Machine Learning for Computational Linguistics

Unsupervised learning

Çağrı Çöltekin

University of Tübingen
Seminar für Sprachwissenschaft

May 24, 2016

Homework 1

Common confusions (mainly about bigrams):

- Word ngrams (typically) do not cross sentence boundaries
- Order is important in a bigram
- While calculating conditional probabilities and PMI for bigrams, you need to use probability of a word given it is the first/second word in the bigram, not its unigram probability
- The base of logarithm does not matter for information theoretic measures. Base only changes the 'unit'. As long as you are consistent, using any base is fine.

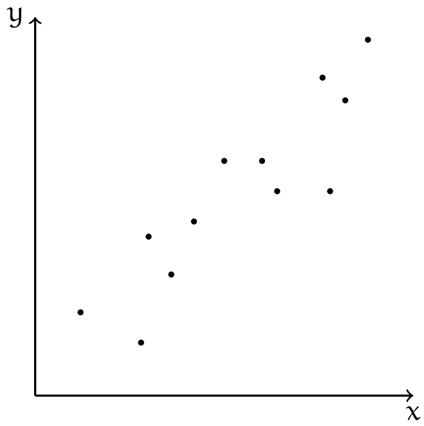
Projects

- Please send me short project proposal document (about one page) by June 13 with
  - the list of the project members
  - a title, brief description
  - whether you have already obtained data for the project or not
  - the methods you intend to apply
- and let me know as soon as you formed your project team

Supervised learning

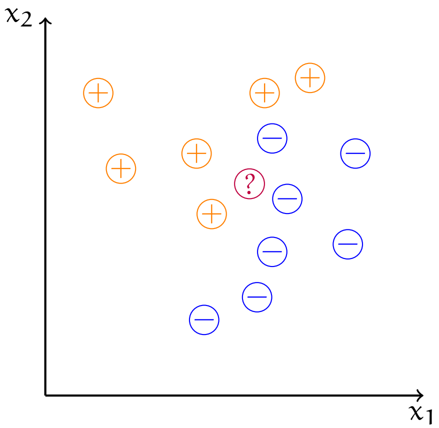
- The methods we studied so far are instances of supervised learning
- In supervised learning, we have a set of predictors x, and want to predict a response or outcome variable y
- During training we have access to both input and output variables
- Typically, training consist of estimating parameters w of a model
- During prediction, we are given x and make predictions based on what we learned (e.g., parameter estimates) during training

Supervised learning: regression



- The response (outcome) variable (y) is a quantitative variable.
- Given the features ($\boldsymbol{x}$) we want to predict the value of y

Supervised learning: classification



- The response (outcome) is a label. In the example: positive ⊕ or negative ⊖
- Given the features (x_1 and x_2), we want to predict the label of an unknown instance ?⃝

Supervised learning: estimating parameters

- Most models/methods estimate a set of parameters $\boldsymbol{w}$ during training
- Often we find the parameters that minimize a cost function $J(w)$
  - For least-squares regression

$$J(w) = \sum_i (\hat{y}_i - y_i)^2$$

  - For logistic regression, the negative log likelihood

$$J(w) = -\log \mathcal{L}(w)$$

- If the cost function is *convex* we can find a *global* minimum using analytic solutions, or search methods such as *gradient descent*

Regularization

- To counteract overfitting to the training data, we typically modify the objective functions to restrict the space of the parameters
- Common regularization methods are
  - L1 regularization minimize

$$J(w) + \lambda\|\boldsymbol{w}\|_1$$

  - L2 regularization minimize

$$J(w) + \lambda\|\boldsymbol{w}\|$$

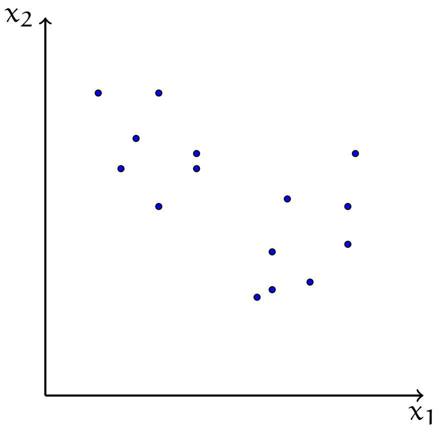
Unsupervised learning

- In unsupervised learning, we do not have labels
- Our aim is to find useful patterns/structure in the data
- Typical unsupervised methods include
  - Clustering: find related groups of instances
  - Density estimation: find a probability distribution that explains the data
  - Dimensionality reduction: find a accurate/useful lower dimensional representation of the data
- Evaluation is difficult: we do not have ‘true’ labels/values
- Sometimes unsupervised methods can be used in conjunction with the supervised methods

Clustering

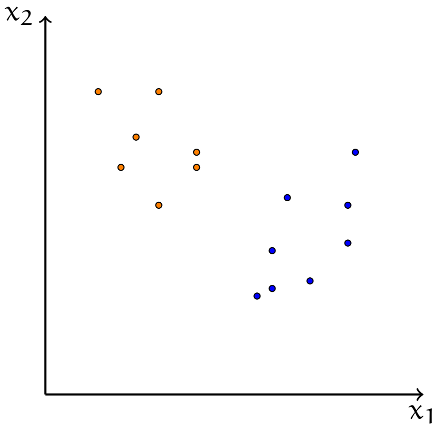
- Our aim is to find groups of instances/items that are similar to each other
  - Clustering similar languages, dialects, documents, users/authors …
- The *distance measure* is important (but also application specific)
- Clustering can be *hierarchical* or non-hierarchical
- Clustering can be *bottom-up* (agglomerative) or top-down (divisive)
- For most (useful) problems we cannot find globally optimum solutions, we often rely on greedy algorithms finding local optima.

