GENERAL ASSEMBLY

INTRO TO DATA SCIENCE

DIMENSIONALITY REDUCTION

I. DIMENSIONALITY REDUCTION
II. PRINCIPAL COMPONENTS ANALYSIS
III. SINGULAR VALUE DECOMPOSITION
IV. OTHER METHODS

EXERCISE:
IV. DIMENSIONALITY REDUCTION IN SCIKIT-LEARN

I. DIMENSIONALITY REDUCTION

Q: What is dimensionality reduction?

Q: What is dimensionality reduction?

A: A set of techniques for reducing the size (in terms of features, records, and/or bytes) of the dataset under examination.

Q: What is dimensionality reduction?

A: A set of techniques for reducing the size (in terms of features, records, and/or bytes) of the dataset under examination.

In general, the idea is to regard the dataset as a matrix and to decompose the matrix into simpler, meaningful pieces.

Q: What is dimensionality reduction?

A: A set of techniques for reducing the size (in terms of features, records, and/or bytes) of the dataset under examination.

In general, the idea is to regard the dataset as a matrix and to decompose the matrix into simpler, meaningful pieces.

Dimensionality reduction is frequently performed as a pre-processing step before another learning algorithm is applied.

Q: What are the motivations for dimensionality reduction?

Q: What are the motivations for dimensionality reduction?

The number of features in our dataset can be difficult to manage, or even misleading (eg, if the relationships are actually simpler than they appear).

For example, suppose we have a dataset with some features that are related to each other.

For example, suppose we have a dataset with some features that are related to each other.

Ideally, we would like to eliminate this redundancy and consolidate the number of variables we're looking at.

For example, suppose we have a dataset with some features that are related to each other.

Ideally, we would like to eliminate this redundancy and consolidate the number of variables we're looking at.

If these relationships are *linear*, then we can use well-established techniques like PCA/SVD.

EXAMPLE: 1D HARMONIC OSCILLATOR

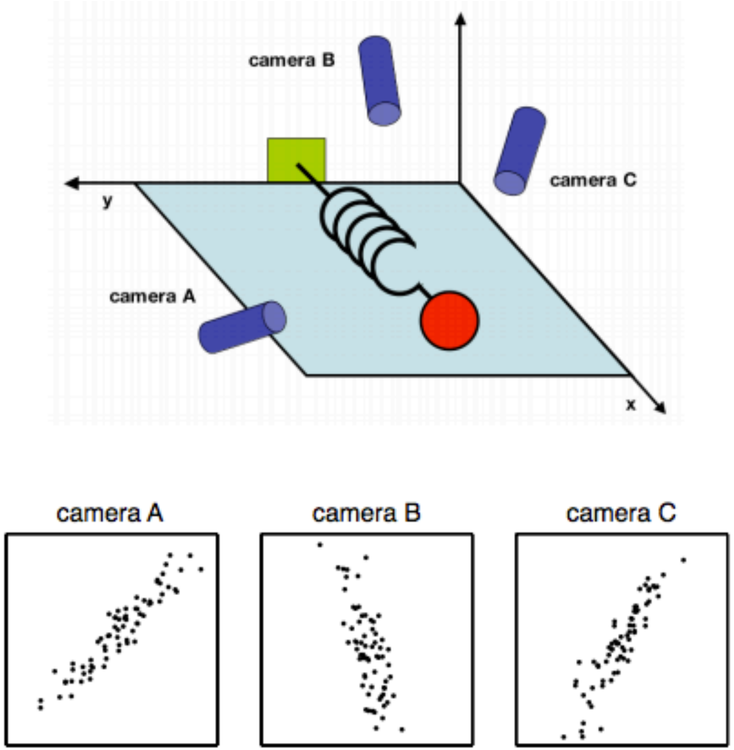


FIG. 1 A toy example. The position of a ball attached to an oscillating spring is recorded using three cameras A, B and C. The position of the ball tracked by each camera is depicted in each panel below.

