Recitation 10

Trees, Bootstrap, Bagging, Random Forest, Adaboost

DS-GA 1003 Machine Learning

CDS

April 5, 2023

Agenda

1. Trees
2. Bootstrap
3. Bagging
4. Boosting
5. Takeaways

Decision Trees

- Non-linear classifier that groups examples hierarchically to make predictions
- Each leaf node refers to a subset of data points and each split is based on a single feature
- Finding the optimal tree is intractable $\rightarrow$ Greedy selection methods
- Prediction is an average over all points present in the current bucket i.e. $\hat{y}_i = \sum_{j=1}^{m} y_j | x_j$

Trees - Q1

Q. How many regions (leaves) will a tree with k node splits have?

Trees - Q1

Q. How many regions (leaves) will a tree with k node splits have?

A. Given a fixed tree, if we split a leaf node we add a single leaf to the tree. Thus k splits corresponds to k + 1 leaves

Trees - Q2

Q. What is the maximum number of regions a tree of height k can have? Recall that the height of a tree is the number of edges in the longest path from the root to any leaf.

Trees - Q2

Q. What is the maximum number of regions a tree of height k can have? Recall that the height of a tree is the number of edges in the longest path from the root to any leaf.

A. A tree of height k can have at most 2^k regions (leaves).

Trees - Q3

Q. Give an upper bound on the depth needed to exactly classify n distinct points in R^1. [Hint: In the worst case each leaf will have a single training point.]

Trees - Q3

Q. Give an upper bound on the depth needed to exactly classify n distinct points in R^1. [Hint: In the worst case each leaf will have a single training point.]

A. A tree of height $\lceil \log_2(n) \rceil$ can put each example into its own bucket.

Misclassification error in a node

- Let's consider the multiclass classification case: $\mathcal{Y} = \{1, 2, \ldots, K\}$.
- Let node m represent region R_m, with N_m observations
- We denote the proportion of observations in R_m with class k by

$$\hat{p}_{mk} = \frac{1}{N_m} \sum_{\{i: x_i \in R_m\}} \mathbf{1}(y_i = k).$$

- We predict the majority class in node m:

$$k(m) = \arg\max_k \hat{p}_{mk}$$

Node Impurity Measures and Split Points

- Misclassification error

$$1 - \hat{p}_{mk(m)}.$$

- The Gini index encourages $\hat{p}_{mk}$ to be close to 0 or 1

$$\sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$$

- Entropy / Information gain

$$-\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk}.$$

Quantifying the Impurity of a Split

Whatever the chosen measure, we use it to quantify candidate splits and select the best candidate. Consider a potential split that produces the nodes R_L and R_R

- Suppose we have N_L points in R_L and N_R points in R_R.
- Let $Q(R_L)$ and $Q(R_R)$ be the node impurity measures for each node.
- We aim to find a split that minimizes the *weighted average of node impurities*:

$$\frac{N_L Q(R_L) + N_R Q(R_R)}{N_L + N_R}$$

Trees - Q4

Q. Suppose we are looking at a fixed node of a classification tree, and the class labels are, sorted by the first feature values,

$$4, 1, 0, 0, 1, 0, 2, 3, 3.$$

We are currently testing splitting the node into a left node containing $4, 1, 0, 0, 1, 0$ and a right node containing $2, 3, 3$. For each of the following impurity measures, give the value for the left and right parts, along with the total score for the split.

1. Gini index.
2. Entropy.

Trees - Q4

1. Gini:
   - Left: $3/6(3/6) + 2/6(4/6) + 1/6(5/6) = 22/36$
   - Right: $1/3(2/3) + 2/3(1/3) = 4/9$
   - Total: $6(22/36) + 3(4/9) = 30/6 = 5$

Trees - Q4

1. Gini:
   - Left: $3/6(3/6) + 2/6(4/6) + 1/6(5/6) = 22/36$
   - Right: $1/3(2/3) + 2/3(1/3) = 4/9$
   - Total: $6(22/36) + 3(4/9) = 30/6 = 5$
2. Entropy:
   - Left: $-3/6 \log(3/6) - 2/6 \log(2/6) - 1/6 \log(1/6)$
   - Right: $-1/3 \log(1/3) - 2/3 \log(2/3)$
   - Total: $6[-3/6 \log(3/6) - 2/6 \log(2/6) - 1/6 \log(1/6)] + 3[-1/3 \log(1/3) - 2/3 \log(2/3)]$.

