COGS 118A, Winter 2020

Supervised Machine Learning Algorithms

Lecture 07: Error Metrics and Perceptron

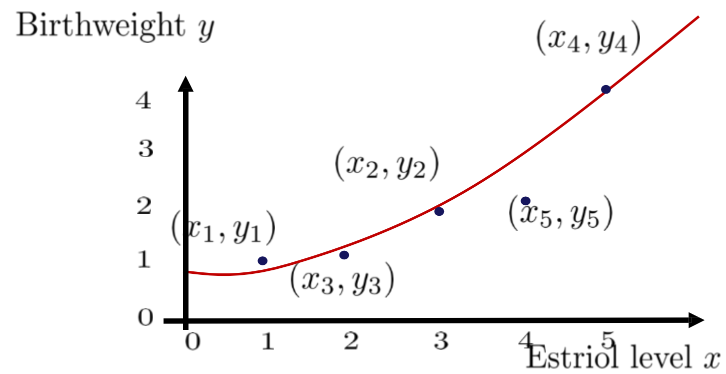
Zhuowen Tu

Put Data Into Matrix Form

$$S_{training} = \{(x_i, y_i), i = 1..n\} = \{(1,1), (3,1.9), (2,1.05), (5,4.1), (4,2.1)\}$$

Basic Equations **Matrix Form**

$$y = w_0 + w_1 x + w_2 x^2 \qquad Y = XW$$



$$1 = w_0 + w_1 \times 1 + w_2 \times 1$$
$$1.9 = w_0 + w_1 \times 3 + w_2 \times 9$$
$$1.05 = w_0 + w_1 \times 2 + w_2 \times 4$$
$$4.1 = w_0 + w_1 \times 5 + w_2 \times 25$$
$$2.1 = w_0 + w_1 \times 4 + +w_2 \times 16$$

In python:

W* = np.dot(numpy.linalg.inv(np.dot(np.transpose(X), X)),np.dot(np.transpose(X),Y))

$$\underset{Y}{\begin{pmatrix} 1 \\ 1.9 \\ 1.05 \\ 4.1 \\ 2.1 \end{pmatrix}} = \underset{X}{\begin{pmatrix} 1,1,1 \\ 1,3,9 \\ 1,2,4 \\ 1,5,25 \\ 1,4,16 \end{pmatrix}} \underset{W}{\begin{pmatrix} w_0 \\ w_1 \\ w_2 \end{pmatrix}}$$

$$W^* = (X^T X)^{-1} X^T Y = \begin{pmatrix} 1.48 \\ -0.67 \\ 0.2321 \end{pmatrix}$$

Why not to use the classification "error" all the time.

Classification error: $e = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}(y_i \neq f(\mathbf{x}_i; W))$

When your training dataset is balanced, e.g. the same number of positives and negatives, then the classification error seems to be a sound metric.

In practice, the given datasets are often unbalanced.

$$S = \{(\mathbf{x}_i, y_i), i = 1..n\}$$

Often: $\underbrace{\boxed{\sum_{i=1}^{n} \mathbf{1}(y_i = 0))}}_{\text{\# of negatives}} \overset{\text{much greater}}{>>} \underbrace{\boxed{\sum_{i=1}^{n} \mathbf{1}(y_i = 1))}}_{\text{\# of positives}}$

For example, if we have **1,000** negative samples and 1 positive sample, we can blindly classify (a trivial solution) every input sample as negative:

Classification error: $e = \frac{1}{1{,}001}$ Seeming low, but misleading!

Error measures and metrics

	classify+	classify-
true label+	true label+ and classify+	true label+ and classify-
true label-	true label- and classify+	true label- and classify-

larger preferred

True positive rate: P(classify + | true label +)

$$= sensitivity = recall$$

smaller preferred

False positive rate: P(classify+ | true label -)

larger preferred

True negative rate: P(classify - | true label-)

$$= specificity$$

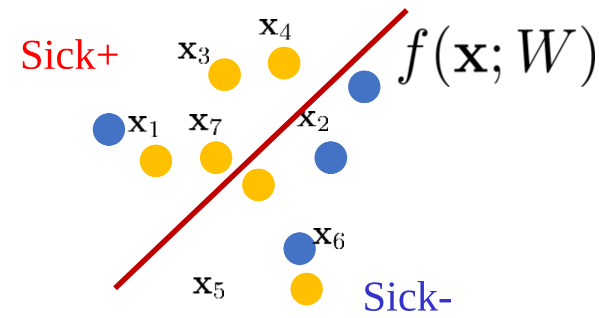
smaller preferred

False negative rate: P(classify- | true label +)

How to compute the errors

$\mathbf{x} = (x_1, ..., x_m), x_i \in \mathbb{R}, \quad \mathbf{x} \in \mathbb{R}^m \quad y \in \{-1, +1\} \quad y = -1\text{: sick-} \quad y = +1\text{: sick+}$

Given: $S_{training} = \{(\mathbf{x}_i, y_i), i = 1..100\}$



Classification (Classify): $f(\mathbf{x}; W) \in \{-1, +1\}$

$\mathbf{x}$ **(features)**	y (sick- or sick+?)	$f(\mathbf{x}; W)$ (classify- or classify +?)
$\mathbf{x}_1$	-1	-1
$\mathbf{x}_2$	-1	$+1$
$\mathbf{x}_3$	$+1$	$+1$
$\mathbf{x}_4$	-1	-1
•	•	•
$\mathbf{x}_{100}$	$+1$	-1

Summary
(confusion matrix)

$f(\mathbf{x}; W)$ \ y	**sick +**	**sick -**	**Total**
Classify +	30	10	40
Classify -	10	50	60
Total	40	60	100

Error measures

Summary

(confusion matrix)

$f(\mathbf{x};W)$ \ y	**sick +**	**sick -**	**Total**
classify +	30	10	40
classify -	10	50	60
Total	40	60	100

y: ground truth labels

$f(\mathbf{x};W)$: prediction

- Having faithful evaluation measures is critical in the success of machine learning.
- Computing the "error" is not unique.

Error Metrics and Evaluation

		Condition (as determined by "Gold standard")			
	Total population	Condition positive	Condition negative	Prevalence = $\frac{\Sigma \text{ Condition positive}}{\Sigma \text{ Total population}}$	
Test outcome	Test outcome positive	**True positive**	**False positive** (Type I error)	Positive predictive value (PPV, Precision) = $\frac{\Sigma \text{ True positive}}{\Sigma \text{ Test outcome positive}}$	False discovery rate (FDR) = $\frac{\Sigma \text{ False positive}}{\Sigma \text{ Test outcome positive}}$
	Test outcome negative	**False negative** (Type II error)	**True negative**	False omission rate (FOR) = $\frac{\Sigma \text{ False negative}}{\Sigma \text{ Test outcome negative}}$	Negative predictive value (NPV) = $\frac{\Sigma \text{ True negative}}{\Sigma \text{ Test outcome negative}}$
	Positive likelihood ratio (**LR+**) = TPR/FPR	True positive rate (TPR, Sensitivity, Recall) = $\frac{\Sigma \text{ True positive}}{\Sigma \text{ Condition positive}}$	False positive rate (FPR, Fall-out) = $\frac{\Sigma \text{ False positive}}{\Sigma \text{ Condition negative}}$	Accuracy (ACC) = $\frac{\Sigma \text{ True positive} + \Sigma \text{ True negative}}{\Sigma \text{ Total population}}$	
	Negative likelihood ratio (**LR−**) = FNR/TNR	False negative rate (FNR) = $\frac{\Sigma \text{ False negative}}{\Sigma \text{ Condition positive}}$	True negative rate (TNR, Specificity, SPC) = $\frac{\Sigma \text{ True negative}}{\Sigma \text{ Condition negative}}$		
	Diagnostic odds ratio (**DOR**) = LR+/LR−				

Classification result is denoted as "Test outcome" here.

