Ensemble: Boosting

Improving Weak Classifiers: Part I

Boosting

Training data:

$$\{(x^1, \hat{y}^1), \cdots, (x^n, \hat{y}^n), \cdots, (x^N, \hat{y}^N)\}$$

$\hat{y} = \pm 1$ (binary classification)

- Guarantee:
  - If your ML algorithm can produce classifier with error rate smaller than 50% on training data
  - You can obtain 0% error rate classifier after boosting.
- Framework of boosting
  - Obtain the first classifier $f_1(x)$
  - Find another function $f_2(x)$ to help $f_1(x)$
    - However, if $f_2(x)$ is similar to $f_1(x)$, it will not help a lot.
    - We want $f_2(x)$ to be complementary with $f_1(x)$ (How?)
  - Obtain the second classifier $f_2(x)$
  - …… Finally, combining all the classifiers
- The classifiers are learned sequentially.

How to Obtain Different Classifiers

- Training on different training data sets
- How to have different training data sets
  - Re-sampling your training data to form a new set
  - Re-weighting your training data to form a new set
  - In real implementation, you only have to change the cost/objective function

$(x^1, \hat{y}^1, u^1) \quad u^1 = \cancel{1} \quad 0.4$

$(x^2, \hat{y}^2, u^2) \quad u^2 = \cancel{1} \quad 2.1$

$(x^3, \hat{y}^3, u^3) \quad u^3 = \cancel{1} \quad 0.7$

$$L(f) = \sum_n l(f(x^n), \hat{y}^n)$$

$$\Downarrow$$

$$L(f) = \sum_n u^n l(f(x^n), \hat{y}^n)$$

Idea of Adaboost

- Idea: training $\boldsymbol{f_2(x)}$ on the new training set that fails $\boldsymbol{f_1(x)}$
- How to find a new training set that fails $f_1(x)$?

ε_1: the error rate of $f_1(x)$ on its training data

$$\varepsilon_1 = \frac{\sum_n u_1^n \delta(f_1(x^n) \neq \hat{y}^n)}{Z_1} \qquad Z_1 = \sum_n u_1^n \qquad \varepsilon_1 < 0.5$$

Changing the example weights from u_1^n to u_2^n such that

$$\frac{\sum_n u_2^n \delta(f_1(x^n) \neq \hat{y}^n)}{Z_2} = 0.5$$

The performance of f_1 for new weights would be random.

Training $f_2(x)$ based on the new weights u_2^n

Re-Weighting Training Data: Part I

- Idea: **training $\boldsymbol{f_2(x)}$ on the new training set that fails $\boldsymbol{f_1(x)}$**
- How to find a new training set that fails $f_1(x)$?

$(x^1, \hat{y}^1, u^1)$	$u^1 = 1$	$u^1 = 1/\sqrt{3}$
$(x^2, \hat{y}^2, u^2)$	$u^2 = 1$	$u^2 = \sqrt{3}$
$(x^3, \hat{y}^3, u^3)$	$u^3 = 1$	$u^3 = 1/\sqrt{3}$
$(x^4, \hat{y}^4, u^4)$	$u^4 = 1$	$u^4 = 1/\sqrt{3}$

$\varepsilon_1 = 0.25$

$f_1(x)$

0.5

$\varepsilon_2 < 0.5$

$f_2(x)$

Re-Weighting Training Data: Part II

- Idea: training $\boldsymbol{f_2(x)}$ on the new training set that fails $\boldsymbol{f_1(x)}$
- How to find a new training set that fails $f_1(x)$?

$$\begin{cases} \text{If } x^n \text{ misclassified by } f_1 \; (f_1(x^n) \neq \hat{y}^n) \\ \qquad u_2^n \leftarrow u_1^n \text{ multiplying } d_1 \\ \text{If } x^n \text{ correctly classified by } f_1 \; (f_1(x^n) = \hat{y}^n) \\ \qquad u_2^n \leftarrow u_1^n \text{ divided by } d_1 \end{cases}$$

increase

decrease

f_2 will be learned based on example weights u_2^n

What is the value of d_1?

Re-Weighting Training Data: Part III

$$\varepsilon_1 = \frac{\sum_n u_1^n \delta(f_1(x^n) \neq \hat{y}^n)}{\boxed{Z_1}} \qquad Z_1 = \sum_n u_1^n$$

$$\frac{\sum_n u_2^n \delta(f_1(x^n) \neq \hat{y}^n)}{\boxed{Z_2}} = 0.5$$

$f_1(x^n) \neq \hat{y}^n \quad u_2^n \leftarrow u_1^n$ multiplying d_1

$f_1(x^n) = \hat{y}^n \quad u_2^n \leftarrow u_1^n$ divided by d_1

$$= \sum_{f_1(x^n) \neq \hat{y}^n} u_1^n d_1$$

$$= \sum_n u_2^n$$

$$= \sum_{f_1(x^n) \neq \hat{y}^n} u_2^n + \sum_{f_1(x^n) = \hat{y}^n} u_2^n$$

$$= \sum_{f_1(x^n) \neq \hat{y}^n} u_1^n d_1 + \sum_{f_1(x^n) = \hat{y}^n} u_1^n / d_1$$

$$\frac{\sum_{f_1(x^n) \neq \hat{y}^n} u_1^n d_1 + \sum_{f_1(x^n) = \hat{y}^n} u_1^n / d_1}{\sum_{f_1(x^n) \neq \hat{y}^n} u_1^n d_1} = 2$$

