Machine Learning for Image Classification
----Part II: Ensemble Approaches

Jianping Fan
Dept of Computer Science
UNC-Charlotte

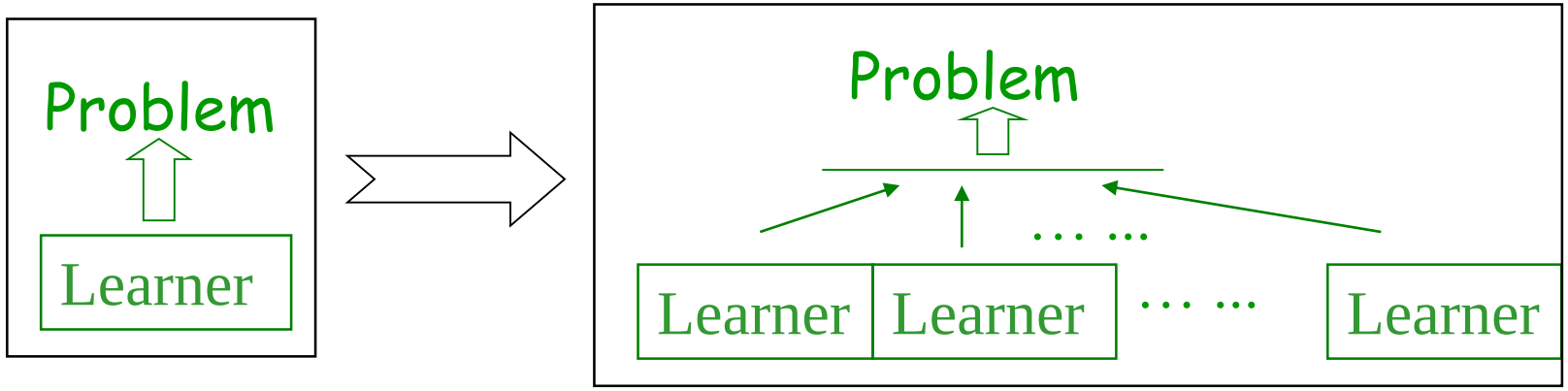
Course Website:
http://webpages.uncc.edu/jfan/itcs5152.html

Ensemble Learning

A machine learning paradigm where multiple learners are used to solve the problem

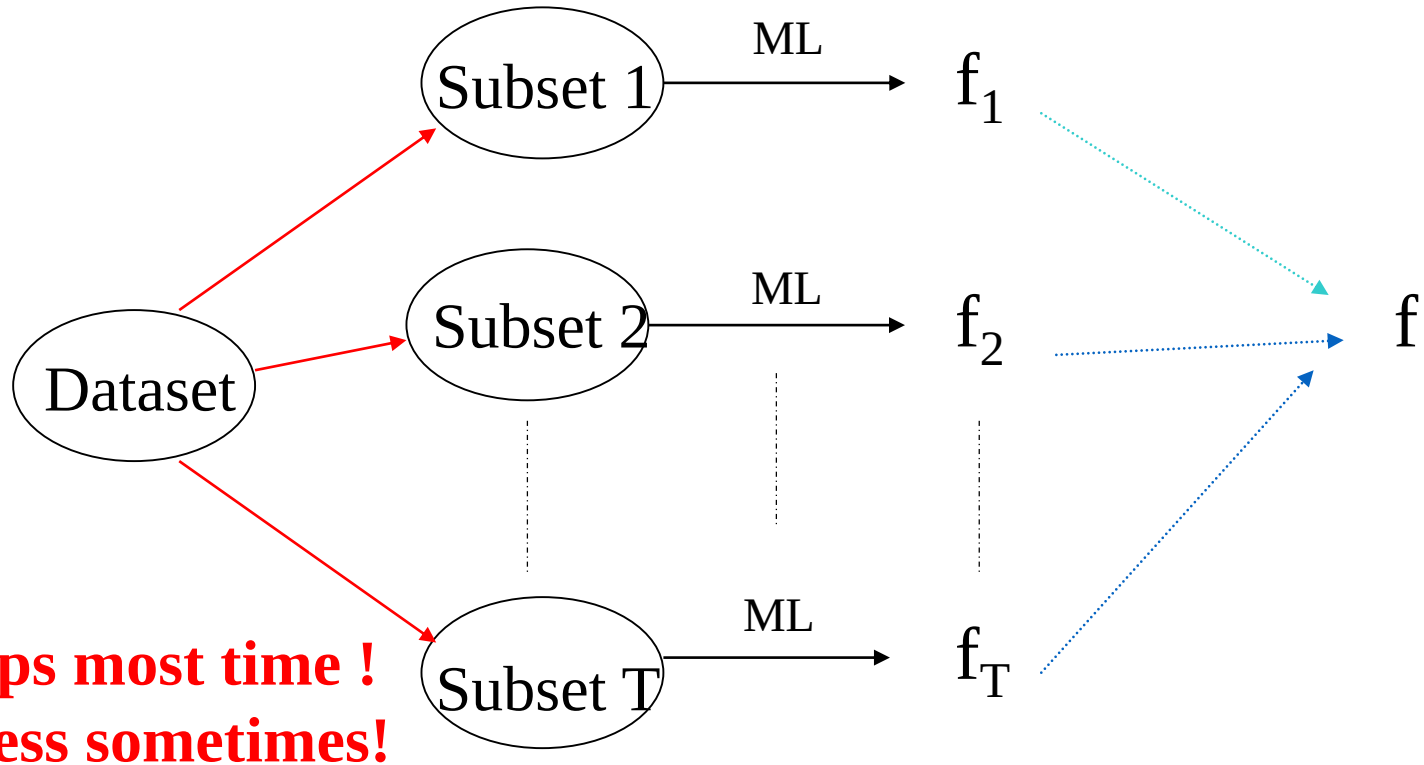
Previously: single classifier

Ensemble: multiple classifiers



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

Ensemble Classifier



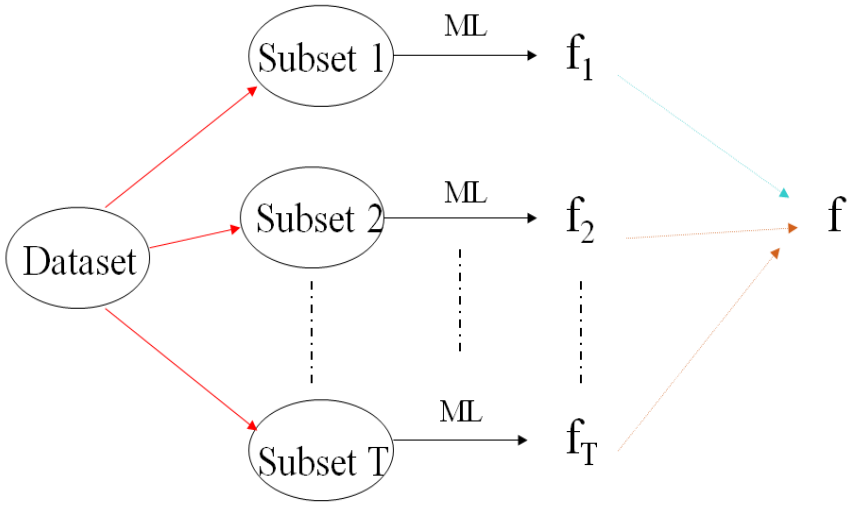
More helps most time !
More is less sometimes!

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

It is not a good idea to randomly combine multiple classifiers together!

Wish List:

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

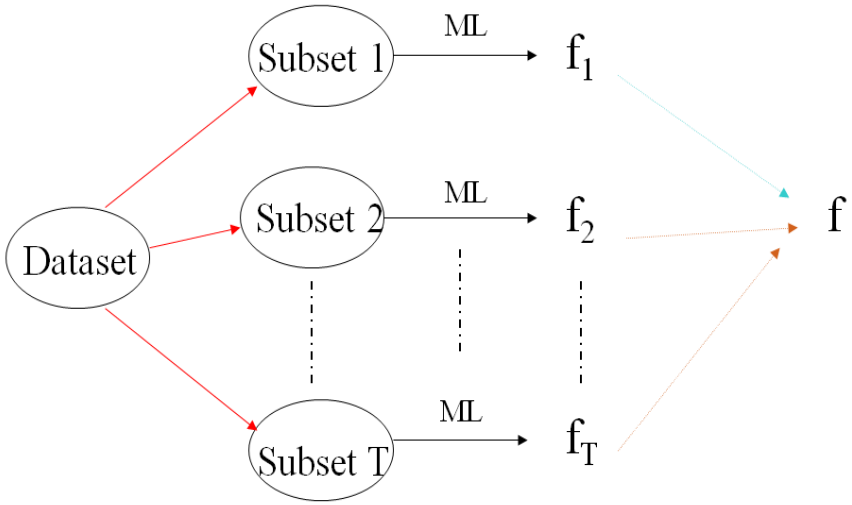


- Each weak classifier may be different from others! Different focuses, capabilities, ……they even can compensate each other!

- Each of them plays different roles!

Ensemble Classifier

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

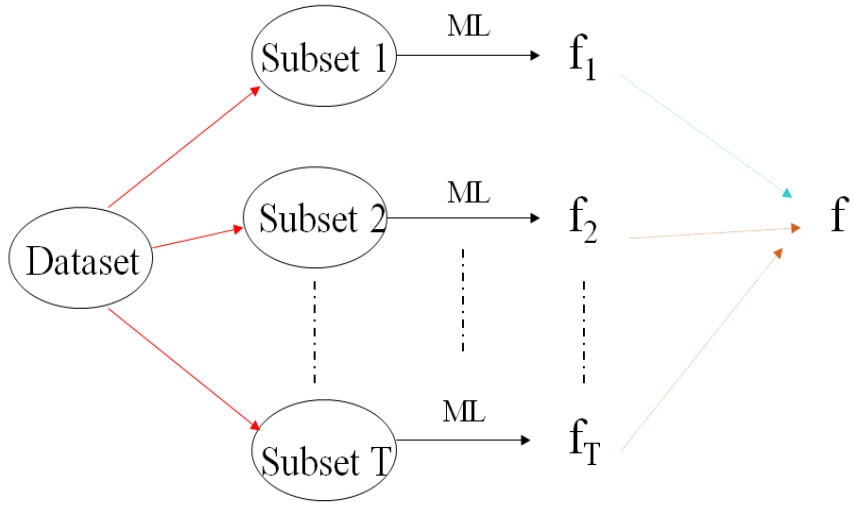


- **Majority voting**: winner take all!
- **Weighted voting**: combine with weights
- **Averaging**: combine with equal weights

Why we learn from data subsets?