Trees

- Interpretable and great at capturing non-linear boundaries
- With the caveat of a high chance of overfitting and high variance

Bootstrap

- Averaging over many samples reduces variance
- But we don't have access to many data samples.
- A **bootstrap method** simulates B independent samples from P by taking B bootstrap samples from the sample $\mathcal{D}_n$.
- Turns out this works quite well
- We also now have B instances of any statistic (median, variance etc.) so we can calculate properties of this distribution

Bootstrap - Q5

Let $X_1, \ldots, X_{2n+1}$ be an i.i.d. sample from a distribution. To estimate the median of the distribution, you can compute the sample median of the data.

- How do we compute an estimate of the variance of the sample median?

Bootstrap - Q5

Let $X_1, \ldots, X_{2n+1}$ be an i.i.d. sample from a distribution. To estimate the median of the distribution, you can compute the sample median of the data.

- How do we compute an estimate of the variance of the sample median?
  - Draw B bootstrap samples $D^1, \ldots, D^B$ each of size $2n+1$. The samples are formed by drawing uniformly with replacement from the original data set $X_1, \ldots, X_{2n+1}$. We will make a total of $B(2n+1)$ draws.
  - For each D^i compute the corresponding median $\hat{m}_i$.
  - Compute the sample variance of the B medians $m_1, \ldots, m_B$.

Bagging: Bootstrap Aggregation

- We draw B bootstrap samples $D^1, \ldots, D^B$ from original data $\mathcal{D}$
- Let $\hat{f}_1, \hat{f}_2, \ldots, \hat{f}_B$ be the prediction functions resulting from training on $D^1, \ldots, D^B$, respectively
- The **bagged prediction function** is a *combination* of these:

$$\hat{f}_{\text{avg}}(x) = \text{Combine}\left(\hat{f}_1(x), \hat{f}_2(x), \ldots, \hat{f}_B(x)\right)$$

- You reduce variance in prediction and incur a cost on interpretability

Bagging - Q6

- What is the average variance of the predictor, $\hat{f}_{\text{avg}}$? (Assume you're taking an average to combine predictions)

Bagging - Q6

- What is the average variance of the predictor, $\hat{f}_{\text{avg}}$? (Assume you're taking an average to combine predictions)

  $\text{Var}(\hat{f}_{\text{avg}}(x)) = \text{Var}\left(\frac{1}{B}\sum_{b=1}^{B}\hat{f}_b(x)\right) = \frac{1}{B^2}\text{Var}\left(\sum_{b=1}^{B}\hat{f}_b(x)\right)$

  Assuming the various f_i are independent prediction functions.

  So $\frac{1}{B^2}\text{Var}\left(\sum_{b=1}^{B}\hat{f}_b(x)\right) = \frac{1}{B^2}B * \text{Var}\left(\hat{f}_1(x)\right) = \frac{1}{B}\text{Var}\left(\hat{f}_1(x)\right)$

Bagging - Q6

- The above equality gives some intuition as to why bagging might reduce variance. But really, but situation is more complicated: the bootstrap samples used for bagging are not independent.
- A bootstrap sample from $\mathcal{D}_n = (x_1, \ldots, x_n)$ is a sample of size n drawn with replacement from $\mathcal{D}_n$. So there can be overlaps between bootstrap samples.

Bagging - Q7

If the variance of the individual predictors that we are bagging is σ^2, and the correlation between them is ρ^2, what is the variance of the bagged predictor?

Bagging - Q7

If the variance of the individual predictors that we are bagging is σ^2, and the correlation between them is ρ^2, what is the variance of the bagged predictor?

$$\text{Var}\left(\frac{1}{B}\sum_{b=1}^{B}\hat{f}_b\right) \tag{1}$$

$$= \frac{1}{B^2}\left[(B * \text{Var}(\hat{f}_1(x)) + B(B-1) * \text{Cov}(\hat{f}_1(x), \hat{f}_2(x))\right] \tag{2}$$

$$= \frac{1}{B}\sigma^2 + \frac{B-1}{B}\rho^2 \tag{3}$$

Random Forest - Q8

Bagging decision trees leads us to the highly popular random forests. However, to make bagging for decision trees work well, we need one more key idea:

We randomly sample features when building trees to reduce the covariance between trees.

Boosting

- Why train the different estimators independently?
- We can do better by making them target the mistakes of one-another

Adaboost Algorithm

Given training set $\mathcal{D} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$.

1. Initialize observation weights $w_i = 1,\ i = 1, 2, \ldots, n$.

Adaboost Algorithm

Given training set $\mathcal{D} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$.

1. Initialize observation weights $w_i = 1$, $i = 1, 2, \ldots, n$.
2. For $m = 1$ to M:
    1. Base learner fits weighted training data and returns $G_m(x)$

Adaboost Algorithm

Given training set $\mathcal{D} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$.

