Ben Tankus

CS 519 HW 1

4-2-21

1 Data (Pre-)Processing (Feature Map)

1. **Take a look at the data. A training example looks like this:**

   37, Private, Bachelors, Separated, Other-service, White, Male, 70, England, <=50K

which includes the following 9 input fields plus one output field (y): age, sector, education, marital-status, occupation, race, gender, hours-per-week, country-of-origin, target Q: What are the positive % of training data? What about the dev set? Does it make sense given your knowledge of the average per capita income in the US?

Train Positive Percent is: 1251 of 5000 or 25%

Dev Positive Percent is: 236 of 1000 or 23.6%

These values make sense for yearly salaries in the 90s. According to google, median salary in the 90s was around 20,000 dollars so 25% of people around $50,000 makes sense

2. **Q: What are the youngest and oldest ages in the training set? What are the least and most amounts of hours per week do people in this set work? Hint: cat income.train.txt.5k | sort -nk1 | head -1**

Min Hours 1 min age 17

Max Hours 99 max age 90

3. There are two types of fields, numerical (age and hours-per-week), and categorical (everything else). The default preprocessing method is to binarize all categorical fields, e.g., race becomes many binary features such as race=White, race=Asian-Pac-Islander, etc. These resulting features are all binary, meaning 1 In principle, we could also convert education to a numerical feature, but we choose not to do it to keep it simple. 1 their values can only be 0 or 1, and for each example, in each field, there is one and only one positive feature (this is the so-called "one-hot" representation, widely used in ML and NLP). Q: Why do we need to binarize all categorical fields?

Answer: The simple reason we need to binarize all categorical fields is for math. One cannot sum, average, or do math of much significance on categorical data without transforming it to numerical (binarizing).

4. Q: If we do not count age and hours, what's maximum possible Euclidean and Manhattan distances between two training examples? Explain.

Answer: Euclidean distance: sqrt(2x7) = **3.7417**, Manhattan: 2x7 = **14**

There are 7 total fields not including age and hours. When binarized, each of these fields have two options, true or false. Assuming the two furthest apart points, their difference will be a function of number of fields, and number of choices, in this case, 2x7. Manhattan distance is just this simple product, Euclidean distance is the square root of this.

5. Why we do not want to binarize the two numerical fields, age and hours? What if we did? How should we define the distances on these two dimensions so that each field has equal weight? (In other words, the distance induced by each field should be bounded by 2 (N.B.: not 1! why?)).

Hint: first, observe that the max distance between two people on a categorial field is 2. If we simply "normalize" a numerical field by, say, age / 100, it might look OK but now what's the max distance between two people on age? Are you treating all fields equally?

Answer: The two numerical fields already can be analyzed without binarizing. The range of both these fields is great enough that binarizing them would create excessive model complexity, greatly increasing the terms in the model. For instance, there are ages ranging from 17 to 90 in the data, meaning there would be roughly 70 additional columns if this field were binarized, for little added value. The same is true for hours. To ensure equal weight to both fields I recommend transforming them both to be percent of maximum fields, where the maximum column value is considered 100, and each data point is a percent of that.

6. Q: How many features do you have in total (i.e., the dimensionality)?

Hint: should be around 90. How many features do you allocate for each of the 9 fields?

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

Answer: In the Dev set there are 88 features not including the numerical features. Including the numerical features there should be around 235. This changes to 92 and 230

respectively in the training set. These numbers are different because of the different categories available in each.

7. Q: How many features would you have in total if you binarize all fields?

Answer: If all fields were binarized there would be 235. If only the numerical features were binarized there would only be 88. In the training set it would be 230 and 88 respectively.

PART 2 Calculating Manhattan and Euclidean Distances

1. Q: Find the five (5) people closest to the last person (in Euclidean distance) in dev, and report their distances:

```
Original Person:
 ['58', 'Private', 'HS-grad', 'Widowed', 'Adm-clerical', 'White', 'Female', '40', 'United-States'
, '<=50K']

Index: 681 e-distance: 0.0324
person: ['30', 'Private', 'HS-grad', 'Widowed', 'Adm-clerical', 'White', 'Female', '40', 'United-
States', '<=50K']

Index: 1010 e-distance: 0.4624000000000001
person: ['55', 'Private', 'HS-grad', 'Widowed', 'Adm-clerical', 'White', 'Female', '40', 'United-
States', '<=50K']

Index: 3680 e-distance: 0.4736000000000001
person: ['49', 'Private', 'HS-grad', 'Widowed', 'Adm-clerical', 'White', 'Female', '20', 'United-
States', '<=50K']

Index: 3769 e-distance: 0.6152
person: ['58', 'Private', 'HS-grad', 'Widowed', 'Adm-clerical', 'White', 'Female', '27', 'United-
States', '<=50K']

Index: 3698 e-distance: 0.7376
person: ['59', 'Private', 'HS-grad', 'Widowed', 'Adm-clerical', 'White', 'Female', '60', 'United-
States', '<=50K']
```

