COMP 562 – Lecture 11

Examples of Unsupervised Learning

- Dimensionality reduction -- lossy compression
- Clustering -- assigning each point to a representative cluster
  - Note: in classification groups are known, in clustering they are learned
- Deconvolution -- splitting mixed signals such as instruments or speakers in sound signal

Applications

- Data summarization/compression
- Denoising, outlier detection
- Feature construction

Clustering

We will first look at a very simple -- and popular -- clustering algorithm called **K-means**

- Randomly choose K cluster center locations (means or centroids)
- Loop until convergence
  1. Assignment of samples to the closest of the K centroids
  2. Re-estimate the cluster centroids or means based on the data assigned to each cluster

K-means Observations

1. Initialization is random -- means of the clusters are slightly preturbed versions of data mean
2. Clusters are assumed to be spherical
3. Must manually choose K
4. Clusters can have zero members -- make sure there is no division with zero

```
mus = numpy.dot(xs,ph.transpose())/(1e-5 + numpy.sum(ph,axis=1))
```

1. There are local minima
   - Non-trivial: poor initialization, too small or too large K
2. Multiple restarts -- from different initializations -- can lead to better solutions

How do we Come up with an Algorithm such as K-means?

We will:

1. Write out a generative model for the data
2. Write out log-likelihood
3. Optimize log-likelihood

The twist compared to our previous model learning is in the fact that not all the variables are observed

Labels, indicating cluster membership, are **hidden** from us

Mixture Models -- Generative Story

Each sample $\mathbf{x}$ is generated by:

1. Selecting a cluster, according to some distribution $\pi = (\pi_1, \ldots, \pi_K)$
$$p(h) = \pi_h$$
2. Using parameters of cluster h generate the sample $\mathbf{x}$
$$p(\mathbf{x} \mid h, \theta) = p(\mathbf{x} \mid \theta_h)$$

For example in **K-means**, you can think of θ_h as the mean of the cluster h

$$p(\mathbf{x} \mid \theta_h) = \prod_{j=1}^{p} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{ -\frac{1}{2\sigma^2} (x_j - \theta_{h,j})^2 \right\}$$

Mixture Models -- Generative Story

Note that generative model talks about how data might have been generated

We want to fit a model under which that data is most probable

To do so, we must write out log-likelihood
$$\mathcal{LL}(\Theta) = \sum_i \log p(\mathbf{x}_i \mid \Theta)$$

and maximize it with respect to parameters Θ

Marginal Log-Likelihood

For a single sample $\mathbf{x}$, we know how to compute $p(h)$ and $p(\mathbf{x} \mid h, \Theta)$ so we can obtain joint

$$p(\mathbf{x}, h \mid \Theta) = p(h)p(\mathbf{x} \mid h, \Theta)$$

This is probability of the full configuration $\mathbf{x}$ **and** cluster membership h

Our data does not contain information about cluster membership, just vectors $\mathbf{x}$

How do we compute $p(\mathbf{x} \mid \Theta)$?

Marginal Log-likelihood

We can use the fact that

$$p(\mathbf{x} \mid \Theta) = \sum_h p(\mathbf{x}, h \mid \Theta) = \sum_h p(h)p(\mathbf{x} \mid h, \Theta).$$

What is the interpretation of this sum?

Marginal Log-likelihood

Now, we can express log-likelihood in terms of probabilities in our model

$$\mathcal{LL}(\Theta) = \sum_i \log p(\mathbf{x}_i \mid \Theta) = \sum_{i=1}^{N} \log \sum_{h_i} p(\mathbf{x}_i, h_i \mid \Theta).$$

Observations

1. For each sample $\mathbf{x}_i$, we have corresponding cluster membership variable h_i
2. There is a sum under the log so we cannot push the log to probability terms

Dealing with Difficult Objectives

Typically, when we run into an objective that is difficult to work with (for example log of sums above), we seek a closely related objective that is easier

Hence, we will not directly optimize the log-likelihood instead we will introduce a lower-bound on the log-likelihood which we can show is tight at the optimum

To accomplish this we a bit of math that you may not be familar with

1. Convex and concave functions
2. Jensen's inequality

Concave and Convex Functions

A function $f(x)$ is concave if

$$f(\lambda x + (1-\lambda)y) \geq \lambda f(x) + (1-\lambda)f(y)$$

for any $\lambda \in [0, 1]$

Examples of concave functions are log, exp, square, etc.

