UNIVERSITY AT ALBANY
STATE UNIVERSITY OF NEW YORK

CSI 436/536 (Fall 2024)

Machine Learning

Lecture 15: Decision Tree and Boosting

Chong Liu

Assistant Professor of Computer Science

Nov 5, 2024

Recap: Risk Decomposition

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

Optimization Error

How close am I from minimizing the empirical risk?

Approximation Error

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

Generalization Error

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	$\hat{R}(\hat{h}) - \hat{R}(h_{\mathrm{ERM}})$	$R(\hat{h}) - \hat{R}(\hat{h})$	$R(h^*) - R(h_{\mathrm{Bayes}})$
Challenges	• Finding ERM for some loss functions is NP-Hard. • Efficiency isn't enough. Need to be scalable.	• We do not observe Risk! • Don't have infinite data. • Large generalization error ⇔ Overfitting	• Don't know data distribution. • No knowledge of Bayes optimal classifier. • Large approx. error ⇔ Underfitting!
What we have learned to address these challenges?	"Just-relax" Surrogate loss, Gradient Descent, SGD	Holdout, Cross-Validation Regularization Statistical learning theory *(not covered)*	Better features More flexible decision boundaries Better probabilistic models **But how to minimize approx. error automatically?**

Often there is a tradeoff.
More **flexible** hypothesis class => smaller approximation error
but larger generalization error (more overfitting) and sometimes harder optimization

Recap: Three main approaches for expanding the hypothesis class (systematically minimizing the approx. error)

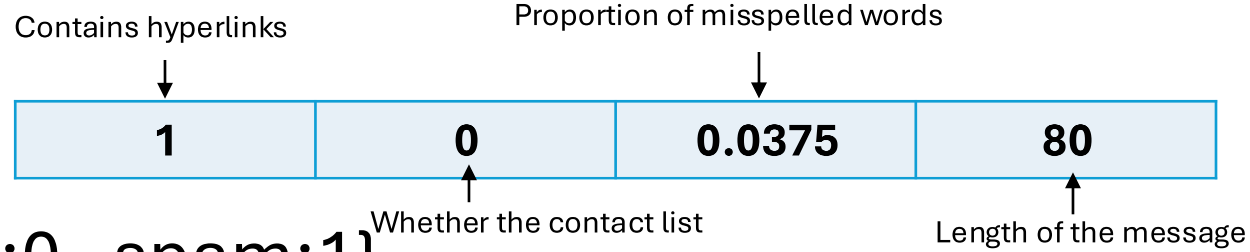
- Boosting and Bagging (Ensemble learning)
  - Combine many weak learners (e.g., decision trees with depth 3) into a strong learner
- Kernel methods (lift features to higher-dimensional space)
  - e.g., adding polynomial expansion, add interaction terms
  - Other nonlinear transformation of the original features
- Deep Learning
  - Train large neural networks using SGD
  - Learn feature representation and classification jointly.

