CS395T: Structured Models for NLP
Lecture 6: Sequence Models III

Greg Durrett

Some slides adapted from Leon Gu (CMU), Taylor Berg-Kirkpatrick (CMU)

Administrivia

- P1 has been updated, didn't include adagrad_trainer.py

Recall: CRFs

$$P(\mathbf{y}|\mathbf{x}) = \frac{1}{Z}\prod_{i=2}^{n}\exp(\phi_t(y_{i-1}, y_i))\prod_{i=1}^{n}\exp(\phi_e(y_i, i, \mathbf{x}))$$

- Using standard feature-based potentials:

$$P(\mathbf{y}|\mathbf{x}) \propto \exp w^\top \left[\sum_{i=2}^{n} f_t(y_{i-1}, y_i) + \sum_{i=1}^{n} f_e(y_i, i, \mathbf{x})\right]$$

- Gradient: gold features - expected features under model

- Compute max path with Viterbi, compute feature expectations from tag probabilities with forward-backward

Structured SVM

$$w^\top f(\mathbf{x}, \mathbf{y}) = \sum_{i=2}^{n} w^\top f_t(y_{i-1}, y_i) + \sum_{i=1}^{n} w^\top f_e(x_i, y_i)$$

Minimize $\lambda \|w\|_2^2 + \sum_{j=1}^{m} \xi_j$

s.t. $\forall j \quad \xi_j \geq 0$

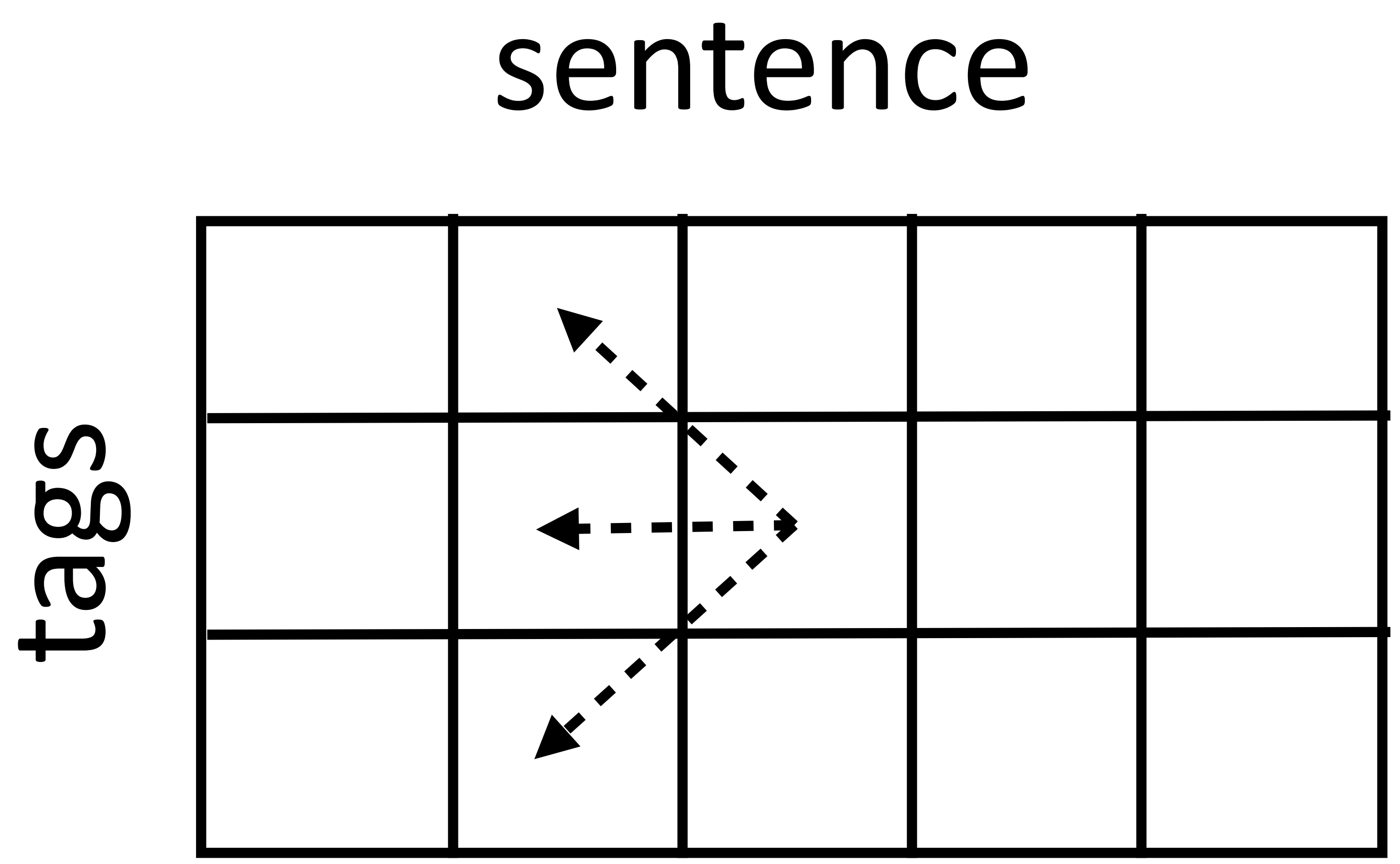
$\forall j \forall \mathbf{y} \in \mathcal{Y} \quad w^\top f(\mathbf{x}_j, \mathbf{y}_j^*) \geq w^\top f(\mathbf{x}_j, \mathbf{y}) + \ell(\mathbf{y}, \mathbf{y}_j^*) - \xi_j$

- Loss-augmented decode can be done with Viterbi
- Only need Viterbi for inference here…hmm…

Viterbi Time Complexity

VBD		VB			
VBN	VBZ	VBP	VBZ		
NNP	NNS	NN	NNS	CD	NN
Fed	raises	interest	rates	0.5	percent

- n word sentence, s tags to consider — what is the time complexity?



- $O(ns^2)$ — s is ~40 for POS, n is ~20

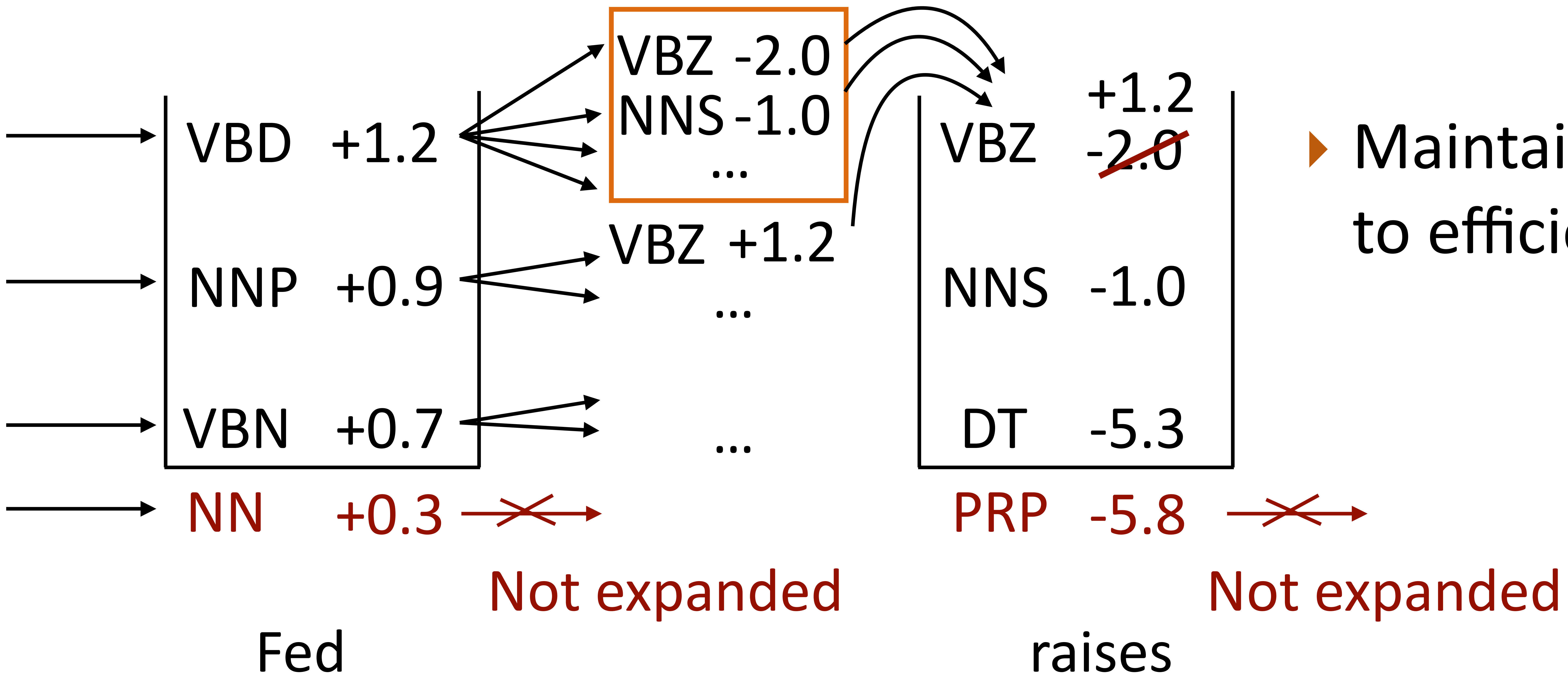
Viterbi Time Complexity

VBD		VB			
VBN	VBZ	VBP	VBZ		
NNP	NNS	NN	NNS	CD	NN
Fed	raises	interest	rates	0.5	percent

- Many tags are totally implausible
- Can any of these be:
  - Determiners?
  - Prepositions?
  - Adjectives?
- Features quickly eliminate many outcomes from consideration — don't need to consider these going forward

Beam Search

- Maintain a beam of *k* plausible states at the current timestep
- Expand all states, only keep *k* top hypotheses at new timestep



- Maintain priority queue to efficiently add things

- Beam size of k, time complexity O(nks log(k))

How good is beam search?

- *k*=1: greedy search
- Choosing beam size:
  - 2 is usually better than 1
  - Usually don't use larger than 50
  - Depends on problem structure

This Lecture

- Unsupervised POS tagging
- EM for learning HMMs
- Gradient-based unsupervised learning
- (briefly) Some writing tips

Unsupervised Learning

- Can we induce linguistic structure? Thought experiment…

a b a c c c c

b a c c c

- What's a two-state HMM that could produce this?

- What if I show you this sequence?

a a b c c a a

- What did you do? Use current model parameters + data to refine your model. This is what EM will do

Part-of-Speech Induction

- Input $\mathbf{x} = (x_1, ..., x_n)$ Output $\mathbf{y} = (y_1, ..., y_n)$
- Assume we don't have access to labeled examples — how can we learn a POS tagger?
- Key idea: optimize $P(\mathbf{x}) = \sum_{\mathbf{y}} P(\mathbf{y}, \mathbf{x})$ ⟵ Generative model explains the data ***x***
- Optimizing marginal log-likelihood with no labels **y**:

$$\mathcal{L}(\mathbf{x}_{1,\ldots,D}) = \sum_{i=1}^{D} \log \sum_{\mathbf{y}} P(\mathbf{y}, \mathbf{x}_i)$$

- non-convex optimization problem

Part-of-Speech Induction

- Input $\mathbf{x} = (x_1, ..., x_n)$ Output $\mathbf{y} = (y_1, ..., y_n)$
- Optimizing marginal log-likelihood with no labels **y**:

$$\mathcal{L}(\mathbf{x}_{1,\ldots,D}) = \sum_{i=1}^{D} \log \sum_{\mathbf{y}} P(\mathbf{y}, \mathbf{x}_i)$$

- Can't use a discriminative model; $\sum_{\mathbf{y}} P(\mathbf{y}|\mathbf{x}) = 1$, doesn't model **x**
- What's the point of this? Model has inductive bias and so should learn some useful latent structure ***y*** (clustering effect)
- EM is just one procedure for optimizing this kind of objective

Expectation Maximization

$$\log \sum_{\mathbf{y}} P(\mathbf{x}, \mathbf{y}|\theta)$$

- Condition on parameters θ

$$= \log \sum_{\mathbf{y}} q(\mathbf{y}) \frac{P(\mathbf{x}, \mathbf{y}|\theta)}{q(\mathbf{y})}$$

- Variational approximation q — this is a trick we'll return to later!

$$\geq \sum_{\mathbf{y}} q(\mathbf{y}) \log \frac{P(\mathbf{x}, \mathbf{y}|\theta)}{q(\mathbf{y})}$$

- Jensen's inequality (uses concavity of log)

$$= \mathbb{E}_{q(\mathbf{y})} \log P(\mathbf{x}, \mathbf{y}|\theta) + \text{Entropy}[q(\mathbf{y})]$$

- Can optimize this lower-bound on log likelihood instead of log-likelihood

Adapted from Leon Gu

Expectation Maximization

$$\log \sum_{\mathbf{y}} P(\mathbf{x}, \mathbf{y}|\theta) \geq \mathbb{E}_{q(\mathbf{y})} \log P(\mathbf{x}, \mathbf{y}|\theta) + \text{Entropy}[q(\mathbf{y})]$$

- Exact equality:

$$\log \sum_{\mathbf{y}} P(\mathbf{x}, \mathbf{y}|\theta) = \mathbb{E}_{q(\mathbf{y})} \log P(\mathbf{x}, \mathbf{y}|\theta) + \text{Entropy}[q(\mathbf{y})] + KL(q(\mathbf{y})||P(\mathbf{y}|\mathbf{x}, \theta))$$

- KL divergence: asymmetric measure of difference between two distributions

$$KL(q(\mathbf{y})||p(\mathbf{y})) = \sum_{\mathbf{y}} q(\mathbf{y}) \log \frac{q(\mathbf{y})}{p(\mathbf{y})}$$

  - Related to cross-entropy (= KL + entropy of q)
  - If $q(\mathbf{y}) = P(\mathbf{y}|\mathbf{x}, \theta)$, KL term is 0 so equality is achieved

Adapted from Leon Gu

Expectation Maximization

$$\log \sum_{\mathbf{y}} P(\mathbf{x}, \mathbf{y}|\theta) \geq \mathbb{E}_{q(\mathbf{y})} \log P(\mathbf{x}, \mathbf{y}|\theta) + \text{Entropy}[q(\mathbf{y})]$$

- If $q(\mathbf{y}) = P(\mathbf{y}|\mathbf{x}, \theta)$, KL term is 0 so equality is achieved
- Expectation-maximization: alternating maximization of the lower bound over *q* and θ
  - Current timestep = *t*, have parameters θ^{t-1}
  - E-step: maximize w.r.t. *q*; that is, $q^t = P(\mathbf{y}|\mathbf{x}, \theta^{t-1})$
  - M-step: maximize w.r.t. θ; that is, $\theta^t = \text{argmax}_\theta \mathbb{E}_{q^t} \log P(\mathbf{x}, \mathbf{y}|\theta)$

Adapted from Leon Gu

EM for HMMs

- Expectation-maximization: alternating maximization
  - E-step: maximize w.r.t. *q*; that is, $q^t = P(\mathbf{y}|\mathbf{x}, \theta^{t-1})$
  - M-step: maximize w.r.t. θ; that is, $\theta^t = \text{argmax}_\theta \mathbb{E}_{q^t} \log P(\mathbf{x}, \mathbf{y}|\theta)$

- E-step: for an HMM: run forward-backward with the given parameters

- Compute $P(y_i = s|\mathbf{x}, \theta^{t-1}),\ P(y_i = s_1, y_{i+1} = s_2|\mathbf{x}, \theta^{t-1})$

  tag marginals at each position

  tag pair marginals at each position

- M-step: need to find parameters to optimize the crazy argmax term

EM for HMMs

- Recall how we maximized log P(**x**,**y**): read counts off data

DT	NN
the	*dog*

→

count(DT, the) = 1
count(DT, dog) = 0
count(NN, the) = 0
count(NN, dog) = 1

→

P(the|DT) = 1
P(dog|DT) = 0
P(the|NN) = 0
P(dog|NN) = 1

- Same procedure, but maximizing P(**x**,**y**) in expectation under *q* means that *q* specifies *fractional counts*

q		
	DT: 0.9	DT: 0.3
	NN: 0.1	**NN: 0.7**
	the	*dog*

→

count(DT, the) = 0.9
count(DT, dog) = 0.3
count(NN, the) = 0.1
count(NN, dog) = 0.7

→

P(the|DT) = 0.75
P(dog|DT) = 0.25
P(the|NN) = 0.125
P(dog|NN) = 0.875

EM for HMMs

- Same for transition probabilities

q	**DT—NN: 0.6** DT—DT: 0.1 NN—DT: 0.2 NN—NN: 0.1
	the *dog*

→

P(DT|DT) = 1/7

P(NN|DT) = 6/7

P(DT|NN) = 2/3

P(NN|NN) = 1/3

How does EM learn things?

