CSE-E4810 Machine Learning and Neural Networks (5 cr)

Lecture 12: Combining Multiple Classifiers

Prof. Juha Karhunen

https://mycourses.aalto.fi/course/view.php?id=13086

Introduction

- Different learning algorithms have different accuracies.
- In Du's and Swamy's book so called No free lunch theorem is presented in Section 2.9.
- Basically it states that there is no single learning algorithm which would always achieve the best performance in all problems.
- Different learning algorithms can be combined to attain higher accuracy.
- For classification, different classifiers using the same classification method can be generated:
  - Using different initializations of a classifier, leading to ensemble averaging.
  - Training a classifier with different data sets created independently

from the original training data set, leading to bagging.

  - Training a classifier using data sets created sequentially from the original training data set, leading to boosting.

- It is also possible to train classifiers using different feature sets (components of the data vectors).
- And of course one could use different classification methods, but this may be costly.
- These techniques combine different classifiers to form a committee machine.
- Different classifiers have been property that they can make errors for different data vectors.
- Ensemble learning has capacity of improving the classification accuracy of any single classifier given the same amount of training information.

- Majority voting is the most popular method of combining classifiers.
- In this method, the data vector is assigned to the class which gets most votes from the individual classifiers.
- In regression problems, a simple but effective method is to determine the final output by computing the average of the outputs of individual learners.
- This averaging can be used also if the outputs of the classifiers have continuous values.
- The material for this lecture has been taken mainly from Chapter 7: "Committee Machines" of the book S. Haykin, "Neural Networks: A Comprehensive Foundation", Prentice-Hall, 1998.
- Complemented with some material from Chapter 20: "Combining Multiple Learners: Data Fusion and Ensemble Learning" from Du's and Swamy's book.

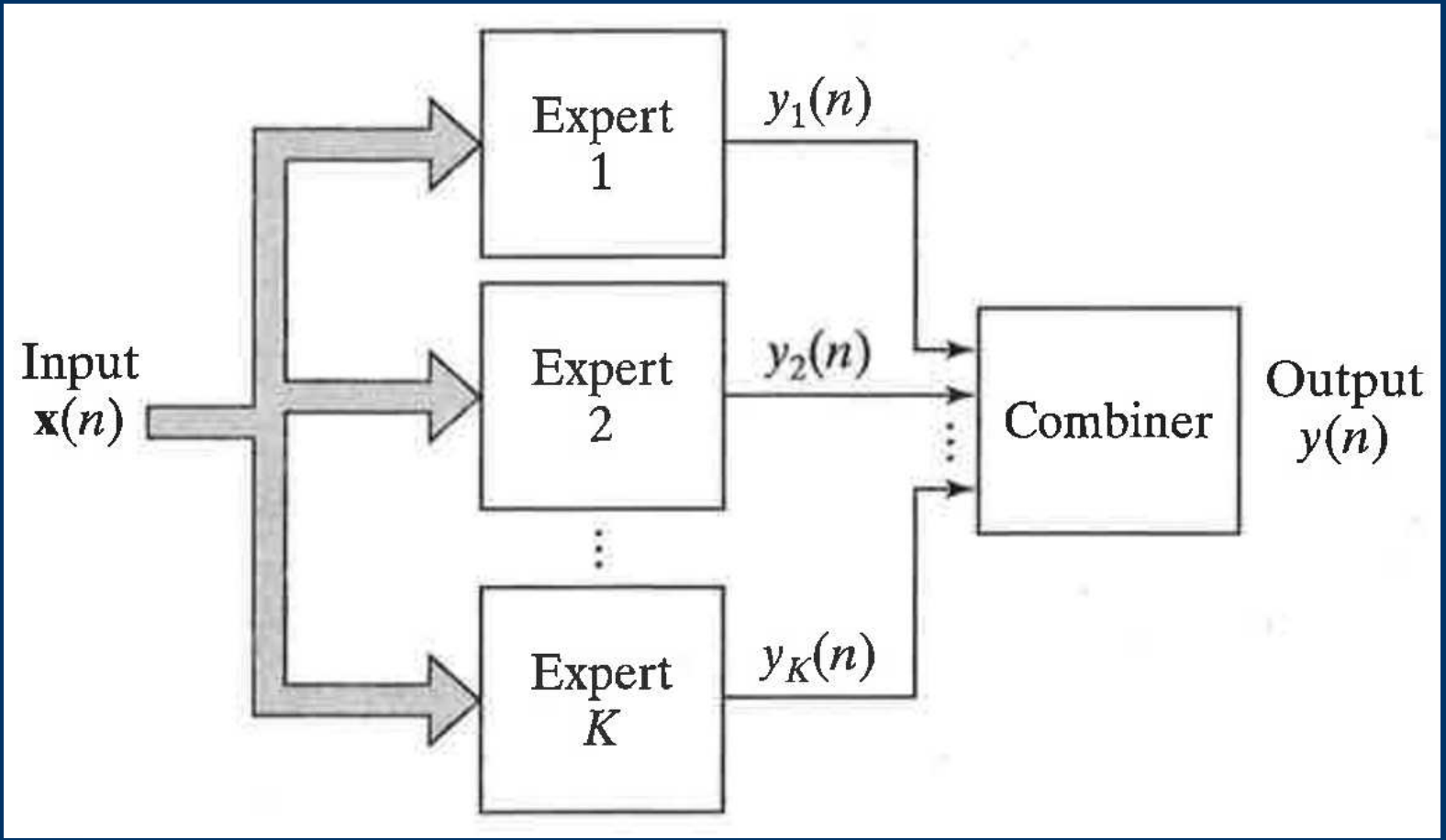

Figure 1: A committee machine based on ensemble averaging.

