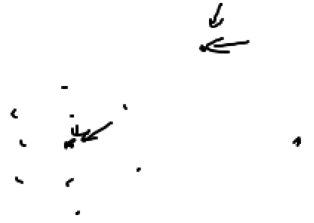
K means clustering

1) assign each obs. randomly to each of the K classes, but so that none of the clusters is empty

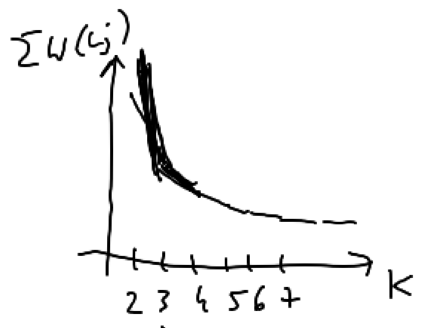
2) $\bar{x}_k$ – centroid $\bar{x}_k = \frac{1}{|C_k|} \sum_{i \in C_k} x_i \in \mathbb{R}^p$



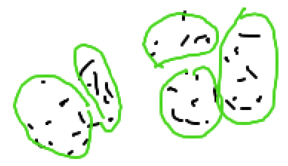
one can prove that $\sum_{j=1}^{K} W(C_j)$ decreases in each iteration of step 2)

- K is given in advance

- The above algorithm may be repeated multiple times to find a better solution



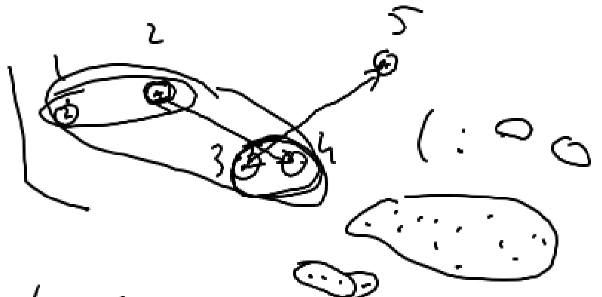
a natural number of clusters here?



Hierachical clustering $x_1, \dots, x_n \in \mathbb{R}^p$

1) begin with n observations and a measure (e.g. Euclidean distance) of all $\binom{n}{2}$ pairwise dissimilarities
   Treat each observation as its own cluster, so we start with n clusters, each having 1 observation

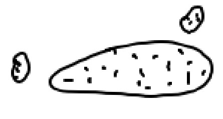
2) for $i = n, n-1, \dots, 2$:
   (a) Examine all pairwise inter-cluster dissimilarities among i clusters and identify the pair of ~~clusters~~ most similar clusters.
   Fuse these two clusters.
   The dissimilarity between these two clusters will give the height at the dendrogram at which fusion should be placed
   (b) compute new pairwise inter-cluster dissimilarities between remaining $(i-1)$ clusters

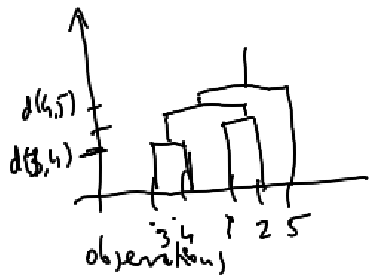


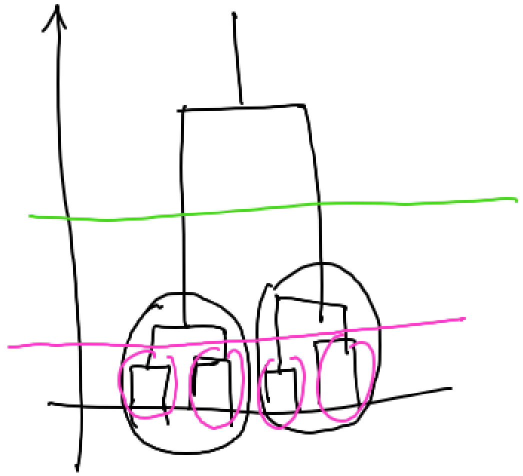
dissimilarities (linkage)

complete $\max_{x \in K_1, y \in K_2} d(x,y)$ average $\frac{1}{|K_1||K_2|} \sum_{x \in K_1, y \in K_2} d(x,y)$

single $\min_{x \in K_1, y \in K_2} d(x,y)$ ward $\sum_{x \in K_1, y \in K_2} d(x,y)$

