lecture follows notes on webpage.

PCA Principal Components Analysis

data dependent dimensionality reduction

we have a dataset x1, x2, …, xn in R^d

Credit for figures:
Roughgarden & Valiant
John Benedetto
Novembre et al
Alex Williams
Sandipan Dey
Victor Lavrenko

Useful for

- visualization
- interpretation
- compression

find intrinsic dimensionality of the data.

Example: $n = 4$ data pts
$d = 4$ dimensions

		kale	taco bell	sashimi	pop tarts
x_A	Alice	10	1	2	7
x_B	Bob	7	2	1	10
x_C	Carolyn	2	9	7	3
x_D	Dave	3	6	10	2
	$\bar{x}$	5.5	4.5	5	5.5

x_i = row of table, i.e. one persons preferences.

Claim:
each row can be expressed ≈ as $x_i = \bar{x} + a_{i1}\vec{v}_1 + a_{i2}\vec{v}_2$

reduce persons preferences from a 4D system (kale, taco bell, sashimi, pop tarts) to a 2D system (v1, v2)

$\bar{x} = (5.5, 4.5, 5, 5.5)$

$\vec{v}_1 = (3, -3, -3, 3)$

$\vec{v}_2 = (1, -1, 1, -1)$

3's on kale and pop tarts (the two are correlated) and -3's on taco bell / sashimi (two are correlated). Captures some structure in data / correlation

notice. $(v_1 \cdot v_2 = 0)$

orthogonal / uncorrelated

		kale	taco bell	sashimi	pop tarts
x_A	Alice	10	1	2	7
x_B	Bob	7	2	1	10
x_C	Carolyn	2	9	7	3
x_D	Dave	3	6	10	2
	$\bar{x}$	5.5	4.5	5	5.5

average columnwise

Claim:

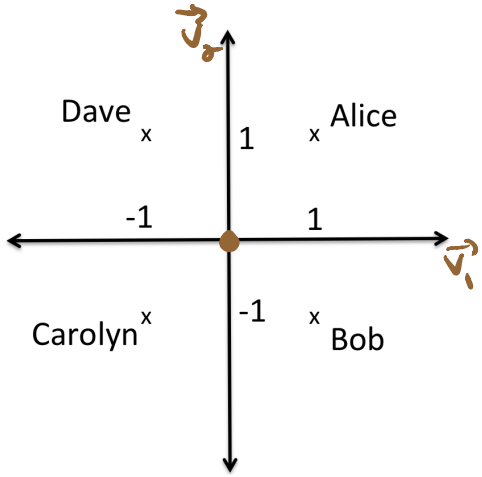
$x_A \approx \bar{x} + 1 \cdot \vec{v}_1 + 1 \cdot \vec{v}_2$

$x_B \approx \bar{x} + 1 \cdot \vec{v}_1 + (-1) \cdot \vec{v}_2$

$x_C \approx \bar{x} + (-1) \cdot \vec{v}_1 + (-1) \cdot \vec{v}_2$

$x_D \approx \bar{x} + (-1) \cdot \vec{v}_1 + 1 \cdot \vec{v}_2$

	(a_{i1}, a_{i2})
Alice	1 1
Bob	1 -1
Carol	-1 -1
Dave	-1 1



origin is \bar{x}

Why bother?

1. visualization
2. helps interpret data

each pt has "$\vec{v}_1$ coordinate"
"$\vec{v}_2$ coordinate"

develops the best coordinate system!

Goal: express n d-dimensional vectors $\vec{x}_1, \ldots, \vec{x}_n \in \mathbb{R}^d$ as linear combinations of k vectors $\vec{v}_1, \ldots, \vec{v}_k \in \mathbb{R}^d$

with k << n ideally

so $\vec{x}_i \approx \sum_{j=1}^{k} a_{ij} \vec{v}_j$

(assuming $\sum_{i=1} x_i = 0$)

Preprocessing:

1. Subtract mean
2. Scale coordinates so that variance is 1 in each coords

[to avoid sensitivity to units]

SVD chooses coordinates to maximize variance. If our units create different magnitudes in our data this can naturally generate more/less variance, but that does not mean the data is more/less correlated in that direction.

