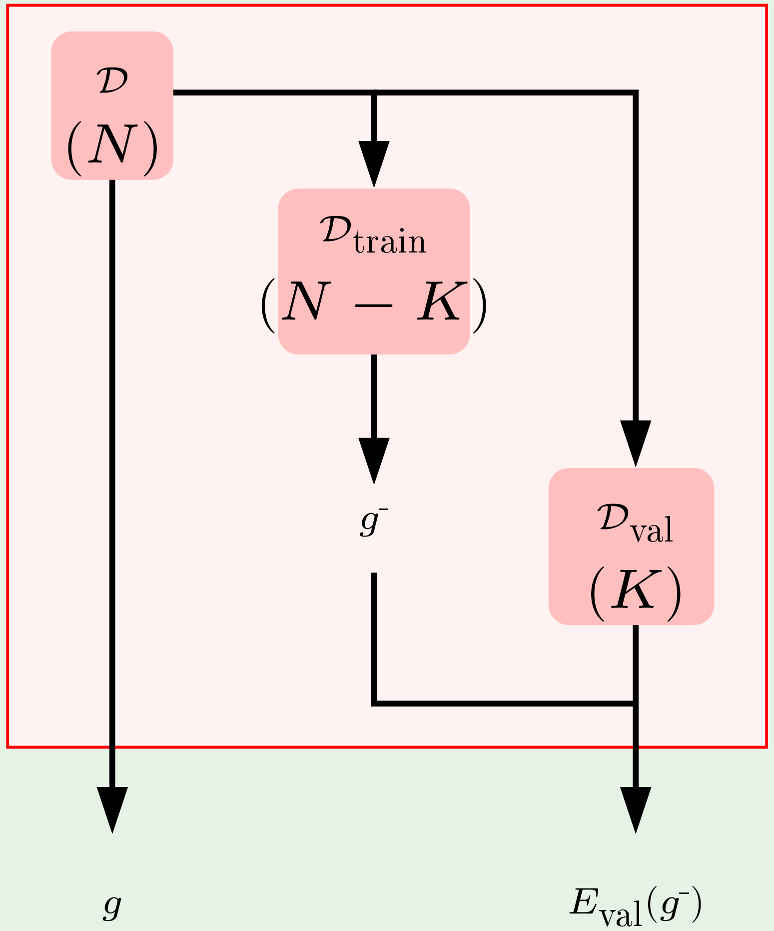
Review of Lecture 13

- Validation

helps us estimate out-of-sample performance

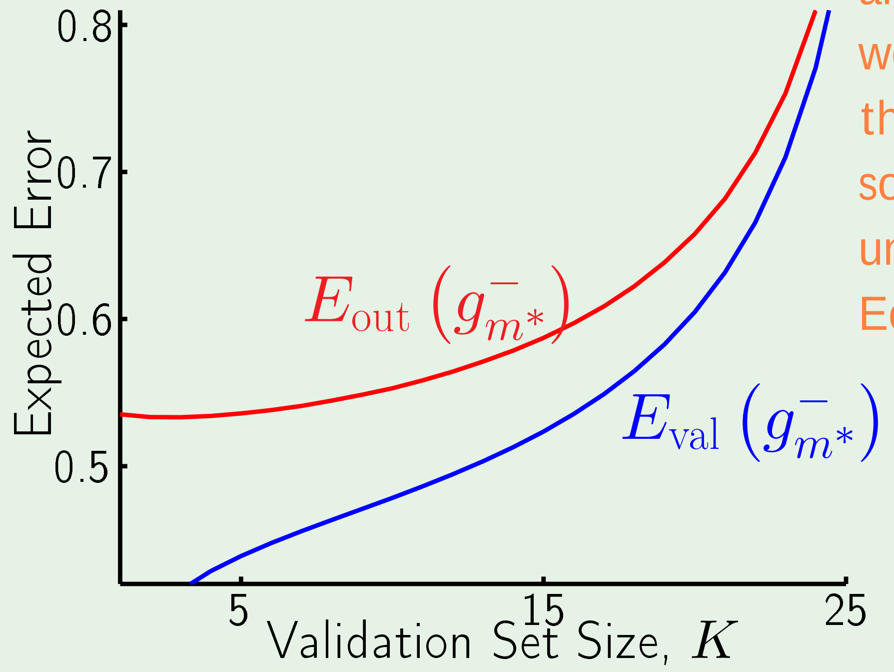


$E_{\text{val}}(g^-)$ estimates $E_{\text{out}}(g)$

unbiased estimate, so Eval can be better or worse than Eout

- Data contamination

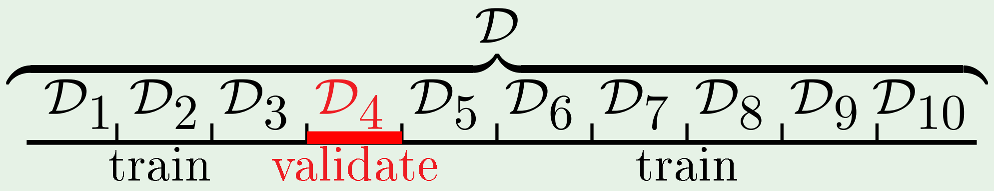
however, once we use validation for model selection, we introduce an optimistic bias since we choose the model that has the best Eval, so Eval no longer unbiased estimate of Eout



$\mathcal{D}_{\text{val}}$ slightly contaminated

- Cross validation

(used in practical examples)



10-fold cross validation

Learning From Data

Yaser S. Abu-Mostafa
California Institute of Technology

Lecture 14: Support Vector Machines

(arguably the most successful classification method in machine learning)

Sponsored by Caltech's Provost Office, E&AS Division, and IST • Thursday, May 17, 2012

