Painless Unsupervised Learning with Features

Berkeley

Taylor Berg-Kirkpatrick  Alexandre Bouchard-Côté  John DeNero  Dan Klein
Basic HMM for POS Induction

\[ x_{i-1} \rightarrow z_{i-1} \rightarrow z_i \rightarrow z_{i+1} \]

\[ x_i \]

\[ x_{i+1} \]
Basic HMM for POS Induction

Transition distribution:

\[ P(z' | z) \]
Basic HMM for POS Induction

Emission distribution:

\[ P(x|z) \]
Parameterization

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

\[\mathcal{X}\]

- John
- Mary
- running
- jumping
Parameterization

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

| $\theta_{x|\text{NNP}}$ | $x$    |
|------------------------|-------|
| 0.1                    | John  |
| 0.0                    | Mary  |
| 0.2                    | running |
| 0.0                    | jumping |
Parameterization

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

\[
\begin{array}{ccc}
\theta_{x|\text{NNP}} & x & f \\
0.1 & \text{John} & +\text{Cap} \\
0.0 & \text{Mary} & +\text{Cap} \\
0.2 & \text{running} & +\text{ing} \\
0.0 & \text{jumping} & +\text{ing} \\
\end{array}
\]
### Parameterization

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

| \( \theta_{x|\text{NNP}} \) | \( x \) | \( f \) | \( e^{w^T f} \) |
|---|---|---|---|
| 0.1 | John | +Cap | 0.3 |
| 0.0 | Mary | +Cap | 0.3 |
| 0.2 | running | +ing | 0.1 |
| 0.0 | jumping | +ing | 0.1 |

**W:**
- +Cap: +1.2
- +ing: -0.3
Parameterization

\[ \theta_{x|z} = \frac{\exp(w^T f(x, z))}{\sum_{x'} \exp(w^T f(x', z))} \]
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
Basic Multinomial:
John ∧ NNP

POS Induction Accuracy

43.2
POS Induction Accuracy

Basic Multinomial:
John ∧ NNP

Rich Features:
John ∧ NNP
+Digit ∧ NNP
+Hyph ∧ NNP
+Cap ∧ NNP
+ing ∧ NNP

43.2

56.0 +12.8
Hard EM without Features

E-Step

θ

M-Step

Z
**Hard EM without Features**

**E-Step: Dynamic Program**

\[ z \leftarrow \arg\max_{z} P(z|x; \theta) \]

**M-Step: Divide Counts**

\[
\theta \leftarrow \arg\max_{\theta} P(x, z; \theta) = \left[ \frac{c(z \to x)}{c(z \to \cdot)}, \ldots \right]
\]
Hard EM with Features

E-Step: Dynamic Program
\[ z \leftarrow \arg\max_z P(z|x; \theta) \]

M-Step: Divide Counts
\[ \theta \leftarrow \arg\max_\theta P(x, z; \theta) \]
\[ = \left[ \frac{c(z \rightarrow x)}{c(z \rightarrow \cdot)}, \ldots \right] \]
**Hard EM with Features**

**E-Step: Dynamic Program**

\[ z \leftarrow \operatorname{argmax}_z P(z|x; \theta) \]

**M-Step: Divide Counts**

\[ \theta \leftarrow \operatorname{argmax}_\theta P(x, z; \theta) \]

\[ = \left[ \frac{c(z \rightarrow x)}{c(z \rightarrow \cdot)}, \ldots \right] \]
Hard EM with Features

E-Step: Dynamic Program
\[ z' \leftarrow \arg\max_z P(z|x; \theta) \]

M-Step: Train Maxent
\[ w' \leftarrow \arg\max_w \log P(x, z; w) \]
Hard EM with Features

\[
\log P(x, z; w) = \sum_i \log P(x_i | z_i; w) + \ldots
\]
Hard EM with Features

$$\log P(x, z; w)$$

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

Maxent training example
Hard EM with Features

\[
\log P(x, z; w)
\]

\[
= \sum_{i} \log P(x_i | z_i; w) + \ldots
\]

Maxent training example

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

Multiplicity
Hard EM with Features

E-Step: Dynamic Program
\[
\mathbf{z} \leftarrow \text{argmax}_{\mathbf{z}} P(\mathbf{z}|\mathbf{x}; \theta)
\]

M-Step: Train Maxent
\[
\mathbf{w} \leftarrow \text{argmax}_{\mathbf{w}} \log P(\mathbf{x}, \mathbf{z}; \mathbf{w})
\]
Hard EM with Features

E-Step: Dynamic Program
\[ z \leftarrow \arg\max_z P(z|x; \theta) \]

M-Step: Train Maxent
\[ w \leftarrow \arg\max_w \log P(x, z; w) \]

