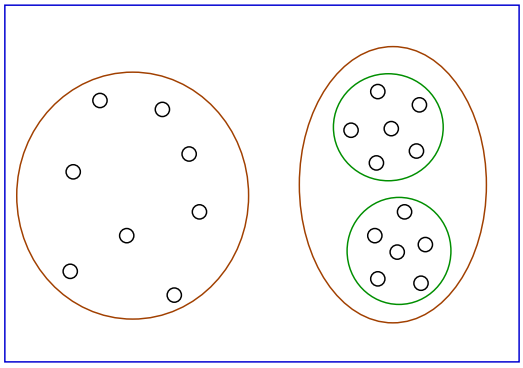
Clustering

DSE 210

Clustering in $\mathbb{R}^d$



Two common uses of clustering:

- Vector quantization
  Find a finite set of representatives that provides good coverage of a complex, possibly infinite, high-dimensional space.
- Finding meaningful structure in data
  Finding salient grouping in data.

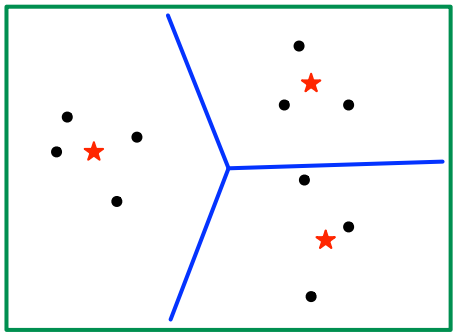
Widely-used clustering methods

1. K-means and its many variants
2. EM for mixtures of Gaussians
3. Agglomerative hierarchical clustering

The k-means optimization problem

- Input: Points $x_1, \ldots, x_n \in \mathbb{R}^d$; integer k
- Output: "Centers", or representatives, $\mu_1, \ldots, \mu_k \in \mathbb{R}^d$
- Goal: Minimize average squared distance between points and their nearest representatives:

$$\text{cost}(\mu_1, \ldots, \mu_k) = \sum_{i=1}^{n} \min_j \|x_i - \mu_j\|^2$$

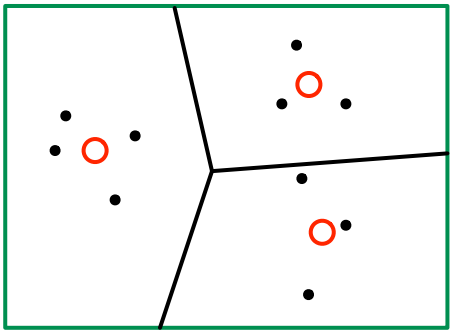


The centers carve $\mathbb{R}^d$ up into k convex regions: μ_j's region consists of points for which it is the closest center.

Lloyd's k-means algorithm

The k-means problem is NP-hard to solve. The most popular heuristic is called the "k-means algorithm".

- Initialize centers $\mu_1, \ldots, \mu_k$ in some manner.
- Repeat until convergence:
  - Assign each point to its closest center.
  - Update each μ_j to the mean of the points assigned to it.



Each iteration reduces the cost $\Rightarrow$ convergence to a local optimum.

Initializing the k-means algorithm

Typical practice: choose k data points at random as the initial centers.

Another common trick: start with extra centers, then prune later.

A particularly good initializer: **k-means++**

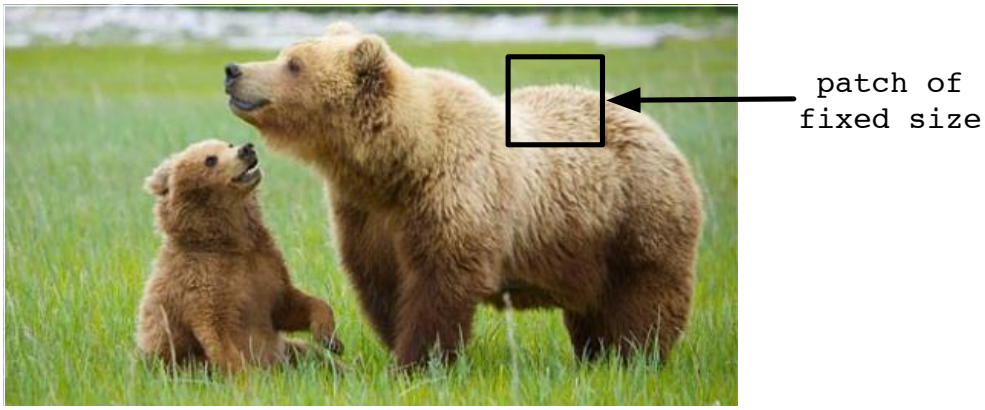
- Pick a data point x at random as the first center
- Let $C = \{x\}$ (centers chosen so far)
- Repeat until desired number of centers is attained:
  - Pick a data point x at random from the following distribution:

$$\Pr(x) \propto \text{dist}(x, C)^2,$$

    where $\text{dist}(x, C) = \min_{z \in C} \|x - z\|$
  - Add x to C

Representing images using k-means codewords

Given a collection of images, how to represent as fixed-length vectors?



- Look at all $\ell \times \ell$ patches in all images.
- Run k-means on this entire collection to get k centers.
- Now associate any image patch with its nearest center.
- Represent an image by a histogram over $\{1, 2, \ldots, k\}$.

Such data sets are truly enormous.

Streaming and online computation

Streaming computation: for data sets that are too large to fit in memory.

- Make one pass (or maybe a few passes) through the data.
- On each pass:
  - See data points one at a time, in order.
  - Update models/parameters along the way.
- There is only enough space to store a tiny fraction of the data, or a perhaps short summary.

Online computation: an even more lightweight setup, for data that is continuously being collected.

- Initialize a model.
- Repeat forever:
  - See a new data point.
  - Update model if need be.

Example: sequential k-means

1. Set the centers $\mu_1, \ldots, \mu_k$ to the first k data points
2. Set their counts to $n_1 = n_2 = \cdots = n_k = 1$
3. Repeat, possibly foreover:
   - Get next data point x
   - Let μ_j be the center closest to x
   - Update μ_j and n_j:

$$\mu_j = \frac{n_j \mu_j + x}{n_j + 1} \quad \text{and} \quad n_j = n_j + 1$$

K-means: the good and the bad

The good:

- Fast and easy.
- Effective in quantization.

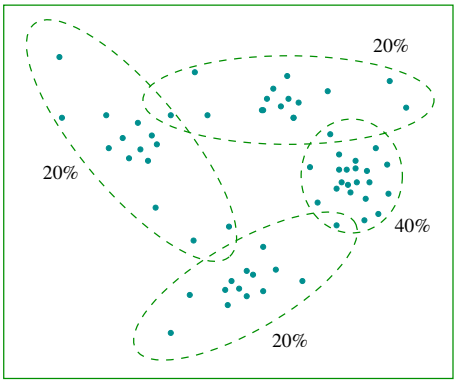
The bad:

- Geared towards data in which the clusters are spherical, and of roughly the same radius.

Is there is a similarly-simple algorithm in which clusters of more general shape are accommodated?

Mixtures of Gaussians

Idea: model each cluster by a Gaussian:



Each of the k clusters is specified by:

- a Gaussian distribution $P_j = N(\mu_j, \Sigma_j)$
- a mixing weight π_j

Overall distribution over $\mathbb{R}^d$: a **mixture of Gaussians**

$$\Pr(x) = \pi_1 P_1(x) + \cdots + \pi_k P_k(x)$$

The clustering task

Given data $x_1, \ldots, x_n \in \mathbb{R}^d$, find the maximum-likelihood mixture of Gaussians: that is, find parameters

- $\pi_1, \ldots, \pi_k \geq 0$ summing to one
- $\mu_1, \ldots, \mu_k \in \mathbb{R}^d$
- $\Sigma_1, \ldots, \Sigma_k \in \mathbb{R}^{d \times d}$

to maximize

$$\begin{aligned}
&\Pr\left(\text{data} \mid \pi_1 P_1 + \cdots + \pi_k P_k\right) \\
&= \prod_{i=1}^{n} \left(\sum_{j=1}^{k} \pi_j P_j(x_i) \right) \\
&= \prod_{i=1}^{n} \left(\sum_{j=1}^{k} \frac{\pi_j}{(2\pi)^{p/2} |\Sigma_j|^{1/2}} \exp\left(-\frac{1}{2} (x_i - \mu_j)^T \Sigma_j^{-1} (x_i - \mu_j) \right) \right)
\end{aligned}$$

where P_j is the distribution of the jth cluster, $N(\mu_j, \Sigma_j)$.

