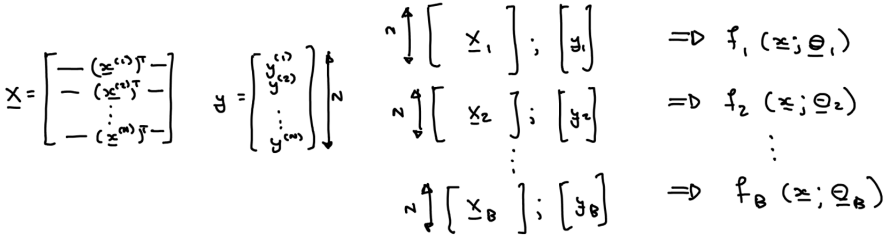
Ensemble methods

Herman Kamper

Bagging

We could overcome problems because of using a single dataset (e.g. overfitting) by collecting more datasets:



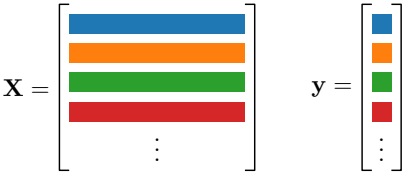
Aggregate model predictions:

- Regression:

$$f(\mathbf{x};\boldsymbol{\theta}) = \frac{1}{B}\sum_{b=1}^{B} f_b(\mathbf{x};\boldsymbol{\theta}_b)$$

- Classification: Majority or weighted voting

But in most cases we cannot just go out and collect more data. So we simulate this process using *bootstrap* samples:



Random forests

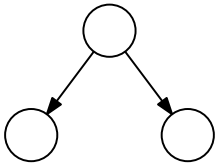
Specifically used with decision trees (regression and classification).

Algorithm:

- Bagging: Train each tree on a different bootstrap sample. But then also ...
- Every time you split, only consider $M < D$ random features.
- Common choice: $M = \sqrt{D}$

Input: $\mathbf{x} \in \mathbb{R}^D$

$x_1,\ x_2,\ x_3,\ \ldots,\ x_D$



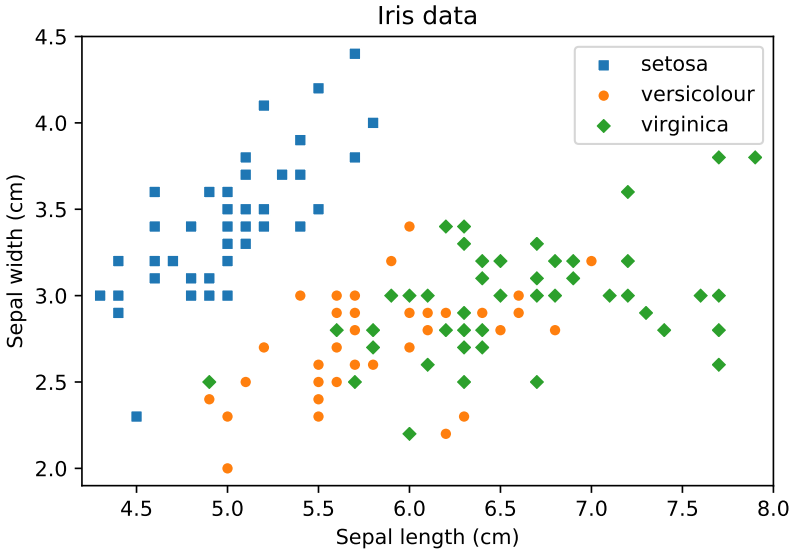
Aggregate model predictions:

- Regression:

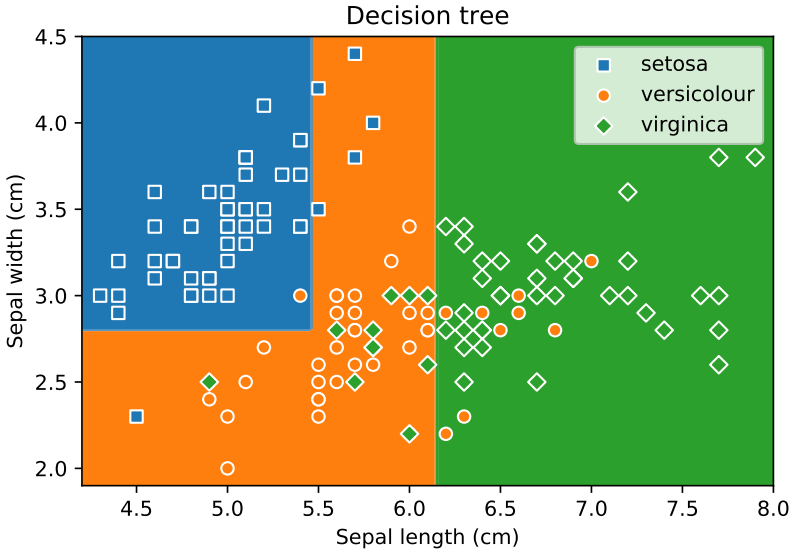
$$f(\mathbf{x}; \boldsymbol{\theta}) = \frac{1}{B} \sum_{b=1}^{B} f_b(\mathbf{x}; \boldsymbol{\theta}_b)$$

- Classification: Majority or weighted voting

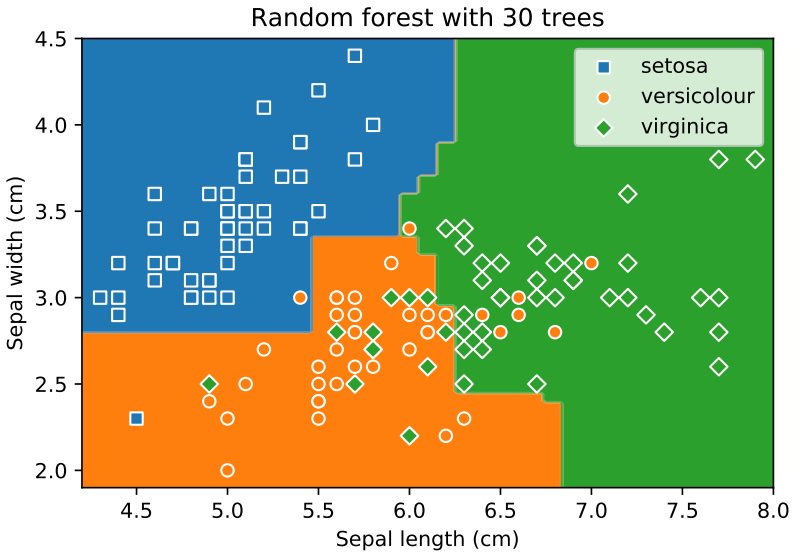
Random forest on iris dataset



Random forest on iris dataset



Random forest on iris dataset



Boosting for regression

Can we combine multiple weak models (just a little bit better than random) to get a "strong" resulting model?

Boosting is different from bagging in that we train a model, look at its mistakes, and then train the next model (instead of just training all the models in one go).

It is different from random forests since it can be used with any weak model (instead of working with decision trees specifically).

Boosting for regression algorithm

Algorithm

- Initialise $r^{(n)} = y^{(n)}$ for all N training items and set $f(\mathbf{x}; \boldsymbol{\theta}) \leftarrow 0$.
- for iteration $b = 1$ to B:
  (a) Fit model $f_b(\mathbf{x}; \boldsymbol{\theta}_b)$ to inputs $\mathbf{X}$, outputs $\mathbf{r}$.
  (b) Update model by adding shrunken version:
  $$f(\mathbf{x}; \boldsymbol{\theta}) \leftarrow f(\mathbf{x}; \boldsymbol{\theta}) + \lambda f_b(\mathbf{x}; \boldsymbol{\theta}_b)$$
  (c) Update the residuals:
  $$r^{(n)} \leftarrow r^{(n)} - \lambda f_b(\mathbf{x}^{(n)}; \boldsymbol{\theta}_b)$$
- Final model: $f(\mathbf{x}; \boldsymbol{\theta}) = \sum_{b=1}^{B} \lambda f_b(\mathbf{x}; \boldsymbol{\theta}_b)$

