Principal Component Analysis (PCA) in Feature Engineering

Lesson Objectives

- After completing this lesson, you should be able to:
  - Understand what Principal Component Analysis (PCA) is
  - Understand PCA's role in feature engineering

PCA: definition

•PCA is a dimension reduction technique. It is unsupervised machine learning, and it has many uses; on this video we only care about its use for feature engineering

PCA: how it works

- The first Principal Component (PC) is defined as the linear combination of the predictors that **captures the most variability of all possible linear combinations**.

- Then, subsequent PCs are derived such that these linear combinations **capture the most remaining variability while also being uncorrelated** with all previous PCs.

Feature Engineering

- "Feature engineering" is a practice where predictors are created and refined to maximize model performance
- It can take quite some time to identify and prepare relevant features

Feature Engineering with PCA

- Basic idea: generate a smaller set of variables that capture most of the information in the original variables
- The new predictors are functions of the original predictors; all the original predictors are still needed to create the surrogate variables

Loading the data

```
!wget https://raw.githubusercontent.com/apache/spark/master/data/mllib/sample_libsvm_data.txt
```

```
--2016-09-25 20:32:33--  https://raw.githubusercontent.com/apache/spark/master/data/mllib/sample_libsvm_data.txt
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 151.101.12.133
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|151.101.12.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 104736 (102K) [text/plain]
Saving to: ‘sample_libsvm_data.txt.1’

100%[======================================>] 104.736     --.-K/s   in 0,09s

2016-09-25 20:32:33 (1,11 MB/s) - ‘sample_libsvm_data.txt.1’ saved [104736/104736]
```

```
from pyspark.mllib.util import MLUtils

data = MLUtils.loadLibSVMFile(sc, 'sample_libsvm_data.txt')
```

Apply PCA and interpret result

```
from pyspark.mllib.feature import PCA
model = PCA(10).fit(rddOfVectors)
pc = model.transform(rddOfVectors)
pc.take(5)
```

```
[DenseVector([1.2139, 0.5646, -0.0223, -0.3093, -0.8195, -0.4247, -0.41, 0.6418, 0.1997, 0.5075]),
 DenseVector([0.628, 1.1669, -0.5142, -0.2483, -0.6245, -0.1183, -0.4505, 1.2456, 0.0253, 0.6838]),
 DenseVector([0.2343, 0.3481, 0.5488, -0.4821, -0.4085, -0.9437, 0.116, 1.0367, -0.0473, 0.2083]),
 DenseVector([1.4381, 1.1669, 0.9339, -0.7461, -1.4831, -1.5778, -0.2062, 0.8903, -0.5252, 0.3207]),
 DenseVector([1.9063, 0.1724, 1.0708, -0.5156, -0.6947, -0.7162, -0.6535, 0.6082, -0.0544, 0.6421])]
```

- Principal components are stored in a local dense matrix.
- The matrix pc is now 10 dimensions, but it represents the variability ‘almost as well’ as the previous 100 dimensions

Pros

- Interpretability (!)
- PCA creates components that are uncorrelated, and Some predictive models prefer little to no collinearity (example linear regression)
- Helps avoiding the '*curse of dimensionality*': Classifiers tend to overfit the training data in high dimensional spaces, so reducing the number of dimensions may help

Pros (2)

•Performance. On further modeling, the computational effort often depends on the number of variables. PCA gives you far fewer variables; this may make any further processing more performant

•For classification problems PCA can show potential separation of classes (if there is a separation).

Cons

•The computational effort often depends greatly on the number of variables and the number of data records.

•PCA seeks linear combinations of predictors that maximize variability, **it will naturally first be drawn to summarizing predictors that have more variation**.

How many principal components to use?

- No simple answer to this question
- But there are heuristics:
  - –find the elbow on the graph for dimensions by variance explained
  - –Set up a ‘variance explained threshold’ (for example, take as many Principal components as needed to explain 95% of the variance

Tip for Best Practice

•Always center and scale the predictors prior to performing PCA (see previous course). Otherwise the **predictors that have more variation will soak the top principal components**

Lesson Summary

- Having completed this lesson, you should be able to:
  - Apply PCA in Spark
  - Use PCA to fix datasets with correlated predictors that could otherwise trip your models!