Similarly, a function $f(x)$ is convex if

$$f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda)f(y)$$

for any $\lambda \in [0, 1]$

The negative of a concave function is convex

Jensen's Inequality

Jensen's inequality

$$f(\mathbb{E}[H]) \geq \mathbb{E}[f(H)]$$

where

$$\mathbb{E}[H] = \sum_{h} p(H = h)h.$$

and f is concave

Bounding log-Likelihood

Starting with log-likelihood

$$\mathcal{LL}(\Theta) = \sum_i \log p(\mathbf{x}_i \mid \Theta) = \sum_{i=1}^N \log \sum_{h_i} p(\mathbf{x}_i, h_i \mid \Theta).$$

we introduce distributions $q_i(h_i)$ and multiply and divide by them

$$\begin{aligned}
\mathcal{LL}(\Theta) = \sum_i \log p(\mathbf{x}_i \mid \Theta) &= \sum_{i=1}^N \log \sum_{h_i} \frac{q_i(h_i)}{q_i(h_i)} p(\mathbf{x}_i, h_i \mid \Theta) \\
&= \sum_{i=1}^N \log \sum_{h_i} q_t(h_i) \frac{p(\mathbf{x}_i, h_i \mid \Theta)}{q_i(h_i)} \\
&= \sum_{i=1}^N \log \mathbb{E}_{q_i} \left[\frac{p(\mathbf{x}_i, h_i \mid \Theta)}{q_i(h_i)} \right] \\
&\geq \sum_{i=1}^N \mathbb{E}_{q_i} \left[\log \frac{p(\mathbf{x}_i, h_i \mid \Theta)}{q_i(h_i)} \right] \\
&= \sum_{i=1}^N \sum_{h_i} q_i(h_i) \log \frac{p(\mathbf{x}_i, h_i \mid \Theta)}{q_i(h_i)}
\end{aligned}$$

Bounding log-Likelihood

Starting with log-likelihood

$$\mathcal{LL}(\Theta) = \sum_i \log p(\mathbf{x}_i \mid \Theta) \geq \sum_{i=1}^N \sum_{h_i} q_i(h_i) \log \frac{p(\mathbf{x}_i, h_i \mid \Theta)}{q_i(h_i)} = \mathcal{B}(\Theta, q)$$

Natural questions that arise:

1. Where do we get q_is?
2. Does optimizing bound result in the same Θ?

$$\operatorname*{argmax}_\Theta \mathcal{LL}(\Theta) \stackrel{?}{=} \operatorname*{argmax}_\Theta \mathcal{B}(\Theta, q)$$

Bounding log-Likelihood

Natural questions that arise:

- Where do we get q_is?
  - A: We use posterior probabilities $p(h_i \mid \mathbf{x}_i, \Theta)$
- $\text{argmax}_\Theta \, \mathcal{LL}(\Theta) \stackrel{?}{=} \text{argmax}_\Theta \, \mathcal{B}(\Theta, q)$?
  - A: Yes, if we use exact posterior probabilities in place of q_is the two objectives coincide and optimal Θs are the same

Expectation-Maximization (EM) Algorithm

Hence we can maximize the bound $\mathcal{B}(\Theta)$ by iterating

1. (E-step) Computing the optimal
$$q_i^{\text{new}} = \underset{q_i}{\text{argmax}} \, \mathcal{B}(\Theta^{\text{old}}, q)$$
2. (M-step) Updating Θ given current $q_i(h_i)$
$$\Theta^{\text{new}} = \underset{\Theta}{\text{argmax}} \, \mathcal{B}(\Theta, q^{\text{new}})$$

