

# Ensembles. Stacked generalization. AdaBoost

Evgeny Burnaev

Skoltech, Moscow, Russia

- 1 Ensembles of classifiers
- 2 Stacked generalization
- 3 Boosting

## 1 Ensembles of classifiers

## 2 Stacked generalization

## 3 Boosting

- Motivation
- Learning of ensembles of classifiers

- A predictor, feature  $\mathbf{x} \in \mathbb{R}^d$  has distribution  $D$
- $f(\mathbf{x})$  is a deterministic function from some concept class
- **Goal:**
  - Based on  $m$  training pairs  $\{(\mathbf{x}_i, y_i = f(\mathbf{x}_i))\}_{i=1}^m$  drawn i.i.d. from  $D$  produce a classifier  $\hat{f}(\mathbf{x}) \in \{0, 1\}$
  - Choose  $\hat{f}$  to have low generalization error
$$R(\hat{f}) = \mathbb{E}_D \left[ 1_{\hat{f}(\mathbf{x}) \neq f(\mathbf{x})} \right]$$

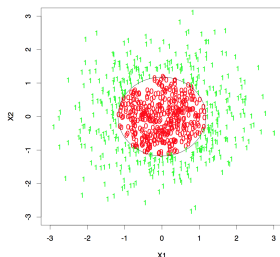
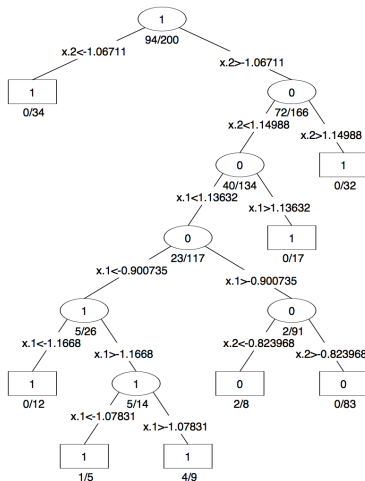
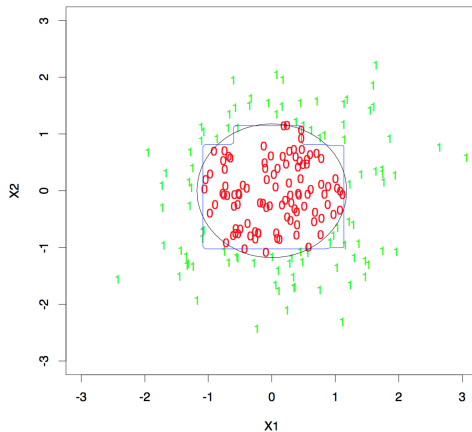


Figure – “Sphere” in  $\mathbb{R}^{10}$

Sample of size 200



Sample of size 200



In case of “Sphere” in  $\mathbb{R}^{10}$  CART produces a rather noisy and inaccurate rule  $\hat{f}(x)$ , with error rates around 30%

- For simplicity we consider a binary classification problem. Let us denote by  $h_1(\mathbf{x}), \dots, h_T(\mathbf{x})$  some binary classifiers
- Typical ensembling procedure has the form
  - Simple voting:

$$H(h_1(\mathbf{x}), \dots, h_T(\mathbf{x})) = \frac{1}{T} \sum_{t=1}^T h_t(\mathbf{x}),$$

- Weighted voting:

$$H(h_1(\mathbf{x}), \dots, h_T(\mathbf{x})) = \sum_{t=1}^T \alpha_t h_t(\mathbf{x}),$$

- Mixture of experts

$$H(h_1(\mathbf{x}), \dots, h_T(\mathbf{x})) = \sum_{t=1}^T g_t(\mathbf{x}) h_t(\mathbf{x})$$

- Final decision

$$f_T(\mathbf{x}) = \text{sign}\{H(h_1(\mathbf{x}), \dots, h_T(\mathbf{x}))\}$$

- Bagging or “bootstrap averaging” averages a given procedure over many samples to reduce its variance
- Let us denote by
  - $S_m = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_m, y_m)\}$  a sample of size  $m$
  - $\hat{h}_S(\mathbf{x})$  a classifier, such as a tree, trained using the sample  $S$
- To bag  $\hat{f}$  we draw bootstrap samples  $S^{*,1}, \dots, S^{*,B}$  each of size  $m$  with replacement from the training data
- In each bootstrap sample  $S^{*,1}, \dots, S^{*,B}$  we use only subset of randomly selected features
- We train classifiers  $\hat{h}_{S^{*,b}}(\mathbf{x})$  on each of  $S^{*,b}$ ,  $b = 1, \dots, B$
- Then

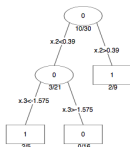
$$\hat{f}_{\text{bag}}(\mathbf{x}) = \text{MajorityVote} \left\{ \left( \hat{h}_{S^{*,b}}(\mathbf{x}) \right)_{b=1}^B \right\}$$

