Principal Components Analysis (PCA)

XDASI Fall 2021

12/6/2021

Contents

Finding Patterns in Data

One of the most common tasks in genomics and systems biology is to group or ***classify*** data by similarity in order to find biologically meaningful patterns.

For example, single-cell sequencing aims to find cells that are of the same type, and to distinguish between different types, based on gene expression data. We may also want to group genes together to find those that change the most between different cell states or experimental conditions, or which are most representative of a specific cell state.

This is the realm of ***machine learning***. There are two main types of machine learning: **supervised** and **unsupervised**. In this class we will go over two types of unsupervised learning: ***clustering*** and ***dimensional reduction with PCA***. These methods can help us answer questions such as:

- How many distinct “classes” of cells are there?
- Which conditions have the most similar “state” in terms of their expressed genes?
- How can I identify genes that give the most information about differences between conditions / cells?
- How can I simplify and describe the data with a smaller number of variables / descriptors?
- How can I visualize the relationships between different conditions?

Dimensional Reduction

Often, the amount of “information” present in a dataset – in terms of the total variability – is not uniformly distributed across measurements. **Why?**

If two dimensions (experimental conditions) are highly **correlated**, then there is some **dependency** between them, and the amount of information gained by including the second dimension in your analysis is small. We will see this when we look at correlations between predictors in the homework assignment on logistic regression.

In such cases, instead of simplifying our data by just selecting a few features to work with (“feature selection”), we can ***project*** the data onto a smaller number of dimensions that capture the “essence” of the variation and different groupings in the data using a much smaller number of explanatory variables, without losing too much of the original information.

Principal Components Analysis

Background Reading

- Introduction to Statistical Learning
  - Introductory: Chapter 6 - PCA Overview
- Advanced: ISLR Chapter 10 - Unsupervised Learning *(through Section 10.2)*

Useful Videos

- Statquest:
  - PCA main ideas (5 min)
  - PCA step-by-step (20 min)
  - PCA practical tips (8 min)
  - PCA in R (9 min)

Overview

Our goal is to identify large-scale patterns across the data, while reducing complexity so that we focus on the most important information. **Principal components** enable us to summarize data using a smaller number of variables that collectively capture most of the **variability** in the original dataset.

A simple 2D example (**Figure 1**). Here the axes are labeled in distances, but they could just as well represent individual ***cells***, where the data points are the epression levels of individual ***genes*** in each cell.

If we take all of the data points and ***project*** them onto one of the axes, we can see that most of the variation is preserved in the projection (called z_1 in the figure). The main difference is that the total variation along the best-fit line in 2D is somewhat squished upon reduction to 1D.

The key concept behind PCA is that we want to capture and preserve the greatest amount of variation in the data using some combination of the original measurements.

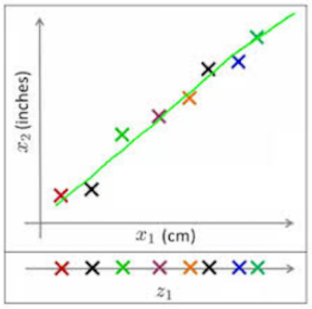

Figure 1: ***Projecting data in two dimensions onto a single dimension***

How does PCA work?

Consider the following two-dimensional dataset (**Figure 2**):

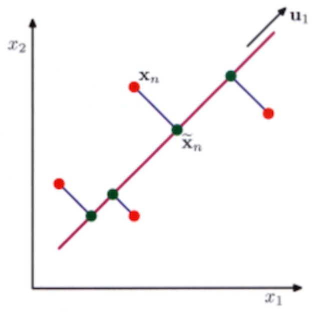

Figure 2: ***Projection onto a principal subspace***

Most of the variation is again along the best-fit line (labeled u_1 in this figure). This is the **principal component** in the dataset, the axis along which most of the variation in the dataset lies.

The best-fit line is a **linear combination** of the original axes. Here the slope of the line is about 1, so u could be described as a mixture of x_1 and x_2, combined in equal proportions.

PCA is a way to **change coordinate systems** so that the **maximum amount of variation is distributed along orthogonal axes**. How do we do this?

- Instead of projecting directly onto one of the original axes, we make a ***linear projection*** of the data onto a lower-dimensional subspace, which we call the **principal subspace**.
- In the new subspace, most of the variation in the data is distributed along the **first principal component**, or **PC1**. In other words, it is the dimension along which the data are most "spread out".
- For three-dimensional data, we project our data onto two new axes that are **orthogonal**, just like our original axes x_1 and x_2. The 1st PC will still contain the greatest amount of variation in the data, the second PC will contain the second greatest amount of variation in the data, and so on.

