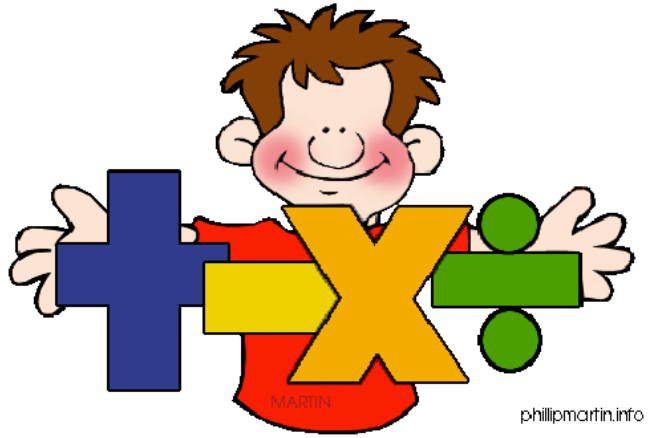
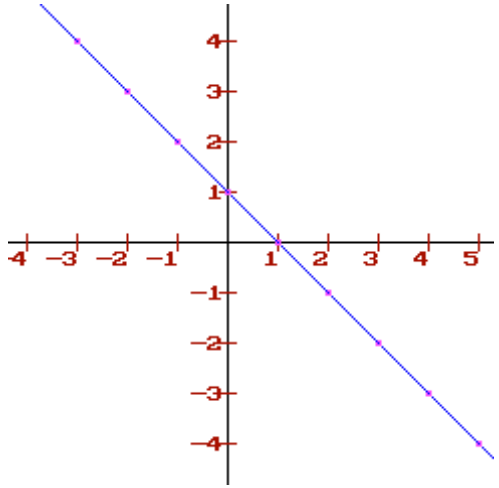
Support Vector Machines

Some basic math



- Equation of a hyperplane (straight line in 2D):

$$w^T x + w_0 = 0$$

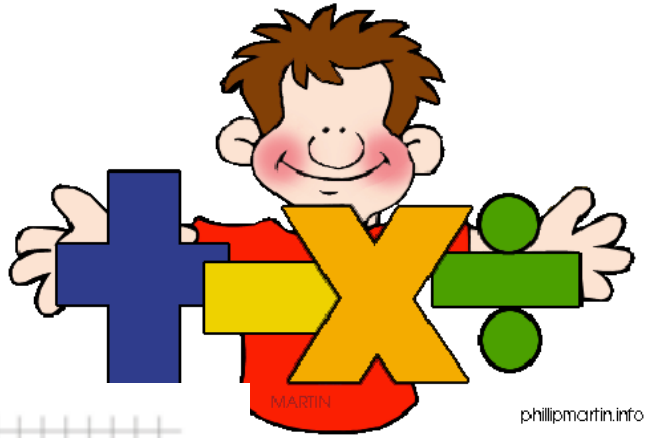


For example, $x_1 + x_2 - 1 = 0$

What is the normal vector to this plane?
=> Simply the vector w

How? Take two points on the line and take dot product with w and you get 0

Some basic math

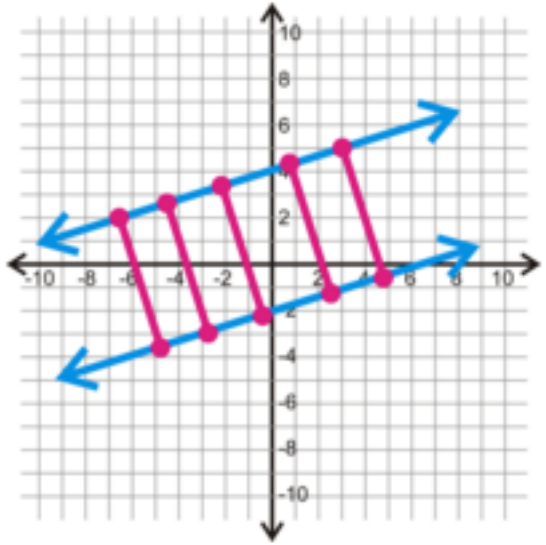


- What is the distance between two parallel lines?

$$ax + by + c_1 = 0$$
$$ax + by + c_2 = 0,$$

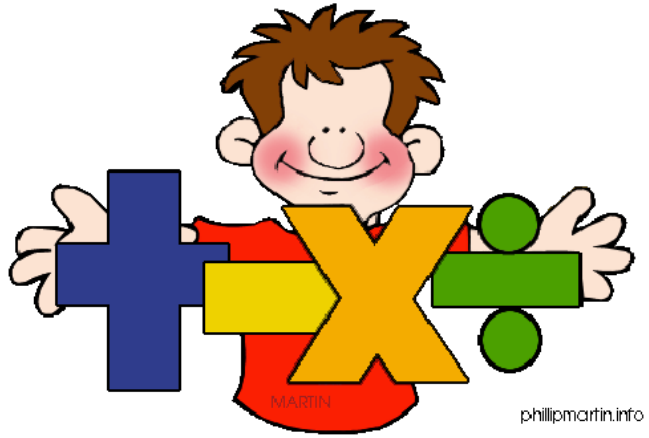
Distance is given by:

$$d = \frac{|c_2 - c_1|}{\sqrt{a^2 + b^2}}.$$



https://en.wikipedia.org/wiki/Distance_between_two_straight_lines

Some basic math



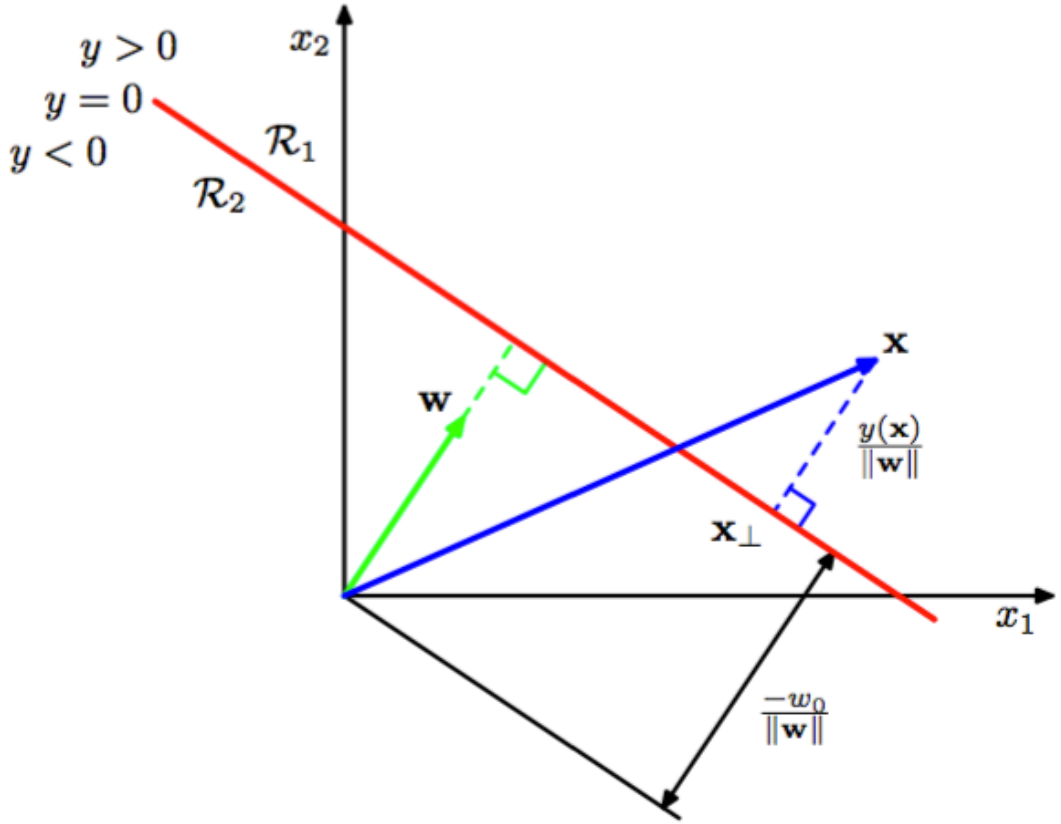
- Let us consider the equation

$$y = w^T x + w_0$$

For the decision surface, y = 0

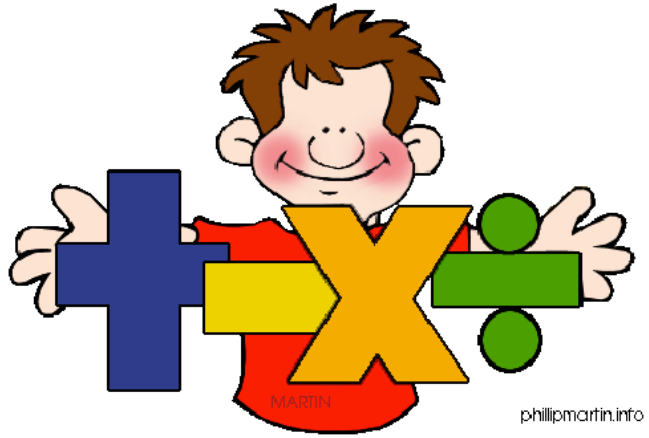
y > 0 for class +

y < 0 for class -.



See https://en.wikipedia.org/wiki/Distance_from_a_point_to_a_line

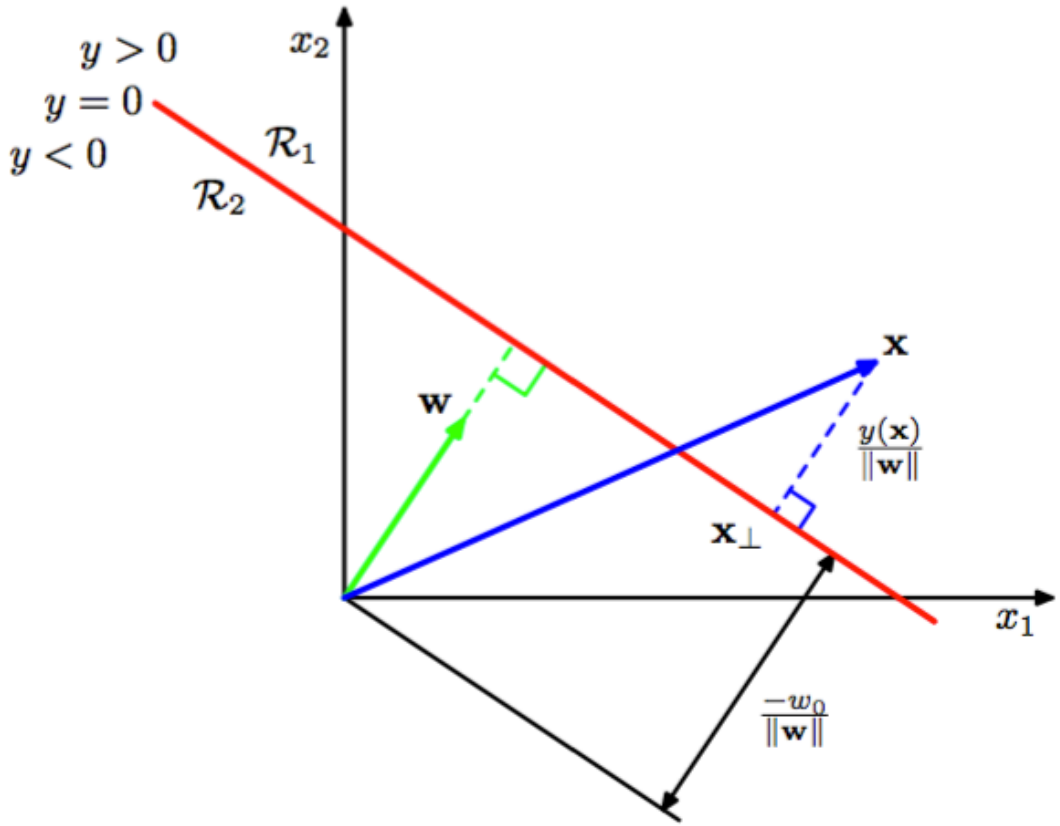
Some basic math



Another property-
What is the perpendicular distance of any point x to this decision surface?

