Combining Multiple Learners

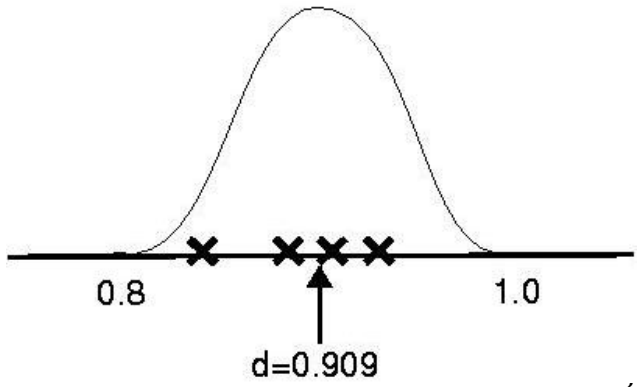
Ethem Chp. 15
Haykin Chp. 7, pp. 351-370

Overview

- Introduction
  - Rationale
- Combination Methods
  - Static Structures
    - Ensemble averaging (Sum,Product,Min rule)
    - Bagging
    - Boosting
    - Error Correcting Output Codes
  - Dynamic structures
    - Mixture of Experts
    - Hierarchical Mixture of Experts

Motivation

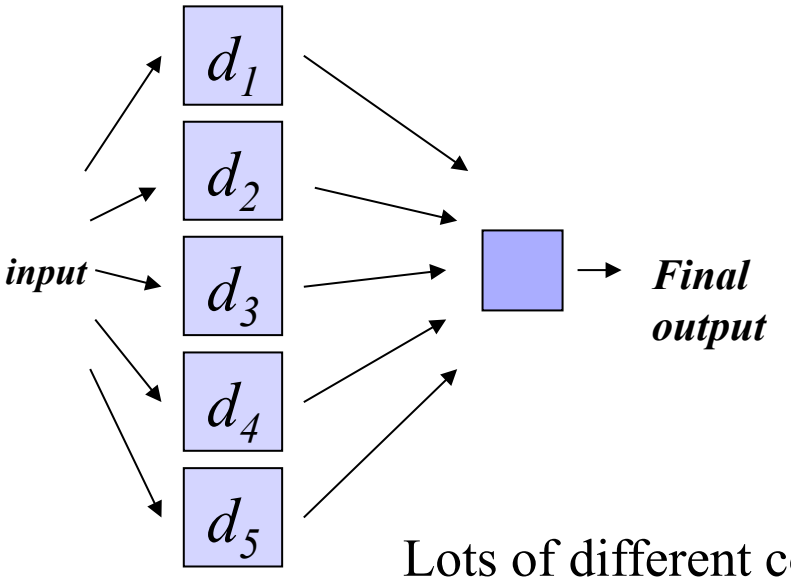
- When designing a learning machine, we generally make some choices:
  - *parameters of machine, training data, representation, etc…*
- This implies some sort of variance in performance
- Why not keep all machines and average?
- ...



Rationale

- No Free Lunch thm: “There is no algorithm that induces the most accurate learner in any domain, all the time.”
  - http://www.no-free-lunch.org/
- Generate a group of base-learners which when combined has higher accuracy
- Different learners use different
  - Algorithms: making different assumptions
  - Hyperparameters: e.g number of hidden nodes in NN, k in k-NN
  - Representations: diff. features, multiple sources of information
  - Training sets: small variations in the sets or diff. subproblems

Reasons to Combine Learning Machines



Lots of different combination methods:

Most popular are *averaging* and *majority voting*.

Intuitively, it seems as though it should work.

We have parliaments of people who vote, and that works …

We *average* guesses of a quantity, and we'll probably be closer…

$$f_{com} = vote(f_i, f_j, f_k, f_l, f_m)$$

Probability of error in case of voting is all the combinations of classifier answers where more than half of the classifiers make a mistake:

Binomial theorem says…

$$P(error) = \sum_{k=\frac{N}{2}+1}^{N} \binom{N}{k} p^k (1-p)^{N-k}$$

. . .but only if they are independent!

What is the implication?

Use many experts and take a vote

A related theory paper…

Tumer & Ghosh 1996

"Error Correlation and Error Reduction in Ensemble Classifiers"

(makes some assumptions, like equal variances)

- Remember the Bayesian perspective (if outputs are posterior probabilities):

$$P(C_i \mid x) = \sum_{\text{all models } \mathrm{M}_j} P(C_i \mid x, \mathrm{M}_j) P(\mathrm{M}_j)$$

- We want the base learners to be
  - Complementary
    - what if they are all the same or very similar
  - Reasonably accurate
    - but not necessarily very accurate

Types of Committee Machines

- **Static structures:** the responses of several *experts* (individual networks) are combined in a way that **does not** involve the input signal.
  - *Ensemble averaging*
  - *Boosting*
- **Dynamic structures**: the input signal actuates the mechanism that combines the responses of the experts.
  - *Mixture of experts*
  - *Hierarchical mixture of experts*

Overview

- Introduction
  - Rationale
- Combination Methods
  - Static Structures
    - Ensemble averaging (Sum,Product,Min rule)
    - Bagging
    - Boosting
    - Error Correcting Output Codes
  - Dynamic structures
    - Mixture of Experts
    - Hierarchical Mixture of Experts