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

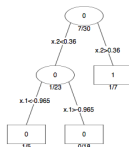


# Example: Bagging

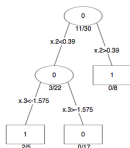
Original Tree



Bootstrap Tree 1



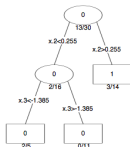
Bootstrap Tree 2



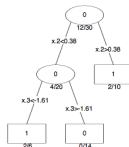
Bootstrap Tree 3

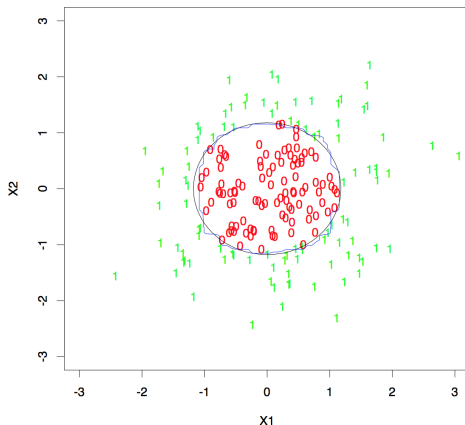


Bootstrap Tree 4



Bootstrap Tree 5





“Sphere” in  $\mathbb{R}^{10}$ : Bagging averages many trees, and produces smoother decision boundaries

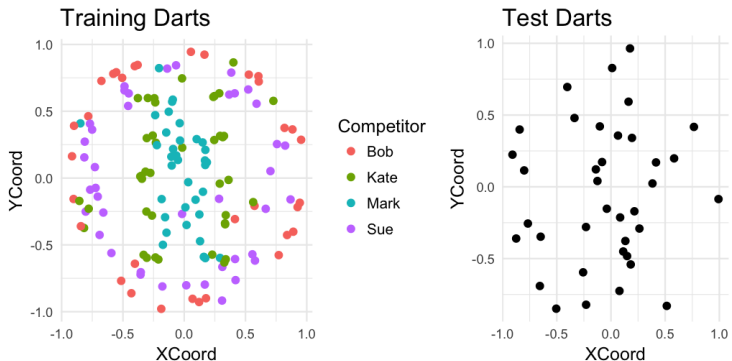
1 Ensembles of classifiers

2 Stacked generalization

3 Boosting

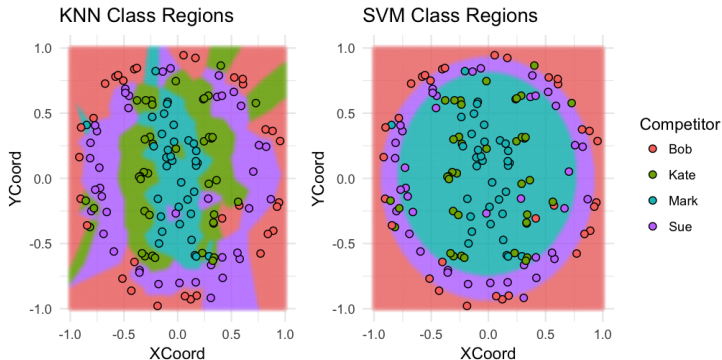
- Motivation
- Learning of ensembles of classifiers

# Stacking motivation: the game of Darts



Picture credit: <http://blog.kaggle.com/2016/12/27/a-kagglers-guide-to-model-stacking-in-practice>

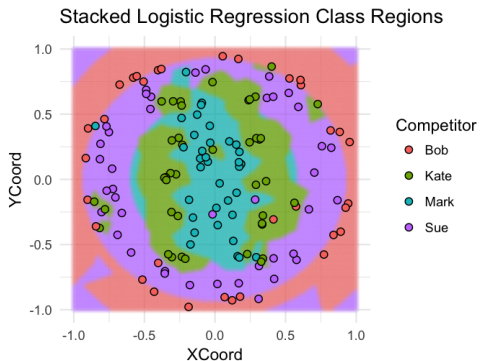
- Select  $k$  nearest neighbours as base model 1
  - Fit base model 1 in the most fancy way possible (grid search for optimal  $k$  using  $K$ -fold cross-validation, etc.)
  - $k$ -NN accuracy on Test Darts: 70%
- 
- Select Support Vector Machine as base model 2
  - Fit base model 2 in the most fancy way possible (different penalizations, grid search for optimal kernel width using  $K$ -fold cross-validation, etc.)
  - SVM accuracy on Test Darts: 78%



Picture credit: <http://blog.kaggle.com/2016/12/27/a-kagglers-guide-to-model-stacking-in-practice>

Stacked generalization aka **stacking**: blend output of weak learners (weak signals) with raw features

- ① Partition train into 5 folds
- ② Create train\_meta/test\_meta: same row/fold Ids as in train/test, empty M1/M2
- ③ For each  $\text{Fold}_i \in \{\text{Fold}_1, \dots, \text{Fold}_5\}$ 
  - Combine the other 4 folds for training  $\rightarrow \text{Fold}_{-i}$
  - Fit each base model to  $\text{Fold}_{-i}$ , predict on  $\text{Fold}_i$ , save predictions to M1/M2 in train\_meta
- ④ Fit each base model to train, predict on test, save predictions to M1/M2 in test\_meta
- ⑤ Fit stacking model  $F$  to train\_meta, using M1/M2 as features
- ⑥ Use the stacked model  $F$  to make final predictions on test\_meta