source: http://www.snl.salk.edu/~shlens/pca.pdf

EXAMPLE: 1D HARMONIC OSCILLATOR

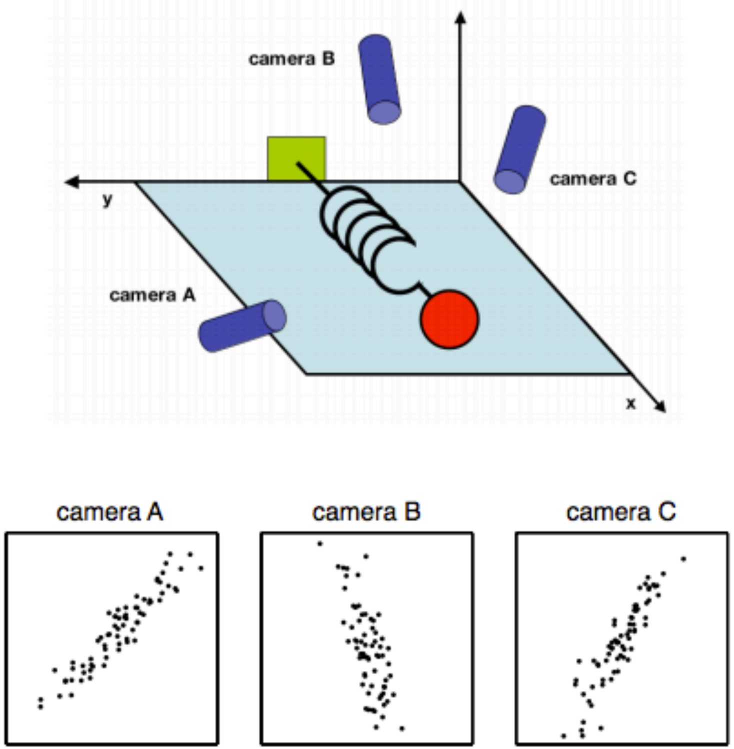


FIG. 1 A toy example. The position of a ball attached to an oscillating spring is recorded using three cameras A, B and C. The position of the ball tracked by each camera is depicted in each panel below.

source: http://www.snl.salk.edu/~shlens/pca.pdf

NOTE

In this case the “truth” is (nearly) one-dimensional. We don’t generally know what the “truth” is, but the same techniques can apply.

The complexity that comes with a large number of features is due in part to the **curse of dimensionality**.

The complexity that comes with a large number of features is due in part to the **curse of dimensionality**.

Namely, the sample size needed to accurately estimate a random variable taking values in a d-dimensional feature space grows exponentially with d (almost).

(More precisely, the sample size grows exponentially with $l \leq d$, the dimension of the manifold *embedded* in the feature space).

Another way of characterizing this is to say that high-dimensional spaces are inherently **sparse**.

Another way of characterizing this is to say that high-dimensional spaces are inherently **sparse**.

ex: A high-dimensional orange contains most of its volume in the rind!

ex: A high-dimensional hypercube contains most of its volume in the corners!

In either case, most of the points in the space are “far” from the center.

In either case, most of the points in the space are “far” from the center.

This illustrates the fact that local methods will break down in these circumstances (eg, in order to collect enough neighbors for a given point, you need to expand the radius of the neighborhood so far that locality is not preserved).

In either case, most of the points in the space are “far” from the center.

This illustrates the fact that local methods will break down in these circumstances (eg, in order to collect enough neighbors for a given point, you need to expand the radius of the neighborhood so far that locality is not preserved).

The bottom line is that high-dimensional spaces can be problematic.

Q: What is the goal of dimensionality reduction?

Q: What is the goal of dimensionality reduction?

We'd like to analyze the data using the most meaningful basis (or **coordinates**) possible.

Q: What is the goal of dimensionality reduction?

We'd like to analyze the data using the most meaningful basis (or **coordinates**) possible.

More precisely: given an n x d matrix X (encoding n observations of a d-dimensional random variable), we want to find a k-dimensional representation of X ($k < d$) that captures the information in the original data, according to some criterion.

Q: What is the goal of dimensionality reduction?

- reduce computational expense
- reduce susceptibility to overfitting
- reduce noise in the dataset
- enhance our intuition

Q: How is dimensionality reduction performed?

Q: How is dimensionality reduction performed?

A: There are two approaches: feature selection and feature extraction.

Q: How is dimensionality reduction performed?
A: There are two approaches: feature selection and feature extraction.

feature selection – selecting a subset of features using an external criterion (*filter*) or the learning algo accuracy itself (*wrapper*)

feature extraction – mapping the features to a lower dimensional space

Feature selection is important, but typically when people say dimensionality reduction, they are referring to *feature extraction*.

Feature selection is important, but typically when people say dimensionality reduction, they are referring to *feature extraction*.

The goal of feature extraction is to create a new set of coordinates that *simplify the representation* of the data.

