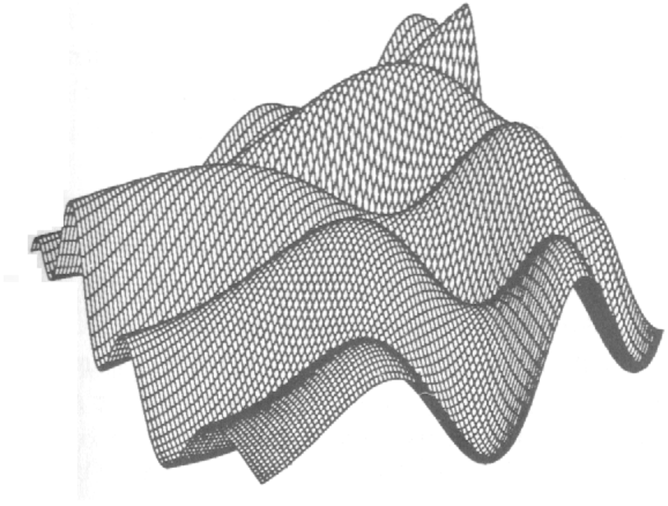


Perceptrons

"From the heights of error,

To the valleys of Truth"

Piyush Kumar

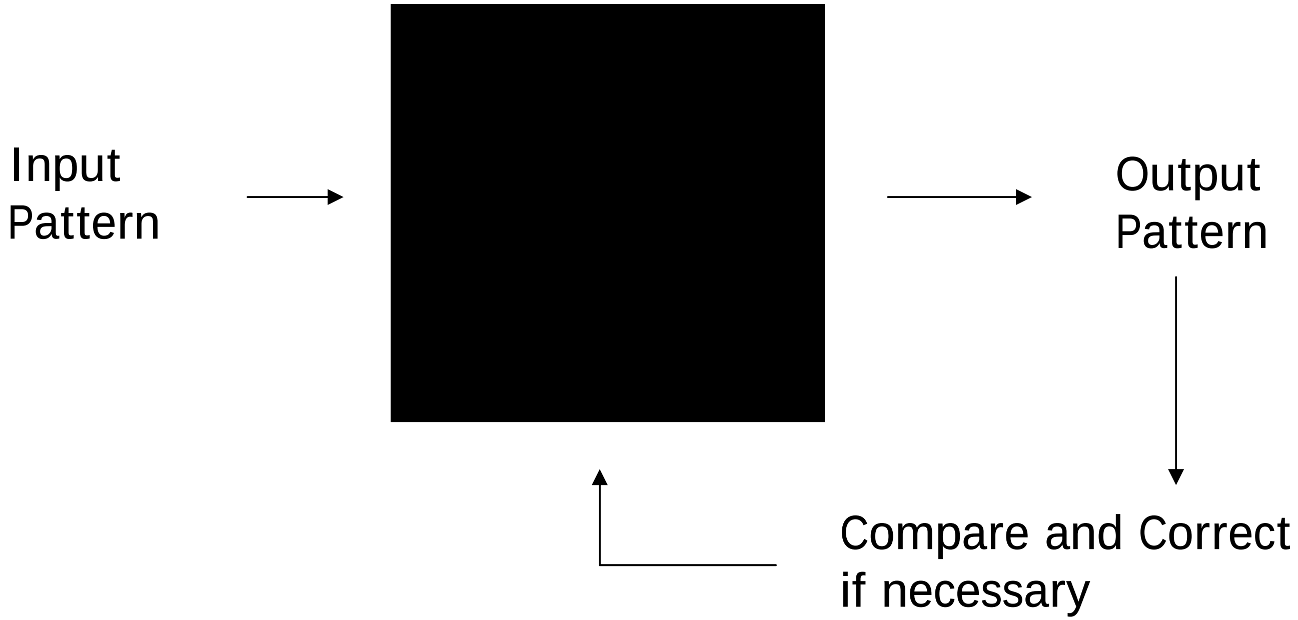
Advanced Computational Geometry

Reading Material

- Duda/Hart/Stork : 5.4/5.5/9.6.8
- Any neural network book (Haykin, Anderson...)
- Look at papers of related people
  - Santosh Vempala
  - A. Blum
  - J. Dunagan
  - F. Rosenblatt
  - T. Bylander

Introduction

- Supervised Learning



Linear discriminant functions

- Definition

  It is a function that is a linear combination of the components of x

  $$g(x) = w^t x + w_0 \qquad (1)$$

  where w is the weight vector and w_0 the bias

- A two-category classifier with a discriminant function of the form (1) uses the following rule:

  Decide ω_1 *if* $g(x) > 0$ and ω_2 if $g(x) < 0$

  $\Leftrightarrow$ Decide ω_1 *if* $w^t x > -w_0$ and ω_2 otherwise

  If $g(x) = 0 \Rightarrow x$ is assigned to either class

LDFs

- The equation $g(x) = 0$ defines the decision surface that separates points assigned to the category ω_1 from points assigned to the category ω_2

- When $g(x)$ is linear, the decision surface is a hyperplane

Classification using LDFs

- Two main approaches
  - Fischer's Linear Discriminant

    Project data onto a line with 'good' discrimination; then classify on the real line

  - Linear Discrimination in d-dimensions

    Classify data using suitable hyperplanes.

