A Tim Book story

Principal Components Analysis (PCA)

GENERAL ASSEMBLY

Agenda

- What situation motivates PCA?
- (High level) What does PCA do for us?
- (Low level) How does PCA work?
- Python!

 © 2022 General Assembly

What situation motivates PCA?

Has this ever happened to you?

- Too many variables
- Some are probably meaningless
- Some are probably redundant

What situation motivates PCA?

Has this ever happened to you?

- Too many variables
- Some are probably meaningless
- Some are probably redundant (ie, multicollinear)

What if I could just have only a few variables that “squeeze” all the information out of the several variables I have?

 © 2022 General Assembly

That's PCA!

Principal components analysis is the act of taking your *x*-variables and condensing them into their **principal components** so that you can use fewer variables that have the same approximately information all all of the variables together.

That's PCA!

Principal components analysis is the act of taking your *x*-variables and condensing them into their **principal components** so that you can use fewer variables that have the same approximately information all all of the variables together.

PCA is an **unsupervised learning** technique for carrying out

dimensionality reduction

That's PCA!

Principal components analysis is the act of taking your *x*-variables and condensing them into their **principal components** so that you can use fewer variables that have the same approximately information all all of the variables together.

PCA is an **unsupervised learning** technique for carrying out

dimensionality reduction

The number of dimensions (variables) we're using

Using fewer of them!

 © 2022 General Assembly

Example

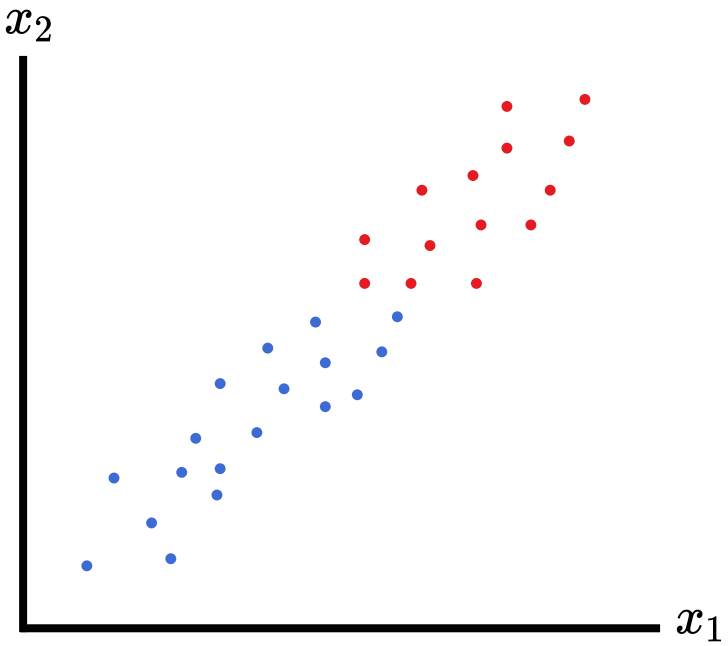
- Suppose you have many *x*s:
  - $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{200}$
- From this, you can create just as many PCs:
  - $\boldsymbol{z}_1, \ldots, \boldsymbol{z}_{200}$
- But maybe, just a few of these PCs have just as much “information” as the full data:
  - $\boldsymbol{z}_1, \ldots, \boldsymbol{z}_{10}$
- We can use $(\boldsymbol{z}_1, \ldots, \boldsymbol{z}_{10})$ in a ML model instead of $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{200})$!
- What’s best, is that the PCs are also guaranteed to be **independent!**

 © 2022 General Assembly

(High level using pictures)

How does PCA work?

