Most of the solutions are from the reference program, provided in the course homepage. Running the reference program might help understand this homework.

1 Perceptron and Averaged Perceptron

Question 1.1

```
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.
```

1.1.1 Solution

259

1.1.2 Grading Scheme

Full credits: exactly the same or with meaningful differences

Question 1.2

```
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: whats your best error rate on dev, and where do you get it? (e.g., at epoch
4.81)
```

1.2.1 Solution

18.77% at epoch 4.81

1.2.2 Grading Scheme

Full credits: exactly the same or with meaningful differences

Question 1.3

```
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().
```

1.3.1 Solution

For 5 epochs:
naive way: 3.452240 seconds smart way: 3.282846 seconds
No significant difference, because numpy couldn't take advantage of sparse vectors, while in smart implementation, the doc-products and updates are O(d) where d=259 but actually only around 10 features are positive.

1.3.2 Grading Scheme

Full credits: no difference or small difference in reported time

Question 1.4

```
Run the averaged perceptron on the training data for 5 epochs.
Q: this time, whats your best error rate on dev, and where do you get it?
(e.g., at epoch 3.27)
```

1.4.1 Solution

error rate: 16.31% at epoch 0.48

1.4.2 Grading Scheme

Full credits: similar results or with meaningful differences

Question 1.5

```
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?
```

1.5.1 Solution

Q1:
Top 5 positive features and weights:
1. (2,'Doctorate') : 6.1424615384615384
2. (2,'Prof-school') : 5.5263076923076921
3. (3,'Married-civ-spouse') : 4.7006923076923073
4. (2,'Masters') : 4.1011538461538457
5. (7,'42') : 2.9313076923076924

Top 5 negative features and weights:
1. (-1, 0) : -4.8220000000000001
2. (4, 'Farming-fishing') : -4.6801538461538463
3. (2, '7th-8th') : -4.2403076923076926
4. (0, '28') : -4.1552307692307693
5. (0, '26') : -3.7490000000000001

Q2:
Weight for Sex=Male and Sex=Female:
Male: -2.0156923076923077 Female: -2.8063076923076924

Both Male and Female features are negative because the dataset itself is mostly negative, and the model is under-fitting (not enough trained or not enough training data). Otherwise the model will attribute the skewed negativeness all to the 'bias' term, and give other features appropriate weights.

1.5.2 Grading Scheme

Q1: similar results(top features not weights) or with meaningful differences.
Q2: Answering dataset skewness deserves full credits, and pointing out under-fitting will get 1 extra point.

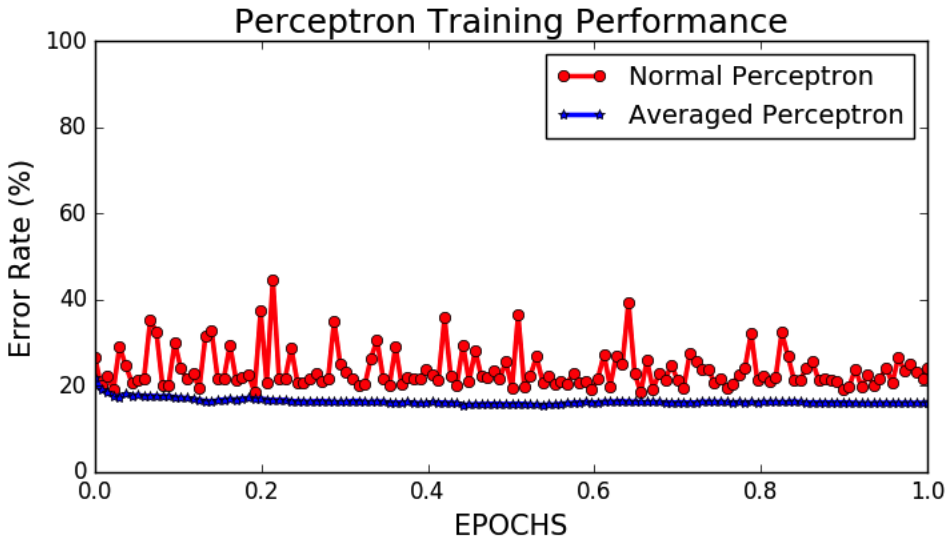
Question 1.6

```
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.
```

Cont.

1.6.1 Solution

Image made by Shangjia Dong (Group 23 with Kai and Chuan)



We could observe that naive perceptron's training curve is way too bumpy and converges slowly, while average perceptron's curve is pretty smooth and converges fast.

1.6.2 Grading Scheme

Full credits: similar curves

2 MIRA and Aggresive MIRA

Question 2.1

```
Implement the default (non-aggressive) MIRA, and its averaged version. Run them
for 5 epochs 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?
```

2.1.1 Solution

Best error rate on dev:
MIRA: 17.64% at epoch 1.81 Averaged MIRA: 16.38% at epoch 0.63

2.1.2 Grading Scheme

Full credits: similar results

Question 2.2

```
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}  {0.1, 0.5, 0.9}),
and where do you get them?
```

Cont.

2.2.1 Solution

p=0.1:
MIRA: 18.10% at epoch 0.81 Averaged MIRA: 16.11% at epoch 0.77

p=0.5:
MIRA: 17.77% at epoch 4.81 Averaged MIRA: 16.05% at epoch 0.37

p=0.9:
MIRA: 17.44% at epoch 4.59 Averaged MIRA: 16.05% at epoch 0.11

2.2.2 Grading Scheme

Full credits: similar results

Question 2.3

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

2.3.1 Solution

1. Generally speaking, MIRA has better performance than perceptron, aggressive MIRA has better performance than non-aggressive one, and averaged MIRA/perceptron has better performance than unaveraged one.
2. Averaging gives us significant improvement and also much faster convergence.
3. As p(aggressivity threshold) increases, aggressive MIRA reaches better accuracy, and when p=0 non-aggressive MIRA gets worst accuracy.

