Latent variable models

Dmitry Vetrov

Research professor at HSE

Senior researcher at Yandex

Head of Bayesian methods research group

http://bayesgroup.ru

Deep|Bayes

Outline

- Mixtures of distributions
- EM-algorithm
- Discrete and continuous latent variables
- Case study: AdaGram

Latent variable modeling: example

- Consider the following problem
- We have a set of points generated from a Gaussian

$$x_i \sim \mathcal{N}(x_i|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

- We need to estimate its parameters μ and σ

Latent variable modeling: example

- Consider the following problem
- We have a set of points generated from a Gaussian

$$x_i \sim \mathcal{N}(x_i|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

- We need to estimate its parameters μ and σ

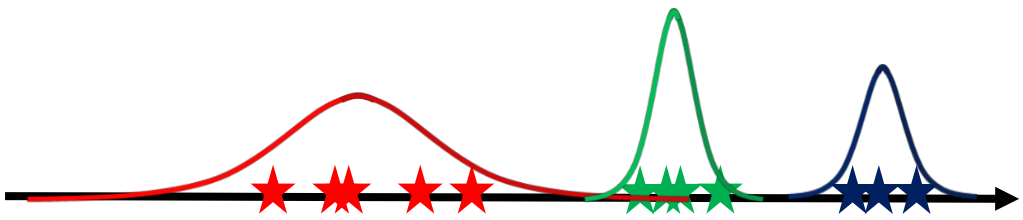
- Solution is simple: we estimate sample mean and variance

Latent variable modeling: example

- Now suppose we're given several sets of points from different gaussians
- We need to estimate the parameters of those gaussians and their weights

Latent variable modeling: example

- Now suppose we're given several sets of points from different gaussians
- We need to estimate the parameters of those gaussians and their weights



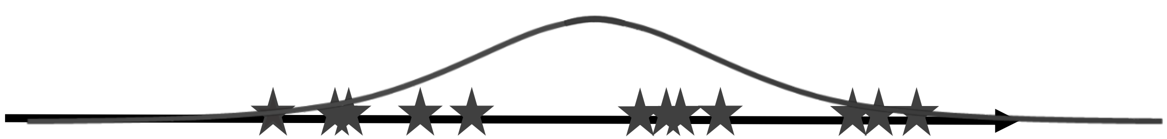
- The problem is as easy if we know what objects were generated from each gaussian

Latent variable modeling: example

- Now what if we do not know what objects were generated by each gaussian
- Of course we could still try to use a single gaussian model…

Latent variable modeling: example

- Now what if we do not know what objects were generated by each gaussian
- Of course we could still try to use a single gaussian model…
- … but there is a better way: latent variable model!



Mixture of gaussians

- For each object x_i we establish additional latent variable z_i which denotes the index of gaussian from which i-th object was generated
- Then our model is

$$p(X,Z|\theta) = \prod_{i=1}^{n} p(x_i, z_i|\theta) = \{\text{Product rule}\} = \prod_{i=1}^{n} p(x_i|z_i,\theta)p(z_i|\theta) =$$

$$\prod_{i=1}^{n} \pi_{z_i}\mathcal{N}(x_i|\mu_{z_i},\sigma_{z_i}^2)$$

- Here $\pi_j = p(z_i = j)$ are prior probability of j-th gaussian and $\theta = \{\mu_j,\sigma_j,\pi_j\}_{j=1}^K$ are the parameters to be estimated
- If we know both X and Z we obtain explicit ML-solution:

$$\theta_{ML} = \arg\max_\theta p(X,Z|\theta) = \arg\max_\theta \log p(X,Z|\theta)$$

Mixture of gaussians

- What if we do not know Z? Then we need to maximize w.r.t. θ the log of incomplete likelihood

$$\log p(X|\theta)$$

Mixture of gaussians

- What if we do not know Z? Then we need to maximize w.r.t. θ the log of incomplete likelihood

$$\log p(X|\theta) = \int q(Z) \log p(X|\theta) dZ$$

Mixture of gaussians

- What if we do not know Z? Then we need to maximize w.r.t. θ the log of incomplete likelihood

$$\log p(X|\theta) \quad = \quad \int q(Z) \log p(X|\theta) dZ \quad = \quad \int q(Z) \log \frac{p(X, Z|\theta)}{p(Z|X, \theta)} dZ$$

Mixture of gaussians

- What if we do not know Z? Then we need to maximize w.r.t. θ the log of incomplete likelihood

$$\log p(X|\theta) = \int q(Z) \log p(X|\theta) dZ = \int q(Z) \log \frac{p(X, Z|\theta)}{p(Z|X, \theta)} dZ =$$

$$\int q(Z) \log \frac{q(Z) p(X, Z|\theta)}{q(Z) p(Z|X, \theta)} dZ$$

Mixture of gaussians

- What if we do not know Z? Then we need to maximize w.r.t. θ the log of incomplete likelihood

$$\log p(X|\theta) = \int q(Z) \log p(X|\theta) dZ = \int q(Z) \log \frac{p(X,Z|\theta)}{p(Z|X,\theta)} dZ =$$

$$\int q(Z) \log \frac{q(Z)p(X,Z|\theta)}{q(Z)p(Z|X,\theta)} dZ =$$

$$\int q(Z) \log \frac{p(X,Z|\theta)}{q(Z)} dZ + \int q(Z) \log \frac{q(Z)}{p(Z|X,\theta)} dZ$$

Mixture of gaussians

- What if we do not know Z? Then we need to maximize w.r.t. θ the log of incomplete likelihood

$$\log p(X|\theta) = \int q(Z) \log p(X|\theta) dZ = \int q(Z) \log \frac{p(X,Z|\theta)}{p(Z|X,\theta)} dZ =$$

$$\int q(Z) \log \frac{q(Z)p(X,Z|\theta)}{q(Z)p(Z|X,\theta)} dZ =$$

Always non-negative!

$$\boxed{\int q(Z) \log \frac{p(X,Z|\theta)}{q(Z)} dZ} + \boxed{\int q(Z) \log \frac{q(Z)}{p(Z|X,\theta)} dZ}$$

Variational lower bound

Mixture of gaussians

- What if we do not know Z? Then we need to maximize w.r.t. θ the log of incomplete likelihood

$$\log p(X|\theta) = \int q(Z) \log p(X|\theta) dZ = \int q(Z) \log \frac{p(X,Z|\theta)}{p(Z|X,\theta)} dZ =$$

$$\int q(Z) \log \frac{q(Z)p(X,Z|\theta)}{q(Z)p(Z|X,\theta)} dZ =$$

Always non-negative!

$$\boxed{\int q(Z) \log \frac{p(X,Z|\theta)}{q(Z)} dZ} + \boxed{\int q(Z) \log \frac{q(Z)}{p(Z|X,\theta)} dZ} =$$

Variational lower bound

$$\mathcal{L}(q,\theta) + KL(q||p) \geq \mathcal{L}(q,\theta)$$

Mixture of gaussians

- What if we do not know Z? Then we need to maximize w.r.t. θ the log of incomplete likelihood

$$\log p(X|\theta) = \int q(Z) \log p(X|\theta) dZ = \int q(Z) \log \frac{p(X,Z|\theta)}{p(Z|X,\theta)} dZ =$$

$$\int q(Z) \log \frac{q(Z)p(X,Z|\theta)}{q(Z)p(Z|X,\theta)} dZ =$$

Always non-negative!

$$\boxed{\int q(Z) \log \frac{p(X,Z|\theta)}{q(Z)} dZ} + \boxed{\int q(Z) \log \frac{q(Z)}{p(Z|X,\theta)} dZ} =$$

Variational lower bound (ELBO)

$$\mathcal{L}(q,\theta) + KL(q||p) \geq \mathcal{L}(q,\theta)$$

- Instead of optimizing $\log p(X|\theta)$ we optimize variational lower bound $\mathcal{L}(q,\theta)$ w.r.t. both θ and $q(Z)$

- The block-coordinate algorithm is known as EM-algorithm

Variational lower bound

Definition. Function $g(\xi, x)$ is called variational lower bound for function $f(x)$ iff

- For all ξ for all x it follows $f(x) \geq g(\xi, x)$
- For any x_0 there exists $\xi(x_0)$ such that $f(x_0) = g(\xi(x_0), x_0)$

