(D1) General remarks, practical considerations

- accuracy is not enough
- confusion matrix
- Receiver Operating Characteristics

accuracy is not enough (I)

observed test set performance of classifiers after training:

assume: results were obtained for imbalanced data

with 75% from class 0, only 25% from class I („*positives*“)

	classifier A	classifier B
accuracy:	**75%**	70%
acc. class 0:	**100%**	70%
acc. class I:	0%	**70%**

problems: – strongly **biased classification tasks**
(e.g. *screening* for rare diseases)

– different bias in training / test / **real world ?**

– Note: two types of error can have
very different consequences (e.g. in medical diagnosis)

confusion matrix

confusion matrix		ground truth (true class membership)	
		1 (positive)	0 (negative)
classifier output (predicted class membership)	1	TP = # of true pos.	FP = # of false pos.
	0	FN = # of false neg.	TN =# of true neg.

true pos. rate $tpr = \frac{TP}{TP + FN}$

true neg. rate $tnr = \frac{TN}{TN + FP}$

false pos. rate $fpr = \frac{FP}{FP + TN}$

false neg. rate $fnr = \frac{FN}{TP + FN}$

Remarks

- rates are not independent, e.g. $tnr + fpr = 1$
- many more quantities derived from confusion matrix, e.g.

$$precision = \frac{TP}{TP + FP}$$

- even more *names* used in different disciplines, e.g.

$tpr = recall = hit\ rate = sensitivity$
$tnr = 1 - fpr = specificity$
$precision = positive\ predictive\ value\ (PPV)$
...

Examples of **performance measures** for imbalanced data sets:

balanced accuracy: *BAC = (tpr + tnr) / 2*

F1-measure:

$$F_1 = \frac{2TP}{2TP + FP + FN}$$

Matthews correlation coefficient:

$$MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

and many more …

Advice: don't even try to remember all these definitions! ☺

Receiver Operating Characteristics

frequently, classification is based on the evaluation of a

discriminative function of input vectors:

assignment to $\begin{cases} \text{class } 0 & \text{if } \; g(\mathbf{x}) \leq 0 \\ \text{class } 1 & \text{if } \; g(\mathbf{x}) > 0 \end{cases}$

e.g.: perceptron with $S = \text{sign}(\mathbf{w} \cdot \mathbf{x})$

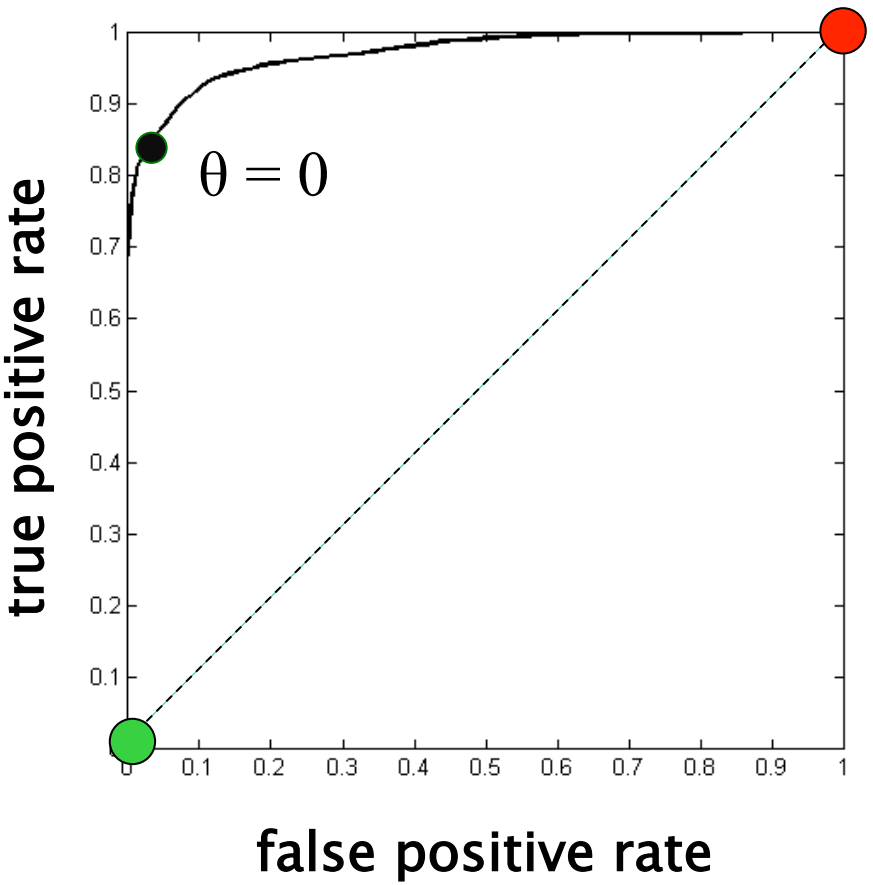
introduce classification bias:

assignment to $\begin{cases} \text{class } 0 & \text{if } \; g(\mathbf{x}) \leq \theta \\ \text{class } 1 & \text{if } \; g(\mathbf{x}) > \theta \end{cases}$

e.g.: perceptron with $S = \text{sign}(\mathbf{w} \cdot \mathbf{x} - \theta)$

- consider the classifier for all possible values of θ
- determine *tpr, fpr, ...* as functions of θ
- plot *tpr vs.* fpr (eliminating the parameter θ)

Receiver Operating Characteristics



$\theta \to +\infty$

all data assigned to class 0

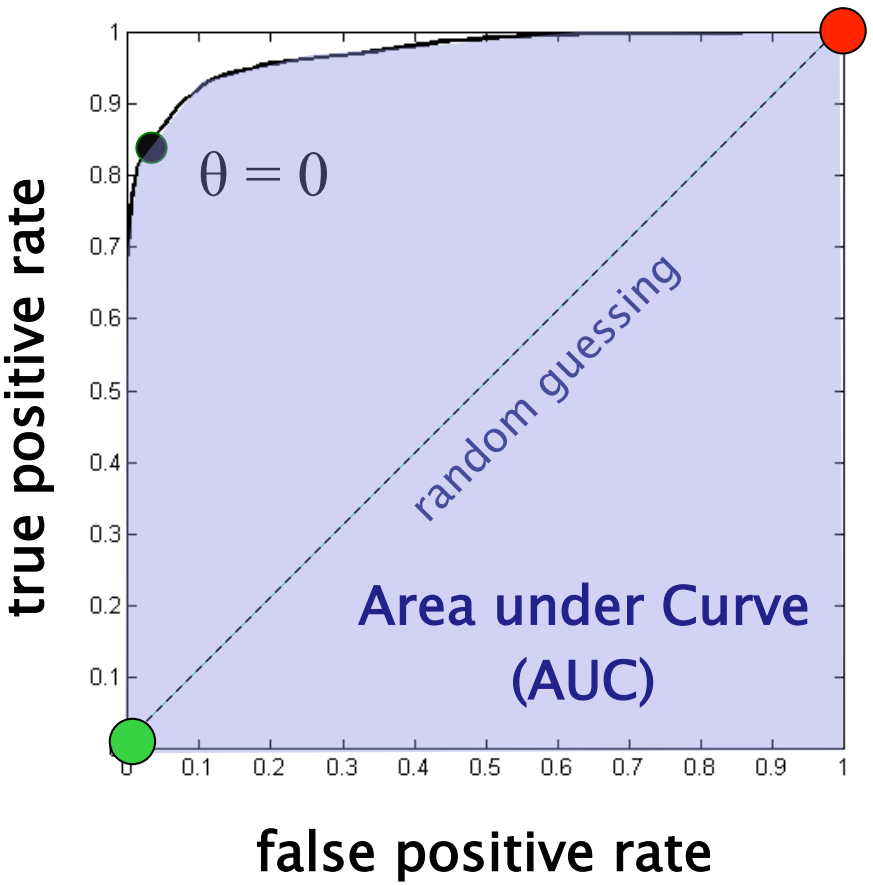
- no false alarms
- no positives detected

