

Machine Learning

Bagging and Boosting

DSC 240

Feb 20, 2025

Instructor: Prof. Yu-Xiang Wang

Last Lecture

- Naïve Bayes models
 - Multinomial Naïve Bayes Model for “text classification”
 - Gaussian Naïve Bayes model
- Loss, Risk and Empirical Risk
 - Risk decomposition: Approximation error, Optimization error and generalization

Recap: Two more final remarks for (Multinomial) Naïve Bayes Classifier for text classification

- How does multinomial naïve Bayes classifiers make prediction?
 - New sentence *“You've been back enough. I'll be back.”*

$$\begin{aligned}\hat{y} &= \operatorname{argmax}_{y_k} p(y = y_k | \mathbf{x}) \\ &= \operatorname{argmax}_{y_k} \frac{p(\mathbf{x} | y = y_k) p(y = y_k)}{p(\mathbf{x})} \\ &= \operatorname{argmax}_{y_k} \prod_j p(\mathbf{x}^j | y = y_k) p(y = y_k)\end{aligned}$$

BoW Feature = {“You've”:1, “been”:1, “back”:2, “enough”:1, “I'll”:1, “be”:1};

Recap: Two more final remarks for (Multinomial) Naïve Bayes Classifier for text classification

- Laplace Smoothing

$$\hat{p}(\text{Word}|y) = \frac{\# \text{ of times "Word" occurs in class } y + \alpha}{\text{Total \# of words in class } y \text{ sentences} + \text{Vocabulary size} \cdot \alpha}$$

- Is there a probabilistic interpretation for Laplace smoothing?
 - Short answer: It is equivalent to adding a Dirichlet prior and replace MLE with MAP.
 - Just like L2-regularization = Gaussian prior

Recap: Loss, Empirical Risk, and Risk

- Loss function

$$\ell(h, (x, y))$$

- Empirical Risk function

$$\hat{R}(h, \text{Data}) = \frac{1}{n} \sum_{i=1}^n \ell(h, (x_i, y_i))$$

- (Population) Risk function

$$R(h, \mathcal{D}) = \mathbb{E}_{\mathcal{D}}[\ell(h, (x_i, y_i))]$$

Recap: Four classifiers of interest

- Bayes Optimal classifier: $h_{\text{Bayes}} = \arg \min_h R(h)$

- Optimal classifier
• within hypothesis class $h^* = \arg \min_{h \in \mathcal{H}} R(h)$

- ERM Classifier: $h_{\text{ERM}} = \arg \min_{h \in \mathcal{H}} \hat{R}(h)$

- My classifier: $\hat{h} = \text{My_Learning_Algorithm}(\text{Data})$

Recap: Risk Decomposition

$$\begin{aligned} & \mathbb{E}[R(\hat{h})] - R(h_{\text{Bayes}}) \\ \leq & \underbrace{\mathbb{E}[\hat{R}(\hat{h}) - \hat{R}(h_{\text{ERM}})]}_{\text{Optimization Error}} + \underbrace{R(h^*) - R(h_{\text{Bayes}})}_{\text{Approximation Error}} + \underbrace{\mathbb{E}[R(\hat{h}) - \hat{R}(\hat{h})]}_{\text{Generalization Error}} \end{aligned}$$

How close am I from minimizing the empirical risk?

How much worse the best “representable” classifier is from the best classifier out there.

How different the empirical risk of my classifier is from its population risk?

Make sure you understand what each kind of error means!

Recap: Machine learning can be viewed as a collection of techniques in minimizing the three types of errors

	Optimization error	Generalization Error	Approximation Error
Definition	$\hat{R}(\hat{h}) - \hat{R}(h_{\text{ERM}})$	$R(\hat{h}) - \hat{R}(\hat{h})$	$R(h^*) - R(h_{\text{Bayes}})$
Challenges	<ul style="list-style-type: none"> Finding ERM for some loss functions is NP-Hard. Efficiency isn't enough. Need to be scalable. 	<ul style="list-style-type: none"> We do not observe Risk! Don't have infinite data. Large generalization error \Leftrightarrow Overfitting 	<ul style="list-style-type: none"> Don't know data distribution. No knowledge of Bayes optimal classifier. Large approx. error \Leftrightarrow Underfitting!
What we have learned to address these challenges?	"Just-relax" Surrogate loss, Gradient Descent, SGD	Holdout, Cross-Validation Regularization Statistical learning theory	Better features More flexible decision boundaries Better probabilistic models But how to minimize approx. error automatically?

Today

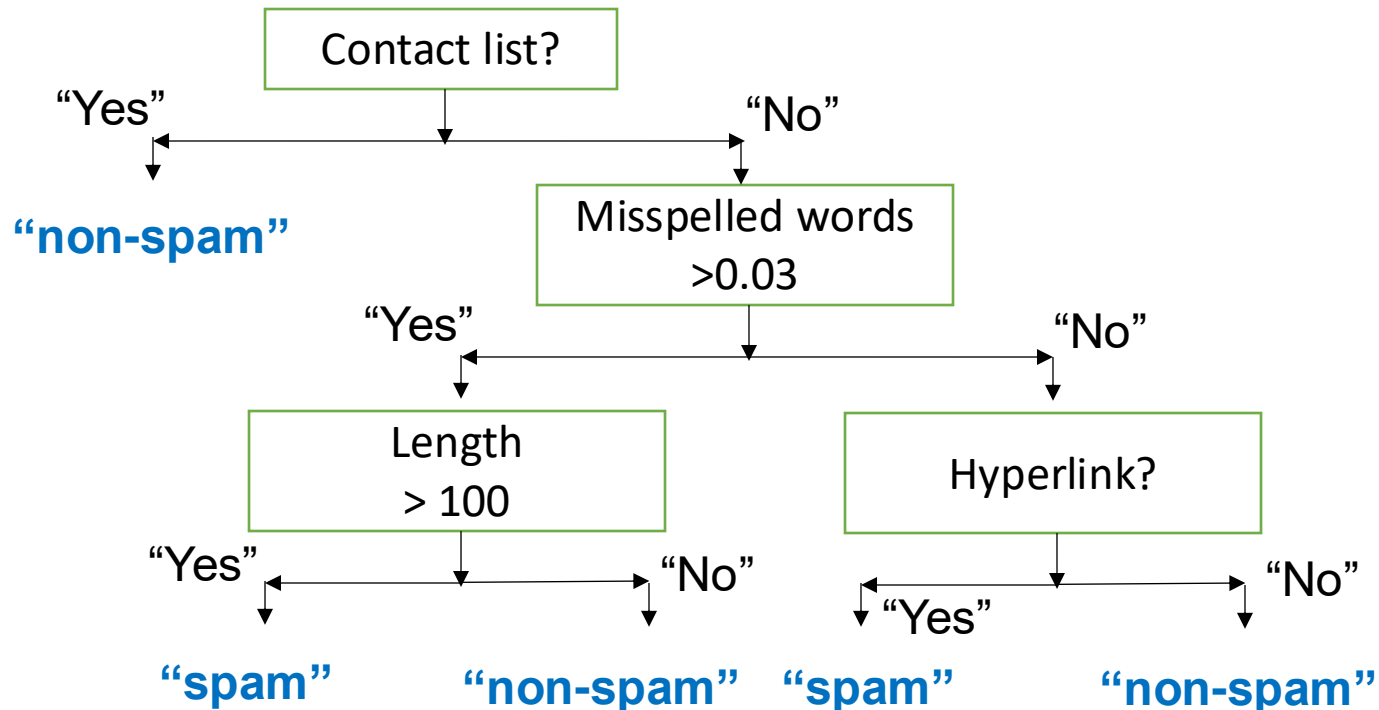
- Start approaches that improve “approximation error”
- Random Forest
- Boosting