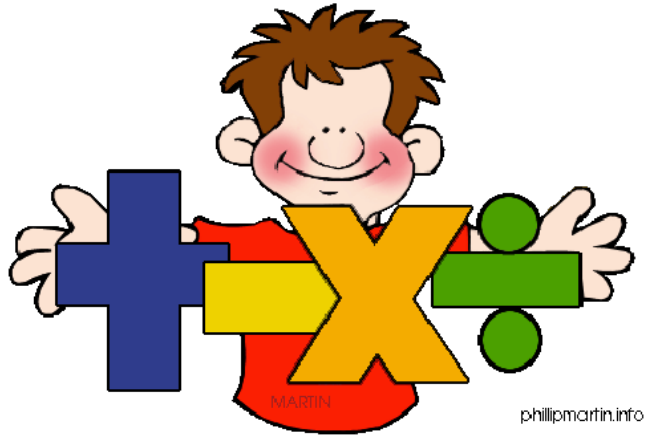
You can easily prove that this is:

$$r = \frac{|y(x)|}{\|w\|}$$



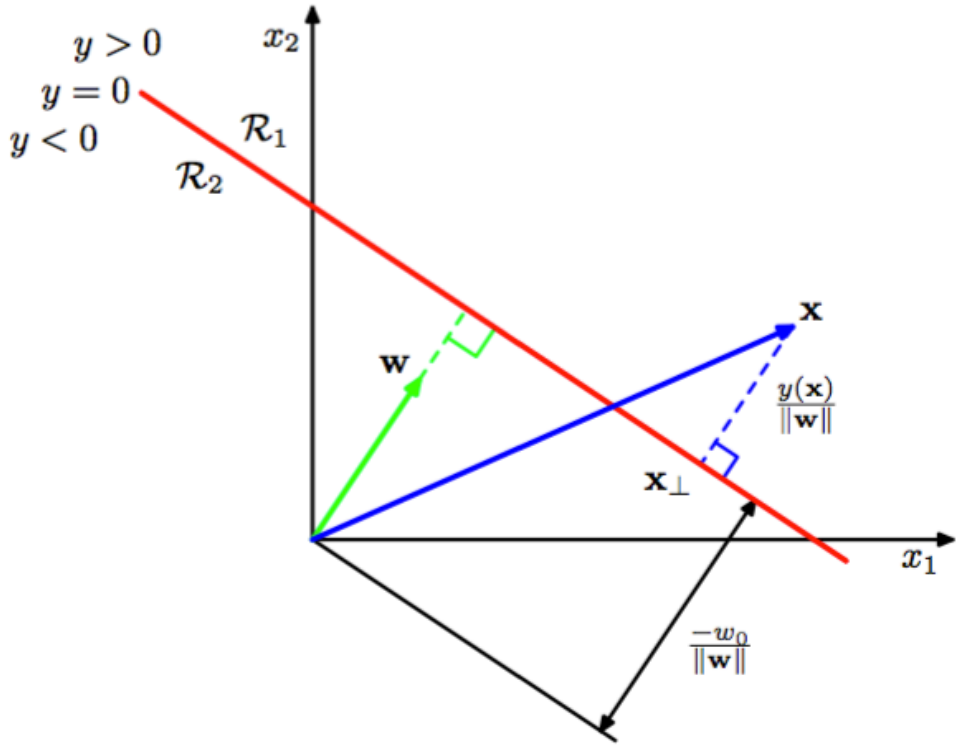
See https://en.wikipedia.org/wiki/Distance_from_a_point_to_a_line

Some basic math



For linearly separable data:
$t_n\, y(x_n) > 0$

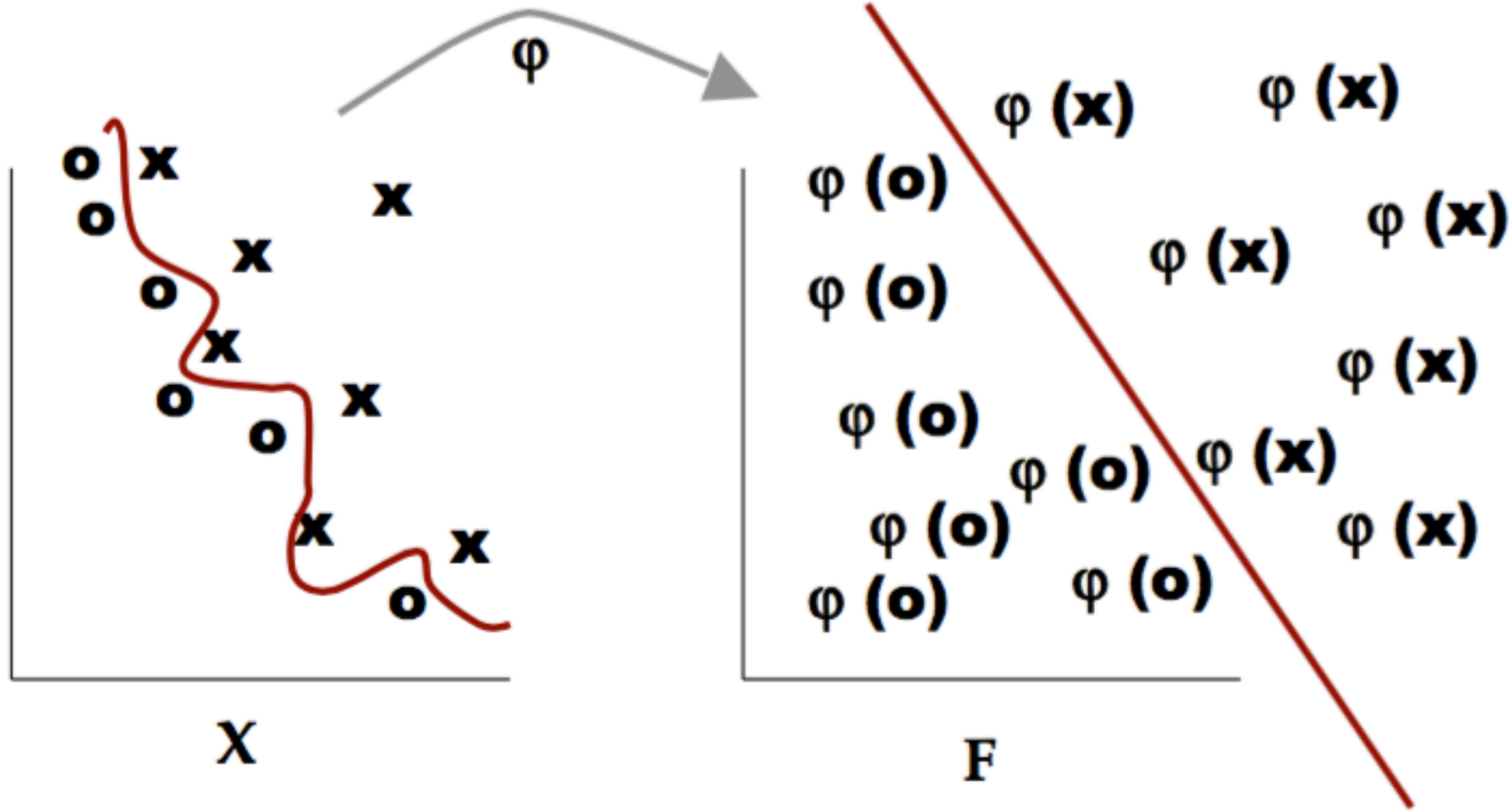
i.e. the target (actual) value and predicted value always have the same sign.



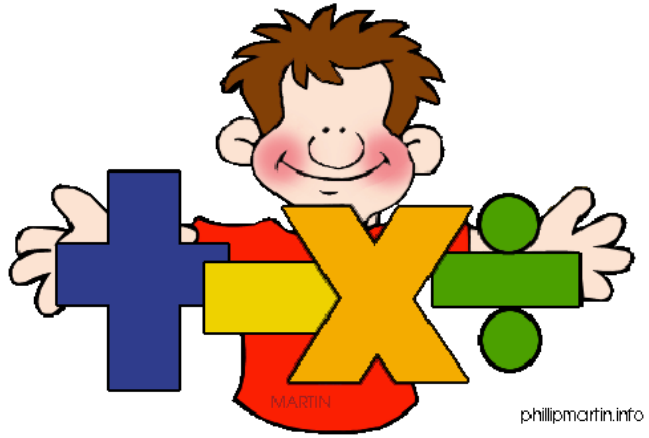
But, what if the data is not linearly separable?

- Can we map it to another space ?
- Transformation x -> ϕ(x)
  e.g. x -> x^2

Transformation to separate



Some basic math

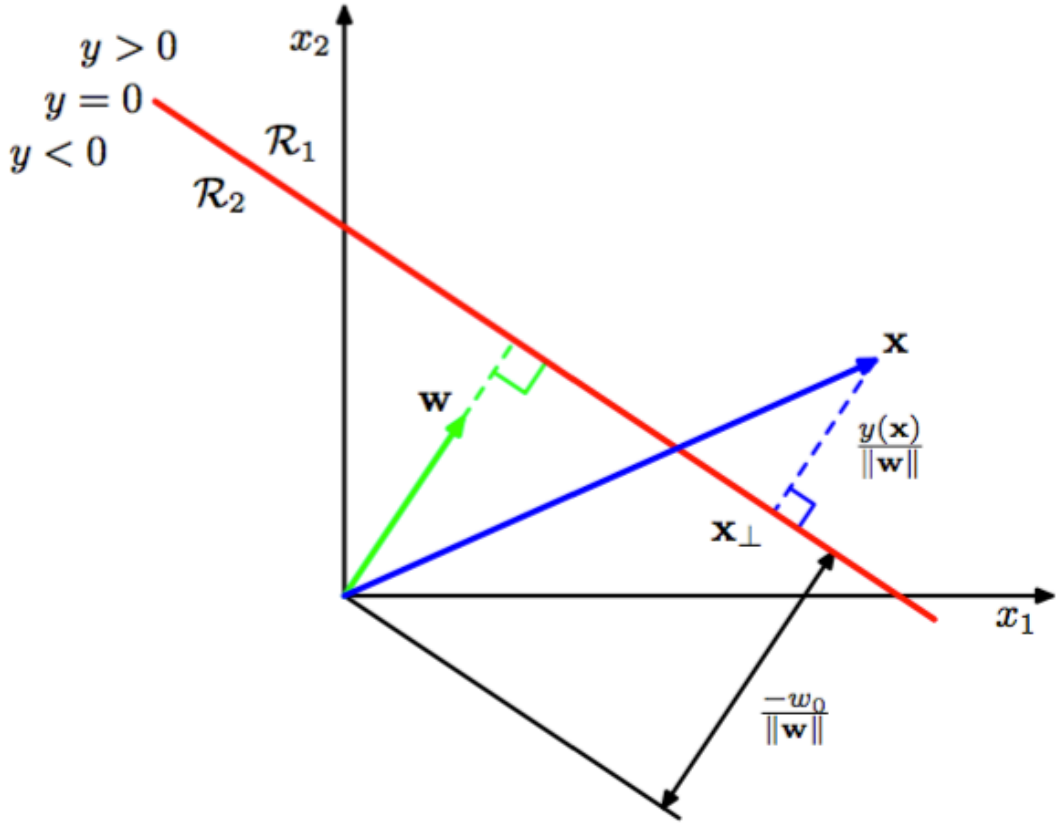


For linearly separable data:
$t_n\, y(x_n) > 0$

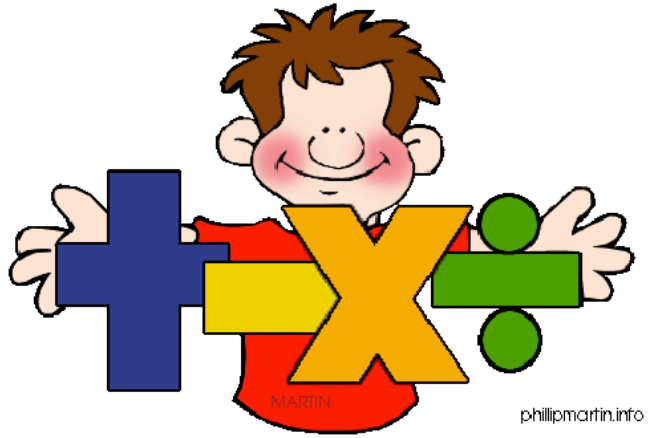
i.e. the target (actual) value and predicted value always have the same sign. So distance can be written as:

$$\frac{|y(x_n)|}{\|w\|} = \frac{t_n y(x_n)}{\|w\|} = \frac{t_n(w^T \emptyset(x) + w_0)}{\|w\|}$$