If we managed to find such variational lower bound then instead of solving

$$f(x) \to \max_x$$

we may iteratively perform block-coordinate updates of $g(\xi, x)$

$$x_n = \arg\max_x g(\xi_{n-1}, x), \quad \xi_n = \xi(x_n)$$

EM algorithm

- To solve

$$\mathcal{L}(q,\theta) = \int q(Z)\log\frac{p(X,Z|\theta)}{q(Z)}dZ \to \max_{q,\theta}$$

we start from initial point θ_0 and iteratively repeat

- E-step: find

$$q(Z) = \arg\max_q \mathcal{L}(q,\theta_0) = \arg\min_q KL(q||p) = p(Z|X,\theta_0)$$

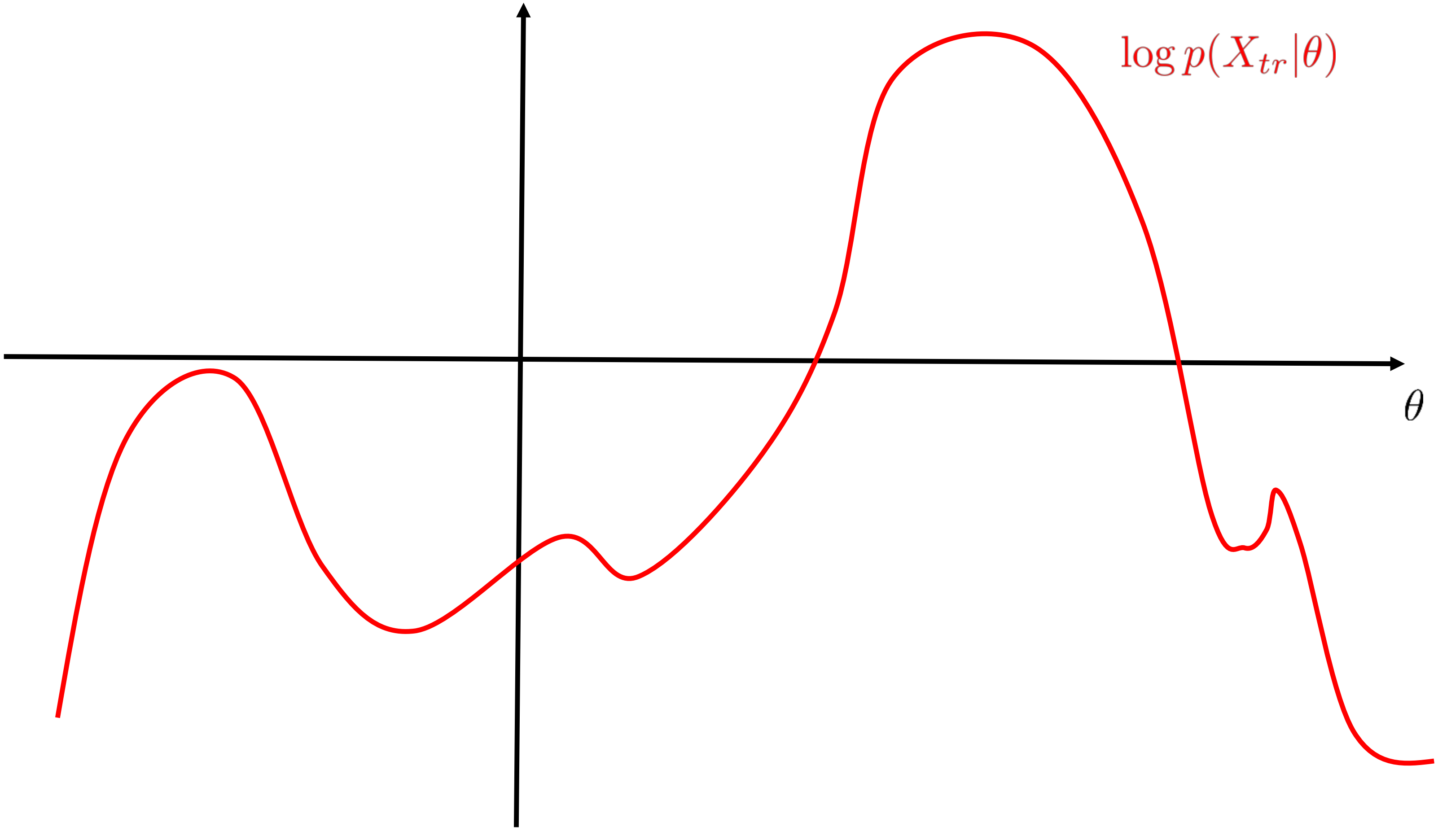
- M-step: solve

$$\theta_* = \arg\max_\theta \mathcal{L}(q,\theta) = \arg\max_\theta \mathbb{E}\log p(X,Z|\theta),$$

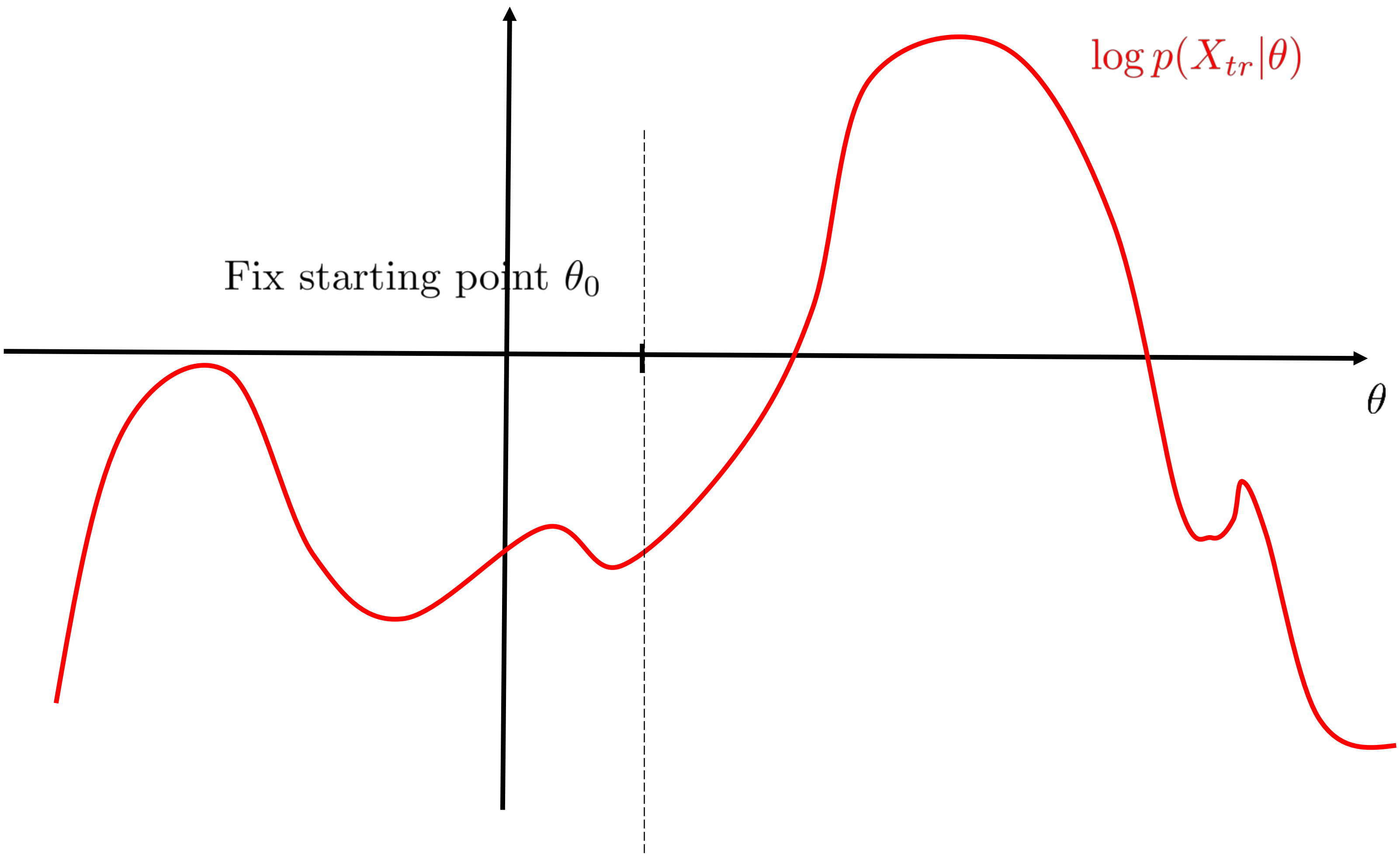
set $\theta_0 = \theta_*$ and go to E-step until convergence