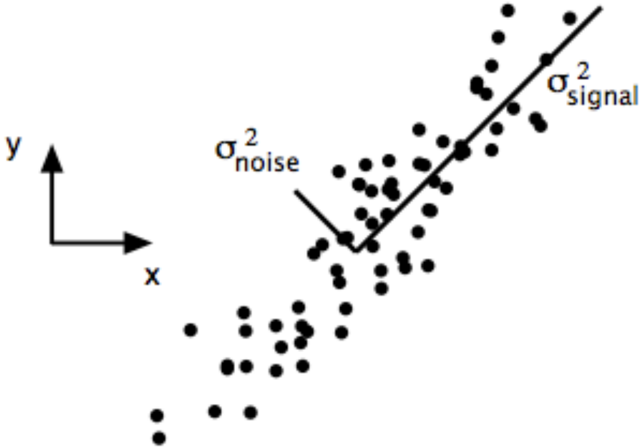


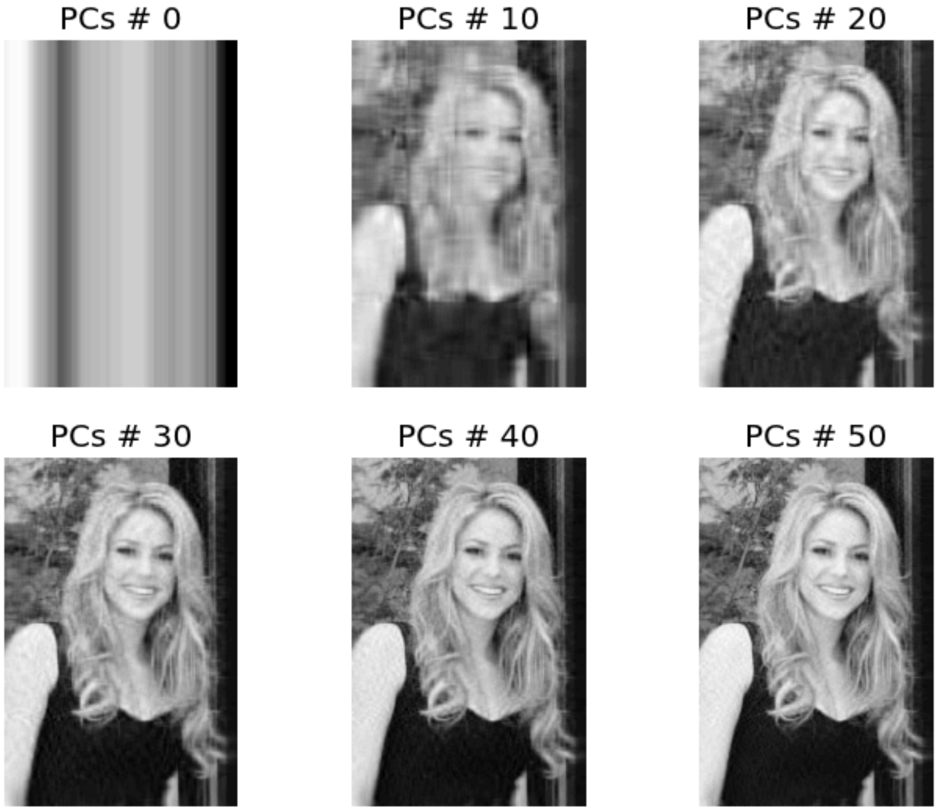
FIG. 2 Simulated data of (x, y) for camera A. The signal and noise variances σ^2_{signal} and σ^2_{noise} are graphically represented by the two lines subtending the cloud of data. Note that the largest direction of variance does not lie along the basis of the recording (x_A, y_A) but rather along the best-fit line.

source: http://www.snl.salk.edu/~shlens/pca.pdf

Q: What are some applications of dimensionality reduction?

Q: What are some applications of dimensionality reduction?

- topic models (document clustering)
- image recognition/computer vision
- bioinformatics (microarray analysis)
- speech recognition
- astronomy (spectral data analysis)
- recommender systems



source: http://glowingpython.blogspot.it/2011/07/pca-and-image-compression-with-numpy.html

II. PRINCIPAL COMPONENT ANALYSIS

Principal component analysis is a dimension reduction technique that can be used on a matrix of any dimensions.

Principal component analysis is a dimension reduction technique that can be used on a matrix of any dimensions.

This procedure produces a new basis, each of whose components retain as much variance from the original data as possible.

Principal component analysis is a dimension reduction technique that can be used on a matrix of any dimensions.

This procedure produces a new basis, each of whose components retain as much variance from the original data as possible.

The PCA of a matrix X boils down to the **eigenvalue decomposition** of the **covariance matrix** of X.

The covariance matrix *C* of a matrix *X* is always square:

$$
\mathrm{C} = \begin{bmatrix}
\mathrm{E}[(X_1-\mu_1)(X_1-\mu_1)] & \mathrm{E}[(X_1-\mu_1)(X_2-\mu_2)] & \cdots & \mathrm{E}[(X_1-\mu_1)(X_n-\mu_n)] \\
\mathrm{E}[(X_2-\mu_2)(X_1-\mu_1)] & \mathrm{E}[(X_2-\mu_2)(X_2-\mu_2)] & \cdots & \mathrm{E}[(X_2-\mu_2)(X_n-\mu_n)] \\
\vdots & \vdots & \ddots & \vdots \\
\mathrm{E}[(X_n-\mu_n)(X_1-\mu_1)] & \mathrm{E}[(X_n-\mu_n)(X_2-\mu_2)] & \cdots & \mathrm{E}[(X_n-\mu_n)(X_n-\mu_n)]
\end{bmatrix}.
$$

off-diagonal elements C_{ij} give the *covariance* between X_i, X_j $(i \neq j)$
diagonal elements C_{ii} give the *variance* of X_i

The *eigenvalue decomposition* of a square matrix C is given by:

$$C = Q \Lambda Q^{-1}$$

The *eigenvalue decomposition* of a square matrix C is given by:

$$C = Q \Lambda Q^{-1}$$

The columns of Q are the **eigenvectors** of C, and the values in Λ are the associated **eigenvalues** of C.

