4.2 Distances between Categorical Variables

May 9, 2019

1 Chapter 4.2 Distance Metrics and Categorical Variables

The distance metrics that we studied in the previous section were designed for quantitative variables. But most data sets contain a mix of categorical and quantitative variables. For example, the Titanic data set contains both quantitative variables, like `age`, and categorical variables, like `sex` and `embarked`. How do we measure the similarity between observations for a data set like this one? The most straightforward solution is to convert the categorical variables into quantitative ones.

```
In [1]: %matplotlib inline
        import numpy as np
        import pandas as pd
        pd.options.display.max_rows = 5

        titanic = pd.read_csv("https://raw.githubusercontent.com/dlsun/data-science-book/master
        titanic
```

Out[1]:

	pclass	survived	name	sex	age	\
0	1	1	Allen, Miss. Elisabeth Walton	female	29.0000	
1	1	1	Allison, Master. Hudson Trevor	male	0.9167	
...	...	...	...	...	...	
1307	3	0	Zakarian, Mr. Ortin	male	27.0000	
1308	3	0	Zimmerman, Mr. Leo	male	29.0000	

	sibsp	parch	ticket	fare	cabin	embarked	boat	body	\
0	0	0	24160	211.3375	B5	S	2	NaN	
1	1	2	113781	151.5500	C22 C26	S	11	NaN	
...	...	...	...	...	...	...	...	...	
1307	0	0	2670	7.2250	NaN	C	NaN	NaN	
1308	0	0	315082	7.8750	NaN	S	NaN	NaN	

	home.dest
0	St Louis, MO
1	Montreal, PQ / Chesterville, ON
...	...
1307	NaN
1308	NaN

```
[1309 rows x 14 columns]
```

1.1 Converting Categorical Variables to Quantitative Variables

Binary categorical variables (categorical variables with two categories) can be converted into quantitative variables by coding one category as 1 and the other category as 0. (In fact, the `survived` column in the Titanic data set is an example of a variable where this has been done.) But what do we do about a categorical variable with more than 2 categories, like `embarked`, which has 3 categories?

We can convert a categorical variable with K categories into K separate 0/1 variables, or **dummy variables**. Each of the K variables is an indicator for one of the K categories. That is, each dummy variable is 1 if the observation fell into that category and 0 otherwise.

Although it is not difficult to create dummy variables manually, the easiest way to create them is the `get_dummies()` function in `pandas`.

```
In [2]: pd.get_dummies(titanic["embarked"])
```

```
Out[2]:       C  Q  S
        0     0  0  1
        1     0  0  1
        ...  .. .. ..
        1307  1  0  0
        1308  0  0  1

        [1309 rows x 3 columns]
```

Since every observation is in exactly one category, each row contains exactly one 1; the rest of the values in each row are 0s.

We can call `get_dummies` on a `DataFrame` to encode multiple categorical variables at once. `pandas` will only dummy-encode the variables it deems categorical, leaving the quantitative variables alone. If there are any categorical variables that are represented in the `DataFrame` using numeric types, they must be cast explicitly to a categorical type, such as `str`. `pandas` will also automatically prepend the variable name to all dummy variables, to prevent collisions between column names in the final `DataFrame`.

```
In [3]: # Convert pclass to a categorical type
        titanic["pclass"] = titanic["pclass"].astype(str)

        # Pass all variables to get_dummies, except ones that are "other" types
        titanic_num = pd.get_dummies(
            titanic.drop(["name", "ticket", "cabin", "boat", "body"], axis=1)
        )
        titanic_num
```

```
Out[3]:       survived      age  sibsp  parch      fare  pclass_1  pclass_2  pclass_3  \
        0            1  29.0000      0      0  211.3375         1         0         0
        1            1   0.9167      1      2  151.5500         1         0         0
        ...        ...      ...    ...    ...       ...       ...       ...       ...
```

```
1307         0  27.0000      0      0    7.2250         0         0         1
1308         0  29.0000      0      0    7.8750         0         0         1

      sex_female  sex_male              ...                \
0              1         0              ...
1              0         1              ...
...          ...       ...              ...
1307           0         1              ...
1308           0         1              ...

      home.dest_Wimbledon Park, London / Hayling Island, Hants  \
0                                                     0
1                                                     0
...                                                 ...
1307                                                  0
1308                                                  0

      home.dest_Windsor, England New York, NY  home.dest_Winnipeg, MB  \
0                                          0                       0
1                                          0                       0
...                                      ...                     ...
1307                                       0                       0
1308                                       0                       0

      home.dest_Winnipeg, MN  home.dest_Woodford County, KY  \
0                          0                             0
1                          0                             0
...                      ...                           ...
1307                       0                             0
1308                       0                             0

      home.dest_Worcester, England  home.dest_Worcester, MA  \
0                                0                        0
1                                0                        0
...                            ...                      ...
1307                             0                        0
1308                             0                        0

      home.dest_Yoevil, England / Cottage Grove, OR  home.dest_Youngstown, OH  \
0                                                 0                         0
1                                                 0                         0
...                                             ...                       ...
1307                                              0                         0
1308                                              0                         0

      home.dest_Zurich, Switzerland
0                                 0
1                                 0
```

```
...                                 ...
1307                                  0
1308                                  0

[1309 rows x 382 columns]
```

Notice that categorical variables, like `pclass`, were converted to dummy variables with names like `pclass_1`, `pclass_2` and `pclass_3`, while quantitative variables, like `age`, were left alone.