- The EM algorithm monotonically increases the lower bound and converges to stationary point of $\log p(X|\theta)$

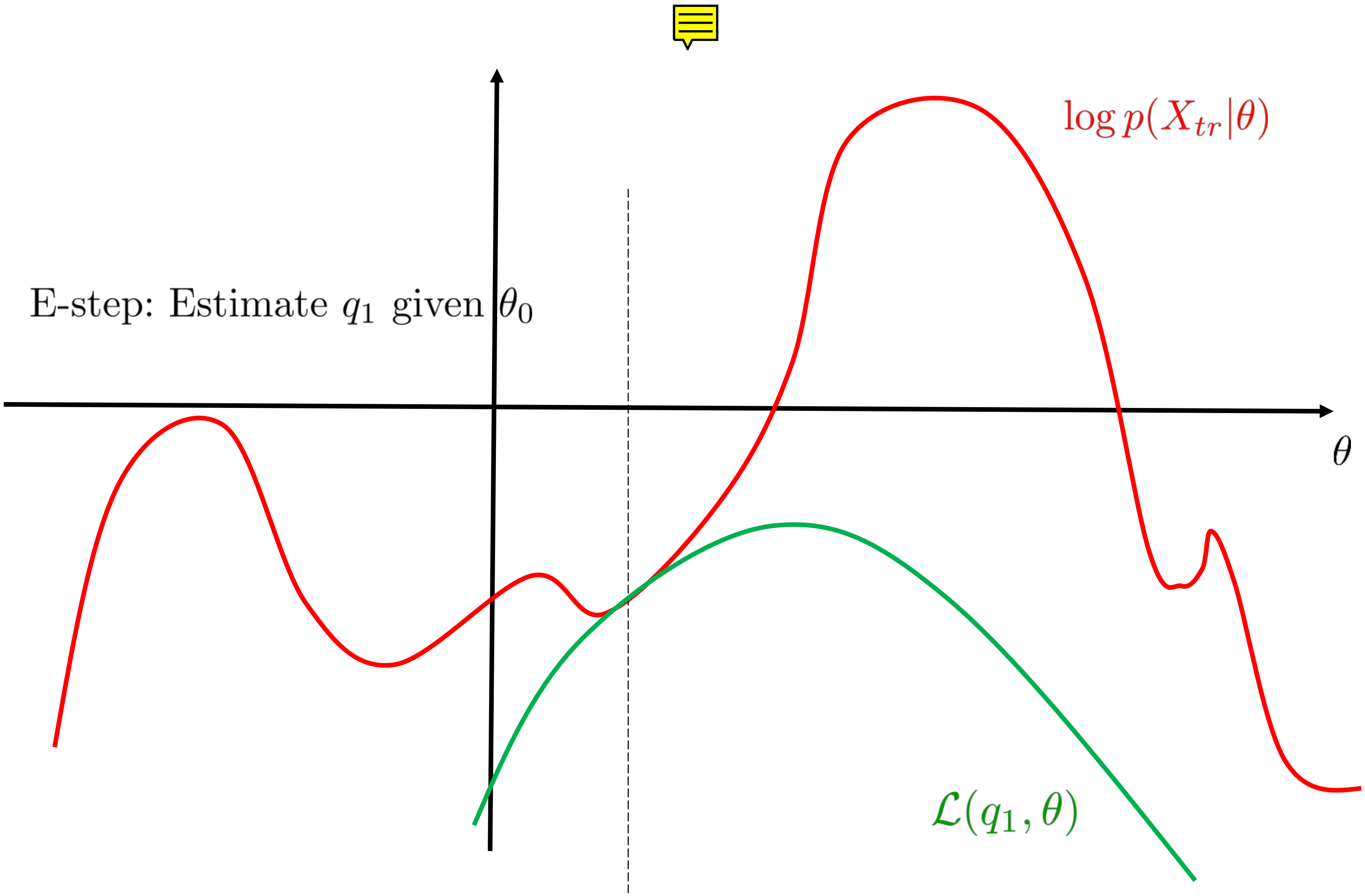
EM-algorithm



EM-algorithm

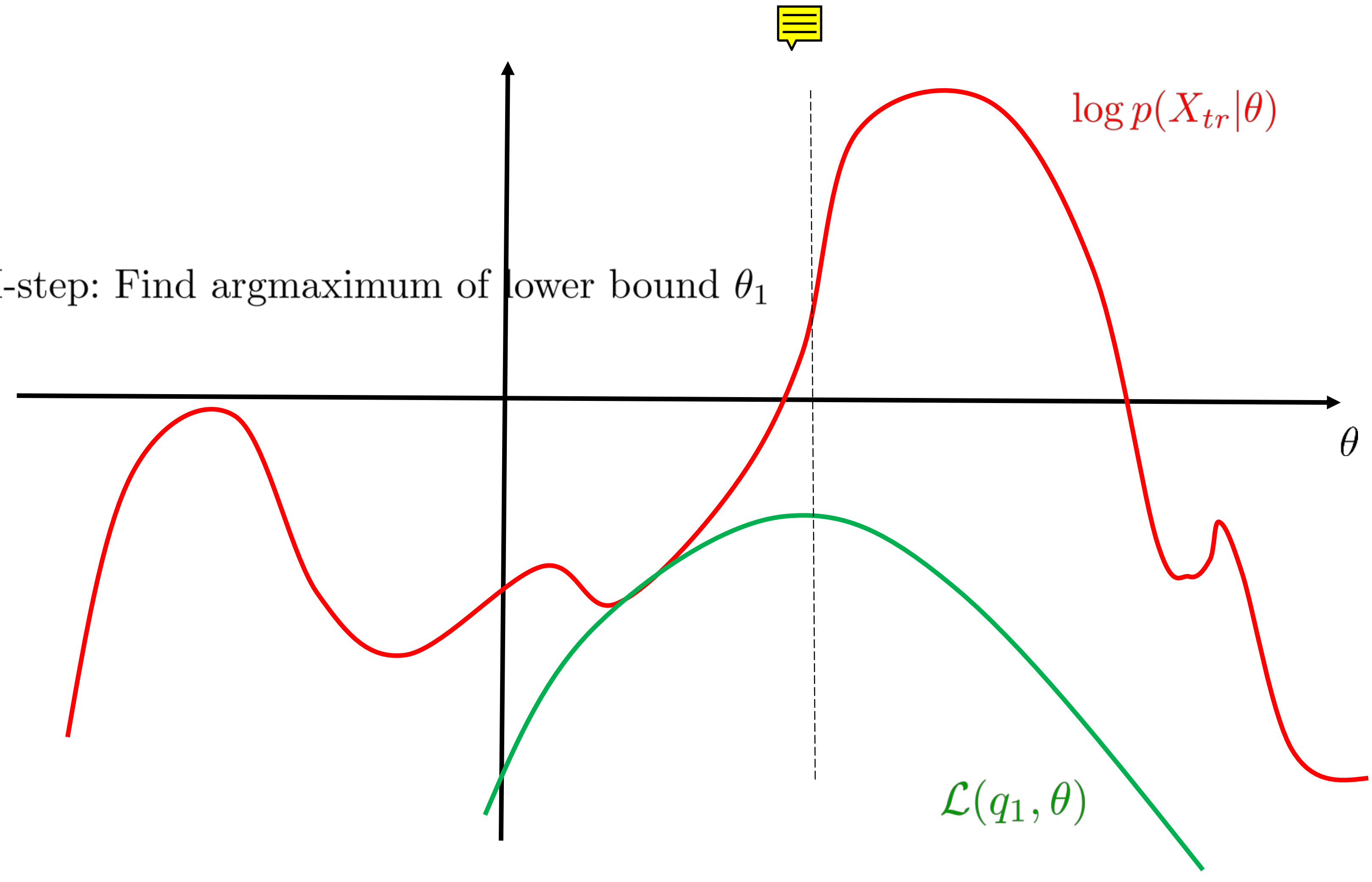


EM-algorithm



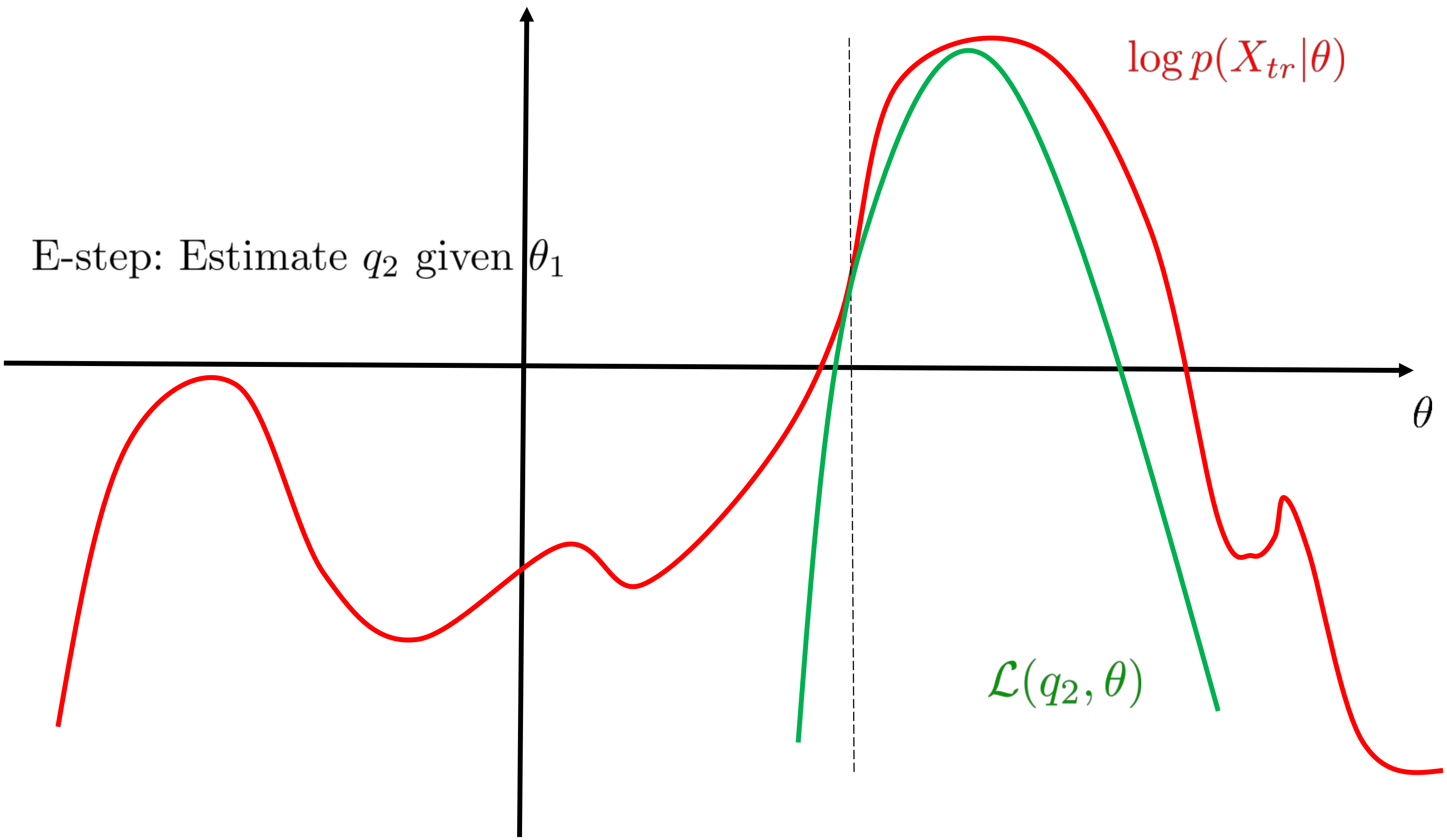
EM-algorithm

M-step: Find argmaximum of lower bound θ_1



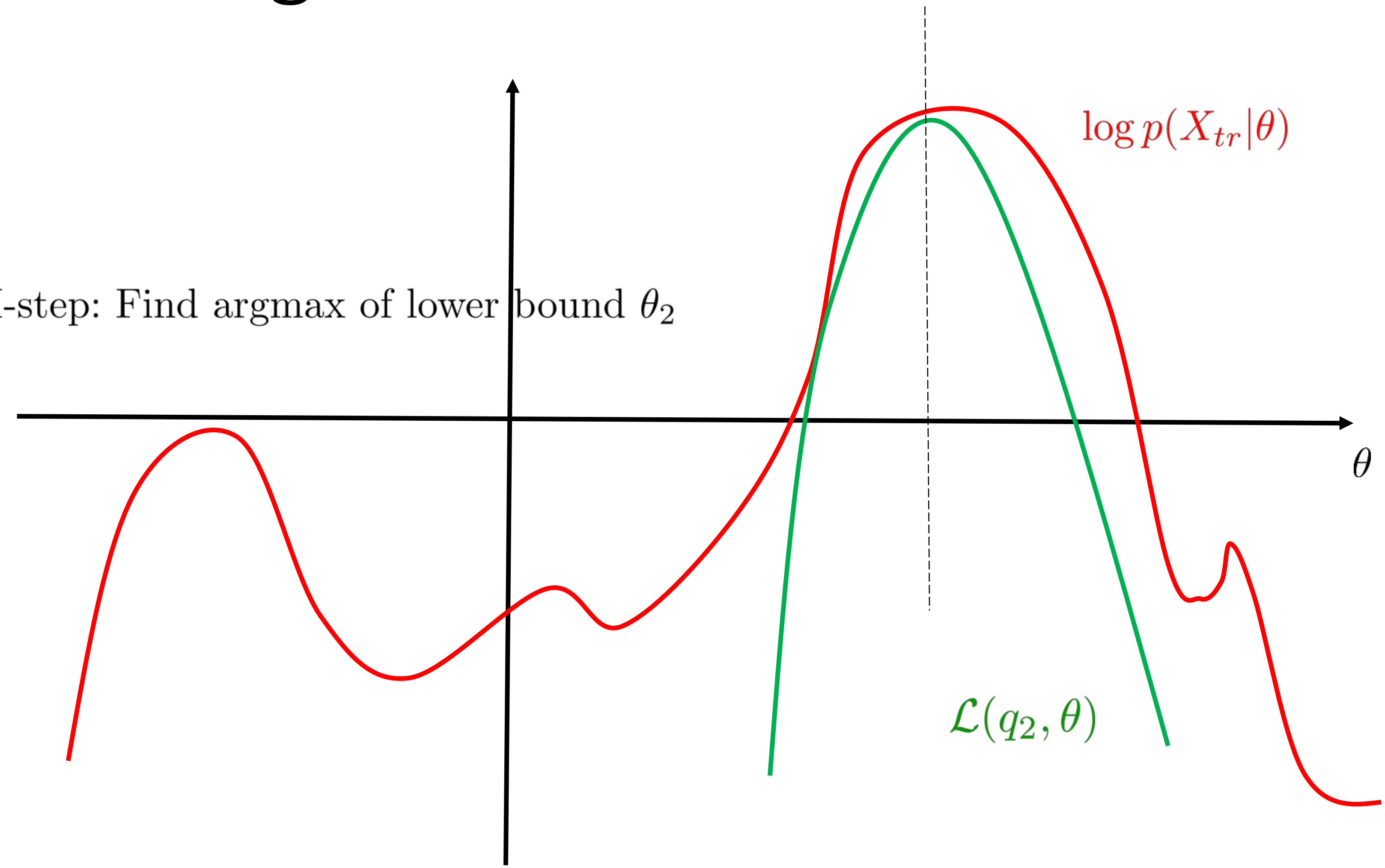
EM-algorithm

E-step: Estimate q_2 given θ_1



EM-algorithm

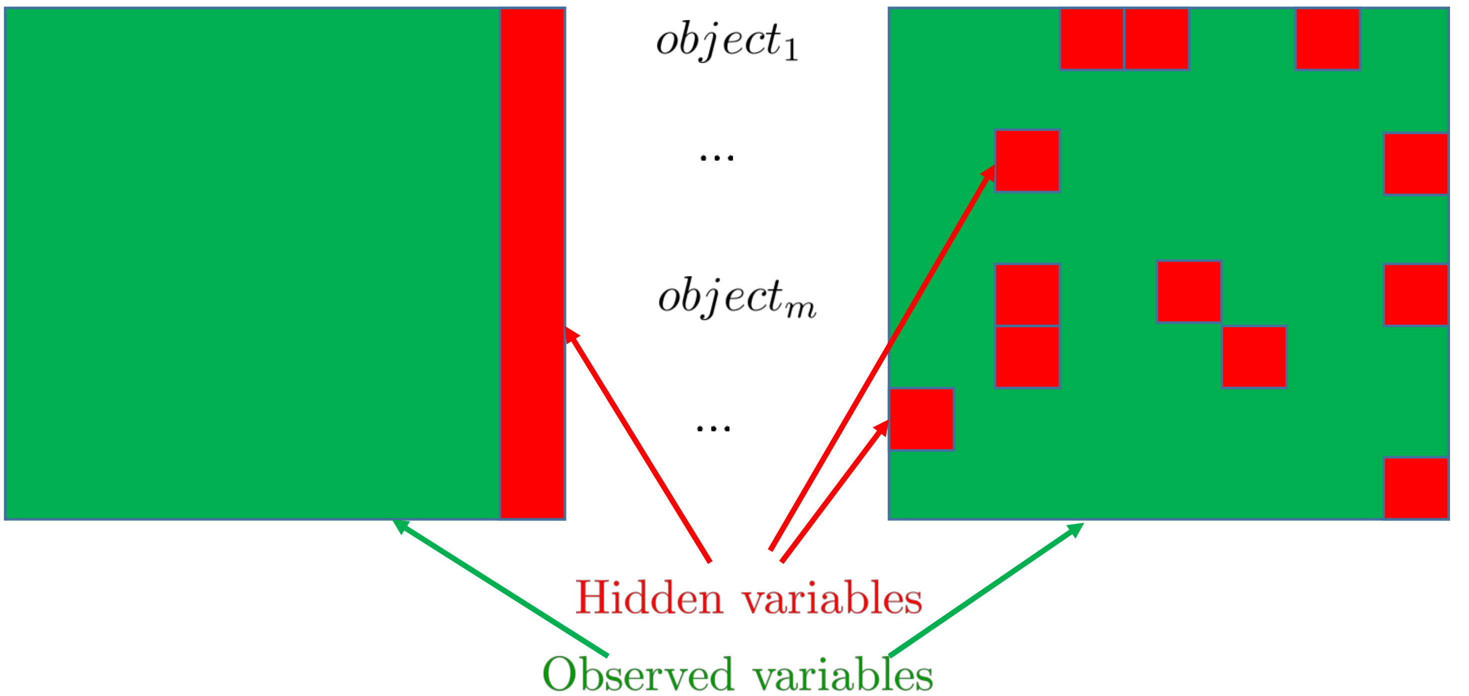
M-step: Find argmax of lower bound θ_2



Benefits of EM algorithm

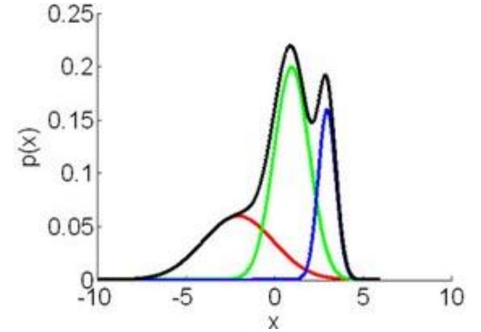
- In many cases (e.g. for the mixture of gaussians) E-step and M-steps can be performed in closed form
- Allows to build more complicated models of data using mixtures of simple distributions
- If true posterior $p(Z|X,\theta)$ is intractable we may search for the closest $q(Z)$ among tractable distributions by solving optimization problem
- Allows to process missing data by treating them as latent variables

General nature of EM-framework



- EM algorithm allows to fill in arbitrary gaps in data
- May deal with both discrete and continuous variables
- Always converges
- Allows multiple extensions

Categorical latent variables



- Assume all $z_i \in \{1, \ldots, K\}$ then marginal