- Initialize (M-step 0):
  - Emissions

    P(the|DT) = **0.9**　　P(the|NN) = 0.05

    P(dog|DT) = 0.05　　P(dog|NN) = **0.9**

    P(marsupial|DT) = 0.05　　P(marsupial|NN) = 0.05

  - Transition probabilities: uniform
- E-step 1: (all values are approximate)

the	dog	the	marsupial
DT: 0.95	DT: 0.05	**DT: 0.95**	DT: 0.5
NN: 0.05	**NN: 0.95**	NN: 0.05	NN: 0.5

  - uniform

How does EM learn things?

- E-step 1:

DT: 0.95	DT: 0.05	**DT: 0.95**	DT: 0.5
NN: 0.05	**NN: 0.95**	NN: 0.05	NN: 0.5
the	*dog*	*the*	*marsupial*

- M-step 1:
  - Emissions aren't so different
  - Transition probabilities (approx): P(NN|DT) = 3/4, P(DT|DT) = 1/4

How does EM learn things?

- E-step 2:

DT: 0.95	DT: 0.05	**DT: 0.95**	DT: 0.30
NN: 0.05	**NN: 0.95**	NN: 0.05	**NN: 0.70**
the	*dog*	*the*	*marsupial*

- M-step 1:
  - Emissions aren't so different
  - Transition probabilities (approx): P(NN|DT) = 3/4, P(DT|DT) = 1/4

How does EM learn things?

- E-step 2:

DT: 0.95	DT: 0.05	**DT: 0.95**	DT: 0.30
NN: 0.05	**NN: 0.95**	NN: 0.05	**NN: 0.70**
the	*dog*	*the*	*marsupial*

- M-step 2:
  - Emission P(marsupial|NN) > P(marsupial|DT)
  - Remember to tag marsupial as NN in the future!
  - Context constrained what we learned! That's how data helped us

How does EM learn things?

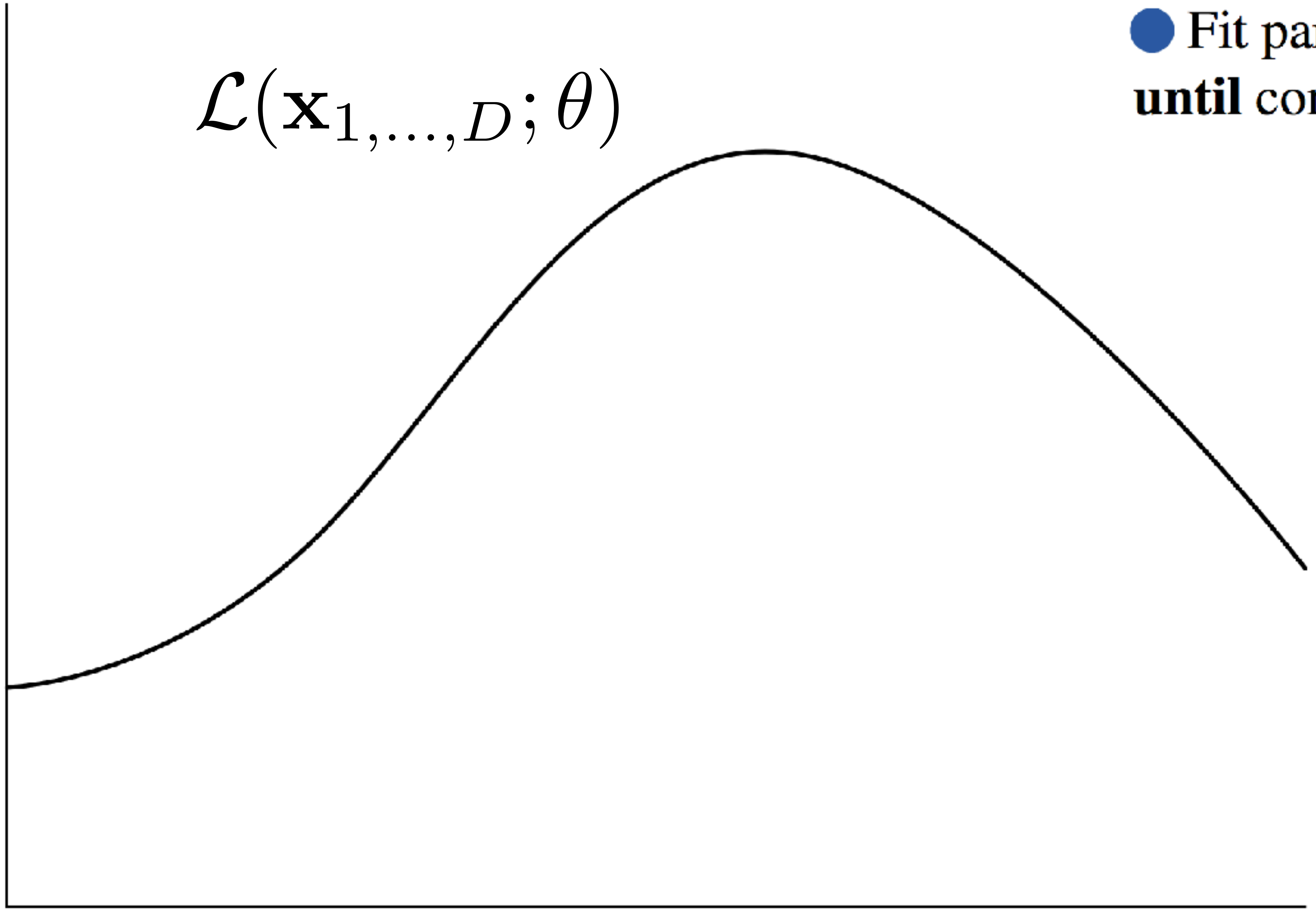
- Can think of q as a kind of “fractional annotation”
- E-step: compute annotations (posterior under current model)
- M-step: supervised learning with those fractional annotations
- Initialize with some reasonable weights, alternate E and M until convergence

EM's Lower Bound

$$\mathcal{L}(\mathbf{x}_{1,\ldots,D}) = \sum_{i=1}^{D} \log \sum_{\mathbf{y}} P(\mathbf{y}, \mathbf{x}_i)$$

Initialize probabilities $\boldsymbol{\theta}$
repeat
- Compute expected counts $\mathbf{e}$
- Fit parameters $\boldsymbol{\theta}$

until convergence



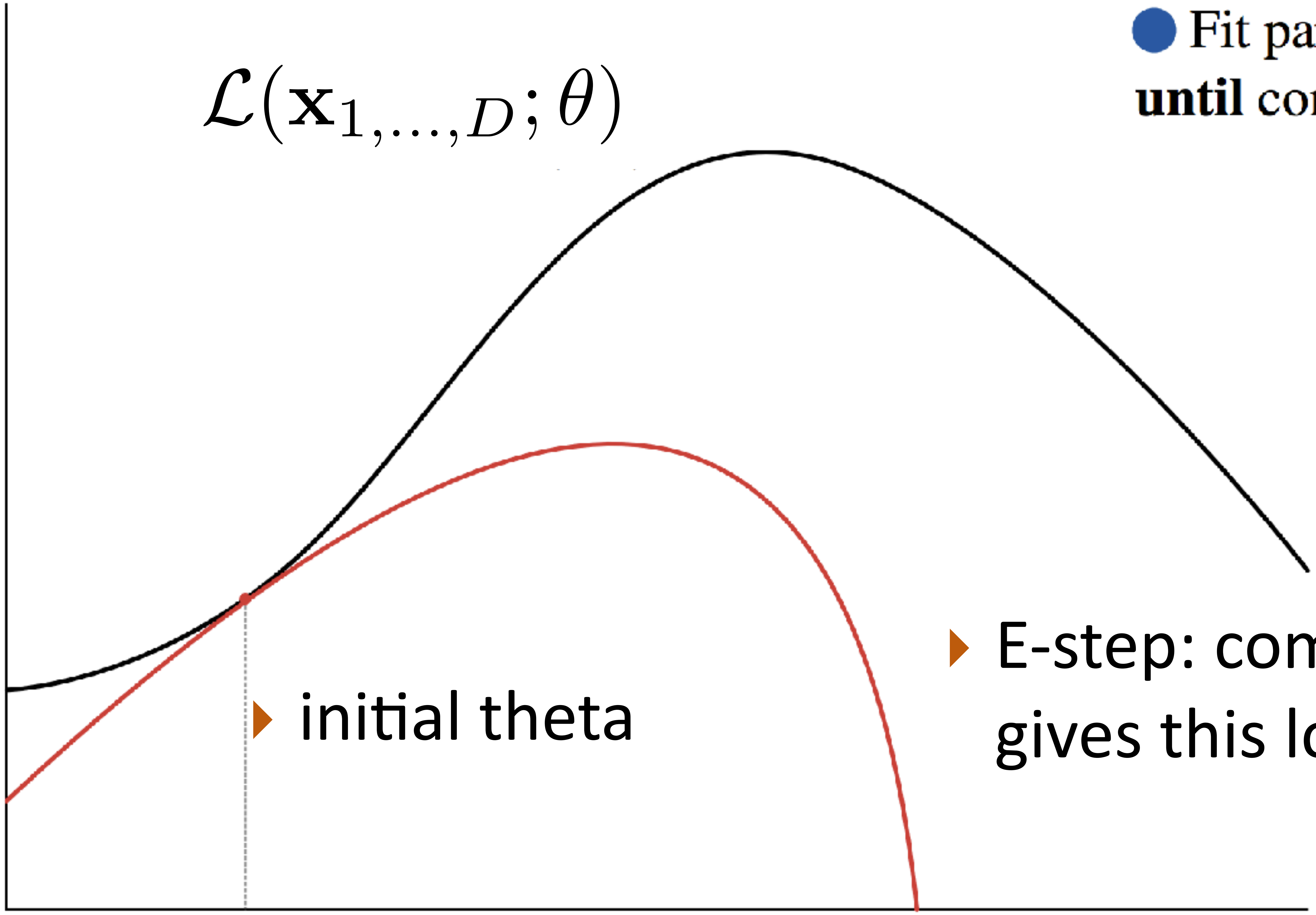
slide credit: Taylor Berg-Kirkpatrick

EM's Lower Bound

$$\mathcal{L}(\mathbf{x}_{1,\ldots,D}) = \sum_{i=1}^{D} \log \sum_{\mathbf{y}} P(\mathbf{y}, \mathbf{x}_i)$$

Initialize probabilities $\boldsymbol{\theta}$
repeat
- Compute expected counts $\mathbf{e}$
- Fit parameters $\boldsymbol{\theta}$

until convergence



- initial theta
- E-step: compute *q* which gives this lower bound

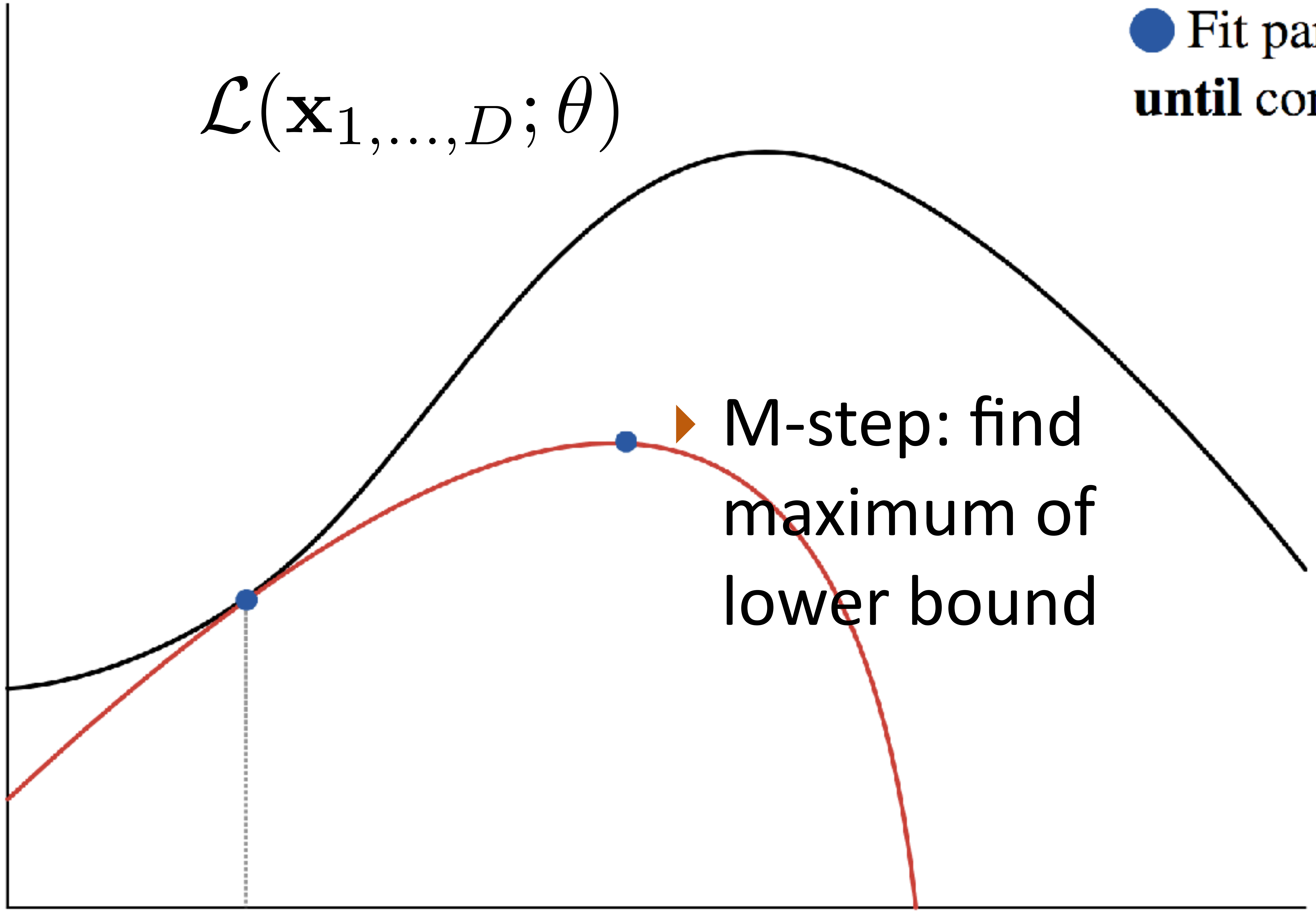
slide credit: Taylor Berg-Kirkpatrick

EM's Lower Bound

$$\mathcal{L}(\mathbf{x}_{1,\dots,D}) = \sum_{i=1}^{D} \log \sum_{\mathbf{y}} P(\mathbf{y}, \mathbf{x}_i)$$

