Lecture 1

Representation Learning:

ML → tell the computer "how to learn" from data

↓

b/c real world data has "structure".

ex) Lease square regression

↓

need to handcrafted feature representation.

for example: Laptop: cpu, ram, weight ...

Problem: Image feature representation is in high-dimensional space

(256 × 256 × 3) ...

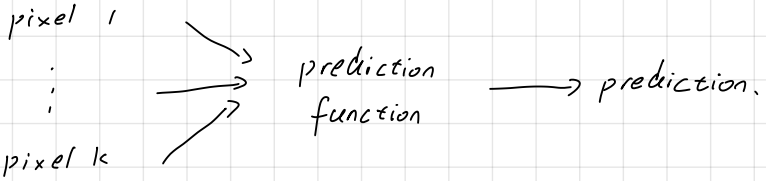
regression model: $w_0 +$

$w_1 \times (\text{pixel } 1) +$

$w_2 \times (\text{pixel } 2) +$

... +

$w_k \times (\text{pixel } k)$.



Doesn't consider the proximity of pixels

(no location information given to the model).

↓

information is contained in the images, but not in individual pixels.

(pattern is not captured!)

↓

Solution: a feature extractor to detect pattern.

pixels → feature extractor → features → prediction function.

building good feature is hard

↓

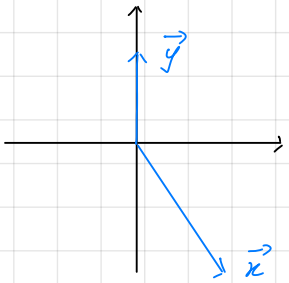
$\Rightarrow$ representation learning.

Linear Algebra

Vector $\vec{x} \in \mathbb{R}^d$ as a coordinate vector $\vec{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{pmatrix}$

ex) $\vec{x} = \begin{pmatrix} 2 \\ -3 \end{pmatrix} \in \mathbb{R}^2$

$\vec{y} = \begin{pmatrix} 0 \\ 2 \end{pmatrix}$



Writting vector in coordinate form requires choosing a basis

⇓

"Standard Basis": $\hat{e}^{(1)}, \cdots, \hat{e}^{(d)}$.

$\hat{e}^{(1)} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$, $\hat{e}^{(2)} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ ← 2nd coord. ← 1st coord.

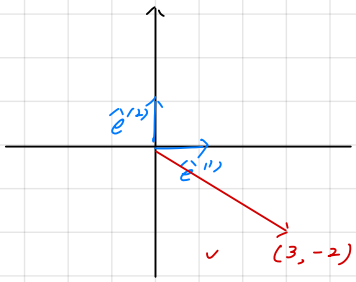
so, $\vec{x} = (x_1 \cdots x_d)^T$

$= x_1 \hat{e}^{(1)} + x_2 \hat{e}^{(2)} + \cdots + x_d \hat{e}^{(d)}$.

↑ amount in 1st coord.

ex). $\vec{x} = (3, -2)^T = x_1 \begin{bmatrix} 1 \\ 0 \end{bmatrix} + x_2 \begin{bmatrix} 0 \\ 1 \end{bmatrix}$.

$x_1 = 3$, $x_2 = -2$



In coordinate form:

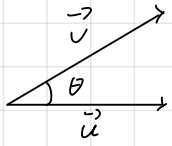
$\hat{e}^{(i)} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \\ 0 \end{pmatrix}$ ← ith coordinate. (Standard basis vector).

where the 1 appears in the i-th place

Dot product:

$\vec{u} \cdot \vec{v} = \|\vec{u}\| \|\vec{v}\| \cos\theta$.

where θ is angel between $\vec{u}$ and $\vec{v}$.



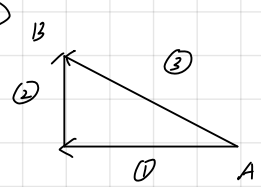
- $\vec{u} \cdot \vec{v} = 0 \iff \vec{u}$ and $\vec{v}$ orthogonal.

$\vec{u} \cdot \vec{v} = 0$
$\rightarrow \|\vec{u}\| \|\vec{v}\| \cos\theta = 0$
$\rightarrow \cos\theta = 0$
$\rightarrow \theta = 90^\circ = \frac{\pi}{2}$.
$\Rightarrow$ orthogonal

$$\vec{u} \cdot \vec{v} = \vec{u}^T \vec{v}$$
$$= (u_1 \cdots u_d) \begin{pmatrix} v_1 \\ \vdots \\ v_d \end{pmatrix} \quad \text{(assumes standard basis)}.$$
$$= u_1 \cdot v_1 + \cdots + u_d \cdot v_d$$
$$= \sum_{i=1}^{d} u_i \cdot v_i$$

ex) $\begin{pmatrix} 3 \\ 7 \\ 2 \\ -5 \end{pmatrix} \cdot \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} = -5.$

Is there any other bases ?



standard basis : ① + ②

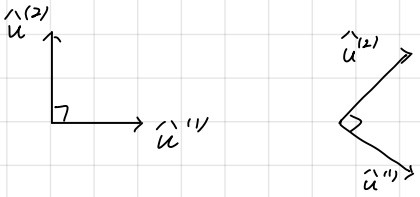
other basis : ③

Orthonormal Bases

A set of vectors $\hat{u}^{(1)} \cdots \hat{u}^{(d)}$ forms orthonormal basis U for $\mathbb{R}^d$ if

- they are mutually orthogonal : $\hat{u}^{(i)} \cdot \hat{u}^{(j)} = 0$
- they are unit vectors : $\|\hat{u}^{(i)}\| = 1$

ex)

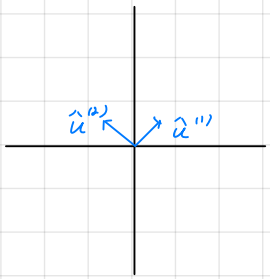


ex) $\hat{u}^{(1)} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix} \quad \hat{u}^{(2)} = \frac{1}{\sqrt{2}} \begin{pmatrix} -1 \\ 1 \end{pmatrix}.$

① $\hat{u}^{(1)} \cdot \hat{u}^{(2)} = \frac{1}{\sqrt{2}}(1,1)^T \frac{1}{\sqrt{2}}(-1,1)^T$

$= \frac{1}{2}(0)$

$= 0.$ ✓



② $\|\hat{u}^{(1)}\| = \frac{1}{\sqrt{2}} \|(1,1)^T\| = 1$

$\|\hat{u}^{(2)}\| = \frac{1}{\sqrt{2}} \|(-1,1)^T\| = 1$ ✓

$\Rightarrow \hat{u}^{(1)}, \hat{u}^{(2)}$ forms orthonormal basis.

- Vectors coordinate depend on the basis used
- If we are using basis $u = \{\hat{u}^{(1)}, \hat{u}^{(2)}\}$. (not necessarily standard basis)

$[\vec{x}]_u = (x_1^u, x_2^u)^T$ ← we write this to denote the coordinates w.r.t. to the basis u.

$\vec{x} = x_1^u \hat{u}^{(1)} + x_2^u \hat{u}^{(2)}$.

amount to scale for the i-th basis.

ex) $\hat{u}^{(1)} = \frac{1}{\sqrt{2}}(1,1)^T$

$\hat{u}^{(2)} = \frac{1}{\sqrt{2}}(-1,1)^T$.

Suppose $[\vec{x}]_u = (3,-4)^T$.

$\vec{x} \cdot \hat{u}^{(1)} = (x_1^u \hat{u}^{(1)} + x_2^u \hat{u}^{(2)}) \cdot \hat{u}^{(1)}$

$= x_1^u \underbrace{\hat{u}^{(1)} \cdot \hat{u}^{(1)}}_{\|\hat{u}\| = 1} + x_2^u \underbrace{\hat{u}^{(2)} \cdot \hat{u}^{(1)}}_{= 0 \text{ b/c orthogonality.}}$

$= x_1^u = 3.$

$\Rightarrow \vec{x} \cdot \hat{u}^{(i)} = x_i^u$ (conclusion result).

ex) $\vec{x} = (2,2)^T$

$\hat{u}^{(1)} = \frac{1}{\sqrt{2}}(1,1)^T$

$\hat{u}^{(2)} = \frac{1}{\sqrt{2}}(-1,1)^T$

since $\vec{x} = \underbrace{x_1^u}_{\text{unknown}} \hat{u}^{(1)} + \underbrace{x_2^u}_{\text{unknown}} \hat{u}^{(2)}$.

$\Rightarrow \begin{bmatrix} 2 \\ 2 \end{bmatrix} = x_1^u \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix} + x_2^u \frac{1}{\sqrt{2}} \begin{bmatrix} -1 \\ 1 \end{bmatrix}$.

$\begin{cases} 2 = x_1^u \frac{1}{\sqrt{2}}(1) + x_2^u \frac{1}{\sqrt{2}}(-1) \\ 2 = x_1^u \frac{1}{\sqrt{2}}(1) + x_2^u \frac{1}{\sqrt{2}}(1) \end{cases}$

... solve x_1^u, x_2^u.

Alternatively, since $\vec{x} = x_1^u \hat{u}^{(1)} + x_2^u \hat{u}^{(2)}$

$\vec{x} \cdot \hat{u}^{(1)} = x_1^u$ (previous result).

$\vec{x} \cdot \hat{u}^{(2)} = x_2^u$

$\Rightarrow x_1^u = \begin{bmatrix} 2 \\ 2 \end{bmatrix} \cdot \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = 2\sqrt{2}$

$x_2^u = \cdots = 0$

Change of Basis

Let $u = \{\hat{u}^{(1)}, \ldots, \hat{u}^{(d)}\}$ be an orthonormal basis.

Then $[\vec{x}]_u = \begin{pmatrix} x_1^u \\ \vdots \\ x_d^u \end{pmatrix} = \begin{pmatrix} \vec{x} \cdot \hat{u}^{(1)} \\ \vdots \\ \vec{x} \cdot \hat{u}^{(d)} \end{pmatrix}$

$\vec{x} = x_1^u \hat{u}^{(1)} + \cdots + x_d^u \hat{u}^{(d)}$,

for k-th coordinate.

$\vec{x} \cdot \hat{u}^{(k)} = x_k^u (\hat{u}^{(k)} \cdot \hat{u}^{(k)}) + 0$

$= x_k^u$

(by dot producting the new basis). (only true for orthonormal basis).

ex) suppose $\vec{x} = (2,1)^T$. $[\vec{x}]_u = (2,1)^T$ for standard basis.

$\hat{u}^{(1)} = \frac{1}{\sqrt{2}}(1,1)^T$ ← change to new basis.

$\hat{u}^{(2)} = \frac{1}{\sqrt{2}}(-1,1)^T$.

$[\vec{x}]_u = \begin{bmatrix} \vec{x} \cdot \vec{u}^{(1)} \\ \vec{x} \cdot \vec{u}^{(2)} \end{bmatrix}$

$\begin{bmatrix} 2 \\ 1 \end{bmatrix} \cdot \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \frac{3}{\sqrt{2}}$

$\begin{bmatrix} 2 \\ 1 \end{bmatrix} \cdot \frac{1}{\sqrt{2}} \begin{bmatrix} -1 \\ 1 \end{bmatrix} = \frac{-1}{\sqrt{2}}$.

$= \frac{1}{\sqrt{2}} \begin{bmatrix} 3 \\ 1 \end{bmatrix}$

ex) $\vec{x} = (-1, 4)$

$\hat{u}^{(1)} \cdot \hat{e}^{(1)} = 3$ $\quad$ $\hat{u}^{(2)} \cdot \hat{e}^{(1)} = 1$

$\hat{u}^{(1)} \cdot \hat{e}^{(2)} = -2$ $\quad$ $\hat{u}^{(2)} \cdot \hat{e}^{(2)} = 5$

$\Rightarrow \hat{u}^{(1)} = \vec{u}^{(1)} = \begin{bmatrix} 3 \\ -2 \end{bmatrix}$ in terms of standard basis.

$\hat{u}^{(2)} = \vec{u}^{(2)} = \begin{bmatrix} -1 \\ 5 \end{bmatrix}$ (new basis).

→ we can check orthogonality:

$\hat{u}^{(1)} \cdot \hat{u}^{(2)} = -13 \neq 0$.

So. $[x]_u \neq \begin{bmatrix} \vec{x} \cdot \hat{u}^{(1)} \\ \vec{x} \cdot \hat{u}^{(2)} \end{bmatrix}$.

$\vec{x} = x_1^u \hat{u}^{(1)} + x_2^u \hat{u}^{(2)}$

$\begin{bmatrix} -1 \\ 4 \end{bmatrix} = x_1^u \begin{bmatrix} 3 \\ -2 \end{bmatrix} + x_2^u \begin{bmatrix} -1 \\ 5 \end{bmatrix}$.

solve for x_1^u & x_2^u.

$x_1^u = -1/13$

$x_2^u = 10/13$

Function of a vector : $f: \mathbb{R}^n \to \mathbb{R}^{n'}$

A transformation $\vec{f}$ is a function that takes in a vector and returns a vector of same dimensionality.

$\downarrow$

$f: \mathbb{R}^n \to \mathbb{R}^n$

- Transformation is a vector field. → calculate corresponding vectors for each point.

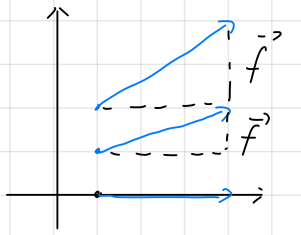
ex)

$f(\vec{x}) = (0, x_2^2)^T$

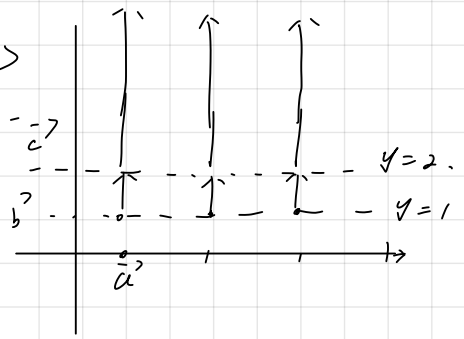
$\vec{a} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ $f(\vec{a}) = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$

$\vec{b} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$ $f(\vec{b}) = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$

$\vec{c} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$ $f(\vec{c}) = \begin{bmatrix} 0 \\ 4 \end{bmatrix}$



for $\vec{f}(\vec{x}) = (3x_1, x_2)$.



Then a transformation is generally:

$\vec{f}(\vec{x}) = (g_1(\vec{x}), g_2(\vec{x}))^T$.

Linear Transformation :

Transformation is linear if

$$\vec{f}(a\vec{x} + b\vec{y}) = a\vec{f}(\vec{x}) + b\vec{f}(\vec{y})$$

ex) To check if a transformation is linear :

$$\vec{f}(\vec{x}) = (x_2, -x_1)^T$$

$$LHS = \vec{f}(\alpha\vec{\delta} + \beta\vec{y}) = \begin{bmatrix} \alpha\delta_2 + \beta y_2 \\ -(\alpha\delta_1 + \beta y_1) \end{bmatrix}$$

$$RHS = \alpha\vec{f}(\vec{\delta}) + \beta\vec{f}(\vec{y}) = \alpha\begin{bmatrix}\delta_2 \\ \delta_1\end{bmatrix} + \beta\begin{bmatrix} y_2 \\ y_1\end{bmatrix} = \begin{bmatrix} \alpha\delta_2 + \beta y_2 \\ -\alpha\delta_1 - \beta y_1 \end{bmatrix}$$

So LHS = RHS, $\vec{f}$ is linear.

- Suppose $\vec{f}$ is a linear transformation:

$$\vec{f}(\vec{x}) = \vec{f}(x_1\hat{e}^{(1)} + x_2\hat{e}^{(2)}).$$
$$= x_1\vec{f}(\hat{e}^{(1)}) + x_2\vec{f}(\hat{e}^{(2)}).$$

$\vec{x} = x_1\hat{e}^{(1)} + x_2\hat{e}^{(2)}$

standard basis.

linear transformation to basis vector.

⇓

- transformation determined by what it does to basis vector.
- Note: do not hold for arbitary transformation

$$\vec{f}(a\vec{x} + \beta\vec{y}) \not= a\vec{f}(\vec{x}) + \beta\vec{f}(\vec{y})$$

$$\vec{f}(x_1\hat{e}_1^{(1)} + x_2\hat{e}^{(2)}) \not= x_1\vec{f}(\hat{e}^{(1)}) + x_2\vec{f}(\hat{e}^{(2)}).$$

ex) $\vec{f}(\vec{x}) = (0, x_2^2)^T$

- b/c of linearity, along any direction $\vec{f}$ change only in scale.

$$\vec{f}(\lambda\vec{x}) = \lambda f(\vec{x}).$$

$\vec{x}$	$\lambda\vec{x}$
$\vec{f}(\vec{x})$	$f(\lambda\vec{x}) = \lambda f(\vec{x})$.

Along a given direction in vector field, can be simplified as unit vector direction.

Transformation in other basis:

$$[\vec{f}(\vec{x})]_u \quad \& \quad [\vec{x}]_u.$$

ex). $\vec{f}(\vec{x}) = (x_1 + x_2,\ x_1 - x_2)^T$. (in standard basis).

↑ linear transformation

$\vec{u}^{(1)} = \frac{1}{\sqrt{2}}(1,1)^T$.

$\vec{u}^{(2)} = \frac{1}{\sqrt{2}}(-1,1)^T$.

