

# THE PERCEPTRON

Eitan Kosman



# AGENDA

- **The Problem**
  - Linear Separability – Def. 1
  - Linear Separability – Def. 2
- **A Biological Neuron**
  - Structure
  - A Mathematical model
- **The Solution – Perceptron**
  - A brief history
  - The algorithm
  - Intuitive interpretation for weights update
  - Theorem: Mistake bound
  - The fall of perceptron
  - Stopping criteria
  - Kernel Perceptron



# THE PROBLEM

Given a set of  $n$  points in  $\mathbb{R}^d$  and labels:

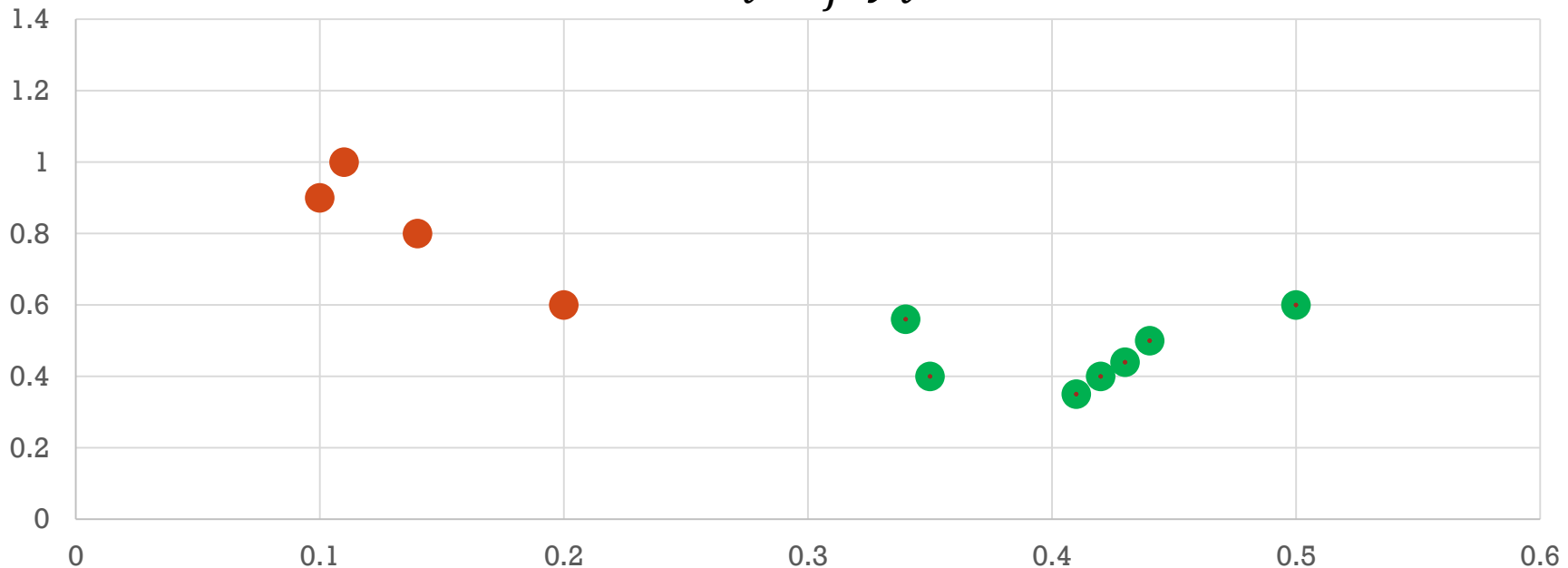
$$X = \{x_i | i \in [n], x \in \mathbb{R}^d\}, Y = \{y_i | i \in [n], y_i \in \{-1, 1\}\}$$

We want to find a transformation:

$$f: X \rightarrow Y$$

s.t.

$$x_i \mapsto_f y_i$$



# DEFINITION (1): LINEAR SEPARABILITY

Let  $X_0, X_1 \subseteq \mathbb{R}^d$  be 2 sets of points.  $X_0, X_1$  are linearly separable if there exist  $n + 1$  real numbers  $w_1, w_2, \dots, w_n, k$  such that:

$$\forall x \in X_0: \sum_{i=1}^n w_i x_i > k$$
$$\forall x \in X_1: \sum_{i=1}^n w_i x_i < k$$

The above terms could also be represented as inner product:  $\langle w, x \rangle$  where:  $w = (w_1, w_2, \dots, w_n)$  and  $x = (x_1, x_2, \dots, x_n)$

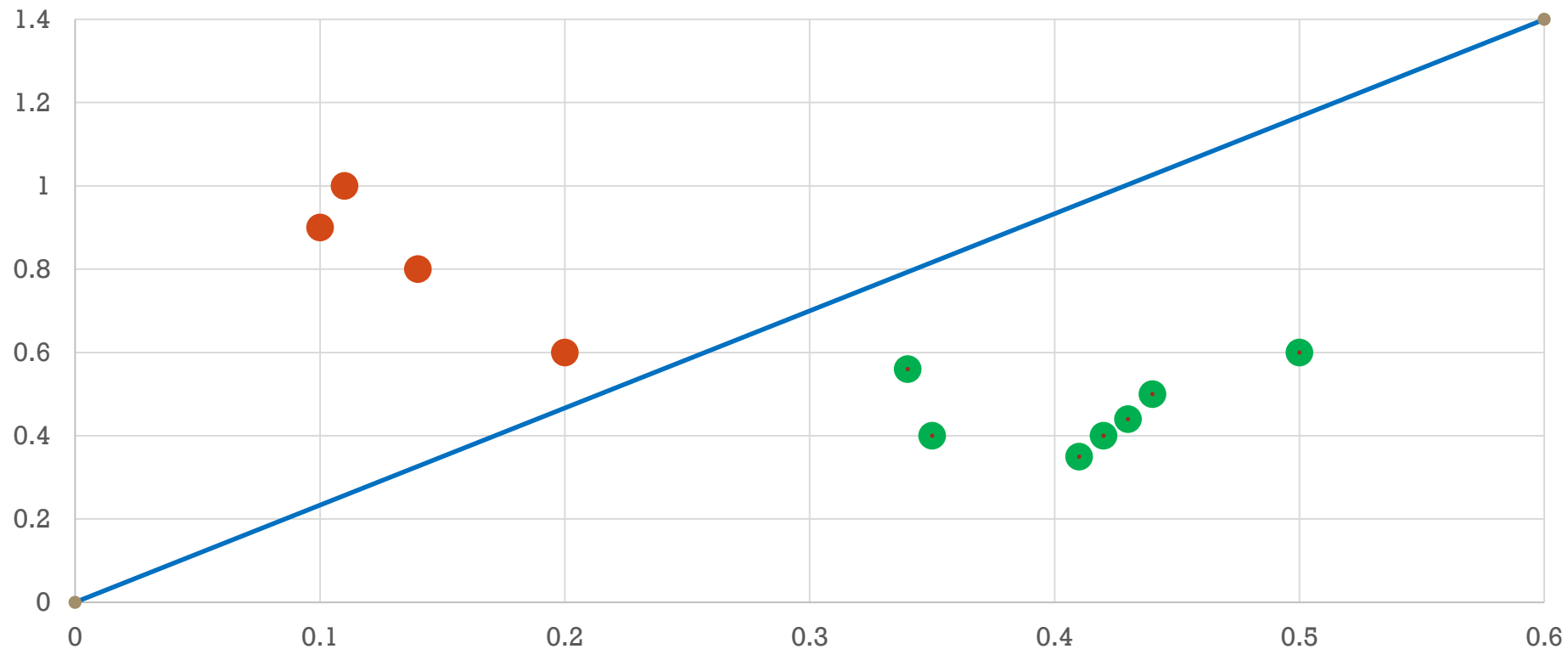


# DEFINITION 1 INTERPRETATION

Given vector  $w$ , we can define a hyper-plane by:

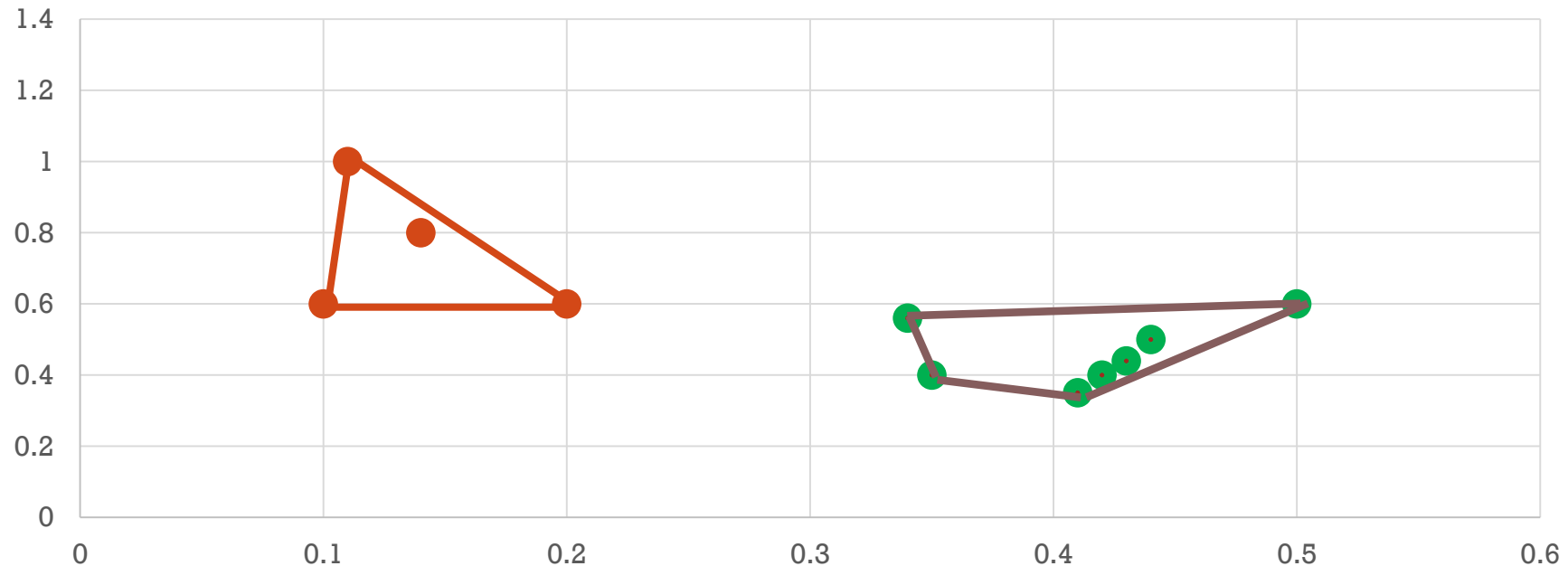
$$\langle w, x \rangle + d = 0$$

Thus, the hyper-plane separates the field into 2 regions such that all points belong to  $X_0$  are in one region and all points belong to  $X_1$  are in the other region.



