Lecture 20: Ensemble Methods

Andreas Wichert
Department of Computer Science and Engineering
Técnico Lisboa

No Free Lunch Theorem

- An algorithm performs well on a certain class of problems then it necessarily pays for that with degraded performance on the set of all remaining problems
- Averaged over all problems, no search/optimization algorithm is expected to perform better than any other search/optimization algorithm.
- Any algorithm that is good at some problems is bad at others.
    - Can we “combine” predictions from multiple models?

No Free Lunch Theorem

For any two learning algorithms $P_1(h|D)$ and $P_2(h|D)$, the following are true, independent of the sampling distribution $P(x)$ and the number n of training points:

1. Uniformly ***averaged*** over all ***target functions F***,
   $\mathcal{E}_1(E|F,n) - \mathcal{E}_2(E|F,n) = 0$.
   - Further, no matter what algorithm we use, there is at least one target function for which random guessing Is better
2. For any ***fixed training set D***, uniformly ***averaged over F***,
   $\mathcal{E}_1(E|F,D) - \mathcal{E}_2(E|F,D) = 0$
   - If a learning system performs well over some set of problems (better than average), it must perform worse than average elsewhere
3. Uniformly averaged over ***all priors P(F)***,
   $\mathcal{E}_1(E|n) - \mathcal{E}_2(E|n) = 0$
4. For any ***fixed training set D***, uniformly ***averaged over P(F***),
   $\mathcal{E}_1(E|D) - \mathcal{E}_2(E|D) = 0$

Value of Ensembles

- No Free Lunch Theorem
  - No single algorithm wins all the time!
- When combing multiple **independent** and **diverse** decisions each of which is at least more accurate than random guessing, random errors cancel each other out, correct decisions are reinforced.
- Examples: Human ensembles are demonstrably better
  - How many jelly beans in the jar?: Individual estimates vs. group average.

Ensemble Learning

- So far – learning methods that learn a *single hypothesis*, chosen form a hypothesis space that is used to make predictions
- Ensemble learning:
  - Select a collection (ensemble) of hypotheses and combine their predictions
- Example
  - Generate 100 different decision trees from the same or different training set and have them vote on the best classification for a new example.
- Key motivation
  - Reduce the error rate. Hope is that it will become much more unlikely that the ensemble will misclassify an example.

Example: Weather Forecast

Reality	⛈	☀	☀	⛈	⛈	☀	☀
1	⛈	⛈ ✗	☀	☀ ✗	⛈	☀	⛈ ✗
2	☀ ✗	☀	☀	☀ ✗	⛈	☀	⛈ ✗
3	⛈	☀	⛈ ✗	⛈	☀ ✗	⛈ ✗	☀
4	⛈	☀	⛈ ✗	⛈	☀ ✗	☀	☀
5	⛈	⛈ ✗	☀	⛈	⛈	⛈ ✗	☀
Combine	⛈	☀	☀	⛈	⛈	☀	☀

Model Combination

The model contains a binary latent variable **c** that indicates which component of the mixture is responsible for generating the corresponding data point.

Latent: (latin hidden), are variables that are not directly observed but are rather inferred.

x is obtained by marginalizing over the latent variable

$$p(\mathbf{x}) = \sum_{\mathbf{c}} p(\mathbf{x}, \mathbf{c})$$

In the case of our Gaussian mixture example, this leads to a distribution of the form

$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \cdot \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \Sigma_k)$$

Model Combination

Training set consists on N observations (sample)

$$X = (\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_\eta, \cdots, \mathbf{x}_N)$$

$$p(X) = \prod_{k=1}^{K} p(\mathbf{x}_n) = \prod_{k=1}^{K} \left(\sum_{\mathbf{c}_n} p(\mathbf{x}_n, \mathbf{c}_n) \right)$$

- Each observed data point x_n has a corresponding latent variable c_n

Bayesian Model Averaging

For different models h

$$h = 1, 2, \cdots, H$$

the marginal distribution os given by

$$p(\mathbf{x}) = \sum_{h=1}^{H} p(X|h) \cdot p(h)$$

This is an example of Bayesian model averaging.

Bayesian Model Averaging

$$p(\mathbf{x}) = \sum_{h=1}^{H} p(X|h) \cdot p(h)$$

This is an example of Bayesian model averaging.

The interpretation: one model is responsible for generating the whole data set, and the probability distribution over h reflects our uncertainty as to which model that is.

As the size of the data set increases, this uncertainty reduces, and the posterior probabilities $p(h|X)$ become increasingly focussed on just one of the models.

- Key difference between Bayesian model averaging and model combination
- In Bayesian model averaging the whole data set is generated by a single model
- When we combine multiple models, different data points within the data set can potentially be generated from different values of the latent variable c and hence by different components.

Committees

- The simplest way to construct a committee is to average the predictions of a set of individual models
- When we averaged a set of low-bias models (corresponding to higher order polynomials), we obtained accurate predictions for the underlying sinusoidal function from which the data were generated.
- In practice, of course, we have only a single data set, and so we have to find a way to introduce variability between the different models within the committee.

