

Learning Structured Predictors

Xavier Carreras



TALP Research Center
Universitat Politècnica de Catalunya

Supervised (Structured) Prediction

- ▶ Learning to predict: given training data

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

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

- ▶ Non-Structured Prediction: outputs \mathbf{y} are atomic
 - ▶ Binary prediction: $\mathbf{y} \in \{-1, +1\}$
 - ▶ Multiclass prediction: $\mathbf{y} \in \{1, 2, \dots, L\}$
- ▶ Structured Prediction: outputs \mathbf{y} are structured
 - ▶ Sequence prediction: \mathbf{y} are sequences
 - ▶ Parsing: \mathbf{y} are trees
 - ▶ ...

Named Entity Recognition

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

Named Entity Recognition

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

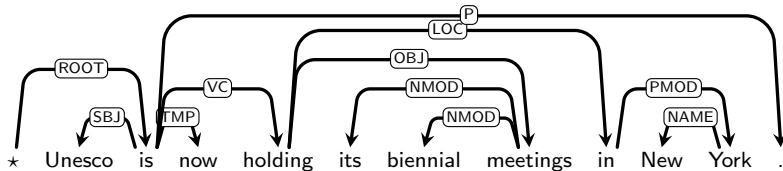
y	PER	PER	-	-	LOC
x	Jack	London	went	to	Paris

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

Part-of-speech Tagging

y	NNP	NNP	VBZ	NNP	.
x	Ms.	Haag	plays	Elianti	.

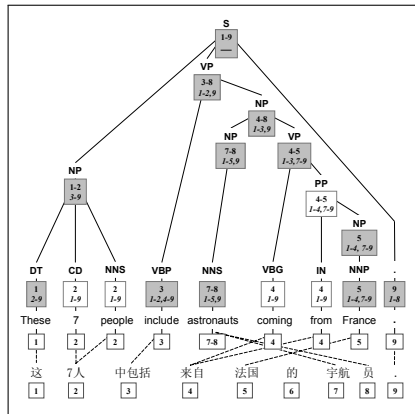
Syntactic Parsing



x are sentences

y are syntactic dependency trees

Machine Translation



(Galley et al 2006)

x are sentences in Chinese

y are sentences in English aligned to x

Object Detection



(Kumar and Hebert 2003)

x are images
 y are grids labeled with object types

Object Detection



(Kumar and Hebert 2003)

x are images
 y are grids labeled with object types

Today's Goals

- ▶ Introduce basic tools for structure prediction
 - ▶ We will restrict to sequence prediction
- ▶ Understand what tools we can use from standard classification
 - ▶ Learning paradigms and algorithms, in essence, work here too
 - ▶ However, computations behind algorithms are prohibitive
- ▶ Understand what tools we can use from existing formalisms for structured data
 - ▶ We will use inference algorithms for tractable computations
 - ▶ E.g., algorithms for HMMs (Viterbi, forward-backward) will play a major role in today's methods

Today's Goals

- ▶ Introduce basic tools for structure prediction
 - ▶ We will restrict to sequence prediction
- ▶ Understand what tools we can use from standard classification
 - ▶ Learning paradigms and algorithms, in essence, work here too
 - ▶ However, computations behind algorithms are prohibitive
- ▶ Understand what tools we can use from existing formalisms for structured data
 - ▶ We will use inference algorithms for tractable computations
 - ▶ E.g., algorithms for HMMs (Viterbi, forward-backward) will play a major role in today's methods

Sequence Prediction

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

Sequence Prediction

- ▶ $\mathbf{x} = x_1x_2 \dots x_n$ are input sequences, $x_i \in \mathcal{X}$
- ▶ $\mathbf{y} = y_1y_2 \dots y_n$ are output sequences, $y_i \in \{1, \dots, L\}$
- ▶ **Goal:** given training data

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

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

- ▶ What is the form of our prediction model?

Approach 1: Local Classifiers

Jack ? London went to Paris

Decompose the sequence into n classification problems:

- ▶ A classifier predicts individual labels at each position

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

- ▶ $\mathbf{f}(\mathbf{x}, i, l)$ 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

Approach 1: Local Classifiers

Jack ? went to Paris

Decompose the sequence into n classification problems:

- ▶ A classifier predicts individual labels at each position

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

- ▶ $\mathbf{f}(\mathbf{x}, i, l)$ 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 x_i

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

- ▶ 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 x_i
 - ▶ **Indicator features**: binary-valued features looking at a single simple property

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

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

More Features for NE Recognition

Jack ^{PER}
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 x_i
 - ▶ is x_i capitalized?
 - ▶ x_i has digits?
 - ▶ prefixes/suffixes of size 1, 2, 3, ...
 - ▶ is x_i a known location?
 - ▶ is 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

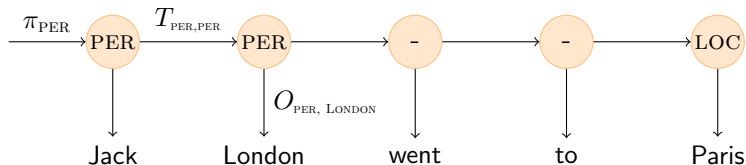
PER PER -
Jack London went to Paris

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

- ▶ Define a number of simple patterns of \mathbf{x} and i
 - ▶ current word x_i
 - ▶ is x_i capitalized?
 - ▶ x_i has digits?
 - ▶ prefixes/suffixes of size 1, 2, 3, ...
 - ▶ is x_i a known location?
 - ▶ is 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

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

Approach 2: HMM for Sequence Prediction

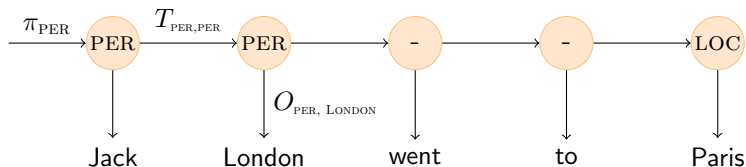


- ▶ Define an HMM where each label is a state
- ▶ Model parameters:
 - ▶ π_l : probability of starting with label l
 - ▶ $T_{l,l'}$: probability of transitioning from l to l'
 - ▶ $O_{l,x}$: probability of generating symbol x given label l
- ▶ Predictions:

$$p(\mathbf{x}, \mathbf{y}) = \pi_{y_1} O_{y_1, x_1} \prod_{i>1} T_{y_{i-1}, y_i} O_{y_i, x_i}$$

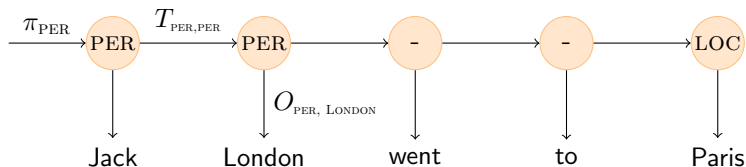
- ▶ Learning: relative counts + smoothing
- ▶ Prediction: Viterbi algorithm

Approach 2: Representation in HMM



- ▶ Label interactions are captured in the transition parameters
- ▶ But interactions between labels and input symbols are quite limited
 - ▶ Only $O_{y_i, x_i} = p(x_i | y_i)$
 - ▶ 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 y_i , token x_i is independent of anything else

Approach 2: Representation in HMM



- ▶ Label interactions are captured in the transition parameters
- ▶ But interactions between labels and input symbols are quite limited
 - ▶ Only $O_{y_i, x_i} = p(x_i | y_i)$
 - ▶ 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 y_i , token 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 x_i
- ▶ **Advantage:** feature-rich
- ▶ **Drawback:** no label interactions

HMM

- ▶ Form:

$$\pi_{y_1} O_{y_1, x_1} \prod_{i>1} T_{y_{i-1}, y_i} O_{y_i, x_i}$$

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

Approach 3: Global Sequence Predictors

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

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

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

Next questions: ...

