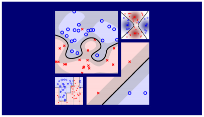
Machine Learning Techniques
(機器學習技法)



Lecture 8: Adaptive Boosting

Hsuan-Tien Lin (林軒田)

htlin@csie.ntu.edu.tw

Department of Computer Science
& Information Engineering

National Taiwan University
(國立台灣大學資訊工程系)

Roadmap

1. Embedding Numerous Features: Kernel Models
2. Combining Predictive Features: Aggregation Models

Lecture 7: Blending and Bagging

blending known diverse hypotheses **uniformly**, **linearly**, or even **non-linearly**; obtaining diverse hypotheses from **bootstrapped data**

Lecture 8: Adaptive Boosting

- Motivation of Boosting
- Diversity by Re-weighting
- Adaptive Boosting Algorithm
- Adaptive Boosting in Action

3. Distilling Implicit Features: Extraction Models

Apple Recognition Problem

- is this a picture of an apple?
- say, want to teach a class of **6 year olds**
- gather photos under CC-BY-2.0 license on Flicker (**thanks to the authors below!**)

(APAL stands for Apple and Pear Australia Ltd)

Dan Foy `https://flic.kr/p/jNQ55`

APAL `https://flic.kr/p/jzP1VB`

adrianbartel `https://flic.kr/p/bdy2hZ`

ANdrzej cH. `https://flic.kr/p/51DKA8`

Stuart Webster `https://flic.kr/p/9C3Ybd`

nachans `https://flic.kr/p/9XD7Ag`

APAL `https://flic.kr/p/jzRe4u`

Jo Jakeman `https://flic.kr/p/7jwtGp`

APAL `https://flic.kr/p/jzPYNr`

APAL `https://flic.kr/p/jzScif`

Apple Recognition Problem

- is this a picture of an apple?
- say, want to teach a class of **6 year olds**
- gather photos under CC-BY-2.0 license on Flicker (**thanks to the authors below!**)

Mr. Roboto.

`https://flic.kr/p/i5BN85`

Richard North

`https://flic.kr/p/bHhPkB`

Richard North

`https://flic.kr/p/d8tGou`

Emilian Robert Vicol

`https://flic.kr/p/bpmGXW`

Nathaniel McQueen

`https://flic.kr/p/pZv1Mf`

Crystal

`https://flic.kr/p/kaPYp`

jfh686

`https://flic.kr/p/6vjRFH`

skyseeker

`https://flic.kr/p/2MynV`

Janet Hudson

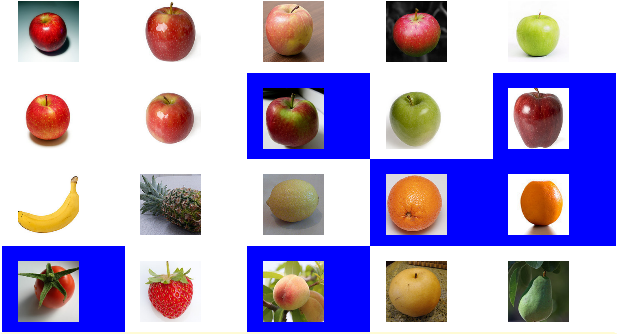
`https://flic.kr/p/7QDBbm`

Rennett Stowe

`https://flic.kr/p/agmnrk`

Our Fruit Class Begins

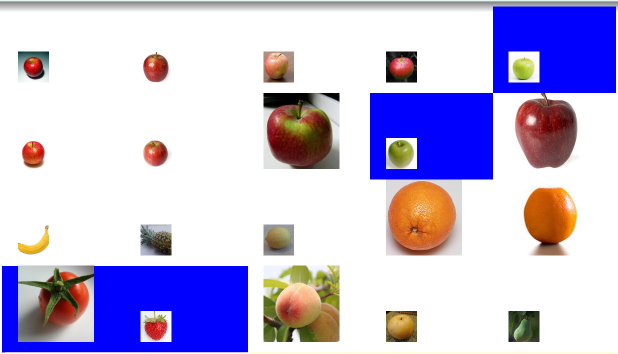
- Teacher: Please look at the pictures of apples and non-apples below. Based on those pictures, how would you describe an apple? Michael?
- Michael: I think apples are **circular**.