Recall: Decision Tree Classifiers

- Input features:

Contains hyperlinks ↓ 1	0	Proportion of misspelled words ↓ 0.0375	80
-------------------------------	---	---	----
- Labels: {non-spam:0, spam:1}

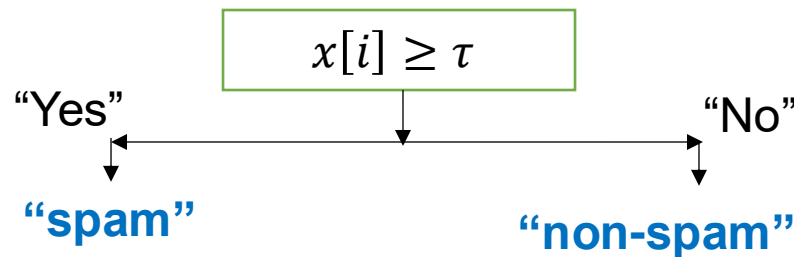
↑	Whether the contact list	↑	Length of the message
---	--------------------------	---	-----------------------



Let's consider a **weak learner** --- e.g., a “Decision Stump” classifier.

- A decision stump classifiers work as follows
 - Take exactly one of the features.
 - Then threshold it.

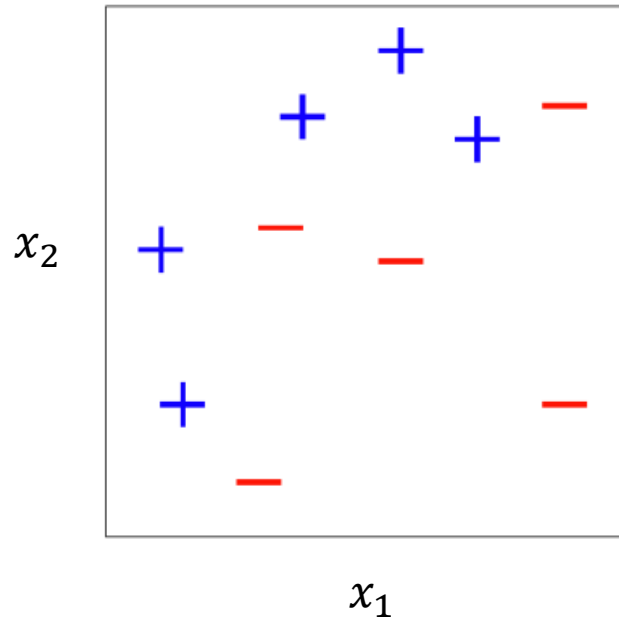
- Example:



- Parameters of the “**decision stump**” classifier
 - i ----- Which feature / coordinate to use?
 - τ ----- Which threshold
 - Leaf labels ----- Assign “spam” to “Yes” or “No”

***Note that it is efficient to solve ERM for decision stump.**

3 min Exercise: Let's work out the decision boundary of a decision stump classifier



- What is the optimal decision stump if x_1 is used? What's the error rate?
- What is the optimal decision stump if x_2 is used? What's the error rate?

Voting classifiers

- Take weak learner $h_1, h_2, h_3, \dots, h_N: \mathcal{X} \rightarrow \mathcal{Y}$

- Voting classifier:

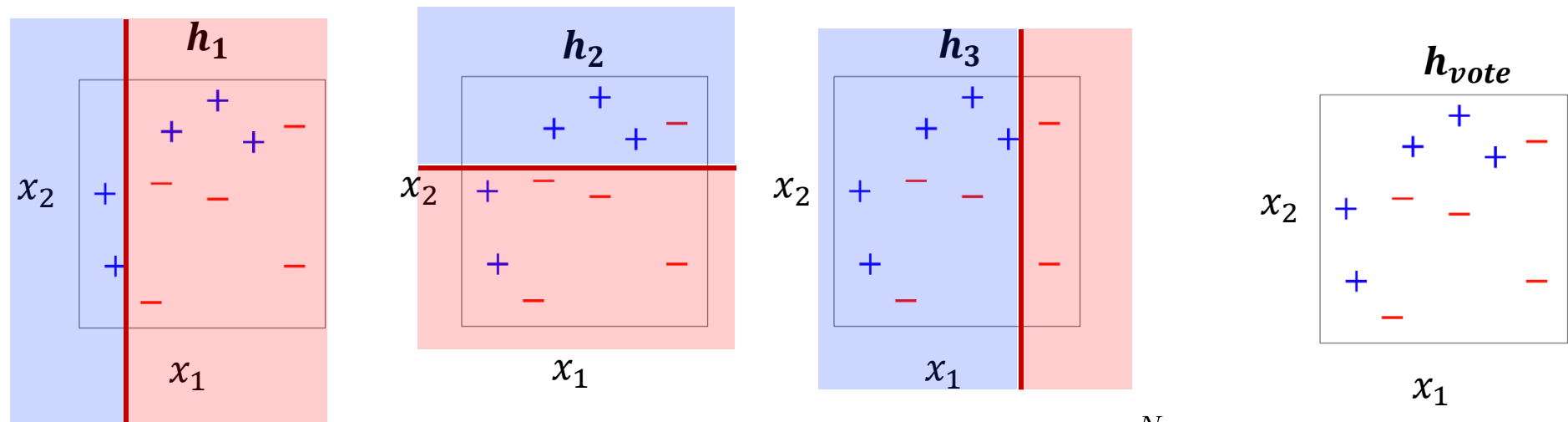
- For classification problems (output space is discrete)

$$h_{\text{vote}}(x) = \arg \max_{y \in \mathcal{Y}} \frac{1}{N} \sum_{i=1}^N \alpha_i \mathbf{1}(h_i(x) = y)$$

- For regression problems (output space is continuous)

$$h_{\text{vote}}(x) = \frac{1}{N} \sum_{i=1}^N \alpha_i h_i(x)$$

3 min Exercise: Let's work out an example!



$$\alpha_1 = \alpha_2 = \alpha_3 = 1.0 \quad h_{\text{vote}}(x) = \arg \max_{y \in \mathcal{Y}} \frac{1}{N} \sum_{i=1}^N \alpha_i \mathbf{1}(h_i(x) = y)$$

