

Ensemble Methods

TP 3 - Boosting: AdaBoost and gradient boosting

Master MALIA

Guillaume Metzler

Institut de Communication (ICOM)
Université de Lyon, Université Lumière Lyon 2
Laboratoire ERIC UR 3083, Lyon, France

guillaume.metzler@univ-lyon2.fr

Abstract

Bagging (TP 2) averages strong, unstable learners trained independently; boosting builds a *weighted* vote of weak learners trained *sequentially*, each one focusing on the errors of the previous ones. In this session we implement AdaBoost with decision stumps and gradient boosting with regression trees, and we start to *prove* things: short derivations explain why AdaBoost's weights have their form and why its training error decreases exponentially fast. We end with a comparison of all the methods seen so far on real data.

About this session – TP 3: Boosting

Goal. By the end of the session you should be able to implement AdaBoost and gradient boosting, explain the role of the weights of the examples and of the voters, prove the exponential decrease of the training error of AdaBoost, tune the learning rate and the number of rounds of gradient boosting, and compare ensemble methods with a proper protocol.

To reread before the session. Section 10.2 (boosting, Algorithm AdaBoost and the empirical error bound of AdaBoost) and Section 10.3 (gradient boosting) of the lecture notes.

Allowed tools. `numpy` for the stumps and the two boosting algorithms (regression trees of `scikit-learn` may be used as base learners in Part 4); `scikit-learn` to check and to compare.

Data. `wdbc`, `pima`, `australian` and `spambase` (4597 e-mails, 57 features) loaded with `load(name)` of `common/tptools.py`, labels in $\{-1, +1\}$, and the `diabetes` regression dataset of `scikit-learn`. The file `tp3_starter.py` contains the skeletons.

Good practice Boosting runs hundreds of rounds, and each round learns a base classifier: anything that does not change from one round to the next (here, the sort of each feature) must be computed once, before the loop. Store the scores $f_t(x) = \sum_{s \leq t} \alpha_s h_s(x)$ of every round to plot the curves without recomputing

anything.

1 Decision stumps on weighted data

A decision stump is a tree of depth one: $h_{j,t,s}(x) = s \cdot \text{sign}(x_j - t)$, with a feature j , a threshold t and a sign $s \in \{-1, +1\}$. Given nonnegative weights $w_1, \dots, w_m$ summing to 1, its weighted error is $\varepsilon_w(h) = \sum_{i=1}^m w_i \mathbb{1}[h(x_i) \neq y_i]$.

1. Let the examples be sorted by x_j , and let $L_k = \sum_{i \leq k} w_i y_i$ (sum over the k first sorted examples) and $W = \sum_i w_i y_i$.
Show that the stump with sign $s = +1$ cutting after the k -th example has weighted error $\frac{1}{2} - \frac{1}{2}(W - 2L_k)$, and that the stump with $s = -1$ has error 1 minus this quantity.
2. Write a class `Stumps(X)` which sorts every feature once in its constructor, and a method `fit(y, w)` which returns the best stump with no loop over the examples (use `np.cumsum`).
3. Check your method against a brute-force search (loops over j , t and s) on 150 examples of `pima`, with uniform and with random weights.
How long do 100 calls take on `spambase`?

2 AdaBoost from scratch

1. Implement AdaBoost (Algorithm AdaBoost of Section 10.2) with your stumps: at round t , learn the stump h_t minimizing the weighted error ε_t , set $\alpha_t = \frac{1}{2} \ln \frac{1-\varepsilon_t}{\varepsilon_t}$, update $w_i \leftarrow w_i e^{-\alpha_t y_i h_t(x_i)} / Z_t$, and return $H_T = \text{sign}(f_T)$ with $f_T = \sum_{t \leq T} \alpha_t h_t$. Store ε_t , α_t , Z_t and the scores f_t on the training and test sets.
2. On `wdbc` and `spambase` (70%/30%), run $T = 500$ rounds and plot, against t (log scales), the training error, the test error, $\prod_{s \leq t} Z_s$ and $\exp(-2 \sum_{s \leq t} (\frac{1}{2} - \varepsilon_s)^2)$.
3. How do ε_t and α_t evolve with t ?
Why?
4. Compare with `AdaBoostClassifier(DecisionTreeClassifier(max_depth=1), n_estimators=500)`.
5. What happens to the test error once the training error is 0?
Is this what you expected after TP 1 (overfitting) and TP 2 (number of trees in bagging)?