The projection is a mathematical transformation that is simply a **linear combination** of the original variables. There are two equivalent ways to think about PCA that are both illustrated in **Figure 2**:

- **Minimize the average projection cost**, i.e. the **mean squared distance** between data points and their projections (the distance from x_n to to $\tilde{x}$).
- **Maximize the variance** of the projected data (along the $\mathbf{u_1}$ axis).

The two formulations give rise to the same algorithm, but in practice the system is solved by maximizing the variance, which turns out to be easier computationally.

A simple case: PCA in two dimensions

Now let's look at a real-life example, from Chapter 6.3.1 of *Introduction to Statistical Learning* (**Figure 3**).

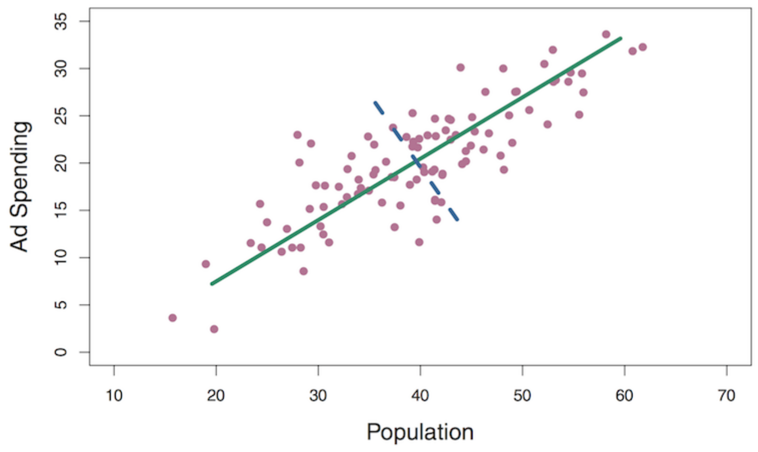

FIGURE 6.14. *The population size* (pop) *and ad spending* (ad) *for* 100 *different cities are shown as purple circles. The green solid line indicates the first principal component, and the blue dashed line indicates the second principal component.*

Figure 3: Population vs. Ad Spending in 100 US cities

We see from the figure that most of the variation in the data falls along the green line. This is the first principal component (PC1). If we were to remove this variation, then we could describe the rest of the variation in terms of the orthogonal dimension shown by the dashed blue line (PC2).

Now we have effectively changed our coordinate system from "Population" (x) vs. "Ad Spending" (y) to a **new coordinate system**, PC1 and PC2 (**Figure 4**). Notice that we have transformed the data by **mean centering**, so that we represent each original dimension as a deviation from its mean.

Relationship between original and PC coordinate systems

Now we want a way to quantify the **loadings** of the original variables on to our new dimensions. For the above example, we can now write, for PC1:

$$Z_1 = 0.839 * (pop - \overline{pop}) + 0.544 * (ad - \overline{ad})$$

For any system, we can define M principal components, $Z_1 ... Z_m$.

Now something interesting has happened. We have transformed our original set of p predictors (population, ad spending) of n variables (cities) to a new set of predictors in M dimensions: the **principal components**.

We can also see that much more variation is captured in PC1 than in PC2 by plotting each of the original variables against each PC:

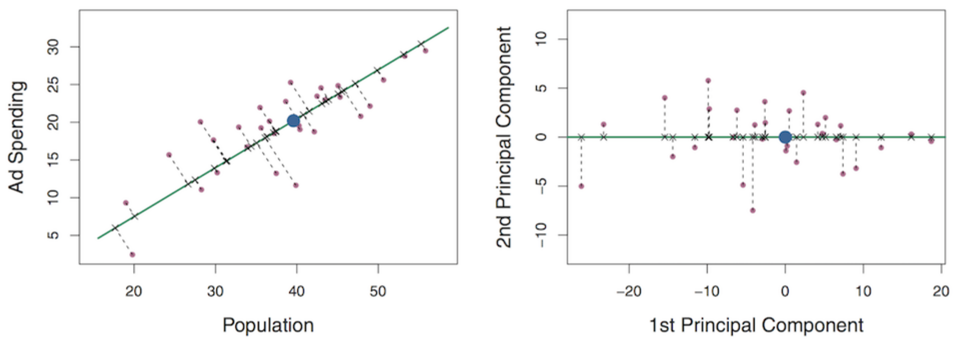

FIGURE 6.15. *A subset of the advertising data. The mean* `pop` *and* `ad` *budgets are indicated with a blue circle.* Left: *The first principal component direction is shown in green. It is the dimension along which the data vary the most, and it also defines the line that is closest to all n of the observations. The distances from each observation to the principal component are represented using the black dashed line segments. The blue dot represents* $(\overline{\texttt{pop}}, \overline{\texttt{ad}})$. Right: *The left-hand panel has been rotated so that the first principal component direction coincides with the x-axis.*