Outline

- Maximizing the margin

- The solution

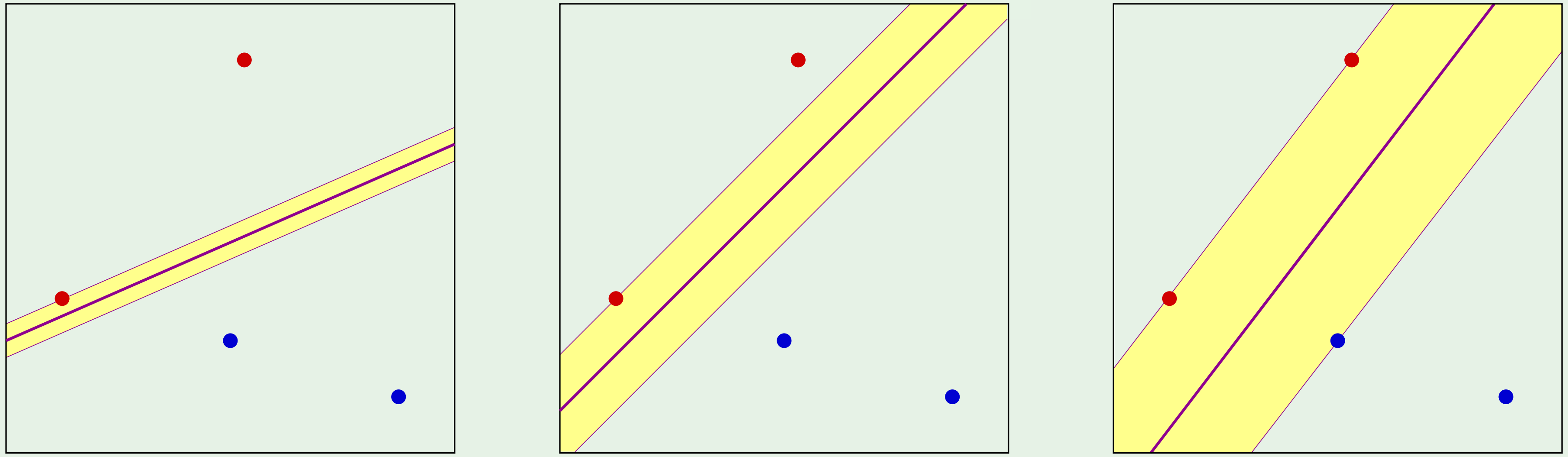
- Nonlinear transforms

Better linear separation

Linearly separable data

Different separating lines

Which is best?



The yellow region is the margin are depicts how much the line can move before it 'crosses over' and makes an error - the margin of error.

Intuitively, the bigger margin is better: consider a process which is generating the data, say it has some noise - a bigger margin means that the chances are a new point will still be on the correct side of the line.

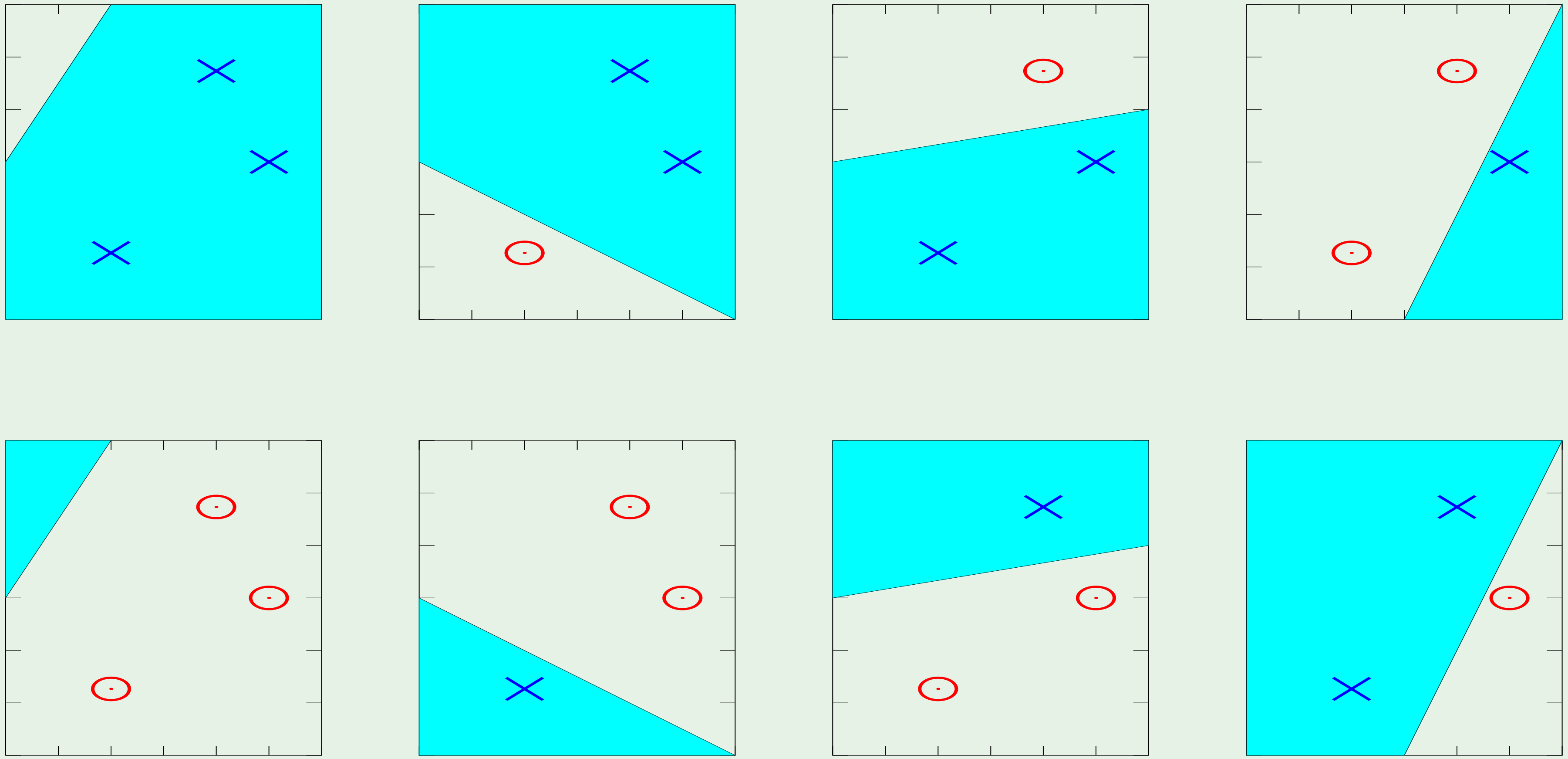
Two questions:

1. Why is bigger margin better?

2. Which $\mathbf{w}$ maximizes the margin?

Remember the growth function?

