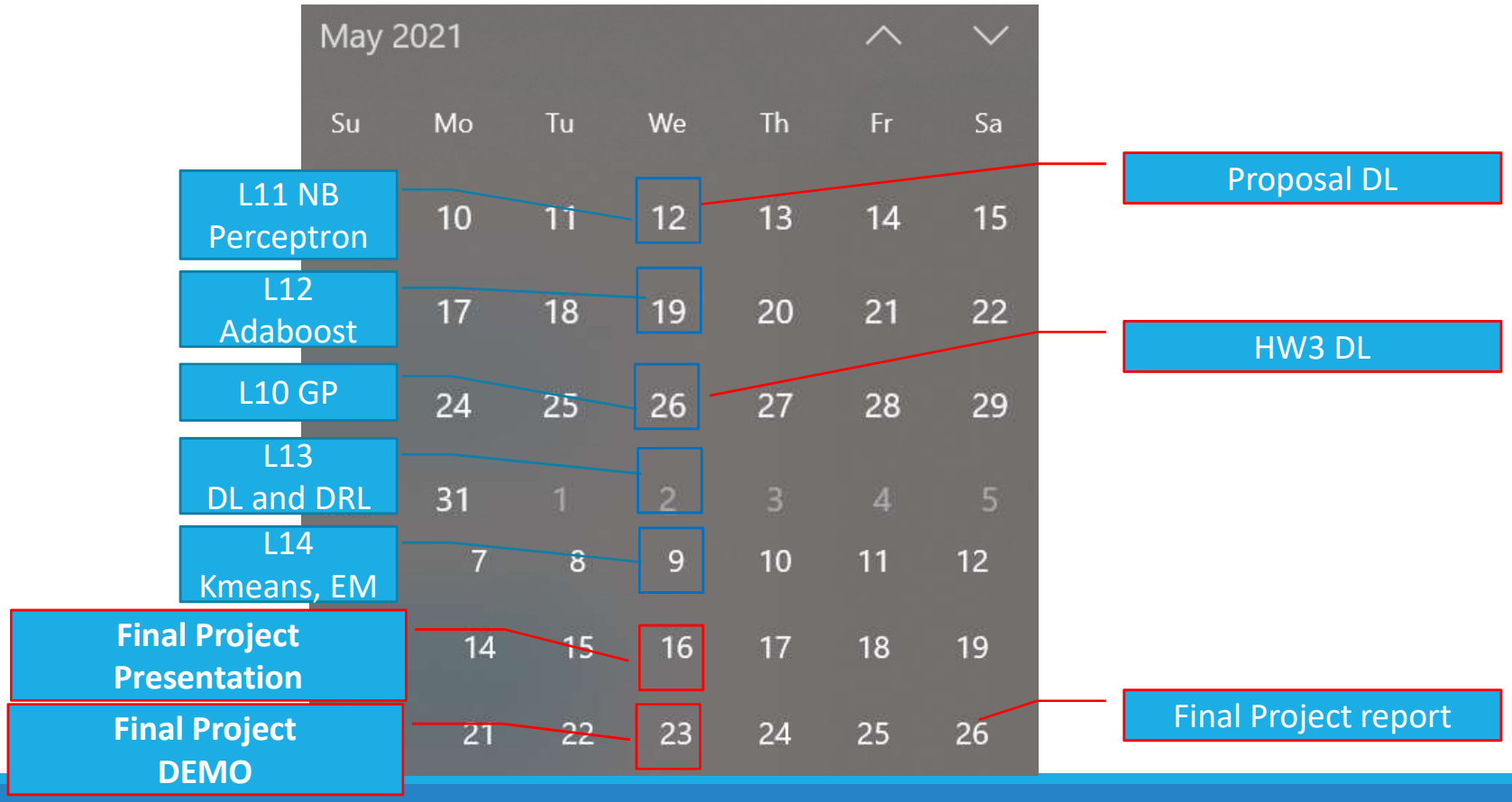
Adaboost

KUO-SHIH TSENG
DEPARTMENT OF MATHEMATICS
NATIONAL CENTRAL UNIVERSITY, TAIWAN

2021/05/19

Course Announcement



Course Announcement

- HW3 was released (Deadline: **5/26 0am**):
- A RL problem (40%)
- A supervised learning problem (40%)
- A RL proof (20%)
- Work on it ASAP!

Course Announcement

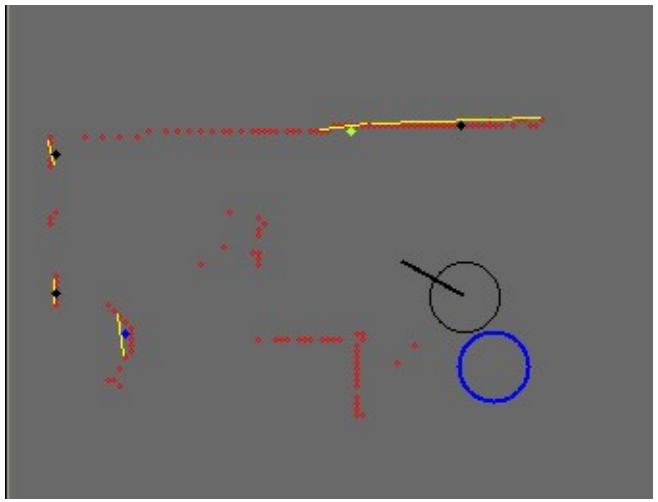
- Project proposal:
- Robots:
  - Team 1: 陳羽暉 蔡沐霖 高文顯　Minibot 1、2
  - Team 2: 陳宇揚　Mdog
  - Team 3: 李家妤　?
  - Team 4: 林寶德　RTF drone
  - Team 5: 邱韋翔　Minibot 3
  - Team 6: 張軒旗　Minibot 4

Outline

- Need
- Supervised learning
- Adaboost
- Ababoost for leg detection
- Experiments
- Adbaboost for image classification
- Adbaboost for forecasting baseball

Need

- The robots need to recognize landmarks for localization.
- The robots need to recognize legs for people tracking.
- Where is the corner? Where is the leg?

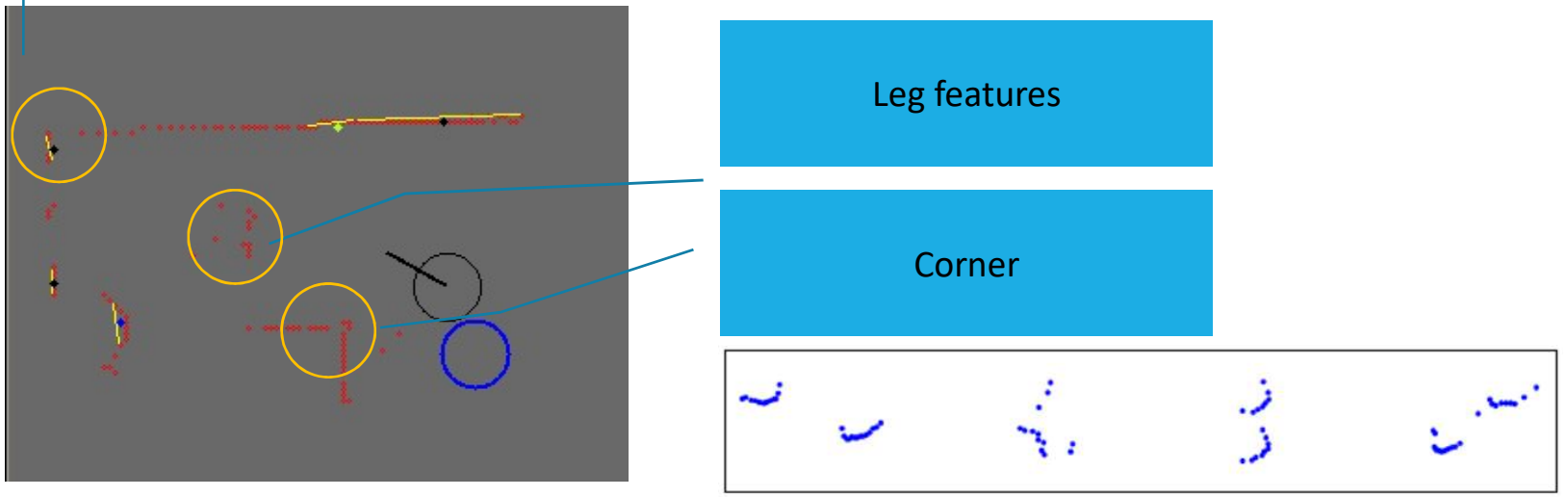


Here is a laser scan data.

Red points represent the laser data
Black circle represent the robot position

Need

- It cannot be solved using a simple thresholds. We should ***train*** the robot to recognize them.

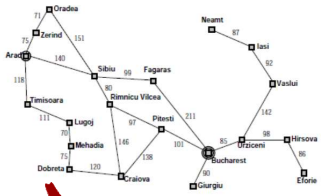


Outline

[Problem solving]

Search problems

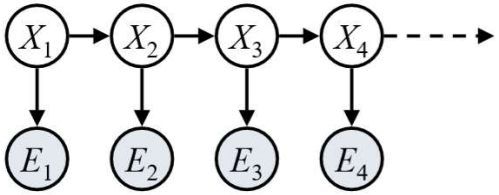
Adversarial Search



[Perception and Uncertainty]

Bayes Theorem

Bayes Filter and Smoothing

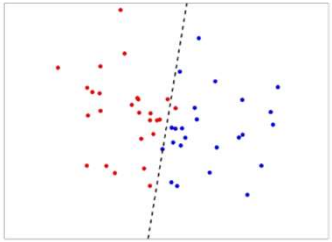


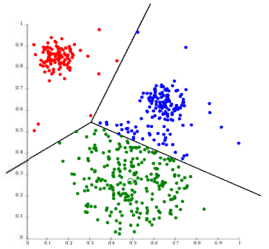
[Learning and Decision-making]

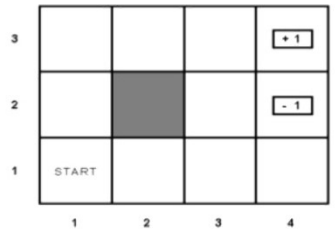
Supervised learning

Unsupervised learning

Reinforcement learning



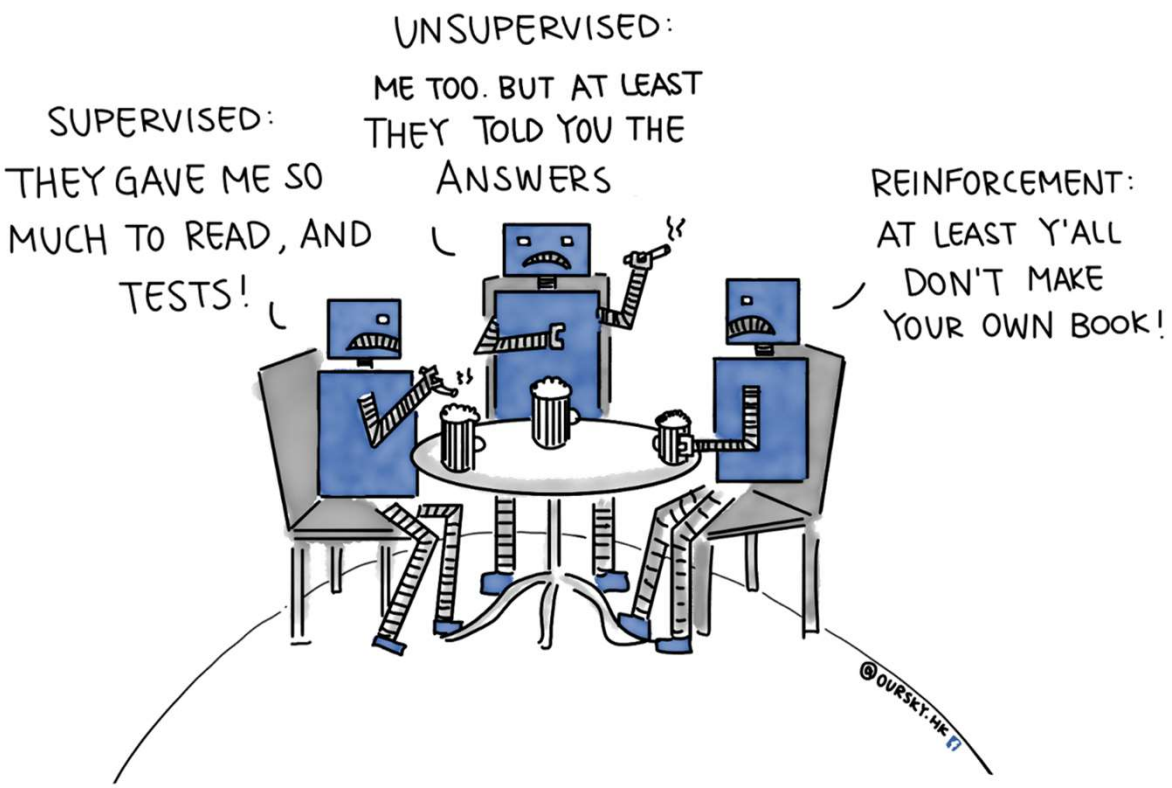




y=XW

y=h(XW)