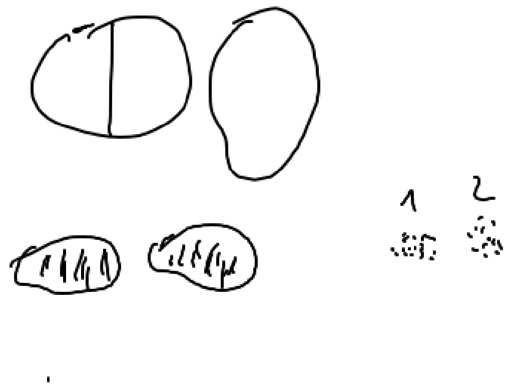



d(4,5)
d(3,4)
3
4
1
2
5
observations



This algorithm gives us a sequence of clusterings, each pair of clusterings has the property that the clusters in one of them are subsets of the clusters in the other

- it is deterministic



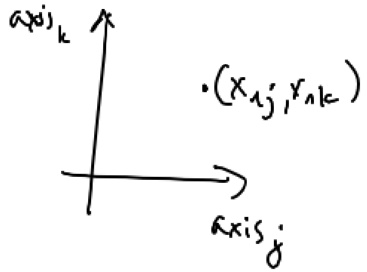
PCA principal component analysis

used to reduce dimensionality

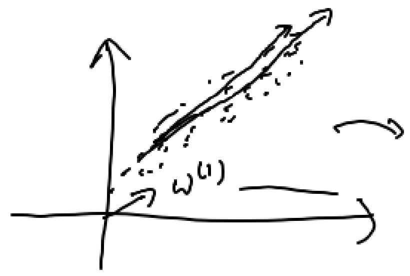
$$X = \begin{bmatrix} x_{11} & \dots & x_{1p} \\ x_{21} & \dots & x_{2p} \\ \vdots & & \\ x_{n1} & \dots & x_{np} \end{bmatrix} \in \mathbb{R}^{n \times p}$$

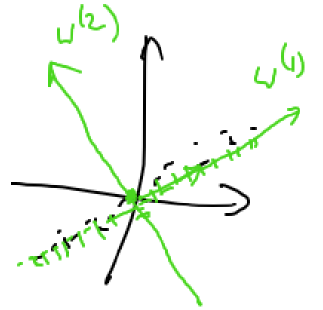
rows – n observations each in $\mathbb{R}^p$

$x_i = (x_{i1}, x_{i2}, \dots, x_{ip}) \in \mathbb{R}^p$



$$w^{(1)} = \begin{bmatrix} w_1^{(1)} \\ \vdots \\ w_p^{(1)} \end{bmatrix}$$





We want to find a new coordinate system $\underbrace{w^{(1)}, w^{(2)}, \dots, w^{(p)}}^{\text{orthonormal basis}}$

such that the data varies the most along $w^{(1)}$, and then $w^{(2)}$, ...

It is convenient to assume that <u>each column of X has mean zero.</u>

the first coordinate of ~~our~~ samples in the coordinate system will have the form:

$$\langle x_i, w^{(1)} \rangle = \sum_{j=1}^{p} x_{ij} w_j^{(1)} \cancel{=} \quad , i = 1, \dots, n$$

we want that

$$\sum_i |\langle x_i, w^{(1)} \rangle|^2 = \sum_i \Big(\underbrace{\sum_{j=1}^{p} x_{ij} w_j^{(1)}}_{\langle x_i, w^{(1)} \rangle} \Big)^2 = \sum_i |(X \cdot w^{(1)})_i|^2 = \| X \cdot w^{(1)} \|^2 = \langle X \cdot w^{(1)}, X \cdot w^{(1)} \rangle =$$

$$= \langle w^{(1)}, \underbrace{X^T X}_{\text{positive def}} \cdot w^{(1)} \rangle$$ symmetric matrix

to be the largest possible

So $w^{(1)}$ should be the eigenvector of X^TX corresponding to the largest eigenvalue

$w^{(2)}$ — " — ______________ the 2nd — " — ______

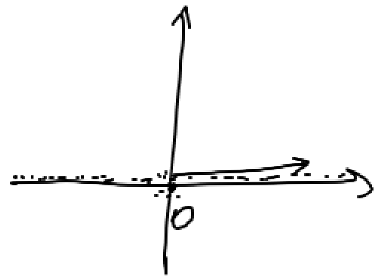
and so on.

