OREGON STATE UNIVERSITY

CS534: Machine Learning
Homework 1

Shangjia Dong, Chuan Tian, Kai Liu

December 20, 2017

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 dimesionality)? Do not forget the bias dimension.

Answer: In total we have 259 features (include 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)

Answer: By using perceptron algorithm, the best error rate on dev is 18.23% at epoch 4.221 as shown in figure 1.1.

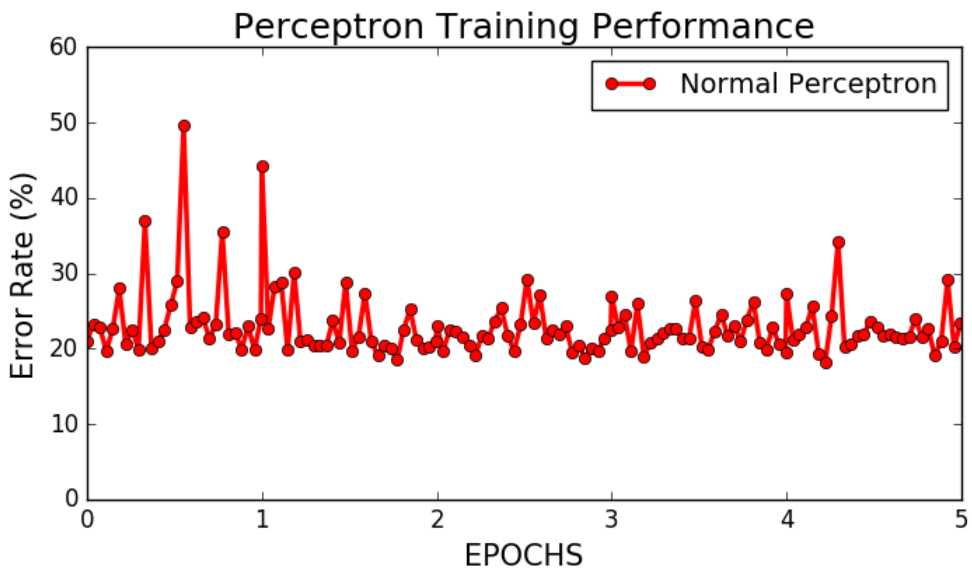


Figure 1.1: Normal perceptron performance

(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().

Answer: The smart averaged perceptron is always faster than naive averaged perceptron.

(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)

Answer: As show in Figure 1.2, by using averaged perceptron algorithm, the best error rate on dev for Naive training is 15.84% at epoch 1.221. For the smart training is 15.84% at epoch 0.810. The error rate is very close, but the speed wise, smart averaged perceptron is much faster than the naive approach.

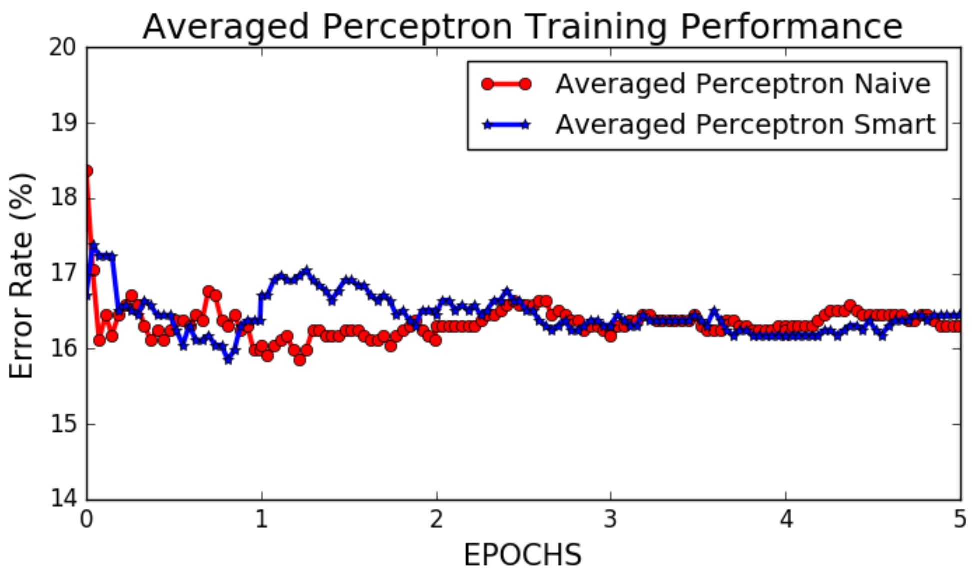


Figure 1.2: Naive and smart averaged perceptron performance comparison

(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?

