Problem 5

Mohamad Lakkis

May, 2024

This PDF is to explain *Problem 5*

Explaining the algorithms that I used to plot the 3 figures

Algorithm 1 Gradient Tree Boosting Algorithm

Initialize $f_0(x) = \arg\min_\gamma \sum_{i=1}^{N} L(y_i, \gamma)$
for $m = 1$ to M **do**
  for $i = 1$ to N **do**
    Compute $r_{im} = -\left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right] f = fm - 1$
  end for
  Fit a regression tree to the targets r_{im}, giving terminal regions R_{jm}, $j = 1, 2, \ldots, J_m$
  for $j = 1$ to J_m **do**
    Compute $\gamma_{jm} = \arg\min_\gamma \sum_{x_i \in R_{jm}} L\left(y_i, f_{m-1}(x_i) + \gamma\right)$
  end for
  Update $f_m(x) = f_{m-1}(x) + \nu \cdot \sum_{j=1}^{J} \gamma_{jm} I(x \in R_{jm})$.
end for
Output $\hat{f}(x) = f_M(x)$

- In order to plot any curve for the gradient tree boosting algorithm, we need to specify the loss we want to use, since it is not specified we will be using be default the log loss for classification(note that if we use the exponential loss, gradient boosting becomes the AdaBoost algorithm, see additional notes for the proof) and the squared loss for regression. In addition to that, we need to specify the number of estimators(i.e. M), the interaction depth of the tree(i.e. J), the learning rate(i.e. ν).
- In figure 15.1, we have used the following parameters: M ranging in values , $J = 5$, $\nu = 1$(to remove penalization). And since in this case we are using classification, we have used the log loss.

- In figure 15.3, we have used the algorithm twice since we have two different settings, the first setting is: M ranging in values, $J = 4$, $\nu = 0.05$, and the second setting is: M ranging in values, $J = 6$, $\nu = 0.05$. And since in this case we are using regression, we have used the exponential loss.

- Final note on this algorithm, γ_{jm} is the weight for each leaf node in the regression tree fitted at each boosting iteration m. It is computed for each terminal region R_{jm} of the tree and represents the value that minimizes the loss function in that region.

Algorithm 2 Random Forest for Regression or Classification(as it is in the book, I wrote here just to explain the parameters that I am going to use to draw the figure)

1: **for** $b = 1$ to B **do**

2: (a) Draw a bootstrap sample $\mathbb{Z}^*$ of size N from the training data.

3: (b) Grow a random-forest tree T_b to the bootstrapped data, by recursively repeating the following steps for each terminal node of the tree, until the minimum node size $n_{\min}$ is reached.

i. Select m variables at random from the p variables.

ii. Pick the best variable/split-point among the m.

iii. Split the node into two daughter nodes.

4: **end for**

5: Output the ensemble of trees $\{T_b\}_1^B$

6: To make a prediction at a new point x:

Regression: $\hat{f}_{\text{rf}}^B(x) = \frac{1}{B}\sum_{b=1}^{B} T_b(x)$.

Classification: Let $\hat{C}_b(x)$ be the class prediction of the b-th random-forest tree. Then $\hat{C}_{\text{rf}}^B(x) = \text{majority vote}\{\hat{C}_b(x)\}_1^B$.

- We have used in figure 15.3, B = M(ranging values), $m = 2, 6$

- As for figure 15.1, we have used B = M(ranging values), m =default value since it is not specified.

- As for the OOB, there is a very convenient way to calculate the out-of-bag-error in sklearn, by simply adding oob score = True as a parameter in the RandomForestClassifier or RandomForestRegressor. If we want to calculate the OOB error manually, there is the pseudo-code in the Additional Notes section.

Algorithm 3 Bagging Algorithm (Bagging a special case of random forest where m = p) (Default values for the parameters are used in this case, since it is not specified)

1: **for** $b = 1$ to B **do**
2: (a) Draw a bootstrap sample $\mathbb{Z}^*$ of size N from the training data.
3: (b) Grow a random-forest tree T_b to the bootstrapped data, by recursively repeating the following steps for each terminal node of the tree, until the minimum node size $n_{\min}$ is reached.

i. Select all p variables (since m = p).

ii. Pick the best variable/split-point among the p.

iii. Split the node into two daughter nodes.

