Tfy-99.275 lecture 8

Feature extraction and selection, evaluation and assessment of methods

Feature Extraction

- reduces the amount of raw data while maintaining (or enhancing) the vital information for the pattern recognition task at hand
- features can range from low-level features (typically measurements, e.g., numerical values like signal amplitudes) to high-level features (e.g., 'symptom' feature for classification into 'diseases' classes)

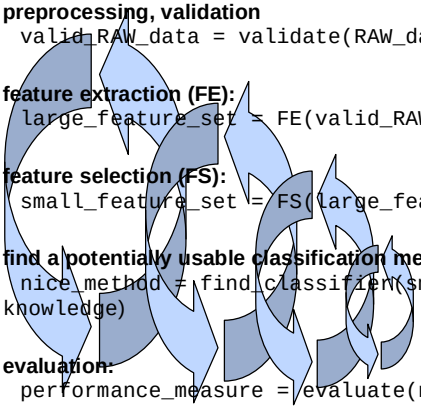
Feature Selection

- reduce the set of data even further; select those features that are most relevant to the pattern recognition task at hand(for example, using principal component analysis)

Stages

- preprocessing, validation
  `valid_RAW_data = validate(RAW_data)`
- feature extraction (FE):
  `large_feature_set = FE(valid_RAW_data)` %transformation from raw data
- feature selection (FS):
  `small_feature_set = FS(large_feature_set)` %selection from large_feature_set
- find a potentially usable classification method:
  `nice_method = find_classifier(small_feature_set,known_outputs,background knowledge)`
- evaluation:
  `performance_measure = evaluate(nice_method, test_data)`
- implementation of 'nicest' method

Stages, in practice, many feedback loops



- **preprocessing, validation**
  `valid_RAW_data = validate(RAW_data)`
- **feature extraction (FE):**
  `large_feature_set = FE(valid_RAW_data)` %transformation from raw data
- **feature selection (FS):**
  `small_feature_set = FS(large_feature_set)` %selection from large_feature_set
- **find a potentially usable classification method:**
  `nice_method = find_classifier(small_feature_set,known_outputs,background knowledge)`
- **evaluation:**
  `performance_measure = evaluate(nice_method, test_data)`
- **implementation of 'nicest' method**

Feature Selection, dimension reduction

- transform/summarise raw data in features descriptive for the task at hand
- adding more features to use in a classifier in theory does not decrease classifier performance (as long as class-conditional pdfs are known)
- in practice added features may degrade performance if the ratio (nr of data cases/nr of features) is low
- curse of dimensionality -> more features means many more parameters to estimate for a classifier
- exact 'minimal acceptable' relationships (nr of data cases/nr of features) differ from classifier to classifier
- general guideline: nr of data cases/nr of features > 10
- the more complex the classifier the higher the ratio should be
- moreover: a limited/salient set of features simplifies classifier implementation, results in higher speed, less memory reqs etc.
- Obviously, a too drastic reduction of nr of features may lead to a loss of performance as well

Feature Vectors

- comprise all features used to describe a pattern; it is a reduced-dimensional representation of that pattern
- the set of all features that could be used to describe a pattern is limited to those actually stated in the feature vector.

Main Questions

1. which techniques to use to generate features?
2. how to select a subset from the features that each technique generates?

1. Selected features are 'best' only by some application-specific criterion, therefore techniques for generation of features tend not to be very portable from one processing problem to another.
2. Typically we make an initial set of features on the basis of theoretical/physiological knowledge, or practical experience, or consultation of literature

Dimension reduction

1 - explorative methods ('data mining')
given a high-dimensional input set we'd like to explore what is going on
(e.g., are there some features correlated to each other? clusters maybe?)
often employs projection onto 2-3 dimensional for visualisation purposes

2 - performance directed methods
try to optimise the set of features so that we get the 'best' possible classification -> performance criterion

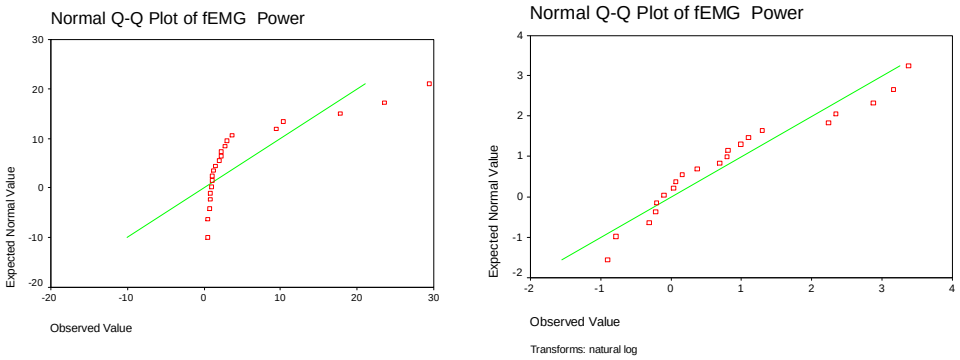
Basic Exploration

- Basics first: check basic statistical descriptors: mean, sd, range and boxplots, histograms, normal probability plots, intervariable scatter plots

