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)

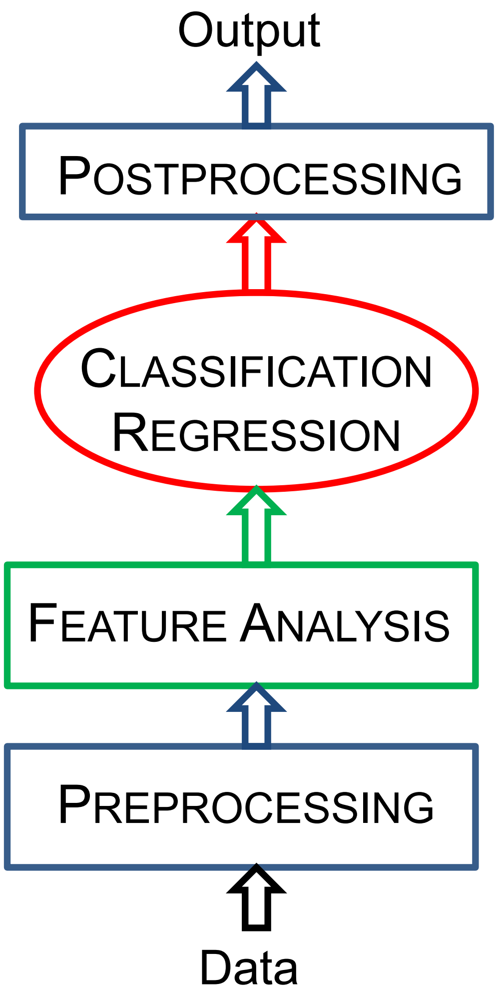
KTH Royal Institute of Technology

January 2018

Lecture overview

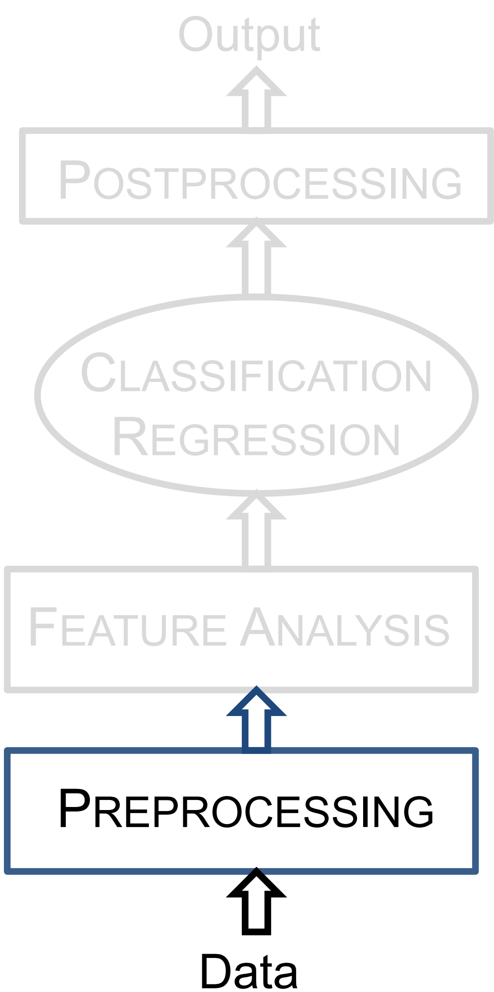
- Data preprocessing and feature extraction
- Error measures
- Parameter optimisation
- Ensemble learning

Pattern recognition pipeline



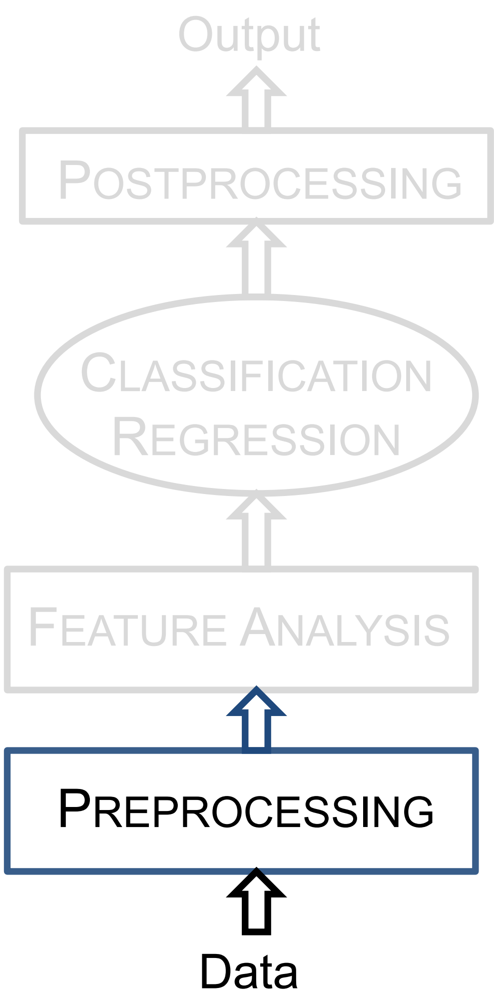
1. Preprocessing
2. Features, low-level data representation
3. Classification / regression with ANN
4. Postprocessing (alternative)

