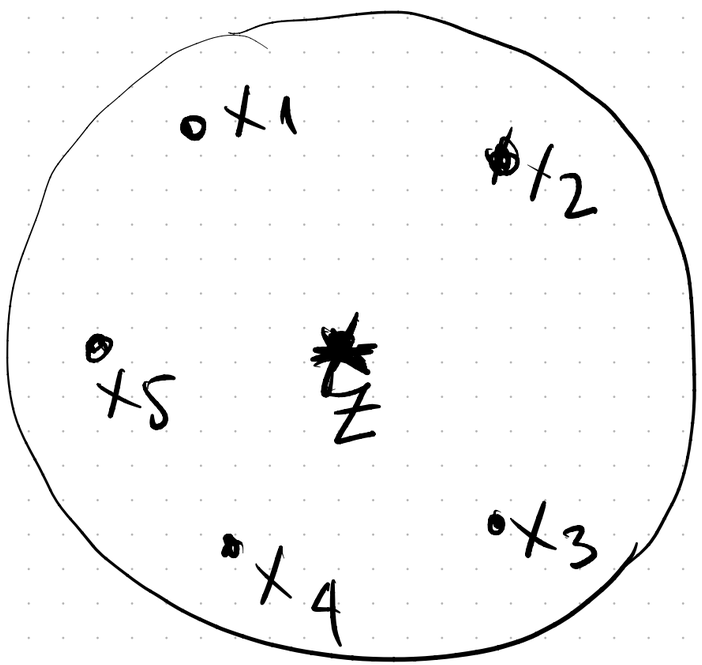
Lecture 7|18

- **KNN**
- optional: tSNE
- Kernel PCA
- Hw5 demo

K Nearest Neighbor: For each point $z \in$ Test

define neighbourhood N_z = { set of closest points (training) to z }



- predict label/class/target by estimating from N_z

classification: predict class Y_z that is most present in N_z Y_z = **mode** { N_z }
(most frequent)

quantity/regression: predict AVG { N_z } label

ex y = house price $\Rightarrow$ pred $Y_z = \underset{x_i \in N_z}{AVG} \{ y_i \}$

not really training, just pred y_z = estimate from closest neighbors to z

- need dist/similarity (kernel) across datapoints

$k_{ij} = sim(x_i, x_j)$

$k_{iz} = sim(x_i, z)$ ...

appropriate for data and for task

ex x_i = patients $\Rightarrow$ k_{ij} = similarity of patients w.r.t diabetes
y = diabetes

---

x_i = images $\Rightarrow$ k_{ij} = similarity of image prod w.r.t. price (willingness to pay)
y = prod. price

---

x_i = email $\Rightarrow$ k_{ij} = similarity of emails w.r.t. spam
y = spam/not

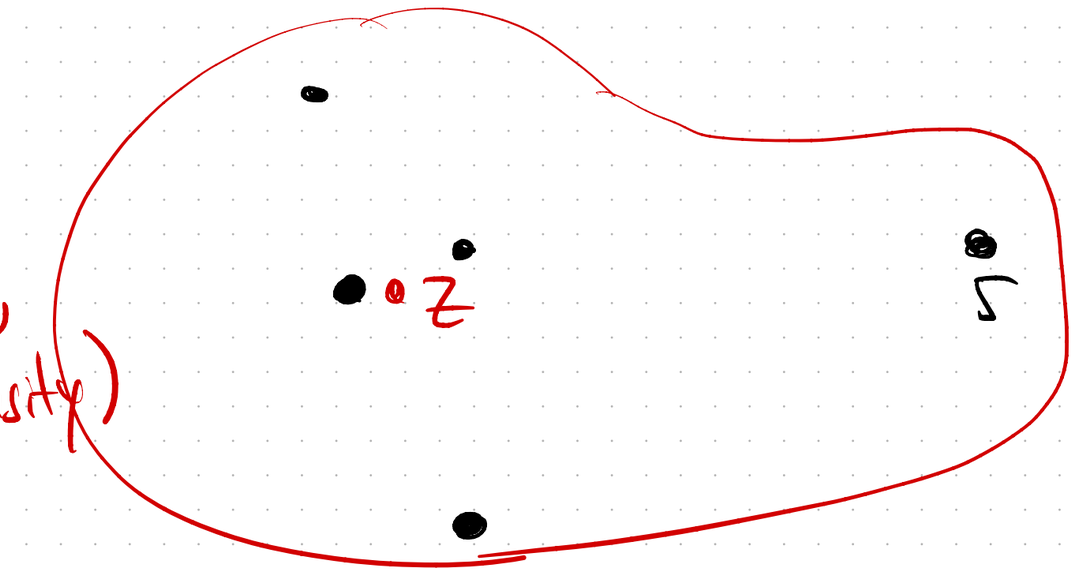
---

x_i = movies $\Rightarrow$ k_{ij} = similarity of movies w.r.t user satisfaction
y = rating

KNN 3 variants

- K=fixed ; ex ⇒ K=5. N_Z always has 5 training points.
  $N_Z = \{$ closest 5 to z $\}$
  - rank all training points by K_{zi} similarity
  - select top k ⇒ N_Z

disadvantage : some z dont have 5 close neighbors (low density)

