Advanced Human Language Technologies

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

Outline

Outline

Examples - Named Entity Recognition (NER)

$\mathbf{y}$	PER	-	QNT	-	-	ORG	ORG	-	TIME
$\mathbf{x}$	Jim	bought	300	shares	of	Acme	Corp.	in	2006

Examples - Named Entity Recognition (NER)

$\mathbf{y}$	PER	-	QNT	-	-	ORG	ORG	-	TIME
$\mathbf{x}$	Jim	bought	300	shares	of	Acme	Corp.	in	2006

$\mathbf{y}$	PER	PER	-	-	LOC
$\mathbf{x}$	Jack	London	went	to	Paris

$\mathbf{y}$	PER	PER	-	-	LOC
$\mathbf{x}$	Paris	Hilton	went	to	London

Examples Part-of-Speech (PoS) Tagging

$\mathbf{y}$	DT	NN	VBZ	IN	DT	JJ	NN	.
$\mathbf{x}$	The	fox	jumps	over	the	lazy	dog	.

Examples Part-of-Speech (PoS) Tagging

y	DT	NN	VBZ	IN	DT	JJ	NN	.
x	The	fox	jumps	over	the	lazy	dog	.

y	DT	NN	NN	VBD	DT	JJ	NN	.
x	The	fox	jumps	scared	the	lazy	dog	.

Outline

Problem Formulation

- $\mathbf{x} = \mathbf{x}_1\mathbf{x}_2\ldots\mathbf{x}_n$ are input sequences, $\mathbf{x}_i \in \mathcal{X}$
- $\mathbf{y} = \mathbf{y}_1\mathbf{y}_2\ldots\mathbf{y}_n$ are output sequences, $\mathbf{y}_i \in \{1, \ldots, L\}$

- **Goal:** given training data

$$\left\{(\mathbf{x}^{(1)}, \mathbf{y}^{(1)}), (\mathbf{x}^{(2)}, \mathbf{y}^{(2)}), \ldots, (\mathbf{x}^{(m)}, \mathbf{y}^{(m)})\right\}$$

learn a predictor $\mathbf{x} \to \mathbf{y}$ that works well on unseen inputs $\mathbf{x}$

- What is the form of our prediction model?

Outline

Outline

Approach 1: Local Classifiers

?
Jack London went to Paris

Decompose the sequence into n classification problems:

- A classifier predicts individual labels at each position

$$\hat{y}_i = \underset{l \in \{\text{LOC}, \text{PER}, \text{-}\}}{\operatorname{argmax}} \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, i, l)$$

- $\mathbf{f}(\mathbf{x}, i, l)$ represents an assignment of label l for x_i
- $\mathbf{w}$ is a vector of parameters, has a weight for each feature of $\mathbf{f}$
  - Use standard classification methods to learn $\mathbf{w}$

Approach 1: Local Classifiers

$$\overset{?}{\text{London}}$$

Jack London went to Paris

Decompose the sequence into n classification problems:

- A classifier predicts individual labels at each position

$$\hat{y}_i = \underset{l \,\in\, \{\text{LOC},\ \text{PER},\ \text{-}\}}{\operatorname{argmax}} \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, i, l)$$

- $\mathbf{f}(\mathbf{x}, i, l)$ represents an assignment of label l for x_i
- $\mathbf{w}$ is a vector of parameters, has a weight for each feature of $\mathbf{f}$
  - Use standard classification methods to learn $\mathbf{w}$
- At test time, predict the best sequence by a simple concatenation of the best label for each position

Indicator Features

- $\mathbf{f}(\mathbf{x}, i, l)$ is a vector of d features representing label l for $\mathbf{x}_i$

$$\mathbf{f}(\mathbf{x}, i, l) = (\ \mathbf{f}_1(\mathbf{x}, i, l), \dots, \mathbf{f}_j(\mathbf{x}, i, l), \dots, \mathbf{f}_d(\mathbf{x}, i, l)\ )$$

- What's in a feature $\mathbf{f}_j(\mathbf{x}, i, l)$?
  - Anything we can compute using $\mathbf{x}$ and i and l
  - Anything that indicates whether l is (not) a good label for $\mathbf{x}_i$
  - Indicator features: binary-valued features looking at a single simple property

$$\mathbf{f}_j(\mathbf{x}, i, l) = \begin{cases} 1 & \text{if } \mathbf{x}_i = \text{London and } l = \text{LOC} \\ 0 & \text{otherwise} \end{cases}$$

$$\mathbf{f}_k(\mathbf{x}, i, l) = \begin{cases} 1 & \text{if } \mathbf{x}_{i+1} = \text{went and } l = \text{LOC} \\ 0 & \text{otherwise} \end{cases}$$

More Features for NE Recognition

	PER			
Jack	London	went	to	Paris

In practice, construct $\mathbf{f}(\mathbf{x}, i, l)$ by ...

- Define a number of simple patterns of $\mathbf{x}$ and i
  - current word $\mathbf{x}_i$
  - is $\mathbf{x}_i$ capitalized?
  - $\mathbf{x}_i$ has digits?
  - prefixes/suffixes of size 1, 2, 3, ...
  - is $\mathbf{x}_i$ a known location?
  - is $\mathbf{x}_i$ a known person?
  - next word
  - previous word
  - current and next words together
  - other combinations
- Generate features by combining patterns with label identities l

More Features for NE Recognition

PER	PER	-		
Jack	London	went	to	Paris

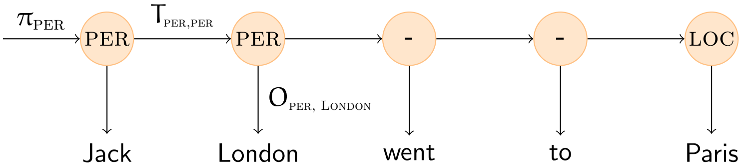
In practice, construct $\mathbf{f}(\mathbf{x}, \mathbf{i}, \mathbf{l})$ by ...

- Define a number of simple patterns of $\mathbf{x}$ and $\mathbf{i}$
  - current word $\mathbf{x_i}$
  - is $\mathbf{x_i}$ capitalized?
  - $\mathbf{x_i}$ has digits?
  - prefixes/suffixes of size 1, 2, 3, ...
  - is $\mathbf{x_i}$ a known location?
  - is $\mathbf{x_i}$ a known person?
  - next word
  - previous word
  - current and next words together
  - other combinations
- Generate features by combining patterns with label identities $\mathbf{l}$

Main limitation: features can't capture interactions between labels!

Outline

Approach 2: HMM for Sequence Prediction

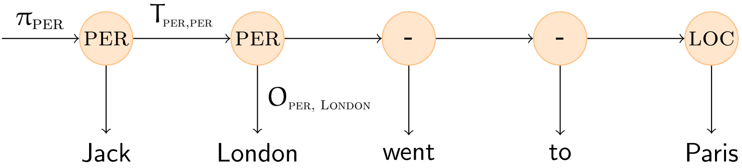


- Define an HMM were each label is a state
- Model parameters:
  - π_l : probability of starting with label l
  - $T_{l,l'}$: probability of transitioning from l to l'
  - $O_{l,x}$: probability of generating symbol x given label l
- Predictions:

$$p(\mathbf{x}, \mathbf{y}) = \pi_{\mathbf{y}_1} O_{\mathbf{y}_1,\mathbf{x}_1} \prod_{i>1} T_{\mathbf{y}_{i-1},\mathbf{y}_i} O_{\mathbf{y}_i,\mathbf{x}_i}$$

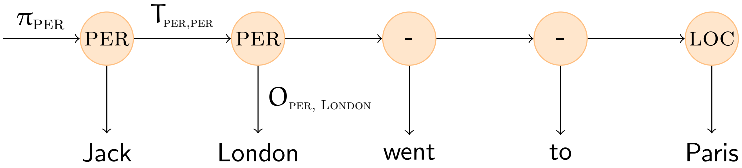
- Learning: relative counts + smoothing
- Prediction: Viterbi algorithm

