DD2437 – Artificial Neural Networks and Deep Architectures (annda)

Lecture 4: **Practical aspects of ANN approaches to pattern recognition problems**

Pawel Herman

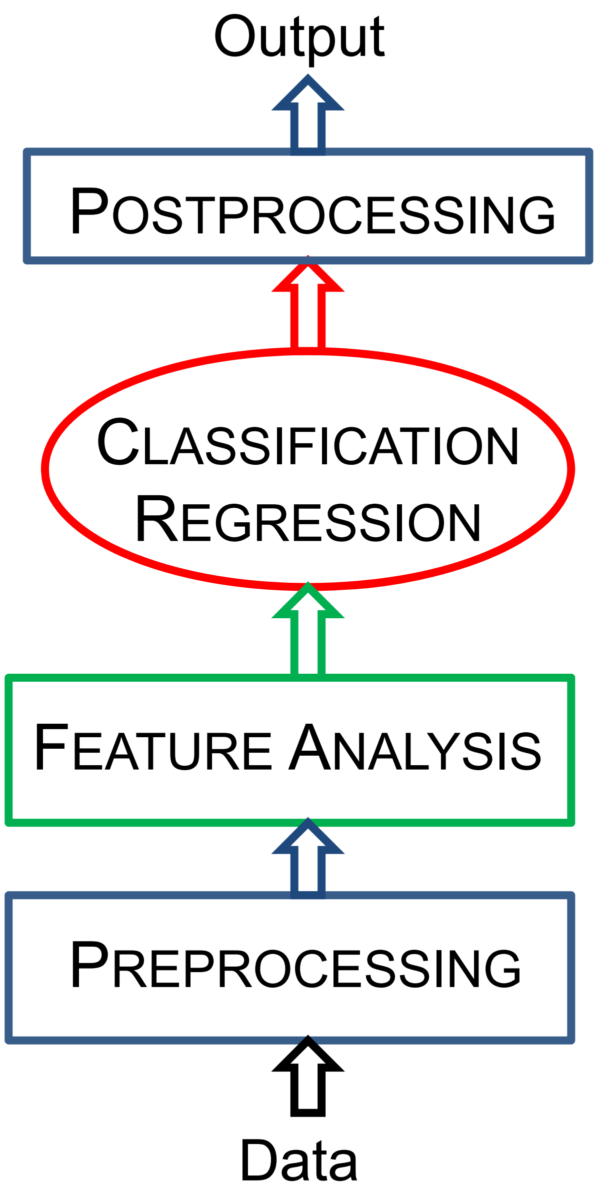
Computational Science and Technology (CST)

KTH Royal Institute of Technology

Lecture overview

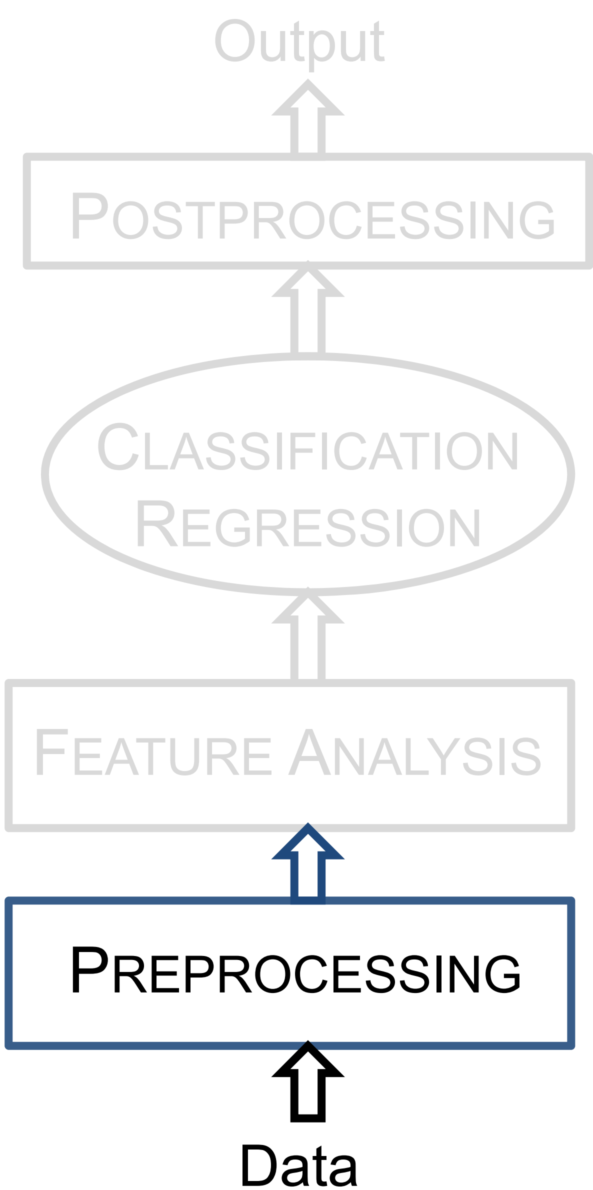
- Data preprocessing and feature extraction
- Error measures
- Parameter optimisation
- Ensemble learning

Pattern recognition pipeline



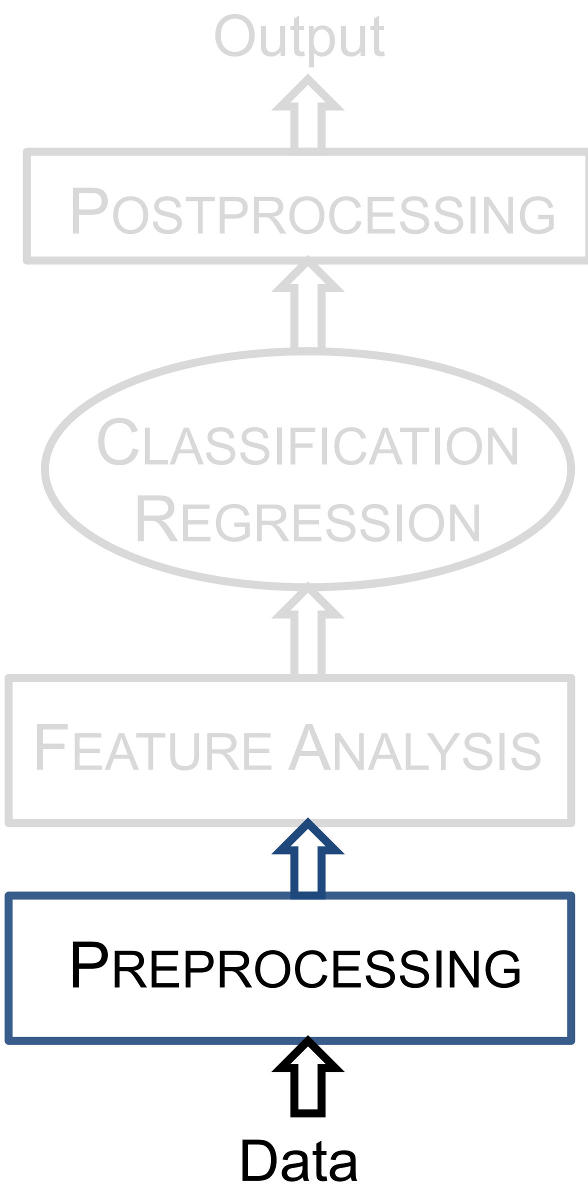
1. Preprocessing
2. Features, low-level data representation
3. Classification / regression with ANN
4. Postprocessing (alternative)

Pattern recognition pipeline



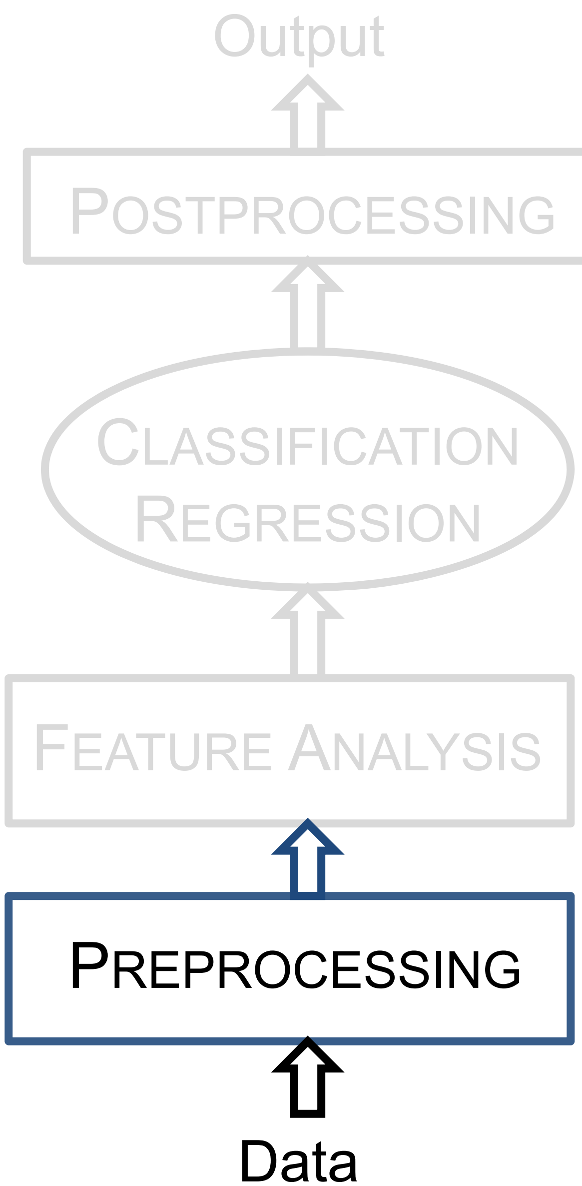
1. Preprocessing
   - familiarise yourself with data and problem
     - what is the objective and assumptions?
     - what data are available?
     - how are/were data generated?
     - type of attributes, their distribution
     - plot data, estimate basic statistics, correlations
     - what is prior knowledge?
   - data quality assessment
   - de-noising, outlier analysis
   - data transformations, normalisation
   - missing data