(Class): Apples are **circular**.

Our Fruit Class Continues

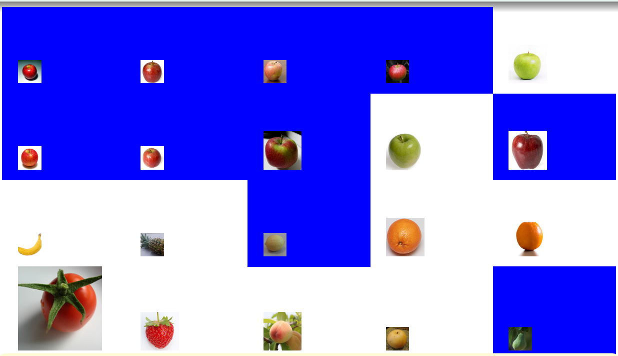
- Teacher: Being circular is a good feature for the apples. However, if you only say circular, you could make several mistakes. What else can we say for an apple? Tina?
- Tina: It looks like apples are **red**.



(Class): Apples are somewhat **circular** and somewhat **red**.

Our Fruit Class Continues More

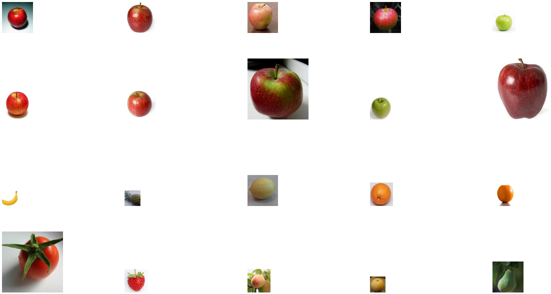
- Teacher: Yes. Many apples are red. However, you could still make mistakes based on circular and red. Do you have any other suggestions, Joey?
- Joey: Apples could also be **green**.



(Class): Apples are somewhat **circular** and somewhat **red** and possibly **green**.

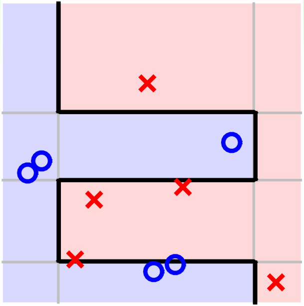
Our Fruit Class Ends

- Teacher: Yes. It seems that apples might be circular, red, green. But you may confuse them with tomatoes or peaches, right? Any more suggestions, Jessica?
- Jessica: Apples have **stems** at the top.



(Class): Apples are somewhat **circular**, somewhat **red**, possibly **green**, and may have **stems** at the top.

Motivation



- students: simple hypotheses g_t (like vertical/horizontal lines)
- (Class): sophisticated hypothesis G (like black curve)
- Teacher: a tactic learning algorithm that **direct the students to focus on key examples**

next: the '**math**' of such an algorithm

Fun Time

Which of the following can help recognize an apple?

1. apples are often circular
2. apples are often red or green
3. apples often have stems at the top
4. all of the above

Fun Time

Which of the following can help recognize an apple?

1. apples are often circular
2. apples are often red or green
3. apples often have stems at the top
4. all of the above

Reference Answer: 4

Congratulations! **You have passed first grade. :-)**

Bootstrapping as Re-weighting Process

$$\mathcal{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), (\mathbf{x}_3, y_3), (\mathbf{x}_4, y_4)\}$$

$$\xRightarrow{\text{bootstrap}} \quad \tilde{\mathcal{D}}_t = \{(\mathbf{x}_1, y_1), (\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), (\mathbf{x}_4, y_4)\}$$

weighted E_{in} on $\mathcal{D}$

$$E_{\text{in}}^{\mathbf{u}}(h) = \frac{1}{4}\sum_{n=1}^{4} u_n^{(t)} \cdot [\![y_n \neq h(\mathbf{x}_n)]\!]$$

$(\mathbf{x}_1, y_1)$, $u_1 = 2$
$(\mathbf{x}_2, y_2)$, $u_2 = 1$
$(\mathbf{x}_3, y_3)$, $u_3 = 0$
$(\mathbf{x}_4, y_4)$, $u_4 = 1$

