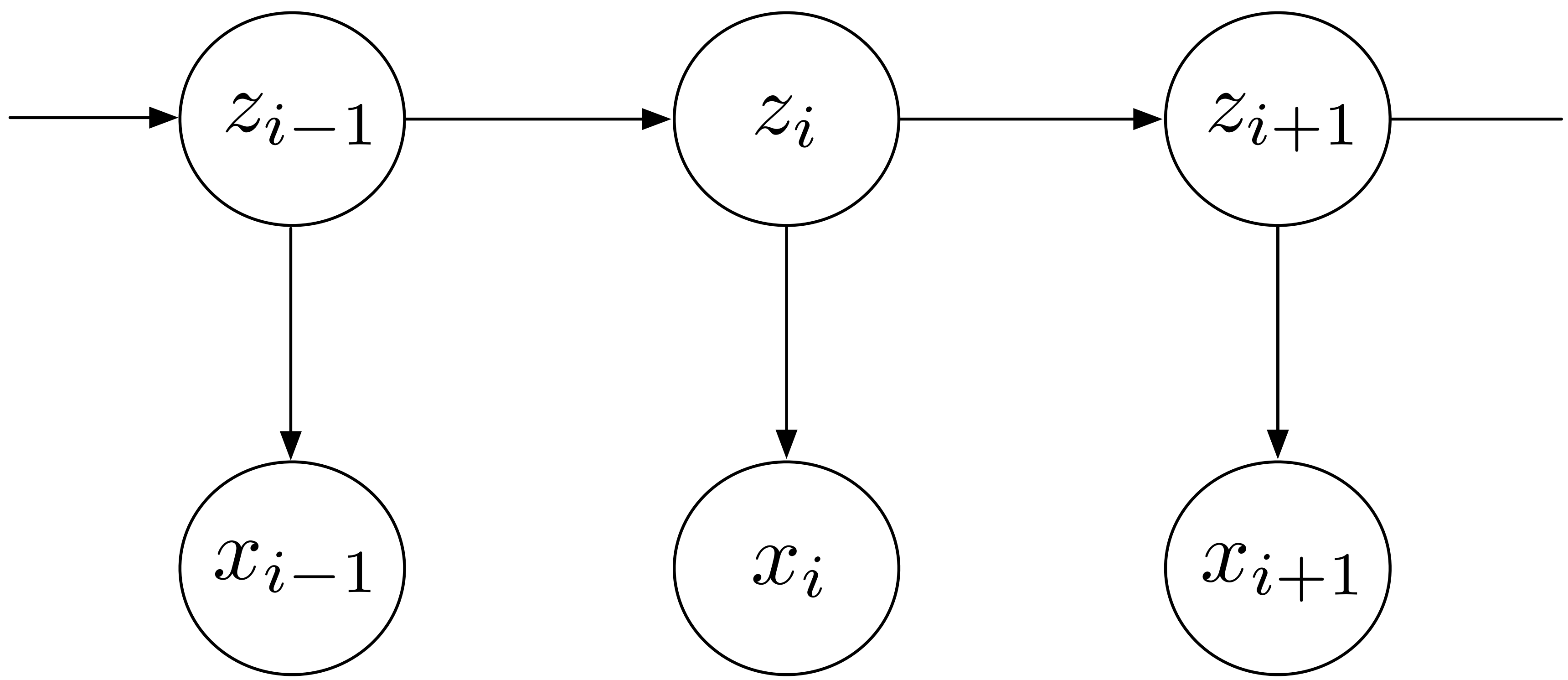
Painless Unsupervised Learning with Features

Berkeley NLP

Taylor Berg-Kirkpatrick   Alexandre Bouchard-Côté
John DeNero   Dan Klein

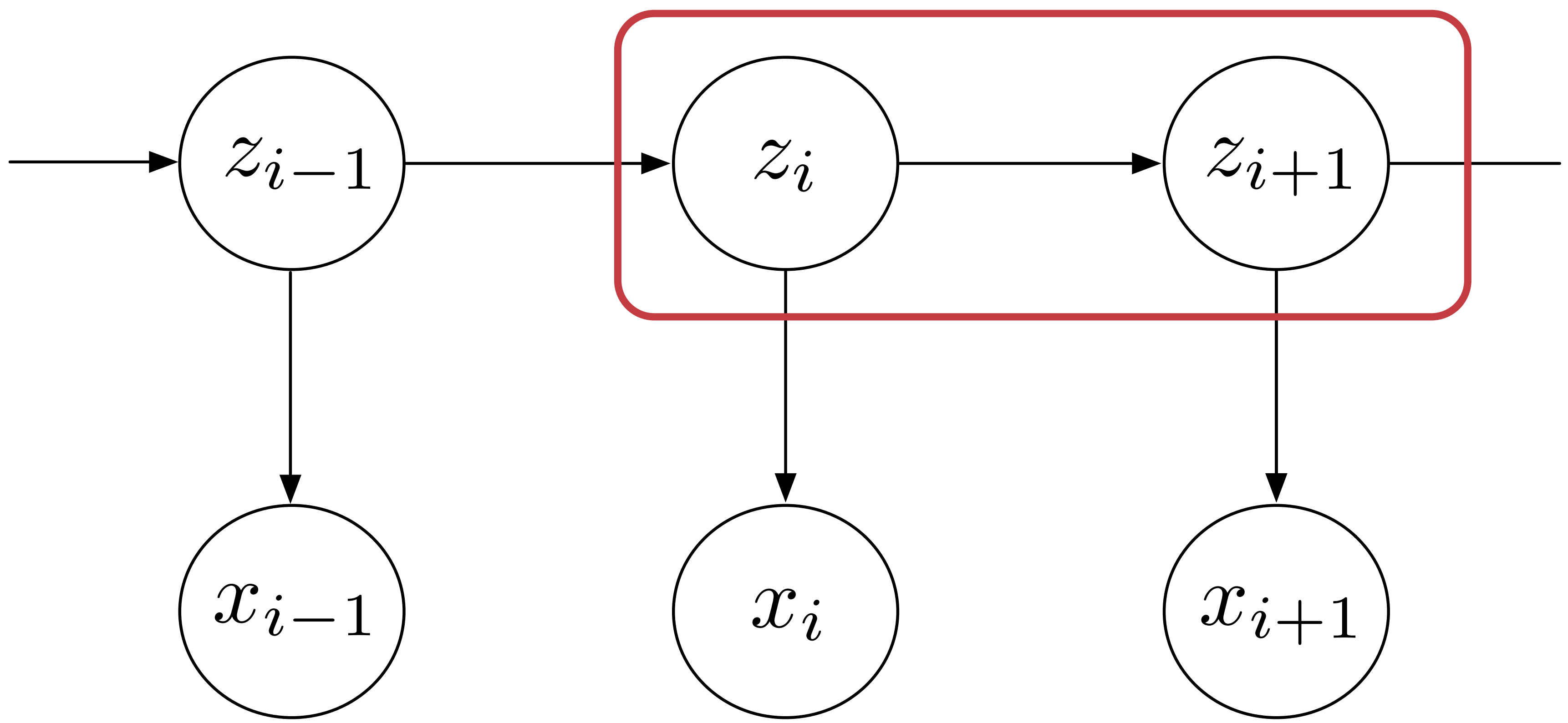
Basic HMM for POS Induction



Basic HMM for POS Induction

Transition distribution:

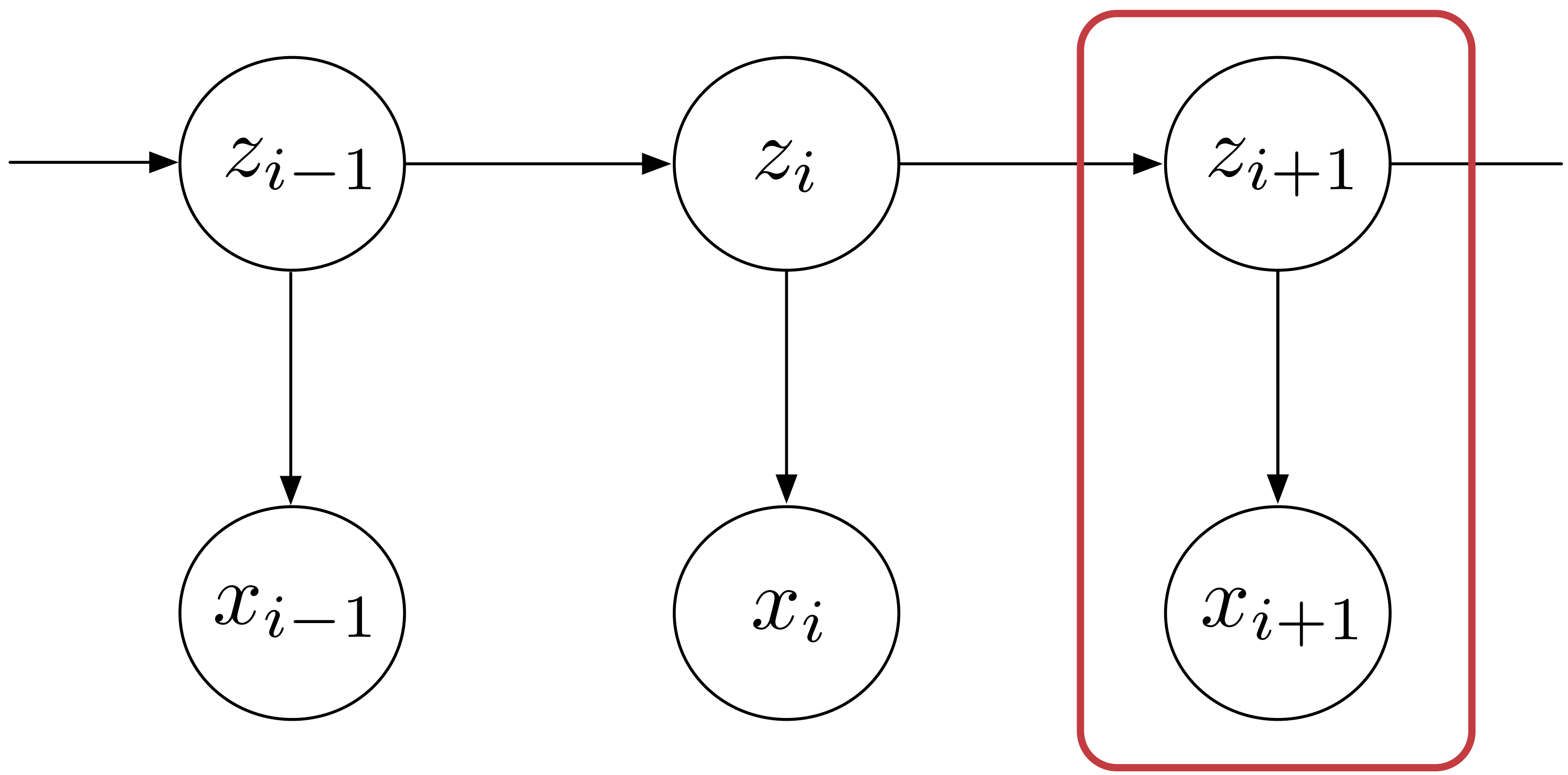
$$P(z'|z)$$



Basic HMM for POS Induction

Emission distribution:

$$P(x|z)$$



Parameterization

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

x
John
Mary
running
jumping

Parameterization

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

$\theta_{x\|\mathrm{NNP}}$	x
0.1	John
0.0	Mary
0.2	running
0.0	jumping

Parameterization

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

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

Parameterization

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

Berkeley NLP

Parameterization

$$\mathbf{w} \dashrightarrow \theta$$

$$\theta_{x|z} = \frac{\exp(\mathbf{w}^\top \mathbf{f}(x, z))}{\sum_{x'} \exp(\mathbf{w}^\top \mathbf{f}(x', z))}$$

Unsupervised Learning with Features

Main idea: local multinomials become maxents

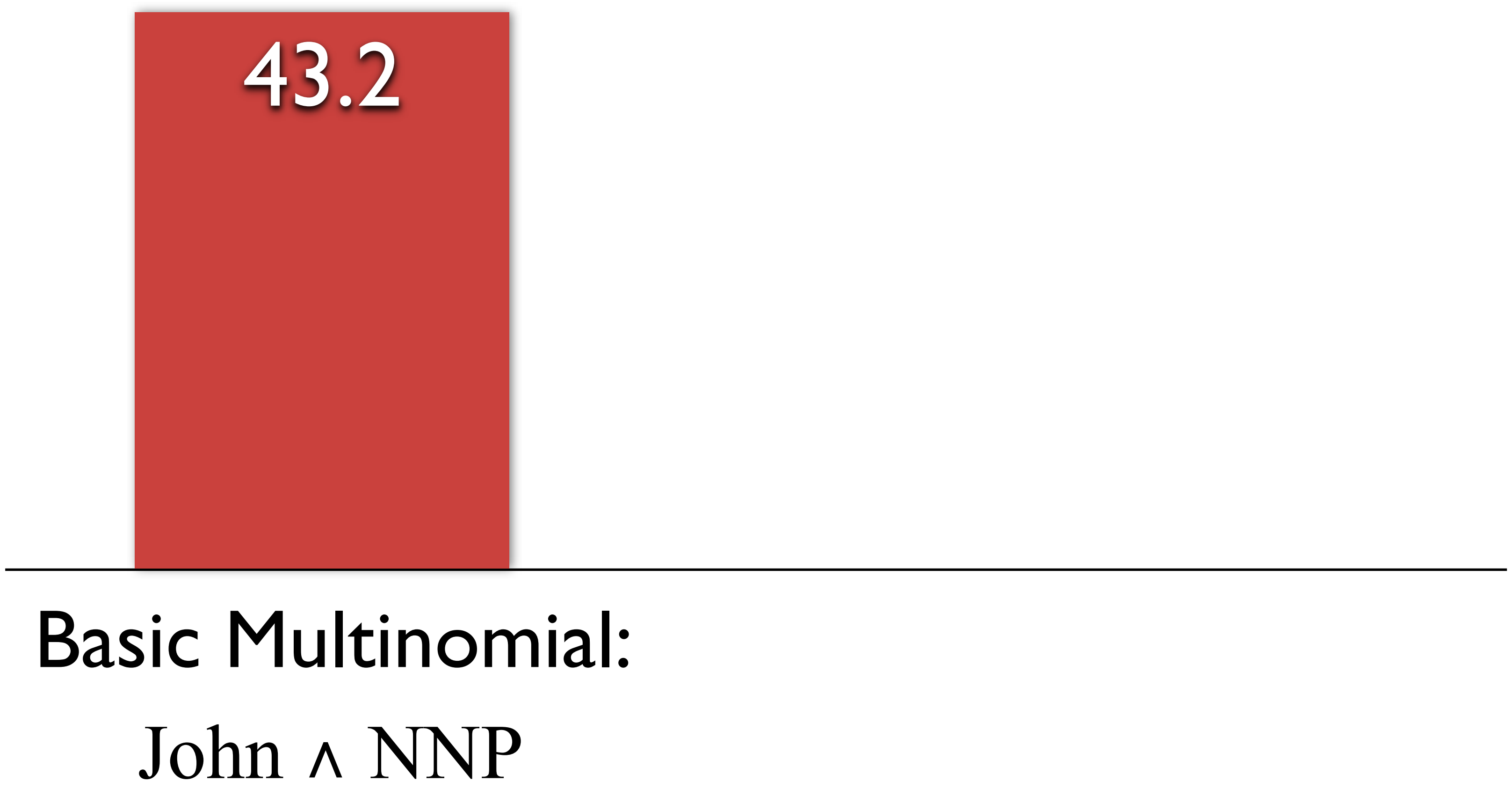
Unsupervised Learning with Features

Main idea: local multinomials become maxents

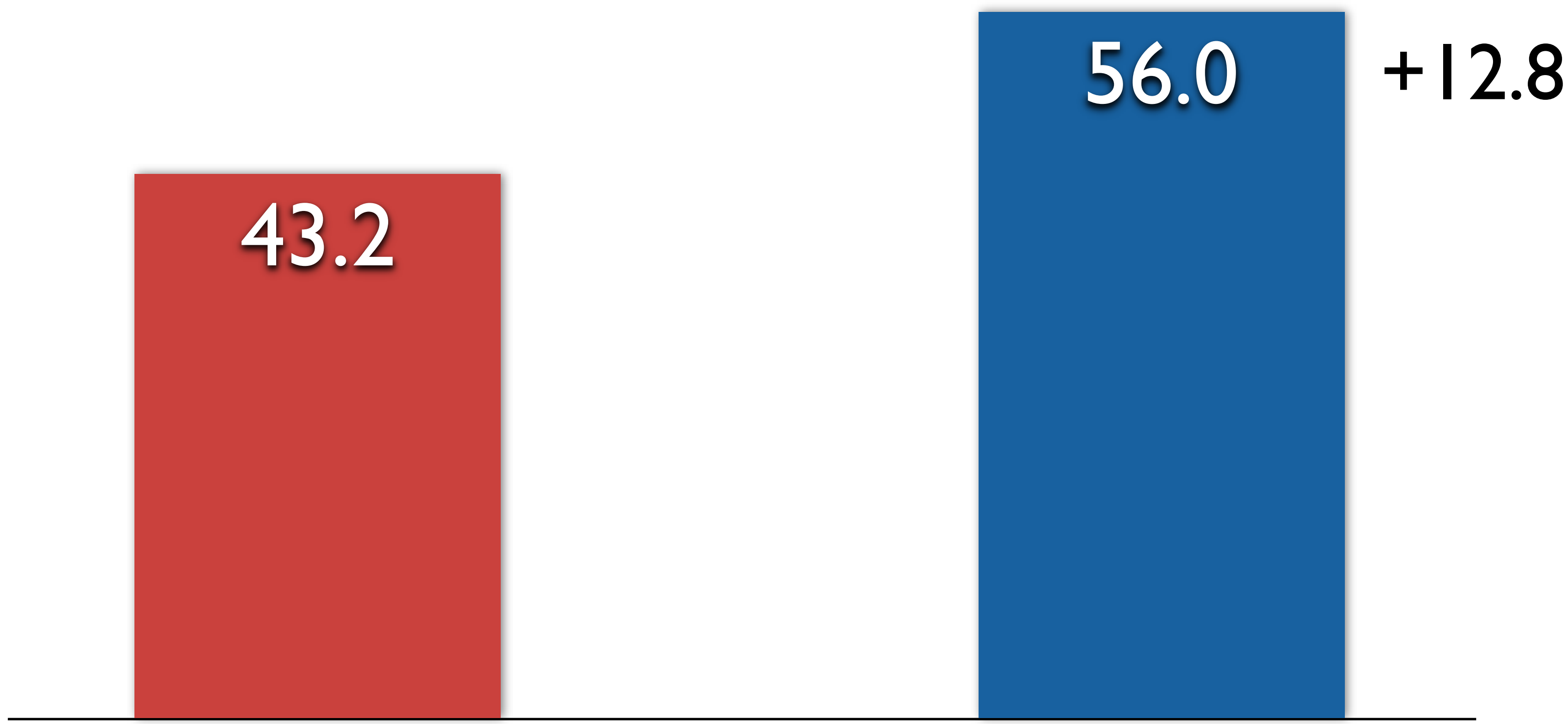
EM + Maxent M-Step =
Unsupervised learning w/ features

POS Induction Accuracy

POS Induction Accuracy



POS Induction Accuracy



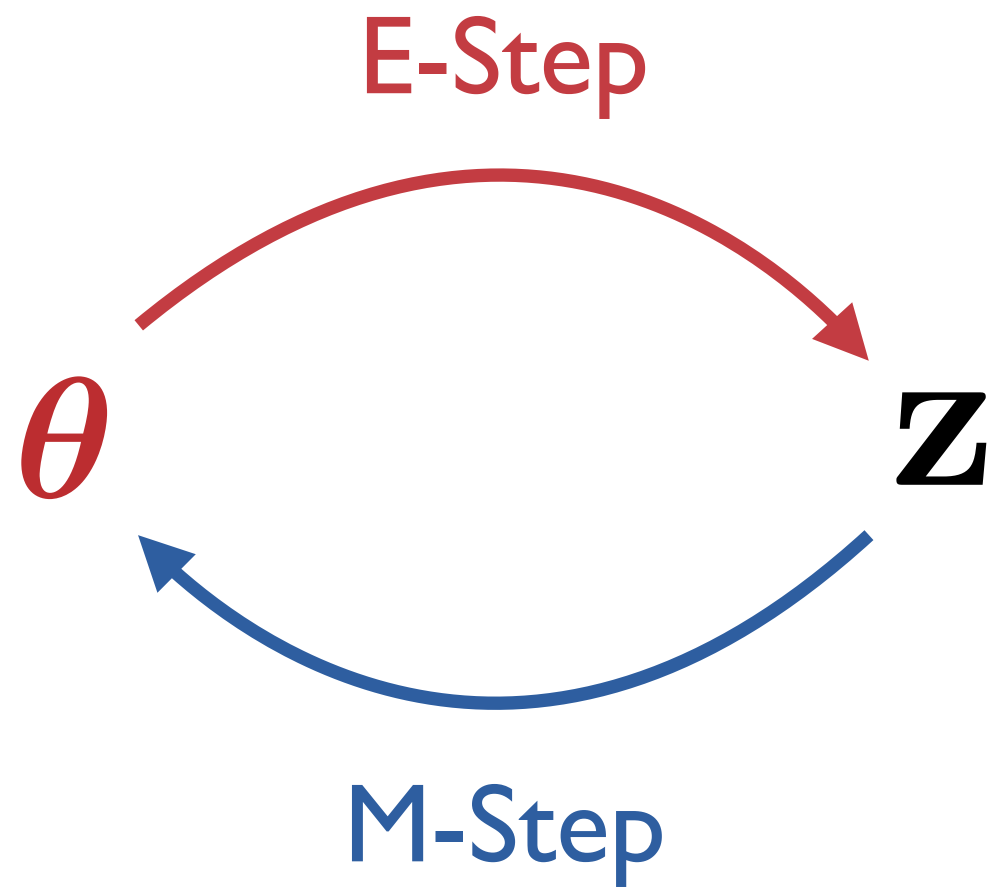
Basic Multinomial:

John ∧ NNP

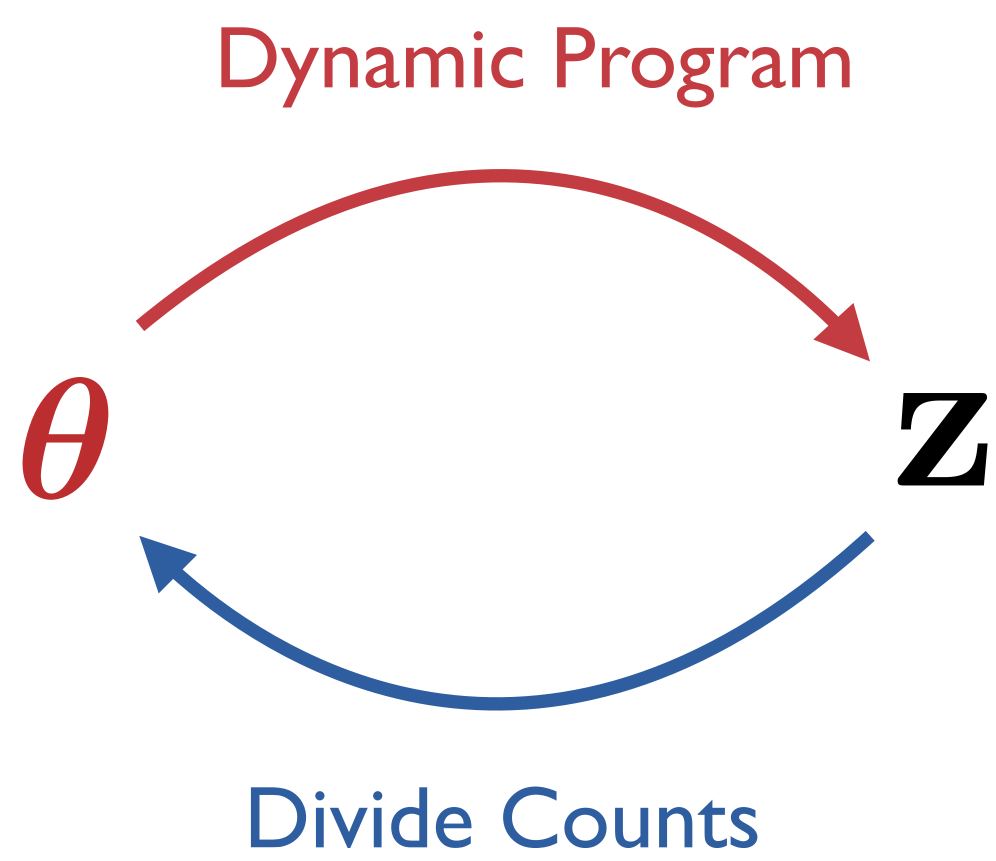
Rich Features:

John ∧ NNP
+Digit ∧ NNP
+Hyph ∧ NNP
+Cap ∧ NNP
+ing ∧ NNP

Hard EM without Features



Hard EM without Features



E-Step: Dynamic Program

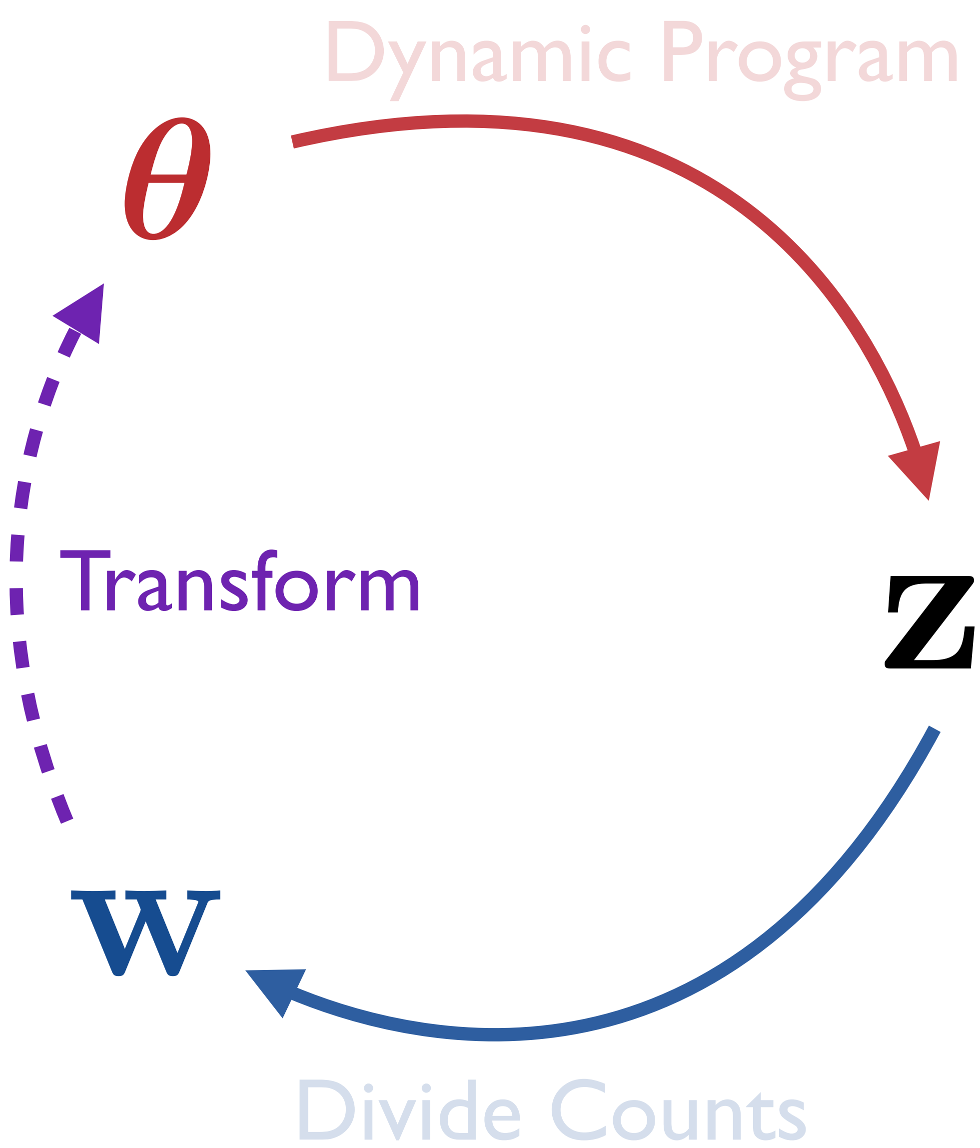
$$\mathbf{z} \leftarrow \operatorname*{argmax}_{\mathbf{z}} P(\mathbf{z} | \mathbf{x}; \boldsymbol{\theta})$$

M-Step: Divide Counts

$$\boldsymbol{\theta} \leftarrow \operatorname*{argmax}_{\boldsymbol{\theta}} P(\mathbf{x}, \mathbf{z}; \boldsymbol{\theta})$$

$$= \left[\frac{c(z \rightarrow x)}{c(z \rightarrow \cdot)}, \ \ldots \right]$$

Hard EM with Features



E-Step: Dynamic Program

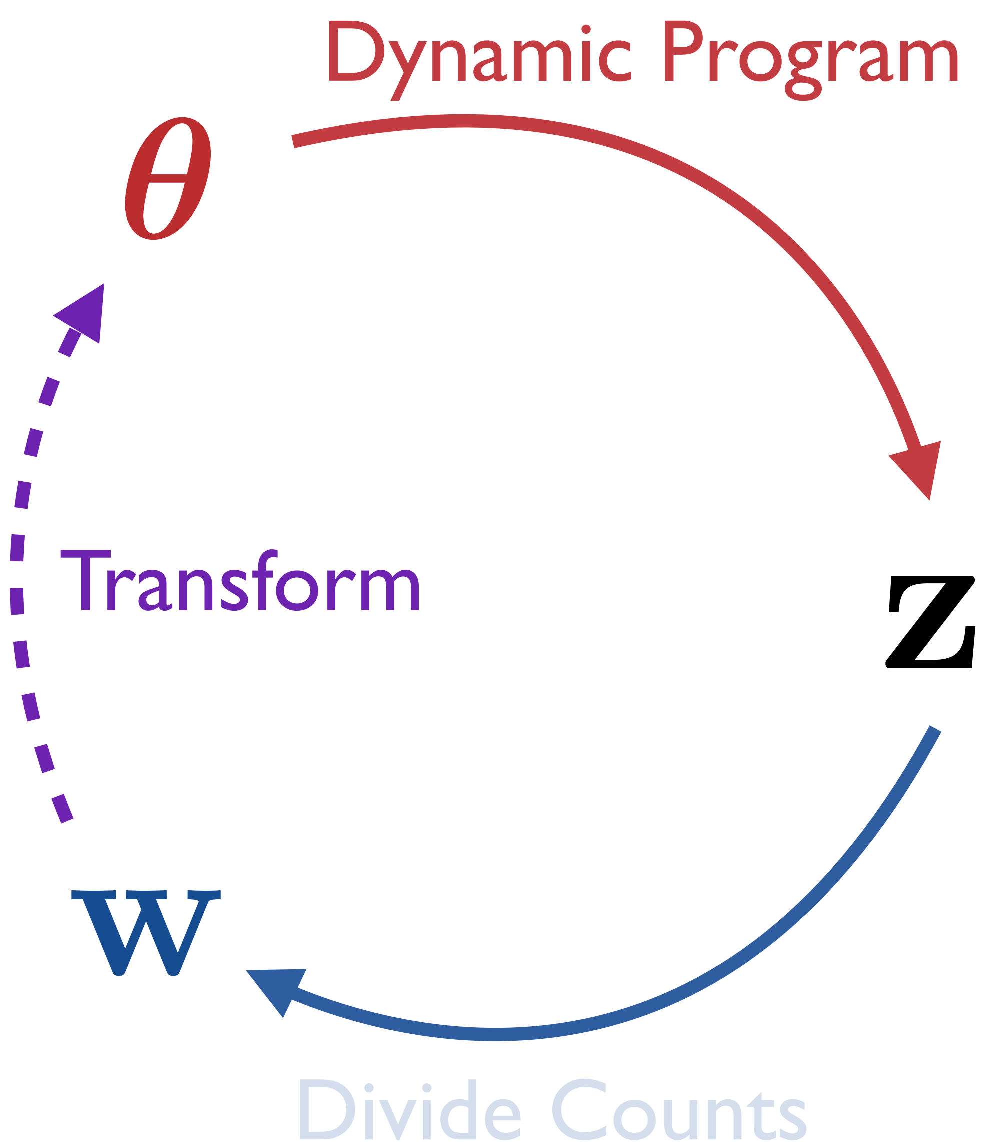
$$\mathbf{z} \leftarrow \underset{\mathbf{z}}{\operatorname{argmax}}\ P(\mathbf{z}|\mathbf{x};\boldsymbol{\theta})$$