Pattern recognition pipeline



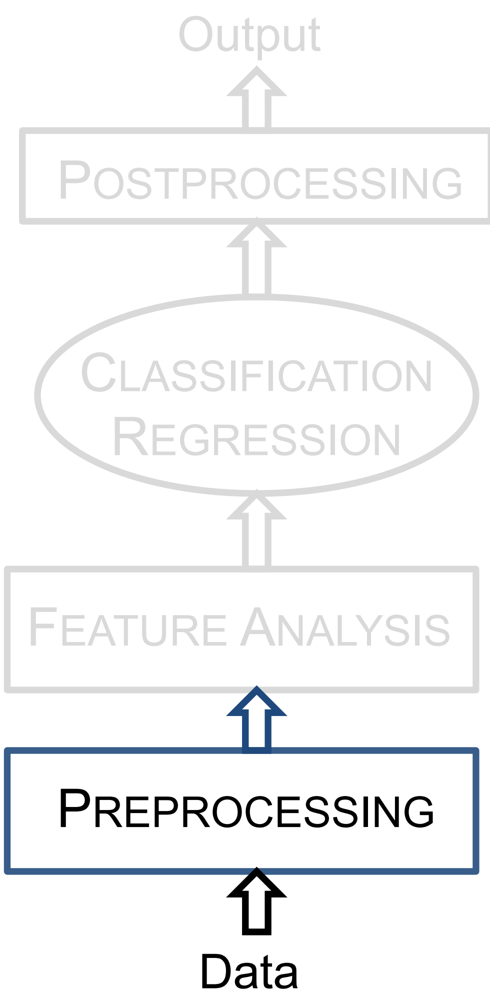
1. Preprocessing
   - familiarise yourself with data and problem
     - what is the objective and assumptions?
     - what data are available?
     - how are/were data generated?
     - type of attributes, their distribution
     - plot data, estimate basic statistics, correlations
     - what is prior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

Pattern recognition pipeline



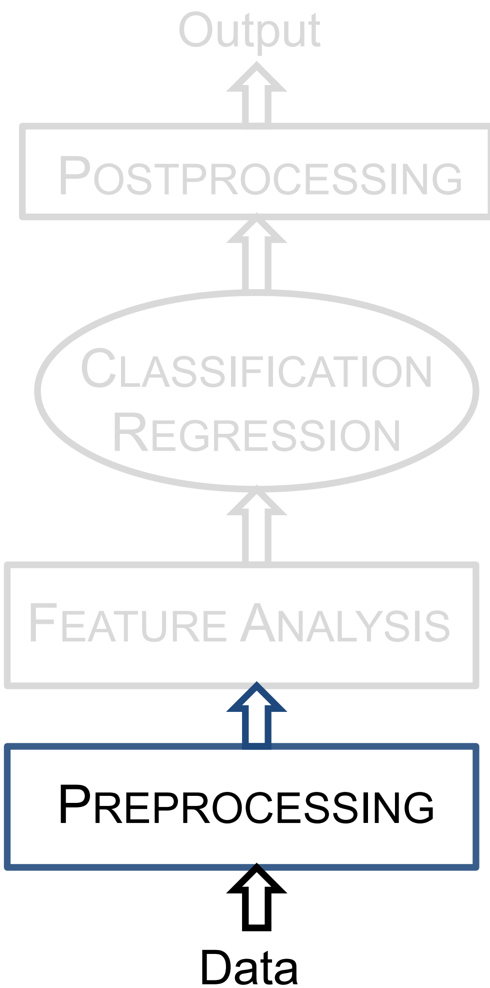
1. Preprocessing
   - familiarise yourself with data and problem
   - data quality assessment
     - train & test data from the same distribution?
     - dimensionality, amount of data
     - dealing with discontinuities
   - de-noising, outlier analysis
   - data transformations, normalisation
   -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