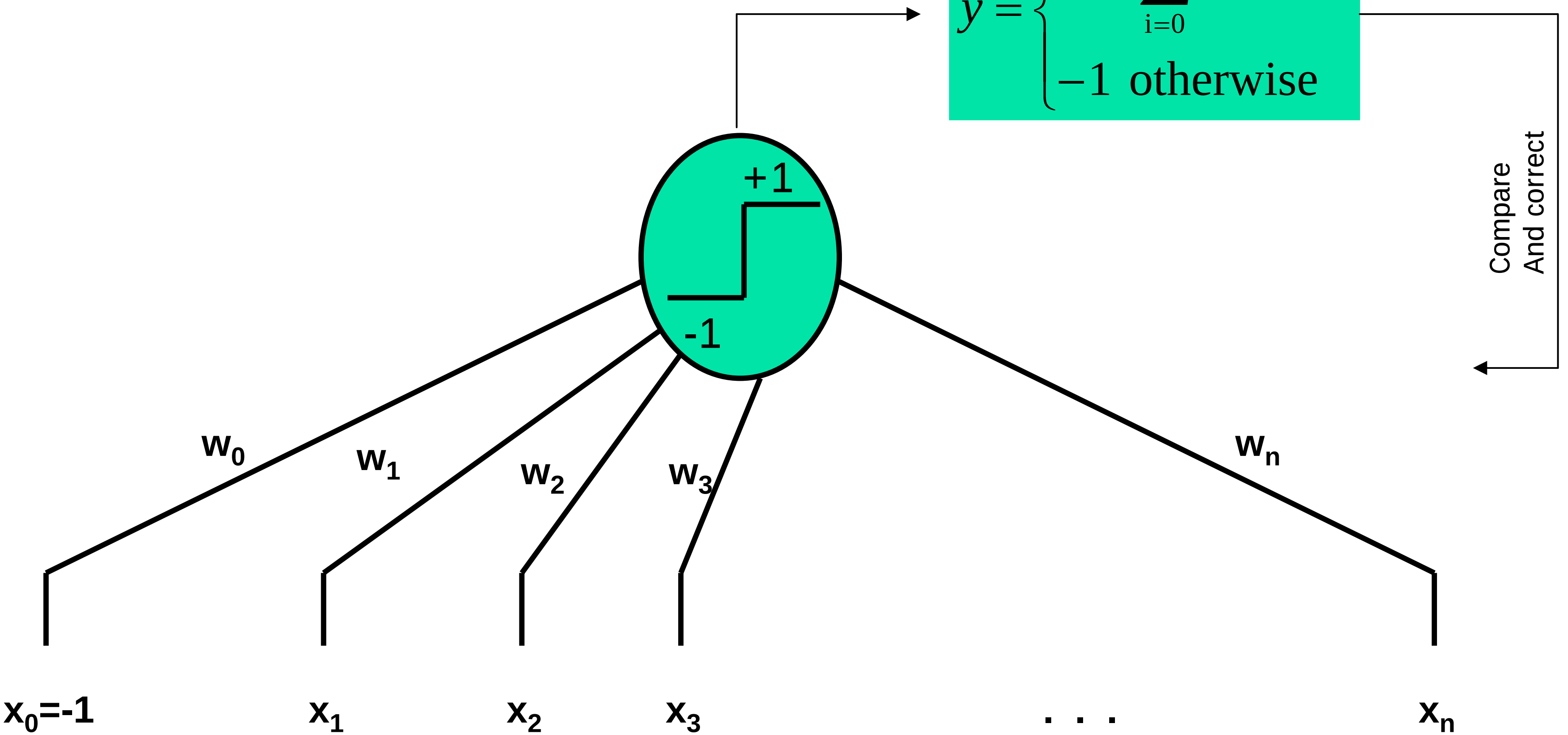
    (We'll use perceptrons to construct these)

Perceptron: The first NN

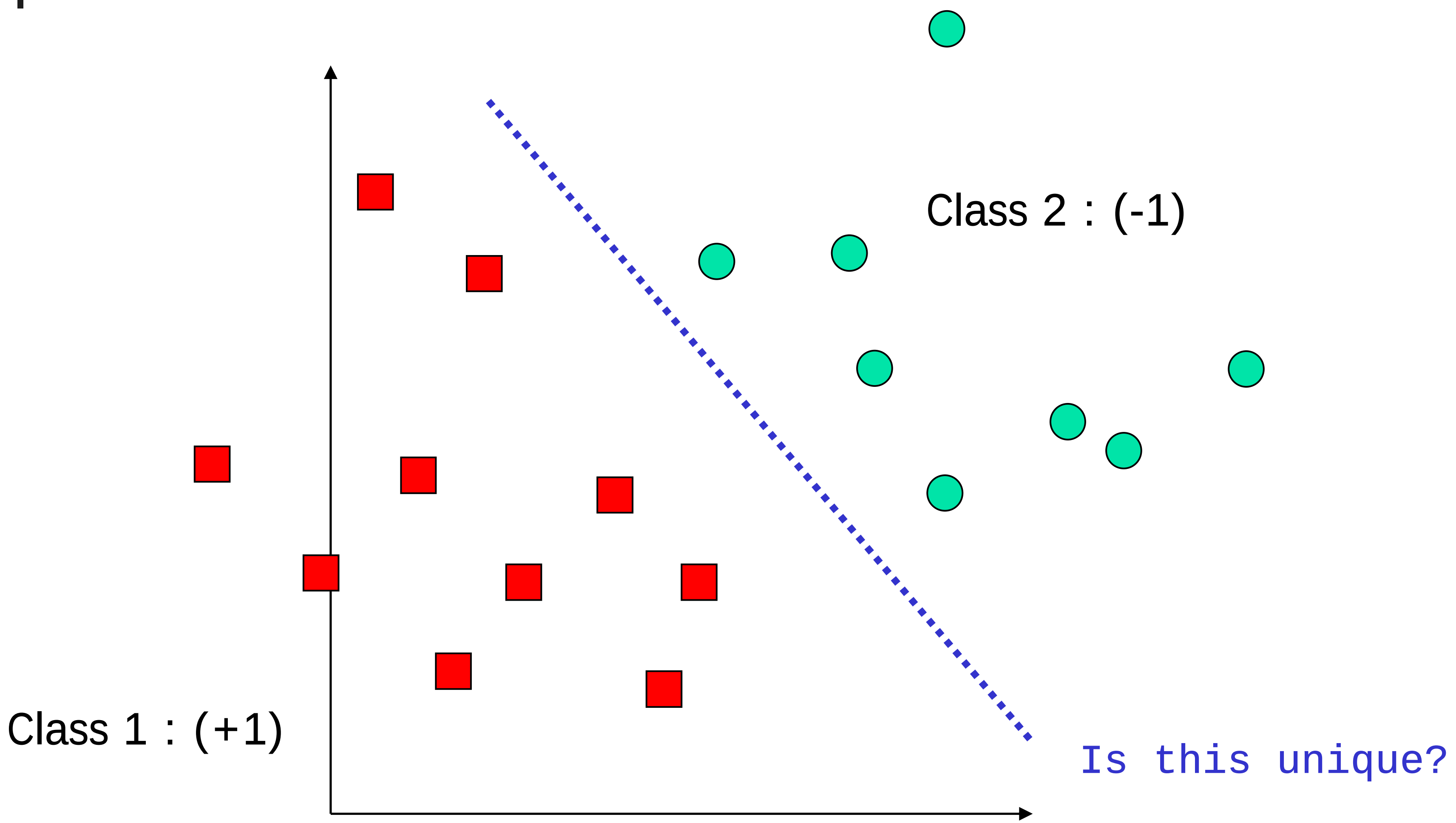
- Proposed by Frank Rosenblatt in 1956
- Neural net researchers accuse Rosenblatt of promising ‘too much’ ☺
- Numerous variants
- We’ll cover the one that’s most geometric to explain ☺
- One of the simplest Neural Network.