Ensemble averaging

- Figure 1 shows a number of differently trained neural networks (experts).
- They share a common input, and their individual outputs are somehow combined to produce an overall output y.
- For simplicity, the outputs of the experts are assumed to be scalar-valued.
- Such a technique is referred to as an ensemble averaging method.
- There are two motivations for using ensemble averaging:
  - If we used instead of a combination of experts a single neural network, we would need a large network with many parameters to be learned.
  - The training time of such a large network is likely to be longer

than for a set of experts trained in parallel.

  - The risk of overfitting increases when the number of parameters to be learned is large compared to the size of training data set.

- The expectation is that the differently trained experts converge to different local minima on the error surface.
- And the overall performance is improved by combining their outputs in a suitable way.
- There are different ways of individually training the expert networks in Figure 1, and also different ways to combine their outputs.
- Consider now the situation where the expert networks have an identical configuration but they are trained from different initial conditions.
- Assume further that the output of the committee machine of Figure 1 is computed simply by averaging the outputs of individual experts.

- This situation is considered theoretically in Section 7.2 of Haykin's book "Neural Networks: A Comprehensive Foundation, 2nd ed.", Prentice-Hall, 1998.
- The conclusions from this analysis are:
  - The bias of the ensemble averaged output of the committee machine is the same as for individual experts (neural networks).
  - The variance of the output of the committee machine is smaller than for individual experts.
- One can therefore apply a training strategy in which the individual experts are purposedly overtrained using different initial conditions.
- Overtraining reduces the bias but increases the variance (recall the discussion in Lecture 6).
- But ensemble averaging of the experts reduces the variance, leaving the bias unchanged.

Bagging

- Bagging is discussed briefly in Section 20.3 in Du's and Swamy's book.
- Bagging, short for Bootstrap aggregating, works by training each classifier or learner on a bootstrap sample.
- Each bootstrap sample, called bag, is created by uniformly sampling instances from the entire training set with replacement.
- The bags have the same size as the original data set.
- But some samples may appear more than once, while others may not appear at all.
- Bagging then construct a new classifier or learner for each bag.
- In regression problems the predictions of these learners are averaged.
- While in classification problems the data vector is assigned to the

class which gets most votes from the learned classifiers.

- The essential idea in bagging is to average many noisy but approximately unbiased models.
- Hence reducing the prediction variance without affection the prediction bias.
- Bagging seems to work especially well for methods having a low bias but high variance.
- Such as decision trees to be discussed later on in this lecture.
- However, in general the performance of bagging is worse than for boosting, to be discussed next.

Boosting

- Boosting is discussed in Section 20.2 in Du's and Swamy's book, and in Section 7.4 in Haykin's book.

- We follow Haykin's presentation because it is clearer and more detailed, and presents a computer experiment.
- Theoretically, boosting is based on the probably approximately correct (PAC) model of learning.
- PAC is discussed briefly in Section 2.8.3 in Du's and Swamy's book, and in more detail in Section 2.15 in Haykin's book (1998).
- PAC is a highly theoretical topic, and we skip its detailed discussion.
- However, we utilize the result of PAC that one can build a strong learning model around several weak learning models.
- In two-class (binary) classification problems a weak learning model (method) means that the error rate of a classifier is just a little less than 50% only.
- Note that for a random guess with no learning the error rate would be around 50% in binary classification problems.

- Boosting modifies the distribution of training samples so that a strong learning method is built around a weak one.
- Boosting can be implemented in three different ways:
  1. Boosting by filtering the training data using different versions of a weak learning algorithm.
  2. Boosting by subsampling which uses a training set having fixed size.
  3. Boosting by reweighting which is not discussed in this lecture and in Haykin's book (1998).
- Boosting by filtering has small memory requirements compared with the other two approaches.
- But it often requires a large training data set.

Boosting by filtering

- In boosting by filtering, the committee machine consists of three experts.
- The algorithm used to train them is called a boosting algorithm.
- The three experts are labeled "first", "second", and "third".
- These three experts are individually trained as follows:
- 1. The first expert is trained on a data set consisting of N_1 training samples.
- 2. The second expert is trained by using the trained first expert to filter another data set for it in the following manner:
- Flip a fair coin simulating a random guess. In practice you can generate a uniformly distributed random number in the interval $[0, 1]$.

- If the result is heads (your random number is between $[0, 0.5)$), pass new data vectors through the first expert.
- And discard correctly classified data vectors until a data vector is misclassified.
- That misclassified data vector is added to the training set for the second expert.
- If the result is tails (your random number is between $[0.5, 1)$), do the opposite.
- Specifically, pass new data vectors through the first expert and discard incorrectly classified data vectors until a data vector is classified correctly.
- That correctly classified data vector is added to the data set for the second expert.

- Continue this process until a total of N_1 examples has been filtered by the first expert, and train the second expert with this data set.
- This coin flipping procedure ensures that if the first expert is tested on the data set of the second expert, it would have an error rate of 50%.
- And the second set of N_1 data vectors used for training the second expert has a distribution that is entirely different from the training set of the first expert.
- 3. The third expert is trained using a third training set which is formed as follows:
- Pass a new data vector through both the first and second experts.
- If these two experts agree in their decisions, discard that data vector.
- But if they disagree, that data vector is added to the training set for the third expert.

- Continue this process until also the training set for the third expert contains N_1 data vectors.
- The third expert is then trained in the usual way, and the training of the committee machine is thereby completed.
- This three-point filtering operation is illustrated in Figure 2.
- The total size of the training set needed for creating the training sets for Experts 2 and 3 may grow large, depending on the problem.
- But each expert needs the same amount, N_1 data vectors, for its actual training.
- The filtering operations describe above make the second and third expert to focus on hard-to-learn parts of the training data.
- In the basic boosting by filtering algorithm simple voting was used for classifying test data vectors not used in learning.

