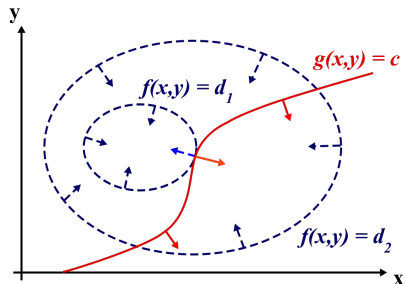
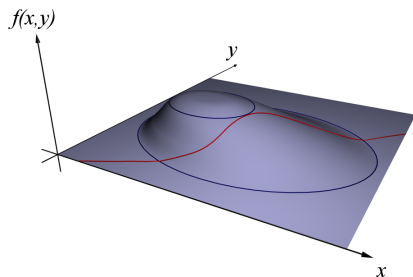


support vector machines

claudio castellini
reusing material by Holger Urbanek

dlr - german aerospace center

math refresher — Lagrange's multipliers



find $\arg \max_{(x,y)} f(x,y)$ subject to $g(x,y) = c$

\Downarrow

find $\arg \max_{(x,y,\lambda)} F(x,y,\lambda) = f(x,y) - \lambda[g(x,y) - c]$

...works fine with inequality constraints, too ($g(x,y) \geq c$), but requires that $\lambda \geq 0$

let's talk about linearly separable sets...

...and how to build a separating hyperplane, then.

first, a little refresher: a (hyper)plane of a d -dimensional space can be defined as a constrained linear function of d variables:

$$f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b = \left(\sum_{i=1}^d w_i x_i \right) + b = 0$$

where $\mathbf{w}, \mathbf{x} \in \mathbb{R}^d$ and $b \in \mathbb{R}$.

we can easily use a hyperplane to separate two (linearly separable) sets...

actually, there are infinite good hyperplanes.

a hyperplane as a separating surface

visualisable example: a hyperplane of \mathbb{R}^2 is the intersection of a 2-dimensional linear function f (which one can see as a flat surface in 3 dimensions) and the $(f = 0)$ -plane:

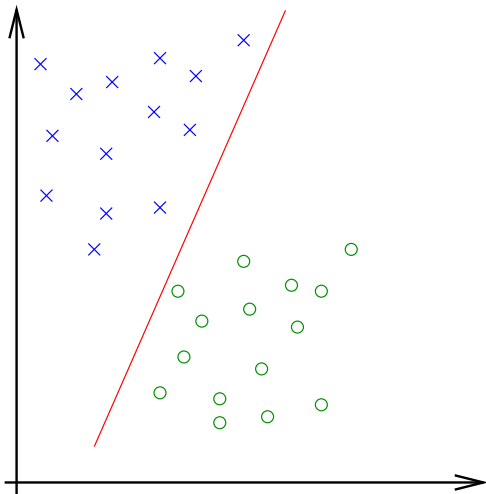
$$f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b = w_1x_1 + w_2x_2 + b = 0$$

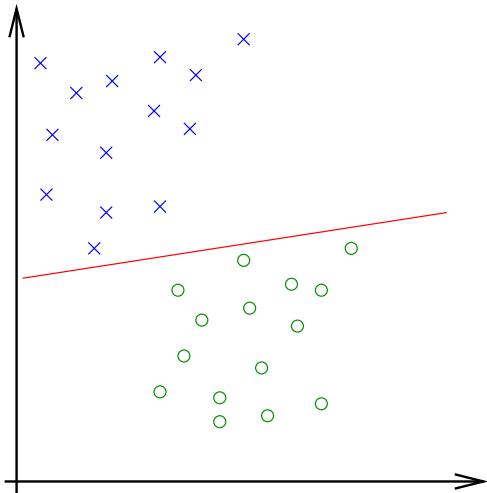
whence, for instance, $x_1 = -\frac{w_2}{w_1}x_2 - \frac{b}{w_1}$, which is simply a straight line. notice that \mathbf{w} and b characterise it completely: b is the *offset* and \mathbf{w} gives us the *direction* of the line

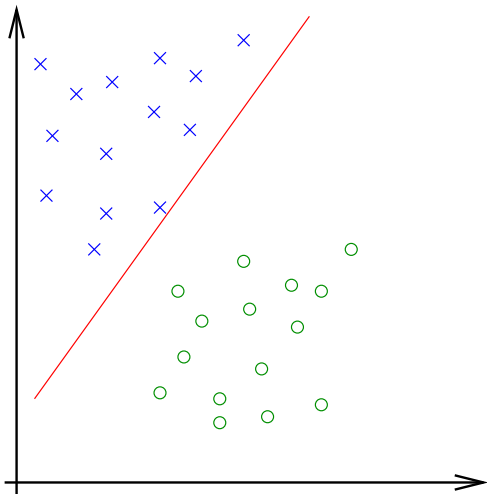
so one can classify $\bar{\mathbf{x}}$ as follows:

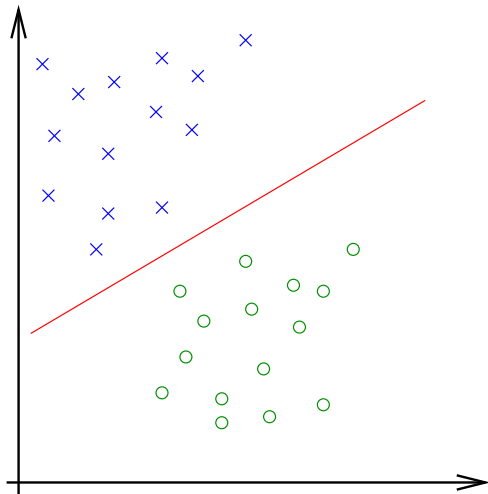
$$\bar{z} = \text{sign}(\mathbf{w} \cdot \bar{\mathbf{x}} + b)$$

remember: $\text{sign}(f(\mathbf{x}))$ is 1 if $f(\mathbf{x}) > 0$, -1 if $f(\mathbf{x}) < 0$









the optimal hyperplane

actually, too many solutions is usually as bad as no solution (*ill-posed problem*). so let us choose the best separating hyperplane!

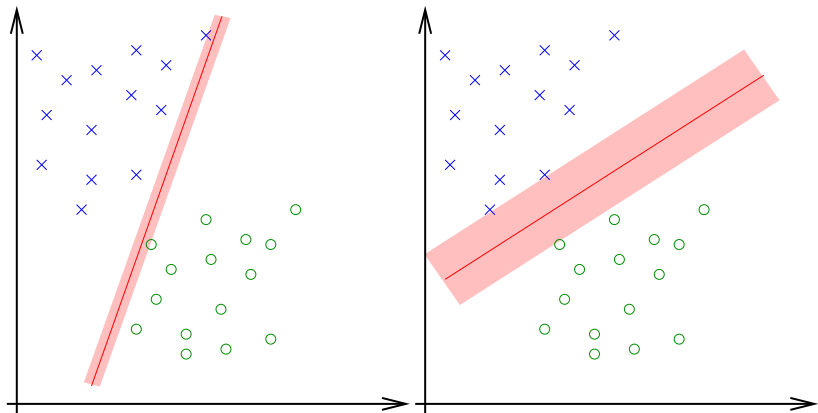
which one of them is "best"? what do we mean "best" here?

we mean, *the one which enjoys the maximum margin*

margin: two times the distance between the hyperplane and the closest point in a class

given the dataset we have, \mathcal{D} , it is the most robust with respect to noise

this criterion turns an ill-posed problem into a well-posed one — it gives us one and only one solution, instead of many (infinite)



non-maximum margin

maximum margin

how to find the optimal hyperplane

let then $\{\mathbf{x}_i, z_i\} \in \mathcal{D}$ with $z_i \in \{-1, 1\}$. it is then very convenient to ask that

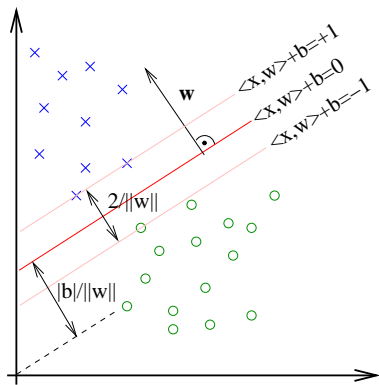
$$\mathbf{w} \cdot \mathbf{x}_i + b \geq 1 \text{ if } z_i = 1$$

$$\mathbf{w} \cdot \mathbf{x}_i + b \leq -1 \text{ if } z_i = -1$$

for each $i = 1, \dots, n$, that is to say

$$z_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 \text{ for all } i = 1, \dots, n$$

the margin is then $\frac{2}{\|\mathbf{w}\|}$, and we want to maximise it.



the primal formulation

therefore: minimise $\frac{1}{2}\|\mathbf{w}\|^2$, with the constraints:

$$z_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1 \geq 0 \text{ for all } i = 1, \dots, n$$

...does that ring a bell? use Lagrange multipliers!

