

# generalisation and cross-validation

claudio castellini

dlr - german aerospace center

## let us take a step back

what are we *really* after here?

the *approximation of an unknown function*

we have convinced ourselves that there is a functional relationship between an input space  $I$  and an output space  $O$ . that means

$$f : I \rightarrow O \text{ that is } f(\mathbf{x}) = z \text{ where } \mathbf{x} \in I, z \in O$$

plus the requirement that  $\mathbf{x}$  won't map onto two different things:

$$\text{if } f(\mathbf{x}) = z_1 \text{ and } f(\mathbf{x}) = z_2 \text{ then } z_1 = z_2$$

so, a function really is an infinite set of ordered pairs:

$$f = \{(\mathbf{x}, z) \text{ such that } \mathbf{x} \in I, z = f(\mathbf{x}) \in O\}$$

or if you prefer, a subset of the Cartesian product of  $I$  and  $O$

$$f \subseteq I \times O$$

## what do we know about $f$ ?

essentially, only  $\mathcal{D} = \{(\mathbf{x}_i, z_i)\}_{i=1}^n \subset f$

how do we build an approximant  $\hat{f}$  such that it "behaves" like  $f$  even on  $f \setminus \mathcal{D}$  ?

if we **strictly** know  $\mathcal{D}$  **only**, it is an ill-posed problem.

the only requirement we can ask for is that  $f \approx \hat{f}$  on  $\mathcal{D}$ , in which case there are too many good solutions!

the simplest is

$$\hat{f}(\mathbf{x}) = \begin{cases} z_i & \text{if } \mathbf{x} = \mathbf{x}_i, \\ 0 & \text{otherwise} \end{cases}$$

now how useful is this "approximation" ?

## assumptions

we actually need to stick in some external knowledge. for instance,

- ▶ that  $\hat{f}$  be "reasonable": continuous, differentiable to the  $k$ th order; a simpler version is
- ▶ that  $\hat{f}$  be a "reasonable composition" of "reasonable basic functions"; e.g., a sum of non-linear functions
- ▶ that  $\hat{f}$  be "not too complex" (regular); e.g., "in linear regression, keep the coefficients small!"

in other words, we need to choose a *hypothesis space*  $\mathcal{H}$ , to which we *restrict our search*

notice that this implicitly reflects our beliefs about  $f$ , the real, unknown function we're after...

if  $\mathcal{H}$  is really badly chosen, then we're very likely to get a bad result, no matter how smart our method is

## assumptions

$\mathcal{H}$  will be a set of "reasonable" functions, such as:

- ▶  $\mathcal{H} = \{\hat{f}(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x}, \mathbf{w} \in \mathbb{R}^k\}$   
the linear functions in  $k$  dimensions
- ▶  $\mathcal{H} = \{\hat{f}(x) = \mathbf{w} \cdot (1, x, x^2, \dots, x^d), \mathbf{w} \in \mathbb{R}^{d+1}\}$   
 $d$ th degree polynomials in 1 dimension
- ▶  $\mathcal{H} = \{\hat{f}(\mathbf{x}) = \sum^k \mathbf{w} \cdot \mathcal{N}(\mu_i, \Sigma_i), \mathbf{w} \in \mathbb{R}^k, \mu_i \in \mathbb{R}^d, \Sigma_i \in \text{mat}(d \times d)\}$   
sum of  $k$  Gaussians in  $d$  dimensions

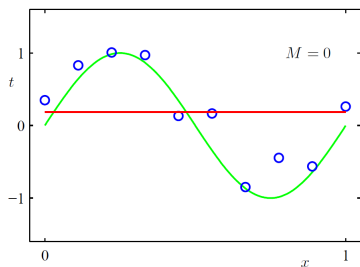
and then we look for the "best" element  $\hat{f} \in \mathcal{H}$  with respect to  $f$

## what about $\mathcal{H}$ ?

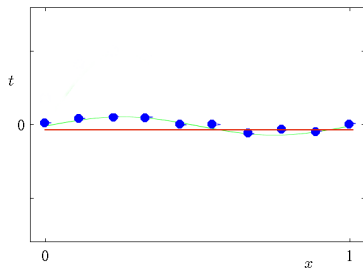
**intuition 1:** if  $\mathcal{H}$  is "too small",  $\hat{f}$  could be "not flexible enough" even to accommodate  $\mathcal{D}$  (let alone  $f \setminus \mathcal{D} \dots$ )

recall linear regression with a constant function

this is called *underfitting* and is generally bad, although consider the linear regression example again, consider  $\mathcal{D}$  only (the blue dots) and now try to zoom out. how does it look?



$\Rightarrow$



## what about $\mathcal{H}$ ?

**intuition 2:** if  $\mathcal{H}$  is "large",  $\hat{f}$  will be flexible enough to accommodate all of  $\mathcal{D}$ . but what happens if we fit  $\mathcal{D}$  too well?

recall linear regression with 9th order polynomial

this is called *overfitting* and is definitely bad

no generalisation, i.e., almost surely bad approximation on  $f \setminus \mathcal{D}$ , plus you will be learning the noise!

**solution:** initially choose a "large"  $\mathcal{H}$ ; then "restrict" it via regularisation

keep the parameters / coefficients small

keep the number of basic functions small

keep it simple!

use grid-search (that really means "try them all"...)

