L14: Mixture models and EM

Supervised vs. unsupervised learning

Mixture models

Expectation maximization (EM)

Supervised vs. unsupervised learning

The pattern recognition methods covered in class up to this point have focused on the issue of classification

- A pattern consisted of a pair of variables $\{x, \omega\}$ where
  - x was a collection of observations or features (feature vector)
  - ω was the concept behind the observation (label)
- Such pattern recognition problems are called supervised (training with a teacher) since the system is given BOTH the feature vector and the correct answer

In the next three lectures we investigate a number of methods that operate on unlabeled data

- Given a collection of feature vectors $X = \{x^{(1}, x^{(2} \ldots, x^{(N}\}$ without class labels ω_i, these methods attempt to build a model that captures the structure of the data
- These methods are called unsupervised (training without a teacher) since they are not provided the correct answer

Although unsupervised learning methods may appear to have limited capabilities, there are several reasons that make them extremely useful

- Labeling large data sets can be a costly procedure (i.e., ASR)
- Class labels may not be known beforehand (i.e., data mining)
- Large datasets can be compressed into a small set of prototypes (kNN)

The supervised and unsupervised paradigms comprise the vast majority of pattern recognition problems

- A third approach, known as reinforcement learning, uses a reward signal (real-valued or binary) to tell the learning system how well it is performing
- In reinforcement learning, the goal of the learning system (or agent) is to learn a mapping from states onto actions (an action policy) that maximizes the total reward

Approaches to unsupervised learning

Parametric (mixture models)

- These methods model the underlying class-conditional densities with a mixture of parametric densities, and the objective is to find the model parameters

$$p(x|\theta) = \sum_{i=1}^{C} p(x|\omega_i, \theta_i) P(\omega_i)$$

- These methods are closely related to parameter estimation (L6)
- Mixture models are the subject of this lecture

Non-parametric (clustering)

- No assumptions are made about the underlying densities, instead we seek a partition of the data into clusters
- These methods will be the subject of the next two lectures
  - L15 will focus on statistical clustering
  - L16 will deal with connectionist approaches

There are two reasons why we cover mixture models at this point

- The solution to the mixture problem (the EM algorithm) is also used for Hidden Markov Models, which will be introduced in just a few lectures
- A particular form of the mixture model problem leads to the most widely used clustering method: the k-means algorithm (a.k.a. vector quantization)

Mixture models

Consider the now familiar problem of modeling a pdf given a dataset $X = \{x^{(1}, x^{(2} \dots, x^{(N}\}$

- If the form of the underlying pdf was known (e.g. Gaussian), the problem could be solved using Maximum Likelihood (L6)
- If the form of the pdf was unknown, the problem had to be solved with non-parametric DE methods such as Parzen windows (L7-8)

We will now consider an alternative DE method: modeling the pdf with a mixture of parametric densities

- These methods are sometimes known as semi-parametric
  - Think of the individual components in the mixture as kernels, except for there is only a few of them, as opposed to one per data point as in L7
- In particular, we will focus on mixture models of Gaussian densities (surprised?)

$$p(x|\theta) = \sum_{c=1}^{C} p(x|\,\theta_c)P(\omega_c)$$

Mixture models can be posed in terms of the ML criterion

- Given a dataset $X = \{x^{(1}, x^{(2} \dots, x^{(N}\}$, find the parameters of the model that maximize the log likelihood of the data

$$\hat{\theta} = argmax\, p(X|\theta) = argmax \sum_{n=1}^{N} \log p\left(x^{(n}|\theta\right) = argmax \sum_{n=1}^{N} \log \sum_{c=1}^{C} p\left(x^{(n}|\, \theta_c\right) P(\omega_c)$$

  - where $\theta_c = \{\mu_c, \Sigma_c\}$ and $P(\omega_c)$ are the parameters and mixing coefficient of the c^{th} mixture component, respectively
  - The mixing coefficients may also be interpreted as priors