By transforming to ϕ(x), we get separability



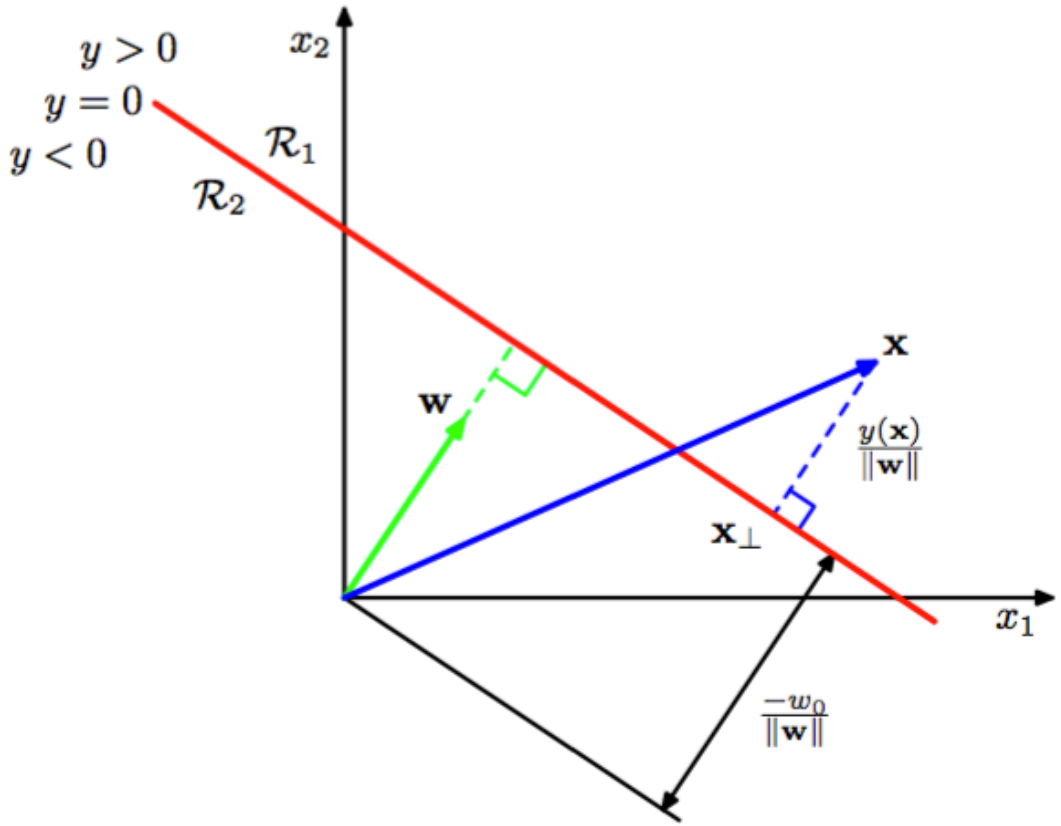
Some basic math



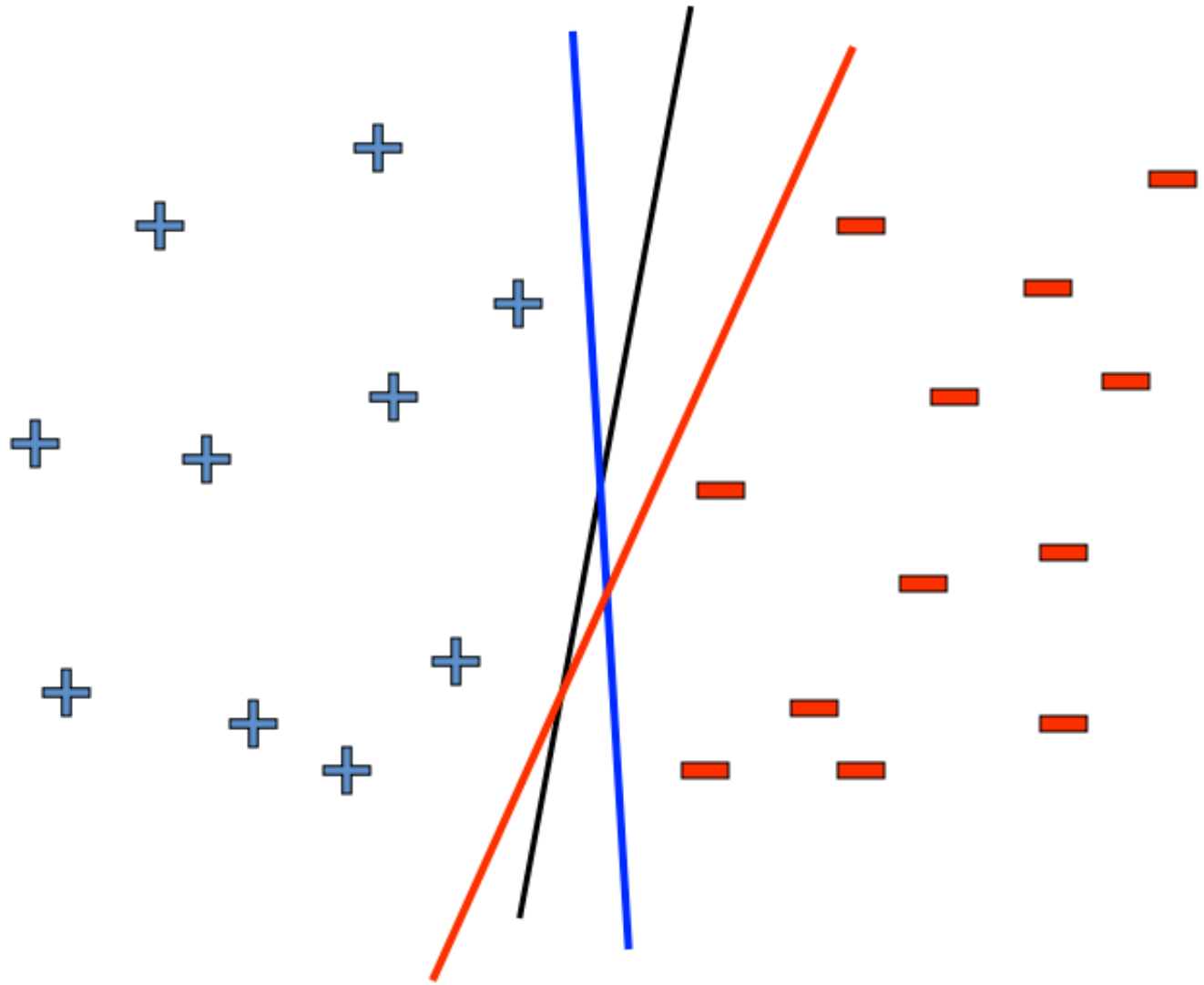
- Distance from (0, 0) to the decision surface:

$$\frac{\mathbf{w}^{\mathrm{T}}\mathbf{x}}{\|\mathbf{w}\|} = -\frac{w_0}{\|\mathbf{w}\|}$$

- OK, fine. BUT…
  Big question – which line do I choose as the decision surface?

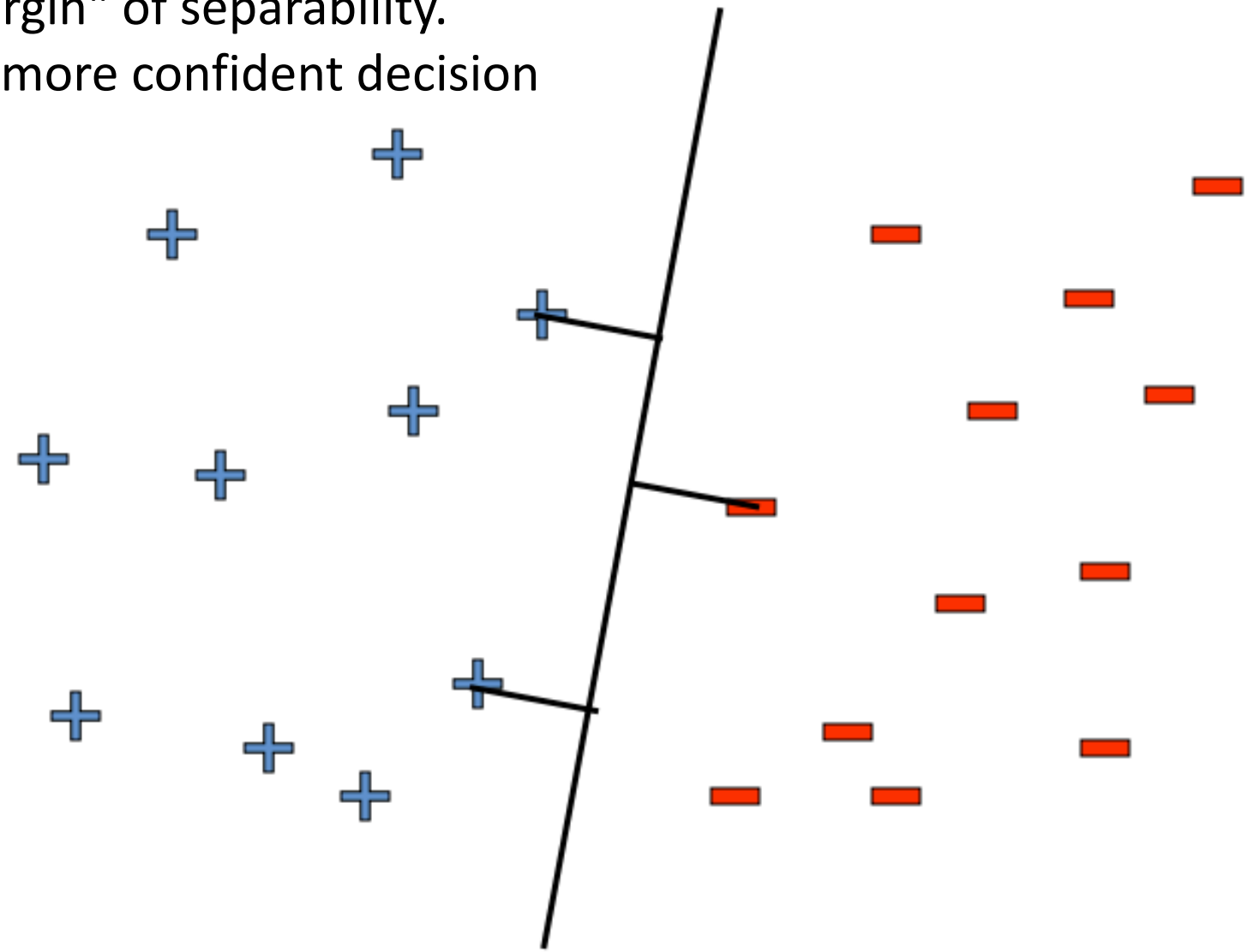


Linear classifiers – which line is better?

