50.007 Machine Learning

Lecture 2

Perceptron

Different kinds of learning problems

A Case Study of Supervised Learning

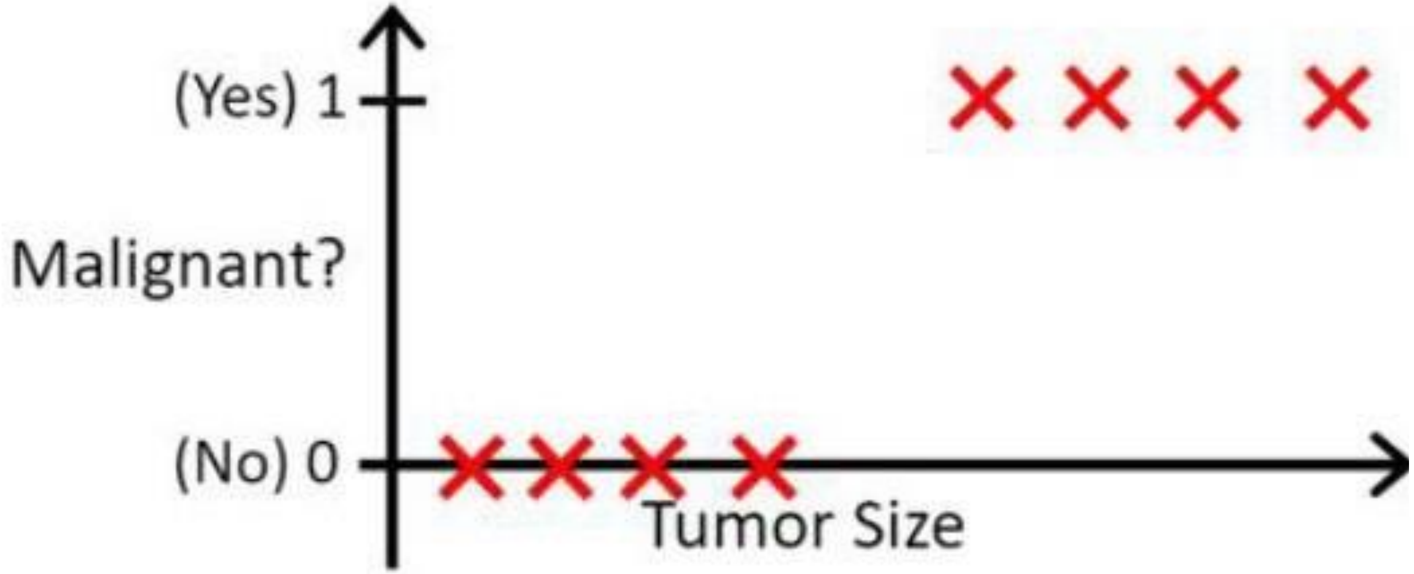
Linear Classifier without Offset

Linear Classifier with Offset

Learning Problems

Supervised Learning

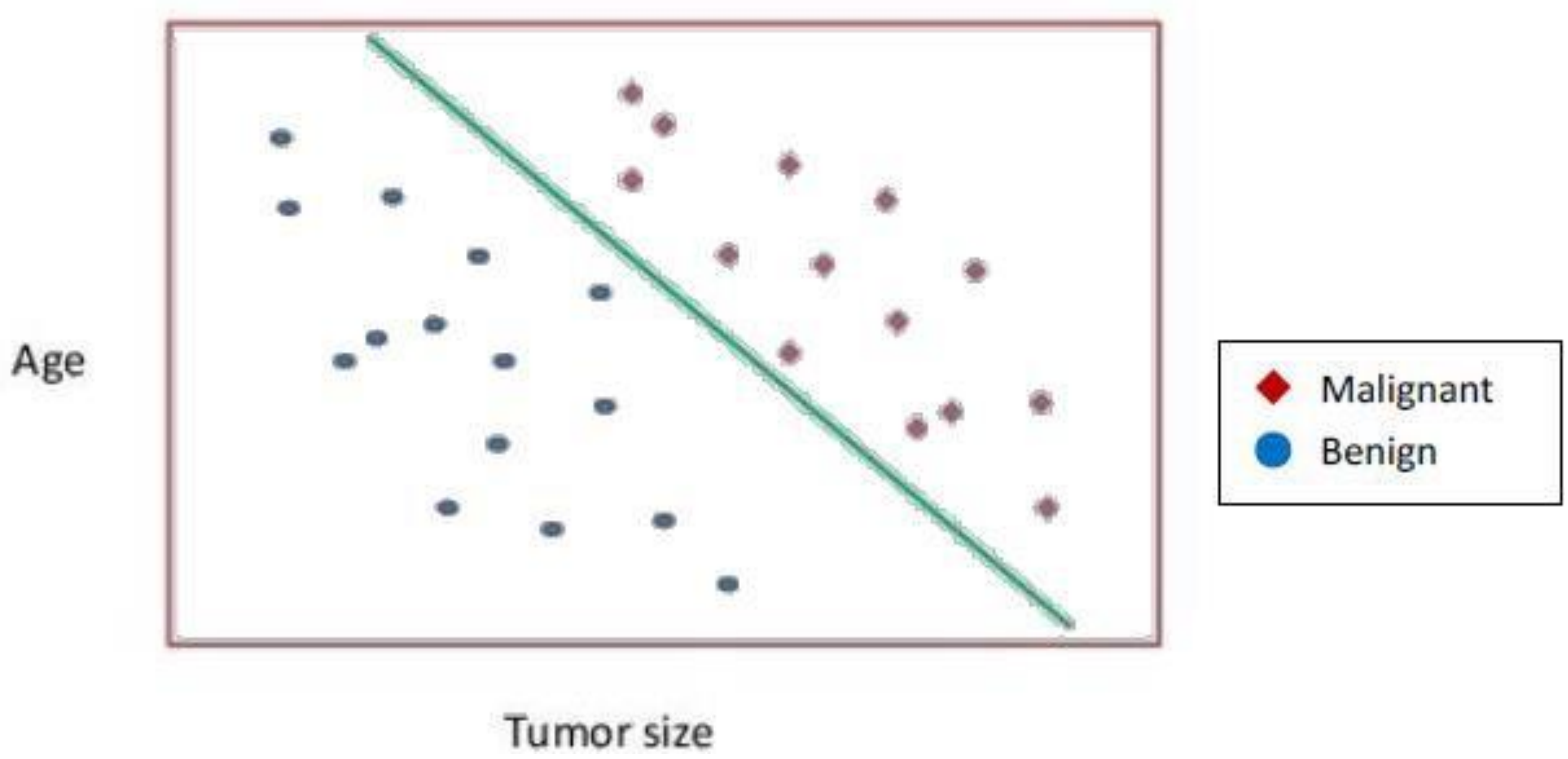
- **Classification (1-d features)**



Learning a function
$y = f(x)$
$x \in \mathbb{R}$
$y \in \{1, 2, \dots, k\}$

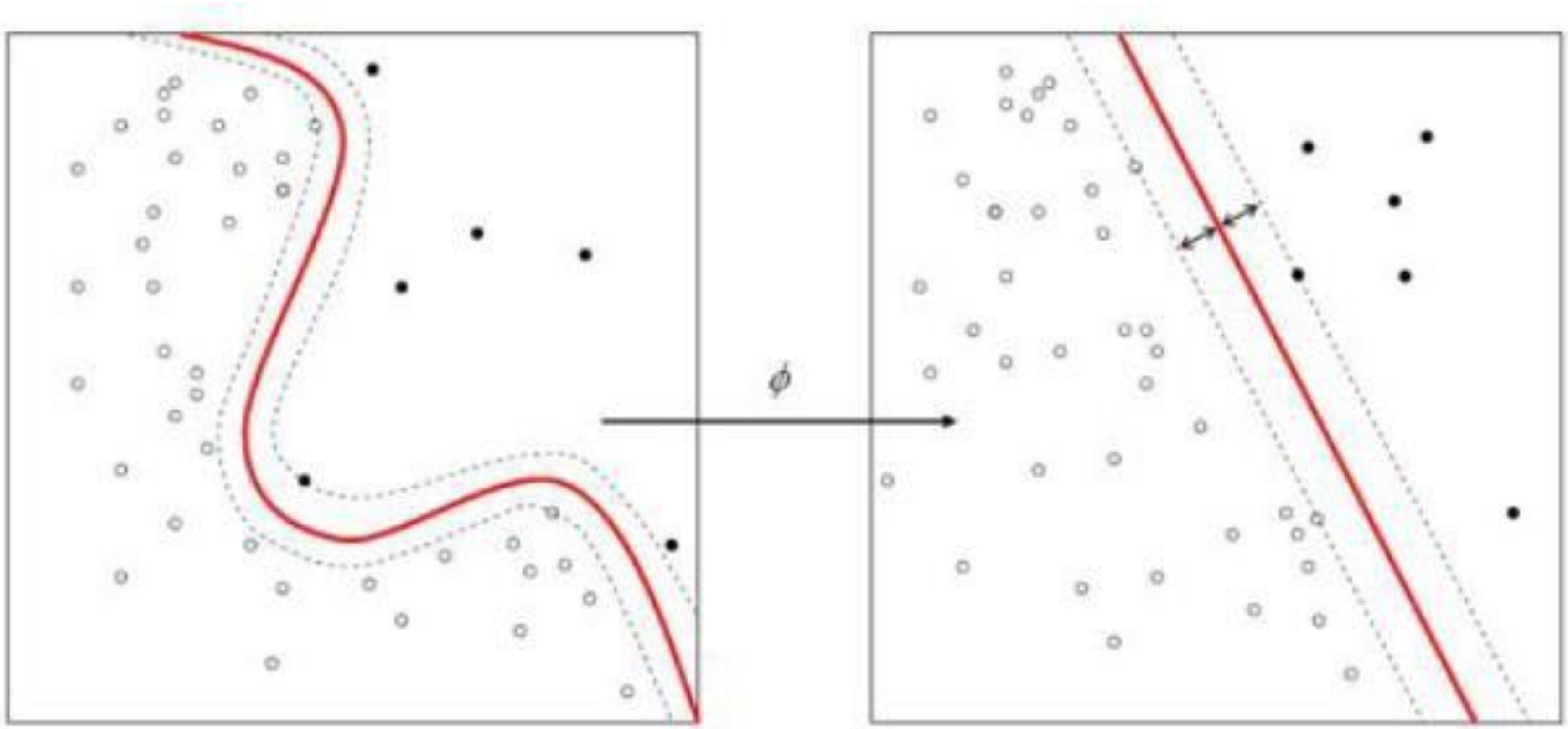
Supervised Learning

- **Classification (2-d features) - Linear**



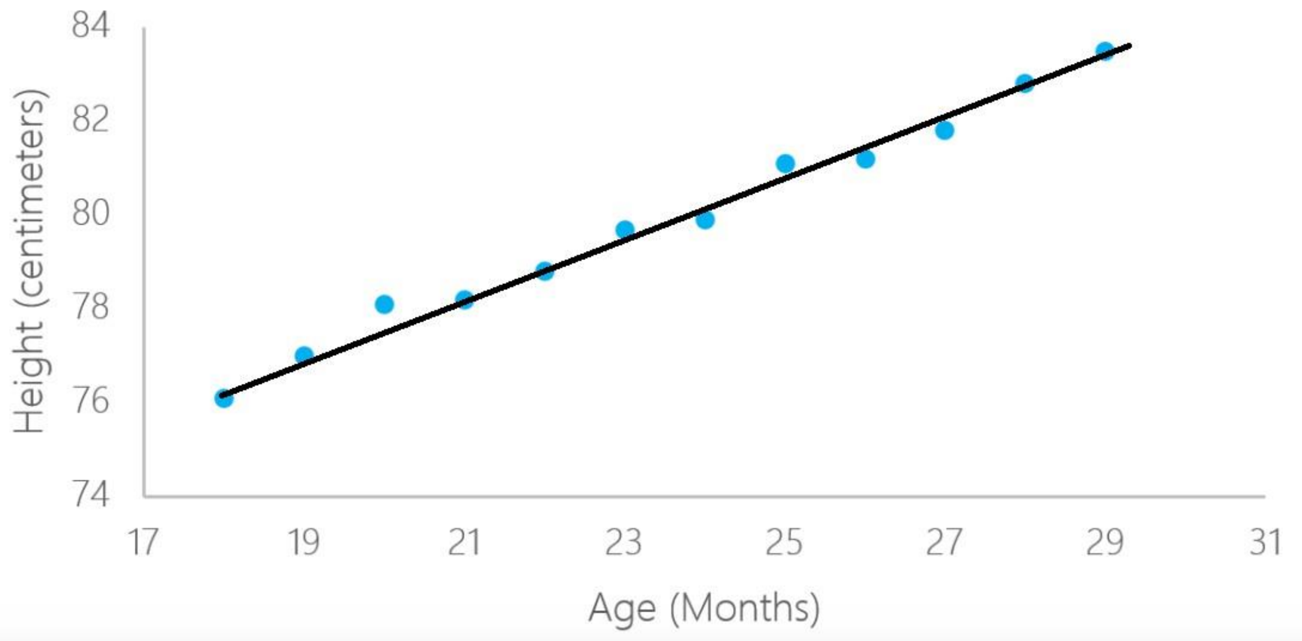
Supervised Learning

- **Classification (Non-Linear)**



Supervised Learning

- **Regression (Linear)**



Learning a function

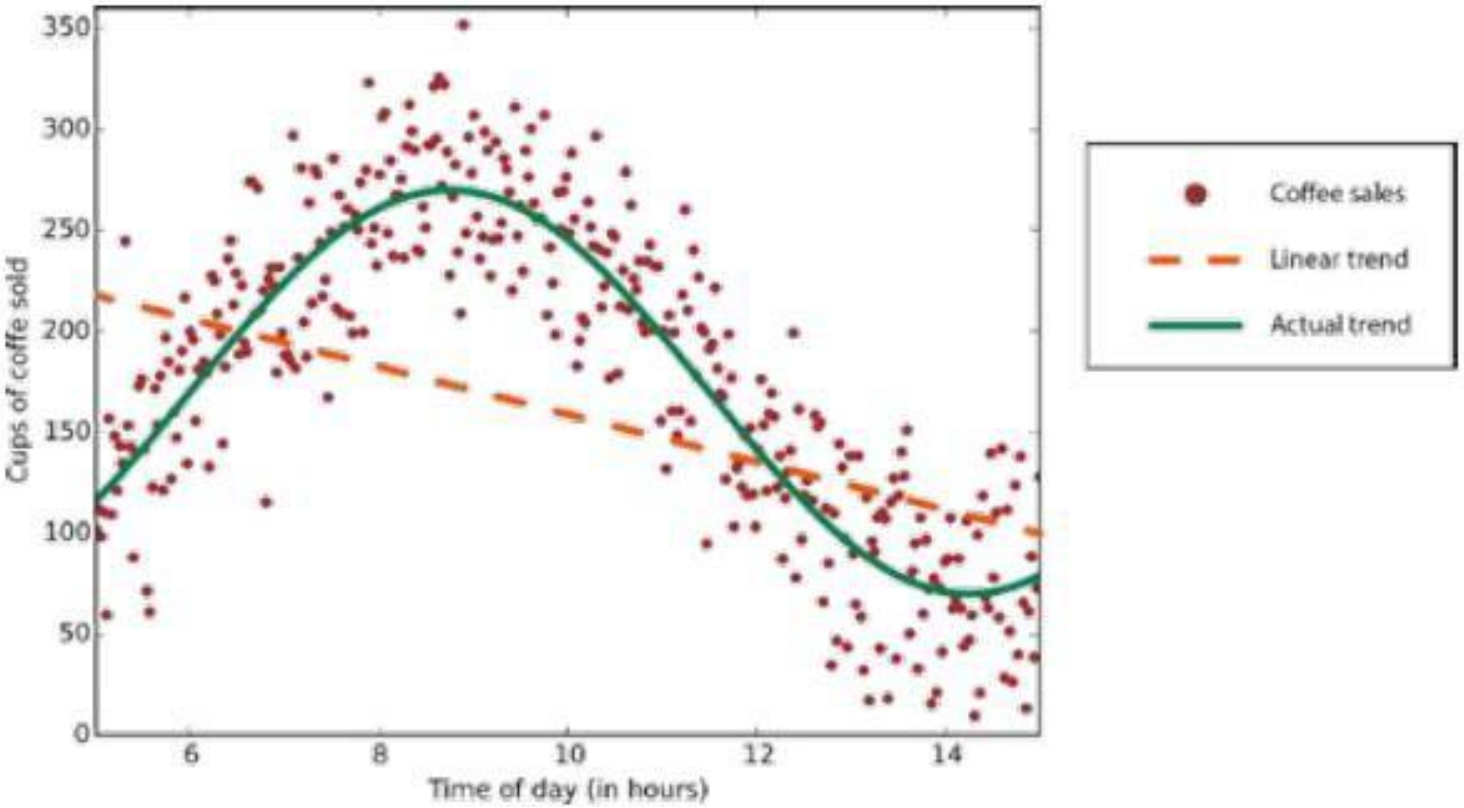
$$y = f(x)$$

$$x \in \mathbb{R}$$

$$y \in \mathbb{R}$$

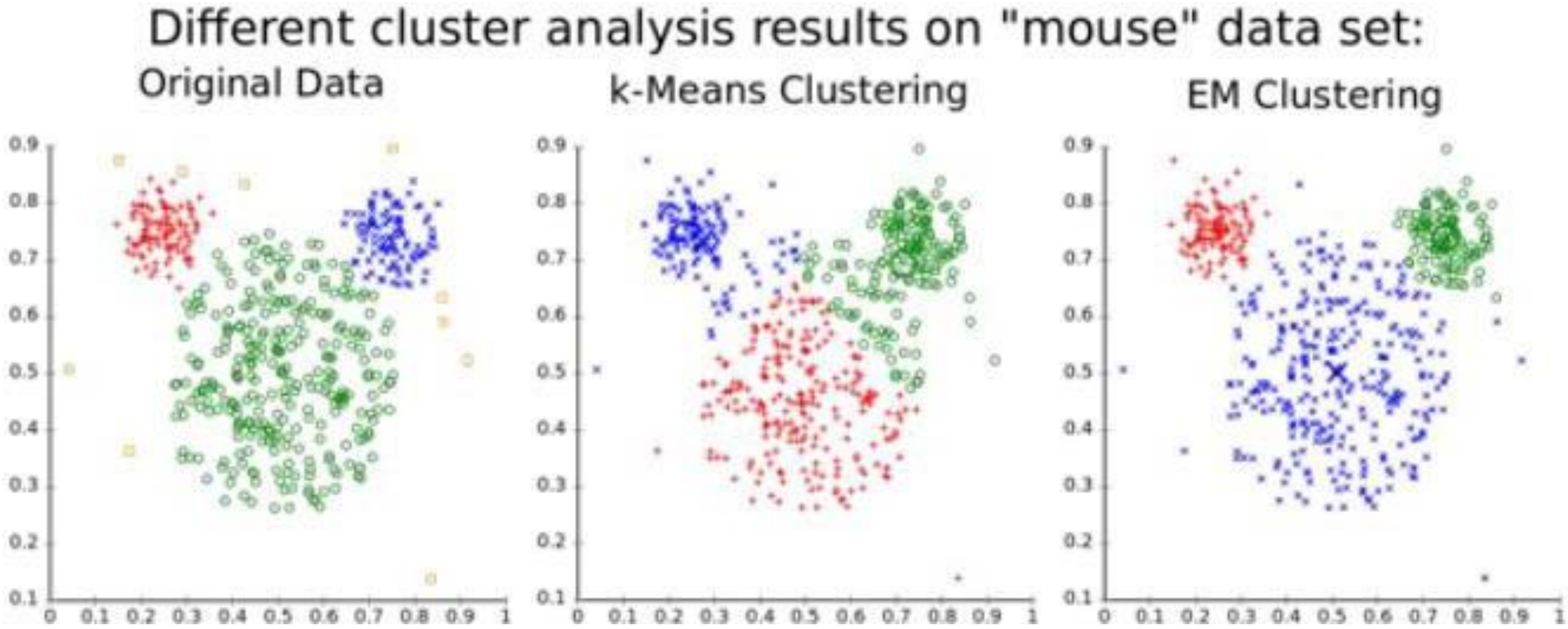
Supervised Learning

- **Regression (Non-Linear)**



Unsupervised Learning

