L3 Apprentissage

Michèle Sebag – Benjamin Monmège
LRI – LSV

9 avril 2013

Overview

Introduction

Boosting
- PAC Learning
- Boosting
- Adaboost

Bagging
- General bagging
- Random Forests

Ensemble learning

Example (AT&T)

Schapire 09

- Categorizing customer's queries (Collect, CallingCard, PersonToPerson,..)
- Example queries:
  - yes I'd like to place a collect call long distance please **Collect**
  - operator I need to make a call but I need to bill it to my office **ThirdNumber**
  - yes I'd like to place a call on my master card please **CallingCard**
  - I just called a number in sioux city and I musta rang the wrong number because I got the wrong party and I would like to have that taken off of my bill **BillingCredit**

Ensemble learning

Example (AT&T)

Schapire 09

- Categorizing customer's queries (Collect, CallingCard, PersonToPerson,..)
- Example queries:
  - yes I'd like to place a collect call long distance please **Collect**
  - operator I need to make a call but I need to bill it to my office **ThirdNumber**
  - yes I'd like to place a call on my master card please **CallingCard**
  - I just called a number in sioux city and I musta rang the wrong number because I got the wrong party and I would like to have that taken off of my bill **BillingCredit**

Remark

- Easy to find rules of thumb with good accuracy

  IF **'card'**, THEN **CallingCard**

- Hard to find a single good rule

Ensemble learning

Procedure

- A learner (fast, reasonable accuracy, i.e. accuracy $>$ random)
- Learn from (a subset of) training set
- Find a hypothesis
- Do this a zillion times (T rounds)
- Aggregate the hypotheses

Critical issues

- Enforce the diversity of hypotheses
- How to aggregate hypotheses

The Wisdom of Crowds: Why the Many Are Smarter Than the Few and How Collective Wisdom Shapes Business, Economies, Societies and Nations. J. Surowiecki, 2004.

Ensemble learning

$$\mathcal{E} = \{(x_i, y_i)\}, \; x_i \in \; X, \; y_i \; \in \; \{-1, 1\}\} \qquad (x_i, y_i) \sim D(x, y)$$

Loop

For $t = 1 \ldots T$, learn $h_{\mathbf{t}}$ from $\mathcal{E}_{\mathbf{t}}$

Result: $H = sign\left(\sum_t \alpha_t h_t\right)$

Requisite: Classifiers h_t must be diverse

Enforcing diversity through:

- Using different training sets $\mathcal{E}_t$ — bagging, boosting
- Using diverse feature sets — bagging
- Enforce h_t decorrelation — boosting

Diversity of h_t, Stability of H

Stability: slight changes of $\mathcal{E}$ hardly modifies h

Stable learners

k-nearest neighbors
Linear discriminant analysis (LDA)

Unstable learners

Neural nets
Decision trees

Instability: Why is it useful?

Bias of $\mathcal{H}$: $Err(h^*)$

$$h^* = \arg\min\{Err(h),\ h \in \mathcal{H}\}$$

Decreases with size/complexity of $\mathcal{H}$ LDA poor...

Variance: $h_1, \ldots h_T$ H = average $\{h_t\}$

$$\text{Variance } (H) = \frac{1}{T} \sum_t ||H - h_t||^2$$

Instable learners

- Large variance
- Small bias (e.g. decision trees and NN are universal approximators)
- Variance(Ensemble) decreases with size of ensemble **if** ensemble elements are not correlated.

$$\text{Variance } (H) \approx \frac{1}{T} Variance(h_t)$$

Why does it work, basics

- Suppose there are 25 base classifiers
- Each one with error rate $\epsilon = .35$
- Assume classifiers are **independent**

Then, probability that the ensemble classifier makes a wrong prediction:

$$\Pr\left(\text{ensemble makes error}\right) = \sum_{i=13}^{2} 5C_{25}^{i}\epsilon^{i}(1-\epsilon)^{25-i} = .06$$

Results

Caruana and Niculesu-Mizil, ICML 2006

MODEL	1ST	2ND	3RD	4TH	5TH	6TH	7TH	8TH	9TH	10TH
BST-DT	0.580	0.228	0.160	0.023	0.009	0.000	0.000	0.000	0.000	0.000
RF	0.390	0.525	0.084	0.001	0.000	0.000	0.000	0.000	0.000	0.000
BAG-DT	0.030	0.232	0.571	0.150	0.017	0.000	0.000	0.000	0.000	0.000
SVM	0.000	0.008	0.148	0.574	0.240	0.029	0.001	0.000	0.000	0.000
ANN	0.000	0.007	0.035	0.230	0.606	0.122	0.000	0.000	0.000	0.000
KNN	0.000	0.000	0.000	0.009	0.114	0.592	0.245	0.038	0.002	0.000
BST-STMP	0.000	0.000	0.002	0.013	0.014	0.257	0.710	0.004	0.000	0.000
DT	0.000	0.000	0.000	0.000	0.000	0.000	0.004	0.616	0.291	0.089
LOGREG	0.000	0.000	0.000	0.000	0.000	0.000	0.040	0.312	0.423	0.225
NB	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.030	0.284	0.686

Overall rank by mean performance across problems and metrics (based on bootstrap analysis).

BST-DT: boosting with decision tree weak classifier

RF: random forest

BAG-DT: bagging with decision tree weak classifier

SVM: support vector machine

ANN: neural nets

KNN: k nearest neighboorhood

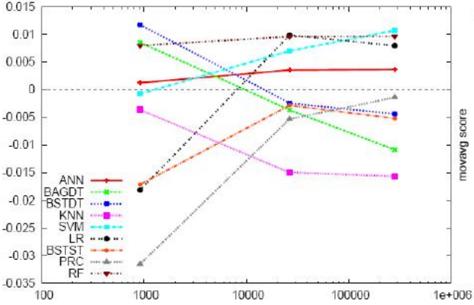
BST-STMP: boosting with decision stump weak classifier

DT: decision tree

LOGREG: logistic regression

NB: naïve Bayesian

Results w.r.t. dimension



Caruana et al., ICML 2008

Moving average standardized scores of each learning algorithm as a function of the dimension.

The rank for the algorithms to perform consistently well:

(1) random forest (2) neural nets (3) boosted tree (4) SVMs

Overview

PAC Learning: Probably Approximately Correct

Valiant 84

Turing award, 2011.

PAC Learning: Probably Approximately Correct

Setting

iid samples drawn after distribution $D(\mathbf{x}, y)$

$$\mathcal{E} = \{(x_i, y_i)\}, \ x_i \in \ X, \ y_i \ \in \ \{-1, 1\}\} \ (x_i, y_i) \sim D(x, y)$$

Strong learnability

Language (= set of target concepts) $\mathcal{C}$ is PAC learnable if there exists algorithm A s.t.

- For all $D(x, y)$
- For all $0 < \delta < 1$, with probability $1 - \delta$ *probably*
- For all error rate $\varepsilon > 0$, *approximately correct*
- There exists a number n of samples, n= Polynom($\frac{1}{\delta}, \frac{1}{\varepsilon}$)
- s.t. $A(\mathcal{E}_n) = \hat{h}_n$ with

$$Pr(Err(\hat{h}_n) < \epsilon) > 1 - \delta$$

$\mathcal{C}$ is polynomially PAC-learnable if
Computational cost learning($\hat{h}_n$) = Pol($\frac{1}{\delta}, \frac{1}{\varepsilon}$)

PAC Learning: Probably Approximately Correct, 2

Weak learnability

- Idem strong learnability
- Except that one only requires error to be $< 1/2$ (just better than random guessing)

$$\varepsilon = \frac{1}{2} - \gamma$$

Question

Kearns & Valiant 88

- Strong learnability $\Rightarrow$ weak learnability
- Weak learnability $\Rightarrow$ some stronger learnability ??