↑ orthonormal basis

$$[\vec{f}(\vec{x})]_u = \begin{bmatrix} \vec{f}(\vec{x}) \cdot \vec{u}^{(1)} \\ \vec{f}(\vec{x}) \cdot \vec{u}^{(2)} \end{bmatrix} = \begin{bmatrix} \delta_1 - \delta_2 \\ -(\delta_1 + \delta_2) \end{bmatrix}$$

$\vec{x} = \delta_1 \hat{u}^{(1)} + \delta_2 \hat{u}^{(2)}$

$\vec{f}(\vec{x}) = \vec{f}(\delta_1 \hat{u}^{(1)} + \delta_2 \hat{u}^{(2)})$

$= \delta_1 \vec{f}(\hat{u}^{(1)}) + \delta_2 \vec{f}(\hat{u}^{(2)})$

calculation.

plug in

$$[\vec{x}]_u = \begin{bmatrix} \vec{x} \cdot \vec{u}^{(1)} \\ \vec{x} \cdot \vec{u}^{(2)} \end{bmatrix} = \cdots$$

Matrices

$$(A\vec{x})_i = \sum_{j=1}^{n} A_{ij} x_j$$

in general:

$$\begin{pmatrix} \uparrow & \uparrow & \uparrow \\ \vec{a}^{(1)} & \vec{a}^{(2)} & \vec{a}^{(3)} \\ \downarrow & \downarrow & \downarrow \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = x_1 \vec{a}^{(1)} + x_2 \vec{a}^{(2)} + x_3 \vec{a}^{(3)}$$

Then, suppose f linear transformation:

$$\vec{f}(\vec{x}) = \vec{f}(x_1 \hat{e}^{(1)} + x_2 \hat{e}^{(2)}) = x_1 \vec{f}(\hat{e}^{(1)}) + x_2 \vec{f}(\hat{e}^{(2)})$$

Standard Basis.

Idea:

A $(n \times n)$ matrix can be interpreted as a compact representation of linear transformation $\vec{f}: K^n \to K^n$.

$A\vec{x} = \vec{f}(\vec{x})$.

$$A = \begin{pmatrix} \uparrow & \uparrow \\ \vec{f}(\hat{e}^{(1)}) & \vec{f}(\hat{e}^{(2)}) \\ \downarrow & \downarrow \end{pmatrix}$$

Matrix as a form of linear transformation.

b/c

$$\vec{f}(\vec{x}) = A\vec{x} = A\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

$$= x_1 f(\hat{e}^{(1)}) + x_2 f(\hat{e}^{(2)}).$$

ex) $\vec{x} = 3\hat{e}^{(1)} - 4\hat{e}^{(2)} = \begin{pmatrix} 3 \\ -4 \end{pmatrix}$

$\vec{f}(\hat{e}^{(1)}) = -\hat{e}^{(1)} + 3\hat{e}^{(2)}$

$\vec{f}(\hat{e}^{(2)}) = 2\hat{e}^{(1)}$

so, $\vec{f}(\vec{x}) = x_1 \hat{f}(\hat{e}^{(1)}) + x_2 \hat{f}(\hat{e}^{(2)})$

$= 3\begin{bmatrix} -1 \\ 3 \end{bmatrix} + -4\begin{bmatrix} 2 \\ 0 \end{bmatrix}$

Alternatively.

$A = \begin{bmatrix} -1 & 2 \\ 3 & 0 \end{bmatrix}$

$\vec{f}(\vec{x}) = Ax = \begin{bmatrix} -1 & 2 \\ 3 & 0 \end{bmatrix}\begin{bmatrix} 3 \\ -4 \end{bmatrix}$.

Matrices in Other Bases.

$A^u = \left([\vec{f}(\hat{u}^{(1)})]_u \cdots [\vec{f}(\hat{u}^{(d)})]_u \right)$

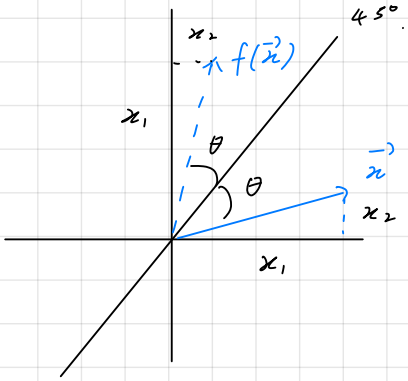
$\vec{f}(\vec{x}) = \vec{f}(x_1^u \hat{u}^{(1)} + x_2^u \hat{u}^{(2)}) = x_1^u \cdot \vec{f}(\hat{u}^{(1)}) + x_2^u \cdot \vec{f}(\hat{u}^{(2)})$

$\Rightarrow [f(\vec{x})]_u = [x_1^u \vec{f}(\hat{u}^{(1)}) + x_2^u \vec{f}(\hat{u}^{(2)})]_u = x_1^u [\vec{f}(\hat{u}^{(1)})]_u + x_2^u [\vec{f}(\hat{u}^{(2)})]_u$

by u = orthonormal basis.

$= A^u [\vec{x}_1]_u$

Consider $\vec{f}$ which mirror a vector over 45° line.



$\vec{f}\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = \begin{bmatrix} x_2 \\ x_1 \end{bmatrix}$

$\Downarrow$

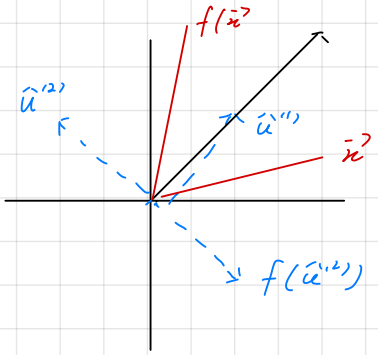
$A\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} x_2 \\ x_1 \end{bmatrix}$

$\Downarrow$

$A = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$

ex) $\hat{u}^{(1)} = \frac{1}{\sqrt{2}}(1,1)^T$

$\hat{u}^{(2)} = \frac{1}{\sqrt{2}}(-1,1)^T$



$\Rightarrow$ $\vec{f}(\hat{u}^{(1)}) = \hat{u}^{(1)}$ transformation of orthogonal basis

$\vec{f}(\hat{u}^{(2)}) = -\hat{u}^{(2)}$.

if u is orthonormal:

$\Downarrow$ reprojected back to u transformed vector to u basis.

$$A^u = \left[\left[\vec{f}(\hat{u}^{(1)})\right]_u, \left[\vec{f}(\hat{u}^{(2)})\right]_u \right]$$

$$= \begin{bmatrix} \hat{u}^{(1)} \cdot \hat{u}^{(1)} & -\hat{u}^{(2)} \cdot \hat{u}^{(1)} \\ \hat{u}^{(1)} \cdot \hat{u}^{(2)} & -\hat{u}^{(2)} \cdot \hat{u}^{(1)} \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

$$[\vec{s}]_u = \begin{bmatrix} \vec{s} \cdot \hat{u}^{(1)} \\ \vec{s} \cdot \hat{u}^{(2)} \end{bmatrix}$$

linear transformation of orthogonal basis as a matrix representation.

Eigenvector & Eigenvalue

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

$\vec{v}$: eigenvector of A; λ: eigenvalue

$\vec{f}$ linear transformation.

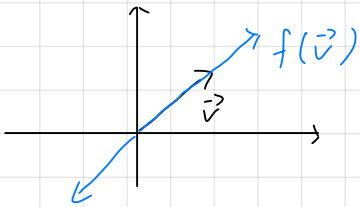
$\Rightarrow$ eigenvector of $\vec{f}$ with eigenvalue λ is a nonzero vector $\vec{v}$ s.t.

$$f(\vec{v}) = \lambda \vec{v}.$$

idea:

When $\vec{f}$ applies to one of its eigenvector, $\vec{f}$ scales it.

(not changing any direction).



$$f(\vec{v}) = \lambda \vec{v}$$

Finding eigenvector => ① Graphically, find vector that are in the same direction after transformation $\vec{f}$.

② Note: not all matrices haven even one eigenvector

Thm: Let A be an $n \times n$ "symmetric" matrix. → ∃ a set of n eigenvector that are mutually orthogonal. (there can be many set).

(Spectral Theorem) $\Rightarrow$ there exist n eigenvectors of A which are all mutually orthogonal.

"symmetric" $\Leftrightarrow$ $A^T = A$

① Symmetric linear transformation have axes of symmetry.
② The axes of symmetry are orthogonal to one another.
③ The action of $\vec{f}$ along Axis of symmetry scales input.
④ Size of scaling might different

$f(\vec{u}^{(1)}) = \lambda_1 \vec{u}^{(1)}$

$f(\vec{u}^{(2)}) = \lambda_2 \vec{u}^{(2)}$.

Note, if A is diagonal, its eigenvector are the standard basis vector

Total Symmetry: infinity many eigenvector

⇓

ex) $A = \begin{bmatrix} 3 & 0 \\ 0 & 3 \end{bmatrix} = 3 \cdot \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = 3I$

every vector is an eigenvector of A!

Eigenvector as basis vector:

If A is symmetry matrix, pick d of its eigenvector $\hat{u}^{(1)}, \cdots \hat{u}^{(d)}$ to form an orthonormal basis.

Any vector $\vec{x}$ can be written in this basis: (eigen decomposition).

$\vec{x} = b_1 \hat{u}^{(1)} + \cdots + b_d \hat{u}^{(d)}$

$\mathcal{U} = \{\hat{u}^{(1)} \cdots \hat{u}^{(d)}\}$ (eigen-basis).

$\Rightarrow [\vec{x}]_{\mathcal{U}} = \begin{bmatrix} b_1 \\ \vdots \\ b_d \end{bmatrix}$ as eigen-basis expression.

$f(\vec{x}) = A\vec{x} = A(b_1 \hat{u}^{(1)} + \cdots + b_d \hat{u}^{(d)})$

$= b_1 (A\hat{u}^{(1)}) + \cdots + b_d (A\hat{u}^{(d)})$

$= b_1(\lambda_1 \hat{u}^{(1)}) + \cdots + b_d(\lambda_d \hat{u}^{(d)})$

$= (b_1\lambda_1)\hat{u}^{(1)} + \cdots + (b_d\lambda_d)\hat{u}^{(d)}$.

So, if A is symmetric, eigenbasis U is an natural basis.

$f(\vec{x}) = A\vec{x} = \underline{\lambda_1} b_1 \hat{u}^{(1)} + \underline{\lambda_2} b_2 \hat{u}^{(2)} + \cdots + \underline{\lambda_d} b_d \hat{u}^{(d)}$

scale of each coordinate ith of $[\vec{x}]_U$ by eigenvalue λ_i

Eigenvector as optimizer

ex) find unit vector $\vec{x}$ s.t. $\|A\vec{x}\|$ largest.

$\Downarrow\Uparrow$

largest in direction of axis of symmetry.

(direction of eigenvector w/ largest eigenvalue).

largest transformation in terms of magnitude.

To maximize/minimize $\|A\vec{x}\| = \|f(\vec{x})\|$ over unit vectors.

pick $\vec{x}$ to be an eigenvector of $\vec{f}$ with largest/smallest eigenvalue.

In particular, maximizer of $\|A\vec{x}\|$ s.t. $\|\vec{x}\| = 1$ is the top eigenvector of A.

pf: Let $\vec{x} = b_1\hat{u}^{(1)} + b_2\hat{u}^{(2)}$ & $\|x\| = 1$ & $A\vec{x} = b_1\lambda_1\hat{u}^{(1)} + b_2\lambda_2\hat{u}^{(2)}$

$b_1^2 + b_2^2 = 1$

$\|A\vec{x}\|^2 = (A\vec{x})\cdot(A\vec{x}) = (b_1\lambda_1\hat{u}^{(1)} + b_2\lambda_2\hat{u}^{(2)}) \cdot (b_1\lambda_1\hat{u}^{(1)} + b_2\lambda_2\hat{u}^{(2)})$.

① $\|\hat{u}^{(i)}\|^2 = 1$

② $\hat{u}^{(i)} \cdot \hat{u}^{(j)} = 0$ for $i \neq j$.

$\rightarrow = \cdots$

$= b_1^2\lambda_1^2 + b_2^2\lambda_2^2$

↑

To maximize, if $|\lambda_1| > |\lambda_2|$ $(\lambda_1^2 > \lambda_2^2)$

then $b_1 = 1$, $b_2 = 0$ maximize,

$\vec{x} = \hat{u}^{(1)}$, $\hat{u}^{(1)}$ is top-eigenvector.

To maximize $\vec{x} A \vec{x}$ over unit vector, pick $\vec{x}$ to be top eigenvector.

pf:

$$\vec{x} A \vec{x} = (b_1 \hat{u}^{(1)} + b_2 \hat{u}^{(2)}) \cdot (b_1 \lambda_1 \hat{u}^{(1)} + b_2 \lambda_2 \hat{u}^{(2)})$$

$$= b_1^2 \lambda_1^2 + 0 + 0 + b_2^2 \lambda_2^2$$

$$= b_1^2 \lambda_1^2 + b_2^2 \lambda_2^2$$

... (same proof as above).

ex) Max $4x_1^2 + 2x_2^2 + 3x_1x_2$ subject to $x_1^2 + x_2^2 = 1$.

$\Downarrow$

$\|\vec{x}\| = 1$

so, $\vec{x} A \vec{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} A \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = a x_1^2 + (b+c) x_1 x_2 + d x_2^2$

$= 4x_1^2 + 3x_1x_2 + 2x_2$, A symmetry

$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$.

$\Rightarrow a = 4$

$b = c = 1.5$

$d = 2$.

To maximize $\vec{x} \cdot A\vec{x}$ s.t. $\|\vec{x}\| = 1$, $A = \begin{bmatrix} 4 & 1.5 \\ 1.5 & 2 \end{bmatrix}$.

find $\vec{x}^*$, top eigenvector of A

Eigenvector as Equilibria:

idea. $\vec{f}(\vec{x})$ rotates $\vec{x}$ toward the "top" eigenvector.

$\vec{v}$ is an equilibrium. $\vec{f}(\vec{u}) = \lambda \vec{u}$

Method of computing top eigenvector/value. (power method).

- initialize $\vec{x}^{(0)}$
- repeat until convergence:

$$\vec{x}^{(i+1)} = A\vec{x}^{(i)} / \|A\vec{x}^{(i)}\|$$

Change of Basis Matrices

$$\vec{x} = \begin{pmatrix} a_1 \\ a_2 \end{pmatrix} = a_1 \hat{e}^{(1)} + a_2 \hat{e}^{(2)}$$

suppose $\hat{u}^{(1)}$, $\hat{u}^{(2)}$:

$$\vec{x} = b_1 \hat{u}^{(1)} + b_2 \hat{u}^{(2)}$$

$b_1 = \vec{x} \hat{u}^{(1)}$

$b_2 = \vec{x} \hat{u}^{(2)}$

(orthonormal basis property).

Represent change of basis as matrix.

$$A = \left[\vec{f}(\hat{e}^{(1)}) \quad \vec{f}(\hat{e}^{(2)}) \right] = \begin{bmatrix} \hat{e}^{(1)} \cdot \hat{u}^{(1)} & \hat{e}^{(2)} \cdot \hat{u}^{(1)} \\ \hat{e}^{(1)} \cdot \hat{u}^{(2)} & \hat{e}^{(2)} \cdot \hat{u}^{(2)} \end{bmatrix}$$

$\hat{u}^{(1)} \dots \hat{u}^{(2)}$ orthonormal basis U

$$= \begin{bmatrix} \hat{u}^{(1)T} \\ \hat{u}^{(2)T} \end{bmatrix}$$

as a matrix

$$\Rightarrow A = U = \begin{pmatrix} \hat{u}^{(1)T} \\ \vdots \\ \hat{u}^{(d)T} \end{pmatrix}$$

U: basis matrix

So change of basis as matmul:

$$\vec{f}(\vec{x}) = [\vec{x}]_U = U\vec{x}.$$

If U is change of basis matrix:

$$\vec{x} = U^T [\vec{x}]_U.$$

ex)

$$\hat{u}^{(1)} = (\sqrt{3}/2, 1/2)^T$$

$$\hat{u}^{(2)} = (-1/2, \sqrt{3}/2)^T$$

$$\vec{x} = (1/2, 1)^T$$

$$\text{So, } [\vec{x}]_U = U\vec{x} = \begin{bmatrix} \sqrt{3}/2 & 1/2 \\ -1/2 & \sqrt{3}/2 \end{bmatrix} \begin{bmatrix} 1/2 \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} -\sqrt{3}/4 + 1/2 \\ -1/4 + \sqrt{3}/2 \end{bmatrix}$$

Diagonalization:

Recall matrix as linear transformation f:

$$A = \left(\vec{f}(\hat{e}^{(1)}) \cdots \vec{f}(\hat{e}^{(d)}) \right)$$

If we use a different basis $\mathcal{U} = \{\vec{u}^{(1)} \cdots \vec{u}^{(d)}\}$:

$$A_{\mathcal{U}} = \left([\vec{f}(\vec{u}^{(1)})]_{\mathcal{U}} \cdots [\vec{f}(\vec{u}^{(d)})]_{\mathcal{U}} \right).$$

$$[\vec{x}]_{\mathcal{U}} = \mathcal{U}\vec{x}$$

$\Rightarrow$ $\vec{y} = A\vec{x}$

$[\vec{y}]_{\mathcal{U}} = A_{\mathcal{U}} [\vec{x}]_{\mathcal{U}}$.

$\Rightarrow$ Suppose A is matrix, find basis $\mathcal{U}$ s.t. $A_{\mathcal{U}}$ is diagonal.

⇓

If A is symmetric matrix, pick d of its eigenvector $\hat{u}^{(1)} \cdots \hat{u}^{(d)}$ to form orthogonal basis.

$$A_{\mathcal{U}} = \left([\vec{f}(\vec{u}^{(1)})]_{\mathcal{U}} \cdots \right) = \begin{bmatrix} \lambda_1 & & 0 \\ & \ddots & \\ 0 & & \lambda_d \end{bmatrix}$$

diagonal ! eigenvalue.

why ?

$$[\vec{f}(\vec{u}^{(1)})]_{\mathcal{U}} = \begin{pmatrix} \vec{f}(\vec{u}^{(1)}) \cdot \vec{u}^{(1)} \\ \vdots \\ \vec{f}(\hat{u}^{(1)}) \cdot \vec{u}^{(d)} \end{pmatrix} \overset{\text{def.}}{=} \begin{pmatrix} \lambda_1 \hat{u}^{(1)} \cdot \hat{u}^{(1)} \\ \vdots \\ \lambda_1 \underbrace{\hat{u}^{(1)} \cdot \hat{u}^{(d)}}_{=0 \text{ b/c orthonormal}} \end{pmatrix} = \begin{pmatrix} \lambda_1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

- matrix of linear transformation $\vec{f}$ in basis of its eigenvector is a diagonal matrix.

entries are eigenvalue.

