Trees, Bagging, and Boosting

He He
(adapted from David Rosenberg's slides)

CDS, NYU

April 14, 2019

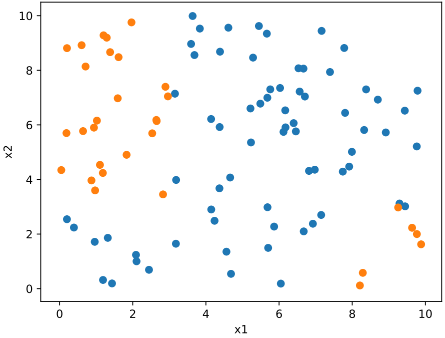
Contents

1. Decision Trees
    - Input Space Partition
    - Node Splitting
    - Trees in General
2. Bagging and Random Forests
    - Variance of an Estimator
    - The Benefits of Averaging
    - Bootstrap
    - Bagging
    - Random Forests
3. Boosting
    - Adaboost: The Algorithm

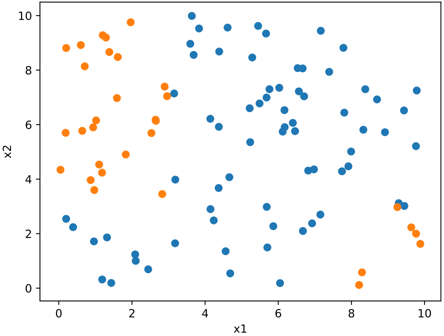
Decision Trees

Motivating example in 2d

- Partition data into different (axis-aligned) regions recursively

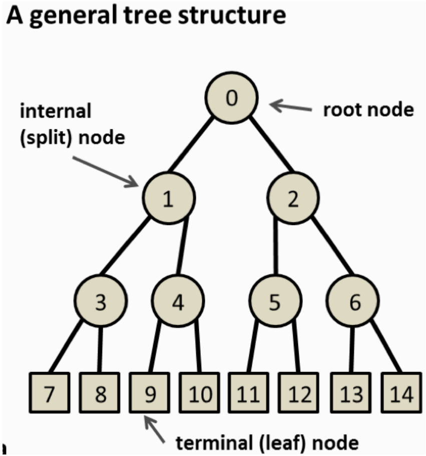


Classification flowchart



Is this a linear or non-linear classifier?

Decision trees setup



We'll only consider

- binary trees (vs multiway trees where nodes can have more than 2 children)
- each node contains a subset of data points
- decisions at each node involve only a single feature (i.e. input coordinate)
- for continuous variables, splits always of the form
$$x_i \leqslant t$$
- for discrete variables, partitions values into two groups

From Criminisi et al. MSR-TR-2011-114, 28 October 2011.

Regularization of decision trees

- What will happen if we keep splitting the data?
  - Every data point will be in its own region—overfitting.
- When to stop splitting? (control complexity of the hypothesis space)
  - Limit number of total nodes.
  - Limit number of terminal nodes.
  - Limit tree depth.
  - Require minimum number of data points in a terminal node.
  - **Backward pruning** – the approach of **CART** (Breiman et al 1984):
    1. Build a really big tree (e.g. until all regions have $\leqslant 5$ points).
    2. Prune the tree back greedily all the way to the root, assessing performance on validation.

How to split

Goal Find a tree that minimize the task loss (e.g., squared loss) within a given complexity.

Problem Finding the optimal binary is computationally intractable.

Solution Greedy algorithm.

- Find the best split (according to some criteria) for a terminal node (initially the root)
- Add two children nodes
- Repeat until a stopping criterion is reached (e.g., max depth)

Evaluate splits

Let's think about what makes a good split.

Which one is better?

Split 1 $R_1 : 8+/2-$ $\quad R2 : 2+/8-$

Split 2 $R_1 : 6+/4-$ $\quad R2 : 1+/9-$

Which one is better?

Split 1 $R_1 : 8+/2-$ $\quad R2 : 2+/8-$

Split 2 $R_1 : 6+/4-$ $\quad R2 : 0+/10-$

In general, we want to produce pure nodes, i.e. close to single-class node.

Misclassification error in a node

Let's formalize things a bit.