Ensemble Classifier

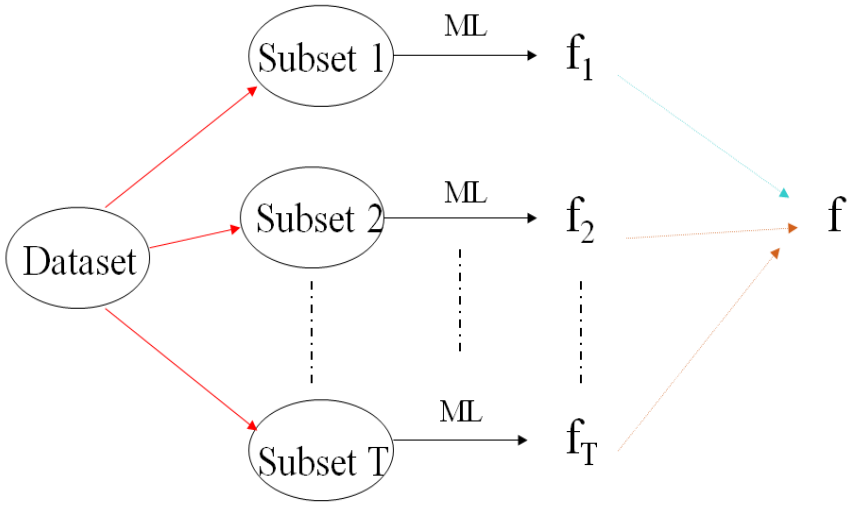
$$H_T(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$



- **What may affect ensemble classifier?**
  - **Diversity** of weak classifiers: we will not hair two ``almost same" persons
  - **Weights** for weak classifier combination: we know they play different not equal roles in final decision

Ensemble Classifier

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

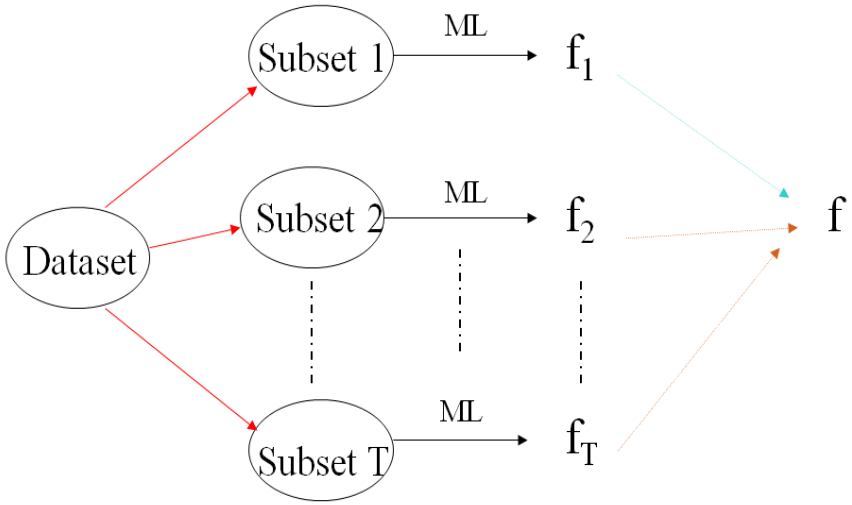


- **How to train a set of classifiers with diverse capabilities?**

1. Using different datasets for the same data-driven learning algorithm
2. Using different learning algorithms to train different classifiers from the same dataset

Ensemble Classifier

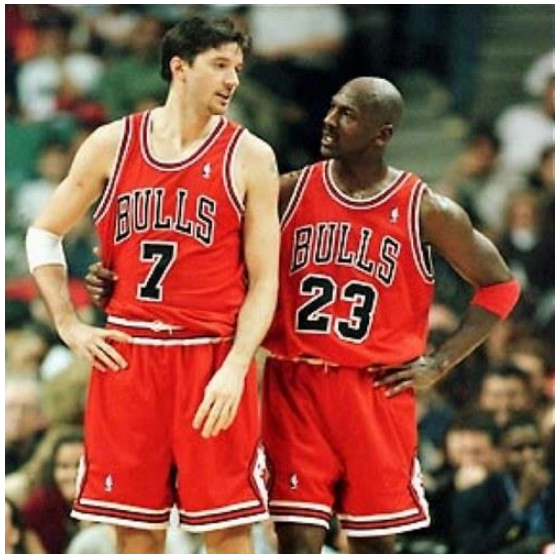
$$H_T(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$

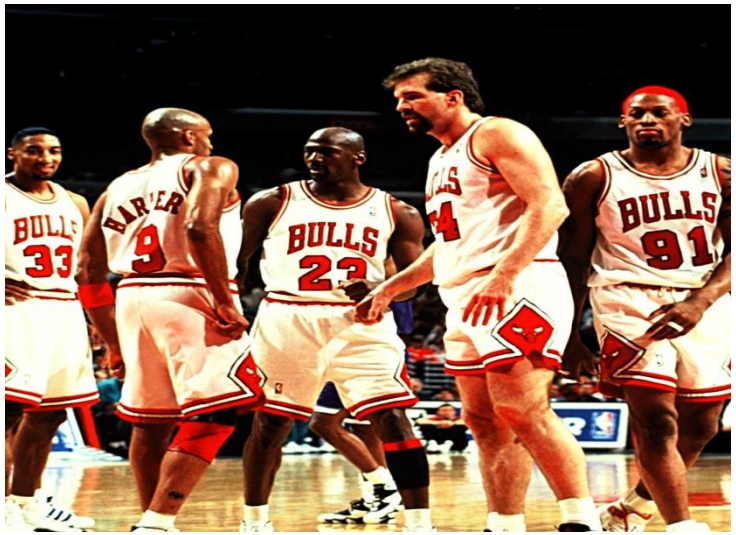


- We may prefer weighted voting for ensemble

How to determine the weights automatically?

Wish Lists: NBA Championship Rule

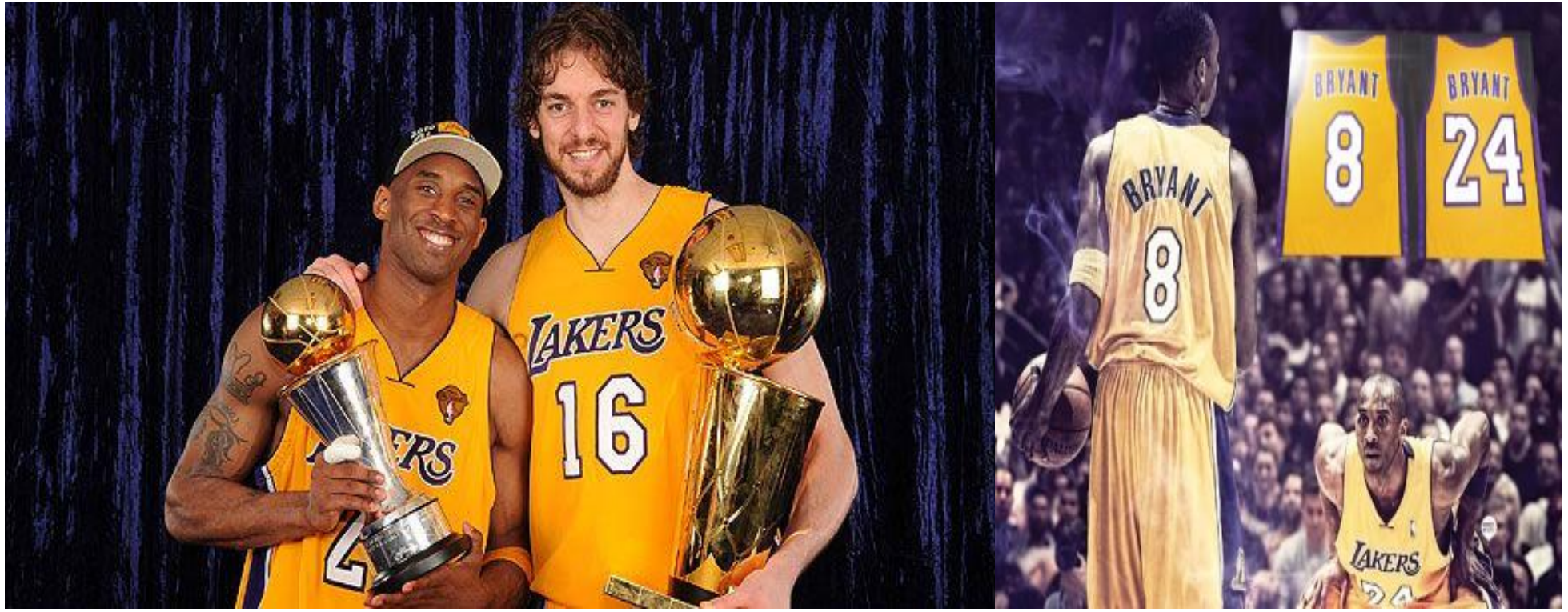




Wish Lists: NBA Championship Rule

Without Shaq, Kobe even cannot got playoffs for several years!

Wish Lists: NBA Championship Rule



Kobe finally finds the solution!

Wish Lists: NBA Championship Rule

How about this man with enough helpers?

Wish Lists: NBA Championship Rule

Yes, you can after I retire or I move to Lakers!

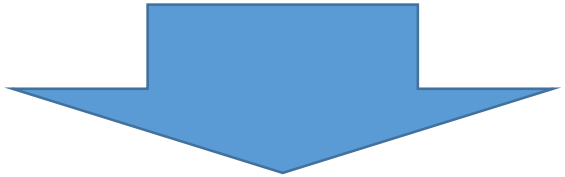
Weak classifiers are not ``weak' at all!
They all are very strong on some places &
They know balance and compensations!

Our observations from NSA examples

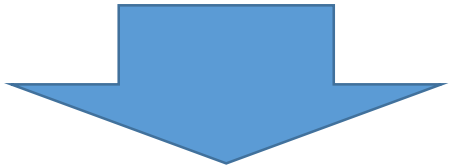
- **Diversity** of weak classifiers is not sufficient, they should compensate each other!
- Weak classifiers **are not weak**! They are very strong at certain places!
- **Weights** should depend on the importance or potential contributions or capabilities!