E_{in} on $\tilde{\mathcal{D}}_t$

$$E_{\text{in}}^{0/1}(h) = \frac{1}{4}\sum_{(\mathbf{x},y)\in\tilde{\mathcal{D}}_t} [\![y \neq h(\mathbf{x})]\!]$$

$(\mathbf{x}_1, y_1)$, $(\mathbf{x}_1, y_1)$
$(\mathbf{x}_2, y_2)$

$(\mathbf{x}_4, y_4)$

each diverse g_t in bagging:
by minimizing bootstrap-weighted error

Weighted Base Algorithm

minimize (regularized)

$$E_{\text{in}}^{\mathbf{u}}(h) = \frac{1}{N} \sum_{n=1}^{N} u_n \cdot \text{err}(y_n, h(\mathbf{x}_n))$$

SVM

$E_{\text{in}}^{\mathbf{u}} \propto C \sum_{n=1}^{N} u_n \widehat{\text{err}}_{\text{SVM}}$ by dual QP

$\Leftrightarrow$ adjusted upper bound

$0 \le \alpha_n \le C u_n$

logistic regression

$E_{\text{in}}^{\mathbf{u}} \propto \sum_{n=1}^{N} u_n \text{err}_{\text{CE}}$ by SGD

$\Leftrightarrow$ sample $(\mathbf{x}_n, y_n)$ with probability proportional to u_n

example-weighted learning:
extension of **class-weighted** learning in Lecture 8 of ML Foundations

Re-weighting for More Diverse Hypothesis

'improving' bagging for binary classification:
how to re-weight for **more diverse hypotheses**?

$$g_t \leftarrow \operatorname*{argmin}_{h \in \mathcal{H}} \left(\sum_{n=1}^{N} u_n^{(t)} [\![y_n \neq h(\mathbf{x}_n)]\!] \right)$$

$$g_{t+1} \leftarrow \operatorname*{argmin}_{h \in \mathcal{H}} \left(\sum_{n=1}^{N} u_n^{(t+1)} [\![y_n \neq h(\mathbf{x}_n)]\!] \right)$$

if g_t '**not good**' for $\mathbf{u}^{(t+1)}$ $\Longrightarrow$ g_t-like hypotheses not returned as g_{t+1}
$\Longrightarrow$ g_{t+1} diverse from g_t

idea: **construct $\mathbf{u}^{(t+1)}$** to make g_t **random-like**

$$\frac{\sum_{n=1}^{N} u_n^{(t+1)} [\![y_n \neq g_t(\mathbf{x}_n)]\!]}{\sum_{n=1}^{N} u_n^{(t+1)}} = \frac{1}{2}$$

'Optimal' Re-weighting

want: $\dfrac{\sum_{n=1}^{N} u_n^{(t+1)} [\![y_n \neq g_t(\mathbf{x}_n)]\!]}{\sum_{n=1}^{N} u_n^{(t+1)}} = \dfrac{\blacksquare_{t+1}}{\blacksquare_{t+1} + \bullet_{t+1}} = \dfrac{1}{2}$, where

$$\blacksquare_{t+1} = \sum_{n=1}^{N} u_n^{(t+1)} [\![y_n \neq g_t(\mathbf{x}_n)]\!], \bullet_{t+1} = \sum_{n=1}^{N} u_n^{(t+1)} [\![y_n = g_t(\mathbf{x}_n)]\!]$$

- need: $\underbrace{\text{(total } u_n^{(t+1)} \text{ of incorrect)}}_{\blacksquare_{t+1}} = \underbrace{\text{(total } u_n^{(t+1)} \text{ of correct)}}_{\bullet_{t+1}}$
- one possibility by **re-scaling (multiplying) weights**, if

(total $u_n^{(t)}$ of incorrect) = 1126 ; (weighted incorrect rate) = $\frac{1126}{7337}$	(total $u_n^{(t)}$ of correct) = 6211 ; (weighted correct rate) = $\frac{6211}{7337}$
incorrect: $u_n^{(t+1)} \leftarrow u_n^{(t)} \cdot 6211$	correct: $u_n^{(t+1)} \leftarrow u_n^{(t)} \cdot 1126$

'optimal' re-weighting under weighted incorrect rate ϵ_t:

multiply incorrect $\propto (1 - \epsilon_t)$; multiply correct $\propto \epsilon_t$

Fun Time

For four examples with $u_n^{(1)} = \frac{1}{4}$ for all examples. If g_1 predicts the first example wrongly but all the other three examples correctly. After the 'optimal' re-weighting, what is $u_1^{(2)}/u_2^{(2)}$?