- ▶ 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?

Approach 3: Global Sequence Predictors

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

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

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

Next questions: ...

- ▶ 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

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

- ▶ How do we represent entire sequences in $\mathbf{f}(\mathbf{x}, \mathbf{y})$?
 - ▶ Look at individual assignments y_i (standard classification)
 - ▶ Look at **bigrams** of outputs labels $\langle y_{i-1}, y_i \rangle$
 - ▶ Look at **trigrams** of outputs labels $\langle y_{i-2}, y_{i-1}, y_i \rangle$
 - ▶ Look at **n -grams** of outputs labels $\langle y_{i-n+1}, \dots, y_{i-1}, y_i \rangle$
 - ▶ Look at the full label sequence \mathbf{y} (intractable)
- ▶ A factored representation will lead to a tractable model

Factored Representations

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

- ▶ How do we represent entire sequences in $\mathbf{f}(\mathbf{x}, \mathbf{y})$?
 - ▶ Look at individual assignments y_i (standard classification)
 - ▶ Look at **bigrams** of outputs labels $\langle y_{i-1}, y_i \rangle$
 - ▶ Look at **trigrams** of outputs labels $\langle y_{i-2}, y_{i-1}, y_i \rangle$
 - ▶ Look at **n -grams** of outputs labels $\langle y_{i-n+1}, \dots, y_{i-1}, y_i \rangle$
 - ▶ Look at the full label sequence \mathbf{y} (intractable)
- ▶ A factored representation will lead to a tractable model

Factored Representations

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

- ▶ How do we represent entire sequences in $\mathbf{f}(\mathbf{x}, \mathbf{y})$?
 - ▶ Look at individual assignments y_i (standard classification)
 - ▶ Look at **bigrams** of outputs labels $\langle y_{i-1}, y_i \rangle$
 - ▶ Look at **trigrams** of outputs labels $\langle y_{i-2}, y_{i-1}, y_i \rangle$
 - ▶ Look at **n -grams** of outputs labels $\langle y_{i-n+1}, \dots, y_{i-1}, y_i \rangle$
 - ▶ Look at the full label sequence \mathbf{y} (intractable)
- ▶ A factored representation will lead to a tractable model

Factored Representations

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

- ▶ How do we represent entire sequences in $\mathbf{f}(\mathbf{x}, \mathbf{y})$?
 - ▶ Look at individual assignments y_i (standard classification)
 - ▶ Look at **bigrams** of outputs labels $\langle y_{i-1}, y_i \rangle$
 - ▶ Look at **trigrams** of outputs labels $\langle y_{i-2}, y_{i-1}, y_i \rangle$
 - ▶ Look at **n -grams** of outputs labels $\langle y_{i-n+1}, \dots, y_{i-1}, y_i \rangle$
 - ▶ Look at the full label sequence \mathbf{y} (intractable)
- ▶ A factored representation will lead to a tractable model

Bigram Indicator Features

	1	2	3	4	5
y	PER	PER	-	-	LOC
x	Jack	London	went	to	Paris

- Indicator features:

$$\mathbf{f}_j(\mathbf{x}, i, y_{i-1}, y_i) = \begin{cases} 1 & \text{if } x_i = \text{"London"} \text{ and} \\ & y_{i-1} = \text{PER and } y_i = \text{PER} \\ 0 & \text{otherwise} \end{cases}$$

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

More Bigram Indicator Features

	1	2	3	4	5
x	Jack	London	went	to	Paris
y	PER	PER	-	-	LOC
y'	PER	LOC	-	-	LOC
y''	-	-	-	LOC	-
x'	My	trip	to	London	...

$\mathbf{f}_1(\dots) = 1$ iff $x_i = \text{"London"}$ and $y_{i-1} = \text{PER}$ and $y_i = \text{PER}$

$\mathbf{f}_2(\dots) = 1$ iff $x_i = \text{"London"}$ and $y_{i-1} = \text{PER}$ and $y_i = \text{LOC}$

$\mathbf{f}_3(\dots) = 1$ iff $x_{i-1} \sim /(\text{in}|\text{to}|\text{at})/$ and $x_i \sim /[\text{A-Z}]/$ and $y_i = \text{LOC}$

$\mathbf{f}_4(\dots) = 1$ iff $y_i = \text{LOC}$ and $\text{WORLD-CITIES}(x_i) = 1$

$\mathbf{f}_5(\dots) = 1$ iff $y_i = \text{PER}$ and $\text{FIRST-NAMES}(x_i) = 1$

More Bigram Indicator Features

	1	2	3	4	5
x	Jack	London	went	to	Paris
y	PER	PER	-	-	LOC
y'	PER	LOC	-	-	LOC
y''	-	-	-	LOC	-
x'	My	trip	to	London	...

$\mathbf{f}_1(\dots) = 1$ iff $x_i = \text{"London"}$ and $y_{i-1} = \text{PER}$ and $y_i = \text{PER}$

$\mathbf{f}_2(\dots) = 1$ iff $x_i = \text{"London"}$ and $y_{i-1} = \text{PER}$ and $y_i = \text{LOC}$

$\mathbf{f}_3(\dots) = 1$ iff $x_{i-1} \sim /(\text{in}|\text{to}|\text{at})/$ and $x_i \sim /[\text{A-Z}]/$ and $y_i = \text{LOC}$

$\mathbf{f}_4(\dots) = 1$ iff $y_i = \text{LOC}$ and $\text{WORLD-CITIES}(x_i) = 1$

$\mathbf{f}_5(\dots) = 1$ iff $y_i = \text{PER}$ and $\text{FIRST-NAMES}(x_i) = 1$

More Bigram Indicator Features

	1	2	3	4	5
x	Jack	London	went	to	Paris
y	PER	PER	-	-	LOC
y'	PER	LOC	-	-	LOC
y''	-	-	-	LOC	-
x'	My	trip	to	London	...