Answer: The five most positive features are:

- Education: Doctorate
- Education: Prof-school
- MaritalStatus: Married-AF
- Age: 78
- Hour: 56

The five most negative features are:

- Age: 20
- Bias
- Hour: 3
- Age: 23

- Education: 1-4th

It does make sense. Higher degree, more salary. And more work hours, the salary will be higher, so does the age.

On the opposite, the lower the degree, the lower the salary. If you are younger, you are less likely to get high salary. Also, if you don't work enough hours, you won't get high salary.

The male and female are both negative because this data set is negative itself. There are only 24% people get paid higher than 50k. That is to say, without training the model much, a naive guess would be the person will be paid less than 50k regardless of the gender. Therefore, the weight for male and female are both negative. But weight for female are more negative than man.

(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

Answer: As illustrated in figure 1.3, the amplitude of perceptron algorithm is larger, but the amplitude of averaged perceptron algorithm is smaller.

We can see that the normal perceptron has more zig-zag fashion, converges slower, while the average perceptron not only have lower error rate, but also converges fast.

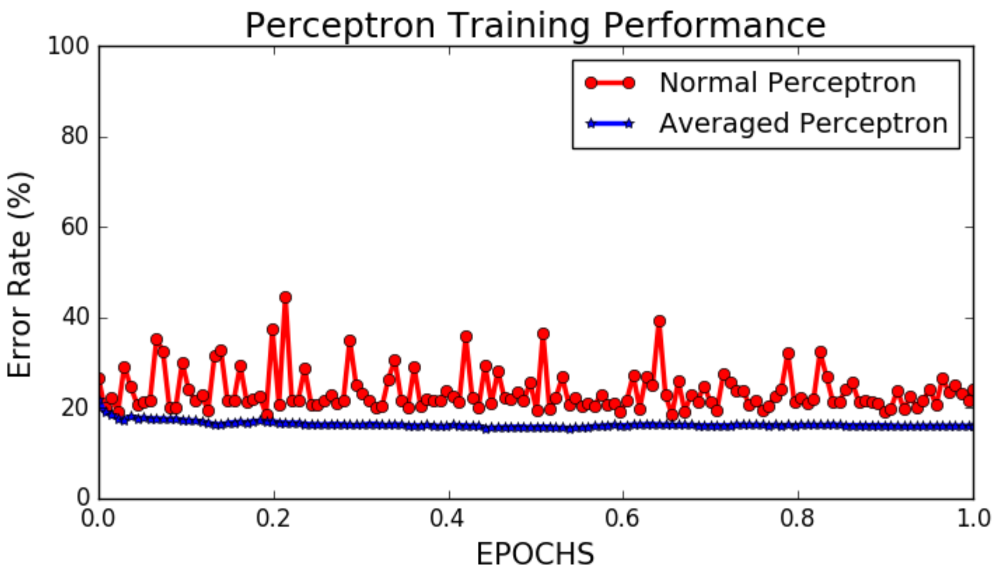


Figure 1.3: Vanilla and averaged perceptron training performance comparison

2 MIRA and Aggressive MIRA

(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?

Answer: As shown in figure 2.1, by using non-aggressive MIRA algorithm, the best error rate on dev is 17.11% at epoch 3.737.
By using averaged MIRA algorithm, the best error rate on dev is 15.97% at epoch 0.405.

