Neural nNetworks: Committee Machines

Prof. dr. sc. Sven Lončarić
Doc. dr. sc. Marko Subašić

Faculty of Electrical Engineering and Computing
University of Zagrebu

http://www.fer.hr/predmet/neumre_a

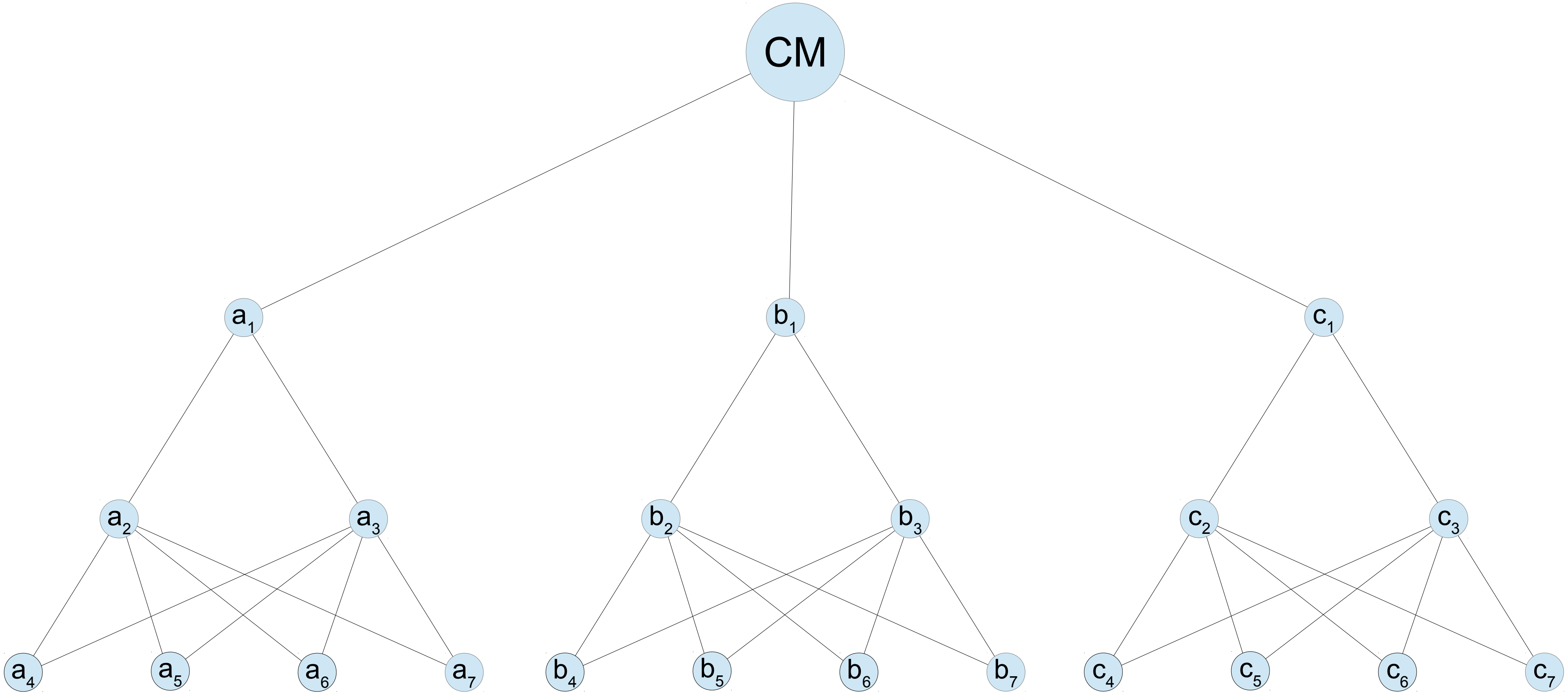
Overview

- Classification using committee machines
- Static structures
  - Group averaging
  - Boosting
- Dinamc structures
  - Mixture of experts
  - Hierarchical mixture of experts

Introduction

- Complex problems can hardly be solved by just one expert
  - Such expert should be very powerfull or “strong”
- Solution: combining of “opinions” from several “weak” experts
- The goal is to obtain the final “opinion” that is more accurate than any individual opinion
- Knowledge fusion
- Modularity