Ensemble Averaging >Voting

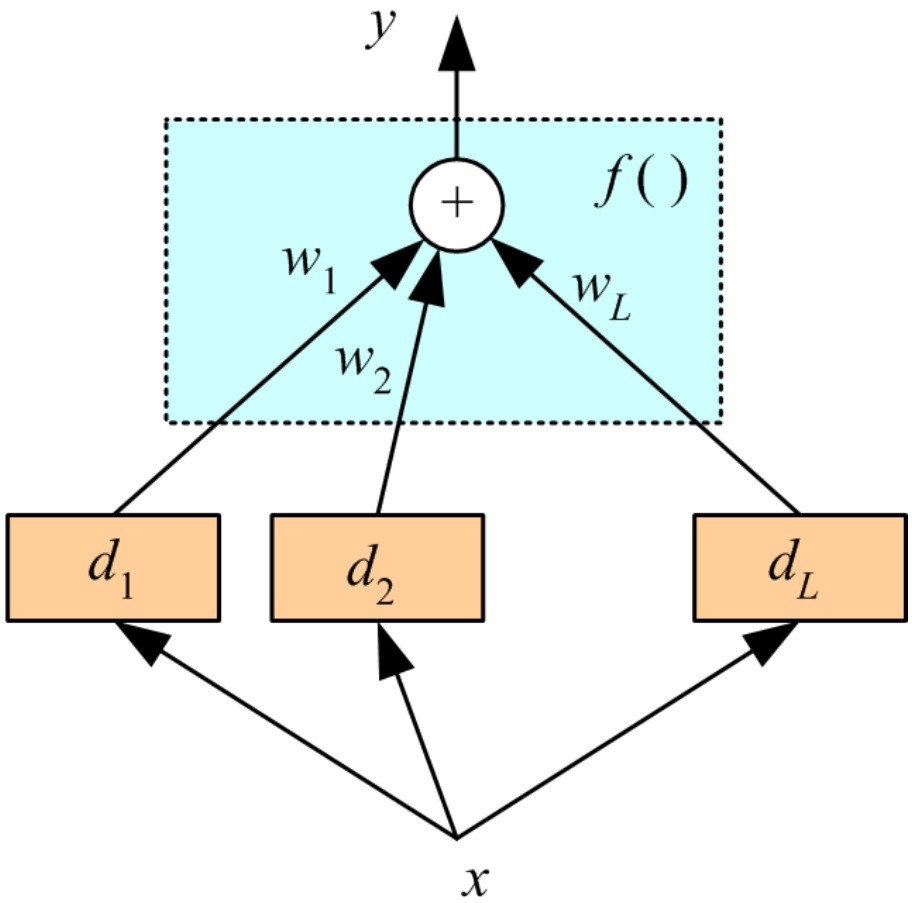
- Regression

$$y = \sum_{j=1}^{L} w_j d_j$$

$$w_j \geq 0 \text{ and } \sum_{j=1}^{L} w_j = 1$$

- Classification

$$y_i = \sum_{j=1}^{L} w_j d_{ji}$$



Ensemble Averaging >Voting

- Regression

$$y = \sum_{j=1}^{L} w_j d_j$$

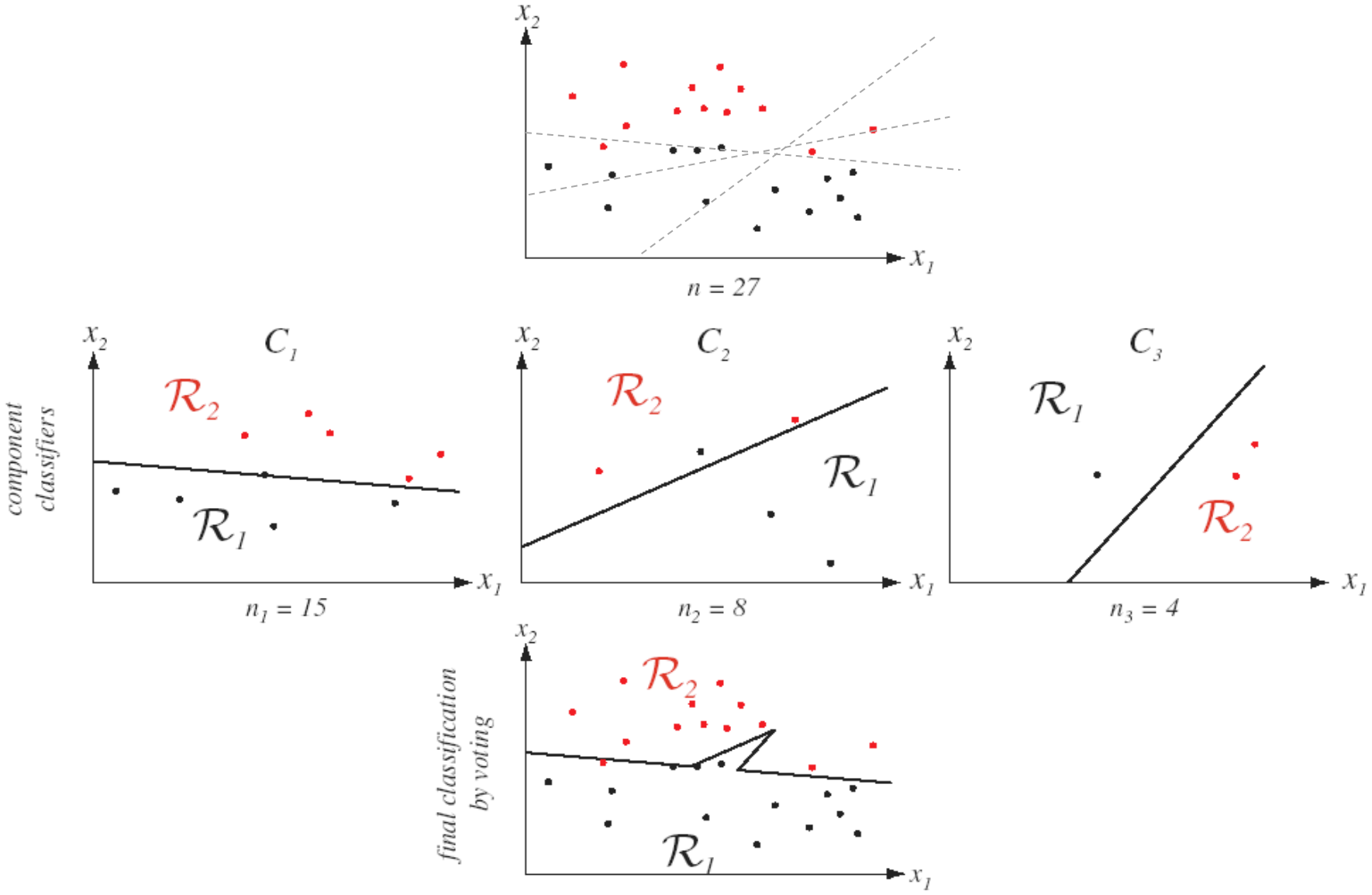
$$w_j \geq 0 \text{ and } \sum_{j=1}^{L} w_j = 1$$

w_j=1/L or better:

w_j proportional to error rate of classifier:
Learned over a validation set

- Classification

$$y_i = \sum_{j=1}^{L} w_j d_{ji}$$



Ensemble Averaging >Voting

If we use a committee machine f_{com} whose output is:

$$f_{com} = \frac{1}{M}\sum_{i=1}^{i=M} d_i$$

Error of combination is guaranteed to be lower than the *average error:*

$$(f_{com} - t)^2 = \frac{1}{M}\sum_i (d_i - t)^2 - \frac{1}{M}\sum_i (d_i - f_{com})^2$$

positive number

(Krogh & Vedelsby 1995)

- Similarly, we can show that if d_j are iid:

$$\text{Bias}^2: \quad (E_D(d) - f)^2$$
$$\text{Variance}: \quad E_D[(E_D(d) - d)^2]$$

$$E[f_{com}] = E\left[\sum_j \frac{1}{L} d_j\right] = \frac{1}{L} L \cdot E[d_j] = E[d_j]$$

$$\text{Var}(f_{com}) = \text{Var}\left(\sum_j \frac{1}{L} d_j\right) = \frac{1}{L^2} \text{Var}\left(\sum_j d_j\right) = \frac{1}{L^2} L \cdot \text{Var}(d_j) = \frac{1}{L} \text{Var}(d_j)$$

- Bias does not change, variance decreases by 1/*L*

  => Average over models with low bias and high variance

Ensemble Averaging

What we can exploit from this fact:

- Combine multiple experts with the same bias and variance, using ensemble-averaging
  - the bias of the ensemble-averaged system would the **same as the bias of one of the individual experts**
  - the variance of the ensemble-averaged system would be **less than the variance of one of the individual experts.**

- We can also purposefully overtrain individual networks, the variance will be reduced due to averaging

Ensemble methods

- Product rule
  - Assumes that representations used by by different classifiers are conditionally independent
- **Sum rule (voting with uniform weights)**
  - Further assumes that posteriors of class probabilities are close to the class priors
  - Very successful in experiments, despite very strong assumptions
    - Committee machine less sensitive to individual errors
- Min rule
  - Can be derived as an approximation to the product/sum rule
- Max rule

- The respective assumptions of these rules are analyzed in Kittler et al. 1998.
- The sum-rule is thought to be the best at the moment

- We have shown that ensemble methods have the same bias but lower variance, compared to individual experts.

