G22.3033-008, Spring 2010

Geometric Modeling

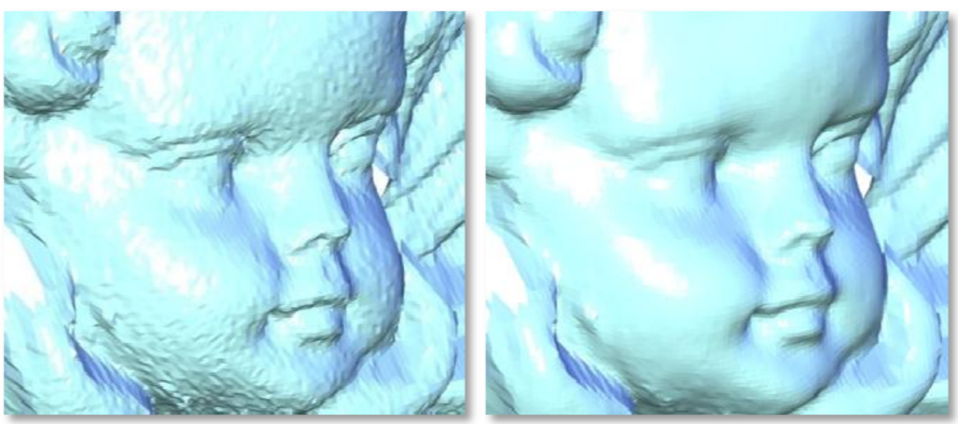
Linear algebra tools for geometric modeling

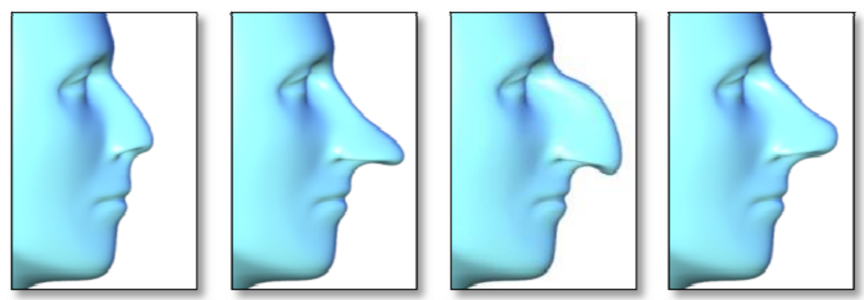
Least squares fitting

Motivation

- Why are we going over this again?
  - Many of the shape modeling methods presented in later lectures minimize functionals of the form

$$\mathbf{c}_{opt} = \underset{\mathbf{c}}{\operatorname{argmin}} \|A\mathbf{c} - \mathbf{b}\|^2$$

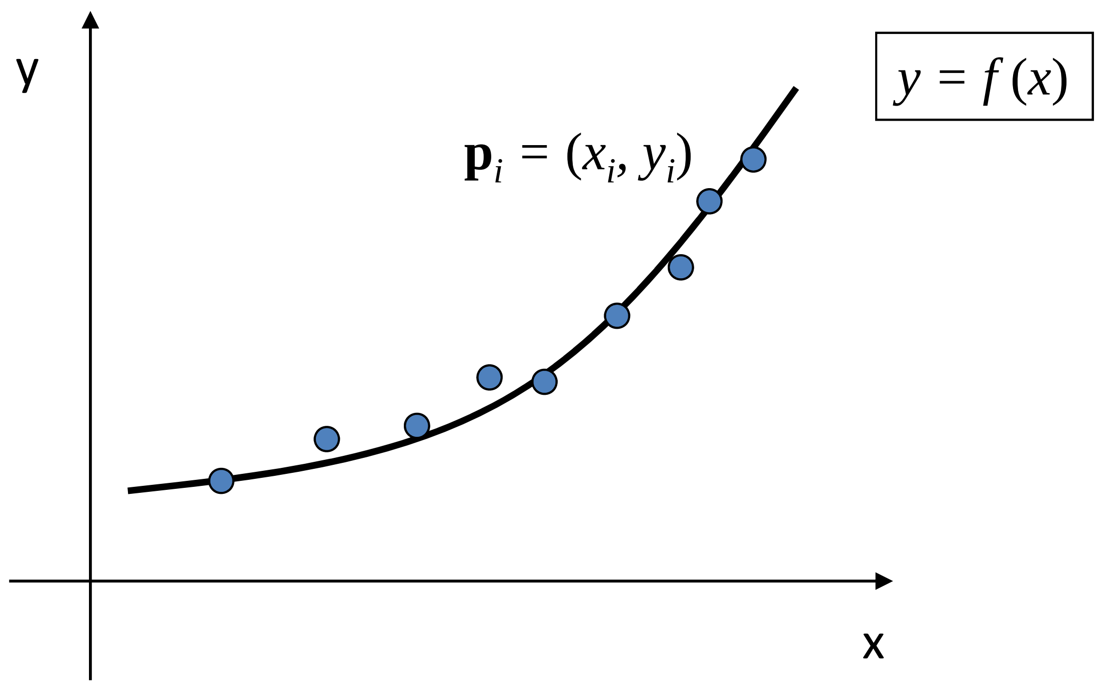




Least squares fitting

Motivation

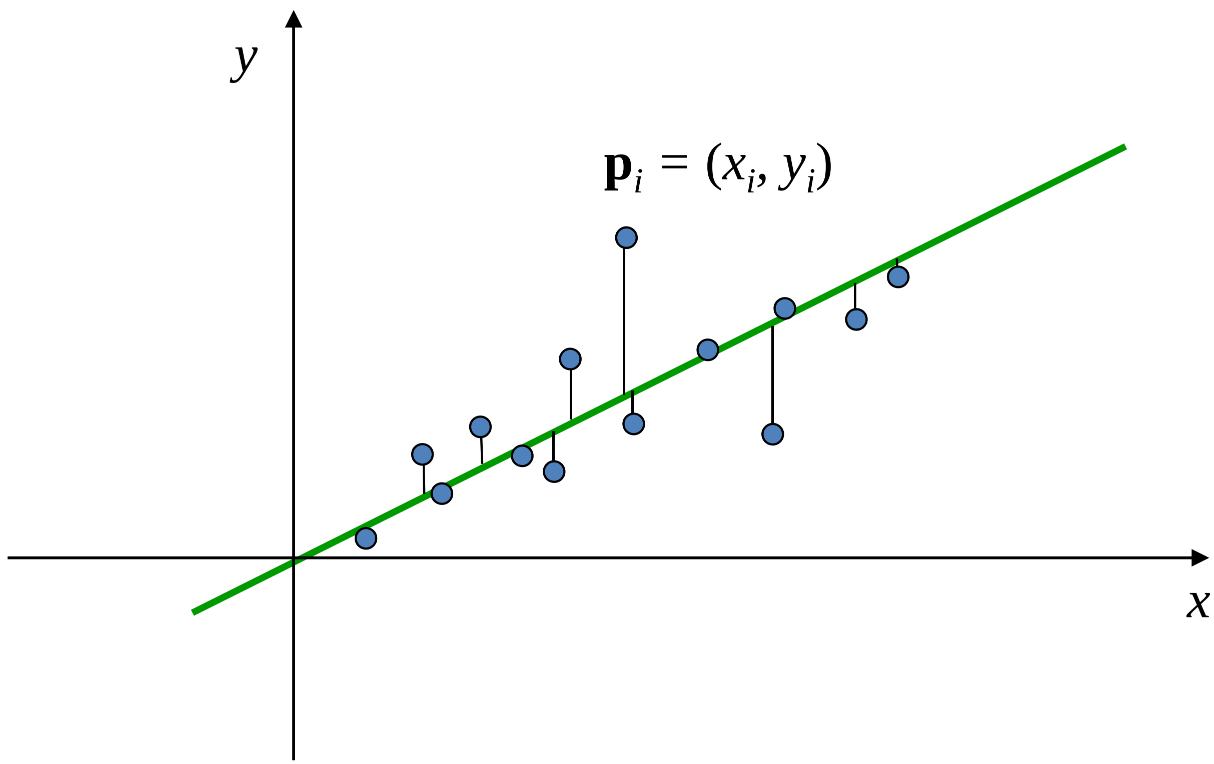
- Given data points, fit a function that is "close" to the points



Simple example

line fitting – 1st order polynomial in 2D

- y-offsets minimization



Simple example

line fitting – 1st order polynomial in 2D

- Find a line $y = ax + b$ that minimizes

$$E(a,b) = \sum_{i=1}^{n} [y_i - (ax_i + b)]^2$$

- $E(a,b)$ is quadratic in the unknown parameters a, b

- Another option would be, for example:

$$AbsErr(a,b) = \sum_{i=1}^{n} \left| y_i - (ax_i + b) \right|$$

- But – it is not differentiable, harder to minimize…

Simple example

line fitting – LS minimization

- To find optimal a, b we differentiate $E(a, b)$:

$$E(a,b) = \sum_{i=1}^{n} [y_i - (ax_i + b)]^2$$

$$\frac{\partial}{\partial a} E(a, b) = \sum_{i=1}^{n} (-2x_i)[y_i - (ax_i + b)] = 0$$

$$\frac{\partial}{\partial b} E(a, b) = \sum_{i=1}^{n} (-2)[y_i - (ax_i + b)] = 0$$

Simple example

line fitting – LS minimization

- We obtain two linear equations for a, b:

$$\sum_{i=1}^{n} (-2x_i)[y_i - (ax_i + b)] = 0$$

$$\sum_{i=1}^{n} (-2)[y_i - (ax_i + b)] = 0$$

Simple example

line fitting – LS minimization

- We get two linear equations for a, b:

$$(1)\quad \sum_{i=1}^{n} [x_i y_i - a x_i^2 - b x_i] = 0$$

$$(2)\quad \sum_{i=1}^{n} [y_i - a x_i - b] = 0$$

Simple example

line fitting – LS minimization

- We get two linear equations for a, b:

$$\left(\sum_{i=1}^{n} x_i^2\right) a + \left(\sum_{i=1}^{n} x_i\right) b = \sum_{i=1}^{n} x_i y_i$$

$$\left(\sum_{i=1}^{n} x_i\right) a + \left(\sum_{i=1}^{n} 1\right) b = \sum_{i=1}^{n} y_i$$