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

Error measures and metrics

	classify+	classify-
sick+	sick+ and classify+	sick+ and classify-
sick-	sick- and classify+	sick- and classify-

larger preferred

True positive rate: P(classify + | sick +)

$$= sensitivity = recall$$

smaller preferred

False positive rate: P(classify+ | sick -)

larger preferred

True negative rate: P(classify - | sick-)

$$= specificity$$

smaller preferred

False negative rate: P(classify- | sick +)

Intuition Test

Summary (confusion matrix)

$f(\mathbf{x};W)$ \ y	**sick +**	**sick -**	**Total**
classify +	30	10	40
classify -	10	50	60
Total	40	60	100

A. High sensitivity, low specificity

B. High sensitivity, high specificity

C. Low sensitivity, low specificity

D. Low sensitivity, high specificity

Error measures

Summary (confusion matrix)

y / $f(\mathbf{x}; W)$	**sick +**	**sick -**	**Total**
classify +	30	10	40
classify -	10	50	60
Total	40	60	100

True positive rate: P(classify+ | sick +) $= \frac{30}{40} = 0.75$

$= sensitivity = recall$

False positive rate: P(classify + | sick -) $= \frac{10}{60} = 0.167$

True negative rate: P(classify - | sick-) $= \frac{50}{60} = 0.833$

$= specificity$

False negative rate: P(classify - | sick +) $= \frac{10}{40} = 0.25$

Error measures

Note that:

True positive rate + False positive + True negative + False negative $\neq 1$

0.75 0.167 0.833 0.25

True positive rate + False negative $= 1$

0.75 0.25

P(classify + | sick +) P(classify - | sick +)

False positive + True negative $= 1$

0.167 0.833

P(classify + | sick -) P(classify - | sick-)

Why not to use the classification "error" all the time.

Classification error: $e = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}(y_i \neq f(\mathbf{x}_i; W))$

In practice, the given datasets are often unbalanced.

$$S = \{(\mathbf{x}_i, y_i), i = 1..n\}$$

For example, if we have **1,000** negative samples and 1 positive sample, we can blindly classify every input sample as negative:

$\boxed{\text{Classification error: } e = \frac{1}{1{,}001}}$

Seeming low, but misleading!

True positive rate: P(classify+ | sick +)

$$= sensitivity = recall = \frac{0}{1} = 0$$

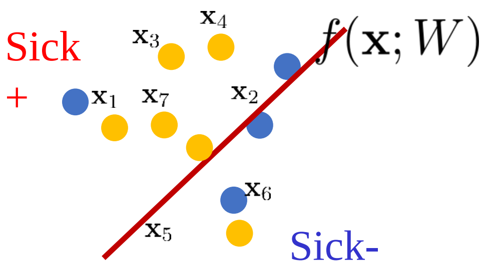
True negative rate: P(classify - | sick-)

$$= specificity = \frac{1000}{1000} = 1$$

A trivial solution leads to poor sensitivity (recall) now!

How to compute the errors

Classification (Test): $f(\mathbf{x}; W) \in \{-1, +1\}$



x (features)	y (sick- or sick+?)	$f(\mathbf{x}; W)$ (classify- or classify +?)
$\mathbf{x}_1$	-1	-1
$\mathbf{x}_2$	-1	$+1$
$\mathbf{x}_3$	$+1$	$+1$
$\mathbf{x}_4$	-1	$+1$
•	•	•
$\mathbf{x}_{100}$	$+1$	$+1$

Summary (confusion matrix)

$f(\mathbf{x}; W)$ \ y	**sick +**	**sick -**	**Total**
classify +	32	15	47
classify -	8	45	53
Total	40	60	100

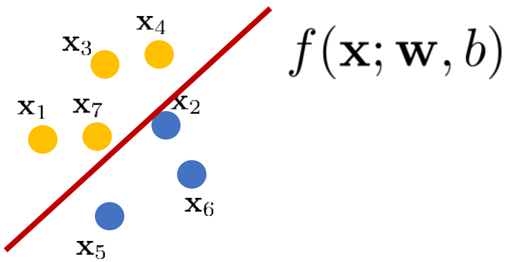
$$f(\mathbf{x}; \mathbf{w}, b) = \begin{cases} +1 & if\ \mathbf{w} \cdot \mathbf{x} + b \geq th \\ -1 & otherwise \end{cases}$$

There is often an additional threshold (*th*) that you can use to adjust the preference between sensitivity and specificity for your desired output, a high sensitivity (e.g. virus detection) vs. high specificity (earthquake alarm).

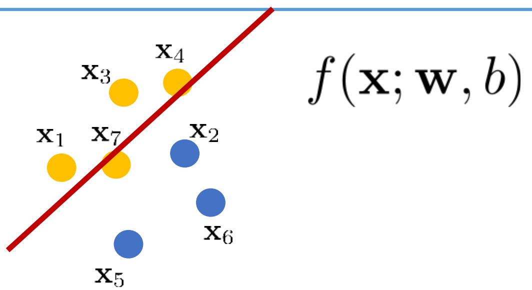
Input: $\mathbf{x} = (x_1, x_2, ...)$

Label: $y \in \{-1, +1\}$

Model parameter: $\mathbf{w}, b$



$$f(\mathbf{x}; \mathbf{w}, b) = \begin{cases} +1 & if\ \mathbf{w} \cdot \mathbf{x} + b \geq 0 \\ -1 & otherwise \end{cases}$$



$$f(\mathbf{x}; \mathbf{w}, b) = \begin{cases} +1 & if\ \mathbf{w} \cdot \mathbf{x} + b \geq th \\ -1 & otherwise \end{cases}$$

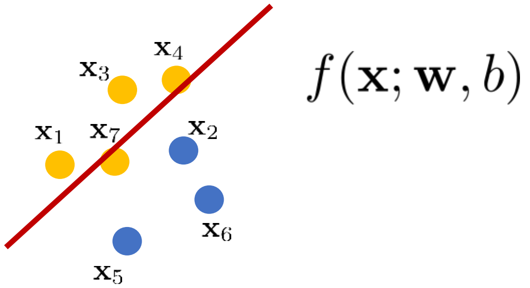
Why there is an additional threshold after training?

Why there is an additional threshold after training?