Framing PCA as an optimization problem.

$k=1$

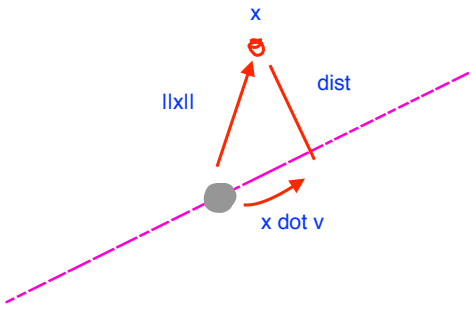
choose $\vec{v} \in \mathbb{R}^d$ with $\|v\|=1$

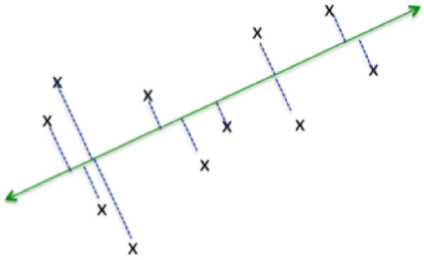
unit vector!

to minimize

$$\frac{1}{n}\sum_{i=1}^{n}\left(\text{dist}\left(x_i \leftrightarrow \text{line defined by } \vec{v}\right)\right)^2$$

least squared error between the line specified by vector v (thru origin?) and data



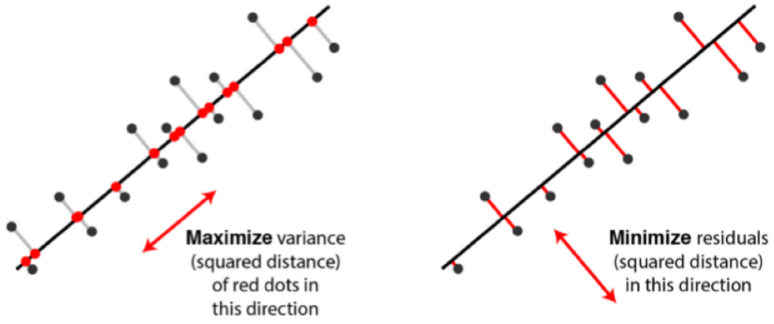


(x dot v) ^2 + dist ^ 2 = ||x||^2 = constant!

Thus minimizing dist is equivalent to maximizing the average (x dot v) = squared projections

MAXIMIZING VARIANCE OF PROJECTED POINTS

Choose v with || v || = 1 that maximizes 1/n sum_i (x_i dot v)



Two equivalent views of principal component analysis.

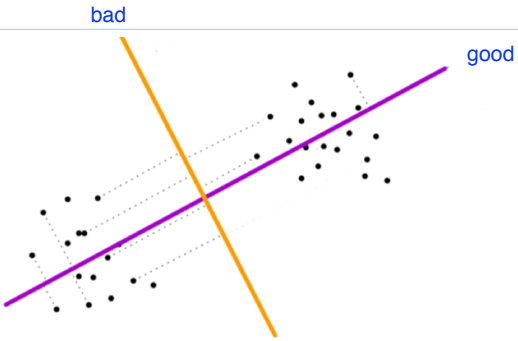
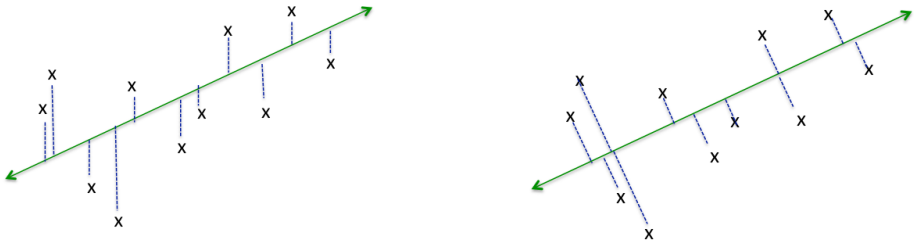


Figure 5: For the good line, the projection of the points onto the line keeps the two clusters separated, while the projection onto the bad line merges the two clusters.

