# Assignment2 AdaBoost Report

A1655611 Xuanyu Zhao

## 1. Definitions for variables used in this report

$D_t$ — The distribution or set of weights over the training set X.
$D_t(i)$ — The weight of distribution $D_t$ on $i$-th training example $x_i$, on round $t$.
$h_t$ — Weak hypothesis on round t, mapping $X \rightarrow \{-1, +1\}$ based on distribution $D_t$.
$\epsilon_t$ — Error of the weak hypothesis $h_t$.
$\alpha_t$ — The weight of weak hypothesis $h_t$ in the final hypothesis $H(x)$.
$Z_t$ — The normalization factor for distribution $D_{t+1}$.
$H(x)$ — The final hypothesis for the given training set.

## 2. Description of AdaBoost (including my understanding of AdaBoost)

### 2.1. General introduction

AdaBoost is a member of Boosting algorithms. “Ada” is short for adaptive, which means AdaBoost could adapt to the error rates of the individual weak hypotheses.

Boosting is a general and provably effective method of producing a very accurate prediction rule by combining rough and moderately inaccurate rules of thumb. Boosting algorithm has an advantage that it often tends not to overfit. After enough iterations, boosting is able to achieve a strong prediction rule with high accuracy. For example, experiments proved that AdaBoost would sometimes continue to drive down the generalization error long after the training error had reached zero.

In general, AdaBoost is a meta-algorithm which takes a weak learner algorithm (low accuracy) as input, and outputs a strong learner algorithm (high accuracy). Therefore, the 1st step is to find the input weak learner. In this assignment, we use Decision Stump as the input weak learner.

In AdaBoost, the output of the 'weak learners' is combined into a weighted sum that represents the final output of the boosted classifier. The individual learners can be weak, but as long as the performance of each one is slightly better than random guessing, the final model can be proven to converge to a strong learner. Moreover, AdaBoost is adaptive in the sense that subsequent weak learners are more focused on the instances misclassified by previous classifiers. Therefore, AdaBoost is sensitive to noisy data and outliers.

### 2.2. AdaBoost Algorithm

AdaBoost takes a training set $(x_1, y_1), \ldots, (x_m, y_m)$ as input, where each $x_i$ belongs to some domain or instance space $X$, and each label $y_i$ is in label set $Y$. In this report we assume $Y=\{-1,$

+1}, although AdaBoost also works for multi-class classification. AdaBoost calls a given weak or base learning algorithm repeatedly in a series of rounds $t = 1, \ldots, T$. On each round AdaBoost will maintain a distribution or set of weights $D_t$ over the training set. The weight of this distribution on training example $i$ on rount $t$ is denoted $D_t(i)$. On each round, training examples with big weights are more important. This means the weak learner will pay more attention to these training examples than others. However, the weights of training examples are dynamically updated after each round.

Initially, weights for every instance are set equally. Then on each round, the weights of incorrectly classified instances are increased so that the weak learner is forced to focus on the hard examples in the training set.

On each round, the weak learner will find a weak hypothesis $h_t$, mapping $X \rightarrow \{-1, +1\}$ based on weight distribution $D_t$. We measure the goodness of $h_t$ by its error

