CPSC 440: Advanced Machine Learning

Expectation Maximization

Mark Schmidt

University of British Columbia

Winter 2022

Outline

Learning with Hidden Values

- We often want to learn with unobserved/missing/hidden/latent values.
- For example, we could have a dataset like this:

$$X = \begin{bmatrix} N & 33 & 5 \\ L & 10 & 1 \\ F & ? & 2 \\ M & 22 & 0 \end{bmatrix}, y = \begin{bmatrix} -1 \\ +1 \\ -1 \\ ? \end{bmatrix}.$$

- Or we could be fitting a mixture model without knowing the clusters.
- Missing values are very common in real datasets.

- An important issue to consider: why is data missing?

Missing at Random (MAR)

- We'll focus on data that is missing at random (MAR):
  - Assume that the reason ? is missing does not depend on the missing value.
    - Formal definition in bonus slides.
  - This definition doesn't agree with intuitive notion of "random":
    - A variable that is *always* missing would be "missing at random".
    - The intuitive/stronger version is missing completely at random (MCAR).
- Examples of MCAR and MAR for digit data:
  - Missing random pixels/labels: MCAR.
  - Hide the the top half of every digit: MAR.
  - Hide the labels of all the "2" examples: not MAR.
- We'll consider MAR, because otherwise you need to model why data is missing.

Missing at Random (MAR) Formally

- Let's formally define MAR in the context of density estimation.
- Our "observed" data would be a matrix X containing ? values.
- Our "complete" data would be the matrix X with the ? values "filled in".
  - We know all x_j^i values in this matrix, even the ones that are ? in the observed data.
- Use $z_j^i = 1$ if x_i^j is ? in the "observed" data.
- We say that data is MAR in the observed data X if

$$z_j^i \perp x_j^i,$$

that the fact that x_j^i is missing (z_j^i) is independent of the value of x_j^i.

  - Specific values of the variables are not being hidden.

Missing at Random: Example

Actions

Question about MAR

Hey folks,

There was an interesting question about MAR today, which I'll re-phrase as:
- Suppose you training examples x^i contain images of digits and the corresponding class labels.
- And suppose that we hide parts of the images of all the 2 digits, but the class label isn't every hidden.
- Is the above MAR?

It depends a little bit on how you assume the data was created, but if you assume the data was created by someone wanting to draw a 2, then this *would* be MAR. In this case, the the class label 2 is causing the image to look a certain way and causing the data to be missing, so if you observe the "cause" it makes the values of the pixels independent of the fact that they are missing.

(We'll show how to do the above reasoning with graphical models later in the course.

Imputation Approach to MAR Variables

- Consider a dataset with MAR values:

$$X = \begin{bmatrix} N & 33 & 5 \\ F & 10 & 1 \\ F & ? & 2 \\ M & 22 & 0 \end{bmatrix}, y = \begin{bmatrix} -1 \\ +1 \\ -1 \\ ? \end{bmatrix}.$$

- Imputation method is one of the first things we might try:
  0. Initialization: find parameters of a density model (often using "complete" examples).
  1. Imputation: replace each ? with the most likely value.
  2. Estimation: fit model with these imputed values.

- You could also alternate between imputation and estimation.

Semi-Supervised Learning

- Important special case of MAR is semi-supervised learning.

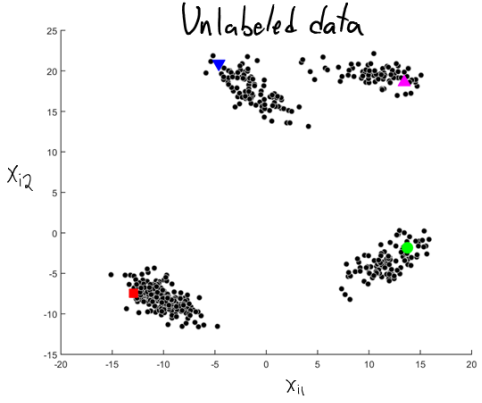
$$X = \begin{bmatrix} & \\ & \end{bmatrix}, \quad y = \begin{bmatrix} \\ \end{bmatrix},$$

$$\bar{X} = \begin{bmatrix} & \\ & \\ & \\ & \\ & \end{bmatrix}, \quad \bar{y} = \begin{bmatrix} ? \\ ? \\ ? \\ ? \\ ? \end{bmatrix}.$$

- Motivation for training on labeled data (X, y) and unlabeled data $\bar{X}$:
  - Getting labeled data is usually expensive, but unlabeled data is usually cheap.
    - For speech recognition: easy to get speech data, hard to get annotated speech data.