Comparison to linear regression



(a) Linear regression

(b) PCA

Figure 2: Linear regression minimizes the sum of squared vertical distances, while PCA minimizes the sum of squared perpendicular distances.

Goal: express n d-dimensional vectors $\vec{x}_1, \ldots, \vec{x}_n \in \mathbb{R}^d$ as linear combinations of k vectors $\vec{v}_1, \ldots, \vec{v}_k \in \mathbb{R}^d$

so $\vec{x}_i \approx \sum_{j=1}^{k} a_{ij} \vec{v}_j$

Larger k

find k dim subspace s.t.

squared distances minimized = variance of projection maximized.

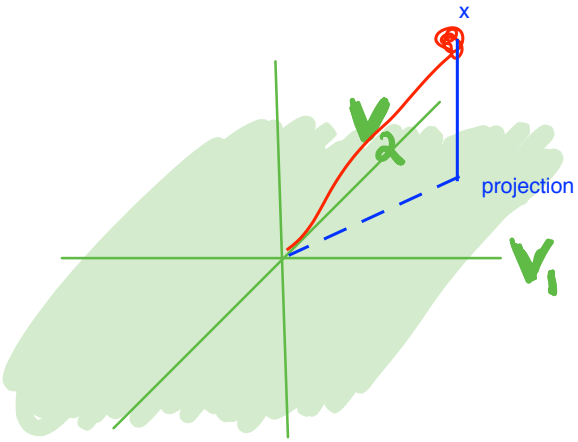
$$\max_{k \text{ dim subspaces } S} \frac{1}{n} \sum_{i=1}^{n} (\text{length of } x_i\text{'s projection})^2$$

define subspace using k orthonormal vectors v1, …, vk in Rd
|| vi || - 1 and (vi, vj) = 0 for i =/= j

Subspace = span{ v1, … , vk}

max 1/n sum_{i=1}^n sum_{j = 1}^k (xi dot vj)^2

p = a1v1 + a2v2 = (x,



Summary:

Problem: compute orthonormal vectors $v_1, \ldots, v_k$

to maximize $\frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{k} (x_i, v_j)^2$

$\sum_{j=1}^{k} (x_i, v_j)^2$ = ‖ projection onto span {v1, …, vk} ‖^2

avg sum of squared projection lengths

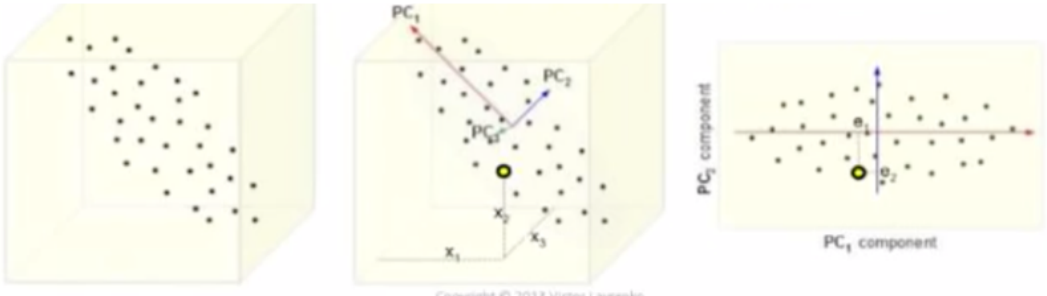
Application: Visualization

① Perform PCA → $\vec{v}_1, \ldots, \vec{v}_k$ top k "principal components"

② $\forall \vec{x}_i$ define "$\vec{v}_1$-coord" (x_i, v_1)
"$\vec{v}_2$-coord" (x_i, v_2)
⋮
"$\vec{v}_k$-coord" (x_i, v_k)

③ Plot points

$x_i \rightarrow ((x_i, v_1), (x_i, v_2), \ldots, (x_i, v_k))$



* look for clusters

* look for pts particularly large along $\vec{v}_1$

Cool example
Genetic data of Europeans
[Novembre et al]

3192 people
200,000 SNPs
DNA positions that
tend to exhibit gene mutation.

