Assignment 5

2021年11月15日 18:47

(a) Gain Ratio

Gain ratio aims to overcome the bias when we test with many outcomes by the information gain measure, it applies a kind of normalization to information gain using a "split information" value defined as

$$SplitInfo_A(D) = -\sum_{j=1}^{v} \left|\frac{D_j}{D}\right| \times \log_2\left(\frac{|D_j|}{|D|}\right).$$

$SplitInfo_A(D)$ represents the potential information generated by splitting the training data set D into v partitions, corresponding to the v outcomes of a test on attribute A.

The gain ratio is defined as

$$GainRatio(A) = \frac{Gain(A)}{SplitInfo_A(D)}$$

The attribute with the maximum gain ratio is selected as the splitting attribute.

$$Gain(A) = Info(D) - Info_A(D)$$

Gain(A), information gain of A, is defined as the difference between original information requirement and the new requirement(i.e., obtained after partitioning on A). It tells us how much would be gained by branching on A.

$$Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times Info(D_j)$$

$Info_A(D)$ is the expected information required to classify a tuple from D based on the partitioning by A.

$$Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i)$$

Info(D) is the expected information needed to classify a tuple in D.

Gini Impurity

The Gini impurity considers a binary split for each attribute, if A has v possible values, then there are 2^v possible subsets, totally (2^v-2)/2 possible ways to form two partitions of the

data.
If a binary split on A partitions D into D1 and D2, the Gini impurity of D given that partitioning is

$$Gini_A(D) = \frac{|D_1|}{|D|} Gini(D_1) + \frac{|D_2|}{|D|} Gini(D_2)$$

The reduction in impurity that would be incurred by a binary split on a discrete or continuous-valued attribute A is

$$\Delta Gini(A) = Gini(D) - Gini_A(D)$$

The attribute that maximizes the reduction in impurity is selected as the splitting attribute.

1.(b) Gini impurity:

$Gini_{Patron \in \{None, some\}}(D) = \frac{6}{12} Gini(D_1) + \frac{6}{12} Gini(D_2)$
$= \frac{6}{12} \cdot (1 - (\frac{2}{6})^2 - (\frac{4}{6})^2) + \frac{6}{12} \cdot (1 - (\frac{2}{6})^2 - (\frac{4}{6})^2)$
$= 0.444$

$Gini_{Patron \in \{Some, Full\}}(D) = \frac{2}{12} Gini(D_1) + \frac{10}{12} Gini(D_2)$
$= \frac{2}{12} \times (1 - 1^2) + \frac{10}{12} \times (1 - (\frac{6}{10})^2 - (\frac{4}{10})^2)$
$= 0.4$

$Gini_{Patron \in \{None, Full\}}(D) = \frac{4}{12} Gini(D_1) + \frac{8}{12} Gini(D_2)$
$= \frac{4}{12} \times (1 - 1^2) + \frac{8}{12} \times (1 - (\frac{2}{8})^2 - (\frac{6}{8})^2)$
$= 0.25$

$0.444 > 0.4 > 0.25$

Therefore, Gini $_{Patrons \in \{None, Full\}}(D)$ is the best split for Patros with a Gini impurity of 0.25.

$Gini_{Type \in \{French\}} = \frac{2}{12} Gini(D_1) + \frac{10}{12} Gini(D_2)$
$= \frac{2}{12} \times (1 - (\frac{1}{2})^2 - (\frac{1}{2})^2) + \frac{10}{12} \times (1 - (\frac{5}{10})^2 - (\frac{5}{10})^2) = 0.5$

$Gini_{Type \in \{Italian\}} = 0.5$

$Gini_{Type \in \{Thai\}} = \frac{4}{12} Gini(D_1) + \frac{8}{12} Gini(D_2)$
$= \frac{4}{12} \times (1 - (\frac{2}{4})^2 - (\frac{2}{4})^2) + \frac{8}{12} \times (1 - (\frac{4}{8})^2 - (\frac{4}{8})^2) = 0.5$