Committee Machines



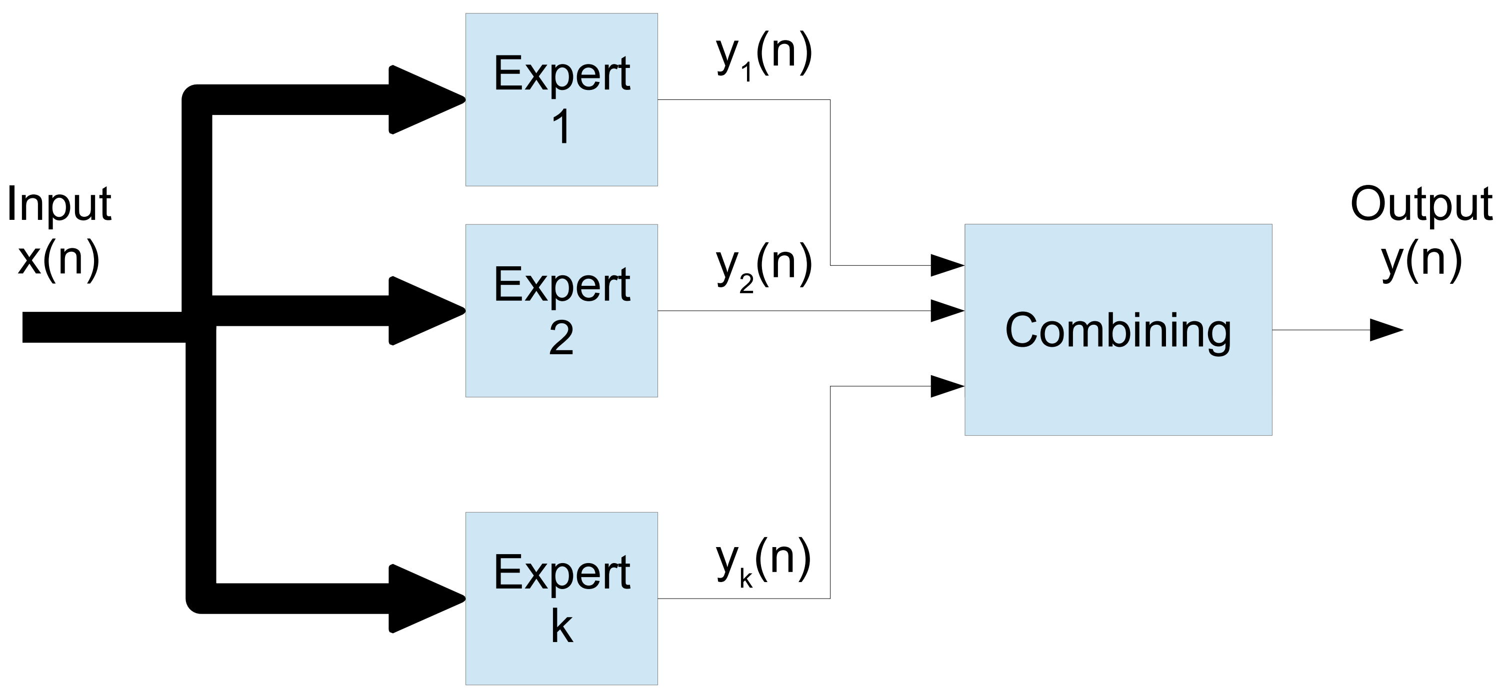
Committee Machines

- Overall architecture resembles one complex neural network
  - Especially if individual experts themselves are neural networks
- The approach is more resiliant to the problem of over-fitting
- It enables problem segmentation and specializatioin of individual experts
  - Over-fitting is even desirable!?
- There is freedom in choosing the way of combining individual experts

Two main categories

1. Statc structure – combining of experts is not influenced by input data
   - Group averaging – linear combination of experts' opinions
   - Boosting – weak eksperts are combined into one strong expert
2. Dinamc structure – input data influences combining of experts
   - Mixture of experts – nonlinear combining
   - Hierarchical mixture of experts – multilayer nonlinear combining/selection

Static structure



Group averaging

- Motivation:
  - Training of all experts and averaging as a whole takes more time than training of individual experts
  - Risk of overfitting is higher with larger and more complex networks

Group averaging

- All experts are trained on the same data
- Since they will be combined, it is even desired to overfit individual experts
- Negative effects of overfitting will be decreased trough combining

Boosting

- General method for improving any learning based algorithm
- Individial experts are trained using different data sets
- Individual experts are called weak classifiers or weak learners
- It is actually desired that they are weak instead of being strong performance-wise

Weak learner

- Strong learner
  - Acceptable success rate is slightly less than 100%
- Weak learner
  - Acceptable success rate is slightly above 50%
  - Only slightly better than tossing a coin
- Boosting creates one strong learner by combining many weak learners