# DEFINITION (2): LINEAR SEPARABILITY

Let  $X_0, X_1 \subseteq \mathbb{R}^d$  be 2 sets of points.  $X_0, X_1$  are linearly separable precisely when their respective convex hulls are disjoint (do not overlap)



# AGENDA

- The Problem
  - Linear Separability – Def. 1
  - Linear Separability – Def. 2
- The Biological Neuron
  - Structure
  - A Mathematical model
- The Solution – Perceptron
  - A brief history
  - The algorithm
  - Intuitive interpretation for weights update
  - Theorem: Mistake bound
  - The fall of perceptron
  - Stopping criteria
  - Kernel Perceptron

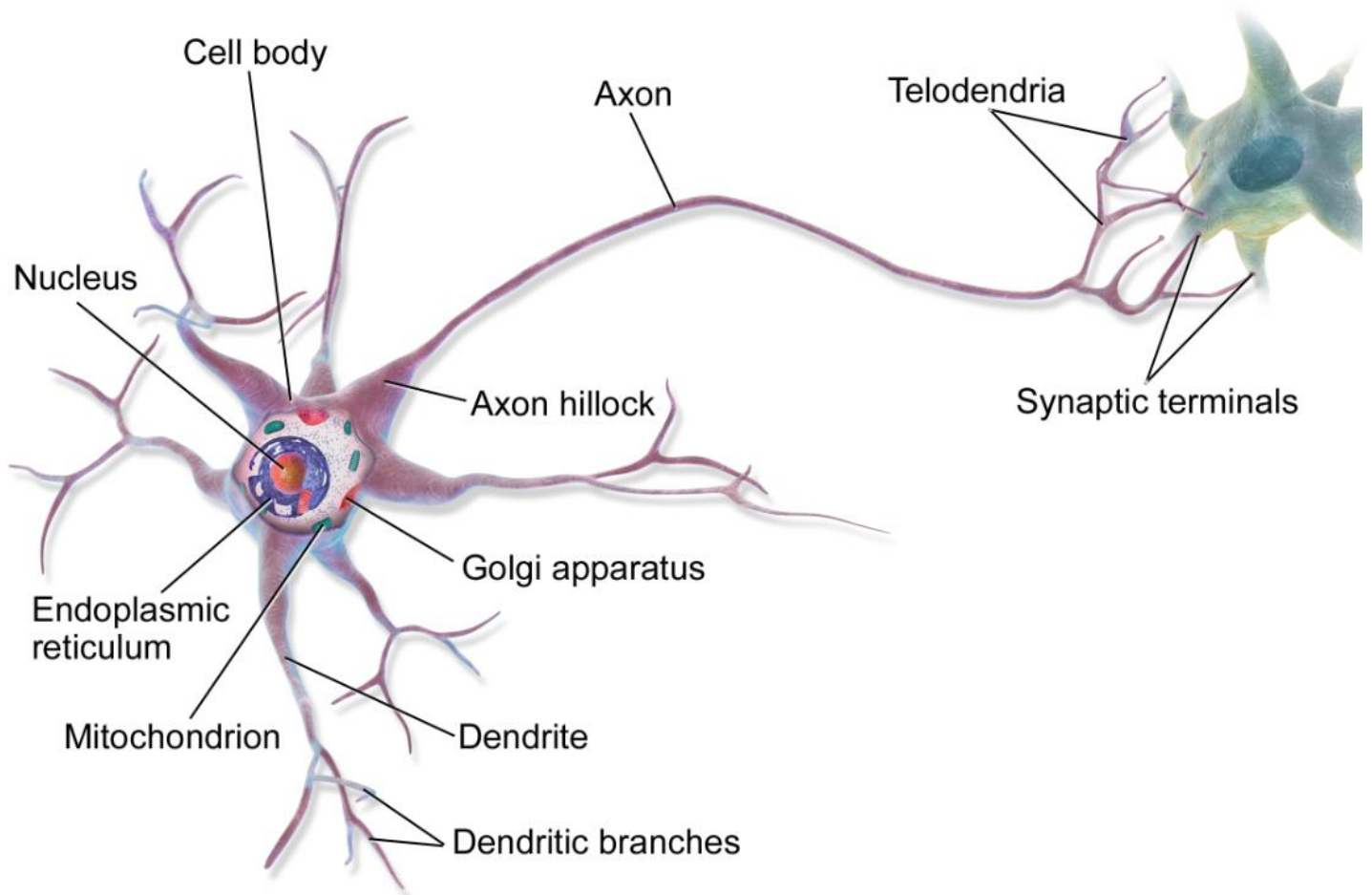


# THE BIOLOGICAL NEURON - STRUCTURE

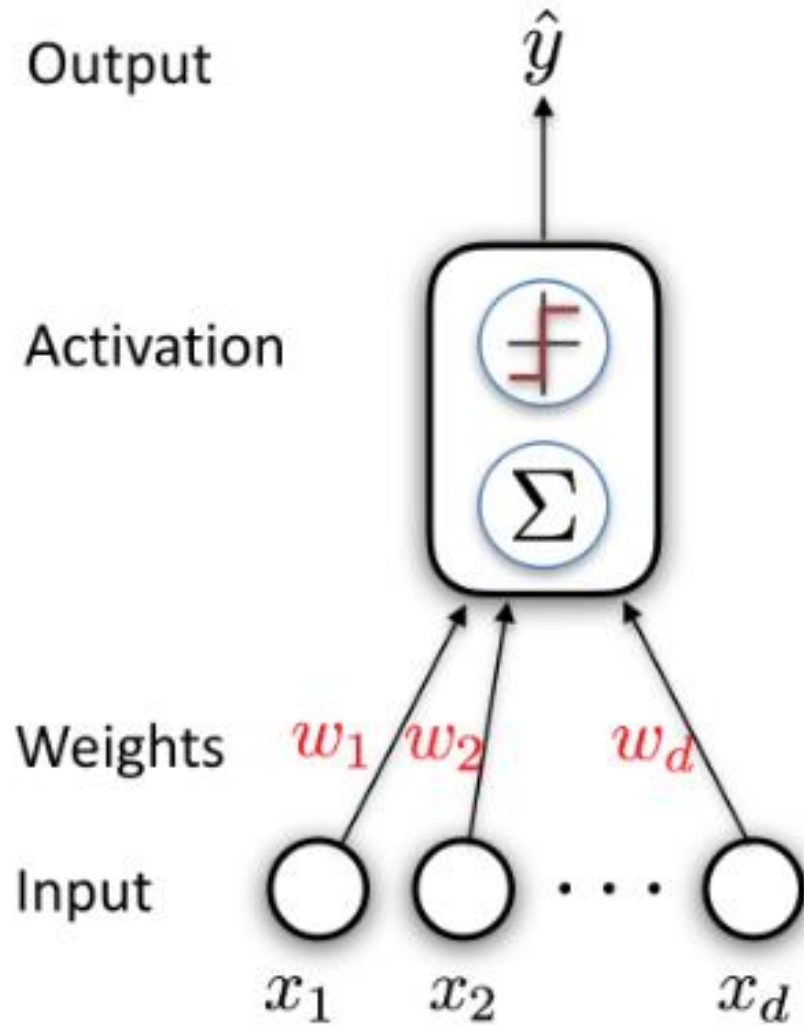
Like any other body cell, the neuron has a cell body which contains a nucleus where the DNA is stored.

From our perspective, the interesting parts are:

- Dendrites – make connections with tens of thousand of other cells; other neurons. They behave as “inputs”.
- Axon – transmits information to different neurons, muscles, and other body cells based on the signals the cell receives. Its signals are received by other cells' dendrites.







# THE BIOLOGICAL NEURON — A MATHEMATICAL MODEL

- We will try to mimic the function of a neuron using mathematical tools. Given an input vector  $x$ :
  - $x$  will be the inputs of the neuron (dendrites).
  - Define a weight,  $w_i$ , for each input, and sum all the multiplications.
  - Output the result as  $\hat{y}$  (Axon)

**There's still a problem – How do we find the weights?**

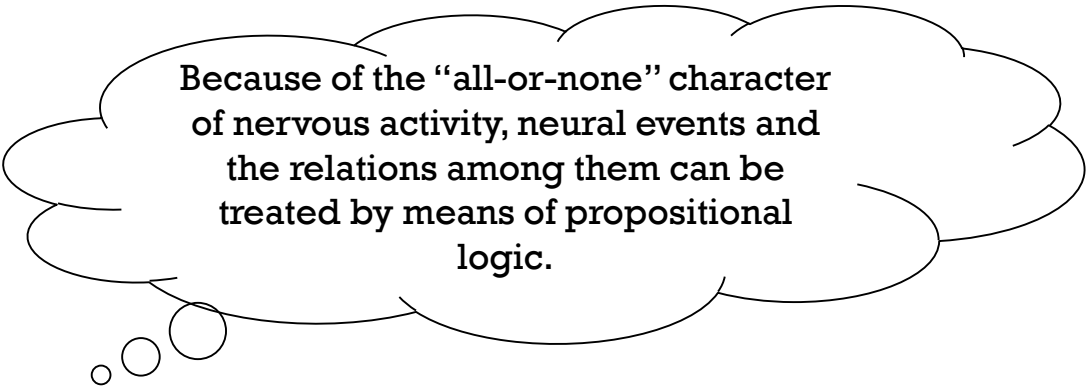


# AGENDA

- The Problem
  - Linear Separability – Def. 1
  - Linear Separability – Def. 2
- The Biological Neuron
  - Structure
  - A Mathematical model
- The Solution – Perceptron
  - A brief history
  - The algorithm
  - Intuitive interpretation for weights update
  - Theorem: Mistake bound
  - The fall of perceptron
  - Stopping criteria
  - Kernel Perceptron



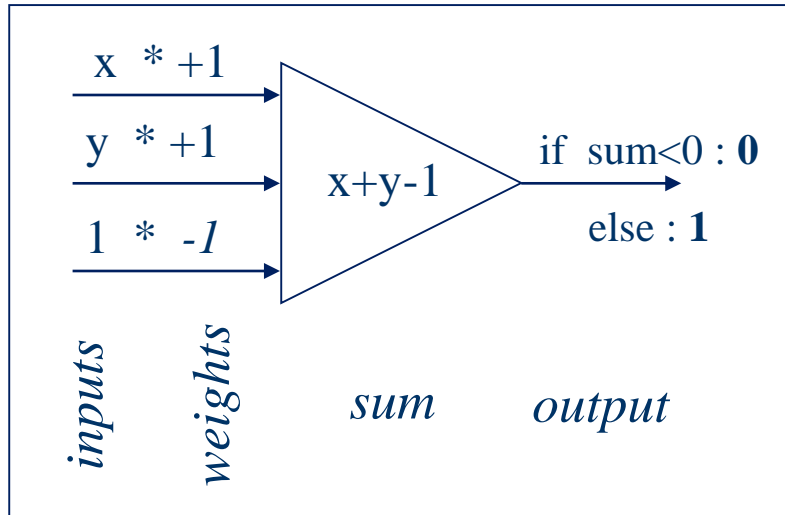
# PREHISTORY



Because of the “all-or-none” character of nervous activity, neural events and the relations among them can be treated by means of propositional logic.

- W.S. McCulloch & W. Pitts (1943). “A logical calculus of the ideas immanent in nervous activity”, Bulletin of Mathematical Biophysics, 5, 115-137
- This paper pointed out that simple artificial “neurons” could be made to perform basic logical operations such as AND, OR and NOT
- They attempted to demonstrate that a Turing machine program could be implemented in a finite network of formal neurons, the base logic unit of the brain.

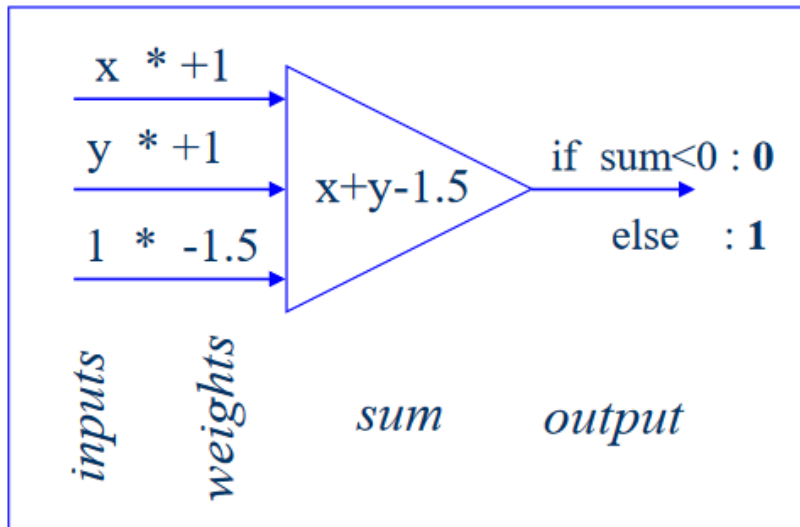




**Truth Table for Logical OR**

$x$	$y$	$x \mid y$
0	0	0
0	1	1
1	0	1
1	1	1

*inputs* *output*



**Truth Table for Logical AND**

$x$	$y$	$x \& y$
0	0	0
0	1	0
1	0	0
1	1	1

*inputs* *output*



# 1958 — THE PERCEPTRON

*Psychological Review*  
Vol. 65, No. 6, 1958

## THE PERCEPTRON: A PROBABILISTIC MODEL FOR INFORMATION STORAGE AND ORGANIZATION IN THE BRAIN<sup>1</sup>

F. ROSENBLATT

*Cornell Aeronautical Laboratory*



# NEW NAVY DEVICE LEARNS BY DOING

---

Psychologist Shows Embryo  
of Computer Designed to  
Read and Grow Wiser

---

WASHINGTON, July. 7 (UPI)  
—The Navy revealed the embryo of an electronic computer today that it expects will be able to walk, talk, see, write, reproduce itself and be conscious of its existence.



