Abstract: Boosting algorithm are greedy methods for forming linear combinations of base hypotheses. The algorithm maintains a distribution on training examples, and this distribution is updated according to a step or the combination coefficient of base hypothesis. The main difference of some AdaBoost algorithms is the different updating step chosen. In this project we give some new steps for AdaBoost algorithms after introducing a new up-bound function of the potential and minimizing it. Some experimental results show that the new steps work well comparing with the original steps and always can achieve a larger margin. Especially for a larger training problem they overtop much.

Keywords: AdaBoost algorithm, Updating step, Potential, Weaker hypothesis

1. Introduce the AdaBoost Algorithms

Boosting algorithm are greedy methods for forming linear combinations of base hypotheses. The input of the algorithm is a set of labeled training examples. In each iteration the algorithm maintains a distribution on these examples, and this distribution is updated after getting a base hypothesis given by some weaker learner. The boosting algorithm increases the weights of those examples that are wrongly classified by the base hypothesis. The output of the algorithm is a decision function that is the linear combination of those base hypotheses. For any input example, the sign of this function gives its prediction.

In the all boosting algorithms, AdaBoost is the most famous one that is proposed by Freund and Schapire\textsuperscript{[1]}. It maintains the linear coefficient of the base hypothesis “adaptively”. There are two important generalization algorithms of the AdaBoost. One is proposed in [2] called AdaBoost\textsubscript{ρ} and the other is AdaBoost\textsubscript{μ} proposed in [3]. The difference among them is that their linear combination coefficient or the step is calculated by different method.
The AdaBoost algorithm is listed as following [1-3]:

**Input:** \( S = \{ x_n, y_n \}_{n=1}^N \).

**Output:** The hypothesis \( h_t(x) \) and their coefficients \( \alpha_t \), \( t = 1, 2, \cdots, T \). Then the decision function is \( I(x) = \text{sign}\left( \sum_{t=1}^T \alpha_t h_t(x) \right) \).

**Initialization:** Given the No. of iterations \( T \). Set \( \mathbf{d}^0_n = 1/N \) for all \( n = 1, 2, \cdots, N \).

**Do for** \( t = 1, 2, \cdots, T \):

- **Step 1:** Train classifier on \( \{ S, d^t \} \) and obtain hypothesis \( h_t : x \rightarrow [-1, 1] \);
- **Step 2:** Calculate the edge of \( h_t(x) \):
  \( \gamma_t = \sum_{n=1}^N d^t_n y_n h_t(x_n) \);
- **Step 3:** If \( \gamma_t = 1 \), then the algorithm return with \( \alpha_t = \text{sign}(\gamma_t) \), \( h_t = h_t \), \( T = 1 \).
- **Step 4:** Otherwise set
  \( \alpha_t = \frac{1}{2} \ln \frac{1 + \gamma_t}{1 - \gamma_t} \);
- **Step 5:** Update \( d^{t+1}_n = d^t_n e^{-\alpha_t y_n h_t(x_n)}/Z_t \) for \( i = 1, \cdots, N \) with \( Z_t = \sum_{n=1}^N d^t_n e^{-\alpha_t y_n h_t(x_n)} \).

Let \( Z_t(\alpha) = \sum_{n=1}^N d^t_n e^{-\alpha y_n} \) and \( d^t_n(\alpha) = d^t_n e^{-\alpha y_n}/Z_t(\alpha) \), where \( u_n^t = y_n h_t(x_n) \). In AdaBoost [1] algorithm, the step \( \alpha_t \) for updating the distribution \( \mathbf{d}^t \) is gotten by keeping \( \mathbf{d}^t(\alpha) \cdot \mathbf{u}^t = 0 \) or keeping the next edge at least 0, because of the convexity of \( Z_t(\alpha) \) with respect to \( \alpha \), which is equivalent to

\[
\min_{\alpha} Z_t(\alpha) = \sum_{n=1}^N d^{t-1}_n e^{-\alpha u_n^t} \quad \text{or} \quad Z_t'(\alpha) = 0.
\] (1)

Although (1) can be solved exactly by any line-search method because \( Z_t(\alpha) \) is convex that clear is proved in [4], for the convenience of analysis, a close form of approximate solution of (1) is obtained by \( \min_{\alpha} \hat{Z}_t(\alpha) \) where

\[
Z_t(\alpha) \leq \hat{Z}_t(\alpha) = \left( \frac{1}{2} (1 + \mathbf{d}^{t-1} \cdot \mathbf{u}^t) e^{-\alpha} + \frac{1}{2} (1 - \mathbf{d}^{t-1} \cdot \mathbf{u}^t) e^{\alpha} \right).
\]

The inequality is because

\[
e^{-\alpha x} \leq \frac{1}{2} (1 + x) e^{-\alpha} + \frac{1}{2} (1 - x) e^{\alpha}, \quad x \in [-1, 1].
\] (2)

which is also important to prove the bounds of AdaBoost algorithm.

