FIT5201 - Data analysis algorithms

Feedbacks:

1. **Most important: working code, reasonable plot and analysis**
2. **Adding comments to your code**
3. **Making your answer to each question self-contained and identifiable**

Module 3:

Linear Models for Classification

(Part A)

Module Objectives

Learn Basic Concepts of Linear Classification

Learn Discriminative Models

Learn Probabilistic Generative Models

Learn Probabilistic Discriminative Models

Basic Concepts of
Linear Models for Classification

Regression vs. Classification

Regression

- Modelling the relationship between input variables and one or more **continuous target variables**.
- Real-life Example:
  - predicting house prices based on increase in sizes of houses
  - predicting exam results based on hours of study
  - predicting number of movie tickets based on number of tweets.

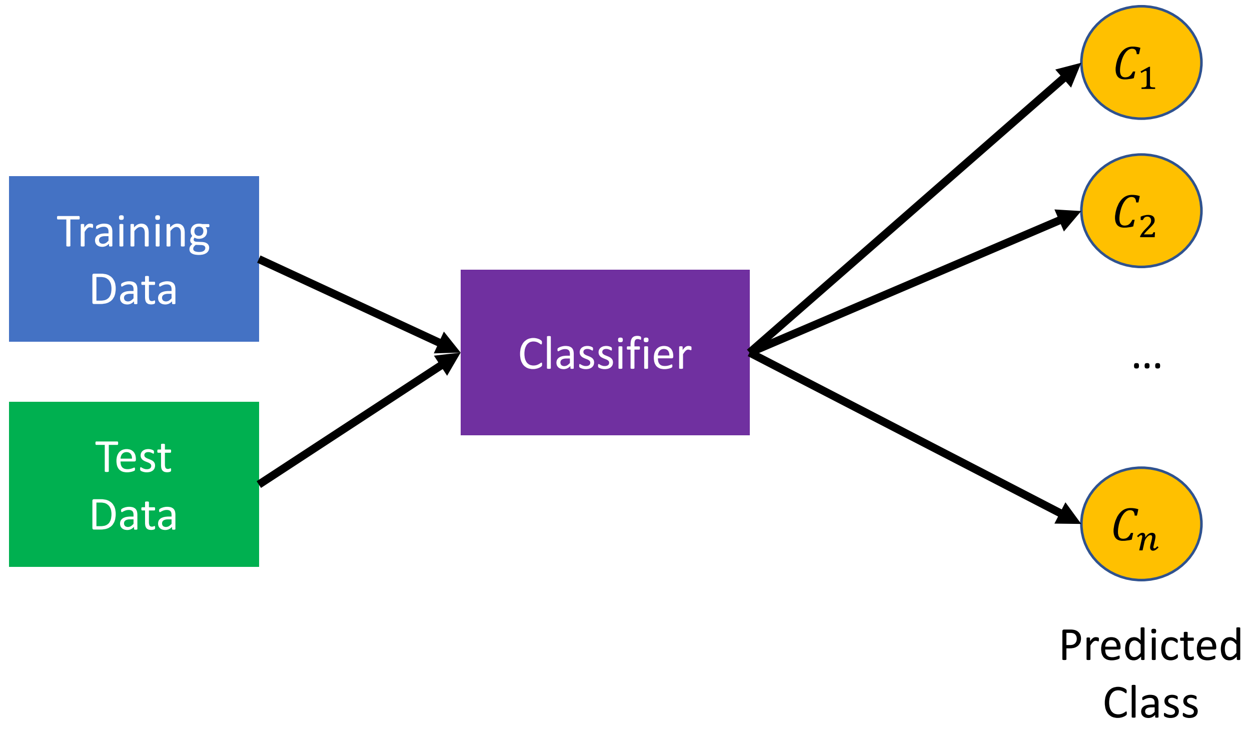
Classification

- Modelling the relationship between input variables and one of k **discrete target variables.**
- Real-life Example:
  - Classify customers into good credit, bad credit, or Grey
  - Predict whether an email is a *Spam* and should be delivered to the Junk folder.
  - Recognise hand-written digit images

Classifier

A Classifier

- An algorithm that implements classification mapping input data to a class (category).



Classifier

A Classifier

- Will use training data to train a classifier to predict the classification of a new unknown data instance

An example of training data

Instance	x_1 (Length)	x_2 (Width)	Class (Y)
1	100	50	TUNA
2	60	20	BASS
3	90	20	TUNA
4	50	30	BASS
…	…	…	…

Simple Linear Classifier

Learn weights:

f(x) = $3x_1 + 2x_2 - 250$

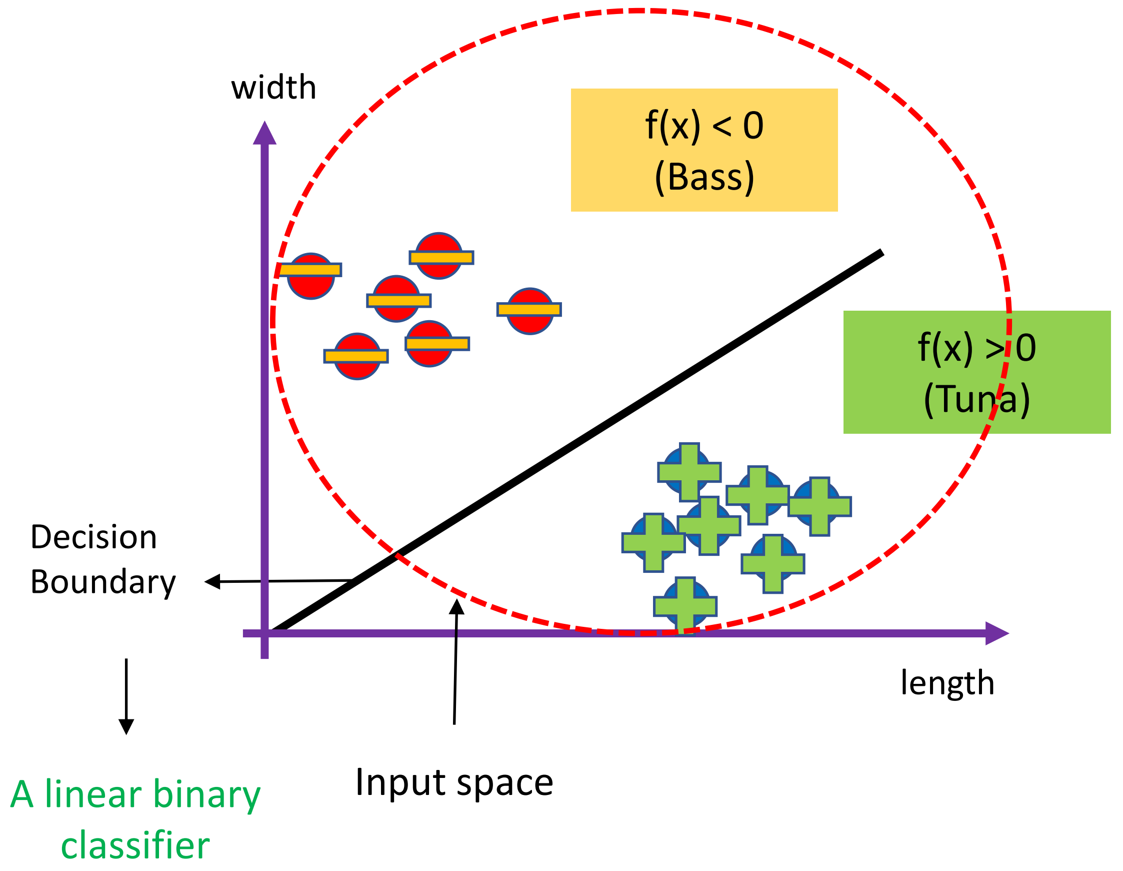
Feature	Weights
x_1	3
x_2	2

Classification:

```
If (f('unknown fish')>0)
        Y= Tuna;
Else
        Y = Bass;
```

Decision Boundary

Suppose only two features had non-zero weights



- Input Space: each input data point is represented by a vector in the input space
- Decision Boundary: the decision regions are separated by the decision boundaries
- Decision Regions: based on the class labels, the input space is divided into decision regions, where each region corresponds to a class label
- Decision Boundary for Linear Models: the decision boundaries are linear functions of the input vector.
  - This module's focus
  - if the input space is D-dimensional, the decision boundary will be a (D-1)-dimensional hyperplane.
- Linear Separability: Datasets whose classes can be separated exactly by linear decision surfaces are said to be linearly separable

Decision Boundary

How to define a hyperplane (i.e. decision boundary)?

- A hyperplane contains all the points in a d-dimensional space satisfying the following:

$$w_0 + w_1 x_1 + w_2 x_2 + \ \ldots + w_d x_d = 0$$

(i.e. $w_0 + \mathbf{w}^{\mathrm{T}}\boldsymbol{x} = 0$): w_0 is called bias or threshold

- Each parameter w_i is thought as as weight on the corresponding input variable.

- The vector containing all the parameters is denoted by $\mathbf{w} = (w_1, \ldots, w_d)$ is the parameter vector (or weight vector)

Linear Classifier

What's the common scenario for linear classification?

Assign each input to only one class.

How to decide whether an input belongs to class Tuna or Bass?

