Ensemble methods

Bagging

Herman Kamper

http://www.kamperh.com/

Bagging:

$$\underline{X} = \begin{bmatrix} — (\underline{x}^{(1)})^T — \\ — (\underline{x}^{(2)})^T — \\ \vdots \\ — (\underline{x}^{(N)})^T — \end{bmatrix} \qquad \underline{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(N)} \end{bmatrix} \Big\updownarrow N$$

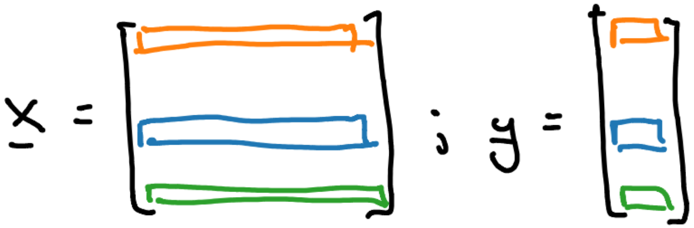
$$N \Big\updownarrow \begin{bmatrix} \underline{X}_1 \end{bmatrix} ; \begin{bmatrix} \underline{y}_1 \end{bmatrix} \Rightarrow f_1(\underline{x}; \underline{\theta}_1)$$

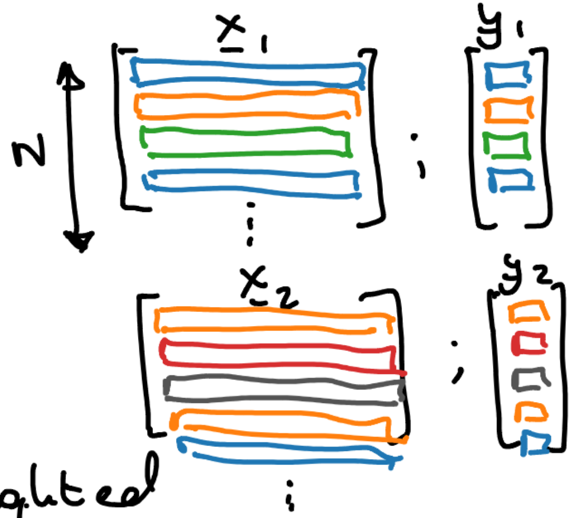
$$N \Big\updownarrow \begin{bmatrix} \underline{X}_2 \end{bmatrix} ; \begin{bmatrix} \underline{y}_2 \end{bmatrix} \Rightarrow f_2(\underline{x}; \underline{\theta}_2)$$

$$\vdots$$

$$N \Big\updownarrow \begin{bmatrix} \underline{X}_B \end{bmatrix} ; \begin{bmatrix} \underline{y}_B \end{bmatrix} \Rightarrow f_B(\underline{x}; \underline{\theta}_B)$$

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





$$\Rightarrow f_1(\underline{x}; \underline{\theta}_1)$$

$$\Rightarrow f_2(\underline{x}; \underline{\theta}_2)$$

$$\vdots$$

$$f_B(\underline{x}; \theta_B)$$

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

Regression

Classification: Majority or weighted voting

Ensemble methods

Random forests

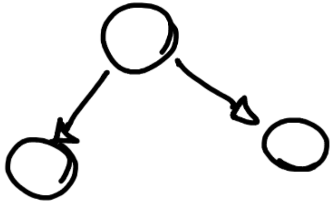
Herman Kamper

http://www.kamperh.com/

Random forests:

- Specifically used with decision and regression trees.
- Use bagging: Train each tree on different bootstrap sample. But then also...
- Every time we split, only consider $M < D$ random features
- $M = \sqrt{D}$ is often used

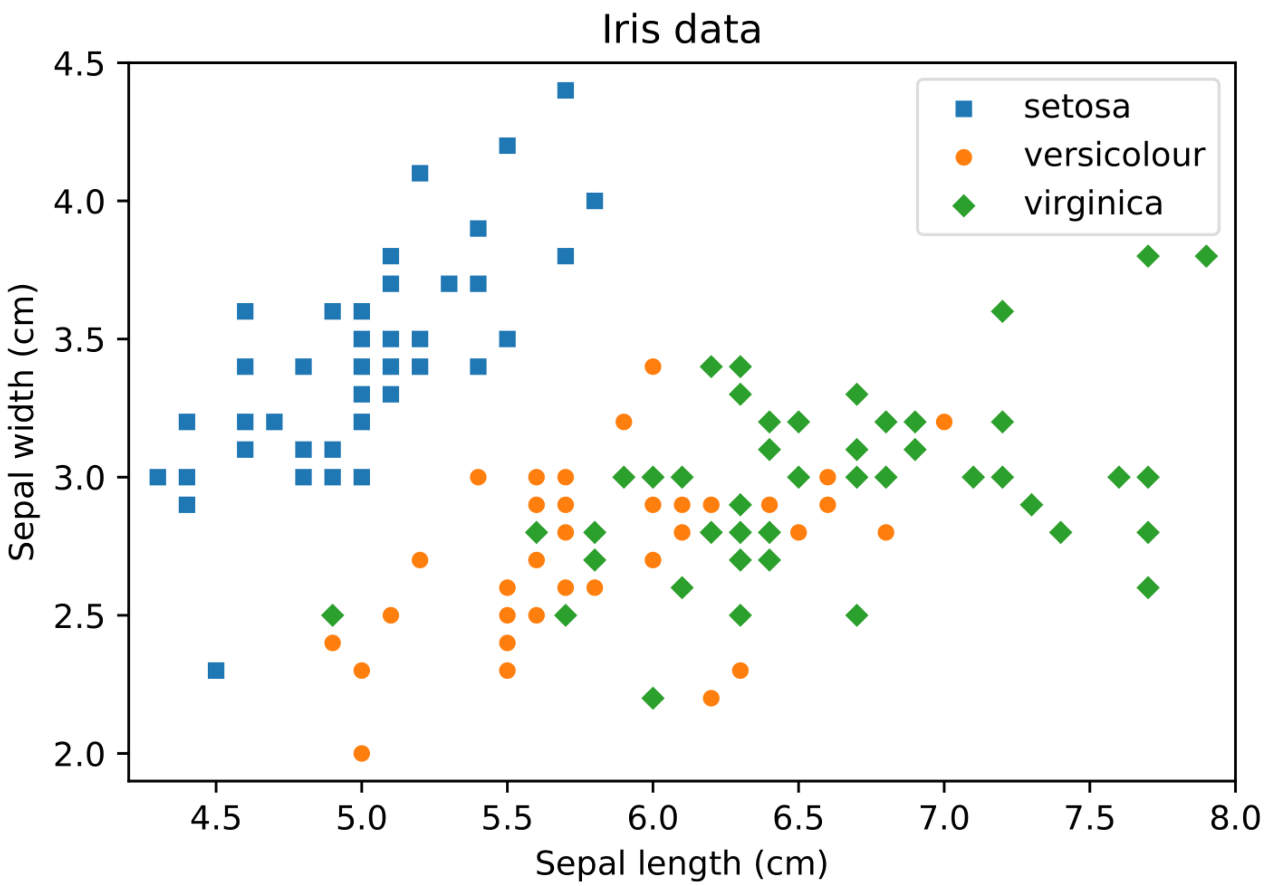
$$f(\underline{x}; \underline{\theta}) = \frac{1}{B} \sum_{b=1}^{B} f_b(\underline{x}; \underline{\theta}_b)$$



