UTS 2019
MACHINE LEARNING – TRAINING ALGORITHM IN A LINEAR CLASSIFICATION MODEL

MOD1: REVIEW AND WHAT LEARNING DOES

LEARNING PROBLEM AND A COMMON FRAMEWORK

LEARNING FRAMEWORK

- Four elements in a learning process
  - Data distribution (black box), producing Data (observation)
  - Hypothesis family (data model)
  - Selection method (training/fitting/learning algorithm)
  - Criterion (loss/objective function)

LEARNING FRAMEWORK

$$x \to y$$

$$\{(x_1, y_1), (x_2, y_2), \dots\}$$

Loss

Algorithm

$h^* \in \mathcal{H}$

$$\mathcal{H} : \{h_1, h_2, \dots,\} : x \to y$$

LEARNING FRAMEWORK

$x \to y$

$\{(x_1, y_1), (x_2, y_2), \dots\}$

SOURCE OF RANDOMISATION

Loss

Algorithm

$h^* \in \mathcal{H}$

$\mathcal{H} : \{h_1, h_2, \dots, \} : x \to y$

LEARNING FRAMEWORK

$x \to y$

$\{(x_1, y_1), (x_2, y_2), \dots\}$

SOURCE OF RANDOMISATION

$h^* \in \mathcal{H}$

Loss

Algorithm

$\mathcal{H} : \{h_1, h_2, \dots, \} : x \to y$

DATA MODEL (FAMILY) MAIN FACTOR UNDER CONTROL

LEARNING FRAMEWORK

$x \to y$

$\{(x_1, y_1), (x_2, y_2), \dots\}$

SOURCE OF RANDOMISATION

$h^* \in \mathcal{H}$

FINAL DELIVERABLE TO DEPLOY AND TO TAKE RISK

Loss

Algorithm

$\mathcal{H} : \{h_1, h_2, \dots, \} : x \to y$

DATA MODEL (FAMILY) MAIN FACTOR UNDER CONTROL

LEARNING FRAMEWORK

$x \to y$

$\{(x_1, y_1), (x_2, y_2), \dots\}$

SOURCE OF RANDOMISATION

DEFINING THE "TASK", NOT DISCUSSED

Loss

Algorithm

$h^* \in \mathcal{H}$

FINAL DELIVERABLE TO DEPLOY AND TO TAKE RISK

$\mathcal{H} : \{h_1, h_2, \dots, \} : x \to y$

DATA MODEL (FAMILY) MAIN FACTOR UNDER CONTROL

LEARNING FRAMEWORK

$x \to y$

$\{(x_1, y_1), (x_2, y_2), \dots\}$

SOURCE OF RANDOMISATION

DEFINING THE "TASK", NOT DISCUSSED

Loss

Algorithm

SELECTION/ LEARNING PROCESS – THIS LECTURE

$h^* \in \mathcal{H}$

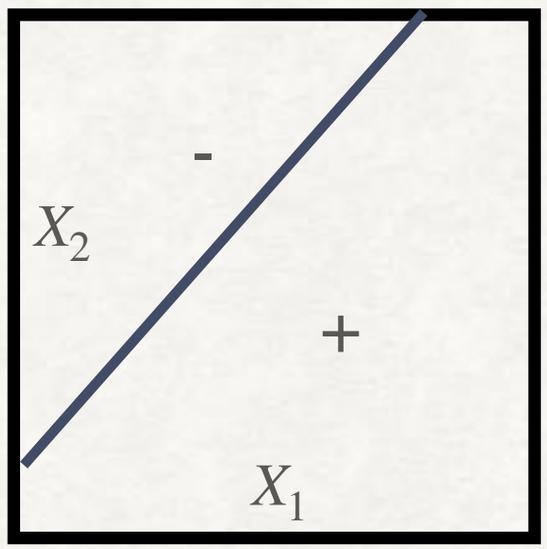
FINAL DELIVERABLE TO DEPLOY AND TO TAKE RISK

$\mathcal{H} : \{h_1, h_2, \dots, \} : x \to y$

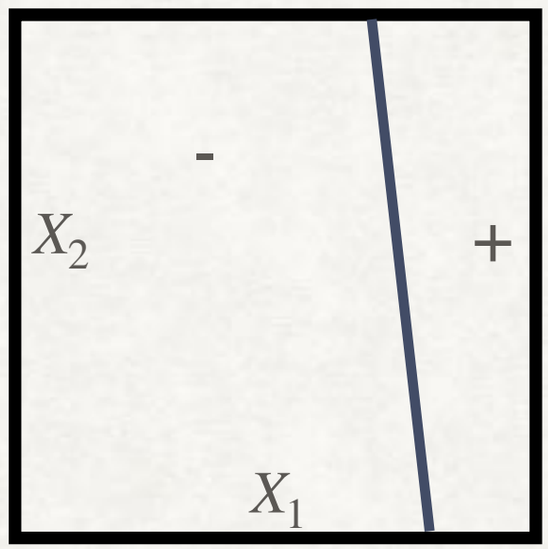
DATA MODEL (FAMILY) MAIN FACTOR UNDER CONTROL

TRAINING

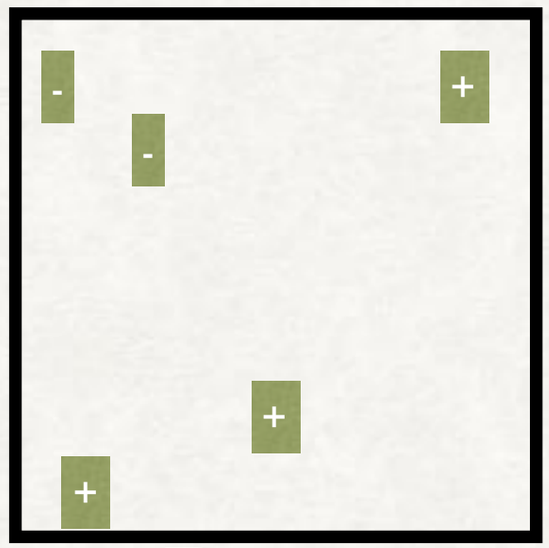
- Select ONE hypothesis from the hypothesis family so the prediction of this hypothesis fit to data.
- Given two hypotheses, and data, formulate the problem into the framework-view



h_1

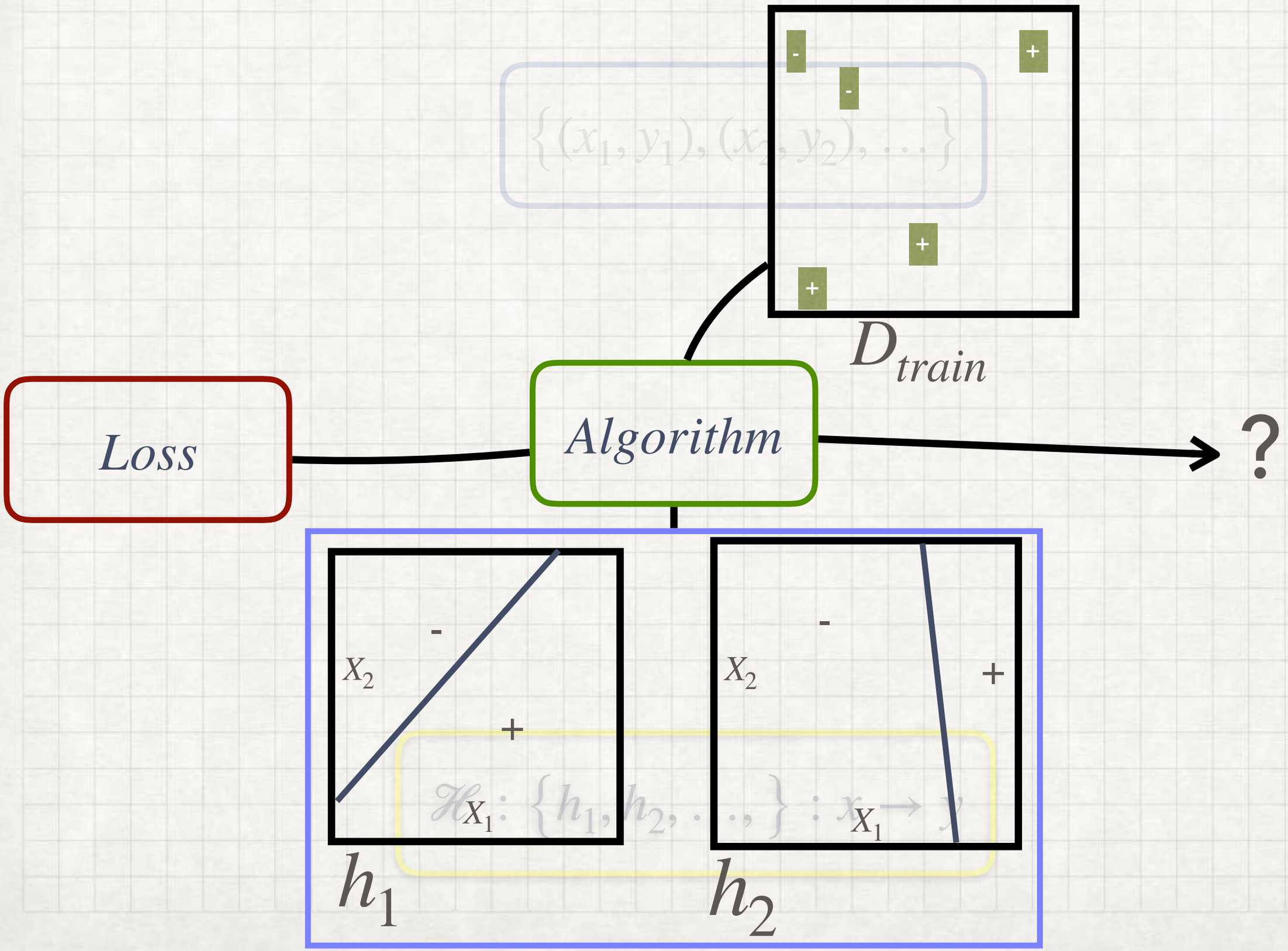


h_2



D_{train}

LEARNING – A SIMPLE SELECTION TASK



LEARNING IN COMPLEX $\mathscr{H}$

- Searching in hypotheses space
  - Structure of $\mathscr{H}$: continuous / discrete? fixed parameterisation? similarity between hypotheses?
  - Guidance by task-related criterion: search in $\mathscr{H}$ / global optimum? bounded? improvement guaranteed?
  - Search algorithm: keep historical hypotheses (momentum)? predictive optimisation? approximation?
- Example: Train a Perceptron.