Transform Parameters
\[ \theta_{x|z} \leftarrow \frac{\exp(w^T f(x, z))}{\sum_{x'} \exp(w^T f(x', z))} \]
EM with Features

E-Step: Dynamic Program
\[ z \leftarrow \arg\max_z P(z|x; \theta) \]

M-Step: Train Maxent
\[ w \leftarrow \arg\max_w \log P(x, z; w) \]

Transform Parameters
\[ \theta_{x|z} \leftarrow \frac{\exp(w^T f(x, z))}{\sum_{x'} \exp(w^T f(x', z))} \]
EM with Features

**E-Step: Dynamic Program**

e(z → x) ← E[c(z → x)]

**M-Step: Train Maxent**

w ← argmax \( w \log P(x, z; w) \)

**Transform Parameters**

\[ \theta_{x|z} ← \frac{\exp(w^T f(x, z))}{\sum_{x'} \exp(w^T f(x', z))} \]
EM with Features

**E-Step: Dynamic Program**

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

**M-Step: Train Maxent**

\[ w \leftarrow \text{argmax}_w \mathbb{E} \left[ \log P(x, z; w) \right] \]

**Transform Parameters**

\[ \theta_{x|z} \leftarrow \frac{\exp(w^T f(x, z))}{\sum_{x'} \exp(w^T f(x', z))} \]
Algorithm 1

Initialize probabilities $\theta$

repeat

- Compute expected counts $e$
- Fit parameters $\theta$

until convergence
EM without Features

\[ L(w) \]

Initialize probabilities \( \theta \)

repeat
- Compute expected counts \( e \)
- Fit parameters \( \theta \)
until convergence

Algorithm 2

EM with Features

Initialize weights \( w \)

repeat
- Compute expected counts \( e \)
- Fit parameters \( w \)
- Transform \( w \) to \( \theta \)
until convergence

Algorithm 3

EM with Features

Initialize weights \( w \)

repeat
- Compute expected counts \( e \)
- Compute \( \nabla L(w, e) \)
- \( w \leftarrow \text{climb}(w, L(w, e), \nabla L(w, e)) \)
- Transform \( w \) to \( \theta \)
until convergence

Algorithm 4

Direct Gradient with Features

Initialize weights \( w \)

repeat
- Compute expected counts \( e \)
- Compute \( L(w) \)
- Compute \( \nabla L(w, e) \)
- \( w \leftarrow \text{climb}(w, L(w, e), \nabla L(w, e)) \)
- Transform \( w \) to \( \theta \)
until convergence
Algorithm 1

**EM without Features**

1. Initialize probabilities $\theta$
2. repeat
   - Compute expected counts $e$
   - Fit parameters $\theta$
3. until convergence

Algorithm 2

**EM with Features**

1. Initialize weights $w$
2. repeat
   - Compute expected counts $e$
   - Fit parameters $w$
3. Transform $w$ to $\theta$
4. until convergence

Algorithm 3

**EM with Features**

1. Initialize weights $w$
2. repeat
   - Compute expected counts $e$
   - Compute $\nabla L(w, e)$
3. $w \leftarrow \text{climb}(w, L(w, e), \nabla L(w, e))$
4. Transform $w$ to $\theta$
5. until convergence

Algorithm 4

**Direct Gradient with Features**

1. Initialize weights $w$
2. repeat
   - Compute expected counts $e$
   - Compute $L(w)$
   - Compute $\nabla L(w, e)$
3. $w \leftarrow \text{climb}(w, L(w), \nabla L(w, e))$
4. Transform $w$ to $\theta$
5. until convergence
Initialize probabilities $\theta$

repeat
  Compute expected counts $e$
  Fit parameters $\theta$
until convergence

Transform $w$ to $\theta$

until convergence
Algorithm 1
EM with Features
Initialize probabilities \( \theta \)
repeat
Compute expected counts \( e \)
Fit parameters \( \theta \)
until convergence

Algorithm 2
EM with Features
Initialize weights \( w \)
repeat
Compute expected counts \( e \)
Fit parameters \( w \)
Transform \( w \)
until convergence

Algorithm 3
EM with Features
Initialize weights \( w \)
repeat
Compute expected counts \( e \)
repeat
Compute \( \nabla (w, e) \)
Compute \( \nabla L (w, e) \)
\( w \) ← climb \( (w, \nabla L (w, e), \nabla (w, e)) \)
until convergence
Transform \( w \)
until convergence

Algorithm 4
Direct Gradient with Features
Initialize weights \( w \)
repeat
Compute expected counts \( e \)
Compute \( L (w) \)
Compute \( \nabla L (w, e) \)
\( w \) ← climb \( (w, L (w), \nabla L (w, e)) \)
until convergence
Transform \( w \)
until convergence
EM without Features