We could try to find the maximum of this function by differentiation

- For $\Sigma_i = \sigma_i I$, the solution becomes [Bishop, 1995]:

$$\frac{\partial}{\partial \mu_c}[\cdot] = 0 \qquad \Rightarrow \hat{\mu}_c = \frac{\sum_n P\left(\omega_c|x^{(n}\right) x^{(n}}{\sum_n P\left(\omega_c|x^{(n}\right)}$$

$$\frac{\partial}{\partial \sigma_c}[\cdot] = 0 \qquad \Rightarrow \hat{\sigma}_c^2 = \frac{1}{d} \frac{\sum_n P\left(\omega_c|x^{(n}\right) \left\|x^{(n} - \hat{\mu}_c\right\|^2}{\sum_n P\left(\omega_c|x^{(n}\right)}$$

$$\frac{\partial}{\partial P(\omega_c)}[\cdot] = 0 \qquad \Rightarrow \hat{P}(\omega_c) = \frac{1}{N} \sum_n P\left(\omega_c|x^{(n}\right)$$

Notice that the previous equations are not a closed form solution

- The model parameters μ_c, Σ_c, and $P(\omega_c)$ also appear on the RHS as a result of Bayes rule!
- Therefore, these expressions represent a highly non-linear coupled system of equations

However, these expressions suggest that we may be able to use a fixed-point algorithm to find the maxima

1) Begin with some "old" value of the model parameters
2) Evaluate the RHS of the equations to obtain "new" parameter values
3) Let these "new" values become the "old" ones and repeat the process

Surprisingly, an algorithm of this simple form can be found which is guaranteed to increase the log-likelihood with every iteration!

- This example represents a particular case of a more general procedure known as the Expectation-Maximization algorithm

The Expectation-Maximization (EM) algorithm

EM is a general method for finding the ML estimate of the parameters of a pdf when the data has missing values

- There are two main applications of the EM algorithm
  - When the data indeed has incomplete, missing or corrupted values as a result of a faulty observation process
  - When assuming the existence of missing or hidden parameters can simplify the likelihood function, which would otherwise lead to an analytically intractable optimization problem; this is the case we discuss in this lecture

Assume a dataset containing two types of features

- A set of features X whose value is known. We call these the incomplete data
- A set of features Z whose value is unknown. We call these the missing data

We now define a joint pdf $p(X, Z|\theta)$ called the complete-data likelihood

- This function is a random variable since the features Z are unknown
- You can think of $p(X, Z|\theta) = h_{X,\theta}(Z)$, for some function $h_{X,\theta}(\cdot)$, where X and θ are constant and Z is a random variable

As suggested by its name, the EM algorithm operates by performing two basic operations over and over

- An Expectation step
- A Maximization step

EXPECTATION

- Find the expected value of $\log p(X, Z|\theta)$ with respect to the unknown data Z, given the data X and the current parameter estimates $\theta^{(i-1}$

$$Q\left(\theta|\theta^{(i-1)}\right) = E_Z\left[\log p(X, Z|\theta)\,|X, \theta^{(i-1}\right]$$

  - where θ are the new parameters that we seek to optimize to increase Q
- Note that X and $\theta^{(i-1}$ are constants, θ is the variable that we wish to adjust, and Z is a random variable defined by $p(Z|X, \theta^{(i-1)})$
- Therefore $Q\left(\theta|\theta^{(i-1)}\right)$ is just a function of θ

MAXIMIZATION

- Find the argument θ that maximizes the expected value $Q\left(\theta|\theta^{(i-1)}\right)$

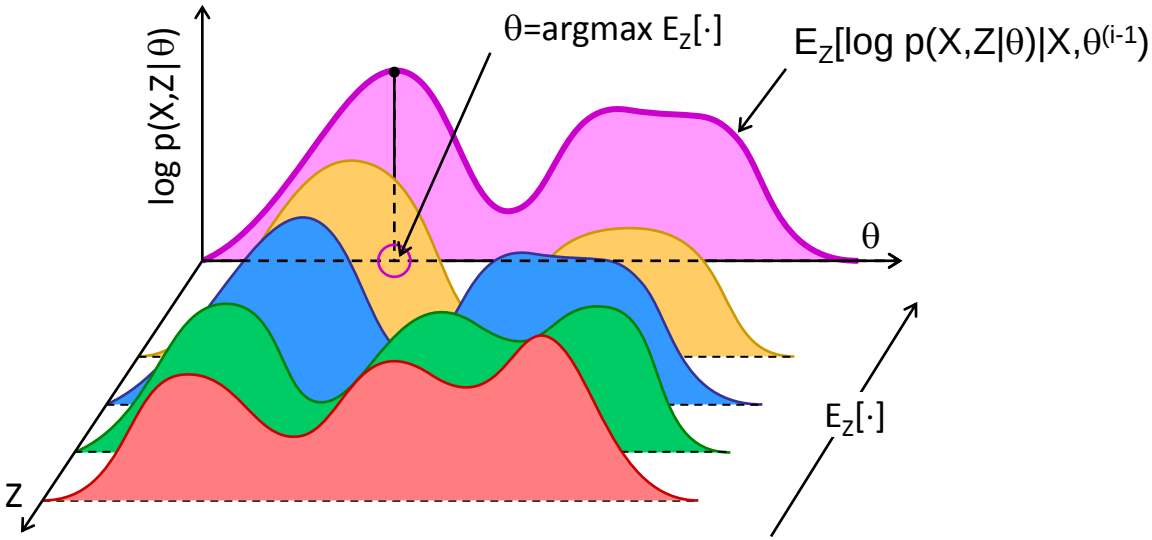
$$\theta^{(i} = \arg\max Q\left(\theta|\theta^{(i-1)}\right)$$