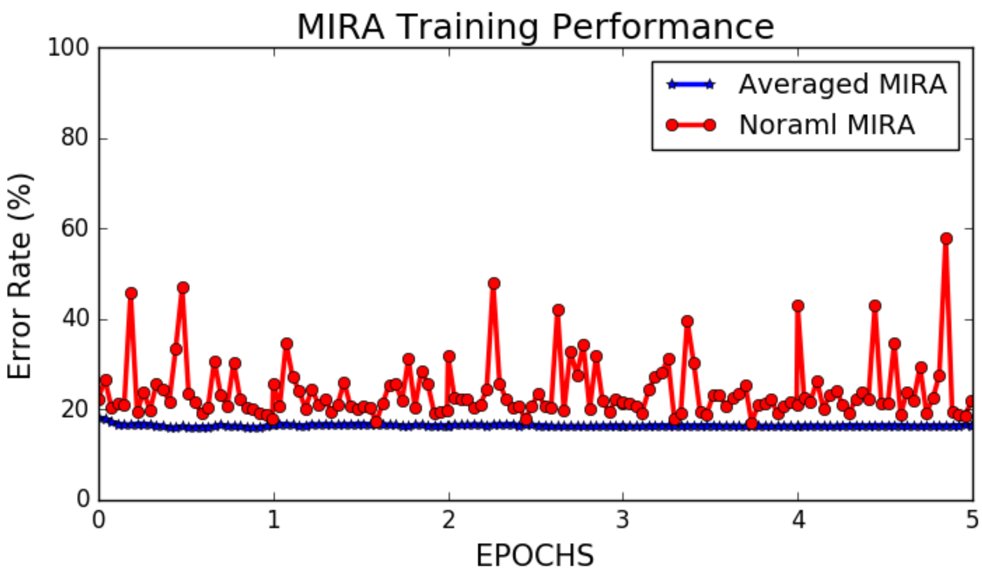


Figure 2.1: Non-aggressive MIRA and Averaged MIRA performance comparison

(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?

Answer:

A. p = 0.1, shown in figure 2.2
   - unavg: best error rate on dev: 17.97%, EPOCH: 3.295
   - avg: best error rate on dev: 15.71%, EPOCH: 0.921

B. p = 0.5, shown in figure 2.3

- unavg: best error rate on dev: 17.37%, EPOCH: 1.405
- avg: best error rate on dev: 15.65%, EPOCH: 0.405

C. p = 0.9, shown in figure 3.1

- unavg: best error rate on dev: 16.51%, EPOCH: 4.037
- avg: best error rate on dev: 15.65%, EPOCH: 0.368

