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

Dataset: predict US crimes

- We want to predict the proportion of violent crimes per 100K population on different locations in the US

- More than 100 predictors. Examples:
  - householdsize: mean people per household
  - racepctblack: percentage of population that is African American
  - pctWWage: percentage of households with wage or salary income in 1989

- For a description of the variables, see the UCI repository (communities and crimes)

Dataset: predict US crimes

- Let's assume that we don't want to operate with those >100 predictors. Why?
  - Some will be collinear (ie highly correlated)
  - It's hard to see relationships in a high-dimensional space

- How do we use PCA to get down to 10 dimensions?

Loading the csv into a dataframe

```
!wget https://s3.eu-central-1.amazonaws.com/dsr-data/UScrime/UScrime2-colsLotsOfNAremoved.csv

crimes = sqlc.read.format("com.databricks.spark.csv") \
            .option("delimiter", ",") \
            .option("header", "true") \
            .option("inferSchema", "true") \
            .load("UScrime2-colsLotsOfNAremoved.csv")
```

```
--2016-09-24 13:25:39--  https://s3.eu-central-1.amazonaws.com/dsr-data/UScrime/UScrime2-colsLotsOfNAremoved.csv
Resolving s3.eu-central-1.amazonaws.com (s3.eu-central-1.amazonaws.com)... 54.231.193.33
Connecting to s3.eu-central-1.amazonaws.com (s3.eu-central-1.amazonaws.com)|54.231.193.33|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 971758 (949K) [application/octet-stream]
Saving to: ‘UScrime2-colsLotsOfNAremoved.csv.1’

100%[=====================================>] 971.758     3,37MB/s   in 0,3s

2016-09-24 13:25:40 (3,37 MB/s) - ‘UScrime2-colsLotsOfNAremoved.csv.1’ saved [971758/971758]
```

```
crimes.count()
```

```
1994
```

Apply PCA and interpret result

```
from pyspark.ml.feature import PCA
pca = PCA(k=10, inputCol="features", outputCol="pca")
model = pca.fit(featuresDF)
pc = model.transform(featuresDF)
pc.toPandas()[:3]
```

	features	pca
0	[0.19, 0.33, 0.02, 0.9, 0.12, 0.17, 0.34, 0.47...	[1.2138889197, 0.564567759337, -0.022284837106...
1	[0.0, 0.16, 0.12, 0.74, 0.45, 0.07, 0.26, 0.59...	[0.627985190195, 1.16689414866, -0.51416430664...
2	[0.0, 0.42, 0.49, 0.56, 0.17, 0.04, 0.39, 0.47...	[0.234349043189, 0.348070144228, 0.54876884160...

- 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!