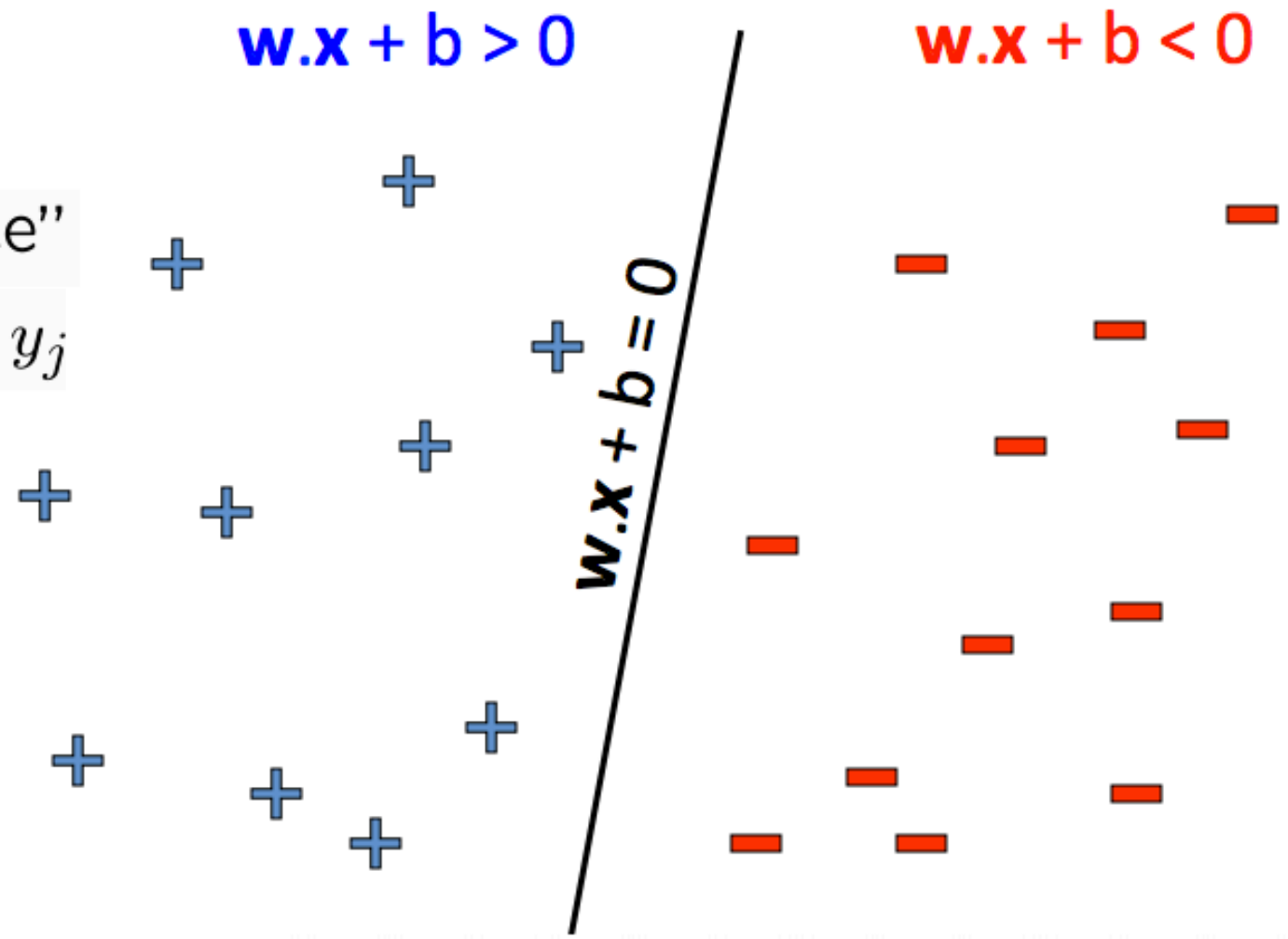


Pick the one with the largest margin!

Note: I want points to be far from this line
to get a good "margin" of separability.
Higher margin => more confident decision



Parameterizing the decision boundary



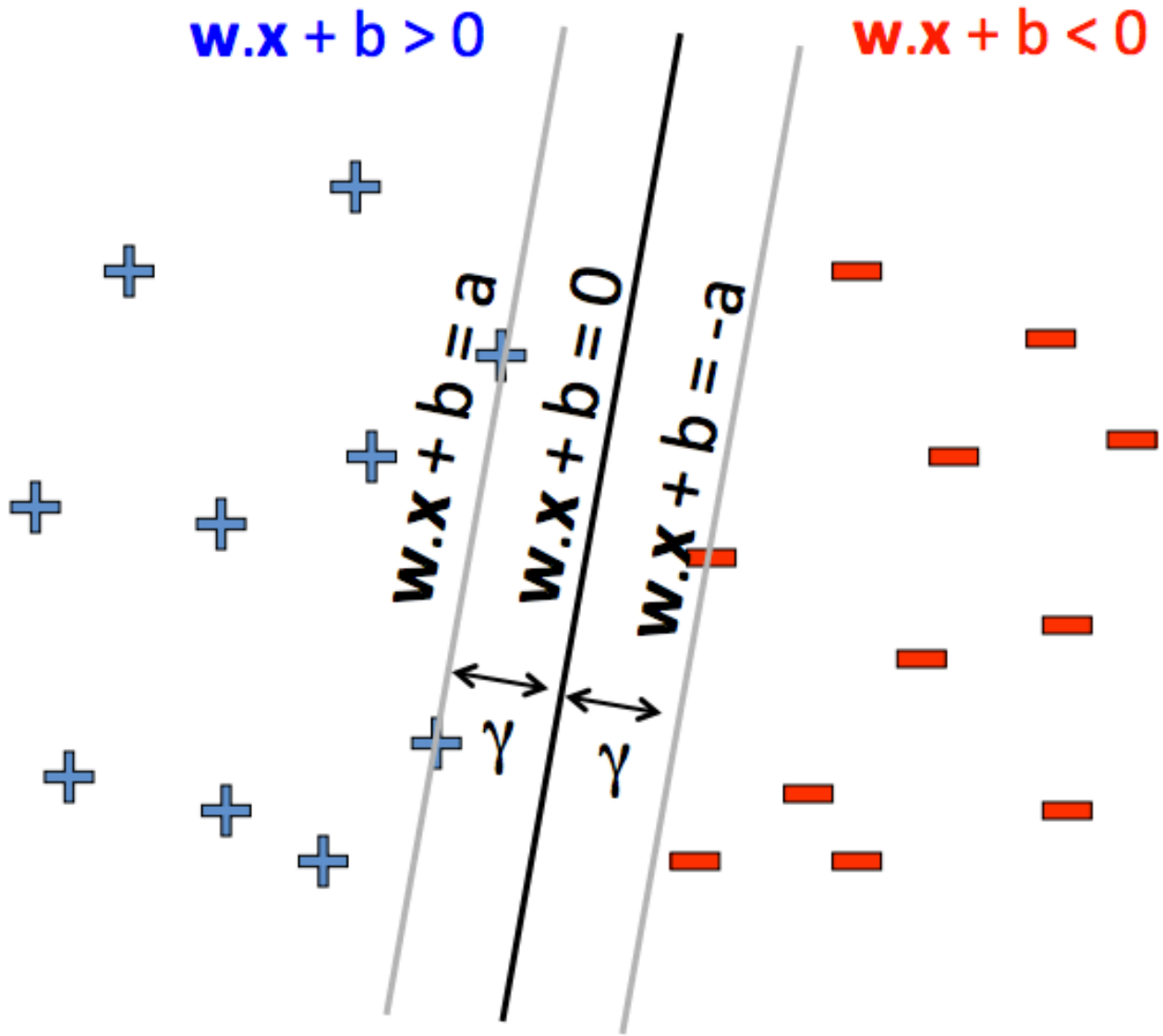
"confidence"

$$\left(\mathbf{w}.\mathbf{x}_j + b\right) y_j$$

Which points are closest to the boundary? They are points that I am concerned about => known as Support Vectors

Labels: $y_i \in \{-1, +1\}$ — class

Maximizing the margin



Distance of closest examples from the line/hyperplane

$$\text{margin} = \gamma = a/\|w\|$$

How did we get this?
Remember earlier slide:
Distance between parallel lines (2 γ):

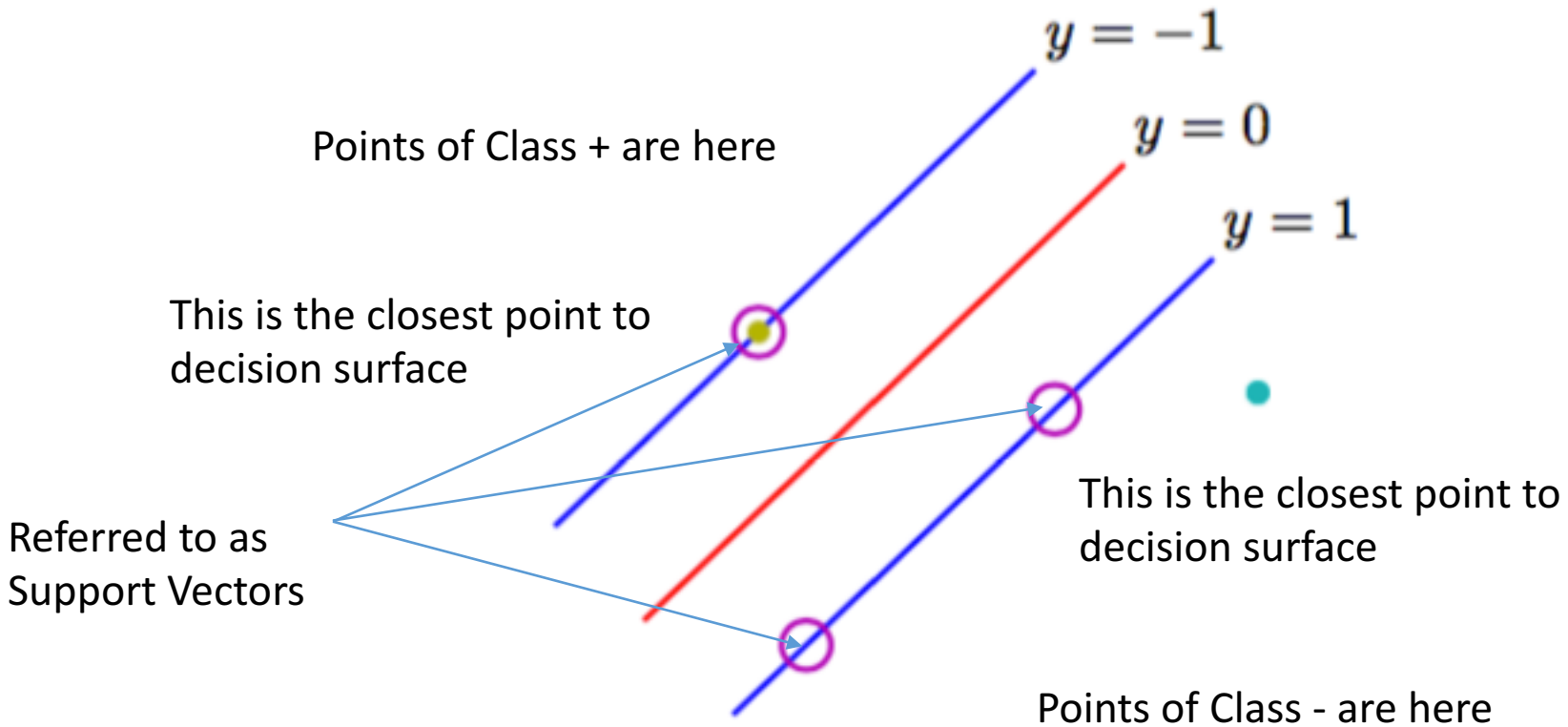
= Difference in intercepts/ ||w||
= a – (-a) / ||w||
= 2a / ||w||

