CS4487
Lecture 7.1: The Expectation Maximization Algorithm

Dr. Kede Ma

Department of Computer Science
City University of Hong Kong

Slides template by courtesy of Benjamin M. Marlin

Course Project: DeepFake Detection V2

- **Background**
  - Techniques for convincing manipulations of digital face images exist for decades via the use of visual effects
  - Recent technologies in deep learning, especially "DeepFake," have led to a dramatic increase in the realism of fake content
  - While there are entertaining applications of DeepFake, potential weaponization of such techniques has raised great concerns
  - DeepFake detection has become a pressing issue and a hot spot of research

Course Project: DeepFake Detection V2

- **Task:** A binary classification problem
  - Input: RGB face images
  - Output: A binary label to indicate whether an image is fake or not
- **Training set**: $12,000$ images, within which $8,000$ images contain fake faces, and the rest images are real
  - Download link: `https://drive.google.com/drive/folders/1n18AqgUF_KqKbcxuLXrTcLq-pyfhSJD9?usp=sharing`
- **Test sets**: Will not be released to the students
  - Students have THREE chances to get access to the test sets
  - The best result among the three will be used for ranking
- **Tip**: It is a good practice to divide the available set further into training, validation, and test set

Course Project: DeepFake Detection V2

- **Group project:** A group of at most three students is allowed
- **What to hand in:** a single ipython notebook (.ipynb) as the project report with source code files included
  - Source files must contain the training code for reproduction
  - An extra PDF file as project report is not recommended
- **When to hand in:** Dec. 4, 2022, 11:59 pm
- **Where to hand in:**
  - Submit the clean and runable test code to the TAs and wait for the result update in a Kaggle in-class competition (link will be available soon)
  - Submit the .ipynb file via Canvas
- **GPUs**: Wait for the confirmation from the CS Lab

Course Project: DeepFake Detection V2

- **Grading** (Totally 30 points)
  - 30.0% - Technical correctness (whether the methodologies/algorithms are correctly used)
  - 30.0% - Experiment and analysis
    - More points for thoroughness and testing interesting cases (e.g., different parameter settings)
    - More points for insightful observations and analysis (e.g., failure analysis)
  - 20.0% - Quality of the written report (organized, complete descriptions, etc.)
  - 10.0% - Quality of project presentation (tentatively held in Week 13)
    - **Note**: you have the option **not** to present your project
  - 10.0% - Reserved for Top-3 teams based on the test set performance

Supervised vs Unsupervised Learning

- Supervised learning considers input-output pairs $(\mathbf{x}, y)$
  - Learn a mapping f from input to output
  - Classification: output $y \in \{-1, 1\}$
  - Regression: output $y \in \mathbb{R}$
  - "Supervised" here means that the algorithm is learning the mapping that we want

- Unsupervised learning only considers the input data $\mathbf{x}$
  - There is no output value
  - **Goal**: Try to discover inherent properties in the data
    - Clustering
    - Dimensionality reduction

Expectation Maximization (EM)

- EM solves a maximum likelihood problem of the form

$$L(\boldsymbol{\theta}) = \sum_{i=1}^{M} \log p(\mathbf{x}^{(i)}; \boldsymbol{\theta}) = \sum_{i=1}^{M} \log \sum_{z^{(i)}=1}^{K} p(\mathbf{x}^{(i)}, z^{(i)}; \boldsymbol{\theta})$$

  - $\boldsymbol{\theta}$: Parameters of the probabilistic model we try to find
  - $\{\mathbf{x}^{(i)}\}_{i=1}^{M}$: Observed training examples
  - $\{z^{(i)}\}_{i=1}^{M}$: Unobserved latent variables (e.g., in GMM, $z^{(i)}$ indicates which one of the K clusters $\mathbf{x}^{(i)}$ belongs to, which is unobserved)

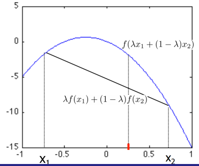
Jensen's Inequality

- Suppose $f : \mathbb{R}^N \mapsto \mathbb{R}$ is **concave**, then for all probability distributions p, we have

$$f(\mathbb{E}_{\mathbf{x}\sim p}[\mathbf{x}]) \geq \mathbb{E}_{\mathbf{x}\sim p}[f(\mathbf{x})]$$

  - The subscript $\mathbf{x} \sim p$ indicates that the expectation is taken w.r.t. random variable $\mathbf{x}$ drawn from the probability distribution p
  - The equality holds if and only if 1) $\mathbf{x}$ is constant or 2) f is an affine function (i.e., $f(\mathbf{x}) = \mathbf{a}^T\mathbf{x} + b$)
  - The above inequality also holds if the expectation is computed for a deterministic function of $\mathbf{x}$, (e.g., $g(\mathbf{x})$)

