HW1 Perceptron

Ming-hung Shih   Shotaro Matsui   Fengfei Zheng

1. Perceptron and Averaged Perceptron (unoptimized)

   1.1 We have 267 features including the bias dimension.

   1.2 Running vanilla perceptron for 5 epochs, we have:
```
vanilla perceptron min error rate of 0.181034482759 at 1.14201510407
```
   1.3 Running a partially disabled version of the program, we got
```
vanilla took 20.847779512405396, naive avg took 6.5841734409332275, smart avg
took 5.913713693618774
```
   So it seems that smart version is indeed faster, but not by a lot as predicted by O(U/DT), partly due to extra subtraction operations in the smart way.

   1.4 Running averaged perceptron for 5 epochs, we have:

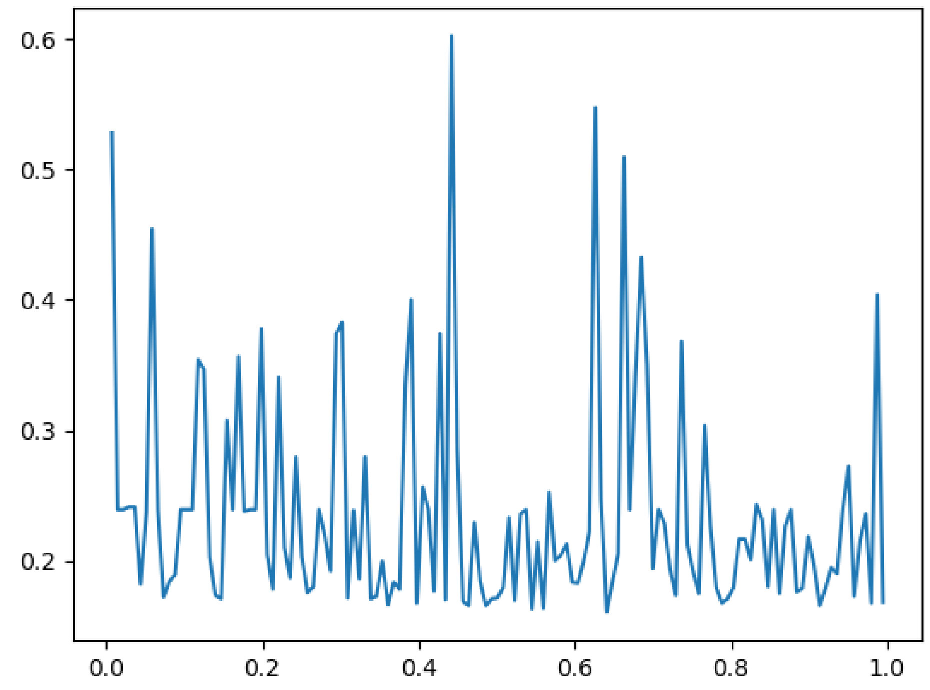
```
naive averaged perceptron min error rate of 0.163129973475 at 0.478909559772
smart averaged perceptron min error rate of 0.163129973475 at 0.478909559772
```
As can be seen, both implementations yield identical results, except for the time difference.

   1.5 We found the top 5 neg/pos features, as well as male/female scores as follows:
```
***Top 5 negative features are:***
('age', 20)
 Columbia
 Bias
 7th-8th
('age', 88)

***Top 5 positive features are:***
 Doctorate
 Prof-school
 Married-civ-spouse
('age', 78)
 Masters

***Male score = -2.39927795174
 Female score = -3.252385338***
```

