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

DEFINING THE "TASK", NOT DISCUSSED

Loss

Algorithm

SELECTION/ LEARNING PROCESS – THIS LECTURE

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

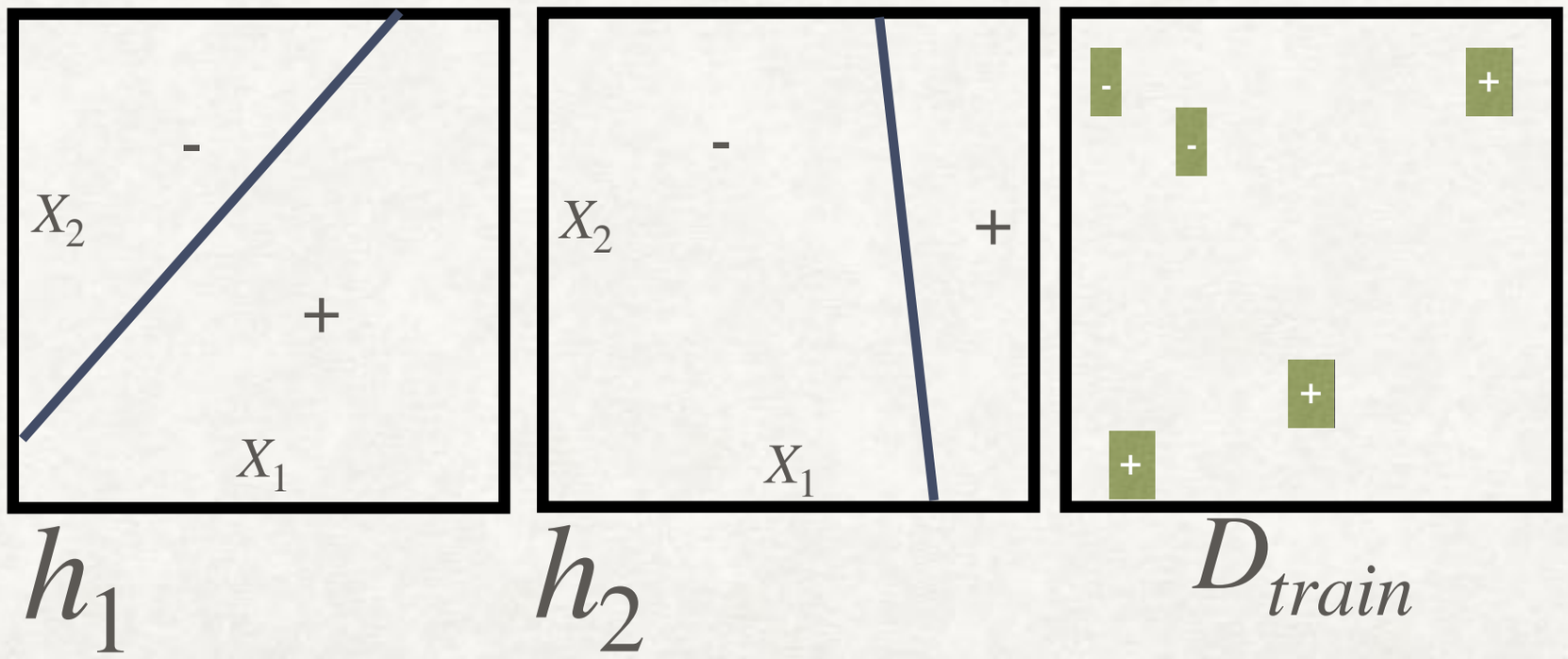
FINAL DELIVERABLE TO DEPLOY AND TO TAKE RISK

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

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



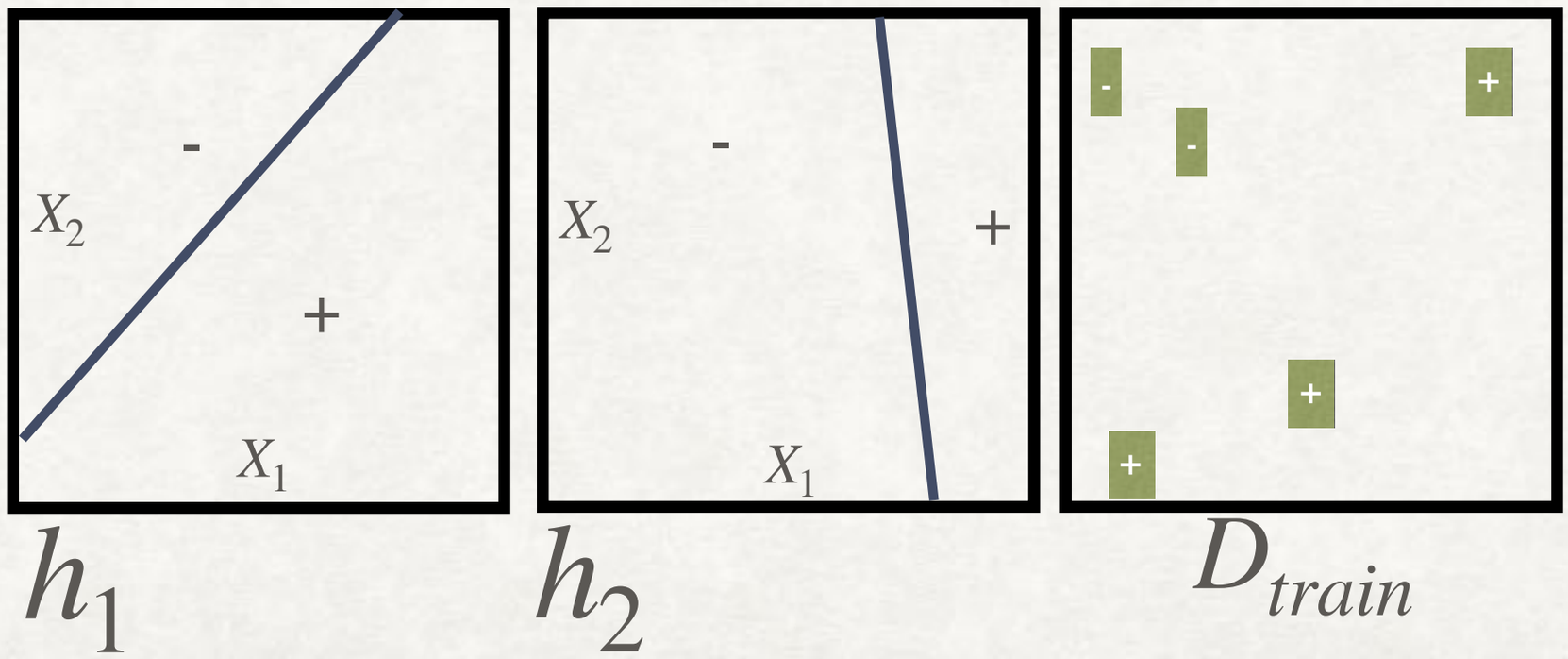
Q: Which hypothesis would you like to select?