These look very close to correct. They all predict the correct value for salary, and match most of the expl anatory variables.

2. Redo the above using Manhattan distance.

```
Index: 681 manhat-distance: 16.559999999999985

person: ['30', 'Private', 'HS-grad', 'Widowed', 'Adm-clerical', 'White', 'Female', '40', 'United-
States', '<=50K']


Index: 1010 manhat-distance: 62.55999999999998

person: ['55', 'Private', 'HS-grad', 'Widowed', 'Adm-clerical', 'White', 'Female', '40', 'United-
States', '<=50K']


Index: 3680 manhat-distance: 88.3199999999999

person: ['49', 'Private', 'HS-grad', 'Widowed', 'Adm-clerical', 'White', 'Female', '20', 'United-
States', '<=50K']
```

```
Index: 3769 manhat-distance: 92.0

person: ['58', 'Private', 'HS-grad', 'Widowed', 'Adm-clerical', 'White', 'Female', '27', 'United-States', '<=50K']


Index: 3698 manhat-distance: 106.71999999999979

person: ['59', 'Private', 'HS-grad', 'Widowed', 'Adm-clerical', 'White', 'Female', '60', 'United-States', '<=50K']
```

I believe the Manhattan distance is also correct, it is producing exactly the same people, just with different (much greater) distances, as is expected.

3. What are the 5-NN predictions for this person (Euclidean and Manhattan)? Are these predictions correct?

Both Manhattan and Euclidean methods predict this person to earn <=50k. This is correct.

4. Redo all the above using 9-NN (i.e., find top-9 people closest to this person first).

```
Original Person:

 ['58', 'Private', 'HS-grad', 'Widowed', 'Adm-clerical', 'White', 'Female', '40', 'United-States', '<=50K']


Index: 681 e-distance: 0.0324

person: ['30', 'Private', 'HS-grad', 'Widowed', 'Adm-clerical', 'White', 'Female', '40', 'United-States', '<=50K']


Index: 1010 e-distance: 0.4624000000000001

person: ['55', 'Private', 'HS-grad', 'Widowed', 'Adm-clerical', 'White', 'Female', '40', 'United-States', '<=50K']


Index: 3680 e-distance: 0.4736000000000001

person: ['49', 'Private', 'HS-grad', 'Widowed', 'Adm-clerical', 'White', 'Female', '20', 'United-States', '<=50K']


Index: 1713 e-distance: 0.81

person: ['66', 'Private', 'HS-grad', 'Widowed', 'Adm-clerical', 'White', 'Female', '40', 'United-States', '<=50K']
```

Index: 3698 e-distance: 0.7376

person: ['59', 'Private', 'HS-grad', 'Widowed', 'Adm-clerical', 'White', 'Female', '60', 'United-States', '<=50K']

Index: 3769 e-distance: 0.6152

person: ['58', 'Private', 'HS-grad', 'Widowed', 'Adm-clerical', 'White', 'Female', '27', 'United-States', '<=50K']

Index: 2003 e-distance: 0.9236

person: ['68', 'Private', 'HS-grad', 'Widowed', 'Adm-clerical', 'White', 'Female', '30', 'United-States', '<=50K']

Index: 2450 e-distance: 1.1664

person: ['75', 'Private', 'HS-grad', 'Widowed', 'Adm-clerical', 'White', 'Female', '40', 'United-States', '<=50K']

Index: 3510 e-distance: 1.414613562373095

person: ['20', 'Private', 'HS-grad', 'Married-civ-spouse', 'Adm-clerical', 'White', 'Female', '40', 'United-States', '<=50K']

Index: 681 manhat-distance: 16.559999999999985

person: ['30', 'Private', 'HS-grad', 'Widowed', 'Adm-clerical', 'White', 'Female', '40', 'United-States', '<=50K']

