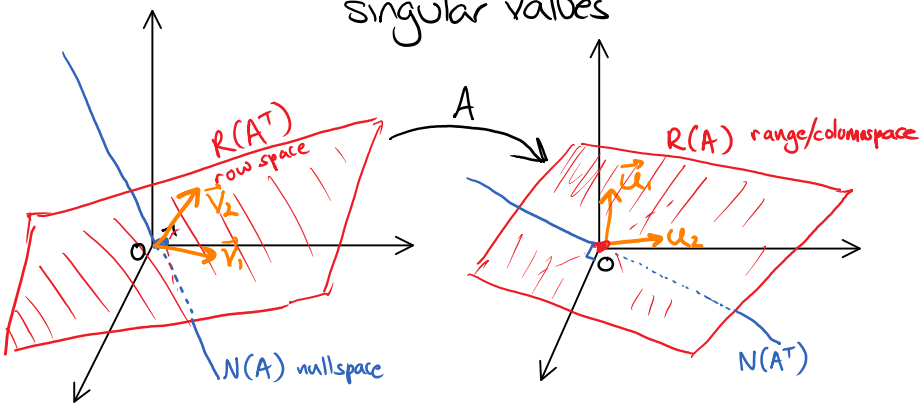
Lecture 18: Principal component analysis (PCA)

Low-Rank Matrices

$\text{Rank}(A) = \dim R(A) = \dim(\text{span of columns})$
$= \dim R(A^T) = \dim(\text{span of rows})$

SVD: $A = \sum_i \sigma_i \vec{u}_i \vec{v}_i^T$

$\text{Rank}(A) = \#$ of nonzero singular values



$\text{Rank}(A) + \dim N(A) = \text{number of columns}$

$\text{Rank}(A) + \dim N(A^T) = \text{number of rows}$

- $\text{Rank}(A) = 0 \iff A = \begin{pmatrix} \text{all 0} \\ \text{matrix} \end{pmatrix}$
- $\text{Rank}(A) = 1 \iff A = \vec{u}\vec{v}^T$ with $\vec{u}, \vec{v}$ nonzero

$A_{ij} = u_i \cdot v_j$ $\quad \begin{pmatrix} \vec{u} \end{pmatrix} \begin{pmatrix} \vec{v}^T \end{pmatrix}$

Ex: $\begin{pmatrix} 1 & 1 & 2 \\ 1 & 1 & 2 \\ 3 & 3 & 6 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 3 \end{pmatrix} (1 \;\; 1 \;\; 2)$

- $\text{Rank}(A) = 2 \qquad A = \underset{\neq 0}{\sigma_1} \vec{u}_1 \vec{v}_1^T + \underset{\neq 0}{\sigma_2} \vec{u}_2 \vec{v}_2^T$

$\text{Rank}(\vec{u}\vec{v}^T + \vec{w}\vec{x}^T) \leq 2$

A

$R(A) = \text{Span}\{\vec{u}, \vec{w}\}$
$R(A^T) = \text{Span}\{\vec{v}, \vec{x}\}$

- $\text{Rank}(A) = 3 \iff A = \vec{u}\vec{v}^T + \vec{w}\vec{x}^T + \vec{y}\vec{z}^T$ for some $\vec{u}, \vec{w}, \vec{y}$, $\vec{v}, \vec{x}, \vec{z}$

$\vdots$ and so on

Observe: $\text{Rank}(A+B) \leq \text{Rank}(A) + \text{Rank}(B)$

Observe: Low-rank matrices have fewer parameters.

$\text{Rank}(A) = k \Rightarrow A = \sum_{i=1}^{k} \vec{u}_i \vec{v}_i^T$ for some vectors

$= \begin{pmatrix} | & & | \\ \vec{u}_1 & \cdots & \vec{u}_k \\ | & & | \end{pmatrix}_{m \times k} \cdot \begin{pmatrix} — \vec{v}_1^T — \\ \vdots \\ — \vec{v}_k^T — \end{pmatrix}_{k \times n}$

$\Rightarrow k \cdot (m+n)$ parameters $\ll m \cdot n$ for small k

Generic matrices, eg., random matrices, will have full rank

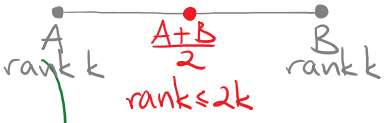
$\text{rank}(A_{m \times n}) = \min\{m, n\}$.

but we would prefer to use low-rank matrices when possible.

Optimizing over low-rank matrices

b/c rank constraints aren't convex

~a common problem, often NP-hard



(eg., https://en.wikipedia.org/wiki/Matrix_completion
https://en.wikipedia.org/wiki/Low-rank_approximation)

$\text{Rank}(A) \leq k \Rightarrow A = \vec{u}_1 \vec{v}_1^T + \cdots + \vec{u}_k \vec{v}_k^T$

$\text{rowspace} = \text{Span}\{\vec{v}_1, \ldots, \vec{v}_k\}$.

each row is a combination, eg.,

$\text{row } 1 = e_1^T A = (u_1)_1 \vec{v}_1^T + \cdots + (u_k)_1 \vec{v}_k^T$

THE SINGULAR-VALUE DECOMPOSITION AND DISTANCE TO LOWER-RANK MATRICES

Example: $A = \begin{pmatrix} 1 & 1/2 & 0 \\ 0 & 1/3 & 1/4 \end{pmatrix}$ $\quad \text{rank}(A) = 4$

Intuitively

What is the rank-one matrix closest to A?

