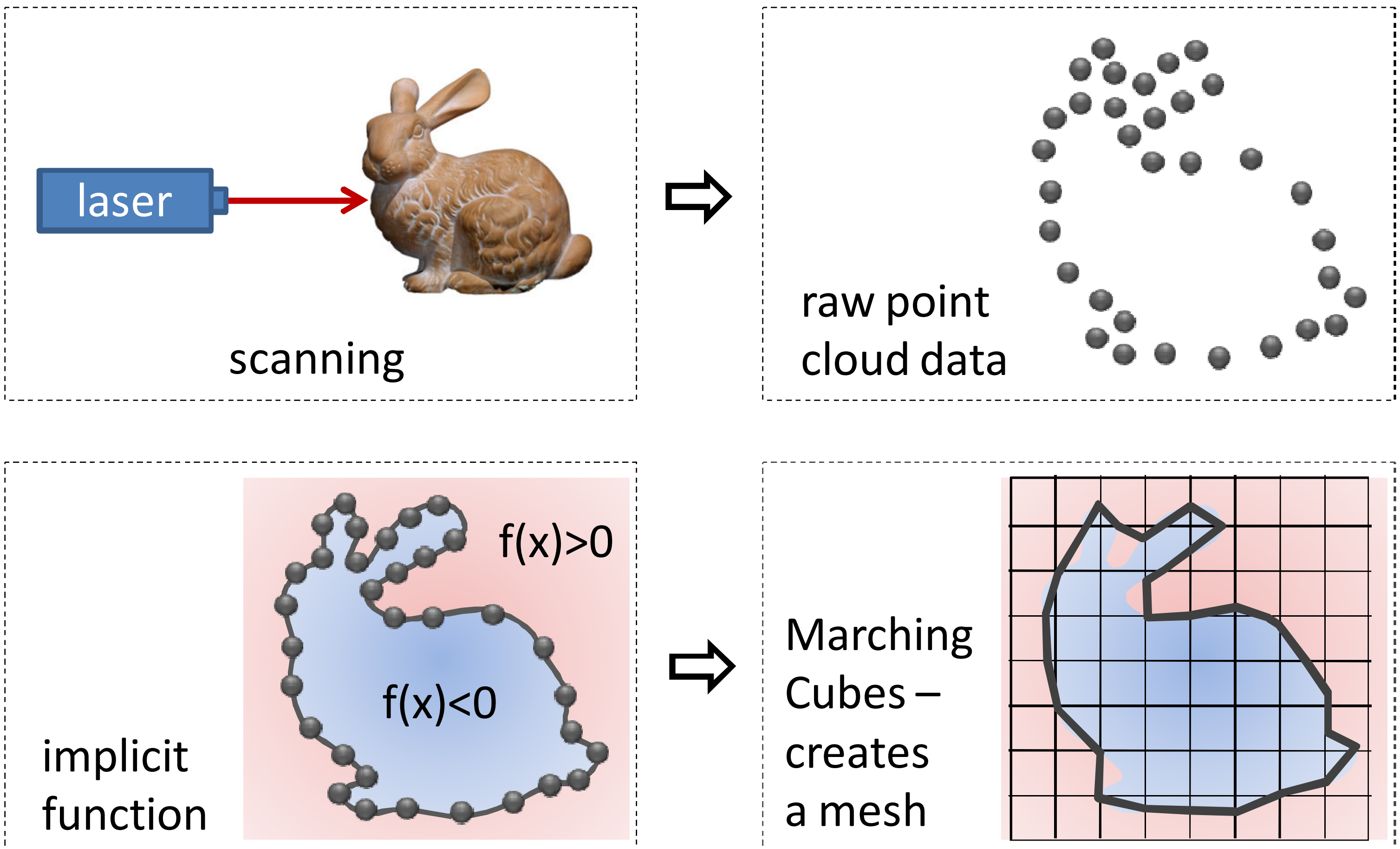
CS 523: Computer Graphics, Spring 2011

Shape Modeling

Linear algebra tools for geometric modeling

Recap

Surface acquisition and reconstruction



Recap

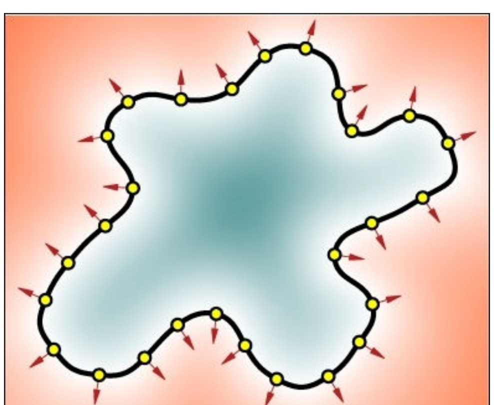
Implicit functions

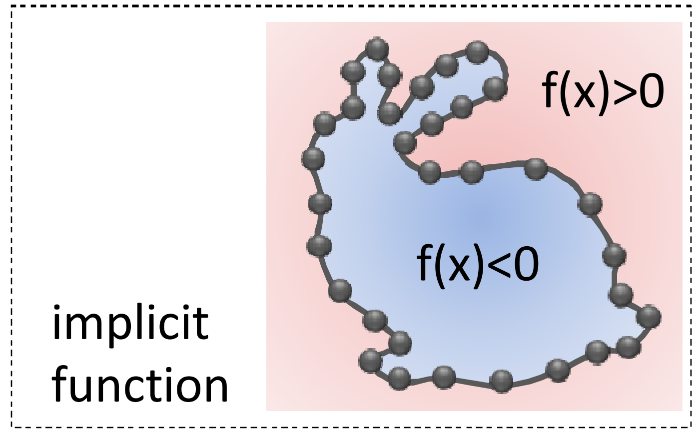
- Implicit function?

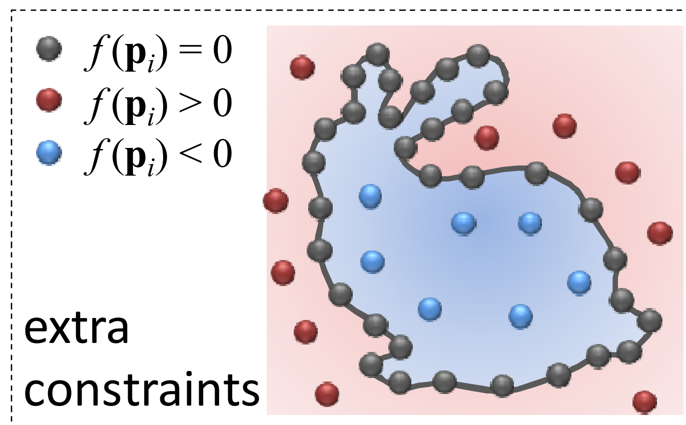
$$f(\mathbf{p}_i) = 0$$

- Need extra constraints to avoid trivial solution

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



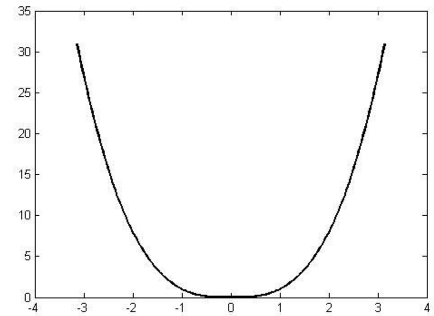




Recap
Implicit functions

- Radial basis function

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



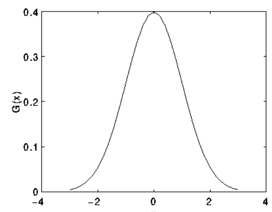
- Constraints: $f(\mathbf{p}_i) = 0,\ f(\mathbf{p}_i + \alpha\,\mathbf{n}_i) = \alpha$
- Need to solve for w_i

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

⇐ Linear problem

Recap

Implicit functions



- Moving least squares

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

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

$$f_{\mathbf{x}}(\mathbf{x}) = c_0 + c_1 x + c_2 y + c_3 x^2 + c_4 xy + c_5 y^2 \ldots$$
$$= \mathbf{c}^{\mathrm{T}} \mathbf{b}(\mathbf{x}).$$

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

⇐ 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

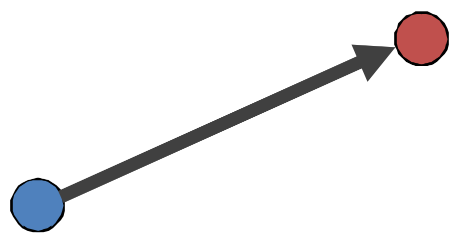
Points and Vectors

Basic definitions

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

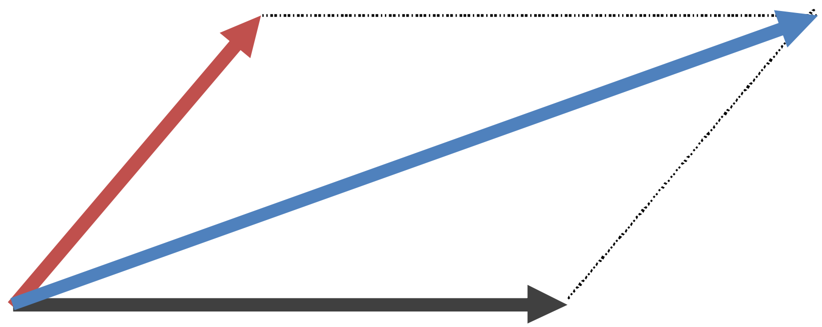
$$\text{Points} \neq \text{Vectors}$$

Point + vector = point

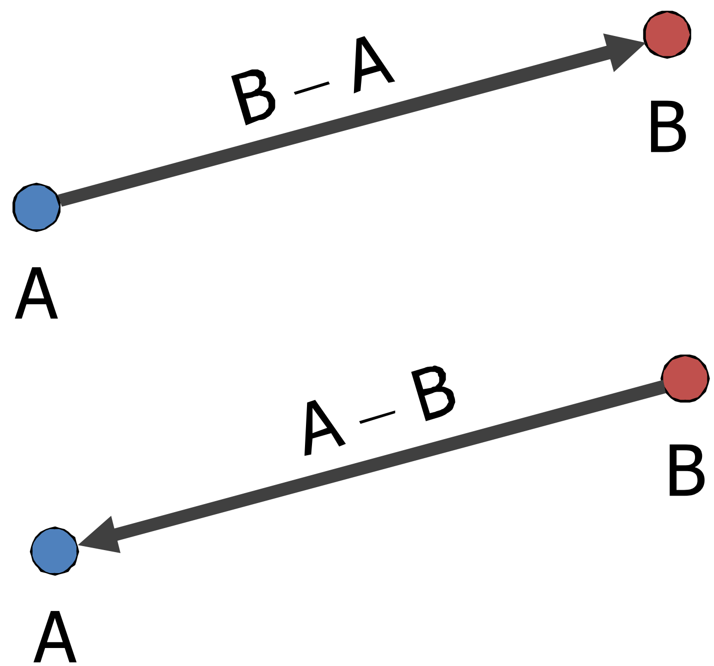


vector + vector = vector

- Parallelogram rule



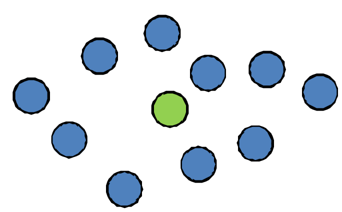
point – point = vector



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.

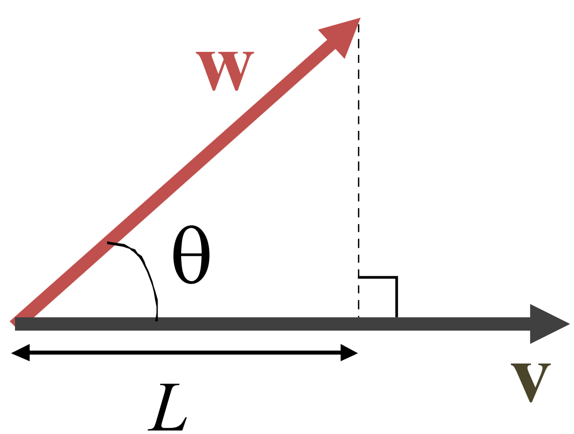
$$\mathbf{c} = \sum_{i=1}^{n} w_i \mathbf{p}_i$$



Dot product

- Defined for vectors:

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



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

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

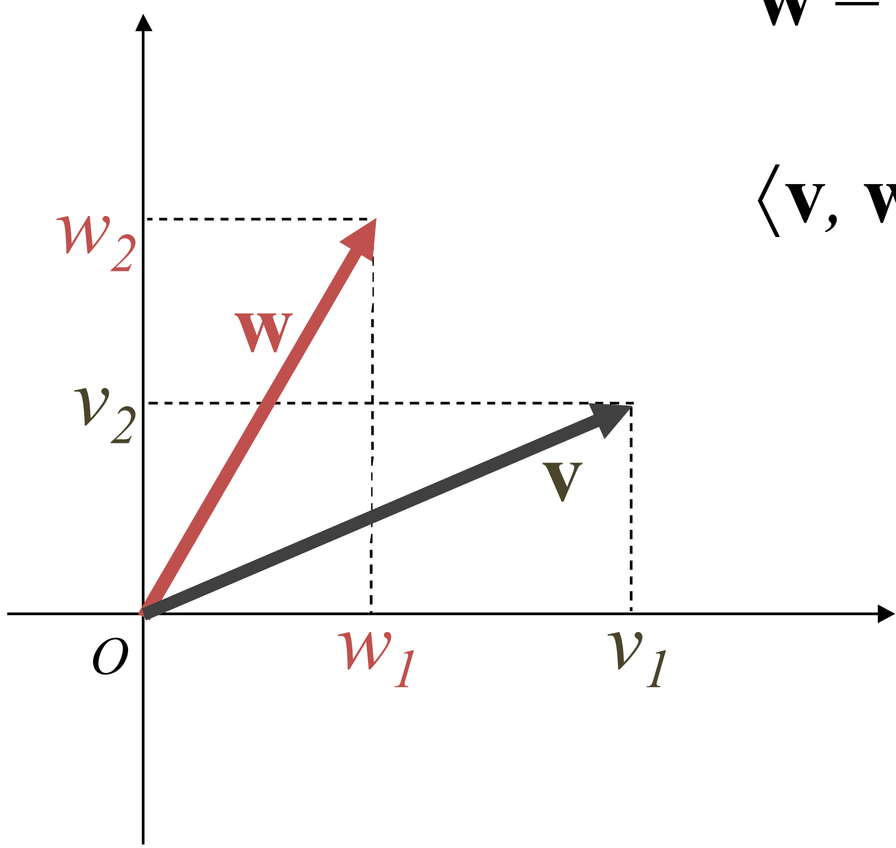
Projection of **w** onto **v**

Dot product

in coordinates

$$\mathbf{v} = (v_1, v_2, \dots, v_d)^T$$
$$\mathbf{w} = (w_1, w_2, \dots, w_d)^T$$