  Matlab:
  nanmax(),nanmean(),nanmedian(),nanmin(),nanstd(), boxplot(), hist(), (histfit()), normplot(), gplotmatrix()

  SPSS: using the ėxamine ’command
  Analyze->Descriptives->Examine (stats + plots)
  (boxplots can also be obtained from the ‘Graphs->Boxplot ’menu)
  Graphs->Scatter

- This gives an impression of distributions of data, possible outliers (artefacts?), normality of data, visual impression of differences in values between different groups, visual impression of correlations between variables



Q-Q plot of EMG (electrical muscle activity) power.

Left: raw data: clearly far away from normally distributed

Right: the same plot after the data was transformed via logarithm transform, close to normal.

Explorative methods, Principal Factor Analysis (PFA)

- typical situation; we measure a lot of different variables, but there is quite some overlap in what they represent
- e.g.,
  - a decathlon with 10 events (variables) of which 5 represent 'speed' abilities of the athlete and 5 represent 'strength'
  - results of a student of 6 course exams at school, 3 representing 'quantitative abilities' and 3 for 'verbal abilities'
- The idea is that groups of variables are representative of common underlying factors ('strength', 'language ability'). Variables within such a group (governed by the same factor) will have high intercorrelation, variables from different groups will have small intercorrelation.
- Each variable can then be written as a linear combination of underlying common factors (plus some independent random 'variable specific' variance generated by a 'unique' factor). The factor coefficients are called 'loadings'.

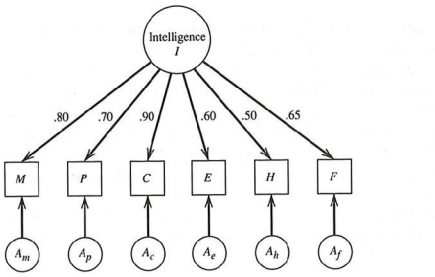


Figure 5.1 Relationship between grades and intelligence.

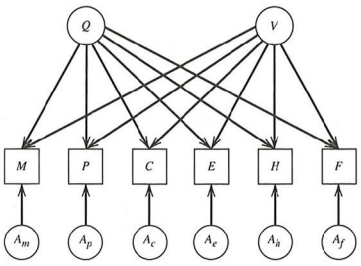


Figure 5.2 Two-factor model.

m = maths, p=physics, c=chemistry, e=english, h=history, f=french

PFA (2)

- The data should have a bivariate normal distribution for each pair of variables, and observations should be independent. The computed estimates are based on the assumption that all unique factors are uncorrelated with each other and with the common factors.
- Several different (iterative methods) exist to obtain factor estimates (SPSS: Analyze->Data Reduction->Factor; Matlab: factoran() (matlab uses the the 'maximum likelihood 'iterative algorihtm, SPSS allows you to choose from 7 different methods)
- Solutions are not unique. Often the initial suggestion of factors doesn't make any 'real-world 'sense and factors need to be 'rotated 'so that each variable is represented by a few (preferably only one) factor. Factor rotation is done by specifying options in the respective SPSS/Matlab calls.
- It is not always guaranteed that we can find some sensible underlying factor solution, and we typically need help from a domain-expert to interpret results.
- Obviously not all data is suitable for this 'factor grouping.' A useful heuristic measure is the KMO statistic (SPSS only) that can be checked (should be > 0.8ish)

KMO measure	suitability for PFA
>0.9	marvelous
0.8+	meritorious
0.7+	middling
0.6+	mediocre
0.5+	miserable

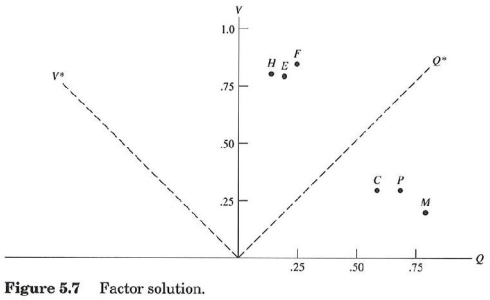


Figure 5.7 Factor solution.

PFA and Principal Component Analysis (PCA)

- If we are not interested in studying underlying factors but just want to reduce the number of variables we can use PCA.
- It is in effect PFA, but now, instead of

PFA:
Each variable can then be written as a linear combination of underlying common factors (plus some independent random variable specific 'variance generated by a unique 'factor). The factor coefficients are called loadings.'

we get, PCA:
Each variable can then be written as a linear combination of underlying common components. The component coefficients are called scores.'

