Welcome!
#pod-031
Week1, Day5

(Reviewed by Deepak)

neuromatch academy

facebook Reality Labs

Agenda

- Tutorial part 1 (Orthonormal basis Vectors)
- Tutorial part 2 (Dimensionality Reduction Analysis)
  - Principal Component Analysis
- Tutorial part 3 (Dimensionality Reduction and Reconstruction)
- Tutorial part 4 (Nonlinear Dimensionality Reduction)
  - t-SNE

W1D5_pod 031

Basis vectors

In mathematics, a set B of elements in a vector space V is called a basis, if every element of V may be written in a unique way as a linear combination of elements of B. The coefficients of this linear combination are referred to as components or coordinates on B of the vector.

Orthonormal basis

In 3 dimensions, standard basis are (1,0,0), (0,1,0), and (0,0,1).

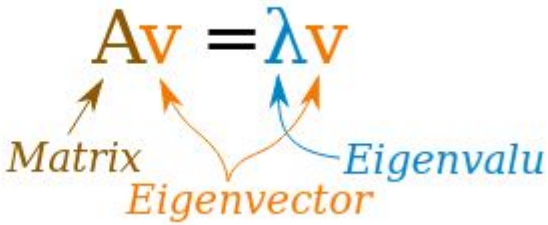
Let $S = \{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k\}$ be a set of vectors in R^n, then S is called an *orthogonal* if

$$\mathbf{v}_i \cdot \mathbf{v}_j = 0$$

for all i not equal to j. An orthogonal set of vectors is called *orthonormal* if all vectors in S are unit vectors.

Eigen Vectors and values

For a square matrix **A**, an Eigenvector and Eigenvalue make this equation true:



If the new transformed vector is just a scaled form of the original vector then the original vector is known to be an eigenvector of the original matrix. Vectors that have this characteristic are special vectors and they are known as eigenvectors. Eigenvectors can be used to represent a large dimensional matrix.

Correlations and covariance

"Covariance" indicates the direction of the linear relationship between variables. "Correlation" on the other hand measures both the strength and direction of the linear relationship between two variables.

Variance

In probability theory and statistics, variance is the expectation of the squared deviation of a random variable from its mean. Informally, it measures how far a set of numbers is spread out from their average value

Ref: http://www.cvip.louisville.edu/wordpress/wp-content/uploads/2010/01/PCA-Tutorial-Sept-09-1.pdf

Relationship between variance & correlation

$$\text{variance}(x) = \text{covariance}(x, x)$$

Relationship between covariance & correlation

$$\text{correlation}(x, y) = \frac{\text{covariance}(x, y)}{\sigma_x \sigma_y}$$

σ_x: standard deviation of x

σ_y: standard deviation of y

So, increase in covariance results in increased correlation.

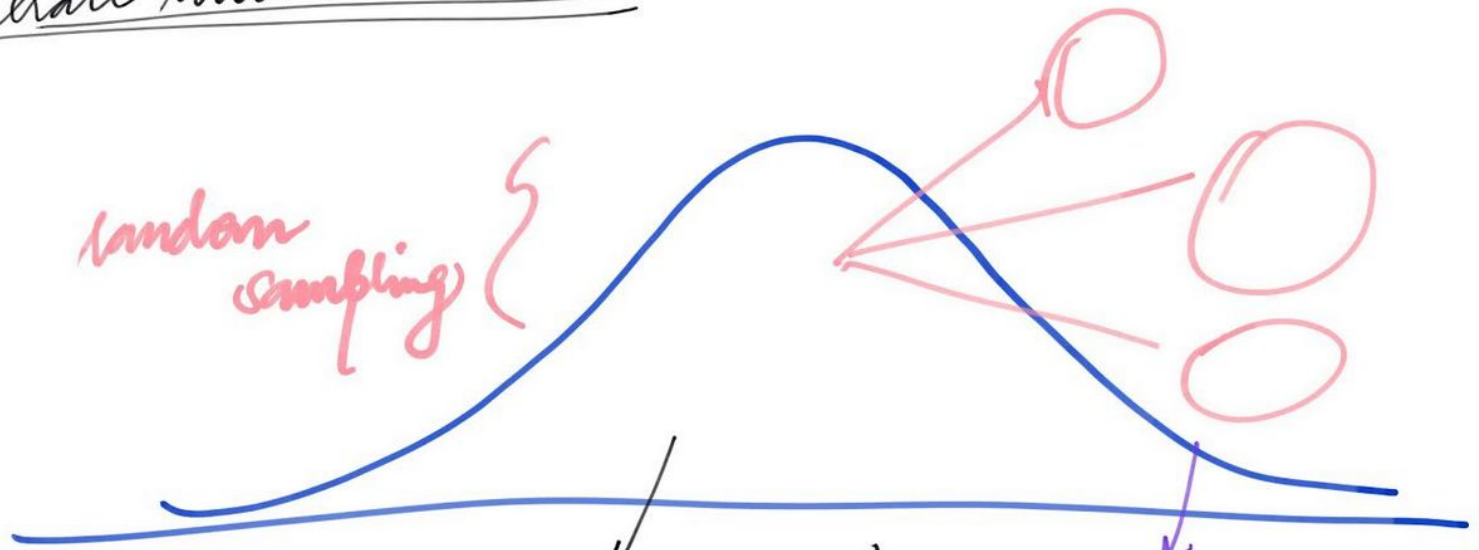
Tutorial #1 Explanations

Overview

How multivariate data can be represented in different orthonormal bases.

- *Generate correlated multivariate data.*
- *Define an arbitrary orthonormal basis.*
- *Project the data onto the new basis.*

Generate Multivariate data



bivariate normal distribution

x_i is marginally normal with μ_i & σ_i^2

$x_i \sim N(\mu_i, \sigma_i^2)$

represent activity / firing rates of two recorded neurons on different trials

Joint distribution for x_1 & x_2

→ correlational coefficient ρ

Normalised covariance $[-1, +1]$

$$\rho = \frac{cov(x_1, x_2)}{\sqrt{\sigma_1^2 \sigma_2^2}}$$

$\mu_i = 0$

$\Rightarrow$ covariance matrix $\Sigma = \begin{pmatrix} var(x_1) & cov(x_1, x_2) \\ cov(x_1, x_2) & var(x_2) \end{pmatrix}$

$$\Sigma = \begin{pmatrix} \sigma_1^2 & cov(x_1, x_2) \\ cov(x_1, x_2) & \sigma_2^2 \end{pmatrix}$$

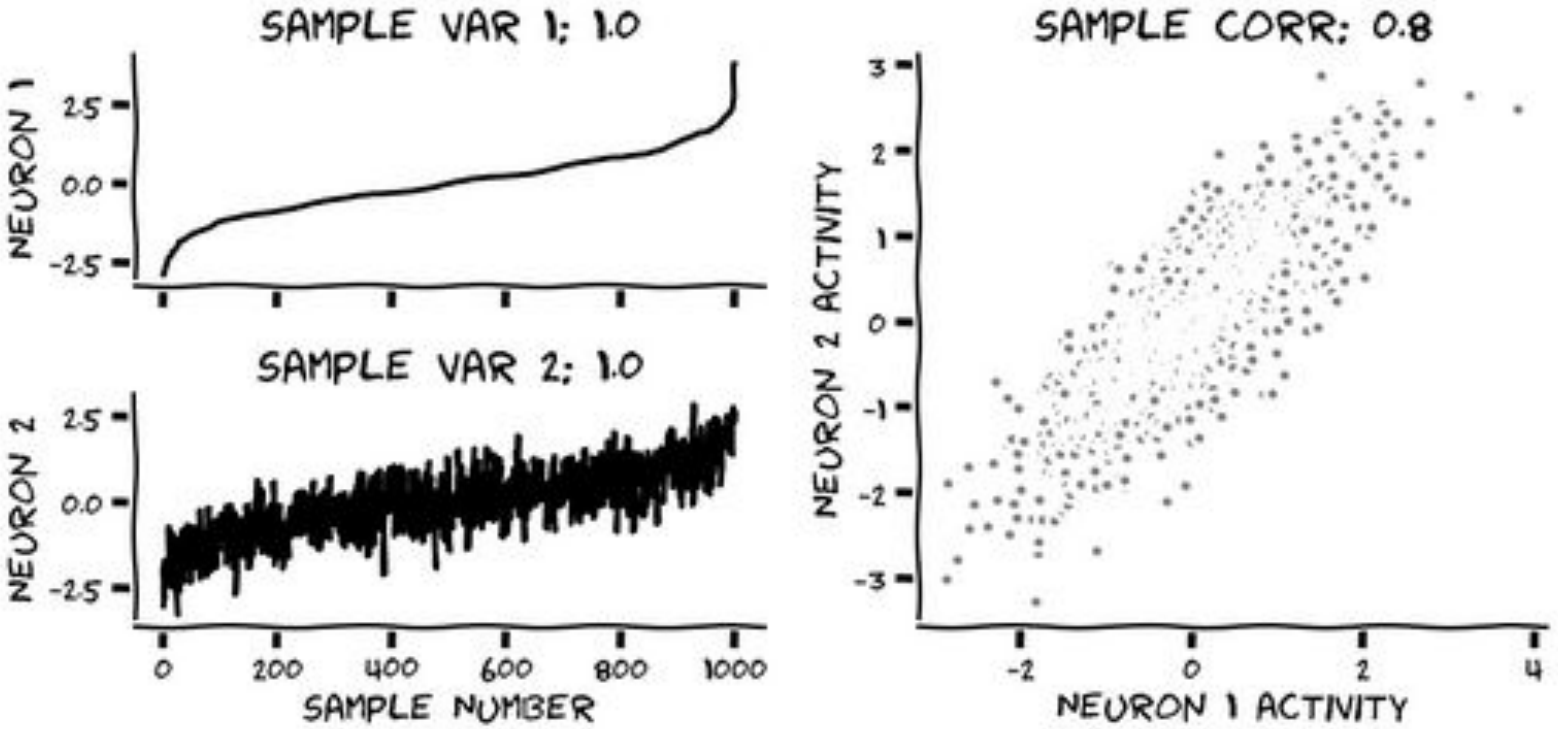
Geometric view of data points

Simply, think of the data points as vectors in a k dimensional space (k=2 in our case).

$$cov(x_1, x_2) = s\sqrt{\sigma_1^2 \sigma_2^2}$$

reflect how changing s affects geometry of simulated data

Effect of varying variance



Covariance and Orthonormal basis

$u = [u_1, u_2]$ & $w = [w_1, w_2]$

2 vectors are orthonormal if:

① $u \cdot w = u_1 w_1 + u_2 w_2 = 0$

② $\|u\| = \|w\| = 1$

→ New arbitrary orthonormal basis.

normalised random vector u

let $w = [-u_2, u_1]$

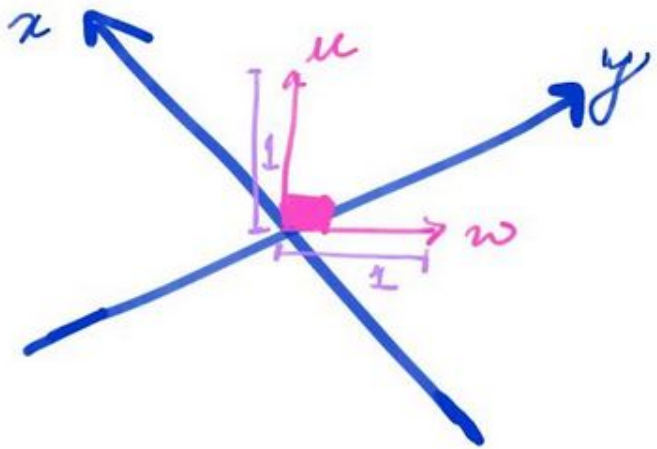
$\|w\| = \sqrt{(-u_2)^2 + u_1^2} = 1$

$u \cdot w = -u_1 u_2 + u_2 u_1 = 0$

stack basis vectors horizontally

$$W = \begin{pmatrix} u_1 & w_1 \\ u_2 & w_2 \end{pmatrix}$$

New orthonormal basis.