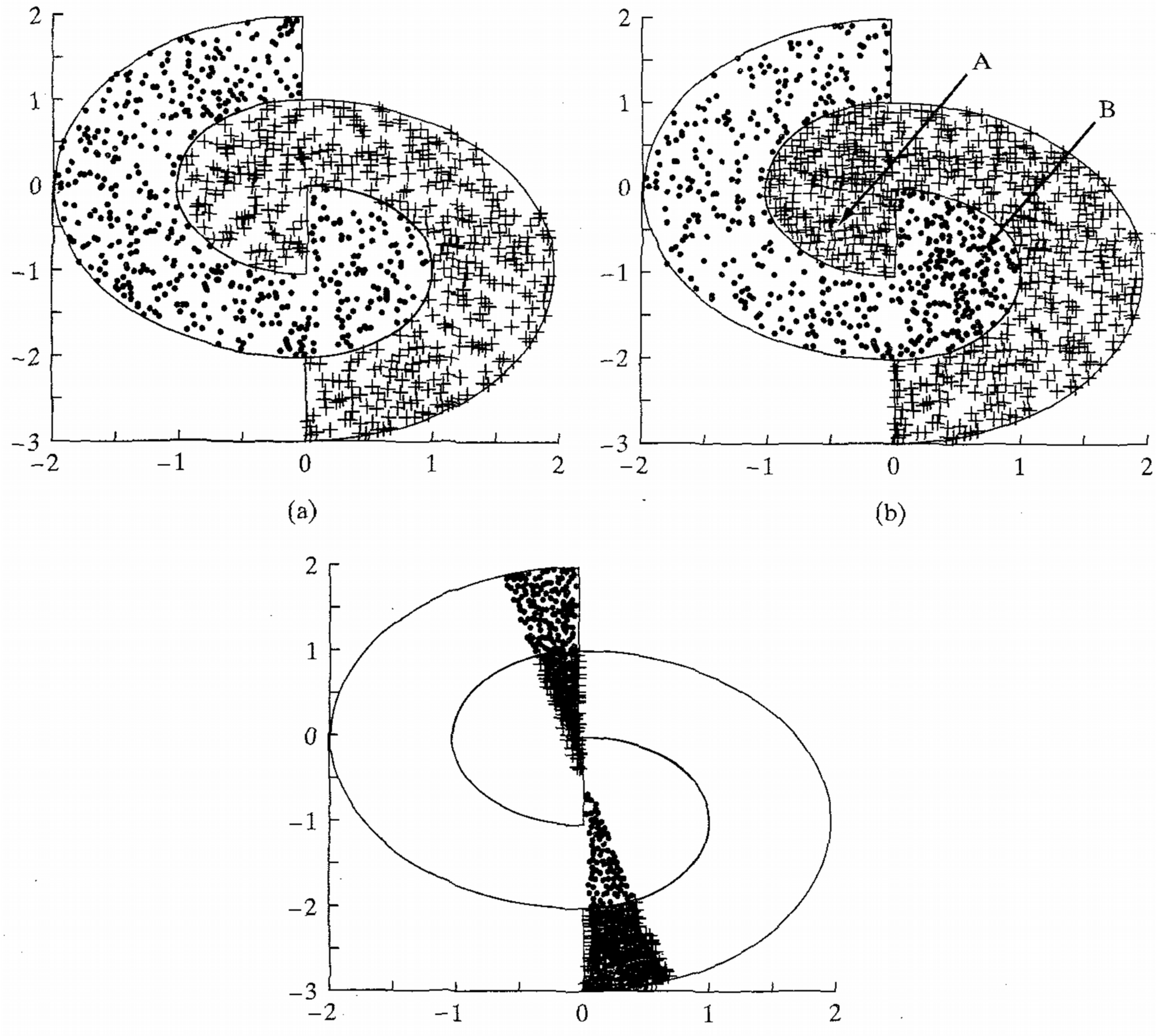
Boosting variants

1. Boosting by filtering – discarding or accepting of individul training samples
2. Boosting by sampling – repetitive sampling of learning data dependant on probability distribution in the learning phase
3. Boosting by weighting – assigning of weights to learning samples

Boosting by filtering

- Assumes large training data set
- Three weak learners
- 1st is trained using N_1 samples
- 2nd is trained using N_1 unused samples on which 1st learner achieves 50% accuracy
  - Random selection of samples that are correctly and incorrectly clasified by 1st learner from the pool of N_2 samples – it is assumed that 1st learner achieved accuracy higher than 50% in the first step
  - A training set with a different distribution is created for training of 2nd learner
- 3rd is trained using another N_1 unused samples that learners 1 i 2 are disagreeing on
  - Random slection from the sample pool of size N_3
- Overal required number of samples is equals $N_1+N_2+N_3$

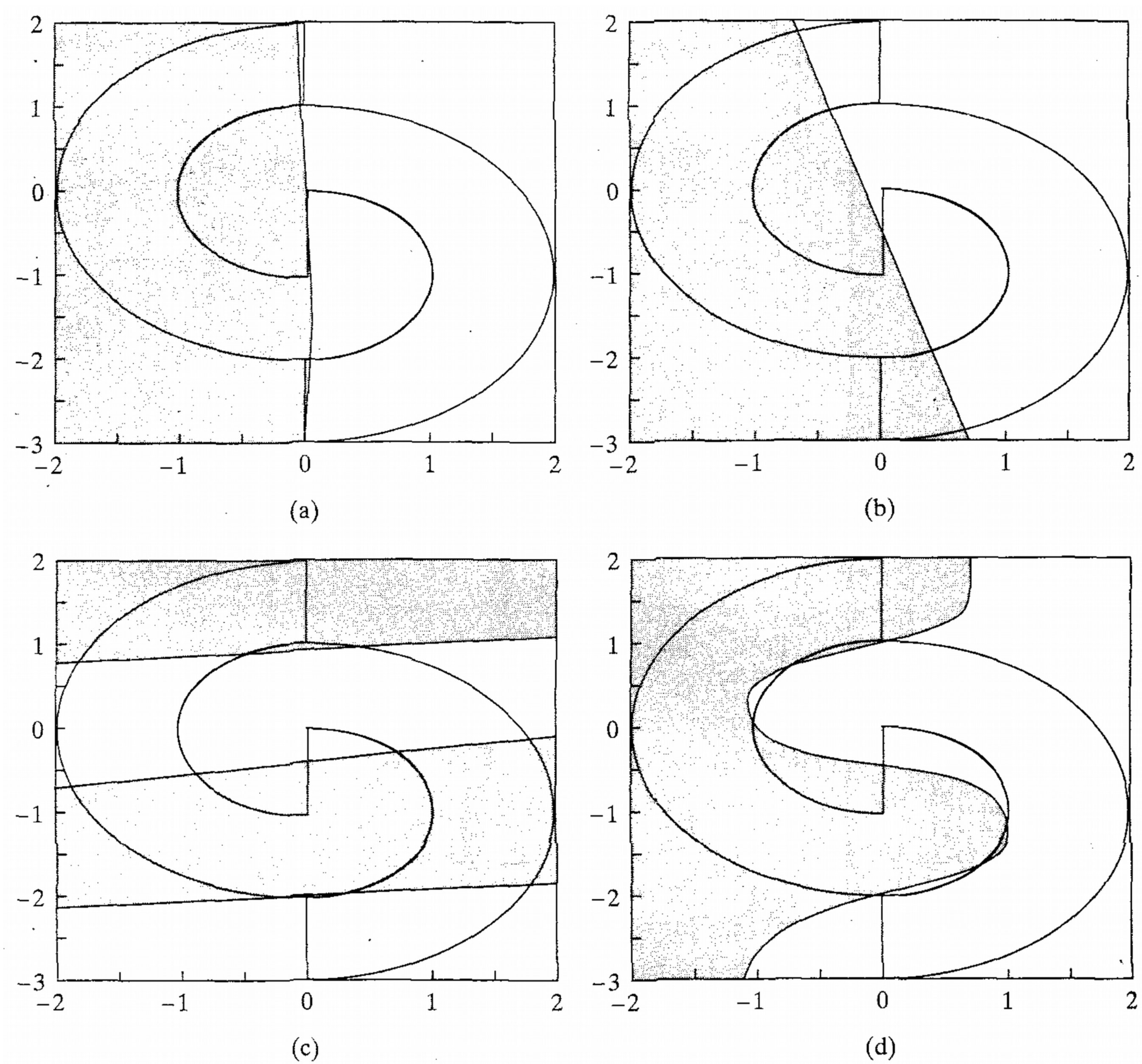
Boosting by filtering



(a) (b)