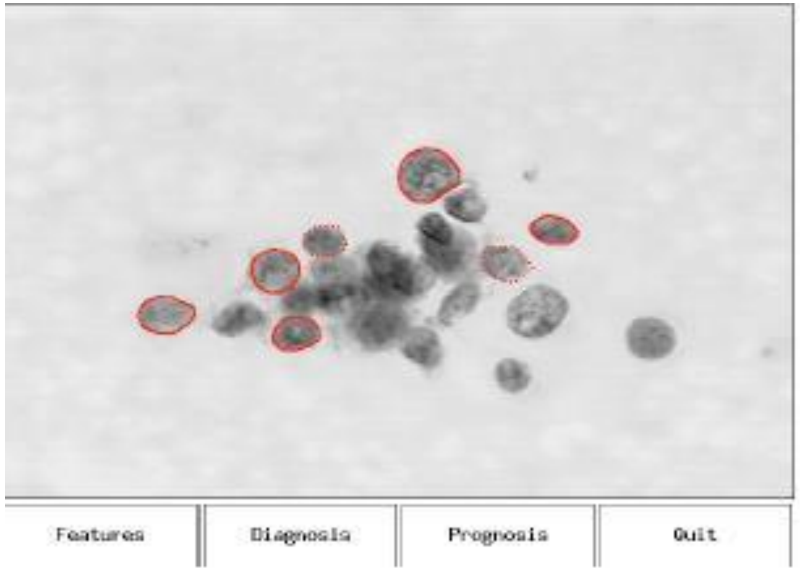
- **Clustering**



A case study

Supervised Learning

Example: A dataset



- Cell samples were taken from tumors in breast cancer patients before surgery, and imaged
- Tumors were excised
- Patients were followed to determine whether or not the cancer recurred, and how long until recurrence or disease free

Example: A dataset

- 30 real-valued variables per tumour
- Two variables that can be predicted:
  - Outcome (R = recurrent, N = non-recurrent)
  - Time (until recurrence, for R, time healthy for N).

tumor size	texture	perimeter	. . .	outcome	time
18.02	27.6	117.5		N	31
17.99	10.38	122.8		N	61
20.29	14.34	135.1		R	27

Terminology

tumor size	texture	perimeter	...	outcome	time
18.02	27.6	117.5		N	31
17.99	10.38	122.8		N	61
20.29	14.34	135.1		R	27

- Columns are called *input variables* or *features* or *attributes*