M-Step: Divide Counts

$$\boldsymbol{\theta} \leftarrow \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\ P(\mathbf{x},\mathbf{z};\boldsymbol{\theta})$$

$$= \left[\frac{c(z \to x)}{c(z \to \cdot)}, \ \cdots\right]$$

Hard EM with Features



E-Step: Dynamic Program

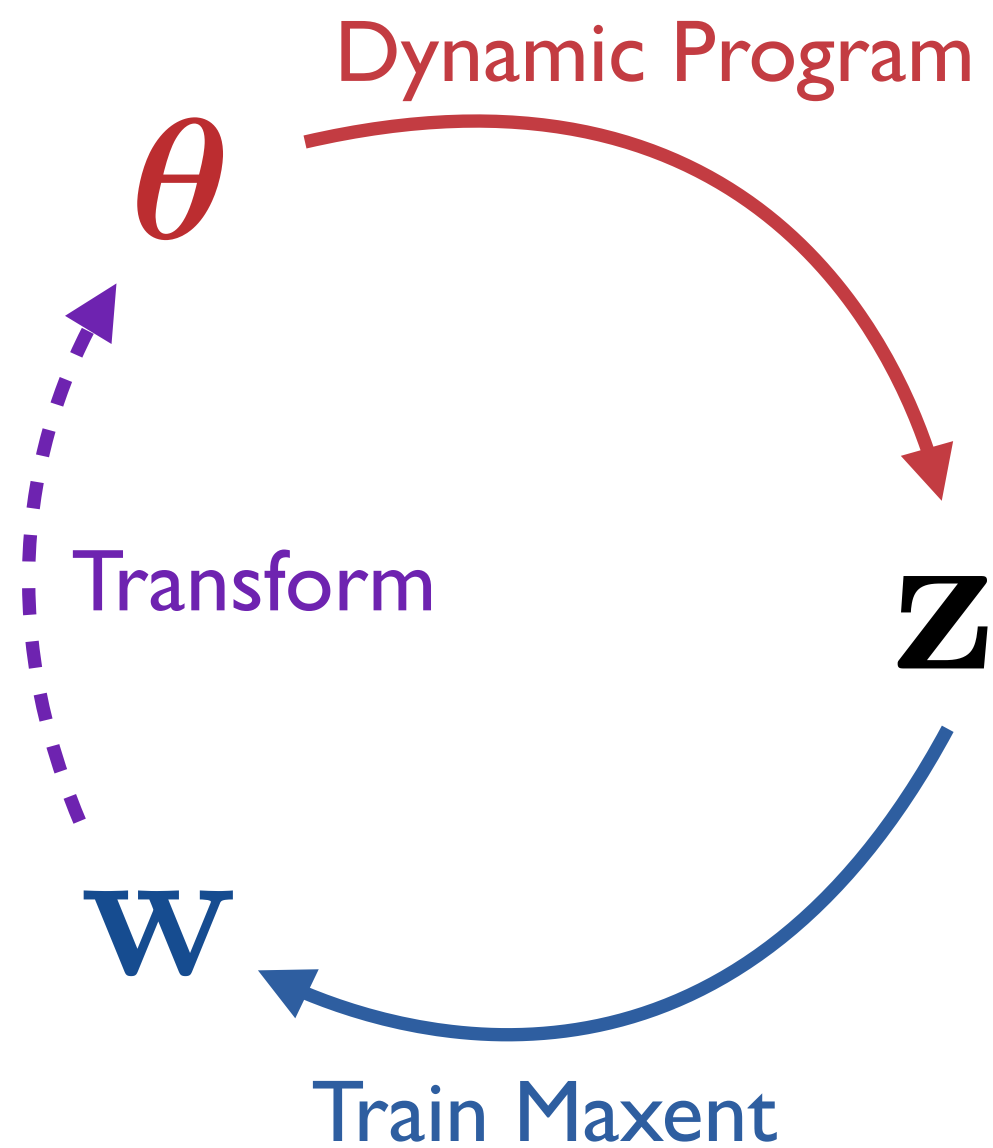
$$\mathbf{z} \leftarrow \operatorname*{argmax}_{\mathbf{z}} P(\mathbf{z}|\mathbf{x};\boldsymbol{\theta})$$

M-Step: Divide Counts

$$\boldsymbol{\theta} \leftarrow \operatorname*{argmax}_{\boldsymbol{\theta}} P(\mathbf{x},\mathbf{z};\boldsymbol{\theta})$$

$$= \left[\frac{c(z \rightarrow x)}{c(z \rightarrow \cdot)}, \ \ldots\right]$$

Hard EM with Features



<u>E-Step: Dynamic Program</u>

$$\mathbf{z} \leftarrow \underset{\mathbf{z}}{\operatorname{argmax}} \; P(\mathbf{z}|\mathbf{x};\boldsymbol{\theta})$$

<u>M-Step: Train Maxent</u>

$$\mathbf{w} \leftarrow \underset{\mathbf{w}}{\operatorname{argmax}} \log P(\mathbf{x},\mathbf{z};\mathbf{w})$$

Hard EM with Features

$$\log P(\mathbf{x}, \mathbf{z}; \mathbf{w})$$

$$= \sum_i \log P(x_i | z_i; \mathbf{w}) \; + \; \ldots$$

Hard EM with Features

$$\log P(\mathbf{x}, \mathbf{z}; \mathbf{w})$$

$$= \sum_i \log P(x_i | z_i; \mathbf{w}) + \ ...$$

Maxent training example

Hard EM with Features

$$\log P(\mathbf{x}, \mathbf{z}; \mathbf{w})$$

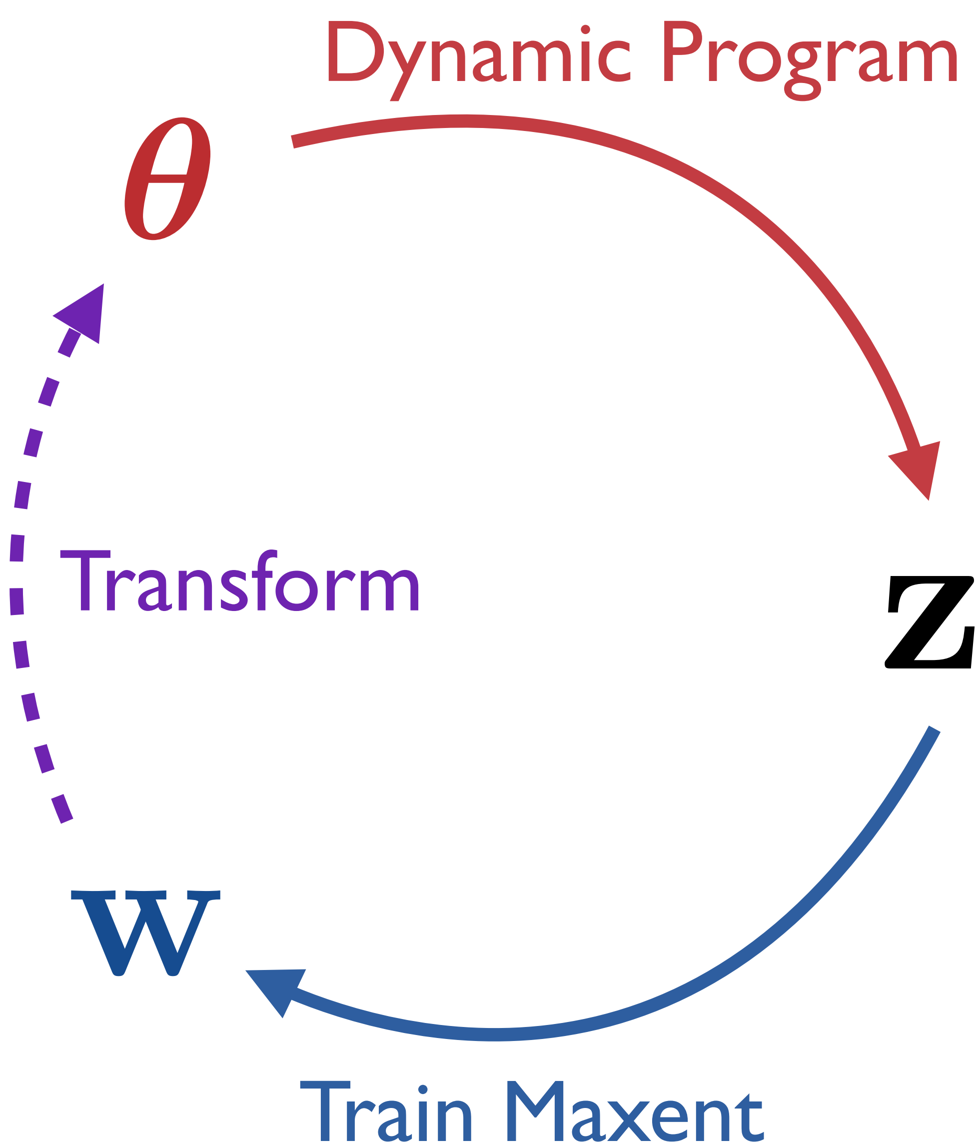
$$= \sum_i \log P(x_i | z_i; \mathbf{w}) + \ldots$$

Maxent training example

$$= \sum_{z,x} c(z \rightarrow x) \log P(x | z; \mathbf{w}) + \ldots$$

Multiplicity

Hard EM with Features



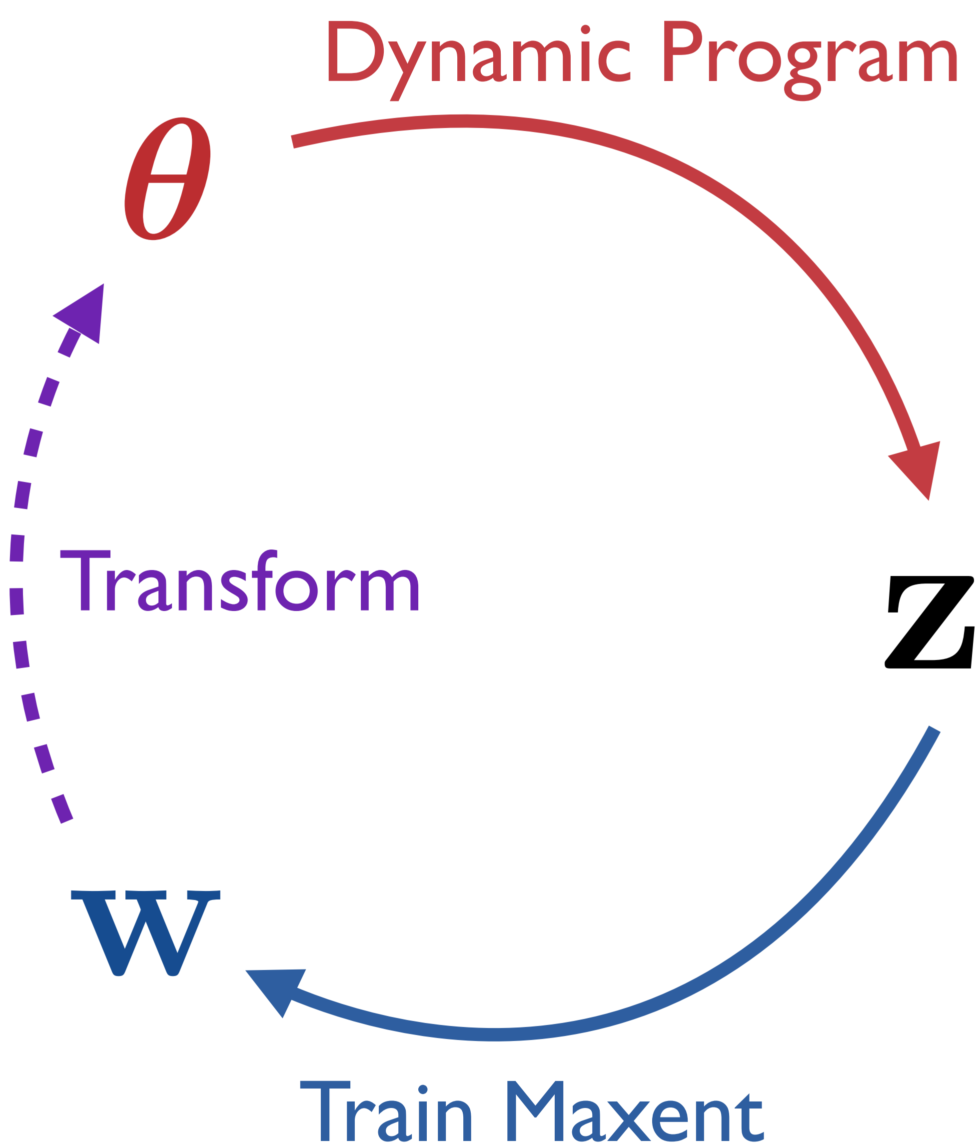
E-Step: Dynamic Program

$$\mathbf{z} \leftarrow \underset{\mathbf{z}}{\operatorname{argmax}}\ P(\mathbf{z}|\mathbf{x};\boldsymbol{\theta})$$

M-Step: Train Maxent

$$\mathbf{w} \leftarrow \underset{\mathbf{w}}{\operatorname{argmax}} \log P(\mathbf{x},\mathbf{z};\mathbf{w})$$

Hard EM with Features



E-Step: Dynamic Program

$$\mathbf{z} \leftarrow \underset{\mathbf{z}}{\operatorname{argmax}} \; P(\mathbf{z}|\mathbf{x};\boldsymbol{\theta})$$

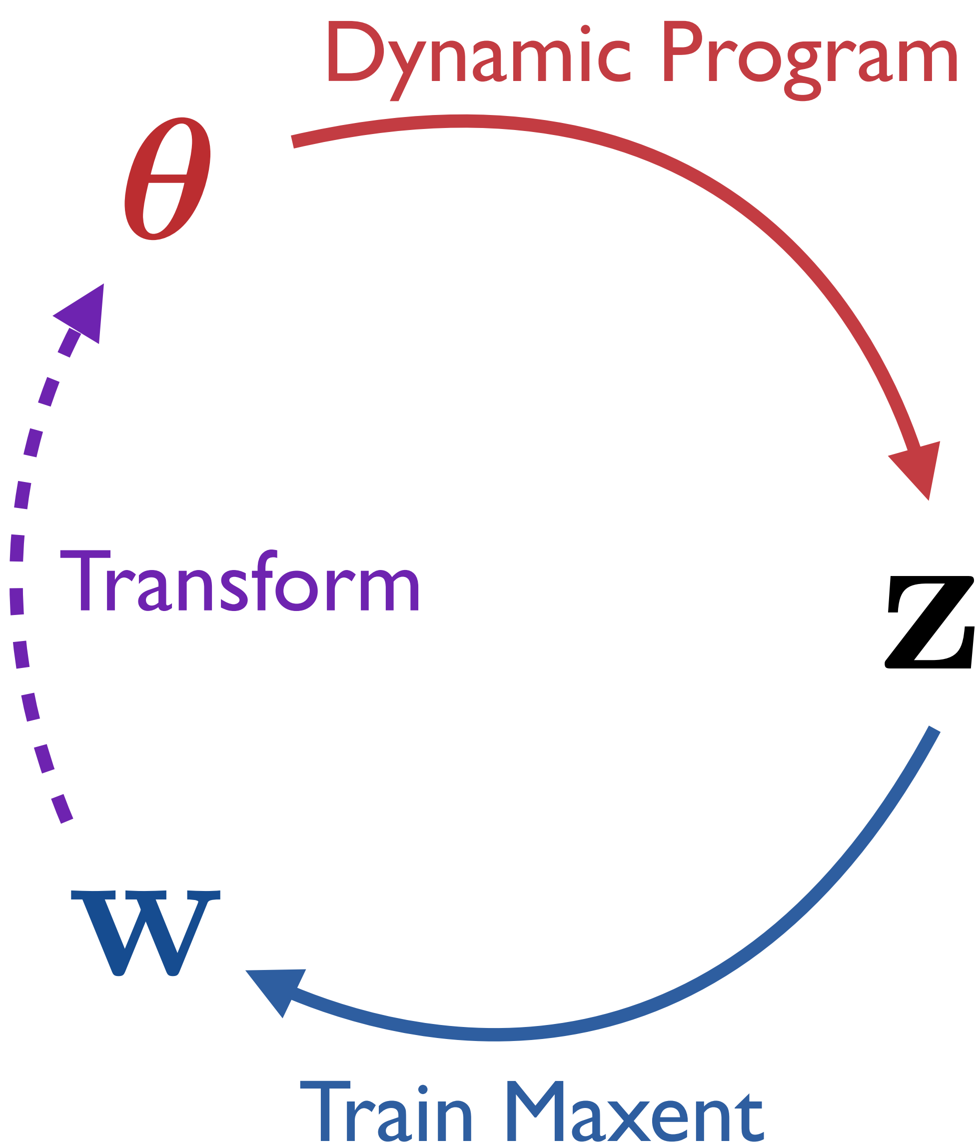
M-Step: Train Maxent

$$\mathbf{w} \leftarrow \underset{\mathbf{w}}{\operatorname{argmax}} \log P(\mathbf{x},\mathbf{z};\mathbf{w})$$

Transform Parameters

$$\theta_{x|z} \leftarrow \frac{\exp(\mathbf{w}^T \mathbf{f}(x,z))}{\sum_{x'} \exp(\mathbf{w}^T \mathbf{f}(x',z))}$$

EM with Features



E-Step: Dynamic Program

$$\mathbf{z} \leftarrow \arg\max_{\mathbf{z}} P(\mathbf{z}|\mathbf{x}; \boldsymbol{\theta})$$

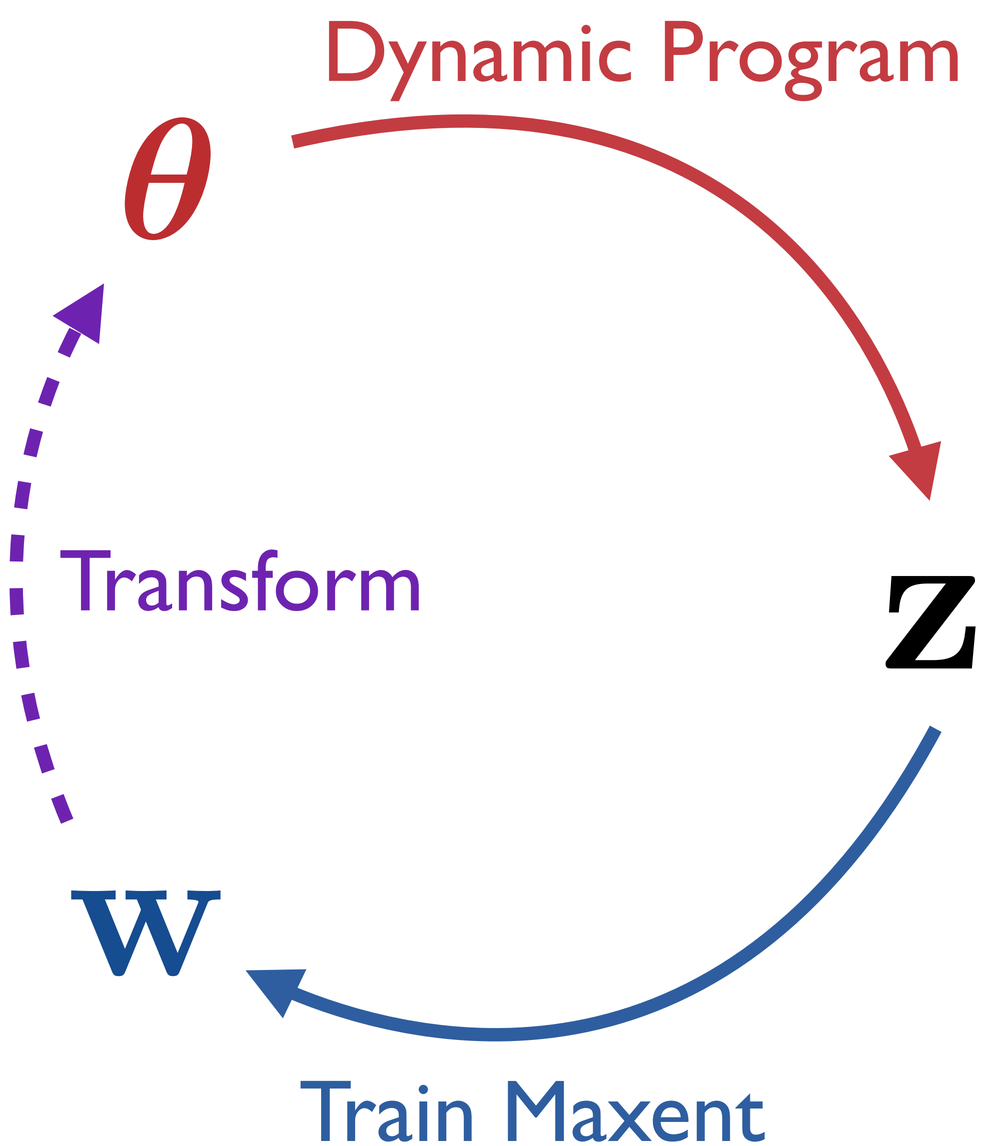
M-Step: Train Maxent

$$\mathbf{w} \leftarrow \arg\max_{\mathbf{w}} \log P(\mathbf{x}, \mathbf{z}; \mathbf{w})$$

Transform Parameters

$$\theta_{x|z} \leftarrow \frac{\exp(\mathbf{w}^T \mathbf{f}(x, z))}{\sum_{x'} \exp(\mathbf{w}^T \mathbf{f}(x', z))}$$

EM with Features



E-Step: Dynamic Program

$$e(z \rightarrow x) \leftarrow \mathbb{E}\big[c(z \rightarrow x)\big]$$

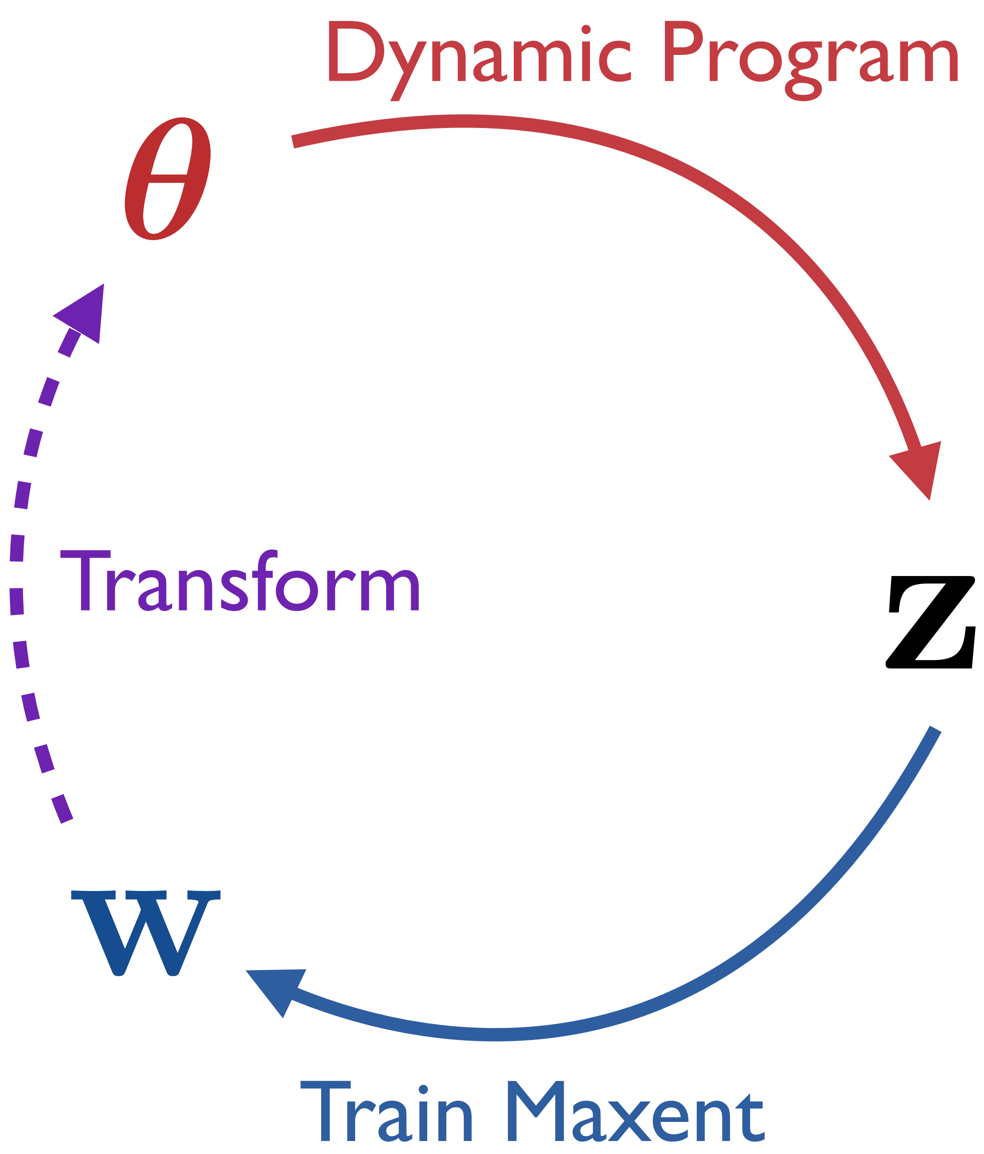
M-Step: Train Maxent

$$\mathbf{w} \leftarrow \underset{\mathbf{w}}{\operatorname{argmax}} \log P(\mathbf{x}, \mathbf{z}; \mathbf{w})$$

Transform Parameters

$$\theta_{x|z} \leftarrow \frac{\exp(\mathbf{w}^T \mathbf{f}(x, z))}{\sum_{x'} \exp(\mathbf{w}^T \mathbf{f}(x', z))}$$

EM with Features



E-Step: Dynamic Program

$$e(z \rightarrow x) \leftarrow \mathbb{E}\left[c(z \rightarrow x)\right]$$

M-Step: Train Maxent

$$\mathbf{w} \leftarrow \operatorname*{argmax}_{\mathbf{w}} \mathbb{E}\left[\log P(\mathbf{x}, \mathbf{z}; \mathbf{w})\right]$$

Transform Parameters

$$\theta_{x|z} \leftarrow \frac{\exp(\mathbf{w}^T \mathbf{f}(x, z))}{\sum_{x'} \exp(\mathbf{w}^T \mathbf{f}(x', z))}$$