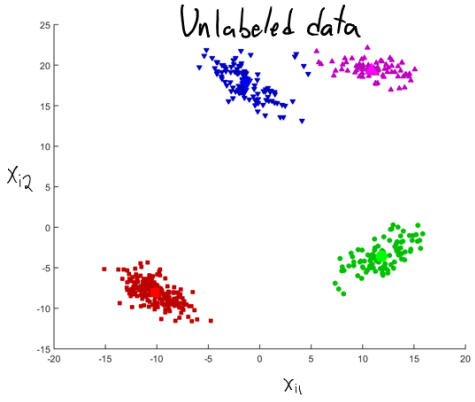
Semi-Supervised Learning

- Why should unlabeled data tell us anything about labels?
  - Usually, we assume that similar features $\rightarrow$ similar labels.
  - Consider the following example where we have one labeled example for each class.



Semi-Supervised Learning

- Why should unlabeled data tell us anything about labels?
  - Usually, we assume that similar features $\rightarrow$ similar labels.
  - Consider the following example where we have one labeled example for each class.



Semi-Supervised Learning

- Important special case of MAR is semi-supervised learning.

$$X = \begin{bmatrix} & \\ & \end{bmatrix}, \quad y = \begin{bmatrix} \\ \end{bmatrix},$$

$$\bar{X} = \begin{bmatrix} & \\ & \\ & \\ & \\ & \end{bmatrix}, \quad \bar{y} = \begin{bmatrix} ? \\ ? \\ ? \\ ? \\ ? \end{bmatrix},$$

- Imputation approach is called self-taught learning:
  - Alternate between guessing $\bar{y}$ and fitting the model with these values.
  - If you use LDA, you get a semi-supervised version of k-means.
    - Alternatve between updating means of clusters, and update the "unknown clusters" $\bar{y}$.

Back to Mixture Models

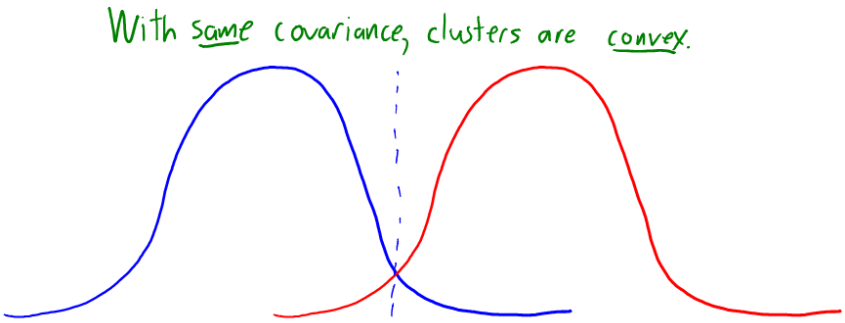
- To fit mixture models we often make n MAR new variables z^i.
- Why???
- Consider mixture of Gaussians, and let z^i be the cluster number of example i:
  - So $z^i \in \{1, 2, \cdots, k\}$ tells you which Gaussian generated example i.
  - Given the z^i it's easy to optimize the parameters of the mixture model.
    - Solve for $\{\pi_c, \mu_c, \Sigma_c\}$ maximizing $p(x^i, z^i)$ (learning step in GDA).
  - Given $\{\pi_c, \mu_c, \Sigma_c\}$ it's easy to optimize the clusters z^i:
    - Find the cluster c maximizing $p(x^i, z_i = c)$ (prediction step in GDA).

Imputation Approach for Mixtures of Gaussians

- Consider mixture of Gaussians with the choice $\pi_c = 1/k$ and $\Sigma_c = I$ for all c.
- Here is the imputation approach for fitting a mixtures of Gaussian:
  - Randomly pick some initial means μ_c.
  - Assigns x^i to the closest mean (imputation of missing z^i values).
    - This is how you maximize $p(x^i, z^i)$ in terms of z^i.
  - Set μ_c to the mean of the points assigned to cluster c (parameter update).
    - This is how you maximize $p(x^i, z^i)$ in terms of μ_c given the z^i.
- This is exactly k-means clustering.