=>

γ = a / ||w||

Margin of Classification

- Without loss of generality, I can set a = 1 (for convenience)

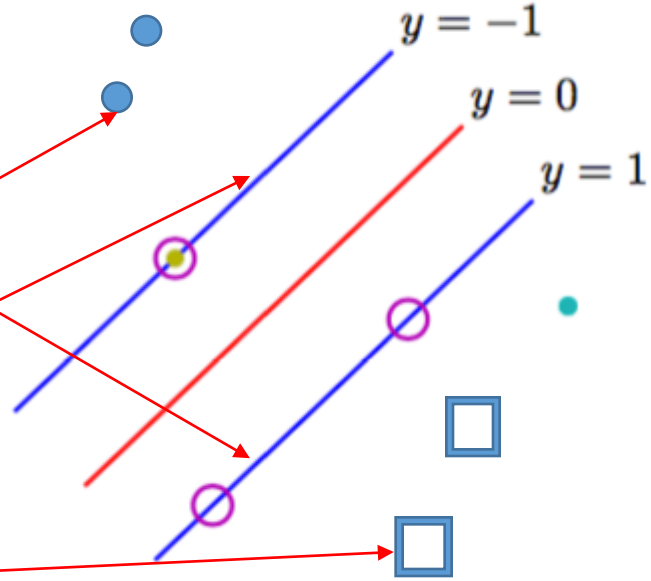


Margin of Classification

- For Support Vectors (SV) on 1 side:

$$y(x) = w^T x + w_0 = 1$$

and on the other side: $y(x) = w^T x + w_0 = -1$



- For other points, |y(x)| > 1
  In fact, larger values of t*y(x)
  are indication of better classification accuracy.
  => They indicate interior points.
- So, the distance on either side to SV is: $\frac{|y(x)|}{\|w\|} = \frac{1}{\|w\|}$

Optimization

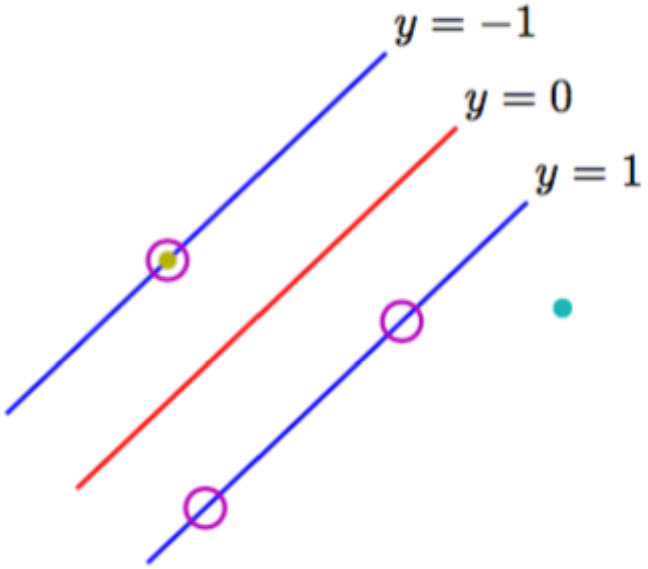
- We want to maximize the margin: 1/||w||
- Are there any constraints?
  - Yes.
  - I want to make sure the points are in the correct area.
    -> Make sure that:
    (Real class) X (Predicted Class) >= 1

$$t_n(w^T x_n + b) \geq 1, \qquad n = 1, 2, \ldots, N$$

Optimization

- We want to maximize: $\frac{1}{\|w\|}$

subject to $t_n(w^T\phi(x_n) + b) \geq 1, \qquad n = 1, 2, \ldots, N$



- We can also transform it into a more mathematically convenient problem:

$$\arg\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2$$

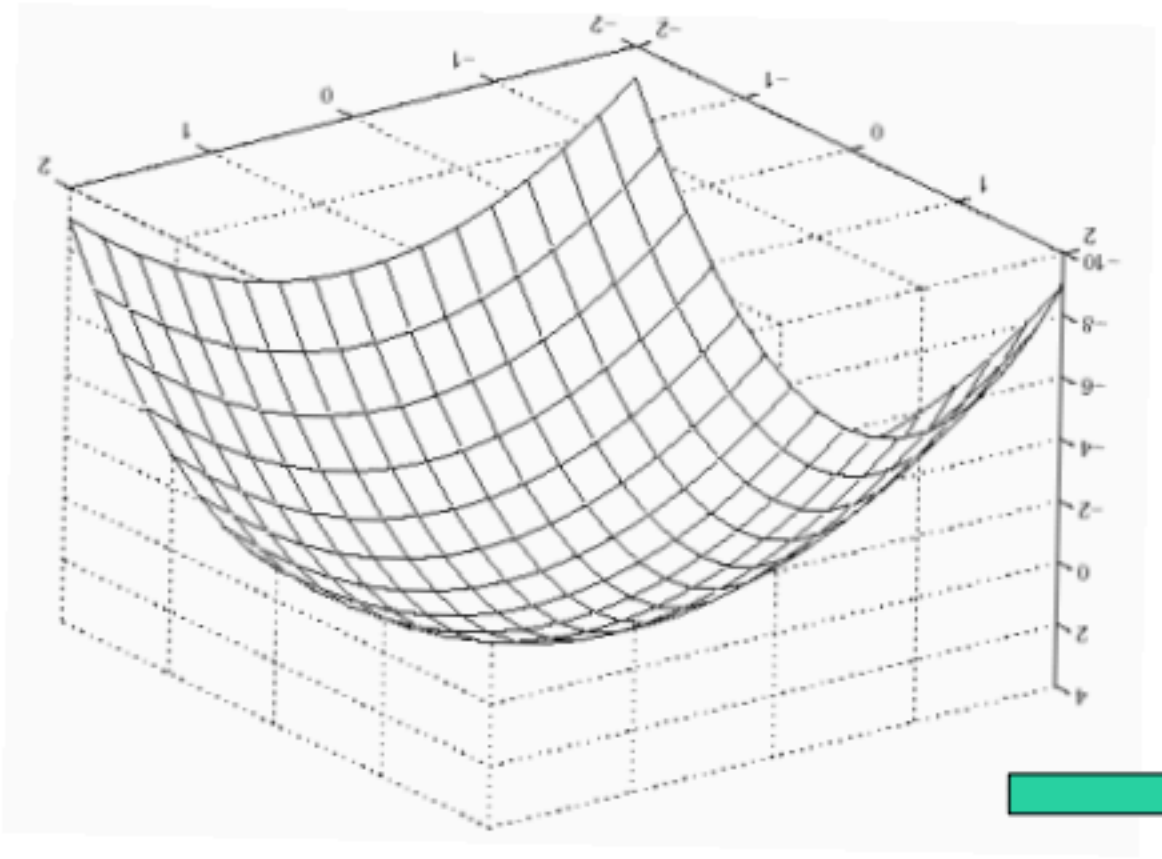
This is the basic idea behind SVM.
One of the most successful classifiers

subject to $t_n(w^T\phi(x_n) + b) \geq 1$

Quadratic Programming (QP)

- The problem we encountered is a class of problems called constrained optimization.
- There is some thing we want to optimize + a set of constraints.
- It can be solved by the Lagrangian multipler method.
- Because it is quadratic, the surface is a paraboloid, with just a single global minimum
  => thus avoiding a problem we had with neural nets!
  => NN can get stuck at a local minima

Example



flatten

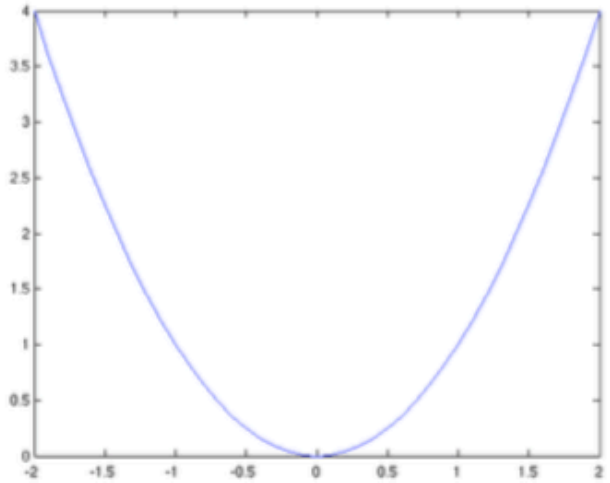
Example: paraboloid $2+x^2+2y^2$ s.t. $x+y=1$

Intuition: find intersection of two functions f, g at a tangent point (intersection = both constraints satisfied; tangent = derivative is 0); this will be a min (or max) for f s.t. the contraint g is satisfied

Constrained Optimiation Example

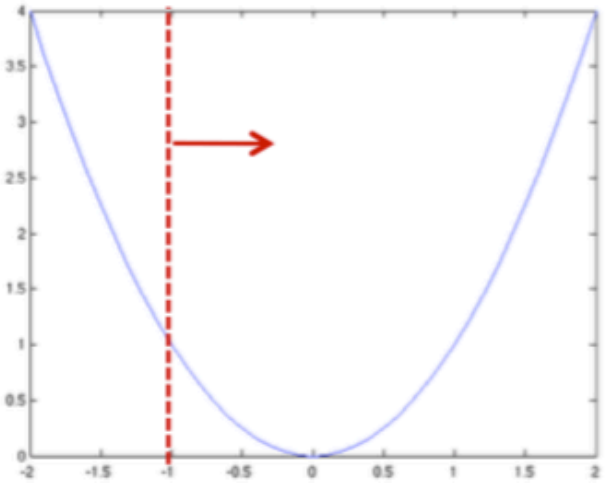
$$\min_x \; x^2$$
$$\text{s.t.} \quad x \geq b$$

$\min_x \; x^2$



$x^* = 0$

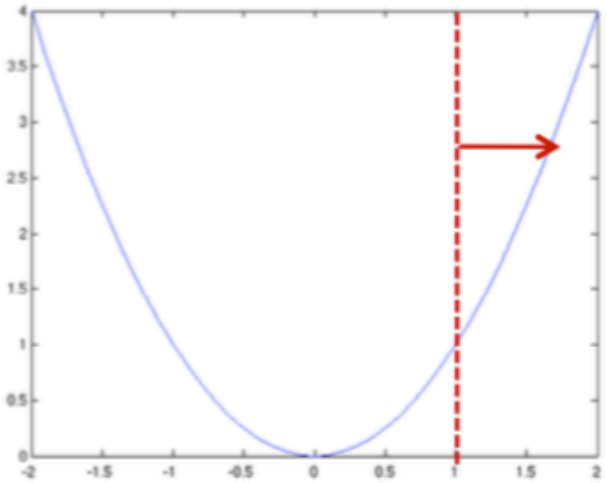
$\min_x \; x^2$
$\text{s.t.} \quad x \geq -1$