"for each constraint $c \geq 0$, subtract λc from the objective function, with $\lambda \geq 0$ "

$$L_P = \frac{1}{2}\|\mathbf{w}\|^2 - \alpha_i (z_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1)$$

$$L_P = \frac{1}{2}\|\mathbf{w}\|^2 - \alpha_i z_i(\mathbf{w} \cdot \mathbf{x}_i + b) + \alpha_i$$

and this must hold for all \mathbf{x}_i at once:

$$L_P = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_i \alpha_i z_i(\mathbf{w} \cdot \mathbf{x}_i + b) + \sum_i \alpha_i$$

$$\alpha_i \geq 0 \text{ for all } i = 1, \dots, n$$

the primal formulation

so we need to minimise L_P :

$$L_P = \frac{1}{2} \|\mathbf{w}\|^2 - \sum_i \alpha_i z_i (\mathbf{w} \cdot \mathbf{x}_i) - \sum_i \alpha_i z_i b + \sum_i \alpha_i$$

in order to do that, let us then force the derivatives to 0:

$$\frac{\partial L_P}{\partial \mathbf{w}} = \mathbf{w} - \sum_i \alpha_i z_i \mathbf{x}_i = 0$$

$$\frac{\partial L_P}{\partial b} = - \sum_i \alpha_i z_i = 0$$

notice that this way we can evaluate $\mathbf{w} = \sum_i \alpha_i z_i \mathbf{x}_i$ (b can also be easily calculated) so that we can now build the separating hyperplane:

$$f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b$$

the dual formulation

now substitute these back into L_P !

$$L_D = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j z_i z_j (\mathbf{x}_i \cdot \mathbf{x}_j)$$

$$\sum_i \alpha_i z_i = 0$$

$$\alpha_i \geq 0 \text{ for all } i = 1, \dots, n$$

L_D is called *dual* of L_P . to maximize the margin, we either minimize L_P or maximize L_D .

it is a convex quadratic programming problem. notwithstanding its ugly face, it is tractable!

support vectors

optimality can also be achieved using the *Karush-Kuhn-Tucker* conditions:

$$\frac{\partial L_P}{\partial \mathbf{w}} = \mathbf{w} - \sum_i \alpha_i z_i \mathbf{x}_i = 0$$

$$\frac{\partial L_P}{\partial b} = - \sum_i \alpha_i z_i = 0$$

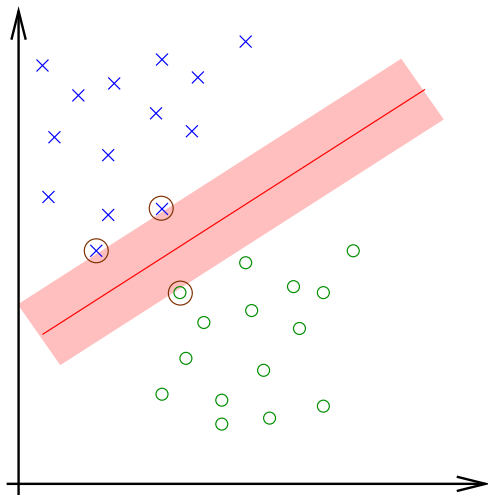
$$z_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1 \geq 0 \quad \text{for all } i = 1, \dots, n$$

$$\alpha_i \geq 0 \quad \text{for all } i = 1, \dots, n$$

$$\alpha_i(z_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1) = 0 \quad \text{for all } i = 1, \dots, n$$

never mind all this; but look at the last condition, which defines *support vectors*

support vectors



think of the separating plane "leaning" on the support vectors —
support vectors *support the plane!*

support vectors

consider it again:

$$\alpha_i \cdot (z_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1) = 0$$

clearly, for non-support-vectors $z_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1 > 0$ so α_i must be 0. in other words, Lagrange multipliers are active ($\alpha_i > 0$) only for support vectors.

but now remember how the solution looked like; in particular,

$$\mathbf{w} = \sum_i \alpha_i z_i \mathbf{x}_i$$

which means that *only support vectors contribute to the solution*.