Approach 2: Representation in HMM



- Label interactions are captured in the transition parameters
- But interactions between symbols and labels are quite limited!
  - Only $O_{\mathbf{y}_i, \mathbf{x}_i} = p(\mathbf{x}_i \mid \mathbf{y}_i)$
  - Not clear how to exploit patterns such as:
    - Capitalization, digits
    - Prefixes and suffixes
    - Next word, previous word
    - Combinations of these with label transitions

Approach 2: Representation in HMM



- Label interactions are captured in the transition parameters
- But interactions between symbols and labels are quite limited!
  - Only $\mathrm{O}_{\mathbf{y}_i,\mathbf{x}_i} = p(\mathbf{x}_i \mid \mathbf{y}_i)$
  - Not clear how to exploit patterns such as:
    - Capitalization, digits
    - Prefixes and suffixes
    - Next word, previous word
    - Combinations of these with label transitions
- Why? HMM independence assumptions:

given label $\mathbf{y}_i$, token $\mathbf{x}_i$ is independent of anything else

Local Classifiers vs. HMM

Local Classifiers

- Form:

$$\mathbf{w} \cdot \mathbf{f}(\mathbf{x}, i, l)$$

- Learning: standard classifiers
- Prediction: independent for each $\mathbf{x}_i$
- Advantage: feature-rich
- Drawback: no label interactions

HMM

- Form:

$$\pi_{\mathbf{y}_1} O_{\mathbf{y}_1, \mathbf{x}_1} \prod_{i>1} T_{\mathbf{y}_{i-1}, \mathbf{y}_i} O_{\mathbf{y}_i, \mathbf{x}_i}$$

- Learning: relative counts
- Prediction: Viterbi
- Advantage: label interactions
- Drawback: no fine-grained features

Outline

Approach 3: Global Sequence Predictors

$\mathbf{y}$:	PER	PER	-	-	LOC
$\mathbf{x}$:	Jack	London	went	to	Paris

Learn a single classifier from $\mathbf{x} \rightarrow \mathbf{y}$

$$\text{predict}(\mathbf{x}_{1:n}) = \underset{\mathbf{y} \in \mathcal{Y}^n}{\text{argmax}}\, \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, \mathbf{y})$$

Approach 3: Global Sequence Predictors

$\mathbf{y}$:	PER	PER	-	-	LOC
$\mathbf{x}$:	Jack	London	went	to	Paris

Learn a single classifier from $\mathbf{x} \rightarrow \mathbf{y}$

$$\text{predict}(\mathbf{x}_{1:n}) = \operatorname*{argmax}_{\mathbf{y} \in \mathcal{Y}^n} \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, \mathbf{y})$$

But ...

- How do we represent entire sequences in $\mathbf{f}(\mathbf{x}, \mathbf{y})$?
- There are exponentially-many sequences $\mathbf{y}$ for a given $\mathbf{x}$, how do we solve the argmax problem?

Factored Representations

$\mathbf{y}$:	PER	PER	-	-	LOC
$\mathbf{x}$:	Jack	London	went	to	Paris

- How do we represent entire sequences in $\mathbf{f}(\mathbf{x}, \mathbf{y})$?

Factored Representations

$\mathbf{y}$:	PER	PER	-	-	LOC
$\mathbf{x}$:	Jack	London	went	to	Paris

- How do we represent entire sequences in $\mathbf{f}(\mathbf{x}, \mathbf{y})$?
  - Look at the full label sequence $\mathbf{y}$ (intractable)

Factored Representations

$\mathbf{y}$:	PER	PER	-	-	LOC
$\mathbf{x}$:	Jack	London	went	to	Paris

- How do we represent entire sequences in $\mathbf{f}(\mathbf{x}, \mathbf{y})$?
  - Look at the full label sequence $\mathbf{y}$ (intractable)
  - Look at n-grams of output labels $\langle \mathbf{y}_{i-n+1}, \ldots, \mathbf{y}_{i-1}, \mathbf{y}_i \rangle$ (too expensive)

Factored Representations

$\mathbf{y}$:	PER	PER	-	-	LOC
$\mathbf{x}$:	Jack	London	went	to	Paris

- How do we represent entire sequences in $\mathbf{f}(\mathbf{x}, \mathbf{y})$?
  - Look at the full label sequence $\mathbf{y}$ (intractable)
  - Look at n-grams of output labels $\langle \mathbf{y}_{i-n+1}, \ldots, \mathbf{y}_{i-1}, \mathbf{y}_i \rangle$ (too expensive)
  - Look at trigrams of output labels $\langle \mathbf{y}_{i-2}, \mathbf{y}_{i-1}, \mathbf{y}_i \rangle$ (possible for small $|\mathcal{Y}|$)

Factored Representations

$\mathbf{y}$:	PER	PER	-	-	LOC
$\mathbf{x}$:	Jack	London	went	to	Paris

- How do we represent entire sequences in $\mathbf{f}(\mathbf{x}, \mathbf{y})$?
  - Look at the full label sequence $\mathbf{y}$ (intractable)
  - Look at n-grams of output labels $\langle \mathbf{y}_{i-n+1}, \ldots, \mathbf{y}_{i-1}, \mathbf{y}_i \rangle$ (too expensive)
  - Look at trigrams of output labels $\langle \mathbf{y}_{i-2}, \mathbf{y}_{i-1}, \mathbf{y}_i \rangle$ (possible for small $|\mathcal{Y}|$)
  - Look at bigrams of output labels $\langle \mathbf{y}_{i-1}, \mathbf{y}_i \rangle$ (definitely tractable)

Factored Representations

$\mathbf{y}$:	PER	PER	-	-	LOC
$\mathbf{x}$:	Jack	London	went	to	Paris

- How do we represent entire sequences in $\mathbf{f}(\mathbf{x}, \mathbf{y})$?
  - Look at the full label sequence $\mathbf{y}$ (intractable)
  - Look at n-grams of output labels $\langle \mathbf{y}_{i-n+1}, \ldots, \mathbf{y}_{i-1}, \mathbf{y}_i \rangle$ (too expensive)
  - Look at trigrams of output labels $\langle \mathbf{y}_{i-2}, \mathbf{y}_{i-1}, \mathbf{y}_i \rangle$ (possible for small $|\mathcal{Y}|$)
  - Look at bigrams of output labels $\langle \mathbf{y}_{i-1}, \mathbf{y}_i \rangle$ (definitely tractable)
  - Look at individual assignments $\mathbf{y}_i$ (standard classification)

Factored Representations

$\mathbf{y}$:	PER	PER	-	-	LOC
$\mathbf{x}$:	Jack	London	went	to	Paris

- How do we represent entire sequences in $\mathbf{f}(\mathbf{x}, \mathbf{y})$?
  - Look at the full label sequence $\mathbf{y}$ (intractable)
  - Look at n-grams of output labels $\langle \mathbf{y}_{i-n+1}, \ldots, \mathbf{y}_{i-1}, \mathbf{y}_i \rangle$ (too expensive)
  - Look at trigrams of output labels $\langle \mathbf{y}_{i-2}, \mathbf{y}_{i-1}, \mathbf{y}_i \rangle$ (possible for small $|\mathcal{Y}|$)
  - Look at bigrams of output labels $\langle \mathbf{y}_{i-1}, \mathbf{y}_i \rangle$ (definitely tractable)
  - Look at individual assignments $\mathbf{y}_i$ (standard classification)
- A factored representation will lead to a tractable model

Bigram Indicator Features

	1	2	3	4	5
$\mathbf{y}$	PER	PER	-	-	LOC
$\mathbf{x}$	Jack	London	went	to	Paris

- Indicator features:

$$\mathbf{f}_j(\mathbf{x}, i, \mathbf{y}_{i-1}, \mathbf{y}_i) = \begin{cases} 1 & \text{if } \mathbf{x}_i = \text{"London" and} \\ & \quad \mathbf{y}_{i-1} = \mathrm{PER} \text{ and } \mathbf{y}_i = \mathrm{PER} \\ 0 & \text{otherwise} \end{cases}$$

e.g., $\mathbf{f}_j(\mathbf{x}, 2, \mathrm{PER}, \mathrm{PER}) = 1, \quad \mathbf{f}_j(\mathbf{x}, 3, \mathrm{PER}, \text{-}) = 0$

More Bigram Indicator Features

	1	2	3	4	5
$\mathbf{x}$	Jack	London	went	to	Paris
$\mathbf{y}$	PER	PER	-	-	LOC
$\mathbf{y}'$	PER	LOC	-	-	LOC
$\mathbf{y}''$	-	-	-	LOC	-
$\mathbf{x}'$	My	trip	to	London	...