1. Circle mis-classified examples.
2. Draw the decision boundary of the voting classifier.

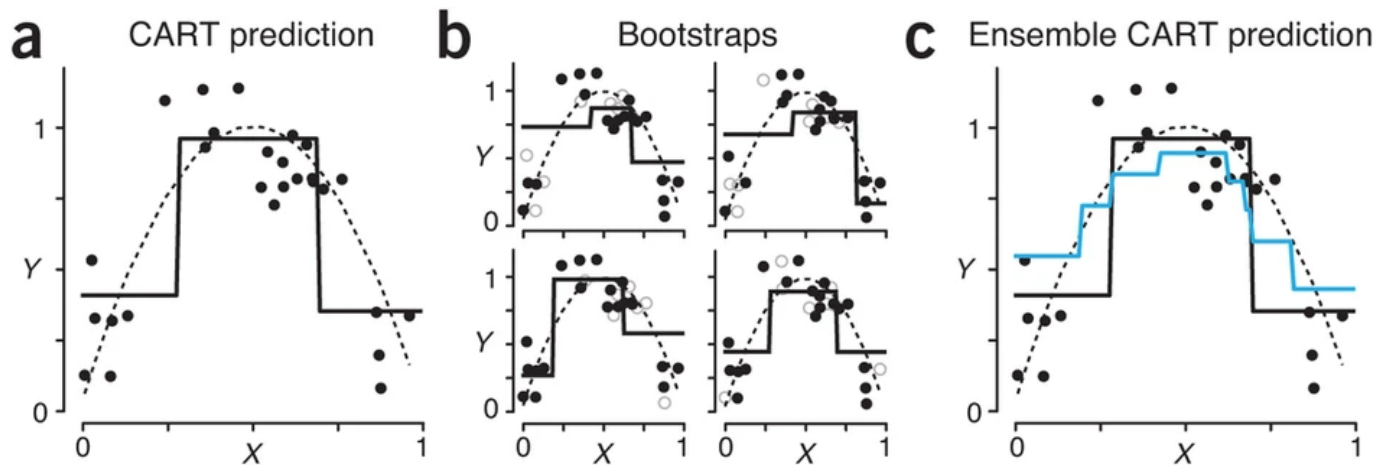
Bagging and **Random Forest Classifier** are both (unweighted) voting classifiers (Breiman, 2001)

- **Bagging (Bootstrapped Aggregation):**
 - Randomly resample data (can be bootstrap sampling or random subsampling)
 - For each sample, fit a weak learner
 - Return a voting classifier with unit weight
- **Random Forest:**
 - Randomly sample subset of the data, randomly sample subset of features
 - Fit a weak learner (typically decision trees) to each sample
 - Return a voting classifier with unit weight

Example: Curve Fitting with Bagging!

Figure 1: Bagging applied to regression using a regression tree.

From: [Ensemble methods: bagging and random forests](#)

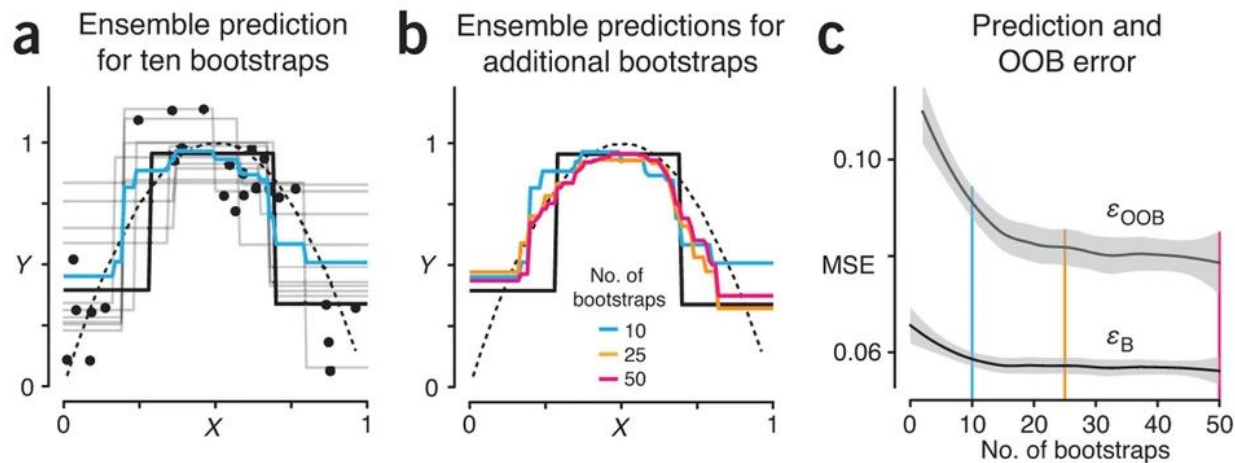


(a) A sample of size $n = 30$ generated from a parabola (dashed line) with added noise and the associated fit from a regression tree (solid black line). (b) Four different bootstrap samples from the sample in (a) and their corresponding regression tree predictions. Solid points are in the bootstrap sample, and some are represented more than once. Hollow points are not in the bootstrap sample and are called out-of-bag (OOB) points. (c) Ensemble regression (blue line) formed by averaging bootstrap regressions in (b). The original regression tree fit from (a) is also shown.

Bagging helps to improved both training error and test error.

Figure 2: Ensemble regressions improve in quality, up to a point, as the number of bootstraps is increased.