Initialize probabilities $\theta$

repeat
  Compute expected counts $e$
  Fit parameters $\theta$
until convergence

Algorithm 1

EM with Features

Initialize weights $w$

repeat
  Compute expected counts $e$
  Fit parameters $w$
  Transform $w$ to $\theta$
until convergence

Algorithm 2

EM with Features

Initialize weights $w$

repeat
  Compute expected counts $e$
  Compute $\nabla \mathcal{L}(w, e)$
  $w \leftarrow \text{climb}(w, \mathcal{L}(w, e), \nabla \mathcal{L}(w, e))$
  Transform $w$ to $\theta$
until convergence

Algorithm 3

Direct Gradient with Features

Initialize weights $w$

repeat
  Compute expected counts $e$
  Compute $\mathcal{L}(w)$
  Compute $\nabla \mathcal{L}(w, e)$
  $w \leftarrow \text{climb}(w, \mathcal{L}(w, e), \nabla \mathcal{L}(w, e))$
  Transform $w$ to $\theta$
until convergence

Algorithm 4
EM without Features

Initialize probabilities $\theta$
repeat
Compute expected counts $e$
Fit parameters $\theta$
until convergence

Algorithm 1

Algorithm 2

Algorithm 3

Algorithm 4

Direct Gradient with Features

Initialize weights $w$
repeat
Compute expected counts $e$
Compute $L(w)$
Compute $\nabla L(w)$
w ← climb $(w, L(w), \nabla L(w))$
until convergence
Transform $w$ to $\theta$
until convergence

$\text{L}(w)$
**EM without Features**

Initialize probabilities $\theta$

repeat
- Compute expected counts $e$
- Fit parameters $\theta$
until convergence

**Algorithm 2**

Initialize weights $w$

repeat
- Compute expected counts $e$
- Fit parameters $w$
- Transform $w$ to $\theta$
until convergence

**Algorithm 3**

Initialize weights $w$

repeat
- Compute $\mathcal{L}(w, e)$
- Compute $\nabla \mathcal{L}(w, e)$
- $w \leftarrow \text{climb}(w, \mathcal{L}(w, e), \nabla \mathcal{L}(w, e))$
until convergence
- Transform $w$ to $\theta$
until convergence

**Algorithm 4**

Initialize weights $w$

repeat
- Compute expected counts $e$
- Compute $\mathcal{L}(w)$
- Compute $\nabla \mathcal{L}(w)$
- $w \leftarrow \text{climb}(w, \mathcal{L}(w), \nabla \mathcal{L}(w))$
until convergence
- Transform $w$ to $\theta$
until convergence
**Algorithm 1**

EM without Features

*Initialize probabilities* $\theta$

repeat

1. Compute expected counts $e$
2. Fit parameters $\theta$

until convergence

**Algorithm 2**

EM with Features

*Initialize weights* $w$

repeat

1. Compute expected counts $e$
2. Fit parameters $w$
3. Transform $w$ to $\theta$

until convergence

**Algorithm 3**

EM with Features

*Initialize weights* $w$

repeat

1. Compute expected counts $e$
2. repeat
   1. Compute $\mathbb{F}(w, e)$
   2. Compute $\nabla \mathbb{F}(w, e)$
   3. $w \leftarrow \text{climb}(w, \mathbb{F}(w, e), \nabla \mathbb{F}(w, e))$
3. Transform $w$ to $\theta$

until convergence

**Algorithm 4**

Direct Gradient with Features

*Initialize weights* $w$

repeat

1. Compute expected counts $e$
2. Compute $L(w)$
3. Compute $\nabla L(w)$
4. $w \leftarrow \text{climb}(w, L(w), \nabla L(w))$
5. Transform $w$ to $\theta$

until convergence
Algorithm 1
EM without Features

Initialize probabilities $\theta$
repeat
Compute expected counts $e$
Fit parameters $\theta$
until convergence

Algorithm 2
EM with Features

Initialize weights $w$
repeat
Compute expected counts $e$
Fit parameters $w$
Transform $w$ to $\theta$
until convergence

Algorithm 3
EM with Features

Initialize weights $w$
repeat
repeat
Compute $\mathbb{L}(w, e)$
Compute $\nabla \mathbb{L}(w, e)$
$w \leftarrow \text{climb}(w, \mathbb{L}(w, e), \nabla \mathbb{L}(w, e))$
until convergence
Transform $w$ to $\theta$
until convergence