Clustering example in two dimensions



- Unlike classification we do not have labels

Clustering example in two dimensions



- Unlike classification we do not have labels
- We want to find 'natural' groups in the data
- Intuitively, similar or closer data points are grouped together

Similarity and distance

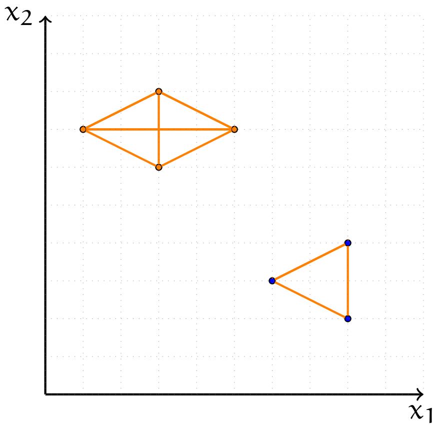
- The notion of distance (similarity) is very important in clustering. A distance measure D,
  - is symmetric: $D(a, b) = D(b, a)$
  - non-negative: $D(a, b) \geqslant 0$ for all a, b, and it $D(a, b) = 0$ iff $a = b$
  - obeys triangle inequality: $D(a, b) + D(b, c) \geqslant D(a, c)$
- The choice of distance is application specific
- A few common choices:
  - Euclidean distance: $\|a - b\| = \sqrt{\sum_{j=1}^{k} (a_j - b_j)^2}$
  - Manhattan distance: $\|a - b\|_1 = \sum_{j=1}^{k} |a_j - b_j|$

Similarity and distance

- The notion of distance (similarity) is very important in clustering. A distance measure D,
  - is symmetric: $D(a, b) = D(b, a)$
  - non-negative: $D(a, b) \geqslant 0$ for all a, b, and it $D(a, b) = 0$ iff $a = b$
  - obeys triangle inequality: $D(a, b) + D(b, c) \geqslant D(a, c)$
- The choice of distance is application specific
- A few common choices:
  - Euclidean distance: $\|a - b\| = \sqrt{\sum_{j=1}^{k} (a_j - b_j)^2}$
  - Manhattan distance: $\|a - b\|_1 = \sum_{j=1}^{k} |a_j - b_j|$
- We will often face with defining distance measures between linguistic units (letters, words, sentences, documents, …)

How to do clustering

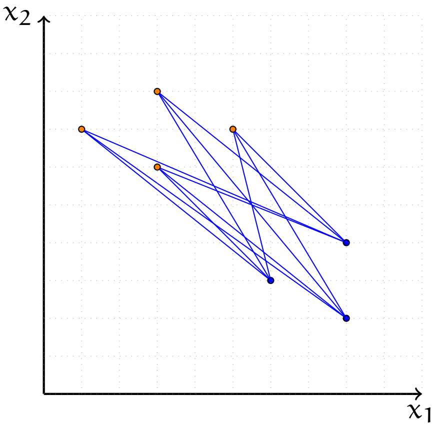
Most clustering algorithms try to minimize the scatter within each cluster. Which is equivalent to maximizing the scatter between clusters



$$\frac{1}{2}\sum_{k=1}^{K}\sum_{C(a)=k}\sum_{C(b)=k} d(a,b)$$

How to do clustering

Most clustering algorithms try to minimize the scatter within each cluster. Which is equivalent to maximizing the scatter between clusters

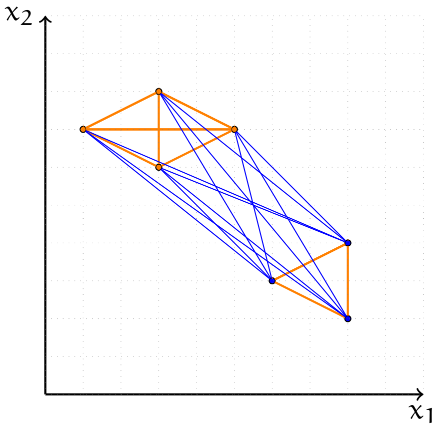


$$\frac{1}{2}\sum_{k=1}^{K}\sum_{C(a)=k}\sum_{C(b)=k} d(a,b)$$

- $$\frac{1}{2}\sum_{k=1}^{K}\sum_{C(a)=k}\sum_{C(b)\neq k} d(a,b)$$

How to do clustering

Most clustering algorithms try to minimize the scatter within each cluster. Which is equivalent to maximizing the scatter between clusters



$$\frac{1}{2}\sum_{k=1}^{K}\sum_{C(a)=k}\sum_{C(b)=k} d(a,b)$$

▶

$$\frac{1}{2}\sum_{k=1}^{K}\sum_{C(a)=k}\sum_{C(b)\neq k} d(a,b)$$

Exact solution (finding global optimum) is not possible for realistic data. We use methods that find a local minimum.

K-means clustering

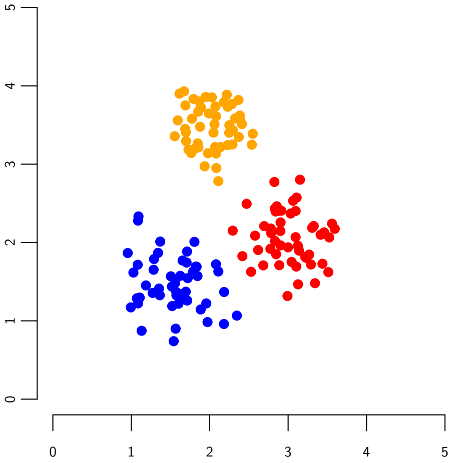
K-means is a popular method for clustering.

1. Randomly choose *centroids*, $m_1, \ldots, m_K$, representing K clusters
2. Repeat until convergence
   - Assign each data point to the cluster of the nearest centroid
   - Re-calculate the centroid locations based on the assignments

Effectively, we are finding a local minimum of the sum of squared Euclidean distance within each cluster

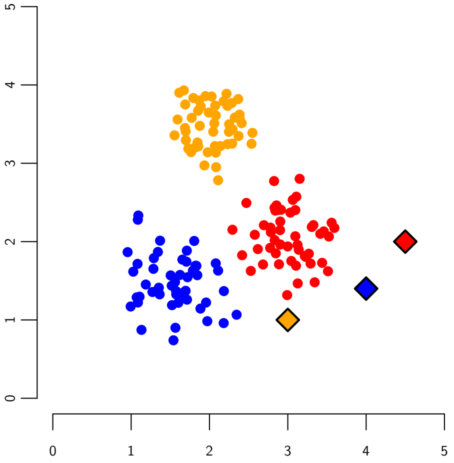
$$\frac{1}{2} \sum_{k=1}^{K} \sum_{C(a)=k} \sum_{C(b)=k} \|a - b\|^2$$

K-means clustering: visualization



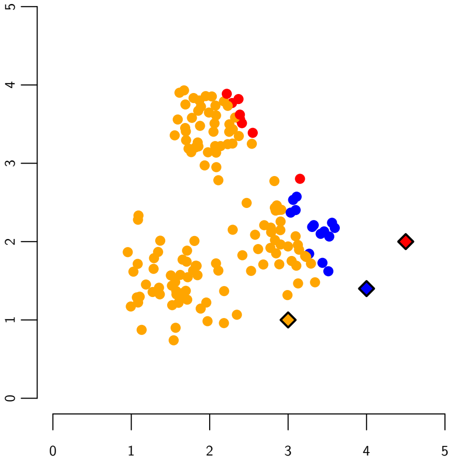
- The data
- Set cluster centroids randomly
- Assign data points to closest centroid
- Recalculate the centroids