MOD2: LINEAR MODELS AND PERCEPTRON

(GENERALISED) LINEAR MODELS

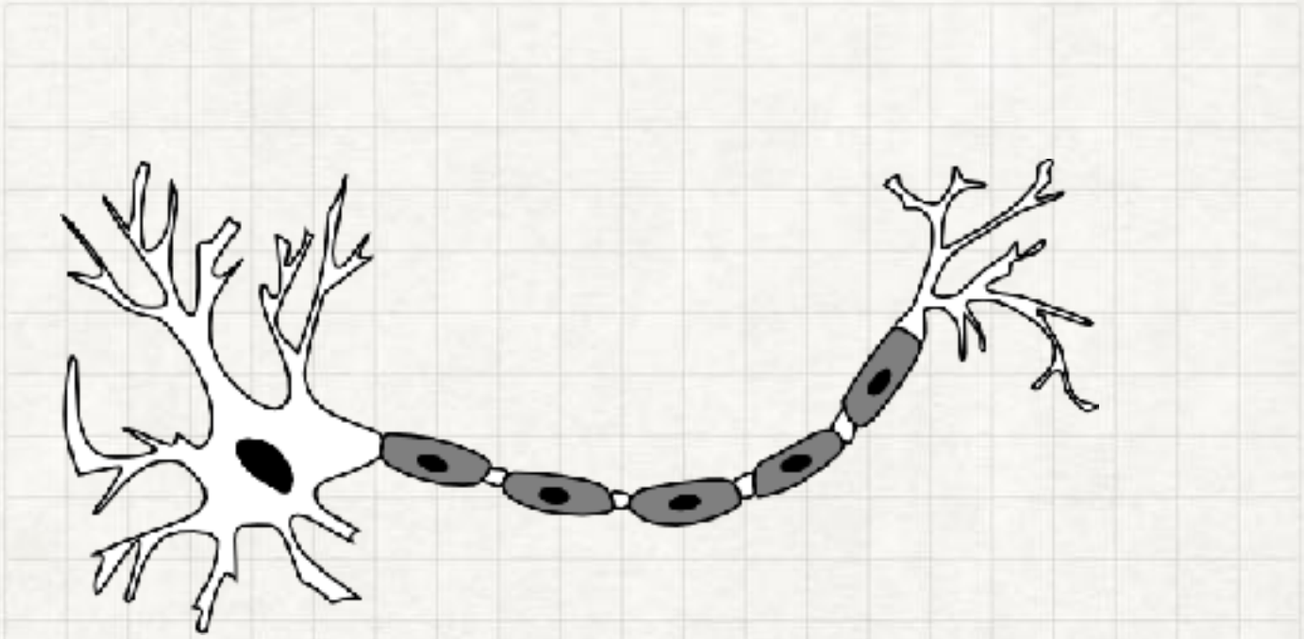
- A General Design
- E.g. let us consider data of two attributes. A linear model computes the weighted sum given a sample, (x_1, x_2):
- $a = w_1 x_1 + w_2 x_2 + b$
- The subsequent operations / decision makings on a, combined with pre-processing steps to prepare x_1 and x_2, make a very rich data modelling toolbox.

PERCEPTRON

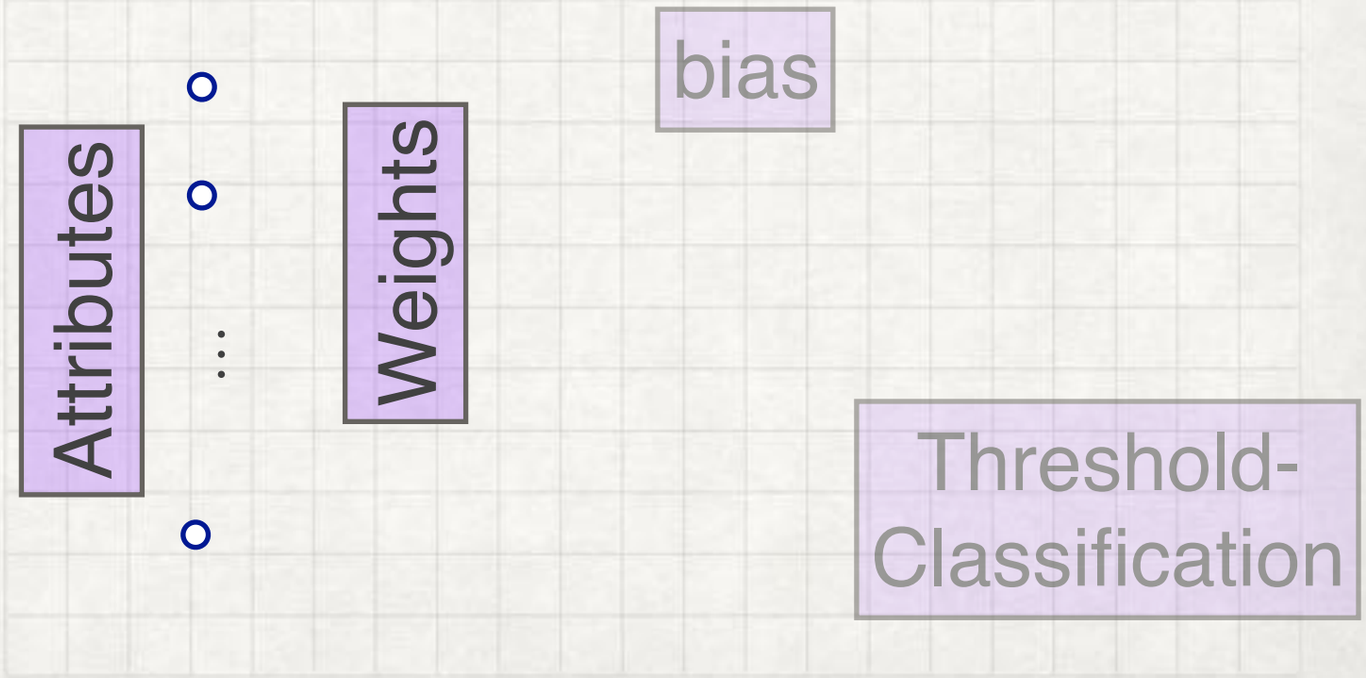
- Rosenblatt, Frank (1957), The Perceptron--a perceiving and recognizing automaton. Report 85-460-1, Cornell Aeronautical Laboratory