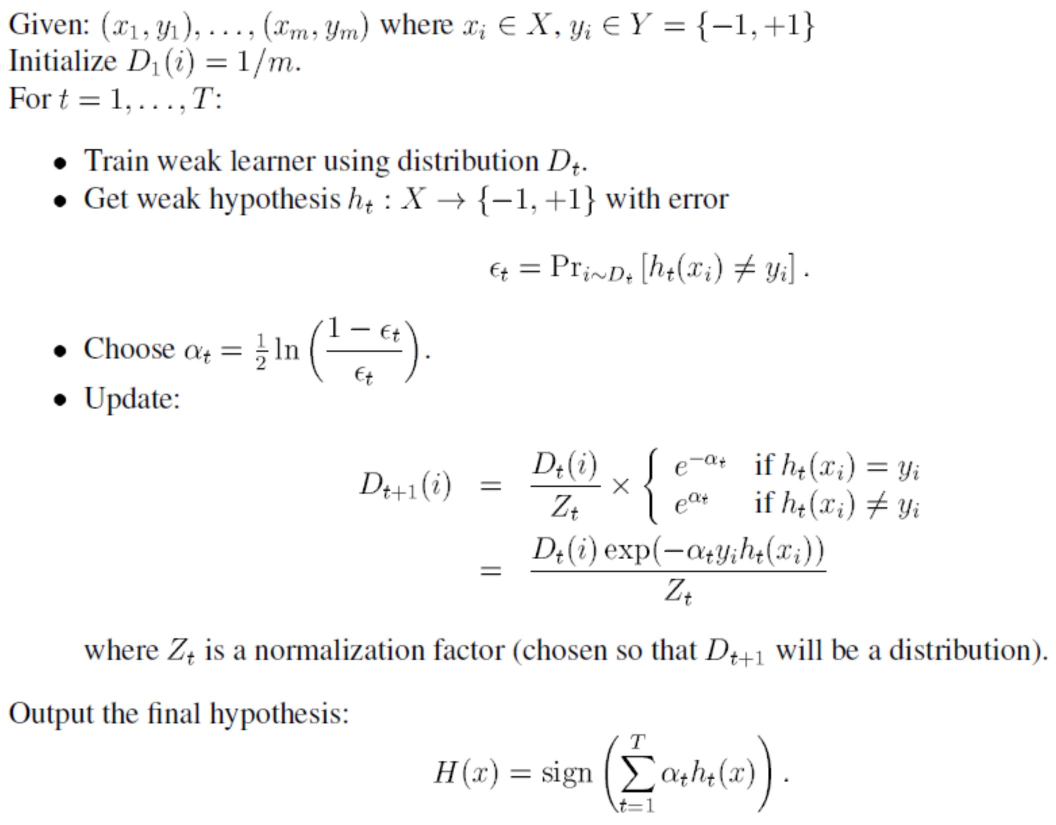
$\epsilon_t = \Pr_{i \sim D_t} [h_t(x_i) \neq y_i] = \sum_{i:h_t(x_i) \neq y_i} D_t(i)$

The error is measured with respect to the distribution $D_t$.

Once $h_t$ is obtained, AdaBoost uses a parameter $\alpha_t = \frac{1}{2}\ln(\frac{1-\epsilon_t}{\epsilon_t})$ for $h_t$. $\alpha_t$ measures the importance that is assigned to $h_t$. In fact, $\alpha_t$ is the weight assigned to $h_t$ in the final hypothesis $H$, which is a weighted majority vote of weak hypotheses $h_t$, $t = 1, \ldots, T$. From the definition we see that $\alpha_t$ gets larger as $\epsilon_t$ gets smaller. If $h_t$ has smaller error, it will have bigger influence in the final $H$ .

After calculating $\alpha_t$, the distribution $D_t$ on instances is updated for next round according to the formula in below diagram. The effect of this rule is to increase the weight of examples misclassified, and to decrease the weight of correctly classified examples. Thus, on next round, the weight tends to concentrate on “hard” examples (of this round). The training process of next round will pay more attention to these “hard” examples.

This process is repeated. After $T$ iterations, AdaBoost outputs the final strong hypothesis $H$ as a weighted majority vote of the $T$ weak hypotheses.



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 weak learner using distribution $D_t$.
- Get weak hypothesis $h_t : X \to \{-1, +1\}$ with error

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

- Choose $\alpha_t = \frac{1}{2}\ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$.
- Update:

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

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

Output the final hypothesis:

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

Figure 1: The boosting algorithm AdaBoost.

Freund, Yoav, Robert Schapire, and N. Abe. "A short introduction to boosting." Journal-Japanese Society For Artificial Intelligence 14.771-780 (1999): 1612.

## 3. Analysis of my implementation

In my implementation, I wrote 2 matlab files. MLassign2.m and testSVM.m. Firstly we run MLassign2.m, which implement AdaBoost and will get a strong hypothesis. User could input the number of iterations they want.