$B = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$

$\|B - A\| = \left\| \begin{pmatrix} 0 & 1/2 & 0 \\ 0 & 1/3 & 1/4 \end{pmatrix} \right\| = 1/2$

Closest rank-two matrix?

$C = \begin{pmatrix} 1 & 1/2 & 0 \\ 0 & 0 & 0 \end{pmatrix}$

$\|C - A\| = 1/3$

Closest rank-3 matrix?

$D = \begin{pmatrix} 1 & 1/2 & 0 \\ \cdots & 1/3 & \end{pmatrix}$

successive approximations

Closest rank-3 matrix?

$$D = \begin{pmatrix} 1 & & & \\ & \frac{1}{2} & & \\ & & \frac{1}{3} & \\ & & & 0 \end{pmatrix}$$

SVD ↓

Theorem 1: Let A have SVD

$$A = \sum_{i=1}^{r} \lambda_i \vec{u}_i \vec{v}_i^T,$$

with sorted singular values

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_r > 0.$$

Then for any $k \leq r$, the closest rank k matrix to A is

$$B = \sum_{i=1}^{k} \lambda_i \vec{u}_i \vec{v}_i^T.$$

Observe: B truncates the smaller singular values of A.

$$\|A - B\| = \lambda_{k+1}$$

Example:

$$\begin{pmatrix} 1 & & \\ & \frac{1}{1000} & \\ & & \frac{1}{1000} \end{pmatrix}$$

has full rank, but is $\frac{1}{1000}$-close to a rank-one (singular) matrix.

Proof: Let C have rank k.

Goal: Find $\vec{x}$ with $\|x\| = 1$, $\|(C-A)\vec{x}\| \geq \lambda_{k+1}$ $\Rightarrow \|C - A\| \geq \lambda_{k+1}$.

Intuition: All vectors x in $\text{Span}\{v_1, v_2, \ldots, v_{k+1}\}$ are stretched by at least λ_{k+1}, and there's no way C cancel out all these $k+1$ dimensions!

find $x \in N(C) \cap \text{Span}\{v_1, \ldots, v_{k+1}\}$ ← new goal!

If these are all $m \times n$ matrices, then $\dim N(C) = n - k$.

Since $\dim \text{Span}\{v_1, \ldots, v_{k+1}\} = k+1$, they must intersect.

For this x, $\|(C-A)x\| = \|Ax\| \geq \lambda_{k+1}\|x\|$. ✓ □

Theorem 2: The same holds for Frobenius norm:

$$B = \sum_{i=1}^{k} \lambda_i \vec{u}_i \vec{v}_i^T \quad \text{achieves} \quad \min_{B:\text{rank}(B)=k} \|A - B\|_F$$

Frobenius norm

$$\|A\|_F = \sqrt{\sum_{i,j} |a_{ij}|^2}$$ ← easy to compute!

Exercise: This satisfies

- $\|cA\|_F = |c| \cdot \|A\|_F$
- $\|A+B\|_F \le \|A\|_F + \|B\|_F$
- $\|A \cdot B\|_F \le \|A\|_F \cdot \|B\|_F$ (like operator norm)

Claim: The Frobenius norm is "basis-independent", i.e.,

$$\|A\|_F = \|UAV\|_F$$

for any unitary matrices U, V. (like operator norm) since unitaries don't change lengths

Proof: **Observe**: $\|A\|_F^2 = \text{Trace}(A^\dagger A)$ ← sum of diagonal elements

Proof:

$$\text{Tr}(A^\dagger A) = \sum_j (A^\dagger A)_{j,j} = \sum_{i,j} (A^\dagger)_{ji} A_{ij}$$
$$= \sum_{i,j} \bar{a}_{ij} a_{ij} = \sum_{i,j} |a_{ij}|^2 \quad \square$$

Fact: Trace is cyclic:

$$\text{Tr}(AB) = \text{Tr}(BA)$$

$\text{Tr}(ABC) = \text{Tr}(CAB) = \text{Tr}(BCA)$

Proof: $\text{Tr}(AB) = \sum_i (AB)_{i,i} = \sum_{i,j} a_{ij} b_{ji} = \sum_{i,j} b_{ji} a_{ij} = \text{Tr}(BA)$ $\square$

Hence

$$\|AV\|_F^2 = \text{Tr}\left[(AV)^\dagger (AV)\right]$$
$$= \text{Tr}\left(V^\dagger A^\dagger A V\right)$$
$$= \text{Tr}\left(A^\dagger A V V^\dagger\right)$$ by cyclic trace
$$= \text{Tr}\left(A^\dagger A\right)$$ V unitary means $V^\dagger = V^{-1}$.

Similarly $\|UA\|_F = \|A\|_F$. $\square$

(The Frobenius norm is the same in all orthonormal bases.)