PERCEPTRON: THE MODEL

- Inspired by neurons

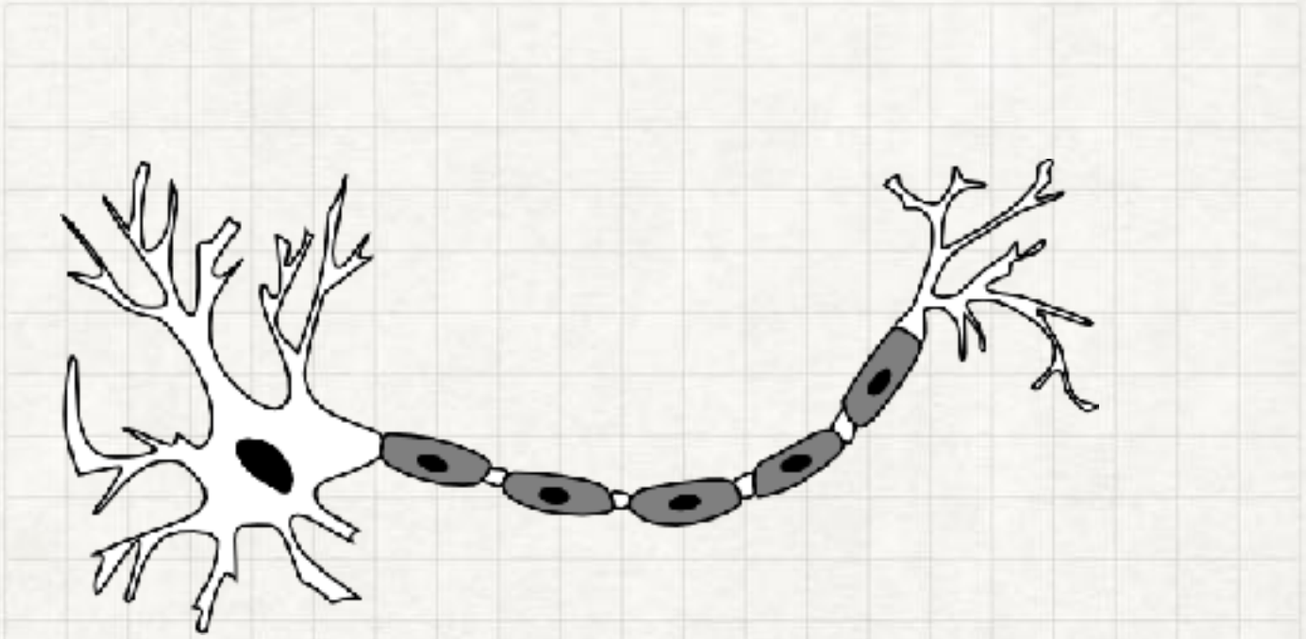


- Making decision by weighted sum of inputs

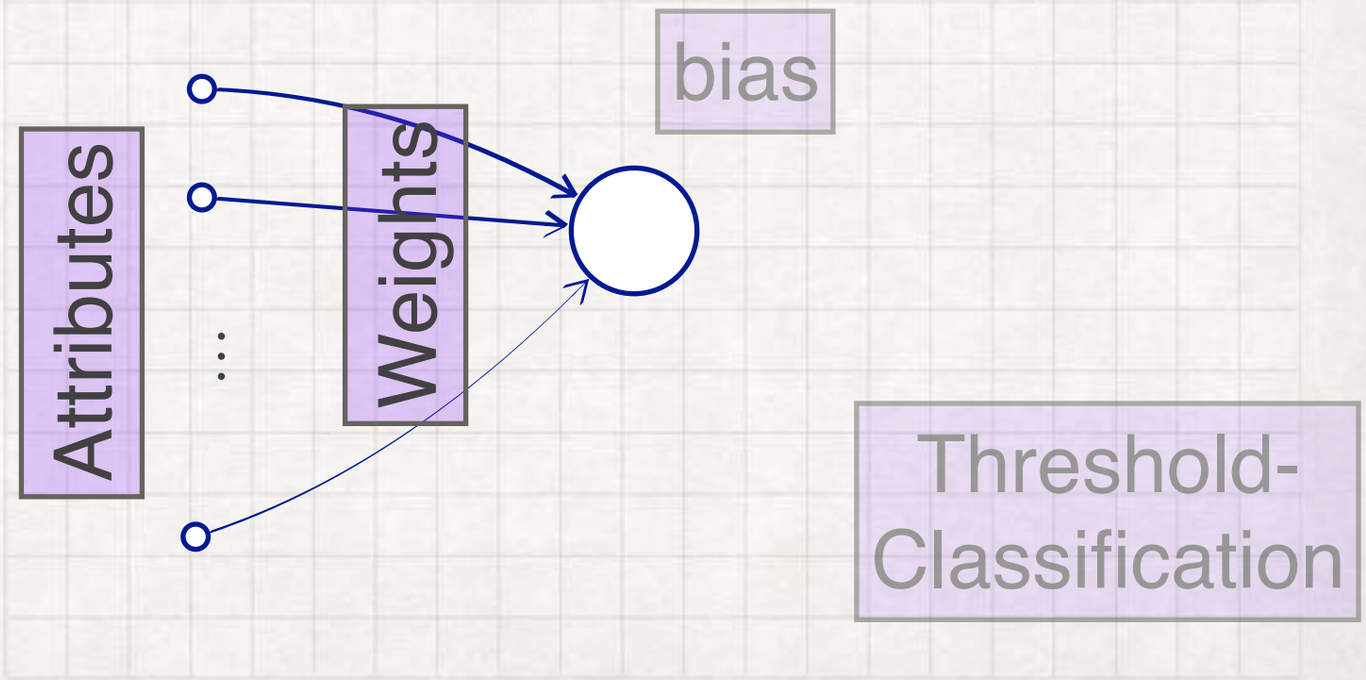


PERCEPTRON: THE MODEL

- Inspired by neurons

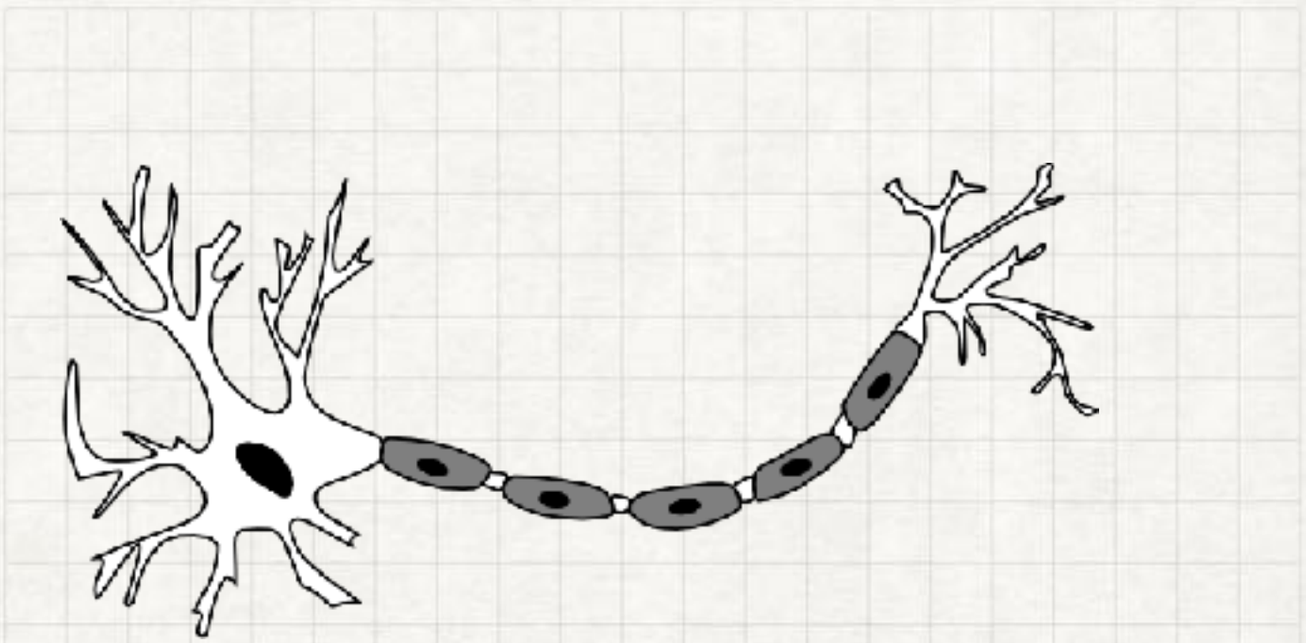


- Making decision by weighted sum of inputs

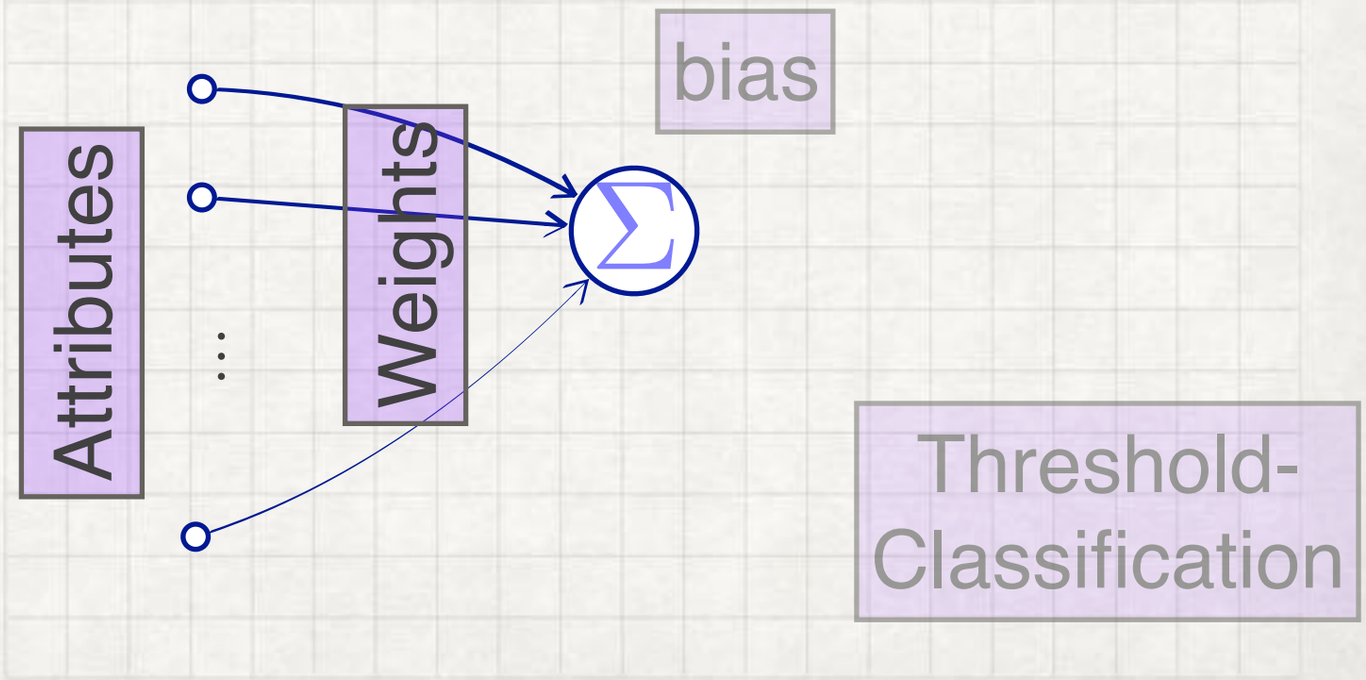


PERCEPTRON: THE MODEL

- Inspired by neurons

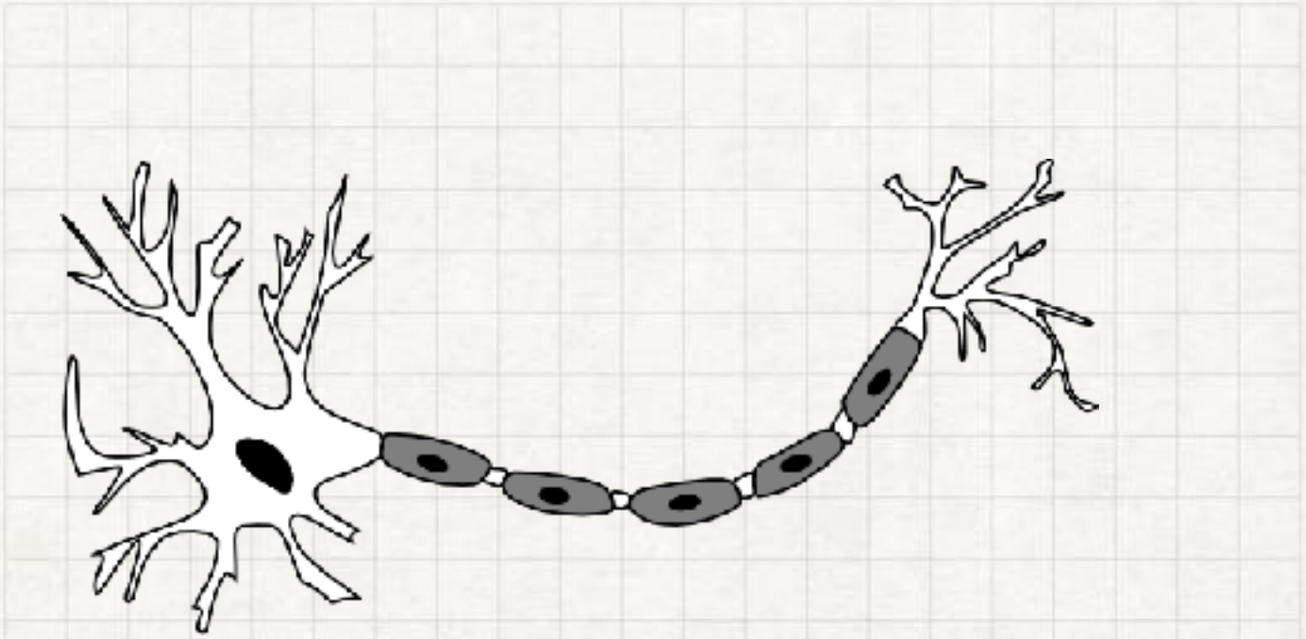


- Making decision by weighted sum of inputs

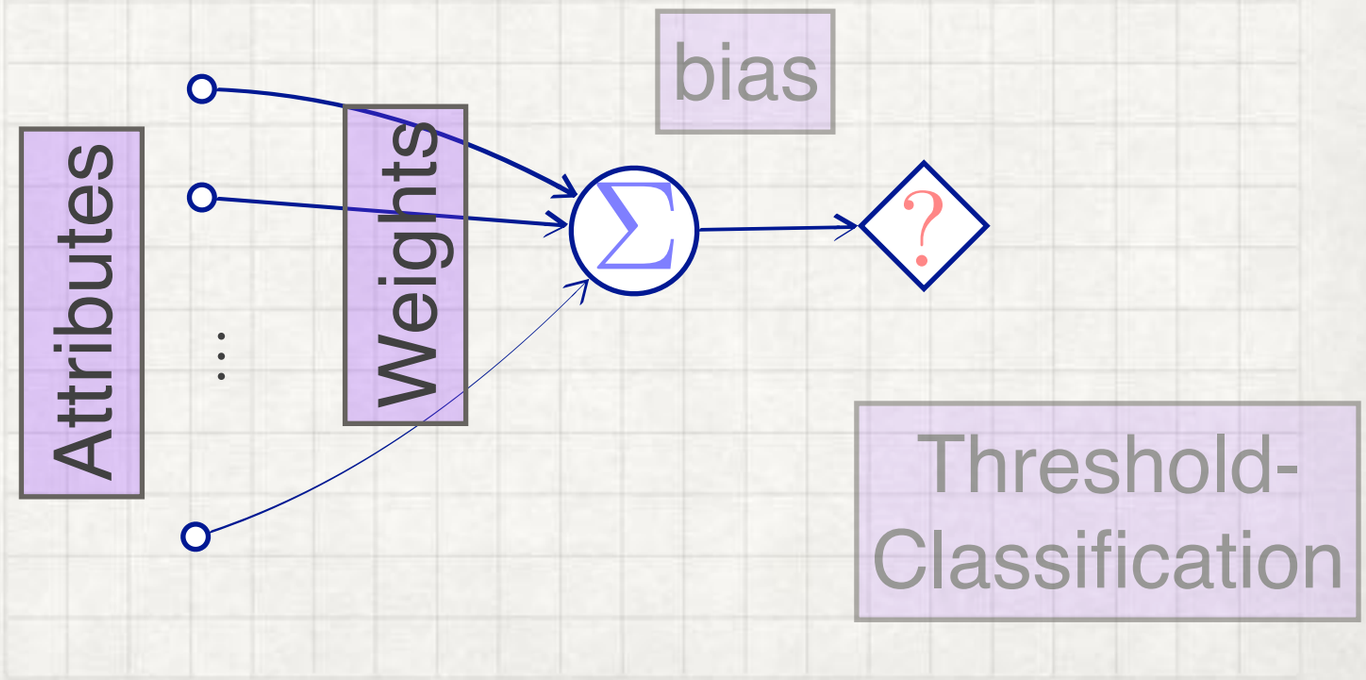


PERCEPTRON: THE MODEL

- Inspired by neurons

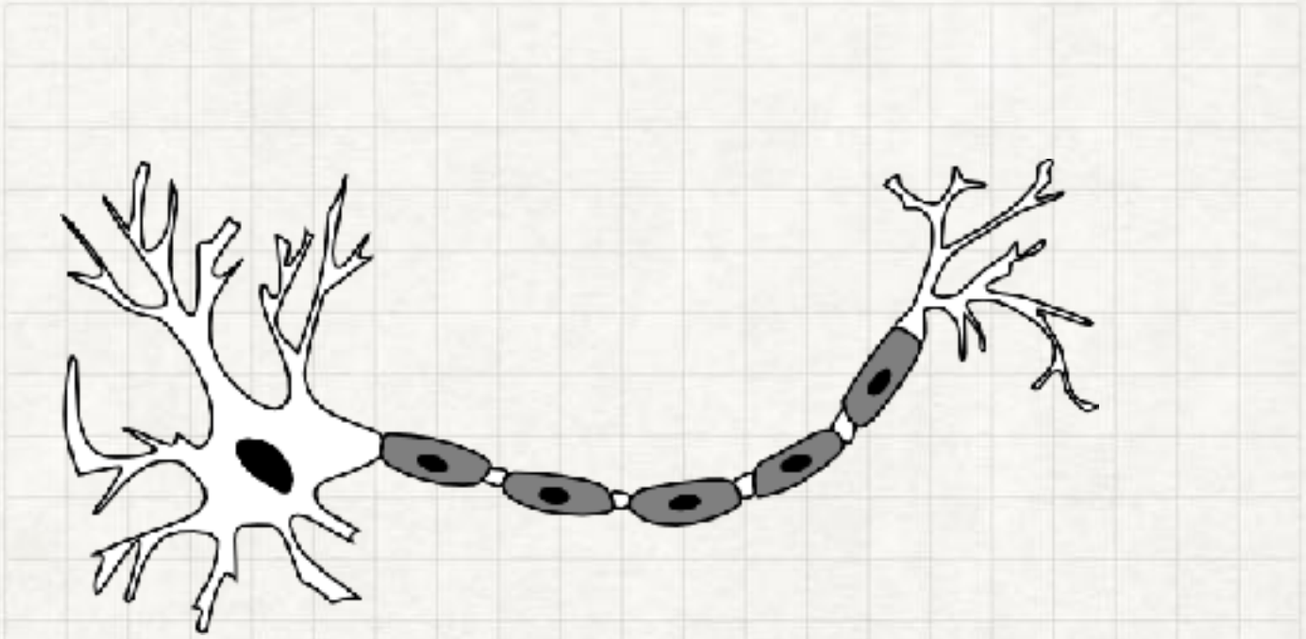


- Making decision by weighted sum of inputs

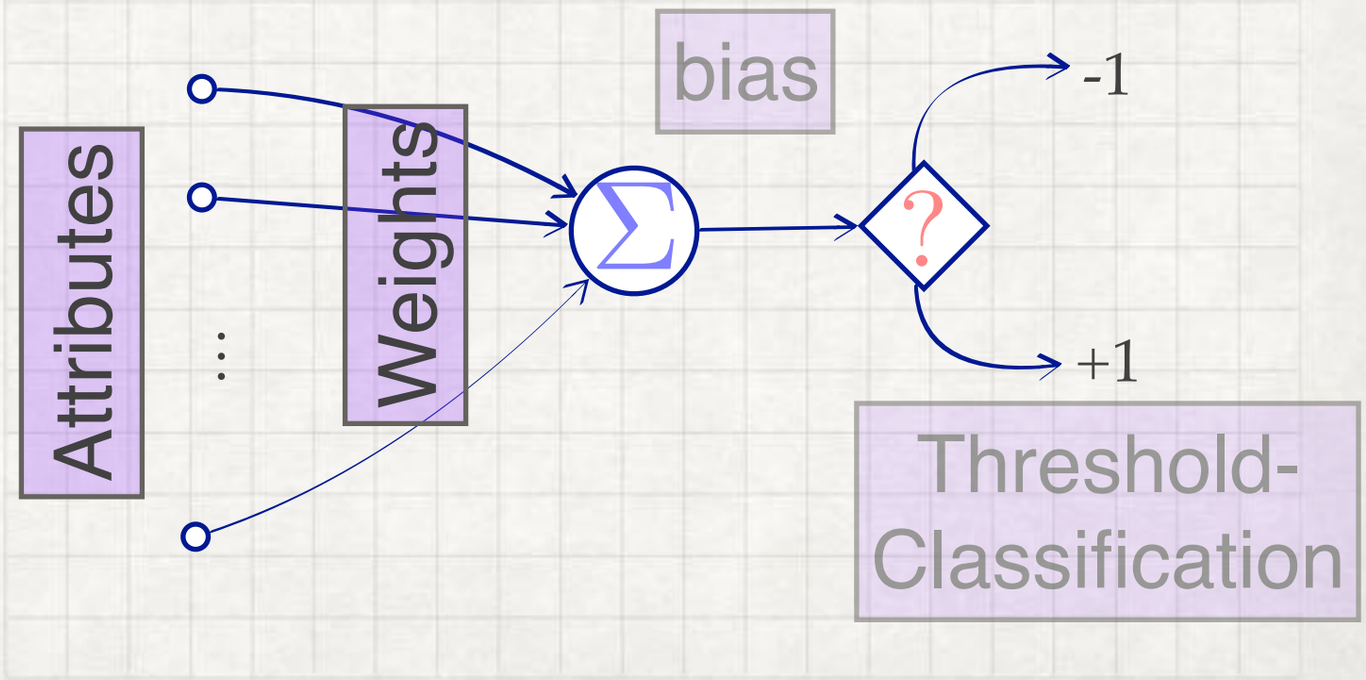


PERCEPTRON: THE MODEL

- Inspired by neurons