Figure 4: Data transformation to PC1 and PC2

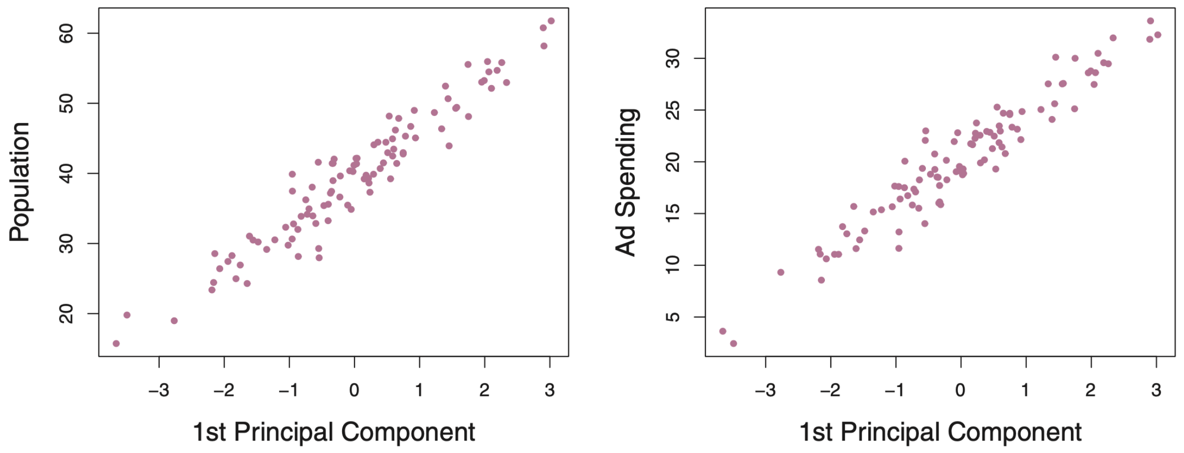

FIGURE 6.16. *Plots of the first principal component scores* z_{i1} *versus* `pop` *and* `ad`*. The relationships are strong.*

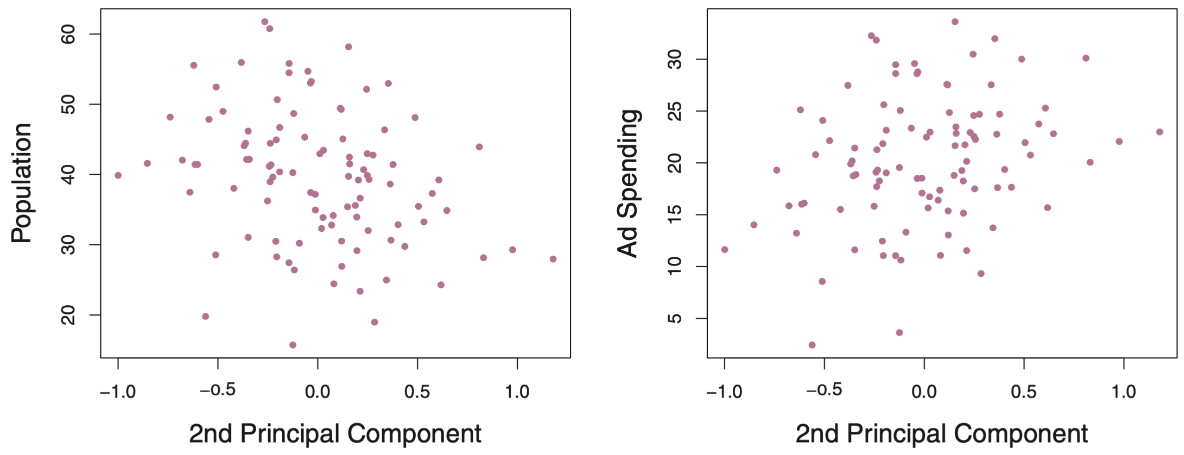

FIGURE 6.17. *Plots of the second principal component scores z_{i2} versus* `pop` *and* `ad`*. The relationships are weak.*

In this example, that doesn't seem like a big deal, but consider gene expression studies, where we measure thousands of genes (our n variables) across many samples (call this p). We can describe this as an n x p matrix.

