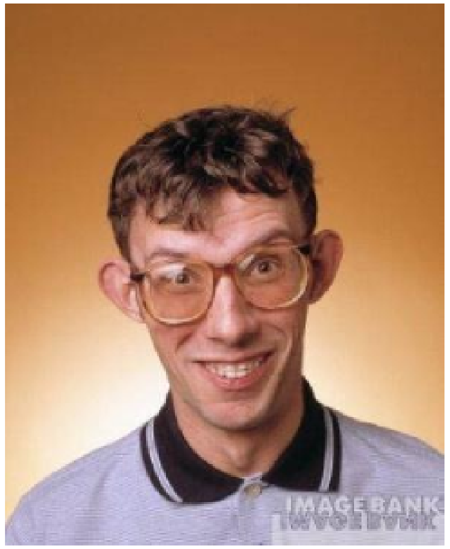
Principal Component Analysis

Motivation

Principal Component Analysis (PCA) is a mathematical technique used for

- reducing dimensions.
- data compression.

Example: What makes a nerd, well, nerd?



- Clothes.
- Weird body language.
- Glasses.
- Number of weird remarks per hour.
- Percentage of rejections from potential partners.
- Many more...

Some of this features are correlated, so it does not make sense to keep track of all of them.

Example: Text classification

When doing text classification, we usually end up with much more features (columns) than data points (rows).

Is that a problem?

Yes, and specially for **unsupervised learning**.

Curse of dimensionality

In high-dimensional space,there is little difference between pairs of samples: this complicates life for algorithms like $k-$means.

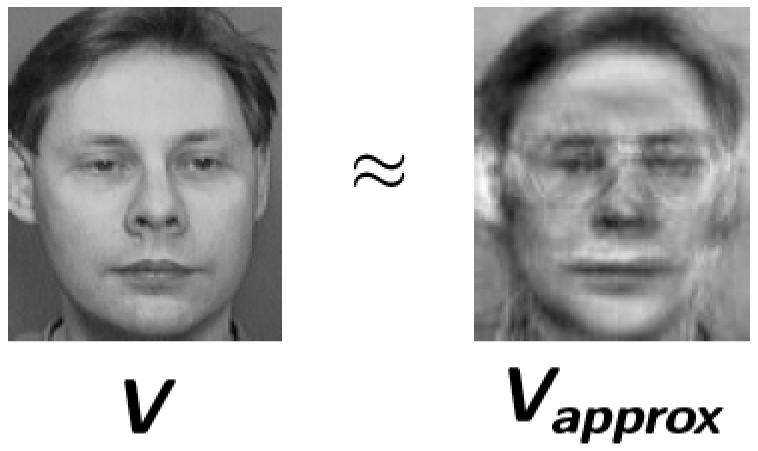
An illustration of this:

- Consider a d dimensional unit sphere and carve out a d dimensional sphere of radius $r < 1$ from it. What do you get?

Image compression

A 600x800 image is a point in $\mathbb{R}^{480000}$!

Definitely worth trying to save some space for analysis..



The computer stores a representation of the image that "makes sense" for (her/him/it)?

Math background (I)

- Let $X, Y\colon \Omega \to \mathbb{R}$ be integrable random variables whose product XY is also integrable. The **covariance** of X and Y is defined as

$$\operatorname{Cov}(X, Y) := \mathbb{E}[(X - \mu_X)(Y - \mu_Y)]$$

- If $X = Y, \operatorname{Cov}(X, Y) = \operatorname{Var}(X) = \mathbb{E}[(X - \mu_X)^2]$ is the expected quadratic deviation of X from its mean μ_X.

Definition (Covariance matrix): Let $X = (X_1, \dots, X_n) \colon \Omega \to \mathbb{R}^n$ be a random vector. Its **covariance matrix** is defined as

$$\mathrm{Cov}(X) := \begin{pmatrix} \mathrm{Var}(X_1) & \cdots & \mathrm{Cov}(X_1, X_n) \\ \mathrm{Var}(X_2) & \cdots & \mathrm{Cov}(X_2, X_n) \\ \vdots & \vdots & \ddots \\ \mathrm{Cov}(X_n, X_1) & \cdots & \mathrm{Var}(X_n) \end{pmatrix}$$

Pearson correlation coefficient

Let $X, Y\colon \Omega \to \mathbb{R}$ be integrable random variables whose product XY is also integrable. The **Pearson correlation coefficient** of X and Y is defined as