Index: 1010 manhat-distance: 62.55999999999998

person: ['55', 'Private', 'HS-grad', 'Widowed', 'Adm-clerical', 'White', 'Female', '40', 'United-States', '<=50K']

Index: 3680 manhat-distance: 88.3199999999999

person: ['49', 'Private', 'HS-grad', 'Widowed', 'Adm-clerical', 'White', 'Female', '20', 'United-States', '<=50K']

Index: 1713 manhat-distance: 82.80000000000004

person: ['66', 'Private', 'HS-grad', 'Widowed', 'Adm-clerical', 'White', 'Female', '40', 'United-States', '<=50K']

Index: 3698 manhat-distance: 106.71999999999979

person: ['59', 'Private', 'HS-grad', 'Widowed', 'Adm-clerical', 'White', 'Female', '60', 'United-States', '<=50K']

```
Index: 3769 manhat-distance: 92.0

person: ['58', 'Private', 'HS-grad', 'Widowed', 'Adm-clerical', 'White', 'Female', '27', 'United-
States', '<=50K']


Index: 2003 manhat-distance: 104.88000000000004

person: ['68', 'Private', 'HS-grad', 'Widowed', 'Adm-clerical', 'White', 'Female', '30', 'United-
States', '<=50K']


Index: 2450 manhat-distance: 99.35999999999987

person: ['75', 'Private', 'HS-grad', 'Widowed', 'Adm-clerical', 'White', 'Female', '40', 'United-
States', '<=50K']


Index: 3510 manhat-distance: 3.8400000000000016

person: ['20', 'Private', 'HS-grad', 'Married-civ-spouse', 'Adm-clerical', 'White', 'Female',
'40', 'United-States', '<=50K']
```

Both methods yielded <=50K for the prediction which is correct.

PART 3 k-nearest Neighbor Classification

1. Implement the basic k-NN classifier (with the default Euclidean distance)

Q: Is there any work in training after finishing the feature map

There is no training the k-NN model, so there is no work in training the after finishing the feature map

Q: What's the time complexity of k-NN to test one example (dimensionality d, size of training set |D|)?

Time complexity steps:

1. Calculating the distance between each person: O(D) * d (all people)
2. Sort the algorithm: O(D^2)
3. Time to check the top matches: k
4. Overall time complexity: O(D) * d + D^2 + k

Q: Do you really need to sort the distances first and then choose the top k? Hint: there is a faster way to choose top k without sorting

It would be faster to search the list of distances for minimum and remove them without replacement. This time complexity would simply be D*k. I'm not really sure how to implement this though.

2. Q: Why the k in k-NN has to be an odd number?

K is typically an odd number because it is then less likely for ties to occur when comparing data.

3. Evaluate k-NN on the dev set and report the error rate and predicted positive rate for k = 1, 3, 5, 7, 9, 99, 999, 9999, e.g., something like:

k=1 dev_err xx.x% (+:xx.x%) k=3 ... ... k=9999 ...

Q: what's your best error rate on dev, and where did you get it? (Hint: 1-NN dev error should be ~23% and its positive % should be ~27%).

```
k = 1 dev_err 22.70% (+:21.90%)
k = 3 dev_err 19.20% (+:19.60%)
k = 5 dev_err 18.30% (+:18.10%)
k = 7 dev_err 18.00% (+:18.20%)
k = 9 dev_err 16.80% (+:20.60%)
k = 99 dev_err 16.90% (+:17.90%)
k = 999 dev_err 18.60% (+:7.40%)
k = 9999 dev_err 23.60% (+:0.00%)

k = 33 dev_err 15.70% (+:18.50%)
```

The best error rate on dev is between k = 9 and k = 99 (~16%). I plotted a few more than instructed to bridge the gap between 9 and 99 and found that k = 33 with an error rate of 15.70% had the lowest error. K = 1 is obviously underfitting, where k over 100 is fitting each point in dev using 10% of the data, which seems large, and likely to overfit the model.

