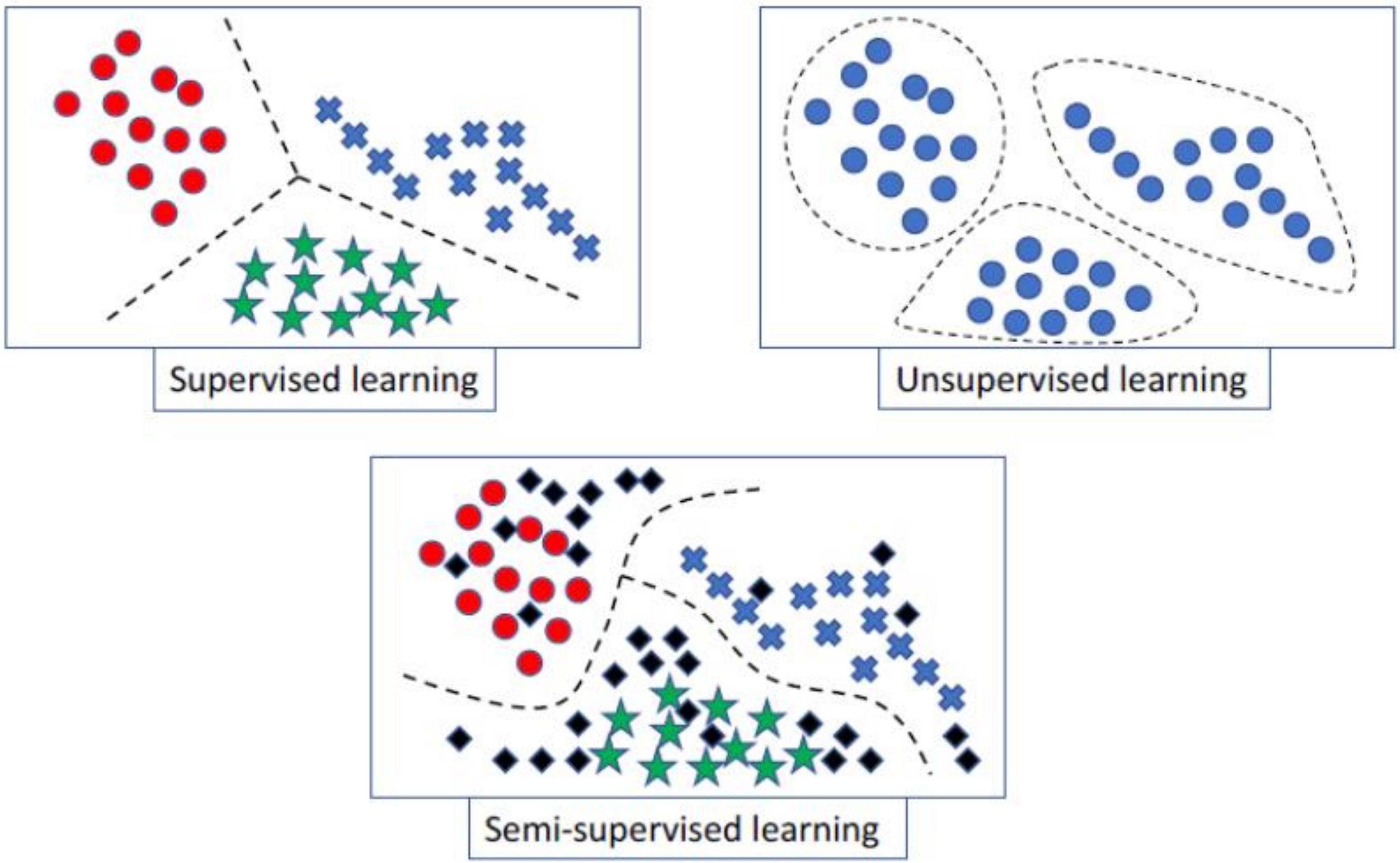
01.112/50.007 Machine Learning

Lecture 2

Perceptron

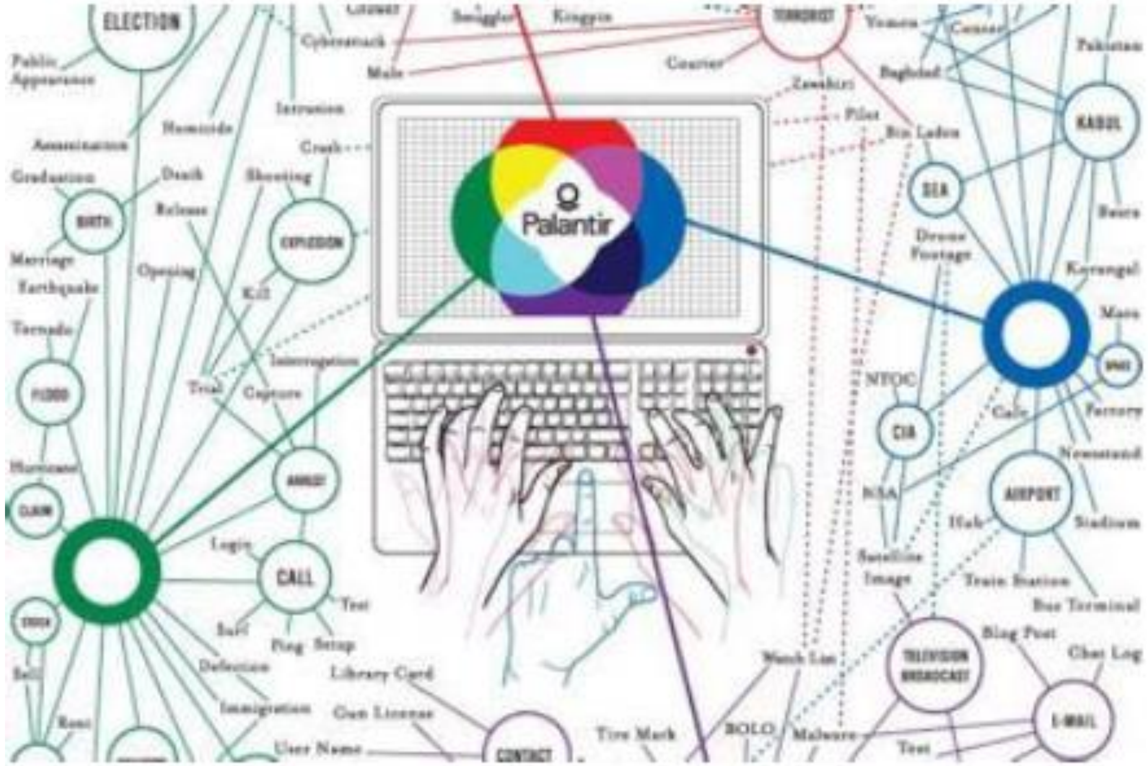
Recap

Types of Machine Learning



Examples

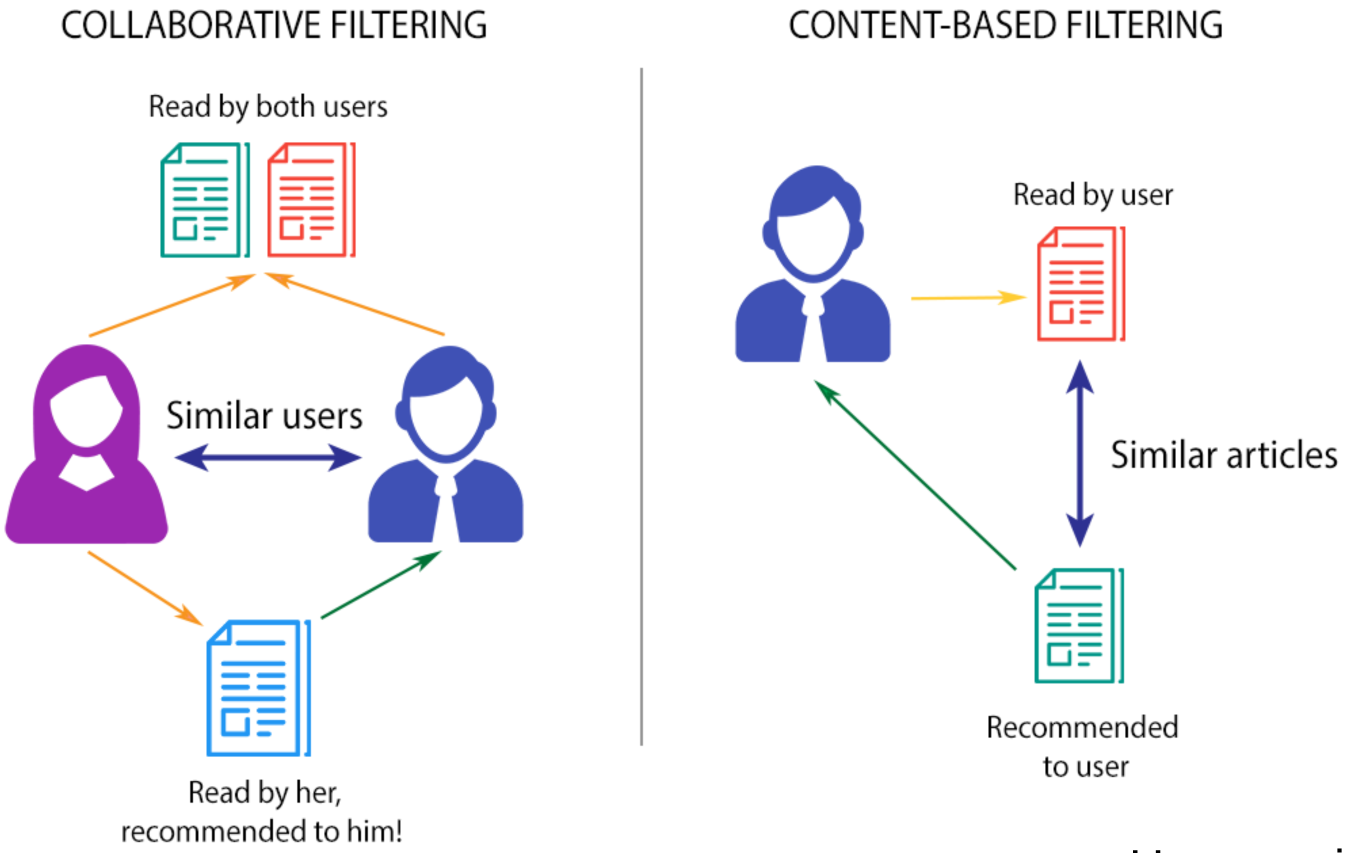
Fraud detection



Supervised learning

https://www.washingtonian.com/2012/01/31/killer-app/

Recommender Systems



Unsupervised learning

https://towardsdatascience.com/brief-on-recommender-systems-b86a1068a4dd

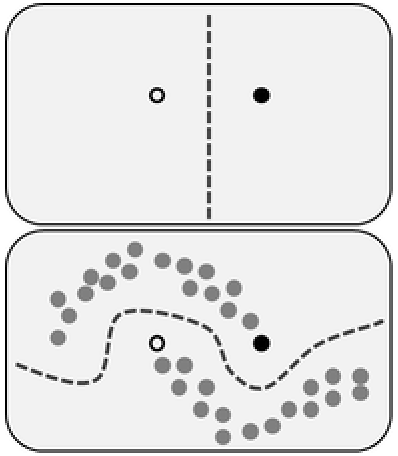
Spam Filters

Bayesian Networks

Supervised learning

Spam Filters

With some labelled emails (spam/not spam) and unlabelled emails in your inbox, we can create a **customized spam filter** for new emails using semi-supervised learning.