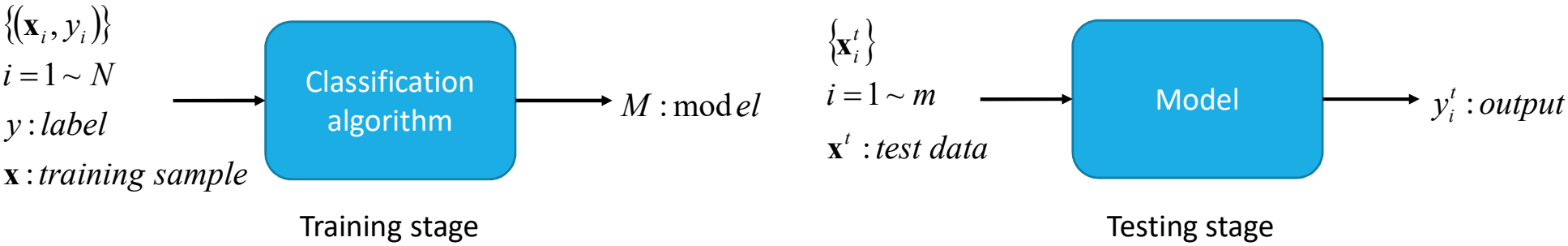
Machine learning



From: Facebook OURSKY

Supervised learning

- Supervised learning:
  - Given data and labels (1 or -1), learn a model for detection.



- How to measure the learned model?

1. Collect data and labels
2. Divided then into two groups, training and testing data.
3. Run training and testing

		True value: T	True value: N
Predicted value	T	True Positive	False Positive
	N	False Negative	True Negative

$$Accuracy = (TP + TN) / N$$

$$\text{Precision} = TP / (TP + FP)$$

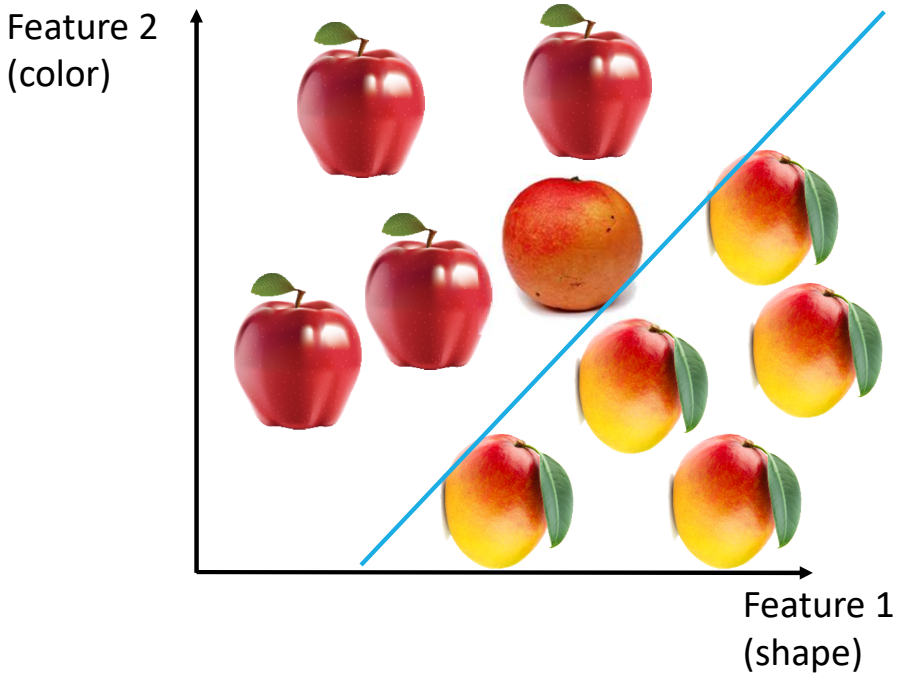
$$\text{Recall} = TP / (TP + FN)$$

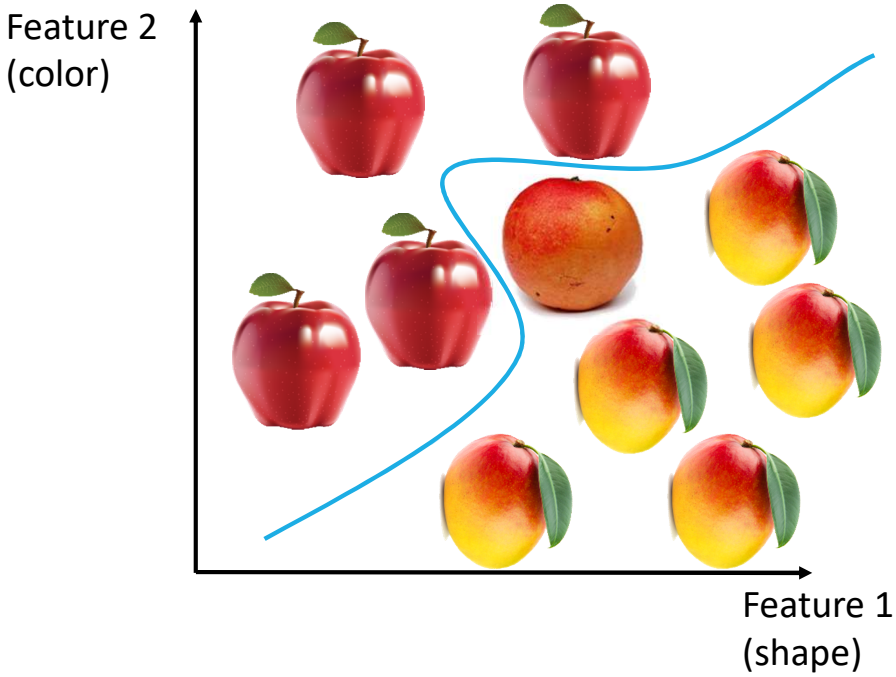
Supervised learning

- Supervised learning:
  - Given data and labels (1 or -1), learn a model for detection.
- There are several approaches:
  - Naïve Bayes
  - Logistic regression
  - Neural network (e.g., perceptron and deep neural network)
  - Support vector machines (SVM)
  - **Adaboost**

Supervised learning

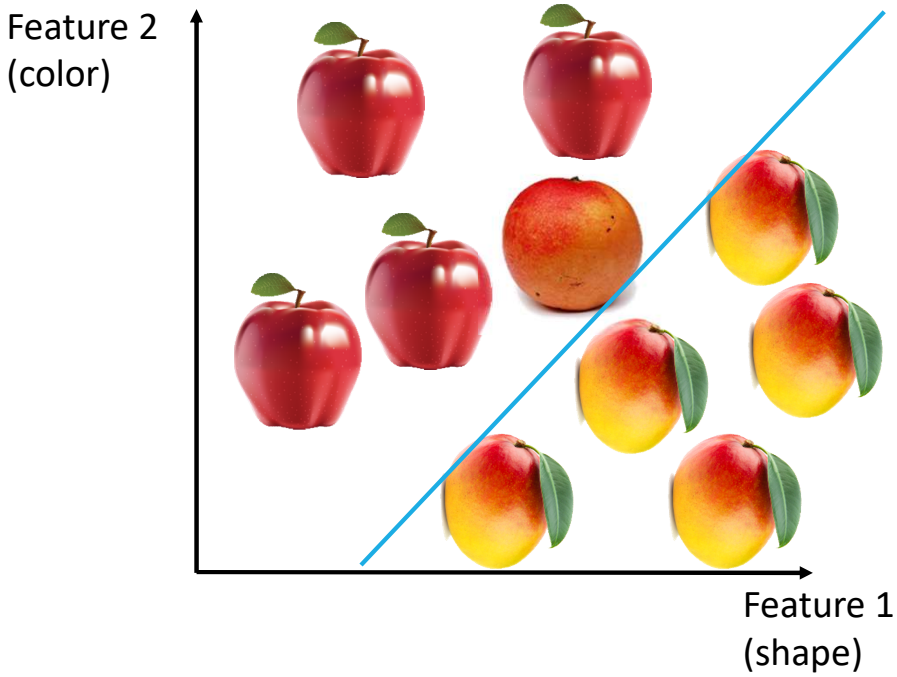
- Linear classifier v.s. non-linear classifier

