

kernels

claudio castellini
reusing material by Holger Urbanek

dlr - german aerospace center

classification

we're given a (two-class) classification problem: $\mathcal{D} = \{\mathbf{x}_i, z_i\}$ where $\mathbf{x}_i \in I$, z_i is either 1 or -1 and $i = 1, \dots, n$

find a function $f(\mathbf{x}), \mathbf{x} \in I$ such that $\text{sign}(f(\mathbf{x}))$ classifies new samples as best as it can, that is

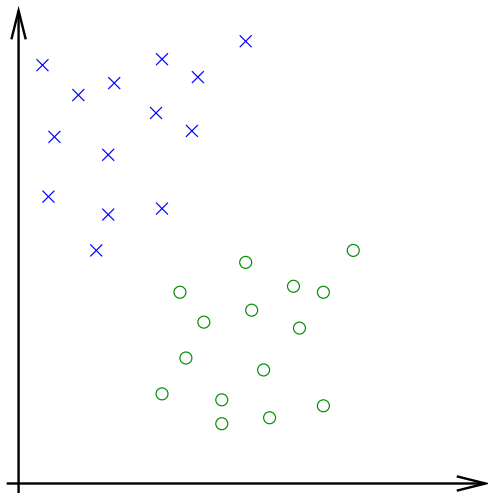
$$f(\mathbf{x}) > 0 \text{ if and only if } z = 1$$

$$f(\mathbf{x}) < 0 \text{ if and only if } z = -1$$

where z is the true label associated with \mathbf{x} .

(we would also like f to get it right with the values in \mathcal{D} ; that is, $f(\mathbf{x}_i) > 0$ if and only if $z_i = 1$ and analogously for $z_i = -1$, but this really is secondary)

a very desirable characteristic:



linear separability

the quest for linear separability

now, obvious fact: **if** the dataset is linearly separable **then** a hyperplane exists which acts as a classifier

(hm, obvious?)

a hyperplane is a linear function in the input space, therefore it is easily found (linear functions are the simplest functions around)!

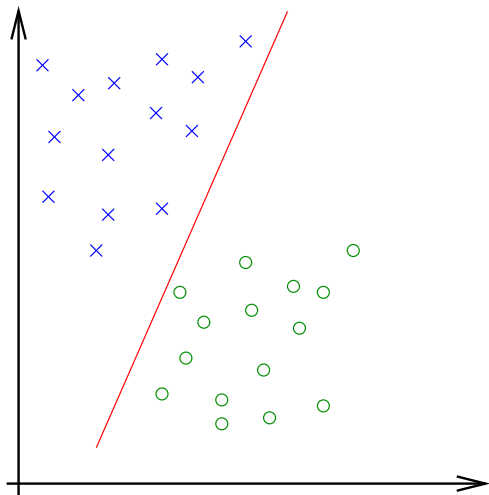
(in 2D, a hyperplane is a straight line

in 3D, a hyperplane is a plane

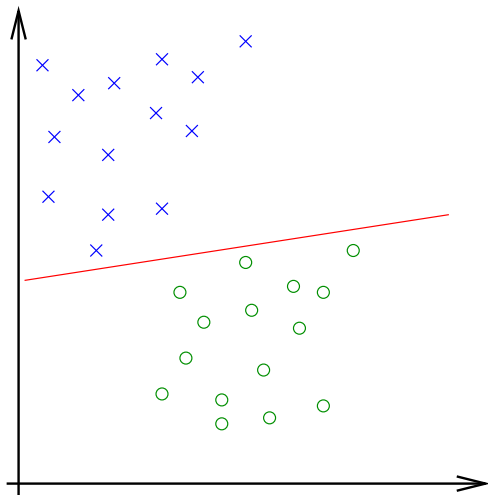
in 4D, ...?)

assume that we have found a good algorithm to build such a classifier.