Diversity of Weak Classifier

- Training different weak classifiers from various data subsets!

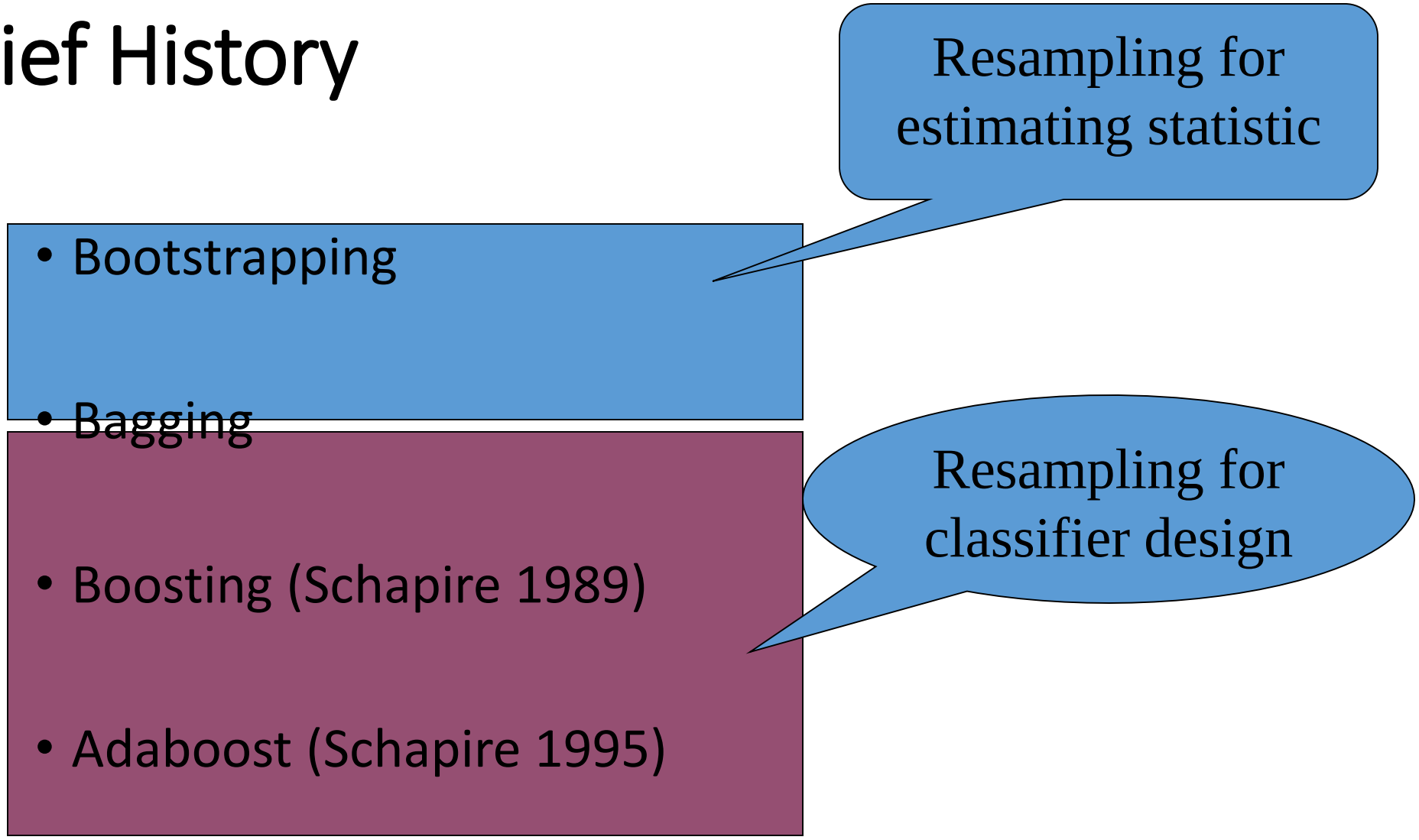


Data-driven learning algorithms may make these weak classifiers to be different!



Sampling various subsets from the same big data set!

A Brief History



How to make weak classifier diverse?

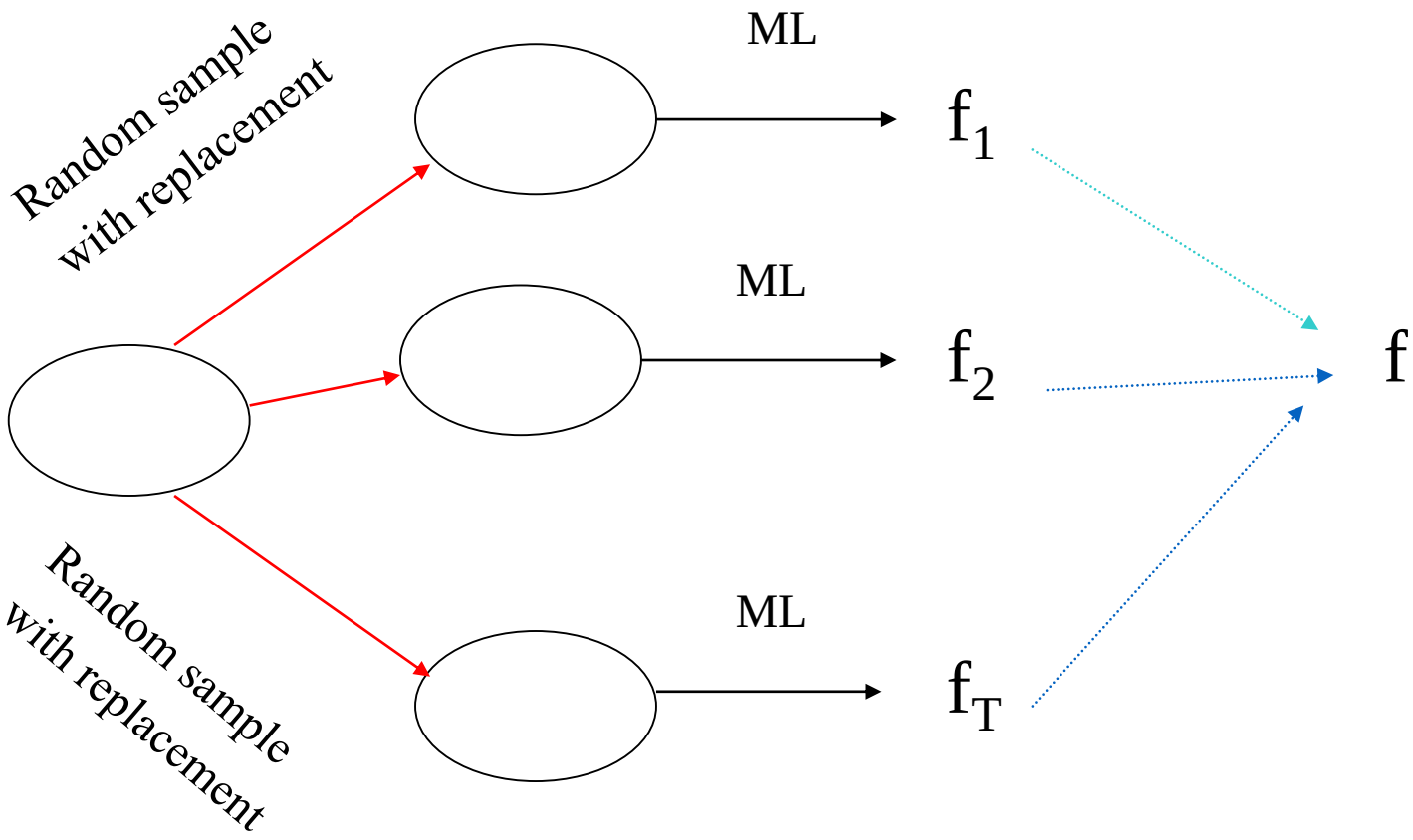
Bootstrap Estimation

- Repeatedly draw *n* samples from *D*
- For each set of samples, estimate a statistic
- The bootstrap estimate is the mean of the individual estimates
- Used to estimate a statistic (parameter) and its variance

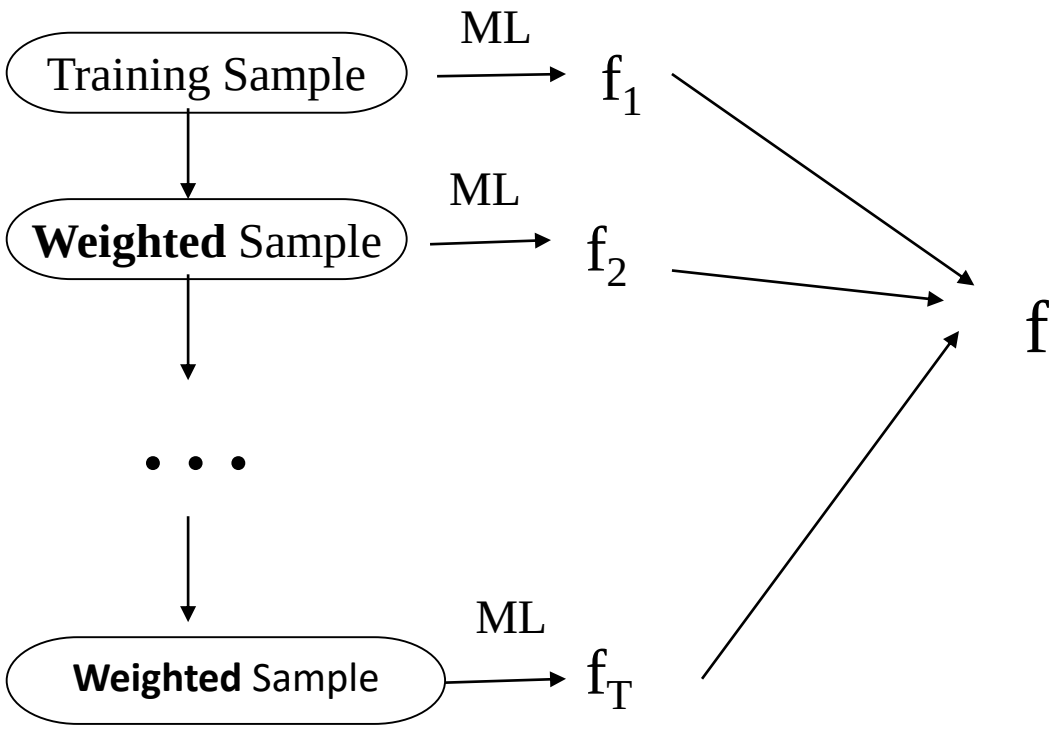
Bagging - Aggregate Bootstrapping

- For i = 1 .. M
  - Draw $n^* < n$ samples from D with replacement
  - Learn classifier C_i
- Final classifier is a vote of C_1 .. C_M
- Increases classifier stability/reduces variance