The *eigenvalue decomposition* of a square matrix C is given by:

$$C = Q\Lambda Q^{-1}$$

The columns of Q are the **eigenvectors** of C, and the values in Λ are the associated **eigenvalues** of C.

For an eigenvector v of C and its eigenvalue λ, we have the important relation:

$$Cv = \lambda v$$

The *eigenvalue decomposition* of a square matrix C is given by:

$$C = Q \Lambda Q^{-1}$$

The columns of Q are the **eigenvectors** of C, and the values in Λ are the associated **eigenvalues** of C.

For an eigenvector v of C and its eigenvalue λ, we have the relation:

$$Cv = \lambda v$$

NOTE

This relationship *defines* what it means to be an eigenvector of C.

The eigenvectors form a basis of the vector space on which C acts (eg, they are orthogonal).

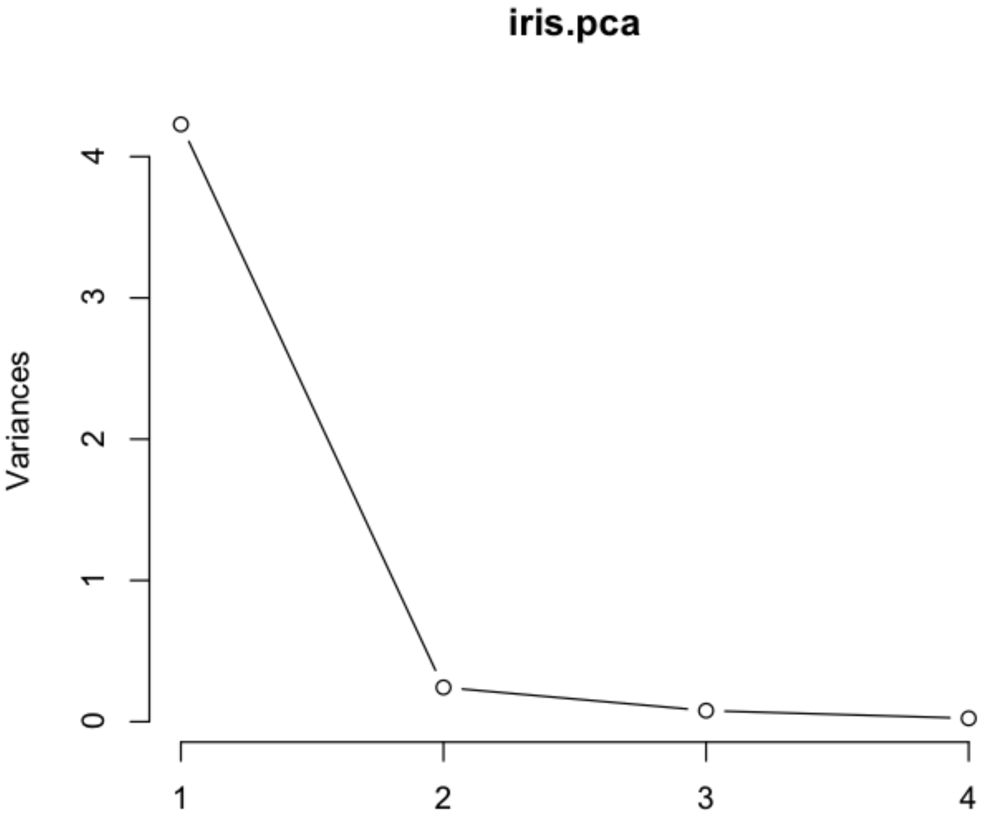
The eigenvectors form a basis of the vector space on which C acts (eg, they are orthogonal).