Bootstrap a data set

- Sampling a dataset with replacement
- Define: Size of the sample and the number of repeats.
- Example:
    - *(0.1, 0.2, 0.3, 0.4, 0.5, 0.6)*
    - Randomly choose the first observation from the dataset
    - *sample = (0.2)*
- This observation is returned to the dataset and we repeat this step 3 more times.
    - *sample = (0.2, 0.1, 0.2, 0.6)*

Bagging

Suppose we generate M bootstrap data sets and then use each to train a separate copy $y_m(\mathbf{x})$ of a predictive model where $m = 1, ..., M$.

The committee prediction is given by

$$y_{COM}(\mathbf{x}) = \frac{1}{M} \sum_{m=1}^{M} y_m(\mathbf{x})$$

This procedure is known as bootstrap aggregation or bagging

Bagging

Suppose the true regression function that we are trying to predict is given by $h(\mathbf{x})$

Output of each of the models can be written as the true value plus an error

$$y_m(\mathbf{x}) = h(\mathbf{x}) + \epsilon$$

$$\mathbb{E}_X\left((y_m(\mathbf{x}) - h(\mathbf{x}))^2\right) = \mathbb{E}_X\left(\epsilon_m(\mathbf{x})^2\right)$$

The average error made by the models acting individually is therefore

$$\mathbb{E}_{AV} = \frac{1}{M}\sum_{m=1}^{M}\mathbb{E}_X\left(\epsilon_m(\mathbf{x})^2\right)$$

The average error made by the models acting individually is therefore

$$\mathbb{E}_{AV} = \frac{1}{M}\sum_{m=1}^{M} \mathbb{E}_X\left(\epsilon_m(\mathbf{x})^2\right)$$

The expected error from the committee is

$$\mathbb{E}_{COM} = \mathbb{E}_X\left(\left(\frac{1}{M}\sum_{m=1}^{M}(y_m(\mathbf{x}) - h(\mathbf{x})\right)^2\right)$$

$$\mathbb{E}_{COM} = \mathbb{E}_X\left(\left(\frac{1}{M}\sum_{m=1}^{M}\epsilon_m(\mathbf{x})\right)^2\right)$$

If we assume that the errors have zero mean and are uncorrelated,

$$\mathbb{E}_X\left(\epsilon_m(\mathbf{x})\right) = 0$$

and

$$\mathbb{E}_X\left(\epsilon_m(\mathbf{x}) \cdot \epsilon_l(\mathbf{x})\right) = 0, \quad m \neq l$$

then we can rewrite

$$\mathbb{E}_{COM} = \frac{1}{M^2}\mathbb{E}_X\left(\left(\sum_{m=1}^{M}\epsilon_m(\mathbf{x})\right)^2\right)$$

$$\mathbb{E}_{COM} = \frac{1}{M^2}\sum_{m=1}^{M}\mathbb{E}_X\left(\epsilon_m(\mathbf{x})^2\right)$$

$$\mathbb{E}_{COM} = \frac{1}{M}\mathbb{E}_{AV}$$

Bagging

$$\mathbb{E}_{COM} = \frac{1}{M}\mathbb{E}_{AV}$$

The average error of a model can be reduced by a factor of M simply by averaging M versions of the model

Key assumption: errors of the individual models are uncorrelated.

In practice, the errors are typically highly correlated, and the reduction in overall error is generally small.

$$\mathbb{E}_{AV} = \frac{1}{M}\sum_{m=1}^{M}\mathbb{E}_X\left(\epsilon_m(\mathbf{x})^2\right) = \mathbb{E}_X\left(\sum_{m=1}^{M}\frac{1}{M}\epsilon_m(\mathbf{x})^2\right)$$

and

$$\mathbb{E}_{COM} = \mathbb{E}_X\left(\left(\frac{1}{M}\sum_{m=1}^{M}\epsilon_m(\mathbf{x})\right)^2\right)$$

$$\left(\sum_{m=1}^{M}\frac{1}{M}\epsilon_m(\mathbf{x})\right)^2 \leq \sum_{m=1}^{M}\frac{1}{M}\epsilon_m(\mathbf{x})^2$$

$$\frac{1}{M}\left(\sum_{m=1}^{M}\epsilon_m(\mathbf{x})\right)^2 \leq \sum_{m=1}^{M}\epsilon_m(\mathbf{x})^2$$

Bagging

and follows that is always (correlation between the models)

$$\mathbb{E}_{COM} \leq \mathbb{E}_{AV}$$

and in the best case in which the errors of the individual models are uncorrelated.

$$\mathbb{E}_{COM} = \frac{1}{M}\mathbb{E}_{AV}$$

Strong and Weak Learners

- Strong Learner:
  - Objective of machine learning
  - Take labeled data for training
  - Produce a classifier which can be *arbitrarily accurate*

- Weak Learner
  - Take labeled data for training
  - Produce a classifier which is *more accurate than random guessing*

Boosting

- Weak Learner: only needs to generate a hypothesis with a training accuracy greater than 0.5, i.e., < 50% error over any distribution
- Learners
  - Strong learners are very difficult to construct
  - Constructing weaker Learners is relatively easy
- Questions: Can a set of **weak learners** create a single **strong learner** ?