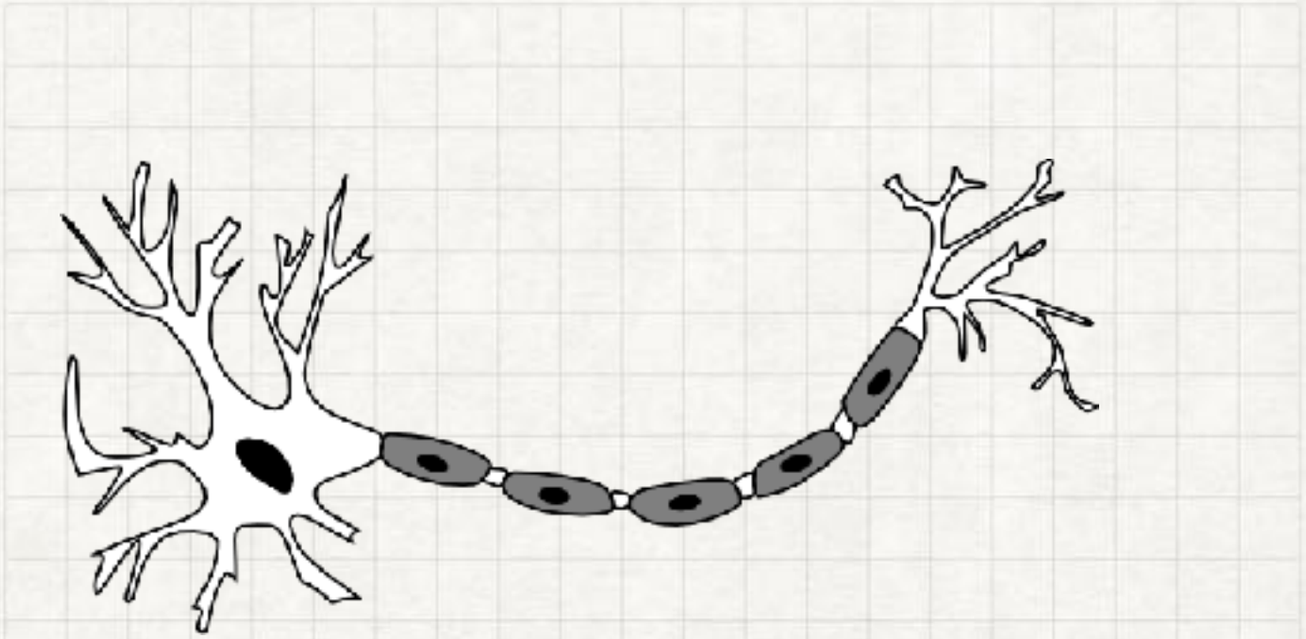


- Making decision by weighted sum of inputs

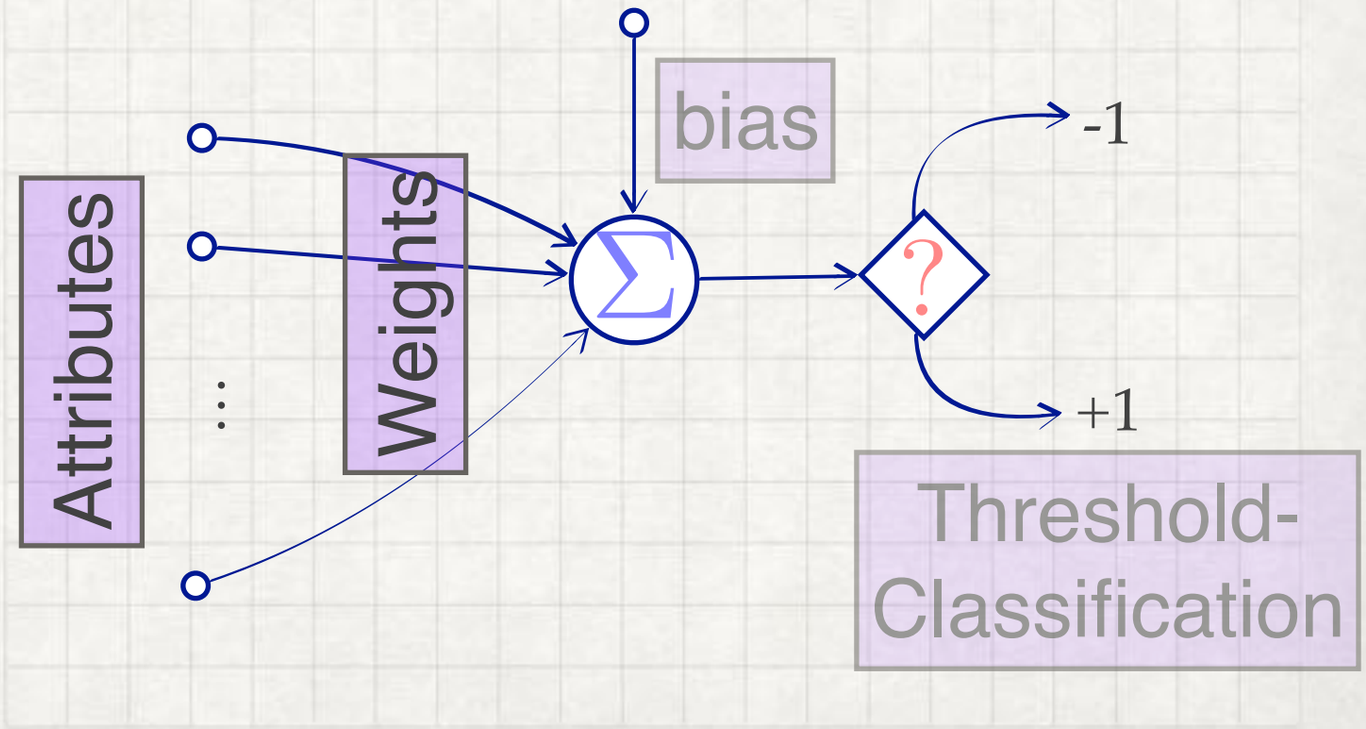


PERCEPTRON: THE MODEL

- Inspired by neurons

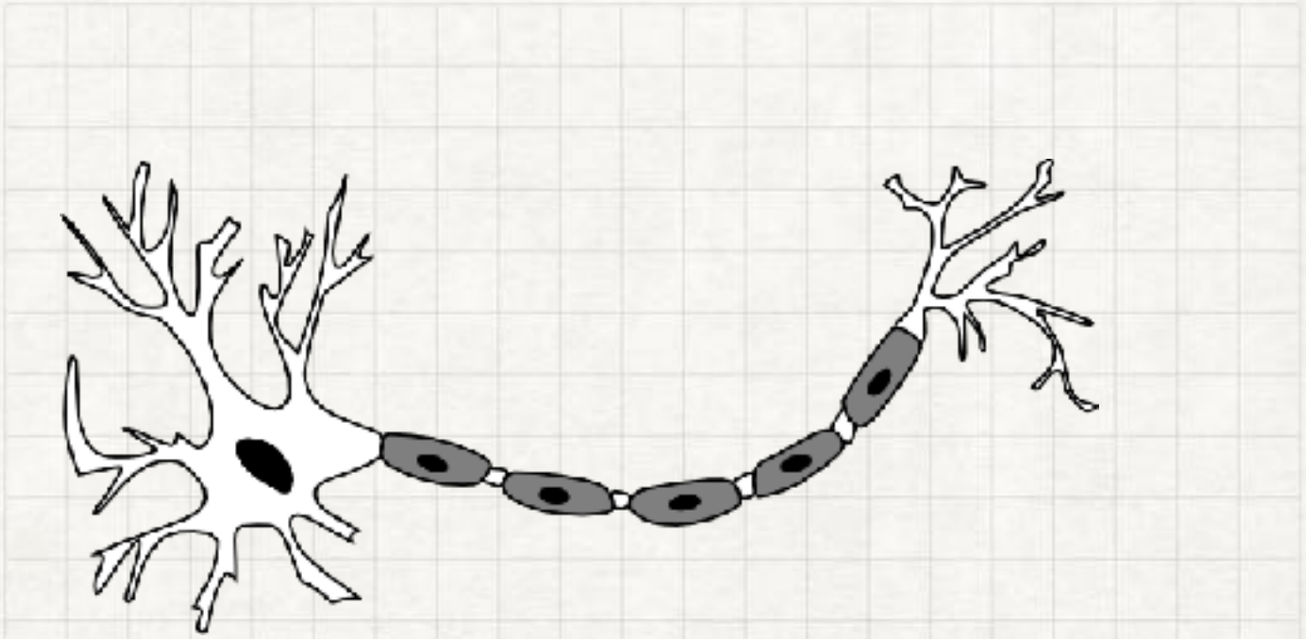


- Making decision by weighted sum of inputs

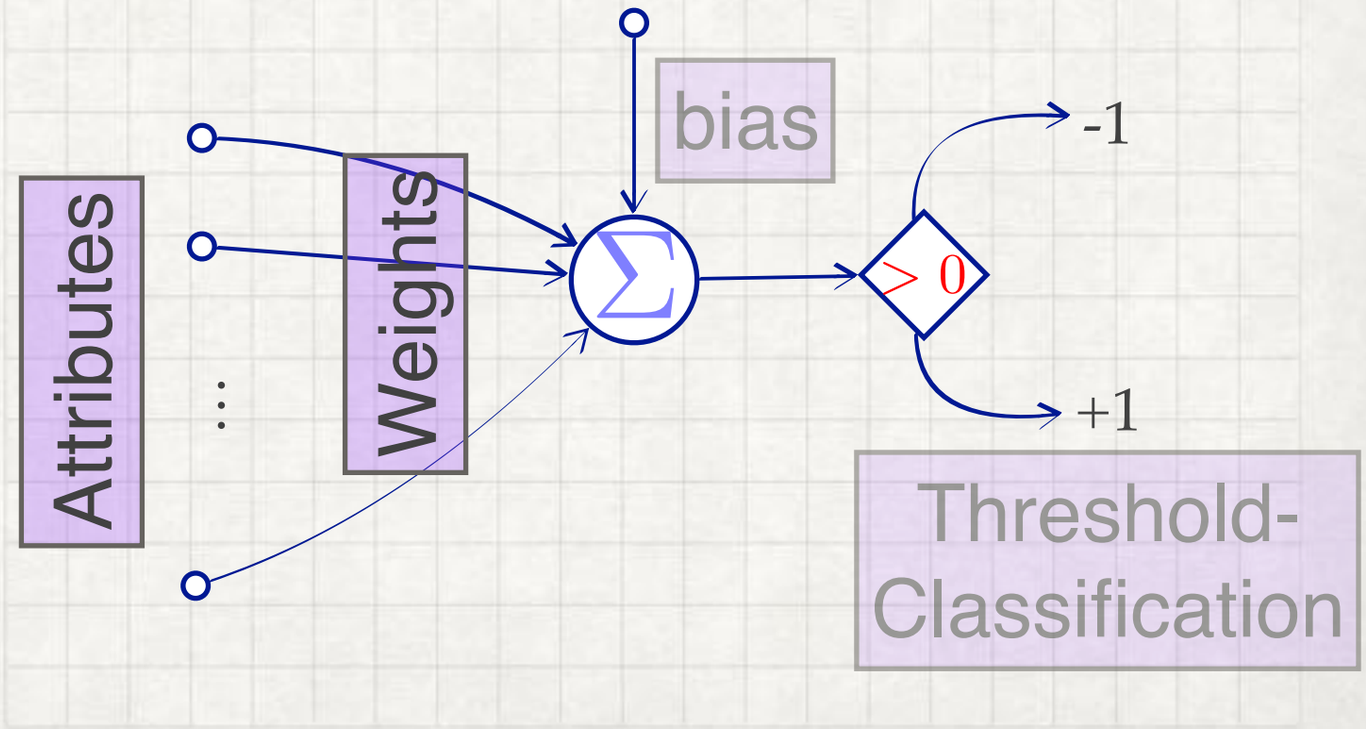


PERCEPTRON: THE MODEL

- Inspired by neurons

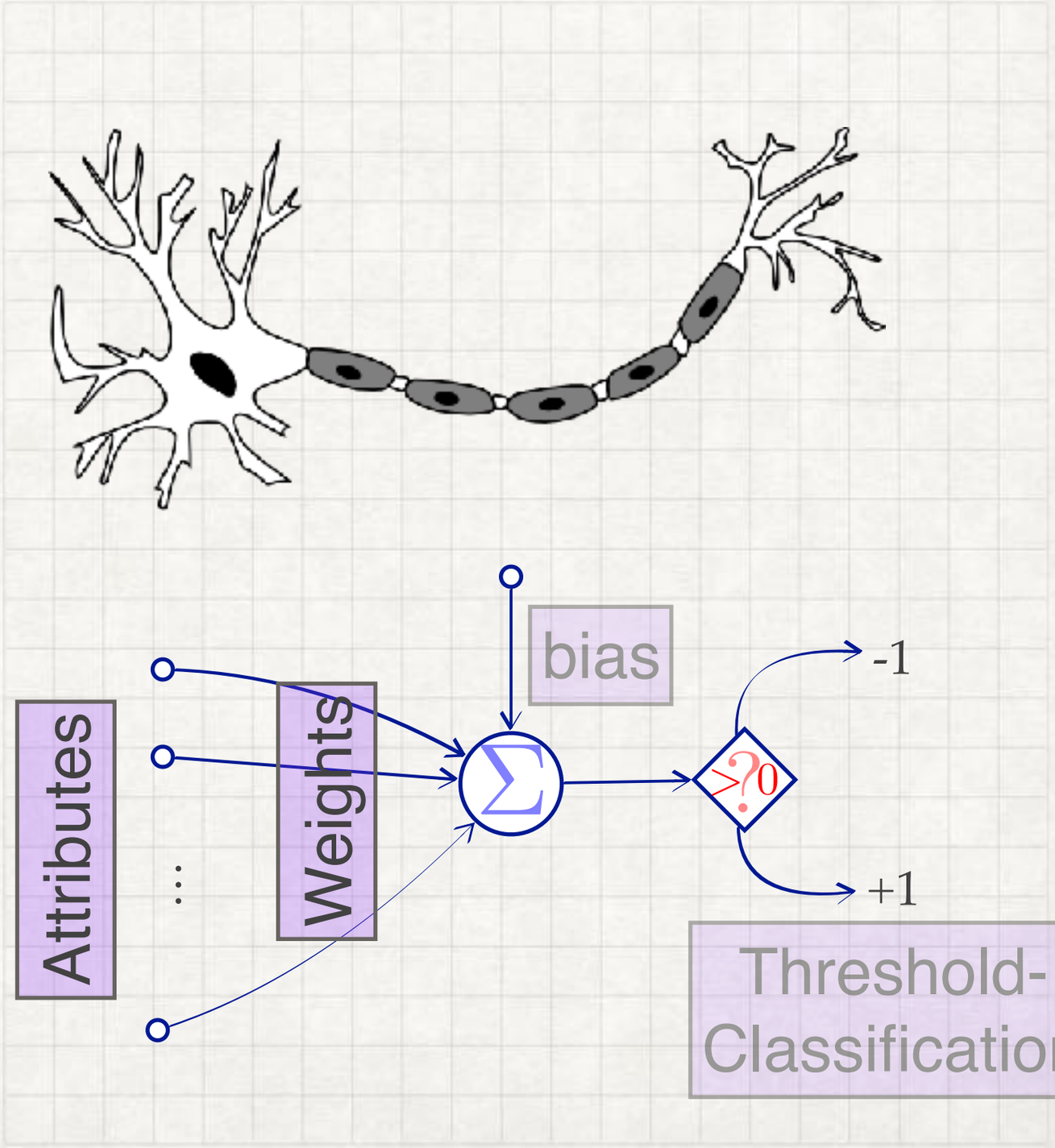


- Making decision by weighted sum of inputs



PERCEPTRON: THE MODEL

- A linear model
- $a = w_1 x_1 + w_2 x_2 + \ldots + w_p x_p + b$
- Decision based on a:
- $y = -1,\ a < 0$
- $y = +1,\ a \geq 0$



PERCEPTRON: THE MODEL

- evaluate:
  - a = x[0] * w[0] + x[1] * w[1] + b
  - y = 1 if a>0 else -1

APPLIED TO DATA *SPACE* – NOT DATA *SAMPLES*