Theorem 2: The same holds for Frobenius norm:

$B = \sum_{i=1}^{k} \lambda_i \vec{u}_i \vec{v}_i^T$ achieves $\min_{B:\text{rank}(B)=k} \|A - B\|_F$

Interpretation:

$$\|A-B\|_F^2 = \sum \|\text{row vectors}\|^2$$

$\text{Rank}(B)=k \Rightarrow$ all its rows lie in k-dim. subspace

If $A=\begin{pmatrix} \text{—} \ \vec{a}_1^T \ \text{—} \\ \vdots \\ \text{—} \ \vec{a}_m^T \ \text{—} \end{pmatrix}$, $B=\sum_i e_i b_i^T$

$$\|A-B\|_F^2 = \sum_i \|\vec{a}_i - \vec{b}_i\|^2$$

The optimization is equivalent to

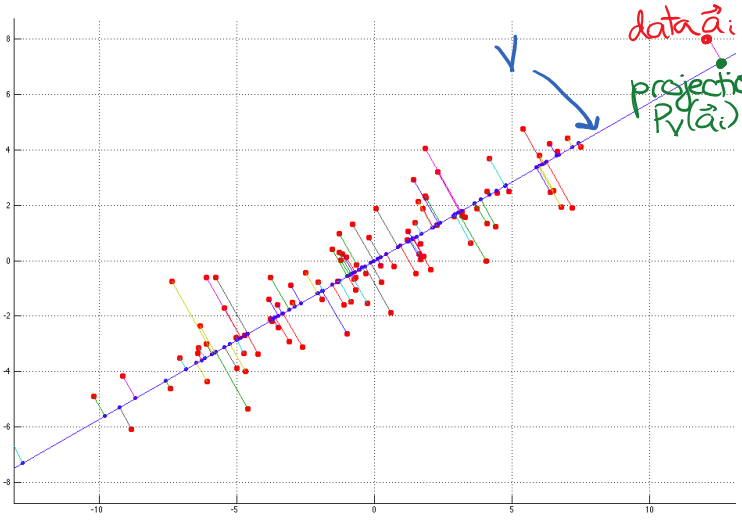
1. Choose a k-dim subspace $V \subseteq \mathbb{R}^n$
2. Choose $\vec{b}_1, \dots, \vec{b}_m \in V$

   $\vec{b}_i$ should minimize $\|\vec{a}_i - \vec{b}_i\|$

   $\Rightarrow \vec{b}_i = \text{Proj}_V(\vec{a}_i)$!

Hence:

$$\min_{B:\ \text{rank}(B)=k} \|A-B\|_F^2 = \min_{\substack{V \subseteq \mathbb{R}^n \\ \dim(V)=k}} \sum_{i=1}^m \|\vec{a}_i - \text{Proj}_V(\vec{a}_i)\|^2$$



If the rows of A $\vec{a}_1, \dots, \vec{a}_m$ are data points, we want to find the dim-k subspace that minimizes the sum of the squared errors.

Claim: The best subspace is

$V = \text{Span}\{\text{first } k \text{ right sing. vectors of } A\}$.

$$\left(\text{so } B=\sum_{i=1}^k \lambda_i \vec{u}_i \vec{v}_i^T\right)$$

Proof: By induction, $k=1$, $k=2$, ...

Since Frob. norm is basis-independent, we can work in the bases of A's left and right singular vectors:

$$\Lambda = \begin{pmatrix} \lambda_1 & & \\ & \lambda_2 & \\ & & \ddots \end{pmatrix}$$

$$A = \begin{pmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \lambda_3 & \\ & & & \ddots \end{pmatrix}$$

k=1: rank(B)=1 means $B = \vec{u}\vec{v}^T$ for some u, v.
We may assume $\|\vec{v}\| = 1$.
Then we'll just use calculus:

$$\sum_{i=1}^{m} \|\vec{a}_i - (\vec{a}_i \cdot \vec{v})\vec{v}\|^2 = \sum \|(I - vv^T)\vec{a}_i\|^2$$

(Projection of $\vec{a}_i$ in direction of $\vec{v}$)

$$= \sum a_i^T (I - vv^T) a_i$$
$$= \sum (\|\vec{a}_i\|^2 - |\vec{a}_i \cdot \vec{v}|^2)$$
$$= \|A\|_F^2 - \sum_i \lambda_i^2 v_i^2$$

Lagrangian

$$\mathcal{L} = \sum_i \lambda_i^2 v_i^2 + \lambda\left(\sum_i v_i^2 - 1\right)$$

$$\frac{\partial}{\partial v_i}\mathcal{L} = 2\lambda_i^2 v_i + 2\lambda v_i$$
$$= 2v_i(\lambda_i^2 + \lambda) = 0 \quad \text{for every } i, \Rightarrow v_i = 0 \text{ or } \lambda_i^2 = -\lambda$$

So the best choice is $\vec{v} = \vec{e}_1 = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$.
if $\lambda_1 \geq \lambda_2 \geq \cdots \geq 0$.

Induction step:

Observe:

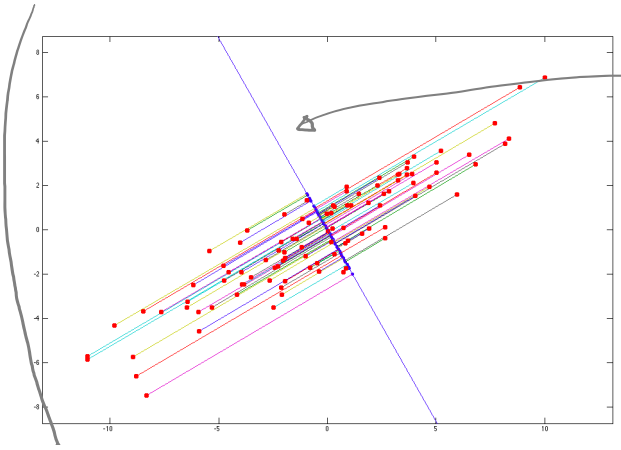
$$\|\vec{a}\|^2 = \|\text{Proj}_V(\vec{a})\|^2 + \|\vec{a} - \text{Proj}_V(\vec{a})\|^2$$

$$\Rightarrow \sum_j \|\vec{a}_j\|^2 = \sum_j \|P_V(a_j)\|^2 + \sum_j \|a_j - P_V(a_j)\|^2$$