Yes: Boost weak classifiers to a strong learner

Boosting

- Originally developed by computational learning theorists to guarantee performance improvements on fitting training data for a ***weak learner*** that only needs to generate a hypothesis with a training accuracy greater than *0.5* (Schapire, 1990).
- Revised to be a practical algorithm, AdaBoost, for building ensembles that empirically improves generalization performance (Freund & Shapire, 1996).

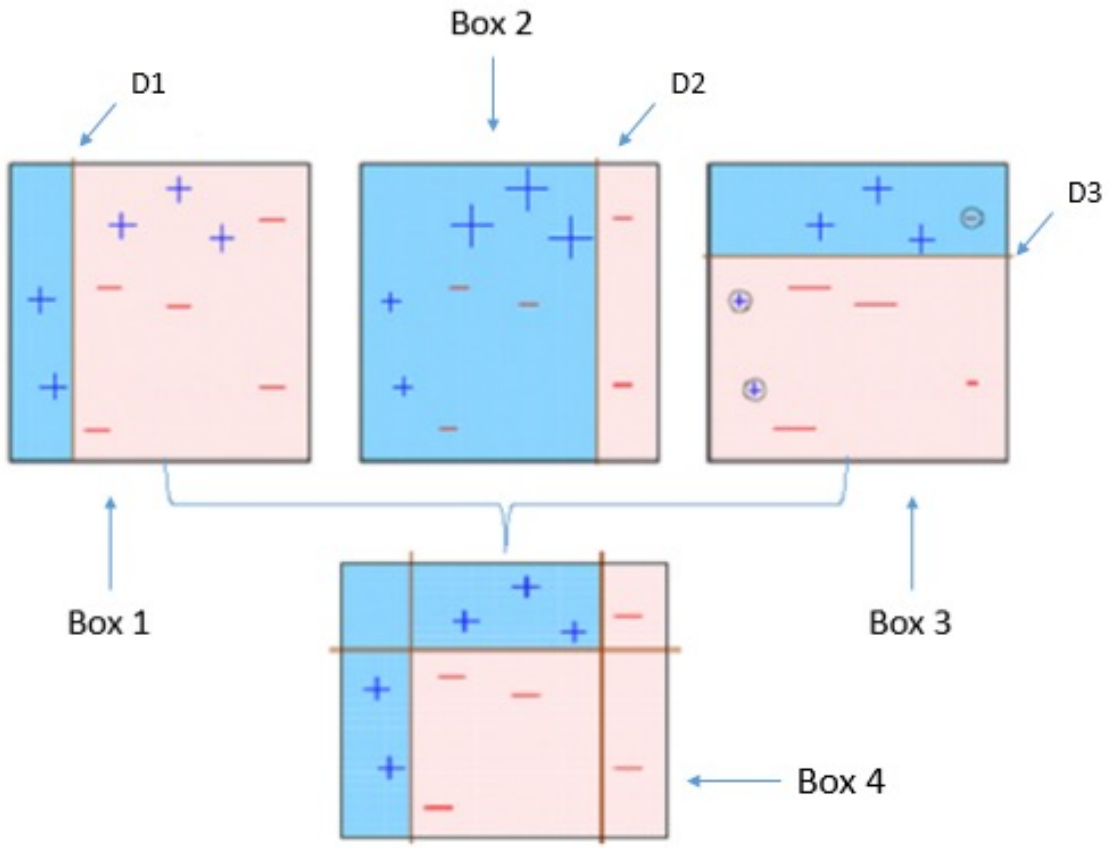
Boosting

- Boosting is a powerful technique for combining multiple base (weak) classifiers to produce a form of committee whose performance can be significantly better than that of any of the base classifiers.

- Boosting can give good results even if the base classifiers have a performance that is only *slightly better than random*, and hence sometimes the base classifiers are known as weak learners.

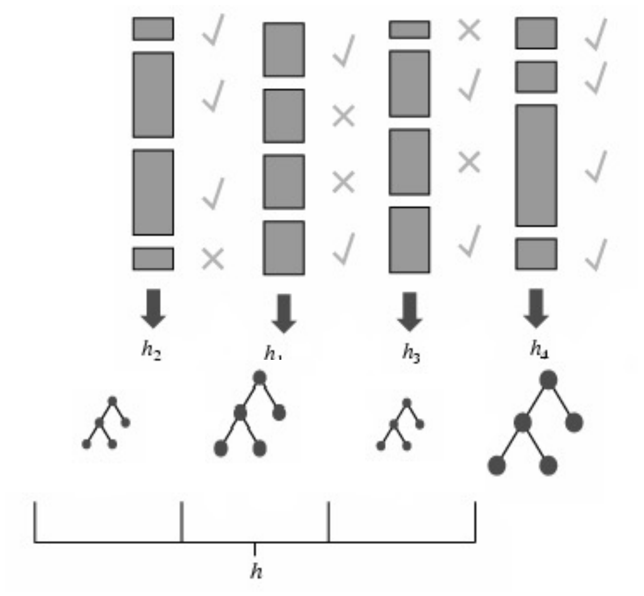
Boosting

- Revised to be a practical algorithm, AdaBoost, for building ensembles that empirically improves generalization performance (Freund & Shapire, 1996).
- Instead of sampling (as in bagging) re-weigh examples!
- Examples are **given weights**.
    - At each iteration, a new hypothesis is learned (weak learner) and the *examples are reweighted* to focus the system on examples that the most recently learned classifier got wrong.
- Final classification based on **weighted vote of weak classifiers**



https://www.analyticsvidhya.com/blog/2015/11/quick-introduction-boosting-algorithms-machine-learning/

Adaptive Boosting



Each rectangle corresponds to an example, with weight proportional to its height.