Perceptrons : A Picture

$$y = \begin{cases} 1 \text{ if } \sum_{i=0}^{n} w_i x_i > 0 \\ -1 \text{ otherwise} \end{cases}$$



Where is the geometry?



Assumption

- Lets assume for this talk that the red and green points in ‘feature space’ are separable using a hyperplane.

Two Category Linearly separable case

Whatz the problem?

- Why not just take out the convex hull of one of the sets and find one of the 'right' facets?
  - Because its too much work to do in d-dimensions.
- What else can we do?
  - Linear programming == Perceptrons
  - Quadratic Programming == SVMs

Perceptrons

- Aka Learning Half Spaces
- Can be solved in polynomial time using IP algorithms.
- Can also be solved using a *simple and elegant* greedy algorithm

(Which I present today)

In Math notation

N samples : $\{(\vec{x}_1, y_1), (\vec{x}_2, y_2), \ldots, (\vec{x}_n, y_n))$

Where y = +/- 1 are labels for the data. $\vec{x} \in \mathbf{R}^d$

Can we find a hyperplane $\vec{w}.\vec{x} = 0$ that separates the two classes? (labeled by y) i.e.

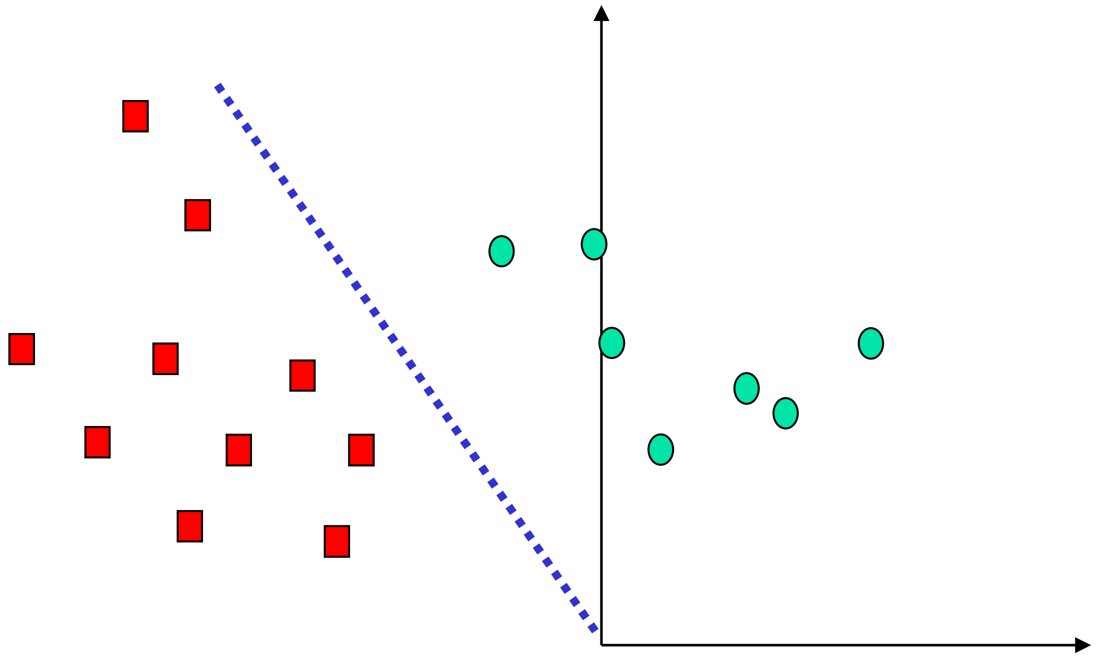
$\vec{x}_j.\vec{w} > 0$: For all j such that y = +1

$\vec{x}_j.\vec{w} < 0$: For all j such that y = -1

Which we will relax later!

