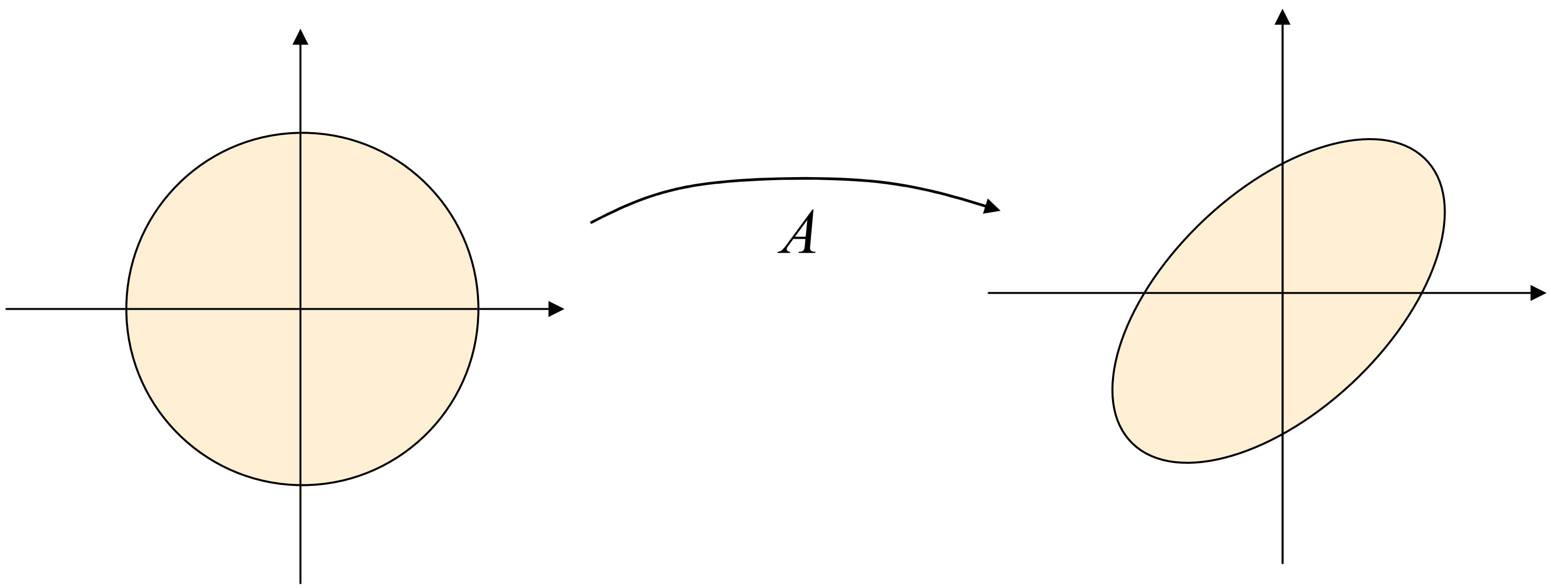
3D Geometry for Computer Graphics

Lesson 2: PCA & SVD

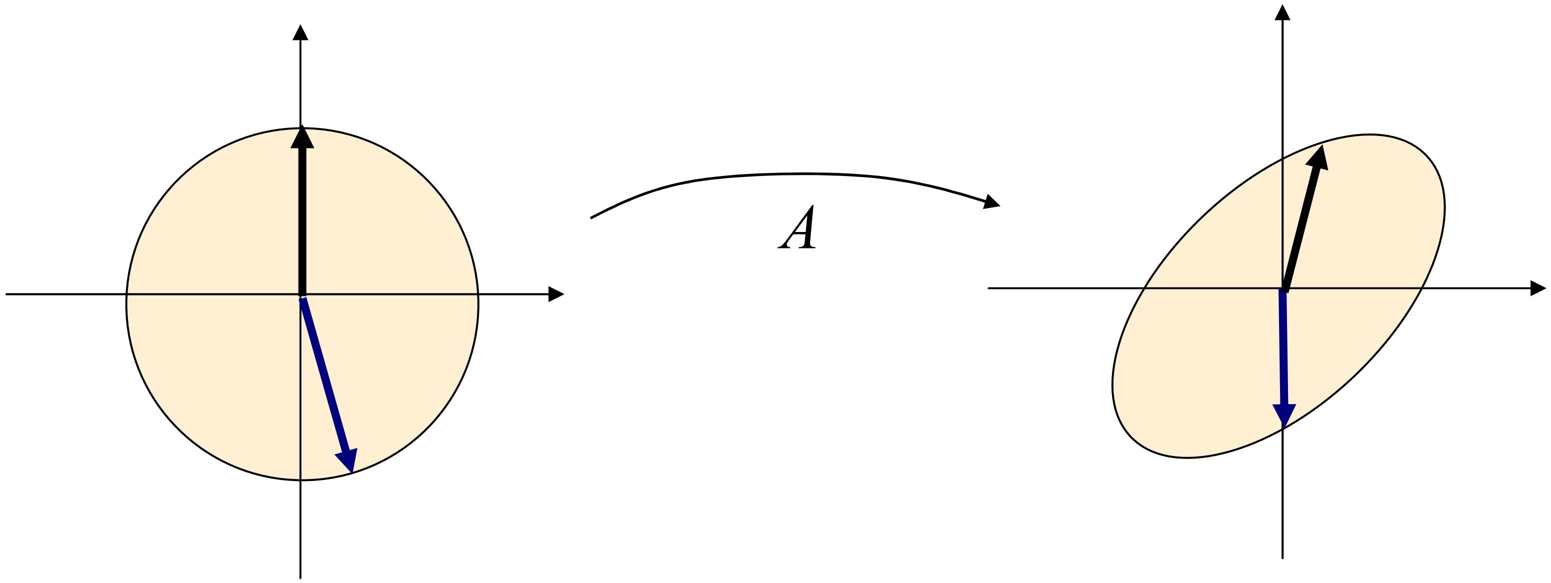
Last week - eigendecomposition

- We want to learn how the matrix A works:



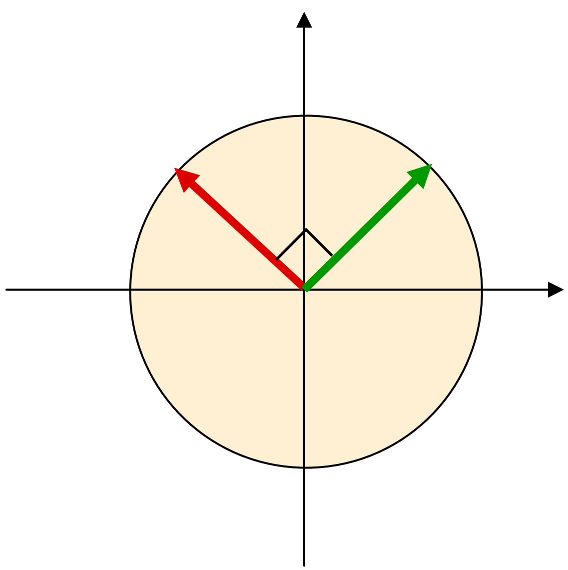
Last week - eigendecomposition

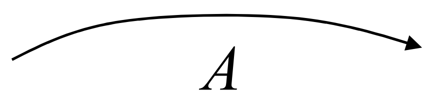
- If we look at arbitrary vectors, it doesn't tell us much.

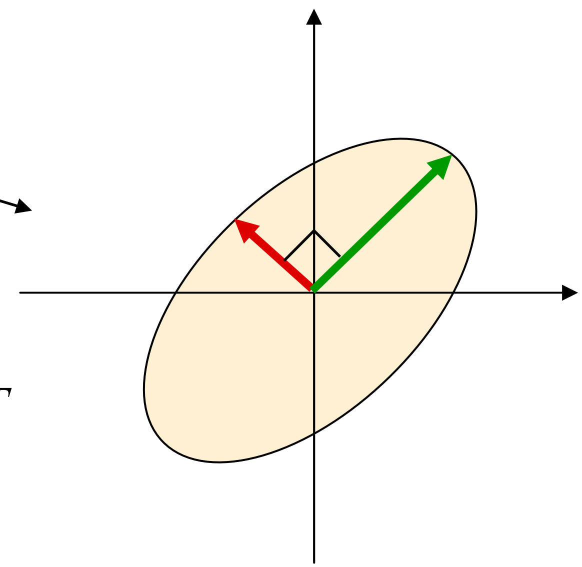


Spectra and diagonalization

- If A is symmetric, the eigenvectors are orthogonal (and there's always an eigenbasis).







$$A = U \Lambda U^T$$

$$\Lambda = \begin{pmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_n \end{pmatrix} \qquad A\mathbf{u}_i = \lambda_i \, \mathbf{u}_i$$

Why SVD…

- Diagonalizable matrix is essentially a scaling.
- Most matrices are not diagonalizable – they do other things along with scaling (such as rotation)

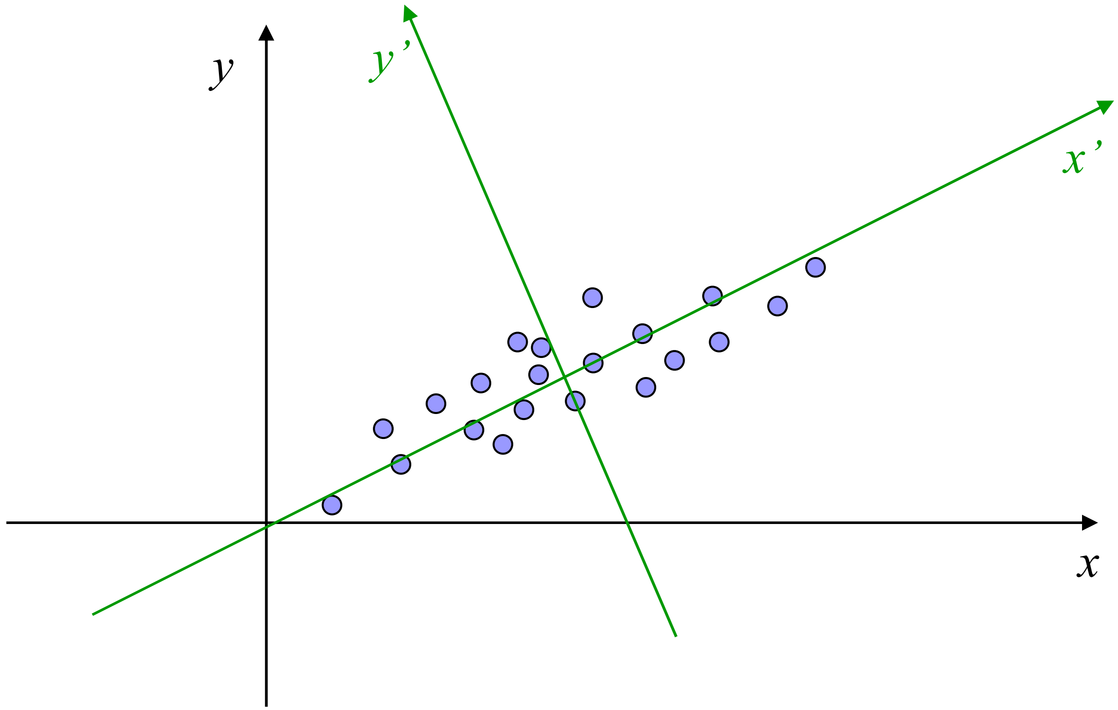
- So, to understand how general matrices behave, only eigenvalues are not enough
- SVD tells us how general linear transformations behave, and other things…