Compute $f(\vec{x})$

① · change basis to eigenbasis $\mathcal{U}$: $\mathcal{U}\vec{x}$

② · apply $\vec{f}$ in eigenbasis with $A_{\mathcal{U}}$.

$[\vec{y}] = A_{\mathcal{U}} [\vec{x}]_{\mathcal{U}}$.

③ · go back to standard basis.

$\mathcal{U}^T [\vec{y}]_{\mathcal{U}}$.

OR $\vec{y} = A\vec{x}$

$\mathcal{U}\vec{y} = \mathcal{U}A\vec{x}$

$[\vec{y}]_{\mathcal{U}} = \mathcal{U}A\mathcal{U}^T [\vec{x}]_{\mathcal{U}}$

$A_{\mathcal{U}} [\vec{x}]_{\mathcal{U}} = \mathcal{U}A\mathcal{U}^T [\vec{x}]_{\mathcal{U}}$

$A_{\mathcal{U}} = \mathcal{U}A\mathcal{U}^T$

$\Rightarrow A = \mathcal{U}^T A_{\mathcal{U}} \mathcal{U}$ b/c $\mathcal{U}^{-1} = \mathcal{U}^T$

$$\vec{f} = U^T\left[Au[U\vec{x}]\right] = A\vec{x}$$

③ ② ①

$$\Rightarrow U^T A U U = A$$

Thm: A be $n \times n$ symmetric matrix.

$\exists$ orthogonal matrix U

diagonal matrix Λ

"U diagonalize A".

s.t. $\boxed{A = U^T \Lambda U}$.

where rows of U are eigenvector of A &

entries of Λ are eigenvalue.

PCA

High dimensional data: $\begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$.

Reduce dimensionality by minimizing loss of information

① $R^n \to R^1$ ($\vec{x} \to z$.)

ex) two features $\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ ← phone width and weight.

⇓

$z \in R^1$ = combination of x_1 and x_1

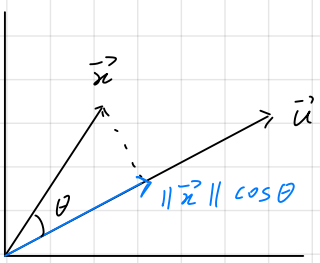
(mixture of feature).

$z = \vec{u} \cdot \vec{x}$ (linear combination).

↑ mixture coefficient, we assume $\|\vec{u}\| = 1$.

idea:

$\vec{x}$ in dir of $\vec{u}$, scale to some magnitude.



since $\vec{u} \cdot \vec{x} = \|\vec{u}\| \cdot \|\vec{x}\| \cos\theta$

$= (\|\vec{x}\| \cos\theta$.

So, $\vec{u}$ define a direction

$\vec{z}^{(i)} = \vec{x}^{(i)} \cdot \vec{u}$ represent position/magnitude of $\vec{x}$ along $\vec{u}$.

↓ we pick:

along direction of "max variance", all dimension are well-seperated.

Algorithm:

pick $\vec{u}$ to be the direction of "max variance".

$z^{(i)} = \vec{x}^{(i)} \cdot \vec{u}$

what is max variance?

⇓

principle component.

Covariance Matrix

$$Var(X) = \frac{1}{n} \sum_{i=1}^{n} (x^{(i)} - \mu)^2$$

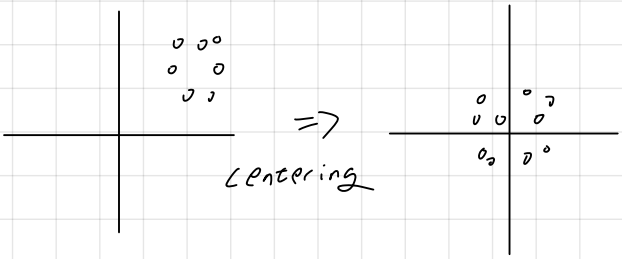
↓

$$\mu = \frac{\sum_{i=1}^{n} x^{(i)}}{n}$$

Measure how x_1, x_2 vary together: → $Cov(x_1, x_2)$

$$= \frac{1}{n} \sum_{i=1}^{n} (\vec{x}_1^{(i)} - \mu_1)(\vec{x}_2^{(j)} - \mu_2)$$

quantify the "trend" or relationship.



⇒ centering

$$\vec{z}^{(i)} = \begin{pmatrix} \vec{x}_1^{(i)} - \mu_1 \\ \vdots \\ \vec{x}_d^{(i)} - \mu_d \end{pmatrix} \quad \text{or} \quad \vec{z}^{(i)} = \vec{x}^{(i)} - \mu.$$

After centering:

$$Cov(x_i, x_j) = \frac{1}{n} \sum_{k=1}^{n} \vec{x}_i^{(k)} \vec{x}_j^{(k)}$$

(Cov of features i & j)

i: dim i; j: dim j; k: data points.

Given data $\vec{x}^{(1)} \dots \vec{x}^{(n)} \in \mathbb{R}^d$.

Sample covariance matrix C is $d \times d$ matrix whose i,j entry is defined to be σ_{ij}.

$$\sigma_{ij} = \frac{1}{n} \sum_{k=1}^{n} \vec{x}_i^{(k)} \vec{x}_j^{(k)}$$

$$C = \begin{bmatrix} \sigma_{11} & \sigma_{12} & \cdots & \sigma_{1d} \\ \vdots & & & \\ \sigma_{d1} & \cdots & & \sigma_{dd} \end{bmatrix}$$

- diagonal are variance $\sigma_{ii} = Cov(x_i, x_i) = Var(x_i)$.
- symmetric matrix.

sometimes,

$$\sigma_{ij} = \frac{1}{n-1} \sum_{k=1}^{n} \vec{x}_i^{(k)} \vec{x}_j^{(k)}$$

(unbiased estimator).

How to compute C:

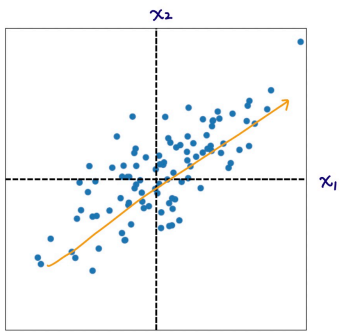
① $X \in \mathbb{R}^{n \times d}$ (n: # data point, d: # dimension.)

② Make Z by centering X: $Z = X - \begin{pmatrix} -\mu_1^T- \\ \vdots \\ -\mu_d^T- \end{pmatrix}$

③ $C = \frac{1}{n} Z^T Z$.

What is Covariance Matrix?

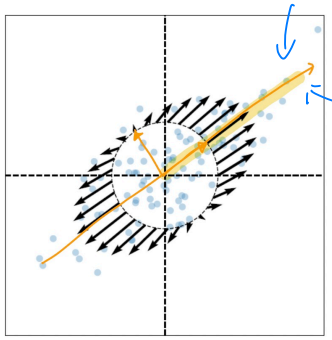
=>



(positive) trend btw x_1 & x_2

$$C \approx \begin{pmatrix} 10 & 5 \\ 5 & 7 \end{pmatrix}$$

top eigenvector is in dir. of max variance.
eigenvalue proportional to variance.



eigenvector describle the data "principle direction".

Eigenvectors:

$$\vec{u}^{(1)} \approx \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$\vec{u}^{(2)} \approx \frac{1}{\sqrt{2}} \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

Not work for "non-linear" data.

idea: top eigenvector of covariance matrix points in direction of max variance.

proof:

let $\vec{u}$ be unit vector. $\|\vec{u}\| = 1$

$z^{(i)} = \vec{x}^{(i)} \cdot \vec{u}$, new features for $\vec{x}^{(i)} \in \mathbb{R}^d$.

$$Var(z) = \frac{1}{n}\sum_{i=1}^{n}(z^{(i)} - \mu_z)^2 = \frac{1}{n}\sum_{i=1}^{n}(\vec{x}^{(i)} \cdot \vec{u} - \mu_z)^2.$$

μ_z = mean of $\{z^{(i)}\}_{i=1}^{n}$ $= \frac{1}{n}\left(\sum_{i=1}^{n} z^{(i)}\right) = \frac{1}{n}\left(\sum_{i=1}^{n} \vec{x}^{(i)} \cdot \vec{u}\right)$

$= \vec{u}\left(\frac{1}{n}\sum_{i=1}^{n}\vec{x}^{(i)}\right)$

$= \vec{u} \cdot \mu_x$. (μ_x: mean of data points)

Variance of data in direction $\vec{u}$ is:

$$Var(z) = \frac{1}{n}\sum_{i=1}^{n}(\vec{x}^{(i)} \cdot \vec{u} - \mu_z)^2$$

$$= \frac{1}{n}\sum_{i=1}^{n}(\vec{x}^{(i)} \cdot \vec{u} - \vec{u} \cdot \mu_x)^2$$

$$= \frac{1}{n}\sum_{i=1}^{n}[\vec{u}(\vec{x}^{(i)} - \mu_x)]^2 = g(\vec{u})$$

those are given by data.

we can denote $Var(z)$ as $g(\vec{u})$

Goal: find unit vector $\vec{u}$ maximizes function g.

$$\max \; g(\vec{u}) \quad \text{s.t.} \quad \|\vec{u}\| = 1.$$

Observation ①: $g(\vec{u})$ can be written as $g(\vec{u}) = \vec{u}^T C \vec{u}$

(C: covariance matrix of data points $\vec{x}^{(i)}$, $i = 1 \cdots n$.

$\left(C = \frac{1}{n}Z^TZ,\ Z = X - \begin{pmatrix} -\mu_1^T- \\ -\mu_i^T- \end{pmatrix}\right)$

Observation ②: our problem becomes

$$\max \; \vec{u}^T C \vec{u} \quad \text{s.t.} \quad \|\vec{u}\|_2 = 1$$

$\Rightarrow$ The solution is:

- $\vec{u}^* = C$'s top (largest) eigenvector
- $g(\vec{u}^*) = \lambda_{max}$ of C.

Observation ① is true b/c: $(a^T b)^2 = a^T b\, b^T a$

$$g(\vec{u}) = Var(z) = \frac{1}{n}\sum_{i=1}^{n}\left[\vec{u}^T(\vec{x}^{(i)} - \mu_x)\right]^2 = \frac{1}{n}\sum_{i=1}^{n}\vec{u}^T(\vec{x}^{(i)} - \mu_x)(\vec{x}^{(i)} - \mu_x)^T\vec{u}$$

$$= \frac{1}{n}\vec{u}^T\left(\sum_{i=1}^{n}(\vec{x}^{(i)} - \mu_x)(\vec{x}^{(i)} - \mu_x)^T\right)\vec{u} = \vec{u}^T\left(\frac{1}{n}Z^T Z\right)\vec{u} = \vec{u}^T C\vec{u}$$

C

$$\begin{bmatrix} \vec{x}^{(1)} - \mu_x & \vec{x}^{(2)} - \mu_x & \cdots & \vec{x}^{(n)} - \mu_x \end{bmatrix} \quad \begin{bmatrix} \vec{x}^{(1)T} - \mu_x^T \\ \vec{x}^{(2)T} - \mu_x^T \\ \vdots \\ \vec{x}^{(n)T} - \mu_x^T \end{bmatrix}$$

Z^T $\qquad$ Z.

This is essentially the covariance matrix.

Observation ② can be proved.

find $\vec{u}$ maximize $\vec{u}^T C\vec{u}$.

① C is covariance matrix $\Rightarrow$ C is symmetry.

② since C is symmetry, eigenvectors are orthonormal. (Spectral Thm)

Then let C's eigenvectors to $\hat{u}^{(1)}, \hat{u}^{(2)}, \cdots, \hat{u}^{(d)}$ and Let

$$\vec{u} = a_1\hat{u}^{(1)} + a_2\hat{u}^{(2)} + \cdots + a_d\hat{u}^{(d)}$$

Then, we have $g(\vec{u}) = \vec{u}^T C\vec{u} = [a_1\hat{u}^{(1)} + \cdots + a_d\hat{u}^{(d)}]^T C[a_1\hat{u}^{(1)} + \cdots + a_d\hat{u}^{(d)}]$.

$$= [a_1\hat{u}^{(1)} + \cdots + a_d\hat{u}^{(d)}]^T(Ca_1\hat{u}^{(1)} + \cdots + Ca_d\hat{u}^{(d)})$$

$$= [a_1\hat{u}^{(1)} + \cdots + a_d\hat{u}^{(d)}]^T(a_1\lambda_1\hat{u}^{(1)} + \cdots + a_d\lambda_d\hat{u}^{(d)})$$

since $C\hat{u}^{(j)} = \lambda_j\hat{u}^{(j)}$. eigenvector property.

$$= a_1^2\lambda_1 + a_2^2\lambda_2 + \cdots + a_d^2\lambda_d$$

since $\hat{u}^{(i)} \perp \hat{u}^{(j)}$.

Without loss of generality, let $\lambda_1 > \lambda_2 > \cdots > \lambda_d$.

Thus, to maximize $g(\vec{u})$, pick $a_1 = 1$, $a_2 = a_3 = \cdots = a_d = 0$.

$\Rightarrow$ $\vec{u}^* = \hat{u}^{(1)}$ (top eigenvector)

$g(\vec{u}^*) = \lambda_1$

. □.

PCA Algorithm:

Given data points $\vec{x}^{(1)} \cdots \vec{x}^{(n)} \in \mathbb{R}^d$.

$$X = \begin{bmatrix} - & \vec{x}^{(1)T} & - \\ & \vdots & \\ - & \vec{x}^{(n)T} & - \end{bmatrix}$$

① Compute covariance matrix C

$$\mu_x = \frac{1}{n} \sum_{i=1}^{n} \vec{x}^{(i)}$$

② Compute the top eigenvector $\vec{u}$ of C

$$C = \frac{1}{n} Z^T Z$$

③ For $i \in \{1, \cdots, n\}$, create new feature.

$$z^{(i)} = \vec{u} \cdot \vec{x}^{(i)}.$$

$$Z = X - \begin{bmatrix} - & \mu_x^T & - \\ & \vdots & \\ - & \mu_x^T & - \end{bmatrix}$$

$$= \begin{bmatrix} - & x^{(1)T} - \mu_x^T & - \\ & \vdots & \\ - & x^{(n)T} - \mu_x^T & - \end{bmatrix}$$

PCA for k principal components ($k \geq 1$)

2 features: $\vec{z} = (z_1, z_2)$.

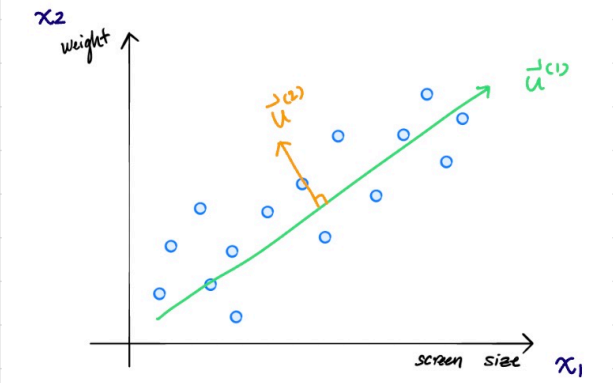
$$z_1 = \vec{u}^{(1)} \cdot \vec{x} = \mu_1^{(1)} x_1 + \cdots + \mu_d^{(1)} x_d.$$

$$z_2 = \vec{u}^{(2)} \cdot \vec{x} = \mu_1^{(2)} x_1 + \cdots + \mu_d^{(2)} x_d.$$

↓ How to find?

① choose $\vec{u}^{(2)}$ to be orthogonal to $\vec{u}^{(1)}$

=> No "redundancy" ($\vec{u}^{(1)}$ & $\vec{u}^{(2)}$ no overlap).



② Since for symmetric matrix, if $\vec{u}$ and $\vec{v}$ are eigenvectors with

distinct eigenvalues, they are automatically orthogonal.

So, choose $\vec{u}^{(2)}$ to be an eigenvector of C.

$\Downarrow$

called the second principle component.

$\vec{u}^{(2)}$ is the second eigenvector of C.
(out of all vectors orthogonal to the principle component, points in direction of max variance).

Thus,

for data $\{\vec{x}^{(1)}, \cdots, \vec{x}^{(n)}\} \in \mathbb{R}^d$,

① Compute covariance matrix C, top $k \leq d$ eigenvectors $\vec{u}^{(1)}, \cdots \vec{u}^{(k)}$.

② For any vector $\vec{x} \in \mathbb{R}$, its new representation in $\mathbb{R}^k$ is:

$\vec{z} = (z_1, \cdots z_k)^T$, where

$z_1 = \vec{x} \cdot \vec{u}^{(1)}$

$z_2 = \vec{x} \cdot \vec{u}^{(2)}$

$\vdots$

$z_k = \vec{x} \cdot \vec{u}^{(k)}$

in matrix form:

$$X_{n \times d} = \begin{bmatrix} - & \vec{x}^{(1)T} & - \\ & \vdots & \\ - & \vec{x}^{(n)T} & - \end{bmatrix}$$

$\vec{\mu} = \frac{1}{n} \sum_{i=1}^{n} \vec{x}^{(i)} \in \mathbb{R}^d$

$C = \frac{1}{n} W^T W$,

$$W_{n \times d} = X_{n \times d} - \begin{bmatrix} - & \vec{\mu}^T & - \\ & \vdots & \\ - & \vec{\mu}^T & - \end{bmatrix} \} n$$

let

$$U_{d \times k} = \begin{bmatrix} \vec{u}^{(1)} & \cdots & \vec{u}^{(k)} \end{bmatrix},$$

$\vec{u}^{(i)}$ is C's top i eigenvector

Thus, $\vec{z} = \begin{bmatrix} - & \vec{u}^{(1)T} & - \\ & \vdots & \\ - & \vec{u}^{(k)T} & - \end{bmatrix} \begin{bmatrix} \\ \vec{x} \\ \\ \end{bmatrix} = U^T x.$

$$= \begin{bmatrix} z_1 \\ \vdots \\ z_k \end{bmatrix} \in \mathbb{R}^k$$

PCA reconstruction point:

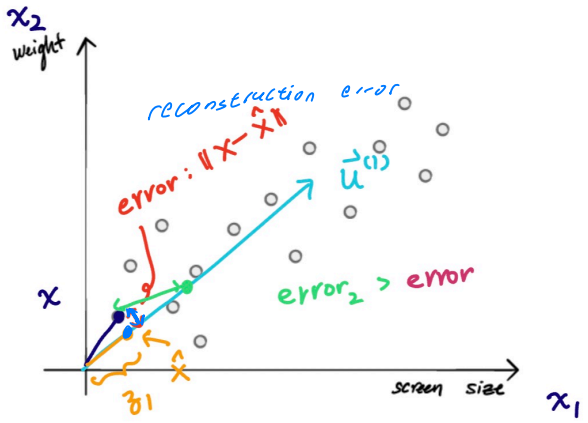
$K^k \to K^d$?

