Machine Learning

Clustering, Self-Organizing Maps
Clustering

The task: partition a set of objects into “meaningful” subsets (clusters). The objects in a subset should be “similar”.

Notations:

Set of Clusters

\[ K \]

Set of indices

\[ I = \{1, 2, \ldots, |I|\} \]

Feature vectors

\[ x^i, i \in I \]

Partitioning

\[ C = (I_1, I_2, \ldots, I_{|K|}), \quad I_k \cap I_{k'} = \emptyset \quad \text{for} \quad k \neq k', \quad \bigcup_k I_k = I \]
Let $x^i \in \mathbb{R}^n$ and each cluster has a “representative” $y^k \in \mathbb{R}^n$.

The task reads:

$$\sum_{k} \sum_{i \in I_k} \|x^i - y^k\|^2 \rightarrow \min_{C, y}$$

Alternative variant is to consider the clustering $C$ as a mapping $C : I \rightarrow K$ that assigns a cluster number to each $i \in I$.

$$\sum_{i} \|x^i - y^{C(i)}\|^2 \rightarrow \min_{y, C}$$

$$\sum_{i} \min_{k} \|x^i - y^k\|^2 \rightarrow \min_{y}$$
K-Means Algorithm

Initialize centers randomly,

Repeat until convergence:

1. Classify:

\[
C(i) = \arg \min_{k'} \| x^i - y^{k'} \|^2 \quad \Rightarrow \quad i \in I_k
\]

2. Update centers:

\[
y^k = \arg \min_y \sum_{i \in I_k} \| x^i - y \|^2 = \frac{1}{|I_k|} \sum_{i \in I_k} x^i
\]

- The task is NP
- converges to a local optimum (depends on the initialization)
Sequential K-Means

Repeat infinitely:

1. Chose randomly a feature vector $x$ from the training data

2. Classify it:
   $$k = \arg \min_{k'} ||x - y^{k'}||$$

3. Update the $k$-th center:
   $$y^k = y^k + \eta(t)(x - y^k)$$

with a decreasing step $\eta(t)$

- converges to the same, as the parallel version
- is a special case of Robbins-Monro Algorithm
Some variants

Other distances, e.g. $\|x^i - y^k\|$ instead of $\|x^i - y^k\|^2$

In the K-Means algorithm the classification step remains the same, the update step – the geometric median of $x^i, i \in I_k$

$$y_k = \arg \min_y \sum_{i \in I_k} \|x^i - y\|$$

(a bit complicated as the average 😞).

Another problem: features may be not additive ($y^k$ does not exist)

Solution: K-Medioid Algorithm ($y^k$ is a feature vector from the training set)
A generalization

Observe (for the Euclidean distance):

\[ \sum_i \| x^i - \bar{x} \|^2 \sim \sum_{ij} \| x^i - x^j \|^2 \]

In what follows:

\[ \sum_k \sum_{ij \in I_k} \| x^i - x^j \|^2 = \sum_k \sum_{ij \in I_k} d(i, j) \rightarrow \min_C \]

with a Distance Matrix \( d \) that can be defined in very different ways.

Example: Objects are nodes of a weighted graph, \( d(i, j) \) is the length of the shortest path from \( i \) to \( j \).

Distances for “other” objects (non-vectors):
- Edit (Levenshtein) distance between two symbolic sequences
- For graphs – distances based on graph isomorphism etc.
An application – color reduction

Objects are pixels, features are RGB-values. Partition the RGB-space into “characteristic” colors.

(8 colors)
Another application – superpixel segmentation

Object are pixels. Features are RGBXY-values.

→ Those pixels belong to the same cluster that are close to each other both spatially and in in the RGB-space

SLIC Superpixels: http://ivrg.epfl.ch/research/superpixels
The task is to “approximate” a dataset by a neural network of a certain **topology**.

An example – stereo in “flatland”.

The input space is 3- (or more) dimensional, the set of points is however isomorphic to a 2D-space (up to noises).
Self-Organizing Maps

SOM-s (usually) consist of RBF-neurons \( r \), each one represents (covers) a “part” of the input space (specified by the centers \( \mu^r \)).

The network topology is given by means of a distance \( d(r, r') \). Example – neurons are nodes of a weighted graph, distances are shortest paths. For the “flatland” example the graph is a 2D-grid with unit weight for all edges.
1. Chose randomly a feature vector $x$ from the training data (white)
2. Compute the “winner”-neuron (dark-yellow)
   \[ r^* = \arg \min_r ||x - \mu^r|| \]
3. Compute the neighborhood of $r^*$ in the network (yellow)
   \[ R = \{ r | d(r^*, r) < \Theta \} \]
4. Update the weights of all neurons from $R$
   \[ \mu^r = \mu^r + (x - \mu^r) \cdot \eta(t, d(r^*, r)) \]
 Self-Organizing Maps, algorithms

\( \eta(t, d) \) is monotonously decreasing with respect to \( t \) (time) \textbf{and} \( d \)

Without 3) – the sequential K-Means.

Parallel variants:
Go through all feature vectors, sum up the gradients, apply.

Example for \( \mathbb{R}^2 \rightarrow \mathbb{R}^2 \):

The network fits into the data distribution (unfolds).
K-Means ↔ Expectation Maximization

EM – compute posteriors for the Expectation step

K-Means – classify object
Conclusion

Before:
1. Probability theory – models, inference, learning
2. → Discriminative learning → Classifiers

Neural networks:
1. Feed-Forward Networks – complex classifiers
2. Hopfield Networks – structured output
3. Cohonen Networks – clustering (unsupervised), model fitting

Next topics – further classifiers:
1. Support Vector Machines, Kernels
2. Empirical Risk minimization
3. Principal Component Analysis