The plan for today

- First we'll see some applications of PCA – Principal Component Analysis that uses spectral decomposition.
- Then look at the theory.
- SVD
  - Basic intuition
  - Formal definition
  - Applications

PCA – the general idea

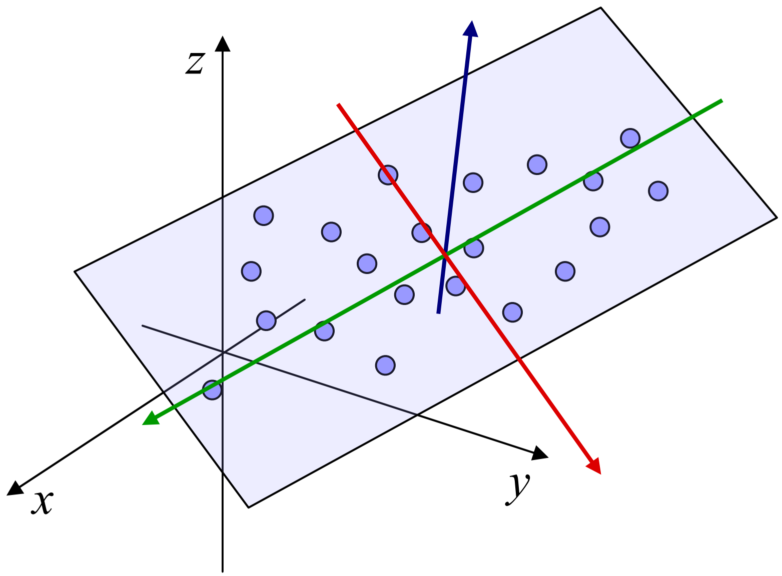
- PCA finds an orthogonal basis that best represents given data set.



- The sum of distances^2 from the x' axis is minimized.

PCA – the general idea

- PCA finds an orthogonal basis that best represents given data set.

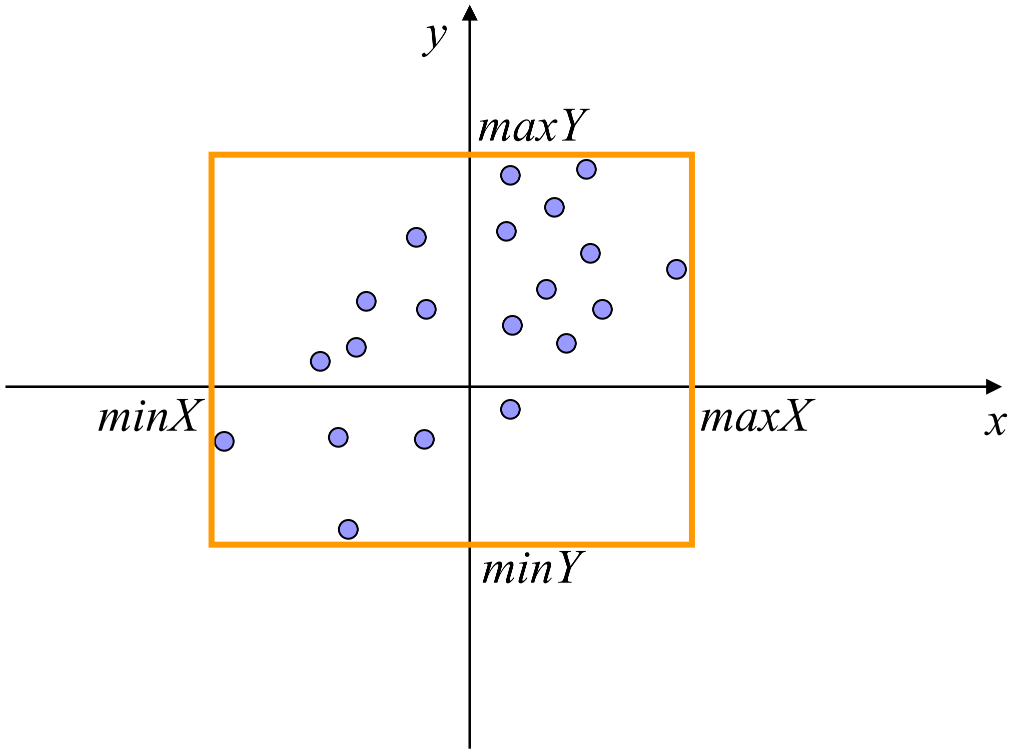


3D point set in standard basis

- PCA finds a best approximating plane (again, in terms of $\Sigma distances^2$)

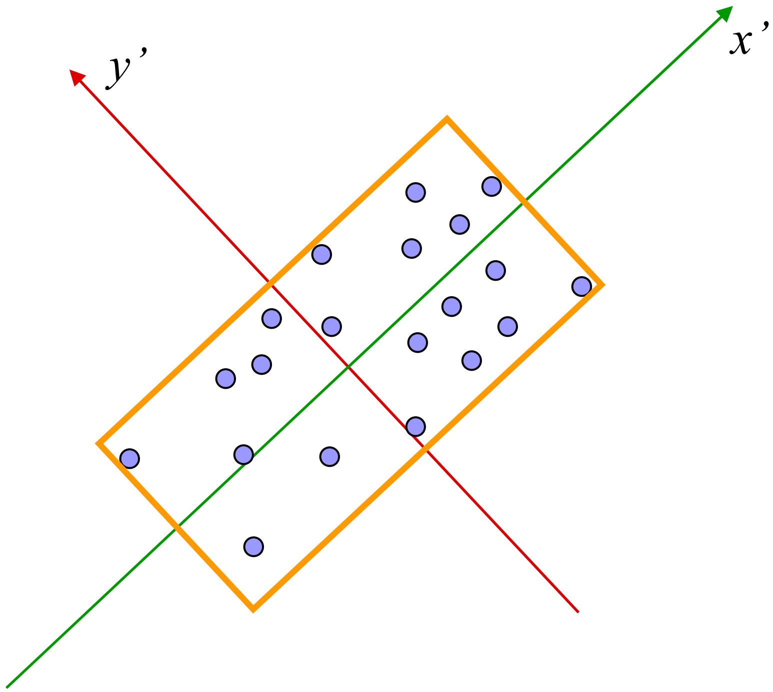
Application: finding tight bounding box

- An axis-aligned bounding box: agrees with the axes



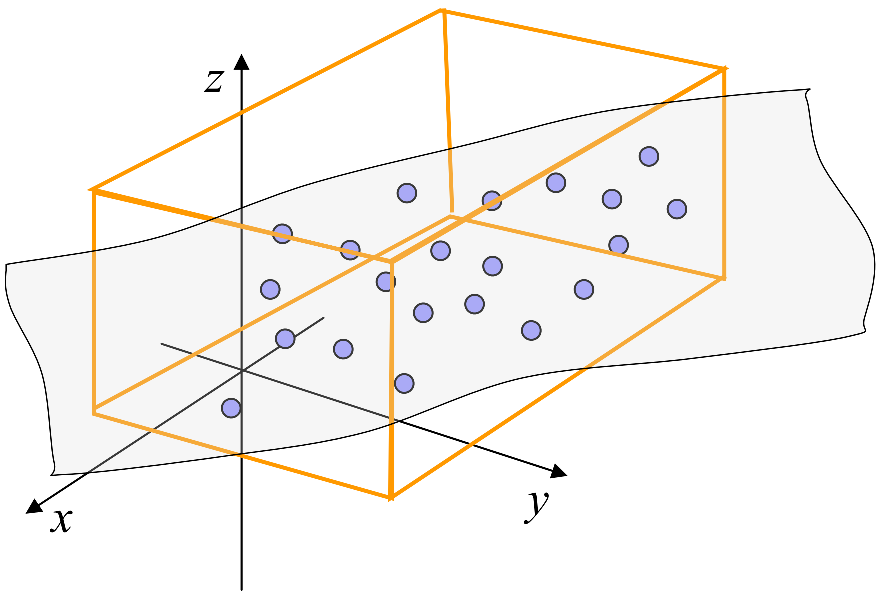
Application: finding tight bounding box

- Oriented bounding box: we find better axes!



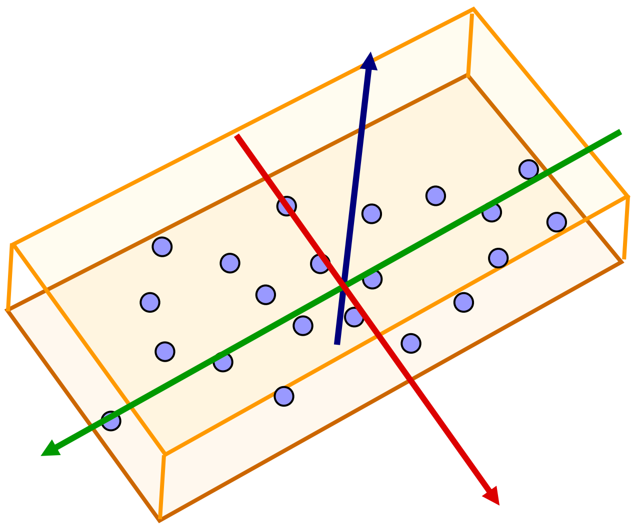
Application: finding tight bounding box

- This is not the optimal bounding box



Application: finding tight bounding box

- Oriented bounding box: we find better axes!

