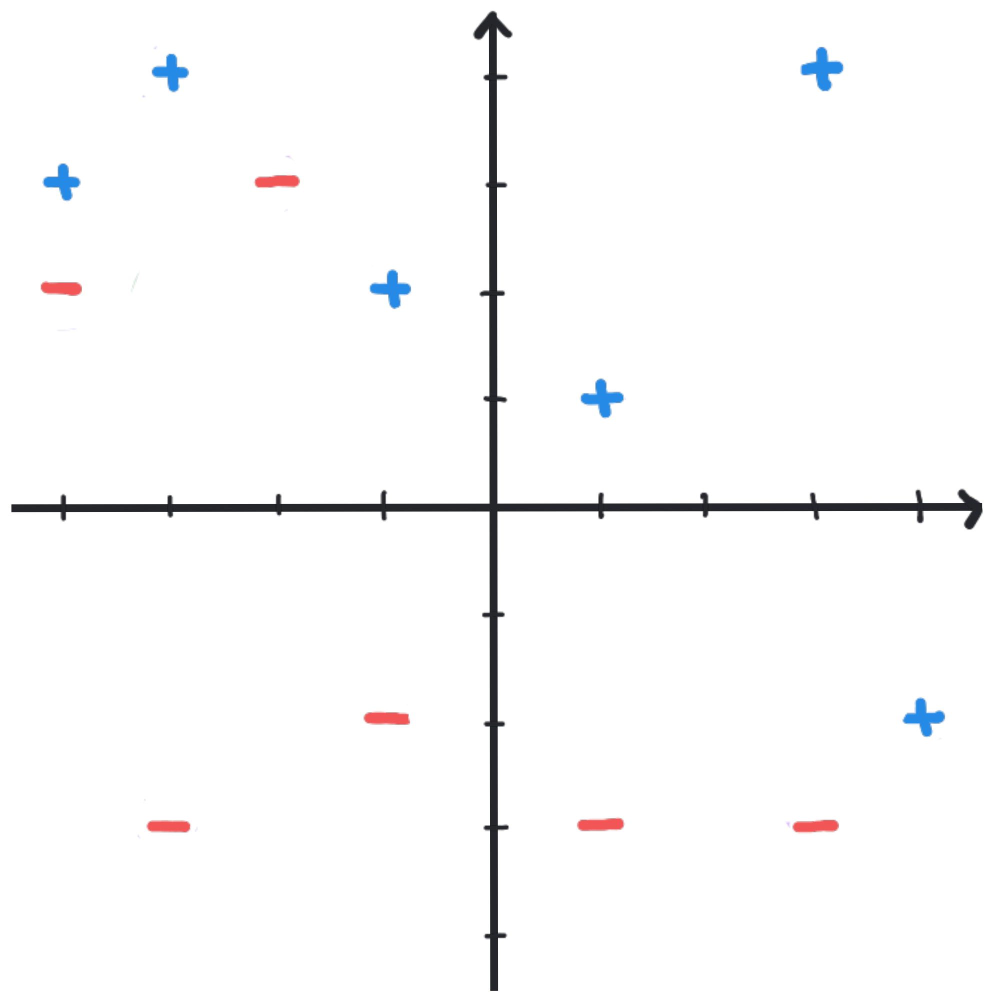


- <u>Perceptron Classifier</u>:

→ Basic Idea: Although gradient descent is beyond our scope, we can implement a simple variant of the perceptron, the first artificial neural network.



$$h(z) = \begin{cases} +1 & z \geq 0 \\ -1 & z < 0 \end{cases}$$



→ Train Weights: $\underline{b} = \begin{bmatrix} b_0 \\ \vdots \\ b_d \end{bmatrix}$, initialize $\underline{b} = \underline{0}$

for $i = 1$ to n_{train}

$Y_{temp} = h\left(\underline{b}^T \begin{bmatrix} 1 \\ \underline{X}_{train,i} \end{bmatrix}\right)$ Guess label. Note that $\underline{b}^T \begin{bmatrix} 1 \\ \underline{X}_{train,i} \end{bmatrix} = b_0 + \sum_{j=1}^{d} b_j X_{train,i,j}$

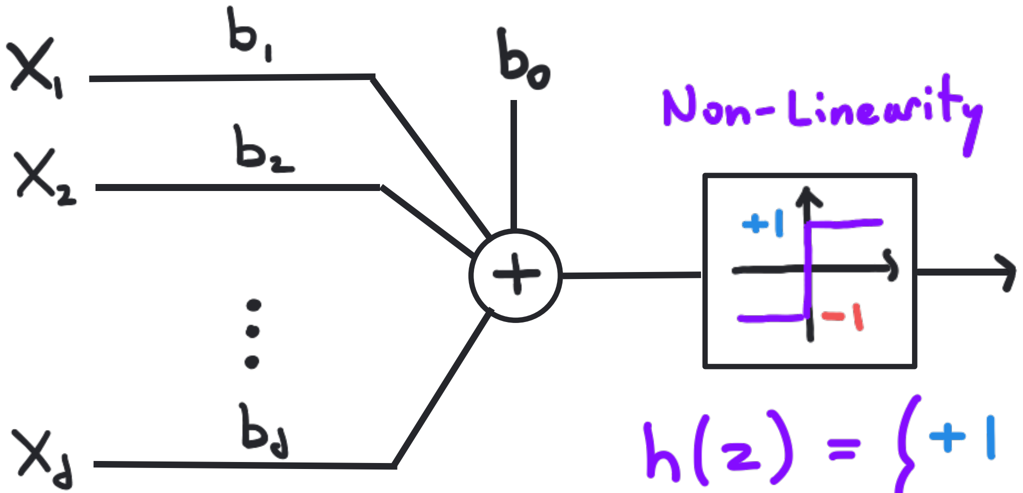
$\underline{b} = \underline{b} + r\left(Y_{train,i} - Y_{temp}\right) \begin{bmatrix} 1 \\ \underline{X}_{train,i} \end{bmatrix}$ Update weights if guess is wrong. $0 < r < 1$ is the learning rate.