Labelled

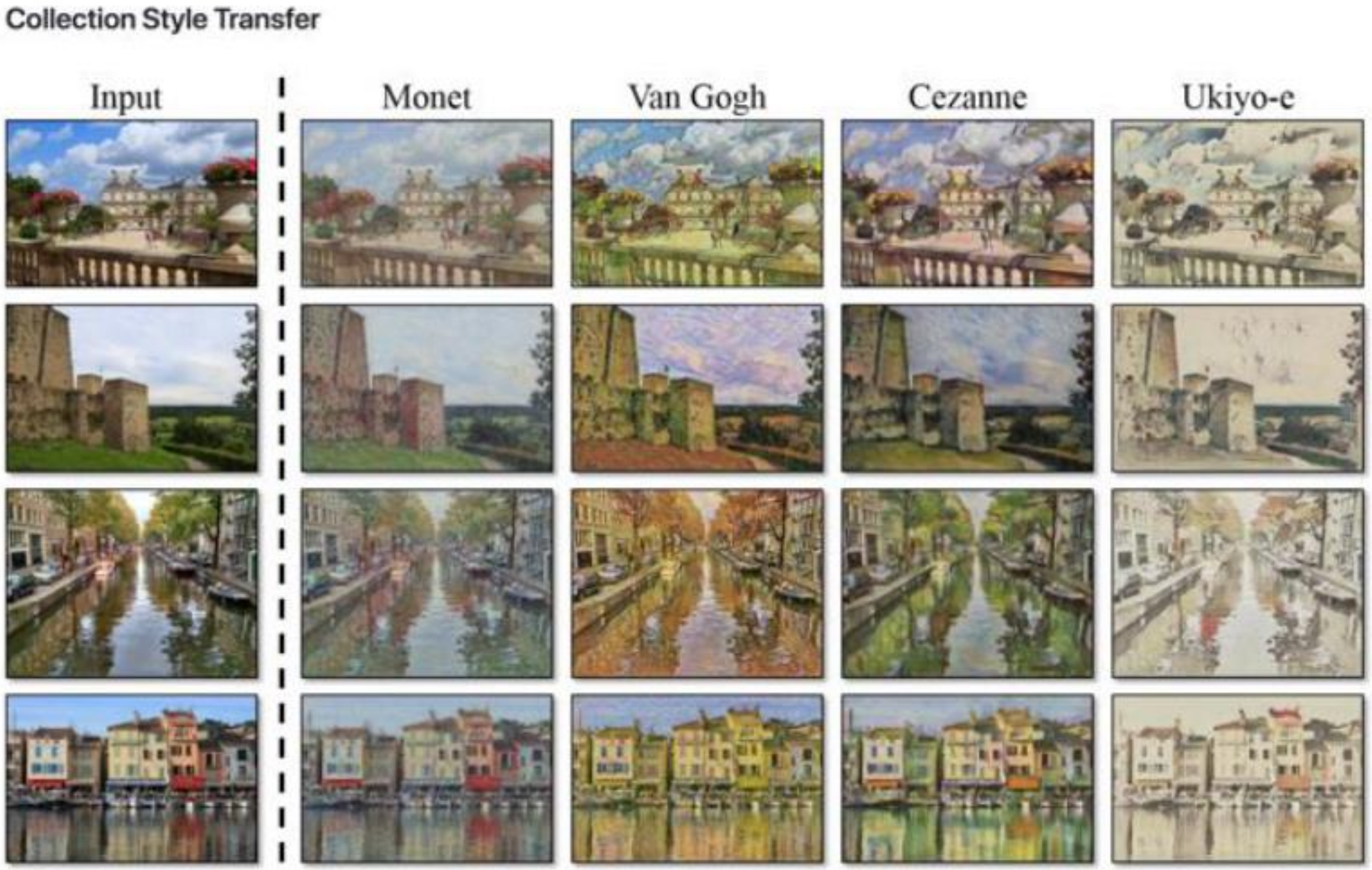
Unlabelled

Performing clustering and then labelling clusters with labelled data

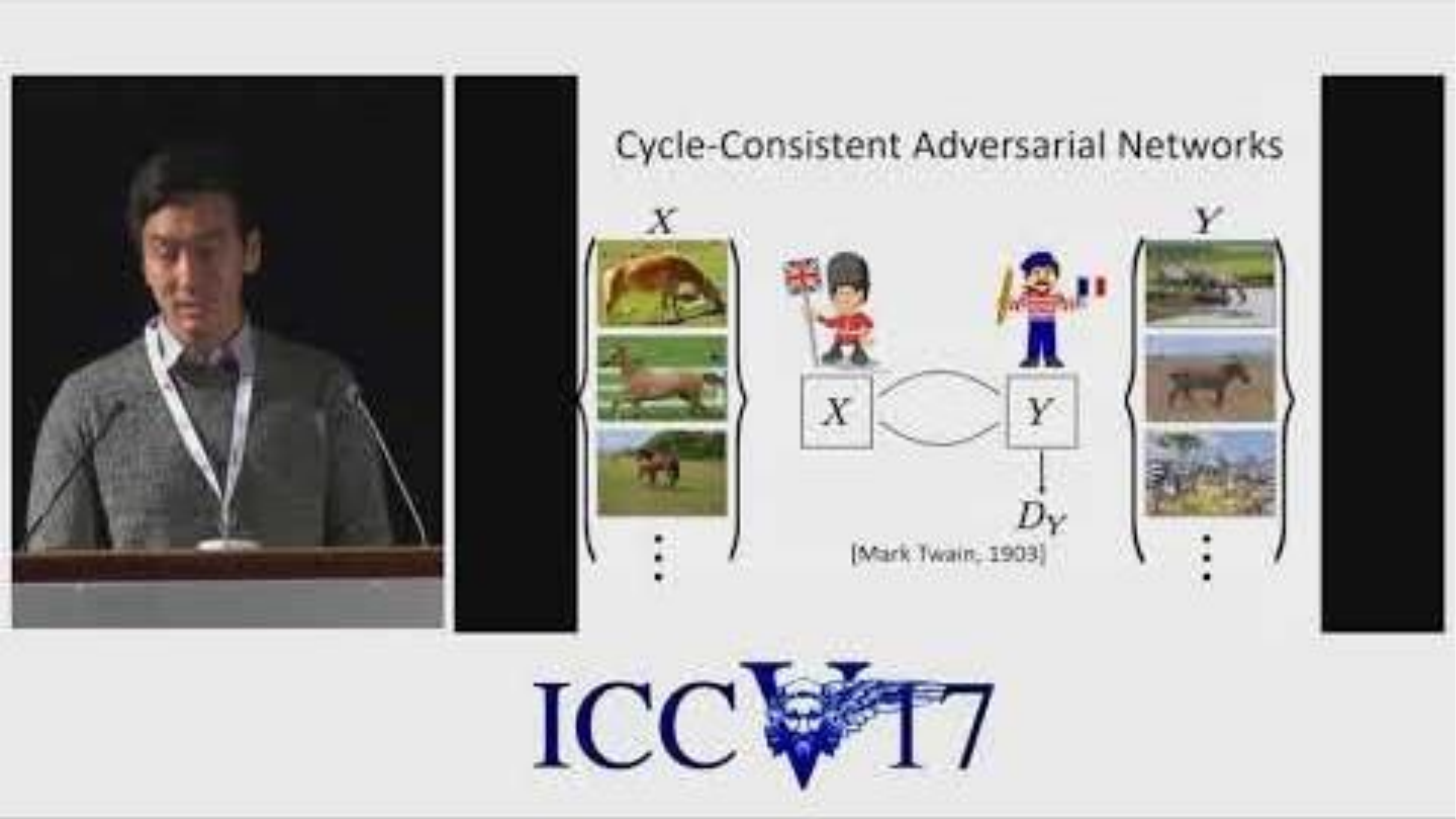
Semi-Supervised learning

https://en.wikipedia.org/wiki/Semi-supervised_learning

Generative Models



Generative Models



Ref: https://junyanz.github.io/CycleGAN/

Linear Classifier

Linear Classifier

- Let's consider a particular constrained set of classifiers (through origin)