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



Dot product

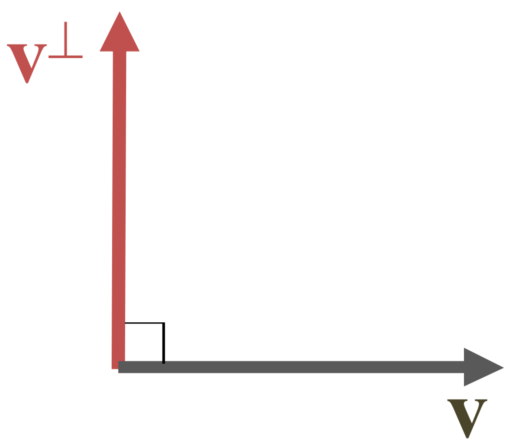
names, notations

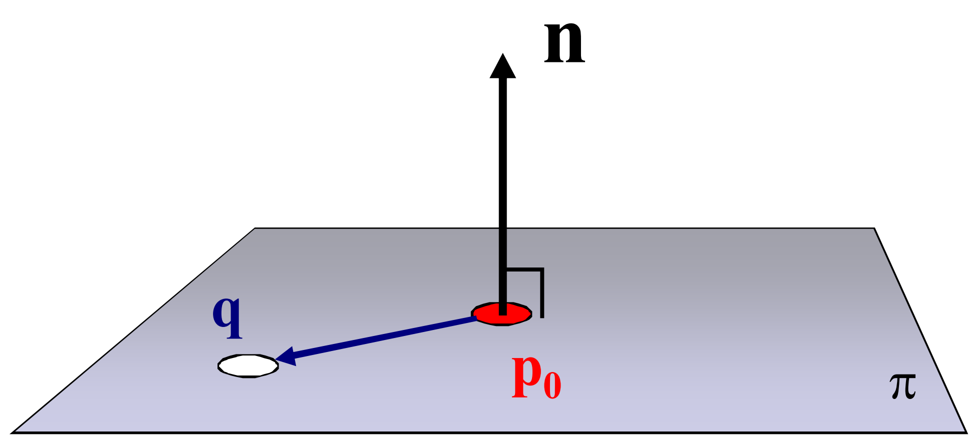
- Dot product is also called inner product
- Notations: $\langle \mathbf{v}, \mathbf{w} \rangle$ or $\mathbf{v} \cdot \mathbf{w}$ or $\mathbf{v}^{\mathrm{T}} \mathbf{w} \ (= \mathbf{w}^{\mathrm{T}} \mathbf{v})$

Dot product

Perpendicular (orthogonal) vectors

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





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

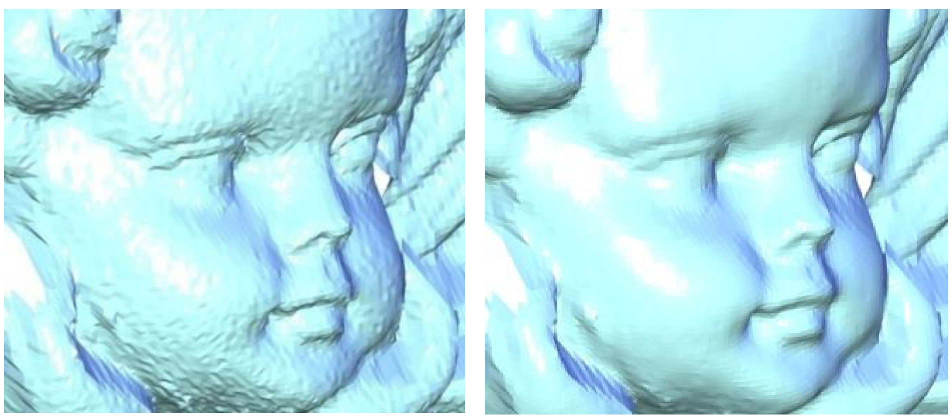
General hyper-plane:
all points $\mathbf{q}$ such that
$\langle \mathbf{q} - \mathbf{p}_0, \mathbf{n} \rangle = 0$

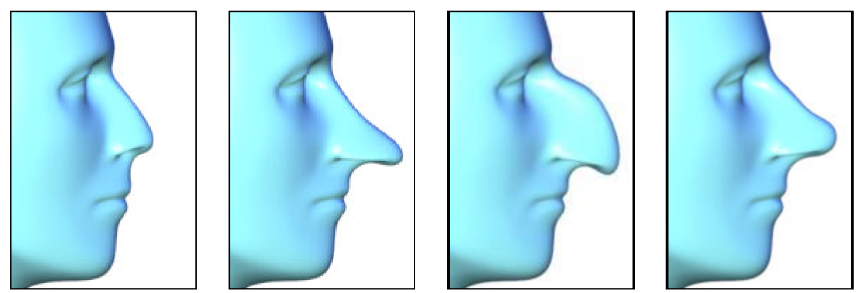
Least squares fitting

Motivation

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

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

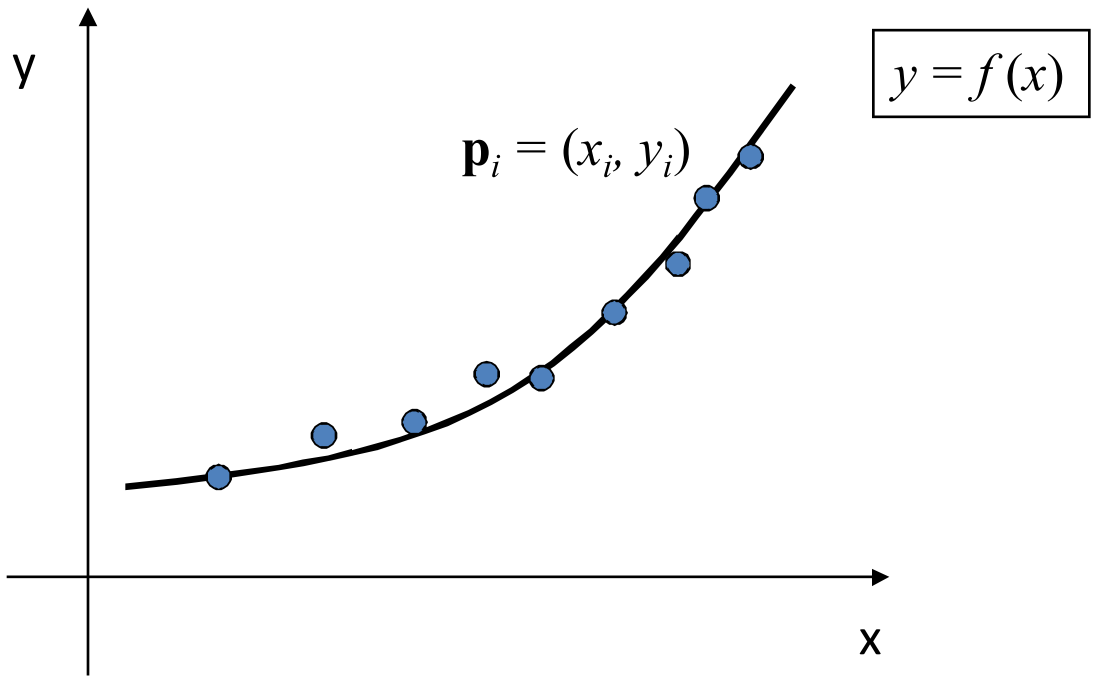




Least squares fitting

Motivation

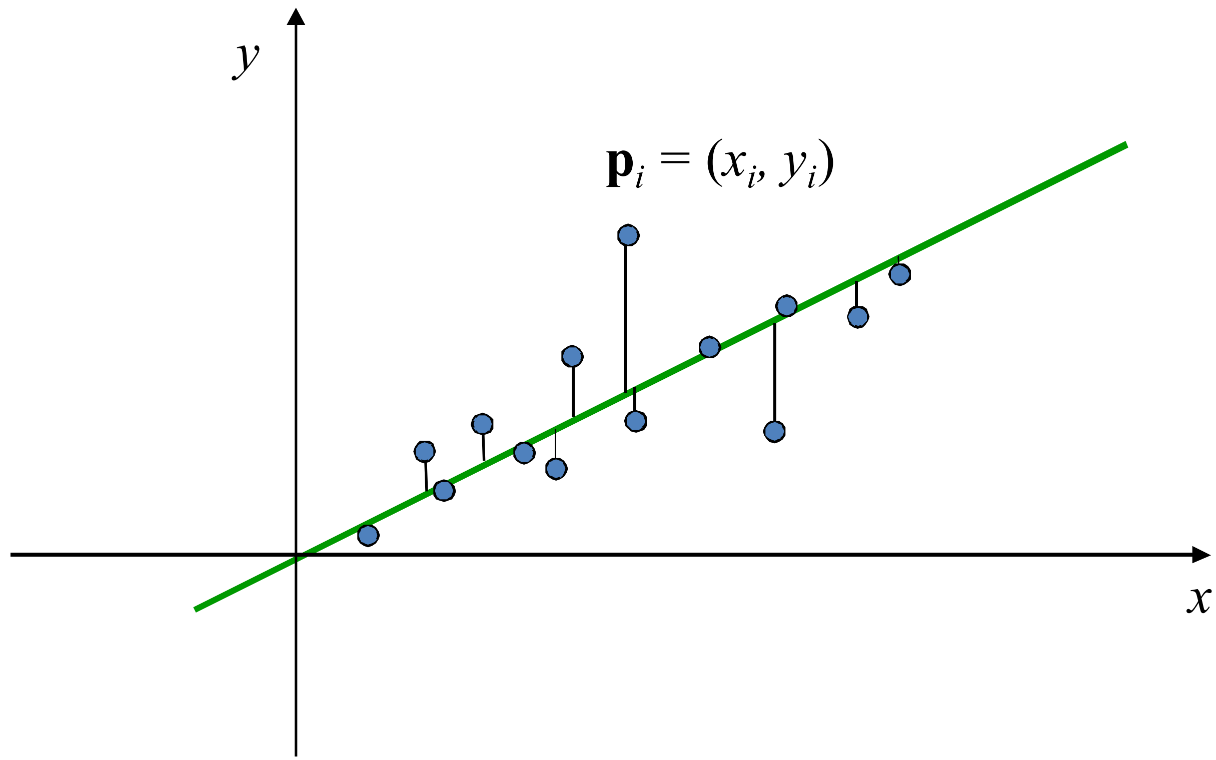
- Given data points, fit a function that is “close” to the points



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 $E(a, b)$:

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

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

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:

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

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

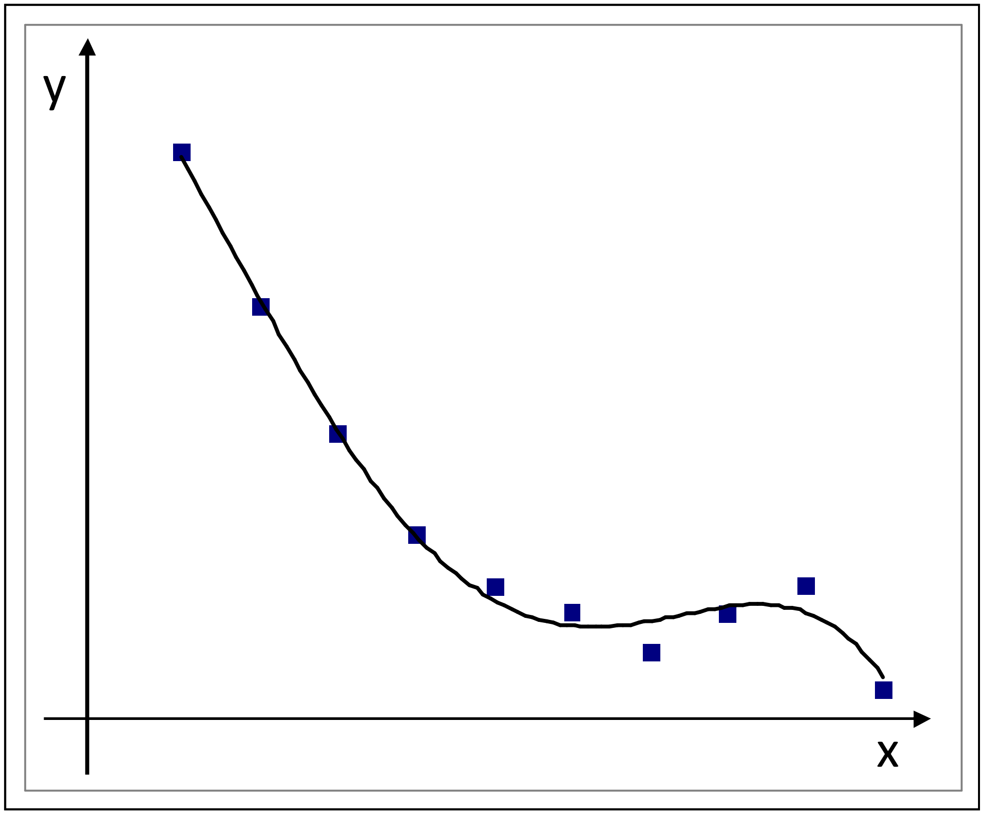
Simple example

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 $c_1, c_2, c_3 \ldots$:

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

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 $n \times k$:

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

Solving linear systems in LS sense

- In matrix form:

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

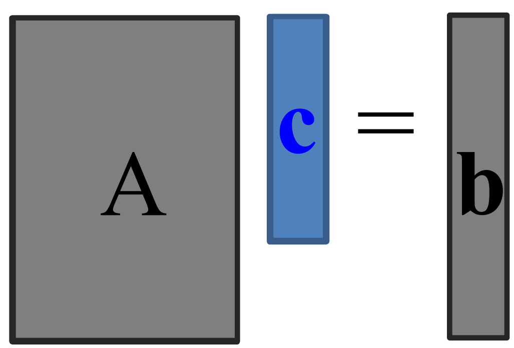
Solving linear systems in LS sense

- 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, $n > k$

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



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

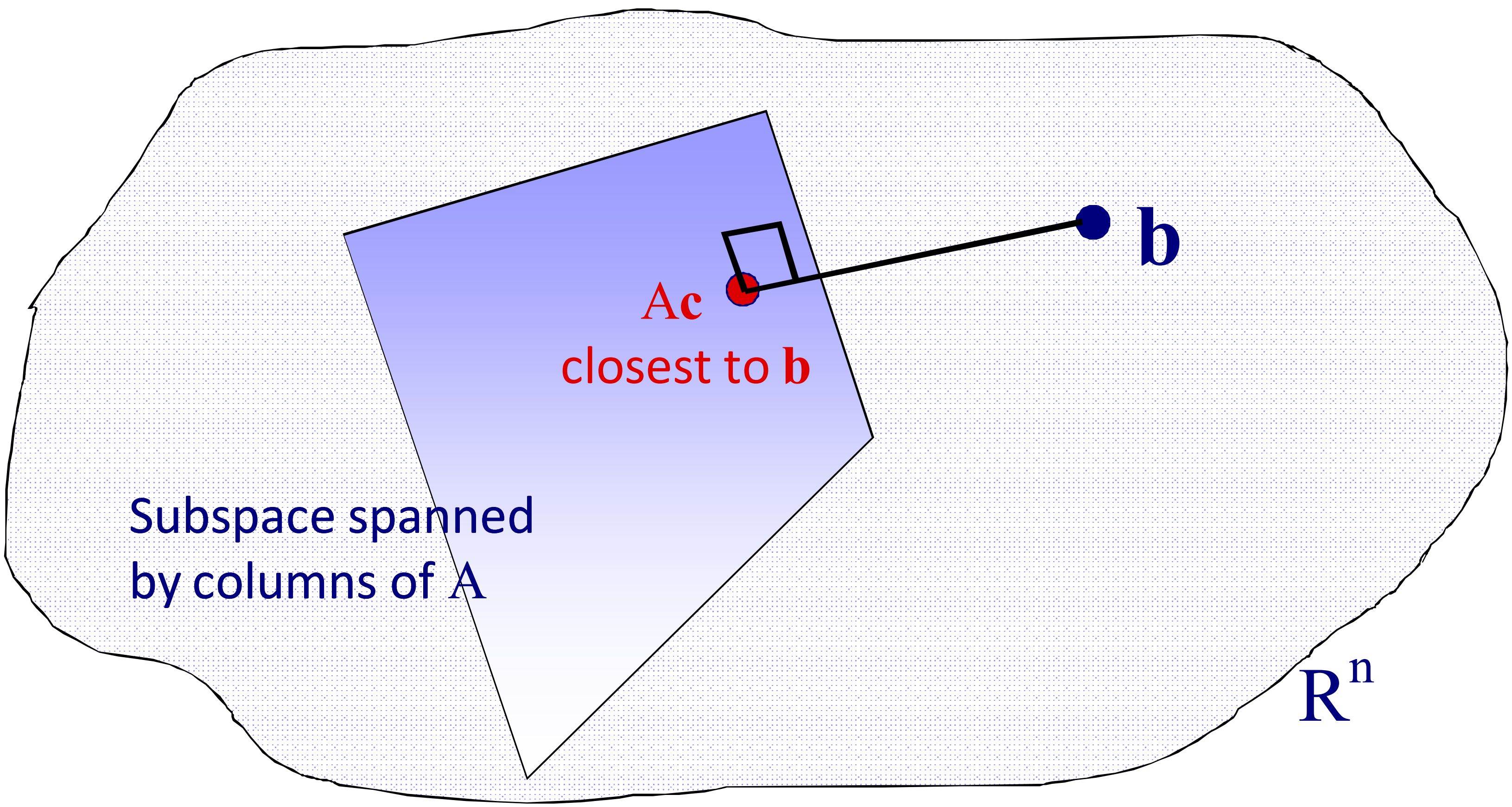
- $\mathbf{c} \in R^k$
- $A\mathbf{c} \in R^n$
- As we vary $\mathbf{c}$, $A\mathbf{c}$ varies over the linear subspace of R^n spanned by the columns of A:

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

This is also known as the column space of A

Finding the LS solution

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



Finding the LS solution

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

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

$$\Updownarrow$$

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

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

$$\Updownarrow$$

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

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

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:

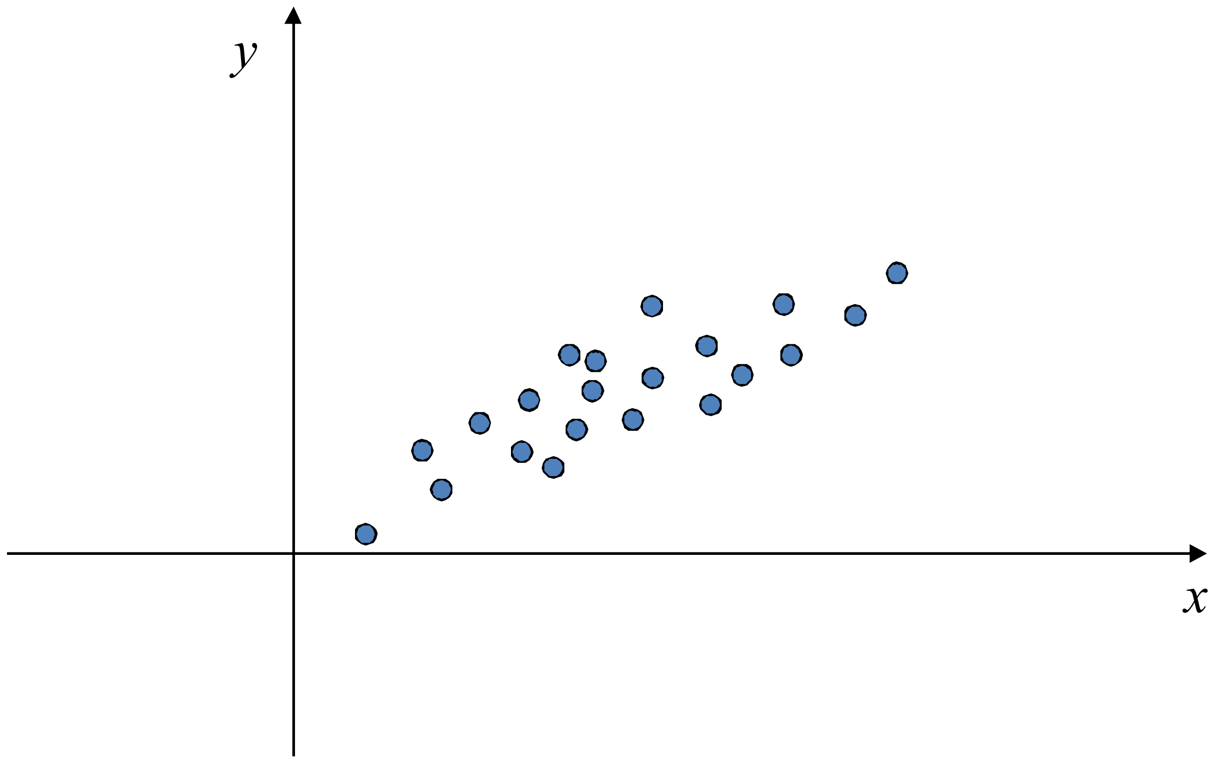
$$\min\,(\mathrm{A}\mathbf{c} - \mathbf{b})^{\mathrm{T}}\, \mathrm{W}^{\mathrm{T}}\mathrm{W}\,(\mathrm{A}\mathbf{c} - \mathbf{b}) \quad \text{where } \mathrm{W} = (w_i)_{ii}$$

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

Principal Component Analysis

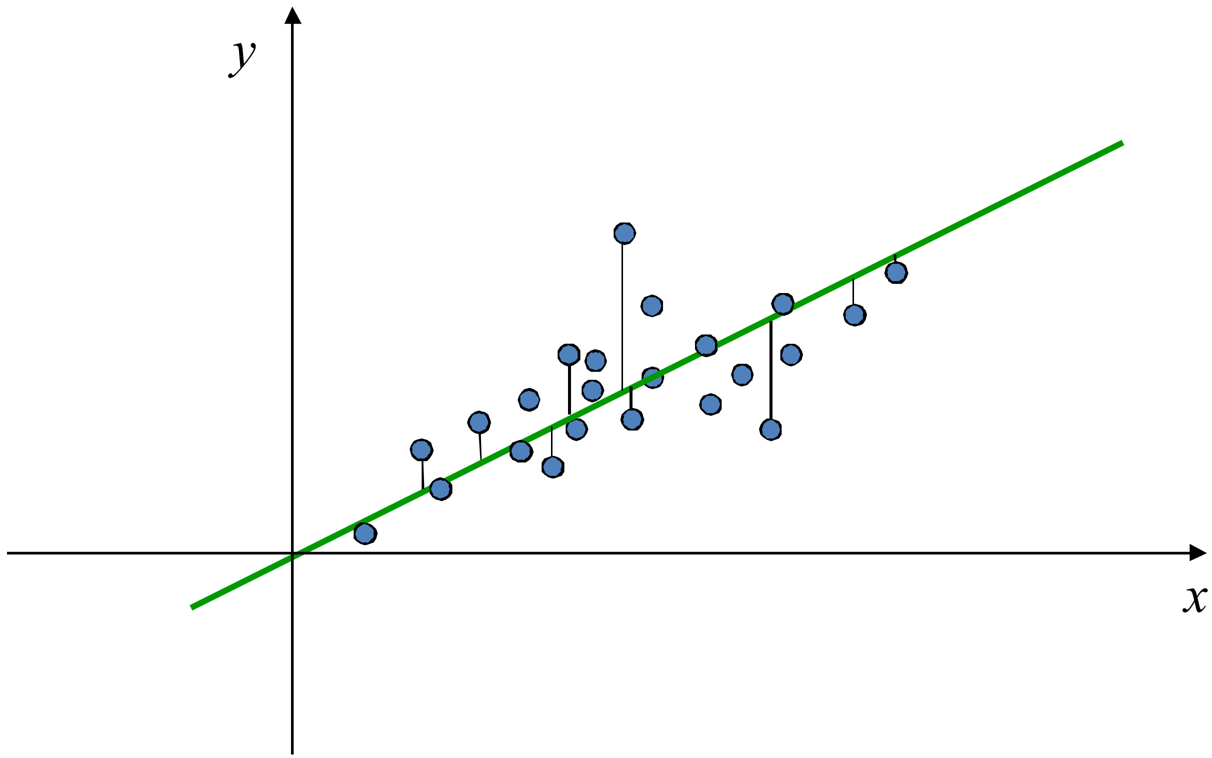
But first, reminder about
eigenvectors and eigenvalues

Motivation



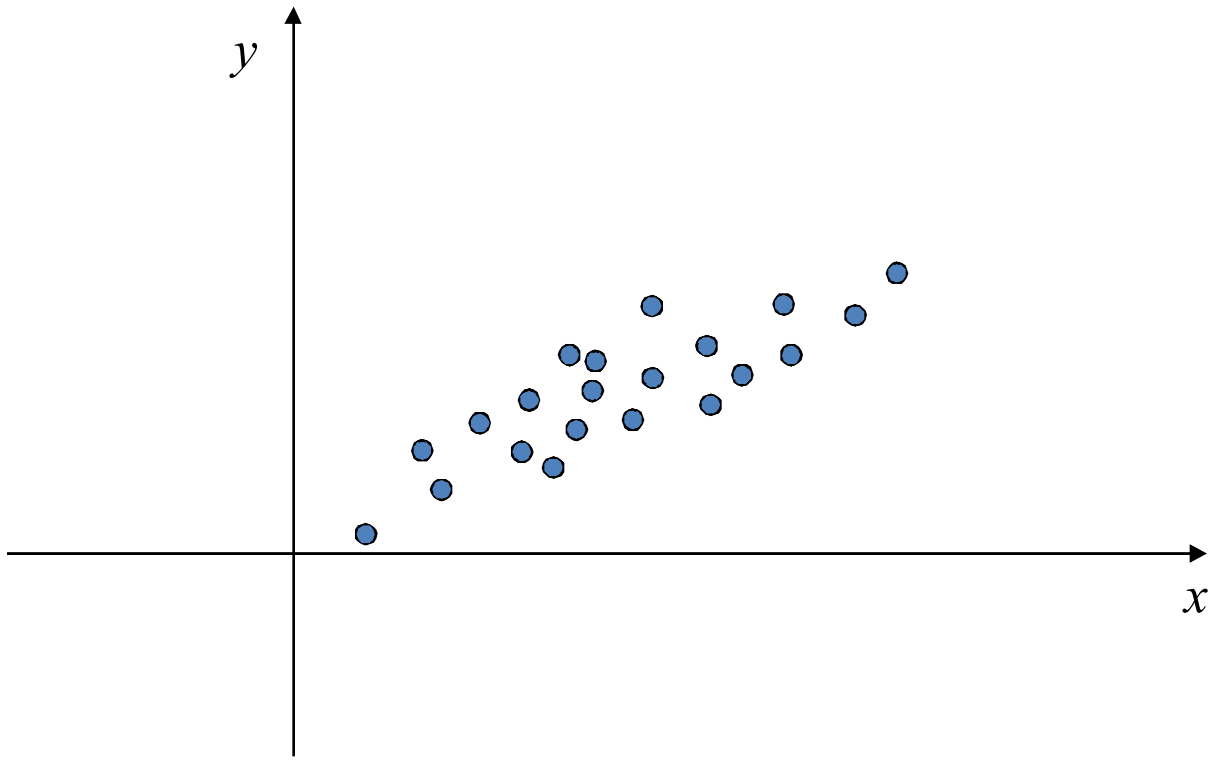
- Given a set of points, find the best line that approximates them