A. h1

B. h2

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

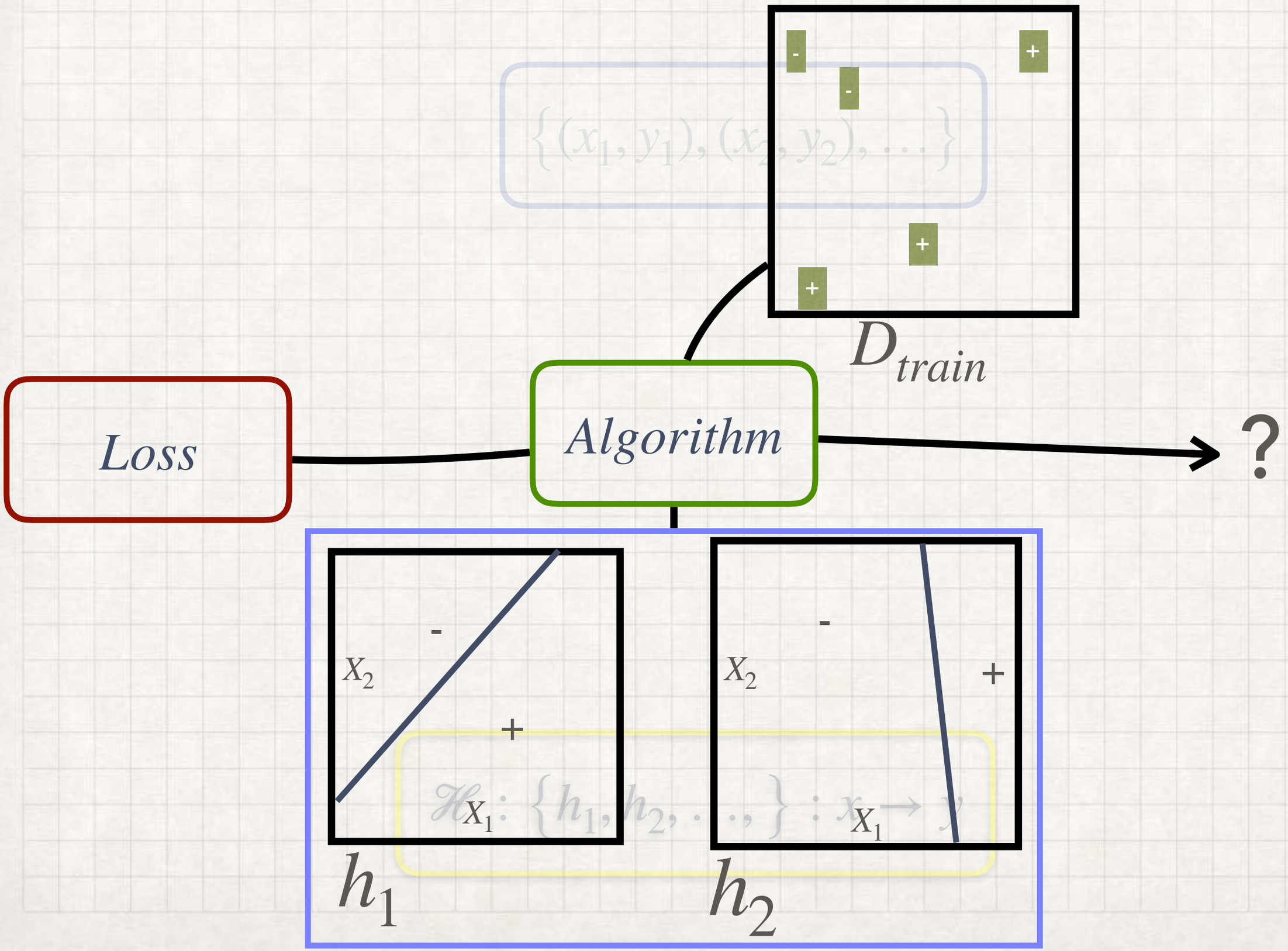


Q: Which hypothesis would you like to select?

A. h1

B. h2

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.

LEARNING IN COMPLEX $\mathscr{H}$

- Searching in hypotheses space
  - Structure of $\mathscr{H}$

Q: Alice is implementing a machine learning technique. Now she is designing the data structure to store individual hypotheses h in $\mathscr{H}$. Which $\mathscr{H}$ allows Alice to allocate a fixed amount of memory for h?

A. Decision Trees

B. AdaBoost

C. Linear Regression

D. 5-Nearest Neighbours

LEARNING IN COMPLEX $\mathscr{H}$

- Searching in hypotheses space
  - Structure of $\mathscr{H}$

Q: Consider models of 2D data. A linear model is specified by a pair of weights, one for each data attribute: (w_1, w_2). Given any two models, A and B, one can obtain new models by interpolation between A and B: $w_1^{New,\alpha} = \alpha w_1^A + (1-\alpha) w_1^B$ (similarly for w_2). Can we do the same for decision trees?

A. Yes.

B. Not obviously.

LEARNING IN COMPLEX $\mathscr{H}$

- Searching in hypotheses space
  - Structure of $\mathscr{H}$

Q: Consider models of 2D data. A linear model is specified by a pair of weights, one for each data attribute: (w_1, w_2). Given any two models, A and B, one can obtain new models by interpolation between A and B: $w_1^{New,\alpha} = \alpha w_1^A + (1-\alpha) w_1^B$ (similarly for w_2). Can we do the same for decision trees?

A. Yes.

B. Not obviously.

COMPLEX STRUCTURE OF $\mathscr{H}$

- Decision trees are of the format such as

h^A

```
X1:cloud-cover > 0.5 ?
   Yes: X2:wind = strong?
      …
   No: X3:humidity > 0.7 ?
      …
```

h^B

```
X2:wind = weak?
   Yes: X3:humidity > 0.5?
      …
```

  - It is not clear how to create hypothesis smoothly traversing between two trees.
  - Even if we artificially make a hypothesis by interpolating the *prediction by* A and that by B

$$h^F(x) = 0.25h^A(x) + 0.75h^B(x)$$

It is not trivial (simple) to find a tree that realises h^F

Consider it this way: you can “mix” beef / fish mince to have a burger as you wish, but this doesn’t mean a new creature “beefish” in the animal domain producing the meat you like.

COST/CRITERION AND THE SEARCH IN $\mathscr{H}$

Q: Consider we are tuning some linear modelling scheme of 2D data:

$$a = w_1 x_1 + w_2 x_2$$

and there is a proposal of increasing w_1 by δ_1: $w_1 \leftarrow w_1 + \delta_1$. How much a will be affected?

