Linear Algebra Methods for Data Mining

Saara Hyvönen, Saara.Hyvonen@cs.helsinki.fi

Spring 2007

The Singular Value Decomposition (SVD) continued

The Singular Value Decomposition

- Any $m \times n$ matrix $\mathbf{A}$, with $m \geq n$, can be factorized

$$\mathbf{A} = \mathbf{U} \begin{pmatrix} \mathbf{\Sigma} \\ \mathbf{0} \end{pmatrix} \mathbf{V}^T,$$

  where $\mathbf{U} \in \mathbb{R}^{m \times m}$ and $\mathbf{V} \in \mathbb{R}^{n \times n}$ are orthogonal,
  and $\mathbf{\Sigma} \in \mathbb{R}^{n \times n}$ is diagonal:

$$\mathbf{\Sigma} = \text{diag}(\sigma_1, \sigma_2, ..., \sigma_n), \quad \sigma_1 \geq \sigma_2 \geq ... \geq \sigma_n \geq 0.$$

- "Skinny version": $\mathbf{A} = \mathbf{U}_1 \mathbf{\Sigma} \mathbf{V}^T$, $\mathbf{U}_1 \in \mathbb{R}^{m \times n}$.

Matrix approximation

Theorem. Let $\mathbf{U}_k = (\mathbf{u}_1\ \mathbf{u}_2\ \ldots\ \mathbf{u}_k)$, $\mathbf{V}_k = (\mathbf{v}_1\ \mathbf{v}_2\ \ldots\ \mathbf{v}_k)$ and $\mathbf{\Sigma}_k = \mathrm{diag}(\sigma_1, \sigma_2, \ldots, \sigma_k)$, and define

$$\mathbf{A}_k = \mathbf{U}_k \mathbf{\Sigma}_k \mathbf{V}_k^T.$$

Then

$$\min_{\mathrm{rank}(\mathbf{B}) \le k} \|\mathbf{A} - \mathbf{B}\|_2 = \|\mathbf{A} - \mathbf{A}_k\|_2 = \sigma_{k+1}.$$

E.g. the best approximation of rank k for the matrix $\mathbf{A}$ is

$$\mathbf{A}_k = \mathbf{U}_k \mathbf{\Sigma}_k \mathbf{V}_k^T.$$

Consequences

- The best rank one approximation of $\mathbf{A}$ is

$$\mathbf{A}_k = \mathbf{U}_1 \mathbf{\Sigma}_1 \mathbf{V}_1^T.$$

- Assume $\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_j > \sigma_{j+1} = 0 = \sigma_{j+2} = ... = \sigma_m$. Then

$$\min_{\text{rank}(\mathbf{B}) \leq j} \|\mathbf{A} - \mathbf{B}\|_2 = \|\mathbf{A} - \mathbf{A}_j\|_2 = \sigma_{j+1} = 0.$$

So the rank of $\mathbf{A}$ is the number of nonzero singular values of $\mathbf{A}$.

Perturbation theory

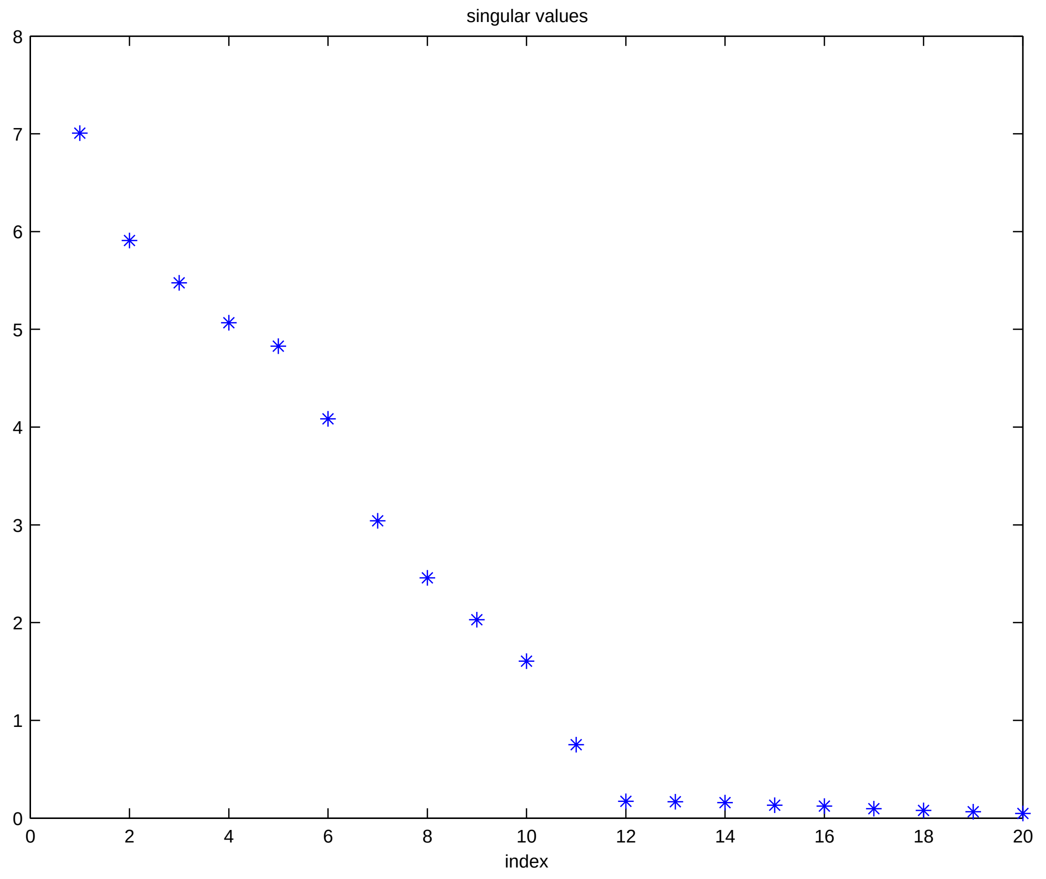
Theorem. If $\mathbf{A}$ and $\mathbf{A}+\mathbf{E}$ are in $\mathbb{R}^{m\times n}$ with $m \geq n$, then for $k = 1...n$

$$|\sigma_k(\mathbf{A}+\mathbf{E}) - \sigma_k(\mathbf{A})| \leq \sigma_1(\mathbf{E}) = \|\mathbf{E}\|_2.$$

Proof. Omitted.