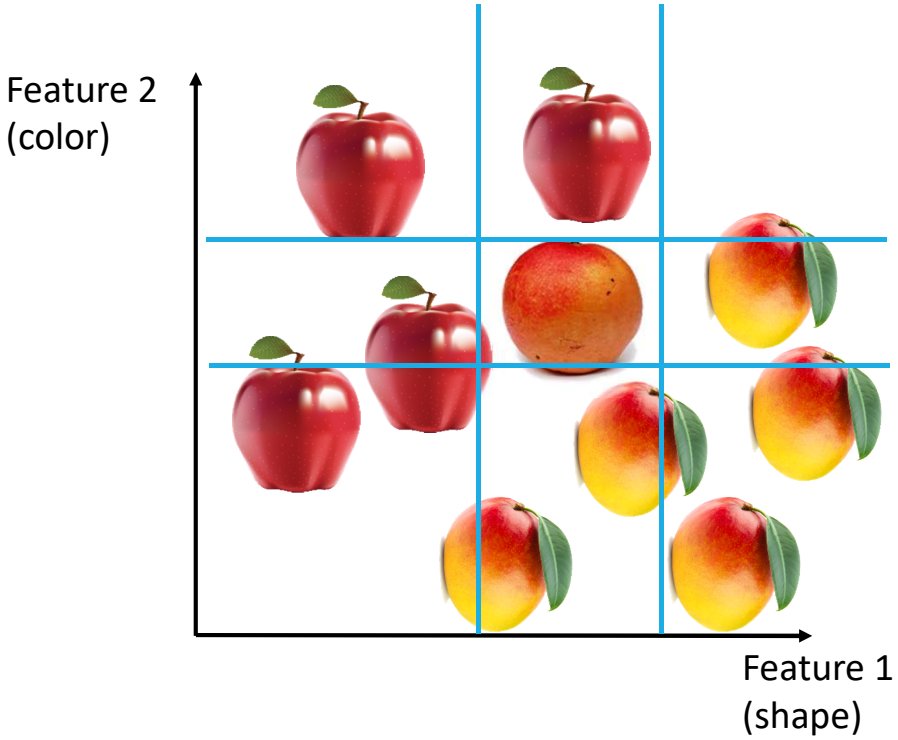




Supervised learning

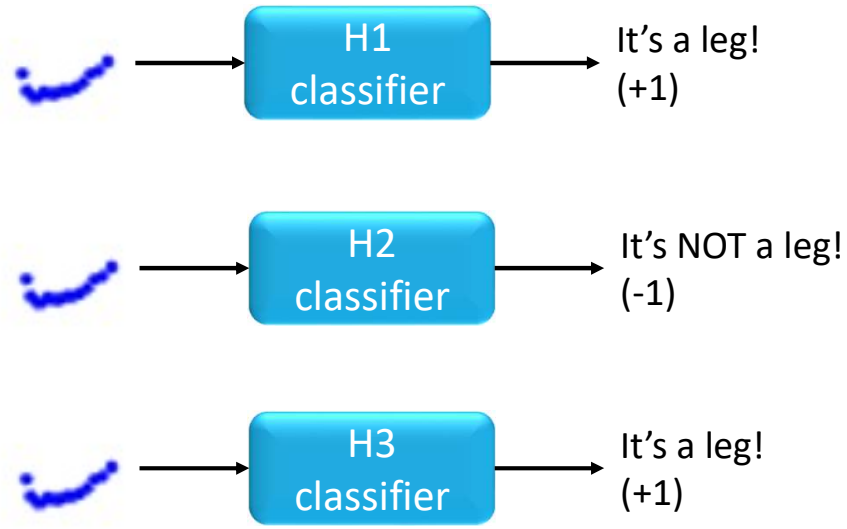
- Linear classifier v.s. ensemble classifier (Adaboost)





Adaboost

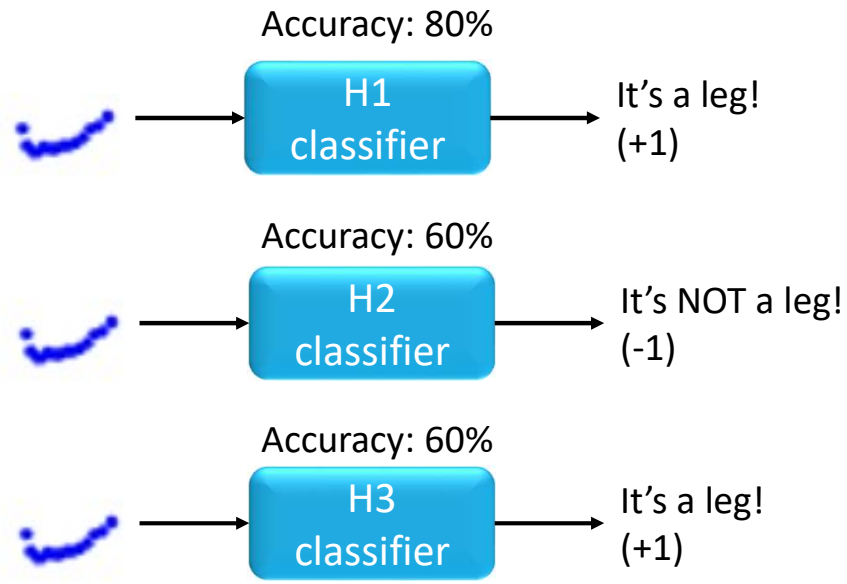
- There are some weak classifiers, which have different predictions.
- Based on these classifiers, it's a leg or not?



Believe majority?

Adaboost

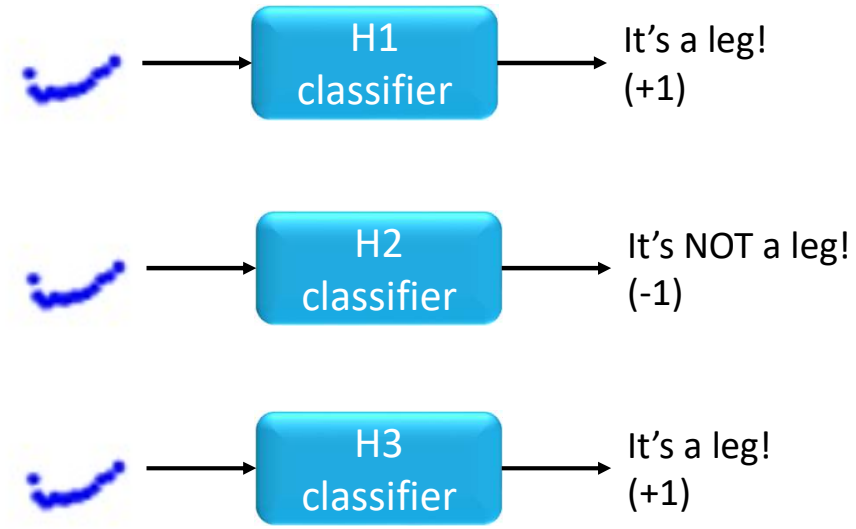
- We have detection rate of each classifier
- Based on these classifiers, it's a leg or not?

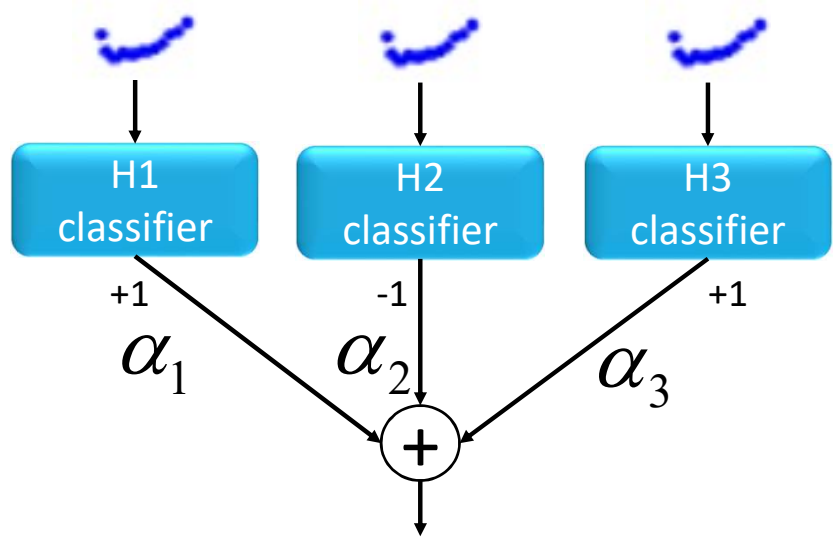


Believe the accurate one?

Adaboost

- The idea of Adaboost is to adjust the weighting of each classifiers. The major vote of the T weak classifiers is the final strong classifier.

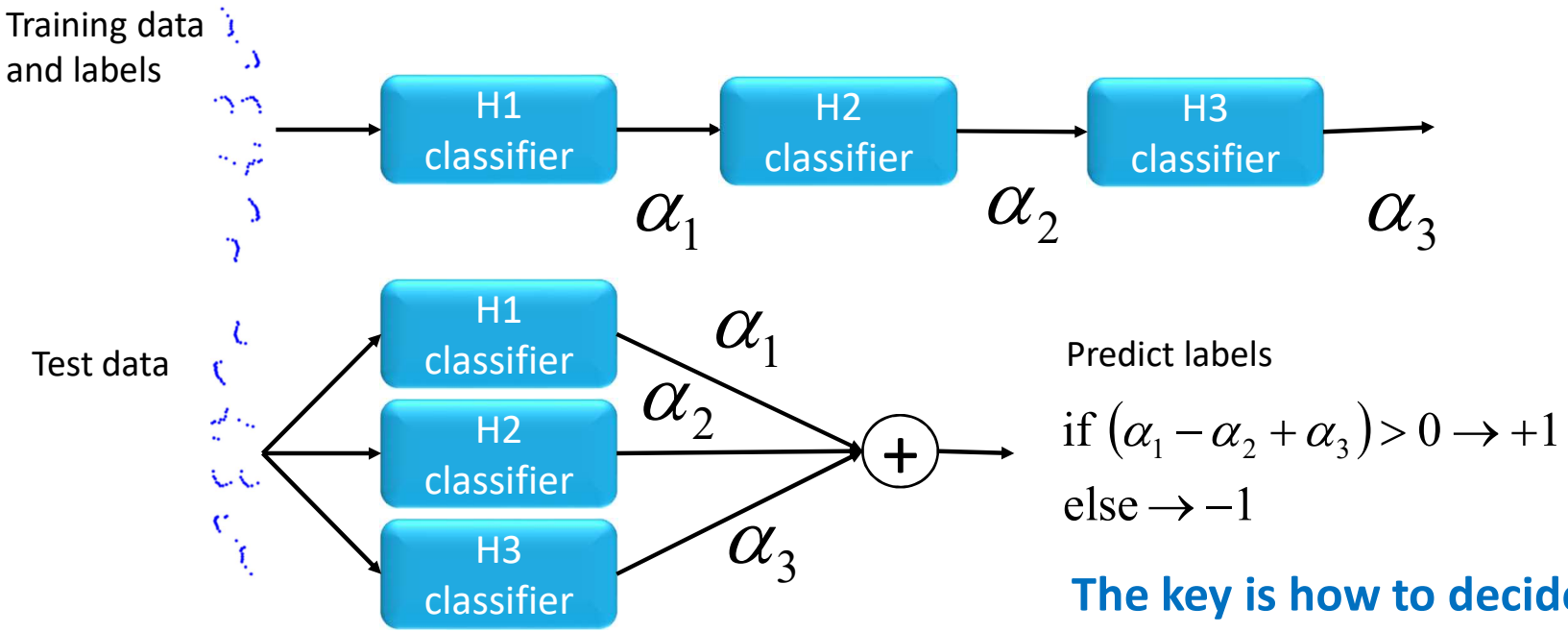




Alpha=???

Adaboost

- The idea of Adaboost is to adjust the weighting of each classifiers. The major vote of the T weak classifiers is the final strong classifier.



The key is how to decide Alpha?