EM without Features

Initialize probabilities $\boldsymbol{\theta}$

EM

repeat

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

until convergence

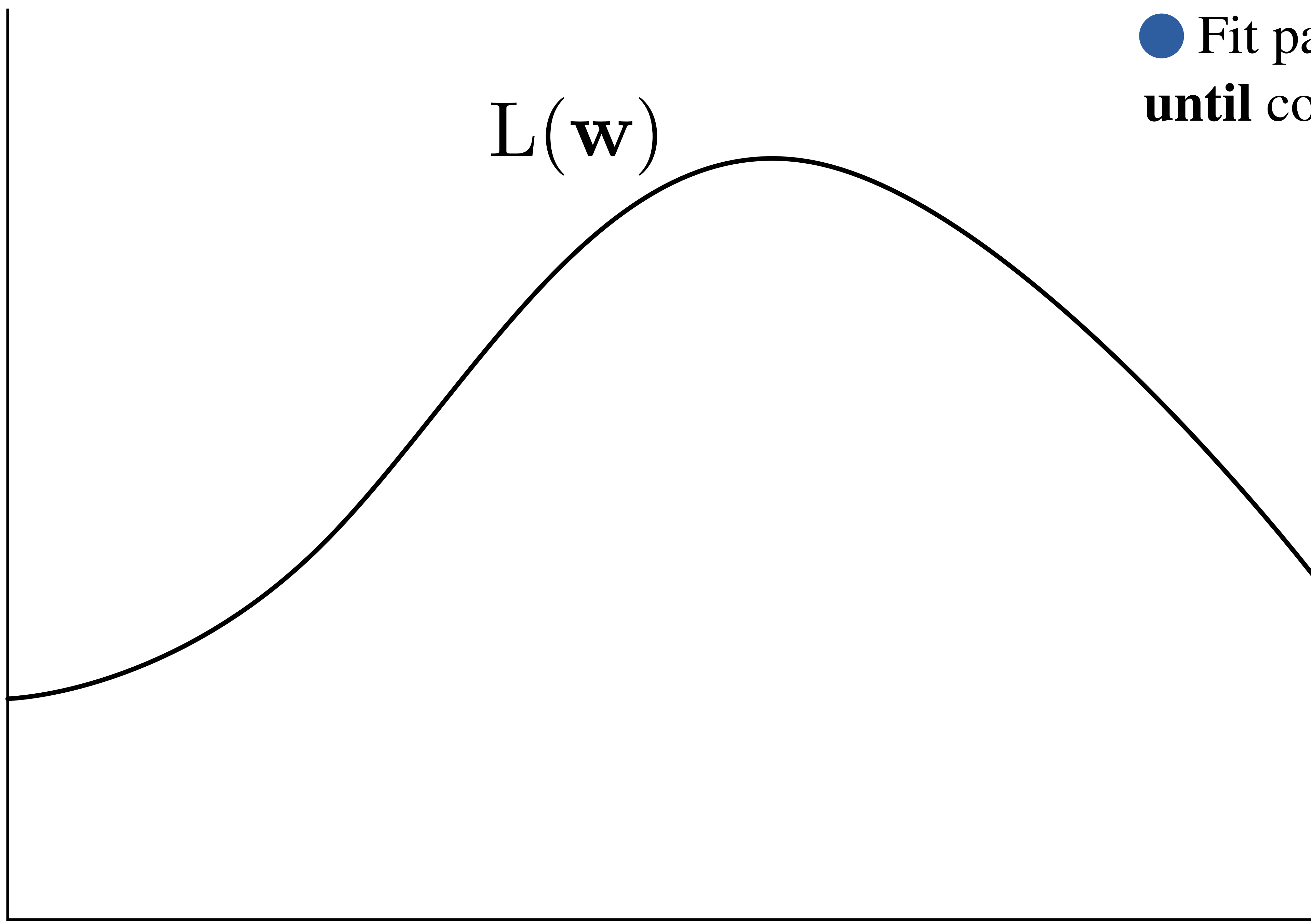
EM without Features

Initialize probabilities $\boldsymbol{\theta}$

repeat

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

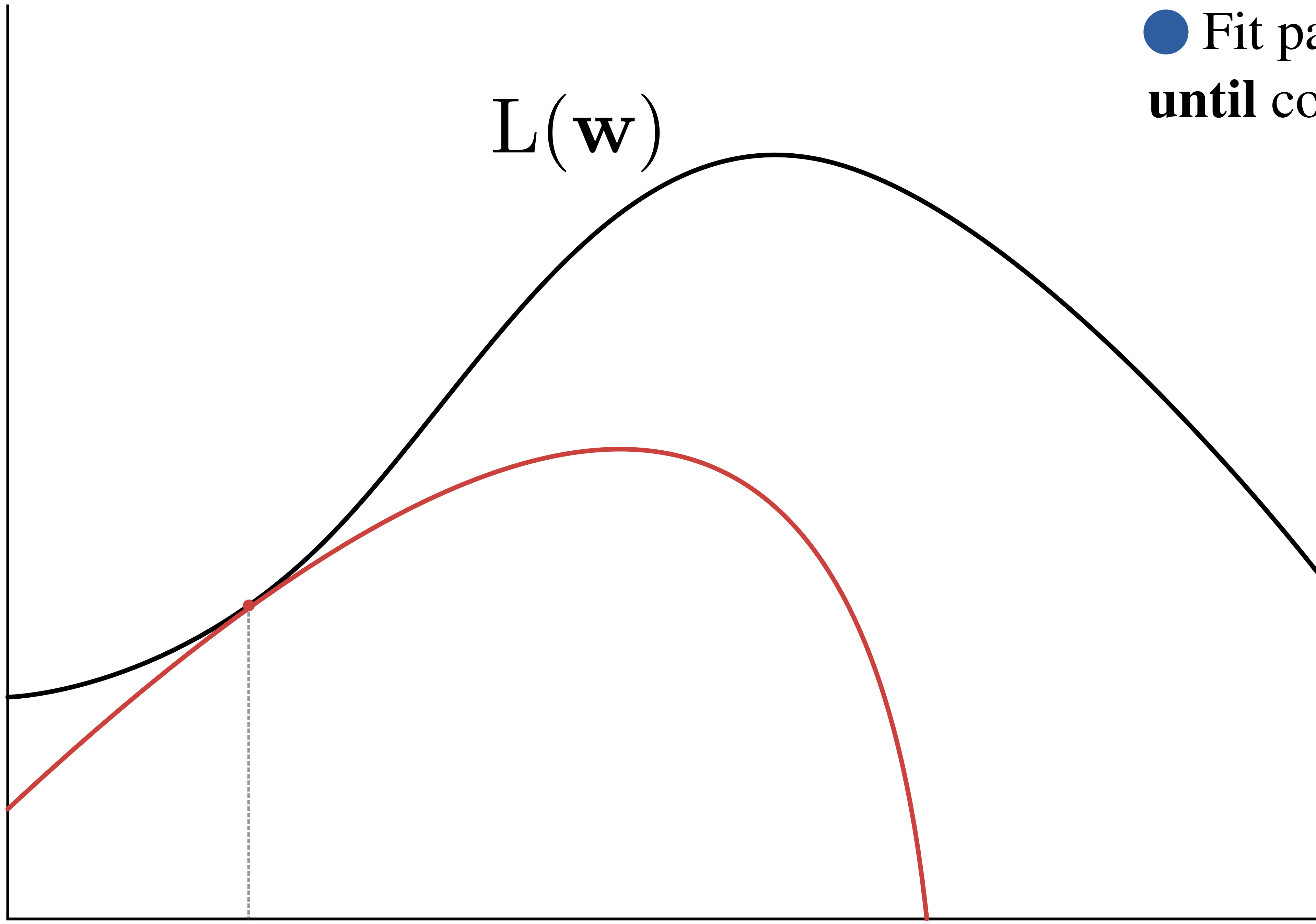
until convergence



EM without Features

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

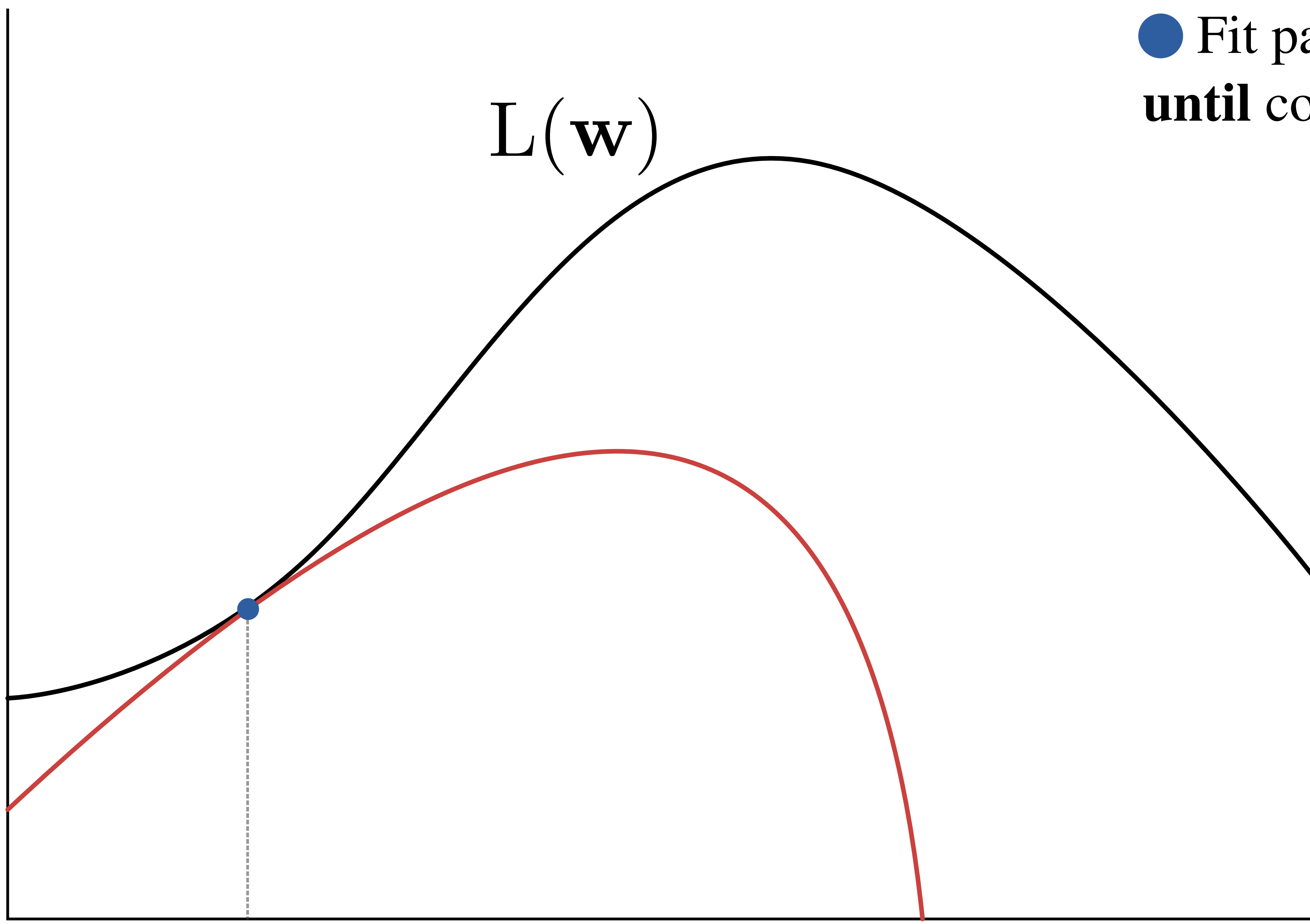
until convergence



EM without Features

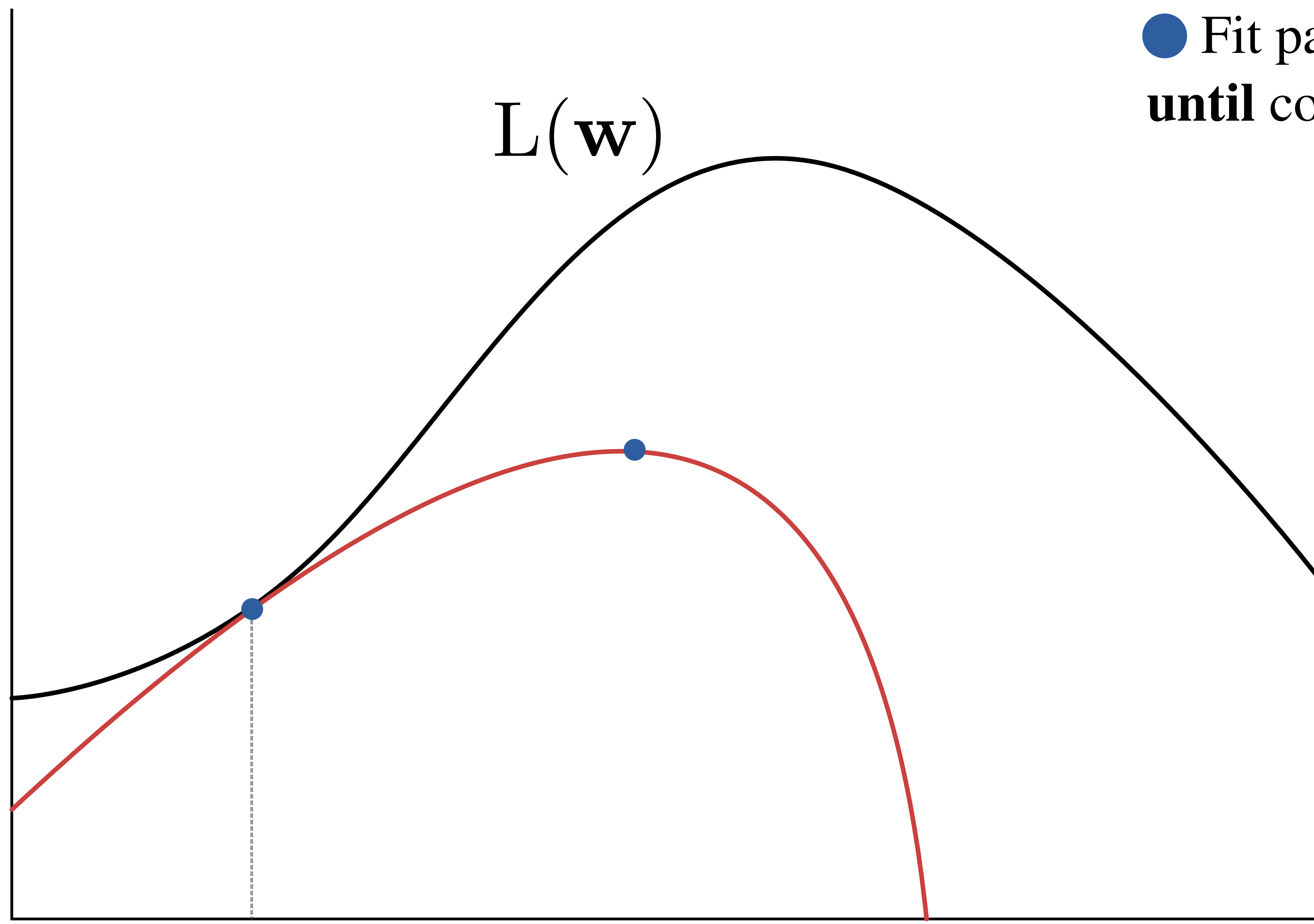
Initialize probabilities $\boldsymbol{\theta}$
repeat
- Compute expected counts $\mathbf{e}$
- Fit parameters $\boldsymbol{\theta}$

until convergence



EM without Features

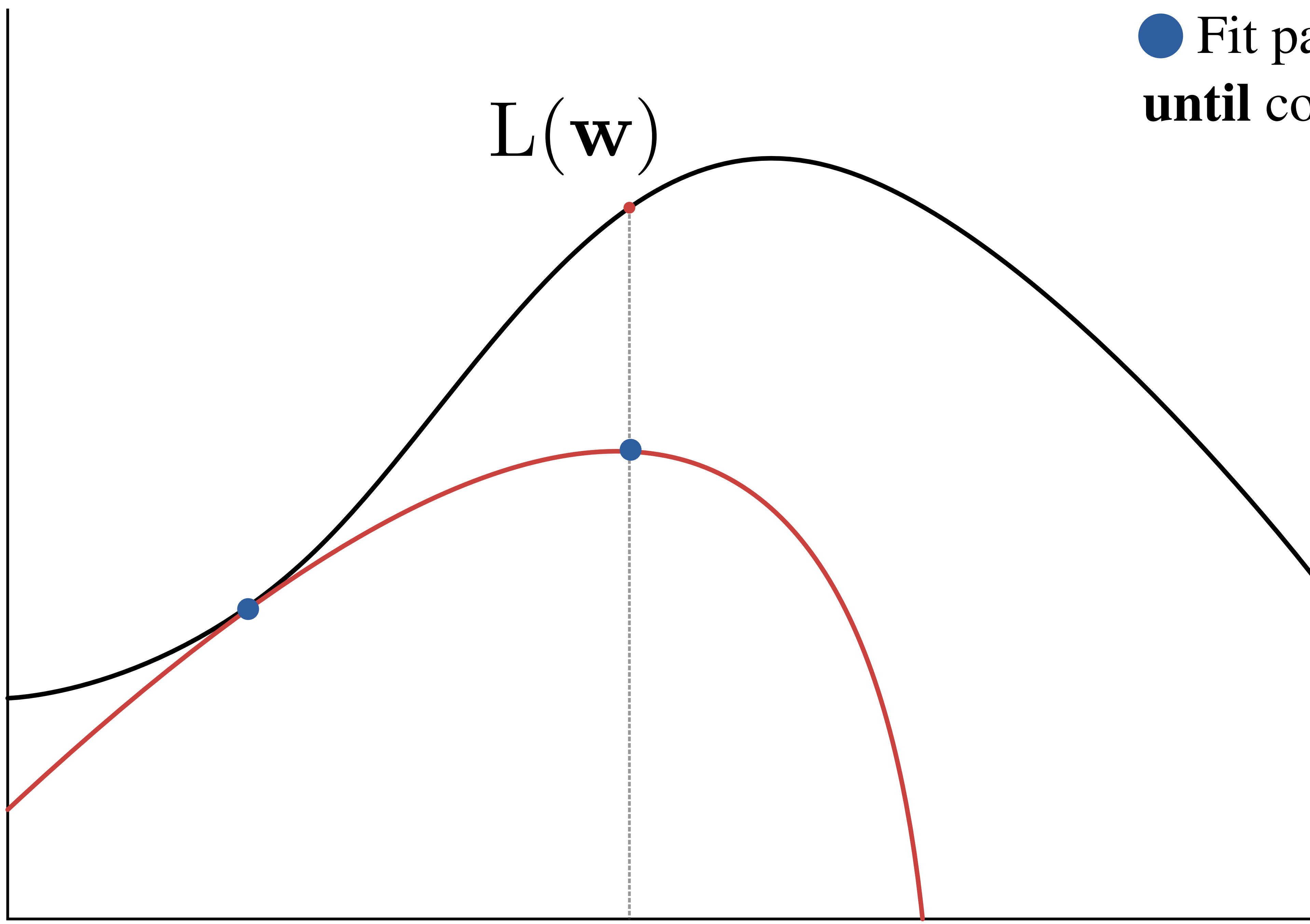
Initialize probabilities $\boldsymbol{\theta}$
repeat
Compute expected counts $\mathbf{e}$
Fit parameters $\boldsymbol{\theta}$
until convergence



EM without Features

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

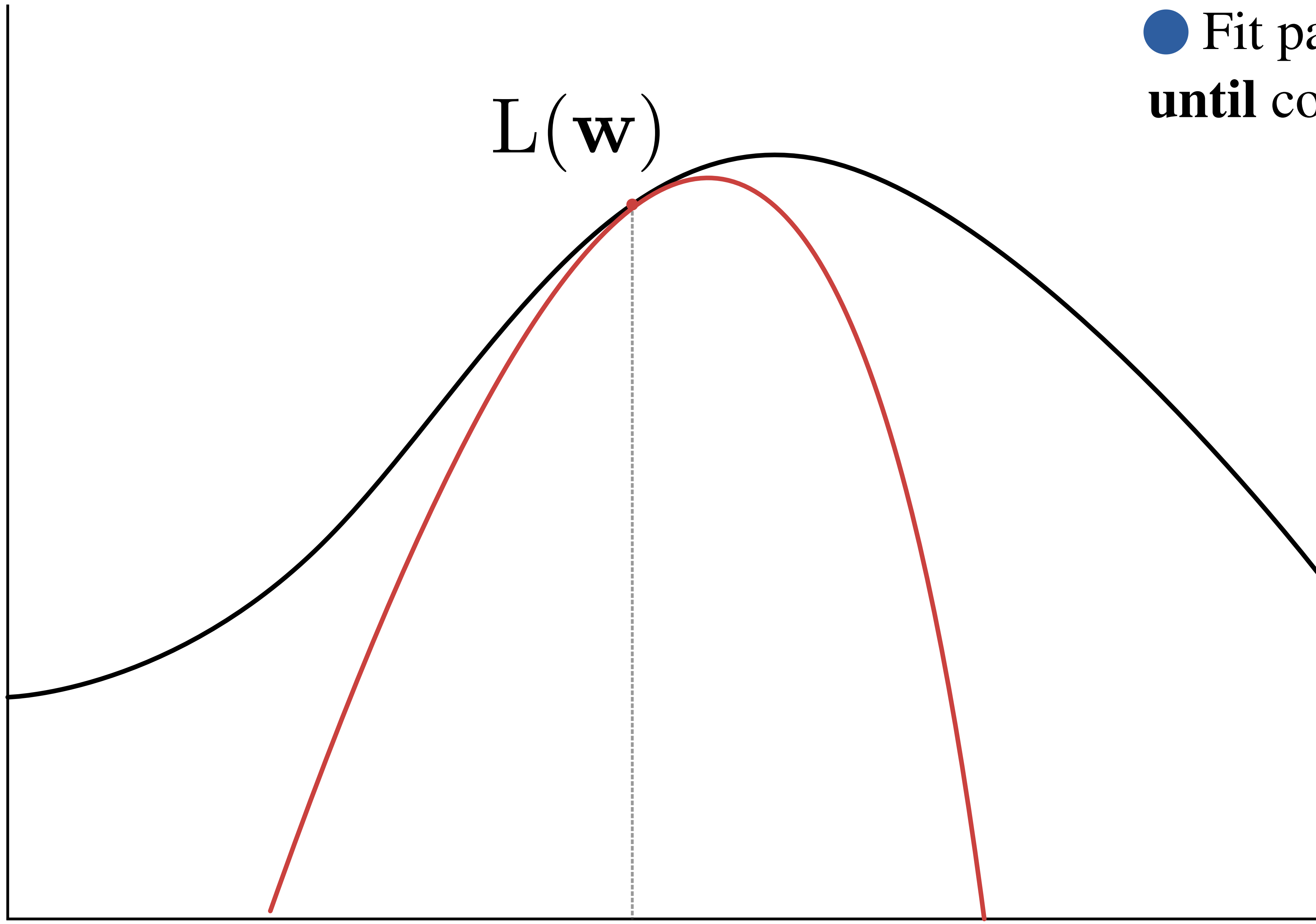
until convergence



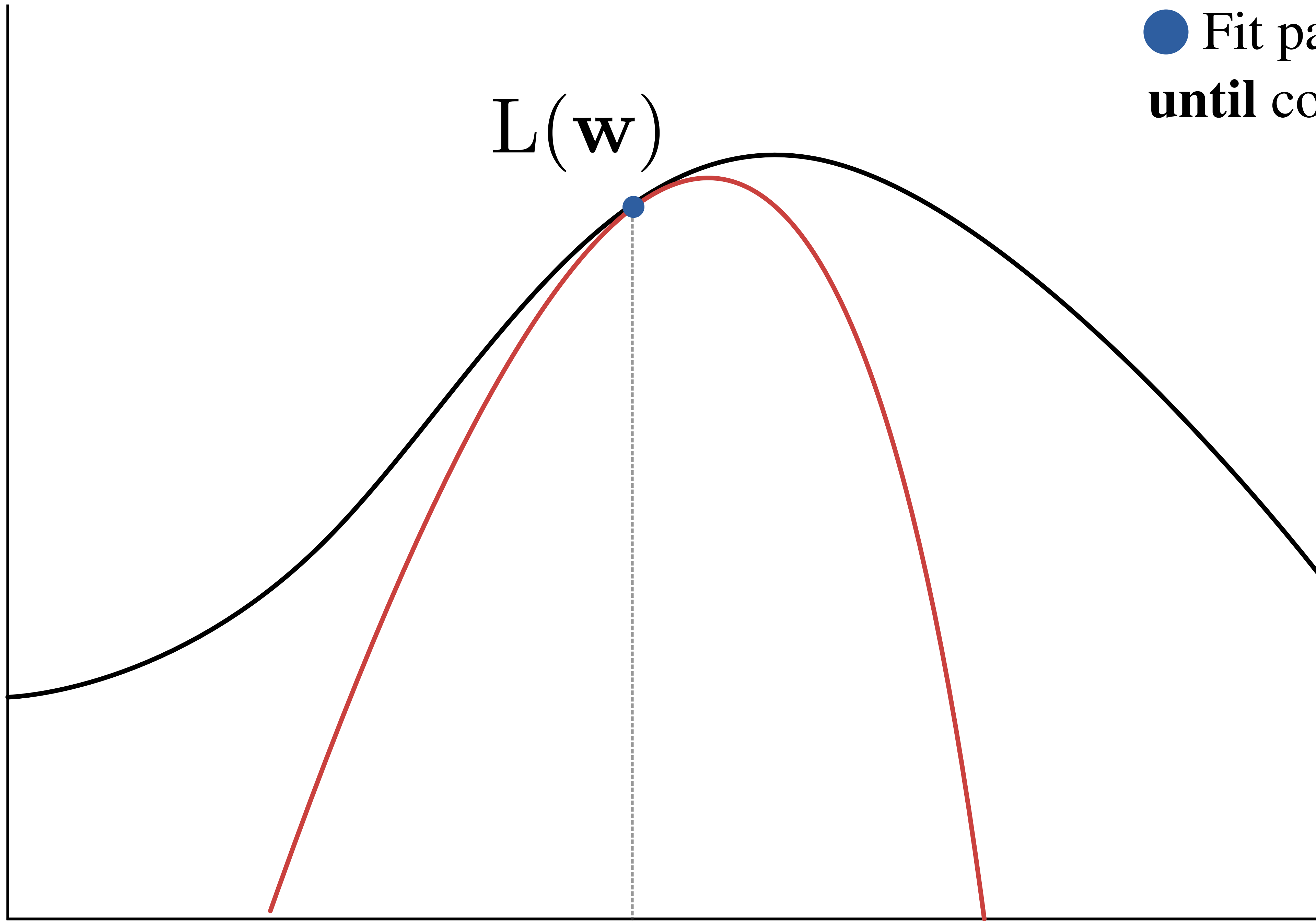
EM without Features

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

until convergence



EM without Features



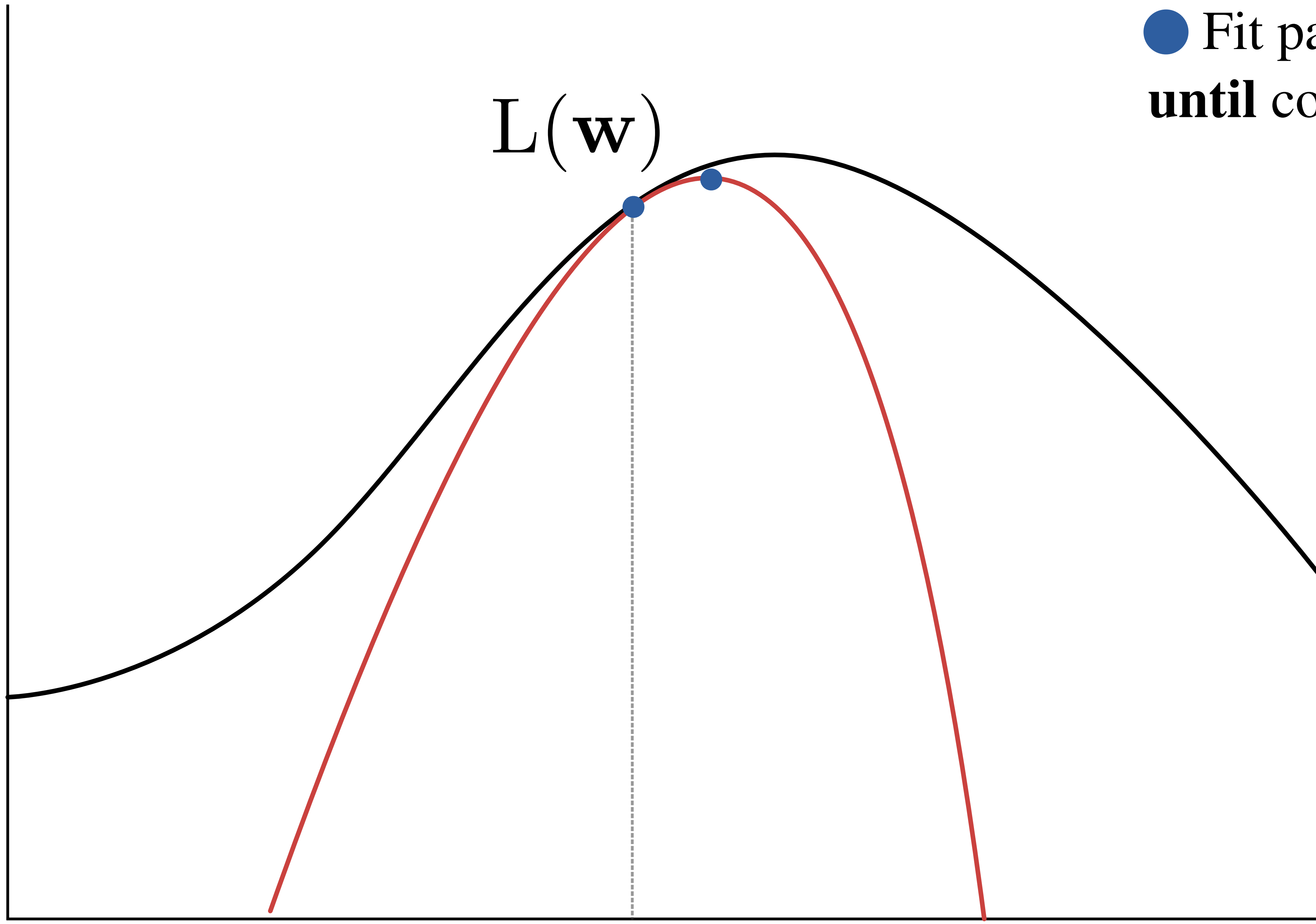
Initialize probabilities $\boldsymbol{\theta}$
repeat
- Compute expected counts $\mathbf{e}$
- Fit parameters $\boldsymbol{\theta}$

until convergence

EM without Features

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

until convergence



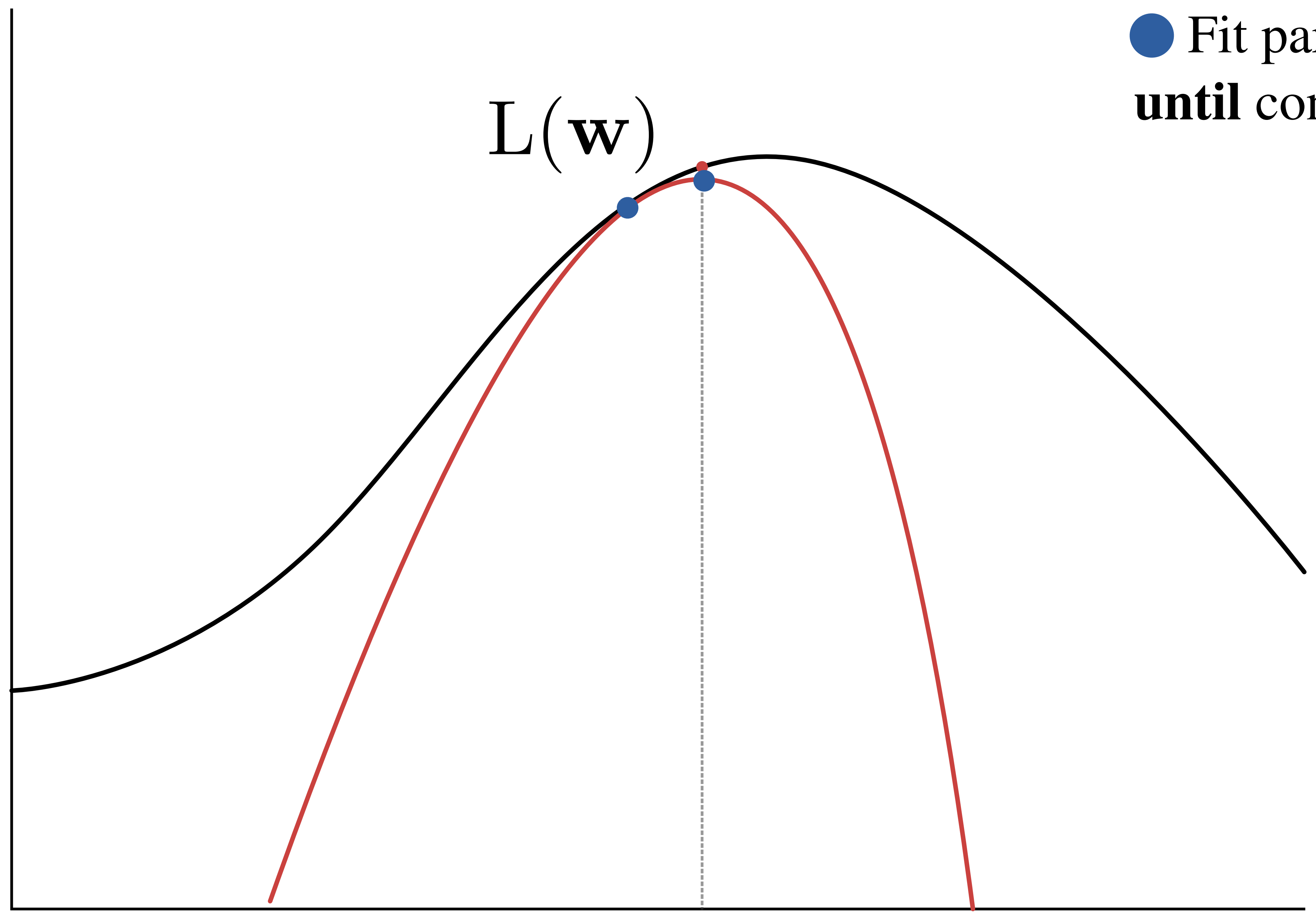
EM without Features

Initialize probabilities $\boldsymbol{\theta}$

repeat

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

until convergence



EM without Features

EM

Initialize probabilities $\boldsymbol{\theta}$

repeat

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

until convergence

EM with Features

EM

Initialize weights $\mathbf{w}$

repeat

- Compute expected counts $\mathbf{e}$
- Fit parameters $\mathbf{w}$
- Transform $\mathbf{w}$ to $\boldsymbol{\theta}$

until convergence

EM with Features

EM

Initialize weights $\mathbf{w}$

repeat

- Compute expected counts $\mathbf{e}$

  Fit Params

  repeat

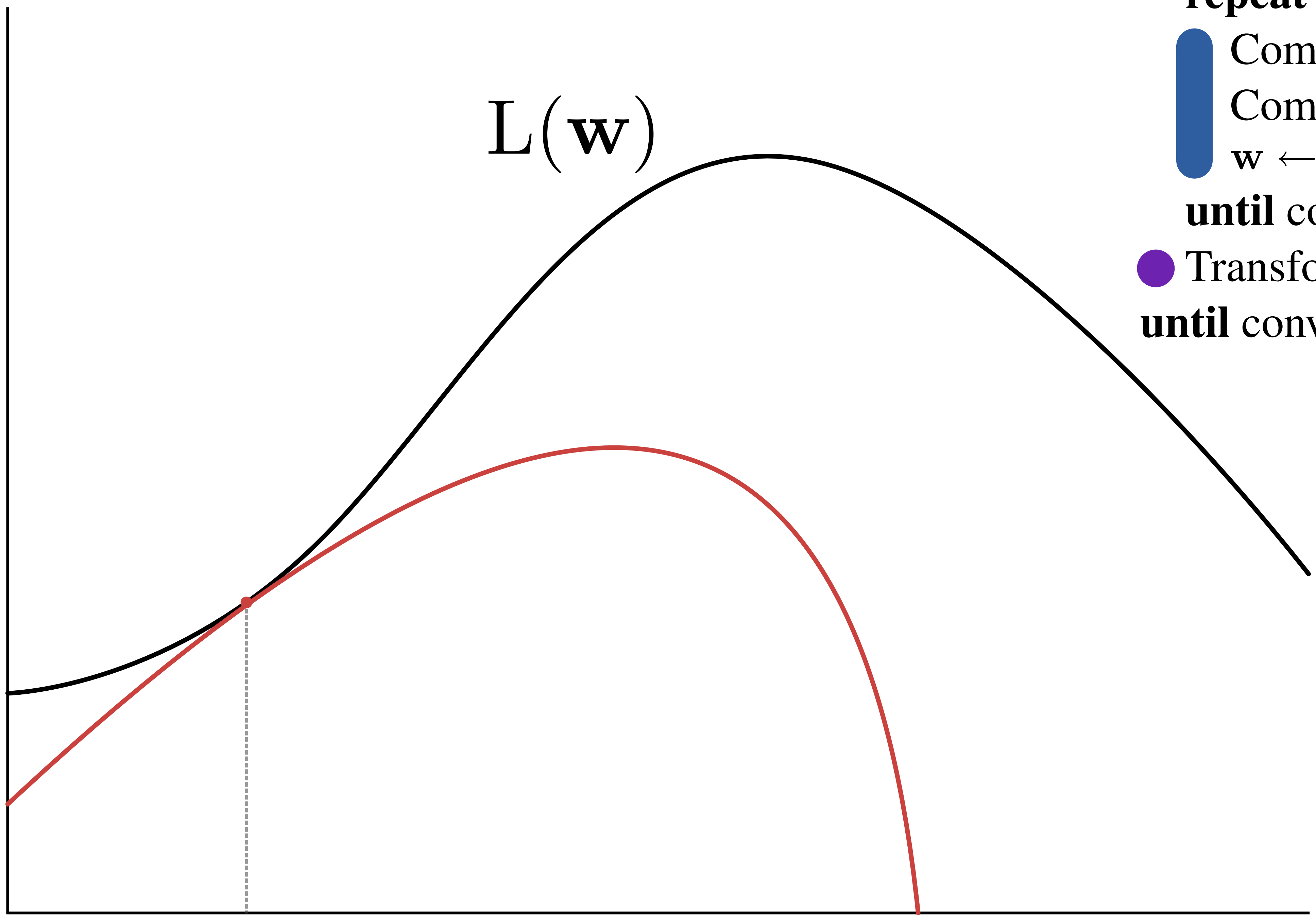
  - Compute $\ell(\mathbf{w}, \mathbf{e})$
  - Compute $\nabla \ell(\mathbf{w}, \mathbf{e})$
  - $\mathbf{w} \leftarrow \text{climb}(\mathbf{w}, \ell(\mathbf{w}, \mathbf{e}), \nabla \ell(\mathbf{w}, \mathbf{e}))$

