

# Föreläsning 4 - Naive Bayes, k-närmaste grannar, ensemblemetoder

Josef Wilzen

2022-08-30

# Outline

- 1 Introduktion
- 2 k-närmaste grannar
- 3 Bayesianska klassificerare
- 4 Ensemblemetoder

# Introduktion

## Ämnen

- K-närmaste grannar
- Naive Bayes
- Ensemblemetoder
  - ▶ Bagging
  - ▶ Random forest
  - ▶ Boosting
- Sammanfattning

# K-närmaste grannar

- Icke-parameterisk metod
- Klassificering och regression
  - ▶  $X = (x_1, \dots, x_p), y$
- Prediktion av testpunkt  $X_{test}$ : beror bara på de k-närmaste grannarna till testpunkten
- Exempel på:
  - ▶ Lazy learning
  - ▶ Prototype learning
  - ▶ kernelmetod, med en uniform kernel
  - ▶ lokal metod

# K-närmaste grannar

- Vi måste definera ett avståndsmått
  - ▶ Euklidiskt avstånd

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{k=1}^n (x_k - y_k)^2}$$

- ▶ Manhattan avstånd (Taxicab)

$$d(\mathbf{x}, \mathbf{y}) = \sum_{k=1}^n |x_k - y_k|$$

- ▶ Se kapitel 2.4 i **IDM** för fler exempel

# K-närmaste grannar klassificering

---

**Algorithm 5.2** The  $k$ -nearest neighbor classification algorithm.

---

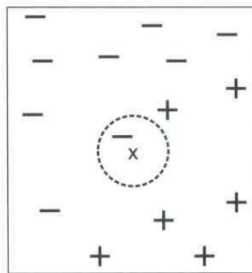
- 1: Let  $k$  be the number of nearest neighbors and  $D$  be the set of training examples.
  - 2: **for** each test example  $z = (\mathbf{x}', y')$  **do**
  - 3:   Compute  $d(\mathbf{x}', \mathbf{x})$ , the distance between  $z$  and every example,  $(\mathbf{x}, y) \in D$ .
  - 4:   Select  $D_z \subseteq D$ , the set of  $k$  closest training examples to  $z$ .
  - 5:    $y' = \underset{v}{\operatorname{argmax}} \sum_{(\mathbf{x}_i, y_i) \in D_z} I(v = y_i)$
  - 6: **end for**
- 

- Majoritet (majority) anges i algoritm 5.2
- Avstånd (weighted distance):

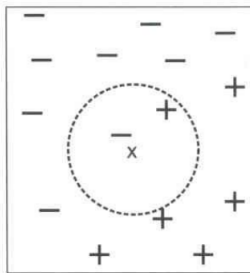
$$y' = \underset{v}{\operatorname{argmax}} \left( \sum_{(\mathbf{x}_i, y_i) \in D_z} w_i I(v = y_i) \right)$$

- Regression: medelvärde/viktat medelvärde av grannarna

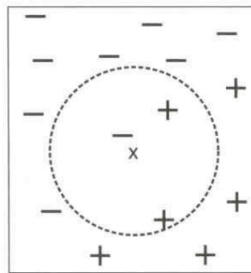
# k-närmarste grannar



(a) 1-nearest neighbor



(b) 2-nearest neighbor



(c) 3-nearest neighbor

# K-närmaste grannar

- Målet med modellen är att prediktera nya observationer
- Påverkas stort av olika skalor – Ett distansmått måste väljas för metoden
- Långsam anpassning – Varje ny observation måste "skapa" en ny modell
- Känslig mot brus – Lokal information används



# K-närmaste grannar

- Val av  $K$  har stor betydelse!
  - ▶ Lättet  $K \rightarrow$  överanpassning, stort  $K \rightarrow$  underanpassning
  - ▶ Vi kan använda korsvalidering för att bestämma  $k$ .
- Producerar godtyckligt utformade beslutsgränser
- Mer än 10 dimensioner i data:
  - ▶ problem
  - ▶ variabelreducerande tekniker, tex PCA

# Bayesianska klassificerare

- Om man vill modellera en icke-deterministisk funktion:
  - ▶ (diet, träning)  $\rightarrow$  (hjärtinfarkt): svårt
  - ▶ (diet, träning)  $\rightarrow \text{Pr}(\text{hjärtinfarkt})$
- Bayes sats:

$$P(Y|\mathbf{X}) = \frac{P(\mathbf{X}|Y)}{P(\mathbf{X})} \cdot P(Y) \propto P(\mathbf{X}|Y) \cdot P(Y)$$

$$\text{posterior} = \frac{\text{likelihood}}{\text{evidence}} \cdot \text{prior} \propto \text{likelihood} \cdot \text{prior}$$

