Ensemble learning

Course of Machine Learning
Master Degree in Computer Science
University of Rome ``Tor Vergata''

Giorgio Gambosi

a.a. 2017-2018

Idea

Improve performance by combining multiple models, in some way, instead of using a single model.

- train a *committee* of L different models and make predictions by averaging the predictions made by each model (bagging)
- use different models to perform predictions in different regions of the input space (decision trees)
- train different models in sequence: the error function used to train a model depend on the performance of previous models (boosting)

Committees

- Average the predictions of a set (committee) of m individual models, each separately trained (Bagging, bootstrap aggregation).
- Generate m training sets by bootstrap on the original dataset $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$
  - A dataset $\mathbf{X}_i$ $(i = 1, \ldots, m)$ of size n is generated by drawing n samples with replacement from $\mathbf{X}$
- The committee prediction in the case of regression is

$$y_c(\mathbf{x}) = \frac{1}{m} \sum_{i=1}^{m} y_i(\mathbf{x})$$

Bagging expected error

Expected error of one model $y_i(\mathbf{x})$ wrt the true function $h(\mathbf{x})$:

$$E_{\mathbf{x}}[(y_i(\mathbf{x}) - h(\mathbf{x}))^2] = E_{\mathbf{x}}[\varepsilon_i(\mathbf{x})^2]$$

Average expected error of the models

$$E_{av} = \frac{1}{m} \sum_{i=1}^{m} E_{\mathbf{x}}[\varepsilon_i(\mathbf{x})^2]$$

Committee expected error

$$E_c = E_{\mathbf{x}} \left[\left(\frac{1}{m} \sum_{i=1}^{m} y_i(\mathbf{x}) - h(\mathbf{x}) \right)^2 \right] = E_{\mathbf{x}} \left[\left(\frac{1}{m} \sum_{i=1}^{m} \varepsilon_i(\mathbf{x}) \right)^2 \right]$$

If $E_{\mathbf{x}}[\varepsilon_i(\mathbf{x})\varepsilon_j(\mathbf{x})] = 0$ if $i \neq j$ (errors are uncorrelated) then $E_c = \frac{1}{m} E_{av}$.

This is usually not verified: errors from different models are highly correlated.

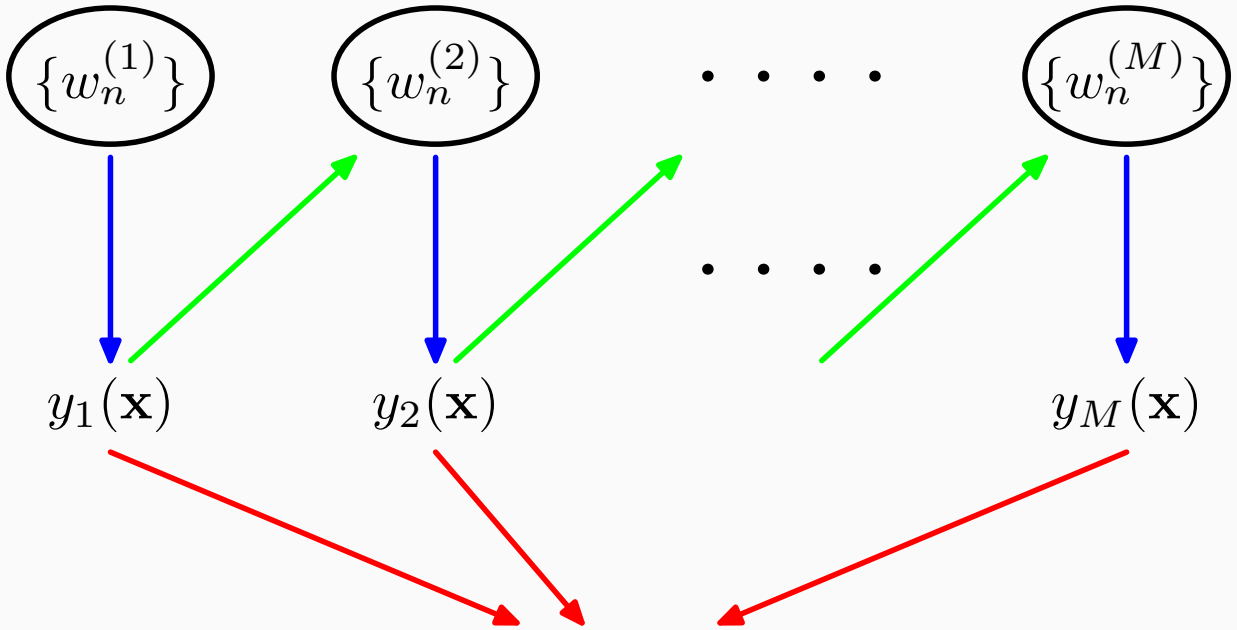
Boosting

Models (weak learners) are combined during the training phase.