  until convergence

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

until convergence

EM with Features



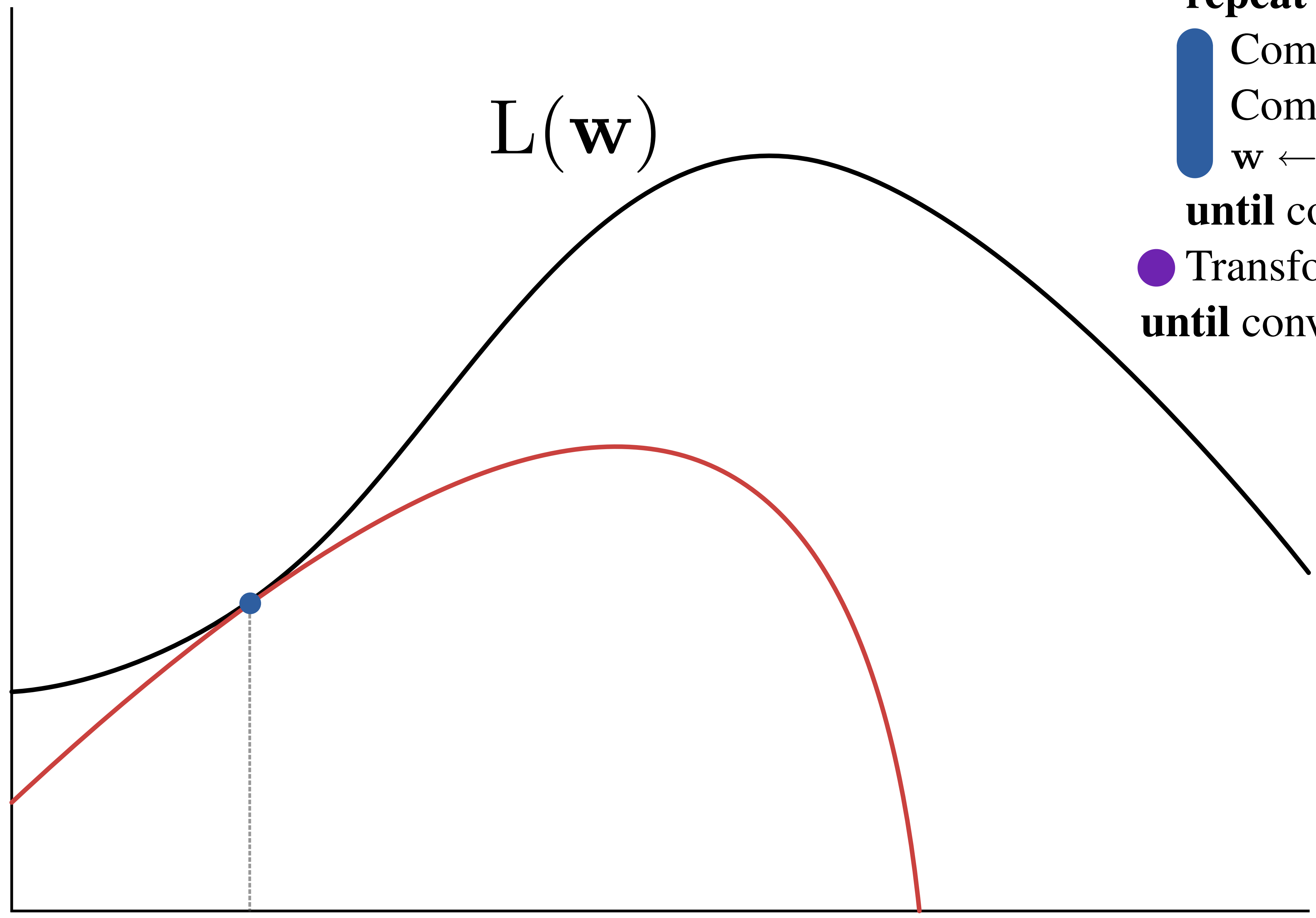
Initialize weights $\mathbf{w}$

repeat

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

until convergence

EM with Features



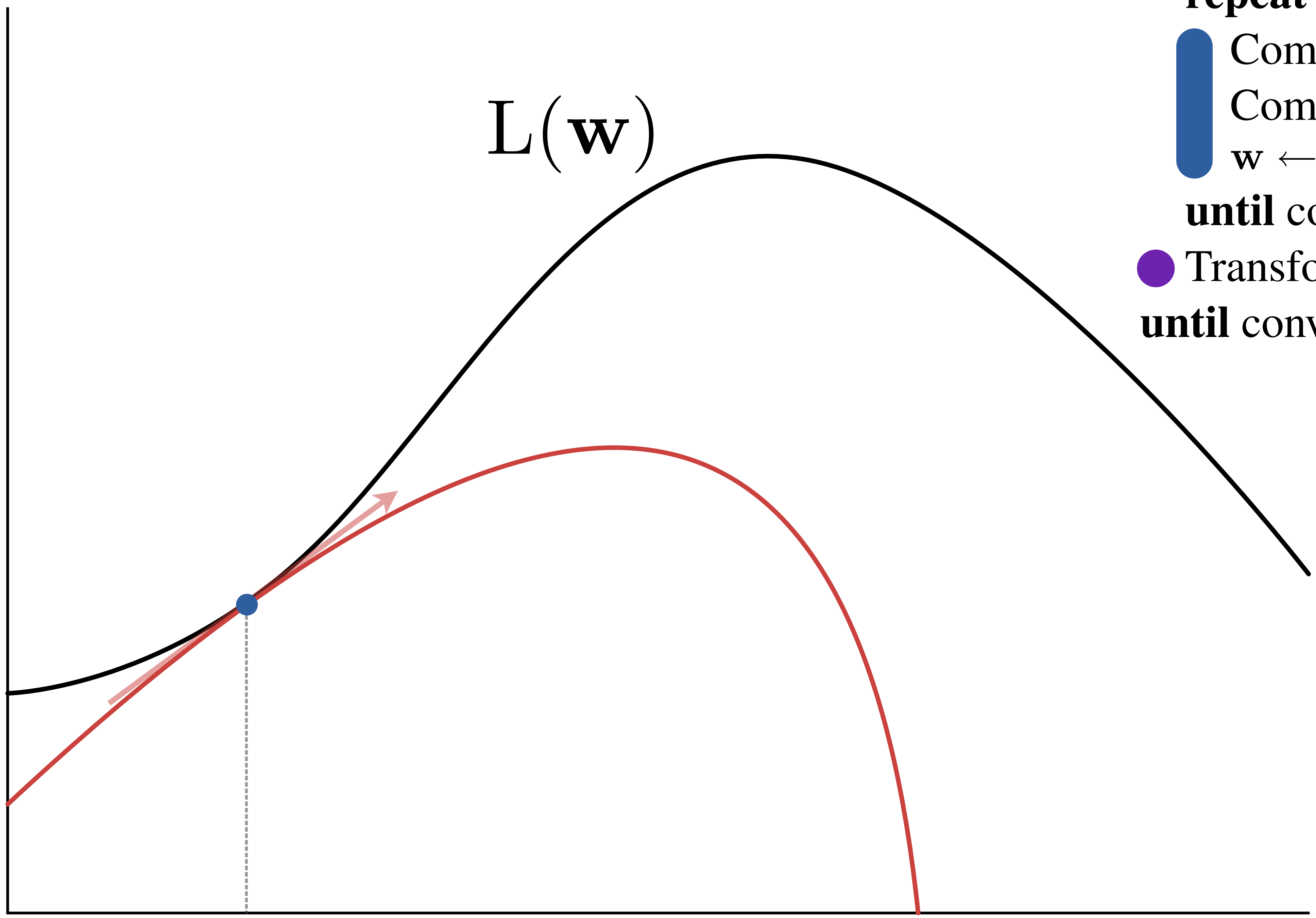
Initialize weights $\mathbf{w}$

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

EM with Features

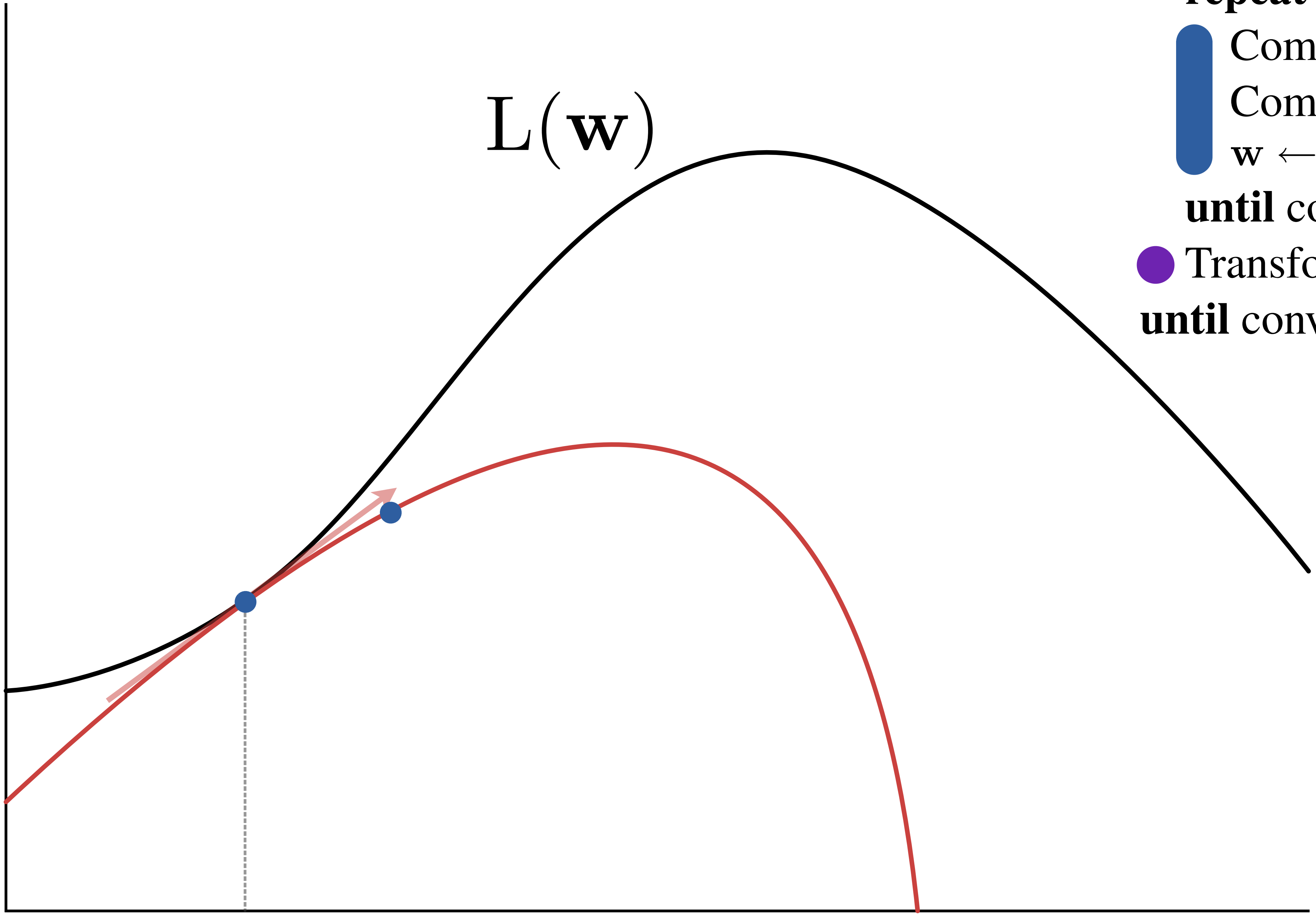


Initialize weights $\mathbf{w}$
repeat
- Compute expected counts $\mathbf{e}$
  repeat
  - Compute $\ell(\mathbf{w}, \mathbf{e})$
  - Compute $\nabla\ell(\mathbf{w}, \mathbf{e})$
  - $\mathbf{w} \leftarrow \text{climb}(\mathbf{w}, \ell(\mathbf{w}, \mathbf{e}), \nabla\ell(\mathbf{w}, \mathbf{e}))$

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

until convergence

EM with Features



Initialize weights $\mathbf{w}$

repeat

- Compute expected counts $\mathbf{e}$

  repeat

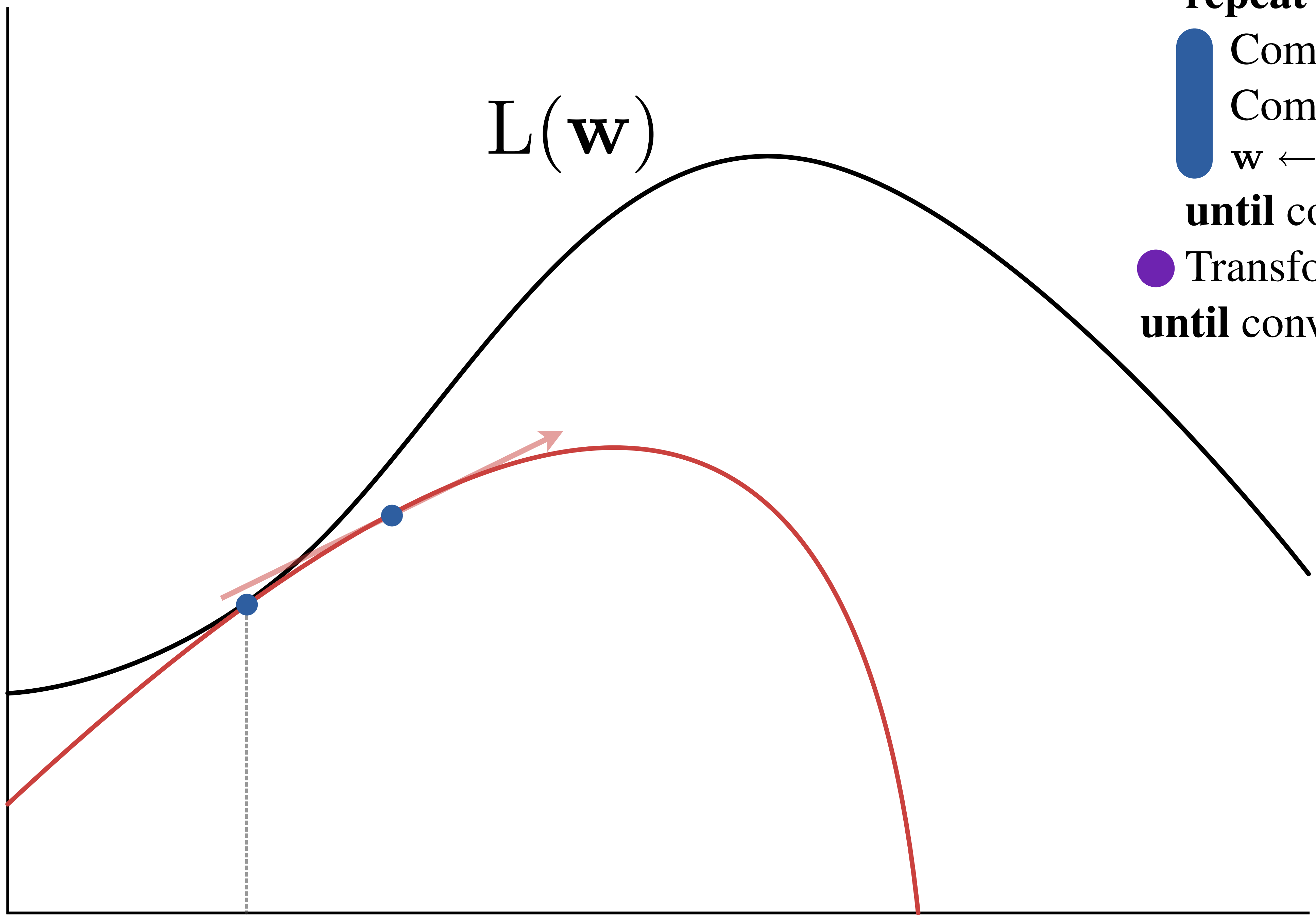
  - Compute $\ell(\mathbf{w}, \mathbf{e})$
  - Compute $\nabla\ell(\mathbf{w}, \mathbf{e})$
  - $\mathbf{w} \leftarrow \mathrm{climb}(\mathbf{w}, \ell(\mathbf{w}, \mathbf{e}), \nabla\ell(\mathbf{w}, \mathbf{e}))$

  until convergence

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

until convergence

EM with Features



Initialize weights $\mathbf{w}$

repeat

Compute expected counts $\mathbf{e}$

repeat

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

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

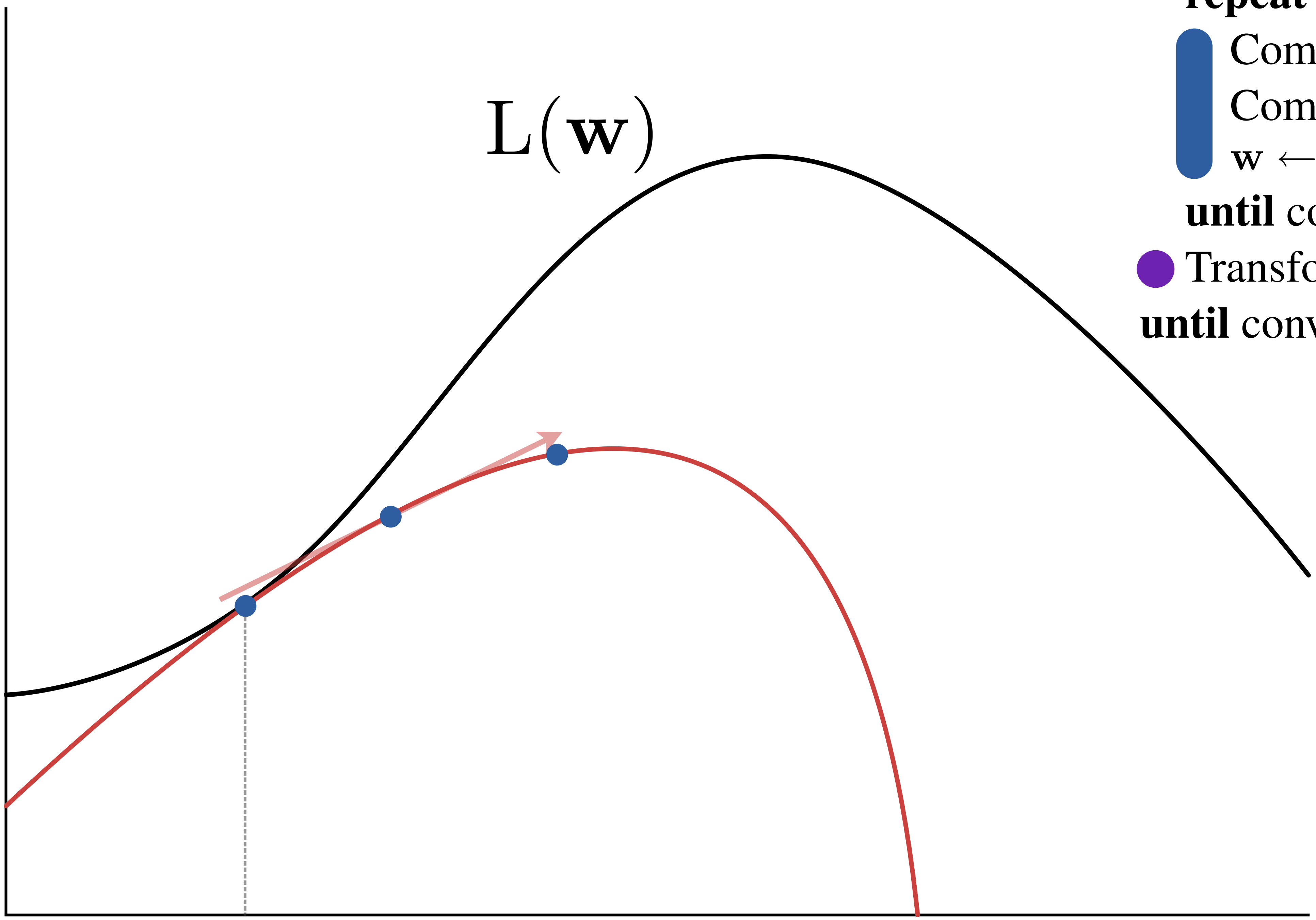
$\mathbf{w} \leftarrow \mathrm{climb}(\mathbf{w}, \ell(\mathbf{w}, \mathbf{e}), \nabla \ell(\mathbf{w}, \mathbf{e}))$

until convergence

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

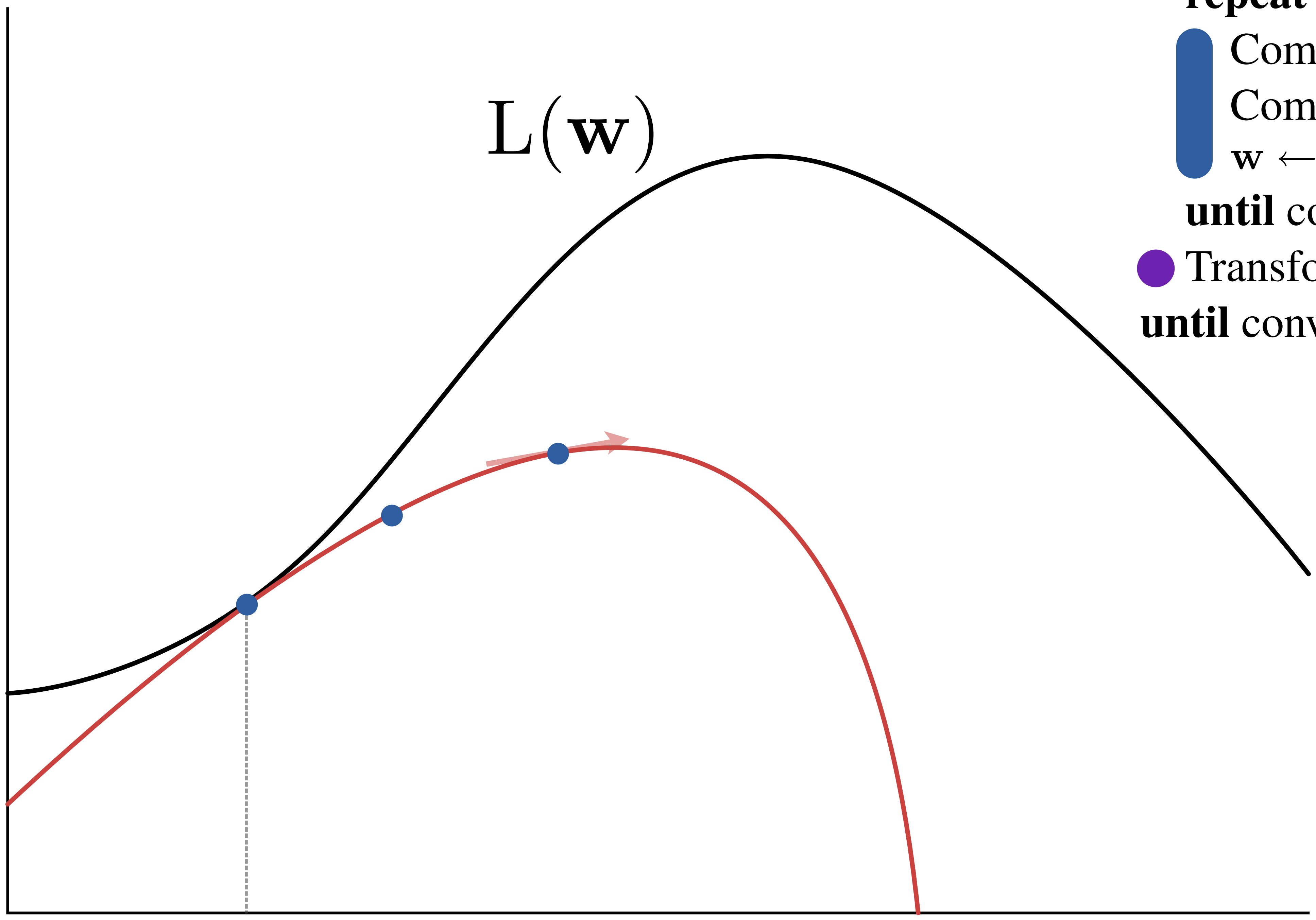
until convergence

EM with Features



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

EM with Features



Initialize weights $\mathbf{w}$

repeat

Compute expected counts $\mathbf{e}$

repeat

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

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

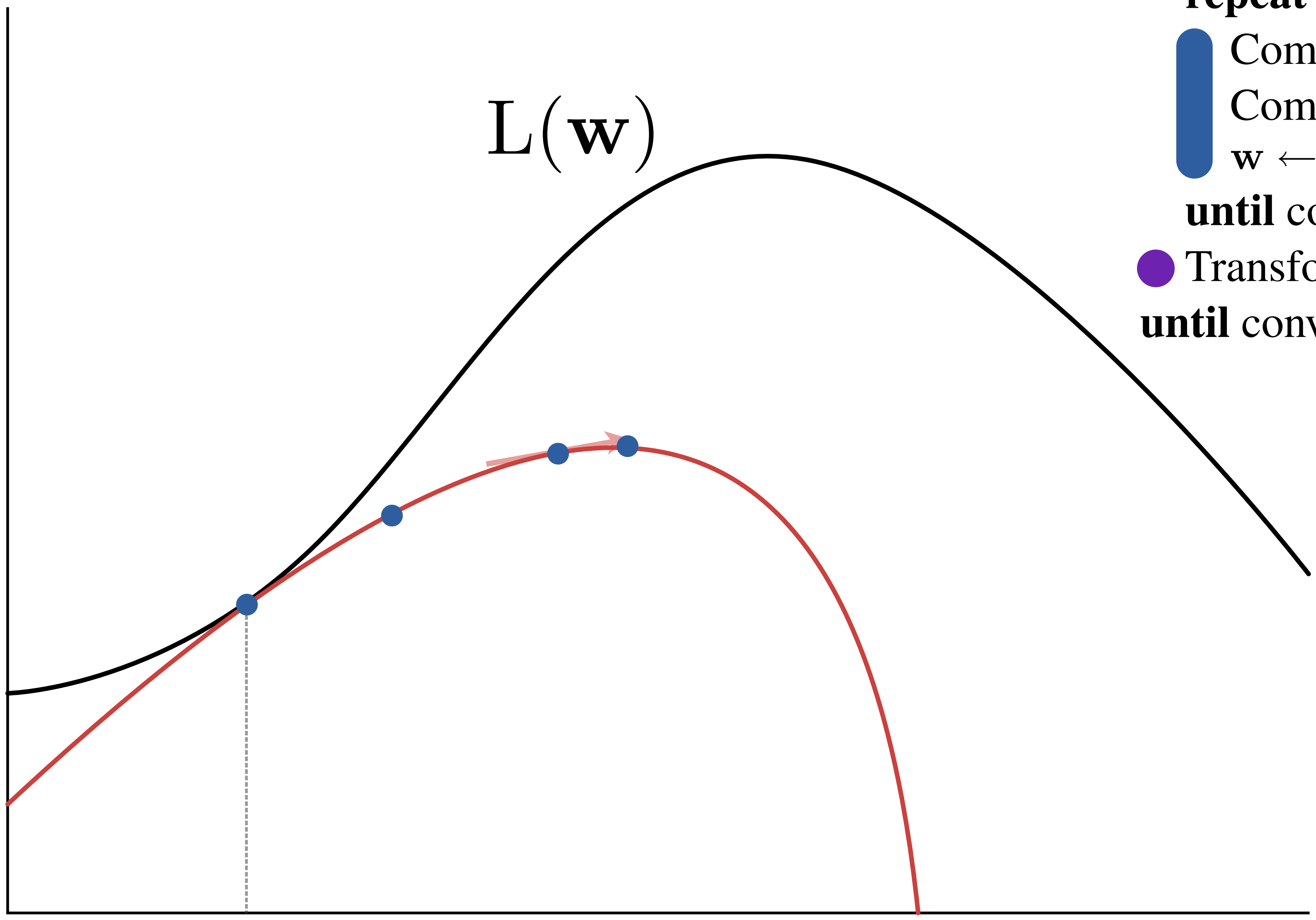
$\mathbf{w} \leftarrow \text{climb}(\mathbf{w}, \ell(\mathbf{w}, \mathbf{e}), \nabla\ell(\mathbf{w}, \mathbf{e}))$

until convergence

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

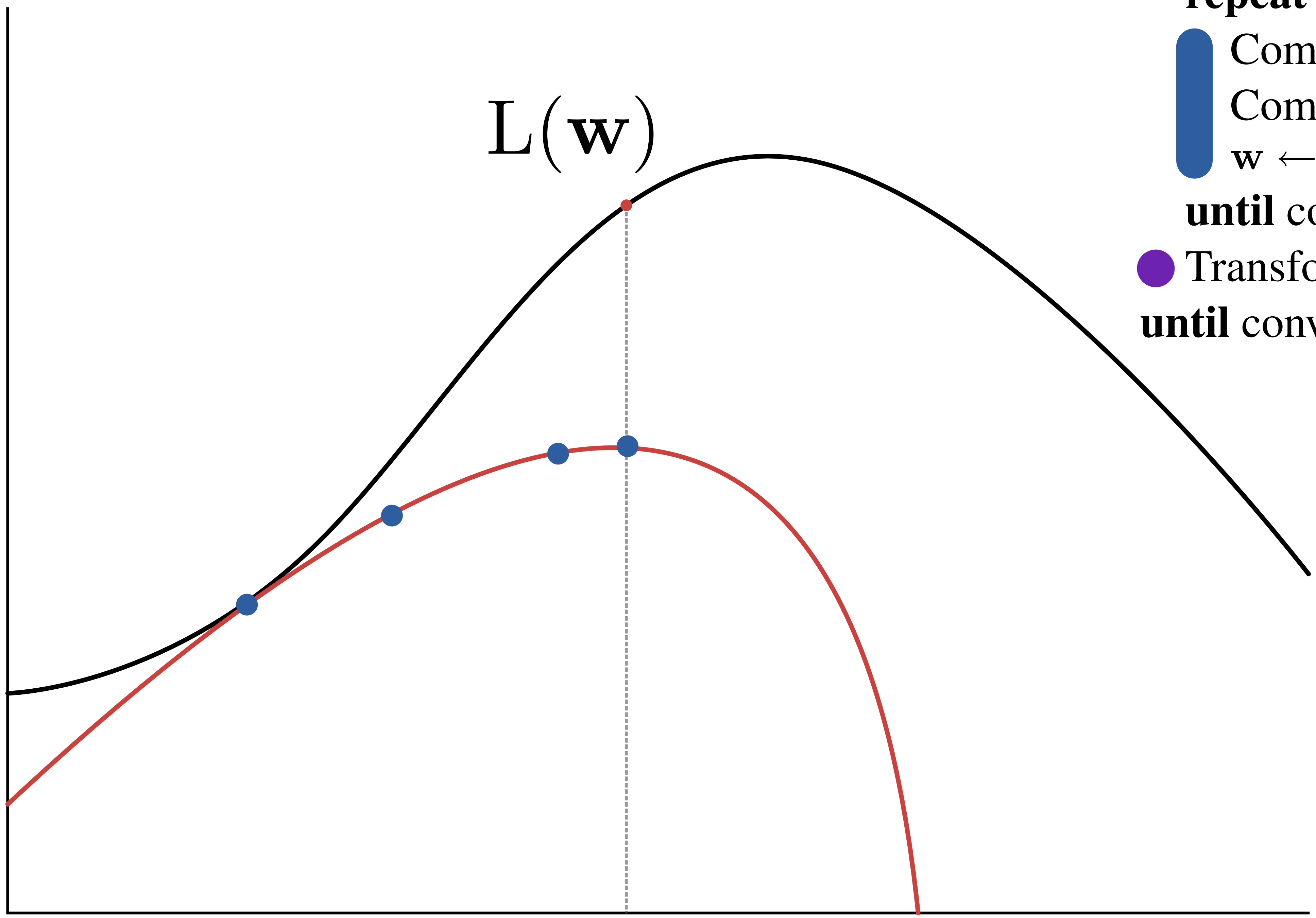
until convergence

EM with Features



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

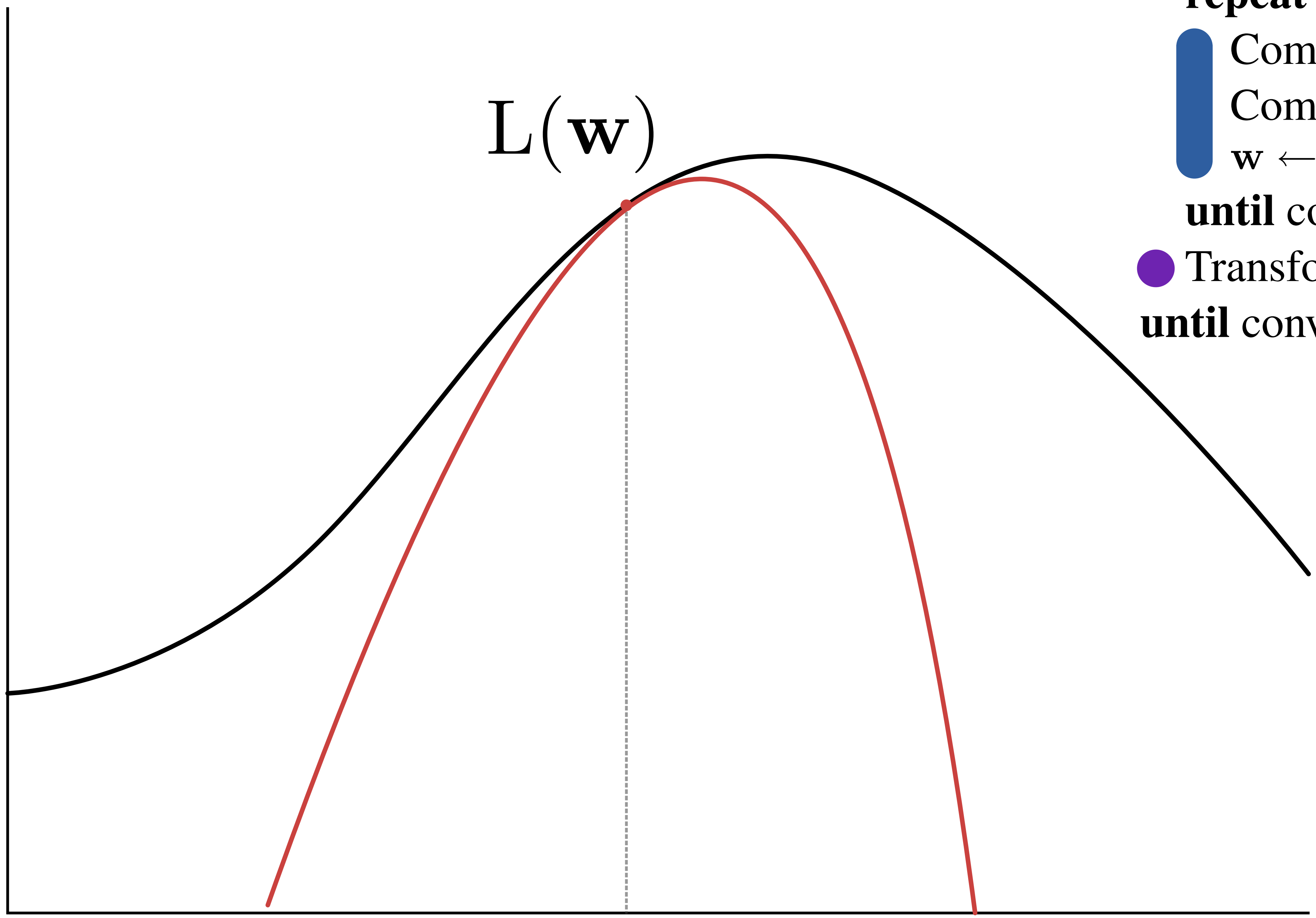
EM with Features



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

EM with Features



Initialize weights $\mathbf{w}$

repeat

Compute expected counts $\mathbf{e}$

repeat

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

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

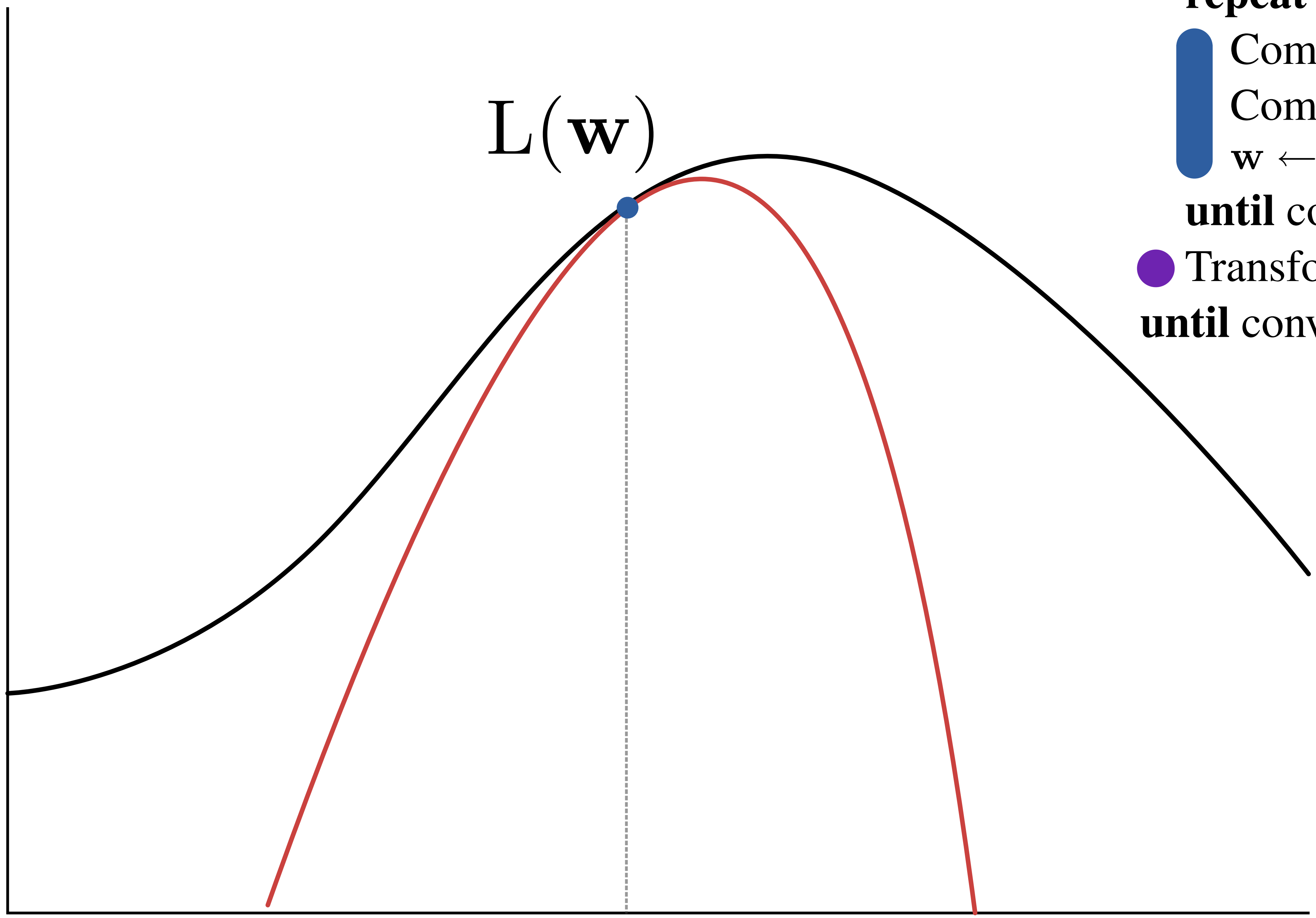
$\mathbf{w} \leftarrow \text{climb}(\mathbf{w}, \ell(\mathbf{w}, \mathbf{e}), \nabla \ell(\mathbf{w}, \mathbf{e}))$

