[70240413 Statistical Machine Learning, Spring, 2018]

Ensemble Methods

Jun Zhu
dcszj@mail.tsinghua.edu.cn
http://bigml.cs.tsinghua.edu.cn/~jun
State Key Lab of Intelligent Technology & Systems
Tsinghua University

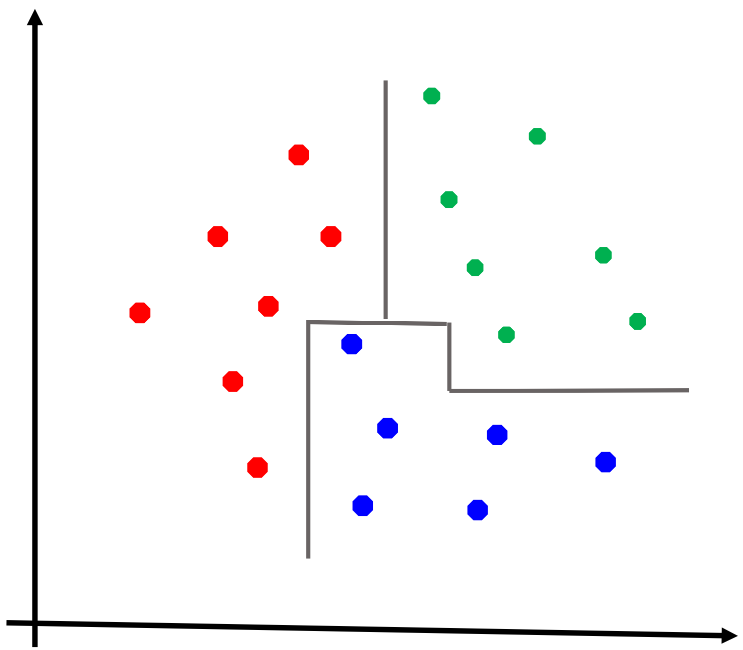
March 20, 2018

Trees, Bagging, Random Forests, Boosting

- Classification trees
- Bagging: Averaging Trees
- Random Forests: Cleverer Averaging of Trees
- Boosting: Cleverest Averaging of Trees

- Methods for improving the performance of weak learners such as Trees. Classification trees are adaptive and robust, but do not generalize well. The techniques discussed here enhance their performance considerably.

Classification

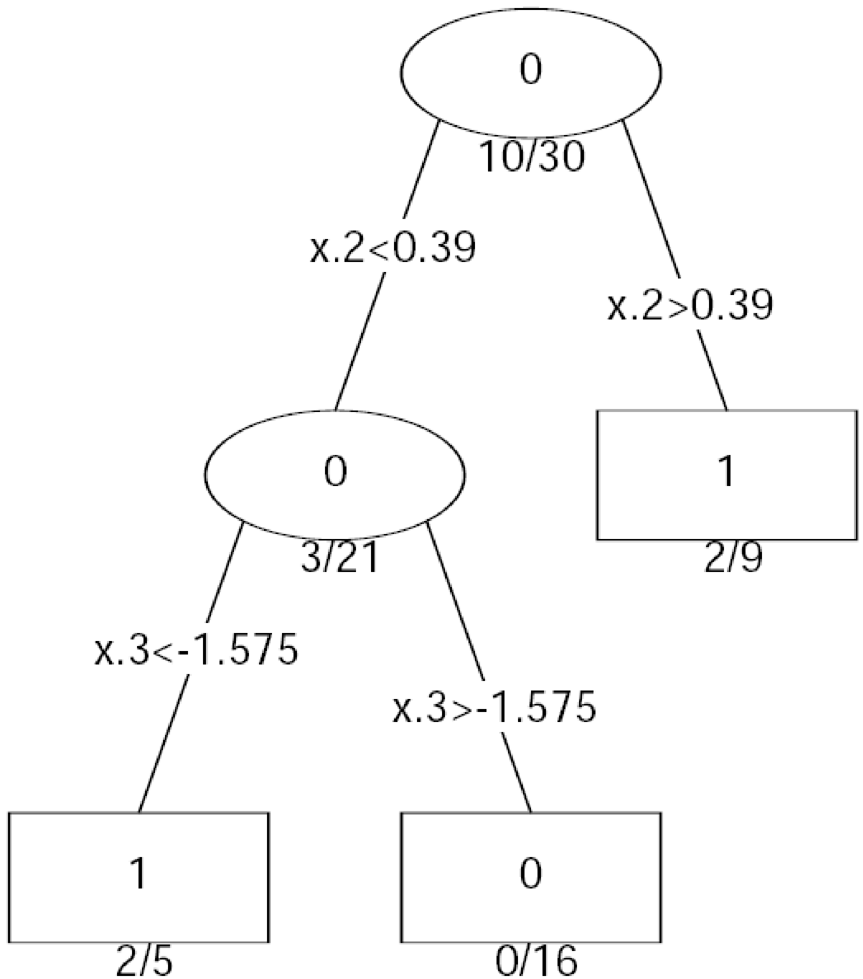


- Feature vector $X = (X1, X2, \ldots, Xp)$
- We hope to build a classification rule $C(X)$ to assign a class label to an individual with feature X.
  - A classifier => a partition of the sample space
  - However, find a general partition is hard if no assumptions

Classification Trees

- Represented by a series of binary splits.
- Each internal node represents a value query on one of the variables — e.g. "*Is* $X3 > 0.4$". If the answer is "Yes", go right, else go left.
- The terminal nodes are the decision nodes. Typically each terminal node is dominated by one of the classes.
- The tree is grown using training data, by recursive splitting.
- The tree is often pruned to an optimal size, evaluated by cross-validation.
- New observations are classified by passing their X down to a terminal node of the tree, and then using majority vote.

Classification Tree

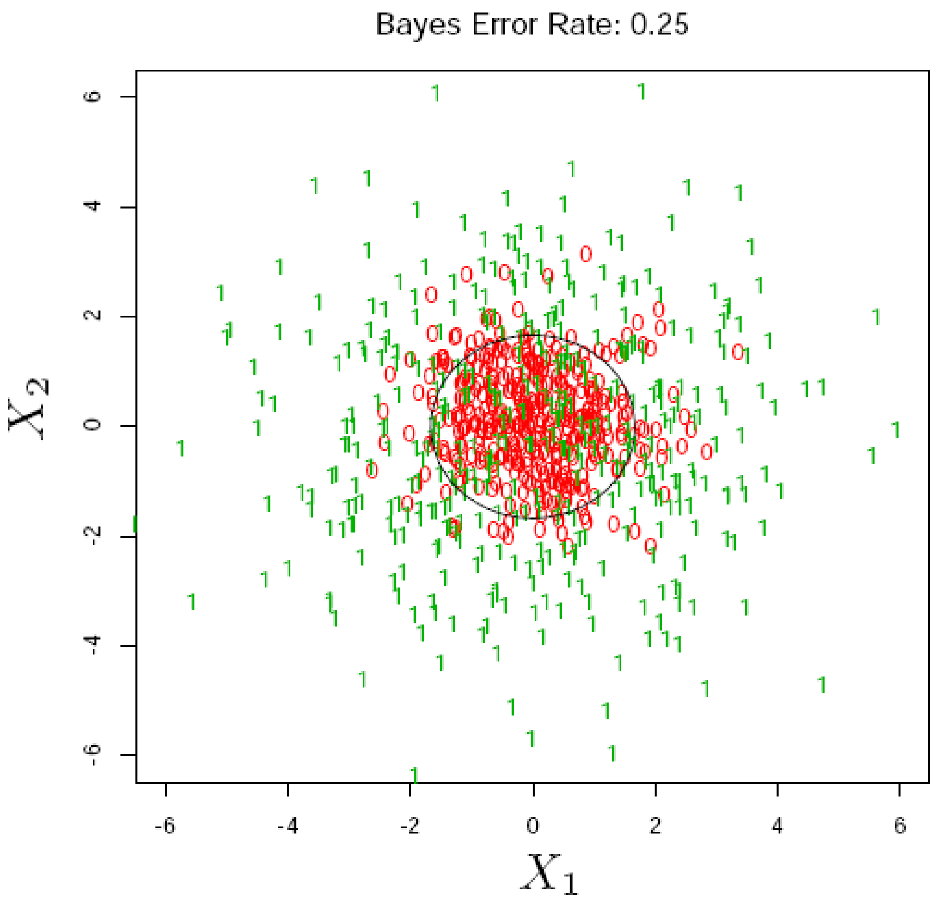


Properties of Trees

- Pros:
  - Can handle huge datasets
  - Can handle mixed predictors---quantitative and qualitative
  - Easily ignore redundant variables
  - Handle missing data elegantly
  - Small trees are easy to interpret