By performing PCA, we can reduce the dimensionality of our dataset from p to something much smaller that explains most of the variation in the dataset. This is particularly convenient for large datasets, such as *single-cell studies*, since our principal components will be some **linear combination** of our original measurements.

Since most of the variation is in PC2, the second-most is in PC2, etc., most of the variation in the entire dataset will be present in a small number of PCs, which we can then use to simplify the description of our data.

A 3D example

What happens when we have more than just two features to compare? Let's consider a small dataset simulated in three dimensions (**Figure 5**). Now we see that, in the same way that PC1 captures the most variation in the dataset, together the first M principal components provide the best M-dimensional approximation of the original dataset. We can extend this approach to any number of dimensions.

As the number of features p increases, it rapidly becomes unweildy to visualize how the data are distributed by plotting every pairwise combination of features (since there are $p * (p-1)/2$ pairwise combinations).

Instead, PC plots provide a nice way to visualize the most significant variation across the dataset. Looking at 2D plots of the first few PCs usually provides a pretty good summary of the overall variation in the original data.

Mathematical formulation for PCA

Let's now find a mathematical way to describe our new coordinate system that generalizes to multiple dimensions. Let's call our set of p predictors $X_1, X_2 ... X_p$, and M represent $M < p$ linear combinations of them. Note that for each feature (condition) X we have n observations (genes), so that each X is a vector of length n.

For the best fit PC1 line, we can now write:

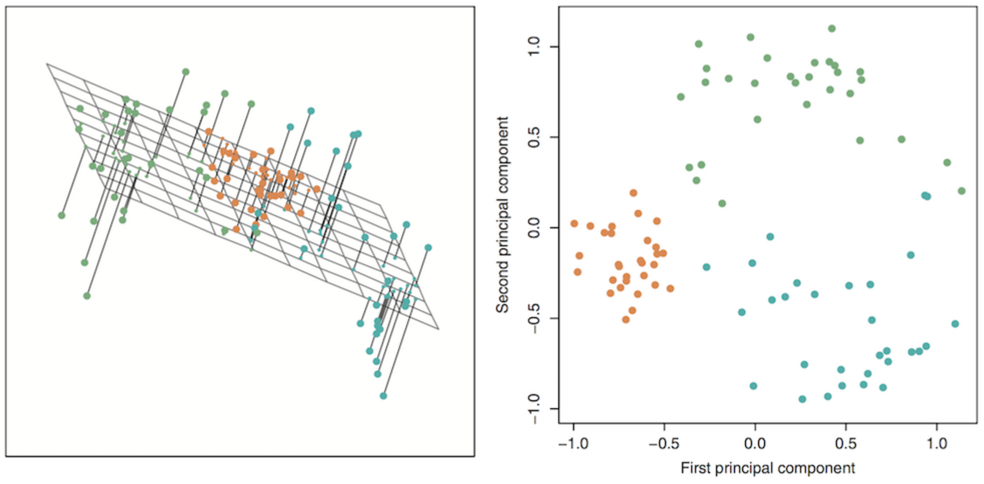

FIGURE 10.2. *Ninety observations simulated in three dimensions.* Left: *the first two principal component directions span the plane that best fits the data. It minimizes the sum of squared distances from each point to the plane.* Right: *the first two principal component score vectors give the coordinates of the projection of the 90 observations onto the plane. The variance in the plane is maximized.*

Figure 5: PC1 and PC2 for a 3D dataset

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

Z_1 is the **first principal component** of the set of features X. It is a **normalized linear combination** of these that has the **largest variance** across the p predictors.

Loading vectors

We call the coefficients ϕ the **loadings** of the predictors onto the principal components. We write the loading vector for the first principal component as:

$$\phi_1 = (\phi_{11}, \phi_{12}, ..., \phi_{p1})$$

In the above example, the loading of "Population" onto the first principal component was 0.839, and the loading of "Ad spending" onto PC1 was 0.544.

Normalization: Centering and Scaling

Notice that we also said the the linear equations are **normalized**. In order to make our measurements across the different variables (e.g. samples) more comparable, we both **center** the data points in each sample by subtracting each measurement from the sample mean, and then **scale** the variation for each PC so that the standard deviation equals one: $s = 1$. (Notice that this is reflected in the units shown for PC1 and PC2 in Figure 5.)

Normalization allows us to quantify the proportion of the total variance that is distributed along each PC. When $M = p$, i.e. the number of PCs equals the number of original dimensions, we have explained 100% of the variation in the original dataset.

Therefore, the *total variance* for $Z1$ across the entire dataset will sum to 1, that is:

$$\sum_{j=1}^{p} \phi_{j1}^2 = 1$$

For the advertising example above, we can see that $\theta_{11}^2 + \theta_{21}^2 = (0.839)^2 + (0.544)^2 = 1$.

