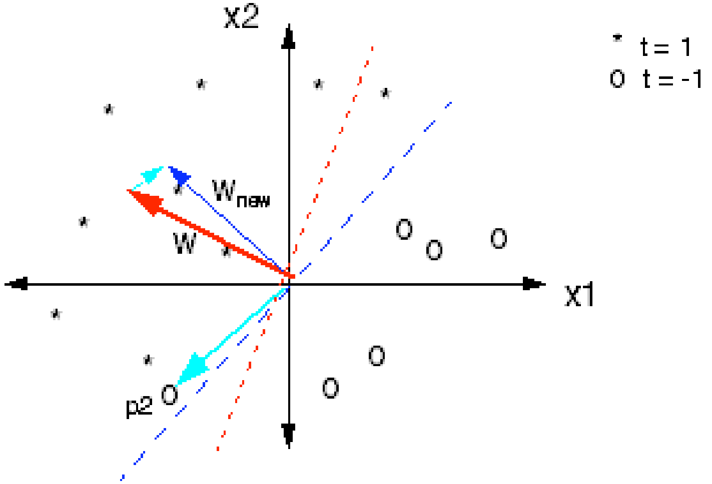
Parameter Estimation:

The Perceptron Algorithm



What we have done so far...

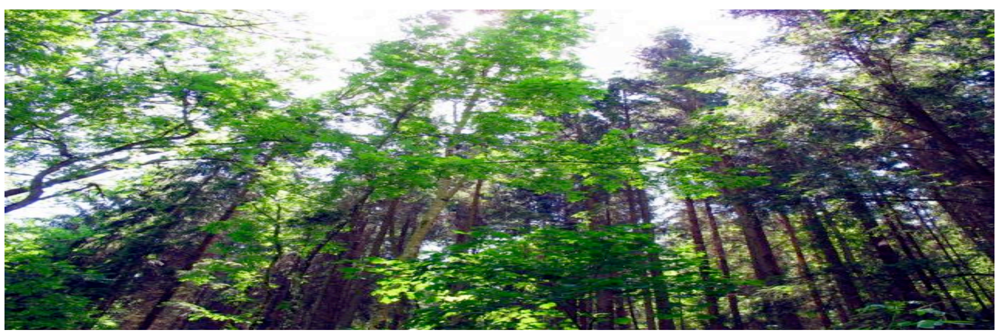
- packed forest
  - as a general representation for many NLP problems
  - formalized as a weighted hypergraph
  - DP algorithms for 1-best and *k*-best on hypergraphs

Baoweier yu Shalong juxing le huitan

Powell held a meeting w/ Sharon

What we have done so far...

- packed forest
  - as a general representation for many NLP problems
  - formalized as a weighted hypergraph
  - DP algorithms for 1-best and *k*-best on hypergraphs



Baoweier yu Shalong juxing le huitan　　Powell held a meeting w/ Sharon

Big Q: where do the *weights* come from?

A Quiz

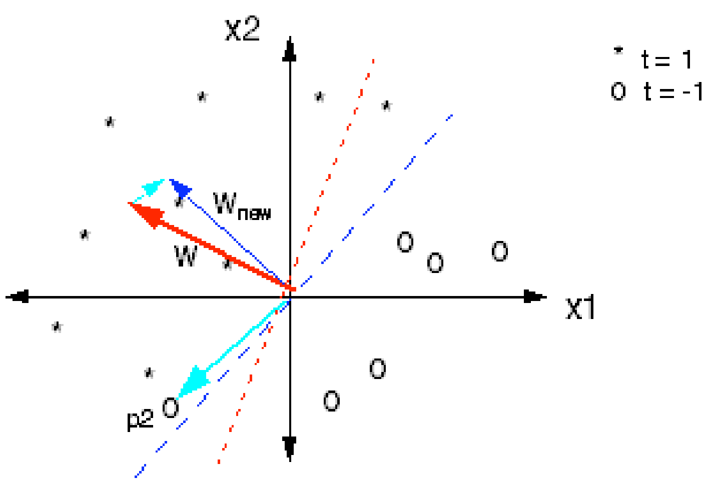
Perceptron is ...

