AdaBoost Regression with Single Layer Neural Networks

Closed AI: Daniel Fiume, David Chen, Tian Yu, Zhucheng Li

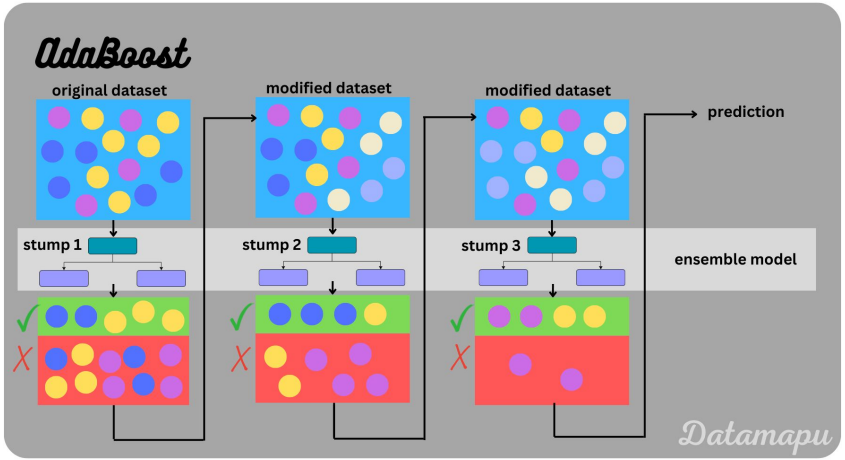
Overview

Boosting is an ensemble learning method where many ‘weak’ learners are combined to collectively make a decision for each data point.

Components:

- Data weights - value from 0 to 1 for each data point
- Learner weights - value from 0 to 1 for each learner

Adaptive Boosting (AdaBoost) dynamically adjusts both the data weights and learner weights iteratively.



Overview

We train each weak learner separately, and **each learner uses the ensemble's previous mistakes to focus on different parts of the data**

- If the ensemble's loss for data point *x* was high, *x*'s weight will be higher for the next learner
- A higher data weight means it's more important for a learner to be correct for that data point

We use each learner's loss to determine its learner weight.

- Lower learner loss ➡ higher weight: its decision is more important to the ensemble

Representation:

$$E(H, T) = \{x \to \Sigma(w_t \cdot h_t(x)) : w \in R^T, \forall t, h_t \in H\}$$

- **H**: the class of base, the weak learner.
- **E**: the ensemble of weak learners.
- **h_t(x)**: Prediction result from the t-th weak learner.
- **w_t**: Weight of the t-th learner.
- **T**: Total number of weak learners.

Regression: MSE Loss

$$L_S(E(H,T)) = \frac{1}{m}\sum_{i=1}^{m}(y_i - E(H,T)(\mathbf{x}_i))^2$$

- **L_s(E(H,T):** Loss function, measuring the error between prediction and true value
- **m**: total number of data points
- **x_i**: i_th samples' input feature vectors
- **y_i**: i_th samples' target values
- **E(H,T)(x_i)**: The predicted value obtained from the weak learner on x_i

Ada Boost Optimizer / Pseudocode

- Start with uniform weights for all training samples.
- Train a weak model (weak learner) on the weighted training data (based on result from previous iteration).
- Calculate error
- Compute weight of learner and data sample weights:
  - Increase weights for incorrect samples, decrease for correct ones.
- Aggregate weak learners, weighted appropriately, to construct the final strong hypothesis..

More Explicitly:

Input:

- Training set $S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\}$
- Weak learner Wl
- Number of estimators T

Each iteration focuses on different parts of the data based on data weights D_i

Initialize: $D^{(1)} = \left(\frac{1}{m}, \ldots, \frac{1}{m}\right)$

For $t = 1, \ldots, T$:

$h_t = WL(D^{(t)}, S)$

$\epsilon_t = \sum_{i=1}^{m} D_i^{(t)} (y_i - h_t(\mathbf{x_i}))^2$

$w_t = \frac{1}{2} \log\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$

$$D_i^{(t+1)} = \frac{D_i^{(t)} \exp(-w_t y_i h_t(\mathbf{x}_i))}{\sum_{j=1}^{m} D_j^{(t)} \exp(-w_t y_j h_t(\mathbf{x}_j))} \quad \forall i = 1, \ldots, m$$

Output:

- The hypothesis: $h_S(\mathbf{x}) = \sum_{t=1}^{T} w_t h_t(\mathbf{x})$

Our Weak Learner: One Layer Neural Networks

What is a **weak learner**:

- Typically a simple learning model that performs slightly better than random guessing, with limited results
- Boosting Algorithm can produce a strong learner by combining multiple weak learners together

We are choosing **One Layer Neural Network** as our weak learner due to its performance in regression tasks.