- Alternatively, we can analyze the **expected error** of the ensemble averaging committee machine to show that it will be **less than the average of the errors made by each individual network**
  - (see Bishop pp.365-66 for the derivation and Haykin experiment on pp. 355-56 given in the next slides)

Computer Experiment
Haykin: pp. 187-198 and 355-356

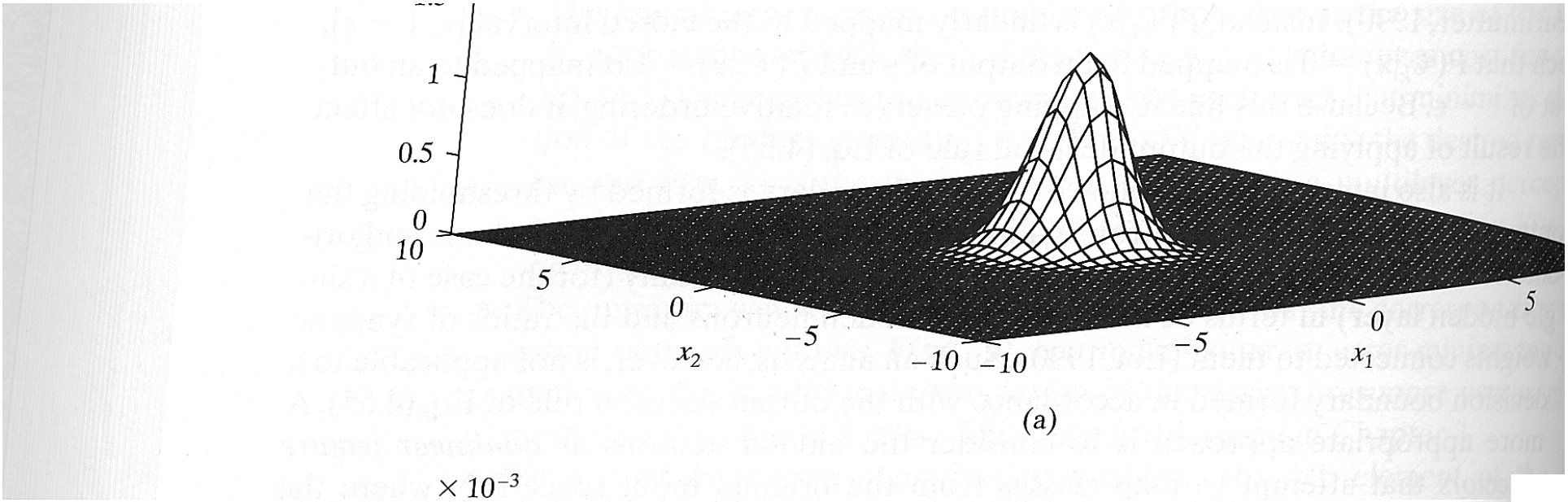
- C1: N([0,0], 1)
- C2: N([2,0], 4)

- Bayes criterion for optimum decision boundary:
  - P(C1|x) > P(C2|x)

- Bayes decision boundary:
  - circular, centered at [-2/3, 0]

- Probability of correct classification by Bayes classifier= 0.81%
  - $1\text{-}P_{error} = 1 - (P(e|C1)\ p(C1) + P(e|C2)P(C2))$

- Simulation results with diff. networks (all with 2 hidden nodes):
  - average of 79.4 and σ = 0.44 over 20 networks



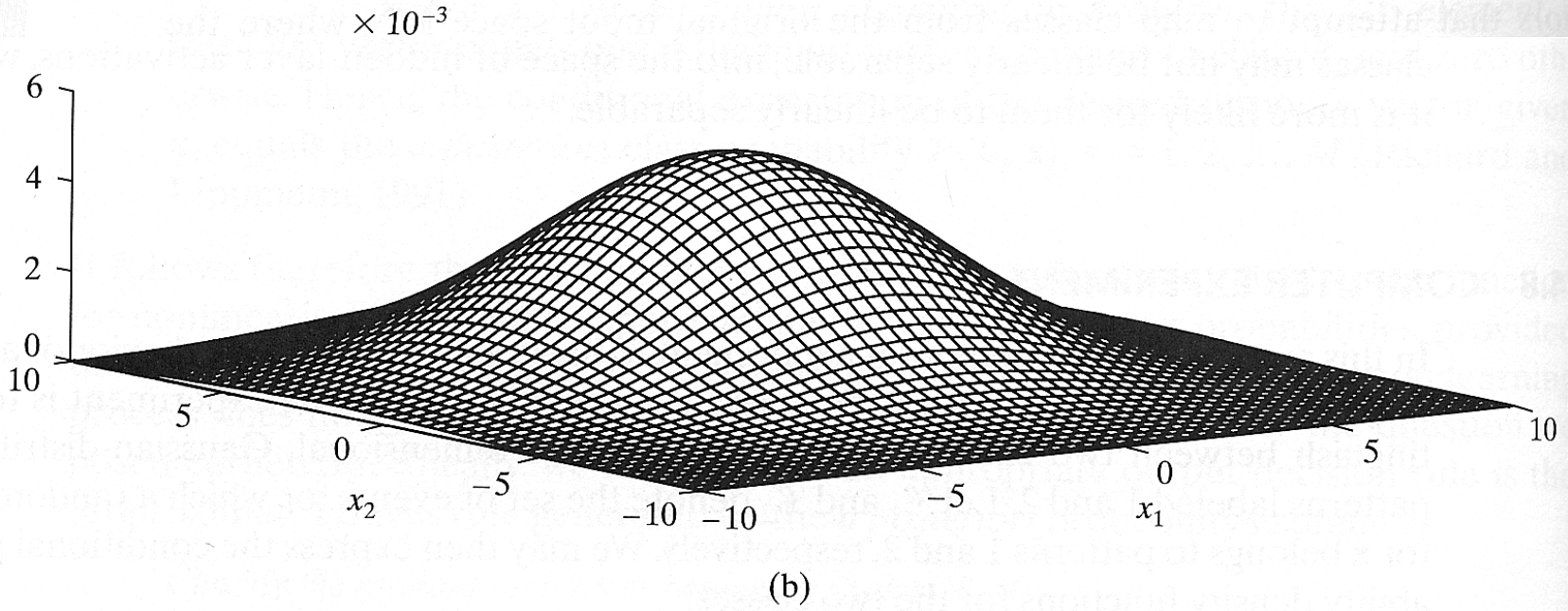


FIGURE 4.13 (a) Probability density function $f_x(\mathbf{x}|\mathscr{C}_1)$; (b) Probability density function $f_x(\mathbf{x}|\mathscr{C}_2)$.