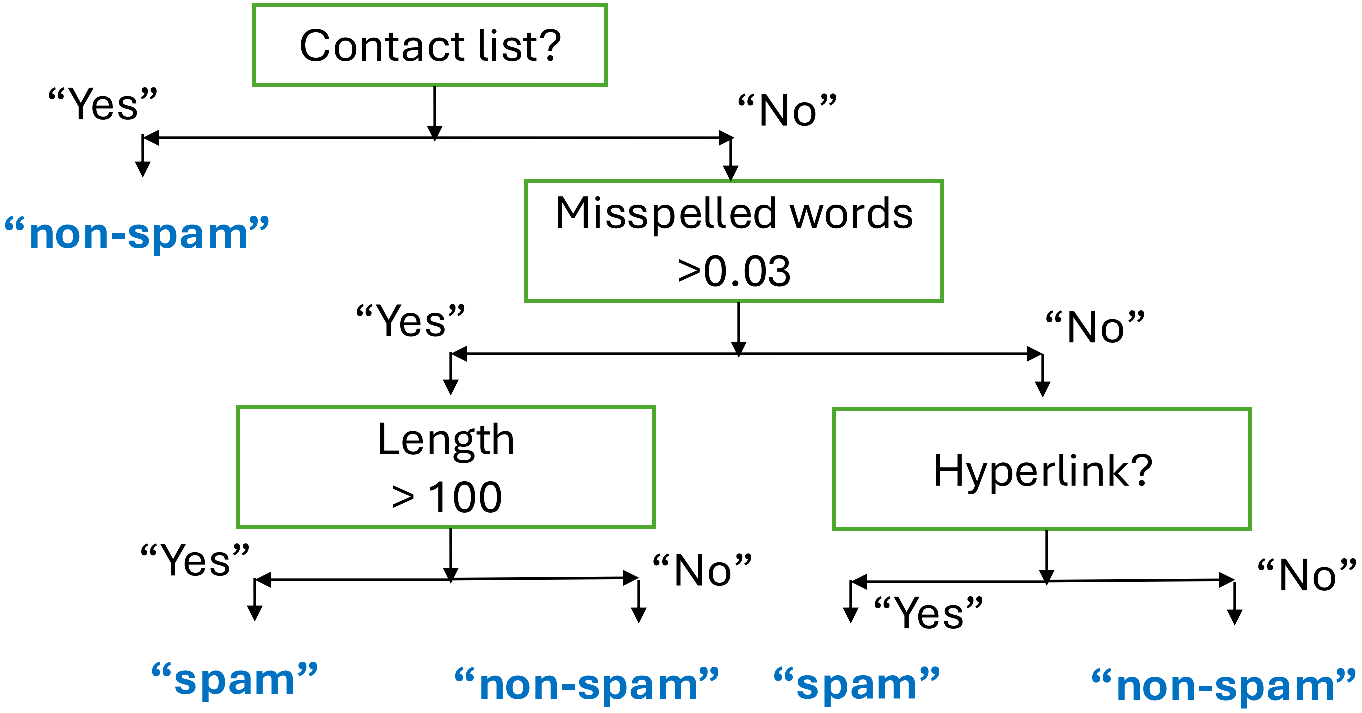
Today

- Review decision trees
- Approaches that minimize “approximation error”
  - Bagging and Random Forest
  - Boosting

Recall: Decision Tree Classifiers

- Input features:
- Labels: {non-spam:0, spam:1}

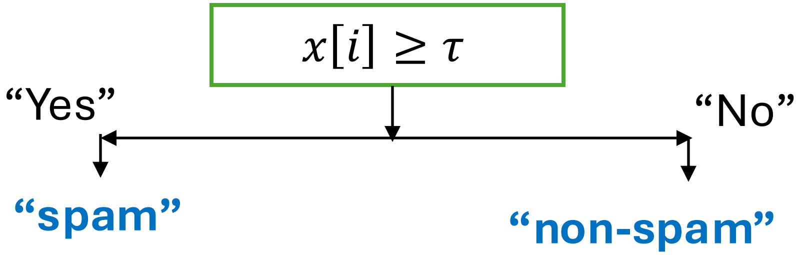




Let's consider a weak learner --- e.g., a "Decision Stump" classifier.

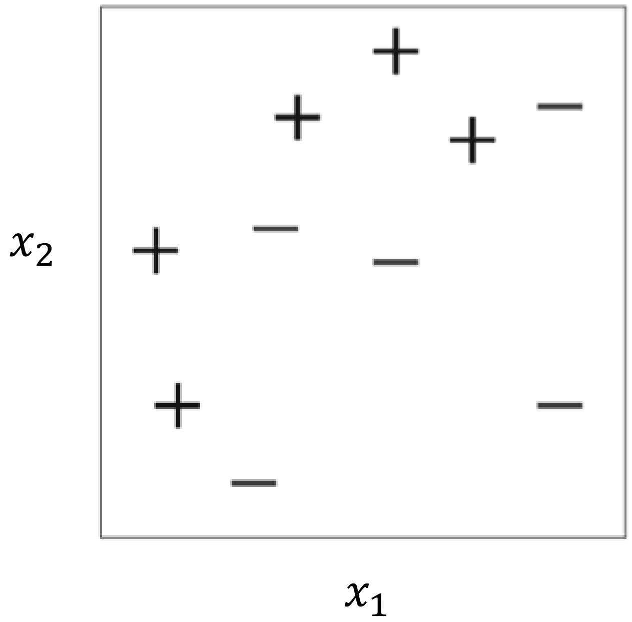
- A decision stump classifiers work as follows
  - Take exactly one of the features.
  - Then threshold it.

- Example:



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

In-class exercise: Let's work out the decision boundary of a decision stump classifier



- What is the optimal decision stump if x1 is used? What's the error rate?
- What is the optimal decision stump if x2 is used? What's the error rate?

Ensemble classifiers

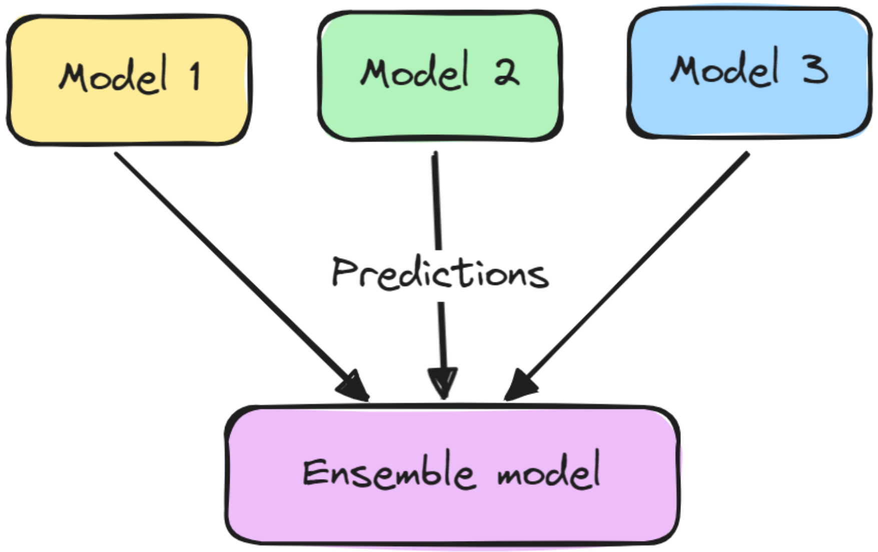
- Take multiple learner $h_1, h_2, h_3, \ldots, h_N$

