K-means and Expectation Maximization

Mike Bianco and Peter Gerstoft
ECE228
5/6/2019
K-means and expectation maximization (EM) can be considered unsupervised learning

• In supervised learning, we have desired machine learning (ML) model output or ‘action’ $y$ based on inputs $x$ (features), and model parameters $\theta$
  – Probabilities of the form: $p(y|x,\theta)$
  – Linear regression and classification, support vector machines, etc.

• In unsupervised learning, we are interested in discovering useful patterns in the features. This can be for discovering latent data ‘causes’ or significant ‘groups’
  – Probabilities of the form: $p(x|\theta)$
  – Principal components analysis (PCA), K-means, dictionary learning, etc.
Unsupervised learning

Unsupervised machine learning is inferring a function to describe hidden structure from "unlabeled" data (a classification or categorization is not included in the observations). Since the examples given to the learner are unlabeled, there is no evaluation of the accuracy of the structure that is output by the relevant algorithm—which is one way of distinguishing unsupervised learning from supervised learning.

We are not interested in prediction

Supervised learning: all classification and regression.

\[ Y = w^T X \]

Prediction is important.
Supervised learning: least square classifier (binary)

Training set \{ (x^1, y^1), (x^2, y^2), (x^3, y^3) \}

We are given the two classes
(green = 0, red = 1)
Unsupervised learning: how are features best divided?

Just have features \( \{(x_1^1, x_2^1), (x_1^2, x_2^2), (x_1^3, x_2^3)\} \)
K-means

• **Input:** Points $\mathbf{x}_1,...,\mathbf{x}_N \in \mathbb{R}^p$; integer $K$
• **Output:** “Centers”, or representatives, $\mu_1,...,\mu_K \in \mathbb{R}^p$
• Output also $\mathbf{z}_1,...,\mathbf{z}_N \in \mathbb{R}^K$

**Goal:** Minimize average squared distance between points and their nearest representatives:

$$cost(\mu_1,...,\mu_K) = \sum_{n=1}^{N} \min_j \|\mathbf{x}_n - \mu_j\|$$

The centers carve $\mathbb{R}^p$ up into $k$ convex regions: $\mu_j$’s region consists of points for which it is the closest center.
The objective function to be minimized is

\[ J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} \| x_n - \mu_k \|^2 \]  

(9.1)

Solving for \( r_{nk} \)

\[ r_{nk} = \begin{cases} 1 & \text{if } k = \arg \min_j \| x_n - \mu_j \|^2 \\ 0 & \text{otherwise} \end{cases} \]  

(9.2)

Differentiating for \( \mu_k \)

\[ 2 \sum_{n=1}^{N} r_{nk} (x_n - \mu_k) = 0 \]  

(9.3)

which we can easily solve for \( \mu_k \) to give

\[ \mu_k = \frac{\sum_n r_{nk} x_n}{\sum_n r_{nk}}. \]  

(9.4)
The two phases of re-assigning data points to clusters and re-computing the cluster means are repeated in turn until there is no further change in the assignments (or until some maximum number of iterations is exceeded). Because each phase reduces the value of the objective function which we can easily solve for, convergence of the algorithm is assured. How-ever, the procedure can be performed easily to give a closed form solution.

Consider first the determination of the cluster means. First we choose some initial values for the means \( \mu_k \), and updating them means are repeated in turn until there is no further change in the assignments (or \( \mu_k \) = \( \mu_k \)), this optimization can be performed easily to give a closed form solution.

The terms involving different \( \mu \) phases we minimize, otherwise we simply assign the data, known as \( k \), to the \( k \). Now consider the optimization of the objective function (9.1) is a linear function of updating \( r_{nk} \), such that each of the variables has zero mean and unit standard deviation. For this example, we have chosen \( \mu_k \) = \( \mu_k \) for whichever value of \( \mu_k \) - means algorithm were studied by MacQueen (1967).

The denominator in this expression is equal to the number of points assigned to cluster \( k \) \( n_k \), and so in this case means \( k \) - means Clustering.

\[ r_{nk} = \begin{cases} 1 & \text{if } k = \arg \min_j \| x_n - \mu_j \|^2 \\ 0 & \text{otherwise} \end{cases} \]

\[ \mu_k = \frac{\sum_n r_{nk} x_n}{\sum_n r_{nk}}. \]
Old Faithful, Kmeans from Murphy
How do we calculate the number of clusters?

