

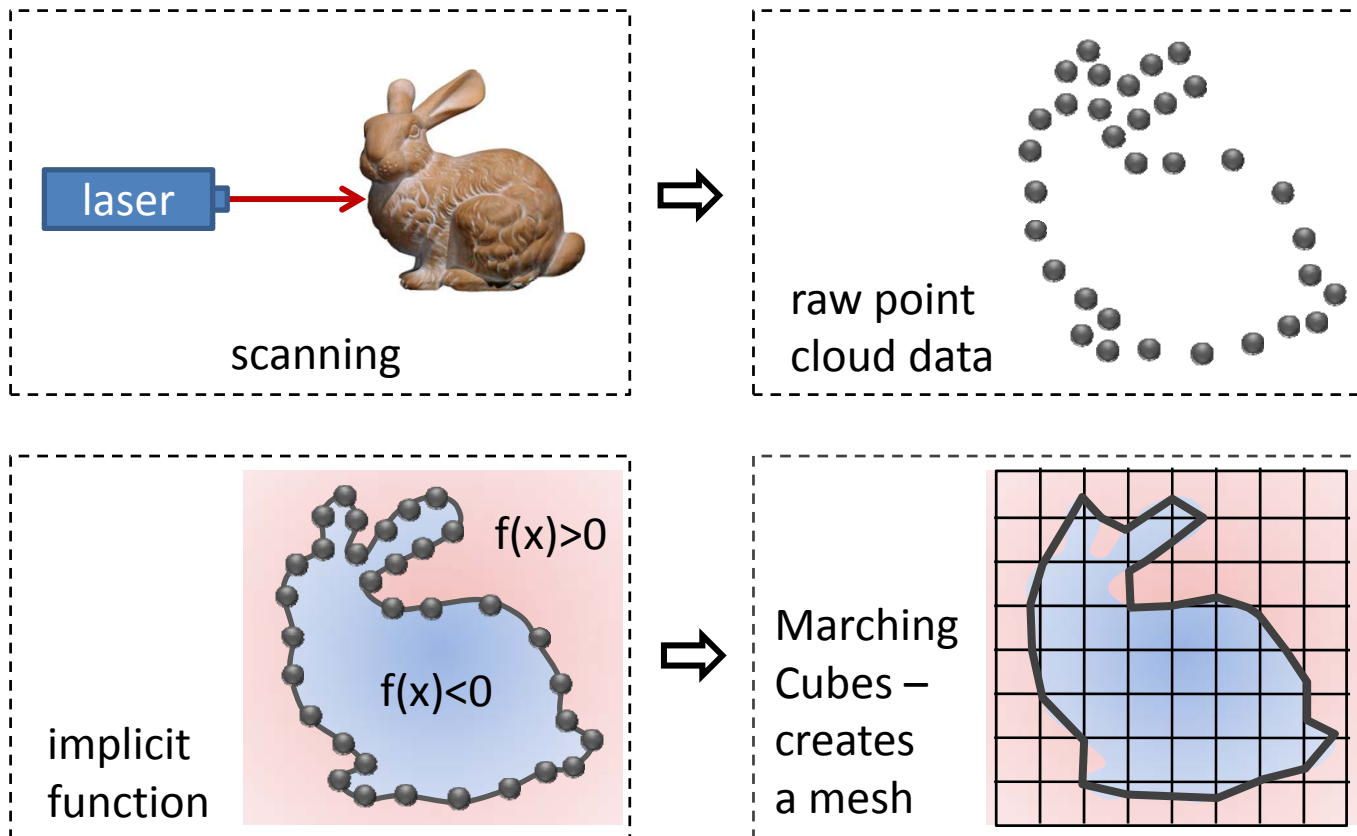
G22.3033-004, Spring 2009

# Interactive Shape Modeling

Linear algebra tools for  
geometric modeling

# Recap

## Surface acquisition and reconstruction



# Recap

## Implicit functions

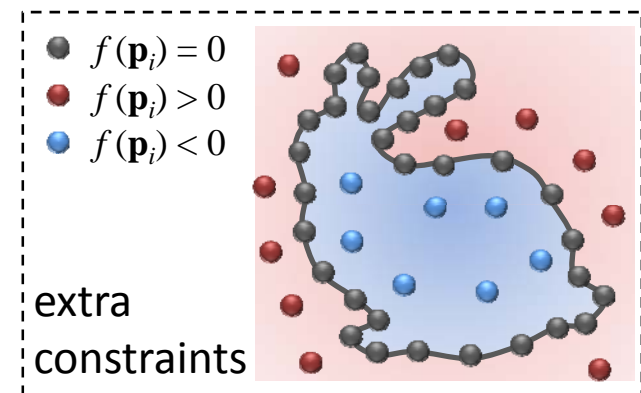
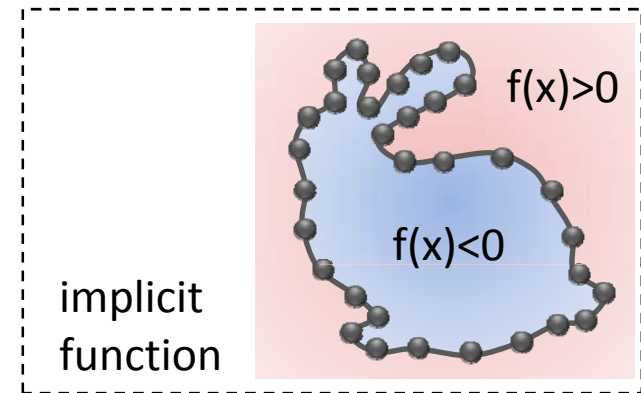
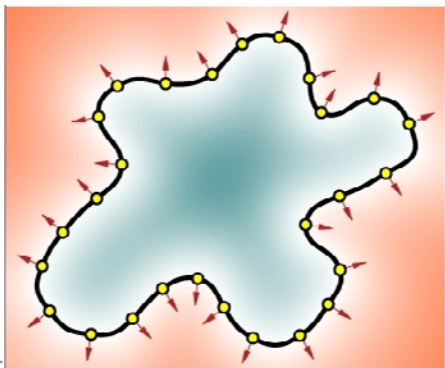
- Implicit function?

$$f(\mathbf{p}_i) = 0$$

- Need extra constraints to avoid trivial solution

$$f(\mathbf{p}_i + \varepsilon \mathbf{n}_i) = +\varepsilon$$

$$f(\mathbf{p}_i - \varepsilon \mathbf{n}_i) = -\varepsilon$$

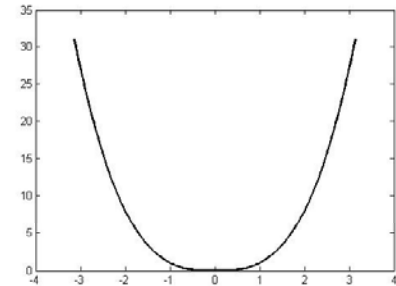


# Recap

## Implicit functions

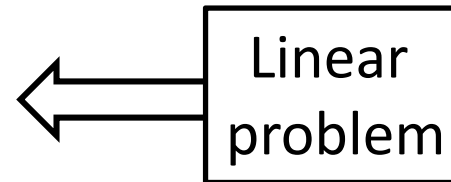
- Radial basis function

$$f_j = \sum_i w_i r(\|\mathbf{p}_i - \mathbf{p}_j\|)$$



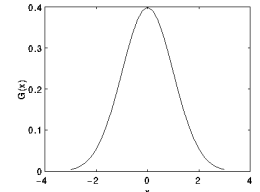
- Constraints:  $f(\mathbf{p}_j) = 0$ ,  $f(\mathbf{p}_i + \alpha \mathbf{n}_i) = \alpha$
- Need to solve for  $w_i$

$$\begin{pmatrix} r(0) & r(\|\mathbf{p}_0 - \mathbf{p}_1\|) & r(\|\mathbf{p}_0 - \mathbf{p}_2\|) & \cdots \\ r(\|\mathbf{p}_1 - \mathbf{p}_0\|) & r(0) & r(\|\mathbf{p}_1 - \mathbf{p}_2\|) & \\ r(\|\mathbf{p}_2 - \mathbf{p}_0\|) & r(\|\mathbf{p}_2 - \mathbf{p}_1\|) & r(0) & \\ \vdots & & & \ddots \end{pmatrix} \begin{pmatrix} w_0 \\ w_1 \\ w_2 \\ \vdots \end{pmatrix} = \begin{pmatrix} f_0 \\ f_1 \\ f_2 \\ \vdots \end{pmatrix}$$



# Recap

## Implicit functions



- Moving least squares

$$f(\mathbf{x}) = f_{\mathbf{x}}(\mathbf{x}); \quad f_{\mathbf{x}}(\mathbf{x}) = \arg \min_{f_{\mathbf{x}} \in \Pi_k^d} \sum_{i=0}^n \|f_{\mathbf{x}}(\mathbf{p}_i) - f_i\|^2 \theta(\|\mathbf{p}_i - \mathbf{x}\|)$$

- Need to solve **locally** for  $f_{\mathbf{x}}$ , where  $f_{\mathbf{x}}$  is a polynomial (solve for the coefficients  $c_k$ )

$$\begin{aligned} f_{\mathbf{x}}(\mathbf{x}) &= c_0 + c_1 x + c_2 y + c_3 x^2 + c_4 xy + c_5 y^2 \dots \\ &= \mathbf{c}^T \mathbf{b}(\mathbf{x}). \end{aligned}$$

$$\min_{\mathbf{c}} \sum_{i=0}^n \|\mathbf{c}^T \mathbf{b}(\mathbf{p}_i) - f_i\|^2 w_i(\mathbf{x})$$

Weighted linear  
least squares  
problem

# RBF vs. MLS

$$f(\mathbf{x}) = \sum_{i=1}^n w_i r(\|\mathbf{x} - \mathbf{p}_i\|)$$

- Need to solve for the weights  $w_i$
- Closed formulation
- Requires solving a linear system of size  $n \times n$  ( $n$  is the number of points!)

$$f(\mathbf{x}) = f_{\mathbf{x}}(\mathbf{x});$$

$$f_{\mathbf{x}}(\mathbf{x}) = \arg \min_{f_{\mathbf{x}} \in \Pi_k^d} \sum_{i=1}^n \|f_{\mathbf{x}}(\mathbf{p}_i) - f_i\|^2 \theta(\|\mathbf{x} - \mathbf{p}_i\|)$$

- Solve for the local polynomial in each  $\mathbf{x}$
- No global closed formula – each point has its own function fit
- Requires solving a linear system of size  $k \times k$  ( $k$  is the order of the polynomial) for each evaluation

# Algebraic tools

Linear least squares

But first reminder: vectors/points,  
inner product, projection

# Points and Vectors

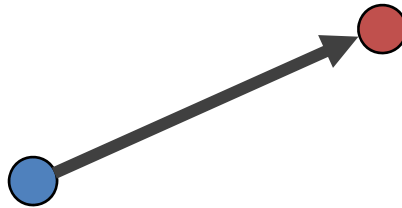
Basic definitions

- Points specify *location* in space (or in the plane).
- Vectors have *magnitude* and *direction* (like velocity).

Points  $\neq$  Vectors

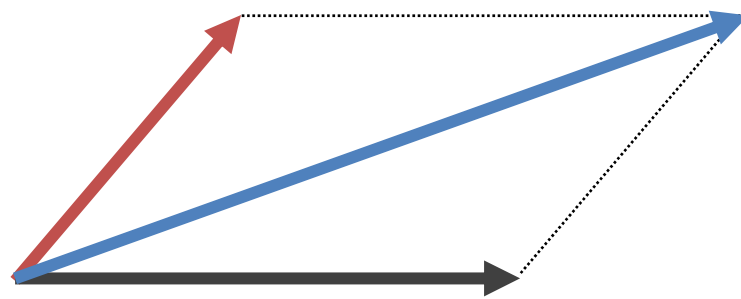


Point + vector = point

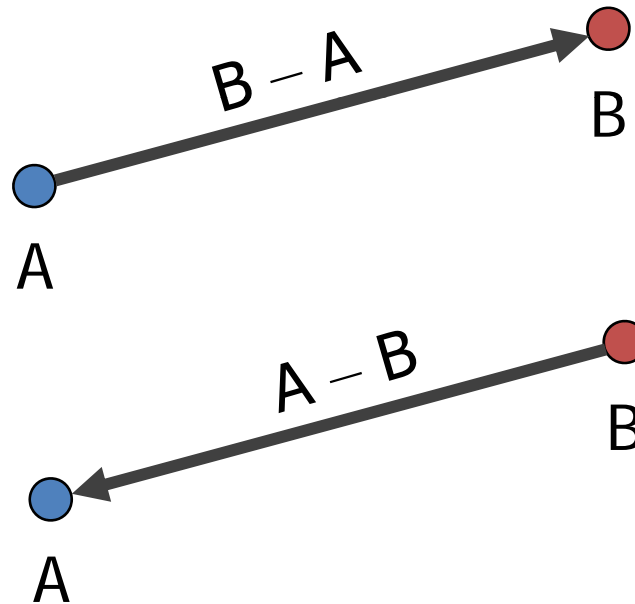


vector + vector = vector

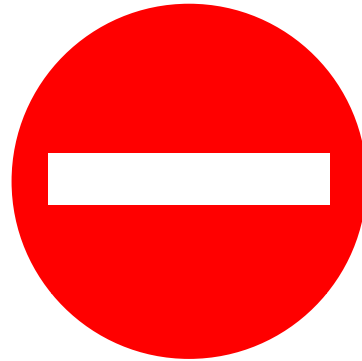
- Parallelogram rule



point – point = vector

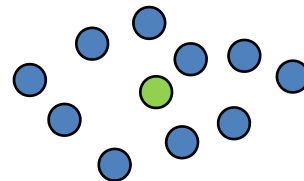


# point + point: not defined!!



- Unless we are computing a weighted average of points (weighted centroid).
  - If the weights sum up to one, the average is meaningful.