The EM algorithm

1. Initialize $\pi_1, \ldots, \pi_k$ and $P_1 = N(\mu_1, \Sigma_1), \ldots, P_k = N(\mu_k, \Sigma_k)$ in some manner.
2. Repeat until convergence:
   - Assign each point x_i fractionally between the k clusters:

$$w_{ij} = \Pr(\text{cluster } j \mid x_i) = \frac{\pi_j P_j(x_i)}{\sum_\ell \pi_\ell P_\ell(x_i)}$$

   - Now update the mixing weights, means, and covariances:

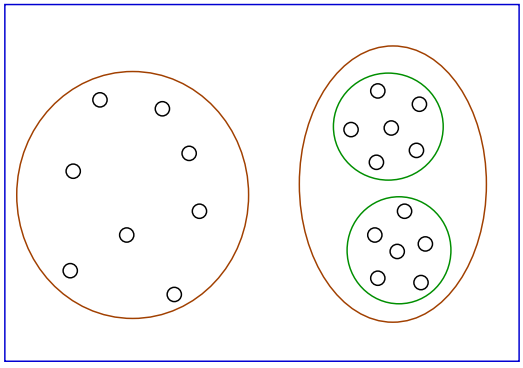
$$\pi_j = \frac{1}{n} \sum_{i=1}^{n} w_{ij}$$

$$\mu_j = \frac{1}{n\pi_j} \sum_{i=1}^{n} w_{ij} x_i$$

$$\Sigma_j = \frac{1}{n\pi_j} \sum_{i=1}^{n} w_{ij} (x_i - \mu_j)(x_i - \mu_j)^T$$

Hierarchical clustering

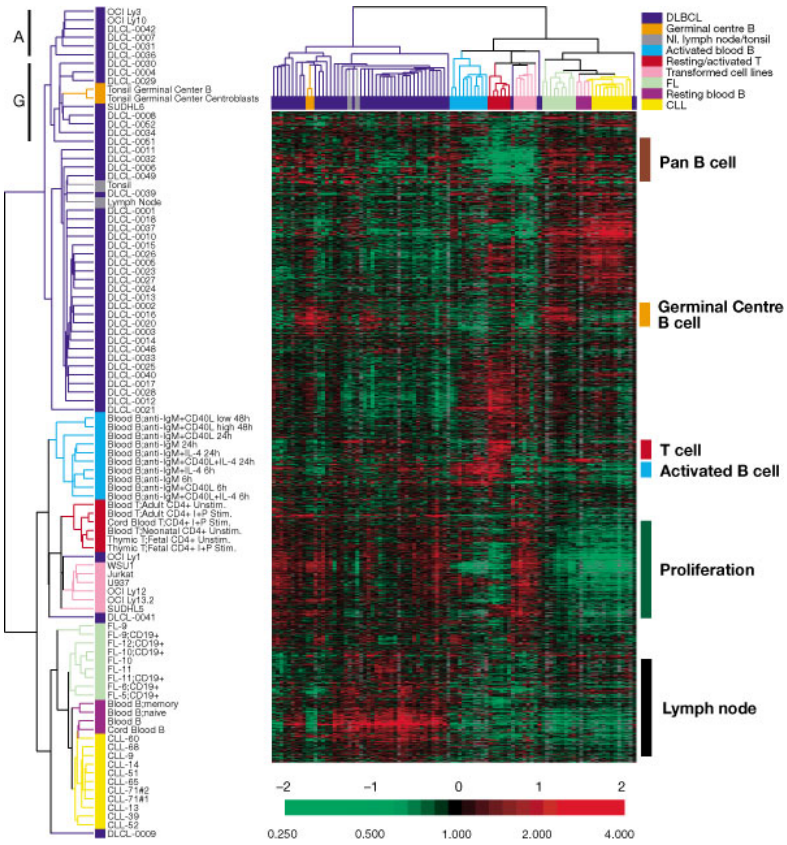
Choosing the number of clusters (k) is difficult.



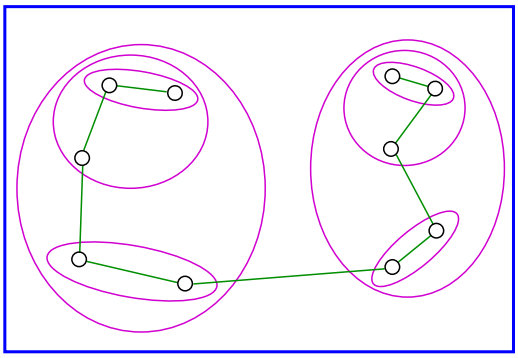
Often there is no single right answer, because of multiscale structure.

Hierarchical clustering avoids these problems.

Example: gene expression data



The single linkage algorithm

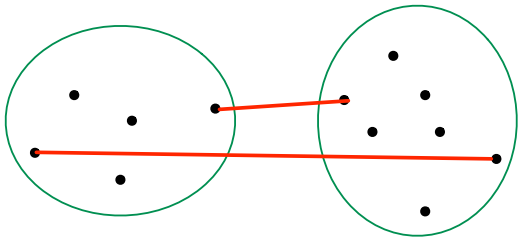


- Start with each point in its own, singleton, cluster
- Repeat until there is just one cluster:
  - Merge the two clusters with the closest pair of points
- Disregard singleton clusters

Linkage methods

- Start with each point in its own, singleton, cluster
- Repeat until there is just one cluster:
  - Merge the two "closest" clusters

How to measure the distance between two clusters of points, C and C'?



- Single linkage

$$\text{dist}(C, C') = \min_{x \in C, x' \in C'} \|x - x'\|$$

- Complete linkage

$$\text{dist}(C, C') = \max_{x \in C, x' \in C'} \|x - x'\|$$

Average linkage

Three commonly-used variants:

1. Average pairwise distance between points in the two clusters

$$\text{dist}(C, C') = \frac{1}{|C| \cdot |C'|} \sum_{x \in C} \sum_{x' \in C'} \|x - x'\|$$

2. Distance between cluster centers

$$\text{dist}(C, C') = \|\text{mean}(C) - \text{mean}(C')\|$$

3. Ward's method: the increase in k-means cost occasioned by merging the two clusters

$$\text{dist}(C, C') = \frac{|C| \cdot |C'|}{|C| + |C'|} \|\text{mean}(C) - \text{mean}(C')\|^2$$

DSE 210: Probability and statistics **Winter 2018**

Worksheet 9 — Clustering

1. For this problem, we'll be using the *animals with attributes* data set. Go to

   `http://attributes.kyb.tuebingen.mpg.de`

   and, under "Downloads", choose the "base package" (the very first file in the list). Unzip it and look over the various text files.

2. This is a small data set that has information about 50 animals. The animals are listed in `classes.txt`. For each animal, the information consists of values for 85 features: does the animal have a tail, is it slow, does it have tusks, etc. The details of the features are in `predicates.txt`. The full data consists of a 50×85 matrix of real values, in `predicate-matrix-continuous.txt`. There is also a binarized version of this data, in `predicate-matrix-binary.txt`.

3. Load the real-valued array, and also the animal names, into Python. Run k-means on the data (from `sklearn.cluster`) and ask for $k = 10$ clusters. For each cluster, list the animals in it. Does the clustering make sense?

4. Now hierarchically cluster this data, using `scipy.cluster.hierarchy.linkage`. Choose Ward's method, and plot the resulting tree using the `dendrogram` method, setting the `orientation` parameter to `'right'` and labeling each leaf with the corresponding animal name.

   You will run into a problem: the plot is too cramped because the default figure size is so small. To make it larger, preface your code with the following:

   ```
   from pylab import rcParams
   rcParams['figure.figsize'] = 5, 10
   ```

   (or try a different size if this doesn't seem quite right). Does the hierarchical clustering seem sensible to you?

5. Turn in an iPython notebook with a transcript of all this experimentation.

Informative projections

DSE 210

Dimensionality reduction

Why reduce the number of features in a data set?