Convergence properties

- It can be shown that
  1) each iteration (E+M) is guaranteed to increase the log-likelihood and
  2) the EM algorithm is guaranteed to converge to a local maximum of the likelihood function

The two steps of the EM algorithm are illustrated below

- During the E step, the unknown features Z are integrated out assuming the current values of the parameters $\theta^{(i-1}$
- During the M step, the values of the parameters that maximize the expected value of the log likelihood are obtained



- **IN A NUTSHELL**: since Z are unknown, the best we can do is maximize the average log-likelihood across all possible values of Z

The EM algorithm and mixture models

Having formalized the EM algorithm, we are now ready to find the solution to the mixture model problem

- To keep things simple, we will assume a univariate mixture model where all the components have the same known standard deviation σ

Problem formulation

- As usual, we are given a dataset $X = \{x^{(1}, x^{(2} \dots, x^{(N}\}$, and we seek to estimate the model parameters $\theta = \{\mu_1, \mu_2 \dots \mu_C\}$
- The following process is assumed to generate each random variable $x^{(n}$
  - First, a Gaussian component is selected according to mixture coeffs $P(\omega_c)$
  - Then, $x^{(n}$ is generated according to the likelihood $p(x|\mu_c)$ of that particular component
- We will also use hidden variables $Z = \{z_1^{(n}, z_2^{(n} \dots z_c^{(n}\}$ to indicate which of the C Gaussian components generated data point $x^{(n}$

Solution

- The probability $p(x, z|\theta)$ for a specific example is

$$p\left(x^{(n}, z_1^{(n}, z_2^{(n} \dots z_c^{(n}|\theta\right) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2\sigma^2}\sum_{c=1}^{C} z_c^{(n}\left(x_c^{(n}-\mu_c\right)^2}$$

  - where only one of the $z_c^{(n}$ can have a value of 1,and all others are zero

– The log-likelihood of the entire dataset is then

$$\log p(X,Z|\theta) = \log \prod_{n=1}^{N} p\left(x^{(n}, z^{(n}|\theta\right) = \sum_{n=1}^{N} \left(\log \frac{1}{\sqrt{2\pi}\sigma} - \frac{1}{2\sigma^2} \sum_{c=1}^{C} z_c^{(n} \left(x^{(n} - \mu_c\right)^2\right)$$

– To obtain $Q\left(\theta|\theta^{(i-1}\right)$ we must then take the expectation over Z

$$\mathrm{E}_Z[\log p(X,Z|\theta)] = \sum_{n=1}^{N} \left(\log \frac{1}{\sqrt{2\pi}\sigma} - \frac{1}{2\sigma^2} \sum_{c=1}^{C} E\left[z_c^{(n}\right] \left(x^{(n} - \mu_c\right)^2 \right)$$

- where we have used the fact that $E[f(z)] = f(E[z])$ for $f(z)$ linear

– $E[z_c^{(n}]$ is simply the probability that $x^{(n}$ was generated by the c^{th} Gaussian component given current model parameters $\theta^{(i-1}$

$$E\left[z_c^{(n}\right] = \frac{p\left(x=x^{(n}|\mu=\mu_c^{(i-1}\right)}{\sum_{q=1}^{C} p\left(x=x^{(n}|\mu=\mu_q^{(i-1}\right)} = \frac{\exp\left(-\frac{1}{2\sigma^2}\left(x^{(n}-\mu_c^{(i-1}\right)^2\right)}{\sum_{q=1}^{C} \exp\left(-\frac{1}{2\sigma^2}\left(x^{(n}-\mu_q^{(i-1}\right)^2\right)} \quad (1)$$