# Kategoriska attribut

- $P(Y = y) = (\text{antalet rader där klassbeteckning är } y) / (\text{totala antalet rader})$
- $P(X_i = x_i | Y = y) = (\text{antalet rader där klassbeteckningen är } y \text{ och attributet är } x) / (\text{antalet rader med klassbeteckningen } y)$

	binary	categorical	continuous	class
Tid	Home Owner	Marital Status	Annual Income	Defaulted Borrower
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

# Kontinuerliga attribut

- Diskretisera data i olika kategorier – Problem:
  - ▶ För få intervall (breda gränser) och man kan missa mycket i aggregeringen.
  - ▶ För många intervall och det blir problem om intervallen har för få observationer.
- Anta en sannolikhetsfördelning för variabeln och skatta parametrarna från träningsmängden
  - ▶ Normalfördelning vanligt

# Grundläggande princip

- Träningsfas: – Skatta sannolikheten  $P(Y|X)$  för alla möjliga  $X$  och  $Y$
- Klassificeringsfas: – Givet  $X'$  skatta klass genom  $Y' = \max_Y P(Y|X')$

# Naiv Bayes klassificerare

Modelantagande:

$$P(\mathbf{X}|Y) = \prod P(X_i|Y)$$

- Vi antar att  $X_i$  är oberoende: likelihooden faktorerar över  $\mathbf{X}$
- Betingade sannolikheter skattas bara för varje  $X_i$  istället för varje kombination av  $\mathbf{X}$

$$P(Y|\mathbf{X}) = \prod P(X_i|Y) P(Y)$$

- Detta ger en enklare, mindre flexibel model, men som går att skatta

# Exempel

- 1 kontinuerlig attribut – 2 klasser
- Hitta beslutsgränsen

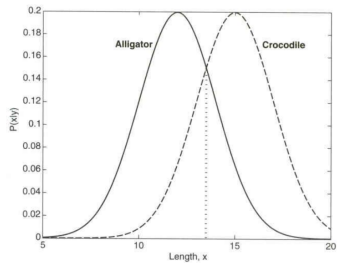


Figure 5.11. Comparing the likelihood functions of a crocodile and an alligator.

# Exempel

Frukter:  $Y$  kan antingen vara äpple, banan eller apelsin

- Egenskaper: färg, form, diameter
- Vi antar att dessa bidrar oberoende till sannolikheten för att  $Y$  är äpple:

$$P(Y = \text{äpple} | X) = P(X_{\text{färg}} | Y) P(X_{\text{form}} | Y) P(X_{\text{diameter}} | Y) P(Y)$$



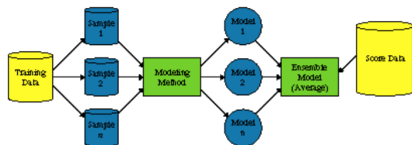
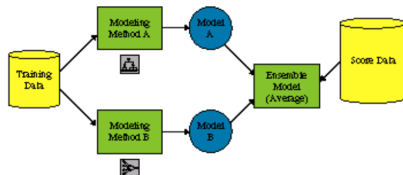
# Egenskaper

- Robusta mot isolerade bruspunkter
- Robusta mot irrelevanta attribut då  $P(X_i|Y)$  blir nästan likformigt fördelad
- Lätt att skatta
- Korrelerade attribut kan väsentligt försämra prestanda
  - ▶ Då behöver vi en mer komplex modell
  - ▶ Simultan sannolikhetsfördelning för likelihooden

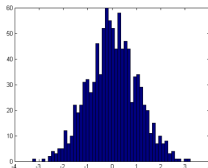
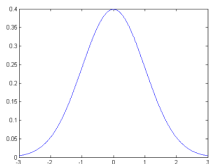
# Ensemblemetoder

Två olika metodfamiljer

- Modellfokuserad
- Datafokuserad
  - ▶ Bootstrapping och Bagging
  - ▶ Boosting



# Bagging och boosting



- Skatta en funktion av den ursprungliga fördelningen  $F(P)$  , ersätts med  $F(P^*)$