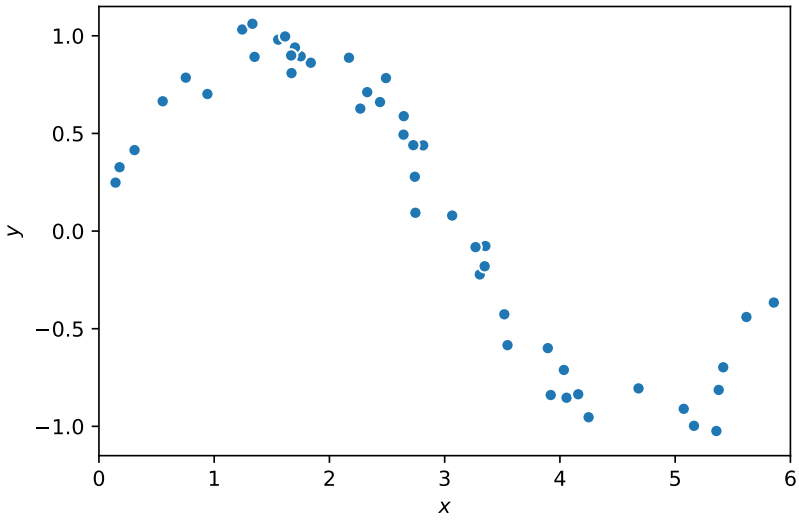
What are we doing?

- At $b = 1$:

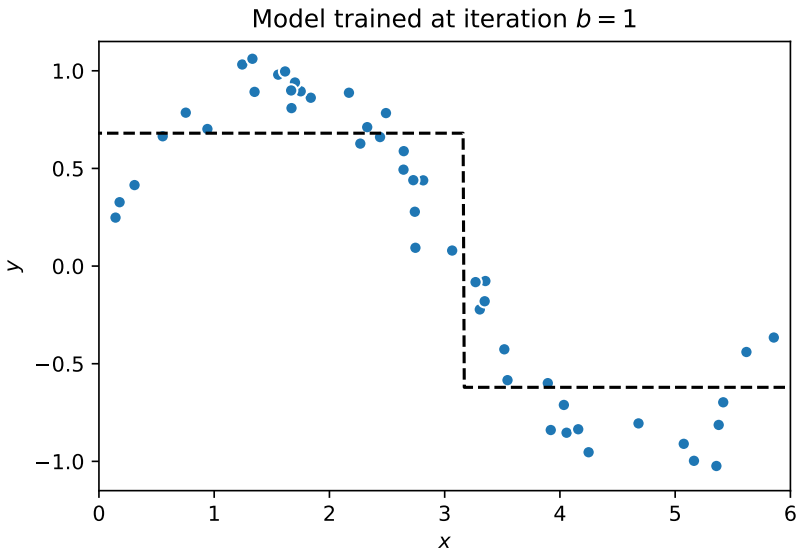
  $\mathbf{r} = \mathbf{y}$, so we are just fitting a model to inputs $\mathbf{X}$, outputs $\mathbf{y}$.
- At $b > 1$:

  Fitting a model to the residuals.