PAC Learning: Probably Approximately Correct, 2

Weak learnability

- Idem strong learnability
- Except that one only requires error to be $< 1/2$ (just better than random guessing)

$$\varepsilon = \frac{1}{2} - \gamma$$

Question

Kearns & Valiant 88

- Strong learnability $\Rightarrow$ weak learnability
- Weak learnability $\Rightarrow$ some stronger learnability ??

Yes !

Strong learnability $\Leftrightarrow$ Weak learnability

- PhD Rob. Schapire 89
- Yoav Freund, MLJ 1990: The strength of weak learnability
- Adaboost: Freund & Schapire 95

Overview

Introduction

Boosting
- PAC Learning
- Boosting
- Adaboost

Bagging
- General bagging
- Random Forests

Weak learnability $\Rightarrow$ Strong learnability

Thm Schapire MLJ 1990

Given algorithm able to learn with error $\eta = 1/2 - \gamma$ with complexity c

under any distribution $D(x, y)$

then there exists algorithm with complexity Pol(c), able to learn with error ε.

Proof (sketch)

- Learn h under $D(x, y)$
- Define $D'(x, y)$: $D(x, y) \wedge \left(Pr(h(x) \neq y) = \frac{1}{2}\right)$
- Learn h' under $D'(x, y)$
- Define $D''(x, y)$: $D(x, y) \wedge (h(x) \neq h'(x))$
- Learn h“ under $D''(x, y)$
- Use $Vote(h, h', h'')$

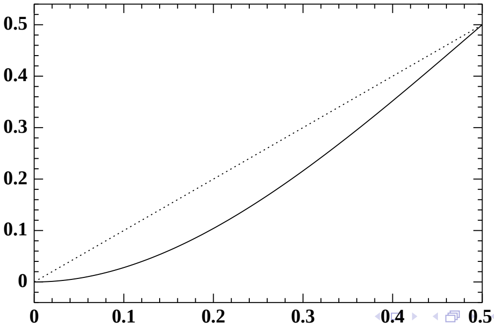
Proof (sketch)

$Vote(h, h', h'')$ true if

h and h' true, or $((h$ or h' wrong), and h'' true)

$$\begin{aligned} Pr(Vote(h, h', h'')\ OK) &= Pr(h\ OK\ and\ h'\ OK) + \\ &\quad Pr(h\ or\ h'\ \neg OK).Pr(h''\ OK) \\ &\geq 1 - (3\eta^2 - 2\eta^3) \end{aligned}$$

$$Err(h) < \eta \Rightarrow Err(Vote(h, h', h'')) < 3\eta^2 - 2\eta^3$$



Overview

Adaboost

Freund Schapire 95

http://videolectures.net/mlss09us_schapire_tab/

Given

algorithm A, weak learner

$$\mathcal{E} = \{(x_i, y_i)\},\ x_i \in \ \mathcal{X},\ \ y_i \ \in \ \{-1, 1\}, i = 1 \ldots n\}$$

Iterate

- For $t = 1, \ldots T$
  - Define distribution D_t on $\{1, \ldots n\}$
    focussing on examples misclassified by h_{t-1}
  - Draw $\mathcal{E}_t$ after D_t — or use example weights
  - Learn h_t

$$Pr_{x_i \sim D_t}(h_t(x_i) \neq y_i) = \varepsilon$$

- Return: weighted vote of h_t

Adaboost

Init: D_1 uniform distribution

$$D_1(i) = \frac{1}{n}$$

Define D_{t+1} as follows

$$\begin{aligned} D_{t+1}(i) &= \frac{1}{Z_t} D_t(i) \times \begin{cases} exp(-\alpha_t) & \text{if } h_t(x_i) = y_i \\ exp(\alpha_t) & \text{if } h_t(x_i) \neq y_i \end{cases} \\ &= \frac{1}{Z_t} D_t(i) \times exp(-\alpha_t h_t(x_i) y_i) \end{aligned}$$

With

- Z_t: normalisation term
- $\alpha_t = \frac{1}{2} ln \left(\frac{1-\varepsilon_t}{\varepsilon_t} \right) > 0$
- $\varepsilon_t = Pr_{x_i \sim D_t}(h_t(x_i) \neq y_i)$

Adaboost

Init: D_1 uniform distribution

$$D_1(i) = \frac{1}{n}$$

Define D_{t+1} as follows

$$\begin{aligned} D_{t+1}(i) &= \frac{1}{Z_t} D_t(i) \times \begin{cases} exp(-\alpha_t) & \text{if } h_t(x_i) = y_i \\ exp(\alpha_t) & \text{if } h_t(x_i) \neq y_i \end{cases} \\ &= \frac{1}{Z_t} D_t(i) \times exp(-\alpha_t h_t(x_i) y_i) \end{aligned}$$

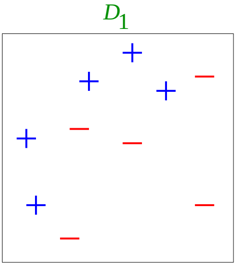
With

- Z_t: normalisation term
- $\alpha_t = \frac{1}{2} ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right) > 0$
- $\varepsilon_t = Pr_{x_i \sim D_t}(h_t(x_i) \neq y_i)$

Final hypothesis

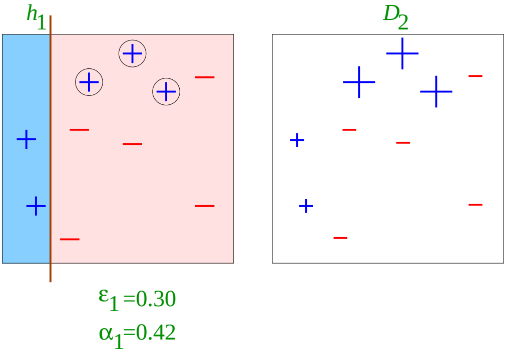
$$H(x) = sign\left(\sum_t \alpha_t h_t(x)\right)$$