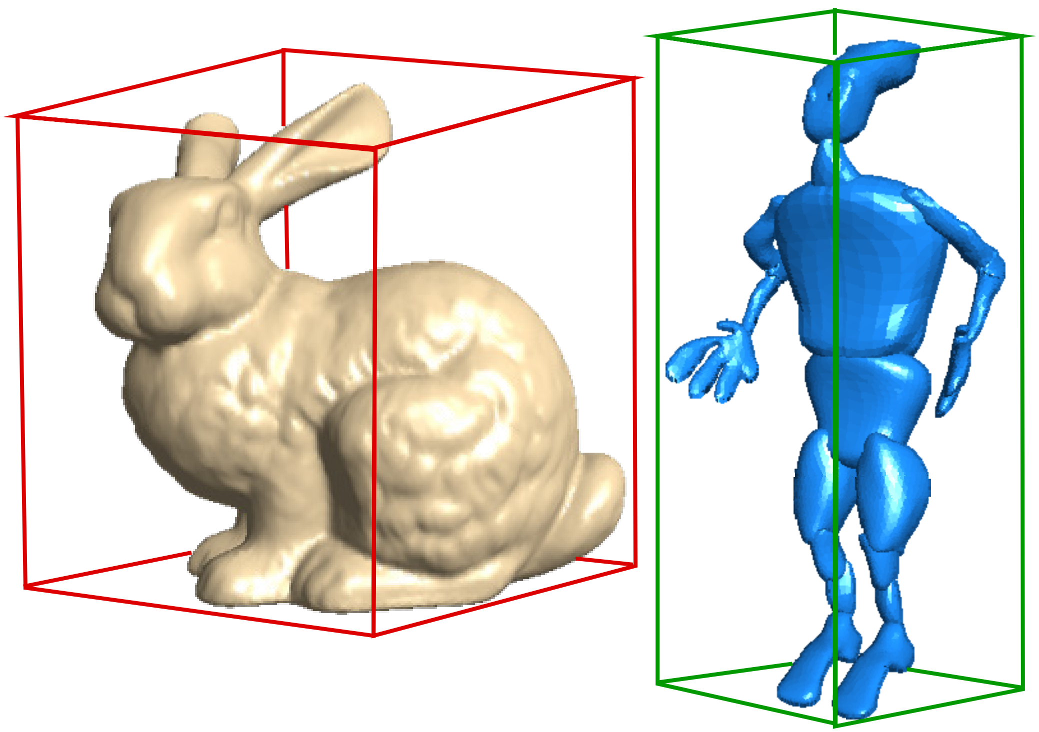


Usage of bounding boxes (bounding volumes)

- Serve as very simple "approximation" of the object
- Fast collision detection, visibility queries
- Whenever we need to know the dimensions (size) of the object

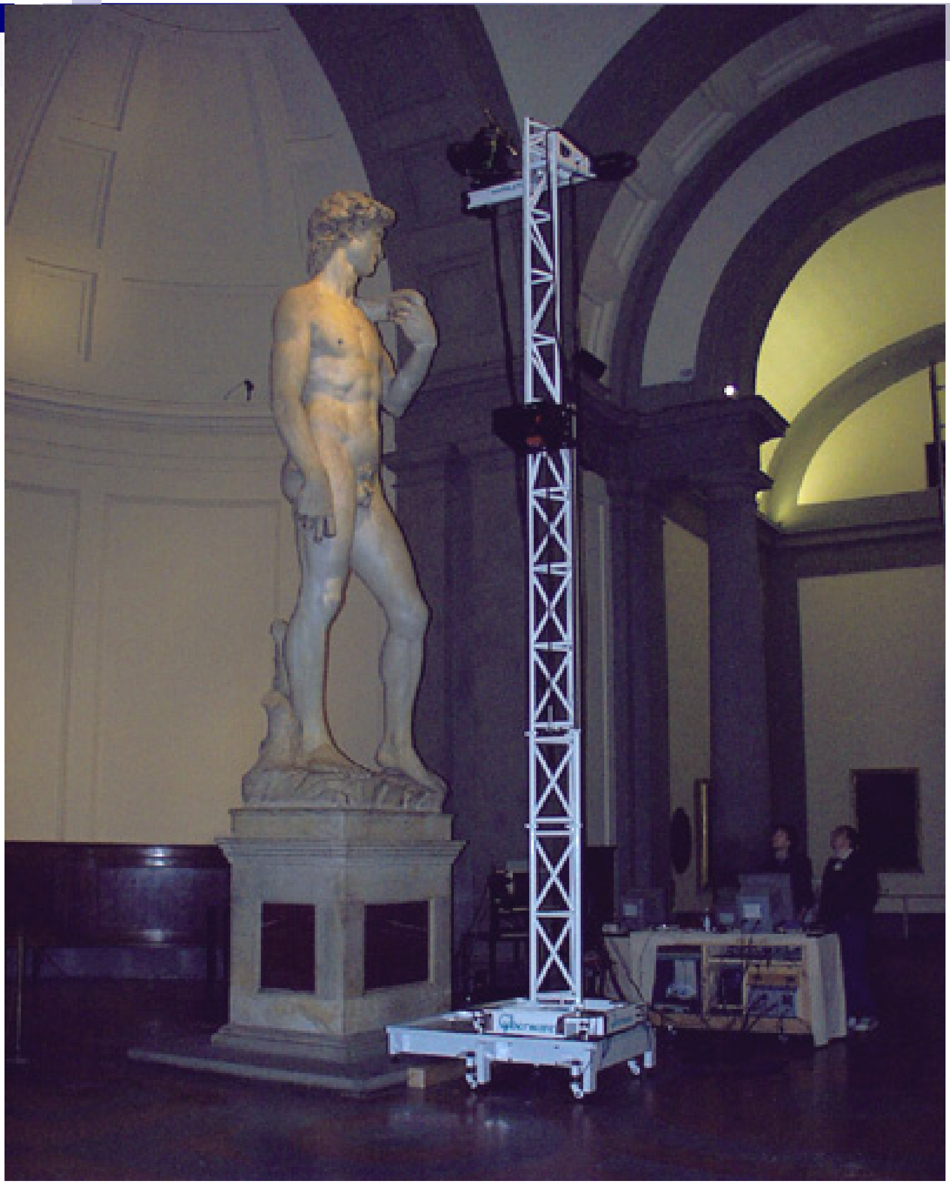
- The models consist of thousands of polygons
- To quickly test that they don't intersect, the bounding boxes are tested
- Sometimes a hierarchy of BB's is used

- The tighter the BB – the less "false alarms" we have



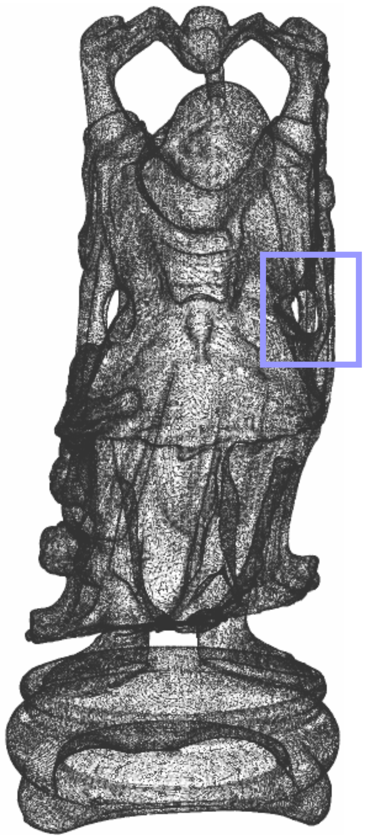
Sample

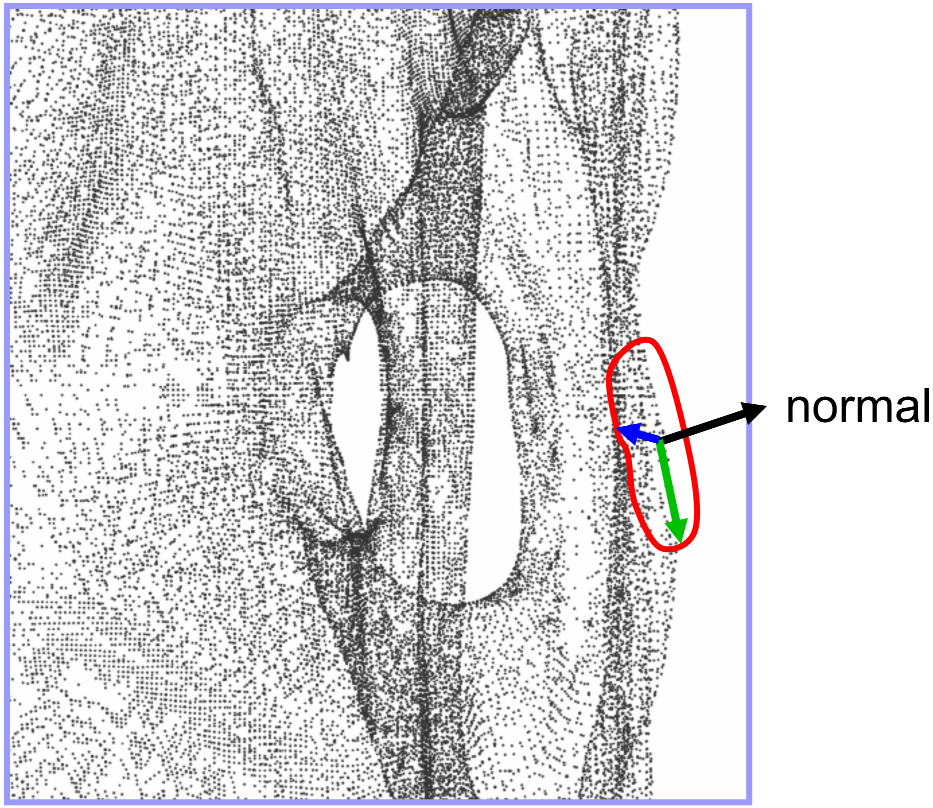
Scanned meshes



Point clouds

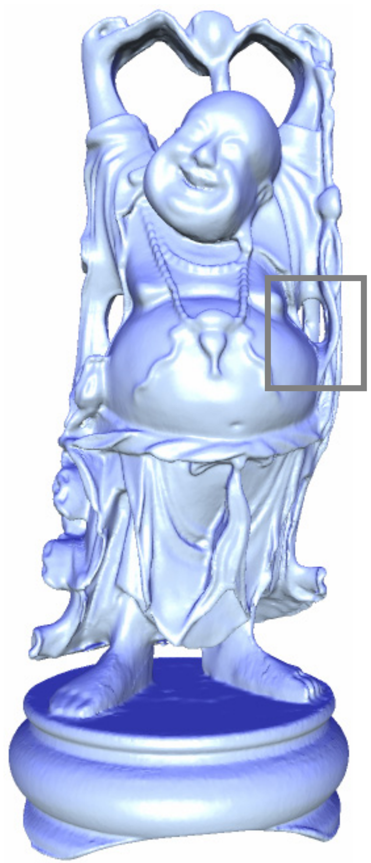
- Scanners give us raw point cloud data
- How to compute normals to shade the surface?