Picture credit: <http://blog.kaggle.com/2016/12/27/a-kagglers-guide-to-model-stacking-in-practice>



- **Bootstrapping**: a general statistical technique for computing sample functionals (and their variance)
- **Bagging**: meta-learner over arbitrary weak algorithms via **bootstrap aggregation**
- **The Random Forest algorithm**: bagging over decision trees
- Stacked generalization aka **stacking**: blend output of weak learners (weak signals) with raw features

1 Ensembles of classifiers

2 Stacked generalization

3 Boosting

- Motivation
- Learning of ensembles of classifiers

1 Ensembles of classifiers

2 Stacked generalization

3 Boosting

- Motivation
- Learning of ensembles of classifiers

- **problem:** filter out spam (junk email)
- gather large collection of examples of **spam** and **non-spam**

From: yoav@att.com	Rob, can you review a paper...	non-spam
From: xa412@hotmail.com	Earn money without working!!!! ...	spam
⋮	⋮	⋮

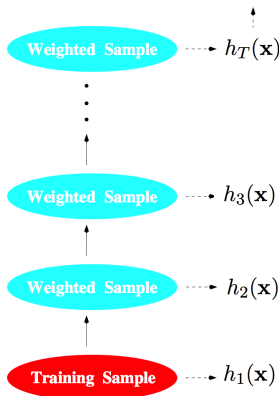
- **goal:** get computer learn from examples to distinguish spam from non-spam
- **main observation:**
  - **easy** to find “rules of thumb” that are “often” correct
    - if ‘v1agr@’ occurs in message, then predict “**spam**”
  - **hard** to find single rule that is very highly accurate

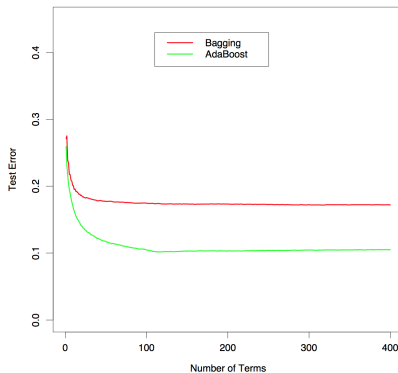
- devise computer program for deriving rough rules of thumb
- apply procedure to subset of emails
- obtain rule of thumb
- apply to 2nd subset of emails
- obtain 2nd rule of thumb
- repeat  $T$  times
- aggregate

## Final Classifier

$$f_T(\mathbf{x}) = \text{sign} \left[ \sum_{t=1}^T \alpha_t h_t(\mathbf{x}) \right]$$

1. How to choose examples on each round?
  - concentrate on “hardest” examples (those most often misclassified by previous rules of thumb)
2. How to combine rules of thumb into single prediction rule?
  - take (weighted) majority vote of rules of thumb





- 2000 points, “Sphere” in  $\mathbb{R}^{10}$ ; Bayes error rate is 0%
- Trees are grown Best First without pruning
- Leftmost iteration is a single tree

1 Ensembles of classifiers

2 Stacked generalization

3 Boosting

- Motivation

- Learning of ensembles of classifiers



- We consider a binary classification with  $Y = \{-1, +1\}$
- As a strong classifier we consider weighted voting scheme, i.e.

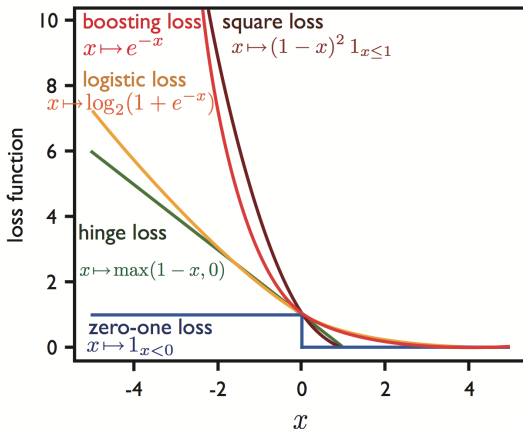
$$\hat{f}_T(\mathbf{x}) = \text{sign} \left( \sum_{t=1}^T \alpha_t h_t(\mathbf{x}) \right)$$

- As a risk we consider accuracy, i.e.

