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?
   In case of *k*-NN, binarizing numerical fields was not encouraged because numbers allowed meaningful interpretation of distance between two records of the data-set.
   Whereas in case of perceptron, predictions of the target variable is not being made on basis of distance or measure of similarity between two records. Rather, the prediction is being made on basis of weights allocated to different features of the data-set.
   Further, *age=40* is a feature that has some weight allocated to it as any other age over proximity of the values between *age=40* and any other age which helps in predicting the target variable.

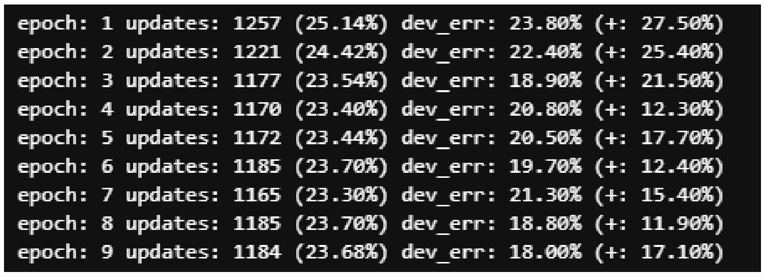
2. How many binary features do you get in total after binarizing all fields? (Hint: around 230; Recall that HW1 had about 90 features). Again:
   231 features with bias by binarizing all fields.

   ```
   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 (25.14%) dev_err: 23.80% (+: 27.50%)
epoch: 2 updates: 1221 (24.42%) dev_err: 22.40% (+: 25.40%)
epoch: 3 updates: 1177 (23.54%) dev_err: 18.90% (+: 21.50%)
epoch: 4 updates: 1170 (23.40%) dev_err: 20.80% (+: 12.30%)
epoch: 5 updates: 1172 (23.44%) dev_err: 20.50% (+: 17.70%)
epoch: 6 updates: 1185 (23.70%) dev_err: 19.70% (+: 12.40%)
epoch: 7 updates: 1165 (23.30%) dev_err: 21.30% (+: 15.40%)
epoch: 8 updates: 1185 (23.70%) dev_err: 18.80% (+: 11.90%)
epoch: 9 updates: 1184 (23.68%) dev_err: 18.00% (+: 17.10%)
```

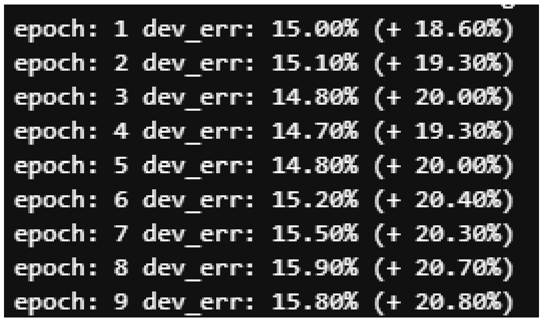
   Q: what's your best error rate on dev, and at which epoch did you get it? (Hint: should be around 19%).
   Best error rate - 18.00% at epoch 9 Best error rate - 18.90% at epoch 3 (Considering only first 5 epochs)

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%).
   Best error rate on dev: 14.7% on epoch 4



```
epoch: 1 dev_err: 15.00% (+ 18.60%)
epoch: 2 dev_err: 15.10% (+ 19.30%)
epoch: 3 dev_err: 14.80% (+ 20.00%)
epoch: 4 dev_err: 14.70% (+ 19.30%)
epoch: 5 dev_err: 14.80% (+ 20.00%)
epoch: 6 dev_err: 15.20% (+ 20.40%)
epoch: 7 dev_err: 15.50% (+ 20.30%)
epoch: 8 dev_err: 15.90% (+ 20.70%)
epoch: 9 dev_err: 15.80% (+ 20.80%)
```

3. Q: What observations can you draw by comparing the per-epoch results of standard and averaged perceptrons? What about their best results? **Better error-rates per-epoch are achieved by using Average perceptron. Best results are obtained at nearly same epoch number.**

4. Q: For the averaged perceptron, what are the five most positive/negative features? Do they make sense?
   Top 5 most positive features - Education=Doctorate, MaritalStatus=Married-with-spouse, Education=Prof-school, NumberOfHoursWorkedPerWeek=65, Country=Iran
   Top 5 most negative features - Age=28, Education=7th-8th, Occupation=Farming-fishing, Age=26, NumberOfHoursWorkedPerWeek=24
   They do make sense because based on initial analysis it's been found that people from Iran are found to have Doctorate degrees and those with Doctorate degrees tend to earn more than 50K.

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?
   This might be because sex=male or sex=female is not as important a feature as others like country or education to predict income of a new example.

6. Q: What is the feature weight of the bias dimension? Does it make sense?
   Feature weight for bias dimension is -114483.

7. Q: Is the update % above equivalent to "training error"?
   No. Update % refers to the number of updates made in each epoch. Whereas training error refers to the misclassification of training examples by the model.

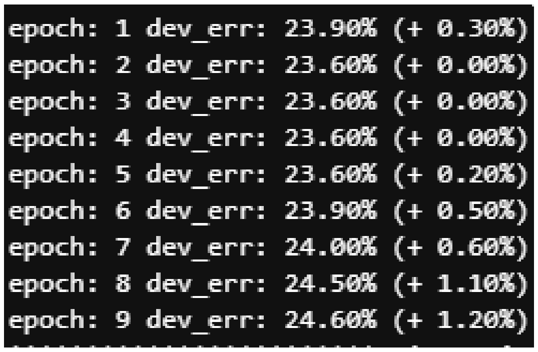
3 Comparing Perceptron with k-NN

1. What are the major advantages of perceptron over k-NN?
   1) k-nn does not have weights involved which doesn't help us understand the reasons behind a given prediction.
   2) In k-nn, prediction is very time-consuming because it involves calculating similarity of each test example with respect to all the training examples, sorting them and then taking the top k neighbors. Whereas in perceptron, once the model having weights for features has been developed during training, predictions for test example is very quick.