$\mathbf{f}_1(\dots) = 1$ iff $x_i = \text{"London"}$ and $y_{i-1} = \text{PER}$ and $y_i = \text{PER}$

$\mathbf{f}_2(\dots) = 1$ iff $x_i = \text{"London"}$ and $y_{i-1} = \text{PER}$ and $y_i = \text{LOC}$

$\mathbf{f}_3(\dots) = 1$ iff $x_{i-1} \sim /(\text{in}|\text{to}|\text{at})/$ and $x_i \sim /[\text{A-Z}]/$ and $y_i = \text{LOC}$

$\mathbf{f}_4(\dots) = 1$ iff $y_i = \text{LOC}$ and $\text{WORLD-CITIES}(x_i) = 1$

$\mathbf{f}_5(\dots) = 1$ iff $y_i = \text{PER}$ and $\text{FIRST-NAMES}(x_i) = 1$

More Bigram Indicator Features

	1	2	3	4	5
x	Jack	London	went	to	Paris
y	PER	PER	-	-	LOC
y'	PER	LOC	-	-	LOC
y''	-	-	-	LOC	-
x'	My	trip	to	London	...

$\mathbf{f}_1(\dots) = 1$ iff $x_i = \text{"London"}$ and $y_{i-1} = \text{PER}$ and $y_i = \text{PER}$

$\mathbf{f}_2(\dots) = 1$ iff $x_i = \text{"London"}$ and $y_{i-1} = \text{PER}$ and $y_i = \text{LOC}$

$\mathbf{f}_3(\dots) = 1$ iff $x_{i-1} \sim /(\text{in}|\text{to}|\text{at})/$ and $x_i \sim /[\text{A-Z}]/$ and $y_i = \text{LOC}$

$\mathbf{f}_4(\dots) = 1$ iff $y_i = \text{LOC}$ and $\text{WORLD-CITIES}(x_i) = 1$

$\mathbf{f}_5(\dots) = 1$ iff $y_i = \text{PER}$ and $\text{FIRST-NAMES}(x_i) = 1$

More Bigram Indicator Features

	1	2	3	4	5
x	Jack	London	went	to	Paris
y	PER	PER	-	-	LOC
y'	PER	LOC	-	-	LOC
y''	-	-	-	LOC	-
x'	My	trip	to	London	...

$\mathbf{f}_1(\dots) = 1$ iff $x_i = \text{"London"}$ and $y_{i-1} = \text{PER}$ and $y_i = \text{PER}$

$\mathbf{f}_2(\dots) = 1$ iff $x_i = \text{"London"}$ and $y_{i-1} = \text{PER}$ and $y_i = \text{LOC}$

$\mathbf{f}_3(\dots) = 1$ iff $x_{i-1} \sim /(\text{in}|\text{to}|\text{at})/$ and $x_i \sim /[\text{A-Z}]/$ and $y_i = \text{LOC}$

$\mathbf{f}_4(\dots) = 1$ iff $y_i = \text{LOC}$ and $\text{WORLD-CITIES}(x_i) = 1$

$\mathbf{f}_5(\dots) = 1$ iff $y_i = \text{PER}$ and $\text{FIRST-NAMES}(x_i) = 1$

Representations Factored at Bigrams

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

- ▶ $\mathbf{f}(\mathbf{x}, i, y_{i-1}, y_i)$
 - ▶ A d -dimensional feature vector of a label bigram at i
 - ▶ Each dimension is typically a boolean indicator (0 or 1)
- ▶ $\mathbf{f}(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^n \mathbf{f}(\mathbf{x}, i, y_{i-1}, y_i)$
 - ▶ 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

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

where

$$\mathbf{f}(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^n \mathbf{f}(\mathbf{x}, i, y_{i-1}, y_i)$$

- Note the linearity of the expression:

$$\begin{aligned} \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, \mathbf{y}) &= \mathbf{w} \cdot \sum_{i=1}^n \mathbf{f}(\mathbf{x}, i, y_{i-1}, y_i) \\ &= \sum_{i=1}^n \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, i, y_{i-1}, y_i) \end{aligned}$$

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

Linear Sequence Prediction

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

where

$$\mathbf{f}(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^n \mathbf{f}(\mathbf{x}, i, y_{i-1}, y_i)$$

- Note the linearity of the expression:

$$\begin{aligned} \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, \mathbf{y}) &= \mathbf{w} \cdot \sum_{i=1}^n \mathbf{f}(\mathbf{x}, i, y_{i-1}, y_i) \\ &= \sum_{i=1}^n \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, i, y_{i-1}, y_i) \end{aligned}$$

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

Linear Sequence Prediction

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

where

$$\mathbf{f}(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^n \mathbf{f}(\mathbf{x}, i, y_{i-1}, y_i)$$

- Note the linearity of the expression:

$$\begin{aligned} \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, \mathbf{y}) &= \mathbf{w} \cdot \sum_{i=1}^n \mathbf{f}(\mathbf{x}, i, y_{i-1}, y_i) \\ &= \sum_{i=1}^n \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, i, y_{i-1}, y_i) \end{aligned}$$

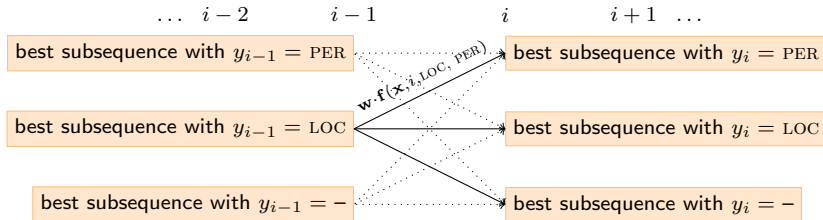
- Next questions:
 - How do we solve the [argmax](#) problem?
 - How do we learn \mathbf{w} ?

Predicting with Factored Sequence Models

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

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

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



Intuition for Viterbi

- ▶ Consider a fixed $\mathbf{x}_{1:n}$
- ▶ Assume we have the best sub-sequences up to position $i - 1$

1 \dots $i - 1$ i

best subsequence with $y_{i-1} = \text{PER}$

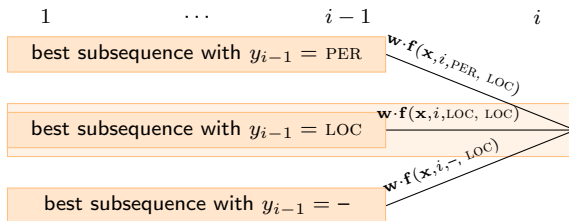
best subsequence with $y_{i-1} = \text{LOC}$

best subsequence with $y_{i-1} = -$

- ▶ What is the best sequence up to position i with $y_i = \text{LOC}$?

Intuition for Viterbi

- ▶ Consider a fixed $\mathbf{x}_{1:n}$
- ▶ Assume we have the best sub-sequences up to position $i - 1$



- ▶ What is the best sequence up to position i with $y_i = \text{LOC}$?

