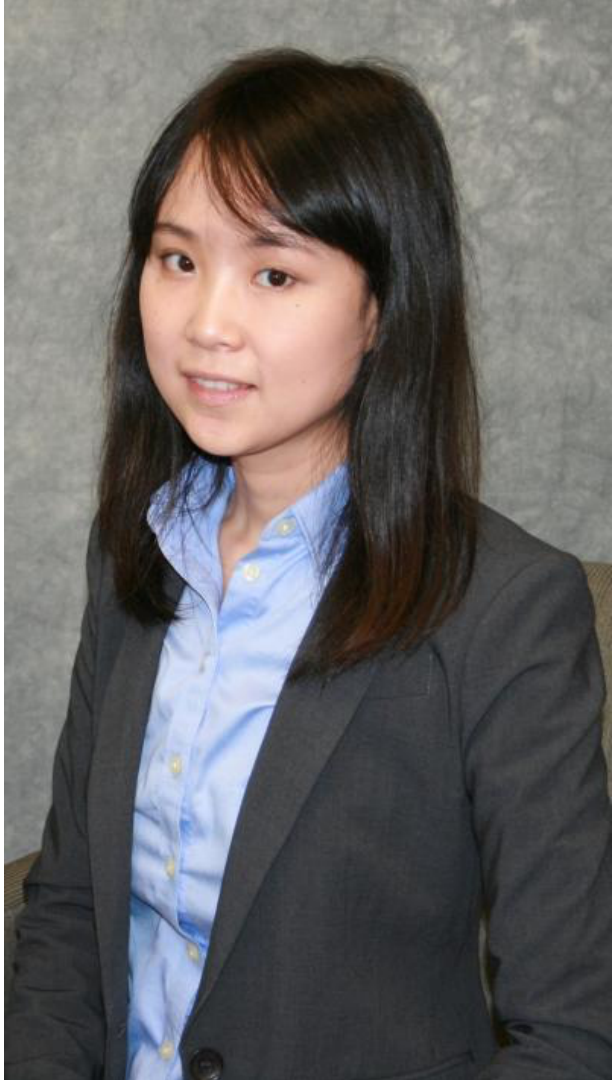
Computational Data Analysis

Machine Learning



Yao Xie, Ph.D.

Associate Professor

Harold R. and Mary Anne Nash Early Career Professor

H. Milton Stewart School of Industrial and Systems Engineering

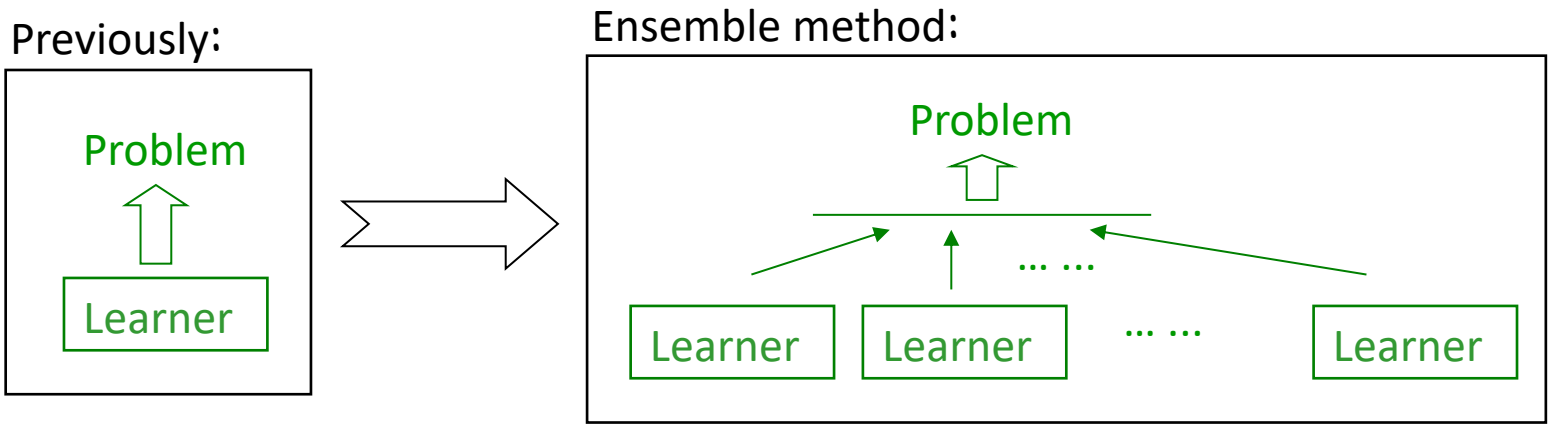
Boosting

Rationale: Combination of methods

- General machine learning tasks, there is no algorithm that is always the most accurate

"Can a set of weak learners create a single strong learner?”

