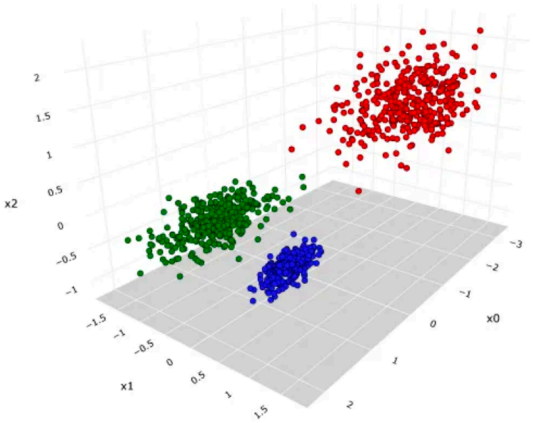
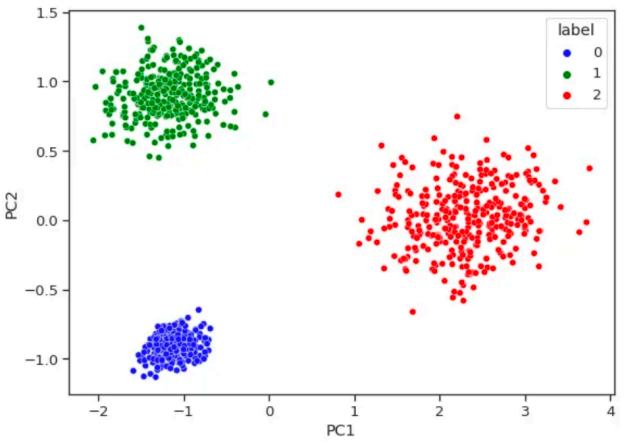
Principle Component Analysis

Reducing the number of features

Principal Components Analysis (PCA), an unsupervised learning algorithm used for data visualization by reducing the number of features in a dataset. (Dimensionality Reduction)

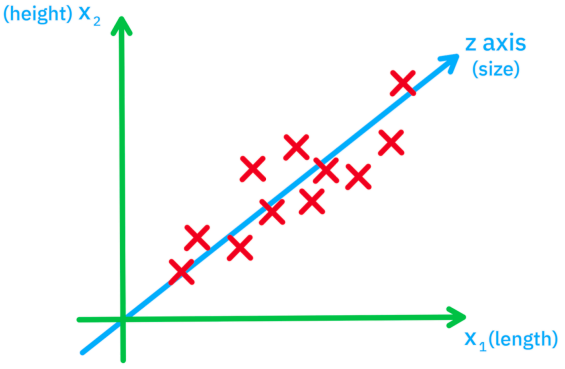
- PCA helps visualize high-dimensional data by reducing it to two or three dimensions, making it easier to plot and analyze.
- It is particularly useful when dealing with datasets that have many features, such as 50 or even thousands.

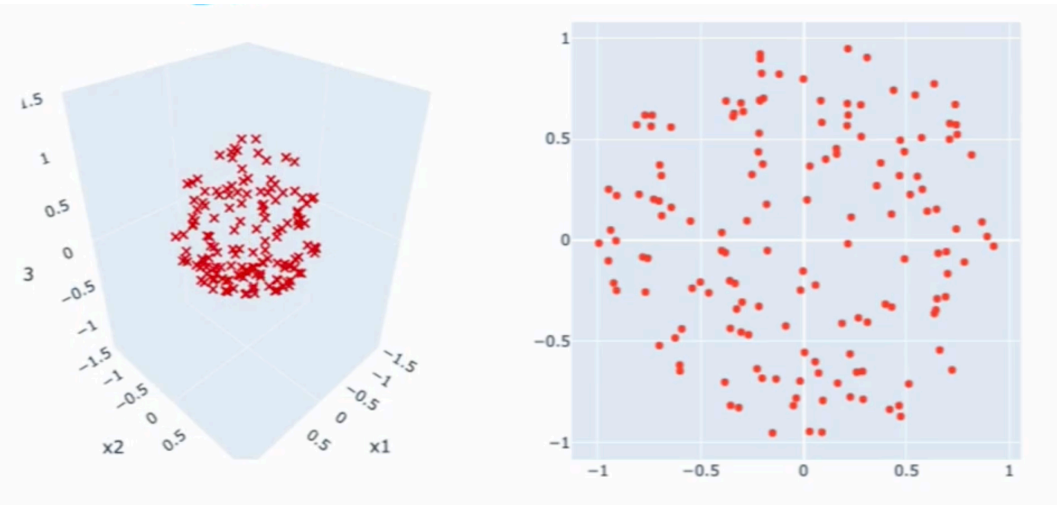




Examples of PCA Application

In the case of passenger cars, we consider two features of a car: its **length** and **height** (as shown in the figure). Since these two measurements are directly correlated, they are not independent. Principal Component Analysis (PCA) identifies a new axis that can capture most of the variation in the data using just one value. In this case, that axis could be interpreted as representing the overall size of the car.