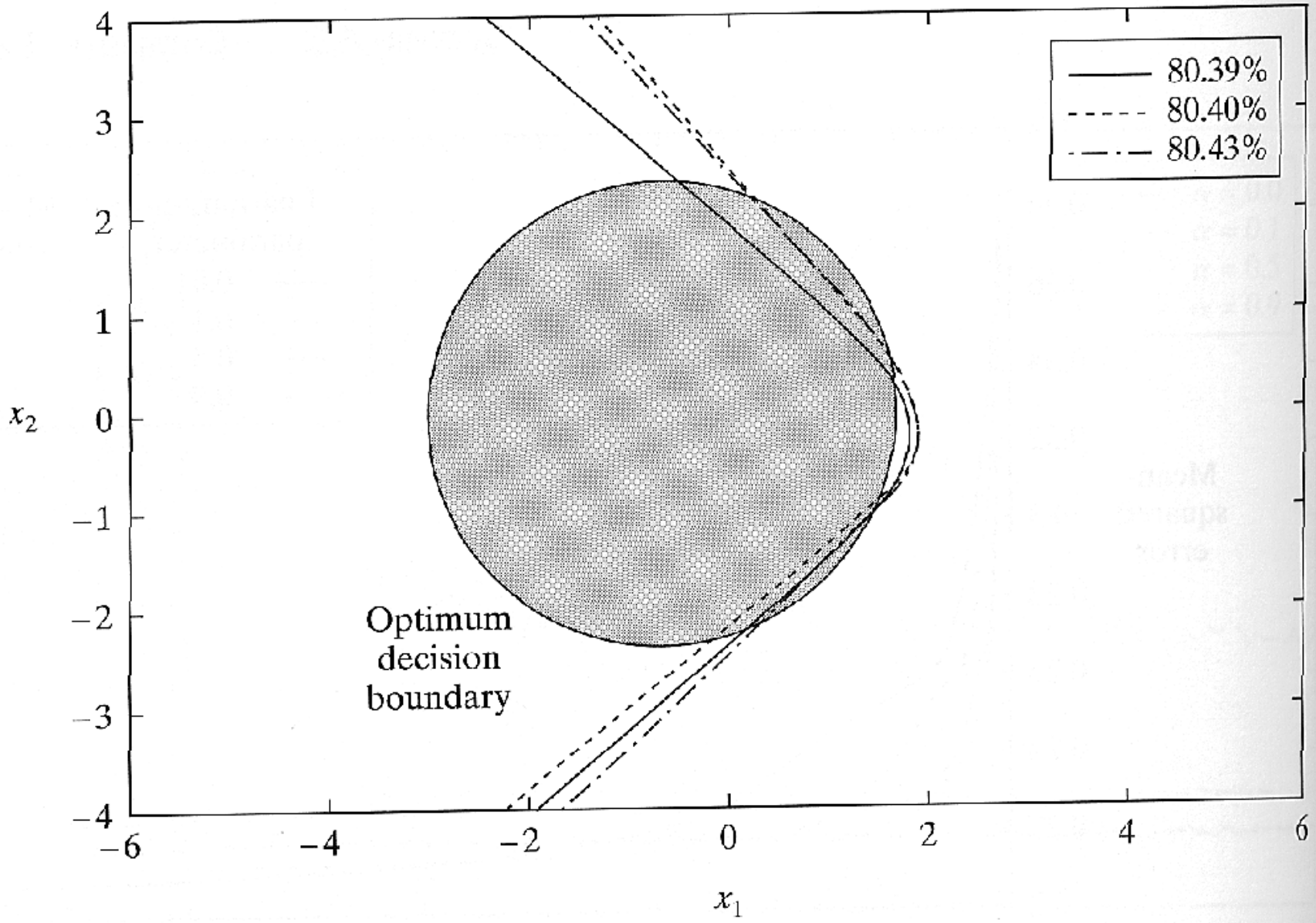


FIGURE 4.17A Plot of three "best" decision boundaries for the classification accuracies: 80.39, 80.40, and 80.43%.

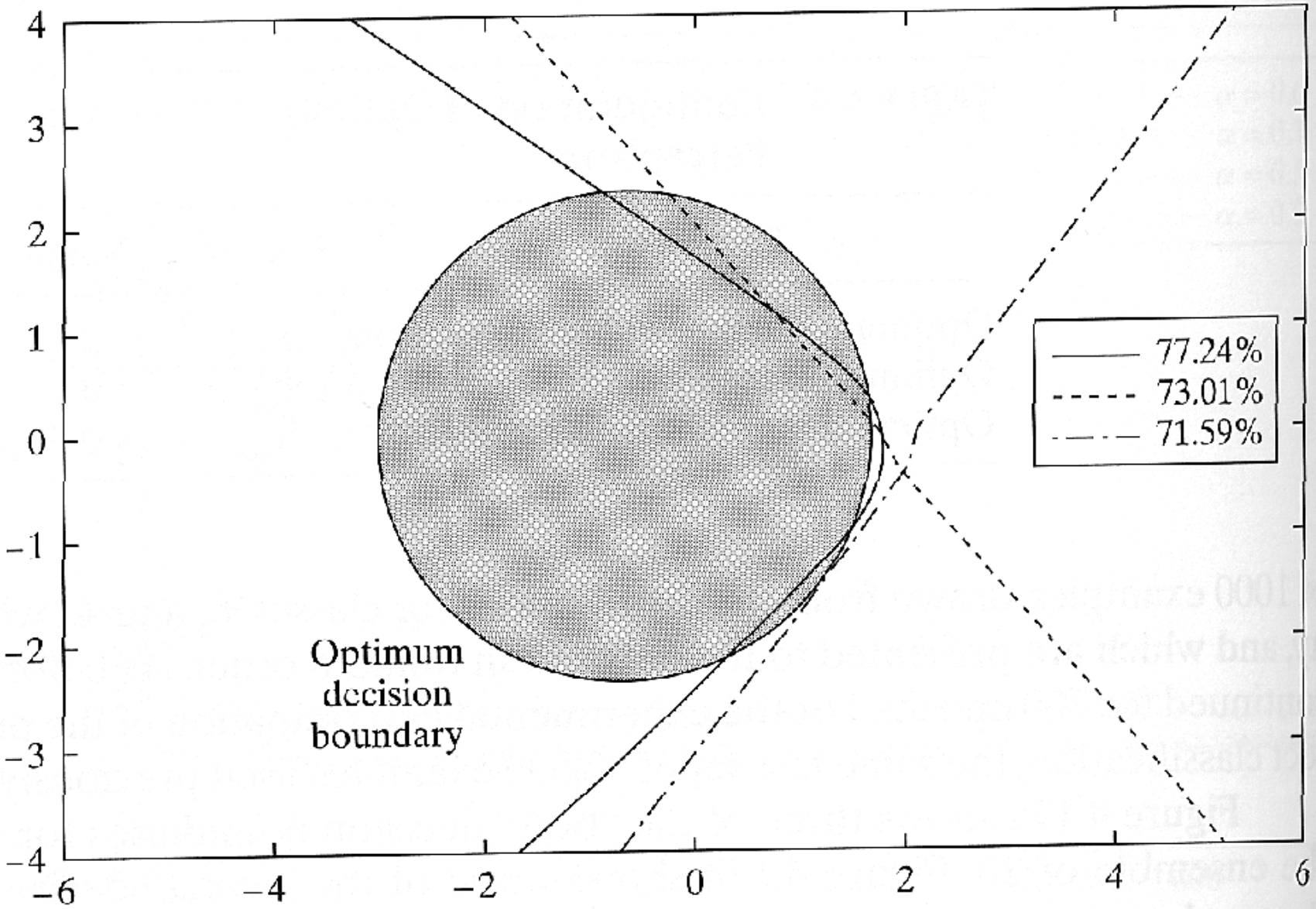


FIGURE 4.17B Plot of three "poorest" decision boundaries for the following classification accuracies: 77.24, 73.01, and 71.59%.

- Combining the outputs of 10 networks, the ensemble average achieves an expected error (ε_D) less than the expected value of the average error of the individual networks, over many trials with different data sets.
  - 80.3% versus 79.4% (average)
  - 1% diff.

TABLE 7.1 Classification Performances of Individual Experts i in a Committee Machine

Expert	Correct classificat percentage
Net 1	80.65
Net 2	76.91
Net 3	80.06
Net 4	80.47
Net 5	80.44
Net 6	76.89
Net 7	80.55
Net 8	80.47
Net 9	76.91
Net 10	80.38

Avg. 79.4

Overview

- Introduction
  - Rationale
- Combination Methods
  - Static Structures
    - Ensemble averaging (Voting)
    - **Bagging**
    - Boosting
    - Error Correcting Output Codes
  - Dynamic structures
    - Mixture of Experts
    - Hierarchical Mixture of Experts

Ensemble Methods > Bagging

Voting method where base-learners are made different by training over slightly different training sets

Bagging (Bootstrap Aggregating) - Breiman, 1996

take a training set D, of size N

for each network / tree / k-nn / etc…

- build a new training set by sampling N examples, randomly with replacement, from D
- train your machine with the new dataset

end for

output is average/vote from all machines trained

- Resulting base-learners are similar because they are drawn from the same original sample
- Resulting base-learners are slightly different due to chance
- Not all data points will be used for training
  - Waste of training set

Ensemble Methods > Bagging

	Single net	Simple ensemble	Bagging
breast cancer	3,4	3,5	3,4
glass	38,6	35,2	33,1
diabetes	23,9	23	22,8

Error rates *on UCI datasets (10-fold cross validation)*

Source: Opitz & Maclin, 1999

- Bagging is better in 2 out of 3 cases and equal in the third.
- Improvements are clear over single experts and even better than a simple ensemble.
- **Bagging is suitable for unstable learning algorithms**
  - Unstable algorithms change significantly due to small changes in the data
    - MLPs, decision trees

Overview

- Introduction
  - Rationale
- Combination Methods
  - Static Structures
    - Ensemble averaging (Voting…)
    - Bagging
    - Boosting
    - Error Correcting Output Codes
  - Dynamic structures
    - Mixture of Experts
    - Hierarchical Mixture of Experts

Ensemble Methods > Boosting

- In Bagging, generating *complementary* base-learners is left to chance and unstability of the learning method

- Boosting – Schapire & Freund 1990
  - Try to generate **complementary** *weak* base-learners by training the next learner on the mistakes of the previous ones
  - Weak learner: the learner is required to perform only **slightly better than random**
    - $\varepsilon < ½$
  - Strong learner: arbitrary accuracy with high probability (PAC)
  - **Convert a weak learning model to a strong learning model by "boosting" it**
    - Kearns and Valient (1988) posed the question, "are the notions of strong and weak learning equivalent?"
    - Schapire (1990) and Freund (1991) gave the first constructive proof.

