Clustering

Signal analyses

What are clustering algorithms?

What is clustering ?

Clustering of data is a method by which large sets of data is grouped into clusters of smaller sets of similar data.

Example:

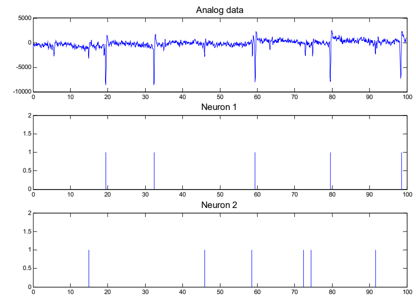
The balls of same color are clustered into a group as shown below :

Thus, clustering means grouping of data or dividing a large data set into smaller data sets of some similarity.

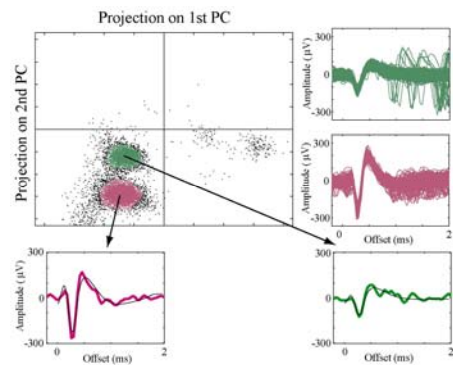
Outline

- K-means
- EM – Expectation Maximization
- Nonparametric pairwise similarity

Spike sorting I



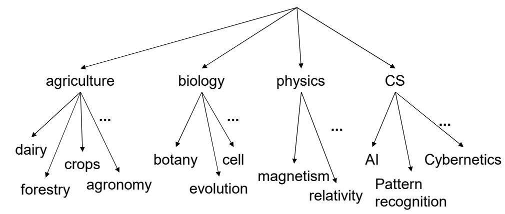
Spike sorting II



Supervised vs. unsupervised learning

- **Supervised Learning**
  - *Classification:* partition examples into groups according to pre-defined categories
  - *Regression:* assign value to feature vectors
  - Requires labeled data for training
- **Unsupervised Learning**
  - *Clustering:* partition examples into groups when no pre-defined categories/classes are available
  - *Novelty detection:* find changes in data
  - *Outlier detection:* find unusual events
  - Only instances required, but no labels

Supervised classification



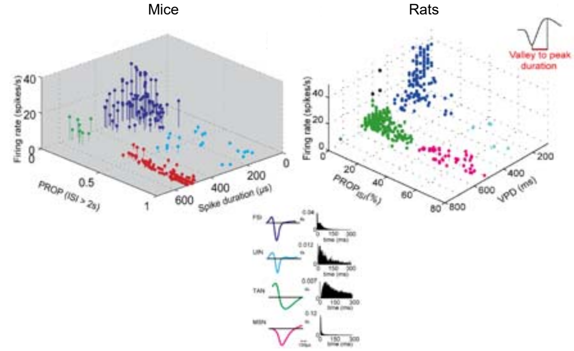
What is a good clustering?

- Internal criterion: a good clustering will produce high quality clusters in which:
  - the intra-cluster similarity is high
  - the inter-cluster similarity is low

  dependence on representation and the similarity measure used

- External criterion: The quality of a clustering is also measured by its ability to discover some or all of the hidden patterns or latent classes

Cell identification in the striatum



How hard is clustering?

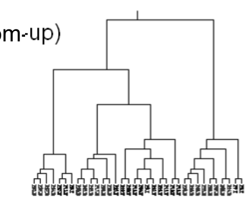
- One option is to consider all possible clusters, and pick the one that has best inter and intra cluster distance properties
- Suppose we are given n points, and would like to cluster them into k-clusters, the number of clusters is:

$$\frac{k^n}{k!}$$

- Too hard to do it optimally using brute force…
- Solution: Iterative optimization algorithms

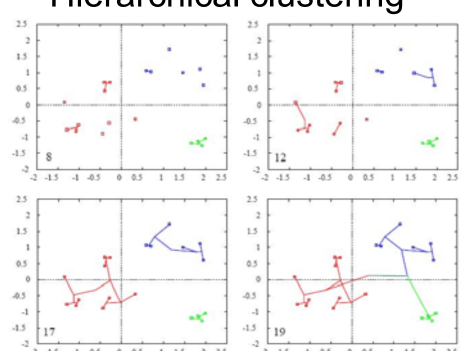
Clustering methods

- **Hierarchical**
  - Agglomerative (bottom-up)
  - Divisive (top-down)