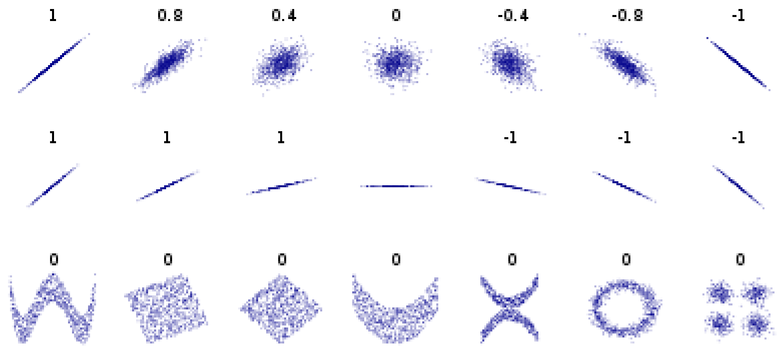
$$\rho(X, Y) = \frac{\mathrm{Cor}(X, Y)}{\mathrm{SD}(X)\,\mathrm{SD}(Y)} \in [-1, 1]$$

Correlation

- Correlations equal to ± 1 are collinear points.
- Values "close" to 1 are interpreted as "the bigger the x, the bigger the y".
- Values "close" to -1 are interpreted as "the bigger the x, the smaller the y".

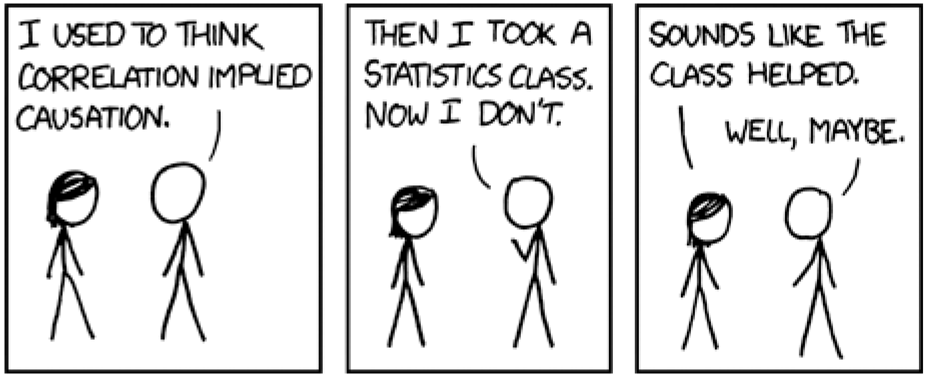
- "Close" depends on application and subject standards.
- $\mathrm{Cov}(X, Y) = 0$ means **uncorrelated**
- $\mathrm{Cov}(X, Y) \neq 0$ means **correlated**.

Correlation sometimes does not mean anything



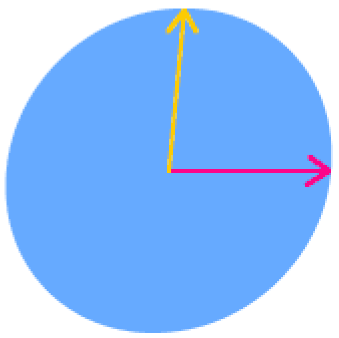
Pearson correlation tells us about **linear** dependence between variables.
In the lower row,

Correlation vs causation


I USED TO THINK CORRELATION IMPLIED CAUSATION.
THEN I TOOK A STATISTICS CLASS. NOW I DON'T.
SOUNDS LIKE THE CLASS HELPED.
WELL, MAYBE.