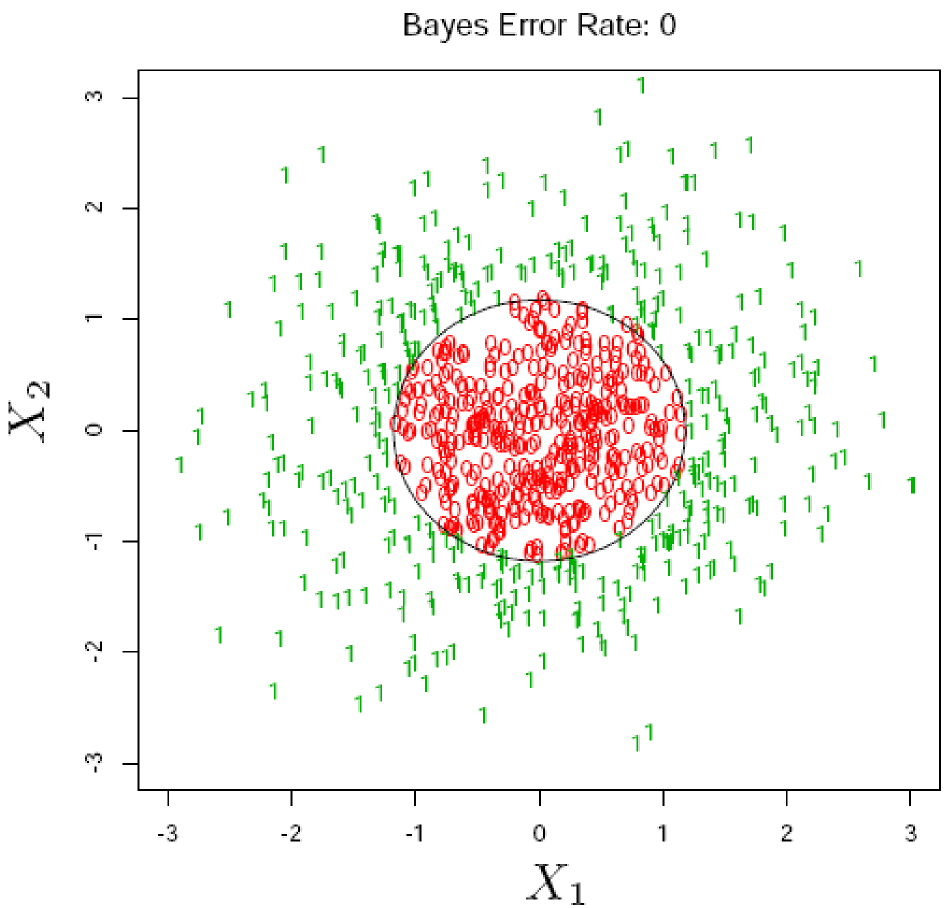
- Cons:
  - Large trees are hard to interpret
  - Instable due to the hierarchical nature --- error at a top level is propagated to all of the splits below it
  - Often prediction performance is poor

Toy Classification Problem



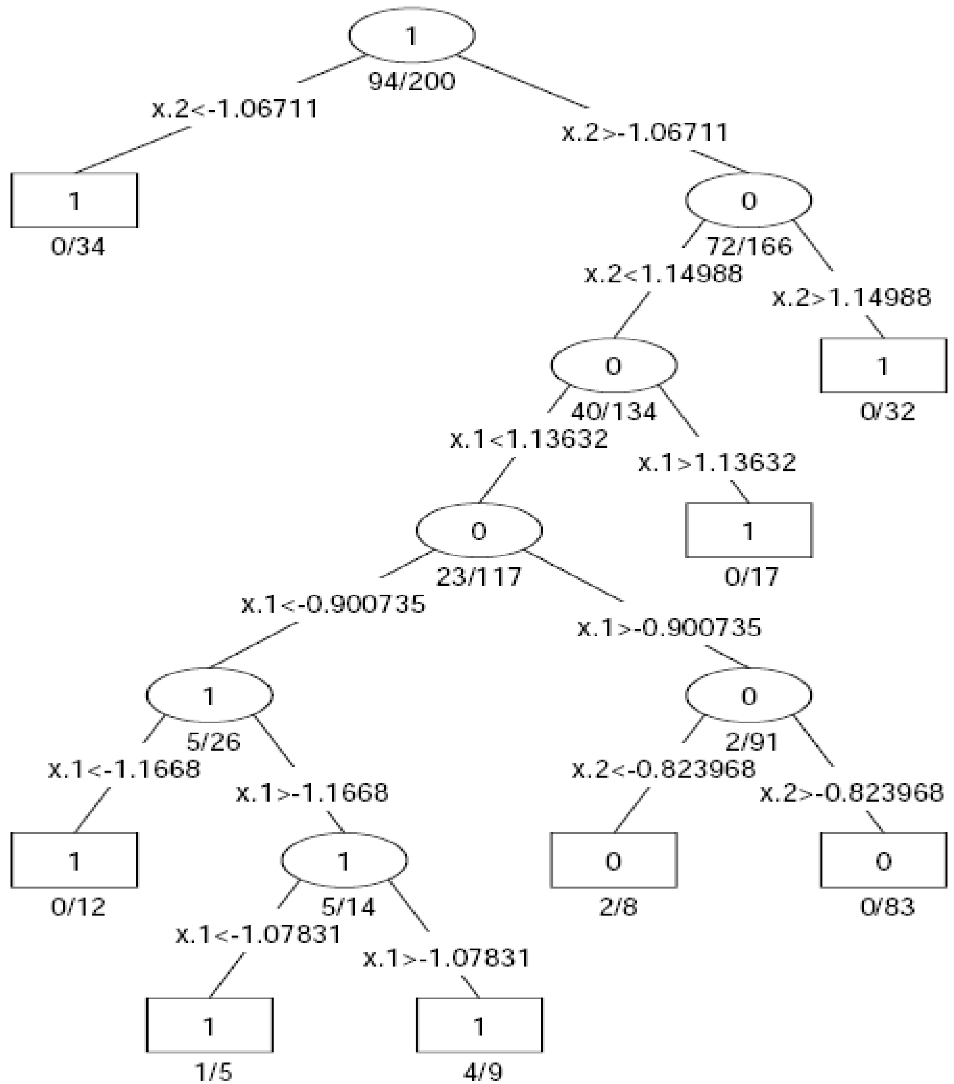
- Data X and Y, with Y taking values $+1$ or -1.
- Here $X = (X_1, X_2)$
- The black boundary is the Bayes Decision Boundary - the best one can do.
- Goal: Given N training pairs (X_i, Y_i) produce a classifier $\hat{C}(X) \in \{-1, 1\}$
- Also estimate the probability of the class labels $P(Y = +1|X)$.

Toy Classification Problem

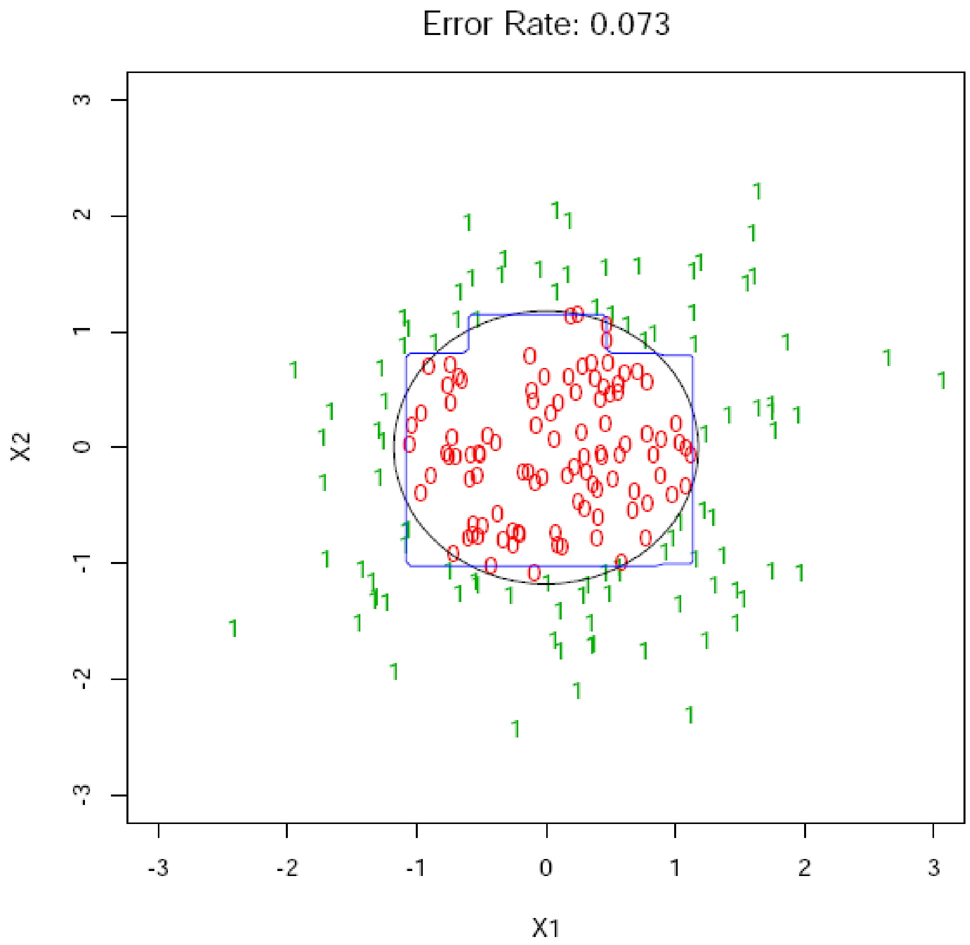


- Deterministic problem; noise comes from sampling distribution of X.
- Use a training sample of size 200.
- Here Bayes Error is 0%.

Classification Tree



Decision Boundary: Tree



When the nested spheres are in 10-dimensions, Classification Trees produces a rather noisy and inaccurate rule $\hat{C}(X)$, with error rates around 30%.

Model Averaging

- Classification trees can be simple, but often produce noisy (bushy) or weak (stunted) classifiers.
  - Bagging (Breiman, 1996): Fit many large trees to bootstrap-resampled versions of the training data, and classify by majority vote.
  - Boosting (Freund & Shapire, 1996): Fit many large or small trees to reweighted versions of the training data. Classify by weighted majority vote.
  - Random Forests (Breiman 1999): Fancier version of bagging.

- In general, Boosting > Random Forests > Bagging > Single Tree.

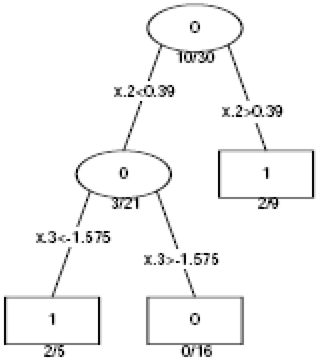
Bagging

- Bagging or bootstrap aggregation averages a given procedure over many samples, to reduce its variance — a poor man's Bayes.
  - See Chap. 8 of ESLII for relation between bagging and Bayes
- Suppose $C(S, x)$ is a classifier, such as a tree, based on our training data S, producing a predicted class label at input point x.
- To bag C, we draw bootstrap samples $\mathcal{S}^{*1}, \ldots \mathcal{S}^{*B}$ each of size N with replacement from the training data. Then

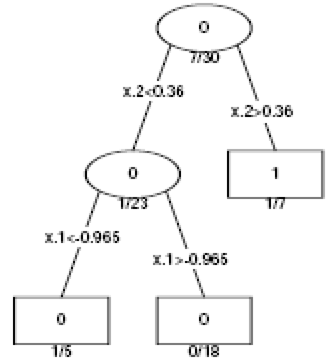
$$\hat{C}_{bag}(x) = \text{Majority Vote } \{C(\mathcal{S}^{*b}, x)\}_{b=1}^{B}.$$

- Bagging can dramatically reduce the variance of unstable procedures (like trees), leading to improved prediction. However any simple structure in C (e.g, a tree) is lost.