until convergence

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

until convergence

EM with Features



Initialize weights $\mathbf{w}$

repeat

- Compute expected counts $\mathbf{e}$

  repeat

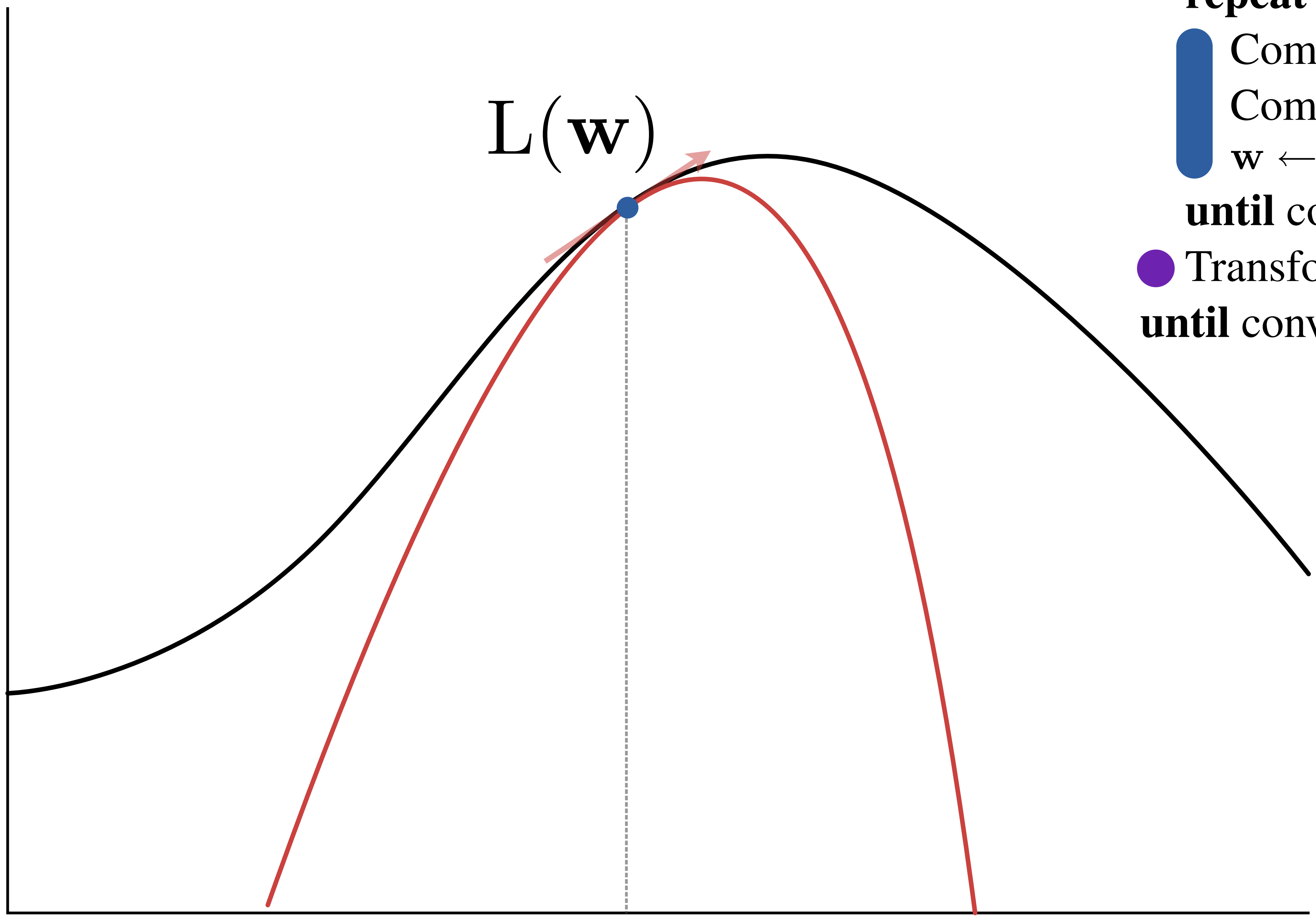
  - Compute $\ell(\mathbf{w}, \mathbf{e})$
  - Compute $\nabla \ell(\mathbf{w}, \mathbf{e})$
  - $\mathbf{w} \leftarrow \text{climb}(\mathbf{w}, \ell(\mathbf{w}, \mathbf{e}), \nabla \ell(\mathbf{w}, \mathbf{e}))$

  until convergence

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

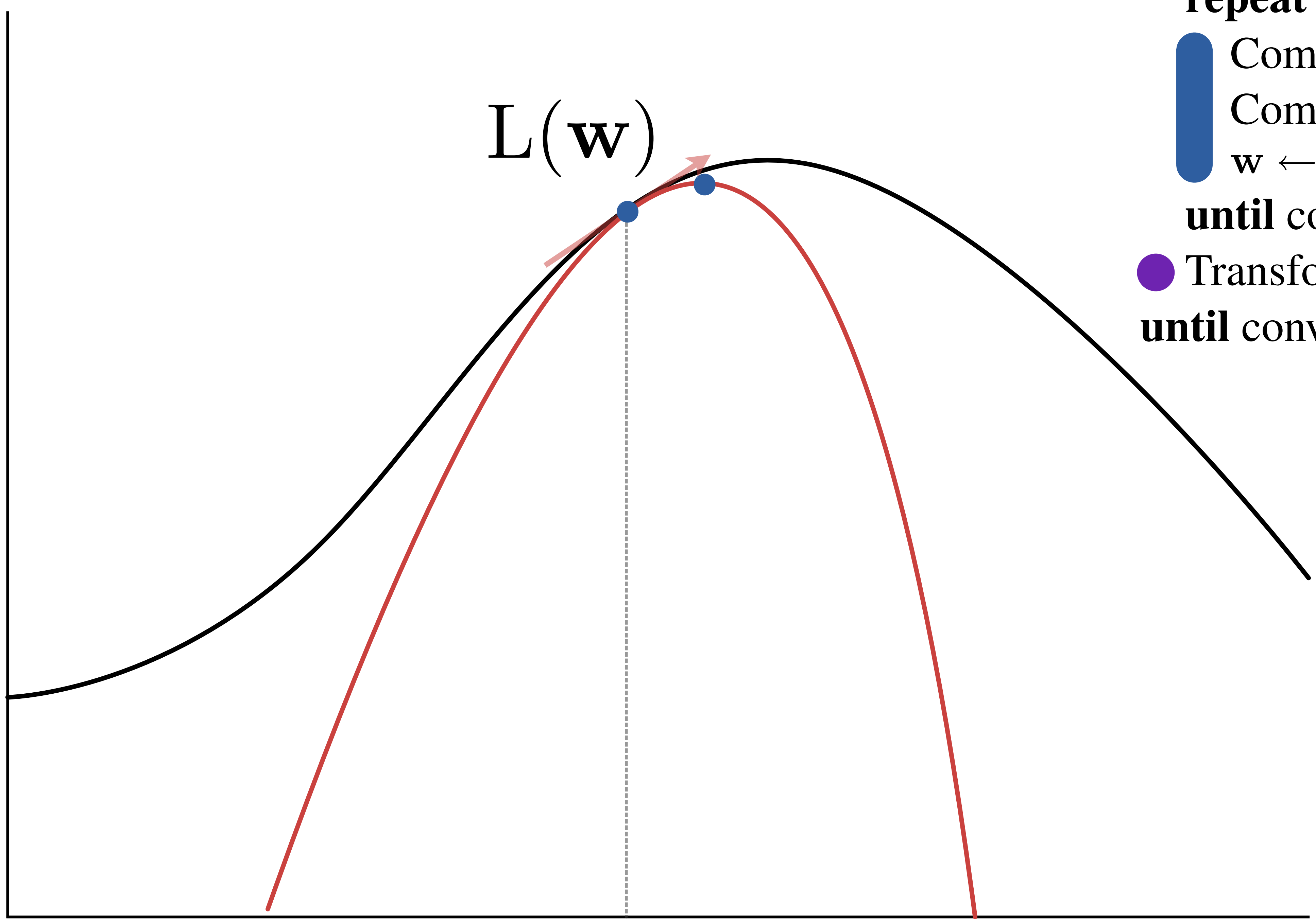
until convergence

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

EM with Features



Initialize weights $\mathbf{w}$

repeat

Compute expected counts $\mathbf{e}$

repeat

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

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

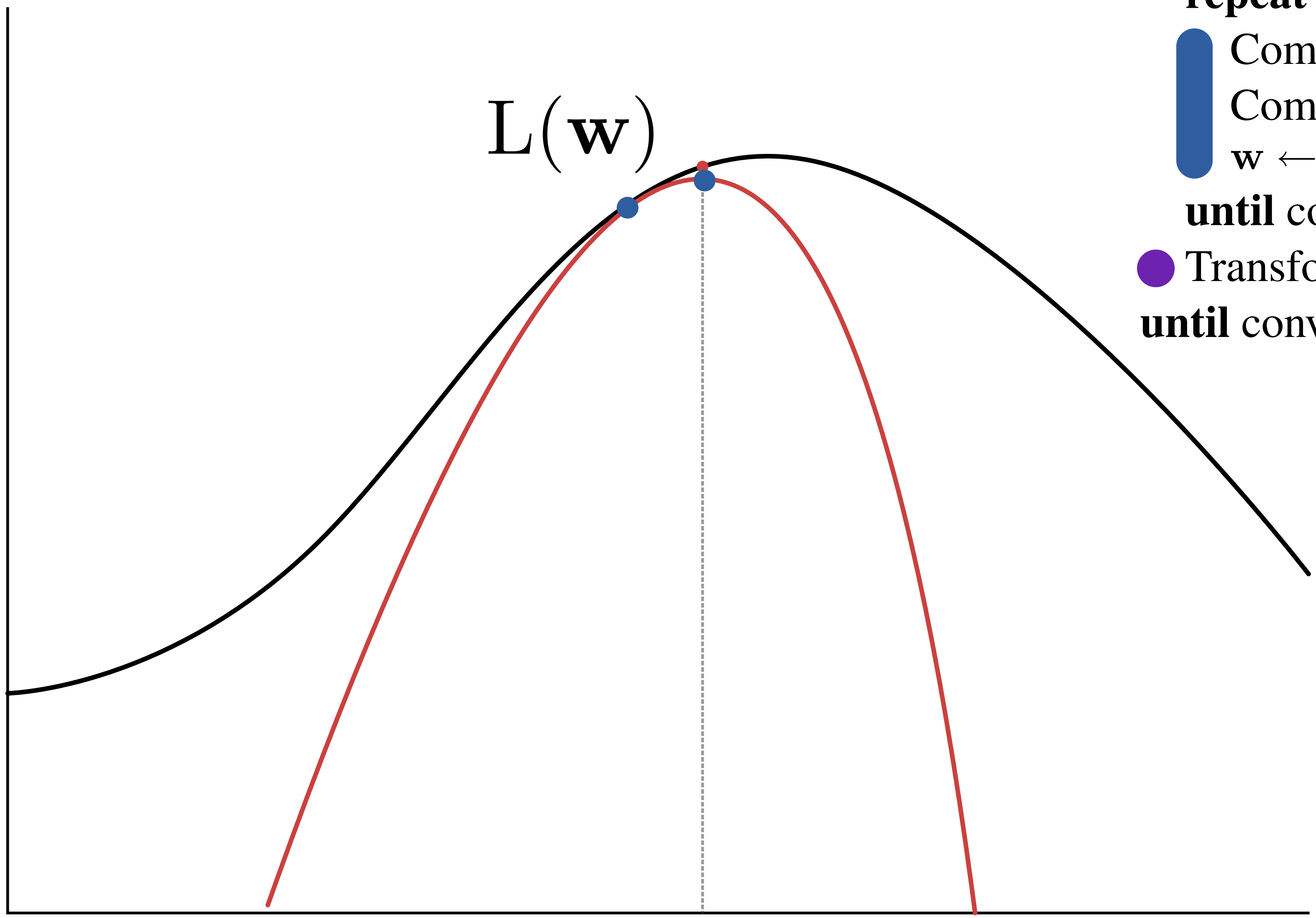
$\mathbf{w} \leftarrow \mathrm{climb}(\mathbf{w}, \ell(\mathbf{w}, \mathbf{e}), \nabla\ell(\mathbf{w}, \mathbf{e}))$

until convergence

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

until convergence

EM with Features



Initialize weights $\mathbf{w}$

repeat

Compute expected counts $\mathbf{e}$

repeat

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

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

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

until convergence

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

until convergence

Direct Gradient with Features

EM w/ 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 $\boldsymbol{\theta}$

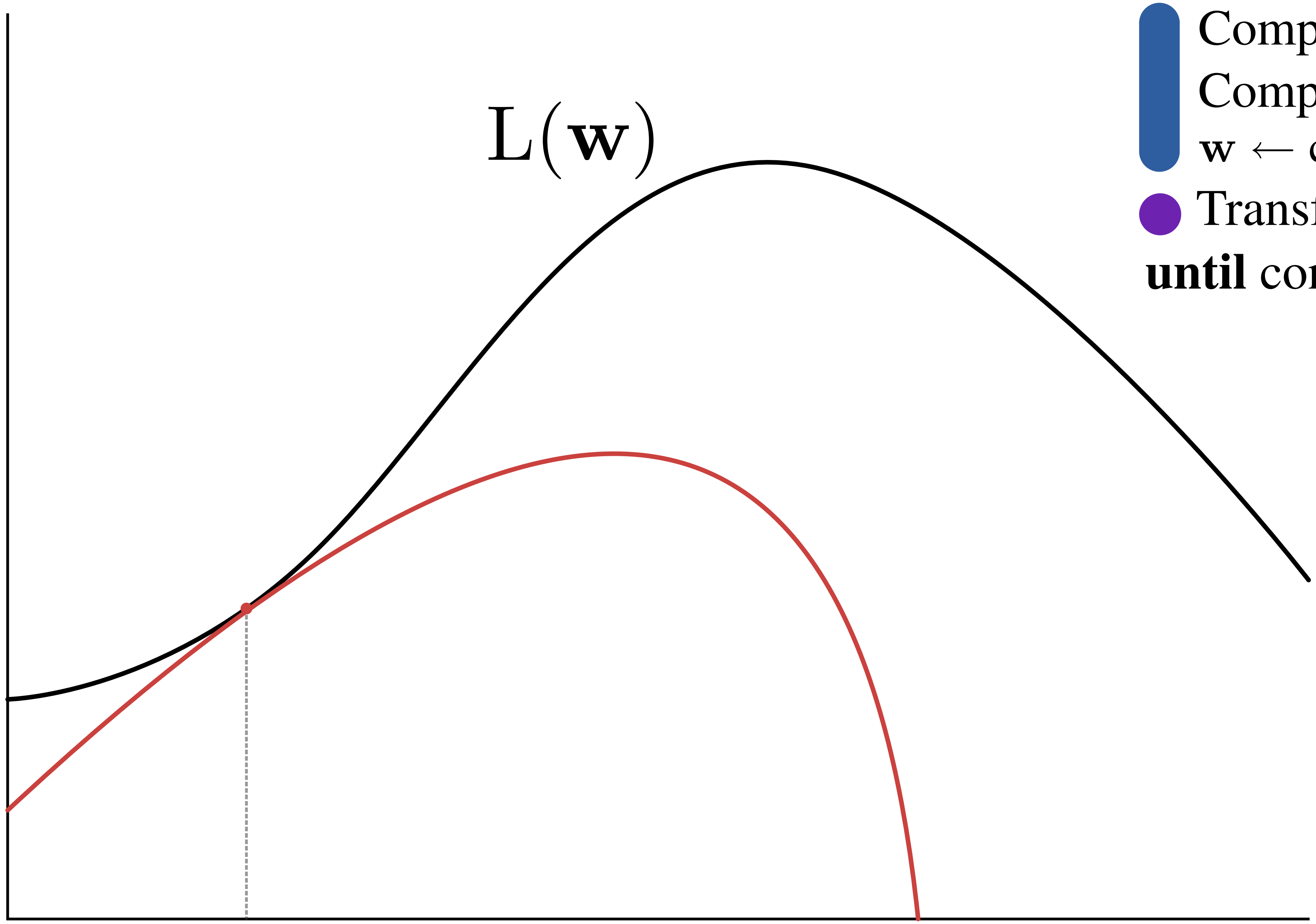
until convergence

DG w/ 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

Direct Gradient 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

Direct Gradient with Features

Initialize weights $\mathbf{w}$

repeat

Compute expected counts $\mathbf{e}$

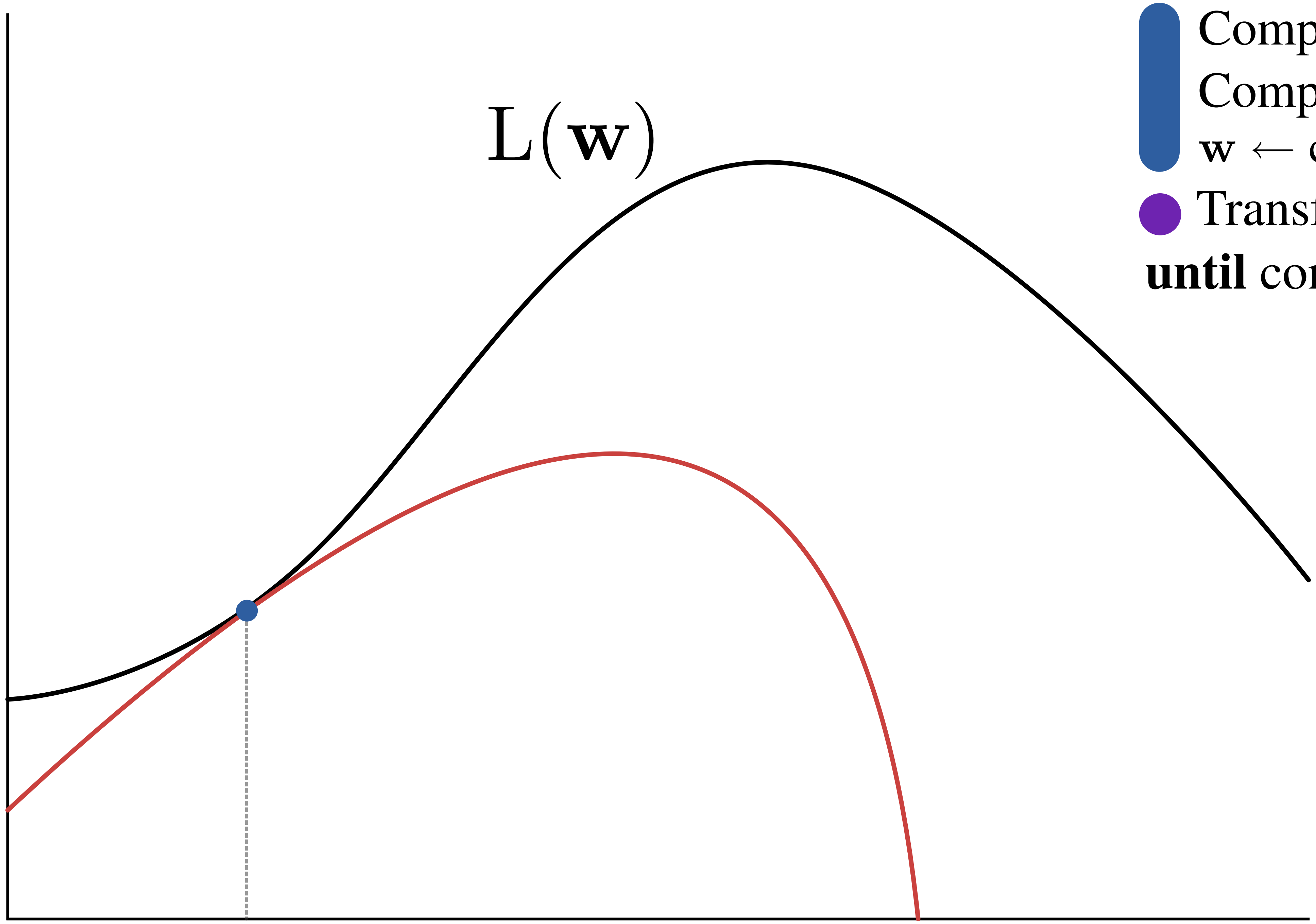
Compute $L(\mathbf{w})$

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

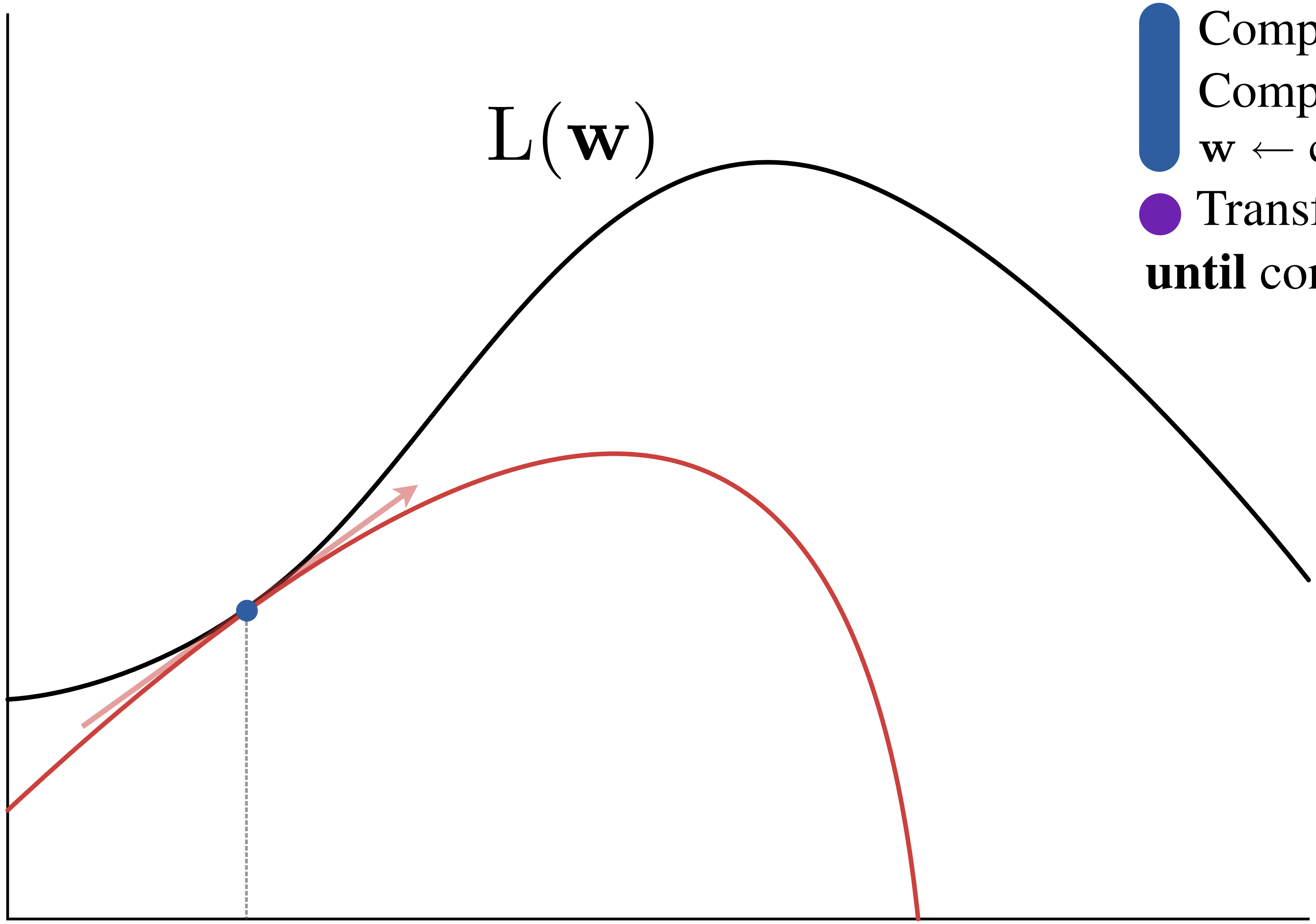
$\mathbf{w} \leftarrow \text{climb}(\mathbf{w}, L(\mathbf{w}), \nabla \ell(\mathbf{w}, \mathbf{e}))$