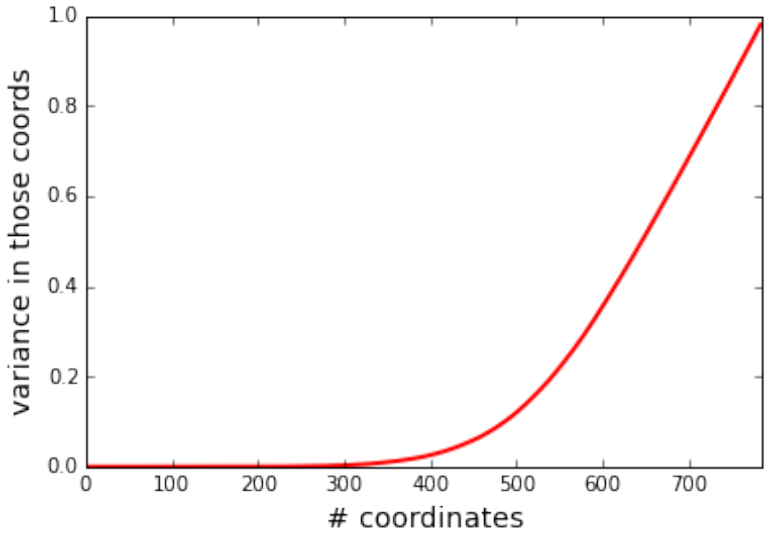
1. It reduces storage and computation time.
2. High-dimensional data often has a lot of redundancy.
3. Remove noisy or irrelevant features.

Example: are all the pixels in an image equally informative?

If we were to choose a few pixels to discard, which would be the prime candidates?

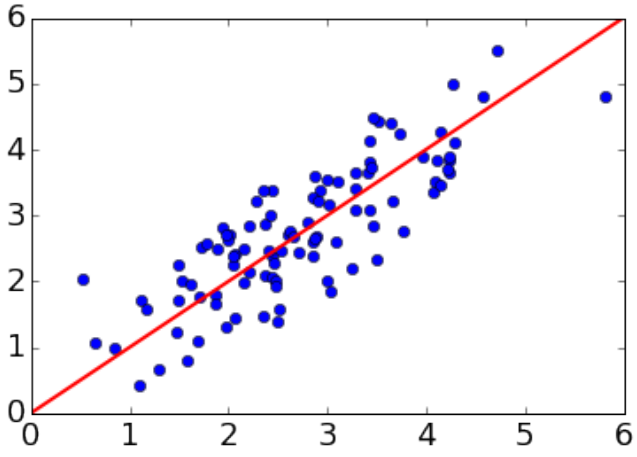
Eliminating low variance coordinates

MNIST: what fraction of the total variance lies in the 100 (or 200, or 300) coordinates with lowest variance?



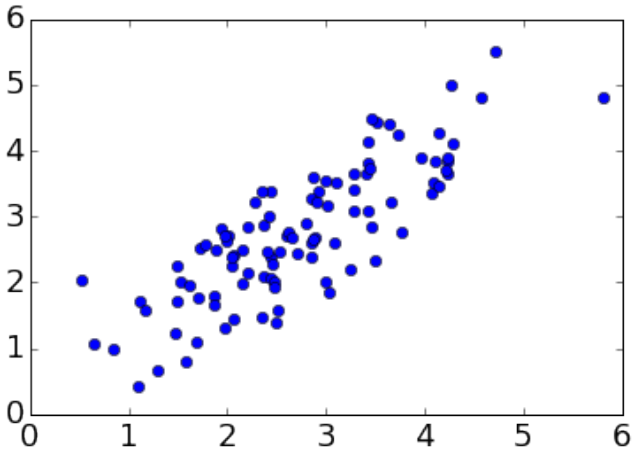
The effect of correlation

Suppose we wanted just one feature for the following data.



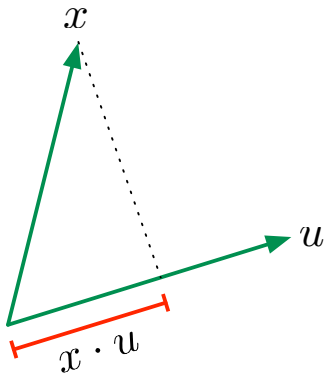
This is the **direction of maximum variance**.

Comparing projections



Projection: formally

What is the projection of $x \in \mathbb{R}^d$ in the **direction** $u \in \mathbb{R}^d$?
Assume u is a unit vector (i.e. $\|u\| = 1$).



Projection is

$$x \cdot u = u \cdot x = u^T x = \sum_{i=1}^{d} u_i x_i.$$

Examples

What is the projection of $x = \begin{pmatrix} 2 \\ 3 \end{pmatrix}$ along the following directions?

1. The x_1-axis?
2. The direction of $\begin{pmatrix} 1 \\ -1 \end{pmatrix}$?

The best direction

Suppose we need to map our data $x \in \mathbb{R}^d$ into just **one** dimension:

$$x \mapsto u \cdot x \quad \text{for some unit direction } u \in \mathbb{R}^d$$

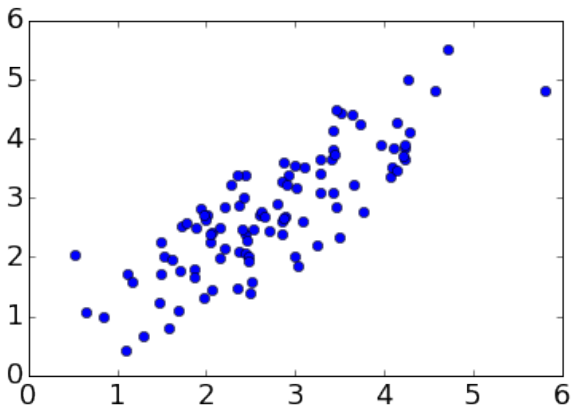
What is the direction u of maximum variance?

Useful fact 1:

- Let Σ be the $d \times d$ covariance matrix of X.
- The variance of X in direction u (the variance of $X \cdot u$) is:

$$u^T \Sigma u.$$

Best direction: example



Here covariance matrix $\Sigma = \begin{pmatrix} 1 & 0.85 \\ 0.85 & 1 \end{pmatrix}$

The best direction

Suppose we need to map our data $x \in \mathbb{R}^d$ into just **one** dimension:

$$x \mapsto u \cdot x \quad \text{for some unit direction } u \in \mathbb{R}^d$$

What is the direction u of maximum variance?

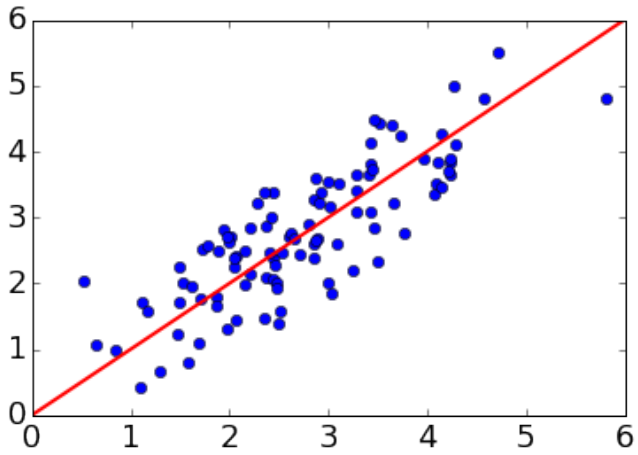
Useful fact 1:

- Let Σ be the $d \times d$ covariance matrix of X.
- The variance of X in direction u is given by $u^T \Sigma u$.

Useful fact 2:

- $u^T \Sigma u$ is maximized by setting u to the first **eigenvector** of Σ.
- The maximum value is the corresponding **eigenvalue**.

Best direction: example



Direction:
first eigenvector of the 2×2 covariance matrix of the data.

Projection onto this direction:
the top **principal component** of the data

Projection onto multiple directions

Projecting $x \in \mathbb{R}^d$ into the k-dimensional subspace defined by vectors $u_1, \ldots, u_k \in \mathbb{R}^d$.

This is easiest when the u_i's are **orthonormal**:

- They have length one.
- They are at right angles to each other: $u_i \cdot u_j = 0$ when $i \neq j$

The projection is a k-dimensional vector:

$$(x \cdot u_1, x \cdot u_2, \ldots, x \cdot u_k) = \underbrace{\begin{pmatrix} \longleftarrow & u_1 & \longrightarrow \\ \longleftarrow & u_2 & \longrightarrow \\ & \vdots & \\ \longleftarrow & u_k & \longrightarrow \end{pmatrix}}_{\text{call this } U^T} \begin{pmatrix} \uparrow \\ x \\ \downarrow \end{pmatrix}$$

U is the $d \times k$ matrix with columns $u_1, \ldots, u_k$.

Projection onto multiple directions: example

E.g. project data in $\mathbb{R}^4$ onto the first two coordinates.