$\theta \to -\infty$

all data assigned to class 1

- all true positives detected
- max. number of "false alarms"

Receiver Operating Characteristics



$\theta \to +\infty$

all data assigned to class 0

- no false alarms
- no positives detected

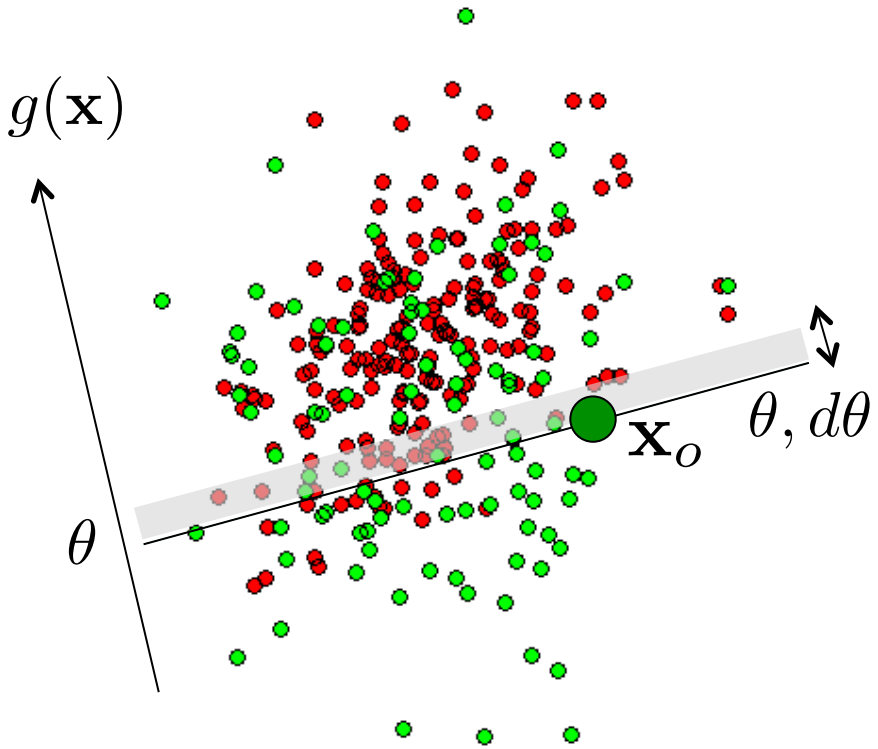
$\theta \to -\infty$

all data assigned to class 1

- all true positives detected
- max. number of "false alarms"

- diagonal corresponds to random guesses with bias
- deviation from random guessing measured by AUC (ROC)

interpretation of AUC (ROC)



select randomly
- one class 0 example $\mathbf{x}_o$
  e.g. with $g(\mathbf{x}_o) = \theta$

- one arbitrary class 1 example $\mathbf{x}_1$

probability that $g(\mathbf{x}_1) \geq g(\mathbf{x}_o)$
is given by $tpr(\theta)$

density of class 0 data at θ
is given by the derivative $\frac{d\,fpr}{d\theta}$

prob. that for a random pair $\{\mathbf{x}_0, \mathbf{x}_1\}$ we have $g(\mathbf{x}_1) \geq g(\mathbf{x}_o)$:

$$\int_{-\infty}^{\infty} tpr(\theta)\,\frac{d\,fpr}{d\theta}\,d\theta \;=\; \int_{0}^{1} tpr\,d\,fpr \;=\; AUC$$

AUC (ROC) = probability for correct order of random pairs with respect to the discriminative function

Remarks

- AUC(ROC) can be used as a quality measure to compare classifiers, e.g. in cross validation
- ROC leaves the choice of a *working point* to the domain expert (e.g. extremal points, $fpr = 1 - tpr$, problem-specific bias, etc.)
- several competing schemes / criteria,

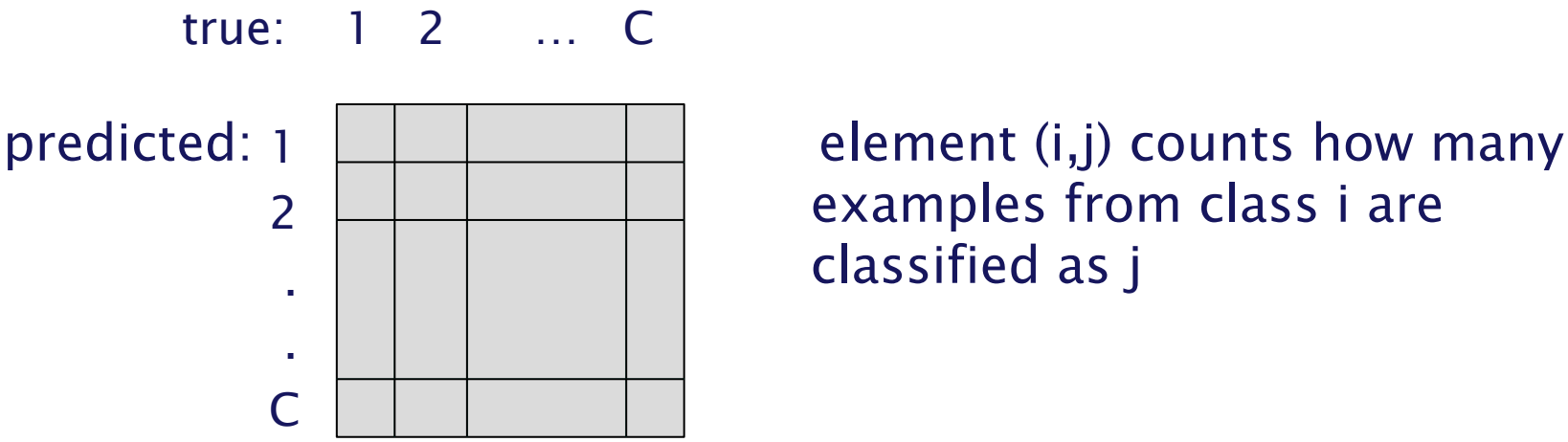
  e.g. **Precision/Recall (PR)**

  claimed to be more appropriate for strong bias (see literature)

  but AUC (PR) without (obvious) statistical interpretation
- see **paper by Fawcett** for an excellent review of ROC

multi-class problems

confusion matrix contains all class-specific accuracies and errors



element (i,j) counts how many examples from class i are classified as j

generalization of ROC etc. is non-trivial

one possibility: define a particular class as "negative"

consider ROC for "one-against-all-other"

many "single quality measures" derived from the confusion matrix suggested in the literature

accuracy is not enough (II)

a machine learning *urban legend* almost true :-)

US military:

- classifier to distinguish US from Russian tanks
- trained on a data set of still images
- nearly **perfect classification** performance (training and also validation / test)
- **complete failure** "in practice"

American tank

Russian tank

university of groningen

accuracy is not enough (II)

to be avoided: *blind* application of ***black box*** machine learning

models should be:

transparent / intuitive / interpretable, ***white box***
e.g.: decision criteria used by the classifier
most important features contributing