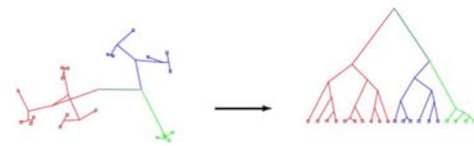


- **Partitioning**
  - K-means
  - Mixture of Gaussians

Hierarchical clustering



Hierarchical clustering



Data with clustering order and distances

Dendrogram representation

K-means clustering

Goal: partition the dataset into K disjoint "clusters" of data points

Algorithm:

- Start with random guess of where the K cluster centers $\mathbf{m}_1 \cdots \mathbf{m}_K$ are
- Repeat the following until cluster centers stop changing:
  - assign each data point to the nearest cluster:

    $p(n,k)=1$ if data point $\mathbf{x}^{(n)}$ is closer to $\mathbf{m}_k$ than to any other $\mathbf{m}_{j\neq k}$

  - move each cluster center to the **mean** of all data points assigned to it:

$$\mathbf{m}_k = \frac{\sum_n p(n,k)\mathbf{x}^{(n)}}{\sum_j p(j,k)}$$

← Vector sum of all data points assigned to cluster k

← Count of all data points assigned to cluster k

$$\mathbf{m}_k = \sum_n w(n,k)\mathbf{x}^{(n)} \quad \text{where} \quad w(n,k) \triangleq \frac{p(n,k)}{\sum_j p(j,k)}$$

Illustration of K-means clustering

initialization

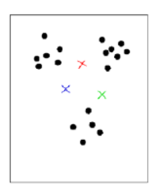


Illustration of K-means clustering

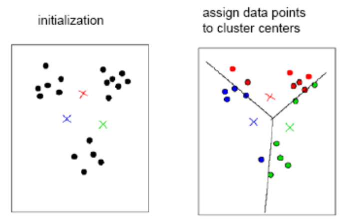


Illustration of K-means clustering

How to measure the distance

- Euclidean distance (L_2 norm):

$$L_2(\vec{x},\vec{x}') = \sum_{i=1}^{m} (x_i - x_i')^2$$

- L1 norm distance

$$L_1(\vec{x},\vec{x}') = \sum_{i=1}^{m} |x_i - x_i'|$$

- Cosine distance

$$\cos(\vec{x},\vec{x}') = \frac{\vec{x} \cdot \vec{x}'}{|\vec{x}| \cdot |\vec{x}'|}$$

- Cross correlations: Pearson's distance

$$d(x_i, y_i) = (1 - CC) = 1 - \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum (x_i - \bar{x})^2}\sqrt{\sum (y_i - \bar{y})^2}}$$

Comments on the K-means Methods

Strength of the K-means:

- *Relatively efficient:* $O(tkn)$*, where* n *is the number of objects,* k *is the number of clusters, and* t *is number of iterations. Normally,* $k,t << n$.
- *Often terminates at a local optimum.*

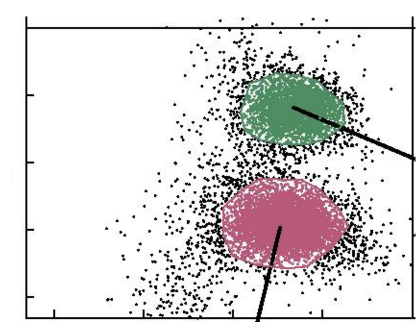
Weakness of the k-means:

- *Applicable only when mean is defined, then what about categorical data?*
- *Need to specify* k*, the number of clusters, in advance.*
- *Unable to handle noisy data and outlines.*
- *Not suitable to discover clusters with non-convex shapes.*

Soft Clustering

- Clustering typically assumes that each instance is given a "hard" assignment to exactly one cluster.
- Does not allow uncertainty in class membership or for an instance to belong to more than one cluster.
- *Soft clustering* gives probabilities that an instance belongs to each of a set of clusters.
- Each instance is assigned a probability distribution across a set of discovered categories (probabilities of all categories must sum to 1).