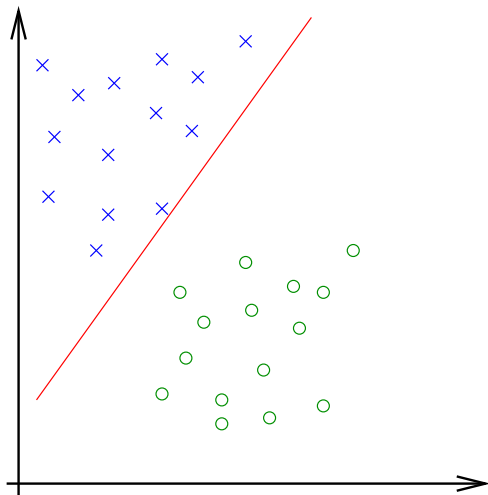
the quest for linear separability



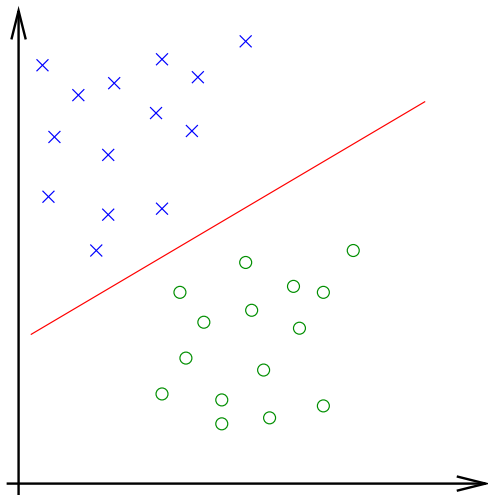
the quest for linear separability



the quest for linear separability



the quest for linear separability



the quest for linear separability

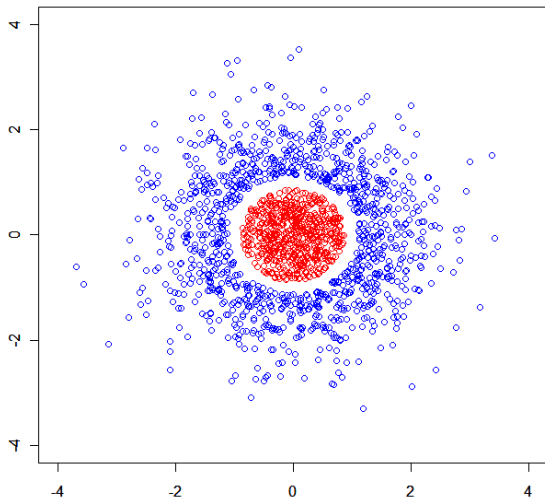
so, if the dataset is linearly separable, it is easy to build a classifier

now, what if the dataset is *not* linearly separable?

- ▶ either we build a more "rugged" separating surface (no longer a hyperplane), for example, a polynomial, or a sum of exponentials/logarithmics, etc.

(in this case, we might need to devise a completely new algorithm...)

- ▶ alternatively, why don't we try to *make the problem linearly separable* and then re-use the old machinery?



yes, we can

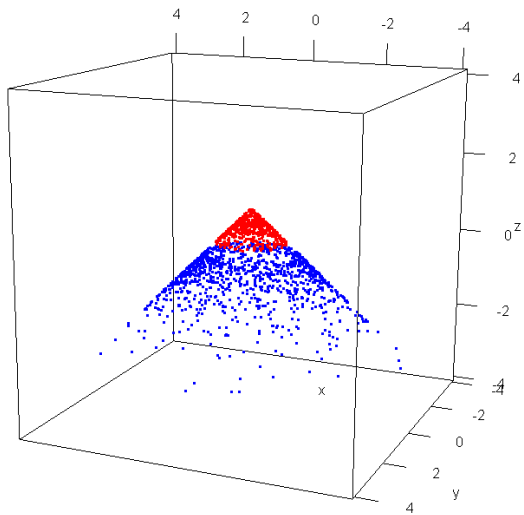
the trick is as simple as that: to map the input space I onto a new space \mathcal{F} where our data *will be linearly separable!*

\mathcal{F} usually has *more dimensions* than I , and that is where we embed the trick

\mathcal{F} is usually called *feature space*

so we need a mapping $\phi : I \rightarrow \mathcal{F}$ contrived in such a way that the image of I through ϕ is linearly separable in \mathcal{F}

try and figure that out!



building ϕ

in this case, we add a third dimension and then plan to separate the two classes with a horizontal plane (i.e., orthogonal to the new axis)

here I is \mathbb{R}^2 , \mathcal{F} is \mathbb{R}^3 and, given $\mathbf{x} \triangleq (x_1, x_2)$,

$$\phi(\mathbf{x}) = (x_1, x_2, -\|\mathbf{x}\|)$$

so, the previous classifier, which could work on linearly separable data sets only, will work in this case, too: just use ϕ !

