CS185C:
Final Exam Notes

Brett Dispoto

May 7, 2020

1 Ensemble Learners

An ensemble learner is defined as:

$$F(x) =< \text{aggegrate function} > (\text{scoring functions...}) \tag{1}$$

1.1 Scoring Function

Where the scoring function is:

$$S(x, V, \Lambda) \tag{2}$$

Where...

1. x is the sample we wish to classify,
2. V is the training data,
3. Λ is a set of function parameters for the model. (initial weights, training param/hyperparam, etc)

1.2 Bagging

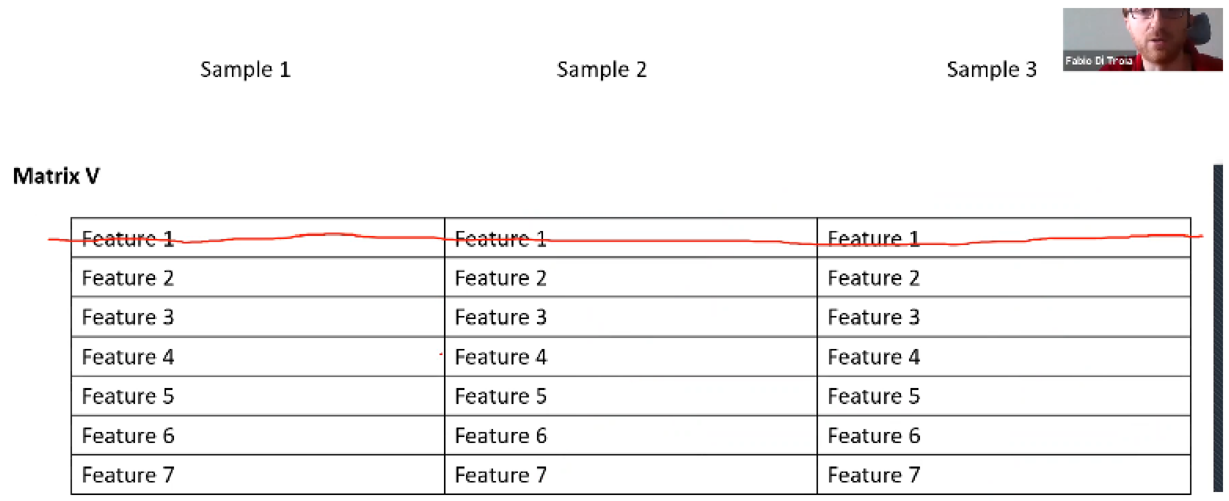
Also called **bootstrap aggegration**, bagging is where *different subsets of the data or features (or both)* are used to generate different scores.

1. The results are then combined in some way, such as sum, average, or max.
2. For bagging we assume that the same scoring model is used for all scores in the ensenble.
3. Bagging is meant to reduce overfitting.
4. Very similar to n-fold validation.
5. Good for small datasets, so we don't overfit on the data. 10 "bags" is probably a good initial choice.
6. If the aggregate function we use is average, then we're actual just doing n-fold validation.
7. However, we can still apply n-fold validation ON TOP OF bagging. (In fact, we should if possible).

For bagging the ensemble equation is:

$$F(S(x, V_1, \Lambda), S(x, V_2, \Lambda), ..., S(x, V_l, \Lambda), \tag{3}$$

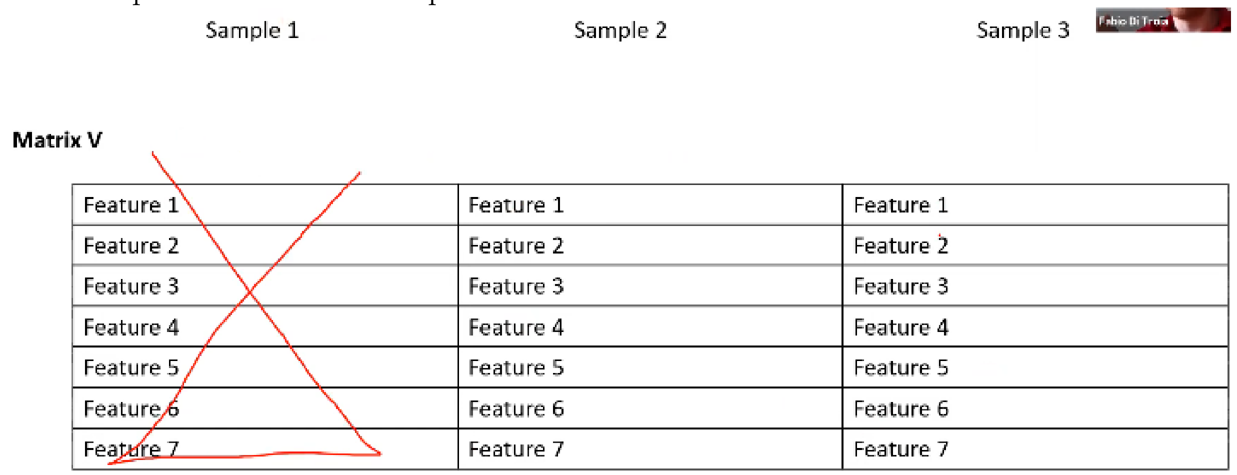
Note that Λ remains constant across all of our scoring fucntions. However, V_i is changing to specify that we are using differnt parts of the dataset. If we want to use diffent features... here's an exmaple for one of our models:



Matrix V

Sample 1	Sample 2	Sample 3
~~Feature 1~~	~~Feature 1~~	~~Feature 1~~
Feature 2	Feature 2	Feature 2
Feature 3	Feature 3	Feature 3
Feature 4	Feature 4	Feature 4
Feature 5	Feature 5	Feature 5
Feature 6	Feature 6	Feature 6
Feature 7	Feature 7	Feature 7

If we want to use different samples... here's an exmaple for one of our models:



Matrix V

Sample 1	Sample 2	Sample 3
Feature 1	Feature 1	Feature 1
Feature 2	Feature 2	Feature 2
Feature 3	Feature 3	Feature 3
Feature 4	Feature 4	Feature 4
Feature 5	Feature 5	Feature 5
Feature 6	Feature 6	Feature 6
Feature 7	Feature 7	Feature 7

More on bagging:

1. How to choose feature selection? Experiments.
2. We can also do subsets of both features and samples for a particular model.
3. This kind of bagging works well for **random forrests**.
4. We have to do an ensemble learner for **EACH** of the malware families. For example, for the WinWebSec malware family, we have multiple WinWebSec HMMs, which will be used in conjunctin to create the WinWebSec-Classifier, which is our "ensembled" WinWebSec HMMs.
5. For the this class, we should **NOT** split by features. It may require too much domain knowledge. Also, there aren't really enough features to do this anyway.

1.3 Boosting

Boosting is where we only differ the model, and not the data. We usually use "weak" classifiers. However, for our class, we only care about boosting *where the classifiers only differ in terms of their parameters* (So, we're not combining different types of models such as GAN, SVM, and HMM.

Voting workes better with boosting than with stacking. However, Fabio still recommends average or max in general.

We can also do random restarts, where we select the best model, then combine the results. HMM with random restarts can be seen as a special case of boosting, where we just pick the max.

1.4 Stacking

We can also use SVM as a meta-classifier, such that we no longer use of these aggegrate functions (max, average, etc..).

2 Steps for Final

Here are the steps to complete for the final. Crossed out means that it's finished.

1. ~~Make an HMM.~~
2. Download the dataset.
3. Add dataset to .gitignore
4. Split the dataset into different directories based on the malware family they belong to.
5. Do bagging. Create a file `Bagging.java`. Allow for command line arguments. The usage should be something like:

   `java Bagging <file_name>`

   where <file_name> is the name of the file we wish to classify.
6. Here is what Bagging.java should do if passed the command line argument "train":
   (a) Train n HMM's for *each* malware familty. Save the HMMs into directories called `$FAMILY.hmms/`.
   (b) Save the HMM's to a file on disk, so they can be reloaded for classiflying.
7. Here is what Bagging.java should do if passed a command line argument of a filename to classify.
   (a) For each malware family:
      i. Load the HMM's from the file,
      ii. Score the sample with each HMM,
      iii. Use some aggregate function (max, sum, average, etc..).
      iv. Write down the score for this malware family.
   (b) Finally, compare all of the scores and pick the highest. The group of HMM'swhich produced the highest score gives us the malware family this sample has been classified as.