- Representation: $h(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle + b$
  - **w**: weight matrix
  - **x**: input vector
  - **b**: bais vector

Changing the Network for Weighted Training

$$L_S(h_t) = \sum_{i=1}^{m} w_i \cdot (y_i - h_t(\mathbf{x}_i))^2$$

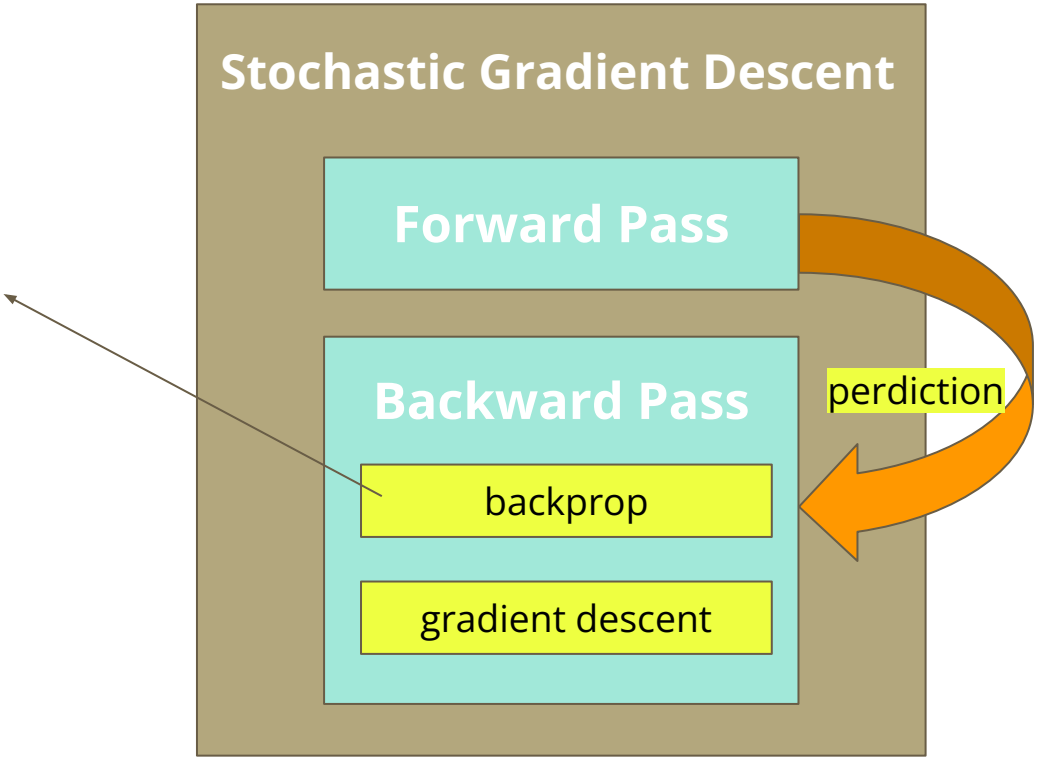
$L_S(h_t)$: Total loss for the model h_t, based on the dataset.

$h_t(x_i)$: Predicted value for input x_i using model h_t.

y_i: True value (target) for the i-th data point.

w_i: Weight for the i-th data point in the loss function.

m: Total number of data points in the dataset.



Previous Work (AdaBoost):

Regular Article

A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting ☆, ☆☆

Yoav Freund, Robert E Schapire †

AT&T Labs, 180 Park Avenue, Florham Park, New Jersey, 07932

Received 19 December 1996, Available online 25 May 2002.