Initialize probabilities $\boldsymbol{\theta}$
repeat
- Compute expected counts $\mathbf{e}$
- Fit parameters $\boldsymbol{\theta}$

until convergence



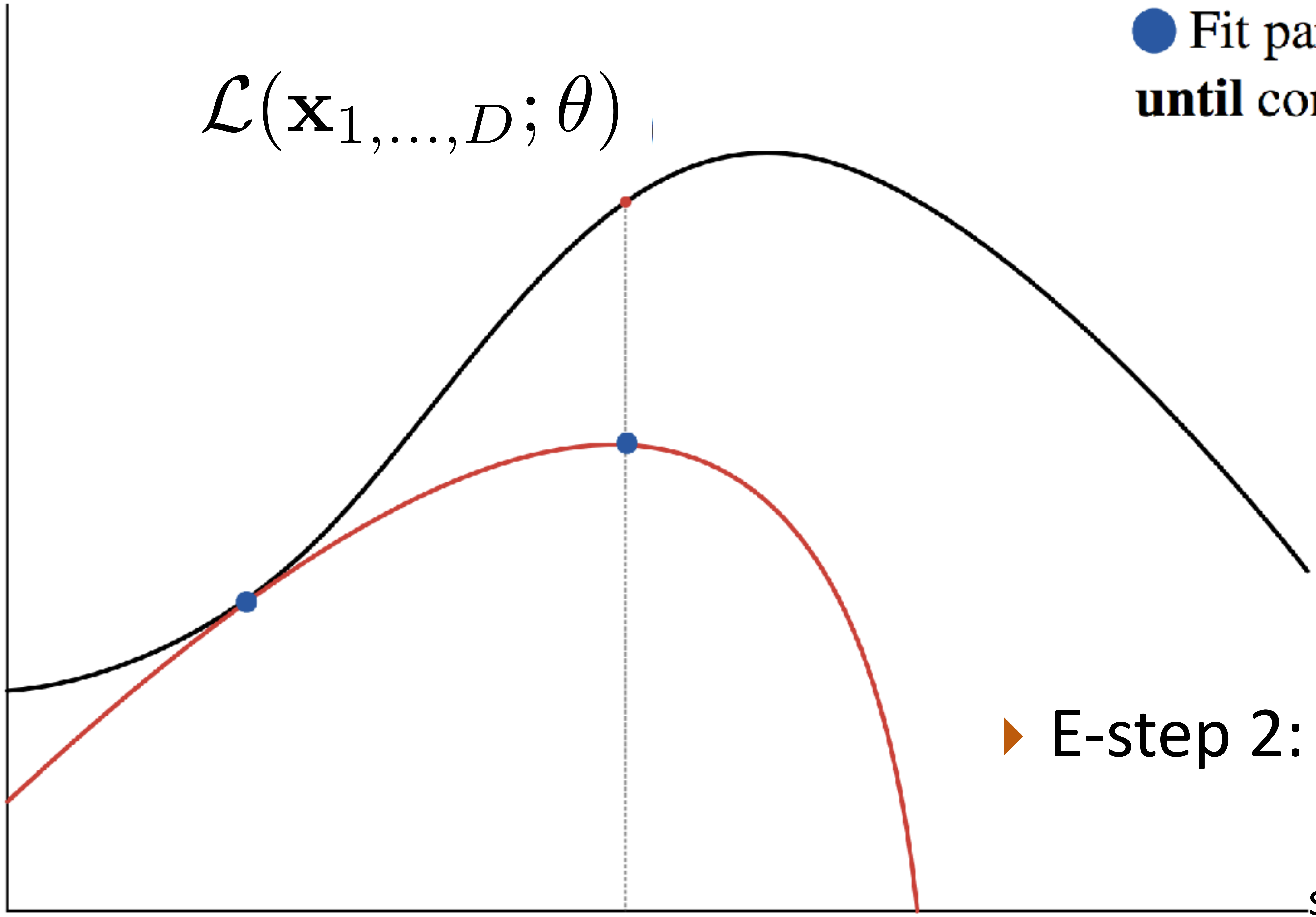
- M-step: find maximum of lower bound

slide credit: Taylor Berg-Kirkpatrick

EM's Lower Bound

$$\mathcal{L}(\mathbf{x}_{1,\ldots,D}) = \sum_{i=1}^{D} \log \sum_{\mathbf{y}} P(\mathbf{y}, \mathbf{x}_i)$$

Initialize probabilities $\boldsymbol{\theta}$
repeat
Compute expected counts $\mathbf{e}$
Fit parameters $\boldsymbol{\theta}$
until convergence

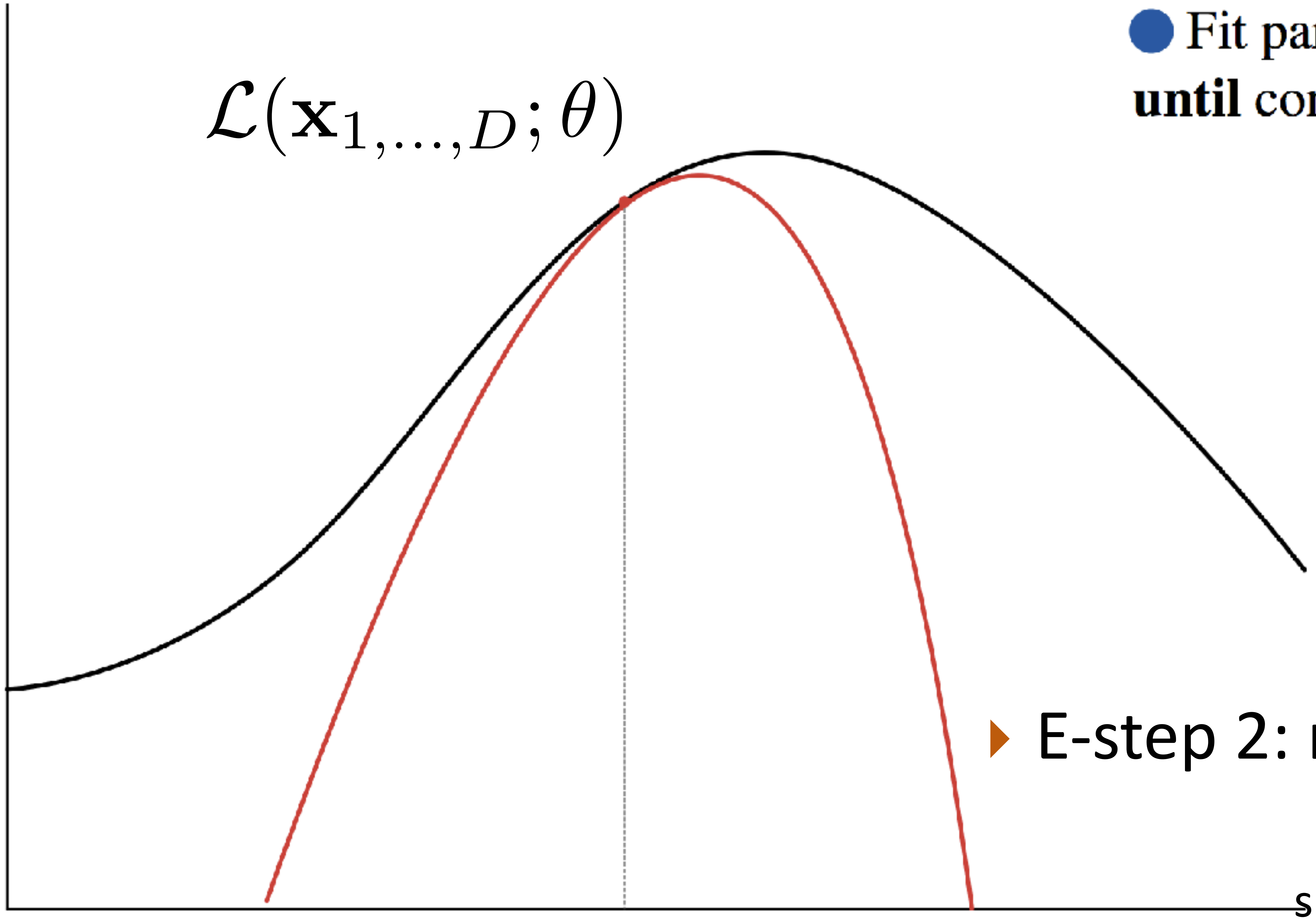


- E-step 2: re-estimate q

slide credit: Taylor Berg-Kirkpatrick

EM's Lower Bound

$$\mathcal{L}(\mathbf{x}_{1,\ldots,D}) = \sum_{i=1}^{D} \log \sum_{\mathbf{y}} P(\mathbf{y}, \mathbf{x}_i)$$



Initialize probabilities $\boldsymbol{\theta}$
repeat
- Compute expected counts $\mathbf{e}$
- Fit parameters $\boldsymbol{\theta}$

until convergence

- E-step 2: re-estimate *q*

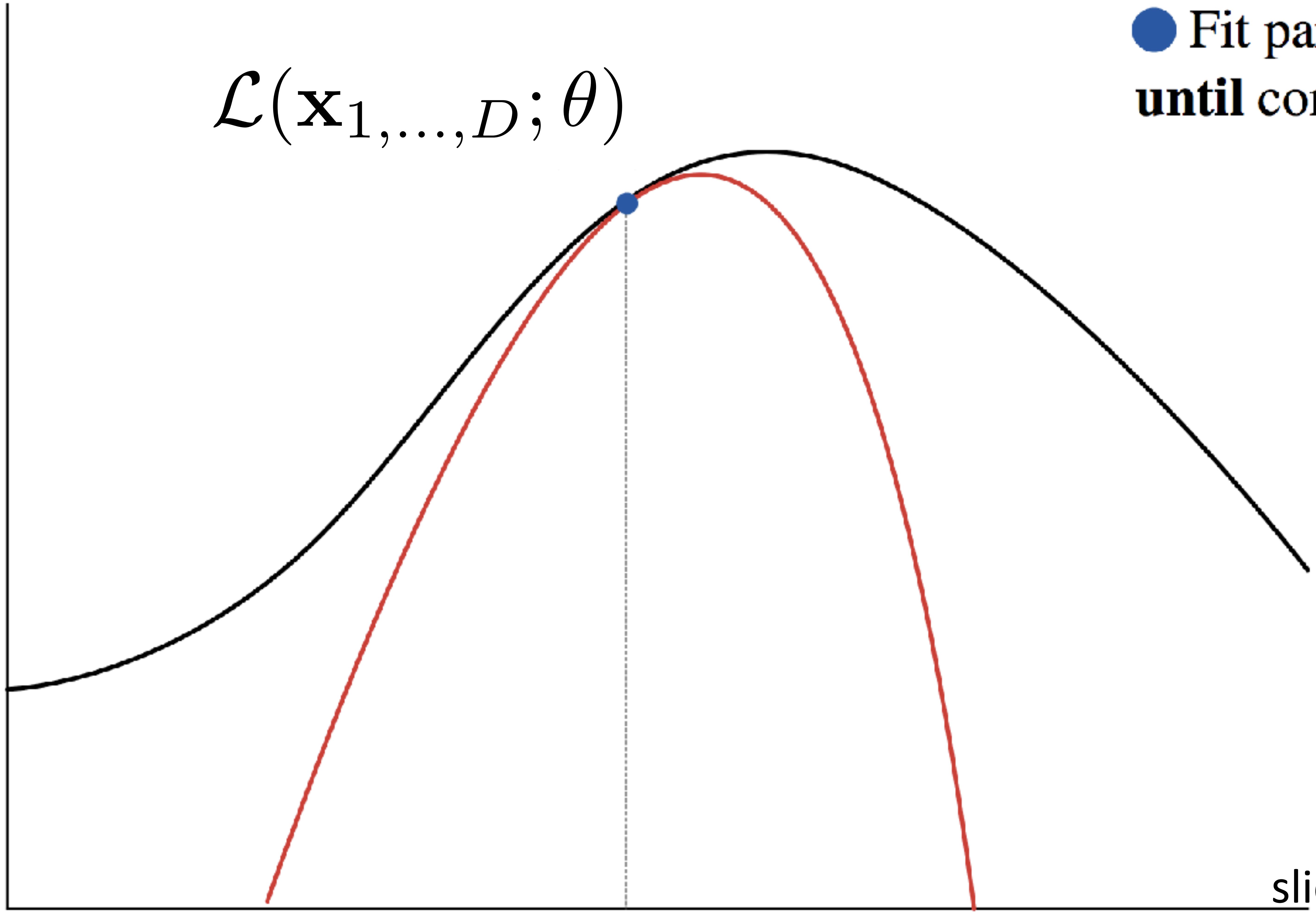
slide credit: Taylor Berg-Kirkpatrick

EM's Lower Bound

$$\mathcal{L}(\mathbf{x}_{1,\ldots,D}) = \sum_{i=1}^{D} \log \sum_{\mathbf{y}} P(\mathbf{y}, \mathbf{x}_i)$$