(c)

Boosting by filtering



Boosting by filtering

- What remains to be decided is the way of combining opinions of all three learners
  - Voting
  - Mean opinion
- Relatively large training data base is required
  - Potential problem

AdaBoost

- Boosting by sampling
- "recycling" the training set
- Algorithm is ADApting to errors of individual weak learners
- Required error of each weak learner is slightly bellow 50%

AdaBoost

1. Each sample is assigned an equal weight – probability equivalent
2. Weak learner is trained using random samples from the training data set that obey dataset distribution
3. **Determine the quality measure of the weak learner based on the success rate**
   - To be used for combining of weak learners
4. **Increase the weights that were wrongly classified by the weak learner – increase their probability – to generate the new distribution**
5. Select the new weak learner and go back to step 2

AdaBoost algorithm

1. Initialize distribution $D_1(i) = 1/N$
2. Train weak learner using distribution D(n)
3. Get the estimated output value F_n:**X**->Y
4. Calculate error $$e_n = \sum_{i:F_n(\boldsymbol{x}_i) \neq d_i} D_n(i)$$
5. $$\beta_n = \frac{e_n}{1 - e_n}$$
6. Update the distribution

$$D_{n+1}(i) = \frac{D_n(i)}{Z_n} * \begin{cases} \beta_n \mathit{if}\ F_n(\boldsymbol{x_i}) = d_i \\ 1\ \mathit{otherwise} \end{cases}$$

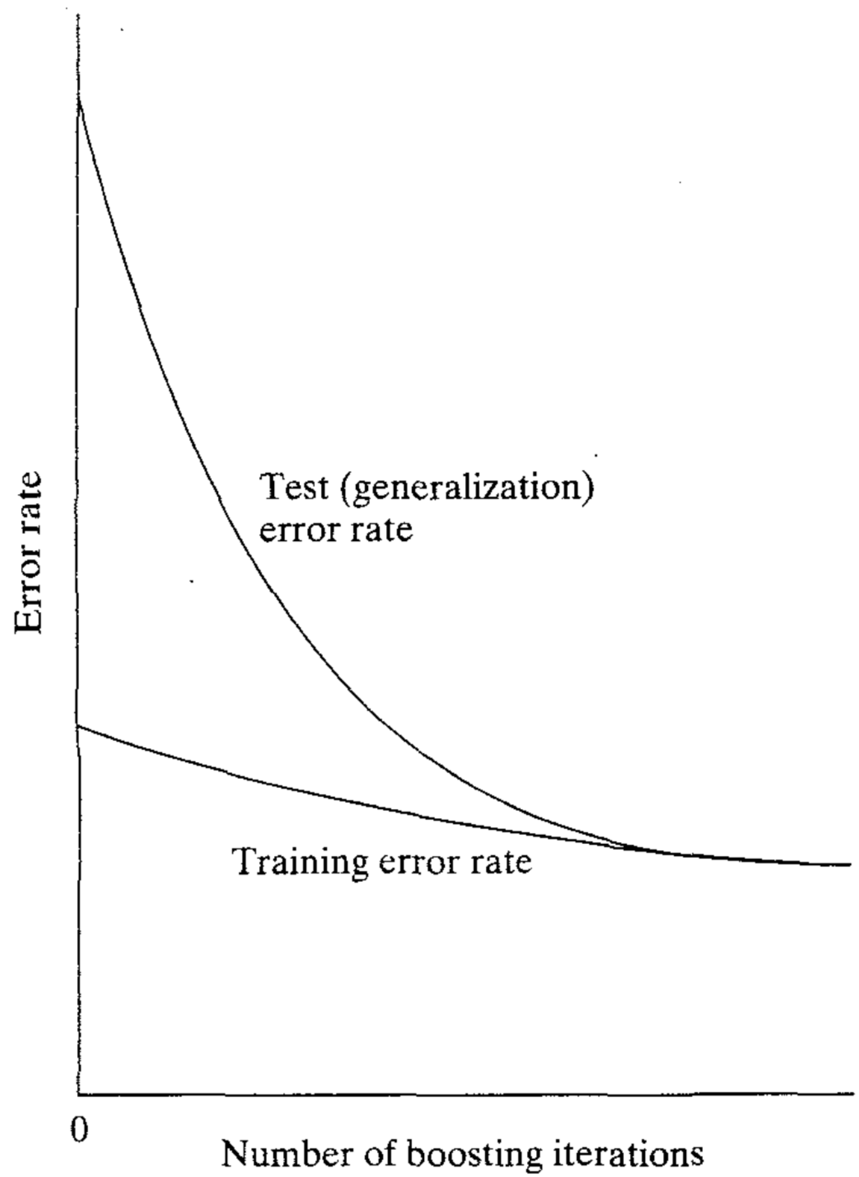
Z_n – normalization constant

Final estimate $$F_{fin}(\boldsymbol{x}) = \underset{d \in D}{\mathit{arg\,max}} \sum_{i:F_n(\boldsymbol{x}) = d} \log \frac{1}{\beta_n}$$

AdaBoost

- By adding new weak learners whose error is slightly less than 50% fainal overall training error is asympthotically approaching 0
- It has been shown that the testing error is also decreasing (better generalization), even when adding new weak learners reduces the training error only very slightly
  - Increasing the complexity of the classifier improves the generalization!
  - An interpretation is that the new weak learners increase separation margin between classes (connection with SVM) and so reduce the testing error
  - There is no risk of overfitting!