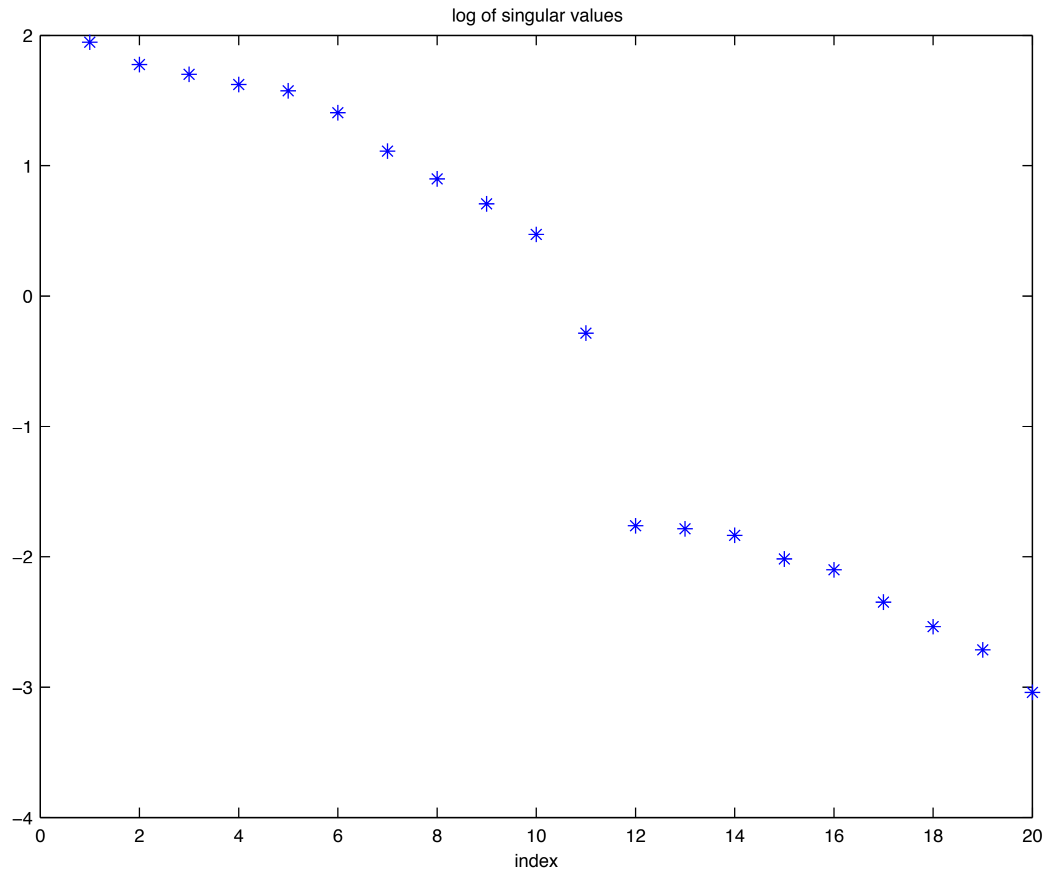
Think of $\mathbf{E}$ as added noise.

Example: low rank matrix plus noise



Example: low rank matrix plus noise

- Assume $\mathbf{A}$ is a low rank matrix plus noise: $\mathbf{A} = \mathbf{A}_* + \mathbf{N}$, where $\|\mathbf{N}\| \ll \|\mathbf{A}_*\|$.
- "Correct" rank can be estimated by looking at singular values: when choosing a good k, look for gaps in singular values!
- When $\|\mathbf{N}\|$ is small, the number of larger singular values is often referred to as the *numerical rank* of $\mathbf{A}$.
- The noise can be removed by estimating the numerical rank k from the singular values, and approximating $\mathbf{A}$ by the truncated SVD $\mathbf{U}_k\mathbf{\Sigma}_k\mathbf{V}_k^T$.


log of singular values
2
1
0
−1
−2
−3
−4
0
2
4
6
8
10
12
14
16
18
20
index

- In the figure, there is a gap between the $11th$ and $12th$ singular values.
- Estimate numerical rank to be 11.
- So to remove noise replace $\mathbf{A}$ by $\mathbf{A}_k = \mathbf{U}_k \mathbf{\Sigma}_k \mathbf{V}_k^T$, where $k = 11$.

Eigenvalue decomposition vs. SVD

- For symmetric $\mathbf{A}$ the singular value decomposition is closely related to the eigendecomposition: $\mathbf{A} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$, $\mathbf{U}$ and $\mathbf{\Lambda}$ eigenvectors and eigenvalues.

- Computation of both the eigendecomposition and the SVD follow the same pattern.

Computation of the eigenvalue decomposition

1. Use Givens to transform $\mathbf{A}$ into tridiagonal form.

2. Use QR iteration with Wilkinson shift μ to transform tridiagonal form to diagonal form: Repeat until converged,

   (i) $\mathbf{QR} = \mathbf{T}_k - \mu_k \mathbf{I}$
   (ii) $\mathbf{T}_{k+1} = \mathbf{RQ} + \mu_k \mathbf{I}$

   The shift parameter μ is the eigenvalue of the 2×2 submatrix in the lower right corner that is closest to the element $\mathbf{A}_{n,n}$ in the lower right corner.

- Once an eigenvalue is found, forget it, reduce (deflate) the problem, and go back to step 2.

Example

```
A=     1     2     3     4     5
       2     1     1     0     1
       3     1     2     1     3
       4     0     1     1     1
       5     1     3     1     3

%After 1st step (with Givens to tridiagonal form:)

A=    1.0000    7.3485   -0.0000   -0.0000    0.0000
      7.3485    5.5370    2.0066    0.0000   -0.0000
     -0.0000    2.0066    0.8979    0.6300    0.0000
     -0.0000    0.0000    0.6300    0.1894    0.5057
      0.0000   -0.0000    0.0000    0.5057    0.3757
```

```
%intermediate results of 4 QR iteration steps:

    5.9399    7.4855    0.0000   -0.0000   -0.0000
    7.4855    0.5901    0.1715   -0.0000    0.0000
    0.0000    0.1715    0.4214    0.8535   -0.0000
    0.0000   -0.0000    0.8535    0.4877   -0.1693
   -0.0000    0.0000   -0.0000   -0.1693    0.5610


    9.3851    5.0747   -0.0000    0.0000   -0.0000
    5.0747   -2.8580    0.0248    0.0000   -0.0000
   -0.0000    0.0248   -0.0138    0.7308    0.0000
    0.0000    0.0000    0.7308    0.9303    0.0319
    0.0000   -0.0000   -0.0000    0.0319    0.5565
   10.7304    2.7336    0.0000    0.0000   -0.0000
    2.7336   -4.2034    0.0042   -0.0000    0.0000
    0.0000    0.0042   -0.1329    0.6387   -0.0000
    0.0000   -0.0000    0.6387    1.0502    0.0001
   -0.0000    0.0000    0.0000    0.0001    0.5558

   11.0949    1.3766   -0.0000    0.0000   -0.0000
    1.3766   -4.5680    0.0007    0.0000   -0.0000
   -0.0000    0.0007   -0.2227    0.5419    0.0000
    0.0000    0.0000    0.5419    1.1400    0.0000
    0.0000   -0.0000   -0.0000    0.0000    0.5558

eig(Aorig)=       -4.6881   -0.4119    0.5558    1.3293   11.2150
```

Flop counts

- Eigenvalues only: about $4n^3/3$.
- Accumulation of the orthogonal transformations to compute the matrix of eigenvectors: about $9n^3$ more.

How about computing the SVD?

- We now know how to compute the eigendecomposition.
- Couldn't we use this to compute the eigenvalues of $\mathbf{A}^T\mathbf{A}$ to get the singular values?
- Well, yes... and NO. This is not the way to do it.
- Forming $\mathbf{A}^T\mathbf{A}$ can lead to loss of information.

Computing the SVD

1. Use Householder transformations to transform $\mathbf{A}$ into bidiagonal form $\mathbf{B}$. Now $\mathbf{B}^T\mathbf{B}$ is tridiagonal.

