Advanced Human Language Technologies

Sequence prediction

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

Outline

Outline

Examples - Named Entity Recognition (NER)

Sequence Prediction
Examples
Approaches
Log-linear Models for Sequence Prediction

y	PER	-	QNT	-	-	ORG	ORG	-	TIME
x	Jim	bought	300	shares	of	Acme	Corp.	in	2006

Examples - Named Entity Recognition (NER)

y	PER	-	QNT	-	-	ORG	ORG	-	TIME
x	Jim	bought	300	shares	of	Acme	Corp.	in	2006

y	PER	PER	-	-	LOC
x	Jack	London	went	to	Paris

y	PER	PER	-	-	LOC
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

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

$\mathbf{y}$	DT	NN	NN	VBD	DT	JJ	NN	.
$\mathbf{x}$	The	fox	jumps	scared	the	lazy	dog	.

Outline

Problem Formulation

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

- **Goal:** given training data

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

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

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

- $\mathbf{f}(\mathbf{x}, i, y)$ represents an assignment of label y 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 $\mathbf{w}$ (MEM, SVM, ...)

Approach 1: Local Classifiers

?
Jack London went to Paris

Decompose the sequence into n classification problems:

- A classifier predicts individual labels at each position

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

- $\mathbf{f}(\mathbf{x}, i, y)$ represents an assignment of label y 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 $\mathbf{w}$ (MEM, SVM, . . .)
- At test time, predict the best sequence by a simple concatenation of the best label for each position

Indicator Features

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

$$\mathbf{f}(\mathbf{x}, i, y) = (\ \mathbf{f}_1(\mathbf{x}, i, y), \ldots, \mathbf{f}_j(\mathbf{x}, i, y), \ldots, \mathbf{f}_d(\mathbf{x}, i, y)\ )$$

- What's in a feature $\mathbf{f}_j(\mathbf{x}, i, y)$?
  - Anything we can compute using $\mathbf{x}$ and i and y
  - Anything that indicates whether y is a good (or bad) label for $\mathbf{x}_i$
  - Indicator features: binary-valued features looking at a single simple property

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

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

More Features for NE Recognition

PER
Jack London went to Paris

In practice, construct $\mathbf{f}(\mathbf{x}, i, y)$ 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?
  - pref/suff 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 possible labels y

More Features for NE Recognition

PER	PER	-		
Jack	London	went	to	Paris

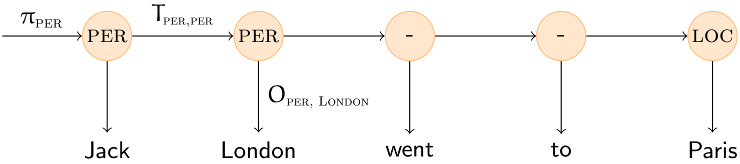
In practice, construct $\mathbf{f}(\mathbf{x}, i, y)$ by ...

- 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?
  - pref/suff 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 possible labels y

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

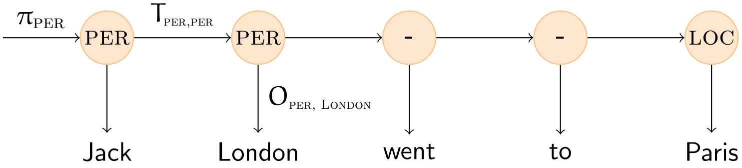
Outline

Approach 2: HMM for Sequence Prediction



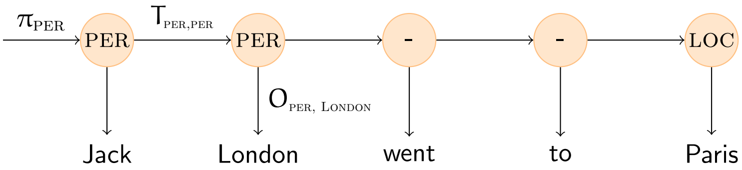
- Define an HMM were each label is a state
- Model parameters:
  - π_y : probability of starting with label y
  - $T_{yy'}$: probability of transitioning from label y to y'
  - O_{yx}: probability of generating symbol x given label y
- Predictions: $p(\mathbf{x}, \mathbf{y}) = \pi_{\mathbf{y}_1} O_{\mathbf{y}_1 \mathbf{x}_1} \prod_{i>1} T_{\mathbf{y}_{i-1}\mathbf{y}_i} O_{\mathbf{y}_i \mathbf{x}_i}$
- 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 $O_{\mathbf{y}_i \mathbf{x}_i} = p(\mathbf{x}_i \mid \mathbf{y}_i)$
  - 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:

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

- 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 $\mathbf{x} \rightarrow \mathbf{y}$

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

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:

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

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

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
$\mathbf{y}''$	-	PER	-	LOC	-

$f_1(x, i, y_{i-1}, y_i) = 1$ **iff** $\mathbf{x}_i =$"London" & $\mathbf{y}_{i-1} = \text{PER}$ & $\mathbf{y}_i = \text{PER}$

$f_2(x, i, y_{i-1}, y_i) = 1$ **iff** $\mathbf{x}_i =$"London" & $\mathbf{y}_{i-1} = \text{PER}$ & $\mathbf{y}_i = \text{LOC}$

$f_3(x, i, y_{i-1}, y_i) = 1$ **iff** $\mathbf{x}_{i-1}$ ~/(in|to|at)/ & $\mathbf{x}_i$ ~/^[A-Z]/ & $\mathbf{y}_i = \text{LOC}$

$f_4(x, i, y_{i-1}, y_i) = 1$ **iff** $\mathbf{y}_i = \text{LOC}$ & $\text{WORLD-CITIES}(\mathbf{x}_i) = 1$

$f_5(x, i, y_{i-1}, y_i) = 1$ **iff** $\mathbf{y}_i = \text{PER}$ & $\text{FIRST-NAMES}(\mathbf{x}_i) = 1$

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
$\mathbf{y}''$	-	PER	-	LOC	-

$f_1(x, i, y_{i-1}, y_i) = 1$ **iff** $\mathbf{x}_i =$"London" & $\mathbf{y}_{i-1} = \textsc{per}$ & $\mathbf{y}_i = \textsc{per}$

$f_2(x, i, y_{i-1}, y_i) = 1$ **iff** $\mathbf{x}_i =$"London" & $\mathbf{y}_{i-1} = \textsc{per}$ & $\mathbf{y}_i = \textsc{loc}$

$f_3(x, i, y_{i-1}, y_i) = 1$ **iff** $\mathbf{x}_{i-1}$ ~/(in|to|at)/ & $\mathbf{x}_i$ ~/^[A-Z]/ & $\mathbf{y}_i = \textsc{loc}$

$f_4(x, i, y_{i-1}, y_i) = 1$ **iff** $\mathbf{y}_i = \textsc{loc}$ & $\textsc{world-cities}(\mathbf{x}_i) = 1$

$f_5(x, i, y_{i-1}, y_i) = 1$ **iff** $\mathbf{y}_i = \textsc{per}$ & $\textsc{first-names}(\mathbf{x}_i) = 1$

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
$\mathbf{y}''$	-	PER	-	LOC	-

$\mathbf{f}_1(x, i, y_{i-1}, y_i) = 1$ **iff** $\mathbf{x}_i =$"London" & $\mathbf{y}_{i-1} = \textsc{per}$ & $\mathbf{y}_i = \textsc{per}$

$\mathbf{f}_2(x, i, y_{i-1}, y_i) = 1$ **iff** $\mathbf{x}_i =$"London" & $\mathbf{y}_{i-1} = \textsc{per}$ & $\mathbf{y}_i = \textsc{loc}$

$\mathbf{f}_3(x, i, y_{i-1}, y_i) = 1$ **iff** $\mathbf{x}_{i-1} \sim$/(in|to|at)/ & $\mathbf{x}_i \sim$/^[A-Z]/ & $\mathbf{y}_i = \textsc{loc}$

$\mathbf{f}_4(x, i, y_{i-1}, y_i) = 1$ **iff** $\mathbf{y}_i = \textsc{loc}$ & $\textsc{world-cities}(\mathbf{x}_i) = 1$

$\mathbf{f}_5(x, i, y_{i-1}, y_i) = 1$ **iff** $\mathbf{y}_i = \textsc{per}$ & $\textsc{first-names}(\mathbf{x}_i) = 1$

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
$\mathbf{y}''$	-	PER	-	LOC	-

$f_1(x, i, y_{i-1}, y_i) = 1$ **iff** $\mathbf{x}_i =$ "London" & $\mathbf{y}_{i-1} = \text{PER}$ & $\mathbf{y}_i = \text{PER}$

$f_2(x, i, y_{i-1}, y_i) = 1$ **iff** $\mathbf{x}_i =$ "London" & $\mathbf{y}_{i-1} = \text{PER}$ & $\mathbf{y}_i = \text{LOC}$