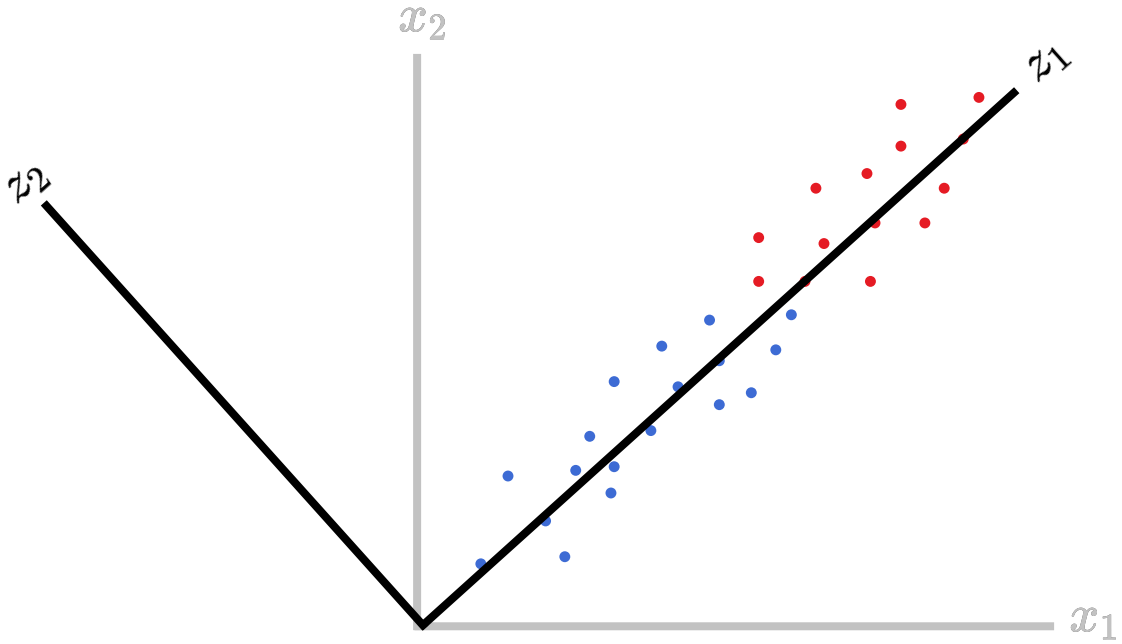
Visual Example



Imagine this classification problem. We could use **two** *x*-variables to tell blue apart from red.

But you might notice that actually we don't need both. One is redundant! There's **multicollinearity!**

Visual Example

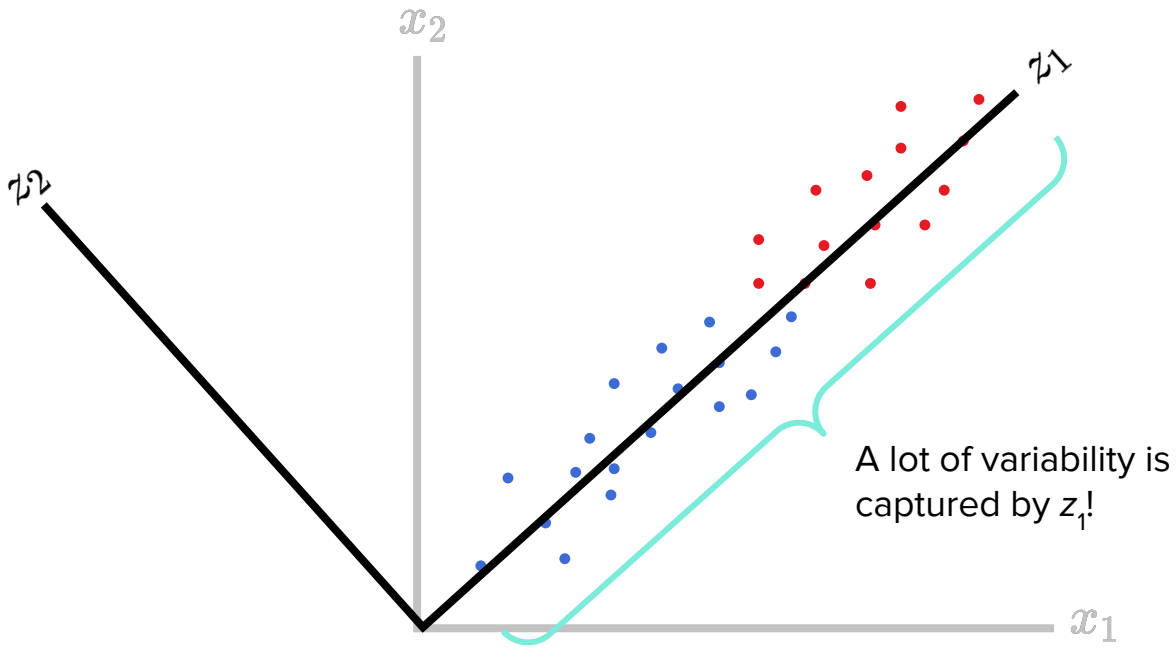


These are the two **principal components***, z_1 and z_2.

