

# Applied Machine Learning: HW2 (15%)

Due Saturday April 27 @ 11:59pm on Canvas

Instructions: same as HW1.

## 1 Feature Map

1. In HW1, we binarized all categorical fields and kept the numerical fields, which makes sense for distance metrics. But for perceptron (and most other classifiers), it's actually better to binarize the numerical fields as well (so that "age" becomes many binary features such as "age=40"). Why?
2. How many binary features do you get in total after binarizing all fields? (Hint: around 230; Recall that HW1 had about 90 features). Again:

```
for i in `seq 1 9`; do cat income.train.txt.5k | cut -f $i -d ',' | sort | uniq | wc -l; done
```

Do not forget the bias dimension.

## 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).  
Note: the difference in speed between the two versions is not that big, as we only have around 230 features, but in HW5 the difference will be huge (we'll have sparse features). See also the last question in EX.

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 Comparing Perceptron with $k$ -NN

1. What are the major advantages of perceptron over  $k$ -NN?
2. Design and execute an experiment to demonstrate your point.

### 4 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) adding the original numerical features (age, hours) as two extra, real-valued, features.
  - (b) centering of each numerical dimension to be zero mean;
  - (c) based on the above, and make each numerical dimension unit variance.
  - (d) adding some binary 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 ( $\geq 50K$ ,  $< 50K$ ) 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?