- Specifically, if the first and second experts in the committee machine agree in their decisions, that class label is selected.
- But if they disagree the label of third expert is selected.
- However, adding the outputs of the three experts yields a better performance than voting.
- It can be shown theoretically that the committee machine based on boosting by filtering decreases the error rate compared with no boosting.
- Boosting performs worse than bagging in the presence of strong noise and outliers.
- But it does not normally suffer from overfitting.
- Boosting is the preferred choice compared with bagging in most problems.

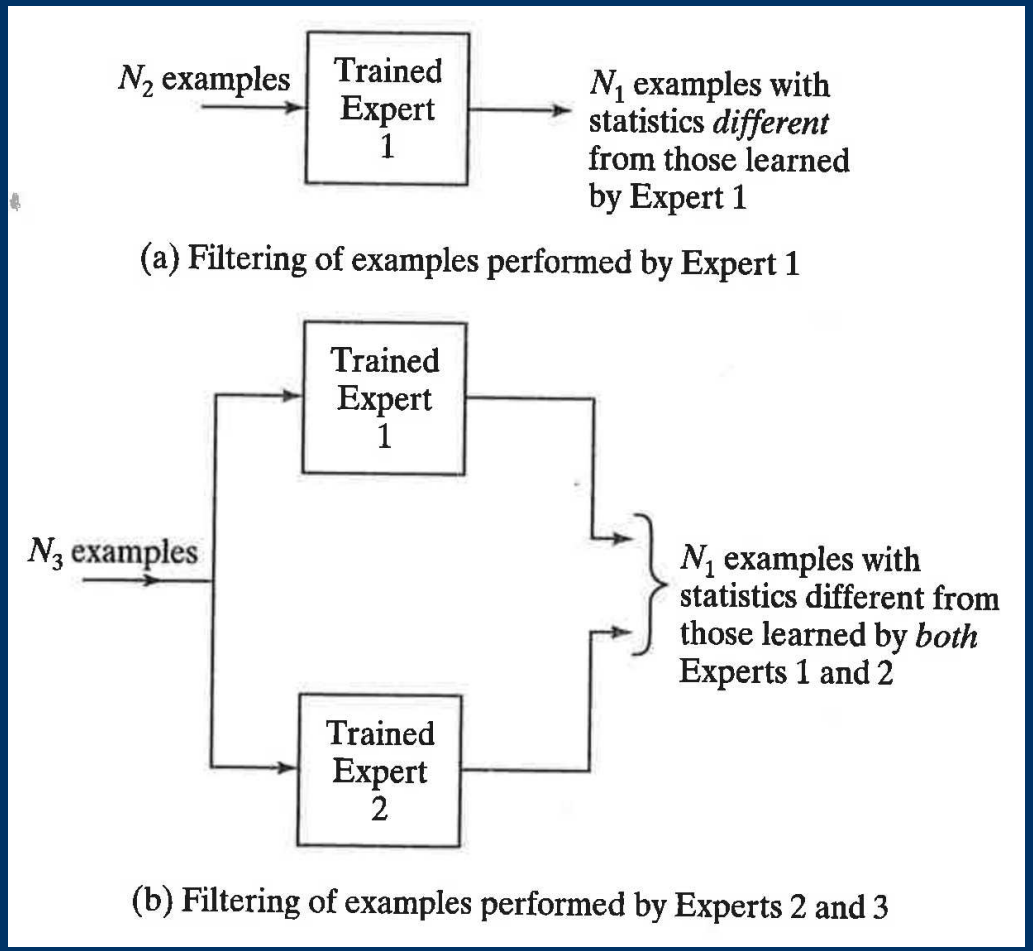

Figure 2: Illustration of boosting by filtering.

AdaBoost

- A practical limitation of boosting by filtering is that it often requires a large training data set.
- This limitation can be overcome by another boosting algorithm called AdaBoost, which belongs to boosting by resampling.
- AdaBoost has access to a weak classifier for which the error rate in binary (two-class) classification problems is only a little below 50%.
- The goal is to create from weak classifiers a strong classifier which has a low error rate.
- AdaBoost adjusts adaptively to the errors of a weak classifier, hence its name.
- AdaBoost operates as follows.
- On iteration n, the boosting algorithm provides to the weak

classifier a distribution $\mathcal{D}_n$ over the training data set $\mathcal{T}$.

- The training set $\mathcal{T}$ consist of data vectors $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_K]$ and their correct class labels.
- In response, the weak classifier computes a mapping $\mathcal{F}_n : \mathbf{X} \rightarrow Y$ that correctly classifies a fraction of the training set $\mathcal{T}$.
- The error is measured with respect to the distribution $\mathcal{D}_n$.
- The process continues for T iterations, and finally AdaBoost combines the mappings (classifiers) $\mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_T$ into a single final mapping (classifier) $\mathcal{F}_{fin}$.
- The distributions $\mathcal{D}_n$ on iteration n are calculated as follows.
- The initial distribution $\mathcal{D}_1$ over the training set $\mathcal{T}$ is uniform:

$$\mathcal{D}_1(i) = \frac{1}{n} \quad \text{for all } i \tag{1}$$

- On iteration n, we have the distribution $\mathcal{D}_n$, mapping (classifier) $\mathcal{F}_n$, and its classification error ϵ_n.
- The next distribution $\mathcal{D}_{n+1}$ is computed by multiplying the weight of i:th training data vector $\mathbf{x}_i$ by the number

$$\beta_n = \frac{\epsilon_n}{1 - \epsilon_n} \tag{2}$$

  if $\mathcal{F}_n$ classifies $\mathbf{x}_i$ correctly.
- For wrong classifications the weight is left unchanged.
- After this the weights are renormalized so that $\mathcal{D}_{n+1}$ is a probability distribution.
- AdaBoost is summarized in Table 7.2 in Haykin's book (1998).
- In effect, easy samples in the training set $\mathcal{T}$ that are correctly classified by many of the previous weak classifiers are given lower weights.

