Algoritma Academy: Unsupervised Machine Learning

Samuel Chan

September 3, 2018

Background

Algoritma

Algoritma is a data science education center based in Jakarta. We organize workshops and training programs to help working professionals and students gain mastery in various data science sub-fields: data visualization, machine learning, data modeling, statistical inference etc. Visit our website for all upcoming workshops.

Libraries and Setup

We'll set-up caching for this notebook given how computationally expensive some of the code we will write can get.

```
knitr::opts_chunk$set(cache=TRUE)
options(scipen = 9999)
```

You will need to use `install.packages()` to install any packages that are not already downloaded onto your machine. You then load the package into your workspace using the `library()` function:

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(FactoMineR)
```

Training Objectives

In this workshop we'll focus our study on a set of widely-used unsupervised learning methods ranging from PCA (Principal Component Analysis), to Clustering, and other pattern discovery approaches where the target variable is not known or defined. Our goal is to develop a solid intuition behind the problem of dimensionality, the mechanism that is at our disposal, and finally solidify our understanding by working on two of the most common real-life business scenarios.

- **Dimensionality**
- The Curse of Dimensionality
- Principal Component Analysis
- Eigenvector and Eigenvalues
- `prcomp` in R
- **Unsupervised Learning Algorithms I**
- Rethinking covariance
- Visualizing PCA
- Using `FactoMineR`
- Practical Applications: eigenfaces
- PCA for Image Processing
- **Unsupervised Learning Algorithms II**
- Clustering Methods
- k-means
- Combining PCA with k-means

Unsupervised Learning

Throughout the Machine Learning Specialization, we've been learning about algorithms that are greatly useful in situations of regression and classification. More generally, we learn to find the parameters for X1, X2 … Xn to explain or predict a "target" response Y.

In the case of unsupervised learning, the situation differs in that there is no such a response Y but rather, we're interested in discovering the structure between X1, X2, to Xn - possibly to identify opportunities for dimensionality reduction or for clustering. Some people have likened unsupervised learning to an exploratory process because it is difficult or impossible to know if the model or any formulation is the "right" one since we don't have a "ground truth" that we use as a measuring stick. Techniques such as cross-validation and AUC do not apply due to the lack of a "ground truth" label.

With that said, unsupervised learning methods can still be very powerful especially in the field of clustering and dimensionality reduction. In this workshop, we'll take an in-depth look at unsupervised algorithms such as PCA and k-means - and see why unsupervised methods such as these are great tools to add to your toolbox.

Figure 1: "Faces at 40x40px"

Dimensionality Reduction

Principle and Motivation

Machine learning is gaining more and more adoption in fields that deal with high-dimensional data such as handwritten digit recognition, internet-of-things (IOT), and face recognition. As a result, the modern data scientists working with these technologies are faced with a "dimensionality" problem that begs for a methodical solution, not just to reduce the dimensionality of the data but to do so while minimizing the loss of information.

As a motivational example, think of the case of a low-resolution image. Do a simple google search for black and white faces at 40x40 pixels[1]. When we treat each image as an input, then our dataset has 1,600 dimensions.

If you zoom in on the faces, we're looking at 1600 individual pixels. When dealing with grayscale images, a strategy can be assigning a value of lightness on the scale of 0 to 1 to each of the 1,600 columns, with 0 being "full white" and 1 being "full black", and depending on the saturity or lightness of each pixel - assign them a value in between.

