CECS 451: Artificial Intelligence
Fall 2017, Semester Project

Project Ideas

Wenlu Zhang, Last Updated: October 11, 2017

Timeline:

1. Project ideas are out. 10/11/2017
2. Zhang will talk briefly about the project during class. 10/13/2017
3. You can choose any one from the list or do your own project related with AI. If you choose your own one, please talk to me and bring specific agenda to get approved. 10/11/2017-10/20/2017
4. Group members finalized (At most 3 members each group). 10/20/2017
5. One page for project progress (including dataset, methodology or some results). 11/17/2017
6. Project demo or presentation (slides only required for paper presentation groups).

Idea 1:
Search

Propose methods to improve any search algorithms in AIMA, need theoretical proof

Idea 2,3:
Game Theory

Implement a game to compete with human players such as chess board etc.

Propose methods to improve any algorithms from Chapter 5 in AIMA, need theoretical proof

Idea 4:
Decision Trees and Random Forests
Adult (Census Income) Dataset from UCI Machine Learning Repository

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 this link.

Experiments:

1. Handle 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.

2. More pre-processing: Please handle the categorical features since they are non-numeric values. 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. 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 features takes the relevant value. Finally, randomly assign the data points to the training set (70%) and validation set (30%).

3. Evaluate on test data: Train your best configuration of DecisionTree Classifier and RandomForest Classifier (both) on the full training data, and report their performance on the test dataset.

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

5. (Optional Experiments) 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 Tree.) 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.

Idea 5:
Food Clustering
US Department of Agriculture Nutrient Database

Look at the USDA nutrient database, one would find various types of foods with their corresponding nutritional contents: http://ndb.nal.usda.gov/ndb/search/list. In this project, we will experiment with 4 food groups: Cereal-Grain-Pasta, Finfish-Shellfish, Vegetables, Fats-Oils. The data contains detailed categorizations of each food item. For example, there are 9 categories of Kale. They range from raw, frozen and unprepared, cooked and boiled, cooked and drained without salt, etc. In addition, common knowledge suggests that major food groups can be further categorized. Vegetables can be leaves/stems, roots, or buds. Based on their nutritional contents, one might expect to see these items clustered in hierarchies, from the major food groups, sub-groups, and finally to variants of the same food items.

Dataset:
The following data files are provided with this project and can be downloaded from this link:

1. dataDescriptions.txt - gives the names of all the attributes (nutrients).
2. dataCereal-grains-pasta.txt, dataFinfish-shellfish.txt, dataVegetables.txt, dataFats-oils.txt The data files, one for each food group.

Please note that the attributes in the data files are delimited by the caret ^ character.

Experiments:
In this project, we will explore Principal Components Analysis (PCA). In addition, we will explore unsupervised learning with K-means and Hierarchical clustering.

1. Numerical values vary widely across different types of nutrients. However, small numerical values in some micro-nutrients such as minerals and vitamins may characterize the food items as well as larger numerical values in macro-nutrients like protein and carbohydrates. Therefore it becomes important to normalize the nutrient values. For this project, transform the features to be in the range [0,1]. E.g., for the j^{th} dimension, the value of the i^{th} data point will become:

$$Normalized(X_{ij}) = \frac{X_{ij} - min(X.j)}{max(X_{.j}) - min(X_{.j})} \quad (1)$$

Where min and max for $X_{.j}$ are calculated over the j^{th} attribute/dimension on the complete dataset.

2. Let's begin PCA analysis. Visualize the data by reconstructing using only the first 2 principal components. Examine the first and second principal components to find their five highest absolute weights an their corresponding nutrients.

3. Apply K-means clustering on the original data, with different sizes of cluster k = [4, 6, 8, 10, 12]. Visualize the result on the reduced dimensions (2-D). Use a different color for each cluster. And explain your results.

4. Next, we want to visualize the possible structures of food using hierarchical clustering and dendrograms. Randomly select 30 food items from each of the 4 food groups. Create a dendrogram and review the labels of the food items from the dendrograms. Do the food items cluster into 4 groups as expected? Identify any distinct clusters and the corresponding food items. Finally, obtain actual clusters from hierarchical clustering.

5. We now want to examine sub-clusters within a food group. Let's take the Cereal-Grain-Pasta food group. Apply K-means using number of clusters k = [5, 10, 25, 50, 75]. Report the largest cluster size for each k and display 10 randomly sampled food items from each cluster. If the largest cluster size is less than 10, display all the food items in that cluster. As k increases, comment on the size of the largest cluster and the uniformity of food items in it.