Suppose new representation of $\vec{x}$ is z_1:

$$z_1 = \vec{x} \cdot \vec{u}^{(1)}$$

Given $\vec{u}$ & z_1, how to reconstruct $\vec{x}$?

idea: $\hat{x} = z_1 \vec{u}^{(1)} \approx \vec{x}$.
(reconstructed)



minimizing reconstruction error

$\Rightarrow$ PCA.

Reconstruction of $\vec{x}$ is.

$$\hat{x} = z_1 \vec{u}^{(1)} + z_2 \vec{u}^{(2)} + \cdots + z_k \vec{u}^{(k)} = U\vec{z}.$$

$\hat{x}$: reconstructed point

z_k: projected amount on each principle component.

Reconstruction Error:

$$\|\vec{x} - \hat{x}\|^2 = \|\vec{x} - U\vec{z}\|^2.$$

$$\text{Total error} = \sum_{i=1}^{\hat{}} \|\vec{x}^{(i)} - U\vec{z}^{(i)}\|^2, \quad \vec{z}^{(i)} = U^T \vec{x}^{(i)}$$

PCA interpretation:

Dimensionality Reduction:

Given an orthonormal basis $\{\hat{u}^{(1)} \cdots \hat{u}^{(k)}\}$.
project data $\vec{x}^{(i)}$ on to this new basis and get the
"new" data point $\vec{z}^{(i)} = \begin{bmatrix} \vec{x}^{(i)} \cdot \hat{u}^{(1)} \\ \vdots \\ \vec{x}^{(i)} \cdot \hat{u}^{(k)} \end{bmatrix} = U^T \vec{x}^{(i)}$, $U_{d \times k} = [\hat{u}^{(1)} \cdots \hat{u}^{(k)}]$.

"new" data matrix: $Z_{n \times k} = \begin{bmatrix} — \ \vec{z}^{(1)T} \ — \\ \vdots \\ — \ \vec{z}^{(n)T} \ — \end{bmatrix}$.

For PCA, we are essentially doing ...

choose the basis $\{\hat{u}^{(1)} \cdots \hat{u}^{(k)}\}$ to be special basic:

the top k eigenvector of C, where C is the covariance matrix of x.

($C_{d\times d} = \frac{1}{n} W^T W$).

benefit: #1: $\vec{z}$ has max variance

#2: reconstruction error is minimize $\|\vec{x} - U\vec{z}\|^2$.

#3: z_i & z_j are uncorrelated!

(new feature).

View #1 Maximizing Variance:

PCA maximizes the total variance of the new data.

(choosing most "interesting" new features which are not redundant).

"Total Variance": sum of variances of column Z (up to principle component k).

$$Z = \begin{bmatrix} — & \vec{z}^{(1)T} & — \\ & \vdots & \\ — & \vec{z}^{(n)T} & — \end{bmatrix}, \quad \underbrace{\vec{z}^{(1)T} \cdots \vec{z}^{(n)T}}_{\text{new data.}} \in \mathbb{R}^k$$

1st dim ... kth dim

total variance $= Var(z_1) + Var(z_2) + \cdots + Var(z_k)$.

$= \sum_{i=1}^{k} \lambda_i$, $\lambda_1 \geq \lambda_2 \cdots \geq \lambda_k$. (total variance = sum of eigenvalue of C).

Claim: choosing U to be top eigenvector of C maximizes the total variance among all choices of orthonormal U.

View #2: Minimizing Reconstruction Error

PCA minimizes the reconstruction error.

↓ the "best" projection of points onto a linear subspace of dimensionality k.

total reconstruction error $= \sum_{i=1}^{n} \|\vec{x}^{(i)} - U\vec{z}^{(i)}\|^2$.

Claim: choosing U to be top eigenvectors of C minimizes reconstruction error, among all choices of orthonormal U.

View #3 Decorrelation

PCA learns new representation by rotating data into a basis where features are uncorrelated.

(natural basis vectors are the principle directions).

eigenvectors of covariance matrix.

proof:

$$z = \begin{bmatrix} z_1 \\ \vdots \\ z_k \end{bmatrix}, \quad Z_{n\times k} = \begin{bmatrix} — & \vec{z}^{(1)T} & — \\ & \vdots & \\ — & \vec{z}^{(n)T} & — \end{bmatrix}$$

$$\vec{\mu}_z = \frac{1}{n}\sum_{i=1}^{n} \vec{z}^{(i)}.$$

Relation between z & x: $\vec{z} = U^T\vec{x}$, $U = [\hat{u}^{(1)} \cdots \hat{u}^{(k)}]$.

$$\Rightarrow \vec{\mu}_z = \frac{1}{n}\sum_{i=1}^{n}\vec{z}^{(i)} = \frac{1}{n}\sum_{i=1}^{n} U^T\vec{x}^{(i)} = U^T\left(\frac{1}{n}\sum_{i=1}^{n}\vec{x}^{(i)}\right) = U^T\vec{\mu}_x.$$

So, to show z_i & z_j uncorrelated, $\forall i \neq j$

$\Rightarrow$ show $\mathrm{Cov}(z_i, z_j) = 0$, $\forall i \neq j$

$\Rightarrow$ Cov of z should be a diagonal matrix.

$$\mathrm{Cov}_z = \frac{1}{n}\tilde{Z}^T\tilde{Z}, \quad \tilde{Z} = Z_{n\times k} - \begin{bmatrix} — & \vec{\mu}_z^T & — \\ & \vdots & \\ — & \vec{\mu}_z^T & — \end{bmatrix}.$$

$$\tilde{z}_j = (\vec{z}^{(j)} - \vec{\mu}_z)^T = (U^T\vec{x}^{(j)} - U^T\vec{\mu}_x)^T$$

$$= (\vec{x}^{(j)} - \vec{\mu}_x)^T U$$

centered original data.

$$\Rightarrow \tilde{Z} = \begin{bmatrix} — & (\vec{x}^{(1)} - \vec{\mu}_x^{(1)}) & — \\ & \vdots & \\ — & (\vec{x}^{(n)} - \vec{\mu}_x^{(n)}) & — \end{bmatrix} U = WU$$

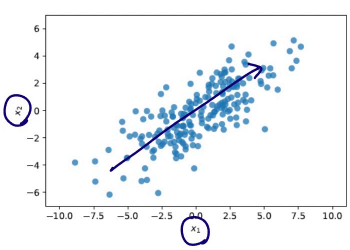
$$\mathrm{Cov}_z = \frac{1}{n}(WU)^T WU$$

$$= \frac{1}{n}U^T W^T W U = U^T\left(\frac{1}{n}W^T W\right)U = U^T C U,$$ where C is Cov. of x.

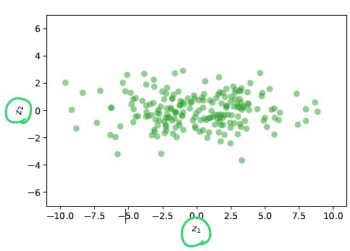
Since U is C's eigenvector, so

$$CU = U\begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_k \end{bmatrix} = U\Lambda$$

$$\Rightarrow Cov_Z = U^T C U$$
$$= U^T U \Lambda$$
$$= \Lambda = \text{diagonal matrix.}$$



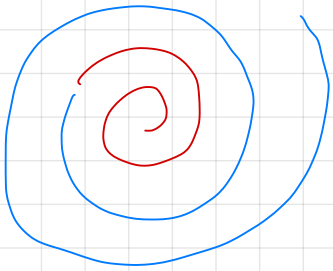
original data:
$Cov_X \neq$ diagonal
$cov(x_1, x_2) > 0$



New data:
no trend, cov_Z = diagonal!
$cov(z_1, z_2) = 0$!

Spectral Embedding

problem:



learn a classifier to split • & •.

⇓

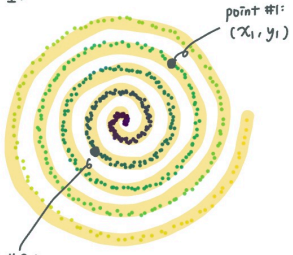
unroll a spiral.

PCA ? => No maximum variance, points are overlapped.
It's just "rotate" the data.

Sol: spectral embedding

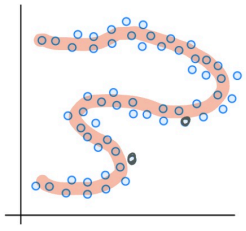
Example:

Example 1:



original data: in $\mathbb{R}^{2 \geq d}$
hidden structure: in $\mathbb{R}^{1 \leq k}$

Example 2:



original data: in $\mathbb{R}^{2 \geq d}$
hidden structure: in $\mathbb{R}^1$

Idea: data expressed with d dimension, but it really confined to k-dimensional region.

original data = d
hidden structure = k $\quad k < d$.

↑ this is called a manifold (a lower dimensional space).

- d is the ambient dimension.
- k is the intrinsic dimension.

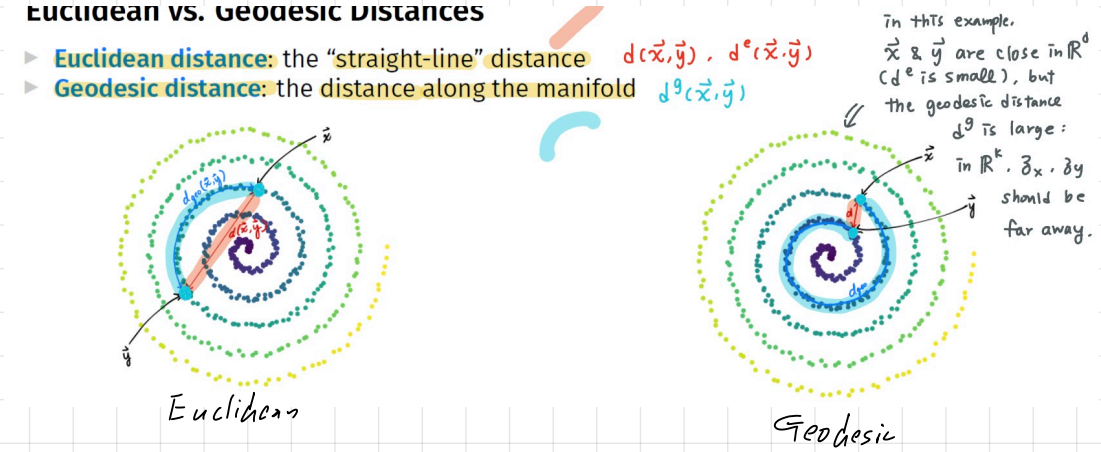
Manifold Learning

Data in high-dimensional ⟶ recover the low-dimensional manifold

linear manifold (PCA)
non-linear manifold (laplacian eigenmap).

Euclidean Distance (dist. in data space)

Geodesic distance. (dist. in manifold space)

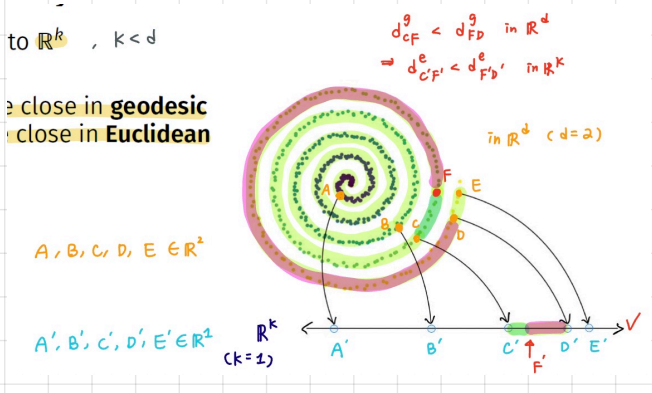


if data ≈ linear manifold, geodesic ≈ euclidean.

Problem: map data in $\mathbb{R}^d$ to $\mathbb{R}^k$, $k < d$ s.t.

if $\vec{x}$ & $\vec{y}$ close in geodesic distance

also close in Euclidean distance.



Solution:

① Given data in R^d, $x \in R^d$.

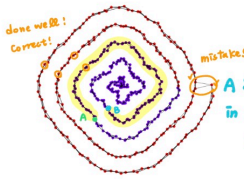
② Built a similarity graph from points

③ Dimensionality reduction: find spectral embedding via graph laplacian.

⇒ Given $\{x^{(i)}\}_{i=1}^n$, $x^{(i)} \in R^d$, compute $\{z^{(i)}\}_{i=1}^n$ $z^{(i)} \in R^k$.

Build a similaring graph:

Step 1:
build a similarity graph from points



Why graphs?

graph is 1D structure estimating the manifold

(i.e. graph can approximate geodesic distance well!) -> approximate geodesic distance well.

Three Approaches

① Eplison neighbors graph.

Create a graph with one node i per point $\vec{x}^{(i)}$.

Add edge between i and j if $\|\vec{x}^{(i)} - \vec{x}^{(j)}\| \leq \varepsilon$.

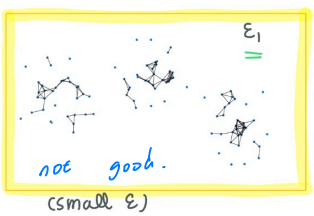
=> unweighted graph.

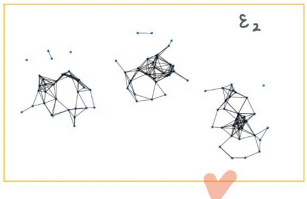
adjacency matrix $W_{n \times n} = [w_{ij}]$, $w_{ij} = 1$ if has edge else 0.

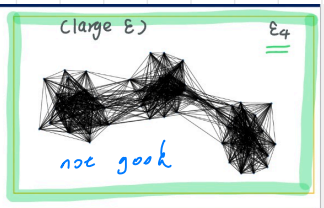
Approach #1: Epsilon Neighbors Graph

What will the graph look like when ε is small? What about when it is large?

$\varepsilon_1 < \varepsilon_2 < \varepsilon_3 < \varepsilon_4$







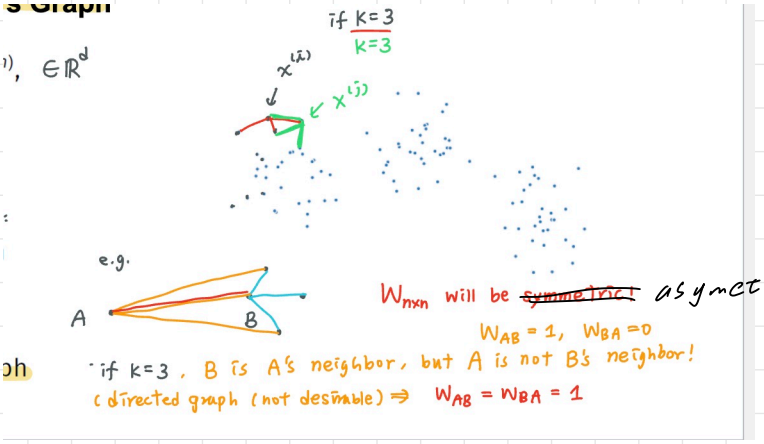
- ε too small, graph underconnected
- ε too large, graph overconnected.

② K-neighbors Graphs

- Create a graph with one node i per point $\vec{x}^{(i)}$
- Add edge between each node i and its K closest neighbors.

=> unweighted graph.

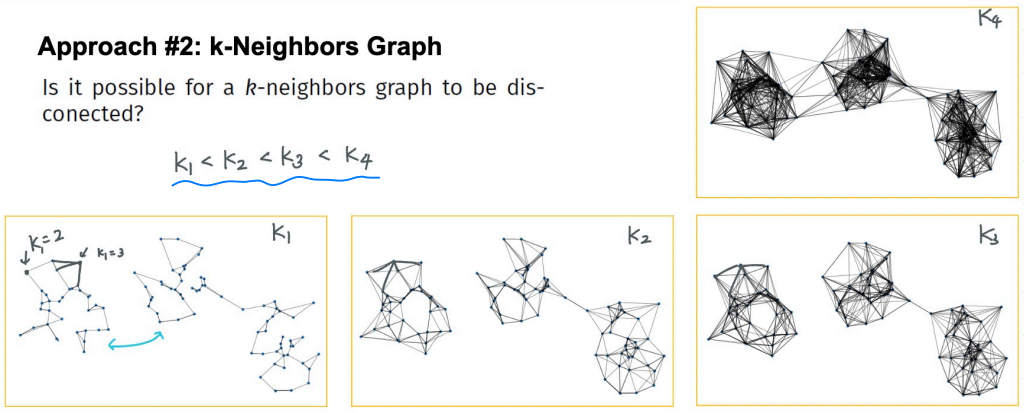
$W_{n\times n} = [w_{ij}]$



Approach #2: k-Neighbors Graph

Is it possible for a *k*-neighbors graph to be disconected?