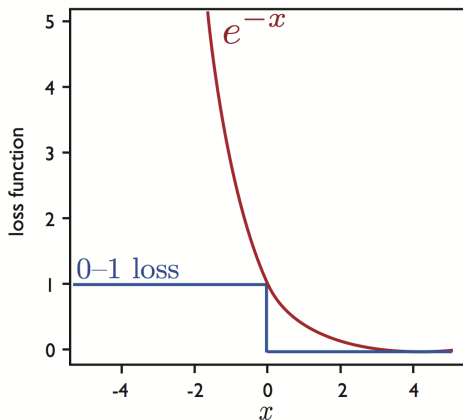
$$\hat{R}(f_T) = \frac{1}{m} \sum_{i=1}^m 1 \left\{ y_i \cdot \left[ \sum_{t=1}^T \alpha_t h_t(\mathbf{x}_i) \right] \leq 0 \right\}$$

- Two main heuristics, underlying boosting
  - We fix  $\alpha_1 h_1(\mathbf{x}), \dots, \alpha_{t-1} h_{t-1}(\mathbf{x})$  when adding  $\alpha_t h_t(\mathbf{x})$
  - We use continuous upper bound for accuracy

- Examples of several convex upper bounds on the zero-one loss



- Objective Function: convex and differentiable



- Since  $1_{z \leq 0} \leq e^{-z}$ , we get that

$$1_{\{y_i \cdot \sum_{t=1}^T \alpha_t h_t(\mathbf{x}_i) \leq 0\}} \leq \exp \left( -y_i \sum_{t=1}^T \alpha_t h_t(\mathbf{x}_i) \right)$$

- Let us consider an upper bound for  $\hat{R}(f)$

$$\begin{aligned} \hat{R}(f_T) &= \frac{1}{m} \sum_{i=1}^m 1_{\{y_i \cdot \sum_{t=1}^T \alpha_t h_t(\mathbf{x}_i) \leq 0\}} \leq \\ &\leq \tilde{R}(f_T) = \frac{1}{m} \sum_{i=1}^m \exp \left( -y_i \sum_{t=1}^T \alpha_t h_t(\mathbf{x}_i) \right) \end{aligned}$$

- We will optimize  $\hat{R}(f_T)$  w.r.t. a new weak classifier  $h_T(\mathbf{x})$  and its weight  $\alpha_T$  given that  $\{\alpha_t, h_t(\mathbf{x})\}_{t=1}^{T-1}$  are fixed

- Let us denote

$$w_{i,T-1} = \exp \left( -y_i \sum_{t=1}^{T-1} \alpha_t h_t(\mathbf{x}_i) \right)$$
$$\tilde{w}_{i,T} = \frac{w_{i,T-1}}{\sum_j w_{j,T-1}}$$

- Then

$$\tilde{R}(f_{T-1}) = \frac{1}{m} \sum_{i=1}^m \underbrace{\exp \left( -y_i \sum_{t=1}^{T-1} \alpha_t h_t(\mathbf{x}_i) \right)}_{w_{i,T-1}} =$$
$$= \frac{1}{m} \sum_{i=1}^m w_{i,T-1}$$

Let us re-write an upper bound  $\tilde{R}(f)$

$$\begin{aligned}\tilde{R}(f_T) &= \frac{1}{m} \sum_{i=1}^m \underbrace{\exp \left( -y_i \sum_{t=1}^T \alpha_t h_t(\mathbf{x}_i) \right)}_{w_{i,T}} = \\ &= \frac{1}{m} \sum_{i=1}^m \underbrace{\exp \left( -y_i \sum_{t=1}^{T-1} \alpha_t h_t(\mathbf{x}_i) \right)}_{w_{i,T-1}} \exp(-y_i \alpha_T h_T(\mathbf{x}_i)) \\ &= \frac{1}{m} \sum_{i=1}^m w_{i,T-1} \exp(-y_i \alpha_T h_T(\mathbf{x}_i))\end{aligned}$$

Let us re-write an upper bound  $\tilde{R}(f)$

$$\begin{aligned}\tilde{R}(f_T) &= \frac{1}{m} \sum_{i=1}^m w_{i,T-1} \exp(-y_i \alpha_T h_T(\mathbf{x}_i)) = \\&= \frac{1}{m} \sum_{i=1}^m \frac{[\sum_{k=1}^m w_{k,T-1}]}{[\sum_{k=1}^m w_{k,T-1}]} w_{i,T-1} \exp(-y_i \alpha_T h_T(\mathbf{x}_i)) = \\&= \frac{1}{m} \left[ \sum_{k=1}^m w_{k,T-1} \right] \sum_{i=1}^m \frac{w_{i,T-1}}{[\sum_{k=1}^m w_{k,T-1}]} \exp(-y_i \alpha_T h_T(\mathbf{x}_i)) = \\&= \underbrace{\frac{1}{m} \sum_{k=1}^m w_{k,T-1}}_{\tilde{R}(f_{T-1})} \cdot \sum_{i=1}^m \tilde{w}_{i,T} \exp(-y_i \alpha_T h_T(\mathbf{x}_i)) = \\&= \tilde{R}(f_{T-1}) \cdot \sum_{i=1}^m \tilde{w}_{i,T} e^{-y_i \alpha_T h_T(\mathbf{x}_i)}\end{aligned}$$