3 Why AdaBoost works: three short proofs

Let $w = w^{(t)}$ be the weights at round t , $h = h_t$, $\varepsilon = \varepsilon_t \in (0, \frac{1}{2})$ and, for $\alpha \in \mathbb{R}$, $Z(\alpha) = \sum_i w_i e^{-\alpha y_i h(x_i)}$.

1. Show that $Z(\alpha) = (1 - \varepsilon)e^{-\alpha} + \varepsilon e^{\alpha}$, that it is minimized by $\alpha_t = \frac{1}{2} \ln \frac{1-\varepsilon}{\varepsilon}$, and that $Z(\alpha_t) = 2\sqrt{\varepsilon(1-\varepsilon)}$.
2. Show that the weighted error of h_t under the *new* weights $w^{(t+1)}$ is exactly $\frac{1}{2}$.
Check it numerically.
What does this imply for h_{t+1} ?
3. Show that $w_i^{(T+1)} = \frac{e^{-y_i f_T(x_i)}}{m \prod_{t \leq T} Z_t}$, and deduce, using $\mathbb{1}[yf(x) \leq 0] \leq e^{-yf(x)}$, that the training error satisfies $\widehat{R}_S(H_T) \leq \prod_{t \leq T} Z_t$.
Then use $1 - x \leq e^{-x}$ to obtain the bound of Section 10.2, $\widehat{R}_S(H_T) \leq \exp(-2 \sum_t (\frac{1}{2} - \varepsilon_t)^2)$.

4 Gradient boosting for regression

Gradient boosting builds $F_M(x) = F_0 + \nu \sum_{k=1}^M g_k(x)$, where $F_0 = \bar{y}$ and each g_k is a small regression tree fitted to the *negative gradient* of the loss at the current predictions; $\nu \in (0, 1]$ is the learning rate (shrinkage).

1. For the squared loss $\ell(y, F) = \frac{1}{2}(y - F)^2$, show that the negative gradient with respect to $F(x_i)$ is the residual $y_i - F_{k-1}(x_i)$.
Write the algorithm.
2. Implement it with `DecisionTreeRegressor(max_depth=2)` as base learner.
On `diabetes`, split the data 60/20/20 (training, validation, test).
For $\nu \in \{1, 0.3, 0.1, 0.03\}$ and up to $M = 1000$ trees, plot the training and validation MSE against M .
3. For each ν , select M on the validation set (early stopping) and report the test MSE.
Compare with `GradientBoostingRegressor` with the same parameters (`criterion="squared_error"`).
4. What is the role of ν ?
Why is boosting, unlike bagging, able to overfit when M grows?

5 Comparing the methods of the course

1. With 5-fold stratified cross-validation, compare on `wdbc`, `pima`, `australian` and `spambase`: a logistic regression (with standardization in a pipeline), a tree of depth 5, a random forest of 200 trees, AdaBoost with 200 stumps and gradient boosting with 200 trees of depth 3 (`scikit-learn`, default values otherwise).
Report the mean and the standard deviation of the error over the folds.
2. Is there a best method?
Which differences seem significant, given the standard deviations?

6 Going further (★)

1. **AdaBoost minimizes the exponential loss.** Show that AdaBoost is a greedy, stage-wise minimization of $\frac{1}{m} \sum_i e^{-y_i f(x_i)}$ over $f = \sum_t \alpha_t h_t$: at round t , with f_{t-1} fixed, the pair (h, α) minimizing $\frac{1}{m} \sum_i e^{-y_i (f_{t-1}(x_i) + \alpha h(x_i))}$ is the stump of minimal weighted error and α_t .
2. **Margins.** The normalised margin of a training example is $y_i f_t(x_i) / \sum_{s \leq t} \alpha_s \in [-1, 1]$.
On `wdbc`, plot the cumulative distribution of the training margins for $t = 10, 50, 500$, and relate it to the evolution of the test error after the training error has reached 0.
3. **Label noise.** On `spambase`, flip 20% of the training labels (the test labels stay clean) and plot the test error of AdaBoost as a function of t up to 1000 rounds. Compare with the noiseless case and with a random forest.
Explain using question 1.