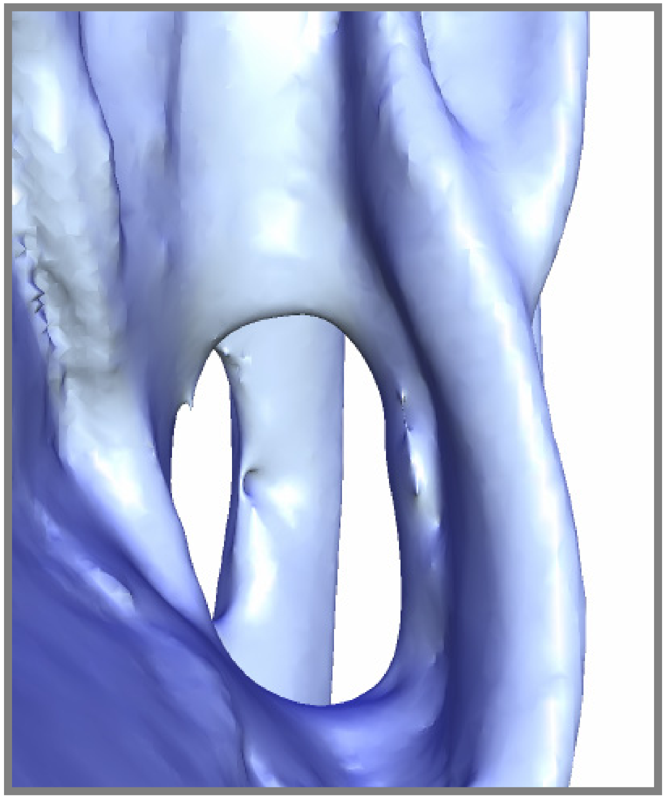




Point clouds

- Local PCA, take the third vector





Notations

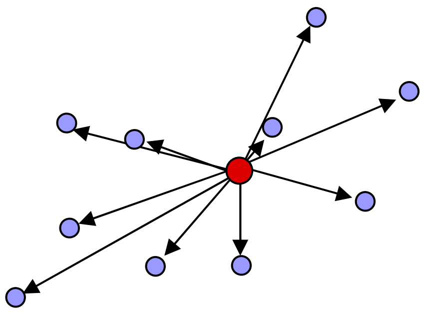
- Denote our data points by $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_\mathrm{n} \in R^d$

$$\mathbf{x_1} = \begin{pmatrix} x_1^1 \\ x_1^2 \\ \vdots \\ x_1^d \end{pmatrix}, \ \mathbf{x_2} = \begin{pmatrix} x_2^1 \\ x_2^2 \\ \vdots \\ x_2^d \end{pmatrix}, \ \ldots, \ \mathbf{x_n} = \begin{pmatrix} x_n^1 \\ x_n^2 \\ \vdots \\ x_n^d \end{pmatrix}$$

The origin of the new axes

- The origin is zero-order approximation of our data set (a point)
- It will be the center of mass:

$$\boxed{\mathbf{m} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{x}_i}$$



- It can be shown that:

$$\boxed{\mathbf{m} = \underset{\mathbf{x}}{\operatorname{argmin}} \sum_{i=1}^{n} \left\| \mathbf{x}_i - \mathbf{x} \right\|^2}$$

Scatter matrix

- Denote $\mathbf{y}_i = \mathbf{x}_i - \mathbf{m}, \quad i = 1, 2, \ldots, n$

$$\boxed{S = YY^T}$$

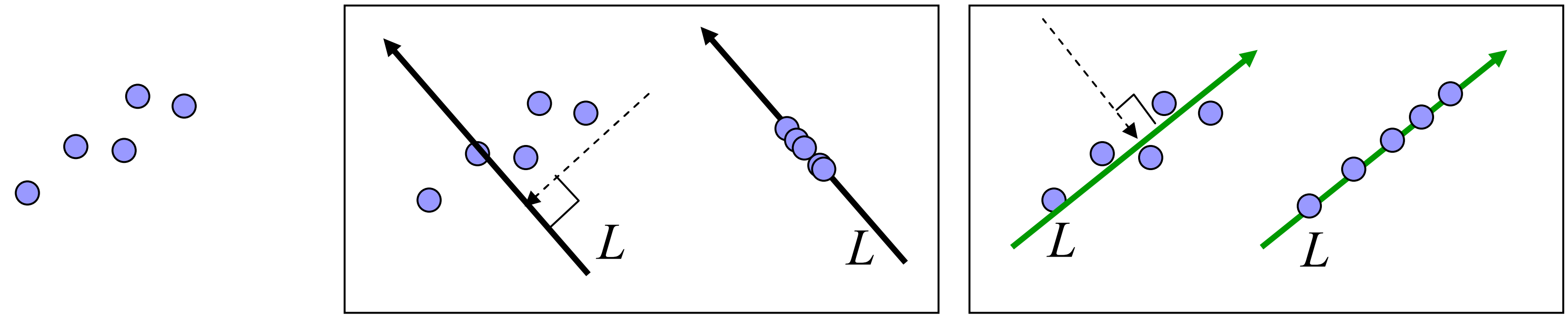
where Y is $d \times n$ matrix with $\mathbf{y}_k$ as columns ($k = 1, 2, \ldots, n$)

$$S = \underbrace{\begin{pmatrix} y_1^1 & y_2^1 & \cdots & y_n^1 \\ y_1^2 & y_2^2 & & y_n^2 \\ \vdots & \vdots & & \vdots \\ y_1^d & y_2^d & \cdots & y_n^d \end{pmatrix}}_{\mathrm{Y}} \underbrace{\begin{pmatrix} y_1^1 & y_1^2 & \cdots & y_1^d \\ y_2^1 & y_2^2 & \cdots & y_2^d \\ \vdots & & & \vdots \\ y_n^1 & y_n^2 & \cdots & y_n^d \end{pmatrix}}_{\mathrm{Y}^T}$$

Variance of projected points

- In a way, S measures variance (= scatterness) of the data in different directions.
- Let's look at a line L through the center of mass $\mathbf{m}$, and project our points $\mathbf{x}_i$ onto it. The variance of the projected points $\mathbf{x}'_i$ is:

$$\mathrm{var}(L) = \frac{1}{n}\sum_{i=1}^{n} \| \mathbf{x}'_i - \mathbf{m} \|^2$$

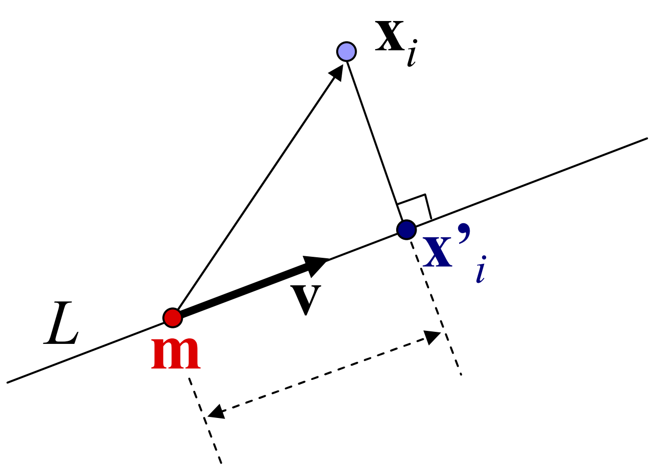


Original set　　Small variance　　Large variance

Variance of projected points

- Given a direction $\mathbf{v}$, $\|\mathbf{v}\| = 1$, the projection of $\mathbf{x}_i$ onto $L = \mathbf{m} + \mathbf{v}t$ is:

$$\boxed{\| \mathbf{x}_i' - \mathbf{m} \| = < \mathbf{v},\ \mathbf{x}_i - \mathbf{m} > / \| \mathbf{v} \| = < \mathbf{v},\ \mathbf{y}_i > = \mathbf{v}^T \mathbf{y}_i}$$



Variance of projected points

- So,

$$\operatorname{var}(L) = \frac{1}{n}\sum_{i=1}^{n} \| \mathbf{x}'_i - \mathbf{m} \|^2 = \frac{1}{n}\sum_{i=1}^{n} (\mathbf{v}^{\mathbf{T}}\mathbf{y}_i)^2 = \frac{1}{n} \| \mathbf{v}^{\mathbf{T}} Y \|^2 =$$

$$= \frac{1}{n} \| Y^T \mathbf{v} \|^2 = \frac{1}{n} < Y^T \mathbf{v},\ Y^{\mathbf{T}} \mathbf{v} > = \frac{1}{n} \mathbf{v}^{\mathbf{T}} Y Y^T \mathbf{v} = \frac{1}{n} \mathbf{v}^{\mathbf{T}} S \mathbf{v} = \frac{1}{n} < S\mathbf{v},\ \mathbf{v} >$$

$$\sum_{i=1}^{n} (\mathbf{v}^T \mathbf{y}_\mathbf{i})^2 = \sum_{i=1}^{n} \left(\begin{pmatrix} v^1 & v^2 & \cdots & v^d \end{pmatrix} \begin{pmatrix} y_i^1 \\ y_i^2 \\ \vdots \\ y_i^d \end{pmatrix} \right)^2 = \left\| \begin{pmatrix} v^1 & v^2 & \cdots & v^d \end{pmatrix} \begin{pmatrix} y_1^1 & y_2^1 & \cdots & y_n^1 \\ y_1^2 & y_2^2 & & y_n^2 \\ \vdots & \vdots & & \vdots \\ y_1^d & y_2^d & \cdots & y_n^d \end{pmatrix} \right\|^2 = \| \mathbf{v}^{\mathbf{T}} Y \|^2$$

Directions of maximal variance

- So, we have: $\mathrm{var}(L) = <S\mathbf{v}, \mathbf{v}>$
- Theorem:

  Let $f: \{\mathbf{v} \in R^d \mid ||\mathbf{v}|| = 1\} \to R,$

  $f(\mathbf{v}) = <S\mathbf{v}, \mathbf{v}>$ (and S is a symmetric matrix).

  Then, the extrema of f are attained at the eigenvectors of S.

- So, eigenvectors of S are directions of maximal/minimal variance!

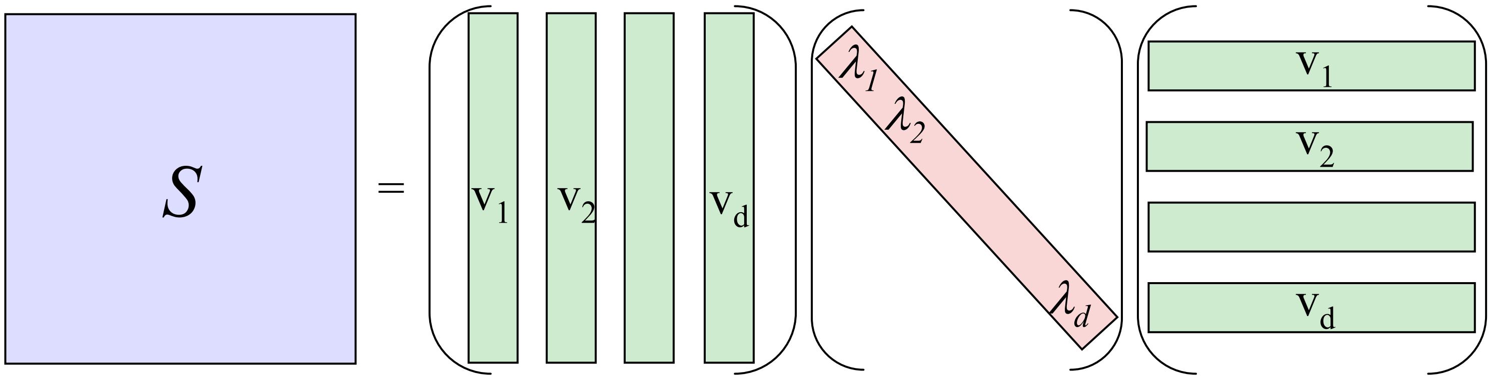
Summary so far

- We take the centered data vectors $\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_\mathrm{n} \in R^d$
- Construct the scatter matrix $S = YY^T$
- S measures the variance of the data points
- Eigenvectors of S are directions of maximal variance.