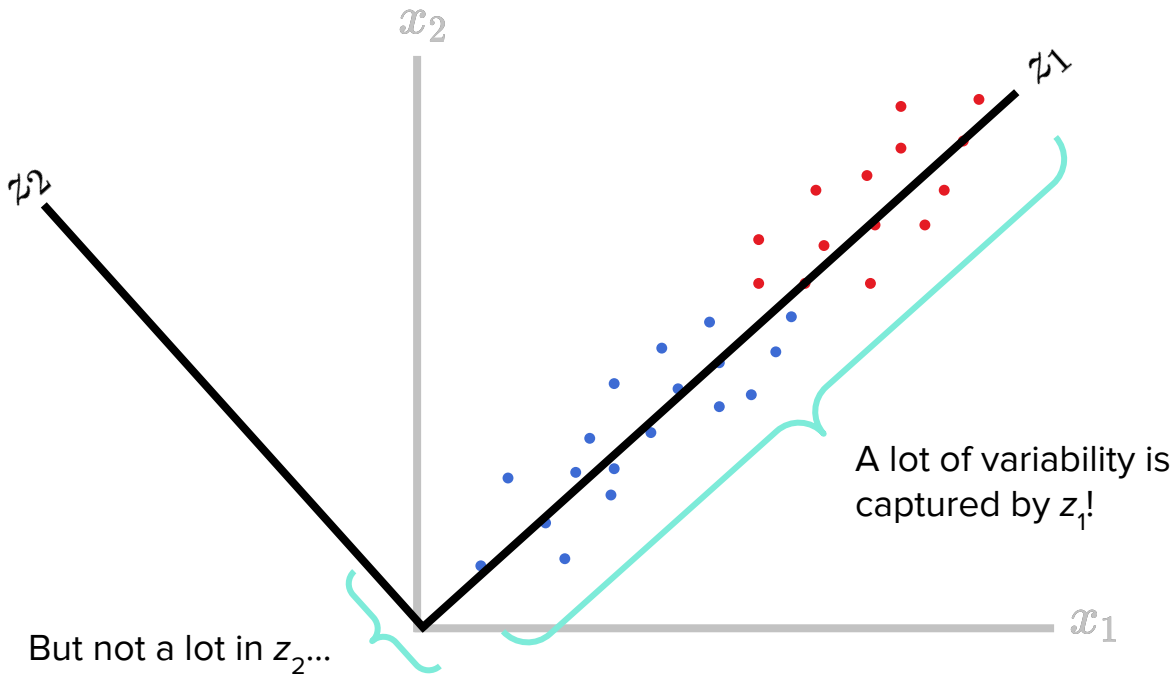
*principal components are sometimes called **rotations** because of this image!