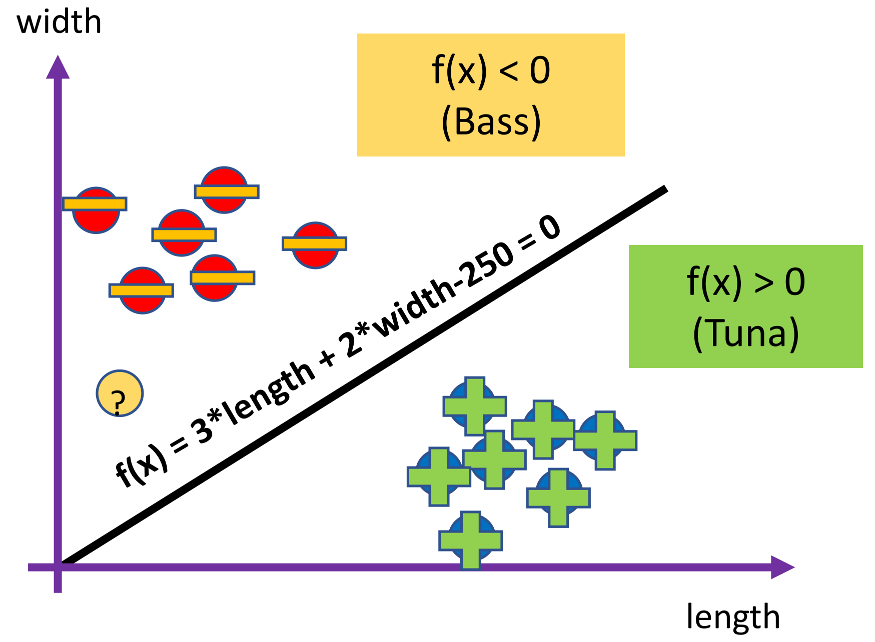
Check on which side of a hyperplane the point is.

Let:

$$f(\mathbf{x}) = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_d x_d = w_0 + \mathbf{w}^{\mathrm{T}}\mathbf{x}$$

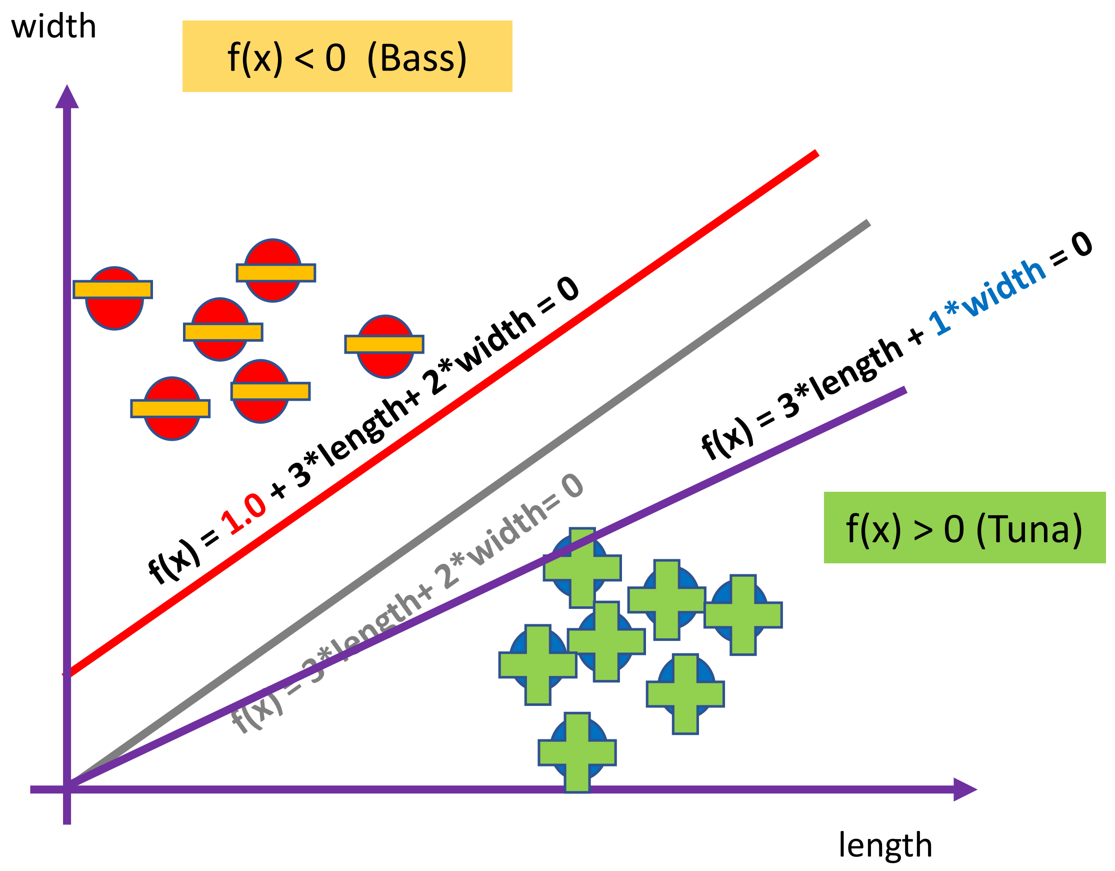
Model:

$$\mathrm{y}(\mathbf{x}) = \mathrm{sign}(f(\mathbf{x})) = \begin{cases} +1 & \text{if } f(\mathbf{x}) \geq 0 \\ -1 & \text{otherwise} \end{cases}$$



$$\mathrm{y}(\mathbf{x}) = \mathrm{sign}(\mathrm{f}(\mathbf{x})) = \begin{cases} +1\ (\text{Tuna}) & \text{if } \mathrm{f}(\mathbf{x}) \geq 0 \\ -1\ (\text{Bass}) & \text{otherwise} \end{cases}$$

Linear Classifier: Effect of Changing Parameters


width
f(x) < 0 (Bass)
f(x) = 1.0 + 3*length+ 2*width = 0
f(x) = 3*length+ 2*width= 0
f(x) = 3*length + 1*width = 0
f(x) > 0 (Tuna)
length

Change Feature Weight:

Input	Parameter	Value
length	w_1	3.0
width	w_2	2 => 1

Add bias:

Input	Parameter	Value
bias	w_0	1.0
length	w_1	3.0

Generalised Linear Models

We can reuse regression learning models for learning classification models

Regression:

- The model prediction $y(\mathbf{x})$ was given by a linear function of the parameter $\mathbf{w}$, where the output is a continuous value:

Linear Regression Model: $\mathrm{y}(\mathbf{x}) = w_0 + \mathbf{w}^{\mathrm{T}}\mathbf{x}$

Classification:

- We **generalise** the linear regression model by applying a non-linear function f:

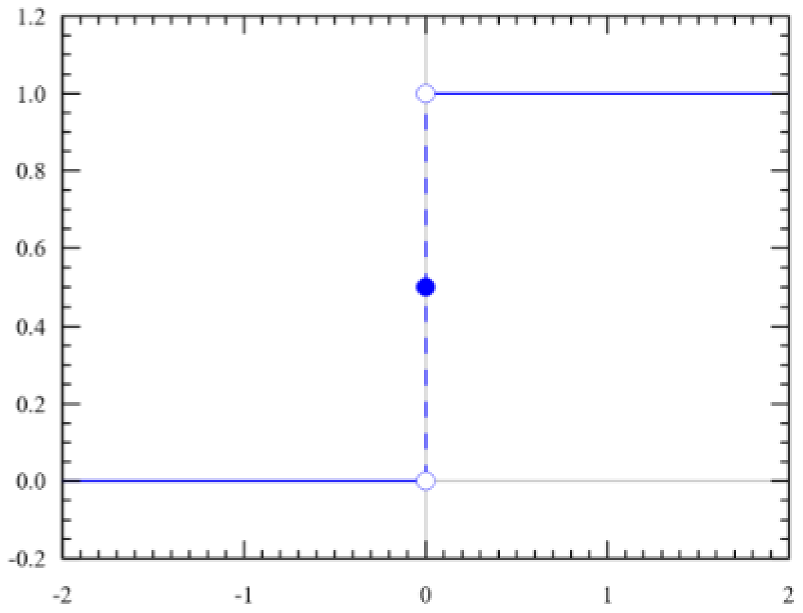
Generalised Linear Model: $\mathrm{y}(\mathbf{x}) = f(w_0 + \mathbf{w}^{\mathrm{T}}\mathbf{x})$

- Function f is known as the activation function,

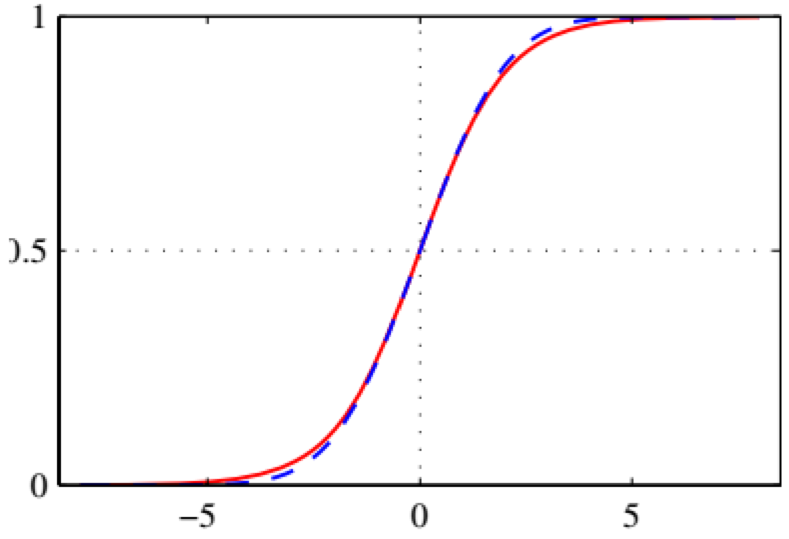
Ex: step function: $f(a) = \begin{cases} 1 & \text{if } a \geq 0 \\ 0 & \text{otherwise} \end{cases}$ (Perceptron, Linear Separability),

logistic function: $f(a) = \frac{1}{1+\exp^{(-a)}}$ (Logistic Regression, probabilistic discriminative model)

Generalised Linear Models



Plot of step function



Plot of logistic sigmoid function

Discriminative vs. Generative Learning Models

Discriminative Models

- Assign the input x to a specific class C_k directly

Probabilistic Discriminative Models

- Produce probability of belonging to each class
  $P_w(\mathcal{C}_k|\mathbf{x})$

Discriminative vs. Generative Learning Models

Discriminative Models

- Direct modelling of classifier

Generative Models

- Indirect modelling of classifier
- Can be used to generate input variables given class labels
- More parameters to learn
- More complexity
- overfitting

$$P(\mathcal{C}_k|\mathbf{x}) \propto \underbrace{P_{\boldsymbol{w}}(\mathcal{C}_k)}_{\text{Class Prior}} \times \underbrace{P_{\boldsymbol{w}}(\mathbf{x}|\mathcal{C}_k)}_{\text{Class Conditional Density}}$$

Discriminative Models

Two Class Discriminant Function

What is a discriminant function?

