Vertica ML Python Workshop

Exercise 4: Decomposition & Normalization

Ouali Badr

December 5, 2019

Executive Summary

> *"Science knows no country, because knowledge belongs to humanity, and is the torch which illuminates the world."*
>
> **Louis Pasteur**

VERTICA ML PYTHON allows the users to use Vertica advanced analytics and Machine Learning with a Python front-end Interface. In this exercise, you'll learn some basics to begin your fantastic Data Science Journey with the API. As a summary:

- Normalize the data using the different methods
- Create a PCA model
- Find a suitable number of principal components to keep

Contents

1 Presentation

Normalizing the data can be very important in a lot of different cases. For example, many ML algorithms are sensible to un-normalized data because they are using the p-distance to learn (Neighbors, KMeans...). It can create complications for the algorithms to converge. When it comes to data sharing, normalization is a way to encode the data and to keep the data distribution. When we know the estimators used to normalize the data, we can easily un-normalize the data and come back to the original distribution which makes it easy to encode/decode when we have access to the keys. We can identify 3 main methods to normalize the data:

- **Zscore:** Average. We reduce and center the feature using the average and standard deviation. This normalization is sensible to outliers.
- **Robust Zscore:** Median. We reduce and center the feature using the median and the median absolute deviation. This normalization is robust to outliers.
- **Minmax:** Min/Max. We reduce the feature by using a bijection to [0,1] The max will reach 1 and the min 0. This normalization is robust to outliers.

Some algorithms are sensible to correlated predictors, the PCA can be a solution before applying the algorithm. The PCA uses an orthogonal transformation to convert a set of observations of possibly correlated variables (entities each of which takes on various numerical values) into a set of values of linearly uncorrelated variables called principal components. Some algorithms are also sensible to the number of predictors. Only keeping valuable information is a way to solve the problem. PCA can also be a solution to encode the data.

During this exercise, we will use different techniques of normalization and the PCA for decomposition. We will use the heart disease dataset. This dataset contains many information of 270 patients including:

- **age:** age in years
- **thalach:** maximum heart rate achieved
- **trestbps:** resting blood pressure (in mm Hg on admission to the hospital)
- **fbs:** (fasting blood sugar > 120 mg/dl) (1 = true; 0 = false)
- **slope:** the slope of the peak exercise ST segment (Value 1: upsloping; Value 2: flat; Value 3: downsloping)
- **ca:** number of major vessels (0-3) colored by flourosopy
- **num:** diagnosis of heart disease (angiographic disease status) (Value 1: < 50% diameter narrowing; Value 2: > 50% diameter narrowing)
- **sex:** sex (1 = male; 0 = female)
- **cp:** hest pain type (Value 1: typical angina; Value 2: atypical angina; Value 3: non-anginal pain; Value 4: asymptomatic)
- **exang:** exercise induced angina (1 = yes; 0 = no)
- **restecg:** resting electrocardiographic results (Value 0: normal; Value 1: having ST-T wave abnormality (T wave inversions and/or ST elevation or depression of > 0.05 mV); Value 2: showing probable or definite left ventricular hypertrophy by Estes' criteria)
- **chol:** serum cholestoral in mg/dl
- **oldpeak:** ST depression induced by exercise relative to rest
- **thal:** 3 = normal; 6 = fixed defect; 7 = reversable defect

The purpose is to find people having heart complications compare to the rest.

2 Functions used during the Exercise

2.1 normalize

Library: vertica_ml_python.vDataframe[]

```
vDataframe[].normalize(self, method: str = "zscore")
```

normalize the column with a specific method.

Parameters

- **method:** *<str>*, optional
  zscore (default) | robust_zscore | minmax
  The method used for the normalization.

Returns

The parent vDataframe.

Example

```
from vertica_ml_python.learn.datasets import load_titanic
titanic = load_titanic(cur)
titanic["fare"].normalize()

#Output
                                    fare
0      2.2335228377568673306163744003
1      2.2335228377568673306163744003
2      2.2335228377568673306163744003
3     -0.6451344183800711827483251581
4      0.2951864297839290254556257484
...                                ...
Name: fare, Number of rows: 1234, dtype: float
```

2.2 PCA

Create a PCA object by using the Vertica Highly Distributed and Scalable PCA on the data.

initialization

Library: vertica_ml_python.learn.decomposition

```
class PCA(
    name: str,
    cursor,
```

```
    n_components: int = 0,
    scale: bool = False,
    method: str = "Lapack")
```

Parameters

- **name:** *<str>*
  Name of the model.
- **cursor:** *<object>*
  DB cursor.
- **n_components:** *<int>*, optional
  The number of components to keep in the model. If this value is not provided, all components are kept. The maximum number of components is the number of non-zero singular values returned by the internal call to SVD. This number is less than or equal to SVD (number of columns, number of rows).
- **scale:** *<bool>*, optional
  A Boolean value that specifies whether to standardize the columns during the preparation step: If true use a correlation matrix instead of a covariance matrix.
- **method:** *<str>*, optional
  The method used to calculate PCA, can be set to LAPACK.

Methods

The PCA object has 3 methods:

```
# Drop the model from the DB
def drop(self)

# Fit the model with the input columns
def fit(self, input_relation: str, X: list)

# Build a vdf from the output relation
def to_vdf(self, n_components: int = 0,  cutoff: float = 1, key_columns: list
    = [], inverse: bool = False)
```

Attributes

The PCA object has only 3 attributes:

```
self.components # The principal components
self.explained_variance # The information about singular values found.
self.mean # The information about columns from the input relation used for
    creating the PCA model
```

Example

```
from vertica_ml_python.learn.decomposition import PCA

# We can build the model
model = PCA("pca_iris", cur)
```

```
model.fit("iris", ["SepalLengthCm", "SepalWidthCm", "PetalWidthCm", "
    PetalLengthCm"])

# We can evaluate the model
model.explained_variance

# Output
                         value    explained_variance    \\
1           2.05544174529956     0.924616207174268    \\
2          0.492182457659266    0.0530155678505349    \\
3          0.280221177097938    0.0171851395250067    \\
4          0.153892907978245    0.0051830854501896    \\
           accumulated_explained_variance
1                        0.924616207174268
2                        0.977631775024804
3                         0.99481691454981

# We can export the model
model.to_vdf(n_components = 2)

# Output
                   col1                   col2
0    -2.68420712510395     0.326607314764391
1    -2.71539061563413    -0.169556847556024
2    -2.88981953961792    -0.137345609605025
3    -2.74643719730873    -0.311124315751989
4    -2.72859298183131     0.333924563568457
...                 ...                    ...
Name: pca_table_iris, Number of rows: 150, Number of columns: 2
```

3 Questions

Turn on Jupyter with the 'jupyter notebook' command. Start the notebook exercise4.ipynb and answer to the following questions.

- **Question 1:** Look at the descriptive statistics of all the elements and compare the median and the average. What do you notice ? Explain why the 'zscore' is a suitable way to normalize the data. Normalize the data.
- **Question 2:** If we decide to build a model, these columns could be correlated one to another. PCA allows a space reduction and create independent components. Create a PCA model using these features.
- **Question 3:** Look at the explained_variance attribute and choose a number of components to keep (we usually consider to keep components which bring the accumulated explained variance to more than 0.95)
- **Question 4:** Why is it important to reduce the number of features before fitting a ML model ?