Sapienza University of Rome

Master in Artificial Intelligence and Robotics
Master in Engineering in Computer Science

Machine Learning

A.Y. 2020/2021

Prof. L. Iocchi, F. Patrizi

Sapienza University of Rome, Italy - Machine Learning (2020/2021)

13. Multiple learners

L. Iocchi, F. Patrizi

with contributions from Valsamis Ntouskos

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 / Ensemble learning

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 set of models $y_m(\mathbf{x})$, for $m = 1, \ldots, M$
2. make predictions with

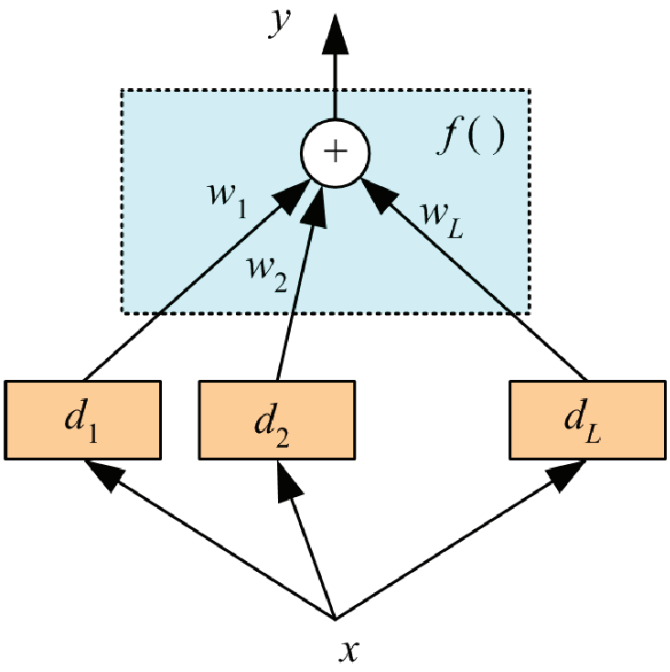
$$y_{voting}(\mathbf{x}) = \sum_{m=1}^{M} w_m y_m(\mathbf{x}) \quad \text{(regression)}$$

$$y_{voting}(\mathbf{x}) = \underset{c}{\operatorname{argmax}} \sum_{m=1}^{M} w_m I(y_m(\mathbf{x}) = c) \quad \text{weighted majority (classification)}$$

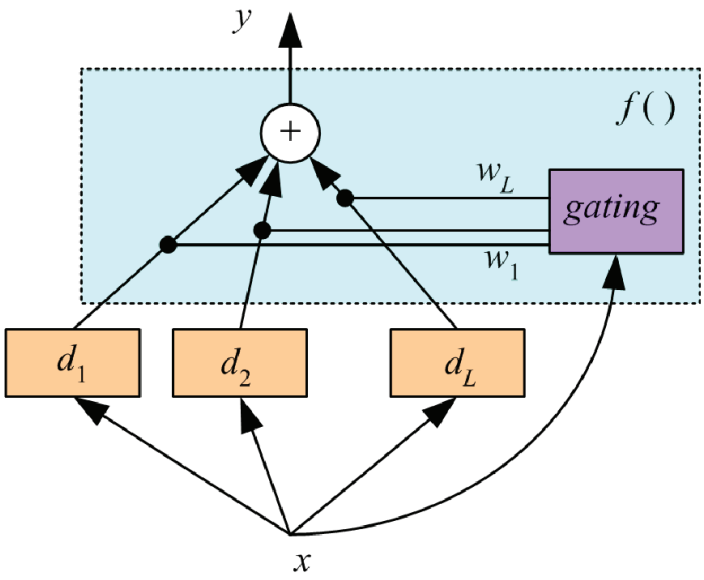
with $w_m \geq 0$, $\sum_m w_m = 1$ (prior probability of each model), $I(e) = 1$ if e is true, 0 otherwise.