$f_3(x, i, y_{i-1}, y_i) = 1$ **iff** $\mathbf{x}_{i-1} \sim$/(in|to|at)/ & $\mathbf{x}_i \sim$/^[A-Z]/ & $\mathbf{y}_i = \text{LOC}$

$f_4(x, i, y_{i-1}, y_i) = 1$ **iff** $\mathbf{y}_i = \text{LOC}$ & $\text{WORLD-CITIES}(\mathbf{x}_i) = 1$

$f_5(x, i, y_{i-1}, y_i) = 1$ **iff** $\mathbf{y}_i = \text{PER}$ & $\text{FIRST-NAMES}(\mathbf{x}_i) = 1$

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
$\mathbf{y}''$	-	PER	-	LOC	-

$f_1(x, i, y_{i-1}, y_i) = 1$ **iff** $\mathbf{x}_i =$"London" & $\mathbf{y}_{i-1} = \text{PER}$ & $\mathbf{y}_i = \text{PER}$

$f_2(x, i, y_{i-1}, y_i) = 1$ **iff** $\mathbf{x}_i =$"London" & $\mathbf{y}_{i-1} = \text{PER}$ & $\mathbf{y}_i = \text{LOC}$

$f_3(x, i, y_{i-1}, y_i) = 1$ **iff** $\mathbf{x}_{i-1} \sim$/(in|to|at)/ & $\mathbf{x}_i \sim$/^[A-Z]/ & $\mathbf{y}_i = \text{LOC}$

$f_4(x, i, y_{i-1}, y_i) = 1$ **iff** $\mathbf{y}_i = \text{LOC}$ & $\text{WORLD-CITIES}(\mathbf{x}_i) = 1$

$f_5(x, i, y_{i-1}, y_i) = 1$ **iff** $\mathbf{y}_i = \text{PER}$ & $\text{FIRST-NAMES}(\mathbf{x}_i) = 1$

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

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

- Note the linearity of the expression:

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

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:

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

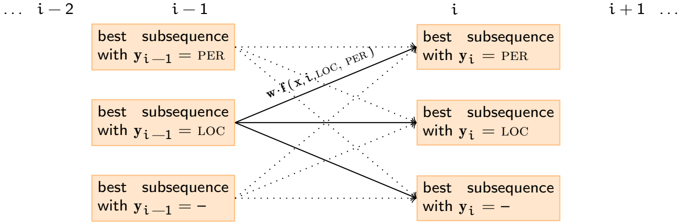
- Next questions:
  - How do we solve the argmax problem?
  - How do we learn $\mathbf{w}$?

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

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

- **Definition:**
  $\delta_i(a)$ = 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, $\forall a \in \mathcal{Y}$:

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

- 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

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

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

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
  $$\{(\mathbf{x}^{(1)}, \mathbf{y}^{(1)}), (\mathbf{x}^{(2)}, \mathbf{y}^{(2)}), \ldots, (\mathbf{x}^{(m)}, \mathbf{y}^{(m)})\}$$
  learn a model $\mathbf{x} \rightarrow \mathbf{y}$

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

- 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
  $$\left\{(\mathbf{x}^{(1)}, \mathbf{y}^{(1)}), (\mathbf{x}^{(2)}, \mathbf{y}^{(2)}), \ldots, (\mathbf{x}^{(m)}, \mathbf{y}^{(m)})\right\}$$
  learn a model $\mathbf{x} \rightarrow \mathbf{y}$

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

- 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 2000)

- 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:

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

Decoding with MEMMs

- Decompose tagging problem:

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

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

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

- We can use the Viterbi algorithm

Parameter Estimation with MEMMs

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

$$P(y_i \mid \mathbf{x}, i, y_{i-1}) = \frac{\exp(\mathbf{w} \cdot \mathbf{f}(\mathbf{x}, i, y_{i-1}, y_i))}{Z(\mathbf{x}, i)}$$

  where

  - $\mathbf{x}$ is an input sequence
  - y_i and y_{i-1} are tags
  - $\mathbf{f}(\mathbf{x}, i, y_{i-1}, y_i)$ is a feature vector of $\mathbf{x}$, the position to be tagged, the previous tag and the current tag

Outline

Conditional Random Fields

(Lafferty, McCallum, Pereira 2001)

- Log-linear model of the conditional distribution:

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

  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

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

Conditional Random Fields

(Lafferty, McCallum, Pereira 2001)