- The components form an orthogonal basis for the data, the first one is an axis that when you project data on it contains as much variance as possible. The succeeding perpendicular ones cover less and less of the variance.

PFA & PCA (2)

- Let's say as an example we measure 20 features available from an experiment : Heartrate, Weight, Length, ..., BloodPressure. We would like to find a new set of features, considerably smaller than 20, that covers most of the information present in the 20 orginal variables.
- What we are going to do is to rewrite these variables as a different set of variables; F1 to F20), with the following equations

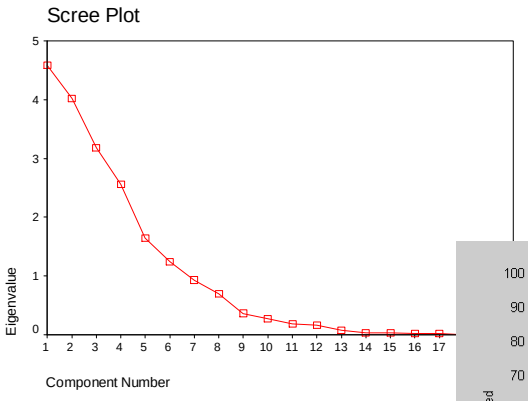
```
HR     = a1*F1 + b1*F2 + c1*F3 + d1*F4 + ..... t1*F20 + Error1
Weight = a2*F1 + b2*F2 + c2*F3 + d2*F4 + ..... t2*F20 + Error2
Length = a3*F1 + b3*F2 + c3*F3 + d3*F4 + ..... t3*F20 + Error3
......
......
BloodP = a20*F1 + b20*F2 + c20*F3 + d20*F4 + ..... t20*F20 + Error20
```

and try to solve F1..F20 out of those.

The "ErrorN" terms are the unique variances. If we do PCA we assume that all those ErrorN terms can be set to 0, and we thus assume that all the variance in the whole variable set is 'common', ie, there is no variance unique to a single variable. In PFA ErrorN can be unequal to 0.

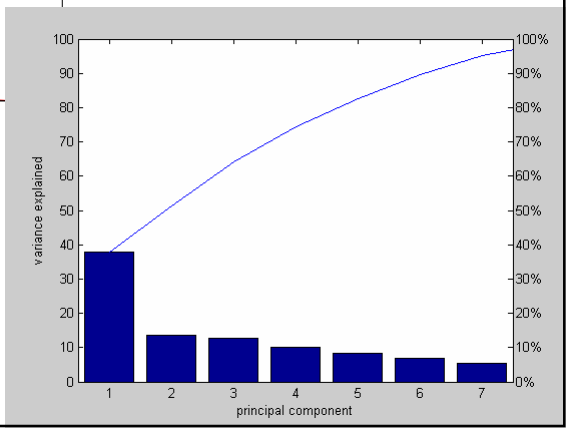
PCA

- There are as many components as there are original variables. Usually only a few of the most important ones are retained (explaining e.g., 90% of the total variance, or retain those having an eigenvalue > 1, examination of Scree plot)
- SPSS: Analyze->Data Reduction->Factor (same as PFA), choose PCA as extraction method; Matlab: princomp(), pareto() (for getting the Scree plot)
- Again the KMO statistic (in SPSS) is a useful heuristic for checking the suitability of the data in the first place.
- Although PFA and PCA are very similar they are used for different purposes; PFA mainly for studying underlying 'structures' in the data generation process, PCA mainly for pure data dimensionality reduction



scree plot according to SPSS

scree plot according to Matlab



PCA continued

- We can get factors F1 to F20 by using the matrix of correlations between all the variables. From there we can see which variables overlap in what they measure. (e.g., 'weight' and 'height' are likely to correlate considerably - taller people tend to weight more than shorter people).
- For any reasonable size of matrix keeping track of correlations becomes rather complex. Try to find factors that account for underlying correlations. (So, in our example, instead of talking about the correlated 'height' and 'weight', we could maybe instead talk about one factor; "bigness"). There are several techniques to get the factors out of the matrix. Typically, one calculates the Eigenvectors of the correlation matrix. In the end we get the expressions for F1 .... F20, that are Eigenvectors of the correlation matrix, that we call principal factors (or, principal components)

- We will find as solution

F1 = α1 * HR + β1 * Weight + γ1 * Length.... τ1 * BloodP

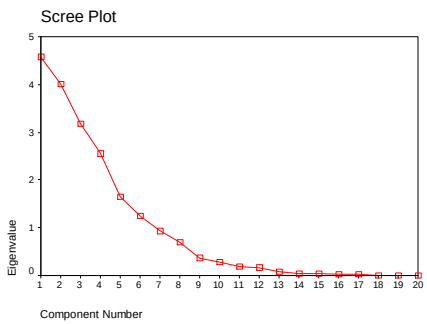
....