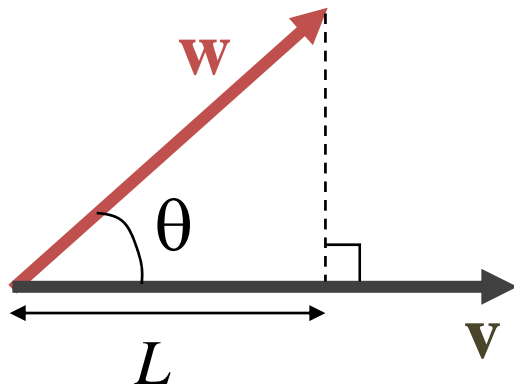
$$\mathbf{c} = \sum_{i=1}^n w_i \mathbf{p}_i$$



# Dot product

- Defined for vectors:

$$\langle \mathbf{v}, \mathbf{w} \rangle = \|\mathbf{v}\| \cdot \|\mathbf{w}\| \cdot \cos \theta$$



$$\cos \theta = L / \|\mathbf{w}\|$$

$$L = \|\mathbf{w}\| \cos \theta = \langle \mathbf{v}, \mathbf{w} \rangle / \|\mathbf{v}\|$$

Projection of  $\mathbf{w}$  onto  $\mathbf{v}$

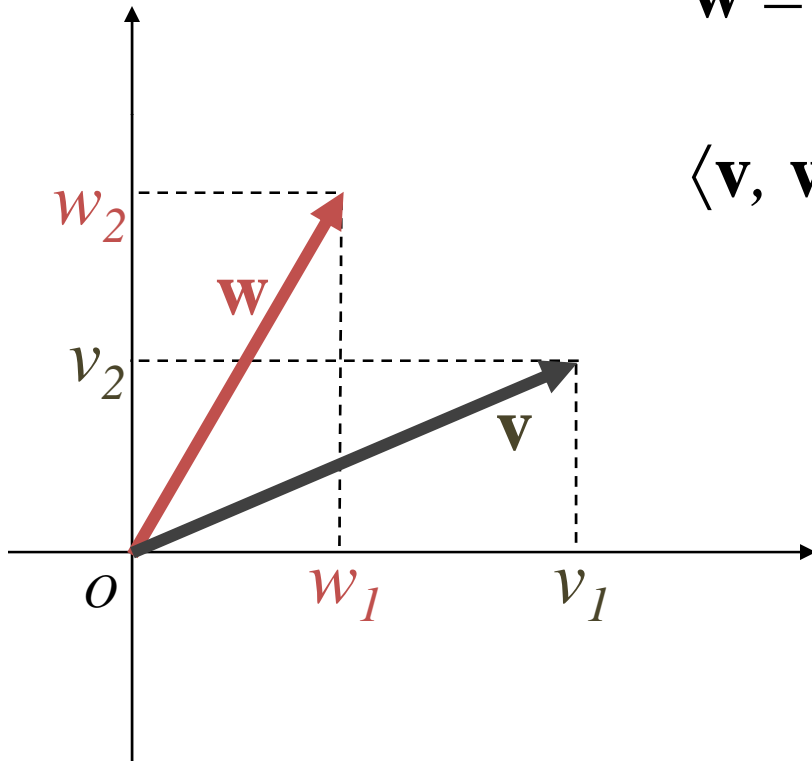
# Dot product

in coordinates

$$\mathbf{v} = (v_1, v_2, \dots, v_d)^T$$

$$\mathbf{w} = (w_1, w_2, \dots, w_d)^T$$

$$\begin{aligned}\langle \mathbf{v}, \mathbf{w} \rangle &= \mathbf{v}^T \mathbf{w} = \mathbf{w}^T \mathbf{v} = \\ &= v_1 w_1 + v_2 w_2 + \dots + v_d w_d\end{aligned}$$



# Dot product

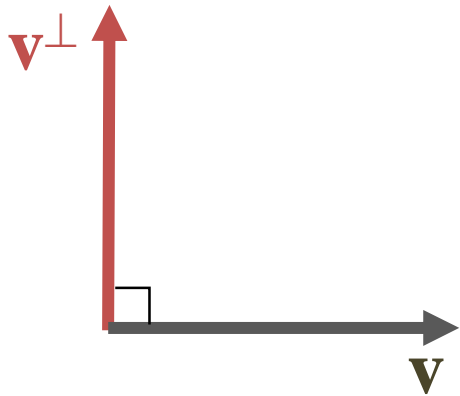
names, notations

- Dot product is also called inner product
- Notations:  $\langle \mathbf{v}, \mathbf{w} \rangle$  or  $\mathbf{v} \cdot \mathbf{w}$  or  $\mathbf{v}^T \mathbf{w}$  ( $= \mathbf{w}^T \mathbf{v}$ )

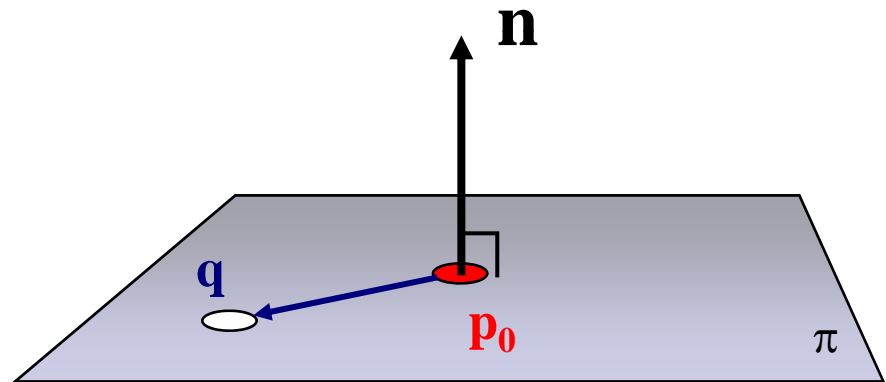
# Dot product

Perpendicular (orthogonal) vectors

$$\langle \mathbf{v}, \mathbf{w} \rangle = \mathbf{v}^T \mathbf{w} = 0$$



In 2D only: if  $\mathbf{v} = (x, y)$   
then  $\mathbf{v}^\perp = \pm(-y, x)$



General hyper-plane:  
all points  $\mathbf{q}$  such that  
 $\langle \mathbf{q} - \mathbf{p}_0, \mathbf{n} \rangle = 0$

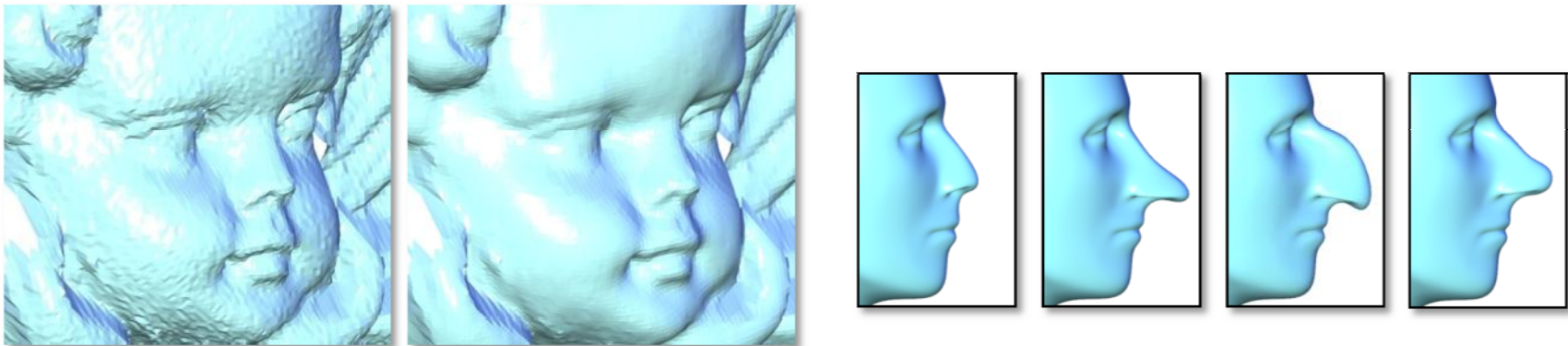


# Least squares fitting

## Motivation

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

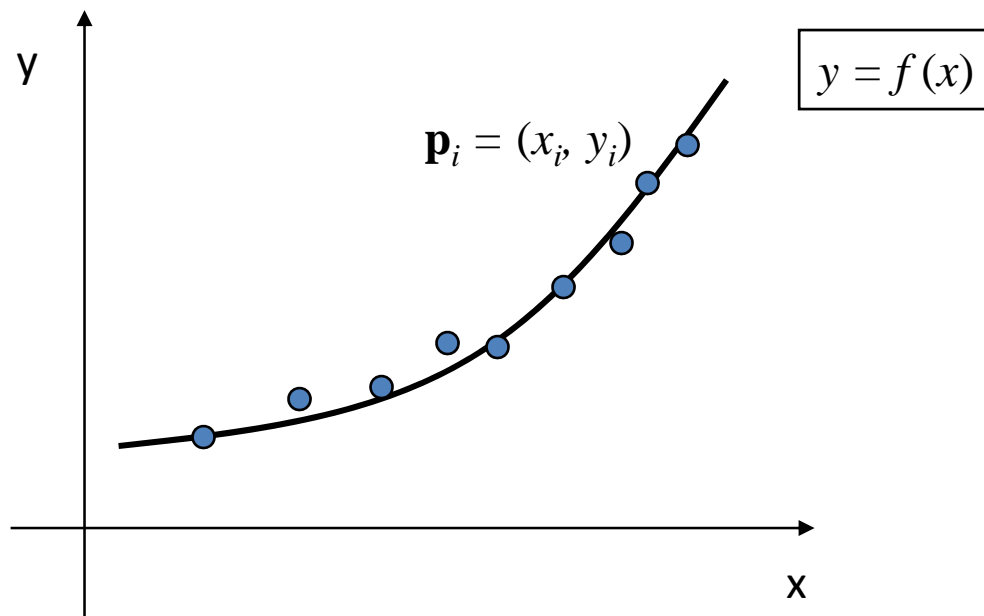
$$\mathbf{c}_{opt} = \underset{\mathbf{c}}{\operatorname{argmin}} \|\mathbf{A}\mathbf{c} - \mathbf{b}\|^2$$



# Least squares fitting

## Motivation

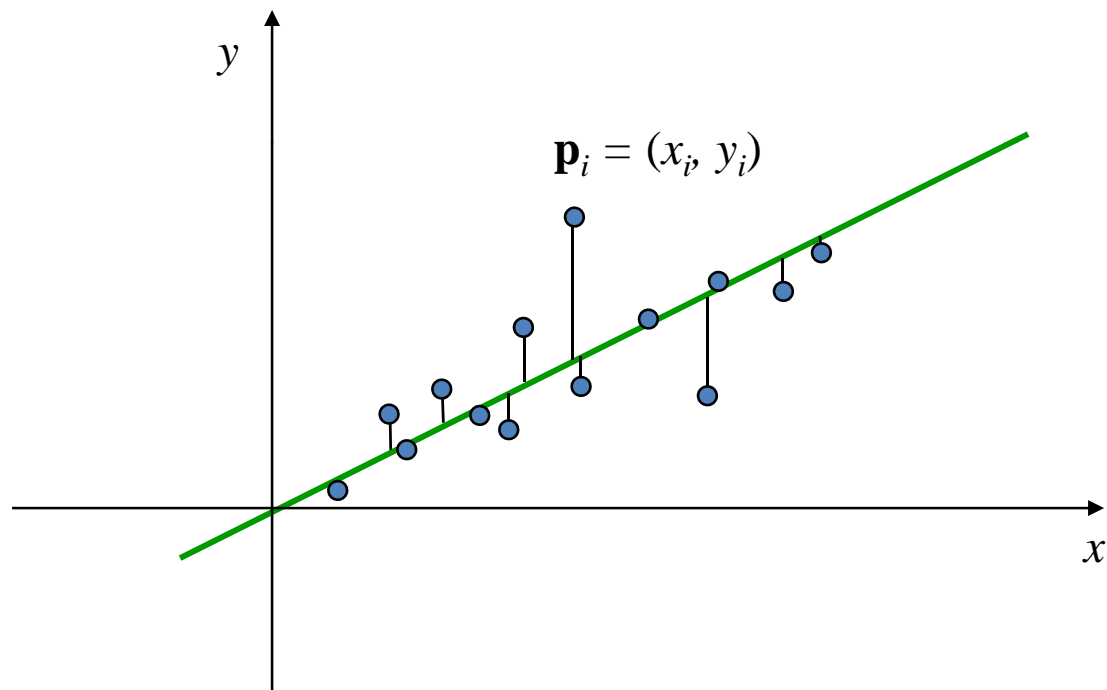
- Given data points, fit a function that is “close” to the points