Toy Example

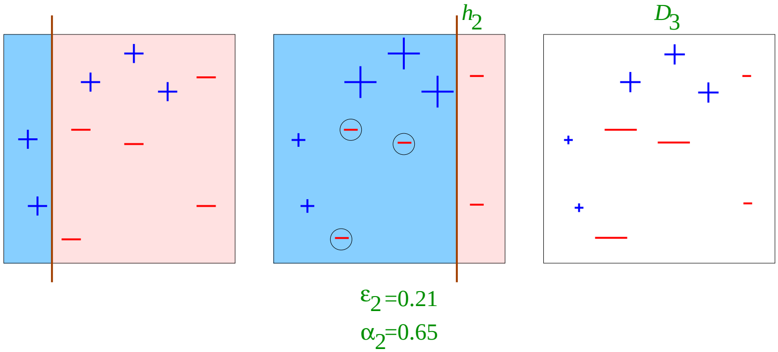


weak classifiers = vertical or horizontal half-planes

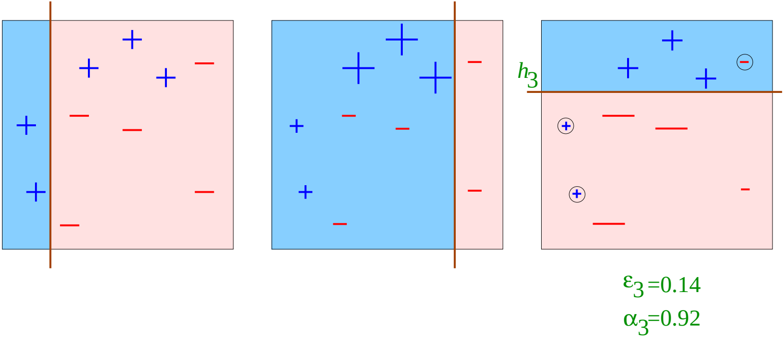
Round 1



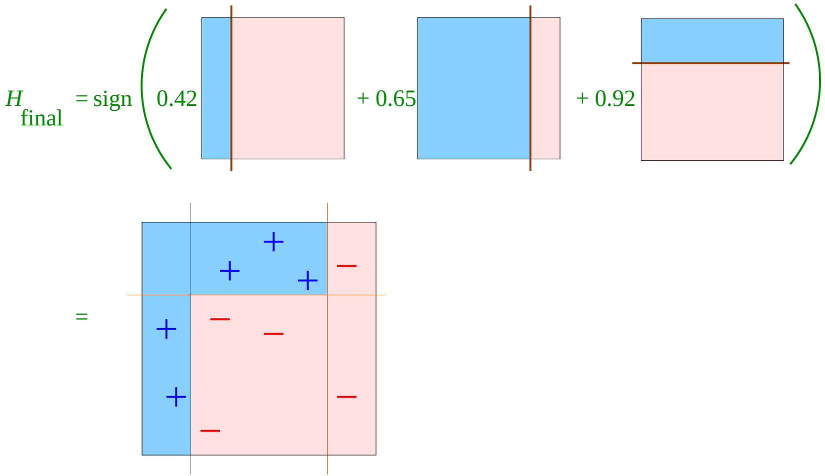
Round 2



Round 3



Final Classifier



Bounding training error

Thm Freund Schapire 95

Let $\varepsilon_t = \frac{1}{2} - \gamma_t$ γ_t edge

Then

$$
\begin{aligned}
Err_{train}(H) &\leq \prod_t \left[2\sqrt{\varepsilon_t(1-\varepsilon_t)} \right] \\
&= \prod_t \sqrt{1-4\gamma_t^2} \\
&\leq exp\left(-2\sum_t \gamma_t^2\right)
\end{aligned}
$$

Analysis

- If A weak learner, $\exists \gamma \ s.t. \ \forall t, \ \gamma_t > \gamma > 0$

$$Err_{train}(H) < exp\left(-2\gamma^2 T\right)$$

- Does not require γ or T to be known a priori

Proof

Note

$$F(x) = \sum_t \alpha_t h_t(x)$$

Step 1: final distribution

$$\begin{aligned} D_T(i) &= D_1(i) \prod_t \left[\frac{1}{Z_t} \; exp\left(-y_i \sum_t \alpha_t h_t(x_i)\right)\right] \\ &= \frac{1}{n} \prod_t \left[\frac{1}{Z_t}\right] \; exp\left(-y_i F(x_i)\right) \end{aligned}$$

Proof

Step 2

$$Err_{train}(H) \leq \prod_t Z_t$$

as

$$Err_{train}(H) = \frac{1}{n} \sum_i \begin{cases} 1 & \text{if } H(x_i) \neq y_i \\ 0 & \text{otherwise} \end{cases}$$

Proof

Step 2

$$Err_{train}(H) \leq \prod_t Z_t$$

as

$$\begin{aligned} Err_{train}(H) &= \frac{1}{n}\sum_i \begin{cases} 1 & \text{if } H(x_i) \neq y_i \\ 0 & \text{otherwise} \end{cases} \\ &= \frac{1}{n}\sum_i \begin{cases} 1 & \text{if } y_i F(x_i) < 0 \\ 0 & \text{otherwise} \end{cases} \end{aligned}$$

Proof

Step 2

$$Err_{train}(H) \leq \prod_t Z_t$$

as

$$\begin{aligned} Err_{train}(H) &= \frac{1}{n}\sum_i \begin{cases} 1 & \text{if } H(x_i) \neq y_i \\ 0 & \text{otherwise} \end{cases} \\ &= \frac{1}{n}\sum_i \begin{cases} 1 & \text{if } y_i F(x_i) < 0 \\ 0 & \text{otherwise} \end{cases} \\ &\leq \frac{1}{n} exp\left(-y_i F(x_i)\right) \end{aligned}$$

Proof

Step 2

$$Err_{train}(H) \leq \prod_t Z_t$$

as

$$\begin{aligned} Err_{train}(H) &= \frac{1}{n}\sum_i \begin{cases} 1 & \text{if } H(x_i) \neq y_i \\ 0 & \text{otherwise} \end{cases} \\ &= \frac{1}{n}\sum_i \begin{cases} 1 & \text{if } y_i F(x_i) < 0 \\ 0 & \text{otherwise} \end{cases} \\ &\leq \frac{1}{n} exp\left(-y_i F(x_i)\right) \\ &= \sum_i D_T(i) \prod_t Z_t \end{aligned}$$

Proof

Step 2

$$Err_{train}(H) \leq \prod_t Z_t$$

as

$$\begin{aligned} Err_{train}(H) &= \frac{1}{n}\sum_i \begin{cases} 1 & \text{if } H(x_i) \neq y_i \\ 0 & \text{otherwise} \end{cases} \\ &= \frac{1}{n}\sum_i \begin{cases} 1 & \text{if } y_i F(x_i) < 0 \\ 0 & \text{otherwise} \end{cases} \\ &\leq \frac{1}{n} exp\left(-y_i F(x_i)\right) \\ &= \sum_i D_T(i) \prod_t Z_t \\ &= \prod_t Z_t \end{aligned}$$

Proof

Step 3

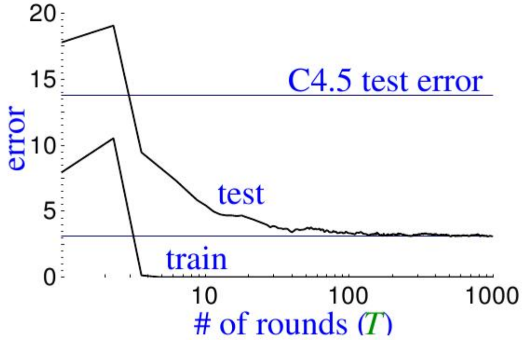
$$Z_t = 2\sqrt{\varepsilon_t(1-\varepsilon_t)}$$

Because

$$\begin{aligned} Z_t &= \sum_i D_t(i) exp\left(-\alpha_t y_i h_t(x_i)\right) \\ &= \sum_{i \ / \ h_t(x_i) \neq y_i} D_t(i) e^{\alpha_t} + \sum_{i \ / \ h_t(x_i) = y_i} D_t(i) e^{-\alpha_t} \\ &= \varepsilon_t e^{\alpha_t} + (1-\varepsilon_t) e^{-\alpha_t} \\ &= 2\sqrt{\varepsilon_t(1-\varepsilon_t)} \end{aligned}$$

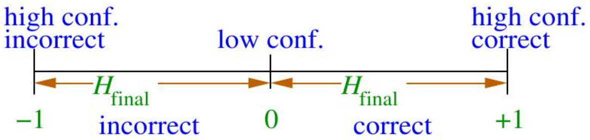
Training error $\neq$ test error ! (overfitting ?)

Observed

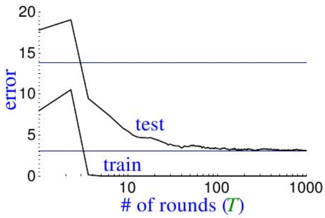


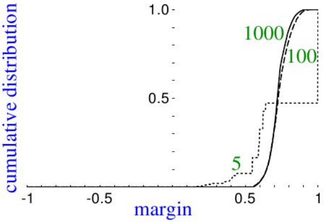
Why ?