$\mathbf{f}_1(\ldots) = 1$ iff $\mathbf{x}_i =$"London" and $\mathbf{y}_{i-1} = \textsc{per}$ and $\mathbf{y}_i = \textsc{per}$

$\mathbf{f}_2(\ldots) = 1$ iff $\mathbf{x}_i =$"London" and $\mathbf{y}_{i-1} = \textsc{per}$ and $\mathbf{y}_i = \textsc{loc}$

$\mathbf{f}_3(\ldots) = 1$ iff $\mathbf{x}_{i-1} \sim$/(in|to|at)/ and $\mathbf{x}_i \sim$/[A-Z]/ and $\mathbf{y}_i = \textsc{loc}$

$\mathbf{f}_4(\ldots) = 1$ iff $\mathbf{y}_i = \textsc{loc}$ and $\textsc{world-cities}(\mathbf{x}_i) = 1$

$\mathbf{f}_5(\ldots) = 1$ iff $\mathbf{y}_i = \textsc{per}$ and $\textsc{first-names}(\mathbf{x}_i) = 1$

More Bigram Indicator Features

	1	2	3	4	5
$\mathbf{x}$	Jack	London	went	to	Paris
$\mathbf{y}$	PER	PER	-	-	LOC
$\mathbf{y}'$	PER	LOC	-	-	LOC
$\mathbf{y}''$	-	-	-	LOC	-
$\mathbf{x}'$	My	trip	to	London	...

$\mathbf{f}_1(\ldots) = 1$ iff $\mathbf{x}_i =$"London" and $\mathbf{y}_{i-1} = \text{PER}$ and $\mathbf{y}_i = \text{PER}$

$\mathbf{f}_2(\ldots) = 1$ iff $\mathbf{x}_i =$"London" and $\mathbf{y}_{i-1} = \text{PER}$ and $\mathbf{y}_i = \text{LOC}$

$\mathbf{f}_3(\ldots) = 1$ iff $\mathbf{x}_{i-1} \sim$/(in|to|at)/ and $\mathbf{x}_i \sim$/[A-Z]/ and $\mathbf{y}_i = \text{LOC}$

$\mathbf{f}_4(\ldots) = 1$ iff $\mathbf{y}_i = \text{LOC}$ and $\text{WORLD-CITIES}(\mathbf{x}_i) = 1$

$\mathbf{f}_5(\ldots) = 1$ iff $\mathbf{y}_i = \text{PER}$ and $\text{FIRST-NAMES}(\mathbf{x}_i) = 1$

More Bigram Indicator Features

	1	2	3	4	5
$\mathbf{x}$	Jack	London	went	to	Paris
$\mathbf{y}$	PER	PER	-	-	LOC
$\mathbf{y}'$	PER	LOC	-	-	LOC
$\mathbf{y}''$	-	-	-	LOC	-
$\mathbf{x}'$	My	trip	to	London	...

$\mathbf{f}_1(\ldots) = 1$ iff $\mathbf{x}_i =$"London" and $\mathbf{y}_{i-1} = \text{PER}$ and $\mathbf{y}_i = \text{PER}$

$\mathbf{f}_2(\ldots) = 1$ iff $\mathbf{x}_i =$"London" and $\mathbf{y}_{i-1} = \text{PER}$ and $\mathbf{y}_i = \text{LOC}$

$\mathbf{f}_3(\ldots) = 1$ iff $\mathbf{x}_{i-1} \sim$/(in|to|at)/ and $\mathbf{x}_i \sim$/[A-Z]/ and $\mathbf{y}_i = \text{LOC}$

$\mathbf{f}_4(\ldots) = 1$ iff $\mathbf{y}_i = \text{LOC}$ and $\text{WORLD-CITIES}(\mathbf{x}_i) = 1$

$\mathbf{f}_5(\ldots) = 1$ iff $\mathbf{y}_i = \text{PER}$ and $\text{FIRST-NAMES}(\mathbf{x}_i) = 1$

More Bigram Indicator Features

	1	2	3	4	5
$\mathbf{x}$	Jack	London	went	to	Paris
$\mathbf{y}$	PER	PER	-	-	LOC
$\mathbf{y}'$	PER	LOC	-	-	LOC
$\mathbf{y}''$	-	-	-	LOC	-
$\mathbf{x}'$	My	trip	to	London	...

$f_1(\ldots) = 1$ iff $\mathbf{x}_i =$"London" and $\mathbf{y}_{i-1} = \text{PER}$ and $\mathbf{y}_i = \text{PER}$

$f_2(\ldots) = 1$ iff $\mathbf{x}_i =$"London" and $\mathbf{y}_{i-1} = \text{PER}$ and $\mathbf{y}_i = \text{LOC}$

$f_3(\ldots) = 1$ iff $\mathbf{x}_{i-1} \sim$/(in|to|at)/ and $\mathbf{x}_i \sim$/[A-Z]/ and $\mathbf{y}_i = \text{LOC}$

$f_4(\ldots) = 1$ iff $\mathbf{y}_i = \text{LOC}$ and $\text{WORLD-CITIES}(\mathbf{x}_i) = 1$

$f_5(\ldots) = 1$ iff $\mathbf{y}_i = \text{PER}$ and $\text{FIRST-NAMES}(\mathbf{x}_i) = 1$

More Bigram Indicator Features

	1	2	3	4	5
$\mathbf{x}$	Jack	London	went	to	Paris
$\mathbf{y}$	PER	PER	-	-	LOC
$\mathbf{y}'$	PER	LOC	-	-	LOC
$\mathbf{y}''$	-	-	-	LOC	-
$\mathbf{x}'$	My	trip	to	London	...