Recall for a moment the K-means algorithm. It alternated analogous two steps:

1. Assigning each sample to a cluster
2. Cluster center computation based on assignments

EM Algorithm for Mixture of Gaussians with Spherical Covariance

The model

$$
\begin{aligned}
p(h \mid \alpha) &= \alpha_h \\
p(\mathbf{x} \mid h, \mu) &= (2\pi)^{-\frac{d}{2}} \exp\left\{ -\frac{1}{2}(\mathbf{x} - \mu_{h_t})^T(\mathbf{x} - \mu_{h_t}) \right\}
\end{aligned}
$$

is a variant of **Mixture of Gaussians**

α_c is an a-priori probability that a sample comes from class c -- also called **mixing proportion**

The bound

$$
\begin{aligned}
\mathcal{B}(\Theta, q) &= \sum_{i=1}^{N} \sum_{h_i} q_i(h_i) \log \frac{p(\mathbf{x}_i, h_i \mid \Theta)}{q_i(h_i)} \\
&= \sum_{i=1}^{T} \sum_{h_i} q_i(h_i) \left[\log \alpha_{h_i} - \frac{d}{2}\log(2\pi) - \frac{1}{2}(\mathbf{x} - \mu_{h_i})^T(\mathbf{x} - \mu_{h_i}) \right] \\
&- \sum_{i=1}^{T} \sum_{h_i} q_i(h_i) \log q_i(h_i)
\end{aligned}
$$

In this case $\Theta = (\alpha_1, \ldots, \alpha_K, \mu_1, \ldots, \mu_K)$

E-step

The E-step

$$
\begin{aligned}
q_i(h_i = k) = p(h_i = k \mid \mathbf{x}_i, \mu) &= \frac{p(\mathbf{x}_i, h_i = k \mid \mu)}{\underbrace{\sum_c p(\mathbf{x}_i, h_i = c \mid \mu)}_{\text{same for all values of } k}} \\
&\propto p(\mathbf{x}_i, h_i = k \mid \mu) \\
&= \alpha_{h_i} (2\pi)^{-\frac{d}{2}} \exp\left\{ -\frac{1}{2}(\mathbf{x} - \mu_h)^T(\mathbf{x} - \mu_h) \right\}
\end{aligned}
$$

E-step: Working in Log-Domain

If the data vectors are long, the computation of joint probabilities can yield very tiny probabilities

Rather than working with probabilities we work with log-probabilities. Hence we store

$$\log q_i(h_i = k) = \log p(\mathbf{x}_i, h_i = k \mid \mu) - \log \sum_c p(\mathbf{x}_i, h_i = c \mid \mu)$$

If all the probabilities are stored in log-domain, then computation of their sum requires exponentation

$$\log \sum_c p(\mathbf{x}_i, h_i = c \mid \mu) = \log \sum_c \exp \underbrace{\log p(\mathbf{x}_i, h_i = c \mid \mu)}_{\text{stored log-probability}}$$

M-step

In M-step we optimize Θ given qs

$$\Theta^{\text{new}} = \underset{\Theta}{\operatorname{argmax}}\, \mathcal{B}(\Theta, q^{\text{new}})$$

In general, we can take derivatives, equate them to zero, and solve:

$$\nabla_\Theta \mathcal{B}(\Theta, q^{\text{new}}) = 0$$

We can show that in our case, the M-step updates are:

$$\mu_c^* = \frac{\sum_i q_i(c)\mathbf{x}_i}{\sum_i q_i(c)}$$

$$\alpha_c^* = \frac{\sum_i q_i(h_i = c)}{N}$$

COMP 562 – Lecture 12

EM Algorithm for Mixture of Gaussians without Covariance

The model

$$p(h \mid \alpha) = \alpha_h$$
$$p(\mathbf{x} \mid h, \mu) = (2\pi)^{-\frac{d}{2}} \exp\left\{ -\frac{1}{2}(\mathbf{x} - \mu_{h_t})^T(\mathbf{x} - \mu_{h_t}) \right\}$$

is a variant of **Mixture of Gaussians**

α_c is an a-priori probability that a sample comes from class c -- also called **mixing proportion**