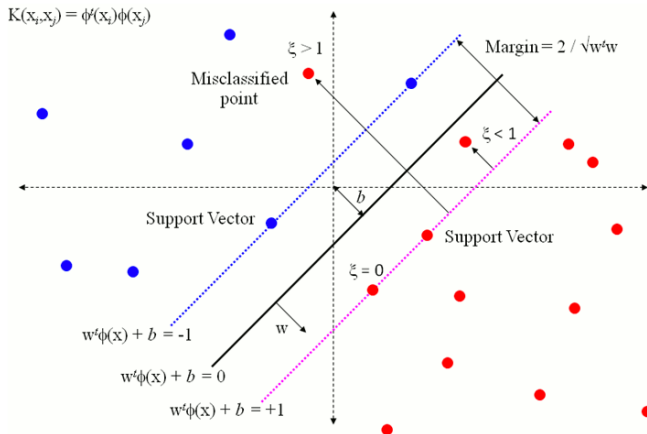
non-support vectors *do not contribute* to the solution, and they can be neglected!

consider the mechanical analogy again: if a point does not support the plane, it can be happily shifted around / removed (unless it becomes a new support vector!)

the non-separable case

we've just set up a minimisation problem which will yield a solution in the linearly separable case (*hard margin*)

what if the problem is not linearly separable? in that case we can introduce n *slack variables*, usually denoted by $\xi_i \geq 0$, and ask that they are minimum (*soft margin*)



the non-separable case

the constraints become

$$\mathbf{w} \cdot \mathbf{x}_i + b \geq 1 - \xi_i \quad \text{if } z_i = 1$$

$$\mathbf{w} \cdot \mathbf{x}_i + b \leq -1 + \xi_i \quad \text{if } z_i = -1$$

that is

$$z_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1 + \xi_i \geq 0 \quad \text{for all } i = 1, \dots, n$$

...and the functional to be minimised is now

$$\frac{1}{2} \|\mathbf{w}\|^2 + C \sum_i \xi_i$$

where $C > 0$ balances a harder / softer margin

the non-separable case

the solution is surprisingly similar to the separable-case one: maximise

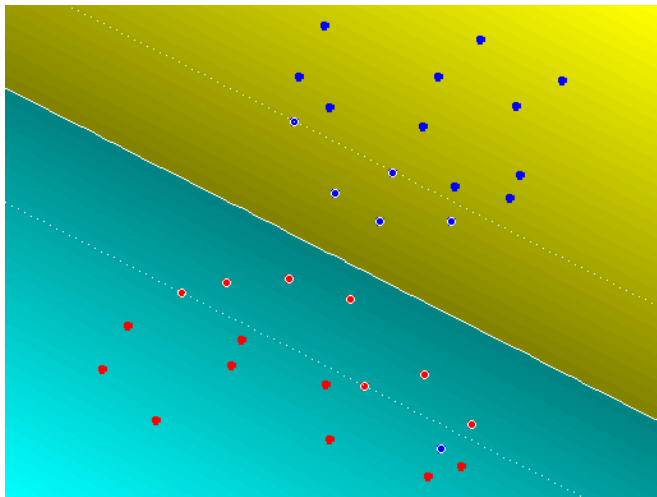
$$L_D = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j z_i z_j (\mathbf{x}_i \cdot \mathbf{x}_j)$$