Simple example

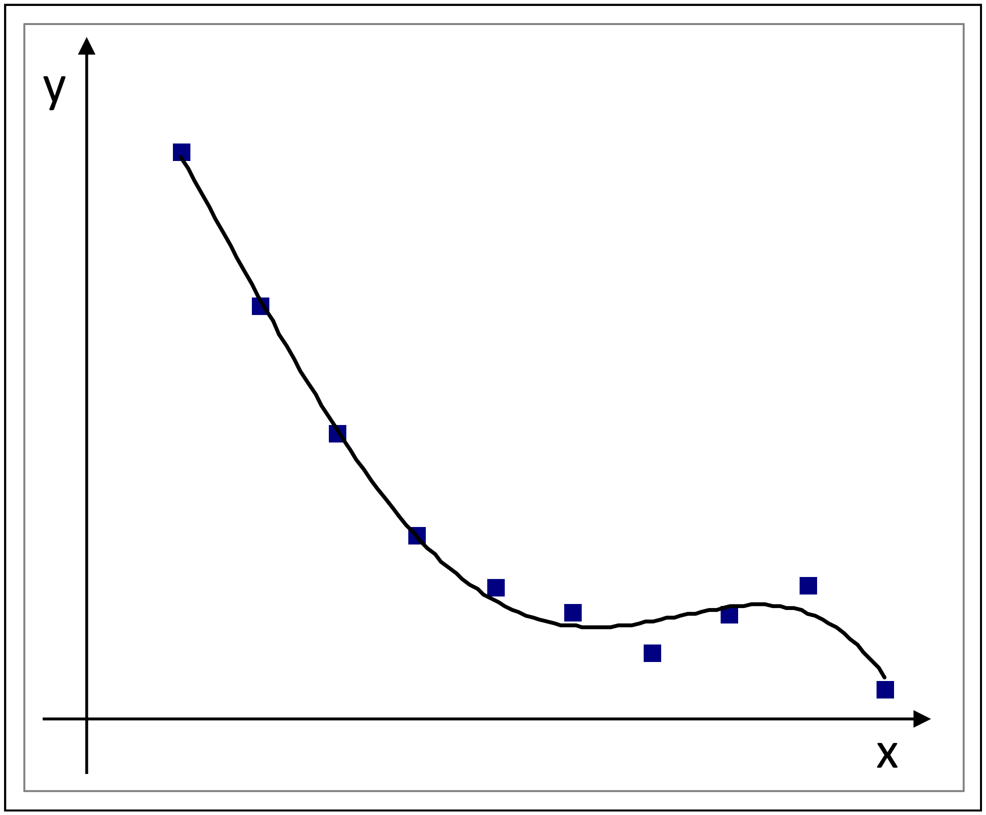
line fitting – LS minimization

- Solve for a, b using e.g. Gauss elimination

- Question: why the solution is the *minimum* for the error function?

$$E(a, b) = \sum_{i=1}^{n} [y_i - (ax_i + b)]^2$$

Fitting polynomials



Fitting polynomials

- Decide on the degree of the polynomial, k
- Want to fit $f(x) = a_k x^k + a_{k-1} x^{k-1} + \ldots + a_1 x + a_0$
- Minimize:

$$E(a_0, a_1, \ldots, a_k) = \sum_{i=1}^{n} [y_i - (a_k x_i^k + a_{k-1} x_i^{k-1} + \ldots + a_1 x_i + a_0)]^2$$

$$\frac{\partial}{\partial a_m} E(a_0, \ldots, a_k) = \sum_{i=1}^{n} (-2x^m)[y_i - (a_k x_i^k + a_{k-1} x_i^{k-1} + \ldots + a_0)] = 0$$

Fitting polynomials

- We get a linear system of $k+1$ equations in $k+1$ variables

$$\begin{pmatrix} \sum_{i=1}^{n} 1 & \sum_{i=1}^{n} x_i & \cdots & \sum_{i=1}^{n} x_i^{\,k} \\ \sum_{i=1}^{n} x_i & \sum_{i=1}^{n} x_i^{\,2} & \cdots & \sum_{i=1}^{n} x_i^{\,k+1} \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{i=1}^{n} x_i^{\,k} & \sum_{i=1}^{n} x_i^{\,k+1} & \cdots & \sum_{i=1}^{n} x_i^{\,2k} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_k \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^{n} 1 \cdot y_i \\ \sum_{i=1}^{n} x_i y_i \\ \\ \sum_{i=1}^{n} x_i^{\,k} y_i \end{pmatrix}$$

General parametric fitting

- We can use this approach to fit any function $f(\mathbf{x})$
  - Specified by parameters $c_1, c_2, c_3, \ldots$
  - The expression $f(\mathbf{x})$ linearly depends on the parameters.
- $f(\mathbf{x}) = c_1 f_1(\mathbf{x}) + c_2 f_2(\mathbf{x}) + \ldots + c_k f_k(\mathbf{x})$
- Minimize – find best $c_1, c_2, c_3 \ldots$:

$$\sum_{i=1}^{n} \|f(\mathbf{p}_i) - f_i\|^2 = \sum_{i=1}^{n} \|\sum_{j=1}^{k} \mathrm{c}_j f_j(\mathbf{p}_i) - f_i\|^2$$

Solving linear systems in LS sense

- Let's look at the problem a little differently:
  - We have data points $\mathbf{p}_i$ and desired function values f_i
  - We would like :

$$\forall\ i = 1, \ldots, n: \quad f(\mathbf{p}_i) = f_i$$

  - Strict interpolation is in general not possible
    - In polynomials: n+1 points define a unique interpolation polynomial of degree n.
    - So, if we have 1000 points and want a cubic polynomial, we probably won't find it…

Solving linear systems in LS sense

- We have an over-determined linear system $n \times k$:

$$f(\mathbf{p}_1) = c_1\, f_1(\mathbf{p}_1) + c_2\, f_2(\mathbf{p}_1) + \ldots + c_k\, f_k(\mathbf{p}_1) = f_1$$
$$f(\mathbf{p}_2) = c_1\, f_1(\mathbf{p}_2) + c_2\, f_2(\mathbf{p}_2) + \ldots + c_k\, f_k(\mathbf{p}_2) = f_2$$
$$\ldots$$
$$f(\mathbf{p}_n) = c_1\, f_1(\mathbf{p}_n) + c_2\, f_2(\mathbf{p}_n) + \ldots + c_k\, f_k(\mathbf{p}_n) = f_n$$

Solving linear systems in LS sense

- In matrix form:

$$\begin{pmatrix} f_1(\mathbf{p}_1) & f_2(\mathbf{p}_1) & \cdots & f_k(\mathbf{p}_1) \\ f_1(\mathbf{p}_2) & f_2(\mathbf{p}_2) & \cdots & f_k(\mathbf{p}_2) \\ & & \cdots & \\ \vdots & \vdots & & \vdots \\ f_1(\mathbf{p}_n) & f_2(\mathbf{p}_n) & \cdots & f_k(\mathbf{p}_n) \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_k \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{pmatrix}$$

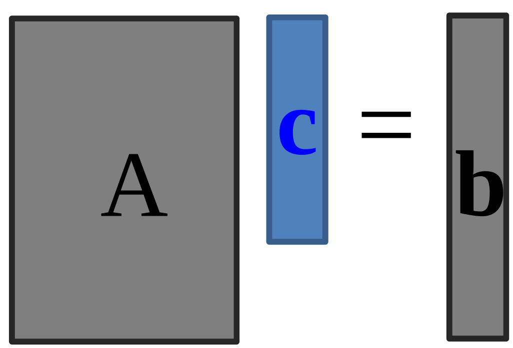
Solving linear systems in LS sense

- In matrix form:

$$\mathrm{A}\mathbf{c} = \mathbf{b}$$

where $\mathrm{A} = (\, f_j(\mathbf{p}_i)\,)_{i,j}$ is a rectangular $n \times k$ matrix, $n > k$

$\mathbf{c} = (c_1, c_2, \ldots, c_k)^{\mathrm{T}}$ $\quad$ $\mathbf{b} = (f_1, f_2, \ldots, f_n)^{\mathrm{T}}$



Solving linear systems in LS sense

- More constrains than variables – no exact solutions generally exist
- We want to find something that is an "approximate solution":

$$\mathbf{c}_{opt} = \underset{\mathbf{c}}{\operatorname{argmin}} \,\|A\mathbf{c} - \mathbf{b}\|^2$$

Finding the LS solution

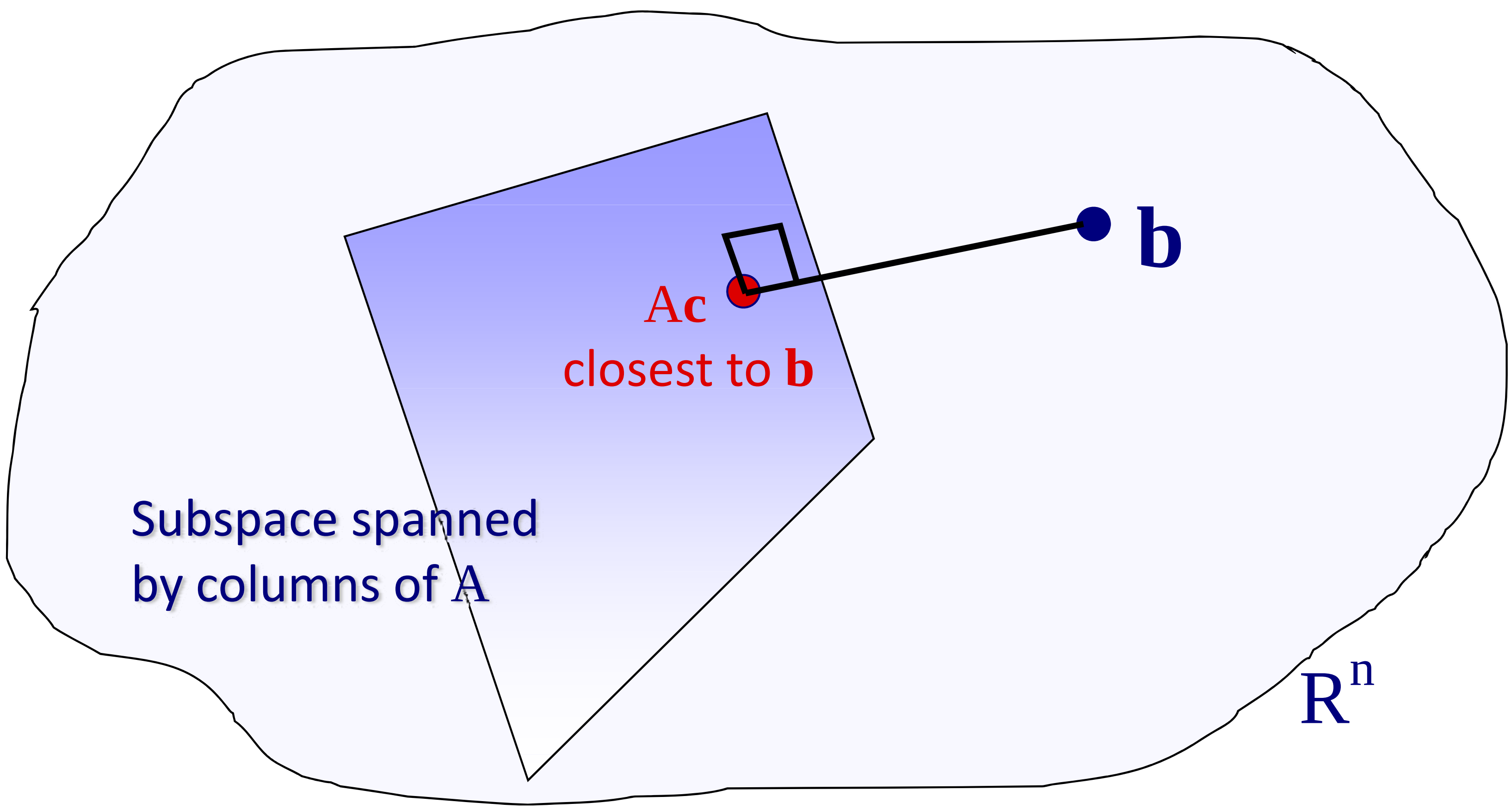
- $\mathbf{c} \in \mathrm{R}^k$
- $A\mathbf{c} \in \mathrm{R}^n$
- As we vary $\mathbf{c}$, $A\mathbf{c}$ varies over the linear subspace of R^n spanned by the columns of A:

$$A\mathbf{c} = \begin{pmatrix} A_1 & A_2 & \cdots & A_k \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ \cdot \\ \cdot \\ c_k \end{pmatrix} = c_1 A_1 + c_2 A_2 + \ldots + c_k A_k$$