Now that we have converted every variable in our data set into a quantitative variable, we can apply the techniques from the previous section (Section 4.1) to calculate distances between observations. For example, to find the passenger who is most similar to the first passenger, Elisabeth Watson, we can find the row with the smallest Euclidean distance to that row in the above `DataFrame`.

```
In [4]: titanic_std = (titanic_num - titanic_num.mean()) / titanic_num.std()
        np.sqrt(
            ((titanic_std - titanic_std.loc[0]) ** 2).sum(axis=1)
        ).sort_values()

Out[4]: 0       0.000000
        238     1.509375
                 ...
        472    41.186785
        511    41.209435
        Length: 1309, dtype: float64
```

The passenger who was most similar to Elisabeth Allen, other than herself, is passenger 238. Let's extract these passengers from the original `DataFrame` to see how similar they really are.

```
In [5]: titanic.loc[[0, 238]]

Out[5]:     pclass  survived                                             name  \
        0        1         1                         Allen, Miss. Elisabeth Walton
        238      1         1  Robert, Mrs. Edward Scott (Elisabeth Walton Mc...

                sex   age  sibsp  parch ticket      fare cabin embarked boat  body  \
        0    female  29.0      0      0  24160  211.3375    B5        S    2   NaN
        238  female  43.0      0      1  24160  211.3375    B3        S    2   NaN

                home.dest
        0    St Louis, MO
        238  St Louis, MO
```

The two passengers are indeed very similar, only differing in age and the number of parents/children accompanying her. They even happen to share the same first two names ("Elizabeth Walton").

2 Exercises

Exercises 1 and 2 use the Ames housing data set (`https://raw.githubusercontent.com/dlsun/data-science-b`

Exercise 1. The neighborhood variable (`Neighborhood`) in this data set is categorical. Convert it to K quantitative variables. What is K in this case?

Based on these K variables only, calculate the Euclidean distance between house 0 and each of the other houses in the data set. What are the possible values of the Euclidean distance? Can you explain what a distance of 0 means, in the context of this variable? What about a distance of 1?

```
In [10]: housing = pd.read_csv("https://raw.githubusercontent.com/dlsun/data-science-book/maste
                               sep="\t")
```

```
In [16]: neighborhood_dummies = pd.get_dummies(housing.Neighborhood)
         neigbhorhood_dummies

         np.sqrt(((neighborhood_dummies - neighborhood_dummies.loc[0]) ** 2).sum(axis=1))
```

```
Out[16]: 0       0.000000
         1       0.000000
                   ...
         2928    1.414214
         2929    1.414214
         Length: 2930, dtype: float64
```

Exercise 2. Suppose that you really like house 0 in the data set, but it is too expensive. Find cheaper homes that are similar to it, by calculating distances after encoding categorical variables as dummy variables. Be sure to actually look at the profiles of the homes that your algorithm picked out as most similar. Do they make sense?

Try different distance metrics and different standardization methods. How sensitive are your results to these choices?

Think: If the goal is to find a "good deal" on a similar house, should sale price be included as a variable in your distance metric?

Hint: There are too many variables in the data set. Do not try to call `pd.get_dummies()` on the entire `DataFrame`! You will want to pare down the number of variables, but be sure to include a mixture of categorical and quantitative variables. Refer to the data documentation for information about the variables.

```
In [31]: housing_num = pd.get_dummies(
             housing.drop(["PID", "MS SubClass", "MS Zoning", "Street", "Alley", "Land Slope",
                           "Condition 2"], axis=1))
         housing_num


         dist_norm = np.sqrt(((housing_num - housing_num.loc[0]) ** 2).sum(axis=1)).sort_values

         print(dist_norm.index[:5])
         housing_num.iloc[dist_norm.index[:5]]
```

```
Int64Index([0, 2686, 2224, 2222, 1016], dtype='int64')
```

Out[31]:

	Order	Lot Frontage	Lot Area	Overall Qual	Overall Cond	Year Built
0	1	141.0	31770	6	5	1960
2686	2687	90.0	33120	6	5	1962
2224	2225	NaN	21579	6	6	1968
2222	2223	NaN	21000	6	5	1953
1016	1017	46.0	20544	7	6	1986

	Year Remod/Add	Mas Vnr Area	BsmtFin SF 1	BsmtFin SF 2
0	1960	112.0	639.0	0.0
2686	1962	0.0	0.0	0.0
2224	1968	0.0	813.0	0.0
2222	1953	184.0	35.0	869.0
1016	1991	123.0	0.0	0.0

	...	Sale Type_New	Sale Type_Oth	Sale Type_VWD
0	...	0	0	0
2686	...	0	0	0
2224	...	0	0	0
2222	...	0	0	0
1016	...	0	0	0

	Sale Type_WD	Sale Condition_Abnorml	Sale Condition_AdjLand
0	1	0	0
2686	1	0	0
2224	0	0	0
2222	0	1	0
1016	1	0	0

	Sale Condition_Alloca	Sale Condition_Family	Sale Condition_Normal
0	0	0	1
2686	0	0	1
2224	0	0	1
2222	0	0	0
1016	0	0	1

	Sale Condition_Partial
0	0
2686	0
2224	0
2222	0
1016	0

[5 rows x 274 columns]