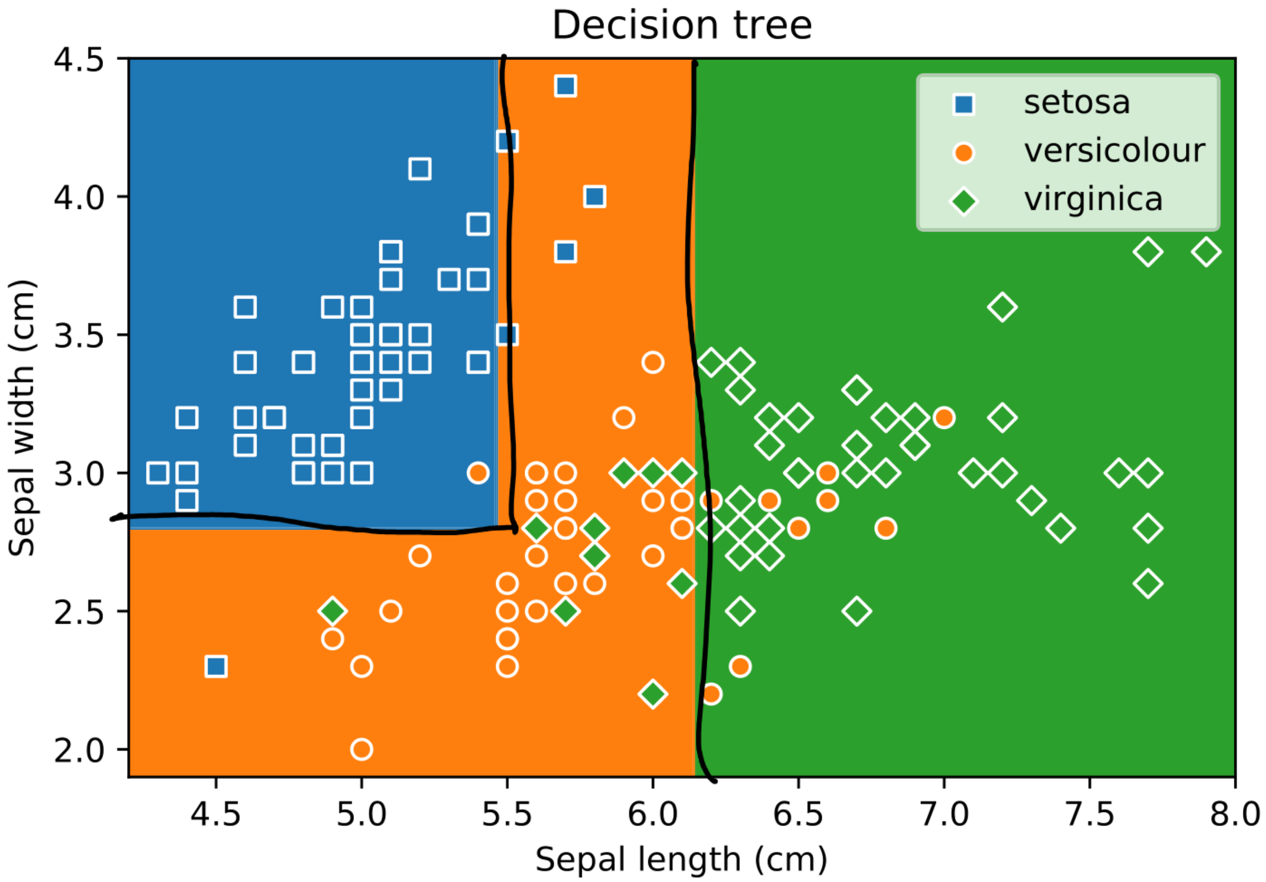
Input: $\underline{x} \in \mathbb{R}^D$

$x_1, \cancel{x_2}, x_3, \ldots \cancel{x_D}$

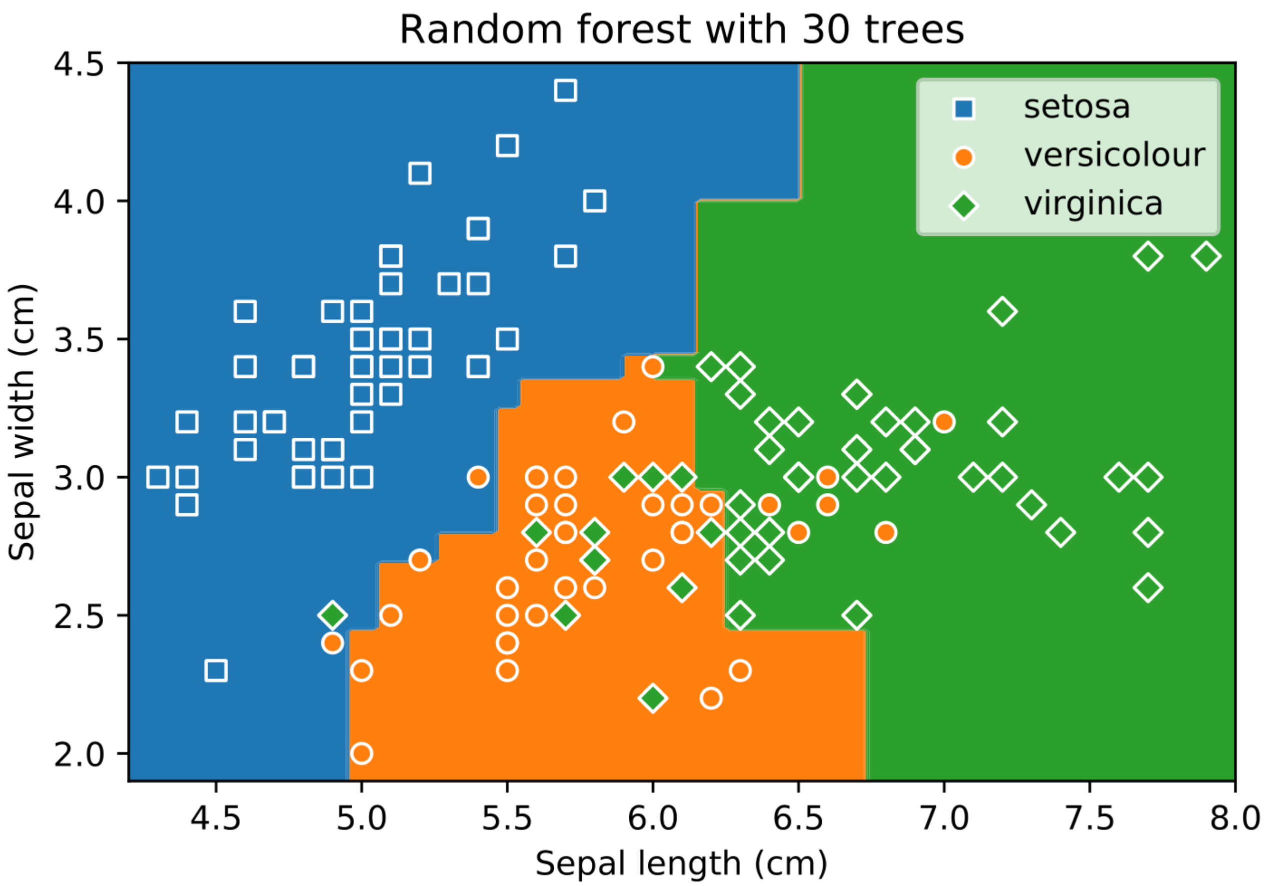
Random forest on Iris data



Random forest on Iris data



Random forest on Iris data



Ensemble methods

Boosting for regression

Can we combine multiple weak models (just a little bit better than random) into a "strong" model.

- Different from bagging in that we train one model, look at mistakes, only then train next
- Can be used with any weak models.

Herman Kamper

http://www.kamperh.com/

Boosting for regression

1. Initialise $r^{(n)} = y^{(n)}$ for all N training items and set $f(\mathbf{x};\boldsymbol{\theta}) = 0$.

2. 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:
   $r^{(n)} \leftarrow r^{(n)} - \lambda f_b(\mathbf{x};\boldsymbol{\theta}_b)$

3. Final model: $f(\mathbf{x};\boldsymbol{\theta}) = \sum_{b=1}^{B} \lambda f_b(\mathbf{x};\boldsymbol{\theta}_b)$

$$\underline{X} = \begin{bmatrix} — (\underline{x}^{(1)})^T — \\ — (\underline{x}^{(2)})^T — \\ \vdots \\ — (\underline{x}^{(N)})^T — \end{bmatrix} ; \quad \underline{r} = \begin{bmatrix} r^{(1)} \\ r^{(2)} \\ \vdots \\ r^{(N)} \end{bmatrix}$$

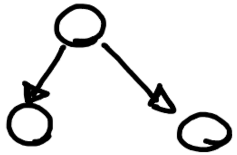
At $b = 1$:

$\underline{r} = \underline{y}$, so we are just fitting a model to inputs $\underline{X}$, outputs $\underline{y}$

At $b > 1$:

Fitting a model to the residuals.