- Kearns and Valiant (1988, 1989)



- Different learners use different
  - Algorithms, Hyperparameters, Representations, Training sets, Subproblems

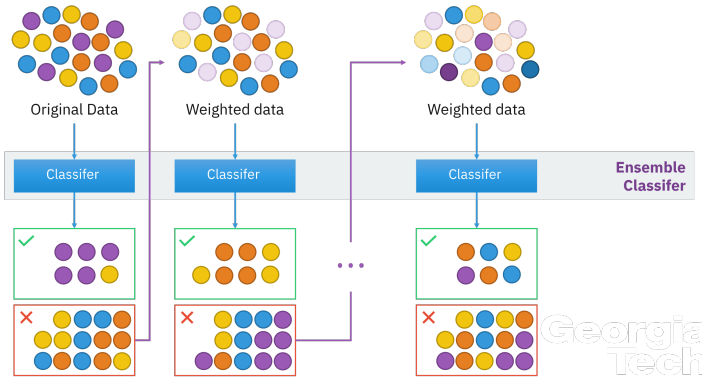
Two main approaches

- Boosting
  - Run weak learner on **weighted** example set
  - Combine weak learners linearly
  - Require knowledge on the performance of weak learner

- Bagging
  - Run weak learners on bootstrap replicates of the training set
  - Average weak leaners
  - Reduces variance

Boosting

- Boosting: general methods of converting weak learner into a more powerful one
  - Each learner is dependent on the previous one and focuses on the previous one's errors
  - Data that are incorrectly predicted in the previous rounds are weighted more heavily when deciding a new learner.
- Questions:
  - How to adjust weights for data?
  - How to combine learners?

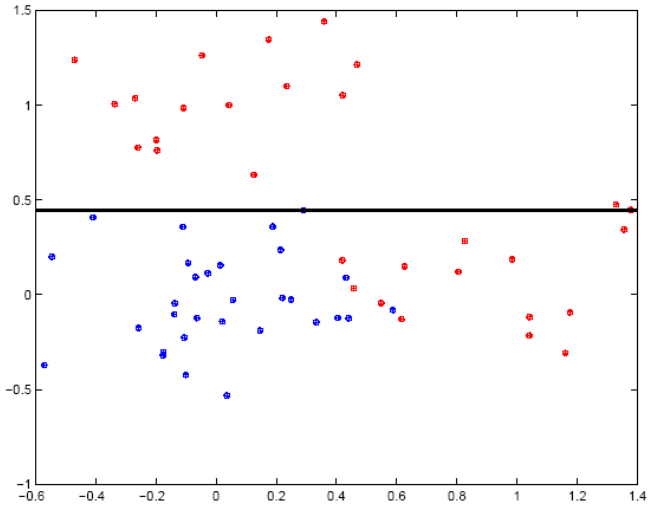


Example of weak learners for classification: Decision stump

- Let $y \in \{\pm 1\}$
- Decision stump for the j-th dimension:

$$h(x; \theta_j) = \text{sign}(w_j x_j + b_j)$$

- Each decision stump only focuses on a single dimension of the feature vector



Boosting setup

- Given a set of base classifiers $\{h_1, h_2, \dots\}$, $h_i: X \to \{1, -1\}$.
- Training data:

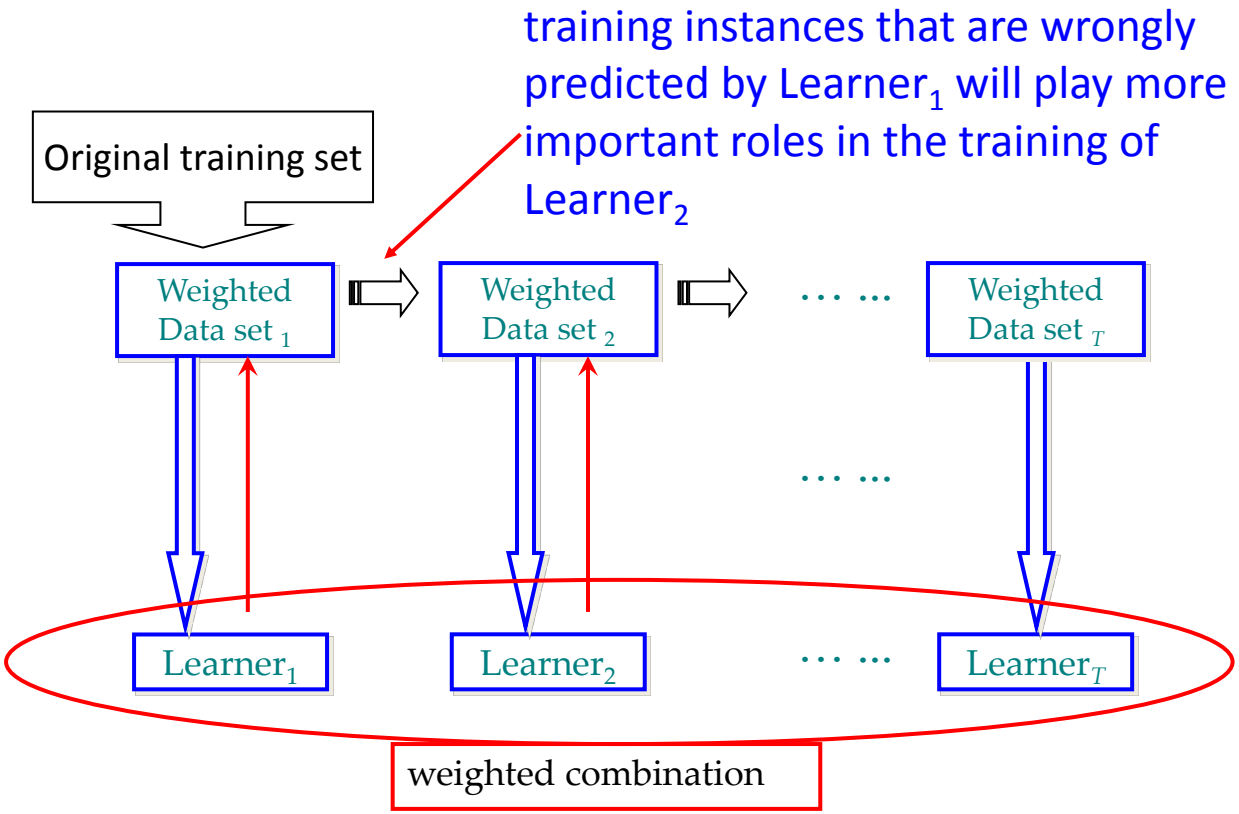
$$(x^i, y^i),\ i = 1, \dots, m, \quad x^i \in X \text{ and } y^i \in \{1, -1\}$$

- Construct
  - a sequence of distributions (weights on data sum up to 1) $D(i), i = 1, \dots m$
  - a sequence $\{\alpha_k\}$ of nonnegative weights of base classifiers
  - Final classifier performs significantly better than any base classifier

AdaBoost (Freund and Schapire)

- AdaBoost is adaptive in the sense that subsequent weak learners are tweaked in favor of those instances misclassified by previous classifiers
- Distribution and weights are adjusted adaptively
- Winner of 2003 Gödel Prize
- There are many variants, here we present one basic version.

Adaboost flow chart



AdaBoost

- Construct $D_t: t = 1, \ldots, T$
- Initialize $D_1(i) = 1/m$
- Given D_t and h_t, decide weights for classifier

$$\epsilon_t = \sum_{i=1}^{m} D_t(i)\mathbb{I}\{y^i \neq h_t(x^i)\}, \qquad \alpha_t = \frac{1}{2}\ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$$

- Update weights for data

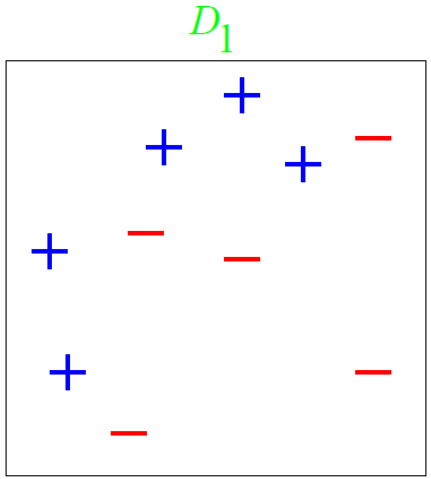
$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} e^{-\alpha_t y^i h_t(x^i)} = \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t} & if\ y^i = h_t(x^i) \\ e^{\alpha_t} & \text{otherwise} \end{cases}$$

where $Z_t = \text{normalizing constant}\ (Z_t = \sum_{i=1}^{m} D_t(i) e^{-\alpha_t y^i h_t(x^i)})$