- Log-linear model of the conditional distribution:

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

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

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

- 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:

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

  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

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

- 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:

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

  where

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

- Features $\mathbf{f}(\ldots)$ 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

Decoding with CRFs

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

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

- We can use the Viterbi algorithm

Parameter Estimation in CRFs

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

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

- We define the conditional log-likelihood of the data (recall lecture on log-linear models):

$$L(\mathbf{w}) = \frac{1}{m} \sum_{k=1}^{m} \log P(\mathbf{y}^{(k)} | \mathbf{x}^{(k)}; \mathbf{w}) - \frac{\lambda}{2} ||\mathbf{w}||^2$$

- $L(\mathbf{w})$ 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 $k = 1 \ldots m$.
- $\frac{\lambda}{2}||\mathbf{w}||^2$ is a regularization penalizing solutions with large norm.
- λ is a parameter controllig the trade-off between fitting the data and model complexity.
- We want $\mathbf{w}$ that maximizes $L(\mathbf{w})$

Learning the Parameters of a CRF

- So we want to find:

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

- 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

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

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:

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

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

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

Computing the gradient

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

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

  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 $O(n|\mathcal{Y}|^2)$ using the forward-backward algorithm

Forward-Backward for CRFs

- For a given $\mathbf{x}$, we can compute in $O(n|\mathcal{Y}|^2)$:

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

- decomposing it as:

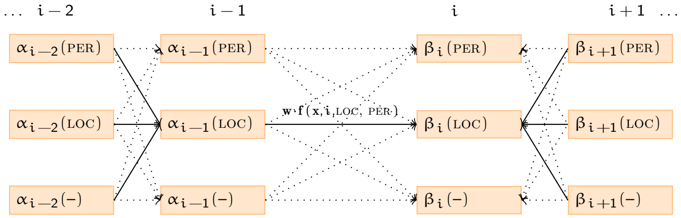
$$\begin{aligned}
\mu_i(a,b) &= \alpha_{i-1}(a) \cdot \exp(\mathbf{w}\cdot\mathbf{f}(\mathbf{x},i,a,b)) \cdot \beta_i(b)/Z \\
\alpha_{i-1}(a) &= \sum_{\substack{\mathbf{y}\in\mathcal{Y}^{i-1} \\ y_{i-1}=a}} \exp(\sum_{j=1}^{i-1} \mathbf{w}\cdot\mathbf{f}(\mathbf{x},j,y_{j-1},y_j)) \\
\beta_i(b) &= \sum_{\substack{\mathbf{y}\in\mathcal{Y}^{n-i+1} \\ y_1=b}} \exp(\sum_{j=2}^{n-i+1} \mathbf{w}\cdot\mathbf{f}(\mathbf{x},i+j-1,y_{i+j-1},y_{i+j})) \\
Z &= P(\mathbf{x}) = \sum_{a\in\mathcal{Y}} \alpha_n(a)
\end{aligned}$$

$\alpha_{i-1}(a)$: Probability that the label sequence for $\mathbf{x}_{1:i-1}$ ends with a.
$\beta_i(b)$: Probability that the label sequence for $\mathbf{x}_{i+1:n}$ starts with b.
Z: Probability of the sequence, normalization factor.

Forward-Backward for CRFs

- $\alpha_i(a)$ and $\beta_i(b)$ can be computed recursively, similarly to Viterbi algorithm:

$$
\begin{aligned}
\alpha_1(a) &= \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, 1, \mathbf{y}_0 = \text{NULL}, a) \\
\alpha_i(a) &= \sum_{b \in \mathcal{Y}} \alpha_{i-1}(b) \cdot \exp(\mathbf{w} \cdot \mathbf{f}(\mathbf{x}, i, b, a)) \\
\beta_n(b) &= 1 \\
\beta_i(b) &= \sum_{a \in \mathcal{Y}} \beta_{i+1}(a) \cdot \exp(\mathbf{w} \cdot \mathbf{f}(\mathbf{x}, i+1, b, a))
\end{aligned}
$$



CRF: Compute the probability of a label sequence

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

where

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

- $Z(\mathbf{x}; \mathbf{w})$ can be efficiently computed using the forward algorithm.

CRFs: summary

- 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

- 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)
- Next Questions: MEMMs or CRFs? HMMs or CRFs?

HMMs for sequence prediction

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

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

- 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 modelled with 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 modelled with 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 modelled with 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 modelled with 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 $\mathbf{y}$.

MEMMs and CRFs

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

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

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