$$p(x_i|\theta) = \sum_{k=1}^{K} p(x_i|k, \theta) p(z_i = k|\theta)$$

  is simply a finite mixture of distributions

- If K is not exponentially large E-step can be performed in closed form

$$q(z_i = k) = p(z_i = k|x_i, \theta) = \frac{p(x_i|k, \theta) p(z_i = k|\theta)}{\sum_{l=1}^{K} p(x_i|l, \theta) p(z_i = l|\theta)}$$

- M-step is simply a sum of finite terms

$$\mathbb{E}_Z \log p(X, Z|\theta) = \sum_{i=1}^{n} \mathbb{E}_{z_i} \log p(x_i, z_i|\theta) = \sum_{i=1}^{n} \sum_{k=1}^{K} q(z_i = k) \log p(x_i, k|\theta)$$

Continuous latent variables

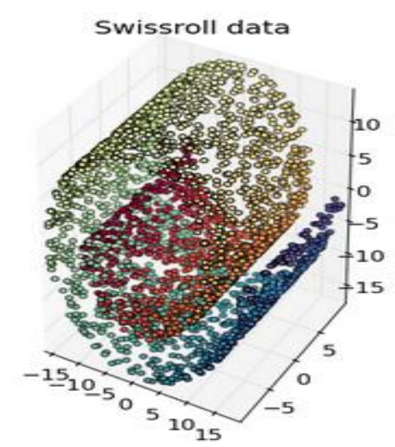
- Continuous varuables can be regarded as a mixture of a continuum of distributions

$$p(x_i|\theta) = \int p(x_i, z_i|\theta) dz_i = \int p(x_i|z_i, \theta) p(z_i|\theta) dz_i$$

- E-step can be done in closed form only in case of **conjugate distributions**, otherwise the true posterior is intractable

$$q(z_i) = p(z_i|x_i, \theta) = \frac{p(x_i|z_i, \theta) p(z_i|\theta)}{\int p(x_i|z_i, \theta) p(z_i|\theta) dz_i}$$

- Typically continuous latent variables are used for dimension reduction also known as **representation learning**



Difficult cases

- Each object has multi-dimensional discrete latent variable $\Rightarrow$ exponentially large sums