(hmmm... could we actually achieve the same result by mapping I onto 2 dimensions? onto 1 dimension?)

and now a big question: how do we determine a suitable ϕ in the general case?

kernels

it actually turns out that... we can't.

no, seriously. in the general case there is no hope of mechanically building a good mapping ϕ .

if you can figure out one by looking at the data, build it and use it.

otherwise, resort to the **kernel trick**:

1. use a "generic" ϕ , mapping I onto a very high dimensional \mathcal{F}
why? because more dimensions means more chances that the problem will be linearly separable in \mathcal{F} !
2. computing ϕ becomes then cumbersome (when not impossible).
compute instead the associated *kernel* K

what is a kernel?

given an input space I , a *kernel* is a binary function from $I \times I$ to the reals:

$$K : I \times I \rightarrow \mathbb{R}$$

$$K(\mathbf{a}, \mathbf{b}) = c \text{ where } c \in \mathbb{R} \text{ and } \mathbf{a}, \mathbf{b} \in I$$

intuition: K will tell us "how similar to each other" are \mathbf{a} and \mathbf{b} .

peculiar characteristic: a kernel $K(\mathbf{a}, \mathbf{b})$ can always be expressed as the inner product of the images of its arguments through a function ϕ :

$$K(\mathbf{a}, \mathbf{b}) = \phi(\mathbf{a}) \cdot \phi(\mathbf{b})$$

what is a kernel?

is every $K(\mathbf{a}, \mathbf{b})$ feasible? not really:

firstly, the space induced by ϕ must be a *Hilbert space*:

- linear space
- with inner product
- complete to the norm

then a function $K(\mathbf{a}, \mathbf{b})$ is a Kernel *Mercer's condition* iff:

- for any $g(\mathbf{x})$ such that $\int g(\mathbf{x})^2 d\mathbf{x}$ is finite,

$$\int K(\mathbf{a}, \mathbf{b}) g(\mathbf{a}) g(\mathbf{b}) d\mathbf{a} d\mathbf{b} \geq 0$$

- (very inconvenient to prove.)

if K stands Mercer's condition, then there exists a ϕ such that

$$K(\mathbf{a}, \mathbf{b}) = \phi(\mathbf{a}) \cdot \phi(\mathbf{b})$$

is a kernel enough?

now suppose we have found a kernel which corresponds to a good candidate ϕ . is the kernel enough? can we use it in place of ϕ itself?

answer is yes, as long as we only need to compute inner products of the form $\phi(\mathbf{a}) \cdot \phi(\mathbf{b})$, and *never* single-vector mappings such as, e.g., $\phi(\mathbf{a})$

good news: it actually *is* the case, at least for support vector machines and for all other "kernel" methods!

we will usually pick a kernel from a set of well-known ones, for which Mercer's condition has been proved.

(and even if Mercer's condition is not fulfilled, it often works fine — in some cases, the optimisation won't converge.)

in the previous example...

we had $\phi(\mathbf{x}) = (x_1, x_2, -\|\mathbf{x}\|)$, which made our problem linearly separable
in this case the kernel is

$$K(\mathbf{a}, \mathbf{b}) = \phi(\mathbf{a}) \cdot \phi(\mathbf{b}) = (a_1, a_2, -\|\mathbf{a}\|) \cdot (b_1, b_2, -\|\mathbf{b}\|) =$$

$$a_1 b_1 + a_2 b_2 + \|\mathbf{a}\| \|\mathbf{b}\| = \mathbf{a} \cdot \mathbf{b} + \|\mathbf{a}\| \|\mathbf{b}\| = \|\mathbf{a}\| \|\mathbf{b}\| (1 + \cos \alpha)$$