- The outcome and time (which we are trying to predict) are called *output variables* or *targets or responses*.
- A row in the table is called *training example* or *instance*
- The whole table is called (*training) data set*.
- The problem of predicting the recurrence is called *(binary) classification*.
- The problem of predicting the time is called *regression*.

More formally

tumor size	texture	perimeter	...	outcome	time
18.02	27.6	117.5		N	31
17.99	10.38	122.8		N	61
20.29	14.34	135.1		R	27

Training data

$$\mathcal{S}_n = \{ (x^{(i)}, y^{(i)}) \mid i = 1, \dots, n \}$$

- Features/Inputs $x^{(i)} = \left(x_1^{(i)}, \dots, x_d^{(i)}\right)^{\mathsf{T}} \in \mathbb{R}^d$
- Response/Output $y^{(i)} \in \mathbb{R}$ or $y \in \{1, 2, \dots, k\}$

More formally

- Let $\mathcal{X}$ denote the space of input values
- Let $\mathcal{Y}$ denote the space of output values
- Given a data set $\mathcal{S}_n \subset \mathcal{X} \times \mathcal{Y}$, find a function:

$$h : \mathcal{X} \rightarrow \mathcal{Y}$$

  such that $h(\mathrm{x})$ is a *"good predictor"* for the value of y.
- h is called a *classifier*
- Problems are categorized by the type of output domain
  - If $\mathcal{Y} = \mathbb{R}$, this problem is called *regression*
  - If $\mathcal{Y}$ is a categorical variable (i.e., part of a finite discrete set), the problem is called *classification*
  - If $\mathcal{Y}$ is a more complex structure (eg graph) the problem is called *structured prediction*

Steps for solving a supervised learning problem:

1. Decide what the input-output pairs are.
2. Decide how to encode inputs and outputs.
   - This defines the input space X and the output space Y.
3. Choose a hypothesis class H (model).
4. Choose an error function (cost function) to define the best hypothesis.
5. Choose an algorithm for searching efficiently through the space of hypotheses (optimizing).