- Object has both discrete and continuous latent variables (e.g. mixture of low-dimensional manifolds) $\Rightarrow$ mixed discrete-continuous distributions over latent variables
- Continuous latent variables come from non-conjugate priors $\Rightarrow$ intractable multi-dimensional intergrals

Difficult cases

- Each object has multi-dimensional discrete latent variable $\Rightarrow$ exponentially large sums
- Object has both discrete and continuous latent variables (e.g. mixture of low-dimensional magnifolds) $\Rightarrow$ mixed discrete-continuous distributions over latent variables
- Continuous latent variables come from non-conjugate priors $\Rightarrow$ intractable multi-dimensional intergrals

Way out: Variational Bayes

Variational Bayes

- Recall that

$$p(Z|X,\theta) = \arg\max_q \mathcal{L}(q,\theta) = \arg\min_q KL(q(Z)||p(Z|X,\theta)),$$

  where extremum is taken with respect to **all possible distributions** $q(Z)$

- What if we limit ourselves with more restricted set of distributions?..

Variational Bayes

- Recall that

$$p(Z|X,\theta) = \arg\max_q \mathcal{L}(q,\theta) = \arg\min_q KL(q(Z)||p(Z|X,\theta)),$$

where extremum is taken with respect to **all possible distributions** $q(Z)$

- What if we limit ourselves with more restricted set of distributions?..