Input: $\mathbf{x} = (x_1, x_2, \ldots)$

Label: $y \in \{-1, +1\}$

Model parameter: $\mathbf{w}, b$



$$f(\mathbf{x}; \mathbf{w}, b) = \begin{cases} +1 & if\ \mathbf{w} \cdot \mathbf{x} + b \geq th \\ -1 & otherwise \end{cases}$$

1. After training, the model parameters (**w**, b) are fixed.
2. To allow the adjustment for the preference between favoring high sensitivity or high specificity, there is an additional threshold to specify after the model has been learned.

Error measures

Summary (confusion matrix)

$f(\mathbf{x}; W)$ \ y	**sick +**	**sick -**	**Total**
classify +	32	15	47
classify -	8	45	53
Total	40	60	100

True positive rate: P(classify + | sick +) $= \frac{32}{40} = 0.8$

$= sensitivity = recall$

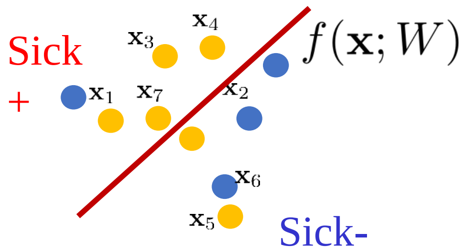
False positive rate: P(classify + | sick -) $= \frac{15}{60} = 0.25$

True negative rate: P(classify - | sick-) $= \frac{45}{60} = 0.75$

$= specificity$

False negative rate: P(classify - | sick +) $= \frac{8}{40} = 0.2$

Error measures



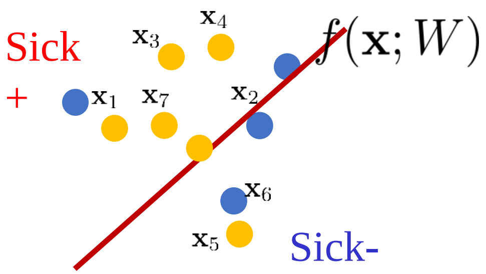
y / $f(\mathbf{x}; W)$	**sick +**	**sick -**	**Total**
classify +	30	10	40
classify -	10	50	60
Total	40	60	100

True positive rate: P(classify+ | sick +) $= \frac{30}{40} = 0.75$

$= sensitivity = recall$

True negative rate: P(classify - | sick-) $= \frac{50}{60} = 0.833$

$= specificity$



y / $f(\mathbf{x}; W)$	**sick +**	**sick -**	**Total**
classify +	32	15	47
classify -	8	45	53
Total	40	60	100

True positive rate: P(classify + | sick +) $= \frac{32}{40} = 0.8$

$= sensitivity = recall$

True negative rate: P(classify - | sick-) $= \frac{45}{60} = 0.75$

$= specificity$

Conclusion for sensitivity and specificity

True positive rate: P(classify + | true label +)

$$= sensitivity = recall$$

True negative rate: P(classify - | true label-)

$$= specificity$$

Using a single number, e.g. classification error, is sometimes insufficient to evaluate the effectiveness of a classifier, especially when the negative and positive samples are unbalanced.

In medical/bio related fields, sensitivity and specificity are often used to evaluate the performance of a classifier.

Ideally we want to have a classifier with 100% sensitivity and 100% specificity, but it is often hard to achieve in practice. We often seek a balance.

Another commonly used metric: precision and recall.

Intuition Test

For 30 data points, 20 are positive and 10 are negative. The model predicts 5 to be positive, among them 4 are accurate prediction.

A. High precision, low recall

B. High precision, high recall

C. Low precision, low recall

D. Low precision, high recall

Precision and Recall

$$\text{precision} = P(sick + | classify+)$$

$$= \frac{P(sick+ \; and \; classify+)}{P(classify+)}$$

$$\text{recall} = P(clasify + | sick+)$$

$$= \frac{P(sick+ \;\; and \;\; classify+)}{P(sick+)}$$

Why not to use the classification "error" all the time?

Classification error: $e = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}(y_i \neq f(\mathbf{x}_i; W))$

In practice, the given datasets are often unbalanced.

$$S = \{(\mathbf{x}_i, y_i), i = 1..n\}$$

For example, if we have **1,000** negative samples and 1 positive sample, we can blindly classify every input sample to be negative:

P(true label + |classify+)

$$= precision = \frac{0}{0}$$

P(classify+|true label +)

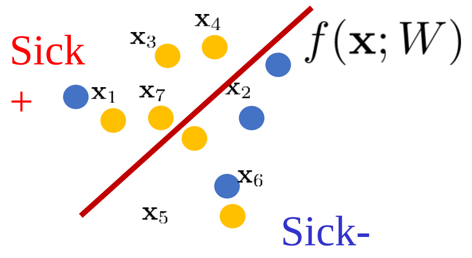
$$= recall = \frac{0}{1} = 0$$

A trivial solution leads to poor sensitivity (recall) now!

Error measures

$$\text{precision} = \frac{P(sick+ \ and \ classify+)}{P(classify+)}$$

$$\text{recall} = \frac{P(sick+ \ \ and \ \ classify+)}{P(sick+)}$$



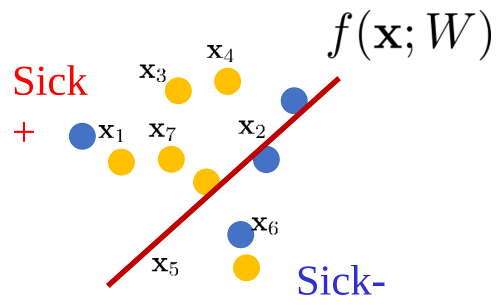
$f(\mathbf{x};W)$ \ y	**sick +**	**sick -**	**Total**
classify +	30	10	40
classify -	10	50	60
Total	40	60	100

P(sick+ | classify +) $= \frac{30}{40} = 0.75$

$= precision$

P(classify+ | sick +) $= \frac{30}{40} = 0.75$

$= recall$



$f(\mathbf{x};W)$ \ y	**sick +**	**sick -**	**Total**
classify +	32	15	47
classify -	8	45	53
Total	40	60	100

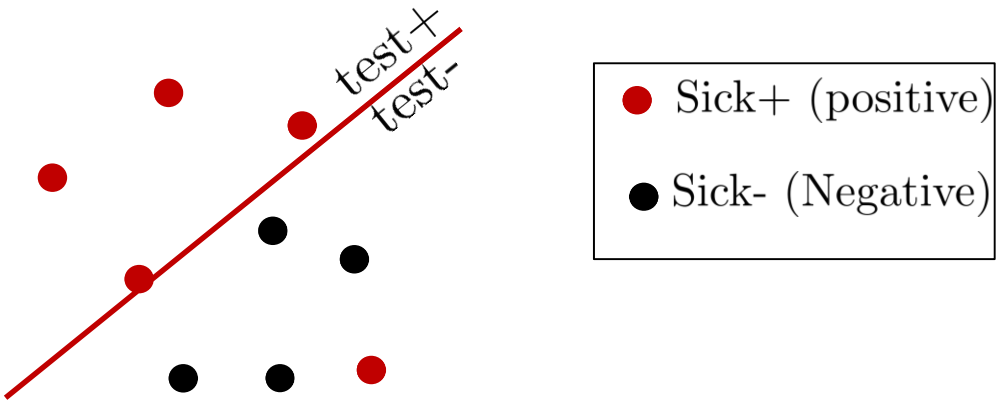
P(sick+ | classify +) $= \frac{30}{47} = 0.64$

