Applied Machine Learning HW1 (15%)

Due ~~Tuesday April 24~~ **Saturday April 28** @ 11:59pm on Canvas; **minor updates in red.**

Instructions:

1. This HW should be done in Python and `numpy` is highly recommended. See the course homepage for a tutorial. Either Python 2 or 3 is fine, though the latter is preferred. If you don't have a Python+numpy installation yourself, you can `ssh access.engr.oregonstate.edu`.
2. Do not use machine learning packages such as `sklearn`, though you can use them to verify your results.
3. Do not use data analysis packages such as `pandas` or `seaborn`. Your code should only depend on standard built-in packages plus `numpy`.
4. Download your HW1 data from the course homepage. It contains the training data, the dev set (held-out), and a semi-blind test set. The test set does not contain target labels, and you will need to predict them using your best model. Part of your grade is based on your prediction accuracy on test.
5. You should submit a single `.zip` file containing `hw1-report.pdf`, `income.test.predicted`, and all your code. Again, LaTeX'ing is recommended but not required.

1 Data (Pre-)Processing (Feature Map)

1. Take a look at the data. A training example looks like this:

   `37, Private, Bachelors, Separated, Other-service, White, Male, 70, England, <=50K`

   which includes the following 9 input fields plus one output field (y):

   age, sector, education, marital-Status, occupation, race, sex, hours-per-week, country-of-origin, target

   Q: What are the positive % of training data? What about the dev set? Does it make sense given your knowledge of average per capita income in the US?

2. The default preprocessing method is to binarize all fields, e.g., *age* becomes many binary features such as `age=1`, `age=2`, etc., and *race* becomes `race=White`, `race=Asian-Pac-Islander`, etc. These resulting features are all binary, meaning their values can only be 0 or 1, and for each example, in each field, there is one and only one positive feature.

   Q: Why do we binarize all fields? In particular, why do we need to binarize the two already numerical fields, *age* and *hours-per-week*?

3. Q: Based on your common sense intuition, what's the most indicative positive feature? Hint: `education=...`
4. Q: What are the youngest and oldest ages in the training set? What are the least and most amounts of hours per week do people in this set work? Hint:

   ```
   cat income.train.txt.5k | sort -nk1 | head -1
   ```

5. Q: Do we need to consider features that do not appear in the training set? E.g., `age=1` does not appear because this is the "adult-income" data set. What if a feature appears on test but not on training data?
6. Q: How many features do you have in total (i.e., the dimensionality)? Do not forget the bias dimension. Hint: should be around 230. How many features do you allocate for each field of the data?

2 Perceptron and Averaged Perceptron

1. Implement the basic perceptron algorithm. Run it on the training data for 5 epochs (each epoch is cycle over the whole training data, also known as an "iteration"). After each epoch, **print the number of updates (i.e., mistakes) in this epoch (and update %, i.e., #_of_updates /$|D|$),** and evaluate on the dev set and report the error rate and predicted positive rate on dev set, e.g., something like:

   ```
   epoch 1 updates 1257 (25.1%) dev_err 23.8% (+:27.5%)
   ...
   epoch 5 updates 1172 (23.4%) dev_err 20.5% (+:17.7%)
   ```

   Q: what's your best error rate on dev, and at which epoch did you get it? (Hint: should be around 19%).

2. Implement the averaged perceptron. You can use either the naive version or the smart version (see slides). Run the averaged perceptron on the training data for 5 epochs, and again report the error rate and predicted positive rate on dev set (same as above).

   Q: This time, what's your best error rate on dev, and at which epoch did you get it? (Hint: around 15%).

3. Q: What observations can you draw by comparing the per-epoch results of standard and averaged perceptrons? What about their best results?

4. Q: For the averaged perceptron, what are the five most positive/negative features? Do they make sense?

5. Q: We know that males are more likely to earn `>50K` on this dataset. But the weights for `Sex=Male` and `Sex=Female` are both negative. Why?

6. Q: What is the feature weight of the bias dimension? Does it make sense?

7. **Q: Is the update % above equivalent to "training error"?**

3 Experimentations

1. Try this experiment: reorder the training data so that all positive examples come first, followed by all negative ones. Did your (vanilla and averaged) perceptron(s) degrade in terms of dev error rate(s)?

2. Try the following feature engineering:

   (a) centering of each dimension to be zero mean;
   (b) based on the above, and make each dimension unit variance.
   (c) adding some combination features (e..g, edu=X and sector=Y)

   Q: which ones (or combinations) helped? did you get a new best error rate?

3. Collect your best model and predict on `income.test.blind` to `income.test.predicted`. The latter should be similar to the former except that the target (`>=50K, <50K`) field is added. Do not change the order of examples in these files.

   Q: what's your best error rate on dev? which setting achieved it?

   Q: what's your predicted positive % on dev? how does it compare with the ground truth positive %?

   Q: what's your predicted positive % on test?

Debriefing (required):

1. Approximately how many hours did you spend on this assignment?
2. Would you rate it as easy, moderate, or difficult?
3. Did you work on it mostly alone, or mostly with other people?
4. How deeply do you feel you understand the material it covers (0%–100%)?
5. Any other comments?