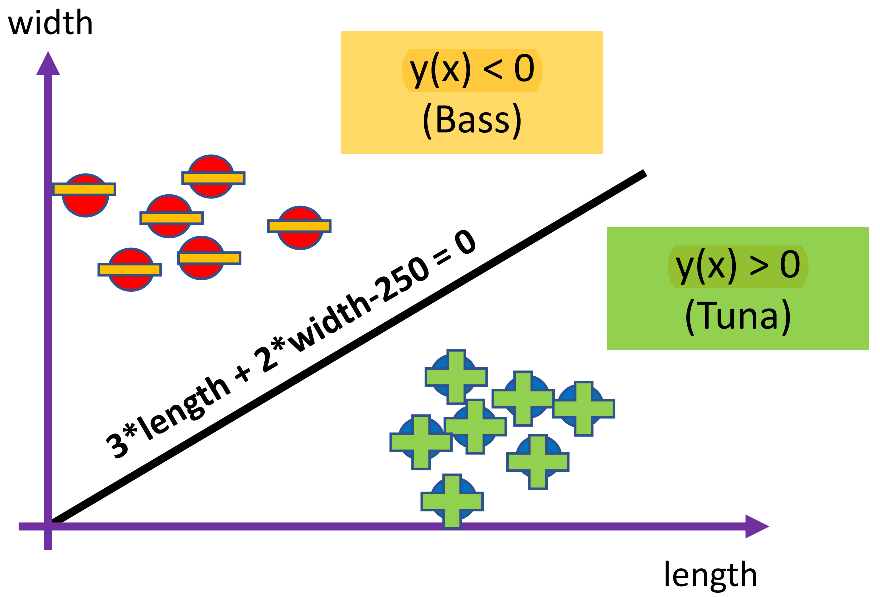
- A function that takes an input vector $\mathbf{x}$ and assign it to one of classes.

Two-class discriminant function

- Linear discriminant: $y(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} + w_0$
- A function that takes an input vector $\mathbf{x}$ and assign it to class C_1 if $y(\mathbf{x}) \geq 0$ and to class C_2 otherwise.

Classification Model:

$$\text{class label} = \begin{cases} \text{Tuna} & \text{if } y(\mathbf{x}) \geq 0 \\ \text{Bass otherwise} \end{cases}$$