Improving Regressors using Boosting Techniques

Harris Drucker
Monmouth University
West Long Branch, NJ 07764
drucker@monmouth.edu

AdaBoostRegressor

```
class sklearn.ensemble.AdaBoostRegressor(estimator=None, *,
n_estimators=50, learning_rate=1.0, loss='linear', random_state=None)
```

Reproducing the Previous Work (sklearn)

We were able to test our boosting algorithm against sklearn directly by modifying our weak learner to work in sklearn model for comparison.

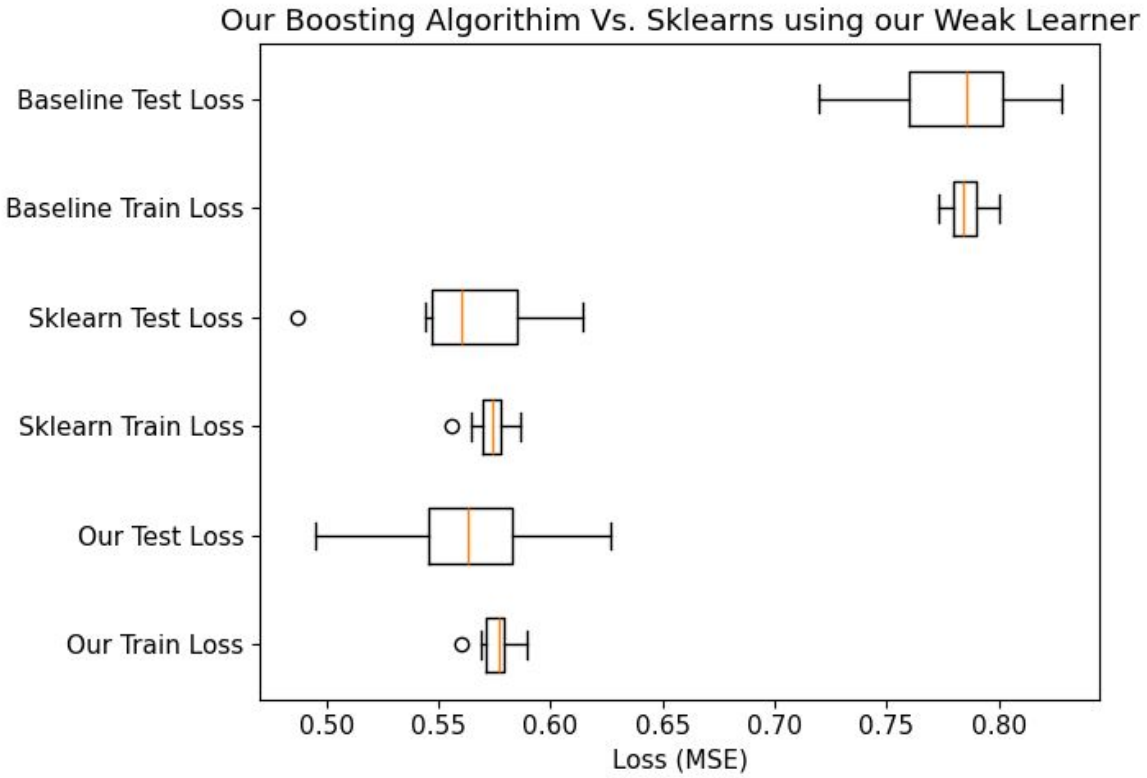
- We also had to modify the One Layer NN to support weighted training/loss.

```
from sklearn.base import BaseEstimator, RegressorMixin

class OneLayerNN(BaseEstimator, RegressorMixin):
    '''
        One layer neural network trained with Stocastic Gradient Descent (SGD)
    '''
```

```
model = AdaBoostRegressor(OneLayerNN(), n_estimators=n_estimators, learning_rate=learning)
```

Results of Reproducing the Previous Work



Dataset: UCI Wine Ratings Dataset

- Predict wine rating based on attributes of the wine

Analysis done over 10 random seeds

Summary

In sum, we successfully reproduced the results of the previous works we looked at and implemented Adaptive Boosting using a 1-layer NN as our weak learner.

Challenges:

- Working with data shapes was buggy
- Figuring out how to incorporate data weights into our weak learner
- Picking hyperparameters (learning weights, num_learners)
- Unit Tests

Most Interesting:

- Comparing our implementation to SciKit Learn
- Adapting the One Layer NN for weighted training

Questions?