‘Gap statistic’, Tibshirani et al. 2001

\[ W_k = \sum_{k=1}^{K} \sum_{n \in C_k} \| x_n - \mu_k \|^2 \]

- \( W_k \): from clustering data

- \( W_{kb}^* \): from clustering reference (uniform) distribution

\[
\text{Gap}(k) = \left( \frac{1}{B} \right) \sum_{b} \log(W_{kb}^*) - \log(W_k)
\]

Standard deviation of within cluster errors from reference data (red)

\[
sd_k = \left[ \left( \frac{1}{B} \right) \sum_{b} \left( \log(W_{kb}^*) - \bar{l} \right)^2 \right]^{1/2}
\]

\[
\bar{l} = \left( \frac{1}{B} \right) \sum_{b} \log(W_{kb}^*)
\]

Choose number of clusters \( K \) by

\[
\hat{k} = \text{smallest } k \text{ such that } \text{Gap}(k) \geq \text{Gap}(k + 1) - s_{k+1}
\]

\[
s_k = sd_k \sqrt{1 + 1/B}
\]

accounting for “simulation error”

https://datasciencelab.wordpress.com/tag/gap-statistic/
\( \hat{k} = \text{smallest } k \text{ such that } \text{Gap}(k) \geq \text{Gap}(k+1) - s_{k+1} \)
Application of K-means to data compression: Vector Quantization

Each pixel $x_i$ is represented by codebook of $K$ entries $\mu_k$

$\text{Encode}(x_i) = \arg\min_k\|x_i - \mu_k\|$

Original is 8 bits/pixel (bpp) with $D=1$ (block size=pixel size)

$K=2$, $\log_2 K$ bits = 1 bpp
$K=4$, 2 bpp

... can get more compression by increasing $D$

e.g. $D=4$, $K=2$: $1/4 = 0.25$ bpp (~3% of original)
Mixtures of Gaussians (1)

Old Faithful geyser:
The time between eruptions has a bimodal distribution, with the mean interval being either 65 or 91 minutes, and is dependent on the length of the prior eruption. Within a margin of error of ±10 minutes, Old Faithful will erupt either 65 minutes after an eruption lasting less than $2\frac{1}{2}$ minutes, or 91 minutes after an eruption lasting more than $2\frac{1}{2}$ minutes.
Mixtures of Gaussians (2)

Combine simple models into a complex model:

\[
p(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x|\mu_k, \Sigma_k)
\]

- \(\forall k : \pi_k \geq 0\)
- \(\sum_{k=1}^{K} \pi_k = 1\)

\(K=3\)
Mixtures of Gaussians (3)
Mixture of Gaussians

- Mixtures of Gaussians

\[
p(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x|\mu_k, \Sigma_k).
\]

- Expressed with latent variable \( z \)

\[
p(x) = \sum_z p(z)p(x|z) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x|\mu_k, \Sigma_k)
\]

\[
p(x|z_k = 1) = \mathcal{N}(x; \mu_k, \Sigma_k)
\]

\[
p(z_k = 1) = \sum_z p(z)
\]

\[
p(x|z) = \sum_z p(z|x)
\]

\[
p(z) = \sum p(z)
\]

\[
p(x, z) = \sum p(x, z)
\]

\[
\forall k : \pi_k \geq 0 \quad \sum_{k=1}^{K} \pi_k = 1
\]
Want to estimate the latent variables for data $X$

- Probability of data given latent representation

$$p(X|\pi, \mu, \Sigma) =$$

- Log likelihood

$$\ln p(X|\pi, \mu, \Sigma) =$$

$$\ln p(X|\pi, \mu, \Sigma) = \sum_{n=1}^{N} \ln \left\{ \sum_{k=1}^{K} \pi_k \mathcal{N}(x_n|\mu_k, \Sigma_k) \right\}.$$
Can’t we just solve for the latent variables by maximizing log likelihood?

- Log likelihood
  \[
  \ln p(X|\pi, \mu, \Sigma) = \sum_{n=1}^{N} \ln \left( \sum_{k=1}^{K} \pi_k N(x_n|\mu_k, \Sigma_k) \right).
  \]

- Take derivative w.r.t. \( \mu_k \):
Can’t we just solve for the latent variables by maximizing log likelihood?