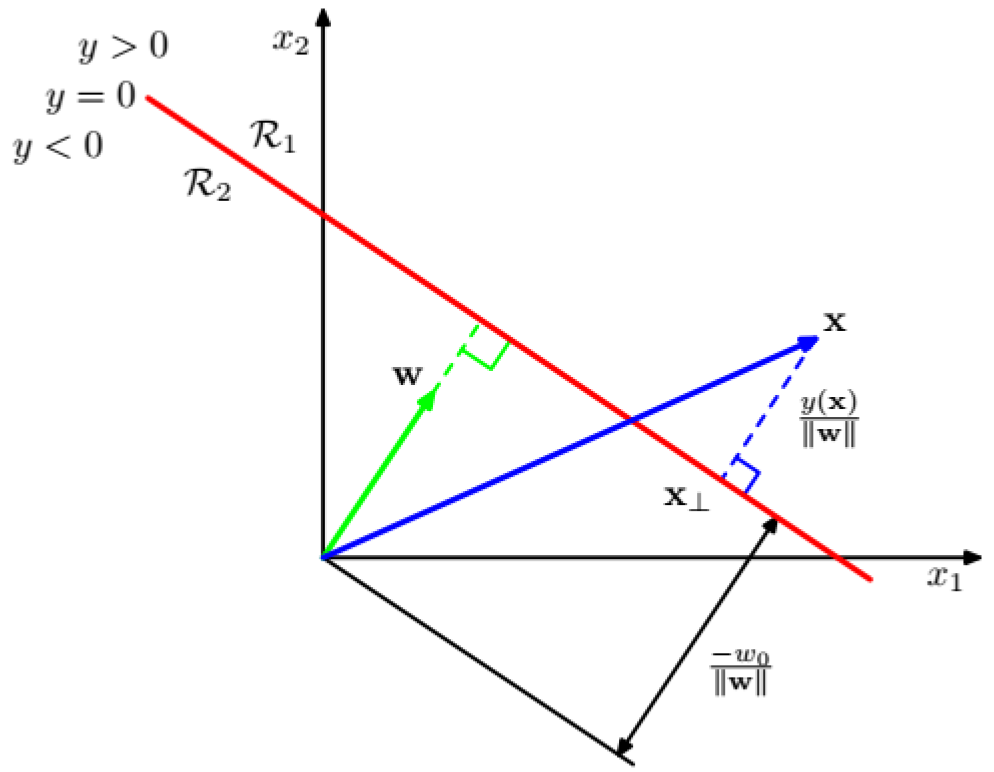


Two Class Discriminant Function

Decision Boundary

- Linear: $\boldsymbol{w}^T \boldsymbol{x} + w_0 = 0$

Geometry Characteristics

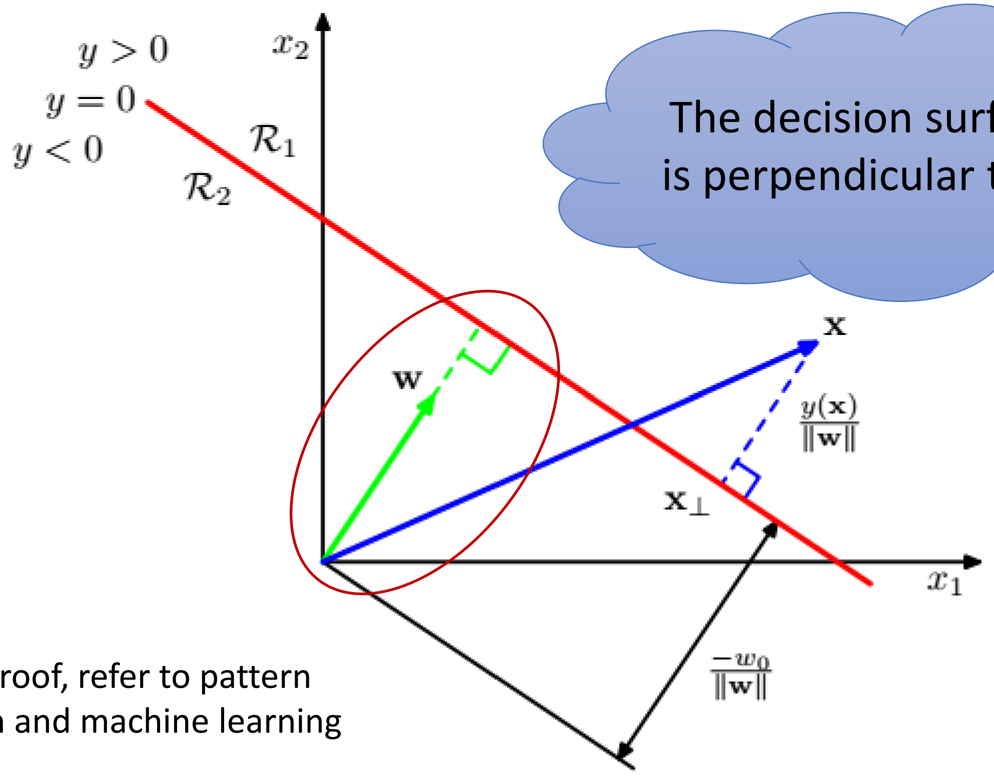


Two Class Discriminant Function

Decision Boundary

- Linear: $\boldsymbol{w}^T\boldsymbol{x} + w_0 = 0$

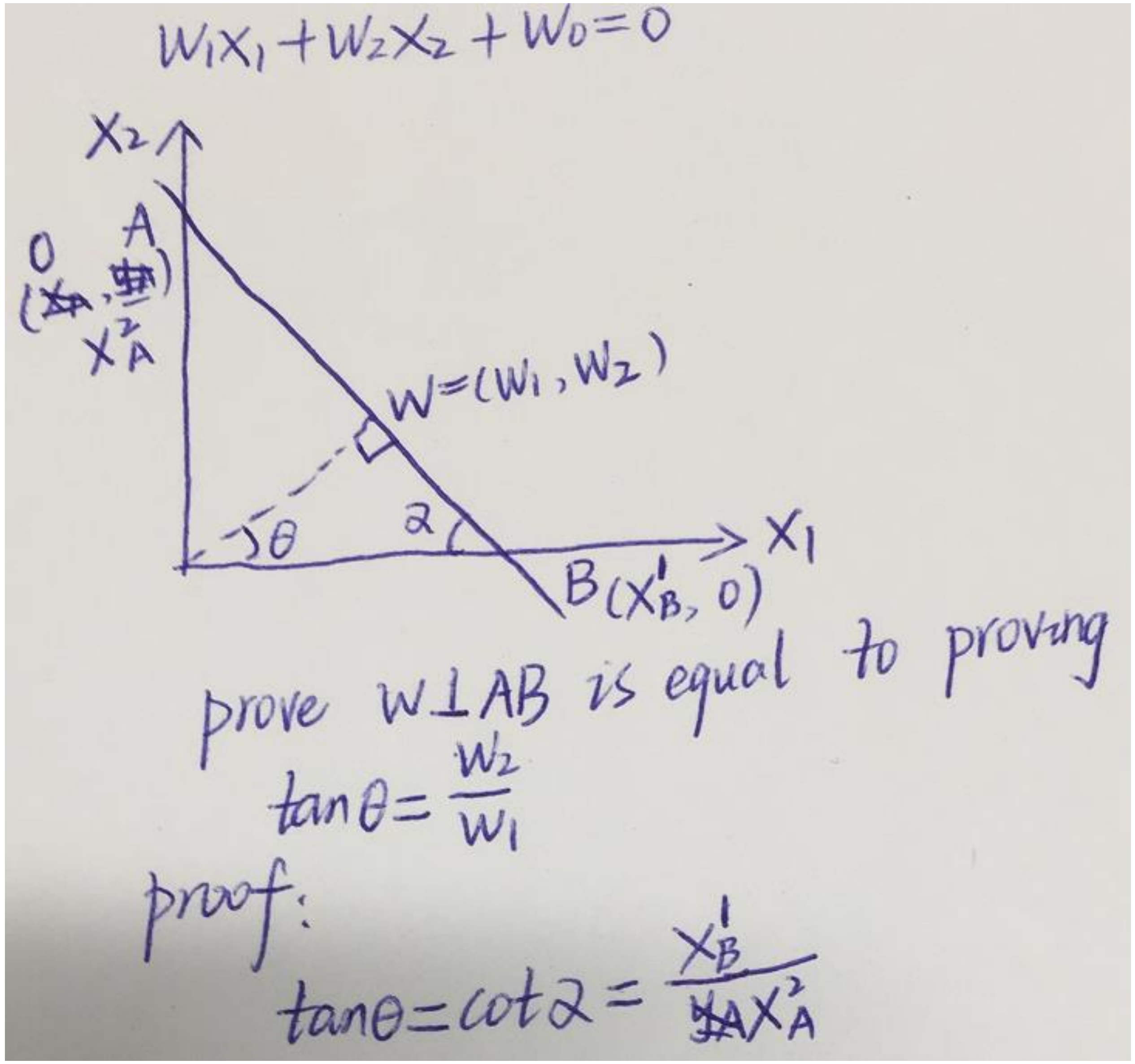
Geometry Characteristics



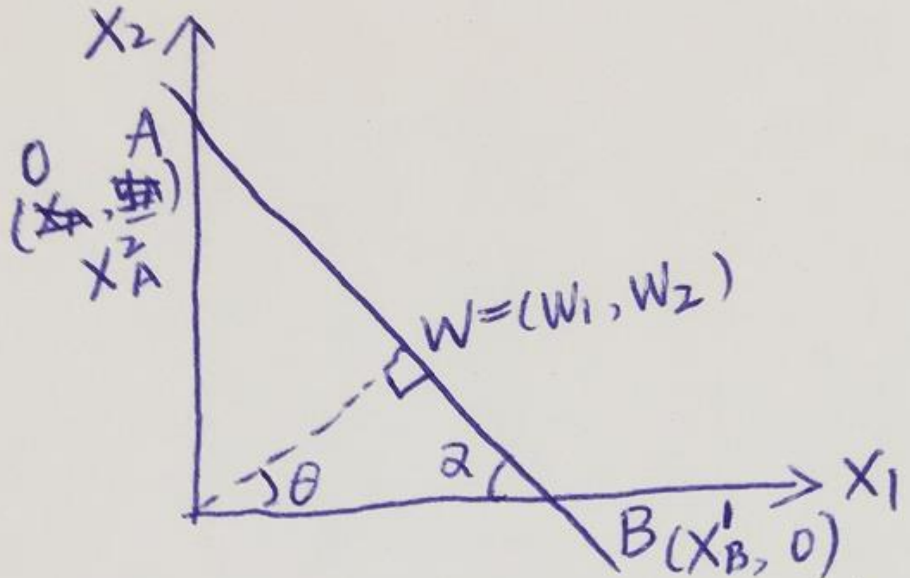
The decision surface is perpendicular to $\boldsymbol{w}$

For more proof, refer to pattern recognition and machine learning

Two Class Discriminant Function



$w_1x_1 + w_2x_2 + w_0 = 0$

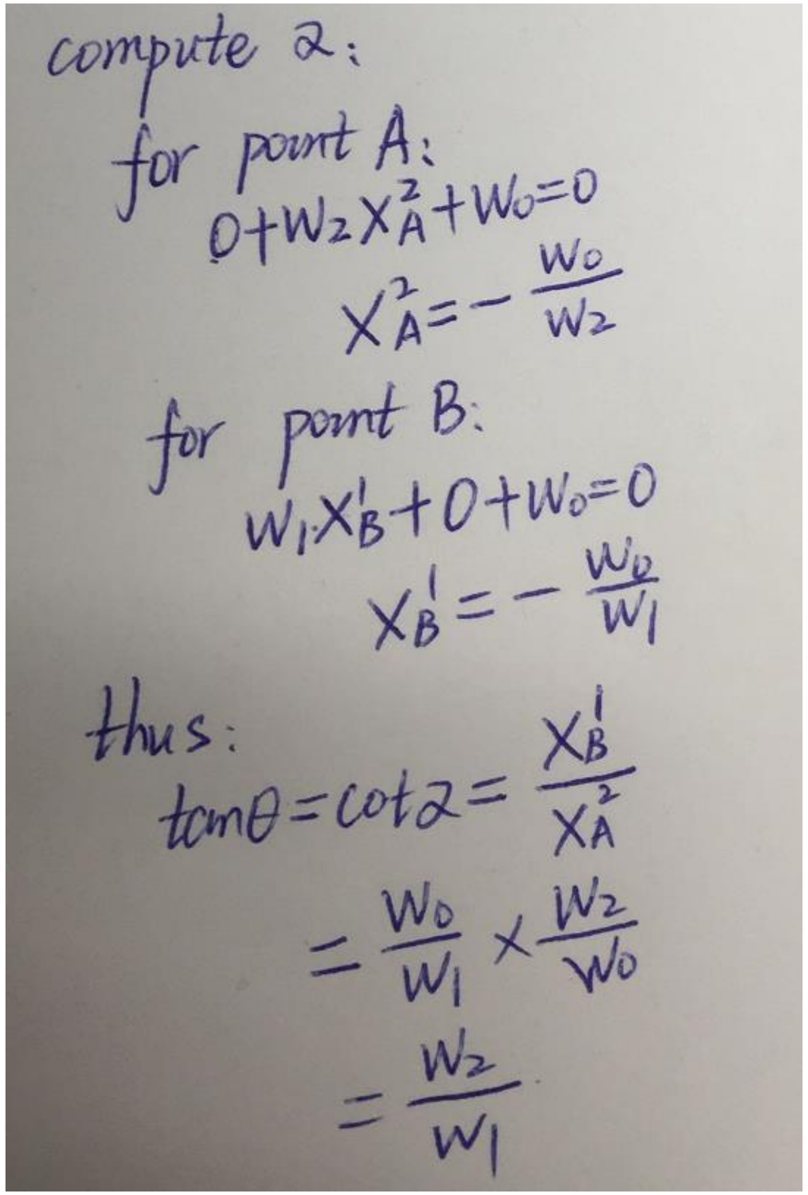


prove $w \perp AB$ is equal to proving

$$\tan\theta = \frac{w_2}{w_1}$$

proof:

$$\tan\theta = \cot\alpha = \frac{x_B^1}{x_A^2}$$



compute α:

for point A:

$$0 + w_2 x_A^2 + w_0 = 0$$

$$x_A^2 = -\frac{w_0}{w_2}$$

for point B:

$$w_1 x_B^1 + 0 + w_0 = 0$$

$$x_B^1 = -\frac{w_0}{w_1}$$

thus:

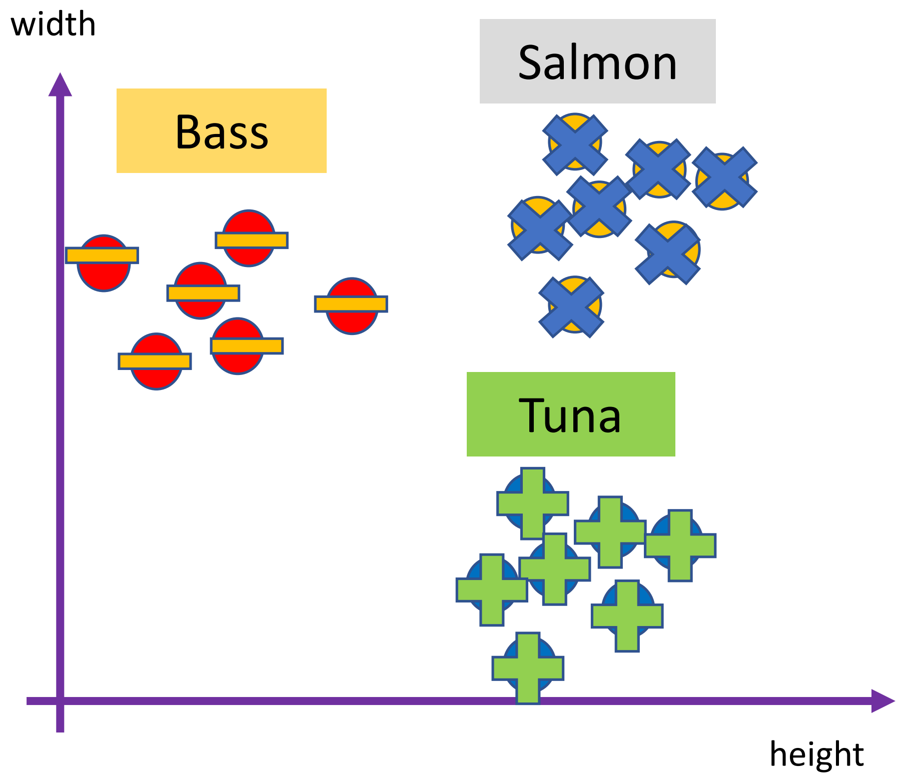
$$\tan\theta = \cot\alpha = \frac{x_B^1}{x_A^2}$$

$$= \frac{w_0}{w_1} \times \frac{w_2}{w_0}$$

$$= \frac{w_2}{w_1}$$

Generalisation to Multiclass Problems

How can we use linear classifiers in general to classify inputs when there are $k > 2$ classes?