# PERCEPTRON — THE ALGORITHM

- The goal is to find a hyper-plane separating 2 known classes.
- Consider definition (1) for linear separability:

$$\forall x \in X_0: \langle w, x \rangle > k$$

$$\forall x \in X_1: \langle w, x \rangle < k$$

$\Downarrow$

$$\forall x \in X_0: \langle w, x \rangle - k > 0$$

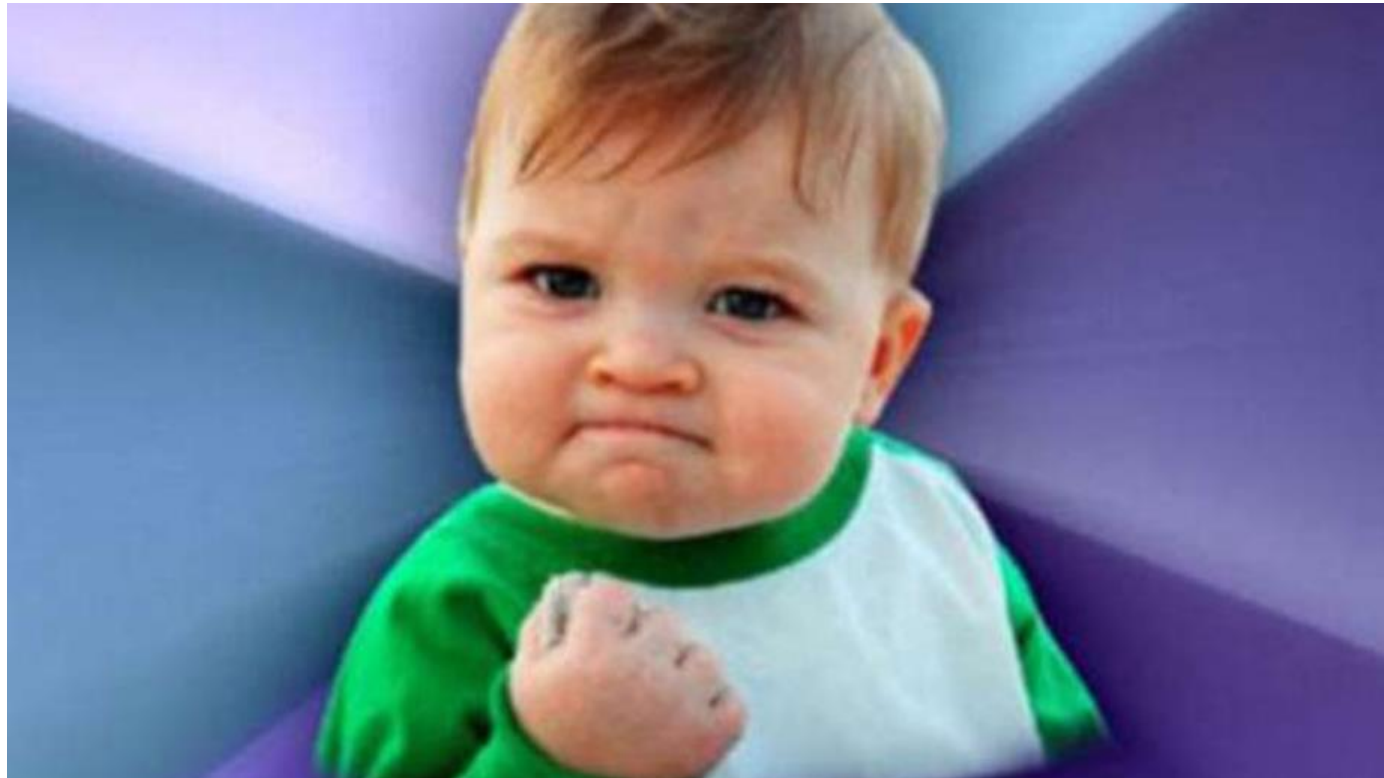
$$\forall x \in X_1: \langle w, x \rangle - k < 0$$



We can eliminate  $k$  by augmenting representation with one dimension:

$$\begin{aligned}x' &= (x, 1) \\ w' &= (w, -k)\end{aligned}$$

$$\langle w', x' \rangle = (w, -k) \begin{pmatrix} x \\ 1 \end{pmatrix} = w \cdot x - k$$