1. 4
2. 3
3. $1/3$
4. $1/4$

Fun Time

For four examples with $u_n^{(1)} = \frac{1}{4}$ for all examples. If g_1 predicts the first example wrongly but all the other three examples correctly. After the 'optimal' re-weighting, what is $u_1^{(2)}/u_2^{(2)}$?

1. 4
2. 3
3. $1/3$
4. $1/4$

Reference Answer: 2

By 'optimal' re-weighting, u_1 is scaled proportional to $\frac{3}{4}$ and every other u_n is scaled proportional to $\frac{1}{4}$. So example 1 is now three times more important than any other example.

Scaling Factor

'optimal' re-weighting: let $\epsilon_t = \frac{\sum_{n=1}^{N} u_n^{(t)} [\![y_n \neq g_t(\mathbf{x}_n)]\!]}{\sum_{n=1}^{N} u_n^{(t)}}$,

multiply incorrect $\propto (1 - \epsilon_t)$; multiply correct $\propto \epsilon_t$

define scaling factor $\blacklozenge_t = \sqrt{\frac{1-\epsilon_t}{\epsilon_t}}$

$$\begin{array}{cccc} \text{incorrect} & \leftarrow & \text{incorrect} & \cdot & \blacklozenge_t \\ \text{correct} & \leftarrow & \text{correct} & / & \blacklozenge_t \end{array}$$

- **equivalent** to optimal re-weighting
- $\blacklozenge_t \geq 1$ iff $\epsilon_t \leq \frac{1}{2}$
  —physical meaning: **scale up incorrect**; **scale down correct**
  —like what Teacher does

scaling-up incorrect examples
leads to **diverse hypotheses**

A Preliminary Algorithm

$\mathbf{u}^{(1)} = ?$

for $t = 1, 2, \ldots, T$

1. obtain g_t by $\mathcal{A}(\mathcal{D}, \mathbf{u}^{(t)})$, where $\mathcal{A}$ tries to minimize $\mathbf{u}^{(t)}$-weighted 0/1 error
2. update $\mathbf{u}^{(t)}$ to $\mathbf{u}^{(t+1)}$ by $\blacklozenge_t = \sqrt{\frac{1-\epsilon_t}{\epsilon_t}}$, where ϵ_t = weighted error (incorrect) rate of g_t

return $G(\mathbf{x}) = ?$

- want g_1 'best' for E_{in}: $u_n^{(1)} = \frac{1}{N}$
- $G(\mathbf{x})$:
  - uniform? but g_2 very bad for E_{in} (**why? :-)**)
  - linear, non-linear? **as you wish**

next: a special algorithm to aggregate **linearly on the fly** with theoretical guarantee

Linear Aggregation on the Fly

$\mathbf{u}^{(1)} = [\frac{1}{N}, \frac{1}{N}, \cdots, \frac{1}{N}]$
for $t = 1, 2, \ldots, T$

1. obtain g_t by $\mathcal{A}(\mathcal{D}, \mathbf{u}^{(t)})$, where ...
2. update $\mathbf{u}^{(t)}$ to $\mathbf{u}^{(t+1)}$ by $\blacklozenge_t = \sqrt{\frac{1-\epsilon_t}{\epsilon_t}}$, where ...
3. compute $\alpha_t = \ln(\blacklozenge_t)$

return $G(\mathbf{x}) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t g_t(\mathbf{x})\right)$

- wish: large α_t for 'good' $g_t \Longleftarrow \alpha_t = \text{monotonic}(\blacklozenge_t)$
- will take $\alpha_t = \ln(\blacklozenge_t)$
  - $\epsilon_t = \frac{1}{2} \Longrightarrow \blacklozenge_t = 1 \Longrightarrow \alpha_t = 0$ (bad g_t zero weight)
  - $\epsilon_t = 0 \Longrightarrow \blacklozenge_t = \infty \Longrightarrow \alpha_t = \infty$ (super g_t superior weight)