$= precision$

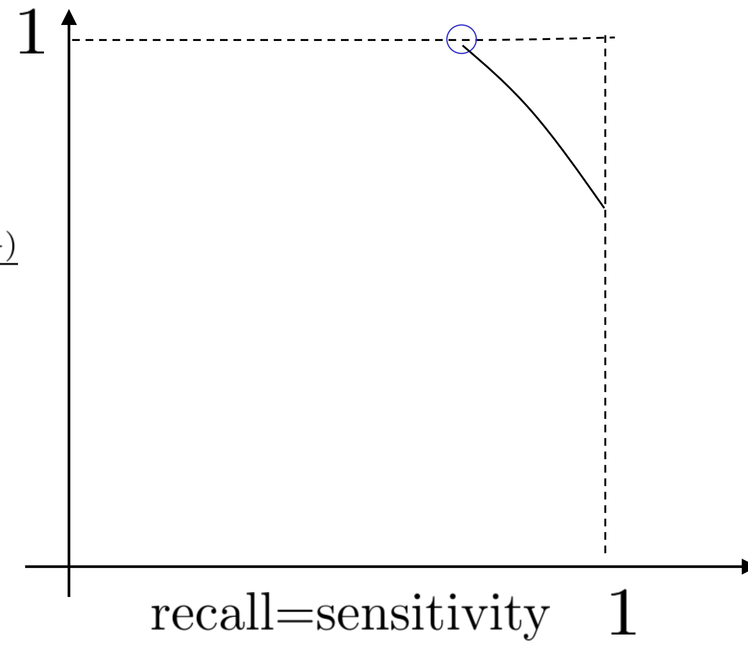
P(classify + | sick +) $= \frac{32}{40} = 0.8$

$= recall$

Precision & Recall



precision $= \frac{P(sick+ \ and \ test+)}{P(test+)}$



$$P(test+|sick+) = \frac{P(sick+ \ and \ test+)}{P(sick+)}$$

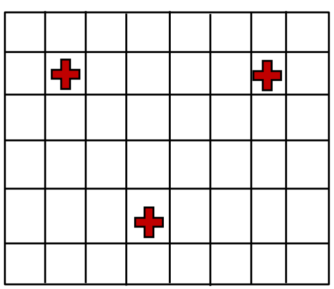
Error metric

$$\text{precision}=P(target|hit) = \frac{\#(target,hit)}{\#(hit)}$$

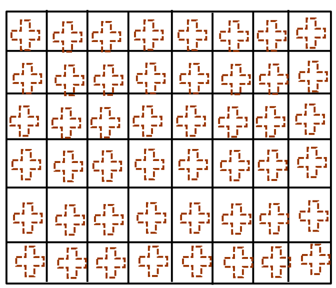
$$\text{recall}=P(hit|target) = \frac{\#(target,hit)}{\#(target)}$$

$$\text{F-value}=\frac{2\times precision\times recall}{precision+recall}$$

6×8 possible locations



3 targets to hit

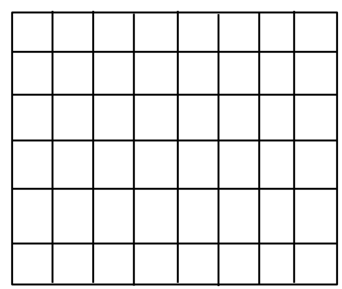


all hit

$\text{precision}=\frac{3}{48}$

$\text{recall}=\frac{3}{3}$

$\text{F-value}=\frac{0.125}{1.0625}$

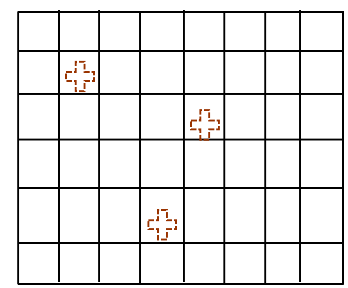


zero hit

$\text{precision}=\frac{0}{0}$

$\text{recall}=\frac{0}{3}$

$\text{F-value}=\frac{0}{1}$



miss one

$\text{precision}=\frac{2}{3}$

$\text{recall}=\frac{2}{3}$

$\text{F-value}\approx 0.667$

Conclusion for precision and recall

Precision: P(true label +| classify +)

Recall: P(classify + | true label +)

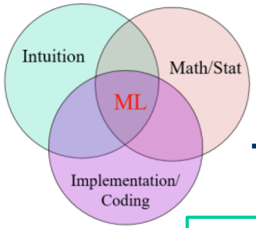
Precision and recall are two combined measures that are widely used in machine learning.

Same as specificity vs. sensitivity, we often seek a balance between precision and recall.

We can even a single number for the evaluation, the F-score:

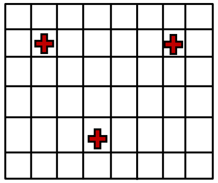
$$\text{F-value}=\frac{2\times precision\times recall}{precision+recall}$$

The f-value is a more faithful classification error metric compared to the direct error, but it is usually hard to directly optimize though.

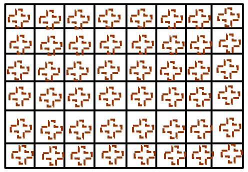


Recap: Error Metrics

Intuition: The overall effectiveness of a discriminative classifier can be evaluated using error metrics. Typically, the averaged error (or $accuracy = 1 - error$) of all the training samples is used for evaluation. However, the error can be misleading, especially for unbalanced dataset where e.g. the number of positive samples is very small w.r.t. the number of negative samples (e.g. cancerous vs. non-cancerous). Therefore, a pair of numbers are often computed precision vs. recall or specificity vs. sensitivity to give a more objective measure. A good classifier often has balanced precision and recall values although ideally we hope to have 1 for both. In machine learning, even the classifier model has been trained and fixed, there often exists a threshold that a user can customize to prefer a higher precision or a higher recall. To have a single number for judging the quality of a classifier, we use the F-score which is a combination of precision and recall within range $[0, 1]$. An extreme/trivial classifier that predicts all the samples as positives (or all as negatives) leads to $F - score = 0$ and the perfect classifier with 0 error attains $F - score = 1$.