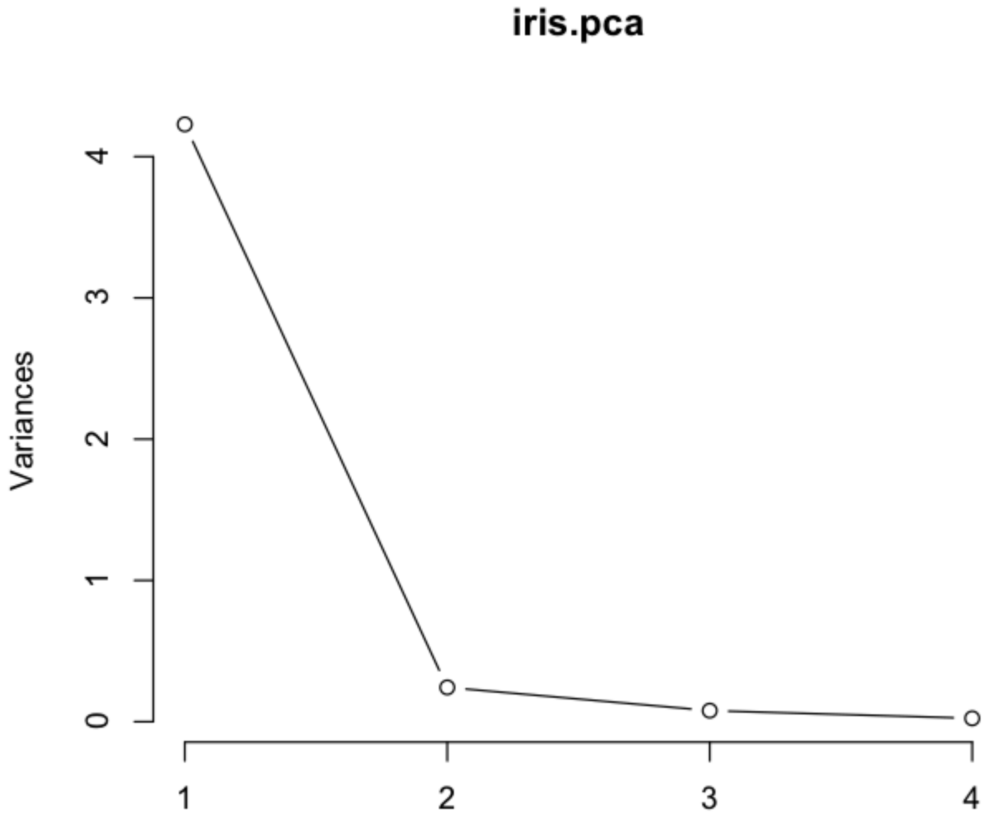
Furthermore the basis elements are ordered by their eigenvalues (from largest to smallest), and these eigenvalues represent the amount of variance explained by each basis element.

The eigenvectors form a basis of the vector space on which C acts (eg, they are orthogonal).

Furthermore the basis elements are ordered by their eigenvalues (from largest to smallest), and these eigenvalues represent the amount of variance explained by each basis element.

This can be visualized in a **scree plot**, which shows the amount of variance explained by each basis vector.


iris.pca
Variances
0
1
2
3
4
1
2
3
4



NOTE

Looking at this plot also gives you an idea of how many principal components to keep.

Apply the *elbow test*: keep only those pc's that appear to the left of the elbow in the graph.

III. SINGULAR VALUE DECOMPOSITION

Consider a matrix X with n rows and d features.

Consider a matrix X with n rows and d features.

The **singular value decomposition** of X is given by:

$$X = U \Sigma V^T$$

Consider a matrix X with n rows and d features.

The **singular value decomposition** of X is given by:

$$\underset{(n \times d)}{X} = \underset{(n \times n)}{U} \; \underset{(n \times d)}{\Sigma} \; \underset{(d \times d)}{V^T}$$

Consider a matrix X with n rows and d features.

The **singular value decomposition** of A is given by:

$$\underset{(n \times d)}{X} = \underset{(n \times n)}{U} \underset{(n \times d)}{\Sigma} \underset{(d \times d)}{V^T}$$

st. U, V are **orthogonal** matrices and Σ is a **diagonal** matrix.

Consider a matrix X with n rows and d features.

The **singular value decomposition** of A is given by:

$$\underset{(n \times d)}{X} = \underset{(n \times n)}{U} \; \underset{(n \times d)}{\Sigma} \; \underset{(d \times d)}{V^T}$$

st. U, V are **orthogonal** matrices and Σ is a **diagonal** matrix.

→ $UU^T = I_n, \; VV^T = I_d$ → $\Sigma_{ij} = 0 \; (i \neq j)$

The **singular value decomposition** of X is given by:

$$\underset{(n \times d)}{X} = \underset{(n \times n)}{U} \; \underset{(n \times d)}{\Sigma} \; \underset{(d \times d)}{V^T}$$

The columns of U & V are the (left- and right-) **singular vectors** of X.

The **singular value decomposition** of X is given by:

$$\underset{(n \times d)}{X} = \underset{(n \times n)}{U} \; \underset{(n \times d)}{\Sigma} \; \underset{(d \times d)}{V^T}$$

The columns of U & V are the (left- and right-) **singular vectors** of X.

These singular vectors provide **orthonormal bases** for the spaces K_n & K_d (columns of U & V, respectively).

The **singular value decomposition** of X is given by:

$$\underset{(n \times d)}{X} = \underset{(n \times n)}{U} \; \underset{(n \times d)}{\Sigma} \; \underset{(d \times d)}{V^T}$$

The nonzero entries of Σ are the **singular values** of X. These are real, nonnegative, and *rank-ordered* (decreasing from left to right).

The **singular value decomposition** of X is given by:

$$\underset{(n \times d)}{X} = \underset{(n \times n)}{U} \; \underset{(n \times d)}{\Sigma} \; \underset{(d \times d)}{V^T}$$

NOTE

The number of singular values is equal to the *rank* of X.

