Introduction to Machine Learning
NPFL 054

http://ufal.mff.cuni.cz/course/npfl054

Barbora Hladká
hladka@ufal.mff.cuni.cz

Martin Holub
holub@ufal.mff.cuni.cz

Charles University,
Faculty of Mathematics and Physics,
Institute of Formal and Applied Linguistics

Ensemble learning methods
Part II: Boosting

Outline

- Bagging vs. boosting
- Simple boosting trees – the regression case
- Adaptive boosting – classification with AdaBoost

Bagging and boosting — the difference

- **Bagging**: each predictor is trained independently
- **Boosting**: each predictor is built on the top of previous predictors trained

  – Like bagging, boosting is also a voting method. In contrast to bagging, boosting actively tries to generate complementary learners by training the next learner on the mistakes of the previous learners.

Boosting

Boosting combines the outputs of many "weak" classifiers ("rules of thumb") to produce a powerfull "commitee."

Motivation

- How to extract rules of thumb that will be the most useful?
- How to combine moderately accurate rules of thumb into a single highly accurate prediction rule?

Basic idea

- Boosting is a method that produces a very accurate predictor by combininig rough and moderately accurate predictors.
- It is based on the observation that finding many rough predictors (rules of thumb) can be easier than finding a single, highly accurate predictor.

Simple boosting with regression trees

1. Initialization: Set $h(x) = 0$ and $r_i = y_i$ for all $i = 1, \ldots, n$ in the training set
2. For $b = 1, \ldots, B$, repeat
   - **(a)** Fit a tree h^b with only d splits to the training set (X, r)
   - **(b)** Update h by adding the new tree

     $$h(x) \longleftarrow h(x) + \lambda h^b(x)$$

   - **(c)** Update the residuals

     $$r_i \longleftarrow r_i - \lambda h^b(x_i)$$

3. Output the boosted model

$$h(x) = \sum_{b=1}^{B} \lambda h^b(x)$$

Boosting with regression trees – tuning parameters

- The number of trees B
- The shrinkage parameter λ
  – A small positive number. This controls the rate at which boosting learns. Typical values are 0.01 or 0.001, and the right choice can depend on the problem. Very small λ can require using a very large value of B in order to achieve good performance.
- The number d of splits in each tree
  – Trees with just $d = 1$ split are called "stumps".

Boosting — Adaboost (Adaptive Boosting)

AdaBoost is a boosting method that repeatedly calls a given weak learner, each time with different distribution over the training data. Then we combine these weak learners into a strong learner.

- originally proposed by Freund and Schapire (1996)
- great success
  — "AdaBoost with trees is the best off-the-shelf classifier in the world." (Breiman 1998)
  — "Boosting is one of the most powerful learning ideas introduced in the last twenty years." (Hastie et al, 2009)

Boosting — Adaboost (Adaptive Boosting)

Key questions

- How to choose the distribution?
- How to combine the weak predictors into a single predictor?
- How many weak predictors should be trained?

Schapire's strategy: Change the distribution over the examples in each iteration, feed the resulting sample into the weak learner, and then combine the resulting hypotheses into a voting ensemble, which, in the end, would have a boosted prediction accuracy.

Binary classification and AdaBoost.M1

AdaBoost.M1 (Freund and Schapire, 1997) is the most popular boosting algorithm

- Consider a binary classification task with the training data

$$Data = \{\langle \mathbf{x}_i, y_i \rangle : \mathbf{x}_i \in \mathbf{X}, y_i \in \{-1, +1\}, i = 1, \dots, n\}$$

- We need to define distribution $\mathcal{D}$ over *Data* such that $\sum_{i=1}^{n} \mathcal{D}_i = 1$.