Pattern recognition pipeline



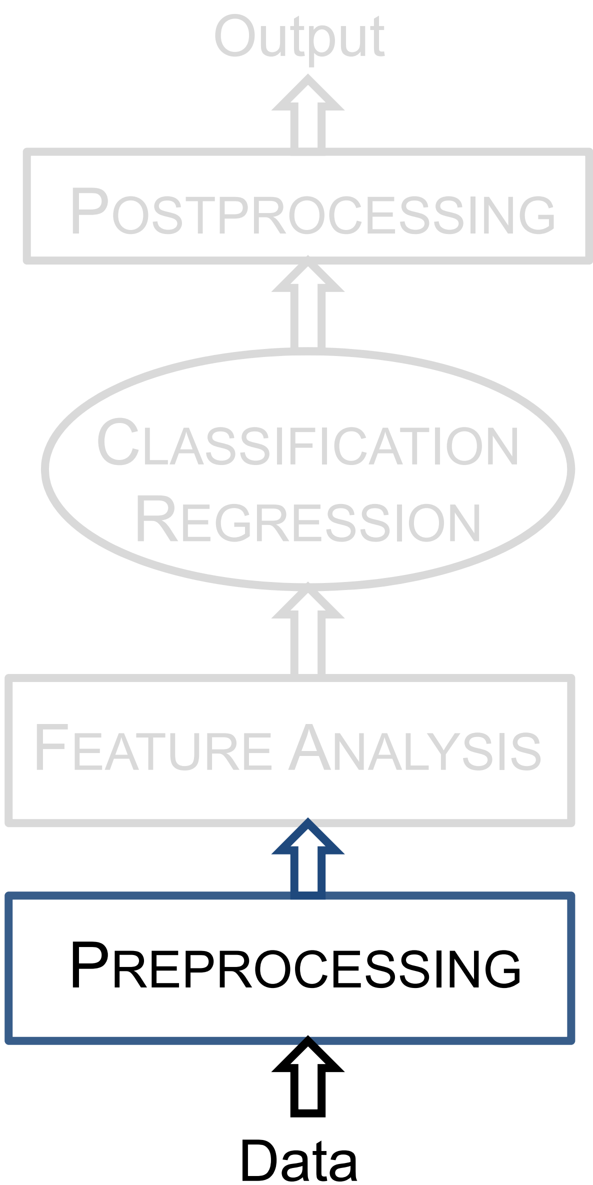
1. Preprocessing
   - familiarise yourself with data and problem
   - data quality assessment
     - train & test data from the same distribution?
     - dimensionality, amount of data
     - dealing with discontinuities
   - de-noising, outlier analysis
   - data transformations, normalisation
   - missing data

Pattern recognition pipeline



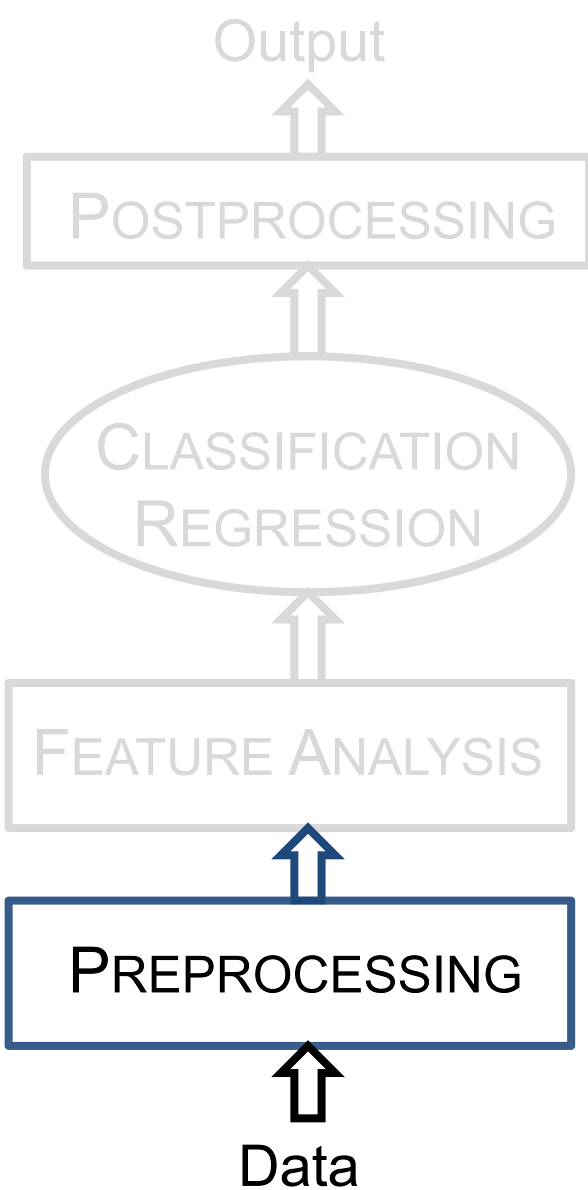
1. Preprocessing
   - familiarise yourself with data and problem
   - data quality assessment
   - de-noising, outlier analysis
     - collect information about noise
     - noise removal
     - outlier detection – remove?
     - filtering
   - data transformations, normalisation
   - missing data

Pattern recognition pipeline



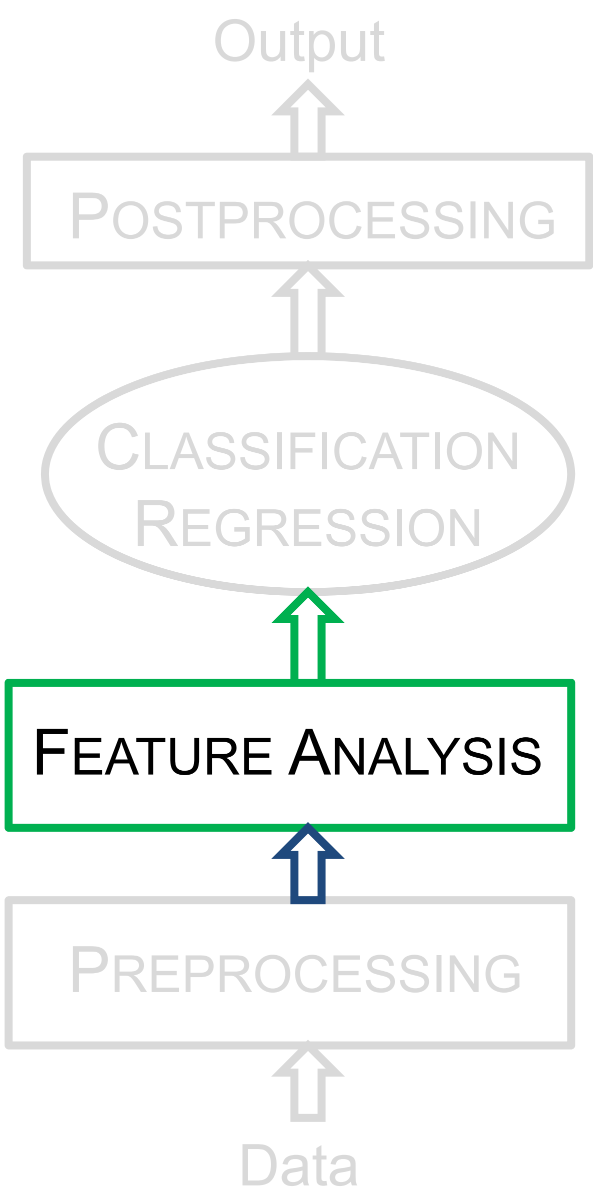
1. Preprocessing
   - familiarise yourself with data and problem
   - data quality assessment
   - de-noising, outlier analysis
   - data transformations, normalisation
     - attribute normalisation
     - whitening
     - scaling
   - missing data

