The EM Algorithm

This lecture introduces an important statistical estimation algorithm known as the EM or "expectation-maximization" algorithm. It reviews the situations in which EM works well and its advantages in those applications. In particular the use of EM to re-estimate the parameters of mixture models is considered. EM is compared to so-called "hard" methods such as the k-means algorithm.

Brief Outline

1. Maximum Likelihood Parameters:
   Why they are useful but hard to find.
2. The EM algorithm: missing information
   interpretation
3. Clustering Examples:
   fake and real mixtures of gaussians

Sam Roweis

The EM Algorithm Lecture
What's the Point ?

- Maximum likelihood parameter estimates:
  One definition of the "best" knob settings.
  Often impossible to find directly.
- The EM Algorithm:
  Finds ML parameters when the original (hard) problem can be broken up into two (easy) pieces:
  1. Estimate some "missing" or "unobserved" data from observed data and current parameters.
  2. Using this "complete" data, find the maximum likelihood parameter estimates.
- For EM to work, two things have to be easy:
  1. Guessing (estimating) missing data from data we have and our current guess of parameters.
  2. Solving for the ML parameters directly given the complete data.
- EM is typically used with mixture models, for example mixture of gaussians.
  The "missing" data are the labels showing which sub-model generated each datapoint.

Maximum Likelihood Parameters

- Given some data $\vec{z}_1, \vec{z}_2, \ldots, \vec{z}_n = \{\vec{z}_i\} = \mathcal{Z}$.
- Choose a model class (parameterized probability density $p(\vec{z}, \vec{\theta})$ over the range of $\vec{z}$).
- The *likelihood* function $L(\mathcal{Z}, \vec{\theta})$ of the data is the joint probability density of $\mathcal{Z}$ under the model:

$$L(\mathcal{Z}, \vec{\theta}) = \prod_i p(\vec{z}_i, \vec{\theta}) \quad \text{for i.i.d datapoints}$$

  L tells us how probable the particular dataset $\mathcal{Z}$ is under our particular choice of model $\vec{\theta}$
- The *maximum likelihood estimates* of the parameters, denoted $\vec{\theta}_{ml}$ are those which maximize $L(\mathcal{Z}, \vec{\theta})$ and depend only on the data (and model class):

$$\vec{\theta}_{ml}(\mathcal{Z}) = \text{argmax}_{\vec{\theta}} \quad L(\mathcal{Z}, \vec{\theta})$$

- This is one definition of the "best" estimate for the parameter vector because:
  1. $\vec{\theta}_{ml}$ makes the *observed* dataset the *most probable* dataset
  2. $\vec{\theta}_{ml}$ minimizes the *cost to communicate* the dataset

Finding ML Parameters

- Small problems: try all possible models $\vec{\theta}$
- Direct maximization: for *smooth* likelihood functions $\vec{\theta}_{ml}$ is a solution to:

$$\nabla L(\mathcal{Z}, \vec{\theta}) = \vec{0} \qquad \text{or}$$

$$\nabla \, log \, L(\mathcal{Z}, \vec{\theta}) = \vec{0}$$

- Example: The gaussian case is easy.
  Given some data points, if we choose our model class to be gaussians then it is easy to compute the maximum likelihood parameter estimates:
  The mean is the mean of the data and the variance is the variance of the data.
- In general, however, we cannot find $\vec{\theta}_{ml}$ directly. We must use an iterative optimization technique.
- Gradient descent (backprop) is one such technique; the EM algorithm is another.
- Convergence, speed, and local minima are all issues.

The EM Algorithm

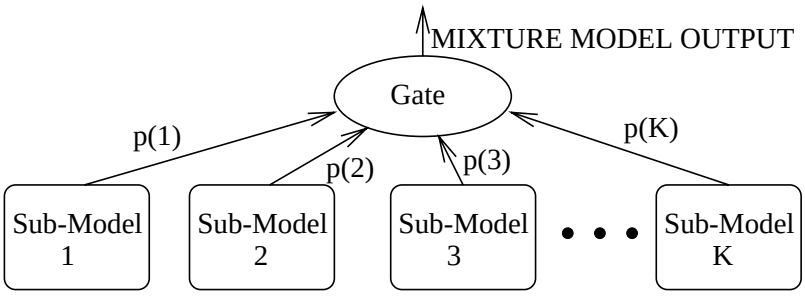
- The EM algorithm:
  - finds maximum likelihood parameters $\vec{\theta}_{ml}$ given a model and some data $\mathcal{Z}$
  - works best in situations where:
    1. the data is incomplete *or* *can be thought of as being incomplete*
    2. the maximum likelihood parameters are easy to find given observed *and* unobserved data
    3. the unobserved data are easy to estimate given some model parameters and the observed data
- The EM algorithm:

  E Step: **E**stimate a probability distribution $\hat{p}(\vec{u})$ for the unobserved data using the current parameter vector $\vec{\theta}^{(t-1)}$ and the observed data $\mathcal{Z}$ (easy).