Adaboost (adaptive boosting)

- Models are trained in sequence: each model is trained using a weighted form of the dataset
- Element weights depend on the performances of the previous models (misclassified points receive larger weights)
- Predictions are performed through a weighted majority voting scheme on all models

Boosting



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

Adaboost

Binary classification, dataset $(\mathbf{X}, \mathbf{t})$ of size n, with $t_i \in \{-1, 1\}$. The algorithm maintains a probability distribution $p(\mathbf{x}) = (p_1, \ldots, p_n)$ on the dataset elements.

- Set $p^{(1)}(\mathbf{x}) = (p_1^{(1)}, \ldots, p_n^{(1)})$, with $p_i^{(1)} = \frac{1}{n}$ for $i = 1, \ldots, n$
- For $j = 1, \ldots, m$:
  - Train a weak learner $y_j(\mathbf{x})$ on $\mathbf{X}$ in such a way to minimize the probability of misclassification wrt to $p^{(j)}(\mathbf{x})$.

$$q^{(j)} = \sum_{\mathbf{x}_i \in \mathcal{E}^{(j)}} p_i^{(j)} < \frac{1}{2}$$

    where $\mathcal{E}^{(j)}$ is the set of dataset elements misclassified by $y_j(\mathbf{x})$. If $\overline{p}^{(j)} > \frac{1}{2}$, consider the reverse learner, which returns opposite predictions for all elements.
  - Compute the learner confidence

$$\alpha_j = \frac{1}{2} \log \frac{1 - q^{(j)}}{q^{(j)}} > 0$$

  - For each $\mathbf{x}_i \in \mathbf{X}$ update the corresponding weight as follows

$$p_i^{(j+1)} = \begin{cases} p_i^{(j)} \frac{e^{\alpha_j}}{2\sqrt{q^{(j)}(1-q^{(j)})}} = \frac{p_i^{(j)}}{2q^{(j)}} > p_i^{(j)} & \text{if } \mathbf{x}_i \in \mathcal{E}^{(j)} \\ p_i^{(j)} \frac{e^{-\alpha_j}}{2\sqrt{q^{(j)}(1-q^{(j)})}} = \frac{p_i^{(j)}}{2(1-q^{(j)})} < p_i^{(j)} & \text{otherwise} \end{cases}$$

Adaboost

The overall prediction is

$$y(\mathbf{x}) = \text{sgn}\left(\sum_{j=1}^{m} \alpha_j y_j(\mathbf{x})\right)$$

since $y_j(\mathbf{x}) \in \{-1, 1\}$, this corresponds to a voting procedure, where each learner vote (class prediction) is weighted by the learner confidence.

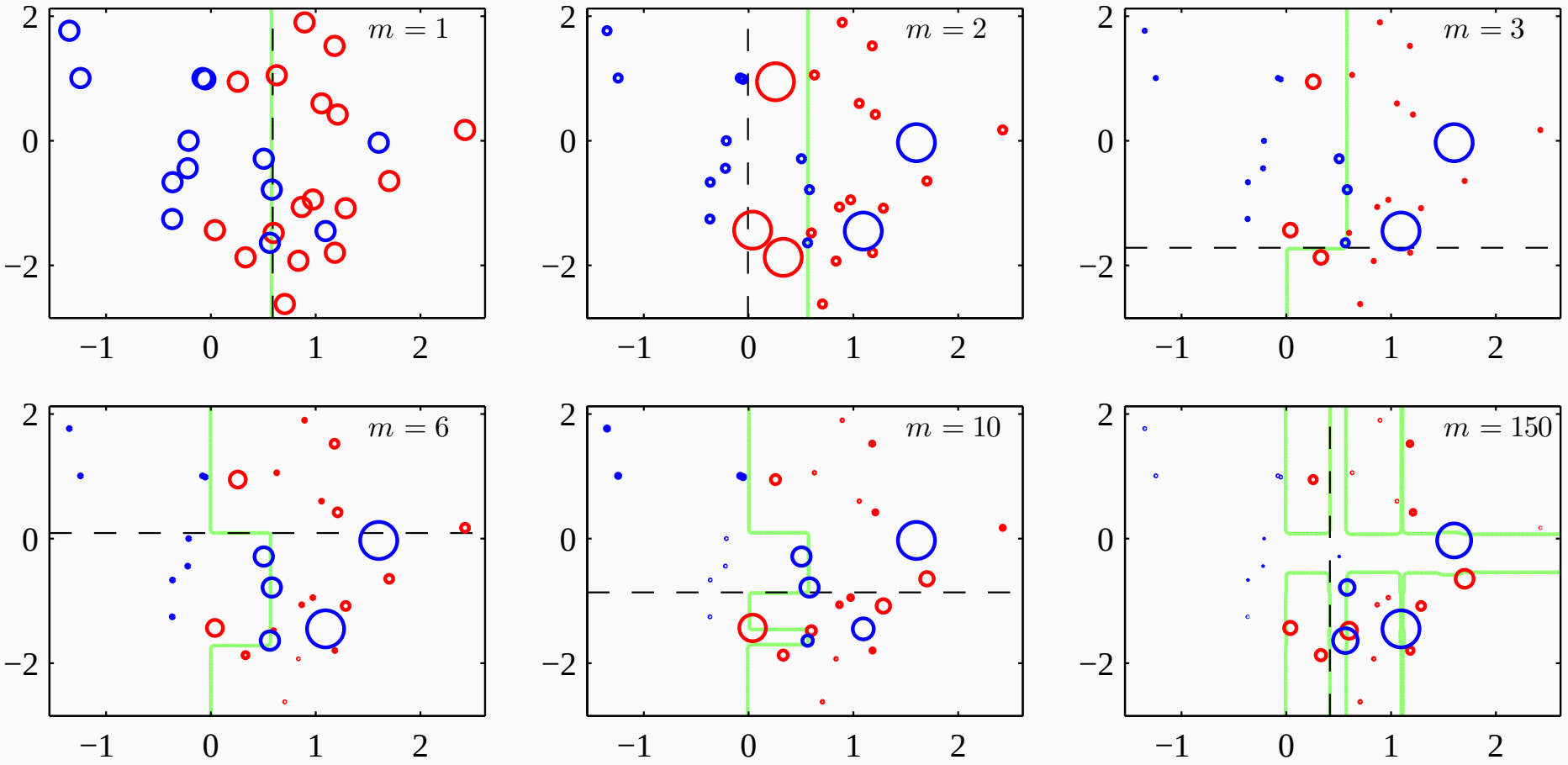
Observe that a weak learner confidence is inversely related to the probability of misclassification. Moreover,

