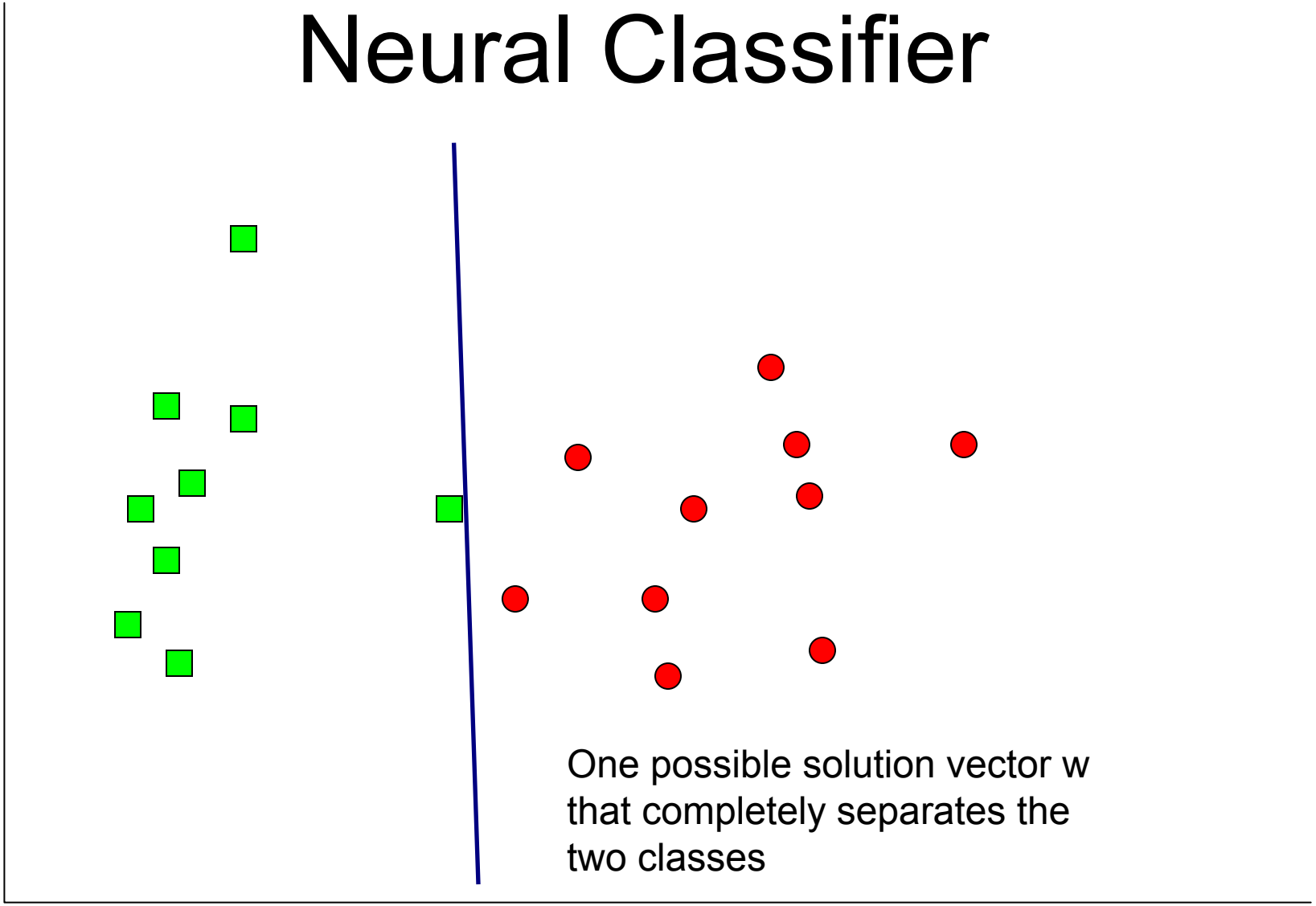
Prof. Marios Savvides

Pattern Recognition Theory

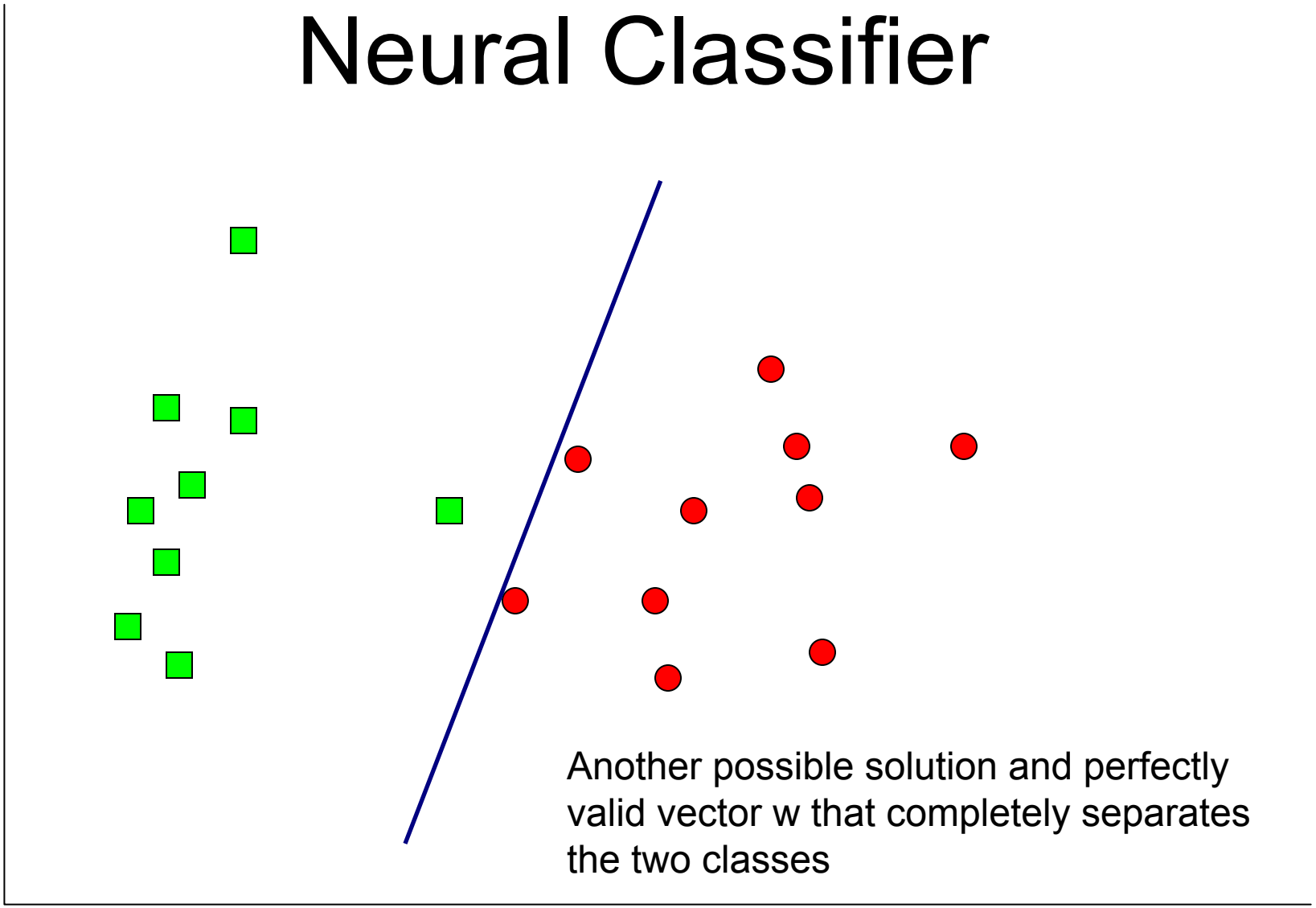
Lecture 11 : Support Vector Machines (SVMs)