Motivation



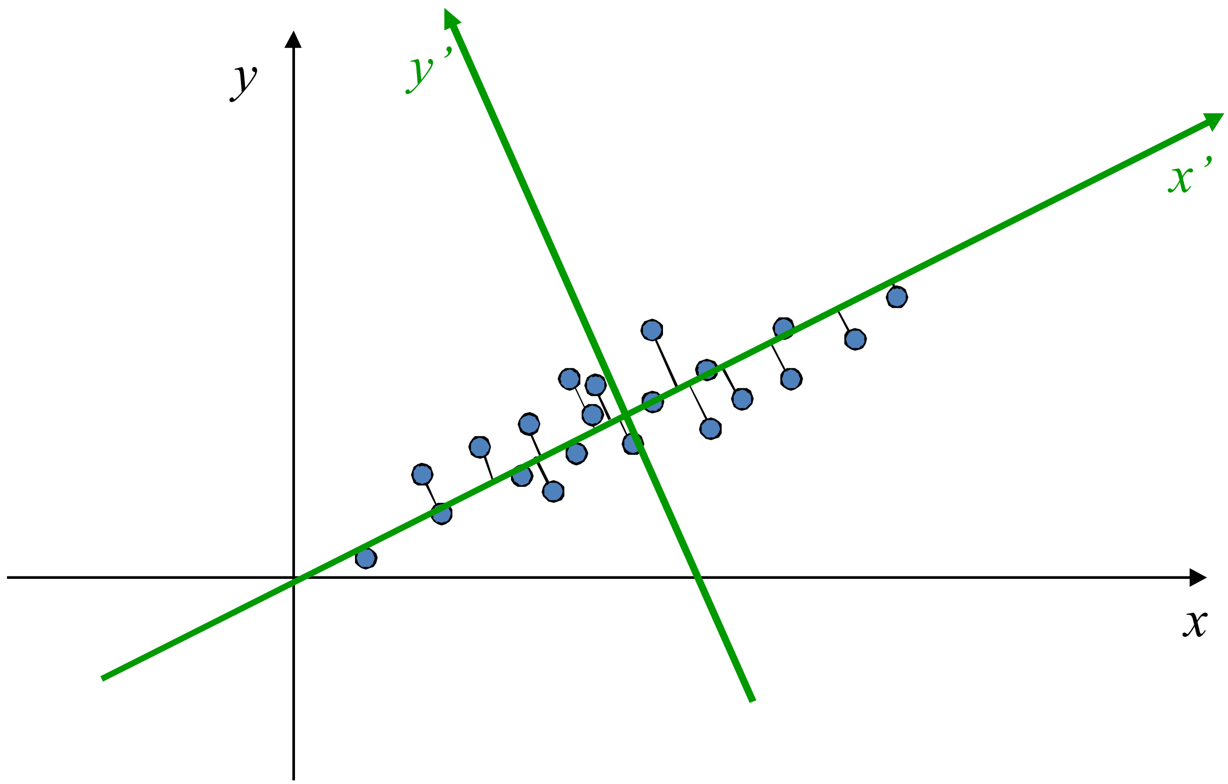
- We just saw how to fit a parametric line $y = ax + b$, but this does not work for vertical lines

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

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)

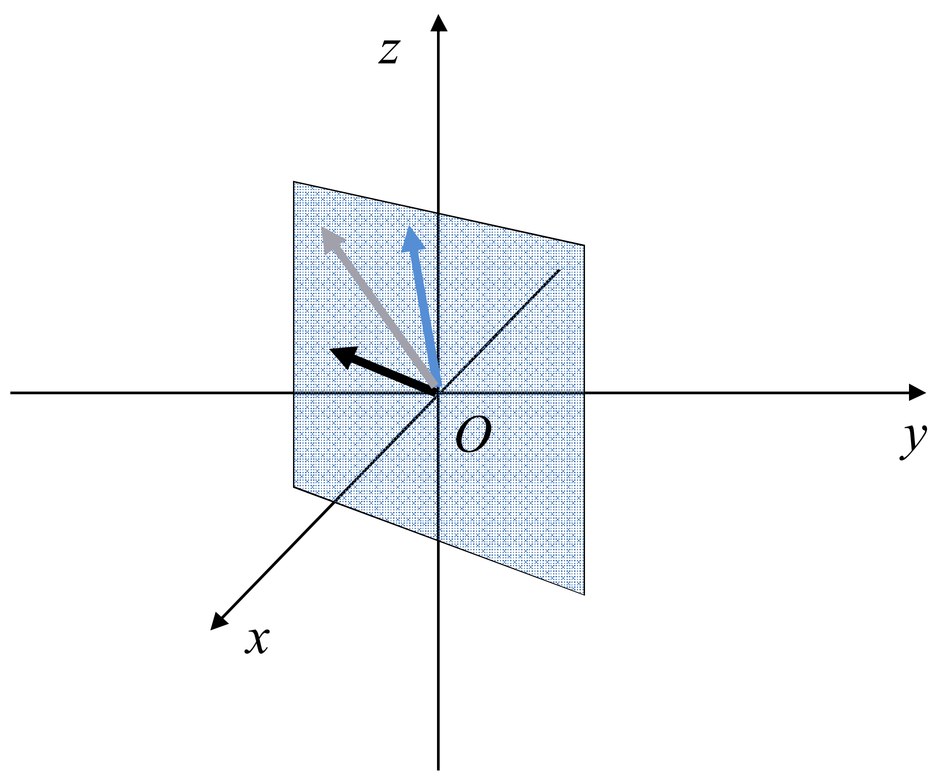
- Formal definition includes axioms about associativity and distributivity of the + and · operators.

- $0 \in V$ always!

Linear algebra recap

Vector space – example

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



Linear algebra recap

Linear independence

- The vectors $\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_\mathrm{k}\}$ are a linearly independent set if:

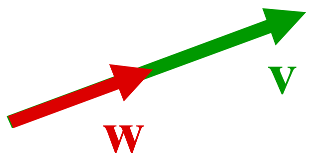
$$\alpha_1\mathbf{v}_1 + \alpha_2\mathbf{v}_2 + \ldots + \alpha_k\mathbf{v}_\mathrm{k} = 0 \quad \Leftrightarrow \quad \alpha_i = 0 \;\forall\, i$$

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

Linear algebra recap

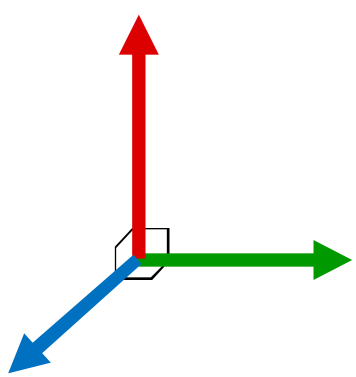
Linear independence

- Parallel vectors are always dependent:



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

- Orthogonal vectors are always independent.



Linear algebra recap

Basis of a vector space V

- $\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_\mathrm{n}\}$ are linearly independent
- $\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_\mathrm{n}\}$ span the whole vector space V:

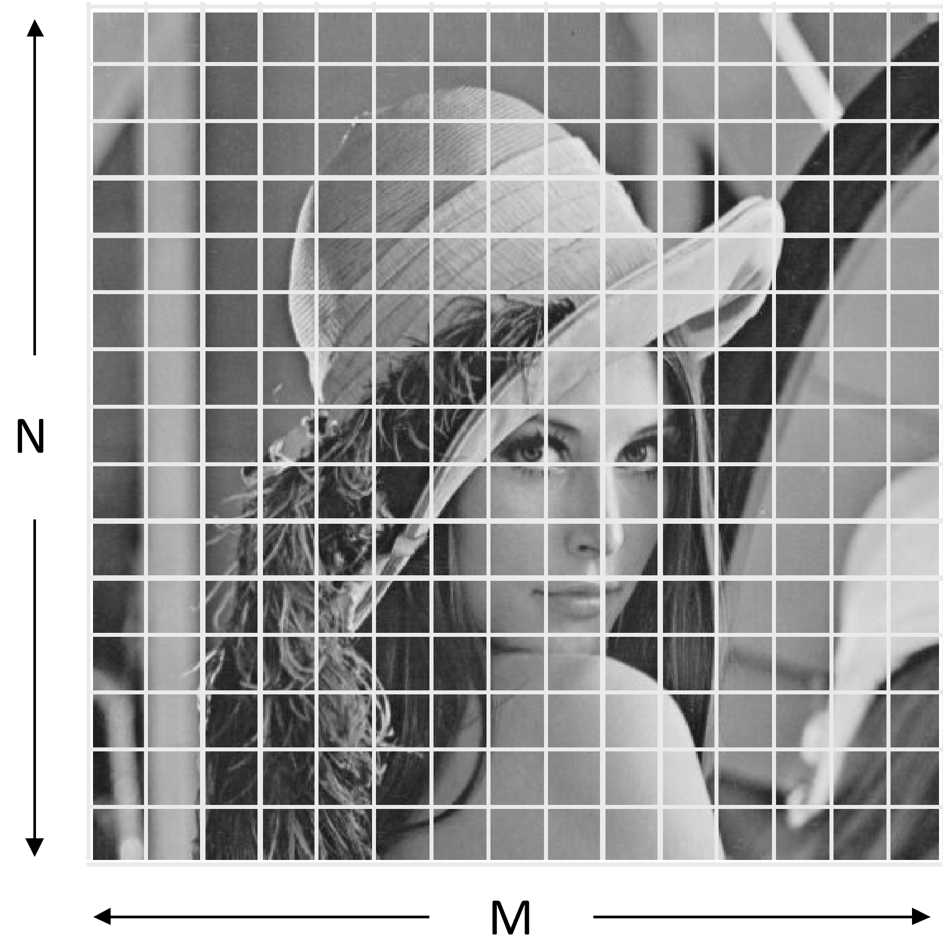
$$V = \{\alpha_1\mathbf{v}_1 + \alpha_2\mathbf{v}_2 + \ldots + \alpha_n\mathbf{v}_\mathrm{n} \mid \alpha_i \text{ is scalar}\}$$

