GENERAL ASSEMBLY

DIMENSIONALITY REDUCTION

Brian Chung

LAST TIME:

-BAGGING (RANDOM SAMPLES FROM THE DATA)
-RANDOM FOREST (RANDOM SAMPLES + RANDOM FEATURES)
-BOOSTING (SEQUENTIAL LEARNING FROM MISTAKES)

- WHEN USE BAGGING VS RANDOM FOREST?
- HYPERPARAMETERS ALL THE SAME FOR BOOSTING?

TYPES OF ML SOLUTIONS

	Continuous	Categorical
Supervised	Regression	Classification
Unsupervised	Dimension Reduction	Clustering

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 is 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 is 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.

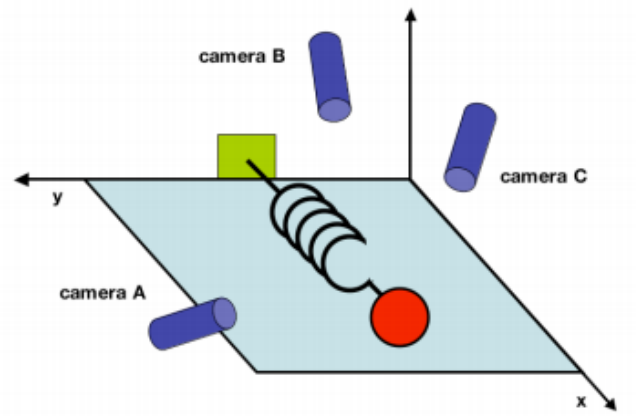
For example, suppose we have a dataset with some features that are related to each other.

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.



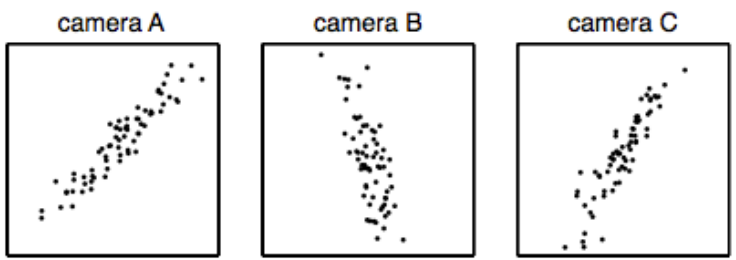
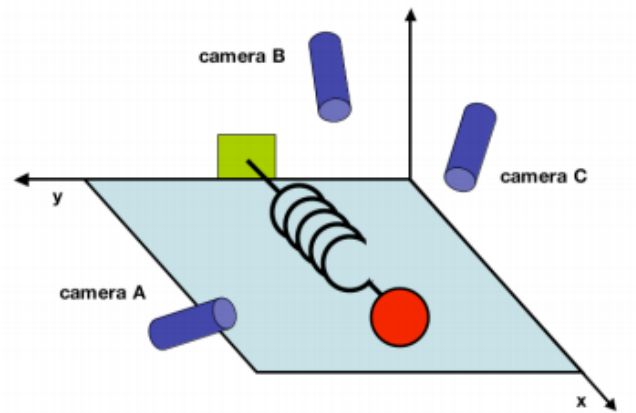


FIG. 1 A toy example. The position of a ball attached to an oscillating spring is recorded using three cameras A, B and C. The position of the ball tracked by each camera is depicted in each panel below.

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



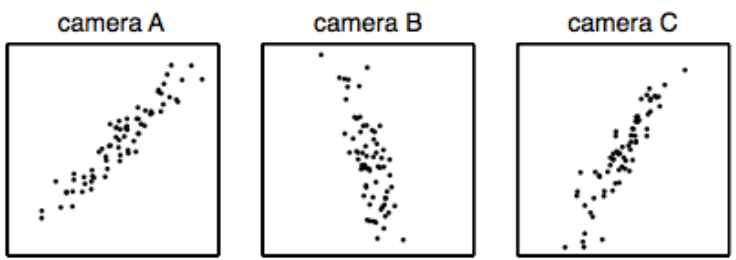


FIG. 1 A toy example. The position of a ball attached to an oscillating spring is recorded using three cameras A, B and C. The position of the ball tracked by each camera is depicted in each panel below.