using only genomic data

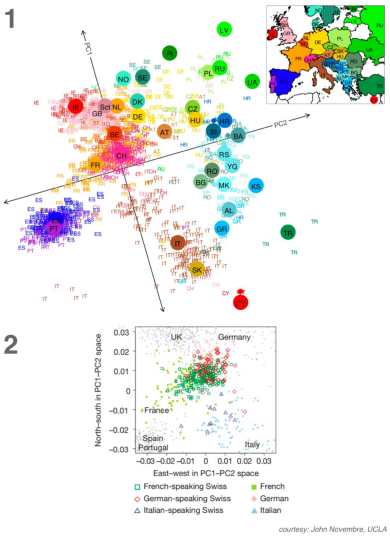


Figure 1: The genetic map of Europe using PCA, with the geographic map of Europe for reference. Figure 2: The same map, but zoomed in on Switzerland. Swiss individuals tend to cluster with countries that speak the same language. *(Courtesy: John Novembre, UCLA)*

DNA strongly corrleated to where people live.

How PCA works

k=1

finding the first principal component

$$\arg\max_{\|v\|=1} \frac{1}{n} \sum_{i=1}^{n} (x_i, v)^2$$

$x_i \in \mathbb{R}^d$

$v \in \mathbb{R}^d$

$$X = \begin{pmatrix} — & x_1 & — \\ & \vdots & \\ — & x_n & — \end{pmatrix}$$

$n \times d$

means are zero as we subtracted out expected value.

finding the first principal component.

Problem: find $\vec{v}$ with $\|v\|=1$

to maximize $v^T A v$

where $A = X^T X$

Suppose magically A diagonal

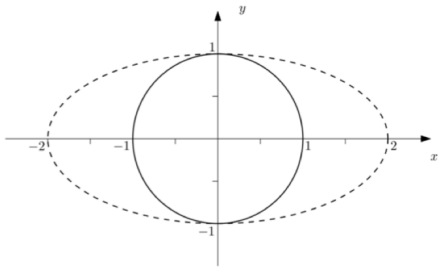


Figure 1: The point (x, y) on the unit circle is mapped to $(2x, y)$.

$$\begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix}$$

$$A = \begin{pmatrix} 3 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0.5 \end{pmatrix}$$

Let v = (v1, v2, v3)

v^TAv = 3v1^2 + 1v2^2 + 0.5v3^2 such that v1^2 + v2^2 + v3^2 = 1

Obviously to maximize the first quantity, just choose the coordinate that has the largest coefficient to one! i.e. set v1 = 1 and v2,v3 = 0

Every symmetric matrix is "diagonals in disguise"

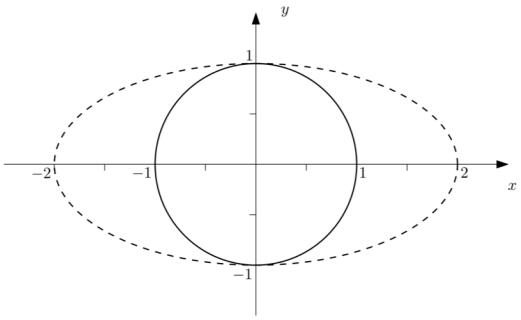


Figure 1: The point (x, y) on the unit circle is mapped to $(2x, y)$.

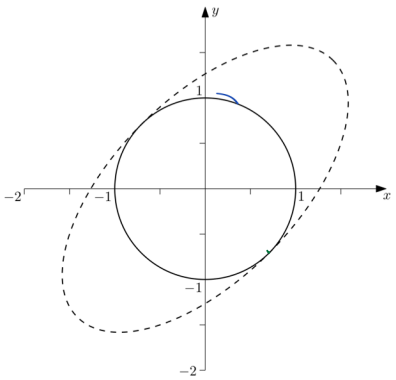


Figure 2: The same scaling as Figure 1, but now rotated 45 degrees.

$$\begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix}$$

$$\begin{pmatrix} \frac{3}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{3}{2} \end{pmatrix} = \underbrace{\begin{pmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{pmatrix}}_{\text{rotate back } 45^\circ} \cdot \underbrace{\begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix}}_{\text{stretch}} \cdot \underbrace{\begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{pmatrix}}_{\text{rotate clockwise } 45^\circ}.$$