2. Use QR iteration with Wilkinson shift μ to transform tridiagonal $\mathbf{B}$ to diagonal form (without forming $\mathbf{B}^T\mathbf{B}$ implicitly!)

- Can be computed in $6mn^2 + 20n^3$ flops.

- Efficiently implemented everywhere.

Sparse matrices

- In many applications only a small number of entries are nonzero (e.g. term-document matrices).
- Iterative methods frequently used in solving sparse problems.
- This is because e.g. transformation to tridiagonal form would destroy sparsity, which leads to excessive storage requirements.
- Also computational complexity might be prohibitively high when dimension of data matrix is very large.

The Singular Value Decomposition

- Any $m \times n$ matrix $\mathbf{A}$, with $m \geq n$, can be factorized

$$\mathbf{A} = \mathbf{U} \begin{pmatrix} \mathbf{\Sigma} \\ \mathbf{0} \end{pmatrix} \mathbf{V}^T,$$

where $\mathbf{U} \in \mathbb{R}^{m \times m}$ and $\mathbf{V} \in \mathbb{R}^{n \times n}$ are orthogonal,
and $\mathbf{\Sigma} \in \mathbb{R}^{n \times n}$ is diagonal:

$$\mathbf{\Sigma} = \text{diag}(\sigma_1, \sigma_2, ..., \sigma_n), \quad \sigma_1 \geq \sigma_2 \geq ... \geq \sigma_n \geq 0.$$

- "Skinny version": $\mathbf{A} = \mathbf{U}_1 \mathbf{\Sigma} \mathbf{V}^T$, $\mathbf{U}_1 \in \mathbb{R}^{m \times n}$.

Facts about SVD

- Equivalent forms of SVD:

$$\mathbf{A}^T\mathbf{A}\mathbf{v}_j = \sigma_j^2\mathbf{v}_j, \quad \mathbf{A}\mathbf{A}^T\mathbf{u}_j = \sigma_j^2\mathbf{u}_j,$$

  where $\mathbf{u}_j$ and $\mathbf{v}_j$ are the columns of $\mathbf{U}$ and $\mathbf{V}$ respectively.

- Let $\mathbf{U}_k$, $\mathbf{V}_k$ and $\mathbf{\Sigma}_k$, be matrices with the k first singular vectors and values, and define

$$\mathbf{A}_k = \mathbf{U}_k\mathbf{\Sigma}_k\mathbf{V}_k^T.$$

  Then

$$\min_{\text{rank}(\mathbf{B})\leq k} \|\mathbf{A}-\mathbf{B}\|_2 = \|\mathbf{A}-\mathbf{A}_k\|_2 = \sigma_{k+1}.$$

Principal components analysis

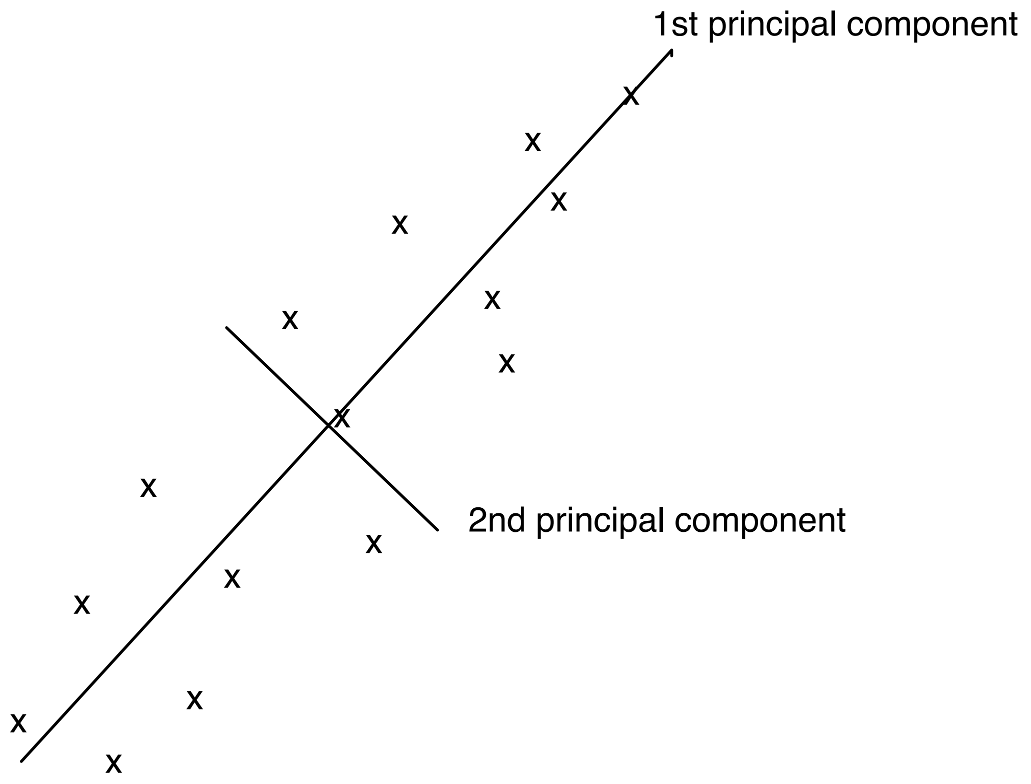
- Idea: look for such a direction that the data projected onto it has maximal variance.

- When found, continue by seeking the next direction, which is orthogonal to this (i.e. uncorrelated), and which explains as much of the remaining variance in the data as possible.

- Ergo: we are seeking linear combinations of the original variables.