Pattern recognition pipeline



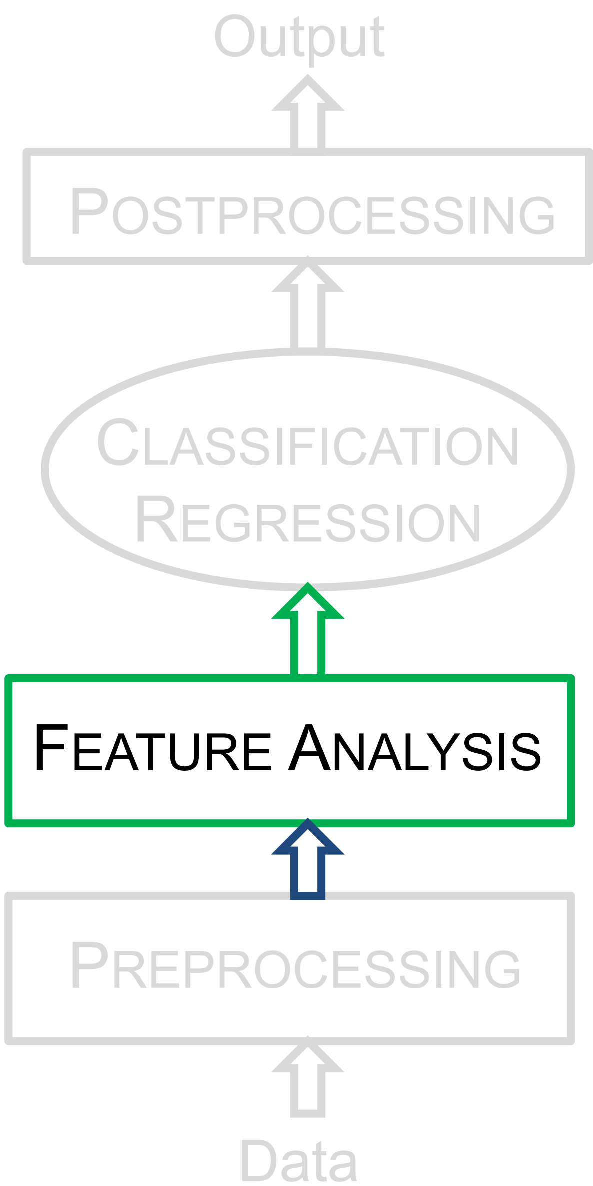
1. Preprocessing
   - familiarise yourself with data and problem
   - data quality assessment
   - de-noising, outlier analysis
   - data transformations, normalisation
   - missing data
     - remove
     - replace with the mean
     - estimate by regression
     - handle by the pattern recognition algorithm

Pattern recognition pipeline



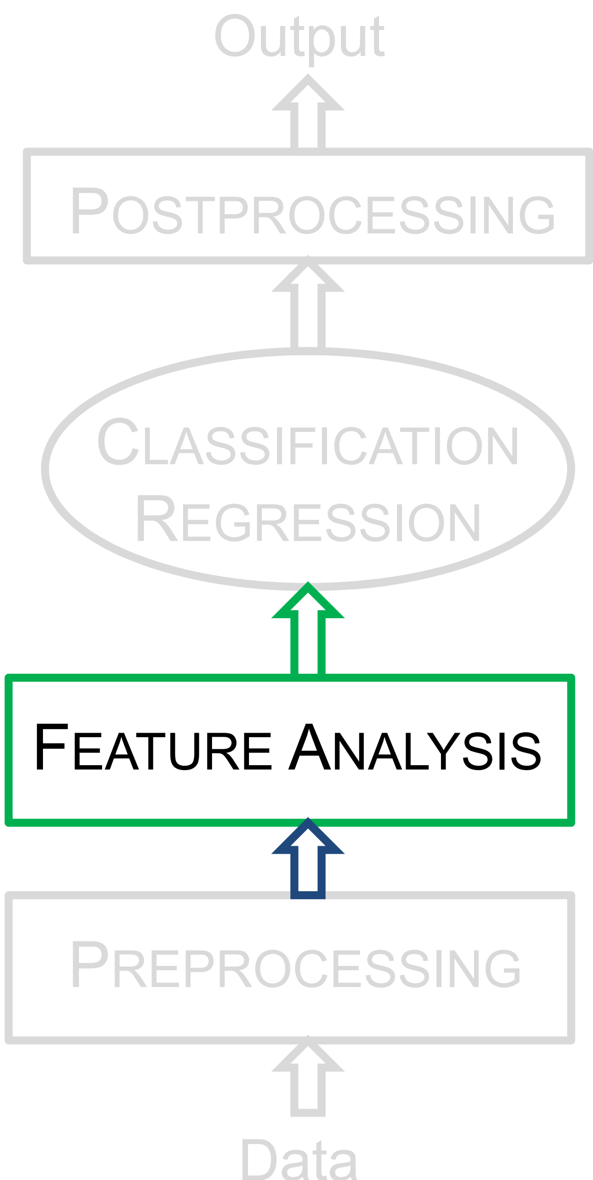
1. Preprocessing
2. Features, low-level data representation
   - dimensionality reduction
     - PCA, SOM, ICA to study data in lower-dim spaces or extract features (projections)
     - decorrelation
   - transformation to a new space
   - feature selection

Pattern recognition pipeline



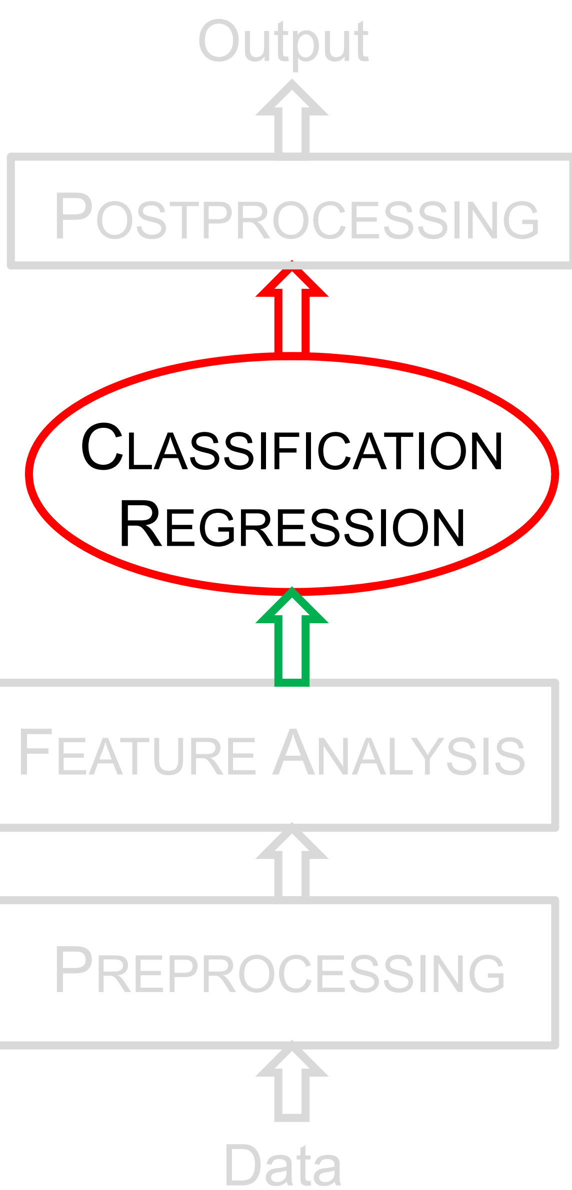
1. Preprocessing
2. Features, low-level data representation
   - dimensionality reduction
   - transformation to a new space
     - low-level data representations, extracting domain specific features
     - invariances (translational, rotational, etc.), symmetries
     - sparsification,redundancy, orthogonalisation
     - encoding, e.g. interval coding
   - feature selection

Pattern recognition pipeline



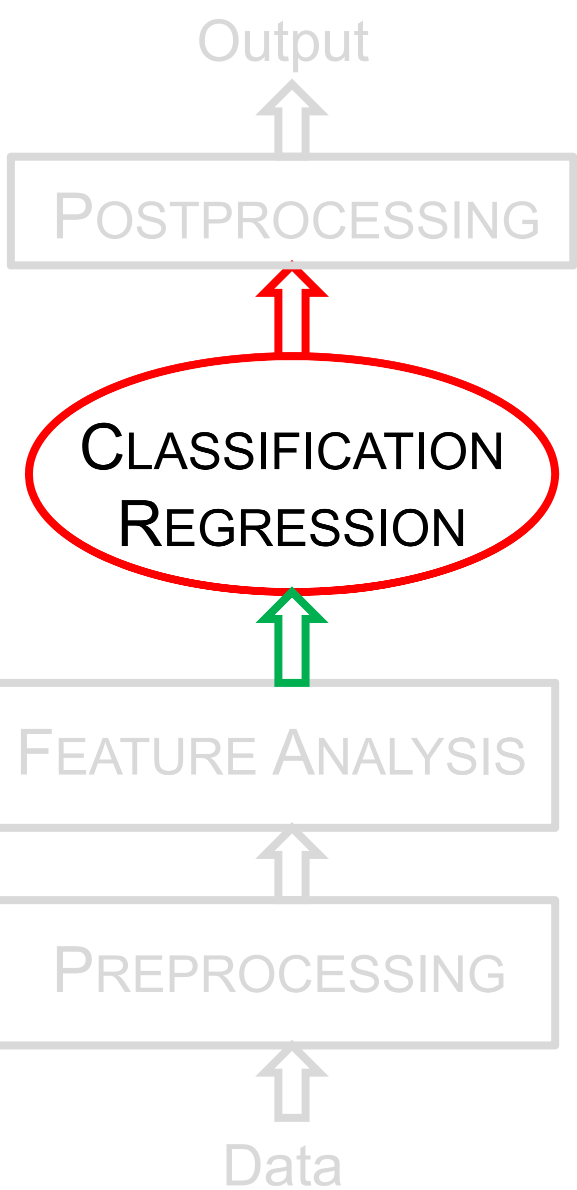
1. Preprocessing
2. Features, low-level data representation
   - dimensionality reduction
   - transformation to a new space
   - feature selection
     - search techniques
     - criteria of evaluation, e.g. filtering, wrapping

