Adaboost for Multilabel Classification

Hongyu Su

Aalto University
February 11, 2013

1 Notations

$$\mathcal{S} = \{(x_1, y_1), \cdots, (x_m, y_m)\} \in \{\mathcal{X}, \mathcal{Y}\}^m$$

is a sequence of training examples where each **instance** x_i belongs to a instance space $\mathcal{X}$ and **label** y_i belongs to a finite label space $\mathcal{Y}$. We first focus on **binary classification** and put constraints on label space $\mathcal{Y} \in \{-1, +1\}$.

We further assumes a **distribution** $\mathcal{D}$ over training examples $\{1, \cdots, m\}$, indices of $\mathcal{S}$. The distribution $\mathcal{D}$ is the weight of the training examples, reveals the importances during training phase. Naturally, we have

$$\sum_{i=1}^{m} D(i) = 1$$

Given training examples $\mathcal{S}$ and distribution $\mathcal{D}$, a **weak**(or **base**) learner computes a **weak**(or **base**) hypothesis h. In general, h has the form $h : \mathcal{X} \to \mathcal{R}$. We interpret the sign of $h(X_i)$ as the predicted label $\{-1, +1\}$ of instance x_i.

2 Adaboost

The idea of **adaboost**[1] is to use weak learner to form a highly accurate prediction rule by calling the weak leaner repeated on different distribution $\mathcal{D}$ of the training examples $\mathcal{S}$.

Let

$$f(x) = \sum_{t=1}^{T} \alpha_t h_t(x),$$

where T is number of iteration, h_t is the tth weak hypothesis. Prediction from adaboost is given by $|f(x)|$.

2.1 Algorithm

A generalised version of Adaboost is shown in Algorithm 1.

Algorithm 1 Generalised version of Adaboost

Require: $\{S = \{(x_1, y_1), \cdots, (x_m, y_m)\}, x_i \in \mathcal{X}, y_i \in \{-1, +1\}\}$
Ensure: Output $H(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$
initialisation: $D_1(i) = \frac{1}{m}, i \in \{1, \cdots, m\}$
for $t = 1$ **to** T **do**
  Train weak leaner using distribution D_t
  Get weak hypothesis h_t: $(X) \to \mathcal{R}$
  Choose $\alpha_t \in \mathcal{R}$ {introduce later}
  Update:

$$Z_t = \sum_{i=1}^{m} D_t(i) \exp(-\alpha_t y_i h_t(x_i))$$

  for $i = 1$ **to** m **do**
    Update:

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

  end for
end for

2.2 Loss Bound

The following **loss bound** holds on training errors of H.

$$\frac{1}{m} |H(x_i) \neq y_i, \forall i| \leq \prod_{t=1}^{T} Z_t$$

Proof
According to update rule in Algorithm 1, we have

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

In addition, we have

$$\begin{aligned} \frac{1}{m} \sum_i |H(x_i) \neq y_i| &\leq \frac{1}{m} \sum_{i=1}^{m} \exp(-y_i f(x_i)) \\ &= \sum_{i=1}^{m} D_{T+1}(i) \left(\prod_t Z_t\right) \\ &= \prod_t Z_t \end{aligned}$$

This tells us that, in order to minimise training error, a reasonable approach is to minimise the bound give above by minimising Z_t in each boosting iteration.

2.3 Choose α_t

Let $u_t(i) = y_t h_t(x_i)$, in tth iteration we have

$$\begin{aligned} Z_t &= \sum_{i=1}^{m} D(i) \exp(-\alpha_t u_t(i)) \\ &\leq \sum_{i=1}^{m} D(i) \left(\frac{1 + u_t(i)}{2} e^{\alpha_t} + \frac{1 - u_t(i)}{2} e^{\alpha_t} \right) \end{aligned}$$

Therefore, a solution that minimizes Z_t is given by

$$\alpha_t = \frac{1}{2} \ln \left(\frac{1 + \sum_{i=1}^{m} D_t(i) u_t(i)}{1 - \sum_{i=1}^{m} D_t(i) u_t(i)} \right)$$

2.4 Compute Weak hypothesis

If we choose

$$\alpha_t = \frac{1}{2} \ln \left(\frac{1 + \sum_{i=1}^{m} D_t(i) u_t(i)}{1 - \sum_{i=1}^{m} D_t(i) u_t(i)} \right),$$

training loss is bounded by

$$\begin{aligned} \frac{1}{m} \sum_{i} |H(x_i) \neq y_i| &\leq \prod_t Z_t \\ &\leq \sqrt{1 - (\sum_{i=1}^{m} D_t(i) u_t(i))^2} \\ &= \sqrt{1 - (\sum_{i=1}^{m} D_t(i) y_i h_t(x_i))^2} \end{aligned}$$

This means, in order to minimise training error, we minimise the error made by weak hypothesis in each boosting iteration.