- Final classifier $H(x) = \text{sign}(\sum_{t=1}^{T} \alpha_t h_t(x))$

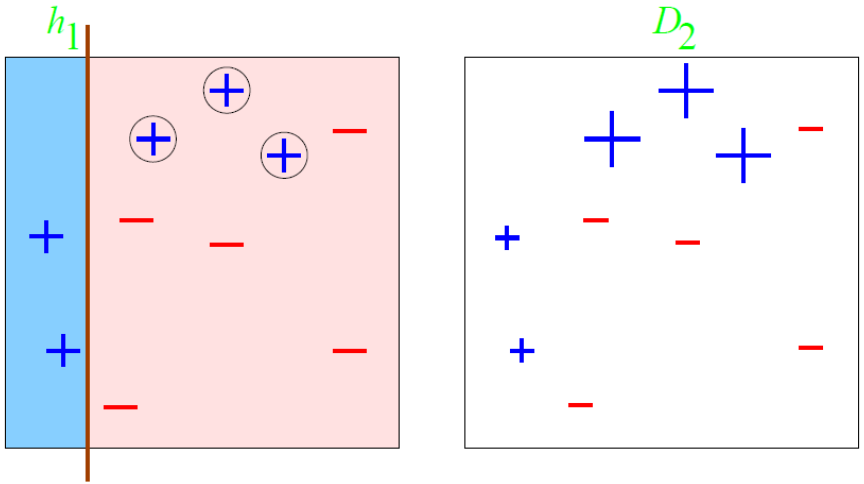
Toy Example

- Weak classifier (rule of thumb): vertical or horizontal half-planes (or decision stump)
- Let $y \in \{\pm 1\}$
- Uniform weights on all examples



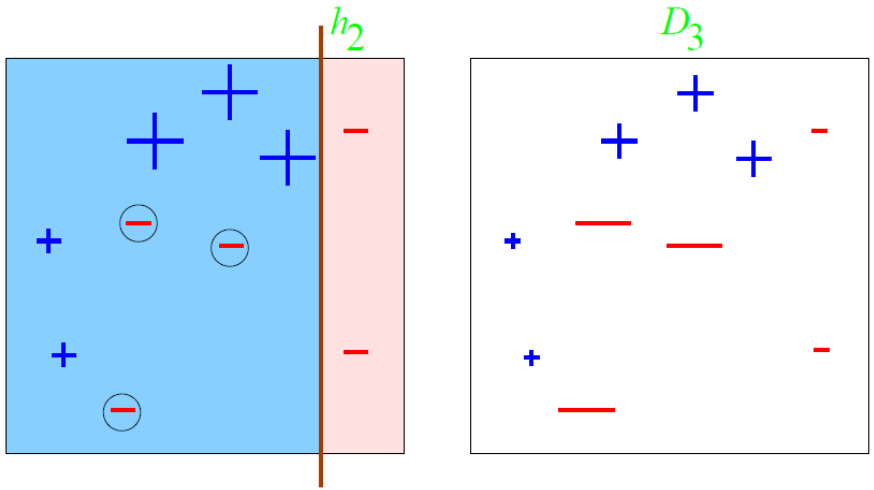
Boosting round 1

- Choose a decision stump (weak classifier)
- Some data points obtain higher weights because they are classified incorrectly



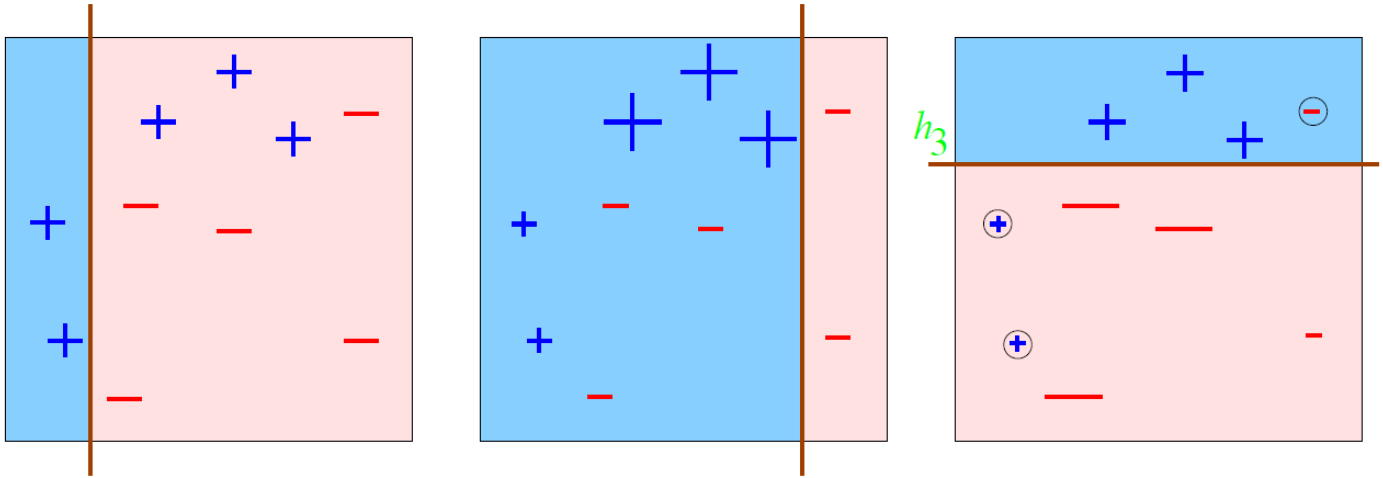
Boosting round 2

- Choose a new decision stump
- Reweight: For incorrectly classified examples, weight increased