Pattern recognition pipeline



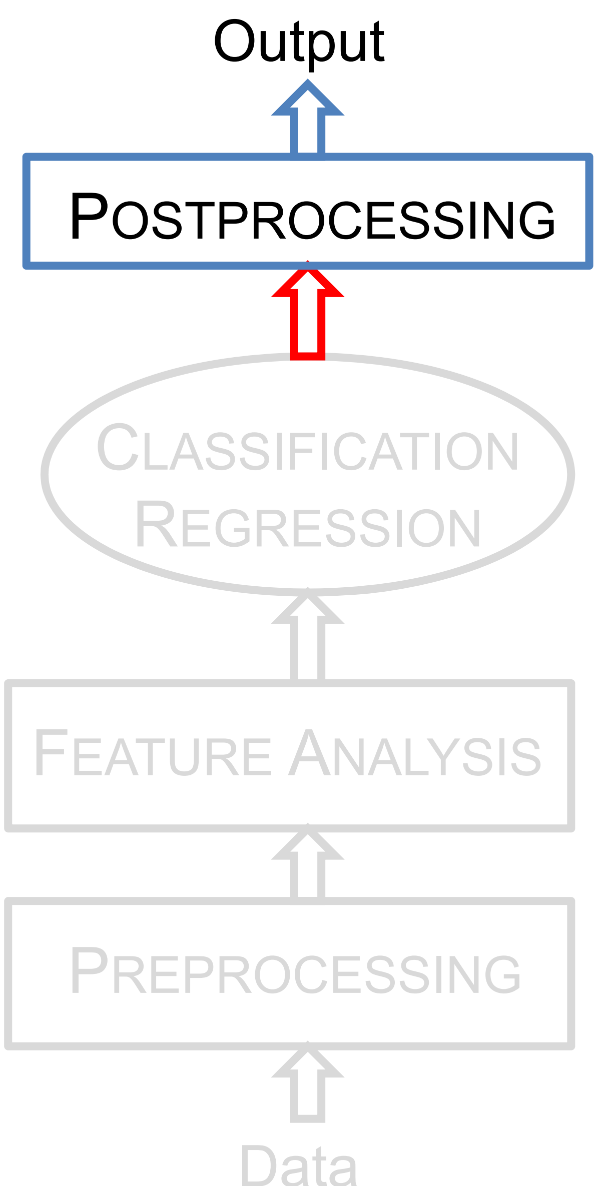
1. Preprocessing
2. Features, low-level data representation
3. Classification / regression with ANN
   - generalisation issues
     - underfitting vs overfitting
     - regularisation, cross-validation
     - assumption about smooth data distribution
   - model selection

Pattern recognition pipeline



1. Preprocessing
2. Features, low-level data representation
3. Classification / regression with ANN
   - generalisation issues
   - model selection
     - validation
     - configuration, hyperparameter optimisation

Pattern recognition pipeline

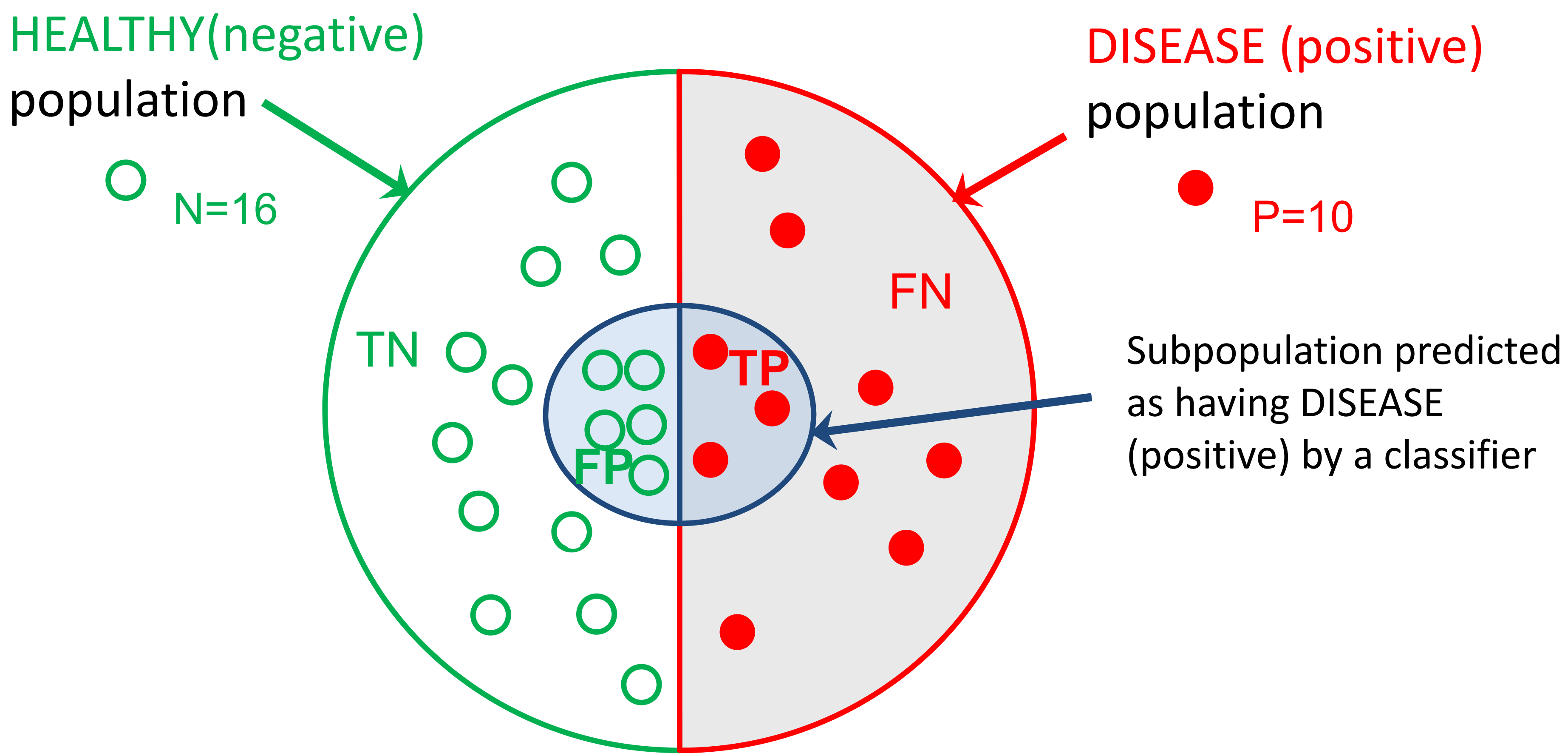


1. Preprocessing
2. Features, low-level data representation
3. Classification / regression with ANN
4. Postprocessing (alternative)
   - interpretation
   - in relation to preprocessing
   - domain-, problem-dependent processing

Error measures – performance metrics

- Decide on the target measure of performance (potentially related to key performance indicators) and specific metric
  - sum square error (with or without normalisation), root-mean-square
  - accuracy for classification tasks *BUT does it suffice*?

Specificity vs sensitivity in classification/diagnostics



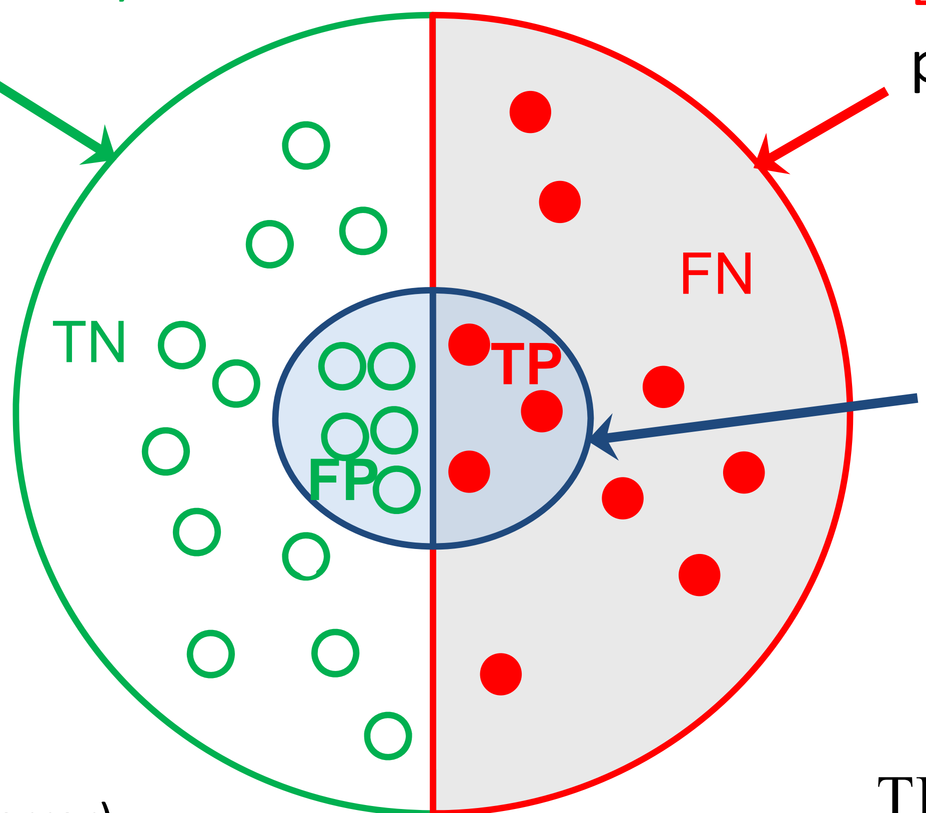
Specificity vs sensitivity in classification/diagnostics