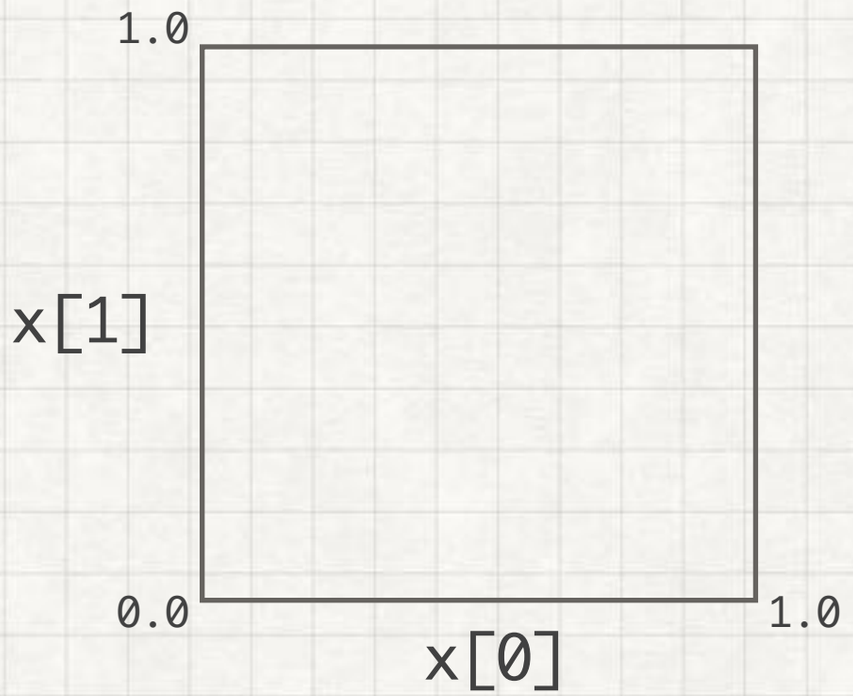
IGNORE BIAS FOR NOW!
You can always choose a suitable threshold after getting a-values with weights.

```
w[0] = 1.0
w[1] = 2.0
a = 1.0 * x[0] + 2.0 * x[1]
```

- Colours are corresponding to “a” values at every possible point in the data space.

GEOMETRIC INTERPRETATION OF A-VALUES

- W has two elements, exactly same form as a data point.
- Consider W as a “virtual data sample” in the data space
- Link from origin, each point in the space can also be treated as a direction.
- A-VALUES == Alignment between the directions of W and a data sample; and W <--> gradient of a-value gradient.



APPLIED TO DATA *SPACE* – NOT DATA *SAMPLES*

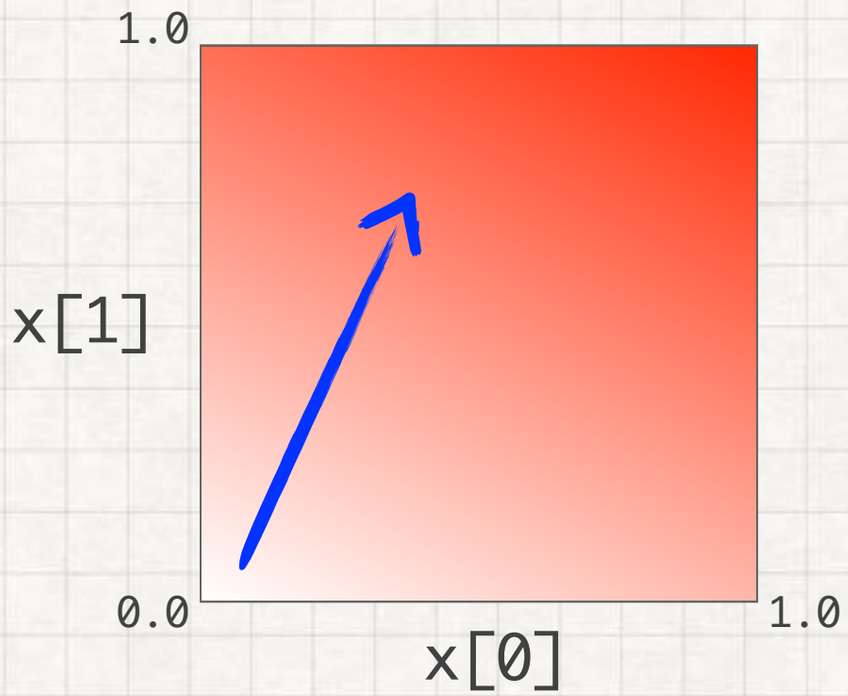
IGNORE BIAS FOR NOW!