$Gini_{Type \in \{Burger\}} = 0.5$

$Gini_{Type \in \{French, Italian\}} = \frac{4}{12} Gini(D_1) + \frac{8}{12} Gini(D_2)$
$= \frac{4}{12} \times (1 - (\frac{2}{4})^2 - (\frac{2}{4})^2) + \frac{8}{12} \times (1 - (\frac{4}{8})^2 - (\frac{4}{8})^2) = 0.5$

$$Gini_{Type\in\{French,\ Italian\}} = \frac{4}{12}Gini(D_1) + \frac{8}{12}Gini(D_2)$$
$$= \frac{4}{12}\times\left(1-\left(\frac{2}{4}\right)^2-\left(\frac{2}{4}\right)^2\right) + \frac{8}{12}\times\left(1-\left(\frac{4}{8}\right)^2-\left(\frac{4}{8}\right)^2\right) = 0.5$$

$\vdots$

Above all, we can conclude that no matter how we split the attribution Type, the Gini impurities are always 0.5.

So, the best binary split for attribute Patrons is on {None, Full} because it minimizes the Gini impurity. Evaluating attribute Type, all Gini impurities of different binary splits are 0.5.

$$Gini(D) = 1-\left(\frac{6}{12}\right)^2-\left(\frac{6}{12}\right)^2 = 0.5$$

$$\Delta Gini(Patron) = 0.5 - 0.25 = 0.25$$
$$\Delta Gini(Type) = 0.5 - 0.5 = 0$$

For 0.25 > 0, the better attribute to split on at the root should be Patrons. The attribute Patrons and splitting subset{None, Full} give the minimum Gini impurity overall, with a reduction in impurity of 0.25.

(c) Gain ratio:

$$SplitInfo_{patron}(D) = -\sum_{j=1}^{v}\frac{|D_j|}{|D|}\times\log_2\left(\frac{|D_j|}{|D|}\right)$$
$$= -\frac{2}{12}\times\log_2\frac{2}{12} - \frac{4}{12}\times\log_2\frac{4}{12} - \frac{6}{12}\times\log_2\frac{6}{12}$$
$$\approx 1.459$$

$$GainRatio(Patron) = Gain(Patro)\,/\,Splitfo_{patron}(D)$$
$$= (Info(D) - Info_{patron}(D))\,/\,Splitfo_{patron}(D)$$
$$= \left[\left(-\frac{1}{2}\log_2\frac{1}{2} - \frac{1}{2}\log_2\frac{1}{2}\right) - \left(\frac{2}{12}\times\left(-\frac{2}{2}\log_2\frac{2}{2}\right) + \frac{4}{12}\times\left(-\frac{4}{4}\log_2\frac{4}{4}\right)\right.\right.$$
$$\left.\left. + \frac{6}{12}\times\left(-\frac{4}{6}\log_2\frac{4}{6} - \frac{2}{6}\log_2\frac{2}{6}\right)\right]\right] / 1.459$$
$$= (1 - 0.459)\,/\,1.459$$
$$= 0.371$$

$$SplitInfo_{type}(D) = -\frac{2}{12}\times\log_2\frac{2}{12}\times 2 - \frac{4}{12}\times\log_2\frac{4}{12}\times 2$$
$$\approx 1.918$$

$$GainRatio(Type) = \left[\left(-\frac{1}{2}\log_2\frac{1}{2} - \frac{1}{2}\log_2\frac{1}{2}\right) - \left(\frac{2}{12}\times\left(-\frac{1}{2}\log_2\frac{1}{2}\times 2\right)\right.\right.$$
$$\left.\left. + \frac{2}{12}\times\left(-\frac{1}{2}\log_2\frac{1}{2}\times 2\right) + \frac{4}{12}\times\log\left(-\frac{1}{2}\log_2\frac{1}{2}\times 2\right) + \frac{4}{12}\times\log_2\left(-\frac{1}{2}\log_2\frac{1}{2}\times 2\right)\right)\right] / 1.918$$
$$= (1-1)\,/\,1.918 = 0$$

For 0.371 > 0, the better attribute to split on at the root is Patrons.