Boosting for regression

Weak learner: 

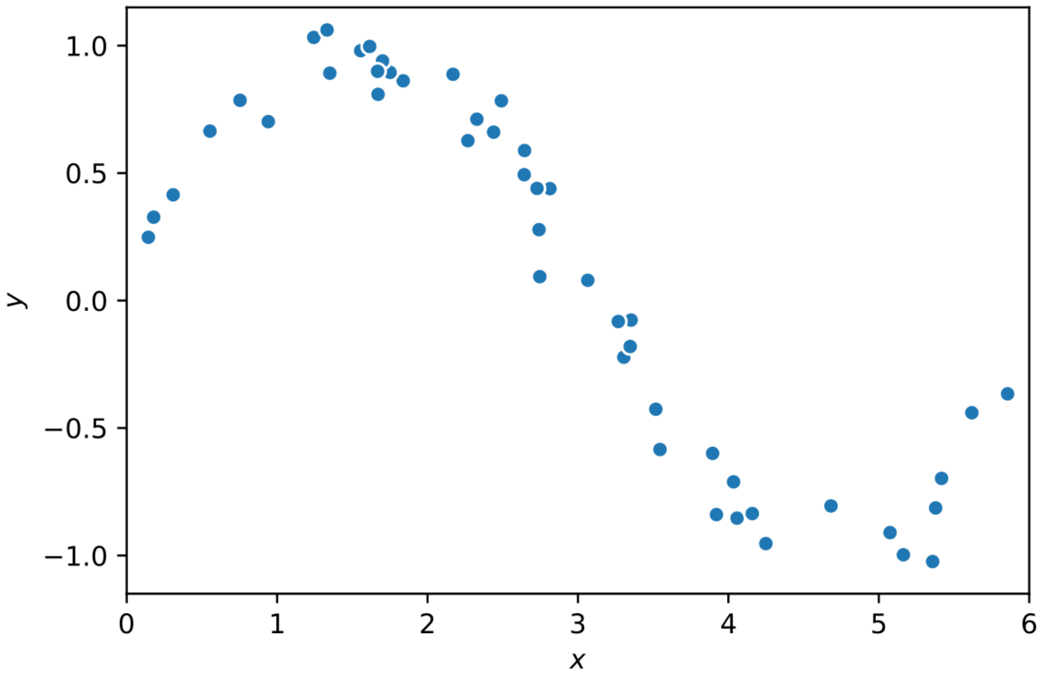
Decision tree stub (only one split)

1. Initialise $r^{(n)} = y^{(n)}$ for all N training items and set $f(\mathbf{x}; \boldsymbol{\theta}) = 0$.

   At $b=1$: $r = y$

2. 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: $r^{(n)} \leftarrow r^{(n)} - \lambda f_b(\mathbf{x}; \boldsymbol{\theta}_b)$

3. Final model: $f(\mathbf{x}; \boldsymbol{\theta}) = \sum_{b=1}^{B} \lambda f_b(\mathbf{x}; \boldsymbol{\theta}_b)$



Boosting for regression

$f_1(\underline{x}; \underline{\theta}_1)$

1. Initialise $r^{(n)} = y^{(n)}$ for all N training items and set $f(\mathbf{x}; \boldsymbol{\theta}) = 0$.

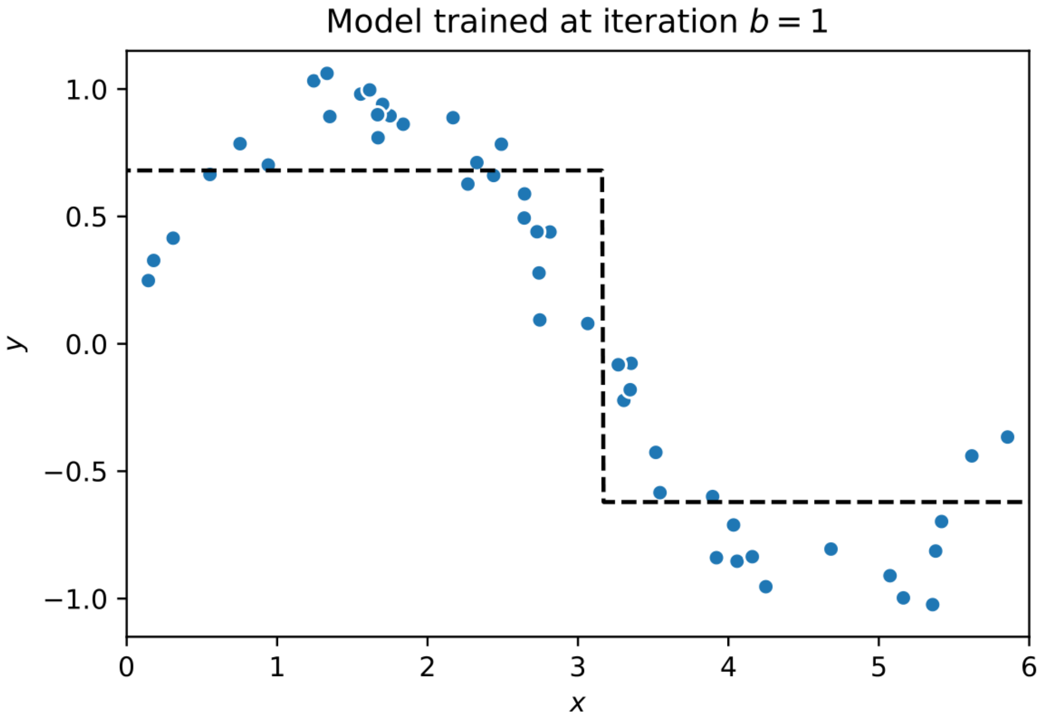
2. for iteration $b = 1$ to B:

   (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:
   $r^{(n)} \leftarrow r^{(n)} - \lambda f_b(\mathbf{x}; \boldsymbol{\theta}_b)$

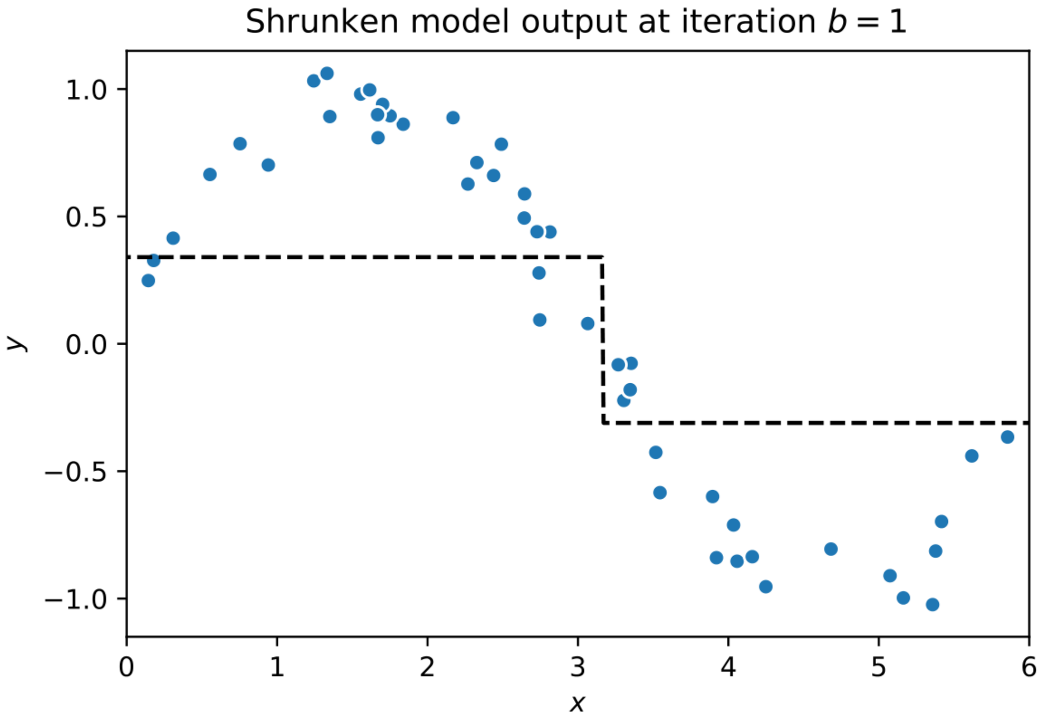
3. Final model: $f(\mathbf{x}; \boldsymbol{\theta}) = \sum_{b=1}^{B} \lambda f_b(\mathbf{x}; \boldsymbol{\theta}_b)$



Boosting for regression

1. Initialise $r^{(n)} = y^{(n)}$ for all N training items and set $f(\mathbf{x}; \boldsymbol{\theta}) = 0$.
2. 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: $r^{(n)} \leftarrow r^{(n)} - \lambda f_b(\mathbf{x}; \boldsymbol{\theta}_b)$
3. Final model: $f(\mathbf{x}; \boldsymbol{\theta}) = \sum_{b=1}^{B} \lambda f_b(\mathbf{x}; \boldsymbol{\theta}_b)$

$\lambda f_1(x; \theta_1)$



Boosting for regression

1. Initialise $r^{(n)} = y^{(n)}$ for all N training items and set $f(\mathbf{x};\boldsymbol{\theta}) = 0$.

2. for iteration $b = 1$ to B:

   (a) Fit model $f_b(\mathbf{x};\boldsymbol{\theta}_b)$ to inputs $\mathbf{X}$, outputs $\mathbf{r}$.