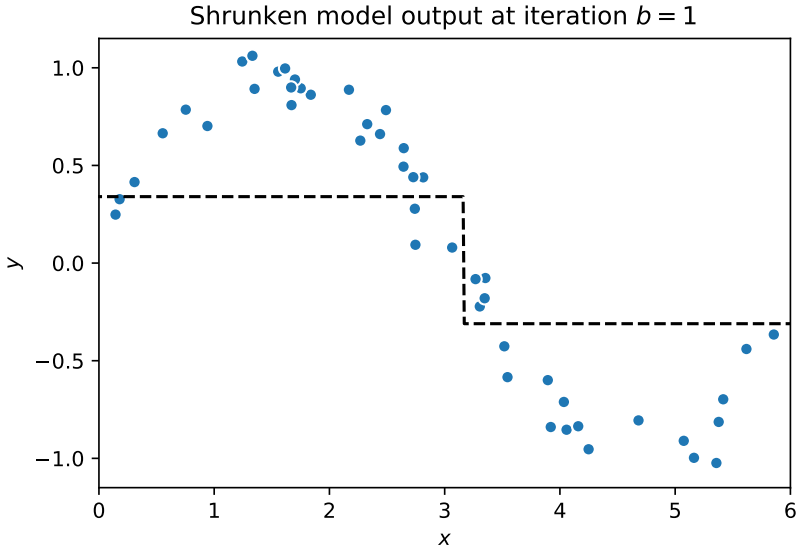
Boosting for regression example



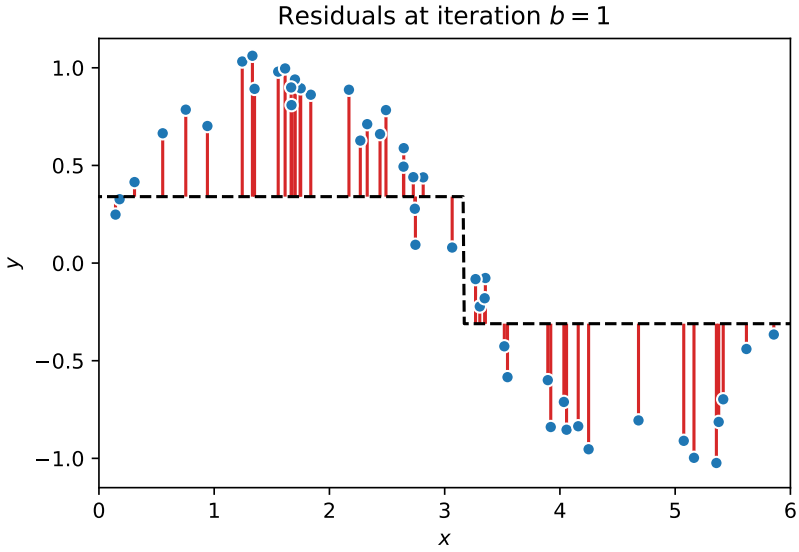
Boosting for regression example



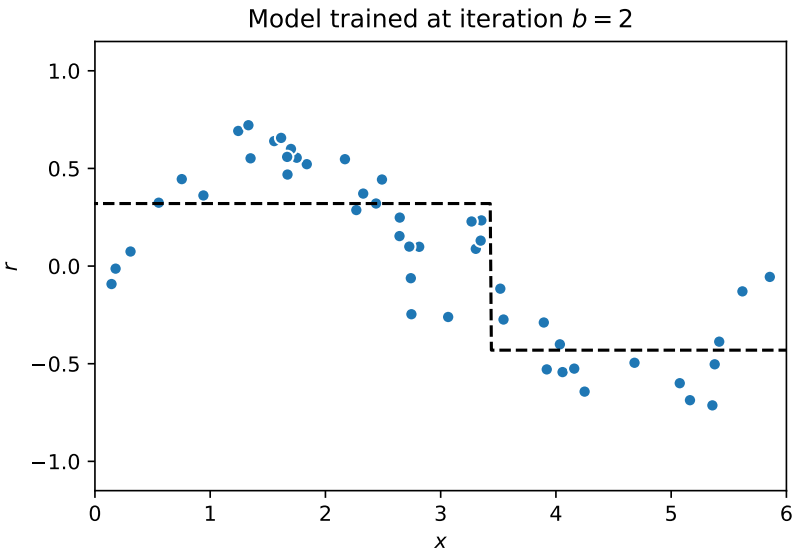
Boosting for regression example



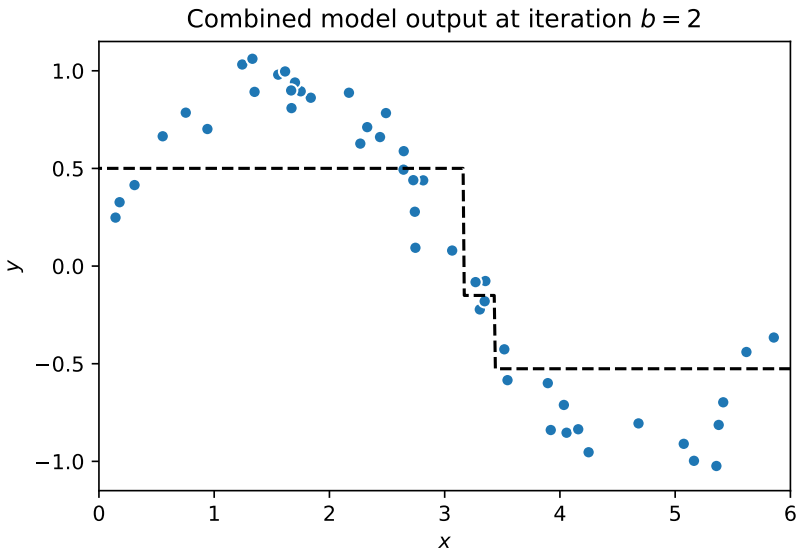
Boosting for regression example



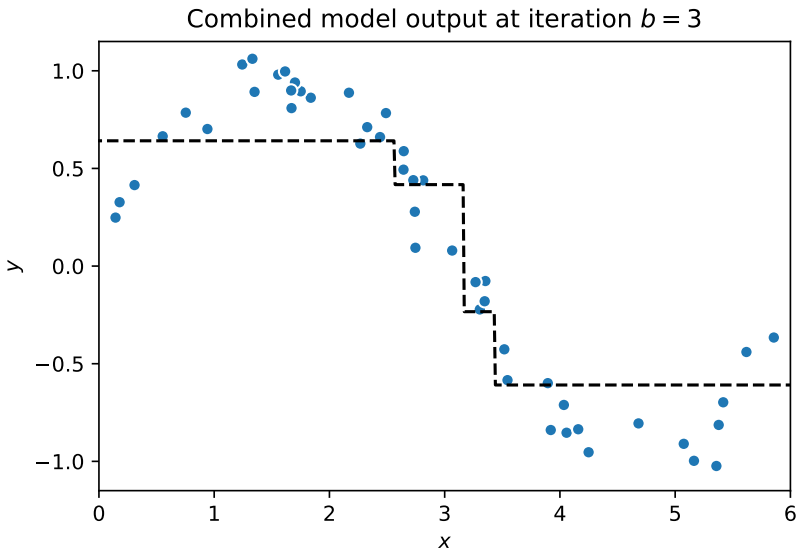
Boosting for regression example



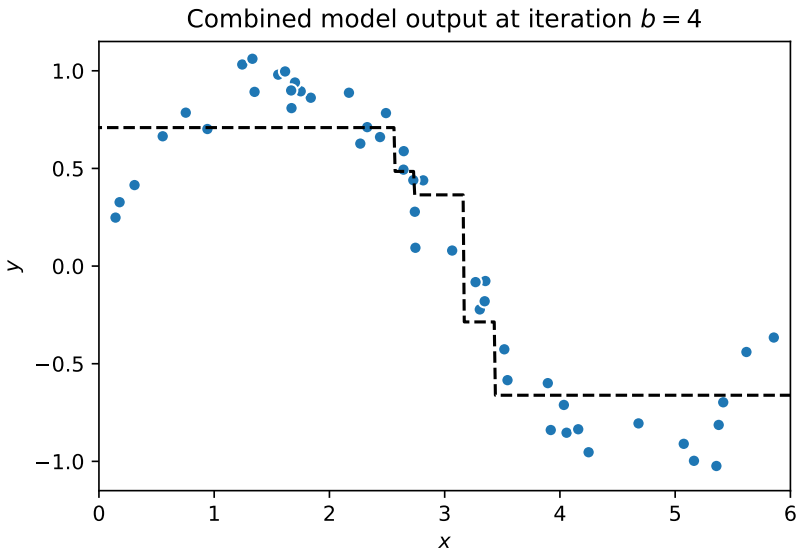
Boosting for regression example



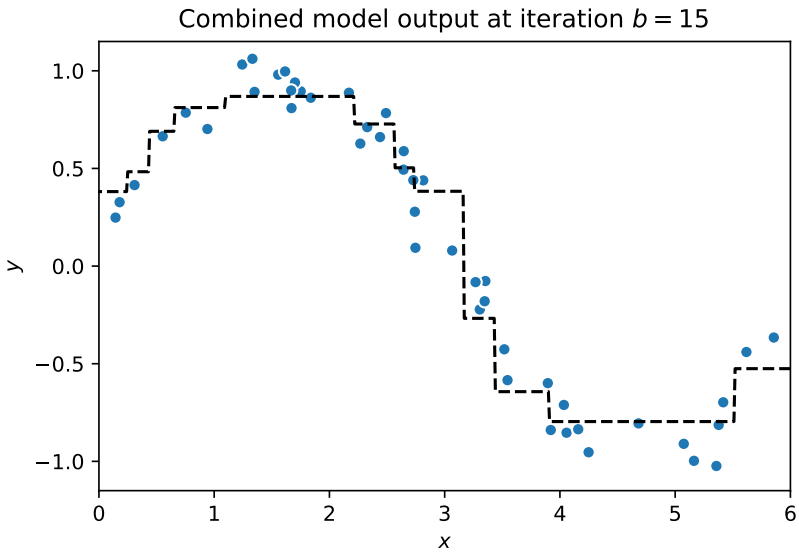
Boosting for regression example



Boosting for regression example



Boosting for regression example



AdaBoost: Boosting for classification

We look at binary classification: $y \in \{-1, 1\}$