Boosting round 3

- Repeat the same process
- Now we have 3 classifiers

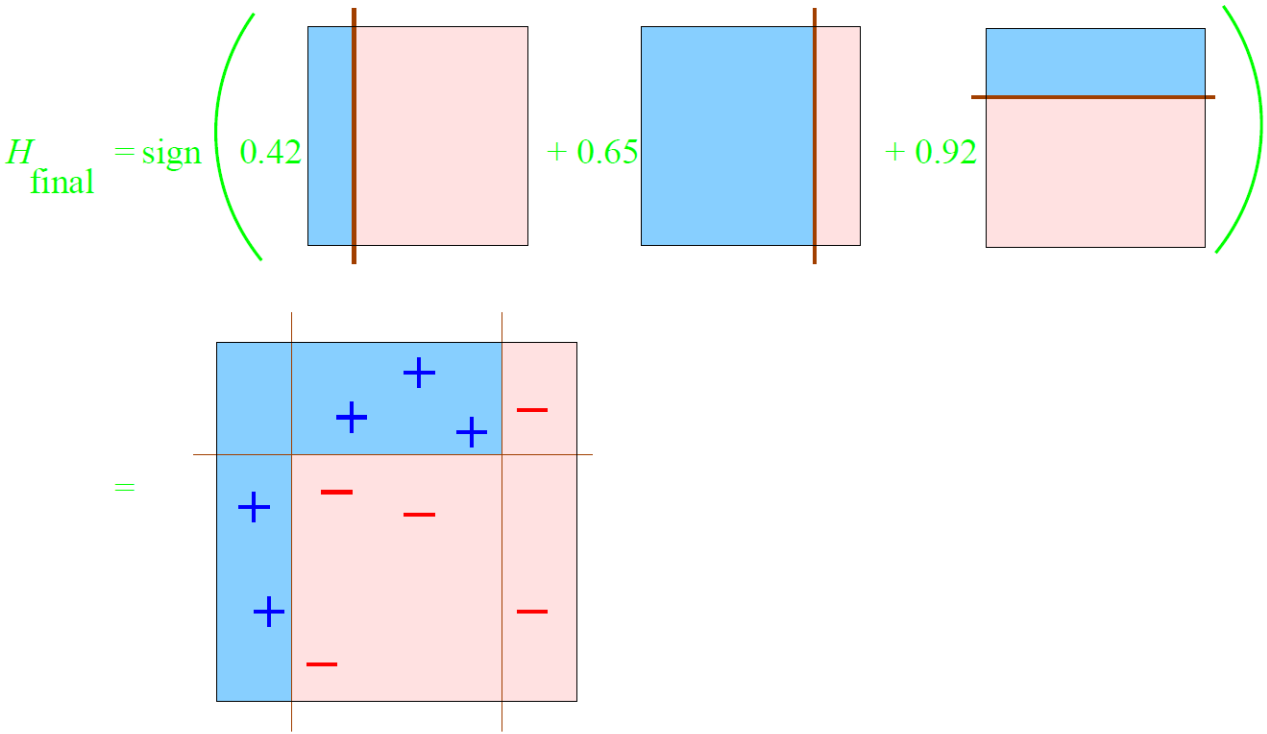


$\varepsilon_3 = 0.14$

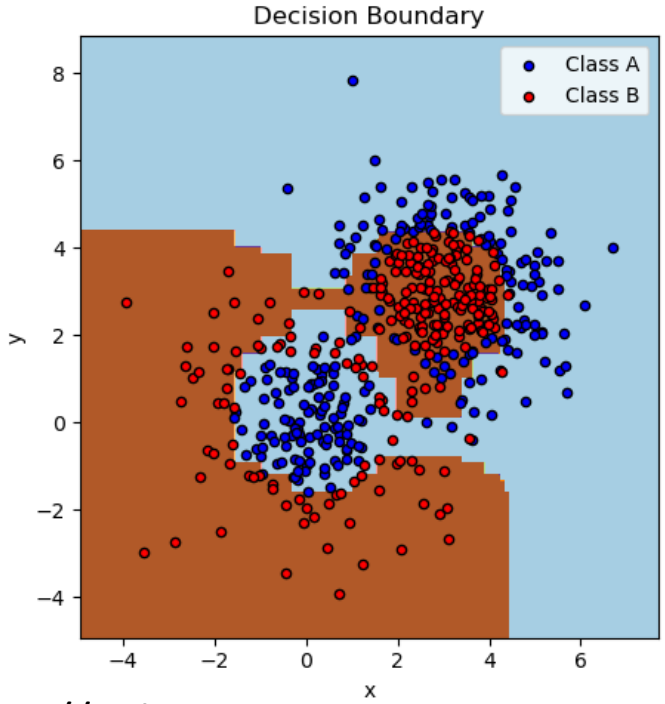
$\alpha_3 = 0.92$

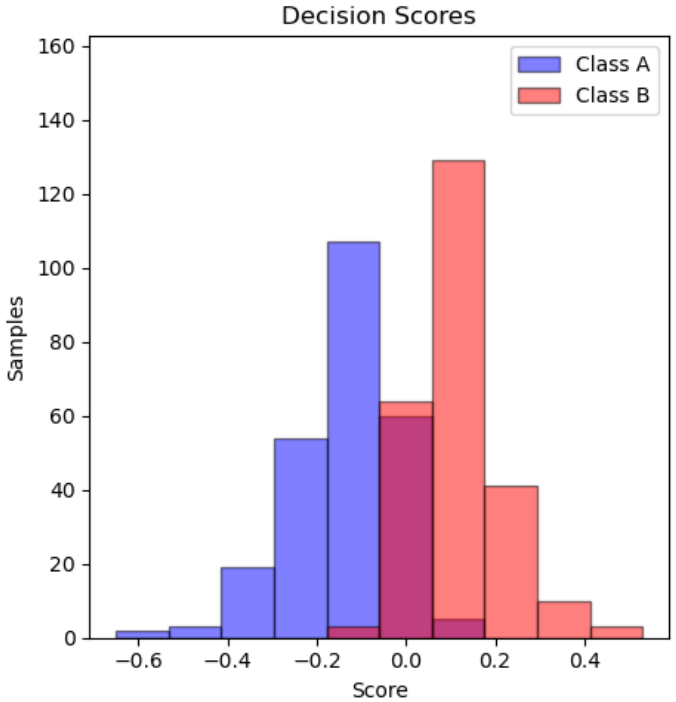
Boosting aggregate classifier

- Final classifier is weighted combination of weak classifiers



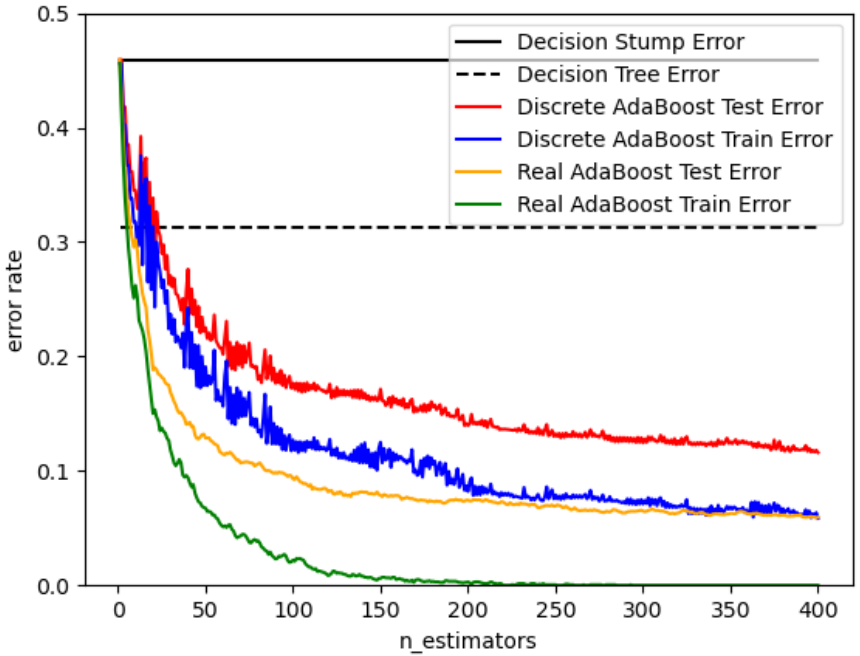
Demo: Two-class AdaBoost