Take vectors $u_1 = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$, $u_2 = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}$ (notice: orthonormal)

Write $U^T = \begin{pmatrix} \longleftarrow & u_1 & \longrightarrow \\ \longleftarrow & u_2 & \longrightarrow \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$

The projection of $x \in \mathbb{R}^4$ is $U^T x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$

The best k-dimensional projection

Let Σ be the $d \times d$ covariance matrix of X.
In $O(d^3)$ time, we can compute its **eigendecomposition**, consisting of

- real **eigenvalues** $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_d$
- corresponding **eigenvectors** $u_1, \ldots, u_d \in \mathbb{R}^d$ that are orthonormal (unit length and at right angles to each other)

Fact: Suppose we want to map data $X \in \mathbb{R}^d$ to just k dimensions, while capturing as much of the variance of X as possible. The best choice of projection is:

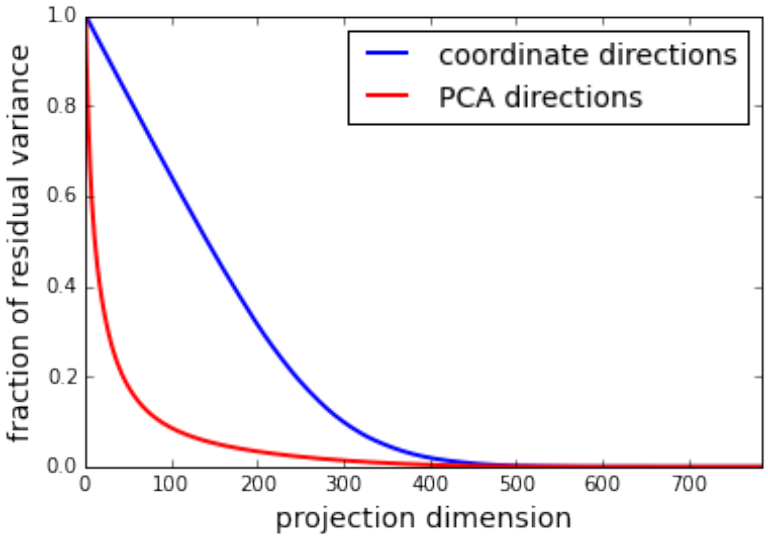
$$x \mapsto (u_1 \cdot x, u_2 \cdot x, \ldots, u_k \cdot x),$$

where u_i are the eigenvectors described above.

This projection is called **principal component analysis** (PCA).

Example: MNIST

Contrast coordinate projections with PCA:



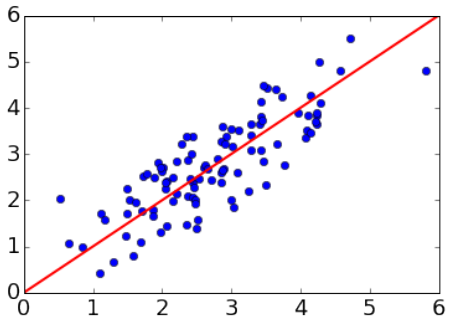
Applying PCA to MNIST: examples

Reconstruct this original image from its PCA projection to k dimensions.

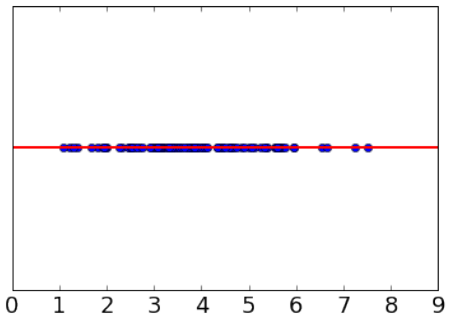
$k = 200$ $k = 150$ $k = 100$ $k = 50$

How do we get these **reconstructions**?

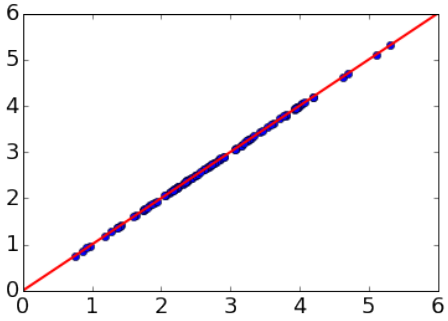
Reconstruction from a 1-d projection



Projection onto $\mathbb{R}$:



Reconstruction in $\mathbb{R}^2$:



Reconstruction from projection onto multiple directions

Projecting into the k-dimensional subspace defined by **orthonormal** $u_1, \ldots, u_k \in \mathbb{R}^d$.

The projection of x is a k-dimensional vector:

$$(x \cdot u_1, x \cdot u_2, \ldots, x \cdot u_k) = \underbrace{\begin{pmatrix} \longleftarrow & u_1 & \longrightarrow \\ \longleftarrow & u_2 & \longrightarrow \\ & \vdots & \\ \longleftarrow & u_k & \longrightarrow \end{pmatrix}}_{\text{call this } U^T} \begin{pmatrix} \uparrow \\ x \\ \downarrow \end{pmatrix}$$

The reconstruction from this projection is:

$$(x \cdot u_1)u_1 + (x \cdot u_2)u_2 + \cdots + (x \cdot u_k)u_k = UU^T x.$$

MNIST: image reconstruction

Reconstruct this original image x from its PCA projection to k dimensions.

$k = 200$ $k = 150$ $k = 100$ $k = 50$

Reconstruction $UU^T x$, where U's columns are top k eigenvectors of Σ.

Case study: personality assessment

What are the dimensions along which personalities differ?

- *Lexical hypothesis*: most important personality characteristics have become encoded in natural language.
- Allport and Odbert (1936): identified 4500 words describing personality traits.
- Group these words into (approximate) synonyms, by manual clustering.
  E.g. Norman (1967):

Spirit	Jolly, merry, witty, lively, peppy
Talkativeness	Talkative, articulate, verbose, gossipy
Sociability	Companionable, social, outgoing
Spontaneity	Impulsive, carefree, playful, zany
Boisterousness	Mischievous, rowdy, loud, prankish
Adventure	Brave, venturous, fearless, reckless
Energy	Active, assertive, dominant, energetic
Conceit	Boastful, conceited, egotistical
Vanity	Affected, vain, chic, dapper, jaunty
Indiscretion	Nosey, snoopy, indiscreet, meddlesome
Sensuality	Sexy, passionate, sensual, flirtatious

- Data collection: subjects whether these words describe them.

Personality assessment: the data

Matrix of data (1 = strongly disagree, 5 = strongly agree)

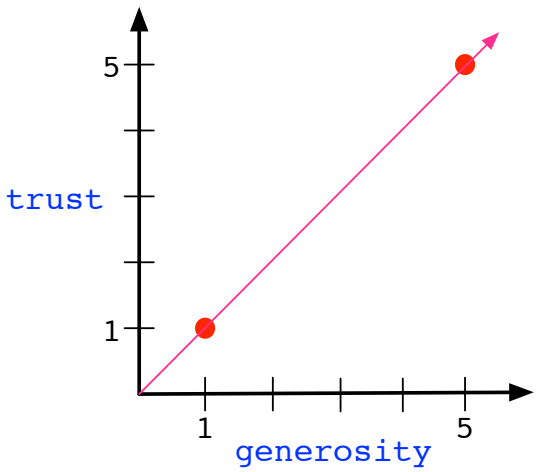
	shy	merry	tense	boastful	forgiving	quiet
Person 1	4	1	1	2	5	5
Person 2	1	4	4	5	2	1
Person 3	2	4	5	4	2	2
		⋮				

How to extract important directions?

- Treat each column as a data point, find tight clusters
- Treat each row as a data point, apply PCA
- Or factor analysis, independent component analysis, etc.

What would PCA accomplish?

E.g.: Suppose two traits (generosity, trust) are so highly correlated that each person either answers "1" to both or "5" to both.



A single PCA dimension would entirely account for both traits.

Personality assessment: the data

Matrix of data (1 = strongly disagree, 5 = strongly agree)

	shy	merry	tense	boastful	forgiving	quiet
Person 1	4	1	1	2	5	5
Person 2	1	4	4	5	2	1
Person 3	2	4	5	4	2	2
		⋮				

Methodology: apply PCA to the rows of this matrix.