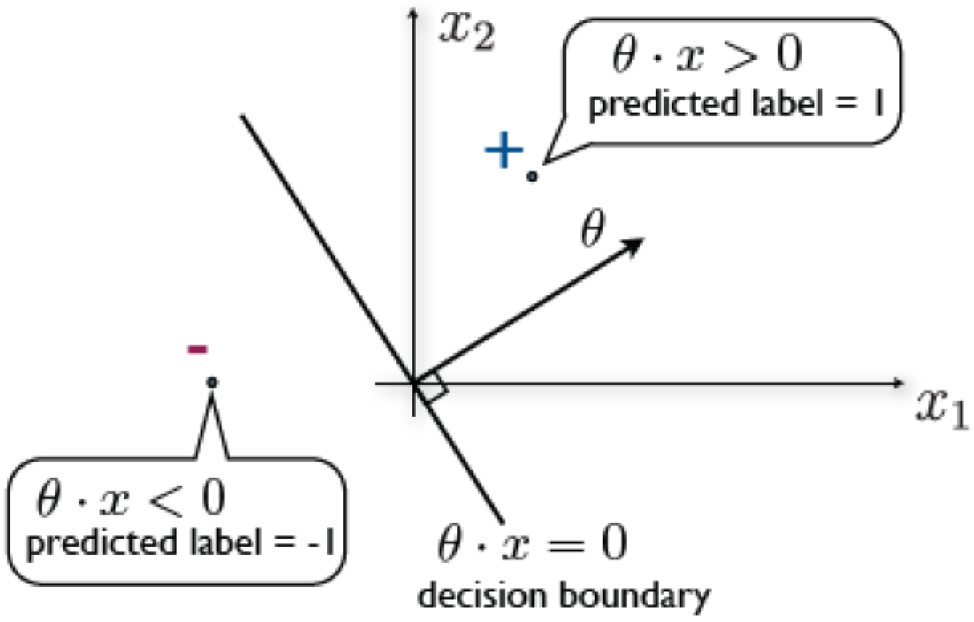
$$h(x;\theta) = \text{sign}(\theta_1 x_1 + \ldots + \theta_d x_d) = \text{sign}(\theta \cdot x) = \begin{cases} +1, & \theta \cdot x \geq 0 \\ -1, & \theta \cdot x < 0 \end{cases}$$

- $\theta \cdot x = \theta^T x$ and $\theta = [\theta_1, \ldots, \theta_d]^T$ is a column vector of real valued parameters or weights.

- Different settings of the **weights** give rise to **different classifiers**.