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

NOTE

Here, the "truth" is (nearly) one-dimensional. In general, we don't 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).

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 A (encoding n observations of a d-dimensional random variable), we want to find a k-dimensional representation of A ($k < d$) 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*

Q: How is dimensionality reduction performed?

A: There are two approaches: feature selection and featur

NOTE

We've already seen one example of feature selection for regression: *LASSO*

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.

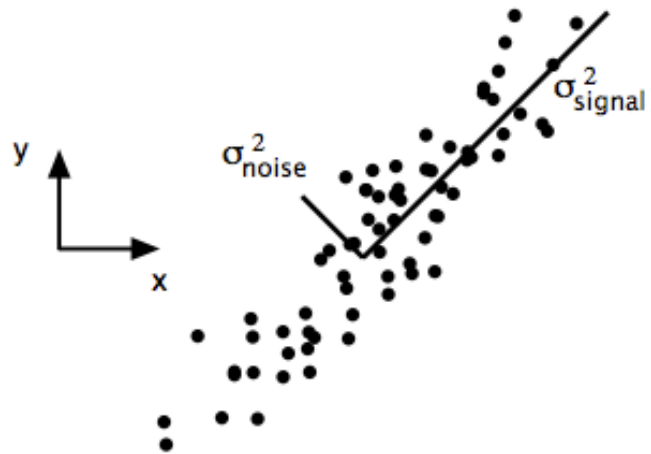


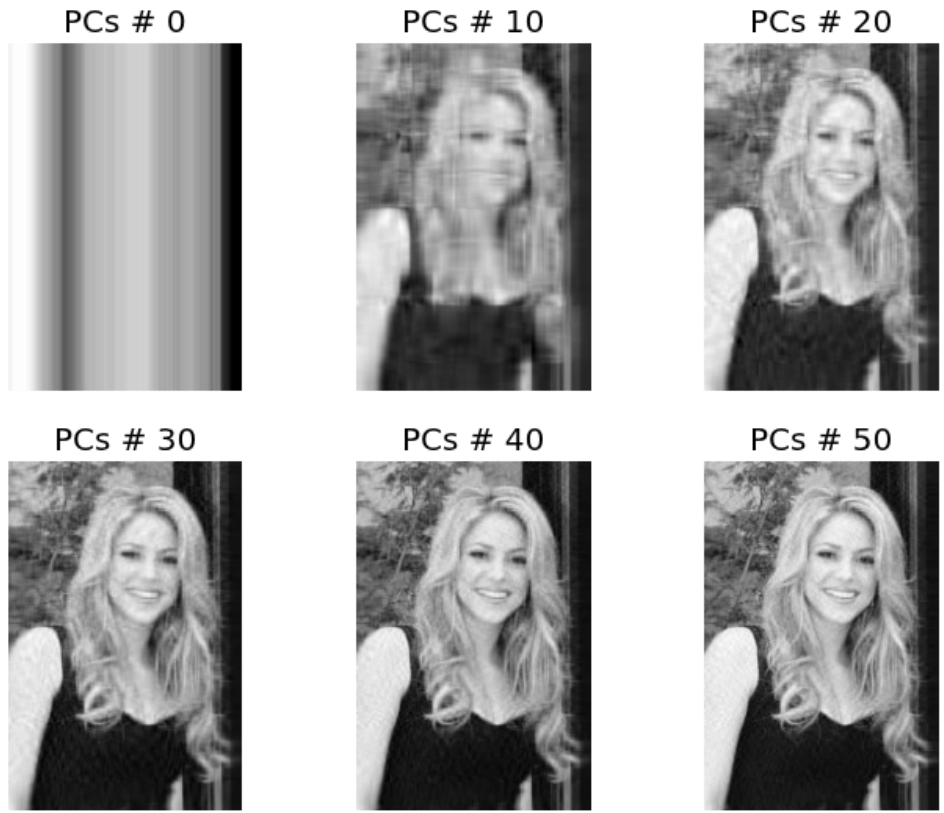
FIG. 2 Simulated data of (x, y) for camera A. The signal and noise variances σ^2_{signal} and σ^2_{noise} are graphically represented by the two lines subtending the cloud of data. Note that the largest direction of variance does not lie along the basis of the recording (x_A, y_A) but rather along the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

The covariance matrix C of a matrix A 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

Wait...what is a covariance?