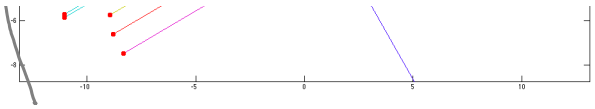
$\|A\|_F^2$

$\therefore$ minimizing sum of squared distances to V $\Leftrightarrow$ maximizing $\sum_j \|P_V(\vec{a}_j)\|^2$

ie., maximizing the projected data's variance



In contrast, projecting to this line minimizes the projections' variance



Let us show the induction for $k=2$. (General argument is similar.)

Let V achieve $\max_{\dim(V)=2} \sum_j \|P_V(\vec{a}_j)\|^2$.

We can choose an orthonormal basis $\vec{\omega}_1, \vec{\omega}_2$ s.t. $\vec{\omega}_2 \perp \vec{v}_1$. (Do you see why?)

$$\sum_j \|P_V(\vec{a}_j)\|^2 = \sum_j |a_j \cdot \omega_1|^2 + \sum_j |a_j \cdot \omega_2|^2$$

$$\sum_j |a_j \cdot v_1|^2$$

For this to be maximal, it must be that $\vec{\omega}_1 = \pm \vec{v}_1$.
Then we just need to maximize over vectors $\vec{\omega}_2$ perpendicular to $\vec{v}_1$.
But that's the $k=1$ problem! So $\vec{\omega}_2 = \vec{v}_2$. □

Observations:

1. The optimal low-rank approximation B is diagonal with respect to the same singular vector bases as A. (Off-diagonal terms don't help.)
2. You don't need to compute the full SVD! Compute the sing. vectors one at a time, largest sing. value to smallest, until you are happy with the approximation. (The algorithm is greedy.)

Remark:

Dimension reduction is a major theme in computational linear algebra and most approaches for data analysis.

- Linear regression, as we have presented it, effectively reduces the dimension of the data by 1, e.g., fitting a 1D line to a 2D cloud of data points.

The Johnson-Lindenstrauss Lemma projects n data points to $\log n$ (random) dimensions, approximately preserving angles & lengths (after rescaling).

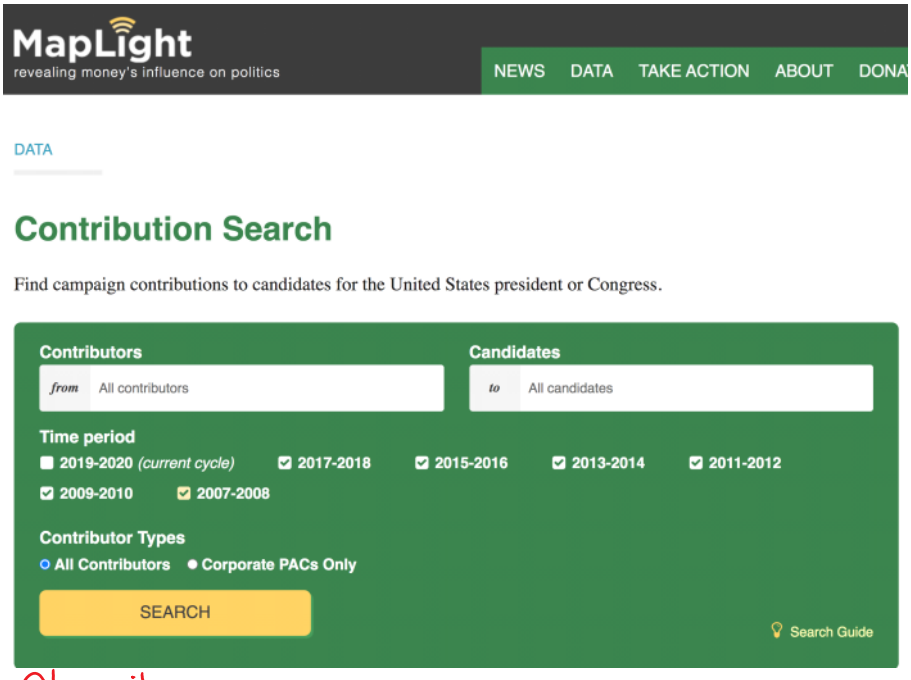
- Matrix rank reduction via the SVD is another example. Moreover, it is used in Principal Component Analysis for reducing the dimensionality of data sets.

We'll see this next.