Neural Classifier



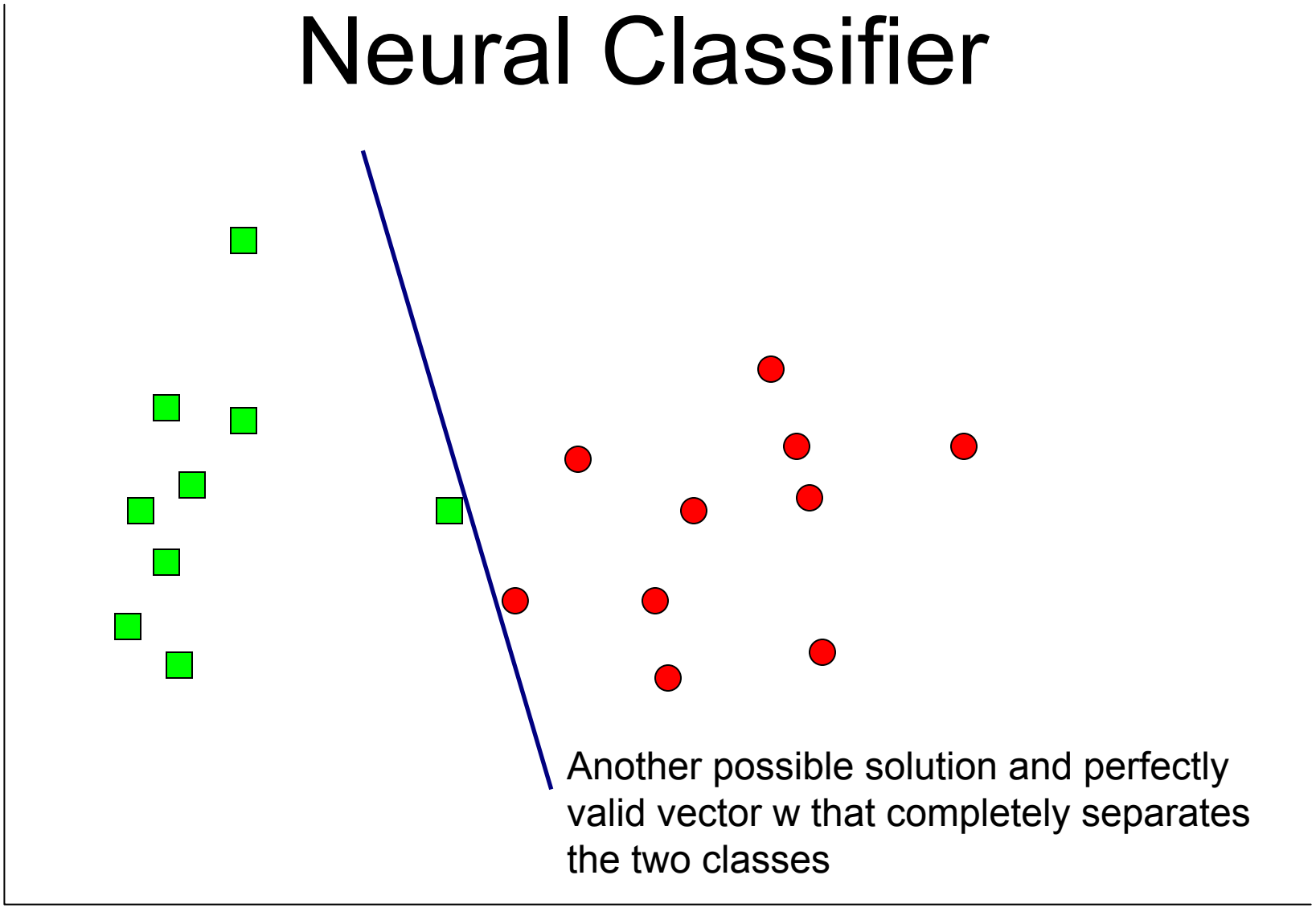
Marios Savvides

Neural Classifier



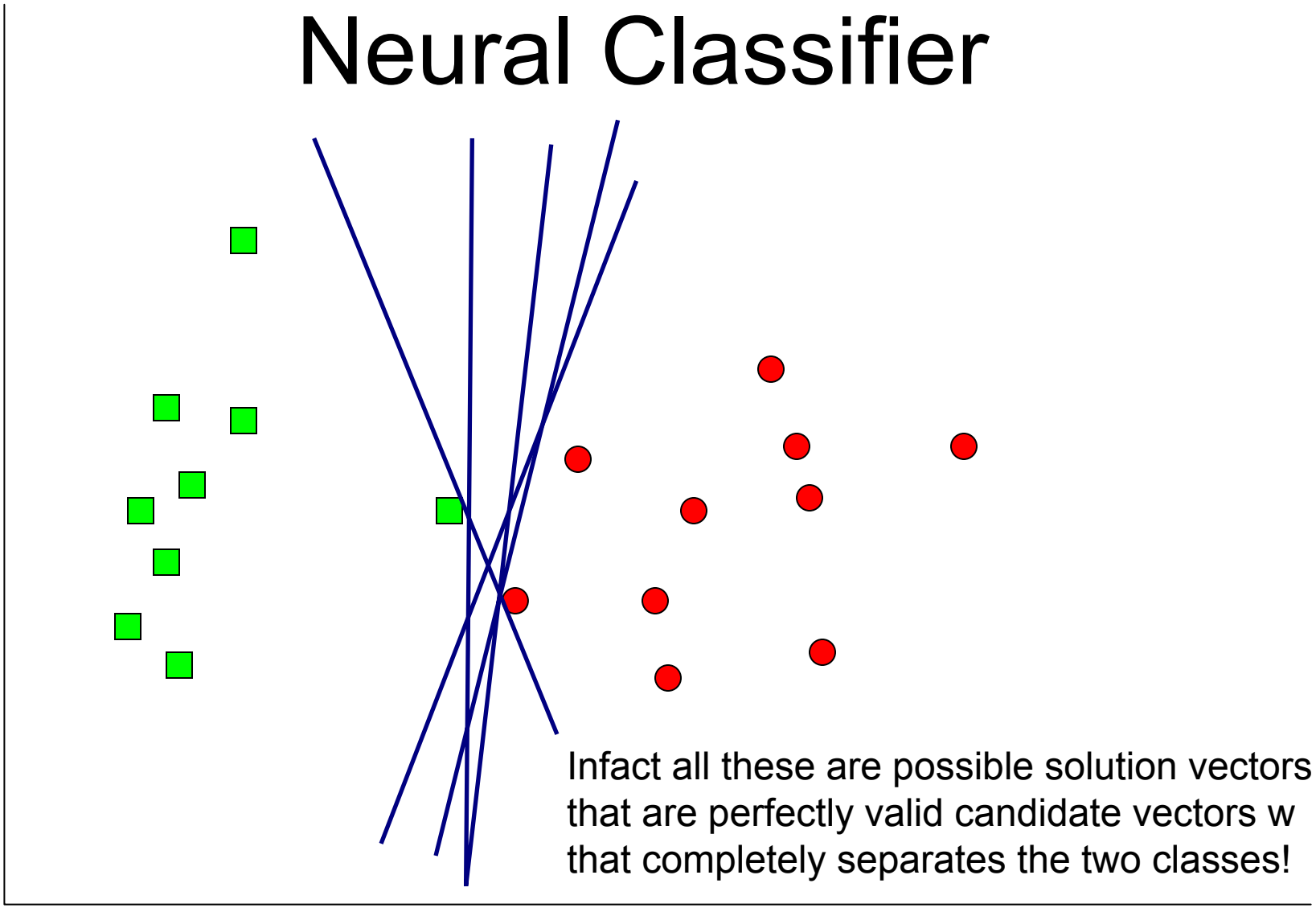
Marios Savvides

Neural Classifier



Marios Savvides

Neural Classifier



Infact all these are possible solution vectors that are perfectly valid candidate vectors w that completely separates the two classes!

