University of Rome "La Sapienza"

Master in Artificial Intelligence and Robotics

Machine Learning

A.Y. 2018/2019

Prof. Luca Iocchi

Sapienza University of Rome, Italy
Master in Artificial Intelligence and Robotics
Machine Learning (2018/19)

18. Multiple learners

Luca Iocchi

Overview

- Combining multiple learners
- Voting
- Bagging
- Boosting
- AdaBoost

Reference
E. Alpaydin. Introduction to Machine Learning. Chapter 17.
C. Bishop. Pattern Recognition and Machine Learning. Chapter 14.

Multiple learners

General idea: instead of training a complex learner/model, train many different learners/models and then combine their results.

Committees: set of models trained on a dataset.

Models can be trained in parallel (*voting* or *bagging*) or in sequence (*boosting*).
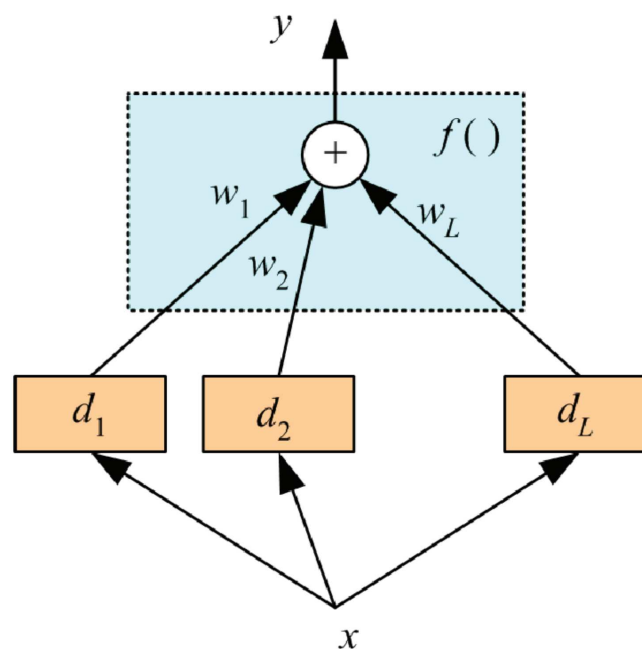
Voting

Given a dataset D

1. use D to train a model $y_m(\mathbf{x})$, for $m = 1, \ldots, M$
2. make predictions with

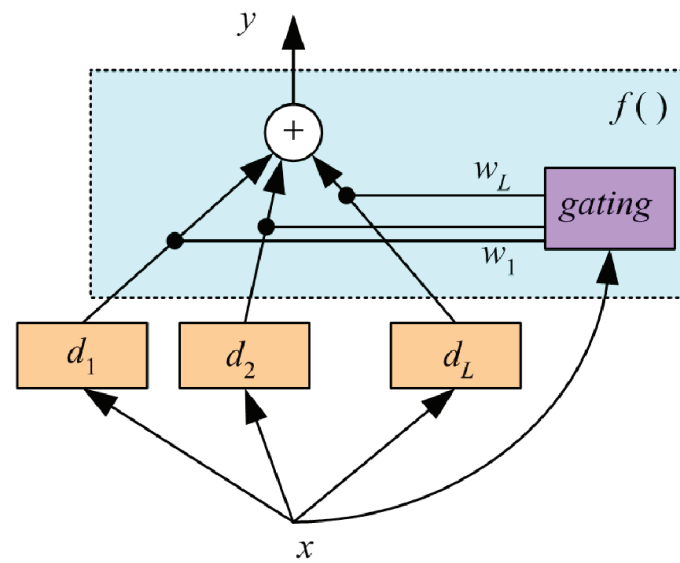$$y_{voting}(\mathbf{x}) = \sum_{m=1}^{M} w_m y_m(\mathbf{x}) \qquad \text{(regression)}$$

$$y_{voting}(\mathbf{x}) = \operatorname*{argmax}_{c} \sum_{m=1}^{M} w_m \delta(y_m(\mathbf{x}) = c) \qquad \text{(classification)}$$