There are numerous good papers and resources that deal with the topic of PCA use in image compression, such as the one by researchers as Institute of Chemical Technology, Prague[2], the one by Czech Institute of Informatics, Robotics and Cybernetics[3], and this one here[4]. By the end of this PCA section, we'll also apply PCA on human faces to see how image compression works in practice.

A 40x40 image is probably not interesting. If you're building an image classifier using photos from an iPhone 7 plus, that's a resolution of 1,920 x 1,080 pixels (more than 2 million dimensions). And that's for a single observation. Recall from your Practical Statistics course that a way we can measure "information" is through variance, so a dimensionality reduction method is essentially concerned with representing as much "variance" as possible in as few dimensions as possible.

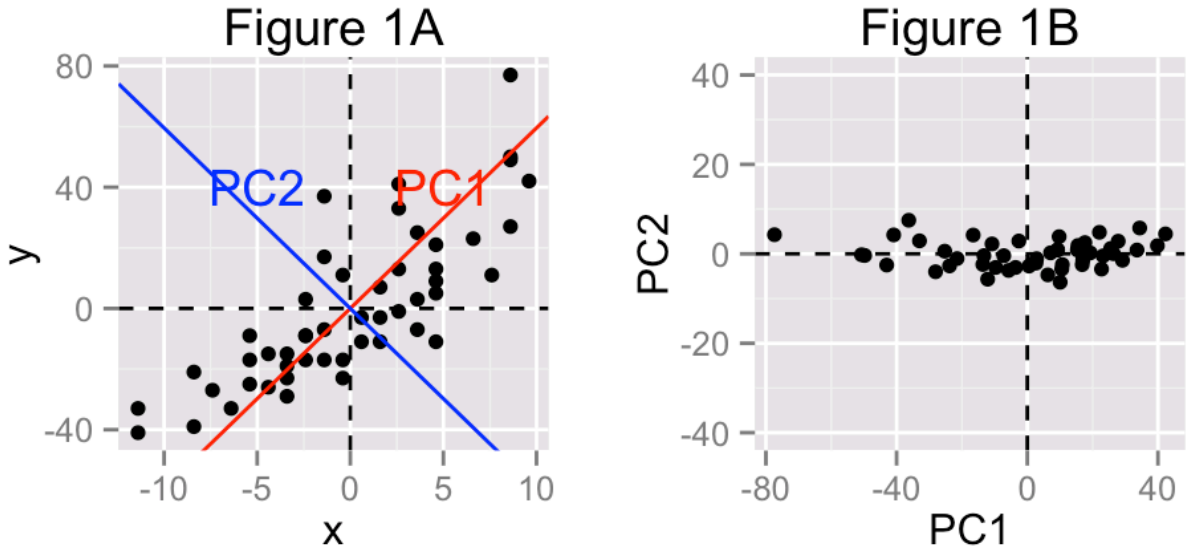
The outcome of this transformation is that our original data (a matrix X) is represented by a linearly transformed matrix, Z, where Z is typically a matrix with a lot fewer dimensions (commonly <10) than X. The first column of Z explains most of the variance within X, and the second explains a smaller amount of variance than the first, and so on until the last column.

The objective of PCA is to find Q, so that such a linear transformation is possible.

[1] Face images at 40x40 resolution
[2] Principal Component Analysis in Image Processing
[3] Principal Component Analysis (PCA) Application to images
[4] Simple Image Classification using Principal Component Analysis (PCA)

Figure 2: “Faces at 40x40px”

If you remember the lessons from last week (Classification 2), I demonstrated the use of `nearZeroVar` and **shown that while eliminate ~50% of the original predictors we still retain enough information** to build a multi-class classifier that has a 99.98% out-of-box accuracy. The features that were eliminated are redundant in that they add little value or that they may just represent "noise". PCA shares the same objective, but does it differently: it looks for correlation within our data and use that redundancy to create a new matrix Z with just enough dimensions to explain most of the variance in the original data. The new variables of matrix Z are called **principal components**.



If you look at Figure 1A, our original data sits on a plane with `x`and `y` coordinates. Two dimensions (x and y, respectively) are required in Figure 1A to describe the variance in our data fully.

However, supposed we identify two other axes to describe the same data, and one of them is directly orthogonal to the other one: we can now measure the variance in our data using just these axes (we call them **principal components**). We identify the PC1 axis as the first principal component because using only one principal component, this would be the one that explain the most amount of variation. The PC2 axis is then selected, again with the objective of explaining the most amount of variation.

If we hold PC2 as constant (say, 0) then we reduce the dimensions from two to only one, which is by projecting each data point onto PC1. We do lose some variation as our observations are not exactly 0 on the PC2 axis, but since they are very close to being 0, the variation we lose from reducing one dimension is a tradeoff we want to accept.

(Recall our lessons from Regression Model, when I introduce `VIF` to show how if variable "Police Expenditure this year" can be sufficiently explained by variable "Police Expenditure last year" then we don't need both variables)

Other applications of PCA:
- Pattern discovery on high dimensional data
- Identify variables that are highly correlated with others
- Visualizing high dimensional data

Dive Deeper: Which of the two following data set are going to be helped most by Principal Component Analysis (PCA)?

```
set.seed(100)
par(mfrow=c(1,2))
x <- runif(100)
```