Viterbi for Linear Factored Predictors

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

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

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

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

$$\begin{aligned}\delta_1(a) &= \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, 1, y_0 = \text{NULL}, a) \\ \delta_i(a) &= \max_{b \in \mathcal{Y}} \delta_{i-1}(b) + \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, i, b, a)\end{aligned}$$

- 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**, learning \mathbf{w} :
 - ▶ Maximum-Entropy Markov Models (local)
 - ▶ Conditional Random Fields (global)
 - ▶ Structured Perceptron (global)
 - ▶ Structured SVM (global)

Log-linear Models for Sequence Prediction

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

Log-linear Models for Sequence Prediction

- ▶ Model the conditional distribution:

$$\Pr(\mathbf{y} \mid \mathbf{x}; \mathbf{w}) = \frac{\exp \{ \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, \mathbf{y}) \}}{Z(\mathbf{x}; \mathbf{w})}$$

where

- ▶ $\mathbf{x} = x_1 x_2 \dots x_n \in \mathcal{X}^*$
- ▶ $\mathbf{y} = y_1 y_2 \dots y_n \in \mathcal{Y}^*$ and $\mathcal{Y} = \{1, \dots, L\}$
- ▶ $\mathbf{f}(\mathbf{x}, \mathbf{y})$ represents \mathbf{x} and \mathbf{y} with d features
- ▶ $\mathbf{w} \in \mathbb{R}^d$ are the parameters of the model
- ▶ $Z(\mathbf{x}; \mathbf{w})$ is a normalizer called the *partition function*

$$Z(\mathbf{x}; \mathbf{w}) = \sum_{\mathbf{z} \in \mathcal{Y}^*} \exp \{ \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, \mathbf{z}) \}$$

- ▶ To predict the best sequence

$$\text{predict}(\mathbf{x}_{1:n}) = \operatorname{argmax}_{\mathbf{y} \in \mathcal{Y}^n} \Pr(\mathbf{y} \mid \mathbf{x})$$

Log-linear Models: Name

- ▶ Let's take the **log** of the conditional probability:

$$\begin{aligned}\log \Pr(\mathbf{y} \mid \mathbf{x}; \mathbf{w}) &= \log \frac{\exp\{\mathbf{w} \cdot \mathbf{f}(\mathbf{x}, \mathbf{y})\}}{Z(\mathbf{x}; \mathbf{w})} \\ &= \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, \mathbf{y}) - \log \sum_y \exp\{\mathbf{w} \cdot \mathbf{f}(\mathbf{x}, \mathbf{y})\} \\ &= \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, \mathbf{y}) - \log Z(\mathbf{x}; \mathbf{w})\end{aligned}$$

- ▶ Partition function: $Z(\mathbf{x}; \mathbf{w}) = \sum_{\mathbf{y}} \exp\{\mathbf{w} \cdot \mathbf{f}(\mathbf{x}, \mathbf{y})\}$
- ▶ $\log Z(\mathbf{x}; \mathbf{w})$ is a constant for a fixed \mathbf{x}
- ▶ In the **log** space, computations are **linear**,
i.e., we model log-probabilities using a linear predictor

Making Predictions with Log-Linear Models

- For tractability, assume $\mathbf{f}(\mathbf{x}, \mathbf{y})$ decomposes into bigrams:

$$\mathbf{f}(\mathbf{x}_{1:n}, \mathbf{y}_{1:n}) = \sum_{i=1}^n \mathbf{f}(\mathbf{x}, i, y_{i-1}, y_i)$$

- Given \mathbf{w} , given $\mathbf{x}_{1:n}$, find:

$$\begin{aligned} \operatorname{argmax}_{\mathbf{y}_{1:n}} \Pr(\mathbf{y}_{1:n} | \mathbf{x}_{1:n}; \mathbf{w}) &= \operatorname{amax}_{\mathbf{y}} \frac{\exp \left\{ \sum_{i=1}^n \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, i, y_{i-1}, y_i) \right\}}{Z(\mathbf{x}; \mathbf{w})} \\ &= \operatorname{amax}_{\mathbf{y}} \exp \left\{ \sum_{i=1}^n \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, i, y_{i-1}, y_i) \right\} \\ &= \operatorname{amax}_{\mathbf{y}} \sum_{i=1}^n \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, i, y_{i-1}, y_i) \end{aligned}$$

- We can use the Viterbi algorithm

Making Predictions with Log-Linear Models

- For tractability, assume $\mathbf{f}(\mathbf{x}, \mathbf{y})$ decomposes into bigrams:

$$\mathbf{f}(\mathbf{x}_{1:n}, \mathbf{y}_{1:n}) = \sum_{i=1}^n \mathbf{f}(\mathbf{x}, i, y_{i-1}, y_i)$$

- Given \mathbf{w} , given $\mathbf{x}_{1:n}$, find:

$$\begin{aligned} \operatorname{argmax}_{\mathbf{y}_{1:n}} \Pr(\mathbf{y}_{1:n} | \mathbf{x}_{1:n}; \mathbf{w}) &= \operatorname{amax}_{\mathbf{y}} \frac{\exp \left\{ \sum_{i=1}^n \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, i, y_{i-1}, y_i) \right\}}{Z(\mathbf{x}; \mathbf{w})} \\ &= \operatorname{amax}_{\mathbf{y}} \exp \left\{ \sum_{i=1}^n \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, i, y_{i-1}, y_i) \right\} \\ &= \operatorname{amax}_{\mathbf{y}} \sum_{i=1}^n \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, i, y_{i-1}, y_i) \end{aligned}$$

- We can use the Viterbi algorithm

Parameter Estimation in Log-Linear Models

$$\Pr(\mathbf{y} \mid \mathbf{x}; \mathbf{w}) = \frac{\exp \{ \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, \mathbf{y}) \}}{Z(\mathbf{x}; \mathbf{w})}$$

How to estimate \mathbf{w} given training data?

Two approaches:

- ▶ MEMMs: assume that $\Pr(\mathbf{y} \mid \mathbf{x}; \mathbf{w})$ decomposes
- ▶ CRFs: assume that $\mathbf{f}(\mathbf{x}, \mathbf{y})$ decomposes

Parameter Estimation in Log-Linear Models

$$\Pr(\mathbf{y} \mid \mathbf{x}; \mathbf{w}) = \frac{\exp \{ \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, \mathbf{y}) \}}{Z(\mathbf{x}; \mathbf{w})}$$

How to estimate \mathbf{w} given training data?

