ASSIGNMENT 5: ENSEMBLE METHODS

Johannes Kofler

JOHANNES KEPLER UNIVERSITY LINZ

Institute for Machine Learning

Copyright statement:

Ensemble methods: basic ideas

- Instead of a single model, multiple models are trained.
- When making predictions, the results of these models are aggregated (e.g. averaged, voting, etc).
- Aim: Reduce
  - Variance
  - Bias
  - Both (if possible)
- We discuss two most commonly used ensemble methods for decision trees, namely:
  - Random forests (RFs); aim: reduce variance
  - Gradient tree boosting; aim: reduce bias
- We start with random forests, as they are conceptually easier.

Random forests (RFs): Part 1: Algorithm

★★

- Training set $\mathbf{Z}$ where each element has d features.
- Hyperparameters: N, k, m, $s_{\min}$, splitting criterion
- For $n = 1$ to N:
  - ☐ Draw bootstrap subsample $\mathbf{Z}^*$ of size k from training data.
  - ☐ Grow a RF tree T_n to bootstrapped data by recursively repeating the following steps for terminal node until minimum node size $s_{\min}$ is reached:
    1. From d features select $m \leq d$
    2. Pick best variable/split among the m features
    3. Split node into two daughter nodes
- Output ensemble of trees = random forest: $\{T_n\}_1^N$

To make prediction for new input $\mathbf{x}$:

- Regression: $\hat{f}_{\mathrm{RF}} = \frac{1}{N} \sum_{n=1}^{N} T_n(\mathbf{x})$
- Classification: Let $\hat{C}_b(\mathbf{x})$ be the class prediction of n-th RF tree. Then $\hat{C}_{\mathrm{RF}}(\mathbf{x}) =$ majority vote $\{\hat{C}_n(\mathbf{x})\}_1^N$.

Random forests (RFs): Part 3: Comments

- Essential idea: Average many noisy but approximately unbiased models to reduce variance
- Since trees are noisy, they benefit from averaging
- Each tree is identically distributed $\rightarrow$ expectation of average of B trees is same as expectation of any of them $\rightarrow$ no bias improvement
- However, the variance of the forest can be computed as $\rho\,\sigma^2 + \frac{1-\rho}{N}\,\sigma^2$ for pairwise correlation coefficient ρ and single tree variance σ^2 (see exercises).
- If N increases, second term disappears, but first remains; size of correlation limits benefits of averaging.
- Basic idea of RFs: Reduce variance by reducing correlation between trees, without increasing individual variance too much. Achieved through random selection of features.

More details about RFs: Part 2: OOB estimates

- RFs allow for assessing generalization performance on basis of training data
- Out of bag (OOB) estimates: For each observation $\mathbf{z}_i = (\mathbf{x}_i, y_i)$ construct its RF predictor by averaging only trees corresponding to bootstrap samples in which $\mathbf{z}_i$ did not appear
- Overall OOB error: averaging OOB errors of all samples.
- OOB error estimate is almost identical to that obtained by N-fold cross validation for large N.

More details about RFs: Part 3: Variable importance

- Mean Gini purity gain (impurity decrease):
    - For all features, average the Gini purity gains of all splits in all trees that involve this feature;
- Mean accuracy decrease:
    - Compute OOB error for each sample.
    - For each feature separately, consider random permutations among the data and compute the OOB errors for the data set with the permuted feature.
    - Then the importance score is computed by averaging the OOB error differences before and after permuting the feature (upon normalization by the standard deviation of the differences).

Basic ideas of boosting methods: Part 1

- Consider two-class problem with outputs $y \in \{-1, 1\}$ and a classifier $g(\mathbf{x}) \in \{-1, 1\}$
- The error rate for l samples is given by $\frac{1}{l} \sum_{i=1}^{l} I(y_i \neq g(\mathbf{x}_i))$. I denotes indicator function.
- Weak classifier: only slightly better than random guessing
- Aim of boosting: sequentially apply weak classification algorithm to repeatedly modified versions of data, producing sequence of weak classifiers $g_n(\mathbf{x})$, $n = 1, ..., N$.
- Predictions from all of them are combined through weighted majority vote to produce final prediction:
$$g(\mathbf{x}) = \mathrm{sign}(\textstyle\sum_{n=1}^{N} \alpha_n \, g_n(\mathbf{x}))$$
- $\alpha_1, ..., \alpha_N$: computed by boosting algorithm. Effect: give higher influence to more accurate classifiers.

Basic ideas of boosting methods: Part 2

- Data modifications at each boosting step n consists of applying weights $w_1^{(n)}, ..., w_l^{(n)}$ to each of training observations $(\mathbf{x}_i, y_i)$, $i = 1, ..., l$
- Start: Initialize all weights by $w_i^{(n=1)} = \frac{1}{l}$
- Successive iterations: weights individually modified, algorithm reapplied to weighted observations.
- At step n: Observations misclassified by $g_{n-1}(\mathbf{x})$ have weights increased, correctly classified samples decreased.
- As iterations proceed: observations that are difficult to classify correctly receive ever-increasing influence.