F20 = α20 * HR + β20 * Weight + γ20 * Length .... τ20 * BloodP

(One of them maybe something like Fx = 0.0001 HR + 0.61 Weight + 0.25 Length + ... -0.002 BloodP, this one is then the one associated with 'bigness' since only the coefficients for Weight and Length are very different from 0.)

- We do not want to keep all 20 F's though, we are only going to retain those that account for most of the variance ('information'). It might be that F1 already explains 66% of all the variance in the original set. F2 adds an additional 20%, and additional 10% by F3 and the remaining 4% is covered by F4 to F30. In that case we can decide to take F1 to F3 to continue with since they already can for 96% explain the correlation between the variables, and it is a nice compact set of only 3 new variables. The rest 4% we leave unexplained then.
- Selection is often done by examining the so-called Scree plot, which plots the Eigenvalues against the factor/component nr.

- Commonly used strategies are to retain all components that have an Eigenvalue bigger than a predefined value (eg, 1), or retain components up to the point where the 'bend' in the Scree plot occurs.



Interpretation of components

- A practical problem is how to interpret the obtained components, what practical meaning does for example
  "F1 = -0.3 * HR + 0.4 * Weight + -0.02 Length + ... 0.1 BloodP "have?
- PCA tends to generate components that are a mixed bag of everything, and do not have an intuitive meaning
  (but we may not even be interested in finding a meaning, we may be just interested in variable set reduction).
- PFA and rotation of the factors can help to generate more understandable factors; usually we need a domain expert though to help interpret them

Explorative Methods; Multi-dimensional scaling methods

- The idea is to project the n-dimensional data onto a lower dimensional space so that the interdistance matrix in the original space is preserved as faithfully as possible in the projected space. ("faithfullness "is defined using a so-called stress function).
- Several methods exist, a popular method is the Sammon mapping.
  Results in a non-linear transformation of data which is not given explicitly (so mapping has to be recalculated if a new measurement is to be added)
- Sammon mapping available as part of the Matlab "SOMToolbox "package (and is often used in conjunction with use of Kohonen's SOM neural net)
- cmdscale() in Matlab is a general procedure for processing "interdistance " matrices using "classical "multi-dimensional scaling

Performance Directed 'Methods for Feature Selection

- Y is the original set of *n* features, *d* is the desired number of features in a selected subset X, with $X \subseteq Y$.
- We have some feature selection criterion, J(X). The higher J is the better subset X we have.
- A possible criterion could be J(X) = (1 –Ex), where Ex is the error obtained with the classifier using X as input.
  This makes J dependent on the choice of classifier as well as on train and test data sets (and the definition of 'error 'of the classifier).
- An exhaustive search to find the best set would require examination of $\binom{n}{d}$ possibilities. This becomes highly impractical for even moderate values of *n.* (e.g., if *n*=15 and we want *d*=5, we would already need to do 3003 examinations)
- No non-exhaustive search can be guaranteed to produce the optimal subset

Feature selection methods

Method	Property	Comments
exhaustive search	evaluates all possible subsets	Guaranteed to find the optimal set, not feasible in practice
branch-and-bound search	evaluates a fraction of the subsets (if dimensions are small) using knowledge about upper bounds of errors	Can find optimal set, but works only if J increases monotonically when we add features (in practice rarely true).
ANN node pruning	examine weights after training; remove nodes with small weight connections	Elegant way of having feature selection and classification performance evaluation in one 'integrated box.' Requires lots of training data.
best individual features	evaluate all features individually, select *d* best ones	Computationally simple and intuitive, but not likely to lead to optimal solution. Very useful in early phases though
sequential forward selection (SFS)	select the best single feature and add one feature at a time which in combination with the already selected features maximises J	Computationally attractive but had disadvantage that once a feature has been added it cannot be removed anymore ('nesting')