The bound

$$\begin{aligned}
\mathcal{B}(\Theta, q) &= \sum_{i=1}^{N} \sum_{h_i} q_i(h_i) \log \frac{p(\mathbf{x}_i, h_i \mid \Theta)}{q_i(h_i)} \\
&= \sum_{i=1}^{T} \sum_{h_i} q_i(h_i) \left[\log \alpha_{h_i} - \frac{d}{2}\log(2\pi) - \frac{1}{2}(\mathbf{x} - \mu_{h_i})^T(\mathbf{x} - \mu_{h_i}) \right] \\
&- \sum_{i=1}^{T} \sum_{h_i} q_i(h_i) \log q_i(h_i)
\end{aligned}$$

In this case $\Theta = (\alpha_1, \ldots, \alpha_K, \mu_1, \ldots, \mu_K)$

Mixture of Gaussians without Covariance -- E-step

In E-step we optimize qs given Θ

$$q_i^{\text{new}} = \underset{q_i}{\operatorname{argmax}}\, \mathcal{B}(\Theta^{\text{old}}, q)$$

In general, we can take derivatives, equate them to zero, and solve:

$$\nabla_{q_i} \mathcal{B}(\Theta^{\text{old}}, q) = 0$$

We can show that in our case, the E-step updates are:

$$\begin{aligned} q_i(h_i = k) = p(h_i = k \mid \mathbf{x}_i, \mu) &= \frac{p(\mathbf{x}_i, h_i = k \mid \mu)}{\underbrace{\sum_c p(\mathbf{x}_i, h_i = c \mid \mu)}_{\text{same for all values of } k}} \\ &\propto p(\mathbf{x}_i, h_i = k \mid \mu) = p(h_i = k \mid \alpha) p(\mathbf{x} \mid h_i = k, \mu) \\ &= \alpha_{h_i} (2\pi)^{-\frac{d}{2}} \exp\left\{ -\frac{1}{2} (\mathbf{x} - \mu_h)^T (\mathbf{x} - \mu_h) \right\} \end{aligned}$$

Mixture of Gaussians without Covariance -- M-step

In M-step we optimize Θ given qs

$$\Theta^{\text{new}} = \underset{\Theta}{\operatorname{argmax}}\, \mathcal{B}(\Theta, q^{\text{new}})$$

In general, we can take derivatives, equate them to zero, and solve:

$$\nabla_{\Theta} \mathcal{B}(\Theta, q^{\text{new}}) = 0$$

We can show that in our case, the M-step updates are:

$$\begin{aligned} \mu_k^* &= \frac{\sum_i q_i(h_i = k) \mathbf{x}_i}{\sum_i q_i(h_i = k)} \\ \alpha_k^* &= \frac{\sum_i q_i(h_i = k)}{N} \end{aligned}$$

EM Algorithm for Mixture of Gaussians with Covariance

The model

$$
\begin{aligned}
p(h \mid \alpha) &= \alpha_h \\
p(\mathbf{x} \mid h, \mu) &= (2\pi)^{-\frac{d}{2}} |\Sigma_h|^{-\frac{1}{2}} \exp\left\{ -\frac{1}{2} (\mathbf{x} - \mu_{h_i})^T \Sigma_h^{-1} (\mathbf{x} - \mu_{h_i}) \right\}
\end{aligned}
$$

This is also a variant of **Mixture of Gaussians** Note that we introduced a covariance matrix per cluster