Two approaches:

- ▶ MEMMs: assume that $\Pr(\mathbf{y} \mid \mathbf{x}; \mathbf{w})$ decomposes
- ▶ CRFs: assume that $\mathbf{f}(\mathbf{x}, \mathbf{y})$ decomposes

Maximum Entropy Markov Models (MEMMs)

(McCallum, Freitag, Pereira '00)

- ▶ Similarly to HMMs:

$$\begin{aligned}\Pr(\mathbf{y}_{1:n} \mid \mathbf{x}_{1:n}) &= \Pr(y_1 \mid \mathbf{x}_{1:n}) \times \Pr(\mathbf{y}_{2:n} \mid \mathbf{x}_{1:n}, y_1) \\ &= \Pr(y_1 \mid \mathbf{x}_{1:n}) \times \prod_{i=2}^n \Pr(y_i \mid \mathbf{x}_{1:n}, \mathbf{y}_{1:i-1}) \\ &= \Pr(y_1 \mid \mathbf{x}_{1:n}) \times \prod_{i=2}^n \Pr(y_i \mid \mathbf{x}_{1:n}, y_{i-1})\end{aligned}$$

- ▶ Assumption under MEMMs:

$$\Pr(y_i \mid \mathbf{x}_{1:n}, \mathbf{y}_{1:i-1}) = \Pr(y_i \mid \mathbf{x}_{1:n}, y_{i-1})$$

Parameter Estimation in MEMMs

- ▶ Decompose sequential problem:

$$\Pr(y_{1:n} \mid \mathbf{x}_{1:n}) = \Pr(y_1 \mid \mathbf{x}_{1:n}) \times \prod_{i=2}^n \Pr(y_i \mid \mathbf{x}_{1:n}, i, y_{i-1})$$

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

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

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
- ▶ Sequence learning reduced to multi-class logistic regression

Conditional Random Fields

(Lafferty, McCallum, Pereira 2001)

- ▶ Log-linear model of the conditional distribution:

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

where

- ▶ $\mathbf{x} = x_1 x_2 \dots x_n \in \mathcal{X}^*$
 - ▶ $\mathbf{y} = y_1 y_2 \dots y_n \in \mathcal{Y}^*$ and $\mathcal{Y} = \{1, \dots, L\}$
 - ▶ $\mathbf{f}(\mathbf{x}, \mathbf{y})$ is a feature vector of \mathbf{x} and \mathbf{y}
 - ▶ \mathbf{w} are model parameters
- ▶ To predict the best sequence

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

- ▶ Assumption in CRF (for tractability):
 $\mathbf{f}(\mathbf{x}, \mathbf{y})$ decomposes into factors

CRFs as Factored Log-Linear Models

- For tractability, $\mathbf{f}(\mathbf{x}, \mathbf{y})$ needs to decompose.
For **bigram** factorizations:

$$\mathbf{f}(\mathbf{x}_{1:n}, \mathbf{y}_{1:n}) = \sum_{i=1}^n \mathbf{f}(\mathbf{x}, i, y_{i-1}, y_i)$$

- The model form is:

$$\begin{aligned} \Pr(\mathbf{y}|\mathbf{x}_{1:n}; \mathbf{w}) &= \frac{\exp \{ \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, \mathbf{y}) \}}{Z(\mathbf{x}, \mathbf{w})} \\ &= \frac{\exp \{ \sum_{i=1}^n \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, i, y_{i-1}, y_i) \}}{Z(\mathbf{x}, \mathbf{w})} \end{aligned}$$

where

$$Z(\mathbf{x}_{1:n}, \mathbf{w}) = \sum_{\mathbf{z} \in \mathcal{Y}^n} \exp \left\{ \sum_{i=1}^n \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, i, z_{i-1}, z_i) \right\}$$

Parameter Estimation in CRFs

- ▶ Given a training set

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

estimate \mathbf{w}

- ▶ Define the conditional log-likelihood of the data:

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

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

Learning the Parameters of a CRF

- ▶ We pose it as a convex optimization problem
- ▶ Find:

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

where

- ▶ The first term is the log-likelihood of the data
- ▶ The second term is a regularization term, it penalizes solutions with large norm (similar to norm-minimization in SVM)
- ▶ λ is a parameter to control the trade-off between fitting the data and model complexity

Learning the Parameters of a CRF

- Find

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

- In general there is no analytical solution to this optimization
- We use iterative techniques, i.e. gradient-based optimization
 1. Initialize $\mathbf{w} = \mathbf{0}$
 2. Take derivatives of $L(\mathbf{w}) - \frac{\lambda}{2} \|\mathbf{w}\|^2$, compute gradient
 3. Move \mathbf{w} in steps proportional to the gradient
 4. Repeat steps 2 and 3 until convergence
- Fast and scalable algorithms exist

Computing the Gradient in CRFs

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

where

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

- ▶ First term: observed mean feature value
- ▶ Second term: expected feature value under current \mathbf{w}
- ▶ In the optimal, observed = expected

Computing the Gradient in CRFs

- ▶ The first term is easy to compute, by counting explicitly

$$\frac{1}{m} \sum_{k=1}^m \sum_i \mathbf{f}_j(\mathbf{x}, i, y_{i-1}^{(k)}, y_i^{(k)})$$

- ▶ The second term is more involved,

$$\sum_{k=1}^m \sum_{\mathbf{y} \in \mathcal{Y}^*} \Pr(\mathbf{y} | \mathbf{x}^{(k)}; \mathbf{w}) \sum_i \mathbf{f}_j(\mathbf{x}^{(k)}, i, y_{i-1}, y_i)$$

because it sums over all sequences $\mathbf{y} \in \mathcal{Y}^*$

Computing the Gradient in CRFs

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

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

where

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

- 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 $O(n|\mathcal{Y}|^2)$

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

- ▶ **Definition:** forward and backward quantities

$$\alpha_i(a) = \sum_{\mathbf{y}_{1:i} \in \mathcal{Y}^i: y_i=a} \exp \left\{ \sum_{j=1}^i \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, j, y_{j-1}, y_j) \right\}$$

$$\beta_i(b) = \sum_{\mathbf{y}_{i:n} \in \mathcal{Y}^{(n-i+1)}: y_i=b} \exp \left\{ \sum_{j=i+1}^n \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, j, y_{j-1}, y_j) \right\}$$

- ▶ $Z = \sum_a \alpha_n(a)$
- ▶ $\mu_i(a, b) = \{\alpha_{i-1}(a) * \exp\{\mathbf{w} \cdot \mathbf{f}(\mathbf{x}, i, a, b)\} * \beta_i(b) * Z^{-1}\}$
- ▶ Similarly to Viterbi, $\alpha_i(a)$ and $\beta_i(b)$ can be computed efficiently in a recursive manner

Forward-Backward for CRFs

- ▶ Assume fixed \mathbf{x} . Calculate in $O(n|\mathcal{Y}|^2)$

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

- ▶ **Definition:** forward and backward quantities

$$\alpha_i(a) = \sum_{\mathbf{y}_{1:i} \in \mathcal{Y}^i: y_i=a} \exp \left\{ \sum_{j=1}^i \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, j, y_{j-1}, y_j) \right\}$$

$$\beta_i(b) = \sum_{\mathbf{y}_{i:n} \in \mathcal{Y}^{(n-i+1)}: y_i=b} \exp \left\{ \sum_{j=i+1}^n \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, j, y_{j-1}, y_j) \right\}$$