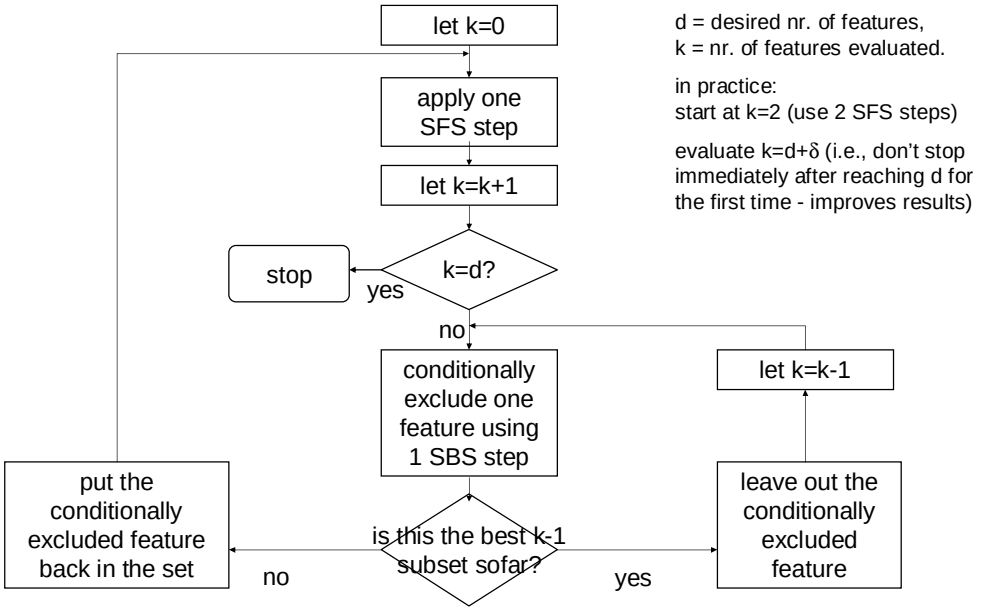
Feature Selection Methods continued

Method	Property	Comments
sequential backward selection (SBS)	start with all *n* features and selectively delete one feature at a time	Nesting; once a feature is deleted if cannot be brought back. Requires more computation than SFS
plus *l* 'take away *r* 'selection	first enlarge the subset by adding *l* best features using SFS then remove *r* using SBS	Avoids nesting. need to define *l* and *r* (*l* > *r*)
sequential forward floating search (SFFS) and sequential backward floating search (SBFS)	generalisation of plus *l* 'take away *r* '. *l* and *r* are determined automatically and updated dynamically	Allows to correct earlier 'mistakes 'by backtracking. Close to optimal solution at affordable computational cost
genetic algorithms (GA)	feature subsets represented as binary strings. a 'fit 'population of strings is aimed for using crossover and mutations	Search space encompasses all subsets (of all sizes). Requires tuning of parameters. For < (20-30) dim problems as good as SFFS, for higher dimensions performance decreases.

Feature selection in practice

- We have the following phases
  - subset definition
    - use subset to train classifier and test performance -> J
  - redefine subset
    - use new subset to train classifier and test performance -> J
  - etc
- apart from the ANN approach this calls for two distinct pieces of software; some logic to (re)define sets and some 'performance evaluation 'procedure
- to get some good estimation of the best subset we don't need a too fancy classifier in practice, simple linear discriminant, logistic regression, or k-nn classifiers are already useful

SFFS algorithm



Comparison of methods

- So, we have a wide choice of features that we can extract, and we have a wide choice of methods to process them.
- Typically, non-trivial biomedical engineering problems are then also approached by many different research groups each using their own favourite methods on the same problem.
- Question: How do we compare those methods objectively, and decide which one is 'best'?

Evaluation and Assessment

- Often, processing and classification systems take many variables as their input, and emit one variable as output (e.g., a system that uses 17 variables to come to one yes/no output indicating a suggestion of cancer).
- The output of signal processing and interpretation methods can in that case be seen as a composite variable (or, index) : we would like to find out the performance of that variable.

Error rate

- An error is here a misclassification. If all errors are of equal importance, a single-error rate can be calculated that summarises the overall performance of a classifier.

$$\text{error rate} = \frac{\text{number of errors}}{\text{number of cases}}$$

Different Error types

- In many applications, a single-error rate is not the most appropriate measure to use.
- For example, diagnosing someone as healthy when one has a life-threatening disease (a false-negative decision) is typically far more serious than the opposite - diagnosing someone as ill when he is in fact healthy (a false positive)
- To lay out different errors one uses usually a *confusion matrix*

Example Confusion Matrix for two classes

	true class	
predicted class	1	2
1	23	10
2	5	123

Two-class classification problems

	Class Positive (C+)	Class Negative (C-)
Prediction Positive (R+)	True Positives (TP)	False Positives (FP)
Prediction Negative (R-)	False Negatives (FN)	True Negatives (TN)

Ratios derived from confusion matrices

- In biomedical applications we often use frequency ratios to come to quantities like specificity and sensitivity.
- E.g., an automatic alarm in the ICU may have a high *sensitivity* in diagnosing breathing problems (correctly classifying problems), but may have poor *specificity* if it also generates many false alarms.

Measures of Classification Performance

- sensitivity: TP/C+
- specificity: TN/C-
- positive predictive value: TP/R+
- negative predictive value: TN/R-
- accuracy: (TP+TN)/((C+)+(C-))

- note: if both types of errors (false positives and false negatives) are not treated equally, a more detailed breakdown of other error rates is needed

Classification performance example

	patient in reality has disease	patient in reality is healthy	
system estimates that patient has disease	true positives (correctly detected) 23	false positives (false alarms) 10	positive prediction value 23/(23+10) = 0.70
system estimates that patient is healthy	false negatives (failure to detect) 5	true negatives (correctly "kept quiet") 123	negative prediction value 123/(5+123) = 0.96
	sensitivity 23/(23+5) = 0.82	specificity 123/(123+10) = 0.92	accuracy (23+123)/ (23+5+10+123) = 0.91

Example Confusion Matrix for three classes

	true class		
predicted class	1	2	3
1	50	0	0
2	0	45	5
3	0	2	45

seperate definitions for sensitivity, specificity etc are used in more-than-2 class classification problems

Costs

- one can use *misclassification costs* to deal with the fact that some errors are more important than others by weighing each type of error appropriately.
- Errors are thus transformed into costs, and one typically wants to minimise this cost function.
- What you actually do is biasing decisions in different directions, as if there were more or fewer cases in a given class

How close to the truth?