Neural Classifiers

- If data is separable, you will always converge to candidate solution vector w which completely separates the classes (perceptron, Ho-Kashyap, MLP, NNs)
- Objective function is to find vector that separates the two classes, and as we have seen there are many vectors that satisfy this criteria.

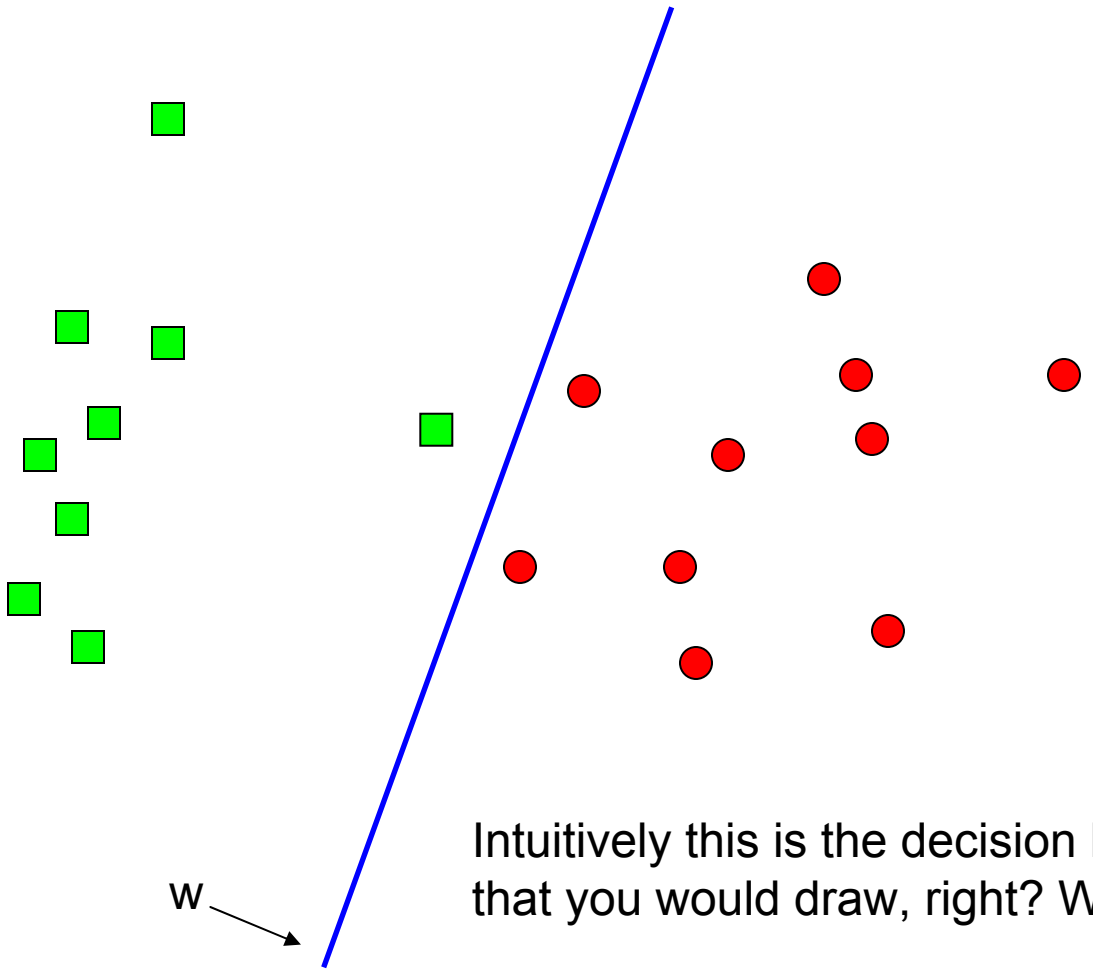
Neural Classifiers

- Neural Nets in general converge to different solution due to the initial weight vectors (usually random).
- That is why in practice (in the old times where we used NN everyday), you would re-initialize NNets and re-train to see if the generalization solution vector that converged offered better generalization (on the cross-validation set).
- So what do we learn, we need an objective function that offers better “generalization”.

Neural Net Generalization

- What do we mean by generalization?
  - We want the hyperplane or decision boundary defined by the solution vector w to not only separate the training vectors but also work well with un-seen test data.
- So…what would you do?

Support Vectors



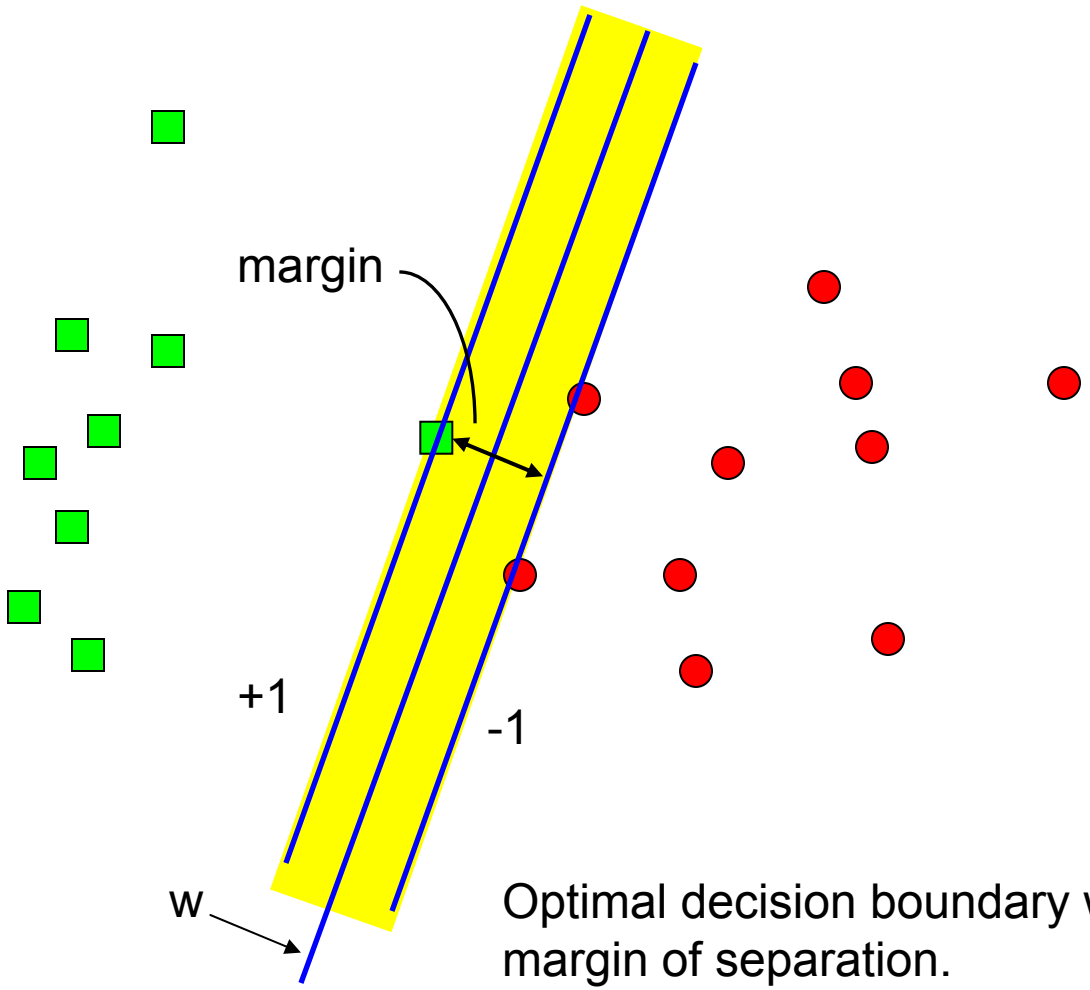
Intuitively this is the decision boundary that you would draw, right? Why?