A. δ_1

B. $w_1 \delta_1$

C. $x_1 \delta_1$

D. x_1

- How much the cost will be affected, if the cost is

$$(3 - a)^2/2$$

A. δ_1

B. $a\delta_1$

C. $(3 - a)x_1\delta_1$

D. $(3 - a)x_1$

Hint:

```
def cost(w1, w2, x1, x2):
  a = w1*x1 + w2*x2
  return (3 - a) * (3 - a) / 2

w1 = 0.1
w2 = 0.5
x1 = 1.0
x2 = 2.0
delta = 0.001

r1 = cost(w1, w2, x1, x2)
r2 = cost(w1+delta, w2, x1, x2)
print(r1, r2)
print((r2 - r1) / delta)
```

- How much the cost will be affected, if the cost is

```
+1 if a>0.3 else 0
```

A. 0 (mostly), ± 1 (rarely)

B. $a\delta_1$

C. $x_1\delta_1$

D. x_1

MOD2: LINEAR MODELS AND PERCEPTRON

(GENERALISED) LINEAR MODELS

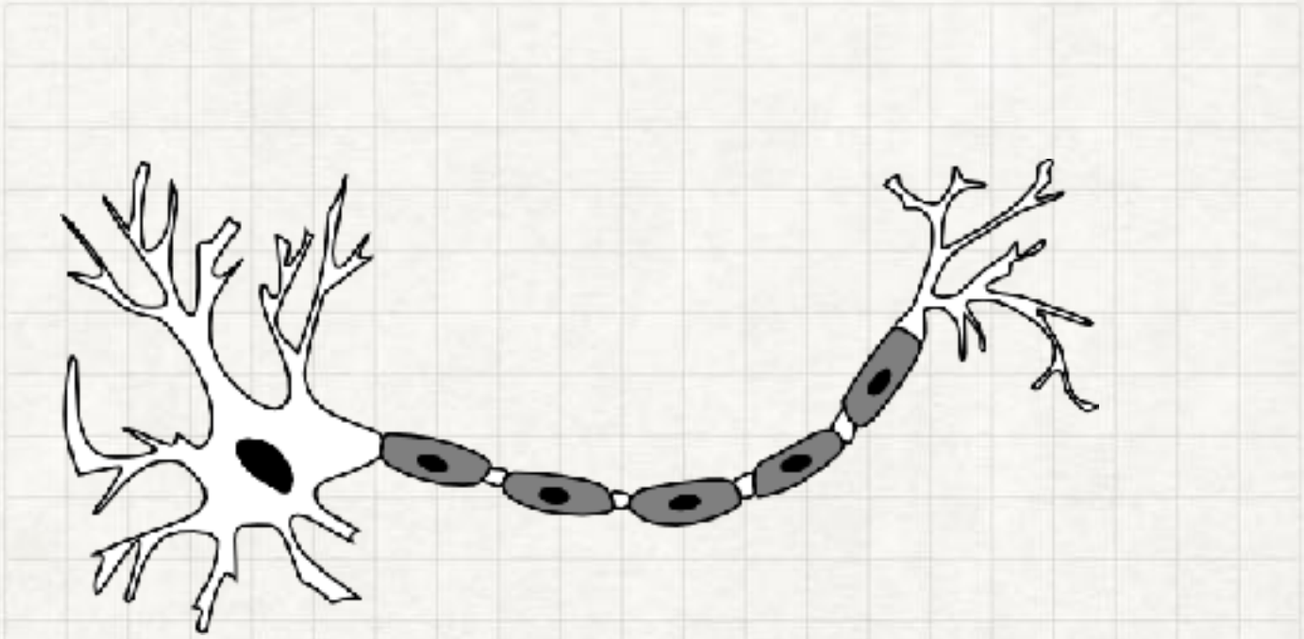
- A General Design
- E.g. let us consider data of two attributes. A linear model computes the weighted sum given a sample, (x_1, x_2):
- $a = w_1 x_1 + w_2 x_2 + b$
- The subsequent operations / decision makings on a, combined with pre-processing steps to prepare x_1 and x_2, make a very rich data modelling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

PERCEPTRON

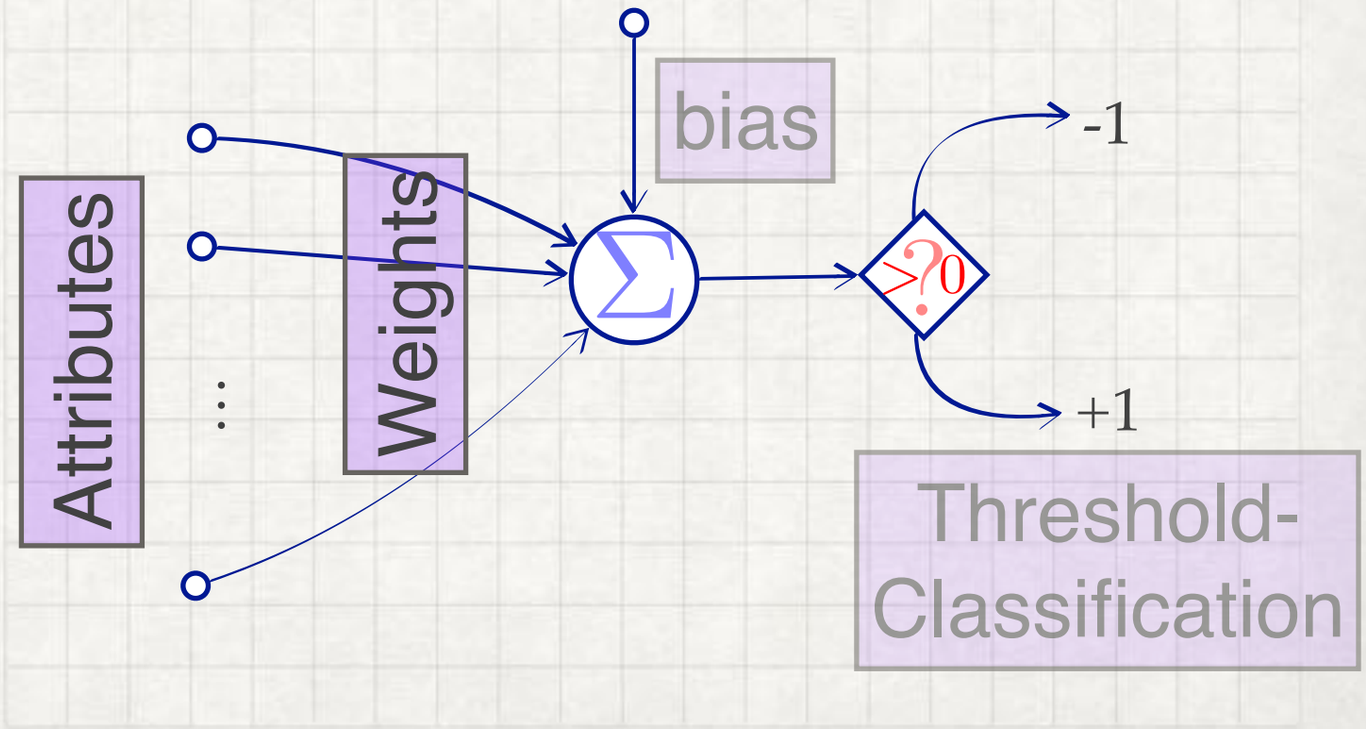
- Rosenblatt, Frank (1957), The Perceptron--a perceiving and recognizing automaton. Report 85-460-1, Cornell Aeronautical Laboratory