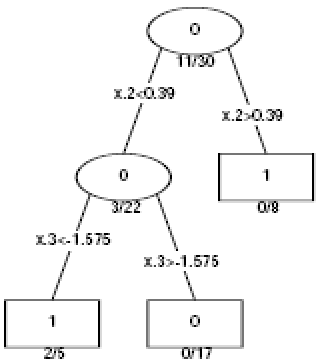
Original Tree



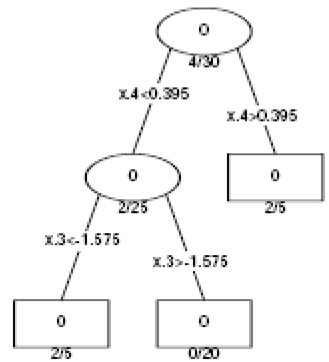
Bootstrap Tree 1



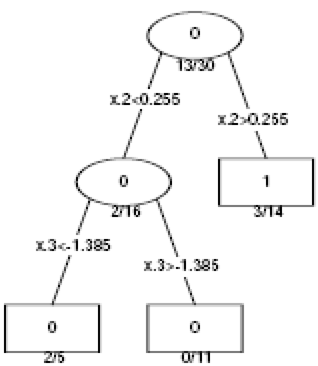
Bootstrap Tree 2



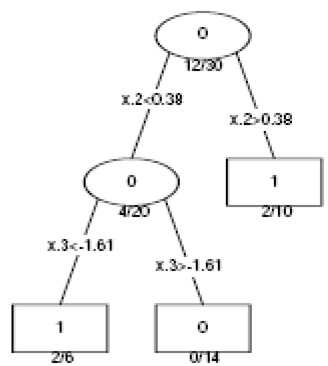
Bootstrap Tree 3



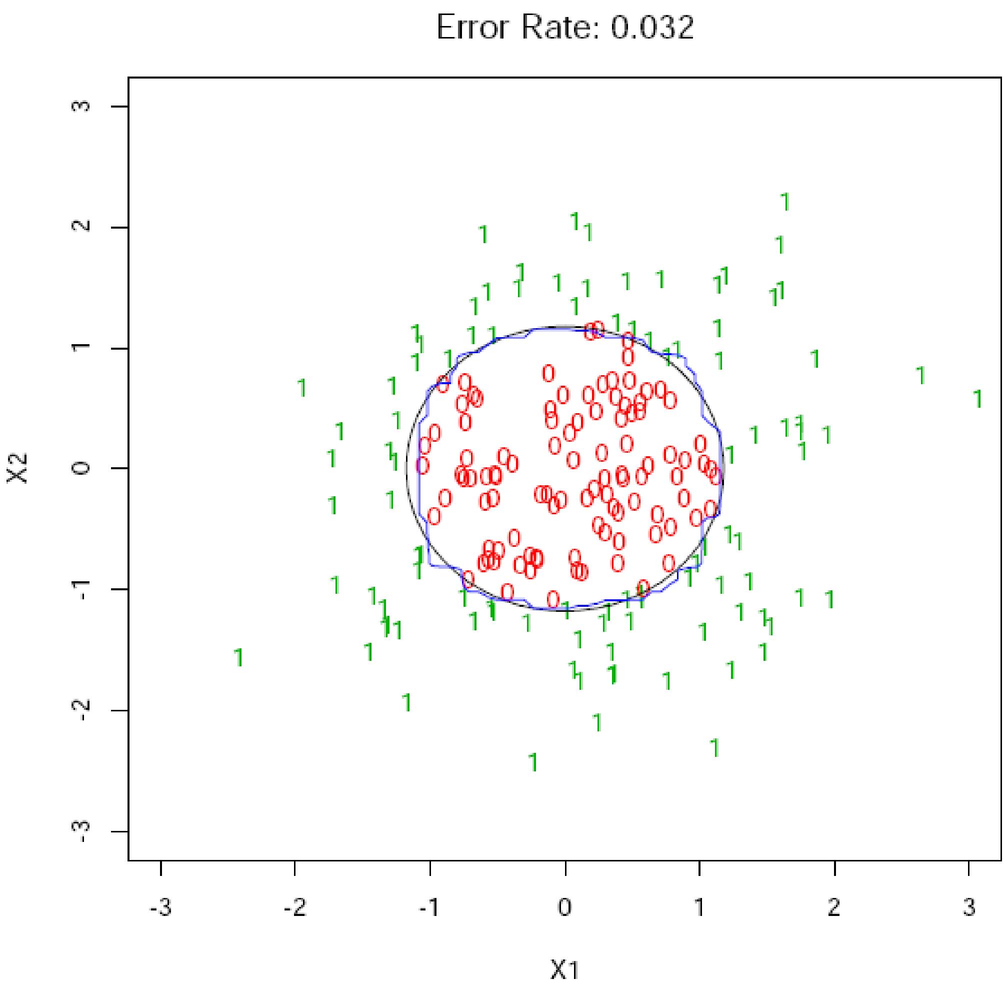
Bootstrap Tree 4



Bootstrap Tree 5



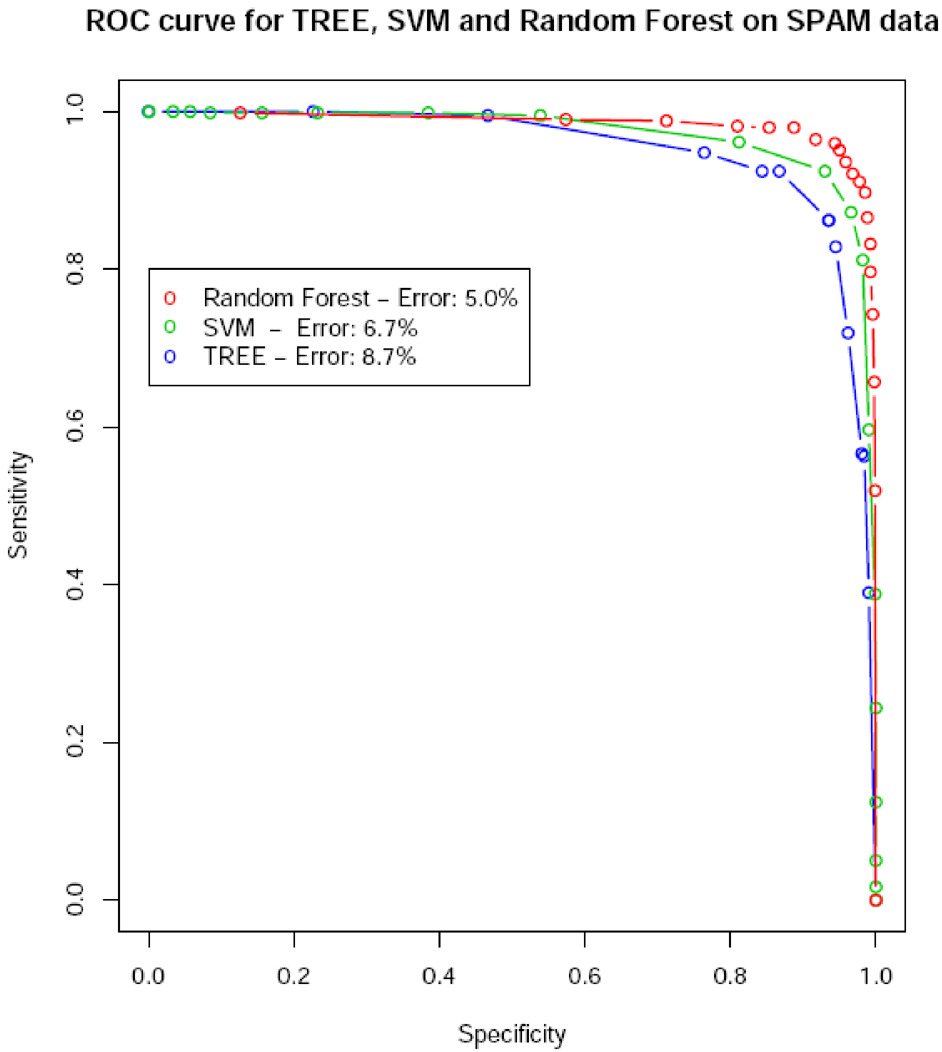
Decision Boundary: Bagging



Bagging averages many trees, and produces smoother decision boundaries.