Let's start off with variance...

$$\sigma^2 = \frac{\sum_{i=1}^{n} (X_i - \overline{X})^2}{n}$$

Variance is the average squared distance from the mean of a dataset. It measures the spread of the data. The square root of the variance is the standard deviation

Variance operates only on 1 dimension. If we had multiple features we could look at the variance of each feature individually, but maybe we want to look at how much two features vary from the mean with respect to each other.

This is called covariance.

Covariance operates between two vectors. If you calculate the covariance between a vector X and itself, you will get X's variance.

$$\operatorname{cov}(X,Y) = \frac{\sum_{i=1}^{n}(X_i - \overline{X})(Y_i - \overline{Y})}{n}$$

Given some data matrix X with features along columns x_1, x_2, x_3, …, x_n,

$$\mathrm{cov}(X) = \begin{pmatrix} \mathrm{cov}(x_1, x_1) & \mathrm{cov}(x_2, x_1) & \mathrm{cov}(x_n, x_1) \\ \mathrm{cov}(x_1, x_2) & \mathrm{cov}(x_2, x_2) & \mathrm{cov}(x_n, x_2) \\ \mathrm{cov}(x_1, x_n) & \mathrm{cov}(x_2, x_n) & \mathrm{cov}(x_n, x_n) \end{pmatrix}$$

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 A is given by:

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

The eigenvalue decomposition of a square matrix A is given by:

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

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

The eigenvalue decomposition of a square matrix A is given by:

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

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

For an eigenvector v of A and its eigenvalue λ, we have the important relation: $Av = \lambda v$

The eigenvalue decomposition of a square matrix A is given by:

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

The columns of Q are the **eigenvectors** *of A, and the valu*
the associated **eigenvalues** *of A.*

NOTE

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

For an eigenvector v of A and its eigenvalue λ, we have the important relation: $Av = \lambda v$

In order to perform PCA, we perform the eigenvalue decomposition of the covariance matrix of our data matrix X

1. *Determine covariance matrix of X*
2. *Perform eigenvalue decomposition to obtain the* ***eigenvectors*** *and* ***eigenvalues*** *of the covariance matrix*

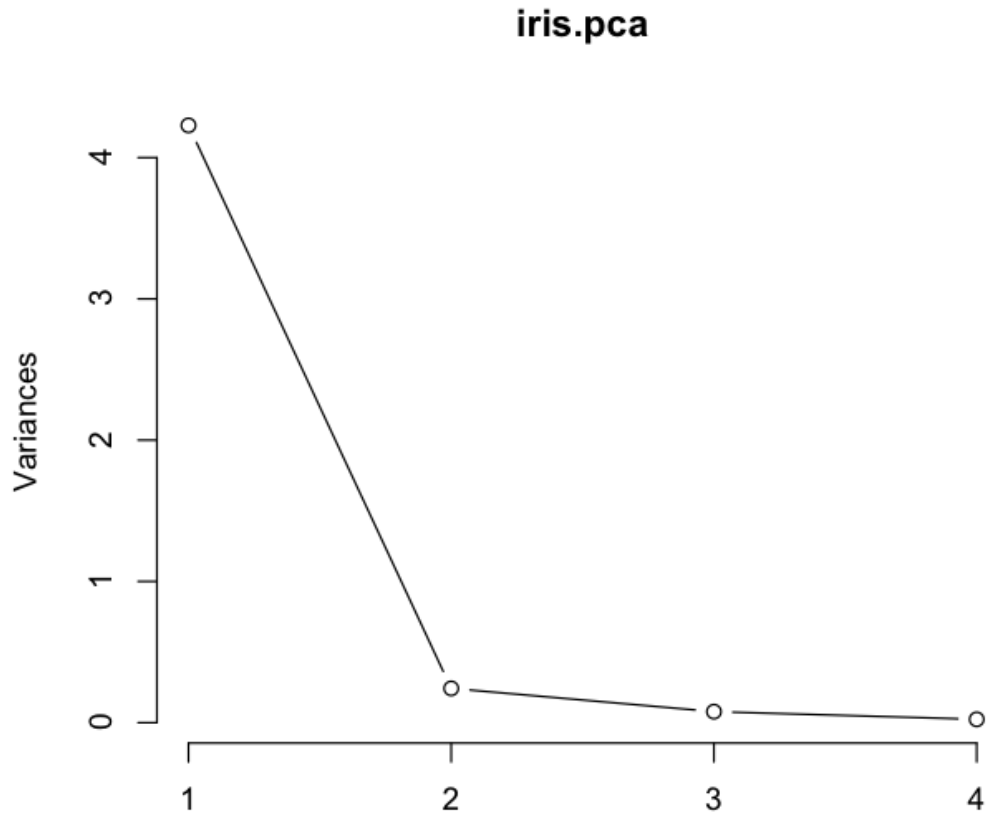
The eigenvectors form a basis of the vector space on which A acts (eg, they are orthogonal).