Pattern recognition pipeline



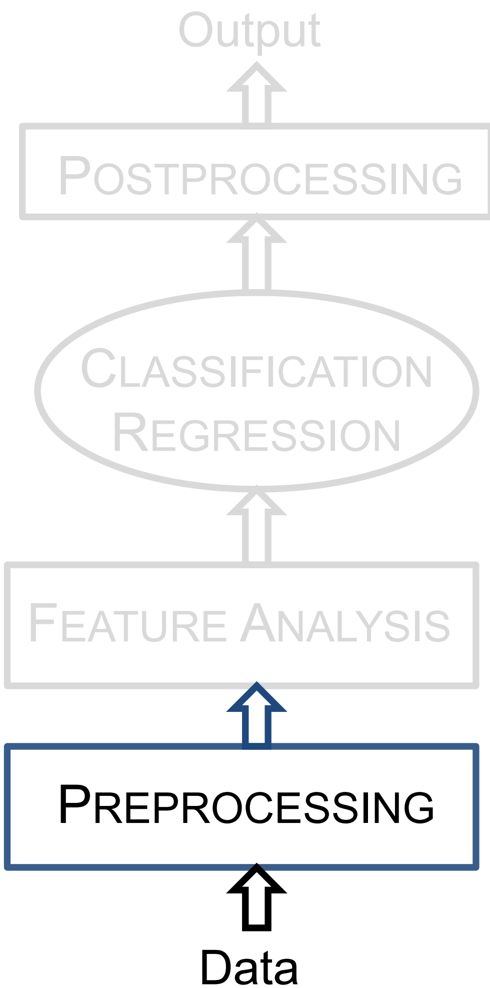
1. Preprocessing
   - familiarise yourself with data and problem
   - data quality assessment
   - de-noising, outlier analysis
     - collect information about noise
     - noise removal
     - outlier detection – remove?
     - filtering
   - data transformations, normalisation
   - missing data

Pattern recognition pipeline



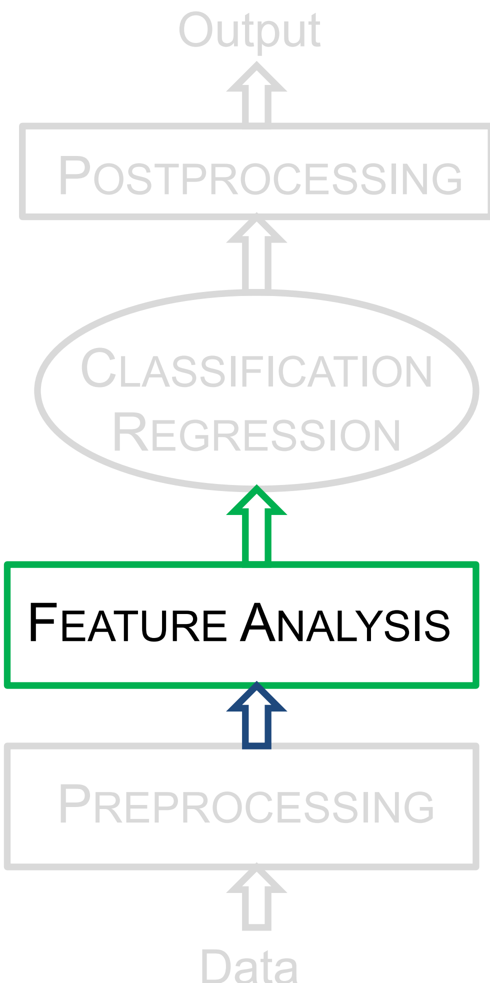
1. Preprocessing
   - familiarise yourself with data and problem
   - data quality assessment
   - de-noising, outlier analysis
   - data transformations, normalisation
     - attribute normalisation
     - whitening
     - scaling
   - missing data

Pattern recognition pipeline



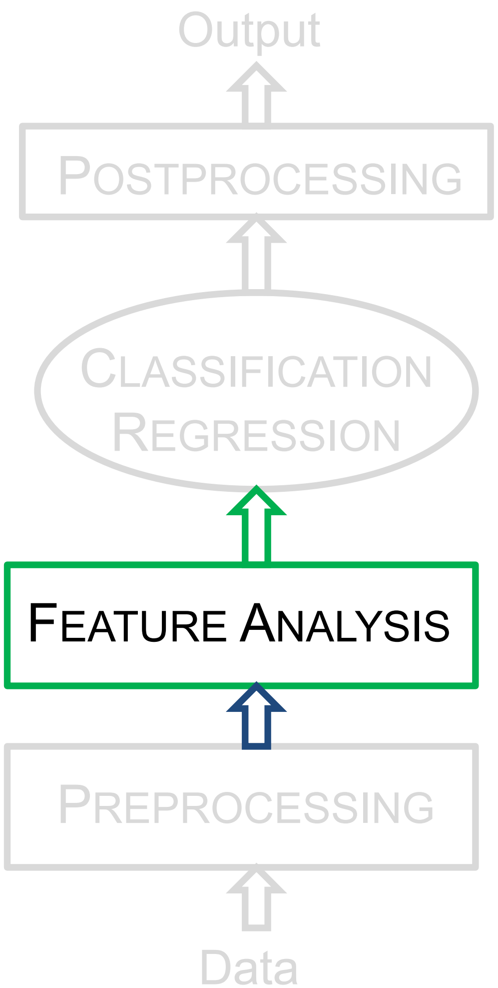
1. Preprocessing
   - familiarise yourself with data and problem
   - data quality assessment
   - de-noising, outlier analysis
   - data transformations, normalisation
   - missing data
     - remove
     - replace with the mean
     - estimate by regression
     - handle by the pattern recognition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