This is also known as the column space of A

Finding the LS solution

- We want to find the closest $\mathrm{A}\mathbf{c}$ to $\mathbf{b}$: $\min_{\mathbf{c}} \|\mathrm{A}\mathbf{c} - \mathbf{b}\|^2$



Finding the LS solution

- The point $\mathrm{A}\mathbf{c}$ closest to $\mathbf{b}$ satisfies:

$$(\mathrm{A}\mathbf{c} - \mathbf{b}) \perp \{\text{subspace of A's columns}\}$$

$$\Updownarrow$$

$$\forall \text{ column } A_i: \quad \langle A_i, \mathrm{A}\mathbf{c} - \mathbf{b} \rangle = 0$$

$$\forall\, i, \quad A_i^{\mathrm{T}}(\mathrm{A}\mathbf{c} - \mathbf{b}) = 0$$

$$\Updownarrow$$

$$\mathrm{A}^{\mathrm{T}}(\mathrm{A}\mathbf{c} - \mathbf{b}) = 0$$

$$\boxed{(\mathrm{A}^{\mathrm{T}}\mathrm{A})\mathbf{c} = \mathrm{A}^{\mathrm{T}}\mathbf{b}}$$

These are called **the normal equations**

Finding the LS solution

- We have a square symmetric system $(\mathrm{A}^{\mathrm{T}}\mathrm{A})\mathbf{c} = \mathrm{A}^{\mathrm{T}}\mathbf{b}$

  $(k \times k)$

- If A has full rank (the columns of A are linearly independent) then $(\mathrm{A}^{\mathrm{T}}\mathrm{A})$ is invertible.

$$\min_{\mathbf{c}} \left\| \mathrm{A}\mathbf{c} - \mathbf{b} \right\|^2$$

$$\Downarrow$$

$$\mathbf{c} = \left(\mathrm{A}^T \mathrm{A}\right)^{-1} \mathrm{A}^T \mathbf{b}$$

Weighted least squares

- If each constraint has a weight in the energy:

$$\min_{\mathbf{c}} \sum_{i=1}^{n} w_i \,(f_{\mathbf{c}}(\mathbf{p}_i) - f_i)^2$$

- The weights $w_i > 0$ and don't depend on $\mathbf{c}$
- Then:

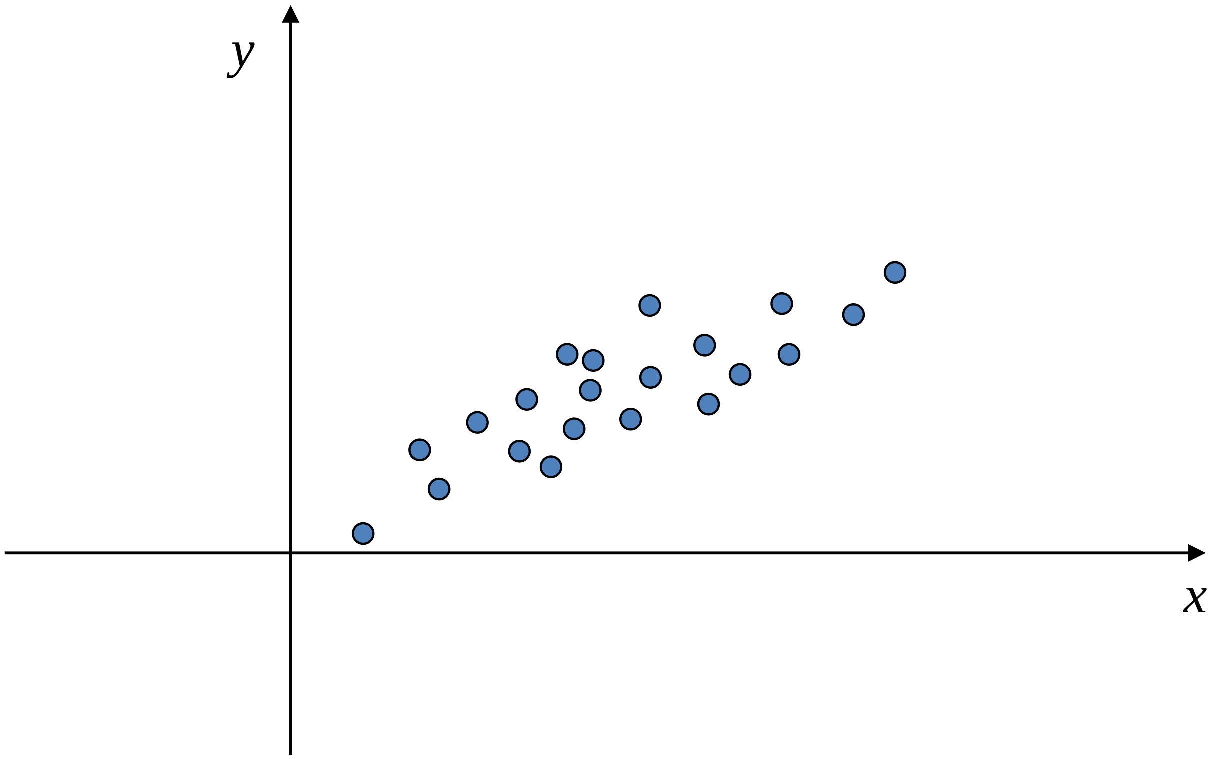
$$\min\,(\mathrm{A}\mathbf{c} - \mathbf{b})^{\mathrm{T}}\,\mathrm{W}^{\mathrm{T}}\mathrm{W}\,(\mathrm{A}\mathbf{c} - \mathbf{b}) \quad \text{where } \mathrm{W} = (w_i)_{ii}$$

$$\boxed{(\mathrm{A}^{\mathrm{T}}\mathrm{W}^2\,\mathrm{A})\,\mathbf{c} \;=\; \mathrm{A}^{\mathrm{T}}\mathrm{W}^2\,\mathbf{b}}$$

Principal Component Analysis

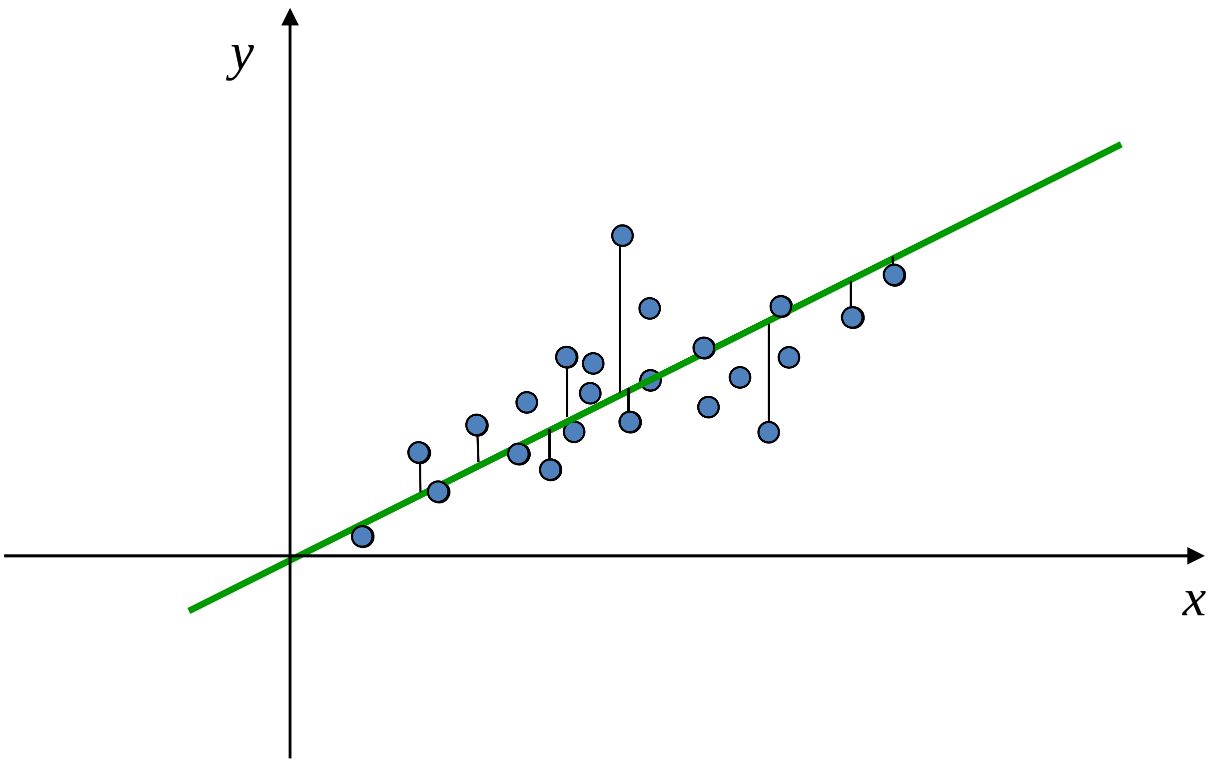
But first, reminder about eigenvectors and eigenvalues