- Consider classification case: $\mathcal{Y} = \{1, 2, \ldots, K\}$.
- What's in a node?
  - Let node m represent region R_m, with N_m observations
  - Denote proportion of observations in R_m with class k by

$$\hat{p}_{mk} = \frac{1}{N_m} \sum_{\{i : x_i \in R_m\}} 1(y_i = k).$$

- Predict the majority class in node m:

$$k(m) = \arg\max_k \hat{p}_{mk}.$$

- Misclassification rate in node m:

$$1 - \hat{p}_{mk(m)}.$$

Node Impurity Measures

How to quantify impurity?

- Three measures of **node impurity** for leaf node m:

  Misclassification error

  $$1-\hat{p}_{mk(m)}.$$

  Gini index

  $$\sum_{k=1}^{K}\hat{p}_{mk}(1-\hat{p}_{mk})$$

  Entropy

  $$-\sum_{k=1}^{K}\hat{p}_{mk}\log\hat{p}_{mk}.$$

- Gini index and entropy work well in practice.

Impurity of a split

A potential split produces two nodes, R_L and R_R. How do we score it?

- Suppose we have N_L points in R_L and N_R points in R_R.
- Let $Q(R_L)$ and $Q(R_R)$ be the node impurity measures for each node.
- Then find split that minimizes the weighted average of node impurities:

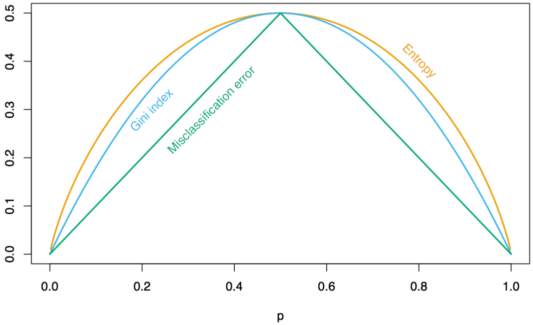
$$\frac{N_L Q(R_L) + N_R Q(R_R)}{N_L + N_R}$$

Example:

$R_1 : 8+/2-$ $\quad$ $R2 : 1+/4-$
What's the weighted misclassification rate?

Two-Class Node Impurity Measures

Consider binary classification. Let p be the relative frequency of class 1.



Misclassification error is not strictly concave thus may not guarantee improvement over the parent node.

HTF Figure 9.3

Finding the Split Point

How to find a split point that minimizes a given impurity measure?

- Consider splitting on the j'th feature x_j.
- If $x_{j(1)}, \ldots, x_{j(n)}$ are the sorted values of the j'th feature,
  - we only need to check split points between adjacent values
  - traditionally take split points halfway between adjacent values:

$$s_j \in \left\{ \frac{1}{2}\left(x_{j(r)} + x_{j(r+1)}\right) \mid r = 1, \ldots, n-1 \right\}. \qquad n-1 \text{ splits} \qquad (1)$$

- Enumerate d features and $n-1$ split points for each feature.

Regression trees

- Predict the mean value of a node

$$k(m) = \text{mean}(y_i \mid x_i \in R_m). \tag{2}$$

- Squared loss as the node impurity measure.
- Everything else remains the same as classification trees.

Categorical features

- For a categorical feature, we split its values into two groups.
- Given a set of categories of size k, how many distinct splits? (its power set)
- Finding the optimal split is intractable in general.
- Approximations

  Numeric encoding Random or proportion of class 0 (for binary classification)

  One-hot encoding May grow imbalanced trees, e.g., left-branching

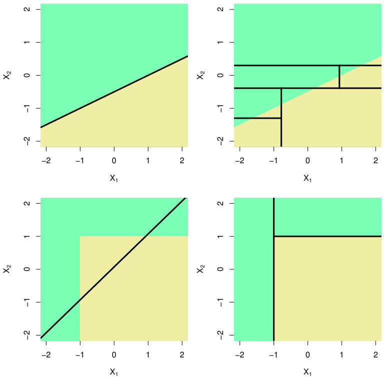
  Binary encoding Robust to large cardinality
- Statistical issues with categorical features
  - If a category has a very large number of categories, we can overfit.
  - Extreme example: Row Number could lead to perfect classification with a single split.