Crosses correspond to *misclassified* examples.

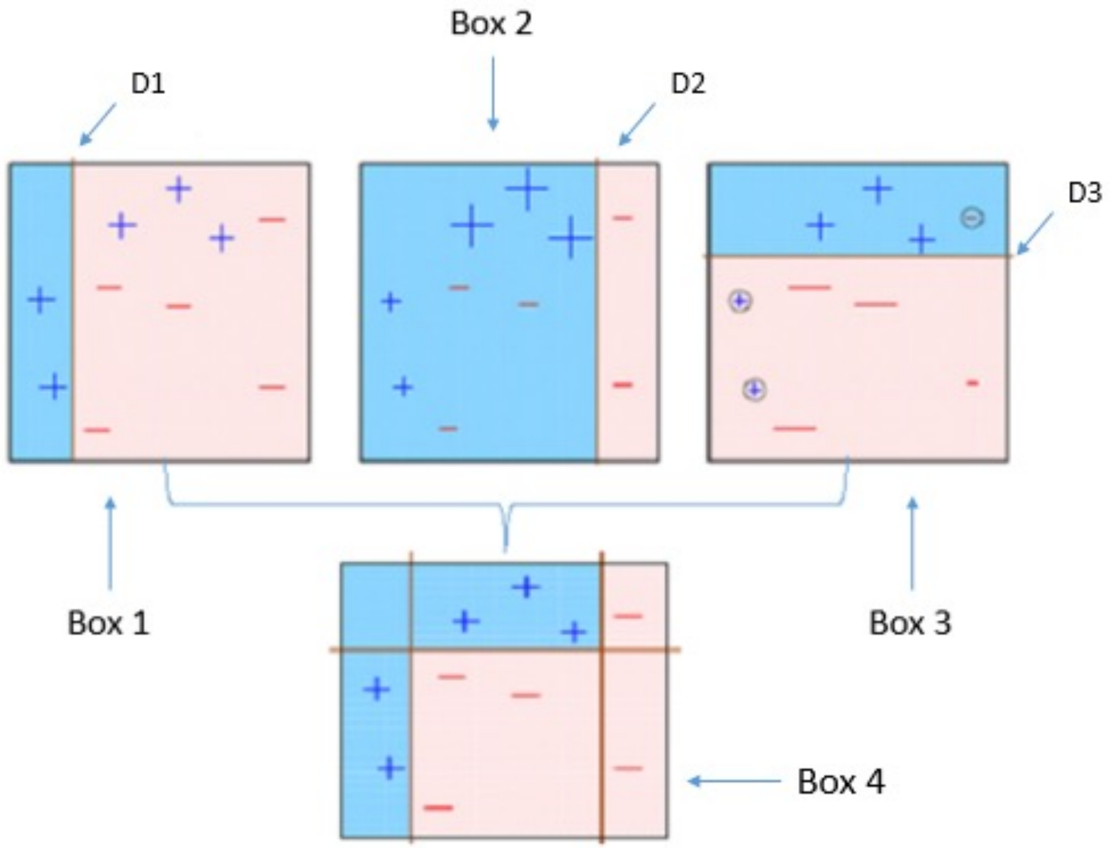
Size of decision tree indicates *the weight of that hypothesis* in the final ensemble.

Construct Weak Classifiers

- Using Different Data Distribution
  - Start with **uniform weighting**
  - During each step of learning
    - *Increase weights* of the examples which are *not correctly learned* by the weak learner
    - *Decrease weights* of the examples which are *correctly learned* by the weak learner
- Idea
  - Focus on difficult examples which are not correctly classified in the previous steps

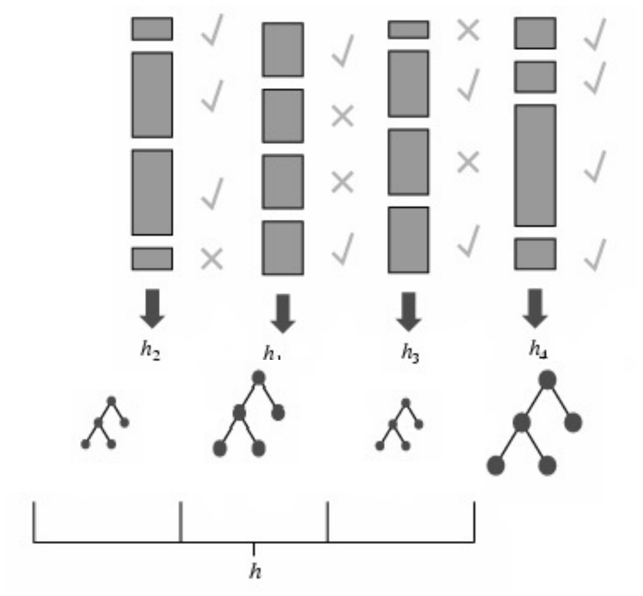
Combine Weak Classifiers

- Weighted Voting
  - Construct strong classifier by weighted voting of the weak classifiers
- Idea
  - **Better** weak classifier gets a **larger weight**
  - Iteratively add weak classifiers
    - Increase accuracy of the combined classifier through minimization of a cost function



https://www.analyticsvidhya.com/blog/2015/11/quick-introduction-boosting-algorithms-machine-learning/

Adaptive Boosting



Each rectangle corresponds to an example, with weight proportional to its height.