3 targets to hit

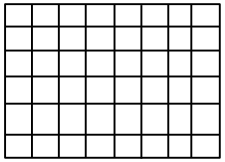


all hit

precision=$\frac{3}{48}$

recall=$\frac{3}{3}$

F-value=$\frac{0.125}{1.0625}$



zero hit

precision=$\frac{0}{0}$

recall=$\frac{0}{3}$

F-value=$\frac{0}{1}$

Math: error: $e = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}(y_i \neq f(\mathbf{x}_i; W))$ $\quad \mathbf{1}(z) = \begin{cases} 1 & if\ z = TRUE \\ 0 & otherwise \end{cases}$

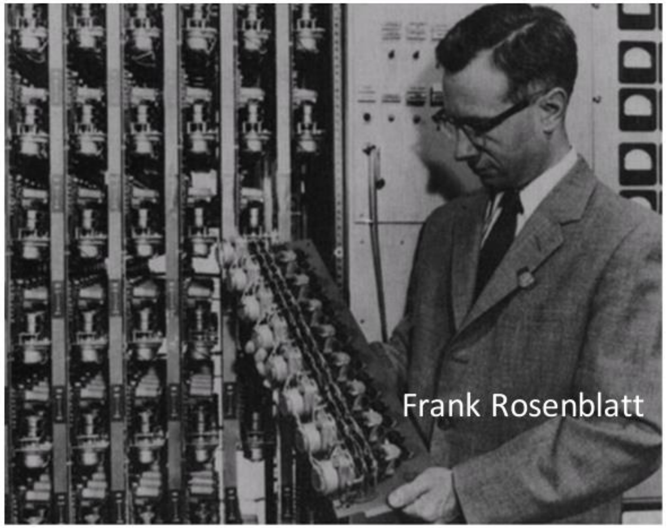
$$\text{precision} = \frac{P(sick+\ and\ test+)}{P(test+)} \qquad \text{recall} = \frac{P(sick+\ \ and\ \ test+)}{P(sick+)}$$

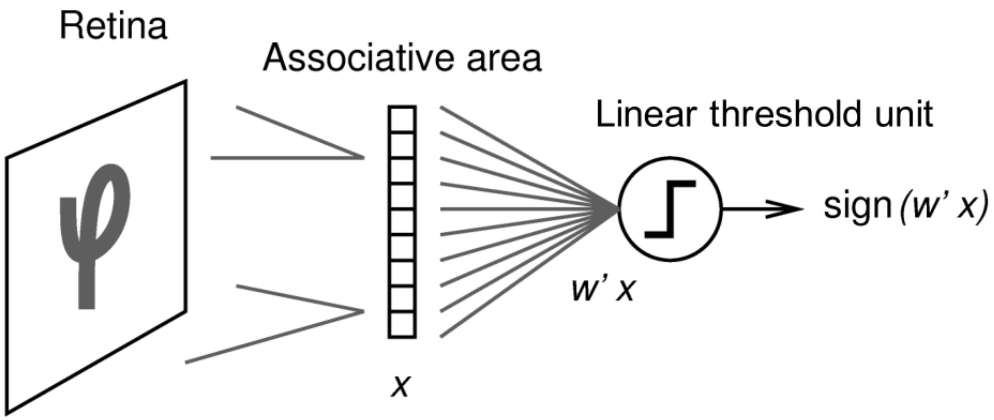
F-value=$\frac{2 \times precision \times recall}{precision + recall}$

Perceptron

Let’s look at a very simple form.

Perceptron





Supervised learning of the weights w using the Perceptron algorithm.

Slides from L. Bottou

Summary of the classification problem

$\mathbf{x} = (x_1, x_2, \ldots)$

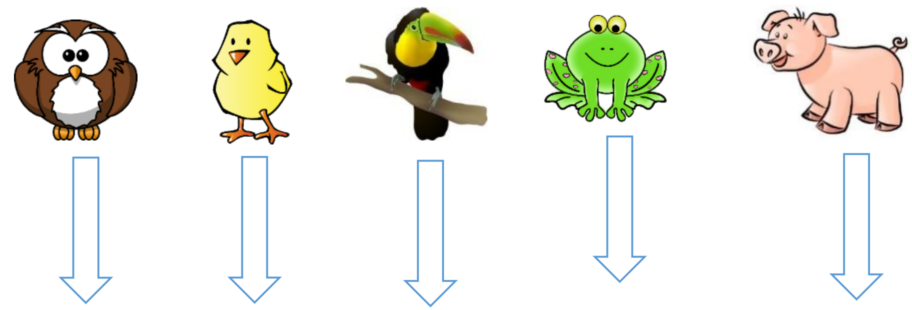
$y = 1(bird)$

x_1: color

x_2 weight

$\ldots$

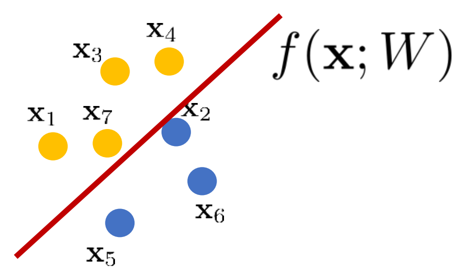
$$S_{training} =$$



$$\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), (\mathbf{x}_3, y_3), (\mathbf{x}_4, y_4), (\mathbf{x}_5, y_5)\}$$