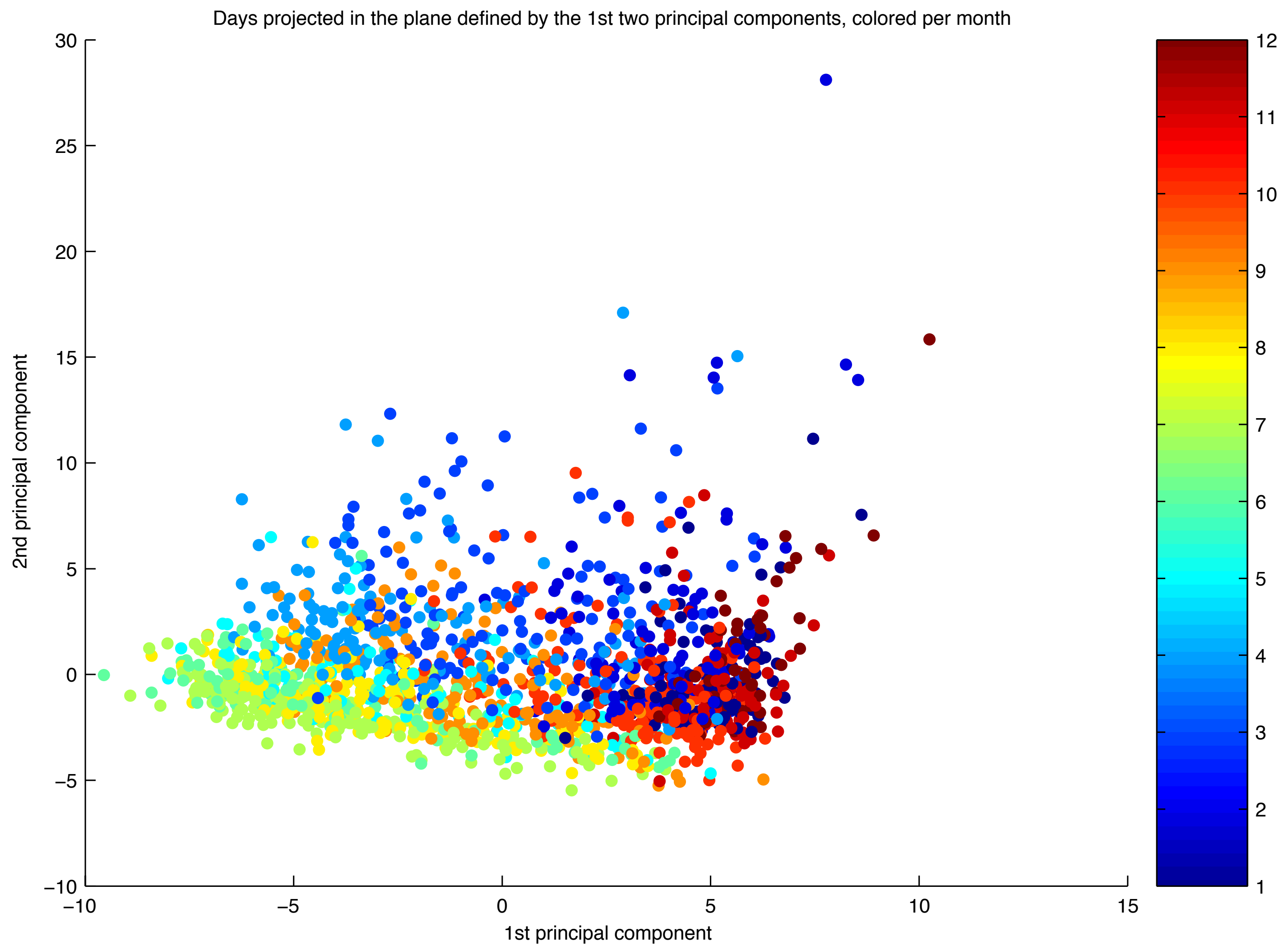
- If we are lucky, we can find a few such linear combinations, or directions, or (principal) components, which describe the data fairly accurately.

- The aim is to capture the intrinsic variability in the data.


1st principal component
2nd principal component

Example: Atmospheric data

- Data: 1500 days, and for each day, we have the mean and the std of around 30 measured variables (temperature, wind speed and direction, rain fall, UV-A radiation, concentration of CO2 etc.)
- Therefore, our data matrix is 1500×60.
- Visualizing things in a 60-dimensional space is challenging!
- Instead, do PCA, and project days onto the plane defined by the first two principal components.



Days projected in the plane defined by the 1st two principal components, colored per month
2nd principal component
1st principal component
30
25
20
15
10
5
0
−5
−10
−10
−5
0
5
10
15
12
11
10
9
8
7
6
5
4
3
2
1

Example: spatial data analysis

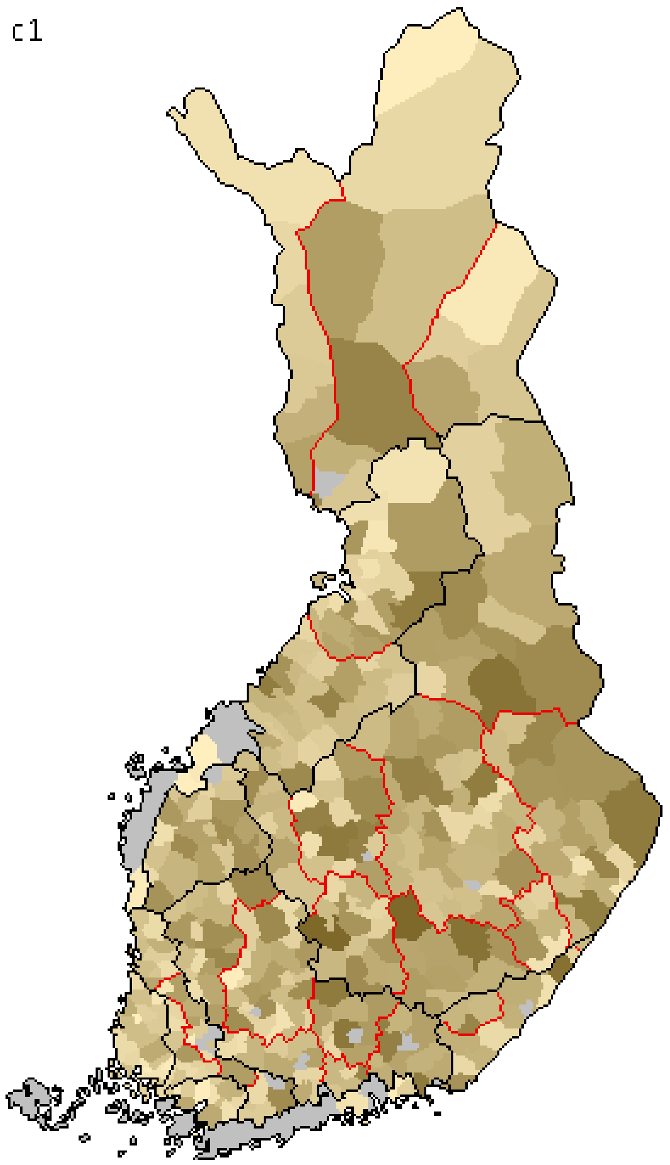
- Data: 9000 dialect words, 500 counties.
- Word-county matrix $\mathbf{A}$:

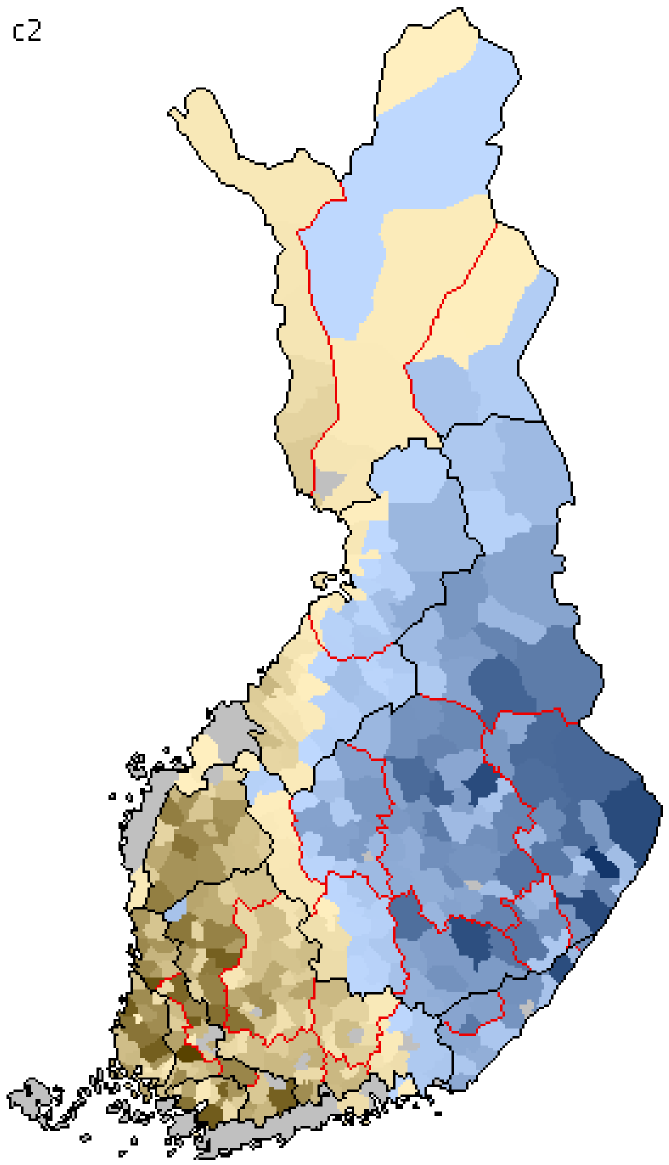
$$\mathbf{A}(i,j) = \begin{cases} 1 & \text{if word i appears in county j} \\ 0 & \text{otherwise.} \end{cases}$$