- If we use *N* test cases to make an *estimation* of the error rate, how do we know how close this *estimated* error rate is to the *true* error rate (=the error rate when the system is used in practice)
- If *N* is very large (>5000) the estimated error is effectively the true error rate. However, in most cases we don't have that many test cases.

Accuracy of error rate (1)

- Let's say we repeat the testing experiment *K* times, each time with a set of *N* test cases. So, in total *K* x *N* test cases.
- The mean of the K estimated error rates on the *K* sets approaches the true error rate.
- We have to find the standard error (SE) of the error estimate.

Accuracy of error rate (2)

- We can approximate this using the following expression:

$$SE = \sqrt{\frac{E(1-E)}{N}} \qquad [1]$$

- E.g., if we have after testing 100 cases an estimated error rate of .20, the standard error is 0.04.
- So, for that system we can expect that on average the estimated error rate is off about 0.04 from the true error rate.
- If we would use 1000 test cases, the standard error would be 0.01

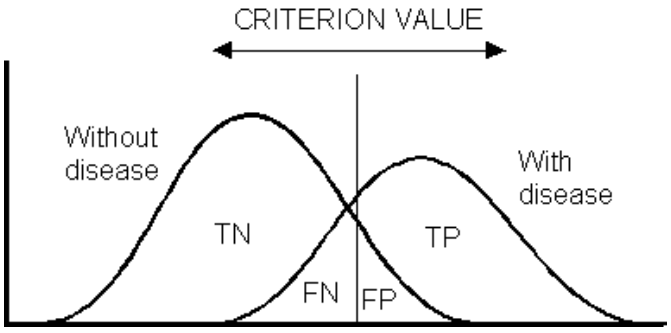
Uncertainties

- Even the 'best' system can still leave substantial uncertainties when it comes to diagnosis, treatment, and patient management decisions.
- These uncertainties need to be measured and stated quantitatively so that both clinician and patient are aware of them.

Tradeoff: Specificity vs Sensitivity

- For most systems it is not possible to realise both a maximum sensitivity and maximum specificity using the same configuration (set of parameters).
- In this case one has to decide which of the two is more important.
- A useful tool for investigating this issue is the ROC (receiver operating characteristic) plot.

Tuning of sensitivity vs. specificity using a cut-off criterion



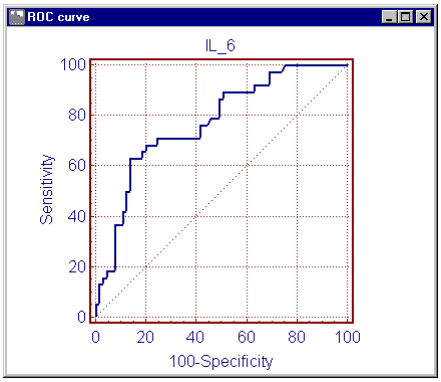
ROC methodology

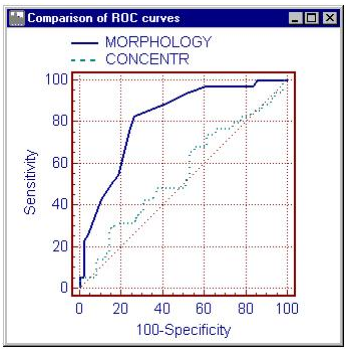
- emerged during WWII in the field of radar detection;
- a radar (Receiver) in a certain configuration has Operating Characteristics like
  - the rate at which it correctly identifies enemy aircraft (sensitivity)
  - the rate at which it correctly identifies things other than enemy aircraft (specificity)
- for one radar configuration we thus have one (sensitivity, specificity) co-ordinate; we can corresponding co-ordinates for all the possible configurations in a graph
- the same idea was brought to medicine in the early 1970s

ROC plot

- plot sensitivity vs. specificity (or rather 1-specificity)
- the area under the plot reflects the accuracy of the system
- meaningful areas range from 0.5 (random classification) to 1.0 perfect classification

ROC plot examples





Applications of ROC plot

- comparison of different systems by comparing their ROC plots and areas
- for developing purposes: use area under curve for training and validation purposes as an alternative to e.g., RMS error
- make decisions about misclassification costs and risks (relative costs of false positive vs false negative outcomes)

Confidence measures

- If a medical practitioner is to perform a patient-care action influenced by the output of certain system it is very important that he has confidence in the system.
- A trained, tested and verified system, does not provide unique solution (the final configuration can be influenced by many factors, incl. number of cases, order of presentation, changes in learning rate, randomisation of weights etc.)
- a repetition of developing/training on the same data is necessary to assess variations resulting from any of those influences

Performance Measurements

- ideally: develop a system and test in real-life immediately (then redevelop and retest in real-life, etc..), this is very expensive and usually not feasible.