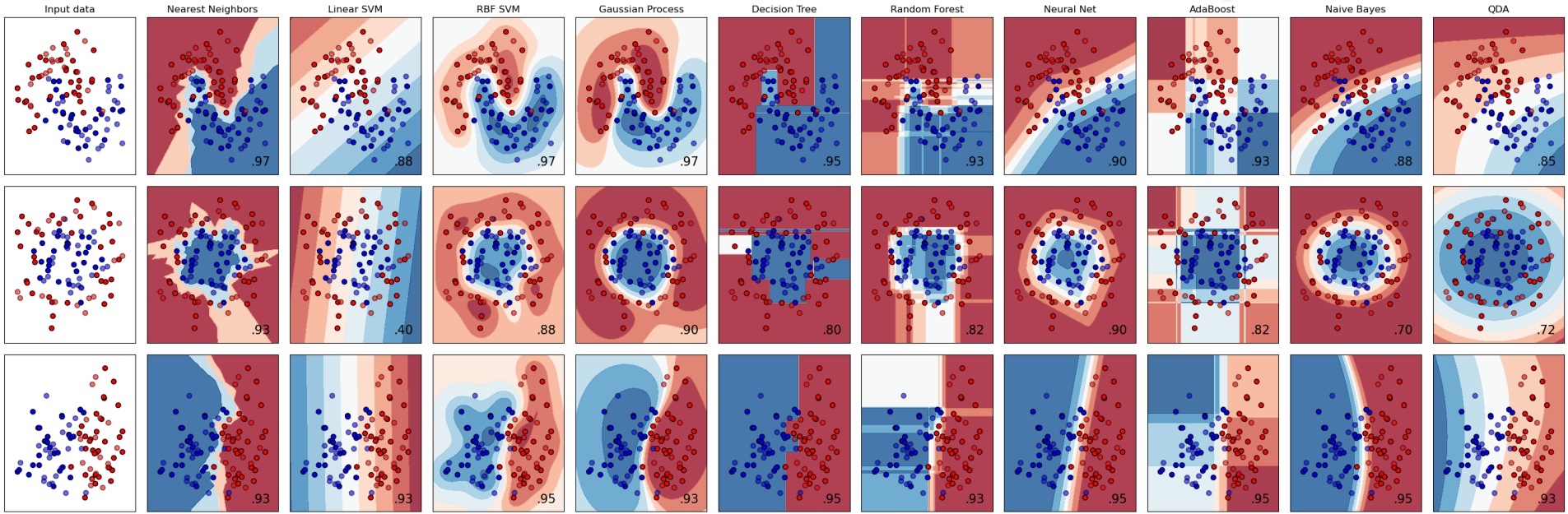
https://scikit-learn.org/stable/auto_examples/ensemble/plot_adaboost_twoclass.html#sphx-glr-download-auto-examples-ensemble-plot-adaboost-twoclass-py

Demo: error rate version number of iterations



https://scikit-learn.org/stable/auto_examples/ensemble/plot_adaboost_hastie_10_2.html#sphx-glr-auto-examples-ensemble-plot-adaboost-hastie-10-2-py

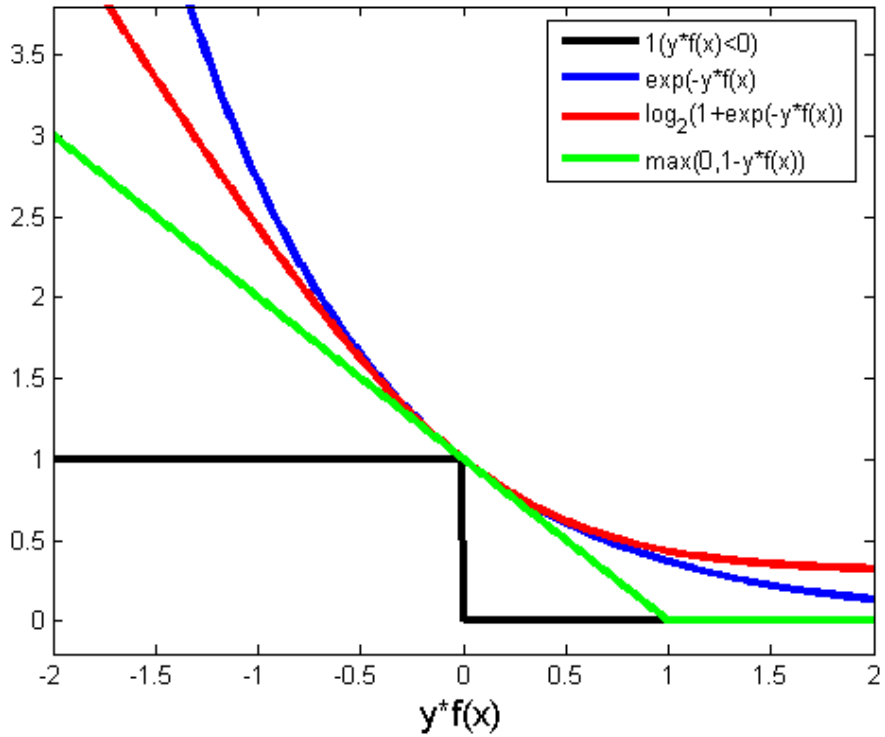
Example: Comparing decision boundaries



https://scikit-learn.org/stable/auto_examples/classification/plot_classifier_comparison.html#sphx-glr-auto-examples-classification-plot-classifier-comparison-py

Loss functions

- Loss function: $l\big(y, f(x)\big)$
- If use classification error
$$\mathbb{I}\{y * f(x) < 0\}$$
- Nonconvex, replacing it using convex upper bound, e.g., exponential loss
$$l\big(y, f(x)\big) = e^{-yf(x)}$$