 © 2022 General Assembly

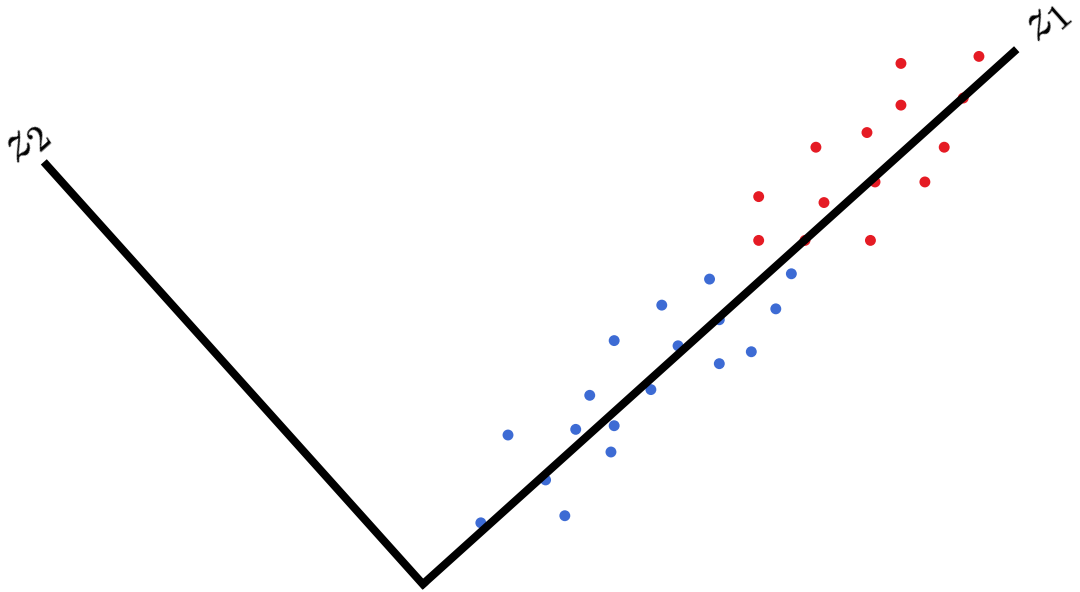
Visual Example



Visual Example



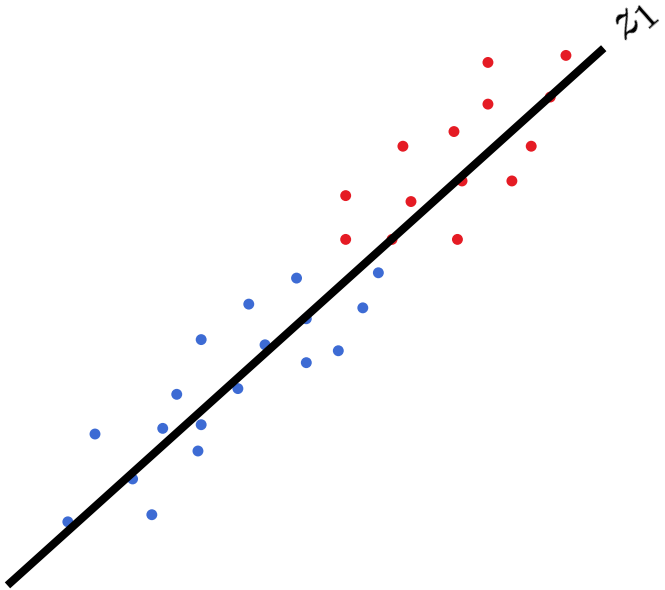
Visual Example



 © 2022 General Assembly

Visual Example

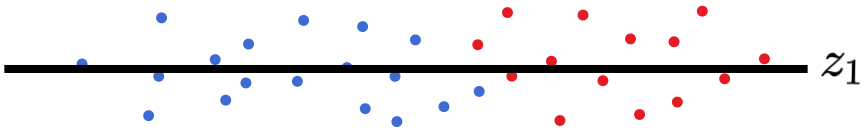
What if we just totally discard z_2 and pretend we only have **one variable**?



 © 2022 General Assembly

Visual Example

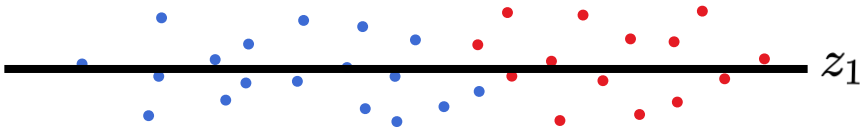
We **reduced the dimensionality** of our data! We used one variable that has the same predictive ability as two!



 © 2022 General Assembly

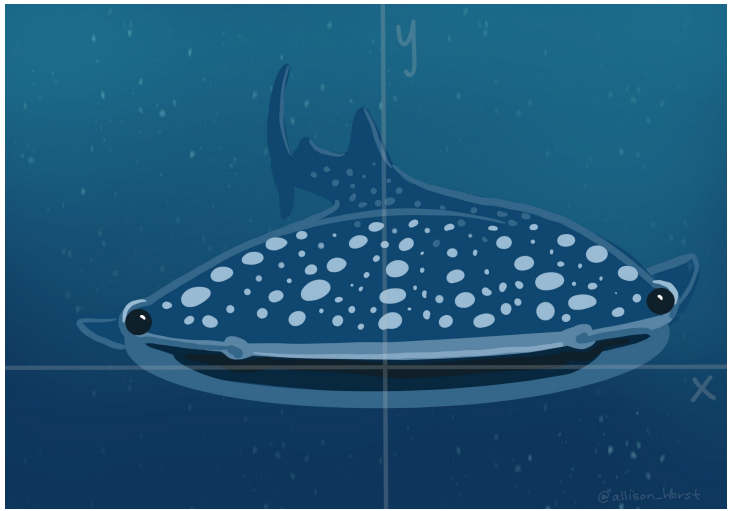
Visual Example

We **reduced the dimensionality** of our data! We used one variable that has the same predictive ability as two!