PERCEPTRON: THE MODEL

- Inspired by neurons

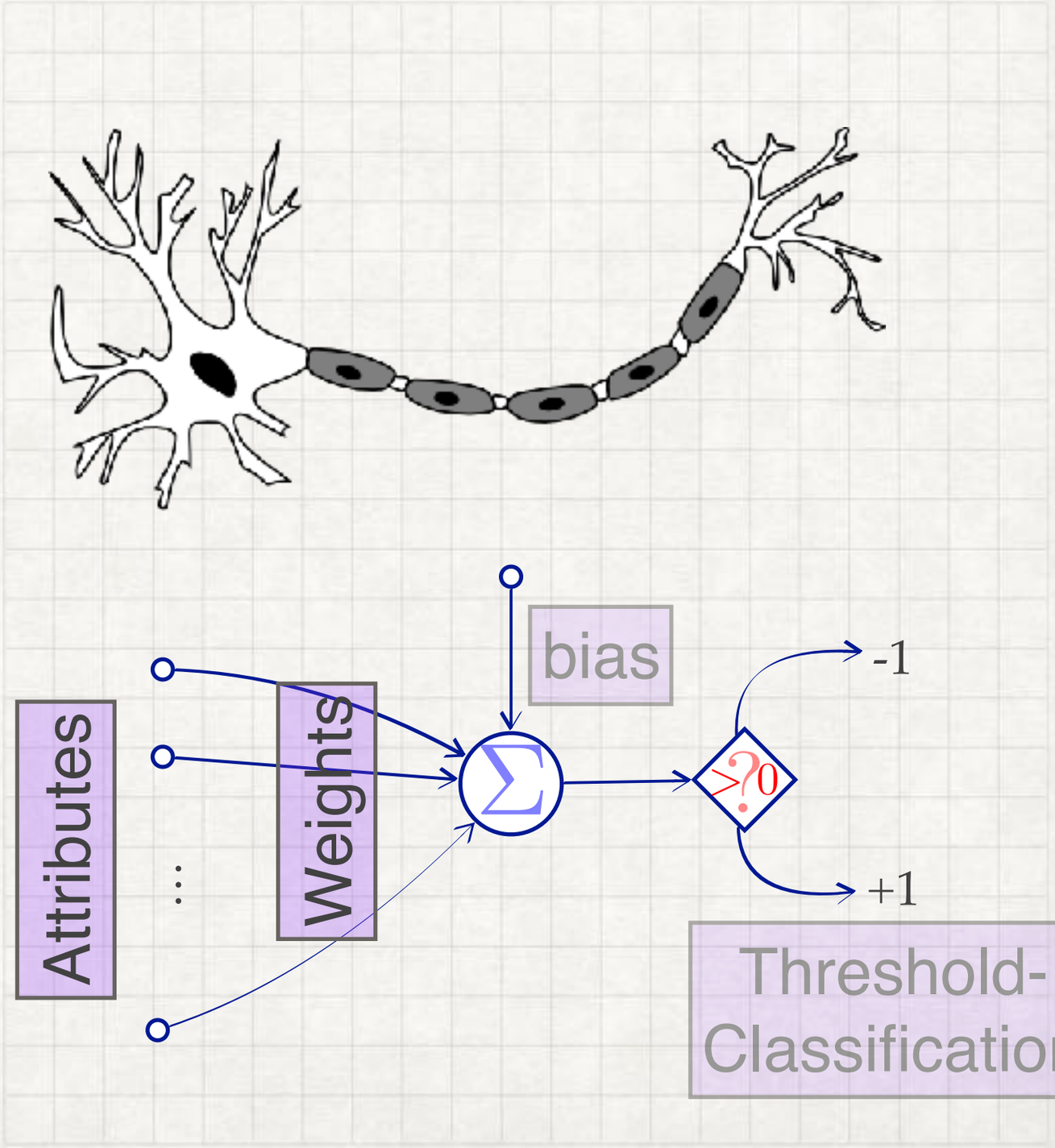


- Making decision by weighted sum of inputs



PERCEPTRON: THE MODEL

- 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*

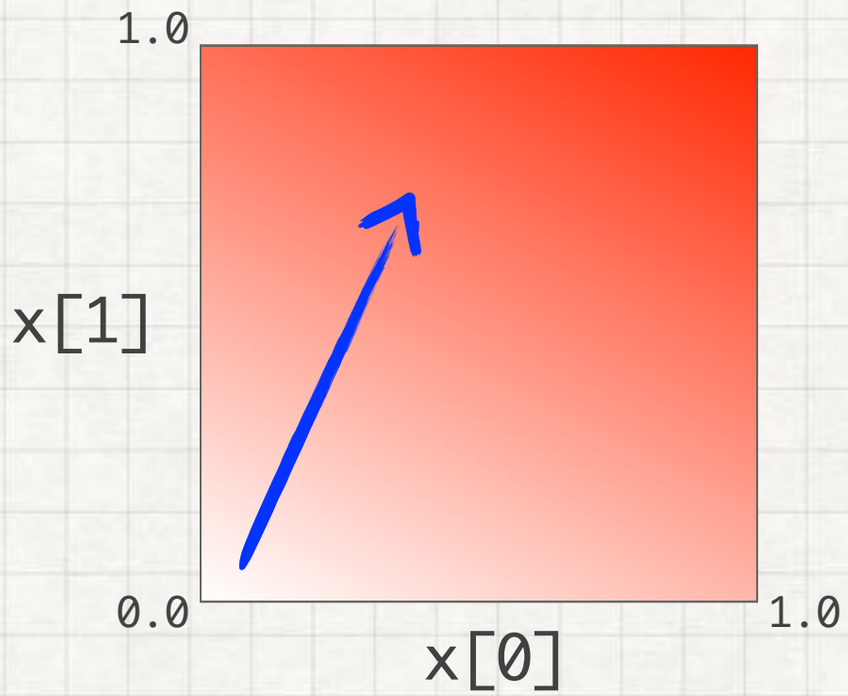
IGNORE BIAS FOR NOW!
You can always choose a suitable threshold after getting a-values with weights.

```
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.

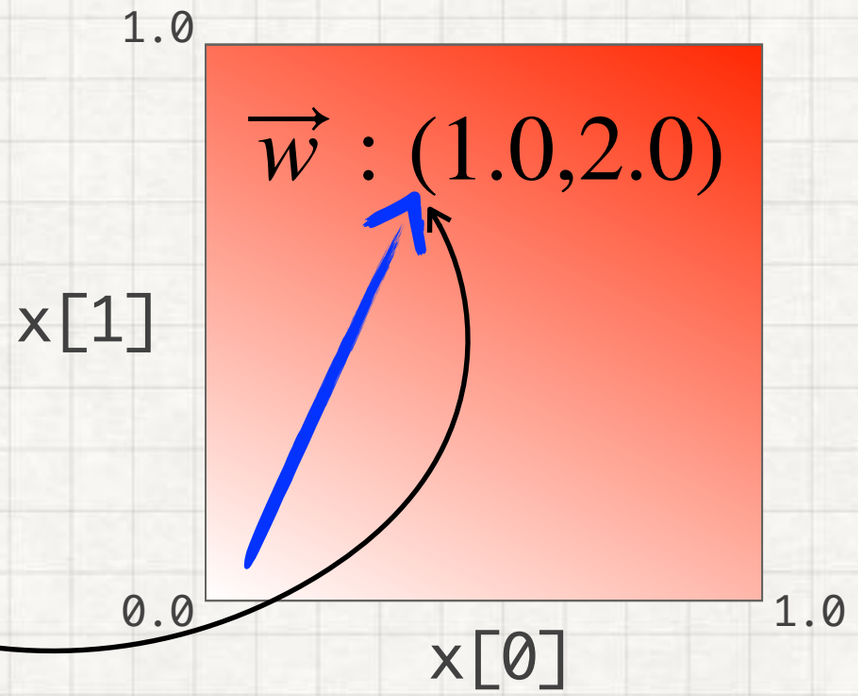
GEOMETRIC INTERPRETATION OF A-VALUES

- 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.



W IN DATA SPACE

```
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.