Marios Savvides

Support Vector Machines

- Goal is to improve generalization.
- How?
- By finding a decision boundary solution vector that “Maximizes” the margin boundary between the two classes.
- So..we don’t care about data samples that are easily classified (i.e. the samples away from the margin). We care about where the errors will lie..i.e. near the margin..which of the data samples are affected.

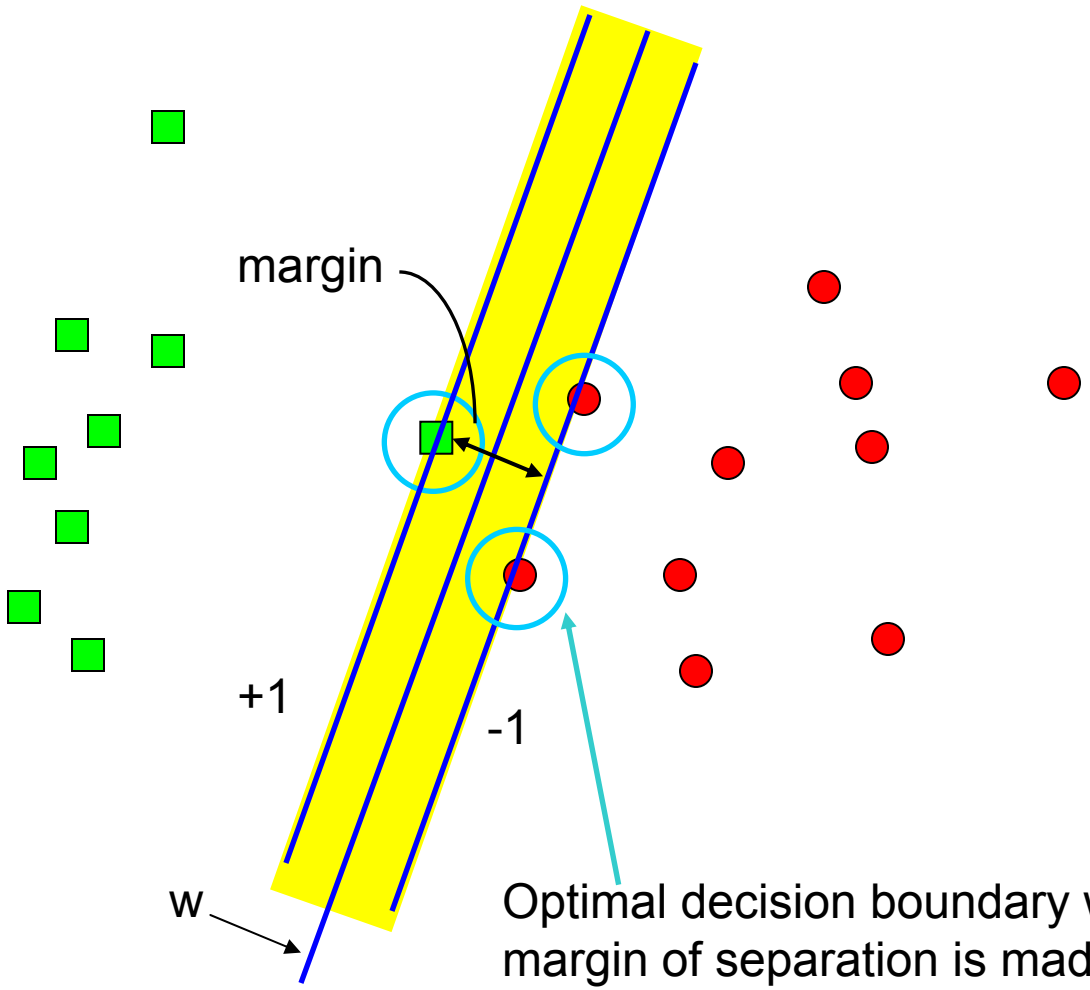
Support Vectors



Optimal decision boundary with maximum margin of separation.

Marios Savvides

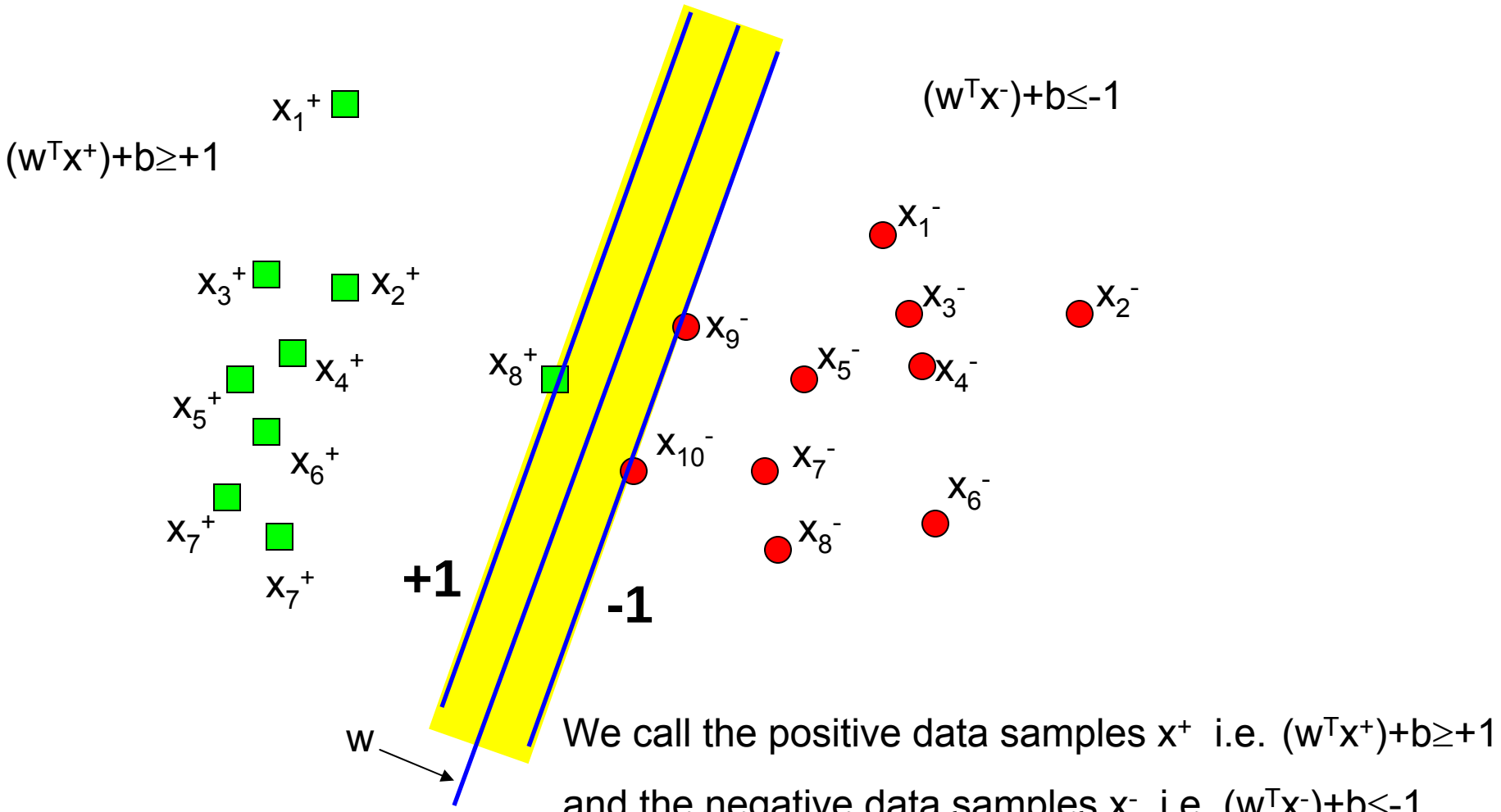
Support Vectors



Optimal decision boundary with maximum margin of separation is made from the support vectors (that *support* this margin)