- Log likelihood

\[
\ln p(X|\pi, \mu, \Sigma) = \sum_{n=1}^{N} \ln \left\{ \sum_{k=1}^{K} \pi_k \mathcal{N}(x_n|\mu_k, \Sigma_k) \right\}.
\]

- Take derivative w.r.t. \(\mu_k\):

\[
0 = -\sum_{n=1}^{N} \frac{\pi_k \mathcal{N}(x_n|\mu_k, \Sigma_k)}{\sum_{j} \pi_j \mathcal{N}(x_n|\mu_j, \Sigma_j)} \Sigma_k (x_n - \mu_k)
\]

\[
\gamma(z_{nk})
\]

“responsibility”, from Bayes’s rule:

\[
\gamma(z_k) \equiv p(z_k = 1|x) = \frac{p(z_k = 1)p(x|z_k = 1)}{\sum_{j=1}^{K} p(z_j = 1)p(x|z_j = 1)} = \frac{\pi_k \mathcal{N}(x|\mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(x|\mu_j, \Sigma_j)}.
\]
Solving for $\mu_k, \Sigma_k$

Take derivative w.r.t. $\mu_k$:
Solving for $\pi_k$

Use Lagrange multipliers with constraint

$$\sum_{k=1}^{K} \pi_k = 1$$
1. Initialize the means $\mu_k$, covariances $\Sigma_k$ and mixing coefficients $\pi_k$, and evaluate the initial value of the log likelihood.

2. **E step.** Evaluate the responsibilities using the current parameter values

   \[
   \gamma(z_{nk}) = \frac{\pi_k \mathcal{N}(x_n | \mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(x_n | \mu_j, \Sigma_j)}. \tag{9.23}
   \]

3. **M step.** Re-estimate the parameters using the current responsibilities

   \[
   \mu_k^{\text{new}} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) x_n \tag{9.24}
   \]

   \[
   \Sigma_k^{\text{new}} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) (x_n - \mu_k^{\text{new}})(x_n - \mu_k^{\text{new}})^T \tag{9.25}
   \]

   \[
   \pi_k^{\text{new}} = \frac{N_k}{N} \tag{9.26}
   \]

   where

   \[
   N_k = \sum_{n=1}^{N} \gamma(z_{nk}). \tag{9.27}
   \]

4. Evaluate the log likelihood

   \[
   \ln p(X|\mu, \Sigma, \pi) = \sum_{n=1}^{N} \ln \left\{ \sum_{k=1}^{K} \pi_k \mathcal{N}(x_n | \mu_k, \Sigma_k) \right\} \tag{9.28}
   \]

   and check for convergence of either the parameters or the log likelihood. If the convergence criterion is not satisfied return to step 2.
Important not to have singularities

Figure 9.7: Illustration of how singularities in the likelihood function arise with mixtures of Gaussians. This should be compared with the case of a single Gaussian shown in Figure 1.14 for which no singularities arise.

If we consider the limit $\sigma_j \to 0$, then we see that this term goes to infinity and so the log likelihood function will also go to infinity. Thus the maximization of the log likelihood function is not a well posed problem because such singularities will always be present and will occur whenever one of the Gaussian components ‘collapses’ onto a specific data point. Recall that this problem did not arise in the case of a single Gaussian distribution. To understand the difference, note that if a single Gaussian collapses onto a data point it will contribute multiplicative factors to the likelihood function arising from the other data points and these factors will go to zero exponentially fast, giving an overall likelihood that goes to zero rather than infinity. However, once we have (at least) two components in the mixture, one of the components can have a finite variance and therefore assign finite probability to all of the data points while the other component can shrink onto one specific data point and thereby contribute an ever increasing additive value to the log likelihood. This is illustrated in Figure 9.7. These singularities provide another example of the severe over-fitting that can occur in a maximum likelihood approach. We shall see that this difficulty does not occur if we adopt a Bayesian approach. For the moment, however, we simply note that in applying maximum likelihood to Gaussian mixture models we must take steps to avoid finding such pathological solutions and instead seek local maxima of the likelihood function that are well behaved. We can hope to avoid the singularities by using suitable heuristics, for instance by detecting when a Gaussian component is collapsing and resetting its mean to a randomly chosen value while also resetting its covariance to some large value, and then continuing with the optimization.
General EM

Given a joint distribution \( p(X, Z|\theta) \) over observed variables \( X \) and latent variables \( Z \), governed by parameters \( \theta \), the goal is to maximize the likelihood function \( p(X|\theta) \) with respect to \( \theta \).