Key aspects of learning problem

- **Set of classifiers H:** modelling
  - Different settings of parameters give different classifiers in the set
- **Learning algorithm / Criterion:** optimizing
- **Generalization**
  - Choice of H (too big ➔ **overfit**, too small ➔ **underfit**)
  - Training data S_n
  - Learning algorithm

Linear Classifier

Linear Classifier

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

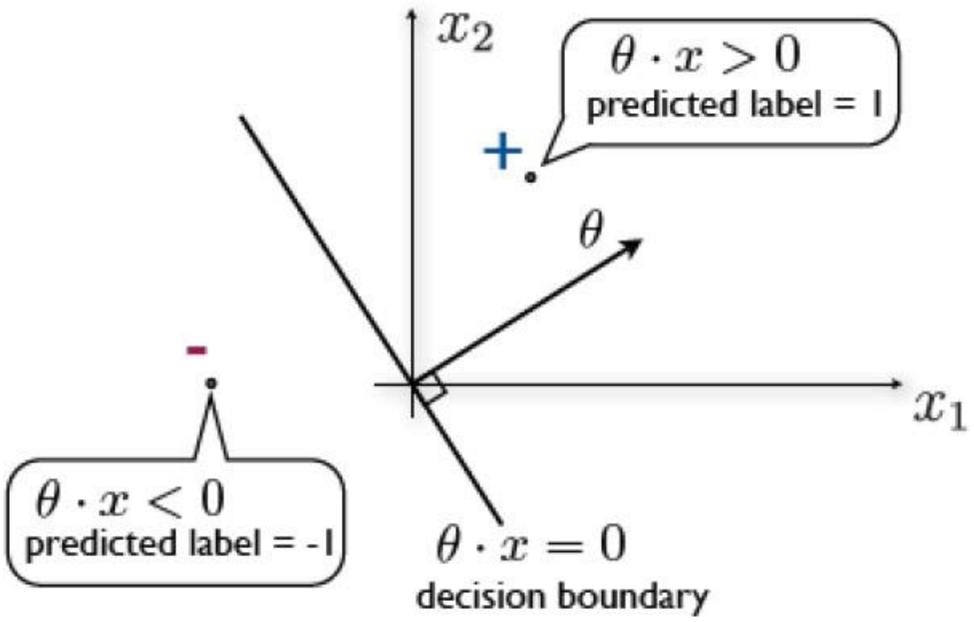
$$h(x;\theta) = \text{sign}(\theta_1 x_1 + \ldots + \theta_d x_d) = \text{sign}(\theta \cdot x) = \begin{cases} +1, & \theta \cdot x \geq 0 \\ -1, & \theta \cdot x < 0 \end{cases}$$

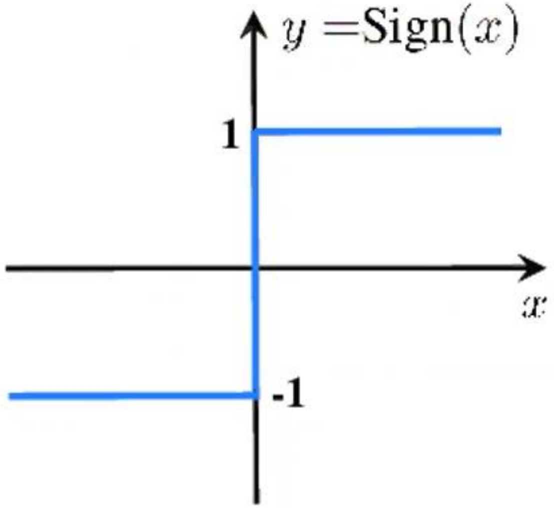
- Where, $\theta \cdot x = \theta^T x$ and $\theta = [\theta_1, \ldots, \theta_d]^T$ is a column vector of real valued parameters or weights.

- 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:

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

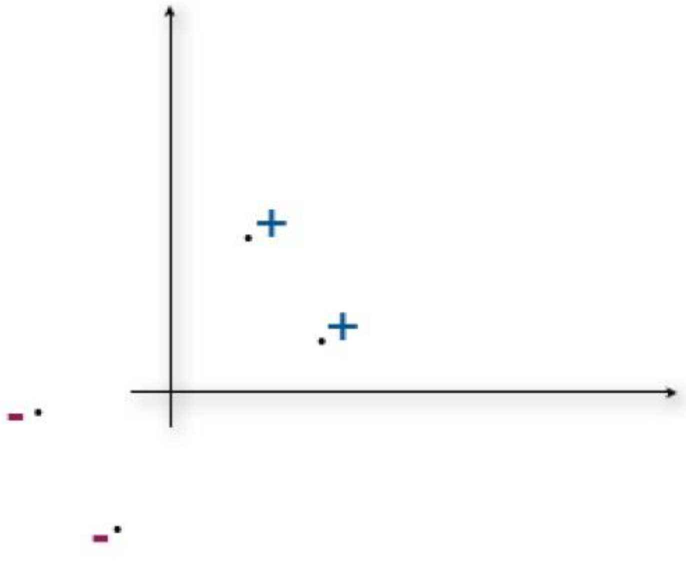
- **Training error:**

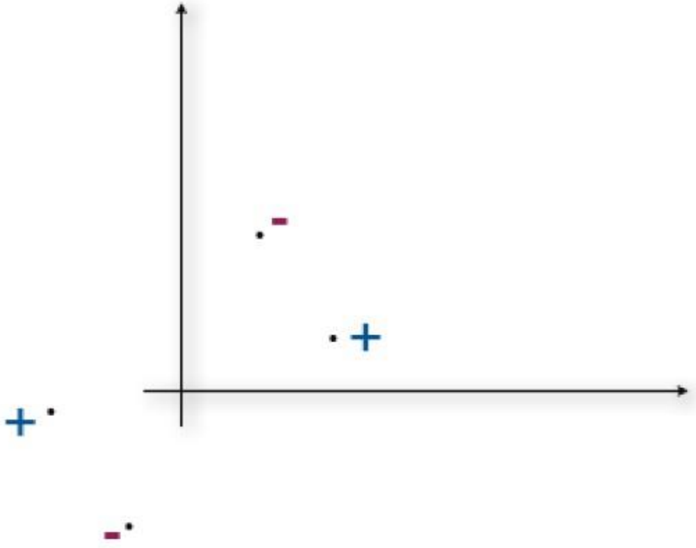
$$\mathcal{E}_n(\theta) = \frac{1}{n}\sum_{t=1}^{n} [\![y^{(t)} \neq h(x^{(t)};\theta)]\!] = \frac{1}{n}\sum_{t=1}^{n} [\![y^{(t)}(\theta \cdot x^{(t)}) \leq 0]\!]$$