Idea 6:
E-mail spam classification

The dataset included for this project is based on a subset of the SpamAssassin Public Corpus. Upper image of Figure 1 shows a sample email that contains a URL, an email address (at the end), numbers, and dollar amounts. While many emails would contain similar types of entities (e.g., numbers, other URLs, or other email addresses), the specific entities (e.g., the specific URL or specific dollar amount) will be different in almost every email. Therefore, one method often employed in processing emails is to "normalize' these values', so that all URLs are treated the same, all numbers are treated the same, etc. For example, we could replace each URL in the email with the unique string "httpaddr" to indicate that a URL was present. This has the effect of letting the spam classifier make a classification decision based on whether any URL was present, rather than whether a specific URL was present. This typically improves the performance of a spam classifier, since spammers often randomize the URLs, and thus the odds of seeing any particular URL again in a new piece of spam is very small.

We have already implemented the following email preprocessing steps: lowercasing; removal of HTML tags; normalization of URLs, email addresses, and numbers. In addition, words are reduced to their stemmed form. For example, "discount", "discounts", "discounted" and "discounting" are all replaced with "discount". Finally, we removed all non-words and punctuation. The result of these preprocessing steps is shown in lower image of Figure 1.

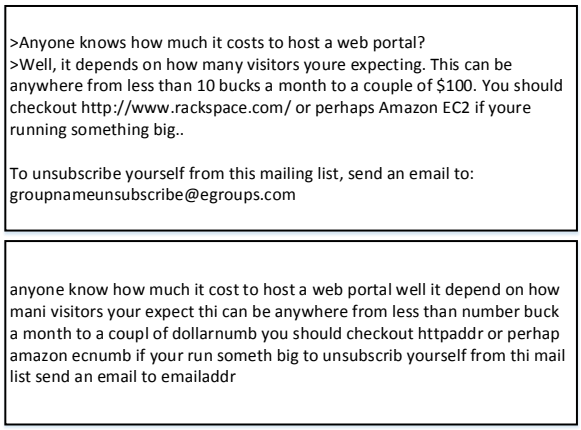

Figure 1: Upper image shows: Sample email in SpamAssassin corpus before pre-processing. Lower image shows: Pre-processed version of the sample email from upper one.

Dataset:
We have provided you with two files: spam_train.txt, spam_test.txt. Each row of the data files corresponds to a single email. The data can be downloaded from this link. The first column gives the label (1=spam,0=not spam).

Experiments:

1. This project will involve your implementing classification algorithms. Before you can build these models and measures their performance, split your training data (i.e. spam_train.txt) into a training and validate set, putting the last 1000 emails into the validation set. Thus, you will have a new training set with 4000 emails and a validation set with 1000 emails. Explain why measuring the performance of your final classifier would be problematic had you not created this validation set.

2. Transform all of the data into feature vectors. Build a vocabulary list using only the 4000 email training set by finding all words that occur across the training set. Note that we assume that the data in the validation and test

sets is completely unseen when we train our model, and thus we do not use any information contained in them. Ignore all words that appear in fewer than X = 30 emails of the 4000 email training set. This is both a means of preventing overfitting and of improving scalability. For each email, transform it into a feature vector x where the ith entry, x_i, is 1 if the ith word in the vocabulary occurs in the email, and 0 otherwise.

3. Train the linear classifier such as Naive Bayes using your training set. How many mistakes are made before the algorithm terminates? Next, classify the emails in your validation set. What is the validation error? Explain your results.

4. Explore some other algorithms to solve spam filter problem. And demonstrate your thoughts.

Idea 7:
Paper Presentation

Read at least 2 papers about a specific topic and give 15-20 mins brief introduction presentation with slides.

Idea 8:
Deep Learning in Digit Recognizer (extra 5 points)

MNIST ("Modified National Institute of Standards and Technology") is the "hello world" dataset of computer vision. Since its release in 1999, this classic dataset of handwritten images has served as the basis for benchmarking classification algorithms. Goal is to correctly identify digits from a dataset of tens of thousands of handwritten images in any deep learning algorithms. (GPU needed)

Dataset:

We have provided you with two files: train.csv, test.csv. The data can be downloaded from this link. The link also includes short description of data set.

Idea 9:
Deep Learning in Object Recognition (extra 5 points)

CIFAR-10 is an established computer-vision dataset used for object recognition. It is a subset of the 80 million tiny images dataset and consists of 60,000 32×32 color images containing one of 10 object classes, with 6000 images per class. It was collected by Alex Krizhevsky, Vinod Nair and Geoffrey Hinton. Goal is to

identify the subject of 60,000 labeled image in any deep learning algorithms. (GPU needed)