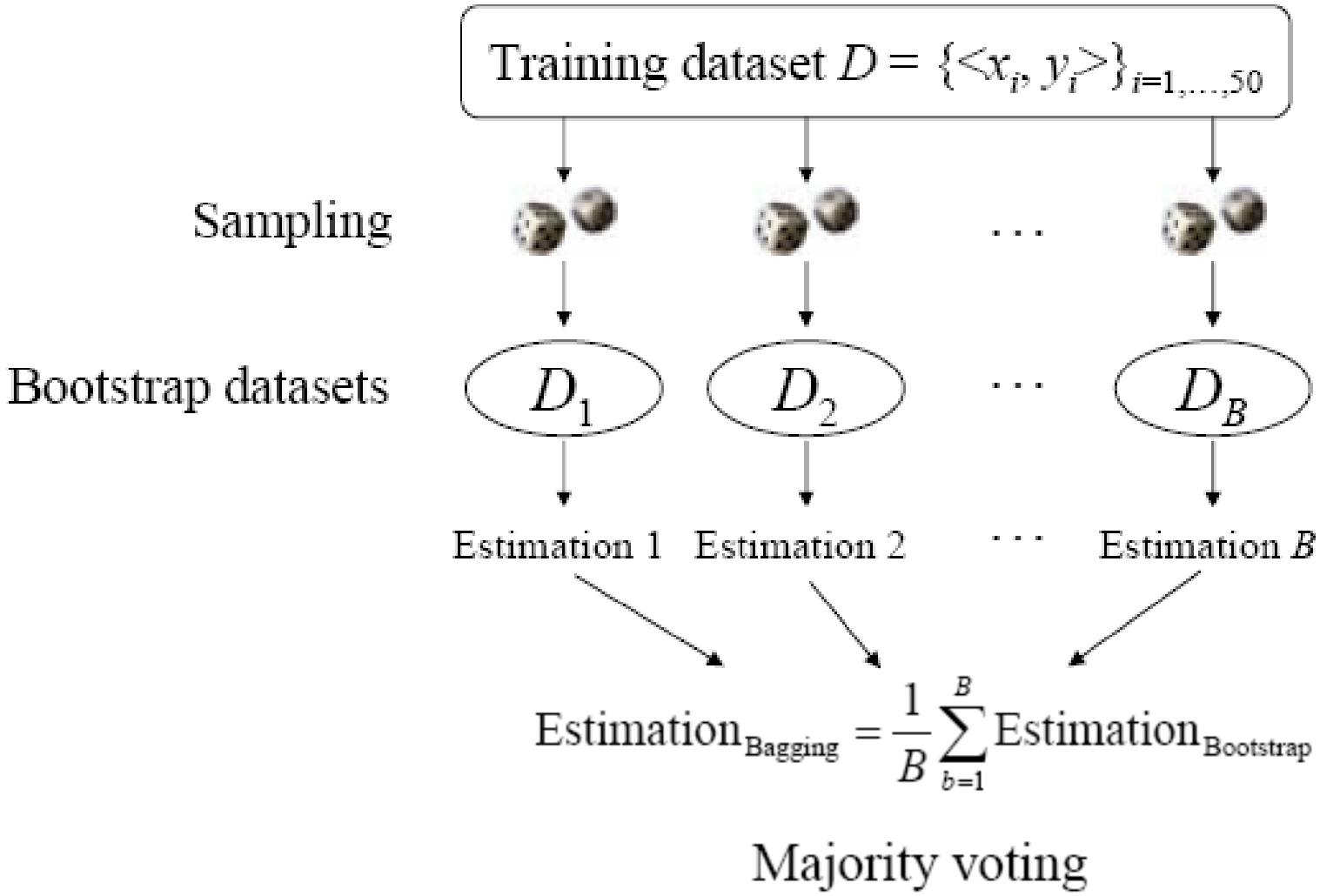
Bagging


Random sample
with replacement
ML
f_1
ML
f_2
Random sample
with replacement
ML
f_T
f

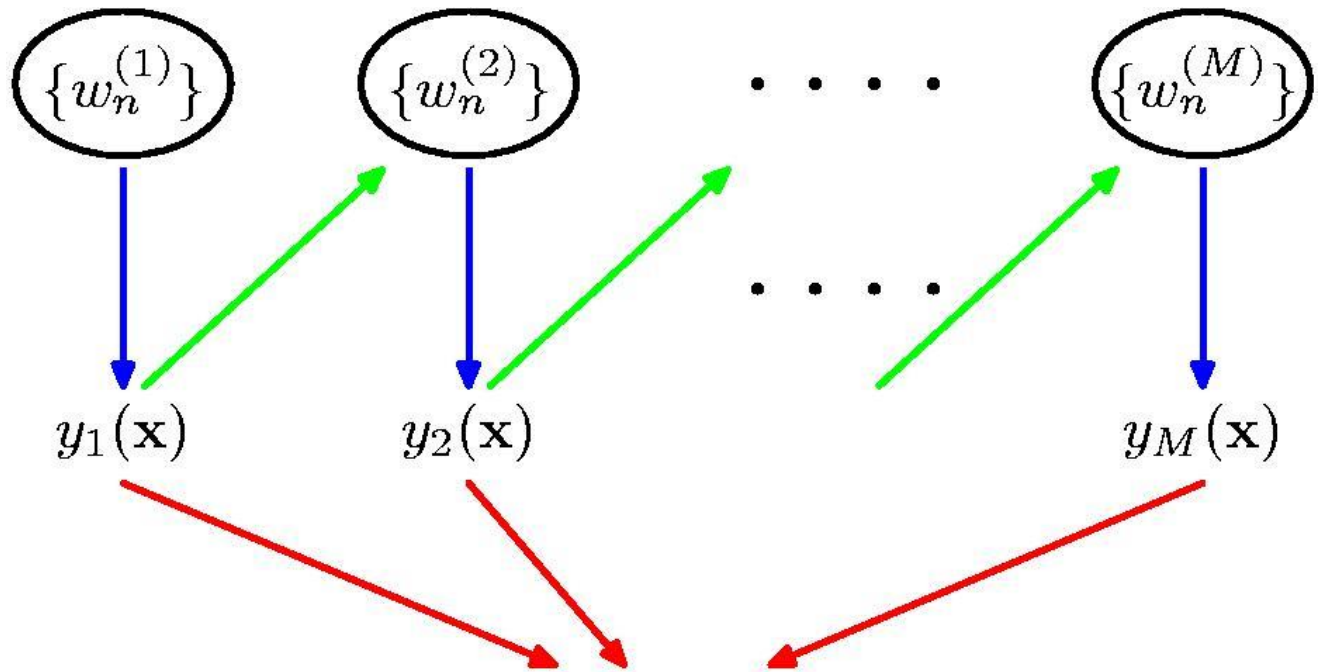
Boosting


Training Sample
ML
f_1
Weighted Sample
ML
f_2
• • •
Weighted Sample
ML
f_T
f

Revisit Bagging

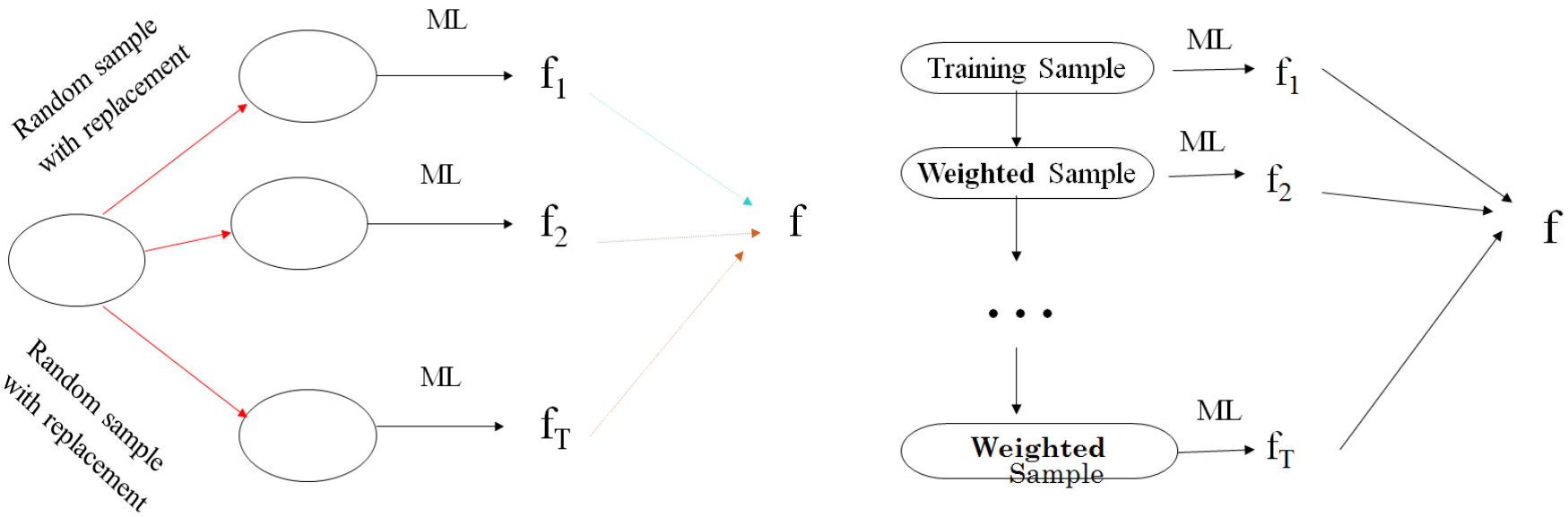


Boosting Classifier



$$Y_M(\mathbf{x}) = \text{sign}\left(\sum_m^M \alpha_m y_m(\mathbf{x})\right)$$

Differences: Bagging *vs*. Boosting



- Boosting brings connections or compensations between data subsets, e.g., they know each other!
- Boosting has special combination rules for weak classifier integration

Bagging *vs* Boosting

- **Bagging**: the construction of complementary base-learners is ***left to chance*** and to the unstability of the learning methods.
- **Boosting**: actively seek to generate complementary base-learner--- training the next base-learner **based on the mistakes of the previous learners**.