$x^* = 0$

Constraint is ineffective

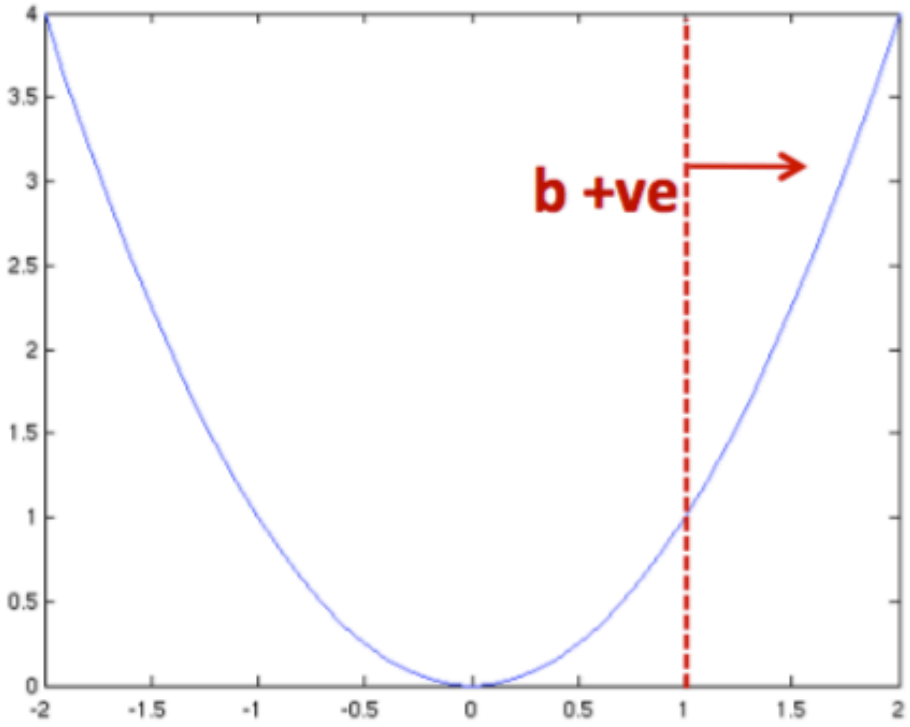
$\min_x \; x^2$
$\text{s.t.} \quad x \geq 1$



$x^* = 1$

Constraint is effective

Constrained Optimiation Example



Primal problem:

$$\min_x \; x^2$$
$$\text{s.t.} \quad x \geq b$$

Moving the constraint to objective function

Lagrangian:

$$L(x, \alpha) = x^2 - \alpha(x - b)$$
$$\text{s.t.} \quad \alpha \geq 0$$

Dual problem:

$$\max_\alpha \; d(\alpha) \rightarrow \min_x L(x, \alpha)$$
$$\text{s.t.} \quad \alpha \geq 0$$

α = 0 constraint is ineffective

α > 0 constraint is effective

Ranking classifiers

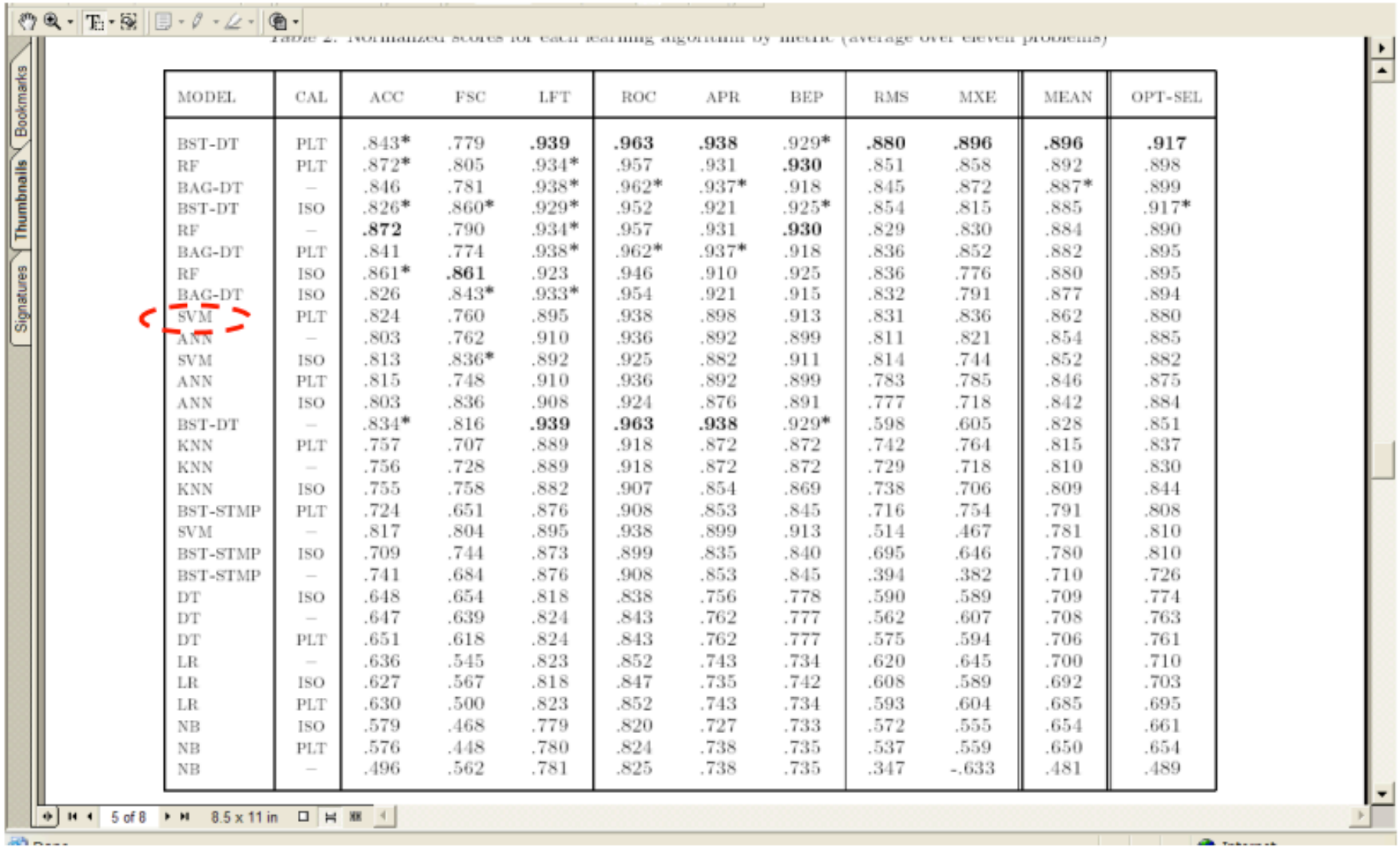


Table 2. Normalized scores for each learning algorithm by metric (average over eleven problems)

MODEL	CAL	ACC	FSC	LFT	ROC	APR	BEP	RMS	MXE	MEAN	OPT-SEL
BST-DT	PLT	.843*	.779	**.939**	**.963**	**.938**	.929*	**.880**	**.896**	**.896**	**.917**
RF	PLT	.872*	.805	.934*	.957	.931	**.930**	.851	.858	.892	.898
BAG-DT	–	.846	.781	.938*	.962*	.937*	.918	.845	.872	.887*	.899
BST-DT	ISO	.826*	.860*	.929*	.952	.921	.925*	.854	.815	.885	.917*
RF	–	**.872**	.790	.934*	.957	.931	**.930**	.829	.830	.884	.890
BAG-DT	PLT	.841	.774	.938*	.962*	.937*	.918	.836	.852	.882	.895
RF	ISO	.861*	**.861**	.923	.946	.910	.925	.836	.776	.880	.895
BAG-DT	ISO	.826	.843*	.933*	.954	.921	.915	.832	.791	.877	.894
SVM	PLT	.824	.760	.895	.938	.898	.913	.831	.836	.862	.880
ANN	–	.803	.762	.910	.936	.892	.899	.811	.821	.854	.885
SVM	ISO	.813	.836*	.892	.925	.882	.911	.814	.744	.852	.882
ANN	PLT	.815	.748	.910	.936	.892	.899	.783	.785	.846	.875
ANN	ISO	.803	.836	.908	.924	.876	.891	.777	.718	.842	.884
BST-DT	–	.834*	.816	**.939**	**.963**	**.938**	.929*	.598	.605	.828	.851
KNN	PLT	.757	.707	.889	.918	.872	.872	.742	.764	.815	.837
KNN	–	.756	.728	.889	.918	.872	.872	.729	.718	.810	.830
KNN	ISO	.755	.758	.882	.907	.854	.869	.738	.706	.809	.844
BST-STMP	PLT	.724	.651	.876	.908	.853	.845	.716	.754	.791	.808
SVM	–	.817	.804	.895	.938	.899	.913	.514	.467	.781	.810
BST-STMP	ISO	.709	.744	.873	.899	.835	.840	.695	.646	.780	.810
BST-STMP	–	.741	.684	.876	.908	.853	.845	.394	.382	.710	.726
DT	ISO	.648	.654	.818	.838	.756	.778	.590	.589	.709	.774
DT	–	.647	.639	.824	.843	.762	.777	.562	.607	.708	.763
DT	PLT	.651	.618	.824	.843	.762	.777	.575	.594	.706	.761
LR	–	.636	.545	.823	.852	.743	.734	.620	.645	.700	.710
LR	ISO	.627	.567	.818	.847	.735	.742	.608	.589	.692	.703
LR	PLT	.630	.500	.823	.852	.743	.734	.593	.604	.685	.695
NB	ISO	.579	.468	.779	.820	.727	.733	.572	.555	.654	.661
NB	PLT	.576	.448	.780	.824	.738	.735	.537	.559	.650	.654
NB	–	.496	.562	.781	.825	.738	.735	.347	-.633	.481	.489

Rich Caruana & Alexandru Niculescu-Mizil, An Empirical Comparison of Supervised Learning Algorithms, ICML 2006

Solution:

- This problem is quadratic in nature with linear constraints.
- Lagrange multiplier solution

Dual SVM – linearly separable case

- Primal problem: $\text{minimize}_{\mathbf{w},b} \quad \frac{1}{2}\mathbf{w}.\mathbf{w}$

$$\left(\mathbf{w}.\mathbf{x}_j + b\right) y_j \geq 1, \ \forall j$$

w – weights on features

- Lagrangian:

$$L(\mathbf{w}, b, \alpha) = \frac{1}{2}\mathbf{w}.\mathbf{w} - \sum_j \alpha_j \left[\left(\mathbf{w}.\mathbf{x}_j + b\right) y_j - 1\right]$$

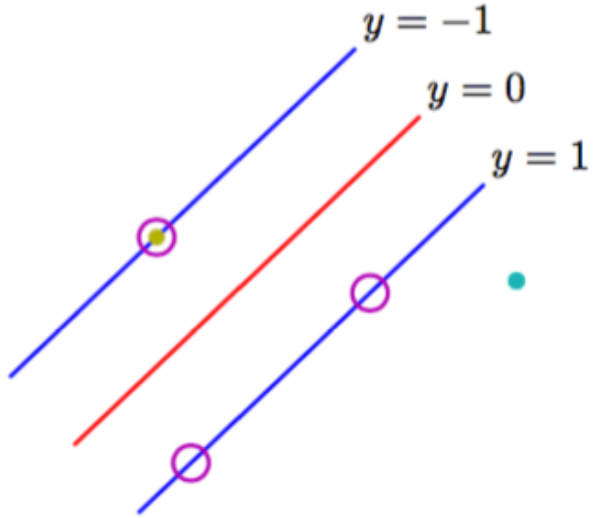
$$\alpha_j \geq 0, \ \forall j$$

α – weights on training pts

Solution: Lagrange multipliers

- Lagrange multiplier solution to this problem can be written as:

$$L(\mathbf{w}, b, \mathbf{a}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{n=1}^{N} a_n \left\{ t_n(\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n) + b) - 1 \right\}$$