The "Big Five" taxonomy

Extraversion
−: quiet (-.83), reserved (-.80), shy (-.75), silent (-.71)
+: talkative (.85), assertive (.83), active (.82), energetic (.82)

Agreeableness
−: fault-finding (-.52), cold (-.48), unfriendly (-.45), quarrelsome (-.45)
+: sympathetic (.87), kind (.85), appreciative (.85), affectionate (.84)

Conscientousness
−: careless (-.58), disorderly (-.53), frivolous (-.50), irresponsible (-.49)
+: organized (.80), thorough (.80), efficient (.78), responsible (.73)

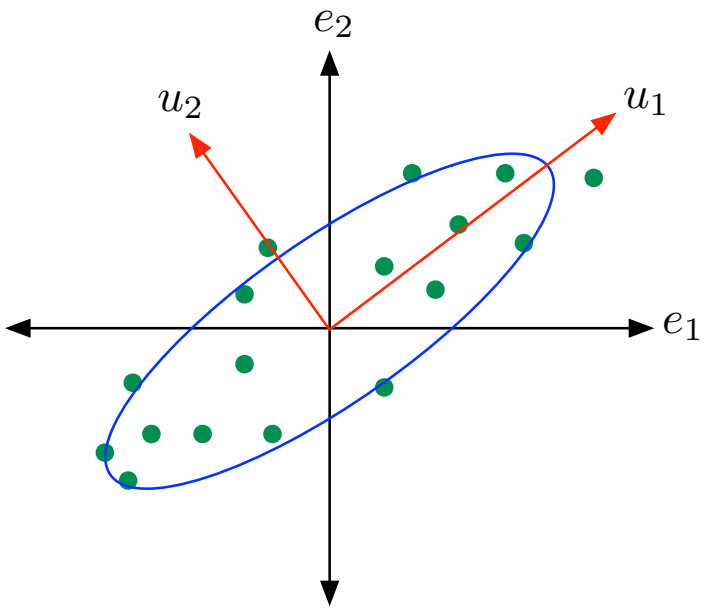
Neuroticism
−: stable (-.39), calm (-.35), contented (-.21)
+: tense (.73), anxious (.72), nervous (.72), moody (.71)

Openness
−: commonplace (-.74), narrow (-.73), simple (-.67), shallow (-.55)
+: imaginative (.76), intelligent (.72), original (.73), insightful (.68)

Singular value decomposition

DSE 210

Moving between coordinate systems



The linear function defined by a matrix

- Any matrix M defines a linear function, $x \mapsto Mx$.
  If M is a $d \times d$ matrix, this maps $\mathbb{R}^d$ to $\mathbb{R}^d$.

- This function is easy to understand when M is **diagonal**:

$$\underbrace{\begin{pmatrix} 2 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 10 \end{pmatrix}}_{M} \underbrace{\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}}_{x} = \underbrace{\begin{pmatrix} 2x_1 \\ -x_2 \\ 10x_3 \end{pmatrix}}_{Mx}$$

  In this case, M simply scales each coordinate separately.

- General symmetric matrices also just scale coordinates separately...
  but in a **different coordinate system!**

Eigenvector and eigenvalue: definition

Let M be a $d \times d$ matrix. We say $u \in \mathbb{R}^d$ is an **eigenvector** of M if

$$Mu = \lambda u$$

for some scaling constant λ. This λ is the **eigenvalue** associated with u.

Key point: M **maps eigenvector** u **onto the same direction**.

Question: What are the eigenvectors and eigenvalues of:

$$M = \begin{pmatrix} 2 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 10 \end{pmatrix} ?$$

Eigenvectors of a real symmetric matrix

Fact: Let M be any real symmetric $d \times d$ matrix. Then M has

- d eigenvalues $\lambda_1, \ldots, \lambda_d$
- corresponding eigenvectors $u_1, \ldots, u_d \in \mathbb{R}^d$ that are orthonormal

Can think of $u_1, \ldots, u_d$ as the axes of the natural coordinate system for M.

Example

$$M = \begin{pmatrix} 1 & -2 \\ -2 & 1 \end{pmatrix} \text{ has eigenvectors } u_1 = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ 1 \end{pmatrix},\ u_2 = \frac{1}{\sqrt{2}}\begin{pmatrix} -1 \\ 1 \end{pmatrix}$$

1. Are these orthonormal?
2. What are the corresponding eigenvalues?

Spectral decomposition

Fact: Let M be any real symmetric $d \times d$ matrix. Then M has orthonormal eigenvectors $u_1, \ldots, u_d \in \mathbb{R}^d$ and corresponding eigenvalues $\lambda_1, \ldots, \lambda_d$.

Spectral decomposition: Another way to write M:

$$M = \underbrace{\begin{pmatrix} \uparrow & \uparrow & & \uparrow \\ u_1 & u_2 & \cdots & u_d \\ \downarrow & \downarrow & & \downarrow \end{pmatrix}}_{U\text{: columns are eigenvectors}} \underbrace{\begin{pmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_d \end{pmatrix}}_{\Lambda\text{: eigenvalues on diagonal}} \underbrace{\begin{pmatrix} \longleftarrow & u_1 & \longrightarrow \\ \longleftarrow & u_2 & \longrightarrow \\ & \vdots & \\ \longleftarrow & u_d & \longrightarrow \end{pmatrix}}_{U^T}$$

Thus $Mx = U\Lambda U^T x$:

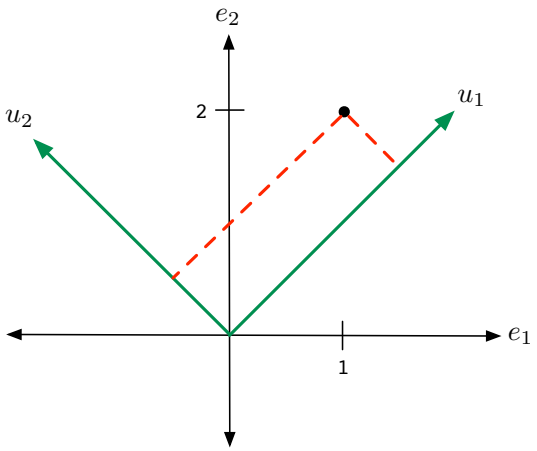
- U^T rewrites x in the $\{u_i\}$ coordinate system
- Λ is a simple coordinate scaling in that basis
- U sends the scaled vector back into the usual coordinate basis

Apply to the matrix we saw earlier: $M = \begin{pmatrix} 1 & -2 \\ -2 & 1 \end{pmatrix}$

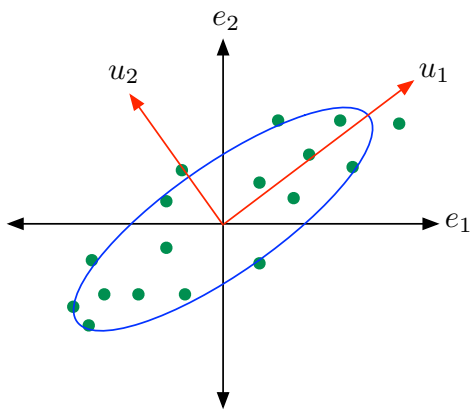
- Eigenvectors $u_1 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix}$, $u_2 = \frac{1}{\sqrt{2}} \begin{pmatrix} -1 \\ 1 \end{pmatrix}$
- Eigenvalues $\lambda_1 = -1$, $\lambda_2 = 3$.

$$M = \begin{pmatrix} 1 & -2 \\ -2 & 1 \end{pmatrix} = \underbrace{\frac{1}{\sqrt{2}} \begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix}}_{U} \underbrace{\begin{pmatrix} -1 & 0 \\ 0 & 3 \end{pmatrix}}_{\Lambda} \underbrace{\frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix}}_{U^T}$$

$$M \begin{pmatrix} 1 \\ 2 \end{pmatrix} = U \Lambda U^T \begin{pmatrix} 1 \\ 2 \end{pmatrix}$$



Principal component analysis revisited



Data vectors $X \in \mathbb{R}^d$

- Covariance matrix Σ is $d \times d$, symmetric.
- Eigenvalues $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_d$
  Eigenvectors $u_1, \ldots, u_d$.
- $u_1, \ldots, u_d$: another basis for data.
- Variance of X in direction u_i is λ_i.
- Projection to k dimensions:
  $x \mapsto (x \cdot u_1, \ldots, x \cdot u_k)$.

What is the covariance of the projected data?

Singular value decomposition (SVD)

For **symmetric** matrices, such as covariance matrices, we have seen:

- Results about existence of eigenvalues and eigenvectors
- The fact that the eigenvectors form an alternative basis
- The resulting spectral decomposition, which is used in PCA

But what about arbitrary matrices $M \in \mathbb{R}^{p \times q}$?