HEALTHY(negative) population

N=16

DISEASE (positive) population

P=10



Subpopulation predicted as having DISEASE (positive) by a classifier

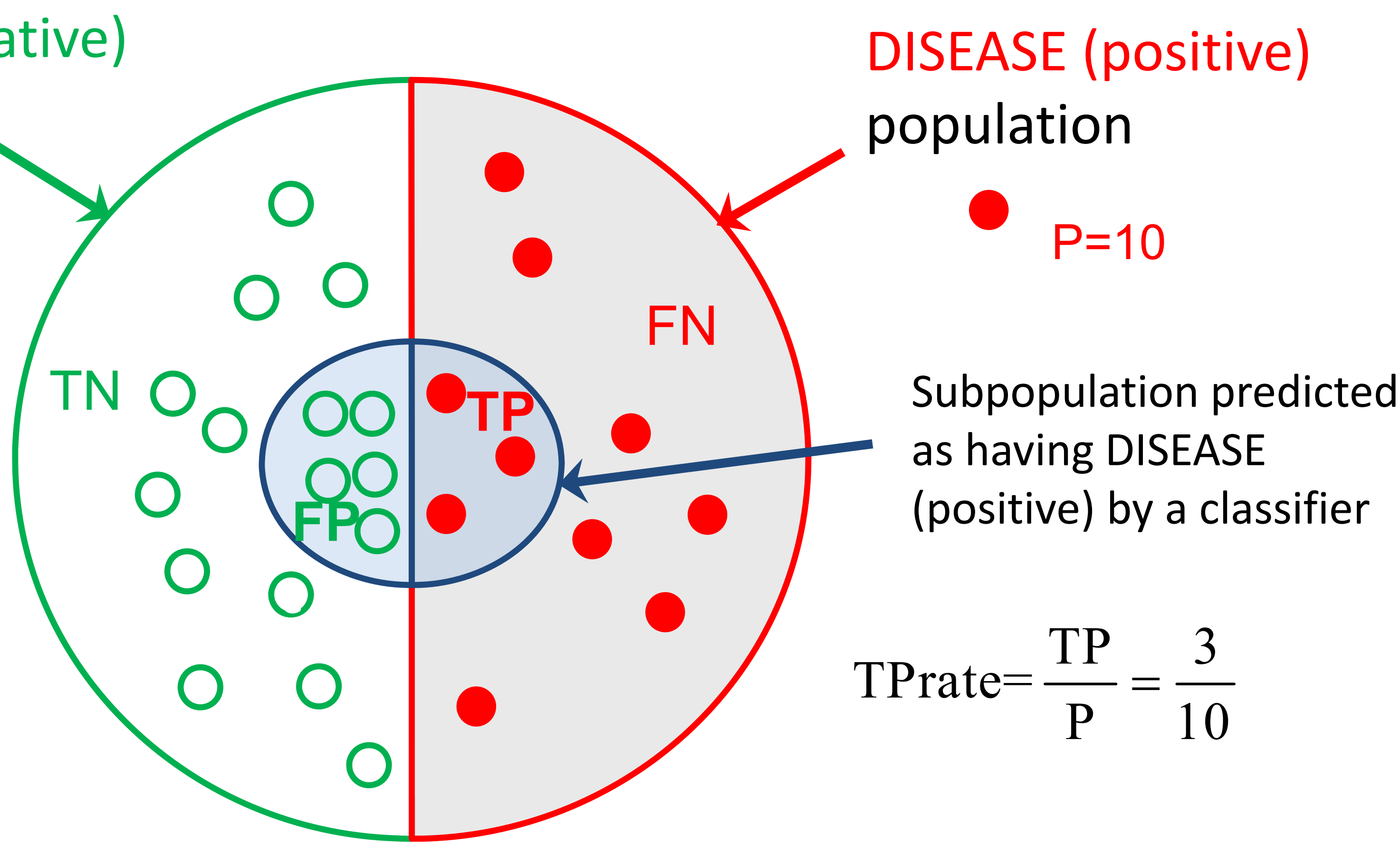
$$\text{FPrate} = \frac{\text{FP}}{\text{N}} = \frac{5}{16}$$ (*Type I* error)

$$\text{TPrate} = \frac{\text{TP}}{\text{P}} = \frac{3}{10}$$

$$\text{TNrate} = \frac{\text{TN}}{\text{N}} = 1 - \text{FPrate} = \frac{11}{16}$$

$$\text{FNrate} = \frac{\text{FN}}{\text{P}} = 1 - \text{TPrate} = \frac{7}{10}$$ (*Type II* error)

Specificity vs sensitivity in classification/diagnostics



$$\text{TNrate}=\frac{\text{TN}}{\text{N}}=\frac{11}{16}$$

$$\text{TPrate}=\frac{\text{TP}}{\text{P}}=\frac{3}{10}$$

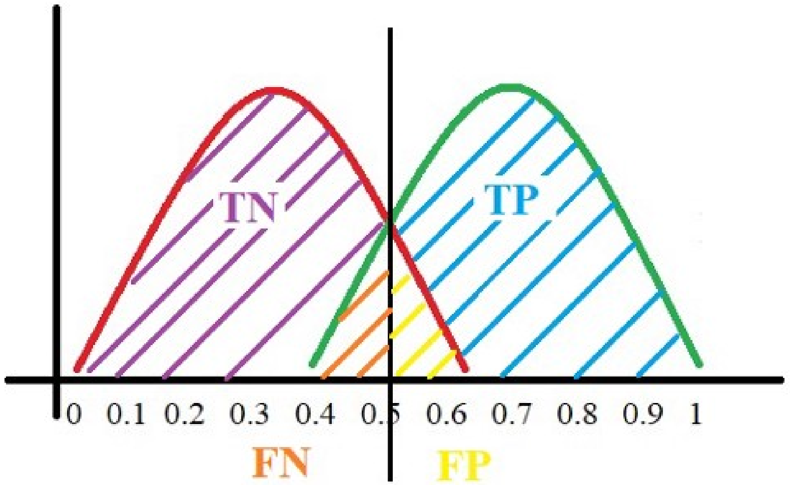
$$\textbf{Specificity}\ (\text{selectivity})=\text{TNrate}=\frac{\text{TN}}{\text{TN+FP}}=\frac{11}{16}$$

$$\textbf{Precision}=\frac{\text{TP}}{\text{TP+FP}}=\frac{3}{8}$$

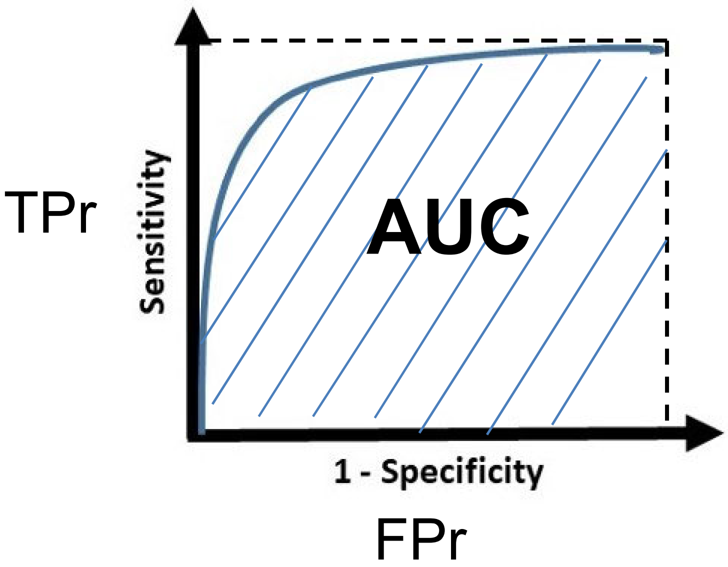
$$\textbf{Sensitivity}\ (\textbf{Recall})=\text{TPrate}=\frac{\text{TP}}{\text{TP+FN}}=\frac{3}{10}$$

$$\textbf{Fscore}=2\times\frac{\text{Precision}\times\text{Recall}}{\text{Precision}+\text{Recall}}$$