Initialize probabilities $\boldsymbol{\theta}$
repeat
- Compute expected counts $\mathbf{e}$
- Fit parameters $\boldsymbol{\theta}$

until convergence

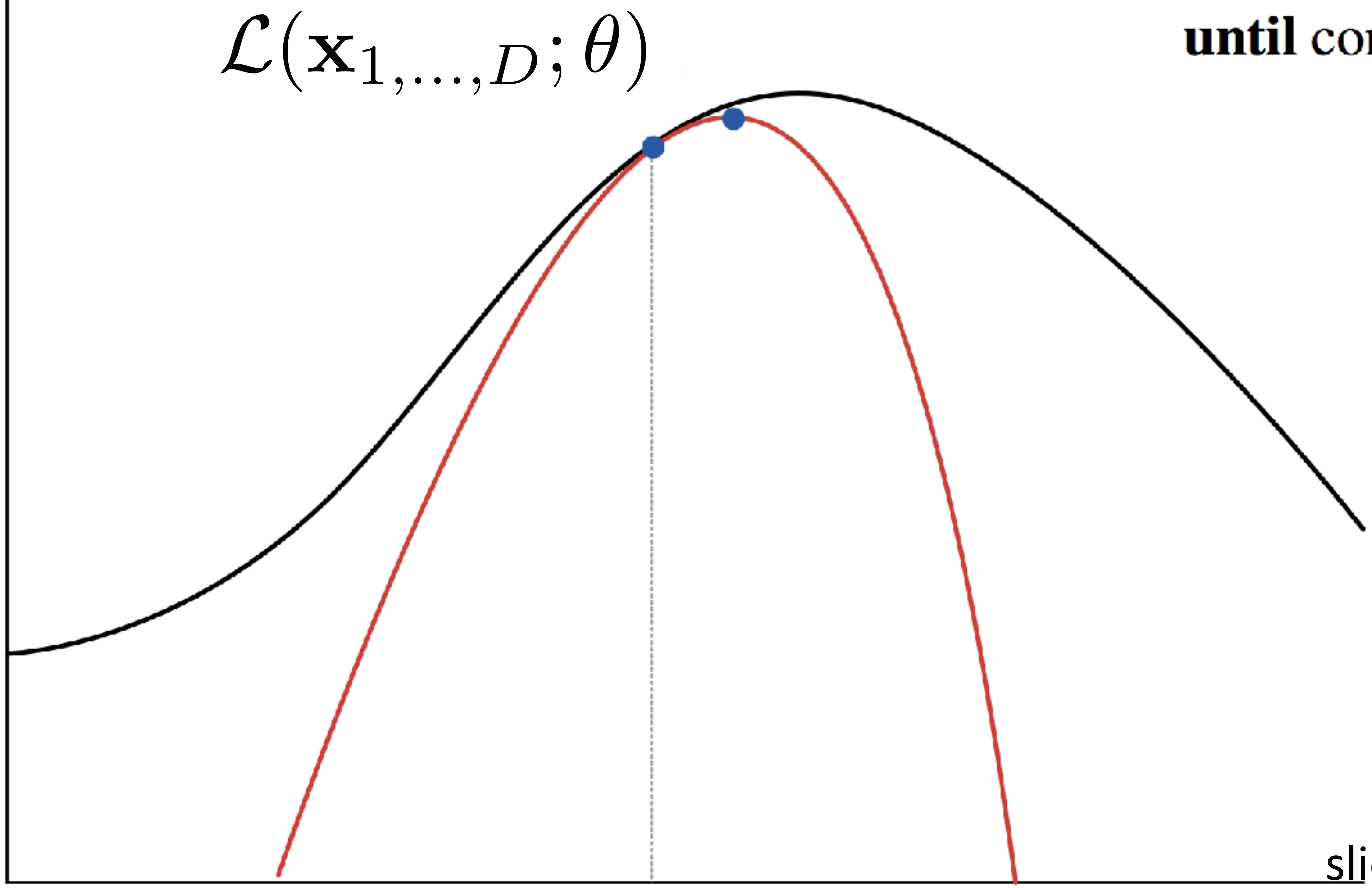


slide credit: Taylor Berg-Kirkpatrick

EM's Lower Bound

$$\mathcal{L}(\mathbf{x}_{1,\ldots,D}) = \sum_{i=1}^{D} \log \sum_{\mathbf{y}} P(\mathbf{y}, \mathbf{x}_i)$$

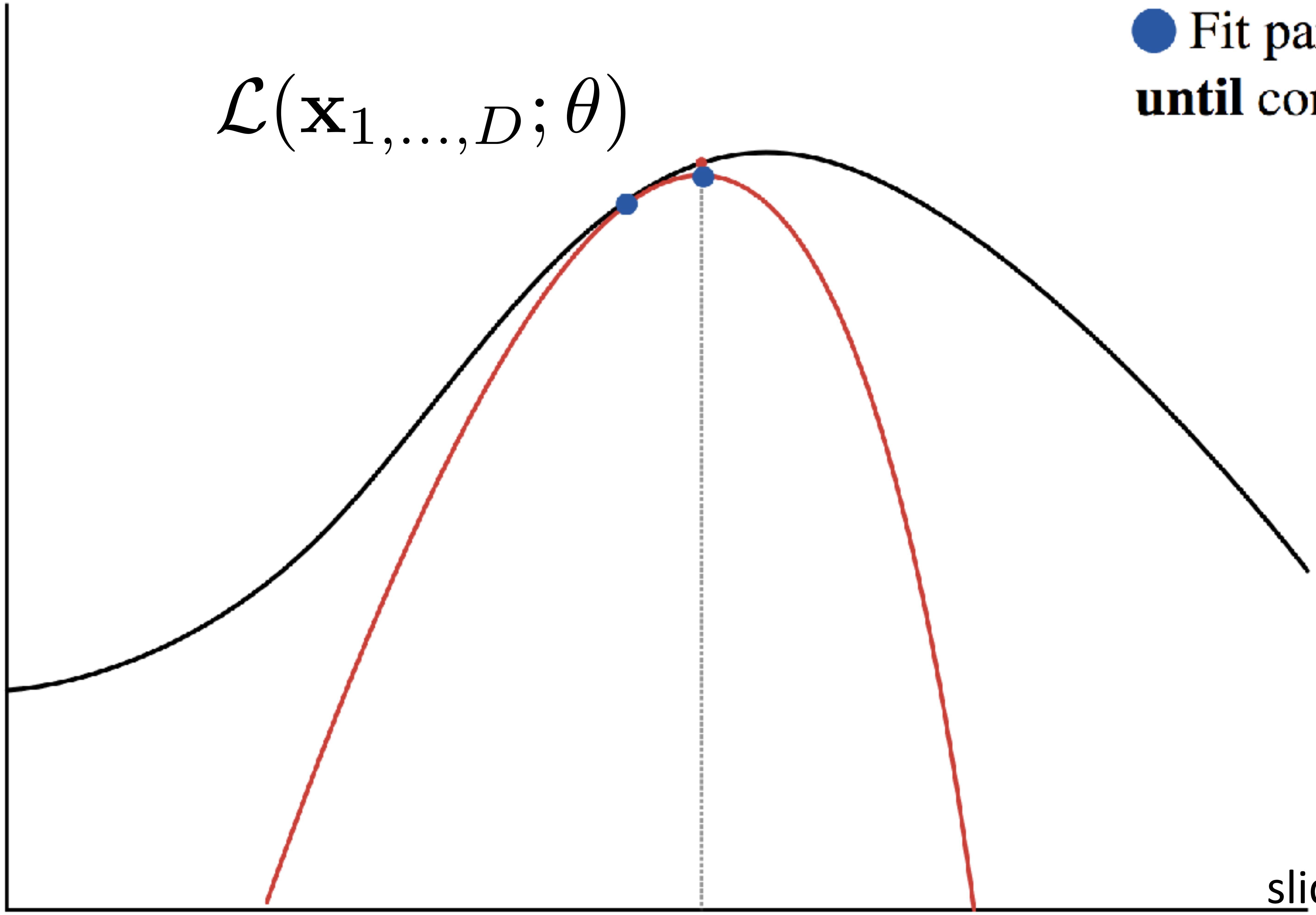
Initialize probabilities $\boldsymbol{\theta}$
repeat
● Compute expected counts e
● Fit parameters $\boldsymbol{\theta}$
until convergence



slide credit: Taylor Berg-Kirkpatrick

EM's Lower Bound

$$\mathcal{L}(\mathbf{x}_{1,\ldots,D}) = \sum_{i=1}^{D} \log \sum_{\mathbf{y}} P(\mathbf{y}, \mathbf{x}_i)$$



Initialize probabilities $\boldsymbol{\theta}$

repeat

- (red) Compute expected counts $\mathbf{e}$
- (blue) Fit parameters $\boldsymbol{\theta}$

until convergence

slide credit: Taylor Berg-Kirkpatrick

Part-of-speech Induction

- Merialdo (1994): you have a whitelist of tags for each word
- Learn parameters on *k* examples to start, use those to initialize EM, run on 1 million words of unlabeled data
- Tag dictionary + data should get us started in the right direction…

Part-of-speech Induction

	Number of tagged sentences used for the initial model						
	0	100	2000	5000	10000	20000	all
Iter	Correct tags (% words) after ML on 1M words						
0	77.0	90.0	95.4	96.2	96.6	96.9	97.0
1	80.5	92.6	95.8	96.3	96.6	96.7	96.8
2	81.8	93.0	95.7	96.1	96.3	96.4	96.4
3	83.0	93.1	95.4	95.8	96.1	96.2	96.2
4	84.0	93.0	95.2	95.5	95.8	96.0	96.0
5	84.8	92.9	95.1	95.4	95.6	95.8	95.8
6	85.3	92.8	94.9	95.2	95.5	95.6	95.7
7	85.8	92.8	94.7	95.1	95.3	95.5	95.5
8	86.1	92.7	94.6	95.0	95.2	95.4	95.4
9	86.3	92.6	94.5	94.9	95.1	95.3	95.3
10	86.6	92.6	94.4	94.8	95.0	95.2	95.2

- Small amounts of data > large amounts of unlabeled data
- Running EM *hurts* performance once you have labeled data

Merialdo (1994)

Does unsupervised learning help?

- Sometimes can produce good representations: Stallard et al. (2012) shows that unsupervised morphological segmentation can work as well as supervised segmentation for Arabic machine translation
- Later in the course: word embeddings produced from “naturally supervised” data

EM with Features

Berg-Kirkpatrick et al. (2010)

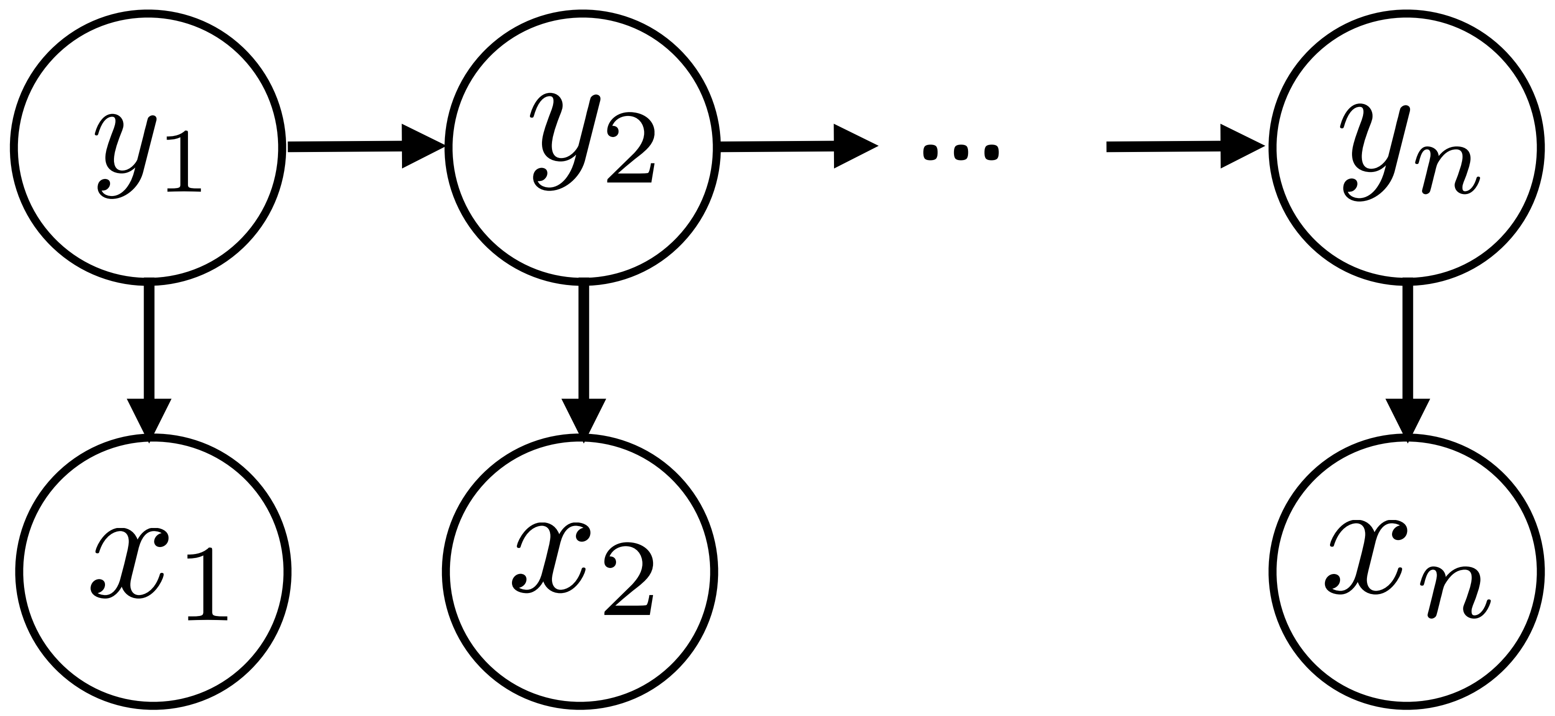
- Can use more sophisticated forms of P(**x,y**)

- Idea: still a generative model, but instead of distributions being multinomials, have them be log-linear models

$$P(x_i|y_i) = \frac{\exp(w^\top f(x_i, y_i))}{\sum_x \exp(w^\top f(x, y_i))}$$

  - normalized over all words

- Features can only look at current word and tag!