- For datasets with multiple features, PCA can create new axes that capture essential information, allowing for effective visualization.

Practical Use of PCA

- PCA is commonly applied to datasets with numerous features, such as economic indicators for countries, to simplify analysis and visualization.
- By reducing dimensions, PCA helps in understanding the underlying structure of the data and identifying patterns or anomalies

PCA algorithm

Here is a note summarizing how PCA works, based on the explanation you provided and the attached file, written in an original way:

How Does PCA Work?

- Start with original data points plotted in a multi-dimensional space (e.g., with features x_1, x_2).
- Suppose you want to reduce two features down to one. PCA finds a **new axis (called the principal component)** that best captures the **spread** or **variance** of the data.
- This new axis is chosen such that when all data points are **projected** onto it (by drawing perpendicular lines to the axis), the projected points are as spread out as possible, capturing the most variance.

- Example: Different axis choices capture different amounts of variance. The principal component is the axis that maximizes this variance.

Important Steps in PCA:

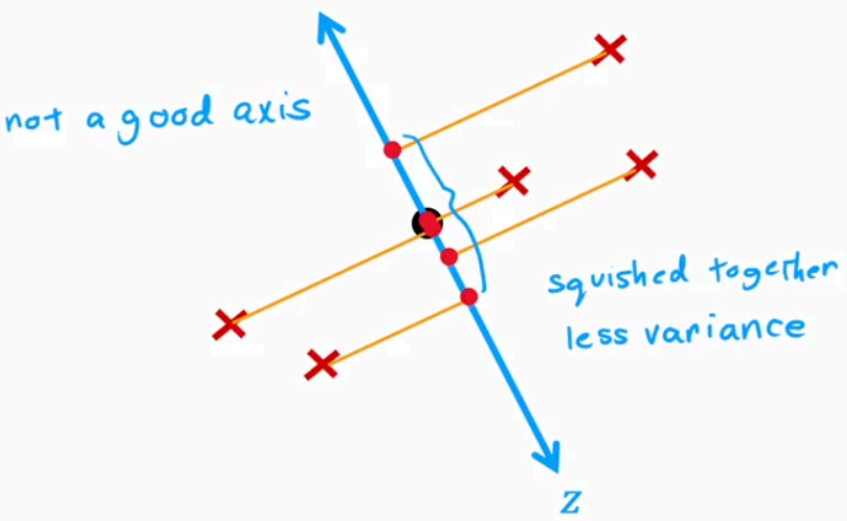
1. **Preprocessing:**

   Normalize features to have zero mean and similar scales, so features with large scales don't dominate the principal components.

2. **Principal Component Selection:**

   Find unit vectors representing new axes (principal components). The first principal component captures the greatest variance, the second is perpendicular to the first and captures the next largest variance, and so on.

What is the bad choosing an axis?



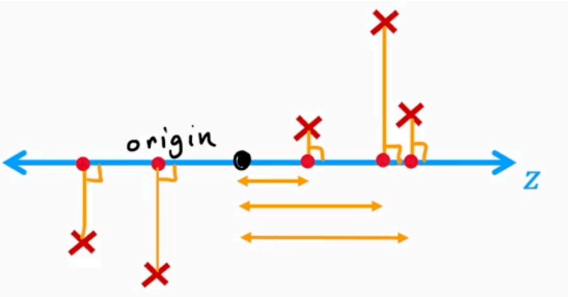
When projecting multidimensional data onto this axis, the points become very close together (or clumped), meaning the **spread (variance) of the data along that axis is small**. ⇒ the chosen axis doesn't represent the **differences** or important information in the data well

⇒ As a result, reducing the data to this axis causes **loss of significant information**, making it a poor representation of the original dataset.