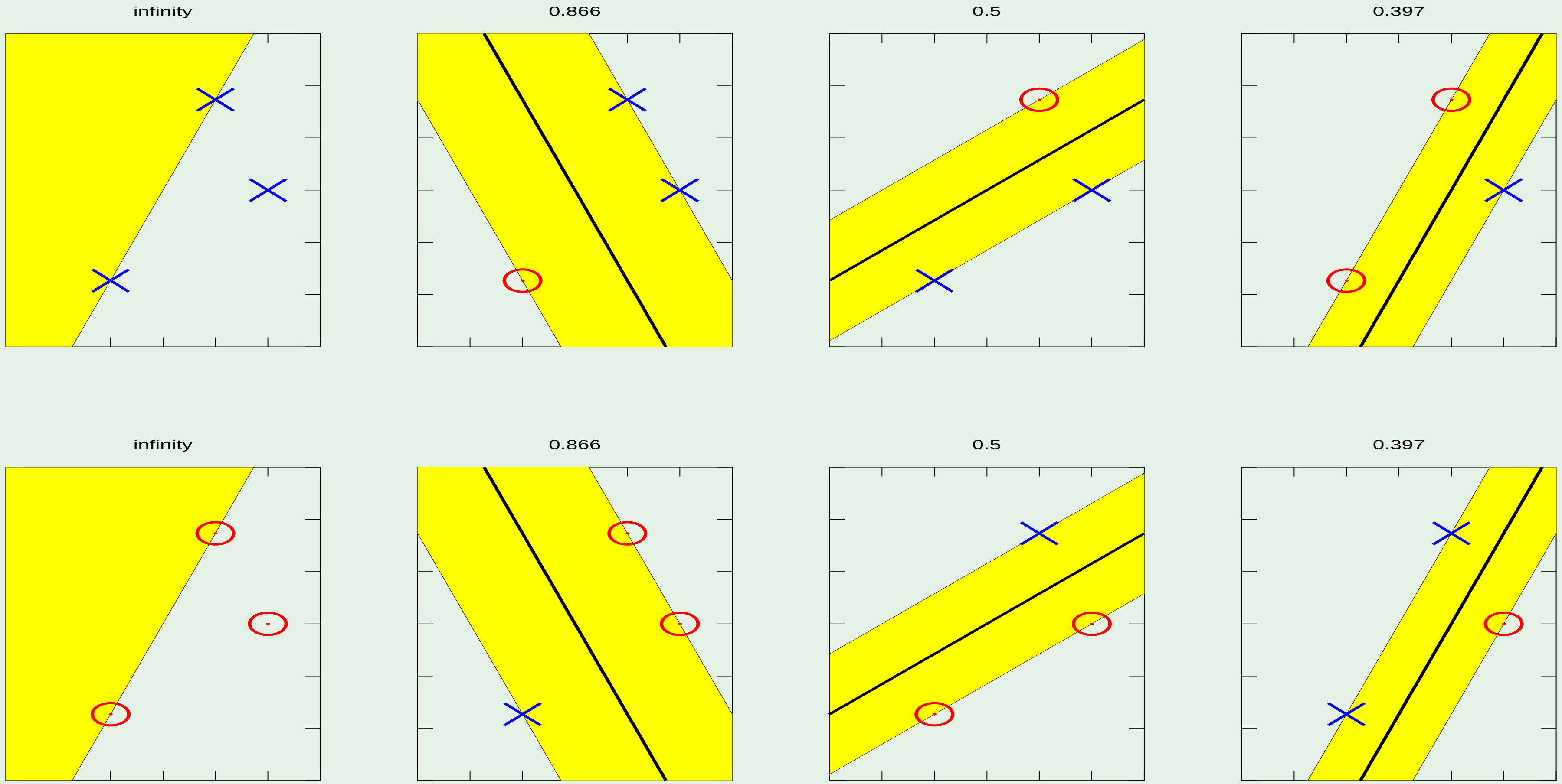
All dichotomies with any line:



Here there are all 2^3 dichotomies, so the growth function is big (and no break point) which is bad for generalization

Dichotomies with fat margin

Fat margins imply fewer dichotomies



If we are required to have atleast this ^^^ size margin for the classifier to be accepted, all the dichotomies on the right are no longer allowed. So we can restrict the growth function by requiring atleast a certain size of margin, hence fat margins imply fewer possible dichotomies. Therefore, if we manage to separate the points with a fat dichotomy, we can say that fat dichotomies have a smaller VC dimension / smaller growth function than if we did not restrict them at all. So bigger margin leads to better Eout.

Finding $\mathbf{w}$ with large margin

A margin is just the distance from a (hyper)plane to a point

wTx is the signal for a given x

Let $\mathbf{x}_n$ be the nearest data point to the plane $\mathbf{w}^{\mathsf{T}}\mathbf{x} = 0$. How far is it?

Technical preparations with no loss of generality which makes the solution friendly later on:

2 preliminary technicalities:

1. **Normalize $\mathbf{w}$:**

$$\left|\mathbf{w}^{\mathsf{T}}\mathbf{x}_n\right| = 1$$

or |wTx_n + b| = 1

built in scale invariance of w - just means that w (the vector) is in its simplest form scaled relative to the training examples

2. **Pull out w_0:**

(d = dimensionality of the data)

$$\mathbf{w} = (w_1, \cdots, w_d) \quad \text{apart from} \quad b$$

The plane is now $\boxed{\mathbf{w}^{\mathsf{T}}\mathbf{x} + b = 0}$ (no x_0)

Computing the distance

The distance between $\mathbf{x}_n$ and the plane $\mathbf{w}^{\intercal}\mathbf{x} + b = 0$ where $|\mathbf{w}^{\intercal}\mathbf{x}_n + b| = 1$

The vector $\mathbf{w}$ is $\perp$ to the plane in the $\mathcal{X}$ space:

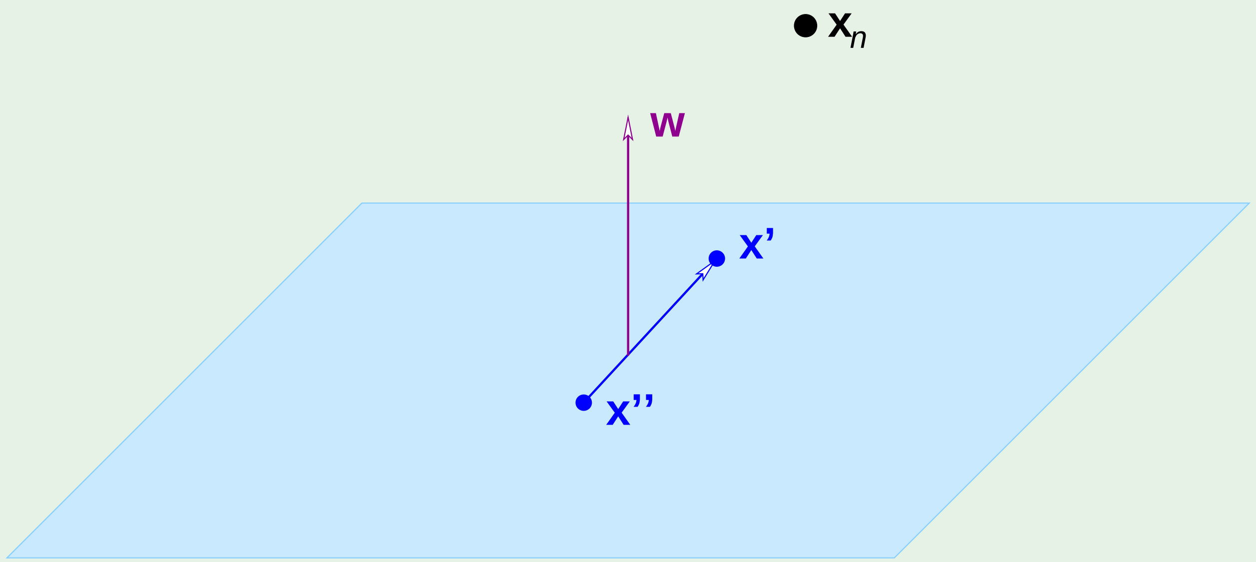
because:

Take $\mathbf{x}'$ and $\mathbf{x}''$ on the plane

$$\mathbf{w}^{\intercal}\mathbf{x}' + b = 0 \quad \text{and} \quad \mathbf{w}^{\intercal}\mathbf{x}'' + b = 0$$

$$\Longrightarrow \quad \mathbf{w}^{\intercal}(\mathbf{x}' - \mathbf{x}'') = 0$$

so wT must be orthogonal to x'-x'', any general vector in the place, hence points out of the plane