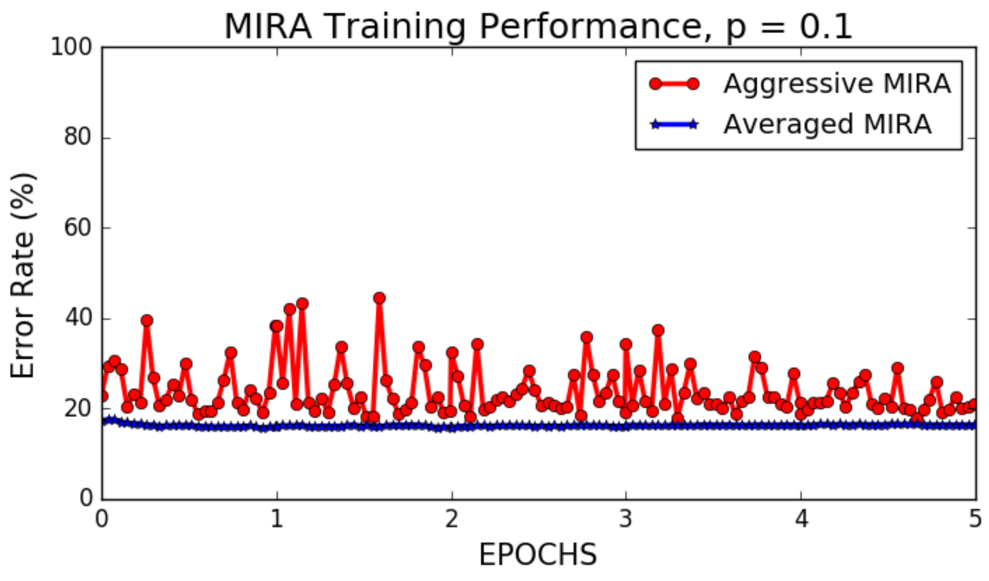


Figure 2.2: MIRA performance, p = 0.1

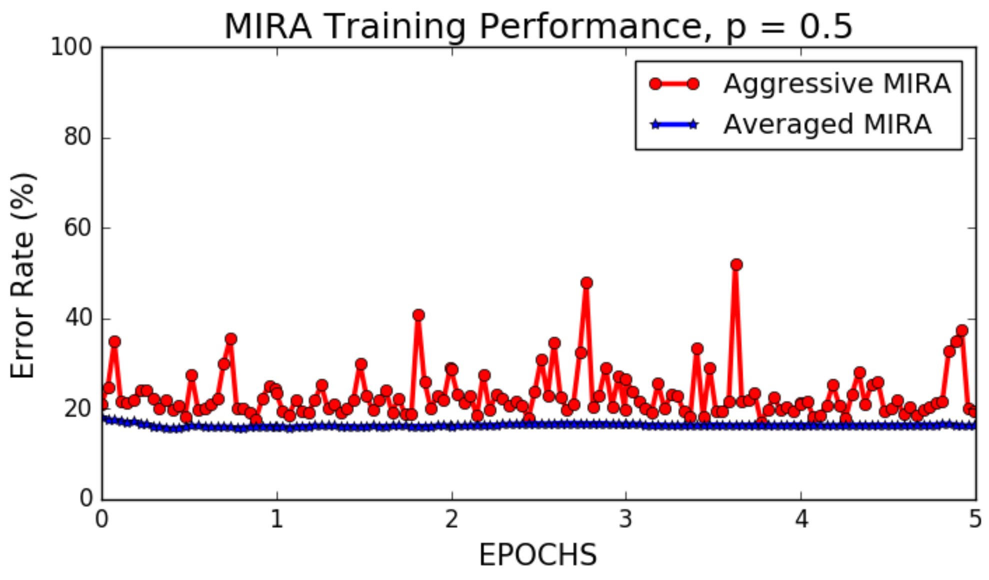


Figure 2.3: MIRA performance, p = 0.5

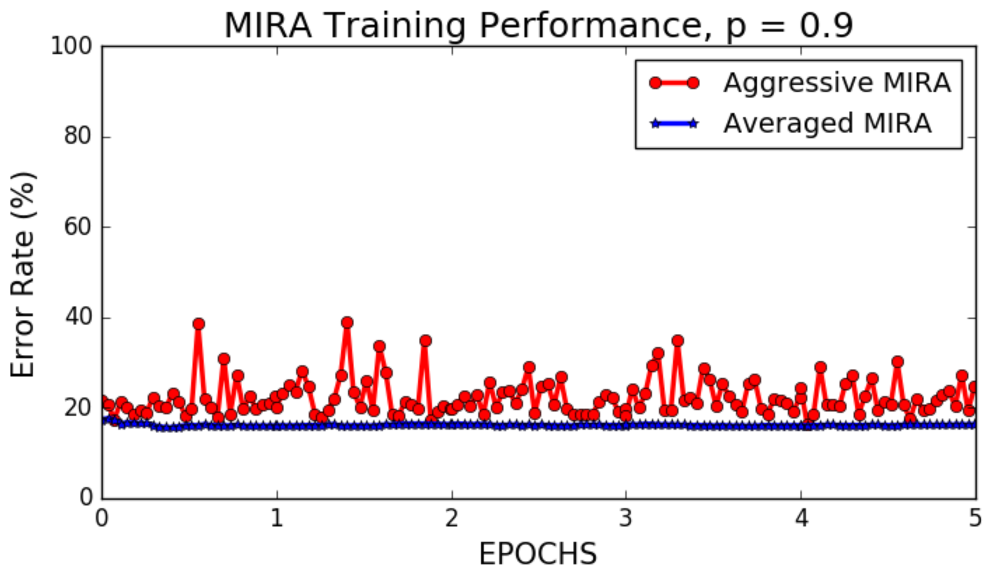


Figure 2.4: MIRA performance, p = 0.9

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

Answer: With increasing the epoch, the amplitude of averaged MIRA algorithm is becoming smaller and smaller, the error rate degrades smoothly. Also, the average MIRA is much more smooth and converges really fast compare to the non-averaged versions. However, for the non-average MIRA's, the more aggressive ones seems to converge faster.
Comparing MIRA's with Perceptrons, the perceptron ones have more zig-zag fashion, even in the averaged versions. That is probably because perceptron can oversolve or undersolve the problem, whereas MIRA tries to find the minimum change to achieve the functional margin. Also both perceptron and MIRA can be hugely influenced by the last examples, so averaging is really helpful.

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?