Pattern recognition pipeline



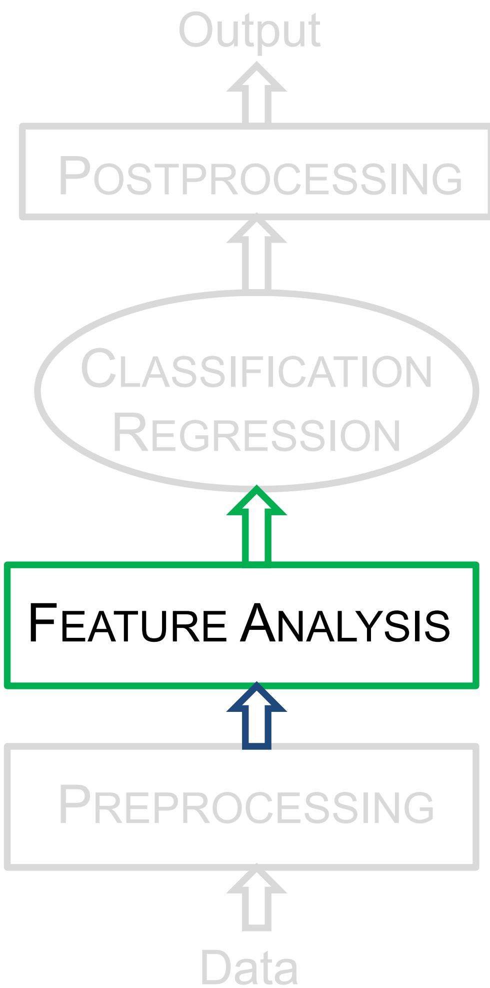
1. Preprocessing
2. Features, low-level data representation
   - dimensionality reduction
     - PCA, SOM, ICA to study data in lower-dim spaces or extract features (projections)
     - decorrelation
   - transformation to a new space
   - feature selection

Pattern recognition pipeline



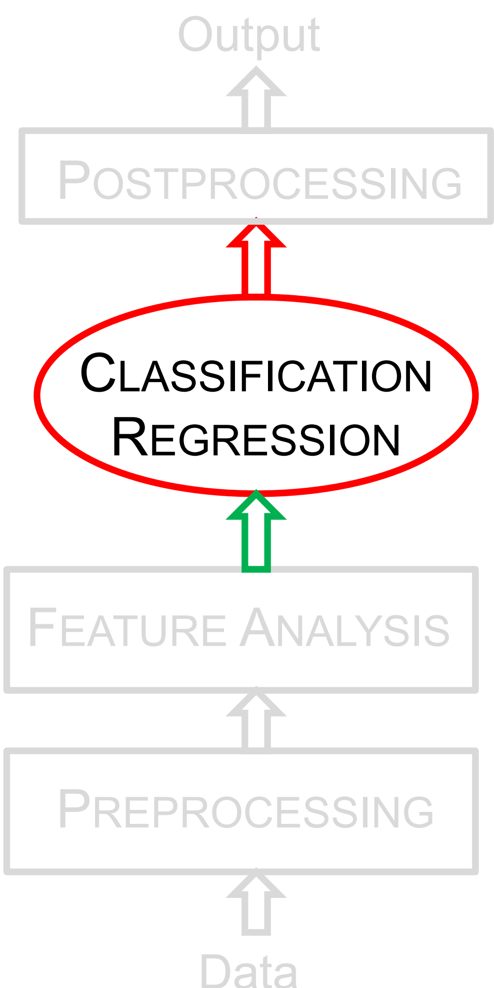
1. Preprocessing
2. Features, low-level data representation
   - dimensionality reduction
   - transformation to a new space
     - low-level data representations, extracting domain specific features
     - invariances (translational, rotational, etc.), symmetries
     - sparsification,redundancy, orthogonalisation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