Principal Component Analysis (PCA)

Example: Campaign contributions to US senators

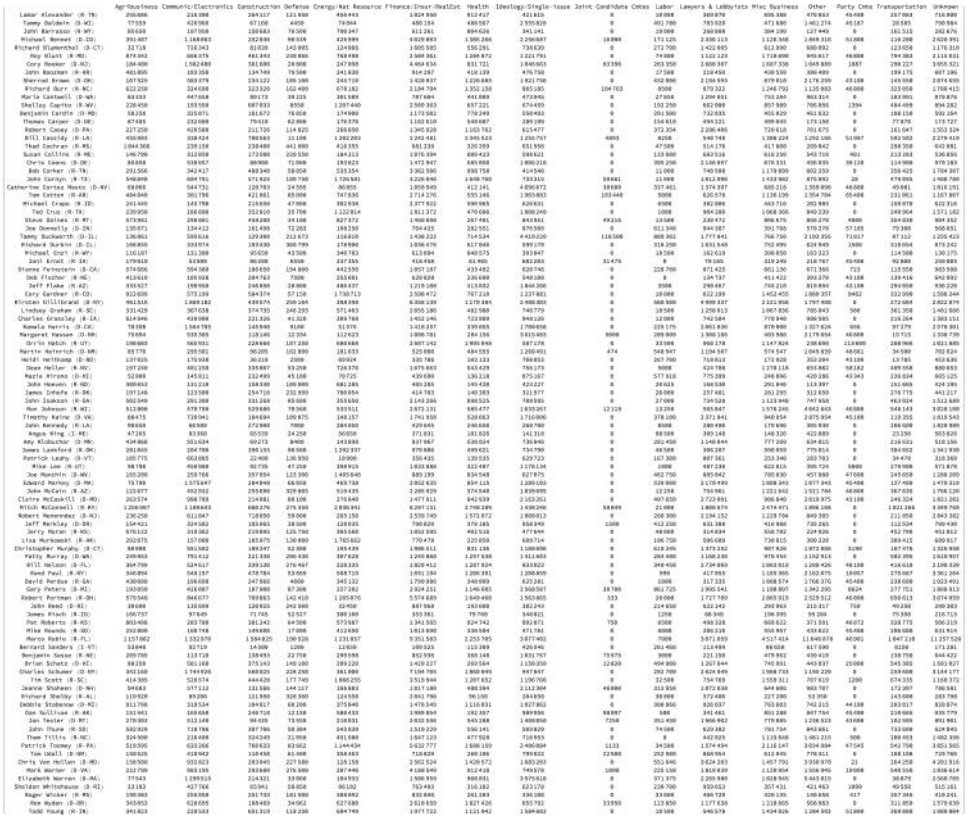
1. Get the data



2. Clean it up

Contributor categories

Senators



3. "Standardize" the data

a) Shift the numbers in each column so the mean $\sum_{i=1}^{17} x_j^{(i)} = 0$.

b) Scale each column so the variance $\sigma_j^2 = \sum_{i=1}^{17} (x_j^{(i)})^2 = 1$.

Note: The scaling may be omitted if the different attributes are known to lie on the same scale.

Here it has the effect of making all categories equally significant, even though some give more than others!

```
{sectors, Plus @@ puredata} // Transpose // Sort[#, #1[[2]] > #2[[2]] &] & // MatrixForm
```

Finance/Insur/RealEst	168 151 725
Unknown	154 842 094
Lawyers & Lobbyists	112 167 979
Other	103 253 137
Ideology/Single-issue	100 385 417
Misc Business	92 373 646
Health	71 844 560
Communic/Electronics	45 991 954
Energy/Nat Resource	40 100 000
Agribusiness	30 637 498
Construction	28 244 716
Transportation	23 257 205
Labor	20 546 123
Education	12 200 463
Defense	10 484 140
Party Cmte	1 690 362
Joint Candidate Cmtes	514 864

```
{height, width} = Dimensions[puredata]
puredatastandardized = puredata // N;
```

$$\text{means} = \frac{1}{\text{height}} \sum_{j=1}^{\text{height}} \text{puredatastandardized}[[j]];$$

```
For[j = 1, j ≤ height, j++,
  puredatastandardized[[j]] -= means;
 ];
```

$$\text{variances} = \frac{1}{\text{height}} \sum_{j=1}^{\text{height}} \text{puredatastandardized}[[j]]^2;$$

```
For[j = 1, j ≤ height, j++,
  puredatastandardized[[j]] /= √variances ;
 ];
{97, 17}
```

4. Take the SVD:

```
{U, M, V} = SingularValueDecomposition[puredatastandardized];
Plus @@ Plus @@ Abs /@ (U.M.Transpose[V] - puredatastandardized)
{Dimensions[U], Dimensions[M], Dimensions[V]}
```

5.96535×10^{-12} ← UMV^T = data matrix ✓

dim U dim M dim V

{{97, 97}, {97, 17}, {17, 17}}

The singular values, sorted from largest on down:

```
M // MatrixForm
```

$$\begin{pmatrix}
33.265 & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\
0. & 12.4751 & 0. & 0. & 0. & 0. & 0. & 0. \\
0. & 0. & 10.0409 & 0. & 0. & 0. & 0. & 0. \\
0. & 0. & 0. & 9.3013 & 0. & 0. & 0. & 0. \\
0. & 0. & 0. & 0. & 7.3885 & 0. & 0. & 0. \\
0. & 0. & 0. & 0. & 0. & 5.60893 & 0. & 0. \\
0. & 0. & 0. & 0. & 0. & 0. & 4.79764 & 0. \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 4.4487 \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0.
\end{pmatrix}$$

Note: The columns of U are the left singular vectors, forming a basis for columnspace $R(data)$.
The columns of V are the right singular vectors, forming a basis for rowspace $R(data^T)$.