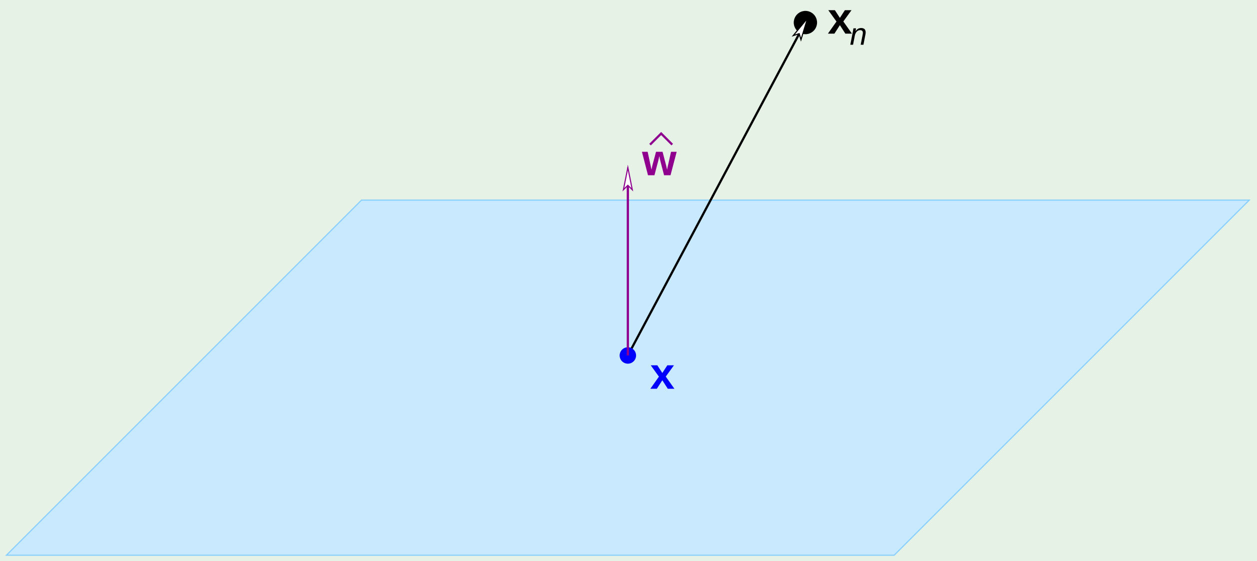


and the distance is ...

Distance between $\mathbf{x}_n$ and the plane:

Take any point $\mathbf{x}$ on the plane

Projection of $\mathbf{x}_n - \mathbf{x}$ on $\mathbf{w}$



$$\hat{\mathbf{w}} = \frac{\mathbf{w}}{\|\mathbf{w}\|} \implies \text{distance} = \big|\overbrace{\hat{\mathbf{w}}^{\intercal}(\mathbf{x}_n - \mathbf{x})}^{\text{size of inner product}}\big|$$

$$\text{distance} = \frac{1}{\|\mathbf{w}\|}\left|\mathbf{w}^{\intercal}\mathbf{x}_n - \mathbf{w}^{\intercal}\mathbf{x}\right| = \frac{1}{\|\mathbf{w}\|}\Big|\overbrace{\mathbf{w}^{\intercal}\mathbf{x}_n + b}^{=\,1\text{ from normalization}} \overbrace{- \mathbf{w}^{\intercal}\mathbf{x} - b}^{=\,0\text{ by definition of the plane}}\Big| = \frac{1}{\|\mathbf{w}\|}$$

The optimization problem

$$\text{Maximize } \frac{1}{\|\mathbf{w}\|}$$

$$\text{subject to } \min_{n=1,2,\ldots,N} \left|\mathbf{w}^{\intercal}\mathbf{x}_n + b\right| = 1$$

(subject to the constraint that the nearest point is at distance 1)

Having a minimum in the constraint makes it a tricky optimization problem so we want a more friendly equivalent problem...

Notice: $\left|\mathbf{w}^{\intercal}\mathbf{x}_n + b\right| = y_n\left(\mathbf{w}^{\intercal}\mathbf{x}_n + b\right)$

because all points are correctly classified, the signal (wTx_n + b) must be equal to the label yn (which must be +/- 1), and 1*1=1 & -1*-1=1 which is the constrained distance of the minimum point(s)

So: $$\text{Minimize } \frac{1}{2}\mathbf{w}^{\intercal}\mathbf{w}$$

$$\text{subject to } y_n\left(\mathbf{w}^{\intercal}\mathbf{x}_n + b\right) \geq 1 \quad \text{for} \quad n = 1, 2, \ldots, N$$

The constraint is equivalent because it is not possible for the minimum to be achieved when yn(...) is > 1. This is because if we had that solution with yn(...) > 1, we can scale w and b proportionately down until one of the yn(...) = 1. Then, this scaled down w and b is smaller than the previous, so we have further minimized (1/2)wTw. Therefore, solving the above optimization gives us a w which necessarily satisfies the constraint with atleast one point at distance 1.

Outline

- Maximizing the margin
- The solution
- Nonlinear transforms

Constrained optimization

Minimize $\quad \dfrac{1}{2}\, \mathbf{w}^\intercal \mathbf{w}$

subject to $\quad y_n \left(\mathbf{w}^\intercal \mathbf{x}_n + b \right) \geq 1 \quad$ for $\quad n = 1, 2, \ldots, N$