PCA

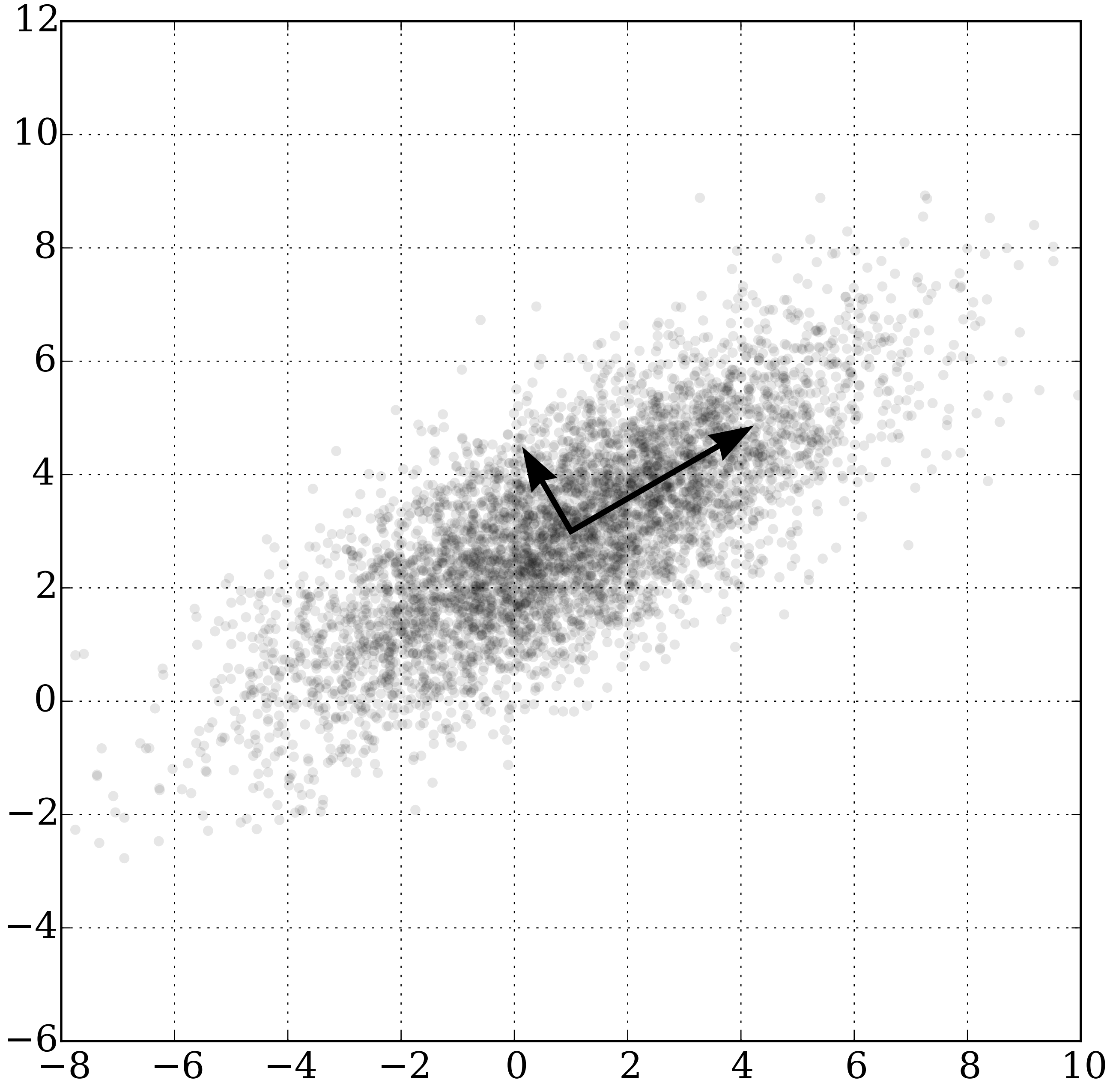
- Every linear transformation can be decomposed on three parts: a rotation, a scaling and a final rotation.



$$M = \begin{bmatrix} M_{1,1} & M_{1,2} \\ M_{2,1} & M_{2,2} \end{bmatrix}$$

PCA (cont)

- Principal Components: the first part of the transformation (the eigenvectors of the covariance matrix).
- The i $-$th eigenvector, λ_i, explains a total of $\frac{\lambda_i}{\sum_j \lambda_j}$ of the variance in the data.



12
10
8
6
4
2
0
−2
−4
−6
−8
−6
−4
−2
0
2
4
6
8
10

Example

Goal: Create a data matrix X with covariance matrix

$$\mathrm{Cov}(X) = \begin{pmatrix} 0.3 & 0.2 \\ 0.2 & 1.0 \\ & \end{pmatrix}$$

and then

- plot the data
- compute the eigenvalues and eigenvectors (= principal components) of this covariance matrix
- rotate the data to the coordinate system given by the principal components
- plot the rotated data