The solution of \( \min_{\alpha} \hat{Z}_t(\alpha) \) can be analytically obtained as

\[
\alpha_t = \frac{1}{2} \ln \frac{1 + \gamma_t}{1 - \gamma_t}, \quad \text{where} \quad \gamma_t = \mathbf{d}^{t-1} \cdot \mathbf{u}^t
\] (3)

In [2], in order to obtain a higher progress, a tighter condition \( \mathbf{d}'(w) \cdot \mathbf{u}' = \rho \) is maintained per iteration, which is equivalent to

\[
\min_{\alpha} Z_t(\alpha) = \sum_{n=1}^N d^{t-1}_n e^{-\alpha(u_n^t - \rho)}.
\] (4)
Similarly, by inequality (2), \[ \min_{\alpha} \tilde{Z}_t(\alpha) := e^{\alpha_0} \left( \frac{1}{2} \left( 1 + d_{t+1} \cdot u_t \right) e^{\alpha} + \frac{1}{2} \left( 1 - d_{t+1} \cdot u_t \right) e^{-\alpha} \right) \] is instead to get closed form of step \( \alpha_t \) and a new algorithm AdaBoost\( \rho \) is proposed.

\[
\alpha_t = \frac{1}{2} \ln \frac{1 + y_t}{1 - y_t} - \frac{1}{2} \ln \frac{1 + \rho}{1 - \rho}.
\]  

(5)

Here \( \rho \) is margin target. If \( \rho \) is chosen properly (slightly blow the maximum margin \( \rho^* \)), AdaBoost\( \rho \) will converge exponentially fast to a combined hypothesis with nearly the maximum margin. Unfortunately the maximum is unseen in advance. In [2] they presented a algorithm to constructs a sequence \( \{\rho_i\}_{i=1}^\beta \) converging to maximum margin \( \rho^* \). It needs to run AdaBoost\( \rho \) several times with binary search scheme.

In [3], another new algorithm called AdaBoost\( \nu \) is given with a new step \( \alpha_t \) as following

\[
\alpha_t = \frac{1}{2} \ln \frac{1 + y_t}{1 - y_t} - \frac{1}{2} \ln \frac{1 + \rho_i}{1 - \rho_i}.
\]  

(6)

where \( \rho_i = \min_{i \leq t} \gamma_i - \nu \) and \( \nu \) is a precision parameter. This new one-pass algorithm finds a non-negative linear combination with margin at least \( \rho^* - \nu \) within \( \frac{2 \ln N}{\nu^2} \) iterations.

2. The main results

In this project, we use a more tight inequality to find an approximate analytic solutions of problems (1) and (4).

A tight up-bound function of \( e^{-\alpha x} \) is given by the following inequality

\[
e^{-\alpha x} \leq \begin{cases} 
1 + x(1 - e^\alpha), & x \in [-1, 0). \\
1 - x(1 - e^\alpha), & x \in [0, 1]
\end{cases}
\]  

(7)

The difference between inequality (2) and (7) is shown in Fig 1.

2.1 New step for AdaBoost method

Firstly, we consider problem (1). By inequality (7), \( Z_t(\alpha) \) in (1) is magnified as the following \( \tilde{Z}_t(\alpha) \)

\[
\tilde{Z}_t(\alpha) := \sum_{u_t \leq 0} d_{i \leq n}^{-1} \left( 1 + u_t(1 - e^\alpha) \right) + \sum_{u_t > 0} d_{i \leq n}^{-1} \left( 1 - u_t(1 - e^\alpha) \right). \]

\[
= 1 + \gamma_t^-(1 - e^\alpha) - \gamma_t^+(1 - e^\alpha)
\]

where \( \gamma_t^- = \sum_{u_t \leq 0} d_{i \leq n}^{-1} u_t \), \( \gamma_t^+ = \sum_{u_t > 0} d_{i \leq n}^{-1} u_t \). Compared inequality (2) and (7), we have \( Z_t(\alpha) \leq \tilde{Z}_t(\alpha) \leq \tilde{Z}_t(\alpha) \).