For any symmetric matrix A we can write A as

$A = Q D Q^T$

Text

orthogonal

diagonal $\begin{pmatrix} \lambda_1 & & 0 \\ & \ddots & \\ 0 & & \lambda_2 \end{pmatrix}$

(orthonormal columns; eigenvectors corresponding to eigenvalues in D)

ORTHOGONAL MATRICES preserve length; i.e. if Q is symmetric then for any x : ll x ll = ll Qx ll

with lambda1 > lambda 2> … > 0 as matrix is positive semi-definite (X^TX is always positive semi-definite)

$$A = QDQ^T$$

$$Q^T = \begin{pmatrix} — & z_1^T & — \\ & \vdots & \\ — & z_n^T & — \end{pmatrix}$$

$$Q = \begin{pmatrix} | & & | \\ z_1 & \cdots & z_n \\ | & & | \end{pmatrix}$$

also has direction of max stretch

"rotated axis" that gets stretched the most.

eigenvector corresponding to the largest eigenvalue (since X^TX is positive semi-definite, we dont have to worry about negative eigenvectors

Solution to maximization problem

$$\max_{v \text{ s.t. } \|v\|=1} v^T A v$$

Every matrix of form $A = X^T X$ can be written as $Q^T D Q$ where all $\lambda_i \geq 0$

$$Q^T = \begin{pmatrix} | & & | \\ z_1 & \cdots & z_n \\ | & & | \end{pmatrix}$$

$$Q = \begin{pmatrix} | & & | \\ z_1 & \cdots & z_n \\ | & & | \end{pmatrix}$$

$$D = \begin{pmatrix} \lambda_1 & & 0 \\ & \ddots & \\ 0 & & \lambda_d \end{pmatrix}$$

Conclusion: for $k=1$, soln is first column of Q

for general k, soln is first k cols of Q

$\equiv$ top k principal components

Computing principal components:

1. SVD. [cubic]
2. power iteration

Algorithm 1
POWER ITERATION

Given matrix $\mathbf{A} = \mathbf{X}^\top \mathbf{X}$:

- Select random unit vector $\mathbf{u}_0$
- For $i = 1, 2, \ldots$, set $\mathbf{u}_i = \mathbf{A}^i \mathbf{u}_0$. If $\mathbf{u}_i / ||\mathbf{u}_i|| \approx \mathbf{u}_{i-1} / ||\mathbf{u}_{i-1}||$, then return $\mathbf{u}_i / ||\mathbf{u}_i||$.

To see why this works:

Suppose $v^1, v^2, \ldots, v^n$ are eigenvectors of A

Then they form an orthonormal basis.

$\Rightarrow \forall$ vector u: $u = \sum_{i=1}^{n} c_i v_i$

$$Au = A \sum c_i v_i = \sum c_i A v_i$$

$$= \sum c_i \lambda_i v_i$$

$$A^k u = A^{k-1} Au = \sum c_i \lambda_i A^{k-1} v_i$$

$$= \sum c_i \lambda_i^k v_i \qquad c_i \lambda_i A^{k-2} \lambda_i v_i \ldots$$

$$= c_1 \lambda_1^k \left[v_1 + \left(\frac{\lambda_2}{\lambda_1}\right)^k v_2 + \left(\frac{\lambda_3}{\lambda_1}\right)^k v_3 + \cdots \right]$$

$$\approx c_1 \lambda_1^k v_1$$

⇒ when we normalize,

get v_1

this depended on $\lambda_2 < \lambda_1$

& then can show that $O\left(\frac{\log n}{\log(\lambda_1/\lambda_2)}\right)$

iterations suffice to get very close.