- Whereas the hard samples that are often misclassified are given higher weights.
- Thus the AdaBoost method focuses the most weight on samples that are the hardest for it to classify.
- The final classification $\mathcal{F}_{fin}$ is computed as a weighted vote of the weak classifiers $\mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_T$.
- That is, for a given input vector $\mathbf{x}$, the final classifier outputs the label that maximizes the sum of the weights of the weak classifiers predicting that label.
- The weight of classifier $\mathcal{F}_n$ is defined to be $\log(1/\beta_n)$, so that a greater weight is given to classifiers with lower error.
- Experiments with AdaBoost show that the test error continues to decrease with more boosting iterations after the training error has reduced essentially to zero.

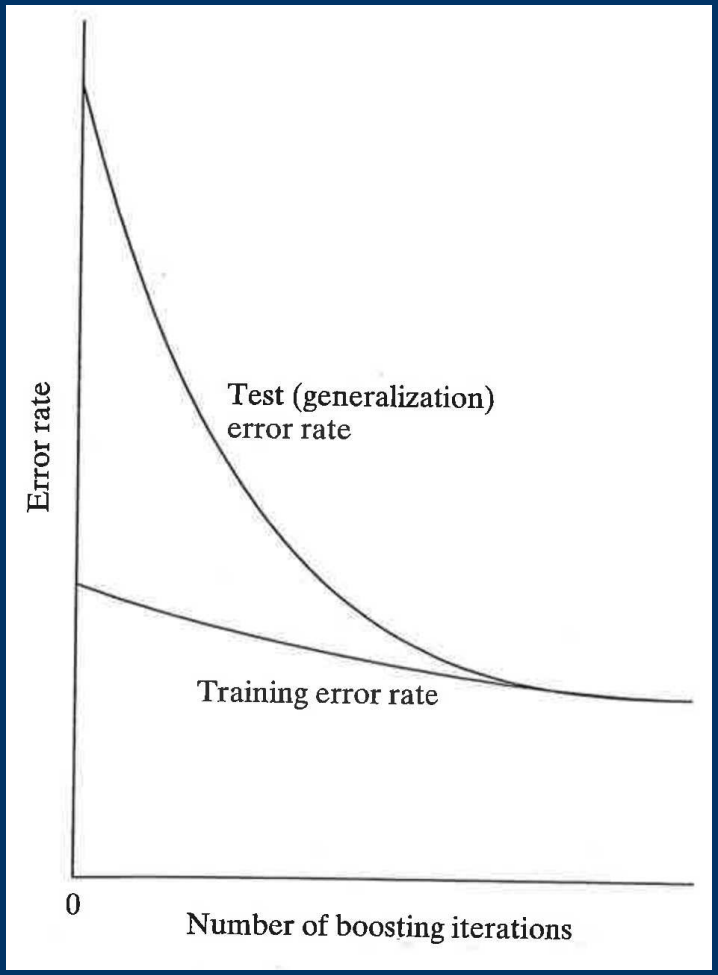

Figure 3: Error performance of the AdaBoost algorithm.

- This phenomenon is illustrated in Figure 3, and a similar result has been reported for boosting by filtering.
- Thus these boosting techniques do not suffer from overfitting.
- These results are quite surprising in light of what we know about the generalization performance of a single neural network.
- A theoretical explanation for this phenomenon is given in Haykin's book in Section 7.4.
- Two key problems for AdaBoost algorithms are:
  - How to select the most discriminative weak learners.
  - How to combine them.
- Many modifications and improvements of the AdaBoost method have been proposed.
- They are described quite briefly and only literally in subsection

20.2.1 of Du's and Swamy's book.

- But you can find more information from the references given there.

A computer experiment with boosting by filtering

- In this experiment, boosting by filtering is applied to a fairly difficult classification problem.
- There are two two-dimensional classes, and their data vectors lie respectively in the areas labeled $\mathcal{C}_1$ and $\mathcal{C}_2$ in Figure 4.
- The committee machine used to solve this problem consists of three experts.
- Each expert is a 2-5-2 multilayer perceptron (MLP) network having two input nodes, five hidden neurons, and two output neurons.
- The backpropagation algorithm was used in training the three expert networks.

