UNSUPERVISED LEARNING

Chapter 10 (part 01)

Outline

- Intro to Unsupervised Learning
- Principal Components Analysis (PCA)
  - Goals
  - Implementation - Conceptual
  - Implementation - Math
  - Interpretations
  - Uses
- Clustering (in slides for Chapter 10, part 2)

Intro to unsupervised learning

- If you don't have a response variable, you can't make a function *f* of inputs to outputs
- Is there anything you *can* do with a bunch observations of predictors?

Unsupervised Learning

- Even without a response variable, you can still look at relationships within the observations
- There are a few things you can do without a response variable - including:
  - Interpret (visualize) relationships among observations through linear algebraic manipulations (rotations of the feature axes): PCA
  - Compress data through dimensionality reduction: PCA
  - Lump similar observations together: Clustering
  - Compress data by substituting cluster centers and distributions for observations: Clustering / Mixture models

PCA - Goals

- Since we have no response variable, we assume that differentiation (variance) among observations captures something meaningful in the domain
- We have a *p*-dimensional space of features, and assume some are more meaningful than others, but all may contribute something to the interpretation
- We wish to explain or summarize these differences with as few parameters as possible. **WHY?**

PCA Intuition – Conceptual (1 of 4)

- Scale all variables (Z-scaling)
- Define a *first* axis of highest variance in feature-space of the observations (this is **component 1**)
- Then we iterate until *p*-axes are selected:
  - Pick one of the remaining axes which are orthogonal to all previously selected axes …
    - …such that this selected axis is the one which has the maximum variance among its datapoints, given the previously-selected axes are fixed
  - When an axis is selected, it is appended to an ordered list of components

PCA Intuition – Conceptual (2 of 4)

- These orthogonal axes can be described in terms of *rotations* to the *p* axes in the original feature dimensions
- The rotations align the new PCA axes along lines of variance, from high to low
- The “first” component is the axis expressing the most variance, and in order, successive component axes capture the ever-decreasing variance in the observations
- Each component axis can be expressed as a set of numerical *loadings* to the original *p* feature dimensions
- For example, the first principal component is:

$$Z_1 = \phi_{11}X_1 + \phi_{21}X_2 + \ldots + \phi_{p1}X_p$$

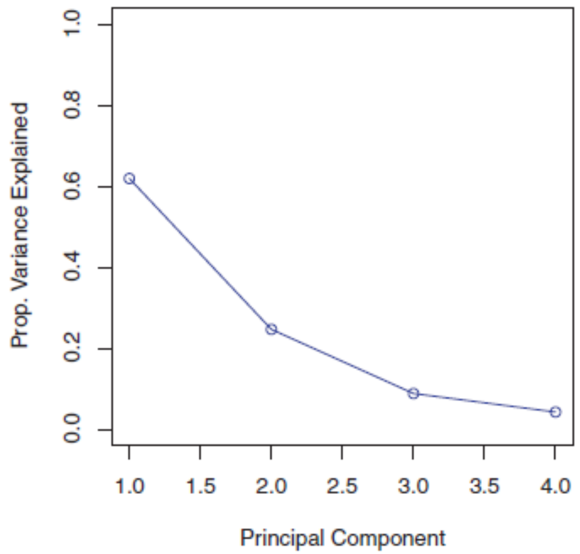
PCA Intuition – Conceptual (3 of 4)

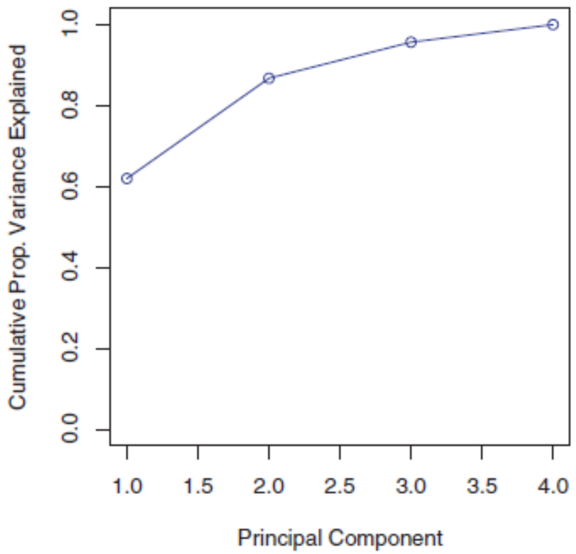
$$Z_1 = \phi_{11}X_1 + \phi_{21}X_2 + \ldots + \phi_{p1}X_p$$

- Given the first principal component loading, we can project value for an observation on that axis using the formula above
- We can repeat the process to obtain the observation's projection onto the other component axes
- The datapoint (Z_1,Z_2,Z_3,…Z_p) is the projection of the observation into the principal component space.

