Introduction to Decesion trees

Notes of class

Iván Andrés Trujillo Abella

FACULTAD DE INGENIERÍA

0.1 Decision trees

Powerful algorithm, based in measures of homogenity in this notes we discuss and construct over the concept of gain information and entropy.

Data: Empty tree ;
while *features to split* **do**
  Select the variable to split data ;
  repeat the before steps again until reach a stopping criteria;
end

0.2 Entropy

Derived from Information theory, we can measure the homogenity of a set, $E = 1$ at maximun disorder, and $E = 0$ when.

We also, can take in mind that the construction is recursive implementation.

Write pseudocode.

```
while stopping criteria is false:
    select the better variable:
        update data:
            select the better variable
```

1 Overfitting

Hint: A big gap in validation error with training error are a strong signal about over fitting.

We can select the better depth $\min E_{validation} - E_{training}$ select

The are some ways of avoid overfiting, however note that this is not a solution to the generalization.

The depth of three reduce training error, therefore decision boundaries are more complex.

1.1 Early stopping

Before of construc the tree: **limit the depth of tree**: We can uses cross validation:

1.2 Prunning

If we select a tree based in $R(T)$ that is resubstitution rate is the rate of responses predicted well with training dataset.

select α we need uses cross validation to select the parameter

1.3 Feature importance

How we can establish, what is the degree of importance of the variables?

hint: if you have two tress that have the same validation error pick always the simplest model.

Algorithm 1: Cross validation

```
Result: Write here the result
initialization;
while While condition do
    instructions;
    if condition then
        instructions1;
        instructions2;
    else
        instructions3;
    end
end
```

Algorithm 2: Cross validation

```
for i ← 2 to l do
    test data i
end
```

Algorithm 3: Simple algorithm

```
for i to 4 do
    print in screen i ;
    if i > 4 then
        print this number is major 4;
        if i = 3 then
            three
        end
    else
        print is lesser ;
    end
end
```

1.4 K fold cross validation

K means samples of K size, therefore we could have $\frac{N}{k}$, where N is the total amount of instances or observations.

$MSE = \frac{1}{k} \sum_{i=1}^{k} Accuracy_i$.

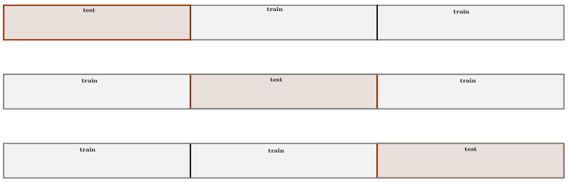
with cross validation we search optimize the parameter max_depth in librarie sickit-learn.

Algorithm 4: K fold: Cross validation algorithm

```
Data: N observations
Split the data in k groups ;
for i in k do
    Test the model in i and train in the rest data ;
    Measure the accuracy in each iteration ;
end
```

Note that could split the data.

1.5 sklearn KFold




```
import numpy as np
e = np.array(('a','b','c','d','e','o','p'))
Kfold = KFold(n_splits=3)
for itrain, itest in Kfold.split(e):
    print('train index', itrain)
    print('test index', itest)
    print('--')
# This return the indices we can attach to the index the X and the Y
  X.iloc[itrain]
  X.iloc[itest]
```

```
from sklearn import DecisionTreeClassifier
model = DecisionTreeClassifier(
criterion='entropy' # by default is gini.
max_depth=None #by default.
)
model.fit(X,y)
```

1.6 Methods

```
predict(X) #  Return the predicted class
predict_proba(X) # Return classes probabilities
score(X,y) where y is the true labels.
```

2 Prunning

2.1 Post pruning

2.1.1 Reduce error pruning

let the tree growth and after chop off,
reduce error pruning.

2.2 Cost complexity pruning

2.3 Weakest link pruning

3 AUC in decision tree

$D(n)$ number of leafs.

Total cost = Measure of fit + Measure of complexity = classification error + number of leaf

4 Search

Greedy algorithms suffer of horizon effect.

4.1 Assess the precision

5 Preprocessing DATA

```
sklearn.preprocessing.OrdinalEncoder # to encode X
sklearn.preprocessing.LabelEcoder() # to label y
```

6 Pruning Uppen

A node t and t_L and t_R left and right child nodes respectively. T represent all nodes, $\tilde{T}$ all leafs. a split is denoted by s the set of all splits S.