- Explanation based on the margin



The margin





	# rounds		
	5	100	1000
train error	0.0	0.0	0.0
test error	8.4	3.3	3.1
% margins ≤ 0.5	7.7	0.0	0.0
minimum margin	0.14	0.52	0.55

Analysis

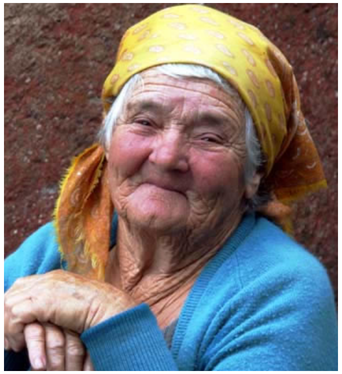
1. Boosting $\Rightarrow$ larger margin
2. Larger margin $\Rightarrow$ lower generalization error
   Why: if margin is large, hypothesis can be approximated by a simple one.

Intuition about Margin

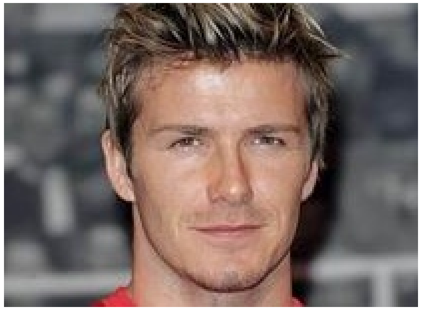
Infant

?

Elderly



Man



?

Woman

Partial conclusion

Adaboost is:

- a way of boosting a weak learner
- a margin optimizer
- (other interpretations related to the minimization of an exponential loss)

However

- ... if Adaboost minimizes a criterion, it would make sense to directly minimize this criterion...
- but direct minimization *degrades* performances....

Main weakness: sample noise

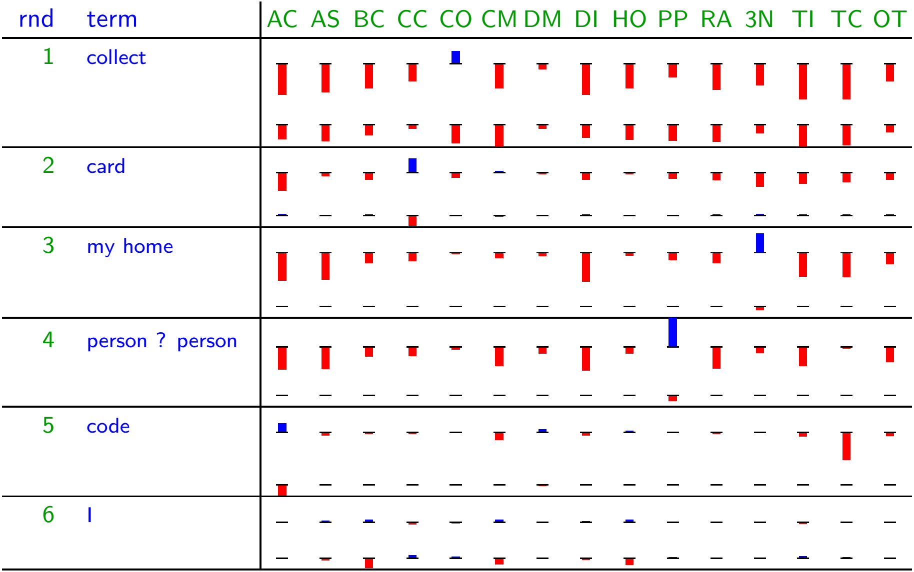
- Noisy examples are rewarded.

Application: Boosting for Text Categorization

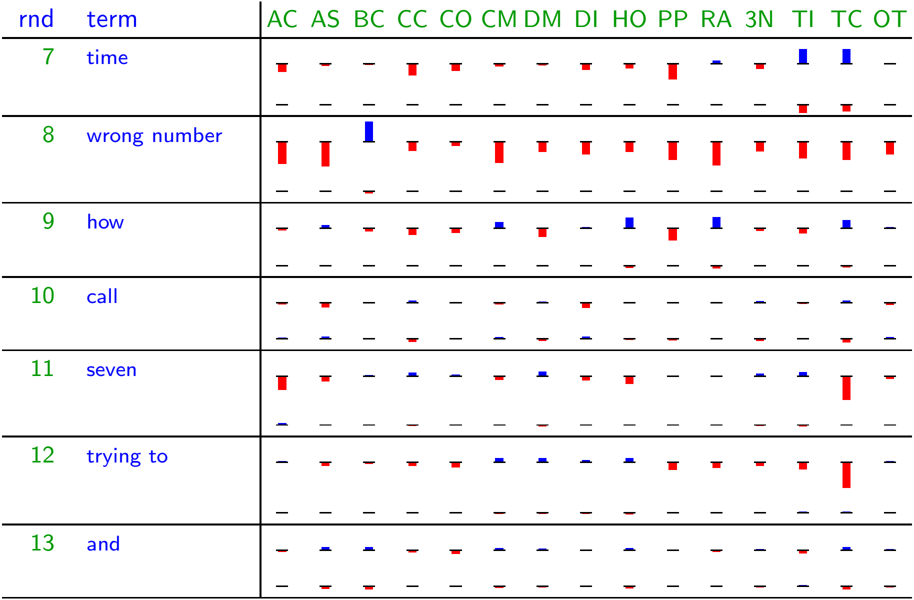
[with Singer]

- weak classifiers: very simple weak classifiers that test on simple patterns, namely, (sparse) *n*-grams
  - find parameter α_t and rule h_t of given form which minimize Z_t
  - use efficiently implemented exhaustive search
- "How may I help you" data:
  - 7844 training examples
  - 1000 test examples
  - categories: AreaCode, AttService, BillingCredit, CallingCard, Collect, Competitor, DialForMe, Directory, HowToDial, PersonToPerson, Rate, ThirdNumber, Time, TimeCharge, Other.

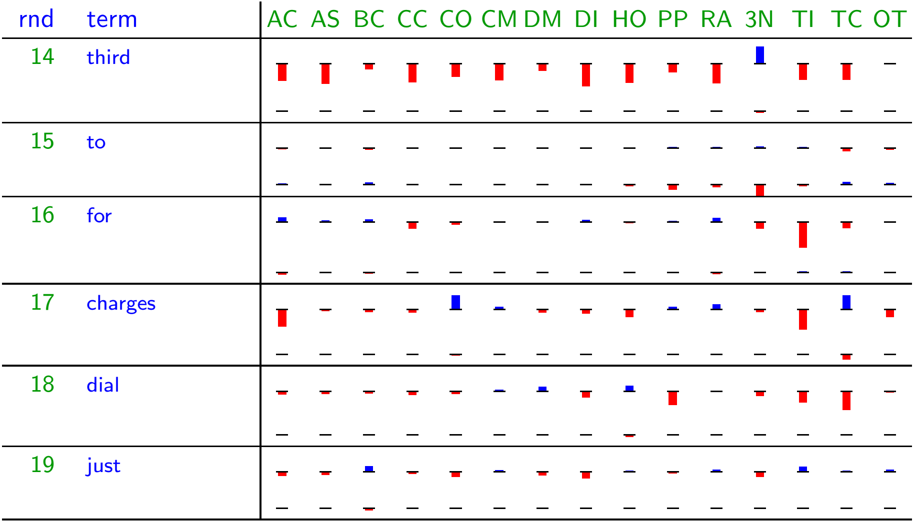
Weak Classifiers



More Weak Classifiers



More Weak Classifiers



Finding Outliers

examples with most weight are often outliers (mislabeled and/or ambiguous)