# Simple example

line fitting – 1<sup>st</sup> order polynomial in 2D

- $y$ -offsets minimization



# Simple example

line fitting – 1<sup>st</sup> 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:

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

- 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  $a$ ,  $b$ :

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

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



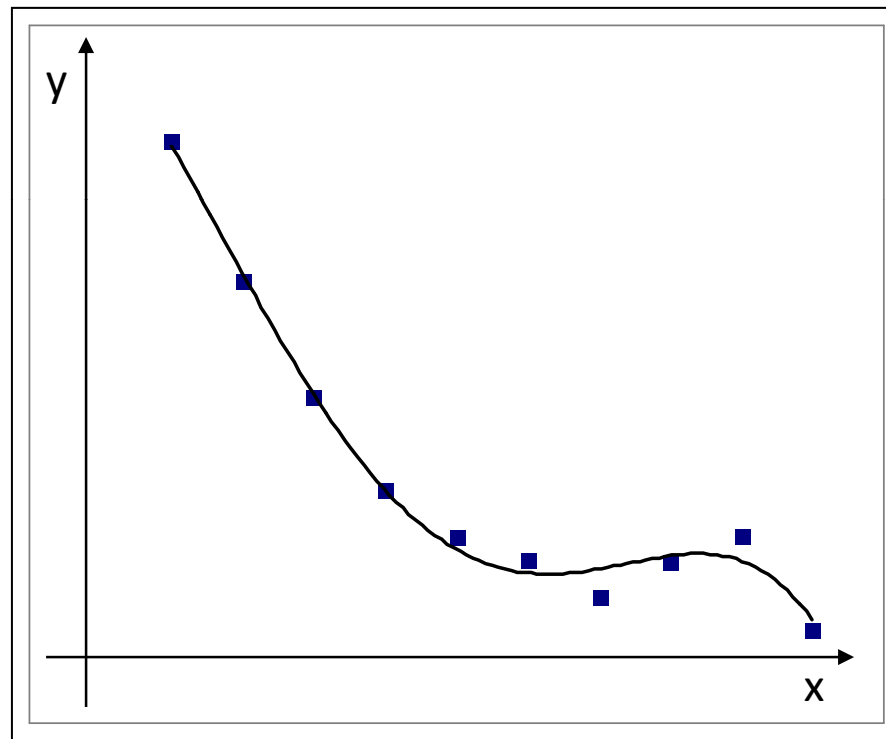
# Simple example

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  $f(x) = a_k x^k + a_{k-1} x^{k-1} + \dots + a_1 x + a_0$
- Minimize:

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

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

# Fitting polynomials

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

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

# General parametric fitting

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

$$\sum_{i=1}^n \|f(\mathbf{p}_i) - f_i\|^2 = \sum_{i=1}^n \left\| \sum_{j=1}^k c_j f_j(\mathbf{p}_i) - f_i \right\|^2$$

# 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 :

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

- 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  $n \times k$ :

$$f(\mathbf{p}_1) = c_1 f_1(\mathbf{p}_1) + c_2 f_2(\mathbf{p}_1) + \dots + c_k f_k(\mathbf{p}_1) = f_1$$

$$f(\mathbf{p}_2) = c_1 f_1(\mathbf{p}_2) + c_2 f_2(\mathbf{p}_2) + \dots + c_k f_k(\mathbf{p}_2) = f_2$$

...

$$f(\mathbf{p}_n) = c_1 f_1(\mathbf{p}_n) + c_2 f_2(\mathbf{p}_n) + \dots + c_k f_k(\mathbf{p}_n) = f_n$$

# Solving linear systems in LS sense

- In matrix form:

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



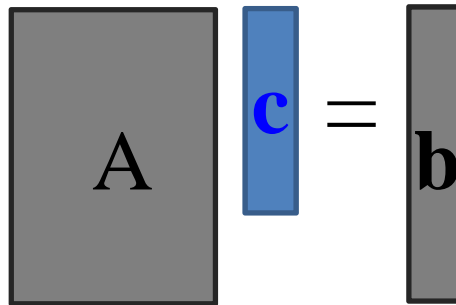
# Solving linear systems in LS sense

- In matrix form:

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

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

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



# Solving linear systems in LS sense

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

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

# Finding the LS solution

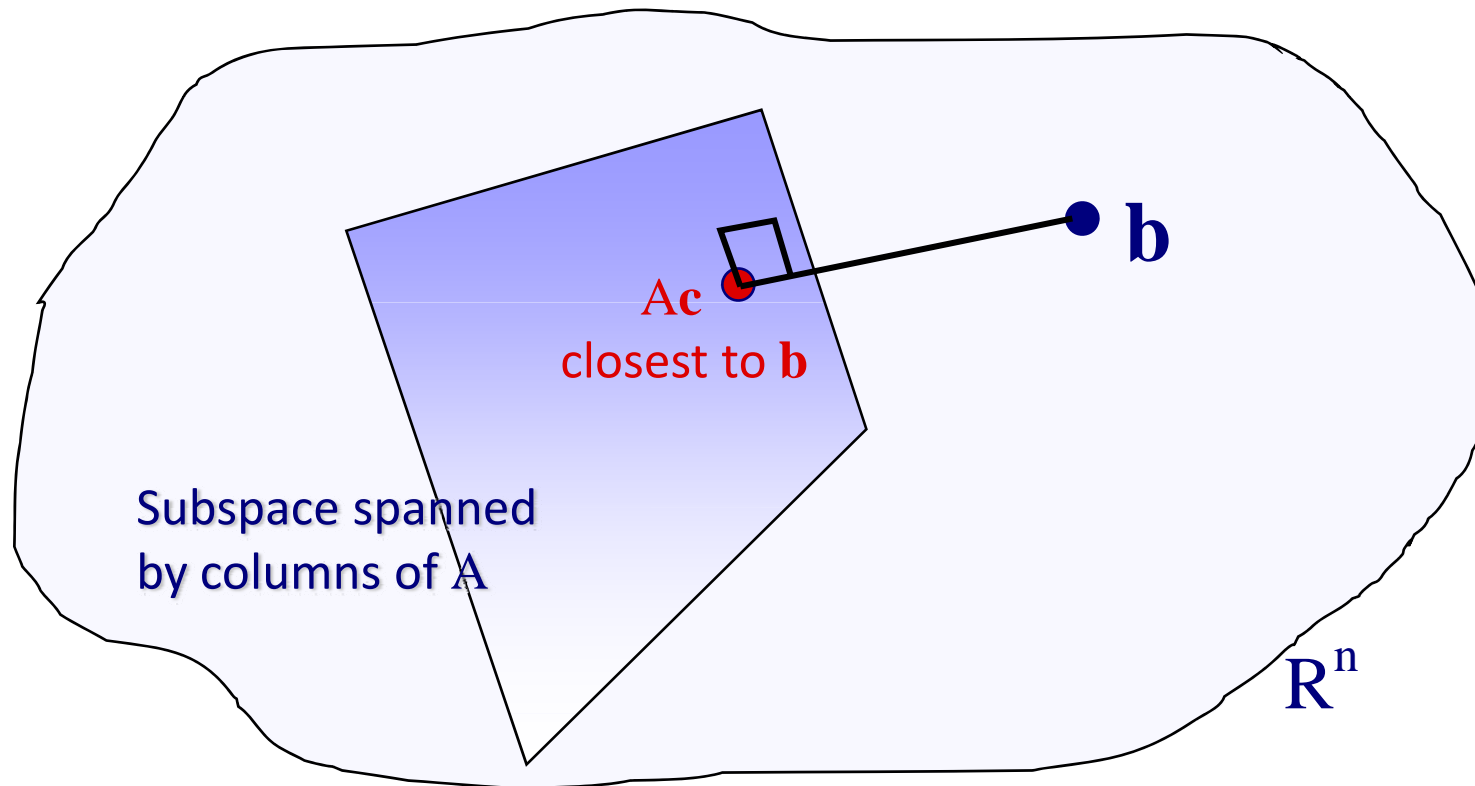
- $\mathbf{c} \in \mathbb{R}^k$
- $A\mathbf{c} \in \mathbb{R}^n$
- As we vary  $\mathbf{c}$ ,  $A\mathbf{c}$  varies over the linear subspace of  $\mathbb{R}^n$  spanned by the columns of  $A$ :

$$A\mathbf{c} = \left( \begin{array}{c|c|c|c} \text{---} & \text{---} & \text{---} & \text{---} \\ | & | & | & | \\ A_1 & A_2 & & A_k \\ | & | & | & | \\ \text{---} & \text{---} & \text{---} & \text{---} \end{array} \right) \begin{array}{c} c_1 \\ c_2 \\ \cdot \\ \cdot \\ c_k \end{array} = c_1 \begin{array}{c} \text{---} \\ | \\ \text{---} \end{array} + c_2 \begin{array}{c} \text{---} \\ | \\ \text{---} \end{array} + \dots + c_k \begin{array}{c} \text{---} \\ | \\ \text{---} \end{array}$$

This is also known as the **column space of  $A$**

# Finding the LS solution

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



# Finding the LS solution

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

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



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

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



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

These are called **the normal equations**

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

# Finding the LS solution

- We have a square symmetric system  $(A^T A)\mathbf{c} = A^T \mathbf{b}$   
(k×k)
- If  $A$  has full rank (the columns of  $A$  are linearly independent) then  $(A^T A)$  is invertible.

$$\min_{\mathbf{c}} \|A\mathbf{c} - \mathbf{b}\|^2$$
$$\Downarrow$$
$$\mathbf{c} = (A^T A)^{-1} A^T \mathbf{b}$$

# 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:

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

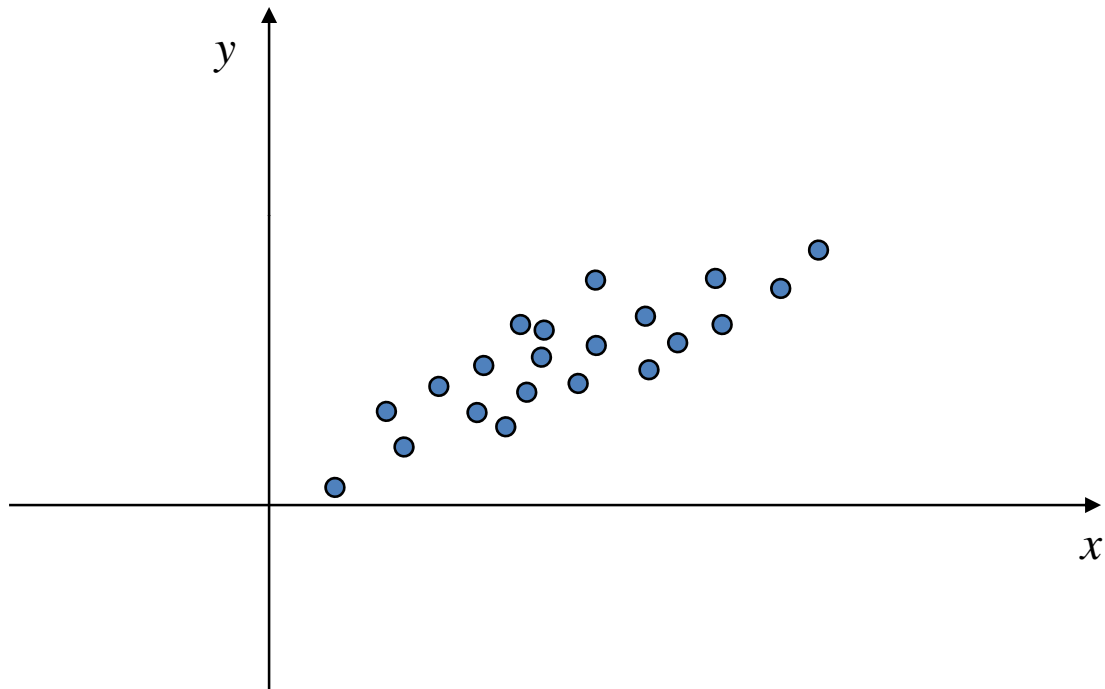
$$(\mathbf{A}^T \mathbf{W}^2 \mathbf{A}) \mathbf{c} = \mathbf{A}^T \mathbf{W}^2 \mathbf{b}$$