Create the data

```python
import matplotlib.pyplot as plt
import numpy as np
# For reproducibility of the results, set the seed:
np.random.seed(0)

mean = np.array([0, 0])
cov = np.array([[0.3, 0.2],
                [0.2, 1.0]])

# Draw 500 samples from a multivariate normal
# distribution with mean `mean`
# and covariance matrix `covariance`:
data = np.random.multivariate_normal(mean, cov, 500)
# Draw a scatter plot of the data:
plt.scatter(data[:,0], data[:,1], color = '.05')
plt.show()
```

Eigenvalues and eigenvectors

```
# Compute the eigenvalues and eigenvectors
# of the covariance:
eig_val, eig_vec = np.linalg.eig(cov)
print("Eigenvalues:" + str(eig_val))
print("Eigenvectors:" + str(eig_vec[0]) \
+ " and " + str(eig_vec[1]))
```

Calculate the principal component

```
b1 = eig_vec[1]
b2 = eig_vec[0]
# This prints the basis vectors
# and confirms that the they have unit length:
b1, b2, np.linalg.norm(b1), np.linalg.norm(b2)
```

PCA in sci-kit learn

```
from sklearn.decomposition import PCA
print(PCA.transform.__doc__)
```

PCA transformation

```
# Perform PC-Transformation:
from sklearn.decomposition import PCA
pca = PCA(n_components=2)
pca.fit(data)
print(pca.components_)
```

Look at the mean of the data (we know it's zero, by construction)

```
np.mean(data, axis=0)
```

```python
# Perform PC-Transformation:
from sklearn.decomposition import PCA
pca = PCA(n_components=2)
pca.fit(data)
data_transformed = pca.transform(data)

# Input data in blue:
plt.scatter(data[:,0], data[:,1], color = 'blue')

# Transformed data in orange:
plt.scatter(data_transformed[:,0], \
data_transformed[:, 1], color = 'orange')
plt.show()
```

Example: Handwritten digits

Use one of `sklearn` datasets.

```python
from sklearn.datasets import load_digits
digits = load_digits()
# The images are given by `digits.data'
# and the corresponding labels are
# given in `digits.target`:
X_digits, y_digits = digits.data, digits.target
print(digits.keys())
```

More about this dataset:

`print(digits.DESCR)`

Looking at the data

```
len(digits.images)
plt.imshow(digits.images[15])
```

Doing PCA

```python
from sklearn.decomposition import PCA
pca = PCA(n_components=10)
X_pca = pca.fit_transform(X_digits)
for i, evr in enumerate(pca.explained_variance_ratio_):
    print("Variance explained by the " \
    + str(i + 1) + "-th principal component:\t" + \
    + str(evr))
```

```python
def plot_pca_scatter():
    colors = ['black', 'blue', 'purple', 'yellow',\
    'white', 'red', 'lime', 'cyan', 'orange', 'gray']
    for i in range(len(colors)):
        px = X_pca[:, 0][y_digits == i]
        py = X_pca[:, 1][y_digits == i]
        plt.scatter(px, py, c=colors[i])
        plt.legend(digits.target_names)
        plt.xlabel('First Principal Component')
        plt.ylabel('Second Principal Component')

n_components=2
plot_pca_scatter()
X_pca.shape, X_digits.shape
```

How do the components themselves look like?

```
plt.imshow(pca.components_[1].reshape((8,8)))
```

Linear Discriminant Analysis

- Aka dimensionality reduction for supervised learning problems

Goal: Model the class conditional distribution of the data, $\mathbb{P}(X \mid y = k)$, and make predictions via **Bayes' rule:**

$$\mathbb{P}(y = k \mid X) = \frac{\mathbb{P}(X \mid y = k)\mathbb{P}(y = k)}{\mathbb{P}(X)}$$

and choose the class that maximizes the conditional probability.

- The conditional distribution of the data is modeled as a multivariate Gaussian distribution:

$$\mathbb{P}(X \mid y = k) \approx \exp\left(-\frac{1}{2}(X - \mu_k)^T \Sigma_k^{-1} (X - \mu_k)\right)$$

- In the case of LDA, we assume that each class has the same covariance matrix, hence:

$$\log\left(\frac{\mathbb{P}(X \mid y=k)}{\mathbb{P}(X \mid y=\ell)}\right)=0$$

if and only if

$$(\mu_k-\mu_\ell)\Sigma^{-1}X=\frac{1}{2}(\mu_k^T\Sigma^{-1}\mu_k-\mu_\ell^T\Sigma^{-1}\mu_\ell)$$