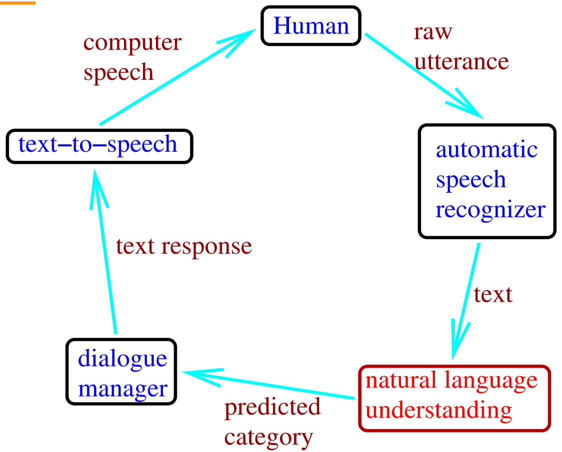
- `I'm trying to make a credit card call` (Collect)
- `hello` (Rate)
- `yes I'd like to make a long distance collect call please` (CallingCard)
- `calling card please` (Collect)
- `yeah I'd like to use my calling card number` (Collect)
- `can I get a collect call` (CallingCard)
- `yes I would like to make a long distant telephone call and have the charges billed to another number` (CallingCard DialForMe)
- `yeah I can not stand it this morning I did oversea call is so bad` (BillingCredit)
- `yeah special offers going on for long distance` (AttService Rate)
- `mister allen please william allen` (PersonToPerson)
- `yes ma'am I I'm trying to make a long distance call to a non dialable point in san miguel philippines` (AttService Other)

Application: Human-computer Spoken Dialogue

[with Rahim, Di Fabbrizio, Dutton, Gupta, Hollister & Riccardi]

- application: automatic "store front" or "help desk" for AT&T Labs' Natural Voices business
- caller can request demo, pricing information, technical support, sales agent, etc.
- interactive dialogue

How It Works



- NLU's job: classify caller utterances into 24 categories (demo, sales rep, pricing info, yes, no, etc.)
- weak classifiers: test for presence of word or phrase

Overview

Introduction

Boosting
- PAC Learning
- Boosting
- Adaboost

Bagging
- General bagging
- Random Forests

Bagging

Breiman96

Enforcing diversity through bootstrap: iterate

- Draw

$$\mathcal{E}_t = n \text{ examples uniformly drawn with replacement from } \mathcal{E}$$

- Learn h_t from $\mathcal{E}_t$

Finally:

$$H(x) = \text{ Vote } (\{h_t(x)\})$$

Bagging vs Boosting

Bagging: h_t independent — parallelisation is possible

Boosting: h_t depends from the previous hypotheses
(h_t covers up $h_1 \ldots h_{t-1}$ mistakes).

Visualization — Dietterich Margineantu 97
In the 2d plane: distance, error
Boosting — Bagging

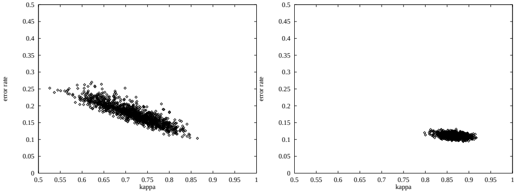


Figure 1: Kappa-Error diagrams for AdaBoost (left) and bagging (right) on the Expf domain.

Analysis

Assume

- $\mathcal{E}$ drawn after distribution P
- $\mathcal{E}_t$ uniformly sampled from $\mathcal{E}$, h_t learned from $\mathcal{E}_t$
- Error of H, average of the h_t:

$$H(x) = \mathbb{E}_{\mathcal{E}_t}[h_t(x)]$$

Error

- Direct error:

$$e = \mathbb{E}_{\mathcal{E}}\mathbb{E}_{X,Y}[(Y - h(X))^2]$$

- Bagging error

$$e_B = \mathbb{E}_{X,Y}[(Y - H(X))^2]$$

- Rewriting e:

$$e = \mathbb{E}_{X,Y}[Y^2] - 2\mathbb{E}_{X,Y}[YH] + \mathbb{E}_{X,Y}\mathbb{E}_{\mathcal{E}}[h^2]$$

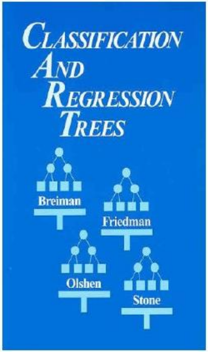
and with Jensen inequality, $\mathbb{E}[Z^2] \geq \mathbb{E}[Z]^2$