K-means clustering: visualization



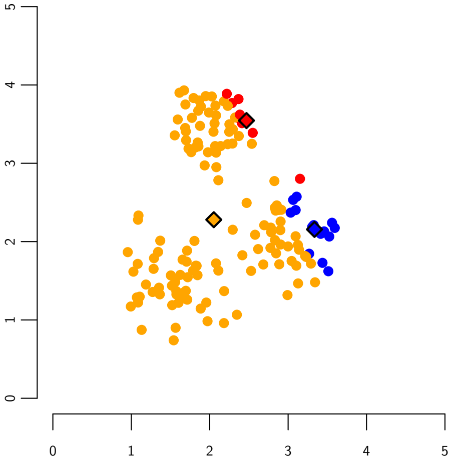
- The data
- Set cluster centroids randomly
- Assign data points to closest centroid
- Recalculate the centroids

K-means clustering: visualization



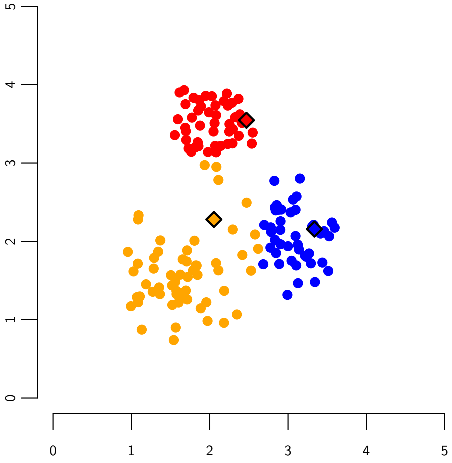
- The data
- Set cluster centroids randomly
- Assign data points to closest centroid
- Recalculate the centroids

K-means clustering: visualization



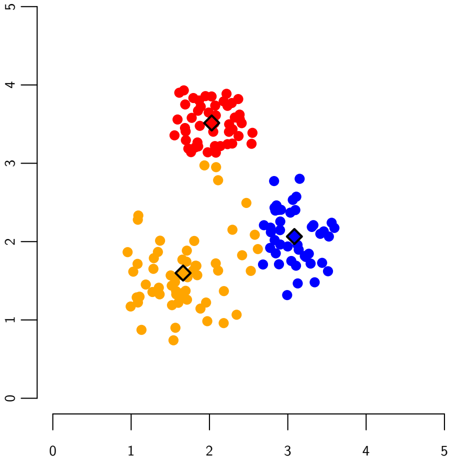
- The data
- Set cluster centroids randomly
- Assign data points to closest centroid
- Recalculate the centroids

K-means clustering: visualization



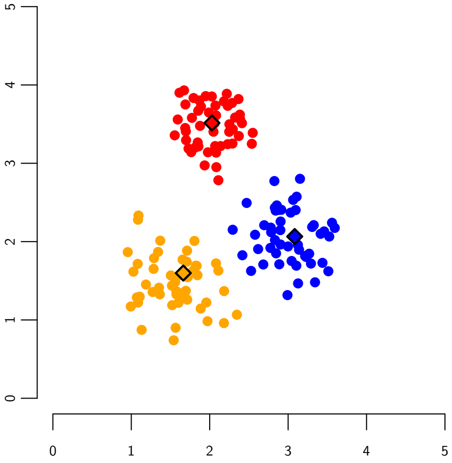
- The data
- Set cluster centroids randomly
- Assign data points to closest centroid
- Recalculate the centroids

K-means clustering: visualization



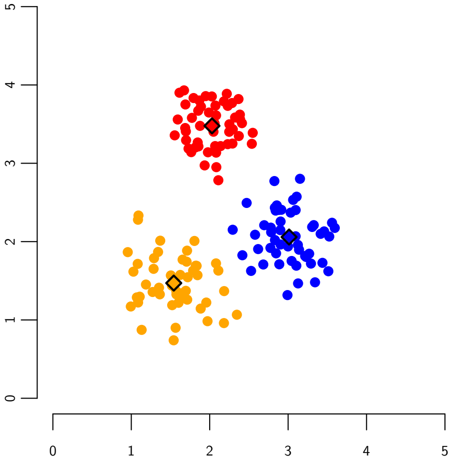
- The data
- Set cluster centroids randomly
- Assign data points to closest centroid
- Recalculate the centroids

K-means clustering: visualization



- The data
- Set cluster centroids randomly
- Assign data points to closest centroid
- Recalculate the centroids

K-means clustering: visualization



- The data
- Set cluster centroids randomly
- Assign data points to closest centroid
- Recalculate the centroids

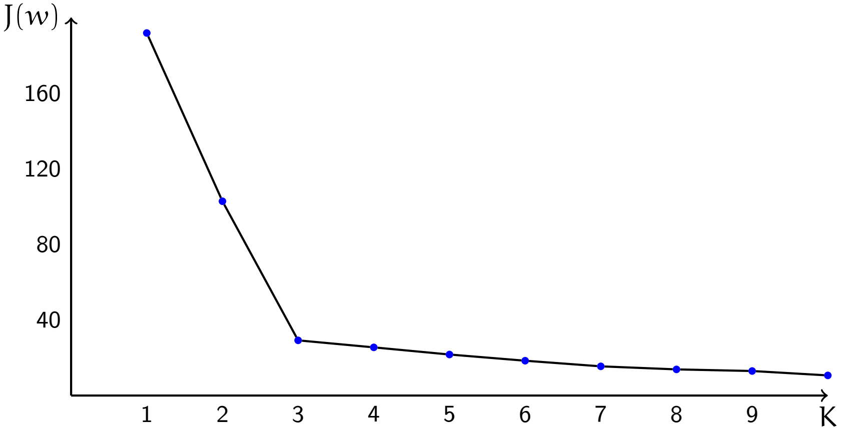
K-means: issues

- K-means requires the data points to be on an Euclidean space
- K-means is sensitive to outliers
- The results are highly sensitive to initialization
  - There are some smarter ways to select initial points
  - One can do multiple initializations, and pick the best (with lowest within-group squares)
- It works well with approximately equal sized round shaped clusters
- We need to specify number of clusters in advance

How many clusters?

- The number of clusters is defined for some problems, e.g., classifying news into a fixed set of topics/interests
- For others, there is no clear way to select the best number of clusters
- The error (within cluster scatter) always decreases with increasing number of clusters, using a test set or cross validation is not useful either
- A common approach is clustering for multiple K values, and picking where there is an 'elbow' in the graph against the error function

How many clusters?