- ▶ $Z = \sum_a \alpha_n(a)$
- ▶ $\mu_i(a, b) = \{\alpha_{i-1}(a) * \exp\{\mathbf{w} \cdot \mathbf{f}(\mathbf{x}, i, a, b)\} * \beta_i(b) * Z^{-1}\}$
- ▶ Similarly to Viterbi, $\alpha_i(a)$ and $\beta_i(b)$ can be computed efficiently in a recursive manner

CRFs: summary so far

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

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

- ▶ Prediction: 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, $\Pr(\mathbf{y}|\mathbf{x}; \mathbf{w})$
- ▶ Computations factorize on label bigrams
- ▶ Model form:

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

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

MEMMs and CRFs

$$\text{MEMMs: } \Pr(\mathbf{y} \mid \mathbf{x}) = \prod_{i=1}^n \frac{\exp \{ \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, i, y_{i-1}, y_i) \}}{Z(\mathbf{x}, i, y_{i-1}; \mathbf{w})}$$

$$\text{CRFs: } \Pr(\mathbf{y} \mid \mathbf{x}) = \frac{\exp \{ \sum_{i=1}^n \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, i, y_{i-1}, y_i) \}}{Z(\mathbf{x})}$$

- ▶ MEMMs locally normalized; CRFs globally normalized
- ▶ MEMM assume that $\Pr(y_i \mid x_{1:n}, y_{1:i-1}) = \Pr(y_i \mid x_{1:n}, y_{i-1})$
- ▶ Both exploit the same factorization, i.e. same features
- ▶ Same computations to compute $\operatorname{argmax}_{\mathbf{y}} \Pr(\mathbf{y} \mid \mathbf{x})$
- ▶ MEMMs are cheaper to train
- ▶ CRFs are easier to extend to other structures (next lecture)

HMMs for sequence prediction

- ▶ \mathbf{x} are the observations, \mathbf{y} are the hidden states
- ▶ HMMs model the joint distribution $\Pr(\mathbf{x}, \mathbf{y})$
- ▶ Parameters: (assume $\mathcal{X} = \{1, \dots, k\}$ and $\mathcal{Y} = \{1, \dots, l\}$)
 - ▶ $\pi \in \mathbb{R}^l$, $\pi_a = \Pr(y_1 = a)$
 - ▶ $T \in \mathbb{R}^{l \times l}$, $T_{a,b} = \Pr(y_i = b | y_{i-1} = a)$
 - ▶ $O \in \mathbb{R}^{l \times k}$, $O_{a,c} = \Pr(x_i = c | y_i = a)$
- ▶ Model form

$$\Pr(\mathbf{x}, \mathbf{y}) = \pi_{y_1} O_{y_1, x_1} \prod_{i=2}^n T_{y_{i-1}, y_i} O_{y_i, x_i}$$

- ▶ Parameter Estimation: maximum likelihood by counting events and normalizing

HMMs and CRFs

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

► In HMMs:

$$\begin{aligned}\hat{\mathbf{y}} &= \text{amax}_{\mathbf{y}} \pi_{y_1} O_{y_1, x_1} \prod_{i=2}^n T_{y_{i-1}, y_i} O_{y_i, x_i} \\ &= \text{amax}_{\mathbf{y}} \log(\pi_{y_1} O_{y_1, x_1}) + \sum_{i=2}^n \log(T_{y_{i-1}, y_i} O_{y_i, x_i})\end{aligned}$$

► An HMM can be “ported” into a CRF by setting:

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

► 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 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 x and y .
 - ▶ CRFs focus on the mapping from x to y .
 - ▶ A priori, it is hard to say which paradigm is better.
 - ▶ Same dilemma as Naive Bayes vs. Maximum Entropy.

Structured Prediction

Perceptron, SVMs, CRFs

Learning Structured Predictors

- ▶ Goal: given training data

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

learn a predictor $\mathbf{x} \rightarrow \mathbf{y}$ with small error on unseen inputs

- ▶ In a CRF:

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

- ▶ To predict new values, $Z(\mathbf{x}; \mathbf{w})$ is not relevant
 - ▶ Parameter estimation: \mathbf{w} is set to maximize likelihood
-
- ▶ Can we learn \mathbf{w} more directly, focusing on errors?

Learning Structured Predictors

- ▶ Goal: given training data

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

learn a predictor $\mathbf{x} \rightarrow \mathbf{y}$ with small error on unseen inputs

- ▶ In a CRF:

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

- ▶ To predict new values, $Z(\mathbf{x}; \mathbf{w})$ is not relevant
 - ▶ Parameter estimation: \mathbf{w} is set to maximize likelihood
- ▶ Can we learn \mathbf{w} more directly, focusing on errors?

The Structured Perceptron

(Collins, 2002)

- ▶ Set $\mathbf{w} = \mathbf{0}$
- ▶ For $t = 1 \dots T$
 - ▶ For each training example (\mathbf{x}, \mathbf{y})
 1. Compute $\mathbf{z} = \operatorname{argmax}_{\mathbf{z}} \sum_{i=1}^n \mathbf{f}(\mathbf{x}, i, z_{i-1}, z_i)$
 2. If $\mathbf{z} \neq \mathbf{y}$

$$\mathbf{w} \leftarrow \mathbf{w} + \sum_i \mathbf{f}(\mathbf{x}, i, y_{i-1}, y_i) - \sum_i \mathbf{f}(\mathbf{x}, i, z_{i-1}, \mathbf{z}_i)$$

- ▶ Return \mathbf{w}

The Structured Perceptron + Averaging

(Freund and Schapire, 1998) (Collins 2002)

- ▶ Set $\mathbf{w} = \mathbf{0}$, $\mathbf{w}^a = \mathbf{0}$
- ▶ For $t = 1 \dots T$
 - ▶ For each training example (\mathbf{x}, \mathbf{y})
 1. Compute $\mathbf{z} = \operatorname{argmax}_{\mathbf{z}} \sum_{i=1}^n \mathbf{f}(\mathbf{x}, i, z_{i-1}, z_i)$
 2. If $\mathbf{z} \neq \mathbf{y}$

$$\mathbf{w} \leftarrow \mathbf{w} + \sum_i \mathbf{f}(\mathbf{x}, i, y_{i-1}, y_i) - \sum_i \mathbf{f}(\mathbf{x}, i, z_{i-1}, \mathbf{z}_i)$$

3. $\mathbf{w}^a = \mathbf{w}^a + \mathbf{w}$

- ▶ Return \mathbf{w}^a / mT , where m is the number of training examples

Perceptron Updates: Example

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

- ▶ Let **y** be the correct output for **x**.
- ▶ Say we predict **z** instead, under our current **w**
- ▶ The update is:

$$\begin{aligned}\mathbf{g} &= \mathbf{f}(\mathbf{x}, \mathbf{y}) - \mathbf{f}(\mathbf{x}, \mathbf{z}) \\ &= \sum_i \mathbf{f}(\mathbf{x}, i, y_{i-1}, y_i) - \sum_i \mathbf{f}(\mathbf{x}, i, z_{i-1}, z_i) \\ &= \mathbf{f}(\mathbf{x}, 2, \text{PER}, \text{PER}) - \mathbf{f}(\mathbf{x}, 2, \text{PER}, \text{LOC}) \\ &\quad + \mathbf{f}(\mathbf{x}, 3, \text{PER}, -) - \mathbf{f}(\mathbf{x}, 3, \text{LOC}, -)\end{aligned}$$

