Drs. Sha & Liu

\{feisha,yanliu.cs\}@usc.edu

November 15, 2014
Outline

1. Administrative matters
   - Quiz 1 Grades

2. Review of last lecture

3. Clustering
Quiz 1 Grades finished and uploaded to Blackboard

Statistics

Distribution

![Histogram of Quiz 1 Grades]
Outline

1. Administrative matters
2. Review of last lecture
   - Neural networks
3. Clustering
Basic idea

**Learning nonlinear basis functions and classifiers**

- Hidden layers are nonlinear mappings from input features to new representation
- Output layers use the new representations for classification and regression

**Learning parameters**

- Stochastic gradient descent
- Large-scale computing
Outline

1. Administrative matters
2. Review of last lecture
3. Clustering
   - Gaussian mixture models
   - EM Algorithm
Clustering

Setup  Given $\mathcal{D} = \{x_n\}_{n=1}^N$ and $K$, we want to output

- $\{\mu_k\}_{k=1}^K$: prototypes of clusters
- $A(x_n) \in \{1, 2, \ldots, K\}$: the cluster membership, i.e., the cluster ID assigned to $x_n$

Example  Cluster data into two clusters.
K-means clustering

**Intuition** Data points assigned to cluster $k$ should be close to $\mu_k$, the prototype.
**K-means clustering**

**Intuition** Data points assigned to cluster $k$ should be close to $\mu_k$, the prototype.

**Distortion measure** (clustering objective function, cost function)

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} \| x_n - \mu_k \|_2^2$$

where $r_{nk} \in \{0, 1\}$ is an indicator variable

$$r_{nk} = 1 \quad \text{if and only if} \quad A(x_n) = k$$
Algorithm

Minimize distortion measure alternative optimization between \( \{ r_{nk} \} \) and \( \{ \mu_k \} \)

- **Step 0** Initialize \( \{ \mu_k \} \) to some values
Algorithm

**Minimize distortion measure** alternative optimization between \( \{r_{nk}\} \) and \( \{\mu_k\} \)

- **Step 0** Initialize \( \{\mu_k\} \) to some values
- **Step 1** Assume the current value of \( \{\mu_k\} \) fixed, minimize \( J \) over \( \{r_{nk}\} \), which leads to the following cluster assignment rule

\[
r_{nk} = \begin{cases} 
1 & \text{if } k = \arg\min_j \|x_n - \mu_j\|^2_2 \\
0 & \text{otherwise}
\end{cases}
\]
Algorithm

Minimize distortion measure alternative optimization between \( \{r_{nk}\} \) and \( \{\mu_k\} \)

- **Step 0** Initialize \( \{\mu_k\} \) to some values
- **Step 1** Assume the current value of \( \{\mu_k\} \) fixed, minimize \( J \) over \( \{r_{nk}\} \), which leads to the following cluster assignment rule
  \[
  r_{nk} = \begin{cases} 
  1 & \text{if } k = \arg \min_j \| x_n - \mu_j \|_2^2 \\
  0 & \text{otherwise}
  \end{cases}
  \]
- **Step 2** Assume the current value of \( \{r_{nk}\} \) fixed, minimize \( J \) over \( \{\mu_k\} \), which leads to the following rule to update the prototypes of the clusters
  \[
  \mu_k = \frac{\sum_n r_{nk} x_n}{\sum_n r_{nk}}
  \]
- **Step 3** Determine whether to stop or return to Step 1
The prototype $\mu_k$ is the means of data points assigned to the cluster $k$, hence the name K-means clustering.

The procedure terminates after a finite number of steps (in general, assuming that there is no tie in comparing distances in Step 1), as the procedure reduces $J$ in both Step 1 and Step 2. Since $J$ is lower bounded by 0, the procedure cannot be infinite.

There is no guarantee the procedure terminates at the global optimum of $J$ — in most cases, the algorithm stops at a local optimum, which depends on the initial values in Step 0.
Example of running K-means algorithm
Application: vector quantization

We can replace our data points with the prototypes $\mu_k$ from the clusters they are assigned to. This is called vector quantization. In other words, we have compressed the data points into i) a codebook of all the prototypes; ii) a list of indices to the codebook for the data points. This compression is obviously lossy as certain information will be lost if we use a very small $K$.

Clustering the pixels in the image and vector quantizing them. From left to right: Original image, quantized one with a large $K$, a medium $K$, and a small $K$. Details are missing due to the higher compression (smaller $K$).
Probabilistic interpretation of clustering?