The rank of a matrix measures its *non-degeneracy.*

The nonzero entries of Σ are the **singular values** of X. These are real, nonnegative, and *rank-ordered* (decreasing from left to right).

For a general SVD, the columns of U are the eigenvectors of XX^T, and the columns of V are the eigenvectors of X^TX.

Also, the singular values of X are the square roots of the eigenvalues of XX^T and X^TX.

For a general SVD, the columns of U are the eigenvectors of XX^T, and the columns of V are the eigenvectors of X^TX.

Also, the singular values of X are the square roots of the eigenvalues of XX^T and X^TX.

NOTE

If data is centered, these are covariance matrices.

Q: How do you interpret the SVD?

Q: How do you interpret the SVD?

A: Recall that given a set of n points in d-dimensional space (eg, a matrix X), we want to find the best $k < d$ dimensional subspace to represent the data.

Q: How do you interpret the SVD?

A: Recall that given a set of n points in d-dimensional sp
matrix X), we want to find the best $k < d$ dimensional o
represent the data.

NOTE

Here "best" refers to the representation that minimizes the squared *orthogonal* distances from the points to the subspace.

Q: How do you interpret the SVD?

A: Recall that given a set of n points in d-dimensional space (eg, a matrix X), we want to find the best $k < d$ dimensional subspace to represent the data.

For $k = 1$, this subspace is a line passing through the origin.

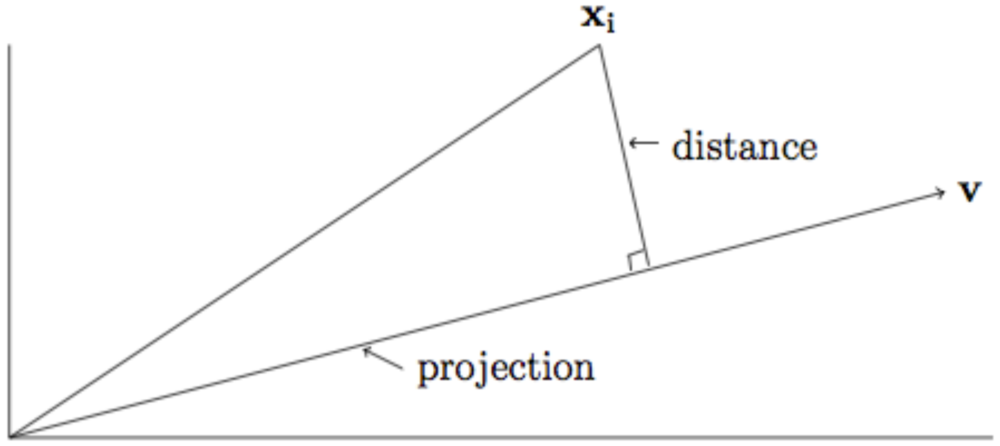


Figure 4.1: The projection of the point $\mathbf{x_i}$ onto the line through the origin in the direction of $\mathbf{v}$

source: http://www.cs.princeton.edu/courses/archive/spring12/cos598C/svdchapter.pdf

For a geometric interpretation of the singular values, consider a unit sphere in R_n and a linear map T (eg, a rotation and a stretch) that sends this sphere to an ellipsoid in R_d.

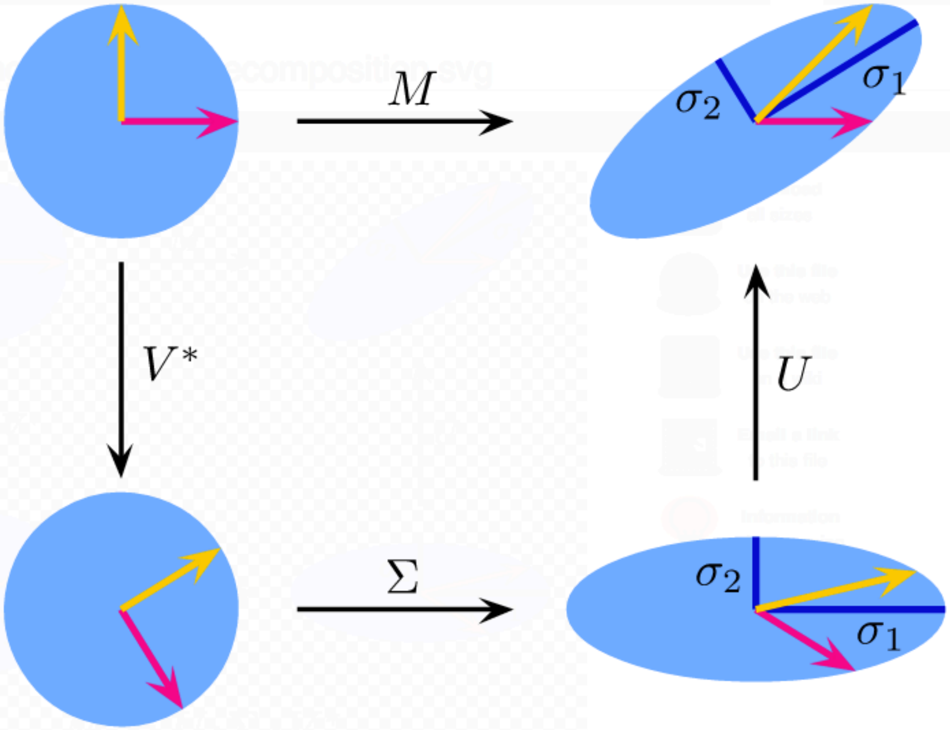
For a geometric interpretation of the singular values, consider a unit sphere in R_n and a linear map T (eg, a rotation and a stretch) that sends this sphere to an ellipsoid in R_d.