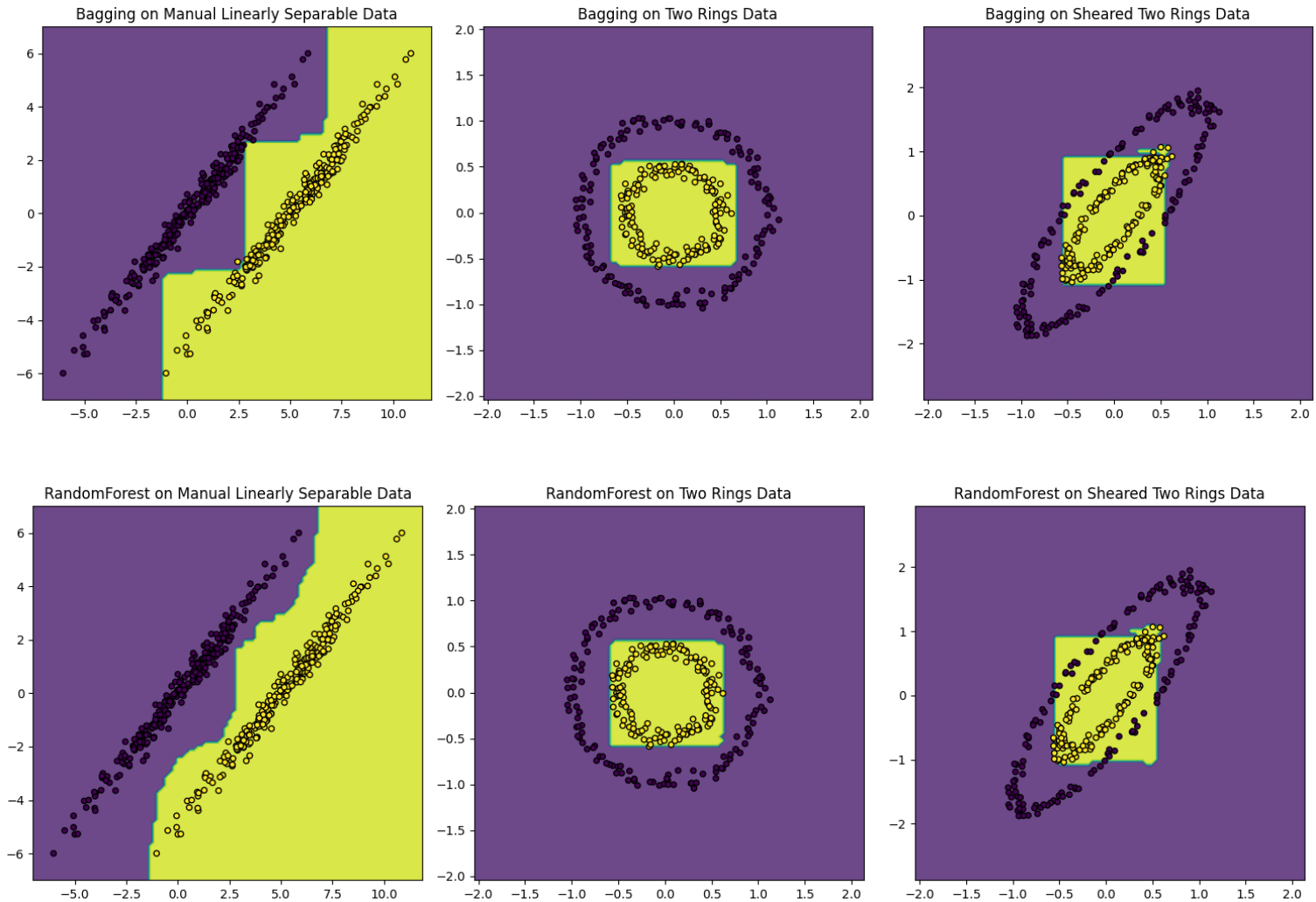
From: [Ensemble methods: bagging and random forests](#)



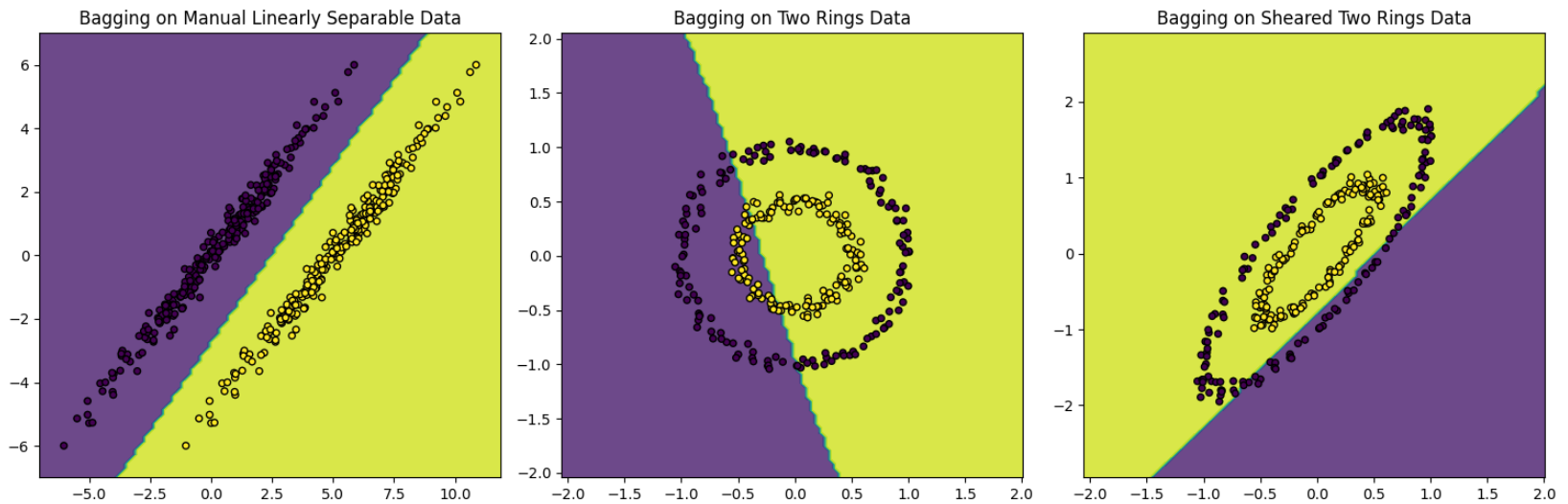
(a) The consensus regression (blue line) for ten bootstrap iterations (gray lines) for data in Figure 1. (b) Ensemble regressions for 10, 25 and 50 bootstrap iterations. (c) The bagged and OOB errors (ϵ_B , ϵ_{OOB}) as a function of the number of bootstraps. The curve is fit to ten simulations at each bootstrap level using locally weighted smoothing. The gray band is the fit's 95% confidence interval.

From paper: <https://www.nature.com/articles/nmeth.4438>

Example: Decision boundary learned by Bagging and RandomForest with DecisionTrees (Depth 5, number of trees = 1000).



Example: Bagging with linear logistic regression



Think about why bagging a linear score function doesn't work.

Boosting (Schapire'89)

(Next few lecture slides borrowed from Aarti Singh.)

- “*The Strength of Weak Learners*” --- can we construct a strong learner (e.g., accuracy 99%) using an ensemble of weak learners (accuracy 51%)? Answer is surprisingly positive!

For each iteration t

1. Come up with a set of weights D_1, \dots, D_n for each training example (based on how incorrectly it was classified by the current classifier h_{vote})
2. Fit a weak learner $h_t = \operatorname{argmin}_h \frac{1}{n} \sum_i D_i \mathbf{1}(h(x_i) \neq y_i)$
3. Figure out a learning rate α_t
4. Update $h_{vote}(x) = \operatorname{sign}(\sum_{j=1}^t \alpha_j h_j(x))$

Theoretically interesting: equivalence between weak and strong learnability.
Practically a very powerful algorithm --- almost the best you can get.

How to come up with the weights?

- AdaBoost (Freund & Schapire'95)

Given: $(x_1, y_1), \dots, (x_m, y_m)$ where $x_i \in X, y_i \in Y = \{-1, +1\}$

Initialize $D_1(i) = 1/m$. **Initially equal weights**

For $t = 1, \dots, T$:

- Train weak learner using distribution D_t . **Naïve bayes, decision stump**

- Get weak classifier $h_t : X \rightarrow \mathbb{R}$.

- Choose $\alpha_t \in \mathbb{R}$. **Magic (+ve)**

- Update:

$$D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

**Increase weight
if wrong on pt i
 $y_i h_t(x_i) = -1 < 0$**

where Z_t is a normalization factor

$$Z_t = \sum_{i=1}^m D_t(i) \exp(-\alpha_t y_i h_t(x_i))$$

**Weights for all
pts must sum to 1
 $\sum_t D_{t+1}(i) = 1$**

How to choose the learning rate?