K-medoids

- K-means requires the data to be on an Euclidean space
- Sometimes, we only have distances between the data points, the features do not lie in an Euclidean space
- K-medoids algorithm is an alternation of K-means
- Instead of calculating centroids, we try to find most typical data point at each iteration
- It is less sensitive to outliers
- It is computationally more expensive than K-means

Density estimation

- K-means treats all data points in a cluster equally
- A 'soft' version of K-means is density estimation for Gaussian mixtures, where
  - We assume the data comes from a mixture of K Gaussian distributions
  - We try to find the parameters of each distribution that maximizes the probability of the data
- Unlike K-means, mixture of Gaussians assigns probabilities for each data point belonging to one of the clusters
- It is typically estimated using the expectation-maximization (EM) algorithm

Density estimation using the EM algorithm

- The EM algorithm (or its variations) is used in many (unsupervised) learning models with latent/hidden variables
- It is closely related to the K-means algorithm

1. Randomly initialize the parameters of K Gaussian distributions (μ, Σ)
2. Iterate until convergence:

E-step Compute probability of each data point belonging to each cluster, given the parameters

M-step Re-estimate the mixture density parameters using the probabilities estimated in the E-step

Hierarchical clustering

- Instead of flat division to clusters as in K-means, hierarchical clustering builds a hierarchy based on similarity of the data points
- There are two main 'modes of operation':

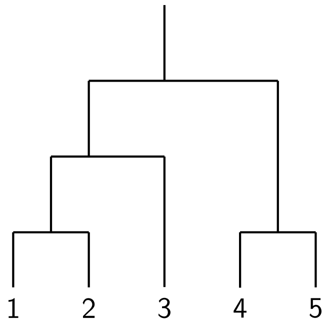
Bottom-up
: or agglomerative clustering starts with individual data points, and merges until a single root node is reached

Top-down
: or divisive clustering starts with a single cluster, and splits until all leaves are single data points

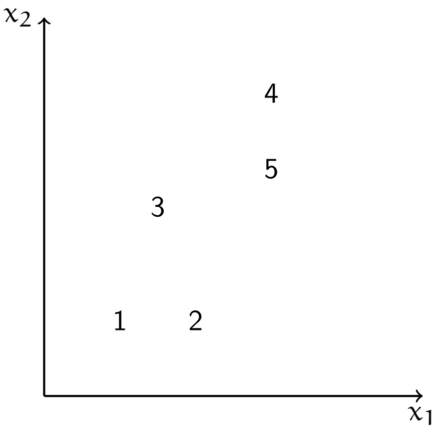
- Hierarchical clustering operates on differences
- The result is a binary tree called dendrogram
- Dendrogram are easy to interpret (especially if data is hierarchical)
- The algorithm does not commit to the number of clusters K from the start, dendrogram could be 'cut' at any height for a particular number of clusters

Agglomerative clustering

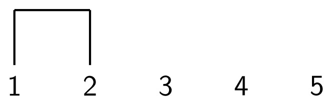
1. Compute the similarity/distance matrix
2. Assign each data point to its own cluster
3. Repeat until no cluster left to merge
   - Pick two clusters that are most similar to each other
   - Merge them into a single cluster