# Principal Component Analysis

But first, reminder about  
eigenvectors and eigenvalues

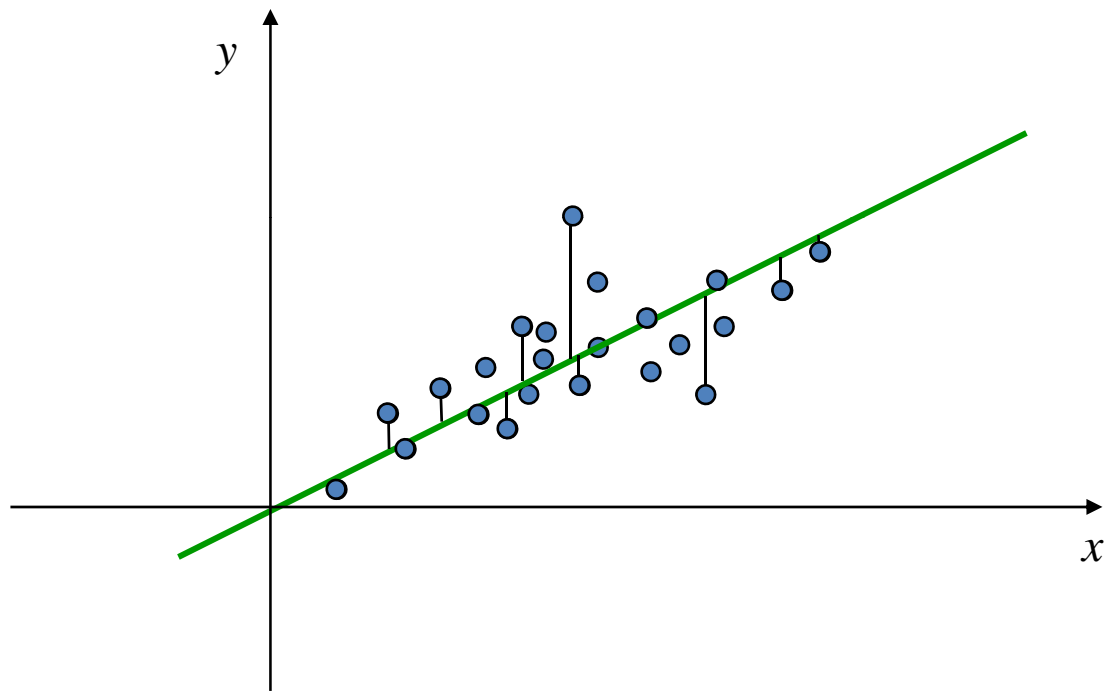


# Motivation



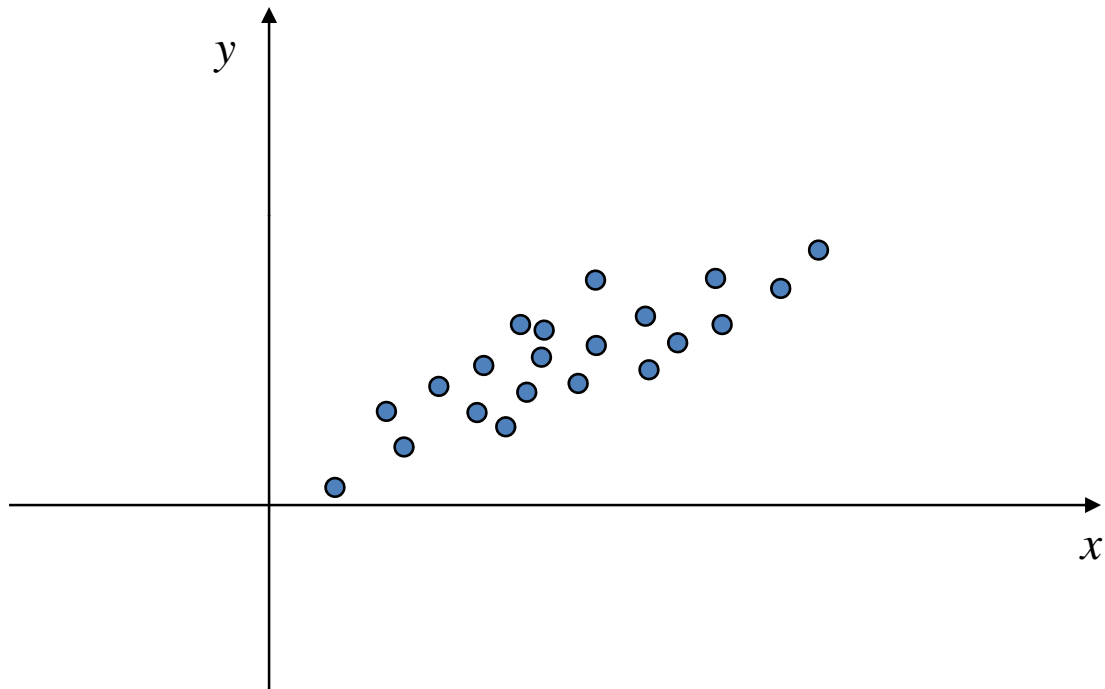
- Given a set of points, find the best line that approximates them

# Motivation



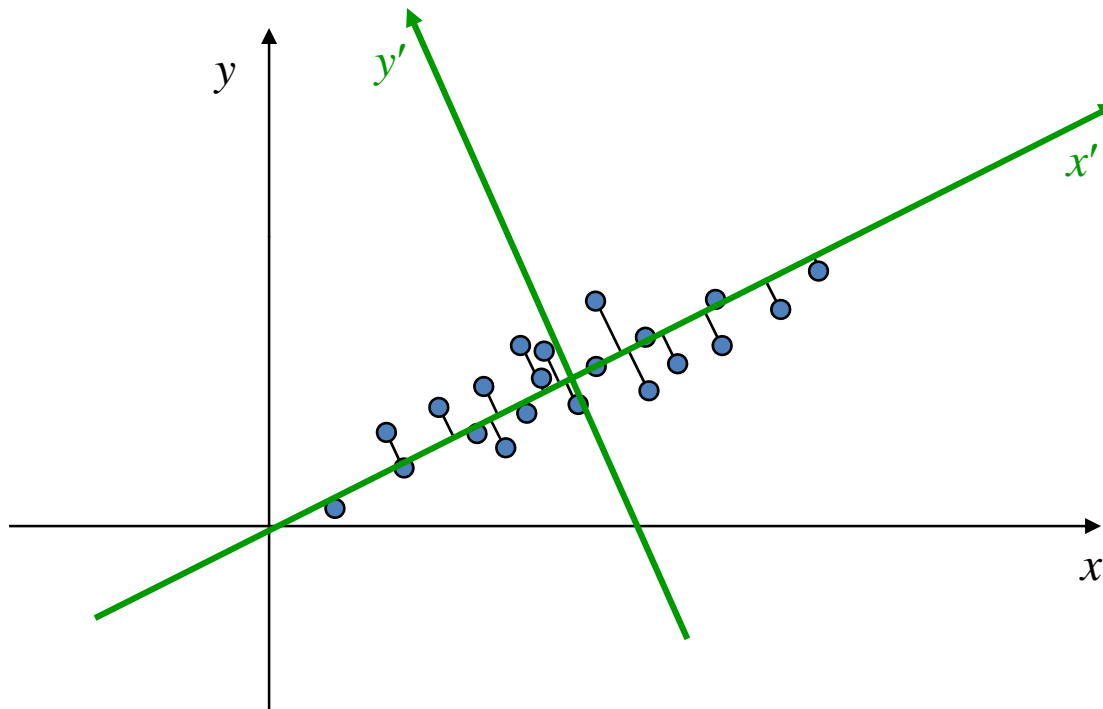
- We just saw how to fit a parametric line  $y = ax + b$ , but this does not work for vertical lines

# 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<sup>2</sup>

# Linear algebra recap

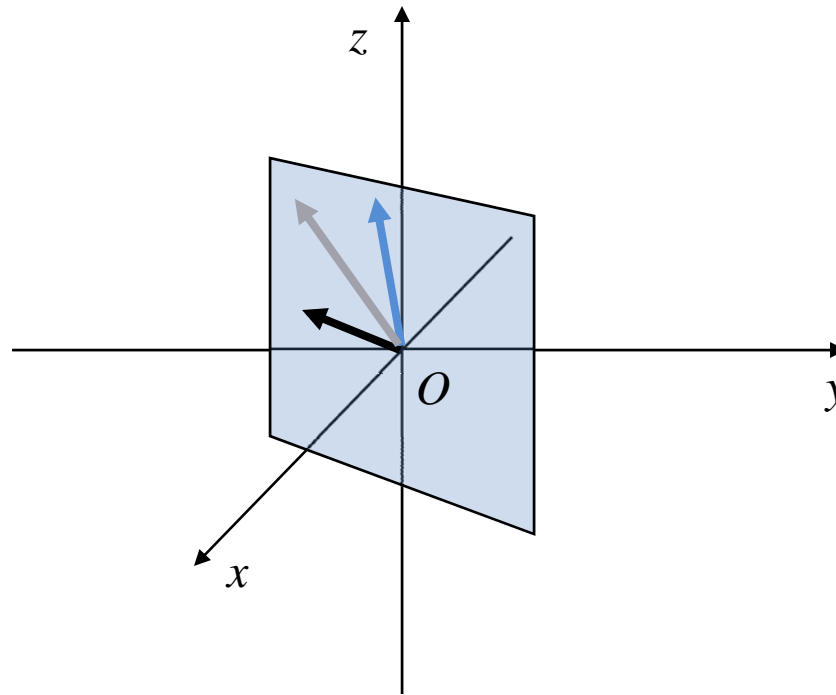
## Vector space

- Informal definition:
  - $V \neq \emptyset$  (a non-empty set of vectors)
  - $\mathbf{v}, \mathbf{w} \in V \Rightarrow \mathbf{v} + \mathbf{w} \in V$  (closed under addition)
  - $\mathbf{v} \in V, \alpha$  is scalar  $\Rightarrow \alpha\mathbf{v} \in V$  (closed under multiplication by scalar)
- Formal definition includes axioms about associativity and distributivity of the  $+$  and  $\cdot$  operators.
- $0 \in V$  always!

# Linear algebra recap

Vector space – example

- Let  $\pi$  be a plane through the origin in 3D
- $V = \{\mathbf{p} - \mathbf{O} / \mathbf{p} \in \pi\}$  is a linear subspace of  $\mathbb{R}^3$



# Linear algebra recap

## Linear independence

- The vectors  $\{\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_k\}$  are a linearly independent set if:

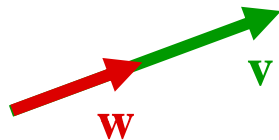
$$\alpha_1 \mathbf{v}_1 + \alpha_2 \mathbf{v}_2 + \dots + \alpha_k \mathbf{v}_k = \mathbf{0} \iff \alpha_i = 0 \quad \forall i$$

- It means that none of the vectors can be obtained as a linear combination of the others.

# Linear algebra recap

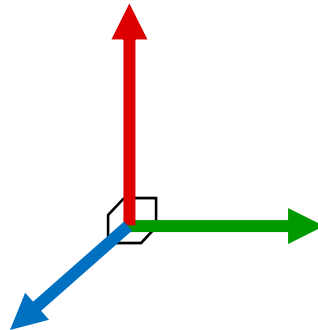
## Linear independence

- Parallel vectors are always dependent:



$$\mathbf{v} = 2.4 \mathbf{w} \Rightarrow \mathbf{v} + (-2.4)\mathbf{w} = \mathbf{0}$$

- Orthogonal vectors are always independent.





# Linear algebra recap

Basis of a vector space  $V$

- $\{\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_n\}$  are **linearly independent**
- $\{\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_n\}$  **span** the whole vector space  $V$ :

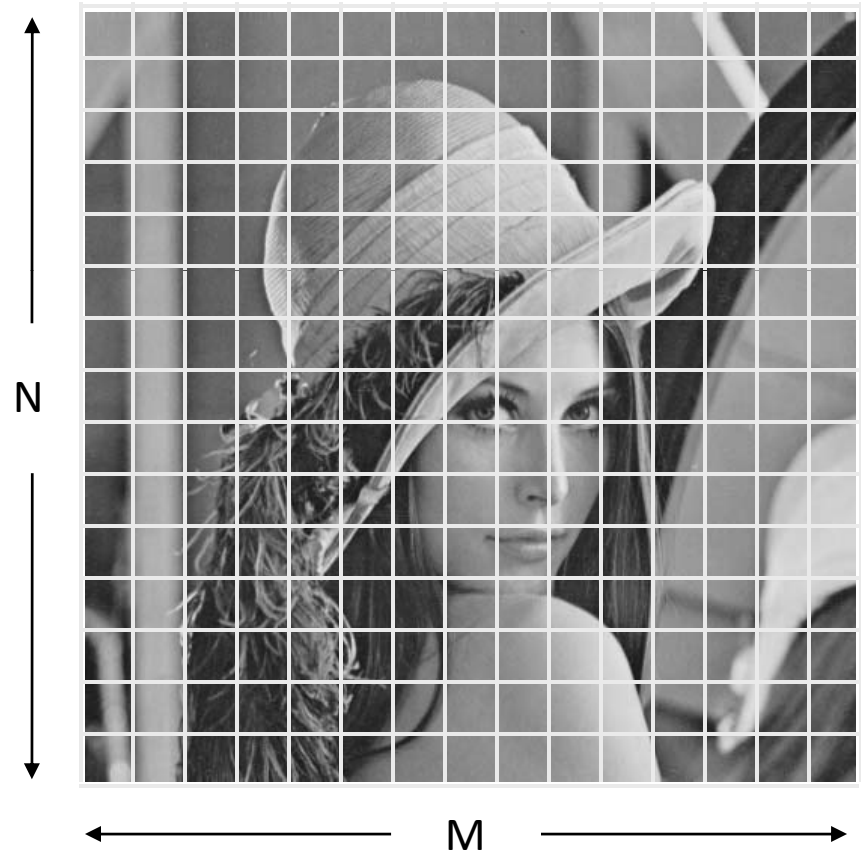
$$V = \{ \alpha_1 \mathbf{v}_1 + \alpha_2 \mathbf{v}_2 + \dots + \alpha_n \mathbf{v}_n \mid \alpha_i \text{ is scalar} \}$$

- Any vector in  $V$  is a **unique** linear combination of the basis.
- The number of basis vectors is called the **dimension** of  $V$ .