Illustration:
$p(x_1) = \lambda$,
$p(x_2) = 1 - \lambda$



EM Derivation

$$
\begin{aligned}
\sum_{i=1}^{M} \log \sum_{z^{(i)}=1}^{K} p(\mathbf{x}^{(i)}, z^{(i)}; \boldsymbol{\theta}) &= \sum_{i=1}^{M} \log \sum_{z^{(i)}=1}^{K} q(z^{(i)}) \frac{p(\mathbf{x}^{(i)}, z^{(i)}; \boldsymbol{\theta})}{q(z^{(i)})} \\
&= \sum_{i=1}^{M} \log \mathbb{E}_{z^{(i)} \sim q} \left[\frac{p(\mathbf{x}^{(i)}, z^{(i)}; \boldsymbol{\theta})}{q(z^{(i)})} \right] \\
&\geq \sum_{i=1}^{M} \mathbb{E}_{z^{(i)} \sim q} \log \left[\frac{p(\mathbf{x}^{(i)}, z^{(i)}; \boldsymbol{\theta})}{q(z^{(i)})} \right] \\
&= \sum_{i=1}^{M} \sum_{z^{(i)}=1}^{K} q(z^{(i)}) \log p(\mathbf{x}^{(i)}, z^{(i)}; \boldsymbol{\theta}) \\
&\quad - \sum_{i=1}^{M} \sum_{z^{(i)}=1}^{K} q(z^{(i)}) \log q(z^{(i)})
\end{aligned}
$$

EM Derivation

$$L(\boldsymbol{\theta}) = \sum_{i=1}^{M} \log \sum_{z^{(i)}=1}^{K} p(\mathbf{x}^{(i)}, z^{(i)}; \boldsymbol{\theta}) \geq \sum_{i=1}^{M} \sum_{z^{(i)}=1}^{K} q(z^{(i)}) \log p(\mathbf{x}^{(i)}, z^{(i)}; \boldsymbol{\theta}) - \sum_{i=1}^{M} \sum_{z^{(i)}=1}^{K} q(z^{(i)}) \log q(z^{(i)}) = \ell(\boldsymbol{\theta})$$

- $\ell(\boldsymbol{\theta})$ is a lower bound of the original objective $L(\boldsymbol{\theta})$
- The equality holds when $\frac{p(\mathbf{x}^{(i)}, z^{(i)}; \boldsymbol{\theta})}{q(z^{(i)})}$ is constant
- This can be achieved for $q(z^{(i)}) = p(z^{(i)} | \mathbf{x}^{(i)}; \boldsymbol{\theta})$

EM Derivation

- The EM algorithm aims to optimize the lower bound $\ell(\boldsymbol{\theta})$

$$\boldsymbol{\theta}^\star = \arg\max_{\boldsymbol{\theta}} \ell(\boldsymbol{\theta}) = \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{M} \sum_{z^{(i)}=1}^{K} q(z^{(i)}) \log \frac{p(\mathbf{x}^{(i)}, z^{(i)}; \boldsymbol{\theta})}{q(z^{(i)})}$$

- EM repeatedly performs the following two steps until convergence. At t-th iteration,
  1. E-step: For each index i, compute

$$q^{(t)}(z^{(i)}) = p(z^{(i)} | \mathbf{x}^{(i)}; \boldsymbol{\theta}^{(t)})$$

  2. M-step: Compute

$$\boldsymbol{\theta}^{(t+1)} = \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{M} \sum_{z^{(i)}} q^{(t)}(z^{(i)}) \log p(\mathbf{x}^{(i)}, z^{(i)}; \boldsymbol{\theta})$$

EM Derivation

- In E-step, we do not fill in the unobserved $z^{(i)}$ with hard values, but find a posterior distribution $q(z^{(i)})$, given $\mathbf{x}^{(i)}$ and $\boldsymbol{\theta}^{(t)}$, i.e.,

$$q^{(t)}(z^{(i)}) = p(z^{(i)}|\mathbf{x}^{(i)};\boldsymbol{\theta}^{(t)})$$

- In M-step, we maximize the lower bound $\ell(\boldsymbol{\theta})$, while holding $q^{(t)}(z^{(i)})$ fixed, which is computed from the E-step
- M-step optimization can be done efficiently in most cases. For example, in GMM, we have closed-form solutions for all parameters