- Ensemble classifier:
  - For classification problems (output space is discrete)

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

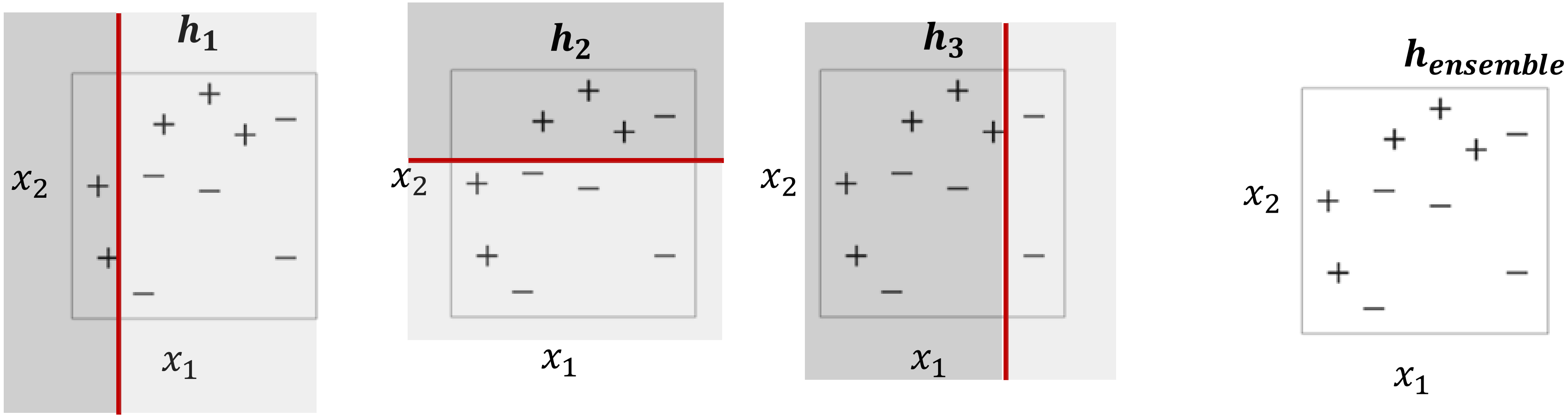
  - For regression problems (output space is continuous)

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



https://www.datacamp.com/tutorial/what-bagging-in-machine-learning-a-guide-with-examples

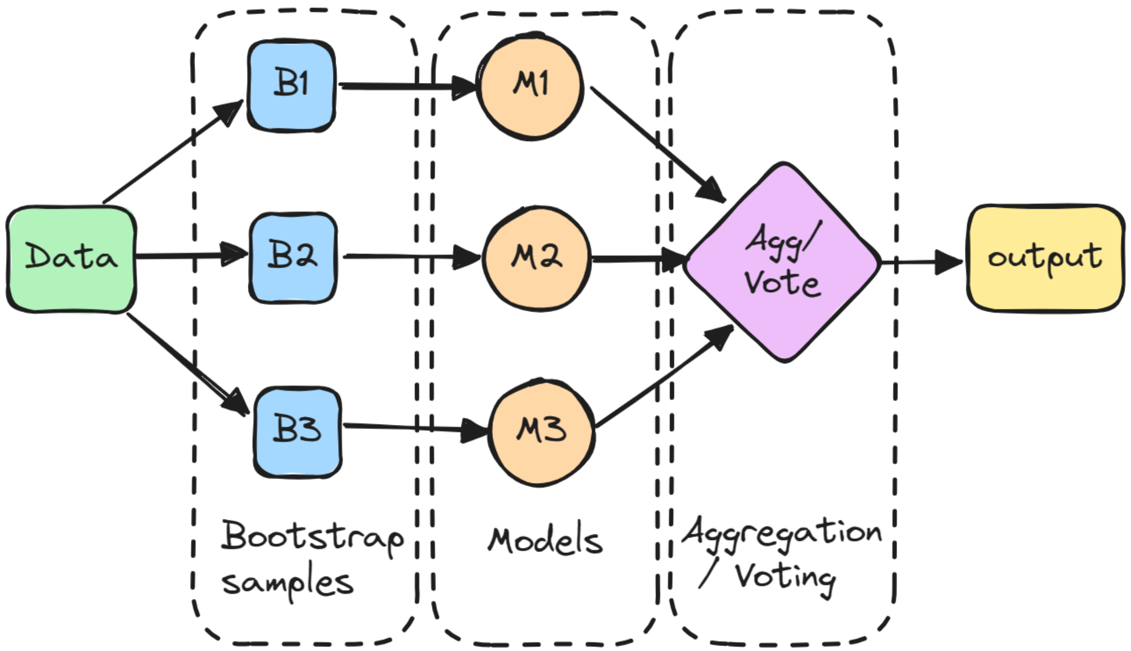
In-class exercise: Let's work out an example!



