Neural Networks and Learning Systems
TBMI 26, 2017

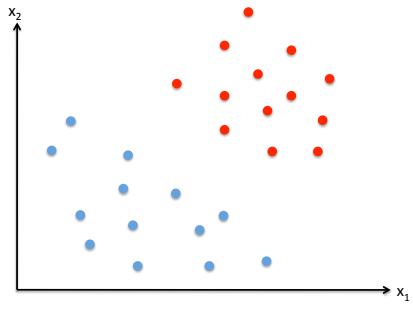
Lecture 8

Clustering & Genetic Algorithms

Magnus Borga
magnus.borga@liu.se

Clustering

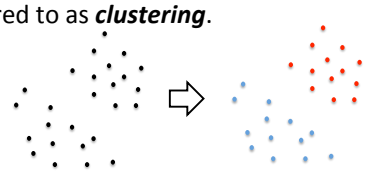
Supervised learning – labeled samples



Unsupervised learning – unlabeled samples

Categorization

- Categorization and grouping of objects based on similar properties is an important functionality in learning and knowledge representation.
- In the machine learning area, this is usually referred to as ***clustering***.



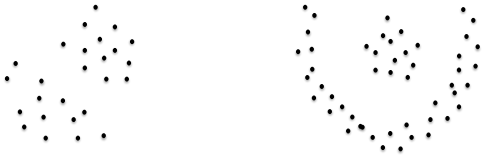
Remember, the computer sees this….

$$\mathbf{X} = \begin{bmatrix} 2.3 & 1.1 & 5.6 & 9.8 & 4.5 & 7.7 & 10.1 & 2.2 & 9.3 & \cdots \\ -1.4 & -4.5 & 2.0 & 1.2 & -0.4 & -4.3 & 7.0 & -3.2 & -1.0 & \cdots \\ 43.2 & 36.3 & 54.6 & 45.3 & 66.3 & 23.9 & 42.8 & 34.3 & 51.2 & \cdots \\ 0.1 & -0.5 & 0.4 & 0.2 & 0.2 & -0.2 & 0.8 & 0.5 & -0.7 & \cdots \end{bmatrix}$$

One feature vector **x**….

What describes a cluster?

- Distances to other points?
- Connectivity?
- Different definitions lead to different algorithms.



Distance in feature space d(**p**,**s**)

- Euclidian (l_2-norm)

$$d(\mathbf{p},\mathbf{s}) = \|\mathbf{p}-\mathbf{s}\|_2 = \sqrt{\sum_i (p_i - s_i)^2}$$

- Manhattan (l_1-norm)

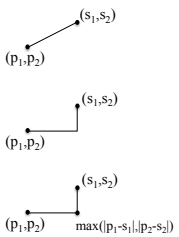
$$d(\mathbf{p},\mathbf{s}) = \|\mathbf{p}-\mathbf{s}\|_1 = \sum_i |p_i - s_i|$$

- Max (l_∞-norm)

$$d(\mathbf{p},\mathbf{s}) = \|\mathbf{p}-\mathbf{s}\|_\infty = \max |p_i - s_i|$$

- Weighted Euclidian

$$d(\mathbf{p},\mathbf{s}) = \|\mathbf{W}(\mathbf{p}-\mathbf{s})\|_2 = \sqrt{(\mathbf{p}-\mathbf{s})^T \mathbf{W}^2 (\mathbf{p}-\mathbf{s})} = \sqrt{\sum_i w_{ii}^2 (p_i - s_i)^2}$$



Also known as the Mahalanobis distance!

Hard vs soft clustering

- Hard clustering – each data point belongs only to one cluster.
- Soft/fuzzy clustering – a data point can belong to several clusters to certain degrees.