Spike sorting II



A better algorithm: Mixture-of-Gaussians clustering

When the data vectors are clustered, it is more appropriate to fit a distribution with multiple peaks. Consider the mixture-of-Gaussians distribution:

$$p(\mathbf{x}; \mathbf{m}_{1 \cdots K}, V_{1 \cdots K}) = \frac{1}{K} \sum_k g_k(\mathbf{x}; \mathbf{m}_k, V_k)$$

↑ mixture distribution

↑ Gaussian distributions, with means and covariances $\mathbf{m}_k, V_k$

How do we fit such a distribution to a set of data vectors $\mathbf{x}^{(1)} \ldots \mathbf{x}^{(N)}$? If we knew which Gaussian is "responsible" for each data vector, we could compute the mean and covariance separately for each Gaussian – from the vectors it is responsible for. This suggests the following iterative algorithm (the EM algorithm):

Iterate the following two steps until convergence:

- Expectation (E-step): Compute $P(x_i \mid E(g))$ for each example given the current model, and probabilistically re-label the examples based on these posterior probability estimates.
- Maximization (M-step): Re-estimate the model parameters from the probabilistically re-labeled data.

Expectation maximization

1. Compute the probability p(n,k) that data point n came from Gaussian k, and the normalized weights w(n,k) which sum to 1 for each Gaussian:

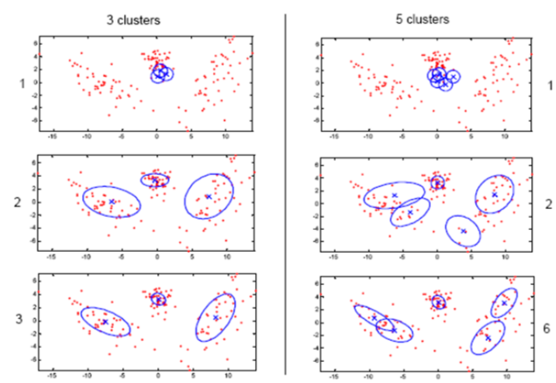
$$p(n,k) = g_k\left(\mathbf{x}^{(n)}\right) \Big/ \sum_j g_j\left(\mathbf{x}^{(n)}\right) \qquad w(n,k) = p(n,k) \Big/ \sum_j p(j,k)$$

2. Re-compute the mean and covariance of all data points that Gaussian k is responsible for, using w(n,k) as weights:

$$\mathbf{m}_k = \sum_n w(n,k)\,\mathbf{x}^{(n)} \qquad V_k = \sum_n w(n,k)\left(\mathbf{x}^{(n)} - \mathbf{m}_k\right)\left(\mathbf{x}^{(n)} - \mathbf{m}_k\right)^T$$

3. Repeat until $\mathbf{m}$, V no longer change.

Example of Mixture-of-Gaussians clustering



Comparison of the two algorithms

In both cases, we compute a quantity p(n,k) that tells us how well data point n fits in cluster k. Then we compute the normalized weights

$$w(n,k) = p(n,k) \Big/ \sum_j p(j,k)$$

and re-compute the cluster centers according to weighted center-of-mass calculation

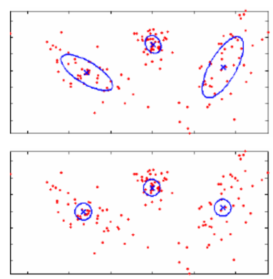
$$m_k = \sum_n w(n,k)\, \mathbf{x}^{(n)}$$

There are two differences:

1. In K-means the "fit" p is either 1 or 0, depending on which is the nearest cluster; In MOG, the values of p vary continuously between 0 and 1, and correspond to probabilities
2. In MOG we also re-compute the covariance matrix, which in turn affects how we determine the fit of data points to clusters; In K-means, the fit is always computed in the same way, corresponding to the assumption of circular clusters

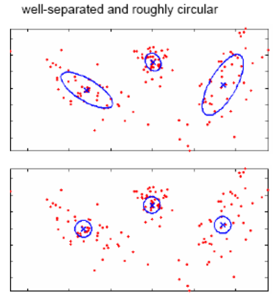
Mixture-of-Gaussians vs. K-means clustering

The results are similar when the clusters are well-separated and roughly circular

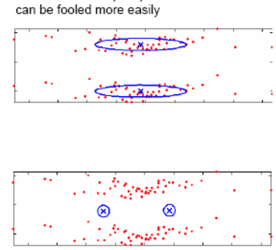


Mixture-of-Gaussians vs. K-means clustering

The results are similar when the clusters are well-separated and roughly circular



But for more complex problems K-means can be fooled more easily



A new nonparametric pairwise clustering algorithm based on iterative estimation of distance profiles

Shlomo Dubnov, Ran El-Yaniv, Yoram Gdalyahu, Elad Schneidman, Naftali Tishby, Golan Yona