Train classifier $f(\mathbf{x}; W)$

W: model parameter

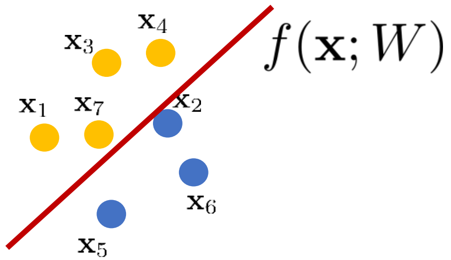


Three key variables we are dealing with

Input: $\mathbf{x} = (x_1, x_2, ...)$

Label: $y \in 0, 1$

Model parameter: W



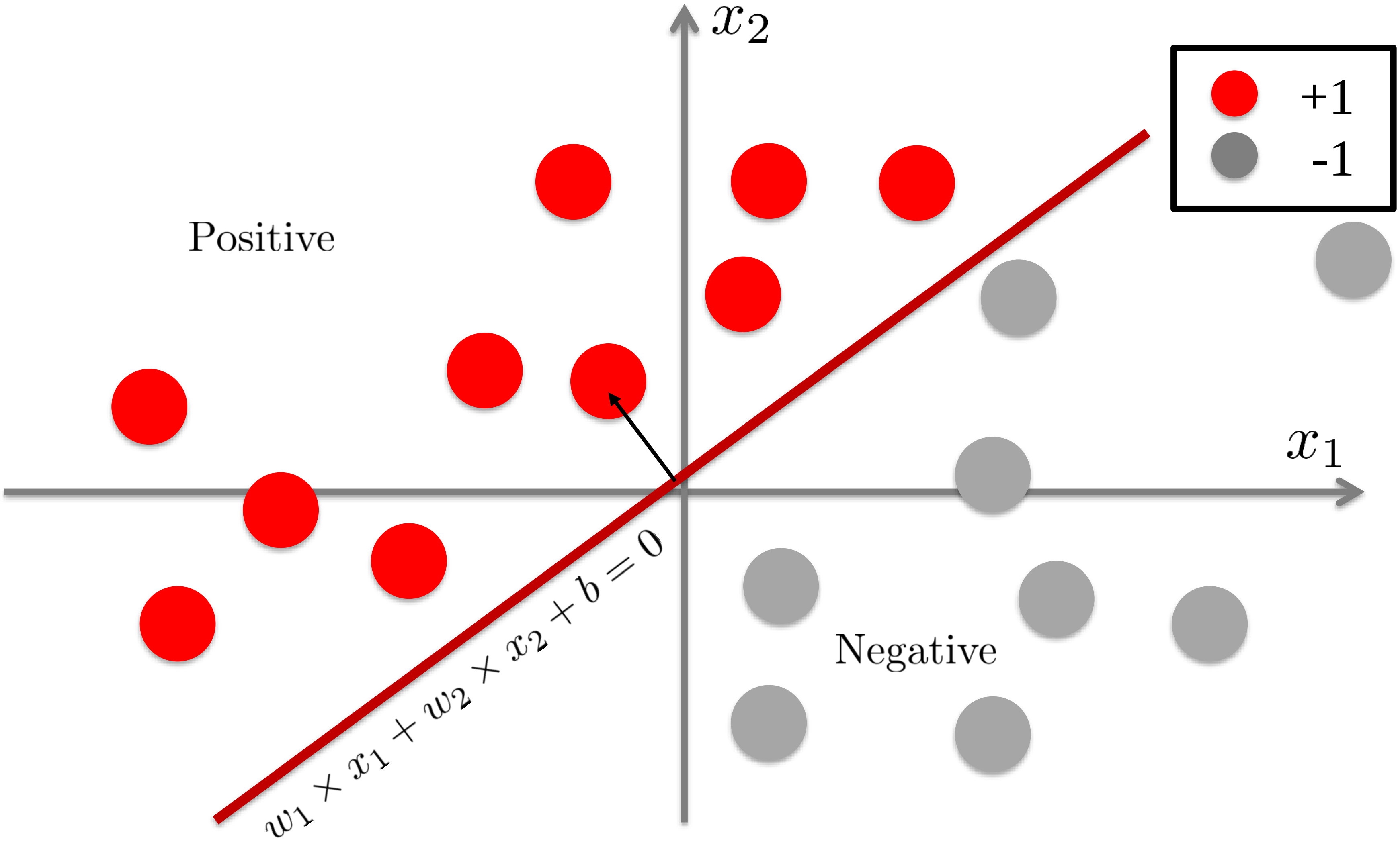
Perceptron

$$\mathbf{x} = (x_1, x_2, ...) \qquad \mathbf{w} = (w_1, w_2, ...)$$

Perceptron:

$$f(\mathbf{x}|\mathbf{w}; b) = \begin{cases} +1 & if\ \mathbf{w}^T\mathbf{x} + b \geq 0 \\ -1 & otherwise \end{cases}$$

Perceptron classifier: $f(x_1, x_2; w_1, w_2, b) = sign(w_1 \times x_1 + w_2 \times x_2 + b)$



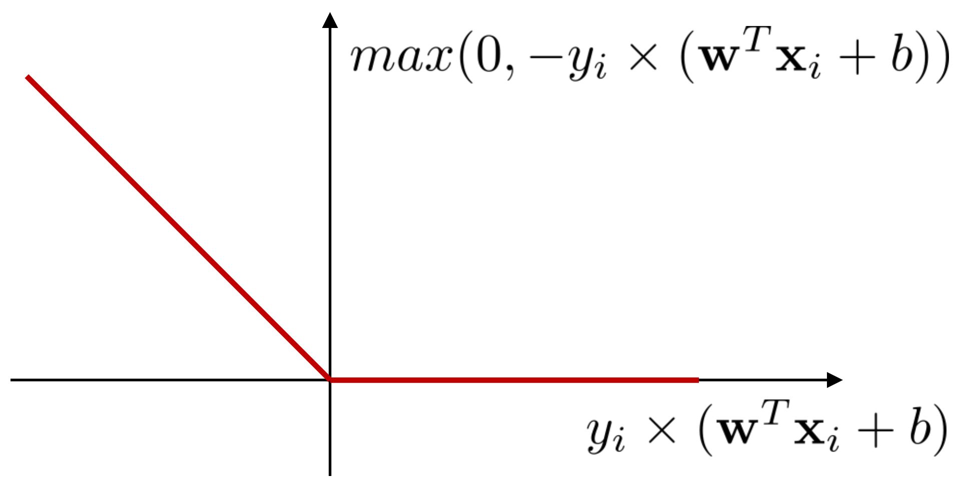
Perceptron training is a special gradient descent algorithm

Perceptron: $f(\mathbf{x};\mathbf{w}) = sign(\mathbf{w}^T\mathbf{x} + b)$

$S = \{(\mathbf{x}_i, y_i), i = 1..n\}$ $\quad y_i \in -1, +1$

no penalty if y_i and $(\mathbf{w}^T\mathbf{x}_i + b)$ have the same sign

Training: Minimize $\mathcal{L}(\mathbf{w}, b) = \sum_i max(0, -y_i \times (\mathbf{w}^T\mathbf{x}_i + b))$



$$\frac{\mathcal{L}(\mathbf{w},\mathbf{b})}{\partial \mathbf{w}} = \begin{cases} 0 & if\ y_i = sign(\mathbf{w}^T\mathbf{x}_i + b) \\ -y_i\mathbf{x}_i & otherwise \end{cases}$$

$$\frac{1}{2}(y_i - sign(\mathbf{w}^T\mathbf{x} + b)) \qquad = \frac{1}{2}(target_i - output_i)$$

Perceptron training is a special gradient descent algorithm

Perceptron: $f(\mathbf{x};\mathbf{w}) = sign(\mathbf{w}^T\mathbf{x} + b)$

$S = \{(\mathbf{x}_i, y_i), i = 1..n\}$

no penalty if y_i and $(\mathbf{w}^T\mathbf{x}_i + b)$ have the same sign

Training: Minimize $\mathcal{L}(\mathbf{w}, b) = \sum_i max(0, -y_i \times (\mathbf{w}^T\mathbf{x}_i + b))$

$$\frac{\mathcal{L}(\mathbf{w},b)}{\partial\mathbf{w}} = \sum_i \begin{cases} 0 & if\ y_i = sign(\mathbf{w}^T\mathbf{x}_i + b) \\ -\frac{1}{2}(target_i - output_i)\mathbf{x}_i & otherwise \end{cases}$$

Gradient decent: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \lambda\frac{\mathcal{L}(\mathbf{w},b)}{\mathbf{w}}$

$\lambda = 2$ (learning rate)

Note the update rule for the perceptron is:

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + (target_i - output_i) \times \mathbf{x}_i$$

if $target_i \neq output_i$

Perceptron training is a special gradient descent algorithm

Perceptron: $f(\mathbf{x};\mathbf{w}) = sign(\mathbf{w}^T\mathbf{x} + b)$

$S = \{(\mathbf{x}_i, y_i), i = 1..n\}$

no penalty if y_i and $(\mathbf{w}^T\mathbf{x}_i + b)$ have the same sign

Training: Minimize $\mathcal{L}(\mathbf{w}, b) = \sum_i max(0, -y_i \times (\mathbf{w}^T\mathbf{x}_i + b))$

$$\frac{\mathcal{L}(\mathbf{w},b)}{\partial b} = \sum_i \begin{cases} 0 & if\ y_i = sign(\mathbf{w}^T\mathbf{x}_i + b) \\ -\frac{1}{2}(target_i - output_i) & otherwise \end{cases}$$

Gradient decent: $b_{t+1} \leftarrow b_t - \lambda \frac{\mathcal{L}(\mathbf{w},b)}{b}$

$\lambda = 2$ (learning rate)

Note the update rule for the perceptron is:

$$b_{t+1} \leftarrow b_t + (target_i - output_i)$$

if $target_i \neq output_i$

Perceptron Learning Algorithm

- Initialize the weights (however you choose)
  - $w_1x_1 + w_2x_2 + b$ (initialize w_1, w_2, and b)
- Step 1: Choose a data point.
- Step 2: Compute the model output for the data point.
- Step 3: Compare model output to the target output.
  - If correct classification, go to Step 5!
  - If not, go to Step 4.