$$W = \begin{pmatrix} u_1 & w_1 \\ u_2 & w_2 \end{pmatrix}$$

$$\left\{ u \cdot w = -u_1 u_2 + u_2 u_1 = 0 \right\}$$

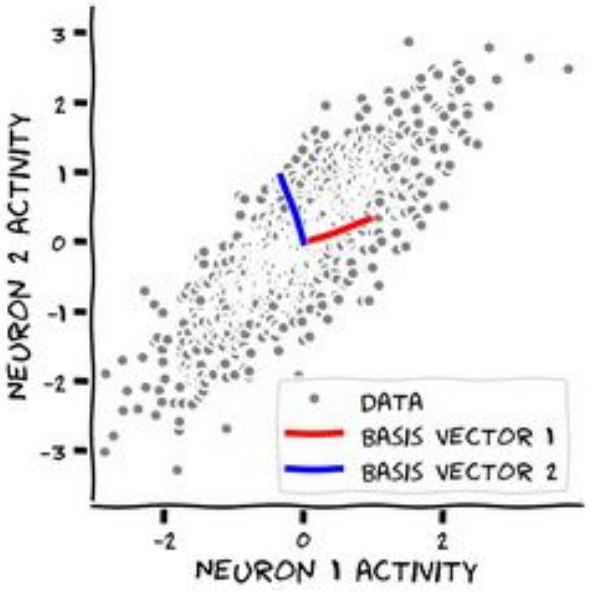
Defining new orthonormal bases through projection

W in terms of new basis :

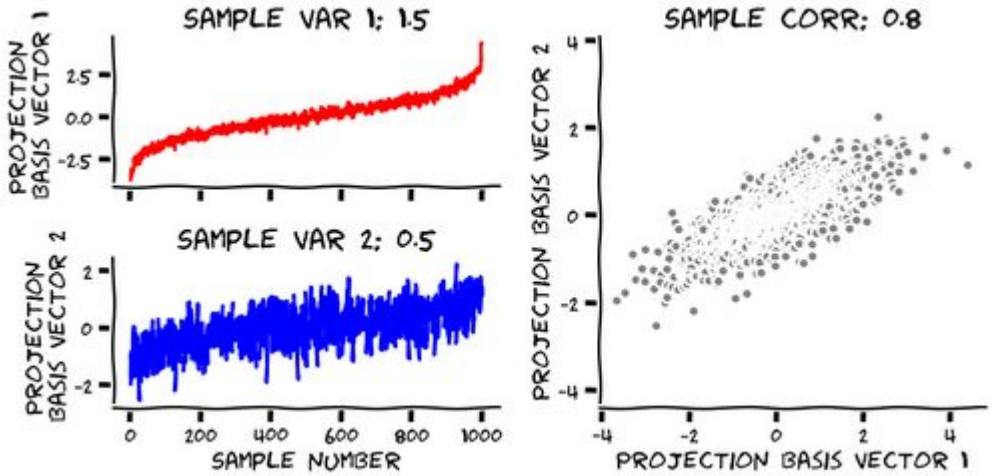
$Y = X W$ → orthonormal

(projecting onto new basis)

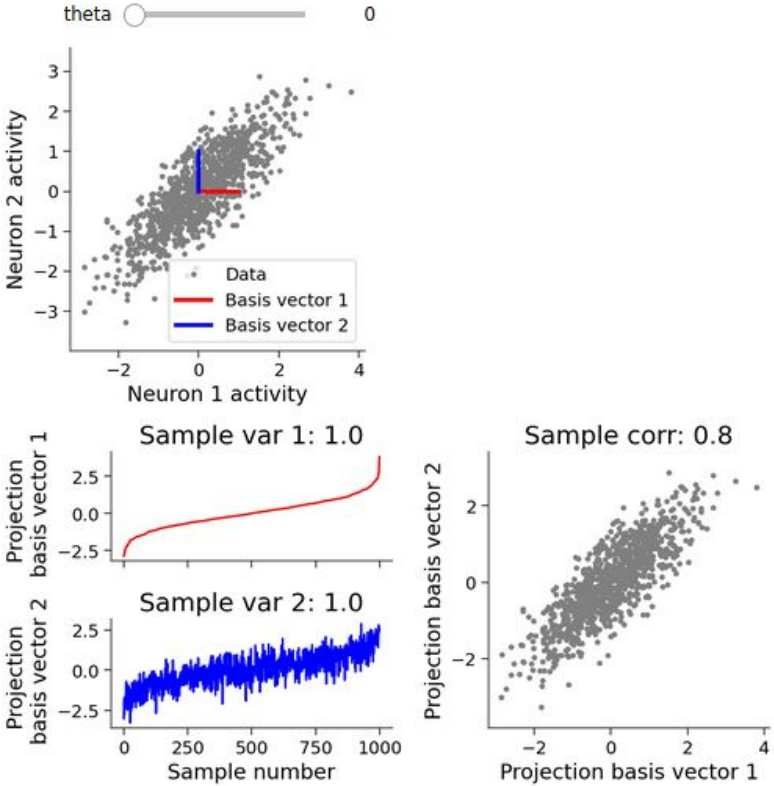
→ transformed data



Defining new orthonormal bases through projection

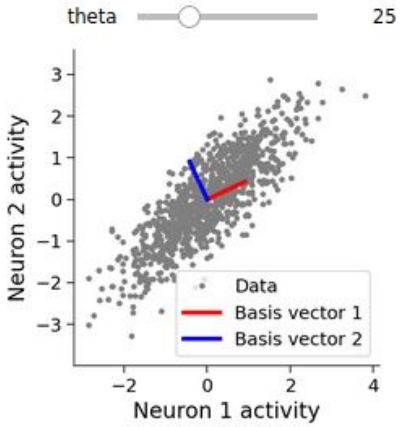


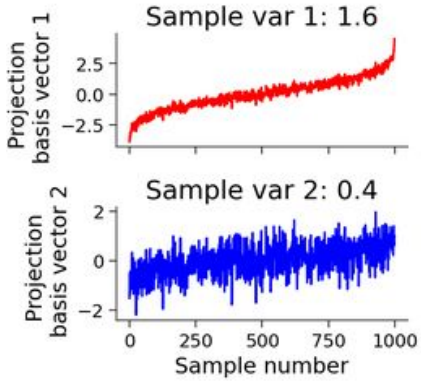
Basis vectors

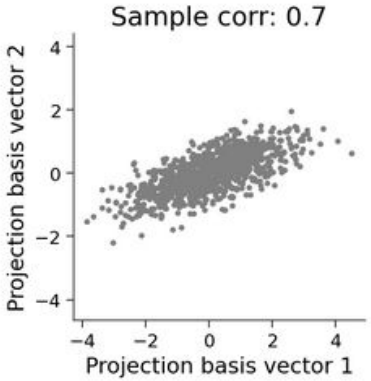


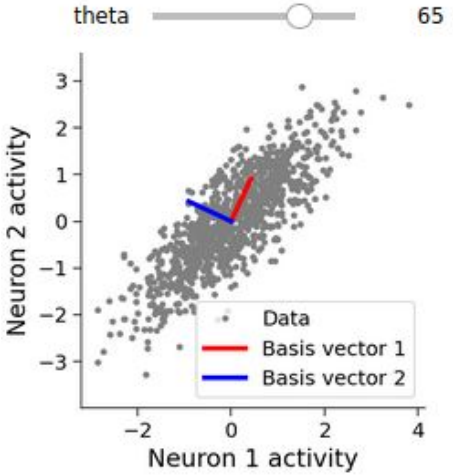
What happens to the projected data as you rotate the basis?

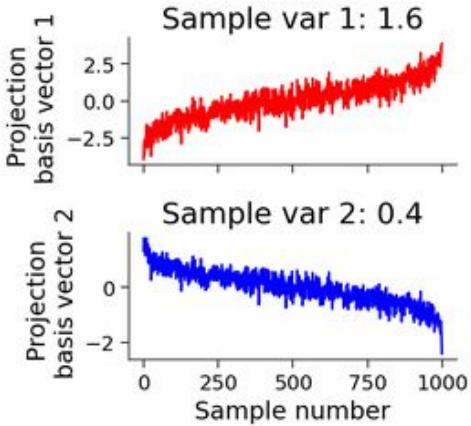
Sample correlation value changes.

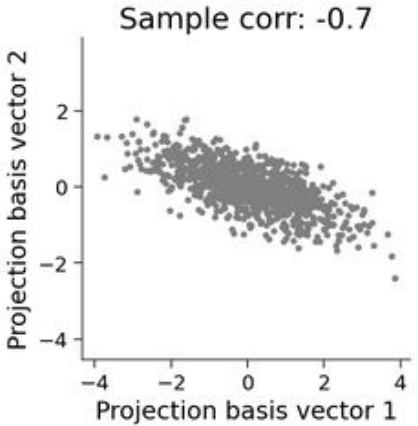












Effect of theta on correlation values

What happens to the correlation coefficient in the new basis? Does it increase or decrease?

As theta increases, sample correlation value/ correlation coefficient decreases.

Why?

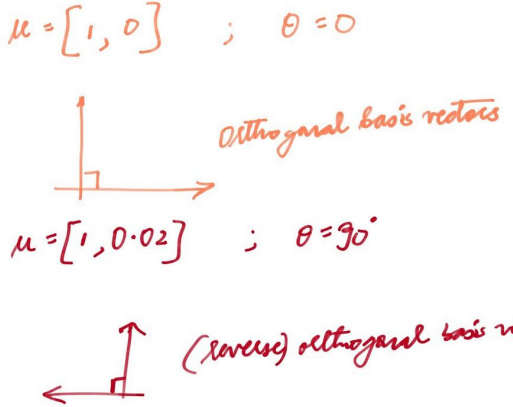
Theta = 0, corr = 0.8

Theta = 90, corr = -0.8

U = [1, tan(theta * np.pi / 180)]

What happens to variance?

Variance is directly proportional to correlation;



$$\frac{\text{covariance}(x,y)}{\sigma_x \sigma_y} = \text{correlation}$$

$$\text{covariance}(x,y) = \text{correlation} \times \text{std dev}$$

How does the correlation coefficient change? How does the variance of the projection onto each basis vector change?

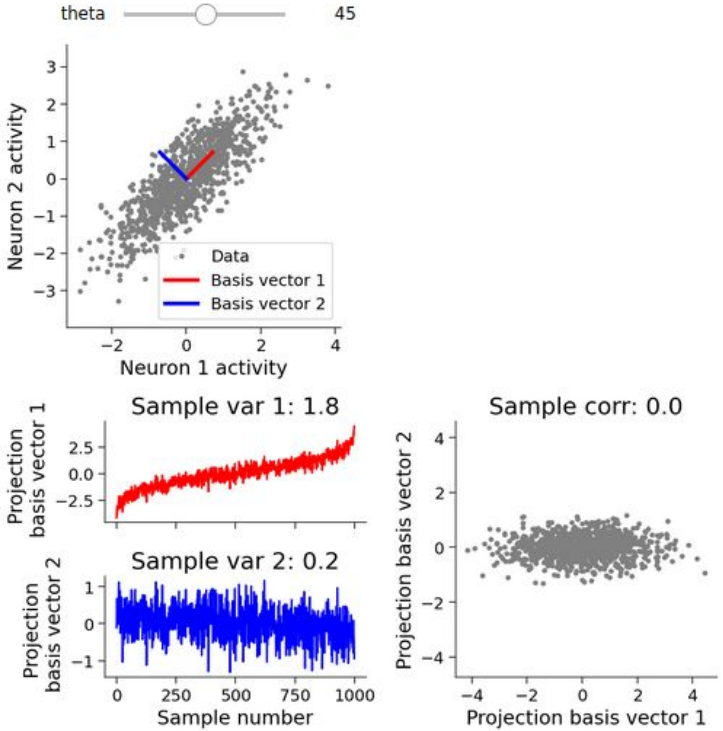
As theta increases, sample correlation value/correlation coefficient decreases.

Variance is directly proportional to correlation;

The projected data (after transforming into the new basis) will generally have a different geometry from the original data. In particular, taking basis vectors that are aligned with the spread of cloud of points decorrelates the data.

Are you able to find a basis in which the projected data is uncorrelated?

Projected data is uncorrelated means covariance is zero and dimensions in the vector space are independent.



WHY?

Why Vectors at all?

In neurobiological modeling we are often dealing with arrays of variables: the activities of all of the neurons in a network at a given time; the firing rate of a neuron in each of many small epochs of time; the weights of all of the synapses impinging on a postsynaptic cell. The natural language for thinking about and analyzing the behavior of such arrays of variables is the language of vectors and matrices. In problems of theoretical neuroscience, we are often dealing with large sets of variables — the activities of a large set of neurons in a network; the development of a large set of synaptic strengths impinging on a neuron. The equations to describe models of these systems are usually best expressed and analyzed in terms of vectors and matrices.