4: **end for**
5: Output the ensemble of trees $\{T_b\}_1^B$
6: To make a prediction at a new point x:
Regression: $\hat{f}_{\text{rf}}^B(x) = \frac{1}{B}\sum_{b=1}^{B} T_b(x)$.
Classification: Let $\hat{C}_b(x)$ be the class prediction of the b-th bagging. Then $\hat{C}_{\text{rf}}^B(x) = \text{majority vote}\{\hat{C}_b(x)\}_1^B$.

Figures and Plots

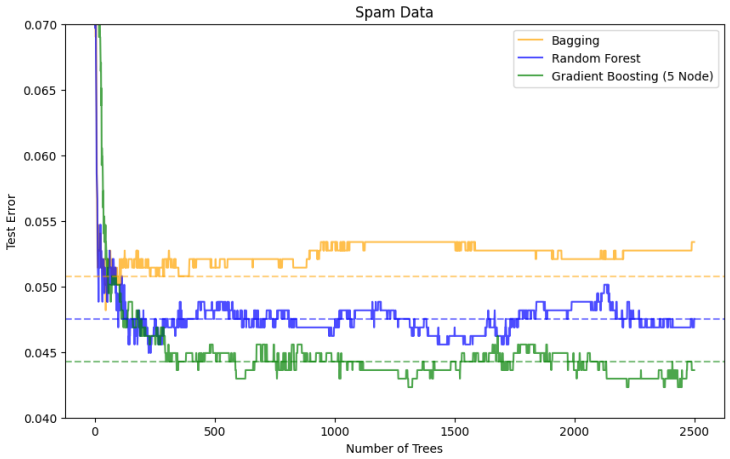

Figure 15.1: Bagging, random forest, and gradient boosting, applied to the spam data

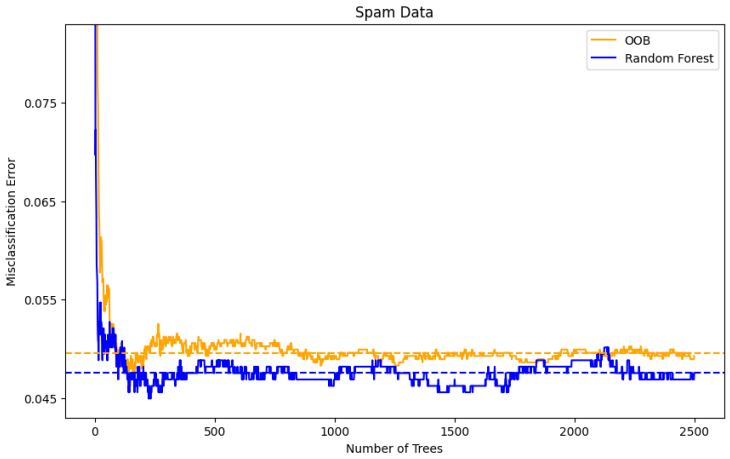

Figure 15.4: oob error computed on the spam training data, compared to the random forest test error computed on the test set.

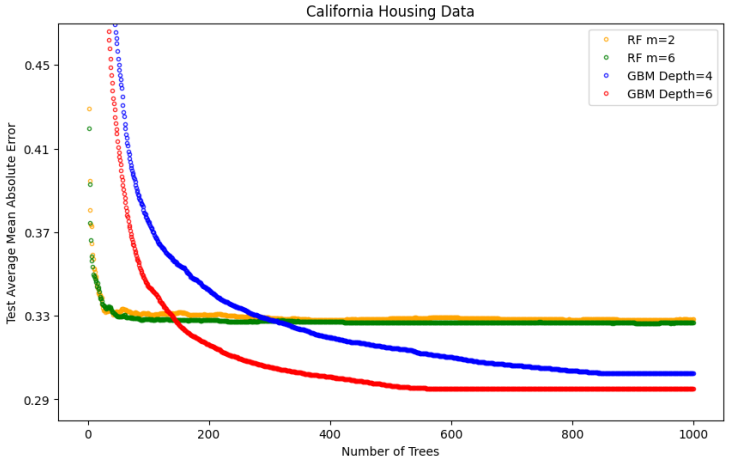

Figure 15.3: Random forests compared to gradient boosting on the California housing data.