$\mathbf{f}_1(\ldots) = 1$ iff $\mathbf{x}_i =$"London" and $\mathbf{y}_{i-1} = \text{PER}$ and $\mathbf{y}_i = \text{PER}$

$\mathbf{f}_2(\ldots) = 1$ iff $\mathbf{x}_i =$"London" and $\mathbf{y}_{i-1} = \text{PER}$ and $\mathbf{y}_i = \text{LOC}$

$\mathbf{f}_3(\ldots) = 1$ iff $\mathbf{x}_{i-1} \sim$/(in|to|at)/ and $\mathbf{x}_i \sim$/[A-Z]/ and $\mathbf{y}_i = \text{LOC}$

$\mathbf{f}_4(\ldots) = 1$ iff $\mathbf{y}_i = \text{LOC}$ and $\text{WORLD-CITIES}(\mathbf{x}_i) = 1$

$\mathbf{f}_5(\ldots) = 1$ iff $\mathbf{y}_i = \text{PER}$ and $\text{FIRST-NAMES}(\mathbf{x}_i) = 1$

Bigram-Factored Representations

$\mathbf{y}$:	PER	PER	-	-	LOC
$\mathbf{x}$:	Jack	London	went	to	Paris

- $\mathbf{f}(\mathbf{x}, i, \mathbf{y}_{i-1}, \mathbf{y}_i) = (\mathbf{f}_1(\mathbf{x}, i, \mathbf{y}_{i-1}, \mathbf{y}_i), \ldots, \mathbf{f}_d(\mathbf{x}, i, \mathbf{y}_{i-1}, \mathbf{y}_i))$
    - A d-dimensional feature vector of a label bigram at i
    - Each dimension is typically a boolean indicator (0 or 1)
- $\mathbf{f}(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n} \mathbf{f}(\mathbf{x}, i, \mathbf{y}_{i-1}, \mathbf{y}_i)$
    - A d-dimensional feature vector of the entire $\mathbf{y}$
    - Aggregated representation by summing bigram feature vectors
    - Each dimension is now a count of a feature pattern

Linear Sequence Prediction

$$\text{best}(\mathbf{x}_{1:n}) = \operatorname*{argmax}_{\mathbf{y} \in \mathcal{Y}^n} \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, \mathbf{y}) = \operatorname*{argmax}_{\mathbf{y} \in \mathcal{Y}^n} \mathbf{w} \cdot \sum_{i=1}^{n} \mathbf{f}(\mathbf{x}, i, \mathbf{y}_{i-1}, \mathbf{y}_i)$$

Linear Sequence Prediction

$$\text{best}(\mathbf{x}_{1:n}) = \operatorname*{argmax}_{\mathbf{y}\in\mathcal{Y}^n} \mathbf{w}\cdot\mathbf{f}(\mathbf{x},\mathbf{y}) = \operatorname*{argmax}_{\mathbf{y}\in\mathcal{Y}^n} \mathbf{w}\cdot\sum_{i=1}^{n}\mathbf{f}(\mathbf{x},i,\mathbf{y}_{i-1},\mathbf{y}_i)$$

- Note the linearity of the expression:

$$\begin{aligned}
\mathbf{w}\cdot\mathbf{f}(\mathbf{x},\mathbf{y}) &= \mathbf{w}\cdot\sum_{i=1}^{n}\mathbf{f}(\mathbf{x},i,\mathbf{y}_{i-1},\mathbf{y}_i) = \sum_{i=1}^{n}\mathbf{w}\cdot\mathbf{f}(\mathbf{x},i,\mathbf{y}_{i-1},\mathbf{y}_i)\\
&= \sum_{i=1}^{n}\sum_{j=1}^{d}\mathbf{w}_j\mathbf{f}_j(\mathbf{x},i,\mathbf{y}_{i-1},\mathbf{y}_i)
\end{aligned}$$

Linear Sequence Prediction

$$\text{best}(\mathbf{x}_{1:n}) = \underset{\mathbf{y} \in \mathcal{Y}^n}{\text{argmax}}\, \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, \mathbf{y}) = \underset{\mathbf{y} \in \mathcal{Y}^n}{\text{argmax}}\, \mathbf{w} \cdot \sum_{i=1}^{n} \mathbf{f}(\mathbf{x}, i, \mathbf{y}_{i-1}, \mathbf{y}_i)$$

- Note the linearity of the expression:

$$\begin{aligned}\mathbf{w} \cdot \mathbf{f}(\mathbf{x}, \mathbf{y}) &= \mathbf{w} \cdot \sum_{i=1}^{n} \mathbf{f}(\mathbf{x}, i, \mathbf{y}_{i-1}, \mathbf{y}_i) = \sum_{i=1}^{n} \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, i, \mathbf{y}_{i-1}, \mathbf{y}_i) \\ &= \sum_{i=1}^{n} \sum_{j=1}^{d} \mathbf{w}_j \mathbf{f}_j(\mathbf{x}, i, \mathbf{y}_{i-1}, \mathbf{y}_i)\end{aligned}$$

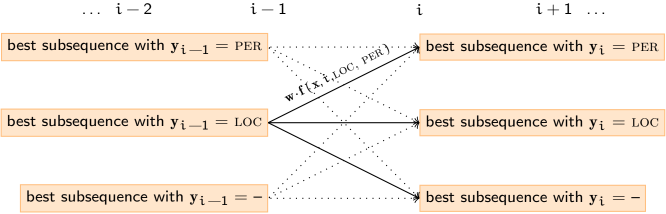
- Next questions:
  - How do we solve the argmax problem?
  - How do we learn $\mathbf{w}$?

Predicting with Factored Sequence Models

- Consider a fixed $\mathbf{w}$. Given $\mathbf{x}_{1:n}$ find:

$$\underset{\mathbf{y} \in \mathcal{Y}^n}{\text{argmax}} \sum_{i=1}^{n} \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, i, \mathbf{y}_{i-1}, \mathbf{y}_i)$$

- We can use the Viterbi algorithm, takes $O(n|\mathcal{Y}|^2)$
- Intuition: output sequences that share bigrams will share scores



Viterbi for Linear Factored Predictors

$$\hat{\mathbf{y}} = \underset{\mathbf{y} \in \mathcal{Y}^n}{\operatorname{argmax}} \sum_{i=1}^{n} \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, i, \mathbf{y}_{i-1}, \mathbf{y}_i)$$

- **Definition:** score of optimal sequence for $\mathbf{x}_{1:i}$ ending with $a \in \mathcal{Y}$

$$\delta_i(a) = \max_{\mathbf{y} \in \mathcal{Y}^i : \mathbf{y}_i = a} \sum_{j=1}^{i} \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, j, \mathbf{y}_{j-1}, \mathbf{y}_j)$$

- Use the following recursions, for all $a \in \mathcal{Y}$:

$$\begin{aligned} \delta_1(a) &= \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, 1, \mathbf{y}_0 = \text{NULL}, a) \\ \delta_i(a) &= \max_{b \in \mathcal{Y}} \delta_{i-1}(b) + \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, i, b, a) \end{aligned}$$

- The optimal score for $\mathbf{x}$ is $\max_{a \in \mathcal{Y}} \delta_n(a)$
- The optimal sequence $\hat{\mathbf{y}}$ can be recovered through *pointers*

Linear Factored Sequence Prediction

$$\text{predict}(\mathbf{x}_{1:n}) = \underset{\mathbf{y} \in \mathcal{Y}^n}{\text{argmax}}\, \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, \mathbf{y})$$

- Factored representation, e.g. based on bigrams
- Flexible, arbitrary features of full $\mathbf{x}$ and the factors
- Efficient prediction using Viterbi
- Next topic: learning $\mathbf{w}$:
  - Maximum-Entropy Markov Models (local)
  - Conditional Random Fields (global)

Outline

Sequence Tagging with Log-Linear Models

- $\mathbf{x}$ are input sequences (e.g. sentences of words)
- $\mathbf{y}$ are output sequences (e.g. sequences of NE tags)
- **Goal:** given training data
  $$\left\{(\mathbf{x}^{(1)}, \mathbf{y}^{(1)}), (\mathbf{x}^{(2)}, \mathbf{y}^{(2)}), \ldots, (\mathbf{x}^{(m)}, \mathbf{y}^{(m)})\right\}$$
  learn a model $\mathbf{x} \rightarrow \mathbf{y}$
- Log-linear models:
  $$\operatorname*{argmax}_{\mathbf{y} \in \mathcal{Y}^n} \mathrm{P}(\mathbf{y}|\mathbf{x}; \mathbf{w}) = \operatorname*{argmax}_{\mathbf{y} \in \mathcal{Y}^n} \frac{\exp(\mathbf{w} \cdot \mathbf{f}(\mathbf{x}, \mathbf{y}))}{Z(\mathbf{x}; \mathbf{w})}$$