Motivation



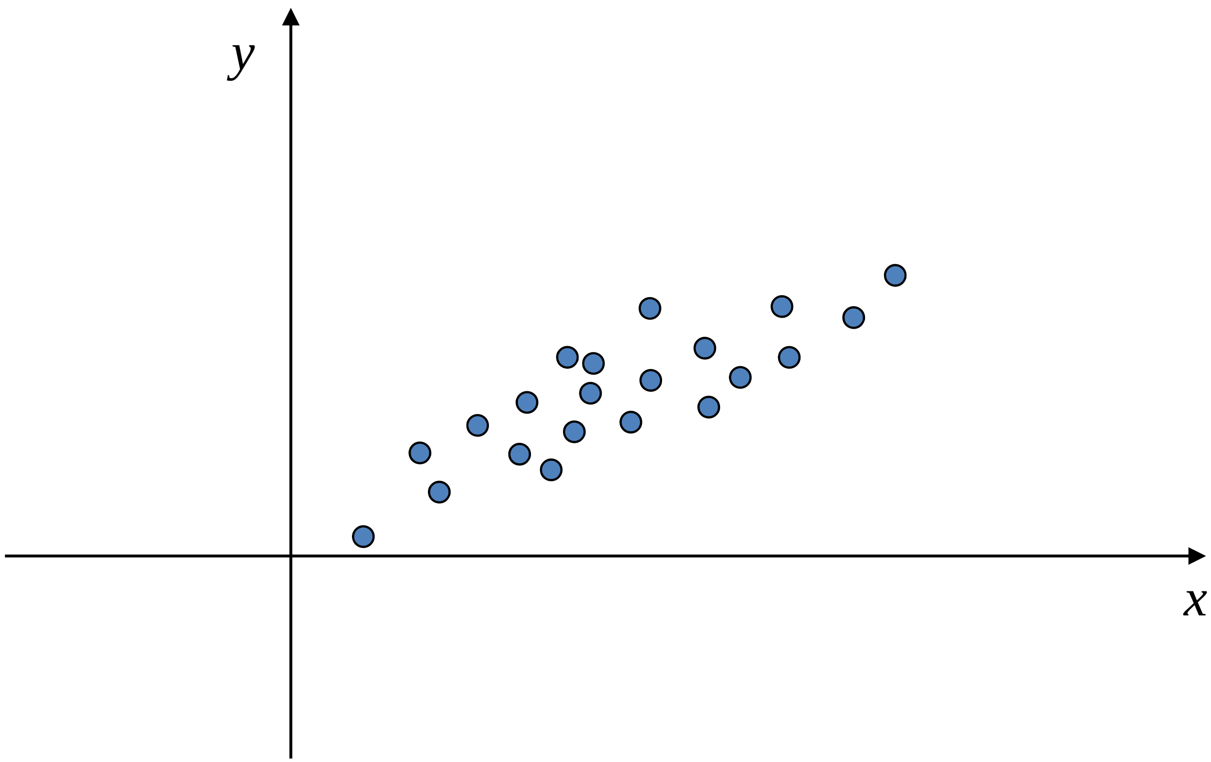
- Given a set of points, find the best line that approximates them

Motivation



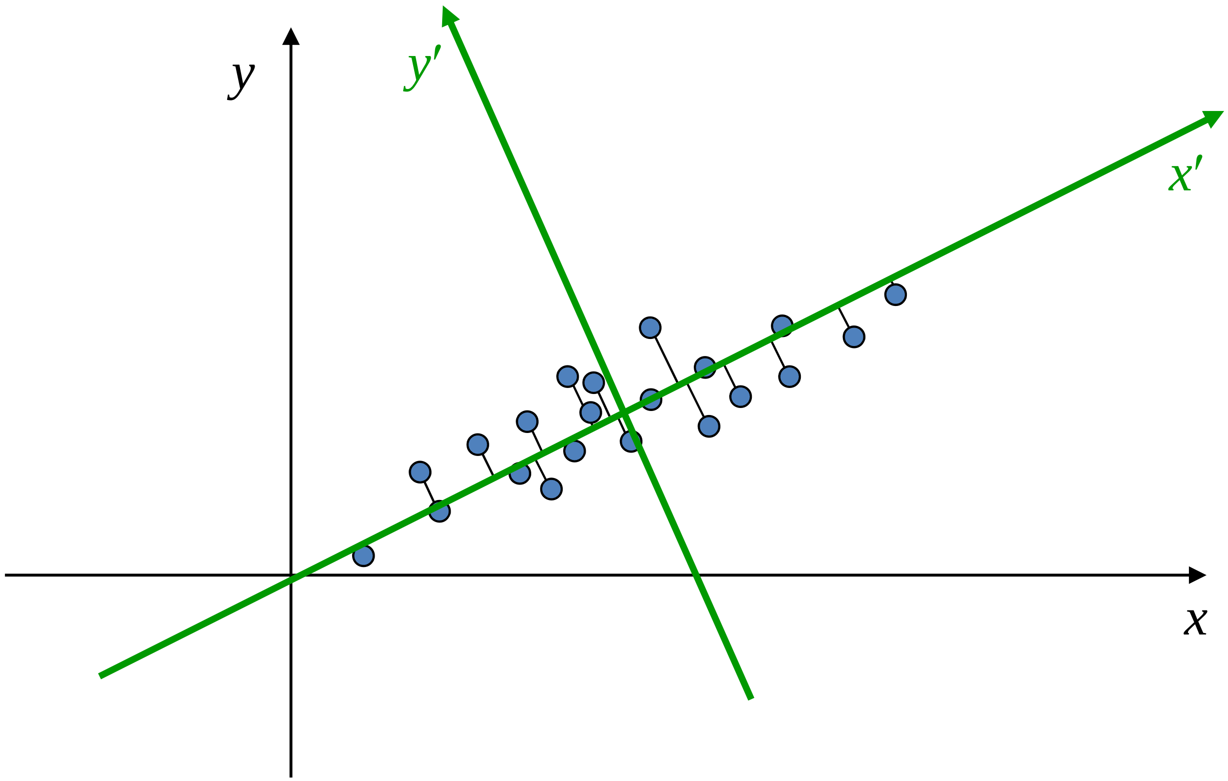
- We just saw how to fit a parametric line $y = ax + b$, but this does not work for vertical lines

Motivation



- How to fit a line such that the true (orthogonal) distances are minimized?

Principal Component Analysis

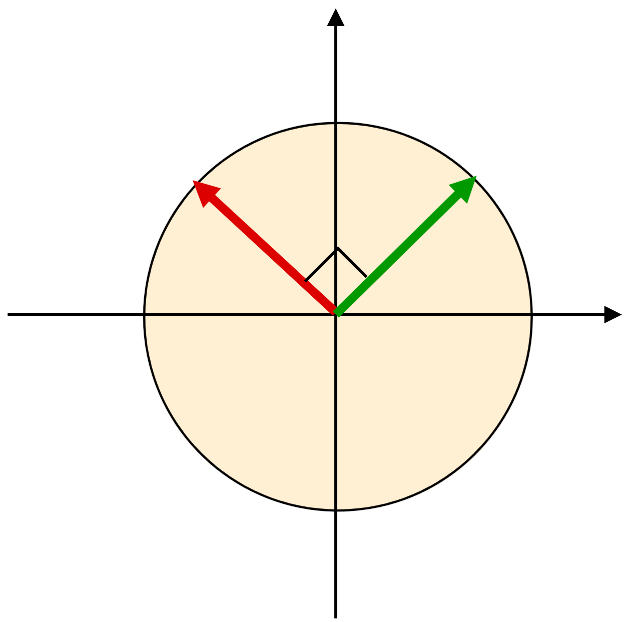


- PCA finds axes that minimize the sum of distances^2

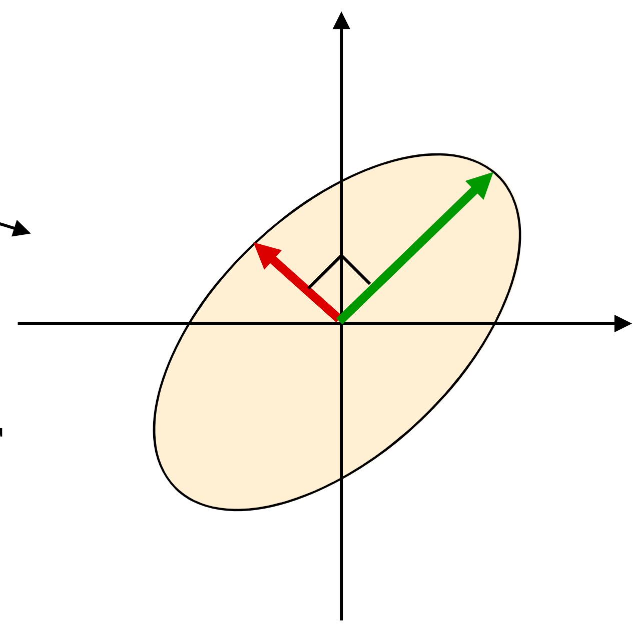
Linear algebra recap

Symmetric matrices

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



$$\xrightarrow{\mathrm{A}}$$



$$\mathrm{A} = \mathrm{UDU}^{\mathrm{T}}$$

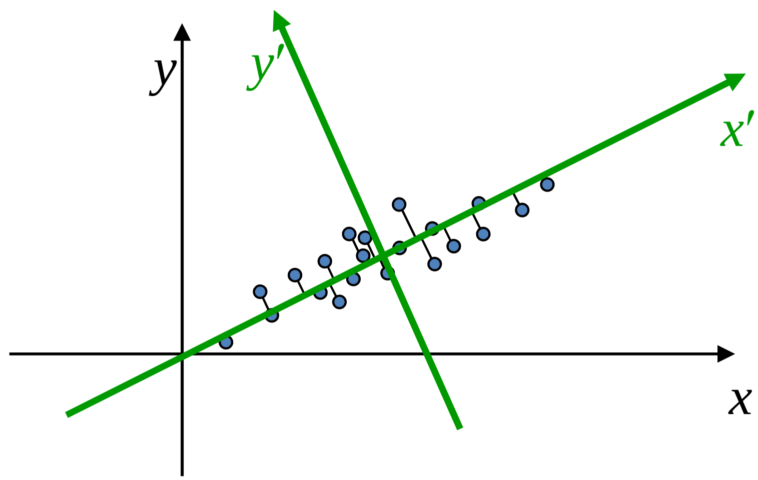
$$\mathrm{D} = \begin{pmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_n \end{pmatrix}$$

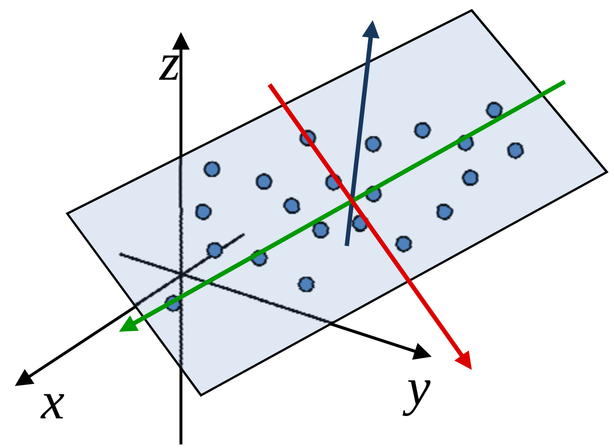
$$\mathrm{A}\mathbf{u}_{\mathrm{i}} = \lambda_i \, \mathbf{u}_{\mathrm{i}}$$