Random Forests

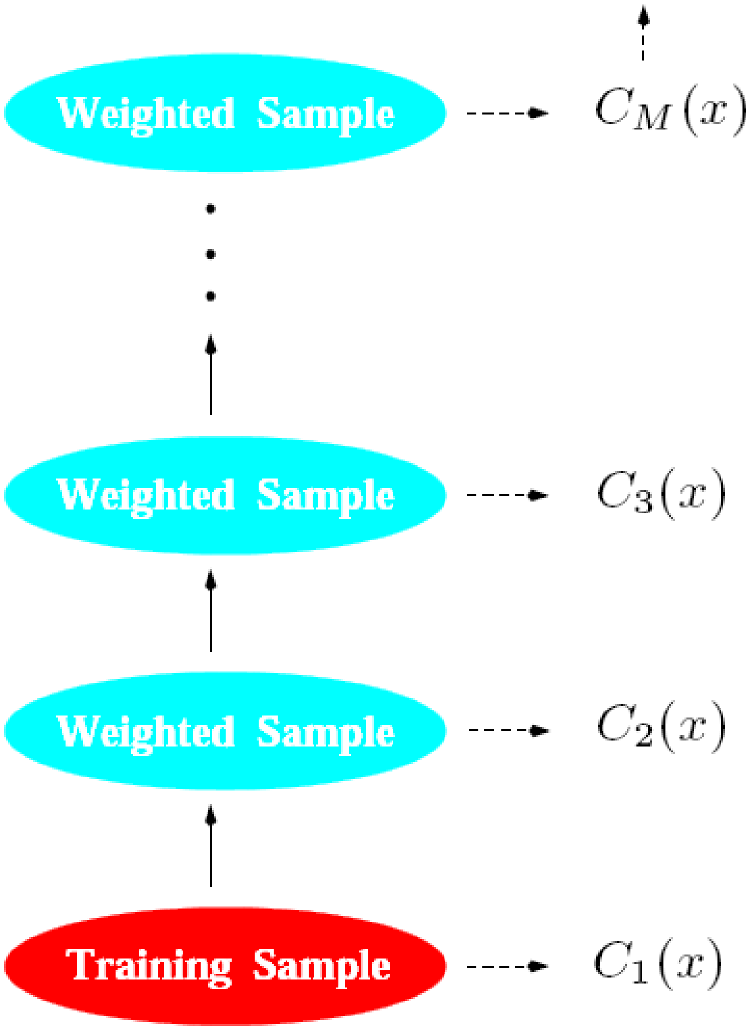
- refinement of bagged trees; quite popular
- at each tree split, a random sample of m features is drawn, and only those m features are considered for splitting. Typically $m = \sqrt{p}$ or $\log_2 p$, where p is the number of features
- For each tree grown on a bootstrap sample, the error rate for observations left out of the bootstrap sample is monitored. This is called the "out-of-bag" error rate.
- random forests tries to improve on bagging by "de-correlating" the trees. Each tree has the same expectation.



TREE, SVM and RF

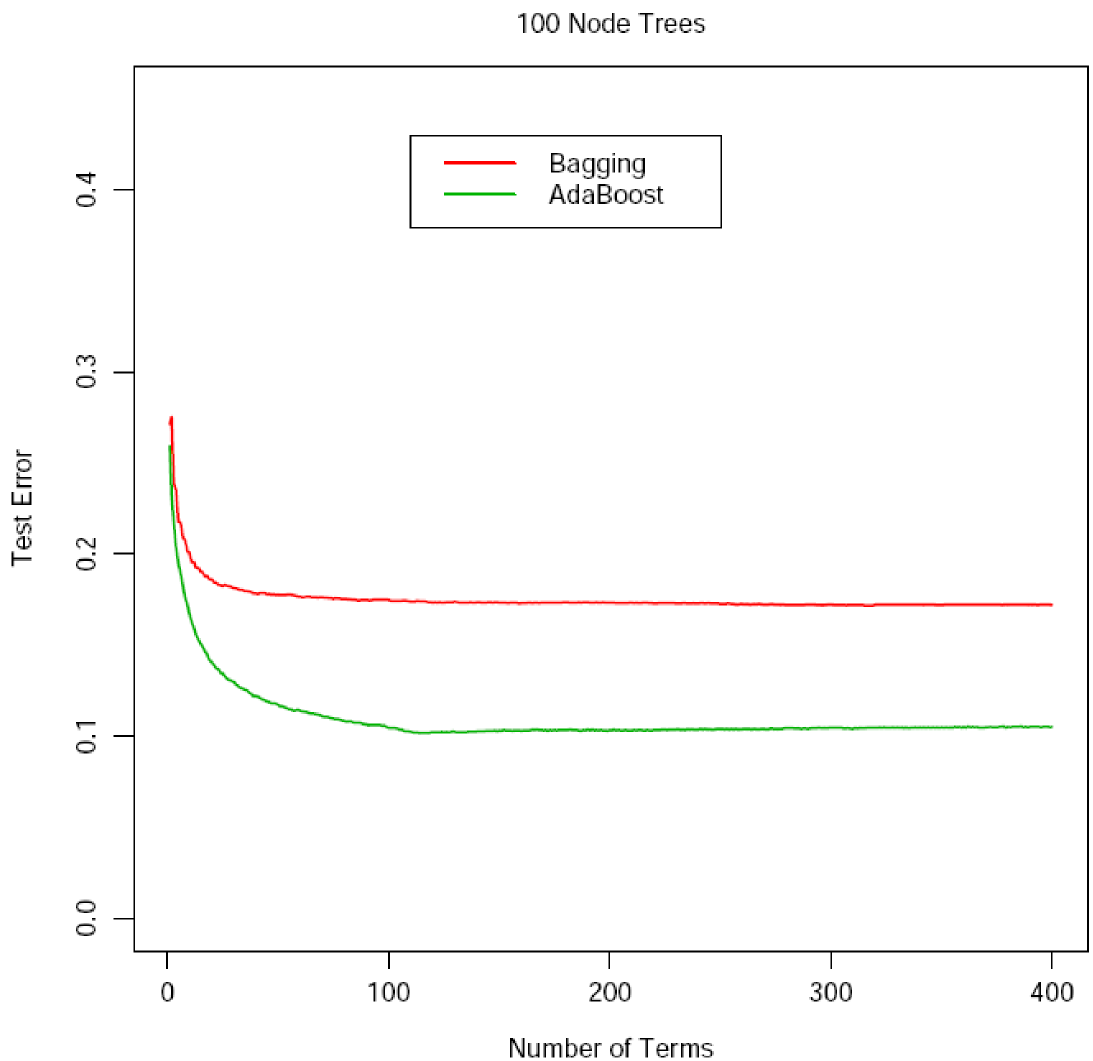
Random Forest dominates both other methods on the SPAM data — 5.0% error. Used 500 trees with default settings for `random Forest` package in `R`.

Boosting



- Average many trees, each grown to re-weighted versions of the training data.
- Final Classifier is weighted average of classifiers:

$$C(x) = \text{sign}\left[\sum_{m=1}^{M} \alpha_m C_m(x)\right]$$

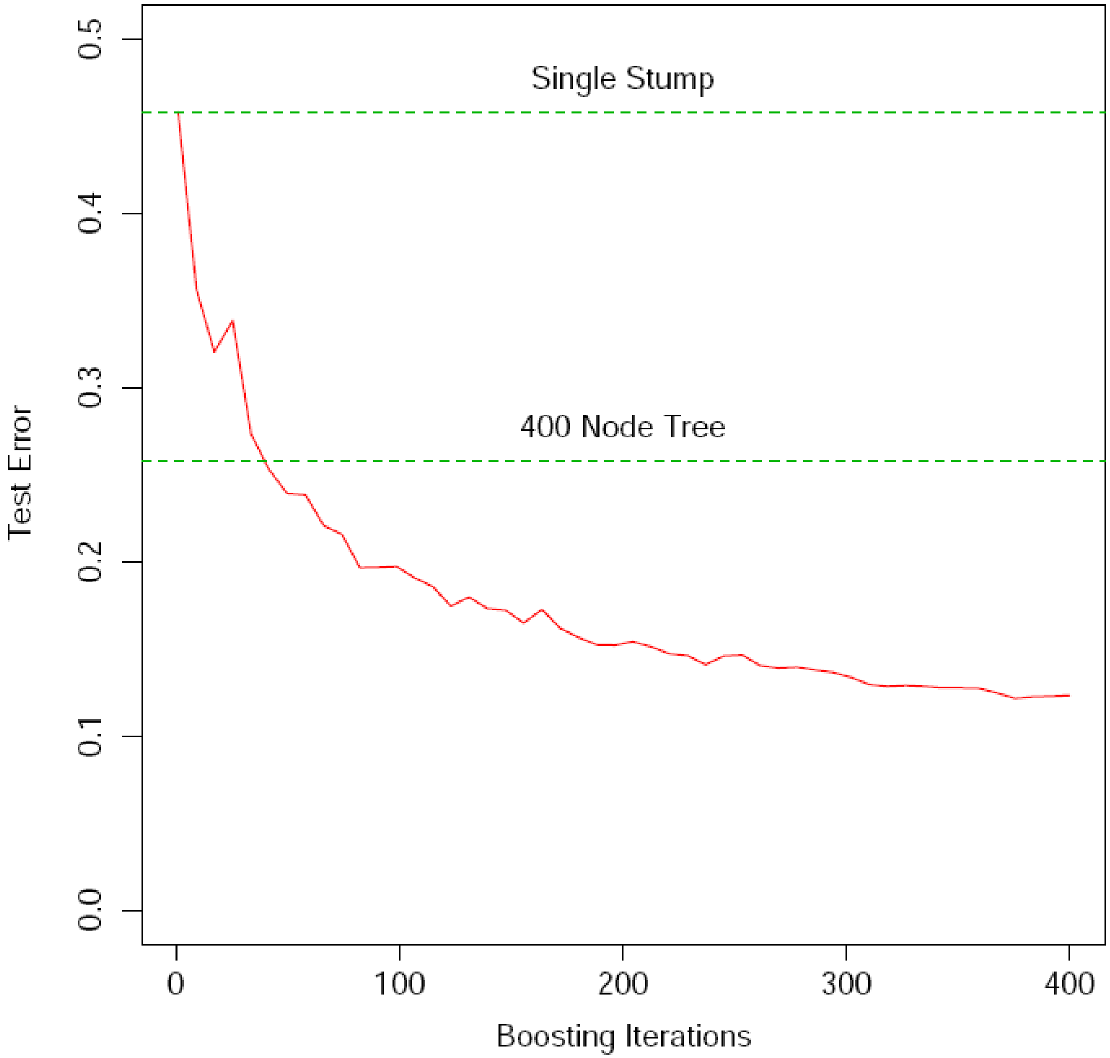


Boosting vs Bagging

- 2000 points from Nested Spheres in R^{10}
- Bayes error rate is 0%.
- Trees are grown best first without pruning.
- Leftmost term is a single tree.