- an extremely simple algorithm
- almost universally applicable
- and works very well in practice

Perceptron is ...

- an extremely simple algorithm
- almost universally applicable
- and works very well in practice

vanilla perceptron
(Rosenblatt, 1958)

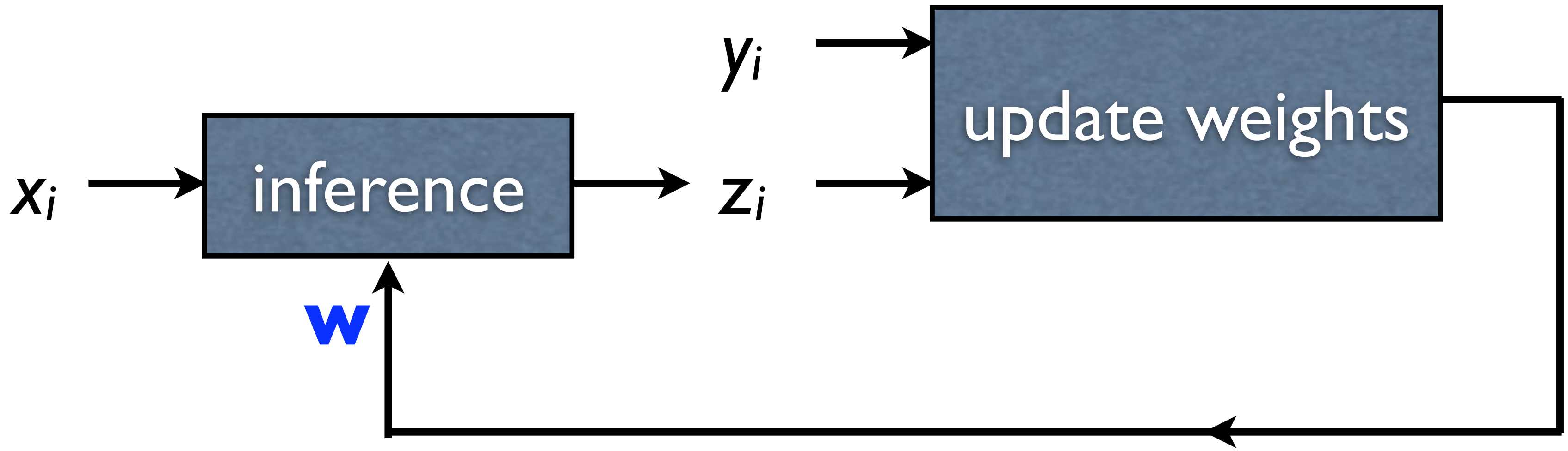


structured perceptron
(Collins, 2002)

the	man	bit	the	dog
DT	NN	VBD	DT	NN

Generic Perceptron

- online-learning: one example at a time
- learning by doing
  - find the best output under the current weights
  - update weights at mistakes



Example: POS Tagging

- gold-standard: DT NN VBD DT NN
  - the man bit the dog

- current output: DT NN NN DT NN
  - the man bit the dog

- assume only two feature classes
  - tag bigrams t_{i-1} t_i
  - word/tag pairs w_i

- weights ++: (NN,VBD) (VBD, DT) (VBD → bit)
- weights −−: (NN, NN) (NN, DT) (NN → bit)

Example: POS Tagging

gold-standard:	DT	NN	VBD	DT	NN	y	$\Phi(x, y)$
	the	man	bit	the	dog	x	

current output:	DT	NN	NN	DT	NN	z	$\Phi(x, z)$
	the	man	bit	the	dog	x	

- assume only two feature classes
  - tag bigrams t_{i-1} t_i
  - word/tag pairs w_i
- weights ++: (NN,VBD) (VBD, DT) (VBD $\rightarrow$ bit)
- weights −−: (NN, NN) (NN, DT) (NN $\rightarrow$ bit)

