Sapienza University of Rome

Master in Artificial Intelligence and Robotics

Machine Learning

A.Y. 2024/2025

Prof. Luca Iocchi

13. Multiple learners

Luca Iocchi

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

two options ↗ training in parallel → indipendent each other
↘ training in sequential

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

we use FULL DATASET

Given a dataset D

1. use D to train a set of models $y_m(\mathrm{x})$, for $m = 1, \ldots, M$

IF WE DON'T WANT USE WEIGHTS WE CAN USE $1/M$

2. make predictions with

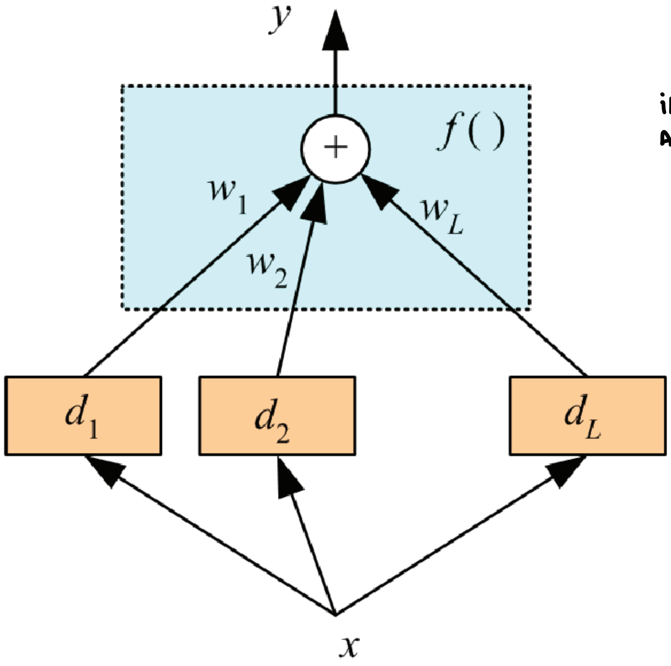
must normalize to 1

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

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

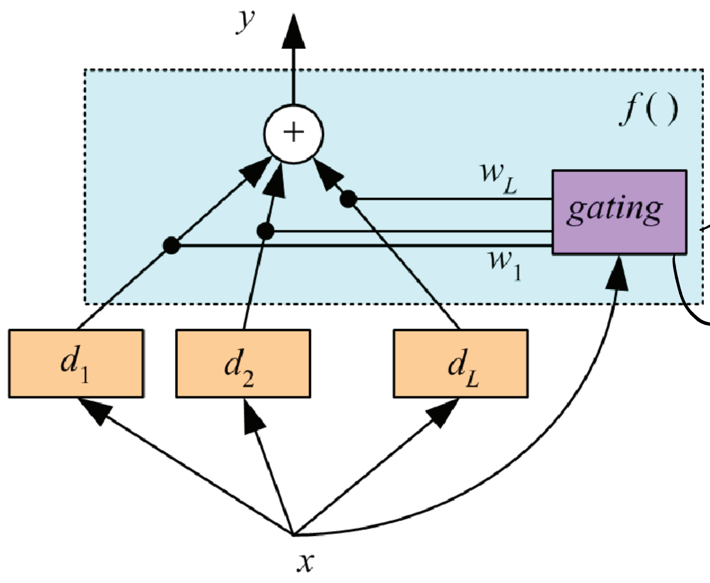
with $w_m \geq 0$, $\sum_m w_m = 1$ (prior probability of each model),
$I(e) = 1$ if e is true, 0 otherwise.