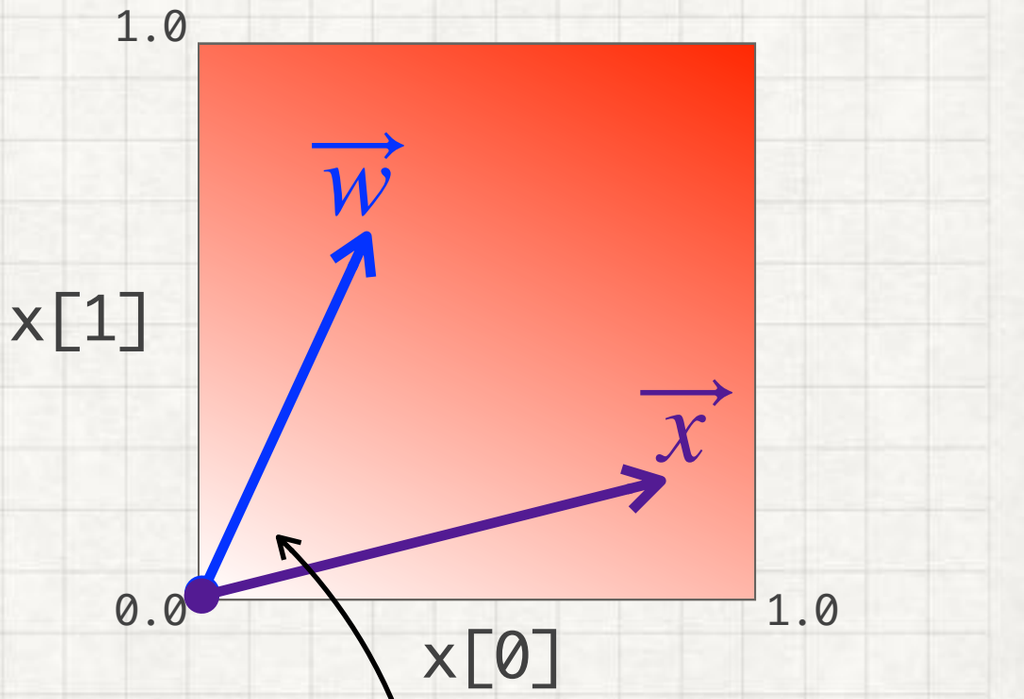
APPLIED TO DATA *SPACE* – NOT DATA *SAMPLES*

```
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.**

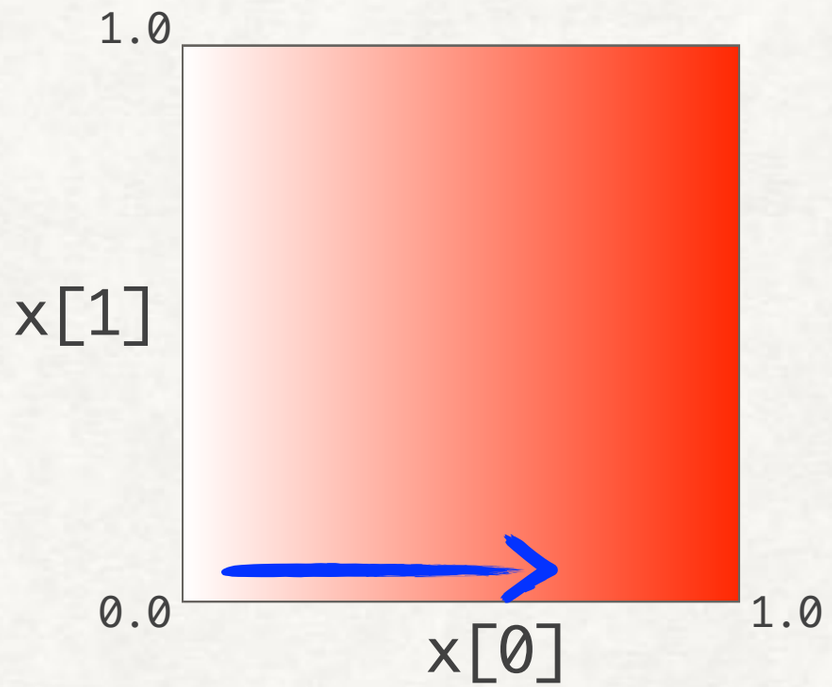


$$w_0 x_0 + w_1 x_1 = \langle \vec{w}, \vec{x} \rangle$$

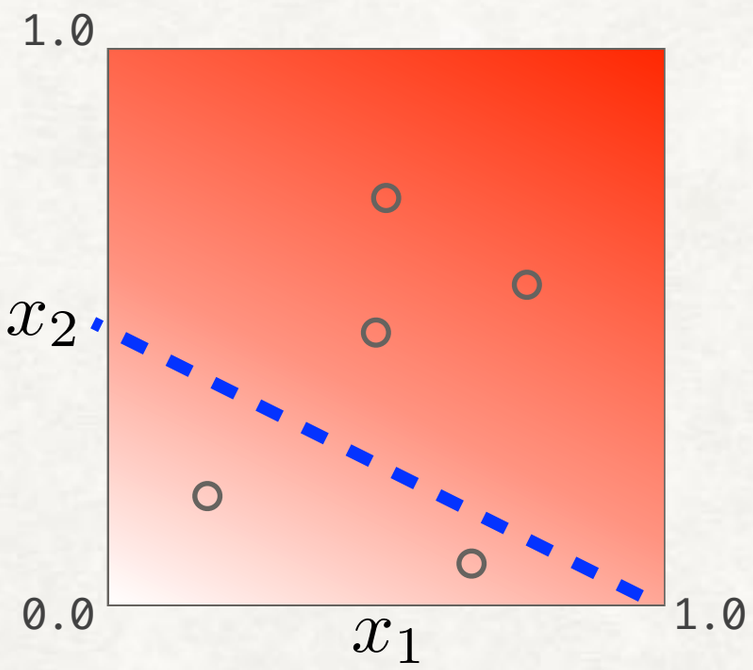
ANOTHER EXAMPLE

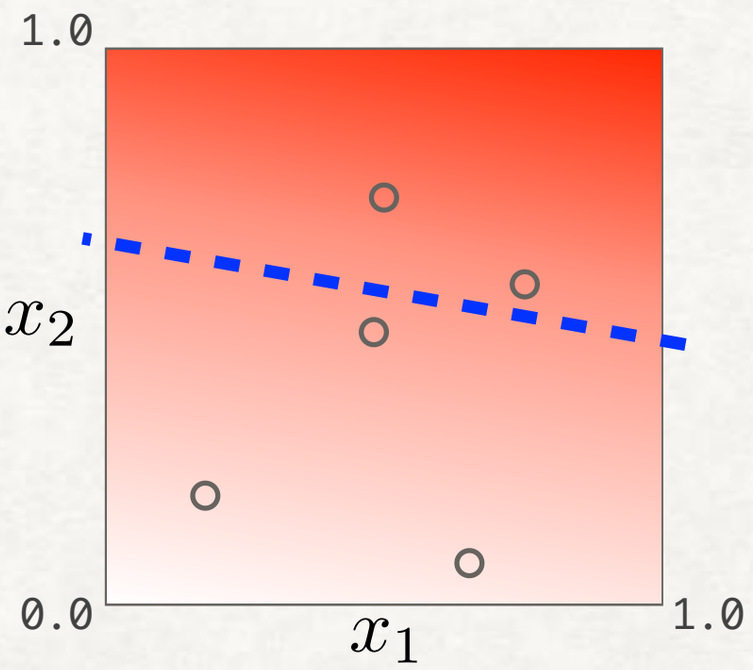
```
w[0] = 1.0
w[1] = 0.0
a = 1.0 * x[0]
```

- Now colours have nothing todo with x[1]



CLASSIFICATION BY THRESHOLDING

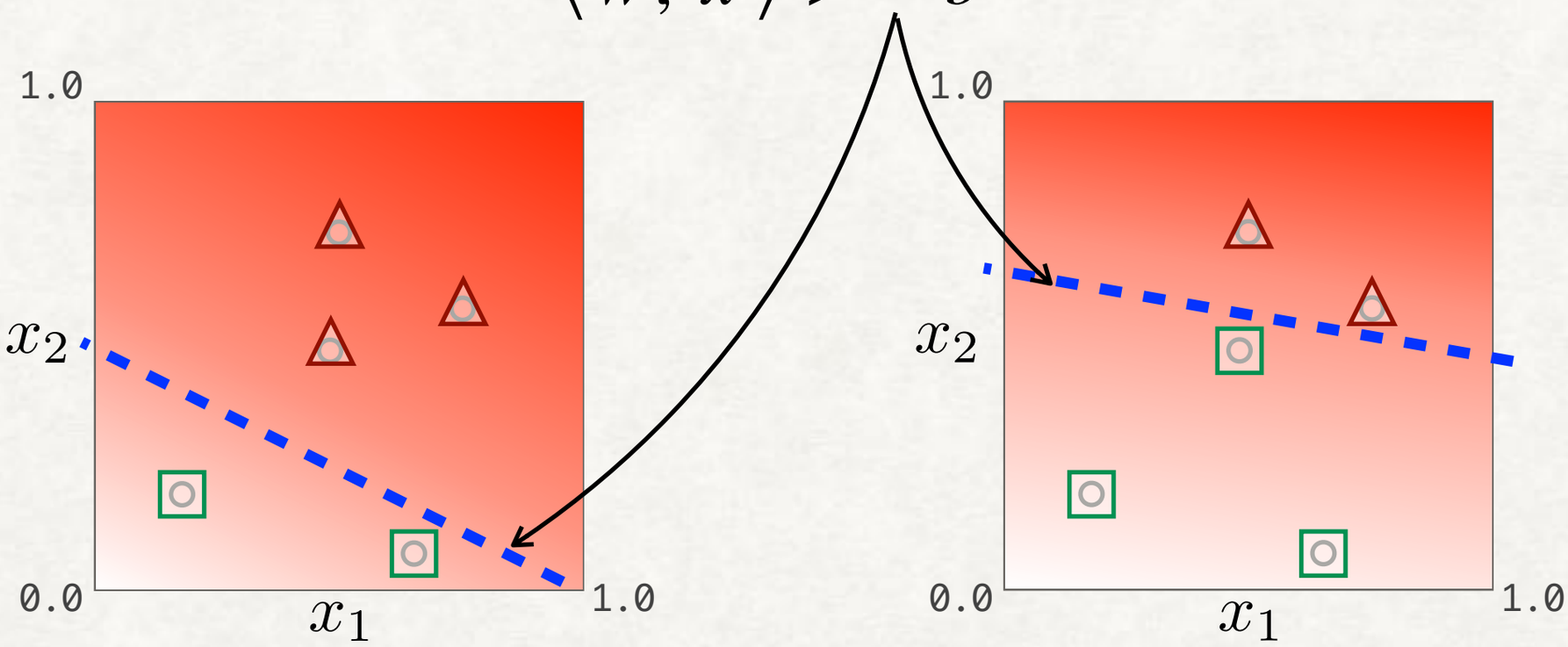




CLASSIFICATION BY THRESHOLDING

$$\langle \vec{w}, \vec{x} \rangle + b > 0$$

$$\langle \vec{w}, \vec{x} \rangle > -b$$



- 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.