The singular vectors of T correspond to the lengths of the axes of the d-dimensional ellipsoid.

For a geometric interpretation of the singular values, consider a unit sphere in R_n and a linear map T (eg, a rotation and a stretch) that sends this sphere to an ellipsoid in R_d.

The singular vectors of T correspond to the lengths of the axes of the d-dimensional ellipsoid.

The singular values give the magnitudes of the projection of each column of the original dataset on the elements of the new basis.



$$M = U \cdot \Sigma \cdot V^*$$

source: http://en.wikipedia.org/wiki/Singular_value_decomposition

Why do we care about SVD?

Why do we care about SVD?

- More numerically stable and can be more efficient to calculate (than PCA)

Why do we care about SVD?

- More numerically stable and can be more efficient to calculate (than PCA)

- Latent semantic analysis, etc.

III. OTHER METHODS

Whereas PCA and SVD create new coordinates by transforming the old coordinates without any accompanying theory of what anything means, **factor analysis** refers to a broader array of techniques.

Whereas PCA and SVD create new coordinates by transforming the old coordinates without any accompanying theory of what anything means, **factor analysis** refers to a broader array of techniques.

In factor analysis, which may be exploratory or confirmatory, we hypothesize that our data depends on some *hidden* or *latent* features.

Whereas PCA and SVD create new coordinates by transforming the old coordinates without any accompanying theory of what anything means, **factor analysis** refers to a broader array of techniques.

In factor analysis, which may be exploratory or confirmatory, we hypothesize that our data depends on some *hidden* or *latent* features.

The old coordinates are then modeled as linear combinations of the latent features.

For example, consider a dataset that represents the results of a decathalon (rows = participants, columns = events, entries = times).

For example, consider a dataset that represents the results of a decathalon (rows = participants, columns = events, entries = times).

Though this dataset contains 10 features X_i, we may be interested in modeling these features as functions of *latent variables* such as the speed and strength of the participants:

$$X_i = \lambda_1 f_1 + \lambda_2 f_2 + \varepsilon$$

For example, consider a dataset that represents the results of a decathalon (rows = participants, columns = events, entries = times).

Though this dataset contains 10 features X_i, we may be interested in modeling these features as functions of *latent variables* such as the speed and strength of the participants:

$$X_i = \lambda_1 f_1 + \lambda_2 f_2 + \varepsilon$$

This is a now model with an error term!

In practice, PCA is often used for factor analysis, after modifying the covariance matrix somewhat. But it can also allow for non-isotropic errors, and there are other methods for fitting as well, and different theoretical concerns.

SVD, PCA, and factor analysis are all linear techniques (eg, we use a linear transformation to embed the data in a lower-dimensional space).

But sometimes linear techniques are not sufficient.