Linear Classifier

$$h(x;\theta) = \text{sign}(\theta_1 x_1 + \ldots + \theta_d x_d) = \text{sign}(\theta \cdot x) = \begin{cases} +1, & \theta \cdot x \geq 0 \\ -1, & \theta \cdot x < 0 \end{cases}$$



Linear Classifier

- Linear classifier through origin:

$$h(x;\theta) = \text{sign}(\theta_1 x_1 + \ldots + \theta_d x_d) = \text{sign}(\theta \cdot x) = \begin{cases} +1, & \theta \cdot x \geq 0 \\ -1, & \theta \cdot x < 0 \end{cases}$$

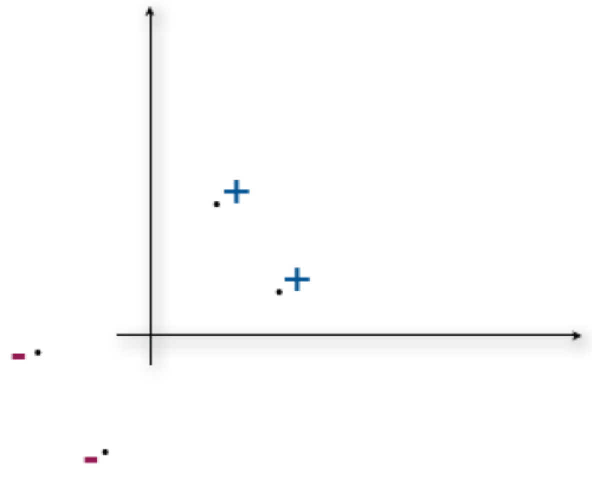
- **Training error:**

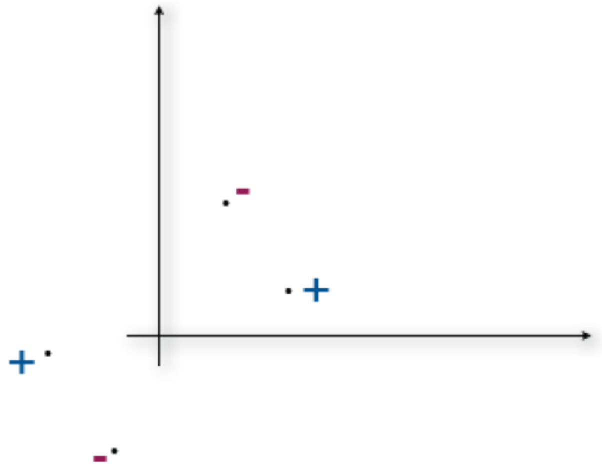
$$\mathcal{E}_n(\theta) = \frac{1}{n}\sum_{t=1}^{n} [\![y^{(t)} \neq h(x^{(t)};\theta)]\!] = \frac{1}{n}\sum_{t=1}^{n} [\![y^{(t)}(\theta \cdot x^{(t)}) \leq 0]\!]$$

- Linear classifier that achieves zero training error is called **realizable**.

Linear Classifier

Definition 1.1 *Training examples* $S_n = \{(x^{(t)}, y^{(t)}), t = 1, \ldots, n\}$ *are linearly separable through origin if there exists a parameter vector* $\hat{\theta}$ *such that* $y^{(t)}(\hat{\theta} \cdot x^{(t)}) > 0$ *for all* $t = 1, \ldots, n$.





How do we find the weights such that they **minimize** the training error?

Perceptron update rule

- Initialize the **weight** ($\theta = 0$).
- For each training example 't' in S_n, classify the instance
  - if the prediction was correct, continue
  - else, $\theta^{(k+1)} = \theta^{(k)} + y^{(t)}x^{(t)}$
- Terminate if the **training error** is zero (realizable) or a pre-determined number of iterations are completed (non-realizable).

Perceptron update rule

What does the update rule ($\theta^{(k+1)} = \theta^{(k)} + y^{(t)} x^{(t)}$) do?

- If the classifier predicted an instance that was negative but it should have been positive…
  - Currently: $\theta \cdot x < 0$
  - Want: $\theta \cdot x \geq 0$
- Adjust the weights θ so that this function values move toward positive.
- If the classifier predicted positive but it should have been negative, shift the weights so that the value moves toward negative.

Perceptron update rule

- If a classification mistake is made on sample 't', i.e.,

$$y^{(t)}(\theta \cdot x^{(t)}) \leq 0$$

- The updated weight vector is:

$$\theta^{(k+1)} = \theta^{(k)} + y^{(t)}x^{(t)}$$

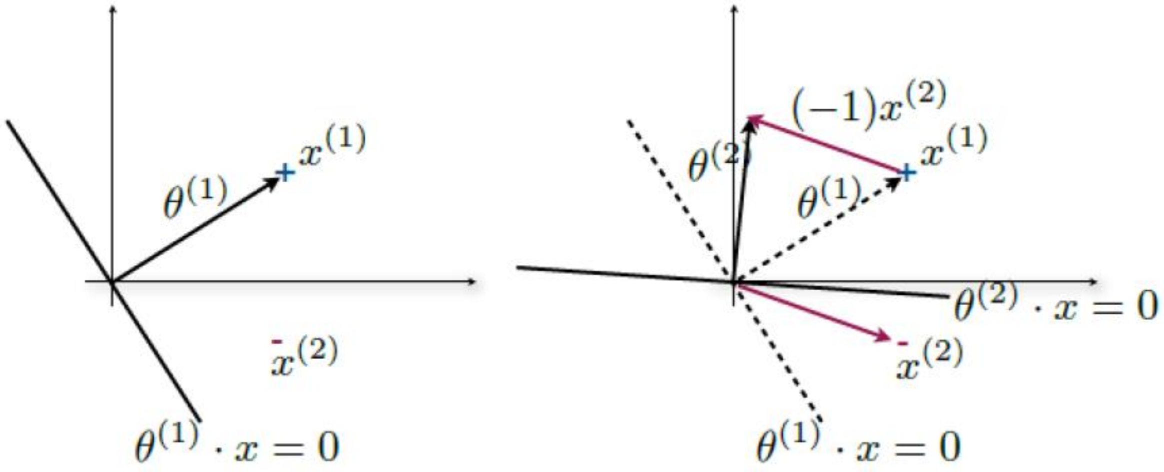
- With updated weights, $y^{(t)}(\theta^{(k)} \cdot x^{(t)})$ becomes more positive and eventually becomes > 0 in a realizable case.