Adaptive Boosting = weak base learning algorithm $\mathcal{A}$ (Student)
+ optimal re-weighting factor $\blacklozenge_t$ (Teacher)
+ 'magic' linear aggregation α_t (Class)

Adaptive Boosting (AdaBoost) Algorithm

$\mathbf{u}^{(1)} = [\frac{1}{N}, \frac{1}{N}, \cdots, \frac{1}{N}]$
for $t = 1, 2, \ldots, T$

1. obtain g_t by $\mathcal{A}(\mathcal{D}, \mathbf{u}^{(t)})$, where $\mathcal{A}$ tries to minimize $\mathbf{u}^{(t)}$-weighted 0/1 error
2. update $\mathbf{u}^{(t)}$ to $\mathbf{u}^{(t+1)}$ by

   $[\![y_n \neq g_t(\mathbf{x}_n)]\!]$ (incorrect examples): $u_n^{(t+1)} \leftarrow u_n^{(t)} \cdot \blacklozenge_t$

   $[\![y_n = g_t(\mathbf{x}_n)]\!]$ (correct examples): $u_n^{(t+1)} \leftarrow u_n^{(t)} / \blacklozenge_t$

   where $\blacklozenge_t = \sqrt{\frac{1-\epsilon_t}{\epsilon_t}}$ and $\epsilon_t = \frac{\sum_{n=1}^{N} u_n^{(t)} [\![y_n \neq g_t(\mathbf{x}_n)]\!]}{\sum_{n=1}^{N} u_n^{(t)}}$
3. compute $\alpha_t = \ln(\blacklozenge_t)$

return $G(\mathbf{x}) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t g_t(\mathbf{x})\right)$

AdaBoost: provable **boosting property**

Theoretical Guarantee of AdaBoost

- From VC bound

$$E_{\text{out}}(G) \le E_{\text{in}}(G) + O\left(\sqrt{\underbrace{O(d_{\text{VC}}(\mathcal{H}) \cdot T \log T)}_{d_{\text{VC}} \text{ of all possible } G} \cdot \frac{\log N}{N}}\right)$$

- **first term can be small**:
  $E_{\text{in}}(G) = 0$ after $T = O(\log N)$ iterations if $\epsilon_t \le \epsilon < \frac{1}{2}$ always
- **second term can be small**:
  overall d_{VC} grows "slowly" with T

boosting view of AdaBoost:
if $\mathcal{A}$ is weak but always **slightly better than random** ($\epsilon_t \le \epsilon < \frac{1}{2}$),
then (AdaBoost+$\mathcal{A}$) can be strong ($E_{\text{in}} = 0$ and E_{out} small)

Fun Time

According to $\alpha_t = \ln(\blacklozenge_t)$, and $\blacklozenge_t = \sqrt{\frac{1-\epsilon_t}{\epsilon_t}}$, when would $\alpha_t > 0$?

1. $\epsilon_t < \frac{1}{2}$
2. $\epsilon_t > \frac{1}{2}$
3. $\epsilon_t \neq 1$
4. $\epsilon_t \neq 0$

Fun Time

According to $\alpha_t = \ln(\blacklozenge_t)$, and $\blacklozenge_t = \sqrt{\frac{1-\epsilon_t}{\epsilon_t}}$, when would $\alpha_t > 0$?

1. $\epsilon_t < \frac{1}{2}$
2. $\epsilon_t > \frac{1}{2}$
3. $\epsilon_t \neq 1$
4. $\epsilon_t \neq 0$

Reference Answer: 1

The math part should be easy for you, and it is interesting to think about the physical meaning: $\alpha_t > 0$ (g_t is useful for G) if and only if the weighted error rate of g_t is better than random!