Further assumption 1

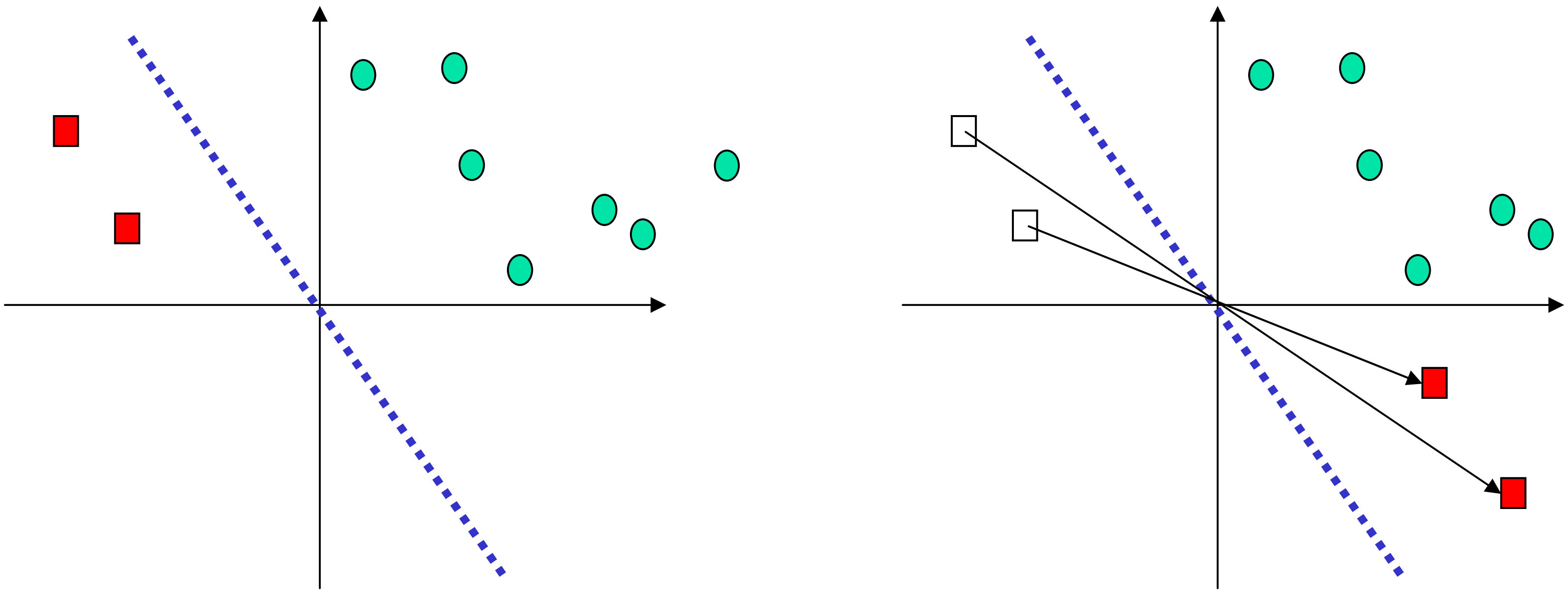
Lets assume that the hyperplane that we are looking for passes thru the origin



Relax now!! ☺

Further assumption 2

- Lets assume that we are looking for a ***halfspace*** that contains a set of points



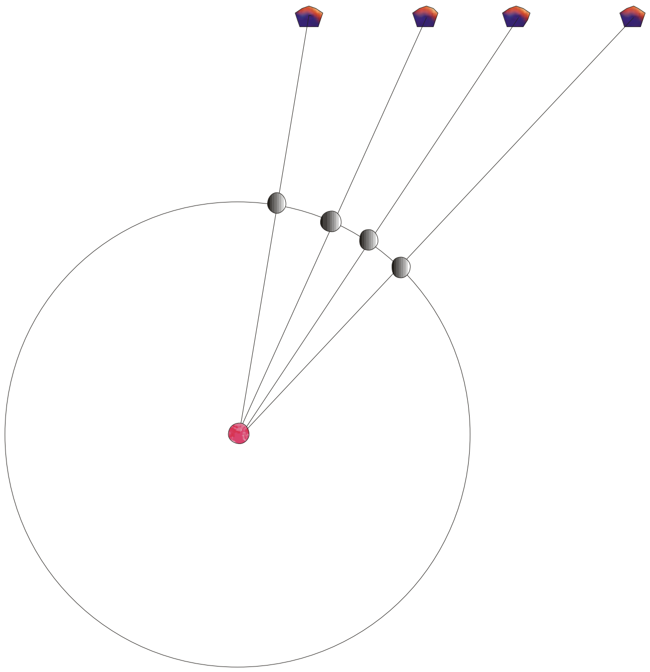
Lets Relax FA 1 now

- "Homogenize" the coordinates by adding a new coordinate to the input.
- Think of it as moving the whole red and blue points in one higher dimension
- From 2D to 3D it is just the x-y plane shifted to z = 1. This takes care of the "bias" or our assumption that the halfspace can pass thru the origin.

Relax now! ☺

Further Assumption 3

- Assume all points on a unit sphere!
- If they are not after applying transformations for FA 1 and FA 2 , make them so.



Restatement 1

- Given: A set of points on a sphere in d-dimensions, such that all of them lie in

  a half-space.

- Output: Find one such halfspace

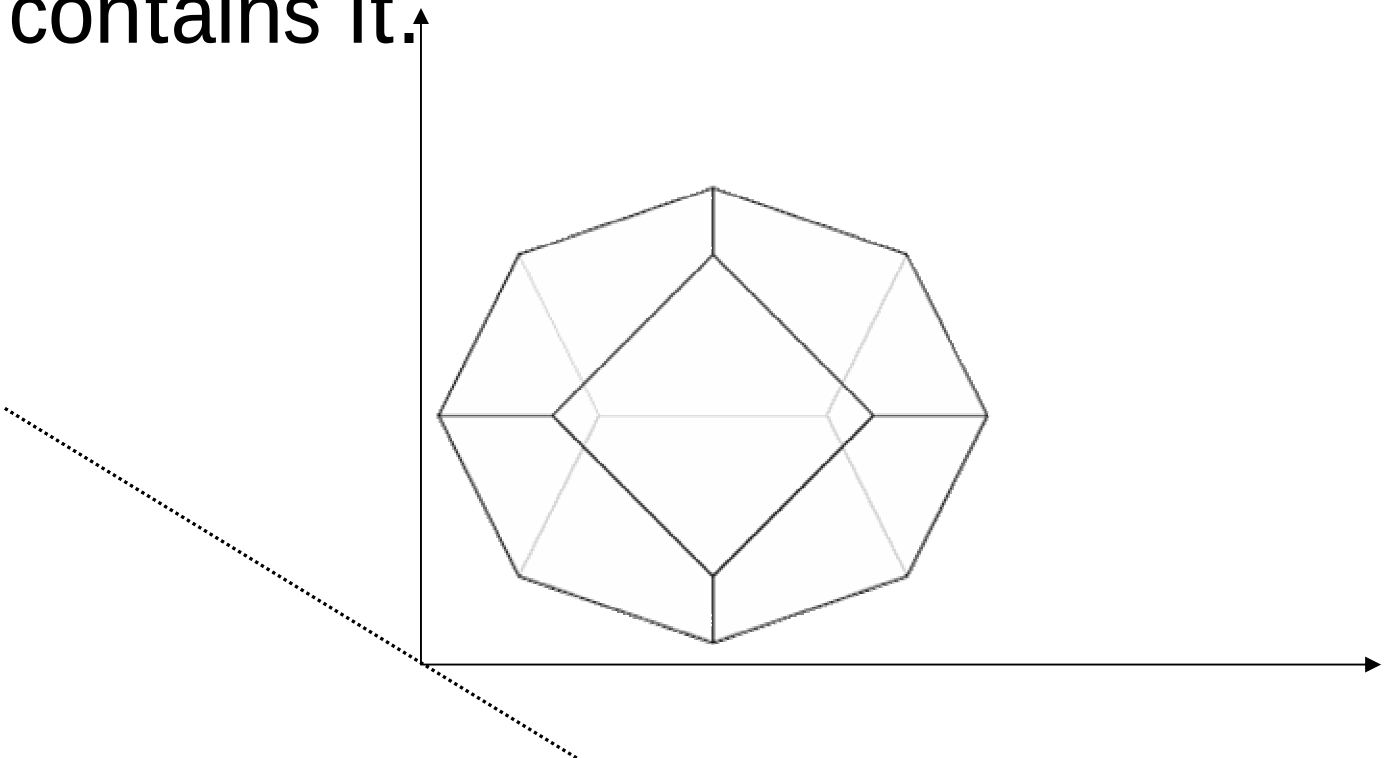
- Note: You can solve the LP feasibility problem.