$$\begin{aligned} y^{(t)}(\theta^{(k+1)} \cdot x^{(t)}) &= y^{(t)}(\theta^{(k)} + y^{(t)}x^{(t)}) \cdot x^{(t)} \\ &= y^{(t)}(\theta^{(k)} \cdot x^{(t)}) + (y^{(t)})^2(x^{(t)} \cdot x^{(t)}) \\ &= y^{(t)}(\theta^{(k)} \cdot x^{(t)}) + \left\| x^{(t)} \right\|^2 \end{aligned}$$

Perceptron update rule

- **Example:** Given the update rule, $\theta^{(k+1)} = \theta^{(k)} + y^{(t)} x^{(t)}$

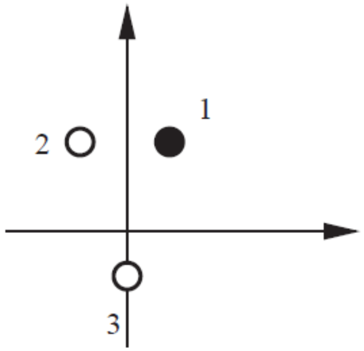
$$\theta^{(1)} = \theta^{(0)} + x^{(1)}$$
$$\theta^{(2)} = \theta^{(1)} + (-1) x^{(2)}$$



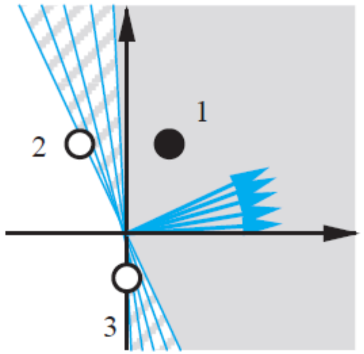
Perceptron update rule

Theorem 2.1 *The perceptron update rule converges after a finite number of mistakes when the training examples are linearly separable through origin.*

- **Test example:**



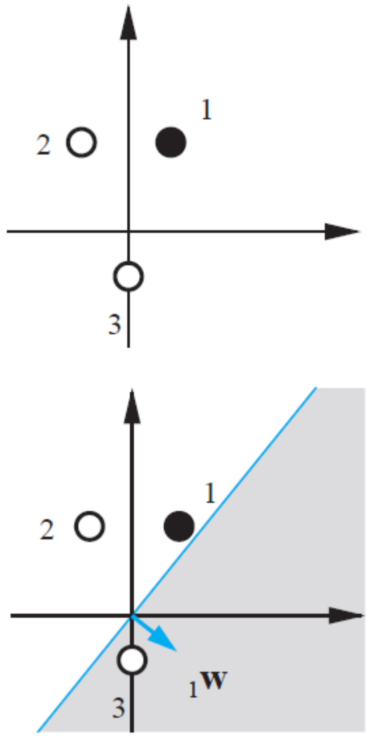
Input data



Weight vectors representing allowable decision boundaries

Perceptron update rule

- **Test example:**



$$x^{(1)} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}, y^{(1)} = 1$$

$$x^{(2)} = \begin{bmatrix} -1 \\ 2 \end{bmatrix}, y^{(2)} = -1$$

$$x^{(3)} = \begin{bmatrix} 0 \\ -1 \end{bmatrix}, y^{(3)} = -1$$

Perceptron update rule:

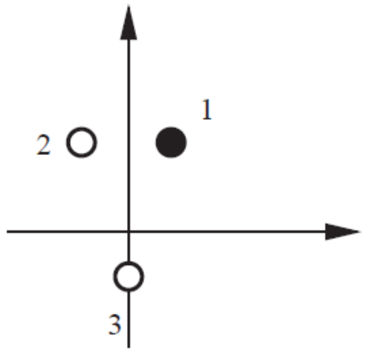
if $y(\theta \cdot x) \leq 0$ then

$$\theta^{(k+1)} = \theta^{(k)} + y^{(t)} x^{(t)}$$

$\theta^{(0)} = \begin{bmatrix} 1 \\ -0.8 \end{bmatrix}$ → **Random weight initialization**

Perceptron update rule

- **Test example:**



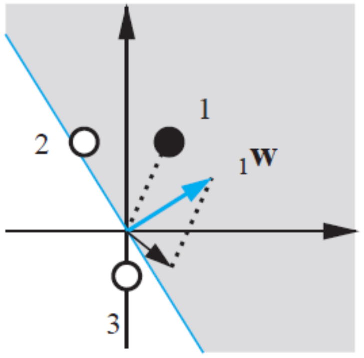
$x^{(1)} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}, y^{(1)} = 1$

$x^{(2)} = \begin{bmatrix} -1 \\ 2 \end{bmatrix}, y^{(2)} = -1$

$x^{(3)} = \begin{bmatrix} 0 \\ -1 \end{bmatrix}, y^{(3)} = -1$

Perceptron update rule:

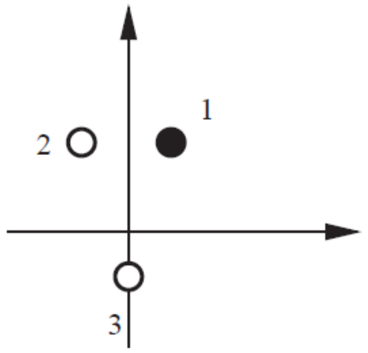
if $y(\theta \cdot x) \leq 0$ then
$\theta^{(k+1)} = \theta^{(k)} + y^{(t)} x^{(t)}$