Decision Stump

want: a '**weak**' base learning algorithm $\mathcal{A}$
that minimizes $E_{\text{in}}^{\mathbf{u}}(h) = \frac{1}{N}\sum_{n=1}^{N} u_n \cdot [\![y_n \neq h(\mathbf{x}_n)]\!]$ **a little bit**

a popular choice: decision stump

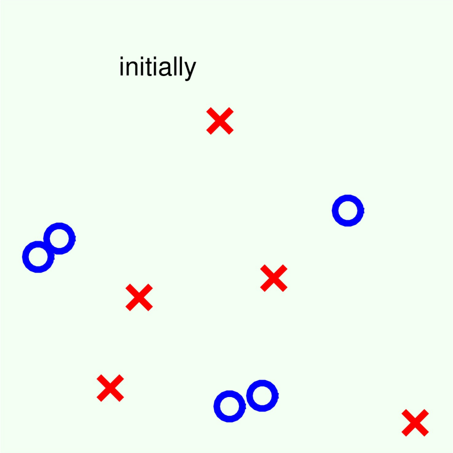
- **in ML Foundations Homework 2, remember? :-)**

$$h_{s,i,\theta}(\mathbf{x}) = s \cdot \operatorname{sign}(x_i - \theta)$$

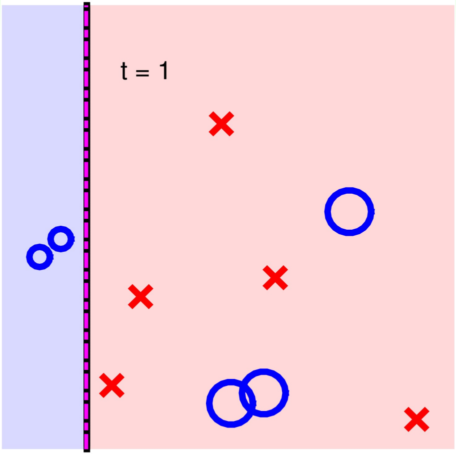
- positive and negative rays on some feature: three parameters (feature i, threshold θ, direction s)
- physical meaning: vertical/horizontal lines in 2D
- efficient to optimize: $O(d \cdot N \log N)$ time

decision stump model:
allows efficient minimization of $E_{\text{in}}^{\mathbf{u}}$
but perhaps **too weak to work by itself**