Scatter matrix - eigendecomposition

- $\boldsymbol{S}$ is symmetric

$\Rightarrow$ $\boldsymbol{S}$ has eigendecomposition: $\boldsymbol{S} = V\Lambda V^{\mathrm{T}}$



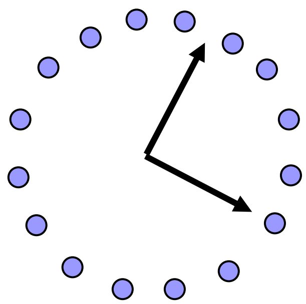
The eigenvectors form orthogonal basis

Principal components

- Eigenvectors that correspond to big eigenvalues are the directions in which the data has strong components (= large variance).
- If the eigenvalues are more or less the same – there is no preferable direction.

- Note: the eigenvalues are always non-negative.

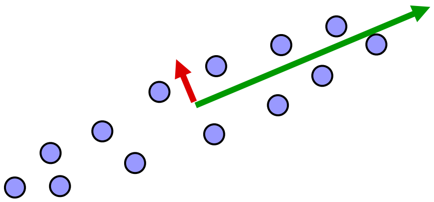
Principal components



- There's no preferable direction
- $\boldsymbol{S}$ looks like this:

$$V \begin{pmatrix} \lambda & \\ & \lambda \end{pmatrix} V^T$$

- Any vector is an eigenvector



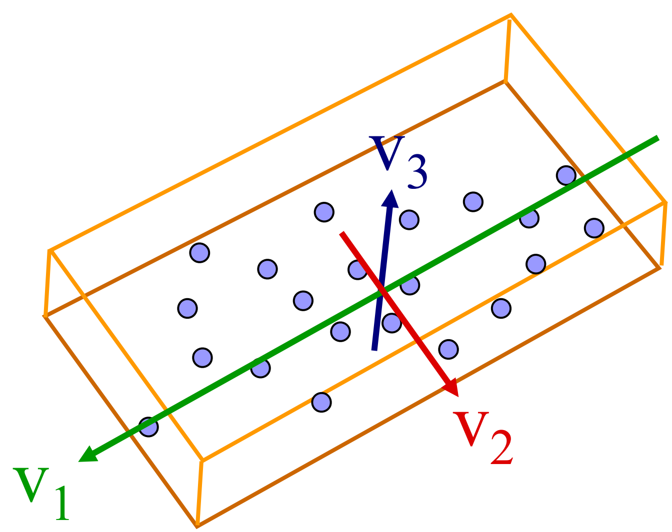
- There is a clear preferable direction
- $\boldsymbol{S}$ looks like this:

$$V \begin{pmatrix} \lambda & \\ & \mu \end{pmatrix} V^T$$

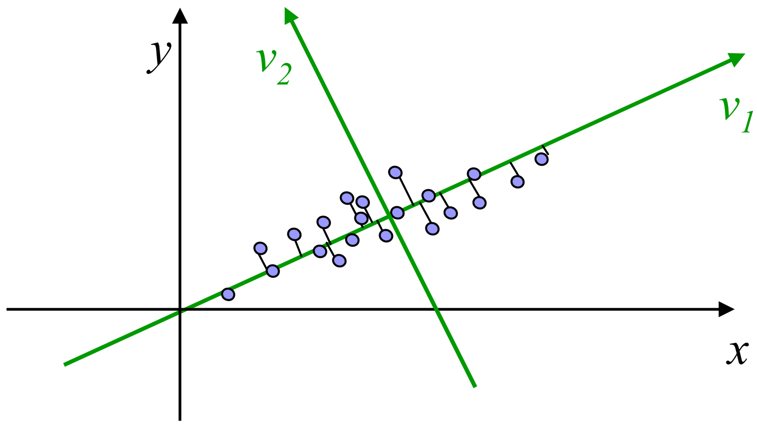
- μ is close to zero, much smaller than λ.

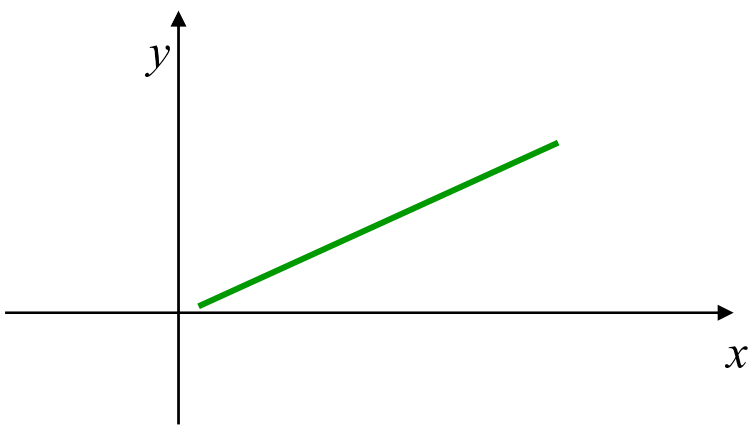
How to use what we got

- For finding oriented bounding box – we simply compute the bounding box with respect to the axes defined by the eigenvectors. The origin is at the mean point $\mathbf{m}$.

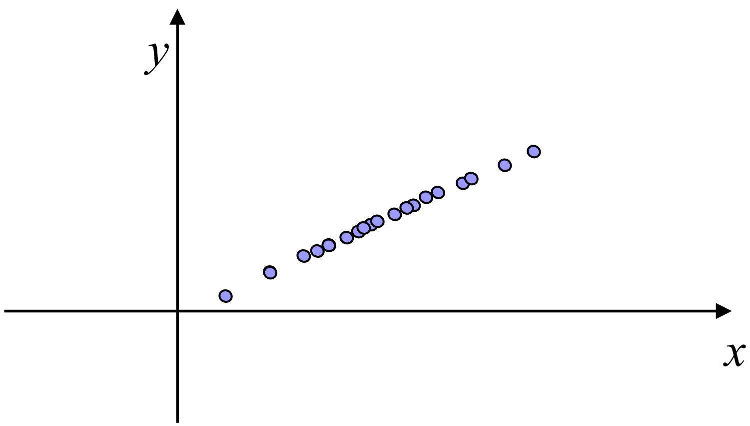


For approximation





This line segment approximates the original data set



The projected data set approximates the original data set

For approximation

- In general dimension d, the eigenvalues are sorted in descending order:

$$\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_d$$

- The eigenvectors are sorted accordingly.
- To get an approximation of dimension $d' < d$, we take the d' first eigenvectors and look at the subspace they span ($d' = 1$ is a line, $d' = 2$ is a plane…)

For approximation

- To get an approximating set, we project the original data points onto the chosen subspace:

$$\mathbf{x}_i = \mathbf{m} + \alpha_1 \mathbf{v}_1 + \alpha_2 \mathbf{v}_2 + \ldots + \alpha_{d'} \mathbf{v}_{d'} + \ldots + \alpha_d \mathbf{v}_d$$

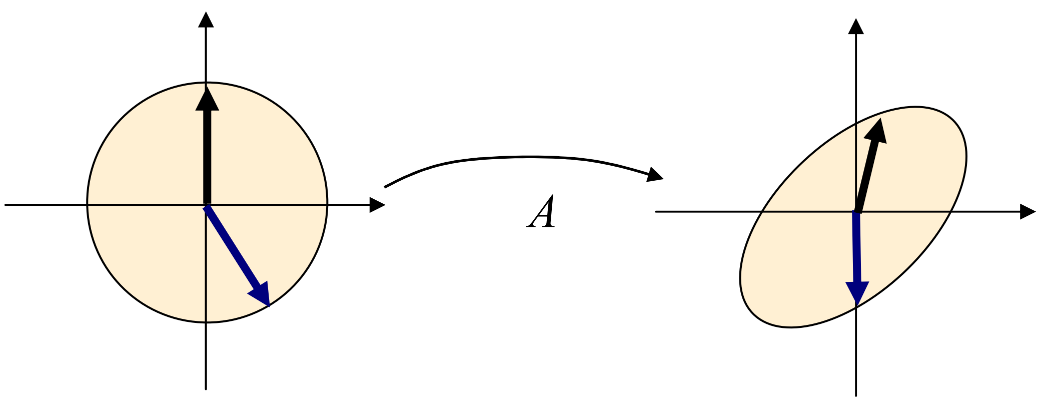
Projection:

$$\mathbf{x}_i' = \mathbf{m} + \underbrace{\alpha_1 \mathbf{v}_1 + \alpha_2 \mathbf{v}_2 + \ldots + \alpha_{d'} \mathbf{v}_{d'}} + 0 \cdot \mathbf{v}_{d'+1} + \ldots + 0 \cdot \mathbf{v}_d$$

SVD

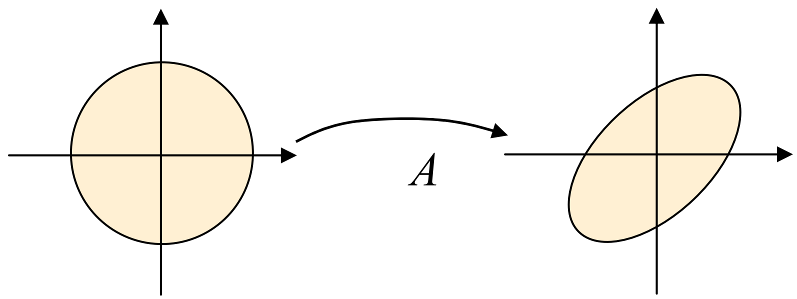
Geometric analysis of linear transformations

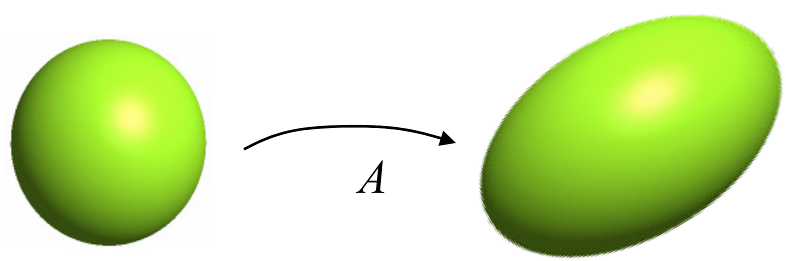
- We want to know what a linear transformation A does
- Need some simple and "comprehendible" representation of the matrix of A.
- Let's look what A does to some vectors
  - Since $A(\alpha\mathbf{v}) = \alpha A(\mathbf{v})$, it's enough to look at vectors $\mathbf{v}$ of <u>unit</u> length



The geometry of linear transformations

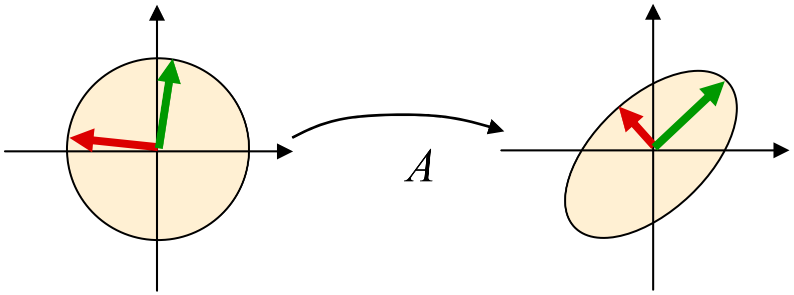
- A linear (non-singular) transform A always takes hyper-spheres to hyper-ellipses.