CS at HUJI

Hierarchical algorithm

- We start with a set of data points {1,2, …, n}
- A symmetric proximity matrix M = $(d_{ij})_{i,j = 1..n}$ is given where d_{ij} is the pairwise (dis)similarity between points i and j.
- If $v = (v_1,v_2,\ldots,v_n)$ is an n-dimensional vector then the length of the vector is $||v||$
- We define dist (u,v) as the proximity measure between two given vectors in sample space

- A 2 step transformation of the similarity matrix:
  - **Normalization:** for each data point *i* we define the distance from all the other points

  $\mathbf{d}_i = (d_{i1}, d_{i2}, \ldots, d_{in})$ ($\mathbf{d}_i$ *is the* i^{th} *column of M)*
  then each $\mathbf{d}_i$ *is divided by its norm* $||\mathbf{d}_i||$ *so that*
  $\mathbf{p}_i = (p_{i1}, p_{i2}, \ldots, p_{in})$ *where* $p_{ij} = \mathbf{d}_i / ||\mathbf{d}_i||$

  - **Re-estimation:** recalculate the distance between points i and j $d^{new}_{ij} = \text{dist}(\mathbf{p}_i, \mathbf{p}_j)$.

- $M^{new}_{ij} = d^{new}_{ij}$ where i, j = 1..n
- Turns out that this algorithm converges fast to a two-valued matrix!

How do we define a distance between two distributions?

- The Kullback – Leibler (KL) divergence is a statistical measure between distributions
- For 2 distributions p_i and p_j the KL divergence is:

$$D^{KL}[\mathbf{p}_i \| \mathbf{p}_j] = \sum_k p_{ik} \log_2 \frac{p_{ik}}{p_{jk}}$$

- However this measure is asymmetrical and unbound

The Jensen-Shannon divergence

- Given two empirical probability distributions (samples) p(x) and q(x) their J-S divergence is defined as:

$$D_\lambda^{JS}[p(x)\|q(x)] = \lambda D^{KL}[p(x)\|r(x)] + (1-\lambda)D^{KL}[q(x)\|r(x)]$$

where $r(x) = \lambda p(x) + (1-\lambda)q(x)$

$$d_{ij}^{new} = D_{\frac{1}{2}}^{JS}[\mathbf{p}_i \| \mathbf{p}_j] = \frac{1}{2}\left(\sum_k p_{ik} \log \frac{p_{ik}}{\frac{1}{2}(p_{ik}+p_{jk})} + \sum_k p_{jk} \log \frac{p_{jk}}{\frac{1}{2}(p_{jk}+p_{ik})}\right)$$

Step 1. Each point is represented by its relation to all other data points

Step 2. the pairwise distance is re-estimated using a statistically motivated proximity measure.

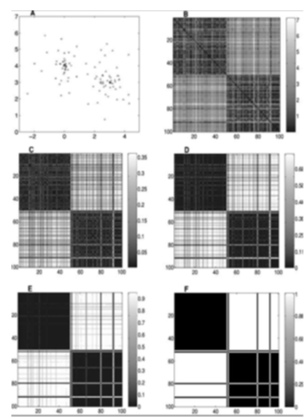
$$p_i = \left(\frac{d_{i1}}{\sum_k d_{ik}} \;,\; \frac{d_{i2}}{\sum_k d_{ik}} \;\cdots\; \frac{d_{in}}{\sum_k d_{ik}} \right)$$

Each vector of distances is transformed into a Probability distribution over the set of data points By normalizing it using the L_1 norm.

$$d_{ij}^{new} = D_{\frac{1}{2}}^{JS}\left[p_i \| p_j\right]$$

The Jensen-Shannon divergence is a modification on the Kullback-Leibler (KL) divergence. It is used to measure The statistical similarity between the distributions p_i and p_j

Data points sampled from two Gaussians



Cross-validated pairwise hierarchical clustering

- Randomly partition data set S into 3 subgroups

$|S1| \approx |S2| \approx |S3| \approx n/3$ $S = S1 \cup S2 \cup S3$

Let $A = S1 \cup S2$ and $B = S2 \cup S3$

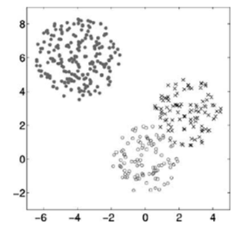
So that $A \cap B = S2$

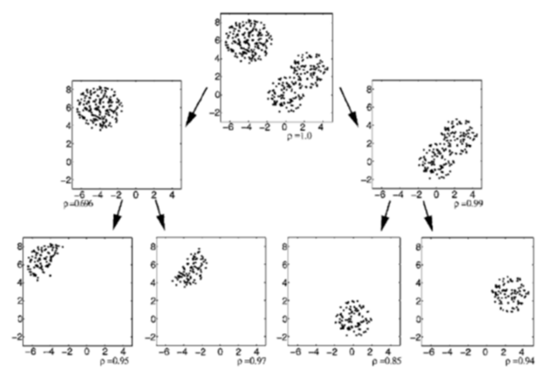
Run the algorithm on A and B and count m – the points in S2 that were clustered similarly in both runs