- Still a generative model but local arcs are parameterized in a log-linear way

- CRFs *don't* have this local parameterization

EM with Features

Key distribution: $P(x|\text{NNP})$

$\mathbf{w}$:

+Cap	+1.2
+ing	-0.3

$\theta_{x\|\text{NNP}}$	x	$\mathbf{f}$	$e^{\mathbf{w}^\top \mathbf{f}}$
0.1	John	+Cap	0.3
0.0	Mary	+Cap	0.3
0.2	running	+ing	0.1
0.0	jumping	+ing	0.1

Berg-Kirkpatrick et al. (2010)

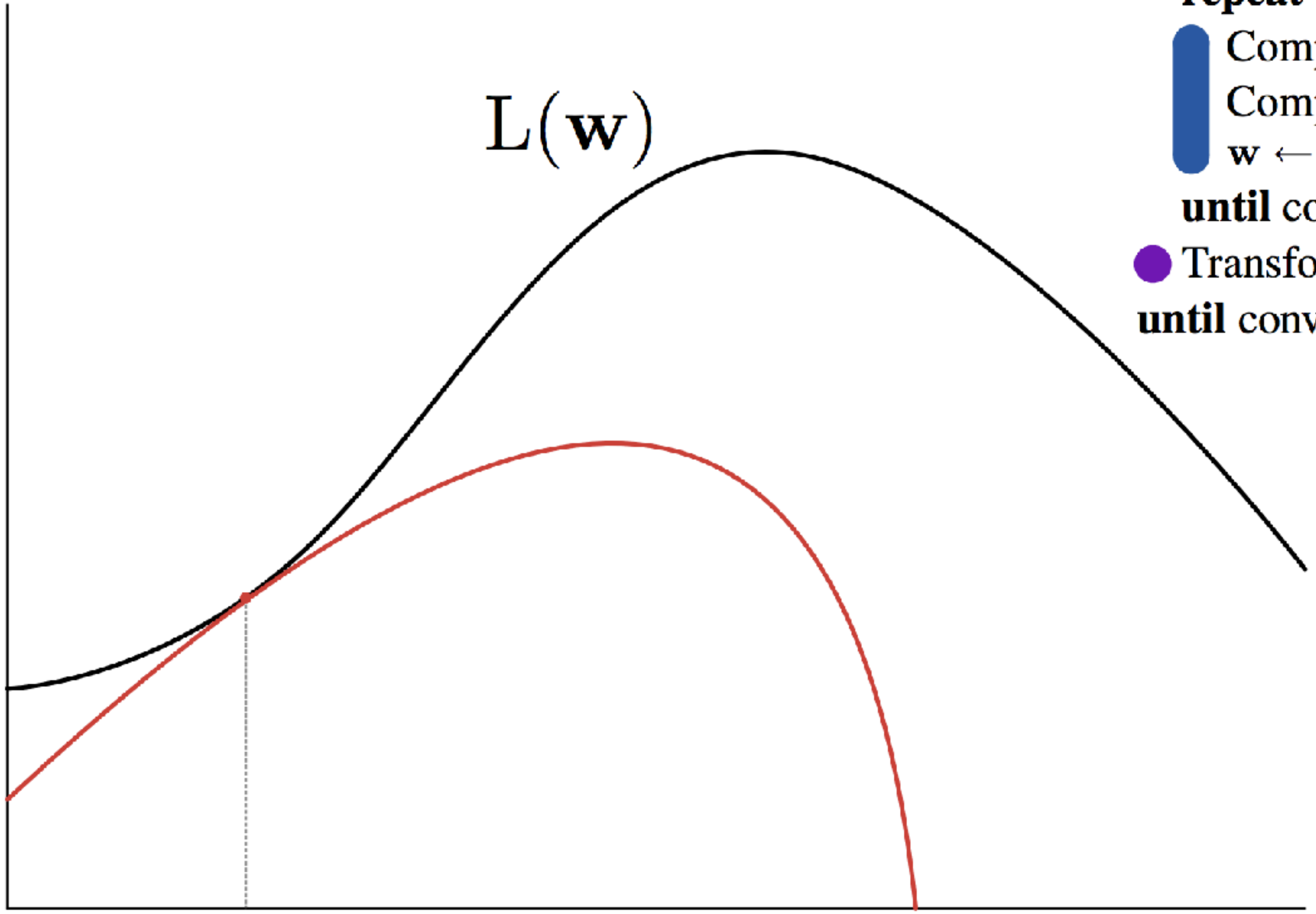
EM with Features

$$P(x_i|y_i) = \frac{\exp(w^\top f(x_i, y_i))}{\sum_x \exp(w^\top f(x, y_i))}$$

- Learning:
  - E-step is the same
  - M-step now requires gradients (slightly different than CRF gradients due to local normalization)
- One approach: can run gradient to completion each M-step (i.e., fully fit the fractional annotations we have)

Berg-Kirkpatrick et al. (2010)

EM with Features



Initialize weights $\mathbf{w}$

repeat

- Compute expected counts $\mathbf{e}$

  repeat

  - Compute $\ell(\mathbf{w}, \mathbf{e})$
  - Compute $\nabla\ell(\mathbf{w}, \mathbf{e})$
  - $\mathbf{w} \leftarrow \text{climb}(\mathbf{w}, \ell(\mathbf{w}, \mathbf{e}), \nabla\ell(\mathbf{w}, \mathbf{e}))$

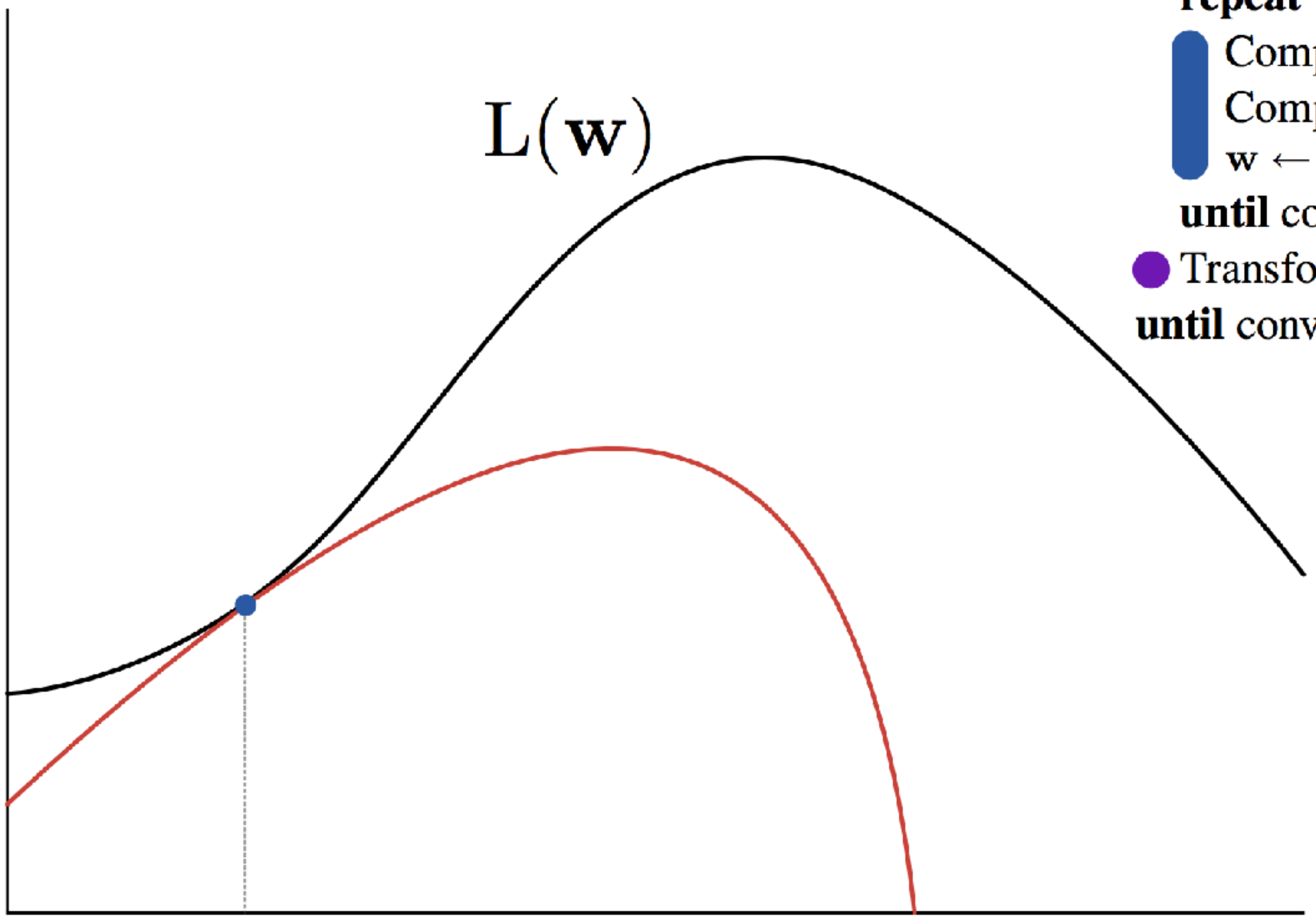
  until convergence

- Transform $\mathbf{w}$ to $\boldsymbol{\theta}$

until convergence

Berg-Kirkpatrick et al. (2010)

EM with Features



Initialize weights $\mathbf{w}$
repeat
- Compute expected counts $\mathbf{e}$
  repeat
  - Compute $\ell(\mathbf{w}, \mathbf{e})$
  - Compute $\nabla\ell(\mathbf{w}, \mathbf{e})$
  - $\mathbf{w} \leftarrow \text{climb}(\mathbf{w}, \ell(\mathbf{w}, \mathbf{e}), \nabla\ell(\mathbf{w}, \mathbf{e}))$

  until convergence
- Transform $\mathbf{w}$ to $\boldsymbol{\theta}$

until convergence

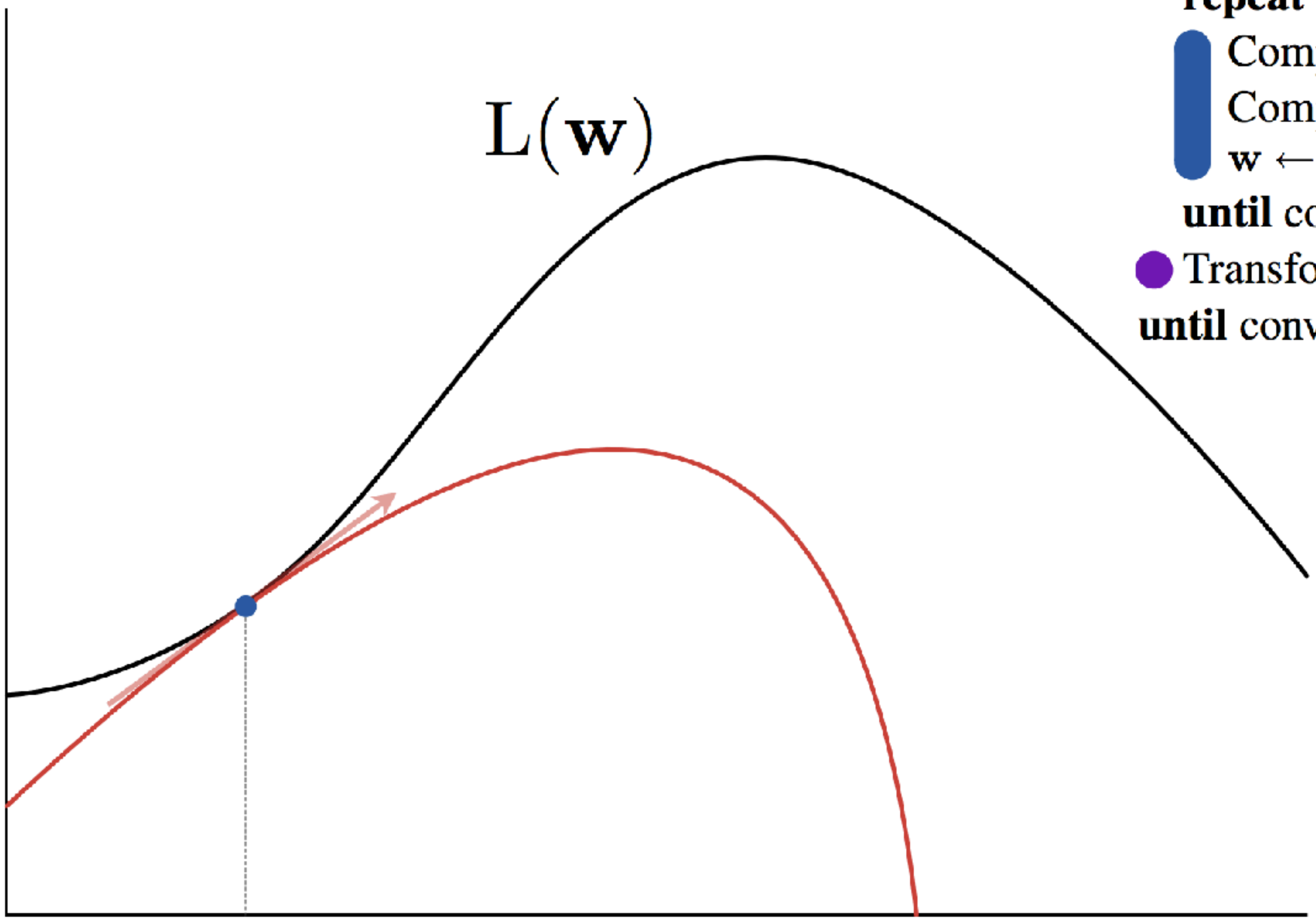
Berg-Kirkpatrick et al. (2010)

EM with Features

Initialize weights $\mathbf{w}$
repeat
- Compute expected counts $\mathbf{e}$
  repeat
  - Compute $\ell(\mathbf{w}, \mathbf{e})$
  - Compute $\nabla\ell(\mathbf{w}, \mathbf{e})$
  - $\mathbf{w} \leftarrow \text{climb}(\mathbf{w}, \ell(\mathbf{w}, \mathbf{e}), \nabla\ell(\mathbf{w}, \mathbf{e}))$