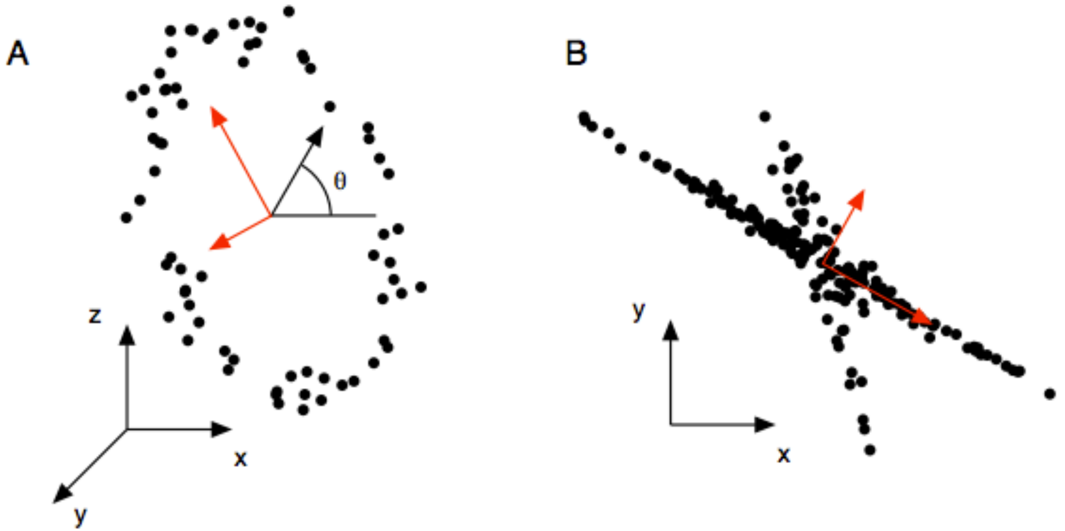
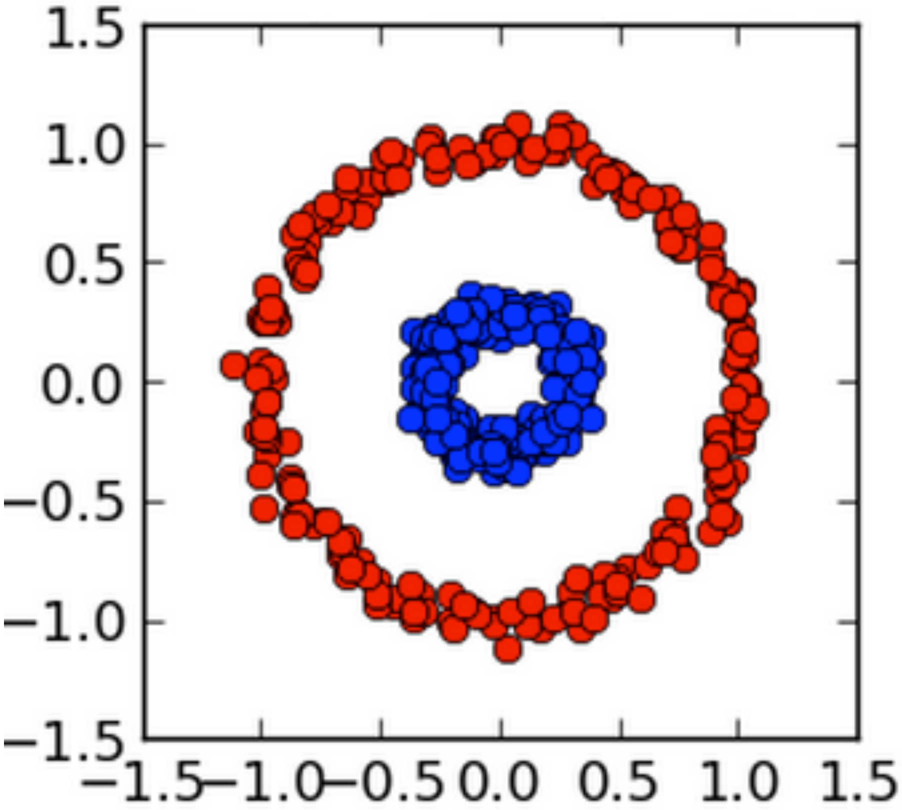


FIG. 6 Example of when PCA fails (red lines). (a) Tracking a person on a ferris wheel (black dots). All dynamics can be described by the phase of the wheel θ, a non-linear combination of the naive basis. (b) In this example data set, non-Gaussian distributed data and non-orthogonal axes causes PCA to fail. The axes with the largest variance do not correspond to the appropriate answer.


1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5

Some methods for nonlinear dimensional reduction (or *manifold learning*) include:

multidimensional scaling: low-dim embedding that preserves pairwise distances

locally linear embedding: approximates local structure of data (neighborhood preserving embedding)

Some methods for nonlinear dimensional reduction (or *manifold learning*) include:

NOTE

See sklearn.manifold

multidimensional scaling: low-dim embedding that pres
pairwise distances

locally linear embedding: approximates local structure of data
(neighborhood preserving embedding)

Some methods for nonlinear dimensional reduction (or *manifold learning*) include:

kernel PCA: exploits PCA dependence on inner product (same logic as SVM)

isomap: nonlinear dimension reduction via MDS using geodesic (surface-bound) distances

Some methods for nonlinear dimensional reduction (or *manifold learning*) include:

kernel PCA: exploits PCA dependence on inner product (same logic as SVM)

isomap: nonlinear dimension reduction via MDS using geodesic (surface-bound) distances

NOTE

See sklearn.decomposition and sklearn.manifold

Some methods for nonlinear dimensional reduction (or *manifold learning*) include:

kernel PCA: exploits PCA dependence on inner product (same logic as SVM)

isomap: nonlinear dimension reduction via MDS usir (surface-bound) distances

NOTE

See sklearn.decomposition and sklearn.manifold

NOTE

And more!

In any case, key difficulties with dimensionality reduction are time/space complexity, randomness (eg different results for different runs), and selecting the number of dimensions in the lower-dim subspace.

In any case, key difficulties with dimensionality reduction are time/space complexity, randomness (eg different results for different runs), and selecting the number of dimensions in the lower-dim subspace.

Furthermore, there's an obvious (bias/variance) tradeoff involved with the number of subspace dimensions and the size of approximation error.

IV. EXERCISE