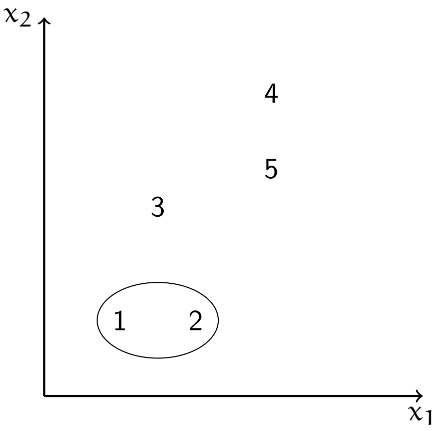


Agglomerative clustering demonstration

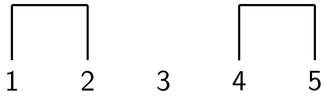


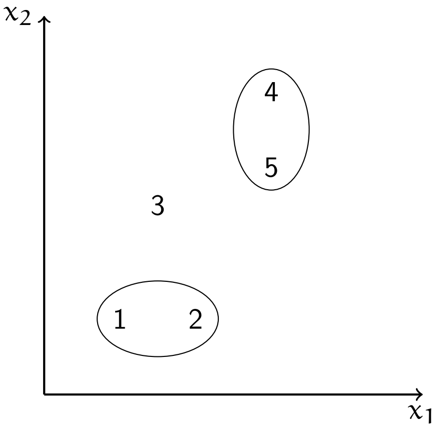
Agglomerative clustering demonstration



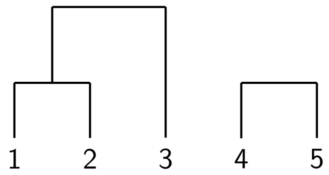


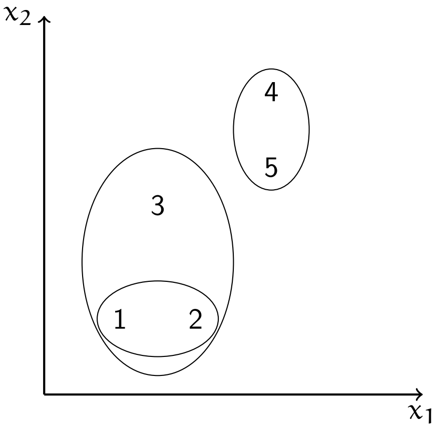
Agglomerative clustering demonstration



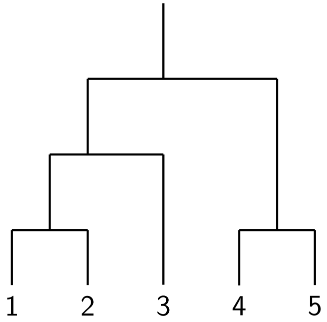


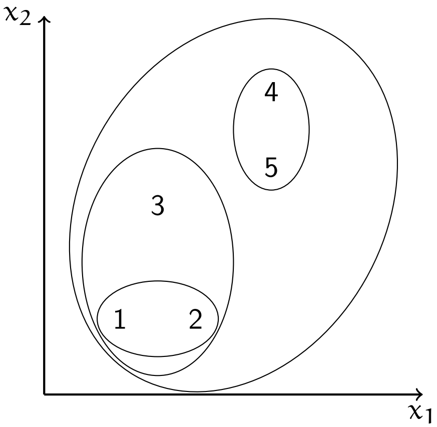
Agglomerative clustering demonstration





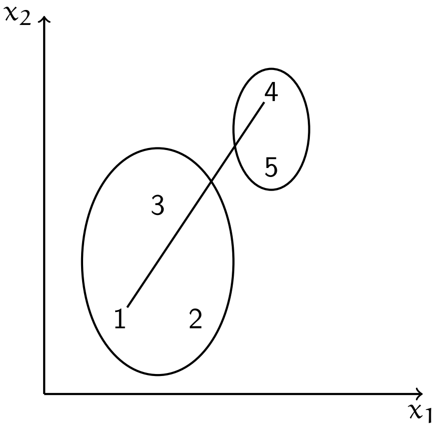
Agglomerative clustering demonstration





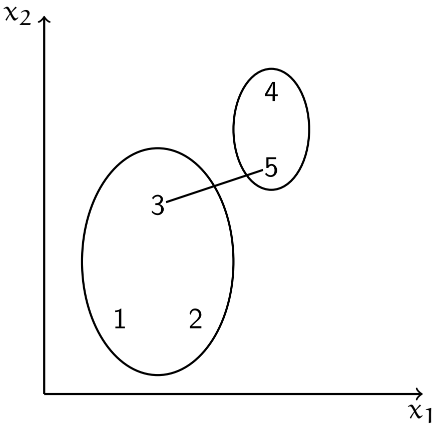
How to calculate between cluster distances

Complete	maximal inter-cluster distance



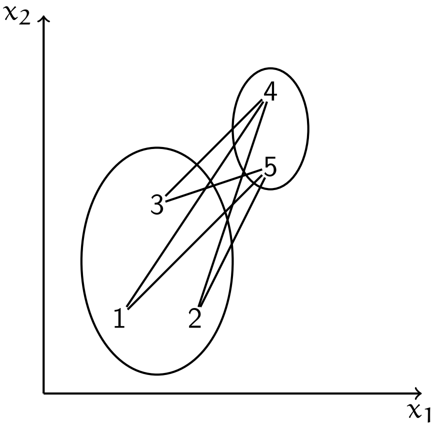
How to calculate between cluster distances

Complete	maximal inter-cluster distance
Single	minimal inter-cluster distance



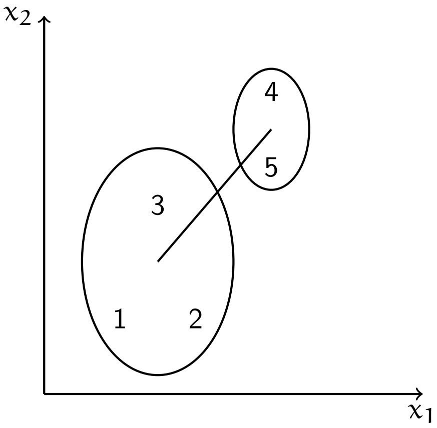
How to calculate between cluster distances

Complete	maximal inter-cluster distance
Single	minimal inter-cluster distance
Average	mean inter-cluster distance



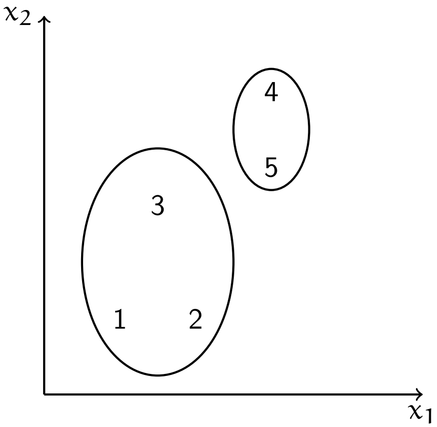
How to calculate between cluster distances

Complete	maximal inter-cluster distance
Single	minimal inter-cluster distance
Average	mean inter-cluster distance
Centroid	distance between the centroids



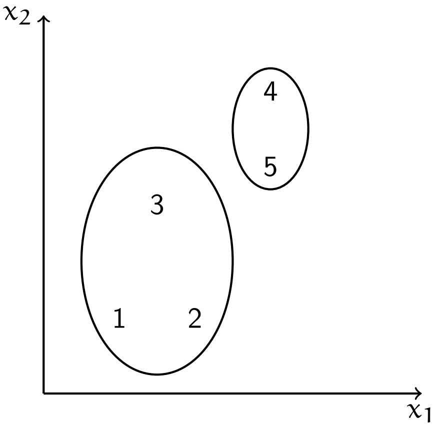
How to calculate between cluster distances

Complete	maximal inter-cluster distance
Single	minimal inter-cluster distance
Average	mean inter-cluster distance
Centroid	distance between the centroids



How to calculate between cluster distances

Complete	maximal inter-cluster distance
Single	minimal inter-cluster distance
Average	mean inter-cluster distance
Centroid	distance between the centroids



Note: single linkage tends to produce unbalanced trees.

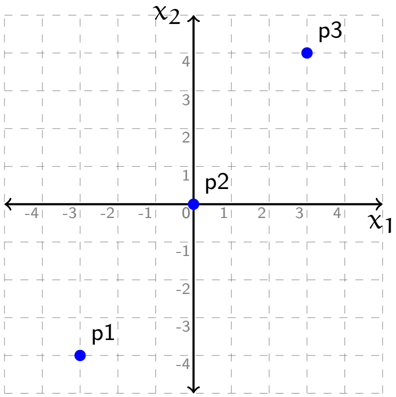
Clustering: some closing notes

- We do not have proper evaluation procedures for clustering results (for unsupervised learning in general)
- Clustering is typically unstable, slight changes in the data or parameter choices may change the results drastically
- Approaches against instability include some validation methods, or producing ‘probabilistic’ dendrograms by running clustering with different options

Principal component Analysis

- Principal component analysis (PCA) is a method for dimensionality reduction
- PCA maps the original data into a lower dimensional space by a linear transformation (rotation)
- The transformed variables retain most of the variation (=information) in the input
- PCA can be used for
  - visualization
  - data compression
  - reducing dimensionality of the input for use in supervised methods
  - eliminating noise

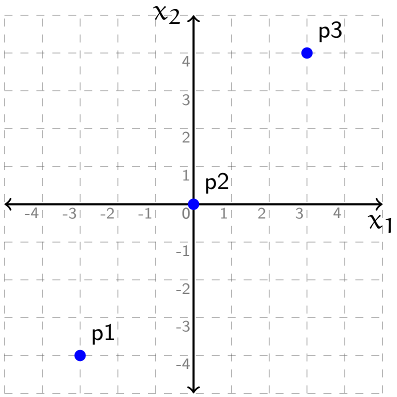
PCA: A toy example



Questions:

- How many dimensions do we have?
- How many dimensions do we need?