# PERCEPTRON — THE ALGORITHM

$P \leftarrow \text{inputs with label } +1$

$N \leftarrow \text{inputs with label } -1$

$w \leftarrow \bar{0}$

*while a stopping criterion isn't met:*

*pick  $(x, y) \in X \times Y$*

$\hat{y} = \text{sign}(\langle w, x \rangle)$

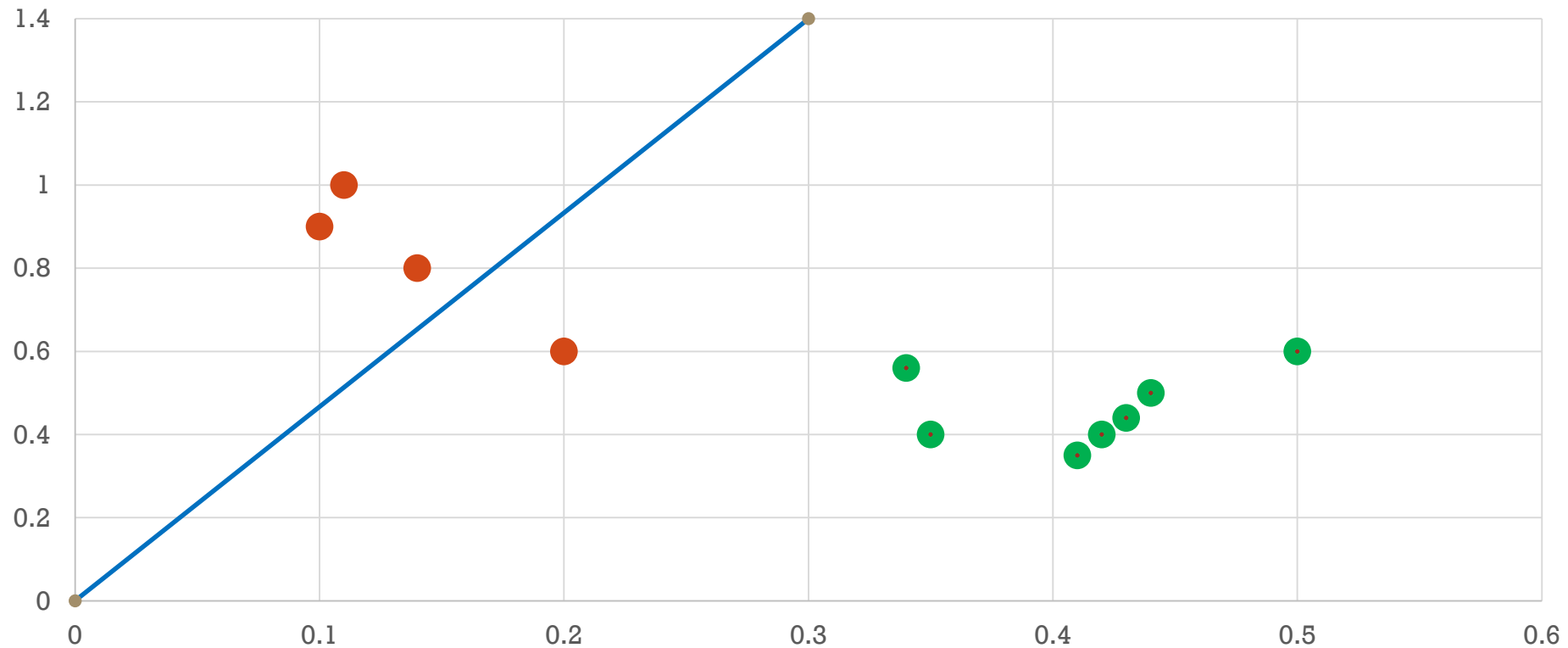
*if  $\hat{y} \neq y$ :*

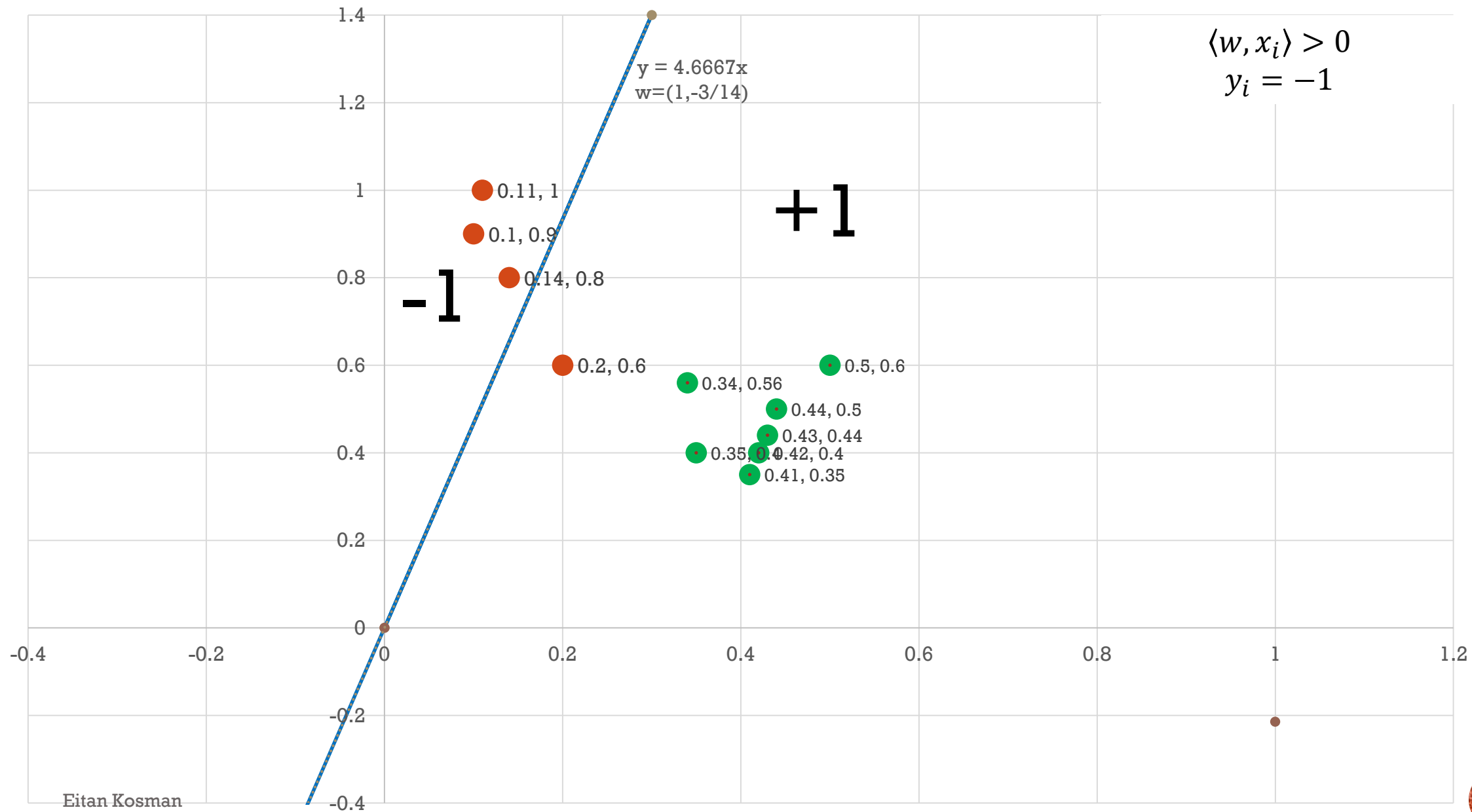
$w \leftarrow w + yx$

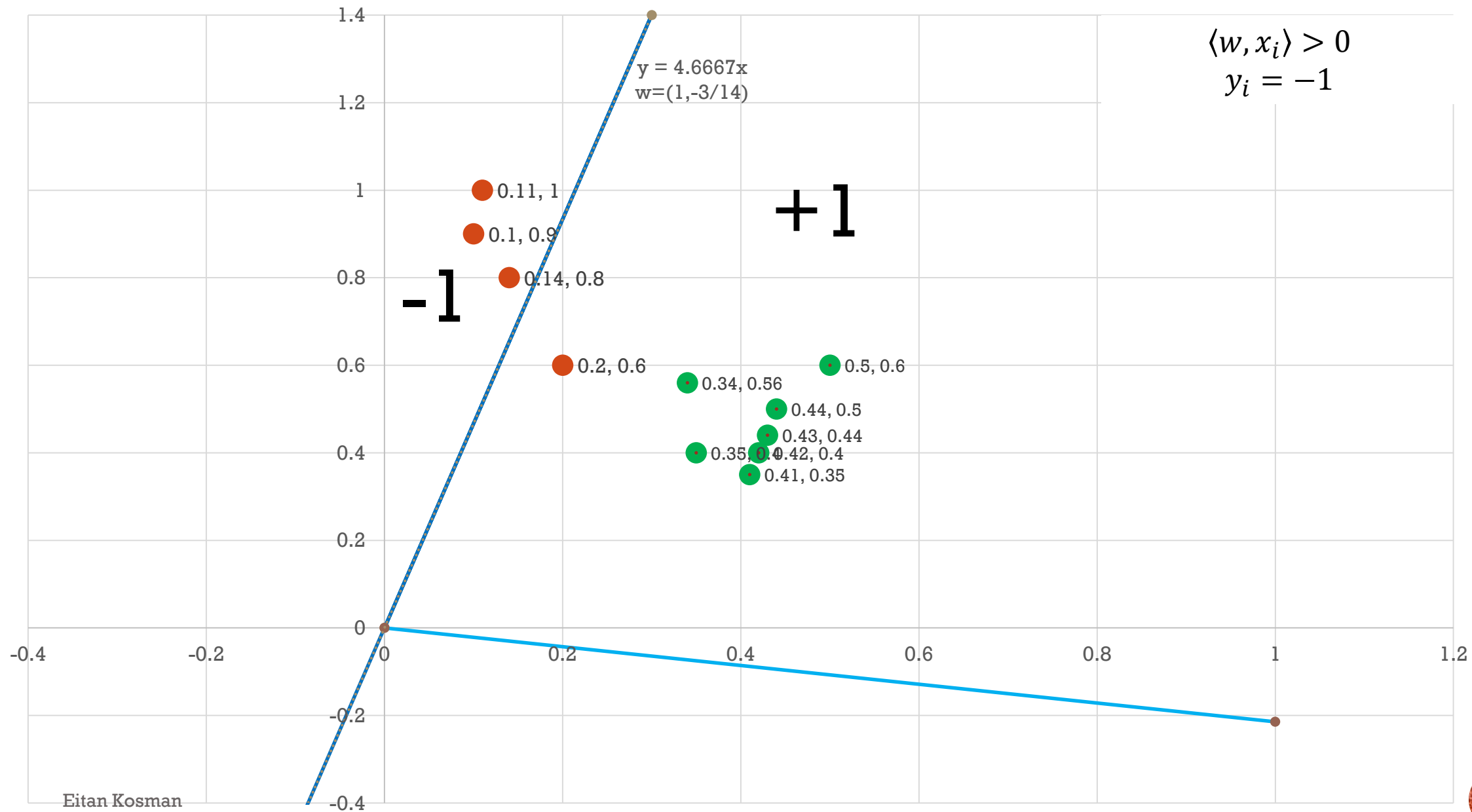


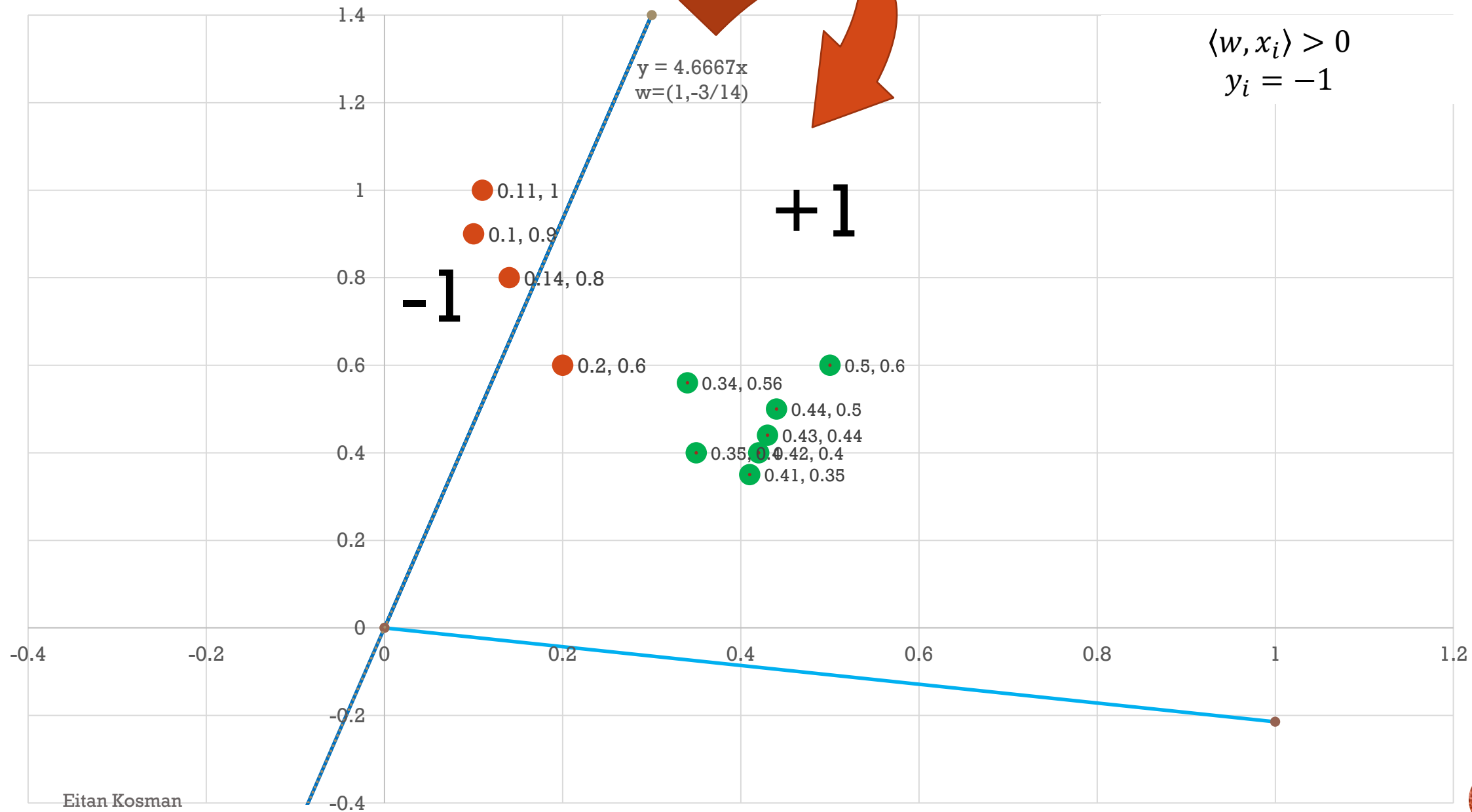
# WEIGHTS UPDATE : INTUITION

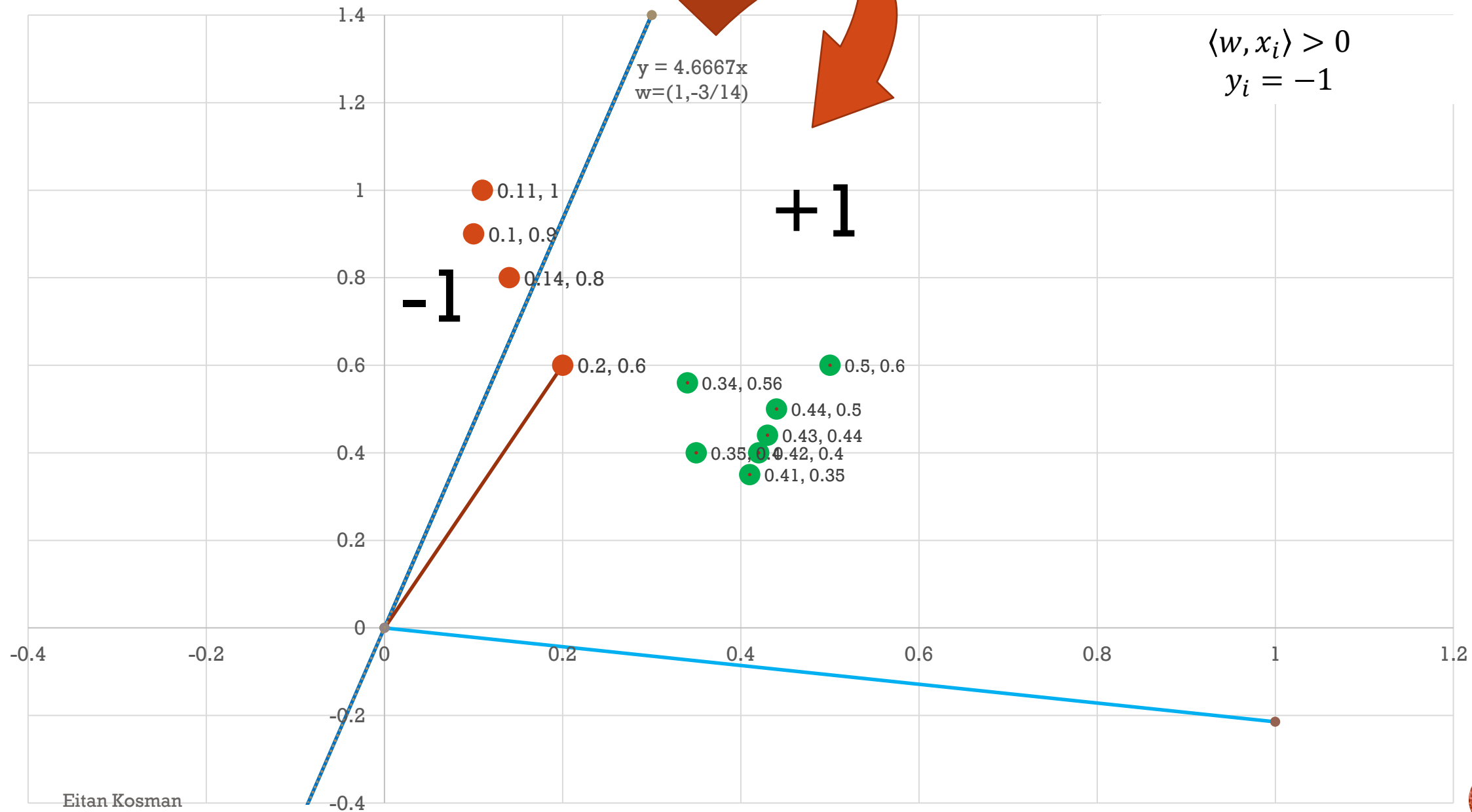
The orange points are from class -1 and the green points are from class +1.  
How would we update the decision line so that it classifies all the points correctly?