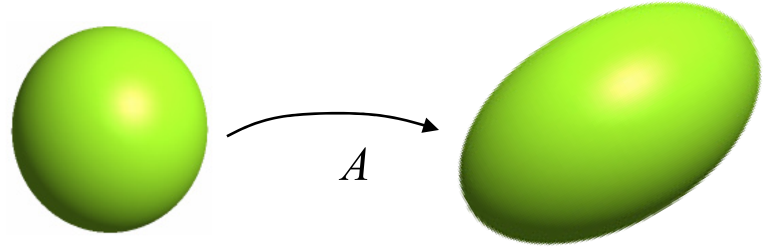




The geometry of linear transformations

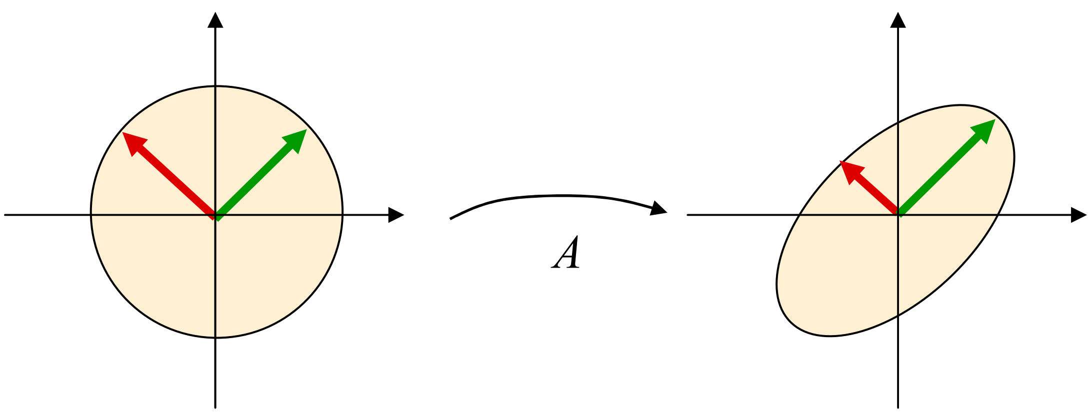
- Thus, one good way to understand what A does is to find which vectors are mapped to the "main axes" of the ellipsoid.





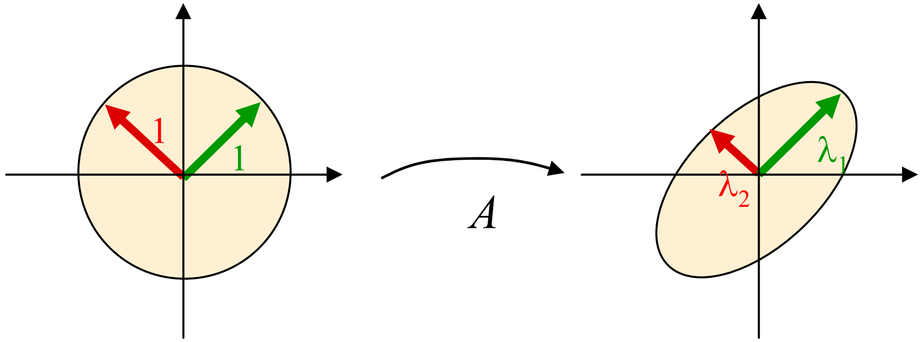
Geometric analysis of linear transformations

- If we are lucky: $\boxed{A = V \Lambda V^T, \quad V \text{ orthogonal}}$ (true if A is symmetric)
- The eigenvectors of A are the axes of the ellipse



Symmetric matrix: eigen decomposition

- In this case A is just a scaling matrix. The eigen decomposition of A tells us which orthogonal axes it scales, and by how much:

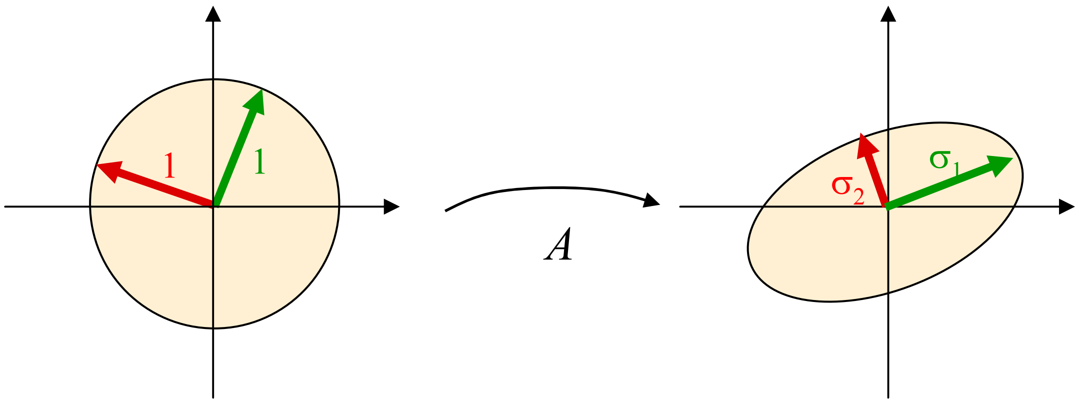


$$A = \left[\mathbf{v}_1 \ \mathbf{v}_2 \ \ldots \ \mathbf{v}_n \right] \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_n \end{bmatrix} \left[\mathbf{v}_1 \ \mathbf{v}_2 \ \ldots \ \mathbf{v}_n \right]^T$$

$$\boxed{A\mathbf{v}_i = \lambda_i \mathbf{v}_i}$$

General linear transformations: SVD

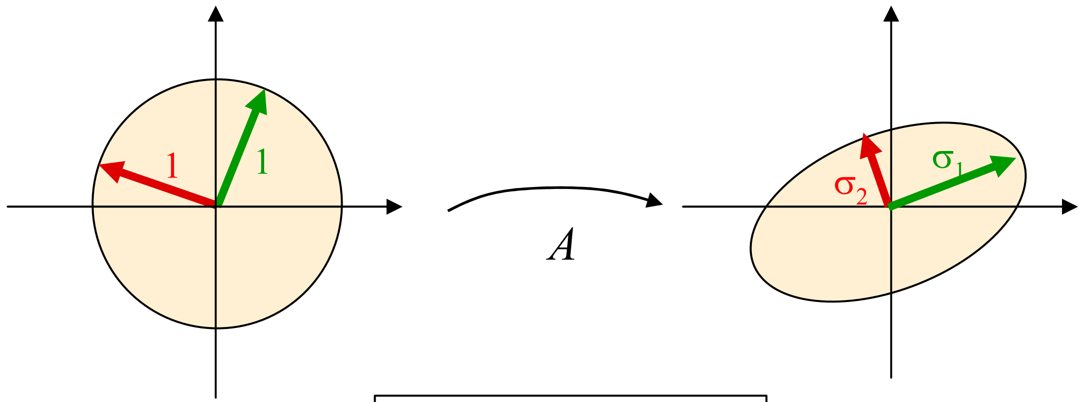
- In general A will also contain rotations, not just scales:



$$\boxed{A = U \Sigma V^T}$$

$$A = \left[\mathbf{u}_1\ \mathbf{u}_2\ \ldots\ \mathbf{u}_n\right] \begin{bmatrix} \sigma_1 & & & \\ & \sigma_2 & & \\ & & \ddots & \\ & & & \sigma_n \end{bmatrix} \left[\mathbf{v}_1\ \mathbf{v}_2\ \ldots\ \mathbf{v}_n\right]^T$$

General linear transformations: SVD



$$\boxed{AV = U\Sigma}$$

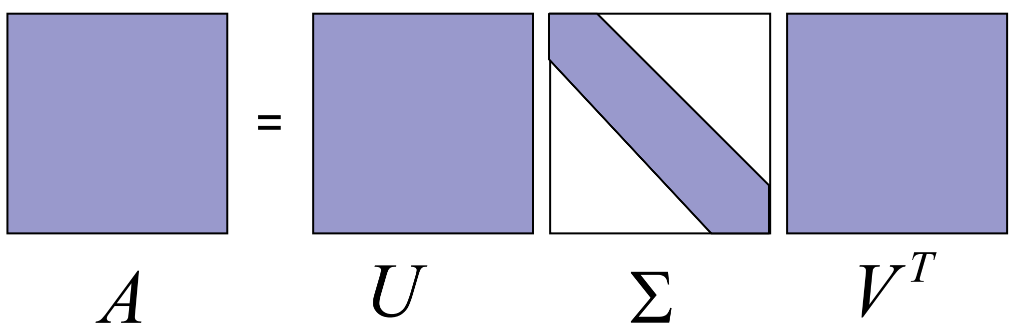
$$\begin{array}{cc} \textit{orthonormal} & \textit{orthonormal} \\ A\left[\mathbf{v}_1\ \mathbf{v}_2\ \ldots\ \mathbf{v}_n\right] = & \left[\mathbf{u}_1\ \mathbf{u}_2\ \ldots\ \mathbf{u}_n\right] \end{array} \begin{bmatrix} \sigma_1 & & & \\ & \sigma_2 & & \\ & & \ddots & \\ & & & \sigma_n \end{bmatrix}$$

$$\boxed{A\mathbf{v_i} = \sigma_i \mathbf{u_i}, \quad \sigma_i \geq 0}$$

SVD more formally

- SVD exists for any matrix
- Formal definition:
  - For square matrices $A \in R^{n\times n}$, there exist orthogonal matrices $U, V \in R^{n\times n}$ and a diagonal matrix Σ, such that all the diagonal values σ_i of Σ are non-negative and