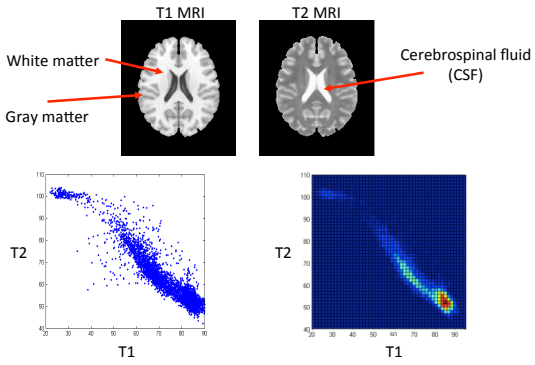
k-Means algorithm

- Assume *k* clusters (user input).
- Represent each cluster with a mean prototype vector $\mathbf{p}_j$ at the cluster center.
- A data point belongs to the cluster with the closest prototype vector (Euclidian distance).

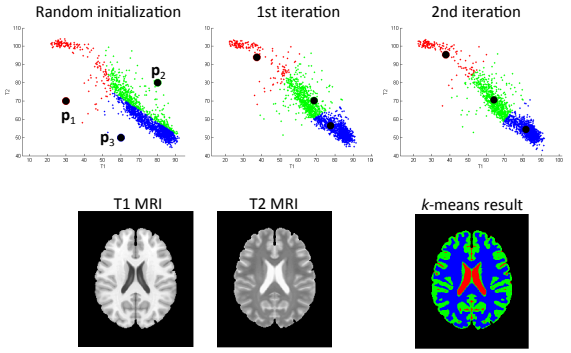
k-Means algorithm, cont.

1. Start with *k* random prototype vectors $\mathbf{p}_j$
2. Iterate:
   1. Assignment: Assign each data vector $\mathbf{x}_i$ to the closest prototype vector $\mathbf{p}_j$. Denote the set of data vectors assigned to cluster $\mathbf{p}_j$ by S_j.
   2. Update all prototype vectors to the mean of the clusters: $\mathbf{p}_j = \frac{1}{|S_j|}\sum_{k \in S_j} \mathbf{x}_k$

k-Means - Example



k-Means - Example



k-Means - Discussion

- Must specify *k*
- Tries to minimize the cost function

$$\varepsilon(\mathbf{p}_1,\ldots\mathbf{p}_k,S_1,\ldots,S_k)=\sum_{i=1}^{k}\sum_{\mathbf{x}_j\in S_i}\left\|\mathbf{x}_j-\mathbf{p}_i\right\|^2$$

- Note that this function is not differentiable as the S_i:s are discrete sets, i.e., we cannot do gradient descent.

Expectation Maximization (EM)- Intro

$$\varepsilon(\mathbf{p}_1,\ldots\mathbf{p}_k,\underbrace{S_1,\ldots,S_k}_{\text{Unknown class labels}})=\sum_{i=1}^{k}\sum_{\mathbf{x}_j\in S_i}\left\|\mathbf{x}_j-\mathbf{p}_i\right\|^2$$

Assume we know $S_1,..,S_k$!

$$\frac{\partial\varepsilon}{\partial\mathbf{p}_i}=\frac{\partial}{\partial\mathbf{p}_i}\left(\sum_{\mathbf{x}_j\in S_i}\left\|\mathbf{x}_j-\mathbf{p}_i\right\|^2\right)=-2\sum_{\mathbf{x}_j\in S_i}\left(\mathbf{x}_j-\mathbf{p}_i\right)=2|S_i|\mathbf{p}_i-2\sum_{\mathbf{x}_j\in S_i}\mathbf{x}_j$$

$$\frac{\partial\varepsilon}{\partial\mathbf{p}_i}=0 \quad\Rightarrow\quad \mathbf{p}_i=\frac{1}{|S_i|}\sum_{\mathbf{x}_j\in S_i}\mathbf{x}_j$$ Mean vector!

Expectation Maximization (EM)- Intro

$$\varepsilon(\underbrace{\mathbf{p}_1,\ldots\mathbf{p}_k}_{\text{Continuous parameters}},S_1,\ldots,S_k)=\sum_{i=1}^{k}\sum_{\mathbf{x}_j\in S_i}\left\|\mathbf{x}_j-\mathbf{p}_i\right\|^2$$

Now, assume we know $\mathbf{p}_1,..,\mathbf{p}_k$!