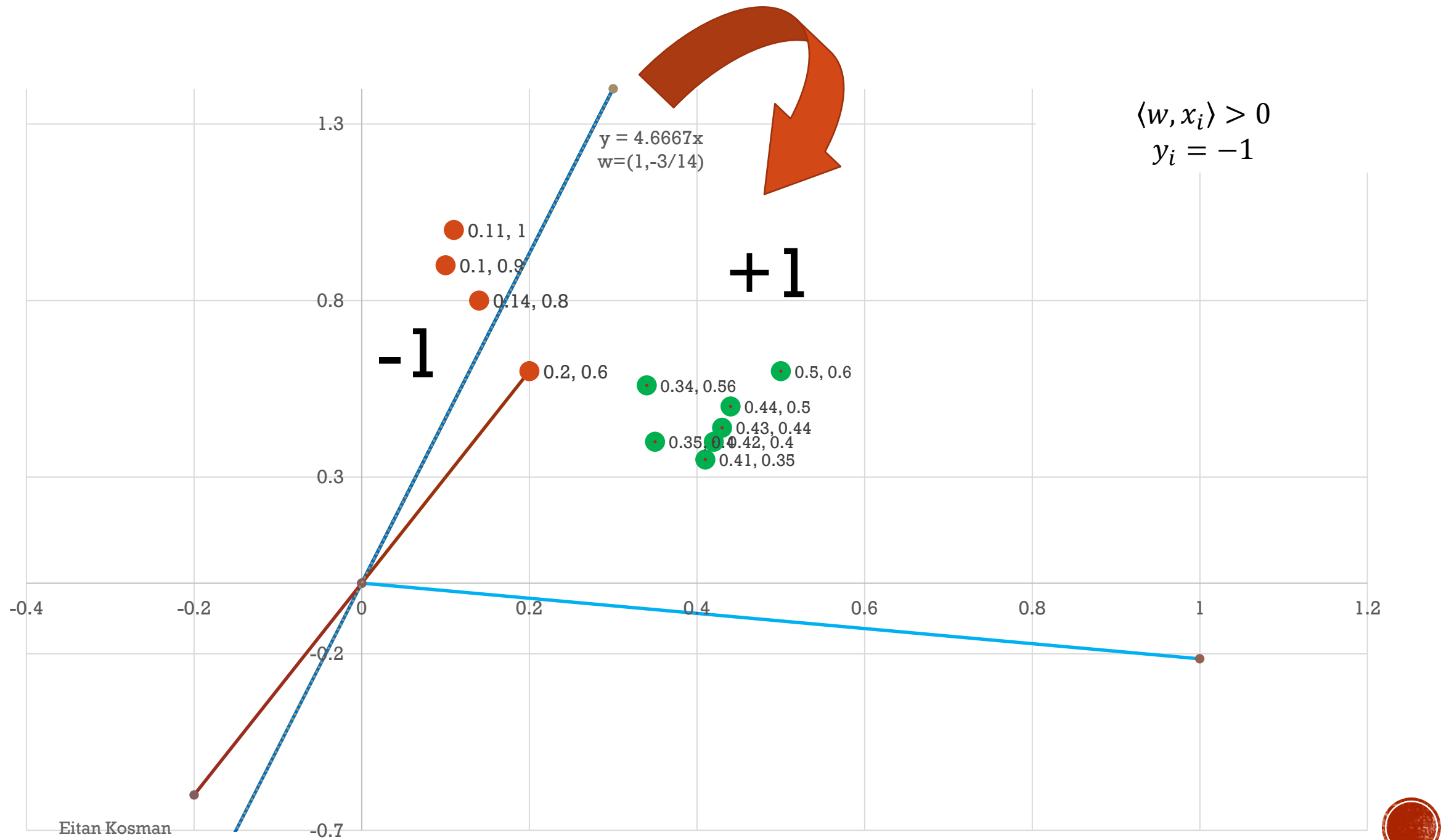


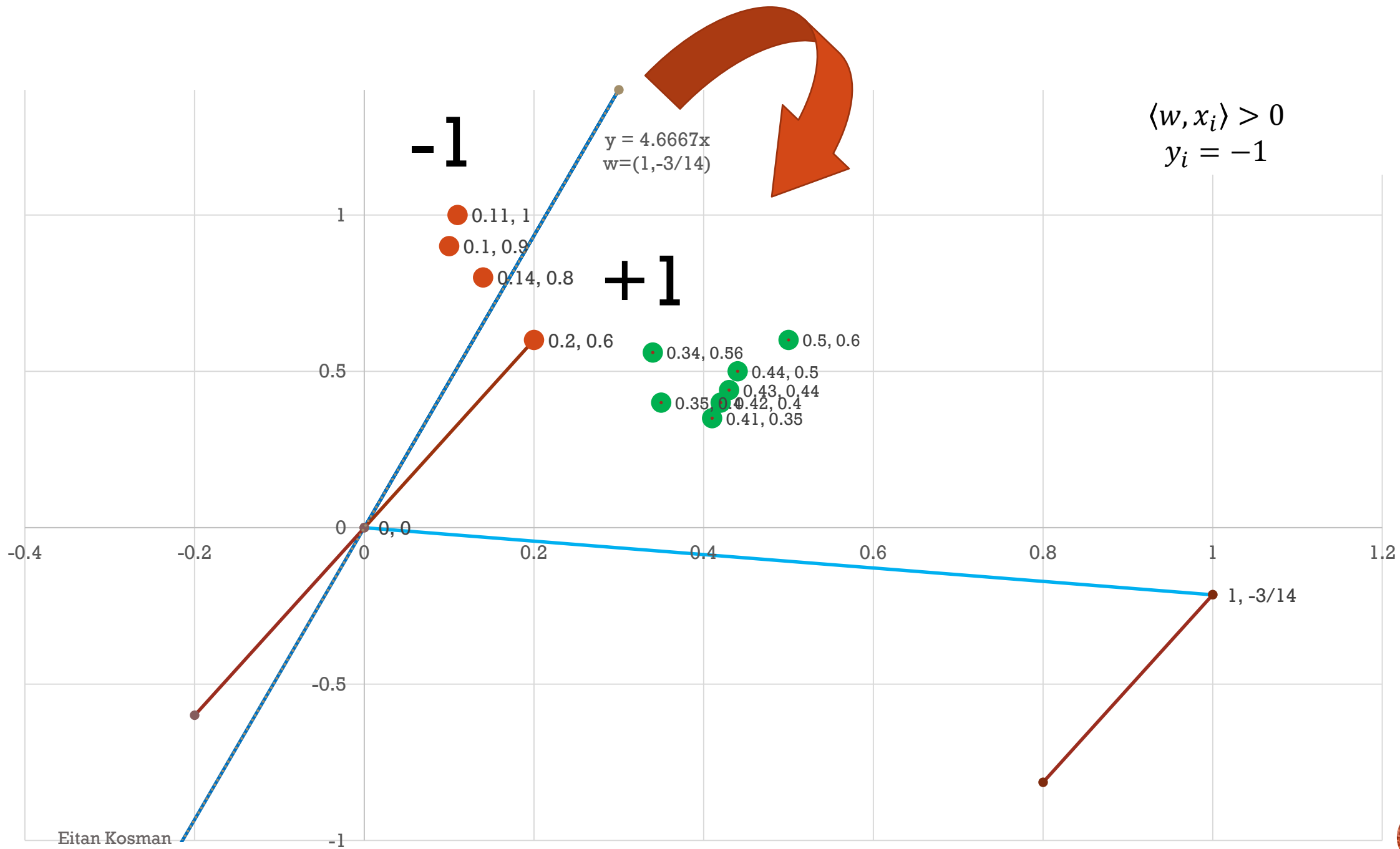




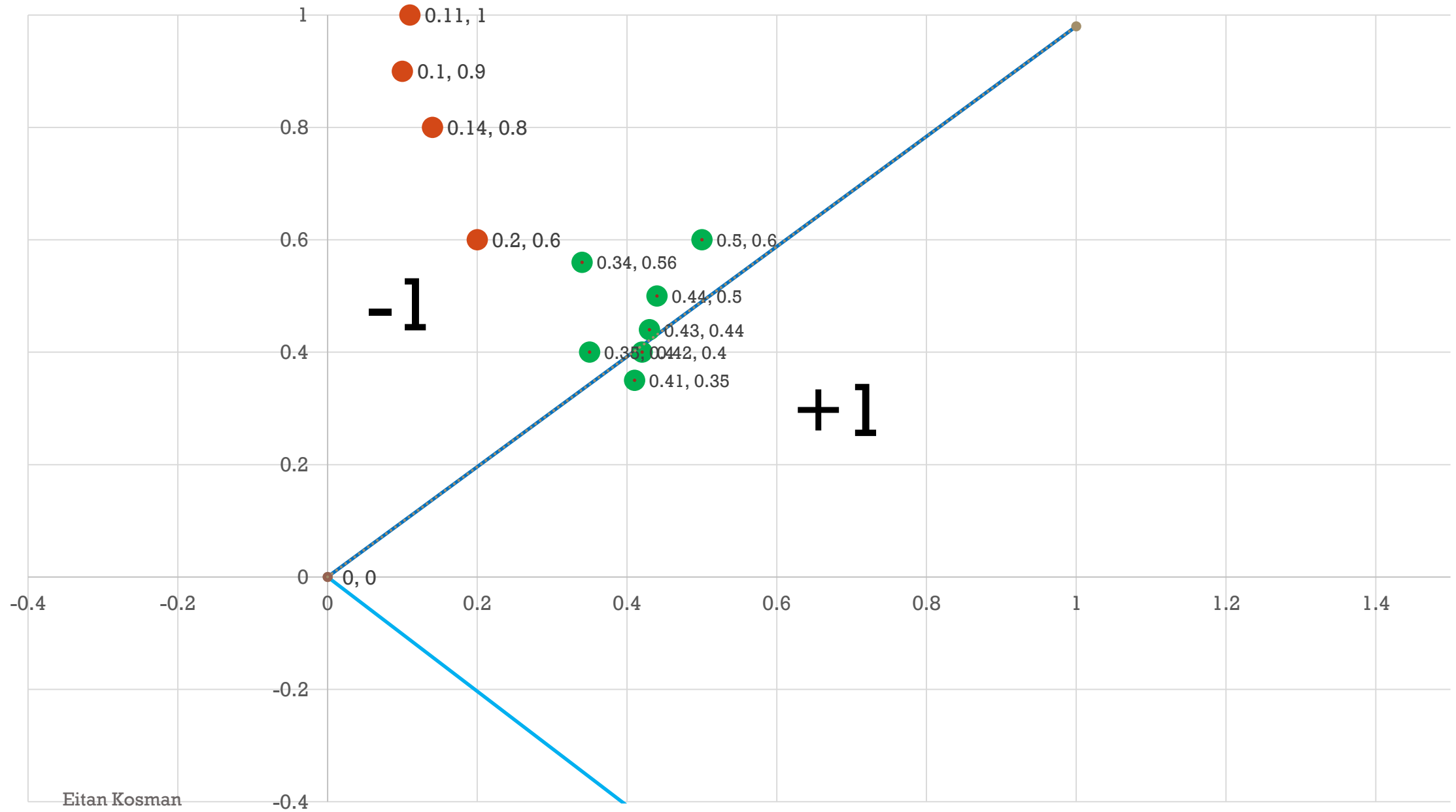












# MISTAKE BOUND

Theorem:

Let  $(x_1, y_1), \dots, (x_n, y_n)$ , where  $x_i \in \mathbb{R}^N$  and  $y_i \in \{-1, 1\}$  be a sequence of labeled examples and assume it is linearly separable.