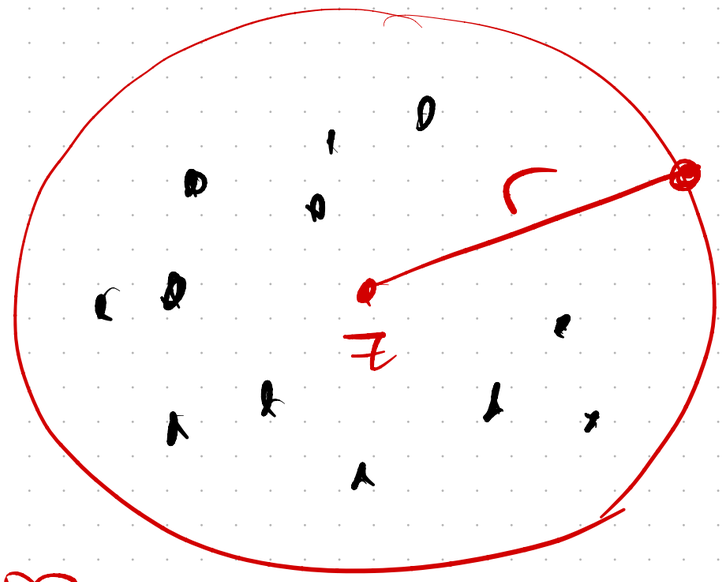


range

- range = fixed & variable

$N_Z = \{ x_i \text{ training} \mid K_{iz} \leq r \}$

$dist(i,z) \leq r$



disadvantage: some z very few neighbors in range

some z : many neighbors in range

- use **all** training points weighted by similarity $k_{iz} = sim(x_i, z)$

regression $predict(z) = \frac{\sum_{i=1}^{N} k_{iz} \cdot y_i}{\sum_{i=1}^{N} k_{iz}}$

y_i → labels / targets

k_{iz} → weights

classification

class label ↑

$score(z, y_e) = \sum_{i=1}^{N} k_{iz} \cdot \boxed{1[y_i = y_e]}$ filter

N_z = all training set, but weighted.

k_{iz} = high ⇒ $x_i \sim z$ ⇒ y_i counts a lot

k_{iz} = low ⇒ x_i not similar to z ⇒ y_i doesn't count much

Kernel PCA

- rewrite PCA primal ⟹ dual form.

dual variables.

why?

$$\text{PCA word} = X \cdot \begin{bmatrix} | & | & & | \\ e_1 & e_2 & \cdots & e_D \\ | & | & & | \end{bmatrix} \overset{?}{=} \underbrace{X X^T}_{K} \cdot B$$

eigenvectors(Σ_{covar})

sorted by desc eigenval.

- can use any valid kernel K (not just linear XX^T)

choose K

ex $K = \text{gaussian} = e^{-\|x_i - x_j\|^2 / 2\sigma^2}$

- compute B for that kernel.

compute

$B = \text{eigenvect}(K)$

intuition we want eigen vectors (privat var) $= X^T \boxed{\beta}$ dual.

detail (skip): show that $e = X^T\beta$ possible for every e

calculate/find β? e eigen vector of Σ (COVAR) $\Rightarrow \Sigma e = \boxed{\lambda} e$ (eigen val)

derivation: $\Sigma e = \lambda e$

$\boxed{\Sigma = \frac{1}{N} X^T X}$ covar estimation

$\boxed{e = X^T \beta}$ → want

Σ

$\boxed{\left(\frac{1}{N} X^T X\right)} \boxed{X^T \beta}^{e} = \lambda \boxed{X^T \beta}^{e}$