- 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

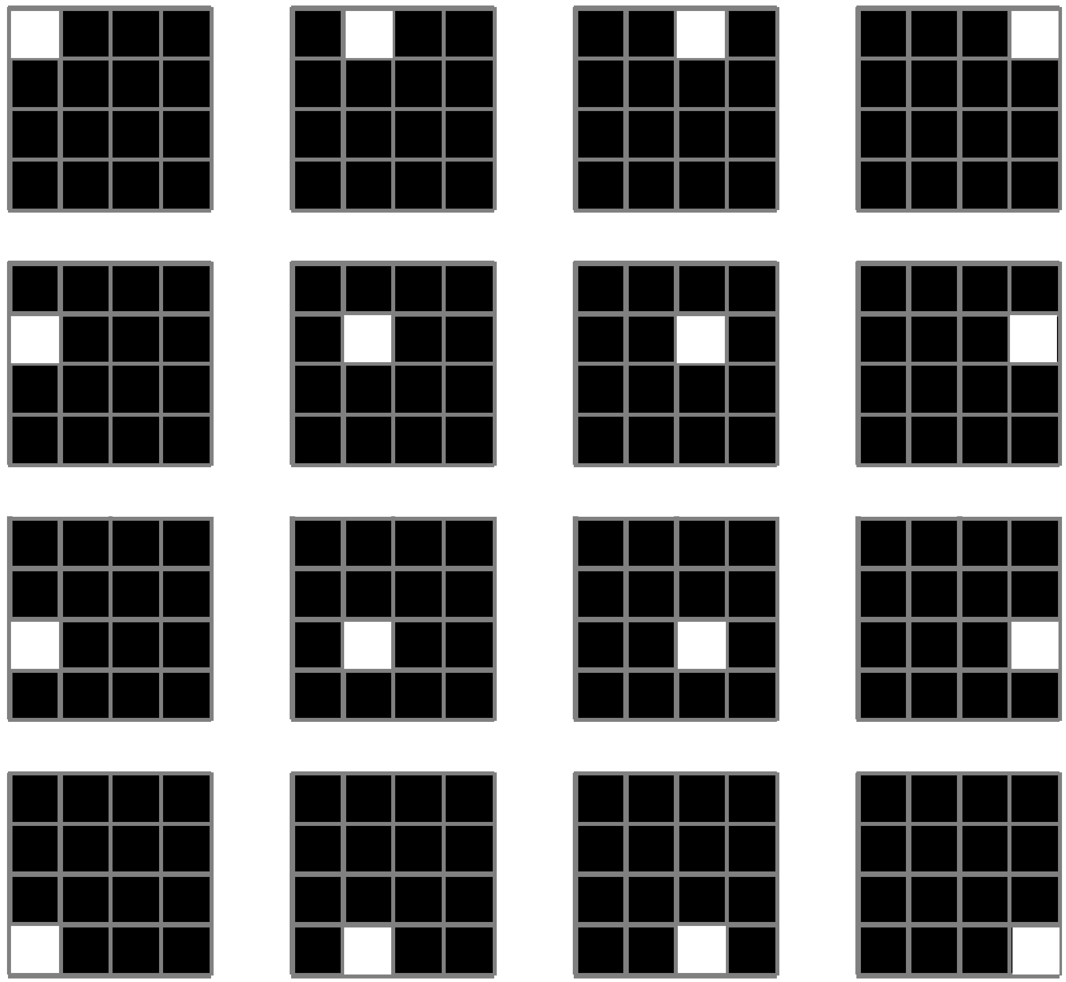
Basis example

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



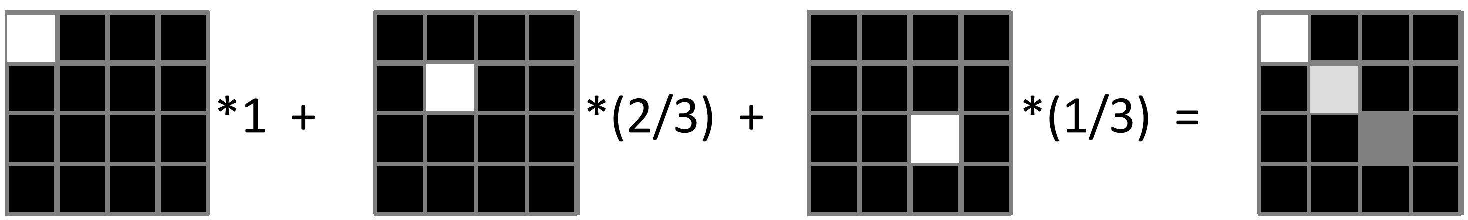
Linear algebra recap

The "standard" basis (4×4)



Linear algebra recap

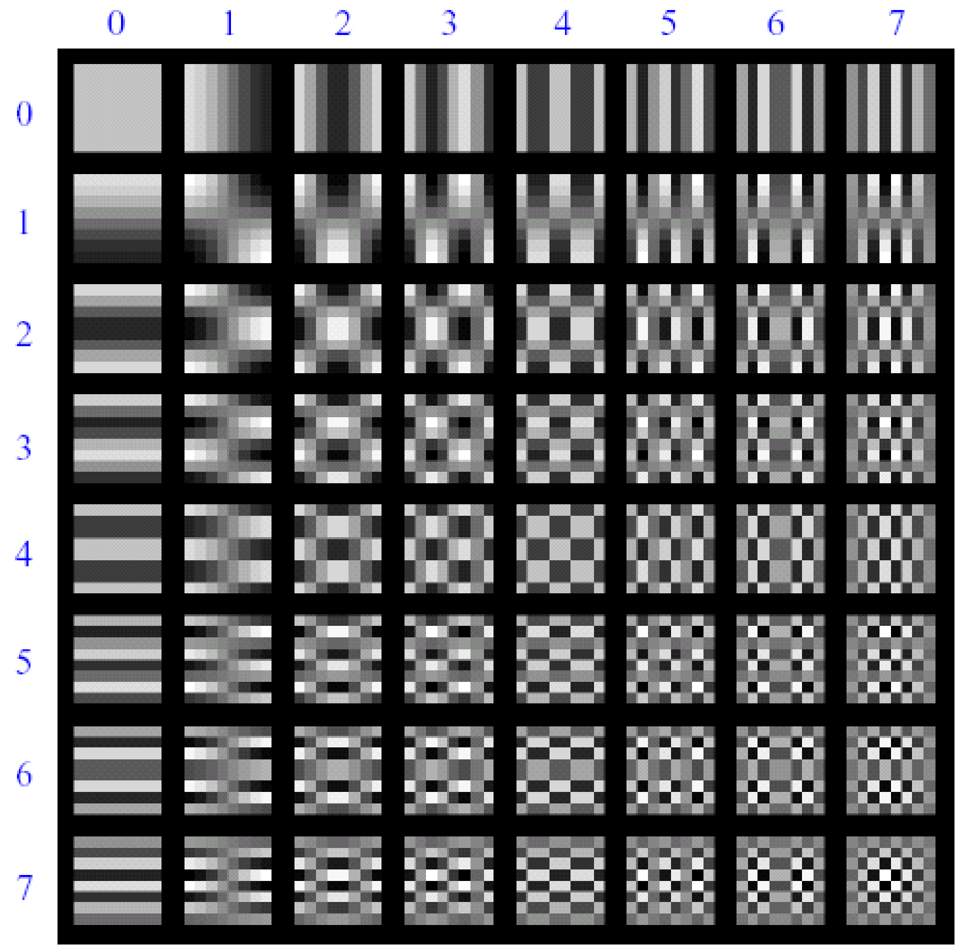
The “standard” basis (4×4) – linear combinations



Linear algebra recap

Discrete cosine basis

- Used for JPEG encoding

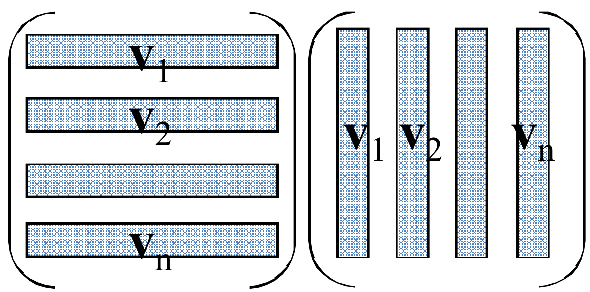


Linear algebra recap

Orthogonal matrices (orthonormal basis)

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

$$I = A^T A =$$



$$= \begin{pmatrix} \mathbf{v}_i^T \mathbf{v}_j \end{pmatrix} = \begin{pmatrix} \delta_{ij} \end{pmatrix}$$

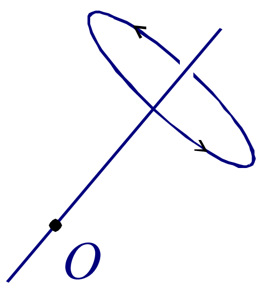
$$\Rightarrow \langle \mathbf{v}_i, \mathbf{v}_i \rangle = 1 \quad \Rightarrow \quad \|\mathbf{v}_i\| = 1; \quad \langle \mathbf{v}_i, \mathbf{v}_j \rangle = 0$$

Linear algebra recap

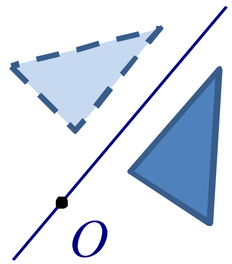
Orthogonal transformations

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

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



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



Linear algebra recap

Eigenvectors and eigenvalues

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

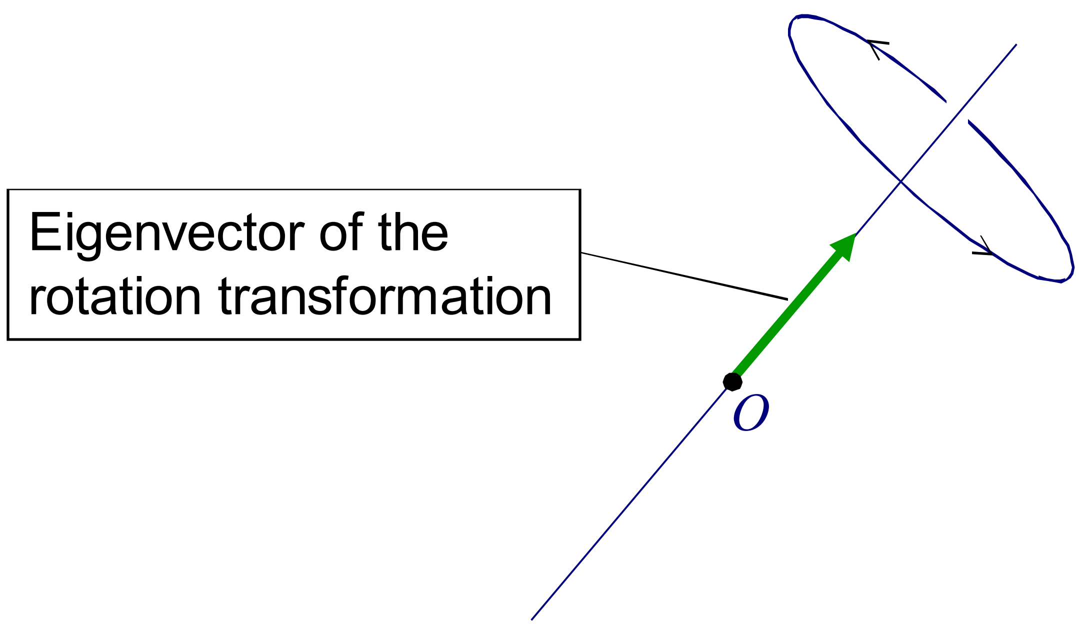
$$\mathrm{A}\mathbf{v} = \lambda\mathbf{v}$$

- $A\mathbf{v} = \lambda\mathbf{v} \Rightarrow A(\alpha\mathbf{v}) = \lambda(\alpha\,\mathbf{v}) \Rightarrow \alpha\,\mathbf{v}$ 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 λ 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:

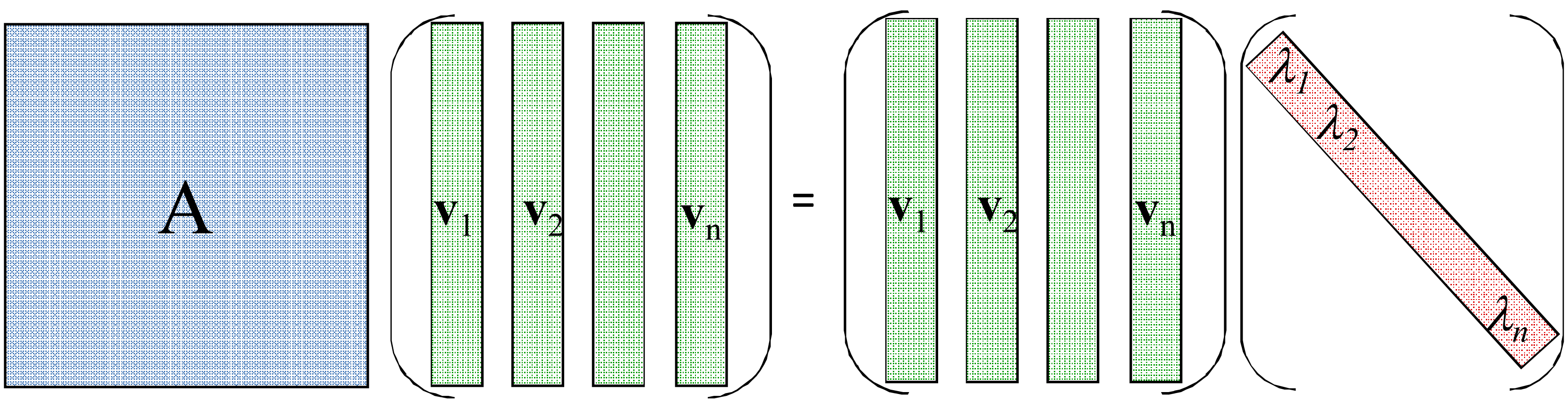


Linear algebra recap

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: $A\mathrm{V} = \mathrm{VD}$

$$
\begin{aligned}
A\mathbf{v}_1 &= \lambda_1 \mathbf{v}_1 \\
A\mathbf{v}_2 &= \lambda_2 \mathbf{v}_2 \\
&\vdots \\
A\mathbf{v}_n &= \lambda_n \mathbf{v}_n
\end{aligned}
$$