$a = (a_1, a_2, \ldots, a_N)^T$ is the Lagrange multiplier vector

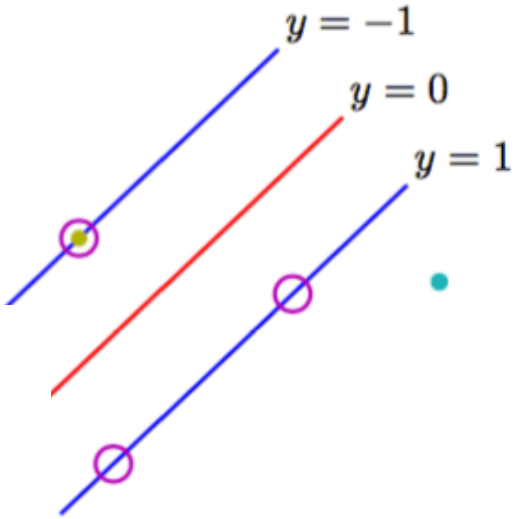
Now, find partial derivative wrt to b and w, set to 0 and solve:

$$\mathbf{w} = \sum_{n=1}^{N} a_n t_n \boldsymbol{\phi}(\mathbf{x}_n)$$

$$0 = \sum_{n=1}^{N} a_n t_n.$$

Solution: Lagrange multipliers

- Plugging these values back into original equation:



$$\widetilde{L}(\mathbf{a}) = \sum_{n=1}^{N} a_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} a_n a_m t_n t_m k(\mathbf{x}_n, \mathbf{x}_m)$$

with respect to a subject to the constraints

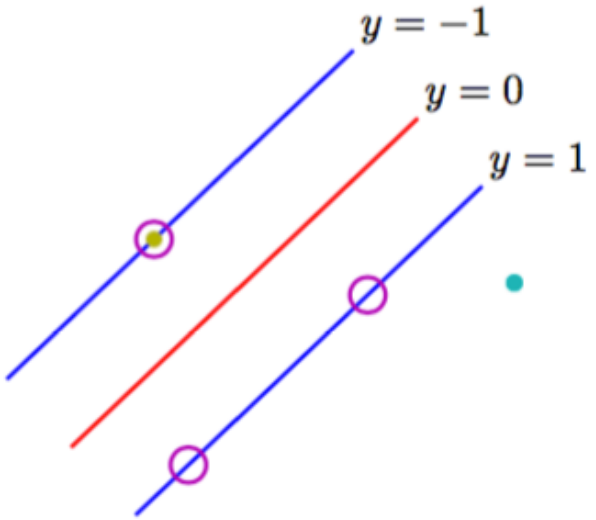
$$a_n \geqslant 0, \qquad n = 1, \ldots, N,$$

$$\sum_{n=1}^{N} a_n t_n = 0.$$

k is the kernel function. $k(x, x') = \emptyset(x)^T \emptyset(x')$

If ϕ(x) = x, then k(x, x') is simply the dot product of vectors in original space.

Solution: Lagrange multipliers



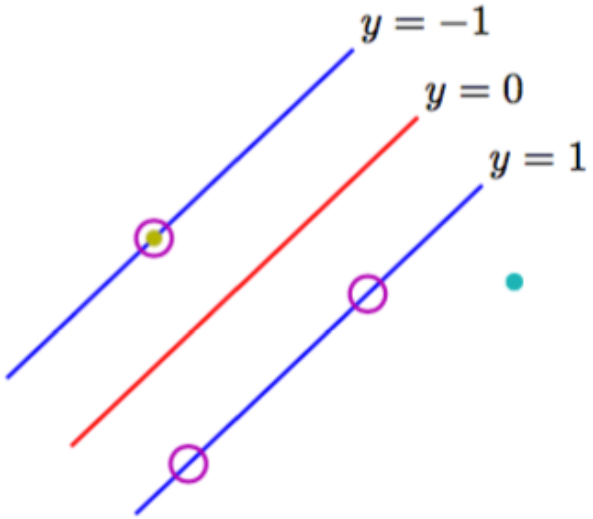
- OK… But how do we use this complicated equation.
- Remember, how do we classify:
  Find y(x) and look at its sign.
- So what is y(x). Substitute value of w into original equation:

$$y(\mathbf{x}) = \sum_{n=1}^{N} a_n t_n k(\mathbf{x}, \mathbf{x}_n) + b.$$

- a_n satisfy some KKT conditions:

$$\begin{aligned} a_n &\geqslant 0 \\ t_n y(\mathbf{x}_n) - 1 &\geqslant 0 \\ a_n \{t_n y(\mathbf{x}_n) - 1\} &= 0. \end{aligned}$$

Solution: Lagrange multipliers



- OK… But how do we use this complicated equation.
- Remember, how do we classify:
  Find y(x) and look at its sign.
- So what is y(x). Substitute value of w into original equation:

$$y(\mathbf{x}) = \sum_{n=1}^{N} a_n t_n k(\mathbf{x}, \mathbf{x}_n) + b.$$

- a_n satisfy so

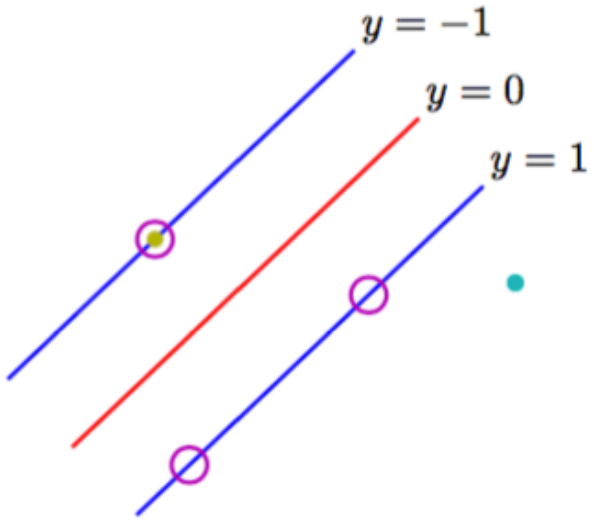
$$\begin{aligned} a_n &\geqslant 0 \\ t_n y(\mathbf{x}_n) - 1 &\geqslant 0 \\ a_n \{t_n y(\mathbf{x}_n) - 1\} &= 0. \end{aligned}$$

Makes no contribution here

=>Either $a_n = 0$ OR $t_n y(x_n) = 1$

This is true only for support vectors

Solution: Lagrange multipliers



- This is really cool!!
- y is a function of all training terms, but the Lagrangian multipliers are 0 for all terms, except support vectors.
- So the problem becomes easier => only worry about support vectors for classification of test cases.
- Also, it's a function of Kernel i.e. k which is a simple dot product in the transformed dimension.

Some famous kernels

Polynomial (typical component of ϕ might be $17q_1^2q_2^3q_4$)

$$K(\mathbf{q}, \mathbf{q}') = (1 + \mathbf{q} \cdot \mathbf{q}')^k$$

Sigmoid (typical component $\tanh(q_1 + 3q_2)$)

$$K(\mathbf{q}, \mathbf{q}') = \tanh(a\mathbf{q} \cdot \mathbf{q}' + b)$$

Gaussian RBF (typical component $\exp(-\frac{1}{2}(q_1 - 5)^2)$)

$$K(\mathbf{q}, \mathbf{q}') = \exp(-\|\mathbf{q} - \mathbf{q}'\|^2/\sigma^2)$$

Kernel Trick

- Can we use this with non-linear data?
- The famous Kernel trick $k(x, x') = \emptyset(x)^T \emptyset(x')$
- What does it accomplish?
- Changes the space from x to ϕ(x) -> which can be non-linear

  Sometimes ϕ(x) is written as z space.
- Note: We only need k(x, x') i.e. the dot product in the space and not transformation of entire values.

Generalized inner product

Given two points $\mathbf{x}$ and $\mathbf{x}' \in \mathcal{X}$, we need $\mathbf{z}^\intercal \mathbf{z}'$

Let $\mathbf{z}^\intercal \mathbf{z}' = K(\mathbf{x}, \mathbf{x}')$ (the kernel) "inner product" of $\mathbf{x}$ and $\mathbf{x}'$

Example: $\mathbf{x} = (x_1, x_2) \longrightarrow$ 2nd-order Φ

$$\mathbf{z} = \Phi(\mathbf{x}) = (1, x_1, x_2, x_1^2, x_2^2, x_1 x_2)$$

$$K(\mathbf{x}, \mathbf{x}') = \mathbf{z}^\intercal \mathbf{z}' = 1 + x_1 x'_1 + x_2 x'_2 +$$

$$x_1^2 x'^2_1 + x_2^2 x'^2_2 + x_1 x'_1 x_2 x'_2$$

The trick

Can we compute $K(\mathbf{x}, \mathbf{x}')$ without transforming $\mathbf{x}$ and $\mathbf{x}'$?

Example: Consider $K(\mathbf{x}, \mathbf{x}') = (1 + \mathbf{x}^\intercal \mathbf{x}')^2 = (1 + x_1 x'_1 + x_2 x'_2)^2$

$$= 1 + x_1^2 x'^2_1 + x_2^2 x'^2_2 + 2x_1 x'_1 + 2x_2 x'_2 + 2x_1 x'_1 x_2 x'_2$$

So, we just need to compute this value for only support vectors. No need to convert all points to higher space

This is an inner product!

$$(1 , x_1^2 , x_2^2 , \sqrt{2}\, x_1 , \sqrt{2}\, x_2 , \sqrt{2}\, x_1 x_2)$$

$$(1 , x'^2_1 , x'^2_2 , \sqrt{2} x'_1 , \sqrt{2} x'_2 , \sqrt{2} x'_1 x'_2)$$

The polynomial kernel

$\mathcal{X} = \mathbb{R}^d$ and $\Phi : \mathcal{X} \to \mathcal{Z}$ is polynomial of order Q

The "equivalent" kernel $K(\mathbf{x}, \mathbf{x}') = (1 + \mathbf{x}^\intercal \mathbf{x}')^Q$

$$= (1 + x_1 x'_1 + x_2 x'_2 + \cdots + x_d x'_d)^Q$$

Compare for $d = 10$ and $Q = 100$

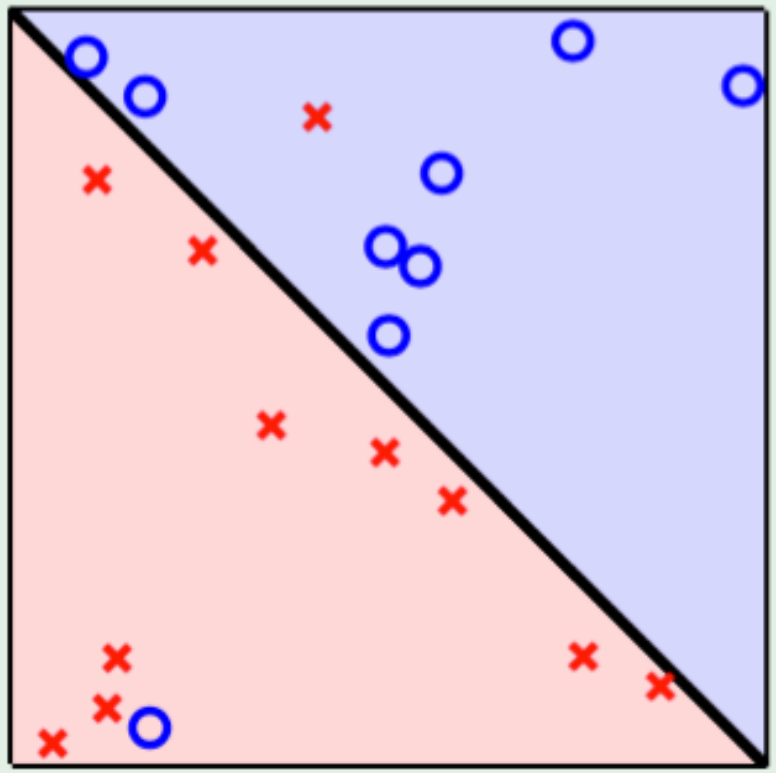
Can adjust scale: $K(\mathbf{x}, \mathbf{x}') = (a\mathbf{x}^\intercal \mathbf{x}' + b)^Q$

Is Perfection Needed?