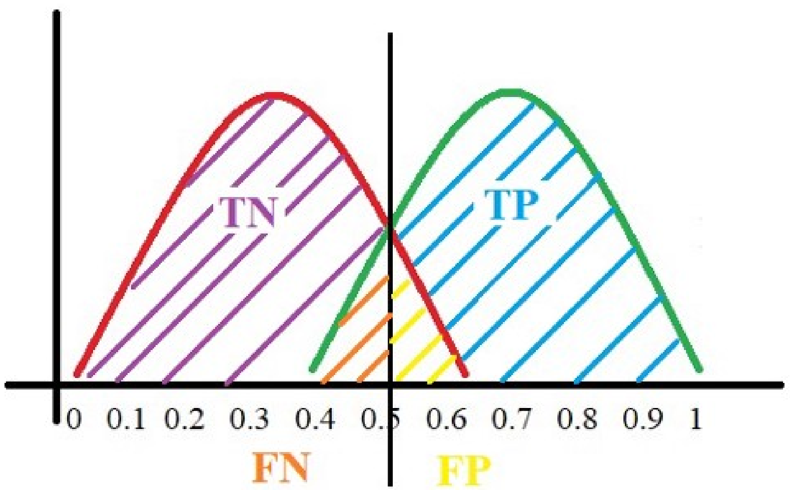
ROC curve in classification/diagnostics



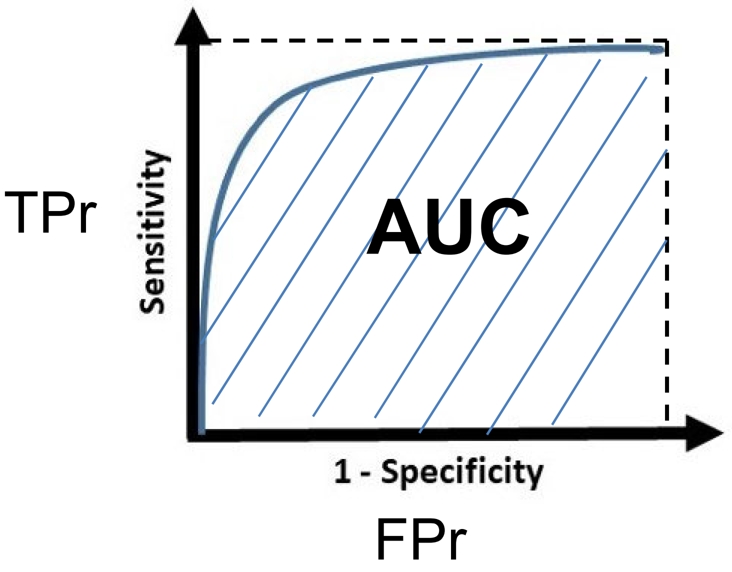
Receiver operating characteristic



ROC curve in classification/diagnostics



Receiver operating characteristic



Error measures – performance metrics

- Decide on the target measure of performance (potentially related to key performance indicators) and specific metric
  - sum square error (with or without normalisation), root-mean-square
  - accuracy for classification tasks
  - precision, recall, ROC curve (area under the curve, AUC)
  - F-score: ***F* = 2*pr* / (*p*+*r*)**, where: *p* - precision, *r* – recall
- More advanced measures
  - weighted errors, e.g. weighted sum of squares
  - probabilistic measures for classification, e.g. cross-entropy for two or multiple classes (if the output represents probabilities by *softmax* activation)

$$-\sum_{c=1}^{M} y_{o,c} \log(p_{o,c})$$

Outline of optimisation algorithms

Beyond gradient descent

- Extensions to gradient descent
- Linear search methods
- Conjugate gradients (+ scaled conjugate gradients)
- Newton’s method (making explicit use of Hessian) and quasi-Newton approach
- The Levenberg-Marquardt algorithm

Committee of networks

- Basic idea: combine weak learners and boost performance
- Concept in opposition to best model selection
- Question of extra computational effort
- Key questions:
  - Which learners? How to train them, on what data?
  - How to combine learners?

Ensemble methods – simple averaging

Model averaging as a general strategy for ensemble methods

The averaged output of the ensemble (committee):

$$y_{COM} = \frac{1}{K}\sum_{i=1}^{K} y_{IND}^{(i)}\ , \qquad \varepsilon_{IND}^{(i)} \to \varepsilon_i \sim MVN(0,\mathbf{C})$$

where: K – the number of weak learners

ε_i – error committed by the i-th weak (individual) learner, $y_{IND}^{(i)}$

cov $\mathbf{C}$ is defined by $\mathbb{E}\left[\varepsilon_i^2\right] = v, \quad \mathbb{E}\left[\varepsilon_i \varepsilon_j\right] = c$

diagonal
error variance of each learner

off-diagonal: **correlations between errors of different learners**

Ensemble methods – simple averaging

Model averaging as a general strategy for ensemble methods

The expected square error of the ensemble:

$$\mathbb{E}\left[\varepsilon_{COM}^2\right] = \mathbb{E}\left[\left(\frac{1}{K}\sum_i y_{IND}^{(i)} - T\right)^2\right] = \mathbb{E}\left[\left(\frac{1}{K}\sum_i \varepsilon_i\right)^2\right] = \frac{1}{K^2}\mathbb{E}\left[\sum_i\left(\varepsilon_i^2 + \sum_{i\neq j}\varepsilon_i\varepsilon_j\right)\right]$$

Ensemble methods – simple averaging

Model averaging as a general strategy for ensemble methods

The expected square error of the ensemble:

$$\mathbb{E}\left[\varepsilon_{COM}^2\right] = \mathbb{E}\left[\left(\frac{1}{K}\sum_i y_{IND}^{(i)} - T\right)^2\right] = \mathbb{E}\left[\left(\frac{1}{K}\sum_i \varepsilon_i\right)^2\right] = \frac{1}{K^2}\mathbb{E}\left[\sum_i\left(\varepsilon_i^2 + \sum_{i\neq j}\varepsilon_i\varepsilon_j\right)\right] = \ldots$$

$$\ldots = \frac{1}{K}\,\mathbb{E}\left[\varepsilon_i^2\right] + \frac{K-1}{K}\,\mathbb{E}\left[\varepsilon_i\varepsilon_j\right]$$

Ensemble methods – simple averaging

Model averaging as a general strategy for ensemble methods

The expected square error of the ensemble:

$$\mathbb{E}\left[\varepsilon_{COM}^2\right] = \mathbb{E}\left[\left(\frac{1}{K}\sum_i y_{IND}^{(i)} - T\right)^2\right] = \mathbb{E}\left[\left(\frac{1}{K}\sum_i \varepsilon_i\right)^2\right] = \frac{1}{K^2}\mathbb{E}\left[\sum_i\left(\varepsilon_i^2 + \sum_{i\neq j}\varepsilon_i\varepsilon_j\right)\right] = \ldots$$

$$\ldots = \frac{1}{K}\,\mathbb{E}\left[\varepsilon_i^2\right] + \frac{K-1}{K}\,\mathbb{E}\left[\varepsilon_i\varepsilon_j\right]$$

If the errors of individual learners are **uncorrelated**

$$\mathbb{E}\left[\varepsilon_i\varepsilon_j\right] = c = 0$$

$$E_{COM} = \mathbb{E}\left[\varepsilon_{COM}^2\right] = \frac{1}{K}\mathbb{E}\left[\varepsilon_{IND}^2\right] = \frac{1}{K}\left(\frac{1}{K}\sum_i^K E_{IND}^{(i)}\right) = \frac{1}{K}\bar{E}_{IND}$$

Ensemble methods – simple averaging

Model averaging as a general strategy for ensemble methods

The expected square error of the ensemble:

$$\mathbb{E}\left[\varepsilon_{COM}^2\right] = \mathbb{E}\left[\left(\frac{1}{K}\sum_i y_{IND}^{(i)} - T\right)^2\right] = \mathbb{E}\left[\left(\frac{1}{K}\sum_i \varepsilon_i\right)^2\right] = \frac{1}{K^2}\mathbb{E}\left[\sum_i\left(\varepsilon_i^2 + \sum_{i\neq j}\varepsilon_i\varepsilon_j\right)\right] = \ldots$$

$$\ldots = \frac{1}{K}\,\mathbb{E}\left[\varepsilon_i^2\right] + \frac{K-1}{K}\,\mathbb{E}\left[\varepsilon_i\varepsilon_j\right]$$