Any $p \times q$ matrix (say $p \leq q$) has a **singular value decomposition**:

$$M = \underbrace{\begin{pmatrix} \uparrow & & \uparrow \\ u_1 & \cdots & u_p \\ \downarrow & & \downarrow \end{pmatrix}}_{p \times p \text{ matrix } U} \underbrace{\begin{pmatrix} \sigma_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sigma_p \end{pmatrix}}_{p \times p \text{ matrix } \Lambda} \underbrace{\begin{pmatrix} \longleftarrow & v_1 & \longrightarrow \\ & \vdots & \\ \longleftarrow & v_p & \longrightarrow \end{pmatrix}}_{p \times q \text{ matrix } V^T}$$

- $u_1, \ldots, u_p$ are orthonormal vectors in $\mathbb{R}^p$
- $v_1, \ldots, v_p$ are orthonormal vectors in $\mathbb{R}^q$
- $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_p$ are **singular values**

Matrix approximation

We can **factor** any $p \times q$ matrix as $M = UW^T$:

$$M = \begin{pmatrix} \uparrow & & \uparrow \\ u_1 & \cdots & u_p \\ \downarrow & & \downarrow \end{pmatrix} \begin{pmatrix} \sigma_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sigma_p \end{pmatrix} \begin{pmatrix} \longleftarrow & v_1 & \longrightarrow \\ & \vdots & \\ \longleftarrow & v_p & \longrightarrow \end{pmatrix}$$

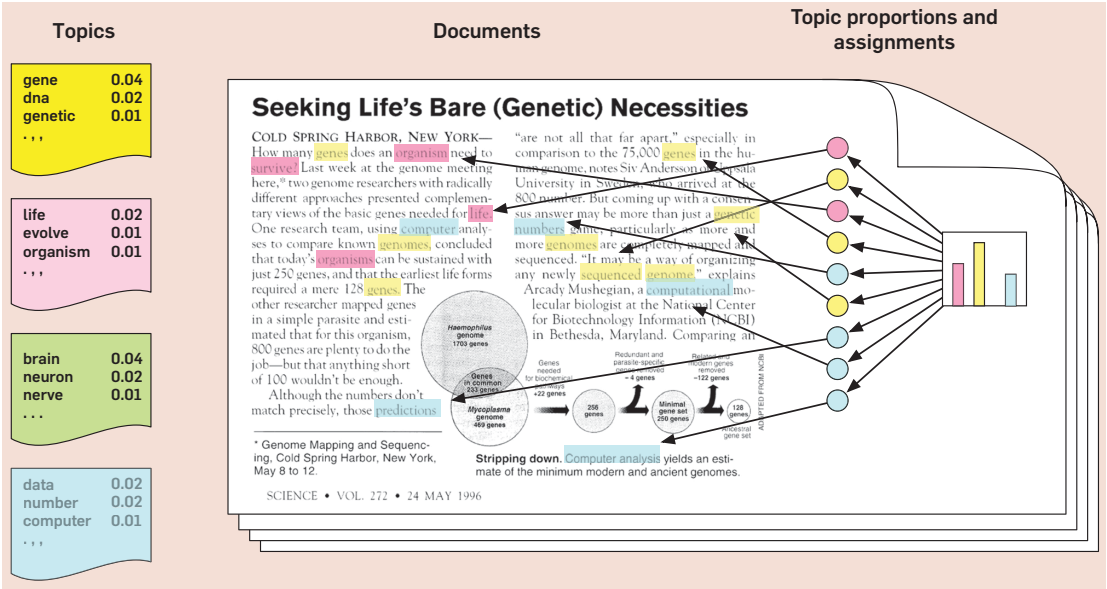
$$= \underbrace{\begin{pmatrix} \uparrow & & \uparrow \\ u_1 & \cdots & u_p \\ \downarrow & & \downarrow \end{pmatrix}}_{p \times p \text{ matrix } U} \underbrace{\begin{pmatrix} \longleftarrow & \sigma_1 v_1 & \longrightarrow \\ & \vdots & \\ \longleftarrow & \sigma_p v_p & \longrightarrow \end{pmatrix}}_{p \times q \text{ matrix } W^T}$$

A concise approximation to M: just take the first k columns of U and the first k rows of W^T, for $k < p$:

$$\widehat{M} = \underbrace{\begin{pmatrix} \uparrow & & \uparrow \\ u_1 & \cdots & u_k \\ \downarrow & & \downarrow \end{pmatrix}}_{p \times k} \underbrace{\begin{pmatrix} \longleftarrow & \sigma_1 v_1 & \longrightarrow \\ & \vdots & \\ \longleftarrow & \sigma_k v_k & \longrightarrow \end{pmatrix}}_{k \times q}$$

Example: topic modeling

Blei (2012):



Latent semantic indexing (LSI)

Given a large corpus of n documents:

- Fix a vocabulary, say of V words.
- Bag-of-words representation for documents: each document becomes a vector of length V, with one coordinate per word.
- The corpus is an $n \times V$ matrix, one row per document.

	cat	dog	house	boat	garden	...
Doc 1	4	1	1	0	2	
Doc 2	0	0	3	1	0	
Doc 3	0	1	3	0	0	
		⋮				

Let's find a concise approximation to this matrix M.

Latent semantic indexing, cont'd

Use SVD to get an approximation to M: for small k,

$$\underbrace{\begin{pmatrix} \longleftarrow \text{ doc } 1 \longrightarrow \\ \longleftarrow \text{ doc } 2 \longrightarrow \\ \longleftarrow \text{ doc } 3 \longrightarrow \\ \vdots \\ \longleftarrow \text{ doc } n \longrightarrow \end{pmatrix}}_{n \times V \text{ matrix } M} \approx \underbrace{\begin{pmatrix} \longleftarrow \theta_1 \longrightarrow \\ \longleftarrow \theta_2 \longrightarrow \\ \longleftarrow \theta_3 \longrightarrow \\ \vdots \\ \longleftarrow \theta_n \longrightarrow \end{pmatrix}}_{n \times k \text{ matrix } \Theta} \underbrace{\begin{pmatrix} \longleftarrow \Psi_1 \longrightarrow \\ \vdots \\ \longleftarrow \Psi_k \longrightarrow \end{pmatrix}}_{k \times V \text{ matrix } \Psi}$$

Think of this as a *topic model* with k topics.

- Ψ_j is a vector of length V describing topic j: coefficient Ψ_{jw} is large if word w appears often in that topic.
- Each document is a combination of topics: θ_{ij} is the weight of topic j in document i.

Document i originally represented by ith row of M, a vector in $\mathbb{R}^V$.
Can instead use $\theta_i \in \mathbb{R}^k$, a more concise "semantic" representation.

The rank of a matrix

Suppose we want to approximate a matrix M by a simpler matrix $\widehat{M}$. What is a suitable notion of "simple"?

- Let's say M and $\widehat{M}$ are $p \times q$, where $p \leq q$.
- Treat each row of $\widehat{M}$ as a data point in $\mathbb{R}^q$.
- We can think of the data as "simple" if it actually lies in a low-dimensional subspace.
- If the rows lie in k-dimensional subspace, we say that $\widehat{M}$ has **rank** k.

The **rank** of a matrix is the number of linearly independent rows.

Low-rank approximation: given $M \in \mathbb{R}^{p \times q}$ and an integer k, find the matrix $\widehat{M} \in \mathbb{R}^{p \times q}$ that is the best rank-k approximation to M.

That is, find $\widehat{M}$ so that

- $\widehat{M}$ has rank $\leq k$
- The approximation error $\sum_{i,j}(M_{ij} - \widehat{M}_{ij})^2$ is minimized.

We can get $\widehat{M}$ directly from the singular value decomposition of M.

Low-rank approximation

Recall: Singular value decomposition of $p \times q$ matrix M (with $p \leq q$):

$$M = \begin{pmatrix} \uparrow & & \uparrow \\ u_1 & \cdots & u_p \\ \downarrow & & \downarrow \end{pmatrix} \begin{pmatrix} \sigma_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sigma_p \end{pmatrix} \begin{pmatrix} \longleftarrow & v_1 & \longrightarrow \\ & \vdots & \\ \longleftarrow & v_p & \longrightarrow \end{pmatrix}$$

- $u_1, \ldots, u_p$ is an orthonormal basis of $\mathbb{R}^p$
- $v_1, \ldots, v_q$ is an orthonormal basis of $\mathbb{R}^q$
- $\sigma_1 \geq \cdots \geq \sigma_p$ are **singular values**

The **best rank-k approximation** to M, for any $k \leq p$, is then

$$\widehat{M} = \underbrace{\begin{pmatrix} \uparrow & & \uparrow \\ u_1 & \cdots & u_k \\ \downarrow & & \downarrow \end{pmatrix}}_{p \times k} \underbrace{\begin{pmatrix} \sigma_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sigma_k \end{pmatrix}}_{k \times k} \underbrace{\begin{pmatrix} \longleftarrow & v_1 & \longrightarrow \\ & \vdots & \\ \longleftarrow & v_k & \longrightarrow \end{pmatrix}}_{k \times q}$$

Example: Collaborative filtering

Details and images from Koren, Bell, Volinksy (2009).