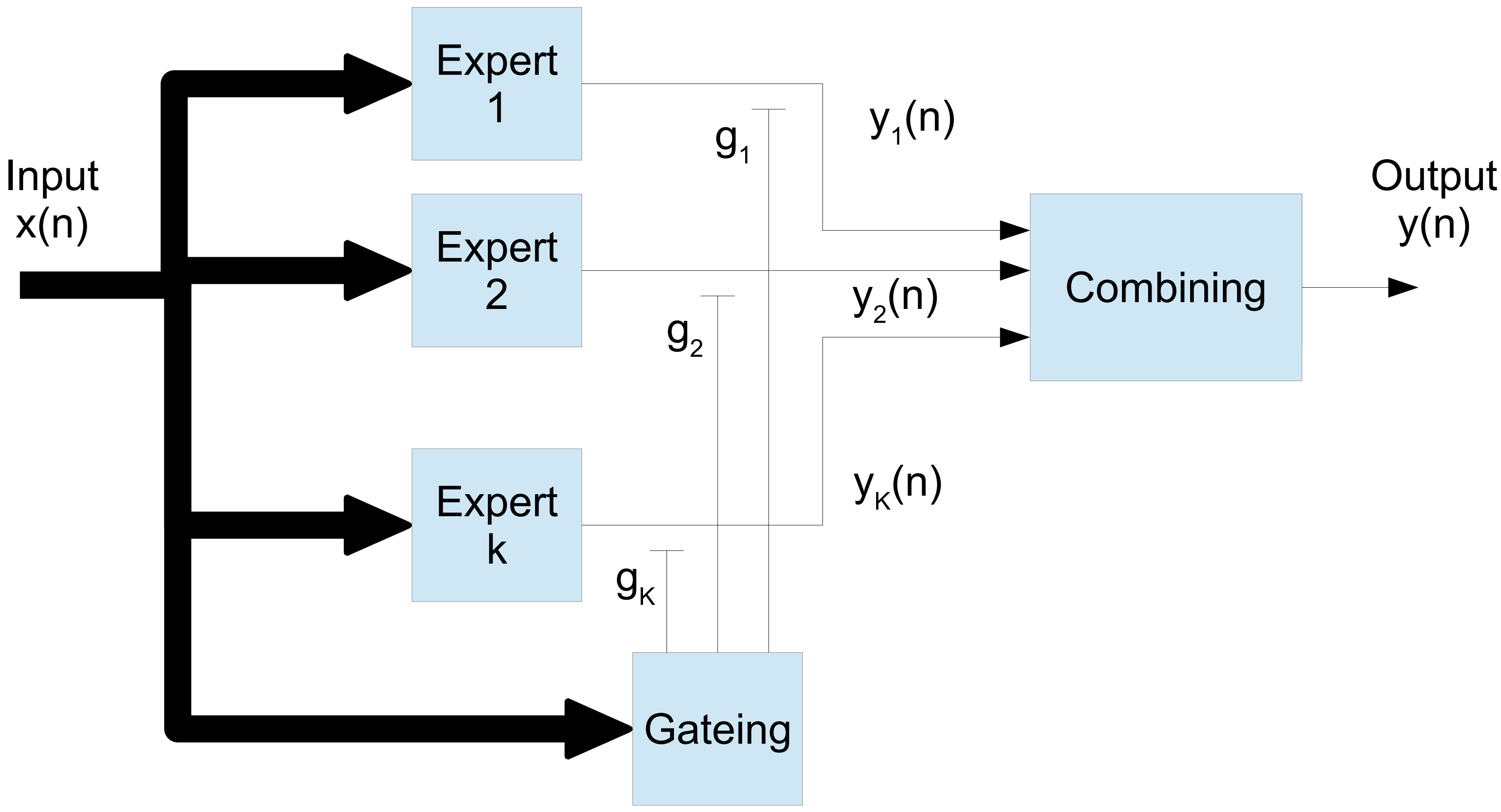
AdaBoost generalization improvement



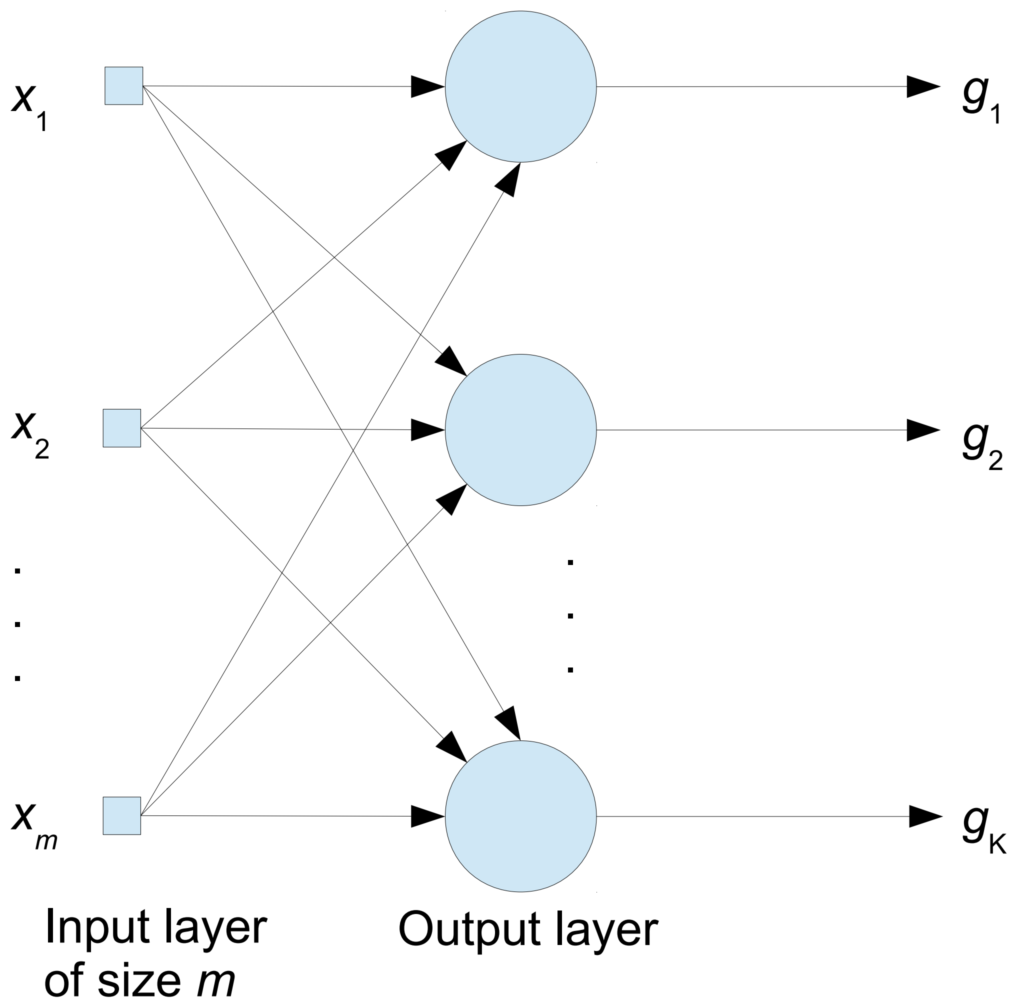
Dynamic structure

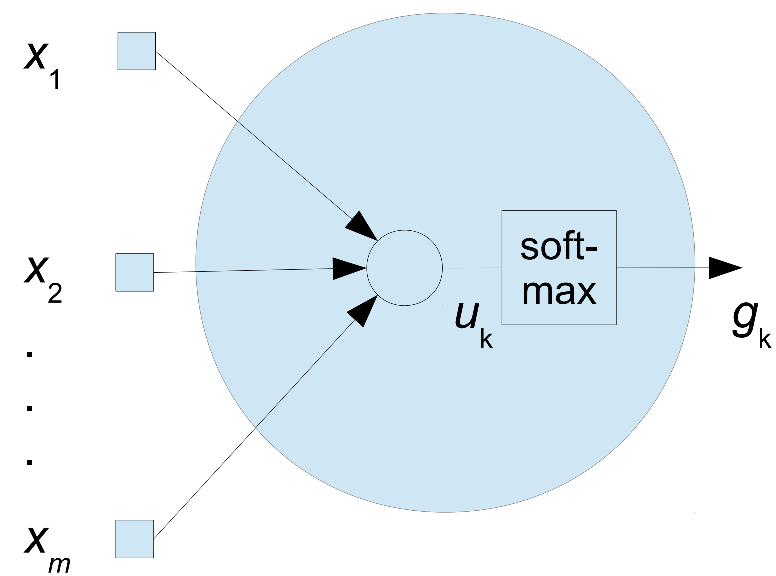
- Dynamic structure – combining of experts depends on the input data
- Experts are organizing themselves and are adapting to the input data
- Individual experts are specialized for training data subsets, but as a whole, they function well over the entire training set
- Dynamic grouping is performed by a gating network