2. (a) Assume "Candy = lime" is x

2. (a) Assume "Candy$_1$ = lime" is x

$$P(h_1|x) = \frac{P(x|h_1)\cdot P(h_1)}{P(x)} = \frac{P(x|h_1)\cdot P(h_1)}{\sum_{i=1}^{3} P(x|h_i)\cdot P(h_i)}$$

$$= \frac{0 \times \frac{1}{4}}{0\times\frac{1}{4} + 0.5\times\frac{1}{2} + 1\times\frac{1}{4}} = 0$$

$$P(h_2|x) = \frac{P(x|h_2)\cdot P(h_2)}{\sum_{i=1}^{3} P(x|h_i)\cdot P(h_i)} = \frac{0.5\times\frac{1}{2}}{0\times\frac{1}{4} + 0.5\times\frac{1}{2} + 1\times\frac{1}{4}} = 0.5$$

$$P(h_3|x) = \frac{1\times\frac{1}{4}}{0\times\frac{1}{4} + 0.5\times\frac{1}{2} + 1\times\frac{1}{4}} = 0.5$$

(b) Assume "Candy$_2$ = cherry" is Y

$$P(x,Y|h_1) = 0, \quad P(x,Y|h_2) = 0.5\times 0.5 = 0.25$$
$$P(x,Y|h_3) = 0$$

$$P(h_1|x,Y) = \frac{P(x,Y|h_1)\cdot P(h_1)}{\sum_{i=1}^{3} P(x,y|h_i)\cdot P(h_i)} = \frac{0\times\frac{1}{4}}{0\times\frac{1}{4} + 0.25\times\frac{1}{2} + 0\times\frac{1}{4}} = 0$$

$$P(h_2|x,Y) = \frac{0.25\times\frac{1}{2}}{0\times\frac{1}{4} + 0.25\times\frac{1}{2} + 0\times\frac{1}{4}} = 1$$

$$P(h_3|x,Y) = 0$$

3. (a) There are 3 independent parameters, variable Size has 2 possible value "Large", "Small", so we need parameters:
P(Size = small | good apple = yes) = 1 - P(Size = large | good apple = yes),
P(Size = small | good apple = no) = 1 - P(Size = large | good apple = no).

Idem, totally we require independent parameters:

1. P(good apple = yes)
2. P(Size = small | good apple = yes)
3. P(Size = small | good apple = no)
4. P(Color = Green | good apple = yes)
5. P(Color = Green | good apple = no)
6. P(Shape = Irregular | good apple = yes)
7. P(Shape = Irregular | good apple = no)

(b)

1. P(good apple = yes) = 4/10
2. P(Size = small | good apple = yes) = 1/4
3. P(Size = small | good apple = no) = 1/2
4. P(Color = Green | good apple = yes) = 0
5. P(Color = Green | good apple = no) = 5/6
6. P(Shape = Irregular | good apple = yes) = 1/4
7. P(Shape = Irregular | good apple = no) = 4/6

For the first conditional probability parameters P(Size = small | good apple = yes), there are 4 good apples in total, but only one of them belong to "Size = small", then we can obtain the conditional probability is 1/4.

(c) x = (Small; Red; Circle)

P(x | y=yes) = P(Size = small | y=yes) * P(Color = red | y=yes) * P(Shape = Circle | y=yes) = 1/4 * 1 * 3/4 = 0.1875
P(x | y=no) = P(Size = small | y=no) * P(Color = red | y=no) * P(Shape = Circle | y=no)
= 3/6 * 1/6 * 2/6 = 0.0278

To find the class that maximizes P(x|y)P(y), we compute

P(x | y=yes) * P(y=yes) = 0.1875 * (4/10) = 0.075
P(x | y=no) * P(y=no) = 0.0278 * (6/10) = 0.0167

P(x | y=yes) / (P(x | y=yes) * P(y=yes) + P(x | y=no) * P(y=no)) = 0.1875/(0.075+0.0167)=2.0447
P(x | y=no) / (P(x | y=yes) * P(y=yes) + P(x | y=no) * P(y=no)) = 0.0278/(0.075+0.0167)=0.3032

Because 2.0447 > 0.3032, we conclude that naïve Bayes predicts y=yes for x.