subject to

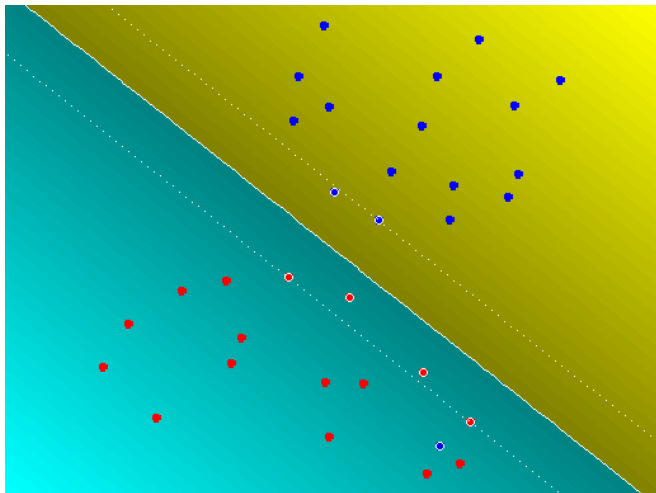
$$\sum_i \alpha_i z_i = 0$$

$$0 \leq \alpha_i \leq C \text{ for all } i = 1, \dots, n$$

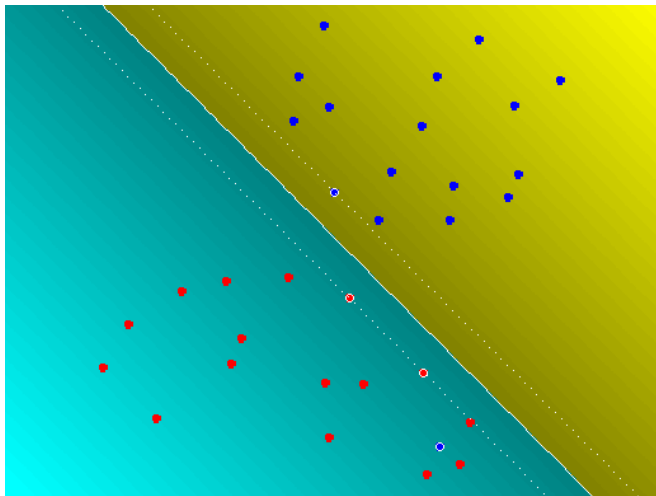
$$C = 1$$



$C = 10$



$C = 100$



and now for the last step

so far we've found a good algorithm to find a separating hyperplane in the linearly separable case

in the non-separable case it will work indeed, but with poorer results

not because of the algorithm, but rather because of the non-linear-separability

there's only a "limited amount of separation" that a humble straight line can provide...

so, it's time to use a kernel!

the kernel trick

the dual problem only contains *inner products* between vectors...

$$L_D = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j z_i z_j (\mathbf{x}_i \cdot \mathbf{x}_j)$$

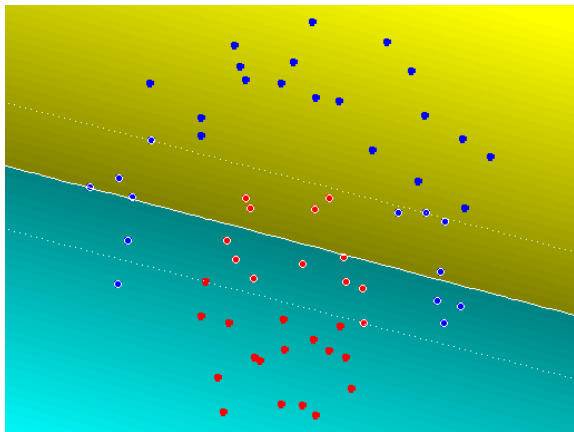
so we just substitute a kernel $K(\cdot, \cdot)$ to the inner product:

$$L_D = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j z_i z_j K(\mathbf{x}_i, \mathbf{x}_j)$$

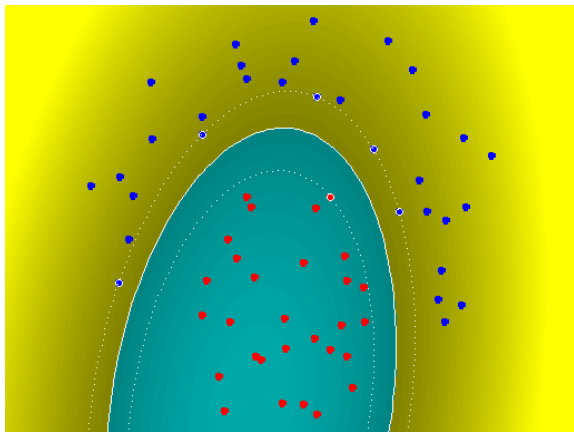
what else do we have to change?

nothing!

the kernel trick



the kernel trick



the kernel trick

in fact, once we've found the optimal α_i s by maximising

$$L_D = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j z_i z_j K(\mathbf{x}_i, \mathbf{x}_j)$$

we can just plug them in the expression of the separating hyperplane. here, too, only inner products are required (remember that $\mathbf{w} = \sum_i \alpha_i z_i \mathbf{x}_i$):

$$f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b = \sum_i \alpha_i z_i (\mathbf{x}_i \cdot \mathbf{x}) + b$$

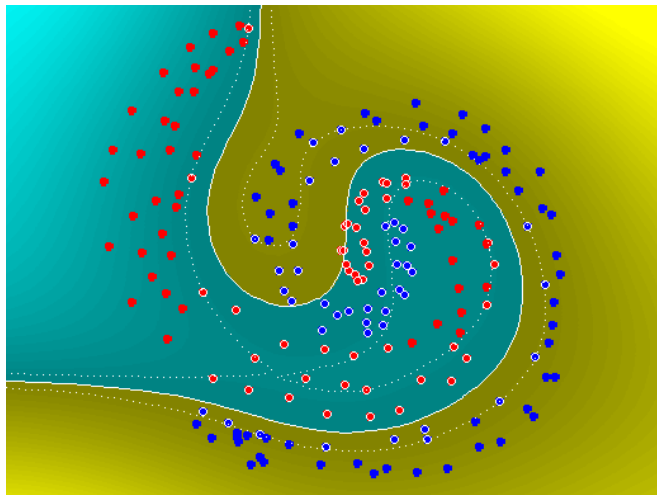
which we can replace with

$$f(\mathbf{x}) = \sum_i \alpha_i z_i K(\mathbf{x}_i, \mathbf{x}) + b = \sum_i \alpha_i z_i [\phi(\mathbf{x}_i) \cdot \phi(\mathbf{x})] + b$$

so we solve a linearly separable classification problem in the feature space, then map the solution back to the input space.