Mixture of experts



Gating network





Mixture of experts

- Neurons of the gating network are non-linear
  - Non-linear function of **x**
- Activation function:

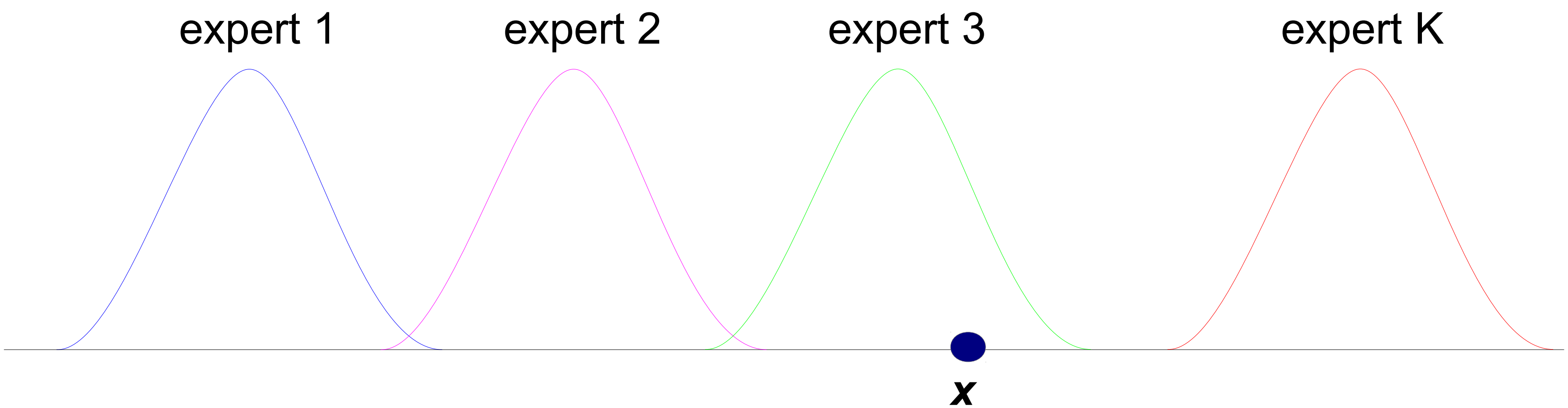
$$g_k = \frac{\exp(u_k)}{\sum_{j=1}^{K} \exp(u_j)}$$

$$u_k = \boldsymbol{a}_k^T \boldsymbol{x}$$

- Derivabile variant of the winner-takes-all – softmax

Gating network

- Can be interpeted as a classifier that maps input vector **x** to a multimodal distribution by selecting a right expert for the given mod



