CSC 240 – Project 2: Calculating Distance Between Items

February 5, 2022

Sammy Potter (spott14@u.rochester.edu)
Justin Lee (jlee363@u.rochester.edu)
CSC 240

1. Main (similarity() and read_data())

Our main function first normalizes all numeric values in the dataset using our 'read_data()' method, then runs the 'similarity()' method, passing two tuples from the dataset. 'Similarity()' calculates the similarity between the two entries for each attribute using the appropriate method. The results are stored in a list, and the mean value of that list is calculated and returned.

SEGMENT I: NORMALIZATION OF DATA AND MAIN FUNCTION

```
def read_data():
  dataframe = pd.read_csv("adult.data")
  dataframe.columns = ["age", "work-class", "fnlwgt", "education", "education-num",
                       "martial-status", "occupation", "relationship",
                       "race", "sex", "capital-gain", "capital-loss",
                       "hours-per-week", "native-country", "wage"]
  mms = MinMaxScaler()
  dataframe["age"] = mms.fit_transform(dataframe[["age"]])
  dataframe["fnlwgt"] = mms.fit_transform(dataframe[["fnlwgt"]])
  dataframe["education-num"] = mms.fit_transform(dataframe[["education-num"]])
  dataframe["capital-gain"] = mms.fit_transform(dataframe[["capital-gain"]])
  dataframe["capital-loss"] = mms.fit_transform(dataframe[["capital-loss"]])
  dataframe["hours-per-week"] = mms.fit_transform(dataframe[["hours-per-week"]])
```

```
  return dataframe


def similarity(x, y):
  sims = [
    numeric(x[0], y[0]),    # Age
    ordinal(x[1], y[1]),    # Workclass
    numeric(x[2], y[2]),    # Fnlwgt
    ordinal(x[3], y[3]),    # Education
    numeric(x[4], y[4]),    # Education-num
    ordinal(x[5], y[5]),    # Marital-status
    nominal(x[6], y[6]),    # Occupation
    nominal(x[7], y[7]),    # Relationship
    nominal(x[8], y[8]),    # Race
    nominal(x[9], y[9]),    # Sex
    numeric(x[10], y[10]),  # Capital-gain
    numeric(x[11], y[11]),  # Capital-loss
```

---

2. Attribute Functions

2.1 Age

The 'age' attribute is a numeric attribute. Therefore, in order to find the similarity in the 'age' attribute between two entries, we calculate the Euclidean distance between their respective normalized ages. We abstracted a Euclidean distance function, as we call it multiple times throughout the project. Our Euclidean distance function follows:

SEGMENT II: AGE (NUMERICAL)

```
def numeric(x, y):
  total = ((x - y) ** 2)
  return 1 - math.sqrt(total)
```

2.2 Workclass

While the obvious solution might have been to interpret the 'workclass' attribute as nominal, we decided to interpret workclass as an ordinal attribute, as different work classes often coincide or overlap with each other: information that would have been lost if 'workclass' was analyzed as a nominal attribute. For example, the attributes "Self-emp-not-inc" and "Self-emp-inc" both indicate a degree of self employment. Similarly, the attributes "Federal-gov," "Local-gov," and "State-gov" indicate a degree of employment in government. As such, we run the following abstracted function in order to calculate the Jaccard coefficient between two entries' 'workclass' attribute:

SEGMENT III: WORKCLASS (ORDINAL)

```
def ordinal(x, y):
  a = x.split("-")
  b = y.split("-")
  intersection = len(list(set(a).intersection(set(b))))
  union = (len(a) + len(b)) - intersection
  return float(intersection) / union
```

2.3 Final Weight (Fnlwgt)

The 'fnlwgt' attribute is another numeric attribute. As such, it is treated the same as the 'age' attribute, and the similarity between two entries' 'fnlwgt' attribute is calculated using the Euclidean distance function. While we already include the Euclidean distance function on a previous page, we also include it here as follows as per the assignment instructions:

SEGMENT IV: FINAL WEIGHT (NUMERIC)

```
def numeric(x, y):
  total = ((x - y) ** 2)
```

```
    return 1 - math.sqrt(total)
```

2.4 Education

The 'education' attribute is another ordinal attribute, as there is a chronological sequence to the attribute. As such, it is treated the same as the 'workclass' attribute, and the similarity between two entries' 'education' attribute is calculated using the Jaccard coefficient function. While we already include the Jaccard coefficient function on a previous page, we also include it here as follows as per the assignment instructions:

SEGMENT V: EDUCATION (NUMERIC)

```
def ordinal(x, y):
  a = x.split("-")
  b = y.split("-")
  intersection = len(list(set(a).intersection(set(b))))
  union = (len(a) + len(b)) - intersection
  return float(intersection) / union
```