1. Choose an initial setting for the parameters \( \theta^{old} \).

2. **E step** Evaluate \( p(Z|X, \theta^{old}) \).

3. **M step** Evaluate \( \theta^{new} \) given by

\[
\theta^{new} = \arg \max_{\theta} Q(\theta, \theta^{old})
\]

(9.32)

where

\[
Q(\theta, \theta^{old}) = \sum_{Z} p(Z|X, \theta^{old}) \ln p(X, Z|\theta).
\]

(9.33)

4. Check for convergence of either the log likelihood or the parameter values. If the convergence criterion is not satisfied, then let

\[
\theta^{old} \leftarrow \theta^{new}
\]

(9.34)

and return to step 2.
Gaussian Mixtures

(a) $L = 1$

(b) $L = 2$

(c) $L = 5$

(d) $L = 10$

(e) $L = 20$

(f) $L = 50$
Kmeans and EM (9.3.2)

\[ \Sigma_k = \epsilon I \]

\[ p(x|\mu_k, \Sigma_k) = \frac{1}{(2\pi\epsilon)^{1/2}} \exp \left\{ -\frac{1}{2\epsilon} \|x - \mu_k\|^2 \right\}. \]  \hspace{1cm} (9.41)

Whereby the responsibilities

\[ \gamma(z_{nk}) = \frac{\pi_k \exp \left\{ -\|x_n - \mu_k\|^2 / 2\epsilon \right\}}{\sum_j \pi_j \exp \left\{ -\|x_n - \mu_j\|^2 / 2\epsilon \right\}}. \]  \hspace{1cm} (9.42)

Becomes delta functions.

And the EM means approach the Kmeans

\[ \mu_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk})x_n \]  \hspace{1cm} (9.17)
Sparse models and dictionaries

- Sparse modeling assumes each signal model can be reconstructed from a few vectors from a large set of vectors, called a dictionary $\mathbf{D}$.
- Adds auxiliary sparse model to measurement model

$$\mathbf{d} = \mathbf{A}\mathbf{m} + \mathbf{n}, \quad \mathbf{m} \approx \mathbf{D}\mathbf{x} \quad \text{and} \quad |\mathbf{x}| \ll Q$$

- Optimization changes from estimating $\mathbf{m}$ to estimating sparse coefficients $\mathbf{x}$.
Sparse models and dictionaries

- Sparse modeling assumes each signal model can be reconstructed from a few vectors from a large set of vectors, called a dictionary $\mathbf{D}$
- Adds auxiliary sparse model to measurement model

\[ \mathbf{d} = \mathbf{A} \mathbf{m} + \mathbf{n}, \quad \mathbf{m} \approx \mathbf{D} \mathbf{x} \text{ and } |\mathbf{x}| \ll Q \]

- Optimization changes from estimating $\mathbf{m}$ to estimating sparse coefficients $\mathbf{x}$

\[
\begin{align*}
\mathbf{m} &= \begin{array}{c}
\text{\(n \times 1\) measurements}
\end{array} \\
\mathbf{D} &= \begin{array}{c}
\text{\(n \times Q\) dictionary}
\end{array} \\
\mathbf{x} &= \begin{array}{c}
\text{\(Q \times 1\) sparse signal}
\end{array} \\
\epsilon &= \begin{array}{c}
\text{error}
\end{array}
\end{align*}
\]

- Sparse objective:

\[
\min_{\mathbf{x}} \| \mathbf{A} \mathbf{D} \mathbf{x} - \mathbf{d} \|_2 \text{ subject to } \| \mathbf{x} \|_0 \leq T
\]
Dictionary can be optimized with dictionary learning

- Sparse modeling assumes each signal model can be reconstructed from a few vectors from a large set of vectors, called a dictionary $D$
- Adds auxiliary sparse model to measurement model

\[ d = Am + n, \quad m \approx Dx \quad \text{and} \quad |x| \ll Q \]

- Optimization changes from estimating $m$ to estimating sparse coefficients $x$

- Sparse objective:

\[ \min_x \| ADx - d \|_2 \quad \text{subject to} \quad \| x \|_0 \leq T \]
Unsupervised learning used for discovering ‘hidden’ structure in data, **no labels**