I used a blue-vs-red classifier to make a point here, but remember, PCA is **unsupervised**! The variability of the principal components has nothing to do with **y**!

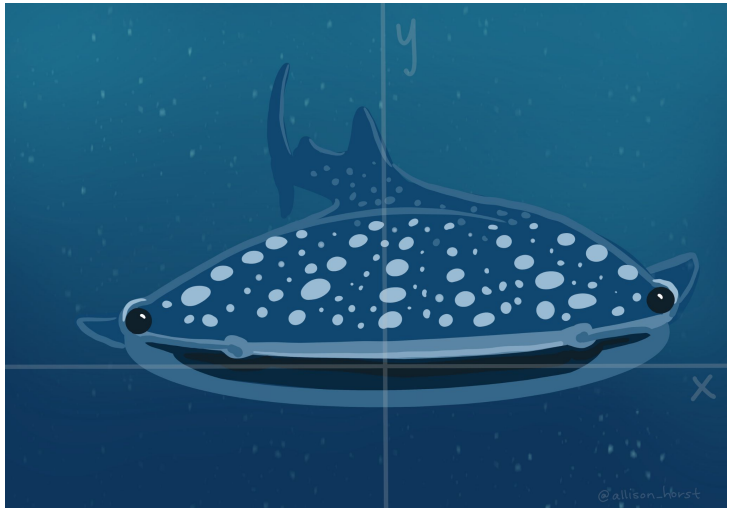
Fun Example



Source: Artist and data scientist @allison_horst's Twitter page

 © 2022 General Assembly

Fun Example





Source: Artist and data scientist @allison_horst's Twitter page

(Low level using matrix algebra)

How does PCA work?



Setup

Let **X** be our *n x p* **scaled** data matrix.

$$\mathbf{X}_{n \times p}$$

 © 2022 General Assembly

Singular Value Decomposition (SVD)

All matrices are guaranteed to have a **singular value decomposition (SVD)** as follows:

$$\mathbf{X} = \mathbf{U}\mathbf{D}\mathbf{V}^T$$

(**Matrix decompositions** are just ways of rewriting certain kinds of matrices. Breaking a matrix into a decomposition is a common trick for solving certain kinds of math problems.)

The SVD of X

$$\mathbf{X} = \mathbf{U}\mathbf{D}\mathbf{V}^T$$

Where:

- **U** is an *n x p* orthogonal matrix (ie, $\mathbf{U}^T\mathbf{U} = \mathbf{I}$)
- **D** is a *p x p* diagonal matrix of **singular values**
- **V** is a *p x p* orthogonal matrix (ie, $\mathbf{V}^T\mathbf{V} = \mathbf{I}$)

 © 2022 General Assembly

U, D, and V

$$\mathbf{U} = \begin{bmatrix} \mathbf{u}_1 & \cdots & \mathbf{u}_p \end{bmatrix}$$

The columns of **U** make up the **left-singular vectors.**

$$\mathbf{D} = \begin{bmatrix} \sigma_1 & & & \\ & \sigma_2 & & \\ & & \ddots & \\ & & & \sigma_p \end{bmatrix}$$

The diagonal values of **D** make up the **singular values**. They're always positive and represent the **relative amount of information** contained in the singular vectors (descending).

$$\mathbf{V}^T = \begin{bmatrix} \mathbf{v}_1^T \\ \vdots \\ \mathbf{v}_p^T \end{bmatrix}$$

The rows of $\mathbf{V}^T$ make up the **right-singular vectors**

 © 2022 General Assembly

The Singular Value Transformation

$$\begin{aligned}\mathbf{Z} &= \mathbf{UD} \\ &= \mathbf{UDV}^T\mathbf{V} \\ &= \mathbf{XV} \\ &\quad n \times p \quad p \times p\end{aligned}$$

The Singular Value Transformation

$$\mathbf{Z} = \mathbf{UD}$$

$$= \mathbf{UDV}^T\mathbf{V}$$

$$= \underset{n \times p}{\mathbf{X}}\underset{p \times k}{\mathbf{V}_k}$$

If we keep *only* the first *k* columns, the result is our **dimensionality-reduced** transformation!

 © 2022 General Assembly

Don’t believe me? Ask beloved dog celebrity, Doug the Pug!



 © 2022 General Assembly

Let’s see for ourselves!



 © 2022 General Assembly