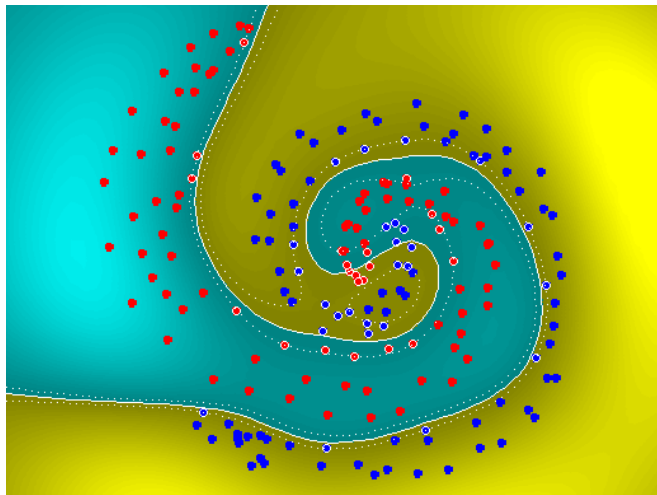
Gaussian kernel

$$\sigma = 1, C = 1000$$



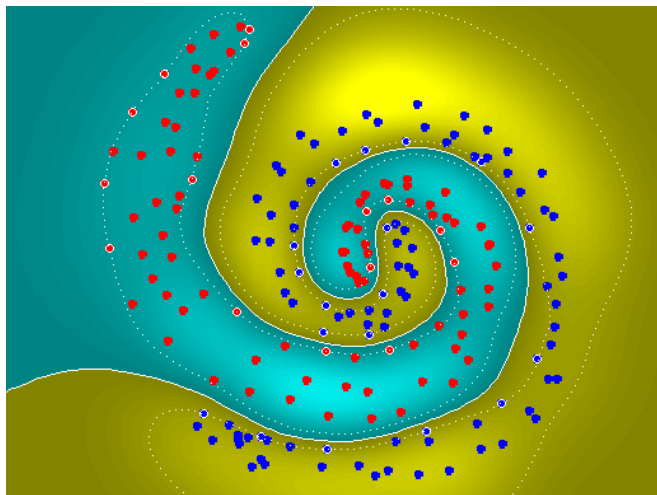
Gaussian kernel

$$\sigma = 0.5, C = 1000$$



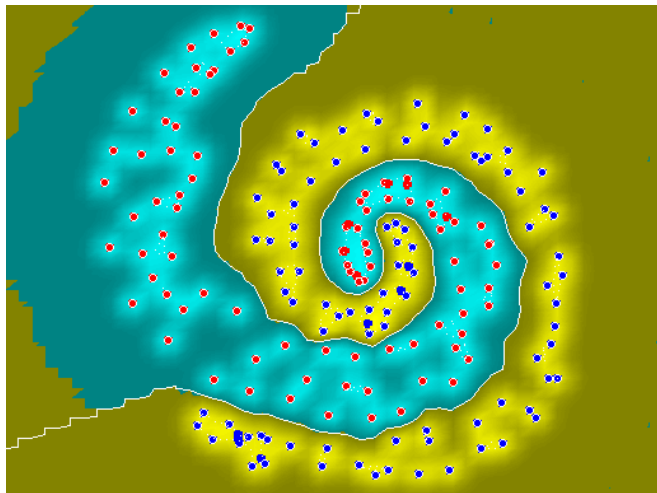
Gaussian kernel

$$\sigma = 0.25, C = 1000$$



Gaussian kernel

$$\sigma = 0.05, C = 1000$$



Gaussian kernel

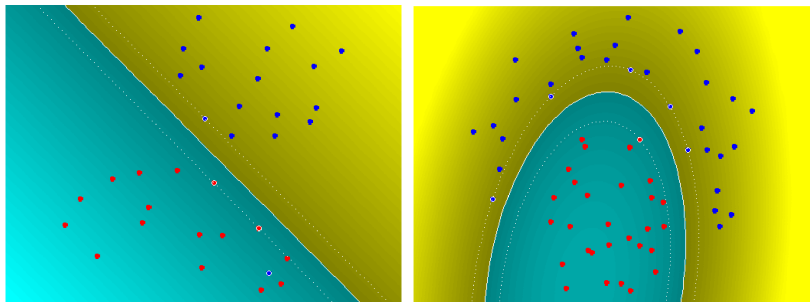
$$\sigma = 0.005, C = 1000$$



Support Vectors

what are support vectors?