- **Clustering in 2D**
  - e.g. K-means

- **Dictionary learning**
  - Image
  - Dictionary learned from image:
    - each "atom" is high-dimensional cluster
  - e.g. K-SVD

---

Elad 2006
Unsupervised learning and dictionary learning

Unsupervised learning used for discovering ‘hidden’ structure in data, **no labels**

**Clustering in 2D**

Data: image patches

Image

Dictionary learned from image: each "atom" is high-dimensional cluster

- e.g. K-means

- e.g. K-SVD
Dictionary learning and sparsity

- Dictionary learning obtains "optimal" sparse modeling dictionaries directly from data
- Dictionary learning was developed in neuroscience (a.k.a. sparse coding) to help understand mammalian visual cortex structure
- Assumes (1) **Redundancy in data**: image patches are repetitions of a smaller set of elemental shapes; and (2) **Sparsity**: each patch is represented with few atoms from dictionary

"Natural" images, patches shown in **magenta**

Learn dictionary $\mathbf{D}$ describing $\mathbf{Y} = [\mathbf{y}_1, ..., \mathbf{y}_I]$

- Each patch is signal $\mathbf{y}_i$
- Set of all patches $\mathbf{Y} = [\mathbf{y}_1, ..., \mathbf{y}_I]$

Sparse model for patch $\mathbf{y}_i$ composed of few atoms from $\mathbf{D}$

$$\hat{\mathbf{x}}_i = \arg\min_{\mathbf{x}_i} \| \mathbf{y}_i - \mathbf{D}\mathbf{x}_i \|_2 \quad \text{subject to} \quad \| \mathbf{x}_i \|_0 \leq T$$
Dictionary learning and sparsity

- Dictionary learning obtains "optimal" sparse modeling dictionaries directly from data.
- Dictionary learning was developed in neuroscience (a.k.a. sparse coding) to help understand mammalian visual cortex structure.
- Assumes (1) **Redundancy in data**: image patches are repetitions of a smaller set of elemental shapes; and (2) **Sparsity**: each patch is represented with few atoms from dictionary.

"Natural" images, patches shown in **magenta**

Learn dictionary $\mathbf{D}$ describing $\mathbf{Y} = [\mathbf{y}_1, \ldots, \mathbf{y}_I]$

- Each patch is signal $\mathbf{y}_i$
- Set of all patches $\mathbf{Y} = [\mathbf{y}_1, \ldots, \mathbf{y}_I]$

Sparse model for patch $\mathbf{y}_i$ composed of few atoms from $\mathbf{D}$

$$\mathbf{y} = \underbrace{\begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \cdots \end{bmatrix}}_{\text{Dictionary}}$$
Checkerboard dictionary example

\( y = R_i s = D x_i \)

\( D \in \mathbb{R}^{n \times Q} \)

\( R_i \in \{0, 1\}^{n \times N} \)

\( \hat{x}_i = \arg\min_{x_i} \| y_i - D x_i \|_2 \) subject to \( \| x_i \|_0 \leq T \)

\( y = \begin{bmatrix} x_1 + \ldots \end{bmatrix} \)
Backup
EM in general

\[ p(X|\theta) = \sum_{Z} p(X, Z|\theta). \]  \hspace{1cm} (9.69)

\[ \ln p(X|\theta) = \mathcal{L}(q, \theta) + \text{KL}(q||p) \]  \hspace{1cm} (9.70)

where we have defined

\[ \mathcal{L}(q, \theta) = \sum_{Z} q(Z) \ln \left\{ \frac{p(X, Z|\theta)}{q(Z)} \right\} \]  \hspace{1cm} (9.71)

\[ \text{KL}(q||p) = - \sum_{Z} q(Z) \ln \left\{ \frac{p(Z|X, \theta)}{q(Z)} \right\}. \]  \hspace{1cm} (9.72)

\[ \ln p(X, Z|\theta) = \ln p(Z|X, \theta) + \ln p(X|\theta) \]  \hspace{1cm} (9.73)

\[ \mathcal{L}(q, \theta) = \sum_{Z} p(Z|X, \theta^{old}) \ln p(X, Z|\theta) - \sum_{Z} p(Z|X, \theta^{old}) \ln p(Z|X, \theta^{old}) \]

\[ = Q(\theta, \theta^{old}) + \text{const} \]  \hspace{1cm} (9.74)