COMP3308/3608, Lecture 10b

ARTIFICIAL INTELLIGENCE

Ensembles of Classifiers

Reference: Russell and Norvig, pp. 748-753

Witten, Frank, Hall and Pal, ch.12

Outline

- **Ensembles of learning algorithms**
- **Motivation**
- **Creating ensembles by:**
  - **Manipulating the training data**
    - **Bagging**
    - **Boosting**
  - **Manipulating the attributes**
    - **Random Forest**

Ensemble of Classifiers

- **So far we have studied *single* classifiers**
- **An *ensemble* of classifiers combines the predictions of *multiple* classifiers (called *base* classifiers) in some way to classify new instances**
- **An ensemble of classifiers = a *committee* of classifiers**

- **Example:**
  - **An ensemble of 50 backpropagation NNs, all trained on the same dataset but having different parameters (e.g. different architecture, initialization, learning rate, etc.)**
  - **To classify new example, the individual NN predictions are combined by taking the majority vote**

Ensembles in Real Life

- **When people make critical decisions, they typically take into account the opinions of several experts rather than one only**
- **Example: A patient wants to have a diagnosis based on symptoms. Instead of asking one doctor, he asks several.**

- **Different ways to combine opinions:**
  1) **If a certain diagnosis occurs more than the others, he chooses it as the final diagnosis, i.e. takes the majority vote *(the intuition behind bagging)***
  2) **Instead of trusting equally each of the doctors, he assigns weights to the "value" of each doctor's diagnosis, based on the accuracies of their previous diagnoses. The final diagnosis is a combination of the weighted diagnoses. *(the intuition behind boosting)***

Motivation

Intuitively, when does combining multiple classifiers help?

1) When the base classifiers do not make the same mistakes

- **i.e. they complement each other, each is an expert in a part of the domain where the others don't perform well**

2) When each base classifier is reasonably accurate

Example - Case 1

- **An ensemble of 25 base classifiers**
- **2 classes – binary classification task**
- **Each base classifier has an error rate ε=0.35 on the test set**
- **To predict the class of a new example, the individual predictions are combined by a majority vote**

- **Case 1: The base classifiers are identical (i.e. make the same mistakes)**

What will be the error rate of the ensemble on the test set?

Example – Case 2

- **Case 2: The base classifiers are independent (i.e. their errors are not correlated)**

When will a new example be misclassified? Only if more than half of the base classifiers predict incorrectly.

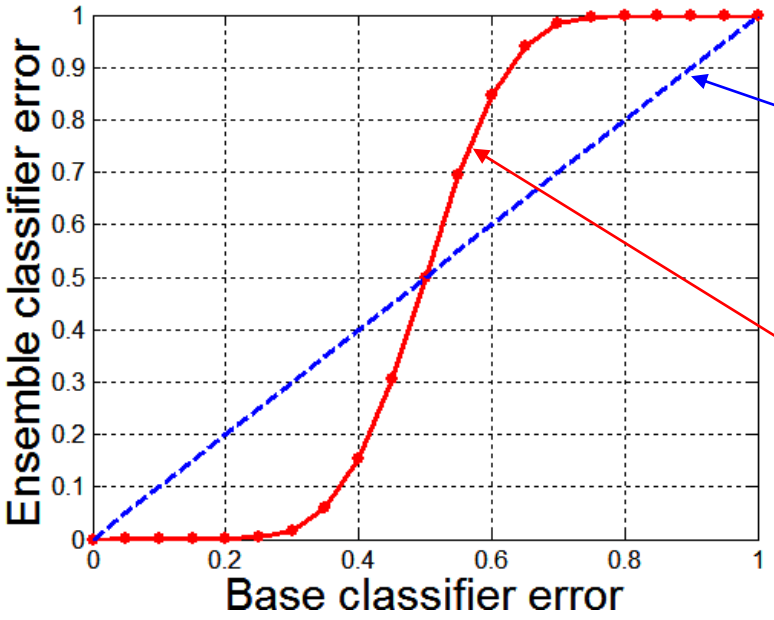
- **Mathematically, the error rate of the ensemble (i.e. probability for wrong prediction) will be**

$$e_{ensemble} = \sum_{i=13}^{25} \binom{25}{i} \varepsilon^{i} (1-\varepsilon)^{25-i} = 0.06$$