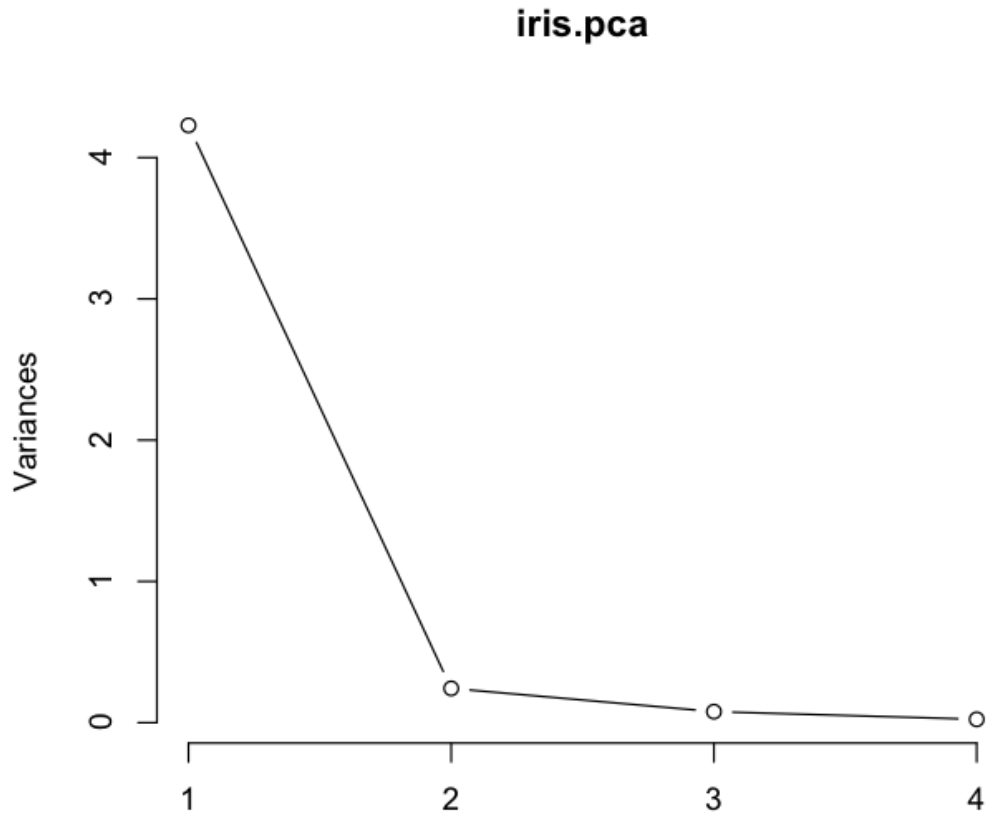
Furthermore the basis elements are ordered by their eigenvalues (from largest to smallest), and these eigenvalues represent the amount of variance explained by each basis element.