$$e \geq e_B$$

Overview

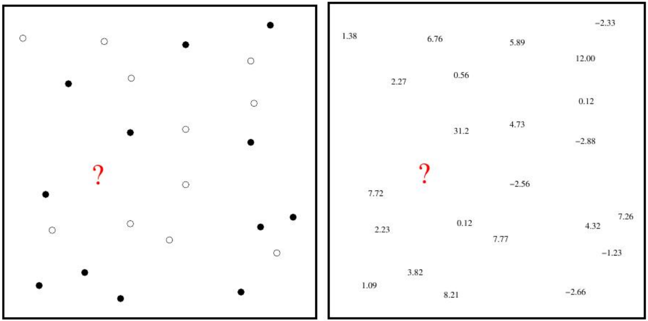
Random Forests

Breiman 00

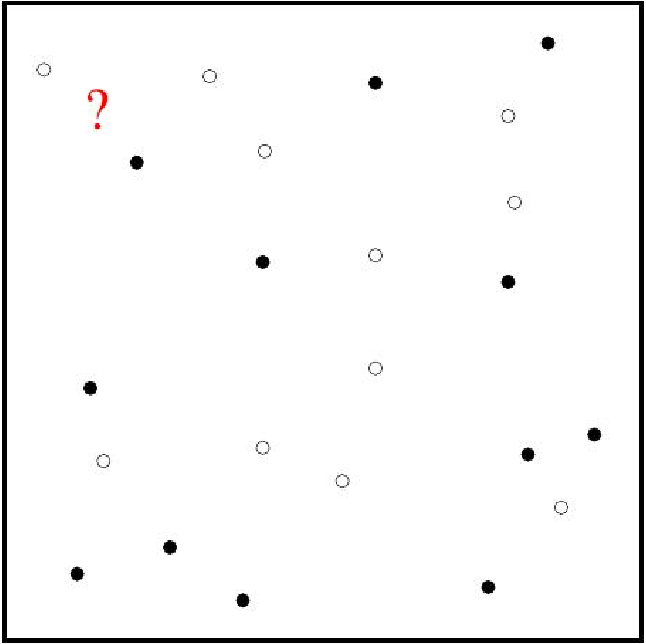


http://videolectures.net/sip08_biau_corfao/

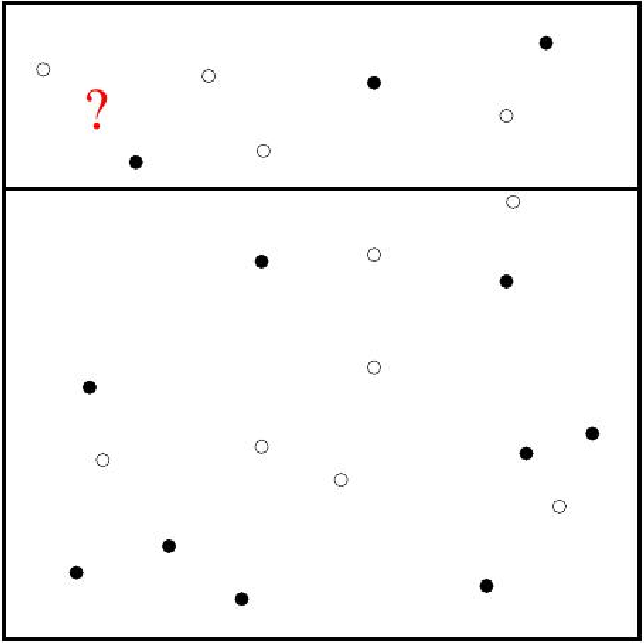
Classification / Regression



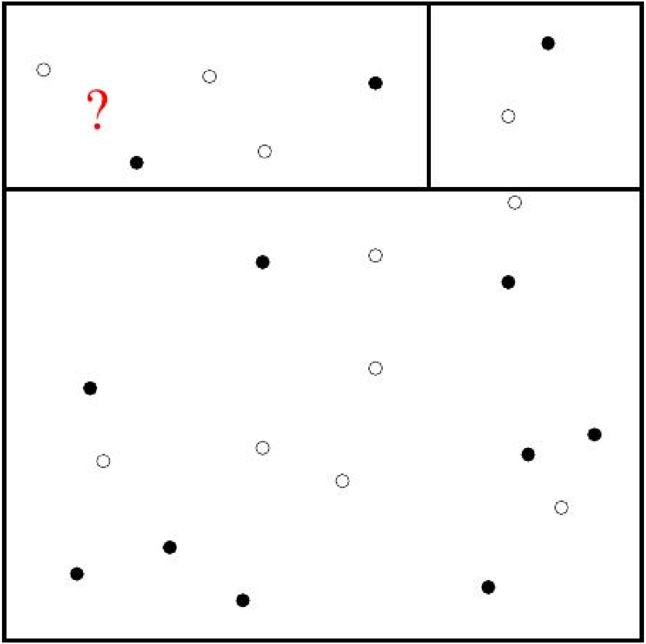
Splitting space



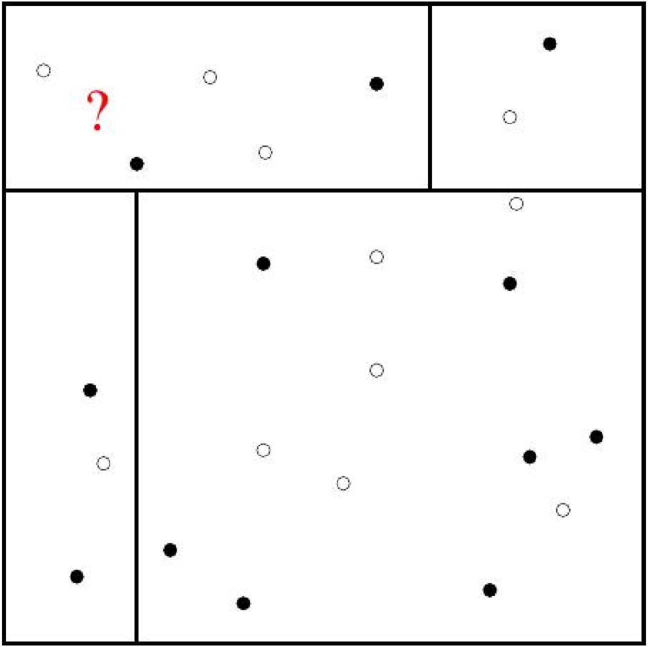
Splitting space



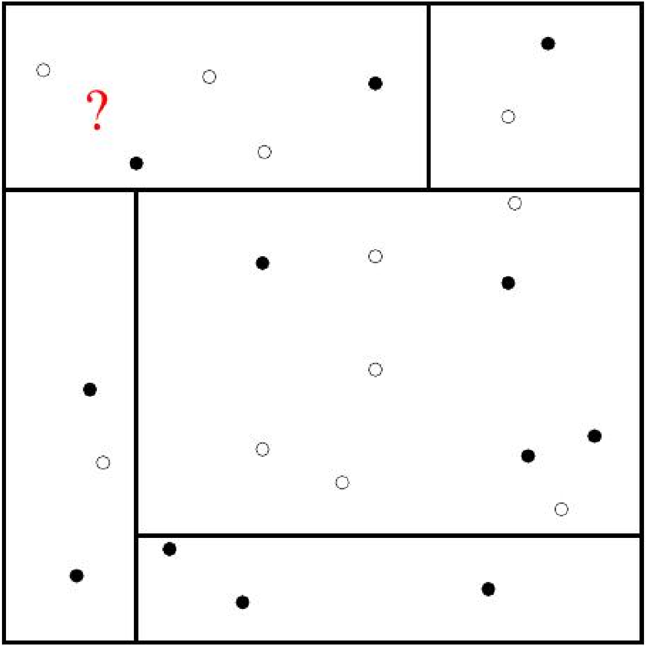
Splitting space



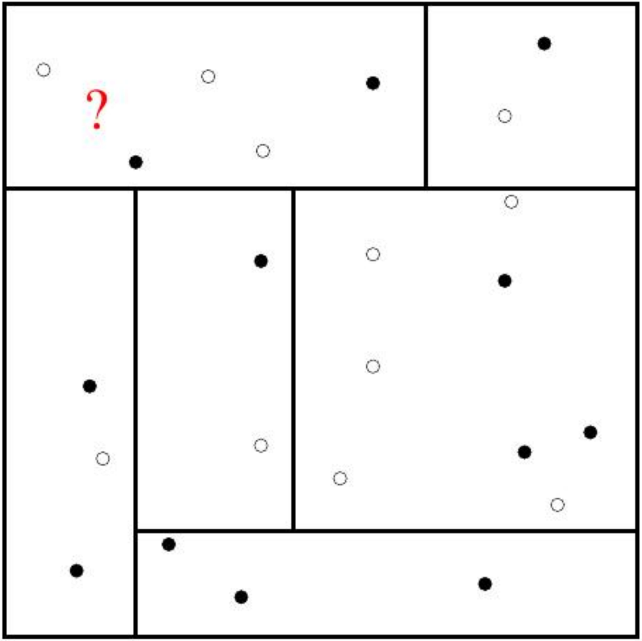
Splitting space



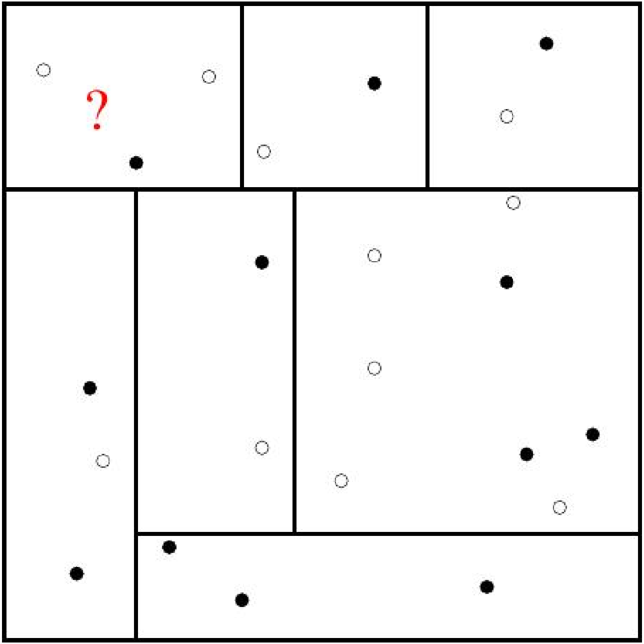
Splitting space



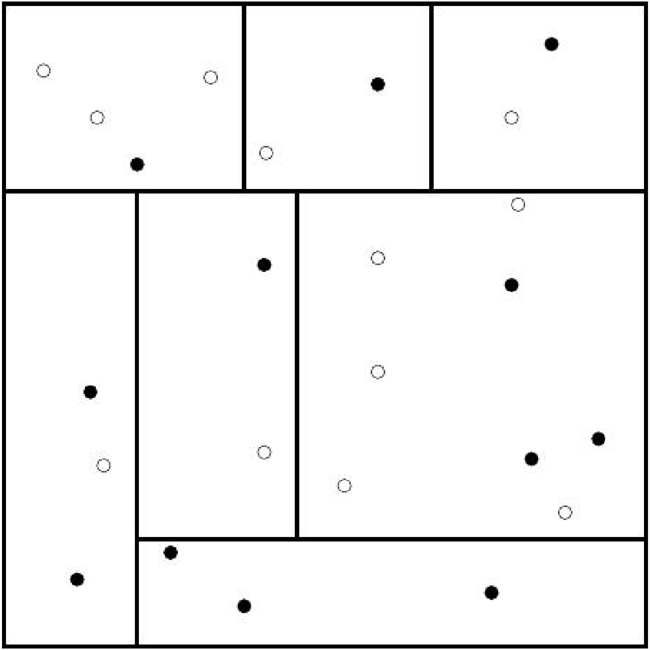
Splitting space



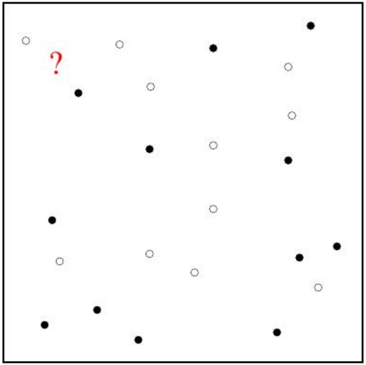
Splitting space



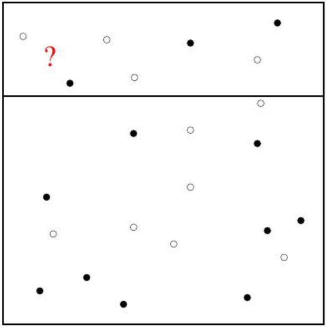
Splitting space



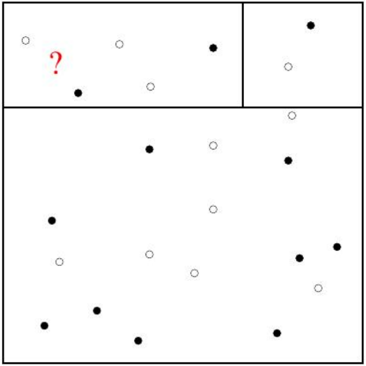
Build a tree

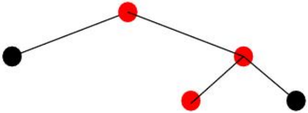


Build a tree