with $w_m \geq 0$ and $\sum_m w_m = 1$ (prior probability of each model)
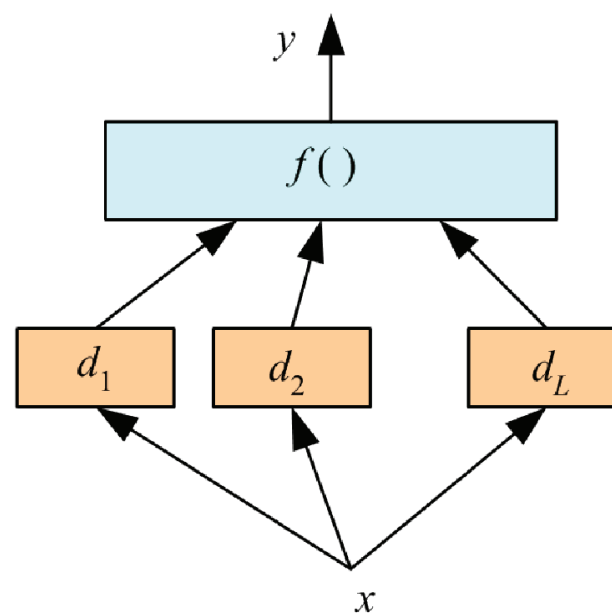
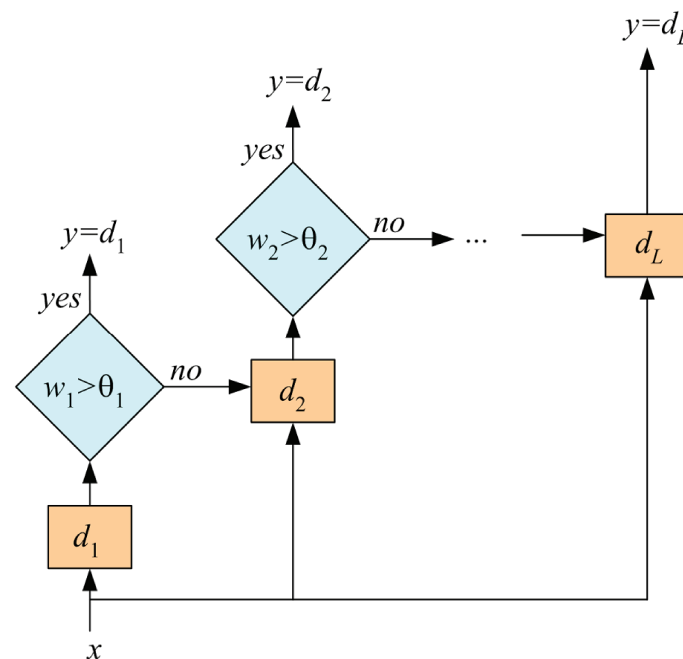# Voting

Mixture of experts

Non linear gating function depending on input

Stacking

Combination function is also learned

Cascading

Cascade learners based on cinfidence thresholds

Bagging

Given a dataset D for a regression problem,

1. generate M bootstrap data sets

2. use each bootstrap data set to train a model $y_m(\mathbf{x})$, for $m = 1, \ldots, M$

3. make predictions with

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

In general, this is better than training any individual model.

Bootstrap data sets chosen with *random sampling with replacement*

Boosting: general approach

Main points:

- Base classifiers (*weak learners*) trained sequentially
- Each classifier trained on weighted data
- Weights depend on performance of previous classifiers
- Points misclassified by previous classifiers are given greater weight
- Predictions based on weighted majority of votes

Boosting: general approach

Base classifiers are trained in sequence using a weighted data set where weights are based on performance of previous classifiers.