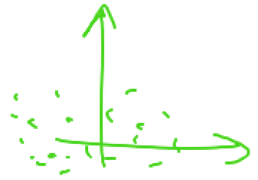
We can rewrite our data in the new coordinate system:

$$\tilde{X} = \begin{bmatrix} \overset{\in \mathbb{R}^p}{\langle x_1, w^{(1)} \rangle} & \langle x_1, w^{(2)} \rangle & & \langle x_1, w^{(p)} \rangle \\ \langle x_2, w^{(1)} \rangle & \langle x_2, w^{(2)} \rangle & & \vdots \\ \vdots & \vdots & \cdots & \\ \langle x_n, w^{(1)} \rangle & \langle x_n, w^{(2)} \rangle & & \langle x_n, w^{(p)} \rangle \end{bmatrix}$$

this column carries the most information about X

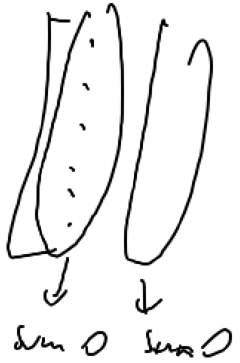
... and so on



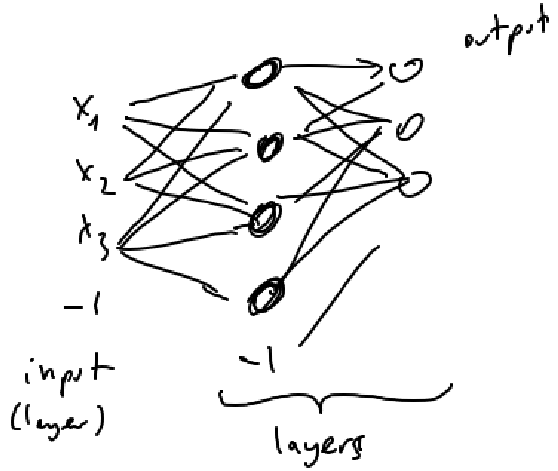


Above it was important to make the data so that $\sum_i x_{ij} = 0$, $j = 1, 2, \dots, p$

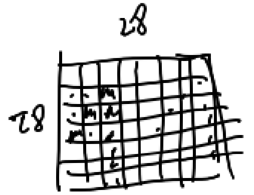
Sometimes it is also desirable to normalise the data, that is rescale each column so that $\sum_i x_{ij}^2 = 1$.



Neural nets



For image/pattern recognition dense layers are not very good for several reasons:



$\longrightarrow$ dense layer treats such an image as a vector of $\underline{\underline{28^2}}$ 784 numbers

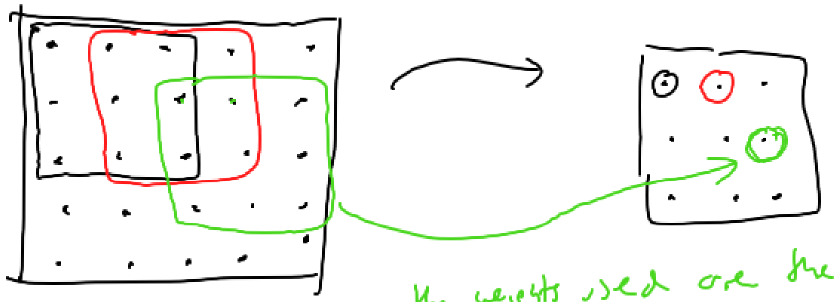
100

1) the geometrical structure is lost
2) the image after a slight shift looks completely different for the dense layer
3) often one has a huge number of weights e.g. 28^2 - input, 100 - neurons $\}$ $\Rightarrow$ $\approx$ 78'500 weights in the first layer

200×200 = 40k

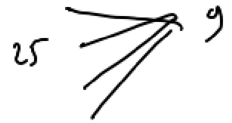
Convolutional neural networks

they exploit the geometrical structure



the weights used are the same
for each output
just 10 weights here (9 inputs + 1 bias)

25 inputs $\longrightarrow$ 9 outputs



9·26 weights for dense