- Upper bound  $\tilde{R}(f_{T-1})$  is fixed from the previous boosting step
- Let us optimize an upper bound for  $\hat{R}(f)$

$$\hat{R}(f_T) \leq \tilde{R}(f_{T-1}) \cdot \sum_{i=1}^m \tilde{w}_{i,T} \exp(-y_i \alpha_T h_T(\mathbf{x}_i)) \rightarrow \min_{\alpha_T, h_T(\cdot)},$$

i.e. we tune only the weak classifier  $h_T(\cdot)$  and its weight  $\alpha_T$



- An upper bound for  $\widehat{R}(f)$

$$\widehat{R}(f_T) \leq \widetilde{R}(f_{T-1}) \cdot \sum_{i=1}^m \widetilde{w}_{i,T} \exp(-y_i \alpha_T h_T(\mathbf{x}_i)) \rightarrow \min_{\alpha_T, h_T(\cdot)}$$

- For  $\widetilde{\mathbf{w}}_T = (\widetilde{w}_{1,T}, \dots, \widetilde{w}_{m,T})$  we define a weighted classification error

$$N_T = N(h_T, \widetilde{\mathbf{w}}_T) = \sum_{i=1}^m \widetilde{w}_{i,T} \cdot 1_{\{y_i \cdot h_T(\mathbf{x}_i) \leq 0\}}, \quad P_T = 1 - N_T,$$

- Since for the weak learner  $y_i \cdot h_T(\mathbf{x}_i) \in \{-1, +1\}$ , then

$$\begin{aligned} \sum_{i=1}^m \widetilde{w}_{i,T} \exp(-y_i \alpha_T h_T(\mathbf{x}_i)) &= \\ &= \sum_{i: y_i \cdot h_T(\mathbf{x}_i) = -1} \widetilde{w}_{i,T} \exp(\alpha_T) + \sum_{i: y_i \cdot h_T(\mathbf{x}_i) = +1} \widetilde{w}_{i,T} \exp(-\alpha_T) = \\ &= N_T \exp(\alpha_T) + (1 - N_T) \exp(-\alpha_T) \end{aligned}$$

- We get that

$$\widehat{R}(f_T) \leq \widetilde{R}(f_{T-1}) \cdot (e^{-\alpha_T} (1 - N_T) + e^{\alpha_T} N_T) \rightarrow \min_{\alpha_T, h_T(\cdot)}$$

- We proved that

$$\widehat{R}(f_T) \leq \widetilde{R}(f_{T-1}) \cdot (e^{-\alpha_T}(1 - N_T) + e^{\alpha_T} N_T) \rightarrow \min_{\alpha_T, h_T(\cdot)}$$

- Let us fix  $h_T(\cdot)$ . Then optimal  $\alpha_T$  is equal to

$$\begin{aligned}\alpha_T^* &= \arg \min_{\alpha_T} (e^{-\alpha_T}(1 - N_T) + e^{\alpha_T} N_T) = \\ &= \frac{1}{2} \log \frac{P_T}{N_T} = \frac{1}{2} \log \frac{1 - N_T(h_T, \widetilde{\mathbf{w}}_T)}{N_T(h_T, \widetilde{\mathbf{w}}_T)}\end{aligned}$$

- For optimal  $\alpha_T^*$  we get that

$$\begin{aligned}\widehat{R}(f_T) &\leq \widetilde{R}(f_{T-1}) \cdot \left( e^{-\alpha_T^*} (1 - N_T) + e^{\alpha_T^*} N_T \right) \\ &\leq \widetilde{R}(f_{T-1}) \cdot \left( 1 - \left( \sqrt{P_T} - \sqrt{N_T} \right)^2 \right) \rightarrow \min_{h_T(\cdot)}\end{aligned}$$

- Then optimal  $h_T(\cdot)$  is given by

$$h_T^*(\cdot) = \arg \max_{h_T(\cdot)} \left( \sqrt{P_T(h_T, \widetilde{\mathbf{w}}_T)} - \sqrt{N_T(h_T, \widetilde{\mathbf{w}}_T)} \right)^2$$

- Let us assume that  $h_T(\cdot)$  is weak learnable, i.e. we can find  $h_T(\cdot)$  such that  $N_T < P_T = 1 - N_T$

- For  $N_T < P_T = 1 - N_T$  the problem

$$h_T^*(\cdot) = \arg \max_{h_T(\cdot)} \left( \sqrt{P_T(h_T, \tilde{\mathbf{w}}_T)} - \sqrt{N_T(h_T, \tilde{\mathbf{w}}_T)} \right)^2$$

reduces to

- The problem

$$h_T^*(\cdot) = \arg \max_{h(\cdot)} \left\{ \sqrt{P_T} - \sqrt{N_T} \right\} = \arg \min_{h(\cdot)} N_T(h, \tilde{\mathbf{w}}_T),$$

where

- weighted classification error

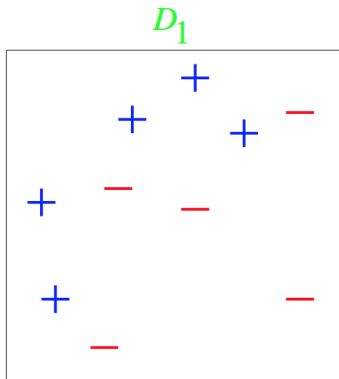
$$N_T = N(h_T, \tilde{\mathbf{w}}_T) = \sum_{i=1}^m \tilde{w}_{i,T} 1_{\{y_i \cdot h_T(\mathbf{x}_i) \leq 0\}}$$