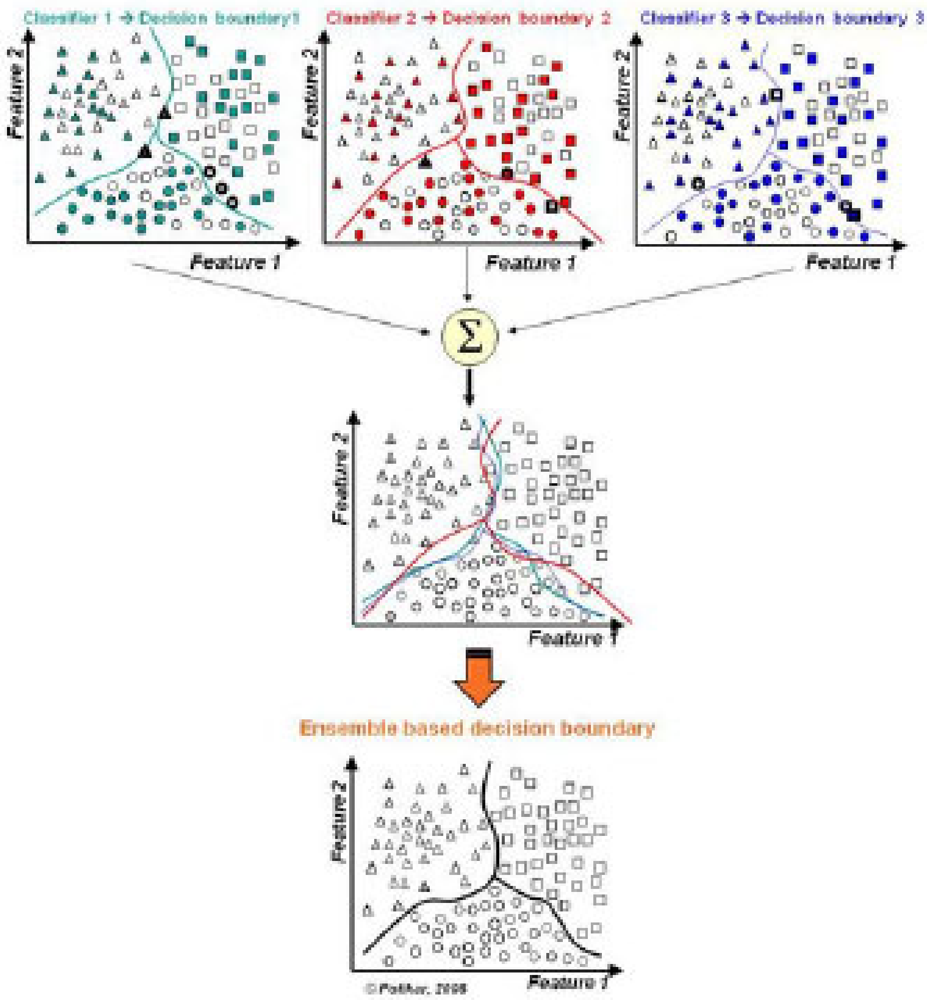
In practice, however, the errors are usually correlated

$$E_{COM} \leq \overline{E}_{INDIV}$$

Bias and variance in ensemble methods

The reduction of error due to reduced variance (without consequences for bias)

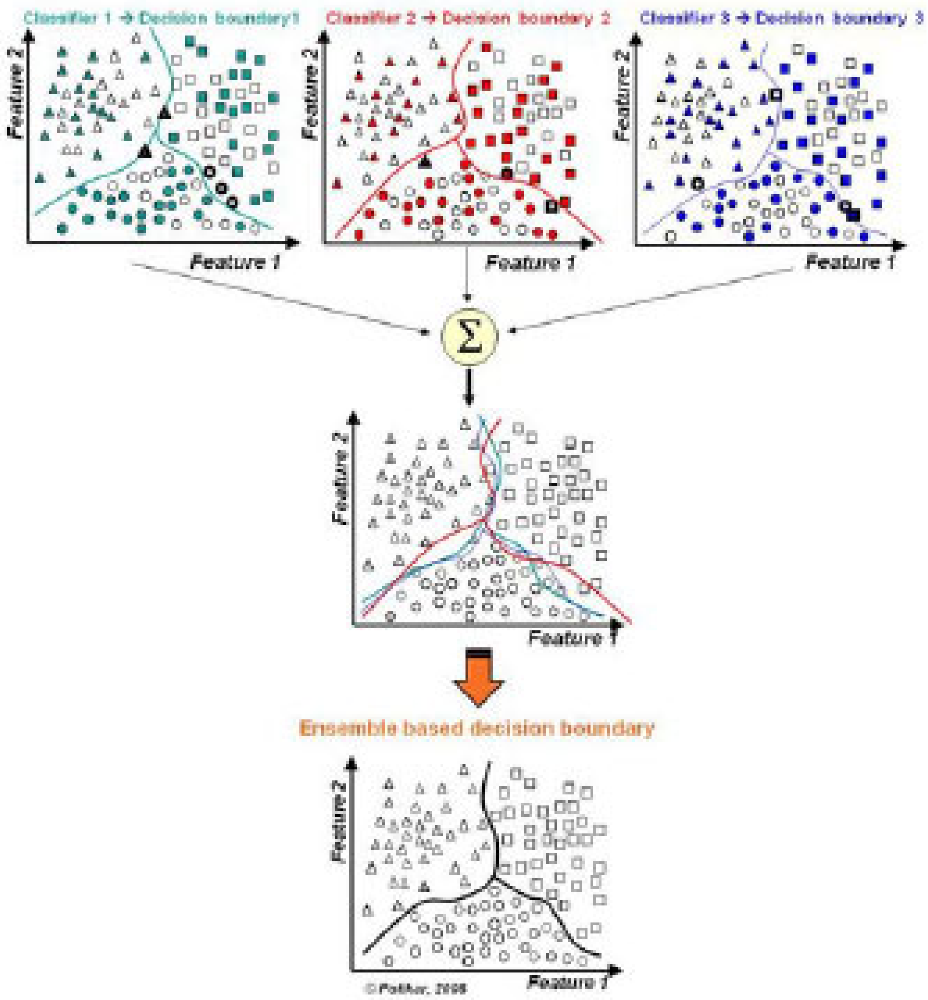
- members of the committee should have relatively *low bias* at the cost of variance, since the extra variance can be removed
- need for diversity and independence of votes/opinions of each learner



Bias and variance in ensemble methods

The reduction of error due to reduced variance (without consequences for bias)

- members of the committee should have relatively *low bias* at the cost of variance, since the extra variance can be removed
- need for diversity and independence of votes/opinions of each learner



Different from individual networks, where bias-variance has to be balanced!

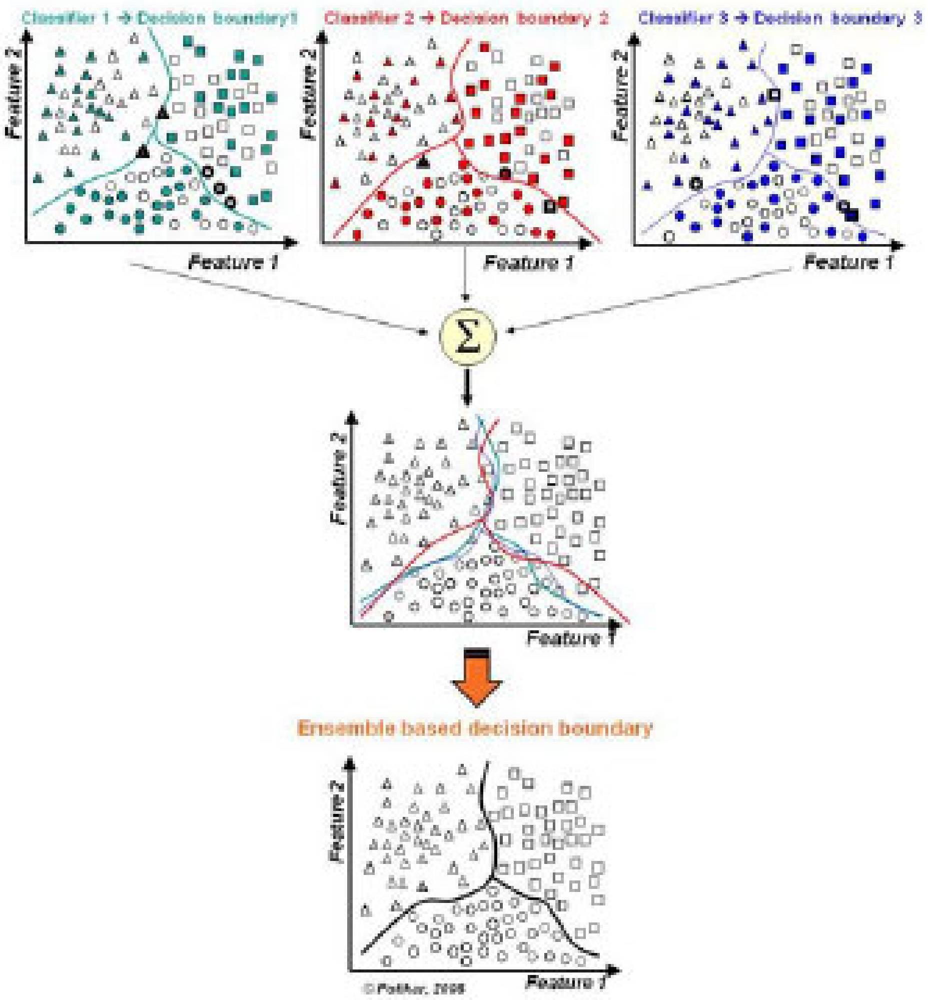
Generalised committee

We can also obtain a *generalised* committee prediction by *weighted combination* of individual predictions:

$$y_{GEN}(\boldsymbol{x}) = \sum_{i=1}^{k} \alpha_i y_i(\boldsymbol{x})$$

It can be shown that

$$E_{GEN} \le E_{COM} \le \overline{E}_{INDIV}$$



Ensemble approaches

Static approaches that do not account for input

- ensemble averaging, bagging
- boosting

Approaches dependent in input

- mixture of experts
- hierarchical mixtures

Bagging

Recipe

- draw a lot of bootstrap samples (sampling with replacement)
- each resample can be treated with additive Gaussian noise (σ=$1/N$)
- train a learner for each bootstrap sample
- combine the outputs of all learners
  - mean or median in regression problems
  - majority vote in classification problems

This is the way to reduce variance, so works well for learners with low bias at the cost of elevated variance.