- **i.e. much lower than the individual error rate (ε=0.35) of the base classifiers**

Error Graph – Ensemble vs Base Classifier

- **For our example of 25 binary base classifiers**

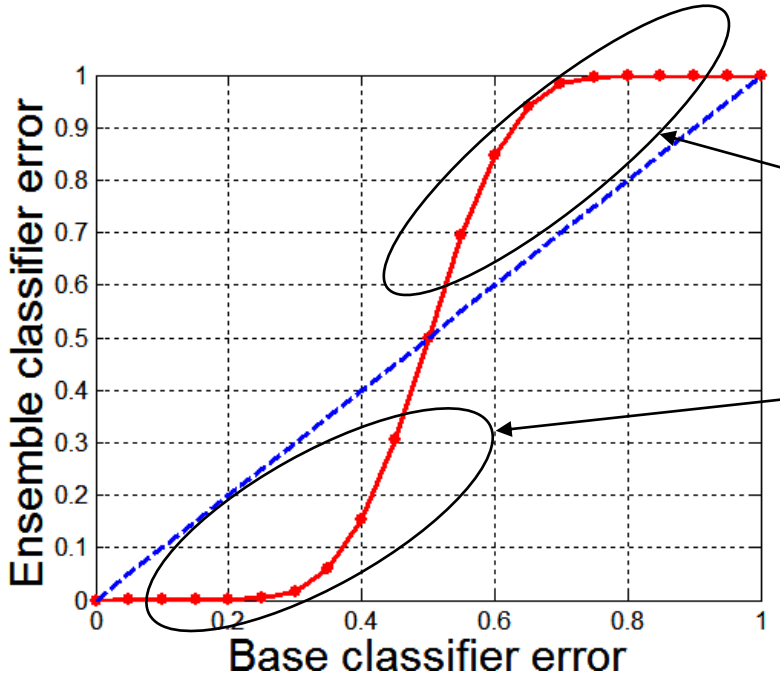


Case 1: base classifiers are identical; ensemble's error = base classifier's error

Case 2: base classifiers are independent; ensemble's error ≠ base classifier's error

Error Graph – Ensemble vs Base Classifier (2)

- **When is the ensemble better and worse than the base classifier?**



Ensemble is worse than a base classifier when $\varepsilon > 0.5$

Ensemble is better than a base classifier when $\varepsilon < 0.5$

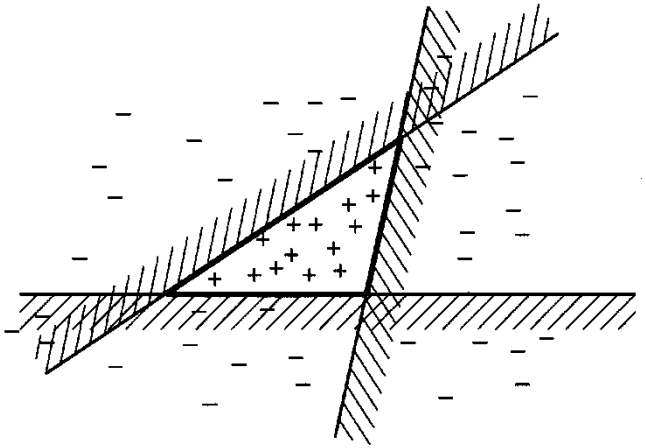
- **The ensemble is better if the error of the base classifier $\varepsilon < 0.5$, i.e. the predictions of the base classifier (which is binary) are better than random guess**

When Ensembles Work Well

- **Conditions for an ensemble to perform better than a single classifier:**
  1) **The base classifiers are good enough, i.e. better than a random guessing (ε <0.5 for binary classifiers)**
  2) **The base classifiers are independent of each other**
     - **In practice, difficult to ensure total independence**
     - **Good results have been achieved when base classifiers are slightly correlated**

An Ensemble Enlarges Hypothesis Space

- **Each single classifier generates 1 hypothesis**
- **An ensemble allows to learn much more expressive hypotheses than the individual ones without much additional computational expenses**



Example:

• 3 linear binary hypotheses, e.g. generated by 3 perceptrons

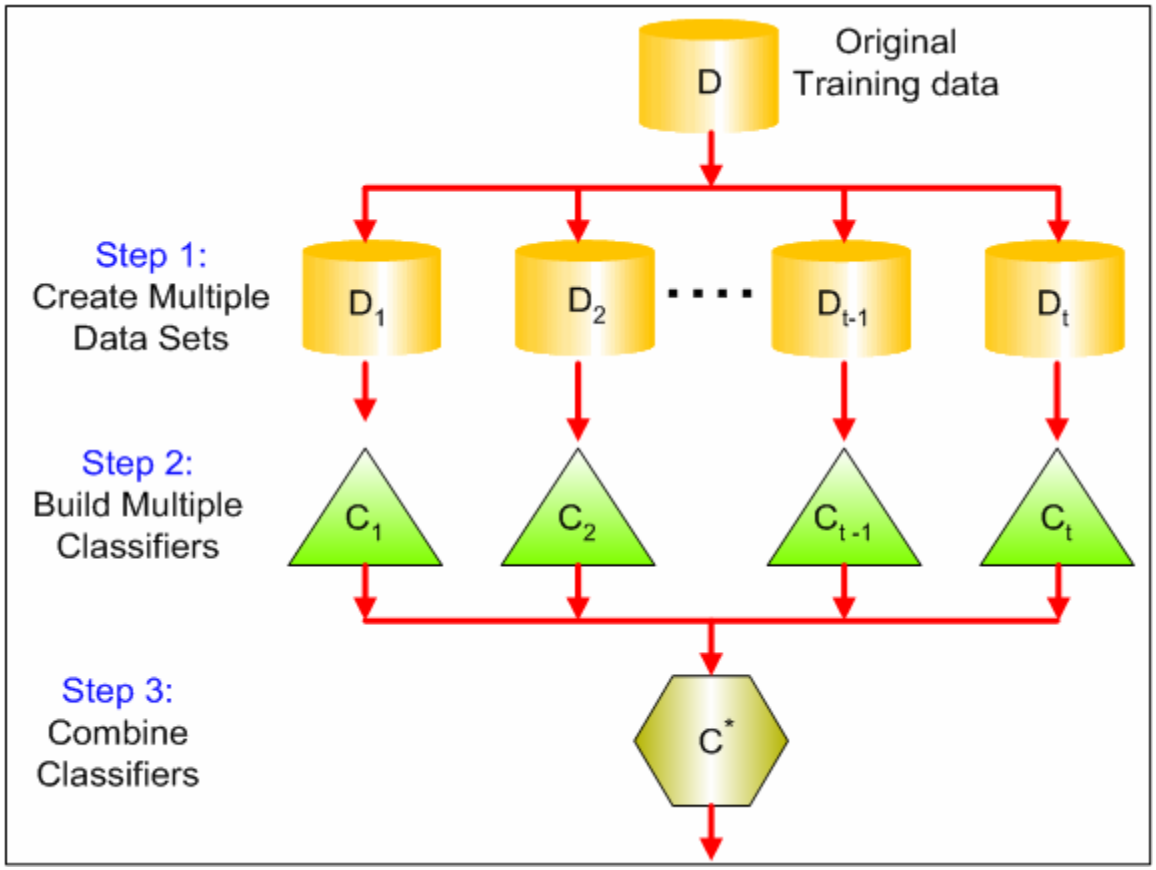
• An ensemble of 3 perceptrons can learn the resulting triangular region - a hypothesis not expressible in the original hypothesis space (of the perceptron)

Methods for Creating Ensembles

- **Effective ensembles consists of base classifiers that are *highly correct* and also *diverse* (i.e. independent, disagree on the training data)**
  - **More generally: if the ensemble members are more accurate than random guessing and their errors are diverse, than the accuracy of the ensemble increases as the ensemble members are added**
- **Methods for creating ensembles focus on generating disagreement among the base classifiers by**
  - **Manipulating the training data – creating multiple training sets by resampling the original data according to some sampling distribution and creating a classifier for each training set (e.g. Bagging and Boosting)**
  - **Manipulating the attributes – using a subset of input features (e.g. Random Forest and Random Subspace)**
  - **Manipulating the class labels – will not be covered (e.g. error-correcting output coding)**
  - **Manipulating the learning algorithm – e.g. building a set of classifiers with different parameters**