Perceptron Learning Algorithm

- Step 4: Update weights using perceptron learning rule.

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + (target_i - output_i) \times \mathbf{x}_i$$
$$b_{t+1} \leftarrow b_t + (target_i - output_i)$$

- Step 5: If you have visited all the data points and they are all corrected classifier, then exit; otherwise visit next data point and go back to step 2.

Initialize the weights

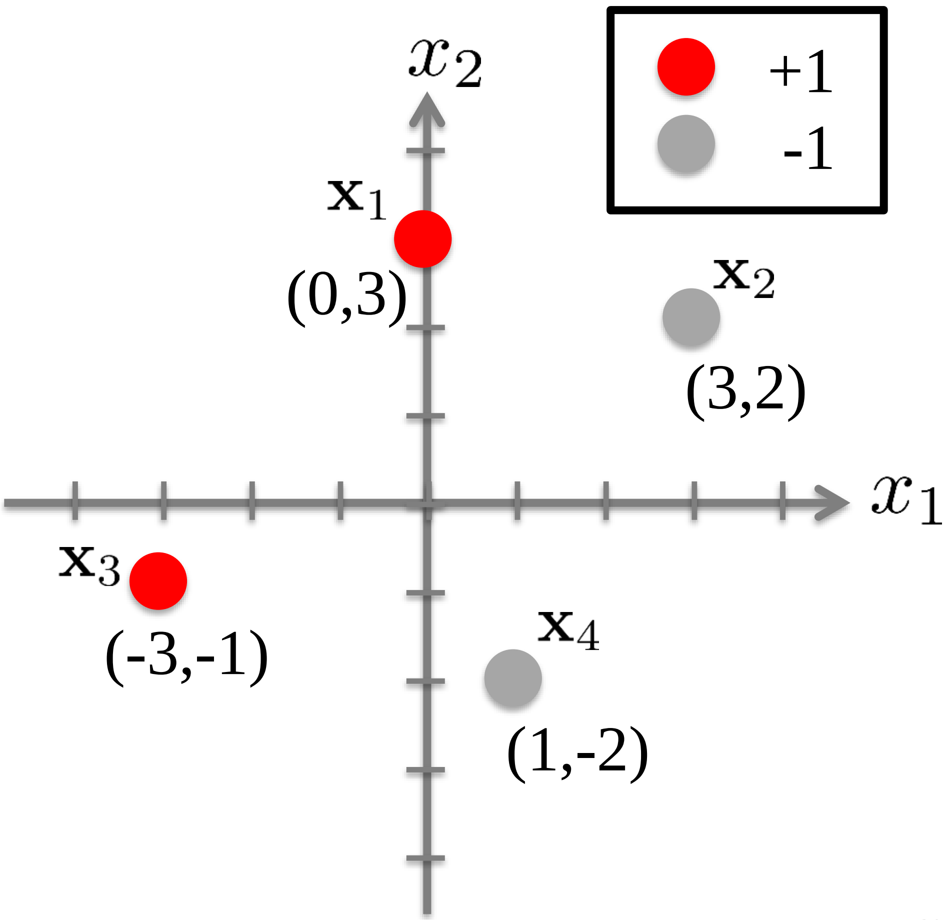
Given a training dataset:

$$S = \{(\mathbf{x}_1 = (0,3), y_1 = +1),$$

$$(\mathbf{x}_2 = (3,2), y_2 = -1),$$

$$(\mathbf{x}_3 = (-3,-1), y_3 = +1),$$

$$(\mathbf{x}_4 = (1,-2), y_4 = -1)\}$$



Initialize the weights

- Choose randomly – but start the weights small!

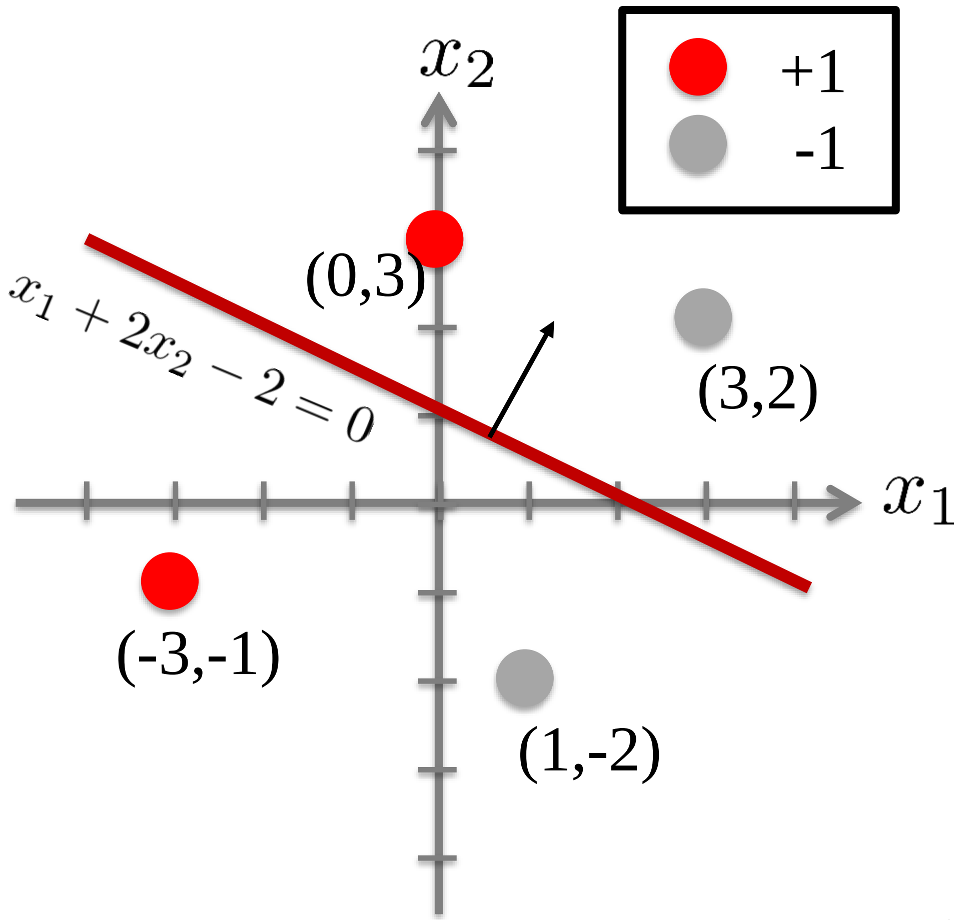
Decision boundary:
$w_1 x_1 + w_2 x_2 + b = 0$