5. Now what?

a) We could approximate the data matrix with a lower-rank matrix, eg., keeping

$$\underset{94}{\begin{pmatrix} | & | \\ \text{first two} & \text{columns of } U \\ | & | \end{pmatrix}} \begin{pmatrix} 33.265 & 0 \\ 0 & 12.4751 \end{pmatrix} \underset{2}{\begin{pmatrix} \text{first two columns} \\ \text{of } V, \text{transposed} \end{pmatrix}}$$

(17)

gives a rank-2 matrix.

Each (2) data point (row) now lies in a 2D subspace of $\mathbb{R}^{17}$.

```
"These are the two most important directions:";
principaldirections = V[[All, {1, 2}]];

principaldirectionslabeled = Append[principaldirections // Transpose, sectors] // Transpose;
"Now let's sort these by the second component (we'll see why in a moment).";
Sort[principaldirectionslabeled,
  #1[[2]] < #2[[2]] &
 ] // MatrixForm
```

$\vec{v}_1$	$\vec{v}_2$	
-0.00387268	-0.707777	Labor
-0.238924	-0.329765	Education
-0.261419	-0.268357	Lawyers & Lobbyists
-0.260683	-0.216817	Ideology/Single-issue
-0.266349	-0.211043	Communic/Electronics
-0.282418	-0.00418024	Other
-0.288545	0.00920218	Unknown
-0.286914	0.0197931	Finance/Insur/RealEst
-0.291324	0.0353523	Misc Business
-0.273799	0.0478608	Health
-0.2148	0.05117	Defense
-0.0122403	0.0995367	Joint Candidate Cmtes
-0.285753	0.110642	Construction
-0.273507	0.174331	Transportation
-0.0855998	0.186206	Party Cmte
-0.241713	0.244521	Agribusiness
-0.248974	0.270343	Energy/Nat Resource

b) Or, if you care only about the data points' positions in the 2D subspace — and not about the subspace's position in $\mathbb{R}^{17}$ — then we can use the first two columns of V to project down to $\mathbb{R}^2$:

```
puredatastandardized // Dimensions
principaldirections // Dimensions
dataprojectedmanually = puredatastandardized.principaldirections;
```

```
{97, 17}
```

```
{17, 2}
```

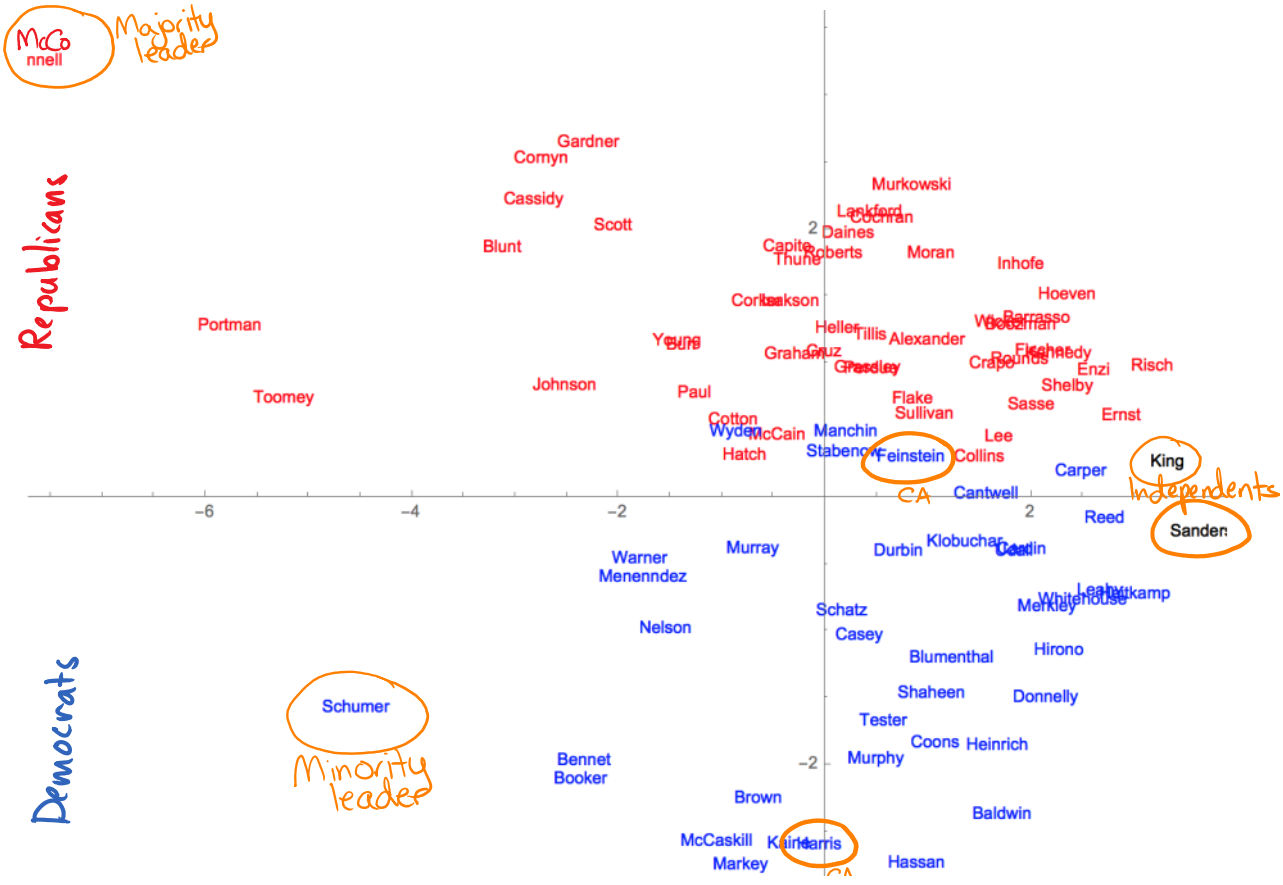
Since $data = U M V^T$, $data * \begin{pmatrix} | & | \\ \text{first two} & \text{columns of } V \\ | & | \end{pmatrix} = \sum_{i=1}^{2} \sigma_i \vec{u}_i \vec{e}_i^T$

