Machine Learning and Computational Statistics, Fall 2014

Problem Set 4: VC dimension & Decision trees

Due: Friday, March 7, 2014 at 5pm (sent to jj1192@nyu.edu)

Instructions. *See problem set policy on the course web site.* You must show **all** of your work and be rigorous in your writeups to obtain full credit. Your answers to the below questions, including plots and all code that you write for this assignment, should be e-mailed as a PDF document to Mick Jermsurawong.

---

1. (15 points) **VC-dimension**

   (a) Show that the VC-dimension of a finite hypothesis set H is at most $\log_2 |H|$.

   (b) Consider a hypothesis class of binary classifiers given by decision trees where we use threshold splits (assume we have only real-valued features, not categorical). The trees have no depth limit, but we restrict each attribute to only be split once on any single path from a root to a leaf (note this is a big restriction!). Show that the VC-dimension is at least 2^d, where d is the dimension of a feature vector.

   (c) Consider the hypothesis class $\mathcal{H}_\alpha$ defined on the real line $x \in \mathbb{R}$ and parameterized by a single parameter α, given by $\mathcal{H}_\alpha = \{x : x \in [\alpha, \alpha+1] \text{ or } x \in [\alpha+2, +\infty]\}$. Show that the VC-dimension of $\mathcal{H}_\alpha$ is exactly 3.

   (Recall that to prove that the VC-dimension is equal to 3, you must (i) demonstrate a set of three points that are shattered by the hypothesis class, and (ii) demonstrate that any set of four or more points *cannot* be shattered by the hypothesis class.)

2. (15 points) **Decision Trees**

   We are writing a nature survival guide and need to provide some guidance about which mushrooms are poisonous and which are safe. (Caution - example only - do not eat any mushrooms based on this table.) We gather some examples of both types of mushroom, collected in a table, and decide to train a *binary* decision tree to classify them for us (two children per node, i.e., each decision chooses some variable to split on, and divides the data into two subsets). We have one real-valued feature (size) and two categorical features (spots and color). Recall that we do binary splits on a real-valued variable by finding the threshhold with the highest information gain (see lecture 12 slides).

y = Poisonous?	x_1 = size (real-valued)	x_2 = spots?	x_3 = color
N	1	N	White
N	5	N	White
N	2	Y	White
N	2	N	Brown
N	3	Y	Brown
N	4	N	White
N	1	N	Brown
Y	5	Y	White
Y	4	Y	Brown
Y	4	Y	Brown
Y	1	Y	White
Y	1	Y	Brown

Do this problem by hand and show all of your work. Your answer must be a **binary tree** *(any branch on x_1 must have an associated threshold).*

(a) What is the entropy of the target variable, "poisonous"?

(b) What is the first attribute a decision tree trained using the entropy or information gain method we discussed in class would use to classify the data?

(c) What is the information gain of this attribute?

(d) Draw the full decision tree learned from this data set (no pruning, no bound on its size).

(e) Now consider the following data, where we wish to predict the target variable Y. Suppose we train a decision tree (again using information gain, and again with no pruning or bound on size).

Y	A	B	C
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	0
0	0	1	1
1	0	1	1
0	1	0	0
1	1	0	1
1	1	1	0
0	1	1	1
1	1	1	1

What would be the *training* error of our classifier? Give as a percentage, and explain why. (*Hint: you can do this by inspection; there are no significant calculations required.*)

3. (15 points) **Decision Trees and Random Forests - Experiments with real data**

Adult (Census Income) Dataset from UCI Machine Learning Repository

After working through the toy example, let's work with a more realistic dataset. The Adult Dataset (Blake and Merz, 1998) was extracted from the 1994 Census database. The prediction task here is to determine whether a person makes over 50K a year. What makes this dataset interesting is: out of 48842 samples, it has about 24% positive (income >50K) and 76% negative (income <= 50K) samples; some of the features have missing values (marked by '?') and there are 6 continuous and 8 categorical features. These observations make it a good candidate for a Decision Tree based approach to classification.

You can refer to `http://archive.ics.uci.edu/ml/datasets/Adult` for more information on the dataset and attributes.

Dataset: *Please do not use the original dataset from the UCI repository.*

For this set of experiments, we have pre-processed the original dataset: removed two unused features (fnlwgt, education_num) and partitioned them into training and testing.