support vectors are the representatives of each class, which are closest to the separating border



in other words, they are the *ambiguous* samples - those for which the classification is likely to fail

"if you can classify those ones, all the rest is easy"

Support Vector Regression

SVMs can be used for regression, too. given $\mathcal{D} = \{\mathbf{x}_i, z_i\}_{i=1}^n$ with $z_i \in \mathbb{R}$, instead of guessing a class as $\text{sign}(\mathbf{w} \cdot \mathbf{x} + b)$, just go... for the function itself!

guess $\hat{f}(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b$, or better, $\hat{f}(\mathbf{x}) = K(\mathbf{w}, \mathbf{x}) + b$ where K is the kernel function

(think carefully of that: is a linear approximation enough? no? then just... switch the kernel!)

Support Vector Regression

recall the classification case, linearly separable problems (no slack variables), a.k.a. "hard margin"

to find \mathbf{w} and b , minimise

$$\frac{1}{2} \|\mathbf{w}\|^2$$

subject to

$$z_i - K(\mathbf{w}, \mathbf{x}_i) - b \leq \epsilon$$

$$-z_i + K(\mathbf{w}, \mathbf{x}_i) + b \leq \epsilon$$

where $\epsilon > 0$ is the tolerance of our approximation, i.e., the "insensitive band" within which we won't punish any error

Support Vector Regression

more realistically (take infeasible constraints into account), we introduce slack variables $\xi_i, \xi_i^* \geq 0$ and minimise

$$L_P = \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^n (\xi_i + \xi_i^*)$$

(*primal* problem) subject to

$$z_i - K(\mathbf{w}, \mathbf{x}_i) - b \leq \epsilon + \xi_i$$

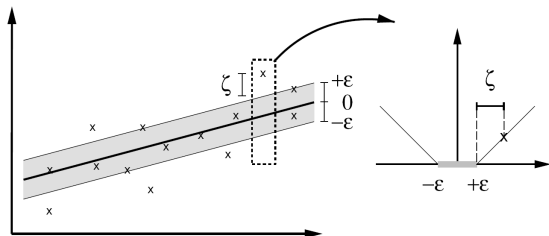
$$-z_i + K(\mathbf{w}, \mathbf{x}_i) + b \leq \epsilon + \xi_i^*$$

compare with the case of classification. here the margin must be soft to allow for approximation errors

how do you apply Lagrange multipliers to this case to get the "dual" problem L_D ?
(very similar to the classification case)

Support Vector Regression

implicitly we are establishing a "loss" functional, the so-called " ϵ -insensitive" error function



[Smola and Schölkopf, 2003]

in general, SVMs are about minimising a ("primal" or "dual") cost functional

$$L = \text{regulariser}(\mathbf{w}) + C \cdot \text{loss}(\mathcal{D})$$

where $C > 0$ controls the trade-off between "flatness" of the solution and the accuracy on the dataset \mathcal{D}

Support Vector Regression

everything else carries on smoothly, just like in classification:

$$\mathbf{w} = \sum_{i=1}^n (\alpha_i - \alpha_i^*) \phi(\mathbf{x}_i), \quad \text{therefore}$$

$$\hat{f}(\mathbf{x}) = K(\mathbf{w}, \mathbf{x}) + b = \sum_{i=1}^n (\alpha_i - \alpha_i^*) K(\mathbf{x}_i, \mathbf{x}) + b$$

and the solution is a weighted sum of "basis" functions, given by the choice of a particular kernel

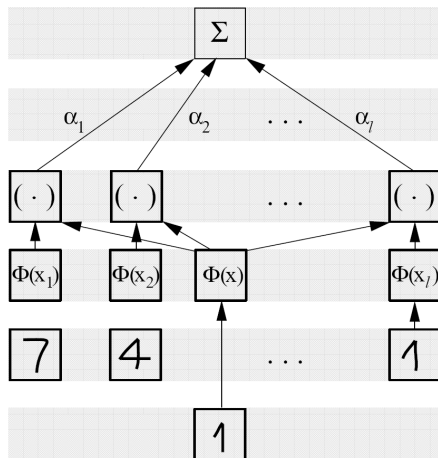
a sum of linear functions if $K(\mathbf{x}_1, \mathbf{x}_2) = \mathbf{x}_1 \cdot \mathbf{x}_2$

a sum of k -degree polynomials if $K(\mathbf{x}_1, \mathbf{x}_2) = (g + \mathbf{x}_1 \cdot \mathbf{x}_2)^k$

a sum of Gaussians with variance σ^2 if $K(\mathbf{x}_1, \mathbf{x}_2) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{|\mathbf{x}_1 - \mathbf{x}_2|^2}{2\sigma^2}}$