...but now, how do we really know how large  $\mathcal{H}$  is ?

## the VC dimension

there is actually a rigorous (although not very useful) characterisation of the "size" / "power" ("capacity") of  $\mathcal{H}$ : the Vapnik-Chervonenkis (VC) dimension

(we restrict to the case of binary classification:  $f(\mathbf{x}) = \{-1 \text{ or } 1\}$ )

consider a hypothesis space  $\mathcal{H}$

then the VC dimension of  $\mathcal{H}$  is the maximum number of samples that can be *shattered* by a  $\hat{f} \in \mathcal{H}$

shattering means: there is *at least* one set of samples,  $\mathcal{D}$ , such that  
for any possible labeling of  $\mathcal{D}$ ,  
there is  $\hat{f} \in \mathcal{H}$  which correctly classifies all samples.

## the VC dimension

there is a nice "game-theoretic" interpretation of the VC dimension: fix a  $\mathcal{H}$  and let  $n = 1$ ; then

- ▶ I choose the layout of  $n$  samples we need to classify
- ▶ you choose the labelling
- ▶ I try to shatter them

if I can shatter them, then repeat with  $n + 1$ ;

if I cannot shatter them, then stop

in the end,  $\text{VCdim}(\mathcal{H})$  is the maximum  $n$  for which I can shatter  $n$  samples

example @ blackboard: shattering samples on a plane using straight lines. (that means linear functions over  $\mathbb{R}^2$ )

in this case  $I = \mathbb{R}^2$  and  $\mathcal{H} = \{w_1x_1 + w_2x_2 + b\}$

## the VC dimension

examples:

linear functions in  $d$  dimensions:  $\text{VCdim}(\mathcal{H}) = d + 1$

polynomials of degree  $k$  in  $d$  dimensions:  $\text{VCdim}(\mathcal{H}) = \binom{d+k-1}{k}$

Gaussians with arbitrary  $\Sigma$ :  $\text{VCdim}(\mathcal{H})$  is infinite!

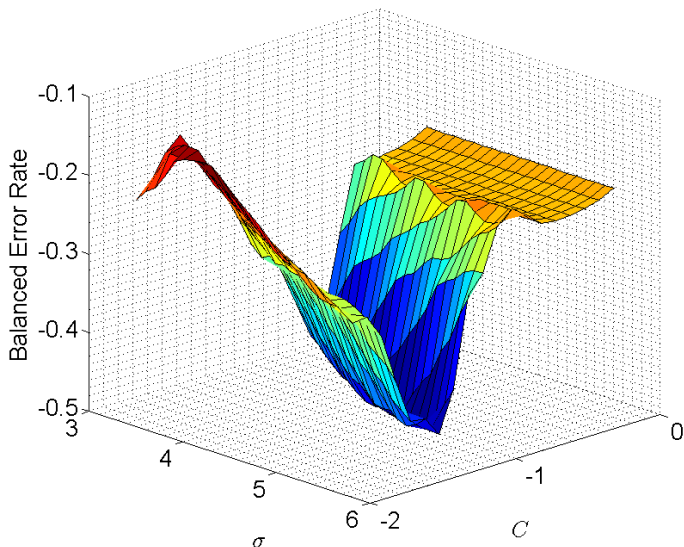
$\text{VCdim}(\mathcal{H})$  is *not* the number of parameters in the function although it is related to it

(actually, the more parameters, usually, the higher the chance of overfitting - remark coming from experience)

SRM (structural risk minimisation) is concerned with finding the best set of hyperparameters. it involves the concept of a *bound* on the error, which depends on the VC dimension

interestingly, there is a much simpler way of estimating hyperparameters: *cross-validation* (switch to Andrew W. Moore's slides)

grid search for  $C, \sigma$ , ball-catching problem



look at the matlab .FIG and rotate it  
(ever heard about Rosenbrock's banana function? well, this is *worse*.)

## grid search and cross-validation...

are actually *paramount* in (applied) machine learning

decide a method, you find the hyperparameters by... testing them all.  
(that is what *grid search* is, really)

that is, for each  $n$ -uple of hyperparameters, evaluate the generalisation  
via cross-validation

this gives you the hyperparameters of the "best" model...

...which you then re-use online. and this is the final test.

# the ML practitioner's cookbook

## **obtain data and make it usable (i.e., build $\mathcal{D}$ )**

build a setup and test it; design an experiment; perform it and gather raw data; look at data and decide filtering/subsampling/preprocessing; look at data again! is it all ok? looks reasonable?

## **choose a method**

is it going to be k-NN? SVM? linear regression? if it is a kernel method, choose a kernel. **is ml really needed at all?**

## **select model / find best hyperparameter(s)**

establish a measure of performance (error); use grid-search to find best (hyper)parameter(s); generalisation must be ensured via, e.g., l.o.o. or cross-validation

## **test!**

and I mean: test on really new data, maybe online, and maybe during a demo. this is the real thing...

## recommended readings

Christopher J. C. Burges, *A Tutorial on Support Vector Machines for Pattern Recognition*, 1998 (in particular, Section 2)

Andrew W. Moore's tutorial on cross-validation (2001)

Andrew W. Moore's tutorial on the VC dimension (2001, passim)