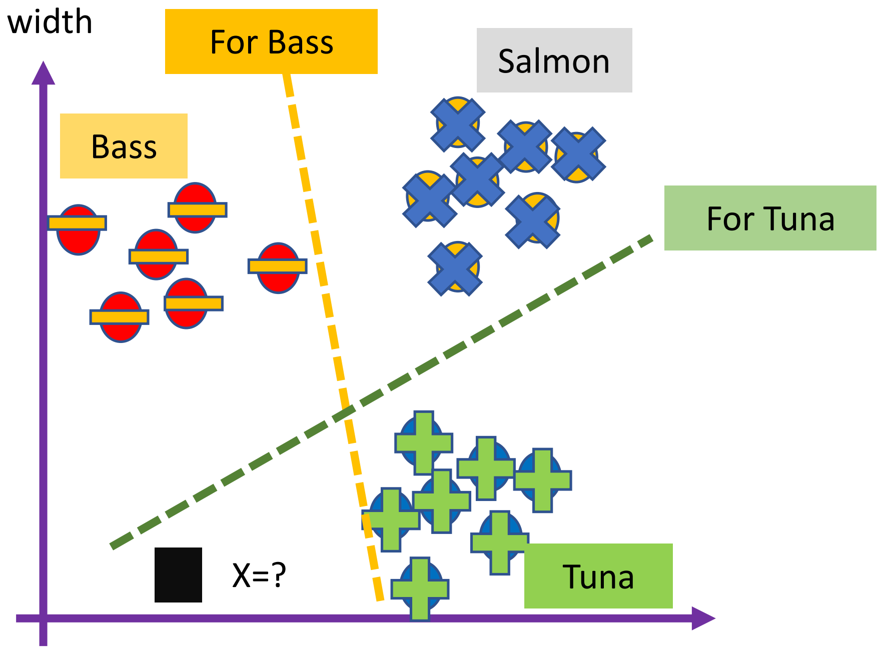


Generalisation to Multiclass Problems

One-versus-the-rest classifier

Idea:

- Transfer multi-class classification into binary classification problem.
- Use $k-1$ discriminant classifiers, each separating one class C_k from the rest
- Problem: Ambiguous regions: X

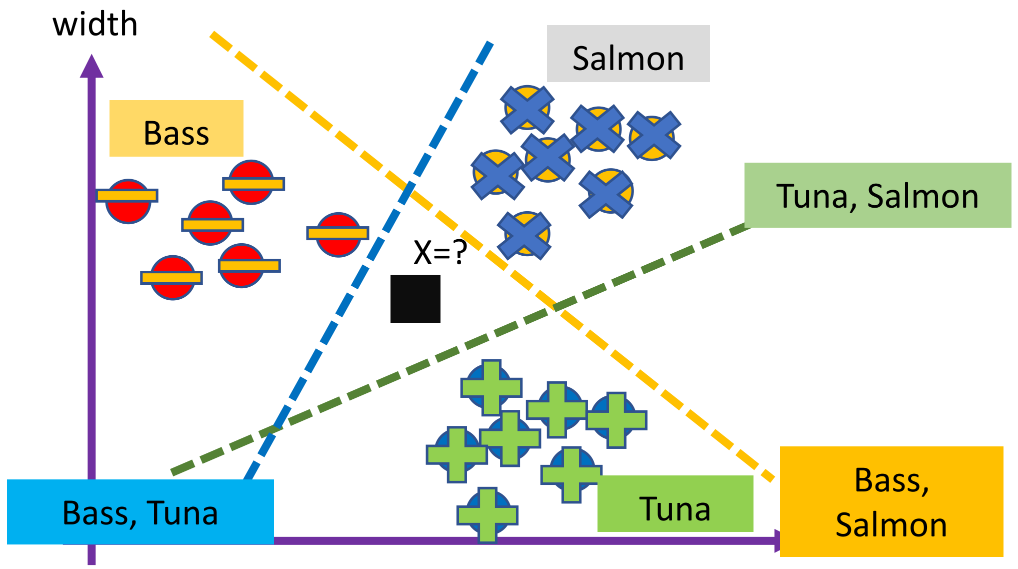


Generalisation to Multiclass Problems

One-versus-one classifier

Idea:

- Build decision surfaces for each pair of classes
- Use $k(k-1)/2$ discriminant functions, each for every possible pair of classes
- Each point is classified by majority vote
- Problem: Ambiguous regions: X



Generalisation to Multiclass Problems

Use K discriminant functions $y_k(\mathbf{x})$

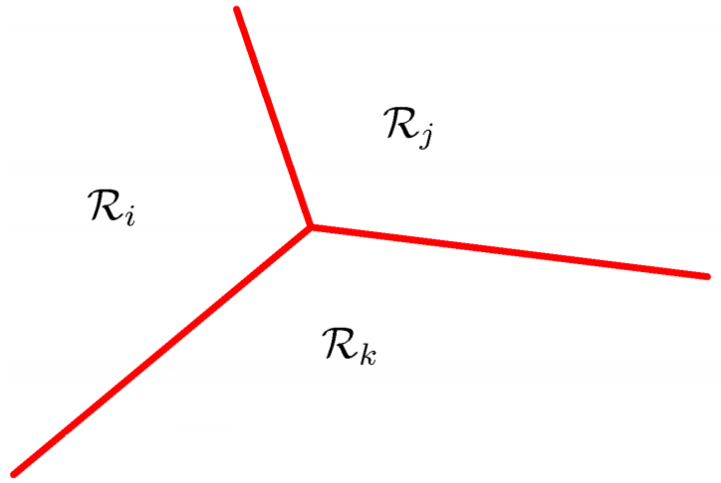
Idea:

- Use the magnitude of $y_k(\mathbf{x})$, not just the sign
- For each class k, we consider a linear function of the form: $y_k(\mathbf{x}) = \mathbf{w}_k^{\mathrm{T}}\mathbf{x} + w_{k0}$
- We then assign a point $\mathbf{x}$ to a class C_k if $y_k(\mathbf{x}) > y_j(\mathbf{x})$ for all $j \neq k$
- That is, $y = \underset{k}{\operatorname{argmax}}\, y_k(\mathbf{x})$

Generalisation to Multiclass Problems

Decision Boundary

- Singly connected and linear
- Without any ambiguous region
- For the boundary between class k and j: $y_k(\mathbf{x}) = y_j(\mathbf{x}) \Rightarrow (\boldsymbol{w_k} - \boldsymbol{w_j})^{\boldsymbol{T}}\mathbf{x} + (w_{k0} - w_{j0}) = 0$



The Perceptron Algorithm

Perceptron

Two-class Perceptron

- The simplest form of a neural network that can be used as a linear classifier
- A generalised linear model in which the input vector $\mathbf{x}$ is transformed to a nonlinear transformation providing a new representation of the input**:**

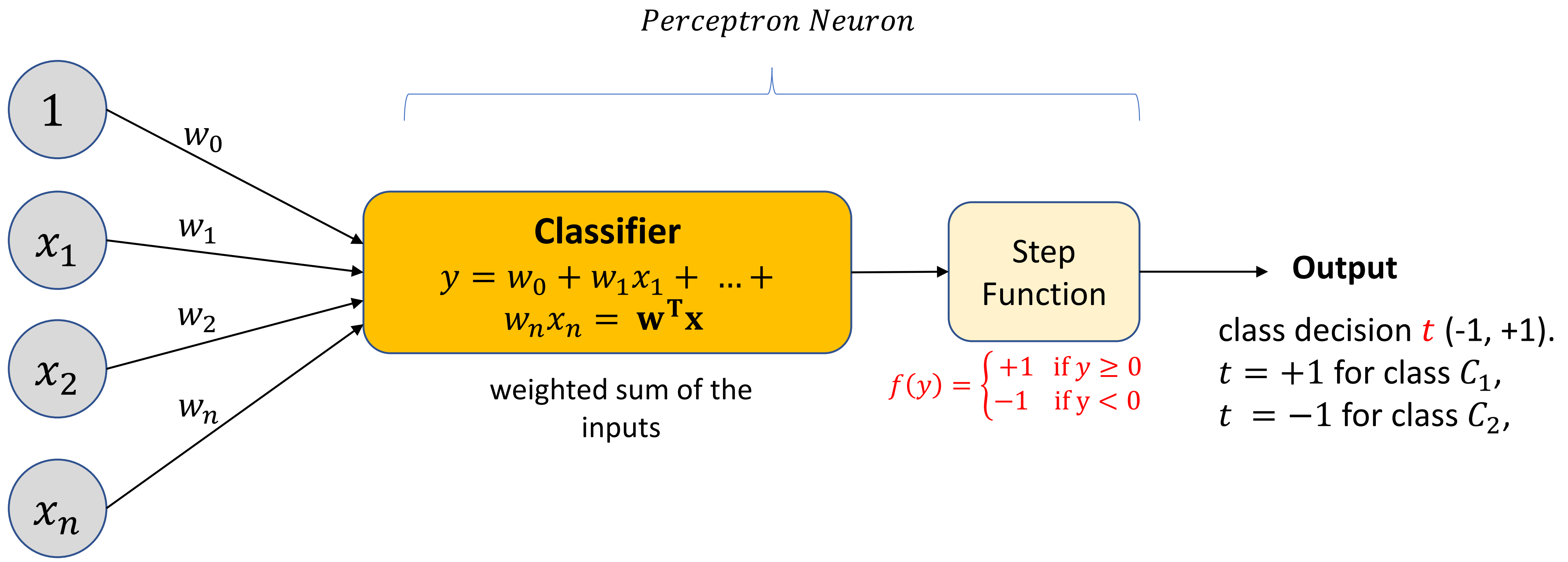
$$y(\mathbf{x}) = f\left(\mathbf{w}^{\mathrm{T}} \cdot \boldsymbol{\phi}(\mathbf{x})\right)$$

where the nonlinear activation function $f(\cdot)$ is given by a modified step function:

$$\mathrm{f}(a) = \begin{cases} +1 & \text{if } a \geq 0 \\ -1 & \text{if a} < 0 \end{cases}$$

- The parameters $\mathbf{w}$ are sometimes called weights (real-valued constants - positive or negative)

Perceptron Classifier


$Perceptron\ Neuron$
1
x_1
x_2
x_n
w_0
w_1
w_2
w_n
Classifier
$y = w_0 + w_1 x_1 + \ldots + w_n x_n = \mathbf{w}^{\mathbf{T}}\mathbf{x}$
weighted sum of the inputs
Step Function
$f(y) = \begin{cases} +1 & \text{if } y \geq 0 \\ -1 & \text{if } y < 0 \end{cases}$
Output
class decision t (-1, +1).
$t = +1$ for class C_1,
$t = -1$ for class C_2,

- The neuron receives multiple inputs and process them to produce an output
- The perceptron calculates 2 quantities: (1) weighted sum of the input features and (2) this sum is then thresholded by the *f* function.