$$\mathbf{w} \in \mathbb{R}^d, \;\; b \in \mathbb{R}$$

Lagrange? $\qquad$ inequality constraints $\Longrightarrow$ KKT

approach

We saw this before

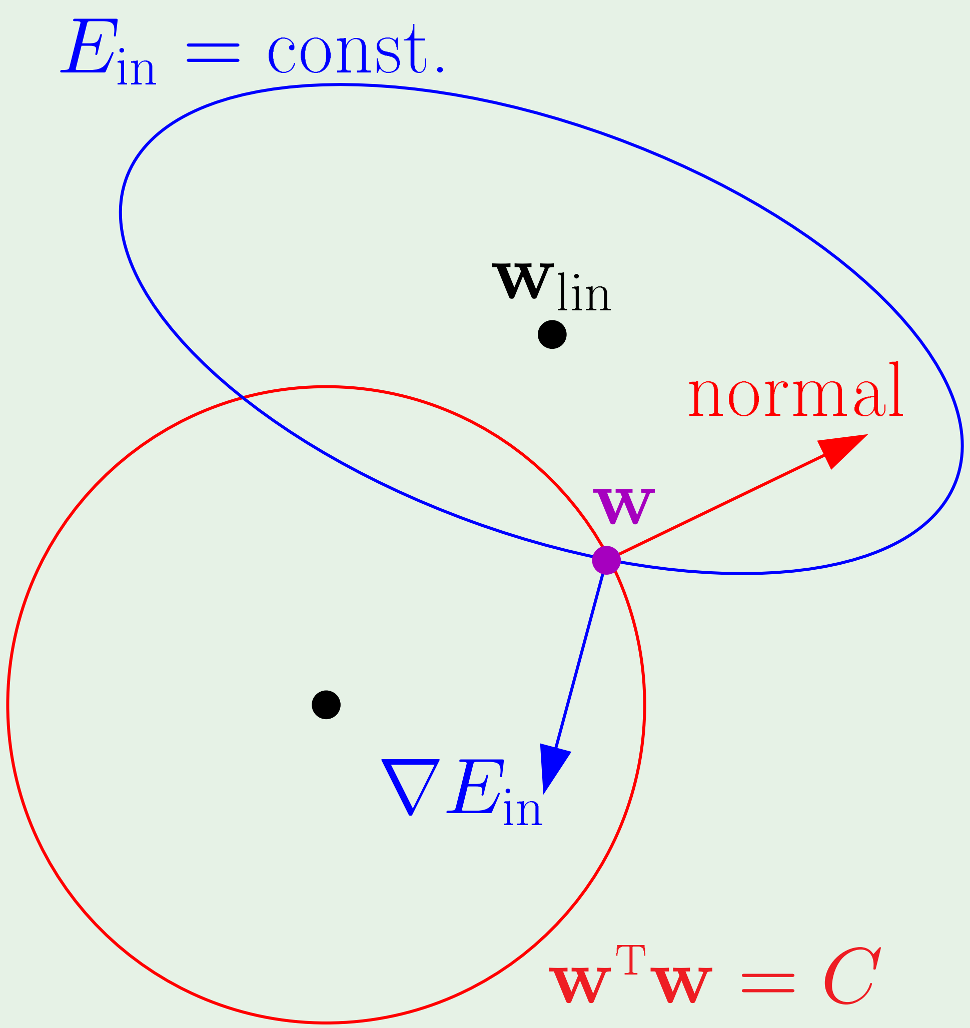
Remember regularization?

Minimize $\quad E_{\text{in}}(\mathbf{w}) \;=\; \frac{1}{N}\,(\mathrm{Z}\mathbf{w}-\mathbf{y})^{\mathsf{T}}(\mathrm{Z}\mathbf{w}-\mathbf{y})$

subject to: $\quad \mathbf{w}^{\mathsf{T}}\mathbf{w} \le C$

∇E_{in} normal to constraint

	optimize	constrain
Regularization:	E_{in}	$\mathbf{w}^{\mathsf{T}}\mathbf{w}$
SVM:	$\mathbf{w}^{\mathsf{T}}\mathbf{w}$	E_{in}



(we constrain Ein = 0 since we require all the points to be classified correctly). Note that both the object to be optimized and the constraint will both blend in the Lagrangian and we end up doing something which is a compromise - there is an equivalency between regularization and SVM: we minimize what is in our mind a constraint and constrain an objective function to be minimized.

Lagrange formulation

This term must be greater than/equal to 0 - can be considered the 'slack'. At the nearest point(s) to the line the slack = 0, and all 'interior' points have slack greater than 0. Note the summation is negated because the inequality was greater than/equal to.

$$\text{Minimize} \quad \mathcal{L}(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\mathbf{w}^{\mathsf{T}}\mathbf{w} - \sum_{n=1}^{N} \alpha_n \overbrace{\left(y_n\left(\mathbf{w}^{\mathsf{T}}\mathbf{x}_n + b\right) - 1\right)}$$

alpha_n are the Lagrange multipliers

w.r.t. $\mathbf{w}$ and b and maximize w.r.t. each $\alpha_n \geq 0$

(The maximization of alpha_n is tricky since it has a restricted range)

$$\nabla_{\mathbf{w}}\mathcal{L} = \mathbf{w} - \sum_{n=1}^{N} \alpha_n y_n \mathbf{x}_n = \mathbf{0}$$

$$\frac{\partial \mathcal{L}}{\partial b} = - \sum_{n=1}^{N} \alpha_n y_n = 0$$

Substituting ...

We substitute to get the dual formulation of the problem, so we can maximise w.r.t. alpha_n >= 0 with no dependancy on w and b.

$$\mathbf{w} \;=\; \sum_{n=1}^{N} \alpha_n y_n \mathbf{x}_n \qquad \text{and} \qquad \sum_{n=1}^{N} \alpha_n y_n \;=\; 0$$

in the Lagrangian

$$\mathcal{L}(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\,\mathbf{w}^{\intercal}\mathbf{w} \;-\; \sum_{n=1}^{N} \alpha_n \left(y_n \left(\mathbf{w}^{\intercal}\mathbf{x}_n + b \right) - 1 \right)$$

we get

$$\mathcal{L}(\boldsymbol{\alpha}) = \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} y_n y_m \,\alpha_n \alpha_m \,\mathbf{x}_n^{\intercal}\mathbf{x}_m$$

Maximize w.r.t. to $\boldsymbol{\alpha}$ subject to $\alpha_n \geq 0$ for $n = 1, \cdots, N$ **and** $\sum_{n=1}^{N} \alpha_n y_n = 0$

(this is a KKT constraint)

The solution - quadratic programming

Minimize the negative of L(alpha) is equivalent to maximizing L(alpha)

$$\min_{\boldsymbol{\alpha}} \quad \frac{1}{2}\,\boldsymbol{\alpha}^{\intercal} \underbrace{\begin{bmatrix} y_1y_1\,\mathbf{x}_1^{\intercal}\mathbf{x}_1 & y_1y_2\,\mathbf{x}_1^{\intercal}\mathbf{x}_2 & \dots & y_1y_N\,\mathbf{x}_1^{\intercal}\mathbf{x}_N \\ y_2y_1\,\mathbf{x}_2^{\intercal}\mathbf{x}_1 & y_2y_2\,\mathbf{x}_2^{\intercal}\mathbf{x}_2 & \dots & y_2y_N\,\mathbf{x}_2^{\intercal}\mathbf{x}_N \\ \dots & \dots & \dots & \dots \\ y_Ny_1\,\mathbf{x}_N^{\intercal}\mathbf{x}_1 & y_Ny_2\,\mathbf{x}_N^{\intercal}\mathbf{x}_2 & \dots & y_Ny_N\,\mathbf{x}_N^{\intercal}\mathbf{x}_N \end{bmatrix}}_{\text{quadratic coefficients}} \boldsymbol{\alpha} + \underbrace{(-\mathbf{1}^{\intercal})}_{\text{linear}}\,\boldsymbol{\alpha}$$

subject to $\quad \underbrace{\mathbf{y}^{\intercal}\boldsymbol{\alpha} = 0}_{\text{linear constraint}}$

$$\underbrace{\mathbf{0}}_{\text{lower bounds}} \leq \boldsymbol{\alpha} \leq \underbrace{\infty}_{\text{upper bounds}}$$

This is simply the process of minimizing a quadratic function with a linear term subject to a linear equality constraint and a range constraint - this is just the expanded version in terms of the numbers in an NxN matrix. (QP will be successful if this is a convex function).