# Bootstrapping

- Skapa  $B$  stickprov **med återläggning** från datamängden
- Beräkna  $F(P_k^*)$  där  $k = 1, \dots, B$
- Ex. Skatta  $Var(e^{\bar{x}})$ 
  - ▶ Skapa  $B$  stickprov med återläggning
  - ▶ Skatta  $T_k = e^{\bar{Z}_k}$ , där  $k = 1, \dots, B$  och  $Z_k$  är stickprov  $k$
  - ▶ Beräkna  $Var(T), T = (T_1, T_2, \dots, T_B)$

# Bagging

Bagging = Bootstrap aggregating

Idén:

- Givet en model  $Y = f(\mathbf{X}) + \varepsilon$ , skatta  $E_P(\hat{f}(\mathbf{X}))$ , där  $P$  är fördelningen av  $(X, y)$
- Lösning: Ersätt  $P$  med  $P^*$ 
  - ▶ Skapa  $B$  bootstrap-urval och skatta  $\hat{f}_b(\mathbf{X})$
  - ▶ Skatta  $E_P(\hat{f}(\mathbf{X}))$  genom att ta medelvärdet av bootstrap-funktionerna

$$\hat{f}_{bag}(\mathbf{X}) = \frac{1}{B} \sum_{b=1}^B \hat{f}_b(\mathbf{X})$$

# Bagging – kommentarer

- Sänker variansen av den anpassade funktionen
- Påverkas starkt av kvalitén av modellen, en bra modell blir bättre men en dålig modell blir sämre
- En linjär funktion sammanfaller asymptotiskt med bootstrap-skattningarna då  $B \rightarrow \infty$
- Den anpassade modellen ska vara global!

# Bagging för klassificering

- Givet  $K$  klasser med  $Z = \{Y_i, \mathbf{X}_i, i = 1, \dots, N\}$ , beräkna indikatorfunktion alt. klass-sannolikheter.

$$\hat{f}(x) = \{p_1(x), \dots, p_K(x)\}$$

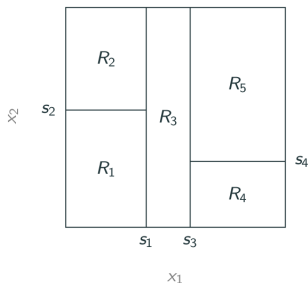
$$\hat{G}(x) = \underset{k}{\operatorname{argmax}} (p_k(x))$$

- Skatta baggingestimat  $\hat{f}_{bag}(\mathbf{X}) = \frac{1}{B} \sum_{b=1}^B \hat{f}_b(\mathbf{X})$  och prediktera klassbeteckning

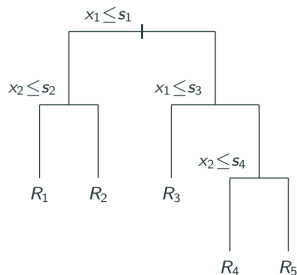
# Classification And Regression Trees

- CART: Partition the input space using recursive binary splitting
  - ▶ Classification: Majority vote within the region.
  - ▶ Regression: Mean of training data within the region.

Partitioning of input space



Tree representation





# Förbättra CART

Flexibiliteten/komplexiteten för trädmodeller beror på träddjupet

- För att få liten bias så vill vi ha ett djupt träd
- Men det leder till hög varians!

Prestationsförmågan av (vanliga) CART är ofta otillräcklig!

# Förbättra CART

- Efterbeskärning:
  - ▶ Skapa djupt träd (liten bias) → beskär till ett mindre (reducera variansen)
- Ensemblemetoder: ta genomsnitt över många trädmodeller
  - ▶ Bagging och Random Forest
  - ▶ Boosted trees

# Random forests

- Bagging kan ge stora förbättringar för trädmodeller!
- Men...
  - ▶ De  $B$  bootstrap-urvalen är korrelerade (stort överlapp av observationer)
  - ▶ Reduktionen i varians blir liten när vi tar medelvärde över korrelerade dataset
- Idé: avkorrelera (decorrelate) de  $B$  trädmodellerna genom att göra slumpmässiga ändringar på modellerna.

# Random forests

- Använd bagging för att skatta  $B$  träd
  - ▶ Vid varje uppdelning/regel: endast en slumpmässig delmängd  $q \leq p$  av de förklarande variablerna används.
- Tumregel:  $q = \sqrt{p}$  vid klassificering,  $q = p/3$  vid regression<sup>1</sup>.
- Vad händer om  $q = p$ ?