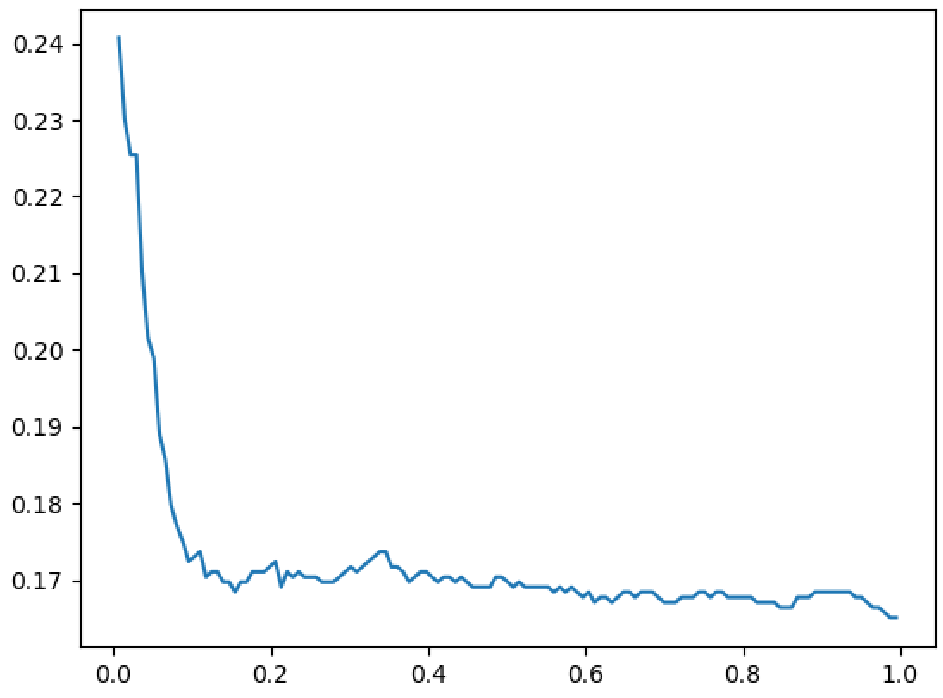
They do make sense. On the negative side, bias dimension has a big sway on the final decision, due to the biased nature of the data. Extremal young and old ages also correlate poorly with income. On the positive side, Doctorate, Prof-schools and Masters are the highest degrees in US education, and correlate well with income statistics. Married-civ-spouse is also a good indicator of stable life and thus relates to income. It should be noted the output ranking should not be read into too much, as it would change with different shuffles of data. What's not changing is that these above features usually are among the top 10 of neg/pos, respectively.

Male and Female both receive negative scores, due to them being in a negatively biased data set. NB that it's not always the case that female has a less negative score. This can be caused by various numerical artifacts, such as males having a higher percentage in the population thus should be more negatively weighed in a negatively biased data set. But in this case, females have worse scores, which is also understandable.

1.6 In the first epoch, vanilla exhibits the following learning behavior (dev error rates):



while averaged perceptron exhibits the following behavior (dev error rates):



As can be seen, averaged perceptron learns the data much more efficiently and predictably. The reason is that in the vanilla version, the decision on the correct model is largely swayed by individual data points, over and over again, in both negative and positive directions, resulting in a model that fails to fairly evaluate the data and account for the contributions from all members. This means that the model doesn't really make much progress and a lot of times are set back by erratic data points, resulting in spikes in the error rate plot. Its averaged counterpart, on the other hand, takes into account the diversity of the dataset and learns to classify new ones more prudently. Surely enough, the error rate steadily goes down, an indication of how learning occurs.

2. MIRA and Aggressive MIRA

   2.1

   "p" is agressivity, p=0 means native MIRA.
   MIRA= True , p= 0.0 , (best error rate)= 17.042 % in epoch 1.658
   MIRA= True , p= 0.0 , (best avg. error rate)= 17.241 % in epoch 1.400

   2.2

   MIRA= True , p= 0.1 , (best error rate)= 17.308 % in epoch 1.658
   MIRA= True , p= 0.1 , (best avg. error rate)= 17.507 % in epoch 0.221

   MIRA= True , p= 0.5 , (best error rate)= 18.236 % in epoch 4.015
   MIRA= True , p= 0.5 , (best avg. error rate)= 17.175 % in epoch 1.400

   MIRA= True , p= 0.9 , (best error rate)= 18.700 % in epoch 3.610
   MIRA= True , p= 0.9 , (best avg. error rate)= 17.175 % in epoch 1.031

   2.3

   average MIRA usually reaches its stable state early in 1st epoch.
   and it seems aggressive update doesn't necessarily help to get better error rate nor convergence.

   Compared with the perceptron in previous section, MIRA has a better accuracy on the unaveraged ones, but poorer accuracy on the averaged ones. MIRA also in general tends to be slower to converge due to its aggressiveness.

3. Experimentation

   (A) Averaged Perceptron.

   3.1 Yes it degrades when dealing with ordered data, and here's how:

   ```
   vanilla perceptron min error rate of 0.239389920424 at 0.257874378339
   naive averaged perceptron min error rate of 0.238063660477 at 2.09983422361
   smart averaged perceptron min error rate of 0.238063660477 at 2.09983422361
   ***Top 5 negative features are:***
   ('hours', 13)
    Handlers-cleaners
   ('age', 50)
   ```

```
        ('hours', 40)
         Bias

        ***Top 5 positive features are:***
        ('age', 52)
        ('hours', 45)
        ('age', 42)
        ('age', 31)
         Prof-specialty

        ***Male score = -0.98320869405
         Female score = -0.507268373549***
```

We start getting nonsensical results.