Pattern recognition pipeline



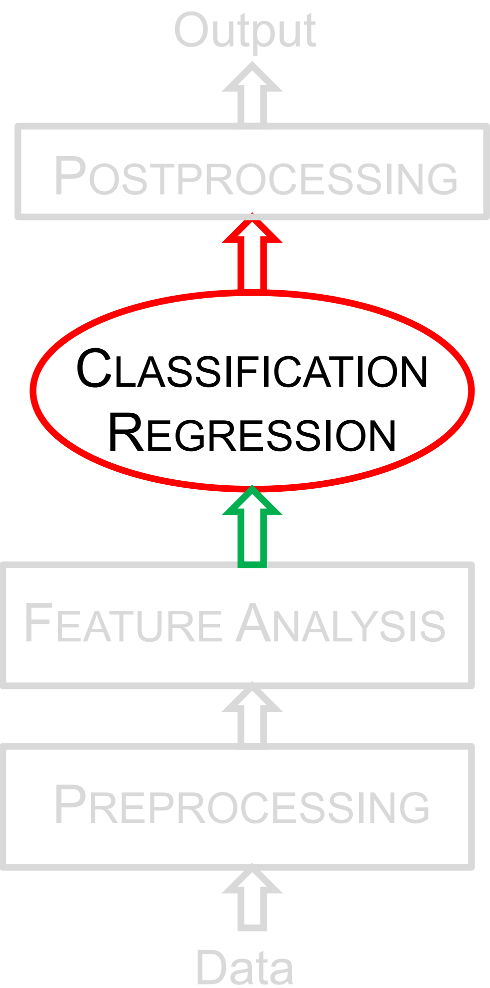
1. Preprocessing
2. Features, low-level data representation
   - dimensionality reduction
   - transformation to a new space
   - feature selection
     - search techniques
     - criteria of evaluation, e.g. filtering, wrapping

Pattern recognition pipeline



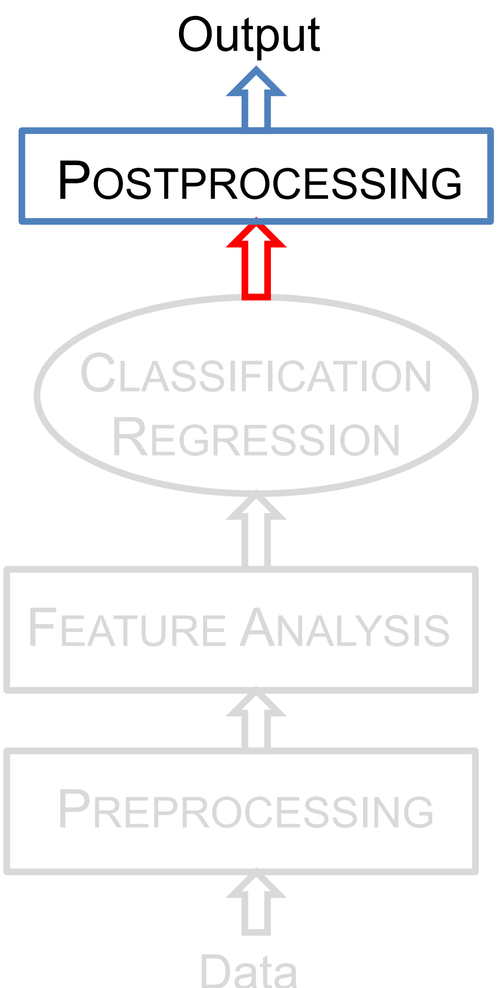
1. Preprocessing
2. Features, low-level data representation
3. Classification / regression with ANN
   - generalisation issues
     - underfitting vs overfitting
     - regularisation, cross-validation
     - assumption about smooth data distribution
   - model selection

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
  - precision, recall, ROC curve (area under the curve, AUC)
  - F-score: $\boldsymbol{F = 2pr / (p+r)}$, where: p - precision, r – recall

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: $\boldsymbol{F = 2pr / (p+r)}$, 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)

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 expected square error of the ensemble:

$$\mathbb{E}\left[\left(\frac{1}{k}\sum_i \epsilon_i\right)^2\right] = \frac{1}{k^2}\mathbb{E}\left[\sum_i \left(\epsilon_i^2 + \sum_{j\neq i} \epsilon_i \epsilon_j\right)\right] = \frac{1}{k}v + \frac{k-1}{k}c.$$

where: k – the number of weak learners

ε_i – error committed by the i-th learner (MVN(0, C))