⟺ You can solve any general LP !!

Take Estie's class if you
Want to know why. ☺

Restatement 2

- Given a convex body (in V-form), find a halfspace passing thru the origin that contains it.



Support Vector Machines

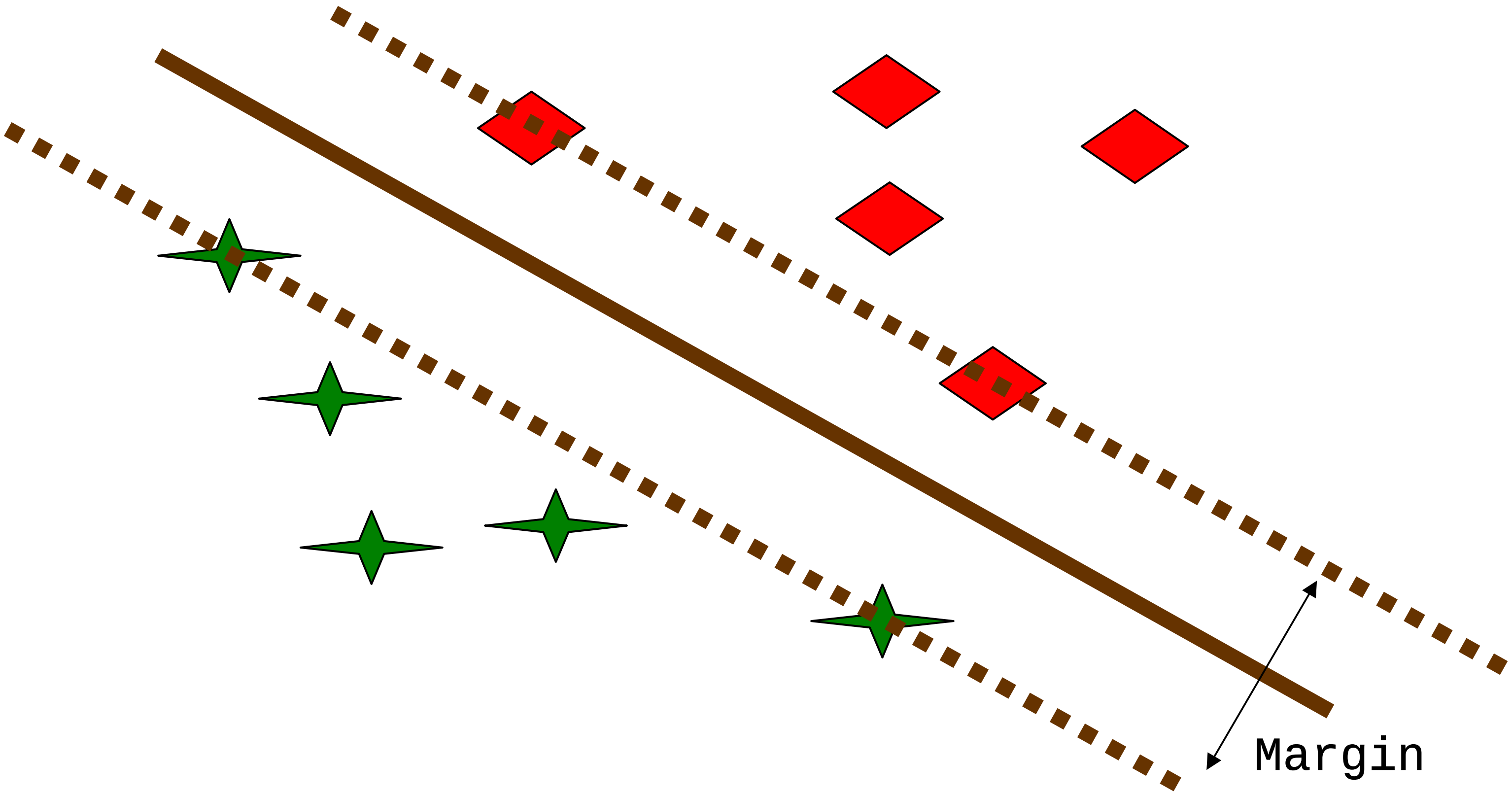
A small break from perceptrons

Support Vector Machines

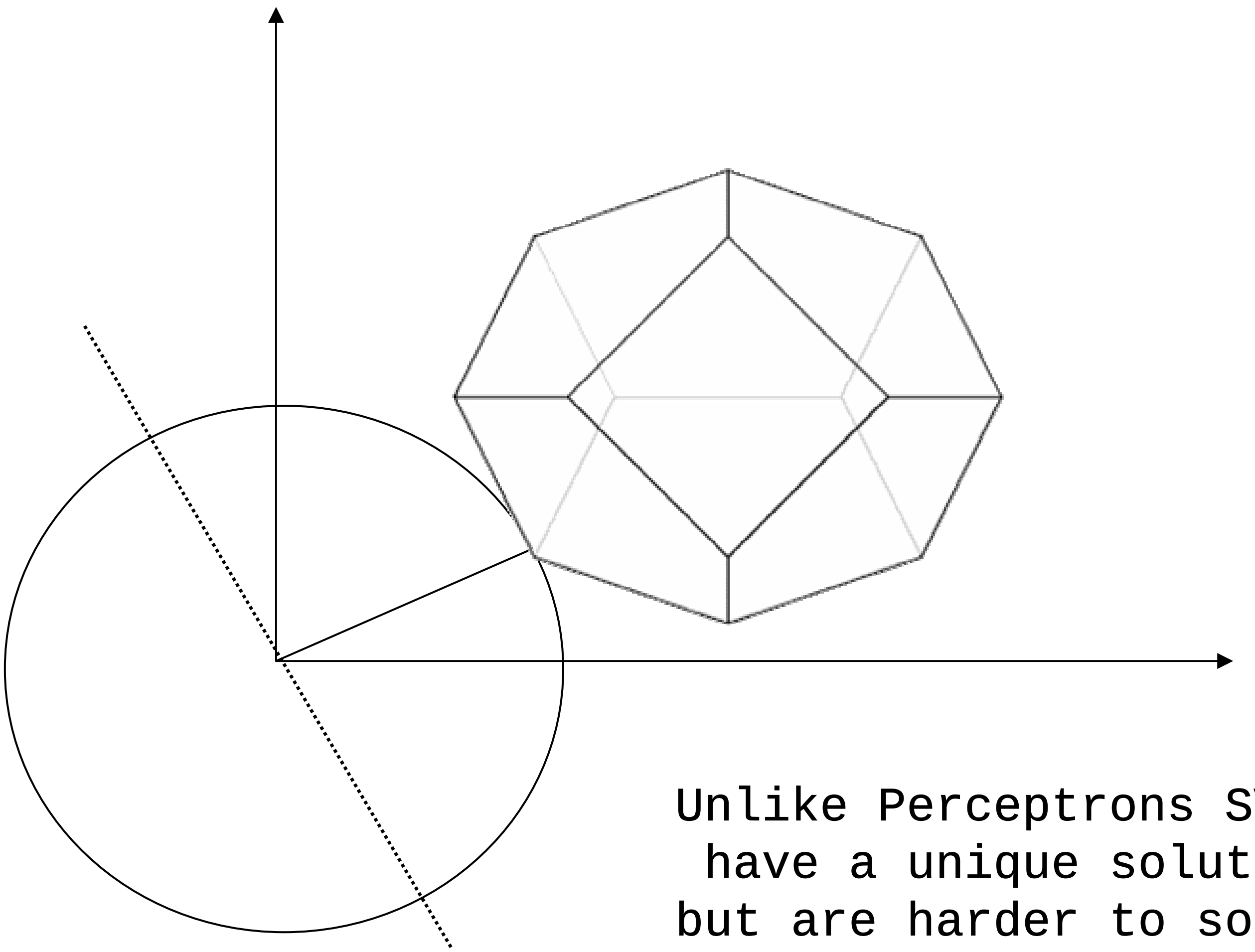
- Linear Learning Machines like perceptrons.
- Map non-linearly to higher dimension to overcome the linearity constraint.
- Select between hyperplanes, Use margin as a test
  (This is what perceptrons don’t do)

From learning theory, maximum margin is good

SVMs



Another Reformulation



Unlike Perceptrons SVMs have a unique solution but are harder to solve. <QP>

Support Vector Machines

- There are very simple algorithms to solve SVMs (as simple as perceptrons)

(If there is enough demand,