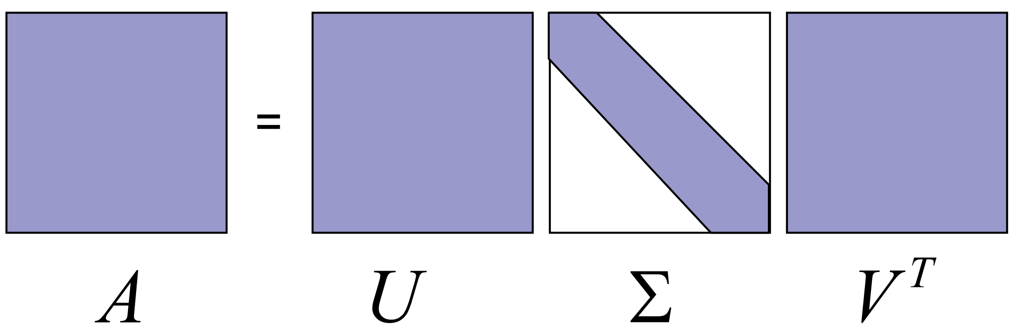
$$A = U \Sigma V^T$$



SVD more formally

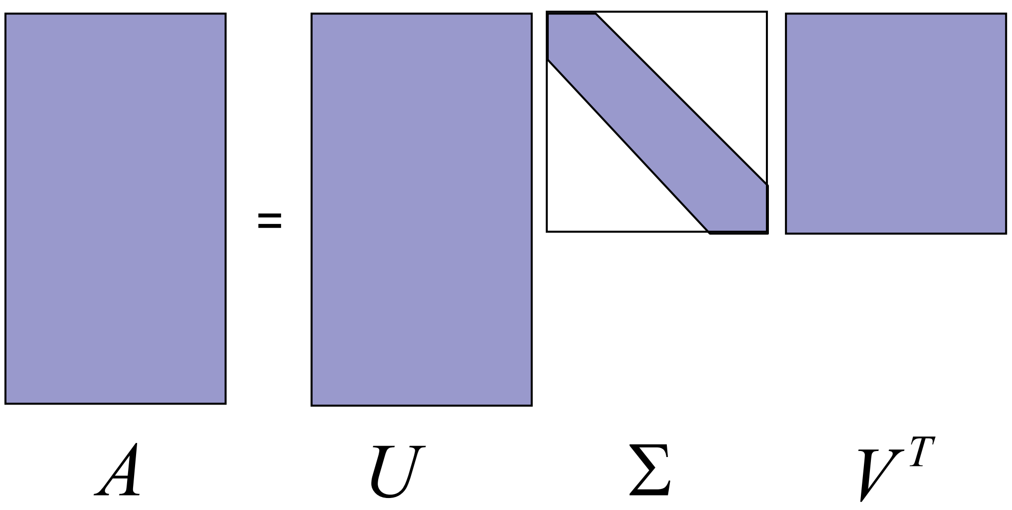
- The diagonal values of Σ ($\sigma_1, \ldots, \sigma_n$) are called the singular values. It is accustomed to sort them: $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_n$
- The columns of U ($\mathbf{u}_1, \ldots, \mathbf{u}_n$) are called the left singular vectors. They are the axes of the ellipsoid.
- The columns of V ($\mathbf{v}_1, \ldots, \mathbf{v}_n$) are called the right singular vectors. They are the preimages of the axes of the ellipsoid.

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



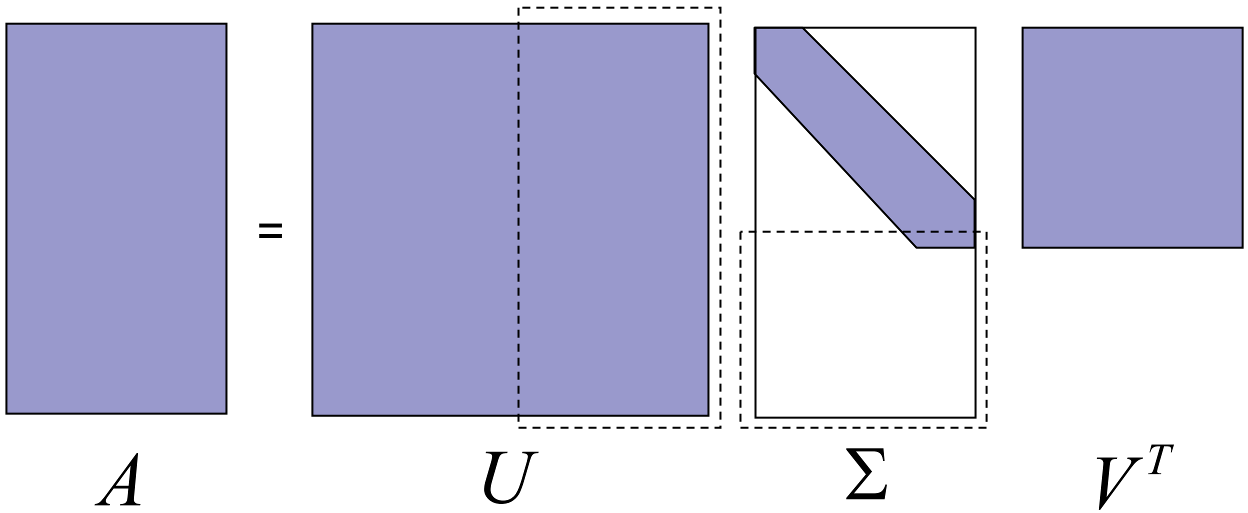
Reduced SVD

- For rectangular matrices, we have two forms of SVD. The reduced SVD looks like this:
  - The columns of U are orthonormal
  - Cheaper form for computation and storage



Full SVD

- We can complete U to a full orthogonal matrix and pad Σ by zeros accordingly

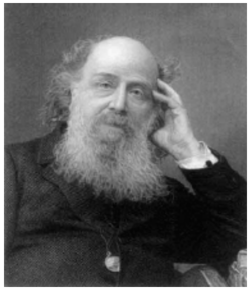


Some history

- SVD was discovered by the following people:

E. Beltrami
(1835 – 1900)

M. Jordan
(1838 – 1922)



J. Sylvester
(1814 – 1897)

E. Schmidt
(1876-1959)

H. Weyl
(1885-1955)

SVD is the "working horse" of linear algebra

- There are numerical algorithms to compute SVD. Once you have it, you have many things:
  - Matrix inverse $\rightarrow$ can solve square linear systems
  - Numerical rank of a matrix
  - Can solve least-squares systems
  - PCA
  - Many more…

Matrix inverse and solving linear systems

- Matrix inverse: $A = U \Sigma V^T$

$$A^{-1} = \left(U \Sigma V^T\right)^{-1} = \left(V^T\right)^{-1} \Sigma^{-1} U^{-1} =$$

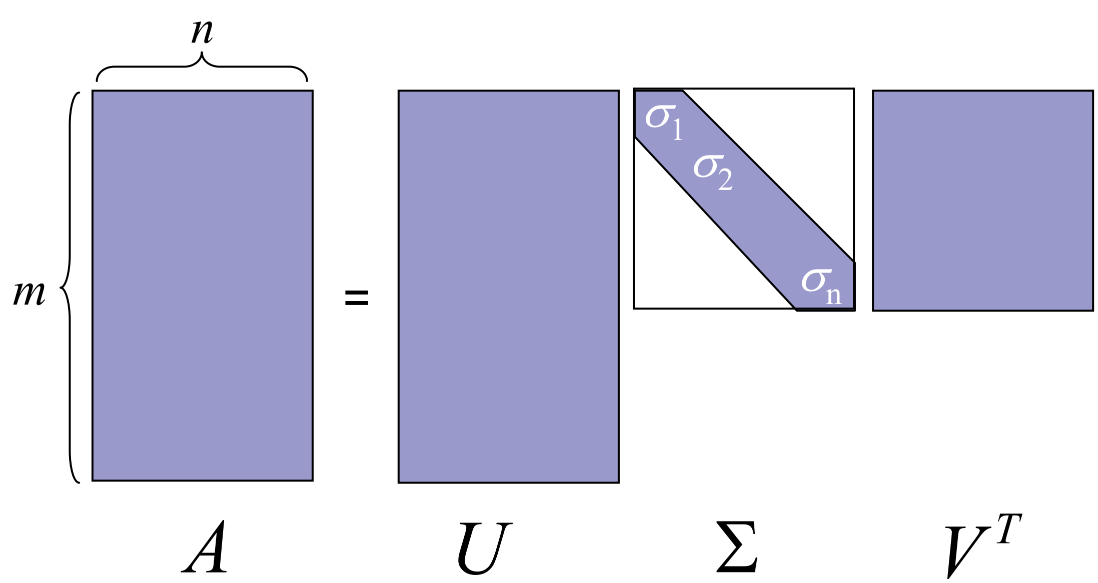
$$= V \begin{bmatrix} \frac{1}{\sigma_1} & & \\ & \ddots & \\ & & \frac{1}{\sigma_n} \end{bmatrix} U^T$$

- So, to solve $A\mathbf{x} = \mathbf{b}$

$$\mathbf{x} = V \Sigma^{-1} U^T \mathbf{b}$$

Matrix rank

- The rank of A is the number of non-zero singular values



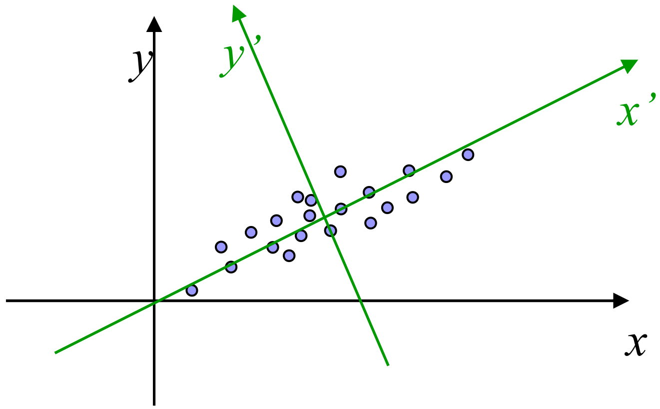
Numerical rank

- If there are very small singular values, then A is close to being singular. We can set a threshold t, so that $\mathrm{numeric\_rank}(A) = \#\{\sigma_i \mid \sigma_i > t\}$

- If $\mathrm{rank}(A) < n$ then A is singular. It maps the entire space R^n onto some subspace, like a plane (so A is some sort of projection).

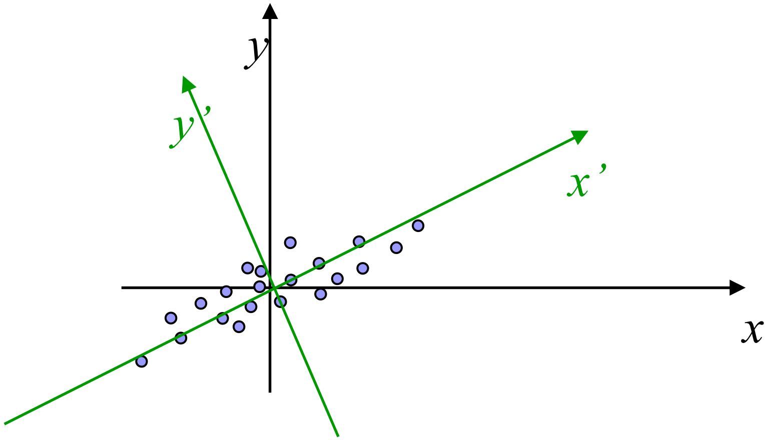
Back to PCA

- We wanted to find principal components



PCA

- Move the center of mass to the origin
  - $\mathbf{p}_i' = \mathbf{p}_i - \mathbf{m}$



PCA

- Constructed the matrix X of the data points.

$$X = \begin{bmatrix} | & | & & | \\ \mathbf{p}'_1 & \mathbf{p}'_2 & \cdots & \mathbf{p}'_n \\ | & | & & | \end{bmatrix}$$

- The principal axes are eigenvectors of $S = XX^T$

$$XX^T = S = U \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_d \end{bmatrix} U^T$$