MOD*:MORE ON PERCEPTRON AND GENERALISED LINEAR MODELS

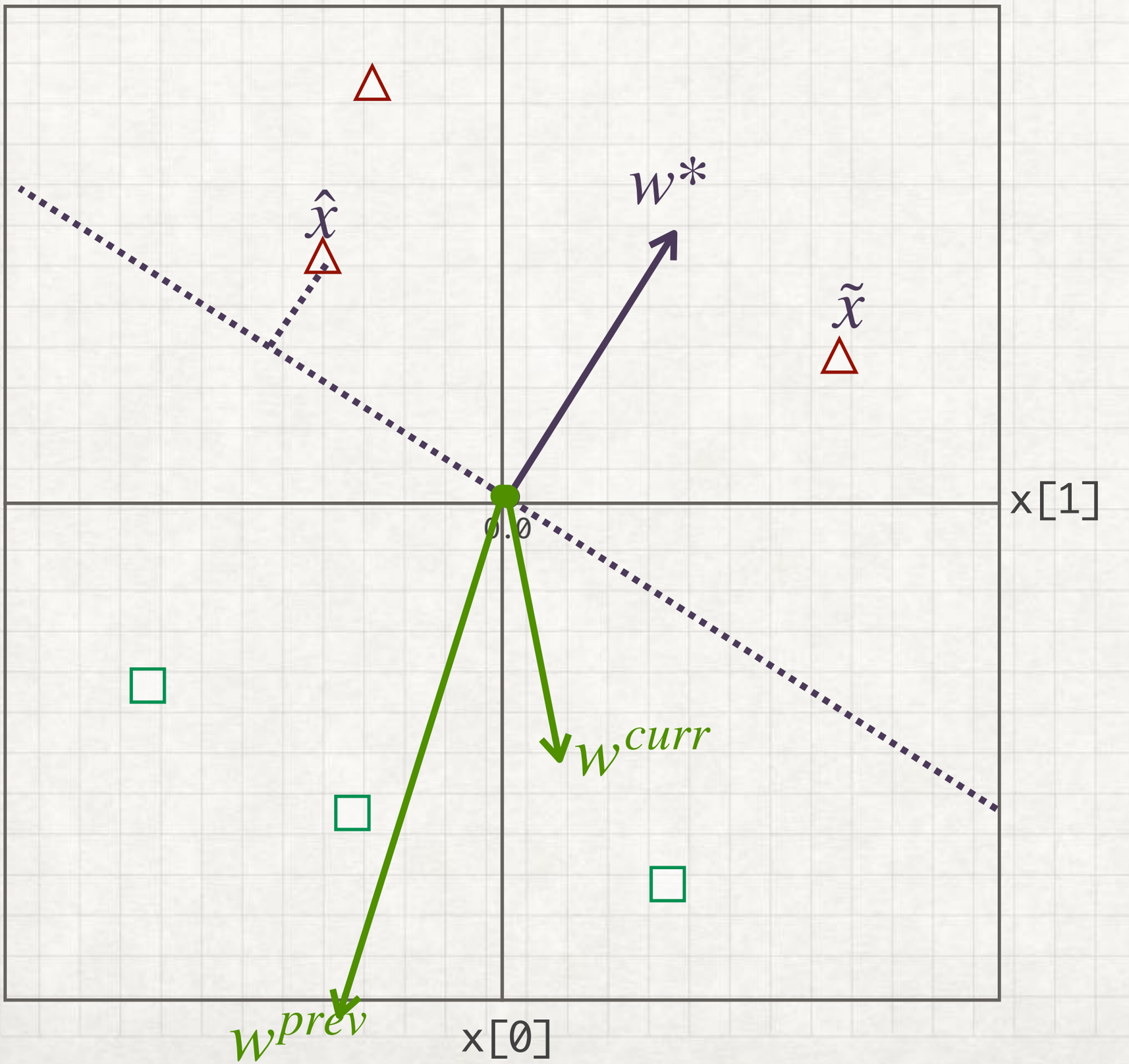
PERCEPTRON MUST WORK!

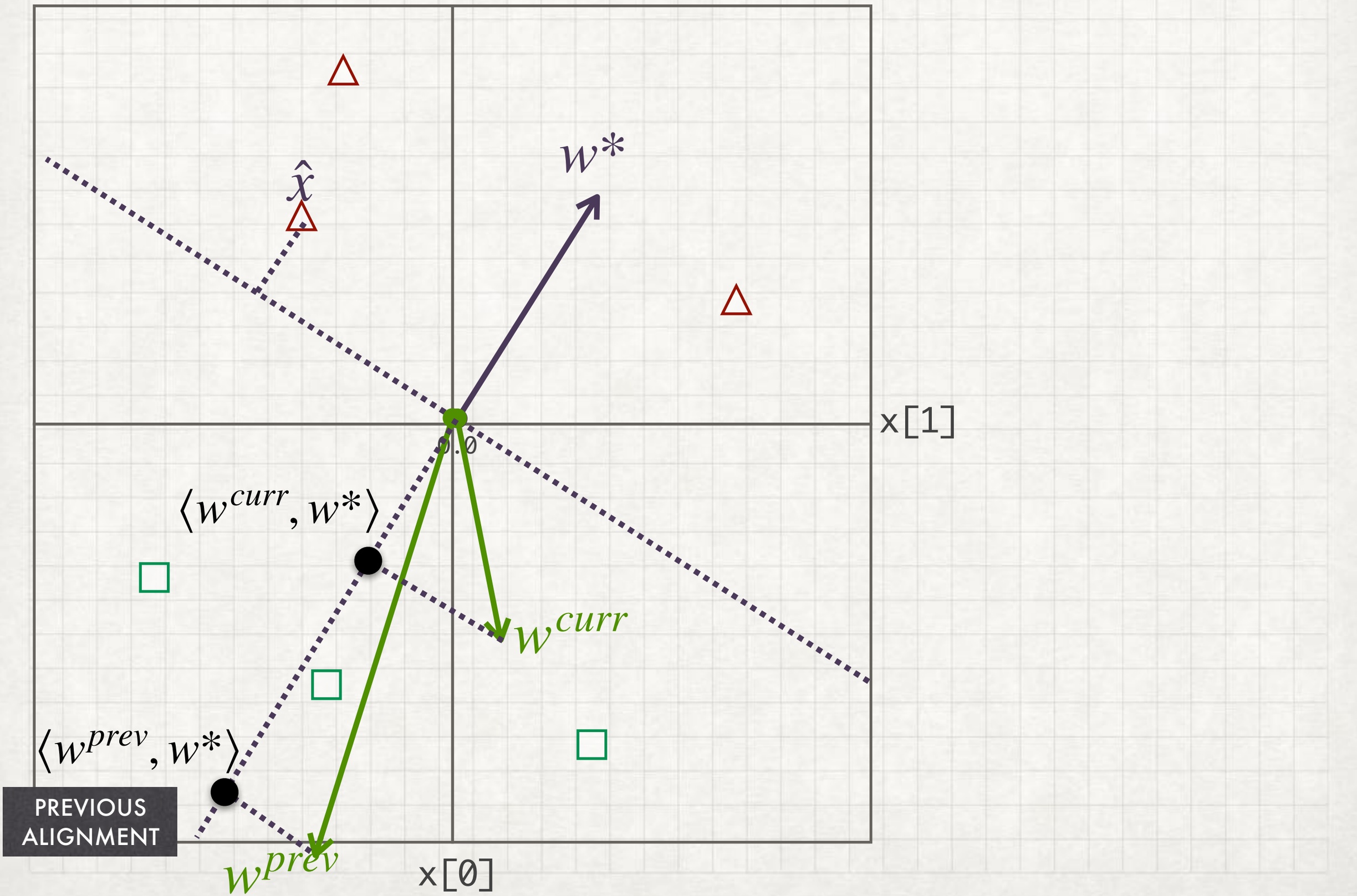
- It is guaranteed that if there is some {W} that gives zero classification error (if there exists one such W, almost certainly there are more, why?), the training approach will achieve zero-classification error.

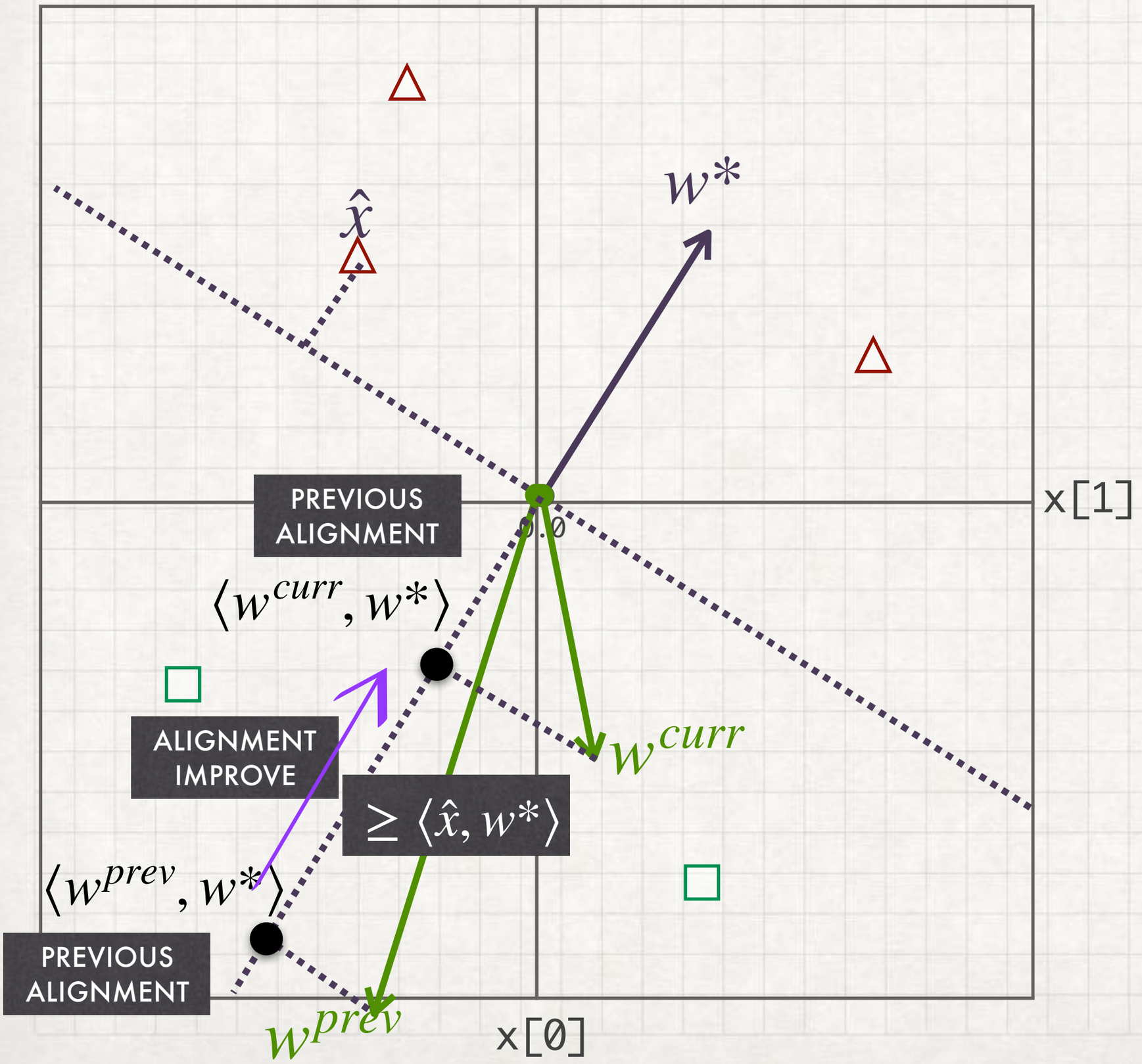
- Sketch of proof
  - Consider such an error free W*
  - Consider the minimum “y-flipped a-value” among all x by W*, must be some p > 0, (why?)
  - Each training step will improve the alignment of the W of question to W* by at least p.