Sapienza University of Rome, Italy - Machine Learning (2020/2021)

Voting

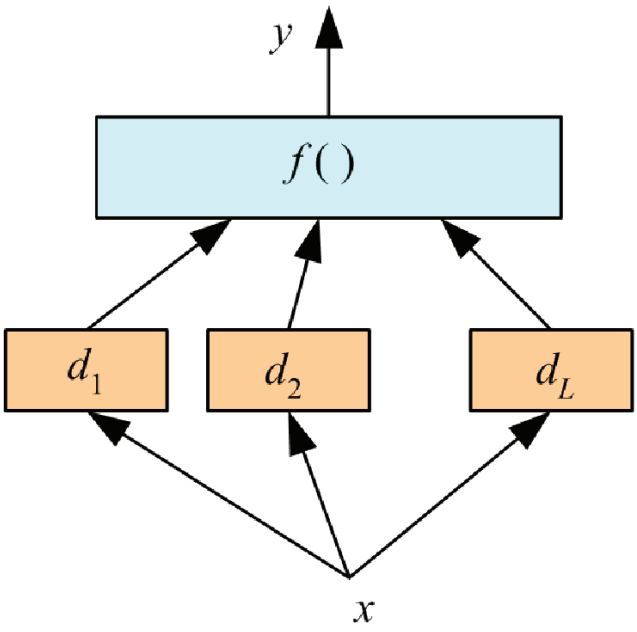


Mixture of experts



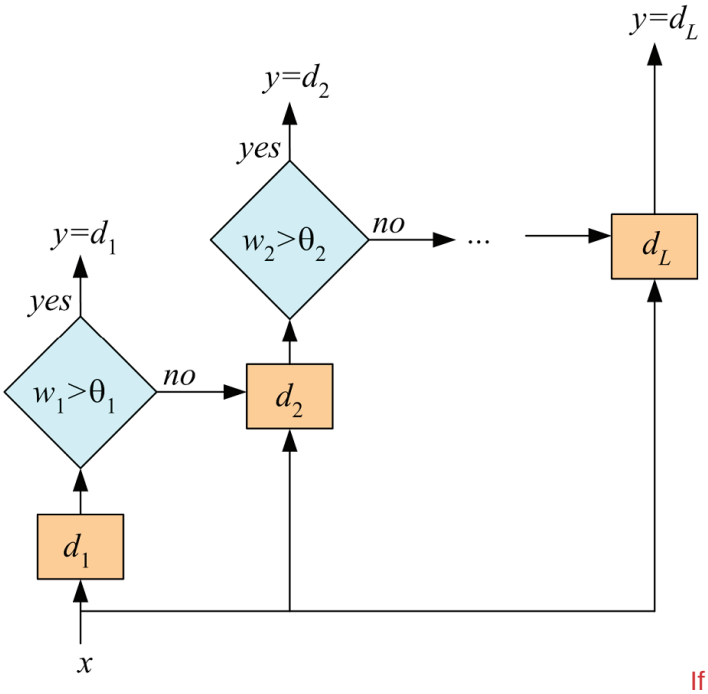
Non linear gating function f depending on input

Stacking



Combination function f is also learned

Cascading



If is minor switch to another

Cascade learners based on confidence thresholds

Bagging

Given a dataset D,

1. generate M bootstrap data sets $D_1, \ldots, D_M$, with $D_i \subset D$
2. use each bootstrap data set D_m to train a model $y_m(\mathbf{x})$, for $m = 1, \ldots, M$
3. make predictions with a voting scheme

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

Solve the problem of data availability

Only one model or slightly different models

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

Bootstrap data sets chosen with *random sampling with replacement*

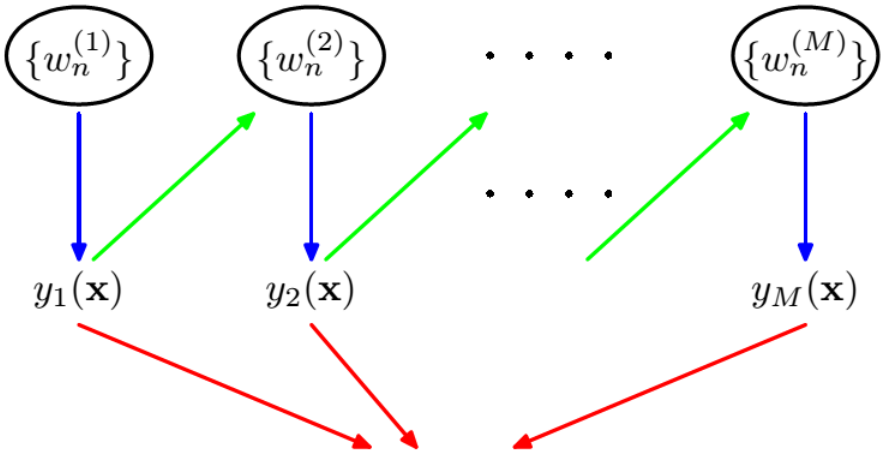
Boosting: general approach

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 $\mathbf{x}_n \in \mathbf{X}$, $t_n \in \{-1, +1\}$

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

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

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

By Some way you update weights by giving more importance to the instances that have been misclassification so far, by showing that these approach actually correspond to sequentially minimizing an esoosnential error.