The eigenvectors form a basis of the vector space on which A 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.

- *PCA assumes that the change in basis is a **linear** projection (i.e. the new set of basis can be described as a linear combination of the original basis)*
- *Large variances have important structure – We assume that principal components associated with larger variances have more signal than noise (important but not always correct!)*
- *Principal components are orthogonal*

III. SINGULAR VALUE DECOMPOSITION

Lots of math ahead. If you are bewildered, don't worry.

Take your time to read through things later, and take it slow :)

Consider a matrix A with n rows and d features.

Consider a matrix A with n rows and d features.

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

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

Consider a matrix A with n rows and d features.

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

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

(n x d) (n x n) (n x d) (d x d)

Consider a matrix A with n rows and d features.

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

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

(n x d) (n x n) (n x d) (d x d)

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

Consider a matrix A with n rows and d features.

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

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

(n x d) (n x n) (n x d) (d x d)

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

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

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

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

(n x d) (n x n) (n x d) (d x d)

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

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

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

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

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

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

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

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

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

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

(n x d) (n x n) (n x d) (d x d)

NOTE

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

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

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

For a general SVD, the columns of U are the eigenvectors of AA^T, and the columns of V are the eigenvectors of A^TA.

Also, the singular values of A are the square roots of the eigenvalues of AA^T and A^TA.

For a general SVD, the columns of U are the eigenvectors of AA^T, and the columns of V are the eigenvectors of A^TA.

Also, the singular values of A are the square roots of the eigenvalues of AA^T and A^TA.

NOTE

If the data is centered, these are proportional to 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 A), 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 A), we want to find the best $k < d$ dimensional … 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 A), 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.

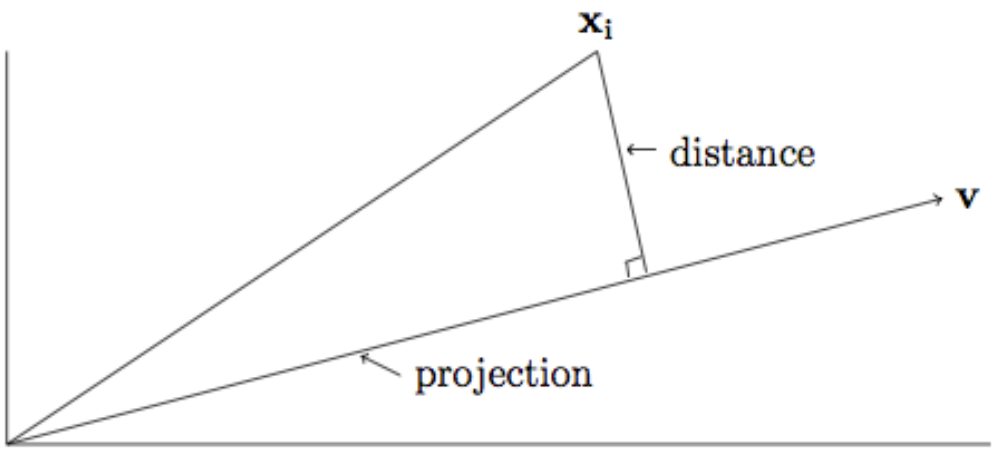


Figure 4.1: The projection of the point $\mathbf{x_i}$ onto the line through the origin in the direction of $\mathbf{v}$

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.