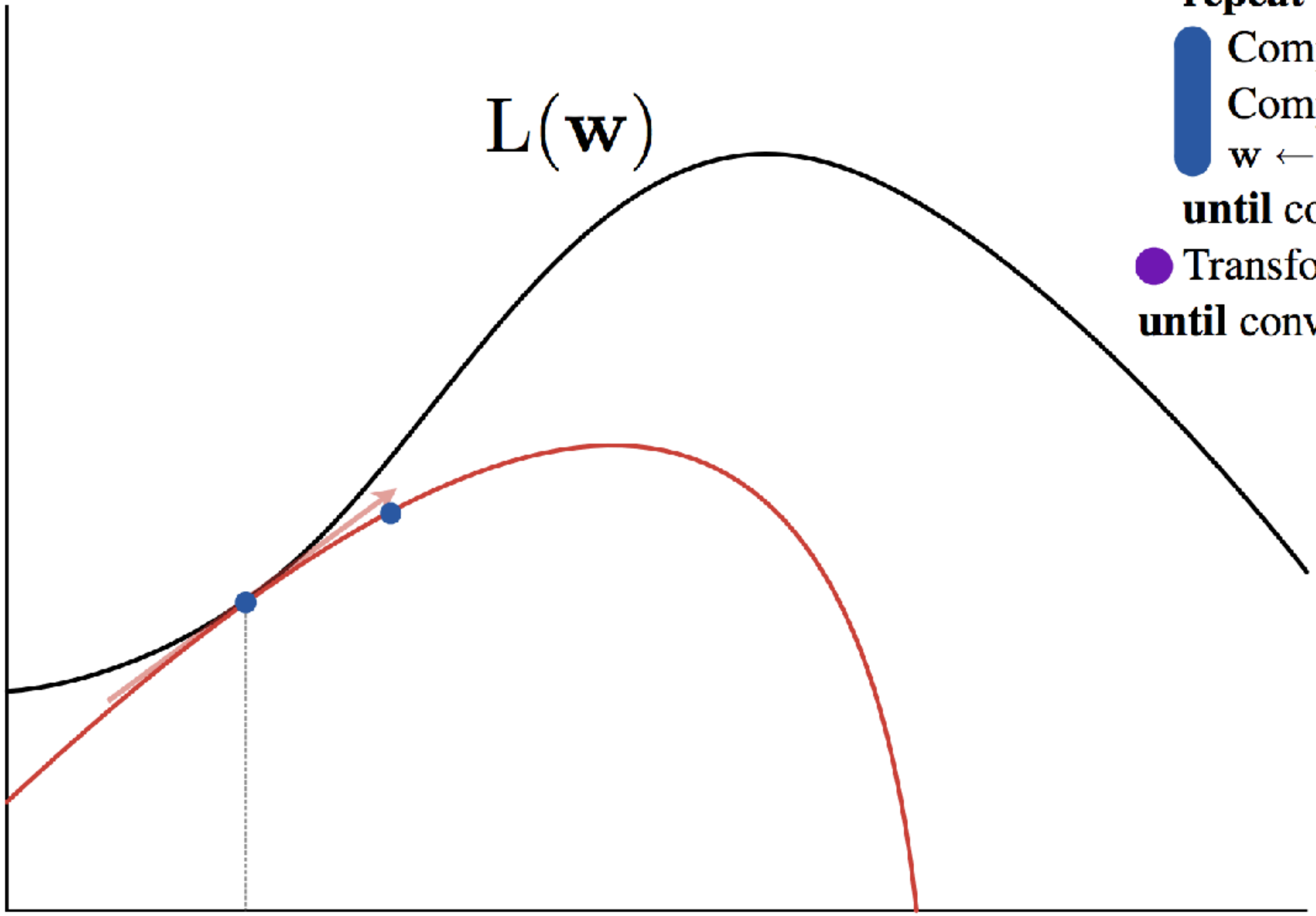
  until convergence
- Transform $\mathbf{w}$ to $\boldsymbol{\theta}$

until convergence



Berg-Kirkpatrick et al. (2010)

EM with Features



Initialize weights $\mathbf{w}$

repeat

- Compute expected counts $\mathbf{e}$

  repeat

  - Compute $\ell(\mathbf{w}, \mathbf{e})$
  - Compute $\nabla\ell(\mathbf{w}, \mathbf{e})$
  - $\mathbf{w} \leftarrow \text{climb}(\mathbf{w}, \ell(\mathbf{w}, \mathbf{e}), \nabla\ell(\mathbf{w}, \mathbf{e}))$

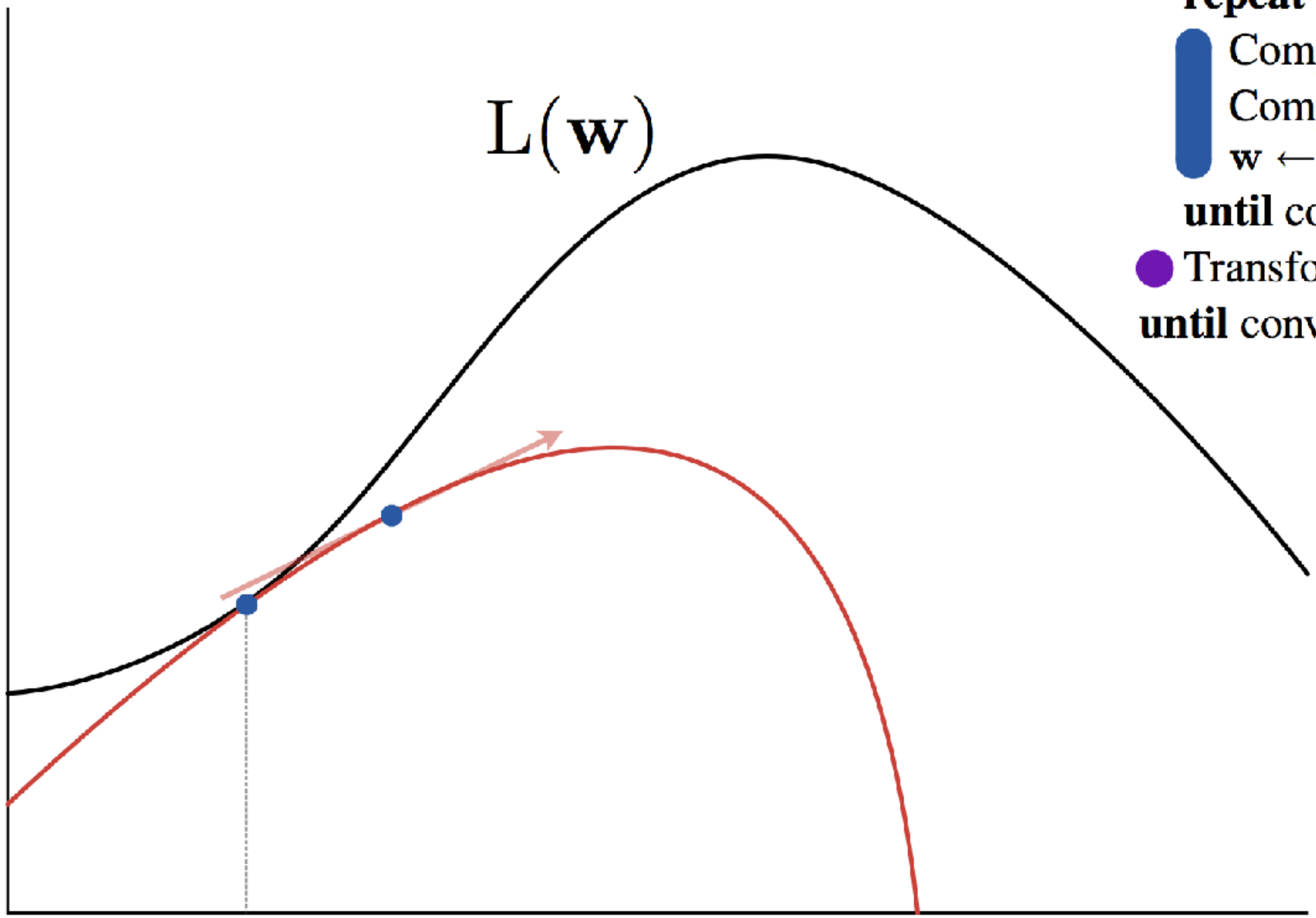
  until convergence

- Transform $\mathbf{w}$ to $\boldsymbol{\theta}$

until convergence

Berg-Kirkpatrick et al. (2010)

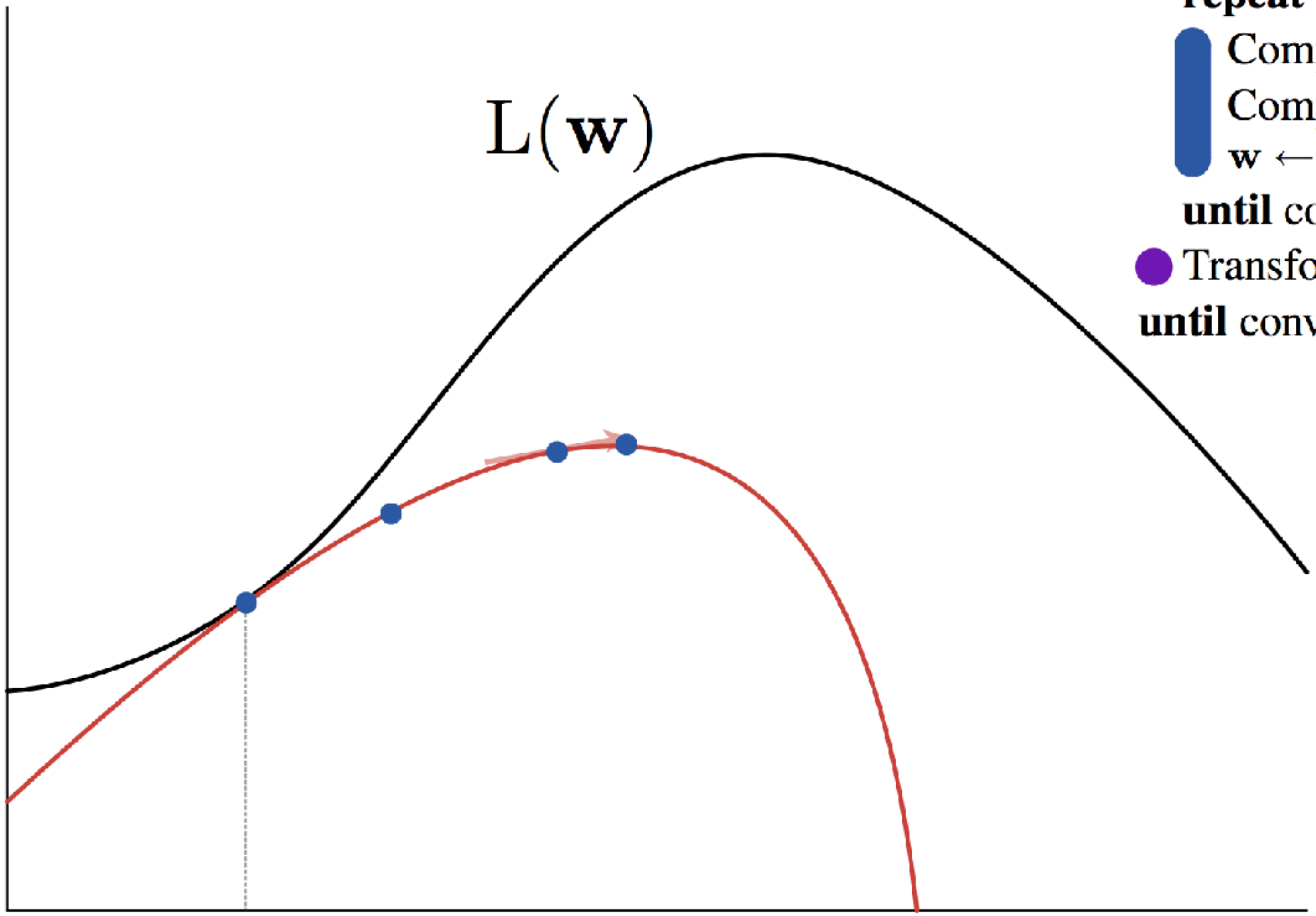
EM with Features



Initialize weights $\mathbf{w}$
repeat
Compute expected counts $\mathbf{e}$
repeat
Compute $\ell(\mathbf{w}, \mathbf{e})$
Compute $\nabla\ell(\mathbf{w}, \mathbf{e})$
$\mathbf{w} \leftarrow \text{climb}(\mathbf{w}, \ell(\mathbf{w}, \mathbf{e}), \nabla\ell(\mathbf{w}, \mathbf{e}))$
until convergence
Transform $\mathbf{w}$ to $\boldsymbol{\theta}$
until convergence

Berg-Kirkpatrick et al. (2010)

EM with Features



Initialize weights $\mathbf{w}$
repeat
● Compute expected counts $\mathbf{e}$
  repeat
    Compute $\ell(\mathbf{w}, \mathbf{e})$
    Compute $\nabla\ell(\mathbf{w}, \mathbf{e})$
    $\mathbf{w} \leftarrow \text{climb}(\mathbf{w}, \ell(\mathbf{w}, \mathbf{e}), \nabla\ell(\mathbf{w}, \mathbf{e}))$
  until convergence
● Transform $\mathbf{w}$ to $\boldsymbol{\theta}$
until convergence

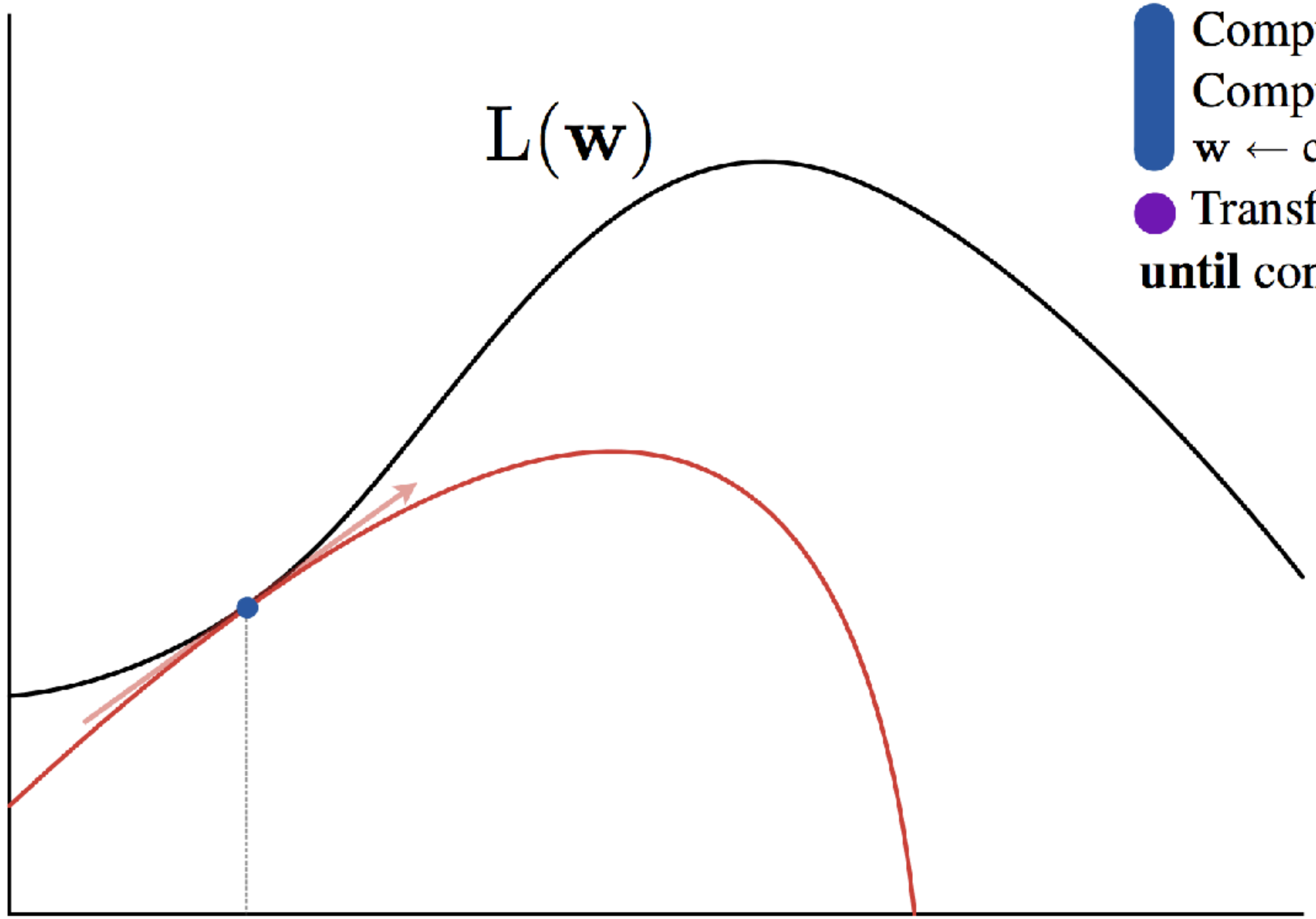
Berg-Kirkpatrick et al. (2010)