- In figure 15.3, we averaged the test error over many test data, so for each split we have a test error, and we have averaged the test error over all the splits. And in this case the error was MAE(i.e. Mean Absolute Error), which can be calculated as follows: $MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{f}_i|$, where y_i is the actual value and $\hat{f}_i$ is the predicted value. So if the general formula over many test data is $MAE_j = \frac{1}{K}\frac{1}{n}\sum_{i=1}^{n}\sum_{k=1}^{K}|y_i - \hat{f}_k(x_i)|$, where K is the number of splits, n is the number of test data, $\hat{f}_k$ is the estimated

function for the k-th split, and MAE_j is the Test Average Mean Absolute Error(i.e. the y-axis in the figure). And we repeat this process for all number of trees ranging from $j = 1, ..., , 1000$.

- In figure 15.3, we can clearly see that the GBM performs better than both random forests algorithms, and the random forest with $m = 6$ performs slightly better than the random forest with $m = 2$.We can notice that for random forest it stabilizes after a certain number of trees(around 150 to 200 trees). But as for GBM with depth = 6, it starts to perform better than the random forests after around 180 trees, and it keeps on improving until 700 trees, then it starts to stabilize. But after 700 trees the best performance comes from this GBM with depth = 6. Similarly, for GBM with depth = 4, it starts to perform better than the random forests after 350 trees, and it keeps on improving until around 850 trees where it starts to stabilize. So the best performing model is the GBM with depth = 6.

- We can see from figure 15.4 that the OOB error is a very good estimate of the random forest test error. And we can see from this graph that if we where to use the random forest algorithm around 500 trees would be efficient since after this point there is not a remarkable improvement in the test error.

- In figure 15.1, we can see that the gradient boosting algorithm performs better than the random forest and bagging algorithms. And we can see that the random forest algorithm performs better than the bagging algorithm. But we must keep in mind how each algorithm will generalize to new data.

- In figure 15.1, we can also see that all three methods show a high test error at the beginning (when the number of trees is low). After a certain point, adding more trees does not reduce the test error and may lead to over-fitting.

Additional Notes

Algorithm 4 Gradient Tree Boosting Algorithm with Exponential Loss

Initialize $f_0(x) = \arg\min_\gamma \sum_{i=1}^{N} L(y_i, \gamma)$
for $m = 1$ to M **do**
    for $i = 1$ to N **do**
        Compute the residuals $r_{im} = -\left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f=f_{m-1}}$
        For exponential loss $L(y_i, f(x_i)) = e^{-y_i f(x_i)}$, the gradient is $r_{im} = y_i e^{-y_i f_{m-1}(x_i)}$
    end for
    Fit a regression tree to the targets r_{im}, giving terminal regions R_{jm}, $j = 1, 2, \ldots, J_m$
    for $j = 1$ to J_m **do**
        Compute $\gamma_{jm} = \arg\min_\gamma \sum_{x_i \in R_{jm}} L(y_i, f_{m-1}(x_i) + \gamma)$
        For exponential loss, this simplifies to $\gamma_{jm} = \frac{1}{2} \log \frac{1-err_{jm}}{err_{jm}}$, where err_{jm} is the weighted error rate for region R_{jm}
    end for
    Update the model $f_m(x) = f_{m-1}(x) + \sum_{j=1}^{J_m} \gamma_{jm} I(x \in R_{jm})$
end for
Output $\hat{f}(x) = f_M(x)$

Algorithm 5 Random Forest with OOB Error Calculation

for $b = 1$ to B **do**
    Draw a bootstrap sample $\mathbb{Z}^*$ of size N from the training data.
    Grow a random-forest tree T_b to the bootstrapped data.
    Identify the (OOB) samples for the b-th tree.
    Predict the OOB samples using the b-th tree and save the predictions.
end for
Add the OOB predictions:
for each observation i in the dataset **do**
    Collect predictions for i from all trees where i is OOB.
    ADD these predictions to get the final OOB prediction for i (average for regression, majority vote for classification).
end for
Calculate the OOB error:
Compute the error between the OOB predictions and the actual values (MSE for regression, classification error rate for classification).
Output the ensemble of trees $\{T_b\}_1^B$ and the OOB error.