Recommender systems: matching customers with products.

- Given: data on prior purchases/interests of users
- Recommend: further products of interest

Prototypical example: Netflix.

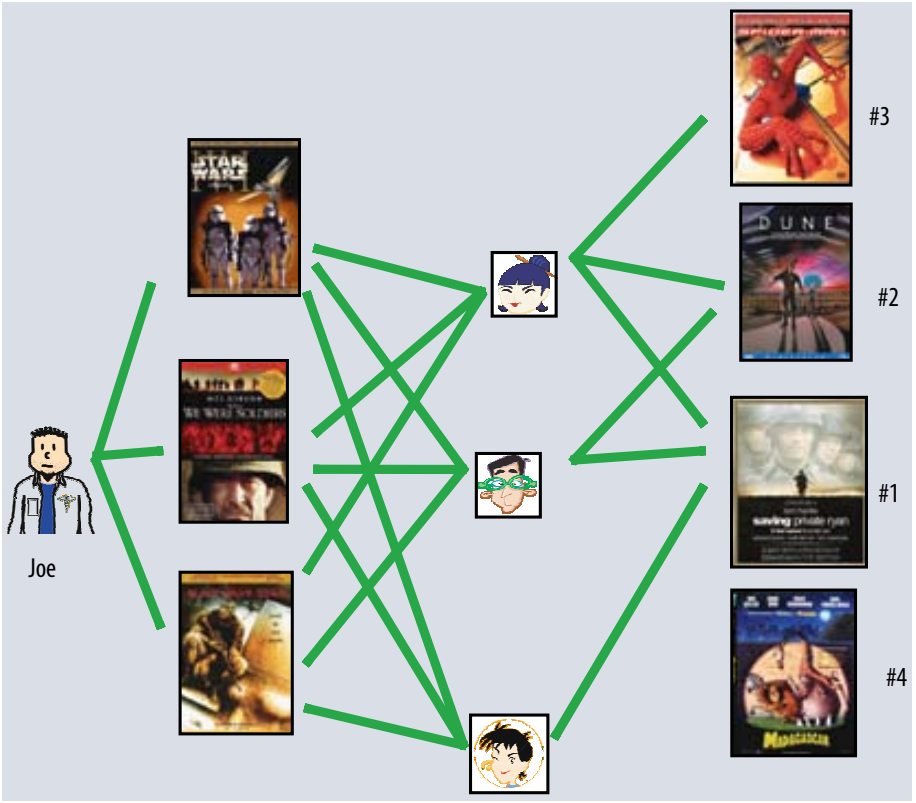
A successful approach: **collaborative filtering**.

- Model dependencies between different products, and between different users.
- Can give reasonable recommendations to a relatively new user.

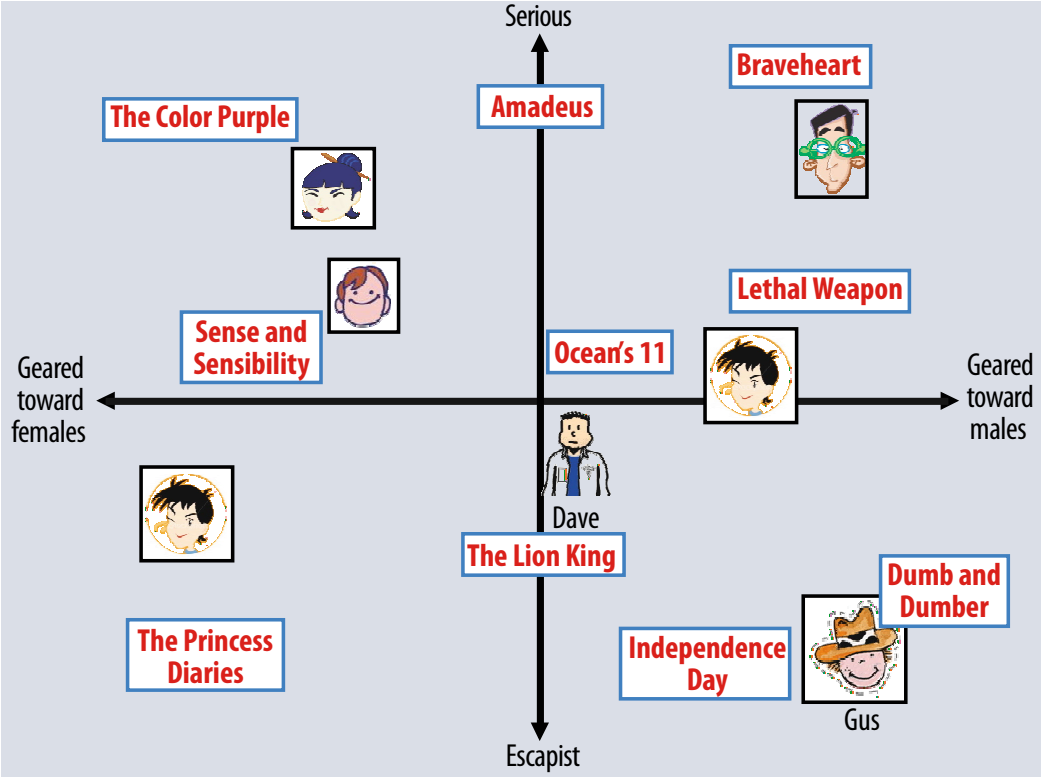
Two strategies for collaborative filtering:

- Neighborhood methods
- Latent factor methods

Neighborhood methods



Latent factor methods



The matrix factorization approach

User ratings are assembled in a large matrix M:

	Star Wars	Matrix	Casablanca	Camelot	Godfather	...
User 1	5	5	2	0	0	
User 2	0	0	3	4	5	
User 3	0	0	5	0	0	
		⋮				

- Not rated = 0, otherwise scores 1-5.
- For n users and p movies, this has size $n \times p$.
- Most of the entries are unavailable, and we'd like to predict these.

Idea: Find the best low-rank approximation of M, and use it to fill in the missing entries.

User and movie factors

Best rank-k approximation is of the form $M \approx UW^T$:

$$\underbrace{\begin{pmatrix} \longleftarrow \text{user 1} \longrightarrow \\ \longleftarrow \text{user 2} \longrightarrow \\ \longleftarrow \text{user 3} \longrightarrow \\ \vdots \\ \longleftarrow \text{user } n \longrightarrow \end{pmatrix}}_{n \times p \text{ matrix } M} \approx \underbrace{\begin{pmatrix} \longleftarrow u_1 \longrightarrow \\ \longleftarrow u_2 \longrightarrow \\ \longleftarrow u_3 \longrightarrow \\ \vdots \\ \longleftarrow u_n \longrightarrow \end{pmatrix}}_{n \times k \text{ matrix } U} \underbrace{\begin{pmatrix} \uparrow & \uparrow & & \uparrow \\ w_1 & w_2 & \cdots & w_p \\ \downarrow & \downarrow & & \downarrow \end{pmatrix}}_{k \times p \text{ matrix } W^T}$$