EM with Features

- Faster approach: after the E-step, just take *one* gradient step
- "Direct gradient" on the marginal log likelihood $\log \sum_{\mathbf{y}} P(\mathbf{x}, \mathbf{y} | \theta)$

Berg-Kirkpatrick et al. (2010)

EM with Features



Initialize weights $\mathbf{w}$

repeat

Compute expected counts $\mathbf{e}$

Compute $L(\mathbf{w})$

Compute $\nabla \ell(\mathbf{w}, \mathbf{e})$

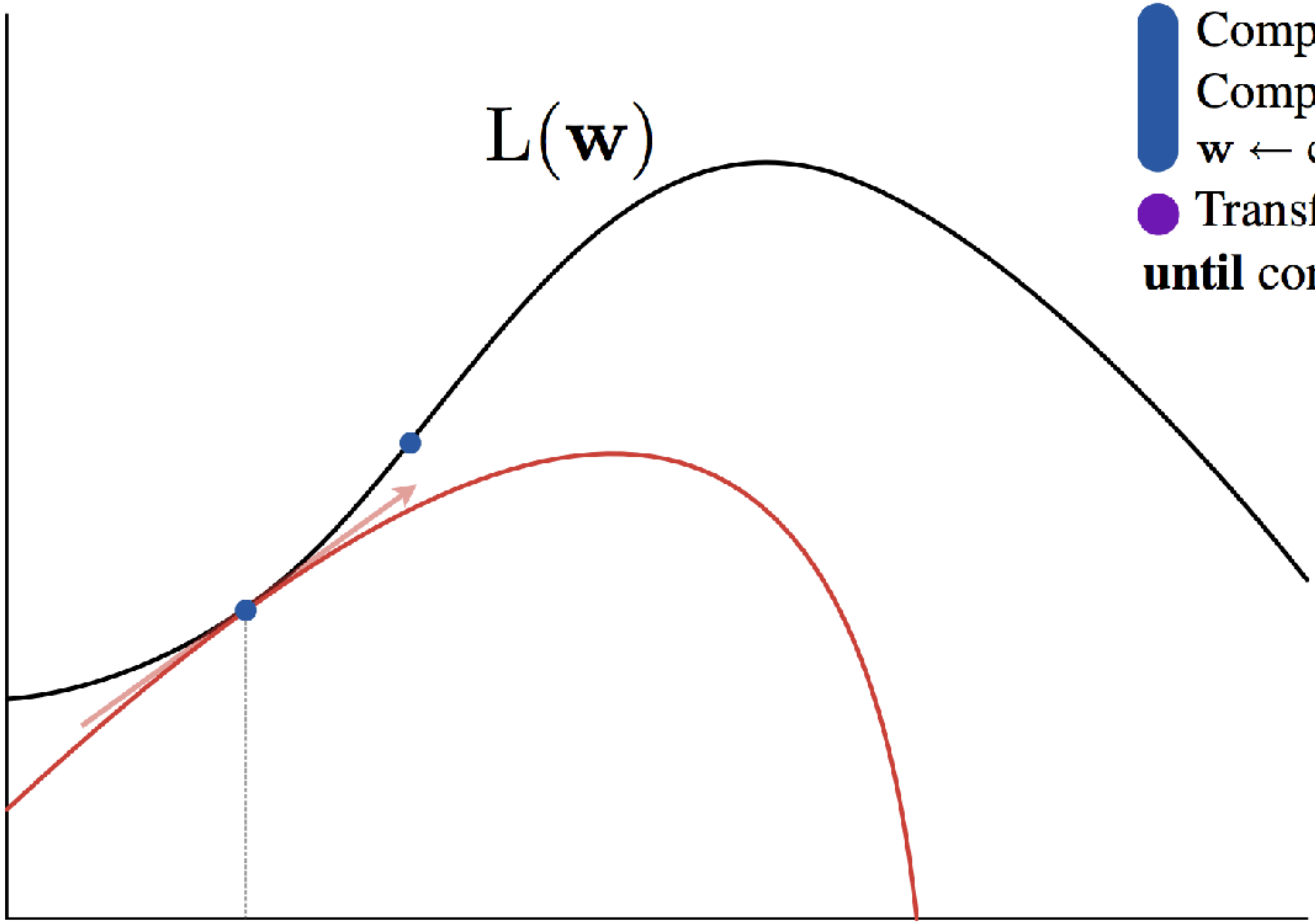
$\mathbf{w} \leftarrow \text{climb}(\mathbf{w}, L(\mathbf{w}), \nabla \ell(\mathbf{w}, \mathbf{e}))$

Transform $\mathbf{w}$ to $\boldsymbol{\theta}$

until convergence

Berg-Kirkpatrick et al. (2010)

EM with Features



Initialize weights $\mathbf{w}$

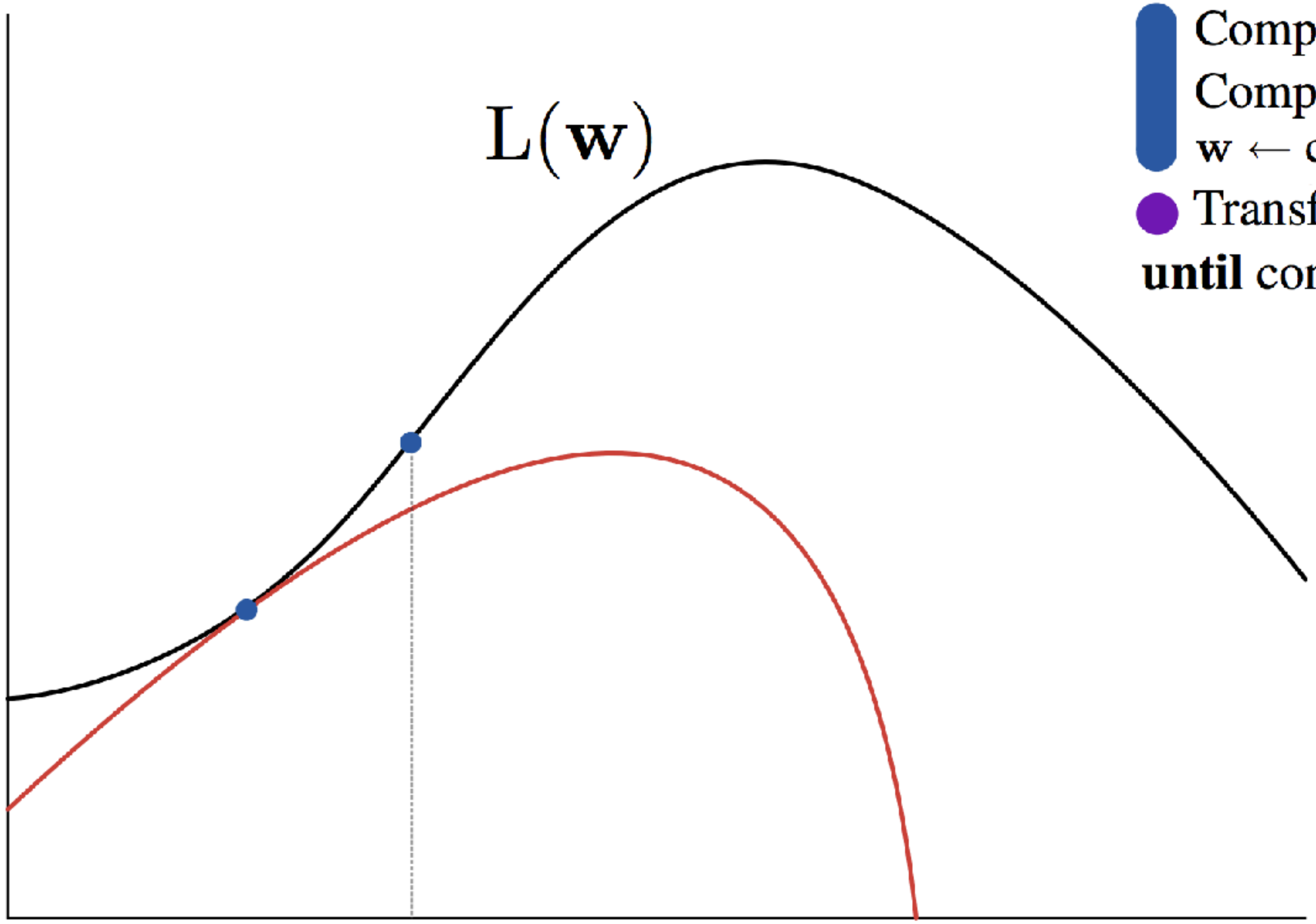
repeat

- Compute expected counts $\mathbf{e}$
- Compute $L(\mathbf{w})$
- Compute $\nabla\ell(\mathbf{w}, \mathbf{e})$
- $\mathbf{w} \leftarrow \text{climb}(\mathbf{w}, L(\mathbf{w}), \nabla\ell(\mathbf{w}, \mathbf{e}))$
- Transform $\mathbf{w}$ to $\boldsymbol{\theta}$

until convergence

Berg-Kirkpatrick et al. (2010)

EM with Features



Initialize weights $\mathbf{w}$

repeat

Compute expected counts $\mathbf{e}$

Compute $L(\mathbf{w})$

Compute $\nabla \ell(\mathbf{w}, \mathbf{e})$

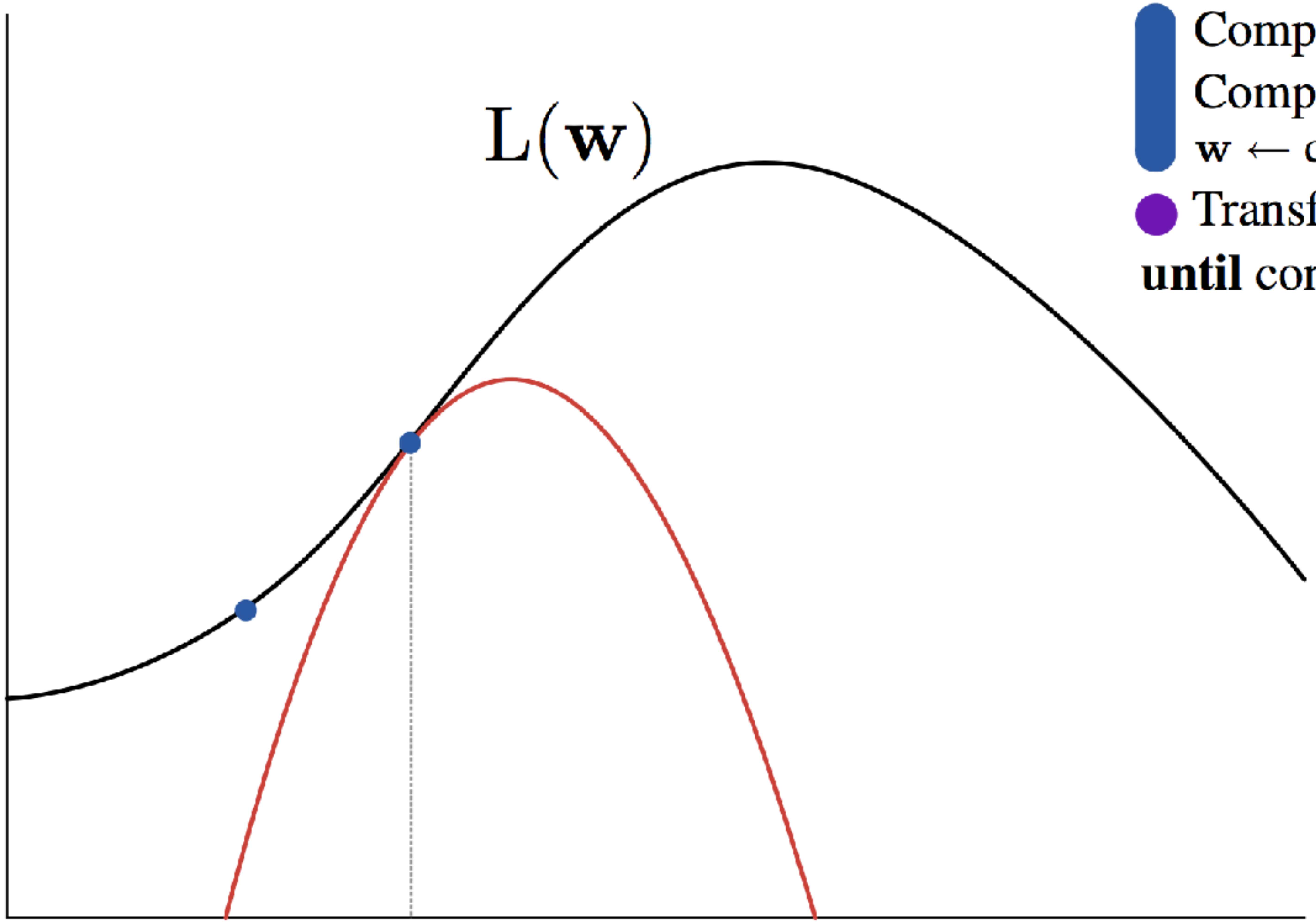
$\mathbf{w} \leftarrow \text{climb}(\mathbf{w}, L(\mathbf{w}), \nabla \ell(\mathbf{w}, \mathbf{e}))$