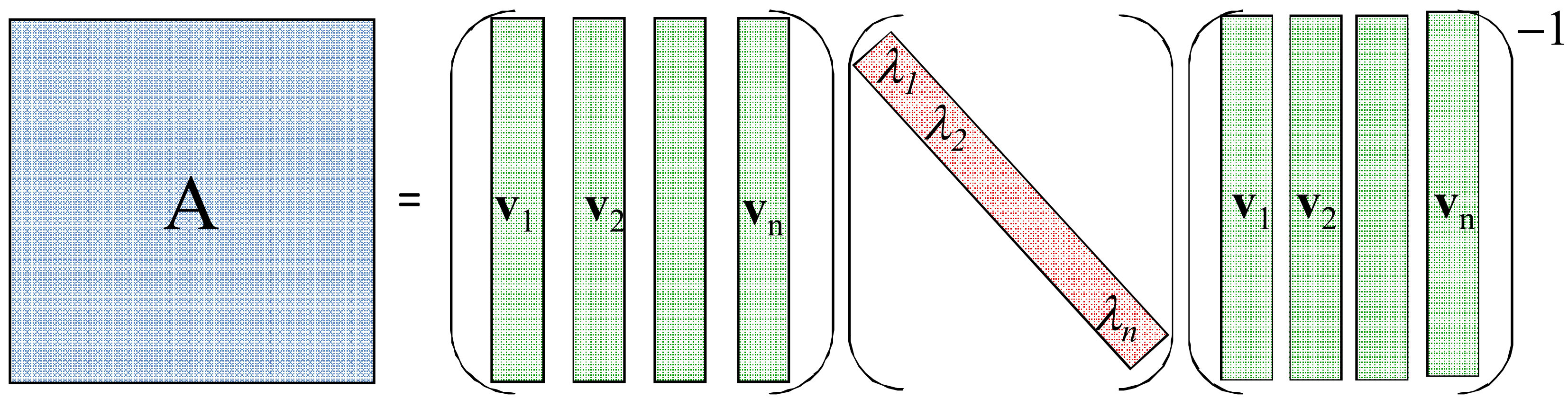


Linear algebra recap

Spectrum and diagonalization

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

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

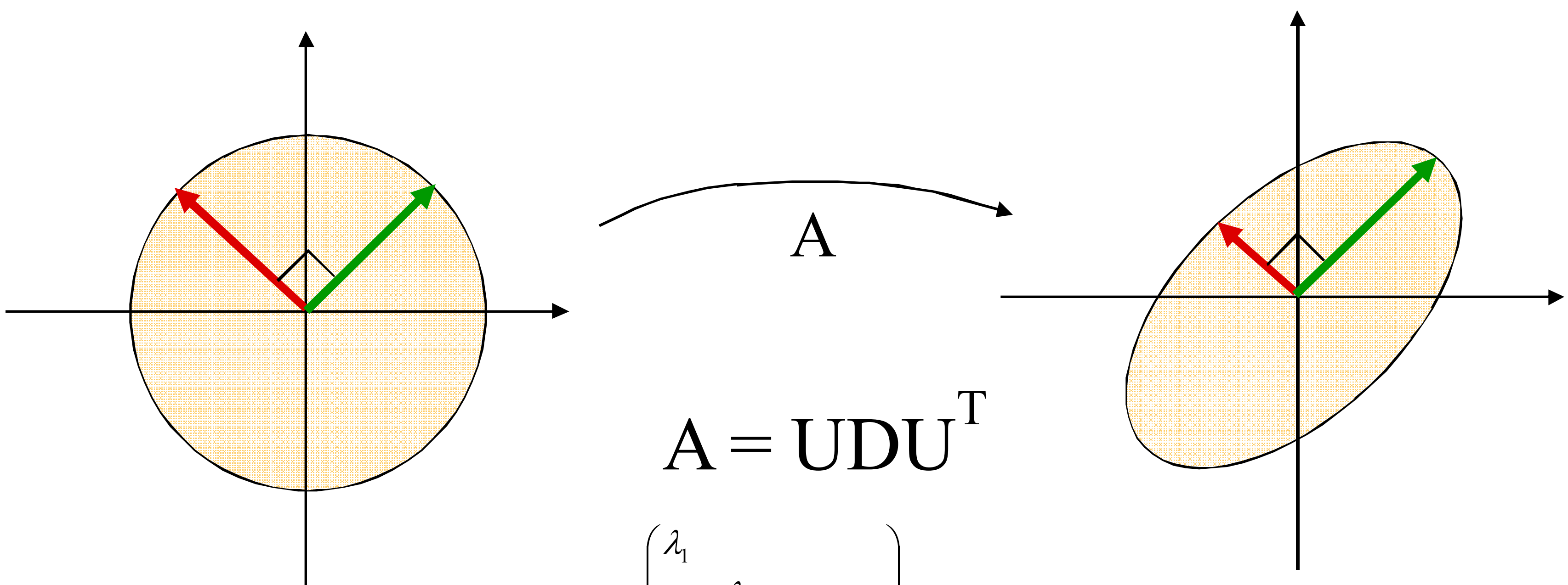


$$A = \mathrm{VDV}^{-1}$$

Linear algebra recap

Symmetric matrices

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



$$A = UDU^T$$

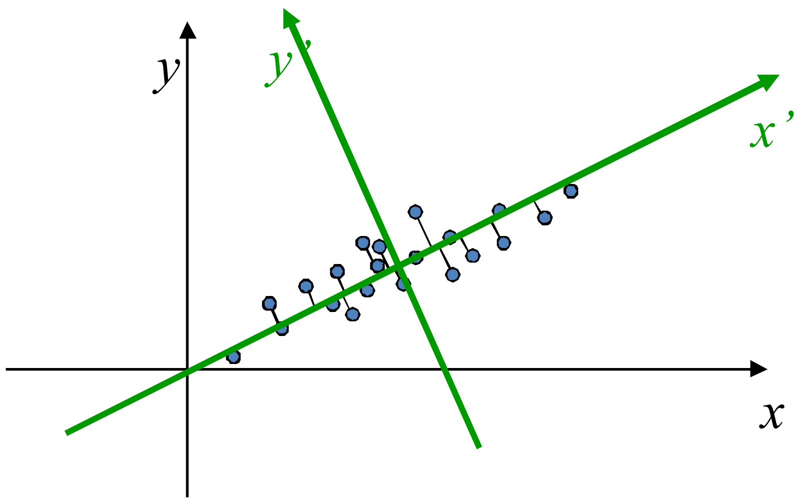
$$D = \begin{pmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_n \end{pmatrix}$$

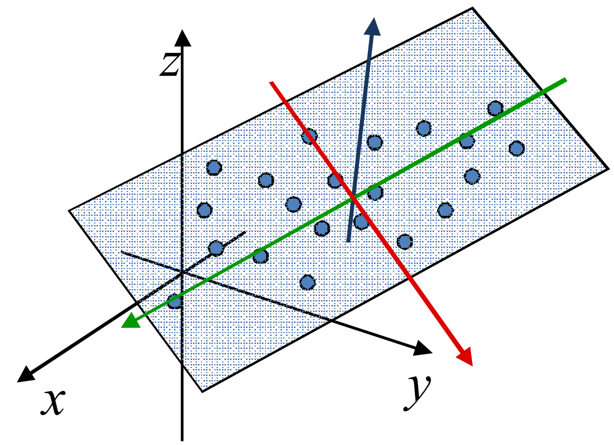
$$A\mathbf{u}_i = \lambda_i \mathbf{u}_i$$

Principal Component Analysis

Basic idea

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



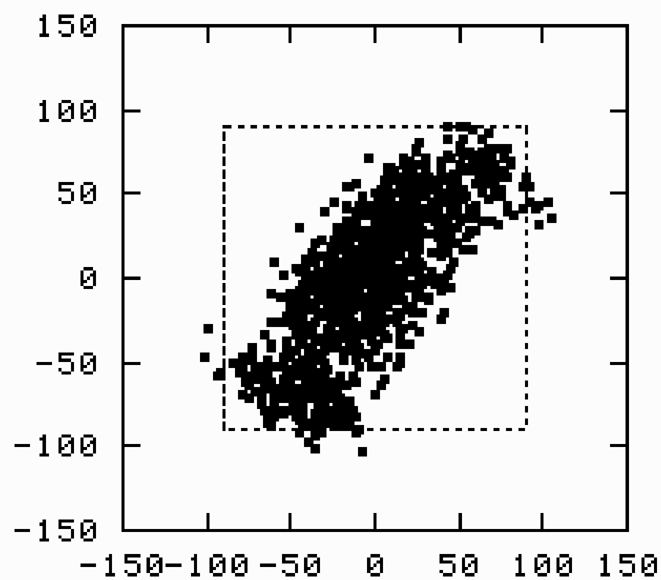


- PCA finds a best approximating line/plane/axes… (in terms of $\Sigma_{distances^2}$)

Principal Component Analysis

Basic idea

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



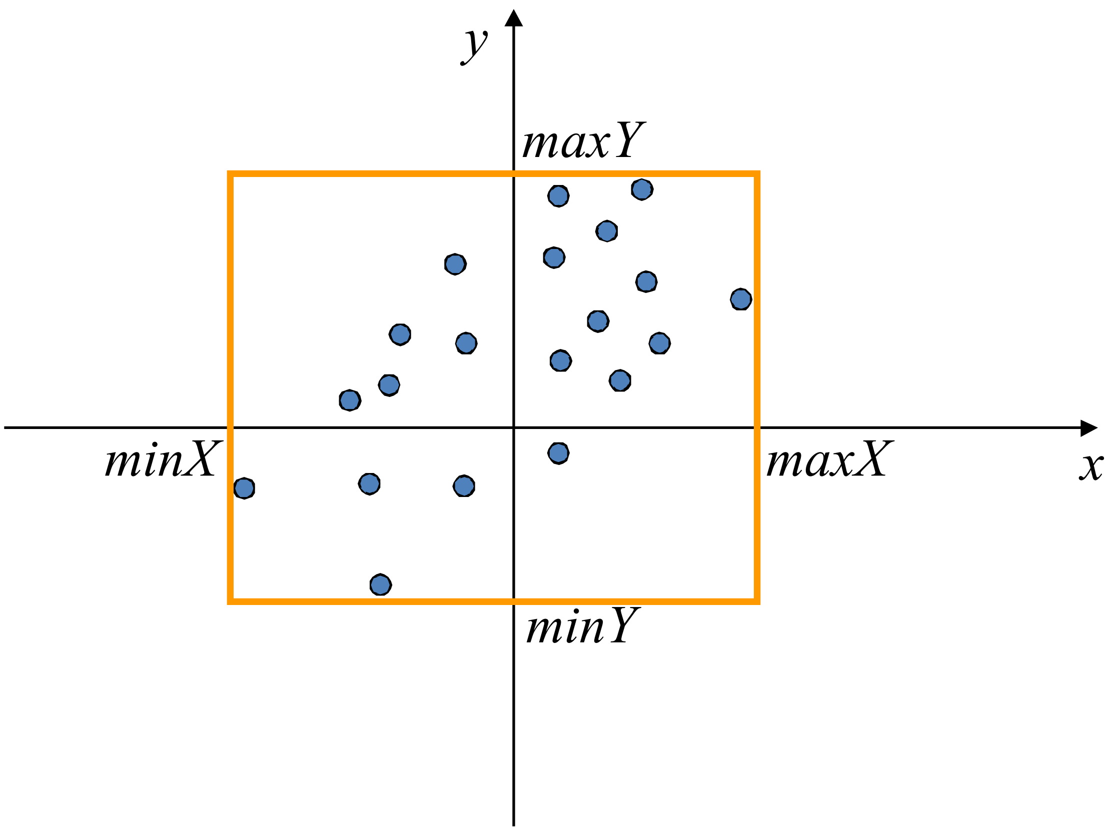
3D point set in standard basis

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

Principal Component Analysis

Applications

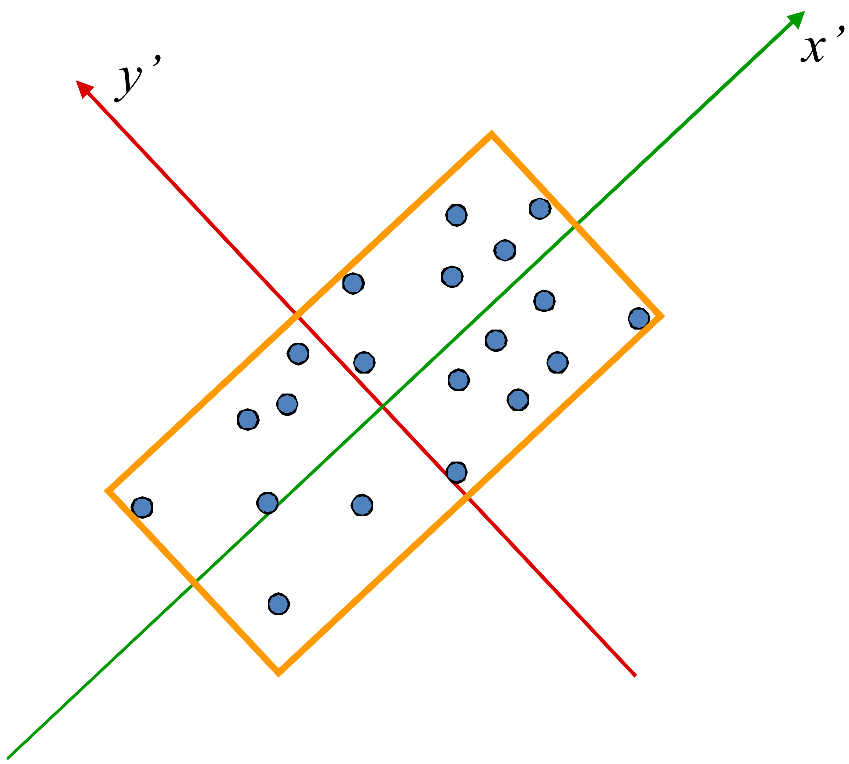
- An axis-aligned bounding box: agrees with the standard axes