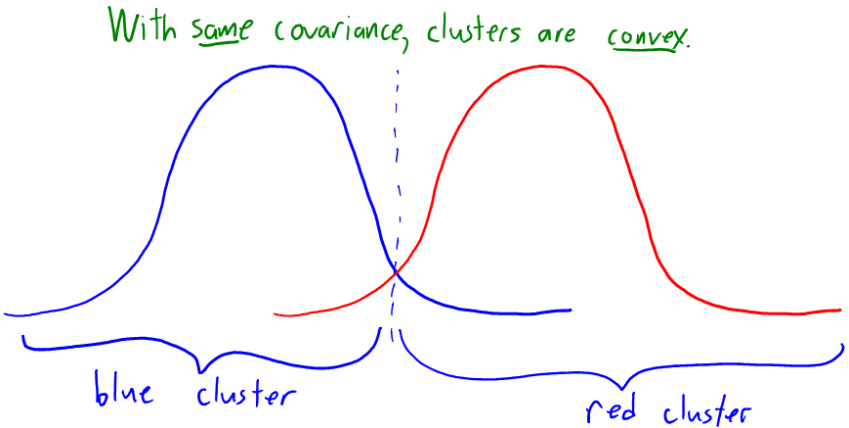
K-Means vs. Mixture of Gaussians

- K-means can be viewed as fitting mixture of Gaussians (common Σ_c).
  - But variable Σ_c in mixture of Gaussians allow non-convex clusters.



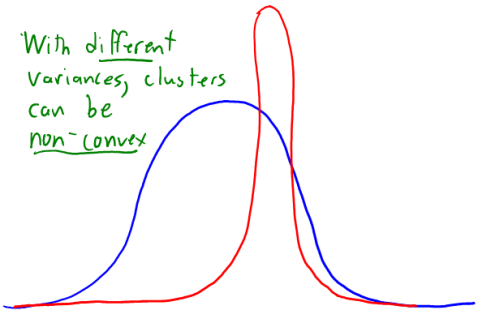
K-Means vs. Mixture of Gaussians

- K-means can be viewed as fitting mixture of Gaussians (common Σ_c).
  - But variable Σ_c in mixture of Gaussians allow non-convex clusters.



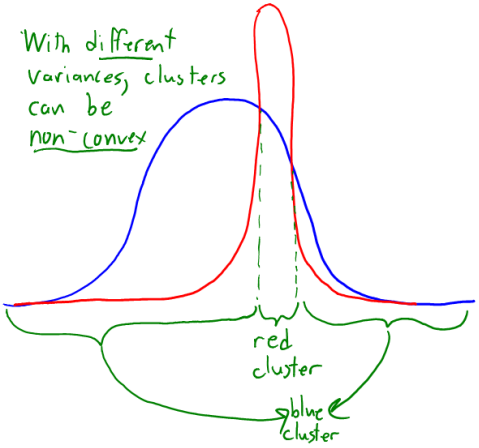
K-Means vs. Mixture of Gaussians

- K-means can be viewed as fitting mixture of Gaussians (common Σ_c).
  - But variable Σ_c in mixture of Gaussians allow non-convex clusters.



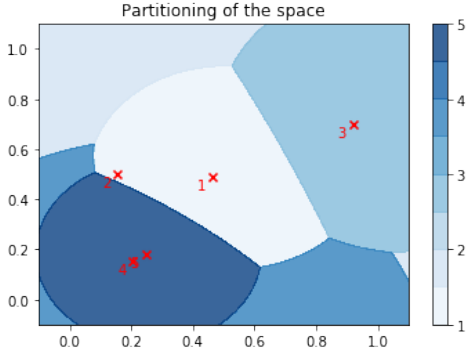
K-Means vs. Mixture of Gaussians

- K-means can be viewed as fitting mixture of Gaussians (common Σ_c).
  - But variable Σ_c in mixture of Gaussians allow non-convex clusters.



