Home Assignment 4
Machine Learning A

FEDERICO FIORIO

October 3, 2022

1 How to Split a Sample into Training and Test Sets (20 points)

$\{(S_1^{train}, S_1^{test}) - - > \hat{L}(\hat{h}_1^*, S_1^{test}) = \frac{1}{n_1}\sum_{i=1}^{n_1} l(\hat{h_1}^*(X_i), Y_i),$
$(S_2^{train}, S_2^{test}) - - > \hat{L}(\hat{h}_2^*, S_2^{test}) = \frac{1}{n_2}\sum_{i=1}^{n_2} l(\hat{h_2}^*(X_i), Y_i),$
...
$(S_m^{train}, S_m^{test}) - - > \hat{L}(\hat{h}_m^*, S_m^{test}) = \frac{1}{n_m}\sum_{i=1}^{n_m} l(\hat{h_m}^*(X_i), Y_i)\}$

in the case we had to the consider the singular train and test sets:

$\mathbb{P}(L(\hat{h}_1^*) \geq \hat{L}(\hat{h}_1^*, S_1^{test}) + \sqrt{\frac{\ln(\frac{1}{\delta})}{2n_1}}) \leq \delta$
$\mathbb{P}(L(\hat{h}_2^*) \geq \hat{L}(\hat{h}_2^*, S_2^{test}) + \sqrt{\frac{\ln(\frac{1}{\delta})}{2n_2}}) \leq \delta$
...
$\mathbb{P}(L(\hat{h}_m^*) \geq \hat{L}(\hat{h}_m^*, S_m^{test}) + \sqrt{\frac{\ln(\frac{1}{\delta})}{2n_m}}) \leq \delta$

But we are in the case in which we want to obtain the best split, so I need to split the confidence among the M cases:

$\mathbb{P}(L(\hat{h}_1^*) \geq \hat{L}(\hat{h}_1^*, S_1^{test}) + \sqrt{\frac{\ln(\frac{m}{\delta})}{2n_1}})$
$\mathbb{P}(L(\hat{h}_2^*) \geq \hat{L}(\hat{h}_2^*, S_2^{test}) + \sqrt{\frac{\ln(\frac{m}{\delta})}{2n_2}})$
...
$\mathrm{P}(\mathrm{L}(^*_m) \geq \hat{L}(\hat{h}_m^*, S_m^{test}) + \sqrt{\frac{\ln(\frac{m}{\delta})}{2n_m}})$

considering a union bound, the previous statement is $\leq than$:

$\mathbb{P}(L(\hat{h}_1^*) \geq \hat{L}(\hat{h}_1^*, S_1^{test}) + \sqrt{\frac{\ln(\frac{m}{\delta})}{2n_1}})$
$+$
$\mathbb{P}(L(\hat{h}_2^*) \geq \hat{L}(\hat{h}_2^*, S_2^{test}) + \sqrt{\frac{\ln(\frac{m}{\delta})}{2n_2}})$
$+$
...

$+$
$\mathrm{P}(\mathrm{L}(^*_m) \geq \hat{L}(\hat{h}^*_m, S^{test}_m) + \sqrt{\frac{\ln(\frac{m}{\delta})}{2n_m}})$

By applying Hoeffding, we get $\leq$ than:

$\sum_{h \in H} \frac{\delta}{m}$
$=$
δ

which is the same as doing:

$\mathbb{P}(L(\hat{h}^*_1) \leq \hat{L}(\hat{h}^*_1, S^{test}_1) + \sqrt{\frac{\ln(\frac{m}{\delta})}{2n_1}})$
$+$
$\mathbb{P}(L(\hat{h}^*_2) \leq \hat{L}(\hat{h}^*_2, S^{test}_2) + \sqrt{\frac{\ln(\frac{m}{\delta})}{2n_2}})$
$+$
$...$
$+$
$\mathrm{P}(\mathrm{L}(^*_m) \leq \hat{L}(\hat{h}^*_m, S^{test}_m) + \sqrt{\frac{\ln(\frac{m}{\delta})}{2n_m}})$

$\geq$ than:

$1 - \sum_{h \in H} \frac{\delta}{m}$
$=$
$1 - \delta$

2 From a lower bound on the expectation to a lower bound on the probability [Pedagogical question, but not for submission]

3 Early Stopping (30 points)

Question 1

a

Returning the last model should mean that the estimation in unbiased because we do not perform a selection of the models and so our last model could be the best one or the worst one, but it's not dependent on the validation set.

b

In this case we have a biased estimation because we are performing a selection, so the best model is biased towards the validation set, if we change data in the validation set it might not remain the best model.

c

Same reasoning as question b, we are performing a selection so the result is biased towards the validation set.