Interpretability

- Trees are certainly easier to explain than other classifiers.
- Can be used to discover non-linear features.
- Small trees seem interpretable. For large trees, maybe not so easy.
- Approximate neural network decision boundaries to gain interpretability
  - Wu M, Hughes M, Parbhoo S, Zazzi M, Roth V, Doshi-Velez F. Beyond Sparsity: Tree Regularization of Deep Models for Interpretability. Association for the Advancement of Artificial Intelligence (AAAI). 2018

Trees vs linear models

Trees have to work much harder to capture linear relations.



Review

Decision trees:

- Non-linear classifier that recursively partitions the input space.
- Non-metric: make no use of geometry, i.e. no inner-product or distances.
- Non-parametric: make no assumption of the data distribution.

Pros:

- Simple to understand.
- Interpretable, feature selection for free.

Cons:

- Poor linear modeling.
- Unstable / high variance, tend to overfit. → Next, how to fix this.

Bagging and Random Forests

Recap: statistic and point estimator

- Observe data $\mathcal{D} = (x_1, x_2, \ldots, x_n)$ sampled i.i.d. from a parametric distribution $p(\cdot \mid \theta)$.
- A **statistic** $s = s(\mathcal{D})$ is any function of the data.
  - E.g., sample mean, sample variance, histogram, empirical data distribution
- A statistic $\hat{\theta} = \hat{\theta}(\mathcal{D})$ is a **point estimator** of θ if $\hat{\theta} \approx \theta$.

Review questions

In frequentist statistics,

- Is θ random?
- Is $\hat{\theta}$ random?
- Is the function $s(\cdot)$ random?

Recap: bias and variance of an estimator

- Statistics are random, so they have probability distributions.
- The distribution of a statistic is called a **sampling distribution**.
- What are some parameters of the sampling distribution we might be interested in?

$$\text{Bias} \quad \text{Bias}(\hat{\theta}) \stackrel{\text{def}}{=} \mathbb{E}\left[\hat{\theta}\right] - \theta.$$

$$\text{Variance} \quad \text{Var}(\hat{\theta}) \stackrel{\text{def}}{=} \mathbb{E}\left[\hat{\theta}^2\right] - \mathbb{E}^2\left[\hat{\theta}\right].$$

- Is bias and variance random?
  - Neither bias nor variance depend on a specific sample $\mathcal{D}_n$. We are taking expectation over $\mathcal{D}$.
- Why do we care about variance?
  - $\hat{\theta}(\mathcal{D}) = x_1$ is an unbiased estimator of the mean of a Gaussian, but would be farther away from θ than the sample mean.

Variance of a Mean

Using a single estimate may have large standard error

- Let $\hat{\theta}(\mathcal{D})$ be an unbiased estimator: $\mathbb{E}\left[\hat{\theta}\right] = \theta$, $\text{Var}(\hat{\theta}) = \sigma^2$.
- We could use a single estimate $\hat{\theta} = \hat{\theta}(\mathcal{D})$ to estimate θ.
- The standard error is $\sqrt{\text{Var}(\hat{\theta})} = \sigma$.

Average of estimates has smaller standard error

- Consider a new estimator that takes the average of i.i.d. $\hat{\theta}_1, \ldots, \hat{\theta}_n$ where $\hat{\theta}_i = \hat{\theta}(\mathcal{D}^i)$.
- Average has the same expected value but smaller standard error:

$$\boxed{\mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n}\hat{\theta}_i\right] = \theta \quad \text{Var}\left[\frac{1}{n}\sum_{i=1}^{n}\hat{\theta}_i\right] = \frac{\sigma^2}{n}} \tag{3}$$

Averaging Independent Prediction Functions

Let's apply averaging to reduce variance of prediction functions.

- Suppose we have B independent training sets from the same distribution ($\mathcal{D} \sim p(\cdot \mid \theta)$).
- Learning algorithm (estimator) gives B decision functions: $\hat{f}_1(x), \hat{f}_2(x), \ldots, \hat{f}_B(x)$
- Define the average prediction function as:

$$\hat{f}_{\text{avg}} \overset{\text{def}}{=} \frac{1}{B} \sum_{b=1}^{B} \hat{f}_b \tag{4}$$

- What's random here?
- The B independent training sets are random, which gives rise to variation among the $\hat{f}_b$'s.
- Concept check: What's the distribution of $\hat{f}$ called? What do we know about the distribution?