Crosses correspond to *misclassified* examples.

Size of decision tree indicates *the weight of that hypothesis* in the final ensemble.

Difference between Bagging and Boosting

- Base classifiers are trained in sequence, each base classifier is trained using a weighted form of the data set in which the weighting coefficient associated with each data point depends on the performance of the previous classifiers.
- Points that are misclassified by one of the base classifiers are given greater weight when used to train the next classifier in the sequence-
- Once all the classifiers have been trained, their predictions are then combined through a weighted majority voting scheme.

Boosting

- Originally developed by computational learning theorists to guarantee performance improvements on fitting training data for a ***weak learner*** that only needs to generate a hypothesis with a training accuracy greater than *0.5* (Schapire, 1990).
- Revised to be a practical algorithm, AdaBoost, for building ensembles that empirically improves generalization performance (Freund & Shapire, 1996).

AdaBoost

Two-class classification problem, in which the training data comprises input vectors

$$(\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_\eta, \cdots, \mathbf{x}_N)$$

and binary target variables

$$(t_1, t_2, \cdots, t_\eta, \cdots, t_N), \qquad t_\eta \in \{-1, 1\}$$

Each data point is given an associated weighting parameter $\omega_\eta = 1/N$

We shall suppose that we have a procedure available for training a base classifier using weighted data to give a function (there are $m = 1, ..., M$ base clasifiers)

$$y_m(\mathbf{x}_\eta) \in \{-1, 1\}$$

AdaBoost

- At each stage of the algorithm trains a new classifier out of *M* classifiers using a data set in which the weighting coefficients are adjusted according to the performance of the previously trained classifier
- It give greater weight to the misclassified data points
- When the base classifiers have been trained, the *M* classifiers are combined to form a committee using coefficients that give different weight to different base classifiers

AdaBoost Algorithm

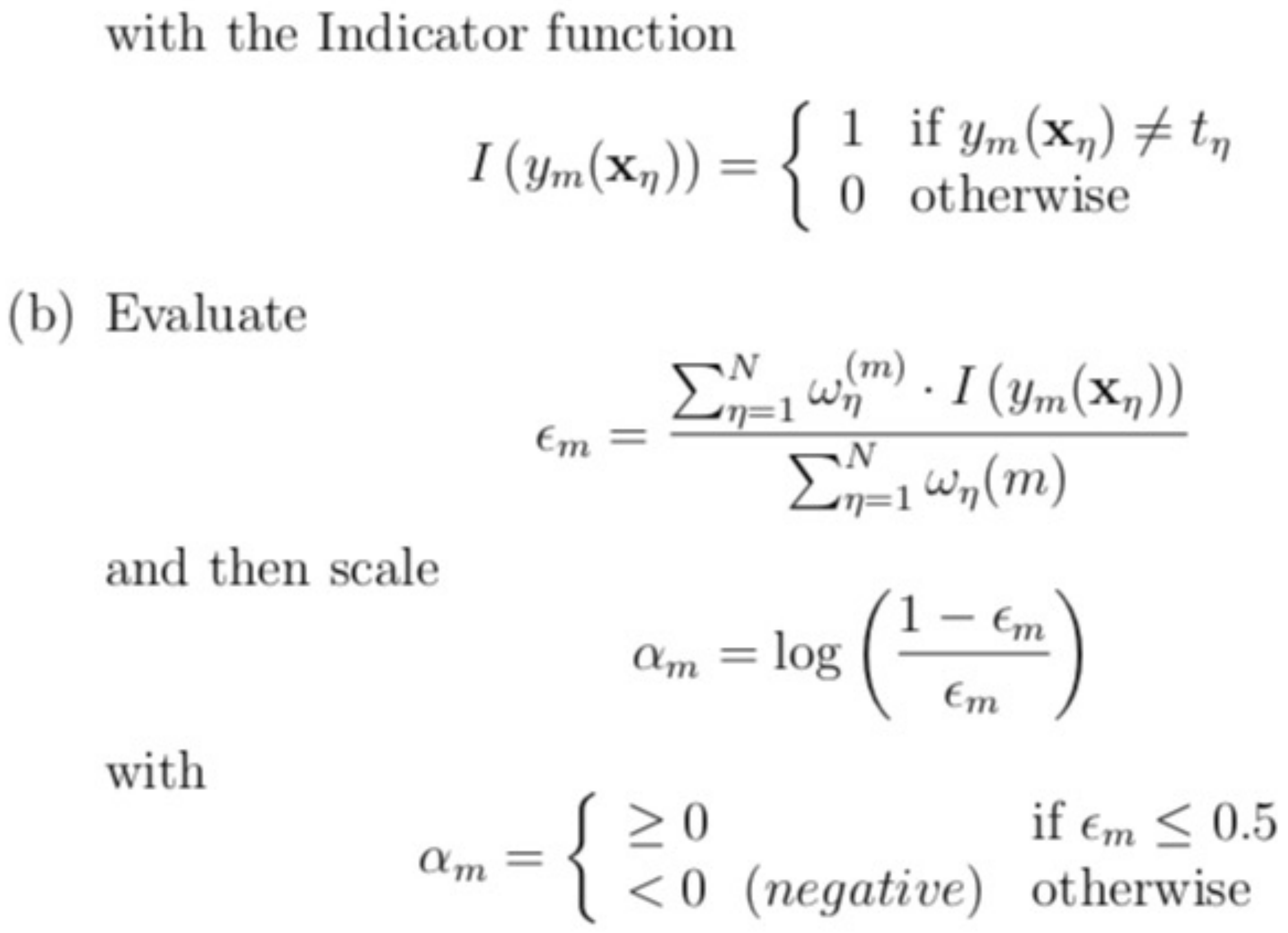
1. Initialize the data weighting coefficients $\{\omega_\eta\}$ to $\omega_\eta^{(1)} = 1/N$ for $\eta = (1, 2, \cdots, N)$ (for all data points)
2. For $m = 1, ..., M$ (for M classifiers):
   (a) Fit (train) a classifier $y_m(\mathbf{x}_\eta)$ to the training data by minimizing the weighted error function for all training patterns