Sequence Tagging with Log-Linear Models

- $\mathbf{x}$ are input sequences (e.g. sentences of words)
- $\mathbf{y}$ are output sequences (e.g. sequences of NE tags)
- **Goal:** given training data
$$\left\{(\mathbf{x}^{(1)}, \mathbf{y}^{(1)}), (\mathbf{x}^{(2)}, \mathbf{y}^{(2)}), \ldots, (\mathbf{x}^{(m)}, \mathbf{y}^{(m)})\right\}$$
learn a model $\mathbf{x} \rightarrow \mathbf{y}$
- Log-linear models:
$$\underset{\mathbf{y} \in \mathcal{Y}^n}{\operatorname{argmax}} \, \mathrm{P}(\mathbf{y} | \mathbf{x}; \mathbf{w}) = \underset{\mathbf{y} \in \mathcal{Y}^n}{\operatorname{argmax}} \frac{\exp(\mathbf{w} \cdot \mathbf{f}(\mathbf{x}, \mathbf{y}))}{Z(\mathbf{x}; \mathbf{w})}$$
- Exponentially many $\mathbf{y}$'s for a given input $\mathbf{x}$

Sequence Tagging with Log-Linear Models

- $\mathbf{x}$ are input sequences (e.g. sentences of words)
- $\mathbf{y}$ are output sequences (e.g. sequences of NE tags)
- **Goal:** given training data
$$\{(\mathbf{x}^{(1)}, \mathbf{y}^{(1)}), (\mathbf{x}^{(2)}, \mathbf{y}^{(2)}), \ldots, (\mathbf{x}^{(m)}, \mathbf{y}^{(m)})\}$$
learn a model $\mathbf{x} \to \mathbf{y}$

- Log-linear models:
$$\operatorname*{argmax}_{\mathbf{y} \in \mathcal{Y}^n} \mathrm{P}(\mathbf{y}|\mathbf{x}; \mathbf{w}) = \operatorname*{argmax}_{\mathbf{y} \in \mathcal{Y}^n} \frac{\exp(\mathbf{w} \cdot \mathbf{f}(\mathbf{x}, \mathbf{y}))}{Z(\mathbf{x}; \mathbf{w})}$$

- Exponentially many $\mathbf{y}$'s for a given input $\mathbf{x}$
- Solution 1: decompose $\mathrm{P}(\mathbf{y} \mid \mathbf{x})$ (MEMMs)
- Solution 2: decompose $\mathbf{f}(\mathbf{x}, \mathbf{y})$ (CRFs)

Outline

Maximum Entropy Markov Models (MEMMs)

(McCallum, Freitag, Pereira '00)

- Notation: $\mathbf{x}_{1:n} = \mathbf{x}_1 \ldots \mathbf{x}_n$
- Similarly to HMMs:

$$\begin{aligned}
P(\mathbf{y}_{1:n} \mid \mathbf{x}_{1:n}) &= P(\mathbf{y}_1 \mid \mathbf{x}_{1:n}) \times P(\mathbf{y}_{2:n} \mid \mathbf{x}_{1:n}, \mathbf{y}_1) \\
&= P(\mathbf{y}_1 \mid \mathbf{x}_{1:n}) \times \prod_{i=2}^{n} P(\mathbf{y}_i | \mathbf{x}_{1:n}, \mathbf{y}_{1:i-1}) \\
&= P(\mathbf{y}_1 | \mathbf{x}_{1:n}) \times \prod_{i=2}^{n} P(\mathbf{y}_i | \mathbf{x}_{1:n}, \mathbf{y}_{i-1})
\end{aligned}$$

- Assumption under MEMMs:

$$P(\mathbf{y}_i | \mathbf{x}_{1:n}, \mathbf{y}_{1:i-1}) = P(\mathbf{y}_i | \mathbf{x}_{1:n}, \mathbf{y}_{i-1})$$

Sequence Tagging: MEMMs

- Decompose tagging problem:

$$P(\mathbf{y}_{1:n} \mid \mathbf{x}_{1:n}) \quad = \quad P(\mathbf{y}_1 | \mathbf{x}_{1:n}) \times \prod_{i=2}^{n} P(\mathbf{y}_i | \mathbf{x}_{1:n}, i, \mathbf{y}_{i-1})$$

- Learn *local* log-linear distributions (i.e. MaxEnt)

$$p(y \mid \mathbf{x}, i, y') = \frac{\exp\{\mathbf{w} \cdot \mathbf{f}(\mathbf{x}, i, y', y)\}}{Z(\mathbf{x}, i, y')}$$

where

- $\mathbf{x}$ is an input sequence
- y and y' are tags
- $\mathbf{f}(\mathbf{x}, i, y', y)$ is a feature vector of $\mathbf{x}$, the position to be tagged, the previous tag and the current tag

Decoding with MEMMs

- Given $\mathbf{w}$, given $\mathbf{x}$, find:

$$
\begin{aligned}
\operatorname*{argmax}_{\mathbf{y}\in\mathcal{Y}^*} P(\mathbf{y} \mid \mathbf{x}; \mathbf{w}) &= \operatorname*{argmax}_{\mathbf{y}} \prod_{i=1}^{n} P(\mathbf{y}_i \mid \mathbf{x}, \mathbf{y}_{i-1}) \\
&= \operatorname*{argmax}_{\mathbf{y}} \frac{\prod_{i=1}^{n} \exp(\mathbf{w} \cdot \mathbf{f}(\mathbf{x}, i, \mathbf{y}_{i-1}, \mathbf{y}_i))}{\prod_{i=1}^{n} Z(\mathbf{x}, i; \mathbf{w})} \\
&= \operatorname*{argmax}_{\mathbf{y}} \prod_{i=1}^{n} \exp(\mathbf{w} \cdot \mathbf{f}(\mathbf{x}, i, \mathbf{y}_{i-1}, \mathbf{y}_i)) \\
&= \operatorname*{argmax}_{\mathbf{y}} \sum_{i=1}^{n} \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, i, \mathbf{y}_{i-1}, \mathbf{y}_i)
\end{aligned}
$$

- We can use the Viterbi algorithm

Outline

Conditional Random Fields
(Lafferty, McCallum, Pereira 2001)

- Log-linear model of the conditional distribution:

$$\mathrm{P}(\mathbf{y}|\mathbf{x};\mathbf{w}) = \frac{\exp\{\mathbf{w}\cdot\mathbf{f}(\mathbf{x},\mathbf{y})\}}{Z(\mathbf{x})}$$

  where
  - $\mathbf{x} = \mathbf{x}_1\mathbf{x}_2\ldots\mathbf{x}_n \in \mathcal{X}^*$
  - $\mathbf{y} = \mathbf{y}_1\mathbf{y}_2\ldots\mathbf{y}_n \in \mathcal{Y}^*$ and $\mathcal{Y} = \{1,\ldots,L\}$
  - $\mathbf{f}(\mathbf{x},\mathbf{y})$ is a feature vector of $\mathbf{x}$ and $\mathbf{y}$
  - $\mathbf{w}$ are model parameters

- To predict the best sequence

$$\hat{\mathbf{y}} = \operatorname*{argmax}_{\mathbf{y}\in\mathcal{Y}^*} \mathrm{P}(\mathbf{y}|\mathbf{x})$$

Conditional Random Fields
(Lafferty, McCallum, Pereira 2001)

- Log-linear model of the conditional distribution:

$$\mathrm{P}(\mathbf{y}|\mathbf{x};\mathbf{w}) = \frac{\exp\{\mathbf{w}\cdot\mathbf{f}(\mathbf{x},\mathbf{y})\}}{Z(\mathbf{x})}$$

where
  - $\mathbf{x} = \mathbf{x}_1\mathbf{x}_2\ldots\mathbf{x}_n \in \mathcal{X}^*$
  - $\mathbf{y} = \mathbf{y}_1\mathbf{y}_2\ldots\mathbf{y}_n \in \mathcal{Y}^*$ and $\mathcal{Y} = \{1,\ldots,L\}$
  - $\mathbf{f}(\mathbf{x},\mathbf{y})$ is a feature vector of $\mathbf{x}$ and $\mathbf{y}$
  - $\mathbf{w}$ are model parameters