- **avoid artif**acts, e.g. due to the hidden data set bias
- gain **better insight** into the data set / problem
- potentially **understand underlying mechanisms**

<u>one</u> suitable framework:
similarity / **distance** based methods
representation / parameterization in terms of **prototypes**

university of groningen

sometimes accuracy is not even the goal

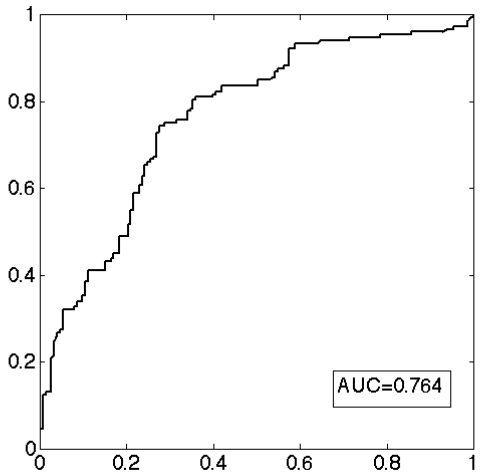
☹ classifier with mediocre performance from small training data sets

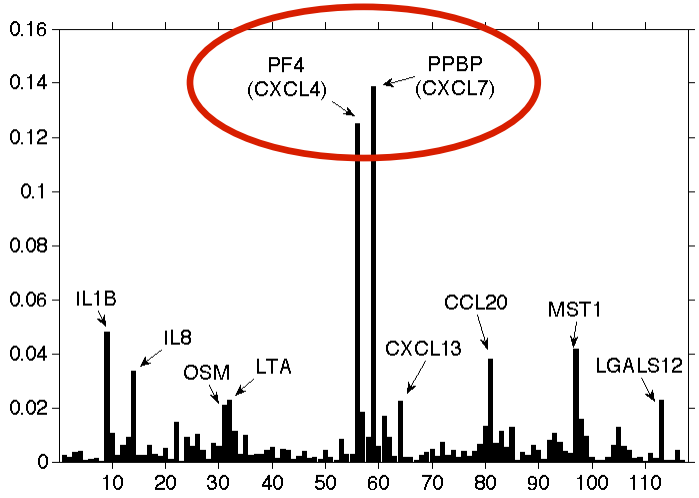
☺ yet: insight into the problem, e.g. most relevant features
in a *white-box* classifier or regression system

example: **bio-marker identification** in medical diagnosis
hints at disease mechanisms
suggests new scientific questions

early Rheumatoid arthritis
vs. resolving inflammation

110 citokine markers
ca. 30 patients only





university of groningen

(D2) validation, over-fitting, bias and variance, regularization

model selection

key problems in supervised learning

model selection: LVQ, Neural Networks, labelled SOM,… ?
how many prototypes, neurons, which kernel ?

data representation: coding, normalization, transformation, … ?

algorithm, (hyper-) parameters:
which training prescription ?
how many training epochs, which learning rate… ?

---

consequences of mismatched **model complexity**:

bias / variance dilemma

overfitting

bias / variance dilemma

model selection:
choice of network architecture, number of layers, nodes, adaptive parameters, etc.

Occam's Razor:

Among different ideas which explain the same observation, accept the simplest

---

The bias-variance dilemma: (illustrative example)

– polynomial regression w.r.t. to an unknown target function: $f(x) = x^3$

– data set $\mathbb{D} = \{ x_i, y_i \}_{i=1}^{P}$ with $-1 \leq x_i \leq 1$ (equidistant)

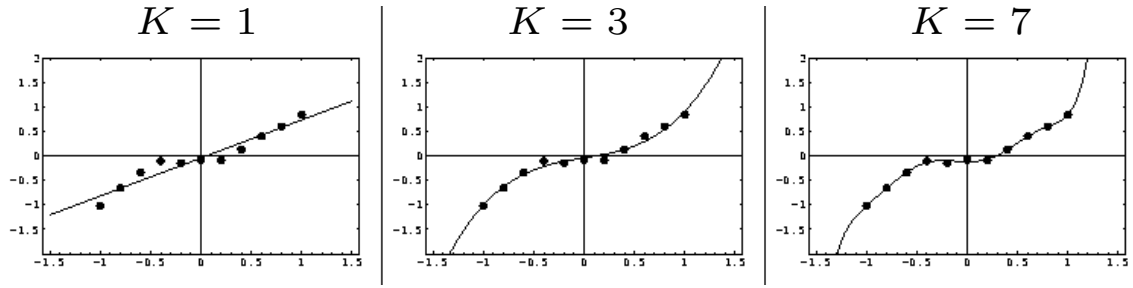
– noisy *labels* $y_i = f(x_i) + \eta_i$ with random η_i uniform from $[-a, a]$ $<y_i> = f(x_i)$

hypothesis: polynomial of degree K, $f_H(x) = \sum_{j=0}^{K} a_j \, x^j$

coefficients a_j obtained from least square fit w.r.t. $\mathbb{D}$
(by minimizing $\sum_i (f_H(x_i) - y_i)^2$))