Averaging reduce variance of predictions

- The average prediction on x_0 is

$$\hat{f}_{\text{avg}}(x_0) = \frac{1}{B}\sum_{b=1}^{B} \hat{f}_b(x_0).$$

- $\hat{f}_{\text{avg}}(x_0)$ and $\hat{f}_b(x_0)$ have the same expected value, but
- $\hat{f}_{\text{avg}}(x_0)$ has smaller variance (see 3):

$$\mathrm{Var}(\hat{f}_{\text{avg}}(x_0)) = \frac{1}{B}\mathrm{Var}\left(\hat{f}_1(x_0)\right)$$

- Problem: in practice we don't have B independent training sets...

The Bootstrap Sample

How do we simulate multiple samples when we only have one?

- A **bootstrap sample** from $\mathcal{D}_n = (x_1, \ldots, x_n)$ is a sample of size n drawn with replacement from $\mathcal{D}_n$.
- Some elements of $\mathcal{D}_n$ will show up multiple times, and some won't show up at all.

How similar are the bootstrap samples?

- Each x_i has a probability of $(1 - 1/n)^n$ of not being selected.
- Recall from analysis that for large n,

$$\boxed{\left(1 - \frac{1}{n}\right)^n \approx \frac{1}{e} \approx .368.} \tag{5}$$

- So we expect ~63.2% of elements of $\mathcal{D}_n$ will show up at least once.

The Bootstrap Method

Definition

A **bootstrap method** is when you simulate having B independent samples from P by taking B bootstrap samples from the sample $\mathcal{D}_n$.

- Given original data $\mathcal{D}_n$, compute B bootstrap samples $D_n^1, \ldots, D_n^B$.
- For each bootstrap sample, compute some function

$$\phi(D_n^1), \ldots, \phi(D_n^B)$$

- Work with these values as though $D_n^1, \ldots, D_n^B$ were i.i.d. samples from P.
- **Amazing fact:** This is often very close to what we'd get with independent samples from P.

Independent vs Bootstrap Samples

- Want to estimate $\alpha = \alpha(P)$ for some unknown P and some complicated α.
- Point estimator $\hat{\alpha} = \hat{\alpha}(\mathcal{D}_{100})$ for samples of size 100.
- Histogram of $\hat{\alpha}$ based on
  - 1000 independent samples of size 100, vs
  - 1000 bootstrap samples of size 100

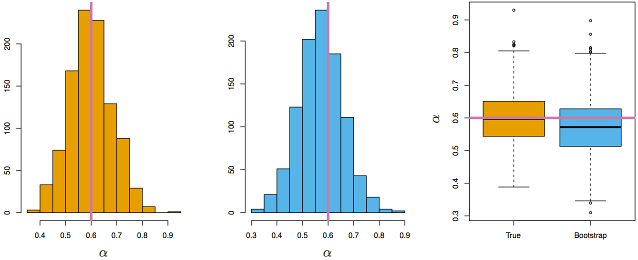


Figure 5.10 from ISLR (Springer, 2013) with permission from the authors: G. James, D. Witten, T. Hastie and R. Tibshirani.

Side note: Bootstrap in Practice

We can use bootstrap to get error bars in a cheap way.

- Suppose we have an estimator $\hat{\theta} = \hat{\theta}(\mathcal{D}_n)$, e.g., the accuracy of your classifier.
- To get error bars, we can compute the "bootstrap variance".
  - Draw B bootstrap samples.
  - Compute sample variance of $\hat{\theta}(\mathcal{D}_n^1), \ldots, \hat{\theta}(\mathcal{D}_n^B)$..
  - Could report

$$\hat{\theta}(\mathcal{D}_n) \pm \sqrt{\text{Bootstrap Variance}}$$

Ensemble methods

Key ideas:

- Averaging i.i.d. estimates reduces variance without making bias worse.
- Can use bootstrap to simulate multiple data samples.

Ensemble methods:

- Combine outputs from multiple models.
  - Same learner on different datasets: ensemble + bootstrap = bagging.
  - Different learners on one dataset: they may make similar errors.
- Parallel ensemble: models are built independently, e.g., bagging
- Sequential ensemble: models are built sequentially, e.g., boosting
  - Try to add new learners that do well where previous learners lack

Bagging

- Draw B bootstrap samples $D^1, \ldots, D^B$ from original data $\mathcal{D}$.
- Let $\hat{f}_1, \hat{f}_2, \ldots, \hat{f}_B$ be the prediction functions from training on $D^1, \ldots, D^B$, respectively.
- The **bagged prediction function** is a **combination** of these:

$$\hat{f}_{\text{avg}}(x) = \text{Combine}\left(\hat{f}_1(x), \hat{f}_2(x), \ldots, \hat{f}_B(x)\right)$$

- How might we combine
  - prediction functions for regression?
  - binary class predictions?
  - binary probability predictions?
  - multiclass predictions?