- To predict the best sequence

$$\hat{\mathbf{y}} = \operatorname*{argmax}_{\mathbf{y}\in\mathcal{Y}^*} \mathrm{P}(\mathbf{y}|\mathbf{x})$$

- Exponentially many $\mathbf{y}$'s for a given input $\mathbf{x}$

Conditional Random Fields
(Lafferty, McCallum, Pereira 2001)

- Log-linear model of the conditional distribution:

$$\mathrm{P}(\mathbf{y}|\mathbf{x};\mathbf{w}) = \frac{\exp\{\mathbf{w}\cdot\mathbf{f}(\mathbf{x},\mathbf{y})\}}{Z(\mathbf{x})}$$

where

  - $\mathbf{x} = \mathbf{x}_1\mathbf{x}_2\ldots\mathbf{x}_n \in \mathcal{X}^*$
  - $\mathbf{y} = \mathbf{y}_1\mathbf{y}_2\ldots\mathbf{y}_n \in \mathcal{Y}^*$ and $\mathcal{Y} = \{1,\ldots,L\}$
  - $\mathbf{f}(\mathbf{x},\mathbf{y})$ is a feature vector of $\mathbf{x}$ and $\mathbf{y}$
  - $\mathbf{w}$ are model parameters

- To predict the best sequence

$$\hat{\mathbf{y}} = \operatorname*{argmax}_{\mathbf{y}\in\mathcal{Y}^*} \mathrm{P}(\mathbf{y}|\mathbf{x})$$

- Exponentially many $\mathbf{y}$'s for a given input $\mathbf{x}$
- Choose $\mathbf{f}(\mathbf{x},\mathbf{y})$ so that $\hat{\mathbf{y}}$ can be computed efficiently

Conditional Random Fields (CRFs)

- The model form is:

$$P(\mathbf{y}|\mathbf{x};\mathbf{w}) = \frac{\exp(\sum_{i=1}^{n} \mathbf{w}\cdot\mathbf{f}(\mathbf{x},i,\mathbf{y}_{i-1},\mathbf{y}_i))}{Z(\mathbf{x},\mathbf{w})}$$

where

$$Z(\mathbf{x},\mathbf{w}) = \sum_{\mathbf{z}\in\mathcal{Y}^*} \exp(\sum_{i=1}^{n} \mathbf{w}\cdot\mathbf{f}(\mathbf{x},i,\mathbf{z}_{i-1},\mathbf{z}_i))$$

- Features $\mathbf{f}(\dots)$ are given (they are problem-dependent)
- $\mathbf{w} \in \mathbb{R}^D$ are the parameters of the model
- CRFs are log-linear models on the feature functions

Conditional Random Fields: Three Problems

- Compute the probability of an output sequence $\mathbf{y}$ for $\mathbf{x}$

$$\mathsf{P}(\mathbf{y}|\mathbf{x};\mathbf{w})$$

- Decoding: predict the best output sequence for $\mathbf{x}$

$$\underset{\mathbf{y}\in\mathcal{Y}^*}{\operatorname{argmax}}\,\mathsf{P}(\mathbf{y}|\mathbf{x};\mathbf{w})$$

- Parameter estimation: given training data

$$((\mathbf{x}^{(1)},\mathbf{y}^{(1)}),(\mathbf{x}^{(2)},\mathbf{y}^{(2)}),\ldots,(\mathbf{x}^{(m)},\mathbf{y}^{(m)})) \quad ,$$

learn parameters $\mathbf{w}$

Decoding with CRFs

- Given $\mathbf{w}$, given $\mathbf{x}$, find:

$$\begin{aligned}
\underset{\mathbf{y} \in \mathcal{Y}^*}{\operatorname{argmax}} \mathsf{P}(\mathbf{y}|\mathbf{x}; \mathbf{w}) &= \underset{\mathbf{y}}{\operatorname{argmax}} \frac{\exp(\sum_{i=1}^{n} \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, i, \mathbf{y}_{i-1}, \mathbf{y}_i))}{Z(\mathbf{x}; \mathbf{w})} \\
&= \underset{\mathbf{y}}{\operatorname{argmax}} \exp(\sum_{i=1}^{n} \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, i, \mathbf{y}_{i-1}, \mathbf{y}_i)) \\
&= \underset{\mathbf{y}}{\operatorname{argmax}} \sum_{i=1}^{n} \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, i, \mathbf{y}_{i-1}, \mathbf{y}_i)
\end{aligned}$$

- We can use the Viterbi algorithm

Viterbi for CRFs

...and MEMMs

- Calculate in $O(nL^2)$:

$$\hat{\mathbf{y}} = \underset{\mathbf{y} \in \mathcal{Y}^n}{\text{argmax}} \sum_{i=1}^{n} \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, i, \mathbf{y}_{i-1}, \mathbf{y}_i)$$

- Define (score of optimal sequence for $\mathbf{x}_{1:i}$ ending with $a \in \mathcal{Y}$):

$$\delta_i(a) = \max_{\mathbf{y} \in \mathcal{Y}^i : \mathbf{y}_i = a} \sum_{j=1}^{i} \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, j, \mathbf{y}_{j-1}, \mathbf{y}_j)$$

- Use the following recursions, for all $a \in \mathcal{Y}$:

$$\begin{aligned} \delta_1(a) &= \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, 1, \mathbf{y}_0 = \text{NULL}, a) \\ \delta_i(a) &= \max_{b \in \mathcal{Y}} \delta_{i-1}(b) + \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, i, b, a) \end{aligned}$$

- The optimal score for $\mathbf{x}$ is $\max_{a \in \mathcal{Y}} \delta_n(a)$
- The optimal sequence $\hat{\mathbf{y}}$ can be recovered through *pointers*

Parameter Estimation in CRFs

- How to estimate model parameters $\mathbf{w}$ given a training set:

$$\left\{(\mathbf{x}^{(1)}, \mathbf{y}^{(1)}), (\mathbf{x}^{(2)}, \mathbf{y}^{(2)}), \ldots, (\mathbf{x}^{(m)}, \mathbf{y}^{(m)})\right\}$$

- Let's define the conditional log-likelihood of the data:

$$L(\mathbf{w}) = \frac{1}{m} \sum_{k=1}^{m} \log P(\mathbf{y}^{(k)} | \mathbf{x}^{(k)}; \mathbf{w})$$

- $L(\mathbf{w})$ measures how well $\mathbf{w}$ explains the data. A good value for $\mathbf{w}$ will give a high value for $P(\mathbf{y}^{(k)} | \mathbf{x}^{(k)}; \mathbf{w})$ for all $k = 1 \ldots m$.
- We want $\mathbf{w}$ that maximizes $L(\mathbf{w})$

Learning the Parameters of a CRF

- Recall previous lecture on log-linear / maximum-entropy models
- Find:

$$\mathbf{w}^* = \underset{\mathbf{w} \in \mathbb{R}^D}{\text{argmax}}\, L(\mathbf{w}) - \frac{\lambda}{2} ||\mathbf{w}||^2$$

  where
  - The first term is the log-likelihood of the data
  - The second term is a regularization term, it penalizes solutions with large norm
  - λ is a parameter to control the trade-off between fitting the data and model complexity

Learning the Parameters of a CRF

- So we want to find:

$$\mathbf{w}^* = \operatorname*{argmax}_{\mathbf{w}\in\mathbb{R}^d} L'(\mathbf{w})$$
$$= \operatorname*{argmax}_{\mathbf{w}\in\mathbb{R}^d} \left(\frac{1}{m} \sum_{k=1}^{m} \log P(y^{(k)}|x^{(k)};\mathbf{w}) - \frac{\lambda}{2}||\mathbf{w}||^2 \right)$$