---

<sup>1</sup>Proposed by Leo Breiman, inventor of random forests. 

# Random forests

---

**Algorithm** Random forest for regression

---

1. For  $b = 1$  to  $B$  (*can run in parallel*)
  - (a) Draw a bootstrap data set  $\tilde{\mathcal{T}}$  of size  $n$  from  $\mathcal{T}$ .
  - (b) Grow a regression tree by repeating the following steps until a minimum node size is reached:
    - i. Select  $q$  out of the  $p$  input variables uniformly at random.
    - ii. Find the variable  $x_j$  among the  $q$  selected, and the corresponding split point  $s$ , that minimizes the squared error.
    - iii. Split the node into two children with  $\{x_j \leq s\}$  and  $\{x_j > s\}$ .
2. Final model is the average the  $B$  ensemble members,

$$\hat{y}_*^{\text{rf}} = \frac{1}{B} \sum_{b=1}^B \tilde{y}_*^b.$$

# Random forest

Slumpmässigt val av variabler:

- - Minskar bias, men ofta mycket långsamt
- - Läger till varians till varje träd
- + Avkorrelerar träden

Ofta dominerar den avkorrelerarand effekten  $\rightarrow$  MSE minskar på testdata

# Random forest

Beräkningsmässiga fördelar:

- Lätt att parallellisera
- $q < p$  minskar kostnad vid varje uppdelning
  - ▶ Bra vid många variabler!
- Inte så många hyperparameterar: funkar ofta bra!
- Kan ibland bli problem med starkt korrelerade variabler

# Boosting

En enkel modell kan vanligtvis fånga vissa aspekter av input-output-relationen.

Kan vi sedan lära oss en ensemble av "svaga modeller", som var och en beskriver någon del av  $X - Y$  förhållandet och sedan kombinera dessa till en "stark modell"?

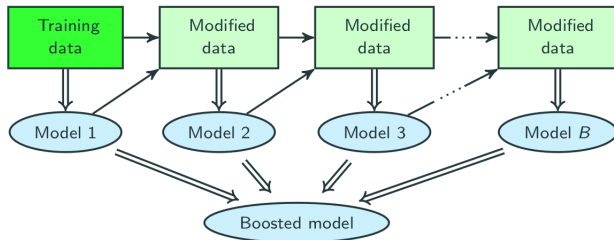
Hur gör vi då?



# Boosting

- Lär sig sekventiellt en ensemble av "svaga modeller"
- Kombinera dessa till en "stark modell"
- Generel approach, kan användas till godtycklig metod inom övervakad inlärning.
- Mycket framgångsrik idé inom maskininlärning!

# Boosting



Modellerna skattas sekventiellt, på ett sådan sätt att varje model försöker fixa misstagen som har gjorts med tidigare modeller.

# Binary classification

We will restrict our attention to binary classification.

- Class labels are  $-1$  and  $1$ , i.e.  $y \in \{-1, 1\}$ .
  - We have access to some (weak) base classifier, e.g. a classification tree.
- 

*Note.* Using labels  $-1$  and  $1$  is mathematically convenient as it allows us to express a majority vote between  $B$  classifiers  $\hat{y}^1(\mathbf{x}), \dots, \hat{y}^B(\mathbf{x})$  as

$$\text{sign}\left(\sum_{b=1}^B \hat{y}^b(\mathbf{x})\right) = \begin{cases} +1 & \text{if more plus-votes than minus-votes,} \\ -1 & \text{if more minus-votes than plus-votes.} \end{cases}$$

# Boosting procedure (for classification)

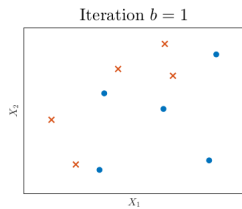
## Boosting procedure:

1. Assign weights  $w_i^1 = 1/n$  to all data points.
  2. For  $b = 1$  to  $B$ 
    - (a) Train a weak classifier  $\hat{y}^b(\mathbf{x})$  on the **weighted training data**  $\{(\mathbf{x}_i, y_i, w_i^b)\}_{i=1}^n$ .
    - (b) *Update the weights*  $\{w_i^{b+1}\}_{i=1}^n$  from  $\{w_i^b\}_{i=1}^n$ :
      - i. Increase weights for all points misclassified by  $\hat{y}^b(\mathbf{x})$ .
      - ii. Decrease weights for all points correctly classified by  $\hat{y}^b(\mathbf{x})$ .
- 