$k_1 < k_2 < k_3 < k_4$



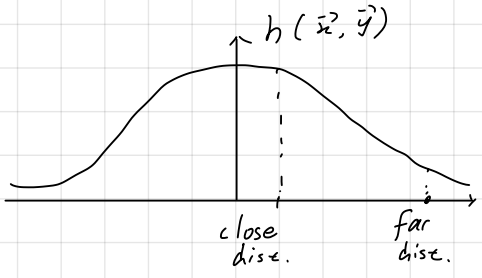
③ Fully Connected Graph

- Create a graph with one node i per point $\vec{x}^{(i)}$.
- Add edge between every pair of nodes.
  Assign weight of $h(\vec{x}^{(i)}, \vec{x}^{(j)})$.

=> weighted graph. in particular, two close points has large weight.

Common similarity function: $h(\vec{x}, \vec{y}) = e^{-\|\vec{x}-\vec{y}\|^2/\sigma^2}$.

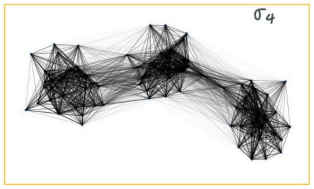
value of σ influence the weight.

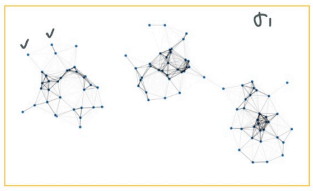


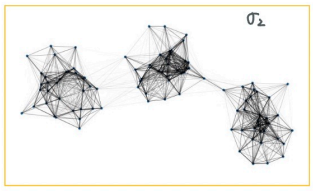
Approach #3: Fully Connected Graph

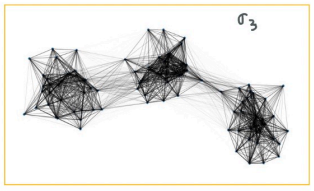
same $\|\vec{x}-\vec{y}\|$ large σ, has large h (more black edge)

$\sigma_1 < \sigma_2 < \sigma_3 < \sigma_4$

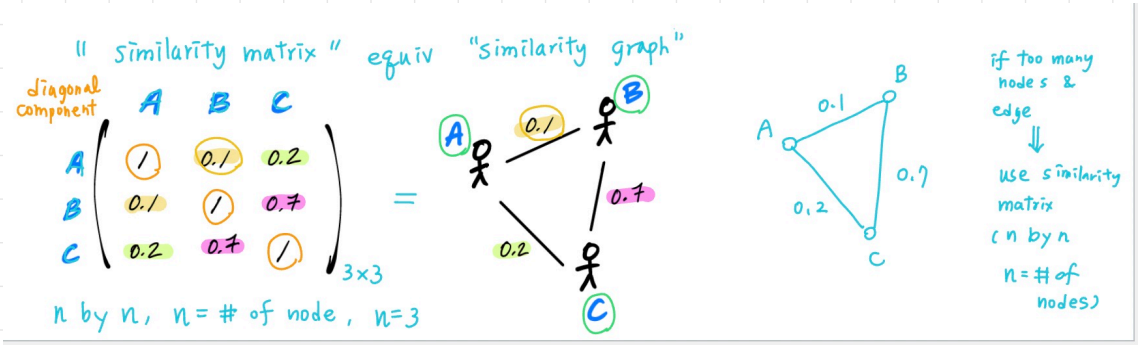








Similarity graph → Similarity matrix.



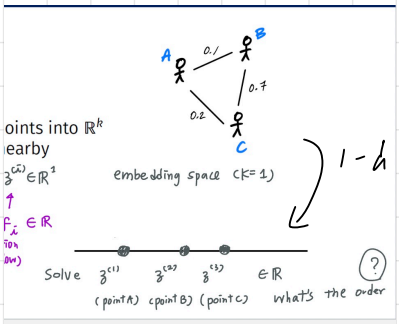
Embed nodes of similarity graph as points.

- Given similarity graph with n nodes & a # of dimensions k.

Compute: an embedding of n points into R^k s.t. similar objects are nearby.

(compute $z^{(i)} \in R^k$, $z^{(i)}$ follows the similarity in similarity graph?

① when k = 1,



- suppose n nodes to embed
- assume numbered 1 ... n
- let $f_1, f_2 \cdots f_n \in R$ be embeddings.
- pack them into vector

$$\vec{f} = \begin{bmatrix} f_1 \\ \vdots \\ f_n \end{bmatrix}$$

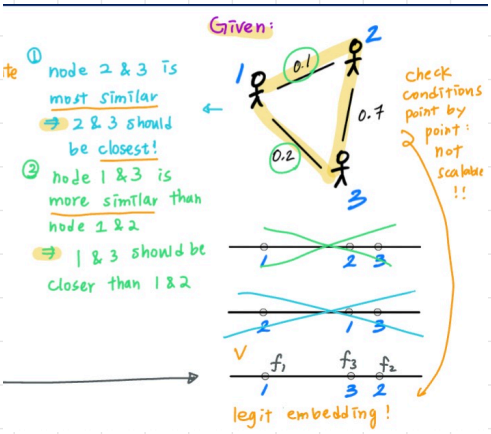
- goal: find a good set of embeddings $\vec{f}$.

⇒ Optimization Problem:

- cost is low if similar points are close.
  (point 2 & 3 are similar if $(f_2 - f_3)^2$ small)
  $w_{ij} \approx 1$

$$cost(\vec{f}) = \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij} (f_i - f_j)^2$$

where w_{ij} is weight between i and j in similarity matrix.



- if $w_{ij} \approx 0$, $(f_i - f_j)^2$ can be large.
- if $w_{ij} \approx 1$, $(f_i - f_j)^2$ should be small.

goal: $\min_{\vec{f}} cost(\vec{f})$

problem: cost is always minimized when $\vec{f} = 0$.

or $\vec{f} = \frac{1}{\sqrt{2}}(1, \cdots, 1)^T$ (if $\|\vec{f}\| = 1$)

Better optimization problem:

$$cost(\vec{f}) = \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij} (f_i - f_j)^2$$, given $n \times n$ similarity matrix W

subject to $\|\vec{f}\| = 1$ and $\vec{f} \perp (1, \cdots, 1)^T$

Define $L = D - W$, (L is the Graph Laplacian matrix).

where D is the degree matrix, $D_{n \times n}$ is diagonal
W is similarity matrix, $W_{n \times n}$
$L_{n \times n}$

Claim:

$$cost(\vec{f}) = \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij} (f_i - f_j)^2 = 2 \vec{f}^T L \vec{f}$$

proof:

$$(f_i - f_j)^2 = f_i^2 - 2 f_i f_j + f_j^2$$

$$\sum_{ij} w_{ij} (f_i - f_j)^2 = \sum_{ij} w_{ij} f_i^2 + \sum_{ij} w_{ij} f_j^2 - 2 \sum_{ij} w_{ij} f_i f_j$$

$$\sum_{ij} w_{ij} f_i^2 = \sum_i f_i^2 \sum_j w_{ij} = \sum_i d_i f_i^2 \quad (d_i = \sum_j w_{ij}, \text{ which is the degree}).$$

$$\sum_{ij} w_{ij} f_j^2 = \sum_j d_j f_j^2 \Leftrightarrow \sum_i d_i f_i^2 .$$

so, $2\sum_i d_i f_i^2 - 2\sum_{ij} w_{ij} f_i f_j = 2\left(\sum_i d_i f_i^2 - \sum_{ij} w_{ij} f_i f_j\right)$.

Laplacian matrix $L = D - W$, $= 2\vec{f}^T L \vec{f}$.

Note that:

$$\text{degree}(i) = \sum_{j=1}^{n} w_{ij} = \text{sum of weights associated with node } i$$

Degree matrix

$$D = \begin{bmatrix} d_{11} & & \\ & \ddots & \\ & & d_{nn} \end{bmatrix},$$

$$d_{ii} = \text{degree}(i) = \sum_{j=1}^{n} w_{ij}$$

$$\text{ith-row} \rightarrow \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1n} \\ \vdots & \vdots & & \vdots \\ w_{i1} & w_{i2} & \cdots & w_{in} \\ \vdots & \vdots & & \vdots \end{bmatrix}$$

Our optimization becomes:

$$\min_{\vec{f}} \text{cost}(\vec{f}) = \vec{f}^T L \vec{f}, \quad L = D - W, \quad (L \text{ symmetric}).$$

Subject to $\|\vec{f}\| = 1$ & $\vec{f} \perp (1 \cdots 1)^T$

By ideas of PCA and also resolve the constraints $\vec{f} \perp (1 \cdots 1)^T$.

Solution: choose $\vec{f}$ to the eigenvector of L with smallest eigenvalue > 0 (orthogonal to $(1, \cdots 1)^T$ by spectral Thm).

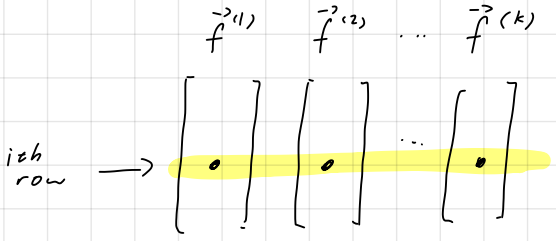
② $k > 1$,

Find bottom k eigenvectors with eigenvalues > 0.

Thus, with k eigenvector $\vec{f}^{(1)} \cdots \vec{f}^{(k)}$, each node is mapped to a point in $\mathbb{R}^k$.

so, consider node i:

new coordinate $f_i^{(1)}, f_i^{(2)} \cdots f_i^{(k)}$



embedding for node i: $z^{(i)} \in \mathbb{R}^k$

(Laplacian Eigenmaps).

Practical Issue:

$L_{norm} = D^{-1/2} L D^{-1/2}$.

where $D^{-1/2}$ is the diagonal matrix whose ith diagonal entry is $1/\sqrt{d_{ii}}$.

Normalize laplacian to find eigenvectors.

Supervised Learning

Example: given others movie rating → predict your rating of the movie.

$\vec{x} = (x_1, x_2, x_3, x_4, x_5)$

$H(\vec{x})$ = prediction.

↑ prediction function.

Regression → number

classification → class labels.

Empirical Risk Minimization (ERM).

① choose a hypothesis class

② choose a loss function

③ minimize expected loss (empirical risk).

Hypothesis class H.

(a set of possible prediction functions).

$H(\vec{x})$ ↑ prediction function.

H: a set of $H(\vec{x})$

- H := linear function
- H := decision tree
- H := neural nets

...

↓

The more complex the hypothesis class, the greater the danger of overfitting

↓

Occam's Razor: assume H is simple

↓

Assume linear prediction function:

$$H(\vec{x}) = w_0 + w_1 x_1 + w_2 x_2 + \cdots \quad \text{(parameterization)}.$$

$$\vec{w} = (w_0, \cdots)^T.$$

w_0.

- If there are d features, there are $d+1$ parameters.

$$H(\vec{x}) = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_d x_d.$$

$$= w_0 + \sum_{i=1}^{d} w_i x_i$$

Augmented feature vector $\longrightarrow$ $Aug(\vec{x})$.

$$\vec{x} = \begin{pmatrix} x_1 \\ \vdots \\ x_d \end{pmatrix}, \quad Aug(\vec{x}) = \begin{pmatrix} 1 \\ x_1 \\ \vdots \\ x_d \end{pmatrix} \in \mathbb{R}^{d+1}$$

$\Rightarrow$ we can write $H(\vec{x})$ as $H(\vec{x}) = Aug(\vec{x}) \cdot \vec{w}$.

- The surface of a prediction function is a hyperplane

Loss function

quantifies how wrong a single prediction is ...

$L(H(\vec{x}^{(i)}), y_i)$

$H(\vec{x}^{(i)})$: prediction of ith data. y_i: ground truth.

Optimize s.t. $H(\vec{x}^{(i)}) \approx y_i$

example loss function:

① Absolute loss: $|H(\vec{x}^{(i)}) - y_i|$

② Square loss: $(H(\vec{x}^{(i)}) - y_i)^2$.

A good H is good on average over all data.

Expected loss (empirical risk):

average loss across all data

$$R(H) = \frac{1}{n}\sum_{i=1}^{n} L(H(\vec{x}^{(i)}), y_i)$$

$\{\vec{x}^{(i)}, y_i\}_i$.

ex) Expected square loss (MSE)

$$Rsq(H) = \frac{1}{n}\sum_{i=1}^{n} (H(\vec{x}^{(i)}) - y_i)^2$$

Minimizing Expected Loss

Find H minimizing

$$Rsq(H) = \frac{1}{n}\sum_{i=1}^{n} (H(x^{(i)}) - y_i)^2$$

↓

find $\vec{w}^*$ $\min_{\vec{w}} Rsq = \min_{\vec{w}} \frac{1}{n}\sum_{i=1}^{n} (\vec{w}\cdot Aug(\vec{x}^{(i)}) - y_i)^2$

↓

closed form solution: (MSE)

features $x_1 \sim x_d$.

$$\vec{w}^* = (X^T X)^{-1} X^T \vec{y}$$

X is design matrix

$$X = \begin{pmatrix} Aug(\vec{x}^{(1)}) \\ \vdots \\ Aug(\vec{x}^{(n)}) \end{pmatrix} = \begin{pmatrix} 1 & x_1^{(1)} & x_2^{(1)} & \cdots & x_d^{(1)} \\ 1 & x_1^{(2)} & x_2^{(2)} & \cdots & x_d^{(2)} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_1^{(n)} & x_2^{(n)} & \cdots & x_d^{(n)} \end{pmatrix}$$

data points $x^{(1)} \cdots x^{(n)}$

Classification vs. Regression

Regression → number

Classification → class label

linear classifier from regressor:

↓

convert output to $\{-1, 1\}$ using

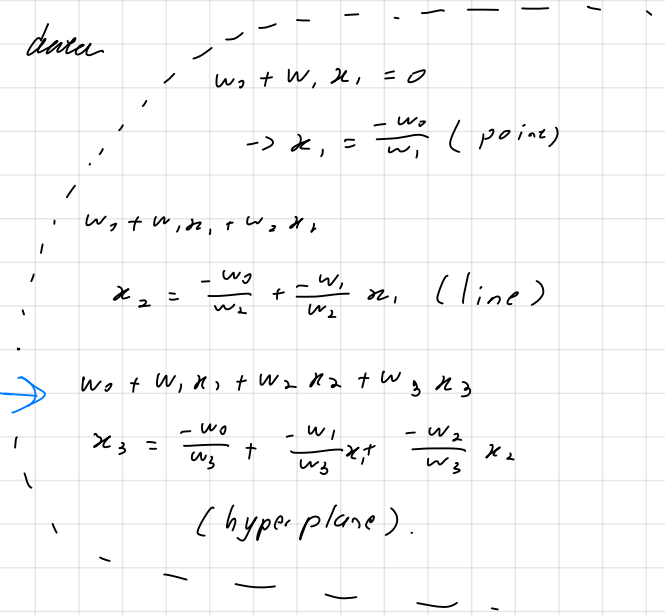
$$sign(z) = \begin{cases} 1 & z > 0 \\ -1 & z < 0 \\ 0 & \text{otherwise.} \end{cases}$$

prediction: $\text{sign}(H(\vec{x}))$.

Decision boundary is place to classify the data

↓

Decision boundary is generally a hyperplane. ($d-1$ dimension)



decison boundary derived from MSE

affected by outliers

Feature Mapping

Learn new representation by creating new features from old feature.

(non-linear pattern). $x \in R^d$.

Non-linear Basis function: $\vec{\varphi}(\vec{x}) = (\varphi_1(\vec{x}), \varphi_2(\vec{x}))$

$\varphi_1 \dots \varphi_d$.

↑ feature map

↓ a new representation.

$(\vec{z} = \vec{\varphi}(\vec{x}))$ by mapping each training data $\vec{x}^{(i)}$ to feature space.

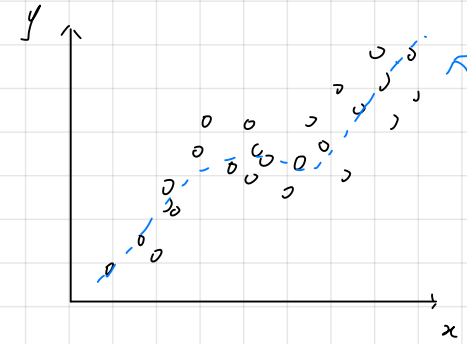
⇓

fit linear prediction func. H in feature space.

$H_f(\vec{z}) = w_0 + w_1 z_1 + \dots + w_d z_d$.

$= w_0 + w_1 \varphi_1(\vec{x}) + \dots + w_d \varphi_d(\vec{x})$.

ex).



how to find $H(\vec{x})$

↓

choose polynomial basis function.

↓

$x \to (x, x^2, x^3, x^4)$ ← new feature.

$\phi_1 \quad \phi_2 \quad \phi_3 \quad \phi_4$

new design matrix:

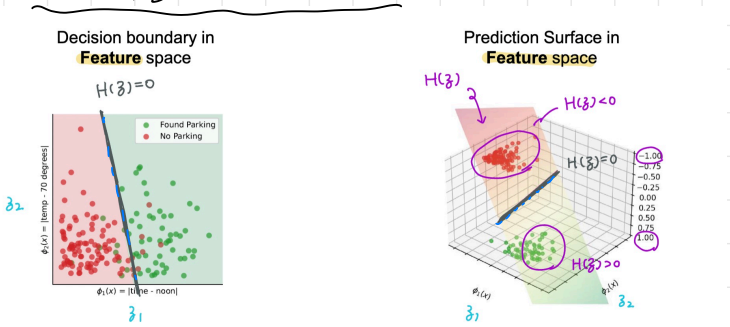
$$X = \begin{pmatrix} 1 & x_1 & x_1^2 & x_1^3 & x_1^4 \\ 1 & x_2 & x_2^2 & x_2^3 & x_2^4 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_n & x_n^2 & x_n^3 & x_n^4 \end{pmatrix}$$

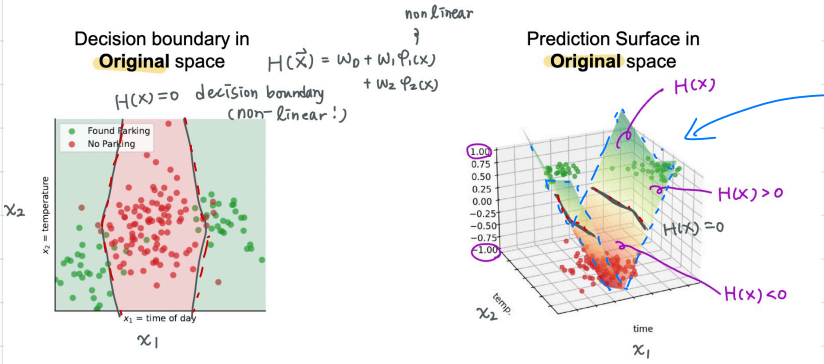
so, $\vec{w} = (X^T X)^{-1} X^T \vec{y}$

model $H(\vec{x}) = \vec{w} \cdot Aug(\vec{z})$

where $\vec{z} = \begin{bmatrix} \phi_1(\vec{x}) \\ \vdots \\ \phi_4(\vec{x}) \end{bmatrix}$.