1. Initialize observation weights $w_i = 1$, $i = 1, 2, \ldots, n$.
2. For $m = 1$ to M:
   1. Base learner fits weighted training data and returns $G_m(x)$
   2. Compute **weighted empirical 0-1 risk**:

$$\text{err}_m = \frac{1}{W} \sum_{i=1}^{n} w_i \mathbf{1}(y_i \neq G_m(x_i)) \quad \text{where } W = \sum_{i=1}^{n} w_i.$$

Adaboost Algorithm

Given training set $\mathcal{D} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$.

1. Initialize observation weights $w_i = 1$, $i = 1, 2, \ldots, n$.
2. For $m = 1$ to M:
   1. Base learner fits weighted training data and returns $G_m(x)$
   2. Compute **weighted empirical 0-1 risk**:

$$\text{err}_m = \frac{1}{W} \sum_{i=1}^{n} w_i \mathbf{1}(y_i \neq G_m(x_i)) \quad \text{where } W = \sum_{i=1}^{n} w_i.$$

   3. Compute $\alpha_m = \ln\left(\frac{1-\text{err}_m}{\text{err}_m}\right)$ [**classifier weight**]

Adaboost Algorithm

Given training set $\mathcal{D} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$.

1. Initialize observation weights $w_i = 1$, $i = 1, 2, \ldots, n$.
2. For $m = 1$ to M:
   1. Base learner fits weighted training data and returns $G_m(x)$
   2. Compute **weighted empirical 0-1 risk**:

$$\text{err}_m = \frac{1}{W} \sum_{i=1}^{n} w_i \mathbf{1}(y_i \neq G_m(x_i)) \quad \text{where } W = \sum_{i=1}^{n} w_i.$$

   3. Compute $\alpha_m = \ln\left(\frac{1-\text{err}_m}{\text{err}_m}\right)$ [**classifier weight**]
   4. Set $w_i \leftarrow w_i \cdot \exp\left[\alpha_m \mathbf{1}(y_i \neq G_m(x_i))\right], \quad i = 1, 2, \ldots, n$ [**example weight adjustment**]

Adaboost Algorithm

Given training set $\mathcal{D} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$.

1. Initialize observation weights $w_i = 1$, $i = 1, 2, \ldots, n$.
2. For $m = 1$ to M:
   1. Base learner fits weighted training data and returns $G_m(x)$
   2. Compute **weighted empirical 0-1 risk**:

$$\text{err}_m = \frac{1}{W} \sum_{i=1}^{n} w_i \mathbf{1}(y_i \neq G_m(x_i)) \quad \text{where } W = \sum_{i=1}^{n} w_i.$$

   3. Compute $\alpha_m = \ln\left(\frac{1-\text{err}_m}{\text{err}_m}\right)$ [**classifier weight**]
   4. Set $w_i \leftarrow w_i \cdot \exp\left[\alpha_m \mathbf{1}(y_i \neq G_m(x_i))\right], \quad i = 1, 2, \ldots, n$ [**example weight adjustment**]
3. Ouptut $G(x) = \text{sign}[\sum_{m=1}^{M} \alpha_m G_m(x)]$.

Adaboost - Q9

Decide whether each of the statements below is true or false.

- The error rate of the ensemble classifier never increases from one round to the next.

Adaboost - Q9

Decide whether each of the statements below is true or false.

- The error rate of the ensemble classifier never increases from one round to the next. **False**. There is no guarantee for error rate not to increase from one round to the next.

Adaboost - Q9

Decide whether each of the statements below is true or false.

- The error rate of the ensemble classifier never increases from one round to the next. **False**. There is no guarantee for error rate not to increase from one round to the next.
- Adaboost accounts for outliers by lowering the weights of training points that are repeatedly misclassified.

Adaboost - Q9

Decide whether each of the statements below is true or false.

- The error rate of the ensemble classifier never increases from one round to the next. **False**. There is no guarantee for error rate not to increase from one round to the next.
- Adaboost accounts for outliers by lowering the weights of training points that are repeatedly misclassified. **False**. Adaboost will increase the weights of training points that are repeatedly misclassified.

Adaboost - Q9

Decide whether each of the statements below is true or false.

- The error rate of the ensemble classifier never increases from one round to the next. **False**. There is no guarantee for error rate not to increase from one round to the next.
- Adaboost accounts for outliers by lowering the weights of training points that are repeatedly misclassified. **False**. Adaboost will increase the weights of training points that are repeatedly misclassified.
- When you update weights, the training point with the smallest weight in the previous round will always increase in weight.