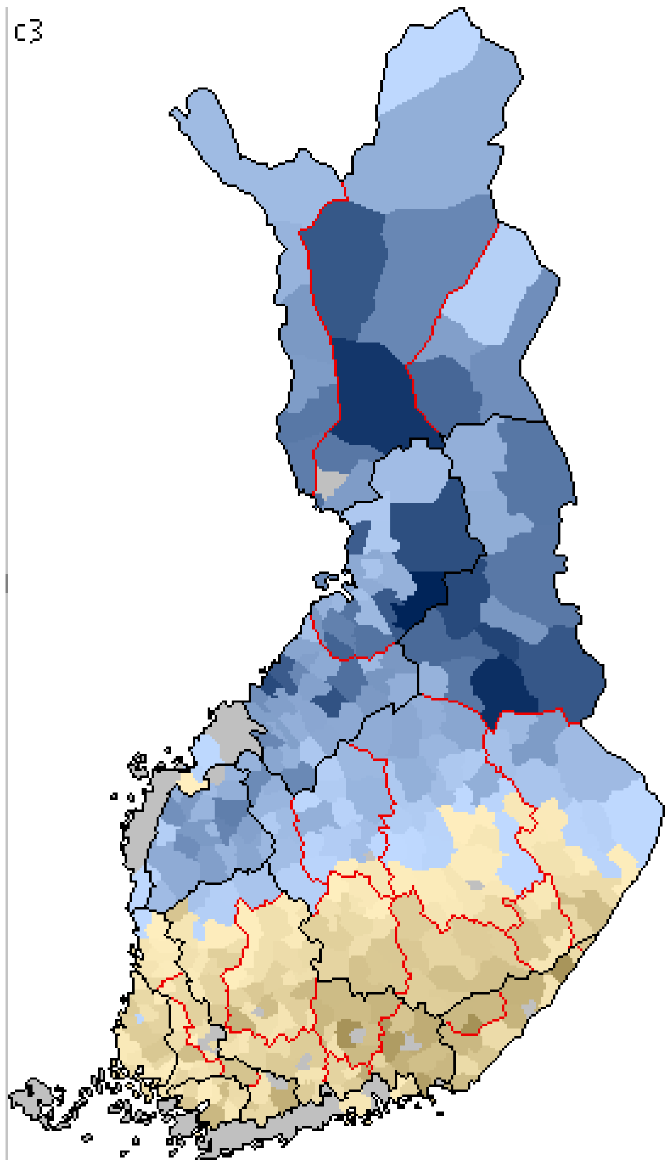
- Apply PCA to this.