Principal Component Analysis

Application: oriented bounding box

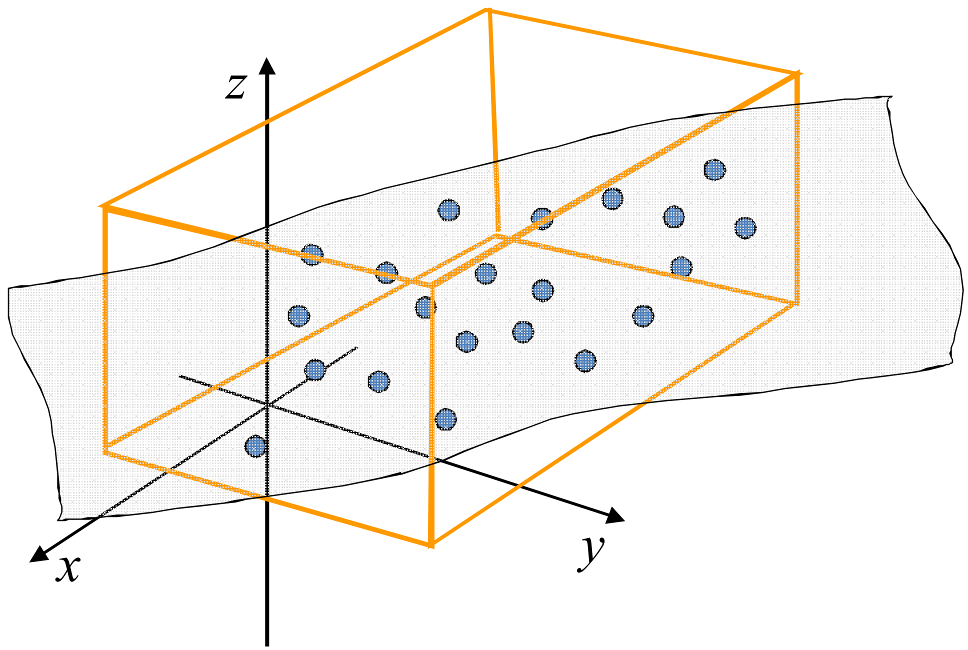
- Tighter fit



Principal Component Analysis

Application: oriented bounding box

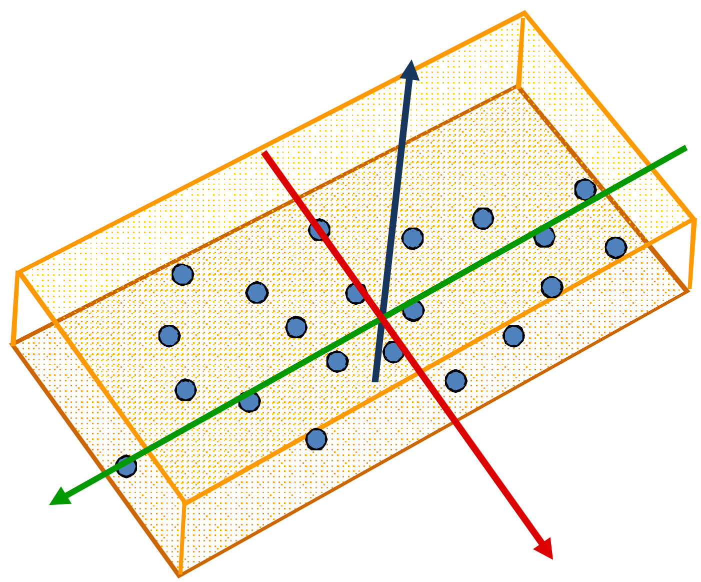
- Axis aligned bounding box



Principal Component Analysis

Application: oriented bounding box

- Oriented bounding box by PCA

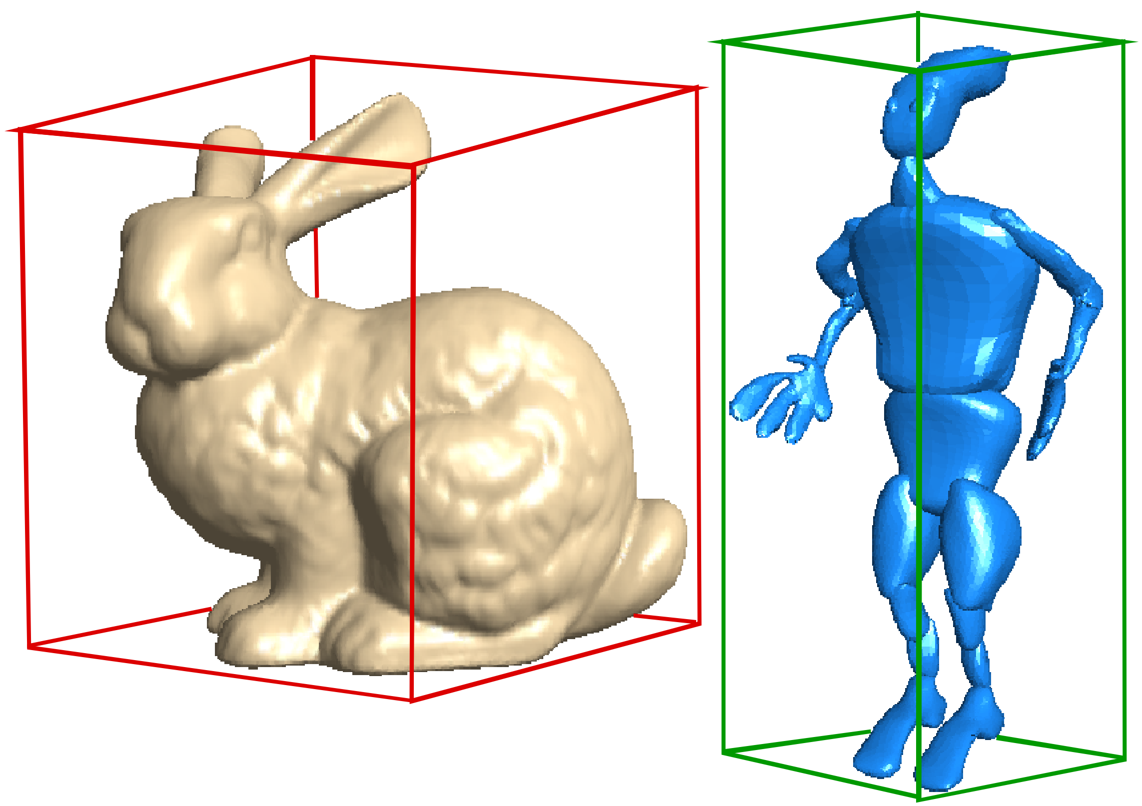


Principal Component Analysis

Application: oriented bounding box

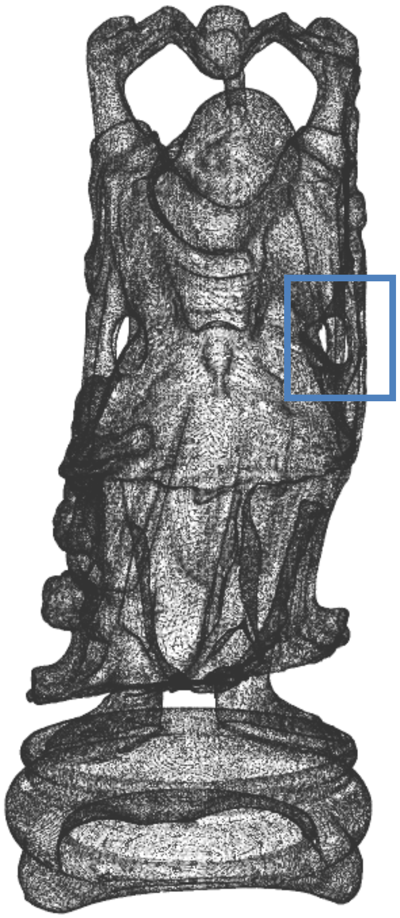
- Serve as very simple "approximation" of the object
- Fast collision detection, visibility queries
- Whenever we need to know the dimensions (size) of the object

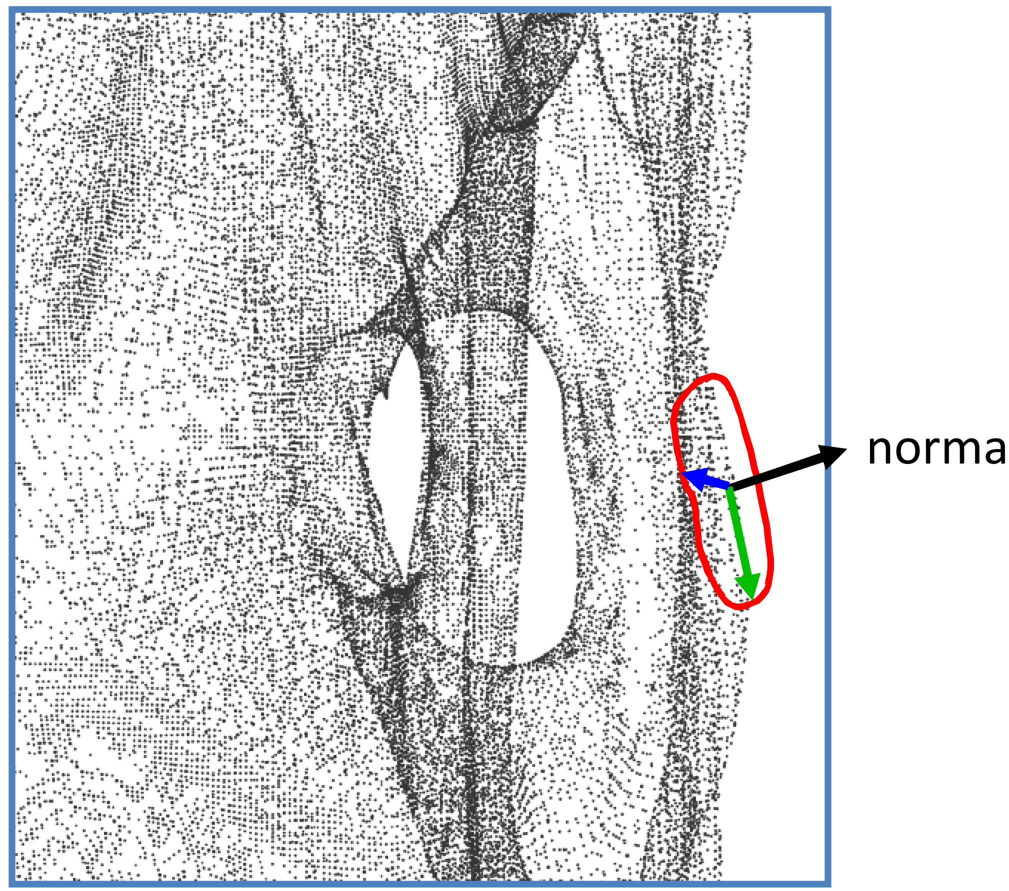
- The models consist of thousands of polygons
- To quickly test that they don't intersect, the bounding boxes are tested
- Sometimes a hierarchy of BB's is used
- The tighter the BB – the less "false alarms" we have



Principal Component Analysis

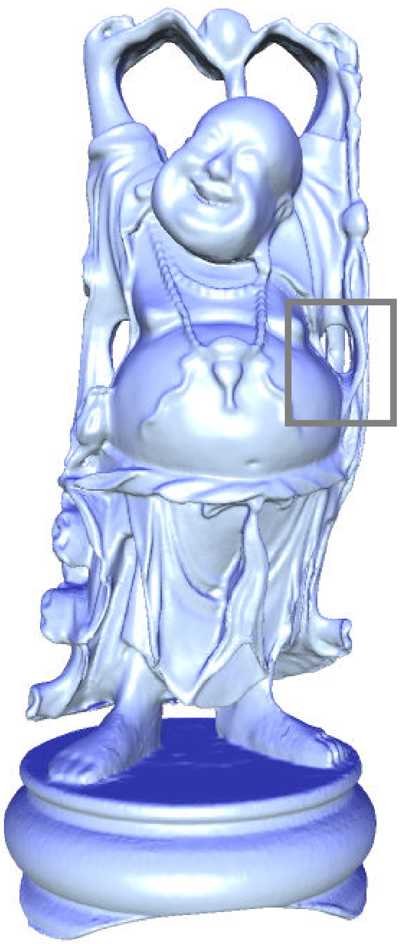
Application: local frame fitting

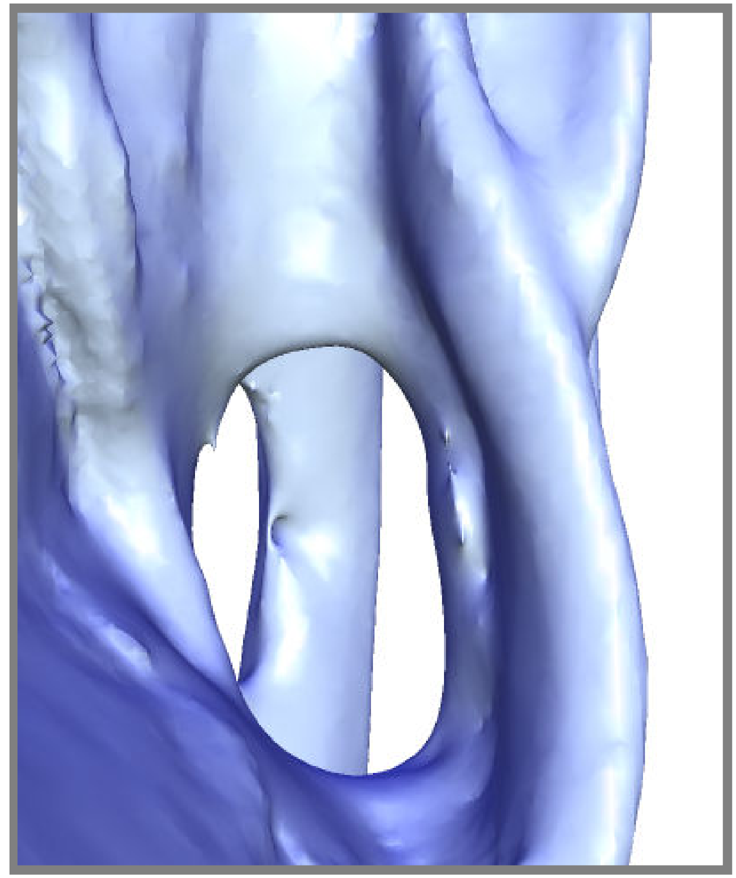




Principal Component Analysis

Application: estimate normals

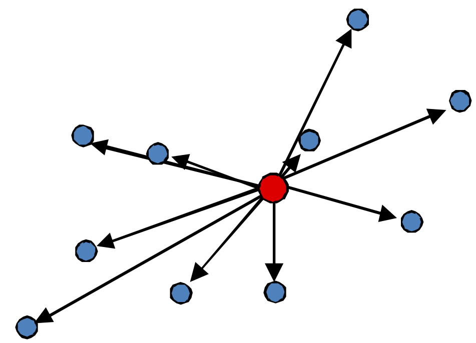




Notations

- Denote our data points by $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n \in R^d$
- Center of mass:

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

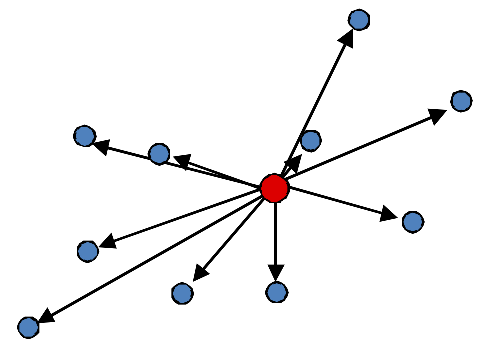


- Vectors from the centroid:

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

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:

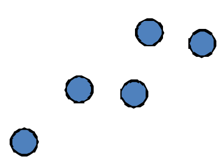
$$\mathbf{m} = \underset{\mathbf{x}}{\operatorname{argmin}} \sum_{i=1}^{n} \left\| \mathbf{x}_i - \mathbf{x} \right\|^2$$

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

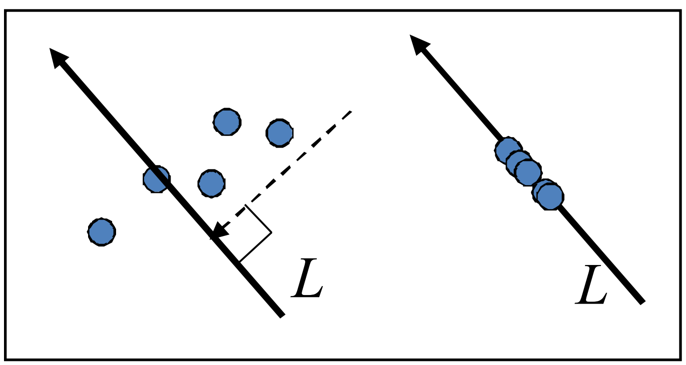
Variance of projected points

- 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:

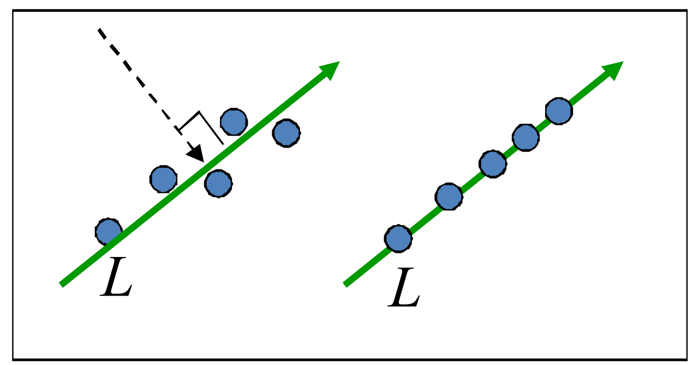
$$\mathrm{var}(L) = \tfrac{1}{n}\sum_{i=1}^{n} \| \mathbf{x}'_i - \mathbf{m} \|^2$$



Original set



Small variance

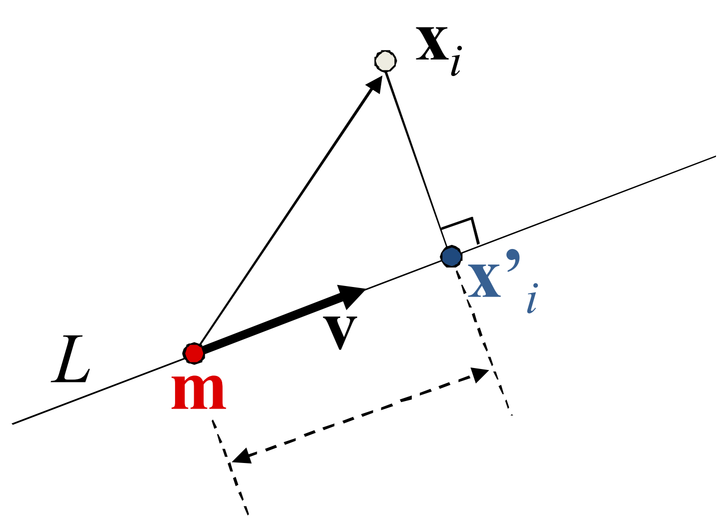


Large variance

Variance of projected points

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

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



Variance of projected points

- So,

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

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

$$\boxed{\mathrm{S} = (1/n)\ \mathrm{Y}\ \mathrm{Y}^{\mathrm{T}}}$$ 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: $\mathrm{var}(L) = \mathbf{v}^{\mathrm{T}} \mathrm{S} \mathbf{v}$
- Theorem:

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

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

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

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

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

- This is the definition of an eigenvector of S

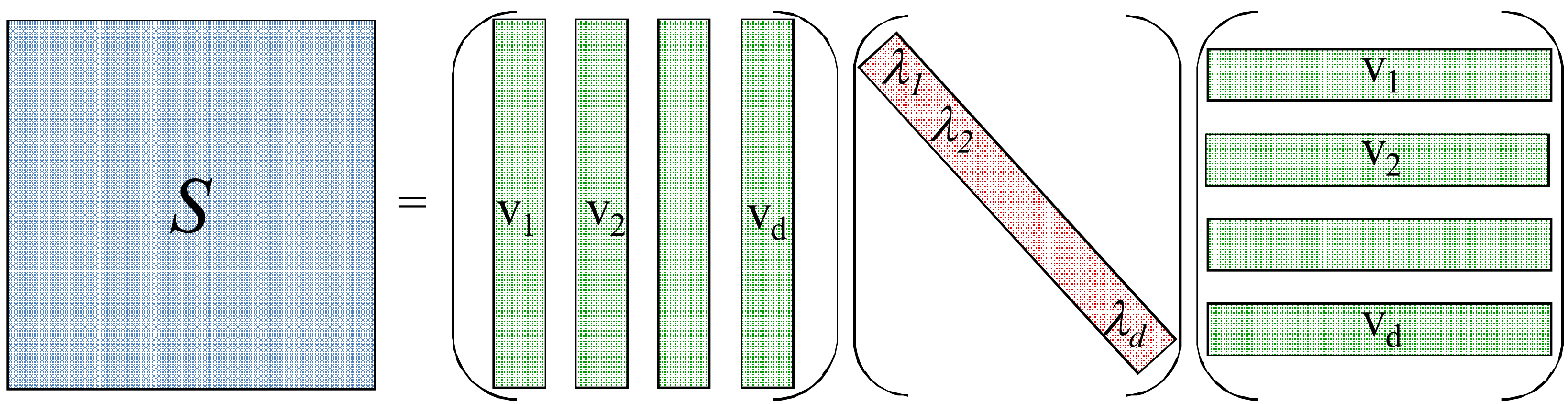
Summary so far

- We take the centered data vectors $\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_n \in R^d$
- Construct the scatter matrix $S = YY^T$
- 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: $\boldsymbol{S} = V\Lambda V^{\mathrm{T}}$



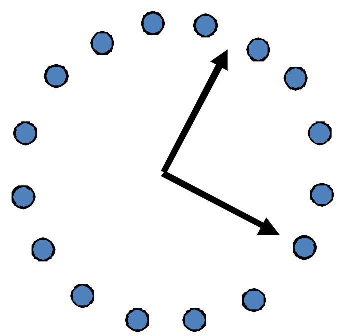
The eigenvectors form orthogonal basis

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. Think why…

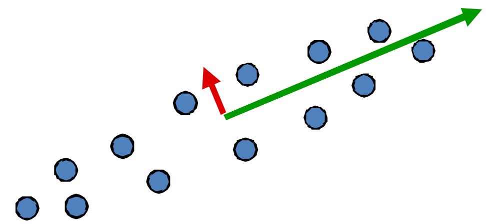
Principal components



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

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

- Any vector is an eigenvector



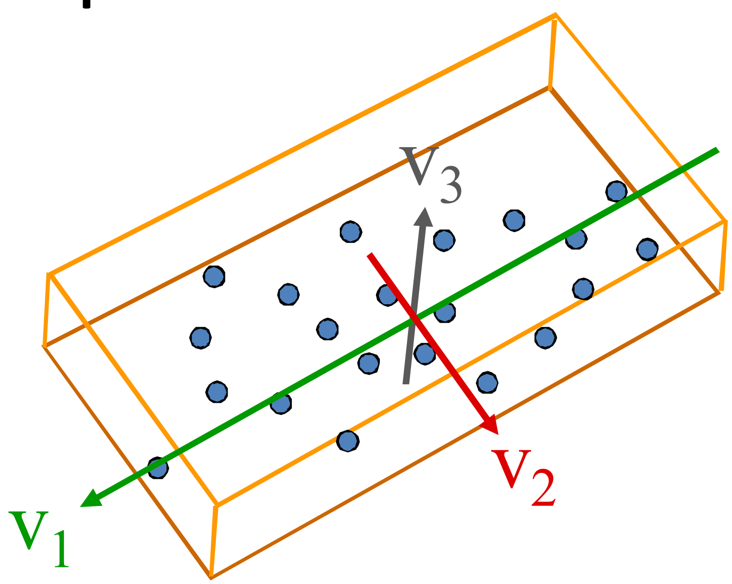
- There's a clear preferable direction
- ***S*** looks like this:

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

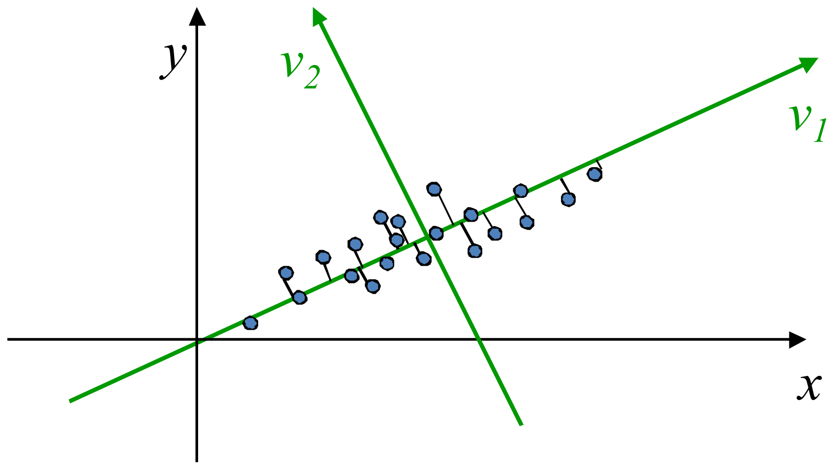
- μ is close to zero, much smaller than λ

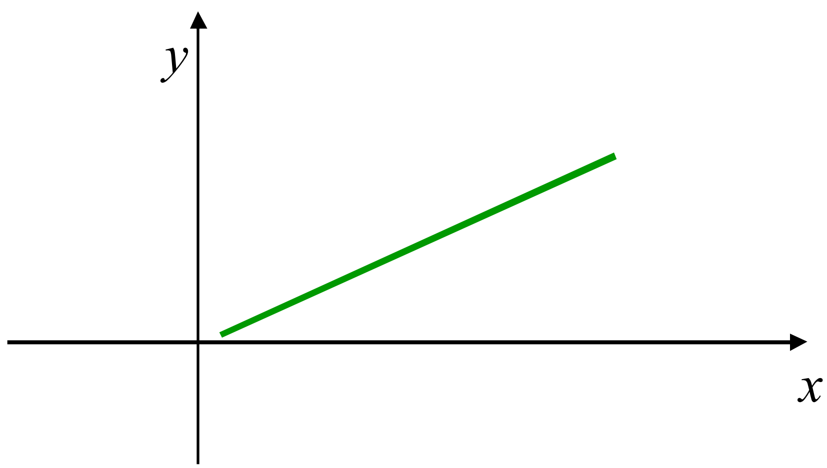
How to use what we got

- 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 $\mathbf{m}$.

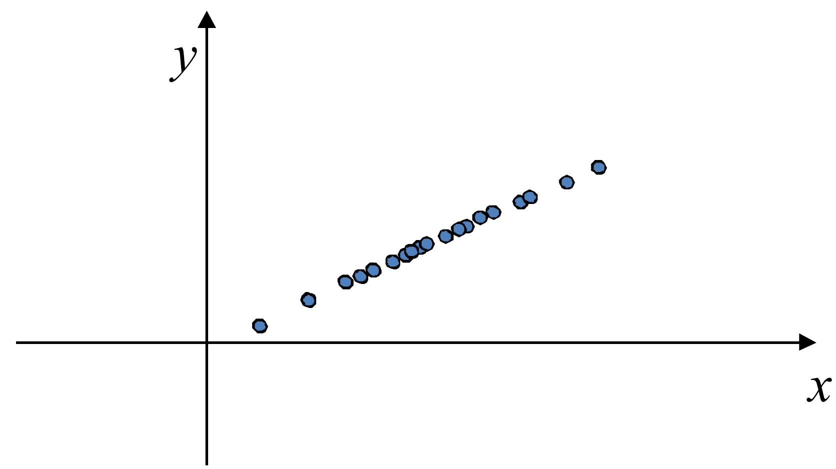


For approximation





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:

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

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

Proof:

$$S = V\Lambda V^T \quad \Rightarrow \quad < S\mathbf{v}, \mathbf{v} > = \mathbf{v}^T S\mathbf{v} = \mathbf{v}^T V\Lambda V^T \mathbf{v} =$$

$$= (V^T\mathbf{v})^T \Lambda (V^T\mathbf{v}) = \mathbf{v}^T \Lambda \mathbf{v} = < \Lambda\mathbf{v}, \mathbf{v} >$$

$$\boxed{< S\mathbf{v}, \mathbf{v} > = \lambda_1 \mathbf{u_1}^2 + \lambda_2 \mathbf{u_2}^2 + \ldots + \lambda_d \mathbf{u_d}^2}$$

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