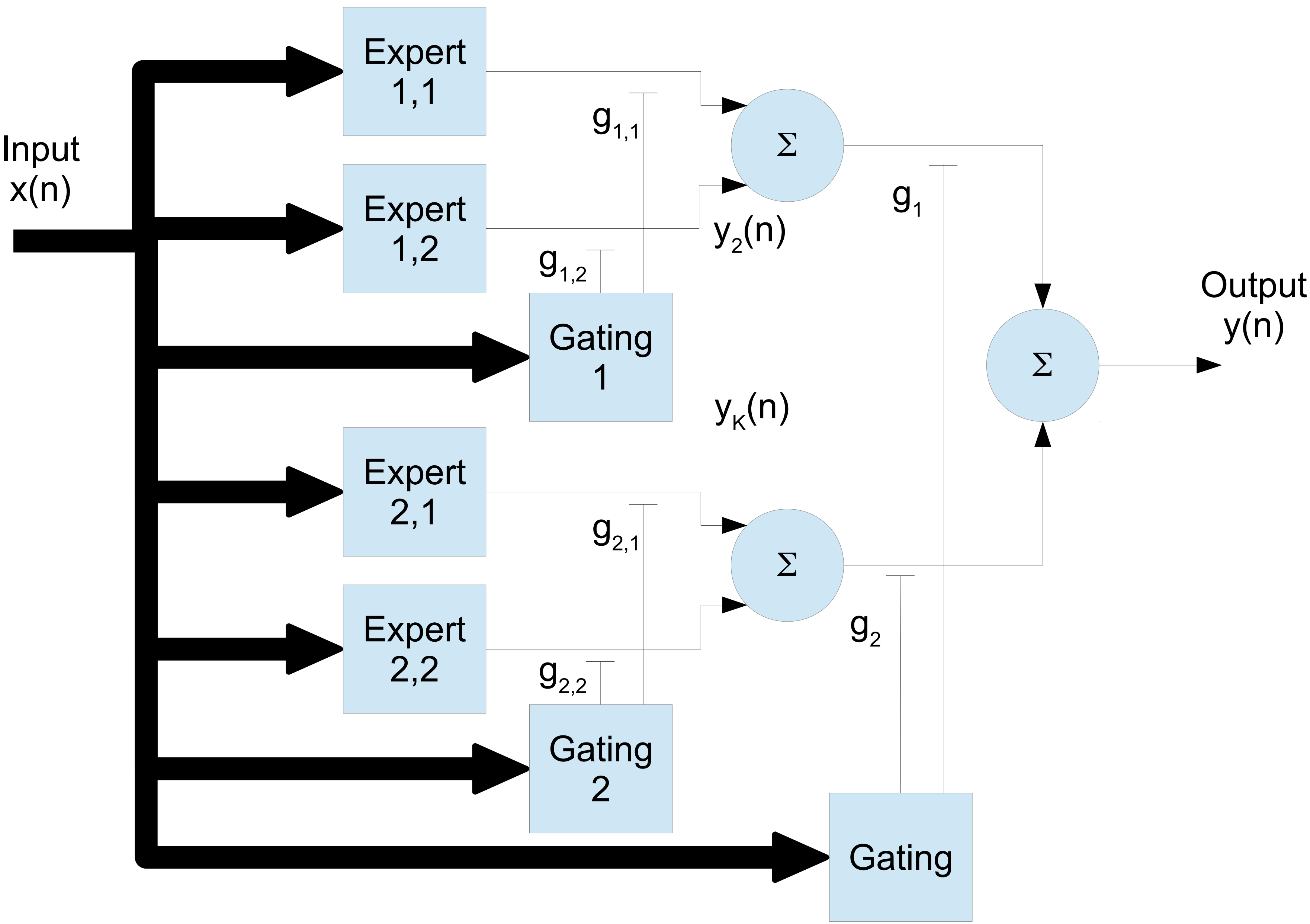
Final output

$$y = \sum_{k=1}^{K} g_k \, y_k$$

$$\sum_{k=1}^{K} g_k = 1 \qquad\qquad 0 \le g_k \le 1 \text{ za sve } k$$

- What remains is to train y_k and g_k

Hierarchical mixture of experts



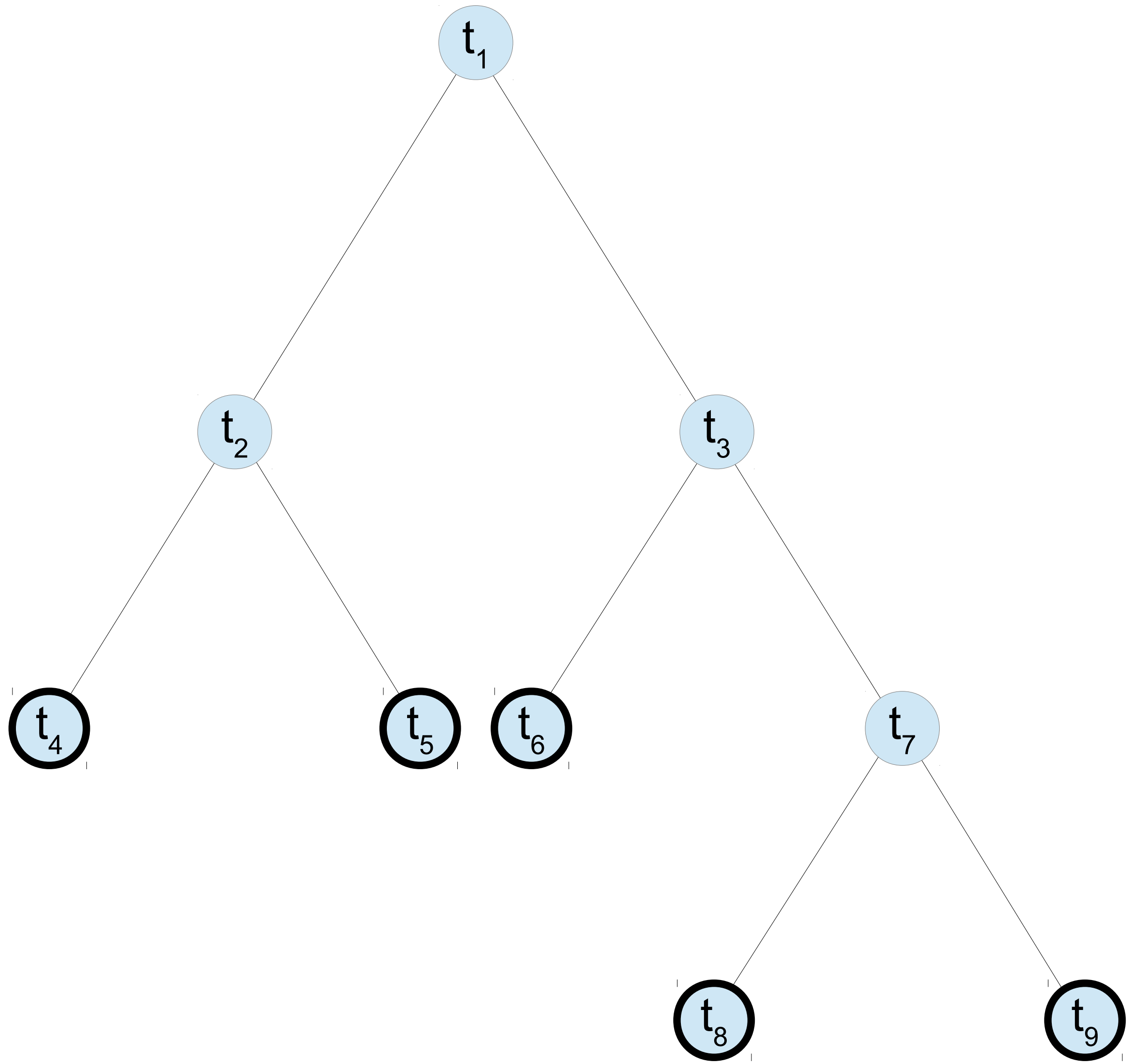
Hierarchical mixture of experts

- Ekstension to mixture of experts
- Tree like structure
- Division of input space into a nested set of subspaces
- Divide and conquer strategy – It is desirable to divide input space into subspaces
- Similar to decision tree that makes hard decisions in different regions of the input space (yes/no)
  - Hard decisions result in loss of information
  - Problem of greediness – once a wrong decision is made, ti can't be undone in the later branches

Hierarchical mixture of experts

- Before estimating parameters, the model has to be determined: number of branches and connection
- One possibility includes executing the standard decicion tree algorithm as CART (classiifcation and regression tree) to initialize HME
  - CART splits the input space by consecutive binary branching
  - Splits in CART become gating nodes in HME
  - Lower complexity of CART compared to HME is used for smart and fast selection of HME architecture
  - CART is improved using *soft* decision making

CART



CART algorithm

1. Selection of splits:

- Determine the average estimate of nod *t*

$$\bar{d}(t) = \frac{1}{N(t)} \sum_{\boldsymbol{x}_i \in t} d_i$$

- Calculate MSE for node *t*

$$E(t) = \frac{1}{N} \sum_{\boldsymbol{x}_i \in t} \left(d_i - \bar{d}(t)\right)^2$$

- Determine overal squared error for terminal tree nodes

$$E(T) = \sum_{t \in T} E(t)$$

- Split the node *t* if that minimizes *E(T)*