Algorithm 4
Direct Gradient with Features

Initialize weights $w$
repeat
Compute expected counts $e$
Compute $\mathbb{L}(w)$
Compute $\nabla \mathbb{L}(w)$
$w \leftarrow \text{climb}(w, \mathbb{L}(w), \nabla \mathbb{L}(w))$
Compute $\nabla \mathbb{L}(w)$
Transform $w$ to $\theta$
until convergence
EM without Features

Initialize probabilities $\theta$

repeat
- Compute expected counts $e$
- Fit parameters $\theta$

until convergence
**EM with Features**

Initialize weights $w$

repeat

- Compute expected counts $e$
- Fit parameters $w$
- Transform $w$ to $\theta$

until convergence
EM with Features

Initialize weights $w$

repeat
  \- Compute expected counts $e$

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

until convergence

\- Transform $w$ to $\theta$

until convergence
Algorithm 1

1. Initialize weights $w$
2. repeat
   1. Compute expected counts $e$
   2. Fit parameters $w$
   3. Transform $w$ to $\theta$
   until convergence
3. until convergence

Algorithm 2

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

Algorithm 3

1. Initialize weights $w$
2. repeat
   1. Compute expected counts $e$
   2. Compute $L(w)$
   3. Compute $\nabla L(w)$
   4. $w \leftarrow \text{climb}(w, L(w), \nabla L(w))$
3. until convergence
4. Transform $w$ to $\theta$
5. until convergence
Initialize weights $w$

repeat
  Compute expected counts $e$
  repeat
    Compute $\ell(w, e)$
    Compute $\nabla \ell(w, e)$
    $w \leftarrow \text{climb}(w, \ell(w, e), \nabla \ell(w, e))$
  until convergence

Transform $w$ to $\theta$

until convergence

$\mathbf{L}(w)$
Algorithm 1

EM with Features

Initialize weights $w$

repeat

Compute expected counts $e$

Fit parameters $w$

Transform $w$ to $\theta$

until convergence

Algorithm 2

EM with Features

Initialize weights $w$

repeat

Compute expected counts $e$

repeat

Compute $\ell(w, e)$

Compute $\nabla \ell(w, e)$

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

until convergence

Transform $w$ to $\theta$

until convergence

Algorithm 3

Direct Gradient with Features

Initialize weights $w$

repeat

Compute expected counts $e$

Compute $L(w)$

Compute $\nabla L(w)$

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

Transform $w$ to $\theta$

until convergence
EM with Features

Initialize weights $w$
repeat
  Compute expected counts $e$
  repeat
    Compute $\ell(w, e)$
    Compute $\nabla \ell(w, e)$
    $w \leftarrow \text{climb}(w, \ell(w, e), \nabla \ell(w, e))$
  until convergence
  Transform $w$ to $\theta$
until convergence
Algorithm 1

Initialize weights \( w \)

repeat
  Compute expected counts \( e \)
  Fit parameters \( w \)
  Transform \( w \) to \( \theta \)
until convergence

Algorithm 2

Initialize weights \( w \)

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

Algorithm 3

Initialize weights \( w \)

repeat
  Compute expected counts \( e \)
  Compute \( L(w) \)
  Compute \( \nabla \ell(w, e) \)
  \( w \leftarrow \text{climb}(w, L(w), \nabla \ell(w, e)) \)
until convergence

Transform \( w \) to \( \theta \)
until convergence

\( L(w) \)
**Algorithm 1**

Initialize weights $w$

repeat

Compute expected counts $e$

Fit parameters $w$

Transform $w$ to $\theta$

until convergence

**Algorithm 2**

Initialize weights $w$

repeat

repeat

Compute $\ell(w, e)$

Compute $\nabla \ell(w, e)$

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

until convergence

Transform $w$ to $\theta$

until convergence

**Algorithm 3**

Initialize weights $w$

repeat

Compute expected counts $e$

Compute $L(w)$

Compute $\nabla L(w)$

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

Transform $w$ to $\theta$

until convergence
Initialize weights \( w \)

**repeat**

- Compute expected counts \( e \)

  **repeat**
  
  - Compute \( \ell(w, e) \)
  - Compute \( \nabla \ell(w, e) \)
  - \( w \leftarrow \text{climb}(w, \ell(w, e), \nabla \ell(w, e)) \)

  **until** convergence

- Transform \( w \) to \( \theta \)