Creating Ensembles by Manipulating the Training Data

- **Creating multiple training sets from the original training set by sampling**
- **Examples: Bagging and Boosting**



Picture from Tan, Steinbach and Kumar, Introduction to Data Mining

Bagging

Bagging

- **Bagging stands for "bootstrap aggregation"**
- **Bootstrap sample:**
  - **Given: a dataset D with *n* example (the original dataset)**
  - ***Bootstrap sample* D' from D: contains also *n* examples, randomly chosen from D with replacement (i.e. some examples from D will appear more than once in D', some will not appear at all)**
  - **On average, 63% of the examples in D will also appear in D'**

Dataset with 10 examples:

Original Data	1	2	3	4	5	6	7	8	9	10
Bagging (Round 1)	7	8	10	8	2	5	10	10	5	9
Bagging (Round 2)	1	4	9	1	2	3	2	7	3	2
Bagging (Round 3)	1	8	5	10	5	5	9	6	3	7

Bagging Algorithm

- **Create *M* bootstrap samples**
- **Use each sample to build a classifier**
- **To classify a new example: get the predictions of each classifier and combine them with a majority vote**
  - **i.e. the individual classifiers receive equal weights**

```
model generation
Let n be the number of instances in the training data.
For each of M iterations:
  Sample n instances with replacement from training data.
  Apply the learning algorithm to the sample.
  Store the resulting model (classifier).

classification
For each of the M models:
  Predict class of testing instance using model.
Return class that has been predicted most often.
```

When is Bagging Useful?

- **Typically performs significantly better than the single classifier and is never substantially worse**
- **Effective for *unstable* classifiers**
  - **Unstable: small changes in the training set result in large changes in predictions, e.g. DTs, neural networks**

- **Applying Bagging to numerical prediction (regression)**
  - **Instead of voting on a class, the individual predictions are averaged**
  - **Was shown theoretically that that averaging over multiple models always reduces the expected value of the mean-squared error (not proved for classification)**

Boosting

Boosting

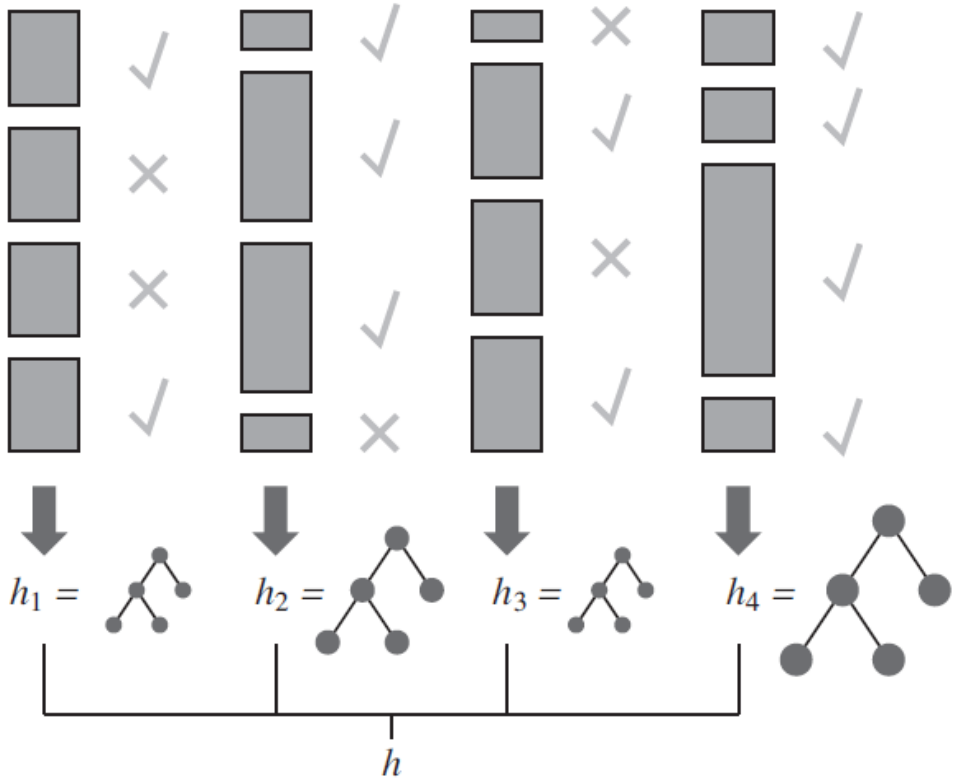
- **The most widely used ensemble method**
- **Idea: Make the classifiers complement each other**
- **How: The next classifier should be created using examples that were difficult for the previous classifiers**