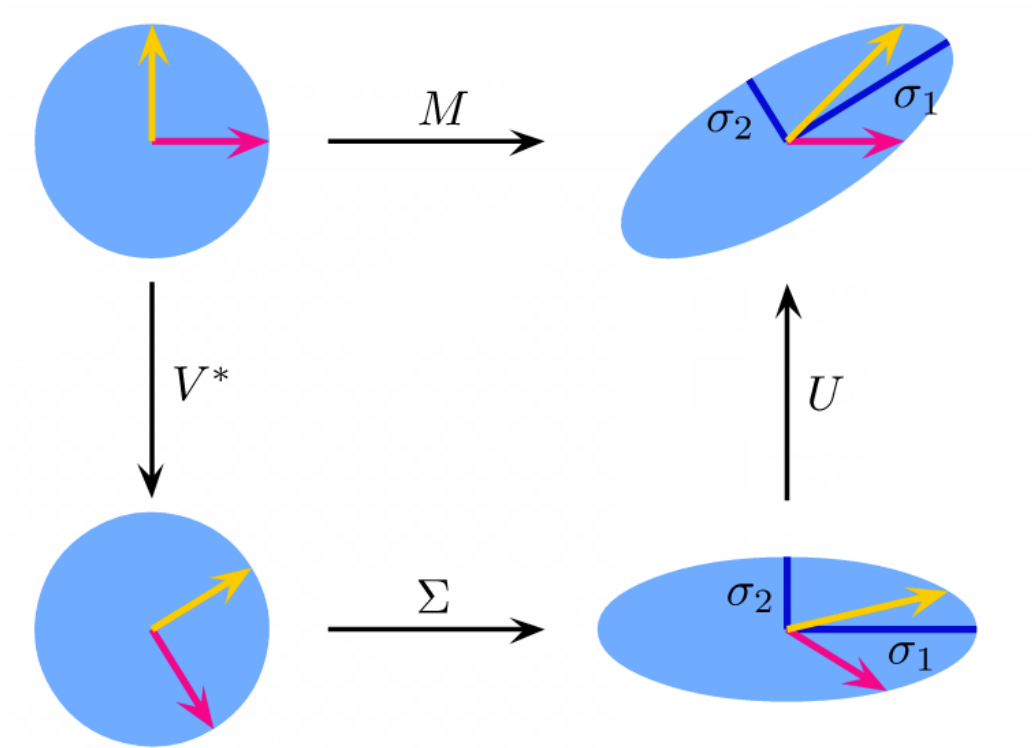
For a geometric interpretation of the singular values, consider a unit sphere in R_n and a linear map T (eg, a rotation and a stretch) that sends this sphere to an ellipsoid in R_d.

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

- *More numerically stable and efficient to calculate than PCA*
- *Can be used in Latent semantic analysis and recommendation systems*

III. OTHER METHODS

Whereas PCA and SVD create new coordinates by transform the old coordinates, **factor analysis** *requires new coordinates to be specified externally.*

Whereas PCA and SVD create new coordinates by transform the old coordinates, **factor analysis** *requires new coordinates to be specified externally.*

These new coordinates are associated with hidden or latent features that we think our data depends on.

Whereas PCA and SVD create new coordinates by transform the old coordinates, **factor analysis** *requires new coordinates to be specified externally.*

These new coordinates are associated with hidden or latent features that we think our data depends on.

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 would allow us to analyze the data in a more fundamental way.

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 as we saw with SVM's, sometimes linear techniques are not sufficient.