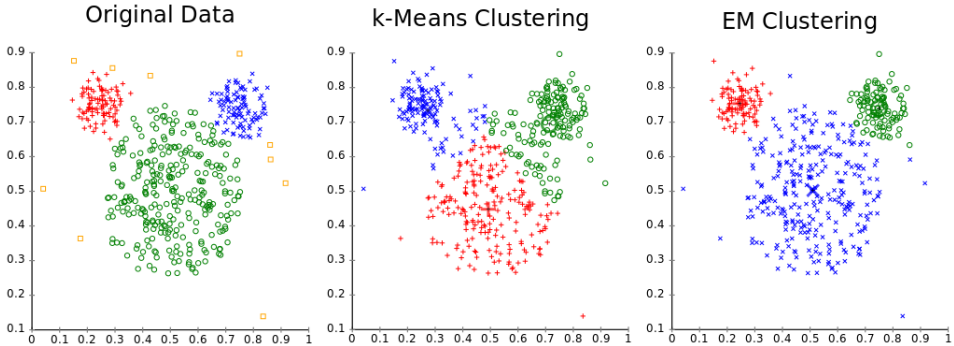
K-Means vs. Mixture of Gaussians

- K-means can be viewed as fitting mixture of Gaussians (common Σ_c).
  - But variable Σ_c in mixture of Gaussians allow non-convex clusters.



K-Means vs. Mixture of Gaussians

- K-means can be viewed as fitting mixture of Gaussians (common Σ_c).
  - But variable Σ_c in mixture of Gaussians allow non-convex clusters.



https://en.wikipedia.org/wiki/K-means_clustering

Outline

Parameters, Hyper-Parameters, and Nuisance Parameters

- Are the ? values "parameters" or "hyper-parameters"?

- Parameters:
  - Variables in our model that we optimize based on the training set.

- Hyper-Parameters:
  - Variables that control model complexity, typically set using validation set.
  - Often become degenerate if we set these based on training data.
  - We sometimes add optimization parameters in here like step-size.
    - Because "optimizing worse" can decrease overfitting.

- Nuisance Parameters:
  - Not part of the model and not really controlling complexity.
  - An alternative to optimizing ("imputation") is to consider all values.
    - Based on marginalization rule for probabilities.
    - Consider all possible imputations, and weight them by their probability.

Drawbacks of Imputation Approach

- Imputation approach to MAR variables treats the ? as parameters.
  - Use density estimator to find the "best" way to "fill in" the missing values.
  - Now fit the "complete data" using a standard method.

- But "hard" assignments of missing values leads to propagation of errors.
  - What if cluster is ambiguous in k-means clustering?
  - What if label is ambiguous in "self-taught" learning?

- Expectation maximization (EM) treats the ? as nuissance parameters:
  - Use probabilities of different assignments ("soft" assignments) instead of imputation.
  - If the MAR values are obvious, this will act like the imputation approach.
  - For ambiguous MAR values, takes into account our uncertainty.

Example: Expectation Maximization for Mixture of Gaussians

- EM for mixtures needs "probability that example i comes from mixture c".
  - We call this the responsibility r_c^i of cluster c for example i.
    - EM computes these given parameters Θ^t at iteration t.

- You can compute the responsibilities with Bayes rule:

$$r_c^i \triangleq p(z^i = c \mid x^i, \Theta^t) = \frac{p(x^i \mid z^i = c, \Theta^t) p(z^i = c \mid \Theta^t)}{\sum_{c'=1}^{k} p(x^i \mid z^i = c', \Theta^t) p(z^i = c' \mid \Theta^t)},$$

  which depends on the mixture proportions $p(z^i = c \mid \Theta^t)$ and the likelihood.
  - Though you may get underflow when computing r_c^i (see bonus for log-domain tricks).

- Can think of imputation/k-means as using $r_c^i = 1$ for most likely cluster.
  - And $r_c^i = 0$ for all other clusters.
- EM insteads "soft-assign" each training example to each cluster.
  - Example i could be 40% in cluster 1, 35% in cluster 2, and 25% in cluster 3.