EM Convergence

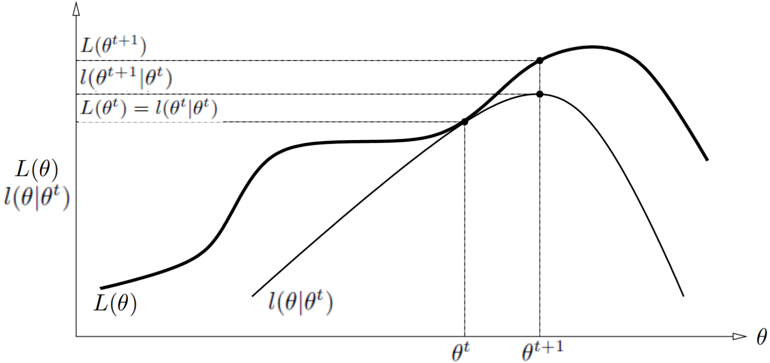
- Assuming $\boldsymbol{\theta}^{(t)}$ and $\boldsymbol{\theta}^{(t+1)}$ are the parameters from two successive iterations of EM, we have

$$
\begin{aligned}
L\left(\boldsymbol{\theta}^{(t)}\right) &\overset{(1)}{=} \sum_{i=1}^{M} \log p(\mathbf{x}^{(i)}; \boldsymbol{\theta}^{(t)}) \overset{(2)}{=} \sum_{i=1}^{M} \log \sum_{z^{(i)}=1}^{K} q(z^{(i)}) \frac{p(\mathbf{x}^{(i)}, z^{(i)}; \boldsymbol{\theta}^{(t)})}{q(z^{(i)})} \\
&\overset{(3)}{=} \sum_{i=1}^{M} \sum_{z^{(i)}=1}^{K} q^{(t)}(z^{(i)}) \log \frac{p(\mathbf{x}^{(i)}, z^{(i)}; \boldsymbol{\theta}^{(t)})}{q^{(t)}(z^{(i)})} \\
&\overset{(4)}{\leq} \sum_{i=1}^{M} \sum_{z^{(i)}=1}^{K} q^{(t)}(z^{(i)}) \log \frac{p(\mathbf{x}^{(i)}, z^{(i)}; \boldsymbol{\theta}^{(t+1)})}{q^{(t)}(z^{(i)})} \\
&\overset{(5)}{\leq} \sum_{i=1}^{M} \log \sum_{z^{(i)}=1}^{K} q^{(t)}(z^{(i)}) \frac{p(\mathbf{x}^{(i)}, z^{(i)}; \boldsymbol{\theta}^{(t+1)})}{q^{(t)}(z^{(i)})} \overset{(6)}{=} L\left(\boldsymbol{\theta}^{(t+1)}\right)
\end{aligned}
$$

EM Convergence

- where
  - (1): by definition - likelihood of the data
  - (2): by marginalization over $z^{(i)}$ and multiplication an arbitrary distribution $q(z^{(i)})$ to both numerator and denominator inside log
  - (3): by Jensen's inequality where equality condition satisfied by setting $q^{(t)}(z^{(i)}) = p(z^{(i)}|\mathbf{x}^{(i)}; \boldsymbol{\theta}^{(t)})$
  - (4): by M-step of EM, where we maximize (3), holding $q^{(t)}(z^{(i)})$ fixed
  - (5): by Jensen's inequality (in reverse order)
  - (6): by definition
- Hence, EM causes the likelihood to increase monotonically

EM Illustration for GMM



Remark

- If we define

$$J(q, \boldsymbol{\theta}) = \sum_{i=1}^{M} \sum_{z^{(i)}=1}^{K} q(z^{(i)}) \log \frac{p(\mathbf{x}^{(i)}, z^{(i)}; \boldsymbol{\theta})}{q(z^{(i)})},$$

  - EM can also be viewed as coordinate ascent on J, in which the E-step maximizes J w.r.t. q, and the M-step maximizes J with respect to $\boldsymbol{\theta}$

Clustering Summary

- **Clustering task**:
  - Given a set of input vectors $\mathcal{D} = \{\mathbf{x}^{(i)}\}_{i=1}^{M}$ with $\mathbf{x}^{(i)} \in \mathbb{R}^N$, group similar $\mathbf{x}^{(i)}$ together into clusters
    - Estimate a cluster center, representing the data points in that cluster
    - Predict the cluster for a new data point
- **Exhaustive clustering**
  - **Cluster shape**: arbitrary shape
  - **Principle**: minimize an assumed clustering criterion with brute-force search
  - **Pros**: optimal under the given clustering criterion
  - **Cons**: impractical to construct the clustering criterion; prohibitive to compute

Clustering Summary

- *K*-means
  - **Cluster shape**: circular
  - **Principle**: minimize distance to cluster center
  - **Pros**: simple and scalable (MiniBatchKMeans)
  - **Cons**: sensitive to initialization; could get bad solutions due to local minima; need to choose K
- **Gaussian mixture model (GMM)**
  - **Cluster shape**: elliptical
  - **Principle**: maximum likelihood using expectation maximization
  - **Pros**: elliptical cluster shapes
  - **Cons**: sensitive to initialization; could get bad solutions due to local minima; need to choose K

Other Things

- **Feature normalization**
  - Feature normalization is typically required clustering
  - E.g., algorithms based on Euclidean distance (K-means)