Each $\mathbf{x}_j$ independently contributes a distance $\|\mathbf{x}_j\text{-}\mathbf{p}_i\|$ to ε

Obvious that ε is minimized if each $\mathbf{x}_j$ is assigned to the set S associated with the closest $\mathbf{p}$!

k-Means algorithm!!

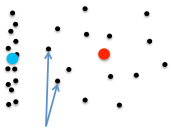
Expectation Maximization

- The *k*-means algorithm is a special instance of a general optimization approach called *Expectation Maximization* (EM).
- EM can be used when we want to estimate model parameters (the prototypes $\mathbf{p}_i$), but for each data sample $\mathbf{x}_j$ there is a hidden/missing parameter (the class labels S_j).
- EM iterates between optimizing the hidden parameters and the model parameters.

Mixture of Gaussians (MoG) clustering

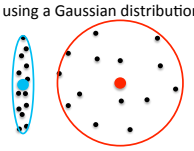
Another application of EM!

k-means



May not be correctly clustered!

Idea: Represent each cluster using a Gaussian distribution



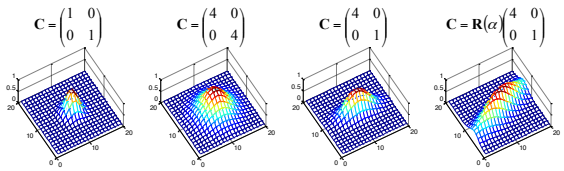
Problem: Each data sample $\mathbf{x}_j$ belongs to one of *k* Gaussian distributions N($\mathbf{p}_i$,$\mathbf{C}_i$), *i*=1..*k*.
Find the sets S_i of samples that belong to distribution N($\mathbf{p}_i$,$\mathbf{C}_i$) and the mean $\mathbf{p}_i$ and covariance matrix $\mathbf{C}_i$.

The Gaussian distribution

$$f(\mathbf{x};\mathbf{p}_i,\mathbf{C}_i) = \frac{1}{(2\pi)^{d/2}|\mathbf{C}_i|^{1/2}} e^{-\frac{1}{2}(\mathbf{x}-\mathbf{p}_i)^T\mathbf{C}_i^{-1}(\mathbf{x}-\mathbf{p}_i)}$$

Dimension

Determinant of covariance matrix



Let's use EM!

Assume we know the hidden parameters S_1,..,S_k!

That is, we know the samples for each set, e.g., S_1 = {$\mathbf{x}_1$, $\mathbf{x}_7$, $\mathbf{x}_{12}$, $\mathbf{x}_{13}$}

We can then estimate the mean $\mathbf{p}_i$ and covariance $\mathbf{C}_i$ using standard estimation for the Gaussian:

$$\mathbf{p}_i = \frac{1}{|S_i|}\sum_{k\in S_i}\mathbf{x}_k$$

$$\mathbf{C}_i = \frac{1}{|S_i|}\sum_{k\in S_i}(\mathbf{x}_k-\mathbf{p}_i)(\mathbf{x}_k-\mathbf{p}_i)^T$$

Let's use EM, cont!

Assume now that we know the Gaussian distribution parameters, i.e., we know all distributions

$$f(\mathbf{x};\mathbf{p}_i,\mathbf{C}_i) = \frac{1}{(2\pi)^{d/2}|\mathbf{C}_i|^{1/2}} e^{-\frac{1}{2}(\mathbf{x}-\mathbf{p}_i)^T\mathbf{C}_i^{-1}(\mathbf{x}-\mathbf{p}_i)}$$

Let the hidden parameter S_i be the set of all data samples for which $f(\mathbf{x};\mathbf{p}_i,\mathbf{C}_i)$ is larger than for all other distributions. That is, each data sample is assigned to the Gaussian distribution **to which it most likely belongs**!