Key idea:

Inference becomes optimization

Variational Bayes

- We approximate posterior $p(Z|X,\theta)$ by optimizing

$$\mathcal{L}(q,\theta) = \int q(Z|\phi) \log \frac{p(X,Z|\theta)}{q(Z|\phi)} dZ \to \max_{\phi}$$

  in some restricted set of distributions $q(Z|\phi)$

- Note that all elements inside intergral are computable
- Maximization of $\mathcal{L}(q,\theta)$ w.r.t. $q(Z|\phi)$ corresponds to minimization of KL-divergence between $q(Z|\phi)$ and $p(Z|X,\theta)$
- We have converted inference problem to optimization problem

$$\mathcal{L}(q,\theta) = \mathcal{L}(\phi,\theta) \to \max_{\phi}$$

  known as **variational E-step**

Block-coordinate optimization

- According to classical EM-algorithm we need to iteratively solve

$$\phi_n = \arg\max_{\phi} \mathcal{L}(\phi, \theta_{n-1}), \quad \text{E-step}$$

$$\theta_n = \arg\max_{\theta} \mathcal{L}(\phi_n, \theta), \quad \text{M-step}$$

- If both any of the problems cannot be solved analytically then we may simply make one step towards each gradient

$$\phi_n = \phi_{n-1} + \eta \nabla_{\phi} \mathcal{L}(\phi, \theta_{n-1})$$

$$\theta_n = \theta_{n-1} + \epsilon \nabla_{\theta} \mathcal{L}(\phi_n, \theta)$$

- Note that we no longer need to compute $\mathcal{L}(\phi, \theta)$ but only its {stochastic} gradients

Nice properties of ELBO

$$\mathcal{L}(\phi, \theta) = \int q(Z|\phi) \log \frac{p(X|Z,\theta)p(Z|\theta)}{q(Z|\phi)} dZ$$

- ELBO itself is often intractable but all we need is to know the optimal values of (ϕ, θ) **rather than the value of ELBO itself**

- We may compute stochastic gradient by removing true expectation with its Monte Carlo estimate (next lecture)

Nice properties of ELBO

$$\mathcal{L}(\phi, \theta) = \int q(Z|\phi) \log \frac{p(X|Z,\theta)p(Z|\theta)}{q(Z|\phi)} dZ$$

- ELBO itself is often intractable but all we need is to know the optimal values of (ϕ, θ) **rather than the value of ELBO itself**
- We may compute stochastic gradient by removing true expectation with its Monte Carlo estimate (next lecture)
- If data is i.i.d. then

$$\log p(X|Z,\theta)p(Z|\theta) = \sum_{i=1}^{n} (\log p(x_i|z_i,\theta) + \log p(z_i|\theta))$$

Nice properties of ELBO

$$\mathcal{L}(\phi, \theta) = \int q(Z|\phi) \log \frac{p(X|Z, \theta)p(Z|\theta)}{q(Z|\phi)} dZ$$

- ELBO itself is often intractable but all we need is to know the optimal values of (ϕ, θ) **rather than the value of ELBO itself**

- We may compute stochastic gradient by removing true expectation with its Monte Carlo estimate (next lecture)

- If data is i.i.d. then

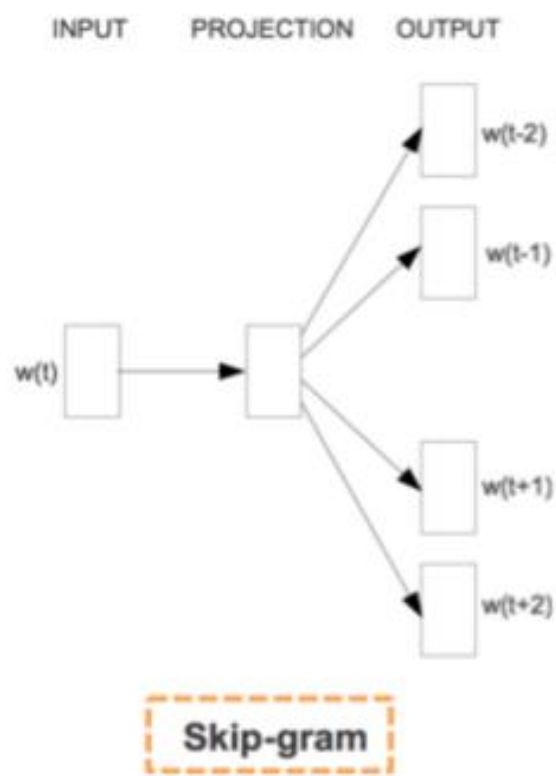
$$\log p(X|Z, \theta)p(Z|\theta) = \sum_{i=1}^{n} (\log p(x_i|z_i, \theta) + \log p(z_i|\theta))$$

- But then we may use **mini-batching** for computing stochastic gradients in case of large datasets

- This makes ELBO optimization highly scalable!

Word2vec model (Mikolov2013)

- Designed for word prediction according to its context
- Transforms words to points in 255-dimensional vector space



Mathematical formulation

… w(t-C) … w(t) … w(t+C) …

X	Y
w(t)	w(t-C)
w(t)	w(t-C+1)
w(t)	…
w(t)	w(t+C-1)
w(t)	w(t+C)
w(t+1)	w(t+1-C)
…	…

$$p(y|x,\theta) = \frac{\exp\left(In(x)^T Out(y)\right)}{\sum_{y'} \exp\left(In(x)^T Out(y')\right)},$$

$$p(Y|X,\theta) \to \max_{\theta}$$

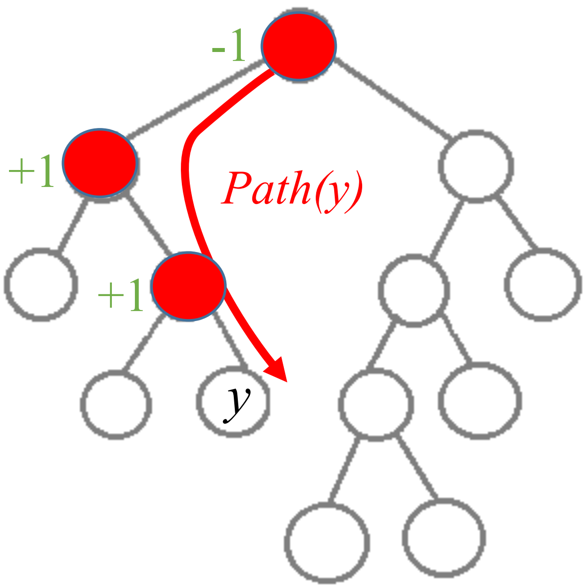
$$\text{where } \theta = \{In, Out\}$$

This is how it should work in ideal case. The problem is with denominator which ensures normalization. It requires $O(V)$ to compute it for each x

Hierarchical soft-max

- Let us construct binary Huffman tree for our dictionary
- Each word y to be predicted corresponds to a leaf in the tree
- Denote $Path(y)$ the sequence of internal nodes from root to leaf y
- Denote $d_{c,y}$ the direction of further path from c to y:

$$d_{c,y} = \begin{cases} +1 & y \text{ is in right subtree} \\ -1 & y \text{ is in left subtree} \end{cases}$$



- Then
$$p(y|x,\theta) = \prod_{c \in Path(y)} \sigma\left(d_{c,y} In(x)^T Out(c)\right),$$
where $\sigma(x) = \frac{1}{1+\exp(-x)}$
- Reduce complexity from $O(V)$ to $O(\log V)$

Semantic properties of representations

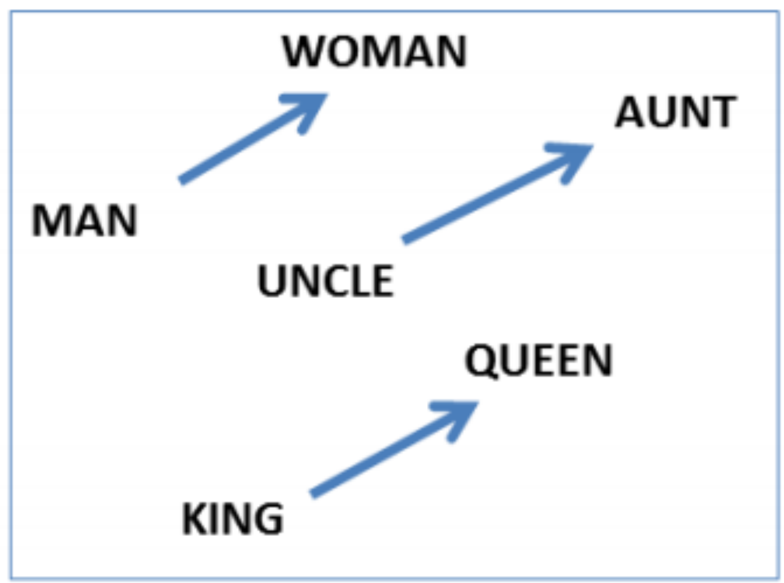
- Most known property of word2vec model: algebraic operations on vectors correspond to semantic operations on senses:

$$In(\texttt{'Paris'}) - In(\texttt{'France'}) + In(\texttt{'Russia'}) \approx In(\texttt{'Moscow'})$$

  Thousands of examples!

- Word2vec seems to capture notions of gender, geogaphy, number, and many other attributes

- Can it be useful for Q&A models?