1. Find the top component, $\mathbf{v}_1$, using power iteration.
2. Project the data matrix orthogonally to $\mathbf{v}_1$:

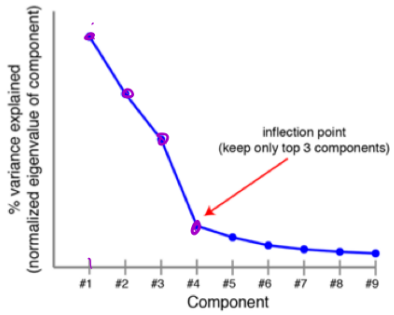
$$\begin{bmatrix} — & \mathbf{x}_1 & — \\ — & \mathbf{x}_2 & — \\ & \vdots & \\ — & \mathbf{x}_m & — \end{bmatrix} \mapsto \begin{bmatrix} — & (\mathbf{x}_1 - \langle \mathbf{x}_1, \mathbf{v}_1 \rangle \mathbf{v}_1) & — \\ — & (\mathbf{x}_2 - \langle \mathbf{x}_2, \mathbf{v}_1 \rangle \mathbf{v}_1) & — \\ & \vdots & \\ — & (\mathbf{x}_m - \langle \mathbf{x}_m, \mathbf{v}_1 \rangle \mathbf{v}_1) & — \end{bmatrix}.$$

   This corresponds to subtracting out the variance of the data that is already explained by the first principal component $\mathbf{v}_1$.
3. Recurse by finding the top $k-1$ principal components of the new data matrix.

"Greedy alg" correctness: k-dim subspace that maximizes norms of projections contains k-1-dimensional space that maximizes norms of projections.

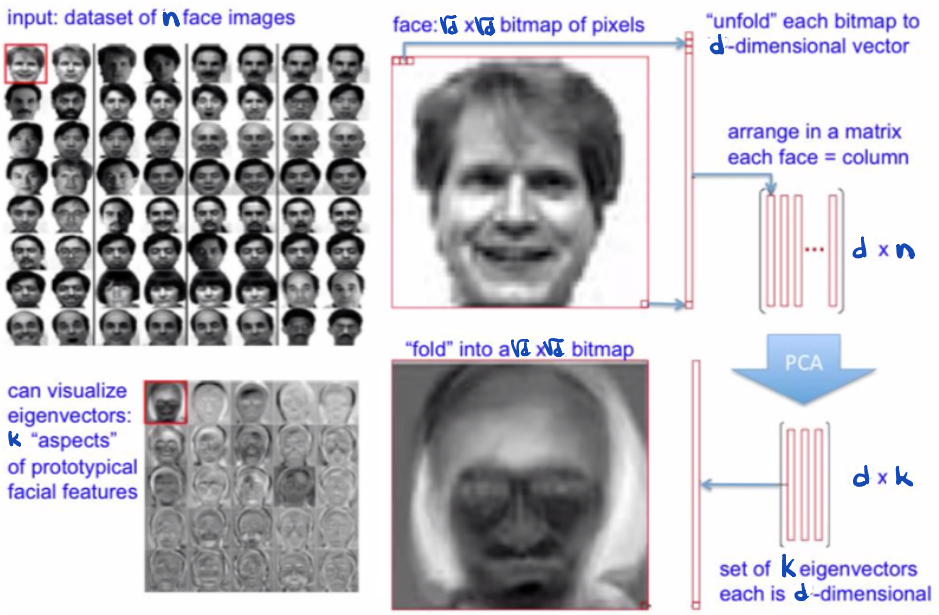
How to choose k?

- For data visualization: a few
- compression: Look at eigenvalues. As soon as small enough, happy.



Scree plot. Principal components are ranked by the amount of variance they capture in the original dataset, a scree plot can provide some sense of how many components are needed.

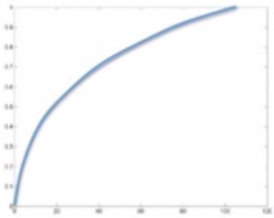
PCA example: Eigen Faces



Eigenfaces Slides by Victor Lavrenko

Eigen Faces: Projection

- Project new face to space of eigen-faces
- Represent vector as a linear combination of principal components
- How many do we need?



(Eigen) Face Recognition

- Face similarity
  - in the reduced space
  - insensitive to lighting expression, orientation
- Projecting new "faces"
  - everything is a face

new face

projected to eigenfaces