2.5 Base Learner

Base learner should be able to work on weighted training data (e.g Decision, KNN, Naive Bayes).

2.6 An Example with Decision Tree

3 AdaboostMH

In multilabel classification setting we have training examples

$$\mathcal{S} = \{(x_1, y_1), \cdots, (x_m, y_m)\} \in \{\mathcal{X}, \mathcal{Y}\}^m.$$

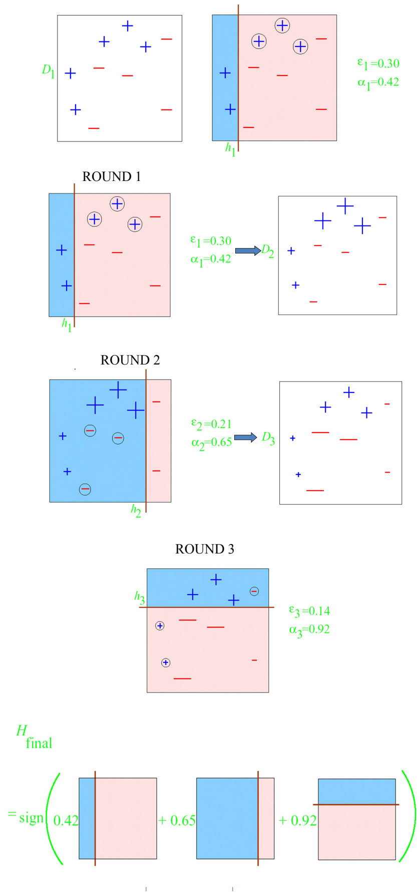

Fig. 1. An Example of Adaboost with decision tree as base learner.

Instead of $\mathcal{Y} \in \{-1, +1\}$, we have $\mathcal{Y} \in \{-1, +1\}^k$.

We define a **distribution** $\mathcal{D}$ over training examples $\{1, \cdots, m\}$ (indices of $\mathcal{X}$) and labels $\{1, \cdots, k\}$ (indices of $\mathcal{Y}$). The distribution $\mathcal{D}$ is the weight of the training examples and labels, reveals the importances during training phase. Naturally, we have

$$\sum_{i=1}^{m} \sum_{l=1}^{k} D(i, l) = 1$$

3.1 Multilabel Hamming Loss

To minimise multilabel hamming loss, one way is to decompose the problem into k orthogonal binary classification problem, which we think $Y \in \{-1, +1\}^k$ as k binary labels defined as $Y[l]$. Multilabel Hamming loss is regarded as the average error rate on k binary problems.

3.2 Algorithm

A generalised version of AdaboostMH is shown in Algorithm 2.

Algorithm 2 Generalised version of Adaboostmh

Require: $S = \{(x_i, y_i), \cdots, (x_m, y_m)\}, x_i \in \mathcal{X}, y_i \in \{-1, +1\}^k$

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

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

for $t = 1$ **to** T **do**

Train weak leaner using distribution D_t

Get weak hypothesis h_t: $\mathcal{X} \times \mathcal{Y} \to \mathcal{R}$

Choose $\alpha_t \in \mathcal{R}$

Update:

$$Z_t = \sum_{i=1}^{m} \sum_{l=1}^{k} D_t(i, l) \exp(-\alpha_t Y_i[l] h_t(x_i, l))$$

for $i = 1$ **to** m **do**

for $l = 1$ **to** k **do**

:

$$D_{t+1}(i, l) = \frac{D_t(i, l) \exp(-\alpha_t Y_i[l] h_t(x_i, l))}{Z_t}$$

end for

end for

end for

3.3 Loss Bound

Multilabel hamming loss for training data is bounded by

$$\mathrm{hloss}(H) \leq \prod_{t=1}^{T} Z_t,$$

where $Z_t = \sum_{i,l} D_t(i,l) \exp(-\alpha_t Y_t[l] h_t(x_i, l))$.

3.4 Choose $\boldsymbol{\alpha}$

α can be chosen by minimizing Z as

$$\alpha_t = \frac{1}{2} \ln \left(\frac{1 + r_t}{1 - r_t} \right),$$

where $r_t = \sum_{i,l} D_t(i,l)(Y_t[l] h_t(x_i, l))$

3.5 Compute Weak Hypothesis

Set α to optimal value, we have

$$Z_t = \sqrt{1 - r_t^2}$$

As a result, the goal of weak learner is to minimize weighted hamming loss with respect to D_t

References

1. Schapire, R.E., Singer, Y.: Improved boosting algorithms using confidence-rated predictions. In: Machine Learning. pp. 80–91 (1999)