Visualization Plot





$H(\vec{x}) = w_0 + w_1 \phi_1(x) + w_2 \phi_2(x)$ ← non linear

tweaking the surface of dareu via feature mapping.

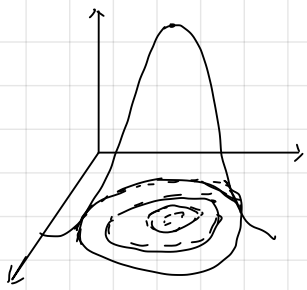
Radial Basis Function (RBF).

A generic basis function that works for many problems:

⇒ common choice:

gaussian basis function: $\varphi(\vec{x}; \vec{\mu}, \sigma) = e^{-\|\vec{x} - \vec{\mu}\|^2 / \sigma^2}$

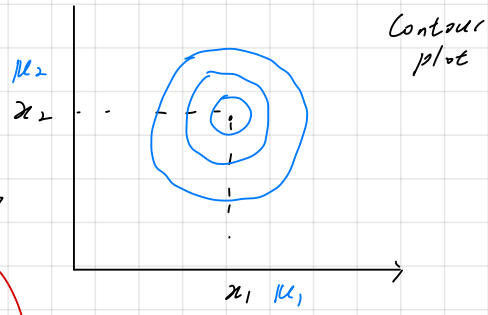
$\vec{\mu}$: center, σ: width



$\varphi(\vec{\mu}) = e^{-0} = 1$

$\varphi(\vec{\mu} \pm 6) = e^{-6^2/6^2} = e^{-1} = 0.369$

if $\|\vec{x} - \vec{\mu}\|^2 >> 1$, $e^{-\text{large positive \#}} = e^{-\infty} \to 0$



so
- if $\vec{x}$ close to $\vec{\mu}$, $\varphi(\vec{x}; \vec{\mu}, 6)$ is large.
- if $\vec{x}$ far from $\vec{\mu}$, $\varphi(\vec{x}; \vec{\mu}, 6)$ is small.

$\Rightarrow$ intuition: φ measures how "similar" $\vec{x}$ is to $\vec{\mu}$. (assume similar objects have close feature vectors).

Procedure:

① pick # of new features: d'.
pick centers, $\vec{\mu}^{(1)} \cdots \vec{\mu}^{(d')}$
pick width: $6_1 \cdots 6_{d'}$ (usually all same)

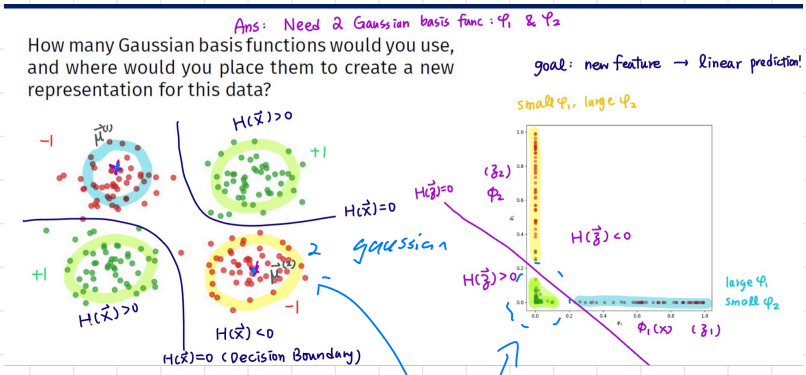
② $\Rightarrow$ define ith basis function: $\varphi_i(\vec{x}) = e^{-\|\vec{x} - \vec{\mu}^{(i)}\|^2 / 6_i^2}$.

so feature map $\vec{\varphi}(\vec{x}) = \begin{bmatrix} \varphi_1(\vec{x}) \\ \vdots \\ \varphi_{d'}(\vec{x}) \end{bmatrix}$ ← "similarity" of $\vec{x}$ to $\vec{\mu}^{(1)}$ ← "similarity" of $\vec{x}$ to $\vec{\mu}^{(d')}$.

③ $\Rightarrow$ Last, we train a linear classifier in this new representation:

$H(\vec{z}) = w_0 + w_1 z_1 + \cdots + w_{d'} z_{d'}$

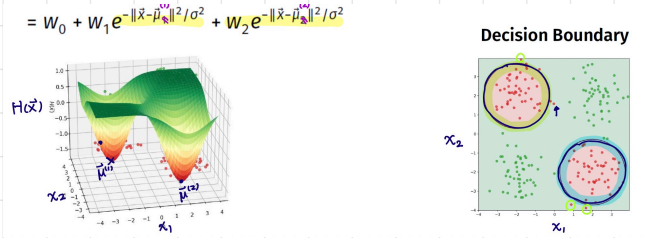
example)



place gaussian in the center (mean) of the data points (same class label have large "similarity").

$H(\vec{x})$ is sum of Gaussians:

$$H(\vec{x}) = w_0 + w_1 \varphi_1(\vec{x}) + w_2 \varphi_2(\vec{x})$$
$$= w_0 + w_1 e^{-\|\vec{x}-\vec{\mu}^{(1)}\|^2/\sigma^2} + w_2 e^{-\|\vec{x}-\vec{\mu}^{(2)}\|^2/\sigma^2}.$$



By increasing # of basis functions, we can make more complex decision surfaces.

RBF Network

Gaussians are ex) of radial basis func.

↓

each basis func. has center $\vec{c}$.

value depends only on distance from center.

$$\varphi(\vec{x}; \vec{c}) = f(\|\vec{x} - \vec{c}\|)$$

function to measure distance.

distance bcw input $\vec{x}$ & center $\vec{c}$

example RBF:

inverse multiquadratic

$$\varphi(\vec{x}, \vec{c}) = \frac{1}{\sqrt{1 + \varepsilon^2 \cdot \|\vec{x} - \vec{c}\|^2}}$$

RBF Network:

① choose basis function $\varphi_1, \cdots, \varphi_d$.

② Transform data to new representation:

$$\vec{x} \mapsto \left(\varphi_1(\vec{x}), \varphi_2(\vec{x}), \cdots, \varphi_d(\vec{x})\right)^T.$$

③ Train.

$$H(\vec{x}) = w_0 + w_1 \varphi_1(\vec{x}) + \cdots + w_{d'} \varphi_{d'}(\vec{x}).$$

RBF network has these parameters:

- parameter of each individual basis function.

- $\vec{\mu}$ (center)
- σ^2 (variance)
- ...

- weight associated to each "new" feature: w_i

Training:

- find parameters of RBFs first $\phi_i(\vec{\mu}^{(i)}, \sigma_i^2)$

↓ through optimization, clustering, randomly, ...

After fixing those parameters

- optimize w's $\vec{H}(\vec{x}; \vec{w})$

which is linear.

How to find parameters of RBFs?

Approaches

① Every data points as a center.

↳

- n basis function (one for each point).
- n features

$$\vec{\phi}(\vec{x}) = (\phi_1(\vec{x}), \phi_2(\vec{x}), \cdots, \phi_n(\vec{x}))^T.$$

problem: overfitting & computationally expensive. $\rightarrow \Theta(nd^2)$

if $d = n$

$\Rightarrow \Theta(n^3)$.

② Random sample

randomly choose k data points as center

problems: undersample / oversample a region.

③ Clustering

- group data points into clusters.
- cluster centers => RBFs.

Inference:

- given a point $\vec{x}$, map it to feature space $\vec{x} \mapsto (\varphi_1(\vec{x}) \cdots \varphi_k(\vec{x}))^T$.
- evaluated Hf in feature space. $Hf(\vec{z})$

K-means clustering for picking parameters of RBFs.

idea:

compress each clustering into a single point while minimizing information loss.

- Given: data $\{x^{(i)}\} \in R^d$ and a parameter k.
- Find: k clusters center, $\vec{\mu}^{(1)} \cdots \vec{\mu}^{(k)}$ so that

average square distance from a data point to nearest cluster center is small

$$\text{Cost}(\vec{\mu}^{(1)} \cdots \vec{\mu}^{(k)}) = \frac{1}{n} \sum_{i=1}^{n} \min_{j \in \{1..k\}} \| \vec{x}^{(i)} - \vec{\mu}^{(j)} \|^2$$

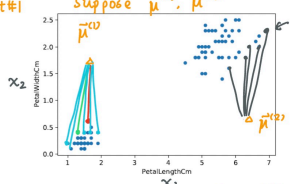
(nearest center; index of data; index of cluster center.)

optimize cost

Which cost is smaller? cost #1 or #2?

cost #2 < cost #1!

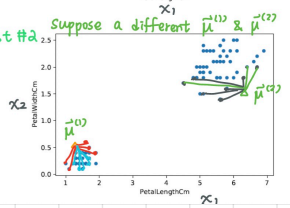
cost #1 Suppose $\vec{\mu}^{(1)}$, $\vec{\mu}^{(2)}$



larger cost

cost #2 Suppose a different $\vec{\mu}^{(1)}$ & $\vec{\mu}^{(2)}$

x of cluster center



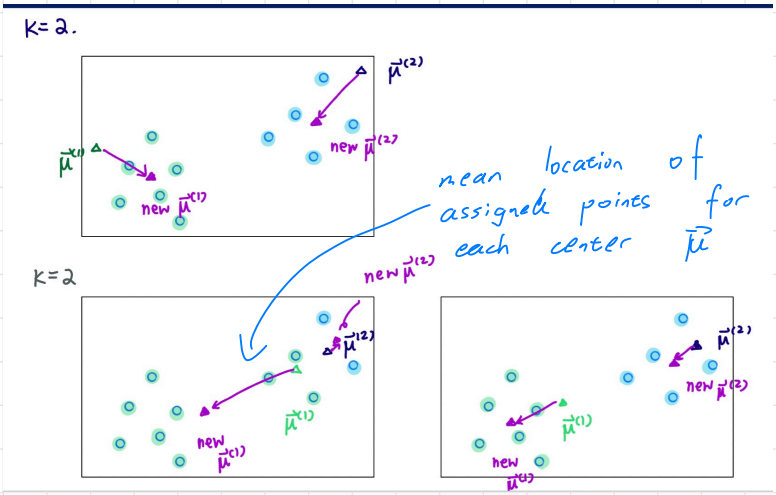
smaller cost.

Lloyds algorithm for K-means:

① initialize centers $\vec{\mu}^{(1)} \cdots \vec{\mu}^{(k)}$

② repeat until convergence:

- assign each point $\vec{x}^{(i)}$ to closet center.
- update each $\vec{\mu}^{(i)}$ as the mean of points assigned to it.



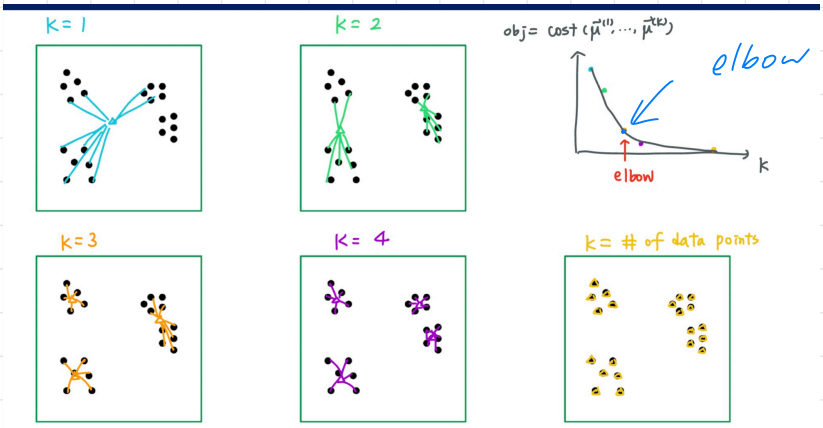
How to choose k ?

Increase k always decrease objective function

↓

Elbow Method:

- run k-means repeatedly with increasing values of k.
- plot the values of the objective as function of k.
- find elbow in plot.



-> good value of k.

RBF network:

- k-mean clustering to find center
- create new features using k RBFs.
- least square classifier.

There is other parameter of RBFs

→ choose via cross validation.

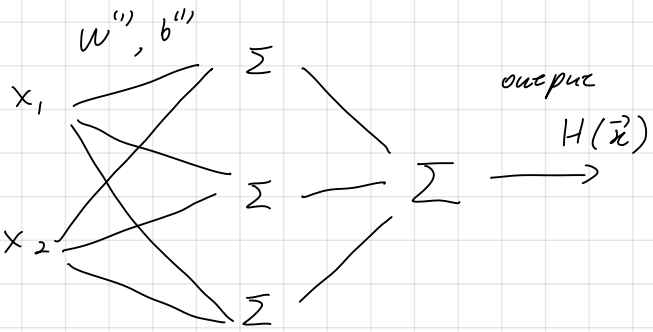
Neural Network

Neurons are organized into layers.

↓

input layers
output layers
hidden layers

ex)



$$\vec{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}.$$

input dim, hidden dim.

$$W^{(1)} = \begin{pmatrix} 2 & -1 & 0 \\ 4 & 5 & 2 \end{pmatrix} \in R^{2\times 3}$$

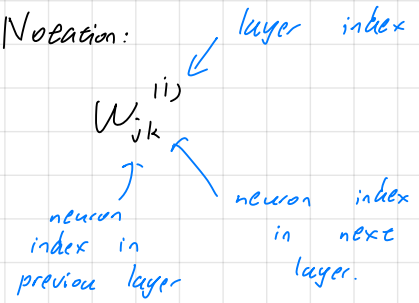
$$\vec{b}^{(1)} = (3, -2, -2)^{-1} \in R^3$$

hidden dim, output dim.

$$W^{(2)} = \begin{pmatrix} 3 \\ 2 \\ -4 \end{pmatrix} \in R^{3\times 1}$$

$$\vec{b}^{(2)} = (-4)^T \in R^1$$

Notation:



- # of cells in input layer determined by dim. of input feature vector.
- # of cells in hidden layer determined by you.
- output layer can have >1 neurons.

These are "fully-connected, feed-forward!" networks with one input.

↓

They are function $H(\vec{x}): \mathbb{R}^n \to \mathbb{R}^1$.

↓

forward pass: compute layer (i), use as input for layer i+1.

- let $z_j^{(i)}$ be output of node j in layer i.
- outputs: $\vec{z}^{(i)} = (z_1^{(i)}, z_2^{(i)}, \cdots)^T$
- $\vec{z}^{(i)} = [w^{(i)}]^T \vec{z}^{(i-1)} + \vec{b}^{(i)}$.

Each layer is a function:

ex) $\cdot H^{(1)}(\vec{x}) = [w^{(1)}]^T \vec{x} + \vec{b}^{(1)}$

$H^{(1)}: \mathbb{R}^2 \to \mathbb{R}^3$

$\cdot H^{(2)}(\vec{z}) = [w^{(2)}]^T \vec{z} + \vec{b}^{(2)}$

$H^{(2)}: \mathbb{R}^3 \to \mathbb{R}^1$

$$H(\vec{x}) = H^{(2)}(H^{(1)}(\vec{x}))$$
$$= \underbrace{w^{(2)T} w^{(1)T}}_{\downarrow \tilde{w}^T} \vec{x} + \underbrace{w^{(2)T} b^{(1)} + b^{(2)}}_{\downarrow \tilde{b}}$$
$$= \tilde{w}^T \vec{x} + \tilde{b}$$

$$H(\vec{x}) = [w^{(2)}]^T \left([w^{(1)}]^T \vec{x} + \vec{b}^{(1)} \right) + \vec{b}^{(2)} = \vec{w} \cdot Aug(\vec{x}).$$

since NN are linear.

Non-linearity → Activation Function

↓

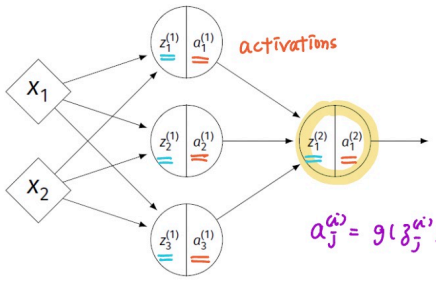
$a_j^{(i)} = g(z_j^{(i)})$ the actual output of neurons.

↑

activation function $g(\cdot)$

$a_j^{(i)}$ ← layer index
↖ neuron index

(non-linear NN)



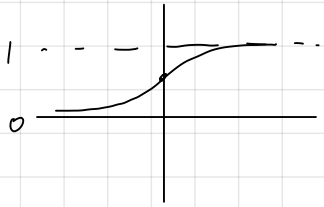
$a_j^{(i)} = g(z_j^{(i)})$

- $z_j^{(i)}$ is the linear activation before g is applied.
- $a_j^{(i)} = g(z_j^{(i)})$ is the actual output of the neuron.