- 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 weight so that this function values becomes more positive.
- If the classifier predicted positive but it should have been negative, shift the weights so that the value becomes more 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.

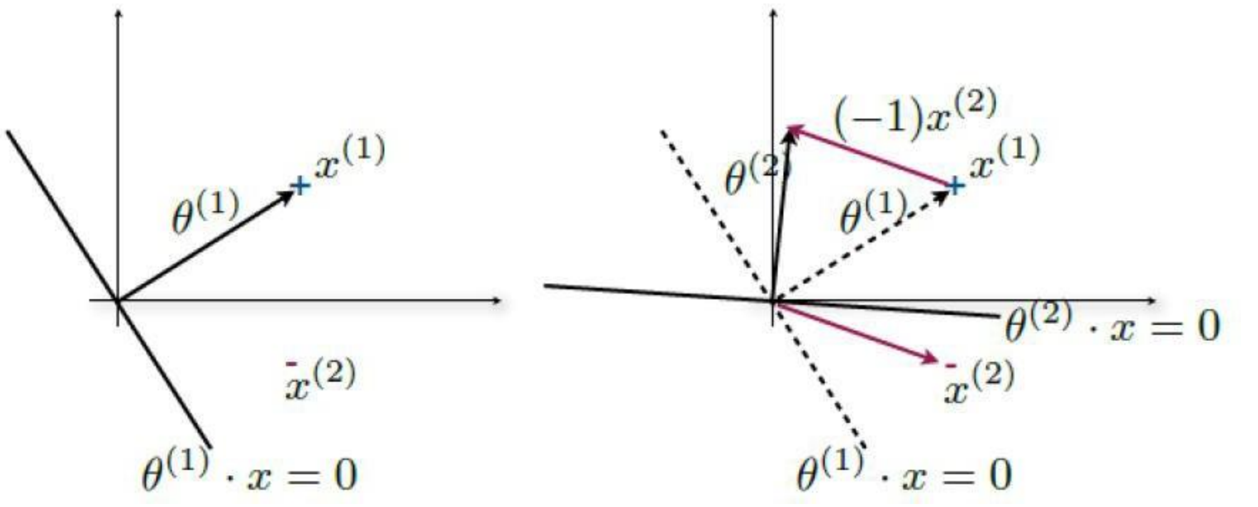
$$\begin{aligned} y^{(t)}(\theta^{(k+1)} \cdot x^{(t)}) &= y^{(t)}(\theta^{(k)} + y^{(t)}x^{(t)}) \cdot x^{(t)} \\ &= y^{(t)}(\theta^{(k)} \cdot x^{(t)}) + (y^{(t)})^2(x^{(t)} \cdot x^{(t)}) \\ &= y^{(t)}(\theta^{(k)} \cdot x^{(t)}) + \left\|x^{(t)}\right\|^2 \end{aligned}$$

Perceptron update rule

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

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

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

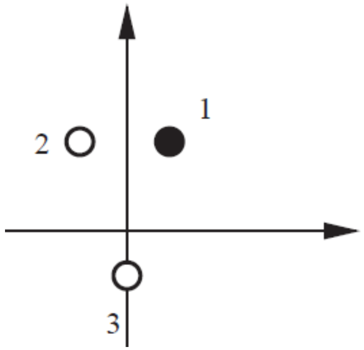


Exercise 1

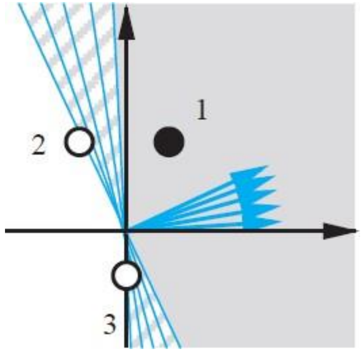
Perceptron update rule

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:**



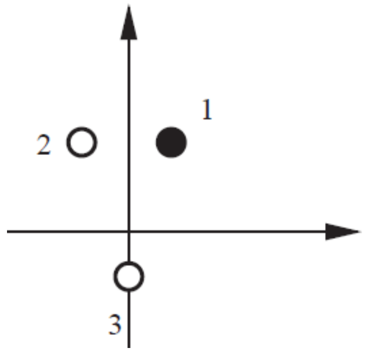
Input data

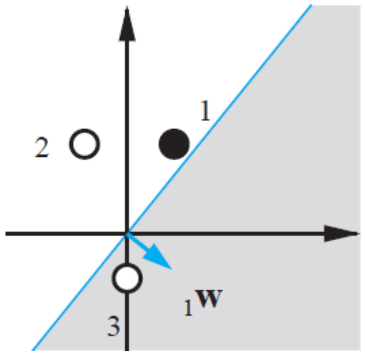


Weight vectors representing allowable decision boundaries

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 $\quad y^{(t)}(\theta^{(k)}.\mathrm{x}^{(t)}) \leq 0 \quad$ then

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

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

➔ **Check if the point is classified correctly:**

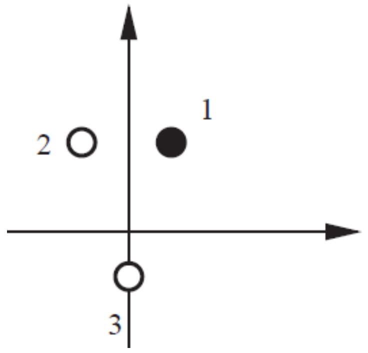
$$y^{(1)}(\theta^{(0)}.\mathrm{x}^{(1)}) = (+1)((1 * 1) + (-0.8 * 2)) = (1 - 1.6) = -0.6 < 0$$