After shuffling, the results are:

```
vanilla perceptron min error rate of 0.184350132626 at 2.91029655554
naive averaged perceptron min error rate of 0.161140583554 at 1.03149751335
smart averaged perceptron min error rate of 0.161140583554 at 1.03149751335

***Top 5 negative features are:***
('age', 20)
 Columbia
('hours', 27)
 Bias
('age', 23)

***Top 5 positive features are:***
 Doctorate
 Prof-school
('hours', 56)
 Married-civ-spouse
 Masters

***Male score = -3.17686498434
 Female score = -2.54854300976***
```

The results become much more reasonable. The reason is that when all we see are positive data, the perceptron model finds its way there and will never, ever be corrected until it comes across a negative example. Once it’s led back to the negative world, it again finds its niche place and won’t be corrected be the subsequent negatives. The perceptron ends up being arrogant and self-complacent, reluctant to make changes even though it’s seeing more data. And when the real test comes, all the bubbles burst.

3.2 In this case, the only feature that works well is the addition of an education level. It turns out that numerical values of age/hours don't work well because income is not linearly related to them. Any attempt to put a numerical feature of such will unfortunately make the data more inseparable.

3.3 In this case, neither a variable learning rate nor normalization help us get further than what feature engineering alone has given us. We ended up not adopting any of these even though we tried hard experimenting.

The results for 3.2 and 3.3 are given below:

```
vanilla perceptron min error rate of 0.163793103448 at 4.64173881009
naive averaged perceptron min error rate of 0.15848806366 at 4.82593479462
smart averaged perceptron min error rate of 0.15848806366 at 4.82593479462

***Top 5 negative features are:***
 Bias
('age', 23)
 Never-married
 Female
('age', 24)

***Top 5 positive features are:***
 Married-civ-spouse
 Prof-school
 Doctorate
 Exec-managerial
('hours', 65)

***Male score = -35.1050874931
 Female score = -23.8721532511***
```

As can be seen, the results are pretty reasonable.

3.4. Using results from 3.2 and 3.3, we got a predicted output in income.test.predicted.txt

(B) MIRA

3.1

```
MIRA:
native:
   MIRA= True , p= 0.5 , (best error rate)= 18.236 % in epoch 4.015
   MIRA= True , p= 0.5 , (best avg. error rate)= 17.175 % in epoch 1.400
reorder:
   MIRA= True , p= 0.5 , (best error rate)= 23.939 % in epoch 0.258
   MIRA= True , p= 0.5 , (best avg. error rate)= 23.939 % in epoch 0.405
random:
   MIRA= True , p= 0.5 , (best error rate)= 17.042 % in epoch 3.979
   MIRA= True , p= 0.5 , (best avg. error rate)= 17.175 % in epoch 0.295
```

update occurs only when x_i is mis-classified.
when all positive examples come first, then weight would be like "classifying anything to positive."
Perceptron would reach this early and would pass all the rest of positive examples, but obviously this is not correct.
And same thing would happen in negative half.
So if we reorders training data (positives first, then negatives) we are kind of wasting data.
Even if the data is linearly separable, it would slow convergence.

Shuffling data would make positive and negative data uniform, which makes data more meaningful.
The reason why the best error rate is achieved earlier than "native" and "reorder" would be this.

The best error rate of "reorder" (non-average) achieved around 1/4, which is near the border of positive and negative.
In fact, when we increase positive examples so that (pos:neg = 1:1), we got

```
    MIRA= True , p= 0.5 , (best error rate)= 23.939 % in epoch 0.516
    MIRA= True , p= 0.5 , (best avg. error rate)= 23.939 % in epoch 0.615
```

in which the best error rate is achieved around 1/2.

3.2
(a)

```
MIRA= True , p= 0.5 , (best error rate)= 17.241 % in epoch 4.973
MIRA= True , p= 0.5 , (best avg. error rate)= 20.159 % in epoch 0.147
```

(b)
adding numerical "age" and "working-hours" besides their binarization.

```
MIRA= True , p= 0.5 , (best error rate)= 18.302 % in epoch 4.789
MIRA= True , p= 0.5 , (best avg. error rate)= 19.430 % in epoch 1.658
```

(c)
rounded (e.g. 48->50, 33->30)

```
MIRA= True , p= 0.5 , (best error rate)= 18.037 % in epoch 2.063
MIRA= True , p= 0.5 , (best avg. error rate)= 19.496 % in epoch 2.689
```

(d)
Preschool < 1st-4th < 5th-6th < 7th-8th < 9th < 10th < 11th < 12th < HS-grad < Prof-school < Assoc-acdm < Assoc-voc < Some-college < Bachelors < Masters < Doctorate

```
MIRA= True , p= 0.5 , (best error rate)= 17.308 % in epoch 0.442
MIRA= True , p= 0.5 , (best avg. error rate)= 18.435 % in epoch 0.221
```

It turns out that professional schools are more prestigious than it sounds. However we try place Prof-school bw Bachelors/Masters:
MIRA= True , p= 0.5 , (best error rate)= 17.573 % in epoch 0.442
MIRA= True , p= 0.5 , (best avg. error rate)= 19.032 % in epoch 0.037

Or place Prof-school bw Masters/Doctors:
MIRA= True , p= 0.5 , (best error rate)= 17.573 % in epoch 0.442
MIRA= True , p= 0.5 , (best avg. error rate)= 19.032 % in epoch 0.037

The accuracy isn't improved, which means that the algorithm is somewhat tolerant of potential inaccuracy in the extraction of features.