- ▶ Perceptron updates are typically **very sparse**

Properties of the Perceptron

- ▶ Online algorithm. Often much more efficient than “batch” algorithms
- ▶ If the data is separable, it will converge to parameter values with 0 errors
- ▶ Number of errors before convergence is related to a definition of *margin*. Can also relate margin to generalization properties
- ▶ In practice:
 1. Averaging improves performance a lot
 2. Typically reaches a good solution after only a few (say 5) iterations over the training set
 3. Often performs nearly as well as CRFs, or SVMs

Averaged Perceptron Convergence

Iteration	Accuracy
1	90.79
2	91.20
3	91.32
4	91.47
5	91.58
6	91.78
7	91.76
8	91.82
9	91.88
10	91.91
11	91.92
12	91.96
...	

(results on validation set for a parsing task)

Margin-based Structured Prediction

- ▶ Let $\mathbf{f}(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^n \mathbf{f}(\mathbf{x}, i, y_{i-1}, y_i)$
- ▶ Model: $\operatorname{argmax}_{\mathbf{y} \in \mathcal{Y}^*} \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, \mathbf{y})$
- ▶ Consider an example $(\mathbf{x}^{(k)}, \mathbf{y}^{(k)})$:
 $\exists \mathbf{y} \neq \mathbf{y}^{(k)} : \mathbf{w} \cdot \mathbf{f}(\mathbf{x}^{(k)}, \mathbf{y}^{(k)}) < \mathbf{w} \cdot \mathbf{f}(\mathbf{x}^{(k)}, \mathbf{y}) \implies \text{error}$
- ▶ Let $\mathbf{y}' = \operatorname{argmax}_{\mathbf{y} \in \mathcal{Y}^* : \mathbf{y} \neq \mathbf{y}^{(k)}} \mathbf{w} \cdot \mathbf{f}(\mathbf{x}^{(k)}, \mathbf{y})$
Define $\gamma_k = \mathbf{w} \cdot (\mathbf{f}(\mathbf{x}^{(k)}, \mathbf{y}^{(k)}) - \mathbf{f}(\mathbf{x}^{(k)}, \mathbf{y}'))$
- ▶ The quantity γ_k is a notion of **margin** on example k :
 $\gamma_k > 0 \iff$ no mistakes in the example
high $\gamma_k \iff$ high confidence

Margin-based Structured Prediction

- ▶ Let $\mathbf{f}(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^n \mathbf{f}(\mathbf{x}, i, y_{i-1}, y_i)$
- ▶ Model: $\operatorname{argmax}_{\mathbf{y} \in \mathcal{Y}^*} \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, \mathbf{y})$
- ▶ Consider an example $(\mathbf{x}^{(k)}, \mathbf{y}^{(k)})$:
 $\exists \mathbf{y} \neq \mathbf{y}^{(k)} : \mathbf{w} \cdot \mathbf{f}(\mathbf{x}^{(k)}, \mathbf{y}^{(k)}) < \mathbf{w} \cdot \mathbf{f}(\mathbf{x}^{(k)}, \mathbf{y}) \implies \text{error}$
- ▶ Let $\mathbf{y}' = \operatorname{argmax}_{\mathbf{y} \in \mathcal{Y}^* : \mathbf{y} \neq \mathbf{y}^{(k)}} \mathbf{w} \cdot \mathbf{f}(\mathbf{x}^{(k)}, \mathbf{y})$
Define $\gamma_k = \mathbf{w} \cdot (\mathbf{f}(\mathbf{x}^{(k)}, \mathbf{y}^{(k)}) - \mathbf{f}(\mathbf{x}^{(k)}, \mathbf{y}'))$
- ▶ The quantity γ_k is a notion of **margin** on example k :
 $\gamma_k > 0 \iff$ no mistakes in the example
high $\gamma_k \iff$ high confidence

Margin-based Structured Prediction

- ▶ Let $\mathbf{f}(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^n \mathbf{f}(\mathbf{x}, i, y_{i-1}, y_i)$
- ▶ Model: $\operatorname{argmax}_{\mathbf{y} \in \mathcal{Y}^*} \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, \mathbf{y})$
- ▶ Consider an example $(\mathbf{x}^{(k)}, \mathbf{y}^{(k)})$:
 $\exists \mathbf{y} \neq \mathbf{y}^{(k)} : \mathbf{w} \cdot \mathbf{f}(\mathbf{x}^{(k)}, \mathbf{y}^{(k)}) < \mathbf{w} \cdot \mathbf{f}(\mathbf{x}^{(k)}, \mathbf{y}) \implies \text{error}$
- ▶ Let $\mathbf{y}' = \operatorname{argmax}_{\mathbf{y} \in \mathcal{Y}^* : \mathbf{y} \neq \mathbf{y}^{(k)}} \mathbf{w} \cdot \mathbf{f}(\mathbf{x}^{(k)}, \mathbf{y})$
Define $\gamma_k = \mathbf{w} \cdot (\mathbf{f}(\mathbf{x}^{(k)}, \mathbf{y}^{(k)}) - \mathbf{f}(\mathbf{x}^{(k)}, \mathbf{y}'))$
- ▶ The quantity γ_k is a notion of **margin** on example k :
 $\gamma_k > 0 \iff$ no mistakes in the example
high $\gamma_k \iff$ high confidence

Mistake-augmented Margins

(Taskar et al, 2004)

$\mathbf{x}^{(k)}$	Jack	London	went	to	Paris
$\mathbf{y}^{(k)}$	PER	PER	-	-	LOC
\mathbf{y}'	PER	LOC	-	-	LOC
\mathbf{y}''	PER	-	-	-	-
\mathbf{y}'''	-	-	PER	PER	-

- ▶ Def: $e(\mathbf{y}, \mathbf{y}') = \sum_{i=1}^n [y_i \neq y'_i]$
e.g., $e(\mathbf{y}^{(k)}, \mathbf{y}^{(k)}) = 0$, $e(\mathbf{y}^{(k)}, \mathbf{y}') = 1$, $e(\mathbf{y}^{(k)}, \mathbf{y}''') = 5$

- ▶ We want a \mathbf{w} such that

$$\forall \mathbf{y} \neq \mathbf{y}^{(k)} : \mathbf{w} \cdot \mathbf{f}(\mathbf{x}^{(k)}, \mathbf{y}^{(k)}) > \mathbf{w} \cdot \mathbf{f}(\mathbf{x}^{(k)}, \mathbf{y}) + e(\mathbf{y}^{(k)}, \mathbf{y})$$

(the higher the error of \mathbf{y} , the larger the separation should be)