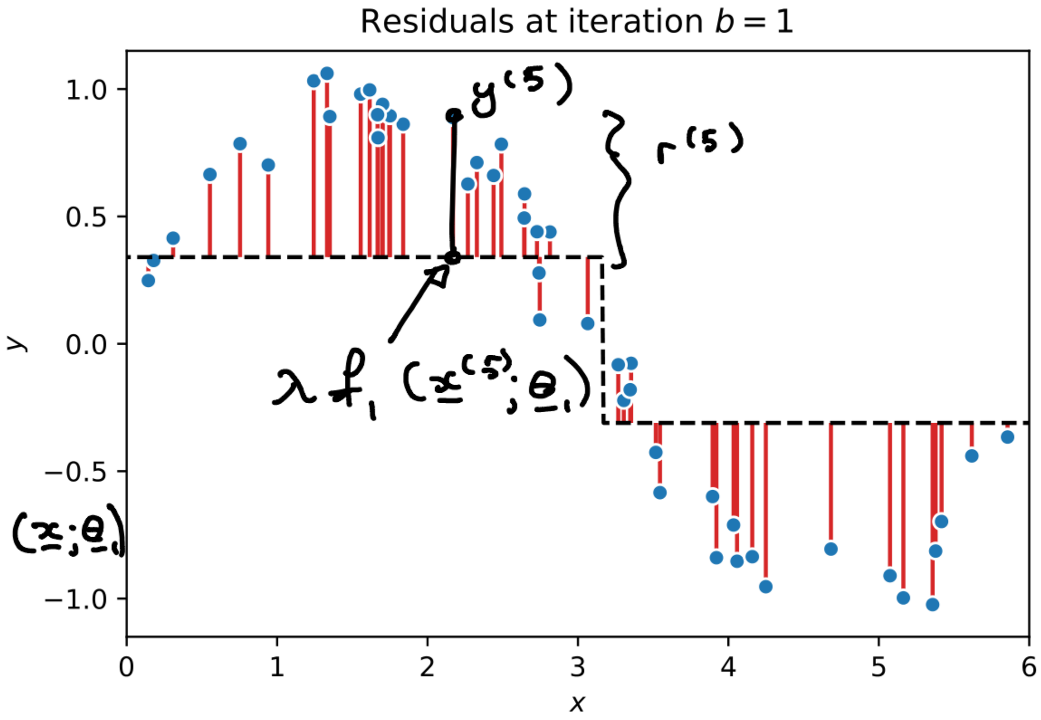
   (b) Update model by adding shrunken version:
   $$f(\mathbf{x};\boldsymbol{\theta}) \leftarrow f(\mathbf{x};\boldsymbol{\theta}) + \lambda f_b(\mathbf{x};\boldsymbol{\theta}_b)$$

   (c) Update the residuals:
   $$r^{(n)} \leftarrow r^{(n)} - \lambda f_b(\mathbf{x};\boldsymbol{\theta}_b)$$

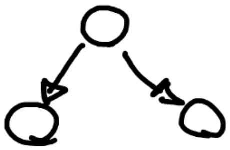
At $b=1$:

$r^{(n)} \leftarrow y^{(n)} - \lambda f_1(\underline{x};\underline{\theta})$

3. Final model: $f(\mathbf{x};\boldsymbol{\theta}) = \sum_{b=1}^{B} \lambda f_b(\mathbf{x};\boldsymbol{\theta}_b)$

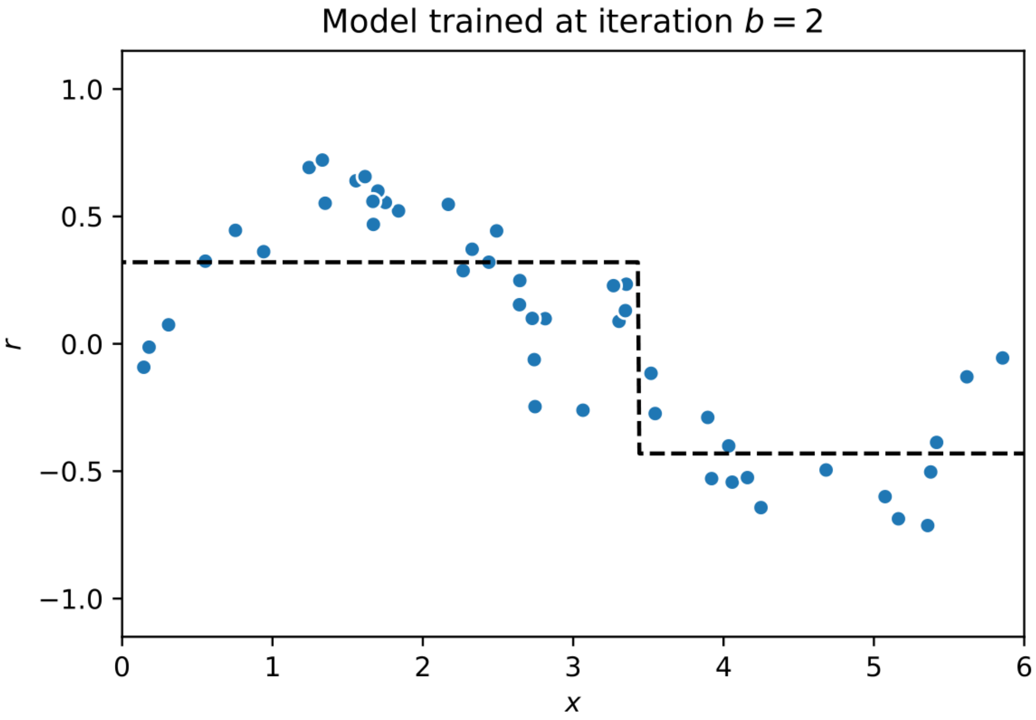


Boosting for regression



$f_2(x; \theta_2)$

1. Initialise $r^{(n)} = y^{(n)}$ for all N training items and set $f(\mathbf{x};\boldsymbol{\theta}) = 0$.
2. for iteration $b = 1$ to B: $\boxed{b=2}$
   (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:
   $r^{(n)} \leftarrow r^{(n)} - \lambda f_b(\mathbf{x};\boldsymbol{\theta}_b)$
3. Final model: $f(\mathbf{x};\boldsymbol{\theta}) = \sum_{b=1}^{B} \lambda f_b(\mathbf{x};\boldsymbol{\theta}_b)$

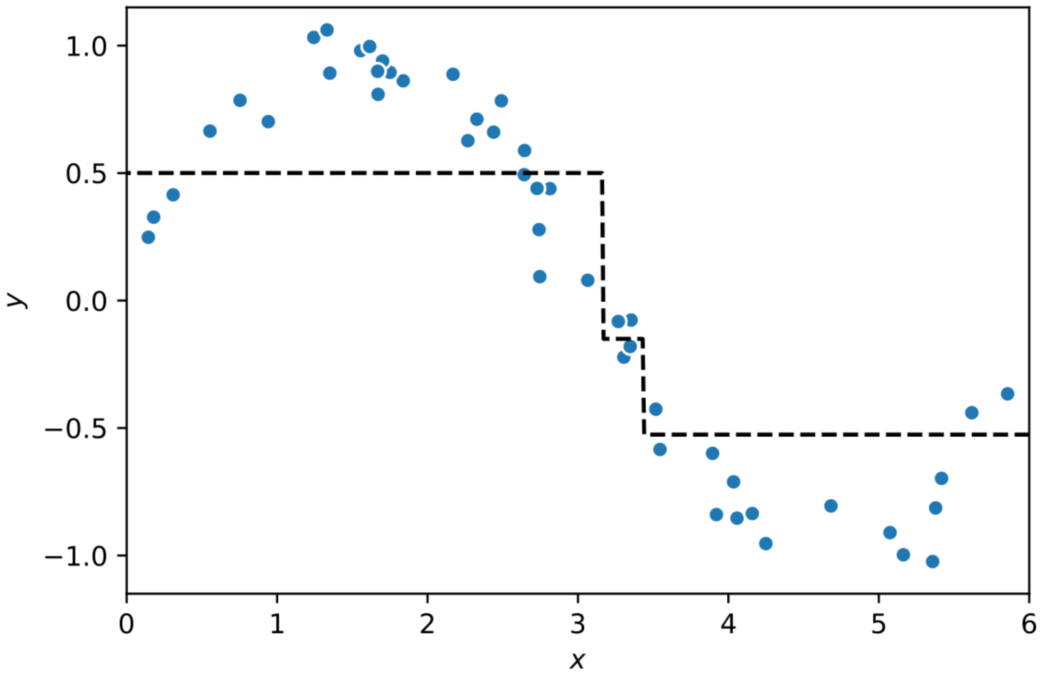


Boosting for regression

1. Initialise $r^{(n)} = y^{(n)}$ for all N training items and set $f(\mathbf{x};\boldsymbol{\theta}) = 0$.
2. for iteration $b = 1$ to B: $\boxed{b=2}$
   (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: $r^{(n)} \leftarrow r^{(n)} - \lambda f_b(\mathbf{x};\boldsymbol{\theta}_b)$
3. Final model: $f(\mathbf{x};\boldsymbol{\theta}) = \sum_{b=1}^{B} \lambda f_b(\mathbf{x};\boldsymbol{\theta}_b)$

$\lambda f_2(\underline{x};\underline{\theta}_2) + \lambda f_1(\underline{x};\underline{\theta}_1)$