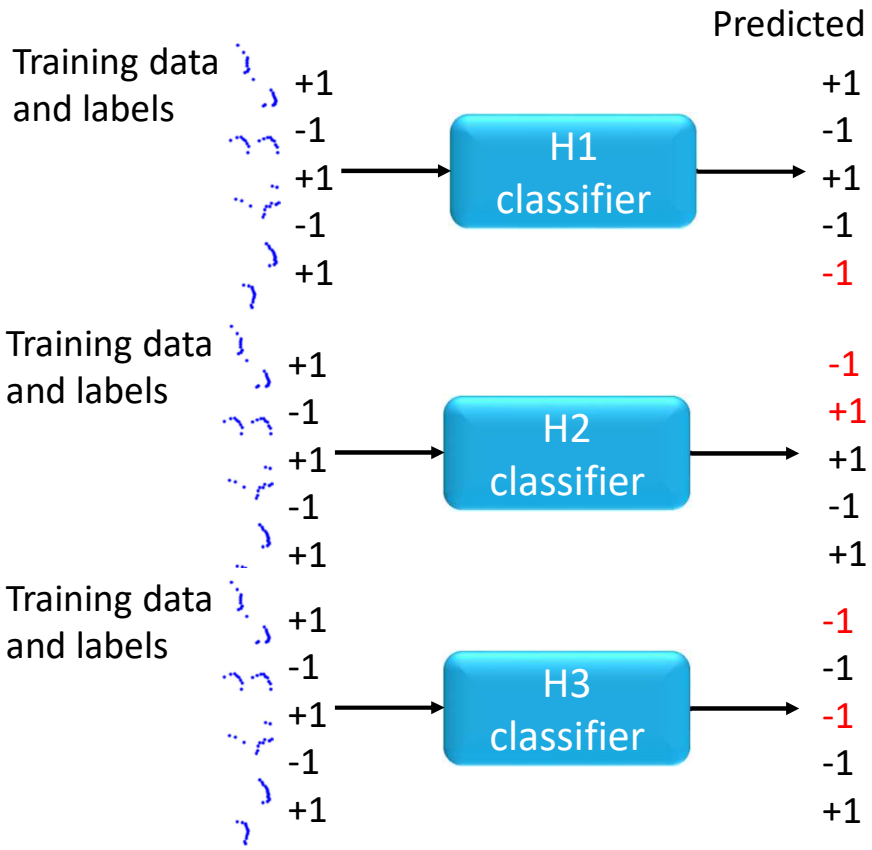
Adaboost

- Input: Set of examples $(e_1, l_1), \ldots, (e_N, l_N)$, where $l_n = +1$ for positive examples and $l_n = -1$ for negative examples.
- Initialize weights $D_1(n) = \frac{1}{2a}$ for $l_n = +1$ and $D_1(n) = \frac{1}{2b}$ for $l_n = -1$, where a and b are the number of positive and negative examples respectively.
- For $t = 1, \ldots, T$: **(T: # of weak classifiers)**
  1) Normalize the weights: $D_t(n) = \frac{D_t(n)}{\sum_{i=1}^{N} D_t(i)}$.
  2) For each feature f_j train a weak classifier h_j using D_t.
  3) For each h_j calculate: $r_j = \sum_{n=1}^{N} D_t(n) l_n h_j(e_n)$, where $h_j(e_n) \in \{+1, -1\}$.
  4) Choose h_j that maximizes $|r_j|$ and set $(h_t, r_t) = (h_j, r_j)$.
  5) Update the weights: $D_{t+1}(n) = D_t(n) \exp(-\alpha_t l_n h_t(e_n))$, where $\alpha_t = \frac{1}{2} \log(\frac{1+r_t}{1-r_t})$.
- The final strong classifier is given by: $H(e) = \text{sign}(F(e))$, where $F(e) = \sum_{t=1}^{T} \alpha_t h_t(e)$.

Assumptions:
The accuracy rate of each classifier is more than 50%.

[3] Kai O. Arras, Oscar Martinez Mozos and Wolfram Burgard "Using Boosted Features for the Detection of People in 2D Range Data," IEEE International Conference on Robotics and Automation, 2007.

Adaboost— Example



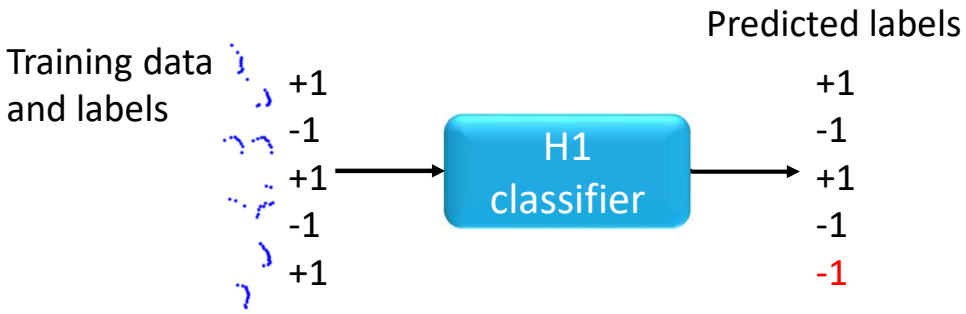
$$a = 3; b = 2;$$

$$r_1 = \frac{1}{2*3} + \frac{1}{2*2} + \frac{1}{2*3} + \frac{1}{2*2} - \frac{1}{2*3} = 0.66$$

$$r_2 = -\frac{1}{2*3} - \frac{1}{2*2} + \frac{1}{2*3} + \frac{1}{2*2} + \frac{1}{2*3} = 0.17$$

$$r_3 = -\frac{1}{2*3} + \frac{1}{2*2} - \frac{1}{2*3} + \frac{1}{2*2} + \frac{1}{2*3} = 0.33$$

Adaboost— Example



$$a = 3; b = 2;$$

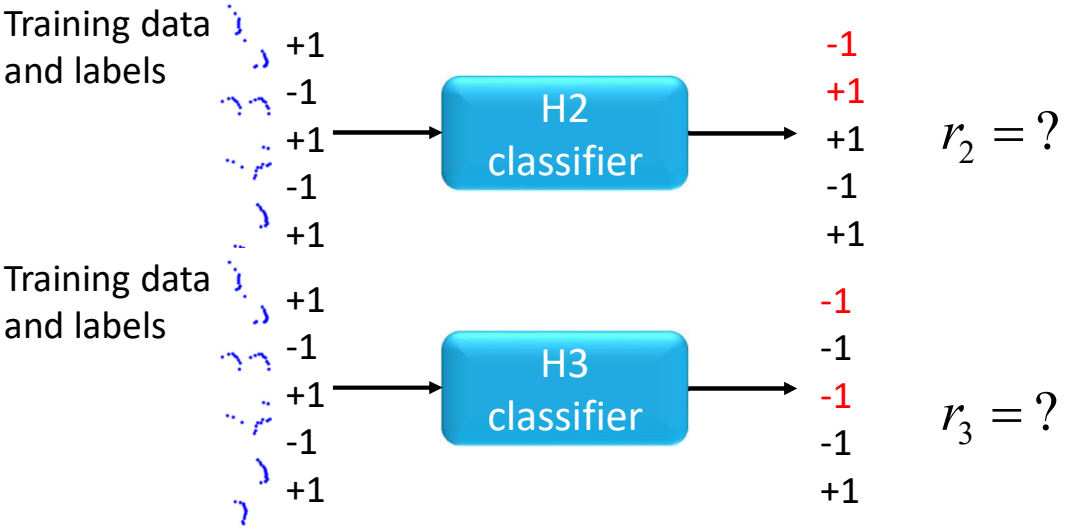
$$r_1 = \frac{1}{2*3} + \frac{1}{2*2} + \frac{1}{2*3} + \frac{1}{2*2} - \frac{1}{2*3} = 0.66$$

$$\alpha_1 = \frac{1}{2}\log\left(\frac{1+0.66}{1-0.66}\right) = 0.06$$

5) Update the weights: $D_{t+1}(n) = D_t(n)\exp(-\alpha_t l_n h_t(e_n))$, where $\alpha_t = \frac{1}{2}\log(\frac{1+r_t}{1-r_t})$.

$$D_1(1) = 0.17, D_2(1) = D_1(1) * 0.94$$
$$D_1(2) = 0.25, D_2(2) = D_1(2) * 0.94$$
$$D_1(3) = 0.17, D_2(3) = D_1(3) * 0.94$$
$$D_1(4) = 0.25, D_2(4) = D_1(4) * 0.94$$
$$D_1(5) = 0.17, D_2(5) = D_1(5) * 1.06$$

Adaboost— Example

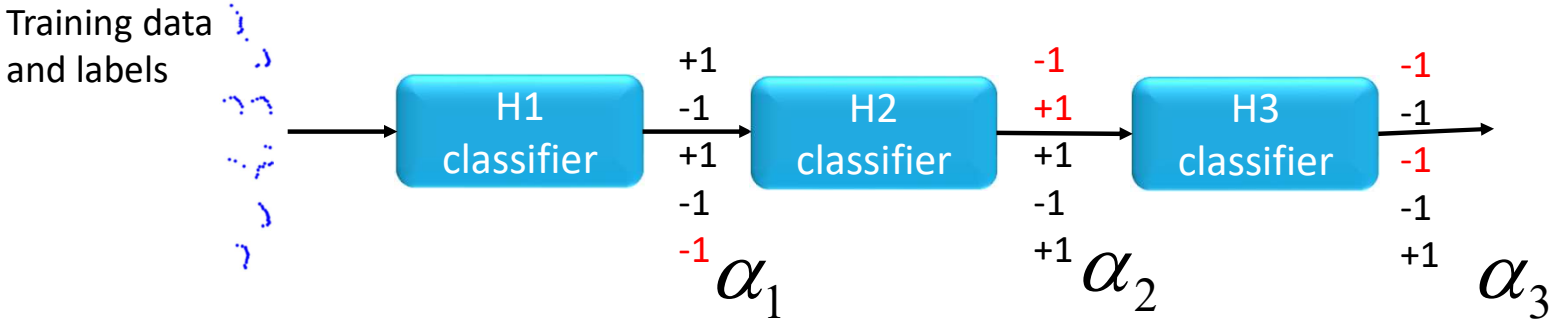


Repeat these steps:

- **Update weighting of each data**
- **Compute alpha values**

Adaboost

- Choose the best classifier and compute alpha.
- Adjust weighting of each data. Unclassified data of the last classifier will have **higher** weighting.
- Repeat these two steps until all alpha values are assigned.
- The proof of Adaboost can be found in Appendix or in [4]

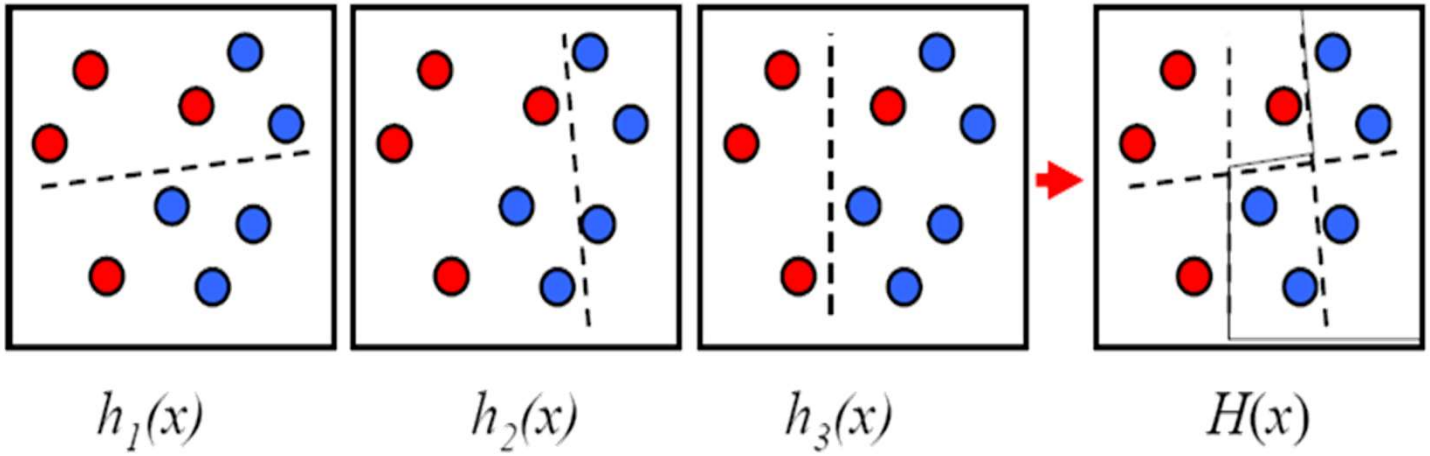


[4] R. E. Schapire and Y. Singer, "Improved boosting algorithms using confidence-rated predictions," Machine Learning, 1999.