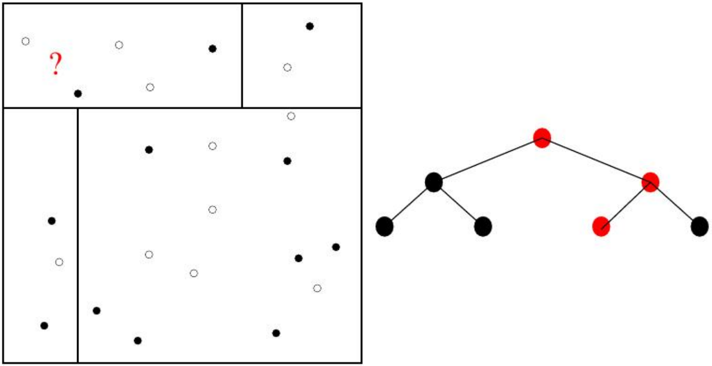


Build a tree

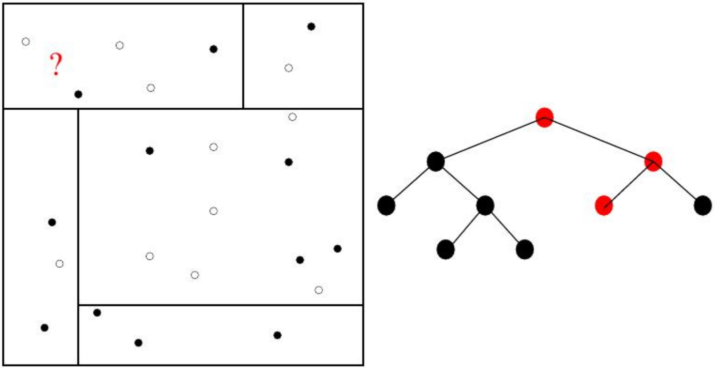




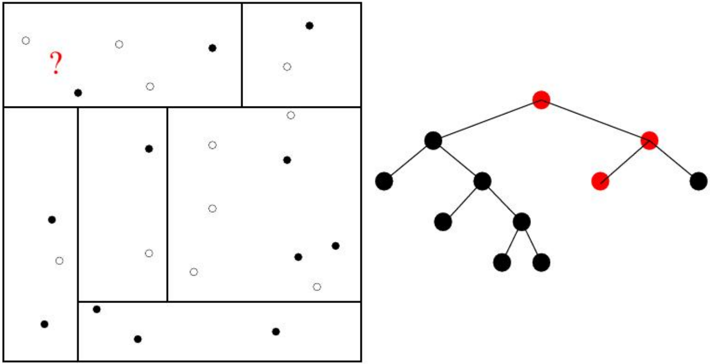
Build a tree



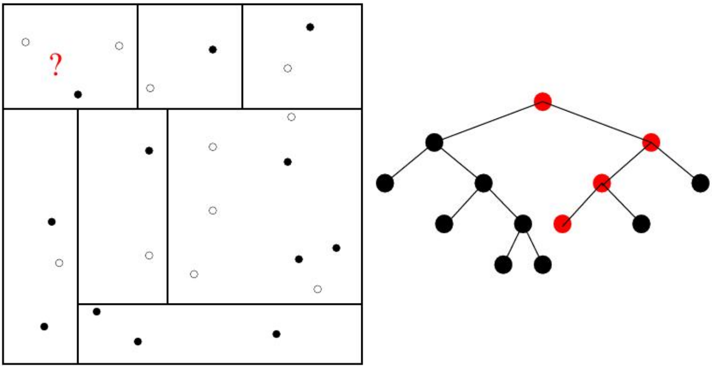
Build a tree



Build a tree



Build a tree



Random forests

Breiman00 stat.berkeley.edu/users/breiman/RandomForests

Principle

- Randomized trees
- Average

Properties

- Fast, easy to implement
- Excellent accuracy
- No overfitting % number of features

Theoretical analysis difficult

KDD 2009 – Orange

Targets

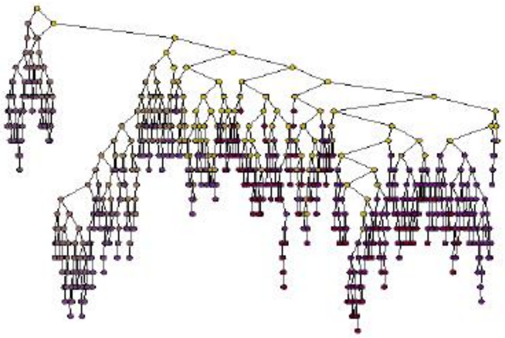
1. Churn
2. Appetency
3. Up-selling

Core Techniques

1. Feature Selection
2. Bounded Resources
3. Parameterless methods

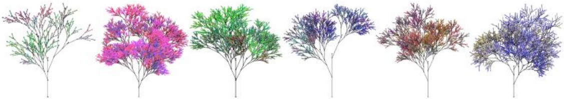
Random tree

- In each node, uniformly select a subset of attribute
- Compute the best one
- Until reaching max depth