- weights

$$w_{i,T-1} = \exp(-y_i f_{T-1}(\mathbf{x}_i)), \quad \tilde{w}_{i,T} = \frac{w_{i,T-1}}{\sum_j w_{j,T-1}}$$

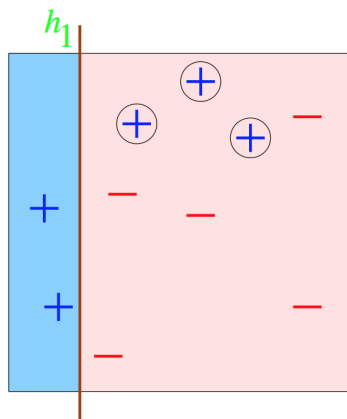
AdaBoost( $S_m = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_m, y_m)\}$ )

1. **for**  $i \leftarrow 1$  **to**  $m$  **do**
2.      $w_{i,1} \leftarrow \frac{1}{m}$
3. **for**  $t \leftarrow 1$  **to**  $T$  **do**
4.     Learn a based classifier:  
       $h_t \leftarrow$  base classif. with small  $N(h_t, \tilde{\mathbf{w}}_t) = \sum_{i=1}^m \tilde{w}_{i,t} 1_{\{y_i \cdot h_t(\mathbf{x}_i) \leq 0\}}$
5.      $\alpha_t \leftarrow \frac{1}{2} \log \frac{1 - N(h_t, \tilde{\mathbf{w}}_t)}{N(h_t, \tilde{\mathbf{w}}_t)}$
7.     **for**  $i \leftarrow 1$  **to**  $m$  **do**
8.          $w_{i,t+1} \leftarrow w_{i,t} \exp(-\alpha_t y_i h_t(\mathbf{x}_i))$
9.          $\tilde{w}_{i,t+1} \leftarrow \frac{w_{i,t+1}}{\sum_{j=1}^m w_{j,t+1}}$
10.     $f_t \leftarrow \sum_{s=1}^t \alpha_s h_s$
10. **return**  $\hat{f}_T = \text{sign}(f_T)$



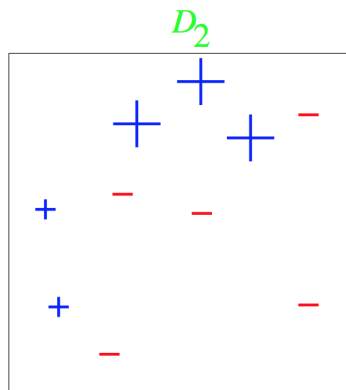
Weak classifiers = vertical or horizontal half-planes