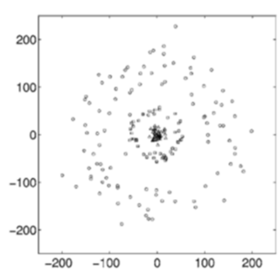
Define $\rho = m/|S2|$ the cross validation index

The cross validation index will be large for structured data set and small for unstructured data set.

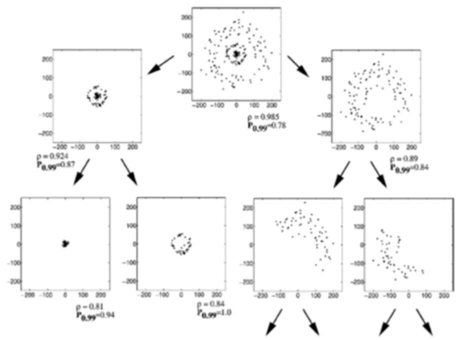
Cross-validated pairwise hierarchical clustering







Cross-validated hierarchical clustering of three concentric rings.



How to apply this method to neural activity

Cohen, D. and Nicolelis, M. A. *JNS* (2004).

Calculating the distance between two trials

$$P(v|r)=\frac{e^{-r}r^{v}}{v!}$$

The probability that a neuron fired *v* spikes while its average firing rate is *r*, was calculated assuming a Poisson distribution

$$P(v_i,v_j|r)=P\left(v_i\middle|\frac{v_i+v_j}{2}\right)*P\left(v_j\middle|\frac{v_i+v_j}{2}\right)$$

The rate vector that is most likely to yield a given spike count during two independent trials (v_i,v_j) is the average of the two spike counts.

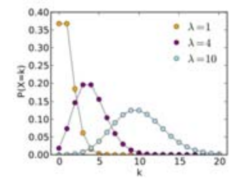
$$d_{ij}=\log\left(\prod_{\{n\}}P(v_{n,i}|r_{n,ij})*P(v_{n,j}|r_{n,ij})\right)$$

$$d_{ij}=\sum_{\{n\}}\left(\log(P(v_{n,i}|r_{n,ij}))+\log(P(v_{n,j}|r_{n,ij}))\right)$$

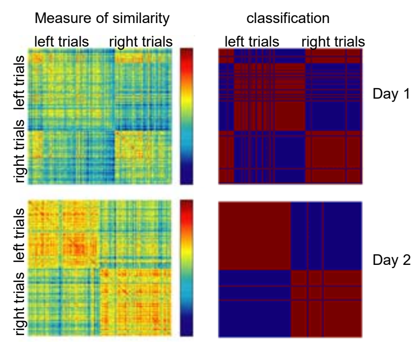
The similarity of two trials d_{ij} is taken as the log-probability that the corresponding spike count vectors were generated independently by the same maximum likelihood rate vector calculated for all the neurons together. d_{ij} is called the similarity matrix

$$p_i=\left(\frac{d_{i1}}{\sum_k d_{ik}},\ \frac{d_{i2}}{\sum_k d_{ik}}\ \cdots\ \frac{d_{in}}{\sum_k d_{ik}}\right)$$

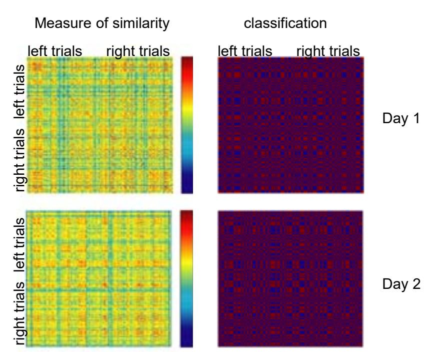
Each vector of distances is transformed into a Probability distribution over the set of data points By normalizing it using the L_1 norm.



Classification according to degree of similarity during movement



Classification according to degree of similarity during the waiting period



Dimensionality reduction

- For example:
  - PCA