$$J_m = \sum_{\eta=1}^{N} \omega_\eta^{(m)} \cdot I\left(y_m(\mathbf{x}_\eta)\right)$$

   with the Indicator function

$$I\left(y_m(\mathbf{x}_\eta)\right) = \begin{cases} 1 & \text{if } y_m(\mathbf{x}_\eta) \neq t_\eta \\ 0 & \text{otherwise} \end{cases}$$

AdaBoost Algorithm



with the Indicator function

$$I\left(y_m(\mathbf{x}_\eta)\right) = \begin{cases} 1 & \text{if } y_m(\mathbf{x}_\eta) \neq t_\eta \\ 0 & \text{otherwise} \end{cases}$$

(b) Evaluate

$$\epsilon_m = \frac{\sum_{\eta=1}^{N} \omega_\eta^{(m)} \cdot I\left(y_m(\mathbf{x}_\eta)\right)}{\sum_{\eta=1}^{N} \omega_\eta(m)}$$

and then scale

$$\alpha_m = \log\left(\frac{1-\epsilon_m}{\epsilon_m}\right)$$

with

$$\alpha_m = \begin{cases} \geq 0 & & \text{if } \epsilon_m \leq 0.5 \\ < 0 & (negative) & \text{otherwise} \end{cases}$$

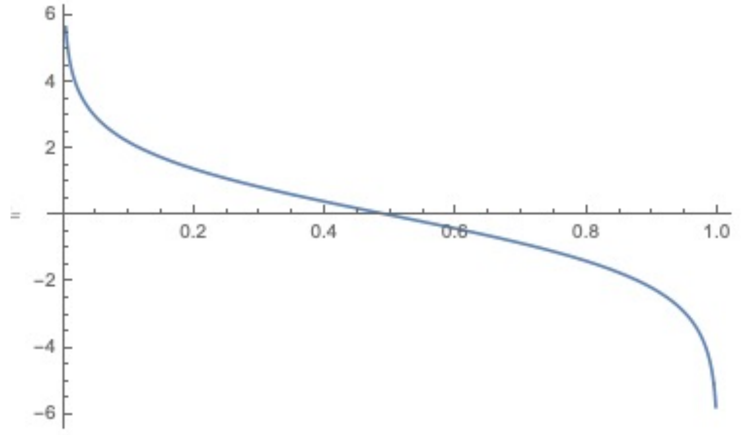
$$\epsilon_m = \frac{\sum_{\eta=1}^{N} \omega_\eta^{(m)} \cdot I\left(y_m(\mathbf{x}_\eta)\right)}{\sum_{\eta=1}^{N} \omega_\eta(m)}$$

and then scale

$$\alpha_m = \log\left(\frac{1-\epsilon_m}{\epsilon_m}\right)$$

with

$$\alpha_m = \begin{cases} \geq 0 & \text{if } \epsilon_m \leq 0.5 \\ < 0 \ \ (\textit{negative}) & \text{otherwise} \end{cases}$$

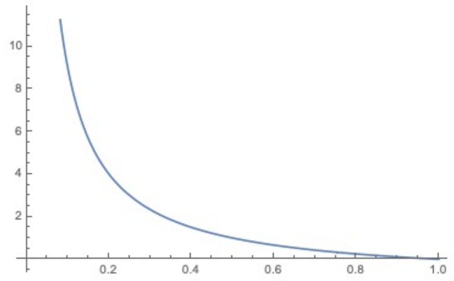


(c) Update the N data weighting coefficients for the next classifier $(m+1)$

$$\omega_\eta^{(m+1)} = \omega_\eta^{(m)} \cdot e^{\alpha_m \cdot I(y_m(\mathbf{x}_\eta))}$$