Example: POS Tagging

- gold-standard: DT NN VBD DT NN y
  - the man bit the dog x

$\Phi(x, y)$

- current output: DT NN NN DT NN z
  - the man bit the dog x

$\Phi(x, z)$

- assume only two feature classes
  - tag bigrams t_{i-1} t_i
  - word/tag pairs w_i
- weights ++: (NN, VBD) (VBD, DT) (VBD $\rightarrow$ bit)
- weights −−: (NN, NN) (NN, DT) (NN $\rightarrow$ bit)

Example: POS Tagging

- gold-standard: DT NN VBD DT NN y
  - the man bit the dog x

$\Phi(x, y)$

- current output: DT NN NN DT NN z
  - the man bit the dog x

$\Phi(x, z)$

- assume only two feature classes
  - tag bigrams t_{i-1} t_i
  - word/tag pairs w_i
- weights ++: (NN, VBD) (VBD, DT) (VBD $\rightarrow$ bit)
- weights −−: (NN, NN) (NN, DT) (NN $\rightarrow$ bit)

Example: POS Tagging

- gold-standard: DT NN VBD DT NN y
  - the man bit the dog x

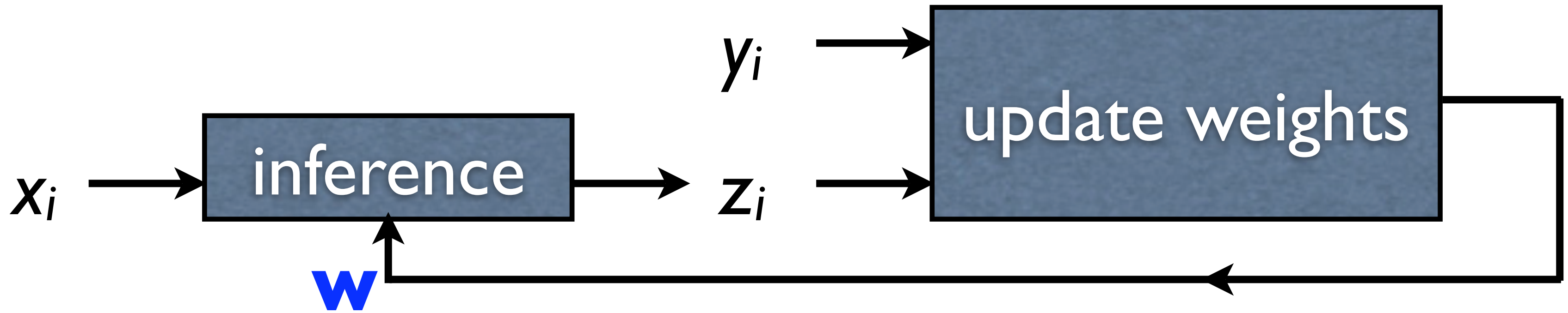
$\Phi(x, y)$

- current output: DT NN NN DT NN z
  - the man bit the dog x

$\Phi(x, z)$

- assume only two feature classes
  - tag bigrams t_{i-1} t_i
  - word/tag pairs w_i
- weights ++: (NN, VBD) (VBD, DT) (VBD $\rightarrow$ bit)
- weights −−: (NN, NN) (NN, DT) (NN $\rightarrow$ bit)

Structured Perceptron



Inputs: Training set (x_i, y_i) for $i = 1 \ldots n$

Initialization: $\mathbf{W} = 0$

Define: $F(x) = \boxed{\mathrm{argmax}_{y \in \mathbf{GEN}(x)}} \mathbf{\Phi}(x, y) \cdot \mathbf{W}$

