Machine Learning HW1 (10%, in groups of 2 or 3)

Instructor: Liang Huang

Due Wed Oct 11 @ 11:59pm

Instructions:

1. Singleton groups are not allowed. Please reply to the Canvas discussion thread to find teammates.
2. Before you submit, please join your group on Canvas under `People / HW1 Groups`. If any group member submits, it's automatically considered submission for his/her group.
3. This HW should be done in Python, with `numpy` being highly recommended though not required.
4. Do not use machine learning packages such as `sklearn`, though you can use them to verify your results.
5. Do not use data analysis packages such as `pandas` or `seaborn`. Your code should only depend on standard built-in packages plus `numpy`.
6. 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 the relative ranking of your test performance.
7. You should submit a single `.zip` file containing `hw1.pdf`, `income.test.predicted`, and all your code. Again, LaTeX'ing is recommended but not required.

1 Perceptron and Averaged Perceptron

1. Implement the basic perceptron algorithm with all features being binarized.

   Q: How many features do you have (i.e., the dimensionality)? Do not forget the bias dimension.

2. Run perceptron on the training data for 5 epochs (also known as "iterations"). For every 1,000 training examples, evaluate on the dev set and report the dev set error rate (e.g., 16.71%).

   Q: what's your best error rate on dev, and where do you get it? (e.g., at epoch 4.81)

3. Implement the averaged perceptron, in both the naive way and the smart way (see slides).

   Q: do you see any difference in speed between the two ways? Measure the time using `time.time()`.

4. Run the averaged perceptron on the training data for 5 epochs.

   Q: this time, what's your best error rate on dev, and where do you get it? (e.g., at epoch 3.27)

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

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

6. Plot the dev error rates for both vanilla and averaged perceptrons for the first epoch. x-axis: epoch ([0:1]), y-axis: dev error rate. Plotting frequency: every 200 training examples.

   Q: what do you observe from this plot?

   Note: you can use `gnuplot` or `matplotlib.pyplot` to make this plot, but not Excel or Matlab.

2 MIRA and Aggressive MIRA

1. Implement the default (non-aggressive) MIRA, and its averaged version. Run them for 5 epoch on the training data, still with an evaluation frequency of 1,000 training examples.

   Q: what are the best error rates on dev (for MIRA and avg. MIRA, res.), and where do you get them?

2. Implement the aggressive version of MIRA, and test the following p (aggressivity threshold): 0.1, 0.5, 0.9.

   Q: what are the best error rates on dev (for {unavg, avg} $\times$ {0.1, 0.5, 0.9}), and where do you get them?

3. Q: what do you observe from these experiments? Also compare them with the perceptron ones.

3 Experimentations

Try the following:

1. Reorder the training data so that positive examples all come first, followed by all negative ones.

   Q: did your perceptron/MIRA algorithms degrade on dev error rate?

   Q: what if you shuffle the data before training?

   Q: can you explain this mystery, i.e., why a randomized order is much better? show a toy example?

2. Try some feature engineering, including but not limited to:

   (a) replacing the binarized numerical features (age, hours) by the original numbers.

   (b) adding the numerical features besides the binarized ones.

   (c) adding binned (i.e., quantized) numerical features.

   (d) adding a numerical feature `education-level`.

   (e) adding some combination features.

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

3. Try some algorithmic engineering, including but not limited to:

   (a) variable learning rate

   (b) centering of each dimension to be zero mean and unit variance.

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

4. Collect your best model and predict on `income.test.txt` 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, and which algorithm (and settings) achieved it?

   Q: what's your % of positive examples on test, and how does it compare to those on train and dev?