Example: Expectation Maximization for Mixture of Gaussians

- The EM algorithm for training mixtures of Gaussans repeats the following steps:
  1. Computer probability that example i is in cluster c based on parameters Θ^t.
  $$r_c^i = p(z^i = c \mid x^i, \Theta^t).$$
  2. Update cluster probabilities based on number of examples soft-assigned to clusters.
  $$\pi_c^{t+1} = \frac{1}{n} \sum_{i=1}^{n} r_c^i.$$
  3. Update means based on means of examples soft-assigned to each cluster.
  $$\mu_c^{t+1} = \frac{1}{\sum_{i=1}^{n} r_c^i} \sum_{i=1}^{n} r_c^i x^i.$$
  4. Update covariances based on covariances of examples soft-assigned to each cluster.
  $$\Sigma_c^{t+1} = \frac{1}{\sum_{i=1}^{n} r_c^i} \sum_{i=1}^{n} r_c^i (x^i - \mu_c^{t+1})(x^i - \mu_c^{t+1})^\top.$$
- Video: `https://www.youtube.com/watch?v=B36fzChfyGU`

Expectation Maximization Notation

- Expectation maximization (EM) is an optimization algorithm for MAR values:
  - Applies to problems that are easy to solve with "complete" data (i.e., you knew ?).
  - Allows probabilistic or "soft" assignments to MAR (or other nuisance) variables.
    - Imputation approach is sometimes called "hard" EM.

- EM is among the most cited paper in statistics.
  - Special cases independently discovered in 50s-70s, general case in 1977.

- EM notation: we use O as observed data and H as hidden (?) data.
  - Semi-supervised learning: observe $O = \{X, y, \bar{X}\}$ but don't observe $H = \{\bar{y}\}$.
  - Mixture models: observe data $O = \{X\}$ but don't observe clusters $H = \{z^i\}_{i=1}^n$.

- We use Θ as parameters we want to optimize.
  - In Gaussian mixtures this will be the π_c, μ_c, and Σ_c variables.

The Two Likelihoods: "Complete" and "Marginal"

- "Complete" likelihood: likelihood with imputed hidden values, $p(O, H \mid \Theta)$.
  - We assume that this is "nice". Maybe it has a closed-form MLE or is convex.
- "Marginal" likelihood: likelihood with unknown hidden values, $p(O \mid \Theta)$.
  - This is our usual likelihood, the thing we actually want to optimize.
- The "complete" and "marginal" likelihoods are related by the marginalization rule:

$$\underbrace{p(O \mid \Theta)}_{\text{"marginal"}} = \sum_{H_1} \sum_{H_2} \cdots \sum_{H_m} p(O, H \mid \Theta) = \sum_{H} \underbrace{p(O, H \mid \Theta)}_{\text{"complete likelihood"}} .$$

where we sum over all possible $H \equiv \{H_1, H_2, \ldots, H_m\}$.

- For mixture models, this sums over all possible clusterings (k^n values).
- Replace the sums by integrals for continuous hidden values.

Expectation Maximization Bound

- The negative log-likelihood (that we want to optimize) thus has the form

$$-\log p(O \mid \Theta) = -\log \left(\sum_H p(O, H \mid \Theta) \right),$$

- which has a sum inside the log.
  - This does not preserve convexity: minimizing it is usually NP-hard.

- Both EM and imputation are based on the approximation:

$$-\log \left(\sum_H p(O, H \mid \Theta) \right) \approx -\sum_H \alpha_H \log p(O, H \mid \Theta)$$

where α_H is some probability for the assignment H to the hidden variables.

- An expectation over "complete" log-likelihood.
- This is useful when the approximation is easier to minimize.
  - The specific α_H chosen by EM makes the approximation tight at Θ^t.

Expectation Maximization Bound

- Each iteration of EM and imputation optimize the approximation:

$$\Theta^{t+1} \in \operatorname*{argmin}_{\Theta} -\sum_{H} \alpha_H^t \log p(O, H \mid \Theta).$$

  where the probabilities α_H^t are updated after each iteration t.

- Imputation sets $\alpha_H^t = 1$ for the most likely H given Θ^t (all other $\alpha_H^t = 0$).
  - It assumes that the imputations are correct, then optimizes with the guess

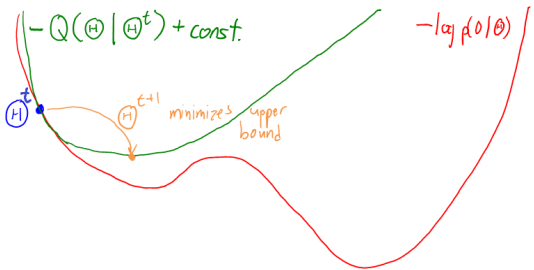
- In EM we set $\alpha_H^t = p(H \mid O, \Theta^t)$, weighting H by probability given Θ^t.
  - It weighs different imputations by their probability, then optimizes.