Note that QP will difficulty with N >~ 10,000s since it involves an NxN matrix; this is because the matrix is dense, the entries could be anything so all of them matter. Hence, there are lots of heuristics to solve this problem (such as hierarchical methods) if N is large. N = 1000 should be OK. There are packages specifically for SVM which use heuristics (e.g. dont explicitly pass on the whole matrix directly to QP but split it into pieces, get SV's for each piece and then get the union etc.) - these can be used when we have too many data points.

QP hands us $\boldsymbol{\alpha}$

Either alpha_n or the slack is zero - so for all interior points (which is the majority of the N points), the slack will be positive so the corresponding alphas must be zero, while alpha is a non-zero positive number when slack=0 i.e. a nearest point. We have seen this before: when C is very large such that the constraint is vacuous and the absolute optimal is inside the constraint, we have no need for regularization and lambda = 0. This is equivalent to the case where we have an interior point so the alpha (multiplier) is zero. When we have to compromise/we ahve an active constraint, lambda is positive. This is the equivalent case to when we have slack = 0 i.e. a nearest point, and positive alpha.

Solution: $\boldsymbol{\alpha} = \alpha_1, \cdots, \alpha_N$

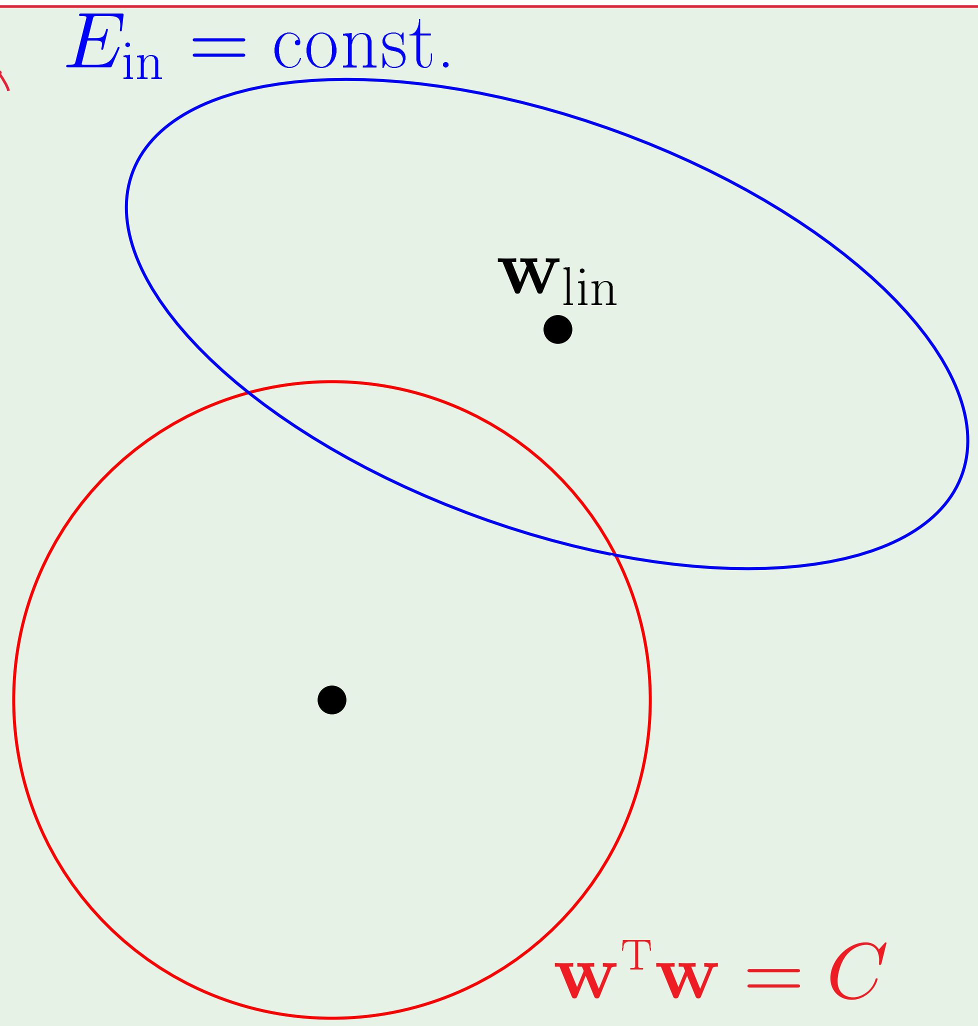
$$\Longrightarrow \quad \mathbf{w} = \sum_{n=1}^{N} \alpha_n y_n \mathbf{x}_n$$

Another KKT condition which is key to defining support vectors:

KKT condition: For $n = 1, \cdots, N$

$$\alpha_n \left(y_n \left(\mathbf{w}^\intercal \mathbf{x}_n + b \right) - 1 \right) = 0$$

We saw this before!



Definition:

$\alpha_n > 0 \Longrightarrow \mathbf{x}_n$ is a $\boxed{\textbf{support vector}}$

(i.e. a nearest point which defines the margin)

We find the maximum margin which classifies the points and it touches some of the +1 and -1 points - these points 'support' the plane and are called support vectors, all other points are interior points.

Support vectors

Closest $\mathbf{x}_n$'s to the plane: achieve the margin

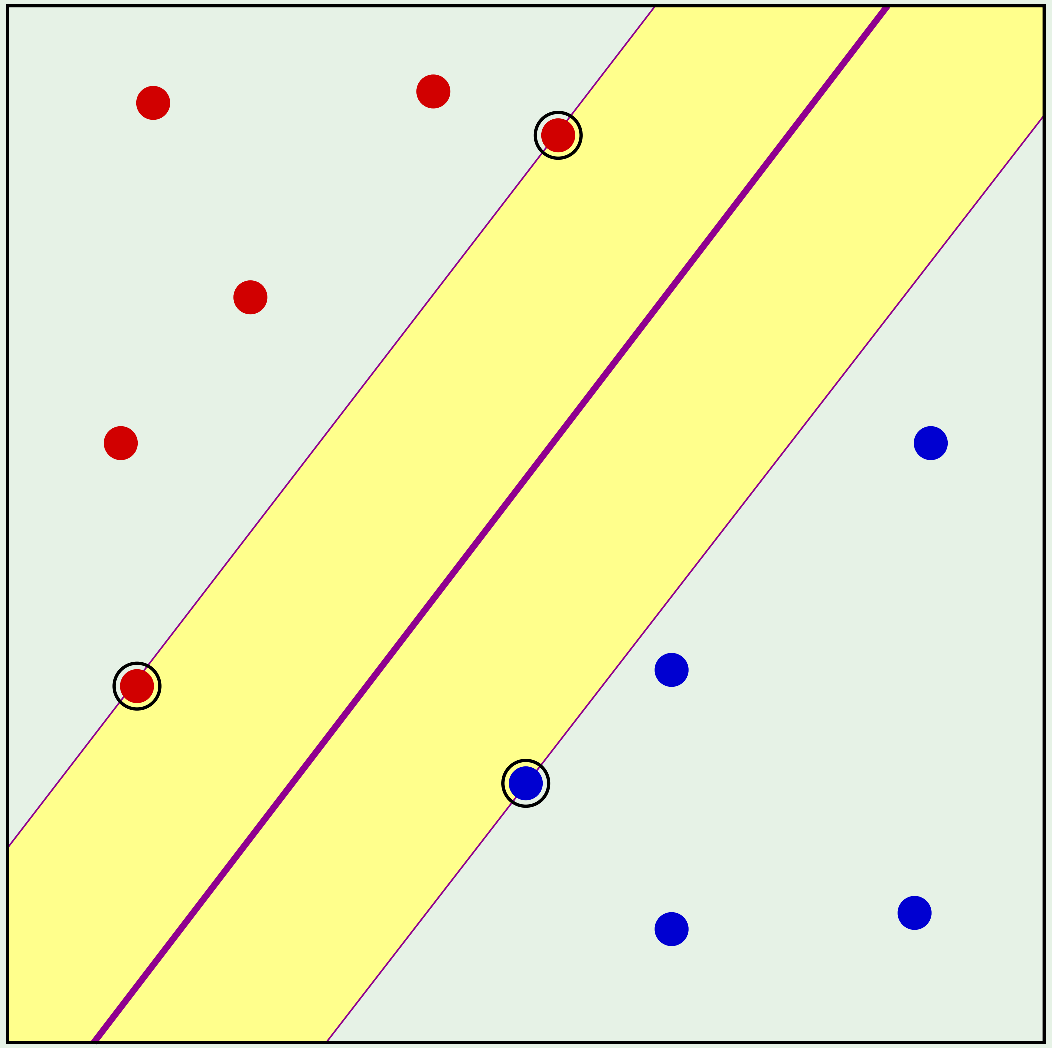
$$\Longrightarrow \quad y_n \left(\mathbf{w}^\intercal \mathbf{x}_n + b \right) = 1$$