$= \sum_{i=1}^{17} \sigma_i \vec{u}_i \vec{v}_i^T$

$\vec{v}_1 \vec{e}_1^T + \vec{v}_2 \vec{e}_2^T$

$= \begin{pmatrix} | & | \\ \sigma_1 \vec{u}_1 & \sigma_2 \vec{u}_2 \\ | & | \end{pmatrix}$

Of course, we could also just have picked out the first two left singular vectors directly.
Let's plot it!



Interpretation: 2nd principal component → political party (maybe liberal vs. conservative?)

1st principal component → ???

Since all entries of the first principal direction are <0, the x-direction above says who raised more money (with the most to the left). The y-direction (second principal direction) says how that money is distributed away from average. Negative means more Labor. Positive means more Energy/Nat. Resources.

Notice that there is a slight funnel effect; candidates who raise more money (on the left) also get it from a more biased distribution of donors.

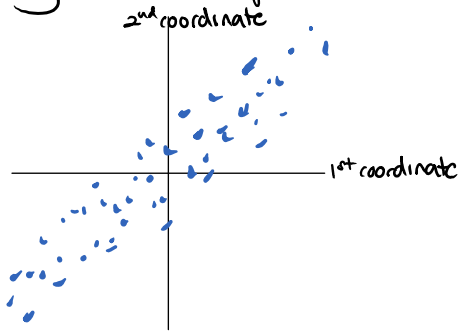
Find correlations, clusters, voids, study more principal components, etc.

Note: Matlab/Mathematica/R all have built-in PCA functions:

```
principalcomponents = PrincipalComponents[puredatastandardized];

"To keep just the first two principal components (first two columns), use:";
principalcomponents[[All, {1, 2}]]
```

Question: Why can't the plot look like this?



Answer: We are plotting $(x_1,y_1),\dots,(x_m,y_m)$, where $(x_1,\dots,x_m)$ is the first left singular vector and $(y_1,\dots,y_m)$ is the second left singular vector. These vectors are orthogonal:

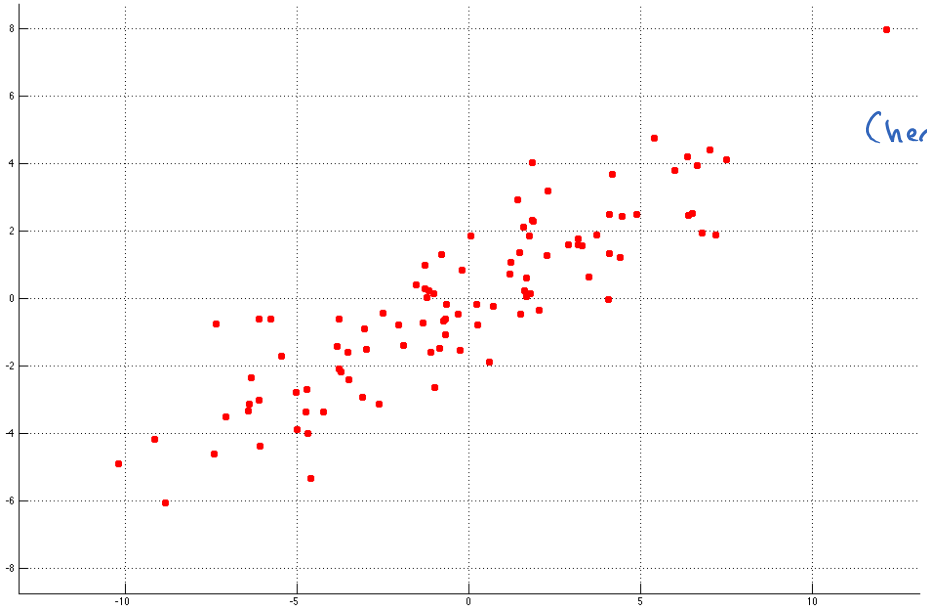
$$\sum_{j=1}^{m} x_j y_j = 0.$$

Informal motivation for PCA:

"The first k principal directions are the k factors that explain the data best."

Formal mathematical motivation:

Given data points $\{(x_1^{(1)},\dots,x_k^{(1)}),(x_1^{(2)},\dots,x_k^{(2)}),\dots,(x_1^{(m)},\dots,x_k^{(m)})\}$,