Multivariate data can be represented in a new orthonormal basis using the dot product. These new basis vectors correspond to specific mixtures of the original variables - for example, in neuroscience, they could represent different ratios of activation across a population of neurons.

Tutorial #2 Explanations

Principal component analysis

Perform PCA by projecting the data onto the eigenvectors of its covariance matrix.

Overview:

- Calculate the eigenvectors of the sample covariance matrix.
- Perform PCA by projecting data onto the eigenvectors of the covariance matrix.
- Plot and explore the eigenvalues.

Calculate the eigenvectors of the the sample covariance matrix

$$\Sigma_{ij} = E[x_i x_j] - E[x_i]E[x_j]$$

i,j th element of covariance matrix

eigenvectors

} unknown ground truth covariance matrix

$$\hat{\Sigma}_{ij} = \frac{1}{N} x_i^T x_j - \bar{x}_i \bar{x}_j$$

sample covariance matrix

$x_i = [x_{i_1}, x_{i_2} \dots x_{i_N}]^T$ # column vector of all measurements of neuron i

$\bar{x}_i$: mean of neuron i across all samples

$$\bar{x}_i = \frac{1}{N} \sum_{k=1}^{N} x_{i_k}$$

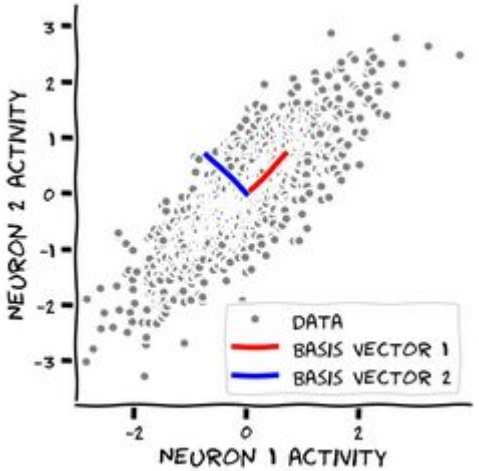
$$\Longrightarrow \hat{\Sigma} = \frac{1}{N} X^T X$$

Eigenvectors of covariance matrix

To calculate the eigenvectors of the covariance matrix to check that they align with the geometry of the data.

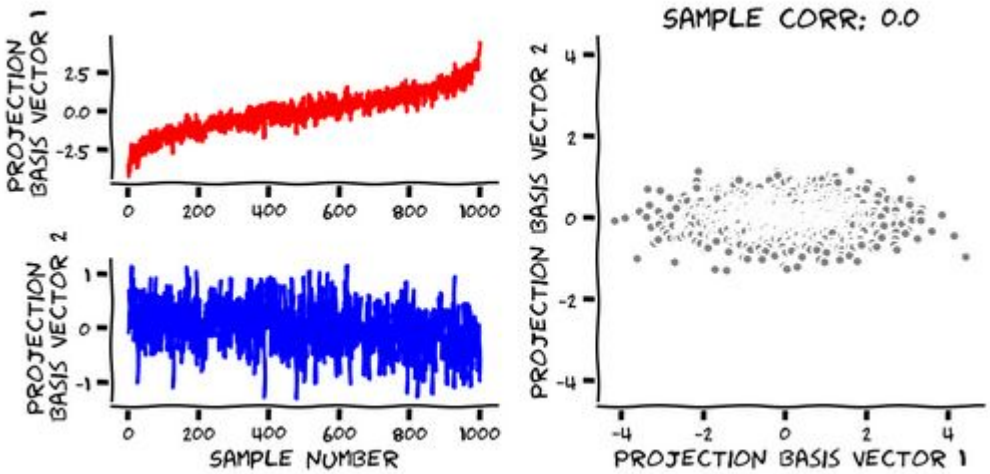
Steps:

- Calculate the eigenvalues and eigenvectors of the sample covariance matrix. (**Hint:** use np.linalg.eigh, which finds the eigenvalues of a symmetric matrix).
- Use the provided code to sort the eigenvalues in descending order.
- Plot the eigenvectors on a scatter plot of the data, using the function plot_basis_vectors.



PCA

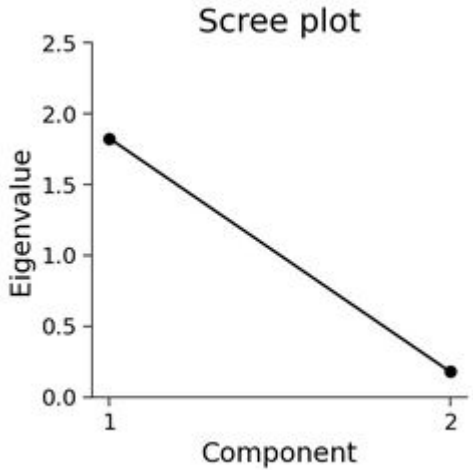
Principal Component Analysis is a dimensionality-reduction method that is often used to reduce the dimensionality of large data sets, by transforming a large set of variables into a smaller one that still contains most of the information in the large set.



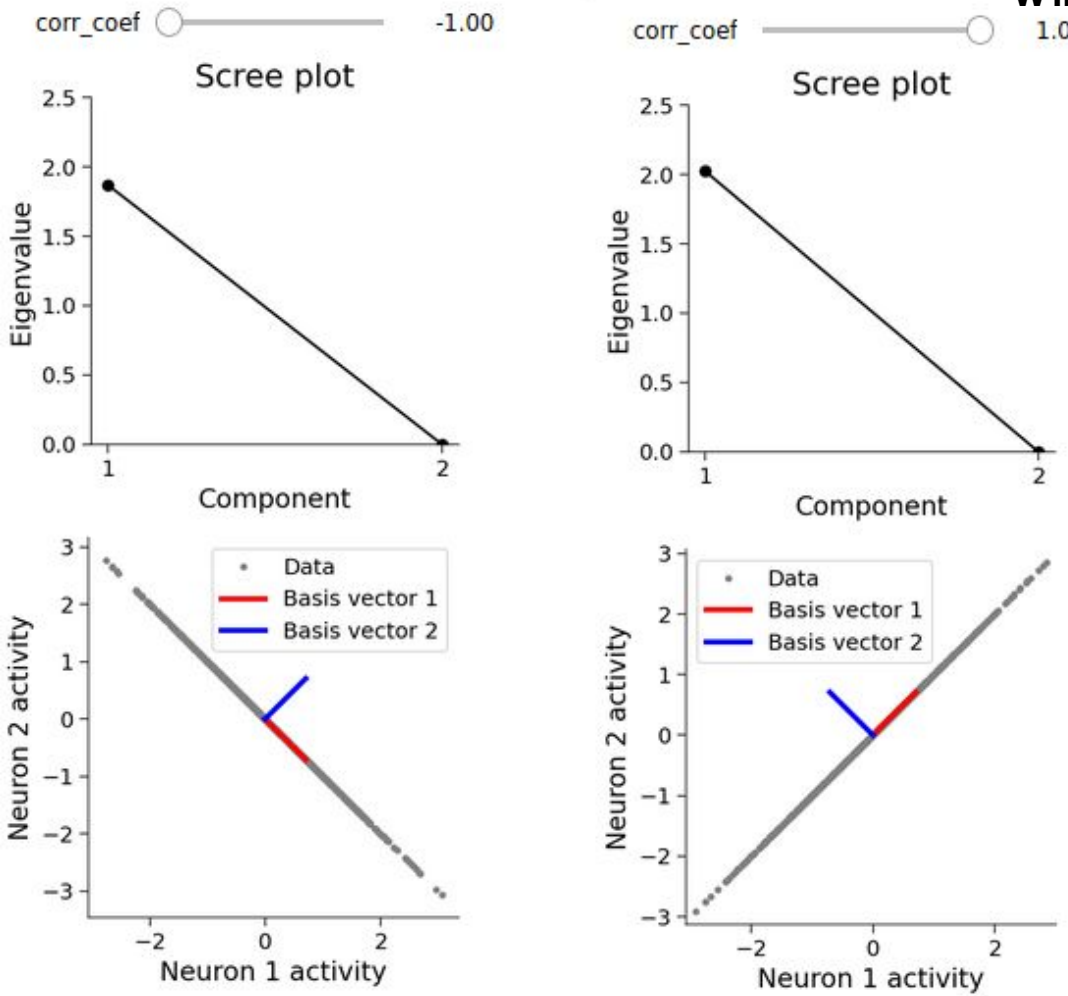
Perform PCA by projecting data onto the eigenvectors

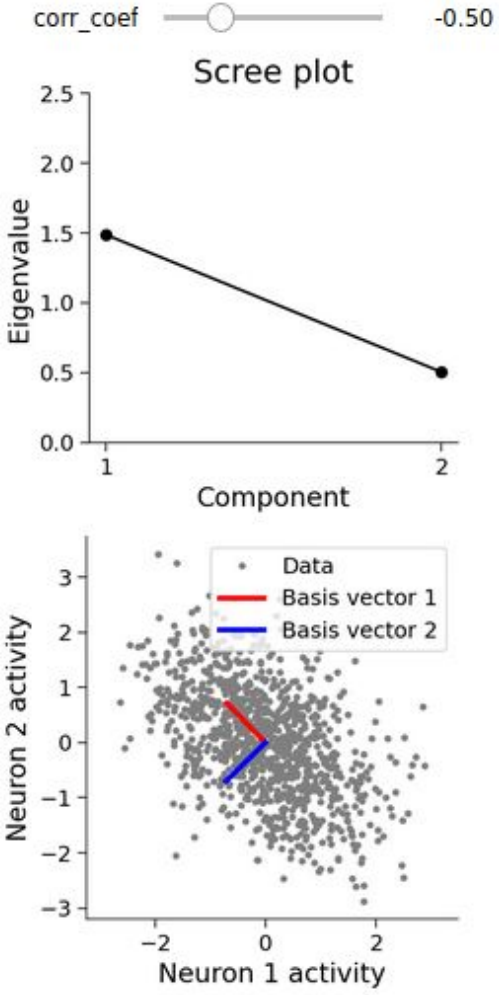
Each eigenvalue describes the variance of the data projected onto its corresponding eigenvector. This is an important concept because it allows us to rank the PCA basis vectors based on how much variance each one can capture

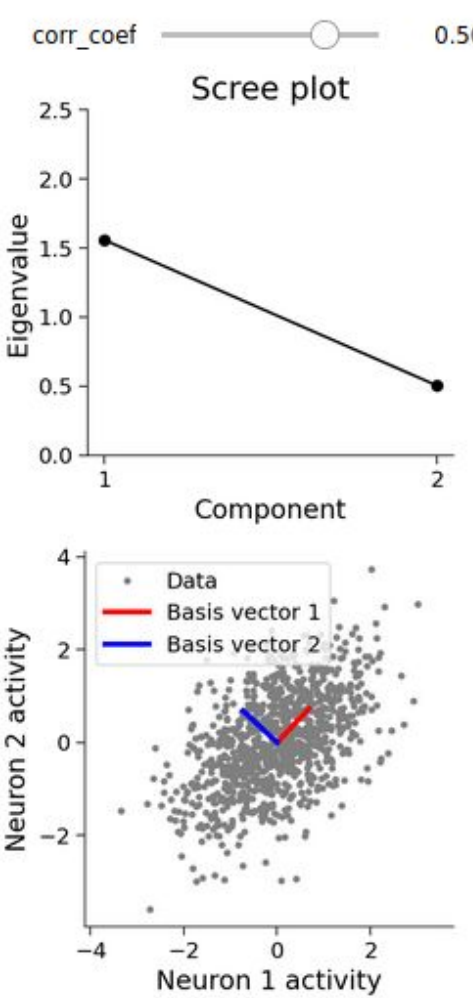
Scree plot

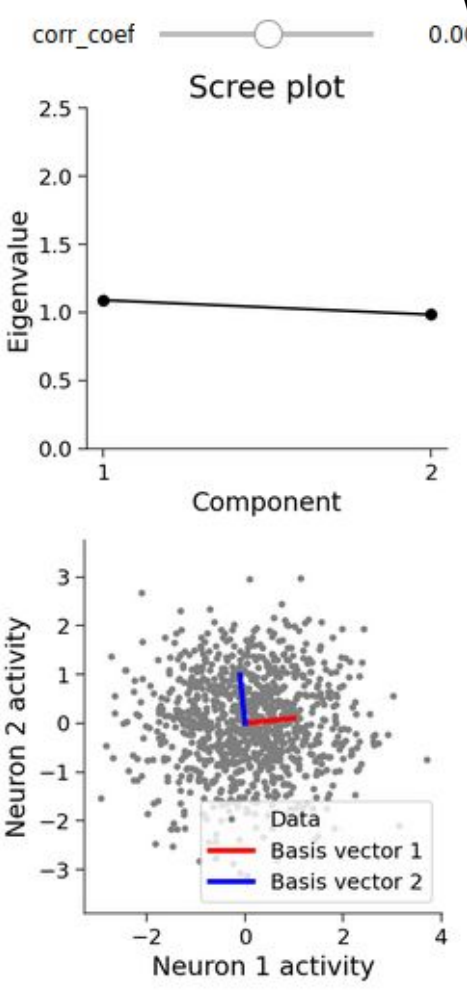


What happens to the eigenvalues as you change the correlation coefficient?