I can try to cover it)

(and If my job hunting lets me ;))

Back to perceptrons

Perceptrons

- So how do we solve the LP ?
  - Simplex
  - Ellipsoid
  - IP methods
  - Perceptrons = Gradient Decent

So we could solve the classification problem using any LP method.

Why learn Perceptrons?

- You can write an LP solver in 5 mins !
- A very slight modification can give u a polynomial time guarantee (Using smoothed analysis)!

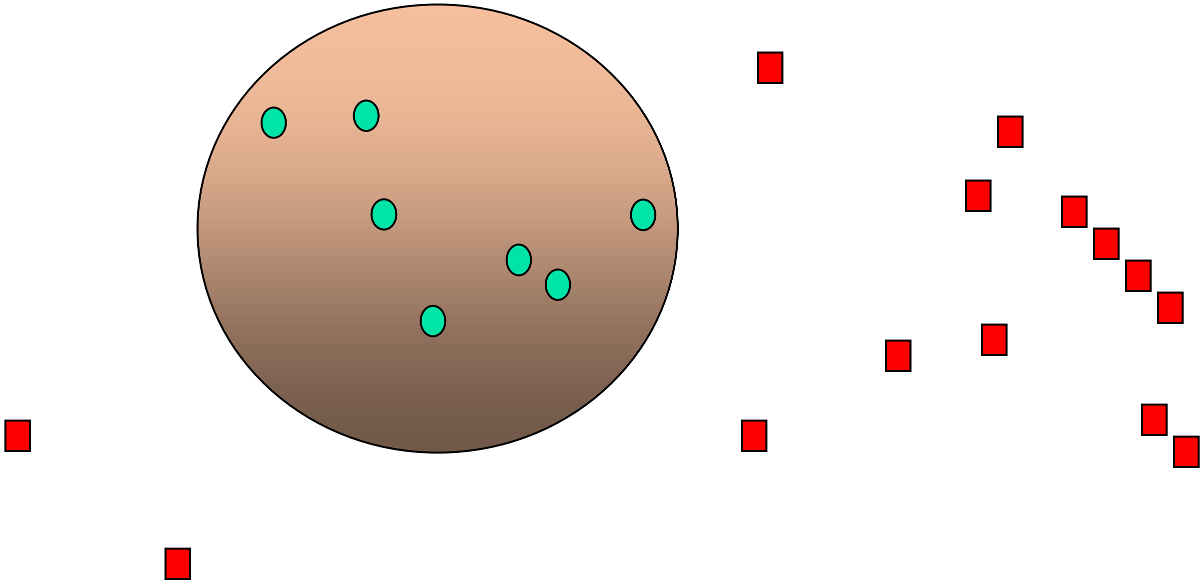
Why learn Perceptrons

- Multiple perceptrons clubbed together are used to learn almost anything in practice. (Idea behind multi layer neural networks)
- Perceptrons have a finite capacity and so cannot represent all classifications. The amount of training data required will need to be larger than the capacity. We'll talk about capacity when we introduce VC-dimension.