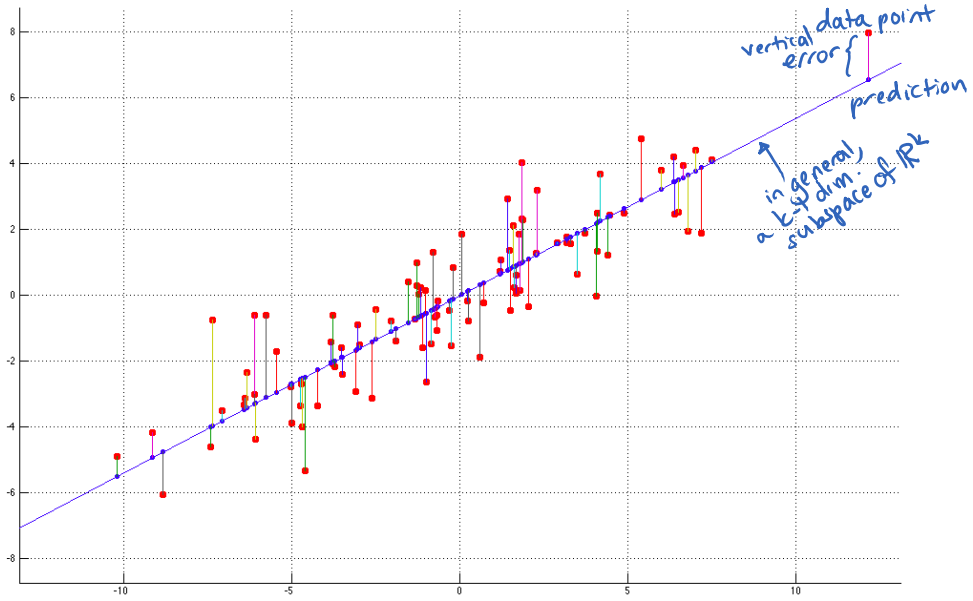


(here $k=2$)

The "method of least squares" lets us predict the last component/attribute in terms of the first $k-1$ components, minimizing the squared error

$$\left\| \begin{pmatrix} 1 & x_1^{(1)} & \cdots & x_{k-1}^{(1)} \\ \vdots & \vdots & & \vdots \\ 1 & x_1^{(m)} & \cdots & x_{k-1}^{(m)} \end{pmatrix} \begin{pmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_{k-1} \end{pmatrix} - \begin{pmatrix} x_k^{(1)} \\ \vdots \\ x_k^{(m)} \end{pmatrix} \right\|^2$$

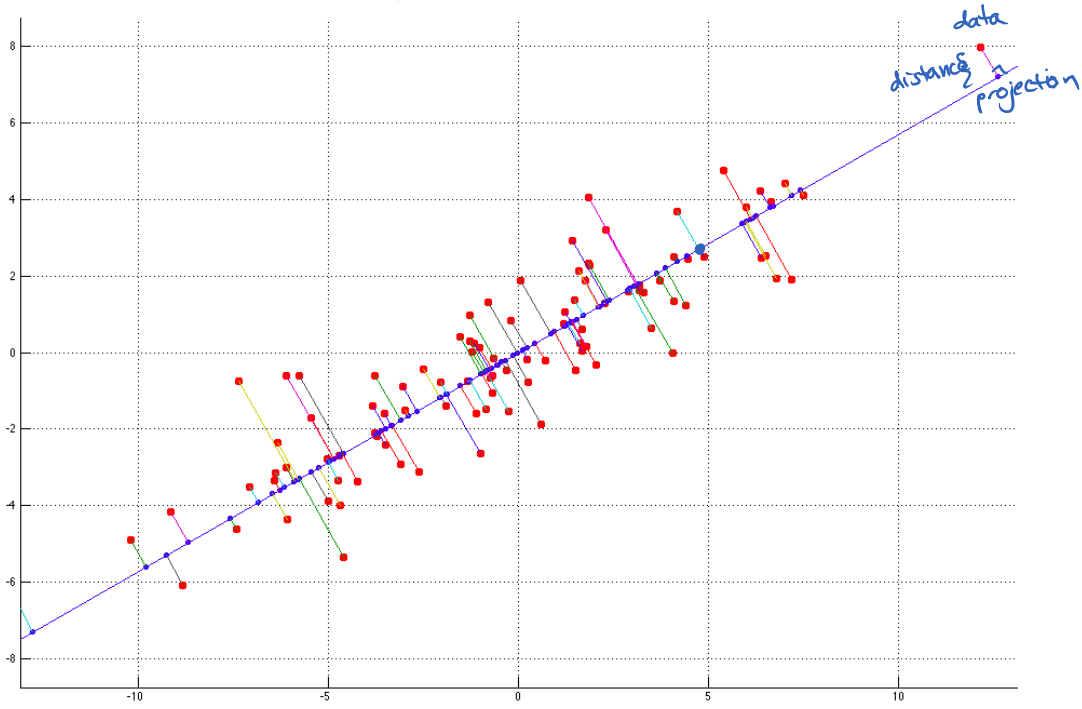
$$= \sum_{j=1}^{m} \left(\alpha_0 + \alpha_1 x_1^{(j)} + \cdots + \alpha_{k-1} x_{k-1}^{(j)} - x_k^{(j)} \right)^2$$



This is most useful if the first k-1 attributes are exact, and the last one imprecise or noisy.

Least-squares regression is not useful for analyzing the campaign finance data, because all the components are imprecise/noisy (equally noisy after scaling the columns), and we aren't just trying to predict one of them. Basically, the data are less structured.

In contrast, principal component analysis (PCA) finds the subspace (with a dimension you can choose) that minimizes the total squared **distances** from the subspace.



Applications of Principal Component Analysis (PCA):

Compression to fewer dimensions:

- saves space
- reduces noise
- useful for visualization & finding patterns
- allows for faster data processing
- reduces over-fitting in machine learning
  (simpler data means you can use simpler hypothesis classes)