4. **Now report both training and testing dev errors:**

```
k = 1 train_err 15.20% (+:22.14%) dev_err 22.70% (+:21.90%)
k = 3 train_err 16.86% (+:20.44%) dev_err 19.20% (+:19.60%)
k = 5 train_err 17.16% (+:18.66%) dev_err 18.30% (+:18.10%)
k = 7 train_err 17.74% (+:18.56%) dev_err 18.00% (+:18.20%)
k = 9 train_err 17.08% (+:20.94%) dev_err 16.80% (+:20.60%)
k = 99 train_err 18.28% (+:18.54%) dev_err 16.90% (+:17.90%)
k = 999 train_err 21.22% (+:7.00%) dev_err 18.60% (+:7.40%)
k = 9999 train_err 25.02% (+:0.00%) dev_err 23.60% (+:0.00%)

k = 33 train_err 18.24% (+:19.38%) dev_err 15.70% (+:18.50%)
```

Q: When k = 1, is training error 0%? Why or why not? Look at the training data to confirm your answer.

When K = 1 we are using only one record to predict the salary, and it's much less likely we will get the correct value. It is not 0% however, because we are still using one value to predict.

5. Q: What trends (train and dev error rates and positive ratios, and running speed) do you observe with increasing k? Do they relate to underfitting and overfitting?

There is certainly a curve to the model accuracy. As K increases it will certainly take the model more time to run. Overfitting will definitely occur at k = n, or 5,000 k's, where positive rate is 0% and training error is 25% (population average). This happens because there are more negative than positive responses, and when the k's are larger than the number of records in the data, the algorithm will use the entire data set to predict salary and gets the same value each time (non-positive).

As the value of K increases from 1 to infinity, the error rate will decline to the lowest point, likely between k = 5 and k = 100, then rise again as the model progresses to overfitting.

Q: What does k = ∞ actually do? Is it extreme overfitting or underfitting? What about k = 1?

K = inf is extreme overfitting and will have error = 0 and k-nn prediction = average of the whole sample. k = 1 is extreme underfitting and will have high error and a wildly inaccurate prediction.

6. Redo the evaluation using Manhattan distance. Better or worse? Any advantage of Manhattan distance?

The numbers are the same, no better, no worse. It does take a bit longer to run the code with Manhattan distance which is worse.

```
k = 1 train_err 15.20% (+:22.14%) dev_err 22.70% (+:21.90%)
k = 3 train_err 16.86% (+:20.44%) dev_err 19.20% (+:19.60%)
k = 5 train_err 17.16% (+:18.66%) dev_err 18.30% (+:18.10%)
k = 7 train_err 17.74% (+:18.56%) dev_err 18.00% (+:18.20%)
k = 9 train_err 17.08% (+:20.94%) dev_err 16.80% (+:20.60%)
k = 33 train_err 18.24% (+:19.38%) dev_err 15.70% (+:18.50%)
k = 99 train_err 18.28% (+:18.54%) dev_err 16.90% (+:17.90%)
k = 999 train_err 21.22% (+:7.00%) dev_err 18.60% (+:7.40%)
k = 9999 train_err 25.02% (+:0.00%) dev_err 23.60% (+:0.00%)
```

7. Redo the evaluation using all-binarized features (with Euclidean). Better or worse? Does it make sense?

```
duration is: 429.9788727760315 seconds.
k = 1 super train_err 0.00% (+:25.02%) super dev_err 1.80% (+:23.00%)
k = 3 super train_err 0.38% (+:24.76%) super dev_err 0.40% (+:23.40%)
k = 5 super train_err 0.42% (+:24.76%) super dev_err 0.60% (+:23.20%)
k = 7 super train_err 0.42% (+:24.76%) super dev_err 0.50% (+:23.10%)
k = 9 super train_err 0.42% (+:24.72%) super dev_err 0.50% (+:23.10%)
k = 99 super train_err 0.48% (+:24.54%) super dev_err 0.80% (+:22.80%)
k = 999 super train_err 1.96% (+:23.06%) super dev_err 1.90% (+:21.70%)
k = 9999 super train_err 25.02% (+:0.00%) super dev_err 23.60% (+:0.00%)
```

This took 430 seconds, which is much slower. The performance of the model is similar in percent positive, and very small in error. This likely means that the model is over-fitting the data.

PART 4 Deployment

Now try more k's and take your best model and run it on the semi-blind test data, and produce income.test.predicted, which has the same format as the training and dev files.

Q: At which k and with which distance did you achieve the best dev results?

The best performance occurs using Euclidean Distance and 33 K's.

Q: What's your best dev error rates and the corresponding positive ratios?