Re-Weighting Training Data: Part IV

$$\varepsilon_1 = \frac{\sum_n u_1^n \delta(f_1(x^n) \neq \hat{y}^n)}{Z_1} \qquad Z_1 = \sum_n u_1^n$$

$$\frac{\sum_n u_2^n \delta(f_1(x^n) \neq \hat{y}^n)}{Z_2} = 0.5$$

$f_1(x^n) \neq \hat{y}^n$ $\quad u_2^n \leftarrow u_1^n$ multiplying d_1

$f_1(x^n) = \hat{y}^n$ $\quad u_2^n \leftarrow u_1^n$ divided by d_1

$$\frac{\sum_{f_1(x^n)\neq\hat{y}^n} u_1^n d_1 + \sum_{f_1(x^n)=\hat{y}^n} u_1^n / d_1}{\sum_{f_1(x^n)\neq\hat{y}^n} u_1^n d_1} = 2 \qquad \frac{\sum_{f_1(x^n)=\hat{y}^n} u_1^n / d_1}{\sum_{f_1(x^n)\neq\hat{y}^n} u_1^n d_1} = 1$$

$$\sum_{f_1(x^n)=\hat{y}^n} u_1^n / d_1 = \sum_{f_1(x^n)\neq\hat{y}^n} u_1^n d_1 \qquad \frac{1}{d_1} \underbrace{\sum_{f_1(x^n)=\hat{y}^n} u_1^n}_{Z_1(1-\varepsilon_1)} = d_1 \underbrace{\sum_{f_1(x^n)\neq\hat{y}^n} u_1^n}_{Z_1\varepsilon_1}$$

$$\varepsilon_1 = \frac{\sum_{f_1(x^n)\neq\hat{y}^n} u_1^n}{Z_1} \qquad \sum_{f_1(x^n)\neq\hat{y}^n} u_1^n = Z_1 \varepsilon_1$$

$$Z_1(1-\varepsilon_1)/d_1 = Z_1 \varepsilon_1 d_1$$

$$d_1 = \sqrt{(1-\varepsilon_1)/\varepsilon_1} > 1$$

Algorithm for AdaBoost

- Giving training data $\{(x^1, \hat{y}^1, u_1^1), \cdots, (x^n, \hat{y}^n, u_1^n), \cdots, (x^N, \hat{y}^N, u_1^N)\}$
  - $\hat{y} = \pm 1$ (Binary classification), $u_1^n = 1$ (equal weights)
- For t = 1, …, T:
  - Training weak classifier $f_t(x)$ with weights $\{u_t^1, \cdots, u_t^N\}$
  - ε_t is the error rate of $f_t(x)$ with weights $\{u_t^1, \cdots, u_t^N\}$
  - For n = 1, …, N:
    - If x^n is misclassified by $f_t(x)$: $\hat{y}^n \neq f_t(x^n)$
    - $u_{t+1}^n = u_t^n \times d_t$ $= u_t^n \times \exp(\alpha_t)$
    - Else:
    - $u_{t+1}^n = u_t^n / d_t$ $= u_t^n \times \exp(-\alpha_t)$

$$d_t = \sqrt{(1 - \varepsilon_t)/\varepsilon_t}$$

$$\alpha_t = ln\sqrt{(1 - \varepsilon_t)/\varepsilon_t}$$

$$u_{t+1}^n \leftarrow u_t^n \times exp(-\hat{y}^n f_t(x^n) \alpha_t)$$

Algorithm for AdaBoost, Continued

- We obtain a set of functions: $f_1(x), \ldots, f_t(x), \ldots, f_T(x)$
- How to aggregate them?
  - Uniform weight:
    - $H(x) = sign(\sum_{t=1}^{T} f_t(x))$
  - Non-uniform weight:
    - $H(x) = sign(\sum_{t=1}^{T} \alpha_t f_t(x))$

Smaller error ε_t, larger weight for final voting

$$\alpha_t = ln\sqrt{(1-\varepsilon_t)/\varepsilon_t}$$

$$u_{t+1}^n = u_t^n \times exp(-\hat{y}^n f_t(x^n)\alpha_t)$$

$\varepsilon_t = 0.1$	$\varepsilon_t = 0.4$
$\alpha_t = 1.10$	$\alpha_t = 0.20$