Combined model output at iteration $b = 2$



[Compare to 3 slides back]

Boosting for regression

$\lambda f_3(\underline{x}; \underline{\theta}_3) + \lambda f_2(\underline{x}; \theta_2) + \lambda f_3(\underline{x}; \underline{\theta}_3)$

1. Initialise $r^{(n)} = y^{(n)}$ for all N training items and set $f(\mathbf{x}; \boldsymbol{\theta}) = 0$.

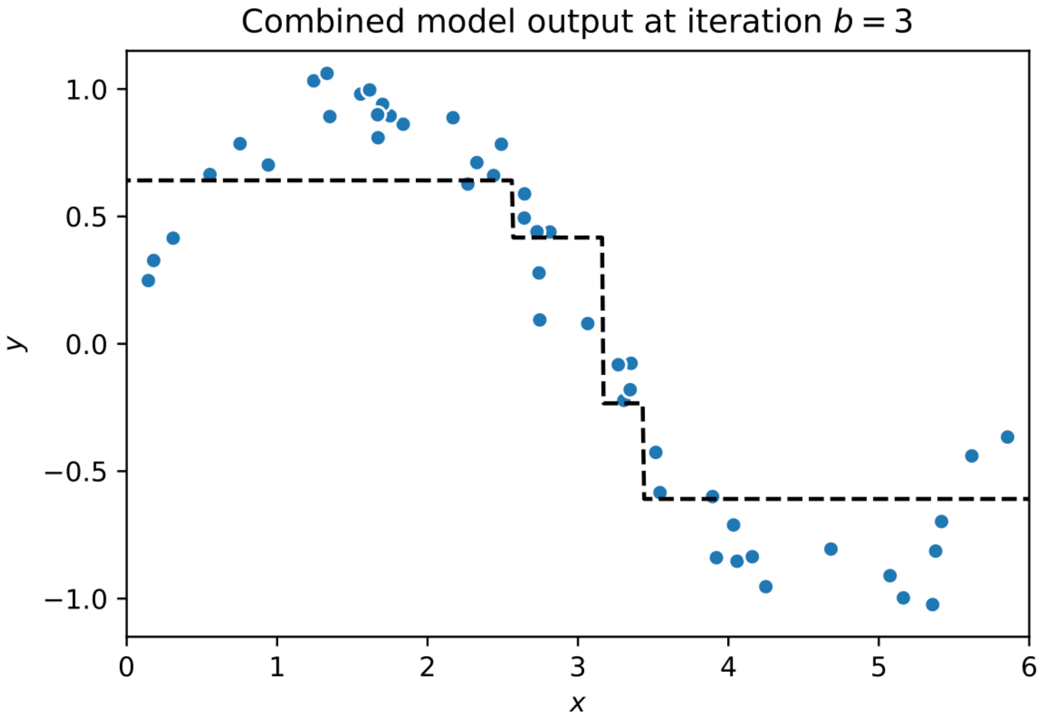
2. for iteration $b = 1$ to B: $\boxed{b=3}$

   (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:
   $r^{(n)} \leftarrow r^{(n)} - \lambda f_b(\mathbf{x}; \boldsymbol{\theta}_b)$

3. Final model: $f(\mathbf{x}; \boldsymbol{\theta}) = \sum_{b=1}^{B} \lambda f_b(\mathbf{x}; \boldsymbol{\theta}_b)$



Boosting for regression

1. Initialise $r^{(n)} = y^{(n)}$ for all N training items and set $f(\mathbf{x};\boldsymbol{\theta}) = 0$.

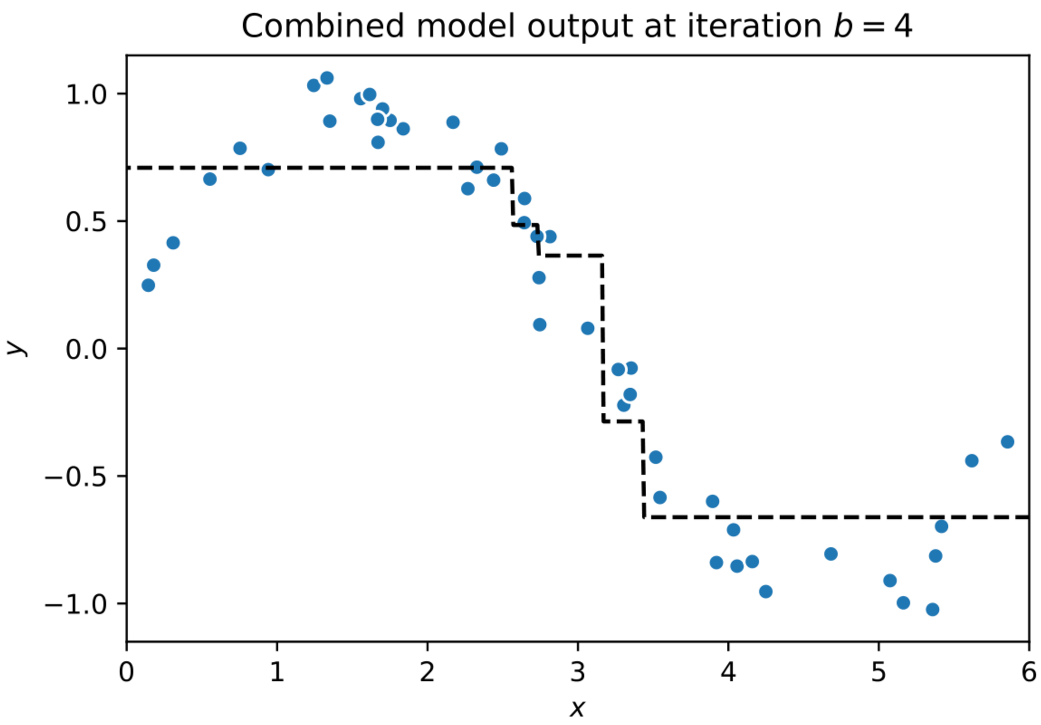
2. for iteration $b = 1$ to B:

   (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: $r^{(n)} \leftarrow r^{(n)} - \lambda f_b(\mathbf{x};\boldsymbol{\theta}_b)$

3. Final model: $f(\mathbf{x};\boldsymbol{\theta}) = \sum_{b=1}^{B} \lambda f_b(\mathbf{x};\boldsymbol{\theta}_b)$



Boosting for regression

1. Initialise $r^{(n)} = y^{(n)}$ for all N training items and set $f(\mathbf{x};\boldsymbol{\theta}) = 0$.

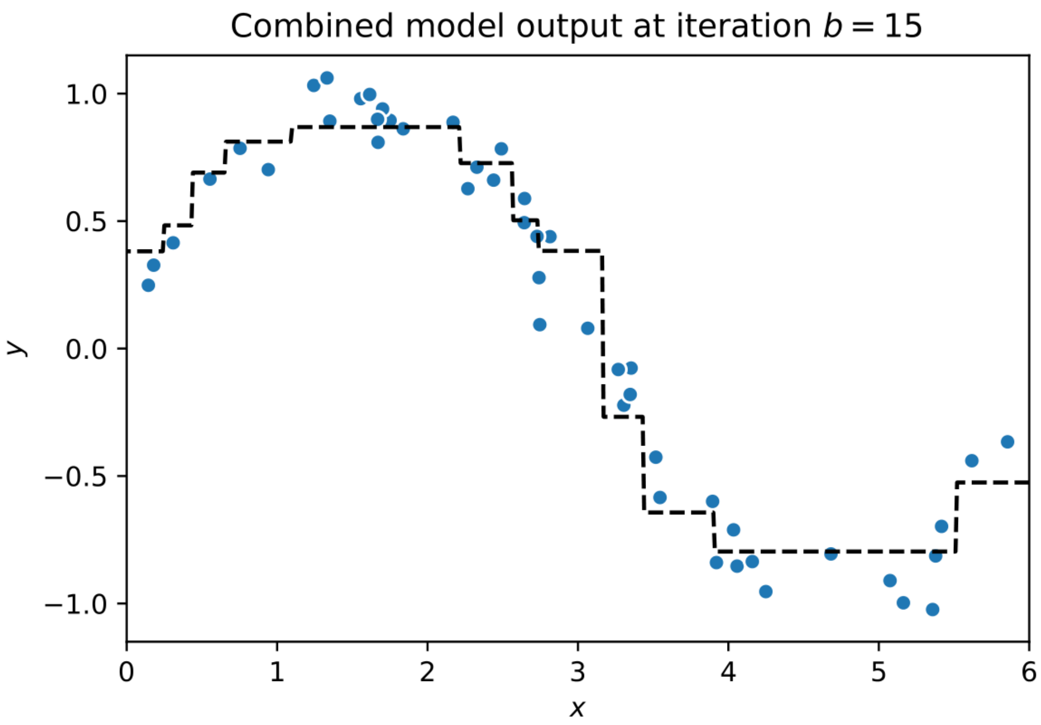
2. for iteration $b = 1$ to B:

   (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:
   $r^{(n)} \leftarrow r^{(n)} - \lambda f_b(\mathbf{x};\boldsymbol{\theta}_b)$

3. Final model: $f(\mathbf{x};\boldsymbol{\theta}) = \sum_{b=1}^{B} \lambda f_b(\mathbf{x};\boldsymbol{\theta}_b)$



Ensemble methods

AdaBoost: Boosting for classification

Combine weak (slightly better than random)

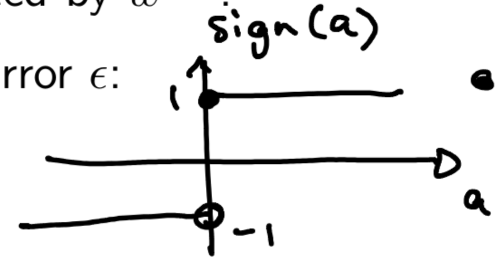
Herman Kamper

http://www.kamperh.com/

- Binary classification
- Building block (principles) for classification models often used in practice

AdaBoost: Boosting for classification

1. Initialise training item weights $w^{(n)} = 1$ for all N training items.
2. 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:
   $$\lambda_b = \frac{1}{2} \log\left(\frac{1-\epsilon}{\epsilon}\right)$$
   (c) Update training item weights:
   $$w^{(n)} \leftarrow w^{(n)} e^{-\lambda_b} \text{ if } f_b(\mathbf{x}^{(n)}; \boldsymbol{\theta}_b) \text{ correct}$$
   $$w^{(n)} \leftarrow w^{(n)} e^{\lambda_b} \text{ if } f_b(\mathbf{x}^{(n)}; \boldsymbol{\theta}_b) \text{ incorrect}$$
3. Final model: $f(\mathbf{x}; \boldsymbol{\theta}) = \text{sign}\left[\sum_{b=1}^{B} \lambda_b f_b(\mathbf{x}; \boldsymbol{\theta}_b)\right]$



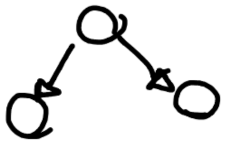
Setting:

- Binary classification: $y \in \{-1, 1\}$
- Going to train B models, each $f_b(\underline{x}; \underline{\theta}_b) \in \{-1, 1\}$
- Going to combine weighted votes:
$$f(\underline{x}; \underline{\theta}) = \text{sign}\left[\sum_{b=1}^{B} \lambda_b f_b(\underline{x}; \underline{\theta}_b)\right]$$

$w^{(1)}$ $\boxed{- (\underline{x}^{(1)})^T - \quad y^{(1)}}$
$w^{(2)}$ $- (\underline{x}^{(2)}) - \quad y^{(2)}$
$\vdots$ $\quad\vdots\quad\vdots$
$w^{(N)}$

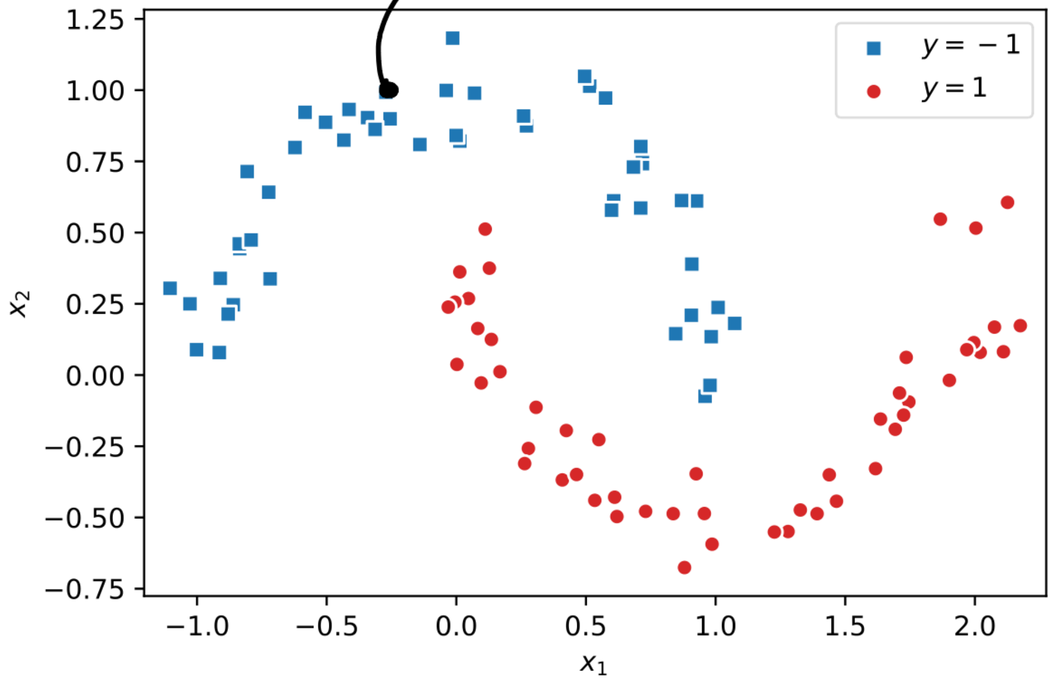
AdaBoost: Boosting for classification

Weak classifier:

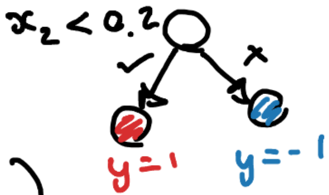


1. Initialise training item weights $w^{(n)} = 1$ for all N training items.
2. for iteration $b = 1$ to B:
   (a) Fit model $f_b(\mathbf{x}; \boldsymbol{\theta}_b)$ so that it minimises classification error weighted by $w^{(n)}$.
   (b) Set model weight using error ϵ:
   $$\lambda_b = \frac{1}{2} \log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