Voting



IN THIS APPROACH WEIGHTS ARE COSTANT

Mixture of experts

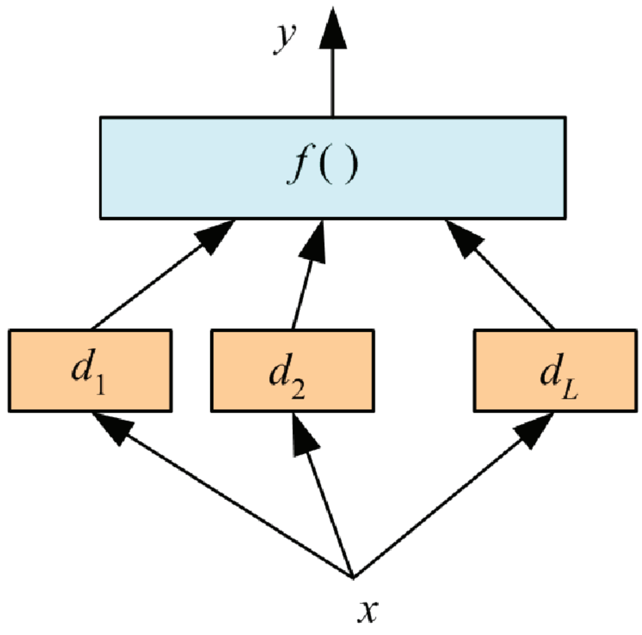


generates weights
depends on x

we can see as a REGRESSION
FUNCTION

Non linear gating function f depending on input

Stacking



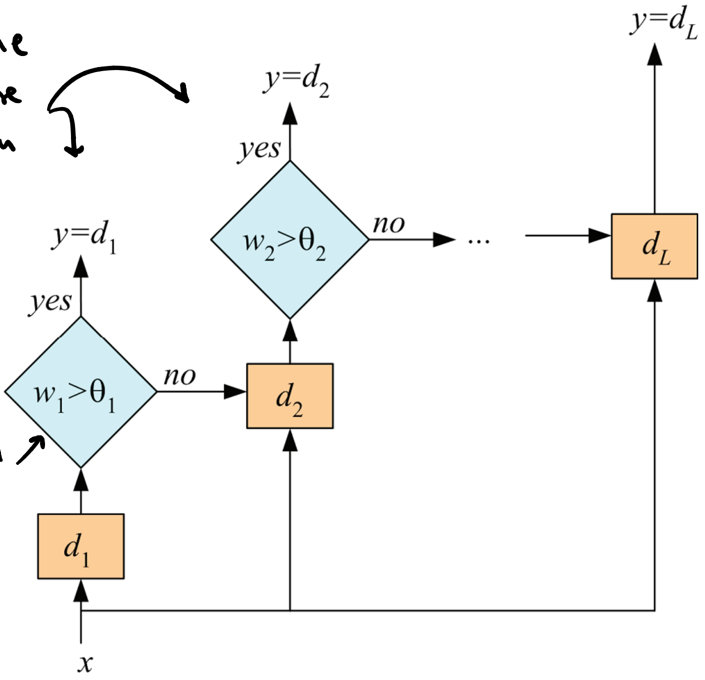
Combination function f is also learned

Cascading

we combine the prediction of some model and we use only one prediction

" ovvero decidiamo noi il criterio "

we decide the decision



Cascade learners based on confidence thresholds

one problem could be overfitting

Bagging

PARALLEL METHOD

PARTITION → $D_1 \cap D_2 = \emptyset$

L'INTERSEZIONE È VUOTA

Given a dataset D,

!! IS NOT PARTITION !!

randomly sample generated by uniform distribution

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 $y_m(\mathrm{x})$, for $m = 1, \ldots, M$
3. make predictions with a voting scheme

THIS REDUCE OVERFITTING BECAUSE doesn't show all dataset to the model

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

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

Bootstrap data sets chosen with *random sampling with replacement*

WE TRAIN MODEL SEQUENTIALY ←

Boosting: general approach

⇒ in general is better respect the other

↳ the second model try to improve performance in region of input of previous model ⇒ same samples are more important than other

↳ "ovvero il successivo cerca di migliorare il modello precedente con i pesi del modello precedente"

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

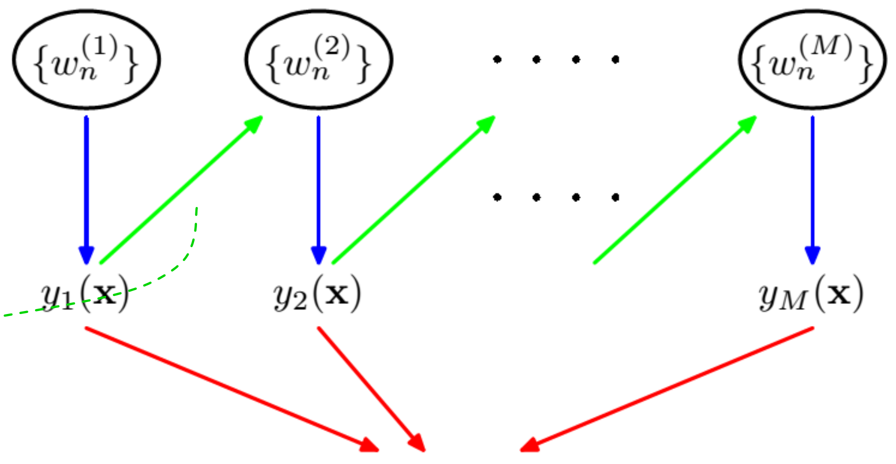
Boosting: general approach

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

this model can go overfitting only in some regions

very probable doesn't go overfitting ←

information of performance



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

AdaBoost

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

1. Initialize $w_n^{(1)} = 1/N$, $n = 1, ..., N$.
2. For $m = 1, ..., M$:
   - Train a weak learner $y_m(\mathrm{x})$ by minimizing the weighted error function:

STANDARD ERROR-FUNCTION → FOR BINARY CLASSIFICATION

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

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

USE FOR PREDICTION OF MODEL

   - Update the data weighting coefficients:

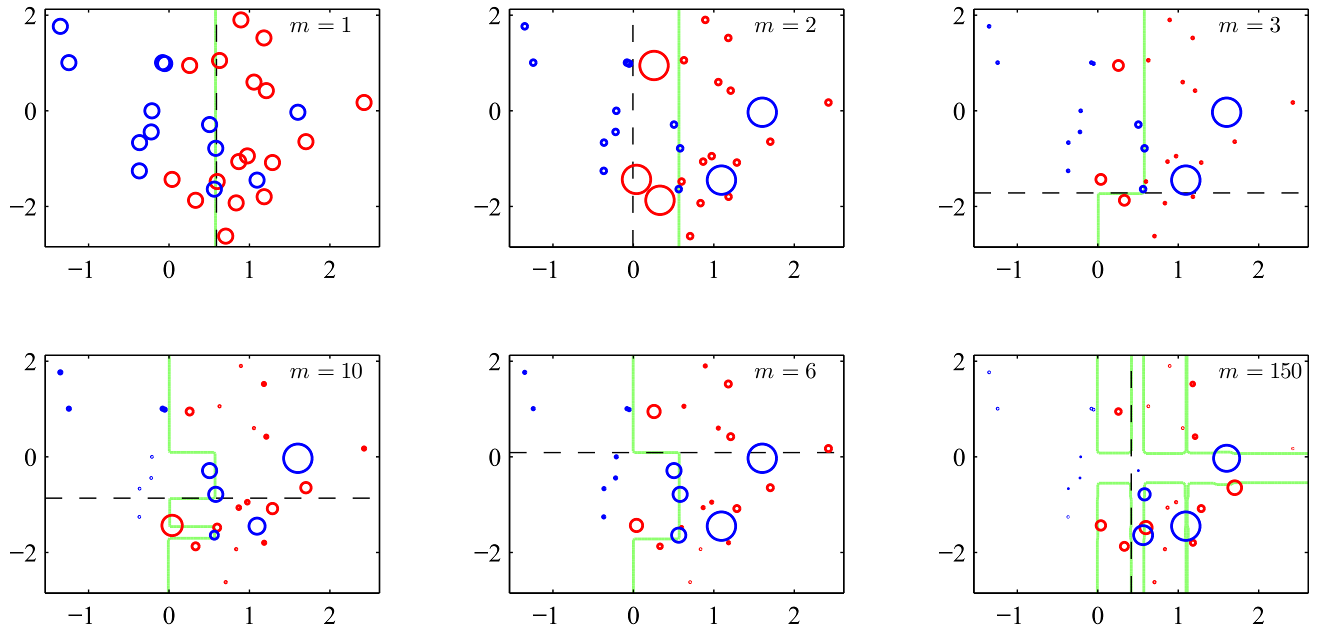
$$w_n^{(m+1)} = w_n^{(m)} \exp[\alpha_m I(y_m(\mathrm{x}_n) \neq t_n)]$$

AdaBoost

3. Output the final classifier

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

AdaBoost



Exponential error minimization

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

Consider the error function

ADABOOST MINIMIZE THIS ERROR FUNCTION

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

where

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

Goal:

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

Exponential error minimization

Sequential minimization. Instead of minimizing E globally

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

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

$$\begin{aligned} E &= \sum_{n=1}^{N} \exp\left[-t_n f_{M-1}(\mathrm{x}_n) - \frac{1}{2} t_n \alpha_M y_M(\mathrm{x}_n)\right] \\ &= \sum_{n=1}^{N} w_n^{(M)} \exp\left[-\frac{1}{2} t_n \alpha_M y_M(\mathrm{x}_n)\right], \end{aligned}$$

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

Exponential error minimization

From sequential minimization of E, we obtain

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

predictions are made with

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

which is equivalent to

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

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

RANDOM FOREST $\rightarrow$ are simple a set of DECISION TREES

- 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