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

until convergence



Direct Gradient with Features



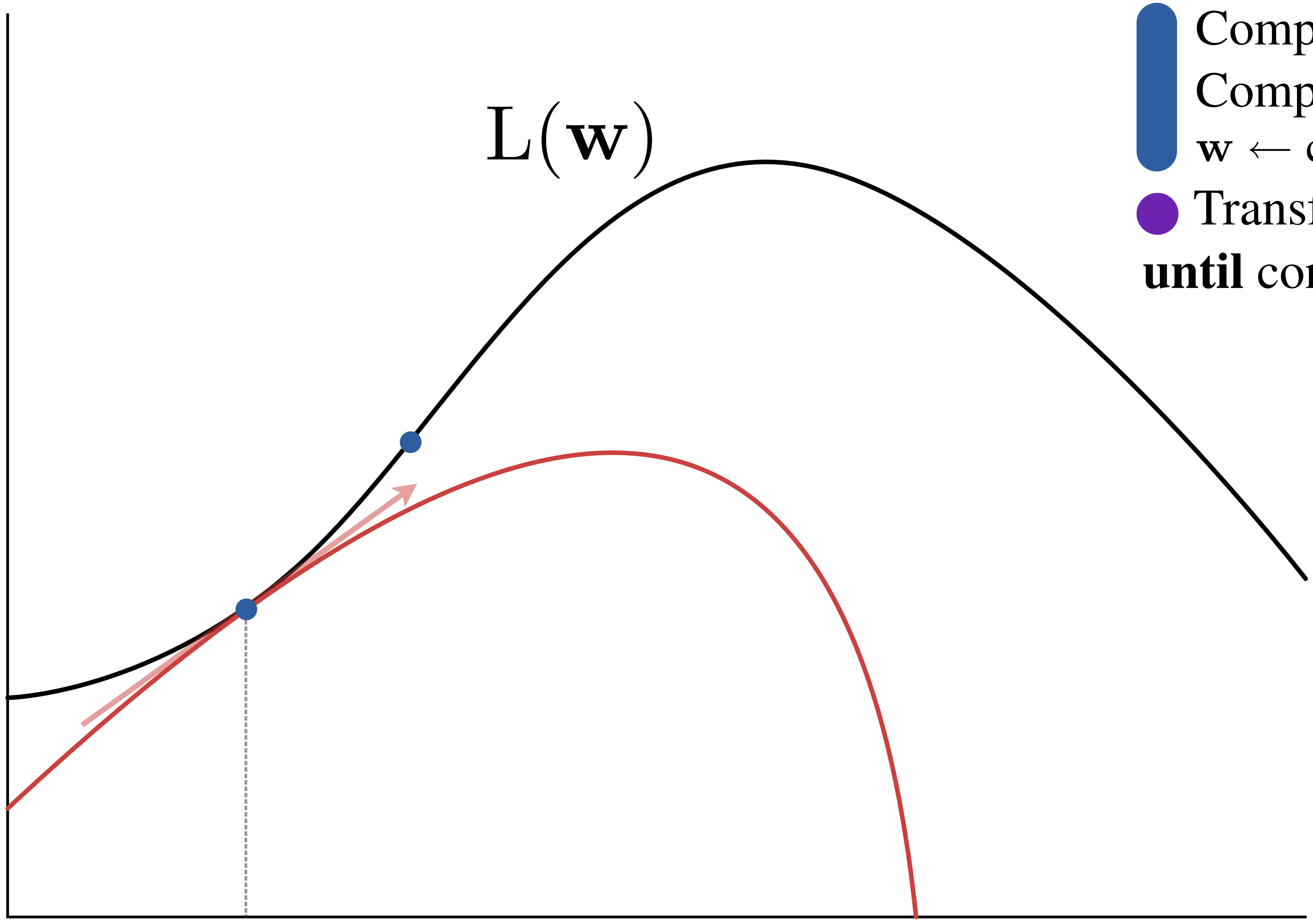
Initialize weights $\mathbf{w}$

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

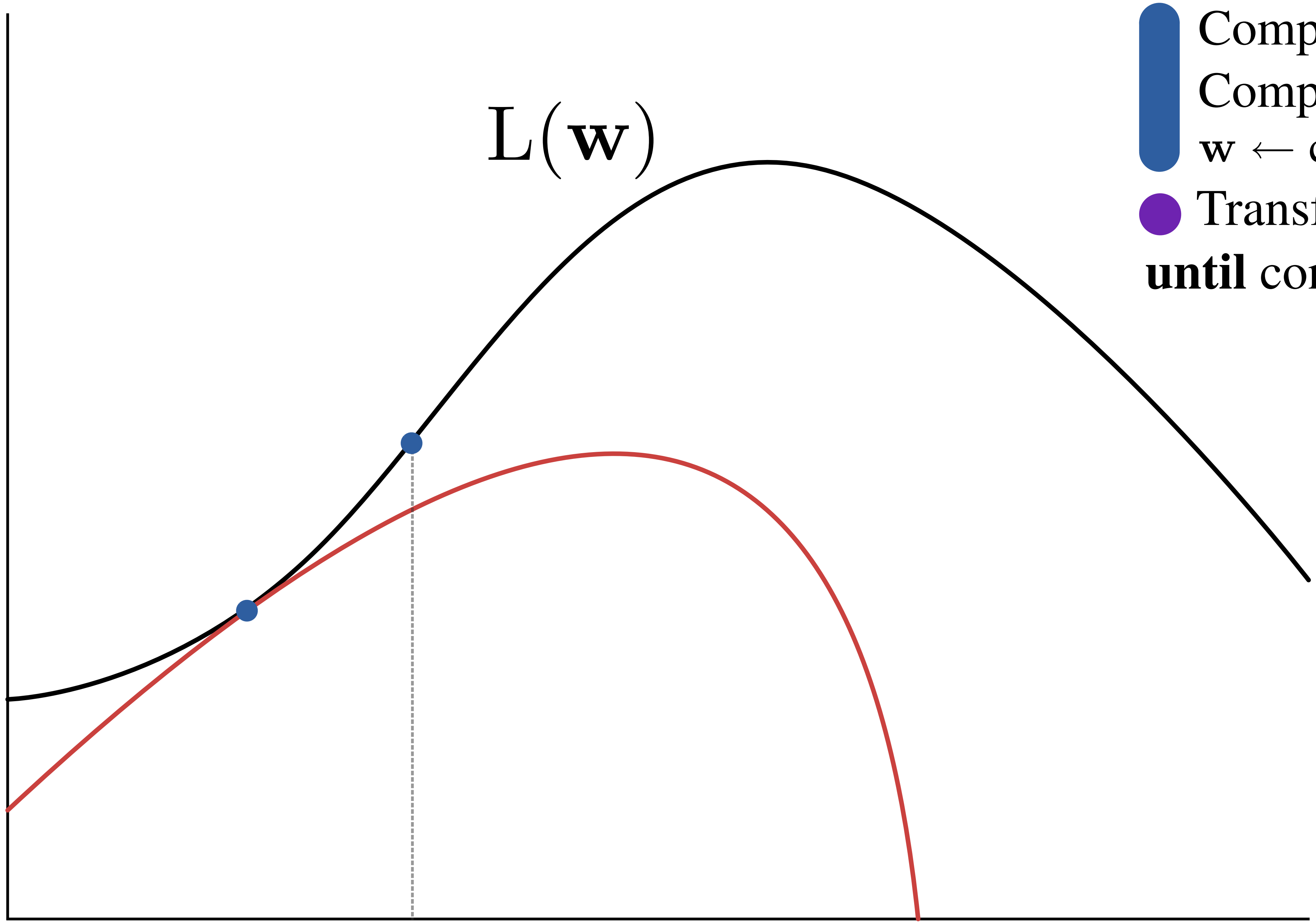
Direct Gradient with Features



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

Direct Gradient 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

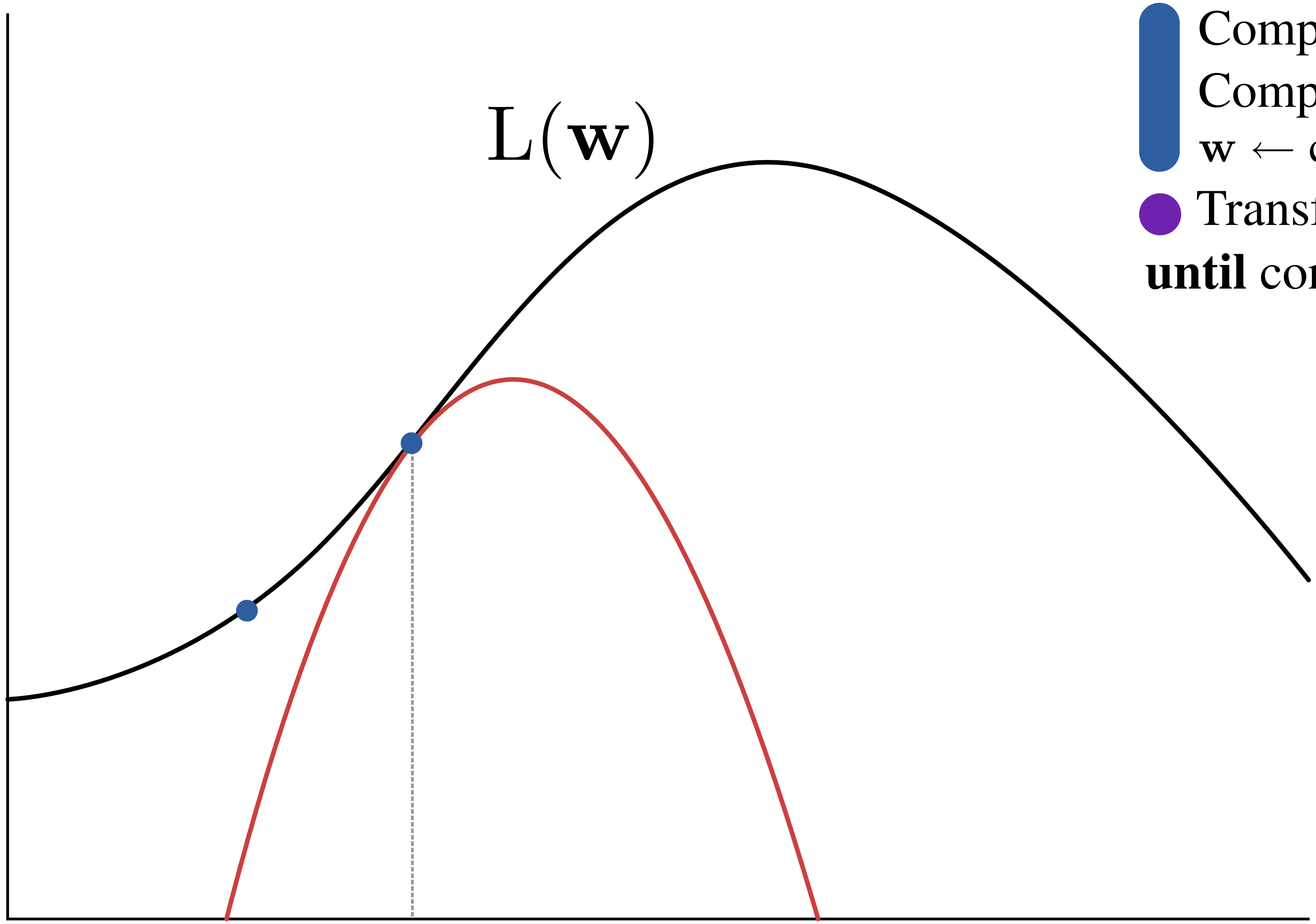
Direct Gradient with Features

Initialize weights $\mathbf{w}$

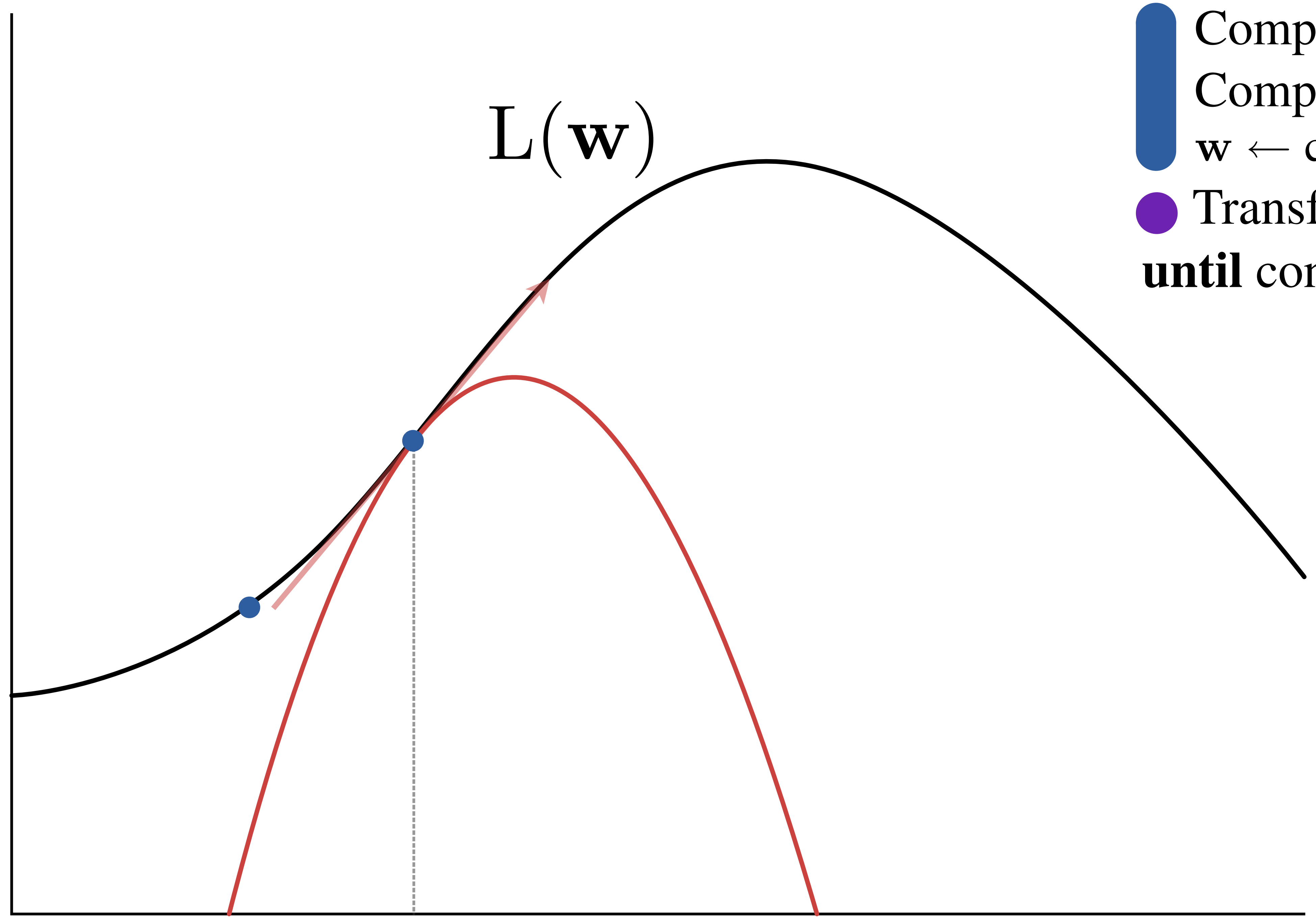
repeat

- 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



Direct Gradient with Features



Initialize weights $\mathbf{w}$

repeat

Compute expected counts $\mathbf{e}$

Compute $L(\mathbf{w})$

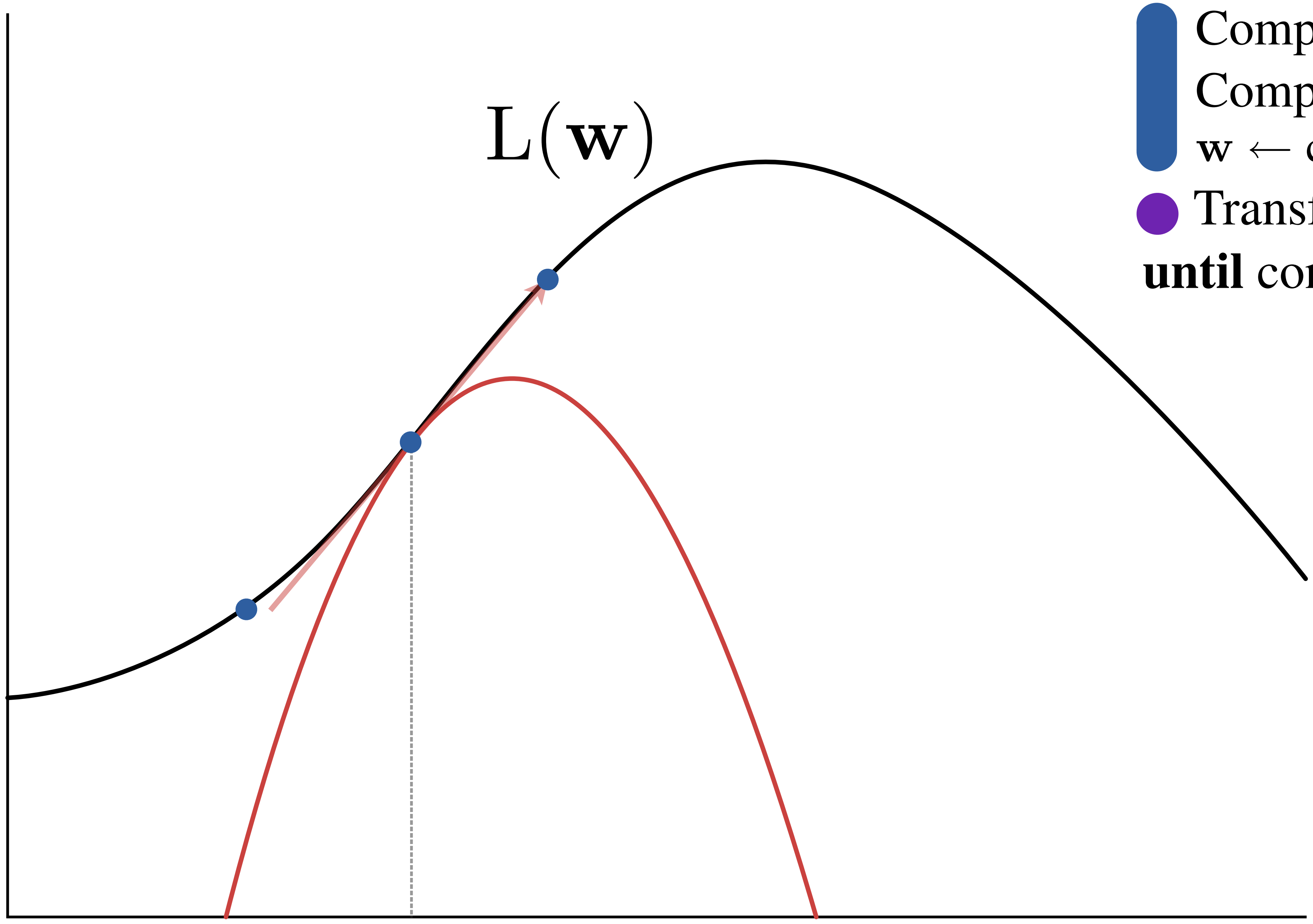
Compute $\nabla \ell(\mathbf{w}, \mathbf{e})$

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

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

until convergence

Direct Gradient 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

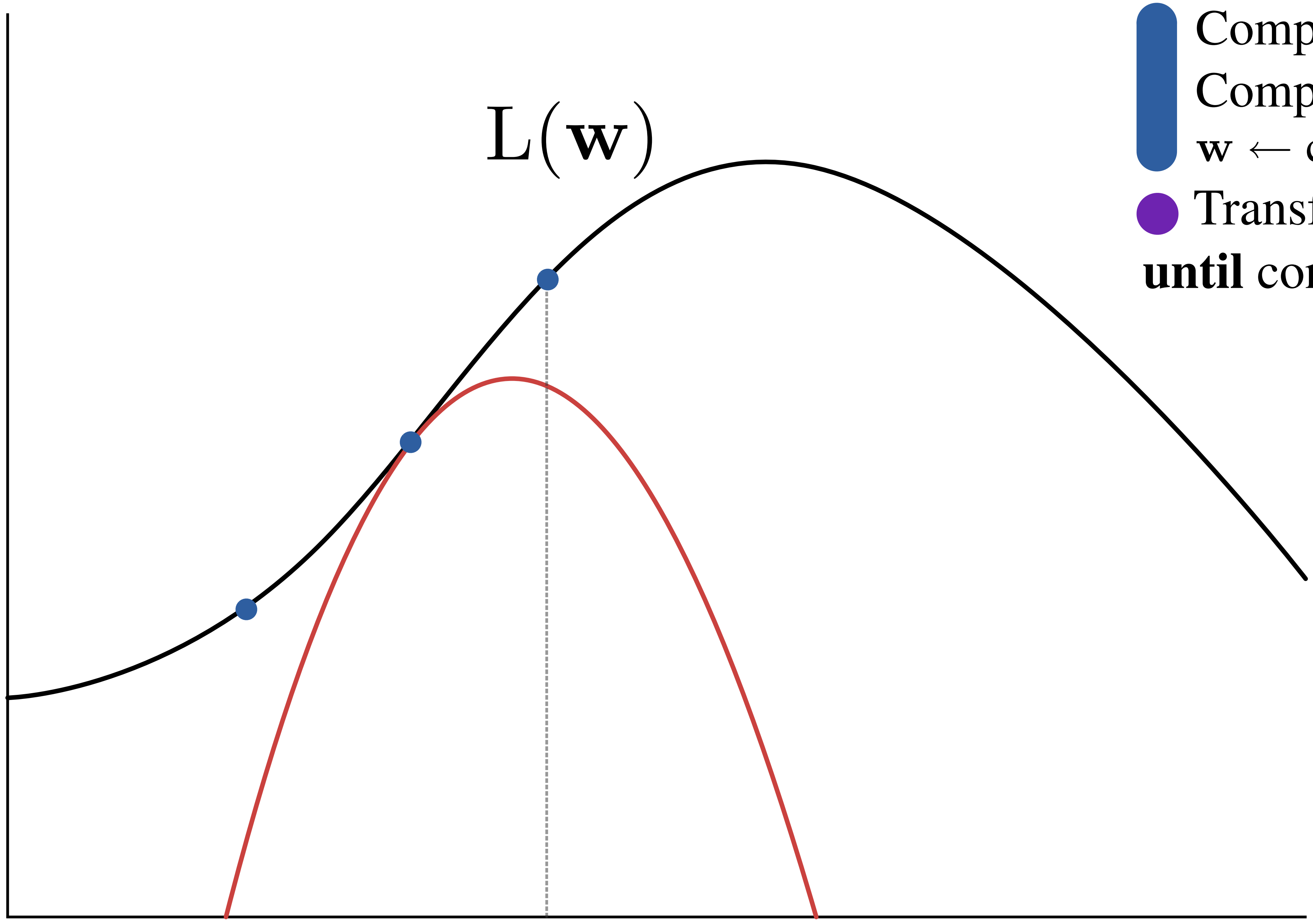
Direct Gradient with Features

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

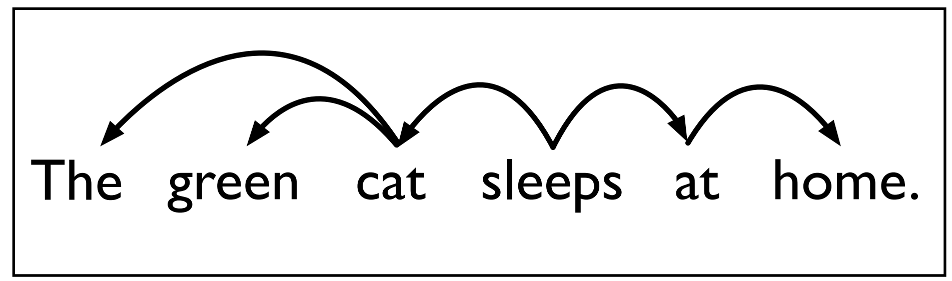


Unsupervised Induction Tasks

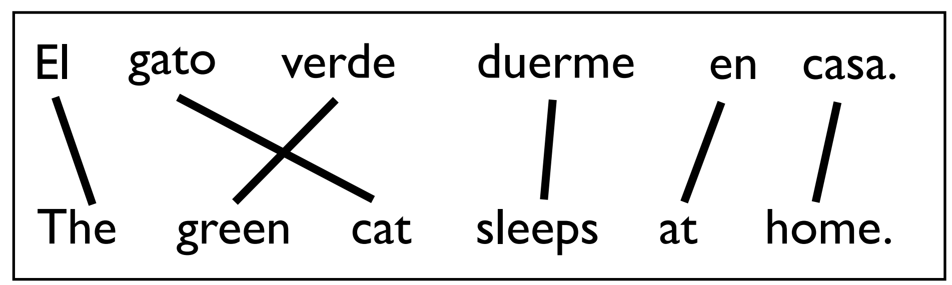
POS Induction:

DT	JJ	NN	VBZ	IN	NN
The	green	cat	sleeps	at	home.

Grammar Induction:



Word Alignment:



Word Segmentation:

[T h e][g r e e n][c a t]

POS Induction Results

DT	JJ	NN	VBZ	IN	NN
The	green	cat	sleeps	at	home.

POS Induction Results

DT	JJ	NN	VBZ	IN	NN
The	green	cat	sleeps	at	home.

Key distribution: $P(\mathrm{John}|\mathrm{NN})$

POS Induction Results

DT	JJ	NN	VBZ	IN	NN
The	green	cat	sleeps	at	home.

Key distribution: $P(\text{John}|\text{NN})$

Features:

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

POS Induction Results

DT	JJ	NN	VBZ	IN	NN
The	green	cat	sleeps	at	home.

Many-to-1 Accuracy

Features:

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

Data:

Train and test on entire WSJ

No tagging dictionary

45 POS tags

Berkeley N L P

POS Induction Results

DT	JJ	NN	VBZ	IN	NN
The	green	cat	sleeps	at	home.

Many-to-1 Accuracy

Features:

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

Data:

Train and test on entire WSJ

No tagging dictionary

45 POS tags

POS Induction Results

DT	JJ	NN	VBZ	IN	NN
The	green	cat	sleeps	at	home.

Features:

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

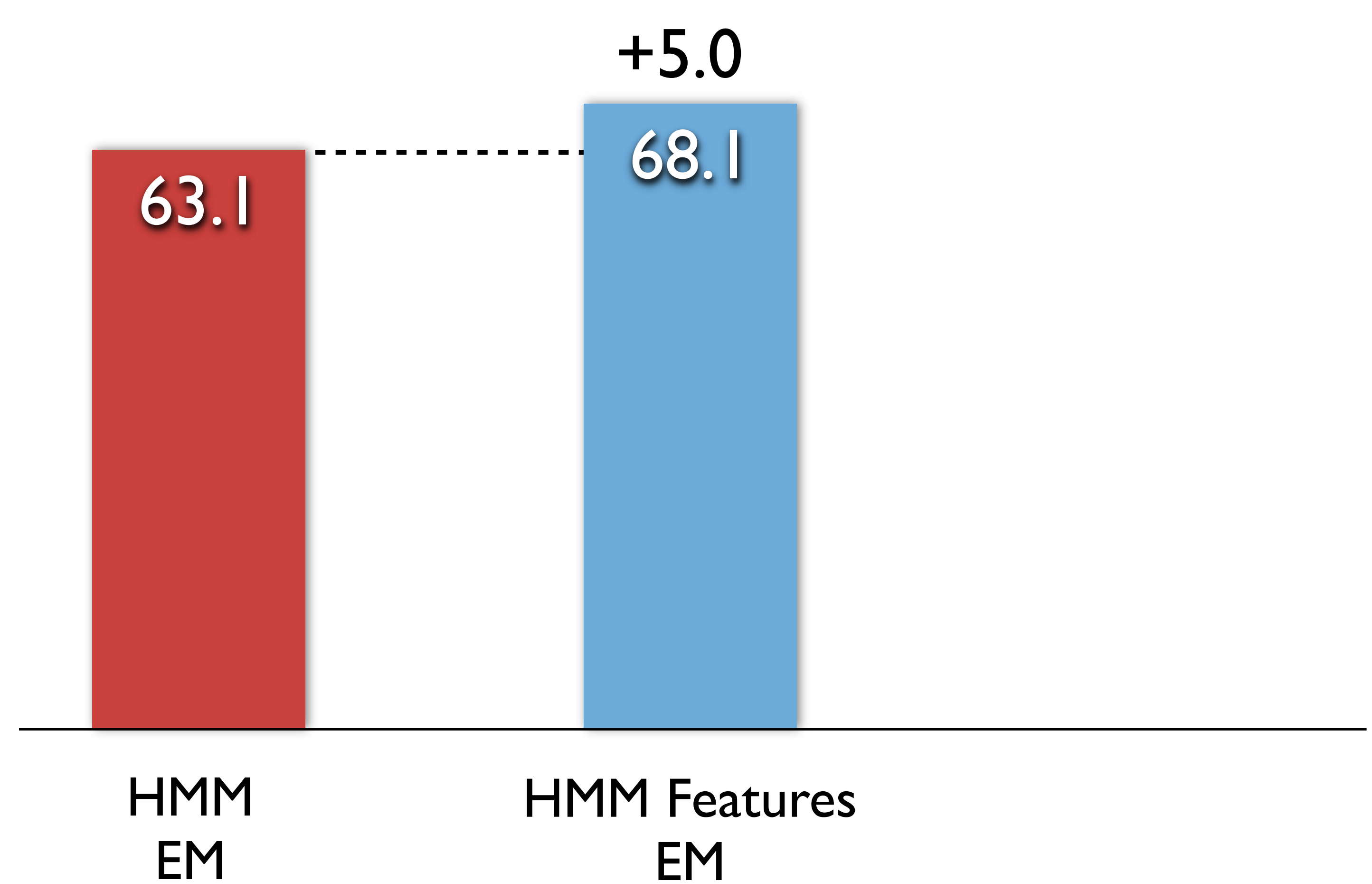
Data:

Train and test on entire WSJ

No tagging dictionary

45 POS tags

Many-to-1 Accuracy



POS Induction Results

DT	JJ	NN	VBZ	IN	NN
The	green	cat	sleeps	at	home.

Features:

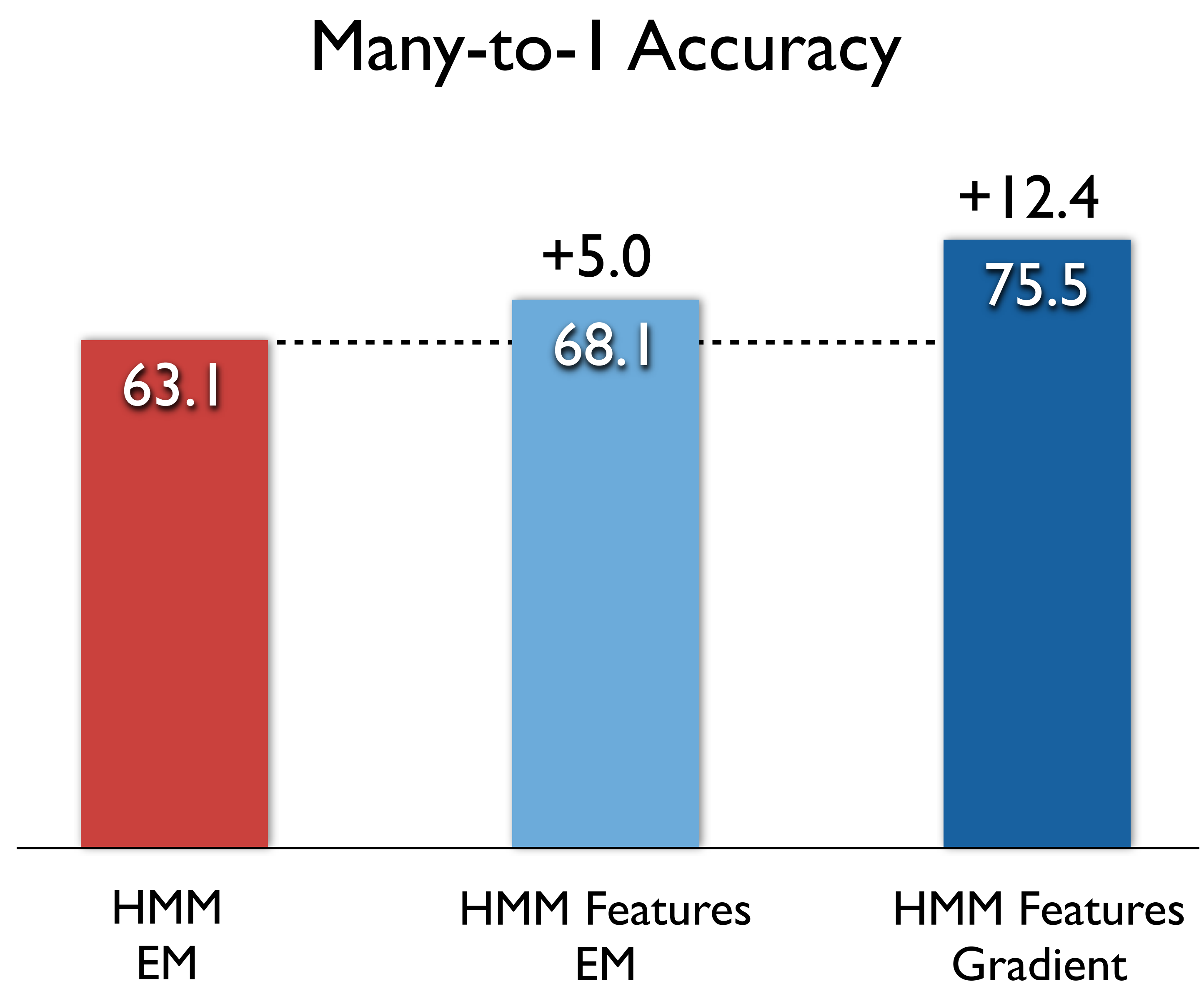
Basic:	John ∧ NNP
Contains-Digit:	+Digit ∧ NNP
Contains-Hyphen:	+Hyph ∧ NNP
Initial-Capital:	+Cap ∧ NNP
Suffix:	+ing ∧ NNP

Data:

Train and test on entire WSJ

No tagging dictionary

45 POS tags



POS Induction Results

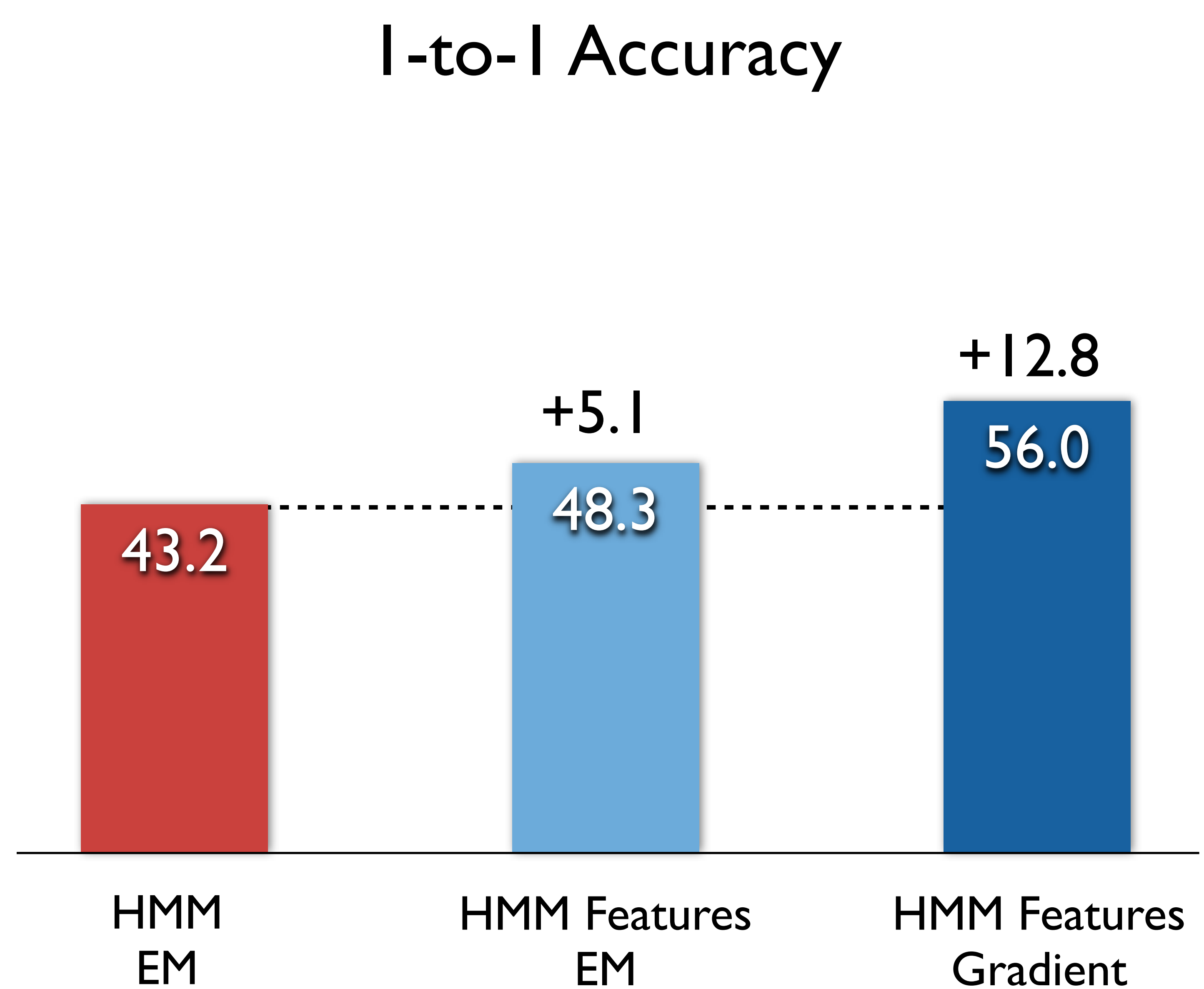
DT	JJ	NN	VBZ	IN	NN
The	green	cat	sleeps	at	home.

Features:

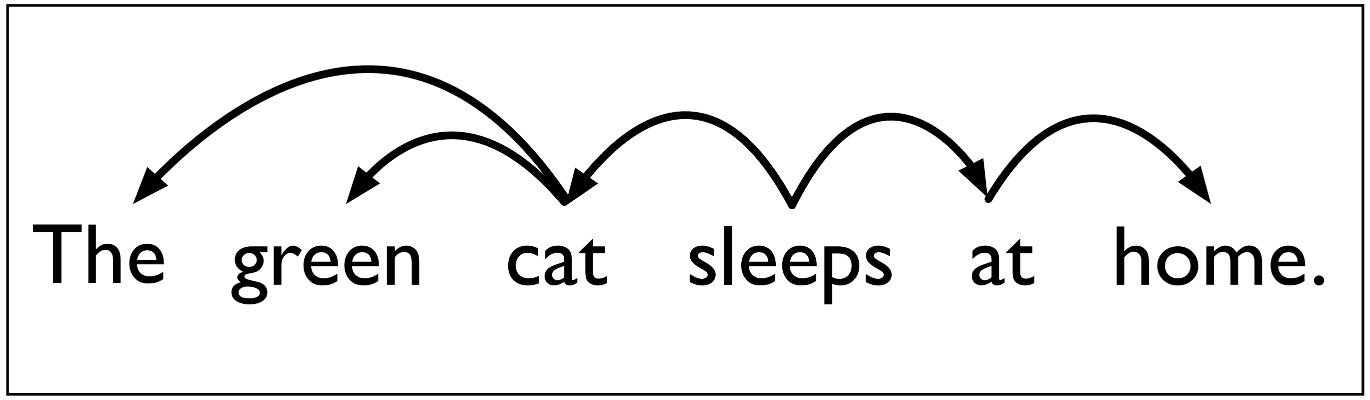
- Basic: John ∧ NNP
- Contains-Digit: +Digit ∧ NNP
- Contains-Hyphen: +Hyph ∧ NNP
- Initial-Capital: +Cap ∧ NNP
- Suffix: +ing ∧ NNP

Data:

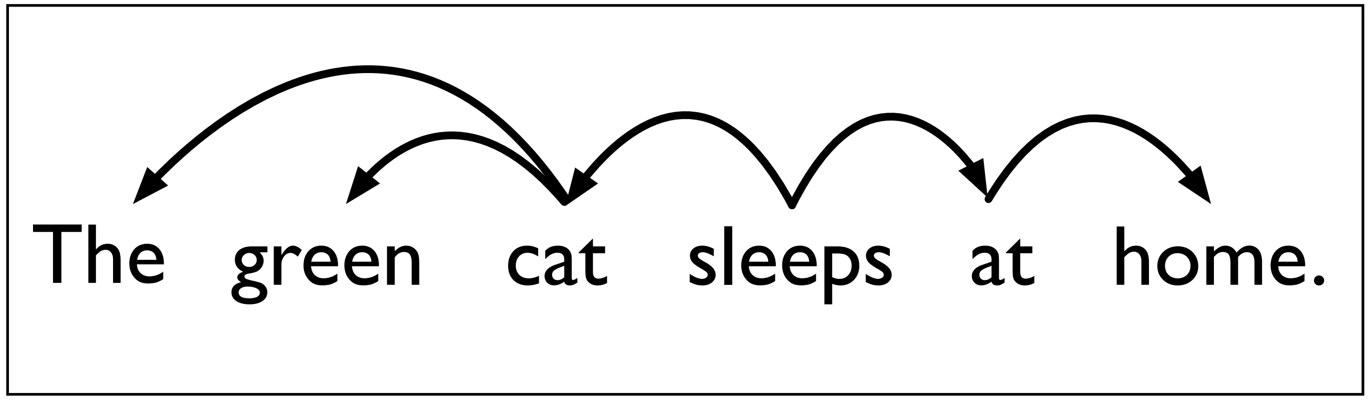
- Train and test on entire WSJ
- No tagging dictionary
- 45 POS tags