Tree aggregation

- h_t: random tree
- Fast and straightforward parallelization



Analysis

Biau et al. 10

$$\mathcal{E} = \{(x_i, y_i)\},\ x_i \in\ [0,1]^d,\ y_i\ \in\ \mathbb{R}, i = 1..n\}\ (x_i, y_i) \sim P(x, y)$$

Goal: estimate

$$r(x) = \mathbb{E}[Y|X = x]$$

Criterion consistency

$$\mathbb{E}[(r_n(X) - r(X))^2]$$

Simplified algorithm

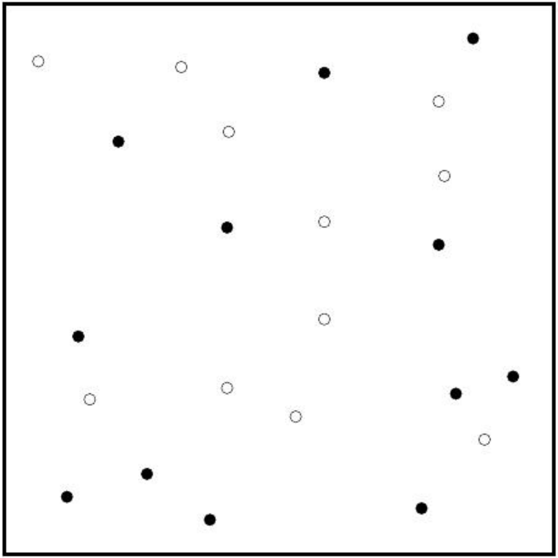
Set $k_n \geq 2$, iterate $\log_2 k_n$ fois:

- Select in each node a feature s.t.

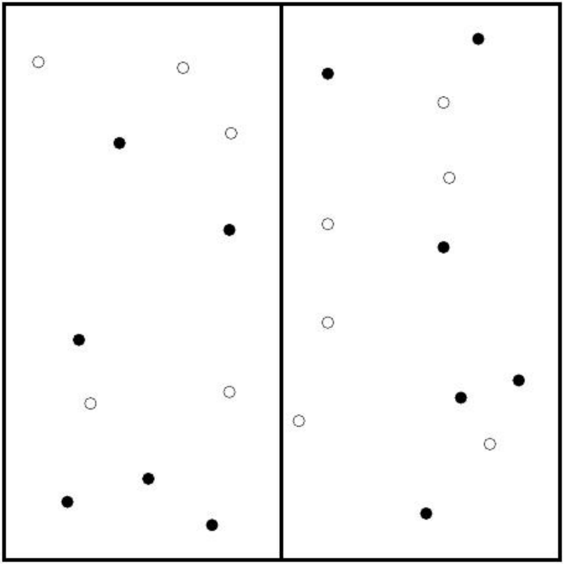
$$\Pr(\text{feature. j selected}) = p_{n,j}$$

- Split: feature j $<$ its median

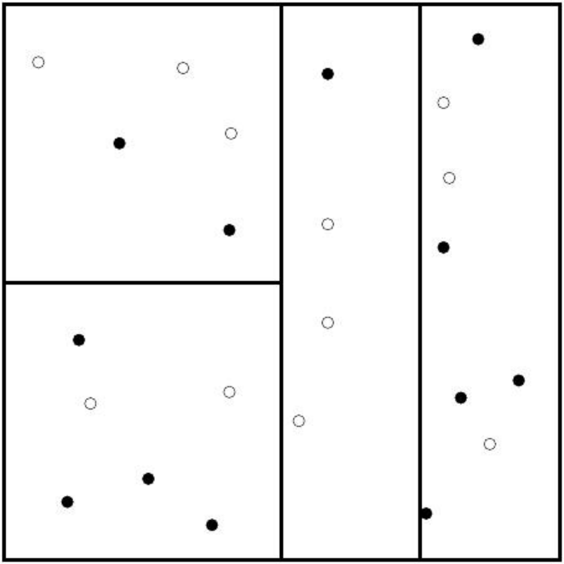
Simplified algorithm



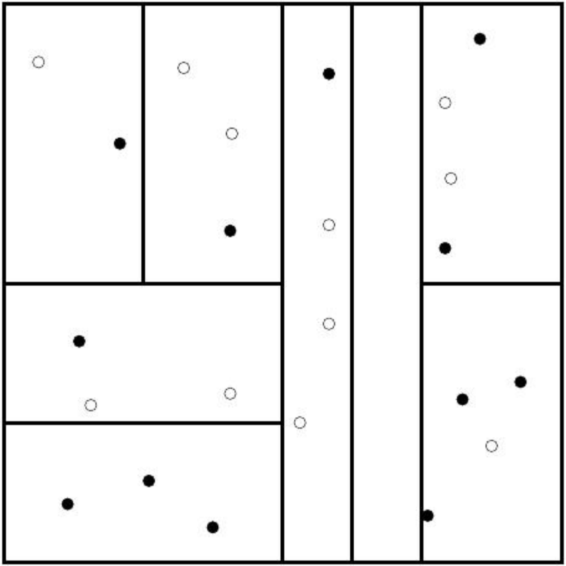
Simplified algorithm



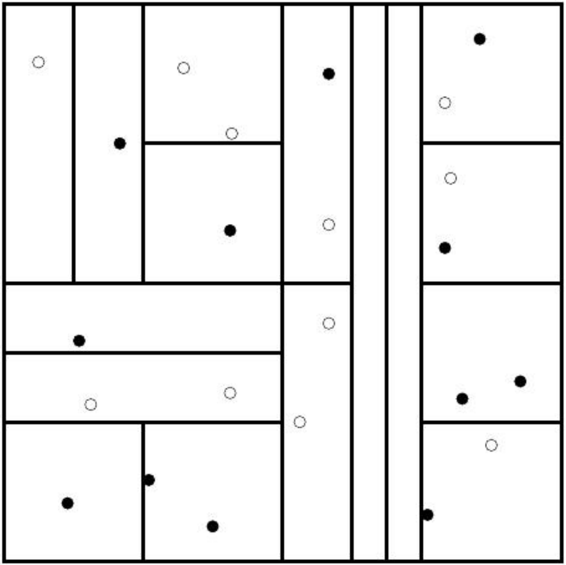
Simplified algorithm



Simplified algorithm



Simplified algorithm



Simplified algorithm

Analysis

- Each tree has $2^{log_2 k_n} = k_n$ leaves
- Each leaf covers $2^{-\lfloor log_2 k_n \rfloor} = 1/k_n$ volume
- Assuming x_i uniformly drawn in $[0,1]^d$, number of examples per leaf is $\approx \frac{n}{k_n}$
- If $k_n = n$, very few examples per leaf

$$r_n(x) = \mathbb{E}\left[\frac{\sum_i y_i \mathbf{1}_{x_i,x \text{ in same leaf}}}{\sum_i \mathbf{1}_{x_i,x \text{ in same leaf}}}\right]$$

Consistency

Thm

r_n is consistent if $p_{nj} \log k_n \to \infty$ forall j and $k_n/n \to 0$ when $n \to \infty$.