Scaling is much more important when variables are measured in different units, such as the population of cities vs. the amount of ad money spent in each state.

PC scores

Above, we have expressed the best-fit projection axes Z in terms of the original axes X.

To complete our formal description, we need a way to describe the relationship between each PC and each of the ***original observations*** x_{ij}, where $i = 1...n$ and $j = 1...p$. For an n x p matrix of expression data, for example, we index each gene with i and each sample with j.

Each of the M principal components Z comprises n linear combinations of p terms. We call these *scores* and denote them as $z_{11}, ..., z_{n1}$. The scores for PC1 take the form:

$$z_{i1} = \phi_{11}x_{i1} + \phi_{21}x_{i2} + ... + \phi_{p1}x_{ip}$$

Just as the loadings relate each PC to the original set of features $X_1...X_p$, so the scores for each PC relate to the individual elements x_{ij}.

How to choose the optimal number of PCs?

In practice, since we are interested in simplifying our description of the data, we want M to be smaller than p. There is no single correct answer to choosing M. One method that is commonly used is to use a ***scree*** plot (**Figure 6**).

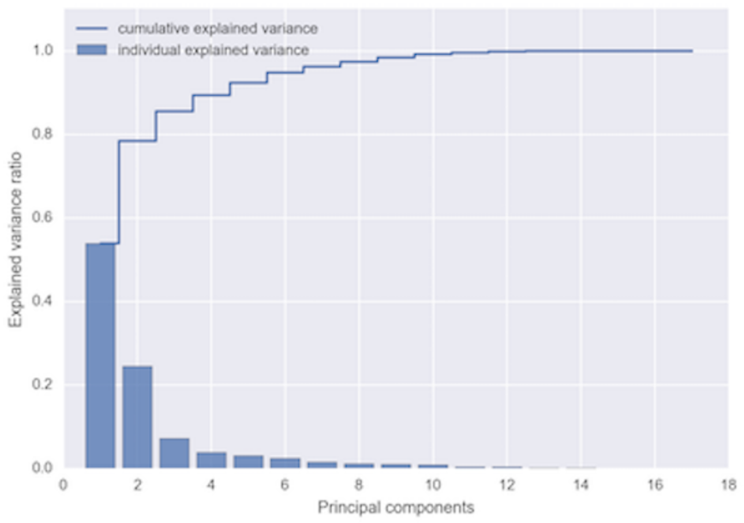

Figure 6: **Scree plot for 4 principal components**

A common approach is to look for an ***elbow*** in the plot, and choose the number of PCs that is one less (because at the elbow, adding PCs adds less and less to the total explained variation). In this example, 2 principal components seem sufficient to explain most of the data. Often, you will have more than 4 PC.

Another way is to decide on the total proportion of the variance you want to be able to explain. For example, you could choose a cutoff of 80-95%, depending on your application.

A brief note on tSNE and UMAP

PCA, tSNE, and UMAP all seek to simplify the way data are represented while preserving distinctive patterns in the data, i.e the most informative features and groupings and the relationships among them.

You can broadly think of these as different ways to look at relationships, based on "far"-ness vs. closeness.

- **PCA**:
  - Identifies a **linear combination** of features (measurements in multiple dimensions) that explain the **total variation** in the original dataset.
  - Used for a wide range of applications, including image analysis, data compression, feature extraction, and data visualization.
- **tSNE and UMAP**:
  - Used to visualize high-dimensional data in 2 or 3 dimensions.
  - Primarily focused on preserving **local structure**.
  - UMAP is better at capturing total variation than tSNE.
  - Global structure is better preserved using UMAP than tSNE.

Especially for single-celled profiling data, **PCA and clustering are often used as a precursor to 2D visualization with UMAP** in order to limit the presentation of data to the portion of the data that actually drive variation between conditions in the dataset (e.g. most commonly, different cell types or sub-types).

We will not go through the intricacies of tSNE and UMAP here, since the algorithms are rather complex. You will see a lot of tSNE maps in older single-cell papers. However, UMAP has largely replaced tSNE in recent years due to several advantages:

The ***Towards Data Science*** website contains several useful blog posts that discuss how tSNE and UMAP work and the differences between them. These can be somewhat involved, so to get started, just read through the general parts that will be easier to digest. **Toward Data Science blog posts on tSNE and UMAP** by *Nikolay Oskolkov*

I have also written **my own guide to tSNE** that draws on some of these sources, which you may peruse if you like. **PDF - HTML**