# Linear algebra recap

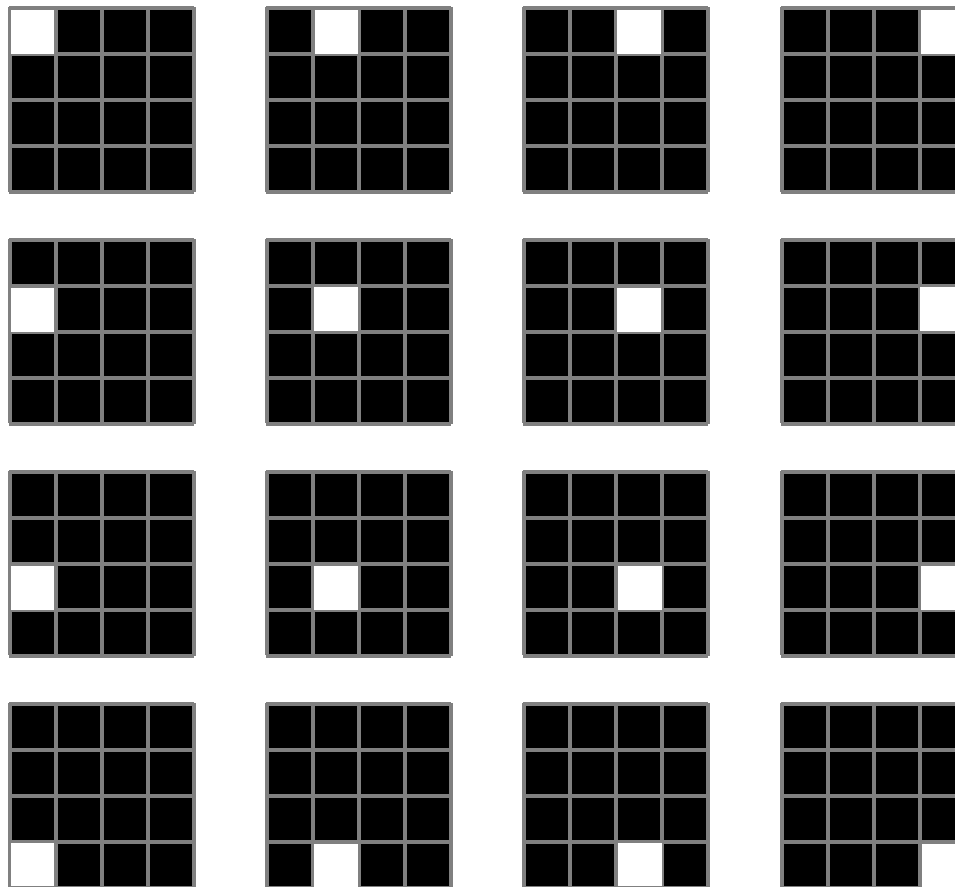
## Basis example

- Grayscale  $N \times M$  images:
  - Each pixel has value between 0 (black) and 1 (white)
  - The image can be interpreted as a vector  $\in \mathbb{R}^{N \cdot M}$



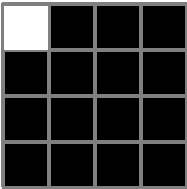
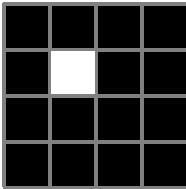
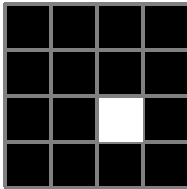
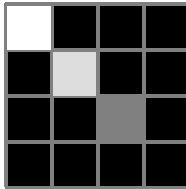
# Linear algebra recap

The “standard” basis ( $4 \times 4$ )



# Linear algebra recap

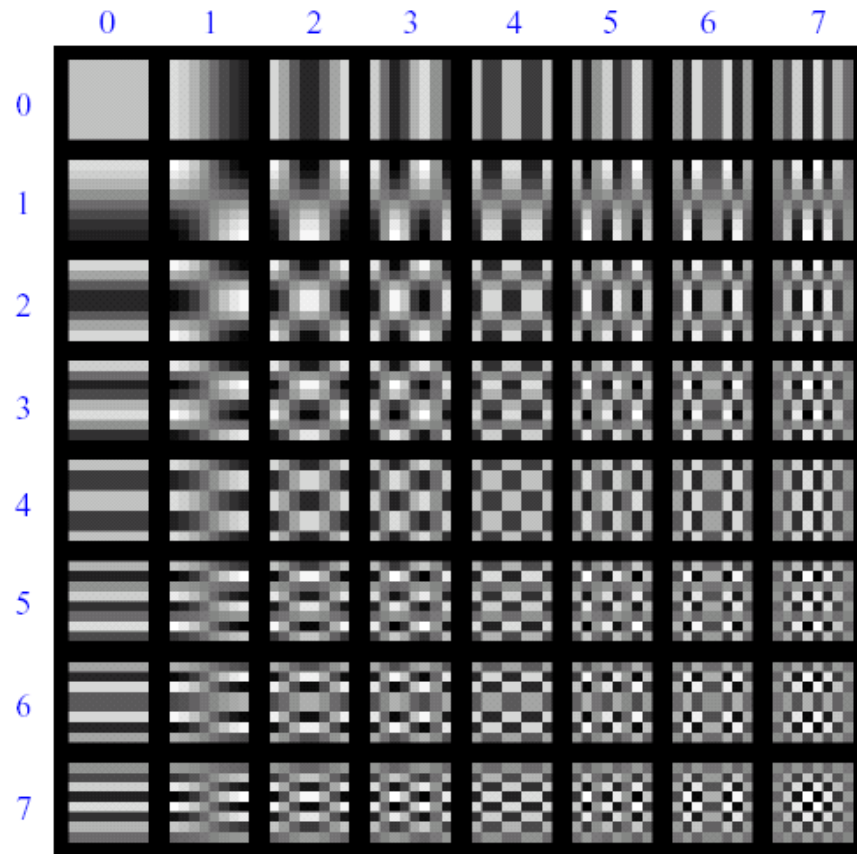
The “standard” basis (4×4) – linear combinations

 $*1 +$  $*(2/3) +$  $*(1/3) =$ 

# Linear algebra recap

Discrete cosine basis

- Used for JPEG encoding



# Linear algebra recap

Orthogonal matrices (orthonormal basis)

- Matrix  $A$  ( $n \times n$ ) is **orthogonal** if  $A^{-1} = A^T$
- Follows:  $AA^T = A^T A = I$
- The rows of  $A$  are **orthonormal vectors!**

$$I = A^T A = \begin{pmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \vdots \\ \mathbf{v}_n \end{pmatrix} \begin{pmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \vdots & \mathbf{v}_n \end{pmatrix} = \begin{pmatrix} \mathbf{v}_i^T \mathbf{v}_j \end{pmatrix} = \begin{pmatrix} \delta_{ij} \end{pmatrix}$$

$$\Rightarrow \langle \mathbf{v}_i, \mathbf{v}_i \rangle = 1 \Rightarrow \|\mathbf{v}_i\| = 1; \quad \langle \mathbf{v}_i, \mathbf{v}_j \rangle = 0$$

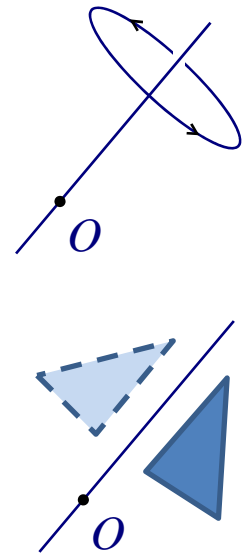
# Linear algebra recap

## Orthogonal transformations

- $A$  is orthogonal matrix  $\Rightarrow A$  represents a linear transformation that **preserves dot product** (i.e., preserves lengths and angles):

$$(A\mathbf{v})^T (A\mathbf{w}) = \mathbf{v}^T A^T A \mathbf{w} = \mathbf{v}^T \mathbf{w}$$

- Therefore,  $\|A\mathbf{v}\| = \|\mathbf{v}\|$  and  $\angle(A\mathbf{v}, A\mathbf{w}) = \angle(\mathbf{v}, \mathbf{w})$



# Linear algebra recap

## Eigenvectors and eigenvalues

- $A$  is a square  $n \times n$  matrix
- $\mathbf{v}$  is called **eigenvector** of  $A$  if:
  - $A\mathbf{v} = \lambda\mathbf{v}$  ( $\lambda$  is a scalar)
  - $\mathbf{v} \neq 0$
- The scalar  $\lambda$  is called **eigenvalue**

$$A\mathbf{v} = \lambda\mathbf{v}$$

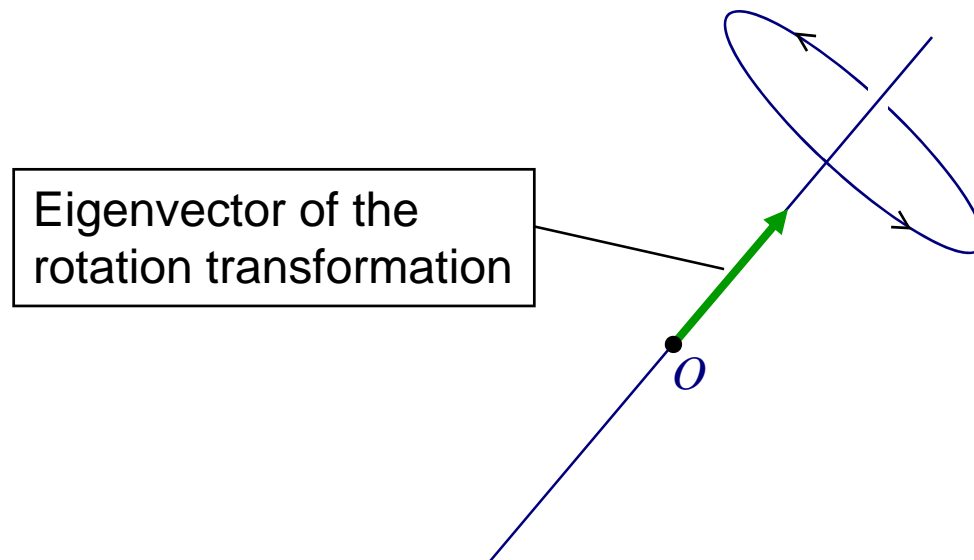
- $A\mathbf{v} = \lambda\mathbf{v} \Rightarrow A(\alpha\mathbf{v}) = \lambda(\alpha\mathbf{v}) \Rightarrow \alpha\mathbf{v}$  is also eigenvector
- $A\mathbf{v} = \lambda\mathbf{v}, A\mathbf{w} = \lambda\mathbf{w} \Rightarrow A(\mathbf{v} + \mathbf{w}) = \lambda(\mathbf{v} + \mathbf{w})$
- Therefore, eigenvectors of the same  $\lambda$  form a **linear subspace**.



# Linear algebra recap

## Eigenvectors and eigenvalues

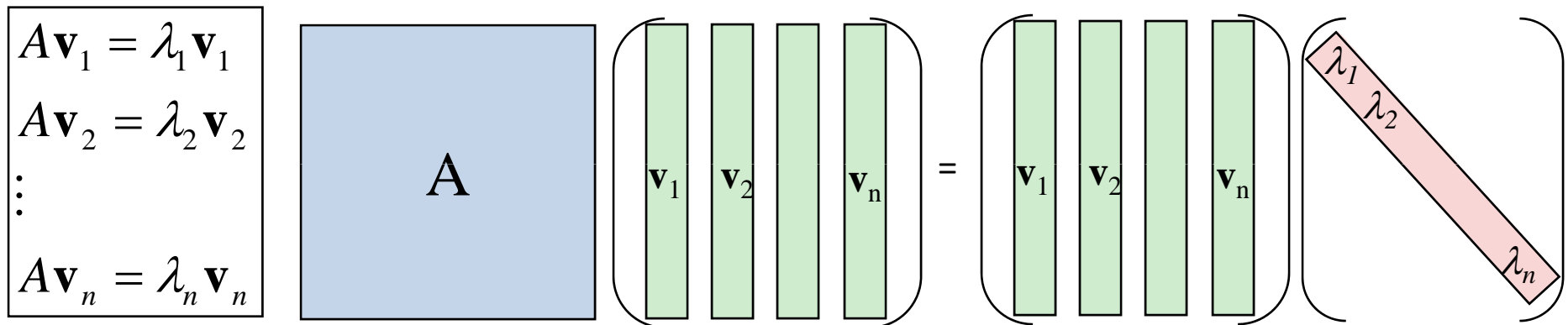
- An eigenvector spans an **axis (subspace of dimension 1)** that remains the same under the transformation  $A$ .
- Example – the axis of rotation:



# Linear algebra recap

## Spectrum and diagonalization

- The set of all the eigenvalues of  $A$  is called the spectrum of  $A$ .
- $A$  is diagonalizable if  $A$  has  $n$  independent eigenvectors. Then:  $AV = VD$



# Linear algebra recap

## Spectrum and diagonalization

- Therefore,  $A = VDV^{-1}$ , where  $D$  is diagonal
- $A$  represents a scaling along the eigenvector axes!