Adaboost

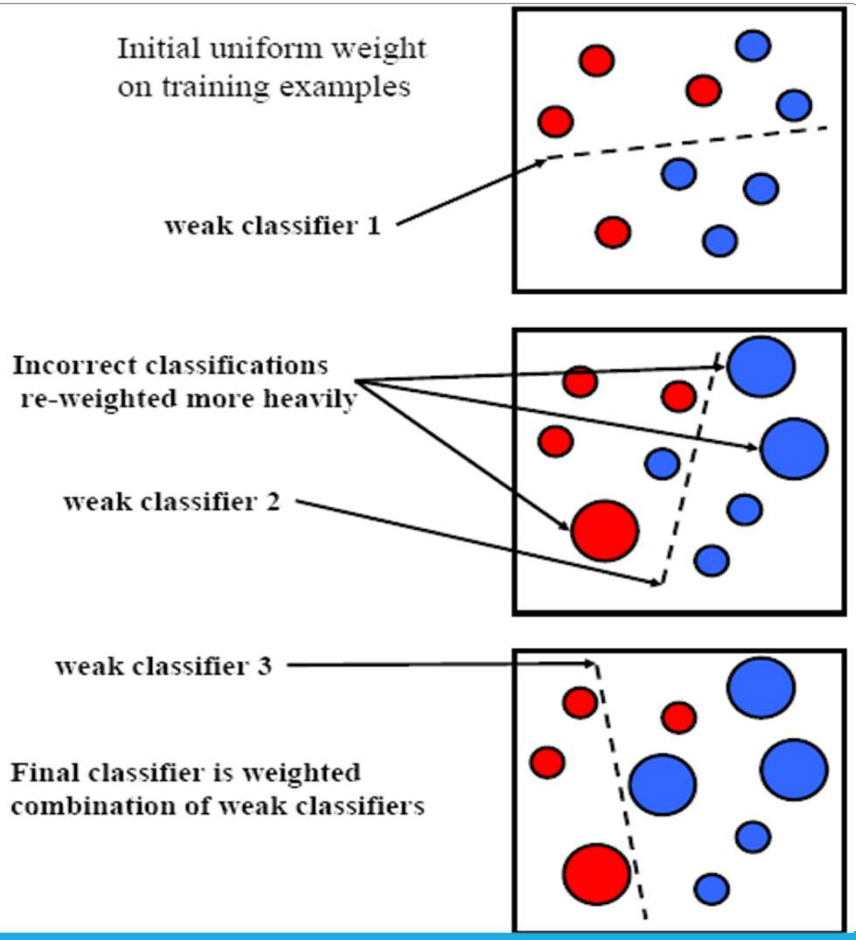
- The major vote of the T weak classifiers is the final strong classifier.

$$H(x) = \mathbf{sign}(\alpha_1 h_1(x) + \alpha_2 h_2(x) + \alpha_3 h_3(x))$$



[5] M. J. Black, Introduction to Computer Vision Course, http://www.cs.brown.edu/people/black/

Adaboost


Initial uniform weight
on training examples
weak classifier 1
Incorrect classifications
re-weighted more heavily
weak classifier 2
weak classifier 3
Final classifier is weighted
combination of weak classifiers

Adaboost

Video

AdaBoost in Action

Kai O. Arras
Social Robotics Lab, University of Freiburg

Nov 2009 Social Robotics Laboratory

Adaboost for Leg Detection

- Adaboost is an ensemble classifier, which consists of weak classifiers.
- In [3], authors proposed 14 features for detecting legs via a 2D laser scanner.
- The first stage is to divide the scans into segments.
- Then, Adaboost decides that a segment is a leg or not.

[3] Kai O. Arras, Oscar Martinez Mozos and Wolfram Burgard "Using Boosted Features for the Detection of People in 2D Range Data," IEEE International Conference on Robotics and Automation, 2007.

Adaboost for Leg Detection

1. Number of points :

$$n = |S_i|$$

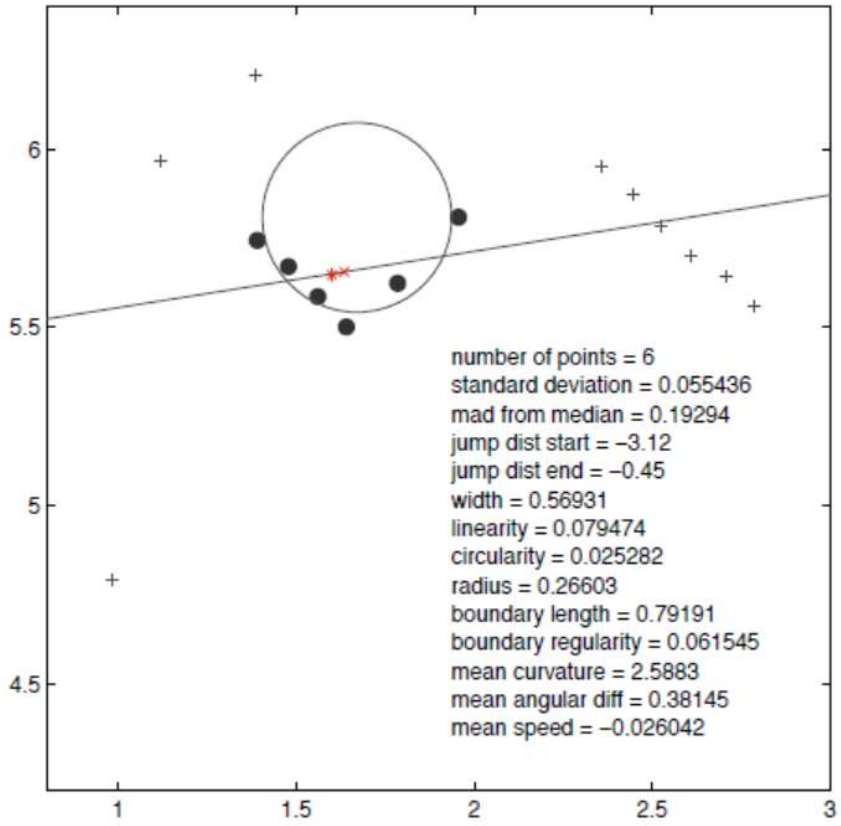
2. Standard deviation :

$$\sigma = \sqrt{\frac{1}{n-1}\sum_j \left\|\mathbf{x}_j - \bar{\mathbf{x}}\right\|^2}$$

3. Mean average deviation from median :

$$\varsigma = \frac{1}{\mathrm{n}}\sum_j \left\|\mathbf{x}_j - \tilde{\mathbf{x}}\right\|$$

$\tilde{\mathbf{x}}$: median



Adaboost for Leg Detection

4. Jump distance from preceeding segment :

5. Jump distance from succeeding segment :

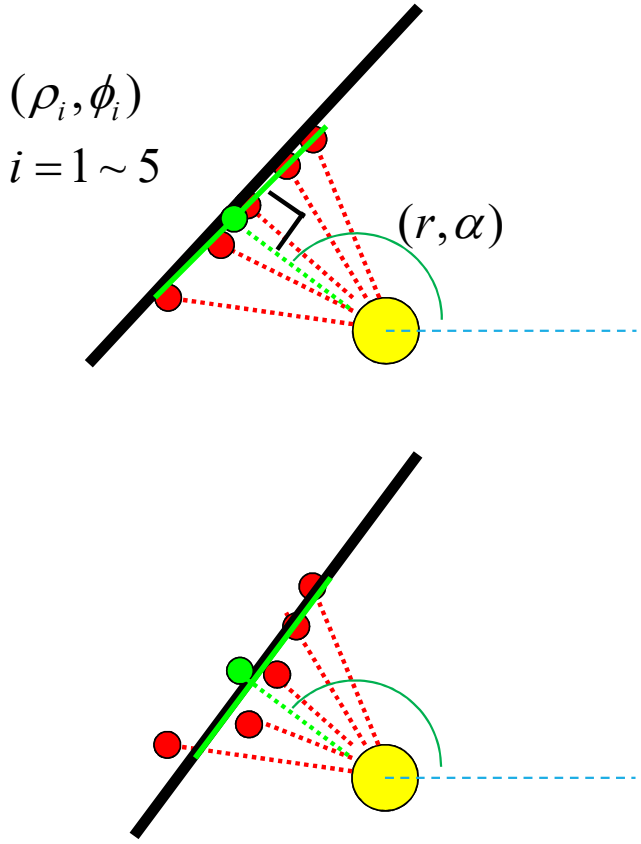
6. Width :

7. Linearity :

A line is represented as (r, α)

$$s_l = \sum_j \left(x_j \cos\alpha + y_j \sin\alpha - r \right)^2$$

where $x_j = \rho_j \cos\phi_j$ and $y_j = \rho_j \sin\phi_j$



Adaboost for Leg Detection

8. Circularity :

A circle is represented as $(X - x_c)^2 + (Y - y_c)^2 = r_c$

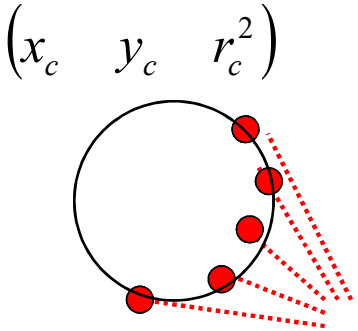
Unkown variables are $x = \begin{pmatrix} x_c & y_c & x_c^2 + y_c^2 - r_c^2 \end{pmatrix}$

x can be solved by pseudo - inverse, $x = (A^T A)^{-1} A^T b$

where

$$A = \begin{pmatrix} -2x_1 & -2y_1 & 1 \\ -2x_2 & -2y_2 & 1 \\ \vdots & \vdots & \vdots \\ -2x_n & -2y_n & 1 \end{pmatrix}, b = \begin{pmatrix} -x_1^2 - y_1^2 \\ -x_2^2 - y_2^2 \\ \vdots \\ -x_n^2 - y_n^2 \end{pmatrix}$$

$$s_c = \sum_{i=1}^{n} \left(r_c - \sqrt{(x_c - x_i)^2 + (y_c - y_i)^2} \right)^2$$

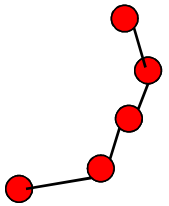


9. Radius : r_c

Adaboost for Leg Detection

10. Boundary length :

$$l = \sum_{j} d_{j,j-1}$$

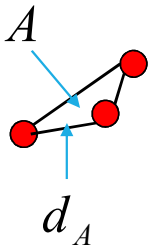


11. Boundary regularity :

standard deviation of $d_{j,j-1}$

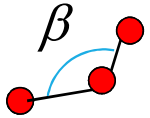
12. Mean curvature :

$$k = \frac{4A}{d_A d_B d_C}, \bar{k} = \sum k_j$$



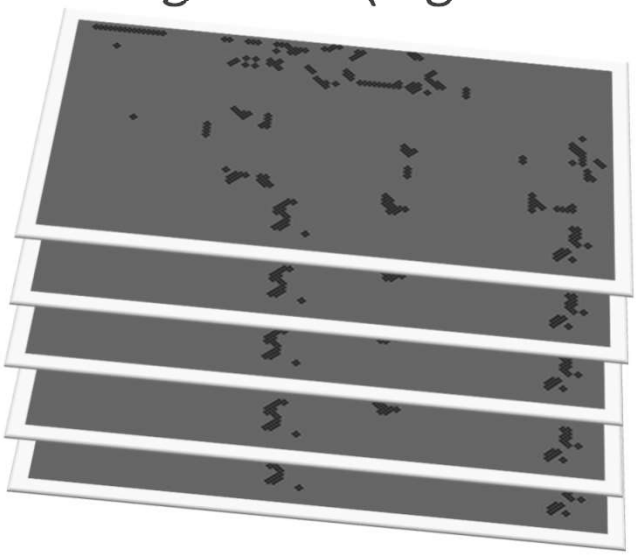
13. Mean angular difference :

$$\beta_j = \angle\left(\overline{x_{j-1}x_j}, \overline{x_j x_{j+1}}\right)$$



Experiments

- After collecting and labeling data, I got 44 segments (leg features) and 390 segments (not leg features).
- The training error is as follows:

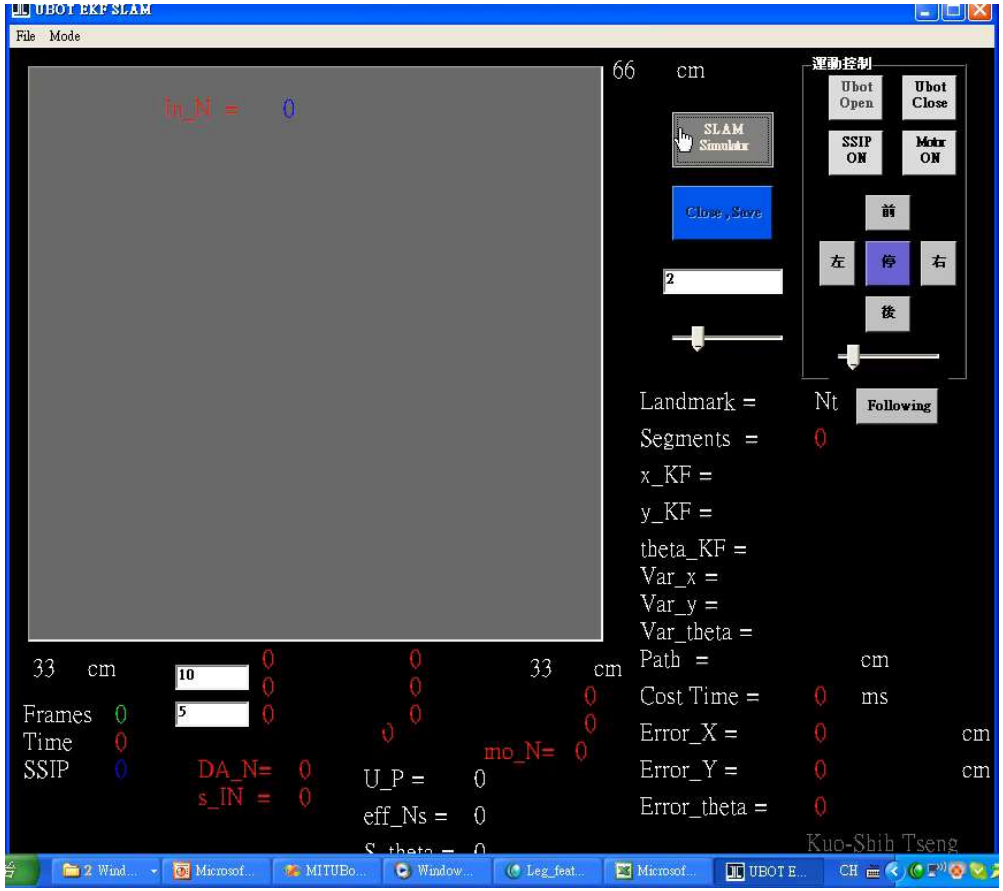


		True value: T	True value: N
Predicted value	T	True Positive=39	False Positive=45
	N	False Negative=5	True Negative=345

$$\Pr ecision = TP/(TP+FP) = 39/(39+45) = 46.4\%$$ ← Precision is low since it is an imbalance data

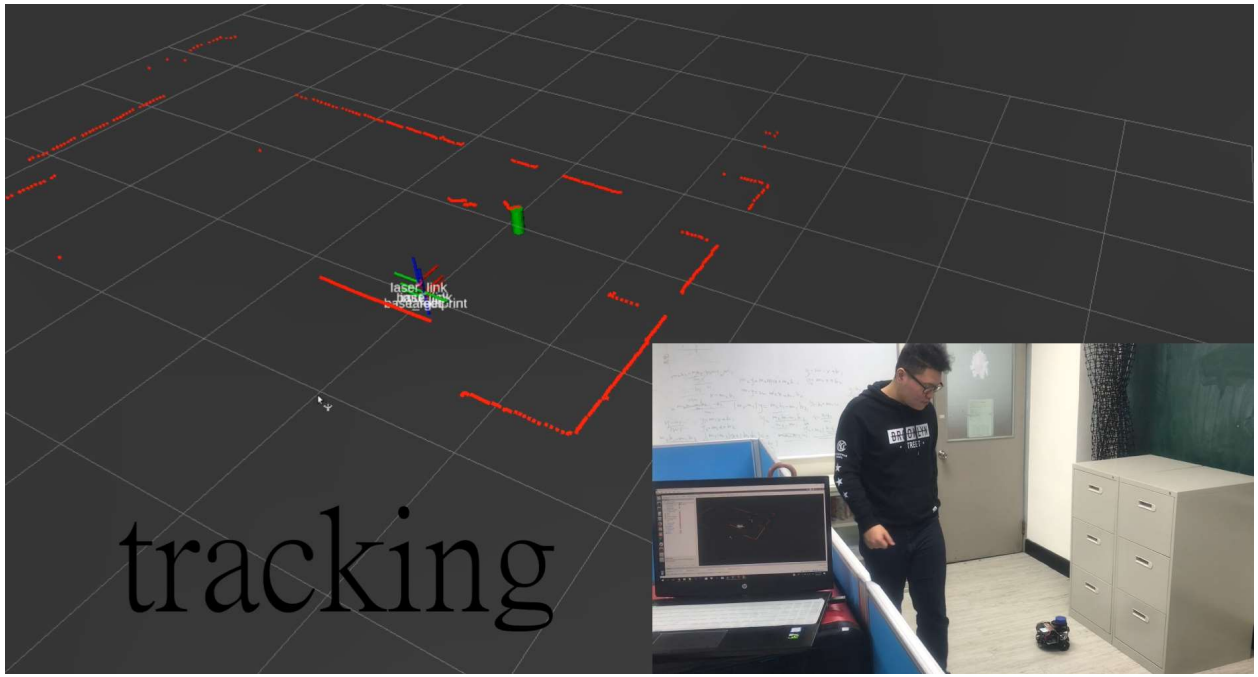
$$\mathrm{Re}\,call = TP/(TP+FN) = 39/(39+5) = 88.6\%$$

Experiments



Experiments

- People detection and tracking



Bing-Xian Lu and Yu-Chung Tsai, “People Tracking with KF on Minibot”, MA5039 final project, 2018.

Adbaboost for image classification

- Given example images

  $(x_1, y_1),\ldots,(x_n, y_n)$

  where $y_i = 0,1$ for negative and positive examples respectively.

- Initialize weights

  $$w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$$

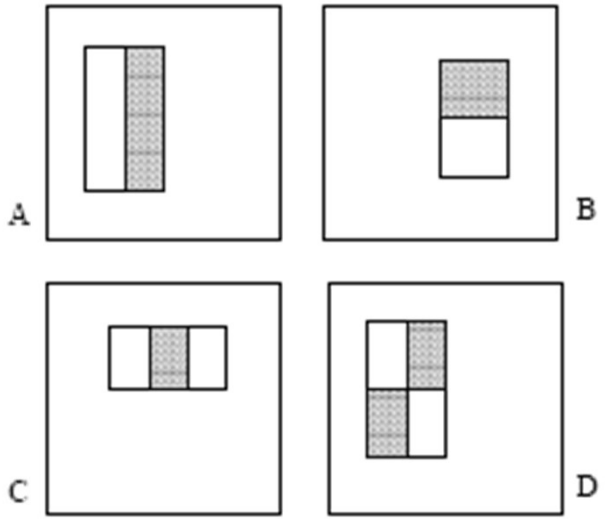
  for $y_i = 0,1$ respectively, where m and l are the number of negatives and positives respectively.

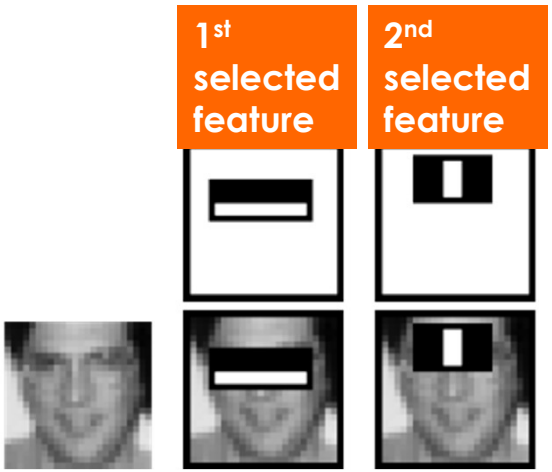
Examples of frontal upright face images for training (yi=1)

Source from: Prof. Tang, NCU CE.

Adbaboost for image classification

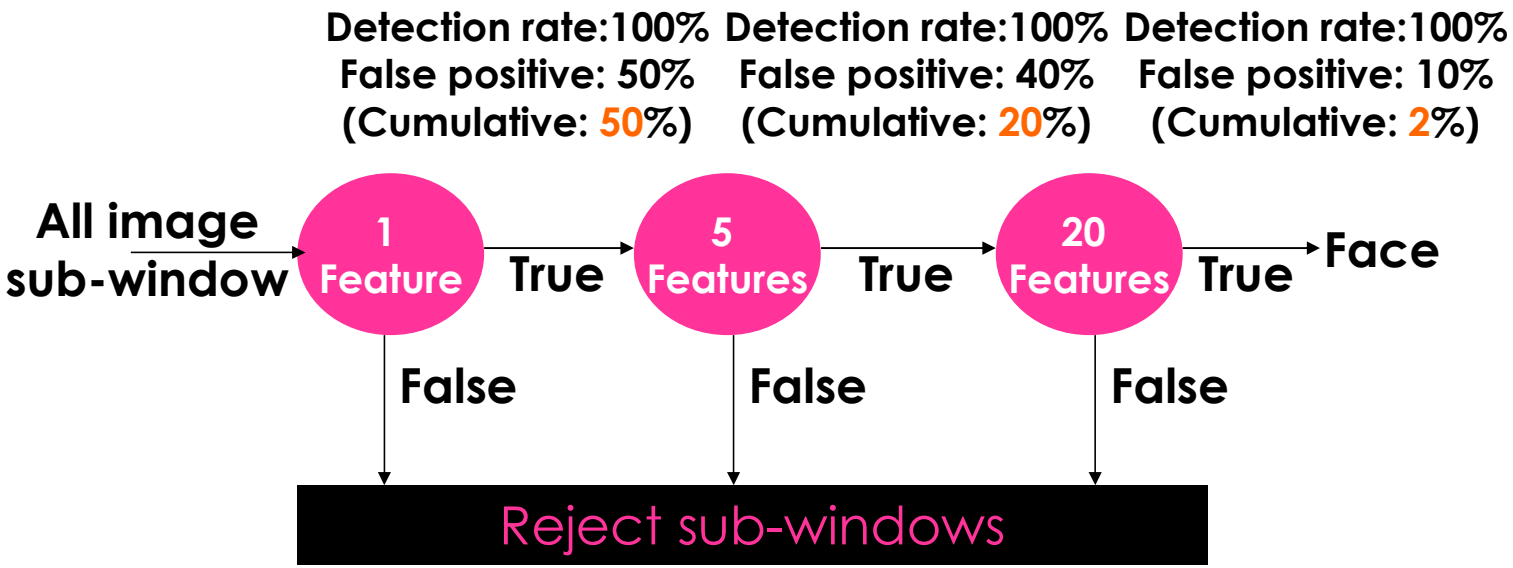
- Haar-like features
- Possible features
  - Two-rectangle feature
  - Three-rectangle feature
  - Four-rectangle feature