Typically \( \gamma_t^- < 0 \), then minimizing \( \tilde{Z}_t(\alpha) \) by solving \( \tilde{Z}_t'(\alpha) = 0 \), and we can get a
simple step

\[ \bar{\alpha}_i = \frac{1}{2} \ln \frac{\gamma_i^+}{-\gamma_i^-}. \] (8)

with \[ \bar{Z}_i(\bar{\alpha}) = \left(1 - \left(-\sqrt{-\gamma_i^-} + \sqrt{\gamma_i^+}\right)^2\right). \]

If \( \gamma_i^- = 0 \) (this means \( u_n' \geq 0 \) for all \( n \) and \( \mathbf{d}^{-1}(\alpha) \cdot \mathbf{u}' \geq 0 \) has already been reached), then \( \bar{Z}_i(\alpha) \) is monotone decreased and its minimizer is reached when \( \alpha = +\infty \). But for keeping the iterations consistent, we cap the step as \( \alpha = 1 \).

In [4], Schapire and Singer proposed similar equation but the hypothesis is limited with range \{−1, 0, 1\}. And a smoothing technique is also discussed in [4].

**2.2 New steps for AdaBoost* method**

Similar as [2, 3], in order to maintain \( \mathbf{d}'(w) \cdot (\mathbf{u}' - \rho) = 0 \) or solve the problem in Eq. (4) approximately, by inequality (7), we magnify \( Z_i(\alpha) \) in Eq. (4) to \( \bar{Z}_i(\alpha) \) as following

\[ Z_i(\alpha) \leq \bar{Z}_i(\alpha) := \left(1 + \gamma_i^- (1 - e^\alpha) - \gamma_i^+ (1 - e^{-\alpha})\right)e^{\alpha} \]

To Minimize \( \bar{Z}_i(\alpha) \), we have

\[ \bar{Z}_i'(\alpha) = \left(b + a e^\alpha + c e^{-\alpha}\right)e^{\alpha} = 0, \] (9)

where \( a = -\gamma_i^- (\rho + 1), \quad b = \rho(1 + \gamma_i^- - \gamma_i^+) \) and \( c = -(1 - \rho)\gamma_i^+ \).

If \( \gamma_i^- < 0 \), we can get a complicated step by solving the equation (9):
\[ \alpha_i = \ln \frac{-b + \sqrt{b^2 - 4ac}}{2a}. \]  

(10)

And if \( \gamma_i = 0 \) or \( a = 0 \) (then \( \gamma^*_i = \gamma_i \)), we have

\[ \alpha_i = \ln \frac{\gamma_i}{1 - \gamma_i} - \ln \frac{\rho}{1 - \rho}. \]

(11)

After omitting \( b \) in Eq. (9), we can get a simple step similar like that in [2, 3] as follows more approximately than (10)(if \( \gamma_i = 0 \), (11) is used too):

\[ \hat{\alpha}_i = \frac{1}{2} \ln \frac{\gamma^*_i}{-\gamma_i} - \frac{1}{2} \ln \frac{1 + \rho}{1 - \rho}, \quad \text{or} \quad \hat{\alpha}_i = \frac{1}{2} \ln \frac{\gamma^*_i}{-\gamma_i} - \frac{1}{2} \ln \frac{1 + \rho}{1 - \rho}. \]

(12)

We can prove \( 0 < \alpha_i \leq \hat{\alpha}_i \) and \( \hat{Z}_i(\alpha_i) \leq \hat{Z}_i(\hat{\alpha}_i) \) because \( b = 1 - d^{-1} \cdot |u| \) in Eq (9) is positive and \( \hat{Z}_i(0) = b + a + c = \rho - d^{-1} \cdot u \) is negative thus 0 and \( \hat{\alpha}_i \) are just on the different side of the \( \alpha_i \).

3. Experimental results

In this section, we give some experiments to illustrate the different effect of the steps involved in this project. The optimal solution of (1) or (4) \( \alpha^* \), called as **Optimal Step** in our experiments, is also obtained as a comparator in our experiments, where Newton method is used to solve (1) or (4):

Updating \( \alpha_{k+1} := \alpha_k + Z'(\alpha_k)/Z^*(\alpha_k) \)

until \( Z'(\alpha_{k+1}) \leq \varepsilon \) is reached for tinny number \( \varepsilon \).  

(13)

In our experiments all the steps are capped into interval \([0, 1]\) to keep the iteration consistent.

3.1 Experiment on email spam data with the hypothesis with rage \([-1, 1]\)

The fist experiment is on email spam dataset by AdaBoost algorithms to compare the step (3) and step (8), where while \( \gamma_i = 0 \), the (3) is used to take place (8). The data is permuted 100 times and is split into 3/4 for training and 1/4 for testing set. The average results are plotted in Fig 2. We select the feature as the weaker learners, where the component of all the instances only is either \(-1\) or \(+1\). In order to get a little bit better hypothesis in every trial, we randomly chose 30 feature and selected the best one as the hypothesis to the algorithm in stage \( t \).