2.3.2 Grading Scheme

Full credits: at least 2 meaningful observations, no need to be the same.

3 Experimentations

```
Try the following:
```

Question 3.1

```
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?
```

3.1.1 Solution

1. Yes.

   Averaged Perceptron: from 16.31% to 22.75%.

   Averaged 0.9 MIRA: from 16.05% to 21.88%

   (getting worse)

2. Error rate goes back to 15.85% (Averaged Perceptron) / 15.85% (Averaged 0.9 MIRA). (getting better, better than baseline)

Cont.

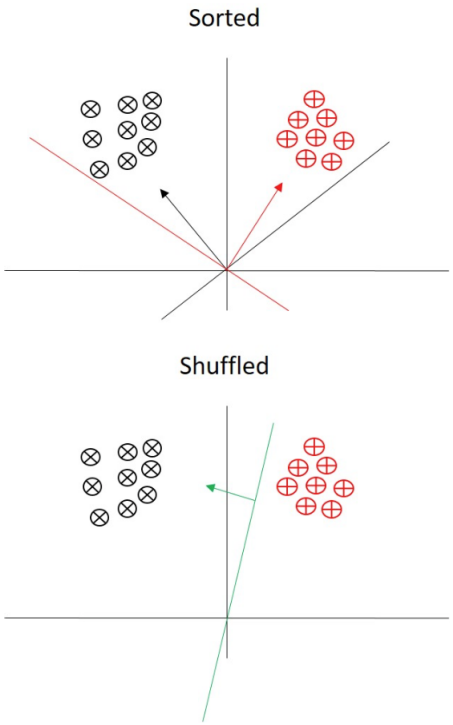


Figure 1: A toy scenario for illustrating the differences between sorting and shuffling training data. (the red/black arrow represents opposite directions)

3. (answer from Yogita Garud, David Greiner, and Peter Ferrero)

   If the data is in fixed order with all positive examples first and nega- tive examples later then weight vector gets affected by a few starting examples and a few ending examples. This takes longer time for convergence. Furthermore, if the positive and negative examples are clustered and not spread evenly about the origin, there may exist a case where learning from positive examples and then negative examples cause the weight vector to over-correct and miss the optimal separating hyperplane for every iteration through the training data. This effect is illustrated below. However, shuffling this same set of training data would force the weight vector towards a hyperplane that would successfully separate the two classes of training data.

3.1.2 Grading Scheme

In general, the error rate would increase a lot in (a) and decrease back in (b). Full credits given to answers described this scenario.

Full credits given to reasonable examples for (3). A geometric view of the example is encouraged.

Question 3.2

```
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?
```

Cont.

3.2.1 Solution

(a) by replacing all the age feature, the error rate of Averaged Perceptron goes from 16.31% to 16.45%, and the error rate of Averaged 0.9 MIRA goes from 16.05% to 17.18%. (getting worse)

(b) by adding one dimension of the age feature, the error rate of Averaged Perceptron goes from 16.31% to 15.72%, and the error rate of Averaged 0.9 MIRA goes from 16.05% to 15.98%. (getting better)

(c) by adding age with 5 bins, the error rate of Averaged Perceptron goes from 16.31% to 15.98%, and Averaged 0.9 MIRA stays the same result. (getting better)

(d) by adding the numerical education feature, the error rate of Averaged Perceptron goes from 16.31% to 15.65%, and the error rate of Averaged 0.9 MIRA goes from 16.05% to 15.98%. (getting better)

(e) Combination features would help this model a lot since it would help discover which combination of features is of affinity to affect the label. Group 23 (Chuan Tian, Shangjia Dong, and Kai Liu) reached the best performance on test set in this homework, by at around 1% than other groups. As discussed in class, their most improvement of them is used all the binary combination features (quadratic feature space) in the model and trained for a long time to fit this model.

3.2.2 Grading Scheme

Full credits given to answers that tried at least 3 of them, with reasonable explanations.

Generally, students should get (at least slight) improvements by trying b), c), d), or e). Those are encouraged.

Question 3.3

```
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?
```

3.3.1 Solution

(a) by using $\frac{1}{\#\text{error}} + \frac{1}{2}$, the error rate goes from 16.31% to 16.05% (Averaged Perceptron).

(b) the error rate of Averaged Perceptron goes from 16.31% to 15.92%, while the error rate of Averaged 0.9 MIRA goes from 16.05% to 15.58%.

3.3.2 Grading Scheme

Full credits given to reasonable answers. Generally, both these two fix would improve the model.

Question 3.4

```
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: whats your best error rate on dev, and which algorithm (and settings)
achieved it?
Q: whats your % of positive examples on test, and how does it compare to those
on train and dev?
```

Cont.

3.4.1 Solution

(answer from Shangjia Dong, Chuan Tian, and Kai Liu)

1. 15.053%. And thats achieved by Average Aggressive MIRA with p = 0.9 at epoch 0.332, on the model that I added interaction between all the binarized features to the model that already includes all binarized features, "Age" and "Hour" as numerical features. I did standardize "Age" and "Hour" before adding them to the model.

2. 22%. And I have the same results on train and dev set. I know the true positive rate on test set is even higher than 25%, so my model is probably still under-fitting.

3.4.2 Grading Scheme

Full points given to the results with reasonable explanations. Generally, the best model needs to at least have some combination of engineering described in section 3.2(b-d). The combination features is great, but it's optional in grading.

Cont.