...and that's all for SVR, folks. have a look at a few graphs:

Support Vector Regression



output $\Sigma \alpha_i k(x, x_i) + b$

weights

dot product $(\Phi(x) \cdot \Phi(x_i)) = k(x, x_i)$

mapped vectors $\Phi(x_i), \Phi(x)$

support vectors $x_1 \dots x_l$

test vector x

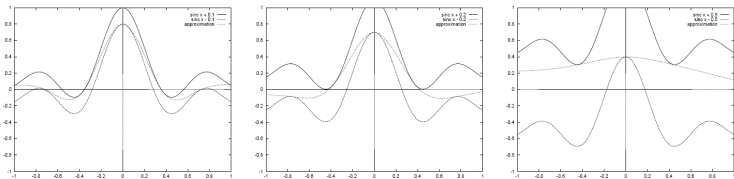


Figure 3: Left to right: approximation of the function $\text{sinc } x$ with precisions $\epsilon = 0.1, 0.2,$ and 0.5 . The solid top and the bottom lines indicate the size of the ϵ -tube, the dotted line in between is the regression.

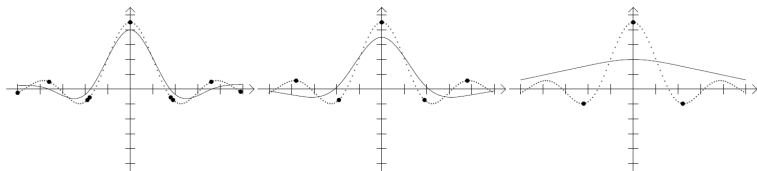


Figure 4: Left to right: regression (solid line), datapoints (small dots) and SVs (big dots) for an approximation with $\epsilon = 0.1, 0.2,$ and 0.5 . Note the decrease in the number of SVs.

given \mathcal{D} , C and ϵ , SVM choose the flattest possible approximation

further notes on SVMs

Support Vector Machines return a *sparse* solution: in

$$\hat{f}(\mathbf{x}) = \sum_{i=1}^n \alpha_i K(\mathbf{x}_i, \mathbf{x}) + b$$

only some (hopefully very few) α s are non-zero, which means that predicting is easier / faster

but as well, few SVs (say the percentage of SVs over n) mean that the problem was easy, therefore that a good relationship exists

anyway, *training* an SVM can be computationally heavy: $O(n^3)$ especially as C grows - need to "take more into account" the error over \mathcal{D} ...

further notes on SVMs

1. the kernel is a similarity function: $K(a, b)$ tells you how similar a and b are

think of a Gaussian, consider $\sum_{i=1}^n \alpha_i K(\mathbf{x}_i, \mathbf{x}) + b$ once again: in prediction, the closer \mathbf{x} to \mathbf{x}_i , the more important summand i is

2. choosing the kernel is tantamount to choosing a hypothesis space \mathcal{H} (RKHS, Reproducing Kernel Hilbert Space)

the kernel "induces" \mathcal{H} , and \mathcal{H} is the space of functions among which you are going to find the solution to the classification / regression problem

further notes on SVMs

3. the Gaussian kernel induces an \mathcal{H} which has *infinite* dimension

you can choose σ arbitrarily, therefore you can accommodate any samples, no matter how close they are
so how do we avoid overfitting?

4. using regularisation:

that is, "keep the \mathbf{w} small", and this is exactly equivalent to **maximising the margin**

5. how do we then balance regularisation vs. loss (under/overfitting)?

guess that... need to find optimal C and σ

how do we find C, σ ? cross-validation and grid-search — topics of the next lecture

references

Christopher J. C. Burges

A Tutorial on Support Vector Machines for Pattern Recognition, 1998

Alex J. Smola and Bernhard Schölkopf

A Tutorial on Support Vector Regression, 2003

Tomaso Poggio and Steve Smale

The Mathematics of Learning: Dealing with Data, 2003

Arnulf B. A. Graf, Felix A. Wichmann, Heinrich H. Bülthoff and Bernhard Schölkopf

Classification of Faces in Man and Machine, 2006