- We'll choose:

$$w_1 = 1 \quad w_2 = 2 \quad b = -2$$

$$\Downarrow$$

$$x_1 + 2x_2 - 2 = 0$$

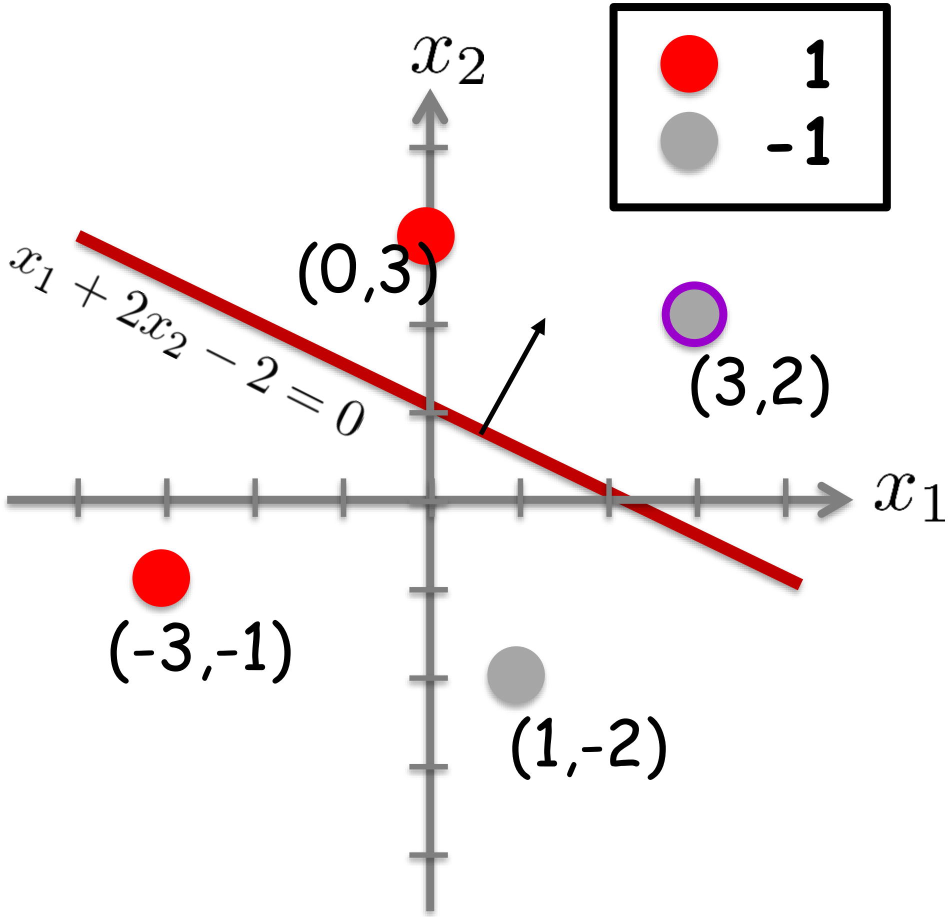


Step 1: Choose a point

- Choose randomly (or sequentially), but remember which you choose!

Say: $(\mathbf{x}_2 = (3, 2), y_2 = -1)$

It's ground-truth label (target) $= -1$: a negative sample.

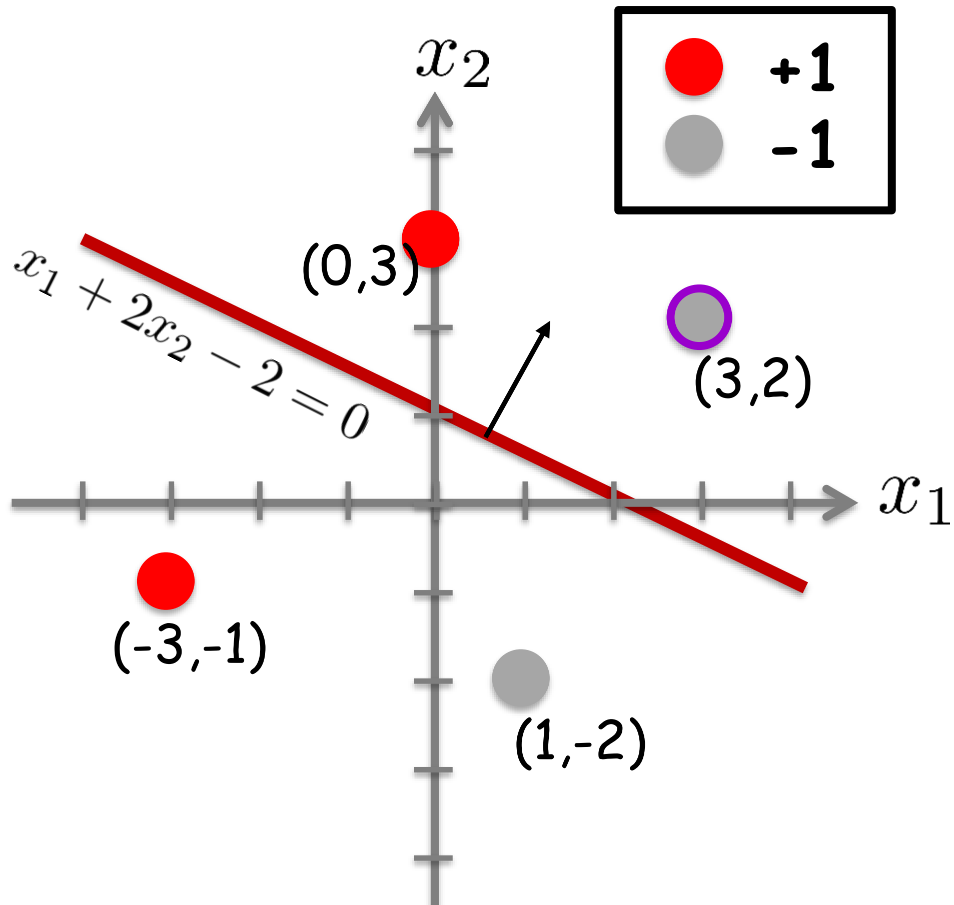


Step 1: Choose a point

Say: $(\mathbf{x}_2 = (3, 2), y_2 = -1)$

$w_1 = 1 \quad w_2 = 2 \quad b = -2$

It's ground-truth label (target) $= -1$: a negative sample.



Perceptron:

$$f(\mathbf{x}|w_1, w_2, b) = \begin{cases} +1 & if\ x_1 + 2x_2 - 2 \geq 0 \\ -1 & otherwise \end{cases}$$