88

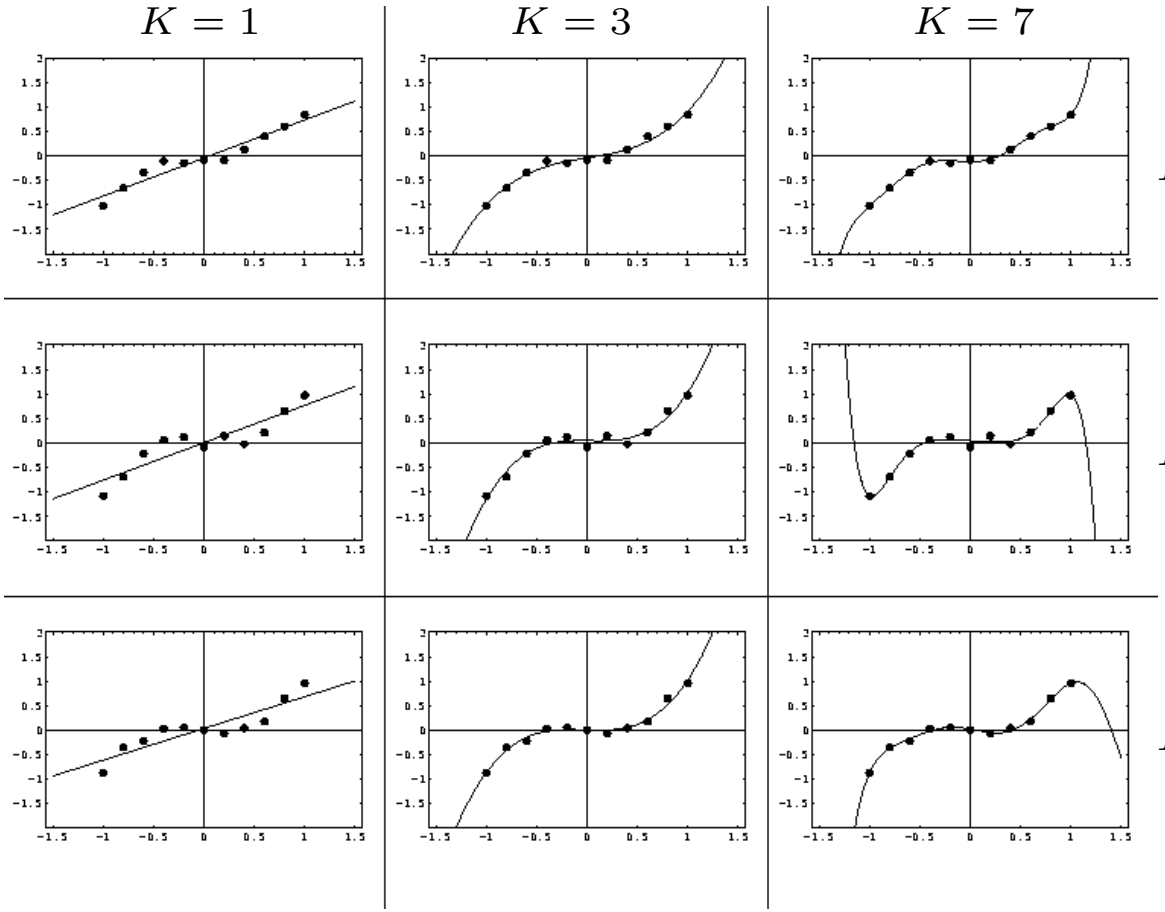
bias / variance dilemma



$\mathbb{D}_1$

one data set, three different models

bias / variance dilemma



example results for three indep. random data sets

K too small: result is very *robust* , almost independent of $\mathbb{D}$
large deviations $(f_H(x_i) - y_i))^2/2$ (*training error*)

K too large: result varies strongly from data set to data set
small *training error,* poor prediction (inter- and extrapolation)

bias / variance dilemma

the bias / variance dilemma (qualitative discussion)

competing aims in training:

low **bias** = small **systematic deviation** from the "true solution"
on *average over many data sets* of the same size

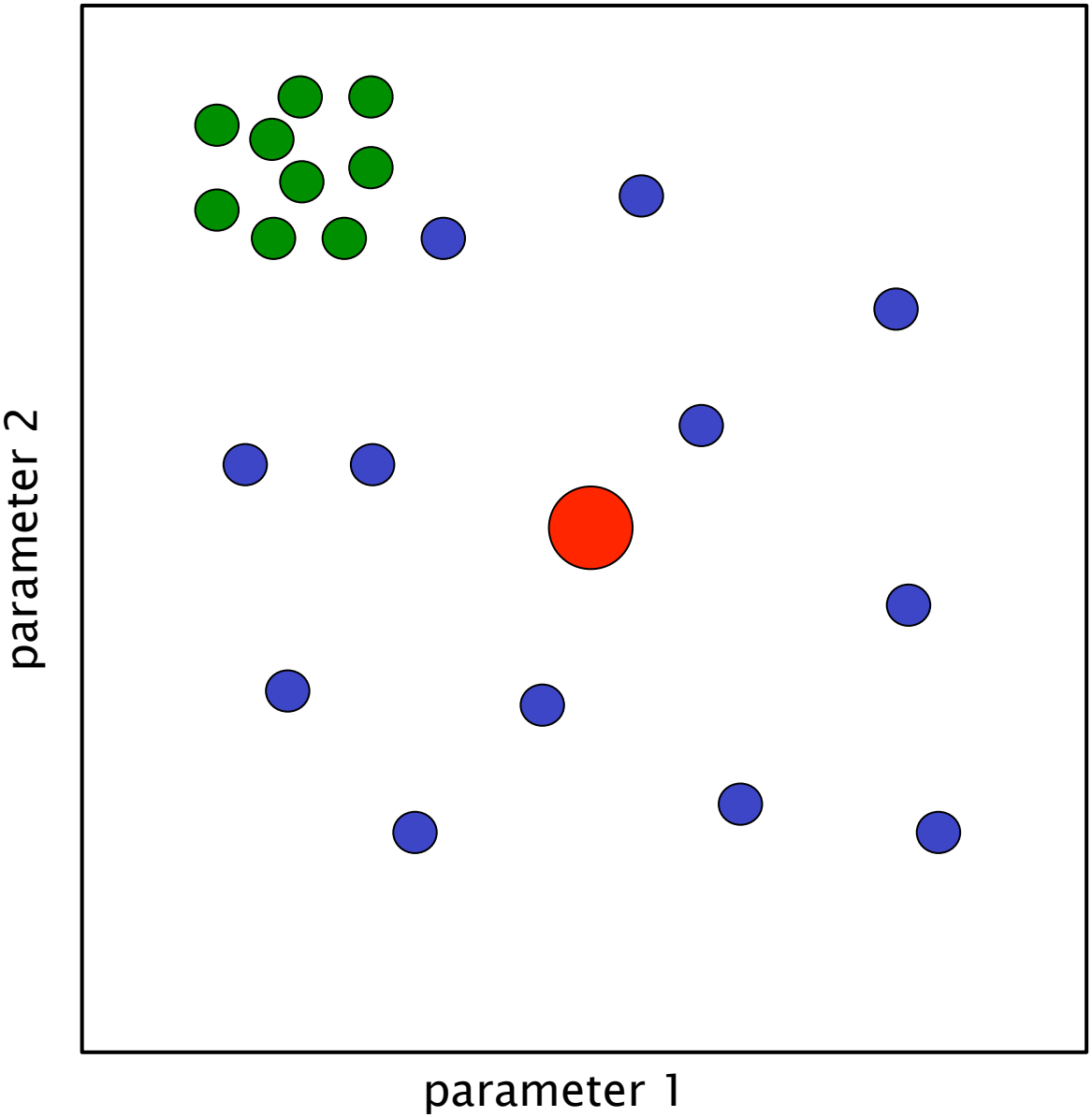
low **variance** = weak dependence on the actual training set,
robustness of the hypothesis

dilemma:

small variance: simple model, *under-fitting* → large bias

small bias: complex model, **over-fitting** → large variance

bias / variance dilemma



target, optimal solution

fit of model A
low bias
high variance

fit of model B
high bias
low variance

general argument:

$\langle \ldots \rangle_{\mathbb{D}} =$ average over random realizations of $\mathbb{D}$

expected quadratic deviation of $f_H(x)$ and $f(x)$ for an arbitrary x:

$$\left\langle \left(f_H(x) - f(x) \right)^2 \right\rangle_{\mathbb{D}}$$ (shorthand: f_H, f w/o arguments)

$$= \left\langle f_H^2 \right\rangle_{\mathbb{D}} - 2 f \left\langle f_h \right\rangle_{\mathbb{D}} + f^2$$

general argument:

$\langle \ldots \rangle_{\mathbb{D}} =$ average over random realizations of $\mathbb{D}$

expected quadratic deviation of $f_H(x)$ and $f(x)$ for an arbitrary x:

$$\left\langle \left(f_H(x) - f(x) \right)^2 \right\rangle_{\mathbb{D}}$$ (shorthand: f_H, f w/o arguments)

$$= \left\langle f_H^2 \right\rangle_{\mathbb{D}} - 2 f \left\langle f_h \right\rangle_{\mathbb{D}} + f^2$$

$$= \underbrace{\left\langle f_H \right\rangle_{\mathbb{D}}^2}_{*} - 2 f \left\langle f_h \right\rangle_{\mathbb{D}} + f^2 + \left\langle f_H^2 \right\rangle_{\mathbb{D}} \underbrace{- 2 \left\langle f_H \right\rangle_{\mathbb{D}}^2 + \left\langle f_H \right\rangle_{\mathbb{D}}^2}_{*}$$ (* add up to 0)

general argument:

$\langle \ldots \rangle_{\mathbb{D}} =$ average over random realizations of $\mathbb{D}$

expected quadratic deviation of $f_H(x)$ and $f(x)$ for an arbitrary x:

$$\left\langle \left(f_H(x) - f(x) \right)^2 \right\rangle_{\mathbb{D}}$$ (shorthand: f_H, f w/o arguments)

$$= \left\langle f_H^2 \right\rangle_{\mathbb{D}} - 2 f \left\langle f_h \right\rangle_{\mathbb{D}} + f^2$$

$$= \underbrace{\left\langle f_H \right\rangle_{\mathbb{D}}^2}_{*} - 2 f \left\langle f_h \right\rangle_{\mathbb{D}} + f^2 + \left\langle f_H^2 \right\rangle_{\mathbb{D}} \underbrace{- 2 \left\langle f_H \right\rangle_{\mathbb{D}}^2 + \left\langle f_H \right\rangle_{\mathbb{D}}^2}_{*}$$ (* add up to 0)

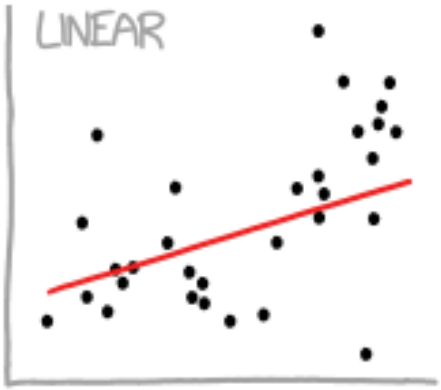
$$= \underbrace{\left(\left\langle f_H \right\rangle_{\mathbb{D}} - f \right)^2}_{bias^2} + \underbrace{\left\langle \left(f_H - \left\langle f_H \right\rangle_{\mathbb{D}} \right)^2 \right\rangle_{\mathbb{D}}}_{variance}$$

bias: systematic deviation of the (mean) fit from the target function

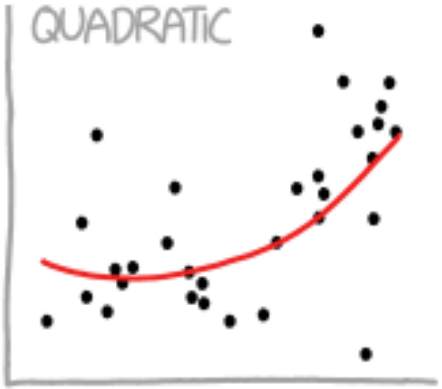
variance: fluctuations with respect to the realization of the data set

(above definitions are for one x, $\int dx \ldots \rightarrow$ gives integrated bias/variance)

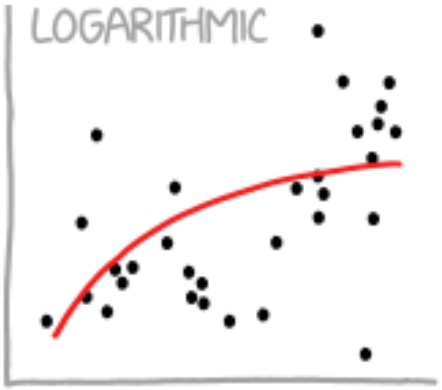
CURVE-FITTING METHODS
AND THE MESSAGES THEY SEND



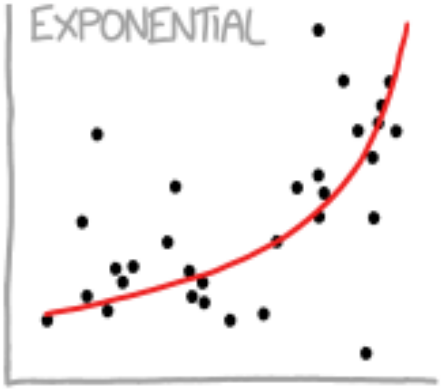
"HEY, I DID A REGRESSION."



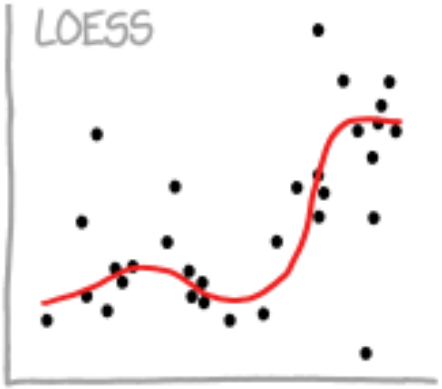
"I WANTED A CURVED LINE, SO I MADE ONE WITH MATH."



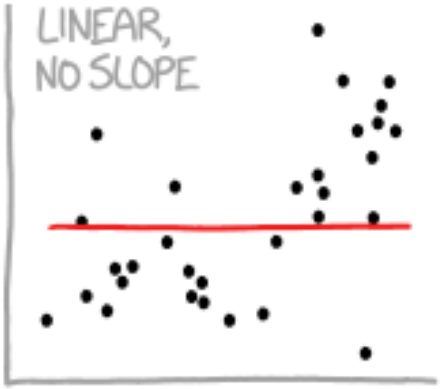
"LOOK, IT'S TAPERING OFF!"



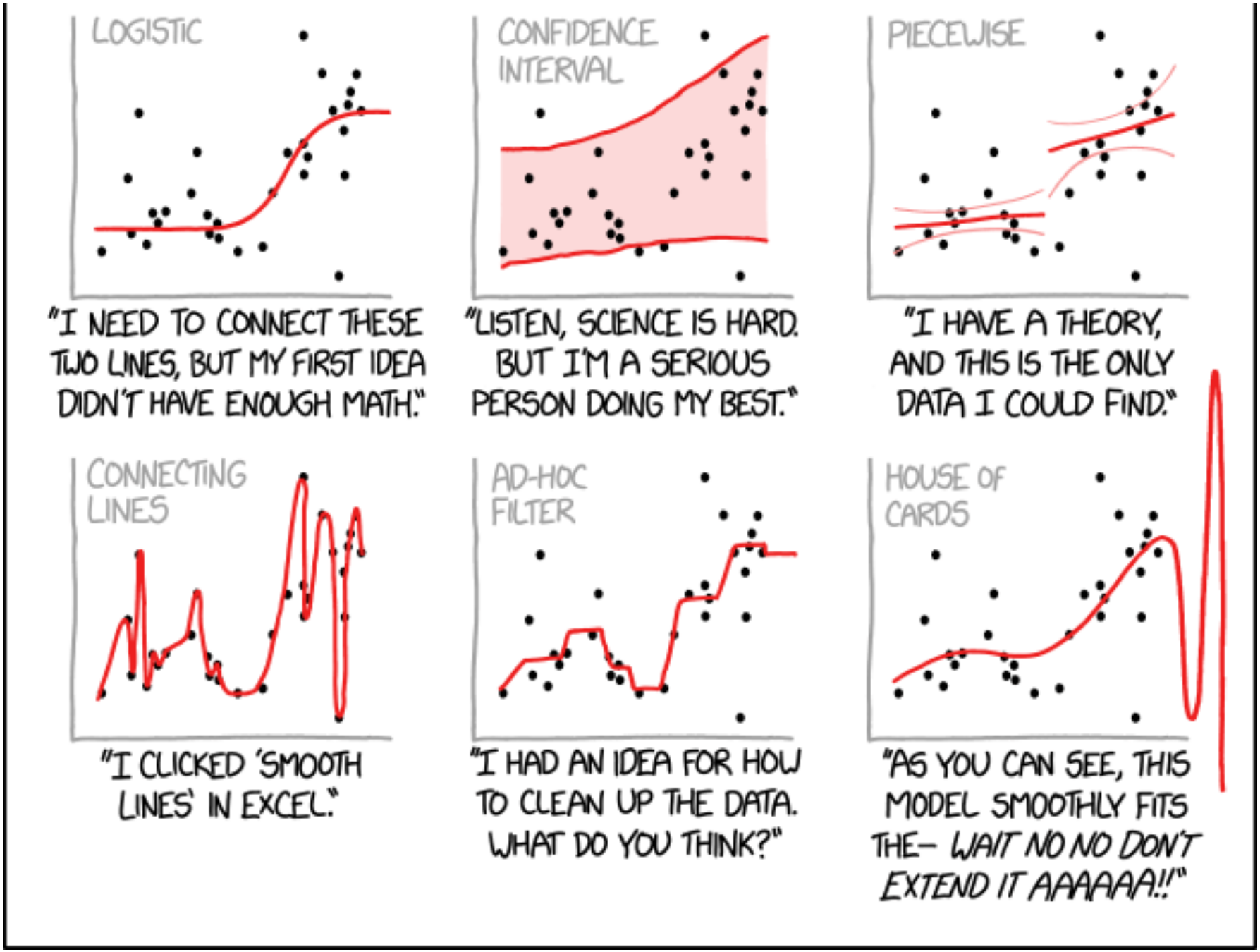
"LOOK, IT'S GROWING UNCONTROLLABLY!"