Principal Component Analysis

Basic idea

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



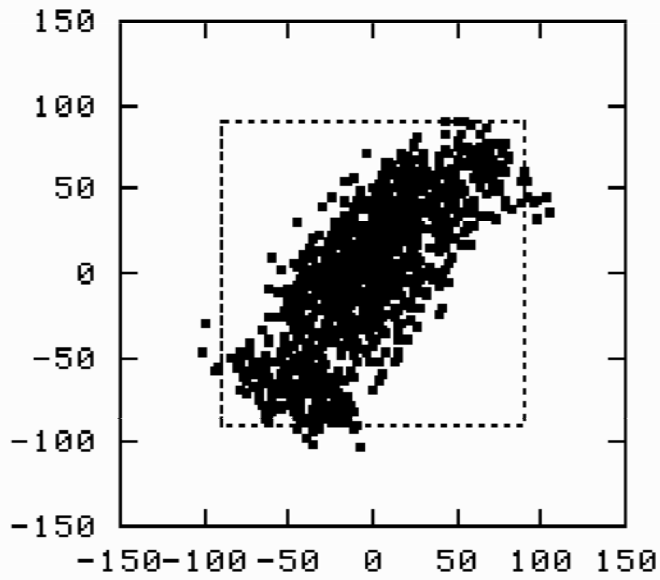


- PCA finds a best approximating line/plane/axes… (in terms of $\Sigma distances^2$)

Principal Component Analysis

Basic idea

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



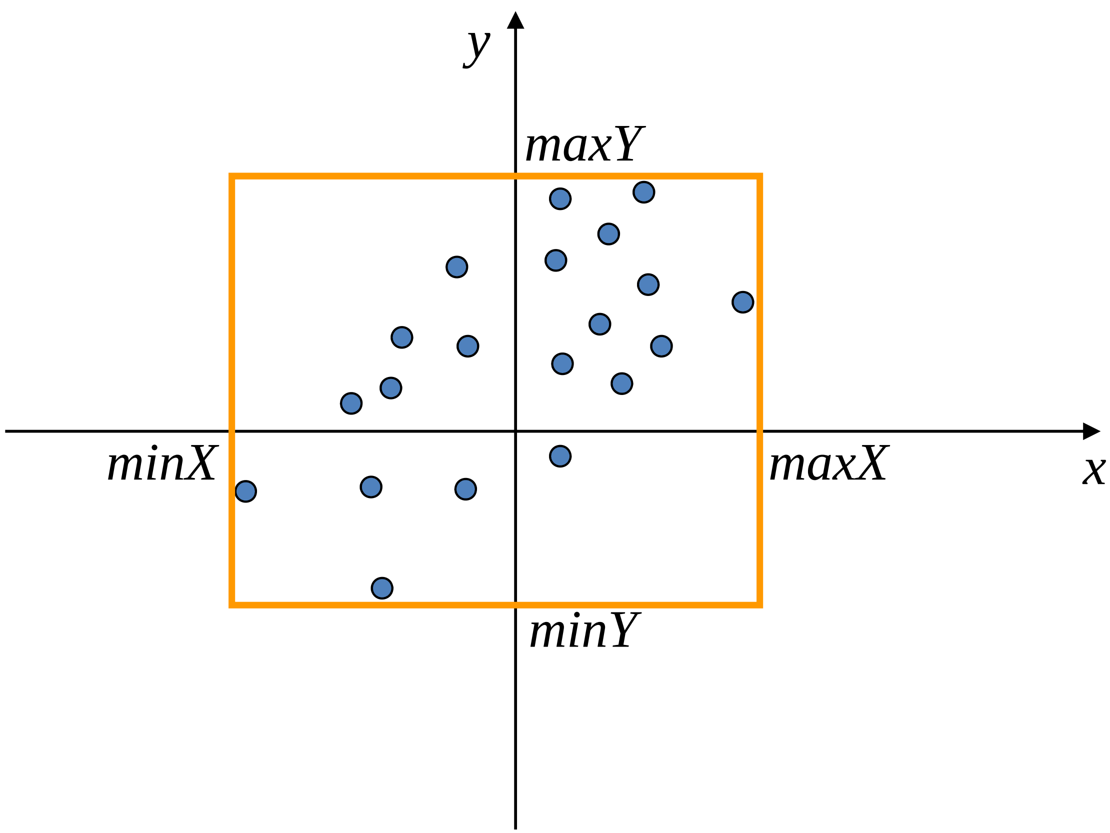
3D point set in standard basis

- PCA finds a best approximating line/plane/axes… (in terms of $\Sigma distances^2$)

Principal Component Analysis

Applications

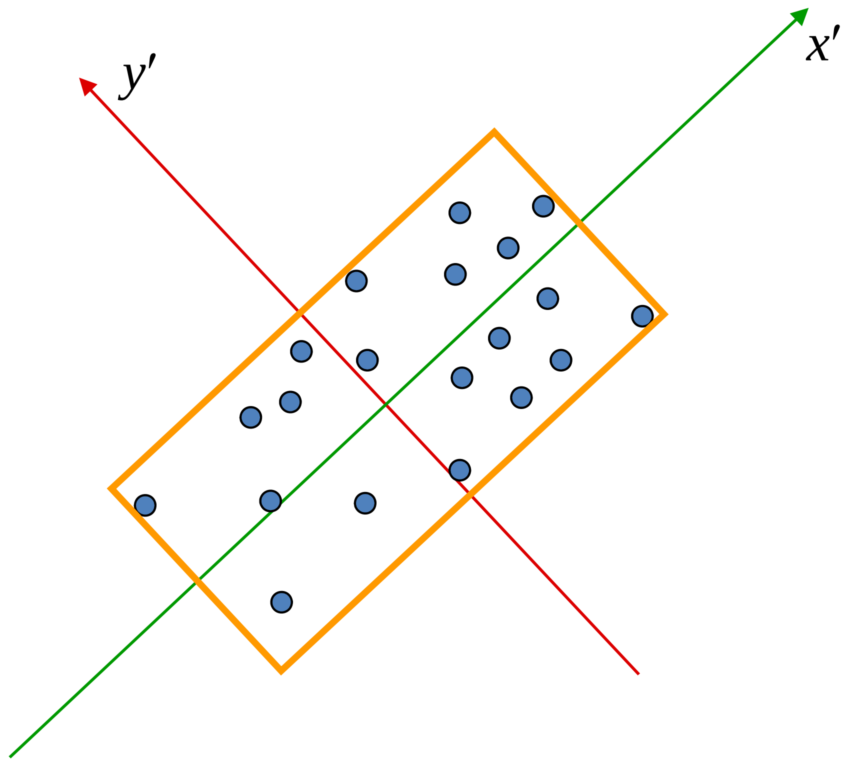
- An axis-aligned bounding box: agrees with the standard axes