can be written as well as

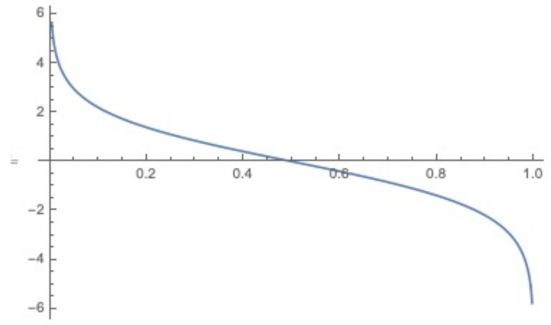
$$\omega_\eta^{(m+1)} = \begin{cases} \omega_\eta^{(m)} \cdot \left(\frac{1-\epsilon_m}{\epsilon_m}\right) & \text{if } I\left(y_m(\mathbf{x}_\eta)\right) = 1 \\ \omega_\eta^{(m)} & \text{otherwise} \end{cases}$$



Make predictions using the final model,

$$Y_M(\mathbf{x}) = sign\left(\sum_{m=1}^{M} \alpha_m \cdot y_m(\mathbf{x})\right)$$

with $sign$ as defined for Perceptrons



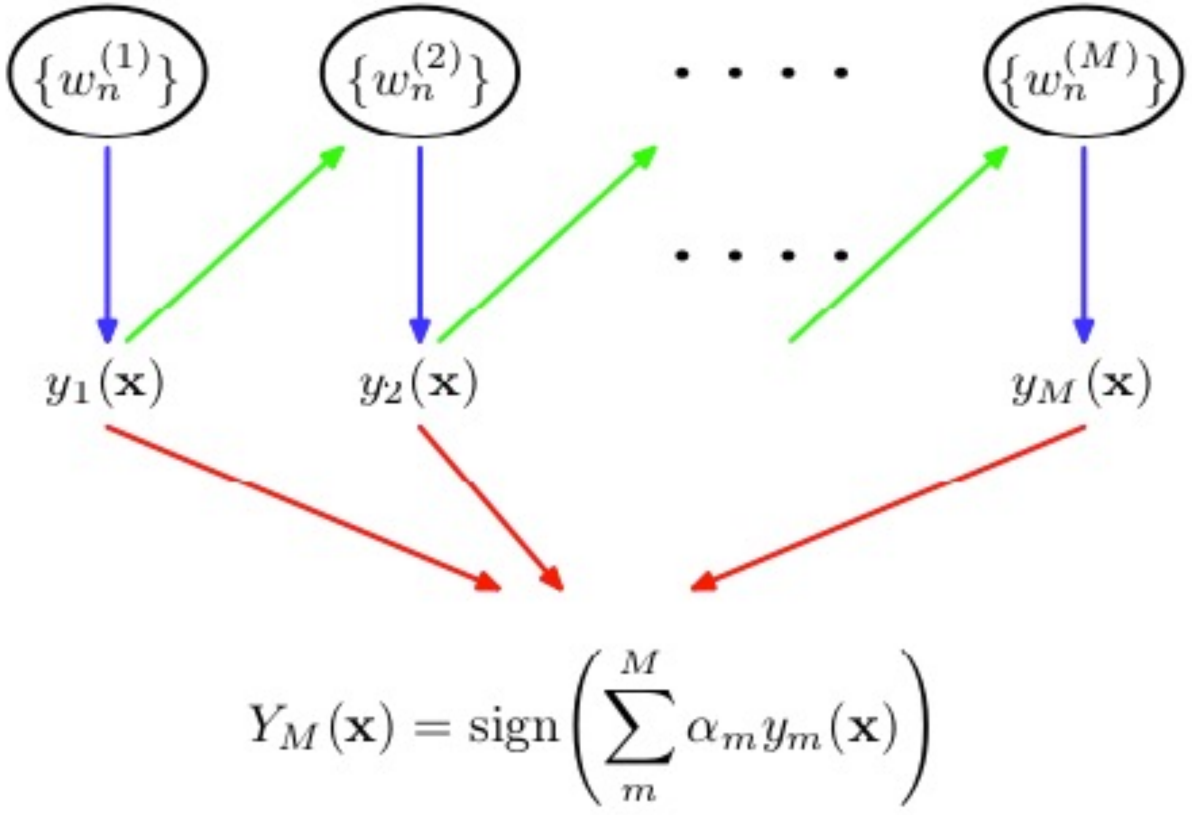
Make predictions using the final model,

$$Y_M(\mathbf{x}) = sign\left(\sum_{m=1}^{M} \alpha_m \cdot y_m(\mathbf{x})\right)$$