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

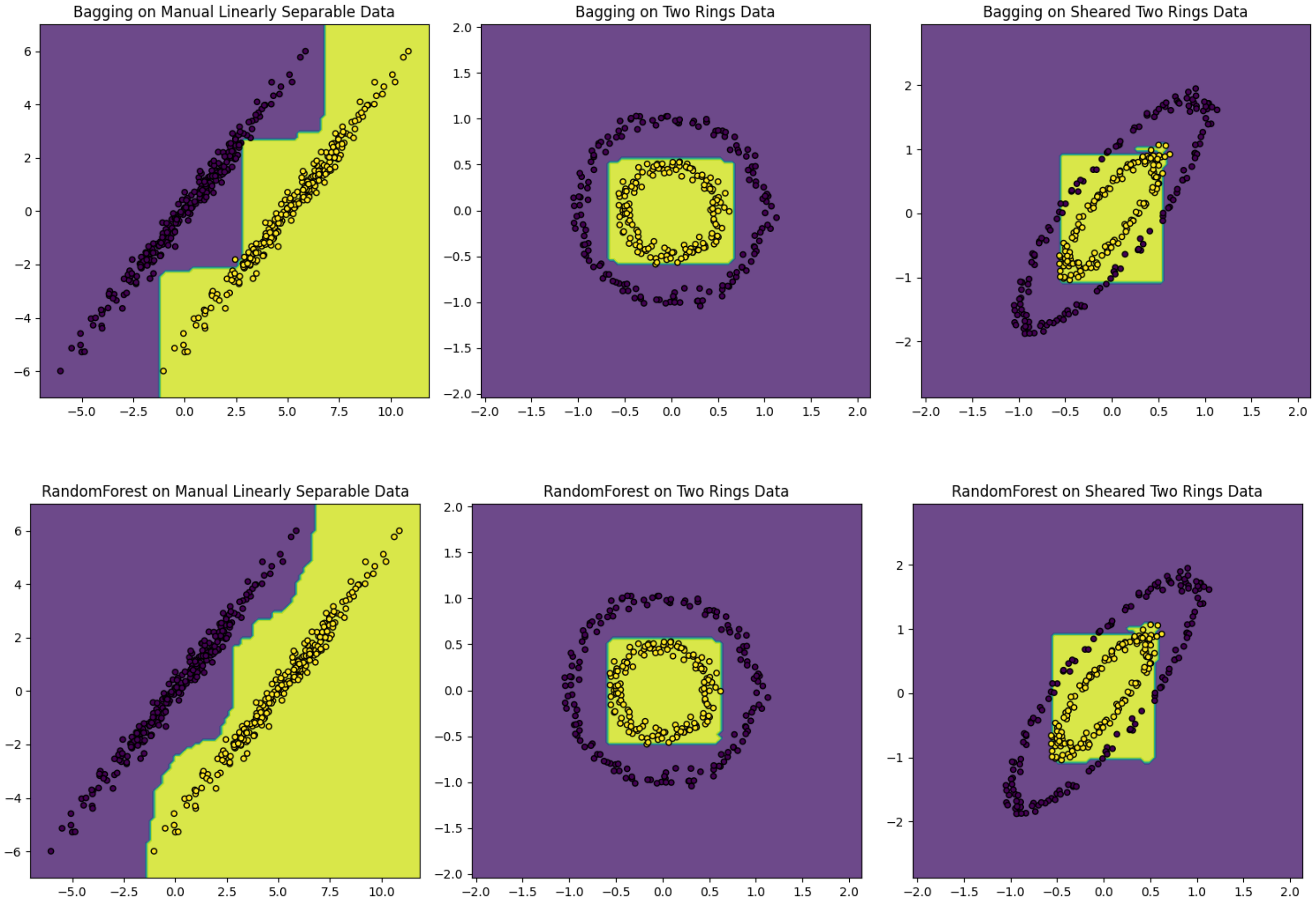
Bagging and Random Forest Classifier are both (unweighted) ensemble classifiers (Breiman, 2001)

- Bagging:
  - Randomly sample subset of data
  - For each sample, fit a learner
  - Return an ensemble classifier
- Random Forest:
  - Randomly sample subset of data and subset of features
  - For each sample, fit a learner (typically decision trees)
  - Return an ensemble classifier



https://www.datacamp.com/tutorial/what-bagging-in-machine-learning-a-guide-with-examples

Example: Decision boundary learned by Bagging and RandomForest with DecisionTrees (Depth 5, n_trees = 1000)



Boosting (Schapire'89)

- "*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 $D_1, \ldots, D_n$ for each training example (based on how incorrectly it was classified by the current classifier $h_{ensemble}$)
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 $h_{ensemble}(x) = \operatorname{sign}\left(\sum_{j=1}^{t} \alpha_j h_j(x)\right)$

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)

Input: Training set $D = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_m, y_m)\}$;
Base learner $\mathcal{L}$;
Number of training rounds T.