Weight Update Rule:
$$D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

$$\alpha_t = \frac{1}{2} \ln \left(\frac{1 - \epsilon_t}{\epsilon_t} \right) \quad \text{[Freund \& Schapire'95]}$$

Weighted training error

$$\epsilon_t = P_{i \sim D_t(i)} [h_t(\mathbf{x}^i) \neq y^i] = \sum_{i=1}^m D_t(i) \underbrace{\delta(h_t(x_i) \neq y_i)}_{\text{Does } h_t \text{ get } i^{\text{th}} \text{ point wrong}}$$

$\epsilon_t = 0$ if h_t perfectly classifies all weighted data pts

$\epsilon_t = 1$ if h_t perfectly wrong $\Rightarrow -h_t$ perfectly right

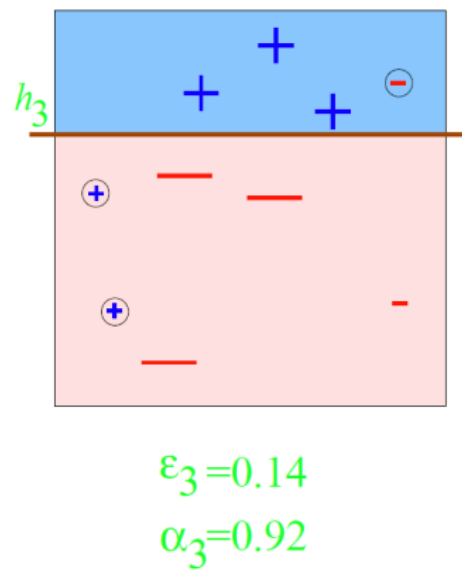
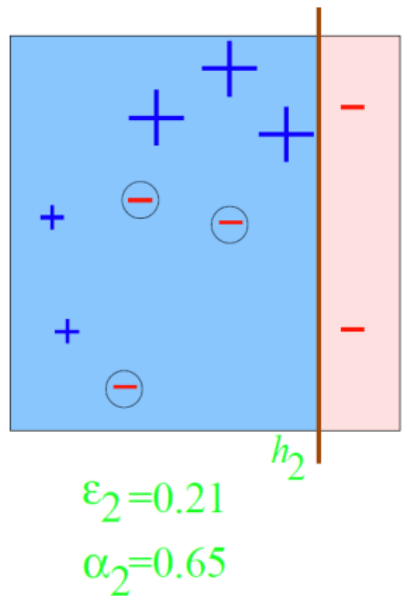
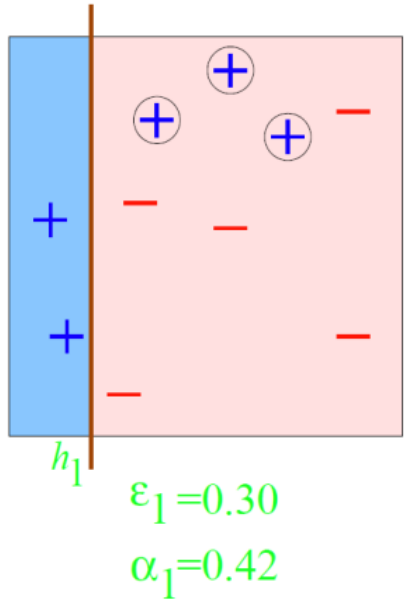
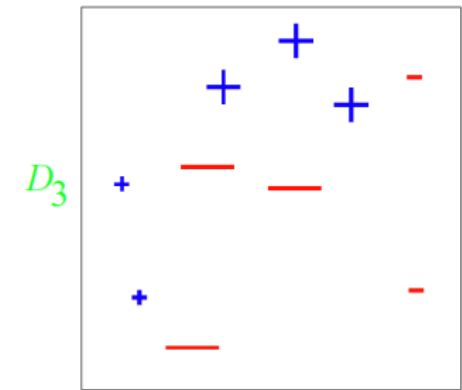
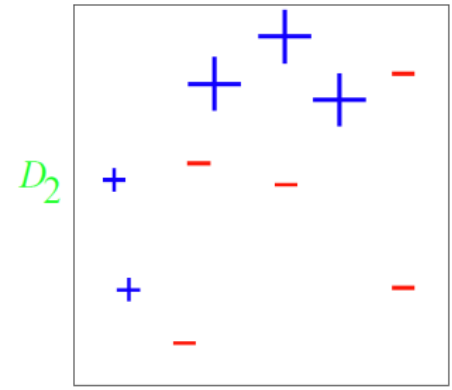
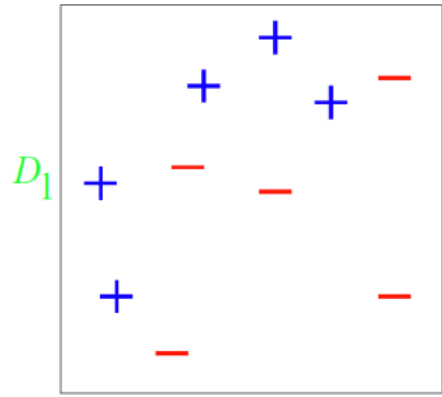
$\epsilon_t = 0.5$