Grammar Induction Results

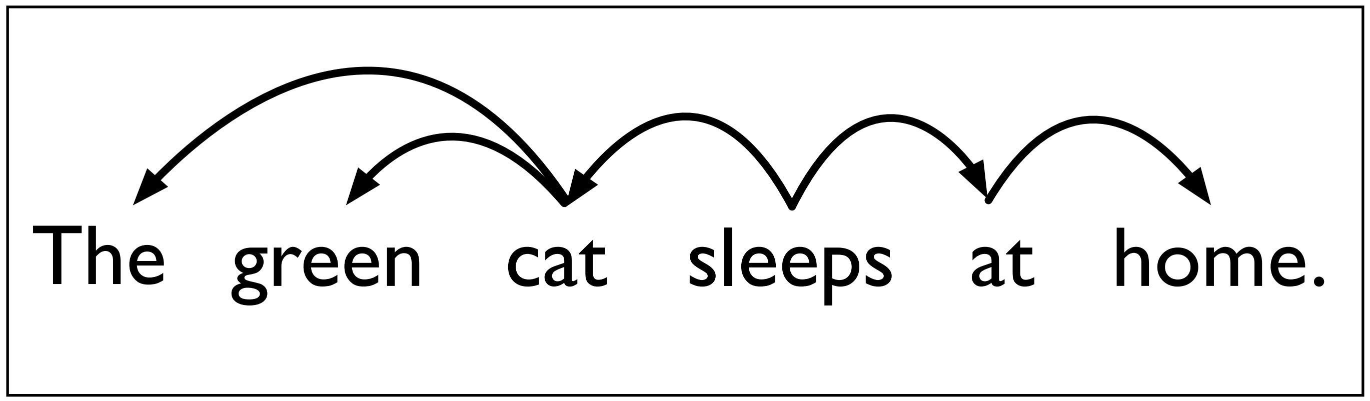


Grammar Induction Results



Key distributions: $P(\mathrm{JJ}|\mathrm{NN})$ $P(\mathrm{stop}|\mathrm{NN})$

Grammar Induction Results

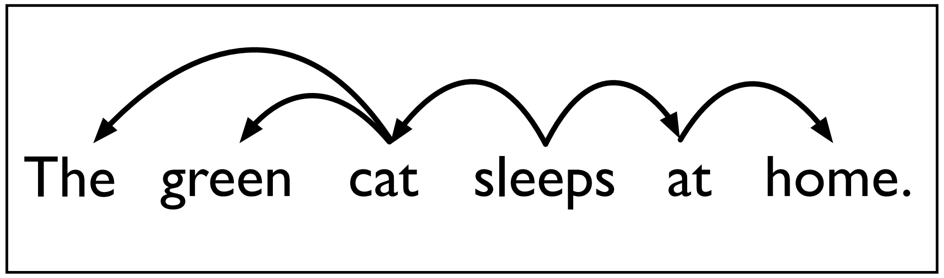


Key distributions: $P(\mathrm{JJ}|\mathrm{NN})$ $P(\mathrm{stop}|\mathrm{NN})$

Features:

Basic:	JJ ∧ NN, JJ ∧ NNS
Noun:	JJ ∧ Noun
Verb:	JJ ∧ Verb
Noun-verb:	JJ ∧ NounOrVerb

Grammar Induction Results



Features:

Basic:	JJ ∧ NN, JJ ∧ NNS
Noun:	JJ ∧ Noun
Verb:	JJ ∧ Verb
Noun-verb:	JJ ∧ NounOrVerb

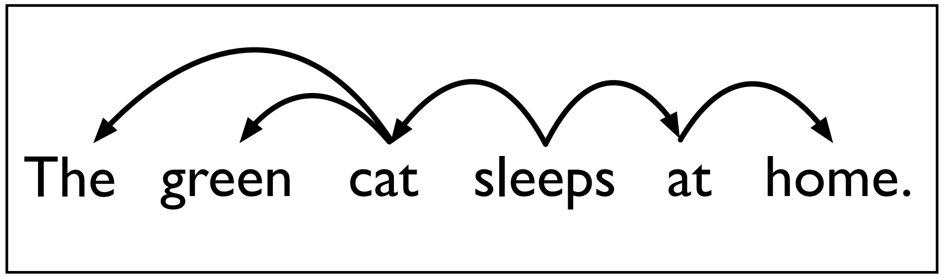
Data:

Train	WSJ10 Sec. 2-21 CTB10 Sec. 1-270
Tune	WSJ10 Sec. 22 CTB10 Sec. 400-454
Test	WSJ10 Sec. 23 CTB10 Sec. 271-300

English Directed Accuracy

Chinese Directed Accuracy

Grammar Induction Results

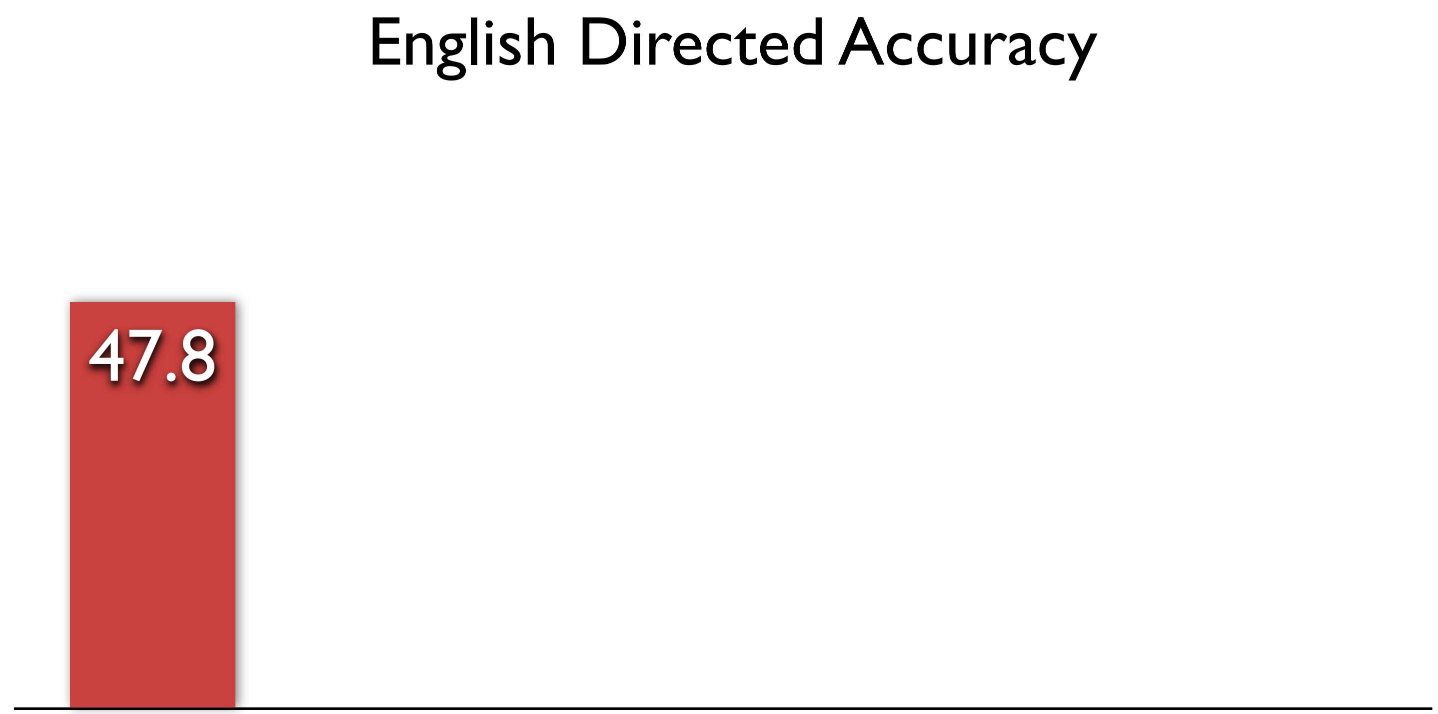


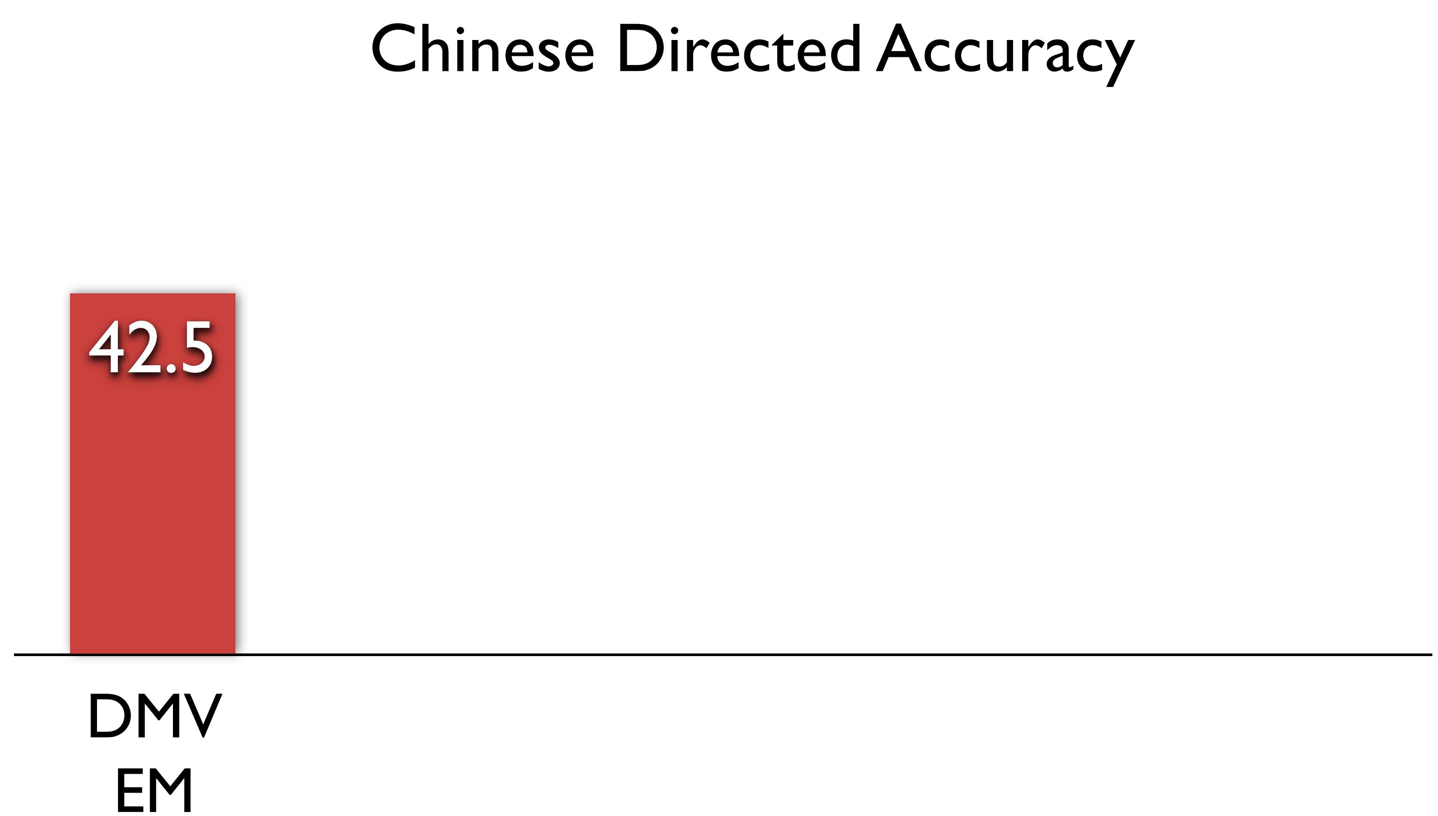
Features:

Basic:	JJ ∧ NN, JJ ∧ NNS
Noun:	JJ ∧ Noun
Verb:	JJ ∧ Verb
Noun-verb:	JJ ∧ NounOrVerb

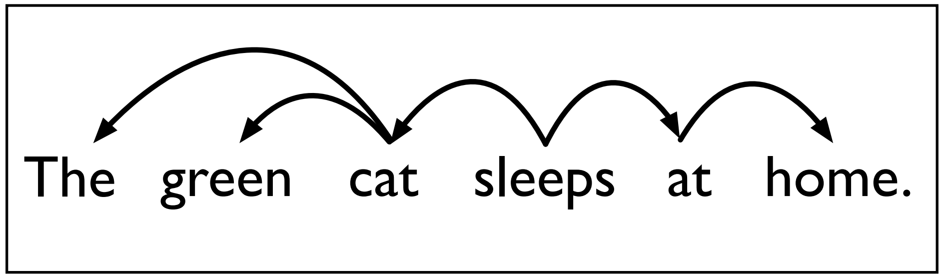
Data:

Train	WSJ10 Sec. 2-21 CTB10 Sec. 1-270
Tune	WSJ10 Sec. 22 CTB10 Sec. 400-454
Test	WSJ10 Sec. 23 CTB10 Sec. 271-300





Grammar Induction Results

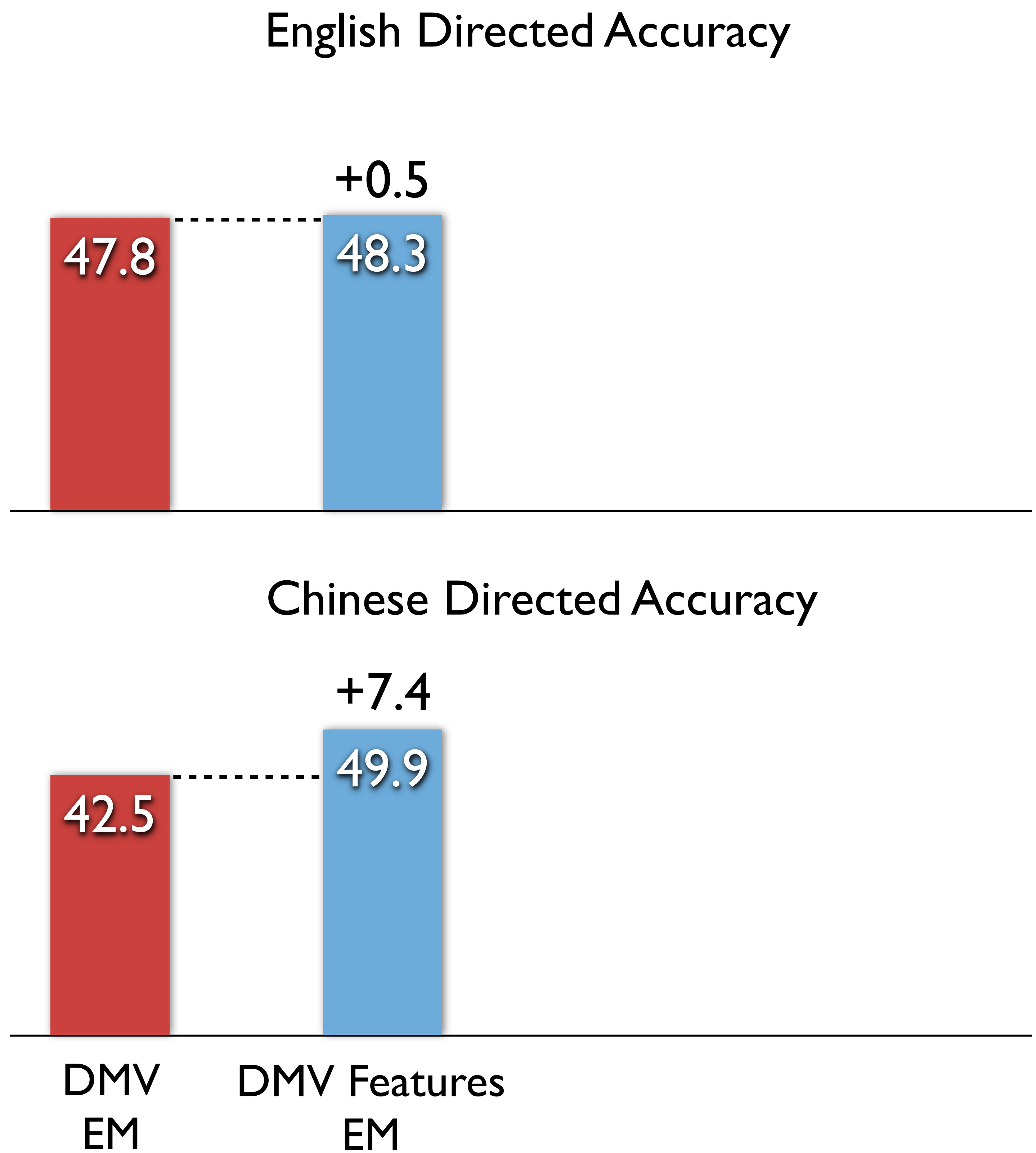


Features:

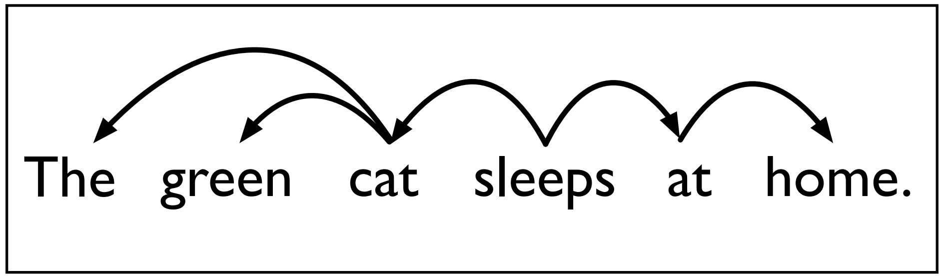
Basic:	JJ $\wedge$ NN, JJ $\wedge$ NNS
Noun:	JJ $\wedge$ Noun
Verb:	JJ $\wedge$ Verb
Noun-verb:	JJ $\wedge$ NounOrVerb

Data:

Train	WSJ10 Sec. 2-21 CTB10 Sec. 1-270
Tune	WSJ10 Sec. 22 CTB10 Sec. 400-454
Test	WSJ10 Sec. 23 CTB10 Sec. 271-300



Grammar Induction Results

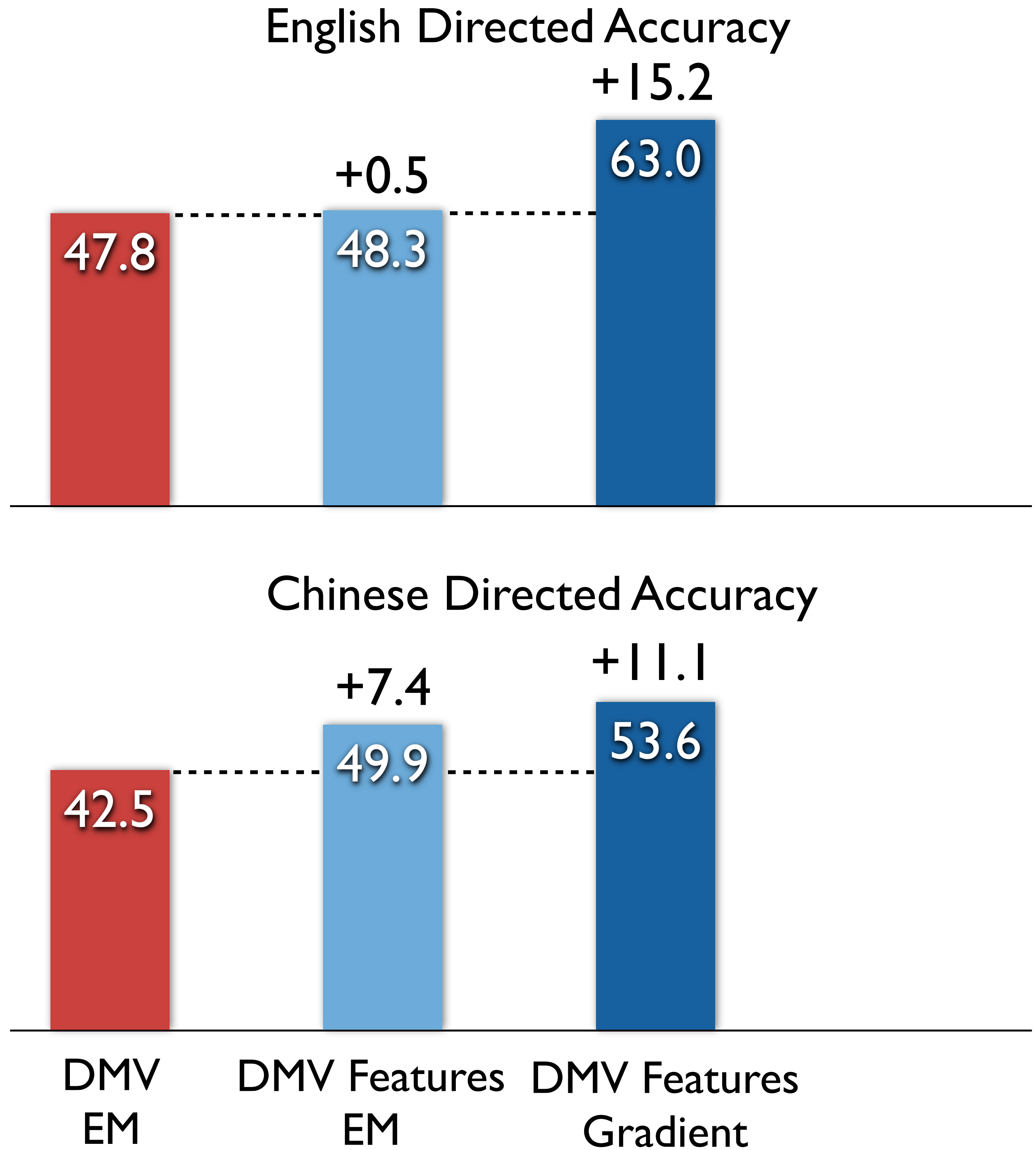


Features:

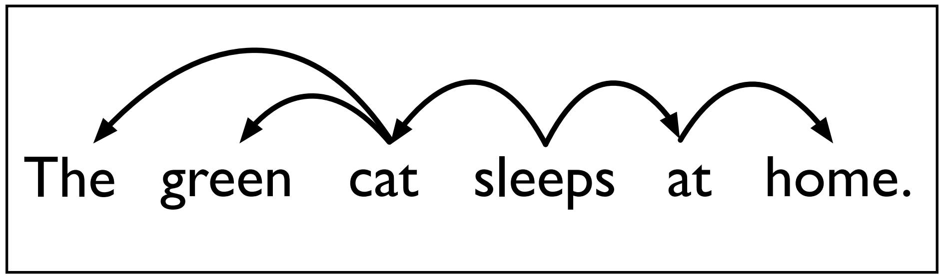
Basic:	JJ $\wedge$ NN, JJ $\wedge$ NNS
Noun:	JJ $\wedge$ Noun
Verb:	JJ $\wedge$ Verb
Noun-verb:	JJ $\wedge$ NounOrVerb

Data:

Train	WSJ10 Sec. 2-21 CTB10 Sec. 1-270
Tune	WSJ10 Sec. 22 CTB10 Sec. 400-454
Test	WSJ10 Sec. 23 CTB10 Sec. 271-300



Grammar Induction Results

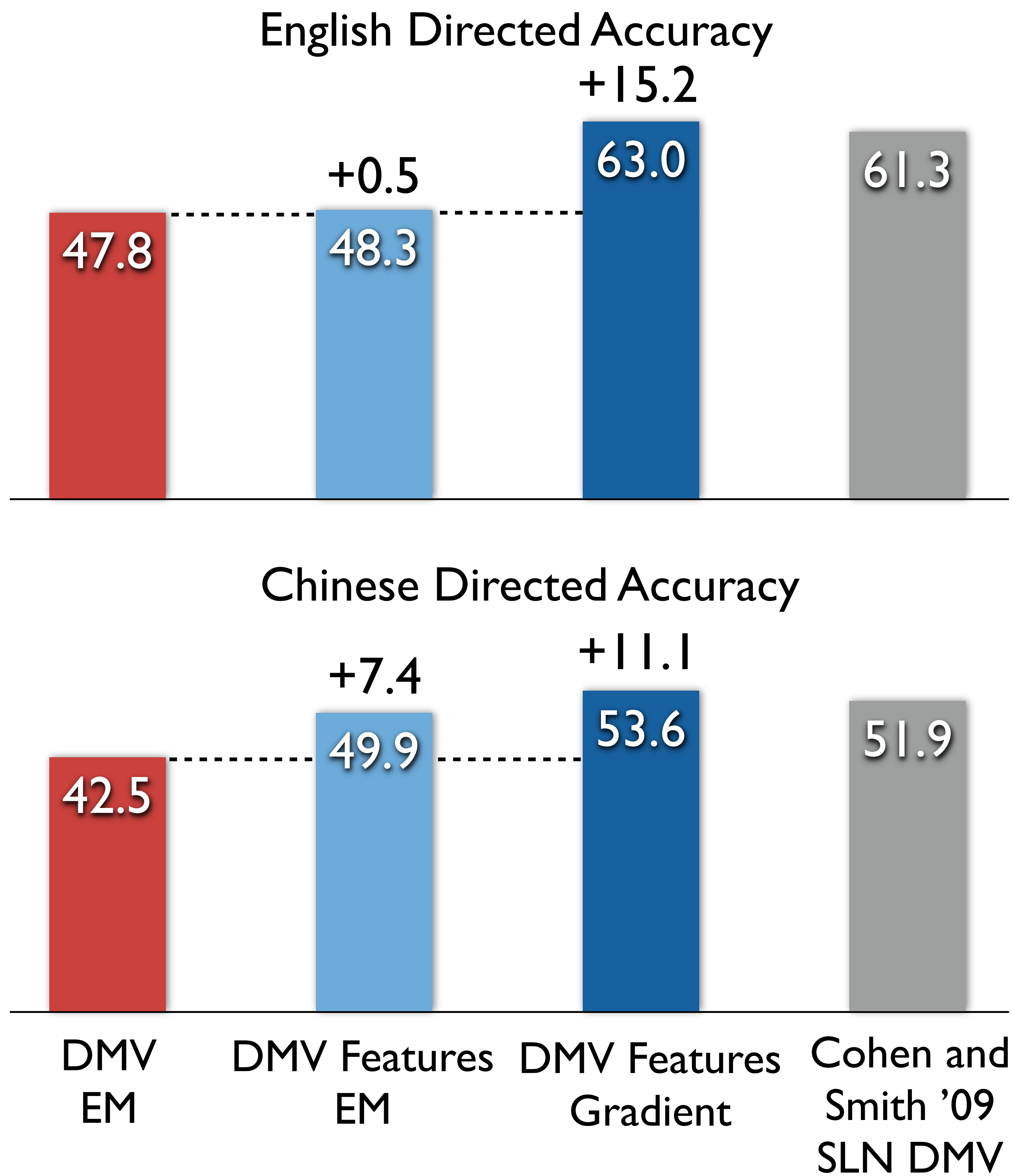


Features:

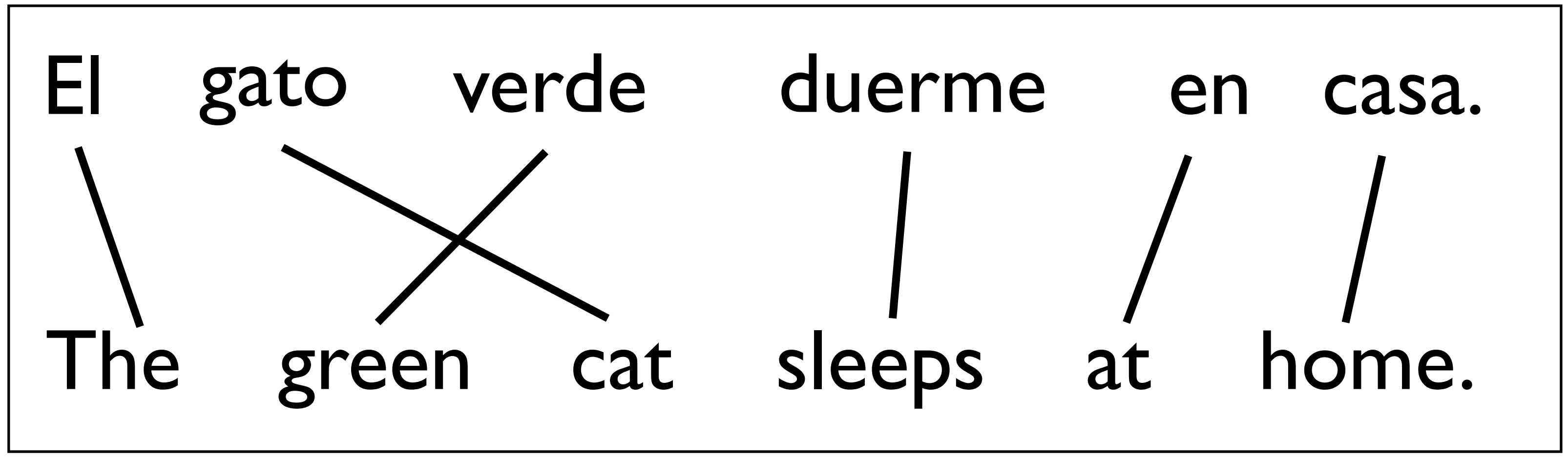
Basic:	JJ ∧ NN, JJ ∧ NNS
Noun:	JJ ∧ Noun
Verb:	JJ ∧ Verb
Noun-verb:	JJ ∧ NounOrVerb

Data:

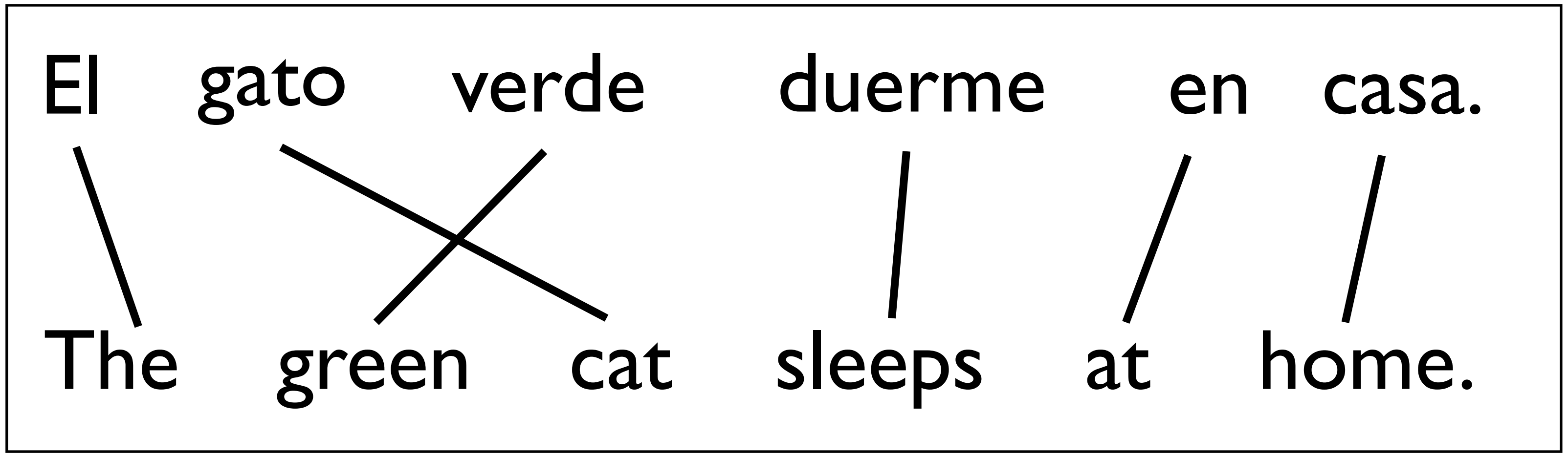
Train	WSJ10 Sec. 2-21 CTB10 Sec. 1-270
Tune	WSJ10 Sec. 22 CTB10 Sec. 400-454
Test	WSJ10 Sec. 23 CTB10 Sec. 271-300



Word Alignment Results

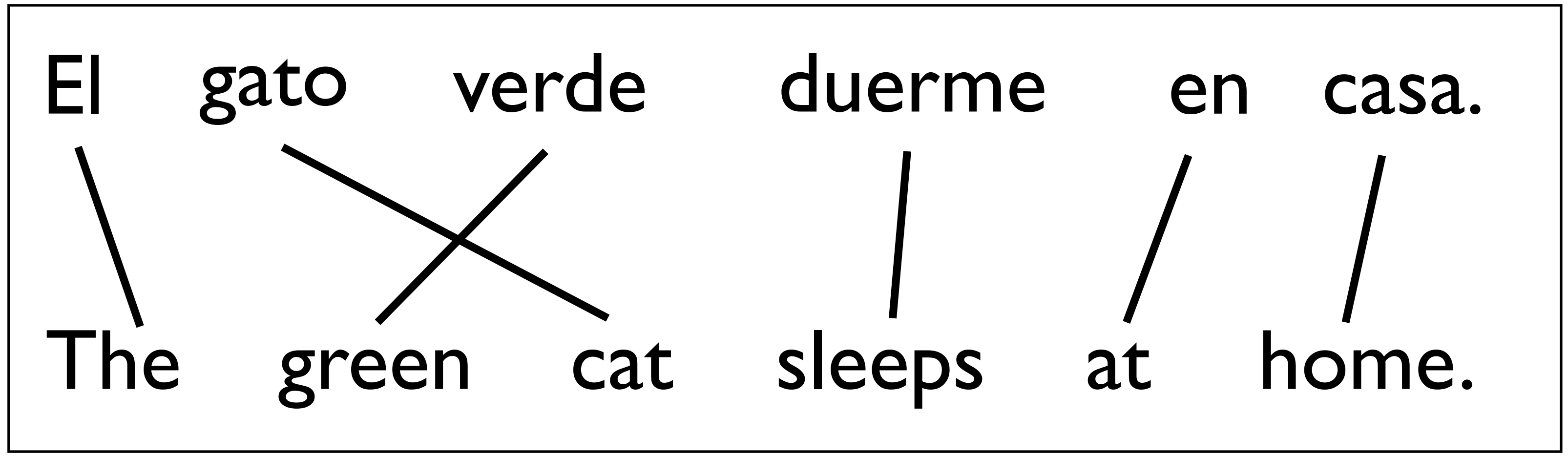


Word Alignment Results



Key distribution: $P(\mathrm{gato}|\mathrm{cat})$

Word Alignment Results

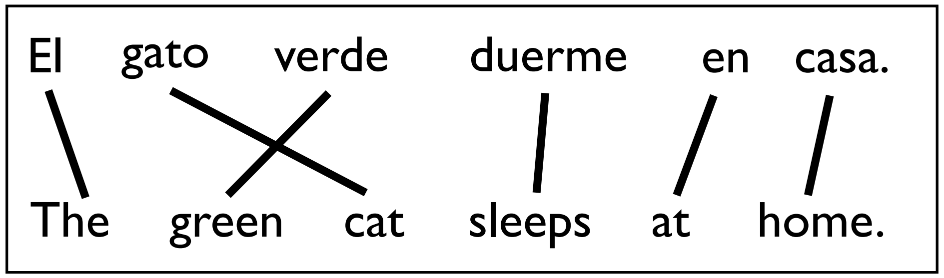


Key distribution: $P(\text{gato}|\text{cat})$

Features:

Basic:	gato ∧ cat
Edit-Distance:	edit(gato,cat) = 2
Dictionary:	(gato,cat) ∈ Dict
Stem:	gato ∧ +stem(cat)
Prefix:	gato ∧ +ca

Word Alignment Results



Alignment Error Rate

Features:

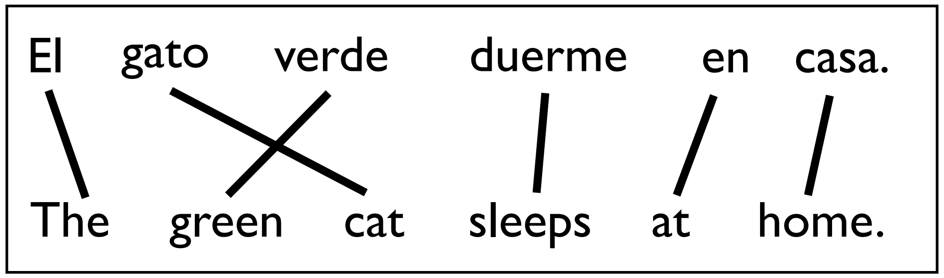
Basic:	gato $\wedge$ cat
Edit-Distance:	edit(gato,cat) = 2
Dictionary:	(gato,cat) $\in$ Dict
Stem:	gato $\wedge$ +stem(cat)
Prefix:	gato $\wedge$ +ca

Data:

Train 10K sentences of FBIS
Chinese-English newswire

Test NIST 2002 Chinese-English dev set

Word Alignment Results



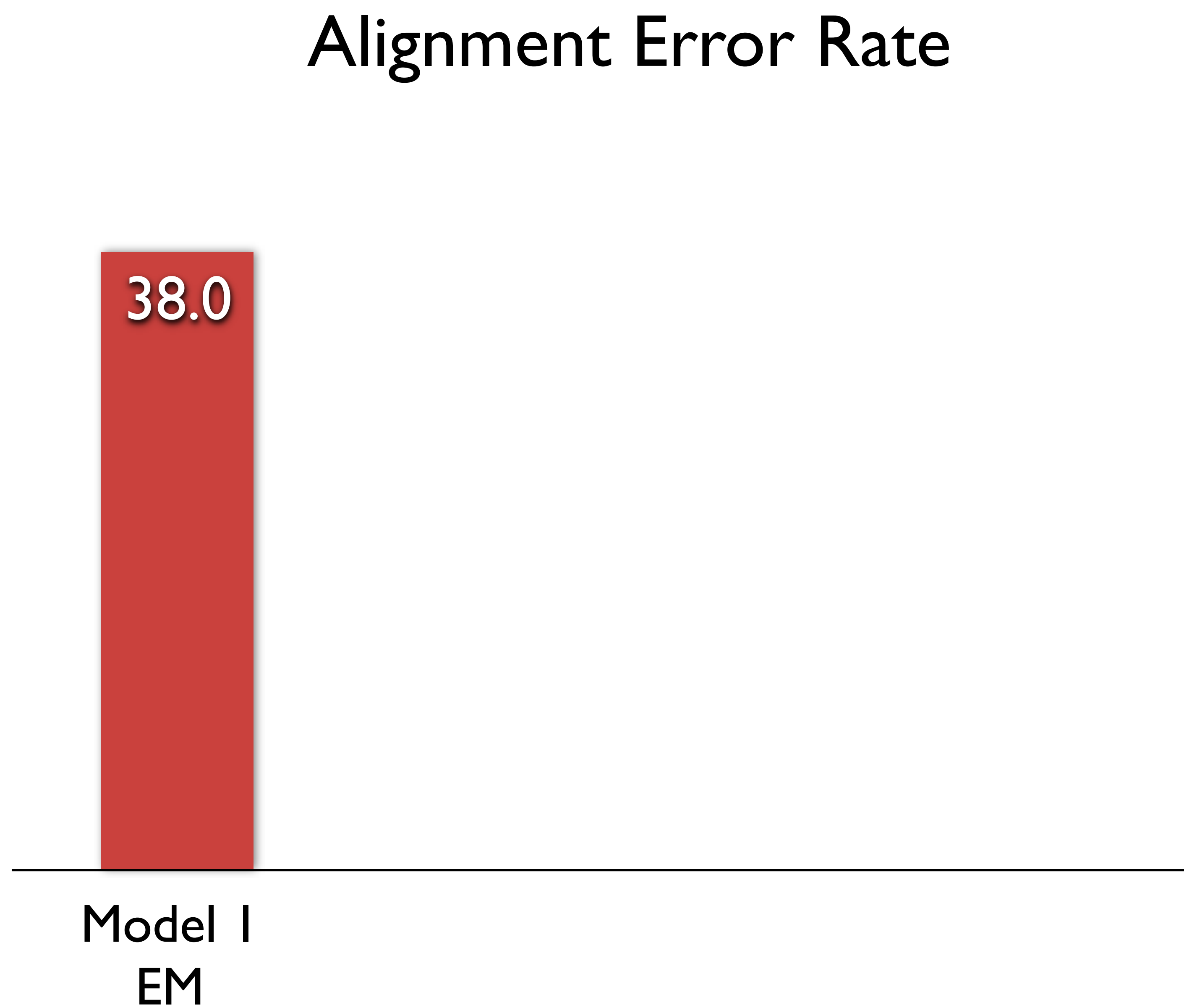
Features:

Basic:	gato ∧ cat
Edit-Distance:	edit(gato,cat) = 2
Dictionary:	(gato,cat) ∈ Dict
Stem:	gato ∧ +stem(cat)
Prefix:	gato ∧ +ca

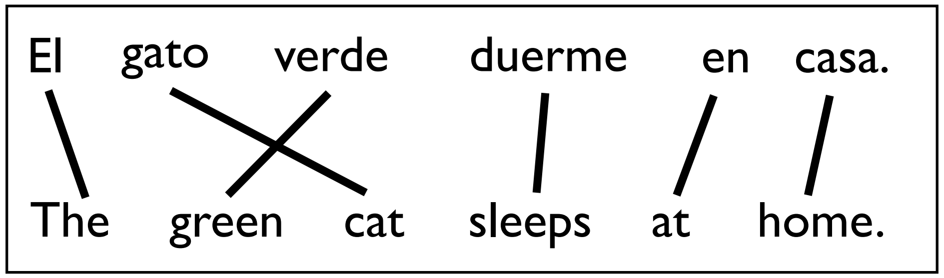
Data:

Train 10K sentences of FBIS Chinese-English newswire

Test NIST 2002 Chinese-English dev set



Word Alignment Results



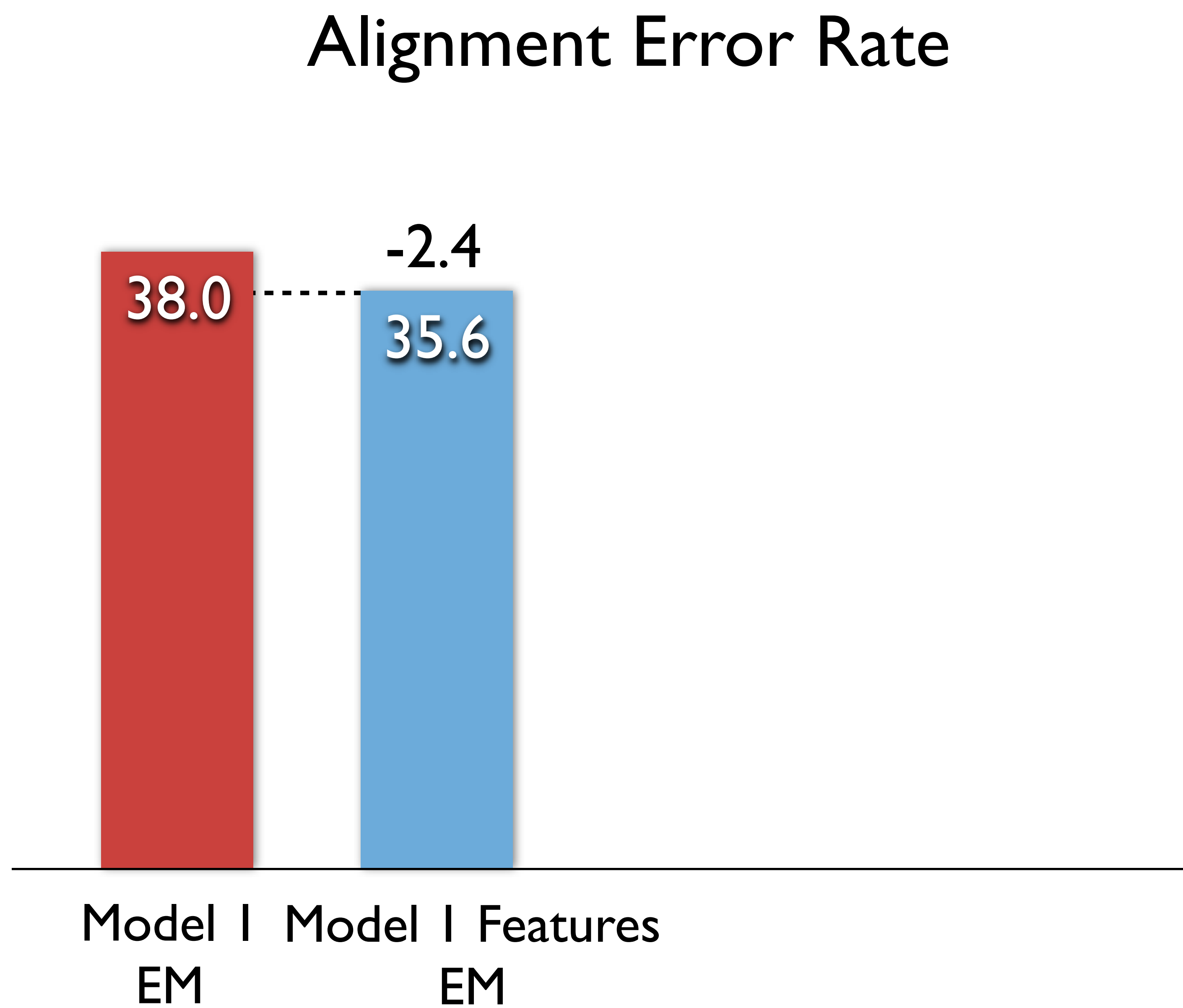
Features:

Basic:	gato ∧ cat
Edit-Distance:	edit(gato,cat) = 2
Dictionary:	(gato,cat) ∈ Dict
Stem:	gato ∧ +stem(cat)
Prefix:	gato ∧ +ca

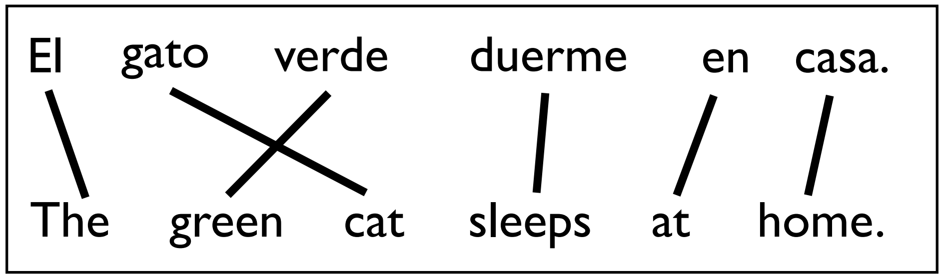
Data:

Train 10K sentences of FBIS Chinese-English newswire

Test NIST 2002 Chinese-English dev set



Word Alignment Results



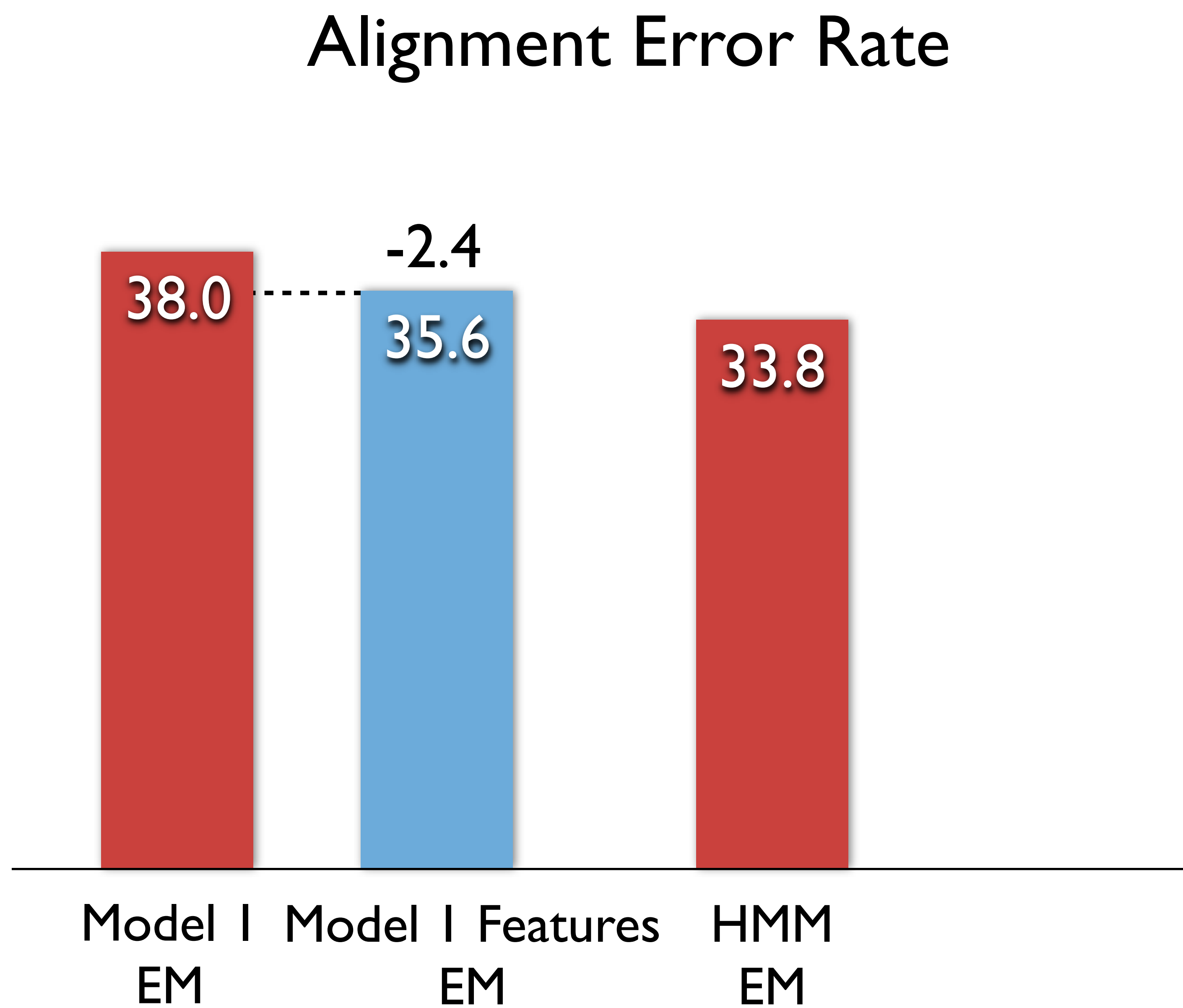
Features:

Basic:	gato ∧ cat
Edit-Distance:	edit(gato,cat) = 2
Dictionary:	(gato,cat) ∈ Dict
Stem:	gato ∧ +stem(cat)
Prefix:	gato ∧ +ca

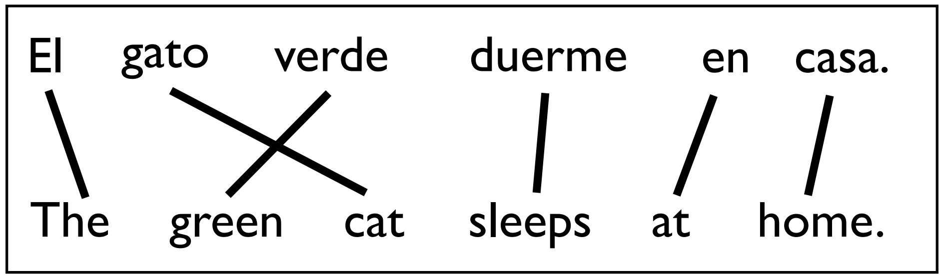
Data:

Train 10K sentences of FBIS Chinese-English newswire

Test NIST 2002 Chinese-English dev set



Word Alignment Results



Features:

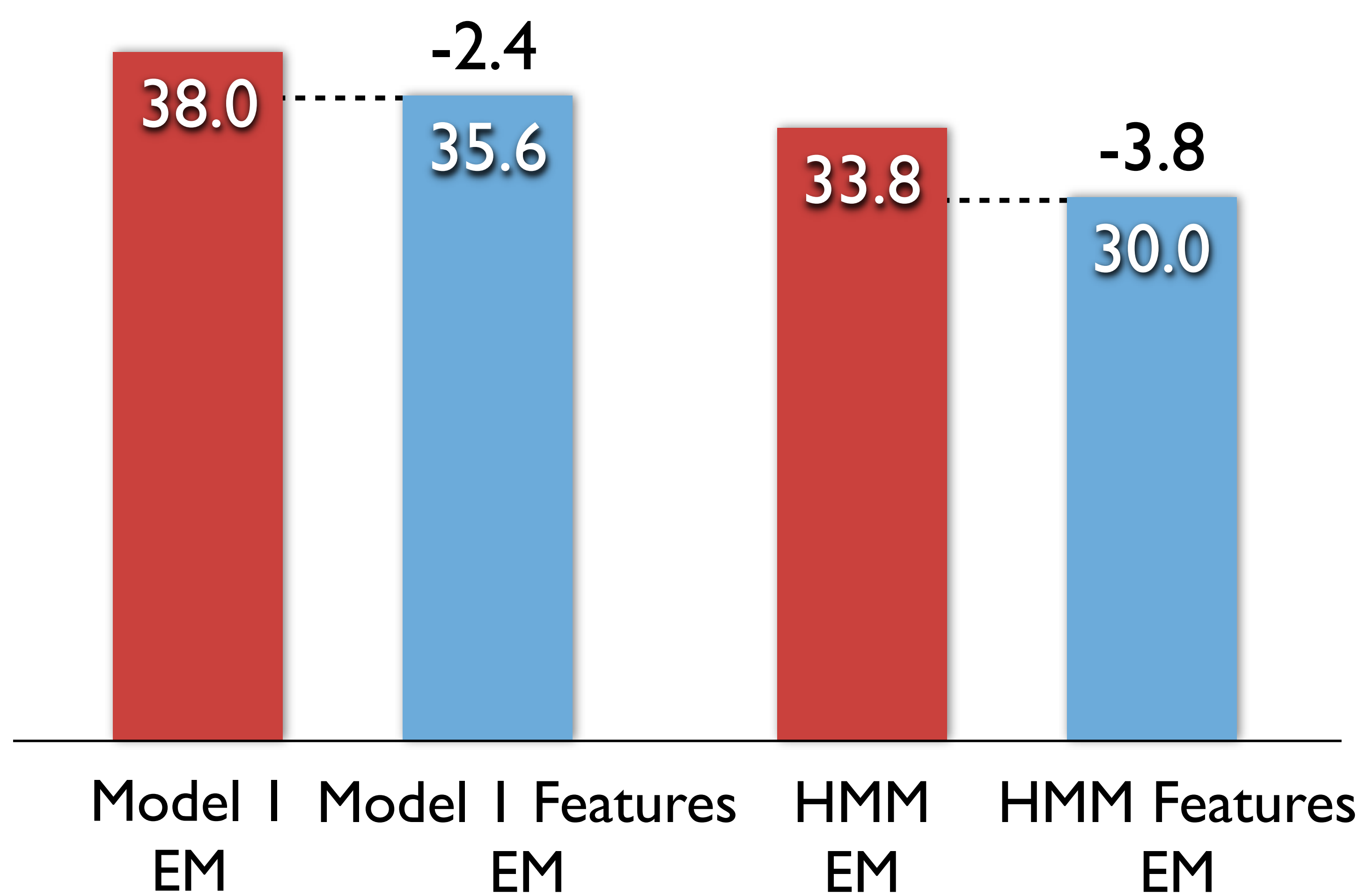
Basic:	gato ∧ cat
Edit-Distance:	edit(gato,cat) = 2
Dictionary:	(gato,cat) ∈ Dict
Stem:	gato ∧ +stem(cat)
Prefix:	gato ∧ +ca

Data:

Train 10K sentences of FBIS Chinese-English newswire

Test NIST 2002 Chinese-English dev set

Alignment Error Rate



Word Segmentation Results

[T h e][g r e e n][c a t]

Word Segmentation Results

[T h e][g r e e n][c a t]

Key distribution: $P(\mathrm{running})$

Word Segmentation Results

[T h e][g r e e n][c a t]

Key distribution: $P(\text{running})$

Features:

Basic:	running
Length:	length(running) = 7
Num-Vowels:	numV(running) = 2
Coarse-Phono-Prefix:	+rAn
Coarse-Phono-Suffix:	+IN

Word Segmentation Results

[T h e][g r e e n][c a t]

Token F1

Features:

Basic:	running
Length:	length(running) = 7
Num-Vowels:	numV(running) = 2
Coarse-Phono-Prefix:	+rAn
Coarse-Phono-Suffix:	+IN

Data:

Train and test on phonetic version of Bernstein-Ratner corpus

Word Segmentation Results

[T h e][g r e e n][c a t]

Features:

Basic:	running
Length:	length(running) = 7
Num-Vowels:	numV(running) = 2
Coarse-Phono-Prefix:	+rAn
Coarse-Phono-Suffix:	+IN

Data:

Train and test on phonetic version of Bernstein-Ratner corpus

Token F1

Word Segmentation Results

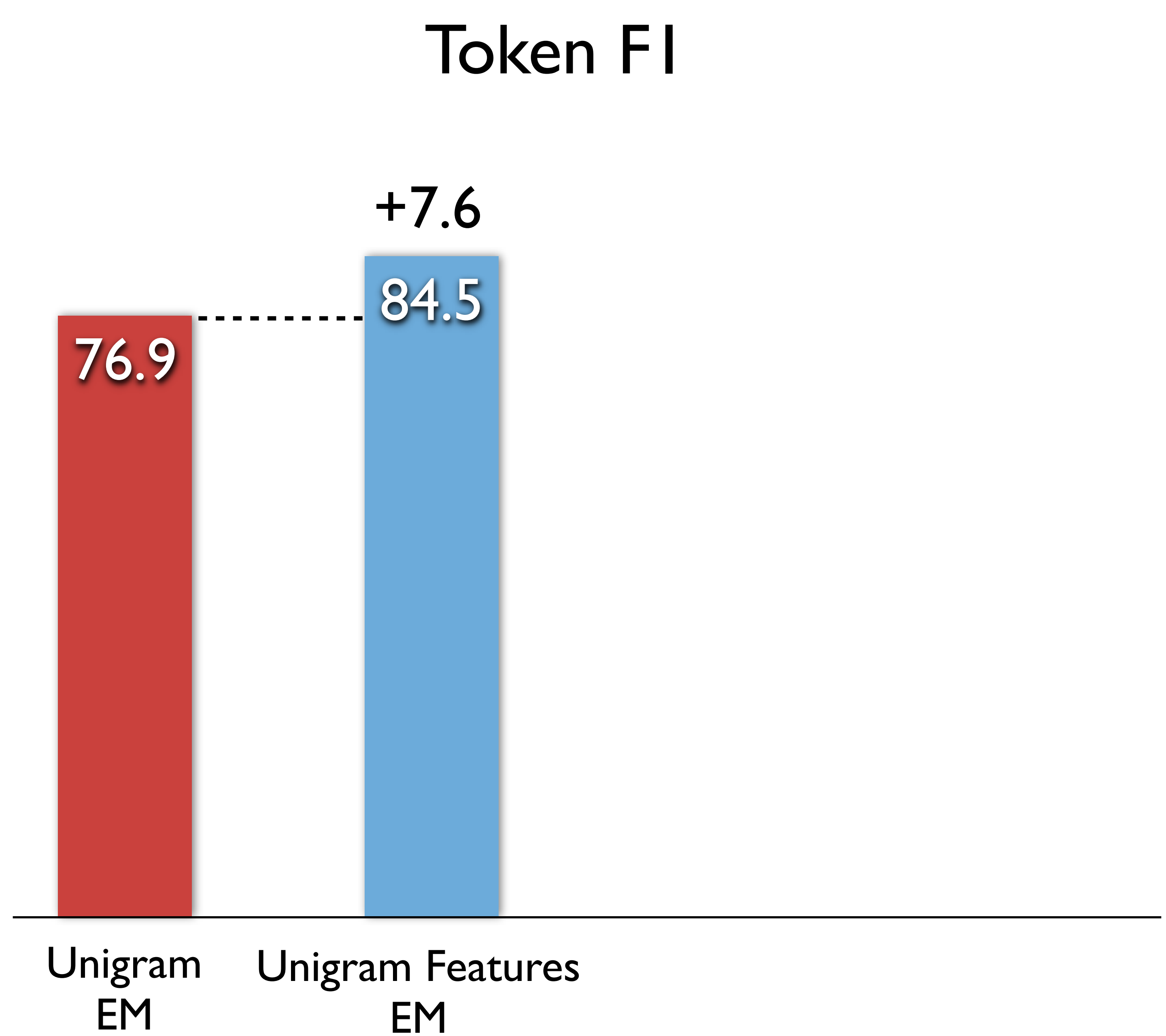
[T h e][g r e e n][c a t]

Features:

Basic:	running
Length:	length(running) = 7
Num-Vowels:	numV(running) = 2
Coarse-Phono-Prefix:	+rAn
Coarse-Phono-Suffix:	+IN

Data:

Train and test on phonetic version of Bernstein-Ratner corpus



Word Segmentation Results

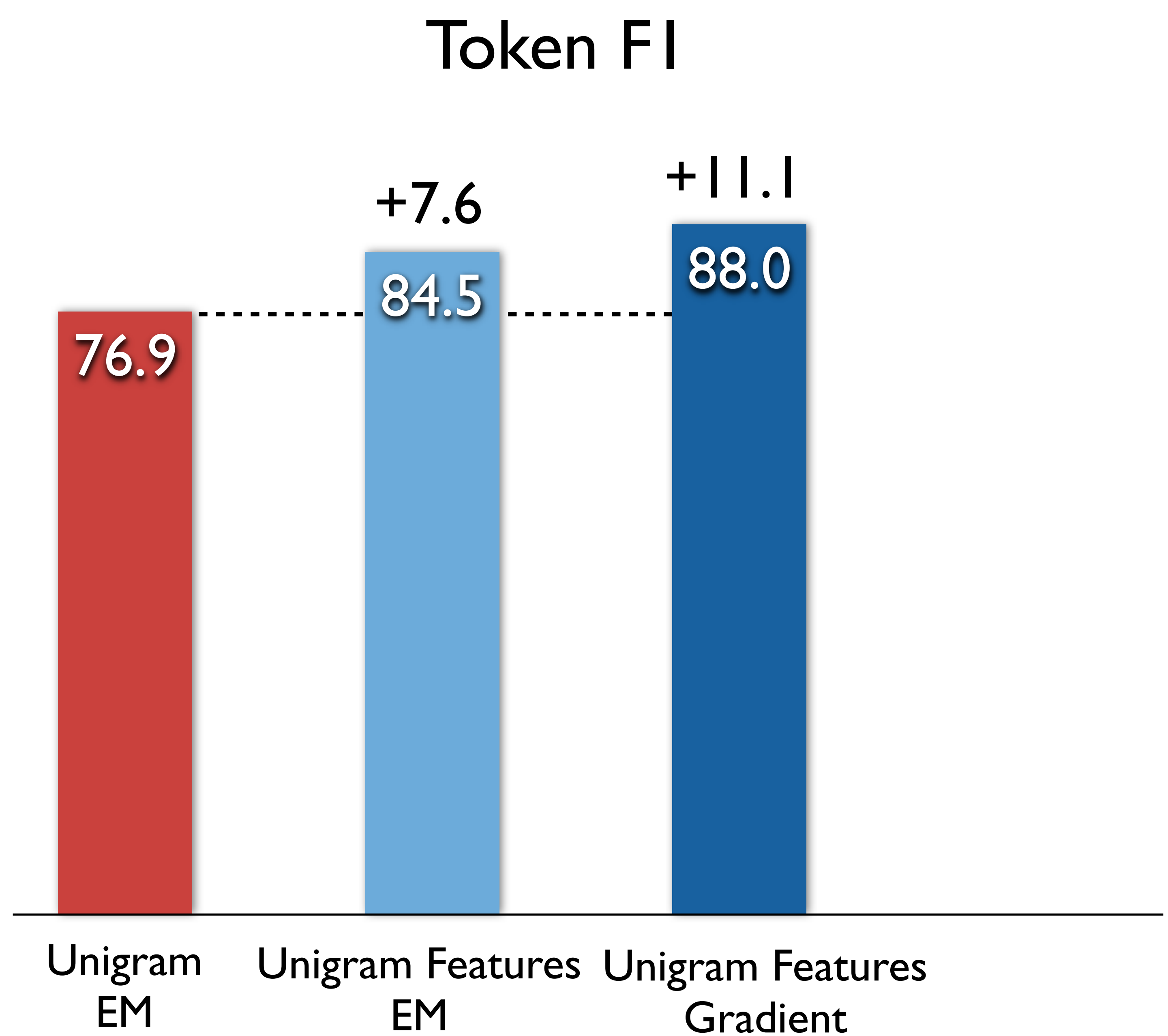
[T h e][g r e e n][c a t]

Features:

Basic:	running
Length:	length(running) = 7
Num-Vowels:	numV(running) = 2
Coarse-Phono-Prefix:	+rAn
Coarse-Phono-Suffix:	+IN

Data:

Train and test on phonetic version of Bernstein-Ratner corpus



Word Segmentation Results

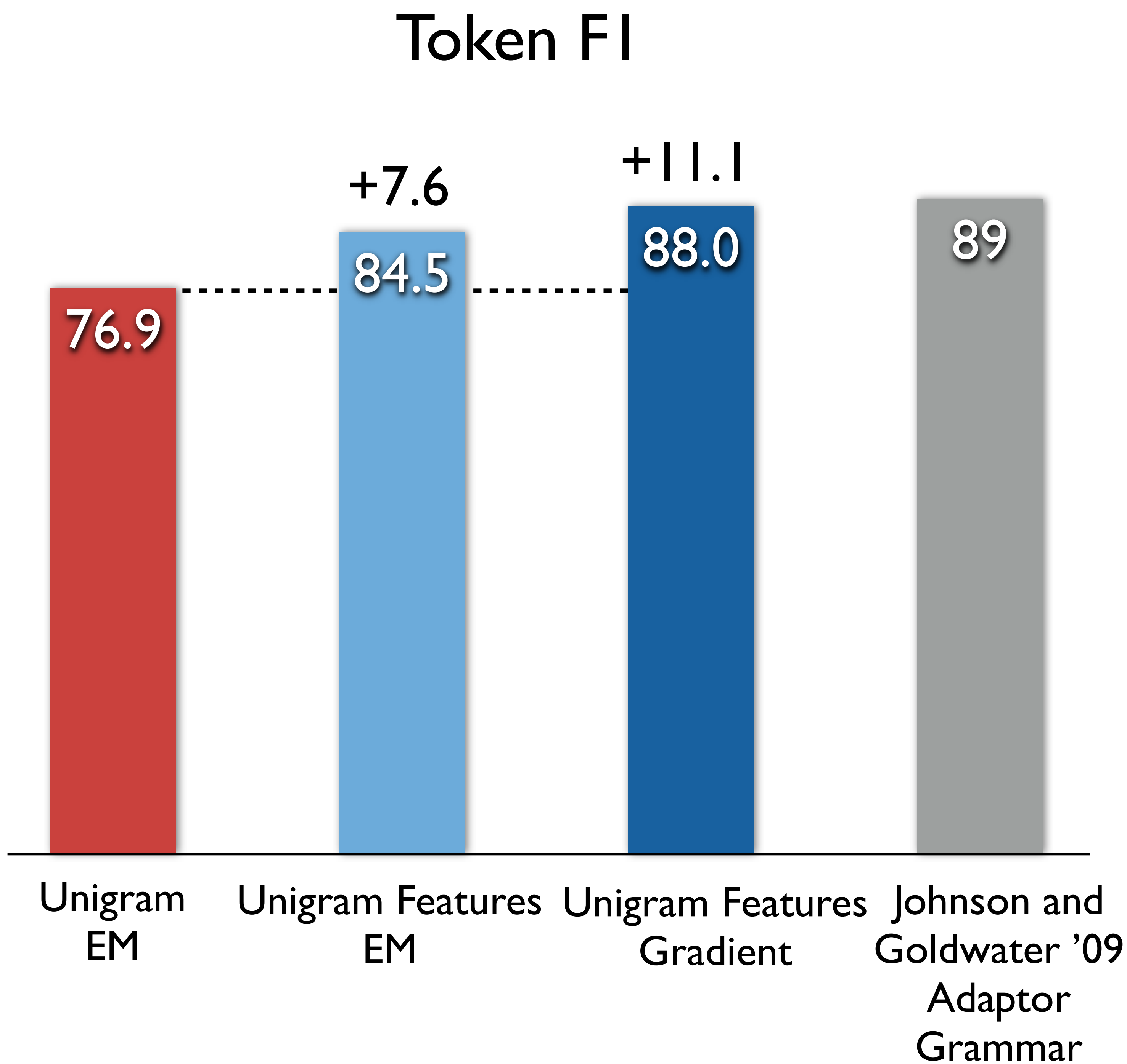
[T h e][g r e e n][c a t]

Features:

Basic:	running
Length:	length(running) = 7
Num-Vowels:	numV(running) = 2
Coarse-Phono-Prefix:	+rAn
Coarse-Phono-Suffix:	+IN

Data:

Train and test on phonetic version of Bernstein-Ratner corpus



Apply to New Models

1. Take a generative model

Apply to New Models

1. Take a generative model

2. Brainstorm features local to the component multinomials

Apply to New Models

1. Take a generative model
2. Brainstorm features local to the component multinomials
3. Run this algorithm

Apply to New Models

1. Take a generative model
2. Brainstorm features local to the component multinomials
3. Run this algorithm
4. Crush your baseline

Conclusion

- State-of-the-art results

Conclusion

- State-of-the-art results
- Can implemented using off-the-shelf NLP tools

Conclusion

- State-of-the-art results
- Can implemented using off-the-shelf NLP tools
- Directly optimizing data-likelihood can outperform EM

Conclusion

- State-of-the-art results
- Can implemented using off-the-shelf NLP tools
- Directly optimizing data-likelihood can outperform EM
- Works on a wide range of induction tasks

Conclusion

Thanks!