From learning theory, limited capacity is good

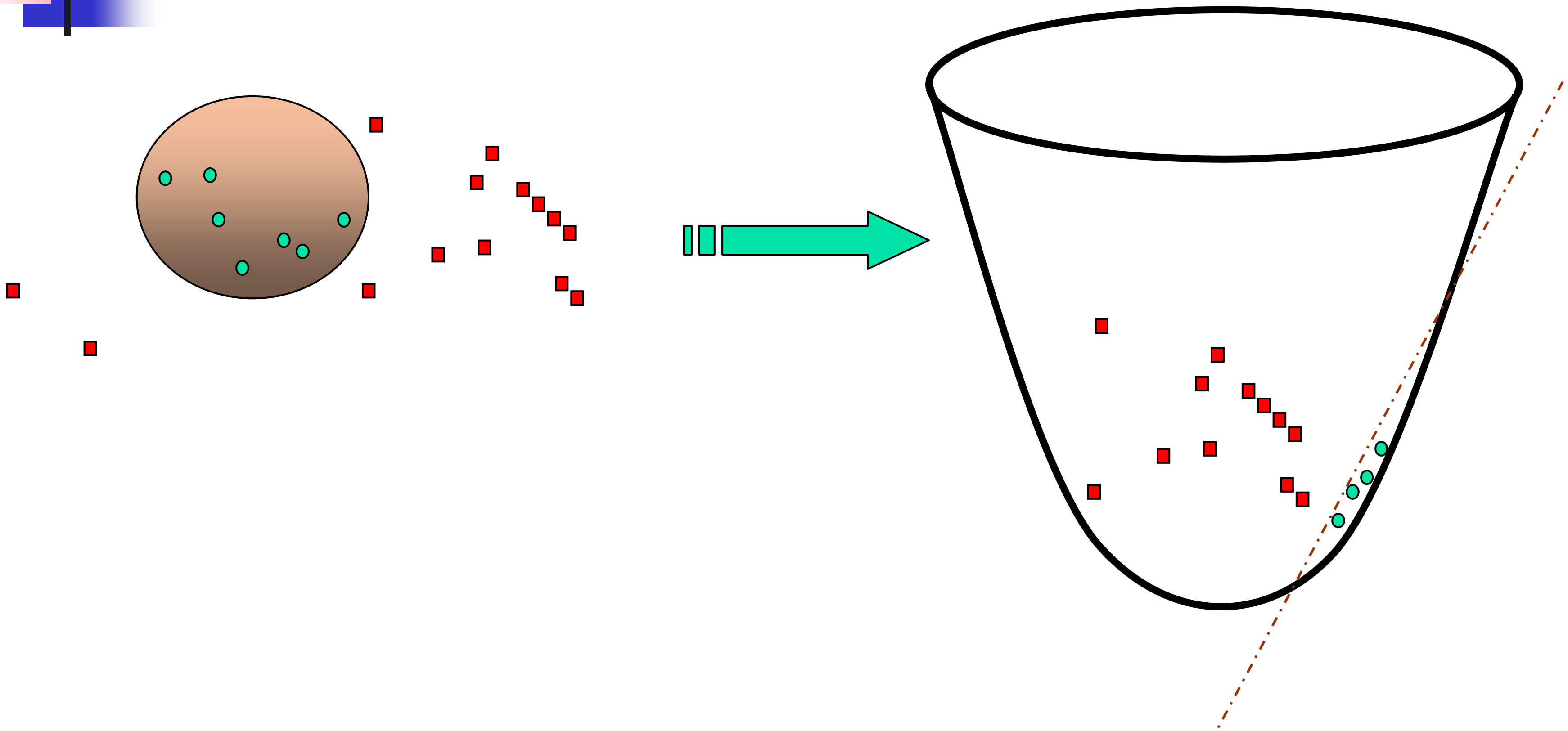
Another twist : Linearization

- If the data is separable with say a sphere, how would you use a perceptron to separate it? (Ellipsoids?)



Delaunay!??

Linearization



Lift the points to a paraboloid in one higher dimension,
For instance if the data is in 2D,

$(x,y) \rightarrow (x,y,x^2+y^2)$

The kernel Matrix

- Another trick that ML community uses for Linearization is to use a function that redefines distances between points.

- Example : $K(x,z) = e^{-\|x-z\|^2/2\sigma}$

- There are even papers on how to learn kernels from data !

Perceptron Smoothed Complexity

Let L be a linear program and let L′ be the same linear program under a Gaussian perturbation of variance sigma^2, where $\text{sigma}^2 <= 1/2d$. For any delta, with probability at least 1 – delta either

The perceptron finds a feasible solution in poly(d,m,1/sigma,1/delta)

L′ is infeasible or unbounded

The Algorithm

In one line

The 1 Line LP Solver!

- Start with a random vector w, and if a point is misclassified do:

$$\vec{w}_{k+1} = \vec{w}_k + \vec{x}_k$$

(until done)

One of the most beautiful LP Solvers I've ever come across…

A better description

```
Initialize w=0, i=0
    do i = (i+1) mod n
    if x_i is misclassified by w
    then w =  w + x_i
    until all patterns classified
Return w
```

That’s the entire code!

Written in 10 mins.