You can always choose a suitable threshold after getting a-values with weights.

```
w[0] = 1.0
w[1] = 2.0
a = 1.0 * x[0] + 2.0 * x[1]
```

- Colours are corresponding to “a” values at every possible point in the data space.

GEOMETRIC INTERPRETATION OF A-VALUES

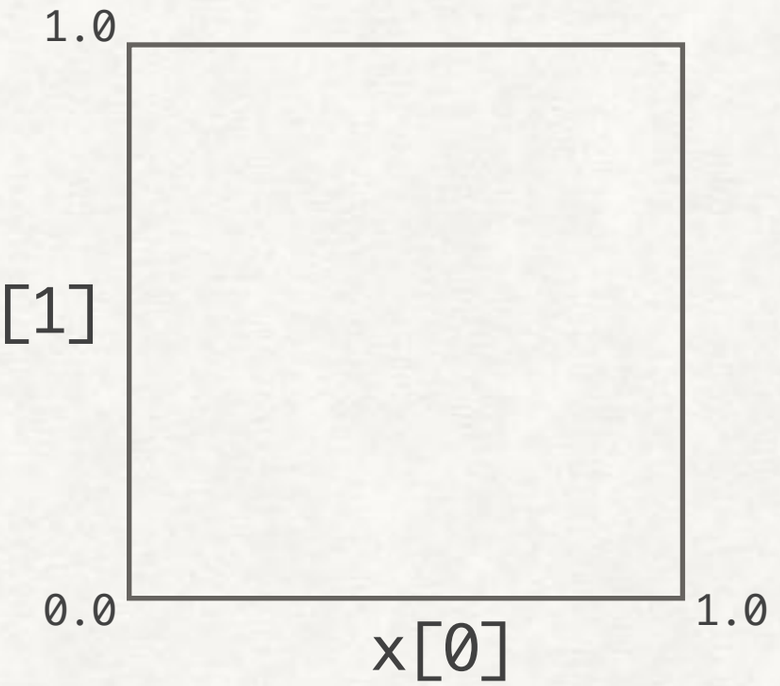
- W has two elements, exactly same form as a data point.
- Consider W as a “virtual data sample” in the data space
- Link from origin, each point in the space can also be treated as a direction.
- A-VALUES == Alignment between the directions of W and a data sample; and W <--> gradient of a-value gradient.



ANOTHER EXAMPLE

```
w[0] = 1.0
w[1] = 0.0
a = 1.0 * x[0]
```

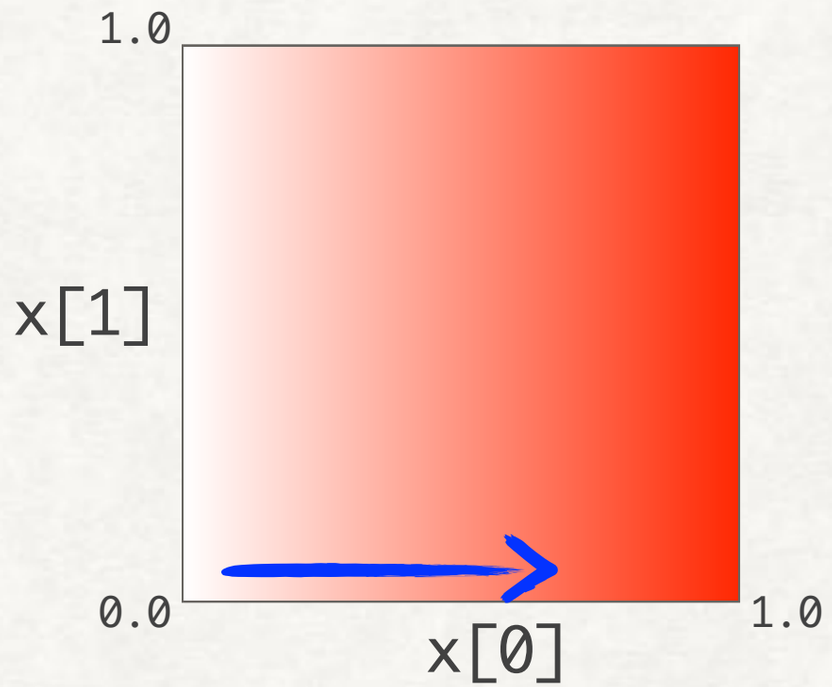
- Now colours have nothing todo with x[1]