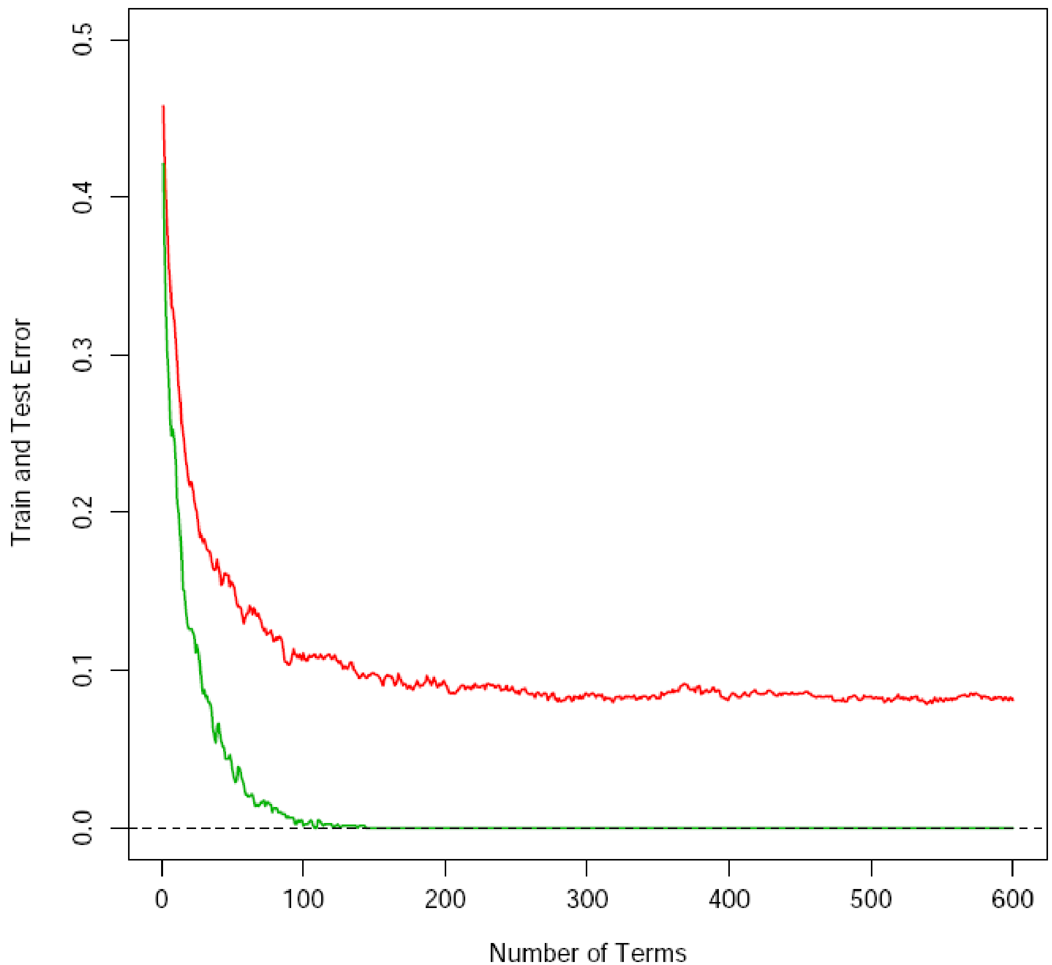
AdaBoost (Freund & Schapire, 1996)

1. Initialize the observation weights $w_i = 1/N,\ i = 1, 2, \ldots, N$.
2. For $m = 1$ to M repeat steps (a)–(d):
   (a) Fit a classifier $C_m(x)$ to the training data using weights w_i.
   (b) Compute weighted error of newest tree
   $$\mathrm{err}_m = \frac{\sum_{i=1}^{N} w_i I(y_i \neq C_m(x_i))}{\sum_{i=1}^{N} w_i}.$$
   (c) Compute $\alpha_m = \log[(1 - \mathrm{err}_m)/\mathrm{err}_m]$.
   (d) Update weights for $i = 1, \ldots, N$:
   $w_i \leftarrow w_i \cdot \exp[\alpha_m \cdot I(y_i \neq C_m(x_i))]$
   and renormalize to w_i to sum to 1.
3. Output $C(x) = \mathrm{sign}\left[\sum_{m=1}^{M} \alpha_m C_m(x)\right]$.



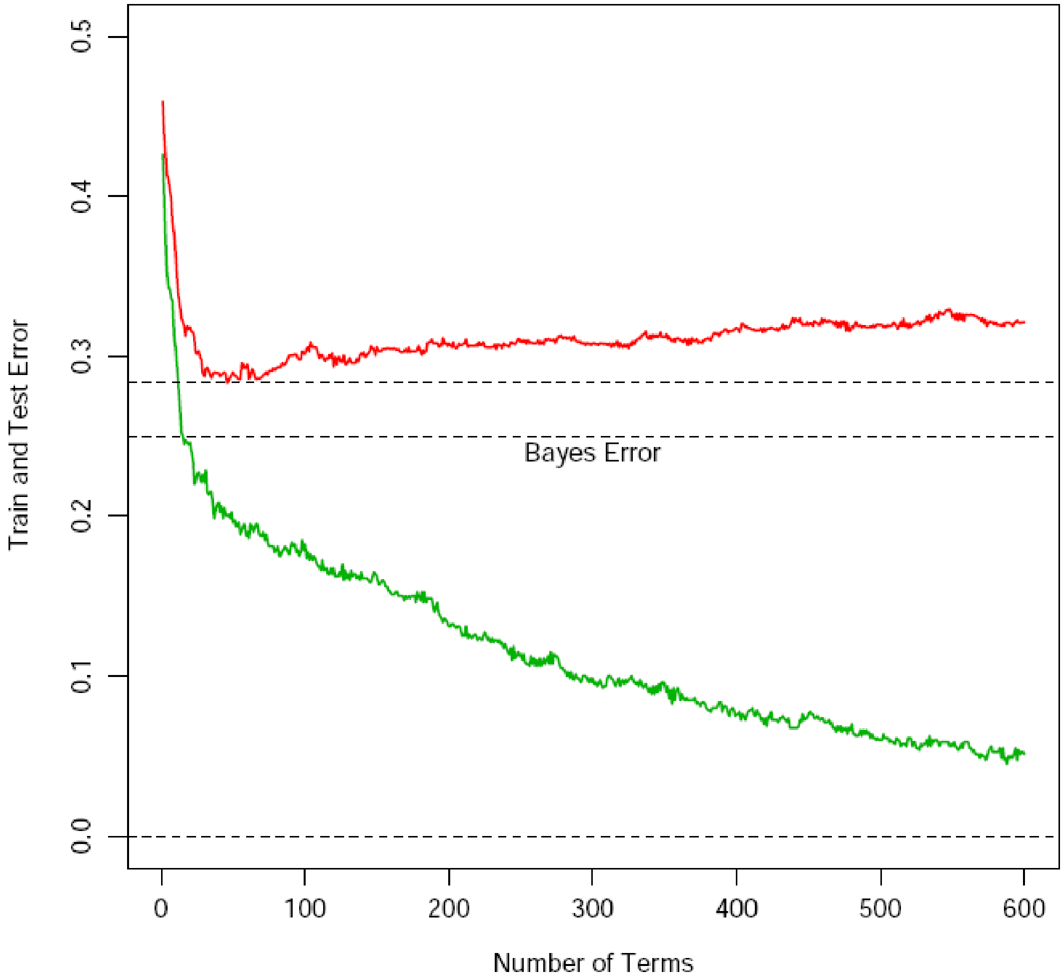
Boosting Stumps

A stump is a two-node tree, after a single split. Boosting stumps works remarkably well on the nested-spheres problem.



Training Error

- Nested spheres in 10-Dimensions.
- Bayes error is 0%.
- Boosting drives the training error to zero.
- Further iterations continue to improve test error in many examples.



Noisy Problems

- Nested Gaussians in 10-Dimensions.
- Bayes error is 25%.
- Boosting with stumps
- Here the test error does increase, but quite slowly.

Stagewise Additive Modeling

Boosting builds an additive model

$$f(x) = \sum_{m=1}^{M} \beta_m b(x; \gamma_m).$$

Here $b(x, \gamma_m)$ is a tree, and γ_m parametrizes the splits.

We do things like that in statistics all the time!

- GAMs: $f(x) = \sum_j f_j(x_j)$
- Basis expansions: $f(x) = \sum_{m=1}^{M} \theta_m h_m(x)$

Traditionally the parameters f_m, θ_m are fit jointly (i.e. least squares, maximum likelihood).

With boosting, the parameters (β_m, γ_m) are fit in a stagewise fashion. This slows the process down, and overfits less quickly.

Additive Trees

- Simple example: stagewise least-squares?
- Fix the past $M-1$ functions, and update the Mth using a tree:

$$\min_{f_M \in Tree(x)} E(Y - \sum_{m=1}^{M-1} f_m(x) - f_M(x))^2$$

- If we define the current residuals to be

$$R = Y - \sum_{m=1}^{M-1} f_m(x)$$

then at each stage we fit a tree to the residuals

$$\min_{f_M \in Tree(x)} E(R - f_M(x))^2$$

Stagewise Least Squares

Suppose we have available a basis family $b(x;\gamma)$ parametrized by γ.

- After $m-1$ steps, suppose we have the model $f_{m-1}(x)=\sum_{j=1}^{m-1}\beta_j b(x;\gamma_j)$.
- At the mth step we solve

$$\min_{\beta,\gamma}\sum_{i=1}^{N}\left(y_i - f_{m-1}(x_i) - \beta b(x_i;\gamma)\right)^2$$

- Denoting the residuals at the mth stage by $r_{im}=y_i-f_{m-1}(x_i)$, the previous step amounts to

$$\min_{\beta,\gamma}(r_{im}-\beta b(x_i;\gamma))^2,$$

- Thus the term $\beta_m b(x;\gamma_m)$ that best fits the current residuals is added to the expansion at each step.

