# Optimization

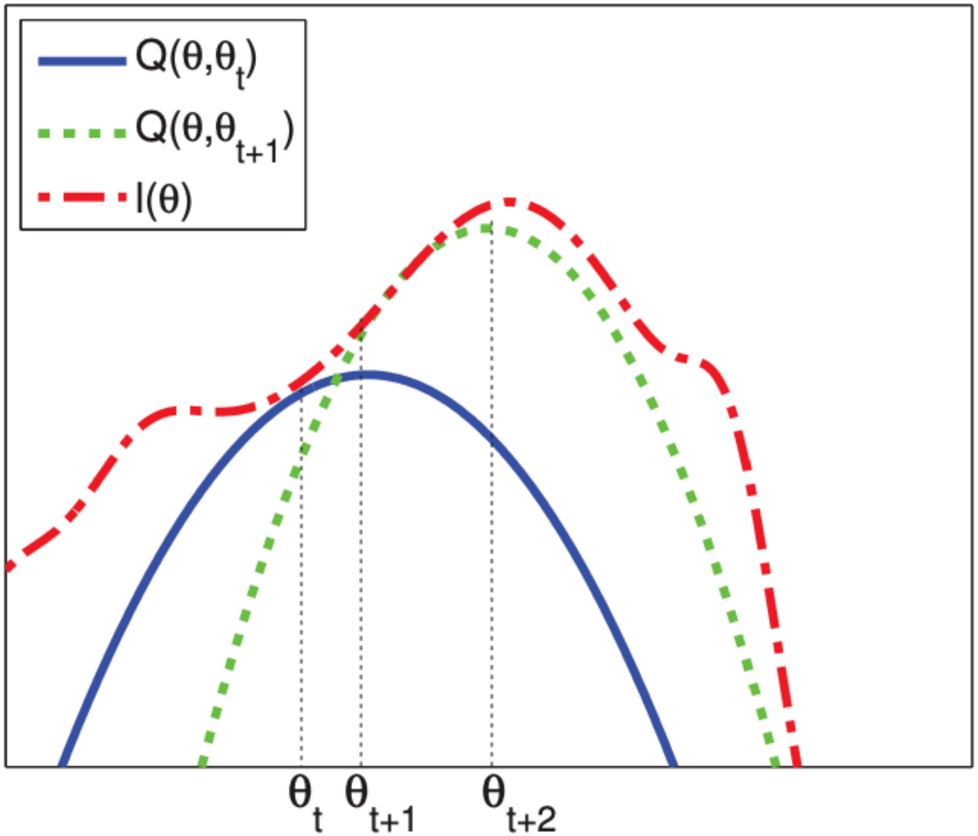
ECE57000: Artificial Intelligence, Fall 2019

David I. Inouye

# Announcements

- Office hours tomorrow (Thurs) moved to **1:30-2:30pm on Friday** (right after class)
  - Apologies for the late notice!
  - May add an extra hour next week if needed

# EM algorithm is **guaranteed** to increase *observed* likelihood, i.e., $\prod_i p_{mixture}(x_i)$



# Proof that it monotonically increases likelihood

- See 11.4.7 in [ML] for full derivation of proof
- Show that $Q(\theta;\ q^t)$ is lower bound observed likelihood $\ell(\theta)$, i.e., $\ell(\theta) \geq Q(\theta; q^t), \forall\theta$
- Choose $q^t(z_i) = p(z_i|x_i, \theta^t)$, which corresponds to $Q(\theta; \theta^t)$
- Show that lower bound is tight at $\theta_t$
- Combines three concepts
  1. Lower bound inequality (Jensen's inequality)
  2. Maximization inequality (M-step)
  3. Tightness of lower bound (E-step)

# Most AI/ML optimizations must be numerically estimated rather than closed-form

- EM algorithm
  - Powerful probabilistic algorithm for hidden/latent variables or missing data
  - Quite general alternating optimization algorithm
  - Can be slow and can get stuck in local minima
- Gradient descent
  - Stochastic gradient descent
  - Primary *current* algorithm for deep learning
  - Can handle very high dimensions
  - Only works under certain conditions
- (Later) Sampling-based optimization (MCMC/Gibbs)

# Vanilla gradient descent has very simple form

- Loss function denoted by $\mathcal{L}(\theta; \mathcal{D})$:

$$\arg\min_{\theta} \mathcal{L}(\theta; \mathcal{D})$$

1. Start at random parameter, e.g., $\theta^0 \sim \mathcal{N}(0, 1)$
2. Iteratively update parameter via **negative gradient** of loss function ($\eta_t$ is step size or

$$\theta^{t+1} = \theta^t - \eta_t \nabla_\theta \mathcal{L}(\theta^t)$$

- $\eta_t$ is **learning rate** (or **step size**)

# Stochastic gradient descent (SGD) merely uses one sample in the gradient calculation

- The loss function can usually be split into a summation of losses $\ell(\theta; x_i)$ for each sample $x_i$:
$\mathcal{L}(\theta; \mathcal{D}) = \frac{1}{n} \sum_{i=1}^{n} \ell(\theta; x_i)$
- SGD approximates the full gradient by the gradient of a *single* sample
  - $\nabla_\theta \mathcal{L}(\theta^t; \mathcal{D}) \approx \nabla_\theta \ell(\theta^t; x_i)$
  - Theoretically, $\mathbb{E}_i[\nabla_\theta \ell(\theta^t; x_i)] = \nabla_\theta \mathcal{L}(\theta^t; \mathcal{D})$
- Loop through all $x_i \in \mathcal{D}$

$$\theta^{t+1} = \theta^t - \eta_t \nabla_\theta \ell(\theta^t; x_i)$$

- One pass through dataset
  - GD: 1 large update with $O(n)$ cost
  - SGD: $n$ smaller updates with $O(1)$ cost each

# Mini-batch SGD (or just SGD) uses a small batch of samples in the gradient calculation

- **Mini-batch SGD** approximates the full gradient by the gradient of a batch of samples
  - Sample mini-batch

$$\theta^{t+1} = \theta^{t} - \eta_t \sum_{k=1}^{b} \frac{1}{b} \nabla_\theta \ell(\theta^{t}; x_k)$$

- One pass through dataset
  - GD: 1 large update
  - SGD: $n$ smaller updates
  - Mini-batch SGD: $\frac{n}{b}$ medium-size updates

# <u>Learning rate / step size</u> is critical for convergence and correctness of algorithm

- If learning rate is **too high**, the algorithm could diverge.
  - Diverge means to actually get farther away from the solution.
- If learning rate **too low**, the algorithm could take a very long time to converge.
- Adaptive learning rates may help **<u>(but not always)</u>**
  - Decreasing step size, $\eta_t = \frac{1}{t}$
  - ADAM – Adaptive Moment Estimation

# Parameter initialization can be important if non-convex or step size incorrect

- If convex function, initial parameter $\theta^0$ **will not** affect final optimization result $\hat{\theta} = \operatorname{argmin}_{\theta} \mathcal{L}(\theta)$.
  - Yay!
  - (Assuming appropriate step size.)

- If *non-convex*, starting position **WILL** affect final converged $\hat{\theta}$.
  - Sad day.
  - But sometimes it's not too bad in practice.

# Demo using PyTorch to automatically compute gradients

- Nice introductory PyTorch tutorial
  - https://towardsdatascience.com/understanding-pytorch-with-an-example-a-step-by-step-tutorial-81fc5f8c4e8e