$$p_i^{(t)} = \frac{1}{n} \frac{1}{\prod_{j \in \mathcal{B}_i} 2q^{(j)} \prod_{j \in \mathcal{G}_i} 2(1 - q^{(j)})}$$

where $\mathcal{B}_i$ is the set of indices of ``bad'' weak learners wrt $\mathbf{x}_i$ (that is ones that misclassify $\mathbf{x}_i$), while $\mathcal{G}_i$ is the set of indices of ``good'' weak learners wrt $\mathbf{x}_i$ (that is ones that correctly classify it).

Since $2q^{(j)} < 1$ (and $2(1 - q^{(j)}) > 1$) it derives that bad learners increase the probability of an element, while good learners decrease it.

Adaboost



Random forest

Application of bagging to a set of (random) decision trees: classification performed by voting.

Decision tree

Partitioning of the input space into hyper-rectangular regions, with edges aligned to the axes.

To each region a different model is assigned.

In decision trees the partitioning is recursive, and the models are simple binary threshold models on a chosen feature.

Root Entry point: associated to the whole space of data

Node Associated to a region (hyper-rectangle) and a model (feature x+threshold θ). Subregions (children nodes) are defined in correspondence to the case $x > \theta$, $x \leq \theta$, respectively.

Leaf Associated to a region (hyper-rectangle) and a class.

Decision tree: classification

Given an item $\mathbf{z} = (z_1, \ldots, z_d)^T$, the decision tree is traversed starting from its root.

At each node traversed, with associated feature x_i and threshold θ_i, a comparison between z_i and θ_i is performed to decided which is the next node to be considered, among the two children nodes. Equivalent to considering smaller and smaller subregions of the space of data.

The procedure halts when a leaf node is reached. The corresponding class is the returned prediction.

Decision tree: construction

The space of data is recursively partitioned by constructing the decision tree from root to leaves.

At each node:

1. How to perform a partition of the corresponding region (choosing feature and threshold)?
2. When stop partitioning? How to assign classes to leaves?

Decision tree: partitioning at each node

Select the feature and threshold such that a given measure is maximized within the intersections of the training set with each subregion.

Measures of class impurity within a set. To be minimized.

- Cross-entropy $-\sum_{i=1}^{K} p_i \log_2 p_i$
- Gini index $\sum_{i=1}^{K} 2p_i(1-p_i)$

where p_i is the fraction of nodes in the region belong to class i

Decision tree: leaves

Often, conditions for deciding when partitioning has to stopped are predefined (maximum tree depth, maximum number of leaves, number of items in a subregion).

When a leaf is reached, the corresponding class can be defined as the majority class in the intersection of the subregion and the training set.

A pruning procedure can be also applied to a large tree: subtrees are merged into single nodes, thus reducing the tree size.

Random forest

Application of ensemble methods (bagging) to improve accuracy by reducing model variance

- Collection of decision trees
- Voting rule to combine their predictions (majority)

Decision trees are grown on random samples of the training set (bagging)

At each node, the best feature+threshold is selected within a random sample of features (usually of size $\sqrt{d}$, to increment tree diversity.