Process:

1: $\mathcal{D}_1(\boldsymbol{x}) = 1/m$; Initially all data points have the same weight

2: **for** $t = 1, 2, \ldots, T$ **do**

3: $h_t = \mathcal{L}(D, \mathcal{D}_t)$; Training on data points

4: $\epsilon_t = P_{\boldsymbol{x}\sim\mathcal{D}_t}(h_t(\boldsymbol{x}) \neq f(\boldsymbol{x}))$; Calculate the error

5: **if** $\epsilon_t > 0.5$ **then break** If binary classifier is too bad

6: $\alpha_t = \frac{1}{2}\ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$; Calculate the weight parameter

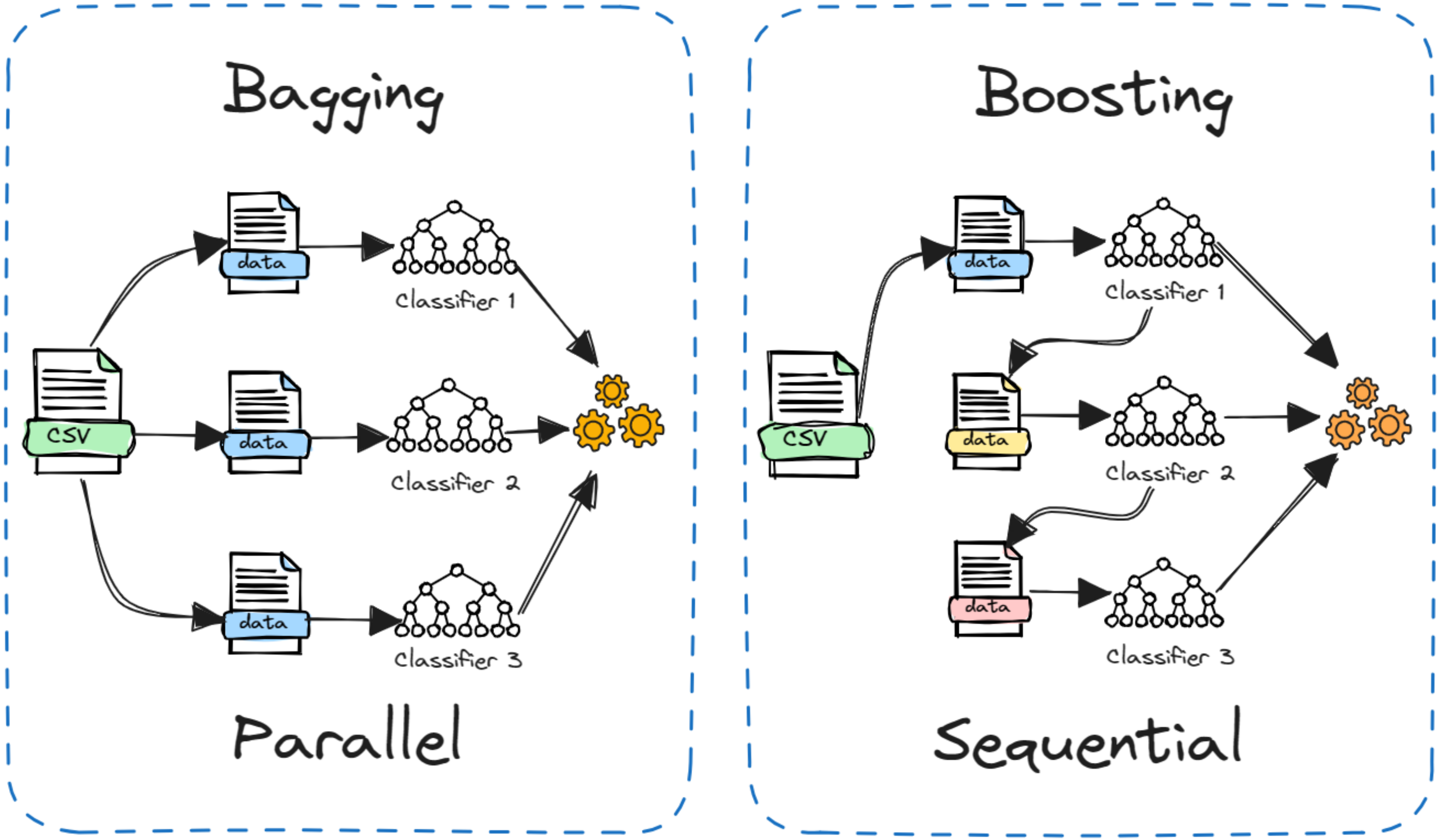
7: $$\mathcal{D}_{t+1}(\boldsymbol{x}) = \frac{\mathcal{D}_t(\boldsymbol{x})}{Z_t} \times \begin{cases} \exp(-\alpha_t), & \text{if } h_t(\boldsymbol{x}) = f(\boldsymbol{x}); \\ \exp(\alpha_t), & \text{if } h_t(\boldsymbol{x}) \neq f(\boldsymbol{x}); \end{cases}$$ Increase the weight of data points that are incorrectly classified; decrease - correct

$$= \frac{\mathcal{D}_t(\boldsymbol{x})\exp(-\alpha_t f(\boldsymbol{x}) h_t(\boldsymbol{x}))}{Z_t};$$