7 Impurity function

$|a|$ means the number or 'cardinal' elements that belong to the set a. (We can write with a indicator function).

8 HyperParamter

8.1 Gridsearch

8.2 Example one

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
df =datasets.load_iris(as_frame=True)
df = df['frame']
X = df.iloc[:,:-1]
y = df.iloc[:,[-1]]
y = y.astype('int')
clf = DecisionTreeClassifier()
fitM = clf.fit(X,y)
clf.score(X,y)
scores =[]
from sklearn.model_selection import KFold
kfolds = KFold(n_splits=5, shuffle=True)
for itrain, itest in kfolds.split(X):
    Xtrain, Xtest = X.iloc[itrain], X.iloc[itest]
    ytrain, ytest = y.iloc[itrain], y.iloc[itest]
    model = clf.fit(Xtrain,ytrain)
    score = accuracy_score(ytest,model.predict(Xtest))
    scores.append(score)
mean = np.mean(scores)
```

```
sd = np.std(scores)
print(mean, sd)
```

9 Confusion Matrix

9.1 Sensitivity and specificity

9.2 Multinomial distribution

Before we need specify the multinomial coefficient that is, $\binom{N}{n_1 \ldots n_k}$ where $n_1 + .. + n_k = N$, note also that $\binom{N}{n_1 \ldots n_k} = \frac{N!}{n_1! n_2! \ldots n_k!}$. Thus we need select n_1 objects from N, select n_2 from $N - n_1$, select n_3 from $(N - n_1 - n_2)$ and thus in the k selection then we have n_k from $(N - n_1 - n_2 - n_3 - \ldots - n_{k-1})$.

10 Minimal cost complexity

Prunning based in minimal cost complexity and weakest link. The problem reduced to minimize the following expression:

$$C_\alpha(T) = R(T) + \alpha|T| \tag{1}$$

where $R(T)$ is miss classification rate in training data and $|T|$ is the number of leaves in the tree. Note that if $\alpha = 0$ then the tree assign to each node a observation. Then we need find the optimize value of α.

Recursively you can uses minimal cost beginning with the last leaves and ascending evaluating (1).

Remember that is a trade off between complexity and accuracy.

for each α we need find the $T_\alpha \subset T_0$ that minimize the expression of cost complexity.

the value of *alpha*

---

for α **to** N **do**
  find $T \subset T_0$ that $\min C_\alpha(T)$;
  Split data in k folds
  for k *in* **do**
    make;

$$\arg\max_{x \in R} f(x) \tag{2}$$

in python or sklearn this effective alpha: https://www.programmersought.com/article/16766848143/ to select alpha among all

```
def alphaZ(Xtrain,ytrain,Xtest,ytest):
    clf = DecisionTreeClassifier(random_state=0)
    path = clf.cost_complexity_pruning_path(Xtrain, ytrain)
    ccp_alphas, impurities = path.ccp_alphas, path.impurities
    clfs = []
    for ccp_alpha in ccp_alphas:
        clf = DecisionTreeClassifier(random_state=0, ccp_alpha=ccp_alpha)
        clf.fit(X_train, y_train)
        clfs.append(clf)
    train_scores = [clf.score(Xtrain, ytrain) for clf in clfs]
    test_scores = [clf.score(Xtest, ytest) for clf in clfs]
    alpha , tscore = ccp_alphas[test_scores.index(max(test_scores))], max(test_scores)
    return alpha, tscore
```

10.1 Questions

How interpret machine learning models a category instead a numerical value?

how we can encode without order.

11 OneHotEncoding

how we can encode categorical variables? this is used to created dummy variables.

```
var1 = ['A','B','C','A','B','A']
var2 = ['real','bot','bot','bot','real','real']
df = pd.DataFrame({"var1":var1, "var2":var2})
for var in df.columns:
    if df[var].dtype=='object':
        df = pd.get_dummies(df,prefix=[var], columns = [var], drop_first=True)
```

12 LabelEncoder

According to the documentation this must be used to the outcome or y variable. due rank the input ant could alter the results.

12.1 Theil index

$$H(x) = p(x) \tag{3}$$

```
import numpy as np
import pandas as pd
pd.read_excel(df)
```

```
print('hello world')
```