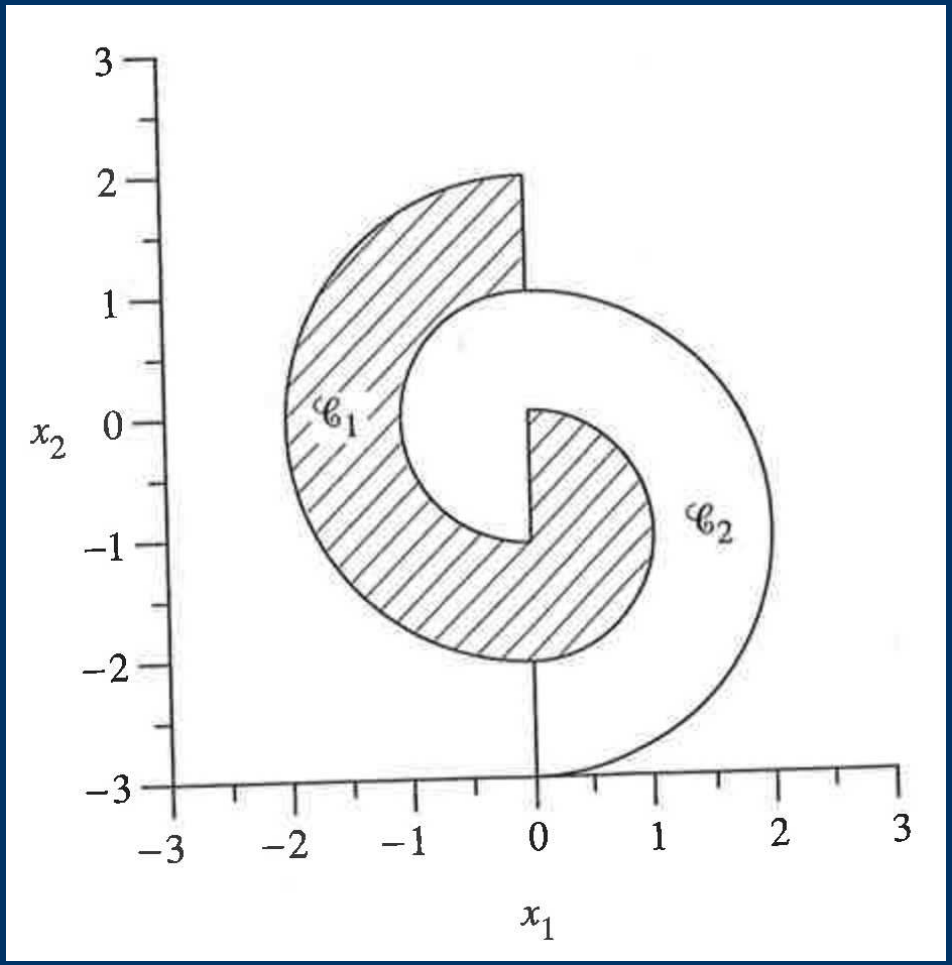

Figure 4: Classes 1 and 2 in experiment on boosting.

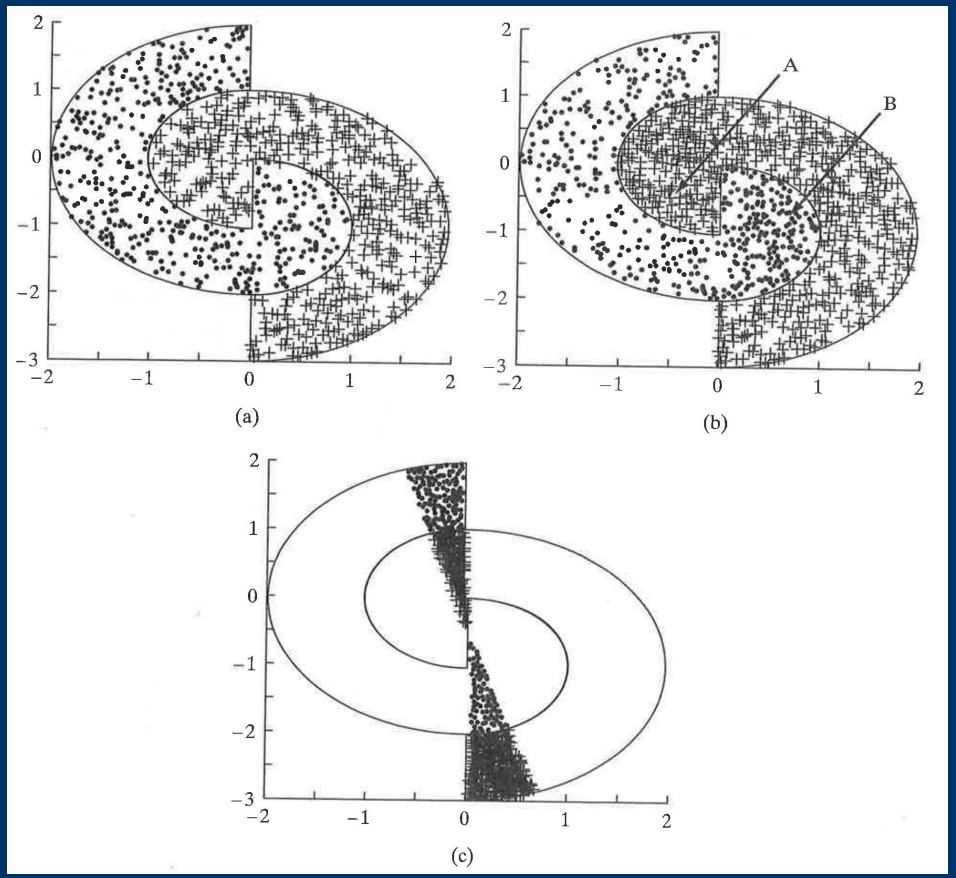

Figure 5: Data vectors used by Expert 1 (a), Expert 2 (b), and Expert 3 (c) in experiment on boosting.

- Figure 5 shows the data vectors used to train the three experts.
- The data shown in Fig. 5a were used to train expert 1.
- The data shown in Fig. 5b were filtered by expert 1 after its training was completed, and it was used to train expert 2.
- The data shown in Fig. 5c was filtered using both experts 1 and 2, and it was used for training expert 3.
- The size of training sets of each expert was $N_1 = 1000$ data vectors.
- Examining these three subfigures we observe:
  - The training data for expert 1 in Fig. 5a are uniformly distributed.
  - The training data for expert 2 in Fig. 5b contains more data points in areas labeled by A and B that are difficult to classify for the expert 1.

  - The training data for expert 3 in in Fig. 5c shows an even greater concentration of data points that are difficult to classify for experts 1 and 2.