8: **end for**

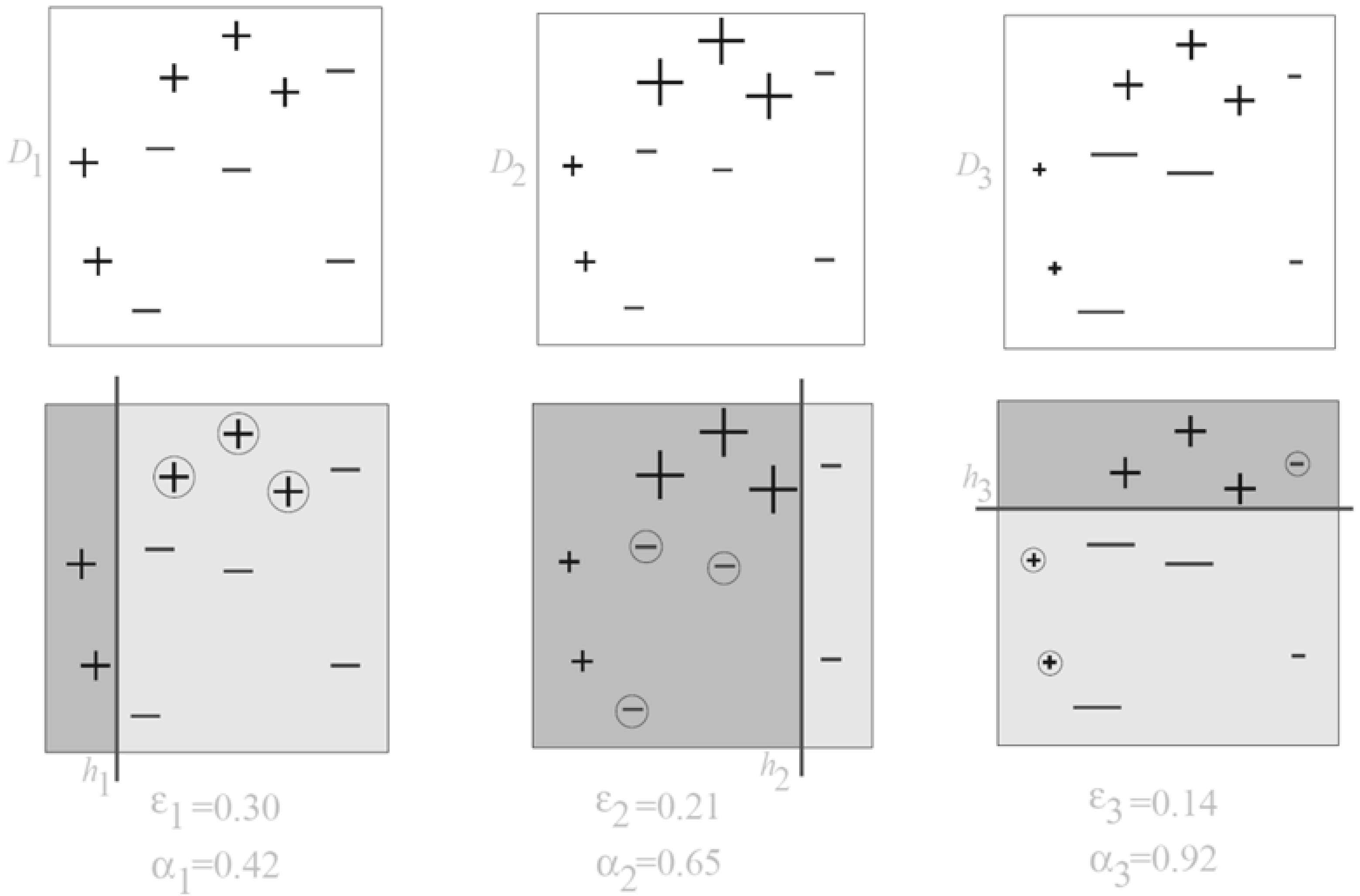
Output: $F(\boldsymbol{x}) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(\boldsymbol{x})\right)$. Output a <u>**weighted**</u> classifier