Principal Component Analysis

Application: oriented bounding box

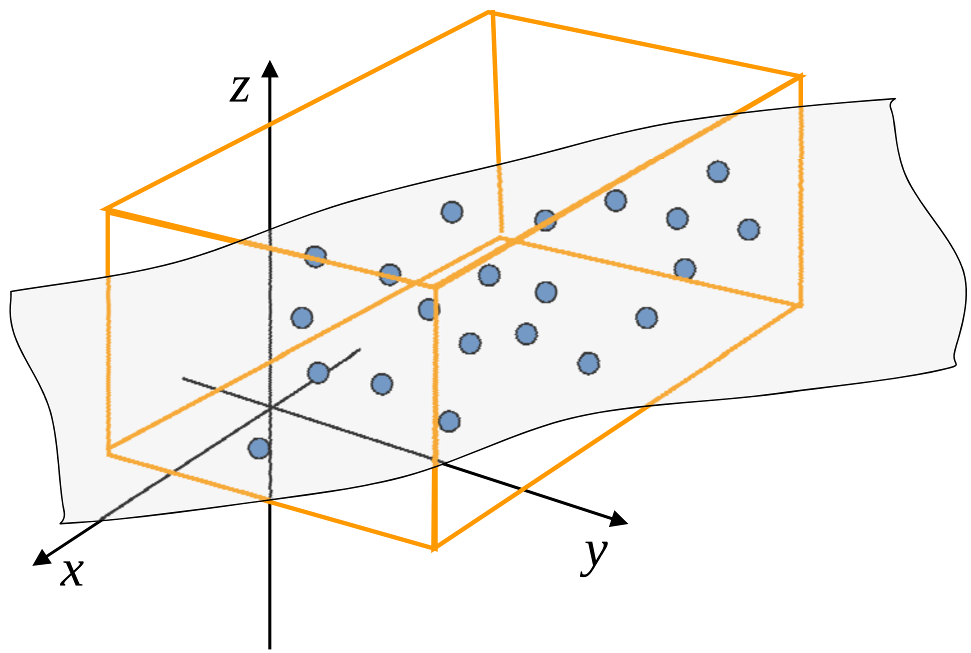
- Tighter fit



Principal Component Analysis

Application: oriented bounding box

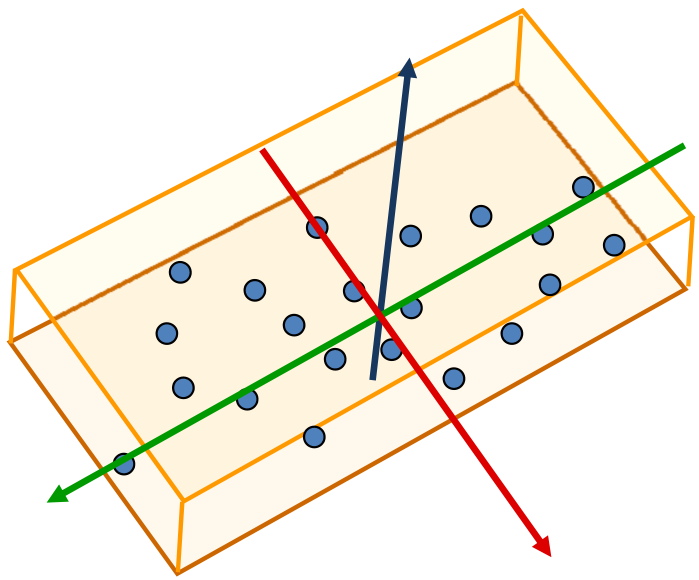
- Axis aligned bounding box



Principal Component Analysis

Application: oriented bounding box

- Oriented bounding box by PCA



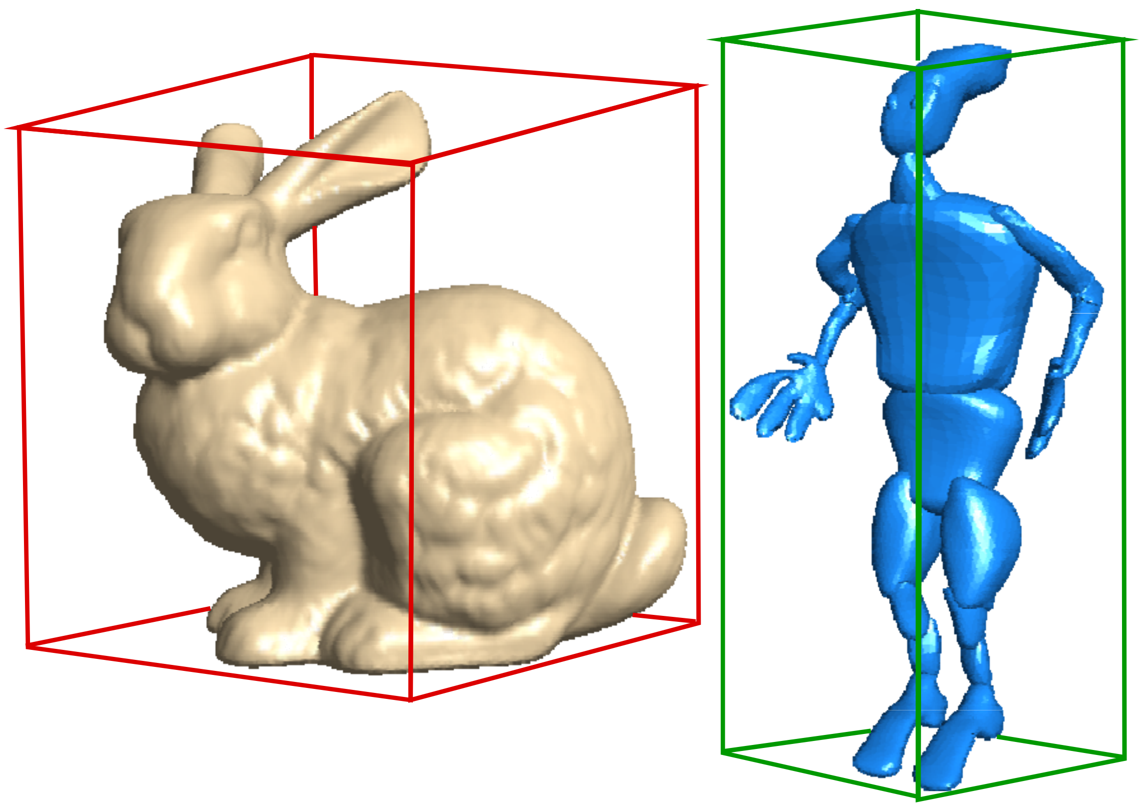
Principal Component Analysis

Application: oriented bounding box

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

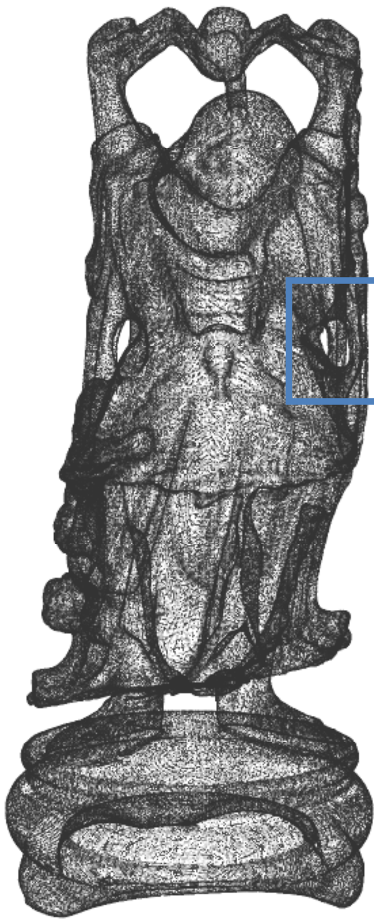
- The models consist of thousands of polygons
- To quickly test that they don't intersect, the bounding boxes are tested
- Sometimes a hierarchy of BB's is used

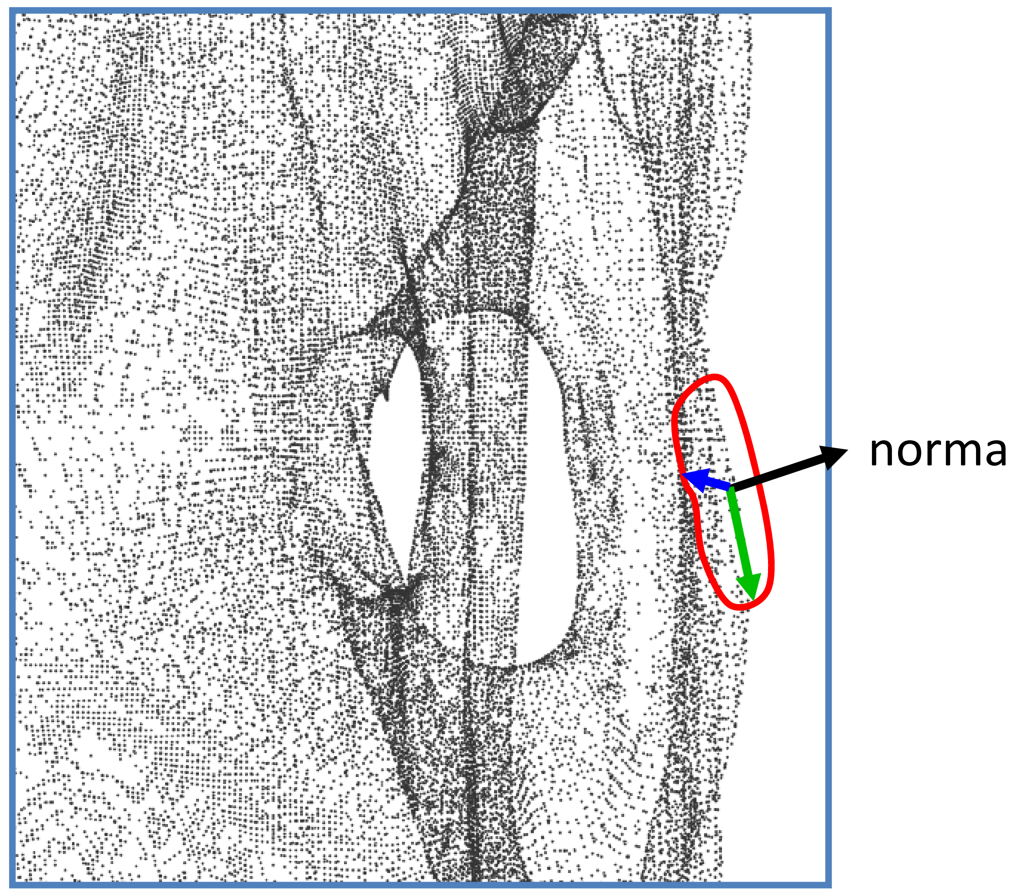
- The tighter the BB – the less "false alarms" we have



Principal Component Analysis

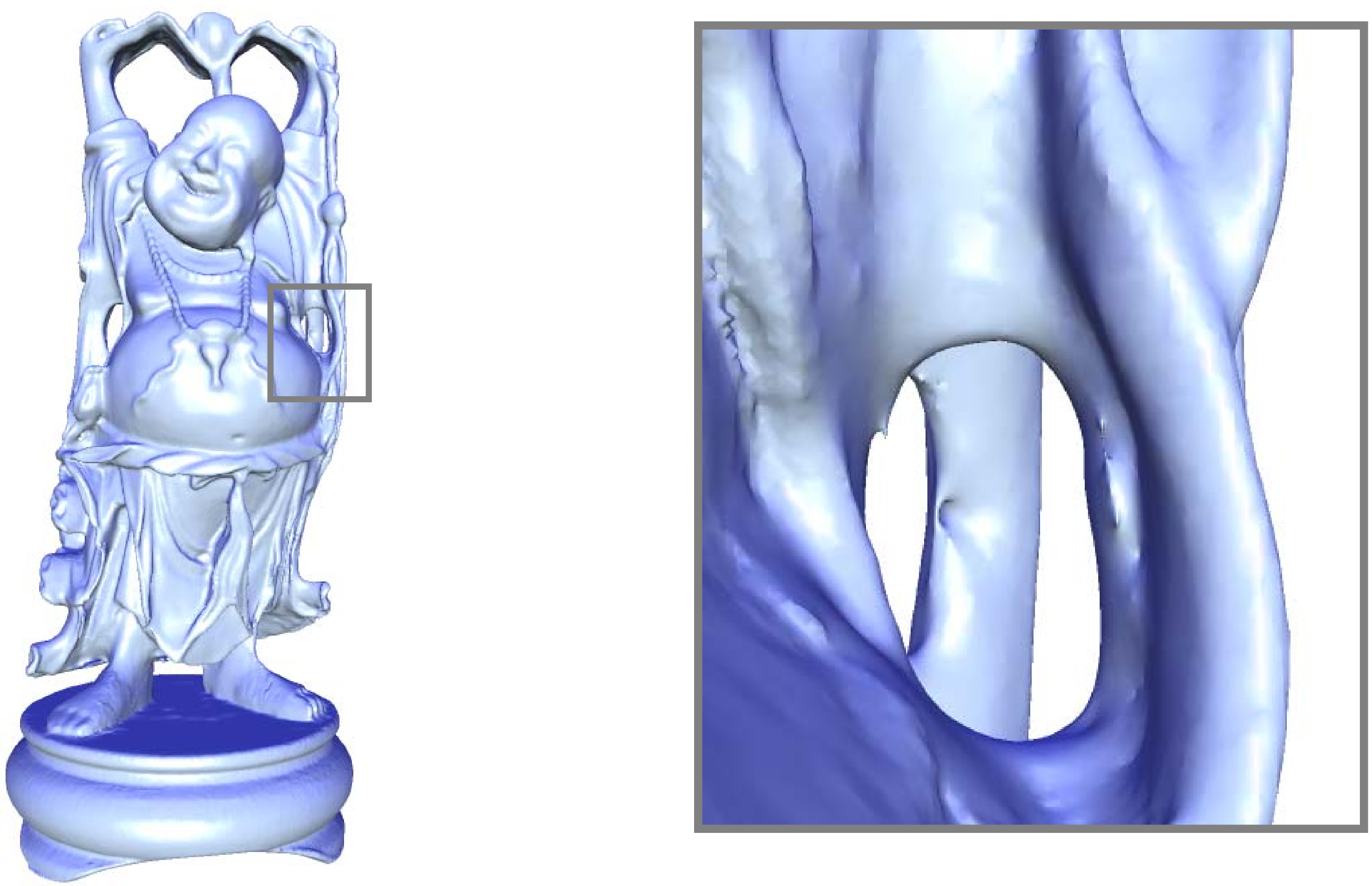
Application: local frame fitting





Principal Component Analysis

Application: estimate normals



Principal Component Analysis

Application: shape alignment

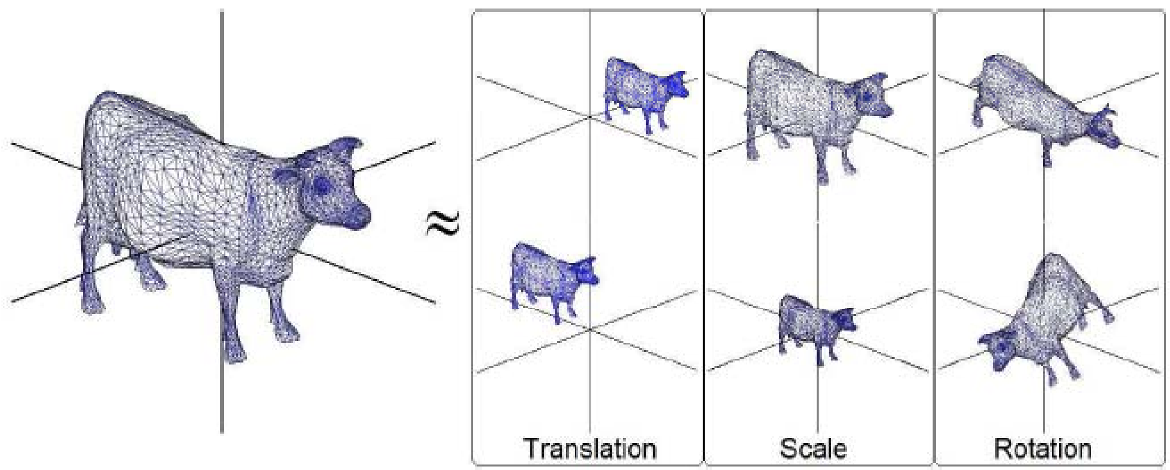
- 3D search engines (see http://shape.cs.princeton.edu/)