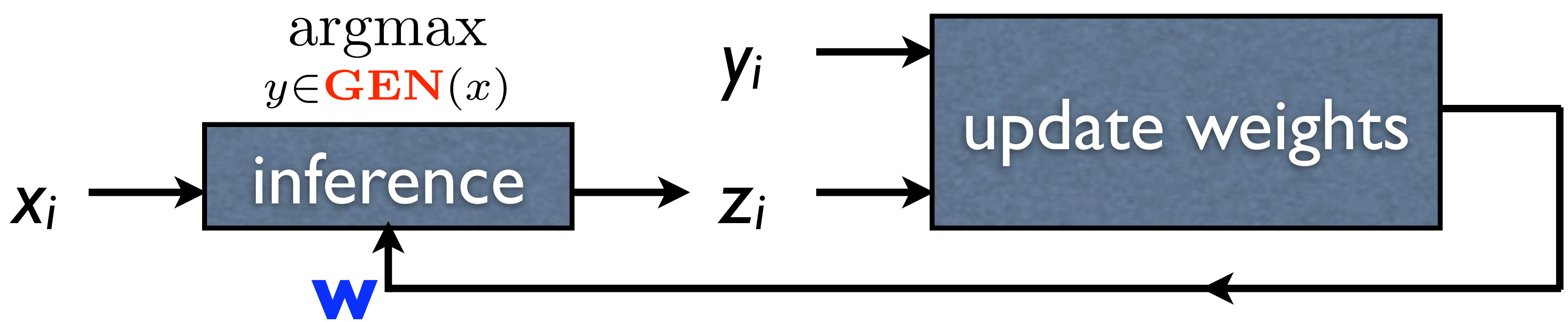
Algorithm: For $t = 1 \ldots T, i = 1 \ldots n$

$z_i = F(x_i)$

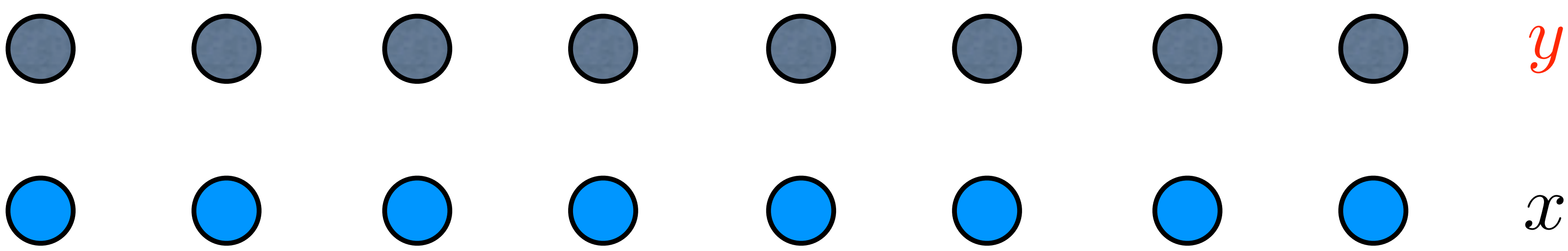
If $(z_i \neq y_i)$ $\boxed{\mathbf{W} \leftarrow \mathbf{W} + \mathbf{\Phi}(x_i, y_i) - \mathbf{\Phi}(x_i, z_i)}$