Understanding boosting: exponential loss

- Combined classifier

$$f_t(x) = \alpha_1 h_1(x;\theta_1) + \cdots + \alpha_t h_t(x;\theta_t)$$

- Exponential loss

$$\hat{L}(f) = \frac{1}{m}\sum_{i=1}^{m} \exp\left(-y^i f_t(x^i)\right)$$

$$= \frac{1}{m}\sum_{i=1}^{m} \exp\left(-y^i\left(f_{t-1}(x^i) + \alpha_t h_t(x^i;\theta_t)\right)\right)$$

$$= \sum_{i=1}^{m} \underbrace{\frac{1}{m}\exp\left(-y^i f_{t-1}(x^i)\right)}_{= D_t(i)\prod_{s=1}^{t} Z_s} \exp\left(-\alpha_t y^i h_t(x^i;\theta_t)\right)$$

Property for the $D_t(i)$

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} e^{-\alpha_t y^i h_t(x^i)}$$

$$= \frac{D_{t-1}(i)}{Z_t Z_{t-1}} e^{-\alpha_{t-1} y^i h_{t-1}(x^i)} e^{-\alpha_t y^i h_t(x^i)}$$

…(Keep expanding)

$$= \frac{e^{-y^i \sum_{s=1}^{t} \alpha_s h_s(x^i)}}{m \prod_{s=1}^{t} Z_s}$$

$$= \frac{e^{-y^i f_t(x^i)}}{m \prod_{s=1}^{t} Z_s}$$

Find h_t: Linearization of loss function

- Assuming α_t is small, the loss criterion can be approximated by Taylor expansion

$$\exp\left(-y^i \alpha_t h_t(x^i;\theta_t)\right) \approx 1 - y^i \alpha_t h_t\left(x^i;\theta_t\right)$$

- We choose a new classifier to minimize the loss function

$$\hat{L}(f) = \left(\prod_{s=1}^{t} Z_s\right) \sum_{i=1}^{m} D_t(i) \exp\left(-\alpha_t y^i h_t(x^i;\theta_t)\right)$$

$$\propto \sum_{i=1}^{m} D_t(i) \left(1 - \alpha_t y^i h_t\left(x^i;\theta_t\right)\right)$$

$$= \sum_{i=1}^{m} D_t(i) - \alpha_t \sum_{i=1}^{m} D_t(i)\, y^i h_t\left(x^i;\theta_t\right)$$

Find h_t : a possible algorithm

- At stage t we find θ_t that maximize "weighted agreement":

$$\theta_t \leftarrow \text{argmax}_\theta \sum_{i=1}^{m} D_t(i)\, y^i h_t\left(x^i;\theta\right)$$

- Note that $y \in \{\pm 1\}$, and $h_t(x) \in \{\pm 1\}$

$$y^i h_t\left(x^i;\theta\right) \in \{\pm 1\}$$

- This approximately minimizes

$$\epsilon_t := \sum_{i=1}^{m} D_t(i)\, I\{y^i \neq h_t\left(x^i;\theta\right)\}$$

Finding α_t

- Then we go back and find the "weight" α_t associated with the new classifier by minimizing the **original** weighted (exponential) loss

$$L(\alpha_t) = \sum_{i=1}^{m} D_t(i) \exp\left(-y^i \alpha_t h_t(x^i; \theta_t)\right)$$

- Set derivative to 0 and solve for α_t

$$\frac{\partial L}{\partial \alpha_t} = -\sum_{i=1}^{m} D_t(i) \exp\left(-y^i \alpha_t h_t\left(x^i; \theta_t\right)\right) y^i h_t\left(x^i; \theta_t\right) = 0$$

Finding α_t

Note that $y^i \in \{\pm 1\}$, and $h_t(x^i) \in \{\pm 1\}$, $y^i h_t(x^i;\theta) \in \{\pm 1\}$

$$\frac{\partial L}{\partial \alpha_t} = -\sum_{i=1}^{m} D_t(i) \exp\left(-y^i \alpha_t h_t(x^i;\theta_t)\right) y^i h_t(x^i;\theta_t)$$

$$= -\sum_{i:h_t(x^i)=y^i} D_t(i) \exp(-\alpha_t) + \sum_{i:h_t(x^i)\neq y^i} D_t(i) \exp(\alpha_t)$$

$$= (\epsilon_t - 1)\exp(-\alpha_t) + \epsilon_t \exp(\alpha_t) = 0$$

$$\Rightarrow \frac{1-\epsilon_t}{\epsilon_t} = \exp(2\alpha_t) \Rightarrow \frac{1}{2}\ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right) = \alpha_t$$

$$\epsilon_t := \sum_{i=1}^{m} D_t(i)\, I\{y^i \neq h_t(x^i;\theta)\}$$

Extensions

- There are many others more recent algorithms such as:

LPBoost, TotalBoost, BrownBoost, xgboost, MadaBoost, LogitBoost.

- Generalization: **Gradient boosting** is a machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically **decision trees**; allowing optimization of an arbitrary differentiable loss function.


SUMMARY