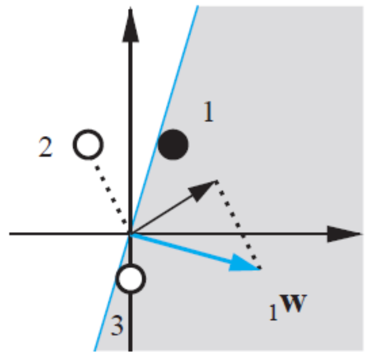


$\theta^{(1)} = \begin{bmatrix} 2 \\ 1.2 \end{bmatrix}$ → **1st Update**

Perceptron update rule

- **Test example:**





$$x^{(1)} = \begin{bmatrix}1\\2\end{bmatrix}, y^{(1)} = 1$$

$$x^{(2)} = \begin{bmatrix}-1\\2\end{bmatrix}, y^{(2)} = -1$$

$$x^{(3)} = \begin{bmatrix}0\\-1\end{bmatrix}, y^{(3)} = -1$$

Perceptron update rule:

if $y(\theta \cdot x) \leq 0$ then

$$\theta^{(k+1)} = \theta^{(k)} + y^{(t)} x^{(t)}$$

$\theta^{(2)} = \begin{bmatrix}3\\-0.8\end{bmatrix}$ → **2^{nd} Update**

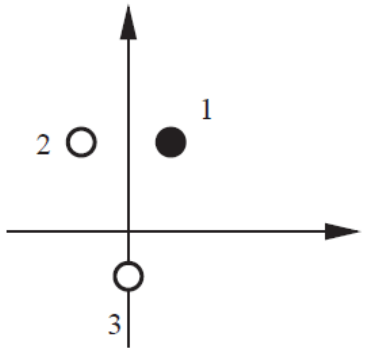
Perceptron update rule

- **Test example:**

$x^{(1)} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}, y^{(1)} = 1$

$x^{(2)} = \begin{bmatrix} -1 \\ 2 \end{bmatrix}, y^{(2)} = -1$

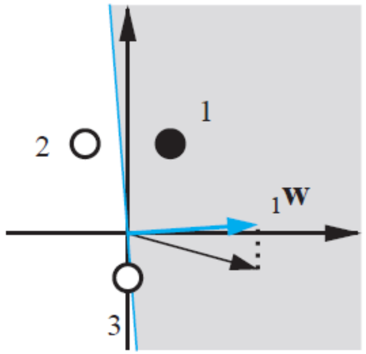
$x^{(3)} = \begin{bmatrix} 0 \\ -1 \end{bmatrix}, y^{(3)} = -1$



Perceptron update rule:

if $y(\theta \cdot x) \leq 0$ then

$\theta^{(k+1)} = \theta^{(k)} + y^{(t)} x^{(t)}$

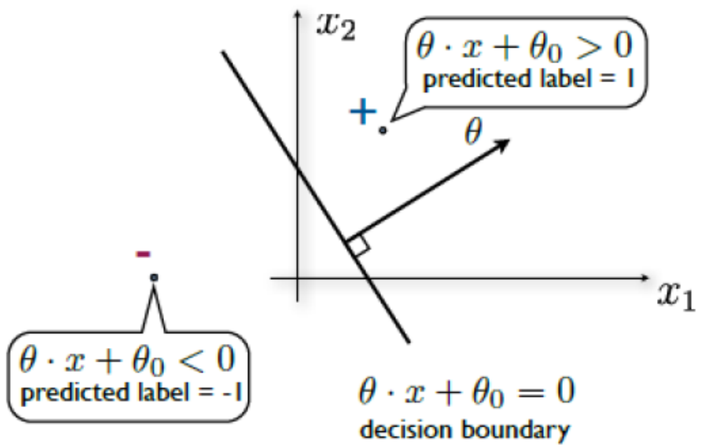


$\theta^{(3)} = \begin{bmatrix} 3 \\ 0.2 \end{bmatrix}$ → **3rd Update**

Linear Classifier with offset

$$h(x;\theta,\theta_0) = \text{sign}(\theta \cdot x + \theta_0) = \begin{cases} +1, & \theta \cdot x + \theta_0 \geq 0 \\ -1, & \theta \cdot x + \theta_0 < 0 \end{cases}$$

- The hyper-plane $\theta \cdot x + \theta_0 = 0$ is **oriented parallel** to $\theta \cdot x = 0$
- Whereas, θ is still **orthogonal** to the decision boundary and $\theta_0 < 0$



Linear Classifier with offset

- **Example:** Suppose we want to **predict** whether a web user will click on an ad for a refrigerator

- Four **features**:
  - Recently searched “refrigerator repair”
  - Recently searched “refrigerator reviews”
  - Recently bought a refrigerator
  - Has clicked on any ad in the recent past
- These are all **binary features** (values can be either 0 or 1)

Linear Classifier with offset

- Suppose these are the weights

Searched “repair”	2.0
Searched “reviews”	8.0
Recent purchase	-15.0
Clicked ads before	5.0
b (intercept)	-9.0

- Evaluate the linear classifier with offset

$$h(x;\theta,\theta_0) = \begin{cases} +1, & \theta \cdot x + \theta_0 \geq 0 \\ -1, & \theta \cdot x + \theta_0 < 0 \end{cases}$$

Linear Classifier with offset

- Suppose these are the weights

Searched “repair”	2.0
Searched “reviews”	8.0
Recent purchase	-15.0
Clicked ads before	5.0
b (intercept)	-9.0

- $\theta.\mathrm{x} + \theta_0 = (2 * 0) + (8 * 1) + (-15 * 0) + (5 * 0) + (-9)$
  $= 8 - 9 = -1$

- Prediction = No

Linear Classifier with offset

- Suppose these are the weights

Searched “repair”	2.0
Searched “reviews”	8.0
Recent purchase	-15.0
Clicked ads before	5.0
b (intercept)	-9.0

- $\theta . \mathrm{x} + \theta_0 = (2 * 1) + (8 * 1) + (-9) = 1$
- Prediction = Yes

Linear Classifier with offset

- Suppose these are the weights

Searched “repair”	2.0
Searched “reviews”	8.0
Recent purchase	-15.0
Clicked ads before	5.0
b (intercept)	-9.0

- $\theta . \mathrm{x} + \theta_0 = (8 * 1) + (5 * 1) + (-9) = 4$
- Prediction = Yes

Linear Classifier with offset

- Suppose these are the weights

Searched "repair"	2.0
Searched "reviews"	8.0
Recent purchase	-15.0
Clicked ads before	5.0
b (intercept)	-9.0

- $\theta.\mathrm{x} + \theta_0 = (8 * 1) + (-15 * 1) + (5 * 1) + (-9) = -11$
- Prediction = No
- If someone bought a refrigerator recently, they probably aren't interested in shopping for another one anytime soon.

