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

the main idea of multiple leraners is to split the computation in order to use the computational budget to train different models instead of one

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

3 classes

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.

The simplest way to construct a committee is to average the predictions of a set of individual models.

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

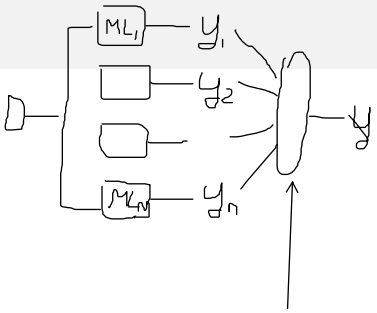
$$y_{voting}(\mathbf{x}) = \sum_{m=1}^{M} w_m y_m(\mathbf{x}) \qquad \text{(regression)}$$

weight but different from boosting

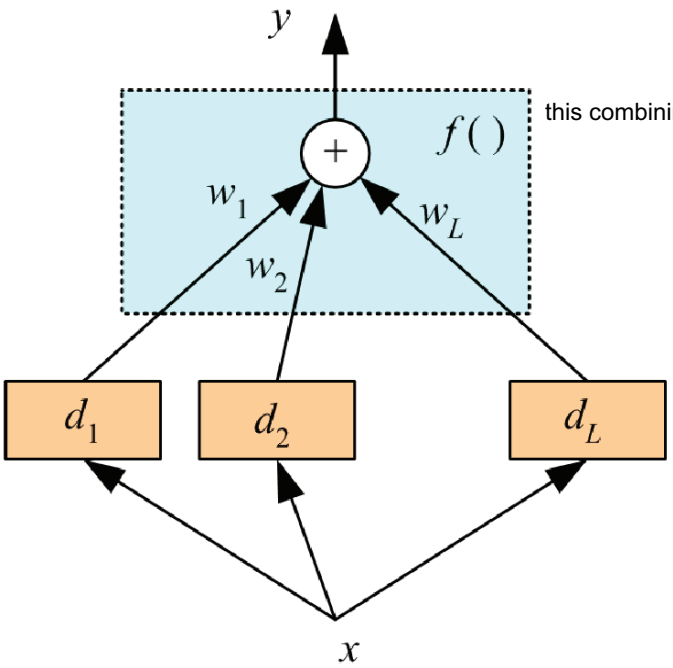
$$y_{voting}(\mathbf{x}) = \underset{c}{\operatorname{argmax}} \sum_{m=1}^{M} w_m I(y_m(\mathbf{x}) = c) \qquad \text{weighted majority (classification)}$$