- The Boosting model consists of component classifiers which we call "experts“ and trains each expert on data sets with different distributions.

- There are three methods for implementing Boosting:
  - Filtering: Filtering the training examples assumes a large source of examples with the examples being discarded or kept during training.
  - Subsampling: Subsampling works with training examples of fixed size which are "resampled" according to a probability distribution during training.
  - Re-weighting: Re-weighting works with a fixed training sample where the examples are "weighted" by a weak learning algorithm.

Boosting – General Approach

take a training set D, of size N

do M times

train a network on D

find all examples in D that the network gets wrong

emphasize those patterns, de-emphasize the others, in a new dataset D2

set D=D2

loop

output is average/vote from all machines trained

General method *– different types in literature, by filtering, sub-sampling or re-weighting, see Haykin Ch.7 for details if you are interested.*

Boosting by Filtering

Original Boosting algorithm, Shapire 1990:

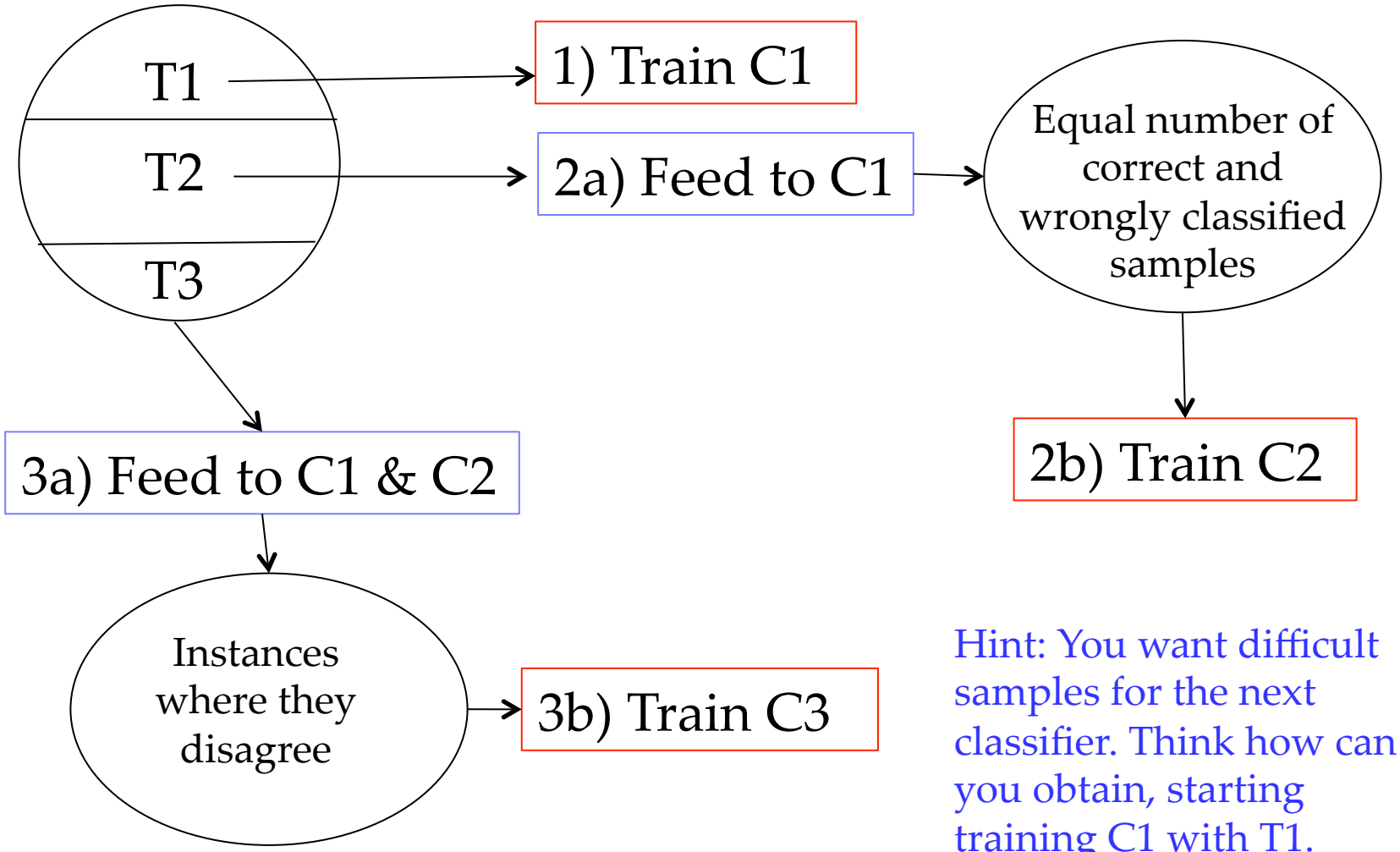
Training:

- **Divide T into 3 sets: T1, T2 and T3**
- **Use T1 to train c1**
- **Feed T2 into c1 and get estimated labels**
  - **Take equal number of correctly & wrongly classified instances in T2 by c1, to train classifier c2**
    - online version possible (H/T – wait for correct or misclassified) – Haykin pp358
- **Feed T3 into c1 and c2**
  - **Add instances where they disagree to the third training set**