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

Ensemble methods – simple averaging

Model averaging as a general strategy for ensemble methods

The expected square error of the ensemble:

$$\mathbb{E}\left[\left(\frac{1}{k}\sum_i \epsilon_i\right)^2\right] = \frac{1}{k^2}\mathbb{E}\left[\sum_i \left(\epsilon_i^2 + \sum_{j\neq i}\epsilon_i\epsilon_j\right)\right] = \frac{1}{k}v + \frac{k-1}{k}c.$$

where: k – the number of weak learners

ε_i – error committed by the i-th learner (MVN(0, C))

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

If the errors are uncorrelated, i.e. c=0:

$$E_{COM} = \frac{1}{k}v = \frac{1}{k}\mathbb{E}\left[\varepsilon_i^2\right] = \frac{1}{k}\left(\frac{1}{k}\left(E_{INDIV}^{(1)} + \ldots + E_{INDIV}^{(k)}\right)\right) = \frac{1}{k}\overline{E}_{INDIV}$$

Ensemble methods – simple averaging

Model averaging as a general strategy for ensemble methods

The expected square error of the ensemble:

$$\mathbb{E}\left[\left(\frac{1}{k}\sum_i \epsilon_i\right)^2\right] = \frac{1}{k^2}\mathbb{E}\left[\sum_i \left(\epsilon_i^2 + \sum_{j\neq i} \epsilon_i \epsilon_j\right)\right] = \frac{1}{k}v + \frac{k-1}{k}c.$$

where: k – the number of weak learners

ε_i – error committed by the i-th learner (MVN(0, C))

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

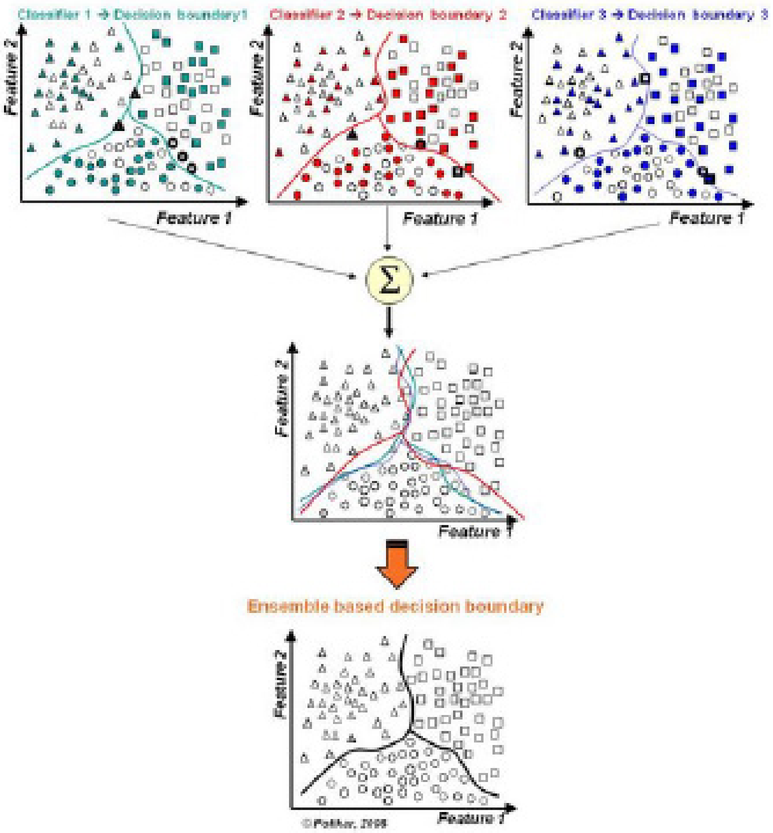
In practice, however, the errors are usually correlated

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

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 <u>extra variance can be removed</u>
- need for diversity and independence of votes/opinions of each learner