2. Design and execute an experiment to demonstrate your point.
A simple experiment to illustrate advantage of perceptron over k-nn is comparing the time taken to determine error-rate for dev set. In case of k-nn, time taken to compute error rate for dev set having 1000 examples with 92 features was easily over 10 seconds whereas time taken to compute error rate for same dev set containing even 231 features was less than 5 seconds while running the perceptron algorithm for 10 epochs.

4 Experimentations

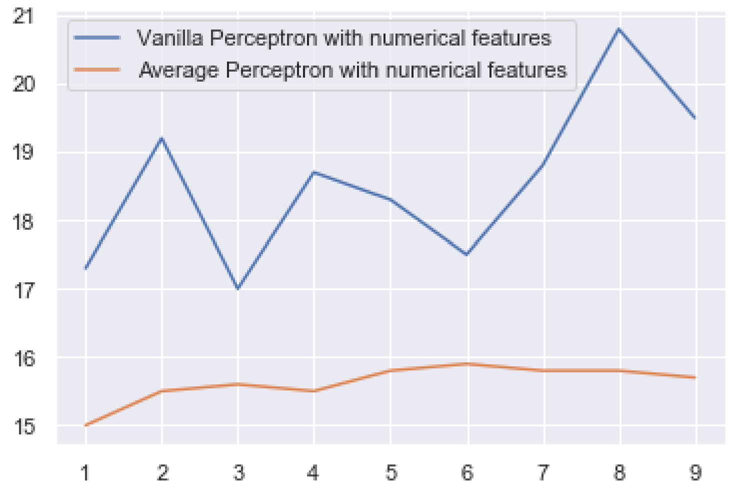
1. Try this experiment: reorder the training data so that all positive examples come first, followed by all negative ones. Did your (vanilla and averaged) perceptron(s) degrade in terms of dev error rate(s)?
Yes, reordering the training data did degrade the error rate(s) on dev set as seen below.



```
epoch: 1 dev_err: 23.90% (+ 0.30%)
epoch: 2 dev_err: 23.60% (+ 0.00%)
epoch: 3 dev_err: 23.60% (+ 0.00%)
epoch: 4 dev_err: 23.60% (+ 0.00%)
epoch: 5 dev_err: 23.60% (+ 0.20%)
epoch: 6 dev_err: 23.90% (+ 0.50%)
epoch: 7 dev_err: 24.00% (+ 0.60%)
epoch: 8 dev_err: 24.50% (+ 1.10%)
epoch: 9 dev_err: 24.60% (+ 1.20%)
```

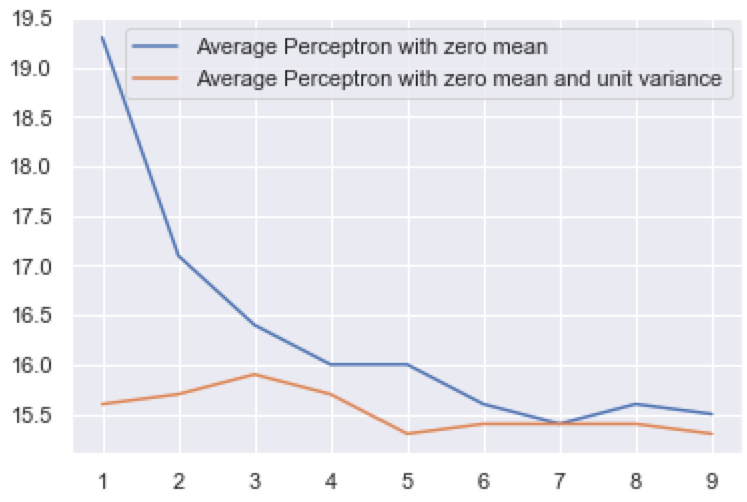
2. Try the following feature engineering:

   (a) adding the original numerical features (age, hours) as two extra, real-valued, features. **Using original numerical features did not improve the error rate on dev set (by using both Vanilla and Average Perceptron) as seen below.**



   (b) centering of each numerical dimension to be zero mean;

   (c) based on the above, and make each numerical dimension unit variance. **Centering the numerical features to have zero mean improved the model's performance.**
   Standardizing numerical features to have zero mean and unit variance improved the model's prediction error rate on dev set even further. This is evident by the following figure.



(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? **Using zero mean and unit variance helped achieve lowest error rate on dev set.**

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? **Zero mean and unit variance on of 15.3% at epoch 5 and epoch 9.**

   Q: what's your predicted positive % on dev? how does it compare with the ground truth positive %? **Predicted positive % on dev is 19.5% and ground truth positive % is 23.6%**

   Q: what's your predicted positive % on test? **Predicted positive % on test is 14.9%.**

Debriefing (required)

1. Approximately how many hours did you spend on this assignment? 26 hours
2. Would you rate it as easy, moderate, or difficult? Moderate
3. Did you work on it mostly alone, or mostly with other people? Mostly alone
4. How deeply do you feel you understand the material it covers (0%–100%)? 80%
5. Any other comments?