Adaboost: Stagewise Modeling

- AdaBoost builds an additive logistic regression model

$$f(x) = \log \frac{\Pr(Y=1|x)}{\Pr(Y=-1|x)} = \sum_{m=1}^{M} \alpha_m G_m(x)$$

by stagewise fitting using the loss function

$$L(y, f(x)) = \exp(-y\, f(x)).$$

- Given the current $f_{M-1}(x)$, our solution for (β_m, G_m) is

$$\arg\min_{\beta, G} \sum_{i=1}^{N} \exp[-y_i(f_{m-1}(x_i) + \beta\, G(x))]$$

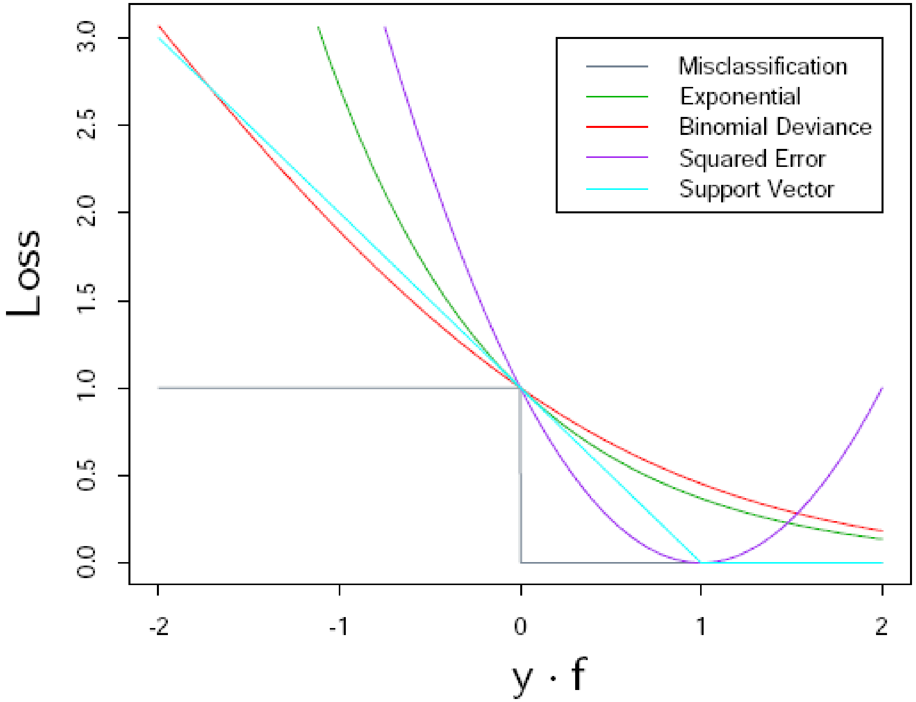
where $G_m(x) \in \{-1, 1\}$ is a tree classifier and β_m is a coefficient.

- With $w_i^{(m)} = \exp(-y_i\, f_{m-1}(x_i))$, this can be re-expressed as

$$\arg\min_{\beta,G} \sum_{i=1}^{N} w_i^{(m)} \exp(-\beta\, y_i\, G(x_i))$$

- We can show that this leads to the Adaboost algorithm; See  pp 343

Why Exponential Loss?



- $e^{-yF(x)}$ is a monotone, smooth upper bound on misclassification loss at x.
- Leads to simple reweighting scheme.
- Has logit transform as population minimizer

$$f^*(x) = \frac{1}{2} \log \frac{\Pr(Y=1|x)}{\Pr(Y=-1|x)}$$

- Other more robust loss functions, like binomial deviance.

General Stagewise Algorithm

We can do the same for more general loss functions, not only least squares.

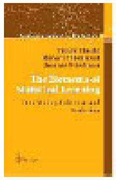
1. Initialize $f_0(x) = 0$.
2. For $m = 1$ to M:
   (a) Compute
   $(\beta_m, \gamma_m) = \arg\min_{\beta,\gamma} \sum_{i=1}^{N} L(y_i, f_{m-1}(x_i) + \beta b(x_i; \gamma))$.
   (b) Set $f_m(x) = f_{m-1}(x) + \beta_m b(x; \gamma_m)$.

Sometimes we replace step (b) in item 2 by

(b*) Set $f_m(x) = f_{m-1}(x) + \nu\beta_m b(x; \gamma_m)$

Here ν is a shrinkage factor, and often $\nu < 0.1$. Shrinkage slows the stagewise model-building even more, and typically leads to better performance.

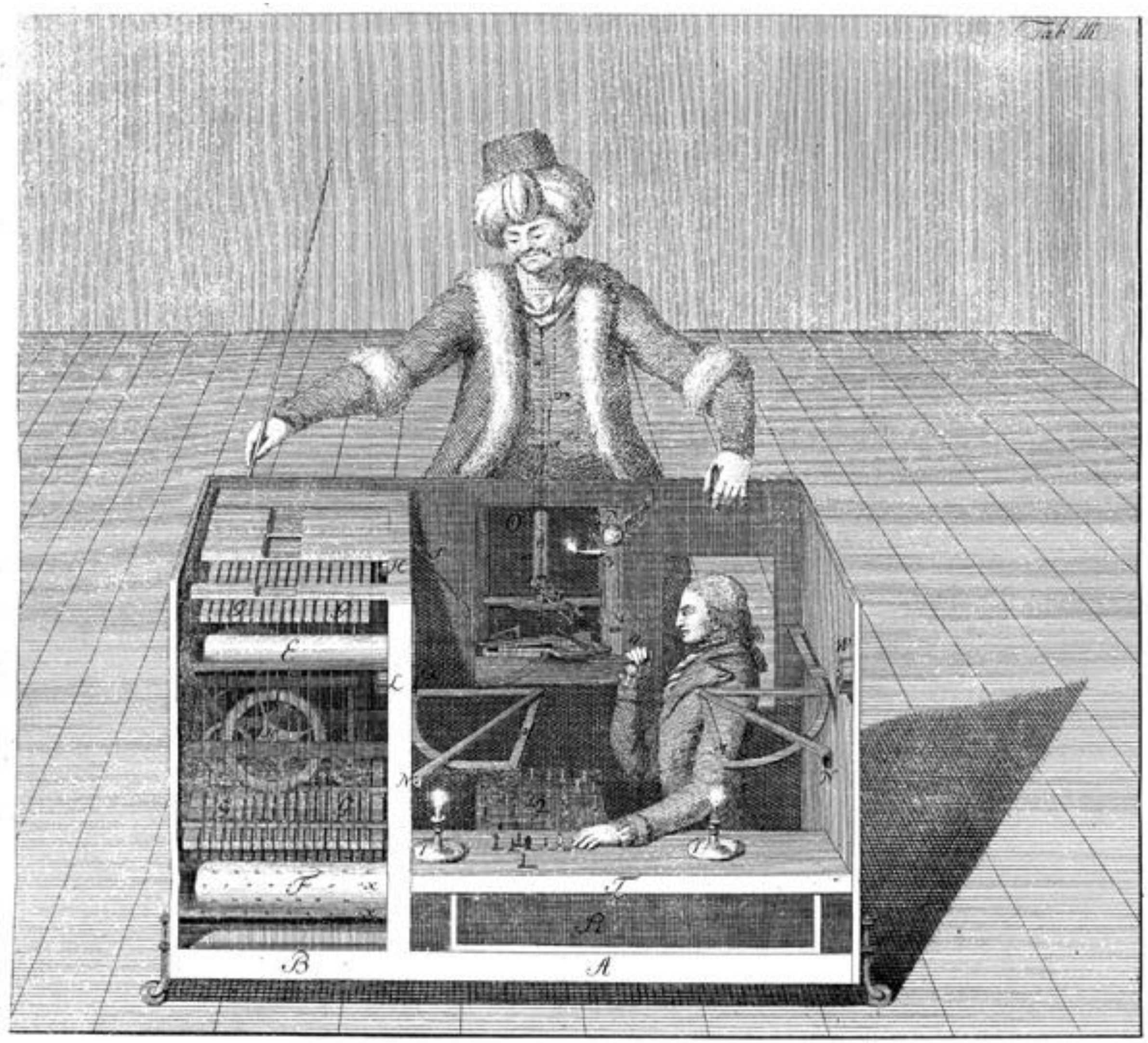
Gradient Boosting

- General boosting algorithm that works with a variety of different loss functions. Models include regression, resistant regression, K-class classification and risk modeling.
- Gradient Boosting builds additive tree models, for example, for representing the logits in logistic regression.
- Tree size is a parameter that determines the order of interaction (next slide).
- Gradient Boosting inherits all the good features of trees (variable selection, missing data, mixed predictors), and improves on the weak features, such as prediction performance.
- Gradient Boosting is described in detail in , section 10.10.

Learning from Crowds

- Garnering wisdom from a council of fools

The Turk



Crowdsourcing for Labeling

- Crowdsourcing helps to collect labels easier, faster and cheaper. But could be low quality.

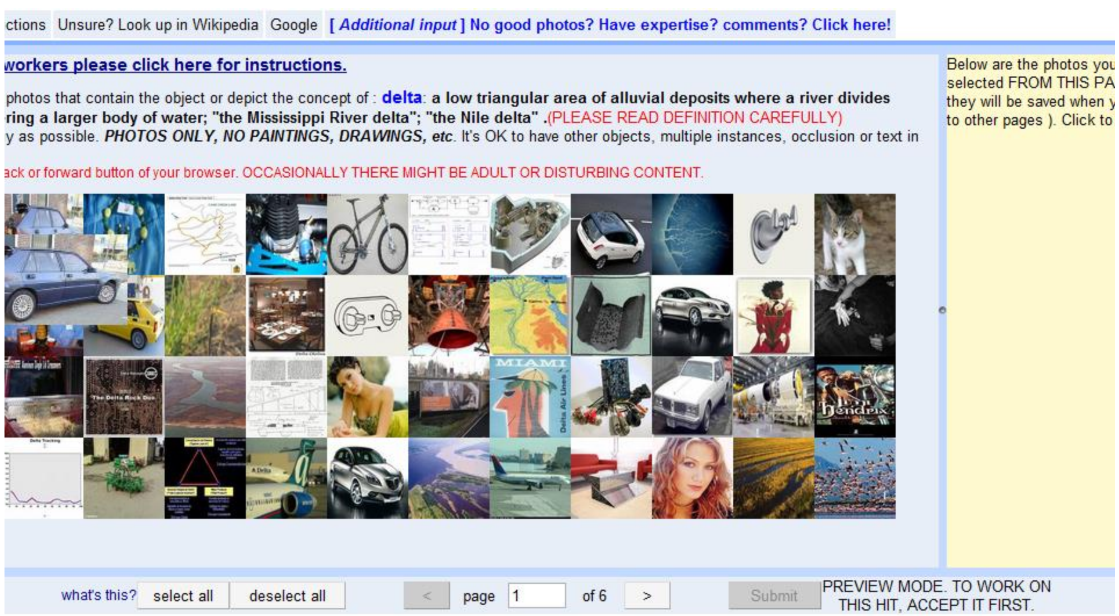
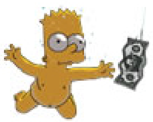


