CSE 190, Fall 2015: Homework 1

Instructions

Please submit your solution **at the beginning of the week 3 lecture (October 12)** or outside of CSE 4102 beforehand. Please complete homework **individually**.

You will need the following files:

50,000 beer reviews : `http://jmcauley.ucsd.edu/cse190/data/beer/beer_50000.json`. You may also use the non-alcoholic beer data if you prefer, though please mention it if you did: `http://jmcauley.ucsd.edu/cse190/data/beer/non-alcoholic-beer.json`

Book descriptions : `http://jmcauley.ucsd.edu/cse190/data/amazon/book_descriptions_50000.json`

Code examples : `http://jmcauley.ucsd.edu/cse190/code/week1.py` (regression) and `http://jmcauley.ucsd.edu/cse190/code/week2.py` (classification)

Executing the code requires a working install of Python 2.7 with the scipy packages installed.

Tasks — Regression (week 1):

1. Compute the following statistics about the data: (a) number of unique items ('beer/beerId'), (b) number of unique users ('user/profileName'), (c) mean for each of the five ratings ('review/appearance', 'review/palate', 'review/overall', 'review/aroma', 'review/taste'), (d) mean ABV ('beer/ABV') (1 mark).

2. Using ordinary linear regression, train a predictor that uses the ABV ('beer/ABV') to predict the taste rating ('review/taste'), i.e.,

$$\text{review/taste} \simeq \theta_0 + \theta_1 \times \text{beer/ABV}.$$

   You may use Python libraries to do so. What are the fitted values of θ_0 and θ_1 (1 mark)?

3. The above regressor may not be very realistic—it assumes that beers get monotonically better or monotonically worse as ABV increases. Perhaps we can do better with a polynomial function, i.e.,

$$\text{review/taste} \simeq \theta_0 + \theta_1 \times \text{beer/ABV} + \theta_2 \times \text{beer/ABV}^2 + \theta_3 \times \text{beer/ABV}^3 \dots$$

   Write down the fitted values for all polynomials up to degree 5, and their Mean Squared Errors (1 mark).

4. If we kept fitting higher and higher degree polynomials, the model will have lower and lower error. But will it generalize well to *new data*? To test this, split the data into 50% train and 50% test sets as follows:

```
train = data[:25000]
test = data[25000:]
```

   Now, fit the model only on the *training* data, and report the MSE on both the training and test sets. Do this for *all polynomial degrees until the performance no longer improves on the test set.* Write down the polynomial equation corresponding to the best model, and its performance on the test set (2 marks).

Classification (week 2):

1. Download the book descriptions data. For this and the next question we will consider identifying "Children's Books" based on words in their descriptions. In class we had trouble when there was 'imbalance' between positive and negative labels. To address this, select all children's books and *the same number of non-children's books* as follows:

```
D_child = [d for d in data if "Children's Books" in d['categories']]
D_notchild =\
  [d for d in data if not("Children's Books" in d['categories'])][:len(D_child)]

data = D_child + D_notchild
random.seed(0)
random.shuffle(data)
```

Split the data so that the first half is used for training and the second half is used for testing as above. First, let's use the following feature vector to train an SVM:

```
X = [[1, "child" in s['description'],
         "magic" in s['description'],
         "funny" in s['description']] for s in data]
```

Using this feature vector, run an SVM classifier (see the code provided in class) – remember to train on the first half and test on the second half. What is the accuracy (percentage of correct classifications) of the predictor on the train and test data (1 mark)?

2. Can you come up with a better predictor (on the test set)? Write down a feature vector with *at most 10* dimensions that has better performance than the one above, and write down its test error (1 mark).

3. Next we'll try and get better performance with the predictor above by using a validation set. Split your data into training, validation, and test sets as follows:

   ```
   X_train = X[:len(X)/2]
   X_valid = X[len(X)/2:3*len(X)/4]
   X_test = X[3*len(X)/4:]
   ```

   The paramater C used to train the SVM controls the regularization level. Run the SVM for all values of $C \in \{0.001, 0.01, 0.1, 1, 10, 100, 1000\}$, and report their training, validation, and test errors (you may use either the feature vector from the first part of the question, or your own feature vector above, but please specify which you used). Which of these test errors best reflects the model's ability to generalize to new data (1 mark)?