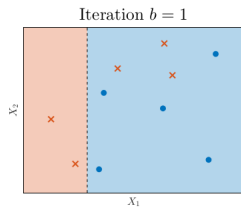
The predictions of the  $B$  classifiers,  $\hat{y}^1(\mathbf{x}), \dots, \hat{y}^B(\mathbf{x})$ , are combined using a **weighted** majority vote:

$$\hat{y}_{\text{boost}}^B(\mathbf{x}) = \text{sign} \left( \sum_{b=1}^B \alpha^b \hat{y}^b(\mathbf{x}) \right).$$

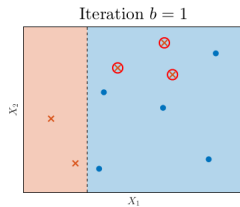
# Boosting illustration



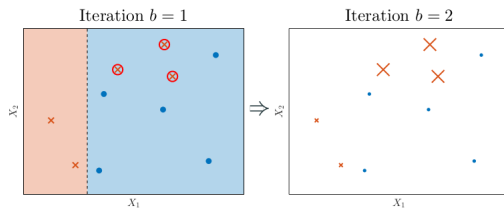
# Boosting illustration



# Boosting illustration

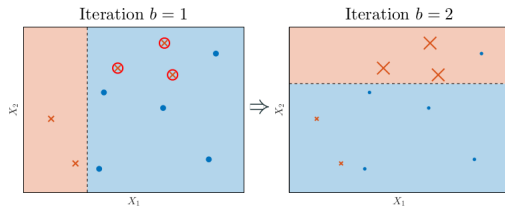


# Boosting illustration

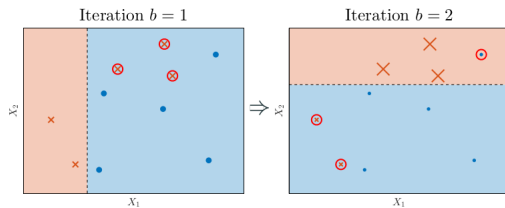




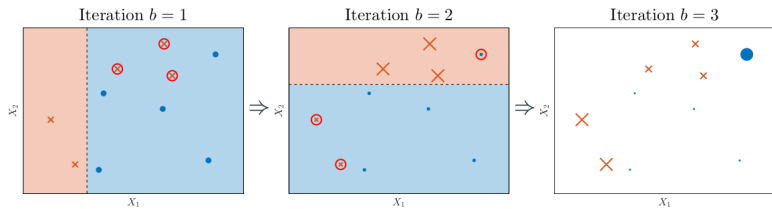
# Boosting illustration



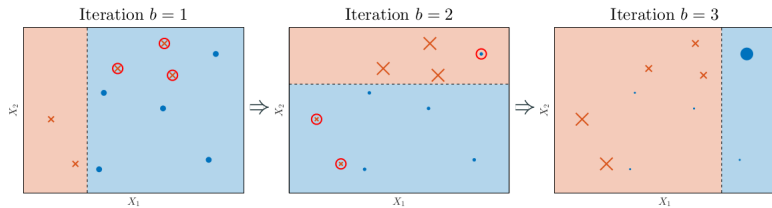
# Boosting illustration



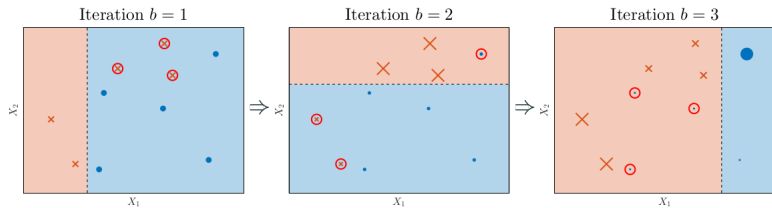
# Boosting illustration



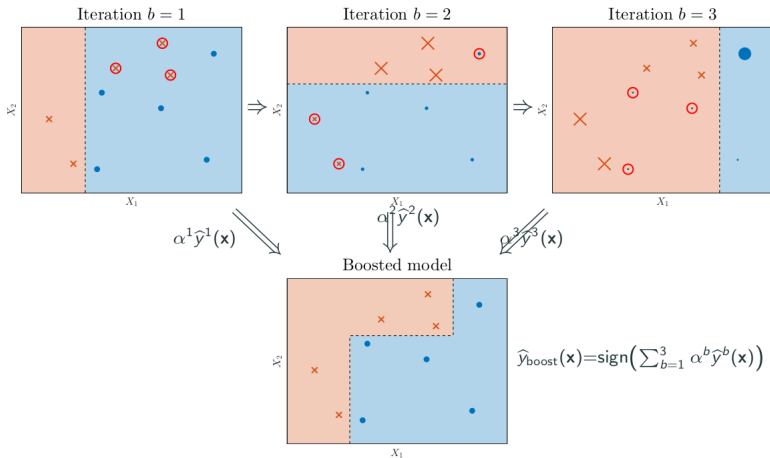
# Boosting illustration



# Boosting illustration



# Boosting illustration



# Tekniska detaljer

- 1 Hur ska vi vikta om data?
- 2 Hur ska vi vikta koefficienterna  $\alpha^1, \alpha^2, \dots, \alpha^B$ ?

Olika boostingalgoritmer svarar olika på dessa frågor. AdaBoost: var den första praktiska algoritmen, svarade på (1) och (2) genom att minimera exponentialförlust

# AdaBoost pseudo-code

## AdaBoost:

1. Assign weights  $w_i^1 = 1/n$  to all data points.
2. For  $b = 1$  to  $B$ 
  - (a) Train a weak classifier  $\hat{y}^b(\mathbf{x})$  on the **weighted training data**  $\{(\mathbf{x}_i, y_i, w_i^b)\}_{i=1}^n$ .
  - (b) *Update the weights*  $\{w_i^{b+1}\}_{i=1}^n$  from  $\{w_i^b\}_{i=1}^n$ :
    - i. Compute  $E_{\text{train}}^b = \sum_{i=1}^n w_i^b \mathbb{I}\{y_i \neq \hat{y}^b(\mathbf{x}_i)\}$
    - ii. Compute  $\alpha^b = 0.5 \log((1 - E_{\text{train}}^b)/E_{\text{train}}^b)$ .