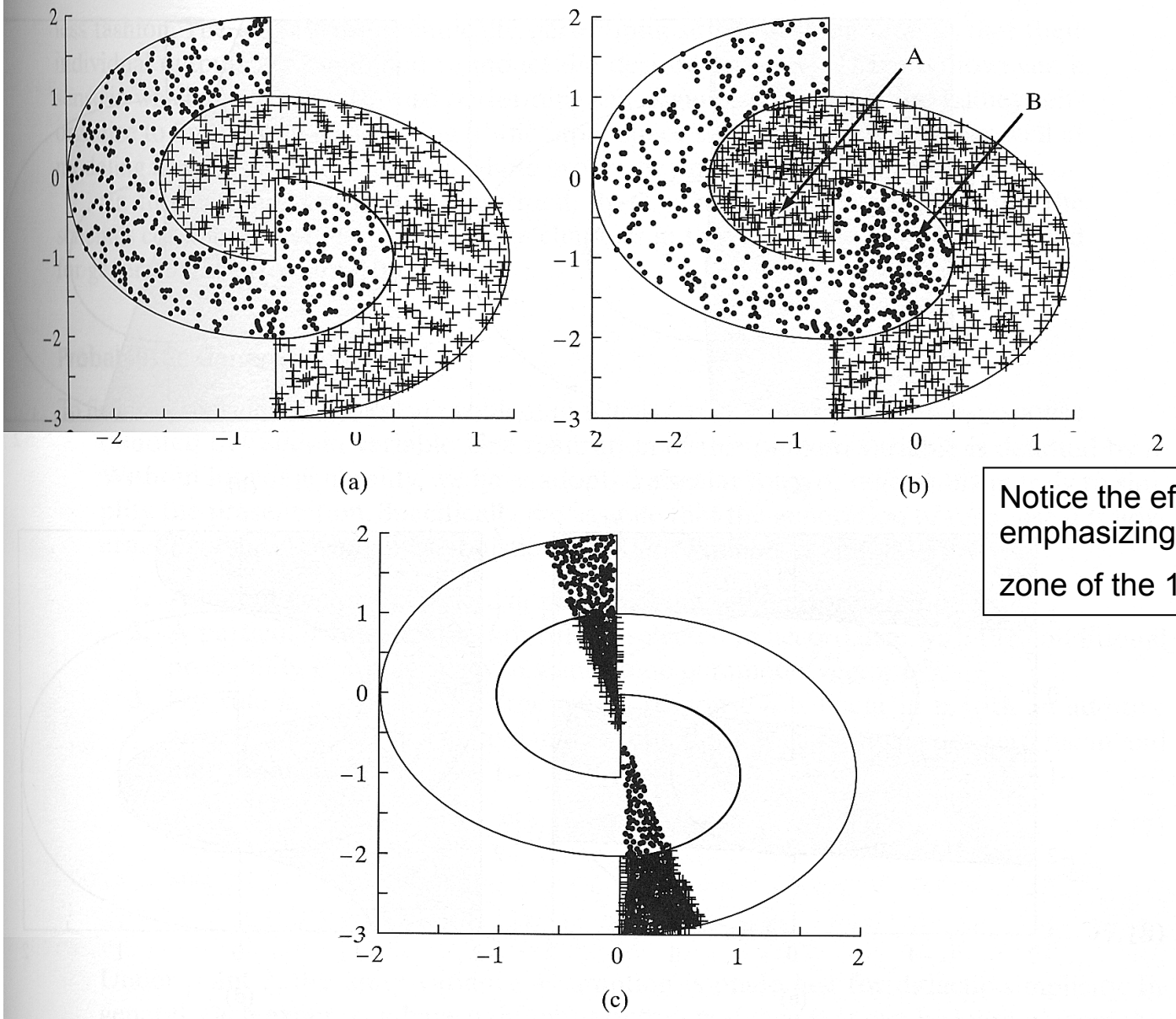
Testing:

- Feed instance to c1 and c2
  - If they agree, take the decision
  - If they dont agree, use c3's decision (=majority decision)

  - **Useful when there are plenty of data**



Hint: You want difficult samples for the next classifier. Think how can you obtain, starting training C1 with T1.



Notice the effect of emphasizing the error

zone of the 1st classifier

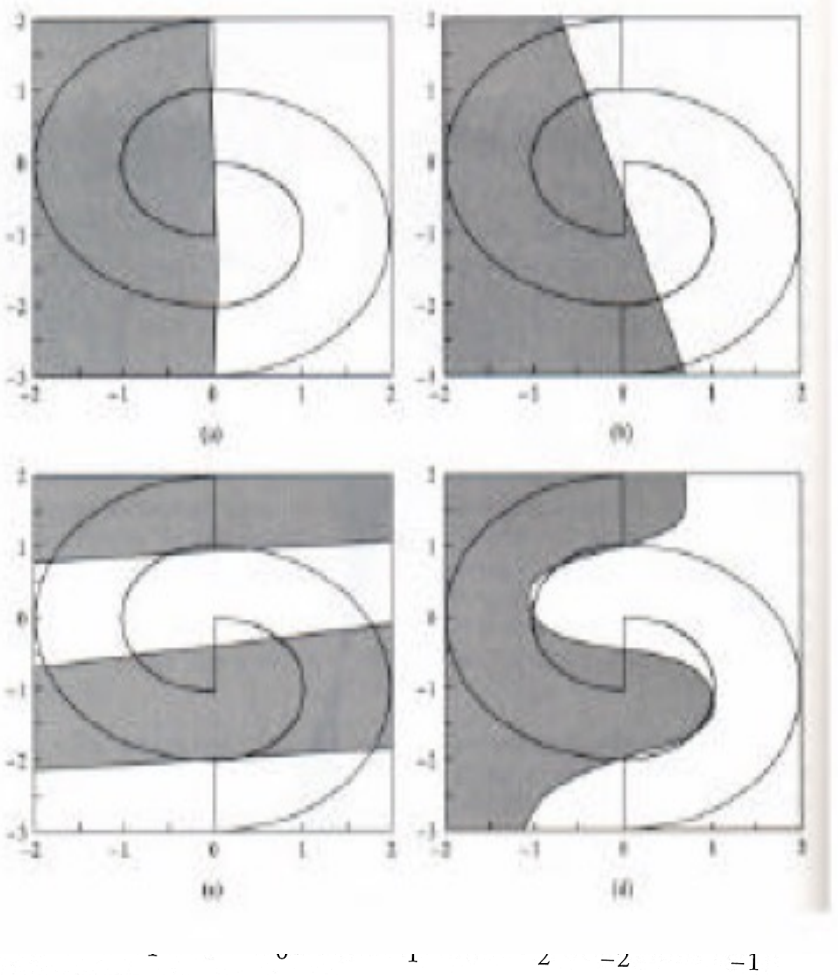
FIGURE 7.6 Scatter plots for expert training in computer experiment on boosting: (a) Expert 1. (b) Expert 2. (c) Expert 3.

Three 2-5-2 MLP networks

Accuracies:

Expert 1 : 75.15 percent
Expert 2 : 71.44 percent
Expert 3 : 68.90 percent

Committee Machine:
91.79% correct



(c)

FIGURE 7.7 Decision boundaries formed by the differen
experiment. (a) Expert 1. (b) Expert 2. (c) Expert 3. (d) Entire c

Boosting by Filtering – ctd.

- Individual experts concentrate on **hard-to-learn areas**
- Training data for each network comes from a **different** distribution
- Output of individual networks can be combined by voting or addition (was found to be better in one work)
- **Requires large amount of training data**
  - Solution: Adaboost (Shapire 1996)
  - Variant of boosting by filtering; short for adaptive boosting

AdaBoost (ADAptive BOOSTing)

- Modify the probabilities of drawing an instance x^t for a classifier j, based on the probability of error of c_j
  - For the next classifier:
    - if pattern x^t is correctly classified, its probability of being selected decreases
    - if pattern x^t is NOT correctly classified, its probability of being selected increases

- All learners must have error less than ½
  - simple, weak learners
  - if not, stop training (the problem gets more difficult for next classifier)

- **AdaBoost Algorithm**

1. The initial distribution is uniform over the training sample.
2. The next distribution is computed by multiplying the weight of example *i* by some number $\beta \in (0,1]$ if the weak hypothesis classifies the input vector correctly;

   otherwise, the weight is unchanged.
3. The weights are normalized.
4. The final hypothesis is a weighted vote of the L weak classifiers

ADVANCED

AdaBoost

Generate a sequence of base-learners each focusing on previous one's errors

(Freund and Schapire, 1996)

Training:

For all $\{x^t, r^t\}_{t=1}^N \in \mathcal{X}$, initialize $p_1^t = 1/N$

For all base-learners $j = 1, \ldots, L$

Randomly draw $\mathcal{X}_j$ from $\mathcal{X}$ with probabilities p_j^t

Train d_j using $\mathcal{X}_j$

For each (x^t, r^t), calculate $y_j^t \leftarrow d_j(x^t)$

Calculate error rate: $\epsilon_j \leftarrow \sum_t p_j^t \cdot 1(y_j^t \neq r^t)$

If $\epsilon_j > 1/2$, then $L \leftarrow j - 1$; stop

$\beta_j \leftarrow \epsilon_j / (1 - \epsilon_j)$

For each (x^t, r^t), decrease probabilities if correct:

If $y_j^t = r^t$ $p_{j+1}^t \leftarrow \beta_j p_j^t$ Else $p_{j+1}^t \leftarrow p_j^t$

Normalize probabilities:

$Z_j \leftarrow \sum_t p_{j+1}^t$; $p_{j+1}^t \leftarrow p_{j+1}^t / Z_j$

Testing:

Given x, calculate $d_j(x), j = 1, \ldots, L$

Calculate class outputs, $i = 1, \ldots, K$:

$$y_i = \sum_{j=1}^{L} \left(\log \frac{1}{\beta_j} \right) d_{ji}(x)$$

Ensemble Methods > Boosting by filtering

	Single net	Simple ensemble	Bagging	AdaBoost
breast cancer	3.4	3.5	3.4	4
glass	38.6	35.2	33.1	31.1
diabetes	23.9	23	22.8	23.3

Error rates on UCI datasets (10-fold cross validation)
Source: Opitz & Maclin, 1999

- These are just some examples. I have included them to give some concrete examples on real datasets.