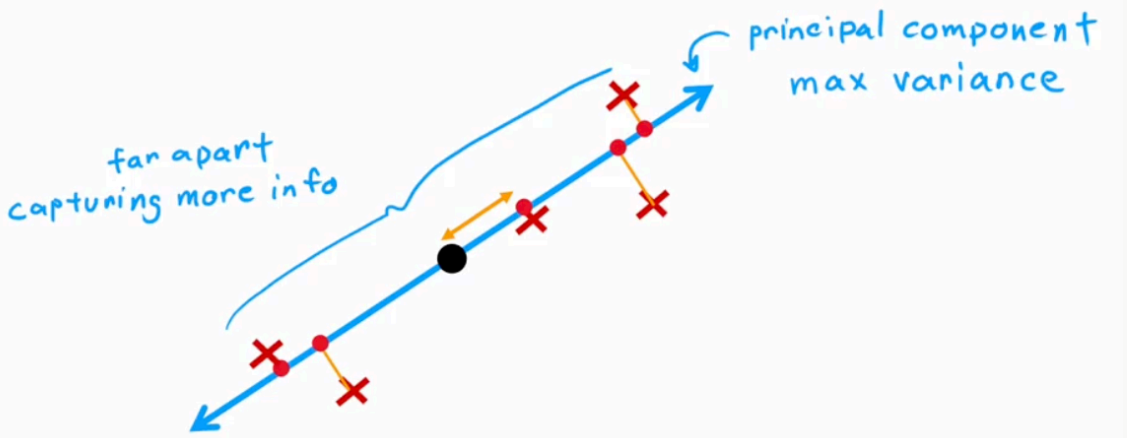
Example: Imagine we have a group of people standing in a park, and we want to take a photo that captures the most people in the frame. If we choose to take the photo from a position that faces a wall, you will only see a few people squished together, missing the majority of the group. ⇒ This is similar to choosing a poor axis in PCA.

3. **Projection:**

   Each data point is projected onto these new axes using dot product operations, yielding new coordinates with fewer dimensions.