    - iii. Compute  $w_i^{b+1} = w_i^b \exp(-\alpha^b y_i \hat{y}^b(\mathbf{x}_i))$ ,  $i = 1, \dots, n$
    - iv. *Normalize*. Set  $w_i^{b+1} \leftarrow w_i^{b+1} / \sum_{j=1}^n w_j^{b+1}$ , for  $i = 1, \dots, n$ .
3. Output  $\hat{y}_{\text{boost}}^B(\mathbf{x}) = \text{sign}\left(\sum_{b=1}^B \alpha^b \hat{y}^b(\mathbf{x})\right)$ .

Y. Freund and R. E. Schapire. **Experiments with a New Boosting Algorithm**. *Proceedings of the 13th International Conference on Machine Learning (ICML)* Bari, Italy, 1996.



2003 Gödel Prize



# Boosting för regressionsträd

---

**Algorithm 8.2** *Boosting for Regression Trees*

---

1. Set  $\hat{f}(x) = 0$  and  $r_i = y_i$  for all  $i$  in the training set.
2. For  $b = 1, 2, \dots, B$ , repeat:
  - (a) Fit a tree  $\hat{f}^b$  with  $d$  splits ( $d + 1$  terminal nodes) to the training data  $(X, r)$ .
  - (b) Update  $\hat{f}$  by adding in a shrunk version of the new tree:

$$\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}^b(x). \quad (8.10)$$

- (c) Update the residuals,

$$r_i \leftarrow r_i - \lambda \hat{f}^b(x_i). \quad (8.11)$$

3. Output the boosted model,

$$\hat{f}(x) = \sum_{b=1}^B \lambda \hat{f}^b(x). \quad (8.12)$$

---

# Boosting

Andra varianter:

- Gradient boosting
  - ▶ XGboost
  - ▶ LightGBM
  - ▶ CatBoost
- Vinner ofta tävlingar i maskininlärning

Bagging	Boosting
Learns base models in parallel	Learns base models sequentially
Uses bootstrapped datasets	Uses reweighted datasets
Does not overfit as $B$ becomes large	Can overfit as $B$ becomes large
Reduces variance but not bias (requires deep trees as base models)	Also reduces bias! (works well with shallow trees)

# Avslut

- Frågor? Kommentarer?
- Kurshemsidan
- Labben