$w^{(25)} = 1$

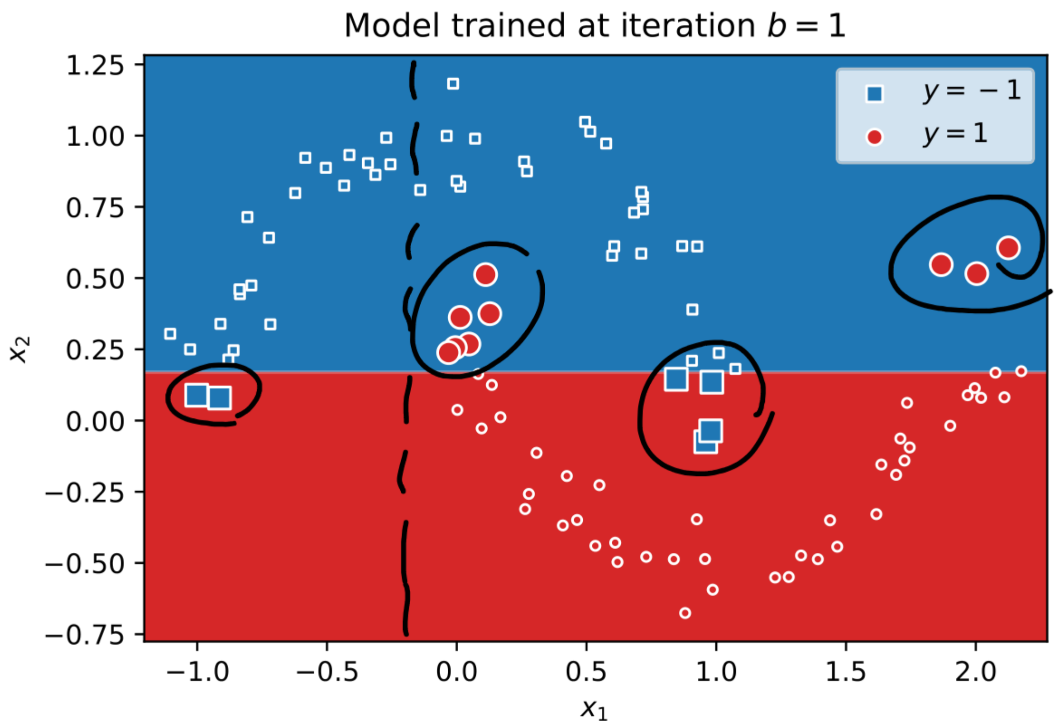


AdaBoost: Boosting for classification



$f_1(\underline{x}; \underline{\theta}_1)$

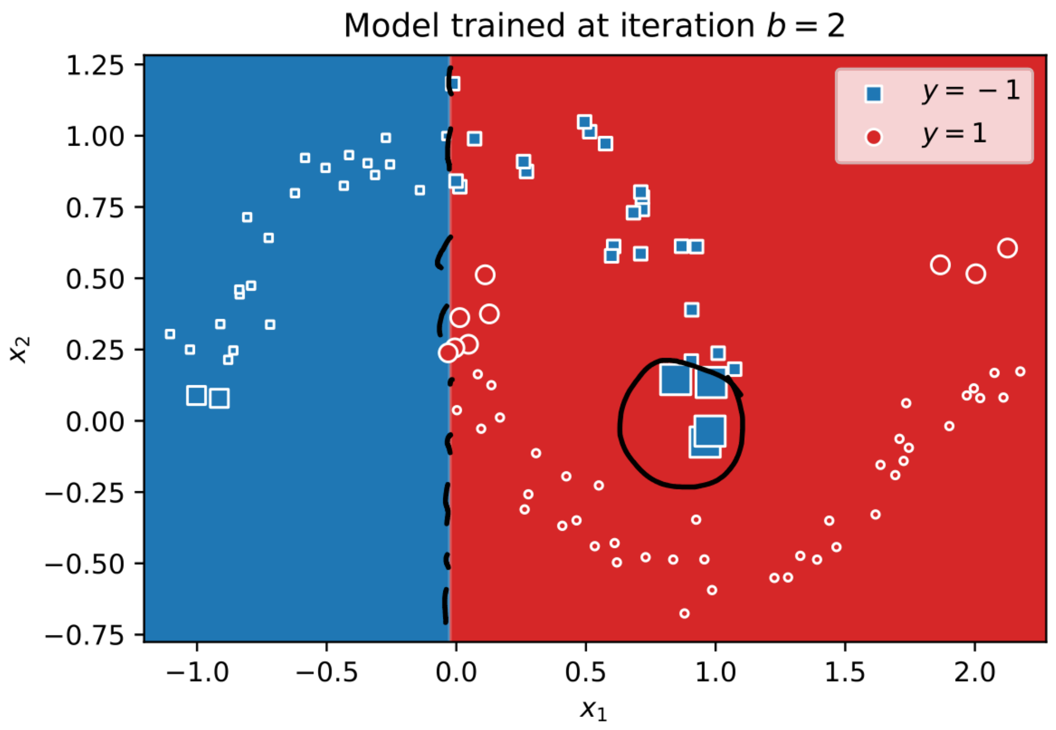
1. Initialise training item weights $w^{(n)} = 1$ for all N training items.
2. for iteration $\boxed{b = 1}$ to B:
   (a) Fit model $f_b(\mathbf{x}; \boldsymbol{\theta}_b)$ so that it minimises classification error weighted by $w^{(n)}$.
   (b) Set model weight using error ϵ:
   $$\lambda_b = \frac{1}{2} \log\left(\frac{1-\epsilon}{\epsilon}\right) \qquad \lambda_1 = 0.87$$
   (c) Update training item weights:
   $$w^{(n)} \leftarrow w^{(n)} e^{-\lambda_b} \text{ if } f_b(\mathbf{x}^{(n)}; \boldsymbol{\theta}_b) \text{ \underline{correct}}$$
   $$w^{(n)} \leftarrow w^{(n)} e^{\lambda_b} \text{ if } f_b(\mathbf{x}^{(n)}; \boldsymbol{\theta}_b) \text{ \underline{incorrect}}$$
3. Final model: $f(\mathbf{x}; \boldsymbol{\theta}) = \operatorname{sign}\left[\sum_{b=1}^{B} \lambda_b f_b(\mathbf{x}; \boldsymbol{\theta}_b)\right]$



AdaBoost: Boosting for classification

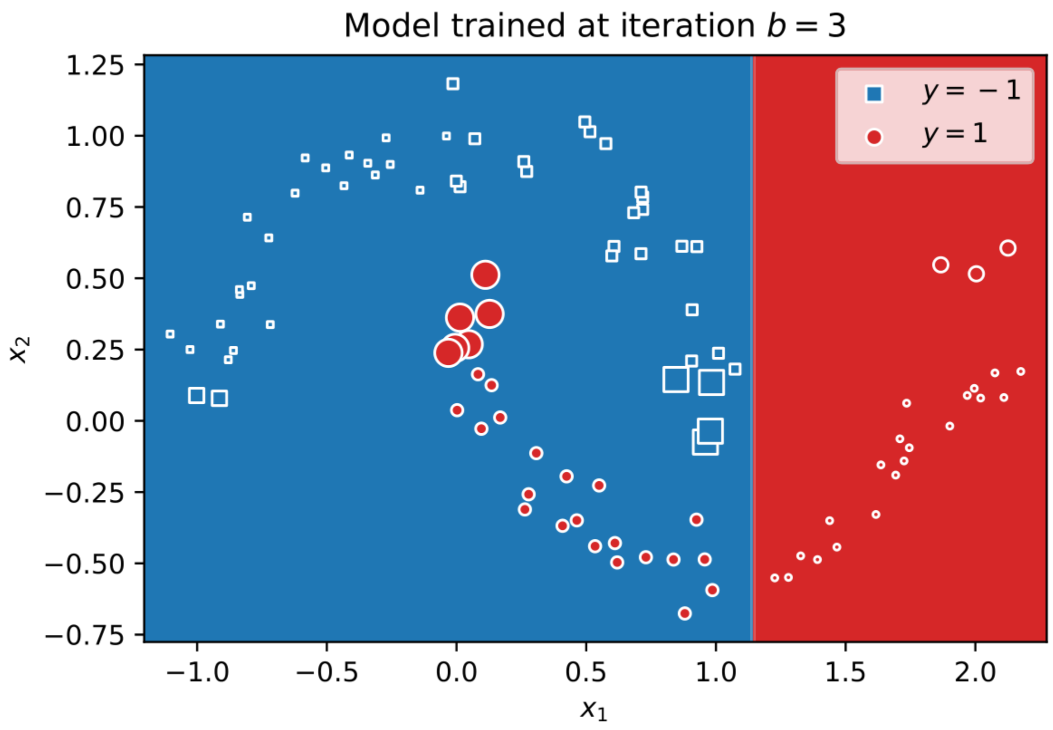
1. Initialise training item weights $w^{(n)} = 1$ for all N training items.
2. for iteration $b = 1$ to B: $\boxed{b=2}$
   (a) Fit model $f_b(\mathbf{x}; \boldsymbol{\theta}_b)$ so that it minimises classification error weighted by $w^{(n)}$.
   (b) Set model weight using error ϵ:
   $$\lambda_b = \frac{1}{2} \log\left(\frac{1-\epsilon}{\epsilon}\right) \qquad \lambda_2 = 0.52$$
   (c) Update training item weights:
   $w^{(n)} \leftarrow w^{(n)} e^{-\lambda_b}$ if $f_b(\mathbf{x}^{(n)}; \boldsymbol{\theta}_b)$ correct
   $w^{(n)} \leftarrow w^{(n)} e^{\lambda_b}$ if $f_b(\mathbf{x}^{(n)}; \boldsymbol{\theta}_b)$ incorrect
3. Final model: $f(\mathbf{x}; \boldsymbol{\theta}) = \text{sign}\left[\sum_{b=1}^{B} \lambda_b f_b(\mathbf{x}; \boldsymbol{\theta}_b)\right]$