We can impose a probabilistic interpretation of the intuition: data points stay close to the centers of their clusters. This is just a statement of how $p(x)$ looks like — we will see how to model this distribution.
We can impose a probabilistic interpretation of the intuition: data points stay close to the centers of their clusters. This is just a statement of how $p(x)$ looks like — we will see how to model this distribution.

The data points seem to form 3 clusters. However, we cannot model $p(x)$ with simple and known distributions. For example, the data do not obey Gaussian distributions as there are seemingly 3 regions where data concentrate.
Instead, we will model each region with a Gaussian distribution. This leads to the idea of Gaussian mixture models (GMMs) or mixture of Gaussians (MoGs).

The problem we are now facing is that i) we do not know which (color) region a data point comes from; ii) the parameters of Gaussian distributions in each region. We need to all of them from *unsupervised* data $\mathcal{D} = \{\mathbf{x}_n\}_{n=1}^N$. 
Gaussian mixture models: formal definition

A Gaussian mixture model has the following density function for $x$

$$p(x) = \sum_{k=1}^{K} \omega_k N(x | \mu_k, \Sigma_k)$$

where

- $K$: the number of Gaussians — they are called (mixture) components
- $\mu_k$ and $\Sigma_k$: mean and covariance matrix of the $k$-th component
- $\omega_k$: mixture weights – they represent how much each component contributes to the final distribution. It satisfies two properties:

$$\forall \ k, \ \omega_k > 0, \ \text{and} \ \sum_{k} \omega_k = 1$$

The properties ensure $p(x)$ is a properly normalized probability density function.
Consider the following joint distribution

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

where \( z \) is a discrete random variable taking values between 1 and \( K \). Denote

\[ \omega_k = p(z = k) \]

and furthermore, assume the conditional distributions are Gaussian distributions

\[ p(x | z = k) = N(x | \mu_k, \Sigma_k) \]

Then, the marginal distribution of \( x \) is

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

Namely, the Gaussian mixture models.
GMMs: example

The conditional distribution between $x$ and $z$ (representing color) are

$$p(x|z = 'red') = N(x|\mu_1, \Sigma_1)$$
$$p(x|z = 'blue') = N(x|\mu_2, \Sigma_2)$$
$$p(x|z = 'green') = N(x|\mu_3, \Sigma_3)$$
The conditional distribution between $x$ and $z$ (representing color) are

\[
\begin{align*}
  p(x \mid z = 'red') &= N(x \mid \mu_1, \Sigma_1) \\
  p(x \mid z = 'blue') &= N(x \mid \mu_2, \Sigma_2) \\
  p(x \mid z = 'green') &= N(x \mid \mu_3, \Sigma_3)
\end{align*}
\]

The marginal distribution is thus

\[
\begin{align*}
p(x) &= p('red')N(x \mid \mu_1, \Sigma_1) + p('blue')N(x \mid \mu_2, \Sigma_2) \\
    &+ p('green')N(x \mid \mu_3, \Sigma_3)
\end{align*}
\]
Parameter estimation for Gaussian mixture models

The parameters in GMMs are $\theta = \{\omega_k, \mu_k, \Sigma_k\}_{k=1}^K$. To estimate, consider the simple case first.

$\mathbf{z}$ is given If we assume $\mathbf{z}$ is observed for every $\mathbf{x}$, then our estimation problem is easier to solve. Particularly, our training data is augmented

$$
\mathcal{D}' = \{\mathbf{x}_n, \mathbf{z}_n\}_{n=1}^N
$$

Note that, for every $\mathbf{x}_n$, we have a $\mathbf{z}_n$ to denote the region/color where the specific $\mathbf{x}_n$ comes from. We call $\mathcal{D}'$ the complete data and $\mathcal{D}$ the incomplete data.
Parameter estimation for Gaussian mixture models

The parameters in GMMs are $\theta = \{\omega_k, \mu_k, \Sigma_k\}_{k=1}^K$. To estimate, consider the simple case first.

---

**$z$ is given** If we assume $z$ is observed for every $x$, then our estimation problem is easier to solve. Particularly, our training data is *augmented*

$$
\mathcal{D}' = \{x_n, z_n\}_{n=1}^N
$$

Note that, for every $x_n$, we have a $z_n$ to denote the region/color where the specific $x_n$ comes from. We call $\mathcal{D}'$ the *complete* data and $\mathcal{D}$ the *incomplete* data.