- Assumption: a weak classifier h_t has the property

$$\mathrm{error}_{\mathcal{D}}(h_t) < 1/2.$$

Adaboost (Adaptive Boosting) — key idea

Classifiers are trained on weighted versions of the original training data set, and then combined to produce a final prediction

- $\boxed{\text{Training examples}} \longrightarrow h_1(x)$

$\downarrow$

- $\boxed{\text{Weighted examples}} \longrightarrow h_2(x)$

$\downarrow$

- $\boxed{\text{Weighted examples}} \longrightarrow h_3(x)$

$\downarrow$

$\vdots$

- $\boxed{\text{Weighted examples}} \longrightarrow h_M(x)$

Final hypothesis $h(x) = \text{sign} \sum_{t=1}^{M} \alpha_t h_t(x)$, where α_t are computed by the boosting algorithm, and weight the contribution of each respective h_t

AdaBoost – iterative algorithm

- Initialize the training distribution $\mathcal{D}_1(i) = 1/n \quad \text{for } i = 1, \ldots, n$
- At each step t
  - Learn h_t using $\mathcal{D}_t$: find the weak classifier h_t with the minimum weighted sample error $\mathrm{error}_{\mathcal{D}_t}(h_t) = \sum_{i=1}^{n} \mathcal{D}_t(i)\,\delta(h(\mathbf{x}_i) \neq y_i)$
  - Set weight α_t of h_t based on the sample error

$$\alpha_t = \frac{1}{2} \ln \left(\frac{1 - \mathrm{error}_{\mathcal{D}_t}(h_t)}{\mathrm{error}_{\mathcal{D}_t}(h_t)} \right)$$

  - Update the training distribution

$$\mathcal{D}_{t+1} = \mathcal{D}_t\, \mathrm{e}^{-\alpha_t y_i h_t(\mathbf{x}_i)} / Z_t \quad \text{where } Z_t \text{ is a normalization factor}$$

  - Stop when impossible to find a weak classifier being better than chance
- Output the final classifier $h_{final}(\mathbf{x}) = \mathrm{sign} \sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})$

AdaBoost – training data weighting

Constructing $\mathcal{D}_t$

- On each round, the weights of incorrectly classified instances are increased so that the algorithm is forced to focus on the hard training examples.
- $\mathcal{D}_1(i) = 1/n \quad \text{for } i = 1, \dots, n$
- given $\mathcal{D}_t$ and h_t (i.e. update $\mathcal{D}_t$):

$$\mathcal{D}_{t+1}(i) = \frac{\mathcal{D}_t(i)}{Z_t} \cdot \begin{cases} e^{-\alpha_t} & \text{if} \quad y_i = h_t(x_i) \\ e^{\alpha_t} & \text{if} \quad y_i \neq h_t(x_i) \end{cases} = \frac{\mathcal{D}_t(i)}{Z_t} \, \mathrm{e}^{-\alpha_t y_i h_t(x_i)},$$

  where Z_t is normalization constant $Z_t = \sum_i \mathcal{D}_t(i) \, \mathrm{e}^{-\alpha_t y_i h_t(x_i)}$

- α_t measures the importance that is assigned to h_t

As the iterations proceed, examples that are difficult to classify correctly receive ever-increasing influence

AdaBoost – base learners weighting

Weights of the base learners α_t

- $error_{\mathcal{D}_t}(h_t) < \frac{1}{2} \Rightarrow \alpha_t > 0$
- the smaller the error, the bigger the weight of the (weak) base learner
- the bigger the weight, the more impact on the (strong) resulting classifier

$$error_{\mathcal{D}_t}(h_1) < error_{\mathcal{D}_t}(h_2) \Longrightarrow \alpha_1 > \alpha_2$$

- $\mathcal{D}_{t+1} = \dfrac{1}{Z_t}\mathcal{D}_t\, \mathrm{e}^{-\alpha_t y_i h_t(\mathbf{x}_i)}$
  The weights of correctly classified instances are reduced while weights of misclassified instances are increased.

AdaBoost.M1 — multiclass problem

Multiclass problem – generalization of the two-class case

- Assume classification task where $Y = \{y_1, \dots, y_k\}$

$$h_t : X \rightarrow Y,$$

$$\mathcal{D}_{t+1}(i) = \frac{\mathcal{D}_t(i)}{Z_t} \cdot \begin{cases} e^{-\alpha_t} & \text{if} \quad y_i = h_t(\mathbf{x_i}) \\ e^{\alpha_t} & \text{if} \quad y_i \neq h_t(\mathbf{x_i}) \end{cases}$$

$$h_{final}(\mathbf{x}) = argmax_{y \in Y} \sum_{\{t \mid h_t(\mathbf{x}) = y\}} \alpha_t.$$

Summary of examination requirements

- Ensembles, bagging, boosting – general principles
- Boosting with regression trees
- AdaBoost