(recall that $\mathbf{a} \cdot \mathbf{b} = \|\mathbf{a}\| \|\mathbf{b}\| \cos \alpha$)

the other way around

now rather than starting from ϕ and building K , let us do it the other way around: consider $K(\mathbf{a}, \mathbf{b}) \triangleq (\mathbf{a} \cdot \mathbf{b})^2$ with $\mathbf{a}, \mathbf{b} \in \mathbb{R}^2$

for this K , one possible ϕ is

$$\phi(\mathbf{x}) = \begin{pmatrix} x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \end{pmatrix}$$

but also

$$\phi(\mathbf{x}) = \frac{1}{\sqrt{2}} \begin{pmatrix} x_1^2 - x_2^2 \\ 2x_1x_2 \\ x_1^2 + x_2^2 \end{pmatrix}$$

and also others.

try to evaluate $\phi(\mathbf{a}) \cdot \phi(\mathbf{b})$, you're going to get $K(\mathbf{a}, \mathbf{b})$ back!

kernels

what kernels are then in current use?

linear: $K(\mathbf{a}, \mathbf{b}) \triangleq \mathbf{a} \cdot \mathbf{b}$

polynomial:

$$K(\mathbf{a}, \mathbf{b}) \triangleq (\mathbf{a} \cdot \mathbf{b})^p \quad \text{or} \quad K(\mathbf{a}, \mathbf{b}) \triangleq (\mathbf{a} \cdot \mathbf{b} + 1)^p$$

Gaussian (radial basis function):

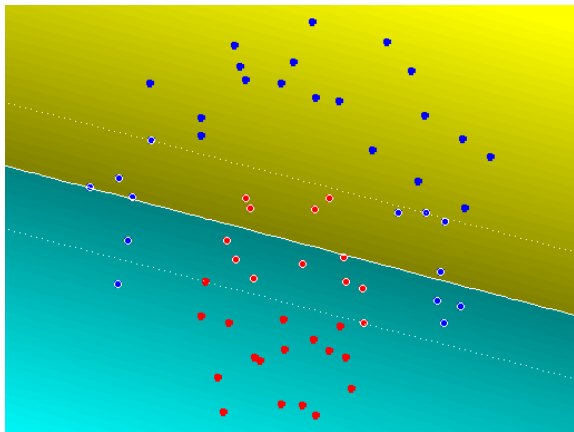
$$K(\mathbf{a}, \mathbf{b}) \triangleq \exp\left(-\frac{\|\mathbf{a} - \mathbf{b}\|^2}{2\sigma^2}\right)$$

sigmoid:

$$K(\mathbf{a}, \mathbf{b}) \triangleq \tanh(k(\mathbf{a} \cdot \mathbf{b}) - d) \quad \text{for some } k, d > 0$$

example

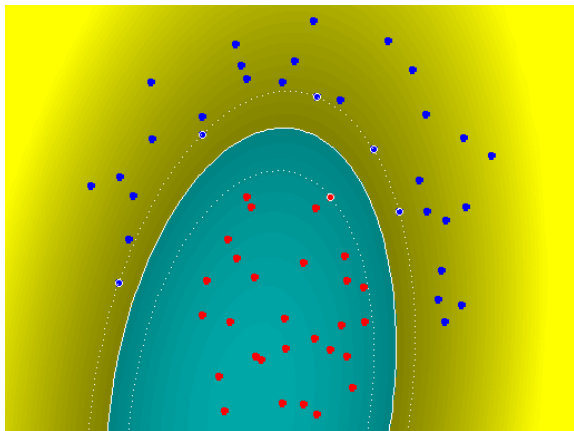
non-linearly-separable data set, linear kernel



...not very good indeed.

example (cont'd)

the same non-linearly-separable data set, quadratic kernel $(\mathbf{a} \cdot \mathbf{b})^2$



...picture this in three dimensions!

conclusions

a kernel is a "plug'n'play" function using which we can change the complexity of the separating surface

at the same time it is an affinity function: it tells us how similar two samples are

using it, we can reuse all the machinery for linear separability...

...and this machinery is the subject of the next lecture (support vector machines)