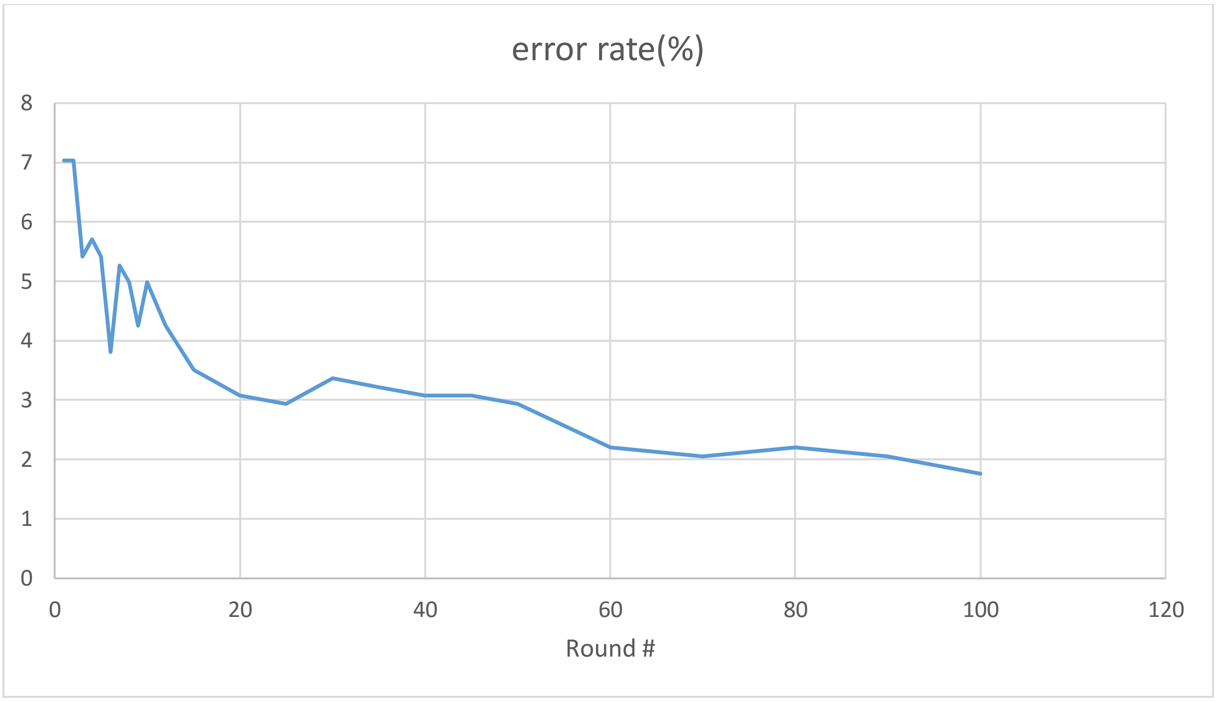
Secondly we run testSVM.m, inside which I train both primal and dual SVMs. I applied these classifiers on breast-cancer.mat and compared the results of AdaBoost hypothesis and trained SVMs.

### 3.1. Statistics of AdaBoost Classifier

T: number of iterations

| T | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 12 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Error rate | 7.03% | 7.03% | 5.42 % | 5.71 % | 5.42 % | 3.81 % | 5.27 % | 4.98 % | 4.25 % | 4.98 % | 4.25 % | 3.51 |
| T | 20 | 25 | 30 | 35 | 40 | 45 | 50 | 60 | 70 | 80 | 90 | 100 |
| Error rate | 3.07% | 2.93% | 3.37 % | 3.22 % | 3.07 % | 3.07 % | 2.93 % | 2.2 % | 2.05 % | 2.2 % | 2.05 % | 1.76 % |

Training error curve against the number of iterations



## 3.2. Statistics of Trained SVMs

Error rate of SVMs on same training set

| C value | 0.1 | 1.1 | 10.1 |
|---|---|---|---|
| Error rate of primal SVM | 2.64% | 2.93% | 2.78% |
| Error rate of dual SVM | 2.78% | 2.93% | 2.78% |

## 3.3. Analysis

From training error curve we observe that AdaBoost continuously decrease the error rate, although there is some fluctuation in early stage (T<50). This proves that AdaBoost is an effective way to combine weak classifiers to create a strong classifier.

The average error rate of trained SVMs is around 2.7%. In AdaBoost, when iteration number is greater than 60, the error rate will be smaller than 2.7%. Therefore, when users run AdaBoost with sufficiently many iterations, AdaBoost may perform better than SVM.

## 4. References

[1] Freund, Yoav, Robert Schapire, and N. Abe. "A short introduction to boosting." Journal-Japanese Society For Artificial Intelligence 14.771-780 (1999): 1612.

[2] Freund, Yoav, and Robert E Schapire. 'A Decision-Theoretic Generalization Of On-Line Learning And An Application To Boosting'. Journal of Computer and System Sciences 55.1 (1997): 119-139.