Denote:

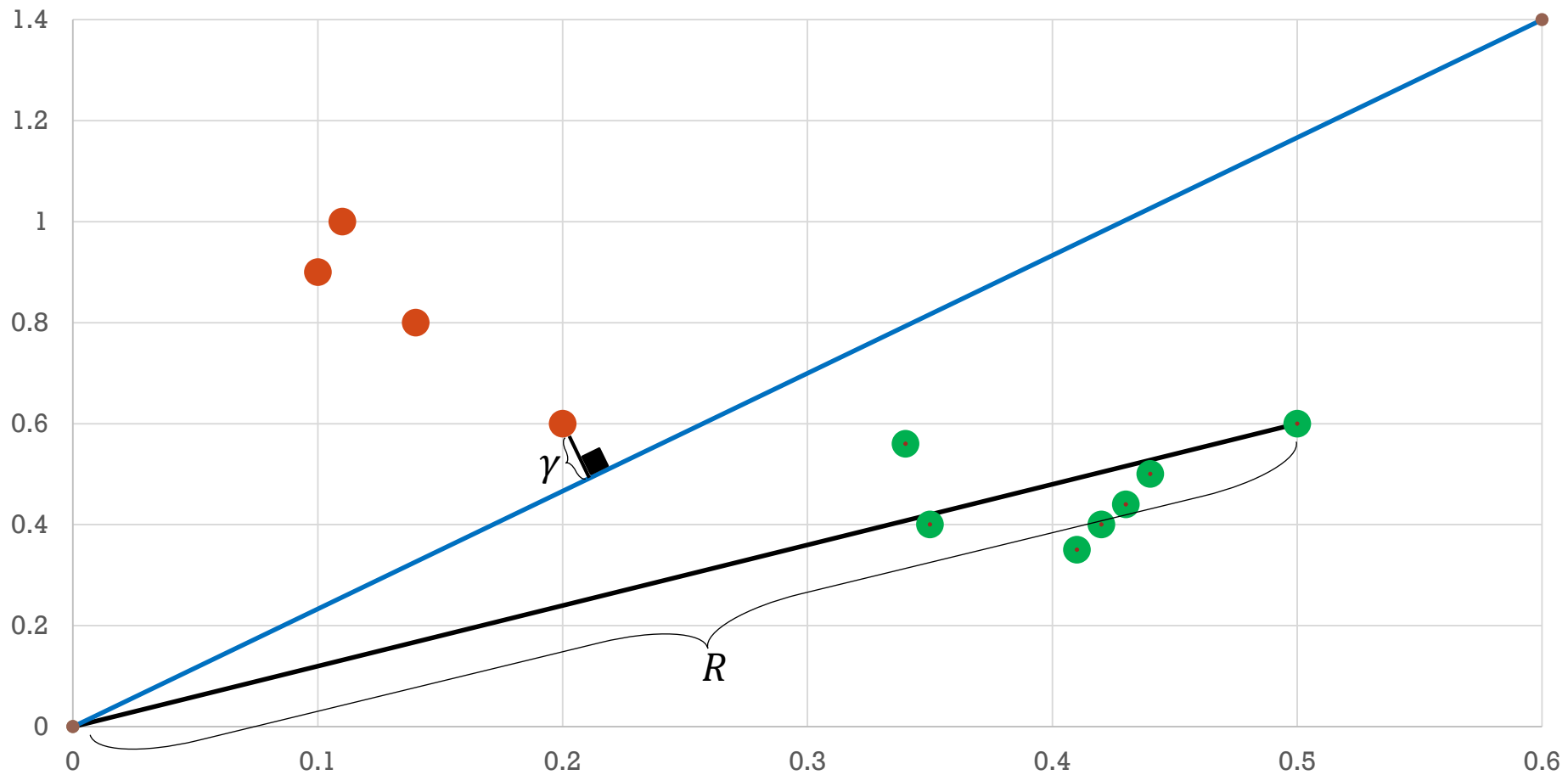
$$R = \max_i ||x_i||$$

Suppose there exists a vector  $w^*$ ,  $\gamma > 0$  such that  $||w^*|| = 1$  and  $\forall i, y_i(w^{*T} x_i) \geq \gamma$ , then the number of mistakes made by the Perceptron algorithm of this sequence of examples is  $O\left(\left(\frac{R}{\gamma}\right)^2\right)$



$$R = \max_i \|x_i\|$$

$$\forall i, y_i(w^{*T} x_i) \geq \gamma$$



# MISTAKE BOUND

Let  $w_1 = 0$  (initial weight vector) and denote  $w_k$  the weight vector after the  $k'$ th mistake.

Lemma 1:  $w_{t+1} \cdot w^* \geq w_t \cdot w^* + \gamma$

Lemma 2:  $\|w_{t+1}\|^2 \leq \|w_t\|^2 + R^2$



# MISTAKE BOUND

Lemma 1:  $w_{t+1} \cdot w^* \geq w_t \cdot w^* + \gamma$

The  $t$ 's update occurred when the perceptron did a mistake on sample  $(x_i, y_i)$ .

If  $y_i = 1$ :

$$w_{t+1} \cdot w^* = (w_t + x_i) \cdot w^* = w_t \cdot w^* + \underbrace{x_i \cdot w^*}_{\geq \gamma} \geq w_t \cdot w^* + \gamma$$

If  $y_i = -1$ :

$$w_{t+1} \cdot w^* = (w_t - x_i) \cdot w^* = w_t \cdot w^* - \underbrace{x_i \cdot w^*}_{\leq -\gamma} \geq w_t \cdot w^* + \gamma$$



# MISTAKE BOUND

Lemma 2:  $||w_{t+1}||^2 \leq ||w_t||^2 + R^2$

The  $t$ 's update occurred when the perceptron did a mistake on sample  $(x_i, y_i)$ .

If  $y_i = 1$ :

$$||w_{t+1}||^2 = ||w_t + x_i||^2 = ||w_t||^2 + 2 \underbrace{w_t \cdot x_i}_{\substack{< 0, \text{ since} \\ \text{a mistake} \\ \text{has occurred}}} + \underbrace{||x_i||^2}_{\leq R^2} \leq ||w_t||^2 + R^2$$

If  $y_i = -1$ :

$$||w_{t+1}||^2 = ||w_t - x_i||^2 = ||w_t||^2 - 2 \underbrace{w_t \cdot x_i}_{\substack{> 0, \text{ since} \\ \text{a mistake} \\ \text{has occurred}}} + \underbrace{||x_i||^2}_{\leq R^2} \leq ||w_t||^2 + R^2$$



# MISTAKE BOUND

Now, equipped with the two lemmas, we know that from Lemma 1:

$$\begin{aligned}w_1 &= \bar{0} \\w_2 \cdot w^* &\geq w_1 \cdot w^* + \gamma = \gamma \\w_3 \cdot w^* &\geq w_2 \cdot w^* + \gamma \geq \gamma + \gamma = 2\gamma\end{aligned}$$

Assume:  $w_t \cdot w^* \geq (t - 1) \cdot \gamma$

Thus –

$$w_{t+1} \cdot w^* \geq w_t \cdot w^* + \gamma \geq (t - 1) \cdot \gamma + \gamma = t \cdot \gamma$$

Moreover, from lemma 2:

$$\begin{aligned}|w_1| &= 0 \\|w_2|^2 &\leq |w_1|^2 + R^2 = R^2 \\|w_3|^2 &\leq |w_2|^2 + R^2 \leq R^2 + R^2 = 2R^2\end{aligned}$$

Assume:  $|w_t|^2 \leq (t - 1)R^2$

Thus –

$$|w_{t+1}|^2 \leq |w_t|^2 + R^2 \leq (t - 1)R^2 + R^2 = tR^2$$



# MISTAKE BOUND

Recap:

After  $T$  mistakes:

$$w_{T+1} \cdot w^* \geq T \cdot \gamma$$

$$|w_{T+1}|^2 \leq TR^2$$

$\Downarrow$

$$\gamma T \leq \underbrace{w_{T+1} \cdot w^*}_{\text{scalar}} = |w_{T+1} \cdot w^*| \stackrel{\substack{\text{Cauchy} \\ \text{Schwarz}}}{\leq} |w_{T+1}| \cdot \underbrace{|w^*|}_{=1} = |w_{T+1}|$$

$\Downarrow$

$$\gamma^2 T^2 \leq |w_{T+1}|^2 \leq TR^2$$

$\Downarrow$

$$T \leq \frac{R^2}{\gamma^2}$$

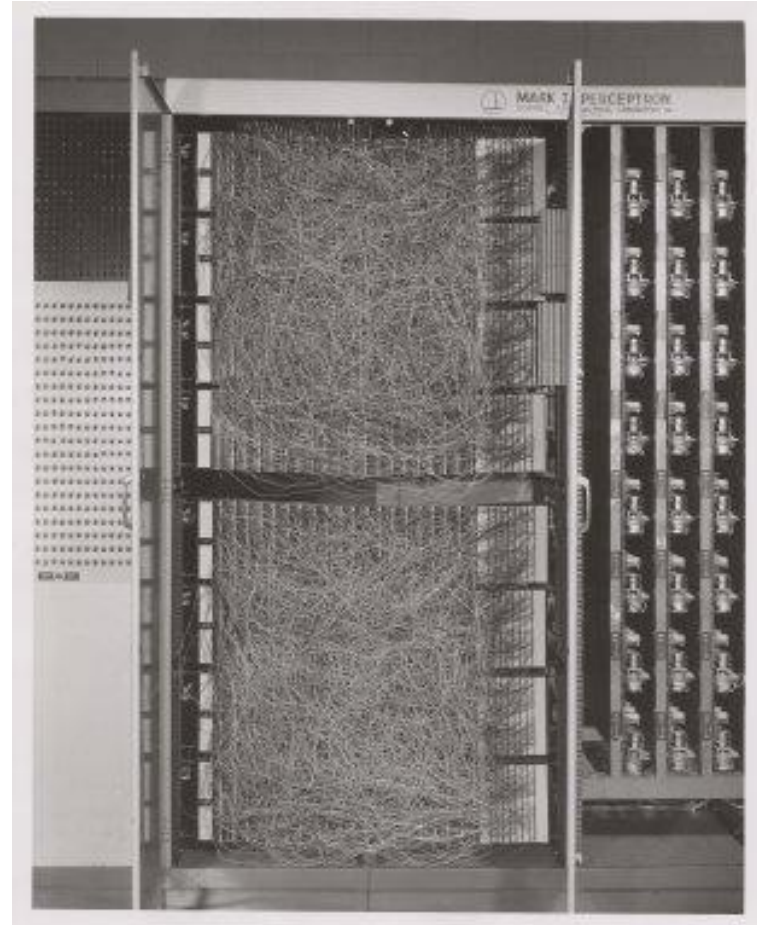




# THE FALL OF THE PERCEPTRON

The first computer built around the concept of perceptron looked like this.