An even better description

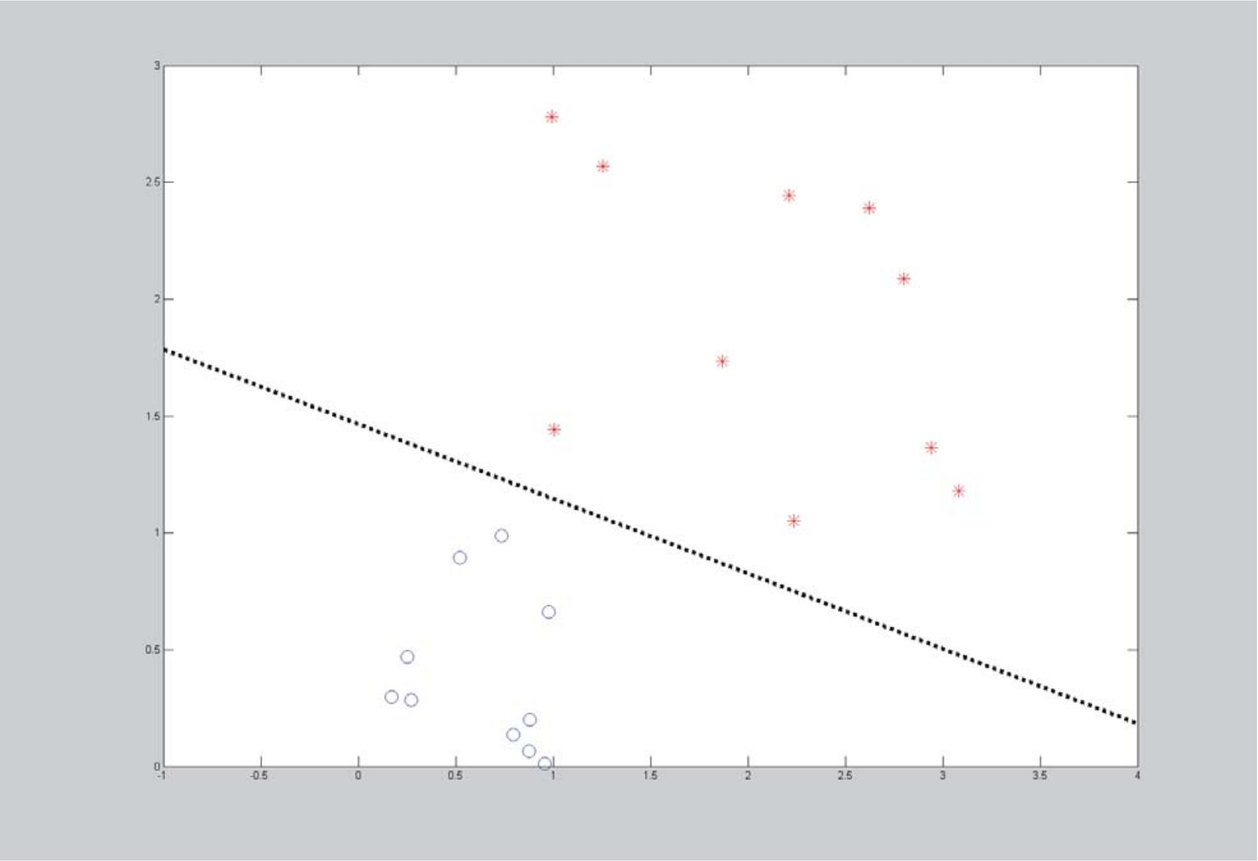
```
function w = perceptron(r,b)
r = [r (zeros(length(r),1)+1)];  % Homogenize
b = -[b (zeros(length(b),1)+1)]; % Homogenize and flip

data = [r;b];                    % Make one pointset
s = size(data);                  % Size of data?
w = zeros(1,s(1,2));             % Initialize zero vector

is_error = true;
while is_error
    is_error = false;
    for k=1:s(1,1)
        if dot(w,data(k,:)) <= 0
            w = w+data(k,:); is_error = true;
        end
    end
end
```

And it can be solve any LP!

An output



In other words

At each step, the algorithm picks any vector x that is misclassified, or is on the wrong side of the halfspace, and brings the normal vector w closer into agreement with that point

Still: Why the hell does it work?

Back to the most advanced presentation tools available on earth !

The blackboard ☺

Wait (Lemme try the whiteboard)

The Convergence Proof

Proof

$$\omega_{i+1} = \omega_i + x_i$$

Let $\hat{\omega}$ be an optimal solution such that $\|\hat{\omega}\| = 1$.

$$(\omega_{i+1} - \alpha\hat{\omega}) = (\omega_i - \alpha\hat{\omega}) + x_i$$

Proof

$$\| \omega_{i+1} - \alpha \hat{\omega} \|^2 = \| \omega_i - \alpha \hat{\omega} \|^2 + \| x_i \|^2 + \underbrace{2(\omega_i - \alpha \hat{\omega}) x_i}_{2\omega_i x_i - 2\alpha \hat{\omega} x_i}$$

Proof

$\omega_i \cdot x_i < 0$ since x_i was misclassified

$$\Rightarrow \quad \|\omega_{i+1} - \alpha\hat{\omega}\|^2 \leq \|\omega_i - \alpha\hat{\omega}\|^2 + 1 - 2\alpha\gamma$$

Let $\gamma = \min_{x_i} x_i \cdot \hat{\omega}$

Proof

Let $\alpha = 1/\gamma$

$$\Rightarrow \quad \| \omega_{i+1} - \alpha \hat{\omega} \|^2 \leq \| \omega_i - \alpha \hat{\omega} \|^2 - 1$$

Proof

$$\Rightarrow \quad 0 \leq \| \omega_{i+k} - \alpha \hat{\omega} \|^2 \leq \| \omega_i - \alpha \hat{\omega} \|^2 - k$$

$$\Rightarrow \quad 0 \leq \| \omega_0 - \alpha \hat{\omega} \|^2 - k$$

Proof

$$\Rightarrow \quad k \leq \|\alpha \hat{w}\|^2 \qquad [\because w_0 = 0]$$

$$\leq \alpha^2 \qquad [\because \|\hat{w}\| = 1]$$

$$\leq \frac{1}{\gamma^2}$$

That’s all folks ☺