2.5 Education-num

The 'education-num' attribute is another numeric attribute. As such, it is treated the same as the 'fnlwgt' attribute, and the similarity between two entries' 'education-num' attribute is calculated using the Euclidean distance function. While we already include the Euclidean distance function on a previous page, we also include it here as follows as per the assignment instructions:

SEGMENT V: EDUCATION-NUM (NUMERIC)

```
def numeric(x, y):
  total = ((x - y) ** 2)
  return 1 - math.sqrt(total)
```

2.6 Marital-status

The 'martial-status' attribute is another ordinal attribute, as there are similarities between certain statuses. For example, "Married-civ-spouse" and "Married-spouse-absent" both indicate a person is married. As such, it is treated similarly to the 'education' attribute, and the similarity between two entries' 'martial-status' attribute is calculated using the Jaccard coefficient function. While we already include the Jaccard coefficient function on a previous page, we also include it here as follows as per the assignment instructions:

SEGMENT VI: MARTIAL-STATUS (ORDINAL)

```
def ordinal(x, y):
  a = x.split("-")
  b = y.split("-")
  intersection = len(list(set(a).intersection(set(b))))
  union = (len(a) + len(b)) - intersection
  return float(intersection) / union
```

2.7 Occupation

We interpreted the 'occupation' attribute as a nominal attribute, as we did not find any useful similarities between the attribute values that would have worked with our ordinal function. The nominal function follows:

SEGMENT VI: OCCUPATION (NOMINAL)

```
def nominal(x, y):
  return int(x == y)
```

2.8 Relationship

We also interpreted the 'relationship' attribute as a nominal attribute, as again we did not find any useful similarities between the attribute values that would have worked with our ordinal function. The nominal function follows:

SEGMENT VII: RELATIONSHIP (NOMINAL)

```
def nominal(x, y):
  return int(x == y)
```

2.9 Race

We also interpreted the 'race' attribute as a nominal attribute, as again we did not find any useful similarities between the attribute values that would have worked with our ordinal function.

SEGMENT VIII: RACE (NOMINAL)

```
def nominal(x, y):
  return int(x == y)
```

2.10 Sex

Since there are only two possible values in the data for the 'sex' attribute, we interpret 'sex' as a binary attribute, returning 1 for equality. We reuse our nominal method, as it provides equivalent functionality.

SEGMENT IX: SEX (BINARY)

```
def nominal(x, y):
  return int(x == y)
```

2.11 Capital Gain

The 'capital-gain' attribute is a numeric attribute, as it is a measurable quantity. Therefore, in order to find the similarity in the 'capital-gain' attribute between two entries, we calculate the Euclidean distance between their respective normalized values. Once again, the function is as follows:

SEGMENT X: CAPITAL GAIN (NUMERIC)

```
def numeric(x, y):
  total = ((x - y) ** 2)
  return 1 - math.sqrt(total)
```

2.12 Capital Loss

Similar to the 'capital-gain' attribute, we interpreted the 'capital-loss' attribute as a numeric value for the same reasons. Once again, we use the Euclidean distance function on the two entries' 'capital-loss' values:

SEGMENT XI: CAPITAL LOSS (NUMERIC)

```
def numeric(x, y):
  total = ((x - y) ** 2)
  return 1 - math.sqrt(total)
```

2.13 Hours-per-Week

We interpret 'hours-per-week' as a numeric value because it is a measurable quantity. As such, we apply our Euclidean function.

SEGMENT XII: HOURS PER WEEK (NUMERIC)

```
def numeric(x, y):
  total = ((x - y) ** 2)
```

```
    return 1 - math.sqrt(total)
```

2.14 Native Country

While the 'native-country' attribute could be considered as a nominal attribute, we thought that that analysis would have resulted in lost information. For example, while the attribute values 'Cambodia' and 'Philippines' are not the same as nominal attributes, they are more similar in culture and history than the 'United-States' and 'China.' As such, we decided to use the Python library 'Geopy' in order to calculate the geographical distance between two locations and divide it by half of the equator's distance, as that is the furthest distance that could be measured on a sphere. This returns a value between 0 and 1, with 1 representing the closest distance and 0 representing the furthest distance. The function follows:

SEGMENT XIII: NATIVE COUNTRY (NUMERIC)

```
def geo_dist(x, y):
  if x == y: return 1
  geolocator = Nominatim(user_agent="GoogleV3")
  spec_country = {
     "Outlying-US(Guam-USVI-etc)": "Guam",
     "Trinadad&Tobago": "Trinidad" }

  x_country = geolocator.geocode(x if x not in spec_country else spec_country[x])
  y_country = geolocator.geocode(y if y not in spec_country else spec_country[y])
  x_coord = (x_country.latitude, x_country.longitude)
  y_coord = (y_country.latitude, y_country.longitude)

  # The furthest you can get from another point on Earth is
  # half its circumference (km)
  furthest = 40075/2

  return 1 - ((GD(x_coord, y_coord).km)/furthest)
```