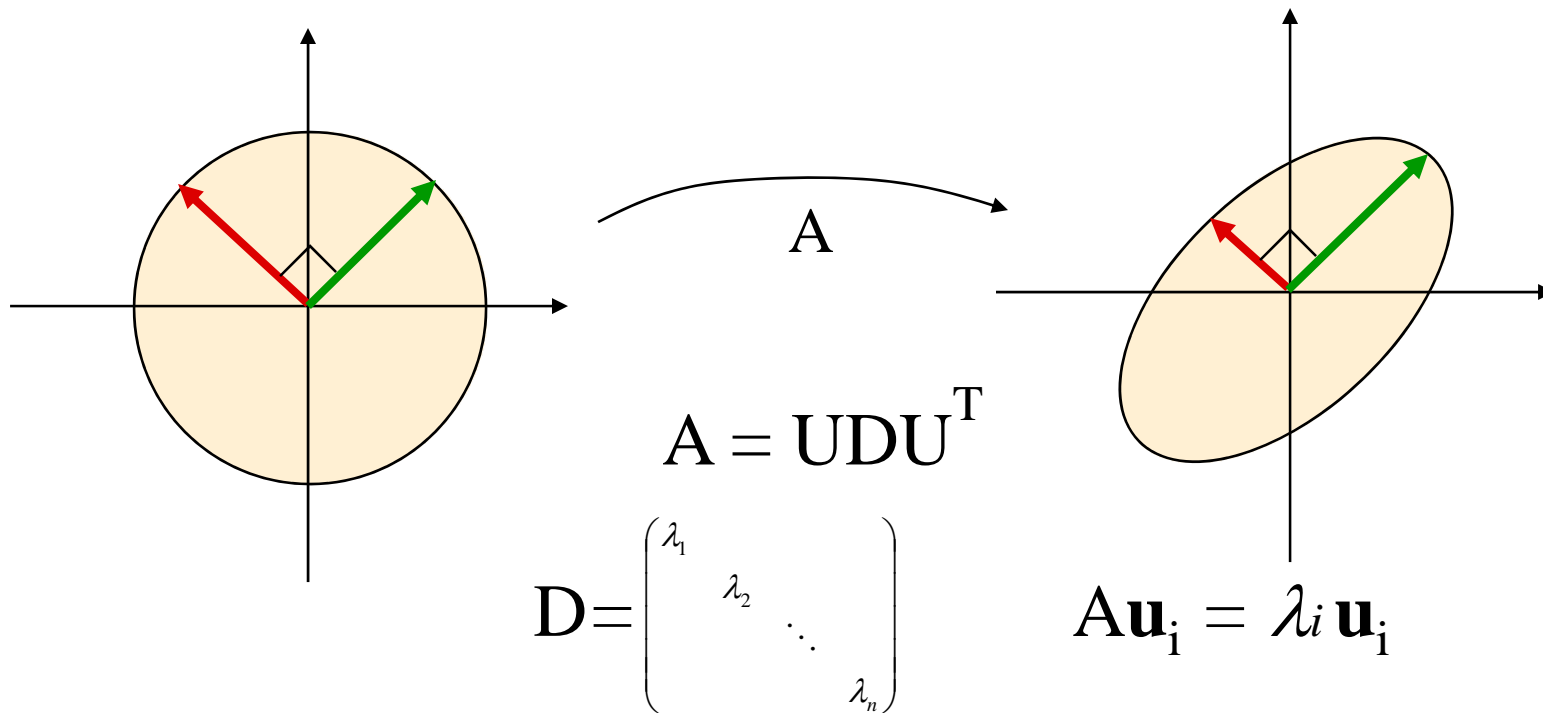
$$A\mathbf{v}_1 = \lambda_1\mathbf{v}_1$$
$$A\mathbf{v}_2 = \lambda_2\mathbf{v}_2$$
$$\vdots$$
$$A\mathbf{v}_n = \lambda_n\mathbf{v}_n$$

$$A = VDV^{-1}$$

# Linear algebra recap

Symmetric matrices

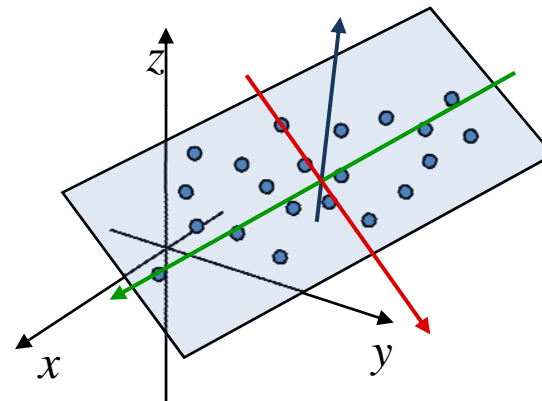
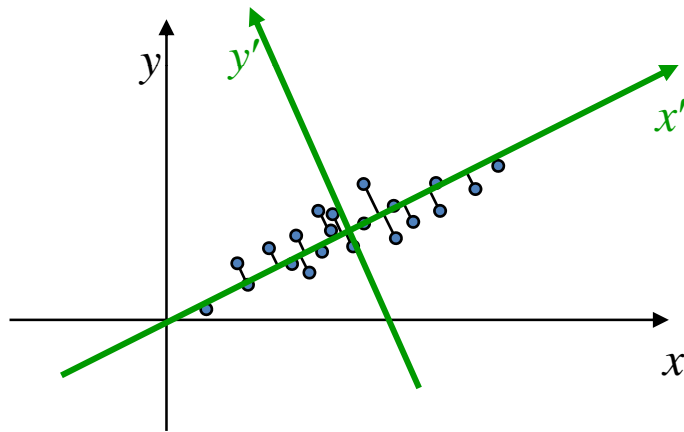
- If  $A$  is *symmetric*, the eigenvectors are *orthogonal* and there's *always an eigenbasis*.



# Principal Component Analysis

Basic idea

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

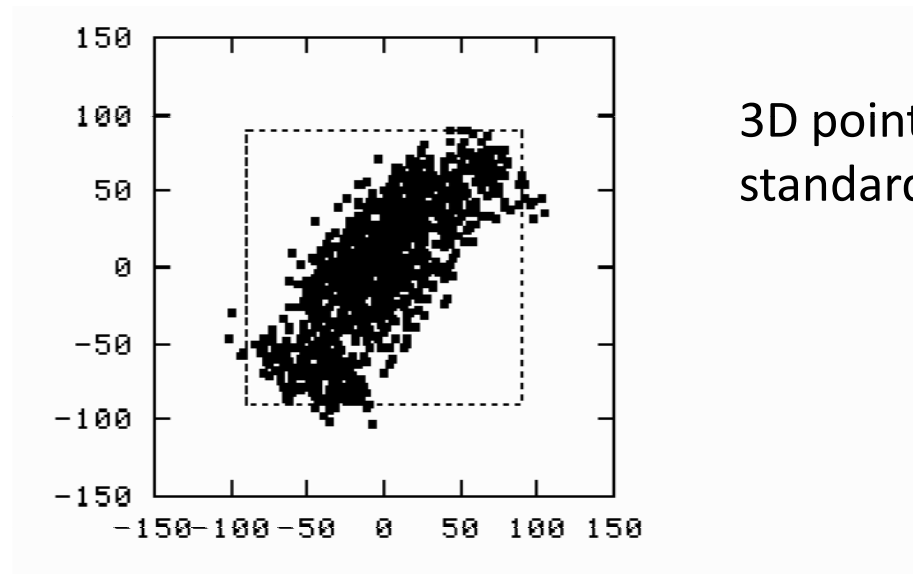


- PCA finds a best approximating line/plane/axes... (in terms of  $\sum distances^2$ )

# Principal Component Analysis

Basic idea

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

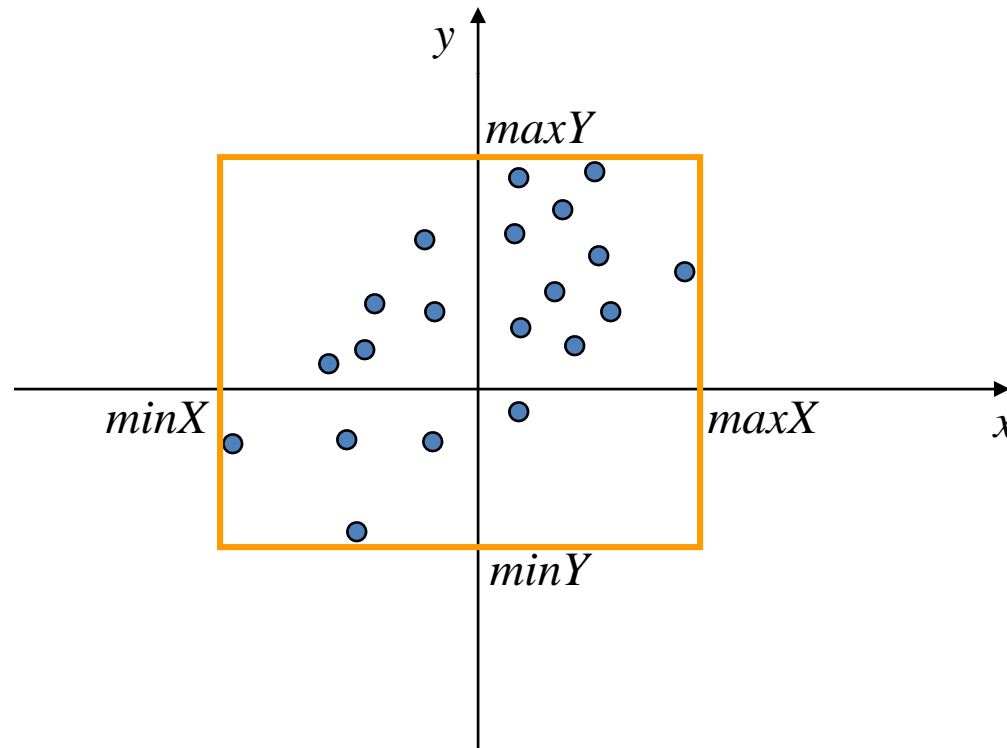


- PCA finds a best approximating line/plane/axes... (in terms of  $\sum distances^2$ )

# Principal Component Analysis

## Applications

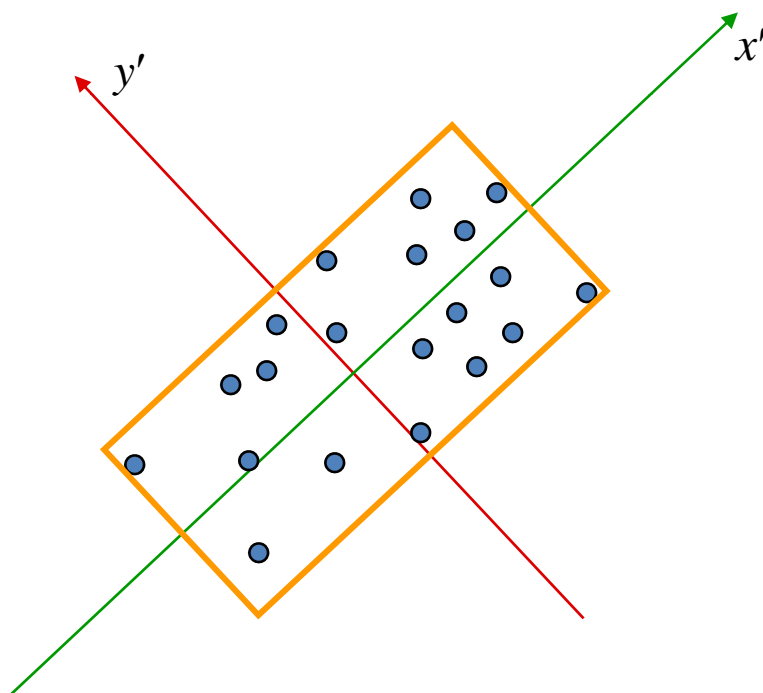
- An axis-aligned bounding box: agrees with the standard axes