Expectation Maximization as Bound Optimization

- We'll show that the EM approximation minimizes an upper bound,

$$\underbrace{-\log p(O \mid \Theta)}_{\text{what we want}} \leq \underbrace{-\sum_{H} p(H \mid O, \Theta^t) \log p(O, H \mid \Theta)}_{Q(\Theta \mid \Theta^t)\text{: what we optimize}} + \text{const.},$$

- Geometry of expectation maximization as "optimizing an upper bound":
  - At each iteration t we optimize a bound on the function.



Expectation Maximization (EM)

- So EM starts with Θ^0 and sets Θ^{t+1} to maximize $Q(\Theta \mid \Theta^t)$.

- This is typically written as two steps:
  1. E-step: Define expectation of complete log-likelihood given last parameters Θ^t,

$$Q(\Theta \mid \Theta^t) = \sum_H \underbrace{p(H \mid O, \Theta^t)}_{\text{fixed weights } \alpha_H^t} \underbrace{\log p(O, H \mid \Theta)}_{\text{nice term}}$$
$$= \mathbb{E}_{H \mid O, \Theta^t}[\log p(O, H \mid \Theta)],$$

     which is a weighted version of the "nice" $\log p(O, H)$ values.
     - For mixtures of Gaussians, E-step updates r_c^i (like clustering step in k-means).
  2. M-step: Maximize this expectation to generate new parameters Θ^{t+1},

$$\Theta^{t+1} = \underset{\Theta}{\operatorname{argmax}}\, Q(\Theta \mid \Theta^t).$$

     - For mixture of Gaussians, M-step updates π_c, μ, and Σ_c (like mean in k-means).
- But I don't like the terms "E-step" and "M-step".
  - For mixture models it separates into two steps, but for many models it doesn't.

Expectation Maximization for Mixture Models

- In the case of a mixture model with extra "cluster" variables z^i, EM uses

$$\begin{aligned}
Q(\Theta \mid \Theta^t) &= \mathbb{E}_{z \mid X,\Theta^t}[\log p(X, z \mid \Theta)] \\
&= \sum_{z^1=1}^{k} \sum_{z^2=1}^{k} \cdots \sum_{z^n=1}^{k} \underbrace{p(z \mid X, \Theta^t)}_{\alpha_z} \underbrace{\log p(X, z \mid \Theta)}_{\text{"nice"}} \quad (k^n \text{ terms}) \\
&= \sum_{z^1=1}^{k} \sum_{z^2=1}^{k} \cdots \sum_{z^n=1}^{k} \left(\prod_{i=1}^{n} p(z^i \mid x^i, \Theta^t) \right) \left(\sum_{i=1}^{n} \log p(x^i, z^i \mid \Theta) \right) \\
&= \text{(see EM notes, tedious use of distributive law and independences)} \\
&= \sum_{i=1}^{n} \sum_{z^i=1}^{k} p(z^i \mid x^i, \Theta^t) \log p(x^i, z^i \mid \Theta) \quad (nk \text{ terms}).
\end{aligned}$$

- Sum over k^n clusterings turns into sum over nk 1-example assignments.
  - Same simplification happens for semi-supervised learning, we'll discuss why later.

Expectation Maximization for Mixture Models

- In the case of a mixture model with extra "cluster" variables z^i EM uses

$$Q(\Theta \mid \Theta^t) = \sum_{i=1}^{n} \sum_{z^i=1}^{k} \underbrace{p(z^i \mid x^i, \Theta^t)}_{r_c^i} \log p(x^i, z^i \mid \Theta).$$

- This is just a weighted version of the usual log-likelihood.
  - Update is solution of a weighted Gaussian, weighted Bernoulli, and so on.
    - Closed-form solution in these simple cases.

- To derive the simple EM updates that were shown for mixture of Gaussians:
  - Take gradient of above and set it to 0, then solve for π_c, μ_c and Σ_c.
    - Then you re-compute responsibilities and repeat.