- In general there is no analytical solution to this optimization
- ... but it is a convex function $\Rightarrow$ We use iterative techniques, i.e. gradient-based optimization
- Very fast algorithms exist (e.g. LBFGS)

Learning the Parameters of a CRF: Gradient step

- Initialize $\mathbf{w} = \mathbf{0}$
- Repeat
  - Compute gradient $\delta = (\delta_1, \ldots, \delta_d)$, where:

$$\delta_j = \frac{\partial L'(\mathbf{w})}{\partial \mathbf{w}_j} \quad \forall j = 1 \ldots d$$

  - Compute step size

$$\beta^* = \operatorname*{argmax}_{\beta \in \mathbb{R}} L'(\mathbf{w} + \beta\delta)$$

  - Move $\mathbf{w}$ in the direction of the gradient

$$\mathbf{w} \leftarrow \mathbf{w} + \beta^*\delta$$

- until convergence ($||\delta|| < \epsilon$)

Computing the gradient

$$
\begin{aligned}
\frac{\partial L'(\mathbf{w})}{\partial \mathbf{w}_j} &= \frac{1}{m}\sum_{k=1}^{m} \mathbf{f}_j(\mathbf{x}^{(k)}, \mathbf{y}^{(k)}) \\
&- \sum_{k=1}^{m} \sum_{\mathbf{y} \in \mathcal{Y}^{n_k}} \mathrm{P}(\mathbf{y}|\mathbf{x}^{(k)}; \mathbf{w})\ \mathbf{f}_j(\mathbf{x}^{(k)}, \mathbf{y}) \\
&- \lambda \mathbf{w}_j
\end{aligned}
$$

where

$$
\mathbf{f}_j(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n} \mathbf{f}_j(\mathbf{x}, i, \mathbf{y}_{i-1}, \mathbf{y}_i)
$$

- First term: observed mean feature value
- Second term: expected feature value under current $\mathbf{w}$

Computing the gradient

- The first term is easy to compute, by counting explicitly over all sequence elements:

$$\frac{1}{m}\sum_{k=1}^{m}\sum_{i}\mathbf{f}_j(\mathbf{x}, i, \mathbf{y}_{i-1}^{(k)}, \mathbf{y}_i^{(k)})$$

- The second term is more involved, because it sums over all sequences $\mathbf{y} \in \mathcal{Y}^{n_k}$

$$\sum_{k=1}^{m}\sum_{\mathbf{y}\in\mathcal{Y}^{n_k}} \mathsf{P}(\mathbf{y}|\mathbf{x}^{(k)}; \mathbf{w}) \sum_{i}\mathbf{f}_j(\mathbf{x}^{(k)}, i, \mathbf{y}_{i-1}, \mathbf{y}_i)$$

Computing the gradient

- For a given training example $(\mathbf{x}^{(k)}, \mathbf{y}^{(k)})$:

$$\sum_{\mathbf{y} \in \mathcal{Y}^{n_k}} P(\mathbf{y}|\mathbf{x}^{(k)}; \mathbf{w}) \sum_{i=1}^{n} \mathbf{f}_j(\mathbf{x}^{(k)}, i, \mathbf{y}_{i-1}, \mathbf{y}_i) = \sum_{i=1}^{n} \sum_{a,b \in \mathcal{Y}} \mu_i^k(a, b) \mathbf{f}_j(\mathbf{x}^{(k)}, i, a, b)$$

where

$$\mu_i^k(a, b) = \sum_{\mathbf{y} \in \mathcal{Y}^{n_k} \;:\; \mathbf{y}_{i-1}=a,\ \mathbf{y}_i=b} P(\mathbf{y}|\mathbf{x}^{(k)}; \mathbf{w})$$

- The quantities μ_i^k can be computed efficiently in $O(nL^2)$ using the forward-backward algorithm

Forward-Backward for CRFs

- Assume fixed $\mathbf{x}$. Calculate in $O(nL^2)$

$$\mu_i(a,b) = \sum_{\mathbf{y} \in \mathcal{Y}^n : \mathbf{y}_{i-1}=a, \mathbf{y}_i=b} P(\mathbf{y}|\mathbf{x};\mathbf{w}) \quad , 1 \leqslant i \leqslant n;\ a,b \in \mathcal{Y}$$

- Define (forward and backward quantities):

$$\alpha_i(a) = \sum_{\mathbf{y} \in \mathcal{Y}^i : \mathbf{y}_i = a} \exp(\textstyle\sum_{j=1}^{i} \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, j, \mathbf{y}_{j-1}, \mathbf{y}_j))$$

$$\beta_i(b) = \sum_{\mathbf{y} \in \mathcal{Y}^{(n-i+1)} : \mathbf{y}_1 = b} \exp(\textstyle\sum_{j=2}^{n-i+1} \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, i+j-1, \mathbf{y}_{j-1}, \mathbf{y}$$

- Compute recursively $\alpha_i(a)$ and $\beta_i(b)$ (similar to Viterbi)
- $Z = \sum_a \alpha_n(a)$
- $\mu_i(a,b) = \alpha_{i-1}(a) \cdot \exp(\mathbf{w} \cdot \mathbf{f}(\mathbf{x}, i, a, b)) \cdot \beta_i(b)/Z$

Compute the probability of a label sequence

$$P(\mathbf{y}|\mathbf{x}, \mathbf{w}) = \frac{1}{Z(\mathbf{x}; \mathbf{w})} \exp(\sum_i \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, i, \mathbf{y}_{i-1}, \mathbf{y}_i))$$

where

$$Z(\mathbf{x}; \mathbf{w}) = \sum_{\mathbf{z} \in \mathcal{Y}^n} \exp(\sum_i \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, i, \mathbf{z}_{i-1}, \mathbf{z}_i))$$

- Compute $Z(\mathbf{x}; \mathbf{w})$ efficiently, using the forward algorithm

CRFs: summary so far

- Log-linear models for sequence prediction, $\mathrm{P}(\mathbf{y}|\mathbf{x};\mathbf{w})$
- Computations factorize on label bigrams
- Model form:

$$\operatorname*{argmax}_{\mathbf{y}\in\mathcal{Y}^*} \sum_{i} \mathbf{w}\cdot\mathbf{f}(\mathbf{x}, i, \mathbf{y}_{i-1}, \mathbf{y}_i)$$

- Decoding: uses Viterbi (from HMMs)
- Parameter estimation:
  - Gradient-based methods, in practice L-BFGS
  - Computation of gradient uses forward-backward (from HMMs)

CRFs: summary so far

- Log-linear models for sequence prediction, $\mathrm{P}(\mathbf{y}|\mathbf{x};\mathbf{w})$
- Computations factorize on label bigrams
- Model form:

$$\underset{\mathbf{y}\in\mathcal{Y}^*}{\operatorname{argmax}} \sum_i \mathbf{w}\cdot\mathbf{f}(\mathbf{x}, i, \mathbf{y}_{i-1}, \mathbf{y}_i)$$

- Decoding: uses Viterbi (from HMMs)
- Parameter estimation:
  - Gradient-based methods, in practice L-BFGS
  - Computation of gradient uses forward-backward (from HMMs)
- Next Questions: MEMMs or CRFs? HMMs or CRFs?

