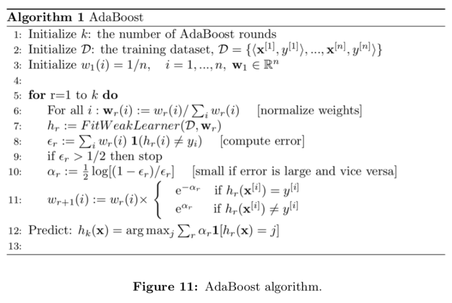
How Weak is a Weak Learner?

Yewon Lee
ylee578@wisc.edu

George Liu
gliu84@wisc.edu

Neil Leonard
nleonard5@wisc.edu



Algorithm 1 AdaBoost

```
Initialize $k$: the number of AdaBoost rounds
Initialize $\mathcal{D}$: the training dataset, $\mathcal{D} = \{\langle \mathbf{x}^{[1]}, y^{[1]} \rangle, ..., \mathbf{x}^{[n]}, y^{[n]} \rangle\}$
Initialize $w_1(i) = 1/n, \quad i = 1, ..., n, \ \mathbf{w}_1 \in \mathbb{R}^n$

for r=1 to $k$ do
    For all $i : \mathbf{w}_r(i) := w_r(i) / \sum_i w_r(i)$    [normalize weights]
    $h_r := FitWeakLearner(\mathcal{D}, \mathbf{w}_r)$
    $\epsilon_r := \sum_i w_r(i)\, \mathbf{1}(h_r(i) \neq y_i)$    [compute error]
    if $\epsilon_r > 1/2$ then stop
    $\alpha_r := \frac{1}{2} \log[(1-\epsilon_r)/\epsilon_r]$    [small if error is large and vice versa]
    $w_{r+1}(i) := w_r(i) \times \begin{cases} e^{-\alpha_r} & \text{if } h_r(\mathbf{x}^{[i]}) = y^{[i]} \\ e^{\alpha_r} & \text{if } h_r(\mathbf{x}^{[i]}) \neq y^{[i]} \end{cases}$
Predict: $h_k(\mathbf{x}) = \arg\max_j \sum_r \alpha_r \mathbf{1}[h_r(\mathbf{x}) = j]$
```

Figure 11: AdaBoost algorithm.

Figure 1. The implementation of ADA boosting we will be using in this project [1]

1. Introduction

Machine learning is an exciting field of research, partly due to the wealth of techniques and plethora of questions to be answered. When navigating such an open field, to make pertinent and interesting contributions we are prioritizing conciseness. As such, we will be asking ourselves: how "weak" is a "weak learner". Using the ADA boosting algorithm, we will be tuning hyper parameters of the algorithm to find a the weakness threshold at which algorithm converges on a meaningful classification algorithm. For our weak learner, we will be using a fully-connected CNN against the MNIST data set. These are standard resources used in the field, which will help us create clear statements about our finding, as we will not get lost in the implementation of a more exotic technique or data set.

2. Motivation

When using ADA boosting, a set of "weak learners" is chained together to make a more successful algorithm. While the rest of the algorithm is clearly defined, the definition of what a "weak learner" is vague and up to user. In our project we hope to quantify this idea by ranging our hyper parameters of the "weak learner" and observing how these values affect the classification power of the overall algorithm. By doing this, we should be able to see at what thresholds these hyper parameters create a successful algorithm.

Figure 2. Examples from the MNIST dataset [2]

3. Evaluation

There are two important evaluation criteria to be responsible for: the error of each "weak learner" and the strength of the overall algorithm. For the individual "weak learners" we will use a traditional classification error as prescribed by the ADA algorithm. As for the overall effectiveness of the algorithm, this is a more subtle question. Due to the nature of question we are looking to answer, we are interested the full range that the overall classification accuracy of algorithm takes. In particular, we are interested in how changing the tuning parameters turns a "weak" algorithm (one with above random guessing but not completely accurate) to a "strong" algorithm (one with a high classification accuracy). A successful implementation of the answering our question will show a smooth transition from a "weak" to a "strong" algorithm. The hyper parameters used to demonstrate this transition will the training size of the CNNs and the number of "weak learners" used in the ADA implementation (i.e the k value from the figure above)

4. Resources

As described above, we will implanting a fully-connected Convolutional Neural Network and training it with the MNIST data set. Due to the relative simplicity of this implementation, there is many options to take. Depending on the difficulty of implementation, we will either be using the standard machine learning package PyTorch, or the open source code from Michael Nielsen's web book "Neural Networks and Deep Learning" [3]. The latter is attractive in that we will have access to every line of code, making it easier to modify for our specifications. The simplicity of our algorithm may make it possible to not use a more powerful library.

5. Contributions

The following are the main tasks in this project:

1. Pre-processing of the MNIST dataset
2. Post-processing of resultant data (collecting, graphing, ect)
3. Model selecting (hyper parameter setting)
4. Model building
5. Model interpretation

With the first author responsible for item 1, the second author responsible for item 2, and the third author reponsible for item 4. The other items will be handled by all the authors.

References

[1]V. Mirjalili and S. Raschka. Python Machine Learning 2nd Edition
[2] Mnist dataset: https://en.wikipedia.org/wiki/MNIST_database
[3] M. Nielsen. http://neuralnetworksanddeeplearning.com/