- **This is achieved by using a *weighed training set***
  - **Each training example has an associated weight (≥0)**
  - **The higher the weight, the more difficult the example was to classify by the previous classifiers**
  - **Examples with higher weight have a higher chance to be selected in the training set for the next classifier**

Boosting - Algorithm

- **Initially all training examples have equal weights, e.g. 1/m, m is the number of training examples**
- **From this set, the first classifier (hypothesis) h1 is generated**
- **It classifies some of the training examples correctly and some incorrectly**
- **We would like the next classifier to learn to classify the misclassified examples (i.e. to develop complementary expertise), so we increase the weights of the misclassified examples and decrease the weights of the correctly classified examples**
- **There is a mechanism for selecting examples for the training set of the next classifier; the examples with higher weight are more likely to be selected**
- **From this new weighed set, we generate classifier h2**
- **Continue until K classifiers are generated**
- **Final ensemble: combine the K classifiers using a weighted vote based on how well the classifier performed on the training set**

How the Boosting Algorithm Works



- **1 rectangle corresponds to 1 example**
- **The height of the rectangle corresponds to the weight of the example**
- **√ and X show how the example was classified by the current hypothesis (classifier)**
- **The size of the DT corresponds to the weight of that hypothesis in the final ensemble**

AdaBoost

- **Many boosting algorithms have been proposed**
  - **Different ways to calculate the weights of the training data and use them to select examples for the training set of the next classifier**
  - **Different ways to combine the hypotheses**
- **AdaBoost is the most popular boosting algorithm**
  - **Proposed by Freud and Shapire in 1996**
- **It uses:**
  - **A probability for selecting the examples for the training set of the next classifier**
  - **A weighed majority vote based on the performance of the base classifiers on the training data**

AdaBoost Algorithm

From Miroslav Kubat, An Introduction to ML, 2nd edition, Springer, 2017

AdaBoost ensemble generation

Input: the training set, T, consisting of m examples; and the user's choice of the induction technique

1. Let $i = 1$. For each $\mathbf{x}_j \in T$, set $p_1(\mathbf{x}_j) = 1/m$.
2. Create subset T_i consisting of m examples randomly selected according to the given probabilities. From T_i, induce C_i.
3. Evaluate C_i on each example, $\mathbf{x}_j \in T$.
   Let $e_i(\mathbf{x}_j) = 1$ if C_i misclassified $\mathbf{x}_j$ and $e_i(\mathbf{x}_j) = 0$ otherwise.
   i) Calculate $\epsilon_i = \sum_{j=1}^{m} p_i(\mathbf{x}_j) e_i(\mathbf{x}_j)$;
   ii) Calculate $\beta_i = \epsilon_i / (1 - \epsilon_i)$
4. Modify the probabilities of correctly classified examples by $p_{i+1}(\mathbf{x}_j) = p_i(\mathbf{x}_j) \cdot \beta_i$
5. Normalize the probabilities to make sure that $\sum_{j=1}^{m} p_{i+1}(\mathbf{x}_j) = 1$.
6. Unless a termination criterion has been met, set $i = i + 1$ and go to 2.

until K hypotheses (classifiers) have been generated

Example

From M. Kubat, An Introduction to ML, 2nd ed., Springer, 2017

- **Shows how AdaBoost re-calculates the probabilities**
- **Training set T containing *m*=10 examples: x_1,…, x_{10}**

=> Initial probabilities: $p(x_i)=1/m=1/10=0.1$

$p_1(x_1)$	$p_1(x_2)$	$p_1(x_3)$	$p_1(x_4)$	$p_1(x_5)$	$p_1(x_6)$	$p_1(x_7)$	$p_1(x_8)$	$p_1(x_9)$	$p_1(x_{10})$
0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1