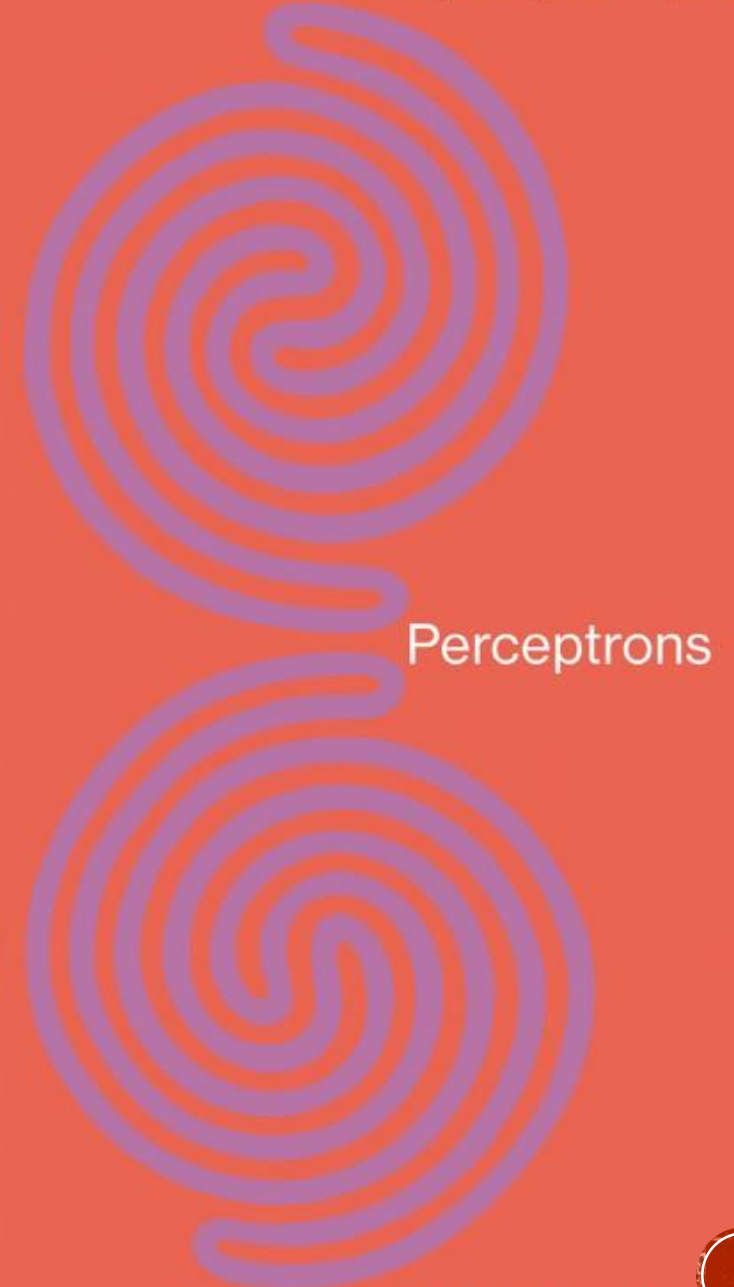
Even the wiring was supposed to simulate the connections of neurons.



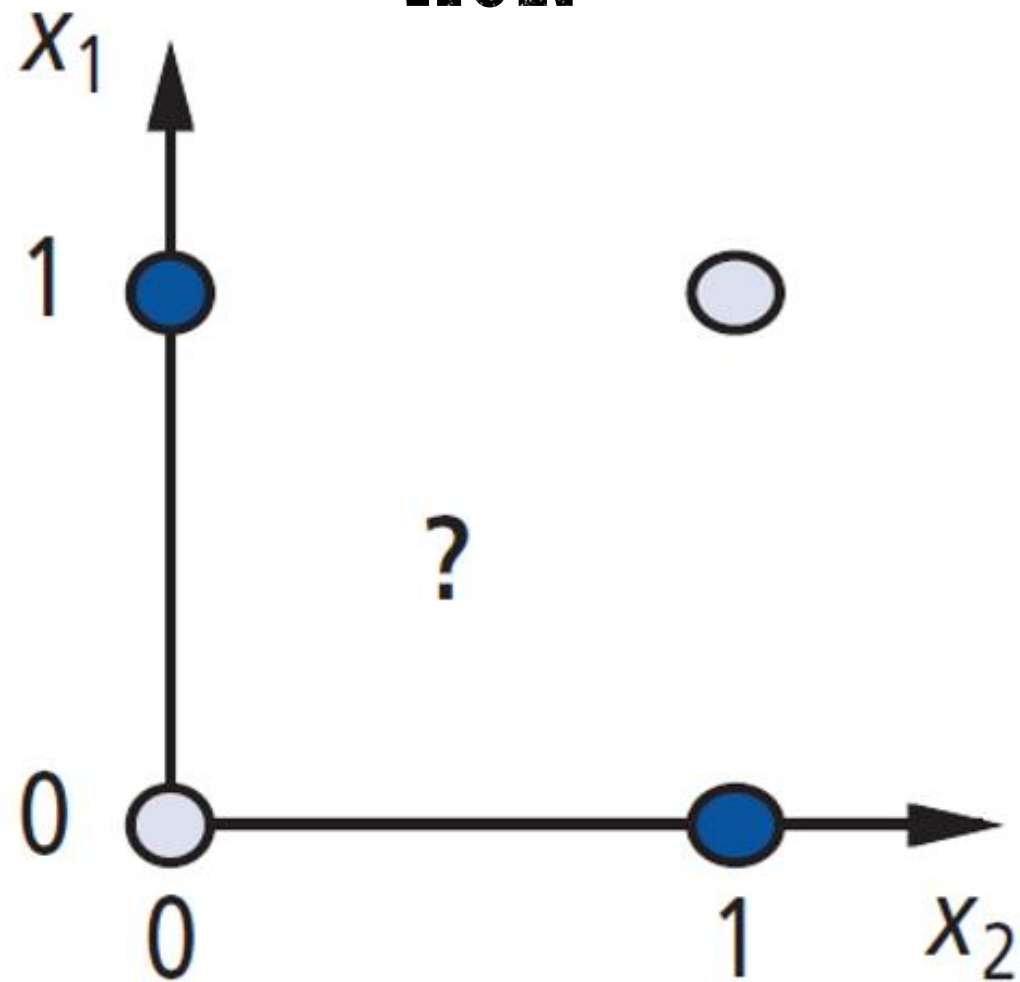
# THE FALL OF THE PERCEPTRON

However, a paper describing the perceptron's shortcomings, particularly that it was effective only at solving simple problems, led to a drastic drop in interest in artificial neural networks in the 1960's.

Unless input categories were “linearly separable”, a perceptron could not learn to discriminate between them. **Example:**



# XOR

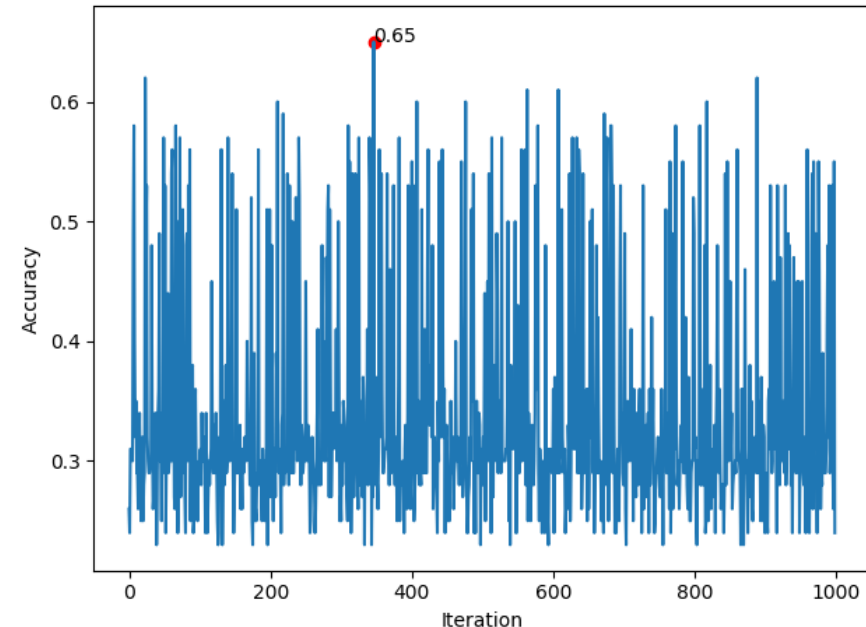
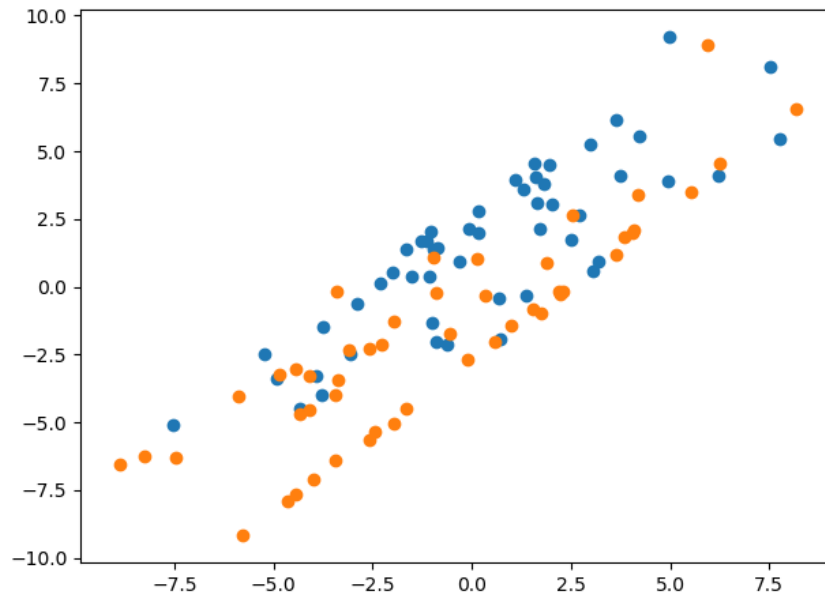


# STOPPING CRITERIONS

The mistake bound holds only if the dataset is linearly separable.

If it's not the case, one could define other criteritions:

- Approach 1: Consider the perceptron as an any-time algorithm. When the user is out of time or resources, return the current weights.
- Approach 2: After each update, calculate the accuracy, and remember the weights corresponding to the highest accuracy:



# KERNEL PERCEPTRON

- Lets have a look at the learning rule:

*if  $\hat{y} \neq y_i$ :*

$$w \leftarrow w + y_i x_i$$

- Thus, we can infer that the weights vector  $w$  learned by the Perceptron's algorithm is a linear combination of all the data points:

$$w = \sum_i \alpha_i y_i x_i$$

- $\alpha_i$  is a “mistake-counter” - how many times the perceptron made a mistake on sample  $x_i$
- We can rewrite the prediction formula of a new observation  $x'$  as:

$$\hat{y} = \text{sign} \left[ \left( \sum_i \alpha_i y_i x_i \right)^T x' \right] = \text{sign} \left[ \sum_i \alpha_i y_i x_i^T x' \right]$$



# KERNEL PERCEPTRON

$$\hat{y} = \text{sign} \left[ \left( \sum_i \alpha_i y_i x_i \right)^T x' \right] = \text{sign} \left[ \sum_i \alpha_i y_i x_i^T x' \right]$$

- In other words, the dual problem is finding the  $\alpha'_i$ 's. The new learning algorithm would loop thru the samples and make predictions, but now it will update a “mistake counter” vector  $\alpha$  rather than updating a weights vector  $w$ .