Basic ideas of boosting methods: Part 3

AdaBoost.M1 algorithm (Friedman et al. 2000):

1. Initialize observation weights $w_i^{(n=1)} = \frac{1}{l}$, $i = 1, ..., l$.
2. For $n = 1$ to N:
   - Find optimal classifier $g_n(\mathbf{x})$ to training data with weights $w_i^{(n)}$.
   - Compute:
   $$\mathrm{err}_n = \frac{\sum_{i=1}^{l} w_i^{(n)} \, I(y_i \neq g_n(\mathbf{x}_i))}{\sum_{i=1}^{l} w_i^{(n)}}$$
   - Compute $\alpha_n = \ln(\frac{1-\mathrm{err}_n}{\mathrm{err}_n})$
   - Set $w_i^{(n+1)} = w_i^{(n)} \cdot \exp(\alpha_n \, I(y_i \neq g_n(\mathbf{x}_i))),\ i = 1, 2, ..., l$
3. Output $g(\mathbf{x}) = \mathrm{sign}(\sum_{n=1}^{N} \alpha_n \, g_n(\mathbf{x}))$

Basic ideas of boosting methods: Part 4

- In some sense: boosting is way of fitting an additive expansion in set of elementary basis functions, i.e.

$$g(\mathbf{x}) = \sum_{n=1}^{N} \beta_n \, b(\mathbf{x}; \gamma_n)$$

  - β_n for $n = 1, ..., N$: expansion coefficients
  - $b(\mathbf{x}; \gamma)$: simple functions with argument $\mathbf{x}$ and parameter γ
- Expansions like this play important role in many situations, e.g.:
  - Signal processing (wavelets)
  - Regression splines
  - Most importantly (for our purposes): trees: γ parametrizes split variables, split points at internal nodes and predictions at terminal nodes
- To fit these models, we minimize a loss function L (e.g. mean squared-error or likelihood-based) averaged over the training data. The choice of L is crucial.

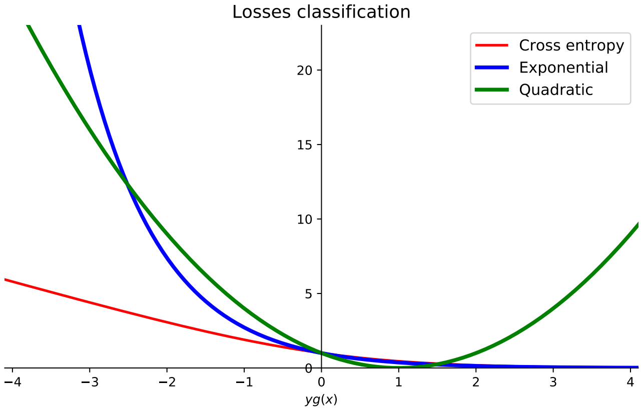
Basic ideas of boosting methods: Part 5

- Need to solve the following optimization problem:

$$\min_{\{\beta_n,\gamma_n\}_{n=1}^N} \sum_{i=1}^l L(y_i, \sum_{n=1}^N \beta_n b(\mathbf{x}_i;\gamma_n))$$

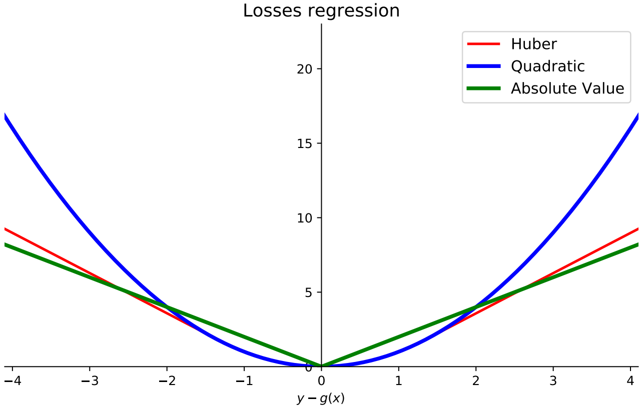
- Forward stagewise additive modelling:
  1. Initialize $g_0(\mathbf{x}) = 0$
  2. For $n = 1$ to N:
     - Compute $(\beta_n, \gamma_n) = \arg\min_{\beta,\gamma} \sum_{i=1}^l L(y_i, g_{n-1}(\mathbf{x}_i) + \beta b(\mathbf{x}_i;\gamma))$
     - Set $g_n(\mathbf{x}) = g_{n-1}(\mathbf{x}) + \beta_n b(\mathbf{x};\gamma_n)$
- At each step n: solve for optimal basis function $b(\mathbf{x};\gamma_n)$ and coefficient β_n to add to current expression $g_{n-1}(\mathbf{x})$. Coefficients from previous steps not changed.
- Can be shown: AdaBoost.M1 is equivalent to forward stagewise additive modelling for $L(y, g(\mathbf{x})) = \exp(-y\, g(\mathbf{x}))$ (see exercises).
- Forward stagewise additive modelling is more robust, can be applied to various loss functions and settings.

Robust losses for classification



- Exponential and quadratic loss tend to punish outliers too strongly

Robust losses for regression



- Quadratic loss tends to punish outliers too strongly; absolute value punishes already close points too strongly