PCA: A toy example

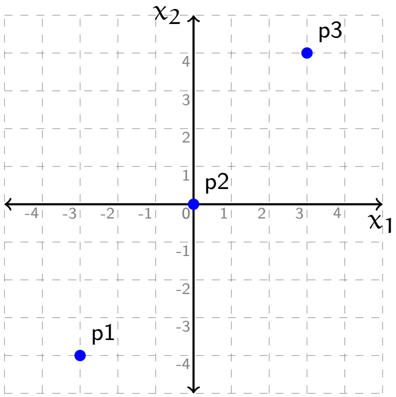


Questions:

- How many dimensions do we have?
- How many dimensions do we need?
- Short divergence: calculate the covariance matrix

$$\Sigma = \begin{bmatrix} ? & ? \\ ? & ? \end{bmatrix}$$

PCA: A toy example

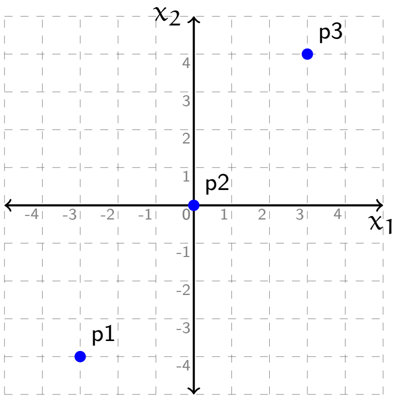


Questions:

- How many dimensions do we have?
- How many dimensions do we need?
- Short divergence: calculate the covariance matrix

$$\Sigma = \begin{bmatrix} \sigma^2_{x_1} & \sigma_{x_2,x_1} \\ \sigma_{x_1,x_2} & \sigma^2_{x_2} \end{bmatrix}$$

PCA: A toy example

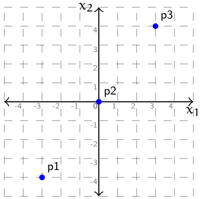


Questions:

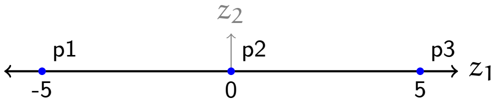
- How many dimensions do we have?
- How many dimensions do we need?
- Short divergence: calculate the covariance matrix

$$\Sigma = \begin{bmatrix} \frac{18}{3} & 8 \\ 8 & \frac{32}{3} \end{bmatrix}$$

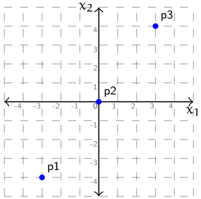
PCA: A toy example (2)



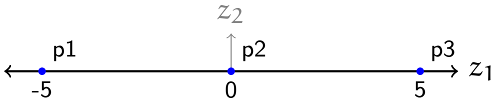
What if we reduce the data to:



PCA: A toy example (2)



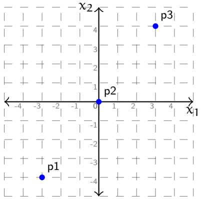
What if we reduce the data to:



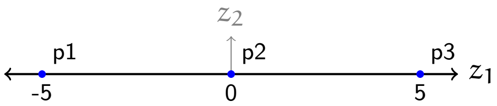
Going back to the original coordinates is easy, rotate using:

$$A = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} = \begin{bmatrix} \frac{3}{5} & -\frac{4}{5} \\ \frac{4}{5} & \frac{3}{5} \end{bmatrix}$$

PCA: A toy example (2)



What if we reduce the data to:

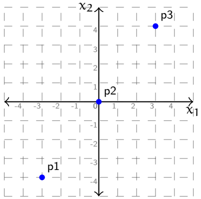


Going back to the original coordinates is easy, rotate using:

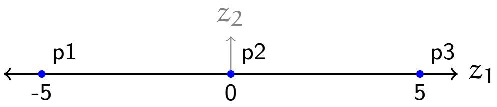
$$A = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} = \begin{bmatrix} \frac{3}{5} & -\frac{4}{5} \\ \frac{4}{5} & \frac{3}{5} \end{bmatrix}$$

$$p1 = A \times \begin{bmatrix} -5 \\ 0 \end{bmatrix} = \begin{bmatrix} -3 \\ -4 \end{bmatrix} \quad p1 = A \times \begin{bmatrix} 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} p1 = A \times \begin{bmatrix} 5 \\ 0 \end{bmatrix} = \begin{bmatrix} 3 \\ 4 \end{bmatrix}$$

PCA: A toy example (2)



What if we reduce the data to:



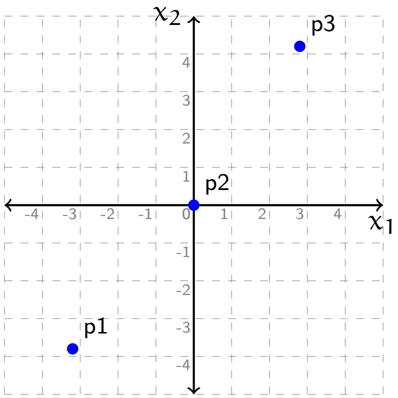
Going back to the original coordinates is easy, rotate using:

$$A = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} = \begin{bmatrix} \frac{3}{5} & -\frac{4}{5} \\ \frac{4}{5} & \frac{3}{5} \end{bmatrix}$$

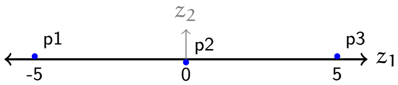
$$p1 = A \times \begin{bmatrix} -5 \\ 0 \end{bmatrix} = \begin{bmatrix} -3 \\ -4 \end{bmatrix} \quad p1 = A \times \begin{bmatrix} 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} p1 = A \times \begin{bmatrix} 5 \\ 0 \end{bmatrix} = \begin{bmatrix} 3 \\ 4 \end{bmatrix}$$

We can recover the original points perfectly. In this example the inherent dimensionality of the data is only 1.