PCA Intuition – Conceptual (4 of 4)

- Each successive principal component explains less of the variance in the data
- A scree plot can be used to visualize the variance explained by the k^{th} component (or cumulative explanation of variance by the k components so far)





PCA Implementation – Code

- A linear algebra technique can provide all of the orthogonal component axes which explain the variance in the features of the observations
- Produce the covariance matrix for the dataset
  `cov_mat=np.cov(X.T)`
- Singular Value Decomposition on the covariance matrix produces a *p* x *p* matrix (U) which contains the ordered loadings of the dataset:
  `u,s,v = np.linalg.svd(cov_mat)`
- Each column in U corresponds to a loading vector in PCA
  - The leftmost column represents the most important component and each successive column to the right represents columns of decreasing importance

PCA Tuning

- Select a desired percentage *v* of the variance to explain
- Choose k (the number of components) such that for the approximation of the datapoints:

$$\frac{\frac{1}{m}\sum_{i=1}^{m}\left\|x^{(i)}-x_{appx}^{(i)}\right\|^2}{\frac{1}{m}\sum_{i=1}^{m}\left\|x^{(i)}\right\|^2}\leq v$$

- A matrix multiplication of the (first k columns of the) U matrix and the dataset yields the projection of the observations into component space: `np.dot(X,u[:,0:componentCount])`
- Alternately, use `sklearn.decomposition.PCA`

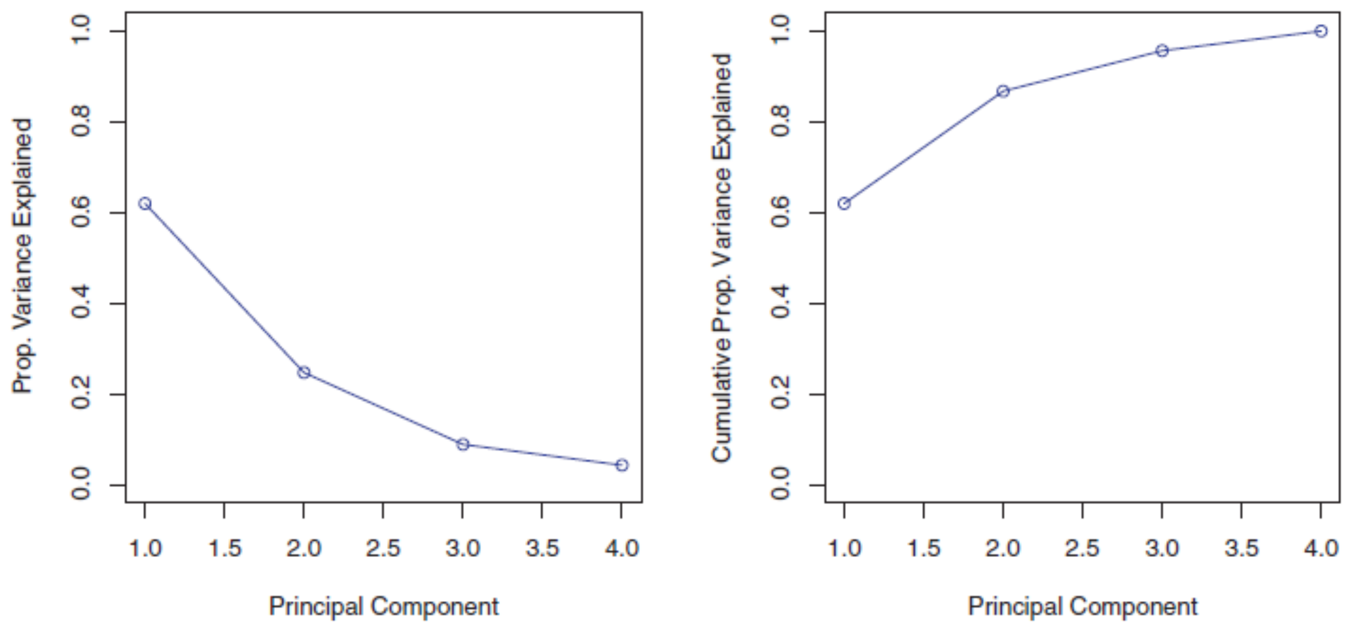
PCA Evaluation – Variance Explanation

- Variance from the m^{th} principal component

$$\frac{1}{n}\sum_{i=1}^{n} z_{im}^2 = \frac{1}{n}\sum_{i=1}^{n}\left(\sum_{j=1}^{p}\phi_{jm}x_{ij}\right)^2$$