MEMMs and CRFs

MEMMs: $$P(\mathbf{y} \mid \mathbf{x}) = \prod_{i=1}^{n} \frac{\exp(\mathbf{w} \cdot \mathbf{f}(\mathbf{x}, i, \mathbf{y}_{i-1}, \mathbf{y}_i))}{Z(\mathbf{x}, i, \mathbf{y}_{i-1}; \mathbf{w})}$$

CRFs: $$P(\mathbf{y} \mid \mathbf{x}) = \frac{\exp(\sum_{i=1}^{n} \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, i, \mathbf{y}_{i-1}, \mathbf{y}_i))}{Z(\mathbf{x})}$$

- MEMMs locally normalized; CRFs globally normalized
- MEMM assume that $P(\mathbf{y}_i \mid \mathbf{x}_{1:n}, \mathbf{y}_{1:i-1}) = P(\mathbf{y}_i \mid \mathbf{x}_{1:n}, \mathbf{y}_{i-1})$
- Both exploit the same factorization, i.e. same features
- Same computations to compute $\text{argmax}_{\mathbf{y}} P(\mathbf{y} \mid \mathbf{x})$
- MEMMs are cheaper to train
- CRFs are easier to extend to other structures (e.g. parsing trees)

HMMs for sequence prediction

- $\mathbf{x}$ are the observations, $\mathbf{y}$ are the (un)hidden states
- HMMs model the joint distributon $\mathsf{P}(\mathbf{x}, \mathbf{y})$
- Parameters: (assume $\mathcal{X} = \{1, \ldots, k\}$ and $\mathcal{Y} = \{1, \ldots, l\}$)
  - $\pi \in \mathbb{R}^l$, $\pi_a = \mathsf{P}(\mathbf{y}_1 = a)$
  - $\mathsf{T} \in \mathbb{R}^{l \times l}$, $\mathsf{T}_{a,b} = \mathsf{P}(\mathbf{y}_i = b | \mathbf{y}_{i-1} = a)$
  - $\mathsf{O} \in \mathbb{R}^{l \times k}$, $\mathsf{O}_{a,c} = \mathsf{P}(\mathbf{x}_i = c | \mathbf{y}_i = a)$
- Model form

$$\mathsf{P}(\mathbf{x}, \mathbf{y}) = \pi_{\mathbf{y}_1} \mathsf{O}_{\mathbf{y}_1, \mathbf{x}_1} \prod_{i=2}^{n} \mathsf{T}_{\mathbf{y}_{i-1}, \mathbf{y}_i} \mathsf{O}_{\mathbf{y}_i, \mathbf{x}_i}$$

- Parameter Estimation: maximum likelihood by counting events and normalizing

HMMs and CRFs

- In CRFs: $\hat{\mathbf{y}} = \text{amax}_{\mathbf{y}} \sum_i \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, i, \mathbf{y}_{i-1}, \mathbf{y}_i)$

- In HMMs:
$$\begin{aligned} \hat{\mathbf{y}} &= \text{amax}_{\mathbf{y}} \, \pi_{\mathbf{y}_1} O_{\mathbf{y}_1, \mathbf{x}_1} \prod_{i=2}^{n} T_{\mathbf{y}_{i-1}, \mathbf{y}_i} O_{\mathbf{y}_i, \mathbf{x}_i} \\ &= \text{amax}_{\mathbf{y}} \log(\pi_{\mathbf{y}_1} O_{\mathbf{y}_1, \mathbf{x}_1}) + \sum_{i=2}^{n} \log(T_{\mathbf{y}_{i-1}, \mathbf{y}_i} O_{\mathbf{y}_i, \mathbf{x}_i}) \end{aligned}$$

- An HMM can be ported into a CRF by setting:

$\mathbf{f}_j(\mathbf{x}, i, y, y')$	$\mathbf{w}_j$

HMMs and CRFs

- In CRFs: $\hat{\mathbf{y}} = \text{amax}_{\mathbf{y}} \sum_i \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, i, \mathbf{y}_{i-1}, \mathbf{y}_i)$

- In HMMs:
$$\hat{\mathbf{y}} = \text{amax}_{\mathbf{y}}\, \pi_{\mathbf{y}_1} O_{\mathbf{y}_1,\mathbf{x}_1} \prod_{i=2}^{n} T_{\mathbf{y}_{i-1},\mathbf{y}_i} O_{\mathbf{y}_i,\mathbf{x}_i}$$
$$= \text{amax}_{\mathbf{y}} \log(\pi_{\mathbf{y}_1} O_{\mathbf{y}_1,\mathbf{x}_1}) + \sum_{i=2}^{n} \log(T_{\mathbf{y}_{i-1},\mathbf{y}_i} O_{\mathbf{y}_i,\mathbf{x}_i})$$

- An HMM can be ported into a CRF by setting:

$\mathbf{f}_j(\mathbf{x}, i, y, y')$	$\mathbf{w}_j$
$i = 1$ & $y' = a$	$\log(\pi_a)$

HMMs and CRFs

- In CRFs: $\hat{\mathbf{y}} = \text{amax}_{\mathbf{y}} \sum_i \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, i, \mathbf{y}_{i-1}, \mathbf{y}_i)$
- In HMMs:
  $$\hat{\mathbf{y}} = \text{amax}_{\mathbf{y}} \pi_{\mathbf{y}_1} O_{\mathbf{y}_1,\mathbf{x}_1} \prod_{i=2}^{n} T_{\mathbf{y}_{i-1},\mathbf{y}_i} O_{\mathbf{y}_i,\mathbf{x}_i}$$
  $$= \text{amax}_{\mathbf{y}} \log(\pi_{\mathbf{y}_1} O_{\mathbf{y}_1,\mathbf{x}_1}) + \sum_{i=2}^{n} \log(T_{\mathbf{y}_{i-1},\mathbf{y}_i} O_{\mathbf{y}_i,\mathbf{x}_i})$$
- An HMM can be ported into a CRF by setting:

$\mathbf{f}_j(\mathbf{x}, i, y, y')$	$\mathbf{w}_j$
$i = 1 \ \& \ y' = a$	$\log(\pi_a)$
$i > 1 \ \& \ y = a \ \& \ y' = b$	$\log(T_{a,b})$

HMMs and CRFs

- In CRFs: $\hat{\mathbf{y}} = \text{amax}_{\mathbf{y}} \sum_i \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, i, \mathbf{y}_{i-1}, \mathbf{y}_i)$

- In HMMs:
  $$\hat{\mathbf{y}} = \text{amax}_{\mathbf{y}}\, \pi_{\mathbf{y}_1} O_{\mathbf{y}_1, \mathbf{x}_1} \prod_{i=2}^{n} T_{\mathbf{y}_{i-1}, \mathbf{y}_i} O_{\mathbf{y}_i, \mathbf{x}_i}$$
  $$= \text{amax}_{\mathbf{y}} \log(\pi_{\mathbf{y}_1} O_{\mathbf{y}_1, \mathbf{x}_1}) + \sum_{i=2}^{n} \log(T_{\mathbf{y}_{i-1}, \mathbf{y}_i} O_{\mathbf{y}_i, \mathbf{x}_i})$$

- An HMM can be ported into a CRF by setting:

$\mathbf{f}_j(\mathbf{x}, i, y, y')$	$\mathbf{w}_j$
$i = 1$ & $y' = a$	$\log(\pi_a)$
$i > 1$ & $y = a$ & $y' = b$	$\log(T_{a,b})$
$y' = a$ & $\mathbf{x}_i = c$	$\log(O_{a,b})$

- Hence, HMM parameters $\subset$ CRF parameters

HMMs and CRFs: main differences

- Representation:
  - HMM "features" are tied to the generative process.
  - CRF features are **very** flexible. They can look at the whole input $\mathbf{x}$ paired with a label bigram (y, y').
  - In practice, for prediction tasks, "good" discriminative features can improve accuracy **a lot**.
- Parameter estimation:
  - HMMs focus on explaining the data, both $\mathbf{x}$ and $\mathbf{y}$.
  - CRFs focus on the mapping from $\mathbf{x}$ to $\mathbf{y}$.
  - A priori, it is hard to say which paradigm is better.
  - Same dilemma as Naive Bayes vs. Maximum Entropy.