Marios Savvides

Support Vectors

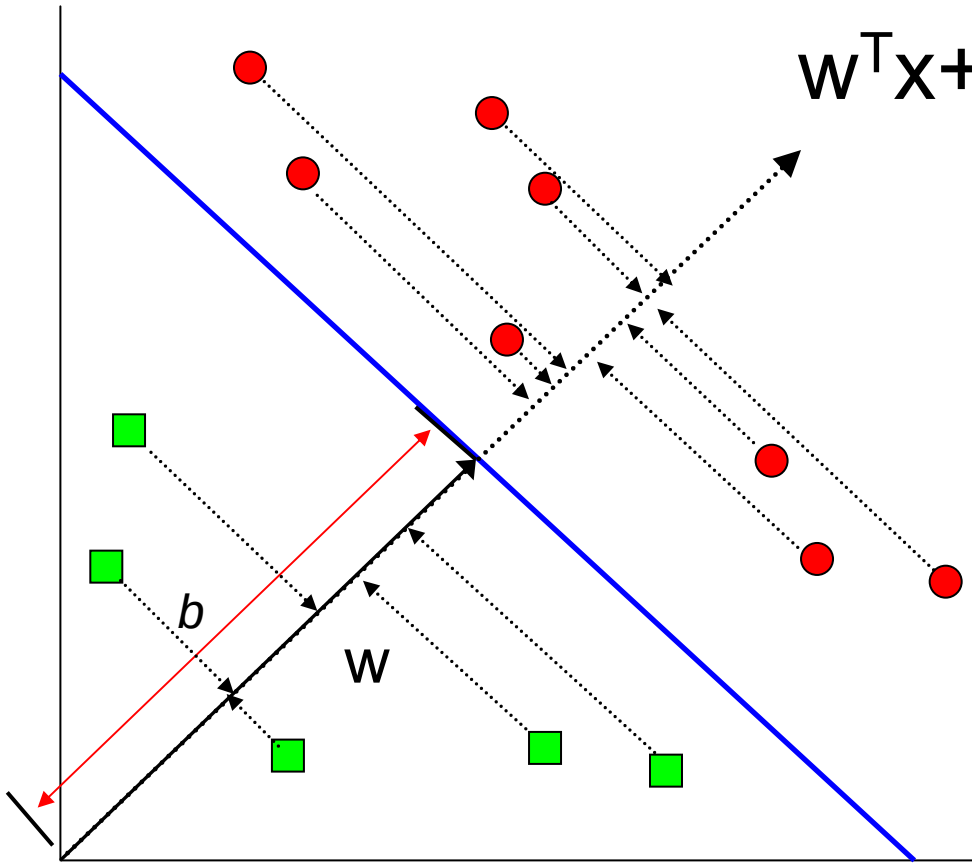


We call the positive data samples x^+ i.e. $(w^T x^+)+b \geq +1$

and the negative data samples x^- i.e. $(w^T x^-)+b \leq -1$

Marios Savvides

Linear Discriminant - revisited

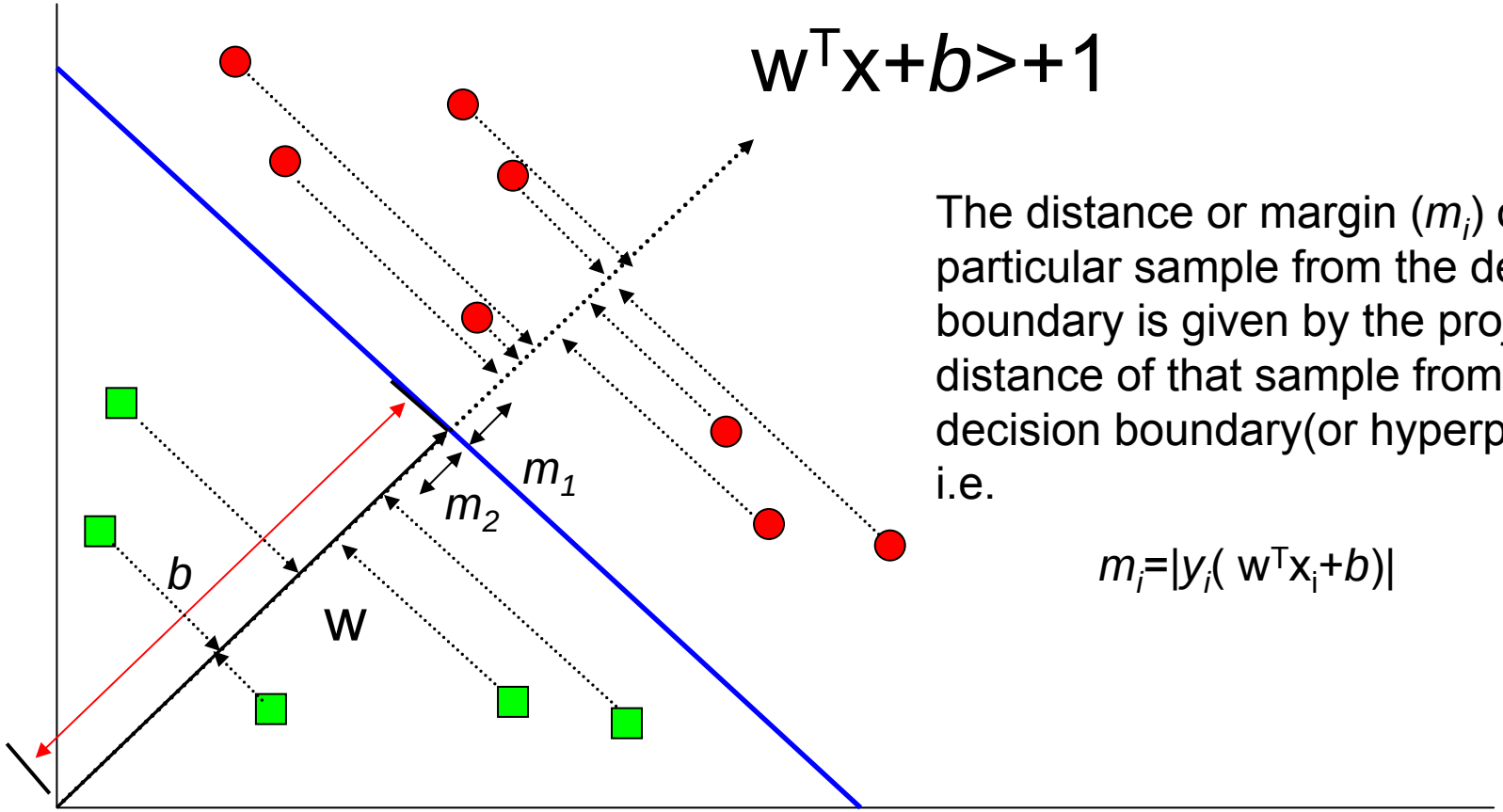


The decision boundary is orthogonal (perpendicular/normal) to the solution vector w.

Since we project samples onto w, we threshold the distance/position where these samples fall. That is why we need offset *b to* shift the decision boundary in the w direction.