Percent of Variance Explained (PVE)

$$\frac{\sum_{i=1}^{n}\left(\sum_{j=1}^{p}\phi_{jm}x_{ij}\right)^2}{\sum_{j=1}^{p}\sum_{i=1}^{n}x_{ij}^2}$$

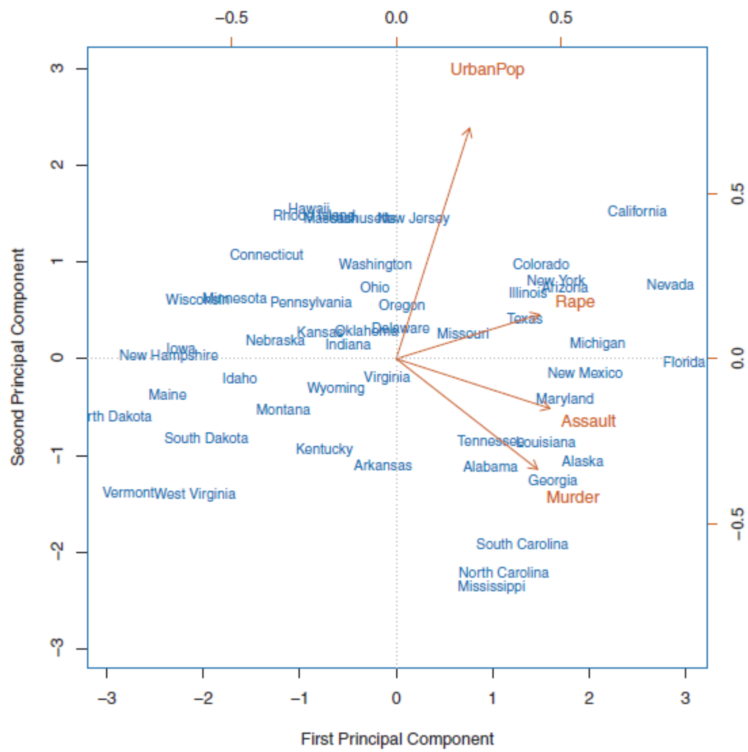


PCA Interpretation

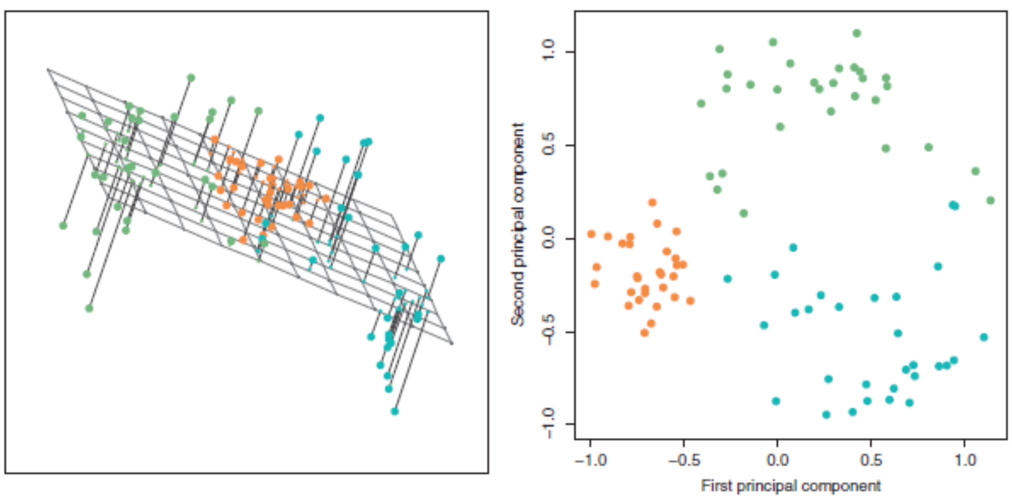
- The projection of the observations into the first k principal components (1..k) represent a lossy approximation of the dataset
- In this reduced space, fewer parameters are used to approximate the data

PCA – Uses

- Visualize important data relationships (in 2D)



- Compression (reduce number of features from p to k)



Mitigate Collinear Features before model fitting

More on understanding PCA

- https://towardsdatascience.com/pca-and-svd-explained-with-numpy-5d13b0d2a4d8
- https://medium.com/@jonathan_hui/machine-learning-singular-value-decomposition-svd-principal-component-analysis-pca-1d45e885e491
- https://www.analyticsvidhya.com/blog/2016/03/practical-guide-principal-component-analysis-python/