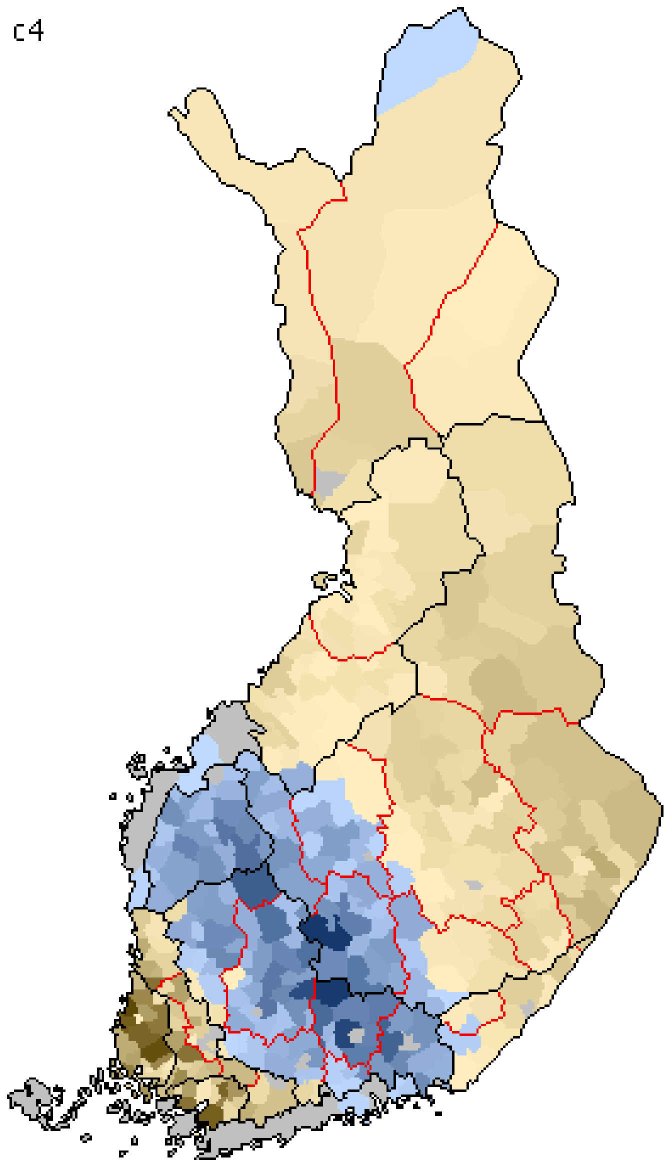
Results obtained by PCA

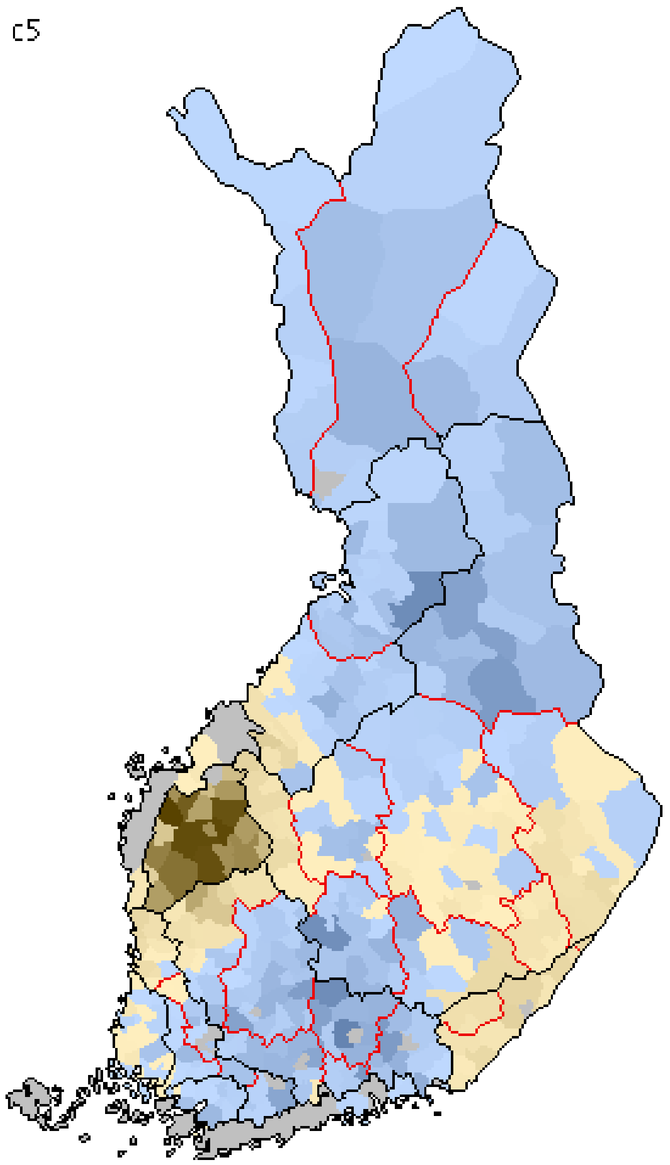
- Data points: words; variables: counties.
- Each principal component tells which counties explain the most significant part of the variation left in the data.
- The first principal component is essentially just the number of words in each county!
- After this, geographical structure of principal components is apparent.
- Note: PCA knows nothing of the geography of the counties.

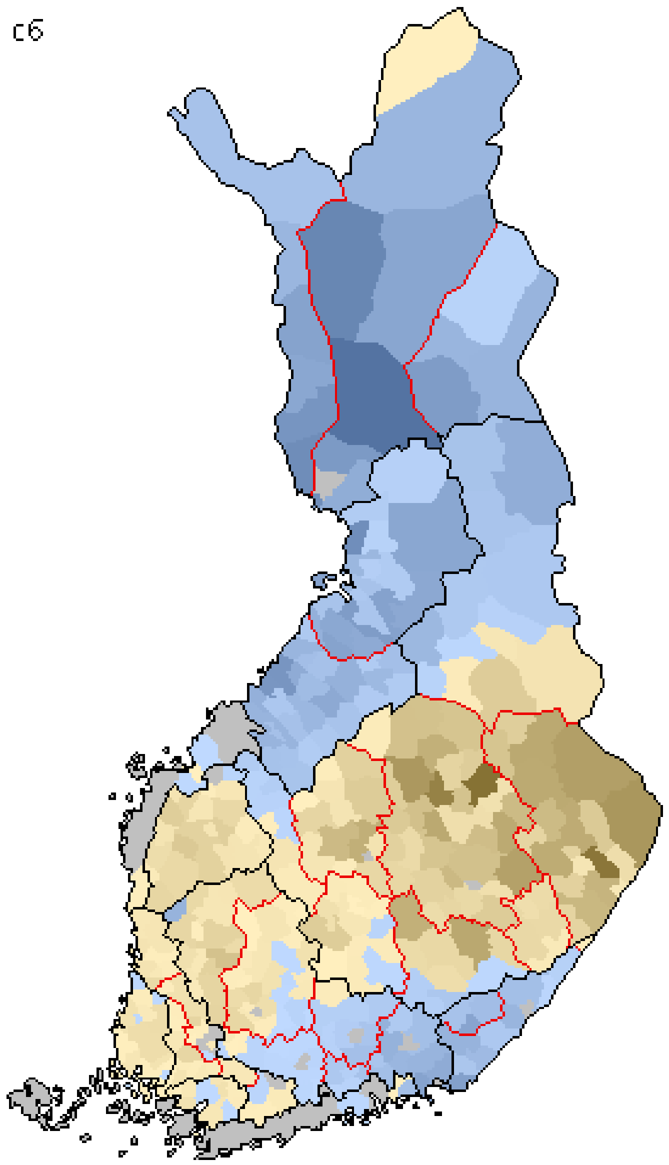

c1


c2


c3


c4


c5


c6

PCA=SVD

Let $\mathbf{A}$ be a $n \times m$ data matrix in which the rows represent the cases.

Each row is a data vector, each column represents a variable. (Note: usually the roles of rows and columns are the other way around!)

$\mathbf{A}$ is centered: the estimated mean is subtracted from each column, so each column has zero mean.

Let $\mathbf{w}$ be the $m \times 1$ column vector of (unknown) projection weights that result in the largest variance when the data $\mathbf{A}$ is projected along $\mathbf{w}$.

Require $\mathbf{w}^T\mathbf{w} = 1$. Projection of $\mathbf{a}$ onto $\mathbf{w}$ is $\mathbf{w}^T\mathbf{a} = \sum_{j=1}^{m} a_j w_j$.

Projection of data along $\mathbf{w}$ is $\mathbf{A}\mathbf{w}$.

Projection of data along $\mathbf{w}$ is $\mathbf{Aw}$.

Variance:

$$\sigma_{\mathbf{w}}^2 = (\mathbf{Aw})^T(\mathbf{Aw}) = \mathbf{w}^T\mathbf{A}^T\mathbf{Aw} = \mathbf{w}^T\mathbf{Cw}$$

where $\mathbf{C} = \mathbf{A}^T\mathbf{A}$ is the covariance matrix of the data ($\mathbf{A}$ is centered!)

Task: maximize variance subject to constraint $\mathbf{w}^T\mathbf{w} = 1$.

Optimization problem: maximize

$$f = \mathbf{w}^T\mathbf{Cw} - \lambda(\mathbf{w}^T\mathbf{w} - 1),$$

λ is the Lagrange multiplier.

Optimization problem: maximize

$$f = \mathbf{w}^T\mathbf{C}\mathbf{w} - \lambda(\mathbf{w}^T\mathbf{w} - 1),$$

λ is the Lagrange multiplier.

Differentiating with respect to $\mathbf{w}$ yields

$$\frac{\partial f}{\partial \mathbf{w}} = 2\mathbf{C}\mathbf{w} - 2\lambda\mathbf{w} = 0$$

Eigenvalue equation: $\mathbf{C}\mathbf{w} = \lambda\mathbf{w}$, where $\mathbf{C} = \mathbf{A}^T\mathbf{A}$.

Solution: singular values and singular vectors of $\mathbf{A}$!!!

More precisely: the first principal component of $\mathbf{A}$ is exactly the first right singular vector $\mathbf{v}_1$ of $\mathbf{A}$.

The solution of our opimization problem is given by

$$\mathbf{w} = \mathbf{v}_1, \quad \lambda = \sigma_1^2,$$

where σ_1 and $\mathbf{v}_1$ are the first singular value and the corresponding right singular vector of $\mathbf{A}$, and

$$\mathbf{C}\mathbf{w} = \lambda \mathbf{w}.$$

Our interest was to maximize the variance, which is given by

$$\mathbf{w}^T \mathbf{C}\mathbf{w} = \mathbf{w}^T \lambda \mathbf{w} = \sigma_1^2 \mathbf{w}^T \mathbf{w} = \sigma_1^2,$$

so the singular value tells about the variance in the direction of the principal component.

What next?