Out-of-Bag Error Estimation

- Each bagged predictor is trained on about 63% of the data.
- Remaining 37% are called **out-of-bag (OOB)** observations.
- For *i*th training point, let

$$S_i = \left\{ b \mid D^b \text{ does not contain } i\text{th point} \right\}.$$

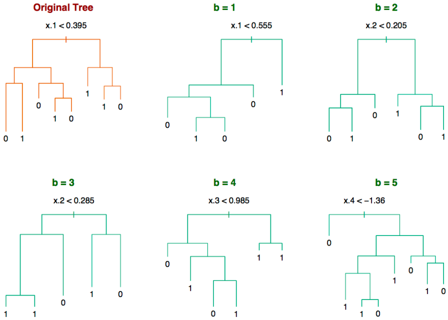
- The **OOB prediction** on x_i is

$$\hat{f}_{\text{OOB}}(x_i) = \frac{1}{|S_i|} \sum_{b \in S_i} \hat{f}_b(x_i).$$

- The OOB error is a good estimate of the test error.
- OOB error is similar to cross validation error – both are computed on training set.

Bagging Classification Trees

- Input space $\mathcal{X} = \mathbf{R}^5$ and output space $\mathcal{Y} = \{-1, 1\}$. Sample size $n = 30$.



- Each bootstrap tree is quite different: different splitting variable at the root
- **High variance**: high degree of model variability from small perturbations of the training data.
- Conventional wisdom: Bagging helps most when base learners are relatively unbiased but has high variance / low stability $\Longrightarrow$ decision trees.

From HTF Figure 8.9

Variance of a Mean of Correlated Variables

Recall the motivating principle of bagging:

- For $\hat{\theta}_1, \dots, \hat{\theta}_n$ i.i.d. with $\mathbb{E}\left[\hat{\theta}\right] = \theta$ and $\mathrm{Var}\left[\hat{\theta}\right] = \sigma^2$,

$$\mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n}\hat{\theta}_i\right] = \mu \qquad \mathrm{Var}\left[\frac{1}{n}\sum_{i=1}^{n}\hat{\theta}_i\right] = \frac{\sigma^2}{n}.$$

- What if $\hat{\theta}$'s are correlated?
- Suppose $\forall i \neq j$, $\mathrm{Corr}(\hat{\theta}_i, \hat{\theta}_j) = \rho$. Then

$$\mathrm{Var}\left[\frac{1}{n}\sum_{i=1}^{n}\hat{\theta}_i\right] = \rho\sigma^2 + \frac{1-\rho}{n}\sigma^2.$$

- For large n, the $\rho\sigma^2$ term dominates – limits benefit of averaging.

Correlation between bootstrap samples

- Averaging $\hat{f}_1, \ldots, \hat{f}_B$ reduces variance if they're based on i.i.d. samples from $P_{\mathcal{X} \times \mathcal{Y}}$
- Bootstrap samples are
  - independent samples from the training set, but
  - are not independent samples from $P_{\mathcal{X} \times \mathcal{Y}}$.
- This dependence limits the amount of variance reduction we can get.
- Solution: reduce the dependence between $\hat{f}_i$'s.

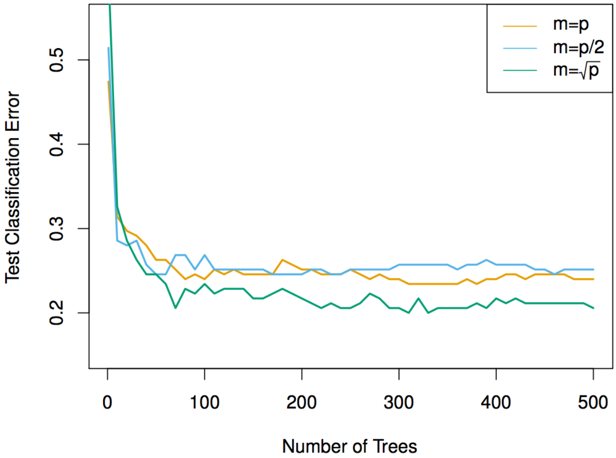
Random Forest

Key idea

Use bagged decision trees, but modify the tree-growing procedure to reduce the dependence between trees.