$$\mathbf{w} = \sum_{\mathbf{x}_n \text{ is SV}} \alpha_n y_n \mathbf{x}_n$$

Solve for b using any SV:

$$y_n \left(\mathbf{w}^\intercal \mathbf{x}_n + b \right) = 1$$

So if we have only 3 support vectors, our constraint of having the largest possible margin means w is effectively 3-d instead of the length of the dimensionality of the data, hence the generalization dividend since we end up with lots fewer effective parameters than the size of w (i.e. many of the elements of w are zero due to many alpha_n=0).



Outline

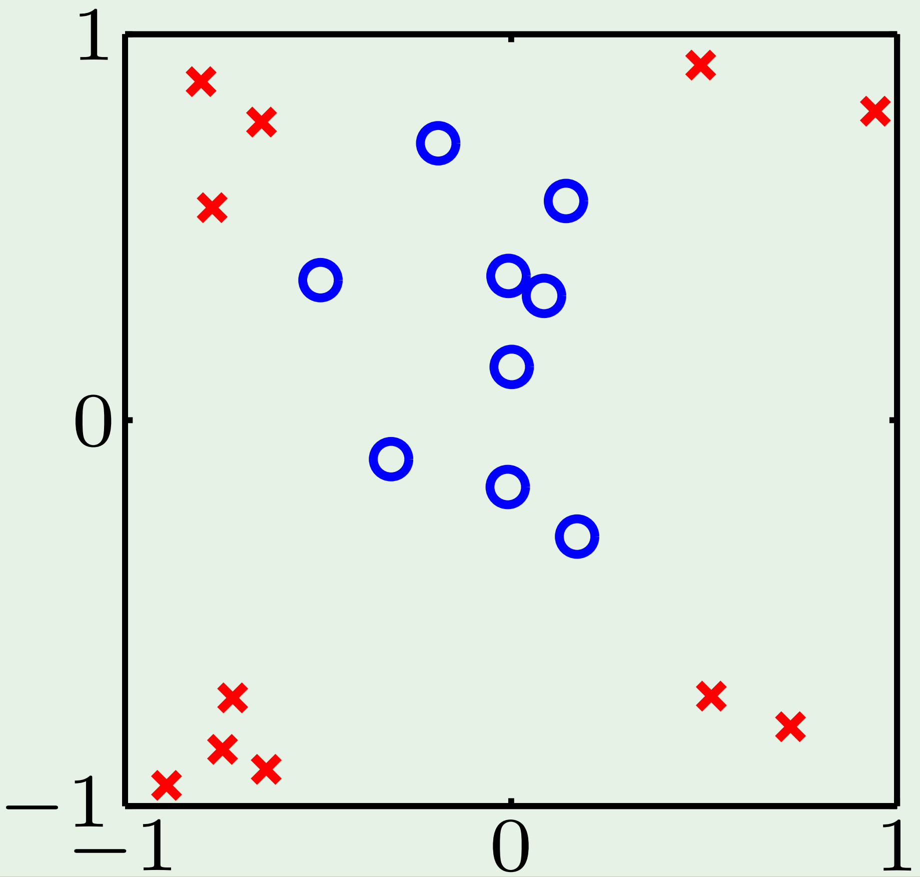
- Maximizing the margin

- The solution

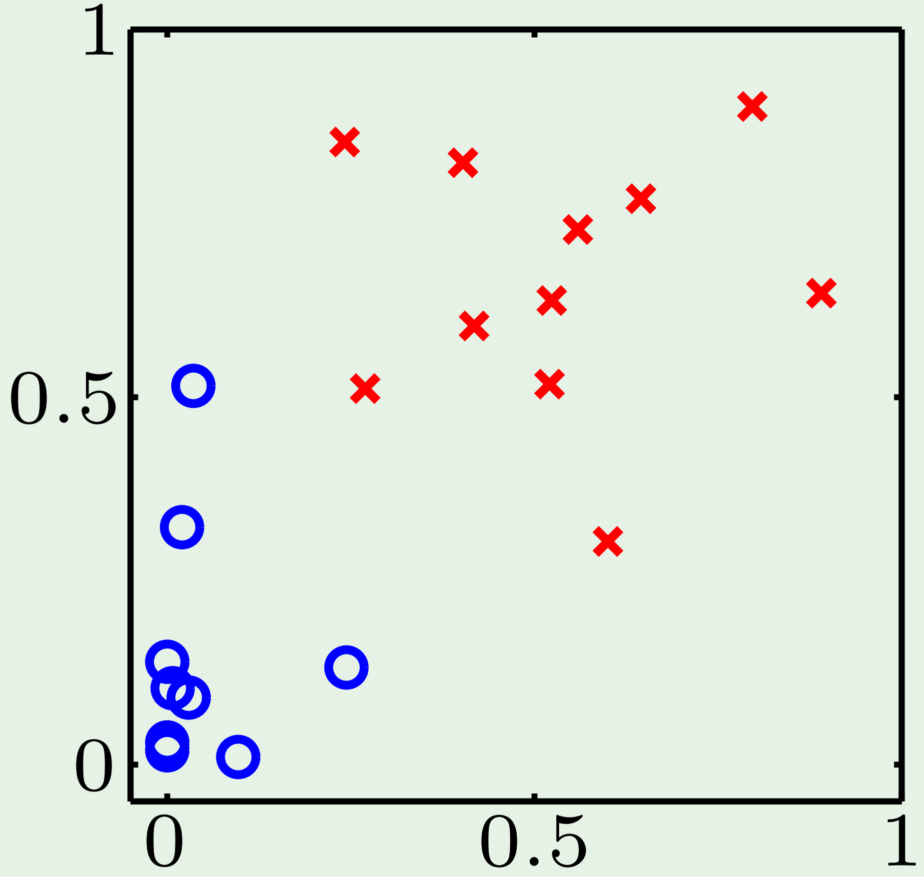
- Nonlinear transforms

z instead of x

$$\mathcal{L}(\boldsymbol{\alpha}) = \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} y_n y_m \, \alpha_n \alpha_m \, \mathbf{z}_n^{\mathsf{T}} \mathbf{z}_m$$