Once the first principal component is found, we continue in the same fashion to look for the next one, which is orthogonal to (all) the principal component(s) already found.

The solutions are the right singular vectors $\mathbf{v}_k$ of $\mathbf{A}$, and the variance in each direction is given by the corresponding singular values σ_k.

How not to compute the PCA:

- In literature one frequently runs across PCA algorithms, which start by computing the covariance matrix $\mathbf{C} = \mathbf{A}^T\mathbf{A}$ of the centered data matrix $\mathbf{A}$, and computes the eigenvalues of this.
- But we already know that this is a bad idea!
  - The condition number of $\mathbf{A}^T\mathbf{A}$ is much larger than that of $\mathbf{A}$.
  - Loss of information.
  - For a sparse $\mathbf{A}$, $\mathbf{C} = \mathbf{A}^T\mathbf{A}$ is no longer sparse!

How to compute the PCA:

Data matrix $\mathbf{A}$, rows=data points, columns = variables (attributes, parameters).

1. Center the data by subtracting the mean of each column.
2. Compute the SVD of the centered matrix $\hat{\mathbf{A}}$ (or the k first singular values and vectors):
$$\hat{\mathbf{A}} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T.$$
3. The principal components are the columns of $\mathbf{V}$, the coordinates of the data in the basis defined by the principal components are $\mathbf{U}\mathbf{\Sigma}$.

Matlab code for PCA

```
%Data matrix A, columns:variables, rows: data points
%matlab function for computing the first k principal components of A.
function [pc,score]=pca(A,k);
[rows,cols]=size(A);
Ameans=repmat(mean(A,1),rows,1); %matrix, rows=means of columns
A=A-Ameans; %centering data
[U,S,V]=svds(A,k); %k is the number of pc:s desired
pc=V;
score=U*S; %now A=scores*pcs'+Ameans;
```

Note on Matlab

- PCA is coded in the statistics toolbox in matlab, BUT...
- DO NOT USE IT!!
- Why? We have so few statistics toolbox licenses, that we run out of them frequently! Better not waste scarce resources on this.
- http://www.cis.hut.fi/projects/somtoolbox

$\mathbf{A}$ vs $\mathbf{A}^T$

- Let $\mathbf{A}$ be a centered data matrix, and $\mathbf{A} = \mathbf{U\Sigma V}^T$.
- The principal components of $\mathbf{A}$ are $\mathbf{V}$, and the coordinates in the basis defined by the principal components are $\mathbf{U\Sigma}$.
- If $\mathbf{A} = \mathbf{U\Sigma V}^T$, then $\mathbf{A}^T = \mathbf{V\Sigma}^T\mathbf{U}^T$.
- So aren't the principal components of $\mathbf{A}^T$ given by $\mathbf{U}$ and the coordinates $\mathbf{V\Sigma}^T$? So the pc's of $\mathbf{A}$ are the new coordinates of $\mathbf{A}^T$ and vica versa, modulo a multiplication by the diagonal matrix $\mathbf{\Sigma}$ (or its inverse)?
- No.

And why not?

Because of the centering of the data!

In general it does not hold, that the transpose of a centered matrix is the same as the centered transpose:

$$(\mathbf{A} - \text{meansofcolumns}(\mathbf{A}))^T \neq (\mathbf{A}^T - \text{meansofcolumns}(\mathbf{A}^T))$$

In practice these are related in special cases.

Singular values tell about variance

The variance in the direction of the k^{th} principal component is given by the corresponding singular value: σ_k^2.

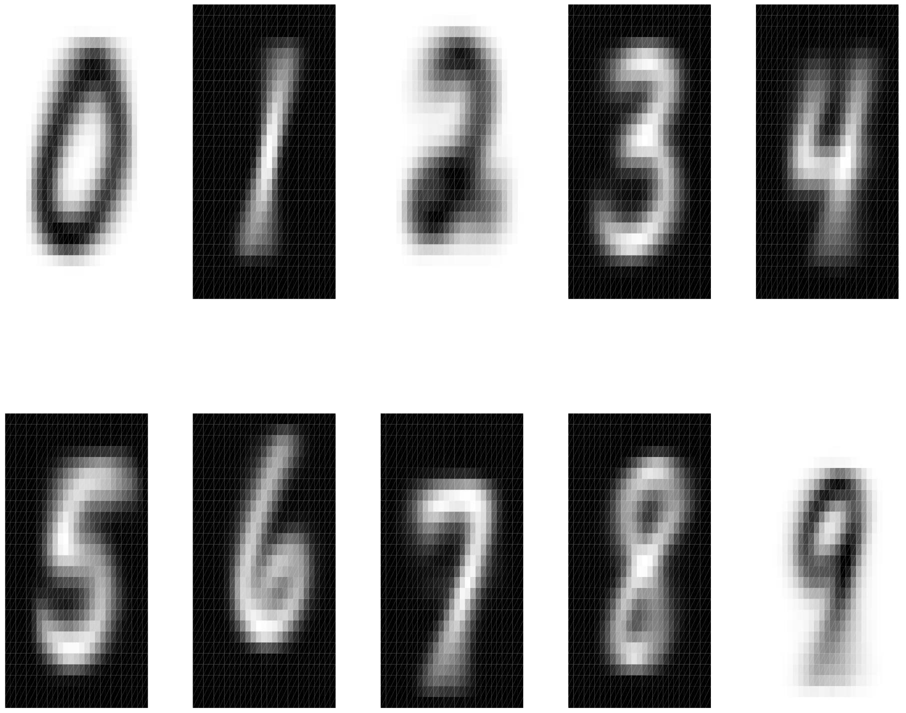
Singular values can be used to estimate how many principal components to keep.

Rule of thumb: keep enough to explain 85% of the variation:

$$\frac{\sum_{j=1}^{k} \sigma_j^2}{\sum_{j=1}^{n} \sigma_j^2} \approx 0.85.$$

Why talk about PCA?