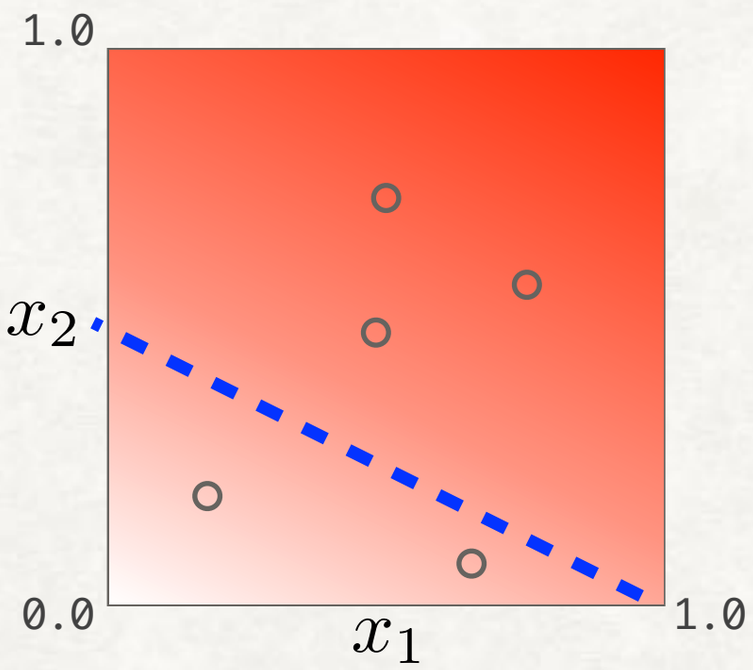
ANOTHER EXAMPLE

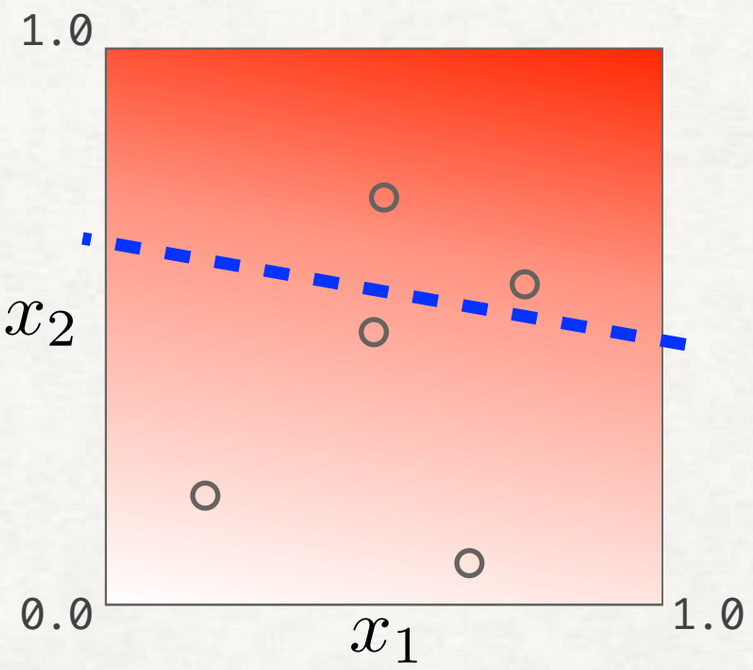
```
w[0] = 1.0
w[1] = 0.0
a = 1.0 * x[0]
```

- Now colours have nothing todo with x[1]

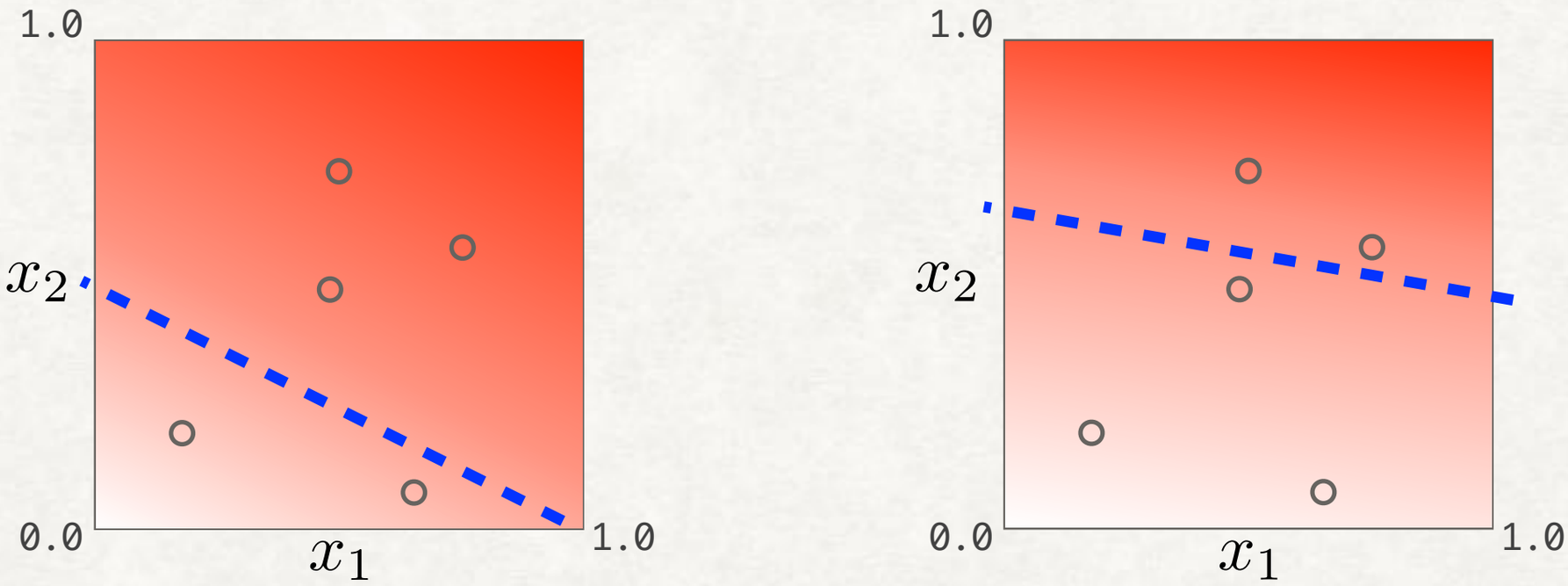


CLASSIFICATION BY THRESHOLDING



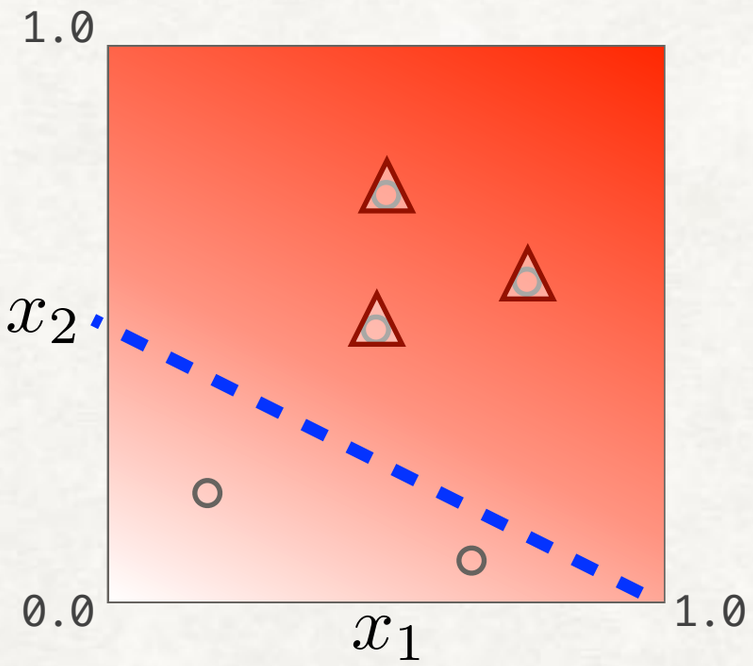


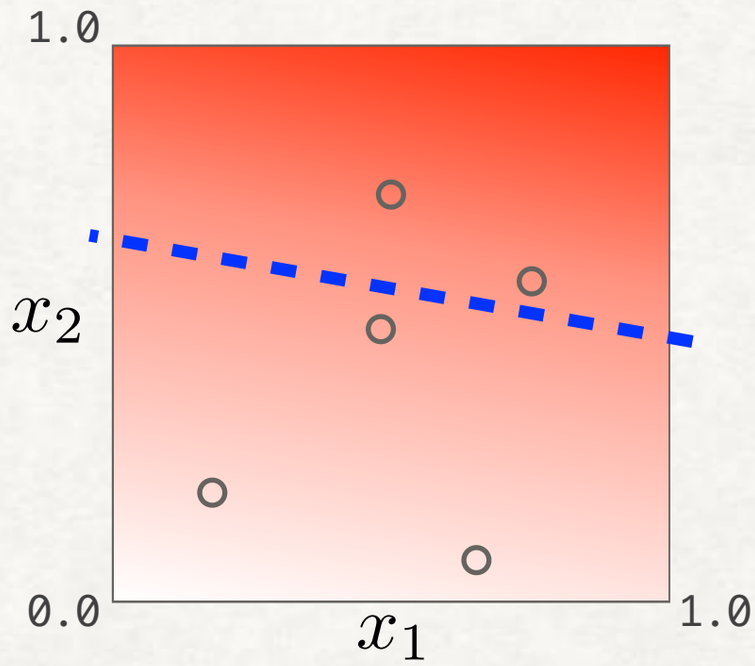
CLASSIFICATION BY THRESHOLDING



- Bias (threshold) can be seen as a triviality, we consider 0-threshold from now on.