Bagging and boosting

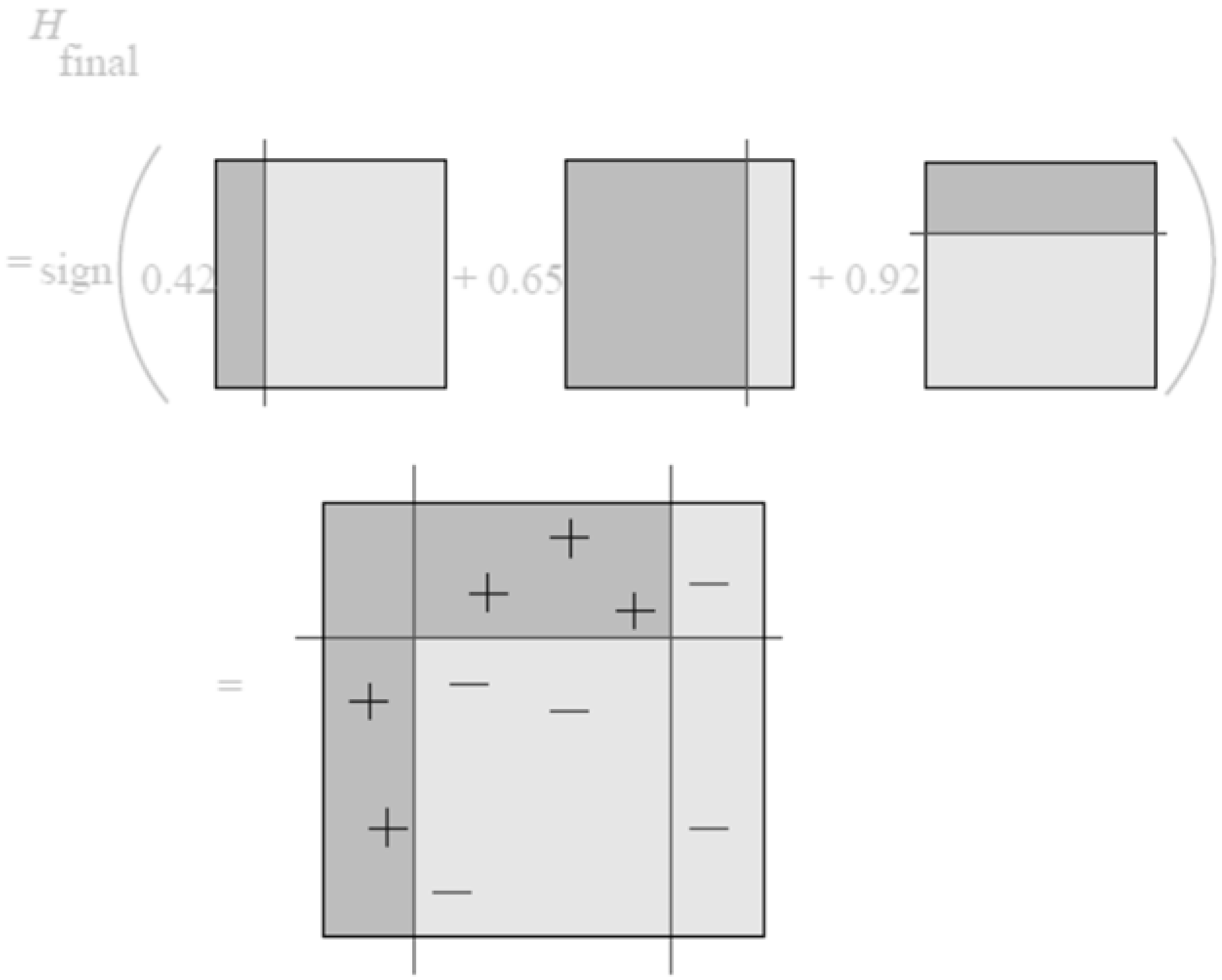


https://www.datacamp.com/tutorial/what-bagging-in-machine-learning-a-guide-with-examples