Structured Hinge Loss

- ▶ Define a mistake-augmented margin

$$\gamma_{k,\mathbf{y}} = \mathbf{w} \cdot \mathbf{f}(\mathbf{x}^{(k)}, \mathbf{y}^{(k)}) - \mathbf{w} \cdot \mathbf{f}(\mathbf{x}^{(k)}, \mathbf{y}) - e(\mathbf{y}^{(k)}, \mathbf{y})$$
$$\gamma_k = \min_{\mathbf{y} \neq \mathbf{y}^{(k)}} \gamma_{k,\mathbf{y}}$$

- ▶ Define loss function on example k as:

$$L(\mathbf{w}, \mathbf{x}^{(k)}, \mathbf{y}^{(k)}) = \max_{\mathbf{y} \in \mathcal{Y}^*} \left\{ e(\mathbf{y}^{(k)}, \mathbf{y}) - \mathbf{w} \cdot \mathbf{f}(\mathbf{x}^{(k)}, \mathbf{y}^{(k)}) - \mathbf{w} \cdot \mathbf{f}(\mathbf{x}^{(k)}, \mathbf{y}) \right\}$$

- ▶ Leads to an SVM for structured prediction
- ▶ Given a training set, find:

$$\operatorname{argmin}_{\mathbf{w} \in \mathbb{R}^D} \sum_{k=1}^m L(\mathbf{w}, \mathbf{x}^{(k)}, \mathbf{y}^{(k)}) + \frac{\lambda}{2} \|\mathbf{w}\|^2$$

Regularized Loss Minimization

- ▶ Given a training set $\{(\mathbf{x}^{(1)}, \mathbf{y}^{(1)}), \dots, (\mathbf{x}^{(m)}, \mathbf{y}^{(m)})\}$.

Find:

$$\operatorname{argmin}_{\mathbf{w} \in \mathbb{R}^D} \sum_{k=1}^m L(\mathbf{w}, \mathbf{x}^{(k)}, \mathbf{y}^{(k)}) + \frac{\lambda}{2} \|\mathbf{w}\|^2$$

- ▶ Two common loss functions $L(\mathbf{w}, \mathbf{x}^{(k)}, \mathbf{y}^{(k)})$:
 - ▶ Log-likelihood loss (CRFs)

$$-\log P(\mathbf{y}^{(k)} \mid \mathbf{x}^{(k)}; \mathbf{w})$$

- ▶ Hinge loss (SVMs)

$$\max_{\mathbf{y} \in \mathcal{Y}^*} \left(e(\mathbf{y}^{(k)}, \mathbf{y}) - \mathbf{w} \cdot (\mathbf{f}(\mathbf{x}^{(k)}, \mathbf{y}^{(k)}) - \mathbf{f}(\mathbf{x}^{(k)}, \mathbf{y})) \right)$$

Learning Structure Predictors: summary so far

- ▶ Linear models for sequence prediction

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

- ▶ Computations factorize on label bigrams
 - ▶ Decoding: using Viterbi
 - ▶ Marginals: using forward-backward
- ▶ Parameter estimation:
 - ▶ Perceptron, Log-likelihood, SVMs
 - ▶ Extensions from classification to the structured case
 - ▶ Optimization methods:
 - ▶ Stochastic (sub)gradient methods (LeCun et al 98) (Shalev-Shwartz et al. 07)
 - ▶ Exponentiated Gradient (Collins et al 08)
 - ▶ SVM Struct (Tsochantaridis et al. 04)
 - ▶ Structured MIRA (McDonald et al 05)

Generic Structure Prediction

Sequence Prediction, Beyond Bigrams

- ▶ It is easy to extend the scope of features to k -grams

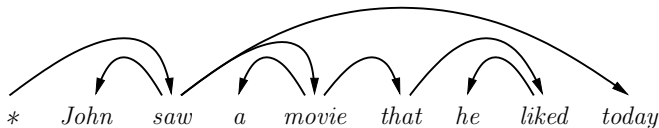
$$\mathbf{f}(\mathbf{x}, i, y_{i-k+1:i-1}, y_i)$$

- ▶ In general, think of state σ_i remembering relevant history
 - ▶ $\sigma_i = y_{i-1}$ for bigrams
 - ▶ $\sigma_i = y_{i-k+1:i-1}$ for k -grams
 - ▶ σ_i can be the state at time i of a deterministic automaton generating \mathbf{y}
- ▶ The structured predictor is

$$\operatorname{argmax}_{\mathbf{y} \in \mathcal{Y}^*} \sum_i \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, i, \sigma_i, y_i)$$

- ▶ Viterbi and forward-backward extend naturally, in $O(nL^k)$

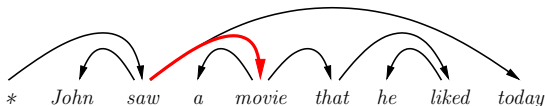
Dependency Structures



- ▶ Directed arcs represent **dependencies** between a **head word** and a **modifier word**.
- ▶ E.g.:
 - movie *modifies* saw,
 - John *modifies* saw,
 - today *modifies* saw

Dependency Parsing: arc-factored models

(McDonald et al. 2005)



- Parse trees decompose into single dependencies $\langle h, m \rangle$

$$\operatorname{argmax}_{\mathbf{y} \in \mathcal{Y}(\mathbf{x})} \sum_{\langle h, m \rangle \in \mathbf{y}} \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, h, m)$$

- Some features: $\mathbf{f}_1(\mathbf{x}, h, m) = [\text{"saw"} \rightarrow \text{"movie"}]$
 $\mathbf{f}_2(\mathbf{x}, h, m) = [\text{distance} = +2]$
- Tractable inference algorithms exist (tomorrow's lecture)

Linear Structured Prediction

- ▶ Sequence prediction (bigram factorization)

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

- ▶ Dependency parsing (arc-factored)

$$\operatorname{argmax}_{\mathbf{y} \in \mathcal{Y}(\mathbf{x})} \sum_{\langle h, m \rangle \in \mathbf{y}} \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, h, m)$$

- ▶ In general, we can enumerate parts $r \in \mathbf{y}$

$$\operatorname{argmax}_{\mathbf{y} \in \mathcal{Y}(\mathbf{x})} \sum_{r \in \mathbf{y}} \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, r)$$

Linear Structured Prediction Framework

- ▶ Abstract models of structures
 - ▶ Input domain \mathcal{X} , output domain \mathcal{Y}
 - ▶ A choice of factorization, $r \in \mathbf{y}$
 - ▶ Features: $\mathbf{f}(\mathbf{x}, r) \rightarrow \mathbb{R}^d$
- ▶ The linear prediction model, with $\mathbf{w} \in \mathbb{R}^d$

$$\operatorname{argmax}_{\mathbf{y} \in \mathcal{Y}(\mathbf{x})} \sum_{r \in \mathbf{y}} \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, r)$$

- ▶ Generic algorithms for Perceptron, CRF, SVM
 - ▶ Require tractable inference algorithms