Query

Principal Component Analysis

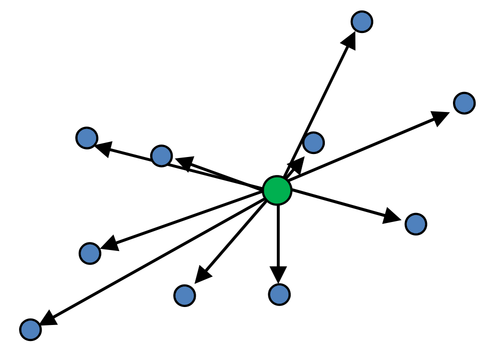
Application: shape alignment

- Can use PCA to find canonical axes and scale for shape comparison



Notations

- Denote our data points by $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n \in R^d$
- Center of mass: $\boxed{\mathbf{m} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{x}_i}$

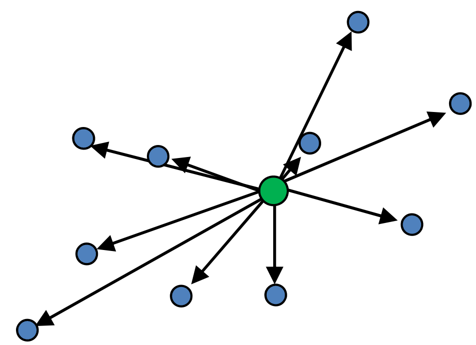


- Vectors from the centroid:

$$\boxed{\mathbf{y}_i = \mathbf{x}_i - \mathbf{m}}$$

The origin of the new axes

- The origin of the new axes will be the center of mass $\mathbf{m}$



- It can be shown that:

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

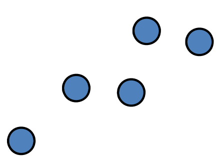
$$\mathbf{m} = \tfrac{1}{n} \sum_{i=1}^{n} \mathbf{x}_i$$

Variance of projected points

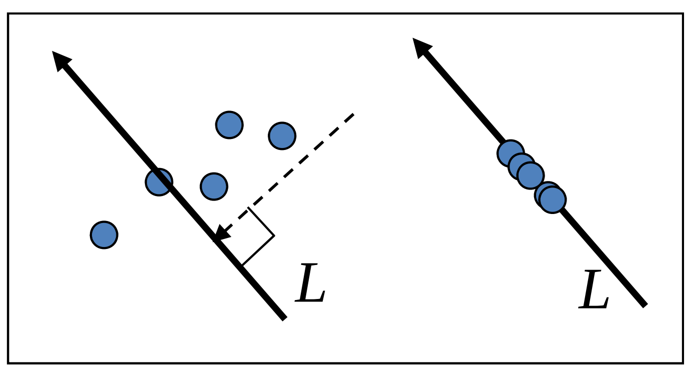
- Let us measure the variance (scatter) of our points in different directions
- Let's look at a line L through the center of mass $\mathbf{m}$, and project our points $\mathbf{x}_i$ onto it. The variance of the projected points $\mathbf{x}'_i$ is:

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

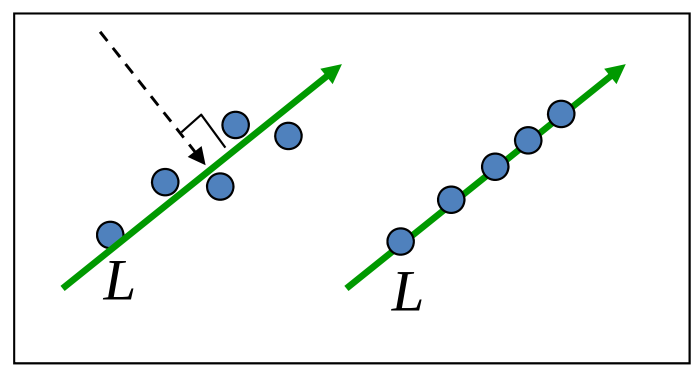
Want to find directions of maximal/minimal variance



Original set



Small variance

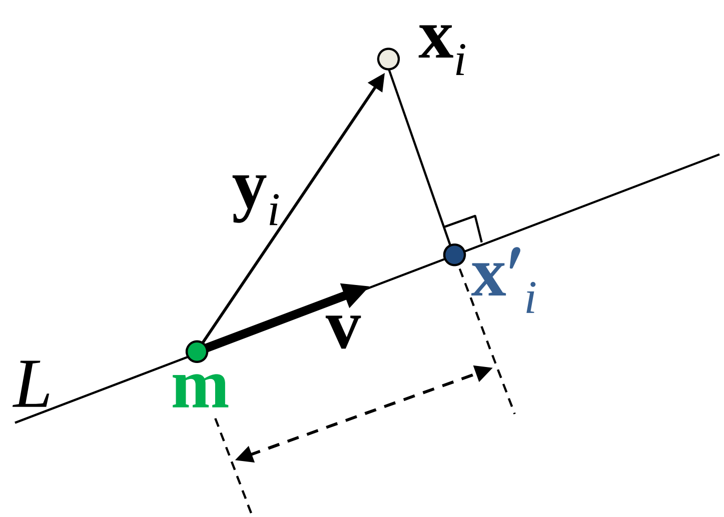


Large variance

Variance of projected points

- Given a direction $\mathbf{v}$, $\|\mathbf{v}\| = 1$
- Line L through $\mathbf{m}$ in the direction of $\mathbf{v}$ is $L(t) = \mathbf{m} + \mathbf{v}t$.

$$\|\mathbf{x}'_i - \mathbf{m}\| = \langle \mathbf{v}, \mathbf{x}_i - \mathbf{m} \rangle / \|\mathbf{v}\| = \langle \mathbf{v}, \mathbf{y}_i \rangle = \mathbf{v}^{\mathrm{T}} \mathbf{y}_i = \mathbf{y}_i^{\mathrm{T}} \mathbf{v}$$



Variance of projected points

- So,

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

$$= \tfrac{1}{n}\left(\mathrm{Y}^{\mathrm{T}}\mathbf{v}\right)^{\mathrm{T}}\left(\mathrm{Y}^{\mathrm{T}}\mathbf{v}\right) = \tfrac{1}{n}\mathbf{v}^{\mathrm{T}}\mathrm{Y}\mathrm{Y}^{\mathrm{T}}\mathbf{v} = \mathbf{v}^{\mathrm{T}}S\mathbf{v}.$$

$$\boxed{\mathrm{S} = (1/n)\ \mathrm{Y}\ \mathrm{Y}^{\mathrm{T}}}$$ Scatter matrix

where Y is a $d \times n$ matrix with $\mathbf{y}_k = \mathbf{x}_k - \mathbf{m}$ as columns.

- The scatter matrix S measures the variance of our points

Directions of maximal variance

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

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

  $f\,(\mathbf{v}) = \mathbf{v}^{\mathrm{T}}\mathrm{S}\mathbf{v}$ (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!

Directions of maximal variance

- Find extrema of $\mathbf{v}^{\mathrm{T}}\mathrm{S}\mathbf{v}$
- side condition $\mathbf{v}^{\mathrm{T}}\mathbf{v}=1$
- Lagrange Multipliers: $\nabla f + \lambda \nabla g = 0$

$$\nabla(\mathbf{v}^{\mathrm{T}}\mathrm{S}\mathbf{v}) + \lambda\nabla(\mathbf{v}^{\mathrm{T}}\mathbf{v} - 1) = 0$$

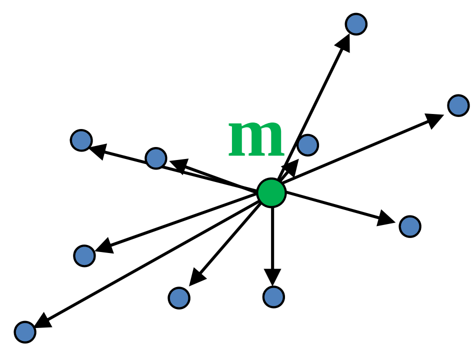
$$\mathrm{S}\mathbf{v} + \lambda\mathbf{v} = 0$$

$$\mathrm{S}\mathbf{v} = -\lambda\mathbf{v}$$

- This is the definition of an eigenvector of S

Summary so far

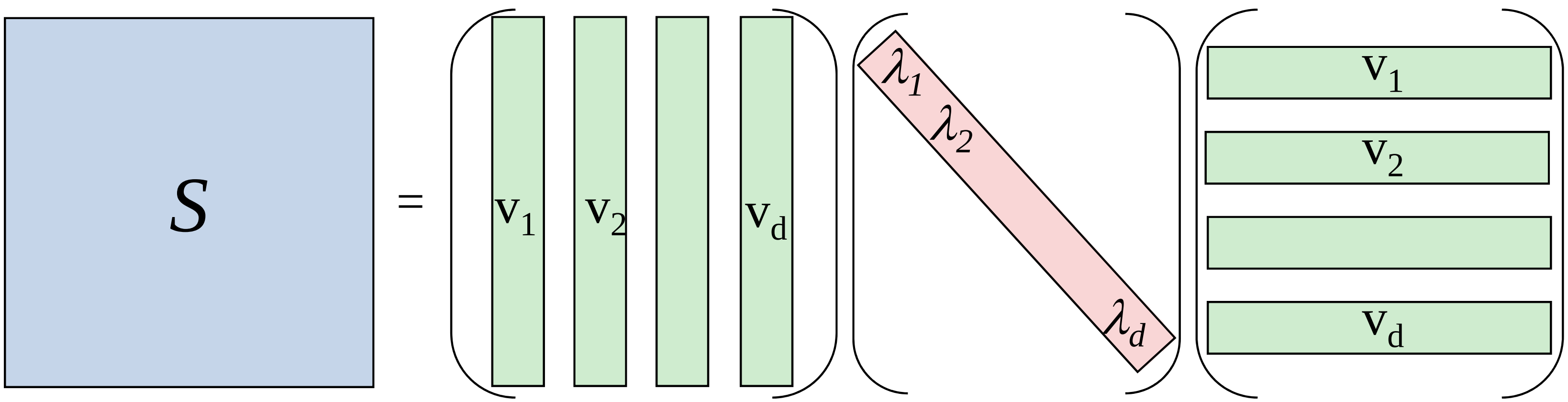
- We take the centered data vectors $\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_n \in R^d$
- Construct the scatter matrix $\mathrm{S} = \mathrm{Y}\,\mathrm{Y}^{\mathrm{T}}$
- S measures the variance of the data points
- Eigenvectors of S are directions of maximal variance.