$$\mathcal{X} \longrightarrow \mathcal{Z}$$



We map the points from X to Z space, and instead of getting a generic separator in Z space, we get the best separator, according to SVM, and then mapping it back, hoping it has dividends in terms of generalization. Note that if we map from 2-d space to 1m-d space, the inner product zT_n*z_m still just gives us a single number even if they are longer vectors than x with dimension of 1m (the increase in computational difficulty is minor since QP is the major contributor to that). Note that the dimensionality of the problem we pass to QP, i.e. the number of alphas that we need to compute, is EXACTLY the same - just one for each data point. So we can map to an enormous space without paying the price for it in terms of the QP optimization. The alphas we get can then be interpreted in the space we created it from (Z), so the w will belong to the Z space.

N.B this is the 'hard' margin version of SVM that tries to fit every point/the margin is satistfied strictly, comparable to the PLA. There is a 'soft' version which, like the 'pocket' algorithm for perceptrons, deals with slightly non-separable data by allowing for a few errors and penalizes for them. For really non-separable data, non-linear transformations are used (usually with soft margin SVM to avoid fitting noise).

"Support vectors" in $\mathcal{X}$ space

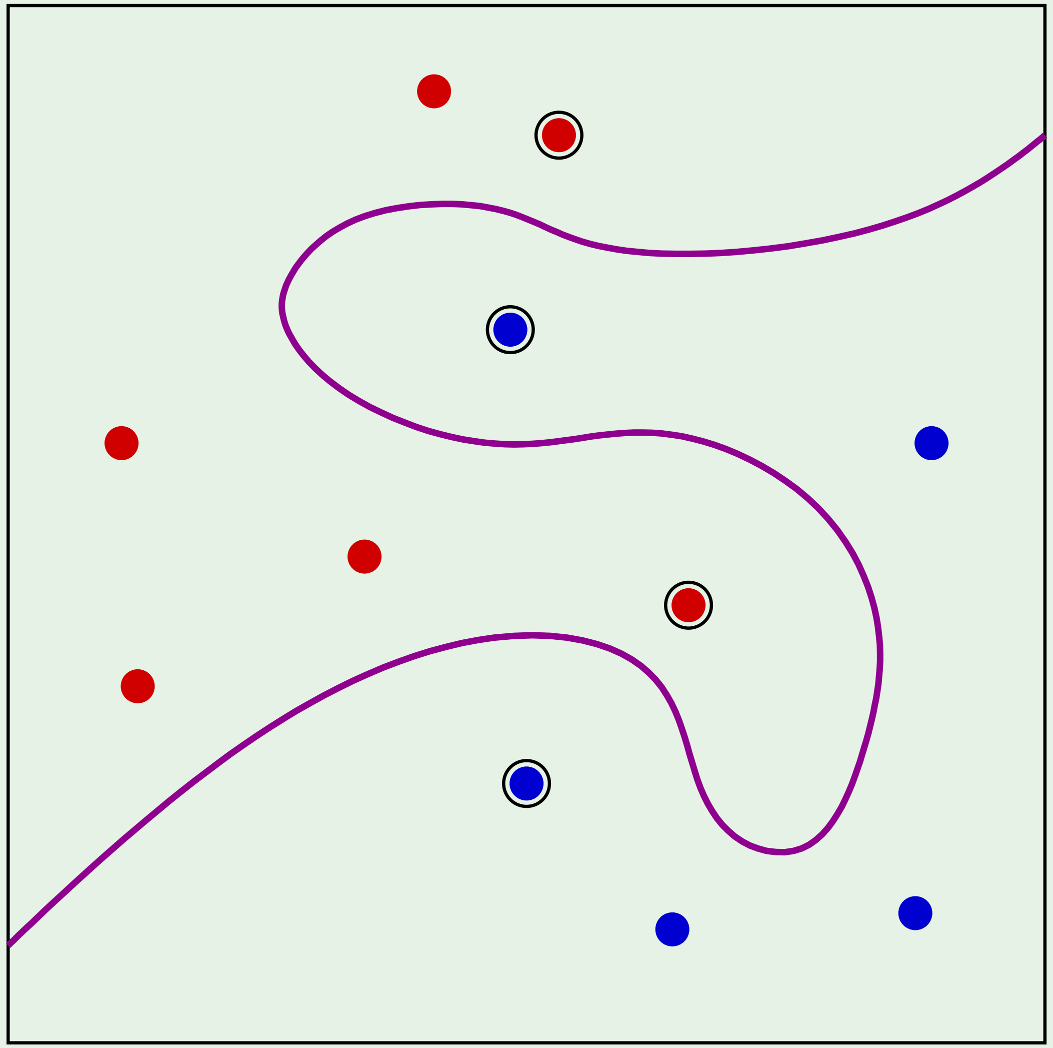
Since there are only 4 SV, there are only 4 parameters really expressing w in the Z-space. This is remarkable if we have transformed to a (e.g.) 100-d Z-space, meaning that w is a 100-d vector with only 4 non-zero elements. So the solution of only 4 SV suggests that effectively, in spite of the fact we use the glory of the 100-d space, we only have 4 parameters and so the generalization behavior will go with the 4 parameters - see below result.

Support vectors live in $\mathcal{Z}$ space

In $\mathcal{X}$ space, "pre-images" of support vectors

The margin is maintained in $\mathcal{Z}$ space

Note the equal distance between the SV's and the surface is only maintained in the Z-space



Generalization result

$$\mathbb{E}[E_{\text{out}}] \leq \frac{\mathbb{E}[\text{\# of SV's}]}{N-1}$$

Due to expectation value, we need to run several versions and get an average to guarantee the bound on Eout. If the expected value lives up to its name and we 'expect the expected value', then the Eout we get in a particular situation will be bounded by the familiar (#effective params/d.o.f./VC dim)/(#examples). The most important aspect is that, much like how QP was unaffected by the nature of the Z-space (i.e. high dimenisonal Z-space does not have a significant effect on computational difficulty of QP), the generalization ability is similarly unaffected - if we have 10 SVs and 1000 data points, we are in good shape regardless of the dimensionality of the space we visited in the mapping. However, note that we cannot just choose a very-high dimensional mapping and things will be fine, because we are still dependant on the # of SV's, so if we go through the machinery and end up with 500 SV's for 1000 data points, we are in trouble - the 'snake' above will traverse around every point and try to fit the data hopelessly .