PCA: A toy example (3)



- What if the variables were not perfectly but strongly correlated?
- We could still do a similar transformation:



- Discarding z_2 results in a small reconstruction error:

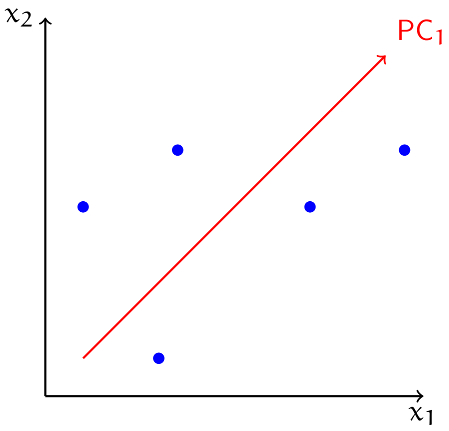
$$\mathrm{p1} = A \times \begin{bmatrix} -5 \\ 0 \end{bmatrix} = \begin{bmatrix} -3 \\ -4 \end{bmatrix}$$

- Note: z_1 (also z_2) is a linear combination of original variables

Why do we want to reduce the dimensionality

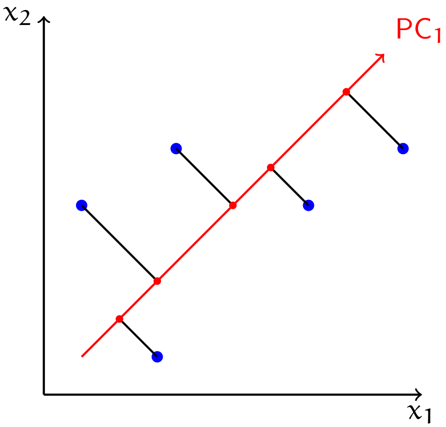
- Visualizing high-dimensional data becomes possible
- If we use the data for supervised learning, we avoid ‘the curse of dimensionality’
- Decorrelation is useful in some applications
- We compress the data (in a lossy way)
- We eliminate noise (assuming a high signal to noise ratio)

Different views on PCA



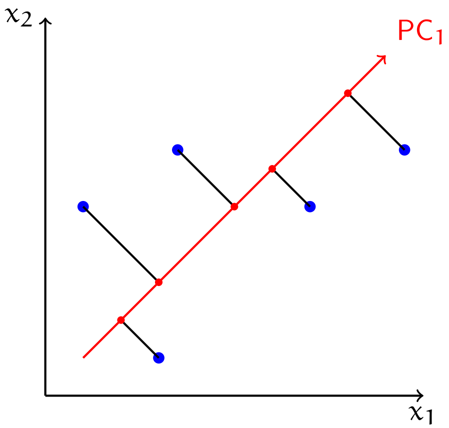
- Find the direction of the largest variance

Different views on PCA



- Find the direction of the largest variance
- Find the projection with the least reconstruction error

Different views on PCA



- Find the direction of the largest variance
- Find the projection with the least reconstruction error
- Find a lower dimensional latent Gaussian variable such that the observed variable is a mapping of the latent variable to a higher dimensional space (with added noise).

How to find PCs

- When viewed as *maximizing variance* or *reducing the construction error*, we can write the appropriate objective function and find the vectors that minimize it
- In latent variable interpretation, we can use EM as in estimating mixtures of Gaussians
- It turns out, the principle components are the eigenvectors of the correlation matrix, where large eigenvalues correspond to components with large variation
- A numerically stable way to obtain principal components is doing singular value decomposition (SVD) on the input data

PCA as matrix factorization (eigenvalue decomposition)

- One can compute PCA by decomposing the covariance matrix as (note $\mathbf{\Sigma} = \mathbf{X}^\mathsf{T}\mathbf{X}$)
$$\mathbf{\Sigma} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\mathsf{T}$$
  - the columns of $\mathbf{U}$ are the principal components (eigenvectors)
  - $\mathbf{\Lambda}$ is a diagonal matrix of eigenvalues
- Another option is SVD, which factorizes the input vector (k variables $\times$ n data points) as
$$\mathbf{X} = \mathbf{U}\mathbf{D}\mathbf{V}^*$$
  - $\mathbf{U}$ ($k \times k$) contains the eigenvectors as before,
  - $\mathbf{D}$ ($k \times k$) diagonal matrix $\mathbf{D}^2 = \mathbf{\Lambda}$
  - $\mathbf{V}^*$ is a $k \times n$ unitary matrix

* The above is correct for standardized variables, otherwise the formulas get slightly more complicated.

A practical example
(with simplified/fake data)

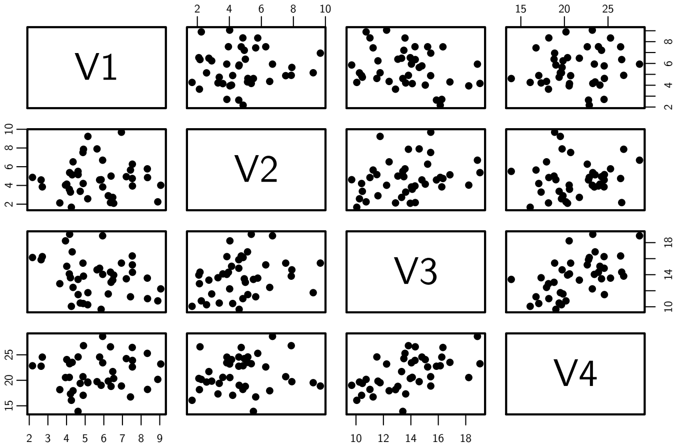
- Our data consists of 'measurements' from speech signal of instances of two vowels, we have 12 measurements for each vowel instance

$$\begin{bmatrix} 5.19 & 4.33 & 14.76 & 30.08 & 14.73 & 7.06 & 15.56 & 24.46 & 8.51 & \ldots \\ 2.99 & 5.25 & 11.69 & 19.27 & 18.02 & 11.04 & 13.34 & 38.13 & 8.70 & \ldots \\ 6.25 & 6.05 & 13.88 & 19.26 & 17.81 & 6.95 & 12.58 & 39.74 & 9.58 & \ldots \\ 7.24 & 5.43 & 15.15 & 18.93 & 15.69 & 10.18 & 14.89 & 34.86 & 10.03 & \ldots \\ 6.07 & 6.27 & 13.34 & 17.60 & 19.98 & 11.04 & 13.28 & 36.02 & 8.66 & \ldots \\ & & & & \ldots & & & & & \end{bmatrix}$$

- How do we visualize this data?
- Are all 12 variables useful?