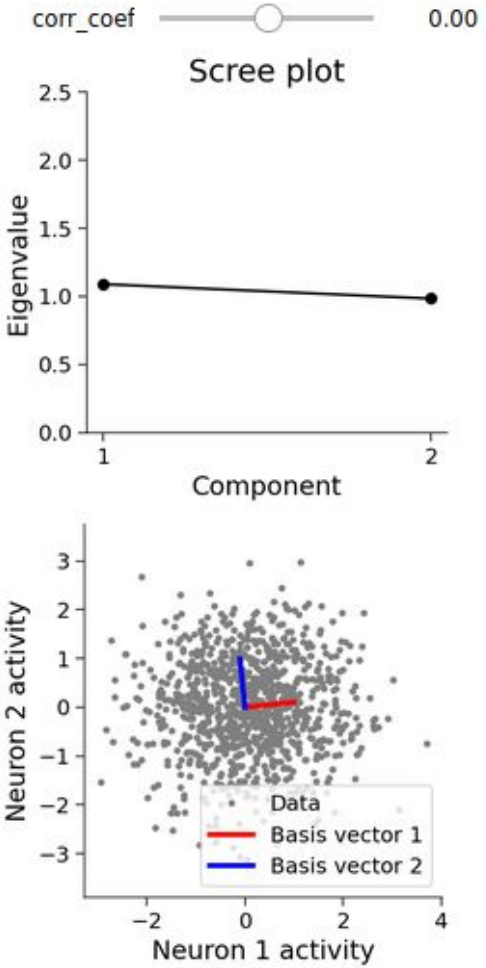



corr_coef
-0.50
Scree plot
Eigenvalue
Component
Data
Basis vector 1
Basis vector 2
Neuron 2 activity
Neuron 1 activity

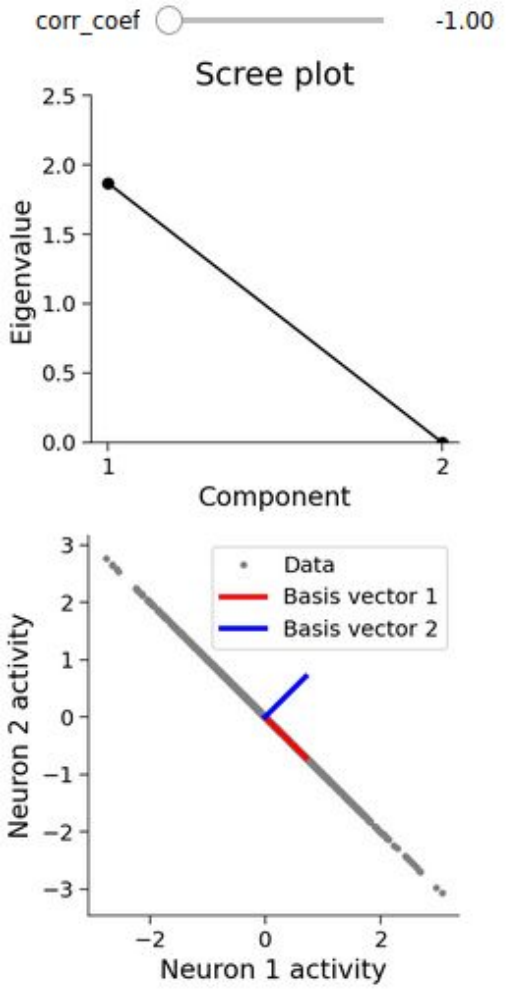

corr_coef
0.50
Scree plot
Eigenvalue
Component
Data
Basis vector 1
Basis vector 2
Neuron 2 activity
Neuron 1 activity

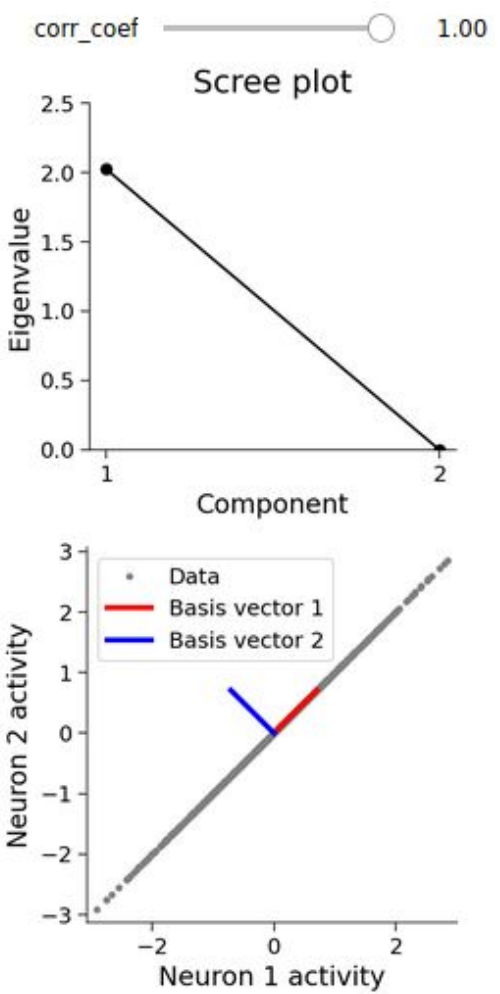

corr_coef
0.00
Scree plot
Eigenvalue
Component
Data
Basis vector 1
Basis vector 2
Neuron 2 activity
Neuron 1 activity

Can you find a value for which both eigenvalues are equal?



Can you find a value for which only one eigenvalue is nonzero?





- Goal of PCA is to find an orthonormal basis capturing the directions of maximum variance of the data. More precisely, the ith basis vector is the direction that maximizes the projected variance, while being orthogonal to all previous basis vectors. Mathematically, these basis vectors are the eigenvectors of the covariance matrix (also called loadings).
- PCA also has the useful property that the projected data (scores) are uncorrelated.
- The projected variance along each basis vector is given by its corresponding eigenvalue. This is important because it allows us rank the "importance" of each basis vector in terms of how much of the data variability it explains. An eigenvalue of zero means there is no variation along that direction so it can be dropped without losing any information about the original data. This property is used to reduce the dimensionality of high-dimensional data.

Organizing information in principal components this way, will allow you to reduce dimensionality without losing much information, and this by discarding the components with low information and considering the remaining components as your new variables.

Tradeoffs: Principal components are less interpretable.

PCA Algorithm

Algorithm

① Standardisation
(so all variables contribute equally)

$$z = \frac{\text{value} - \text{mean}}{\text{std dev}}$$

② variance/covariance

$$\Sigma = \begin{pmatrix} \text{var}(a) & \text{cov}(a,b) \\ \text{cov}(a,b) & \text{var}(b) \end{pmatrix}$$

to understand how variables are varying with each other

highly correlated information ⇒ probably redundant

③ eigenvectors & eigen values

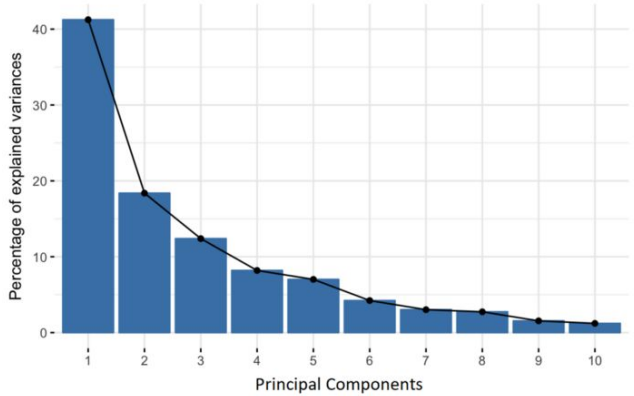
(principal components of the data)

↳ new variables constructed as combinations

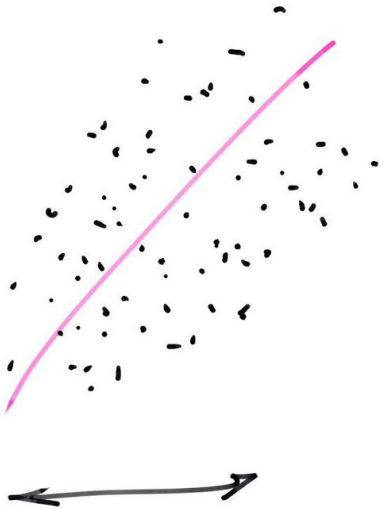
mixtures of initial variables

(uncorrelated)

So, the idea is 10-dimensional data gives you 10 principal components, but PCA tries to put maximum possible information in the first component, then maximum remaining information in the second and so on, until having something like shown in the scree plot below.

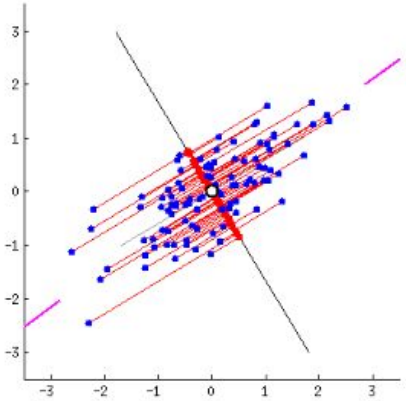


Geometrically speaking, principal components represent the directions of the data that explain a **maximal amount of variance**, that is to say, the lines that capture most information of the data. The relationship between variance and information here, is that, the larger the variance carried by a line, the larger the dispersion of the data points along it, and the larger the dispersion along a line, the more the information it has.



larger variance
larger dispersion
more information
eigen vector
(direction of axes where there's most variance)
eigen value
(coefficient of eigen vector)

As there are as many principal components as there are variables in the data, principal components are constructed in such a manner that the first principal component accounts for the **largest possible variance** in the data set. For example, let's assume that the scatter plot of our data set is as shown below, can we guess the first principal component ? Yes, it's approximately the line that matches the purple marks because it goes through the origin and it's the line in which the projection of the points (red dots) is the most spread out. Or mathematically speaking, it's the line that maximizes the variance (the average of the squared distances from the projected points (red dots) to the origin).



The second principal component is calculated in the same way, with the condition that it is uncorrelated with (i.e., perpendicular to) the first principal component and that it accounts for the next highest variance.

Choose whether to keep all these components or discard those of lesser significance (of low eigenvalues), and form with the remaining ones a matrix of vectors that we call *Feature vector*.

So, the feature vector is simply a matrix that has as columns the eigenvectors of the components that we decide to keep. This makes it the first step towards dimensionality reduction, because if we choose to keep only *p* eigenvectors (components) out of *n*, the final data set will have only *p* dimensions.

look for an "elbow" in the scree plot, to determine which eigenvalues are signficant.