$y_i\ (w^+x+b) \geq 0 \quad y_i=[-1, +1]$

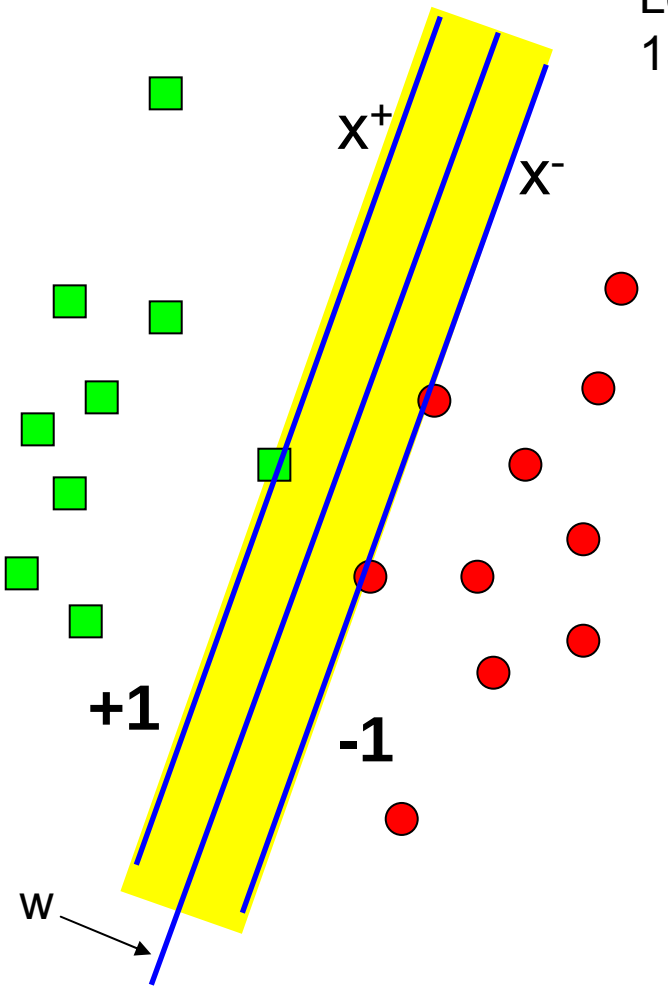
Linear Discriminant - margins



The distance or margin (m_i) of a particular sample from the decision boundary is given by the projected distance of that sample from the decision boundary(or hyperplane) i.e.

$$m_i=|y_i(w^T x_i+b)|$$

Support Vectors



Let x^+ denote a positive point with functional margin of 1 and x^- denote a negative point respectively.

This implies:

$w^T x^+ + b = +1$

$w^T x^- + b = -1$

The functional margin of the resulting classifier m is

$$m = \left(\left\langle \frac{\mathbf{w}}{\|\mathbf{w}\|}, \mathbf{x}^+ \right\rangle - \left\langle \frac{\mathbf{w}}{\|\mathbf{w}\|}, \mathbf{x}^- \right\rangle \right)$$

$$= \frac{1}{\|\mathbf{w}\|} \left(\langle \mathbf{w}, \mathbf{x}^+ \rangle - \langle \mathbf{w}, \mathbf{x}^- \rangle \right)$$

$$= \frac{2}{\|\mathbf{w}\|}$$

Recall

- If we want to **maximize** the margin $\frac{2}{\|\mathbf{w}\|}$ is

equivalent to **minimizing** $\langle \mathbf{w}, \mathbf{w} \rangle = \mathbf{w}^{\mathbf{T}}\mathbf{w}$

Thus for a support vector machine we want a hyperplane that maximizes this margin!!

Support Vector Optimization Criteria

- So…we want to maximize the margin by minimizing

$$\mathbf{w}^{\mathrm{T}}\mathbf{w}$$

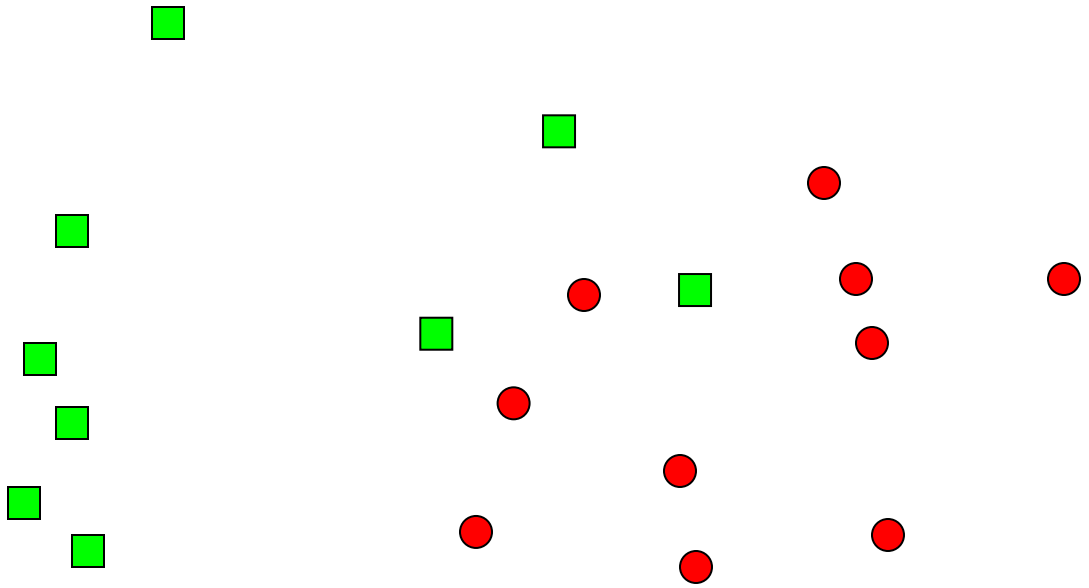
Subject the inequality constraints

$$y_i\left(\langle \mathbf{w}, \mathbf{x}_i \rangle + b\right) \geq 1$$

i.e. positive samples must fall on the positive side (vice-versa for negative data samples) y_i=[-1,+1]

- Can be solved via Quadratic Programming (QP)..programs exist in matlab..never need to write them yourself!

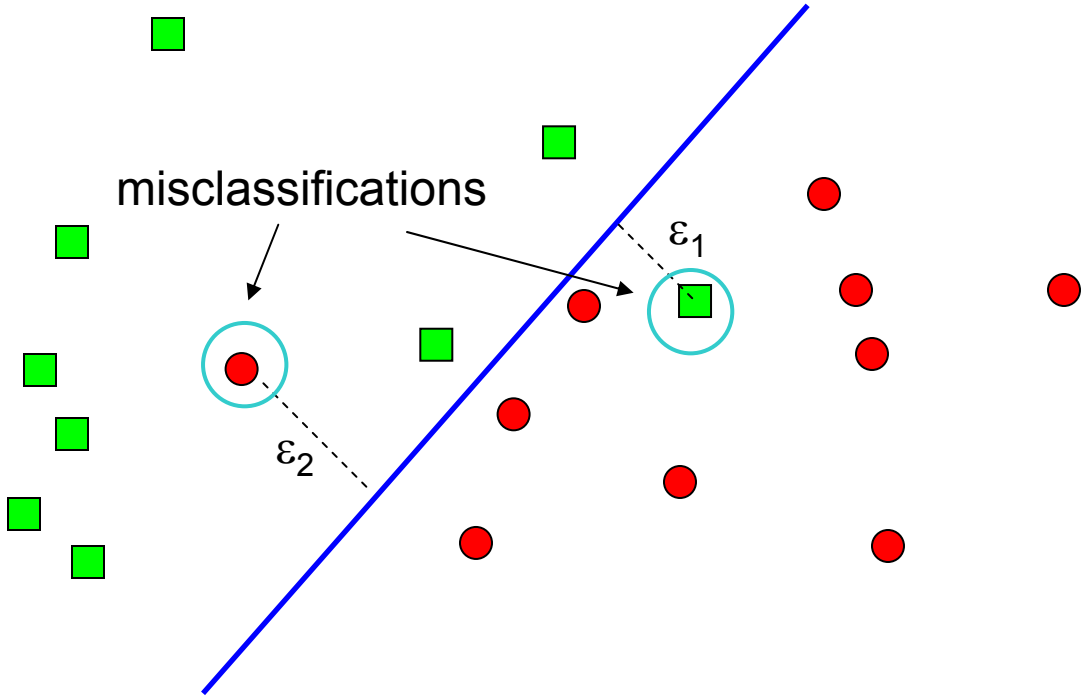
Houston…we have a problem!