"I'M SOPHISTICATED, NOT LIKE THOSE BUMBLING POLYNOMIAL PEOPLE."



"I'M MAKING A SCATTER PLOT BUT I DON'T WANT TO."


LOGISTIC
"I NEED TO CONNECT THESE TWO LINES, BUT MY FIRST IDEA DIDN'T HAVE ENOUGH MATH."
CONFIDENCE INTERVAL
"LISTEN, SCIENCE IS HARD. BUT I'M A SERIOUS PERSON DOING MY BEST."
PIECEWISE
"I HAVE A THEORY, AND THIS IS THE ONLY DATA I COULD FIND."
CONNECTING LINES
"I CLICKED 'SMOOTH LINES' IN EXCEL."
AD-HOC FILTER
"I HAD AN IDEA FOR HOW TO CLEAN UP THE DATA. WHAT DO YOU THINK?"
HOUSE OF CARDS
"AS YOU CAN SEE, THIS MODEL SMOOTHLY FITS THE- WAIT NO NO DON'T EXTEND IT AAAAAA!!"

model selection

key problems in supervised learning

model selection: LVQ, Neural Networks, labelled SOM,...?
how many prototypes, neurons, which kernel... ?

data representation: coding, normalization, transformation, ... ?

algorithm, (hyper-) parameters:
which training prescription ?
how many training epochs, which learning rate... ?

---

training: based on performance with respect to training data
aim : low error with respect to new data, **generalization**

how can we test the generalization ability?

Validation procedures

basic idea: split available data $D = \left\{ \left\{ \xi^{\mu}, S^{\mu} \right\} \right\}_{\mu=1}^{P}$

(randomly) into disjoint sets:

$$D_{training} = \left\{ \left\{ \xi^{\mu}, S^{\mu} \right\} \right\}_{\mu=1}^{Q} \qquad D_{test} = \left\{ \left\{ \xi^{\mu}, S^{\mu} \right\} \right\}_{\mu=Q+1}^{P}$$

→ estimate of test error E_{test} (e.g. number of misclassifications)

→ comparison/choice of different models, algorithms, settings...

→ prediction of performance with respect to novel data (?)

validation

problems:

- **lack of data**
  can we afford to *waste* example data *only* for validation ?

- **representative results ?**
  lucky / unlucky set composition can give misleading outcome !

- **variation of results ?**
  how safe is the prediction ? error bars of the estimates ?

n-fold cross-validation

example strategy: " n-fold cross-validation "

split data $D = \left\{ \left\{ \xi^{\mu}, S^{\mu} \right\} \right\}_{\mu=1}^{P}$ (randomly) into **n** disjoint sets

$$D = \bigcup_{i=1}^{n} D^{(i)} \qquad D_{train}^{(i)} = D \setminus D^{(i)} \qquad D_{test}^{(i)} = D^{(i)}$$

all data | training data (i) | test data (i)

- repeat training n times
- calculate average training / test errors (and variances)

- repeat cross-validation for different models, parameters, etc.
- select the best system with respect to test errors
  (model, number of units, learning rate, …)

cross-validation

remarks:

- **which n** in n-fold cross-validation ?

  larger n → larger fraction of D used in each training run
  → more estimates of E_{test} / smaller test sets
  → higher computational effort

  extreme case: n = P
  use all but one examples for training, test on single example,
  repeat P times "leave-one-out estimate"

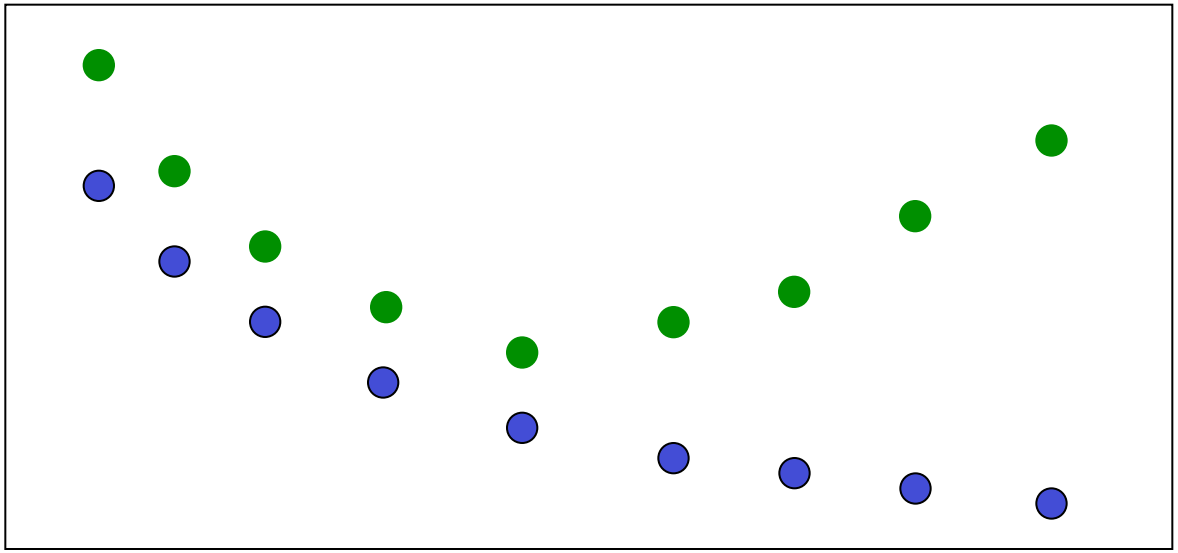
- **statistics ?**

  n results are not statistically independent
  highly overlapping training sets!
  → difficult to estimate *variance*

over-fitting

test / training errors (e.g. observed in cross-validation)

vs. complexity of the model (e.g. # of prototypes, neurons, …)



in general:

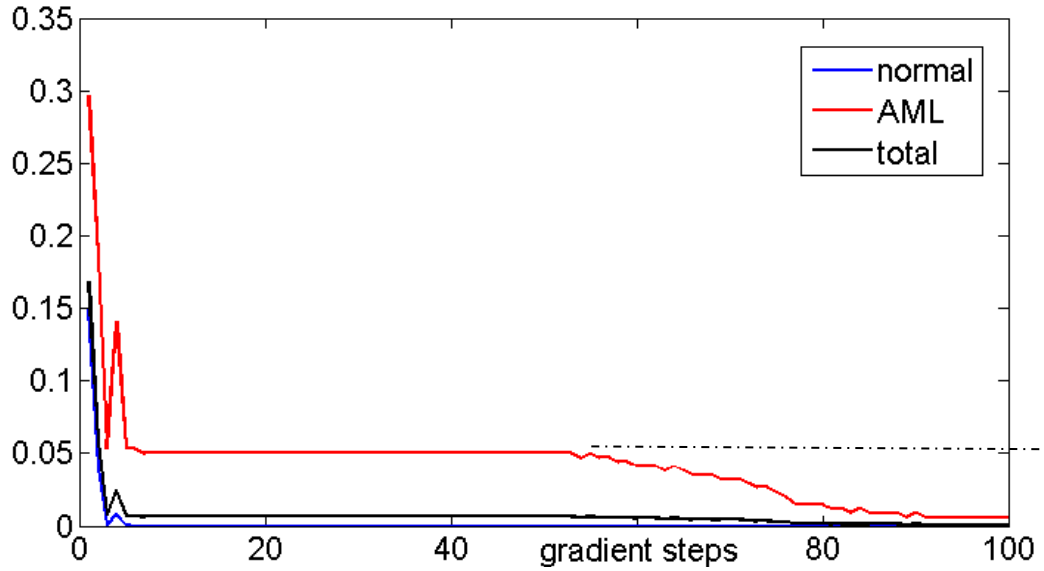
$E_{train} < E_{test}$

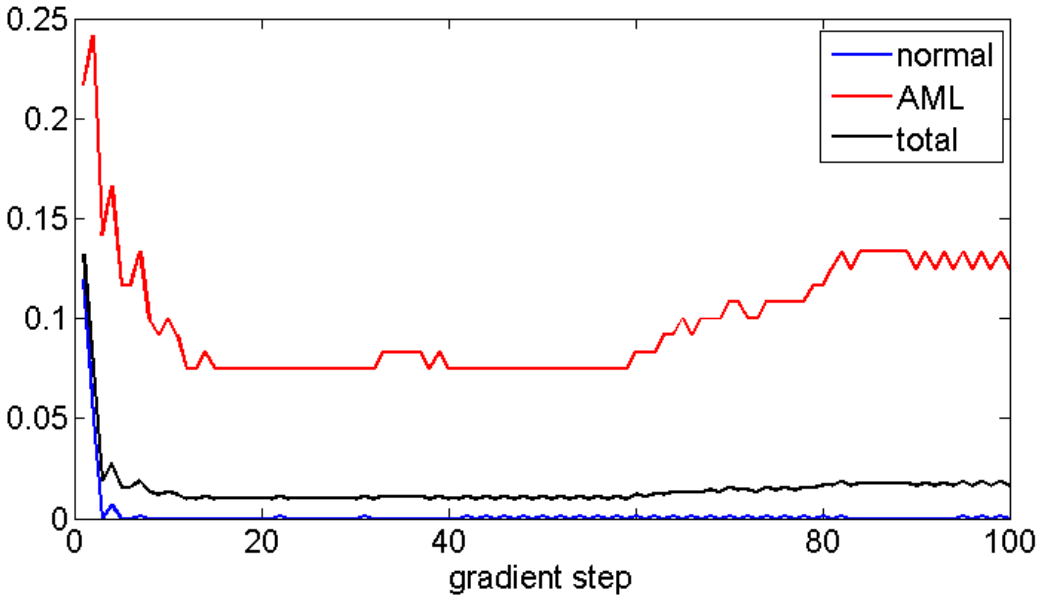
- expect: better classification (of D_{train}) with increasing complexity

- classifier / regression can become over-specific to training set !

  over-fitting (low training, high test error)

over-fitting





Leukemia diagnosis,
[Biehl, Bunte, Schneider PlosOne 2013]

overfitting for large training 'time'

i.e. *too thorough* optimization of the cost function

over-fitting

Remarks

Critical assumption: data set represents the "real world"

Validation procedures can overfit !!!

example: selection of parameters or features

based on E_{test} in cross-validation

- does depend on the entire data set D
- unclear performance with respect to new data

strategies:
- second level of validation (extra data?)
- base parameter and feature selection on training set performance, if possible

over-fitting

Randomized validation

even n-fold CV can be subject to un/lucky set composition

repeat n-CV many times over randomized splits

or (simpler):

split data randomly into X% test, (100-X)% training data

repeat and average over many random splits

Regularization

algorithm (hyper-) parameters can control ***effective complexity***

i.e. the degree to which the training error can be minimized

e.g. restricted magnitude of weights

limited number of training epochs

drop-out (training of randomized subsets of parameters)

...many more...

regularization

competing aims of training:
low bias – good approximation on average over all possible data sets
low variance – robustness with respect to particular realizations of the data set

---

Regression with neural networks

feedforward networks with continuous activation are **universal approximators**
... can implement arbitrary non-linear (smooth) functions

example formulation: **Cybenko's Theorem**

a *soft-committee machine* with output $\sigma(\boldsymbol{\xi}) = \sum_{j=1}^{K} g(\mathbf{w}_j \cdot \boldsymbol{\xi} - \theta_j),$

sigmoidal activation functions $g(x)$ (e.g. $g(x) = \mathrm{erf}(x)$ or $g(x) = \tanh(x)$)

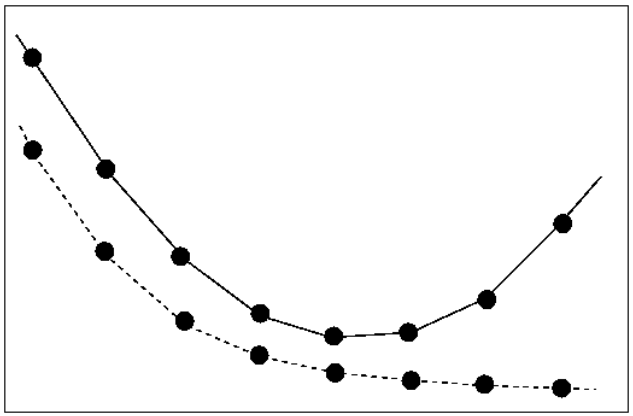
adaptive weight vectors $\mathbf{w}_j \in I\!R^N$, and adaptive threshold values $\theta_j \in I\!R$

can approximate every (continuous, differentiable) function $(I\!R^N \to I\!R)$

to arbitrary precision

. . . **provided** the number of *hidden units* K is **large enough**

frequently observed behavior (schematically)



over-fitting:

an overly complex adaptive system

can yield a low training error E

but very poor generalization ability

Model Selection (here in terms of neural networks)

choice of appropriate network complexity (e.g. size of the hidden layer)

requires **validation,** i.e. the estimation of ε_g based on test data (see later section)

idealzed training scenario:

– large amount of **noise-free data** available

– large test set $\rightarrow$ reliable validation $\rightarrow$ determine **perfectly matching network** by comparing, e.g., the performance of different network sizes K

– minimization of training error $E \rightarrow$ **good generalization**

problems:

– **costly validation schemes** (computation, extra data required)

– minimization of E: local minima, other **numerical difficulties**

– **noisy data sets** $\rightarrow$ (strict) minimization of E ...
is not required / may be disadvantageous (**overfitting**)

some possible strategies/methods to adapt the network architecture (number of hidden units) to the complexity of the data during training:

- **constructive approaches**
  start with a simple network, add units or layers
  example: *tiling like learning in multilayer networks*

- **pruning strategies**
  start with a complex network, remove units, layers, or single weights
  examples: *pruning, optimal brain damage, optimal brain surgery,...*