Discussing of EM for Mixtures of Gaussians

- EM and mixture models are used in a ton of applications.
  - One of the default unsupervised learning methods.
  - Not just for mixture models:
    - Semi-supervised learning.
    - Density estimation with missing values in matrix.
- EM usually doesn't reach global optimum.
  - Classic solution: restart the algorithm from different initializations.
  - Lots of work in CS theory on getting better initializations (like "k-means++").
- MLE for some clusters may not exist (e.g., only responsible for one point).
  - Use MAP estimates or remove these clusters.
- EM does not fix "propagation of errors" from imputation approach.
  - But it reduces problem by incorporating a "confidence" over different imputations.
- Can you make it robust?
  - Use mixture of Laplace of student t distributions.
  - Don't have closed-form EM steps: compute responsibilities then need to optimize.

Outline

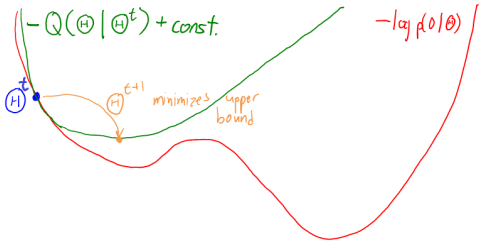
Monotonicity of EM

- Classic result is that EM iterations are monotonic:

$$\log p(O \mid \Theta^{t+1}) \geq \log p(O \mid \Theta^{t}),$$

- We don't need a step-size and this is useful for debugging.

- We can show this by proving that the below picture is "correct":



- The Q function leads to a global bound on the original function.
- At Θ^t the bound matches original function.
  - So if you improve on the Q function, you improve on the original function.

Monotonicity of EM

- Let's show that the Q function gives a global upper bound on NLL:

$$
\begin{aligned}
-\log p(O \mid \Theta) &= -\log \left(\sum_H p(O, H \mid \Theta) \right) && \text{(marginalization rule)} \\
&= -\log \left(\sum_H \alpha_H \frac{p(O, H \mid \Theta)}{\alpha_H} \right) && \text{(for } \alpha_H \neq 0\text{)} \\
&\leq -\sum_H \alpha_H \log \left(\frac{p(O, H \mid \Theta)}{\alpha_H} \right),
\end{aligned}
$$

because $-\log(z)$ is convex and the α_H are a convex combination.

Monotonicity of EM

- Using that log turns multiplication into addition we get

$$\begin{aligned} -\log p(O \mid \Theta) &\leq -\sum_H \alpha_H \log\left(\frac{p(O, H \mid \Theta)}{\alpha_H}\right) \\ &= \underbrace{-\sum_H \alpha_H \log p(O, H \mid \Theta)}_{Q(\Theta \mid \Theta^t)} + \underbrace{\sum_H \alpha_H \log \alpha_H}_{\text{negative entropy}} \\ &= -Q(\Theta \mid \Theta^t) - \text{entropy}(\alpha), \end{aligned}$$

  so we have the first part of the picture, $-\log p(O \mid \Theta^{t+1}) \leq -Q(\Theta|\Theta^t) + \text{const}$.
  - Entropy is a measure of how "random" the α_H values are.
  - Q behaves more like true objective for H that are more "predictable".

- Now we need to show that this holds with equality at Θ^t.

Bound on Progress of Expectation Maximization

- To show equality at Θ^t we use definition of conditional probability,

$$p(H \mid O, \Theta^t) = \frac{p(O, H \mid \Theta^t)}{p(O \mid \Theta^t)} \quad \text{or} \quad \log p(O \mid \Theta^t) = \log p(O, H \mid \Theta^t) - \log p(H \mid O, \Theta^t)$$

- Multiply by α_H and summing over H values,

$$\sum_H \alpha_H \log p(O \mid \Theta^t) = \underbrace{\sum_H \alpha_H \log p(O, H \mid \Theta^t}_{Q(\Theta^t \mid \Theta^t)} - \sum_H \alpha_H \log \underbrace{p(H \mid O, \Theta^t)}_{\alpha_H}.$$

- Which gives the result we want:

$$\log p(O \mid \Theta^t) \underbrace{\sum_H \alpha_H}_{=1} = Q(\Theta^t \mid \Theta^t) + \text{entropy}(\alpha),$$

Bound on Progress of Expectation Maximization

- We have shown the following two bounds for EM:

$$\underbrace{\log p(O \mid \Theta)}_{\text{what we want to maximizie}} \geq \underbrace{Q(\Theta \mid \Theta^t)}_{\text{what EM maximizes}} + \text{entropy}(\alpha)$$

$$\underbrace{\log p(O \mid \Theta^t)}_{\text{what we want to maximize}} = \underbrace{Q(\Theta^t \mid \Theta^t)}_{\text{what EM maximizes}} + \text{entropy}(\alpha).$$