- Build a collection of trees independently (in parallel).
- When constructing each tree node, restrict choice of splitting variable to a randomly chosen subset of features of size m.
  - Avoid dominance by strong features.
- Typically choose $m \approx \sqrt{p}$, where p is the number of features.
- Can choose m using cross validation.

Random Forest: Effect of m size



From An Introduction to Statistical Learning, with applications in R (Springer, 2013) with permission from the authors: G. James, D. Witten, T. Hastie and R. Tibshirani.

Review

- Usual approach is to build very deep trees—low bias but high variance
- Ensembling many models reduces variance
  - Motivation: Mean of i.i.d. estimates has smaller variance than single estimate.
- Use bootstrap to simulate many data samples from one dataset
  - $\Longrightarrow$ Bagged decision trees
- But bootstrap samples (and the induced models) are correlated.
- Bagging seems to work better when we are combining a diverse set of prediction functions.
  - $\Longrightarrow$ random forests (randomized tree building)

Boosting

Overview

Bagging Reduce variance of a low bias, high variance estimator by ensembling many estimators trained in parallel.

Boosting Reduce the error rate of a high bias estimator by ensembling many estimators trained in sequential.

- A **weak/base learner** is a classifier that does slightly better than random.
- Weak learners are like "rules of thumb":
  - "Viagra" $\implies$ spam
  - From a friend $\implies$ not spam
- **Key idea**:
  - Each weak learner focuses on different examples (reweighted data)
  - Weak learners have different contributions to the final prediction (reweighted classifier)

AdaBoost: Setting

- Binary classification: $\mathcal{Y} = \{-1, 1\}$
- Base hypothesis space $\mathcal{H} = \{h : \mathcal{X} \to \{-1, 1\}\}$.
- Typical base hypothesis spaces:
  - **Decision stumps** (tree with a single split)
  - Trees with few terminal nodes
  - Linear decision functions

Weighted Training Set

Each base learner is trained on weighted data.

- Training set $\mathcal{D} = ((x_1, y_1), \ldots, (x_n, y_n))$.
- Weights $(w_1, \ldots, w_n)$ associated with each example.
- **Weighted empirical risk**:

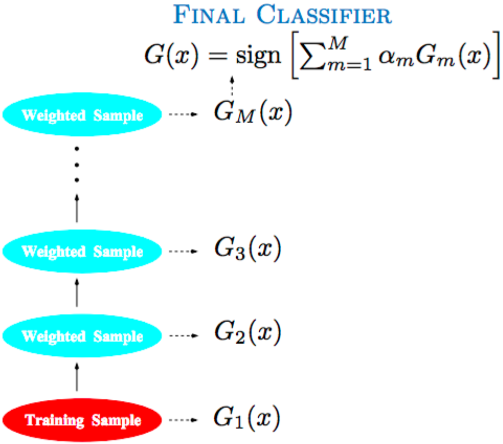
$$\hat{R}_n^w(f) = \frac{1}{W} \sum_{i=1}^{n} w_i \ell(f(x_i), y_i) \quad \text{where } W = \sum_{i=1}^{n} w_i$$

- Examples with larger weights have more influence on the loss.

AdaBoost - Rough Sketch

- Training set $\mathcal{D} = ((x_1, y_1), \ldots, (x_n, y_n))$.
- Start with equal weight on all training points $w_1 = \cdots = w_n = 1$.
- Repeat for $m = 1, \ldots, M$:
  - Find base classifier $G_m(x)$ that tries to fit weighted training data (but may not do that well)
  - Increase weight on the points $G_m(x)$ misclassifies
- So far, we've generated M classifiers: $G_1, \ldots, G_M : \mathcal{X} \to \{-1, 1\}$.