Boosting (Schapire 1989)

- Randomly select $n_1 < n$ samples from D without replacement to obtain D_1
  - Train weak learner C_1

- Select $n_2 < n$ samples from D with half of the samples misclassified by C_1 to obtain D_2
  - Train weak learner C_2

- Select all samples from D that C_1 and C_2 disagree on
  - Train weak learner C_3

- Final classifier is vote of weak learners

AdaBoost (Schapire 1995)

- Instead of sampling, re-weight
  - Previous weak learner has only 50% accuracy over new distribution
- Can be used to learn weak classifiers
- Final classification based on weighted vote of weak classifiers

Adaboost Terms

- Learner = Hypothesis = Classifier
- Weak Learner: < 50% error over any distribution
- Strong Classifier: thresholded linear combination of weak learner outputs

AdaBoost

Adaptive Boosting

A learning algorithm

Building a strong classifier a lot of weaker ones

AdaBoost Concept

$$h_1(x) \in \{-1, +1\}$$
$$h_2(x) \in \{-1, +1\}$$
$$\vdots$$
$$h_T(x) \in \{-1, +1\}$$

weak classifiers

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

strong classifier

slightly better than random

AdaBoost

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

- **How to train weak classifiers and make them compensate each other?**

- **How to determine the weights automatically? We do expect such weights depending on their performances and capabilities.**

Weaker Classifiers

$$h_1(x) \in \{-1, +1\}$$
$$h_2(x) \in \{-1, +1\}$$
$$\vdots$$
$$h_T(x) \in \{-1, +1\}$$

weak classifiers

slightly better than random

- Each weak classifier learns by considering one simple feature
- T most beneficial features for classification should be selected
- How to
  - define features?
  - select beneficial features?
  - train weak classifiers?
  - manage (weight) training samples?
  - associate weight to each weak classifier?

The Strong Classifiers

$$h_1(x) \in \{-1, +1\}$$
$$h_2(x) \in \{-1, +1\}$$
$$\vdots$$
$$h_T(x) \in \{-1, +1\}$$

How good the strong one will be?

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

weak classifiers

strong classifier

slightly better than random

The AdaBoost Algorithm

Given: $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in X, y_i \in \{-1, +1\}$

Initialization: $D_1(i) = \frac{1}{m}, i = 1, \ldots, m$ $D_t(i)$:probability distribution of x_i's at time t

For $t = 1, \ldots, T$:

- Find classifier $h_t : X \to \{-1, +1\}$ which minimizes error wrt D_t ,i.e.,

$$h_t = \arg\min_{h_j} \varepsilon_j \quad \text{where } \varepsilon_j = \sum_{i=1}^{m} D_t(i)[y_i \neq h_j(x_i)]$$

minimize weighted error

- Weight classifier: $\alpha_t = \frac{1}{2} \ln \frac{1 - \varepsilon_t}{\varepsilon_t}$

for minimize exponential loss

- Update distribution: $D_{t+1}(i) = \frac{D_t(i) \exp[-\alpha_t y_i h_t(x_i)]}{Z_t}$, Z_t is for normalization

Give error classified patterns more chance for learning.

The AdaBoost Algorithm

Given: $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in X, y_i \in \{-1, +1\}$

Initialization: $D_1(i) = \frac{1}{m}, i = 1, \ldots, m$

For $t = 1, \ldots, T$:

- Find classifier $h_t : X \to \{-1, +1\}$ which minimizes error wrt D_t ,i.e.,

$$h_t = \arg\min_{h_j} \varepsilon_j \quad \text{where } \varepsilon_j = \sum_{i=1}^{m} D_t(i)[y_i \neq h_j(x_i)]$$

- Weight classifier: $\alpha_t = \frac{1}{2} \ln \frac{1 - \varepsilon_t}{\varepsilon_t}$

- Update distribution: $D_{t+1}(i) = \frac{D_t(i) \exp[-\alpha_t y_i h_t(x_i)]}{Z_t}$, Z_t is for normalization

Output final classifier: $sign\left(H(x) = \sum_{t=1}^{T} \alpha_t h_t(x) \right)$

Observations for AdaBoost

- **Diversity** of weak classifiers is enhances by compensation: the current weak classifier focuses on the samples which the previous ones make wrong predictions!

$$h_t = \arg\min_{h_j} \varepsilon_j \quad \text{where } \varepsilon_j = \sum_{i=1}^{m} D_t(i)[y_i \neq h_j(x_i)] \qquad D_{t+1}(i) = \frac{D_t(i)\exp[-\alpha_t y_i h_t(x_i)]}{Z_t}$$

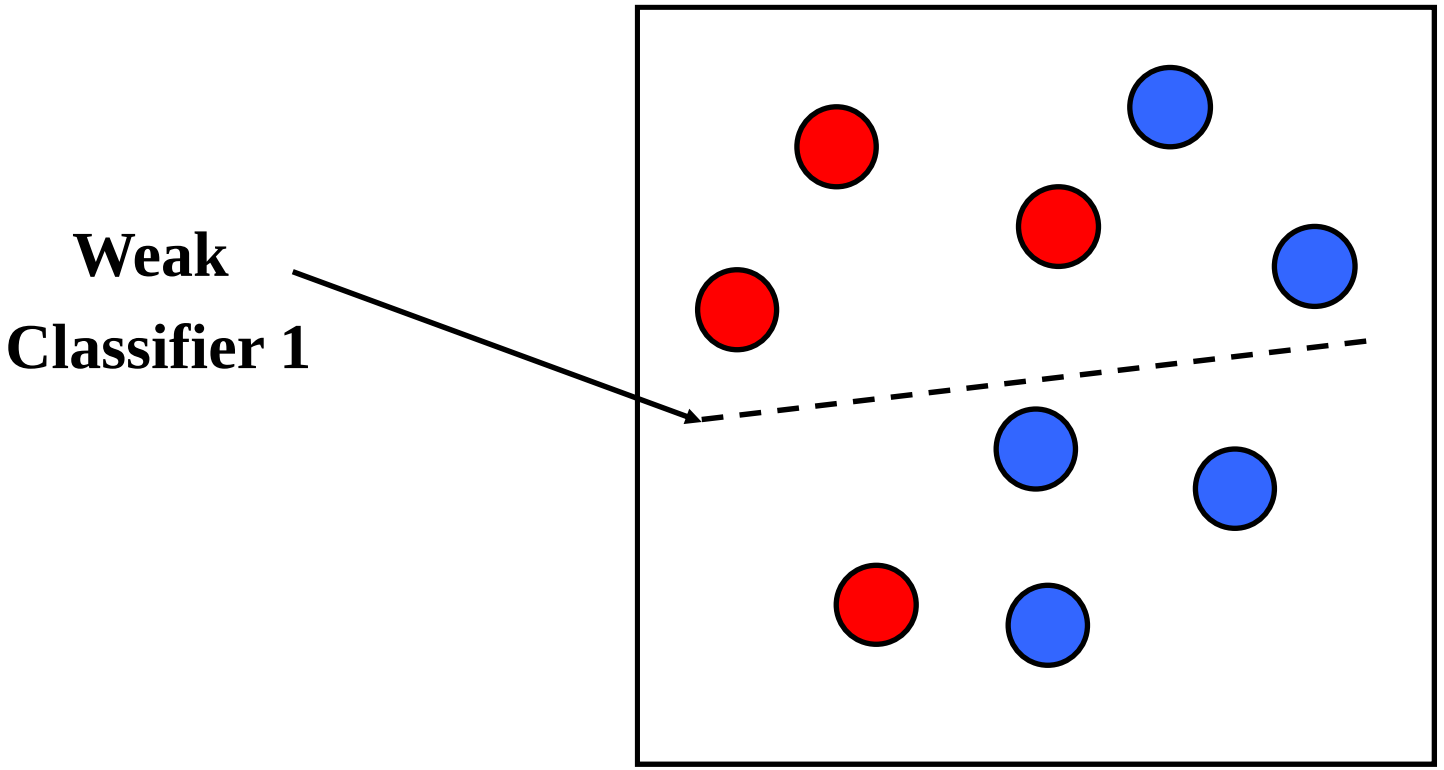
- **Weights** for weak classifier combination largely depends on their performance or capabilities!

$$sign\left(H(x) = \sum_{t=1}^{T} \alpha_t h_t(x)\right) \qquad \alpha_t = \frac{1}{2}\ln\frac{1-\varepsilon_t}{\varepsilon_t}$$

Compare these with our wish lists!

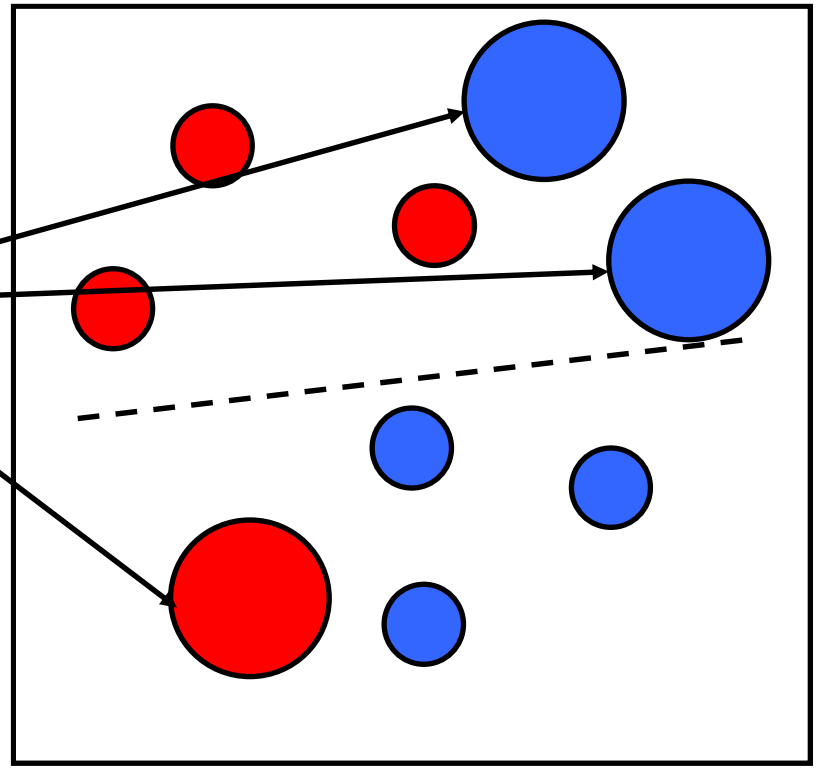
Boosting illustration

Some samples are misclassified!