$\alpha_t = \infty$

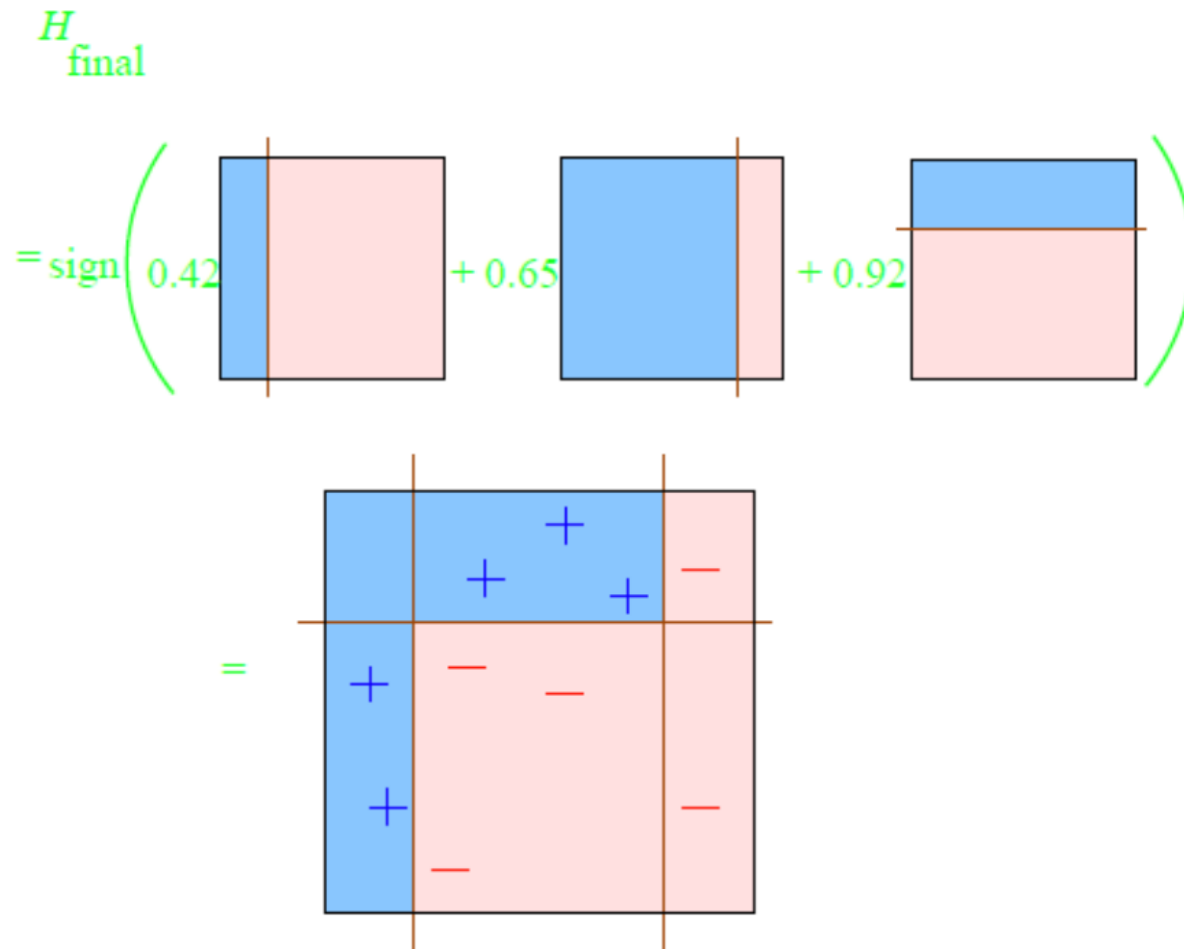
$\alpha_t = -\infty$

$\alpha_t = 0$

Example of AdaBoost with Decision Stumps

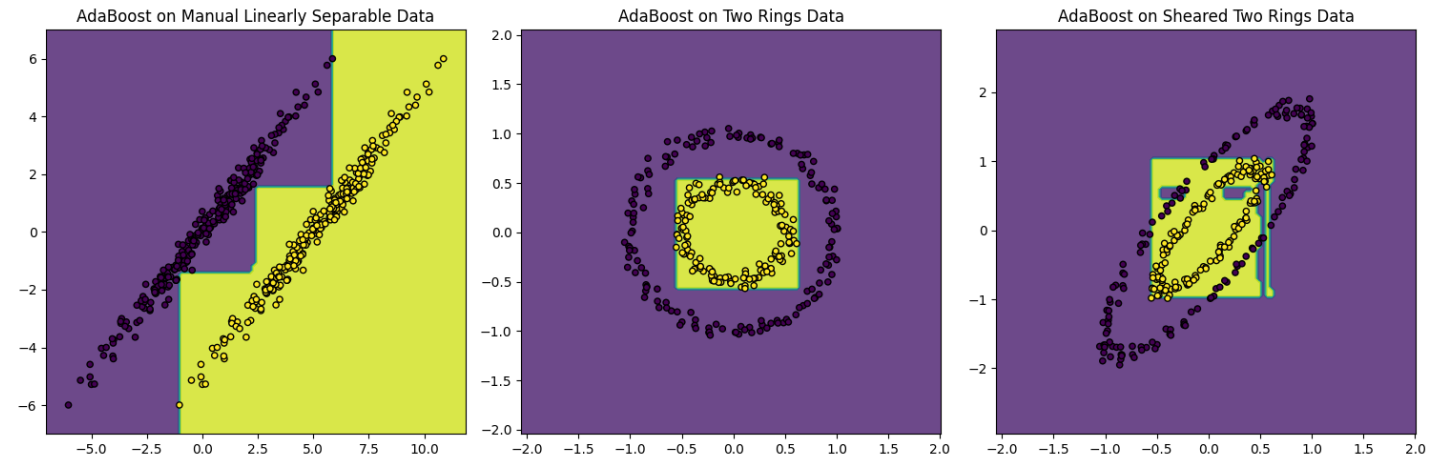


Final classifier from

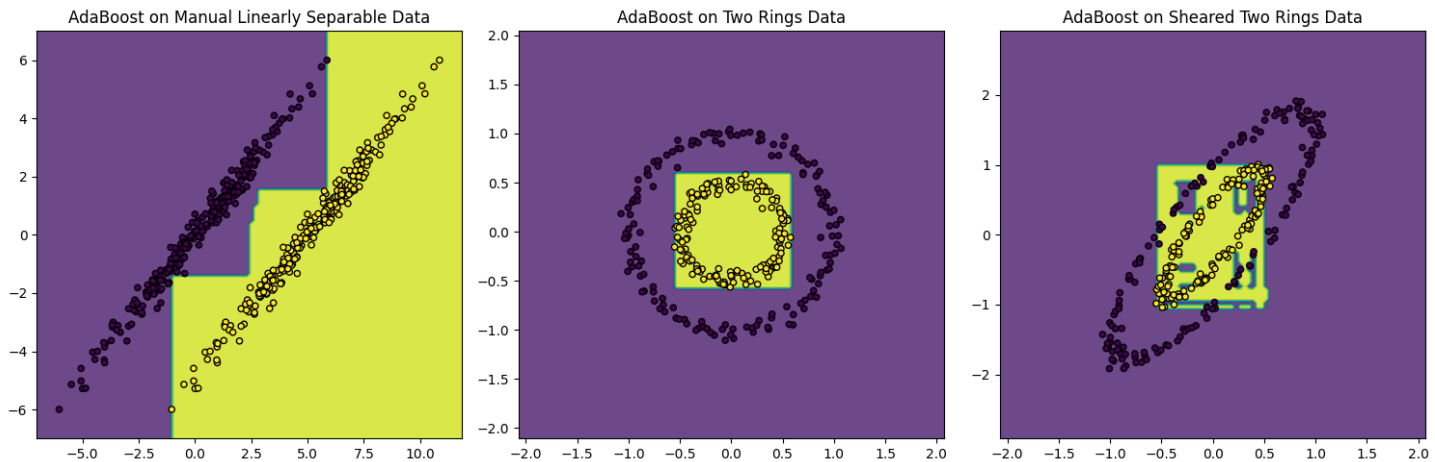


AdaBoost on our three examples

- With Decision Stumps, 100 trees