– monitor generalization ability by means of validation schemes

– select network complexity in order to avoid **underfitting / overfitting**

Regularization compromise:

– use a powerful (fixed) architecture, e.g. with large K

– restrict the number of degrees of freedom effectively

control of the *effective complexity* by some parameter in the training procedure

Example 1) **weight decay**

restrict the magnitude of weights in the network

units with activity $\sigma = g(\sum_j w_j \xi_j) \approx g(0) + g'(0)\,(\mathbf{w} \cdot \boldsymbol{\xi})$ for $|\,\mathbf{w}\,| \approx 0$

small weights effectively *linearize* the activation function

consider **modified cost function** $\widehat{E} = E + \lambda \frac{1}{2} \sum_j w_j^2$ (with $\lambda > 0$)

Regularization compromise:

– use a powerful (fixed) architecture, e.g. with large K

– restrict the number of degrees of freedom effectively

control of the *effective complexity* by some parameter in the training procedure

Example 1) **weight decay**

restrict the magnitude of weights in the network

units with activity $\sigma = g(\sum_j w_j \xi_j) \approx g(0) + g'(0)\,(\mathbf{w} \cdot \boldsymbol{\xi})$ for $|\,\mathbf{w}\,| \approx 0$

small weights effectively *linearize* the activation function

consider **modified cost function** $\widehat{E} = E + \lambda \frac{1}{2} \sum_j w_j^2$ (with $\lambda > 0$)

minimize $\widehat{E}$ instead of E:

$\mathbf{w}(t+1) = \mathbf{w}(t) - \eta \boldsymbol{\nabla}\, \widehat{E} = \mathbf{w}(t) - \eta \boldsymbol{\nabla}\, \widehat{E} - \bar{\lambda}\mathbf{w} = \mathbf{w}(t)\,(1-\bar{\lambda}) - \eta\, \boldsymbol{\nabla}\, E$

2 steps: (1) gradient descent w.r.t. E, (2) weight decay by factor $(1-\bar{\lambda})\ 1 \qquad (\bar{\lambda} = \eta\lambda)$

weight decay effectively **smoothens** the network output

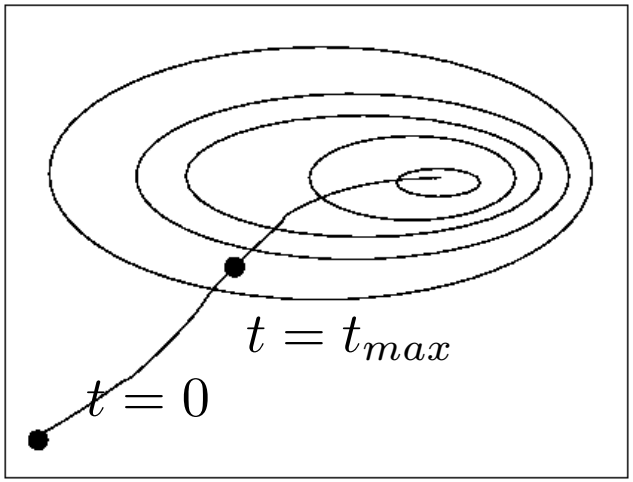
and restricts the complexity that can be achieved

Example 2 **early stopping**

– restricts the number of learning steps (e.g. gradient descent up to t_{max})

– does not allow for the thorough minimization of E

– for initialization **w** ≈ 0, *early stopping* effectively realizes *weight decay*



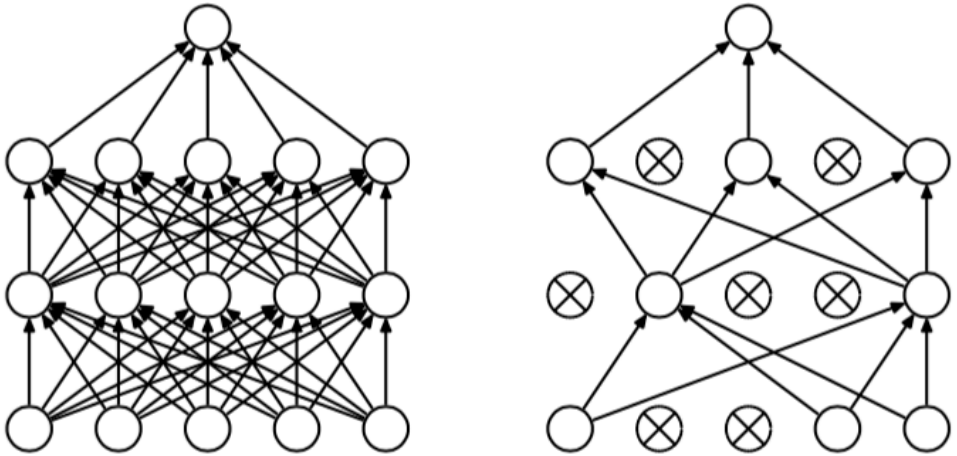
practical problems:

– determination of t_{max} by validation schemes

– result depends strongly on initialization

example 3: dropout

effective restriction of the network complexity in the training:



(a) Standard Neural Net (b) After applying dropout.

Dropout Neural Net Model. **Left**: A standard neural net with 2 hidden layers. **Right**: An example of a thinned net produced by applying dropout to the network on the left. Crossed units have been dropped.

In every update step, only a randomly selected subset of units is trained

illustration from:
Srivastava, Hinton et al.
J. Machine Learning Res.
15: 1929-1958 (2014)

Training of randomized, simpler sub-networks

Working phase uses full network $\approx$ ensemble of many simple systems

concluding remarks

It's a zoo ...

e.g. classification:

- K-NN, LVQ, supervised SOM/NG etc.
- Discriminant Analysis, Logistic Regression
- Perceptron, Support Vector Machine
- Multilayered Neural Networks (many types)
- Decision Trees, Forests of Trees
- Gaussian Process Classifiers, ...

regression
...

clustering:
...

reinforcement learning: ...

dimension reduction: ...

novelty detection: ...

...

in the zoo...

Think first:

- what is the goal ? performance only, deeper insight, ...
- is it realistic ? (machine learning ≠miracles)
- literature: has it (or something similar) been done before ?

Inspect the data:

- representation, normalization, transformations...
- unsupervised analysis: correlations, clusters, structures ?
- employ visualization techniques

in the zoo...

Start simple:

- e.g. K-NN classifier, linear regression, PCA, k-means
- compare to baseline algorithms
- increase level of sophistication if necessary / promising
- the latest trend is not necessarily the best for your problem

Accuracy is not enough:

- try to obtain insight
- employ interpretable models/systems, vsualization
- proper testing/validation with respect to suitable measures
- beware of artefacts, biased data ...

Some challenges (keywords)

Imbalanced data
Incomplete data
Noisy data ...

Functional data
Privileged information
heterogeneous / mixed data
non-vectorial data (graphs, relational data) ...

Transfer learning
Lifelong learning
Representation learning
Interpretable models
Learning causal relations ...

References

A few textbooks:

T. Kohonen.
Self-Organizing Maps.
Berlin: Springer, 3rd Edition, 2000.

C.M. Bishop.
Pattern Recognition and Machine Learning.
Cambridge, UK: Cambridge University Press, 2007.

T. Hastie, R. Tibshirani, J. Friedman.
The Elements of Statistical Learning: Data Mining, Inference, and Prediction. 2nd ed. New York: Springer, 2009.

I. Goodfellow, Y. Bengio, A. Courville
Deep Learning.
Cambridge MA: MIT Press, 2016.