  M Step: find the **M**aximum likelihood parameters given the observed data $\mathcal{Z}$ and the estimate $\hat{p}(\vec{u})$ of the unobserved data. Set $\vec{\theta}^{(t)}$ to these (also easy).

Typical Applications of EM

- EM is generally applied in situations where underlying model for the data is a *mixture model*.
- In a mixture model, there are many "sub-models", each of which has its own probability distribution which describes how it generates data when it is active.
  There is also a "mixer" or "gate" which controls how often each sub-model is active.



- Generally it is easy to find the ML parameters for each sub-model if we know which of the datapoints it generated. But our data is often unlabeled.
- So we use EM to estimate which sub-model was responsible for generating each point and then we find the ML parameters based on these estimates.
- Then we use the new ML parameters to re-estimate the responsibilities and iterate.

An Example: "Almost-Mixture of Gaussians"

- Recall that the probability implied by "hard" clustering is a collection of "clipped" gaussians.
- There is a faster way to learn in this model than competitive learning. Iterate:
  1. Quantize the datapoints using the existing prototype vectors and a nearest neighbour rule.
  2. Replace each prototype vector by the mean of the datapoints it quantized.
- This is the k-means algorithm (also known as Lloyd's algorithm or the Lloyd-Max algorithm).
- Each iteration is guaranteed to reduce the expected reconstruction error (increase the likelihood) and the algorithm is guaranteed to converge.
- The initial prototype vectors can be chosen on datapoints or at random.
- Notice that for any point in the space, the generative model has *only one way* to produce it.

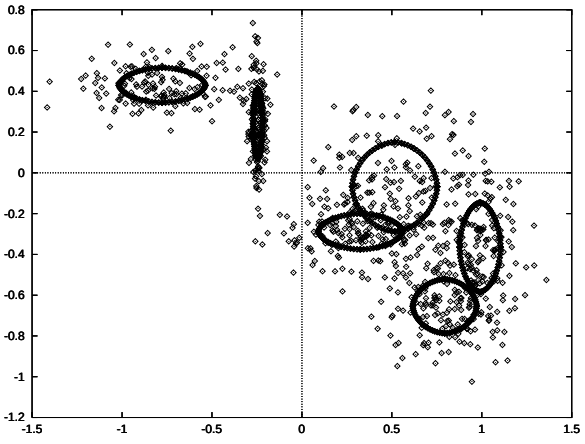
Real Mixture of Gaussians

- Goal: Fit a true MOG to $\mathcal{Z}$ maximizing $p(\mathcal{Z}|\vec{\theta})$
- Unobserved data $\mathcal{U}$ are the labels telling *which* gaussian generated each datapoint $\vec{z}_i$
- Parameter vector $\vec{\theta}$ are the means/variances
- In this case EM says:

  E Step: Compute the *responsibility* r that gaussian number k has for point $\vec{z}_i$ as:

$$r(k,i) = \frac{p(\vec{z}_i|k)}{\sum_{k'} p(\vec{z}_i|k')}$$

  M Step: Replace the mean and variance of each gaussian with the *responsibility weighted* average across data



- Notice that for any point in the space, the generative model has *many ways* to produce it.

A Report Card for EM

- Some good things about EM:
  - no learning rate parameter
  - very fast for low dimensions
  - each iteration guaranteed to improve likelihood
  - adapts unused units rapidly
  - finds both the model θ and the optimal stochastic coding distribution S for θ
- Some bad things about EM:
  - can get stuck in local minima
  - ignores model cost (how many blobs ?)
  - both steps require considering *all* explanations of the data which is an exponential amount of work in the dimension of θ
- Two interesting notes:
  - The Baum-Welsh algorithm for training HMMs is a special case of EM
  - The Boltzmann machine learning rule is a generalized EM algorithm

Some Ways to Improve EM

- Generalized variants - do not find maximum likelihood model or Boltzmann distribution
- Incremental variants - recalculate model after each unobserved variable
- Sparse variants - freeze some values for a while
- Use a model that permits only factorial distributions over its parameters
  (a network analogy: must communicate hidden units independently)
  PCA is purely factorial, Clustering is opposite
- This is the idea behind the Wake-Sleep algorithm and Helmholtz machines