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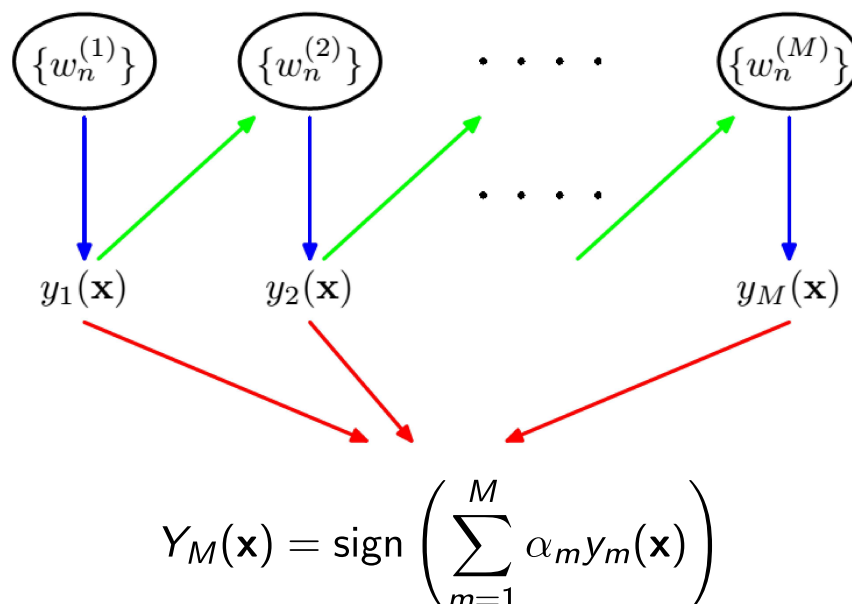
AdaBoost

Given $D = \{(\mathbf{x}_1, t_1), \ldots, (\mathbf{x}_N, t_N)\}$, where $\mathbf{x}_n \in \mathcal{X}$, $t_n \in \{-1, +1\}$

1. Initialize $w_n^{(1)} = 1/N$, $n = 1, \ldots, N$.
2. For $m = 1, \ldots, M$:

   - Train a weak learner $y_m(\mathbf{x})$ by minimizing the weighted error function:

   $$J_m = \sum_{n=1}^{N} w_n^{(m)} I(y_m(\mathbf{x}_n) \neq t_n), \text{ with } I(e) = \begin{cases} 1 & \text{if } e \text{ is true} \\ 0 & \text{otherwise} \end{cases}$$

   - Evaluate: $\epsilon_m = \dfrac{\sum_{n=1}^{N} w_n^{(m)} I(y_m(\mathbf{x}_n) \neq t_n)}{\sum_{n=1}^{N} w_n^{(m)}}$ and $\alpha_m = \ln\left[\dfrac{1 - \epsilon_m}{\epsilon_m}\right]$
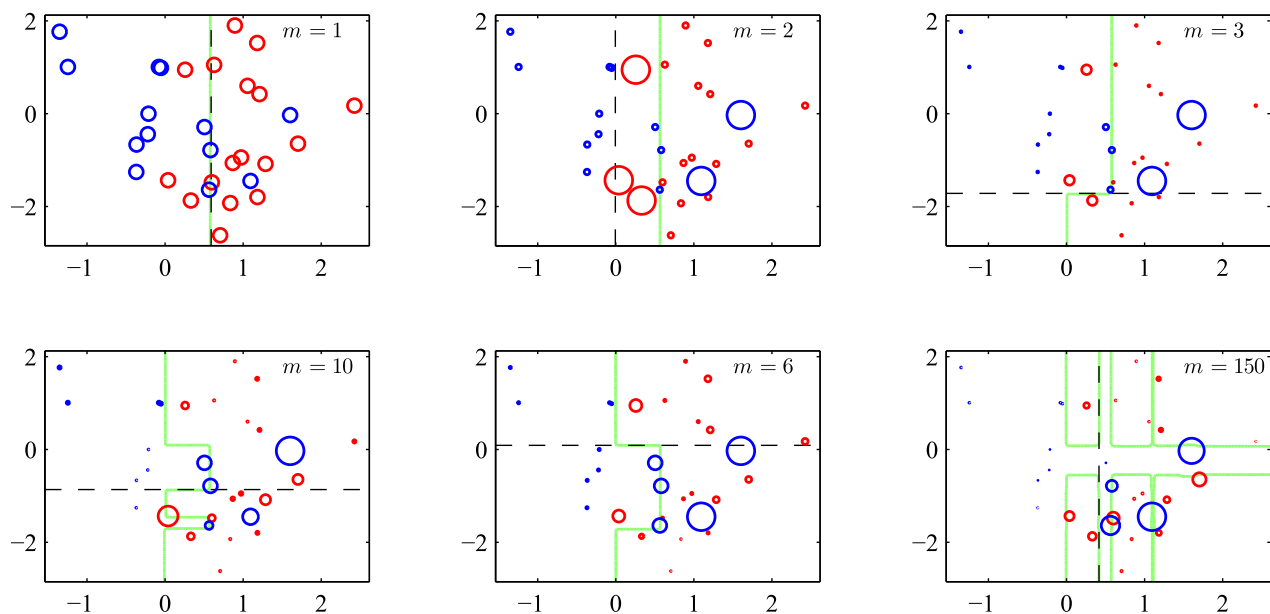   - Update the data weighting coefficients:

   $$w_n^{(m+1)} = w_n^{(m)} \exp[\alpha_m I(y_m(\mathbf{x}_n) \neq t_n)]$$

AdaBoost

3. Output the final classifier

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

AdaBoost

Exponential error minimization

AdaBoost can be explained as the sequential minimization of an exponential error function.

Consider the error function

$$E = \sum_{n=1}^{N} \exp[-t_n f_M(\mathbf{x}_n)],$$

where

$$f_M(\mathbf{x}) = \frac{1}{2} \sum_{m=1}^{M} \alpha_m y_m(\mathbf{x}), \quad t_n \in \{-1, +1\}$$

Goal:

$$\text{minimize } E \text{ w.r.t. } \alpha_m, y_m(\mathbf{x}), \ m = 1, \ldots, M$$

Exponential error minimization

Sequential minimization. Instead of minimizing E globally

- assume $y_1(\mathbf{x}), \ldots, y_{M-1}(\mathbf{x})$ and $\alpha_1, \ldots, \alpha_{M-1}$ fixed;
- minimize w.r.t. $y_M(\mathbf{x})$ and α_M.

Making $y_M(\mathbf{x})$ and α_M explicit we have:

$$E = \sum_{n=1}^{N} \exp\left[-t_n f_{M-1}(\mathbf{x}_n) - \frac{1}{2} t_n \alpha_M y_M(\mathbf{x}_n)\right]$$

$$= \sum_{n=1}^{N} w_n^{(M)} \exp\left[-\frac{1}{2} t_n \alpha_M y_M(\mathbf{x}_n)\right],$$

with $w_n^{(M)} = \exp[-t_n f_{M-1}(\mathbf{x}_n)]$ constant as we are optimizing w.r.t. α_M and $y_M(\mathbf{x})$.

Exponential error minimization

From sequential minimization of E, we obtain

$$w_n^{(m+1)} = w_n^{(m)} \exp[\alpha_m I(y_m(\mathbf{x}_n) \neq t_n)]$$

predictions are made with

$$\text{sign}(f_M(\mathbf{x})) = \text{sign}\left(\frac{1}{2} \sum_{m=1}^{M} \alpha_m y_m(\mathbf{x})\right)$$

which is equivalent to

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

thus proving that AdaBoost minimizes such error function.

AdaBoost Remarks

Advantages:

- fast, simple and easy to program
- no prior knowledge about base learner is required
- no parameters to tune (except for M)
- can be combined with any method for finding base learners
- theoretical guarantees given sufficient data and base learners with moderate accuracy

Issues:

- Performance depends on data and the base learners
  (can fail with insufficient data or when base learners are too weak)
- Sensitive to noise

Summary

- Instead of designing a learning algorithm that is accurate over the entire space, one can focus on finding base learning algorithms that only need to be better than random.
- Combined learners theoretically outperforms any individual learner.
- AdaBoost practically outperforms many other base learners in many problems.