AdaBoost: Schematic



From ESL Figure 10.1

AdaBoost - Rough Sketch

- Training set $\mathcal{D} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$.
- Start with equal weight on all training points $w_1 = \cdots = w_n = 1$.
- Repeat for $m = 1, \ldots, M$:
    - Base learner fits weighted training data and returns $G_m(x)$
    - Increase weight on the points $G_m(x)$ misclassifies
- Final prediction $G(x) = \operatorname{sign}\left[\sum_{m=1}^{M} \alpha_m G_m(x)\right]$. (recall $G_m(x) \in \{-1, 1\}$)
- What are desirable α_m's?
    - nonnegative
    - larger when G_m fits its weighted $\mathcal{D}$ well
    - smaller when G_m fits weighted $\mathcal{D}$ less well

Adaboost: Weighted Classification Error

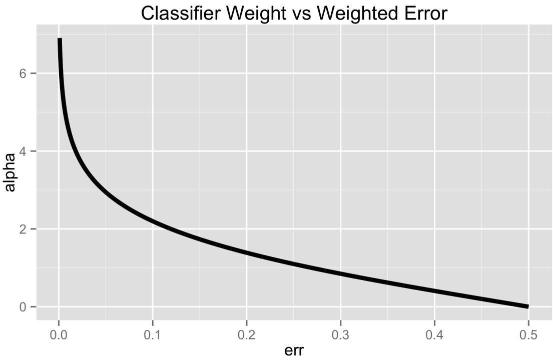
- Weights of base learners depend on their performance. How to evaluate each base learner?
- In round m, base learner gets a weighted training set.
  - Returns a base classifier $G_m(x)$ that minimizes weighted $0-1$ error.
- The **weighted 0-1 error** of $G_m(x)$ is

$$\text{err}_m = \frac{1}{W}\sum_{i=1}^{n} w_i 1(y_i \neq G_m(x_i)) \quad \text{where } W = \sum_{i=1}^{n} w_i.$$

- Notice: $\text{err}_m \in [0,1]$.

AdaBoost: Classifier Weights

- The weight of classifier $G_m(x)$ is $\alpha_m = \ln\left(\frac{1-\mathrm{err}_m}{\mathrm{err}_m}\right)$.



- Higher weighted error $\Longrightarrow$ lower weight

Adaboost: Example Reweighting

We want the base learner to focus more on examples misclassified by the previous learner.

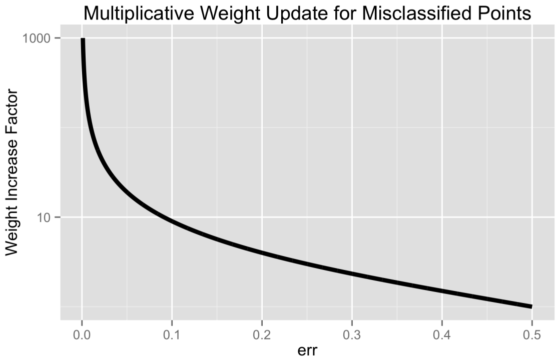
- We train G_m to minimize weighted error, and it achieves err_m.
- Then $\alpha_m = \ln\left(\frac{1-\text{err}_m}{\text{err}_m}\right)$ is the weight of G_m in final ensemble.
- Suppose w_i is weight of example i before training:
  - If G_m classfies x_i correctly, then w_i is unchanged.
  - Otherwise, w_i is increased as

$$\begin{aligned} w_i &\leftarrow w_i e^{\alpha_m} \\ &= w_i \left(\frac{1-\text{err}_m}{\text{err}_m}\right) \end{aligned}$$

  - For $\text{err}_m < 0.5$ (weak learner), this always increases the weight.

Adaboost: Example Reweighting

- Any misclassified point has weight adjusted as $w_i \leftarrow w_i \left(\frac{1-\text{err}_m}{\text{err}_m}\right)$.



- The smaller err_m, the more we increase weight of misclassified points.

AdaBoost: Algorithm

Given training set $\mathcal{D} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$.

1. Initialize observation weights $w_i = 1$, $i = 1, 2, \ldots, n$.
2. For $m = 1$ to M:
    1. Base learner fits weighted training data and returns $G_m(x)$
    2. Compute weighted empirical 0-1 risk:

$$\mathrm{err}_m = \frac{1}{W} \sum_{i=1}^{n} w_i 1(y_i \neq G_m(x_i)) \quad \text{where } W = \sum_{i=1}^{n} w_i .$$

    3. Compute classifier weight: $\alpha_m = \ln\left(\frac{1 - \mathrm{err}_m}{\mathrm{err}_m}\right)$.
    4. Update example weight: $w_i \leftarrow w_i \cdot \exp\left[\alpha_m 1(y_i \neq G_m(x_i))\right]$
3. Return voted classifier: $G(x) = \operatorname{sign}\left[\sum_{m=1}^{M} \alpha_m G_m(x)\right]$.