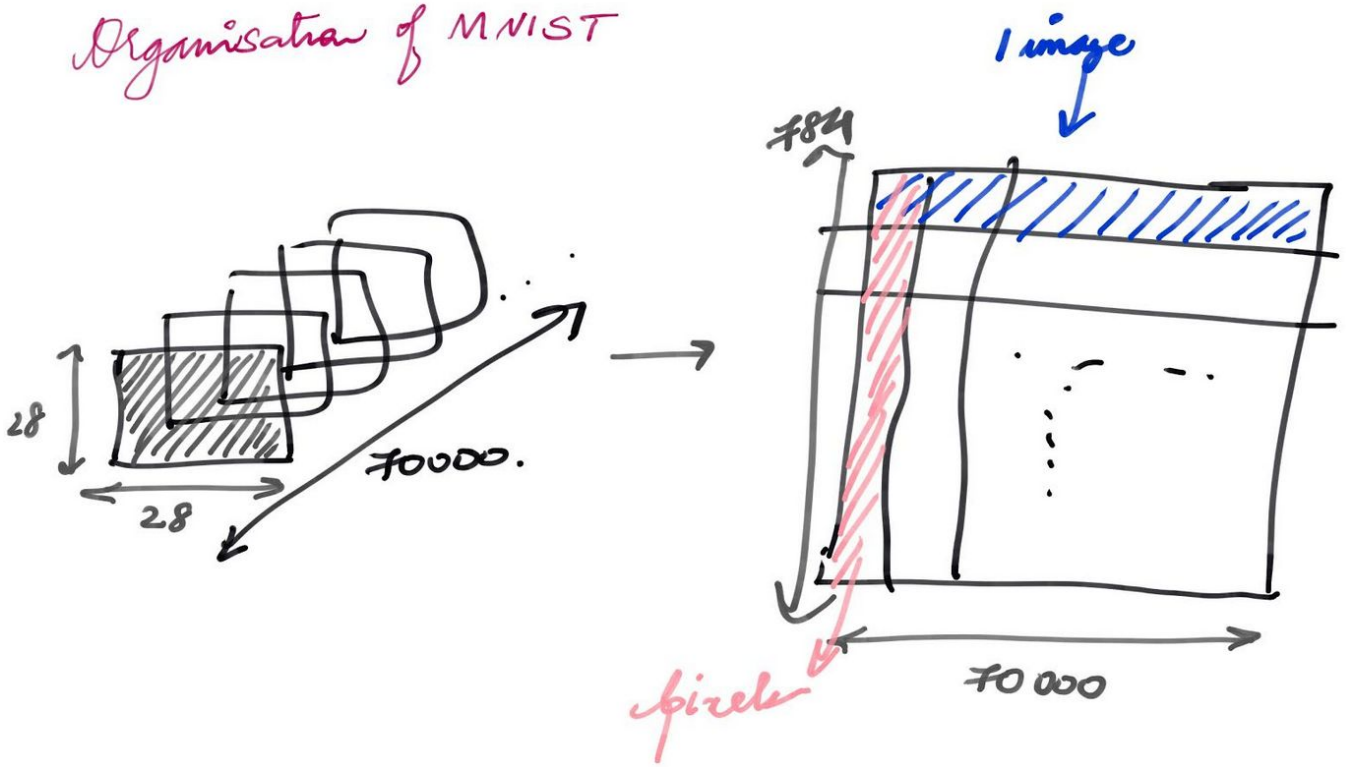
In this step, which is the last one, the aim is to use the feature vector formed using the eigenvectors of the covariance matrix, to reorient the data from the original axes to the ones represented by the principal components (hence the name Principal Components Analysis). This can be done by multiplying the transpose of the original data set by the transpose of the feature vector.

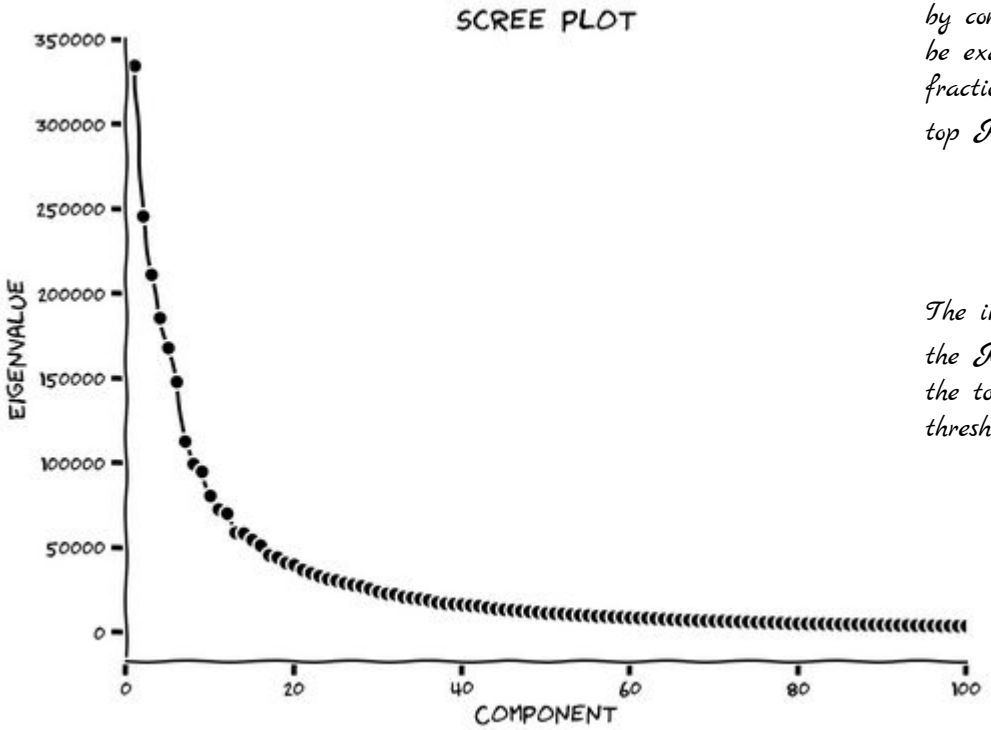
$$FinalDataSet = FeatureVector^T * StandardizedOriginalDataSet^T$$

* * *

Tutorial #3
Explanations

PCA on MNIST



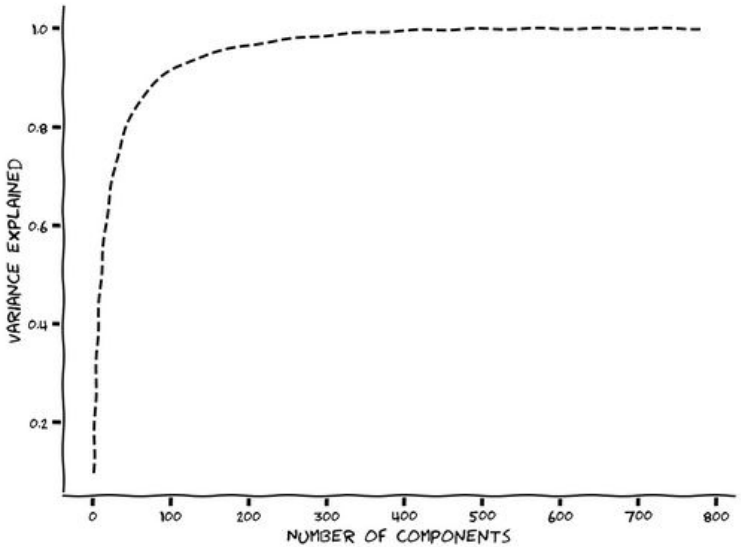


most of the eigenvalues are near zero, with fewer than 100 having large values. Another common way to determine the intrinsic dimensionality is by considering the variance explained. This can be examined with a cumulative plot of the fraction of the total variance explained by the top K components, i.e.,

$$\text{var explained} = \frac{\sum_{i=1}^{K} \lambda_i}{\sum_{i=1}^{N} \lambda_i}$$

The intrinsic dimensionality is often quantified by the K necessary to explain a large proportion of the total variance of the data (often a defined threshold, e.g., 90%).

How many principal components are required to explain 90% of the variance? K which from the elbow of the graph is ~100
How does the intrinsic dimensionality of this dataset compare to its extrinsic dimensionality? 700 to 100 that's a 7x improvement



We can use this fact to perform dimensionality reduction, i.e., by storing the data using only 100 components rather than the samples of all 784 pixels. Remarkably, we will be able to reconstruct much of the structure of the data using only the top 100 components.

PCA

$$S = XW$$

(project X onto eigenvectors of covariance matrix)

$$S = XW^T$$

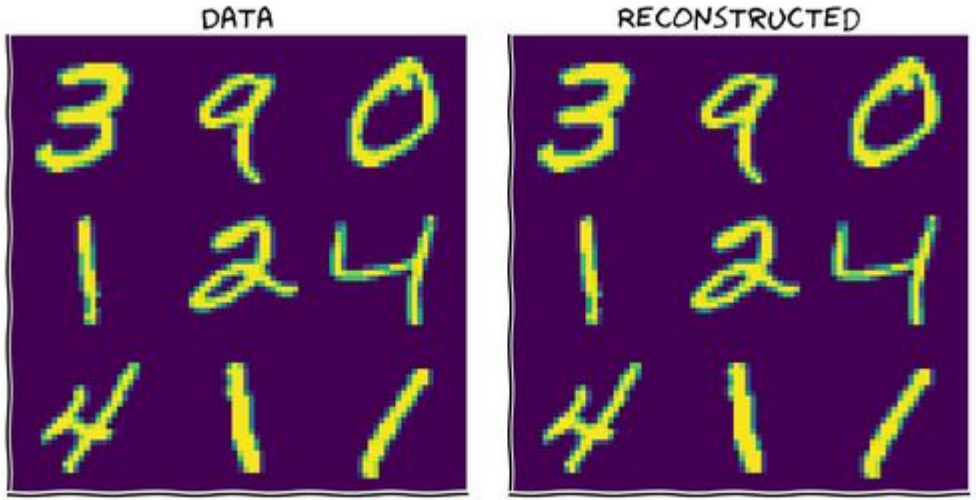
since $W^T = W^{-1}$ (orthogonal property)

Multiplying W^T on each side

$$X = SW^T$$

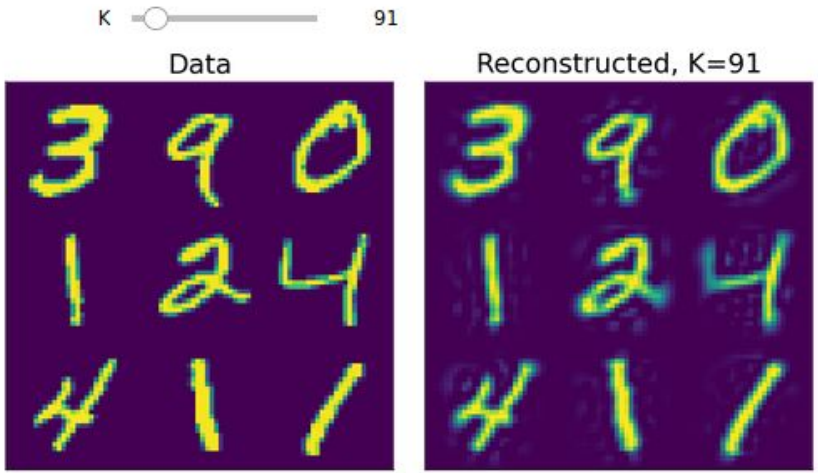
$$\hat{X} = S_{1:K} \left(W_{1:K} \right)^T$$ } k columns only!

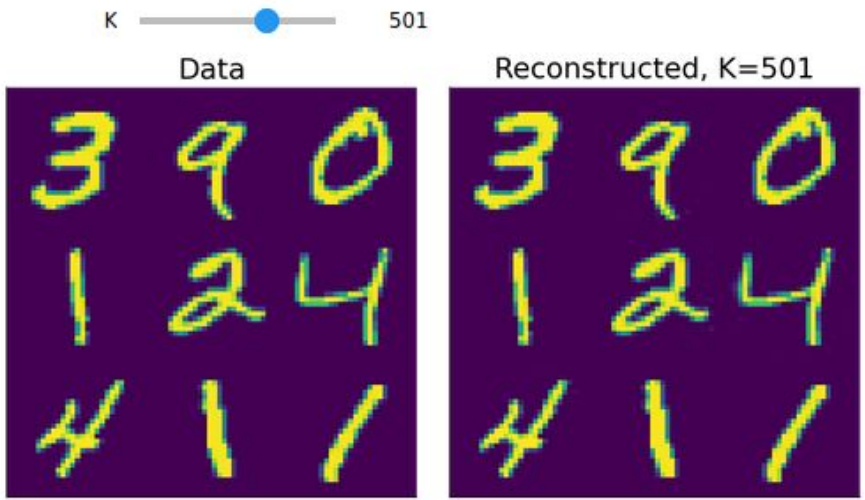
Reconstructed data vs Original data



How many principal components are necessary to reconstruct the numbers (by eye)? How does this relate to the intrinsic dimensionality of the data?