➔ **Update $\theta^{(0)}$:**

$$\theta^{(1)} = \theta^{(0)} + y^{(1)} x^{(1)} = \begin{bmatrix} 1 \\ -0.8 \end{bmatrix} + \begin{bmatrix} 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 2 \\ 1.2 \end{bmatrix}$$

Perceptron update rule

- **Test example:**



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

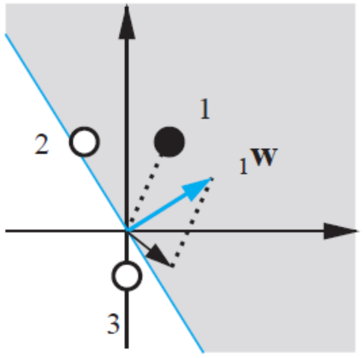
$x^{(2)} = \begin{bmatrix} -1 \\ 2 \end{bmatrix}, y^{(2)} = -1$

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

Perceptron update rule:

if $\quad y^{(t)}(\theta^{(k)}.x^{(t)}) \leq 0 \quad$ then

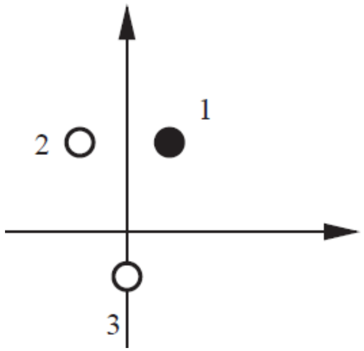
$$\theta^{(k+1)} = \theta^{(k)} + y^{(t)}x^{(t)}$$

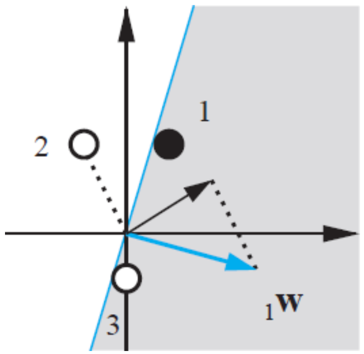


$\theta^{(1)} = \begin{bmatrix} 2 \\ 1.2 \end{bmatrix}$ ➔ **1st Update**

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:

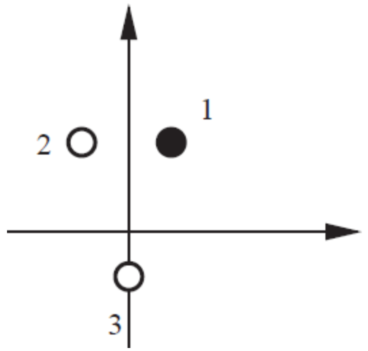
if $\quad y^{(t)}(\theta^{(k)}.x^{(t)}) \leq 0 \quad$ then

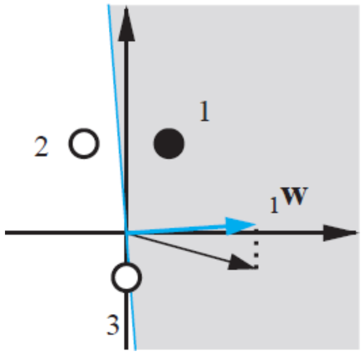
$$\theta^{(k+1)} = \theta^{(k)} + y^{(t)}x^{(t)}$$

$$\theta^{(2)} = \begin{bmatrix} 3 \\ -0.8 \end{bmatrix}$$ ➔ **2nd 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 $\quad y^{(t)}(\theta^{(k)}.\mathrm{x}^{(t)}) \leq 0 \quad$ then

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

$$\theta^{(3)} = \begin{bmatrix} 3 \\ 0.2 \end{bmatrix}$$

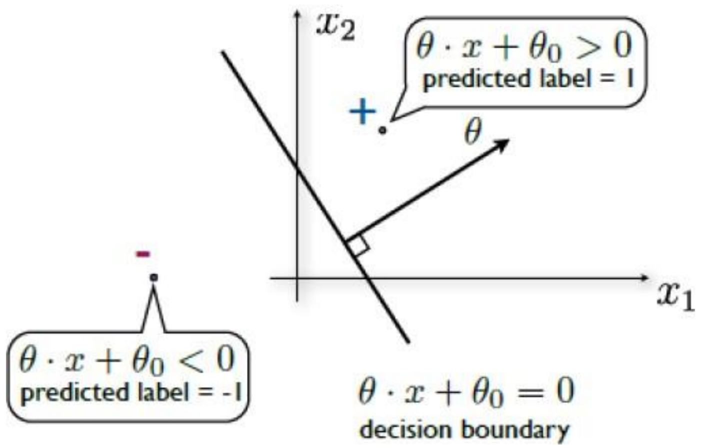
➔ **3rd Update**

Linear Classifier (with offset)

Linear Classifier with offset

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

- 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 $\theta_0 < 0$



Exercise 2

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, θ

Searched “repair”	2.0
Searched “reviews”	8.0
Recent purchase	-15.0
Clicked ads before	5.0
Offset	-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, θ (highlighted weight indicates that the feature value is 1)

Searched “repair”	2.0
Searched “reviews”	8.0
Recent purchase	-15.0
Clicked ads before	5.0
Offset	-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, θ (highlighted weight indicates that the feature value is 1)

Searched “repair”	2.0
Searched “reviews”	8.0
Recent purchase	-15.0
Clicked ads before	5.0
Offset	-9.0

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

Linear Classifier with offset

- Suppose these are the weights, θ (highlighted weight indicates that the feature value is 1)

Searched "repair"	2.0
Searched "reviews"	8.0
Recent purchase	-15.0
Clicked ads before	5.0
Offset	-9.0

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

Linear Classifier with offset

- Suppose these are the weights, θ (highlighted weight indicates that the feature value is 1)

Searched "repair"	2.0
Searched "reviews"	8.0
Recent purchase	-15.0
Clicked ads before	5.0
Offset	-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, θ (highlighted weight indicates that the feature value is 1)

Searched "repair"	2.0
Searched "reviews"	8.0
Recent purchase	-15.0
Clicked ads before	5.0
Offset	-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).