AdaBoost with Decision Stumps

- After 1 round:

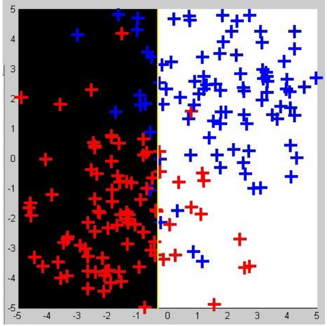


Figure: Plus size represents weight. Blackness represents score for red class.

KPM Figure 16.10

AdaBoost with Decision Stumps

- After 3 rounds:

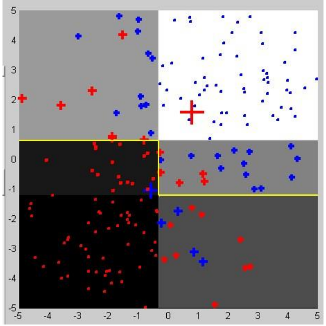


Figure: Plus size represents weight. Blackness represents score for red class.

KPM Figure 16.10

AdaBoost with Decision Stumps

- After 120 rounds:

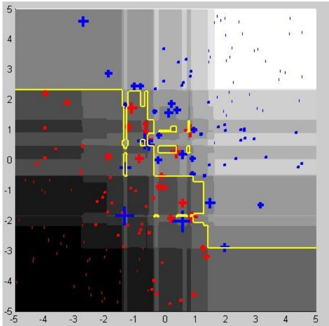
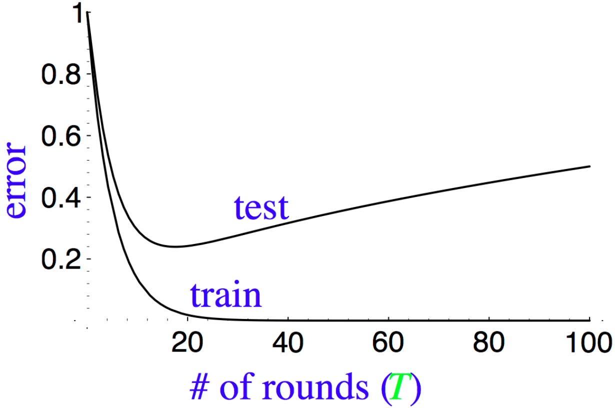


Figure: Plus size represents weight. Blackness represents score for red class.

KPM Figure 16.10

Typical Train / Test Learning Curves

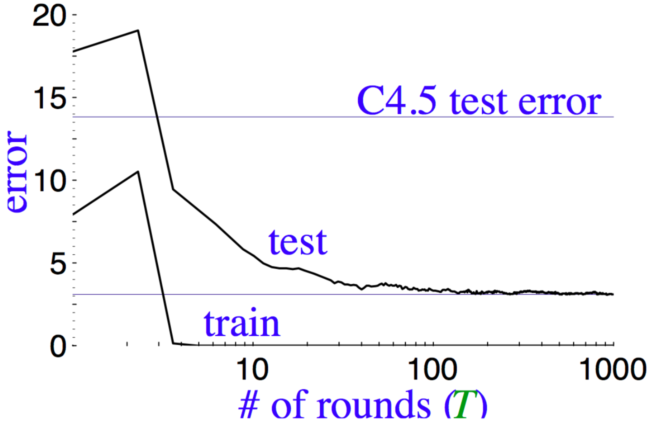
- Might expect too many rounds of boosting to overfit:



From Rob Schapire's NIPS 2007 Boosting tutorial.

Learning Curves for AdaBoost

- In typical performance, AdaBoost is surprisingly resistant to overfitting.
- Test continues to improve even after training error is zero!



From Rob Schapire's NIPS 2007 Boosting tutorial.

Summary

- Shallow decision tree + boosting
  - "best off-the-shelf classifier in the world" (Hastie et al., 2009)
  - Used in the first successful real-time face detector (Viola and Jones, 2001)
  - XGBoost: very popular in competitions
- Next week
  - What is the objective function of Adaboost?
  - Generalize to other loss functions.