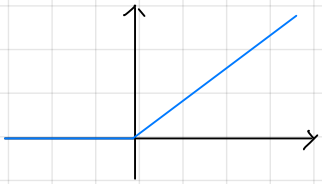
$g(\cdot)$: activation function (sometimes $\sigma(\cdot)$)

example of activation function:

- $\sigma(z) = \frac{1}{1+e^{-z}}$ sigmoid



- $g(z) = \max\{0, z_j\} = \begin{cases} z, & z \geq 0 \\ 0, & z < 0. \end{cases}$

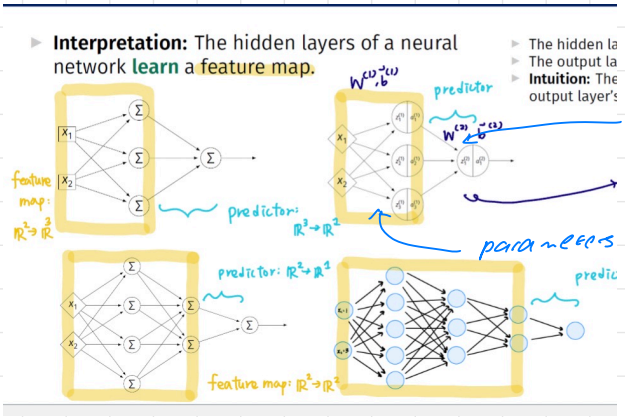


Activation of output neurons can be different than activation of hidden neurons.

Interpretation from feature map:

↓

hidden layer of a neural networks learn a feature map.



parameters for prediction.

update together.

parameters for feature map

Training:

Empirical risk minimization.

- a training set $\{(\vec{x}^{(i)}, y_i\}$.
- initialize prediction function H.

· pick loss function.

· minimize risk w.r.t. loss.

↓

gradient descent.

$$\nabla_{\vec{w}} R(\vec{w}) = \left(\partial R/\partial w_0, \partial R/\partial w_1 \cdots, \partial R/\partial w_k \right)^T.$$

loss

weighted all data points

prediction. ground truth

$$R_{sq}(\vec{w}) = \frac{1}{n} \sum_{i=1}^{n} \left(H(\vec{x}^{(i)}) - y_i \right)^2 = \frac{1}{n} \sum_{i=1}^{n} \left(Aug(\vec{x}^{(i)}) \cdot \vec{w} - y_i \right)^2.$$

↓ w.r.t. loss

$$\nabla_{\vec{w}} R(\vec{w}) = \frac{1}{n} \sum_{i=1}^{n} \frac{\partial \ell}{\partial H} \nabla_{\vec{w}} H(\vec{x}; \vec{w})$$

$\nabla_{\vec{w}} H$ = $\begin{bmatrix} \partial H/\partial w_{11}^{(1)} \\ \vdots \\ \partial H/\partial w_{21}^{(3)} \end{bmatrix}$ w.r.t. all weights of all layers.

↓ chain-rule.

What is ∇H?

suppose H is neural net with activation

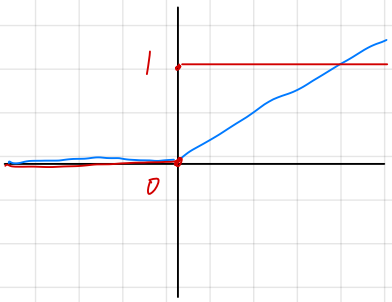
ex) $H(\vec{x}, \vec{w}) = \sigma(w^{(3)T} \sigma(w^{(2)T} \sigma(w^{(1)T} x + b^{(1)}) + b^{(2)}) + b^{(3)})$.

↓ chain rule?

note: often useful to pack all weights into a parameter vector $\vec{w}$.

$$\vec{w} = (w_{11}^{(1)}, w_{12}^{(1)} \cdots, b_1^{(1)}, b_2^{(1)}, w_{11}^{(2)}, w_{12}^{(2)} \cdots b_1^{(2)}, b_2^{(2)})$$

Aside: derivative of RELU.



$$g(z) = \max\{0, z\}$$

$$g'(z) = \begin{cases} 0, & z \le 0 \\ 1, & z > 0 \end{cases}$$

chain rule:

$$f(g(x)), \quad \frac{d}{dx} f(g(x)) = f'(g(x)) g'(x)$$

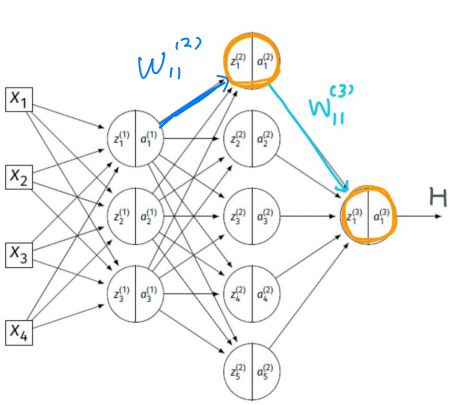
ex) $H = a_1^{(2)} = g(z_1^{(2)})$

↓

$$z_1^{(2)} = w_{11}^{(2)} \cdot a_1^{(1)} + w_{21}^{(2)} \cdot a_2^{(1)}$$

$$= w_{11}^{(2)} g(w_{11}^{(1)} x_1 + w_{21}^{(1)} x_2 + w_{31}^{(1)} x_3) + w_{21}^{(2)} g(w_{12}^{(1)} x_1 + w_{22}^{(1)} x_2 + w_{32}^{(1)} x_3)$$

$$\frac{\partial H}{\partial w_{11}^{(1)}} = g'(z_1^{(2)}) \frac{\partial z_1^{(2)}}{\partial w_{11}^{(1)}} \left(w_{11}^{(2)} g'(z_1^{(1)}) \cdot x_1 \right)$$



① last layer:

$$z_1^{(3)} = w_{11}^{(3)} a_1^{(2)} + \dots$$

$$\frac{\partial H}{\partial w_{11}^{(3)}} = g'(z_1^{(3)}) \frac{\partial z_1^{(3)}}{\partial w_{11}^{(3)}} = g'(z_1^{(3)}) \cdot a^{(2)}$$

② Second last layer:

$$\frac{\partial H}{\partial w_{11}^{(2)}} = g'(z_1^{(3)}) \frac{\partial z_1^{(3)}}{\partial w_{11}^{(2)}} = g'(z_1^{(3)}) \cdot w_{11}^{(3)} \cdot g'(z_1^{(2)}) \cdot a_1^{(1)}$$

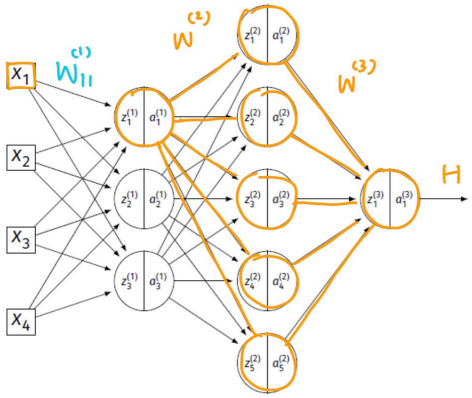
$$z_1^{(3)} = w_{11}^{(3)} a_1^{(2)} + \dots$$

$$\frac{\partial a_1^{(2)}}{\partial w_{11}^{(2)}}$$

$$a_1^{(2)} = g(z_1^{(2)}) = g(w_{11}^{(2)} a_1^{(1)}) + \dots$$

③ First layer

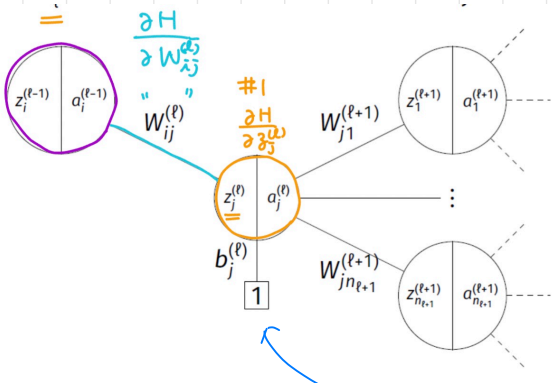
↓ depend upon all terms in orange.



Back propagation:

Consider arbitary node in layer "ℓ" of a NN.

- Let g be the activation function.
- n_ℓ denotes the # of nodes in layer ℓ.



- $$\frac{\partial H}{\partial w_{ij}^{(\ell)}} = \frac{\partial H}{\partial z_j^{(\ell)}} \cdot \frac{\partial z_j^{(\ell)}}{\partial w_{ij}^{(\ell)}} = \frac{\partial H}{\partial z_j^{(\ell)}} a_i^{(\ell-1)}$$

since $z_j^{(\ell)} = w_{ij}^{(\ell)} a_i^{(\ell-1)} + \sum_{k \neq j} w_{kj}^{(\ell)} a_k^{(\ell-1)}$

- $$\frac{\partial H}{\partial b_j^{(\ell)}} = \frac{\partial H}{\partial z_j^{(\ell)}} \cdot \frac{\partial z_j^{(\ell)}}{\partial b_j^{(\ell)}} = \frac{\partial H}{\partial z_j^{(\ell)}}$$

$\frac{\partial z_j^{(\ell)}}{\partial b_j^{(\ell)}} = 1$

b/c

Claim #1

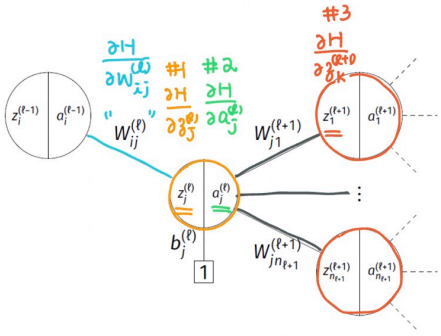
$$\frac{\partial H}{\partial W_{ij}^{(\ell)}} = \frac{\partial H}{\partial z_j^{(\ell)}} a_i^{(\ell-1)}$$

Claim #2

$$\frac{\partial H}{\partial z_j^{(\ell)}} = \frac{\partial H}{\partial a_j^{(\ell)}} g'(z_j^{(\ell)})$$

Claim #3

$$\frac{\partial H}{\partial a_j^{(\ell)}} = \sum_{k=1}^{n_{\ell+1}} \frac{\partial H}{\partial z_k^{(\ell+1)}} W_{jk}^{(\ell+1)}$$



$$a_j^{(\ell)} = g(z_j^{(\ell)})$$

$$\frac{\partial H}{\partial z_j^{(\ell)}} = \frac{\partial H}{\partial a_j^{(\ell)}} \frac{\partial a_j^{(\ell)}}{\partial z_j^{(\ell)}} = g'(z_j^{(\ell)})$$

Now, how to calculate $\frac{\partial H}{\partial a_j^{(\ell)}}$?

$$H = \phi_1(z_1^{(\ell+1)}) + \phi_2(z_2^{(\ell+1)}) + \cdots + \phi_{n_{\ell+1}}(z_{n_{\ell+1}}^{(\ell+1)})$$

$$\to W_{j1}^{(\ell+1)} \cdot a_j^{(\ell)} \quad \to W_{j2}^{(\ell+1)} \cdot a_j^{(\ell)}$$

$$\frac{\partial H}{\partial a_j^{(\ell)}} = \frac{\partial H}{\partial z_1^{(\ell+1)}} \frac{\partial z_1^{(\ell+1)}}{\partial a_j^{(\ell)}} + \frac{\partial H}{\partial z_2^{(\ell+1)}} \frac{\partial z_2^{(\ell+1)}}{\partial a_j^{(\ell)}} + \cdots$$

$$= \sum_{k=1}^{n_{\ell+1}} \frac{\partial H}{\partial z_k^{(\ell+1)}} W_{jk}^{(\ell+1)} \quad \#$$

- Derivative in layer ℓ depends on derivatives in layer $(\ell+1)$.

↓ recursive formulas:

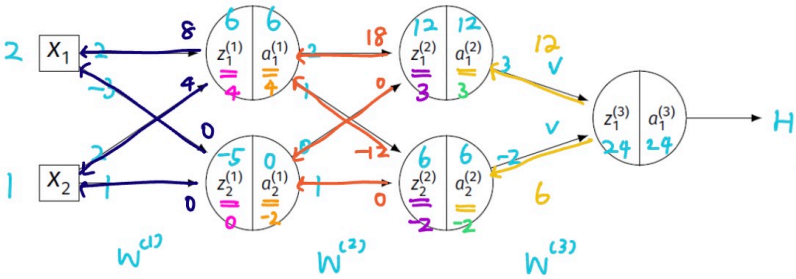
- given input $\vec{x}$ and a current parameter vector $\vec{w}$.
  - evaluate the network to compute $z_i^{(\ell)}$ and $a_i^{(\ell)}$ for all nodes.
  - for each layer ℓ from last to first.

- $\partial H / \partial a_j^{(\ell)} = \sum_{k=1}^{n_{\ell+1}} \frac{\partial H}{\partial z_k^{(\ell+1)}} W_{jk}^{(\ell+1)}$ ← from last layer / next layer.
- $\partial H / \partial z_j^{(\ell)} = \frac{\partial H}{\partial a_j^{(\ell)}} g'(z_j^{(\ell)})$
- $\partial H / \partial w_{ij}^{(\ell)} = \frac{\partial H}{\partial z_j^{(\ell)}} a_i^{(\ell-1)}$
- $\partial H / \partial b_j^{(\ell)} = \frac{\partial H}{\partial z_j^{(\ell)}}$

example).

Compute the entries of the gradient given: Suppose no bias

$W^{(1)} = \begin{pmatrix} 2 & -3 \\ 2 & 1 \end{pmatrix}$ $W^{(2)} = \begin{pmatrix} 2 & 1 \\ 0 & 1 \end{pmatrix}$ $W^{(3)} = \begin{pmatrix} 3 \\ -2 \end{pmatrix}$ $\vec{x} = (2, 1)^T$ $g(z) = \text{ReLU}$



$\frac{\partial H}{\partial a_j^{(\ell)}} = \sum_{k=1}^{n_{\ell+1}} \frac{\partial H}{\partial z_k^{(\ell+1)}} W_{jk}^{(\ell+1)}$ $\frac{\partial H}{\partial z_j^{(\ell)}} = \frac{\partial H}{\partial a_j^{(\ell)}} g'(z_j^{(\ell)})$ $\frac{\partial H}{\partial W_{ij}^{(\ell)}} = \frac{\partial H}{\partial z_j^{(\ell)}} a_i^{(\ell-1)}$

Q: what's $\frac{\partial H}{\partial \vec{W}}$?

① $\frac{\partial H}{\partial z_j^{(3)}} = \frac{\partial H}{\partial a_j^{(3)}} g'(z_1^{(3)}) = 1 \cdot 1 = 1$

$\left\{ \frac{\partial H}{\partial a_j^{(3)}} = 1, \; z_1^{(3)} = 24 \rightarrow g'(24) = 1 \right.$

② $\frac{\partial H}{\partial W_{ij}^{(3)}} = \underbrace{\frac{\partial H}{\partial z_j^{(3)}}}_{①} a_i^{(2)}$

$\left\{ \begin{array}{l} \frac{\partial H}{\partial W_{1j}^{(3)}} = 1 \cdot a_1^{(2)} = 1 \cdot 12 = 12 \\ \frac{\partial H}{\partial W_{2j}^{(3)}} = 1 \cdot a_2^{(2)} = 1 \cdot 6 = 6 \end{array} \right.$

③ $\frac{\partial H}{\partial a_j^{(2)}} = \sum_{k=1}^{n_3} \frac{\partial H}{\partial z_k^{(3)}} W_{jk}^{(3)} = W_{j1}^{(3)}$

$\{ n_3 = 1, \; \frac{\partial H}{\partial z_k^{(3)}} = 1$

$\left\{ \frac{\partial H}{\partial a_1^{(2)}} = W_{11}^{(3)} = 3, \; \frac{\partial H}{\partial a_2^{(2)}} = W_{21}^{(3)} = -2 \right.$

---

④ $\frac{\partial H}{\partial z_j^{(2)}} = \underbrace{\frac{\partial H}{\partial a_j^{(2)}}}_{③} g'(z_j^{(2)}) = \begin{cases} \frac{\partial H}{\partial a_j^{(2)}}, & \text{if } z_j^{(2)} > 0 \\ 0 & \text{o.w.} \end{cases}$

$\frac{\partial H}{\partial z_1^{(2)}} = \frac{\partial H}{\partial a_1^{(2)}} = 3$

$\frac{\partial H}{\partial z_2^{(2)}} = \frac{\partial H}{\partial a_2^{(2)}} = -2$

⑤ $\frac{\partial H}{\partial W_{ij}^{(2)}} = \underbrace{\frac{\partial H}{\partial z_j^{(2)}}}_{④} \cdot a_i^{(1)}$

$\left\{ \begin{array}{l} \frac{\partial H}{\partial W_{11}^{(2)}} = \frac{\partial H}{\partial z_1^{(2)}} \cdot a_1^{(1)} = 3 \cdot 6 = 18 \\ \frac{\partial H}{\partial W_{12}^{(2)}} = \frac{\partial H}{\partial z_2^{(2)}} \cdot a_1^{(1)} = -2 \cdot 6 = -12 \\ \frac{\partial H}{\partial W_{21}^{(2)}} = \frac{\partial H}{\partial z_1^{(2)}} a_2^{(1)} = 3 \cdot 0 = 0 \\ \frac{\partial H}{\partial W_{22}^{(2)}} = \frac{\partial H}{\partial z_2^{(2)}} \cdot a_2^{(2)} = -2 \cdot 0 = 0 \end{array} \right.$

---

⑥ $\frac{\partial H}{\partial a_j^{(1)}} = \sum_{k=1}^{n_2} \frac{\partial H}{\partial z_k^{(2)}} W_{jk}^{(2)} = \underbrace{\frac{\partial H}{\partial z_1^{(2)}}}_{④} W_{j1}^{(2)} + \underbrace{\frac{\partial H}{\partial z_2^{(2)}}}_{④} W_{j2}^{(2)} = 3 \cdot W_{j1}^{(2)} + (-2) \cdot W_{j2}^{(2)}$

$\{ n_2 = 2$ (④) = 3, (④) = −2

$\left\{ \begin{array}{l} \frac{\partial H}{\partial a_1^{(1)}} = 3 \cdot W_{11}^{(2)} + (-2) \cdot W_{12}^{(2)} = 3 \cdot 2 + (-2) \cdot 1 = 4 \\ \frac{\partial H}{\partial a_2^{(1)}} = 3 \cdot W_{21}^{(2)} + (-2) \cdot W_{22}^{(2)} = 3 \cdot 0 + (-2) \cdot 1 = -2 \end{array} \right.$

⑦ $\frac{\partial H}{\partial z_j^{(1)}} = \underbrace{\frac{\partial H}{\partial a_j^{(1)}}}_{⑥} g'(z_j^{(1)})$

$\left\{ \begin{array}{l} \frac{\partial H}{\partial z_1^{(1)}} = \underbrace{\frac{\partial H}{\partial a_1^{(1)}}}_{4} \underbrace{g'(z_1^{(1)})}_{1 \; (\because z_1^{(1)} > 0)} = 4 \\ \frac{\partial H}{\partial z_2^{(1)}} = \underbrace{\frac{\partial H}{\partial a_2^{(1)}}}_{-2} \cdot \underbrace{g'(z_2^{(1)})}_{0 \; (\because z_2^{(1)} < 0)} = 0 \end{array} \right.$

⑧ $\frac{\partial H}{\partial W_{ij}^{(1)}} = \frac{\partial H}{\partial z_j^{(1)}} \cdot a_i^{(0)}$

$= \underbrace{\frac{\partial H}{\partial z_j^{(1)}}}_{(⑦)} \cdot x_i$

$\{ a^{(0)} = \vec{x} = \text{input}$
$a_i^{(0)} = x_i$

$\left\{ \begin{array}{l} \frac{\partial H}{\partial W_{11}^{(1)}} = \frac{\partial H}{\partial z_1^{(1)}} \cdot x_1 = 4 \cdot 2 = 8 \\ \frac{\partial H}{\partial W_{12}^{(1)}} = \frac{\partial H}{\partial z_2^{(1)}} \cdot x_1 = 0 \cdot 2 = 0 \\ \frac{\partial H}{\partial W_{21}^{(1)}} = \frac{\partial H}{\partial z_1^{(1)}} \cdot x_2 = 4 \cdot 1 = 4 \\ \frac{\partial H}{\partial W_{22}^{(1)}} = \frac{\partial H}{\partial z_2^{(1)}} \cdot x_2 = 0 \cdot 1 = 0 \; \text{※} \end{array} \right.$

idea: $f(g(x))$

goal: $\frac{df}{dx} = \frac{df}{dg}\frac{dg}{dx}$.

input $x \longrightarrow g \longrightarrow f$ output $f(g(x))$

Backpropagation via gradient descent.

gradient of loss w.r.t. the weights.