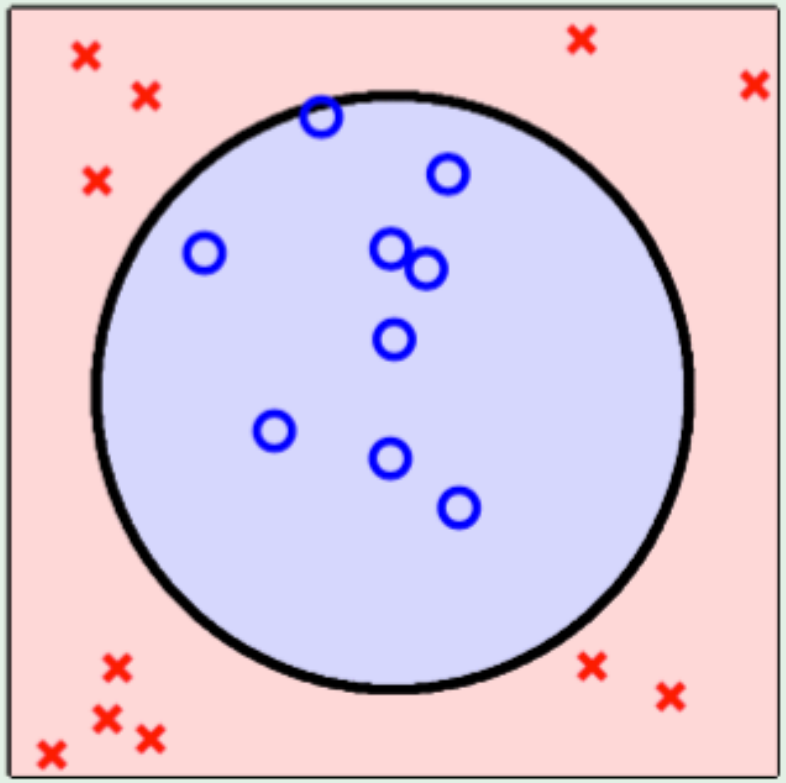
- Do we always strive for 100% accuracy?

Two types of non-separable

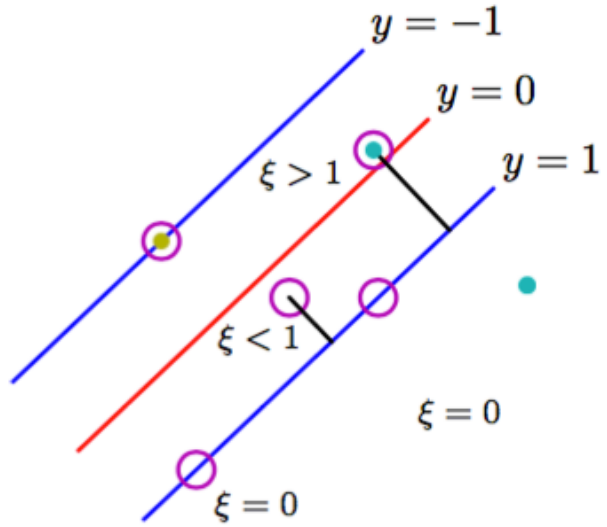
slightly:



seriously:



Slack Variables



- We define slack variables, one for each data point.

$$\xi_n \geqslant 0$$

- It refers to the distance it needs to be moved to get it on the correct side of the decision surface.
- For points inside correct boundary,

$$\xi_n = 0$$

otherwise:

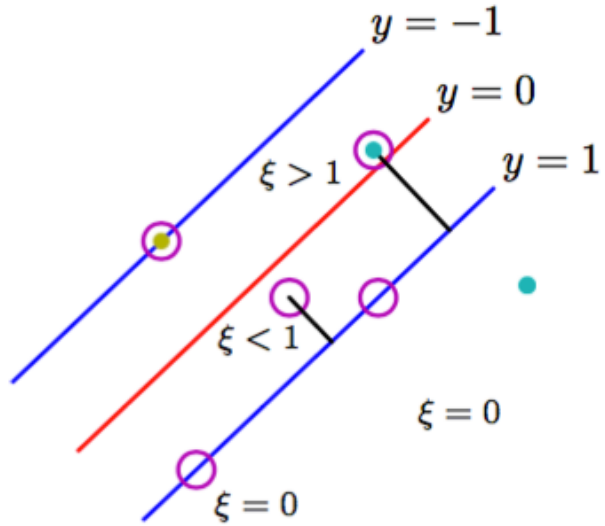
$$\xi_n = |t_n - y(\mathbf{x}_n)|$$

- If a point is on the decision boundary, we need to move it 1 unit to get it to the correct side, so $\xi_n = 1$

Slack Variables

- Our earlier exact definition changes a bit now:

$$t_n y(\mathbf{x}_n) \geqslant 1 - \xi_n, \qquad n = 1, \ldots, N$$

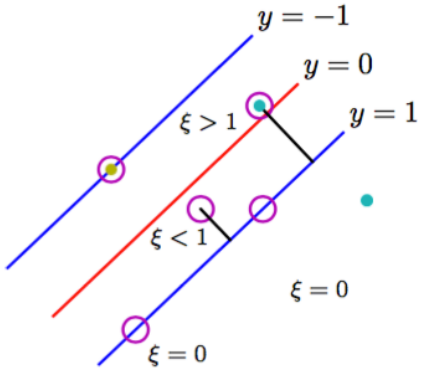


- New optimization condition:
  Minimize:

$$C \sum_{n=1}^{N} \xi_n + \frac{1}{2} \|\mathbf{w}\|^2$$

What does C represent? Trade-off between slack penalty and margin condition

Slack Variables



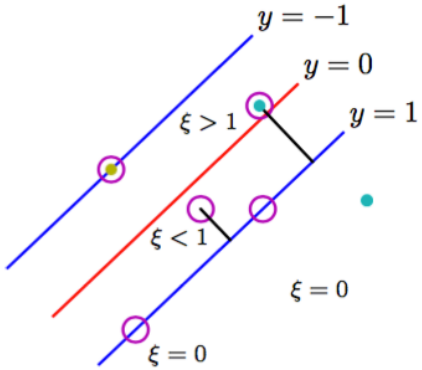
- New Lagrangian:

$$L(\mathbf{w}, b, \mathbf{a}) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{n=1}^{N}\xi_n - \sum_{n=1}^{N} a_n \{t_n y(\mathbf{x}_n) - 1 + \xi_n\} - \sum_{n=1}^{N}\mu_n\xi_n$$

subject to:

$$\begin{aligned} a_n &\geqslant 0 \\ t_n y(\mathbf{x}_n) - 1 + \xi_n &\geqslant 0 \\ a_n\left(t_n y(\mathbf{x}_n) - 1 + \xi_n\right) &= 0 \\ \mu_n &\geqslant 0 \\ \xi_n &\geqslant 0 \\ \mu_n\xi_n &= 0 \end{aligned}$$

Slack Variables



- Find values:

$$\frac{\partial L}{\partial \mathbf{w}} = 0 \quad \Rightarrow \quad \mathbf{w} = \sum_{n=1}^{N} a_n t_n \boldsymbol{\phi}(\mathbf{x}_n)$$

$$\frac{\partial L}{\partial b} = 0 \quad \Rightarrow \quad \sum_{n=1}^{N} a_n t_n = 0$$

$$\frac{\partial L}{\partial \xi_n} = 0 \quad \Rightarrow \quad a_n = C - \mu_n.$$

to get

$$\widetilde{L}(\mathbf{a}) = \sum_{n=1}^{N} a_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} a_n a_m t_n t_m k(\mathbf{x}_n, \mathbf{x}_m)$$

subject to

$$0 \leqslant a_n \leqslant C$$

$$\sum_{n=1}^{N} a_n t_n = 0$$

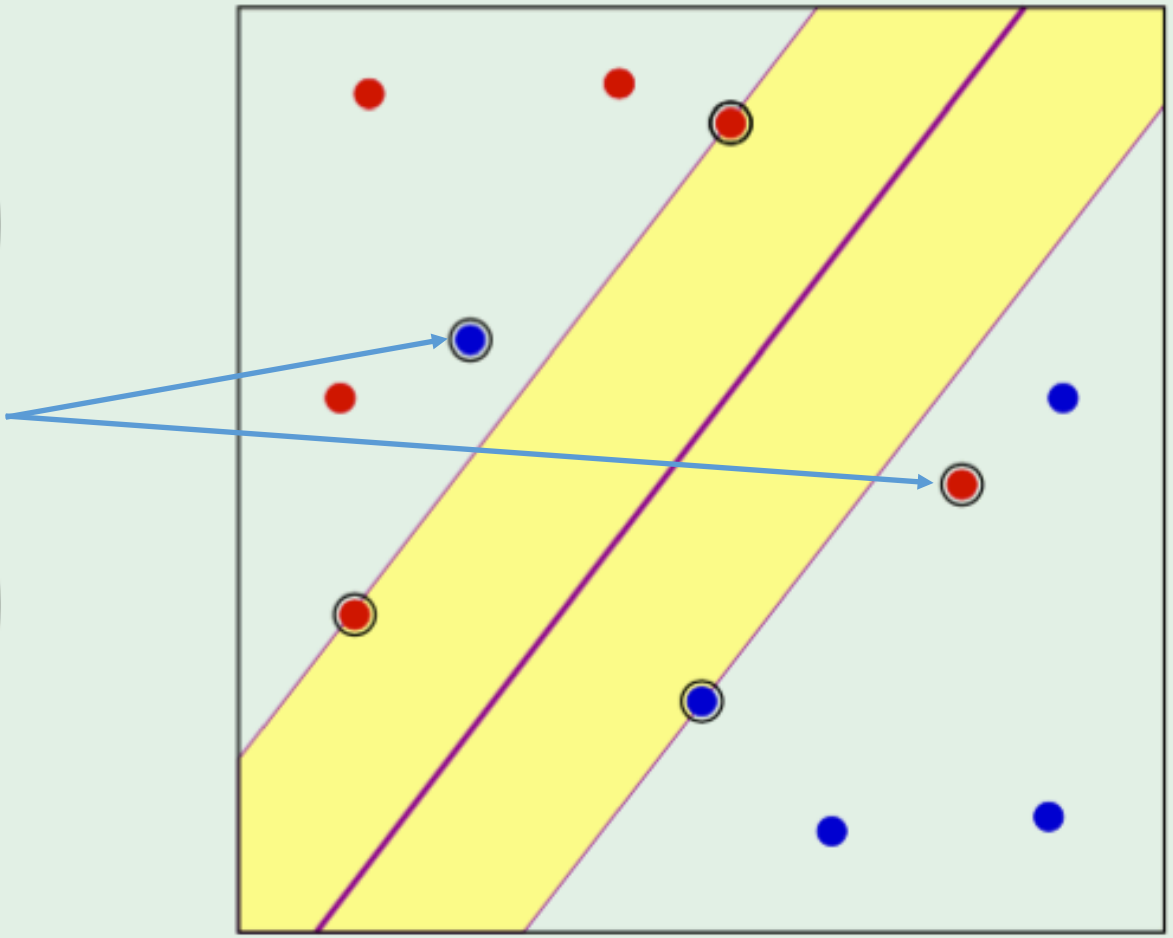
Types of support vectors

margin support vectors $(0 < \alpha_n < C)$

$t_n \, (\mathbf{w}^\intercal \mathbf{x}_n + b) = 1 \qquad (\xi_n = 0)$

non-margin support vectors $(\alpha_n = C)$

$t_n \, (\mathbf{w}^\intercal \mathbf{x}_n + b) < 1 \qquad (\xi_n > 0)$



Soft Margin SVM

- Hard margin SVM => you insist data should be linearly separable
- Soft margin SVM => you introduce slack variables (ξ) so that a few points are not perfectly separable.

- In which case do you obtain 100% training accuracy?

Points to think about

- Advantages of SVM:
  - Computation is simplified, as you only worry about support vectors
  - Can transform points to a different space $x \rightarrow \phi(x)$. So we can classify non-linearly separable data.
- Disadvantages of SVM:
  - What would happen if support vectors are noise points (after all, everything depends on these points)
  - How do you find best transformation or kernel
  - Best value of C
  - Much parameter tuning needed.