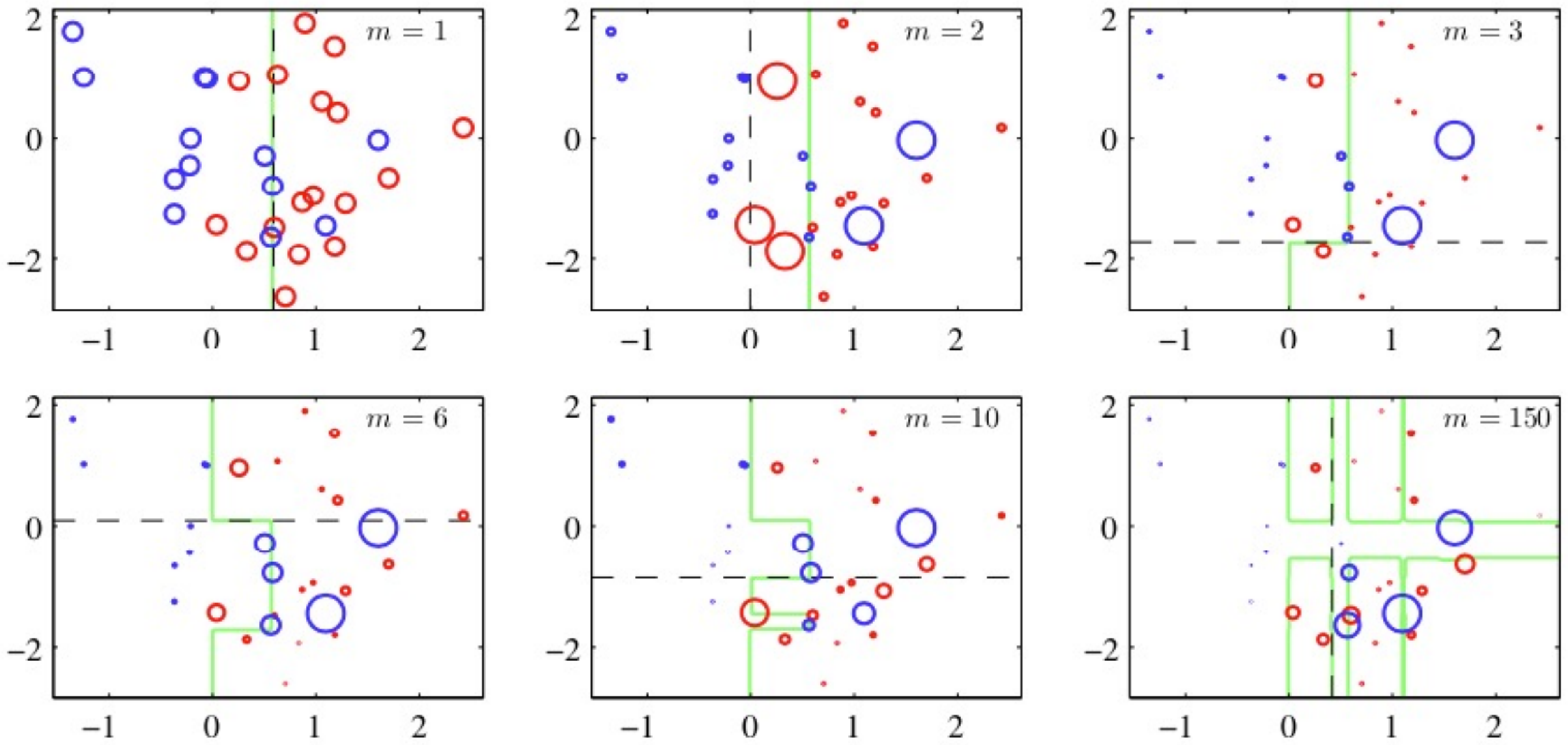
with $sign = sgn_0$ as defined for Perceptrons

α_m gives greater weight to the more accurate classifiers when computing the overall output

AdaBoost


$\{w_n^{(1)}\}$
$\{w_n^{(2)}\}$
$\{w_n^{(M)}\}$
$y_1(\mathbf{x})$
$y_2(\mathbf{x})$
$y_M(\mathbf{x})$
$Y_M(\mathbf{x}) = \mathrm{sign}\left(\sum_m^M \alpha_m y_m(\mathbf{x})\right)$

- The first base classifier $y^1(\boldsymbol{x})$ is trained using weighting coefficients $\omega_\eta^{(1)}$ that are all equal, which therefore corresponds to the usual procedure *n* for training a single classifier.
- The weighting coefficients $\omega_\eta^{(m)}$ are increased for data points $\boldsymbol{x}_\eta$ that are misclassified and decreased for data points that are correctly classified.
- Classifiers are therefore forced to place greater emphasis on points that have been misclassified by previous classifiers, and data points that continue to be misclassified by successive classifiers receive ever greater weight



$m = 1$
$m = 2$
$m = 3$
$m = 6$
$m = 10$
$m = 150$
2
0
−2
−1
0
1
2

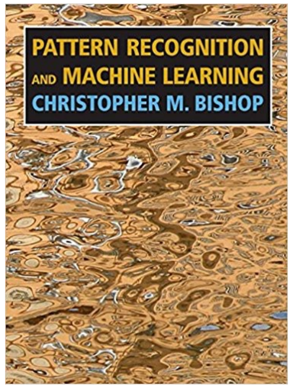
Experimental Results on Ensembles
(Freund & Schapire, 1996; Quinlan, 1996)

- Ensembles have been used to improve generalization accuracy on a wide variety of problems.

- On average**, Boosting** provides a larger increase in accuracy than **Bagging**.

- Boosting on rare occasions can degrade accuracy.
- Boosting is particularly subject to over-fitting when there is significant noise in the training data.

Literature



- Christopher M. Bishop, Pattern Recognition and Machine Learning (Information Science and Statistics), Springer 2006
  - Chapter 14