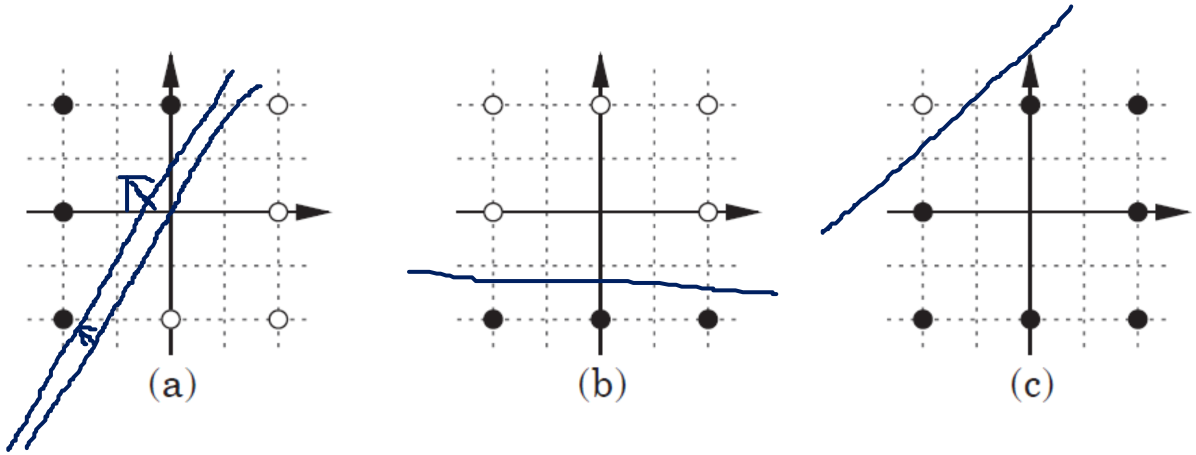
Exercise 3

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?

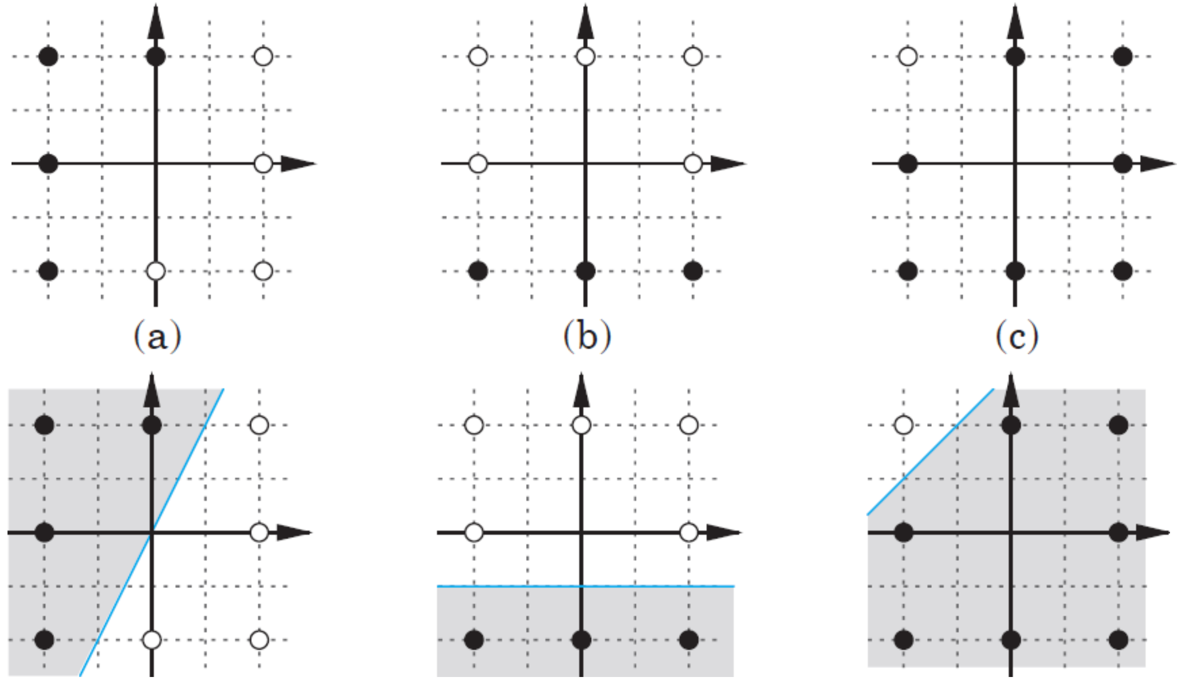
Linear Classifier with offset

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



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

    $$\boxed{\theta_0^{(k+1)} = \theta_0^{(k)} + y^{(t)}}$$

$(1, x_1^t, x_2^t, \ldots, x_d^t)$

$(\theta_0^k, \theta_1^k, \theta_2^k, \ldots, \theta_d^k)$

- Terminate if the **training error** is zero (realizable) or a pre-determined number of iterations are completed (non-realizable).

Perceptron update rule (with offset)

- Imagine all our x^t are added "1" as the first element:
  x^{t*} = (1, x_1^t, x_2^t, … x_d^t)

Meanwhile, all our θ^k are added a constant θ_0^k as the first element:
  θ^{k*} = (θ_0^k, θ_1^k, θ_2^k, … θ_d^k)

- And we have our y^{t*} = 1 (while $x^{t*} \cdot \theta^{k*} \geq 0$)
  = -1 (while $x^{t*} \cdot \theta^{k*} < 0$)

If we have the same update rule: $\theta^{k+1*} = \theta^{k*} + y^{t*} \cdot x^{t*}$, will $y^{t*}(\theta^{k+1*}$ $x^{t*})$ still be bigger than $y^{t*}(\theta^{k*}$ $x^{t*})$?

Intended Learning Outcomes

- Given a set of training examples, find out if they are **linearly separable (with and without offset)**.
- How does **Perceptron algorithm** work and apply to select the best classifier in a **realizable case**.
- What is the **guarantee of the Perceptron algorithm** when the dataset is linearly separable.
- Application of the **linear classifier to real data**.