- in practice: we have a set of data available with which we can work. Dilemma: do we use it to extract knowledge from (training), or do we use it to test our system on?

Apparent vs. True Error Rate

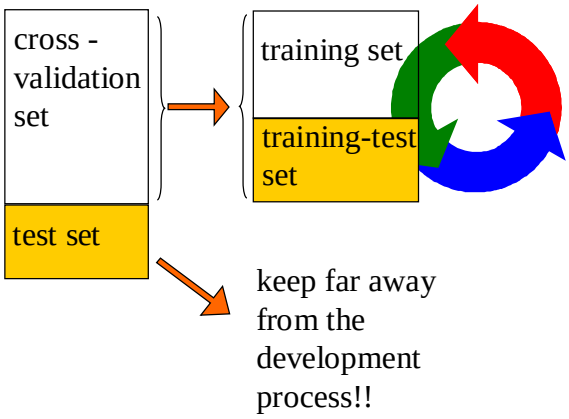
Recall

- The apparent/estimated error rate is the error rate on sample cases that were used to design or build the classifier.
- The true error rate is the error rate that we experience when presenting new cases to the classifier in the ‘real use’ phase.

How many cases to use for testing?

- From the earlier presented equation [1] for determining the standard error of the error rate estimation we can see that the more samples we use for testing, the better our accuracy for estimating the error rate gets. So, we would like to uses as much as possible cases for testing.
- On the other hand; we want to use as much as possible cases for training the system, since that is expected to give us better results.
- We thus have to make a trade-off between the two.

it is common practice to divide the total data set into 2 parts: the *cross-validation* (CV) set (this is what we work with) and the *test set*.
The test set is left absolutely untouched during the entire development process, only when we have finalised our classifier we can use it for performance assessment.



the cross validation set is again split into 2 parts; *training* and *training-test* set.
the patterns that fall into the training and training-test sets are 'circulated'

For example, if we have 100 patterns in our CV set, we can take 90 as training patterns and 10 to have an estimate of the performance of a classifier.

Next, we can take 10 other patterns for testing and the remaining 90 as training patterns, retrain the classifier and test.

Then, take again 10 other patterns to test etc.

After 10 times doing this we would have used all patterns for training exactly once (and 10 estimates for the performance). In this example we speak of 10-fold cross-validation.

In general this is called *k*-fold cross-validation with *k* the number of times we repeat. The choice of *k* is typically dependent on the total number of patterns available (in the extreme case: equal to the number of patterns).

k-fold cross validation results

- the final estimation of the error rate is the average of the *k* error rates obtained.
- Its standard error can again be obtained using eq. 1 in which *N* is the total number of cases (in this example, 100).
  (note: this gives a slightly optimistically biased result for the standard error, since here not all cases are independent [they are used for training *and* testing]. For N > 200 however, the bias is very small)

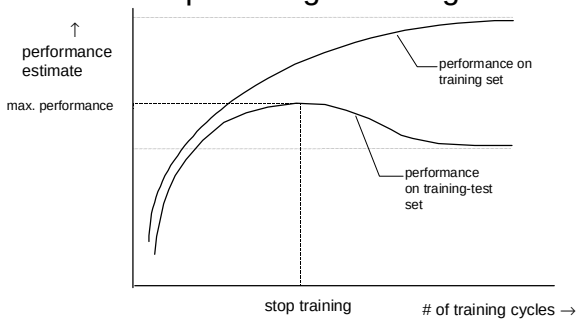
Performance Estimates

based on	**performance**	
training sets	apparent performance	highly biased
training-test sets (k-fold cv)	cv estimate	biased
test set	(approximation of) true performance	unbiased?

we need to measure in **'real' life** have a **real estimate** of the **true performance!!!**

Overspecialisation

- During training, apparent performance may keep increasing while performance on training-test set does not. This indicates 'overspecialisation' or ‘overfitting ’of the classifier (=bad), one should stop training at the right time.