Perceptron Model

Perceptron Model

- Input: $\mathbf{x}$, Basis function $\phi(\mathbf{x})$
- Parameter: $\mathbf{w}$
- Decision Boundary: $\mathbf{w}^{\mathrm{T}}\mathbf{x} = 0$ (for the presentation simplicity, we assume $\phi(\mathbf{x}) = \mathbf{x}$)
- Model: $y = \begin{cases} +1 & \text{if } \mathbf{w}^{\mathrm{T}}\mathbf{x} \geq 0 \\ -1 & \text{Otherwise} \end{cases}$

Perceptron Learning Objective

What is objective of Perceptron Learning?

- Finding the decision surface (hyperplanes) that minimizes the number of misclassified training data points.
- Determine the weights (parameters) $\mathbf{w}$ to minimise an error function.

Error Function in Perceptron Learning

- Intuition: *minimise the number of misclassified training data points.*
- misclassified conditions: $[\boldsymbol{w} \cdot \phi(\boldsymbol{x}_n) t_n < 0] \vee [t_n = -1 \wedge \boldsymbol{w} \cdot \phi(\boldsymbol{x}_n) = 0]$
- Definition of an error function of Perceptron: $y(\mathbf{x}) = f\left(\mathbf{w}^{\mathrm{T}} \cdot \boldsymbol{\phi}(\mathbf{x})\right)$:

$E(\mathbf{w}) = -\sum_{n \in M} \mathbf{w} \cdot \boldsymbol{\phi}(\mathbf{x}_n) t_n$, where M is the set of *misclassified* examples, and t_n is the true label of $\mathbf{x}_n$

Perceptron Learning Algorithm

Initialise $\mathbf{w}$ with random values

Repeat until all data points are classified correctly or the error $< \epsilon$:

- For each training point $\mathbf{x}$, perform the following
  - Compute the activation function

$$y = \begin{cases} +1 & \text{if } \mathbf{w}^{\mathrm{T}}\mathbf{x} \geq 0 \\ -1 & \text{Otherwise} \end{cases}$$

  - If $y = t$, don't change the weights
  - Else (i.e. $y \neq t$) , update the weights:

$$\mathbf{w} \leftarrow \mathbf{w} + \eta t_n \phi(\boldsymbol{x}_n)$$

(η: learning rate)

$$E(\mathbf{w}) = -\sum_{n\in M} \mathbf{w} \cdot \boldsymbol{\phi}(\mathbf{x}_n) t_n$$

$$\nabla E(\mathbf{w}) = -\boldsymbol{\phi}(\mathbf{x}_n) t_n$$

Note: As the parameters change, the data points that were correctly modified might change to being misclassified

Perceptron Learning Algorithm

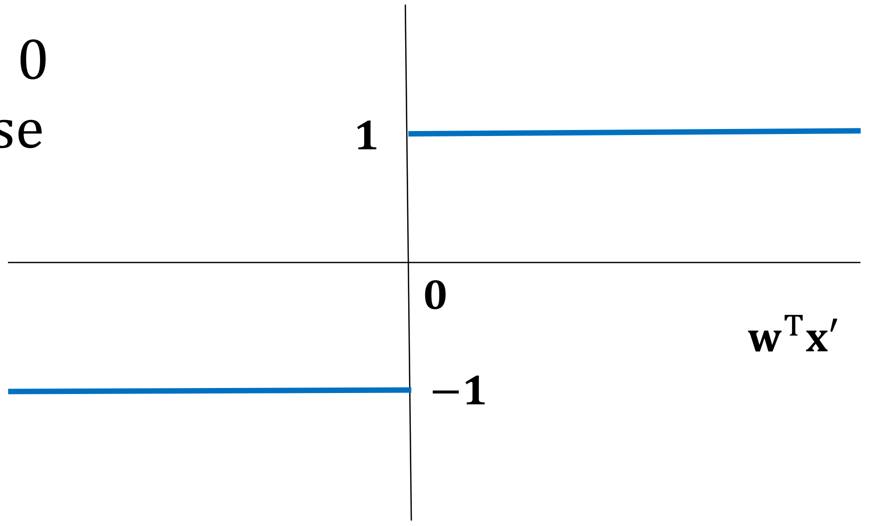
Testing Phase:

Assume that we have already obtained $\mathbf{w}$.

Then Perceptron can classify a new input $\mathbf{x}'$:

- Calculate

$$y = \begin{cases} +1 & \text{if } \mathbf{w}^{\mathrm{T}}\mathbf{x}' \geq 0 \\ -1 & \text{Otherwise} \end{cases}$$



Perceptron Learning Example

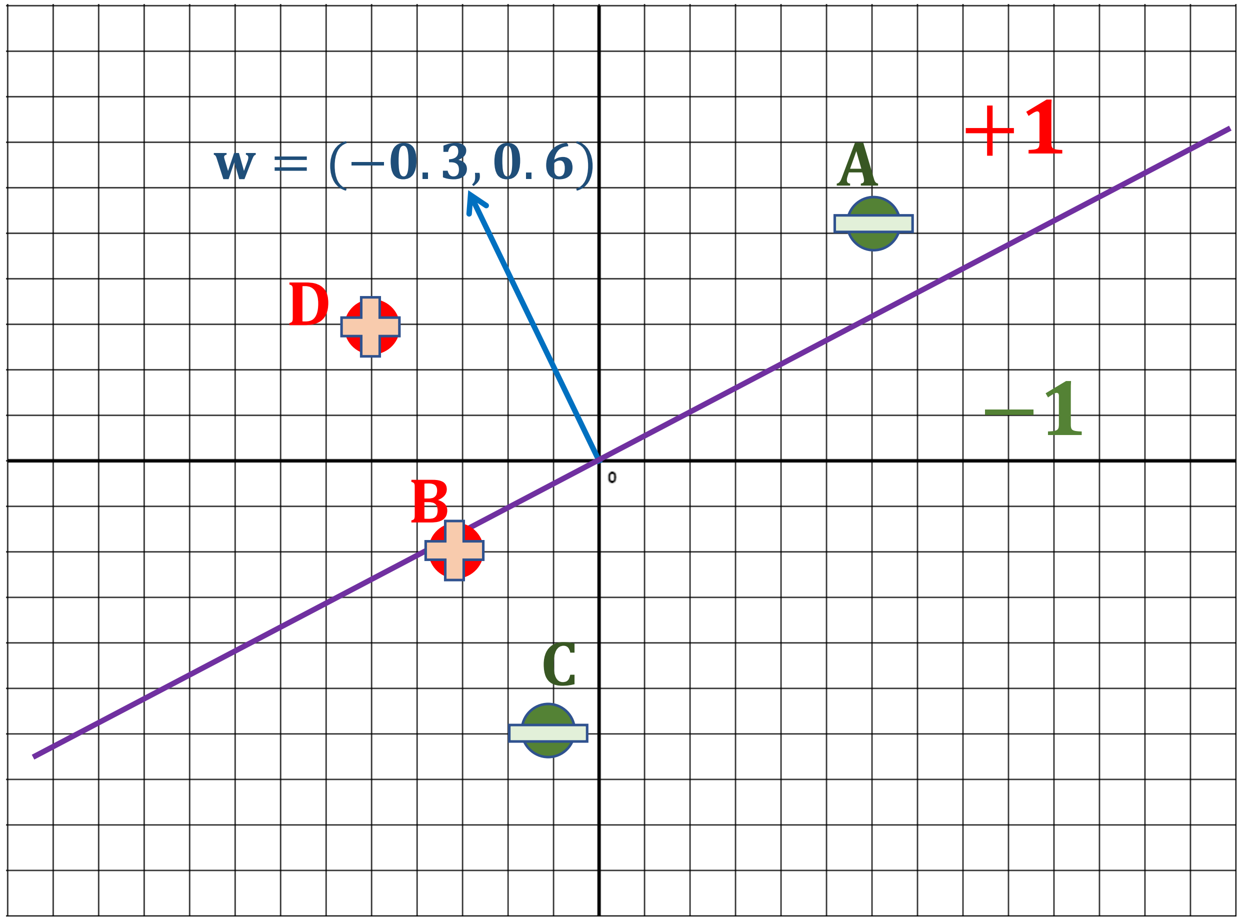
Training data

Item	x_1	x_2	Class
A	0.6	0.5	-1
B	-0.3	-0.2	1
C	-0.1	-0.6	-1
D	-0.5	0.3	1

For input A

- $\mathbf{w}^{\mathrm{T}}\mathbf{x} = (-0.3 * 0.6 + 0.6 * 0.5) = 0.12 \ \geq 0$
- $y = 1$, but $t = -1$
- Update the weights
  - $w_1 = w_1 + tx_1 = -0.3 - 0.6 = -0.9$
  - $w_2 = w_2 + tx_2 = 0.6 - 0.5 = 0.1$