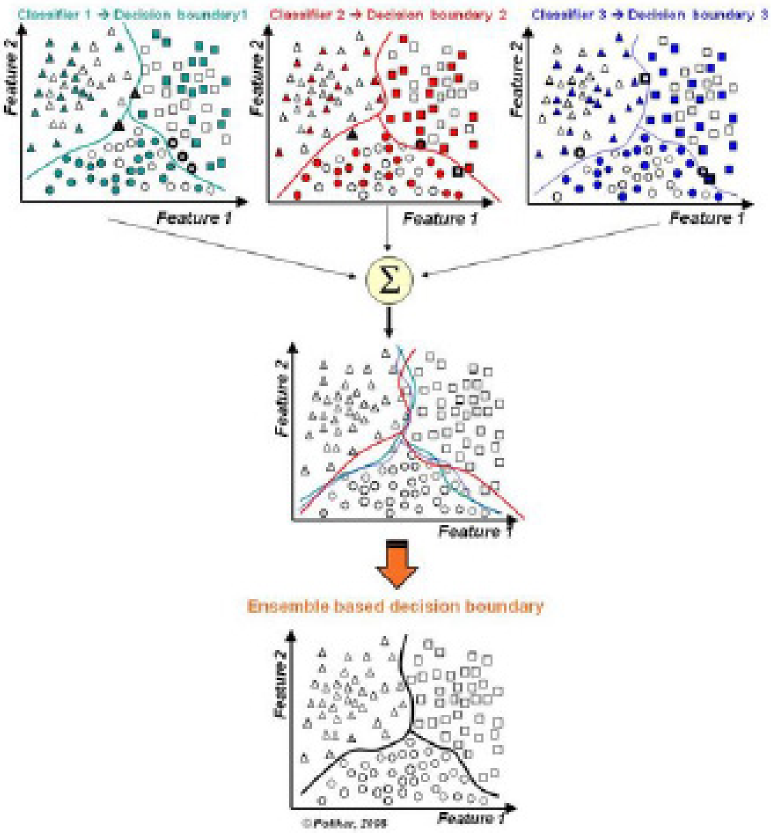


Bias and variance in ensemble methods

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 <u>extra variance can be removed</u>
- need for diversity and independence of votes/opinions of each learner

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



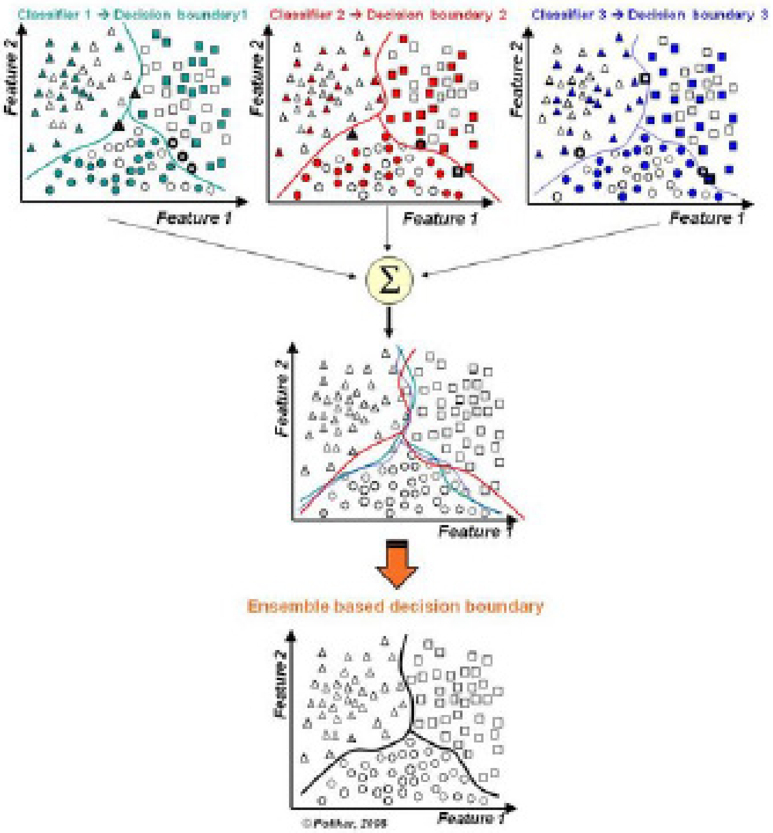
Generalised committee

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

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



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 ($\sigma=1/N$)
- 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

Boosting

General idea

- iteratively train weak learners on misclassified data
- weigh classifiers depending on their performance