Given $\mathcal{D}'$, the maximum likelihood estimation of the $\theta$ is given by

$$
\theta = \arg \max_{\theta} \log \mathcal{D}' = \sum_n \log p(x_n, z_n)
$$
Parameter estimation for GMMs: complete data

The likelihood — which we will refer to *complete* likelihood is decomposable

$$\sum_n \log p(x_n, z_n) = \sum_n \log p(z_n)p(x_n|z_n) = \sum_k \sum_{n:z_n=k} \log p(z_n)p(x_n|z_n)$$

where we have grouped data by its values $z_n$. Let us introduce a binary variable $\gamma_{nk} \in \{0, 1\}$ to indicate whether $z_n = k$. We can rewrite our decomposition as
Parameter estimation for GMMs: complete data

The likelihood — which we will refer to complete likelihood is decomposable

$$\sum_n \log p(x_n, z_n) = \sum_n \log p(z_n) p(x_n | z_n) = \sum_k \sum_{n: z_n = k} \log p(z_n) p(x_n | z_n)$$

where we have grouped data by its values $z_n$. Let us introduce a binary variable $\gamma_{nk} \in \{0, 1\}$ to indicate whether $z_n = k$. We can rewrite our decomposition as

$$\sum_n \log p(x_n, z_n) = \sum_k \sum_n \gamma_{nk} \log p(z = k) p(x_n | z = k)$$

Note that we have used a “dummy” variable $z$ to denote all the possible values $x_n$’s true $z_n$ can take — but only one of the possible values is given in $D'$. 
Parameter estimation for GMMs: solution for complete data

Substituting our assumption about the conditional distributions, we have

\[
\sum_n \log p(x_n, z_n) = \sum_k \sum_n \gamma_{nk} \left[ \log \omega_k + \log N(x_n | \mu_k, \Sigma_k) \right]
\]

Regrouping, we have

\[
\sum_n \log p(x_n, z_n) = \sum_k \sum_n \gamma_{nk} \log \omega_k + \sum_k \left\{ \sum_n \gamma_{nk} \log N(x_n | \mu_k, \Sigma_k) \right\}
\]
Parameter estimation for GMMs: solution for complete data

Substituting our assumption about the conditional distributions, we have

$$\sum_n \log p(x_n, z_n) = \sum_k \sum_n \gamma_{nk} \left[ \log \omega_k + \log N(x_n | \mu_k, \Sigma_k) \right]$$

Regrouping, we have

$$\sum_n \log p(x_n, z_n) = \sum_k \sum_n \gamma_{nk} \log \omega_k + \sum_k \left\{ \sum_n \gamma_{nk} \log N(x_n | \mu_k, \Sigma_k) \right\}$$

Note that, the term inside the braces depends on $k$-th component’s parameters. It is now easy to show that (left as a homework exercise), the maximum likelihood estimation of the parameters are

$$\omega_k = \frac{\sum_n \gamma_{nk}}{\sum_k \sum_n \gamma_{nk}} , \quad \mu_k = \frac{1}{\sum_n \gamma_{nk} \sum_n \gamma_{nk} x_n} \sum_n \gamma_{nk} x_n$$

$$\Sigma_k = \frac{1}{\sum_n \gamma_{nk} \sum_n \gamma_{nk}} \sum_n \gamma_{nk} (x_n - \mu_k)(x_n - \mu_k)^T$$
Intuition

Since $\gamma_{nk}$ is binary, the previous solution is nothing but:

- For $\omega_k$: count the number of data points whose $z_n$ is $k$ and divide by the total number of data points (note that $\sum_k \sum_n \gamma_{nk} = N$)
- For $\mu_k$: get all the data points whose $z_n$ is $k$, compute their mean
- For $\Sigma_k$: get all the data points whose $z_n$ is $k$, compute their covariance matrix

This intuition is going to help us to develop an algorithm for estimating $\theta$ when we do not know $z_n$. 
Parameter estimation for GMMs: incomplete data

When $z_n$ is not given, we can guess which region/color $x_n$ comes from by computing the posterior probability

$$p(z_n = k | x_n) = \frac{p(x_n | z_n = k) p(z_n = k)}{p(x_n)} = \frac{p(x_n | z_n = k) p(z_n = k)}{\sum_{k' = 1}^{K} p(x_n | z_n = k') p(z_n = k')}$$

Note that, to compute the posterior probability, we need to know the parameters $\theta$. Let us for a second, we pretend we know the value of the parameters thus we can compute the posterior probability.