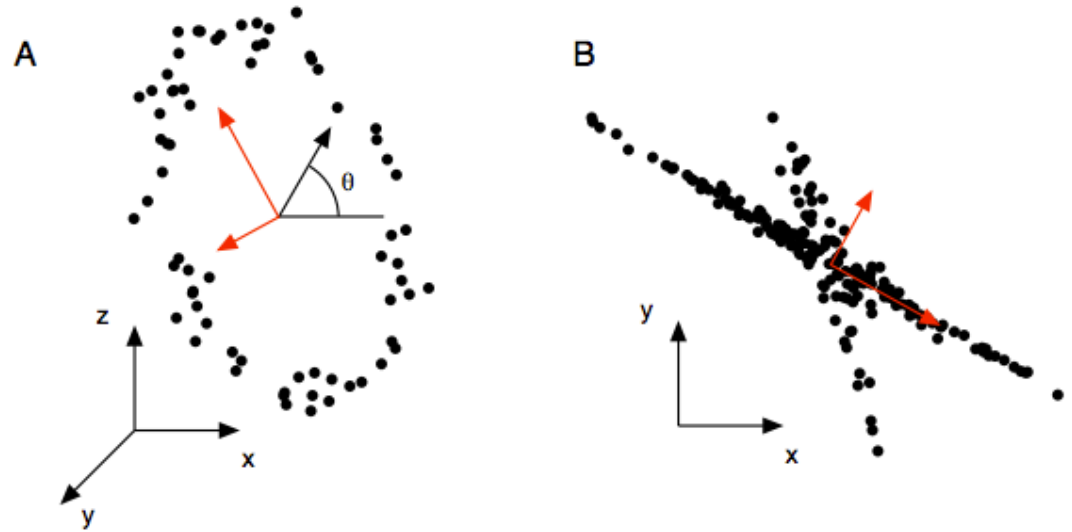
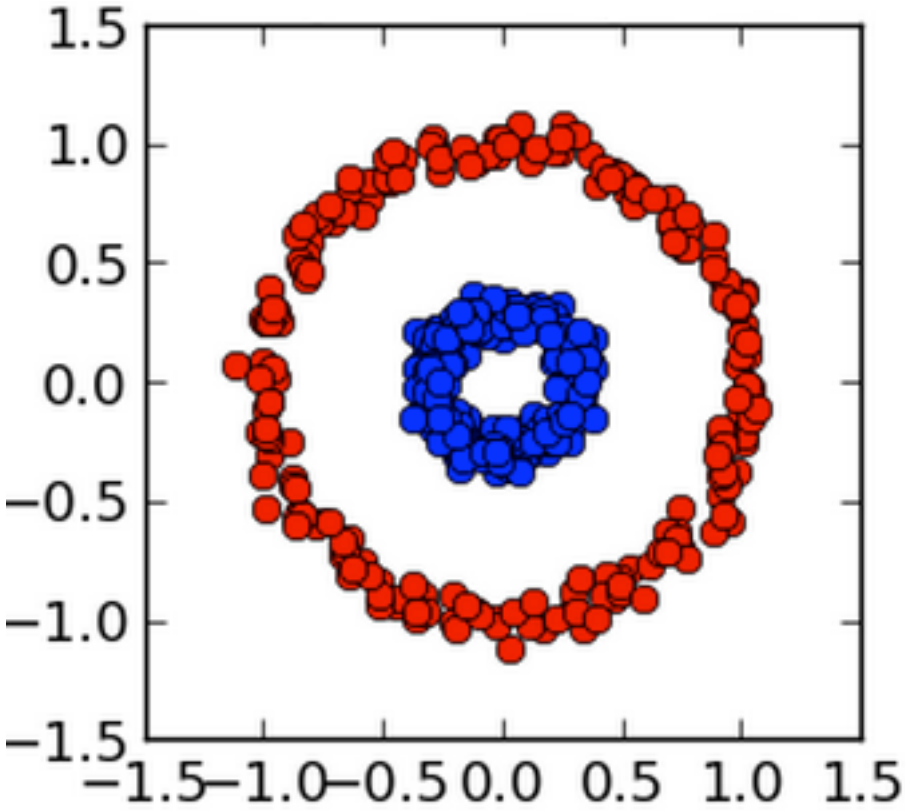


FIG. 6 Example of when PCA fails (red lines). (a) Tracking a person on a ferris wheel (black dots). All dynamics can be described by the phase of the wheel θ, a non-linear combination of the naive basis. (b) In this example data set, non-Gaussian distributed data and non-orthogonal axes causes PCA to 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*

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

multidimensional scaling: *low-dim embedding that pres*
pairwise distances

locally linear embedding: *approximates local structure of data*

NOTE
See 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 dim reduction via MDS using geodesic (surface-bound) distances*

In any case, the 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, the 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 between the number of subspace dimensions and the size of approximation error.

THAT'S IT!

- Exit Tickets: DAT1 - Lesson 14 - Dimensionality Reduction
- More DR resources:
- PCA https://www.cs.princeton.edu/picasso/mats/PCA-Tutorial-Intuition_jp.pdf
- PCA http://ufldl.stanford.edu/wiki/index.php/PCA
- SVD vs PCA http://math.stackexchange.com/questions/3869/what-is-the-intuitive-relationship-between-svd-and-pca
- How are projects coming?