Boosting

General idea

- iteratively train weak learners on misclassified data
- weigh classifiers depending on their performance and weigh up (boost) misclassified samples

Boosting

General idea

- iteratively train weak learners on misclassified data
- weigh classifiers depending on their performance and weigh up (boost) misclassified samples

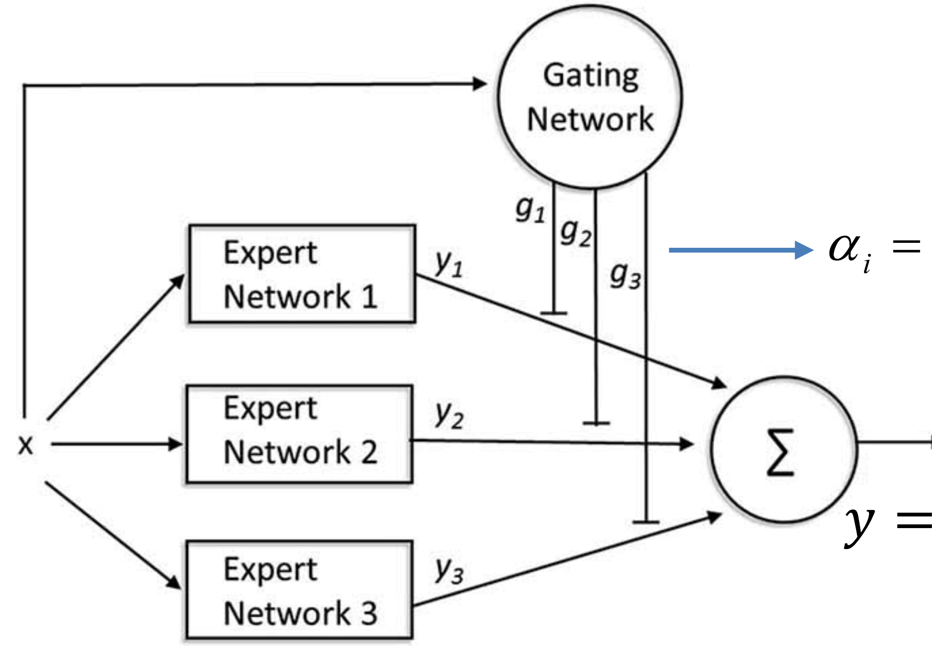
Typical practice

- train a classifier and test it
- allocate (or modify) weights to data in the error function depending whether they were misclassified (boost their importance)
- train another classifier
- to obtain final output, weigh classifiers depending on their performance (weighing hypotheses for a given input depending on the generated error)

Among common methods, AdaBoost is most popular.

Mixtures of experts

- suitable for problems that are not homogenous -> data fusion
- basic idea to train classifiers on sub-problems and aggregate by a linear combination

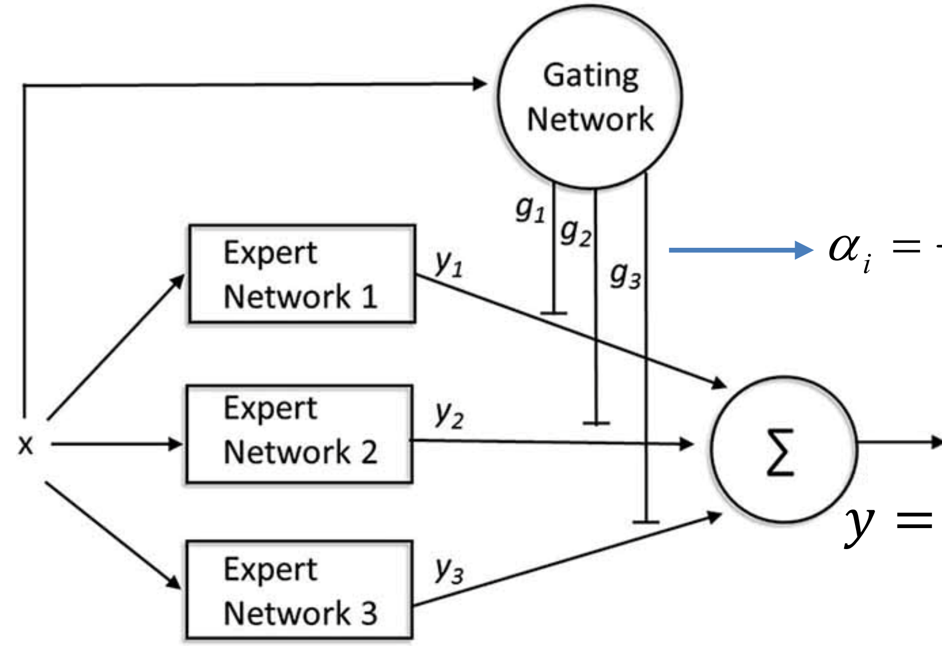


$$\alpha_i = \frac{\exp(g_i)}{\sum_j^K \exp(g_j)}$$

$$y = \frac{1}{K}\sum_i \alpha_i y_i$$

Mixtures of experts

- suitable for problems that are not homogenous -> data fusion
- basic idea to train classifiers on sub-problems and aggregate by a linear combination
- weights for combining the output of individual experts, α, can be trained simultaneously with the learners (gradient descent or EM algorithm)



$$\alpha_i = \frac{\exp(g_i)}{\sum_j^K \exp(g_j)}$$

$$y = \frac{1}{K} \sum_i \alpha_i y_i$$

Negative log-likelihood for the mixture

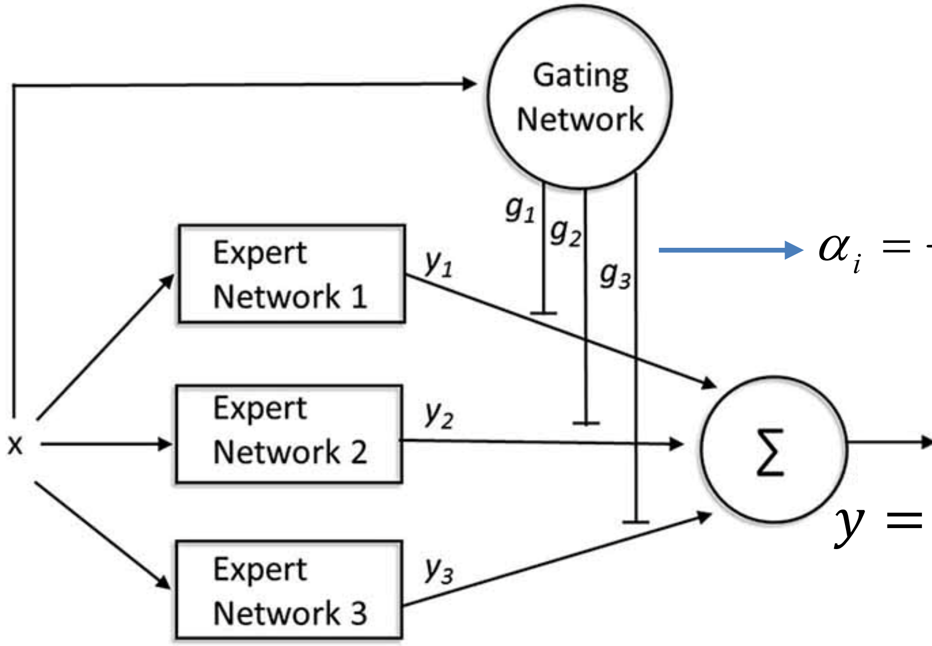
$$E = -\sum_n \ln\left(\sum_{i=1}^{K} \alpha_i(\boldsymbol{x}_n) \varphi_i(\boldsymbol{t}^n \mid \boldsymbol{x}^n) \right)$$

$$\varphi_i(\boldsymbol{t} \mid \boldsymbol{x}) = \mathbb{N}(\|\boldsymbol{t} - \boldsymbol{\mu}(\boldsymbol{x})\|, 1)$$

soft clustering of inputs takes place by means of learning gating function weights

Mixtures of experts

- suitable for problems that are not homogenous -> data fusion
- basic idea to train classifiers on sub-problems and aggregate by a linear combination
- weights for combining the output of individual experts, α,can be trained simultaneously with the learners (gradient descent or EM algorithm)
- alternatively, gating could be a mechanism to select only one learner for making a prediction (not for learning)



$$\alpha_i = \frac{\exp(g_i)}{\sum_j^K \exp(g_j)}$$

$$y = \frac{1}{K} \sum_i \alpha_i y_i$$

Negative log-likelihood for the mixture

$$E = -\sum_n \ln\left(\sum_{i=1}^K \alpha_i(\boldsymbol{x}_n) \varphi_i(\boldsymbol{t}^n \mid \boldsymbol{x}^n) \right)$$

$$\varphi_i(\boldsymbol{t} \mid \boldsymbol{x}) = \mathbb{N}(\|\boldsymbol{t} - \boldsymbol{\mu}(\boldsymbol{x})\|, 1)$$