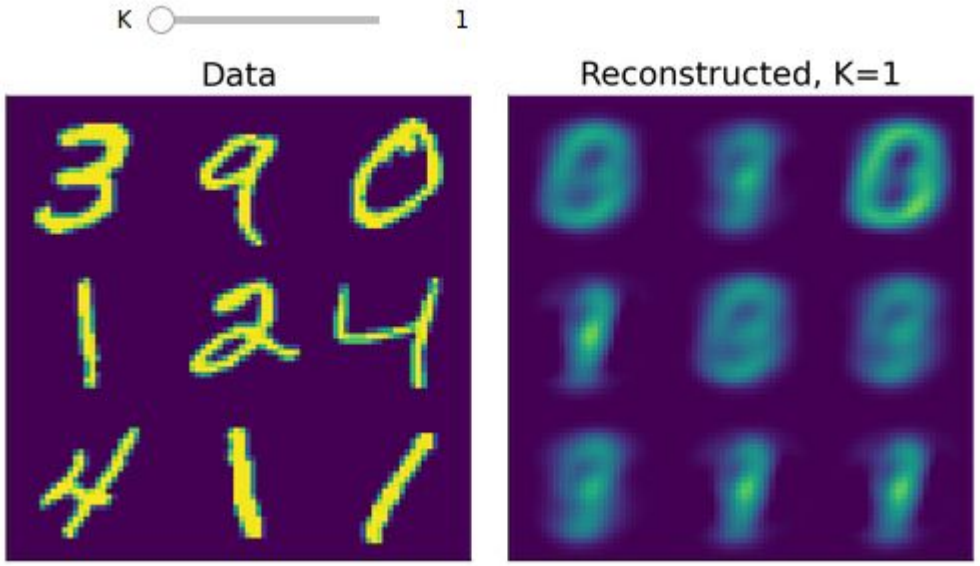
Nothing less than K=100 works and >100 its all the same





Do you see any information in the data with only a single principal component?

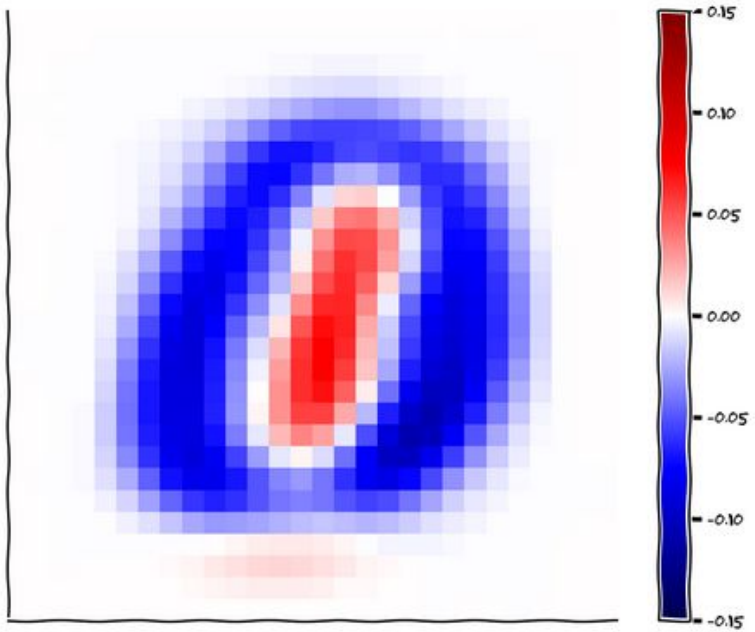
No, too blur!



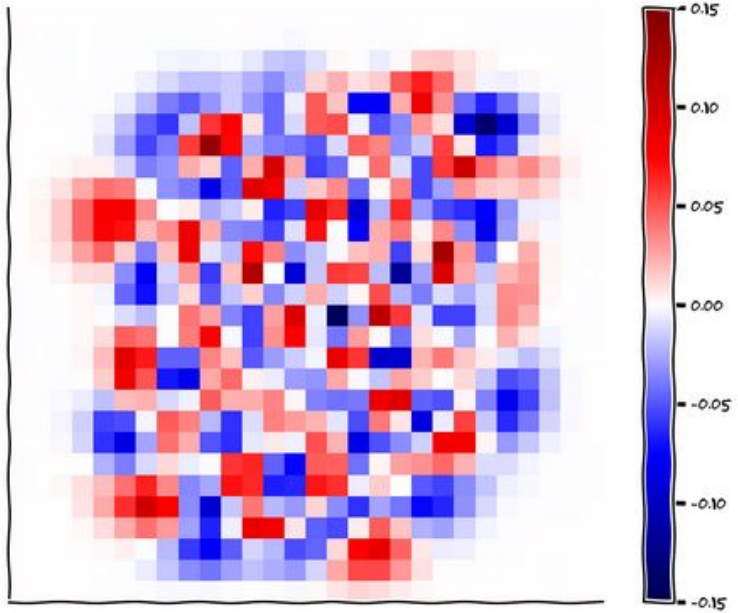
What structure do you see? Which pixels have a strong positive weighting? Which have a strong negative weighting? What kinds of images would this basis vector differentiate?

As the basis vector value increases, noise levels increase. It's just optimal around K. As the principal component number increases, distribution of weight changes and there seem to be more zero weights in the 500-700 basis vectors as opposed to 1 or 100.

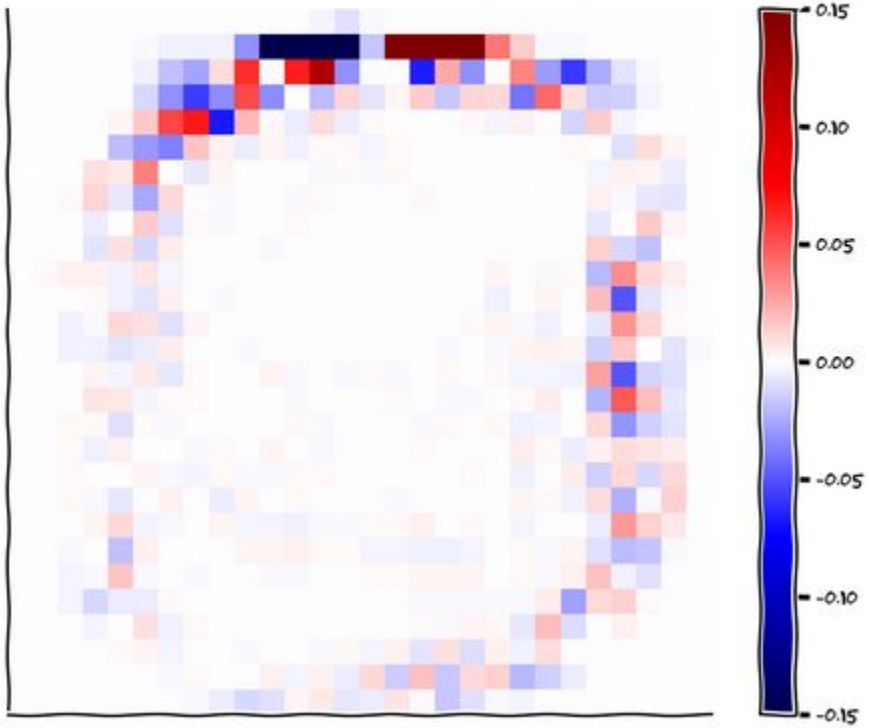
```
# to_remove solution
# Plot the weights of the first principal component
with plt.xkcd():
  plot_MNIST_weights(evectors[:, 0])
```



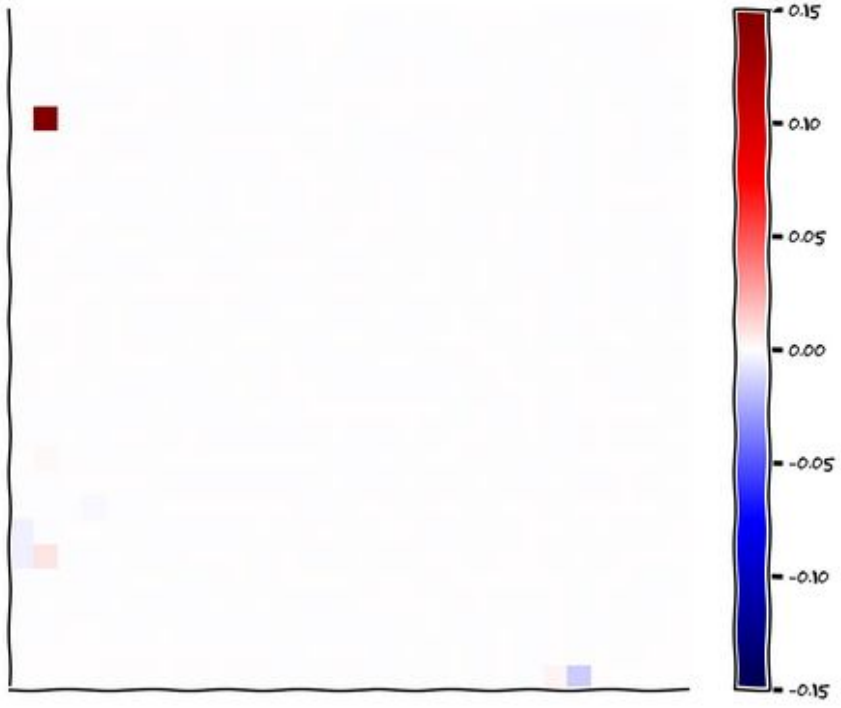

```
# to_remove solution
# Plot the weights of the first principal component
with plt.xkcd():
  plot_MNIST_weights(evectors[:, 100])
```



```
# to_remove solution
# Plot the weights of the first principal component
with plt.xkcd():
  plot_MNIST_weights(evectors[:, 500])
```



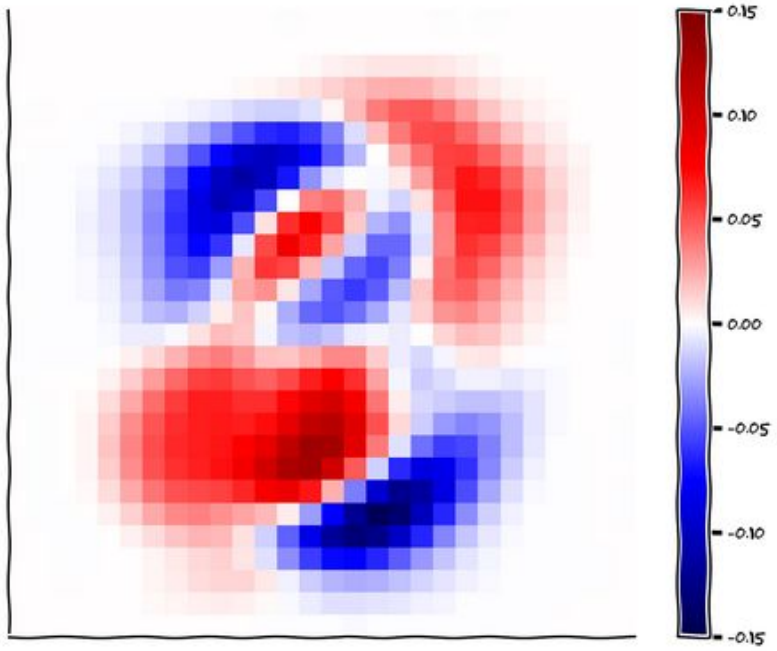

```
# to_remove solution
# Plot the weights of the first principal component
with plt.xkcd():
  plot_MNIST_weights(evectors[:, 700])
```



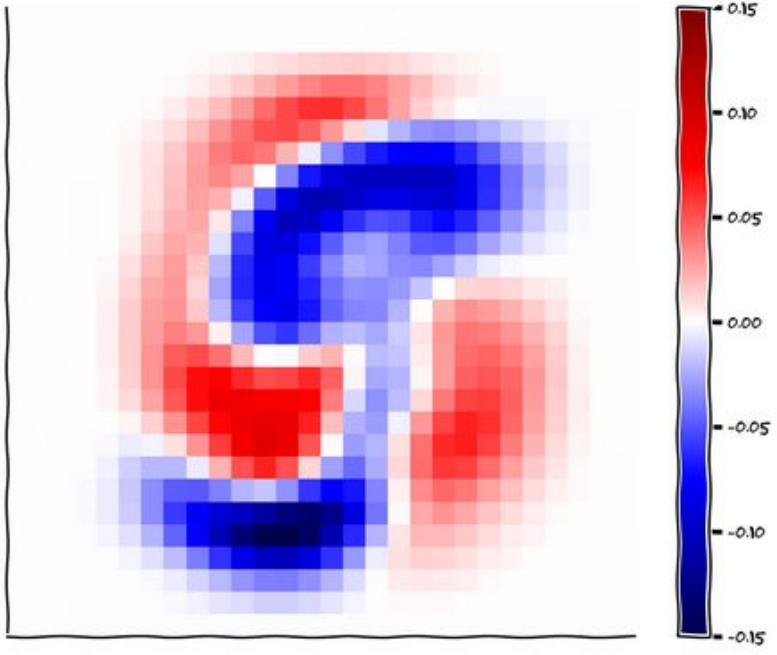
Try visualizing the second and third basis vectors. Do you see any structure? What about the 100th basis vector? 500th? 700th?