Mixture of Gaussians - Algorithm

1. Start with k random Gaussians ($\mathbf{p}_j, \mathbf{C}_j$)
2. Iterate:
   1. Assign each sample $\mathbf{x}_i$ to the most likely Gaussian

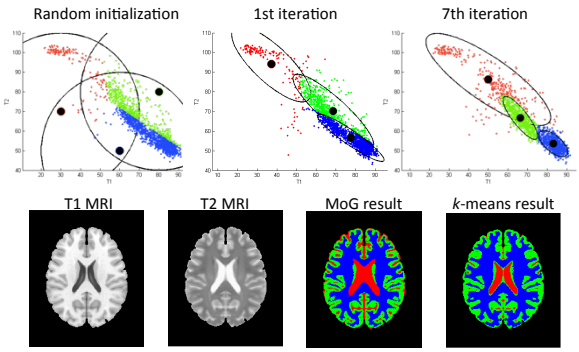
      $$\max f(\mathbf{x}_i;\mathbf{p}_j,\mathbf{C}_j) = \frac{1}{(2\pi)^{d/2}|\mathbf{C}_j|^{1/2}} e^{-\frac{1}{2}(\mathbf{x}_i-\mathbf{p}_j)^T\mathbf{C}_j^{-1}(\mathbf{x}_i-\mathbf{p}_j)}$$

      Denote the set of samples assigned to Gaussian j by S_j.
   2. Update all Gaussian means and covariances:

      $$\mathbf{p}_j = \frac{1}{|S_j|}\sum_{k\in S_j}\mathbf{x}_k$$

      $$\mathbf{C}_j = \frac{1}{|S_j|}\sum_{k\in S_j}(\mathbf{x}_k-\mathbf{p}_i)(\mathbf{x}_k-\mathbf{p}_i)^T$$

Mixture of Gaussians - Example



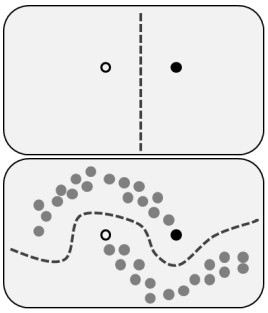
Summary of *k*-Means and MoG clustering

- Must choose the number of clusters *k* manually.
- Different initializations may give different results.
- May converge to degenerate solutions, e.g., empty clusters.
- MoG allows for elliptic cluster shapes with different sizes (using the Mahalanobis distance).

Semi-supervised learning

- Supervised learning requires labeled training data.
- Labeled data are limited or expensive!
- Use a mix of labeled and un-labeled data
- Use e.g. clustering to label the un-labeled data

Semi-supervised learning



From Wikipedia

Genetic algorithms

Genetic algorithms a.k.a. Evolutionary algorithms

- Biologically inspired optimization method – simulated evolution.
- Evolve a population of solution candidates by applying biological evolution rules:
  - Natural selection - survival of the fittest
  - Reproduction
  - Mutation
- Used for difficult optimization problems which standard optimization approaches don't handle well.

Real-life applications

- Design
  - Electronic circuits
  - Mechanical constructions
  - Chemistry (molecular design)

- Logistics
  - Transportation
  - Routing

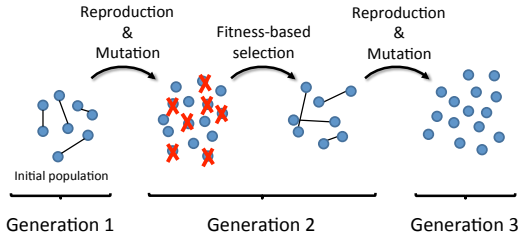
Telecommunication lines
Oil pipelines
Central warehouses
Data networks

- Scheduling

Airports
Schools
etc.

- Discrete problems.
- May have many constraints to consider.
- Relatively easy to evaluate how good a solution is.

Genetic Algorithm

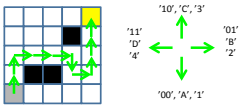


GA algorithm outline

1. Init a population of candidates
2. Evolve *T* generations
   1. Evaluate fitness of each candidate
   2. Sample a new population based on the fitness
   3. Mutate candidates in the new population
   4. Cross-over (mating) between candidates
3. Select best candidate after *T* generations as the solution.