PCA

- We can compute the principal components by SVD of X:

$$\underline{X = U\Sigma V^T}$$

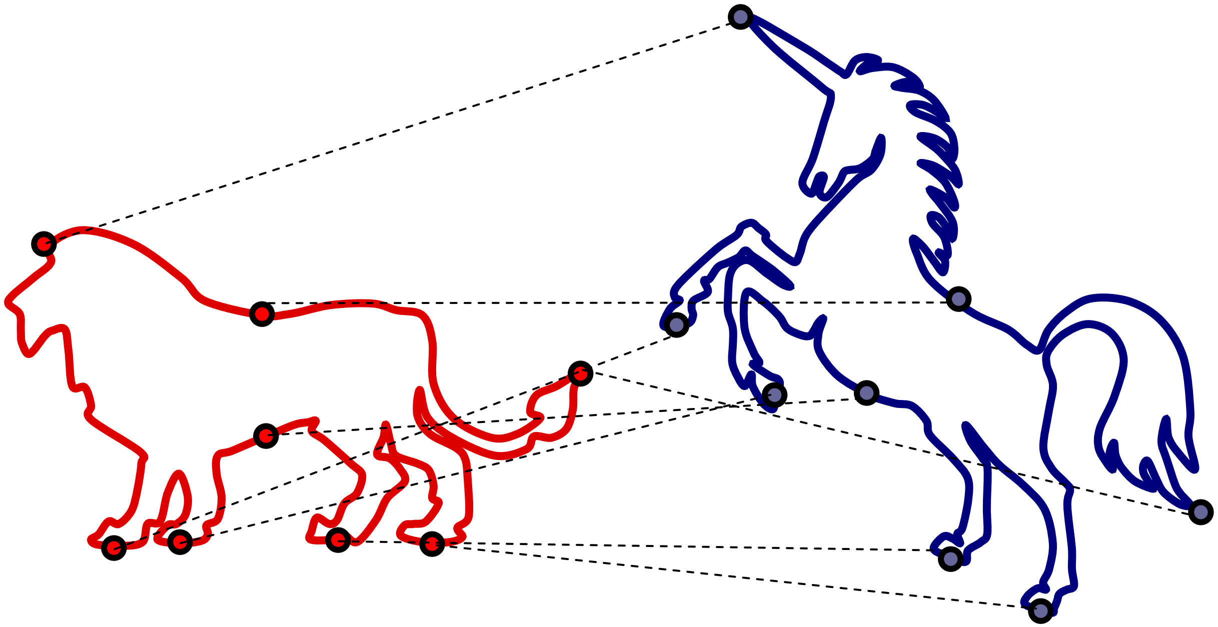
$$XX^T = U\Sigma V^T (U\Sigma V^T)^T =$$

$$= U\Sigma V^T V \Sigma^T U^T = \underline{U\tilde{\Sigma}^2 U^T}$$

- Thus, the left singular vectors of X are the principal components! We sort them by the size of the singular values of X.

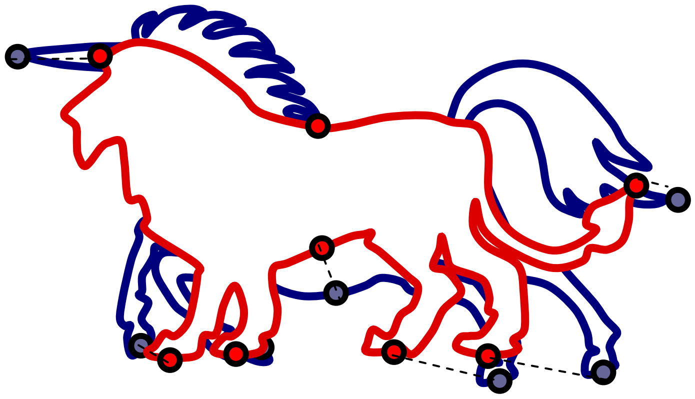
Shape matching

- We have two objects in correspondence
- Want to find the rigid transformation that aligns them



Shape matching

- When the objects are aligned, the lengths of the connecting lines are small.



Shape matching – formalization

- Align two point sets

$$P = \{\mathbf{p}_1,\ \ldots,\ \mathbf{p}_n\} \text{ and } Q = \{\mathbf{q}_1,\ \ldots,\ \mathbf{q}_n\}.$$

- Find a translation vector $\mathbf{t}$ and rotation matrix R so that:

$$\sum_{i=1}^{n} \left\| \mathbf{p}_i - (R\mathbf{q}_i + \mathbf{t}) \right\|^2 \text{ is minimized}$$

Shape matching – solution

- Turns out we can solve the translation and rotation separately.
- Theorem: if $(R, \mathbf{t})$ is the optimal transformation, then the points $\{\mathbf{p}_i\}$ and $\{R\mathbf{q}_i + \mathbf{t}\}$ have the same centers of mass.

$$\boxed{\bar{\mathbf{p}} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{p}_i} \qquad \boxed{\bar{\mathbf{q}} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{q}_i}$$

$$\bar{\mathbf{p}} = \frac{1}{n}\sum_{i=1}^{n} \left(R\mathbf{q}_i + \mathbf{t}\right)$$

$$\Downarrow$$

$$\bar{\mathbf{p}} = R\left(\frac{1}{n}\sum_{i=1}^{n} \mathbf{q}_i\right) + \mathbf{t} = R\bar{\mathbf{q}} + \mathbf{t}$$

$$\boxed{\mathbf{t} = \bar{\mathbf{p}} - R\bar{\mathbf{q}}}$$

Finding the rotation R

- To find the optimal R, we bring the centroids of both point sets to the origin:

$$\mathbf{p}_i' = \mathbf{p}_i - \mathbf{p} \qquad \mathbf{q}_i' = \mathbf{q}_i - \mathbf{q}$$

- We want to find R that minimizes

$$\sum_{i=1}^{n} \left\| \mathbf{p}_i' - R\mathbf{q}_i' \right\|^2$$

Finding the rotation R

$$\sum_{i=1}^{n} \left\| \mathbf{p}'_i - R\mathbf{q}'_i \right\|^2 = \sum_{i=1}^{n} \left(\mathbf{p}'_i - R\mathbf{q}'_i \right)^T \left(\mathbf{p}'_i - R\mathbf{q}'_i \right) =$$

$$= \sum_{i=1}^{n} \left(\mathbf{p}'^T_i \mathbf{p}'_i - \mathbf{p}'^T_i R\mathbf{q}'_i - \mathbf{q}'^T_i R^T \mathbf{p}'_i + \mathbf{q}'^T_i \underbrace{R^T R}_{\mathrm{I}} \mathbf{q}'_i \right)$$

These terms do not depend on R,
so we can ignore them in the minimization

Finding the rotation R

$$\min \sum_{i=1}^{n} \left(-\mathbf{p}_i'^T R \mathbf{q}_i' - \mathbf{q}_i'^T R^T \mathbf{p}_i' \right) = \max \sum_{i=1}^{n} \left(\mathbf{p}_i'^T R \mathbf{q}_i' + \underbrace{\mathbf{q}_i'^T R^T \mathbf{p}_i'}_{\text{this is a scalar}} \right).$$

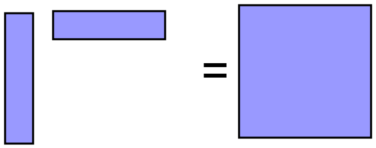
$$\mathbf{q}_i'^T R^T \mathbf{p}_i' = \left(\mathbf{q}_i'^T R^T \mathbf{p}_i' \right)^T = \mathbf{p}_i'^T R \mathbf{q}_i'$$

$$\Rightarrow \max \sum_{i=1}^{n} \left(2\mathbf{p}_i'^T R \mathbf{q}_i' \right) = \max \sum_{i=1}^{n} \left(\mathbf{p}_i'^T R \mathbf{q}_i' \right)$$

Finding the rotation R

$$\sum_{i=1}^{n}\left(\mathbf{p}_i'^{T} R \mathbf{q}_i'\right) = Trace\left(\sum_{i=1}^{n} R \mathbf{q}_i' \mathbf{p}_i'^{T}\right) = Trace\left(R \sum_{i=1}^{n} \mathbf{q}_i' \mathbf{p}_i'^{T}\right)$$

$$H = \sum_{i=1}^{n} \mathbf{q}_i' \mathbf{p}_i'^{T}$$



$3 \times 1 \quad 1 \times 3 \quad = \quad 3 \times 3$

$$Trace(A) = \sum_{i=1}^{n} A_{ii}$$

Finding the rotation R

- So, we want to find R that maximizes

$$Trace(RH)$$

- Theorem: if M is symmetric positive definite (all eigenvalues of M are positive) and B is any orthogonal matrix then

$$Trace(M) \geq Trace(BM)$$

- So, let's find R so that RH is symmetric positive definite. Then we know for sure that $Trace(RH)$ is maximal.

Finding the rotation R

- This is easy! Compute SVD of H:

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

- Define R: $R = VU^T$
- Check RH:

$$RH = \left(VU^T\right)\left(U \Sigma V^T\right) = V \Sigma V^T$$

This is a symmetric matrix,
Its eigenvalues are $\sigma_i \geq 0$
So RH is positive!

Summary of rigid alignment:

- Translate the input points to the centroids:

$$\mathbf{p}_i' = \mathbf{p}_i - \mathbf{p} \qquad\qquad \mathbf{q}_i' = \mathbf{q}_i - \mathbf{q}$$

- Compute the "covariance matrix"

$$H = \sum_{i=1}^{n} \mathbf{q}_i' \mathbf{p}_i'^T$$

- Compute the SVD of H:

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

- The optimal rotation is

$$R = V U^T$$

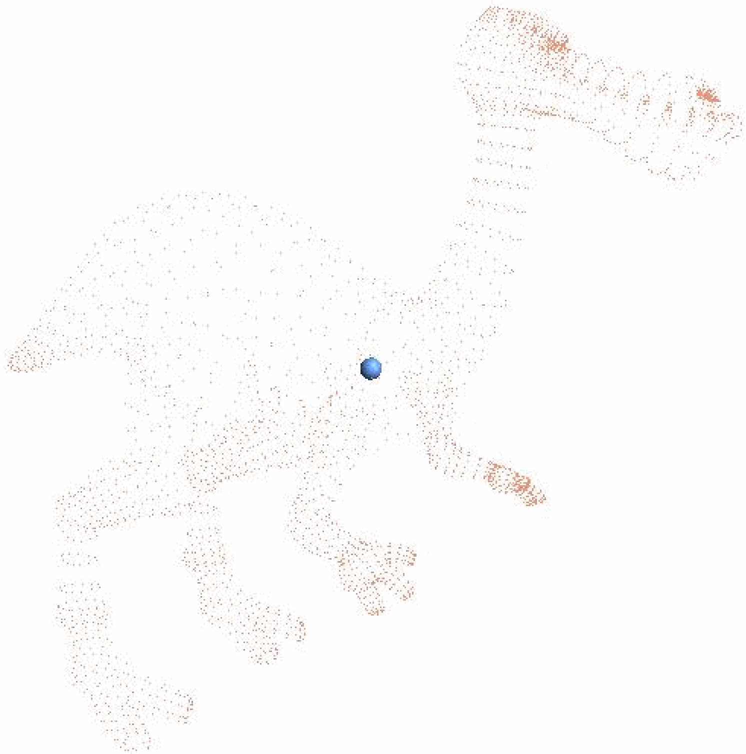
- The translation vector is

$$\mathbf{t} = \mathbf{p} - R\mathbf{q}$$

Complexity

- Numerical SVD is an expensive operation
- We always need to pay attention to the dimensions of the matrix we're applying SVD to.

Small somewhat related example



The End