Continuity of distribution!

```
# to_remove solution
# Plot the weights of the first principal component
with plt.xkcd():
  plot_MNIST_weights(evectors[:, 2])
```




```
# to_remove solution
# Plot the weights of the first principal component
with plt.xkcd():
  plot_MNIST_weights(evectors[:, 3])
```



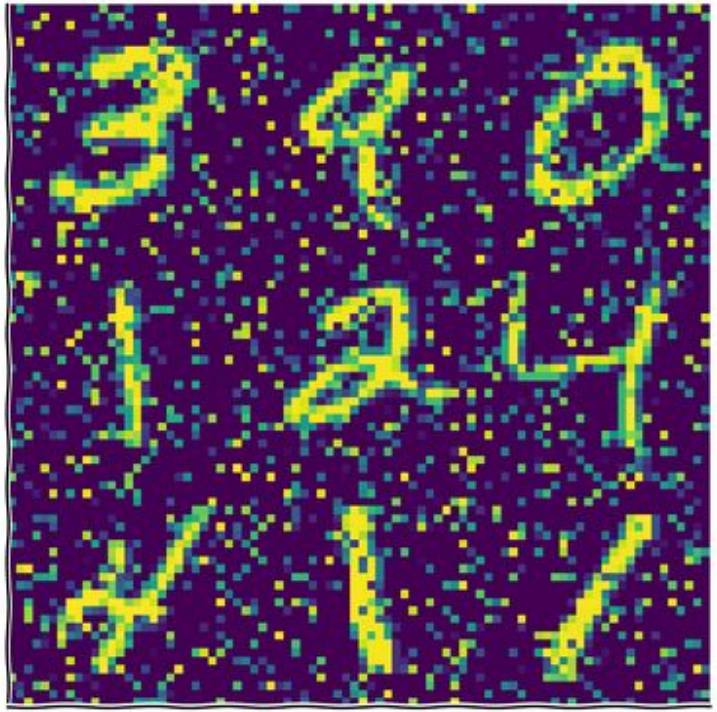
Summary

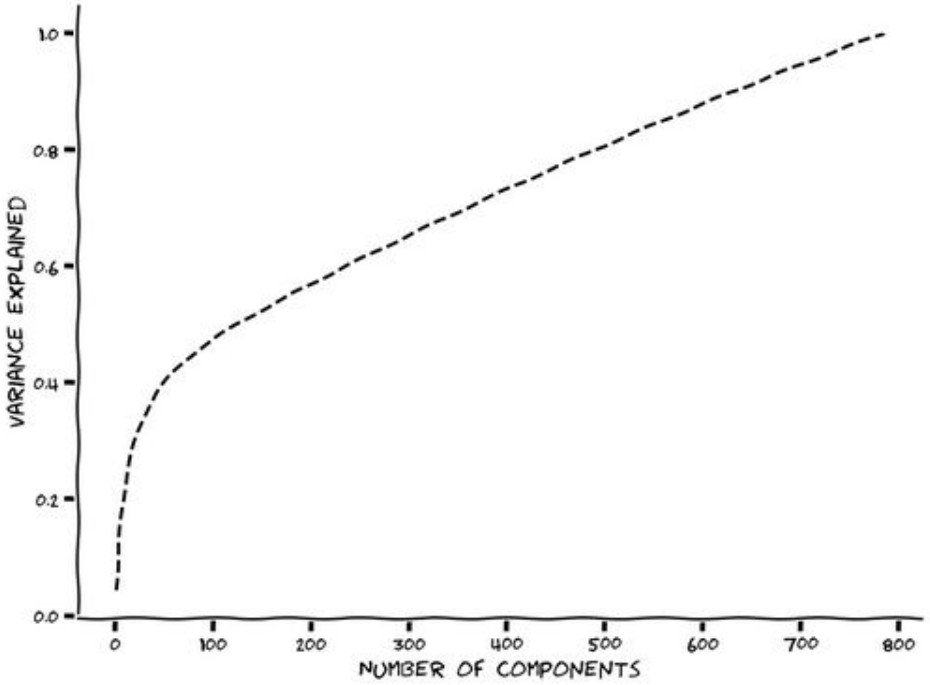
- PCA for dimensionality reduction by selecting the top principal components: useful as the intrinsic dimensionality (K) is often less than the extrinsic dimensionality (N) in neural data.
- K can be inferred by choosing the number of eigenvalues necessary to capture some fraction of the variance.
- reconstruct an approximation of the original data using the top K principal components. In fact, an alternate formulation of PCA is to find the K dimensional space that minimizes the reconstruction error.
- Noise tends to inflate the apparent intrinsic dimensionality, however the higher components reflect noise rather than new structure in the data. PCA can be used for denoising data by removing noisy higher components.
- In MNIST, the weights corresponding to the first principal component appear to discriminate between a 0 and 1.

Tutorial #3 Bonus Explanations

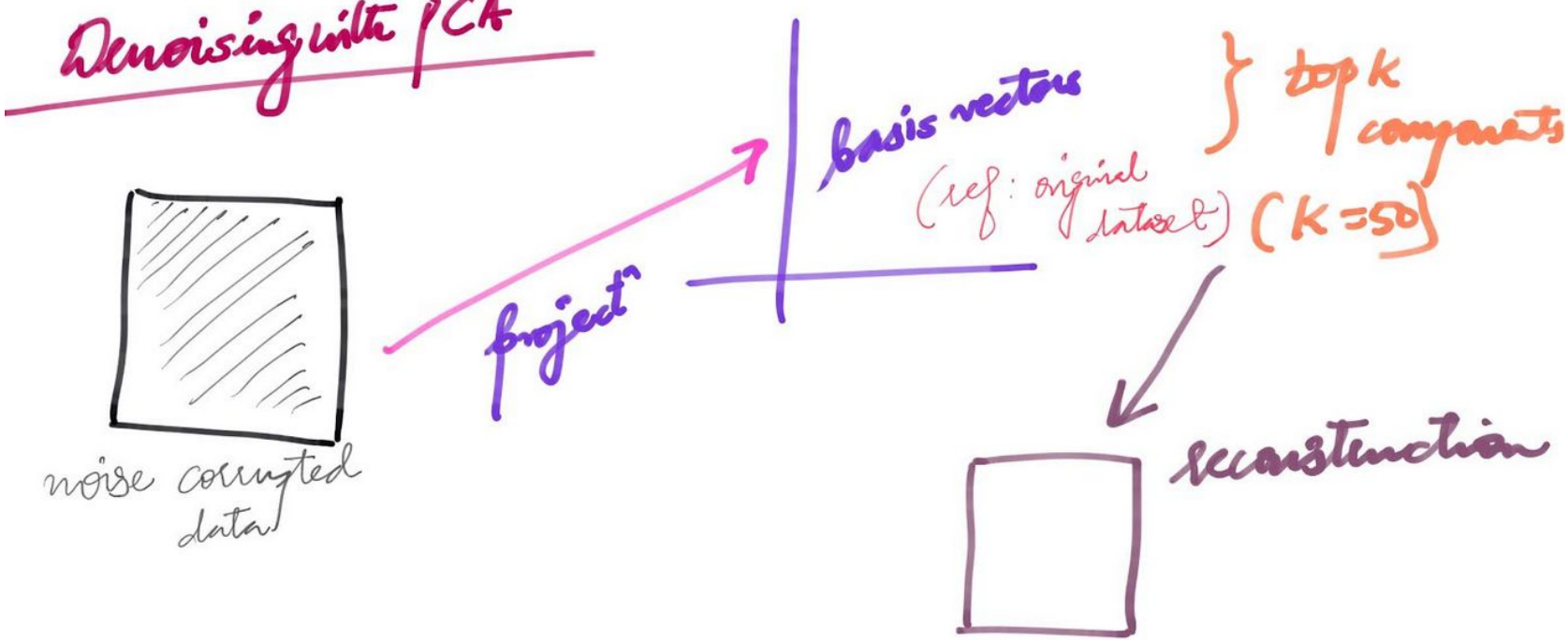
Denoising using PCA

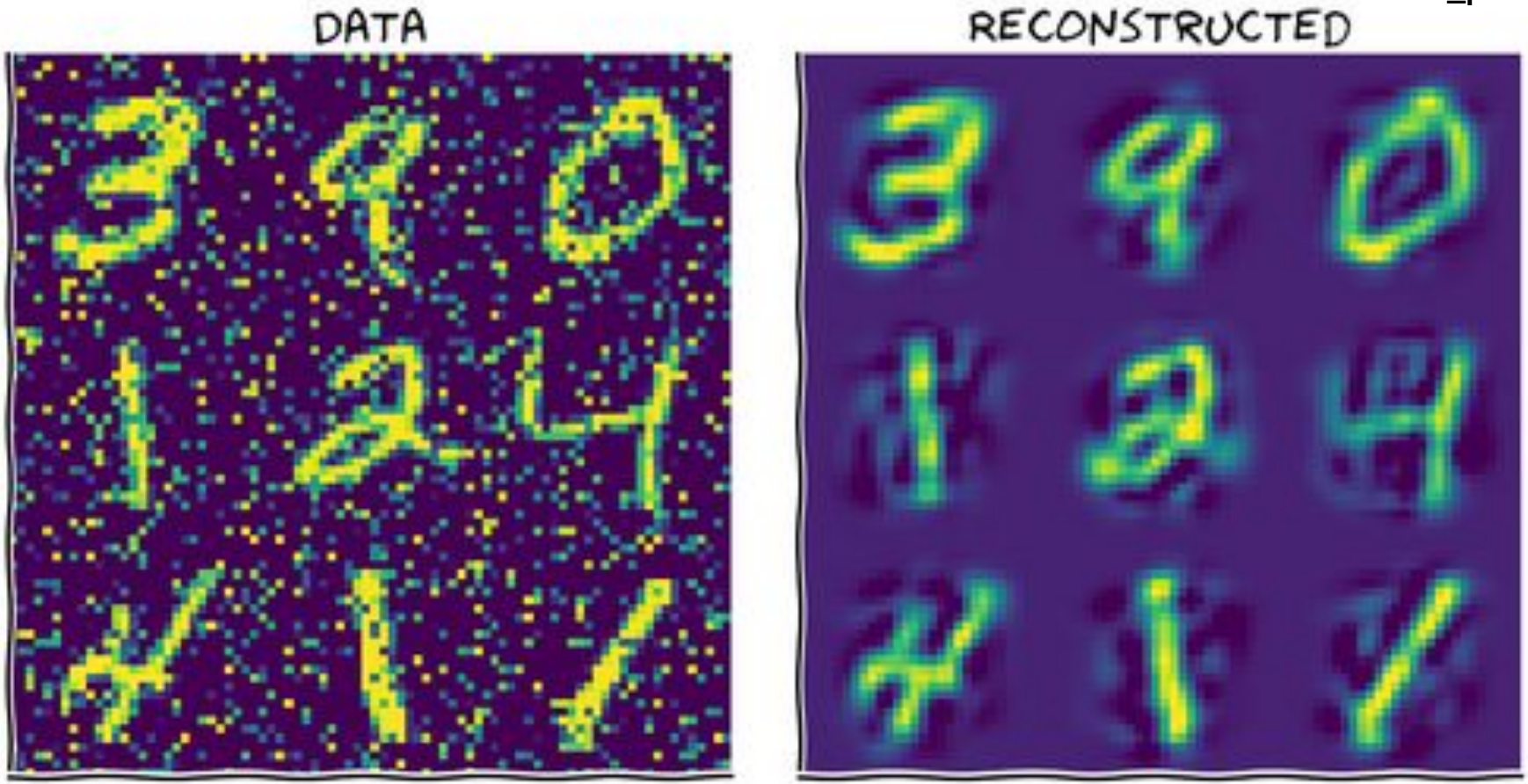
- *PCA finds an optimal low-dimensional basis to minimize the reconstruction error. Because of this property, PCA can be useful for denoising corrupted samples of the data.*
- *add salt-and-pepper noise to the original data and see how that affects the eigenvalues.*





PCA to perform denoising by projecting the noise-corrupted data onto the basis vectors found from the original dataset. By taking the top K components of this projection, we can reduce noise in dimensions orthogonal to the K-dimensional latent space.