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

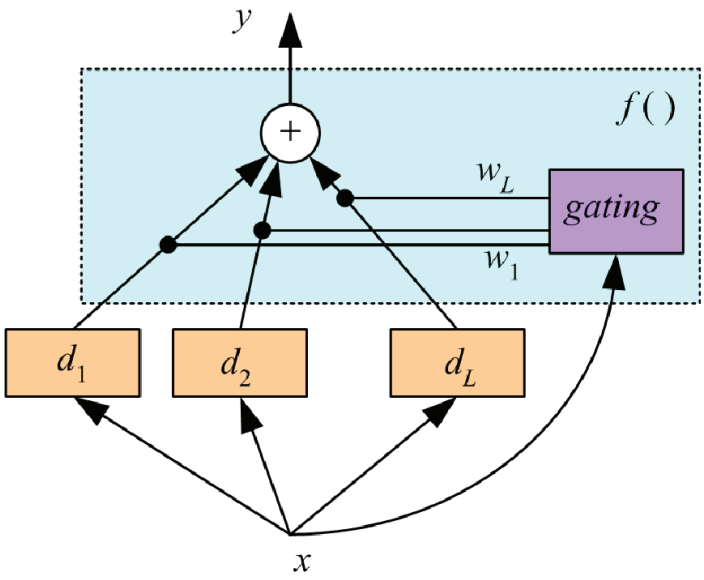
Voting



this combining leyer can be fixed or trainable

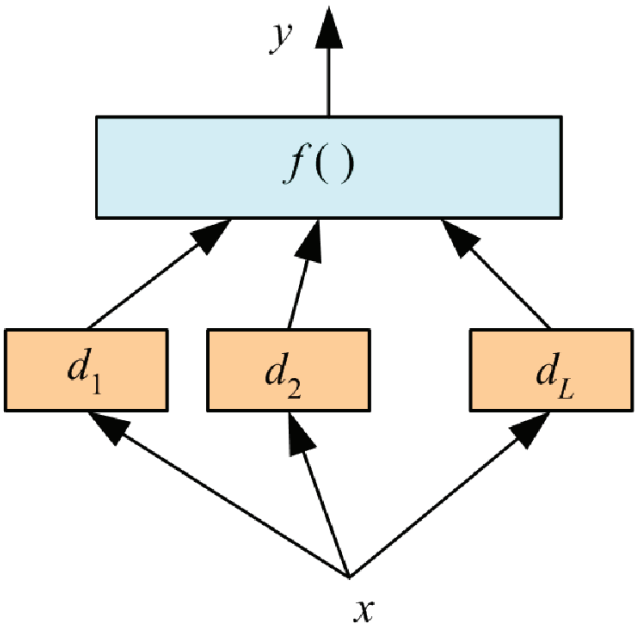


Mixture of experts



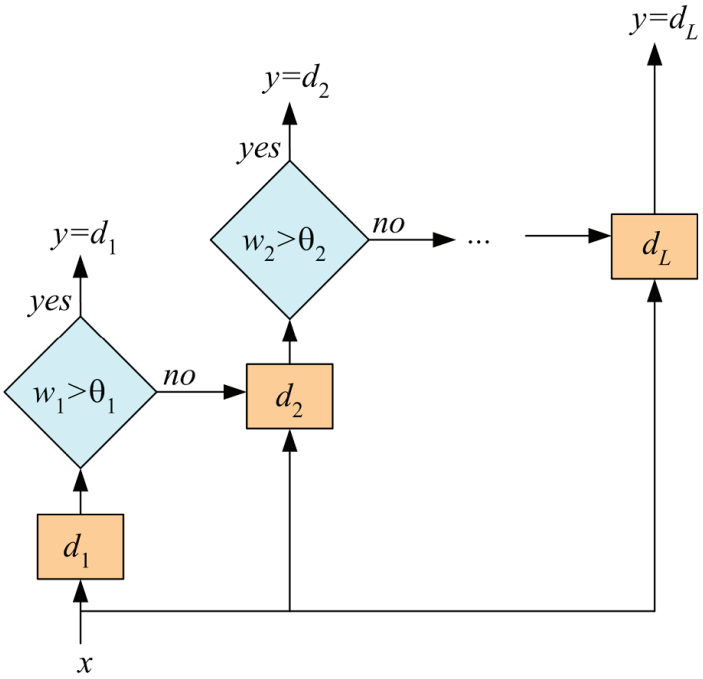
Non linear gating function f depending on input

Stacking



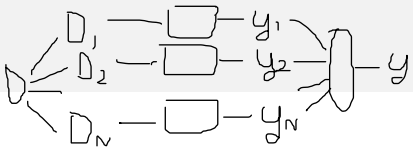
Combination function f is also learned

Cascading



Cascade learners based on confidence thresholds

we set this



Bagging

very simlar to voting but now we use a preprocessing in which the original dataset is sampled in different subsets (in these we can also have repetition, it can be also not a partition) and then these are used to train the models

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

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

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

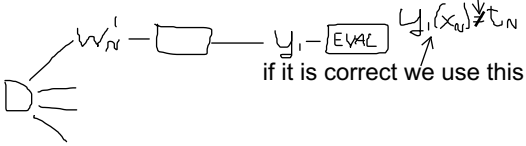
Bootstrap data sets chosen with *random sampling with replacement*

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

Boosting: general approach

in this approach we have a sequential Learning. Similar to bugging but the models are trained in a sequential order and the info of the next dataset depends on the previous one. More complex but more effective

if it makes an error, the weight is w_N^2



if it is correct we use this weight

to each sample we associate a weight that is different for each layer of the process.
this weight says how much a sample is important in the dataset

$D = \{(x_N, t_N)_{N=1}^{N} \dots\}$ $\quad w_N^1 = \frac{1}{N}$

weights are uniformly distributed

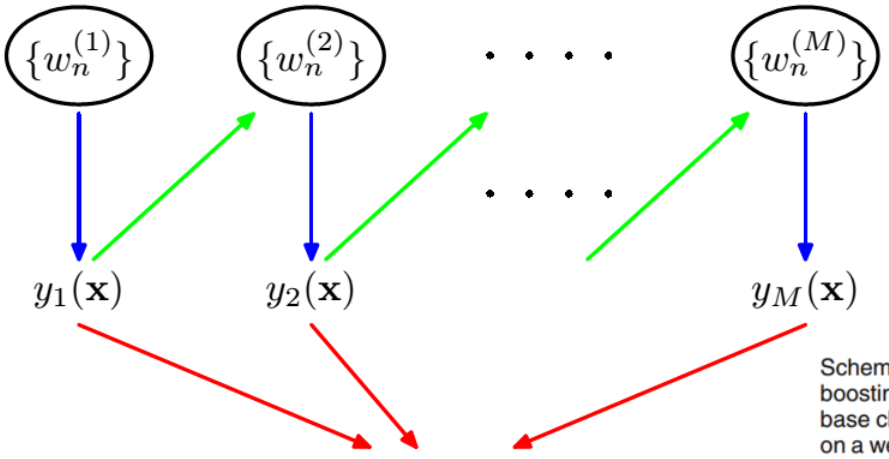
Main points:

- 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

we train the model incrementally

Boosting: general approach

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



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

Schematic illustration of the boosting framework. Each base classifier $y_m(\mathbf{x})$ is trained on a weighted form of the training set (blue arrows) in which the weights $w_n^{(m)}$ depend on the performance of the previous base classifier $y_{m-1}(\mathbf{x})$ (green arrows). Once all base classifiers have been trained, they are combined to give the final classifier $Y_M(\mathbf{x})$ (red arrows).

AdaBoost

At each stage of the algorithm, AdaBoost trains a new classifier using a data set in which the weighting coefficients are adjusted according to the performance of the previously trained classifier so as to give greater weight to the misclassified data points.

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 $m = 1, ..., M$:

Finally, when the desired number of base classifiers have been trained, they are combined to form a committee using coefficients that give different weight to different base classifiers.

- 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

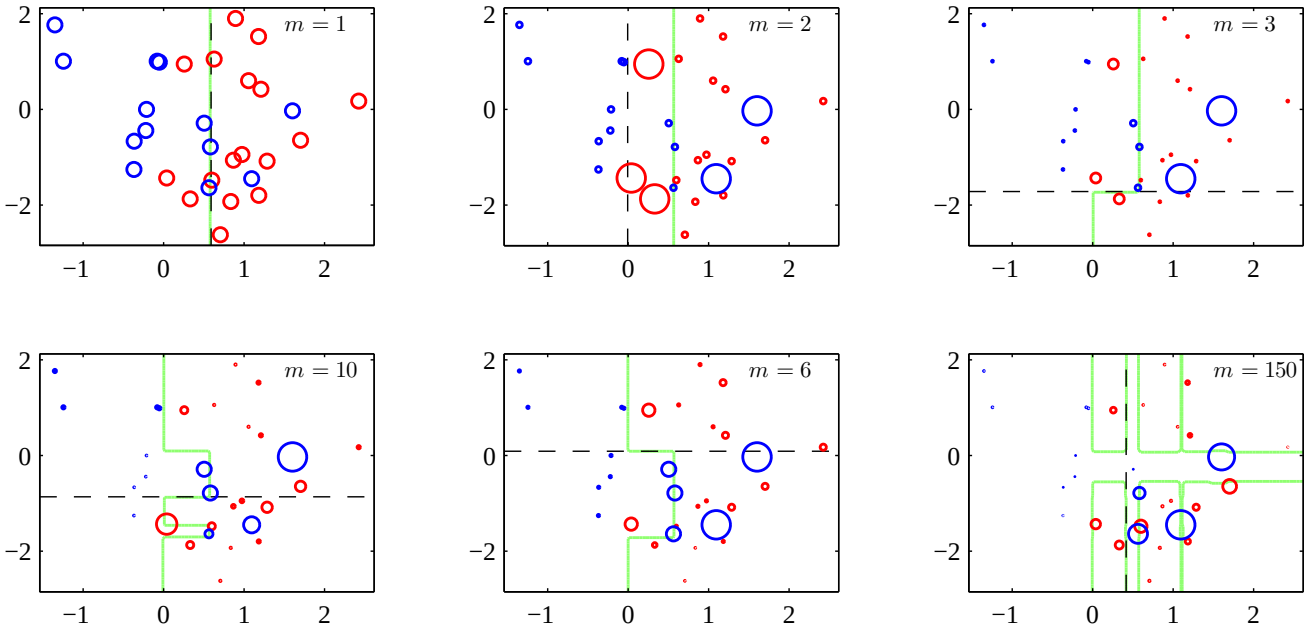
3. Output the final classifier

Make predictions using the final model, which is given by

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

AdaBoost

m is the number of base learners trained



Exponential error minimization

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

The exponential error function that is minimized by the AdaBoost algorithm differs from those considered in previous chapters.

Consider the error function

we first consider the expected error given by

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

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:

minimize E 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

What happens when we have a small computational budget and small dataset? the studies show that to train several small NN is much better than training a large one. This means that it is even faster and can be done with much less computational resourches