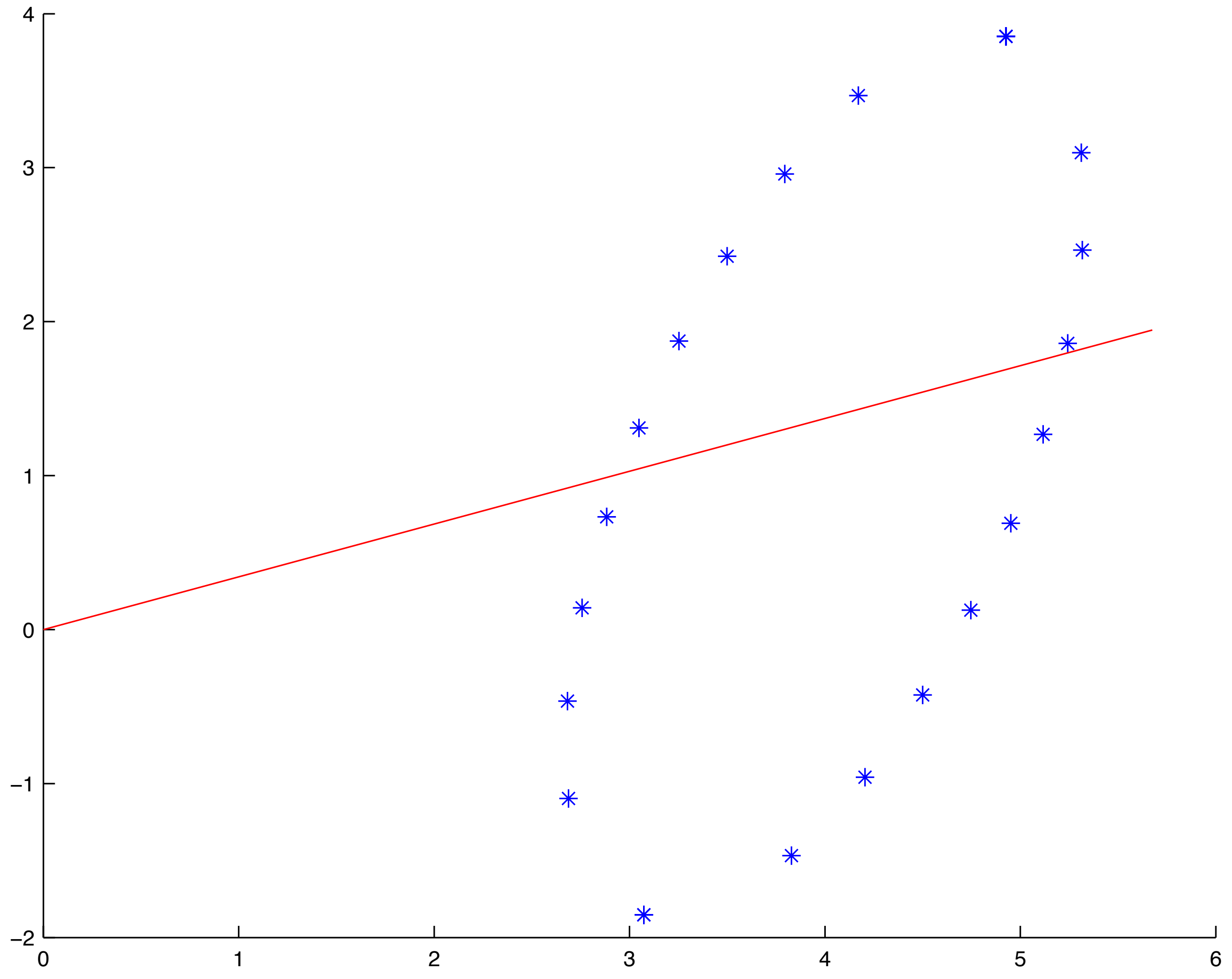
Why not just stick to SVD? Singular vectors=principal components:

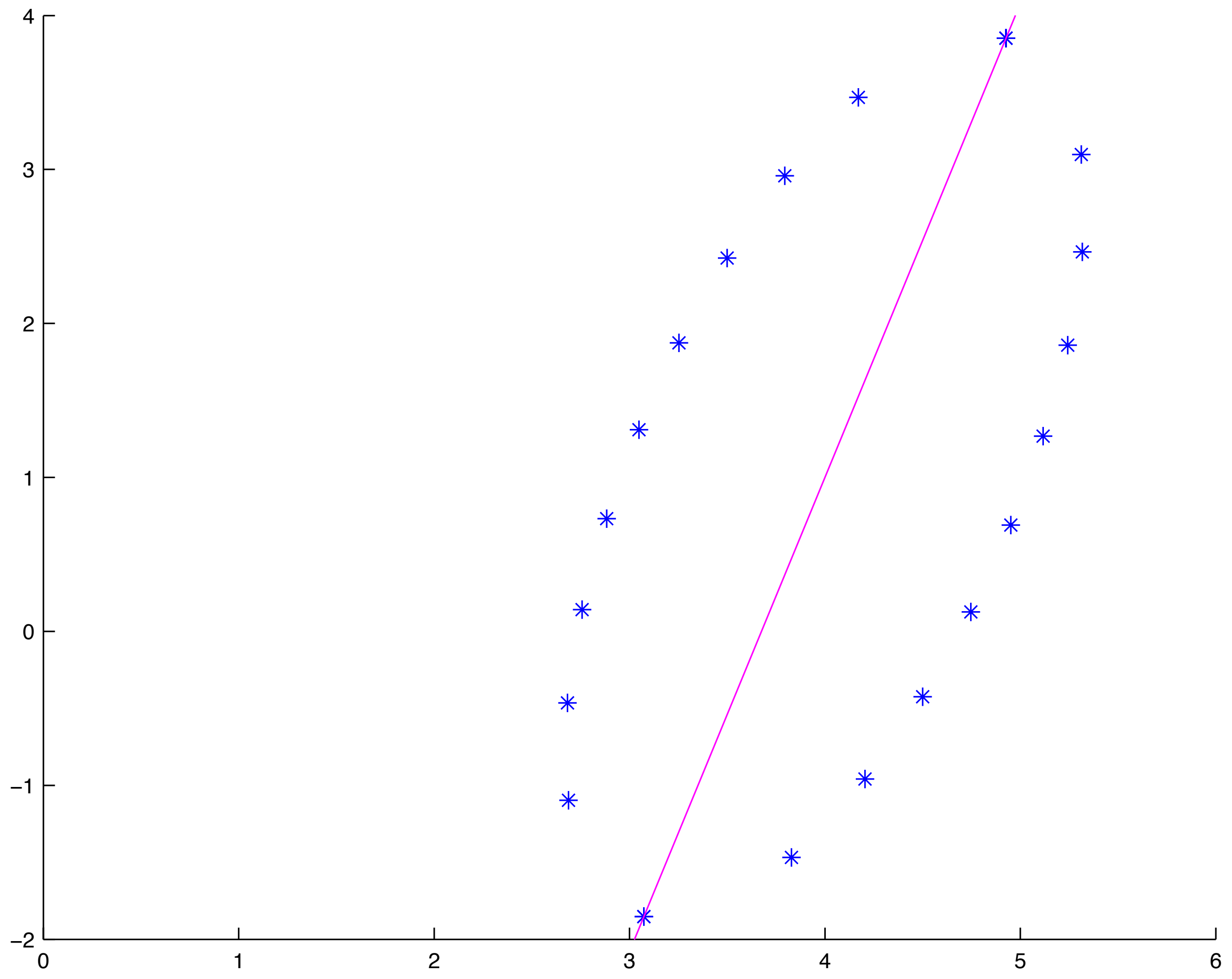


Centering is central

SVD will give vectors that go through the origin.

Centering makes sure that the origin is in the middle of the data set.


4
3
2
1
0
−1
−2
0
1
2
3
4
5
6



4
3
2
1
0
−1
−2
0
1
2
3
4
5
6

Summary: PCA

- PCA is SVD done on **centered** data.
- PCA looks for such a direction that the data projected onto it has maximal variance.
- When found, PCA continues by seeking the next direction, which is orthogonal to all the previously found directions, and which explains as much of the remaining variance in the data as possible.
- Principal components are uncorrelated.

PCA is useful for

- data exploration
- visualizing data
- compressing data
- outlier detection
- ratio rules

References

[1] Lars Eldén: Matrix Methods in Data Mining and Pattern Recognition, SIAM 2007.

[2] R. A. Horn and C. R. Johnson, Matrix Analysis, Cambridge University Press 1985.

[3] D. Hand, H. Mannila, P. Smyth, Principles of Data Mining, The MIT Press, 2001.