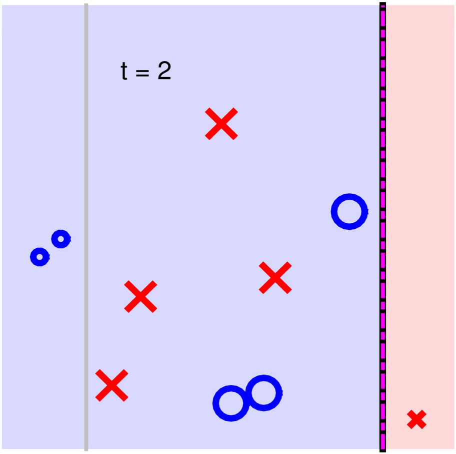
A Simple Data Set



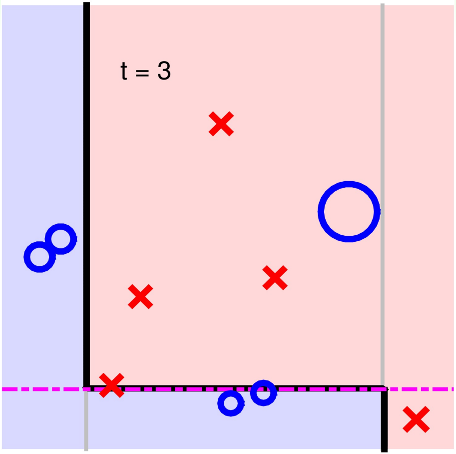
A Simple Data Set



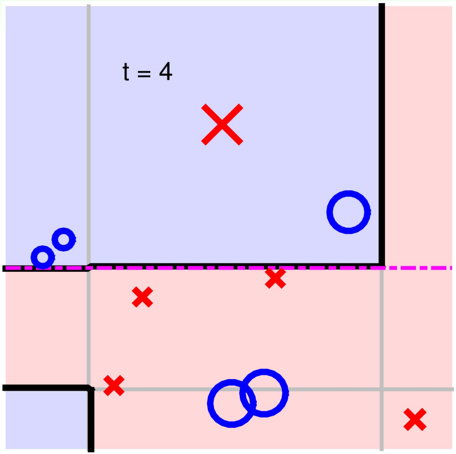
A Simple Data Set



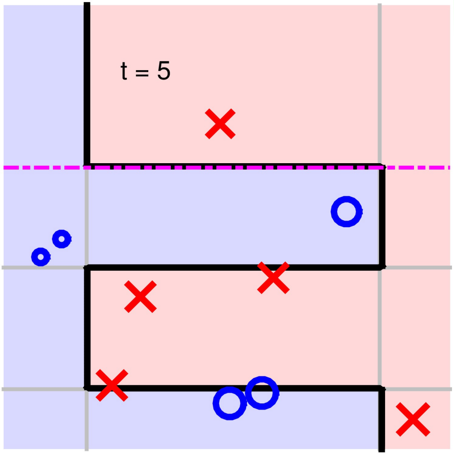
A Simple Data Set



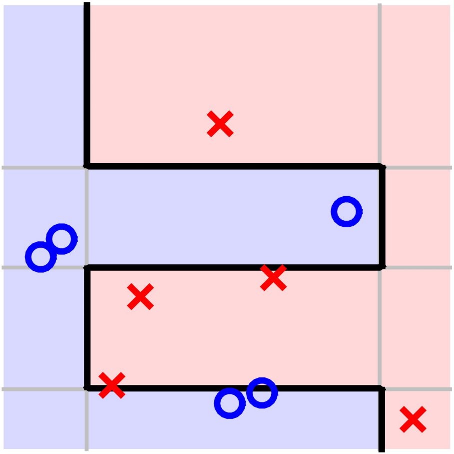
A Simple Data Set



A Simple Data Set

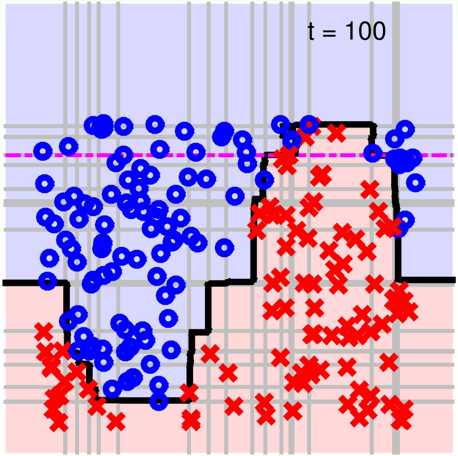


A Simple Data Set



'Teacher'-like algorithm works!

A Complicated Data Set



AdaBoost-Stump: **non-linear yet efficient**

AdaBoost-Stump in Application

original picture by F.U.S.I.A. assistant and derivative work by Sylenius via Wikimedia Commons

The World's First 'Real-Time' Face Detection Program

- AdaBoost-Stump as core model: linear aggregation of key patches selected out of 162,336 possibilities in 24x24 images —**feature selection** achieved through AdaBoost-Stump
- modified linear aggregation *G* to rule out non-face earlier —**efficiency** achieved through modified linear aggregation

AdaBoost-Stump:
efficient feature selection and aggregation

Fun Time

For a data set of size 9876 that contains $\mathbf{x}_n \in \mathbb{R}^{5566}$, after running AdaBoost-Stump for 1126 iterations, what is the number of distinct features within $\mathbf{x}$ that are effectively used by G?

1. $0 \leq$ number ≤ 1126
2. $1126 <$ number ≤ 5566
3. $5566 <$ number ≤ 9876
4. $9876 <$ number

Fun Time

For a data set of size 9876 that contains $\mathbf{x}_n \in \mathbb{R}^{5566}$, after running AdaBoost-Stump for 1126 iterations, what is the number of distinct features within $\mathbf{x}$ that are effectively used by G?

1. $0 \le \text{number} \le 1126$
2. $1126 < \text{number} \le 5566$
3. $5566 < \text{number} \le 9876$
4. $9876 < \text{number}$

Reference Answer: 1

Each decision stump takes only one feature. So 1126 decision stumps need at most 1126 distinct features.

Summary

1. Embedding Numerous Features: Kernel Models
2. Combining Predictive Features: Aggregation Models

Lecture 8: Adaptive Boosting

- Motivation of Boosting
  aggregate weak hypotheses for strength
- Diversity by Re-weighting
  scale up incorrect, scale down correct
- Adaptive Boosting Algorithm
  two heads are better than one, theoretically
- Adaptive Boosting in Action
  AdaBoost-Stump useful and efficient

- **next: learning conditional aggregation instead of linear one**

3. Distilling Implicit Features: Extraction Models