Good choice of axis ⇒ max variance

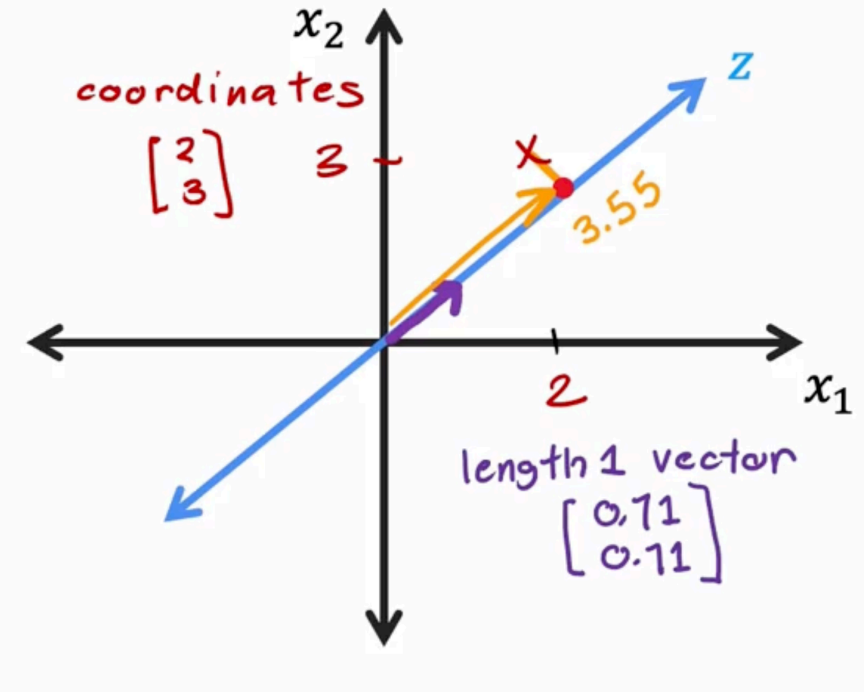


4. **Reconstruction (Approximation):**

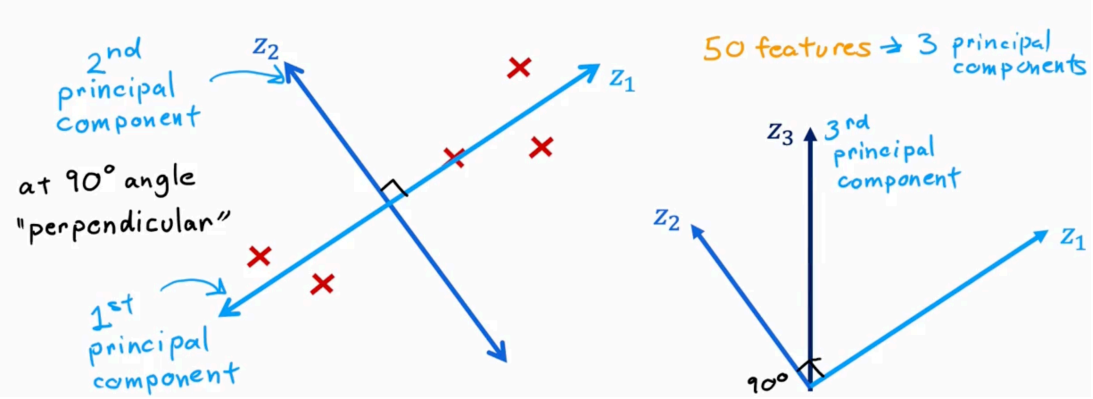
   It is possible to approximately reconstruct the original data from its lower-dimensional representation by reversing the projection, though some information is lost.

   - Once the new **z-axis** is established, represented by the unit vector (z_1, z_2), any data point (x_1, x_2) can be projected onto this axis. This is done by computing the dot product between the data vector and the axis vector. (Matrix Multiplication)

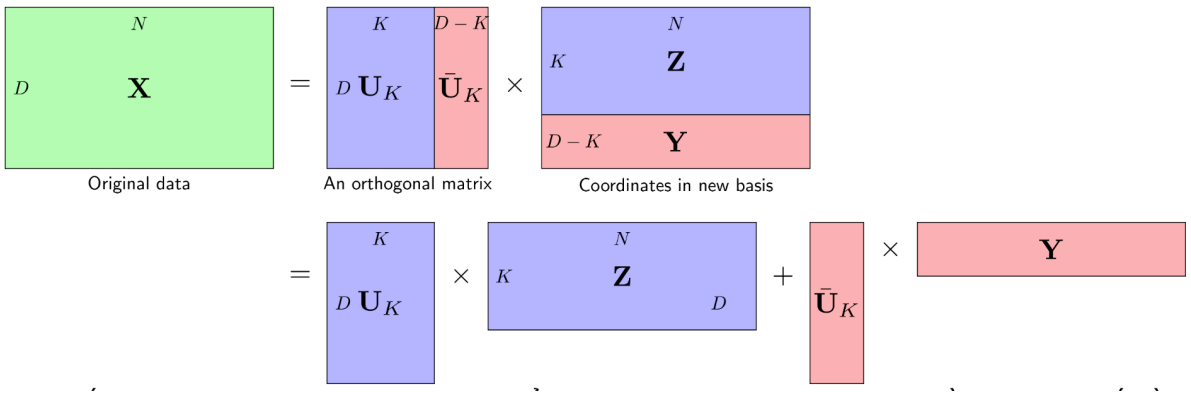
     $$z = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \cdot \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} 2 \\ 3 \end{bmatrix} \cdot \begin{bmatrix} 0.71 \\ 0.71 \end{bmatrix} = 2 \cdot 0.71 + 3 \cdot 0.71 = 3.55$$



- When additional axes are generated in later steps, each of them is constructed to be orthogonal (perpendicular) to all the previously defined axes.



The diagram show the idea behind PCA:



The Idea of PCA: https://machinelearningcoban.com/2017/06/15/pca

---

Differences from Linear Regression:

- PCA is **unsupervised** (no output labels), whereas *linear regression is supervised* (predicts output from inputs).
- PCA minimizes **distance** from data points to the axis **perpendicularly**, treating all features equally.
- Linear regression minimizes vertical distance to a line predicting a dependent variable.
- PCA aims to reduce features while preserving information; linear regression aims to find relationships between variables.

Code Implementation (e.g., with scikit-learn):

- Scale and normalize data.
- Use PCA's `fit` method to find principal components.
- Use `explained_variance_ratio_` to understand how much variance each principal component captures.
- Use `transform` to project data onto principal components.

```
# given an input array of features X, fit the data
# to n principle components
pca_1 = PCA(n_components=1)
pca_1.fit(X)

# captures percent of variability from original data;
# one value per principle component
pca_1.explained_variance_ratio_

# project each training example to a single value
# for the final, single feature
X_trans_1 = pca_1.transform(X)
X_reduced_1 = pca.inverse_transform(X_trans_1)
```

Applications of PCA:

- Mainly used for **visualizing** high-dimensional data in 2D or 3D.
- Historically used for **data compression** and **speeding up training** in some models, but less common now due to advances in storage and computing power.