CART algorithm

- From all possible splits S in node t, select the split s^* that splits the node t to t_L and t_D so that

$$\Delta E(s,t) = E(t) - E(t_L) - E(t_D)$$

$$\Delta E(s^*,t) = \max_{s \in S} \Delta E(s,t)$$

- In this way $E(T)$ is minimized

CART algorithm

2. Select terminal nodes based on the predetermined threshold β

$$\max_{s \in S} \Delta E(s, t) < \beta$$

3. Least square estimation of terminal node's parameters

$$\boldsymbol{w}(t) = \boldsymbol{X}^{+} \boldsymbol{d}(t)$$

where $\mathbf{X}^{+}$ is pseudoinverse of matrix **X** that holds all inputs $x_i \in t$, $\mathbf{d}(t)$ holds all d_i from t

- Final result is minimisation of sum of squared erros

Initialization of HME

- Each tree split defines a multidimesional surface defined by

$$\boldsymbol{a}^T \boldsymbol{x} + b = 0$$

1. Apply CART to the training data
2. Set the weights of the experts to corresponding CART terminal nodes' weights **w**
3. Fro the gating network:
   a) Set the gating weight vector to point in direction orthogonal to corresponding splits
   b) Set the lenghts of weight vector to a small random values

- To estimate final model parameters use EM algorithm

Discussion

- Mixture of experts
  - Reducing error using overfitting
  - Solving the problem of data variability trough different training initializations
- Boosting
  - Reduction of error to the desired level by introducing additional weak learners
- Hierarchical mixture of experts
  - Compromise between simplicity of CART (simplicity -> easier insight in the essence of the problem) and complexity of MLP (though powerfull, black box approach provides no insight into the problem details)