- First observation:

We can define a threshold  $s$ , and during learning we can zero (and consider dropping samples from the dataset) any mistake-counter that reaches a value above it, i.e.:

*if  $\alpha_i \geq s$ :*

*$\alpha_i \leftarrow 0$*

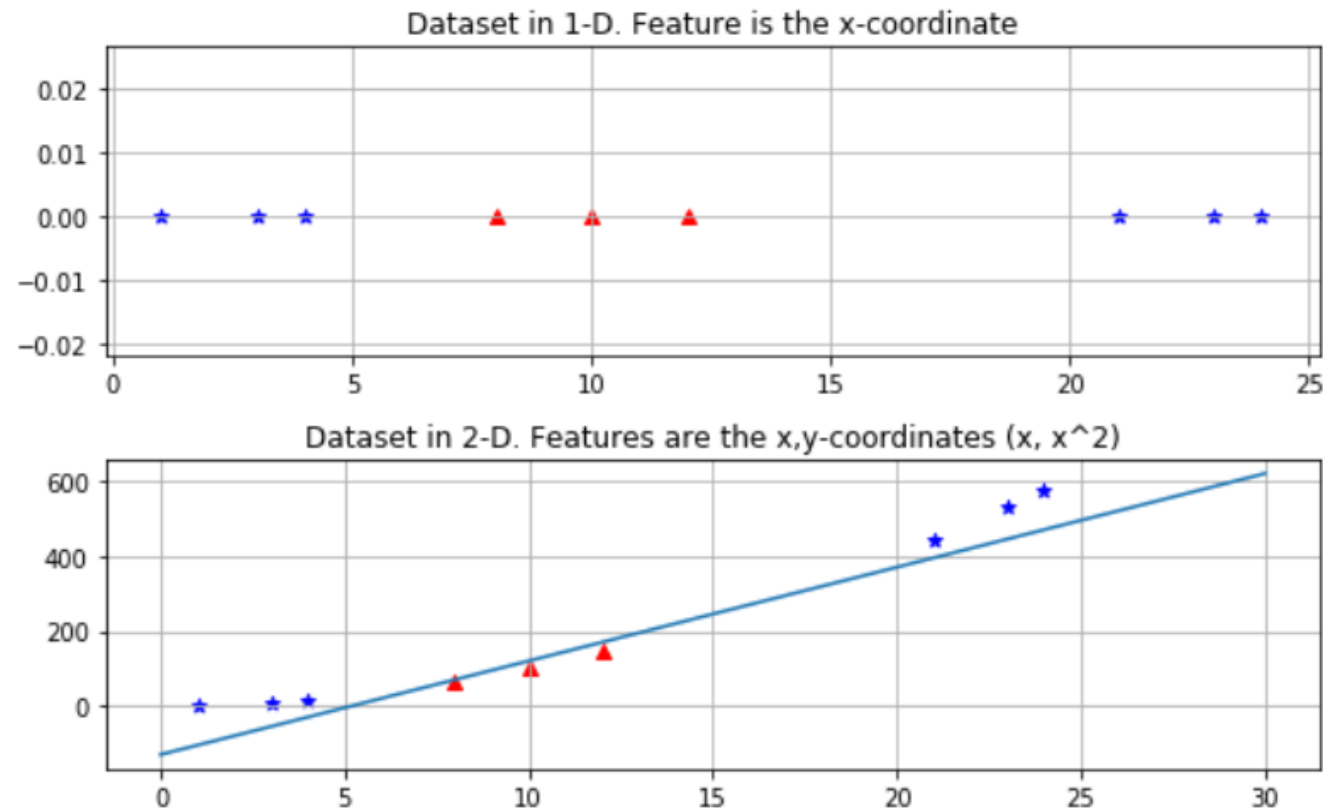
*drop  $x_i$  from the dataset*

- This could help us detecting outliers



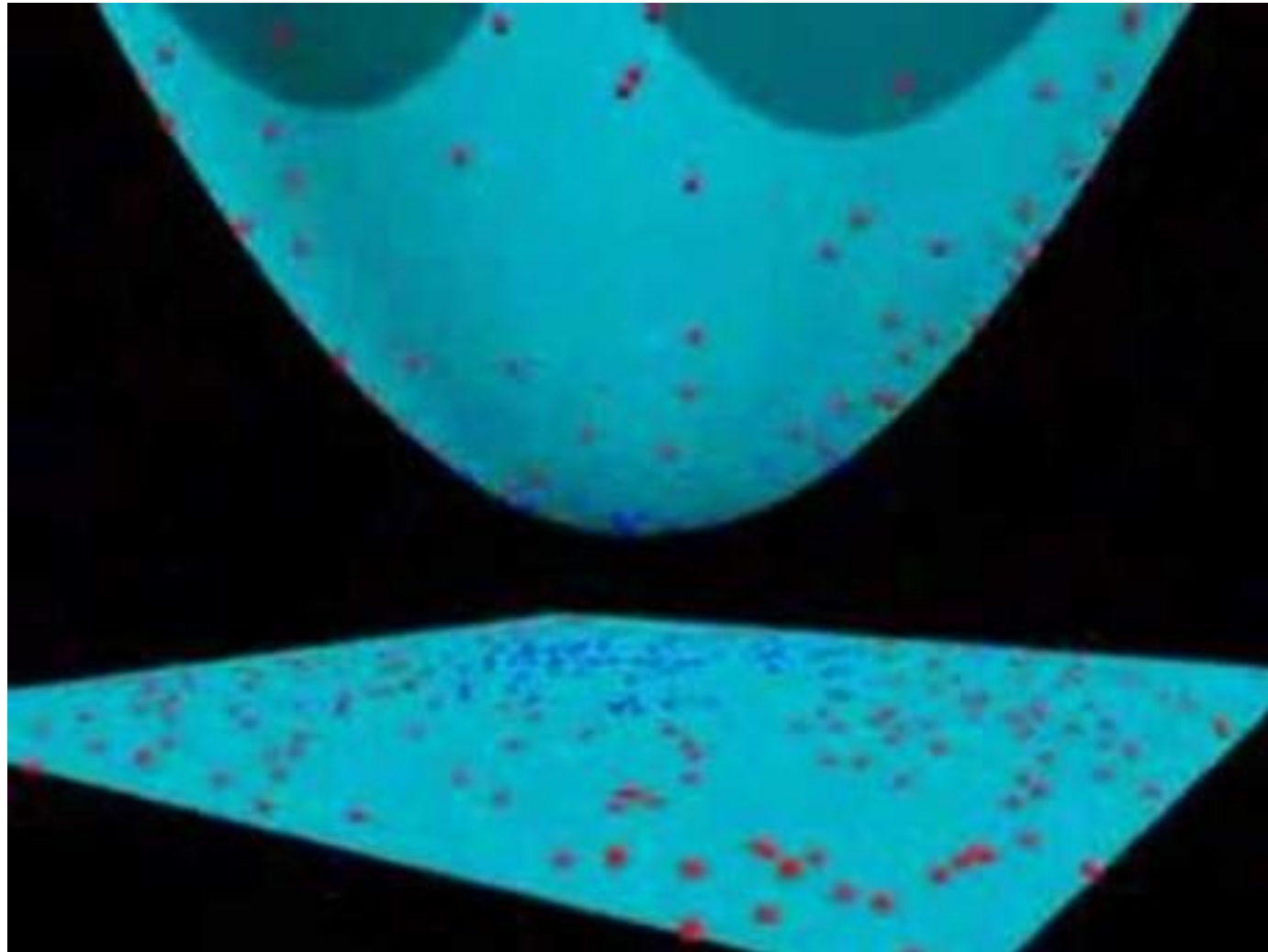
# KERNEL PERCEPTRON

- Second observation:
- While there exist datasets that aren't linearly separable in a given form (set of features), we can find a transformations to another space where the points are linearly separable:



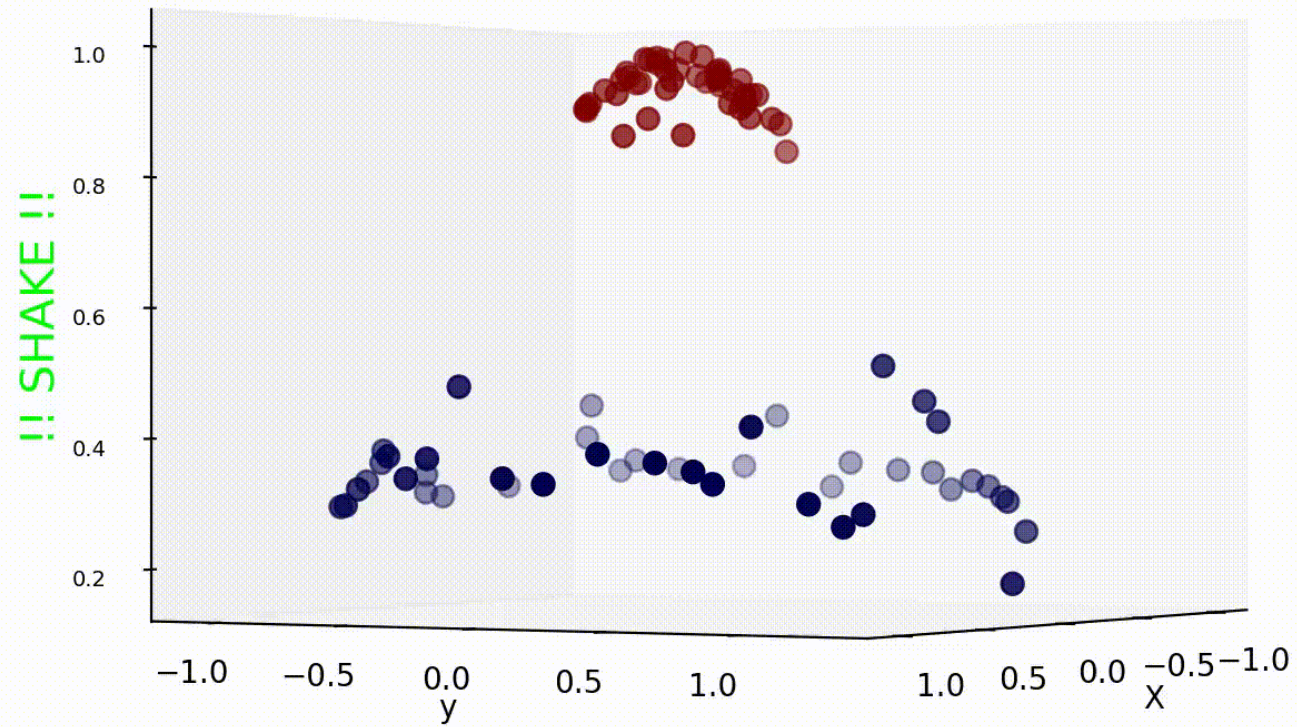


# KERNEL PERCEPTRON





# KERNEL PERCEPTRON



# KERNEL PERCEPTRON

- Our next goal is to find transformations to spaces where our datasets are linearly separable. But firstly, we have to find a method to apply these transformations effectively.
- Let  $\mathbb{R}^n$  be the original features' space and  $\mathbb{R}^m$  be the new features' space. We want to find a transformation:

$$\varphi: \mathbb{R}^n \rightarrow \mathbb{R}^m$$

- The original prediction rule was:

$$\hat{y} = \text{sign} \left[ \sum_i \alpha_i y_i x_i^T x' \right]$$

- In the new features' space, the prediction rule would become:

$$\hat{y} = \text{sign} \left[ \sum_i \alpha_i y_i \varphi(x_i)^T \varphi(x') \right]$$

- In a low dimensions space, we cannot deal with more complex datasets.
- In a high dimensions space, the computations become very slow.
- A new trick called – “**The Kernel Trick**” – comes to the rescue!



# KERNEL PERCEPTRON

- It makes it possible to get the same results as if you added many features, without adding them in practice.
- Usually, we define a kernel  $K$  such that  $K(x, y) = \varphi^T(x)\varphi(y)$  and find a direct formula that doesn't involve any transformation to a higher dimension space.
- Example:

$$\varphi(y) = \varphi\left(\begin{bmatrix} u_1 \\ u_2 \end{bmatrix}\right) = \begin{bmatrix} 1 \\ \sqrt{2}u_1 \\ \sqrt{2}u_2 \\ \sqrt{2}u_1u_2 \\ u_1^2 \\ u_2^2 \end{bmatrix}$$

The inner product:

$$\varphi^T(u)\varphi(v) = 1 + 2u_1v_1 + 2u_2v_2 + 2u_1u_2v_1v_2 + u_1^2v_1^2 + u_2^2v_2^2 = (1 + u^T v)^2$$

Note that computing that dot product  $u^T v$  in the original features' space is less expensive than computing the dot product in the transformed space.



# KERNEL PERCEPTRON

- Examples of the most used kernel functions:

$K(x, y) = (x^T y + 1)^p$  – *Polynomial kernel of degree*

$K(x, y) = e^{-\frac{1}{2\sigma^2}|x-y|^2}$  – *Gaussian kernel*

$K(x, y) = e^{-\gamma|x-y|^2}$  – *RBG kernel*

$K(x, y) = \tanh(\eta x^T y + \theta)$  – *Sigmoid kernel*



# KERNEL PERCEPTRON — THE ALGORITHM

Initialize a mistake-counter vector  $\alpha \leftarrow 0$

While some stopping criterion isn't met:

For each  $x_j, y_j$  in the training set:

Predict  $\hat{y} = \text{sign}(\sum_i \alpha_i y_i K(x_i, x_j))$

If  $\hat{y} \neq y_j$ :

$$\alpha_j \leftarrow \alpha_j + 1$$

- The prediction rule:

$$\hat{y} = \text{sign} \left[ \sum_i \alpha_i y_i K(x_i, x') \right]$$