DATA
RECONSTRUCTED

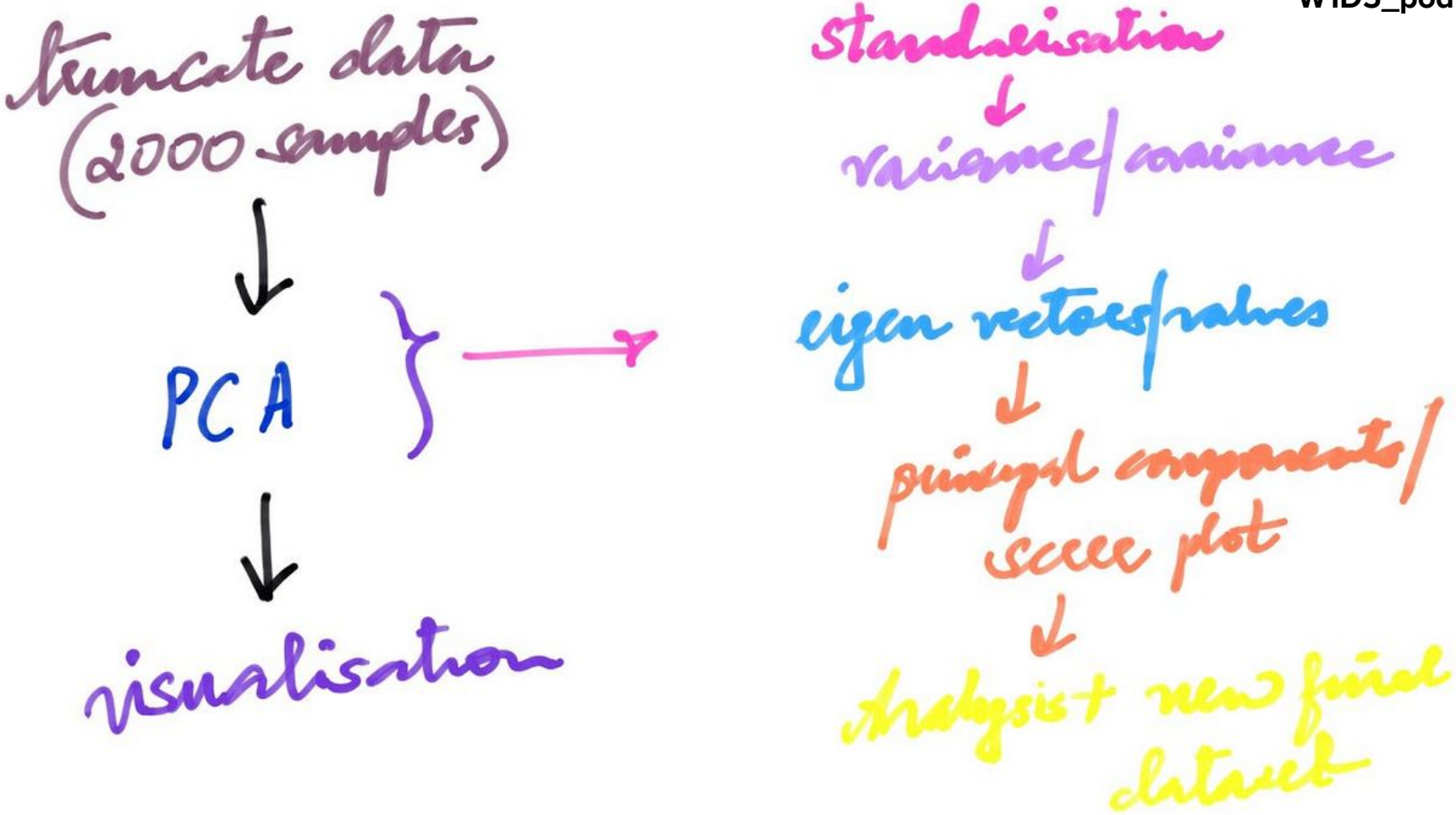
Tutorial #4
Explanations

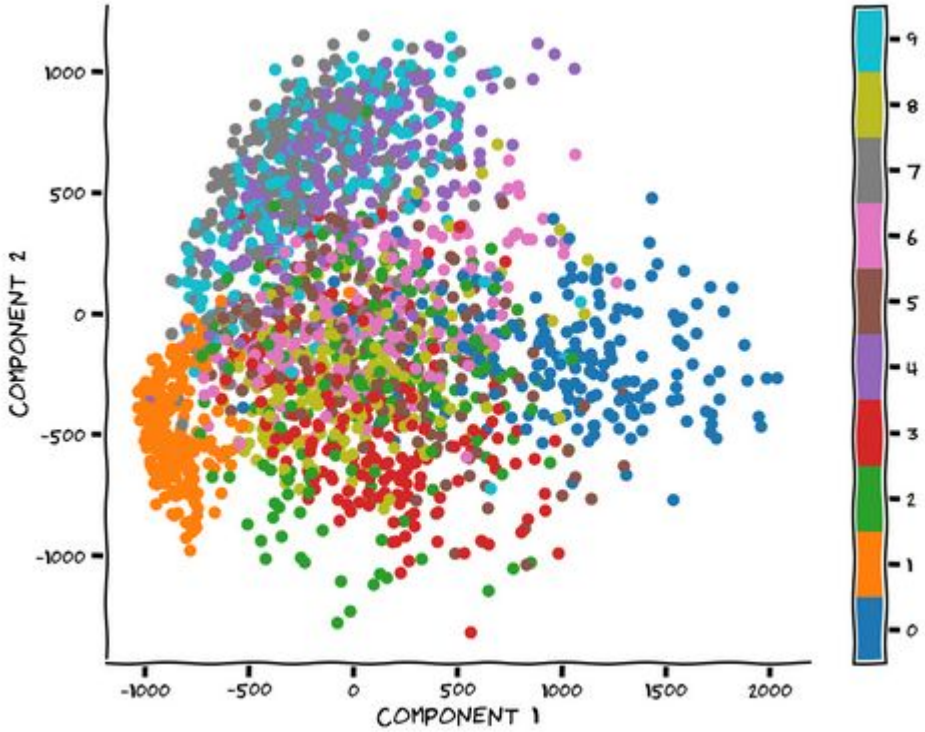
Explore how dimensionality reduction can be useful for visualizing and inferring structure in your data comparing PCA with t-SNE, a nonlinear dimensionality reduction method.

2D => 2 component PCA

```
PCA(COPY=TRUE, ITERATED_POWER='AUTO',
N_COMPONENTS=2, RANDOM_STATE=NONE,
SVD_SOLVER='AUTO', TOL=0.0, WHITEN=FALSE)
```

For better visualization, take only the first 2,000 samples of the data


truncate data
(2000 samples)
PCA
visualisation
standardisation
variance/covariance
eigen vectors/values
principal components/
scree plot
analysis + new final
dataset



Are different samples corresponding to the same numeral clustered together? Is there much overlap?

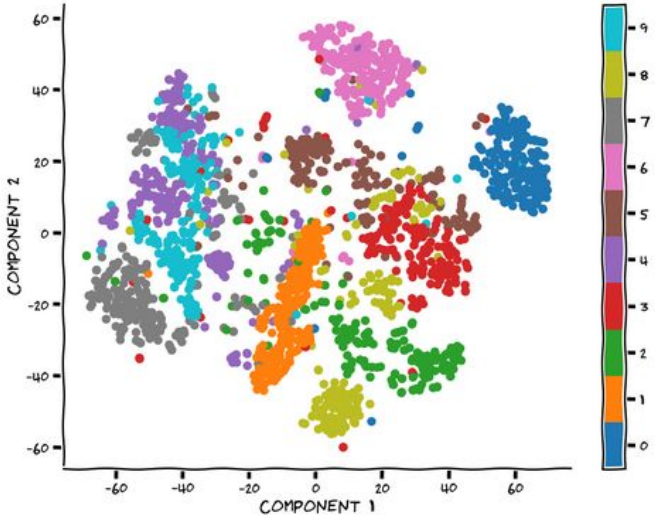
There's overlap between numerals that are alike. For instance, 7 and 9, 6 and 8;

Do some pairs of numerals appear to be more distinguishable than others?

1, 0, 2 appear more distinguishable than say 9, 7, 8 etc.

T-SNE (T-DISTRIBUTED STOCHASTIC NEIGHBOR EMBEDDING)

A tool that converts similarities between data points to joint probabilities and tries to minimize the Kullback-Leibler divergence between the joint probabilities of the low-dimensional embedding and the high-dimensional data. t-SNE has a cost function that is not convex, i.e. with different initializations we can get different results.



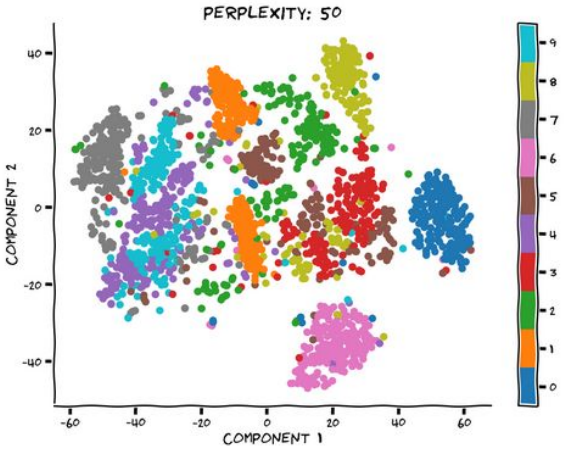

```
tsne_model = TSNE(n_components=2, perplexity=30, random_state=2020)
```

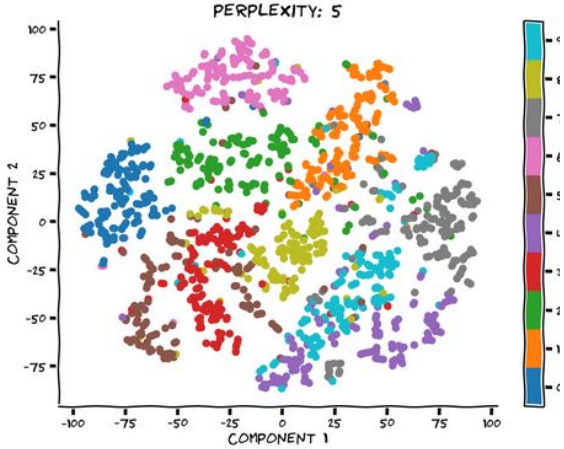
Explored for more structure.

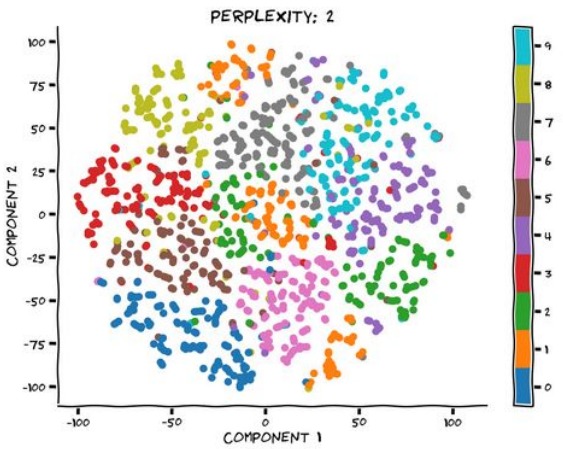
Can be used to find our embedding (i.e. the low-dimensional representation of the data) and stored them in model

Perplexity

t-SNE has a free parameter (the perplexity) that roughly determines how global vs. local information is weighted.







What changes compared to your previous results using perplexity equal to 50? Do you see any clusters that have a different structure than before?

Location and structure of cluster changes.

What changes in the embedding structure for perplexity equals to 5 or 2?

Overall cluster structure changes (more distributed at perplexity=2)

PcA vs t-sne

- difference between linear and nonlinear dimensionality reduction. While nonlinear methods can be more powerful, they can also be sensitive to noise. In contrast, linear methods are useful for their simplicity and robustness.
- Using t-SNE, we could visualize clusters in the data corresponding to different digits. While PCA was able to separate some clusters (e.g., 0 vs 1), it performed poorly overall.
- However, the results of t-SNE can change depending on the choice of perplexity.

References:

Computational Neuroscience by University of Washington
https://www.coursera.org/lecture/computational-neuroscience/change-of-basis-and-pca-by-rich-pang-57AT2

PCA
https://builtin.com/data-science/step-step-explanation-principal-component-analysis

T-SNE: Distill paper.