  **until** convergence
Algorithm 1

\textbf{EM with Features}

Initialize weights $w$

\textbf{repeat}

Compute expected counts $e$

\textbf{repeat}

Fit parameters $w$

Transform $w$ to $\theta$

\textbf{until convergence}

Algorithm 2

\textbf{EM with Features}

Initialize weights $w$

\textbf{repeat}

Compute expected counts $e$

\textbf{repeat}

Compute $\ell(w, e)$

Compute $\nabla \ell(w, e)$

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

\textbf{until convergence}

Transform $w$ to $\theta$

\textbf{until convergence}

Algorithm 3

\textbf{Direct Gradient with Features}

Initialize weights $w$

\textbf{repeat}

Compute expected counts $e$

Compute $L(w)$

Compute $\nabla L(w)$

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

\textbf{until convergence}

Transform $w$ to $\theta$

\textbf{until convergence}
Algorithm 1
Initialize weights $w$
repeat
  Compute expected counts $e$
  Fit parameters $w$
  Transform $w$ to $\theta$
until convergence

Algorithm 2
Initialize weights $w$
repeat
  Compute expected counts $e$
  repeat
    Compute $\ell(w, e)$
    Compute $\nabla \ell(w, e)$
    $w \leftarrow \text{climb}(w, \ell(w, e), \nabla \ell(w, e))$
  until convergence
  Transform $w$ to $\theta$
until convergence

Algorithm 3
Initialize weights $w$
repeat
  Compute expected counts $e$
  Compute $L(w)$
  Compute $\nabla L(w)$
  $w \leftarrow \text{climb}(w, L(w), \nabla L(w))$
  Transform $w$ to $\theta$
until convergence
EM with Features

**Algorithm 1**

Initialize weights $w$

repeat

compute expected counts $e$

fit parameters $w$

transform $w$ to $\theta$

until convergence

**Algorithm 2**

Initialize weights $w$

repeat

repeat

compute $\ell(w, e)$

compute $\nabla \ell(w, e)$

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

until convergence

transform $w$ to $\theta$

until convergence

**Algorithm 3**

Initialize weights $w$

repeat

compute expected counts $e$

compute $L(w)$

compute $\nabla L(w)$

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

transform $w$ to $\theta$

until convergence
Algorithm 1
Initialize weights $w$
repeat
    Compute expected counts $e$
    Fit parameters $w$
    Transform $w$ to $\theta$
until convergence

Algorithm 2
Initialize weights $w$
repeat
    Compute expected counts $e$
    repeat
        Compute $\ell(w, e)$
        Compute $\nabla \ell(w, e)$
        $w \leftarrow \text{climb}(w, \ell(w, e), \nabla \ell(w, e))$
    until convergence
    Transform $w$ to $\theta$
until convergence

Algorithm 3
Initialize weights $w$
repeat
    Compute expected counts $e$
    Compute $L(w)$
    Compute $\nabla L(w, e)$
    $w \leftarrow \text{climb}(w, L(w), \nabla L(w, e))$
until convergence
Transform $w$ to $\theta$
until convergence

$L(w)$

Initialize weights $w$
repeat
    Compute expected counts $e$
    repeat
        Compute $\ell(w, e)$
        Compute $\nabla \ell(w, e)$
        $w \leftarrow \text{climb}(w, \ell(w, e), \nabla \ell(w, e))$
    until convergence
    Transform $w$ to $\theta$
until convergence
Initialize weights $w$
repeat
  Compute expected counts $e$
repeat
  Compute $\ell(w, e)$
  Compute $\nabla\ell(w, e)$
  $w \leftarrow \text{climb}(w, \ell(w, e), \nabla\ell(w, e))$
until convergence
Transform $w$ to $\theta$
until convergence
Algorithm 1

Initialize weights $w$

repeat
  Compute expected counts $e$
  Fit parameters $w$
  Transform $w$ to $\theta$
until convergence

Algorithm 2

Initialize weights $w$

repeat
  Compute $\ell(w, e)$
  Compute $\nabla \ell(w, e)$
  $w \leftarrow$ climb($w, \ell(w, e), \nabla \ell(w, e)$)
until convergence

Transform $w$ to $\theta$
until convergence

Algorithm 3

Initialize weights $w$

repeat
  Compute expected counts $e$
  Compute $L(w)$
  Compute $\nabla L(w, e)$
  $w \leftarrow$ climb($w, L(w), \nabla L(w, e)$)
until convergence

Transform $w$ to $\theta$
until convergence

$L(w)$

Graphical representation of EM with Features
**Algorithm 1**

Initialize weights $w$

repeat

Compute expected counts $e$

Fit parameters $w$

Transform $w$ to $\theta$

until convergence

**Algorithm 2**

Initialize weights $w$

repeat

repeat

Compute $\mathbb{E}(w, e)$

Compute $\nabla \mathbb{E}(w, e)$

$w \leftarrow$ climb($w$, $\mathbb{E}(w, e)$, $\nabla \mathbb{E}(w, e)$)

until convergence

Transform $w$ to $\theta$

until convergence

**Algorithm 3**

Initialize weights $w$

repeat

Compute expected counts $e$

Compute $L(w)$

Compute $\nabla L(w, e)$

$w \leftarrow$ climb($w$, $L(w, e)$, $\nabla L(w, e)$)

until convergence

Transform $w$ to $\theta$

until convergence
Direct Gradient with Features

**EM w/ Features**

1. Initialize weights $w$
2. \textbf{repeat}
   - Compute expected counts $e$
     \textbf{repeat}
     - Compute $\ell(w, e)$
     - Compute $\nabla \ell(w, e)$
     - $w \leftarrow \text{climb}(w, \ell(w, e), \nabla \ell(w, e))$
   \textbf{until} convergence
3. Transform $w$ to $\theta$
4. \textbf{until} convergence