Learning rate: $\eta = 1$, initial $\mathbf{w} = (-0.3, 0.6)$

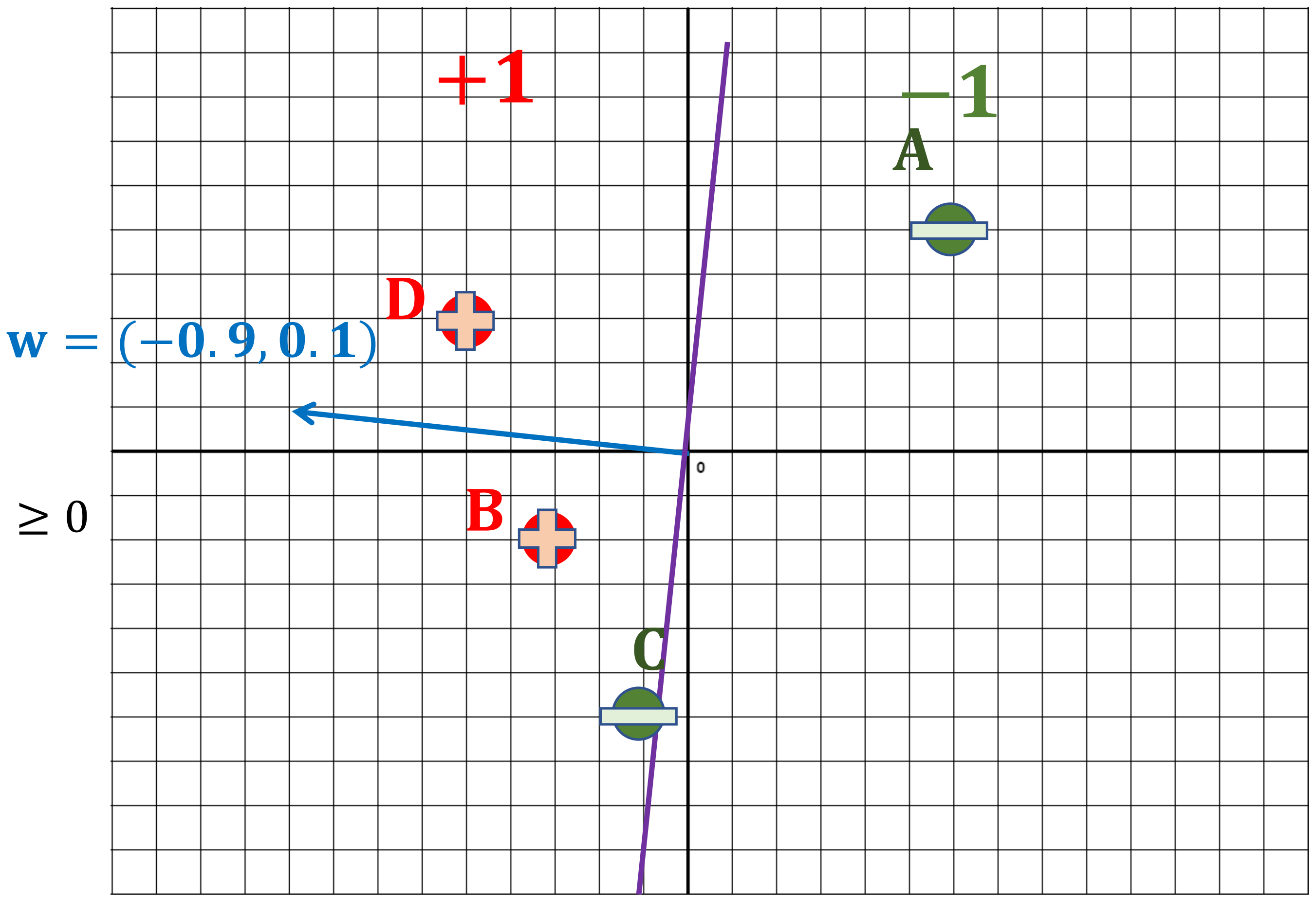


Perceptron Learning Example

Training data

Item	x_1	x_2	Class
A	0.6	0.5	-1
B	**-0.3**	**-0.2**	**1**
C	-0.1	-0.6	-1
D	-0.5	0.3	1

Learning rate: $\eta = 1$, $\mathbf{w} = (-0.9, 0.1)$



For input B

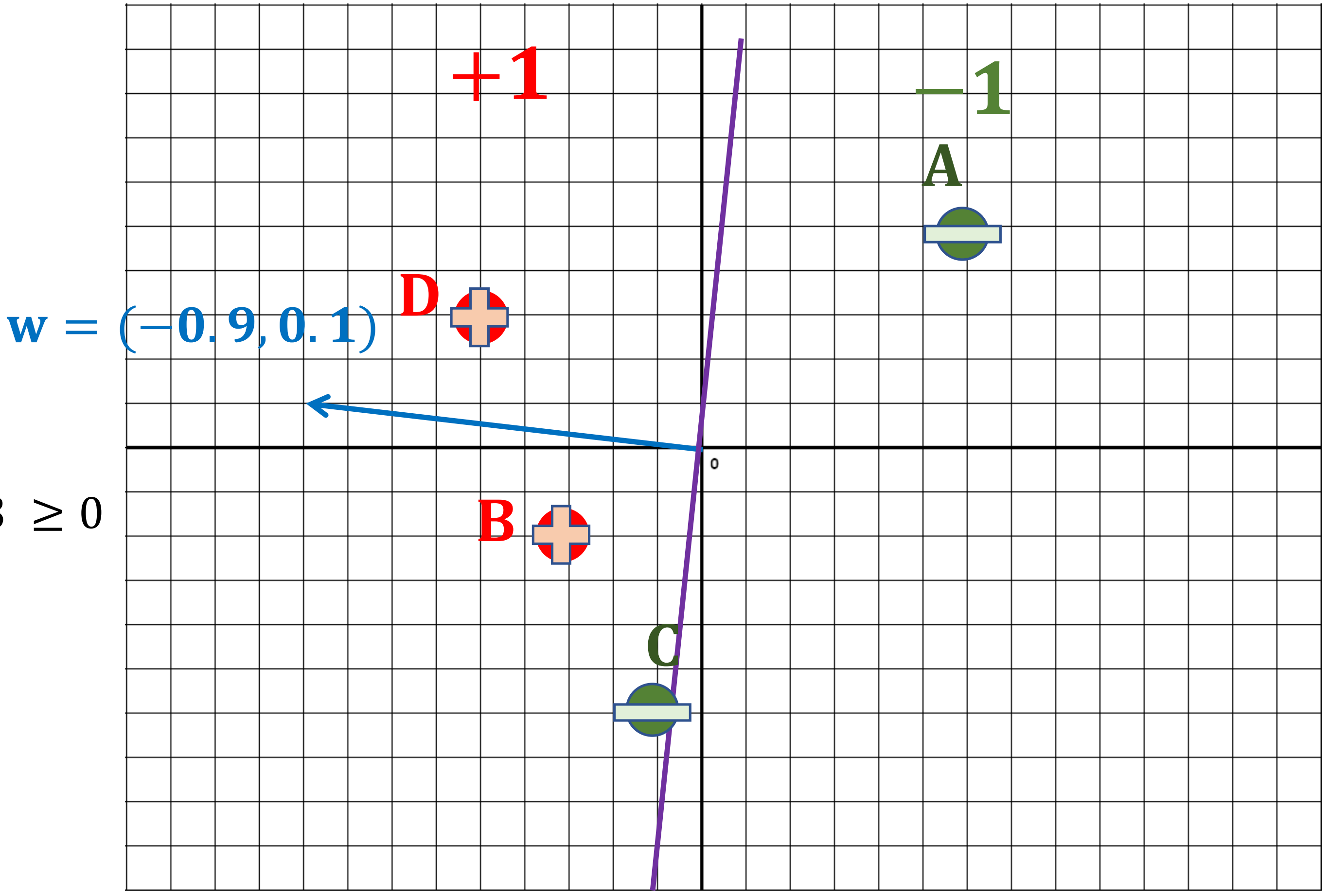
- $\mathbf{w}^{\mathrm{T}}\mathbf{x} = (-0.9 * -0.3 + 0.1 * -0.2) = 0.25 \geq 0$
- $y = 1$, and $t = 1$
- Don't update the weights

Perceptron Learning Example

Training data

Item	x_1	x_2	Class
A	0.6	0.5	-1
B	-0.3	-0.2	1
C	-0.1	-0.6	-1
D	-0.5	0.3	1

Learning rate: $\eta = 1$, $\mathbf{w} = (-0.9, 0.1)$



For input C

- $\mathbf{w}^{\mathrm{T}}\mathbf{x} = (-0.9 * -0.1 + 0.1 * -0.6) = 0.03 \geq 0$
- $y = 1$, and $t = -1$
- Update the weights
  - $w_1 = w_1 + tx_1 = -0.9 + 0.1 = -0.8$
  - $w_2 = w_2 + tx_2 = 0.1 + 0.6 = 0.7$