The results in Fig. 2 show that new step can get a result the same better as the original one. This is not strange because at this situation \( h^i(x_s) \in \{-1, 1\} \), all the inequalities (2) and (7) are all tight then \( \hat{Z}_i(\alpha) = \hat{Z}_i(\alpha) \). The steps used should be the same one all the time. The small difference is because the hypothesis chosen per trial is totally random.
3.2 Experiment on artificial data

In order to see the difference between step $\alpha_t$ in (8) and $\alpha_t$ in (3) or the difference between step $\alpha_t$ in (10) and $\alpha_t$ in (6), we have to find a problem with hypothesis $h(x_n) \in [-1,1]$.

Similarly as in [3], we generate an artificial data with 100 dimensional and containing 98 nuisance dimensions with uniform noise. The other two dimensions are plot exemplary in Figure 3.

---

**Fig. 2** The compare between NEW and Original steps on email spam dataset.

![Fig. 2](image.png)

**3.2 Experiment on artificial data**

In order to see the difference between step $\alpha_t$ in (8) and $\alpha_t$ in (3) or the difference between step $\alpha_t$ in (10) and $\alpha_t$ in (6), we have to find a problem with hypothesis $h(x_n) \in [-1,1]$.

Similarly as in [3], we generate an artificial data with 100 dimensional and containing 98 nuisance dimensions with uniform noise. The other two dimensions are plot exemplary in Figure 3.

![Figure 3](image.png)

**Figure 3.** The two discriminative dimensions of our separable 100 dimensional data.
In the experiments, 20000 data are generated and all the results shown are averaged over 200 splits into $m$ training and 20000-$m$ test examples, where $m$ varies from 50 to 500.

This is a linear classification problem, and the hypothesis should be $h(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x}$. We prune this linear function as (14) to keep its domain into $[-1,1]$.

$$h_w(\mathbf{x}) = \begin{cases} 
-1 & \mathbf{w} \cdot \mathbf{x} < -1 \\
\mathbf{w} \cdot \mathbf{x}, \mathbf{w} \cdot \mathbf{x} \in [-1,1], \text{ with } \|\mathbf{w}\| = 1 \\
1 & \mathbf{w} \cdot \mathbf{x} > -1 
\end{cases}$$  \hspace{1cm} (14)

A very weak learner is chosen to obtain the weak hypothesis $h_w$ according to the following three steps:

1) Randomly choose $K$ pairs points $(\mathbf{x}_i^1, \mathbf{x}_i^2)$ form the training data according to the distribution $\mathbf{d}^{t-1}$;

2) $K$ weight vectors are generated as $\mathbf{w}_i = \mathbf{x}_i^1 - \mathbf{x}_i^2$, $i = 1, 2, \cdots, K$;

3) $h_w = \arg \max_{h_{w_i}} \sum_{n=1}^{N} h_{w_i}(\mathbf{x}_n)$.

In my experiments, $K = 10$.

We implement AdaBoost algorithms with updating step $\tilde{\alpha}_t$ in (8), $\alpha_t$ in (3) and optimal step gotten by procedure (13) on this dataset respectively, and also have accomplished AdaBoost$_\rho$ with updating step in Eq. (6), (10) and (12).

At first data, we run AdaBoost methods with three different steps above and set $T = 300$ and $T = 1000$ separately. The test accuracies are listed in Table 1 (and the training accuracies are 100% always and not given here).

It shows that differences between them are tiny, and original AdaBoost always has a little better than others and the new step gets the second rank. The differences of the step are the margin they achieve and we will compare them lately.