$f_2(\underline{x}; \underline{\theta}_2)$



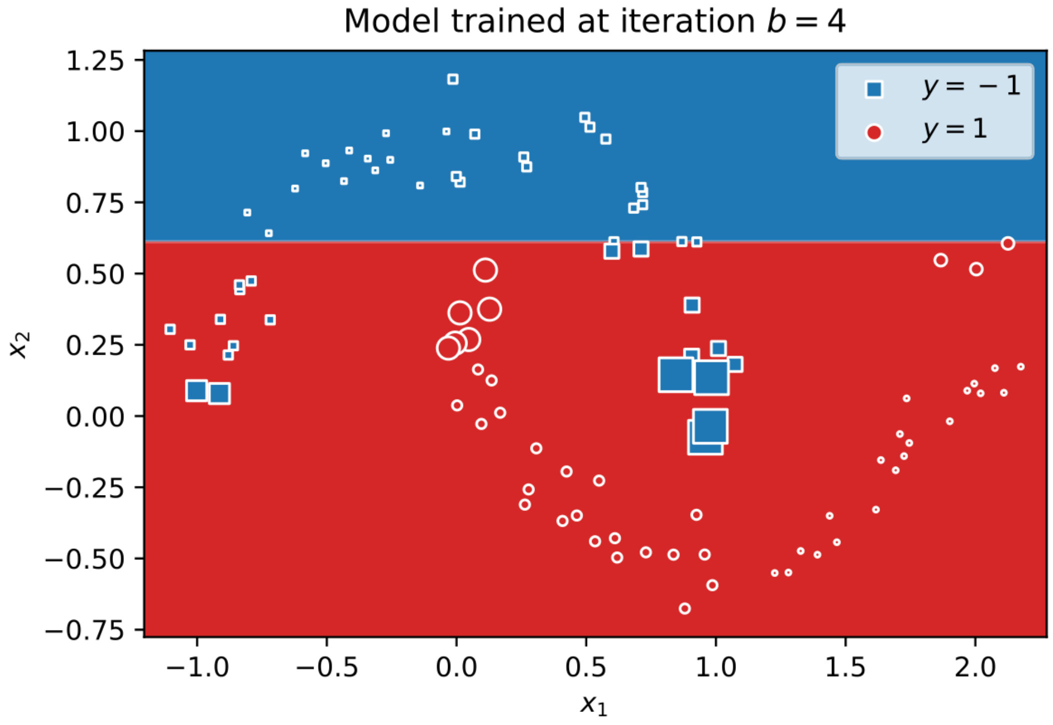
AdaBoost: Boosting for classification

1. Initialise training item weights $w^{(n)} = 1$ for all N training items.
2. for iteration $b = 1$ to B: $\boxed{b=3}$
   (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:
   $$\lambda_b = \frac{1}{2} \log \left(\frac{1-\epsilon}{\epsilon} \right)$$
   (c) Update training item weights:
   $$w^{(n)} \leftarrow w^{(n)} e^{-\lambda_b} \text{ if } f_b(\mathbf{x}^{(n)}; \boldsymbol{\theta}_b) \text{ correct}$$
   $$w^{(n)} \leftarrow w^{(n)} e^{\lambda_b} \text{ if } f_b(\mathbf{x}^{(n)}; \boldsymbol{\theta}_b) \text{ incorrect}$$
3. Final model: $f(\mathbf{x}; \boldsymbol{\theta}) = \operatorname{sign} \left[\sum_{b=1}^{B} \lambda_b f_b(\mathbf{x}; \boldsymbol{\theta}_b) \right]$



AdaBoost: Boosting for classification

1. Initialise training item weights $w^{(n)} = 1$ for all N training items.
2. 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:
   $$\lambda_b = \frac{1}{2} \log \left(\frac{1 - \epsilon}{\epsilon} \right)$$
   (c) Update training item weights:
   $w^{(n)} \leftarrow w^{(n)} e^{-\lambda_b}$ if $f_b(\mathbf{x}^{(n)}; \boldsymbol{\theta}_b)$ correct
   $w^{(n)} \leftarrow w^{(n)} e^{\lambda_b}$ if $f_b(\mathbf{x}^{(n)}; \boldsymbol{\theta}_b)$ incorrect
3. Final model: $f(\mathbf{x}; \boldsymbol{\theta}) = \operatorname{sign} \left[\sum_{b=1}^{B} \lambda_b f_b(\mathbf{x}; \boldsymbol{\theta}_b) \right]$



AdaBoost: Boosting for classification

$f(\underline{x}; \underline{\theta}) = \lambda_1 f_1(\underline{x}; \underline{\theta}_1) + \lambda_2 f_2(\underline{x}; \underline{\theta}_2) + \ldots + \lambda_{20} f_{20}(\underline{x}; \underline{\theta}_{20})$

1. Initialise training item weights $w^{(n)} = 1$ for all N training items.

2. 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:

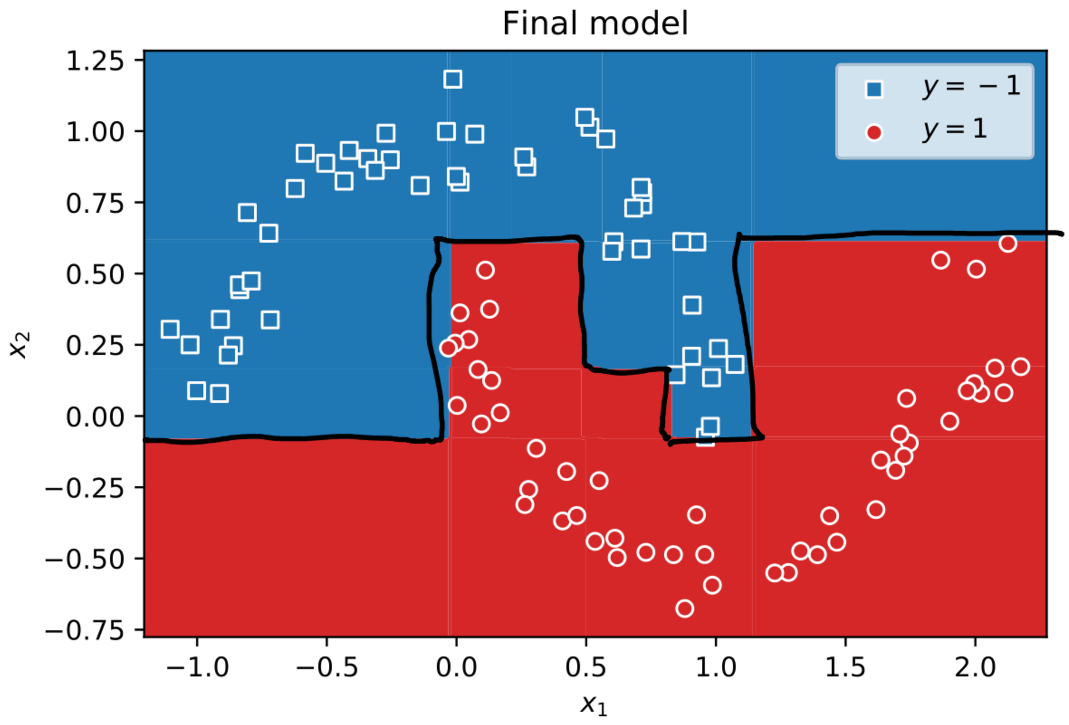
   $$\lambda_b = \frac{1}{2} \log \left(\frac{1 - \epsilon}{\epsilon} \right)$$

   (c) Update training item weights:

   $w^{(n)} \leftarrow w^{(n)} e^{-\lambda_b}$ if $f_b(\mathbf{x}^{(n)}; \boldsymbol{\theta}_b)$ correct

   $w^{(n)} \leftarrow w^{(n)} e^{\lambda_b}$ if $f_b(\mathbf{x}^{(n)}; \boldsymbol{\theta}_b)$ incorrect

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



AdaBoost: Boosting for classification

1. Initialise training item weights $w^{(n)} = 1$ for all N training items.
2. 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:
   $$\lambda_b = \frac{1}{2} \log\left(\frac{1-\epsilon}{\epsilon}\right)$$
   (c) Update training item weights:
   $w^{(n)} \leftarrow w^{(n)} e^{-\lambda_b}$ if $f_b(\mathbf{x}^{(n)}; \boldsymbol{\theta}_b)$ correct
   $w^{(n)} \leftarrow w^{(n)} e^{\lambda_b}$ if $f_b(\mathbf{x}^{(n)}; \boldsymbol{\theta}_b)$ incorrect
3. Final model: $f(\mathbf{x}; \boldsymbol{\theta}) = \operatorname{sign}\left[\sum_{b=1}^{B} \lambda_b f_b(\mathbf{x}; \boldsymbol{\theta}_b)\right]$

(a) Minimise $\sum_{n=1}^{N} w^{(n)} \mathbb{I}\{y^{(n)} \neq f_b(\underline{x}^{(n)}; \underline{\theta}_b)\}$

(b) $\epsilon = \dfrac{\sum_{n=1}^{N} w^{(n)} \mathbb{I}\{y^{(n)} \neq f_b(\underline{x}^{(n)}; \underline{\theta}_b)\}}{\sum_{n=1}^{N} w^{(n)}}$

If $\epsilon \to 0$ (very good classifier)
then λ_b big
If $\epsilon \to \frac{1}{2}$ (bad/random classifier)
then $\lambda_b \to 0$

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

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

AdaBoost: Boosting for classification

1. Initialise training item weights $w^{(n)} = 1$ for all N training items.
2. 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:
   $$\lambda_b = \frac{1}{2} \log\left(\frac{1-\epsilon}{\epsilon}\right)$$
   (c) Update training item weights:
   $w^{(n)} \leftarrow w^{(n)} e^{-\lambda_b}$ if $f_b(\mathbf{x}^{(n)}; \boldsymbol{\theta}_b)$ correct
   $w^{(n)} \leftarrow w^{(n)} e^{\lambda_b}$ if $f_b(\mathbf{x}^{(n)}; \boldsymbol{\theta}_b)$ incorrect
3. Final model: $f(\mathbf{x}; \boldsymbol{\theta}) = \operatorname{sign}\left[\sum_{b=1}^{B} \lambda_b f_b(\mathbf{x}; \boldsymbol{\theta}_b)\right]$

Further reading:

- Raúl Rojas, "AdaBoost and the superbowl of classifiers: A tutorial introduction to adaptive boosting", 2009.
- ESL, Chapter 10