Train B models, each $f_b(\mathbf{x}; \boldsymbol{\theta}_b) \in \{-1, 1\}$.

Combine weighted votes:

$$f(\mathbf{x}; \boldsymbol{\theta}) = \text{sign}\left[\sum_{b=1}^{B} \lambda_b f_b(\mathbf{x}; \boldsymbol{\theta}_b)\right]$$

AdaBoost: Boosting for classification

Algorithm

- Initialise training item weights $w^{(n)} = 1$ for all N training items.
- for iteration $b = 1$ to B:
  (a) Fit model $f_b(\mathbf{x}; \boldsymbol{\theta}_b)$ so that it minimises classification error weighted by $w^{(n)}$.
  (b) Set model weight using error ϵ:
  $$\lambda_b = \frac{1}{2} \log \left(\frac{1 - \epsilon}{\epsilon} \right)$$
  where ϵ is the normalised error term.
  (c) Update training item weights:
  $$w^{(n)} \leftarrow w^{(n)} e^{-\lambda_b} \text{ if } f_b(\mathbf{x}^{(n)}; \boldsymbol{\theta}_b) \text{ correct}$$
  $$w^{(n)} \leftarrow w^{(n)} e^{\lambda_b} \text{ if } f_b(\mathbf{x}^{(n)}; \boldsymbol{\theta}_b) \text{ incorrect}$$
- Final model: $f(\mathbf{x}; \boldsymbol{\theta}) = \text{sign} \left[\sum_{b=1}^{B} \lambda_b f_b(\mathbf{x}; \boldsymbol{\theta}_b) \right]$

Details

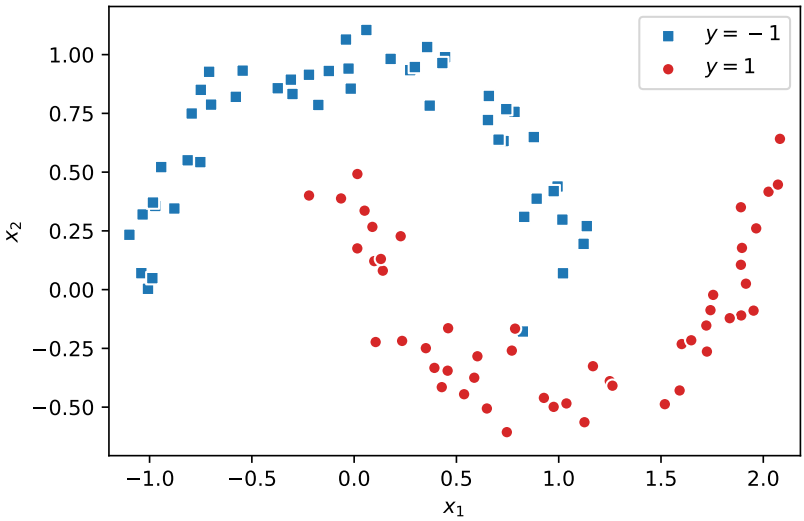
(a) Goal: Minimise $\sum_{n=1}^{N} w^{(n)} \mathbb{I} \left\{ y^{(n)} \neq f_b(\mathbf{x}; \boldsymbol{\theta}_b) \right\}$

(b) Error term: $\epsilon = \dfrac{\sum_{n=1}^{N} w^{(n)} \mathbb{I} \left\{ y^{(n)} \neq f_b(\mathbf{x}; \boldsymbol{\theta}_b) \right\}}{\sum_{n=1}^{N} w^{(n)}}$