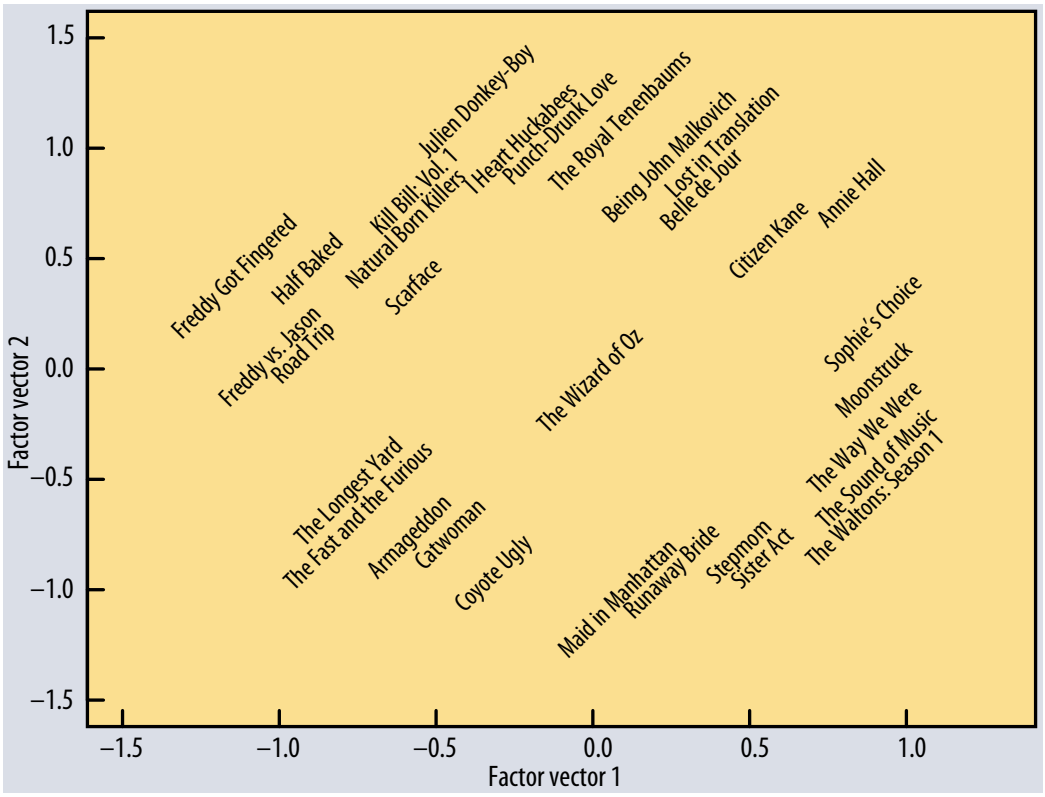
Thus user i's rating of movie j is approximated as

$$M_{ij} \approx u_i \cdot w_j$$

This "latent" representation embeds users and movies within the same k-dimensional space:

- Represent ith user by $u_i \in \mathbb{R}^k$
- Represent jth movie by $w_j \in \mathbb{R}^k$

Top two Netflix factors



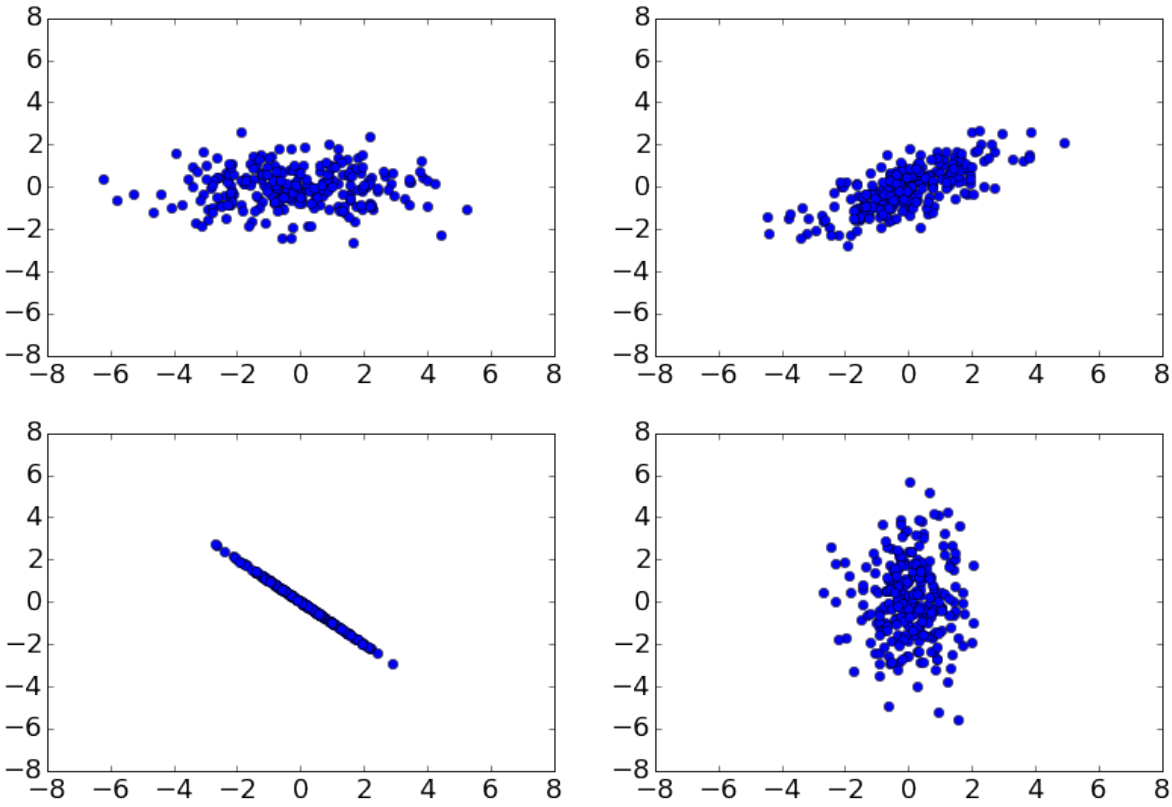
DSE 210: Probability and statistics **Winter 2018**

Worksheet 10 — PCA and SVD

1. Is the following set of vectors an orthonormal basis of $\mathbb{R}^3$? Explain why or why not.

$$\begin{pmatrix} 3 \\ 4 \\ 0 \end{pmatrix}, \begin{pmatrix} 4 \\ -3 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

2. The following four figures show different 2-dimensional data sets. In each case, make a rough sketch of an ellipsoidal contour of the covariance matrix and indicate the directions of the first and second eigenvectors (mark which is which).



3. Let $u_1, u_2 \in \mathbb{R}^d$ be two vectors with $\|u_1\| = \|u_2\| = 1$ and $u_1 \cdot u_2 = 0$. Define

$$U = \begin{pmatrix} \uparrow & \uparrow \\ u_1 & u_2 \\ \downarrow & \downarrow \end{pmatrix}$$

(a) What are the dimensions of each of the following?

- U
- U^T
- UU^T
- $u_1 u_1^T$

(b) What are the differences, if any, between the following four mappings?

- $x \mapsto (u_1 \cdot x, u_2 \cdot x)$
- $x \mapsto (u_1 \cdot x)u_1 + (u_2 \cdot x)u_2$
- $x \mapsto U^T x$
- $x \mapsto UU^T x$

4. Recall the *animals with attributes* data set from Worksheet 9, which has information about 50 animals, each represented as a vector in $\mathbb{R}^{85}$.

   We would like to visualize these animals in 2-d. Show how to do this with a PCA projection from $\mathbb{R}^{85}$ to $\mathbb{R}^2$. Show the position of each animal, and label them with their names. (Remember from Worksheet 9 how to enlarge the figure. This time you might want to ask for size 10,10.)

   Does this *embedding* seem sensible to you?

5. In lecture, we looked at the effect of projecting the MNIST data set of handwritten digits to lower-dimension: from 784 to 200, 150, 100, 50 dimensions. We found that the reconstruction error was fairly low for 150-dimensional projections, but noticeable for 50-dimensional projections.

   We now investigate these issues further.

   (a) Let $X \in \mathbb{R}^d$ have covariance matrix Σ. Suppose the eigenvalues of Σ are $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_d$, and suppose the corresponding eigenvectors are $u_1, u_2, \ldots, u_d$. Then it can be shown that X has an overall variance of $\lambda_1 + \cdots + \lambda_d$, and that when X is projected onto the top k eigenvectors, the residual variance (the information that gets lost) is $\lambda_{k+1} + \cdots + \lambda_d$. Therefore, for this projection, the fraction of lost information, intuitively speaking, is

   $$F(k) = \frac{\lambda_{k+1} + \cdots + \lambda_d}{\lambda_1 + \cdots + \lambda_d}$$

   Compute these fractions for the MNIST data set, for $k = 200, 150, 100, 50, 25$.

   (b) Suppose we are allowed a different projection for each digit. We would then expect that we can project to an even lower dimension while maintaining roughly the same amount of information. Test whether this is true as follows: for each digit $j = 0, 1, 2, \ldots, 9$,

   - Obtain the PCA projection to k dimensions, for $k = 200, 150, 100, 50, 25$.
   - Compute the fraction $F_j(k)$, for each such value of k.
   - Pick a random instance of the digit. Show the original digit as well as its reconstruction at each of these five values of k. (Note: the original images have pixel values in the range 0-255, but this might not be true of the reconstructions; therefore, you may need to clip the reconstructed pixels to lie within this range before displaying the image.)

   Show all the fractions $F_j(k)$ in a table. Which digit seems to be the most amenable to low-dimensional projection?