**DG w/ Features**

1. Initialize weights $w$
2. \textbf{repeat}
   - Compute expected counts $e$
     - Compute $L(w)$
     - Compute $\nabla \ell(w, e)$
     - $w \leftarrow \text{climb}(w, L(w), \nabla \ell(w, e))$
   \textbf{until} convergence
3. Transform $w$ to $\theta$
4. \textbf{until} convergence
**Direct Gradient with Features**

**Algorithm 1**

Initialise weights $w$

repeat

1. Compute expected counts $e$
2. Fit parameters $w$
3. Transform $w$ to $\theta$

until convergence

**Algorithm 2**

Initialise weights $w$

repeat

1. Compute expected counts $e$
2. Compute $\mathcal{L}(w)$
3. Compute $\nabla \mathcal{L}(w, e)$
4. $w \leftarrow \text{climb}(w, \mathcal{L}(w), \nabla \mathcal{L}(w, e))$
5. Transform $w$ to $\theta$

until convergence

**Algorithm 3**

Initialise weights $w$

repeat

1. Compute expected counts $e$
2. Compute $\mathcal{L}(w)$
3. Compute $\nabla \mathcal{L}(w, e)$
4. $w \leftarrow \text{climb}(w, \mathcal{L}(w), \nabla \mathcal{L}(w, e))$
5. Transform $w$ to $\theta$

until convergence

**Diagram:**

- $L(w)$
- Red dot indicates the computed expected counts $e$
- Blue line indicates the computed $\mathcal{L}(w)$
- Thick blue line indicates the $\nabla \mathcal{L}(w, e)$
- Green line indicates the climb operation
- Purple dot indicates the transformation of $w$ to $\theta$
Direct Gradient with Features

Initialize weights $w$

repeat

Compute expected counts $e$

Fit parameters $w$

Transform $w$ to $\theta$

until convergence

Algorithm 1

EM with Features

Initialize weights $w$

repeat

Compute expected counts $e$

Fit parameters $w$

Transform $w$ to $\theta$

until convergence

Algorithm 2

Direct Gradient with Features

Initialize weights $w$

repeat

Compute expected counts $e$

Compute $\mathcal{L}(w)$

Compute $\nabla \mathcal{L}(w, e)$

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

Transform $w$ to $\theta$

until convergence

Algorithm 3
Initialize weights $w$

repeat
  Compute expected counts $e$
  Compute $L(w)$
  Compute $\nabla l(w, e)$
  $w \leftarrow \text{climb}(w, L(w), \nabla l(w, e))$
  Transform $w$ to $\theta$
until convergence
Direct Gradient with Features

Initialize weights $w$

repeat

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

until convergence
Direct Gradient with Features

Initialize weights \( w \)

repeat

- Compute expected counts \( e \)
- Fit parameters \( w \)
- Transform \( w \) to \( \theta \)

until convergence

Algorithm 1

EM with Features

Initialize weights \( w \)

repeat

- Compute expected counts \( e \)
- Fit parameters \( w \)
- Transform \( w \) to \( \theta \)

until convergence

Algorithm 2

EM with Features

Initialize weights \( w \)

repeat

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

until convergence

Algorithm 3

Direct Gradient with Features

Initialize weights \( w \)

repeat

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

until convergence
Direct Gradient with Features

Algorithm 1

Initialize weights $w$

repeat

Compute expected counts $e$

Fit parameters $w$

Transform $w$ to $\theta$

until convergence

Algorithm 2

EM with Features

Initialize weights $w$

repeat

Compute expected counts $e$

repeat

Compute $\nabla \ell(w, e)$

Compute $\nabla L(w)$

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

Transform $w$ to $\theta$

until convergence

Algorithm 3

Direct Gradient with Features

Initialize weights $w$

repeat

Compute expected counts $e$

Compute $L(w)$

Compute $\nabla L(w)$

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

Transform $w$ to $\theta$

until convergence
Direct Gradient with Features

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

$\text{Algorithm 2}$

$\text{Direct Gradient with Features}$

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

$\text{Algorithm 3}$

$\text{EM with Features}$

Initialize weights $w$
repeat
Compute expected counts $e$
Fit parameters $w$
Transform $w$ to $\theta$
until convergence
Direct Gradient with Features