- Figures 6 (a)-(c) display the decisions boundaries formed by experts 1, 2, and 3, respectively.
- Figure 6 (d) shows the overall decision boundary of the committee machine formed by these three expert.
- This boundary is obtained by simply summing their outputs.
- The probabilities of correct classification for the three experts on test data were:
  - Expert 1: 75.15%
  - Expert 2: 71.44%
  - Expert 3: 68.90%
  - Entire committee machine: 91.79%

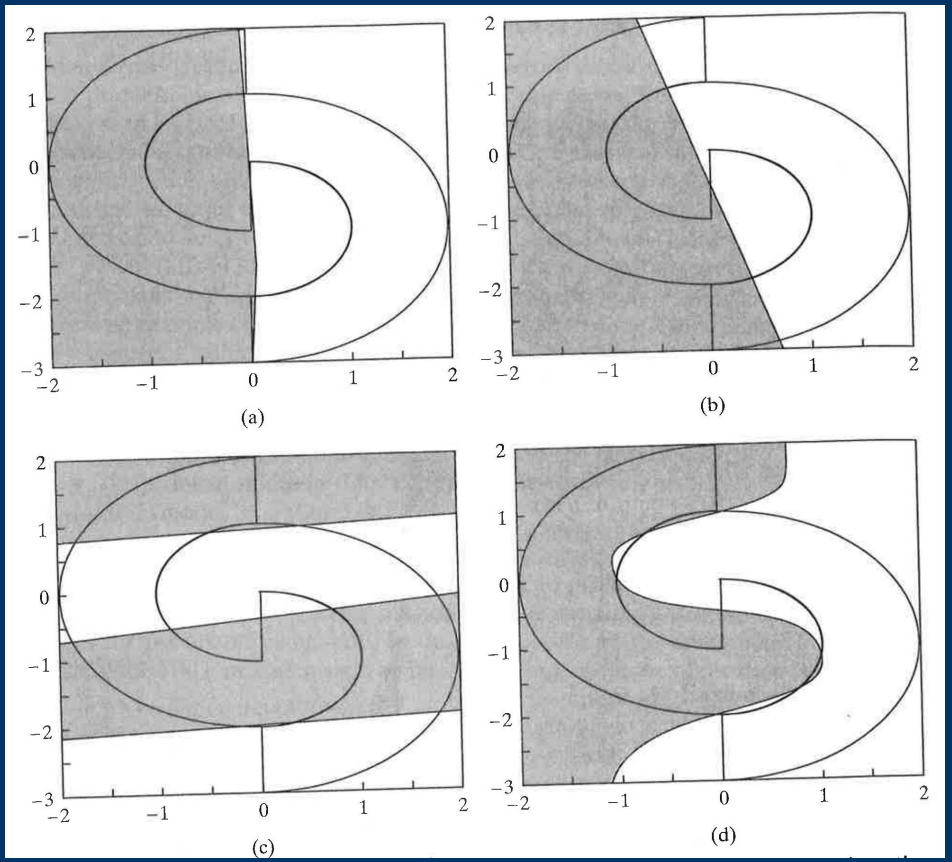

Figure 6: Decisions boundaries of Expert 1 (a), Expert 2 (b), Expert 3 (c), and Entire committee machine (d).

- The test data set contained 32,000 data vectors.
- The overall decision boundary in 6 (d) constructed using boosting by filtering is further evidence of the good classification performance of the committee machine.

Classification and regression trees (CART)

- Classification and regressions trees or CART models are called also decision trees.
- They are defined recursively by partitioning the input space, and defining a local model in each resulting region.
- This can be represented by a decision tree, with one leaf per region.
- Decision trees are discussed in Chapter 9 in the book E. Alpaydin, "Introduction to Machine Learning, 2nd ed.", The MIT Press, 2010.
- And in Section 16.2 in the book K. Murphy, "Machine Learning: A

Probabilistic Perspective", The MIT Press, 2012.

- It is an excellent book with more than 1000 pages(!) but it does not discuss neural networks.
- We are not going into details of decision trees such as their growing and pruning but just represent the basic idea.
- In its simplest form, a decision tree splits coordinate axes into regions.
- For example, the first node of the tree asks whether the variable x_1 is less than some threshold t_1.
- If yes, we then ask whether x_2 is less than some threshold t_2.
- If yes, we have arrived to a leaf of the tree.
- If no, we ask whether x_1 is less than t_3, and so on.
- We can then associate a constant mean response with each of the

regions defined by the decision tree.

- We can generalize this to the classification setting by storing the distribution over class labels in each leaf, instead of the mean response.
- Let us consider a simple toy example taken from Figure 1.1 of Murphy's book, shown in Figure 7.
- The figure is somewhat blurred but it shows colored shapes belonging to two classes.
- The shapes in the left square (yes) belong to the class 1 with the label (output value) $y = 1$.
- And the shapes in the right square (no) belong to the second class having the class label (output) $y = 0$.
- Below these two squares there are three test cases.