- What happens when data is non-separapable?

We can not draw a linear hyperplane to separate the data..so..do we pack our bags and go home?....ofcourse not….atleast not yet..

Introduce error metric ε



- We still can have a hyperplane that can not fully separate the true classes but tries to maximize margin while minimizing the total error $\sum \varepsilon_i$

Slack variables

- We call the error distances ε the ‘slack’ variables.

- We want to cut some slack to the support vector machine so they can get the job done.

So what do we optimize now?

- In non-separapable cases we still want to maximize the margin but we also want to minimize the error distances.

- So…. Our optimization formulation says we:
- MINIMIZE

$$\frac{1}{2}\mathbf{w}^{\mathbf{T}}\mathbf{w} + C\sum_{i=1}^{N}\varepsilon_i$$

Is that all? What about our inequality constraints?

- We add some slack to our data so we minimize:

$$\frac{1}{2}\mathbf{w}^{\mathbf{T}}\mathbf{w} + C\sum_{i=1}^{N}\varepsilon_i$$

- Subject to the following inequality constraints:

$$\mathbf{w}^{\mathbf{T}}\mathbf{x}_i + b \geq 1 - \varepsilon_i \qquad for \quad +ve \quad class$$

$$\mathbf{w}^{\mathbf{T}}\mathbf{x}_i + b \leq -1 + \varepsilon_i \qquad for \quad -ve \quad class$$

$$\varepsilon_i > 0 \qquad \forall i$$

A Dual problem in a dual universe?

- The optimization problem set-up was is whats called its ‘primal’ form.
- We can re-formulate the optimization problem of maximizing margin in what’s called a ‘Dual’ form..why?
- We can do cool things, like use Kernels that allow us to work in a higher dimensional space (without actually computing this space). – whats called the kernel trick….but first…..

Primal Langrangian Form

- Primal form:

$$L(\mathbf{w}, b, \alpha) = \tfrac{1}{2}\mathbf{w}^{\mathbf{T}}\mathbf{w} - \sum_{i=1}^{N} \alpha_i \left[y_i(\mathbf{w}^{\mathbf{T}}\mathbf{x}_i + b) - 1 \right]$$

margin

constraints

- Where α_i are the lagrange multipliers $\alpha_i \geq 0$
- We want to optimize this function

Primal Langrangian Form contd..

- Primal form:

$$L(\mathbf{w}, b, \alpha) = \tfrac{1}{2}\mathbf{w}^{\mathbf{T}}\mathbf{w} - \sum_{i=1}^{N} \alpha_i \left[y_i(\mathbf{w}^{\mathbf{T}}\mathbf{x}_i + b) - 1 \right]$$

- Differentiate with respect w to find

$$\frac{L(\mathbf{w}, b, \alpha)}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i = \mathbf{0}$$

$$\mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i$$

Primal Langrangian Form contd..

- Primal form:

$$L(\mathbf{w}, b, \alpha) = \tfrac{1}{2}\mathbf{w}^{\mathbf{T}}\mathbf{w} - \sum_{i=1}^{N} \alpha_i \left[y_i(\mathbf{w}^{\mathbf{T}}\mathbf{x}_i + b) - 1 \right]$$

- Differentiate with respect b to find

$$\frac{L(\mathbf{w}, b, \alpha)}{\partial b} = \sum_{i=1}^{N} \alpha_i y_i = 0$$

$$\sum_{i=1}^{N} \alpha_i y_i = 0$$

Dual Langrangian Form contd..

- Primal form:

$$L(\mathbf{w}, b, \alpha) = \tfrac{1}{2}\mathbf{w}^{\mathbf{T}}\mathbf{w} - \sum_{i=1}^{N} \alpha_i \left[y_i(\mathbf{w}^{\mathbf{T}}\mathbf{x}_i + b) - 1 \right]$$

- Substitute $\mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i$ back into above Eq.

$$L(\mathbf{w}, b, \alpha) = \tfrac{1}{2}\mathbf{w}^{\mathbf{T}}\mathbf{w} - \sum_{i=1}^{N} \alpha_i \left[y_i(\mathbf{w}^{\mathbf{T}}\mathbf{x}_i + b) - 1 \right]$$

$$= \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} y_i y_j \alpha_i \alpha_j \left\langle x_i, x_j \right\rangle - \sum_{i=1}^{N}\sum_{j=1}^{N} y_i y_j \alpha_i \alpha_j \left\langle x_i, x_j \right\rangle + \sum_{i=1}^{N} \alpha_i$$

$$= \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} y_i y_j \alpha_i \alpha_j \left\langle x_i, x_j \right\rangle$$

What do we optimize?

- Dual form:

$$\sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} y_i y_j \alpha_i \alpha_j \left\langle x_i, x_j \right\rangle$$

- Subject to the constraints $\sum_{i=1}^{N} \alpha_i y_i = \mathbf{0}$ and $\alpha_i \geq 0$

- Again this is solvable with QP.

Ok…what the important thing to get from this Dual formulation?

- Remember when we optimized the dual lagrangian we got

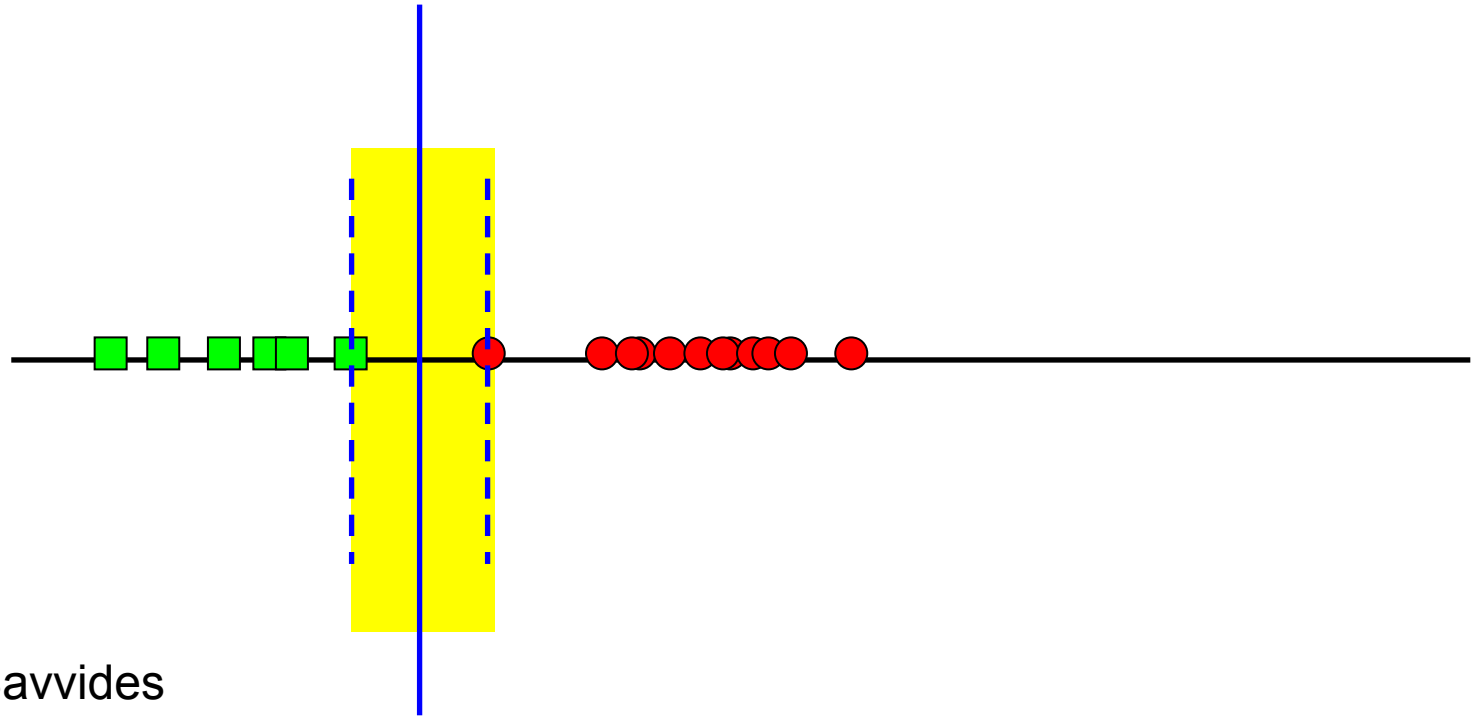
$$\frac{L(\mathbf{w}, b, \alpha)}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i = \mathbf{0}$$

$$\mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i \qquad \alpha_i \geq 0 \qquad y_i \in \{-1, +1\}$$