Best dev error rate is 15.7%, corresponding positive rate is 18.5%.

Q: What's the positive ratio on test?

The positive ratio according to validate.py is 18.6%

Validate.py output:

Your positive rate is 18.6%.

Your positive rate seems reasonable.

Your prediction file has been validated for formatting.

This does not guarantee that your prediction accuracy will be good though.

PART 5 Observations

1. Q: Summarize the major drawbacks of k-NN that you observed by doing this HW. There are a lot!

The major k-NN drawbacks are that it is a very computationally expensive algorithm that doesn't provide the best results. It's fairly trivial to understand, and one of the only machine learning methods that you can really visualize what it's doing well, but it's computationally difficult to implement, and can't be used well for predictions. K-NN also requires the storage of the whole dataset in memory, which makes it impossible to use for large data sets.

2. Q: Do you observe in this HW that best-performing models tend to exaggerate the existing bias in the training data? Is it due to overfitting or underfitting? Is this a potentially social issue?

Machine learning algorithms, KNN especially, tend to exaggerate training bias. When a model is overfit it will further exaggerate training bias, however my model has selected a k-value that hopefully is robust to that. This is also potentially a social issue, and I could certainly check this by examining how specific data points behave VS the model prediction, but it's tired, and I'm late.

3. Q: What numpy tricks did you use to speed up your program so that it can be fast enough to print the training error? Hint: (a) broadcasting (such as matrix - vector); (b) np.linalg.norm(..., axis=1); (c) np.argsort() or np.argpartition(); (d) slicing. The main idea is to do as much computation in the vector-matrix format as possible (i.e., the Matlab philosophy), and as little in Python as possible.

I used mostly broadcasting and np.linalg.norm to speed up my algorithm. I used anything I could find on slack.

4. **How many seconds does it take to print the training and dev errors for k = 99 on ENGR servers? Hint: use time python ... and report the user time instead of the real time. (Mine was about 14 seconds).**

I was not able to get onto the ENGR servers but using jupyter notebook on a local server my model took 160.86 seconds.

5. What is a Voronoi diagram (shown in k-NN slides)? How does it relate to k-NN?

A Voronoi diagram is one of the best ways to visualize the k-NN method. You can think of it as a mosaic, where each mosaic piece (Vronoi Cell) is the region defined by a nearest-neighbor outcome.

Debriefing (required in your report)

1. Approximately how many hours did you spend on this assignment?

A lot. I time myself when working on classes and I spent 25 hours of time on this assignment in the first three weeks of the class.

2. Would you rate it as easy, moderate, or difficult?

Immensely difficult. I consider myself pretty good at code for a data person. Professionally I am currently a Data Analyst for a LinkedIn company and code for them every day. I fairly frequently implement AI algorithms in my work, using the typical AI packages. In my academic life I performed very well in both CS 511 and 512, getting 101% and 97.5% in each class respectively.

I understand the need to prevent the use of pandas/scikitlearn in this assignment. It certainly taught me a lot, but most of it was through other students and going to office hours. Many of the questions I didn't even understand what it was asking, and it was VERY different from lecture. There was a lecture where we were walked through binarization, but it just didn't work on my computer, and nothing past part 3 was discussed anywhere in the class.

Honestly I feel like I'm missing a massive section of content, and this whole class relies on some knowledge that I just don't have. The only way I made it through the homework is because of office hours, which not everyone can attend. I saw the same 10 people in office hours every time I went, and I feel like many of the other students were left out for one reason or another.

3. Did you work on it mostly alone, or mostly with other people?

Mostly alone with a bit of help from the TA's and other students.

4. How deeply do you feel you understand the material it covers (0%–100%)?

I understand K-NN and Euclidean distance 100%, and I understand how to call pd.getdummies 100%, probably 15% better than I did before this homework.

I absolutely have no idea how binarization in numpy works, and how/what the mapping function does.

5. Any other comments?

- Start the class with the windows guide. I lost about a week because I could not follow lecture with my windows computer.
- Ask questions that you teach us how to answer. I felt completely unprepared on most of these questions. Most of the lecture is "here is numpy" and the homework is expecting a high level of numpy knowledge.

- I appreciate the TA's willingness to help, the slack channel has been very helpful. Maybe figure out why so few are showing up to office hours. Are the other students just awesome and have no issues? Or maybe they are struggling and just can't make the times work? I don't know.