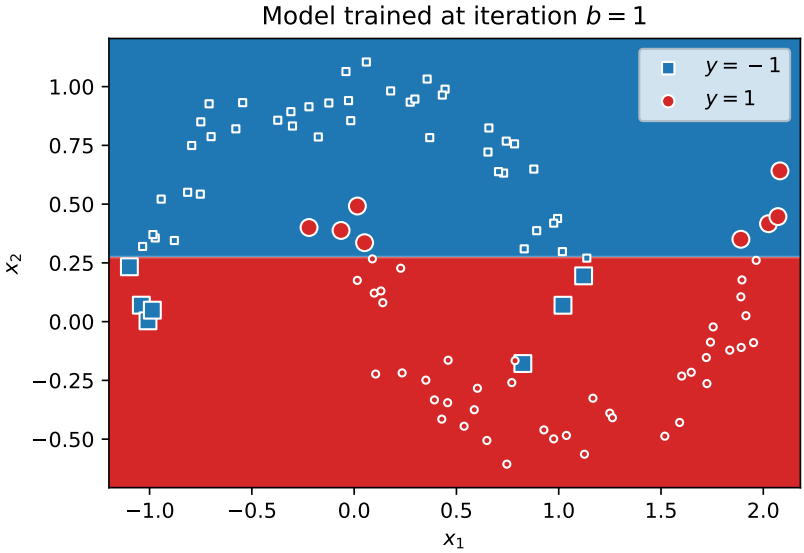
(c) Correct: $w^{(n)} \leftarrow w^{(n)} \sqrt{\dfrac{\epsilon}{1 - \epsilon}}$