Question 2

first case, I select only 1 model in an unbiased way, no need to distribute confidence δ:
$\mathbb{P}(L(h_{100}) \leq \hat{L}(h_{100}, S_{val}) + \sqrt{\frac{\ln(\frac{1}{\delta})}{2n}}) \geq 1 - \delta$
second case, T hyp. to choose from:
$\mathbb{P}(L(h_{t^*}) \leq \hat{L}(h_{t^*}, S_{val}) + \sqrt{\frac{\ln(\frac{T}{\delta})}{2n}}) \geq 1 - \delta$
third case, every epoch gets a slice of the "apple" (prob. failure) equal to the series in the slides, applying Occam's razor and the series in the slides:
$\mathbb{P}(L(h_{t^*}) \leq \hat{L}(h_{t^*}, S_{val}) + \sqrt{\frac{\ln(\frac{t^*(t^*+1)}{\delta})}{2n}}) \geq 1 - \delta$

Question 3

$\mathbb{P}(L(h_{t^*}) \leq \hat{L}(h_{t^*}, S_{val}) + \sqrt{\frac{\ln(\frac{t^*(t^*+1)}{\delta})}{2n}}) \geq 1 - \delta$
considering only:
$\sqrt{\frac{\ln(\frac{t^*(t^*+1)}{\delta})}{2n}})$
$\sqrt{\frac{\ln(t^*(t^*+1))-\ln(\delta)}{2n}})$
$\sqrt{\frac{\ln(t^*)+\ln(t^*+1)-\ln(\delta)}{2n}})$

with $T_{MAX} >> e^n$ it makes no sense to keep training the model.
$>>$ means order of magnitude higher.
with T_{MAX} of that magnitude the bound gets higher than 1 and becomes useless

Question 4

With the series in the slides, the bound becomes:
$\mathbb{P}(L(h_{t^*}) \leq \hat{L}(h_{t^*}, S_{val}) + \sqrt{\frac{\ln(\frac{2^{t^*}}{\delta})}{2n}}) \geq 1 - \delta$
I can obtain T_{max} considering only :
$\sqrt{\frac{\ln(\frac{2^{T_{max}}}{\delta})}{2n}}$
$\sqrt{\frac{\ln(2^{T_{max}})-\ln(\delta)}{2n}}$
$\sqrt{T_{max}\frac{\ln(2)-\ln(\delta)}{2n}}$
for $T_{max} >> 2n$ it makes no sense to keep training.

Question 5

a

non adaptive bound:
$\mathbb{P}(L(h_t^*) \leq \hat{L}(h_t^*, S_{val}) + \sqrt{\frac{\ln(\frac{T}{\delta})}{2n}}) \geq 1 - \delta$
adaptive bound:
$\mathbb{P}(L(h_{t^*}) \leq \hat{L}(h_{t^*}, S_{val}) + \sqrt{\frac{\ln(\frac{t^*(t^*+1)}{\delta})}{2n}}) \geq 1 - \delta$
taking into consideration only the bound for the non adaptive training and the multiplication for $\sqrt{2}$:
$\sqrt{2}\hat{L}(h_t^*, S_{val}) + \sqrt{2}\sqrt{\frac{\ln(\frac{T}{\delta})}{2n}}$

the first part is greater than 1 so I've removed it

$\geq$

$\sqrt{\frac{2\ln(\frac{T}{\delta})}{2n}}$

$\geq$

$\sqrt{\frac{\ln(\frac{T^2}{\delta^2})}{2n}}$

transformation:

$\delta^2 - - > \delta * \frac{T}{T+1}$
$\geq$

$\sqrt{\frac{\ln\frac{T^2(T+1)}{\delta T}}{2n}}$

$\geq$

$\sqrt{\frac{\ln\frac{T(T+1)}{\delta}}{2n}}$

for what is assumed in the slides:

$\geq$

$\sqrt{\frac{\ln\frac{T^*(T^*+1)}{\delta}}{2n}}$

$\geq$

$\sqrt{\frac{\ln\frac{t^*(t^*+1)}{\delta}}{2n}}$

which is the same as the adaptive bound

b

NON ADAPTIVE BOUND:

$T_{MAX} = e^n$

$\delta \leq \frac{1}{2}$

using this one instead of e^n

$\delta e^n \leq e^n$

$\sqrt{\frac{\ln(\frac{\delta e^n}{\delta})}{2n}} =$

$\sqrt{\frac{\ln(e^n)}{2n}} =$

$\sqrt{\frac{n}{2n}} =$

$\sqrt{\frac{1}{2}}$

so for e^n or something bigger we get a value which is higher than $1/\sqrt{2}$

c

4 Random Forests (50 points)

4.1 Analyzing Satellite Data (35 points)

PROVIDING CODE:

```
In [4]: import numpy as np
        import pandas as pd

        df = pd.read_csv("landsat_train.csv",header=None)

In [5]: df
```

Out[5]:

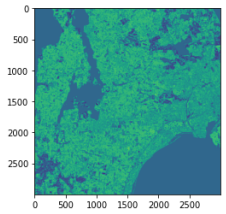
	0	1	2	3	4	5	6	7	8	9
0	6	2696	2416	2179	1412	3882	1988	1407	1025	4867
1	6	2691	2413	2177	1406	3879	1986	1408	1023	4856
2	6	2690	2413	2178	1404	3880	1988	1411	1023	4852
3	6	2691	2415	2184	1405	3878	1998	1420	1025	4851
4	6	2706	2428	2201	1413	3891	2021	1436	1033	4874
...	...	...	...	...	...	...	...	...	...	...
4999995	4	2541	2268	1885	2442	2200	2045	1071	1058	4574
4999996	4	2545	2271	1887	2443	2204	2048	1072	1059	4580
4999997	4	2548	2273	1890	2445	2206	2051	1073	1060	4586
4999998	4	2543	2270	1888	2441	2202	2050	1070	1058	4580
4999999	4	2540	2267	1885	2440	2199	2045	1068	1056	4575

5000000 rows × 10 columns

```
In [7]: labels = df.pop(0)

In [11]: #building classifier
         from sklearn.ensemble import RandomForestClassifier

         RF = RandomForestClassifier(n_estimators = 10, criterion= "gini",bootstrap=True, max_features= None)
         RF.fit(df, labels)

Out[11]: RandomForestClassifier(max_features=None, n_estimators=10)
         In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
         On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [15]: #training set accuracy
         y_pred = RF.predict(df)

         from sklearn import metrics
         print("training accuracy: ", metrics.accuracy_score(labels, y_pred)*100)

         training accuracy:  99.99538
```

Figure 1: first part of code

```
         y_pred = RF.predict(df)

         from sklearn import metrics
         print("training accuracy: ", metrics.accuracy_score(labels, y_pred)*100)

         training accuracy:  99.99588

In [6]: val = pd.read_csv("landsat_validation.csv",header=None)
        val_labels = val.pop(0)

In [7]: #validation set accuracy
        y_pred = RF.predict(val)

        from sklearn import metrics
        print("validation accuracy: ", metrics.accuracy_score(val_labels, y_pred)*100)

        validation accuracy:  75.15405545846755

In [8]: test = pd.read_csv("landsat_area.csv",header=None)

In [9]: pixel_predictions = np.array(RF.predict(test))

In [10]: image = pixel_predictions.reshape(3000,3000 )

In [17]: import matplotlib.pyplot as plt
         plt.imshow(image);
```




```
In [11]: pixel_predictions

Out[11]: array([4, 4, 4, ..., 4, 4, 4], dtype=int64)
```

Figure 2: second part of code

PROVIDING ANSWERS:

1)

compute the validation accuracy using the instances provided in landsat validation.csv.

What is the validation accuracy?
validation accuracy: 75.03013721605501

2)

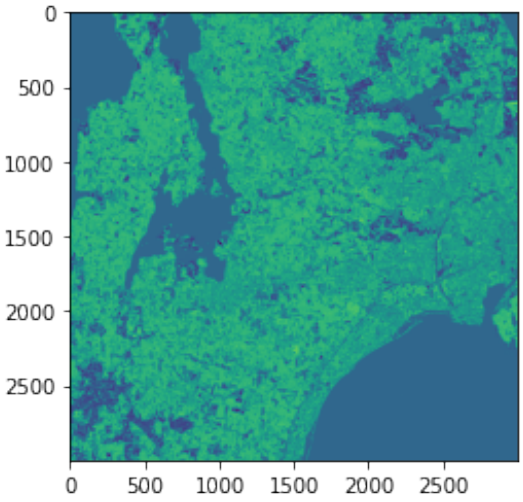


Figure 3: plots of the predictions

The area is Copenhagen!

3)

The height of the tree is based on n samples, so in the worst case, in order to classify n samples we will have a number n of leaves.
For a binary tree, in order to have n leaves you will have to build it with an height of n-1. Because in each step, since we are in the worst case scenario, as n grows also the number of leaves grows, and in order to have a number of leaves which is equal to n, the height of the tree must be adjusted and increased as n increases.

4.2 Normalization (15 points)

1)

Yes, Knn is affected by normalization, this happens because this algorithm is based on a distance measure.
A distance measure like the euclidean distance is affected by different scales of features, for example, if my data samples have 2 features, 1 of these features has a range between 2 and 3 and the other feature has a range of values between 4000 and 5000 then, the second feature will always have an higher weight compared to the first one and so the algorithm will be more biased towards this second one.
If instead we normalize the data properly, the two features will have same magnitudes and so the algorithm will not be biased towards one or the other feature.

2)

Tree based algorithms and random forests are not influenced by different scales of data,

mainly because they are based on a measure of impurity and not on a distance measure, so a choice in a node is taken independently from the data scales.
For example, if I have a sample X with 2 features one with range 1, 2 and the other with value range from 4000 and 5000 and I want to create a binary decision tree on this sample X, I will obtain something like:
if $X_1 \geq 1.2$ then classify as first class, if $X_1 < 1.2$ then refer to second feature.
the same happens for the second feature:
if $X_2 \geq 4300$ then classify as first class, if $X_2 < 4300$ then classify as second class.
Now if I want to normalize the data, I will still get the same decisions but instead based on different values based on the scale.
The decisions based on the second feature will become for example:
if $X_2 \geq 0.3$ then classify as first class, if $X_2 < 0.3$ then classify as second class.
But the meaning remain the same ann the algorithm will not be affected by the change.