### Table 1. The experimental testing accuracies corresponding different steps and training sizes.

<table>
<thead>
<tr>
<th></th>
<th>$m=50$</th>
<th>$m=100$</th>
<th>$m=200$</th>
<th>$m=500$</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>T = 300</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Original step</td>
<td>98.49±0.27</td>
<td>98.97±0.14</td>
<td>99.24±0.10</td>
<td>99.46±0.07</td>
</tr>
<tr>
<td>New step</td>
<td>98.35±0.33</td>
<td>98.78±0.19</td>
<td>99.06±0.13</td>
<td>99.32±0.10</td>
</tr>
<tr>
<td>Optimal step</td>
<td>98.33±0.34</td>
<td>98.72±0.21</td>
<td>98.99±0.15</td>
<td>99.23±0.11</td>
</tr>
<tr>
<td><strong>T = 1000</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Original step</td>
<td>98.34±0.30</td>
<td>98.83±0.18</td>
<td>99.13±0.12</td>
<td>99.42±0.08</td>
</tr>
<tr>
<td>New step</td>
<td>98.35±0.35</td>
<td>98.76±0.21</td>
<td>99.05±0.15</td>
<td>99.27±0.09</td>
</tr>
<tr>
<td>Optimal step</td>
<td>98.34±0.35</td>
<td>98.77±0.22</td>
<td>99.04±0.16</td>
<td>99.25±0.10</td>
</tr>
</tbody>
</table>

In Table 1, it observes that a long iteration has not any advantage to improve the test accuracy, so we can not do any more analysis for $T = 1000$. The later experiments are obtained when $T = 300$ to show the margin the different methods achieved.
In Figure 4, the steps of different schemes with AdaBoost algorithms are plotted. It illustrates that the original AdaBoost is often smaller than the new step and the optimal step. It shows that the new step is closer to optimal than original one.

Figure 4. The mean steps of different schemes with AdaBoost Algorithm. The value is averaged over 200 split trials. It shows that the new step is closer to optimal than original one.

In Figure 5, the steps of different schemes with AdaBoost algorithms are plotted. It illustrates that the original AdaBoost is often small, the new step in Eq. (10) in the middle and approximate one is always keep a large step.

Figure 5. The mean steps of different schemes with AdaBoost algorithms. The value is averaged over 200 split trials. It shows that the new step is closer to optimal than original one.

In Figure 6, we plot the averaged margin gotten by all the schemes for the same dataset.
The dotted lines are gotten by AdaBoost and the solid lines are obtained by AdaBoost $\rho^*$. Firstly it shows AdaBoost $\rho^*$ can achieve a better margin than AdaBoost, because the solid lines are always higher than the dotted lines. The only except one is on the last plot, but it should get a higher result after some more iterations (not show here). Secondly, it also shows that original AdaBoost and AdaBoost $\rho^*$ work well for small training problem with large margin ($m=50$). They are reached the largest margin respectively in its kind of algorithms. But for others larger training problem (should be with a smaller margin), our new steps (Eq. (8), Eq. (10) and (12)) works better than others. At least our new schemes are converged fast than others.

In Figure 6, the averaged margin gotten by all the schemes, including obtained by AdaBoost and AdaBoost $\rho^*$. It shows AdaBoost $\rho^*$ can achieve a better margin than AdaBoost, and the original AdaBoost and AdaBoost $\rho^*$ work well for small training problem with large margin ($m=50$), but our new schemes works better than other for a larger training problem (should be with a smaller margin).

In Figure 7, the resulting margins of 200 split problems are plotted respectively, where the problems are sorted by the margin of original AdaBoost algorithm gotten. It shows that for the same problem, the margin of different scheme achieved is distinct. But it is clearly that our new step often achieved a larger margin than the optimal scheme, and again shows that original AdaBoost method has advantages of dealing with small training problem than other two methods. But for larger training problem, our new steps are best scheme in all methods.

In Figure 8, the experimental results of the original AdaBoost $\rho^*$, our new step and its approximate one are plotted. It again shows that for smaller size problem with a bigger
margin, the original one is best. But for larger size problem, our new step and its approximate one can obtain a better result.

Figure 7. The margin obtained after 300 iterations for 200 split problems from the same data. The problems are sorted according to the margin obtained by original AdaBoost algorithm. It is apparent that the new step is always better than optimal step, and for large training problem it is also better than the original step of AdaBoost.

Figure 8. The margin obtained after 300 iterations for 200 split problems from the same data. The problems are sorted according to the margin obtained by original AdaBoost algorithm. It is apparent that the new step and its approximation always achieve better results for large training problem.
The experiments in this section partly prove that the proposed steps have some advantages over the original AdaBoost and AdaBoost\(_\rho\)\(^*\). But we have not proven the bounds of those algorithms with the new steps. We will work further to obtain them. We also plan to use the method given in [4] to compare all the steps with the complete corrective scheme.

4. Conclusion
Some new steps for updating the distribution in AdaBoost algorithms are discussed in this project. Based on a more tight inequality, we give some new steps to achieve a better margin. Some experimental results show that the new steps have some advantages over the original ones. Of course, there are some aspects have not finished yet, such the upper bounds of the iterations, the experimental results on some practical dataset, especially on some nonlinear classification problems and soft margin problems. We will work on it gradually.

5. Reference