Figure from ImageNet

Author: L. Fei-Fei

Multiple Labels and Aggregation

				
	Apricot	Peach	Peach	Peach
	Apricot	Apricot	Apricot	Peach

Majority Voting (MV)

- Items: $i \in [M]$
- Each have a ground truth: $y_i \in [D]$
- Workers: $j \in [N]$
- Worker labels: $x_{ij} \in [D]$, $\boldsymbol{x}_i$: $\{x_{ij}, \forall j\}$

- Majority Voting: find the most frequent labels

$$\hat{y}_i = \underset{d \in [D]}{\operatorname{argmax}} \sum_{j=1}^{N} \mathbb{I}(x_{ij} = d), \forall i \in [M]$$

[Tian & Zhu. Max-margin Majority Voting for Learning from Crowds. NIPS 2015]

Constraint Formulation

- Expansion Expression
  - Def: $\boldsymbol{g}(\boldsymbol{x}_i, d) \in \{0,1\}^N$, element j is $\mathbb{I}(x_{ij} = d)$

$\boldsymbol{x}_i$: (1 -1 -1 -1) ⇒ $\boldsymbol{g}(\boldsymbol{x}_i, 1)$: (1 0 0 0)
$\boldsymbol{g}(\boldsymbol{x}_i, -1)$: (0 1 1 1)

- Constraint Formulation
  - MV is equivalent to find $\boldsymbol{y}$ satisfying the constraints:

$$\mathbf{1}_N^\top \boldsymbol{g}(\boldsymbol{x}_i, y_i) - \mathbf{1}_N^\top \boldsymbol{g}(\boldsymbol{x}_i, d) \geq 0, \quad \forall i, d$$

[Tian & Zhu. Max-margin Majority Voting for Learning from Crowds. NIPS 2015]

Max Margin Majority Voting (M³V)

- We introduce worker weights $\boldsymbol{\eta} \in \mathbb{R}^N$:

$$\hat{y}_i = \mathrm{argmax}_{d \in [D]}\, \boldsymbol{\eta}^\top \boldsymbol{g}(\boldsymbol{x}_i, d)$$

- Incorporate max-margin principle to estimate $\boldsymbol{\eta}$

$$\inf_{\boldsymbol{\eta}, \boldsymbol{y}} \|\boldsymbol{\eta}\|_2^2$$

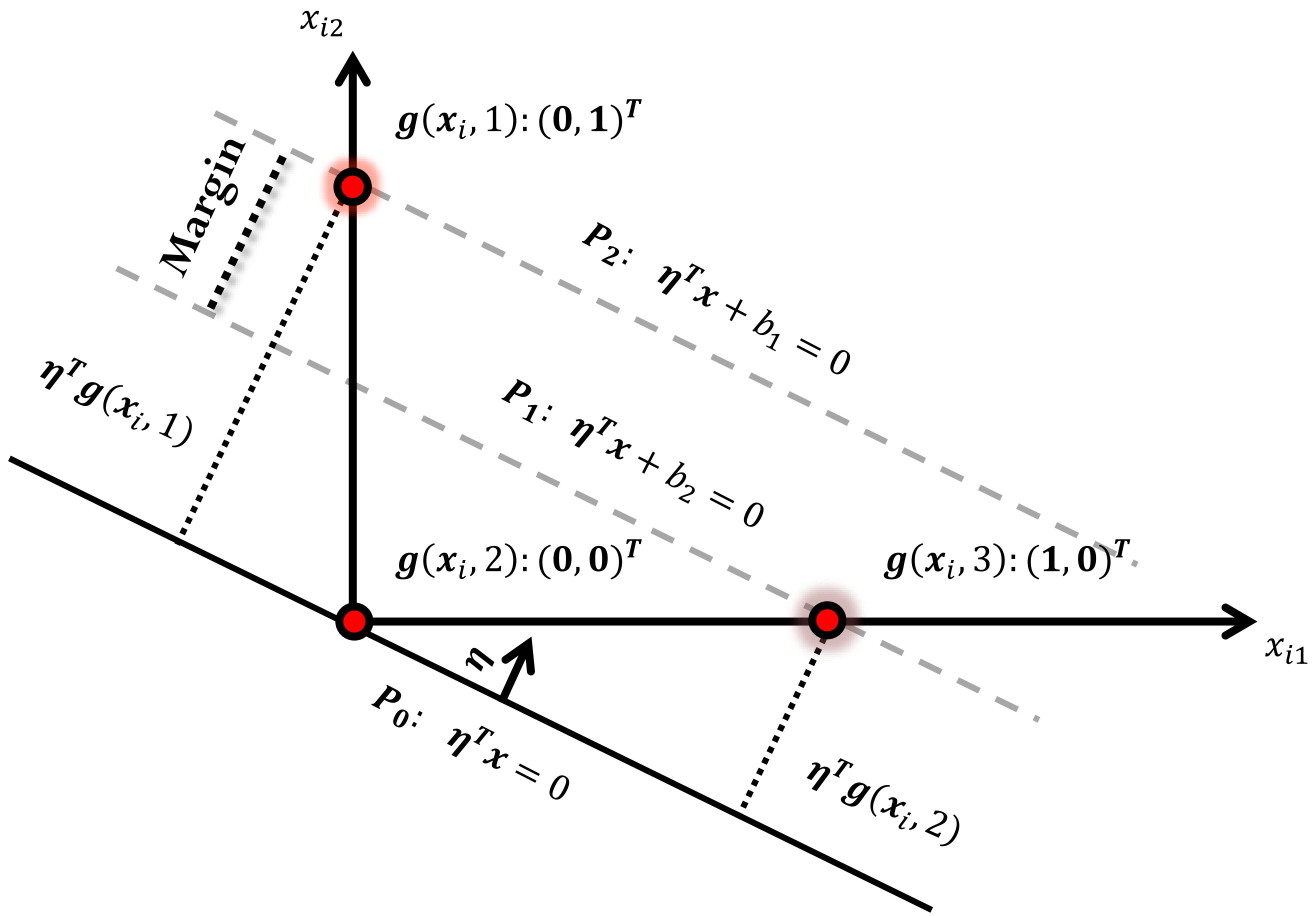
$$\mathrm{s.\,t.}: \boldsymbol{\eta}^\top \boldsymbol{g}_i^\Delta(d) \geq \ell_i^\Delta(d), \forall i \in [M], d \in [D]$$

where $\boldsymbol{g}_i^\Delta(d) := \boldsymbol{g}(\boldsymbol{x}_i, y_i) - \boldsymbol{g}(\boldsymbol{x}_i, d)\ ^2$ and $\ell_i^\Delta(d) = \ell \mathbb{I}(y_i \neq d)$.

- A soft version is solved by standard SVM solvers

[Tian & Zhu. Max-margin Majority Voting for Learning from Crowds. NIPS 2015]

Geometric Interpreting



Maximize the crowdsourcing margin

Dawid-Skene Model (DS)

- Define and estimate worker confusion matrices.
  - ϕ_j is the confusion matrix of worker j
  - $\phi_{jkd} = p(x_{ij} = d \mid y_i = k), \forall i$

	Apricot	Peach
Apricot	0.8	0.2
Peach	0.4	0.6

Worker Label

Ground Truth

CrowdSVM

- Consider Majority Voting and confusability in a single model.

M³V:

$$\inf_{\xi_i \geq 0, \boldsymbol{\eta}, \boldsymbol{y}} \|\boldsymbol{\eta}\|_2^2 + c \sum_i \xi_i$$

$$\mathrm{s.t.}: \boldsymbol{\eta}^\top \boldsymbol{g}_i^\Delta(d) \geq \ell_i^\Delta(d) - \xi_i, \forall i \in [M], d \in [D]$$

+

DS:

$$\inf_{q(\boldsymbol{\Phi}, \boldsymbol{\eta})} \mathcal{L}\left(q(\boldsymbol{\Phi}, \boldsymbol{\eta}); \boldsymbol{y}\right),$$

$$\mathcal{L}(q; \boldsymbol{y}) := \mathrm{KL}(q \| p_0(\boldsymbol{\Phi}, \boldsymbol{\eta})) - \mathbb{E}_q[\log p(\boldsymbol{X} | \boldsymbol{\Phi}, \boldsymbol{y})]$$

→

CrowdSVM:

$$\inf_{\xi_i \geq 0, q \in \mathcal{P}, \boldsymbol{y}} \mathcal{L}(q(\boldsymbol{\Phi}, \boldsymbol{\eta}); \boldsymbol{y}) + c \cdot \sum_i \xi_i$$

$$\mathrm{s.t.}: \mathbb{E}_q[\boldsymbol{\eta}^\top \boldsymbol{g}_i^\Delta(d)] \geq \ell_i^\Delta(d) - \xi_i, \forall i \in [M], d \in [D],$$

Variational Inference

regularized Bayesian inference (Zhu et al. 2014)

Gibbs CrowdSVM

- From average loss to expected loss
- Average:

$$\mathcal{R}_m(q;\boldsymbol{y}) = \sum_{i=1}^{M} \max_{d=1}^{D} \big(\ell_i^{\Delta}(d) - \mathbb{E}_q[\boldsymbol{\eta}^\top \boldsymbol{g}_i^{\Delta}(d)]\big)_+$$