4. (a)

We pick a random sample of the entire Data set, which size does not have to be the size of the whole data set. Also, a data point can be present more than once in the data used to train a single tree, which is called Sampling with Replacement or Bootstrapping. Ensemble models work best if the individual models are uncorrelated. So in RFs, we randomly selecting certain features to evaluate at each node, this avoids including features that have a very high predictive power in every tree, while creating many un-correlated trees. Above all, for randomness, we use random data and random features.

The whole process goes as follows:

1. Create a bootstrapped data set for each tree
2. Create a decision tree using its own data set, but at each node use a random sub sample of features to split on.
3. Repeat all these three steps hundreds of times to build a massive forest with many different trees. Here the CART methodology is used to grow the trees. The trees are grown to maximum size and are not pruned.

To predict a test point, we pass it to every trees and obtain an overall, aggregated prediction. During classification of the test point, each tree votes and the most popular class is returned.

(b)

d - the maximum depth of the individual trees. The larger d, the more chance it has of overfitting the training data. This is not such a big problem in RFs, as we have many individual trees.

m - number of random features to include at each node for splitting.

T - if you have a small number of trees while you are not using the whole data set to train each tree, then some observations

can be left out. So appropriately increasing the number of trees generally reduce model error. But it should not be too large, we need consider the cost of a higher training time.

(c) the expected number of unique samples from the original set of n samples in the bootstrapped sample is (1-1/e)n, ≈0.632n. Suppose that we have an original sample x1, x2, …, xn with n items inside. We draw items with replacement.

For the first draw, it is easy to know that the probability of choosing any one item(eg.x1) is 1-1/n; In total we need n draws, all of which are independent, so the probability of never choosing this item on all draws is (1-1/n)^n.

Now, we let n gets larger and larger, that means take the limit as n goes toward infinity:

$$\lim_{n\to\infty}\left(1-\frac{1}{n}\right)^n = \frac{1}{e} \approx 0.368$$

That's the probability of an item not being chosen in n draws. Then we can obtain the probability of an item being chosen in n draws:

$$1-\frac{1}{e} \approx 0.632$$

In conclusion, the the expected number of unique samples in n draws is (1-1/e)n.

(d)

I don't agree with Professor Forest's claims. A small m does reduce correlation of trees, however, increasing m makes individual trees more powerful. The optimal value of m achieves a trade off between these two opposing effects, and typically lies somewhere in the middle of the range, but not 1. Instead of setting m=1 curtly, the best m depends on the problem, it should be treated as a tuning parameter. Moreover, recommended default values are m = p/3 for regression problems and m = p^(1/2) for classification problems. (P is the number of features/predictors)

Extra Credit.

TP = a = 2588, FN = b = 412, FP = c = 46, TN = d =6954
P = TP + FP = 2634, N = FN + TN = 7366, All = 10000

1.

(a) Sensitivity = TP/P = 2588/2634 = 0.983

(b) Specificity = TN/N = 6954/7366 = 0.944

(c) Accuracy = (TP+TN)/All = (2588+6954)/10000 = 0.954

(d) Precision = TP/(TP+FP) = 2588/(2588+46) = 0.983

(e) Recall = R = TP/(TP+FN) = 2588/(2588+412) = 0.863

(e) Recall = R = TP/(TP+FN) = 2588/(2588+412) = 0.863

(f) F1 score = 2Precision*R/(Precision + R) = 2*0.983*0.863/(0.983 +0.863) = 0.919

I don't agree with Professor Griffin, precision should have a higher priority than recall, because commonly we would rather reading a spam email than missing an important email, which means the ideal spam detector must guarantee high precision.