Scatter matrix eigendecomposition

- S is symmetric

$\Rightarrow$ S has eigendecomposition: $S = VDV^T$



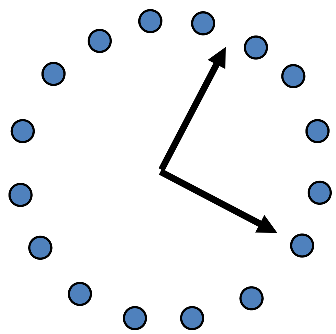
The eigenvectors form orthogonal basis

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. Think why…

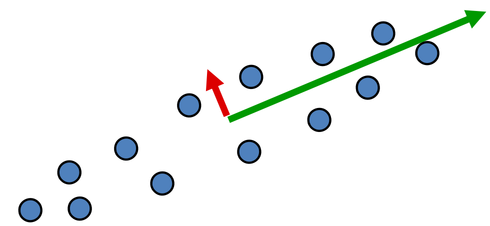
Principal components



- There's no preferable direction
- S looks like this:

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

- Any vector is an eigenvector



- There's a clear preferable direction
- S looks like this:

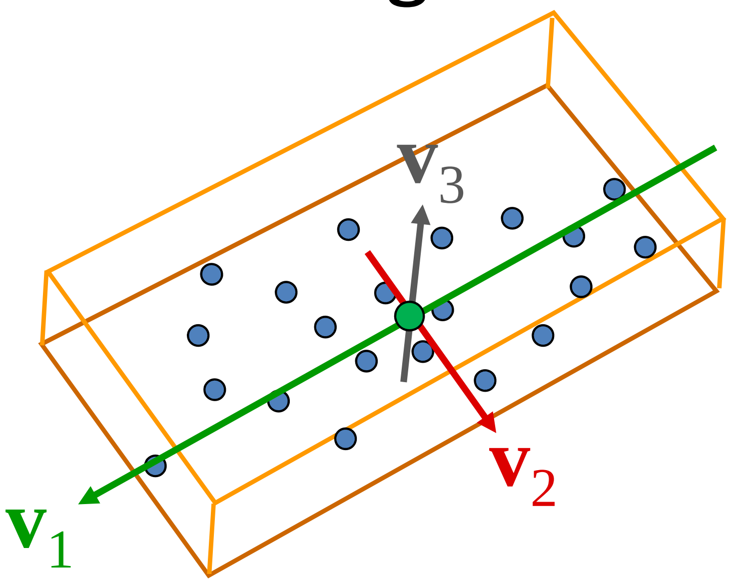
$$\mathrm{V}\begin{pmatrix}\lambda & \\ & \mu\end{pmatrix}\mathrm{V}^{\mathrm{T}}$$

- μ is close to zero, much smaller than λ

How to use what we got

Oriented bounding box

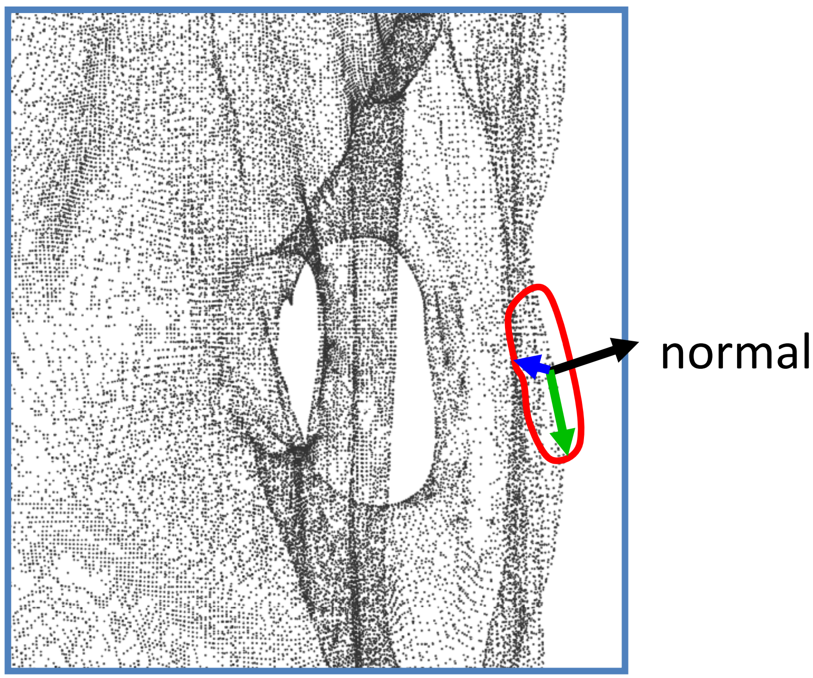
- For finding oriented bounding box or alignment – we simply compute the bounding box with respect to the axes defined by the eigenvectors. The origin is at the centroid $\mathbf{m}$.



How to use what we got

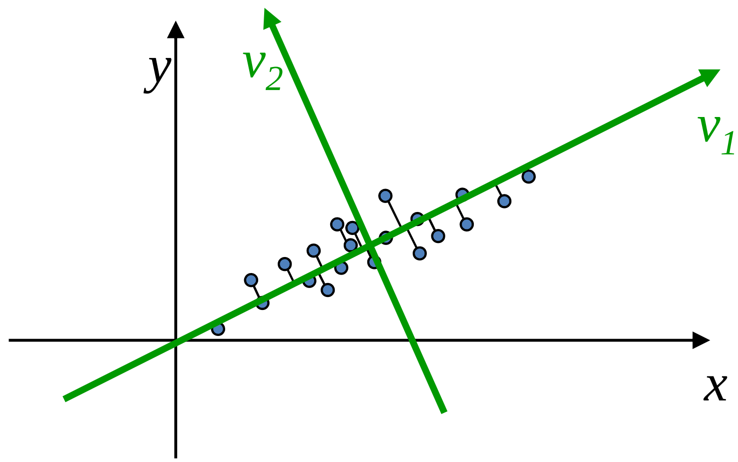
Local frame/normal estimation

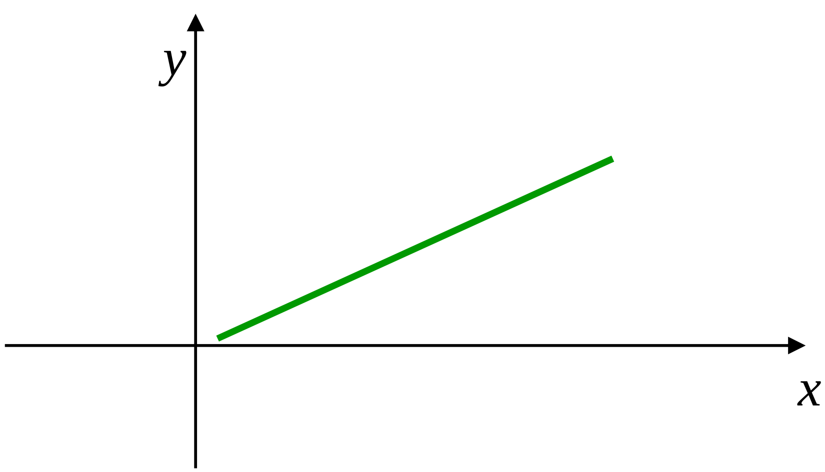
- Sort the eigenvectors by ascending eigenvalues
- The eigenvector with $\lambda \approx 0$ is the normal



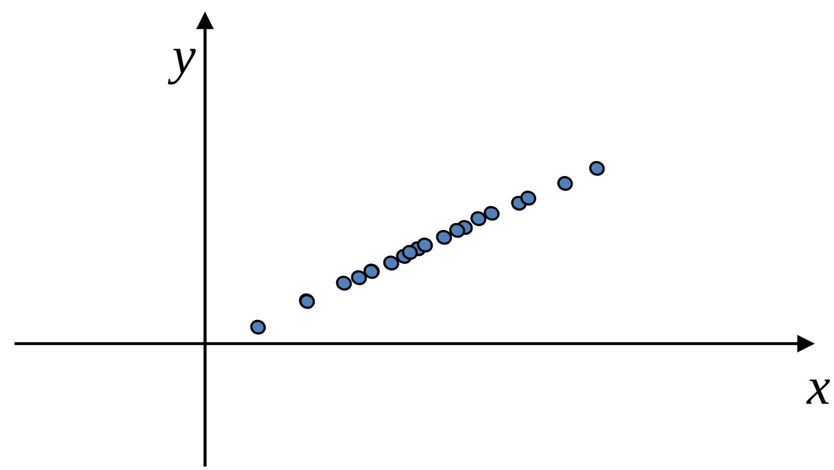
How to use what we got

Dimensionality reduction / approximation





This line segment approximates the original data set



The projected data set approximates the original data set

How to use what we got

Dimensionality reduction / approximation

- Each image is 64x64
- Vector in R^{64x64}
- But in fact all the faces live on a low-dimensional subspace
- Can find meaningful axes with PCA and other methods
  - face pose
  - expression
  - …

How to use what we got

Dimensionality reduction / 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…)

How to use what we got

Dimensionality reduction / 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:

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

Technical remarks:

- $\lambda_i \geq 0,\ i = 1,\ldots,d$ (such matrices are called positive semi-definite). So we can indeed sort by the magnitude of λ_i
- Theorem: $\lambda_i \geq 0 \quad \Leftrightarrow \quad \langle \mathrm{S}\mathbf{v}, \mathbf{v}\rangle \geq 0 \quad \forall \mathbf{v}$

Proof:

$$\begin{aligned}\mathrm{S} &= \mathrm{VDV}^{\mathrm{T}} \Rightarrow \langle \mathrm{S}\mathbf{v}, \mathbf{v}\rangle = \mathbf{v}^{\mathrm{T}}\mathrm{S}\mathbf{v} = \mathbf{v}^{\mathrm{T}}\,\mathrm{VDV}^{\mathrm{T}}\mathbf{v} = \\ &= (\mathrm{V}^{\mathrm{T}}\mathbf{v})^{\mathrm{T}}\,\mathrm{D}\,(\mathrm{V}^{\mathrm{T}}\mathbf{v}) = \mathbf{w}^{\mathrm{T}}\,\mathrm{D}\,\mathbf{w} = \\ &= \lambda_1 w_1^{\,2} + \lambda_2 w_2^{\,2} + \ldots + \lambda_d w_d^{\,2}\end{aligned}$$

Therefore, $\lambda_i \geq 0 \Leftrightarrow \langle \mathrm{S}\mathbf{v}, \mathbf{v}\rangle \geq 0 \quad \forall \mathbf{v}$