How is that going to help us?
Estimation with soft $\gamma_{nk}$

We are going to pretend $p(z_n = k | x_n)$ as $\gamma_{nk}$ which should be binary – but now is regarded as “soft” assigning $x_n$ to $k$-th component. With that in mind, we have

$$
\gamma_{nk} = p(z_n = k | x_n)
$$

$$
\omega_k = \frac{\sum_n \gamma_{nk}}{\sum_k \sum_n \gamma_{nk}}
$$

$$
\mu_k = \frac{1}{\sum_n \gamma_{nk}} \sum_n \gamma_{nk} x_n
$$

$$
\Sigma_k = \frac{1}{\sum_n \gamma_{nk}} \sum_n \gamma_{nk} (x_n - \mu_k)(x_n - \mu_k)^T
$$

In other words, every data point $x_n$ is assigned to a component fractionally according to $p(z_n = k | x_n)$ — sometimes, this quantity is also called “responsibility”.

Since we do not know θ to begin with, we cannot compute the soft γ_{nk}. However, we can invoke an iterative procedure and alternate between estimating γ_{nk} and using the estimated γ_{nk} to compute the parameters

- **Step 0**: guess θ with initial values
- **Step 1**: compute γ_{nk} using the current θ
- **Step 2**: update θ using the just computed γ_{nk}
- **Step 3**: go back to Step 1

Questions: i) is this procedure correct, for example, optimizing a sensible criteria? ii) practically, will this procedure ever stop instead of iterating forever?

The answer lies in the EM algorithm — a powerful procedure for model estimation with unknown data.
EM algorithm: motivation and setup

As a general procedure, EM is used to estimate parameters for probabilistic models with hidden/latent variables. Suppose the model is given by a joint distribution

$$p(x|\theta) = \sum_z p(x, z|\theta)$$

where $x$ is the observed random variable and $z$ is hidden.

We are given data containing only the observed variable $D = \{x_n\}$ where the corresponding hidden variable values $z$ is not included. Our goal is to obtain the maximum likelihood estimate of $\theta$. Namely, we choose

$$\theta = \arg \max \log D = \arg \max \sum_n \log p(x_n|\theta)$$

$$= \arg \max \sum_n \log \sum_z p(x_n, z_n|\theta)$$

The objective function $\ell(\theta)$ is called incomplete log-likelihood.
Expected (complete) log-likelihood

The difficulty with incomplete log-likelihood is that it needs to sum over all possible values that \( z_n \) can take, then take a logarithm. This log-sum format makes computation intractable. Instead, the EM algorithm uses a clever trick to change this into sum-log form.

To this end, we define the following

\[
Q_q(\theta) = \sum_n \mathbb{E}_{z_n \sim q(z_n)} \log p(x_n, z_n | \theta) \\
= \sum_n \sum z_n q(z_n) \log p(x_n, z_n | \theta)
\]

which is called **expected (complete) log-likelihood** (with respect to \( q(z) \). \( q(z) \) is a distribution over \( z \). Note that \( Q_q(\theta) \) takes the form of sum-log, which turns out to be tractable.
Examples

Consider the previous model where \( x \) could be from 3 regions. We can choose \( q(z) \) any valid distribution. This will lead to different \( Q_q(\theta) \). Note that \( z \) here represents different colors.

- \( q(z = k) = 1/3 \) for any of 3 colors. This gives rise to
  \[
  Q_q(\theta) = \sum_n \frac{1}{3} [\log p(x_n, 'red'|\theta) + \log p(x_n, 'blue'|\theta) + \log p(x_n, 'green'|\theta)]
  \]

- \( q(z = k) = 1/2 \) for 'red' and 'blue', 0 for 'green'. This gives rise to
  \[
  Q_q(\theta) = \sum_n \frac{1}{2} [\log p(x_n, 'red'|\theta) + \log p(x_n, 'blue'|\theta)]
  \]
Which $q(z)$ to choose?

We will choose a special $q(z) = p(z|x; \theta)$, i.e., the posterior probability of $z$. We define

$$Q(\theta) = Q_{z \sim p(z|x; \theta)}(\theta)$$

and we will show

$$\ell(\theta) = Q(\theta) + \sum_n H[p(z|x_n; \theta)]$$

where $H[p]$ is the entropy of the probabilistic distribution $p$:

$$H[p(x)] = - \int p(x) \log p(x) dx$$