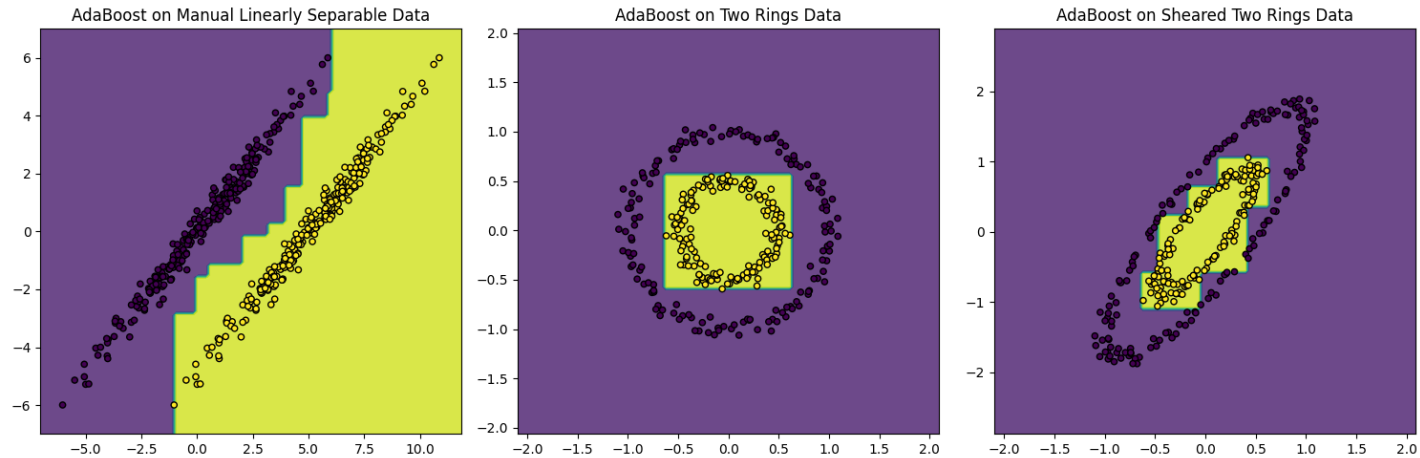


- With Decision Stumps, 200 Trees



AdaBoost on our three examples

- With Decision Trees with Depth 2, 20 trees



- 'With Decision Trees with Depth 5, 20 trees

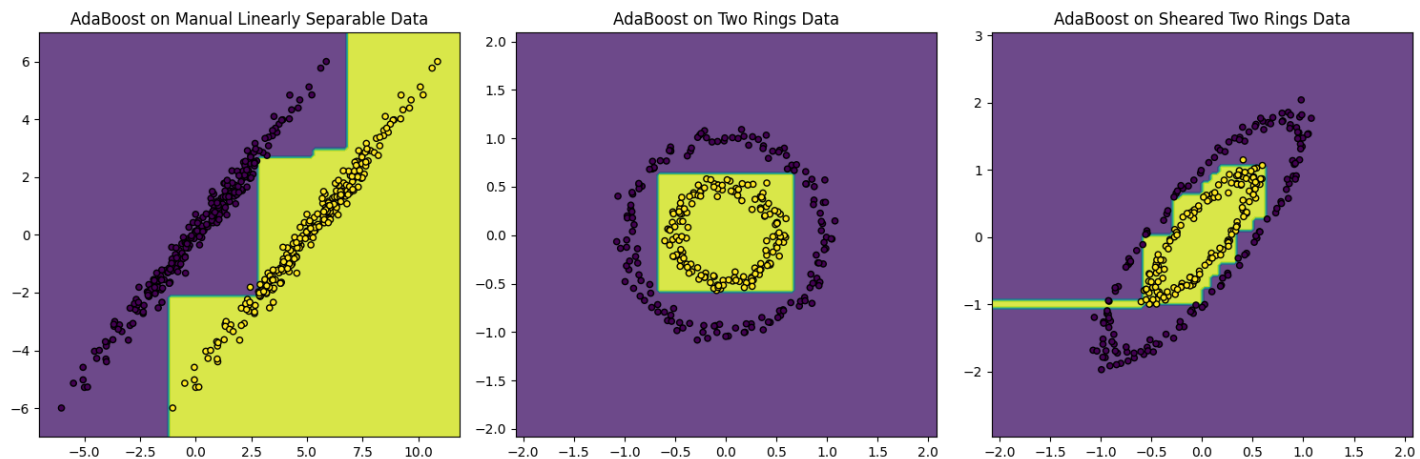
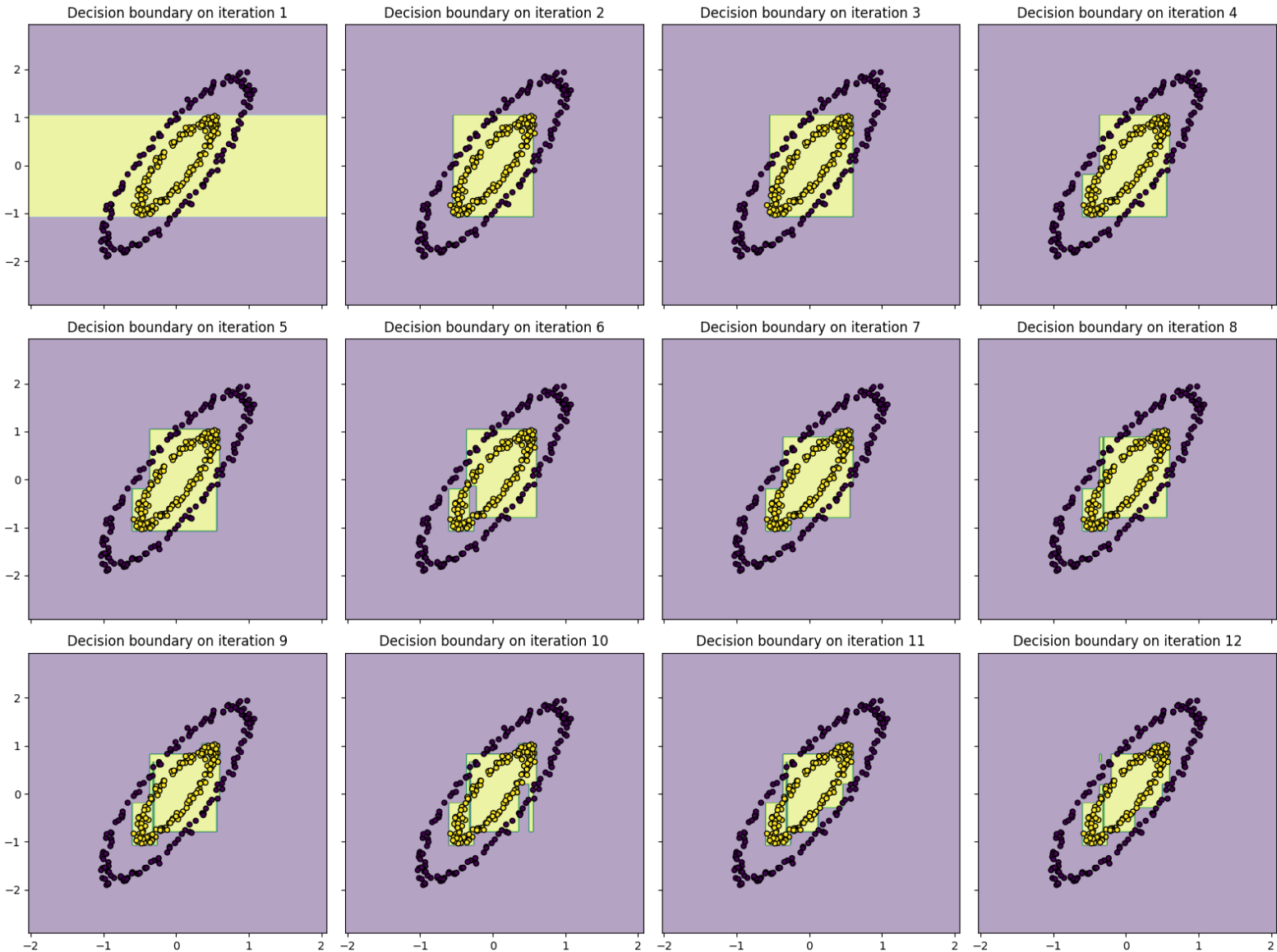
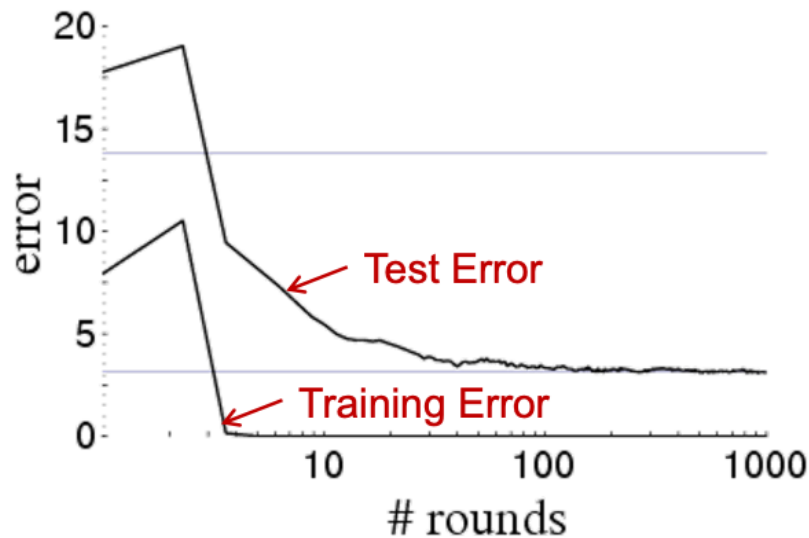