- **Using this probability distribution and sampling with replacement, examples are selected for inclusion in training set T_1. Using T_1, a classifier C_1 is created.**
- **C_1 is tested on T_1. Suppose that it classifies correctly the first 7 examples (*e*=0) and incorrectly the remaining 3 (*e*=1).**
- **=> the weighted error of the classifier is:** $\epsilon_1 = \sum_{j=1}^{10} p_1(x_j) \cdot e_1(x_j) = 0.3$
- **=>the term β for weight modifications is:** $\beta_1 = \epsilon_1/(1-\epsilon_1) = 0.43$
- **The probabilities of the <u>correctly</u> classified examples are modified as:**

$p(x_j) = p(x_j) \cdot \beta_1$

Example (2)

- **New probabilities:**

$p_2(x_1)$	$p_2(x_2)$	$p_2(x_3)$	$p_2(x_4)$	$p_2(x_5)$	$p_2(x_6)$	$p_2(x_7)$	$p_2(x_8)$	$p_2(x_9)$	$p_2(x_{10})$
0.043	0.043	0.043	0.043	0.043	0.043	0.043	0.1	0.1	0.1

- **After normalization:** **The normalization is done by dividing each probability by the sum of all probabilities**

$p_2(x_1)$	$p_2(x_2)$	$p_2(x_3)$	$p_2(x_4)$	$p_2(x_5)$	$p_2(x_6)$	$p_2(x_7)$	$p_2(x_8)$	$p_2(x_9)$	$p_2(x_{10})$
0.07	0.07	0.07	0.07	0.07	0.07	0.07	0.17	0.17	0.17

All 10 probabilities sum to 1

- **Result: the weights of the misclassified examples have increased (from 0.1 to 0.17), while the weights of the correctly classified examples have decreased (from 0.1 to 0.07)**
- **=>At the next iteration, the misclassified examples have a higher chance to be selected for the training set for the next classifier**
- **This is how we make the next classifier focus on the more difficult examples**

AdaBoost - Pseudocode

function ADABOOST($examples$, L, K) **returns** a weighted-majority hypothesis
  inputs: $examples$, set of N labeled examples $(x_1, y_1), \ldots, (x_N, y_N)$
    L, a learning algorithm
    K, the number of hypotheses in the ensemble
  local variables: **w**, a vector of N example weights, initially $1/N$
    h, a vector of K hypotheses
    z, a vector of K hypothesis weights

  for $k = 1$ **to** K **do** // Generating the ensemble of K hypotheses (classifiers)
    $\mathbf{h}[k] \leftarrow L(examples, \mathbf{w})$
    $error \leftarrow 0$
    for $j = 1$ **to** N **do**
      if $\mathbf{h}[k](x_j) \neq y_j$ **then** $error \leftarrow error + \mathbf{w}[j]$
    for $j = 1$ **to** N **do**
      if $\mathbf{h}[k](x_j) = y_j$ **then** $\mathbf{w}[j] \leftarrow \mathbf{w}[j] \cdot error/(1 - error)$
    $\mathbf{w} \leftarrow$ NORMALIZE(**w**)
    $\mathbf{z}[k] \leftarrow \log(1 - error)/error$ // Weights z for the weighted majority vote
  return WEIGHTED-MAJORITY(**h**, **z**)

Calculating the weight wj for each example j based on the probability, which depends on the error