- Hidden variables: h_i -- cluster membership for sample i
- Parameters: $\Theta = (\underbrace{\alpha_1, \ldots, \alpha_K}_{\text{proportions}}, \underbrace{\mu_1, \ldots, \mu_K}_{\text{means}}, \underbrace{\Sigma_1, \ldots, \Sigma_K}_{\text{covariances}})$

EM Algorithm for Mixture of Gaussians with Covariance

We plug-in probabilities $p(\mathbf{x}_i \mid h_i, \Theta)$ and $p(h_i \mid \alpha)$ in the bound

$$
\begin{aligned}
\mathcal{B}(\Theta, q) \qquad &= \sum_{i=1}^{N} \sum_{h_i} q_i(h_i) \log \frac{p(\mathbf{x}_i, h_i \mid \Theta)}{q_i(h_i)} \\
&= \sum_{i=1}^{N} \sum_{h_i} q_i(h_i) \Big[\log \alpha_{h_i} - \frac{d}{2} \log(2\pi) - \frac{1}{2} \log |\Sigma_{h_i}| \\
&\qquad - \frac{1}{2} (\mathbf{x}_i - \mu_{h_i})^T \Sigma_{h_i}^{-1} (\mathbf{x}_i - \mu_{h_i}) \Big] \\
&- \sum_{i=1}^{N} \sum_{h_i} q_i(h_i) \log q_i(h_i)
\end{aligned}
$$

Mixture of Gaussians with Covariance -- E-step

$$q_i(h_i = k) = p(h_i = k \mid \mathbf{x}_i, \mu) = \underbrace{\frac{p(\mathbf{x}_i, h_i = k \mid \mu)}{\sum_c p(\mathbf{x}_i, h_i = c \mid \mu)}}_{\text{same for all values of } k}$$

$$\propto p(\mathbf{x}_i, h_i = k \mid \mu) = p(h_i = k \mid \alpha) p(\mathbf{x} \mid h_i = k, \mu)$$

$$= \alpha_{h_i} (2\pi)^{-\frac{d}{2}} |\Sigma_{h_i}^{-1}| \exp\left\{ -\frac{1}{2} (\mathbf{x} - \mu_{h_i})^T \Sigma_{h_i}^{-1} (\mathbf{x} - \mu_{h_i}) \right\}$$

Implementation:

```
q = numpy.zeros((K,N))            # clusters x samples
q = logjointp(x,Theta)            # compute all joints at once
loglik = numpy.sum(logsumexp(q)) # compute loglikelihood
q = q - logsumexp(q)              # normalizing across clusters
```

Mixture of Gaussians with Covariance -- M-step

Updates for parameters of prior probability $p(h \mid \alpha)$

$$\alpha_k^* = \frac{\sum_i q_i(h_i = k)}{N}$$

Updates for means of clusters

$$\mu_k^* = \frac{\sum_i q_i(h_i = k) \mathbf{x}_i}{\sum_i q_i(h_i = k)}$$

Updates for covariances of clusters

$$\Sigma_k^* = \frac{\sum_i q_i(h_i = k)(\mathbf{x}_i - \mu_k^*)(\mathbf{x}_i - \mu_k^*)^T}{\sum_i q_i(h_i = k)}$$

Debugging EM Algorithm

1. Log-likelihood should always go up!
2. Synthetic data is your friend, if you generate data from your model you get samples and cluster membership
3. E-step computes cluster membership based on parameters. Use this!
   - Synthesize data from ground truth parameters
   - Start your EM from ground truth parameters, not random initialization
   - Does your E step associate samples with correct clusters?
   - Select one sample and look at its posterior probability for the cluster it came from
4. M-step updates parameters based on cluster membership. Use this!
   - Using synthetic data, set `q` to be one-hot according to ground truth
   - Start your M-step with this `q`
   - If you don't get parameters back that are close to the ground truth
   - To isolate a broken update, let M-step update just one parameter (for example mus)
5. Starting your EM with ground truth parameters should not budge too much

Between these tricks you should be able to isolate source of your problem