- Subtracting these and using $\Theta = \Theta^{t+1}$ gives a stronger result,

$$\log p(O \mid \Theta^{t+1}) - \log p(O \mid \Theta^t) \geq Q(\Theta^{t+1} \mid \Theta^t) - Q(\Theta^t \mid \Theta^t),$$

  that we improve objective by at least the decrease in Q.
  - This isn't enough for convergence, but EM converges under weak assumptions.
- Inequality holds for any choice of Θ^{t+1}.
  - Approximate M-steps are ok: we just need to decrease Q to improve likelihood.

- Unlike imputation that optimizes MAR values, considers all possible imputations.
  - MAR values "nuissance parameters": there might not be obvious "correct" imputations.

EM for MAP Estimation

- We can also use EM for MAP estimation. With a prior on Θ our objective is:

$$\underbrace{\log p(O \mid \Theta) + \log p(\Theta)}_{\text{what we optimize in MAP}} = \log \left(\sum_H p(O, H \mid \Theta) \right) + \log p(\Theta).$$

- EM iterations take the form of a regularized weighted "complete" NLL,

$$\Theta^{t+1} \in \underset{\Theta}{\text{argmax}} \left\{ \underbrace{\sum_H \alpha_H^t \log p(O, H \mid \Theta)}_{Q(\Theta \mid \Theta^t)} + \log p(\Theta) \right\},$$

- Now guarantees monotonic improvement in MAP objective.
  - This still has a closed-form solution for "conjugate" priors (defined later).

- For mixture of Gaussians with $-\log p(\Theta_c) = \lambda \text{Tr}(\Theta_c)$ for precision matrices Θ_c:
  - Closed-form solution that satisfies positive-definite constraint (no $\log|\Theta|$ needed).

Alternate View of EM as BCD

- We showed that given α the M-step minimizes in Θ the function

$$F(\Theta, \alpha) = -\mathbb{E}_{\alpha}[\log p(O, H \mid \Theta)] - \text{entropy}(\alpha).$$

- The E-step minimizes this function in terms of α given Θ.
  - Setting $\alpha_H = p(H \mid O, \Theta)$ minimizes it.
- Note that F is not the NLL, but F and the NLL have same stationary points.
- From this perspective, we can view EM as a block coordinate descent method.
- This perspective is also useful if you want to do approximate E-steps.

Alternate View of EM as KL-Proximal

- Using definitions of expectation and entropy and α in the last slide gives

$$\begin{aligned}
F(\Theta, \alpha) &= -\sum_H p(H \mid O, \theta^t) \log p(O, H \mid \Theta) + \sum_H p(H \mid O, \theta^t) \log p(H \mid O, \theta^t) \\
&= -\sum_H p(H \mid O, \theta^t) \log \frac{p(O, H \mid \theta)}{p(H \mid O, \theta^t)} \\
&= -\sum_H p(H \mid O, \theta^t) \log \frac{p(H \mid O, \theta) p(O \mid \theta)}{p(H \mid O, \theta^t)} \\
&= -\sum_H \log p(O \mid \Theta) - \sum_H p(H \mid O, \theta^t) \log \frac{p(H \mid O, \theta)}{p(H \mid O, \theta^t)} \\
&= NLL(\Theta) + \mathsf{KL}(p(H \mid O, \theta^t) \mid\mid p(H \mid O, \theta)).
\end{aligned}$$

- From this perspective, we can view EM as a "proximal point" method.
  - Classical proximal point method uses $\frac{1}{2}\|\theta^t - \theta\|^2$, EM uses KL divergence.
- From this view we can see that EM doesn't depend on parameterization of Θ.
- If we linearize NLL and we multiply KL term by $1/\alpha_k$ (step-size), we get the natural gradient method.

EM Alternatives

- Are there alternatives to for optimizing marginal likelihood EM?
  - Could use gradient descent, SGD, and so on.
    - We now know that EM converges faster than gradient descent for standard mixture models.
  - Many variations on EM to speed up its convergence (for example, "adaptive" bound optimization).
  - Spectral and other recent methods have some global guarantees.