  - Eventually, W and W* will be aligned sufficiently that W makes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

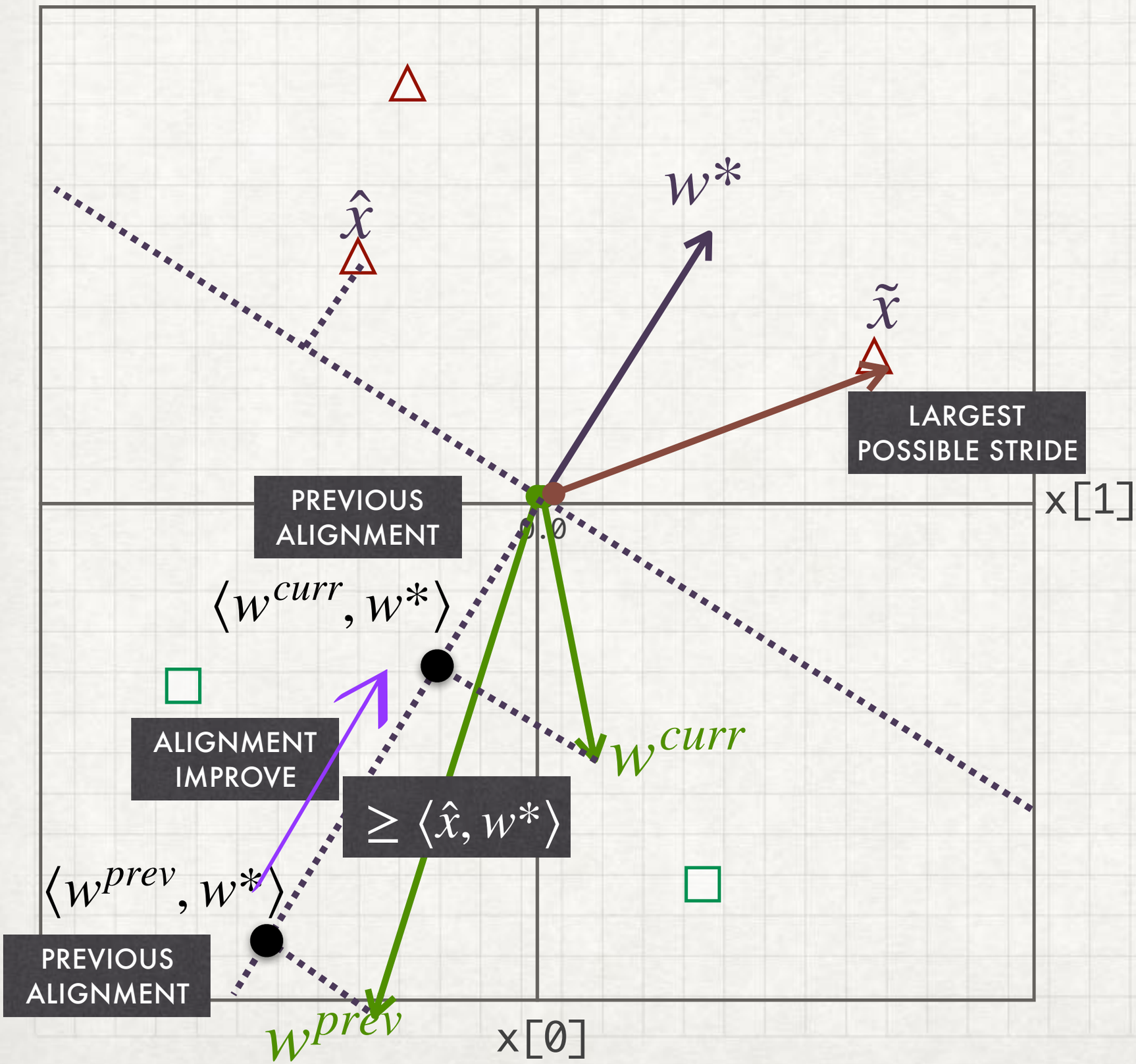
- Original proof by the guy we met above

- 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.


w^*
$\hat{x}$
$\tilde{x}$
x[1]
0.0
w^{curr}
w^{prev}
x[0]


$\hat{x}$
w^*
x[1]
0.0
$\langle w^{curr}, w^* \rangle$
w^{curr}
$\langle w^{prev}, w^* \rangle$
PREVIOUS ALIGNMENT
w^{prev}
x[0]


w^*
$\hat{x}$
PREVIOUS ALIGNMENT
0.0
x[1]
$\langle w^{curr}, w^* \rangle$
ALIGNMENT IMPROVE
w^{curr}
$\geq \langle \hat{x}, w^* \rangle$
$\langle w^{prev}, w^* \rangle$
PREVIOUS ALIGNMENT
w^{prev}
x[0]


w^*
$\hat{x}$
$\tilde{x}$
LARGEST POSSIBLE STRIDE
PREVIOUS ALIGNMENT
0.0
x[1]
$\langle w^{curr}, w^* \rangle$
ALIGNMENT IMPROVE
$\geq \langle \hat{x}, w^* \rangle$
w^{curr}
$\langle w^{prev}, w^* \rangle$
PREVIOUS ALIGNMENT
w^{prev}
x[0]

$\hat{x}$

w^*

$\tilde{x}$

LARGEST POSSIBLE STRIDE

PREVIOUS ALIGNMENT

0.0

x[1]

$\langle w^{curr}, w^* \rangle$

ALIGNMENT IMPROVE

$\geq \langle \hat{x}, w^* \rangle$

w^{curr}

$\langle w^{prev}, w^* \rangle$

PREVIOUS ALIGNMENT

w^{prev}

x[0]

$$\left[\frac{\langle \hat{x}, w^* \rangle}{\|\tilde{x}\|}\right]^2$$

MIN IMPROVE OF

$$\left[\frac{\langle w, w^* \rangle}{\|w\|}\right]^2$$

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