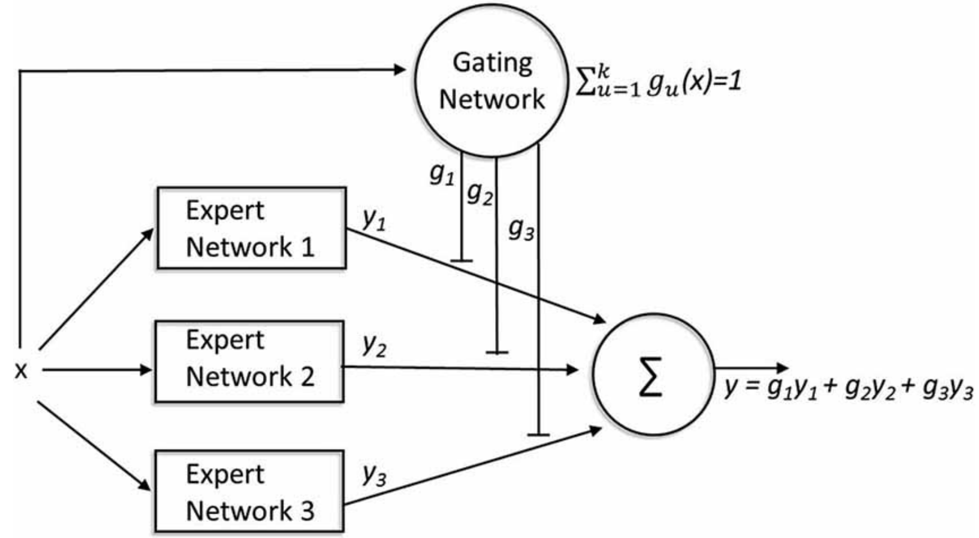
Typical practice

- 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 over the iterations)

Among common methods, AdaBoost is most popular.

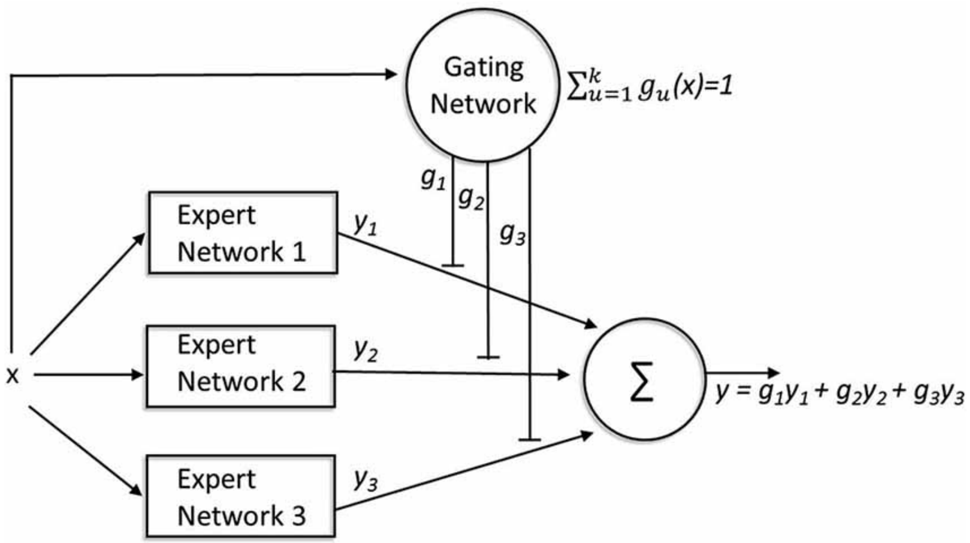
Mixtures of experts

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



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)



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

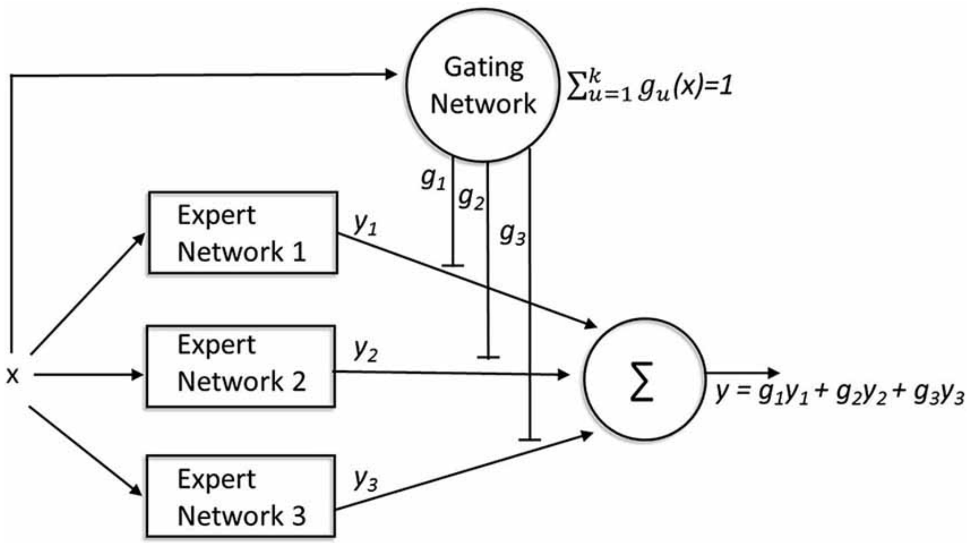
$$\varphi_i(\boldsymbol{t} \mid \boldsymbol{x}) = \mathbb{N}(\|\boldsymbol{t} - \boldsymbol{\mu}(\boldsymbol{x})\|, 1)$$

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

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)



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

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

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