(e)
Introduced {gender} x {marital-status} and {race} x {nationality}

MIRA= True , p= 0.5 , (best error rate)= 18.833 % in epoch 1.658
MIRA= True , p= 0.5 , (best avg. error rate)= 17.440 % in epoch 0.553

(combination of (a) and (e))
MIRA= True , p= 0.5 , (best error rate)= 16.247 % in epoch 3.684
MIRA= True , p= 0.5 , (best avg. error rate)= 20.955 % in epoch 4.900

(combination of (d) and (e))
MIRA= True , p= 0.5 , (best error rate)= 18.302 % in epoch 2.947
MIRA= True , p= 0.5 , (best avg. error rate)= 18.501 % in epoch 0.221

(combination of (a) and (d))
MIRA= True , p= 0.5 , (best error rate)= 19.761 % in epoch 2.947
MIRA= True , p= 0.5 , (best avg. error rate)= 21.817 % in epoch 4.863

(combination of (a), (d) and (e))
MIRA= True , p= 0.5 , (best error rate)= 19.761 % in epoch 2.947
MIRA= True , p= 0.5 , (best avg. error rate)= 21.883 % in epoch 4.973

among (a) and (d), nothing could update average-MIRA's best error rate.

(a), (b) and (c) is similar approach.
Numerical feature gives us more precise relationship between input values (e.g. age "23" is closer to "30" than "1", which are all binarized to 0 in our implementation) comparing to binary or binned features.

So we thought (a) should be better than binarized and binned ones, but the result was against our guess.

If it is not caused by bug, probably because age and working-hours are not linearly related with income, as explained previously as well.

(a), (d) and (e) seems better than the rest, but simply combining them does not give us a good result.
Combination of (a) and (e) gave us new native-MIRA best error rate, but not average-MIRA.

3.3
(a)
learning rate = 1000/(1000+t), where t is current data number (so 1 to 27,145*epoch in income.train.txt)
MIRA= True , p= 0.5 , (best error rate)= 16.844 % in epoch 0.700
MIRA= True , p= 0.5 , (best avg. error rate)= 16.976 % in epoch 0.184
kind of better than normal

(b)
MIRA= True , p= 0.5 , (best error rate)= 17.374 % in epoch 0.258
MIRA= True , p= 0.5 , (best avg. error rate)= 19.231 % in epoch 0.037
standardization really hastes convergence.

(a) and (b)
MIRA= True , p= 0.5 , (best error rate)= 16.910 % in epoch 1.105
MIRA= True , p= 0.5 , (best avg. error rate)= 17.573 % in epoch 1.547

Variable learing rate updates average-MIRA best error rate.
Variable learning rate is good in performance, and standardization is good in convergence.
Their combination is kind of killing each other's good points.
It seems that constant learning rate works better when standardizing input.

3.4
combination of 3.2(a) and 3.2(e) updates non-average-MIRA best error.
variable learing rate updates average-MIRA best error rate, and also non-average-MIRA is good.

when combining those 2, we get
MIRA= True , p= 0.5 , (best error rate)= 18.236 % in epoch 3.758
MIRA= True , p= 0.5 , (best avg. error rate)= 23.408 % in epoch 0.147
which is not good.

so our best error rate is 16.976 %, which is
p=0.5, using variable learning rate 1000/(1000+t).
In this setting, we got 21.869% (330/1509) positive examples.

This prediction is named "income.test.predicted"
while positive examples for other 2 files are like this,
positive rate of income.train.txt = 24.936 %, 6769 / 27145
positive rate of income.dev.txt = 23.939 %, 361 / 1508
We think we got reasonable prediction.

for reference, we tried non-average-MIRA best 16.247 %, which is
p=0.5, using 3.2(a) replacing binarized numerical features by original numbers, and 3.2(e) adding combination features.
In this setting, WE got 23.194% (350/1509) positive examples, which seems also reasonable.
this prediction is named "income.test.predicted.nonavg"

To sum up, we end up having THREE (3) potential predictions, which are: “income.test.predicted.txt”,
"income.test.predicted"
and "income.test.predicted.nonavg".