- Training error falls with each boosted classifier

- Generalization error also tends to fall:
  - Improved generalization performance over 22 benchmark problems, equal accuracy in one, worse accuracy in 4 problems [Shapire 1996].

  - Shapire et al. explain the success of AdaBoost due to its property of increasing the margin, with the analysis involving the confidence of the individual classifiers [Shapire 1998].

Overview

- Introduction
  - Rationale
- Combination Methods
  - Static Structures
    - Ensemble averaging (Voting)
    - Bagging
    - Boosting
    - Error Correcting Output Codes
  - Dynamic structures
    - Mixture of Experts
    - Hierarchical Mixture of Experts

Error-Correcting Output Codes

- *K* classes; *L* sub-problems (Dietterich and Bakiri, 1995)
- Code matrix **W** specifies each dichotomizer's task in its columns, where rows are the classes

- One per class

  $L=K$

$$\mathbf{W} = \begin{bmatrix} +1 & -1 & -1 & -1 \\ -1 & +1 & -1 & -1 \\ -1 & -1 & +1 & -1 \\ -1 & -1 & -1 & +1 \end{bmatrix}$$

- Pairwise

  $L=K(K-1)/2$

  not feasible for large K

$$\mathbf{W} = \begin{bmatrix} +1 & +1 & +1 & 0 & 0 & 0 \\ -1 & 0 & 0 & +1 & +1 & 0 \\ 0 & -1 & 0 & -1 & 0 & +1 \\ 0 & 0 & -1 & 0 & -1 & -1 \end{bmatrix}$$

- Full code $L=2^{(K-1)}-1$

$$\mathbf{W} = \begin{bmatrix} -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & +1 & +1 & +1 & +1 \\ -1 & +1 & +1 & -1 & -1 & +1 & +1 \\ +1 & -1 & +1 & -1 & +1 & -1 & +1 \end{bmatrix}$$

- With reasonable L, find **W** such that the Hamming distance btw rows and columns are maximized.

- Voting scheme $$y_i = \sum_{j=1}^{L} w_j d_{ji}$$

- No guarantee that subtasks for the dichotomies will be simple

- Code matrix and dichotomizers not optimized together

Overview

- Introduction
  - Rationale
- Combination Methods
  - Static Structures
    - Ensemble averaging (Voting…)
    - Bagging
    - Boosting
    - Error Correcting Output Codes (end of the chapter)
  - Dynamic structures
    - Mixture of Experts
    - Stacking
    - Cascading

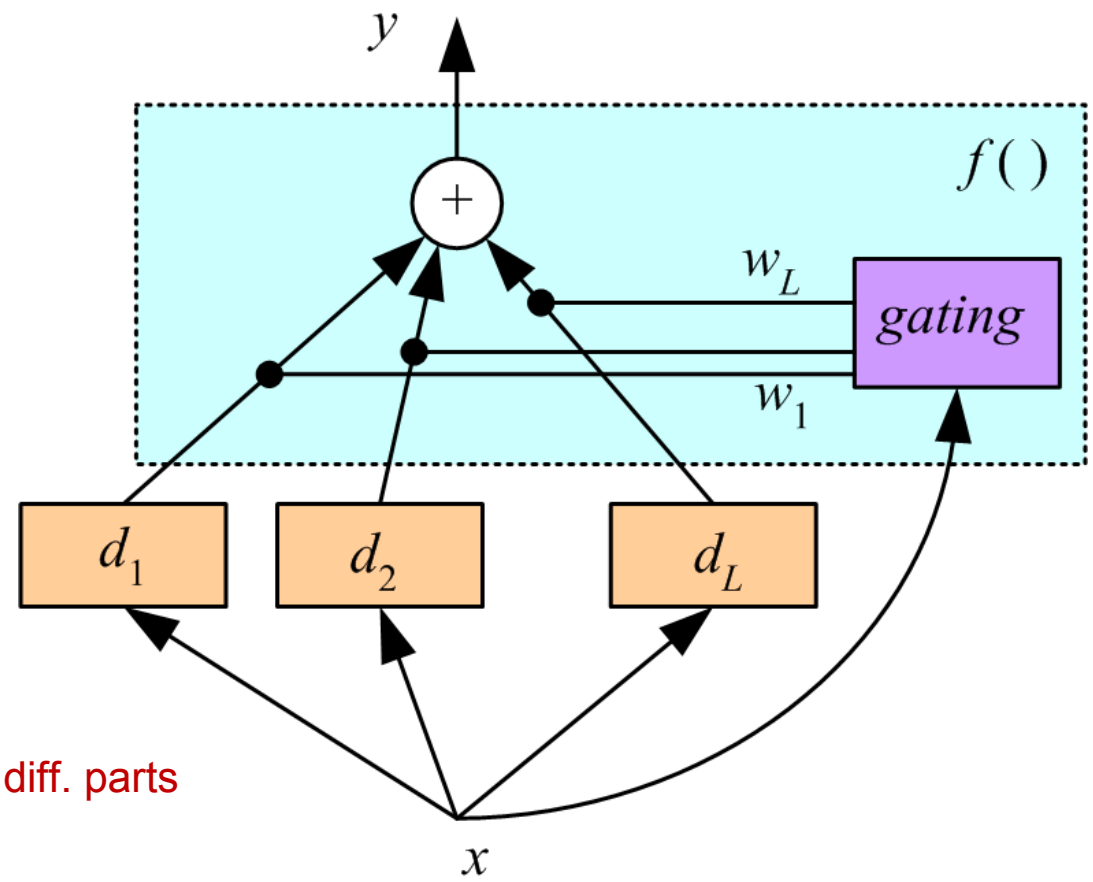
Dynamic Methods > Mixtures of Experts

Voting where weights are input-dependent (gating) – not constant

$$y = \sum_{j=1}^{L} w_j d_j$$

(Jacobs et al., 1991)