Word ambiguity

- Suppose we want to answer the question
  `When was the Battle of Waterloo?`
- Well... It depends on whether the following holds true:

$$In(\texttt{'Waterloo'}) - In(\texttt{'Battle'}) + In(\texttt{'Date'}) \approx In(\texttt{'1815'})$$

- Even if we succeed we will not be able to answer any questions about the song or the railway station

$$In(\texttt{'Waterloo'}) = ?$$

$In(\texttt{'Euston'})$
$In(\texttt{'London'})$
$In(\texttt{'Station'})$

$In(\texttt{'Napoleon'})$
$In(\texttt{'Austerlitz'})$
$In(\texttt{'Battle'})$

$In(\texttt{'ABBA'})$
$In(\texttt{'Mamma mia'})$
$In(\texttt{'Voulez-vous'})$

Word2vec summary

Pros

- Learns untrivial and abstract concepts
- Extremely computationally effective (less than an hour of training on the whole Wikipedia using SGD)
- Usable not only for the words (sentences, abstracts, graphs, etc.)

Cons

- Unique representation for each word regardless of the meaning of the particular word occurence
- Dependant on the choice of a tree in hierarchical soft-max

Multi-sense extension of skip-gram

- For simplicity assume we know the number of meanings for each word
- Define the latent variable z_i that indicates meaning of particular word occurence x_i
- Let us search for vector representations of meanings rather than words $In(x_i, z_i)$
- Now it is easy to define the probability of y_i given the context word and its meaning:

$$p(y_i|x_i, z_i, \theta) = \prod_{c \in Path(y_i)} \sigma\left(d_{c,y_i} In(x_i, z_i)^T Out(c)\right),$$

where $\sigma(x) = \frac{1}{1+\exp(-x)}$

Multi-sense extension of skip-gram

- We have defined $p(y_i|x_i, z_i, \theta)$. To finish model we need to set $p(z_i|x_i)$ that is prior probability of particular meaning for a given word

- In case of absense of any knowledge we may just set it to uniform distribution

$$p(z_i = k|x_i) = \frac{1}{K(x_i)},$$

  where $K(x_i)$ is total number of meanings for word x_i

- Now we have complete discriminative model

$$p(y_i, z_i|x_i, \theta) = p(y_i|x_i, z_i, \theta)p(z_i|x_i)$$

- If we knew z_i this would be just standard skip-gram model with additional context words

- Since we do not know it we can now use EM-algorithm that will both estimate our parameters $\{In(x, z), Out(c)\}$ and the probabilities of meanings of x_i given its neighbour: $p(z_i|x_i, y_i, \theta)$

Naïve EM algorithm

- E-step: For each training object estimate the distribution on latent variable

$$p(z_i = k|x_i, y_i, \theta) = \frac{p(y_i|x_i, k, \theta)p(z_i = k|x_i)}{\sum_{l=1}^{K(x_i)} p(y_i|x_i, l, \theta)p(z_i = l|x_i)}$$

We can do this in explicit manner assuming the number of meanings is reasonably small

```
Our train arrived to Waterloo at 2pm
```

$$\texttt{Waterloo - ?} \quad \begin{cases} \texttt{Station} & \texttt{0.76} \\ \texttt{Battle} & \texttt{0.21} \\ \texttt{Song} & \texttt{0.03} \end{cases}$$

Naïve EM algorithm

- E-step: For each training object estimate the distribution on latent variable

$$p(z_i = k|x_i, y_i, \theta) = \frac{p(y_i|x_i, k, \theta)p(z_i = k|x_i)}{\sum_{l=1}^{K(x_i)} p(y_i|x_i, l, \theta)p(z_i = l|x_i)}$$

  We can do this in explicit manner assuming the number of meanings is reasonably small

- M-step: Optimize w.r.t. $\theta = \{In(x, z), Out(c)\}$

$$\mathbb{E} \log p(Y|Z, X, \theta)p(Z|X) \to \max_\theta$$

  Equivalent to training standard skip-gram with increased number of context words

- Seems computationally efficient?..

Naïve EM algorithm

- E-step: For each training object estimate the distribution on latent variable

$$p(z_i = k|x_i, y_i, \theta) = \frac{p(y_i|x_i, k, \theta)p(z_i = k|x_i)}{\sum_{l=1}^{K(x_i)} p(y_i|x_i, l, \theta)p(z_i = l|x_i)}$$

We can do this in explicit manner assuming the number of meanings is reasonably small

- M-step: Optimize w.r.t. $\theta = \{In(x, z), Out(c)\}$

$$\mathbb{E} \log p(Y|Z, X, \theta)p(Z|X) \to \max_\theta$$

Equivalent to training standard skip-gram with increased number of context words

- Seems computationally efficient?.. NO!

- We'll need to recompute $p(z|x, y, \theta)$ for **each** object (In Wikipedia2012 there is about 10^9 of words) to make just **single** iteration of EM

Large scale EM

- Remember our scheme
- E-step: For each training object estimate the distribution on latent variable

$$p(z_i = k|x_i, y_i, \theta) = \frac{p(y_i|x_i, k, \theta)p(z_i = k|x_i)}{\sum_{l=1}^{K(x_i)} p(y_i|x_i, l, \theta)p(z_i = l|x_i)}$$

  We can do this in explicit manner assuming the number of meanings is reasonably small

- M-step: Optimize w.r.t. $\theta = \{In(x, z), Out(c)\}$

$$\mathbb{E} \log p(Y|Z, X, \theta)p(Z|X) \to \max_{\theta}$$

  Equivalent to training standard skip-gram with increased number of context words

Large scale EM

- Remember our scheme
- E-step: For each training object estimate the distribution on latent variable

$$p(z_i = k|x_i, y_i, \theta) = \frac{p(y_i|x_i, k, \theta)p(z_i = k|x_i)}{\sum_{l=1}^{K(x_i)} p(y_i|x_i, l, \theta)p(z_i = l|x_i)}$$

  We can do this in explicit manner assuming the number of meanings is reasonably small

- M-step: Optimize w.r.t. $\theta = \{In(x, z), Out(c)\}$

$$\mathbb{E} \log p(Y|Z, X, \theta)p(Z|X) \to \max_\theta$$

  Equivalent to training standard skip-gram with increased number of context words

- What if on M-step we try to make a single step towards stochastic gradient of $\mathbb{E} \log p(Y|Z, X, \theta)p(Z|X)$?

Large-scale EM

- Consider the gradient of $\mathbb{E} \log p(Y|Z, X, \theta)p(Z|X)$ in detail

$$\nabla_\theta \mathbb{E}_Z \log p(Y|Z, X, \theta)p(Z|X)$$

Large-scale EM

- Consider the gradient of $\mathbb{E} \log p(Y|Z, X, \theta)p(Z|X)$ in detail

$$\nabla_\theta \mathbb{E}_Z \log p(Y|Z, X, \theta)p(Z|X) = \nabla_\theta \mathbb{E}_Z \sum_{i=1}^{n} (\log p(y_i|z_i, x_i, \theta) + \log p(z_i|x_i))$$

Large-scale EM

- Consider the gradient of $\mathbb{E} \log p(Y|Z, X, \theta)p(Z|X)$ in detail

$$\nabla_\theta \mathbb{E}_Z \log p(Y|Z, X, \theta)p(Z|X) = \nabla_\theta \mathbb{E}_Z \sum_{i=1}^{n} (\log p(y_i|z_i, x_i, \theta) + \log p(z_i|x_i)) =$$

$$\sum_{i=1}^{n} \mathbb{E}_{z_i} (\nabla_\theta \log p(y_i|z_i, x_i, \theta) + \nabla_\theta \log p(z_i|x_i))$$

Large-scale EM

- Consider the gradient of $\mathbb{E} \log p(Y|Z,X,\theta)p(Z|X)$ in detail

$$\nabla_\theta \mathbb{E}_Z \log p(Y|Z,X,\theta)p(Z|X) = \nabla_\theta \mathbb{E}_Z \sum_{i=1}^{n} \left(\log p(y_i|z_i,x_i,\theta) + \log p(z_i|x_i)\right) =$$

$$\sum_{i=1}^{n} \mathbb{E}_{z_i} \left(\nabla_\theta \log p(y_i|z_i,x_i,\theta) + \boxed{\nabla_\theta \log p(z_i|x_i)}\right) = \sum_{i=1}^{n} \mathbb{E}_{z_i} \left(\nabla_\theta \log p(y_i|z_i,x_i,\theta)\right)$$