$X^T X X^T \beta = N\lambda X^T \beta$

X left: $\boxed{X X^T} \boxed{X X^T} \beta = N\lambda \boxed{X X^T} \beta$

get rid of K?: $K \cdot K \cdot \beta = N\lambda K \cdot \beta$

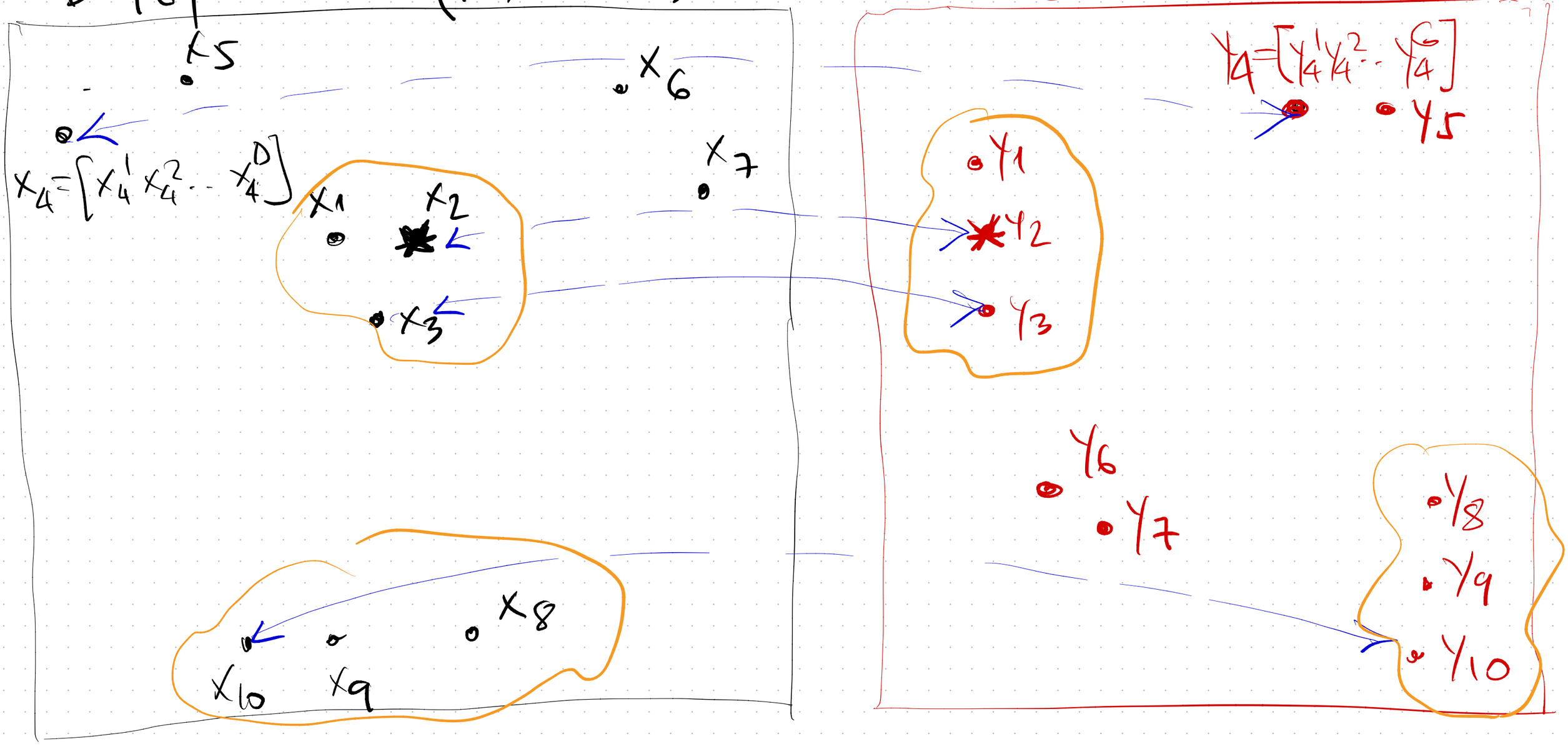
$K \cdot \beta = \boxed{N\lambda}^{\text{scalar}} \cdot \beta$ → β = eigenvector(K)

$N\lambda$ = coresp eigenval

t-SNE representation of data D-dim ⟹ G-dim (G ≪ D)

D-dim X-space (original) $x_i \Rightarrow y_i$ G-dim Y-space (new)

$D = 784$ $x = (x^1 x^2 \dots x^D)$ $G = 3$ $y = (y^1 y^2 \dots y^G)$



- do not worry about exact distance / positions (lin algebra)

$dist(x_1, x_5)$?? $dist(y_1, y_5)$

$angle(x_1 x_4 x_7)$?? $angle(y_1 y_4 y_7)$

- care about: preserve "close" vs "far" neighbors.

x_2 much closer to x_3 than to x_{10} $\Longleftrightarrow$ y_2 much closer to y_3 than to y_{10}

one time / Fixed

P = distribution of distances (X)

$$P_{ij} = \frac{\exp(-\|x_i - x_j\|^2 / 2\sigma^2)}{\text{normalized}}$$

$N \times N$

Q = distribution of distances (Y)

≠ not gaussian. use t-distrib.

t-distrib

$$Q_{ij} = \frac{\left[1 / (1 + \|y_i - y_j\|^2)\right]}{\text{normalized}}$$

$N \times N$

not fixed, updated with Y

- Given $X_{N \times D}$ compute $\underset{N \times N}{P}$ (search for "width" σ^2 by perplexity)
- want $Y_{N \times G}$ such that $Q_{dist}(Y) \simeq P_{dist}(X)$

$$\text{MIN OBJ} = KL(P \| Q) = \sum_{ij} P_{ij} \log \frac{P_{ij}}{Q_{ij}}$$

P_{ij} → fixed

P = fixed
Q = variable (Y)

- init Y = random (better cluster)
- iterate (Grad descent)
  - compute Q = t-dist (Y positions)
  - update Y based Grad Descent Rule

$$\frac{\partial OBJ}{\partial Y_i} = 4 \sum_j (P_{ij} - Q_{ij})(y_i - y_j) \frac{1}{1 + \|y_i - y_j\|^2}$$

Q_{ij} numerator

easy to compute (fast using plain vanilla - kind of comp on graph degrees)

$$y_i^{new} = y_i + \lambda \frac{\partial OBJ}{\partial y_i} + \beta (y_i - y_i^{prev})$$

momentum