↓

by iterative optimization.

$f: \mathbb{R}^d \to \mathbb{R}^k$.

gradient of f at $\vec{x}$

$$\vec{\nabla} f(\vec{x}) = \left(\frac{\partial f}{\partial x_1}(\vec{x}) \cdots \frac{\partial f}{\partial x_d}(\vec{x})\right)^T$$

$$\vec{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_d \end{bmatrix} \in \mathbb{R}^d.$$

↓ also a function mapping from $\mathbb{R}^d \to \mathbb{R}^k$.

idea from taylor expansion:

$$f(x_0 + z_x, y_0 + z_y) \approx f(x_0, y_0) + \vec{z} \cdot \vec{\nabla} f(x_0, y_0).$$

① ② ③

① < ② $\Rightarrow$ ③ should be negative:

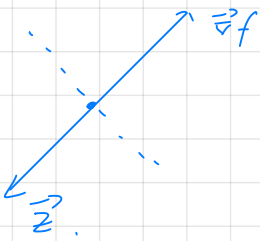
$\Rightarrow \vec{z} \cdot \vec{\nabla} f(x_0, y_0) < 0$.

which $\vec{z}$ can we decrease the value of ② the most?

$\Rightarrow \vec{z} \approx -\vec{\nabla} f$ b/c

↓

this is direction of steepest descent at $\vec{x}$.



$\|\vec{\nabla} f(\vec{x})\|$ measures steepness of ascent.

- direction orthogonal to $\vec{\nabla} f(\vec{x})$, f does not change!

So gradient descent as follow:

pick learning rate $\eta > 0$.

repeat until convergence.

$$\vec{x}^{(i+1)} = \vec{x}^{(i)} - \eta \vec{\nabla} f(\vec{x}^{(i)}).$$

↑ steepest descent direction.

stop when $\| \vec{x}^{(i)} - \vec{x}^{(i+1)} \|$ is small.

So, in Neural Network Training.

consider square loss:

$$\nabla_{\vec{w}} R(\vec{w}) = \frac{1}{n} \sum_{i=1}^{n} \frac{\partial \ell}{\partial H} \nabla_{\vec{w}} H(\vec{x}^{(i)}; \vec{w})$$

$$= \frac{2}{n} \sum_{i=1}^{n} (H(\vec{x}^{(i)}) - y_i) \nabla_{\vec{w}} H(\vec{x}^{(i)}; \vec{w}).$$

↓ continue backprop the errors through the layers.

Difficulties of training NN:

- when activation are non-linear, risk of NN. is highly non-convex.
- vanishing gradient problems

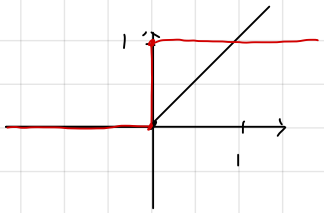
↓ gradient of weight $\frac{\partial H}{\partial w}$ at earlier layers can be very small

↓ result

the update of w is very slow by gradient descent.

↓ b/c gradient accumulated by multiplication

one mitigation: use ReLU instead of sigmoid:



- computational cost is high.

↓ possible solution

Batch learning

for batch size $m << n$ the # data points.

↓

compute stochastic gradient based on the batch of data points (m).

$$\nabla R(\vec{w}) = \sum_{i \in B} \vec{\nabla} \ell(H(\vec{x}^{(i)}; \vec{w}), y_i)$$

batch

pro: fase to compute

con: approximation of full gradient is noisy.

Convolution Neural Networks

↓

image recognition by line detector

shape detector

...

face detector

↓

Convolution with an edge filter.

(detect edges in the image)

vertical / horizontal / diagonal.

example)

0 0 255
0 0 255
0 0 255

(*) => result (255 x 3)

0 -1 1
0 -1 1
0 -1 1

↑ ↑

detect transition (from no edge to edge)

example)

move the filter over the entire image, repeat procedure.

0	0	0	0	0	0
0	0	0.9	0	0	0.7
0	0	0.9	0	0	0.8
0	0	0.8	0	0	0.9
0	0	0.7	0	0	0

*

0	-1	1
0	-1	1
0	-1	1

=

2D - activation map.

2D - convolution of filter with the image.

Typically, 3 x 3 or 5 x 5.

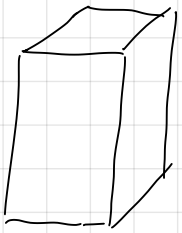
Variations: different stride, image padding → hyperparameter in convolution.

(stride = 1 => move 1 pixel)

controls how far to move

Filter must have 3 channels:

ex). 3x3x3



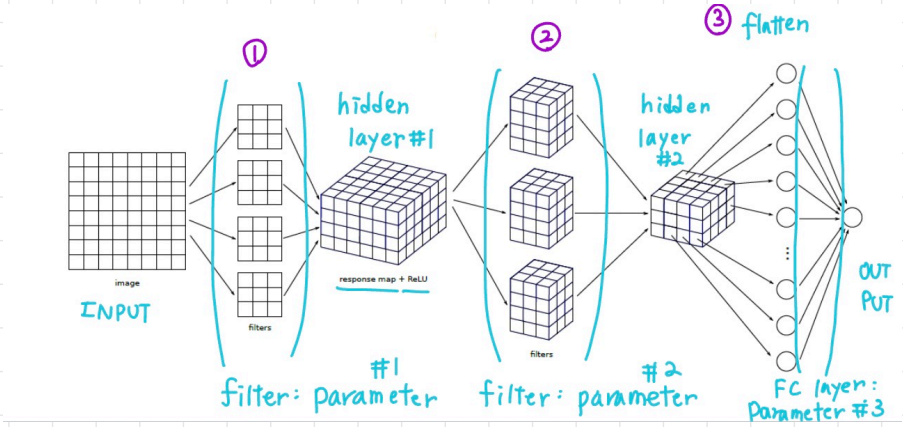
3 channels

filter (same channel as image)

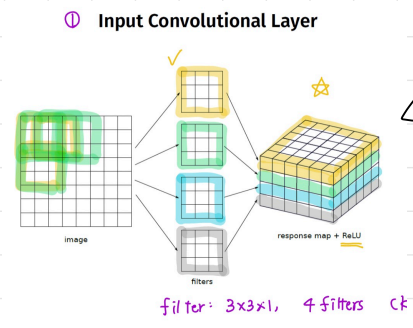
output still 2D

CNN: use convolution in early layers to create new feature representation.

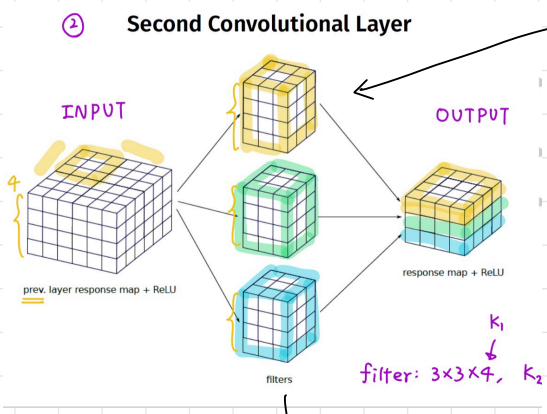
filters are learned



k filters will result in k chanels in the next layer.



activation func. applied entrywise.

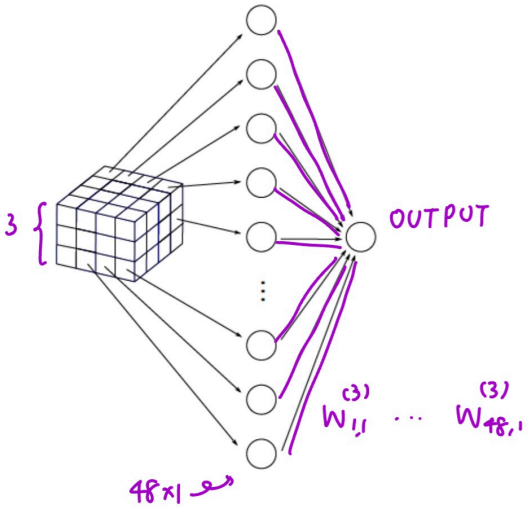


3×3×4

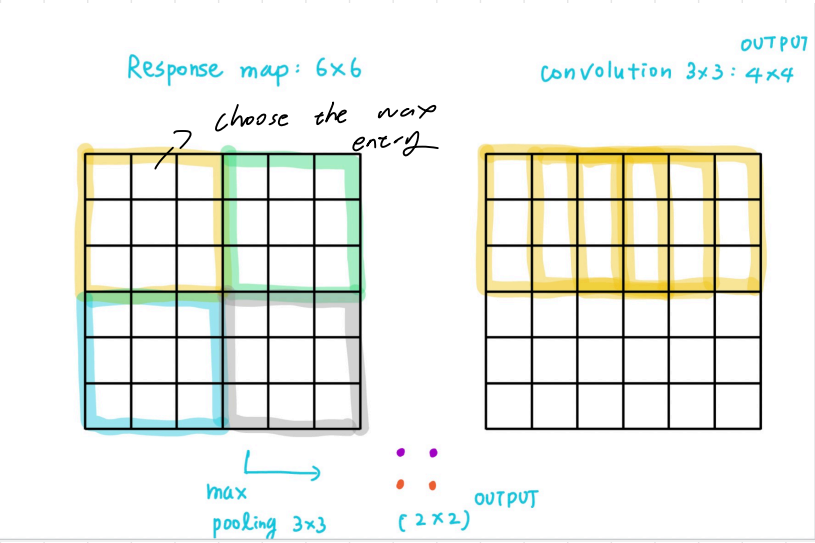
must match the channel of input

output will be 3 channels.

③ **Flattening**: stretch the te last layer to fully connected.



Max pooling :



Autoencoder

representation learning is

↳ finds an encoding function : encode($\vec{x}$) : $R^d \to R^k$.

- captures useful aspect of data distribution.
- encode can decrease dimensionality.
- preserve as much info about $\vec{x}$ as possible.

by encoding, we hope to decode and reconstruct original data.

$$\vec{x} \approx \text{decode}(\text{encode}(\vec{x})).$$

$\Downarrow$ reconstruction.

Reconstruction error: $\sum_{i=1}^{n} \| \vec{x}^{(i)} - \text{decode}(\text{encode}(\vec{x}^{(i)})) \|^2$.

Trivial solution:

$\text{encode}(\vec{x}) = \vec{x} = \text{decode}(\vec{x})$, which is not useful.

example) PCA

encode by projecting onto top k eigenvectors.

$\text{encode}(\vec{x}) = U^T \vec{x}$

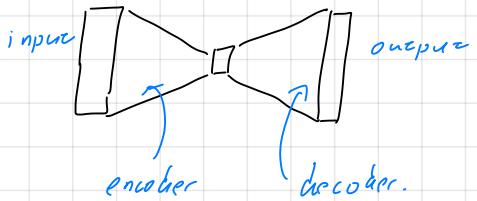
$\text{decode}(\vec{z}) = U \vec{z}$

, where columns of U be top k eigenvectors of covariance matrix.

Encoder / decoder as Neural Net

decode(encode($\vec{x}$)) as a NN. (non-linear activation).

$\mathbb{R}^d \to \mathbb{R}^k \to \mathbb{R}^d$



During training,

$$\sum_{i=1}^{n} \| \vec{x}^{(i)} - H(\vec{x}^{(i)}) \|^2 = \sum_{i=1}^{n} \sum_{j=1}^{d} (\vec{x}_j^{(i)} - (H(\vec{x}^{(i)}))_j)^2.$$

← minimiz reconstruction error.

Autoencoder is generalization of PCA

such that performs orthogonal projection.

$$\text{encoder}(\vec{x}) = U^T \vec{x}$$

$$\text{decode}(\vec{z}) = U \vec{z}$$

So PCA minimizes reconstruction error subject to constraint that columns of U are orthonormal.

However, autoencoder learns (non-orthogonal) projection into same space of PCA

Use of Autoencoder:

- dimensionality reduction (non-linear) ($k < d$.)
- denoising autoencoder ($k > d$). greater dimension.
  - ↓
  - add random noise to each $\vec{x}^{(i)}$ to get $\tilde{x}^{(i)}$.
  - train NN s.t. $H(\tilde{x}^{(i)}) \simeq \vec{x}$.

Causality

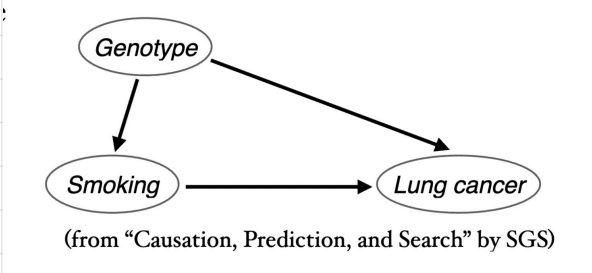
Association: X & Y are associated iff

$$\exists\, x_1 \neq x_2,\quad P(Y \mid X = x_1) \neq P(Y \mid X = x_2)$$

Causality: $\exists\, x_1 \neq x_2,\quad P(Y \mid do(X = x_1)) \neq P(Y \mid do(X = x_2))$.

X is manipulated/intervened to x_1, "do" operation.

Directed graph can represent causal relationship.



(from "Causation, Prediction, and Search" by SGS)

ML can be boosted and more interpretable if we find causality.

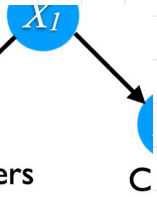
smoking → X_1

X_2 Yellow fingers → X_3 cough.

ML only relys on association with no direct causality.

3 Questions in AI.

- **Association**: Would the person cough if we *find* he/she has yellow fingers?
  $P(X3 \mid X2=1)$ *Seeing*
- **Intervention**: Would the person cough if we *make sure* that he/she has yellow fingers?
  $P(X3 \mid do\ (X2=1))$ *Doing/Intervening*
- **Counterfactual**: Would George cough *had* he had yellow fingers, *given that he does not have yellow fingers and coughs*?
  $P(X3_{\,X2=1} \mid X2 = 0, X3 = 1)$ *Imagining/Retrospection/ Creativity*



Causality is useful!

Causality is requi

How to find causality?

↓

randomized controlled trials

(require lots of resources)

find causal relationship from passive observational data under assumption.

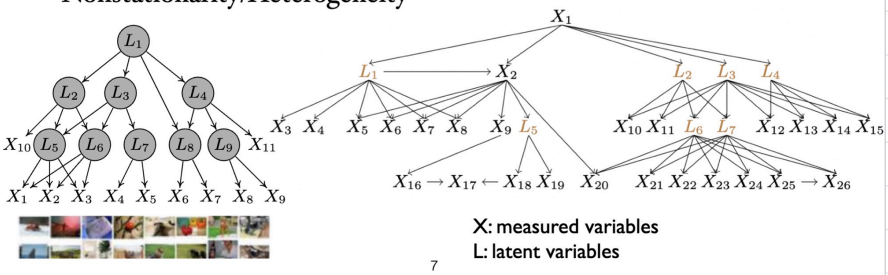
↓

causal discovery & inference.

(discover causal relationship & estimate causal effects from observational data).

Causal discovery in complex scenarios:

- Nonlinearities; Mixed Continuous & Discrete Variables
- Latent Confounding; Latent Causal Representation Learning
- Subsampling / Temporally Aggregation in time series
- Measurement Error; Selection Bias; Missing Values
- Nonstationarity/Heterogeneity



7

causal discovery example for latent discovery

Big 5:
openness (O); conscientiousness (C); extraversion (E); agreeableness (A); neuroticism (N)