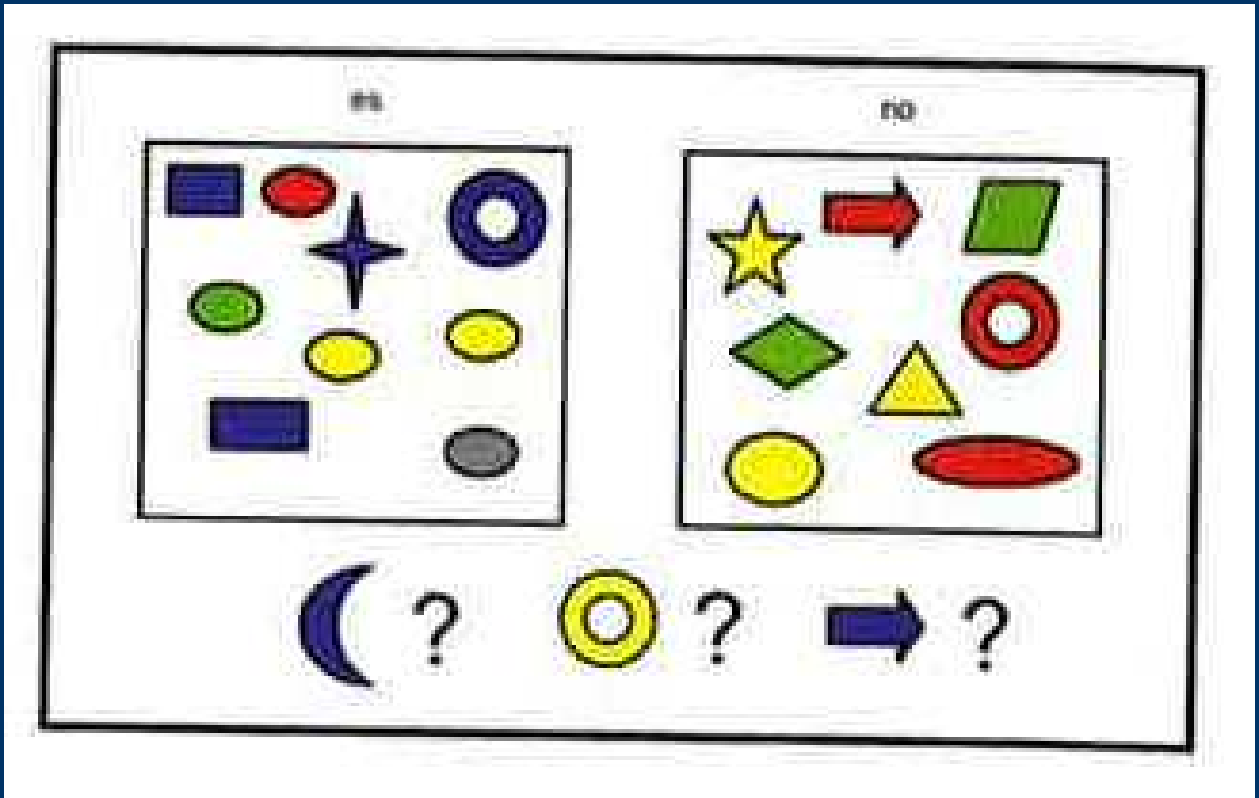

Figure 7: Some labeled training examples of colored shapes, along with 3 unlabeled test cases

- A reasonable guess for first test case, a blue crescent, is class $y = 1$ because all the blue shapes are in the left square with the label $y = 1$.
- The other two test cases are difficult to classify as they have features from both classes.
- We do not bother now about these test cases but present a simple decision tree for the labeled data in Figure 7.
- It is shown in Figure 8.
- A leaf labeled as (n_1, n_0) means that there are n_1 positive examples that match this path, and n_0 negative examples.
- In this tree, most of the leaves are pure, meaning that they only have examples of one class or the other.

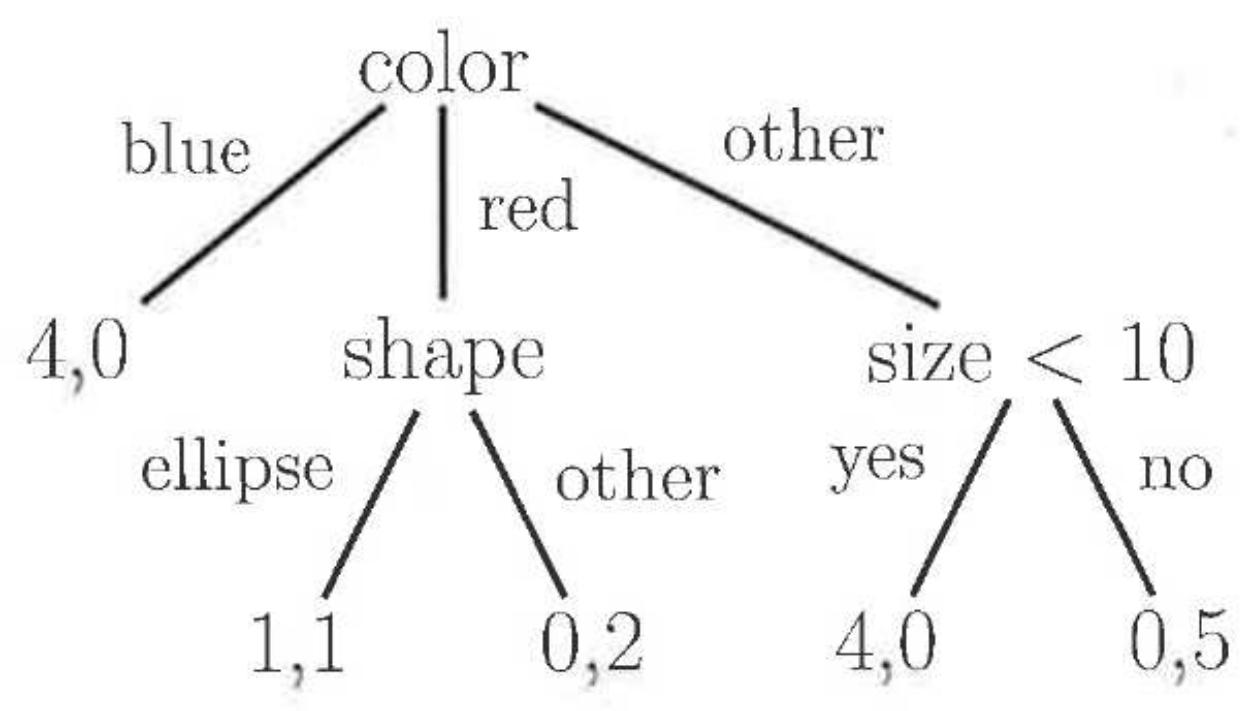

Figure 8: A simple decision tree for the data in Figure 7. A leaf labeled as (n_1, n_0) means that there are n_1 positive examples that match this path, and n_0 negative examples.

- The only exception is leaf representing red ellipses, which has a label distribution of $(1, 1)$.
- We could distinguish the classes of these ellipses by adding a further test (node) measuring their size.
- However, it is not always desirable to construct trees that perfectly model the training data, due to overfitting.
- CART models (decision trees) are popular for several reasons:
  - + They are easy to interpret.
  - + They can easily handle mixed discrete and continuous inputs.
  - + They are insensitive to monotone transformations of the inputs because they are based on ranking the data points.
  - + They perform automatic variable selection.
  - + They are relatively robust against outliers.
  - + They scale well to large data sets.