Incorrect: $w^{(n)} \leftarrow w^{(n)} \sqrt{\dfrac{1 - \epsilon}{\epsilon}}$

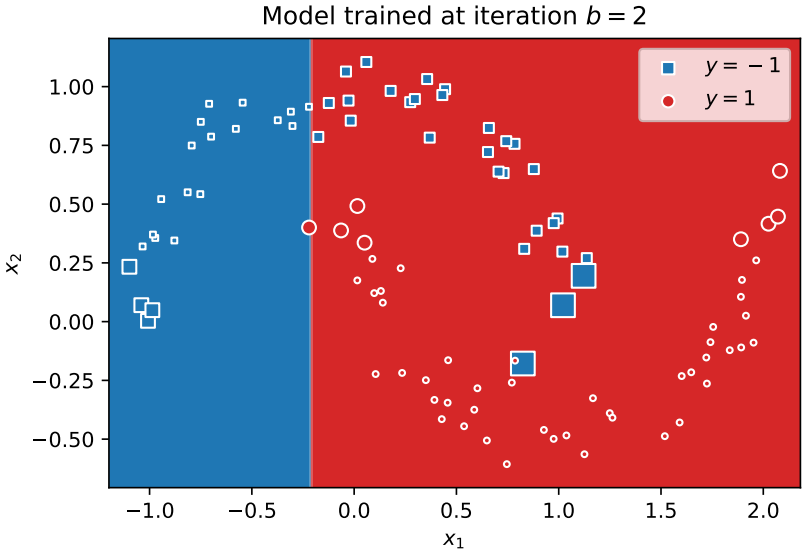
AdaBoost example



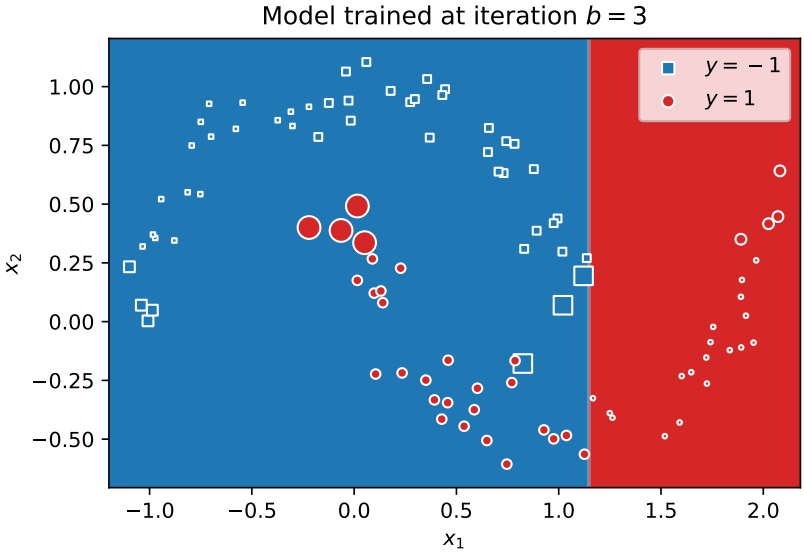
AdaBoost example



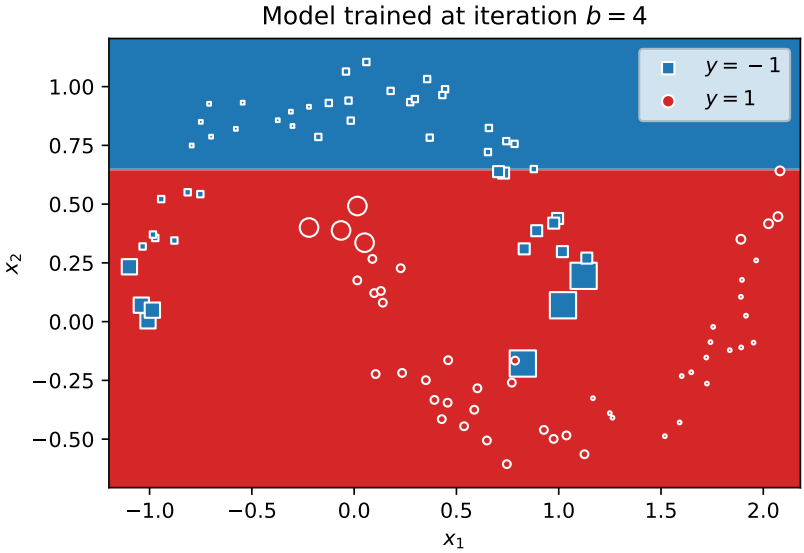
AdaBoost example



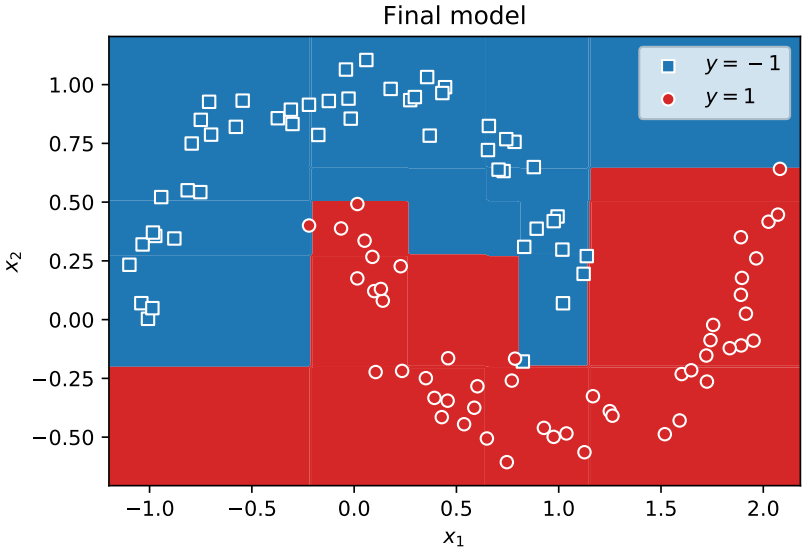
AdaBoost example



AdaBoost example



AdaBoost example



In practice

Many of the machine learning competitions on Kaggle are won with boosting-based methods, especially on problems with tabular data.

Videos covered in this note

- Ensemble methods 1: Bagging (13 min)
- Ensemble methods 2: Random forests (7 min)
- Ensemble methods 3: Boosting for regression (21 min)
- Ensemble methods 4.1: AdaBoost for classification - Setup (10 min)
- Ensemble methods 4.2: AdaBoost for classification - Step-by-step (15 min)
- Ensemble methods 4.3: AdaBoost for classification - Details (11 min)

Reading

- ISLR 8.2.1
- ISLR 8.2.2
- ISLR 8.2.3

Further reading

R. Rojas, "AdaBoost and the super bowl of classifiers: A tutorial introduction to adaptive boosting," *Freie Universität Berlin*, 2009.

T. Hastie, R. Tibshirani, and J. H. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, 2nd ed., 2009, Chapter 10.