Illustration of the iterations of AdaBoost (Base learner: Depth 2 Decision tree)



Boosting algorithms often do not overfit.
Example on Digit classification original paper.

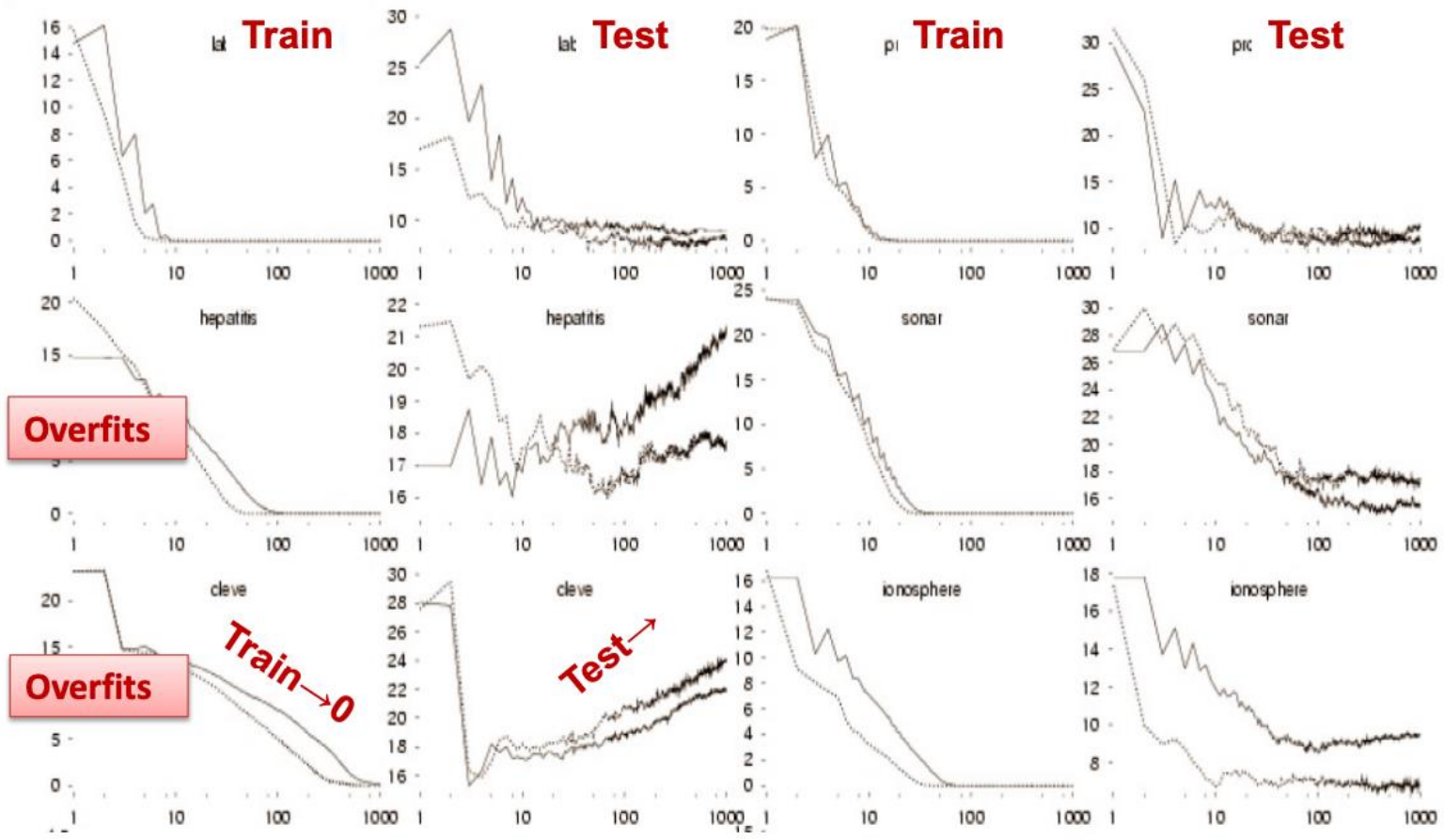
[Schapire, 1989]



- Boosting often,
 - Robust to overfitting
 - Test set error decreases even after training error is zero
- If classes are well-separated, subsequent weak learners agree and hence more rounds does not necessarily imply that final classifier is getting more complex.

It over fits on some datasets but not on others... No overfitting in the “large margin” case, but could overfit when the Bayes optimal has large error.

AdaBoost and AdaBoost.MH on Train (left) and Test (right) data from Irvine repository. [Schapire and Singer, ML 1999]



Final notes about boosting

- Boosting is a favorite among machine learners and data scientists
 - A large majority of Kaggle competitions were won using **XGBoost** --- a computationally efficient distributed implementation of a variant of AdaBoost known as Gradient Boosting.
 - XGBoost: <https://github.com/dmlc/xgboost>
- Boosting can be interpreted as gradient descent
 - Each tree is fitted to best approximate the negative gradient direction
 - Adding tree to the ensemble moves towards that direction.
- Feature learning perspective:
 - Every new tree is a new (greedily selected) feature.
 - The final classifier uses a linear combination of these learned features.

Next Lecture

- Feature expansion and kernels
- Neural Networks