## Toy Example: Round 1

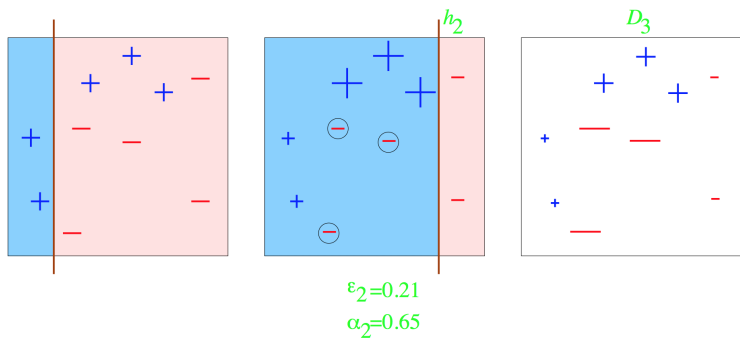


$$\varepsilon_1 = 0.30$$

$$\alpha_1 = 0.42$$

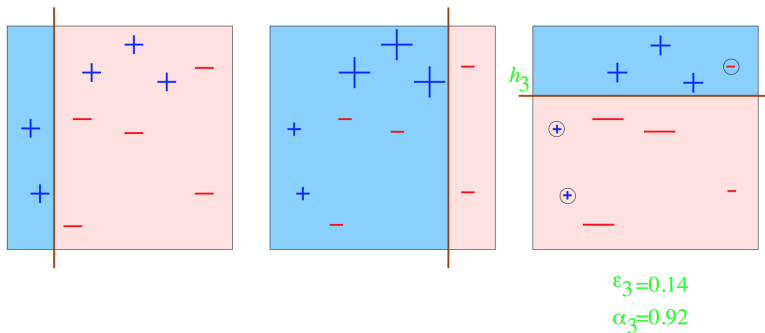


## Toy Example: Round 2

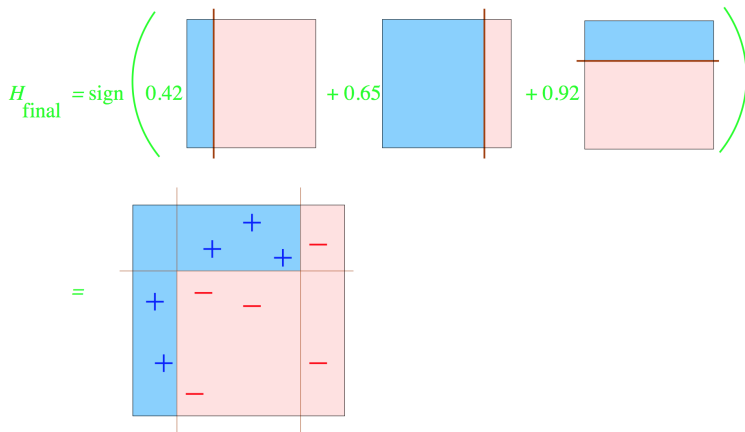




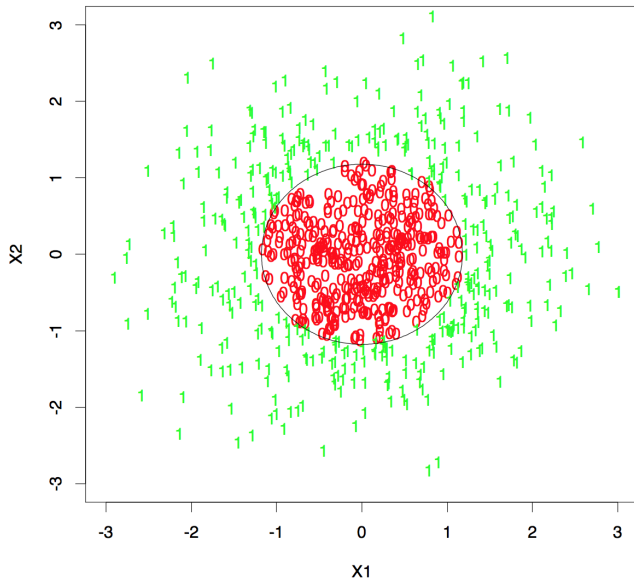
## Toy Example: Round 3



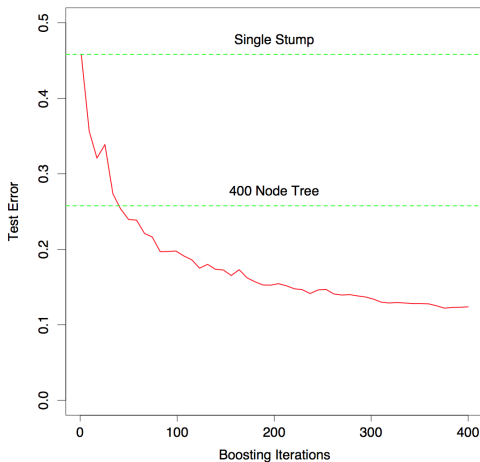
## Toy Example: Final Classifier



## Example: “Sphere” in $\mathbb{R}^{10}$

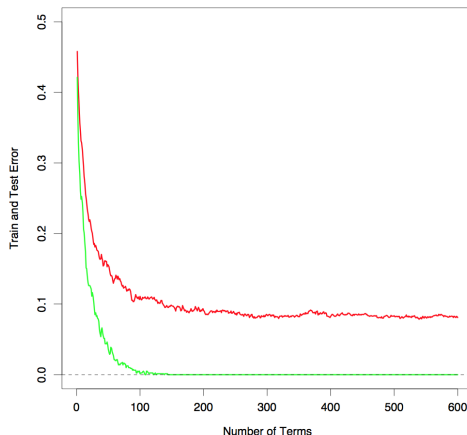


# Boosting Stumps



“Sphere” in  $\mathbb{R}^{10}$ : A stump is a two-node tree, after a single split. Boosting stumps works remarkably well on this problem

## Stumps



“Sphere” in  $\mathbb{R}^{10}$ : Boosting drives the training error to zero. Further iterations continue to improve test error in many examples

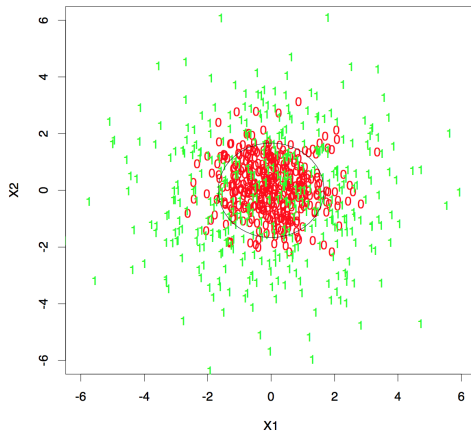
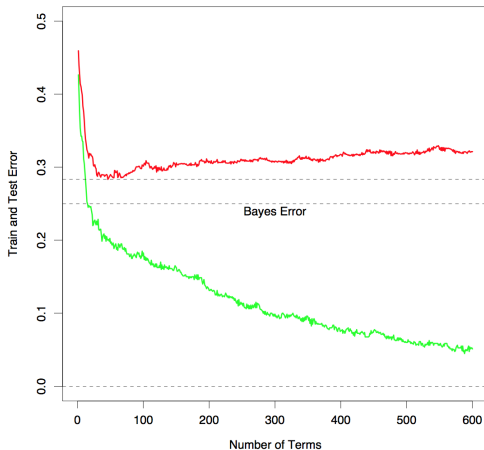