Boosting illustration

Weights increased
for misclassified samples!
& new weak classifier will
pay more attention on them!



The AdaBoost Algorithm

Given: $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in X$, $y_i \in Y = \{-1, +1\}$

Initialize $D_1(i) = 1/m$.

For $t = 1, \ldots, T$:

- Train base learner using distribution D_t.
- Get base classifier $h_t : X \to \mathbb{R}$.
- Choose $\alpha_t \in \mathbb{R}$.
- Update:

$$D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

where Z_t is a normalization tion).

typically $\alpha_t = \frac{1}{2} \ln \left(\frac{1 - \epsilon_t}{\epsilon_t} \right)$ where $\epsilon_t = \Pr_{i \sim D_t} [h_t(x_i) \neq y_i]$

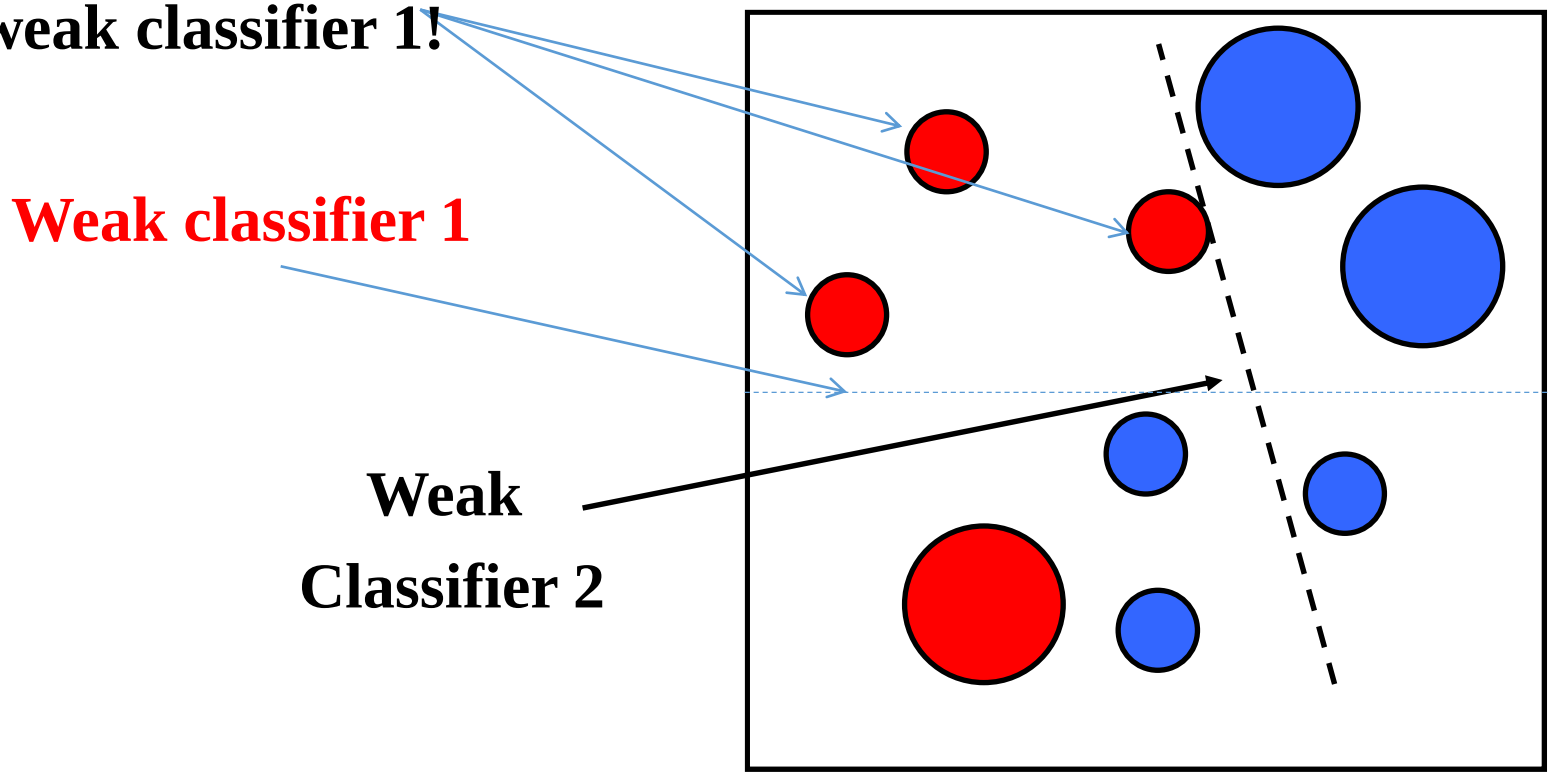
the weights of incorrectly classified examples are increased so that the base learner is forced to focus on the hard examples in the training set

Output the final classifier:

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

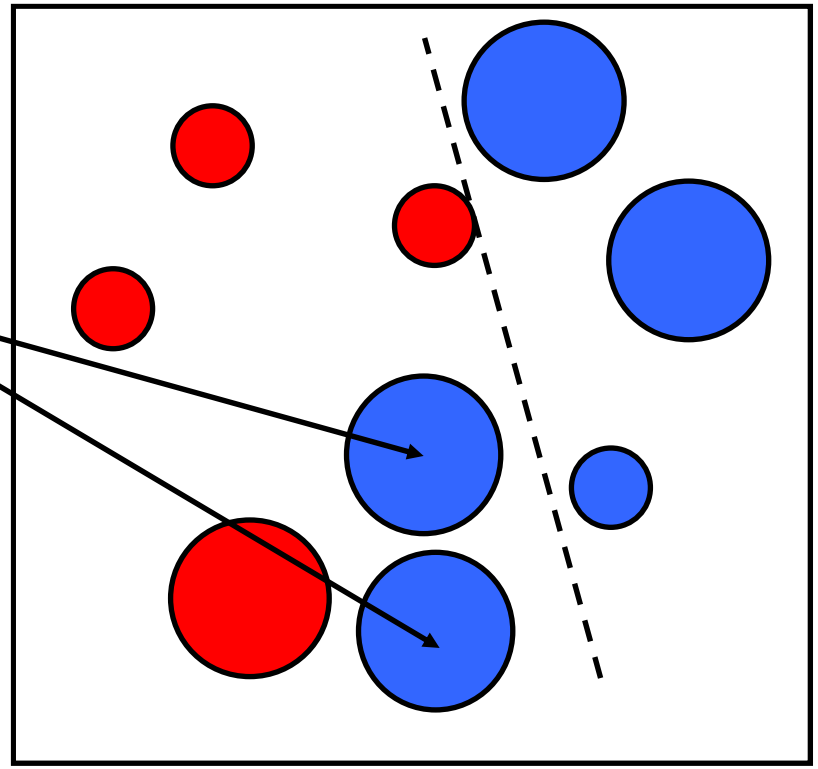
Boosting illustration

Weak classifier 2 will not pay attention on these samples which have good predictions by weak classifier 1!



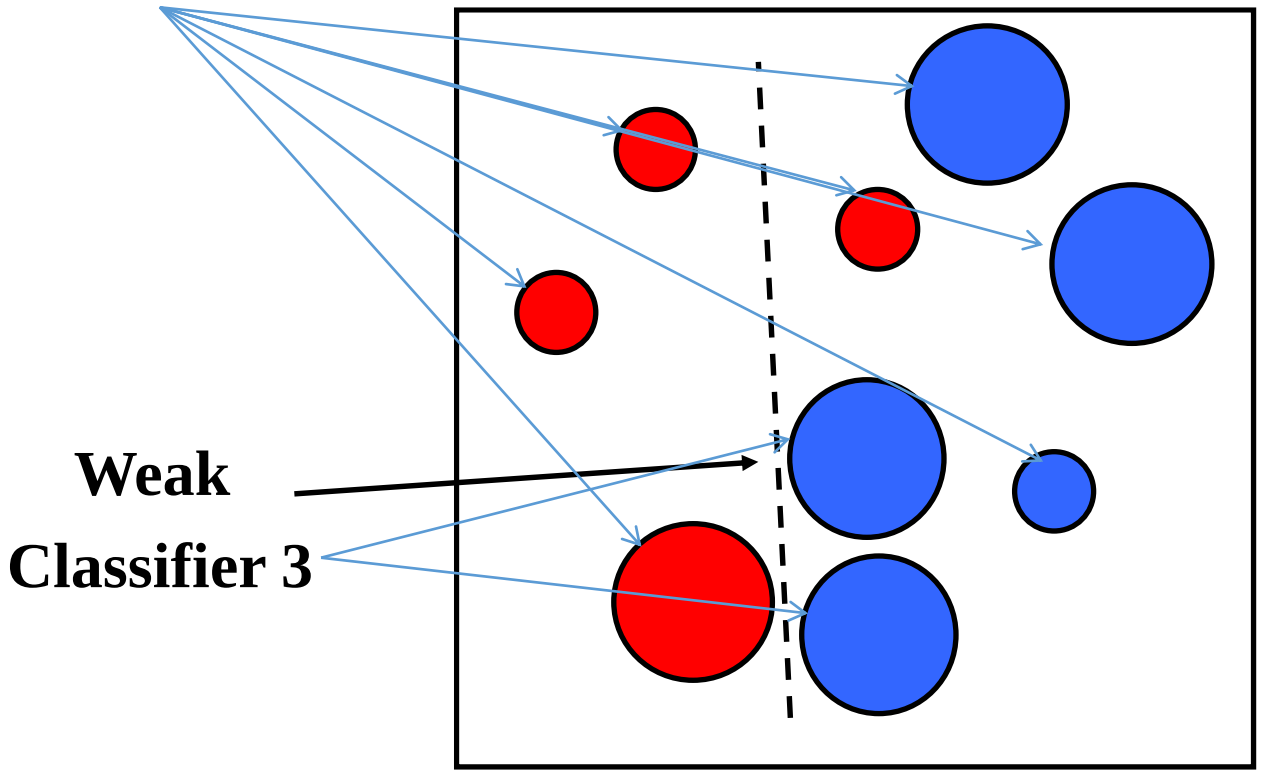
Boosting illustration

Weights increased
for misclassified samples!
& new weak classifier will
pay more attention on them!