Linear Classifier with offset

- Suppose these are the weights

Searched “repair”	2.0
Searched “reviews”	8.0
Recent purchase	-15.0
Clicked ads before	5.0
b (intercept)	-9.0

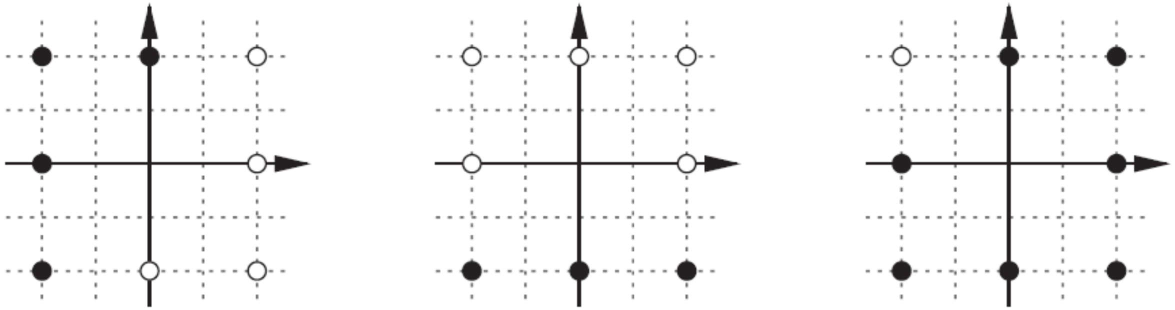
- $\theta . \mathrm{x} + \theta_0 = -9$
- Prediction = No
- Since most people don’t click ads, the “default” prediction is that they will not click (the intercept pushes it to negative).

Linear Classifier with offset

- If training examples are linearly separable through origin, they are also linearly separable with offset. Is the converse true? Why not?

Linear Classifier with offset

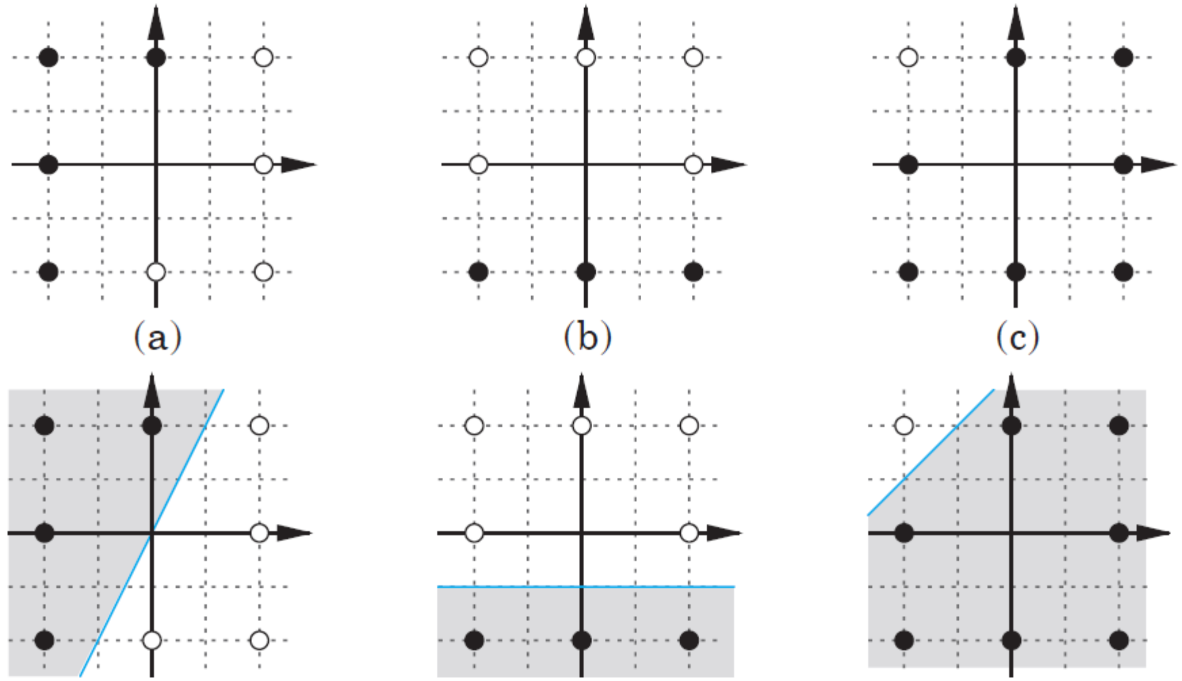
- If training examples are linearly separable through origin, they are also linearly separable with offset. Is the converse true? Why not?



(a) (b) (c)

Linear Classifier with offset

- If training examples are linearly separable through origin, they are also linearly separable with offset. Is the converse true? Why not?



Linear Classifier with offset

Perceptron update rule:

- Initialize the **weight** ($\theta = 0$).
- For each training example 't' in S_n, classify the instance
  - if correct, continue
  - else, $\theta^{(k+1)} = \theta^{(k)} + y^{(t)} x^{(t)}$

    $\theta_0^{(k+1)} = \theta_0^{(k)} + y^{(t)}$
- Terminate if the **training error** is zero (realizable) or a pre-determined number of iterations are completed (non-realizable).

Intended Learning Outcomes

- Given a set of training examples, find out if they are **linearly separable**.
- Use of **Perceptron algorithm** to select the best classifier in a **realizable case**.
- Application of the **Perceptron algorithm to real data**.