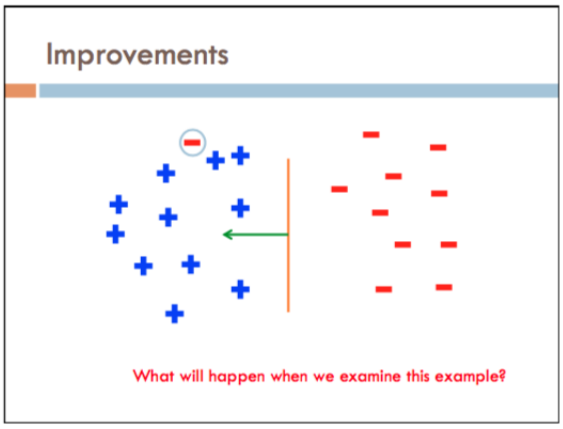
  Answer: By using perceptron algorithm, the best error rate on dev is 23.94%.
  By using MIRA algorithm, the best error rate on dev is 23.54%.

- Q: what if you shuffle the data before training?

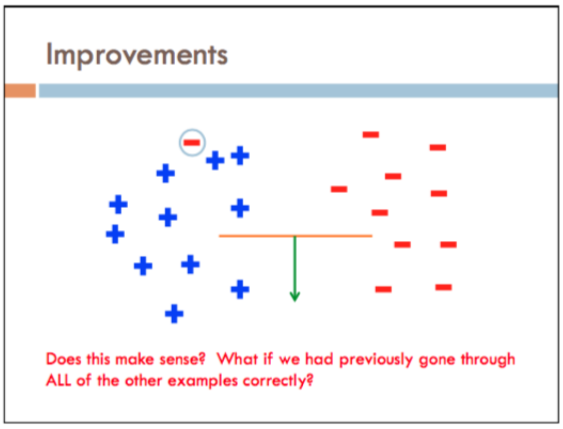
  Answer: By using perceptron algorithm, the best error rate on dev is 18.77%.
  By using MIRA algorithm, the best error rate on dev is 18.37%.

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

  *Answer:*I think so. Let's begin with a pathological example, and then extend from that. By looking at the picture, we can see that, if we process all the blue points first, and the red points on the right handside, then after a while the weight vector would be almost perfectly horizontal (that is, the dividing line is vertical), and it doesn't update anymore. However, if we process the red point on the lefthand side that causes the inseparable problem last, or close to last, then the weight vector would be flat or close to flat. In similar reasoning, in reality there are probably more than one problematic points, of each color. If we process them last or close to last, they will mess up the weight vector pretty badly, though probably wouldn't cause it to be vertical (so our decision line is flat). That could probably explain the phenomenon we observed.


Improvements

What will happen when we examine this example?


Improvements

Does this make sense? What if we had previously gone through ALL of the other examples correctly?

Figure 3.1: From Dr. David Kauchak's lecture notes. Dr. David Kauchak is an Assistant Professor from Pomana College.

(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?

Answer: In light of Dr. Huang's hint on Thursday, the model with all features binarized seems to be underfitting, instead of overfitting. And I happen to know that this model is the most complicated one you can have, without any squared or product of feature(s). So I never thought of replacing the binarized numerical features by the original ones – that could make the problem worse! Instead, I added Age and Hour as numerical features in. Binning features and adding a numerical feature education-level could have been helpful, but I went directly to adding the combination features, which is what our "naive model" lack of. And that helped a lot. I think that decrease the error rate by 0.5 or so.

However, I do have concerns about the implementation of squaring the features. I wrote a function to get all the interactions between features, but it takes forever to run. I looked spent hours researching about efficient implementation "by hand", but the only thing that comes up is from the PolynomialFeatures of sklearn, which could be coded by clever algorithms like Dynamic Programming, or even could be written in C. Since "algorithm" itself is not the focus of this class, I didn't take more time to investigate this. Any suggestions would be more than welcome, though.

(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?

Answer: Since MIRA already has variable learning rate, and standardizing binary features doesn't really make sense, we standardized the numerical features: "Age" and "Hour" before adding them to the model. That helps. We have a best error rate of 15.65% now. To be honest, adding those numerical features without standardizing didn't seem to do much good. But after standardizing, it helps decreasing the error rate by 0.1 or so.

(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.

a Q: what's your best error rate on dev, and which algorithm (and settings) achieved it?

  Answer: 15.053%. And that's 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.

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

  Answer: 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. But I'm happy about the result.