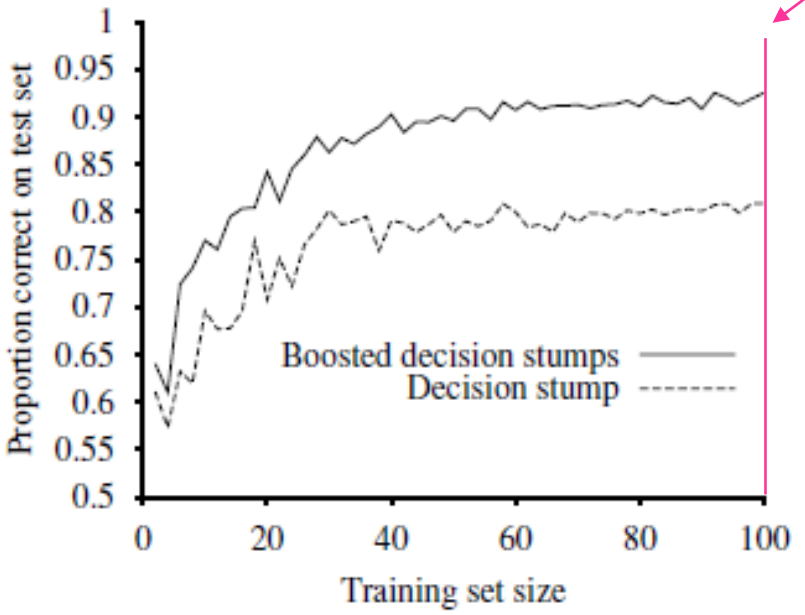
Figure 18.34 The ADABOOST variant of the boosting method for ensemble learning. The algorithm generates hypotheses by successively reweighting the training examples. The function WEIGHTED-MAJORITY generates a hypothesis that returns the output value with the highest vote from the hypotheses in **h**, with votes weighted by **z**.

Important Property of AdaBoost (Boosting Theorem)

- **If the base learning algorithm L is a *weak learning algorithm,* then AdaBoost will return an ensemble that classifies the training data perfectly for large enough K**
- **Weak learner is a classifier which classification performance is slightly better than random guessing (i.e. 50% for binary classification)**
- **Thus, AdaBoost boosts the weak learning algorithm into a strong learning algorithm (on the training data)**
- **This is independent on how expressive the original hypothesis space is, or how complex the function being learnt is**

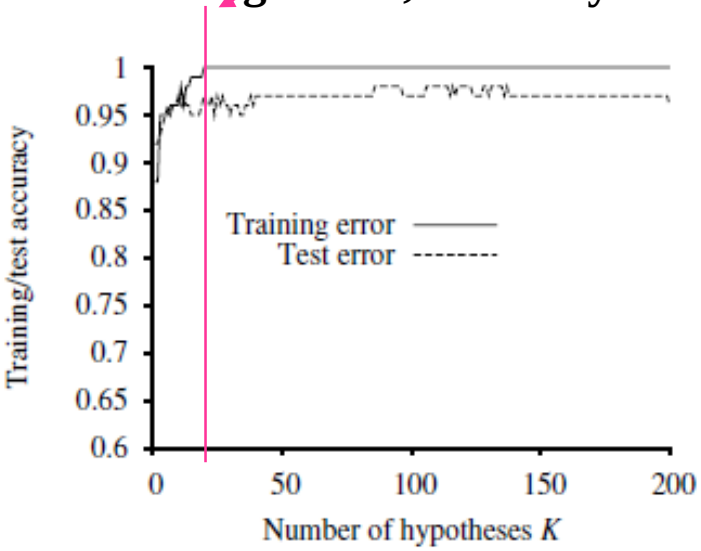
AdaBoost – Example 1

- **Ensembles typically perform better than individual classifiers**
- **Example: boosted decision stumps (AdaBoost ensemble combining 5 decision stumps) vs 1 decision stump**
  - **Decision stump = 1-level DT (1 test node only)**
- **E.g. compare the performances at training set size =100 examples:**
  - **1 decision stump: 80% accuracy on test set**
  - **Ensemble of 5 decision stumps: 93% accuracy on test set**



AdaBoost – Example 2

- **Accuracy vs number of hypotheses K (for a set of 100 training examples)**
- **As we add more base classifiers (decision stumps), the accuracy on the training data increases and reaches 100% for K=20 (consistent with the Boosting Theorem – a perfect fit on the training data is achieved)**
- **As we keep adding more base classifiers:**
  - **The accuracy on the training set remains 100%**
  - **But the accuracy on the test data continues to increase after the accuracy on training data has reached 100%!**
    - **e.g. K=20, accuracy on test set=95%; K=137, accuracy on test set=98%**



- **This result was found across different data sets and classifiers**
- **Contradiction! Ockham's razor – don't make the hypothesis more complex than necessary as this will lead to overfitting**
- **Here - accuracy improves as the ensemble gets more complex**
- **Various explanations, e.g. adding base classifier allows the ensemble to be more definitive in distinguishing between positive and negative ex.**