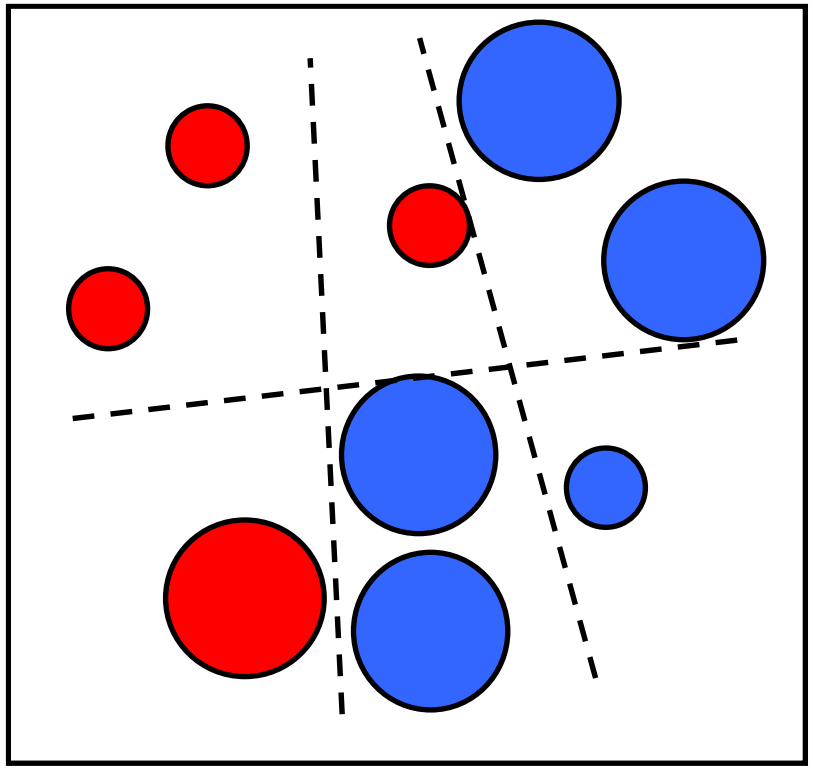
Boosting illustration

Weak classifier 3 will not pay attention on these samples which have good predictions by weak classifiers 1 & 2!



Boosting illustration

Final classifier is a combination of 3 weak classifiers



Observations from this intuitive example

- **Current weak classifier** pay more attentions on the samples which are misclassified by the previous weak classifiers, thus they can compensate each other on final decision!
- It has provided an **easy-to-hard** solution for weak classifier training!
- **Weights** for weak classifier combination largely depends on their performance or capabilities!

What goal the AdaBoost wants to reach?

The AdaBoost Algorithm

Given: $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in X, y_i \in \{-1, +1\}$

Initialization: $D_1(i) = \frac{1}{m}, i = 1, \ldots, m$

For $t = 1, \ldots, T$:

- Find classifier $h_t : X \to \{-1, +1\}$ which minimizes error wrt D_t ,i.e.,

$$h_t = \arg\min_{h_j} \varepsilon_j \quad \text{where } \varepsilon_j = \sum_{i=1}^{m} D_t(i)[y_i \neq h_j(x_i)]$$

- Weight classifier: $\alpha_t = \frac{1}{2} \ln \frac{1 - \varepsilon_t}{\varepsilon_t}$

- Update distribution: $D_{t+1}(i) = \frac{D_t(i) \exp[-\alpha_t y_i h_t(x_i)]}{Z_t}$, Z_t is for normalization

Output final classifier: $sign\left(H(x) = \sum_{t=1}^{T} \alpha_t h_t(x) \right)$

What goal the AdaBoost wants to reach?

The AdaBoost Algorithm

Given: $(x_1, y_1), \ldots, (x_m, y_m)$ wh

Initialization: $D_1(i) = \frac{1}{}$

For $t = 1, \ldots, T$:

They are goal dependent.

- Find classifier $h_t : X \to \{-1,$ minimizes error wrt D_t ,i.e.,

$$h_t = \arg\min_{h_j} \varepsilon_j \quad \text{where } \varepsilon_j = \sum_{i=1}^{m} D_t(i)[y_i \neq h_j(x_i)]$$

- Weight classifier: $\alpha_t = \frac{1}{2} \ln \frac{1-\varepsilon_t}{\varepsilon_t}$

- Update distribution: $D_{t+1}(i) = \frac{D_t(i) \exp[-\alpha_t y_i h_t(x_i)]}{Z_t}$ where Z_t is for normalization

Output final classifier: $sign\left(H(x) = \sum_{t=1}^{T} \alpha_t h_t(x) \right)$

Goal

Final classifier: $sign\left(H(x) = \sum_{t=1}^{T} \alpha_t h_t(x) \right)$

Minimize exponential loss

$$loss_{\exp}\left[H(x)\right] = E_{x,y}\left[e^{-yH(x)}\right]$$

Goal

Final classifier: $sign\left(H(x) = \sum_{t=1}^{T} \alpha_t h_t(x) \right)$

Minimize exponential loss

$$loss_{\exp}\left[H(x)\right] = E_{x,y}\left[e^{-yH(x)}\right]$$

Maximize the margin $yH(x)$

Goal

Minimize $loss_{\exp}\left[H(x)\right] = E_{x,y}\left[e^{-yH(x)}\right]$

Final classifier: $sign\left(H(x) = \sum_{t=1}^{T} \alpha_t h_t(x)\right)$

Define $H_t(x) = H_{t-1}(x) + \alpha_t h_t(x)$ with $H_0(x) = 0$

Then, $H(x) = H_T(x)$

$$E_{x,y}\left[e^{-yH_t(x)}\right] = E_x\left[E_y\left[e^{-yH_t(x)} \mid x\right]\right]$$

$$= E_x\left[E_y\left[e^{-y[H_{t-1}(x)+\alpha_t h_t(x)]} \mid x\right]\right]$$

$$= E_x\left[E_y\left[e^{-yH_{t-1}(x)}e^{-y\alpha_t h_t(x)} \mid x\right]\right]$$

$$= E_x\left[e^{-yH_{t-1}(x)}\left[e^{-\alpha_t}P(y = h_t(x)) + e^{\alpha_t}P(y \neq h_t(x))\right]\right]$$

Minimize $loss_{\exp}\left[H(x)\right] = E_{x,y}\left[e^{-yH(x)}\right]$

$\alpha_t = ?$

Final classifier: $sign\left(H(x) = \sum_{t=1}^{T} \alpha_t h_t(x)\right)$

Define $H_t(x) = H_{t-1}(x) + \alpha_t h_t(x)$ with $H_0(x) = 0$

Then, $H(x) = H_T(x)$

$$E_{x,y}\left[e^{-yH_t(x)}\right] = E_x\left[e^{-yH_{t-1}(x)}\left[e^{-\alpha_t}P(y = h_t(x)) + e^{\alpha_t}P(y \neq h_t(x))\right]\right]$$

Set $\frac{\partial}{\partial \alpha_t} E_{x,y}\left[e^{-yH_t(x)}\right] = 0$

$$\Rightarrow E_x\left[e^{-yH_{t-1}(x)}\left[\underbrace{-e^{-\alpha_t}P(y = h_t(x)) + e^{\alpha_t}P(y \neq h_t(x))}_{0}\right]\right] = 0$$

Minimize $loss_{\exp}\left[H(x)\right] = E_{x,y}\left[e^{-yH(x)}\right]$

$\alpha_t = ?$

Final classifier: $sign\left(H(x) = \sum_{t=1}^{T} \alpha_t h_t(x)\right)$

Define $H_t(x) = H_{t-1}(x) + \alpha_t h_t(x)$ with $H_0(x) = 0$

Then, $H(x) = H_T(x)$

$$\Rightarrow \alpha_t = \frac{1}{2}\ln\frac{P(y = h_t(x))}{P(y \neq h_t(x))} \quad \Rightarrow \alpha_t = \frac{1}{2}\ln\frac{1-\varepsilon_t}{\varepsilon_t}$$

$P(x_i, y_i) = D_t(i)$

$$\varepsilon_t = P(\text{error}) \approx \sum_{i=1}^{m} D_t(i)[y_i \neq h_j(x_i)]$$

$$\Rightarrow E_x\left[e^{-yH_{t-1}(x)}\left[\underbrace{-e^{-\alpha_t}P(y = h_t(x)) + e^{\alpha_t}P(y \neq h_t(x))}_{0}\right]\right] = 0$$

$\alpha_t = ?$

Given: $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in X, y_i \in \{-1, +1\}$

Initialization: $D_1(i) = \frac{1}{m}, i = 1, \ldots, m$

For $t = 1, \ldots, T$:

- Find classifier $h_t : X \to \{-1, +1\}$ which minimizes error wrt D_t ,i.e.,

$$h_t = \arg\min_{h_j} \varepsilon_j \quad \text{where } \varepsilon_j = \sum_{i=1}^{m} D_t(i)[y_i \neq h_j(x_i)]$$

- Weight classifier: $\alpha_t = \frac{1}{2} \ln \frac{1-\varepsilon_t}{\varepsilon_t}$

- Update distribution: $D_{t+1}(i) = \frac{D_t(i) \exp[-\alpha_t y_i h_t(x_i)]}{Z_t}$, Z_t is for normalization

Output final classifier: $sign\left(H(x) = \sum_{t=1}^{T} \alpha_t h_t(x) \right)$

Define $H_t(x) = \ldots$

Then, $H(x) = H \ldots$

$$\Rightarrow \alpha_t = \frac{1}{2} \ln \frac{P(y = h_t(x))}{P(y \neq h_t(x))} \Rightarrow \alpha_t = \frac{1}{2} \ln \frac{1-\varepsilon_t}{\varepsilon_t}$$

$$P(x_i, y_i) = D_t(i)$$

$$\varepsilon_t = P(\text{error}) \approx \sum_{i=1}^{m} D_t(i)[y_i \neq h_j(x_i)]$$

$$\Rightarrow E_x \left[e^{-yH_{t-1}(x)} \left[\underbrace{-e^{-\alpha_t} P(y = h_t(x)) + e^{\alpha_t} P(y \neq h_t(x))}_{0} \right] \right] = 0$$

$D_{t+1} = ?$

Given: $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in X, y_i \in \{-1, +1\}$

Initialization: $D_1(i) = \frac{1}{m}, i = 1, \ldots, m$

For $t = 1, \ldots, T$:

- Find classifier $h_t : X \to \{-1, +1\}$ which minimizes error wrt D_t ,i.e.,

$$h_t = \arg\min_{h_j} \varepsilon_j \quad \text{where} \quad \varepsilon_j = \sum_{i=1}^{m} D_t(i)[y_i \neq h_j(x_i)]$$

- Weight classifier: $\alpha_t = \frac{1}{2} \ln \frac{1 - \varepsilon_t}{\varepsilon_t}$

- Update distribution: $D_{t+1}(i) = \frac{D_t(i) \exp[-\alpha_t y_i h_t(x_i)]}{Z_t}$, Z_t is for normalization

Output final classifier: $sign\left(H(x) = \sum_{t=1}^{T} \alpha_t h_t(x) \right)$

Define $H_t(x) =$

Then, $H(x) = H$

$$\Rightarrow \alpha_t = \frac{1}{2} \ln \frac{P(y = h_t(x))}{P(y \neq h_t(x))} \quad \Rightarrow \alpha_t = \frac{1}{2} \ln \frac{1 - \varepsilon_t}{\varepsilon_t}$$

$$P(x_i, y_i) = D_t(i)$$

$$\varepsilon_t = P(\text{error}) \approx \sum_{i=1}^{m} D_t(i)[y_i \neq h_j(x_i)]$$

$$\Rightarrow E_x \left[e^{-yH_{t-1}(x)} \left[\underbrace{-e^{-\alpha_t} P(y = h_t(x)) + e^{\alpha_t} P(y \neq h_t(x))}_{0} \right] \right] = 0$$

Minimize $loss_{\exp}[H(x)] = E_{x\sim D,y}\left[e^{-yH(x)}\right]$

$D_{t+1} = ?$

Final classifier: $sign\left(H(x) = \sum_{t=1}^{T} \alpha_t h_t(x)\right)$

Define $H_t(x) = H_{t-1}(x) + \alpha_t h_t(x)$ with $H_0(x) = 0$

Then, $H(x) = H_T(x)$

$$E_{x,y}\left[e^{-yH_t}\right] = E_{x,y}\left[e^{-yH_{t-1}}e^{-y\alpha_t h_t}\right] \approx E_{x,y}\left[e^{-yH_{t-1}}\left(1 - y\alpha_t h_t + \frac{1}{2}\alpha_t^2 y^2 h_t^2\right)\right]$$

$$\Rightarrow h_t = \arg\min_h E_{x,y}\left[e^{-yH_{t-1}}\left(1 - y\alpha_t h + \frac{1}{2}\alpha_t^2 y^2 h^2\right)\right] \qquad y^2h^2 = 1$$

$$\Rightarrow h_t = \arg\min_h E_{x,y}\left[e^{-yH_{t-1}}\left(1 - y\alpha_t h + \frac{1}{2}\alpha_t^2\right)\right]$$

$$\Rightarrow h_t = \arg\min_h E_x\left[E_y\left[e^{-yH_{t-1}}\left(1 - y\alpha_t h + \frac{1}{2}\alpha_t^2\right)\right] | \, x\right]$$

Minimize $loss_{\exp}\left[H(x)\right] = E_{x\sim D,y}\left[e^{-yH(x)}\right]$

$D_{t+1} = ?$

Final classifier: $sign\left(H(x) = \sum_{t=1}^{T} \alpha_t h_t(x)\right)$

Define $H_t(x) = H_{t-1}(x) + \alpha_t h_t(x)$ with $H_0(x) = 0$

Then, $H(x) = H_T(x)$

$$\Rightarrow h_t = \arg\max_h E_x\left[1\cdot h(x)e^{-H_{t-1}(x)}\cdot P(y=1\,|\,x) + (-1)\cdot h(x)e^{H_{t-1}(x)}\cdot P(y=-1\,|\,x)\right]$$

$$\Rightarrow h_t = \arg\max_h E_x\left[E_y\left[e^{-yH_{t-1}}(yh)\right]|\,x\right]$$

$$\Rightarrow h_t = \arg\min_h E_x\left[E_y\left[e^{-yH_{t-1}}(-y\alpha_t h)\right]|\,x\right]$$

$$\Rightarrow h_t = \arg\min_h E_x\left[E_y\left[e^{-yH_{t-1}}\left(1 - y\alpha_t h + \frac{1}{2}\alpha_t^2\right)\right]|\,x\right]$$

Minimize $loss_{\exp}\left[H(x)\right] = E_{x\sim D,y}\left[e^{-yH(x)}\right]$

$D_{t+1} = ?$

Final classifier: $sign\left(H(x) = \sum_{t=1}^{T} \alpha_t h_t(x)\right)$

Define $H_t(x) = H_{t-1}(x) + \alpha_t h_t(x)$ with $H_0(x) = 0$

Then, $H(x) = H_T(x)$

$$\Rightarrow h_t = \arg\max_h E_x\left[1\cdot h(x)e^{-H_{t-1}(x)}\cdot P(y=1\mid x) + (-1)\cdot h(x)e^{H_{t-1}(x)}\cdot P(y=-1\mid x)\right]$$

$$\Rightarrow h_t = \arg\max_h E_{x,y\sim e^{-yH_{t-1}(x)}P(y|x)}\left[yh(x)\right]$$

maximized when $y = h(x) \quad \forall x$

$$\Rightarrow h_t(x) = sign\left(E_{x,y\sim e^{-yH_{t-1}(x)}P(y|x)}\left[y\mid x\right]\right)$$

$$\Rightarrow h_t(x) = sign\left(P_{x,y\sim e^{-yH_{t-1}(x)}P(y|x)}(y=1\mid x) - P_{x,y\sim e^{-yH_{t-1}(x)}P(y|x)}(y=-1\mid x)\right)$$

Minimize $loss_{\exp}\left[H(x)\right] = E_{x\sim D,y}\left[e^{-yH(x)}\right]$

$D_{t+1} = ?$

Final classifier: $sign\left(H(x) = \sum_{t=1}^{T} \alpha_t h_t(x)\right)$

Define $H_t(x) = H_{t-1}(x) + \alpha_t h_t(x)$ with $H_0(x) = 0$

Then, $H(x) = H_T(x)$

At time t $x, y \sim e^{-yH_{t-1}(x)} P(y \mid x)$

$\Rightarrow h_t(x) = sign\left(P_{x,y\sim e^{-yH_{t-1}(x)}P(y|x)}(y=1 \mid x) - P_{x,y\sim e^{-yH_{t-1}(x)}P(y|x)}(y=-1 \mid x)\right)$

$D_{t+1} = ?$

Given: $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in X, y_i \in \{-1, +1\}$

Initialization: $D_1(i) = \frac{1}{m}, i = 1, \ldots, m$

For $t = 1, \ldots, T$:

- Find classifier $h_t : X \to \{-1, +1\}$ which minimizes error wrt D_t, i.e.,

$$h_t = \arg\min_{h_j} \varepsilon_j \quad \text{where} \quad \varepsilon_j = \sum_{i=1}^{m} D_t(i)[y_i \neq h_j(x_i)]$$

- Weight classifier: $\alpha_t = \frac{1}{2} \ln \frac{1 - \varepsilon_t}{\varepsilon_t}$

- Update distribution: $D_{t+1}(i) = \frac{D_t(i) \exp[-\alpha_t y_i h_t(x_i)]}{Z_t}$, Z_t is for normalization

Output final classifier: $sign\left(H(x) = \sum_{t=1}^{T} \alpha_t h_t(x) \right)$

Define $H_t(x) =$

Then, $H(x) = H$

At time t $\quad x, y \sim e^{-yH_{t-1}(x)} P(y \mid x)$

At time 1 $\quad x, y \sim P(y \mid x)$ $\qquad P(y_i \mid x_i) = 1 \Rightarrow D_1(1) = \frac{1}{Z_1} = \frac{1}{m}$

At time t+1 $\quad x, y \sim e^{-yH_t(x)} P(y \mid x) \equiv D_t e^{-\alpha_t y h_t(x)}$

$$\Rightarrow D_{t+1}(i) = \frac{D_t(i) \exp[-\alpha_t y_i h_t(x_i)]}{Z_t}, \; Z_t \text{ is for normalization}$$

Pros and cons of AdaBoost

Advantages

- Very simple to implement
- Does feature selection resulting in relatively simple classifier
- Fairly good generalization

Disadvantages

- Suboptimal solution
- Sensitive to noisy data and outliers

Intuition

- Train a set of weak hypotheses: h_1, …., h_T.

- The combined hypothesis H is a **weighted** majority vote of the T weak hypotheses.

  ➔Each hypothesis h_t has a weight α_t.

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

- During the training, focus on the examples that are misclassified.

  ➔ At round t, example x_i has the weight $D_t(i)$.

Basic Setting

- Binary classification problem
- Training data:

$$(x_1, y_1), \ldots, (x_m, y_m), \text{where } x_i \in X, y_i \in Y = \{-1, 1\}$$

- $D_t(i)$: the weight of x_i at round t. $D_1(i)=1/m$.

- A learner L that finds a weak hypothesis h_t: X ➔ Y given the training set and D_t

- The error of a weak hypothesis h_t:

$$\epsilon_t = \Pr_{i \sim D_t}[h_t(x_i) \neq y_i] = \sum_{i: h_t(x_i) \neq y_i} D_t(i).$$

The basic AdaBoost algorithm

For t=1, …, T

- Train weak learner using training data and D_t
- Get ht: X ➔ {-1,1} with error $\varepsilon_t = \sum_{i:h_t(x_i)\neq y_i} D_t(i)$
- Choose $\alpha_t = \frac{1}{2}\ln\frac{1-\varepsilon_t}{\varepsilon_t}$
- Update $D_{t+1}(i) = \frac{D_t(i)}{Z_t} * \begin{cases} e^{-\alpha_t} & \text{if } h_t(x_i) = y_i \\ e^{\alpha_t} & \text{if } h_t(x_i) \neq y_i \end{cases}$

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

The general AdaBoost algorithm

For $t = 1, \ldots, T$:

- Train weak learner using distribution D_t.
- Get weak hypothesis $h_t : \mathcal{X} \to \mathbb{R}$.
- Choose $\alpha_t \in \mathbb{R}$.
- Update:

$$D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

where Z_t is a normalization factor (chosen so that D_{t+1} will be a distribution).

Output the final hypothesis:

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

Pros and cons of AdaBoost

Advantages

- Very simple to implement
- Does feature selection resulting in relatively simple classifier
- Fairly good generalization

Disadvantages

- Suboptimal solution
- Sensitive to noisy data and outliers