- The important thing to note is that the optimal hyperplane w with maximum margin is a linear combination of the training sets. More importantly it is linear combination of the support vectors (the lagrange multipliers a_i specify which samples to use and how much or it can be 0.

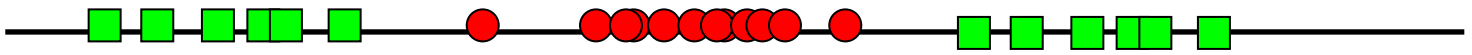
Example SVM

- No problemo….



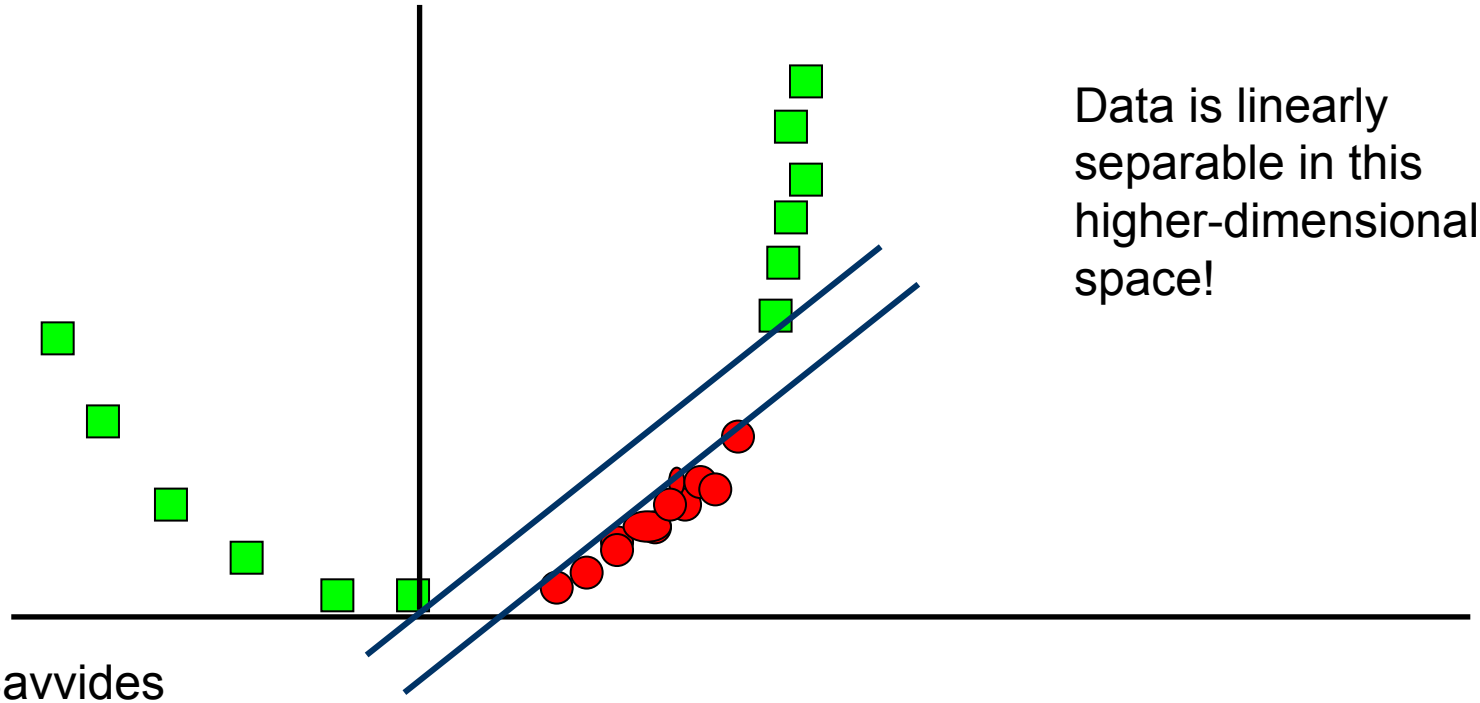
Example SVM

- Major problemo….what now?



Example SVM

- Remember permitting a non-linear transformation to higher dimensional space. $F_k = (x_k, x_k^2)$



Data is linearly separable in this higher-dimensional space!

Wooow…now…Hold on a second..

- Don’t we usually say….we want to reduce the dimensionality of data in pattern recognition problems?
- Correct..PCA does dimensionality reduction…but as we just saw there are cases where the data is NOT separable and we need to map our data into a *higher-dimensional space* where it might be linearly separable!

Mapping functions

$$\phi : X \rightarrow F$$

This is a non-linear map from the input space to some feature space. Looking back to our SVM linear machine we now have:

$$\sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} y_i y_j \alpha_i \alpha_j \left\langle \Phi(x_i), \Phi(x_j) \right\rangle$$

$$\mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i \Phi(\mathbf{x}_i)$$

Project on to direction vector

$$\mathbf{w}^{\mathbf{T}} \Phi(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i y_i \left\langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}\ ) \ \right\rangle$$

Mapping functions

$$\phi : X \rightarrow F$$

This is a non-linear map from the input space to some feature space. Looking back to our SVM linear machine we now have:

$$\sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} y_i y_j \alpha_i \alpha_j \left\langle \Phi(x_i), \Phi(x_j) \right\rangle$$

$$\mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i \Phi(\mathbf{x}_i)$$

Project on to direction vector

$$\mathbf{w}^{\mathbf{T}} \Phi(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i y_i \left\langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}) \right\rangle$$

This is want we need to compute efficiently without actually having to form the mapping.

Mapping functions

$$\phi : X \rightarrow F$$

This is a non-linear map from the input space to some feature space. Looking back to our SVM linear machine we now can define a kernel function K(x,y)

$$K(\mathbf{x}, \mathbf{y}) = \left\langle \Phi(\mathbf{x}), \Phi(\mathbf{y}\ ) \ \right\rangle$$

What are examples of some commonly used kernel functions?

The efficiency trick is that we need to compute K(x,y) without actually compute the mapping of our data into higher dimensions (i.e. compute K(x,y) without having to form Φ(x) and Φ(y) as computation and memory may be too great! – Kernel's must satisfy Mercer's condition to ensure that kernel actually forms a dot product in some higher dimensional space.

Example Kernel Functions

- Polynomial

  $K(a,b)= (<a,b>+1)^p$

Expand $<a,b>^2=<(a(1)b(1)+a(2)b(2))^2$ $\rightarrow$ ($R^2 \rightarrow R^3$)

$= [a(1)b(1)]^2+2a(1)b(1)a(2)b(2)+[a(2)b(2)]^2$

- Radial basis Style Kernel Function

  $K(a,b)=\exp(- (a-b)^2 / 2\sigma^2)$

- Neural Net Style Kernel Function

  $K(a,b)=\tanh(k <a,b>-\delta)$