More on Boosting

- **It was shown that boosting on new data only fails if:**
  - **The individual classifiers are too "complex" for the training data size, or their training errors become "too large" quickly (in a sense described by Schapire et al. 1997)**
- **=> Boosting allows building a powerful combined classifier from very simple ones, e.g. simple DTs generated by 1R (1-level DTs)**
- **Variations**
  - **LogitBoost – Friedman et al. 2000**
  - **Gradient Bosting - Breiman 1997, Friedman 1999**
    - **Different idea - while AdaBoost updates the weights of the examples at each iteration, Gradient Boosting adds a new model that minimizes the error of the previous model**

Boosting and Bagging – Similarities & Differences

- **Similarities**
  - **Use voting (for classification) and averaging (for prediction) to combine the outputs of the individual learners**
  - **Combine classifiers of the same type, e.g. DTs**
- **Differences**
  - **In Bagging the ensemble members are built separately; in boosting they are built iteratively – the new ensemble members are influenced by the performance of the previous ones**
    - **A new ensemble member is encouraged to become an expert for the examples classified incorrectly by the previous ensemble member**
    - **Intuitive justification: ensemble members should be experts that complement each other**
  - **Combing the opinions of the individual ensemble members:**
    - **Bagging – equal weighting (all experts are equally influential)**
    - **Boosting – weighed based on performance (i.e. some experts are more influential)**

Random Forest

Creating Ensembles by Manipulating the Attributes (Features)

- **Each base classifier uses only a subset of the features**
- **E.g. training data described with K features, create an ensemble of M classifiers each using only L features (L<K) from the training data**

  1) Create multiple (smaller) subsets of features from the original feature set by randomly selecting the features

  2) The result is a multiple versions of the training data, each containing only the selected features

  3) Build a classifier for each version of the training data

  4) Combine predictions with a majority vote
- **Examples: Random Subspace (similar to above) and Random Forest**
- **Random Forest**
  - **Combines decision trees**
  - **Bagging to generate bootstrap samples + uses only a subset of the available attributes for selecting the most important attribute**

Random Forest – Bagging of Random Trees

Given:

n **- number of training examples, *m* – number of all features, *k* – number of features to be used by each ensemble member (k<m), *M* – number of ensemble members**

Create RandomForest of *M* trees:

For each of *M* iteration

1. Bagging – generate a bootstrap sample

Sample *n* instances with replacement from training data

2. Random feature selection for selecting the best attribute

Grow decision tree without pruning. At each step select the best feature to split on by considering only *k* randomly selected features and calculating IG.

Classification:

Apply the new example to each of the *t* decision trees starting from the root. Assign it to the class corresponding to the leaf. Combine the decisions of the individual trees by majority voting.

Random Forest - Discussion

- **Performance (as in any ensemble) depends on**
  - **Accuracy of the individual trees (strength)**
  - **Correlation with each other**
- **Ideally we would like highly accurate individual trees and less correlated**
- **Bagging and random feature selection are used to generate diversity and reduce the correlation**
- **The number of features *k***
  - **As *k* increases, both the strength and the correlation increase**
  - **Rule of thumb: $k=\log_2 m$**
- **Random Forest typically outperforms a single DT**
- **It was proven that Random Forest does not overfit**
- **Random Forest is faster than AdaBoost and gives comparable accuracy results**

Summary

- **Ensembles combine the predictions of several classifiers**
- **They work when the individual classifiers are accurate and diverse (uncorrelated, do not make the same mistakes)**
- **Diversity is generated by manipulating the**
  - **training data (Bagging, Boosting)**
  - **attributes (Random Forest = bagging + random selection of attributes)**
  - **learning algorithm**
- **Some ensembles combine classifiers of the same type, some not**
- **Most ensembles use a majority vote to make predictions on new data, others used a weighted vote**
- **Ensembles are often the winning solution in ML competitions**

Textbook on Ensembles

- **L. I. Kuncheva (2014). Combining Pattern Classifiers: Methods and Algorithms, 2nd Edition, Wiley.**

  http://au.wiley.com/WileyCDA/WileyTitle/productCd-1118315235.html