# Principal Component Analysis

Application: oriented bounding box

- Tighter fit

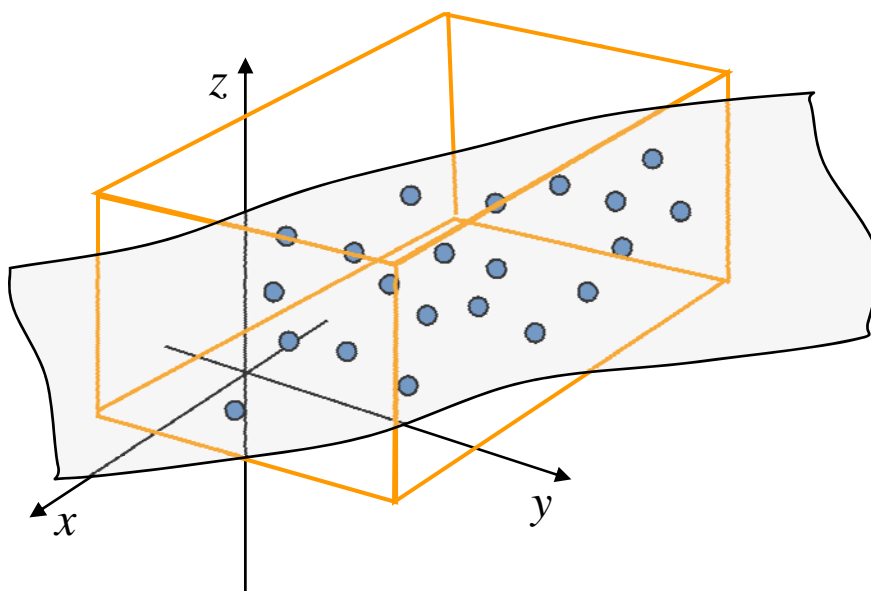




# Principal Component Analysis

Application: oriented bounding box

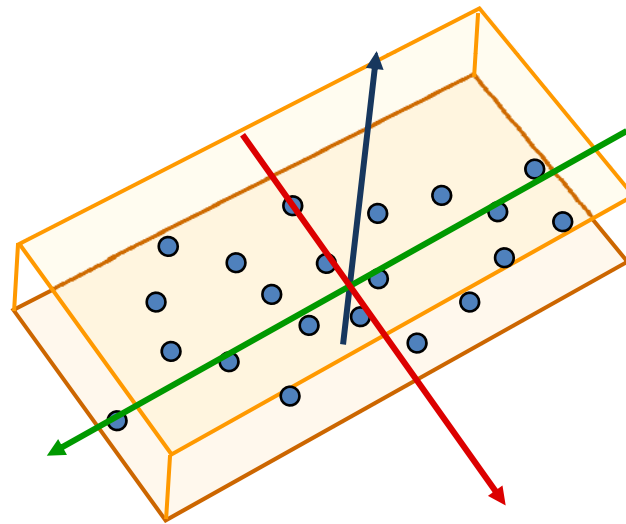
- Axis aligned bounding box



# Principal Component Analysis

Application: oriented bounding box

- Oriented bounding box by PCA

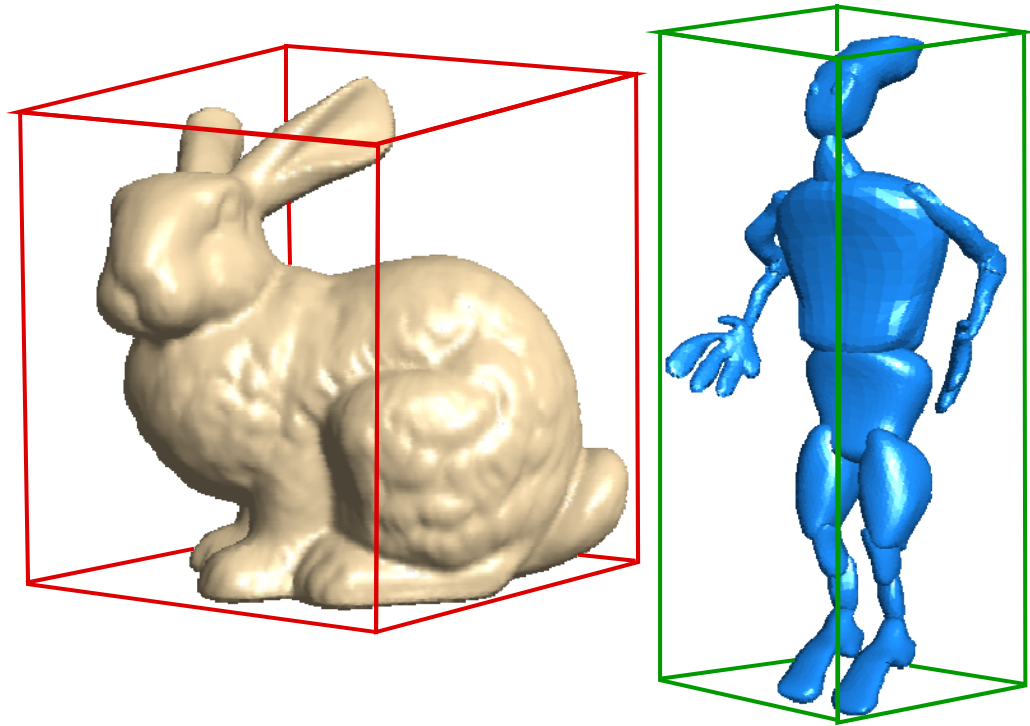


# Principal Component Analysis

Application: oriented bounding box

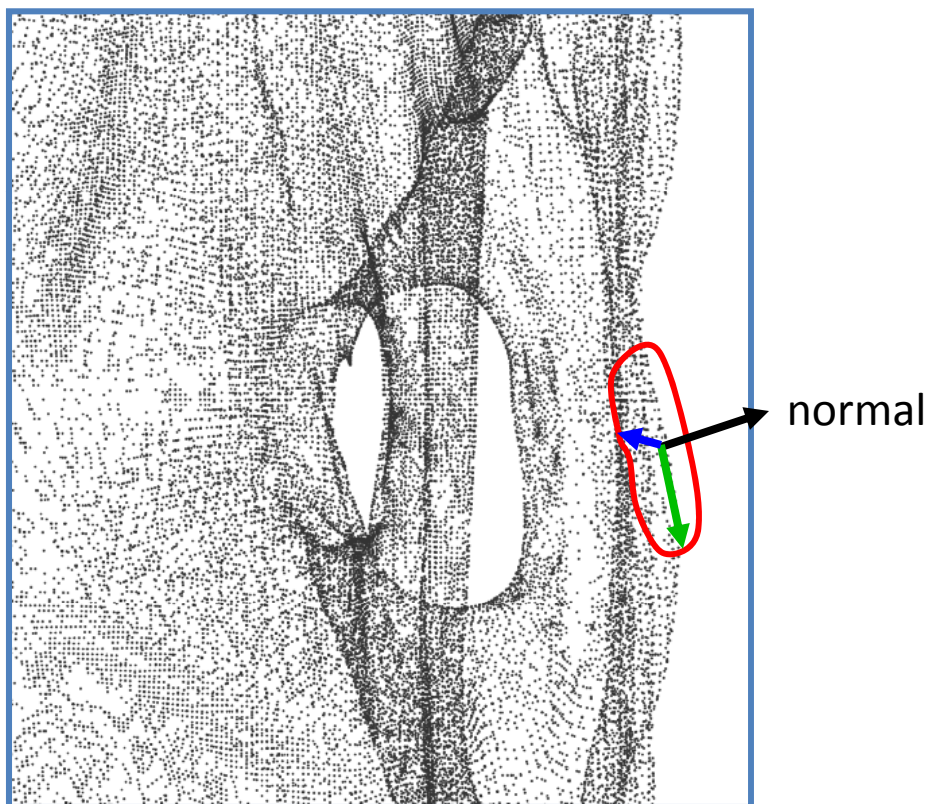
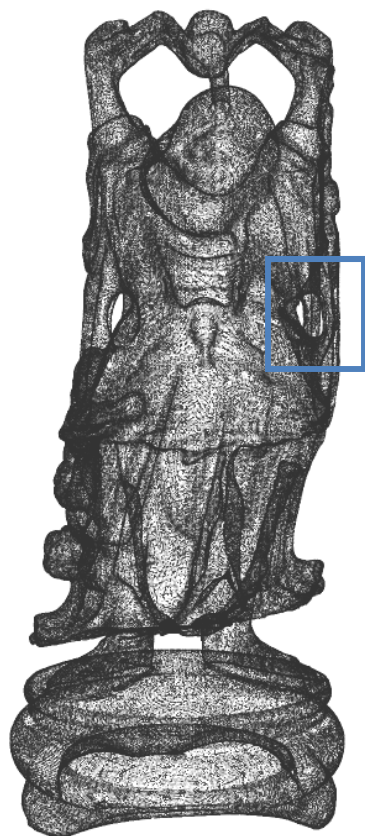
- Serve as very simple “approximation” of the object
- Fast collision detection, visibility queries
- Whenever we need to know the dimensions (size) of the object

- 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



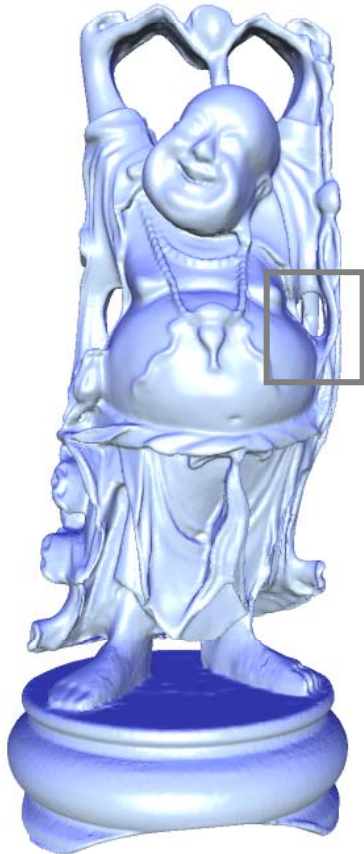
# Principal Component Analysis

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/>)



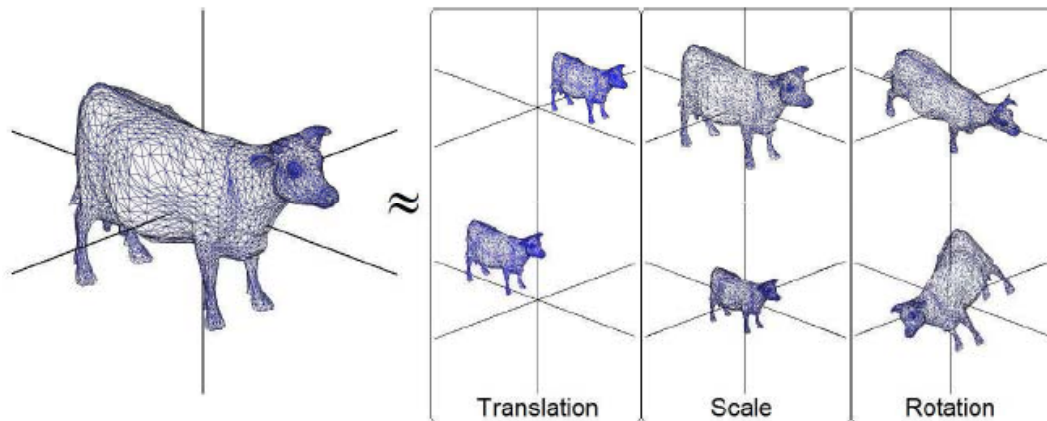
Query



# Principal Component Analysis

Application: shape alignment

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

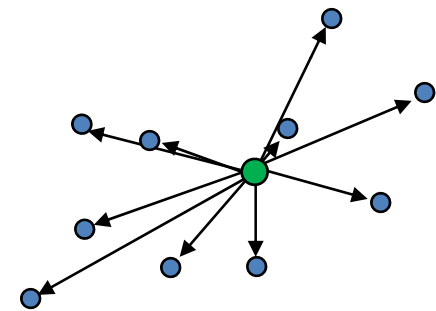


# Notations

- Denote our data points by  $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n \in R^d$

- Center of mass:

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



- Vectors from the centroid:

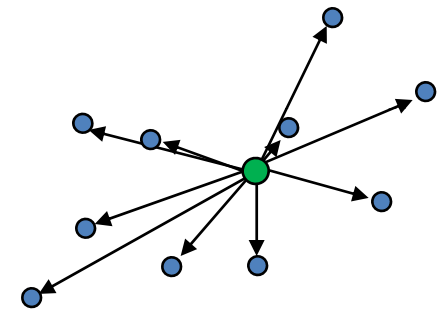
$$\mathbf{y}_i = \mathbf{x}_i - \mathbf{m}$$



# 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:

$$\mathbf{m} = \operatorname{argmin}_{\mathbf{x}} \sum_{i=1}^n \|\mathbf{x}_i - \mathbf{x}\|^2$$



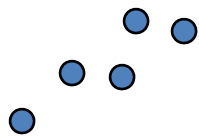
$$\mathbf{m} = \frac{1}{n} \sum_{i=1}^n \mathbf{x}_i$$

# Variance of projected points

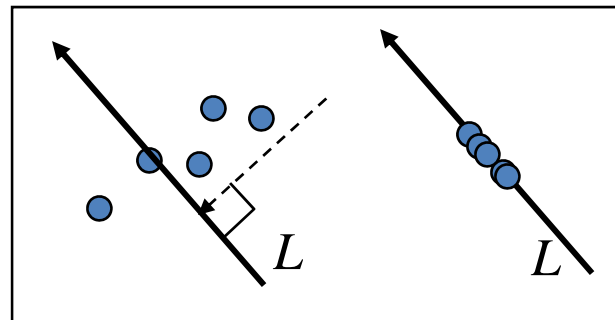
- Let us measure the variance (scatter) of our points in different directions
- Let's look at a **line**  $L$  through the center of mass  $\mathbf{m}$ , and project our points  $\mathbf{x}_i$  onto it. The **variance** of the **projected** points  $\mathbf{x}'_i$  is:

Want to find directions of maximal/minimal variance

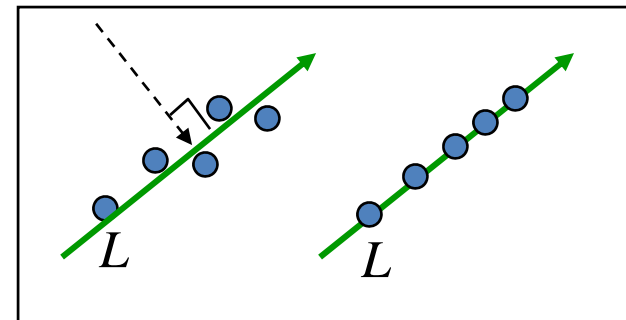
$$\text{var}(L) = \frac{1}{n} \sum_{i=1}^n \|\mathbf{x}'_i - \mathbf{m}\|^2$$



Original set



Small variance

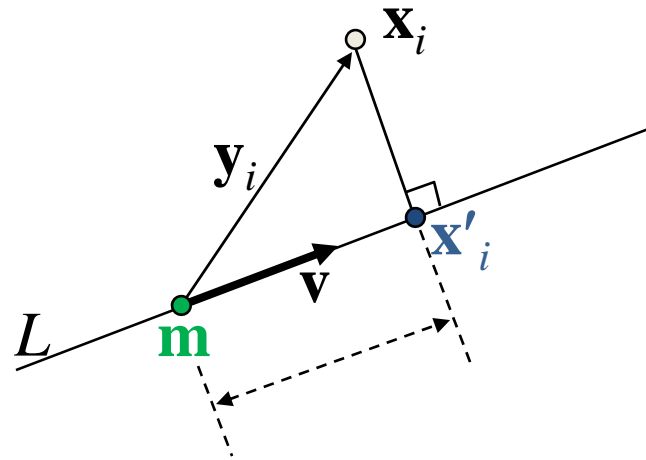


Large variance

# Variance of projected points

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

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



# Variance of projected points

- So,
$$\begin{aligned}\text{var}(L) &= \frac{1}{n} \sum_{i=1}^n \|\mathbf{x}'_i - \mathbf{m}\|^2 = \frac{1}{n} \sum_{i=1}^n (\mathbf{y}_i^T \mathbf{v})^2 = \frac{1}{n} \|\mathbf{Y}^T \mathbf{v}\|^2 = \\ &= \frac{1}{n} (\mathbf{Y}^T \mathbf{v})^T (\mathbf{Y}^T \mathbf{v}) = \frac{1}{n} \mathbf{v}^T \mathbf{Y} \mathbf{Y}^T \mathbf{v} = \mathbf{v}^T \mathbf{S} \mathbf{v}.\end{aligned}$$

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

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

- The scatter matrix  $\mathbf{S}$  measures the variance of our points

# Directions of maximal variance

- So, we have:  $\text{var}(L) = \mathbf{v}^T \mathbf{S} \mathbf{v}$

- Theorem:

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

$$f(\mathbf{v}) = \mathbf{v}^T \mathbf{S} \mathbf{v} \text{ (and } S \text{ is a symmetric matrix).}$$

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

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

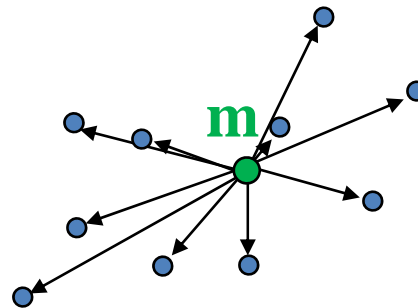
$$\mathbf{S} \mathbf{v} + \lambda \mathbf{v} = 0$$

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

- This is the definition of an eigenvector of  $\mathbf{S}$

# Summary so far

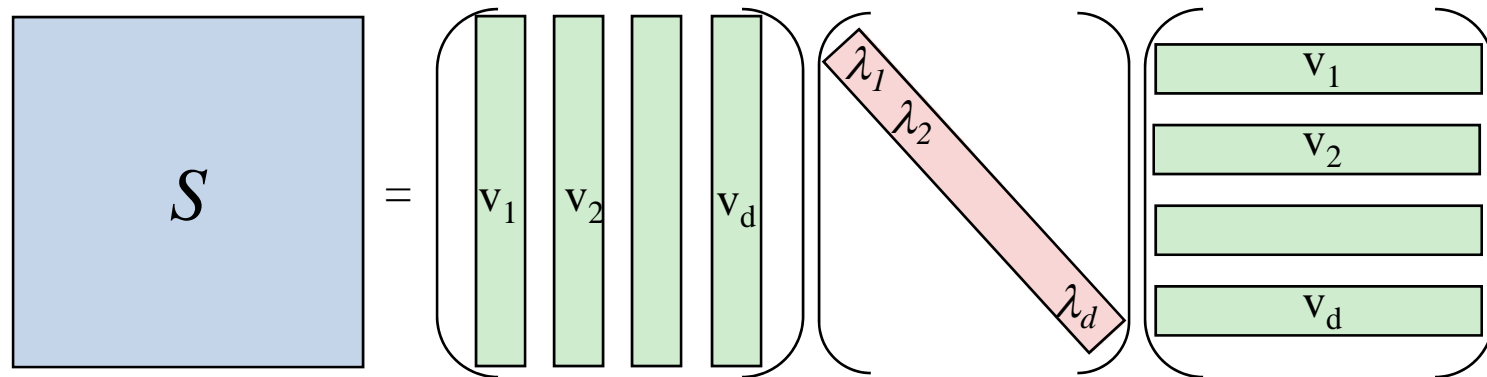
- We take the centered data vectors  $\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n \in R^d$
- Construct the scatter matrix  $S = Y Y^T$
- $S$  measures the variance of the data points
- Eigenvectors of  $S$  are directions of maximal variance.



# Scatter matrix eigendecomposition

- $S$  is symmetric

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



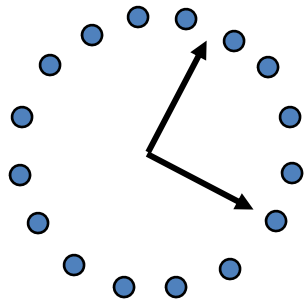
The eigenvectors form  
orthogonal basis



# 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...

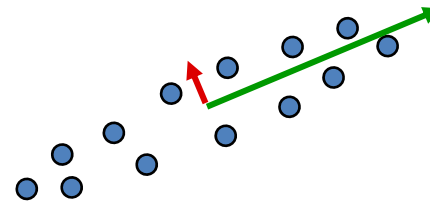
# Principal components



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

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

- Any vector is an eigenvector



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

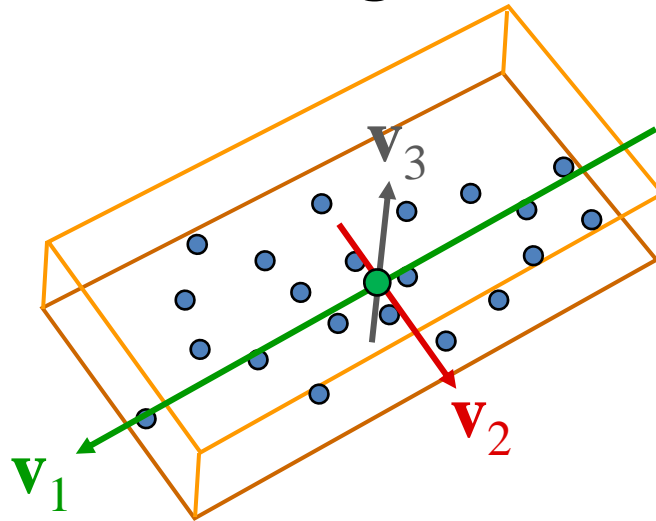
$$\mathbf{V} \begin{pmatrix} \lambda & \\ & \mu \end{pmatrix} \mathbf{V}^T$$

- $\mu$  is close to zero, much smaller than  $\lambda$

# How to use what we got

## Oriented bounding box

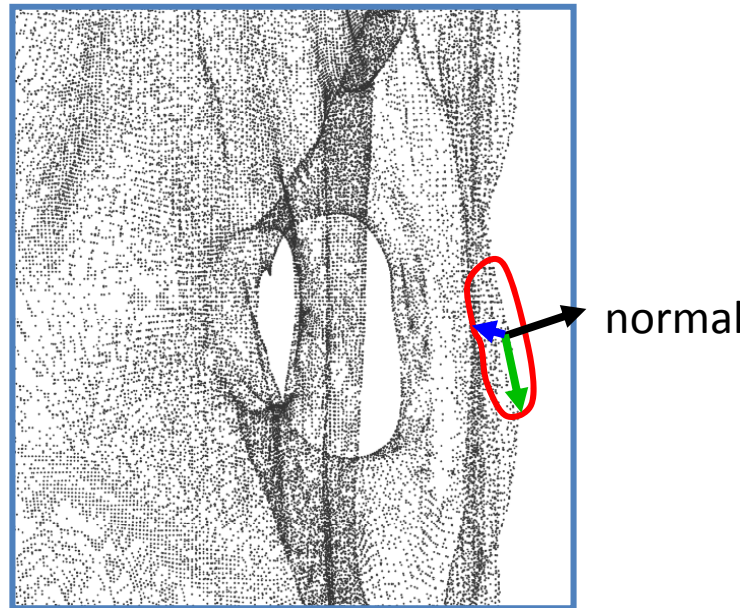
- For finding oriented bounding box or alignment – we simply compute the bounding box with respect to the axes defined by the eigenvectors. The origin is at the centroid  $\mathbf{m}$ .



# How to use what we got

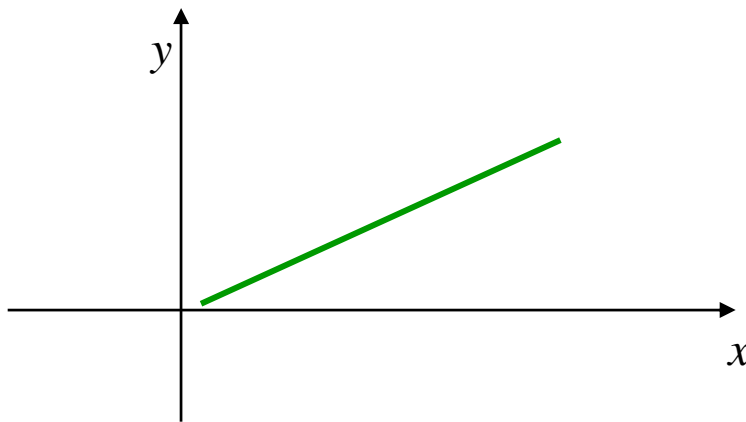
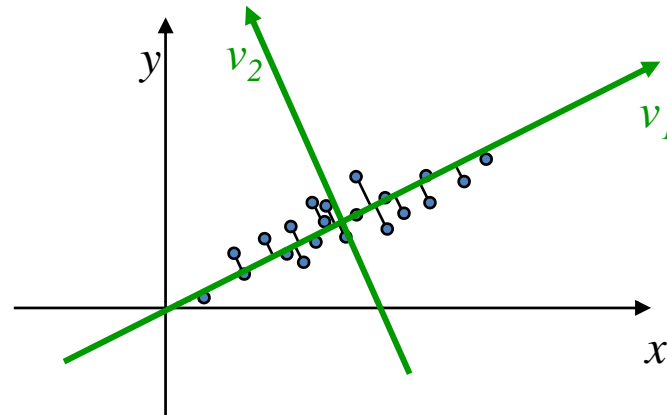
Local frame/normal estimation

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

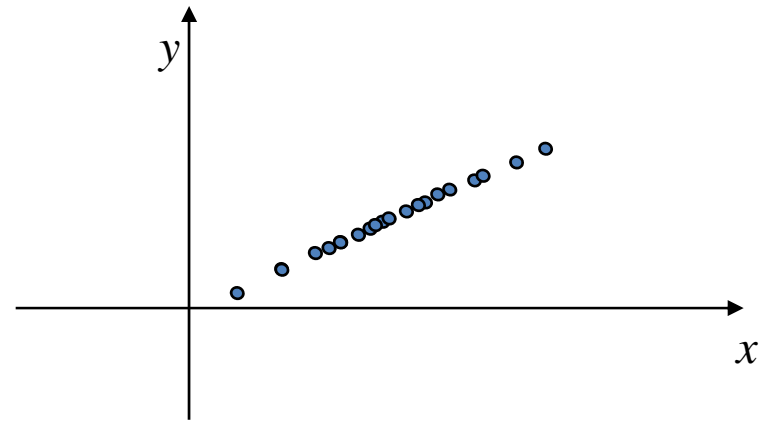


# How to use what we got

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  $\mathbb{R}^{64 \cdot 64}$
- 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:

$$\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_d$$

- 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:

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

Projection:

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



# Technical remarks:

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

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

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