Perceptron Learning Example

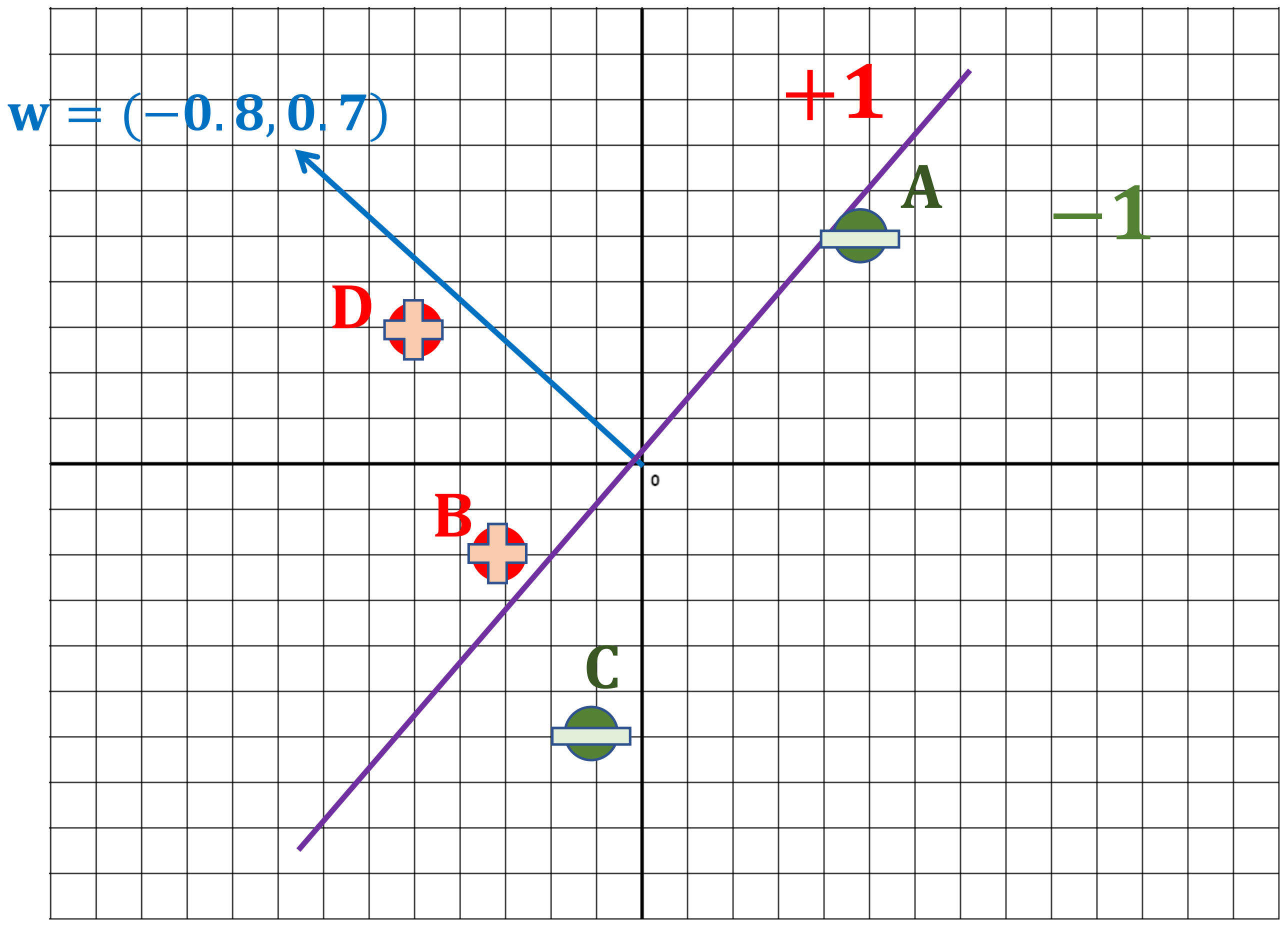
Training data

Item	x_1	x_2	Class
A	0.6	0.5	-1
B	-0.3	-0.2	1
C	-0.1	-0.6	-1
D	**-0.5**	**0.3**	**1**

For input D

- $\mathbf{w}^{\mathrm{T}}\mathbf{x} = (-0.8 * -0.5 + 0.7 * 0.3) = 0.61 \;\; \geq 0$
- $y = 1$, and $t = 1$
- Don't update the weights

Learning rate: $\eta = 1, \mathbf{w} = (-0.8, 0.7)$



Perceptron Learning Example

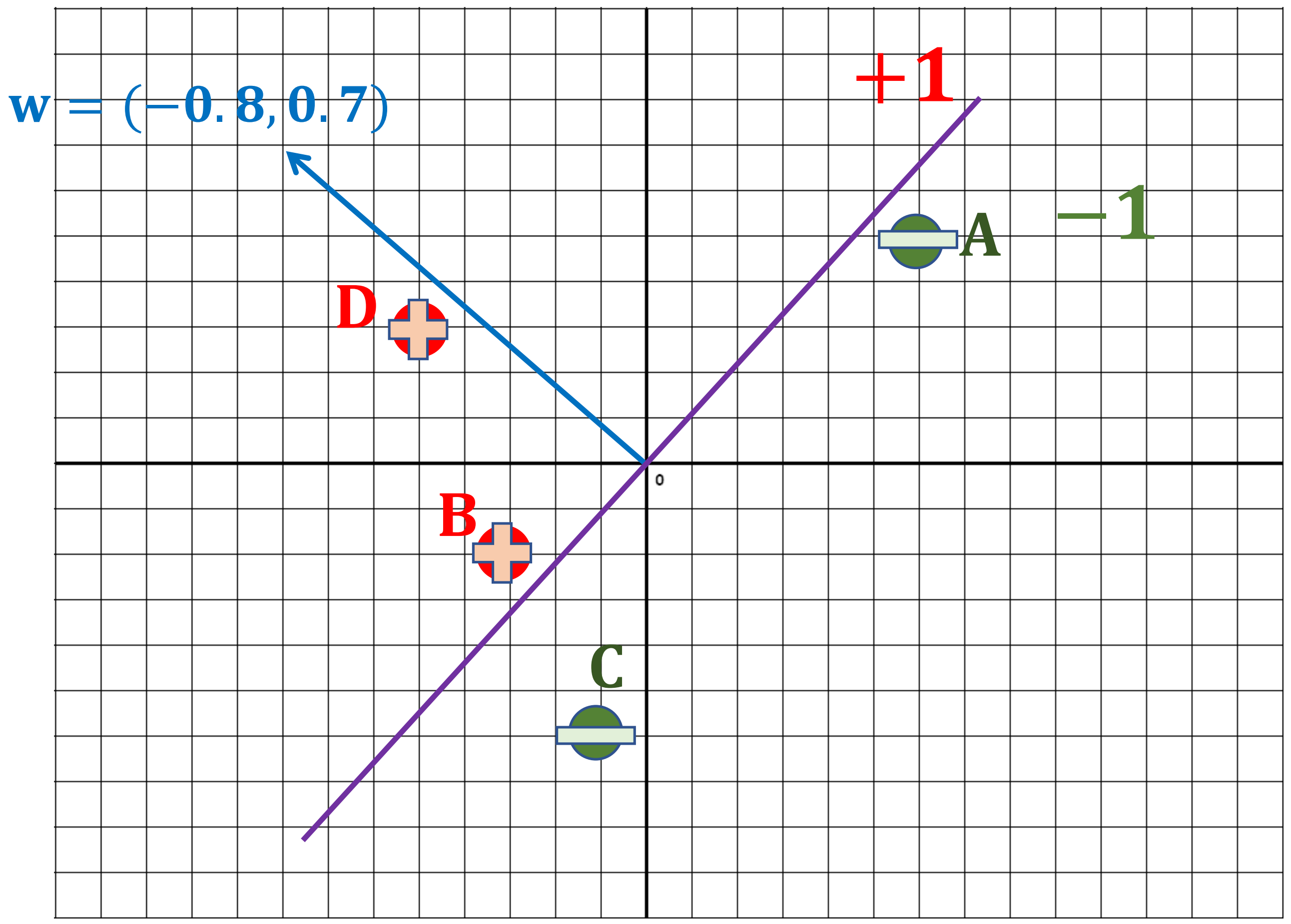
Training data

Item	x_1	x_2	Class
A	0.6	0.5	-1
B	-0.3	-0.2	1
C	-0.1	-0.6	-1
D	-0.5	0.3	1

For input A

- $\mathbf{w}^{\mathrm{T}}\mathbf{x} = (-0.8 * 0.6 + 0.7 * 0.5) = -0.13 < 0$
- $y = -1$, and $t = -1$
- Don't update the weights
- **After this step, all data items have been correctly classified with the weights**

Learning rate: $\eta = 1$, $\mathbf{w} = (-0.8, 0.7)$



Important Remarks

- If the data is linearly separable, Perceptron is guaranteed to find a perfect weight vector. However, it may need too many iterations over the data to converge to a perfect weight vector

- Perceptron may not converge if the data is not separable. In this case, it may cycle through some weight vectors without stopping

- Perceptron is sensitive to initialisation. Two runs of the algorithm may converge to different perfect weights if they visit the training examples in the training set in different orders.

Multiclass Perceptron

Note:

- y_n: predicted label of $\mathbf{x}_n$, t_n: true label of $\mathbf{x}_n$
- Maintain a set of k weight vectors, where k is the number of classes.

Algorithm

Repeat until all data points are classified correctly or the error $< \epsilon$:

- For each training point $\mathbf{x}_n$, perform the following
    - Predict the class whose weight vector produces the highest score: $y_n = \arg\max_k (\mathbf{w}_k^{\mathrm{T}} \mathbf{x}_n)$
    - If correct, do nothing
    - If wrong, update the weights (i.e. if $(y_n \neq t_n)$)
        - reduce score of wrong answer: $\mathbf{w}_{y_n} = \mathbf{w}_{y_n} - \eta \mathbf{x}_n$
        - increase score of right answer: $\mathbf{w}_{t_n} = \mathbf{w}_{t_n} + \eta \mathbf{x}_n$

Tutorial (Week 5)

- Practice how to generate synthetic data for binary classification problem
- Build your perceptron model based on the data

What will we learn in Week 6?

Learn Probabilistic Generative Models

Learn Probabilistic Discriminative Models

THANK YOU
for your attention!