Representation

String coding:



- **Binary**: '10100101010001101010'
- **Letters**: 'CCBBBABCCC'
- **Integers**: '3322212333'

'1467325' Permutation, one of each number, e.g, scheduling

Real numbers:

x = 4.8

Mutation

Random changes in a gene

String coding:

Mutate each position with a small probability p_m

'CCBBBABCCC' → 'CABBBABCCC'

Permutation: 1 4 6 7 3 2 5 → 1 3 6 7 4 2 5

Real numbers:

$x = 4.8 \rightarrow x = 4.8 + \eta$

Small random perturbation

Reproduction / Cross-over / Recombination

String coding:

1-point cross-over

'CCBBB|ABCCC' → 'CCBBBCACAC'

'ABCCC|CACAC' → 'ABCCCABCCC'

Uniform cross-over

'CCBBBABCCC' → 'CCBBBABCCC' → 'CBBBCAACCC'

'ABCCCCACAC' → 'ABCCCCACAC' → 'ACCCBCBCAC'

Reproduction / Cross-over / Recombination, cont

String coding, cont:

Permutation

P1: 7 6 5 4 3 2 1

P2: 1 4 6 7 3 2 5

Take a subsequence from P1 and add missing numbers from P2.

1 6 5 4 7 3 2

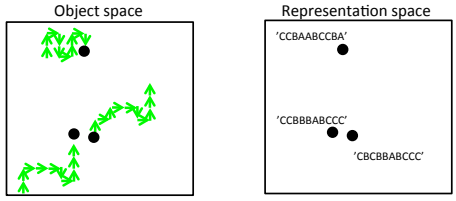
Real numbers:

x = 4.8
y = 8.9

→ $\alpha x + (1-\alpha)y$

More on representation

- A small change in the "object space" should correspond to a small change in the representation space, i.e., it should be possible to reach similar "objects" with a small mutation in the representation.
- A mutation that creates a large modification of the object is not so much of a problem



More on representation, cont.

Example: Encoding of integer numbers

Dec.	Binary coding	Gray coding
0	'000'	'000'
1	'001'	'001'
2	'010'	'011'
3	'011'	'010'
4	'100'	'110'
5	'101'	'111'
6	'110'	'101'
7	'111'	'100'

- For example, in binary coding, all three positions must be mutated to get from 3 to 4.
- In the Gray coding, jumps to neigboring integers always correspond to a mutation in only one position!

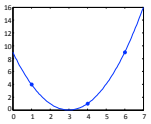
Fitness and selection

Sample individuals for the next generation based on their fitness

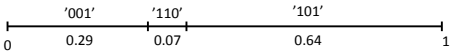
Example: Maximize the function $(x-3)^2$

Initial pop.	x-value	Fitness $f(x) = (x-3)^2$	Probability
'001'	1	4	4/14 = 0.29
'110'	4	1	1/14 = 0.07
'101'	6	9	9/14 = 0.64
		Sum: 14	Sum: 1.0

Gray coding



Sampling: Draw uniform random number in [0,1] and select the individual corresponding to interval.



The Schema Theorem

Schema = a subset of strings having identical values in certain positions

Ex. * 1 * 0 is a schema representing

0 1 0 0
0 1 1 0
1 1 0 0
1 1 1 0

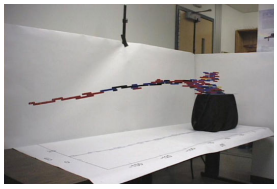
Locked values

The Schema Theorem

- Suppose individuals are selected for reproduction with a probability proportional to their fitness.
- **Schemas representing individuals that are <u>better</u> than average <u>grow exponentially</u> in numbers.**
- Schemas representing individuals that are <u>worse</u> than average <u>decrease exponentially</u> in numbers.

More examples…

LEGO construction



Evolving creatures

Genetic Algorithms - Summary

- The representation must permit meaningful mutations and cross-overs, otherwise the GA is reduced to a pure random search.
- Never guaranteed to actually find the optimal solution – GAs are (hopefully) better than a pure random search.
- Can be used as optimization technique in machine learning problems, e.g., for neural networks.