- 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:

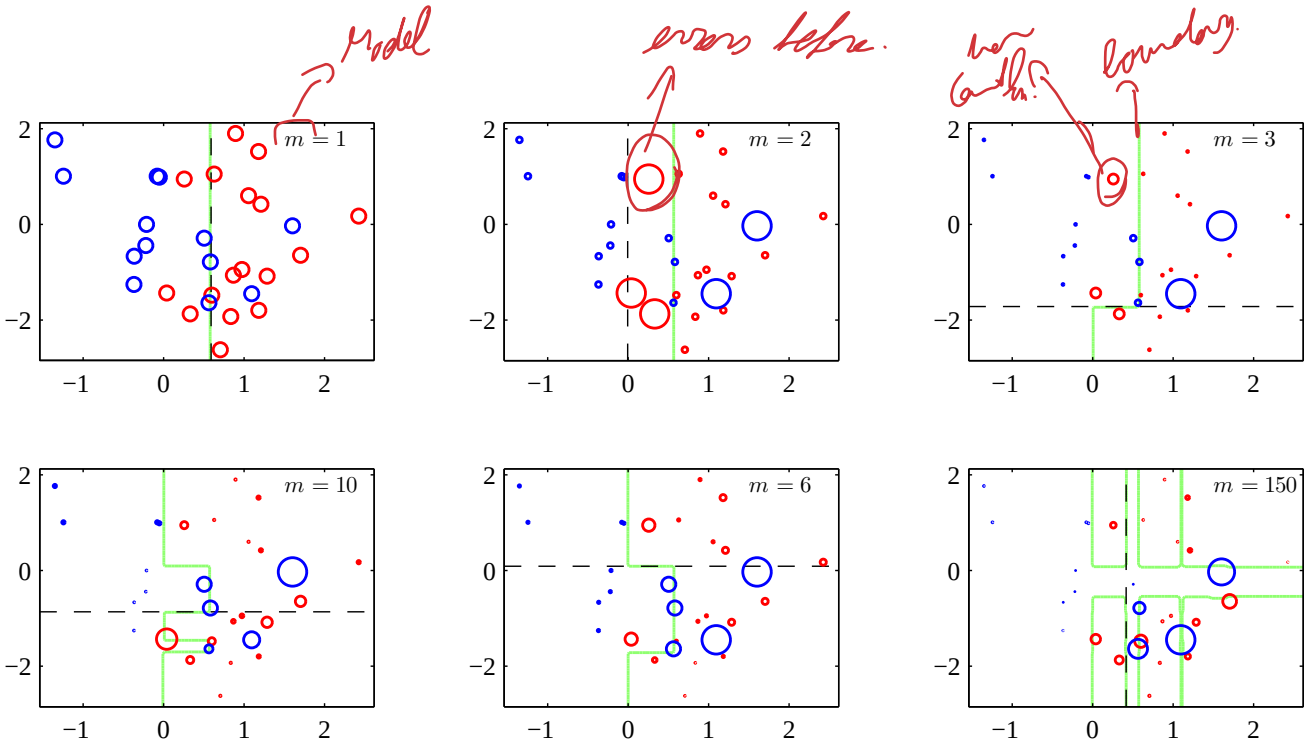
$$w_n^{(m+1)} = w_n^{(m)} \exp[\alpha_m I(y_m(\mathbf{x}_n) \neq t_n)]$$

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

$$w_n^{(m+1)} = w_n^{(m)} \exp[\alpha_m I(y_m(\mathbf{x}_n) \neq t_n)] \text{ and } \alpha_m = \ln\left[\frac{1-\epsilon_m}{\epsilon_m}\right]$$

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
- Ensembles of small DNN outperform very deep NN in some cases