Initialize weights $w$

repeat

- Compute expected counts $e$
- Fit parameters $w$

Transform $w$ to $\theta$ until convergence

Algorithm 1

EM with Features

Initialize weights $w$

repeat

- Compute expected counts $e$
- Fit parameters $w$

Transform $w$ to $\theta$ until convergence

Algorithm 2

Direct Gradient with Features

Initialize weights $w$

repeat

- Compute expected counts $e$
- Compute $L(w)$
- Compute $\nabla L(w, e)$

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

Transform $w$ to $\theta$ until convergence

Algorithm 3
Direct Gradient with Features

Initialize weights $w$

repeat
- Compute expected counts $e$
- Fit parameters $w$
- Transform $w$ to $\theta$
until convergence

Algorithm 1

Algorithm 2

Algorithm 3
Unsupervised Induction Tasks

POS Induction:

Grammar Induction:

Word Alignment:

Word Segmentation:
POS Induction Results

<table>
<thead>
<tr>
<th>DT</th>
<th>JJ</th>
<th>NN</th>
<th>VBZ</th>
<th>IN</th>
<th>NN</th>
</tr>
</thead>
<tbody>
<tr>
<td>The green cat sleeps at home.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
POS Induction Results

The green cat sleeps at home.

Key distribution: $P(John|NN)$
POS Induction Results

The green cat sleeps at home.

Key distribution: \( P(John|NN) \)

Features:

Basic: \( John \land NN \)
Contains-Digit: \( +Digit \land NN \)
Contains-Hyphen: \( +Hyph \land NN \)
Initial-Capital: \( +Cap \land NN \)
Suffix: \( +ing \land NN \)
POS Induction Results

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

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

HMM
EM

63.1
POS Induction Results

Features:
- Basic: John \(\land\) NNP
- Contains-Digit: +Digit \(\land\) NNP
- Contains-Hyphen: +Hyph \(\land\) NNP
- Initial-Capital: +Cap \(\land\) NNP
- Suffix: +ing \(\land\) NNP

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

Many-to-1 Accuracy

<table>
<thead>
<tr>
<th>Feature</th>
<th>HMM EM</th>
<th>HMM Features EM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>63.1</td>
<td>68.1</td>
</tr>
</tbody>
</table>

+5.0
POS Induction Results

Many-to-1 Accuracy

Features:
- Basic: John \( \wedge \) NNP
- Contains-Digit: +Digit \( \wedge \) NNP
- Contains-Hyphen: +Hyph \( \wedge \) NNP
- Initial-Capital: +Cap \( \wedge \) NNP
- Suffix: +ing \( \wedge \) NNP

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

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

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

1-to-1 Accuracy

HMM EM: 43.2
HMM Features EM: 48.3
HMM Features Gradient: 56.0

+5.1
+12.8
Grammar Induction Results

The green cat sleeps at home.
The green cat sleeps at home.

Key distributions:  \( P(JJ|NN) \)  \( P(\text{stop}|NN) \)
Grammar Induction Results

Key distributions: $P(JJ|NN), P(stop|NN)$

Features:

Basic: $JJ \land NN, JJ \land NNS$

Noun: $JJ \land Noun$

Verb: $JJ \land Verb$

Noun-verb: $JJ \land NounOrVerb$
Grammar Induction Results

English Directed Accuracy

The green cat sleeps at home.

Features:

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

Chinese Directed Accuracy

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 \land NN, JJ \land NNS
Noun: JJ \land Noun
Verb: JJ \land Verb
Noun-verb: JJ \land 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

The green cat sleeps at home.

47.8
42.5
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

- DMV: 47.8
- EM: 48.3
+0.5

Chinese Directed Accuracy

- DMV: 42.5
- EM: 49.9
+7.4
The green cat sleeps at home.

**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**

- **DMV**
  - 47.8
- **DMV Features**
  - 48.3
- **DMV Features Gradient**
  - +15.2

**Chinese Directed Accuracy**

- **DMV**
  - 42.5
- **DMV Features**
  - 49.9
- **DMV Features Gradient**
  - +11.1
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**

- DMV EM: 47.8
- DMV Features EM: 48.3
- DMV Features Gradient: 63.0
- Cohen and Smith '09 SLN DMV: 61.3

**Chinese Directed Accuracy**

- DMV EM: 42.5
- DMV Features EM: 49.9
- DMV Features Gradient: 53.6
- Cohen and Smith '09 SLN DMV: 51.9

The green cat sleeps at home.
El gato verde duerme en casa.