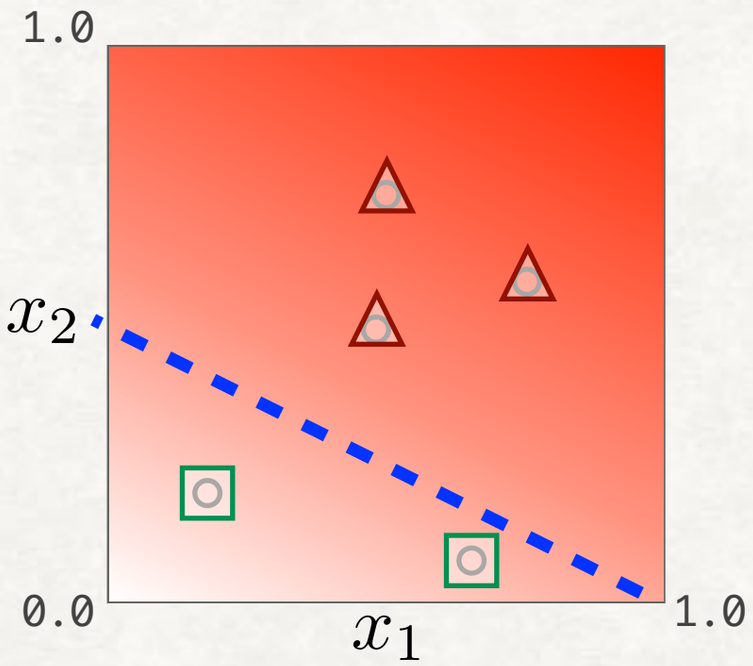
CLASSIFICATION BY THRESHOLDING

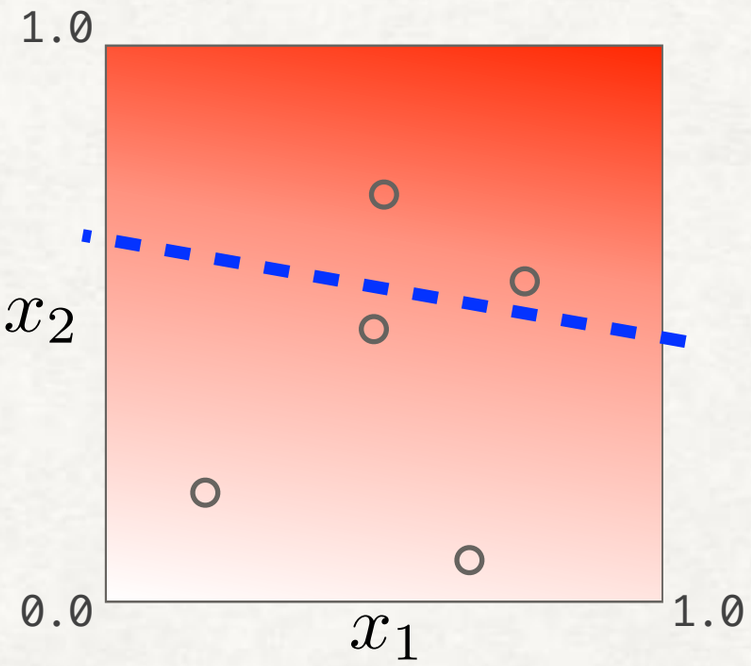




- Bias (threshold) can be seen as a triviality, we consider 0-threshold from now on.

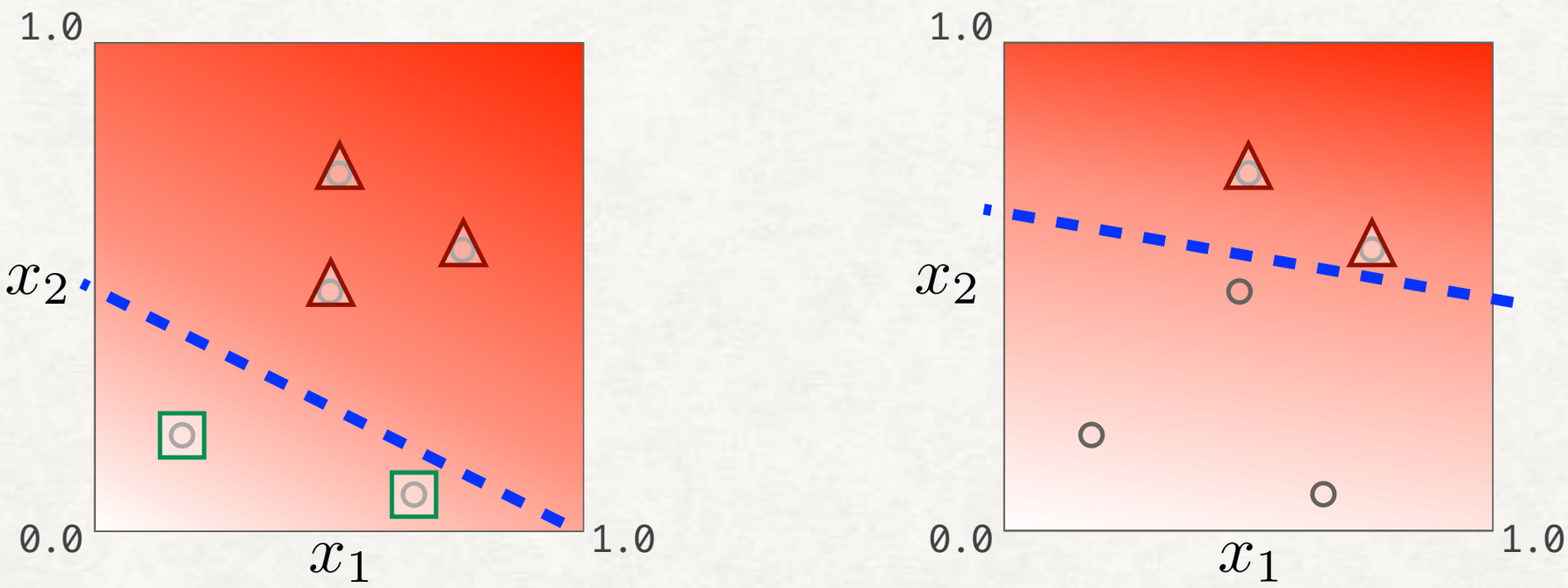
CLASSIFICATION BY THRESHOLDING





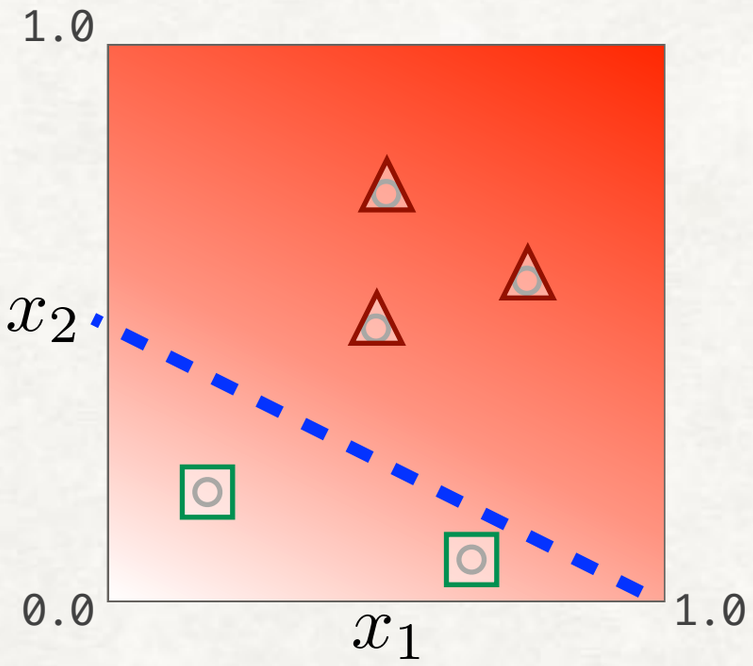
- Bias (threshold) can be seen as a triviality, we consider 0-threshold from now on.

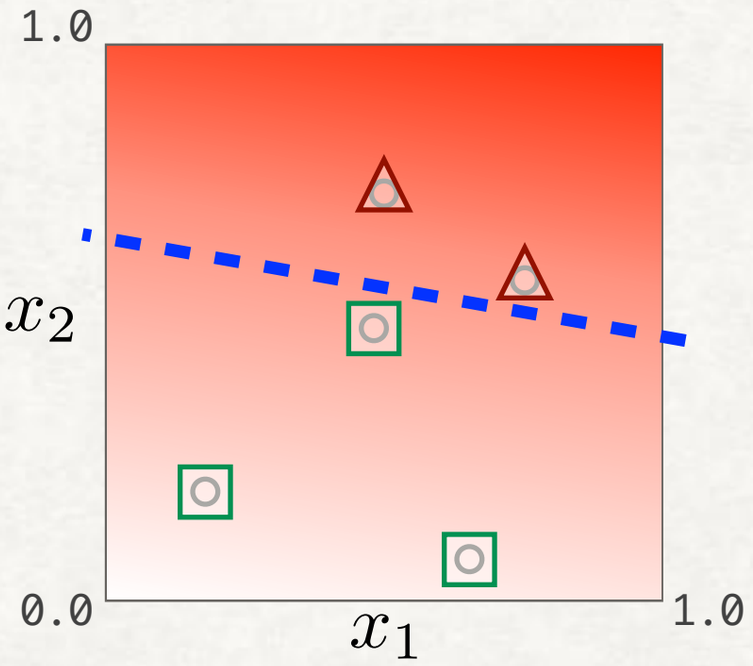
CLASSIFICATION BY THRESHOLDING



- Bias (threshold) can be seen as a triviality, we consider 0-threshold from now on.

CLASSIFICATION BY THRESHOLDING





- Bias (threshold) can be seen as a triviality, we consider 0-threshold from now on.

MOD3: TRAINING A PERCEPTRON

FITTING A PERCEPTRON TO DATA

while there is mis-classified data, say x

- if x: +1,
- if x: -1,

Refer to the notebook for the Perceptron training algorithm. It is highly recommended to try to "discover" the training method by your own.

PERCEPTRON MUST WORK!

- It is guaranteed that if there is some {W} that gives zero classification error (if there exists one such W, almost certainly there are more, why?), the training approach will achieve zero-classification error.

- Sketch of proof
  - Consider such an error free W*
  - Consider the minimum “y-flipped a-value” among all x by W*, must be some p > 0, (why?)
  - Each training step will improve the alignment of the W of question to W* by at least p.
  - Eventually, W and W* will be aligned sufficiently that W makes no error.

- Original proof by the guy we met above

- A good reference CM. Bishop, 1995, “Neural Network for Pattern Recognition”, Oxford Press: Chapter 3.5
- See YS Abu-Mostafa et al. 2012 (check UTS Online, course reference books) for a modern treatment: Chapter 1.1.2, more interpretation, less rigorous.

MORE ON LINEAR MODELS

- Widely used for classification and regression — and a seemingly slight change of
  - the learning goal: how you penalise when the model making errors,
  - the learning step: how you manipulate the W-value to go for a lesser punishment (or better reward, if you'd like a more lenient and optimistic mind-framework 🙂)
  - the learning strategy: how you adapt your learning steps (see above) along with the changing situation

  can fundamentally alter the practical behaviour and theoretical traits of the models.

  Comprehensive accessible and free authority in such matters: Introduction to Statistical Learning: http://www-bcf.usc.edu/~gareth/ISL/

  The book web also refers to a (highly recommended) video course, linear models are Ch3 and Ch4.

- Linear models, and perceptrons are also used as components in more complicated models. We will revisit our old friend soon!

Thanks