Figure – “Gaussians” in  $\mathbb{R}^{10}$ . Bayes error is 25%

## Stumps



“Gaussians” in  $\mathbb{R}^{10}$ . Bayes error is 25%. Here the test error does increase, but quite slowly

[Video: AdaBoost in Action]

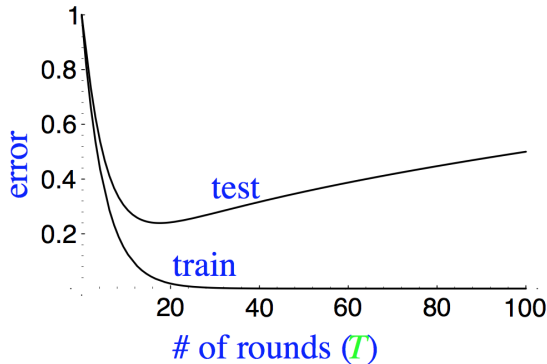
<https://www.youtube.com/watch?v=k4G2VCu0MMg>



- **Base Learners:** decision trees, quite often just decision stumps (trees of depth one)
- Boosting stumps
  - data in  $\mathbb{R}^d$ , e.g.  $d = 2$  ( $\text{height}(\mathbf{x}), \text{weight}(\mathbf{x})$ )
  - associate a stump to each component
  - pre-sort each component:  $O(dm \log m)$
  - at each round, find best component and threshold
  - total complexity:  $O((m \log m)N + mdT)$
  - stumps are not weak learners (XOR problem)
- For SVM boosting usually is not effective
- Additional stopping criterion: error increase on a separate validation set

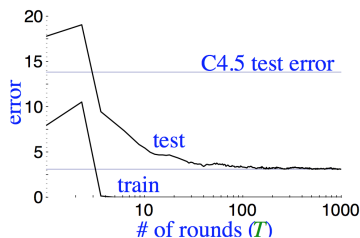
- AdaBoost assigns larger weights to harder examples
- Applications:
  - Detecting mislabeled examples
  - Dealing with noisy data: regularization based on the average weight assigned to a point (soft margin idea for boosting)

## How will Test Error Behave? (A First Guess)



Expect:

- training error to continue to drop (or reach zero)
- test error to increase when  $h_{\text{final}}$  becomes “too complex”
  - “Occam’s razor”
  - overfitting: hard to know when to stop training



(boosting C4.5 on  
"letter" dataset)

## Expect:

- test error does not increase, even after 1000 rounds
  - (total size  $> 2,000,000$  nodes)
- test error continues to drop even after training error is zero!

	# rounds		
	5	100	1000
train error	0.0	0.0	0.0
test error	8.4	3.3	3.1

- Occam's razor wrongly predicts "simpler" rule is better



- **Bias:** not made any worse by bagging multiple hypotheses

$$\begin{aligned}\mathbb{E}_{\mathbf{x},y} \left[ \left( \mathbb{E}_{S_m} \left[ \frac{1}{T} \sum_{t=1}^T h_t(\mathbf{x}|S_m) \right] - \mathbb{E}[y|\mathbf{x}] \right)^2 \right] &= \\ &= \mathbb{E}_{\mathbf{x},y} \left[ \left( \frac{1}{T} \sum_{t=1}^T \mathbb{E}_{S_m} [h_t(\mathbf{x}|S_m)] - \mathbb{E}[y|\mathbf{x}] \right)^2 \right] = \\ &= \mathbb{E}_{\mathbf{x},y} \left[ \left( \mathbb{E}_{S_m} [h(\mathbf{x}|S_m)] - \mathbb{E}[y|\mathbf{x}] \right)^2 \right]\end{aligned}$$

- **Variance:**  $T$  times lower for uncorrelated hypotheses, yet not as much an improvement for highly correlated

$$\begin{aligned}\mathbb{E}_{\mathbf{x},y} \left[ \mathbb{E}_{S_m} \left[ \left( \frac{1}{T} \sum_{t=1}^T h_t(\mathbf{x}|S_m) - \mathbb{E}_{S_m} \left[ \frac{1}{T} \sum_{t=1}^T h_t(\mathbf{x}|S_m) \right] \right)^2 \right] \right] &= \\ &= \frac{1}{T} \mathbb{E}_{\mathbf{x},y} \left[ \mathbb{E}_{S_m} \left[ \left( h(\mathbf{x}|S_m) - \mathbb{E}_{S_m} [h(\mathbf{x}|S_m)] \right)^2 \right] \right] + \\ &\quad + \frac{T(T-1)}{T^2} \mathbb{E}_{\mathbf{x},y} \left[ \mathbb{E}_{S_m} \left[ \left( h(\mathbf{x}|S_m) - \mathbb{E}_{S_m} [h(\mathbf{x}|S_m)] \right) \times \right. \right. \\ &\quad \left. \left. \times \left( \tilde{h}(\mathbf{x}|S_m) - \mathbb{E}_{S_m} [\tilde{h}(\mathbf{x}|S_m)] \right) \right] \right]\end{aligned}$$