In general, experts or gating can be non-linear

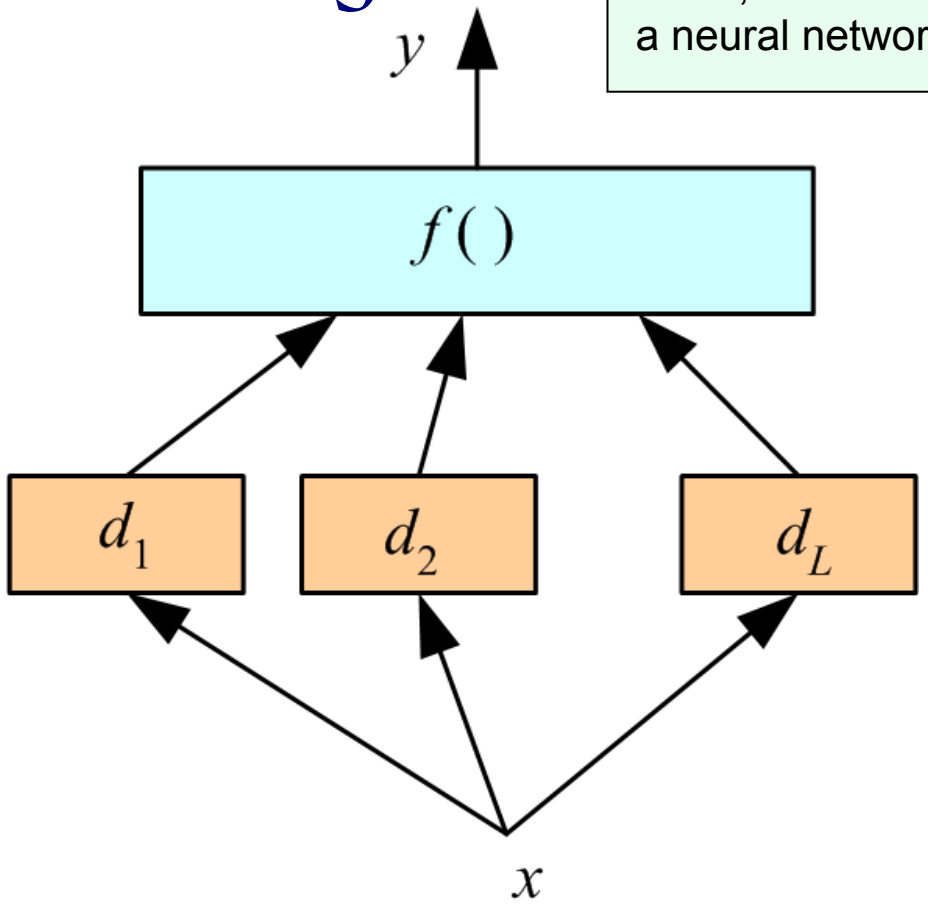


Base learners become experts in diff. parts of the input space

Dynamic Methods > Stacking

f need not be linear, it can be a neural network

- Wolpert 1992



We cannot train f() on the training data; combiner should learn how the base-learners make errors.

Leave-one-out or k-fold cross validation

Learners should be as different as possible, to complement each other ideally using different learning algorithms

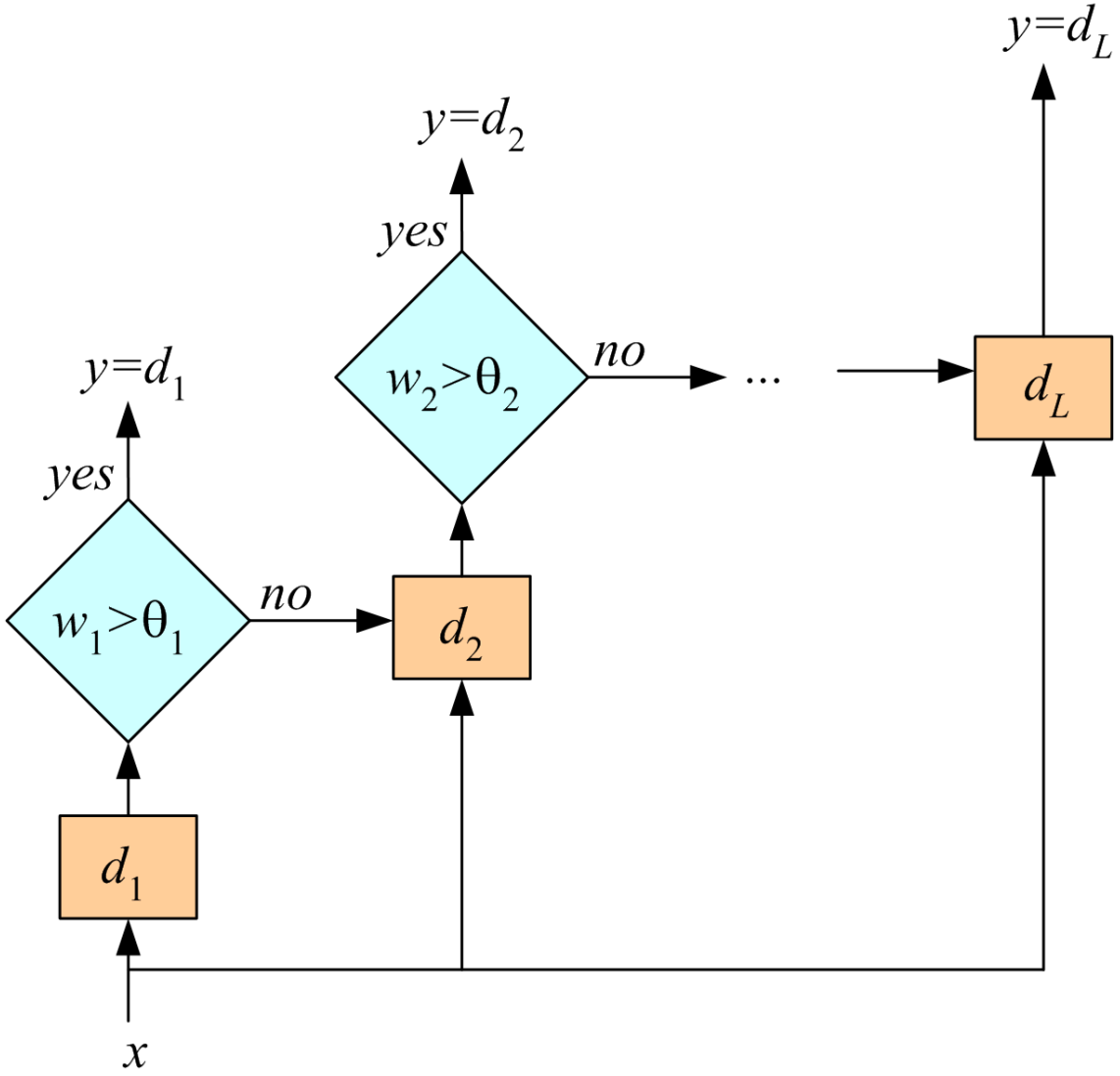
Dynamic Methods > Cascading

Cascade learners in order of complexity

Use d_j only if preceding ones are not confident

Training must be done on samples for which the previous learner is not confident

Note the difference compared to boosting



Dynamic Methods > Cascading

- Cascading assumes that the classes can be explained by small numbers of “rules” in increasing complexity, and a small set of exceptions not covered by the rules

General Rules of Thumb

- Components should exhibit low correlation - understood well for regression, not so well for classification. "Overproduce-and-choose" is a good strategy.
- Unstable estimators (e.g. NNs, decision trees) benefit most from ensemble methods. Stable estimators like k-NN tend not to benefit.
- Boosting tends to suffer on noisy data.
- Techniques manipulate either training data, architecture of learner, initial configuration, or learning algorithm. Training data is seen as most successful route; initial configuration is least successful.
- Uniform weighting is almost never optimal. Good strategy is to set the weighting for a component proportional to its error on a validation set.

References

- M. Perrone – review on ensemble averaging (1993)
- Thomas G. Dietterich. *Ensemble Methods in Machine Learning* (2000). Proceedings of First International Workshop on Multiple Classifier Systems
- David Opitz and Richard Maclin. *Popular Ensemble Methods: An Empirical Study* (1999). Journal of Artificial Intelligence Research, volume 11, pages 169-198
- R. A. Jacobs and M. I. Jordan and S. J. Nowlan and G. E. Hinton. *Adaptive Mixtures of Local Experts* (1991). Neural Computation, volume 3, number 1, pages 79-87
- Stuart Haykin - *Neural Networks: A Comprehensive Foundation* (Chapter 7)
- **Ensemble bibliography:** http://www.cs.bham.ac.uk/~gxb/ensemblebib.php
- **Boosting resources:** http://www.boosting.org