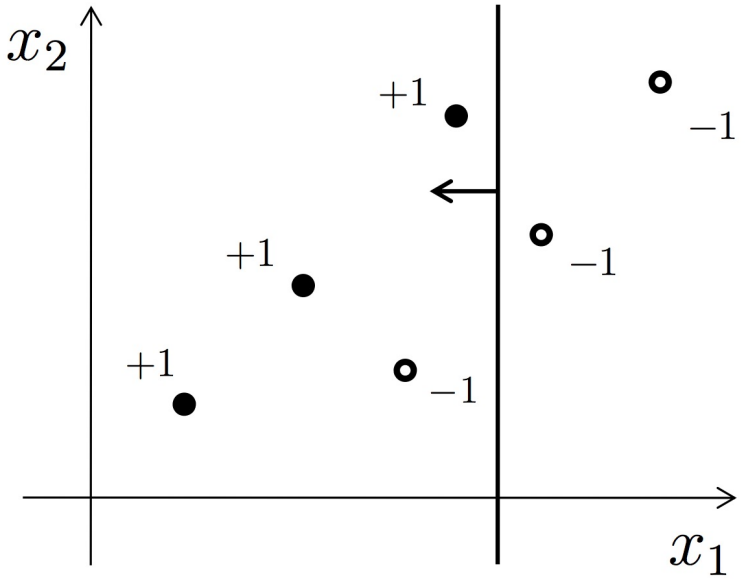
Adaboost - Q9

Decide whether each of the statements below is true or false.

- The error rate of the ensemble classifier never increases from one round to the next. **False**. There is no guarantee for error rate not to increase from one round to the next.
- Adaboost accounts for outliers by lowering the weights of training points that are repeatedly misclassified. **False**. Adaboost will increase the weights of training points that are repeatedly misclassified.
- When you update weights, the training point with the smallest weight in the previous round will always increase in weight. **False**. It will increase weight only if it is misclassified in the current round.

Adaboost - Q10

Consider building an ensemble of decision stumps Gm with the AdaBoost algorithm. $f(x) = \text{sign}\left(\sum_{m=1}^{M} \alpha_m G_m(x)\right)$ displays a few labeled point in two dimensions as well as the first stump we have chosen. A stump predicts binary ± 1 values, and depends only on one coordinate value (the split point). The little arrow in the figure is the normal to the stump decision boundary indicating the positive side where the stump predicts $+1$. All the points start with uniform weights.



Adaboost - Q10

- Circle all the point(s) in Figure 1 whose weight will increase as a result of incorporating the first stump (the weight update due to the first stump).

Adaboost - Q10

- Circle all the point(s) in Figure 1 whose weight will increase as a result of incorporating the first stump (the weight update due to the first stump). (sol.) The only misclassified negative sample.

Adaboost - Q10

- Circle all the point(s) in Figure 1 whose weight will increase as a result of incorporating the first stump (the weight update due to the first stump). (sol.) The only misclassified negative sample.
- Draw in the same figure a possible stump that we could select at the next boosting iteration. You need to draw both the decision boundary and its positive orientation.

Adaboost - Q10

- Circle all the point(s) in Figure 1 whose weight will increase as a result of incorporating the first stump (the weight update due to the first stump). (sol.) The only misclassified negative sample.
- Draw in the same figure a possible stump that we could select at the next boosting iteration. You need to draw both the decision boundary and its positive orientation. (sol.) The second stump will also be a vertical split between the second positive sample (from left to right) and the misclassified negative sample.

Adaboost - Q10

- Circle all the point(s) in Figure 1 whose weight will increase as a result of incorporating the first stump (the weight update due to the first stump). (sol.) The only misclassified negative sample.
- Draw in the same figure a possible stump that we could select at the next boosting iteration. You need to draw both the decision boundary and its positive orientation. (sol.) The second stump will also be a vertical split between the second positive sample (from left to right) and the misclassified negative sample.
- Will the second stump receive higher coefficient in the ensemble than the first? In other words, will $\alpha_2 > \alpha_1$? Briefly explain your answer. (no calculation should be necessary).

Adaboost - Q10

- Circle all the point(s) in Figure 1 whose weight will increase as a result of incorporating the first stump (the weight update due to the first stump). (sol.) The only misclassified negative sample.
- Draw in the same figure a possible stump that we could select at the next boosting iteration. You need to draw both the decision boundary and its positive orientation. (sol.) The second stump will also be a vertical split between the second positive sample (from left to right) and the misclassified negative sample.
- Will the second stump receive higher coefficient in the ensemble than the first? In other words, will $\alpha_2 > \alpha_1$? Briefly explain your answer. (no calculation should be necessary). (sol.) $\alpha_2 > \alpha_1$ because the point that the second stump misclassifies will have a smaller relative weight since it is classified correctly by the first stump.

Takeaways

- Trees are an interpretable, non-linear classifier
- To reduce variance, we can run bagging/random forest instead of relying on a single tree
- (Potentially) Better than training the predictors independently we can make them correct each other with boosting