Example of AdaBoost with Decision Stumps

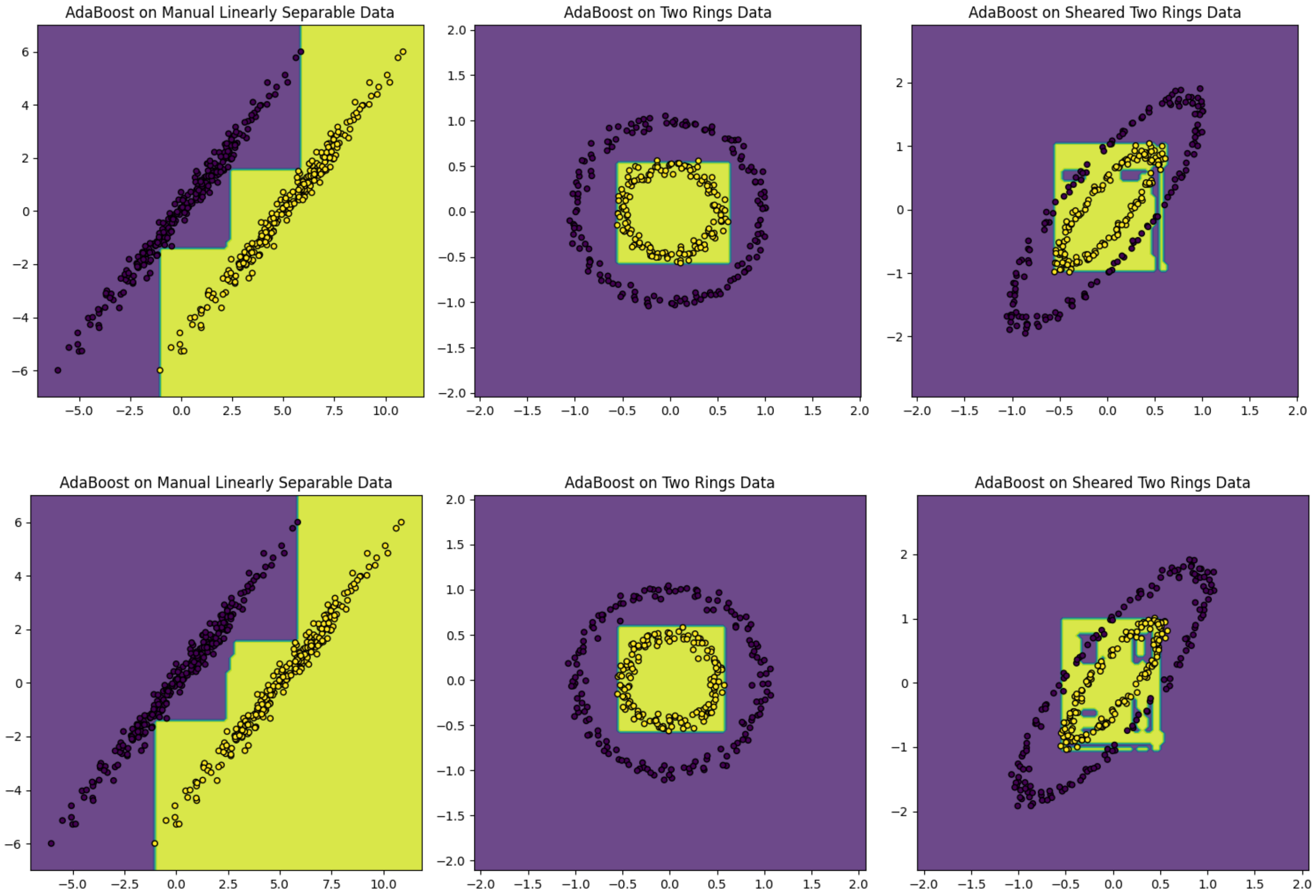


Final classifier from



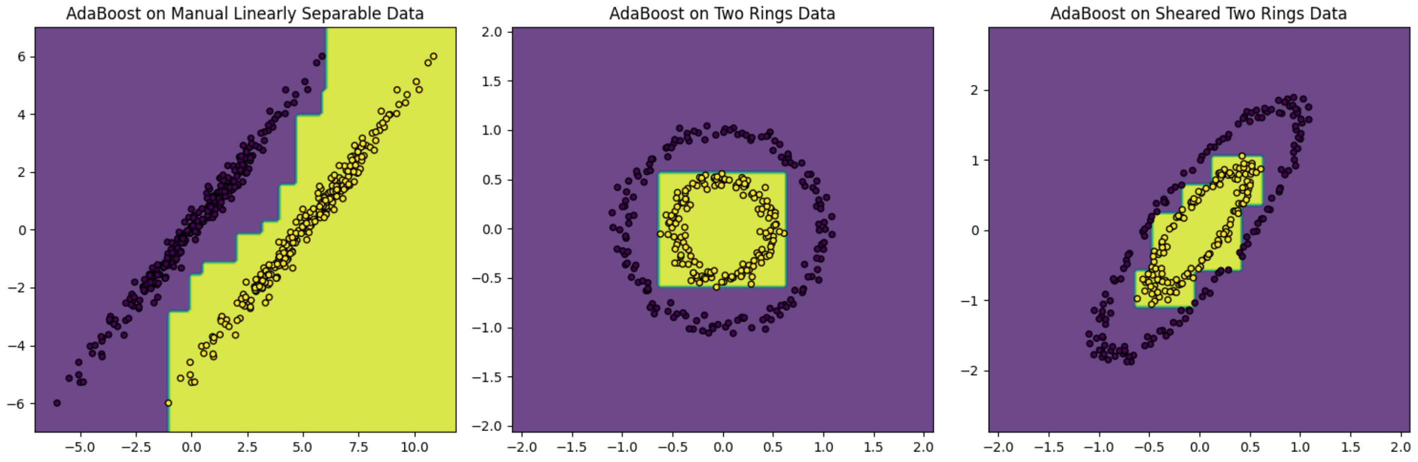
AdaBoost on our three examples

- With Decision Stumps, 100 trees

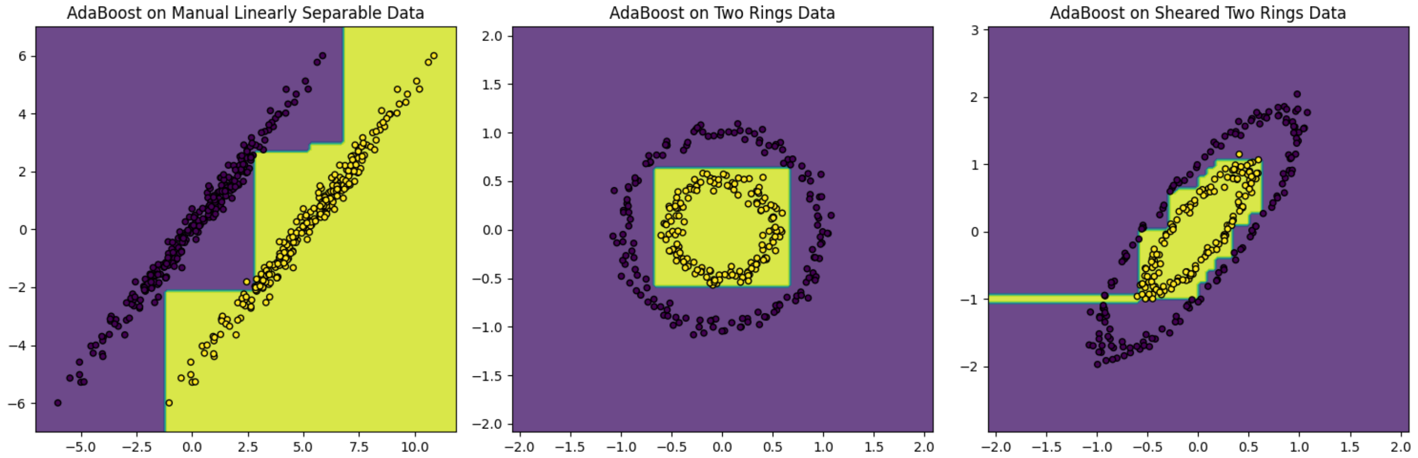


AdaBoost on our three examples

- With Decision Trees with Depth 2, 20 trees



- With Decision Trees with Depth 5, 20 trees



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.