Some Guidelines

- Using resampling, i.e. repeated train-and-test partitions, estimate the error rate. Sometimes also the complexity fit must be estimated; select the classifier with the lowest error rate.
- Apply the identical classification method to *all* sample cases
- For sample size greater than 100, use cross validation, e.g, 10-fold cv is acceptable if we have hundreds of samples
- For sample sizes less then 100, use the leaving-one-out method (i.e., use N-1 cases for training, 1 for testing, and repeat this process N times)
- For fewer than 50 cases we need special methods (e.g., bootstrapping methods) and suitable statistics

More is not always better

- One might be tempted to think that:
  - the more features we can use the better it is
  - a more theoretically powerful method is always better
- This is *not* always the case!

More data is not always better

- As we have seen in the feature extraction section; a proper selection of features is essential. Adding all kinds of extra variables that do not contain any new information makes life harder for a classifier.
- You can see this in the apparent error rate, but it will become especially clear in the true error rate.

Simple vs. Complex methods

- We usually want to make a generalisation on the basis of a training data set, not some kind of look-up table that fits exactly (and only) to the training data.
- A simpler solution will often generalise better than a complex one.

Some pitfalls

- testing on training cases
- training on test cases
- error estimates for small sample sizes are *much* less reliable than those for large samples.
  (note: if we have 10000 cases, but only 2% of them represent a patient with a disease and the others are from healthy subject we also have a small sample size if we want to classify between healthy/disease !!)

Complications in performance assessment

In many biomedical problems we have some serious complications when trying to measure the exact performance of a system, due to

- lack of an accepted ”golden standard” against which we can verify our system
- often subjective (non-numerical) scales are used as measure of patient state

Lack of a golden standard

- Many interpretations of patient data are done using expertise of a given observer (clinician). One clinician will often make subtly different decisions (based on experience, different ”culture” etc) than another.
- In other cases there may not even be a generally accepted quantity that would represent a certain concept (e.g., ”depth of anaesthesia”, ”pain”)
- In such cases it is usually not the best strategy to try to develop a system that exactly reproduces the behaviour of one specific clinician.
- Use several experts to provide ”right” output, measure inter-expert variance and try to develop a system that generates outputs that falls ”within the variance”
- If experts A, B, and C each have labelled the data with desired outputs, we should try to produce a system S that generates output values so that we cannot determine (with statistical tests) whether produced output came from A, B, C, or S.

Subjective scales

- Many 'patient states' are assessed by mapping observations to a subjective scale which is ordinal but cannot be used to perform straightforward arithmetic on.

observers assessment of alertness/sedation (OAA/S) (e,g, during operations)
5- Replies to spoken commands, eyes open, awake
4- Sedated, replies to spoken commands, mild hypnosis
3- Ceases to reply to loud commands, eye lid reflex present
2- No reply to spoken commands, no eye lid reflex
1- No reaction to TOF (train-of-four) stimulation (50mA) with movement
0- No reaction to tetanic stimulation with movement

Ramsay scale for level of sedation (e.g in ICU)
Ramsay sedation scale
1- Anxious or restless or both
2- Cooperative, orientated and tranquil
3- Responding to commands
4- Brisk response to stimulus
5- Sluggish response to stimulus
6- No response to stimulus

Alderete scale to assess recovery after operation

	Score
A. Activity	
1. Ability to move all four extremities	2
2. Ability to move two extremities	1
3. Unable to move any extremity	0
B. Respiration	
1. Ability to deep breathe and cough	2
2. Respiratory effort limited and dyspnea present	
3. No spontaneous respiratory effort evident	0
C. Circulation	
1. Systolic arterial pressure +/- 20mm Hg of pre-sedation level	2
2. SAP +/- 20 to 50mm Hg of pre-sedation level	1
3. Systolic arterial pressure +/- 50 or higher of pre-sedation level	0
D. Level of consciousness	
1. Full alertness with ability to answer questions	2
2. Patient can be aroused by verbal stimuli	1
3. Verbal stimuli fails to elicit responses	0
E. Temperature	
between 35.6 and 37.5	2
between 35 and 35.6	1
less than 35 or greater than 37.5.	0

Total score needs to be at least 8 before patient can be discharged

Assessment of performance in case of ordinal scales

- The only thing we can say that the levels are ordinal, i.e., ordered in some given manner (level N is 'less pain/less sedated/.. 'than level N+1), but we cannot say that a patient with OAA/S level 4 is 'two times more awake 'than one with level 2, or that the difference between level 4 and level 3 is equal to the difference between level 2 and level 1.
- In such cases it is not very useful to examine straightforward correlation coefficients, mean squared errors etc as an assessment of practical performance because the appropriate metrics do not apply.
- To measure the practical performance of monitors in such a case we use statistic methods that concentrate on assessing how well the monitor output values follow changes of the patient state on the scale. For example, if a patient is getting more deeply under anesthesia, OAAS level decreases, the monitor output should follow in the same direction (eg, downgoing), if the patient state is stable the monitor output should stay stable, and if the patient wakes up the monitor output should follow with an increasing output value.
  The criteria of assessment of performance then become consistent behaviour at all times and speed of following the patient state.