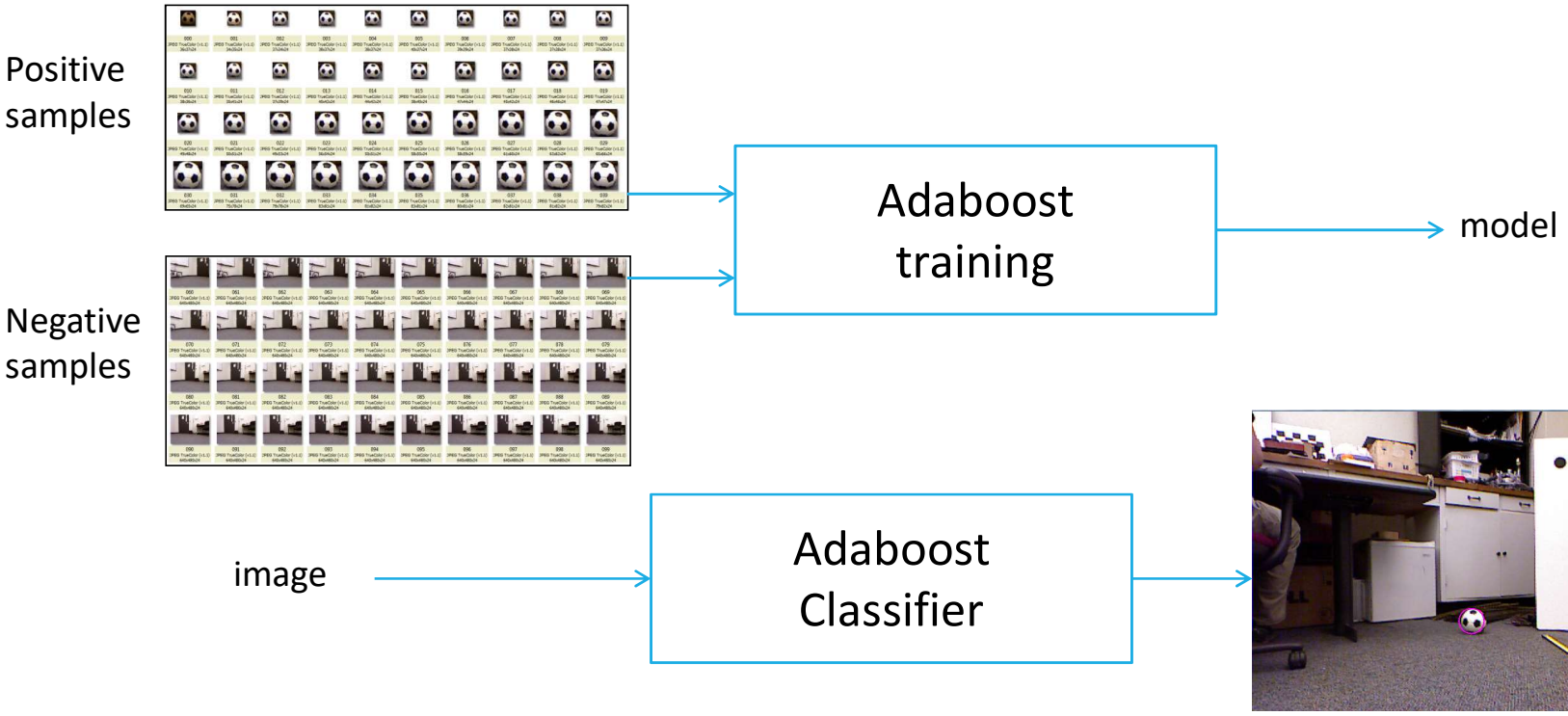
Adbaboost for image classification

- **Motivation**:
  - Increase detection performance (decreased false positive rate)
  - Radically minimize computation

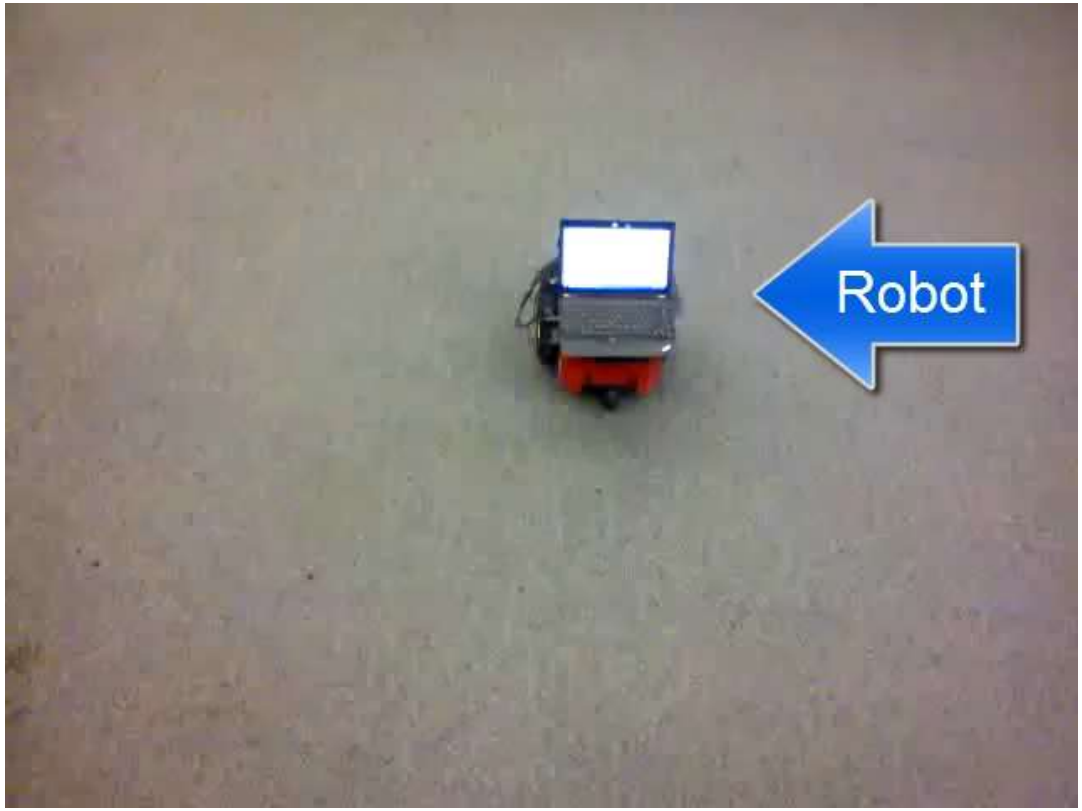


Adbaboost for image classification

- Soccer ball detection



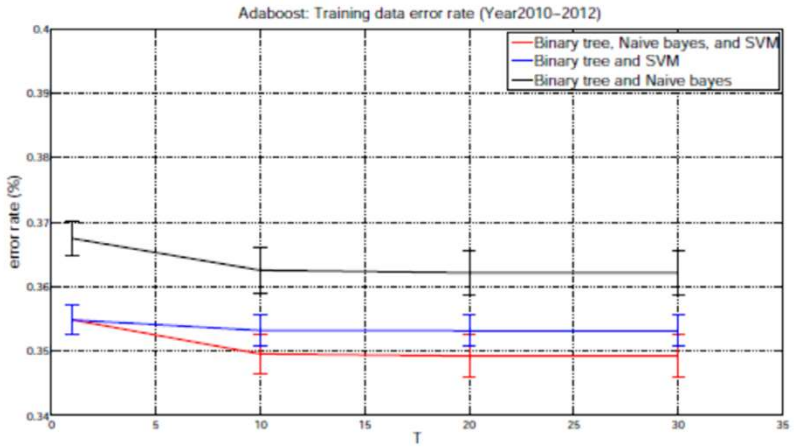
Adbaboost for image classification



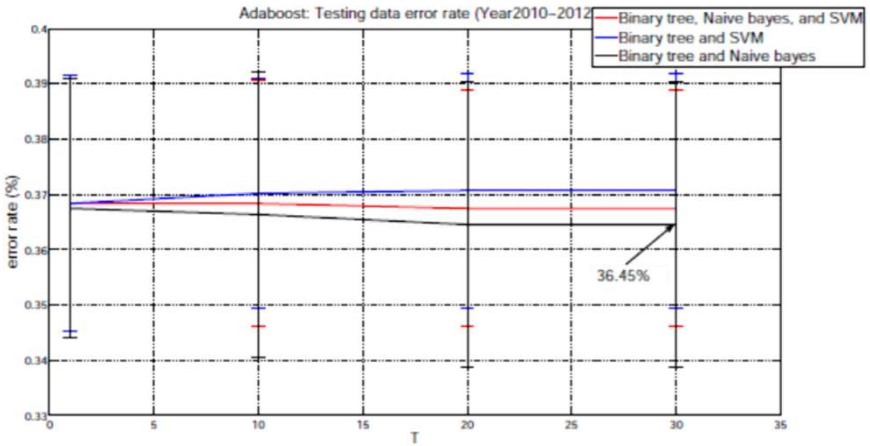
Adbaboost for forecasting baseball

- Forecasting Baseball outcome via Adaboost
  - Decision tree *361
  - Naïve Bayes *20
  - SVM *20

Given: the players' data of two teams
Find: the outcome (win/lose) of the game



(a) Training data rate



(b) Testing data rate

Cheng-Yu Hung, Wei-Cheng Liao, and Kuo-Shih Tseng, "Predict the Outcome of Today's Game Via Machine Learning Approach," Machine learning final project, 2013.