Transform $\mathbf{w}$ to $\boldsymbol{\theta}$

until convergence

Berg-Kirkpatrick et al. (2010)

EM with Features

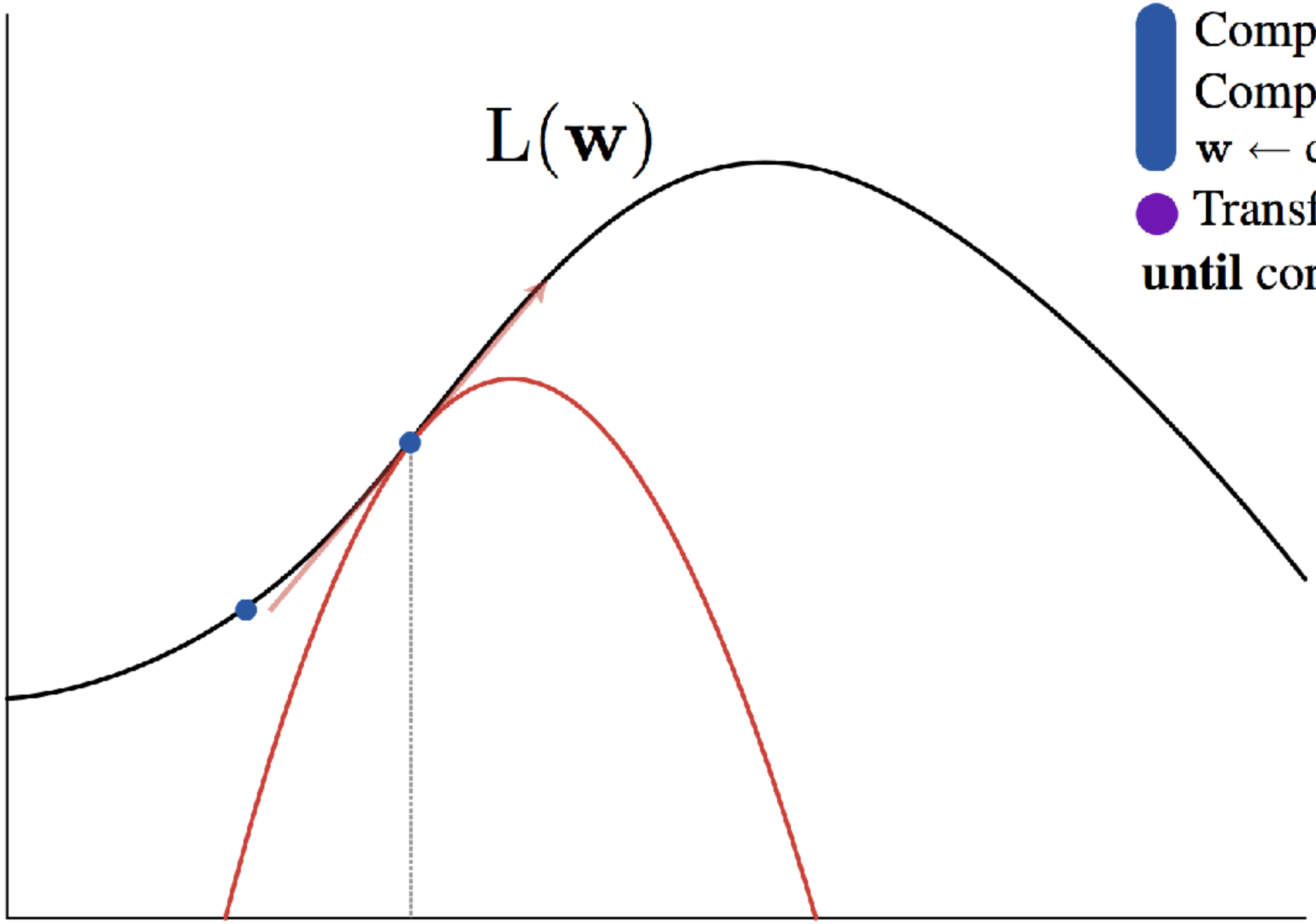


Initialize weights $\mathbf{w}$
repeat
- Compute expected counts $\mathbf{e}$
- Compute $L(\mathbf{w})$
- Compute $\nabla\ell(\mathbf{w}, \mathbf{e})$
- $\mathbf{w} \leftarrow \text{climb}(\mathbf{w}, L(\mathbf{w}), \nabla\ell(\mathbf{w}, \mathbf{e}))$
- Transform $\mathbf{w}$ to $\boldsymbol{\theta}$

until convergence

Berg-Kirkpatrick et al. (2010)

EM with Features



Initialize weights $\mathbf{w}$

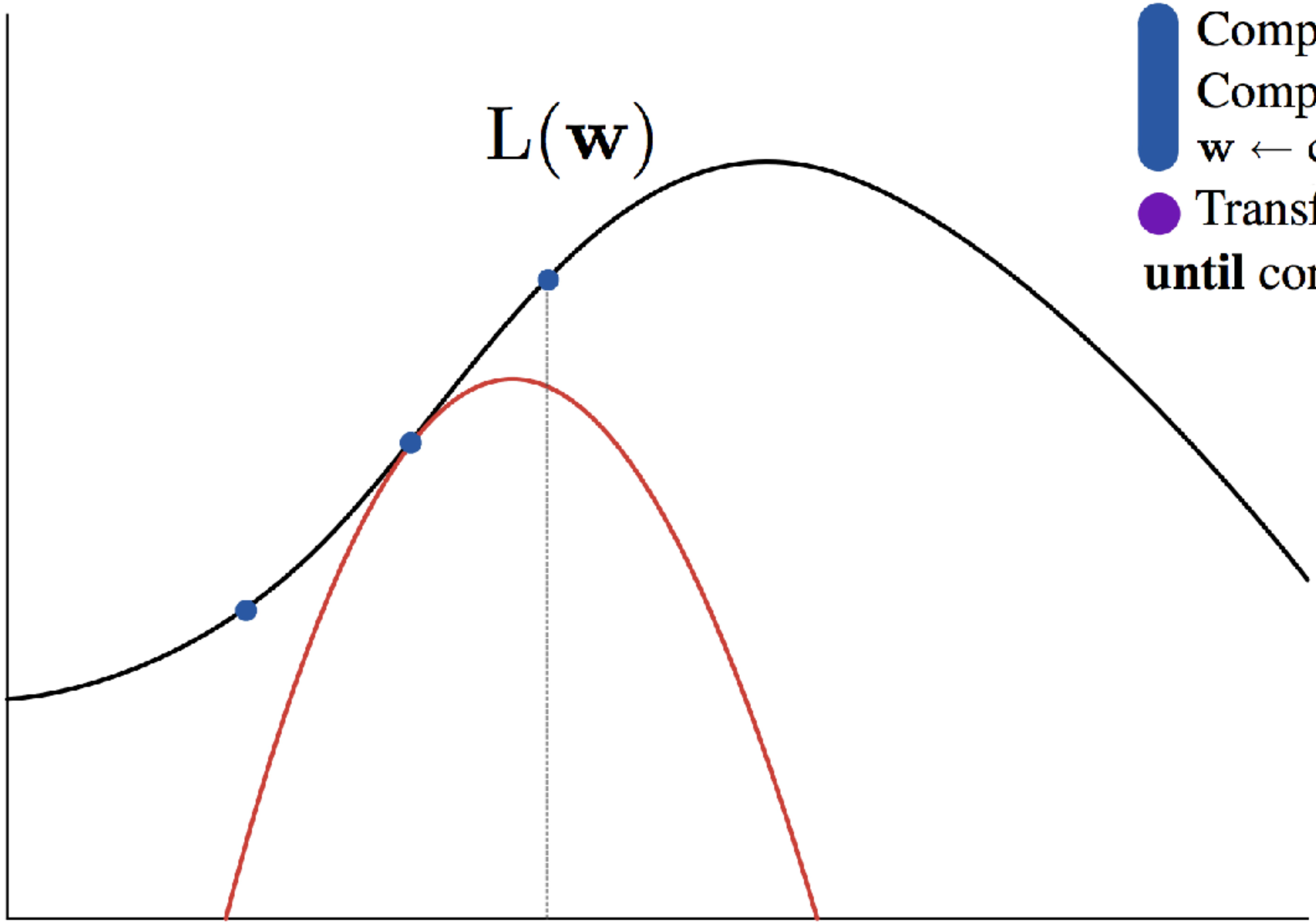
repeat

- Compute expected counts $\mathbf{e}$
- Compute $L(\mathbf{w})$
- Compute $\nabla \ell(\mathbf{w}, \mathbf{e})$
- $\mathbf{w} \leftarrow \text{climb}(\mathbf{w}, L(\mathbf{w}), \nabla \ell(\mathbf{w}, \mathbf{e}))$
- Transform $\mathbf{w}$ to $\boldsymbol{\theta}$

until convergence

Berg-Kirkpatrick et al. (2010)

EM with Features



Initialize weights $\mathbf{w}$

repeat

- Compute expected counts $\mathbf{e}$
- Compute $L(\mathbf{w})$
- Compute $\nabla \ell(\mathbf{w}, \mathbf{e})$
- $\mathbf{w} \leftarrow \text{climb}(\mathbf{w}, L(\mathbf{w}), \nabla \ell(\mathbf{w}, \mathbf{e}))$
- Transform $\mathbf{w}$ to $\boldsymbol{\theta}$

until convergence

Berg-Kirkpatrick et al. (2010)

EM with Features

- Faster approach: after the E-step, just take *one* gradient step
- "Direct gradient" on the marginal log likelihood $\log \sum_{\mathbf{y}} P(\mathbf{x}, \mathbf{y} | \theta)$
- Looks a lot like CRF training: compute marginals, estimate gradient based on them

Berg-Kirkpatrick et al. (2010)

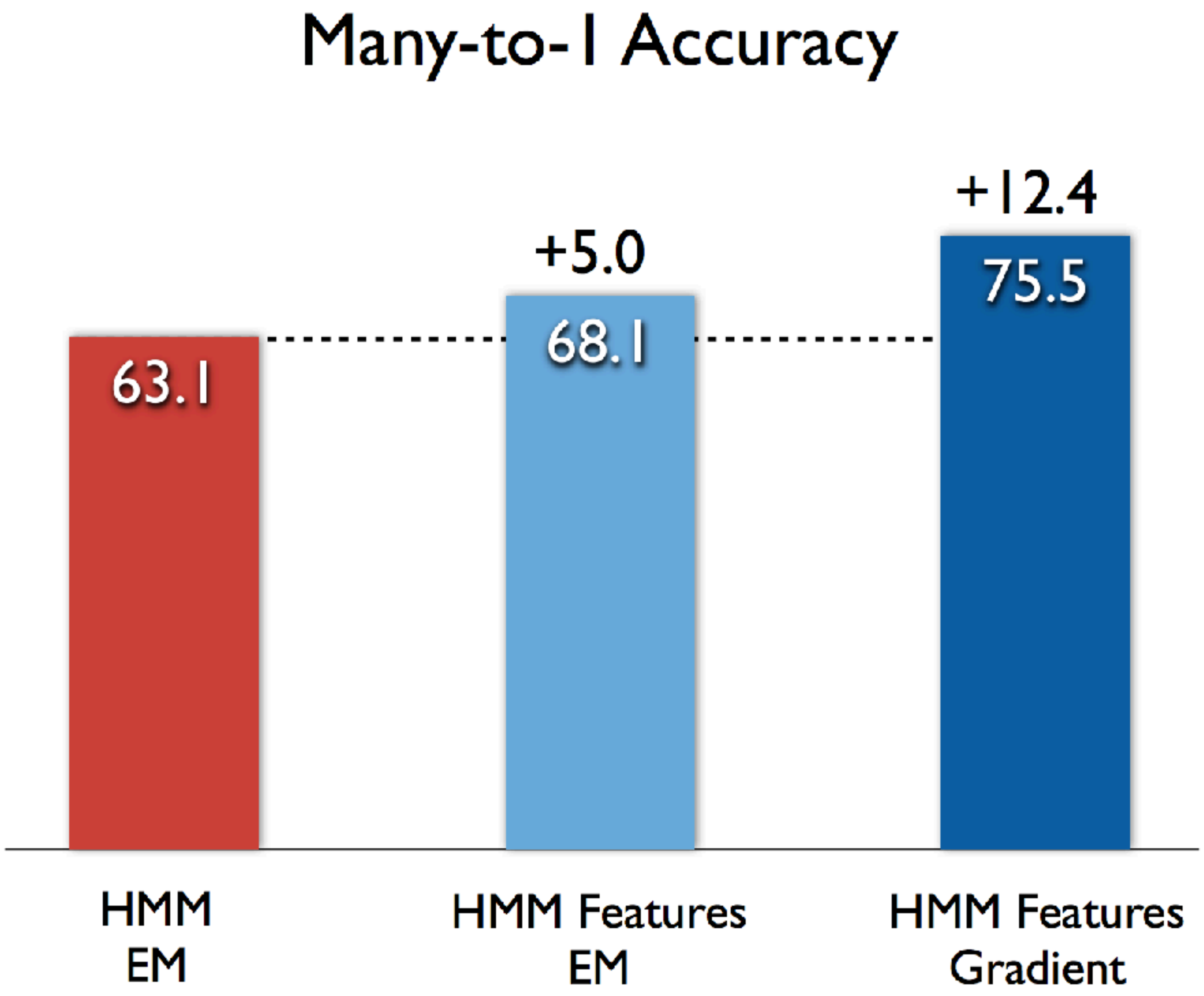
Evaluating Direct Gradient

- Different setup: don't assume any initialization, you just have 45 tags that you need to learn

- "Many-to-one" accuracy: map each of your learned tags to its closest gold tag, evaluate how many words are tagged correctly

Berg-Kirkpatrick et al. (2010)

Evaluating Direct Gradient



Features:

Basic:	John ∧ NNP
Contains-Digit:	+Digit ∧ NNP
Contains-Hyphen:	+Hyph ∧ NNP
Initial-Capital:	+Cap ∧ NNP
Suffix:	+ing ∧ NNP

- Also strong results on grammar induction, word alignment, and morphological segmentation

Takeaways

- EM sort of works for POS induction
- A supervised system on a little bit of labeled data gives better POS accuracy, but unsupervised learning can still learn useful representations for downstream tasks (like machine translation)
- EM isn’t restricted to multinomial distributions or ones with closed-form M-step updates: the M-step can be gradient ascent
- “Direct gradient” can work really well!

Next Time

- Constituency parsing
- In two lectures: dependency parsing (project 2)