Data files should be downloaded from the course website. *http://cs.nyu.edu/~dsontag/courses/ml14/*

Software: We will be using Decision Tree and Random Forest implementations of the `python` based Machine Learning library `scikit-learn`.

`http://scikit-learn.org/stable/modules/tree.html#classification`

`http://scikit-learn.org/stable/modules/ensemble.html#random-forests`

If you are not comfortable with `python` or `scikit-learn`, you may use your favorite Machine Learning toolkit (`R, Matlab, Weka` etc.) with respective implementations of Decision Trees and Random Forests. Adventurous students are encouraged to implement their own version of the said classifiers.

Experiments:

(a) Handle missing values: The `scikit-learn` implementation of Decision Trees does not support missing values. In this step we will use a very simple method to fill in the missing values. For continuous values - you can substitute with the `mean` or `median` value of the particular feature, for categorical values - you can substitute with the `mode`. Calculation of these statistics should be done only on training data.

You can skip this step if you choose to work with some other Decision Tree implementation which can handle missing values internally.

(b) More pre-processing: Another limitation of `scikit-learn's DecisionTreeClassifier` and `RandomTreeClassifier` is that they do not accept non-numeric values. That is, the categorical features cannot be used as is. (The same problem would arise had you wanted to use a SVM for this classification problem.)

Again, you may skip this step if your chosen Decision Tree implementation accepts non-numeric and hence categorical features.

One way to overcome this is to transform the feature space, making one binary valued feature out of each value of the categorical features, while keeping the numeric features intact. Pseudocode goes something like this:

```
INITIALIZE new_features = []
FOR EACH feat IN original_features:
    IF feat.type == CONTINUOUS or NUMERIC:
        new_features.append(feat.name)
    IF feat.type == CATEGORICAL:
        FOR each feat.value:
            new_feat_name=feat.name + SOME_SEPARATOR + feat.value
            new_features.append(new_feat_name)
```

When transforming the original data record, numeric (i.e. continuous) features remain unchanged, and each of the binary valued features that replace the categorical features is set to 1 or 0 depending on whether the original categorical feature takes the relevant value.

Finally, randomly assign the data points to the training set (70%) and validation set (30%).

(c) Tune Decision Tree classifier: As mentioned in class, full depth Decision Trees are prone to overfitting. One way to address this is via pruning, but `scikit-learn` does not support pruning. Instead, it has two parameters that you can tune: `max_depth`, which limits the depth of the decision tree, and `min_samples_leaf`, which requires that every leaf has at least this many data points.

Generate two plots where on the X-axis you vary one of the parameters (see below) and on the y-axis you show classification accuracy. Each plot should have two lines on it, one for accuracy on `training_set` and another for accuracy on `validation_set`:

1. `max_depth = [1,2,3...30]`
2. `min_samples_leaf = [1,2,3...50]`

Save your best choices of parameters.

Create a visualization of the top 3 levels of your Decision Tree obtained using the optimal `max_depth and min_samples_leaf` found above.. In `scikit-learn` you can use `pydot` which uses `graphviz` binaries; refer the tutorial for example `http://scikit-learn.org/stable/modules/tree.html`.

Optional: You can try other parameters and their combinations to find the best performing classifier on the validation-set.

(d) Tune Random Forest Classifier: One of the parameters to tune the Random Forest Classifier is the number of trees in the ensemble (`n_estimators` in `scikit-learn`). As before, generate two plots with classification accuracy (Y-Axis) of `training_set` and `validation_set` vs. (X-Axis):

1. `n_estimators = [1,2,3...50]`, with all other parameters set to default.
2. `n_estimators = [1,2,3...50]`, with `max_depth` and `min_samples_leaf` chosen using your best results from question (c).

(e) Evaluate on test data: Train your best configuration of `DecisionTreeClassifier` and `RandomForestClassifier` (both) on the full training data, and report their performance on the test dataset.

(f) *Optional Experiments*:

i) If you find some interesting visualizations or metrics of performance, feel free to attach them.

ii) Training Random Forests without a validation set? How many trees are enough?

For Random Forests you don't need to keep aside a validation set to get an unbiased estimate of the test set error (this is only for Random Forests, which using bagging, not for Decision Trees). Instead, one can use the out of bag error estimate (see `http://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm`). Note also that with ensemble methods you can add as many trees as you can afford computationally, without worrying about overfitting.