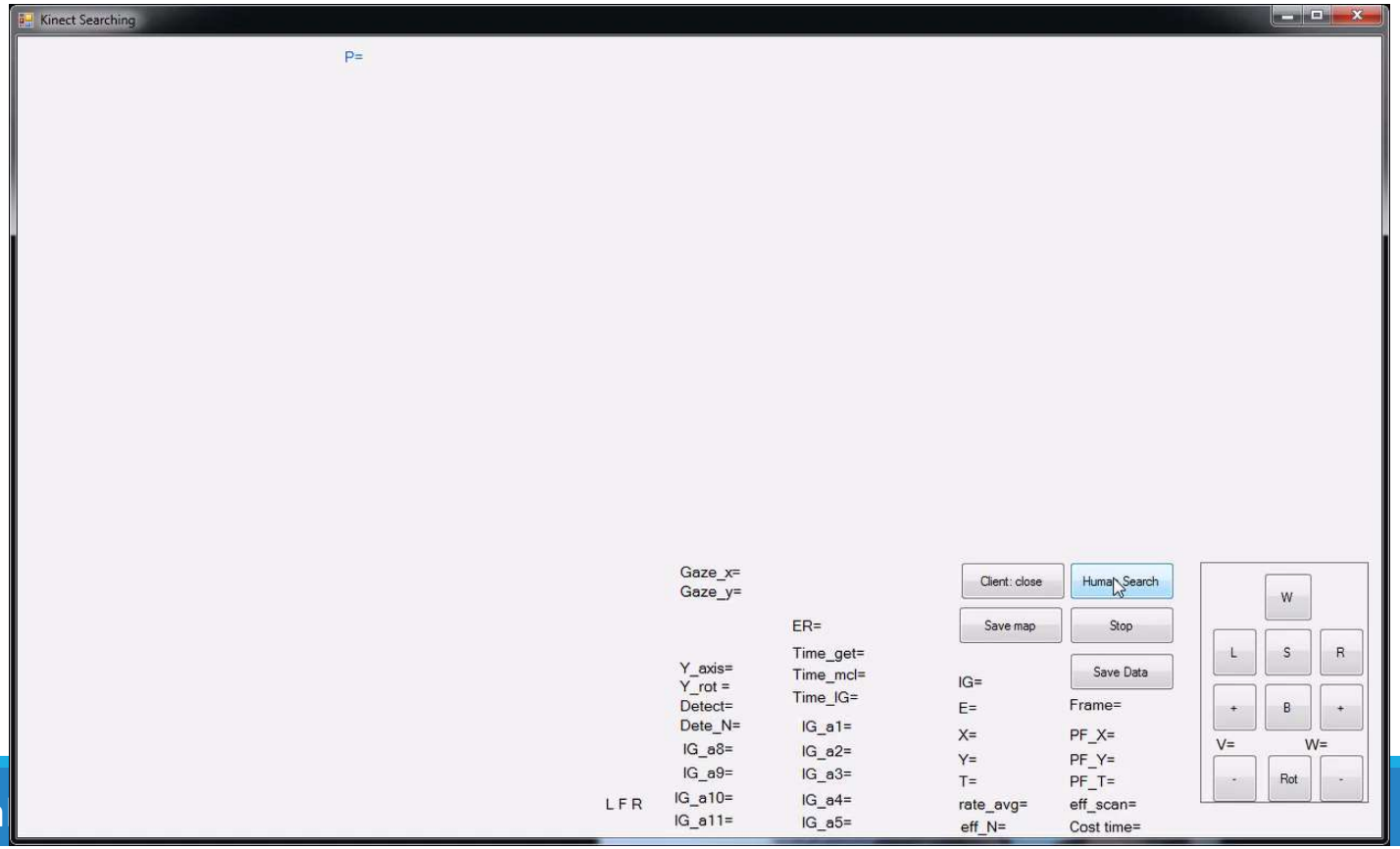
Conclusions

- Supervised learning:
  - Given data and labels (1 or -1), learn a model for detection.
- Unsupervised learning:
  - Given data, divided them into several groups.
- Reinforcement learning:
  - Given the rewards, the robot will learn the optimal actions.
- Adaboost is an ensemble classifier. Before 2013, it is one of the most powerful classifier. If you try to work in Google, they will ask you Adaboost algorithms. (In 2012, deep learning shows the best performance of image classification.)

Q&A

How do humans detect objects?

- Humans have a visual attention model, so human's detection could be better or worse than robot detection.



Adbaboost for image classification

How about Multi-class Adaboost?

Algorithm 1 *AdaBoost (Freund & Schapire 1997)*

1. *Initialize the observation weights* $w_i = 1/n,\ i = 1, 2, \ldots, n.$
2. *For* $m = 1$ *to* M*:*
   (a) *Fit a classifier* $T^{(m)}(\boldsymbol{x})$ *to the training data using weights* w_i.
   (b) *Compute*
   $$err^{(m)} = \sum_{i=1}^{n} w_i \mathbb{I}\left(c_i \neq T^{(m)}(\boldsymbol{x}_i)\right) / \sum_{i=1}^{n} w_i.$$
   (c) *Compute*
   $$\alpha^{(m)} = \log \frac{1 - err^{(m)}}{err^{(m)}}.$$
   (d) *Set*
   $$w_i \leftarrow w_i \cdot \exp\left(\alpha^{(m)} \cdot \mathbb{I}\left(c_i \neq T^{(m)}(\boldsymbol{x}_i)\right)\right),\ i = 1, 2, \ldots, n.$$
   (e) *Re-normalize* w_i.
3. *Output*
   $$C(\boldsymbol{x}) = \arg\max_k \sum_{m=1}^{M} \alpha^{(m)} \cdot \mathbb{I}(T^{(m)}(\boldsymbol{x}) = k).$$

• **K = 2: SAMME reduces to AdaBoost!**

Algorithm 2 *SAMME*

1. *Initialize the observation weights* $w_i = 1/n,\ i = 1, 2, \ldots, n.$

2. *For* $m = 1$ *to* M*:*

   (a) *Fit a classifier* $T^{(m)}(\boldsymbol{x})$ *to the training data using weights* w_i.

   (b) *Compute*

$$err^{(m)} = \sum_{i=1}^{n} w_i \mathbb{I}\left(c_i \neq T^{(m)}(\boldsymbol{x}_i)\right) / \sum_{i=1}^{n} w_i.$$

   (c) *Compute*

$$\alpha^{(m)} = \log \frac{1 - err^{(m)}}{err^{(m)}} + \log(K-1). \qquad (1)$$

   (d) *Set*

$$w_i \leftarrow w_i \cdot \exp\left(\alpha^{(m)} \cdot \mathbb{I}\left(c_i \neq T^{(m)}(\boldsymbol{x}_i)\right)\right),\ i = 1, \ldots, n.$$

   (e) *Re-normalize* w_i.

3. *Output*

$$C(\boldsymbol{x}) = \arg\max_k \sum_{m=1}^{M} \alpha^{(m)} \cdot \mathbb{I}(T^{(m)}(\boldsymbol{x}) = k).$$

Ref.: J. Zhu, H. Zou, and S. Rosset, and T. Hastie, "Multi-class adaboost," accepted by *Statistics and Its Interface*, Special issue on data mining and machine learning, 2009.

Adaboost

Input : $(x_1, y_1),...,(x_N, y_N)$ where $x_i \in X, y_i \in \{-1,+1\}$

1. Initialize : $D_1 = 1/N$

2. For t = 1 ~ T

   2.1 Train a weak classifier $h_t(x)$ on weighted training data

   $$\varepsilon_t = \sum_{i:h_t(x_i)\neq y_i} D_t(i)$$

   2.2 Compute voting weight of $h_t(x)$: $\alpha_t = \frac{1}{2}\log\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right)$

   2.3 Recompute weights :

   $$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t}, \text{if } y_i = h_t(x_i) \\ e^{\alpha_t}, \text{if } y_i \neq h_t(x_i) \end{cases}, \text{where } Z_t \text{ is a normalization factor}$$

3. $H(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$

Adaboost

The training error of Adaboost is at most $\exp\left(-2\sum_{t=1}^{T}\gamma_t^2\right)$

where $\varepsilon_t = \frac{1}{2} - \gamma_t$

EX1:

$\varepsilon_1 = 0.2, \varepsilon_2 = 0.1, \varepsilon_3 = 0.15$

$\text{error} \le \exp\left(-2\sum_{t=1}^{T}\gamma_t^2\right)$

$= \exp\left[-2\left(0.3^2 + 0.4^2 + 0.35^2\right)\right]$

$= 47.4\%$

EX2:

$\varepsilon_1 = 0.2, \varepsilon_2 = 0.1, \varepsilon_3 = 0.15, \varepsilon_4 = 0.1$

$\text{error} \le \exp\left(-2\sum_{t=1}^{T}\gamma_t^2\right)$

$= \exp\left[-2\left(0.3^2 + 0.4^2 + 0.35^2 + 0.4^2\right)\right]$

$= 35.1\%$

Adaboost

The training error (ε) of Adaboost is at most $\exp\left(-2\sum_{t=1}^{T}\gamma_t^2\right)$, where $\varepsilon_t = \frac{1}{2} - \gamma_t$

$$\varepsilon = \frac{1}{N}\sum_i \begin{cases} 1, \text{if } y_i \neq H(x_i) \\ 0, \text{else} \end{cases}$$

$$= \frac{1}{N}\sum_i \begin{cases} 1, y_i f(x_i) \leq 0 \\ 0, \text{else} \end{cases}$$

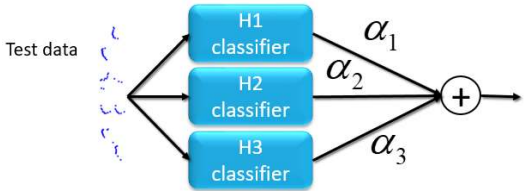
$$y_i f(x_i) = \begin{cases} 1, \text{if } y_i = H(x_i) \\ -1, \text{if } y_i \neq H(x_i) \end{cases}$$

$$\leq \frac{1}{N}\sum_i \exp(-y_i f(x_i))$$

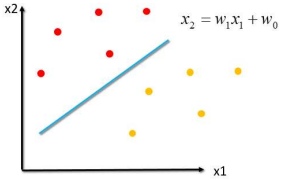
$$\text{if } z \leq 0, e^{-z} \geq 1$$

$$= F(\alpha)$$

Try to find
1. alpha values?
2. error bound?



Tune α



Tune W

Adaboost

Try to find
1. alpha values?
2. error bound?

$$\varepsilon = \frac{1}{N}\sum_i \begin{cases} 1, \text{if } y_i \neq H(x_i) \\ 0, \text{else} \end{cases}$$

$$= \frac{1}{N}\sum_i \begin{cases} 1, y_i f(x_i) \leq 0 \\ 0, \text{else} \end{cases}$$

$$\leq \frac{1}{N}\sum_i \exp(-y_i f(x_i))$$

Adaboost Outcome

$$= \sum_i D_{T+1}(i)\prod_t Z_t$$

$$= \prod_t Z_t$$

Try to minimize this value!

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t}, \text{if } y_i = h_t(x_i) \\ e^{\alpha_t}, \text{if } y_i \neq h_t(x_i) \end{cases}$$

$$= \frac{D_t(i)\exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

$$D_{T+1}(i) = D_1(i)\frac{\exp(-\alpha_t y_i h_1(x_i))}{Z_1}\cdots\frac{\exp(-\alpha_t y_i h_T(x_i))}{Z_T}$$

$$D_{T+1}(i) = \frac{1}{N}\frac{\exp\left(-y_i\sum_t \alpha_t h_t(x_i)\right)}{\prod_t Z_t}$$

$$D_{T+1}(i) = \frac{1}{N}\frac{\exp(-y_i f(x_i))}{\prod_t Z_t}$$

Adaboost

Try to find
1. alpha values?
2. error bound?

Minimize Z_t

$$Z_t = \sum_i D_t(i) \times \begin{cases} e^{-\alpha_t}, \text{if } h_t(x_i) = y_i \\ e^{\alpha_t}, \text{if } h_t(x_i) \neq y_i \end{cases}$$

$$= \sum_{i:h_t(x_i)=y_i} D_t(i) e^{-\alpha_t} + \sum_{i:h_t(x_i)\neq y_i} D_t(i) e^{\alpha_t}$$

$$= e^{-\alpha_t}(1-\varepsilon_t) + e^{\alpha_t}\varepsilon_t$$

$$\frac{dZ_t}{d\alpha} = -e^{-\alpha_t}(1-\varepsilon_t) + e^{\alpha_t}\varepsilon_t = 0$$

$$\Rightarrow \alpha_t = \frac{1}{2}\log\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right) \quad \text{(\#1)}$$

$$Z_t = e^{-\alpha_t}(1-\varepsilon_t) + e^{\alpha_t}\varepsilon_t$$

$$= 2\sqrt{\varepsilon_t(1-\varepsilon_t)}$$

$$= \sqrt{1-4\gamma_t^2} \qquad \because \varepsilon_t = \frac{1}{2} - \gamma_t$$

$$\leq e^{-2\gamma_t^2} \qquad \because 1 + x \leq e^x$$

$$\varepsilon = \prod_t Z_t \leq \exp\left(-2\sum_{t=1}^{T}\gamma_t^2\right) \quad \text{(\#2)}$$