The green cat sleeps at home.
Word Alignment Results

Key distribution: \( P(gato|cat) \)
Word Alignment Results

Key distribution: \( P(gato|cat) \)

Features:

- **Basic:** \( gato \land cat \)
- **Edit-Distance:** \( \text{edit}(gato,cat) = 2 \)
- **Dictionary:** \( (gato,cat) \in \text{Dict} \)
- **Stem:** \( gato \land +\text{stem}(cat) \)
- **Prefix:** \( gato \land +ca \)
Word Alignment Results

Alignment Error Rate

Features:

Basic: \quad gato \land cat

Edit-Distance: \quad \text{edit}(gato, cat) = 2

Dictionary: \quad (gato, cat) \in \text{Dict}

Stem: \quad gato \land +\text{stem}(cat)

Prefix: \quad gato \land +ca

Data:

Train 10K sentences of FBIS
Chinese-English newswire

Test NIST 2002 Chinese-English dev set
Word Alignment Results

Features:

Basic: \( gato \land cat \)
Edit-Distance: \( \text{edit}(gato, cat) = 2 \)
Dictionary: \( (gato, cat) \in \text{Dict} \)
Stem: \( gato \land +\text{stem}(cat) \)
Prefix: \( gato \land +ca \)

Data:

Train 10K sentences of FBIS Chinese-English newswire
Test NIST 2002 Chinese-English dev set
Word Alignment Results

**Features:**

- **Basic:** \( gato \land cat \)
- **Edit-Distance:** \( \text{edit}(gato, cat) = 2 \)
- **Dictionary:** \( (gato, cat) \in \text{Dict} \)
- **Stem:** \( gato \land +\text{stem}(cat) \)
- **Prefix:** \( gato \land +ca \)

**Data:**

- **Train:** 10K sentences of FBIS Chinese-English newswire
- **Test:** NIST 2002 Chinese-English dev set

**Alignment Error Rate:**

- Model I: EM 38.0
- Model I Features: EM 35.6
- -2.4
Word Alignment Results

Features:
- Basic: \( gato \land cat \)
- Edit-Distance: \( \text{edit}(gato,\text{cat}) = 2 \)
- Dictionary: \( (gato,\text{cat}) \in \text{Dict} \)
- Stem: \( gato \land +\text{stem}(\text{cat}) \)
- Prefix: \( gato \land +\text{ca} \)

Data:
- Train 10K sentences of FBIS Chinese-English newswire
- Test NIST 2002 Chinese-English dev set

Alignment Error Rate:

<table>
<thead>
<tr>
<th>Model</th>
<th>Features</th>
<th>HMM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model I</td>
<td>EM</td>
<td>EM</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EM</td>
</tr>
<tr>
<td></td>
<td>38.0</td>
<td>35.6</td>
</tr>
<tr>
<td></td>
<td>-2.4</td>
<td>33.8</td>
</tr>
</tbody>
</table>

The green cat sleeps at home.
Word Alignment Results

Features:

Basic: \( gato \land cat \)

Edit-Distance: \( \text{edit}(gato, cat) = 2 \)

Dictionary: \( (gato, cat) \in \text{Dict} \)

Stem: \( gato \land +\text{stem}(cat) \)

Prefix: \( gato \land +ca \)

Data:

Train 10K sentences of FBIS Chinese-English newswire

Test NIST 2002 Chinese-English dev set

Alignment Error Rate

38.0 35.6 33.8 30.0

-2.4 -3.8

Model 1 EM Model 1 Features EM HMM EM HMM Features EM
Word Segmentation Results

[The][green][cat]
Word Segmentation Results

[The][green][cat]

Key distribution: \( P(\text{running}) \)
Word Segmentation Results

Features:

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

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

[The][green][cat]

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

[The][green][cat]

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

Unigram EM: 76.9
Unigram Features EM: 84.5

+7.6
Word Segmentation Results

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

Features:

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

Data:

Train and test on phonetic version of Bernstein-Ratner corpus

<table>
<thead>
<tr>
<th>Method</th>
<th>Unigram EM</th>
<th>Unigram Features EM</th>
<th>Unigram Features Gradient</th>
</tr>
</thead>
<tbody>
<tr>
<td>Token F1</td>
<td>76.9</td>
<td>84.5</td>
<td>88.0</td>
</tr>
<tr>
<td>+7.6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>+11.1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Word Segmentation Results

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

- [The][green][cat]
- Token F1
  - Unigram EM: 76.9
  - Unigram Features EM: 84.5 (+7.6)
  - Unigram Features Gradient: 88.0 (+11.1)
  - Johnson and Goldwater '09 Adaptor Grammar: 89
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!