Does not depend on *{In, Out}*

Large-scale EM

- Consider the gradient of $\mathbb{E} \log p(Y|Z, X, \theta)p(Z|X)$ in detail

$$\nabla_\theta \mathbb{E}_Z \log p(Y|Z, X, \theta)p(Z|X) = \nabla_\theta \mathbb{E}_Z \sum_{i=1}^{n} (\log p(y_i|z_i, x_i, \theta) + \log p(z_i|x_i)) =$$

$$\sum_{i=1}^{n} \mathbb{E}_{z_i} (\nabla_\theta \log p(y_i|z_i, x_i, \theta) + \boxed{\nabla_\theta \log p(z_i|x_i)}) = \sum_{i=1}^{n} \mathbb{E}_{z_i} (\nabla_\theta \log p(y_i|z_i, x_i, \theta))$$

Does not depend on *{In, Out}*

- Its unbiased estimate is simply

$$\mathbb{E}_{z_i} \nabla_\theta \log p(y_i|z_i, x_i, \theta) = \sum_{j=1}^{K(x_i)} p(z_i = k|y_i, x_i, \theta) \nabla_\theta \log p(y_i|k, x_i, \theta)$$

Large-scale EM

- Consider the gradient of $\mathbb{E} \log p(Y|Z, X, \theta) p(Z|X)$ in detail

$$\nabla_\theta \mathbb{E}_Z \log p(Y|Z,X,\theta)p(Z|X) = \nabla_\theta \mathbb{E}_Z \sum_{i=1}^{n} (\log p(y_i|z_i, x_i, \theta) + \log p(z_i|x_i)) =$$

$$\sum_{i=1}^{n} \mathbb{E}_{z_i} (\nabla_\theta \log p(y_i|z_i, x_i, \theta) + \boxed{\nabla_\theta \log p(z_i|x_i)}) = \sum_{i=1}^{n} \mathbb{E}_{z_i} (\nabla_\theta \log p(y_i|z_i, x_i, \theta))$$

Does not depend on *{In, Out}*

- Its unbiased estimate is simply

We know from E-step

$$\mathbb{E}_{z_i} \nabla_\theta \log p(y_i|z_i, x_i, \theta) = \sum_{j=1}^{K(x_i)} \boxed{p(z_i = k|y_i, x_i, \theta)} \nabla_\theta \log p(y_i|k, x_i, \theta)$$

Large-scale EM

- Consider the gradient of $\mathbb{E} \log p(Y|Z, X, \theta) p(Z|X)$ in detail

$$\nabla_\theta \mathbb{E}_Z \log p(Y|Z, X, \theta) p(Z|X) = \nabla_\theta \mathbb{E}_Z \sum_{i=1}^{n} (\log p(y_i|z_i, x_i, \theta) + \log p(z_i|x_i)) =$$

$$\sum_{i=1}^{n} \mathbb{E}_{z_i} (\nabla_\theta \log p(y_i|z_i, x_i, \theta) + \boxed{\nabla_\theta \log p(z_i|x_i)}) = \sum_{i=1}^{n} \mathbb{E}_{z_i} (\nabla_\theta \log p(y_i|z_i, x_i, \theta))$$

Does not depend on *{In, Out}*

- Its unbiased estimate is simply

We know from E-step

$$\mathbb{E}_{z_i} \nabla_\theta \log p(y_i|z_i, x_i, \theta) = \sum_{j=1}^{K(x_i)} \boxed{p(z_i = k|y_i, x_i, \theta)} \nabla_\theta \log p(y_i|k, x_i, \theta)$$

- But to compute it we only need to know $p(z_i|y_i, x_i, \theta)$ for single training instance!

Sketch of the final algorithm

- Build Huffman tree for the dictionary
- Fix initial approximation for each $\theta = \{In(x, z), Out(c)\}$
- Do one pass through training data
  - Compute the probabilities of meanings for x_i

$$p(z_i|x_i, y_i, \theta) = \frac{p(y_i|x_i, z_i, \theta)p(z_i|x_i)}{\sum_{k=1}^{K(x_i)} p(y_i|x_i, k, \theta)p(z_i = k|x_i)}$$

  - Make one step towards stochastic gradient:

$$\theta_{new} = \theta_{old} + \alpha_i \sum_{k=1}^{K(x_i)} p(z_i = k|x_i, y_i, \theta)\nabla_\theta \log p(y_i|x_i, k, \theta)$$

What was not covered in this talk

- Each word occurence is present $2C$ times in training set and of course the corresponding x_i should have the same meaning
- We may use so-called non-parametric Bayesian inference to automatically define the number of meanings for each word
- To do this we need to set a special prior on $p(z_i|x_i)$ using so-called **Chinese restaurant process**
- To obtain tractable approximations for $p(z_i|x_i, y_i)$ we'll need to use Stochastic variational inference (Hoffman, 2013) which is similar to large-scale EM described above

Experiments: Multiple meanings

Closest words to "platform"		
fwd	stabling	software
sedan	turnback	ios
fastback	pebblemix	freeware
chrysler	citybound	netfront
hatchback	metcard	linux
notchback	underpass	microsoft
rivieraoldsmobile	sidings	browser
liftback	tram	desktop
superoldsmobile	cityrail	interface
sheetmetal	trams	newlib

Closest words to "sound"	
puget	sequencer
sounds	multitrack
island	synths
shoals	audiophile
inlet	stereo
bay	sampler
hydrophone	sequencers
quoddy	headphones
shore	reverb
buoyage	multitracks

Computer is now able to assign different semantic representations to different occurrences of same word depending on the context

Experiments: word disambiguation

- We run AdaGram with $\alpha = 0.2$
- 5 meanings for 'Waterloo' were found
- Let us try to make disambiguation

Experiments: word disambiguation

- We run AdaGram with $\alpha = 0.2$
- 5 meanings for 'Waterloo' were found
- Let us try to make disambiguation

```
Who won the Battle of Waterloo?
```

Experiments: word disambiguation

- We run AdaGram with $\alpha = 0.2$
- 5 meanings for 'Waterloo' were found
- Let us try to make disambiguation

```
Who won the Battle of Waterloo?
```

Probabilities of meanings
0.0000098
0.997716
0.0000309
0.00207717
0.00016605

Experiments: word disambiguation

- We run AdaGram with $\alpha = 0.2$
- 5 meanings for 'Waterloo' were found
- Let us try to make disambiguation

```
Who won the Battle of Waterloo?
```

Probabilities of meanings
0.0000098
0.997716
0.0000309
0.00207717
0.00016605

Closest words:
"sheriffmuir"
"agincourt"
"austerlitz"
"jena-auerstedt"
"malplaquet"
"königgrätz"
"mollwitz"
"albuera"
"toba-fushimi"
"hastenbeck"

Experiments: word disambiguation

- We run AdaGram with $\alpha = 0.2$
- 5 meanings for 'Waterloo' were found
- Let us try to make disambiguation

```
Our train has departed from Waterloo at 1100pm
```

Experiments: word disambiguation

- We run AdaGram with $\alpha = 0.2$
- 5 meanings for 'Waterloo' were found
- Let us try to make disambiguation

```
Our train has departed from Waterloo at 1100pm
```

Probabilities of meanings
0.948032
0.00427984
0.000470485
0.0422029
0.0050148

Experiments: word disambiguation

- We run AdaGram with $\alpha = 0.2$
- 5 meanings for 'Waterloo' were found
- Let us try to make disambiguation

```
Our train has departed from Waterloo at 1100pm
```

Probabilities of meanings
0.948032
0.00427984
0.000470485
0.0422029
0.0050148

Closest words:
"paddington"
"euston"
"victoria"
"liverpool"
"moorgate"
"via"
"london"
"street"
"central"
"bridge"

Downloads

- Code and documentation available

https://github.com/sbos/AdaGram.jl

- Trained models available

https://yadi.sk/d/W4FtSjA5o3jUL

- Paper available

S. Bartunov, D. Kondrashkin, A. Osokin, D. Vetrov. Breaking Sticks and Ambiguities with Adaptive Skip-gram. In *AISTATS 2016*

http://arxiv.org/abs/1502.07257

Resume: LVM

- Can fill in missing data
- Can reveal the structure in data (manifolds, clusters)
- Can find hidden information in training data
- Can handle unknown factors caused by our choice of θ, e.g. in **reinforcement learning**
- Can be used for constructing more flexible models of data with better predictive abilities
- When working with large datasets training time is approximately the same as for the analogous models without latent variables