– These two expressions define the Q function

– The second step (Maximization) consists of finding the values $\{\mu_1, \mu_2 \dots \mu_C\}$ that maximize the Q function

$$\theta = \underset{\mu_1 \dots \mu_C}{\operatorname{argmax}} Q\left(\theta | \theta^{(i-1)}\right)$$

$$= \underset{\mu_1 \dots \mu_C}{\operatorname{argmax}} \sum_{n=1}^{N} \left(\log \frac{1}{\sqrt{2\pi}\sigma} - \frac{1}{2\sigma^2} \sum_{c=1}^{C} E\left[z_c^{(n}\right] \left(x^{(n} - \mu_c\right)^2 \right)$$

$$= \underset{\mu_1 \dots \mu_C}{\operatorname{argmin}} \sum_{n=1}^{N} \sum_{c=1}^{C} E\left[z_c^{(n}\right] \left(x^{(n} - \mu_c\right)^2$$

– Which, computing the zeros of the partial derivative, yields:

$$\mu_c = \frac{1}{N} \sum_{n=1}^{N} E\left[z_c^{(n}\right] x^{(n} \quad (2)$$

– Equations (1) and (2) define a fixed-point algorithm that can be used to converge to a (local) maximum of the log-likelihood function

Relation to k-means clustering

A widely used vector quantization procedure can be derived from the EM algorithm by taking the limit $\sigma \to 0$

- In this case, we can see that $E[z_c^{(n}]$ collapses to 0 or 1

$$E\left[z_c^{(n}\right] = \frac{e^{-\frac{1}{2\sigma^2}\left(x^{(n}-\mu_c^{(i-1}\right)^2}}{\sum_{q=1}^{C} e^{-\frac{1}{2\sigma^2}\left(x^{(n}-\mu_q^{(i-1}\right)^2}} = \begin{cases} 1 & if \ \left\|x^{(n}-\mu_c^{(i-1}\right\|^2 > \left\|x^{(n}-\mu_q^{(i-1}\right\|^2 \ \forall q \\ 0 & otherwise \end{cases}$$

- And the EM update formula reduces to the popular k-means algorithm, which will be discussed in the next lecture

EM solution to generic mixture models

A derivation of the update equations for the full-covariance mixture model can be found in [Bilmes, 1997; Nabney, 2002]

- The final equations are provided here for those of you interested in experimenting with mixture models

$$P^{(i}(\omega_c) = \frac{1}{N}\sum_{n=1}^{N} P^{(i-1}\left(\omega_c|x^{(n}\right)$$

$$\mu_c^{(i} = \frac{\sum_{n=1}^{N} P^{(i-1}\left(\omega_c|x^{(n}\right)x^{(n}}{\sum_{n=1}^{N} P^{(i-1}(\omega_c|x^{(n})}$$

$$\Sigma_c^{(i} = \frac{\sum_{n=1}^{N} P^{(i-1}\left(\omega_c|x^{(n}\right)\left(x^{(n} - \mu_c^{(i}\right)\left(x^{(n} - \mu_c^{(i}\right)^T}{\sum_{n=1}^{N} P^{(i-1}(\omega_c|x^{(n})}$$

- Notice where the new parameters $\theta^{(i}$ and old parameters $\theta^{(i-1}$ appear on the RHS and compare these expressions to those in page 6

Example

The annulus problem

- A training set of $N = 900$ examples was generated using a uniform pdf inside an annulus with inner and outer radii of 1 and 2 units, respectively
- A mixture model with $C = 30$ Gaussians was used to model the distribution of the training set

Training procedure

- The centers of the Gaussians were initialized by choosing 30 arbitrary points from the training set
- The covariance matrices were initialized to be diagonal, with a large variance compared to that of the training data
  - To avoid singularities, at every iteration the covariance matrices computed with EM were regularized with a small multiple of the identity matrix
- Components whose mixing coefficients fell below a threshold were trimmed
  - This allowed the algorithm to produce a compact model with only a few of the initial C=30 Gaussian components

Illustrative results are provided in the next page