end

→ Use trained weights as a linear classifier.

- <u>Perceptron Classifier</u>:

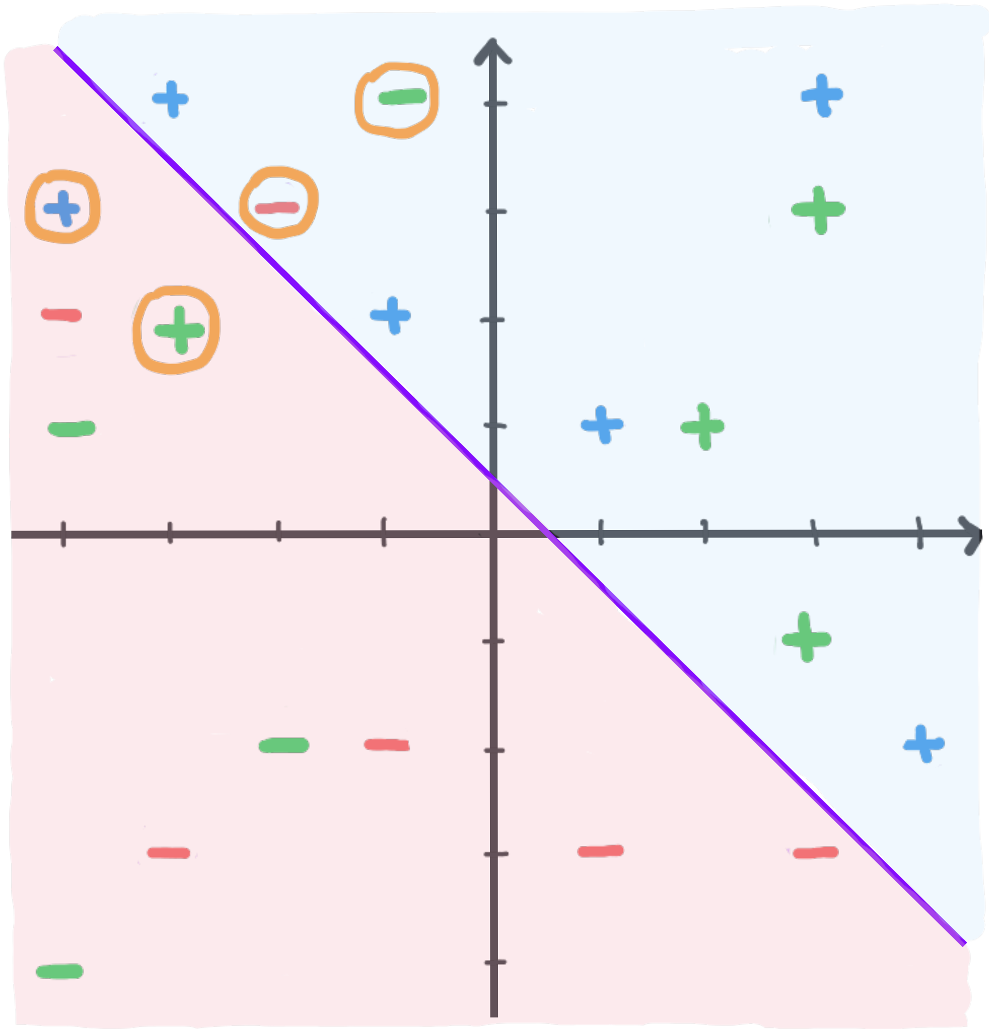
→ Basic Idea: Although gradient descent is beyond our scope, we can implement a simple variant of the perceptron, the first artificial neural network.



$$h(z) = \begin{cases} +1 & z \geq 0 \\ -1 & z < 0 \end{cases}$$

$$D_{perceptron}(\underline{x}) = \begin{cases} +1 & \underline{b}^T \begin{bmatrix} 1 \\ \underline{x} \end{bmatrix} \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

where $\underline{b} = \begin{bmatrix} b_0 \\ \vdots \\ b_d \end{bmatrix}$ are the trained weights.



$n_{train} = 12$

Training Errors = 2

Training Error Rate $= \frac{2}{12}$
$= 16.7\%$

$n_{test} = 8$

Test Errors = 2

Test Error Rate $= \frac{2}{8} = 25\%$