+ They can be modified to handle missing inputs.

- However, CART models also have some disadvantages.
  - The most important one is that they are not among the best performing classification and regression methods.
  - A related problem is that the trees are unstable: small changes to the input data can have large effects on the structure of the tree.
  - In other words, trees are high variance estimators.

Random Forests

- One way to reduce the variance of an estimate is to average together many estimates.
- The first idea is to use bagging discussed earlier by training M different trees with equally sized data sets.
- They are created by random sampling of the original data set with

replacement.

- One can then compute the average of these trees

$$f(\mathbf{x}) = \frac{1}{M} \sum_{m=1}^{M} f_m(\mathbf{x}) \tag{3}$$

where f_m is the m:th tree.

- Unfortunately bagging can result in highly correlated predictors, limiting the amount of variance reduction that is possible.
- A random forest is a collection of classification trees.
- They were introduced in Breiman's paper given in the references.
- The data sets are created similarly as in bagging by random sampling of the original data set with replacement.
- But the variance reduction is improved by random selection of input variables.

- For each data set, a classification tree is formed.
- After the trees have been constructed, a new data vector is dropped down through each tree.
- The data vector is assigned to the class that gets most votes from all the trees belonging to the random forest.
- When used for regression, the predictions from individual trees are simply averaged.
- As discussed soon, some versions of random forests have excellent performance in classification problems.
- And they are simpler to train and tune than boosting methods.
- Therefore, the random forest method is popular, and is implemented in a variety of software packages.

Other methods

- Mixture-of-experts is a divide-and-conquer algorithm that contains a gating network for soft partitioning of the input space.
- And expert networks for modeling each of these partitions.
- This method is discussed extensively in Chapter 7 of Haykin's book (1998, 2nd edition), but no longer in its latest 3rd edition in 2009.
- As well as in Murphy's book.
- Bayesian committee machine does not belong to our course.
- Stacking replaces a simple average by a weighted average, where the weights take into account the complexity of the model.
- Aggregation operators combine data from several sources to improve the quality of information.

A comparison of classifiers

- In our course, we have introduced several types of classifiers and presented some illustrative results on them.
- But mainly for a two-dimensional toy problem of two overlapping classes with Gaussian probability densities.
- In real-world problems, the mutual performance of classifiers may vary between different data sets.
- Recently, an important paper on a very extensive comparison of classifiers has been published:
- M. Fernadez-Delgado, E. Cernadas, S. Barro, and D. Amorim, "Do we need hundreds of classifiers to solve real world classification problems", Journal of Machine Learning Research", Vol. 15, pp. 3133-3181, 2014.

- In this paper, the performance of 179 classifiers arising from 17 families were evaluated with 121 data sets.
- The goal was to achieve significant conclusions about the performance of classifiers which do not depend on a few data sets.
- Most of these data set are from the UCI machine learning data repository: https://archive.ics.uci.edu/ml/datasets.html
- The 17 families of classifiers include discriminant analysis, Bayesian classifiers, neural networks, support vector machines, and decision trees.
- And rule-based classifiers, boosting, bagging, stacking, random forests and other ensembles, generalized linear models, and nearest-neighbors.
- As well as partial least squares and principal component regression, logistic and multinomial regression, multiple adaptive regression

splines, and other methods.

- Each of these families include usually many somewhat different versions, for example 20 classifiers belonging to discriminant analysis and 8 to random forests.
- The best classifier was a version of random forest which achieved 94.1% classification accuracy.
- However, the difference is statistically not significant with the second best classifier having 92.3% accuracy.
- It was a support vector machine (SVM) with Gaussian kernel.
- A few models / methods are clearly better than the remaining ones:
  - Random forests;
  - SVM with Gaussian and polynomial kernels;
  - Extreme learning machine using Gaussian kernels of varying

widths;
  - A version of decision tree;
  - A committee of 5 small multilayer perceptron networks.
- You can find references in which these methods have been described in detail in the paper given above.
- The clearly best family of classifiers is random forests, having 3 classifiers among the 5 best ones.
- It is followed by support vector machines having 4 classifiers in the top-10.
- Neural networks had 5 members and boosting ensembles 3 among the 20 best classifiers.
- But radial-basis function networks and learning vector quantization were not among the best classifiers.

- However, deep learning methods to be discussed on the last lecture were not included in this comparison.
- Obviously because there is not yet any freely available easy-to-use software for them.
- One should also take into account that the classification problems tested in this comparison are usually fairly small or moderately dimensional.
- Especially in biological and internet applications data vectors may have a very high dimensionality.
- The problems can be quite sparse so that the amount of training pairs can be much less than the dimensionality of data vectors.
- Many well-known methods may not be practical simply because of their high computational load, requiring learning of too many parameters.

References

1. S. Haykin, "Neural Networks: A Comprehensive Foundation", Prentice-Hall, 1998. Chapter 7, "Committee Machines", pp. 351-391.
2. K.-L. Du and M. Swamy, "Neural Networks and Statistical Learning", Springer, 2014. Chapter 20, "Combining Multiple Learners: Data Fusion and Ensemble Learning", pp. 621-643.
3. E. Alpaydin, "Introduction to Machine Learning, 2nd ed.", The MIT Press, 2010. Chapter 9, "Decision Trees", pp. 185-208.
4. K. Murphy, "Machine Learning: A Probabilistic Perspective", The MIT Press, 2012. Section 16.2, "Classification and Regression Trees (CART)", pp. 544-552.

5. M. Fernadez-Delgado, E. Cernadas, S. Barro, and D. Amorim, "Do we Need Hundreds of Classifiers to Solve Real World Classification Problems", Journal of Machine Learning Research", Vol. 15, pp. 3133-3181, 2014.

6. L. Breiman, "Random Forests", Machine Learning, Vol. 45, pp. 5-32, 2001.