- Expected:

$$\mathcal{R}'_m(q(\boldsymbol{\Phi},\boldsymbol{\eta});\boldsymbol{y}) = \mathbb{E}_q\left[\mathcal{R}(\boldsymbol{\eta},\boldsymbol{y})\right] \qquad \mathcal{R}(\boldsymbol{\eta},\boldsymbol{y}) = \sum_{i=1}^{M} \max_{d\in[D]} \left(\ell_i^{\Delta}(d) - \boldsymbol{\eta}^\top \boldsymbol{g}_i^{\Delta}(d)\right)_+$$

$$\boxed{\inf_{q\in\mathcal{P}} \mathcal{L}\Big(q(\boldsymbol{\Phi},\boldsymbol{\eta},\boldsymbol{y})\Big) + \mathbb{E}_q\left[\sum_{i=1}^{M} 2c(\zeta_{is_i})_+\right],}$$

where $\zeta_{id} = \ell_i^{\Delta}(d) - \boldsymbol{\eta}^\top \boldsymbol{g}_i^{\Delta}(d)$, $s_i = \operatorname{argmax}_{d\neq y_i} \zeta_{id}$

- Introduce augmented variables to de Gibbs Sampling

Gibbs CrowdSVM

- Unconstraint Form:

$$\inf_{q \in \mathcal{P}, \boldsymbol{y}} \mathcal{L}(q(\boldsymbol{\Phi}, \boldsymbol{\eta}); \boldsymbol{y}) + c \cdot \mathcal{R}_m(q(\boldsymbol{\Phi}, \boldsymbol{\eta}); \boldsymbol{y}),$$

where $\mathcal{R}_m(q; \boldsymbol{y}) = \sum_{i=1}^{M} \max_{d=1}^{D} \big(\ell_i^{\Delta}(d) - \mathbb{E}_q[\boldsymbol{\eta}^\top \boldsymbol{g}_i^{\Delta}(d)]\big)_+$ is the posterior regularization.

- Non-conjugate for $\boldsymbol{\eta}$, so we introduce augment variable $\boldsymbol{\lambda}$

$$q(\boldsymbol{\Phi}, \boldsymbol{\eta}, \boldsymbol{y}, \boldsymbol{\lambda}) \propto p_0(\boldsymbol{\Phi}, \boldsymbol{\eta}, \boldsymbol{y}) \prod_{i=1}^{M} p(\boldsymbol{x}_i | \boldsymbol{\Phi}, y_i) \psi(y_i, \lambda_i | \boldsymbol{x}_i, \boldsymbol{\eta}).$$

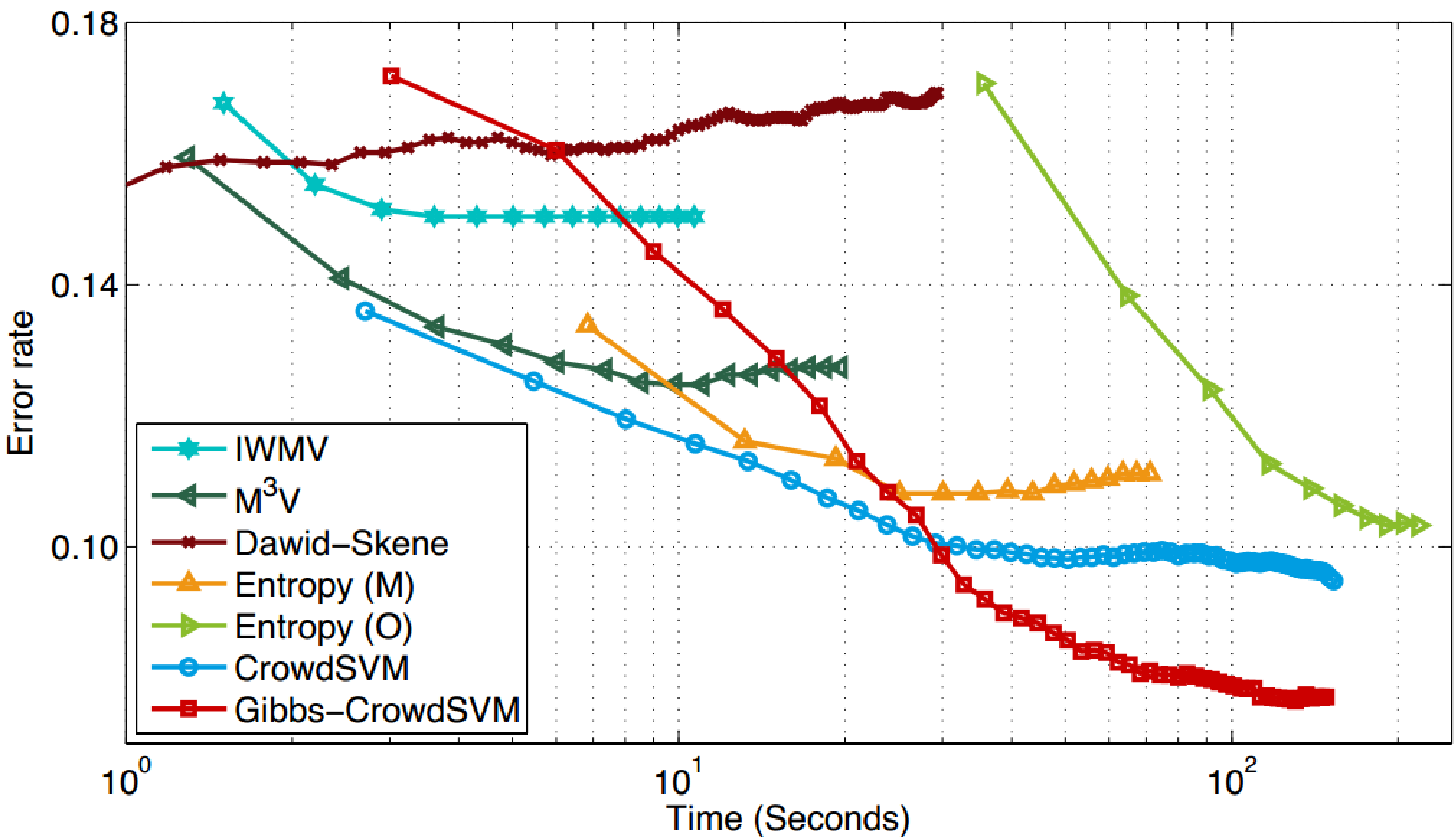
Experimental Results

Dataset	Classes	Labels	Items	Workers
Web Search	5	15,567	2,665	177
Age	7	10,020	1,002	165
Bluebirds	2	4,214	108	39
Flowers	2	2,366	200	36

Table 2: Error-rates (%) of different estimators on four datasets.

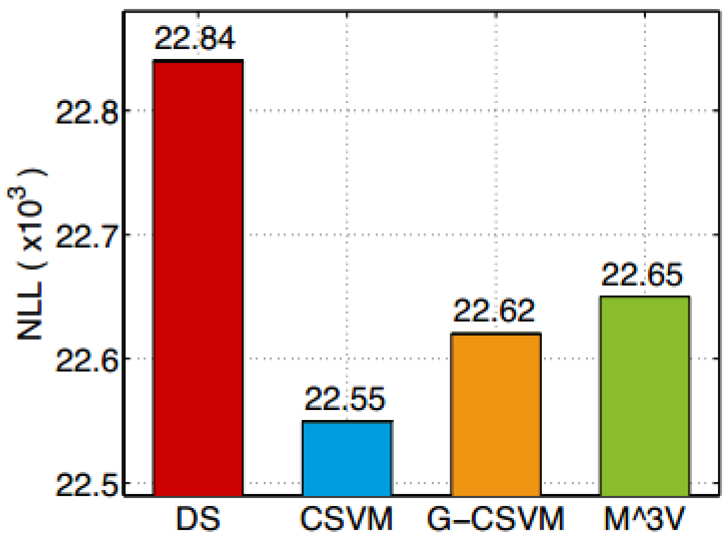
	Method	Web Search	Age	Bluebirds	Flowers
I	MV	26.90	34.88	24.07	22.00
	IWMV	15.04	34.53	27.78	19.00
	M^3V	**12.74**	**33.33**	**20.37**	**13.50**
II	DS	16.92	39.62	10.19	**13.00**
	DS+Prior	13.26	34.53	10.19	13.50
	CrowdSVM	**9.42**	**33.33**	10.19	13.50
III	ME	10.40	**31.14**	**8.33**	13.00
	G-CrowdSVM	**7.99 ± 0.26**	32.98 ± 0.36	10.37±0.41	**12.10 ± 1.07**

Convergence

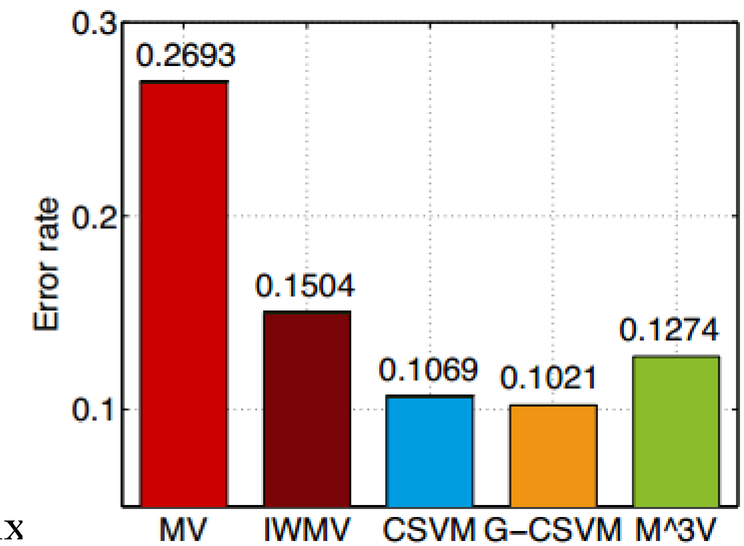


Generative vs. Discriminative

- Both component benefits from the other



: fix



we

only use η for prediction.

What you need to know

- Classification tree
- Model averaging techniques
  - Bagging
  - Random forests
  - Boosting
- Learning from crowds
  - Still an active direction

Thank You!