A practical example

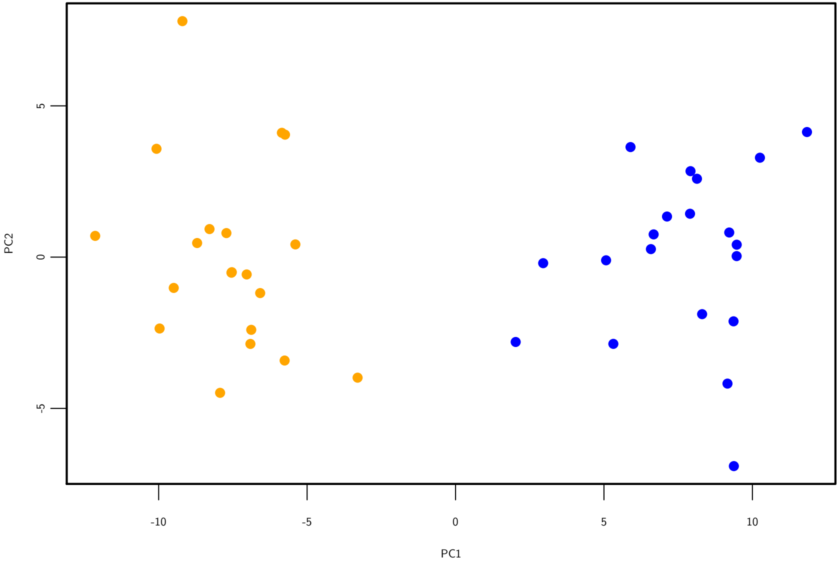
Visualizing with pairwise scatter plots



...

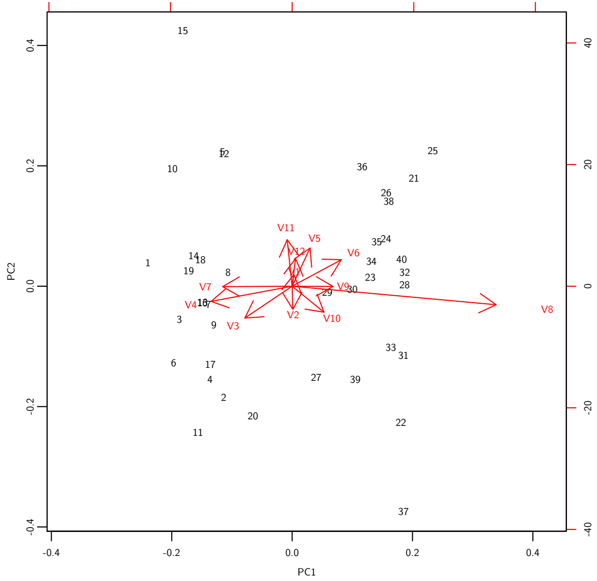
A practical example

Plotting the first two principal components



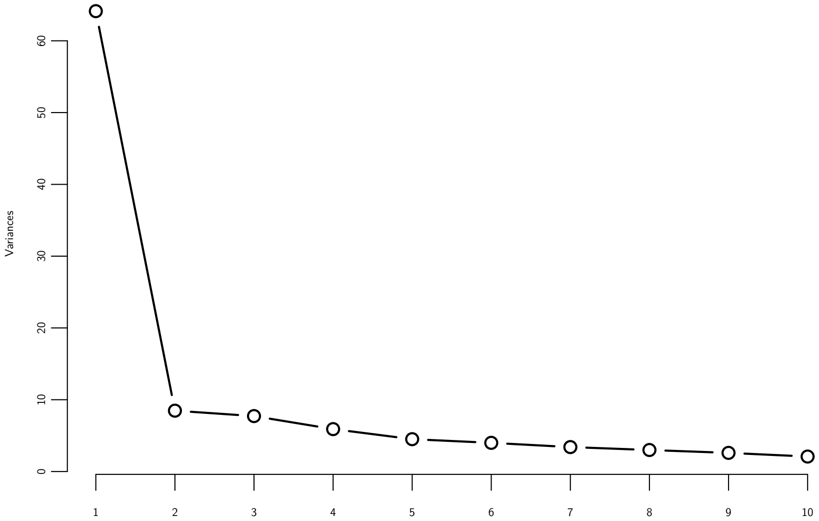
A practical example

Biplot



A practical example

How many components to keep? (scree plot)



Some practical notes on PCA

- Variables need to be centered
- Scales of the variables matter, standardizing may be a good idea depending on the units/scales of the individual variables
- The sign of the principal component (vector) is not important
- If there are more variables than the data points, we can still calculate the principal components, but there will be at most $n - 1$ PCs
- PCA will be successful if variables are linearly correlated, there are extensions for dealing with nonlinearities (e.g., kernel PCA, ICA)

Unsupervised learning: a summary (so far)

- In unsupervised learning, we do not have labels. Our aim is to find/exploit (latent) structure in the data
- We studied a number of related methods

  Clustering finds groups in the data

  Mixture densities are a 'soft' version of the clustering, assuming data is generated by a number of distributions

  Dimensionality reduction methods try to summarize the data with fewer variables/dimensions
- The evaluation of unsupervised methods are problematic, without knowing what we should exactly find in the data

Exercises with unsupervised learning

You can find the data set we will use on the course web page. The data a matrix with a phoneme on each row, and a context on each column. The cells are counts of the phoneme observed in the indicated context.

- Try both k-means and hierarchical clustering on the data set
- You can use
  - R: `kmeans` and `hclust` (you also need `dist` for calculating distances)
  - Python/sklearn: `sklearn.cluster` in python
- You may want to compare your results with IPA chart to see the clustering you observe has any linguistic basis
- Try different hierarchical clustering methods
- Try with and without normalization of the counts

Derivation of PCA by maximizing the variance

- We focus on the first PC (z_1), which maximizes the variance of the data onto itself
- We are interested only on the direction, so we choose z_1 to be a unit vector ($\|z_1\| = 1$)
- Remember that to project a vector onto another we simply use dot product, So the projected data points are zx_i for $i = 1, \ldots, N$.
- The variance of the projected data points (that we want to maximize) is,

$$\sigma_{z_1} = \frac{1}{N} \sum_{i}^{N} (\boldsymbol{z}_1 \boldsymbol{x}_i - \boldsymbol{z}_1 \bar{\boldsymbol{x}}_i)^2 = \boldsymbol{z}_1^T \Sigma \boldsymbol{z}$$

where Σ_x is the covariance matrix of the unprojected data

Derivation of PCA by maximizing the variance (cont.)

- The problem becomes maximize

$$z_1^T \Sigma z$$

with the constraint $\|z_1\| = z_1^T z_1 = 1$

- Turning it into a unconstrained optimization problem with Lagrange multipliers, we minimize

$$z_1^T \Sigma z + \lambda_1 (1 - z_1^T z_1)$$

- Taking the derivative and setting it to 0 gives us

$$\Sigma z_1 = \lambda_1 z_1$$

Note: by definition, z_1 is an eigenvector of Σ, and λ_1 is the corresponding eigenvalue

- z_1 is the first principal component, we can now compute the second principal component with the constraint that it has to be orthogonal to the first one