Output: Parameters $\mathbf{W}$

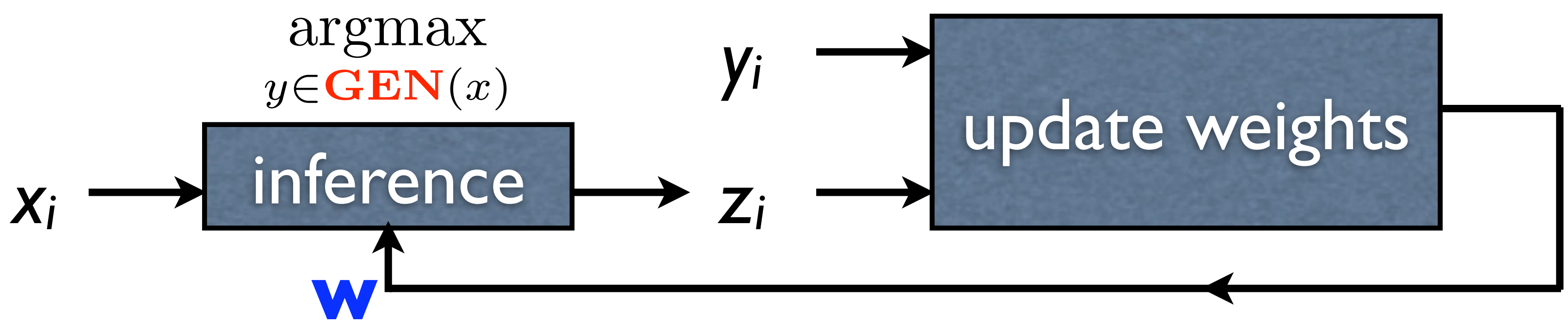
Efficiency vs. Expressiveness



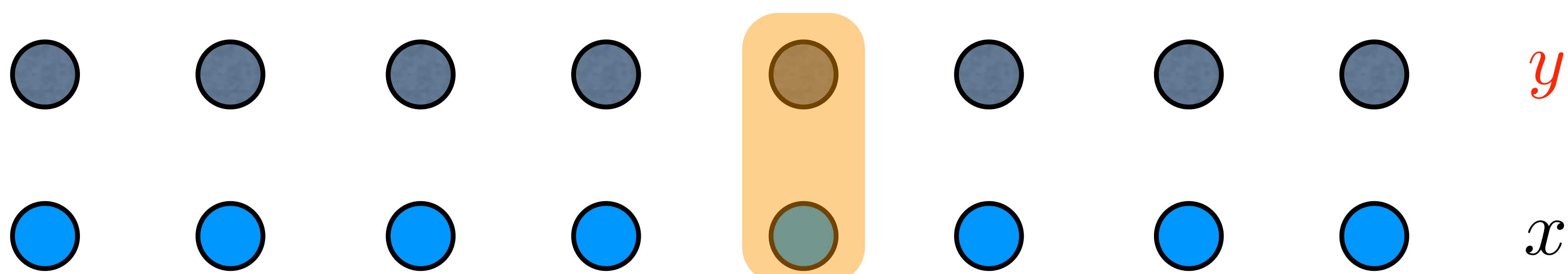
- the inference (argmax) must be efficient
  - either the search space GEN(x) is small, or factored
  - features must be local to y (but can be global to x)
    - e.g. bigram tagger, but look at all input words (cf. CRFs)



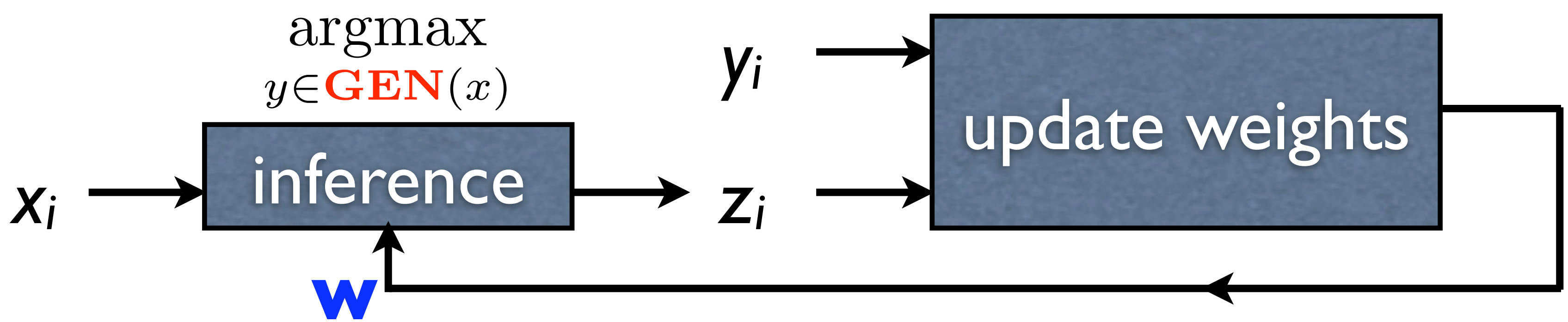
Efficiency vs. Expressiveness



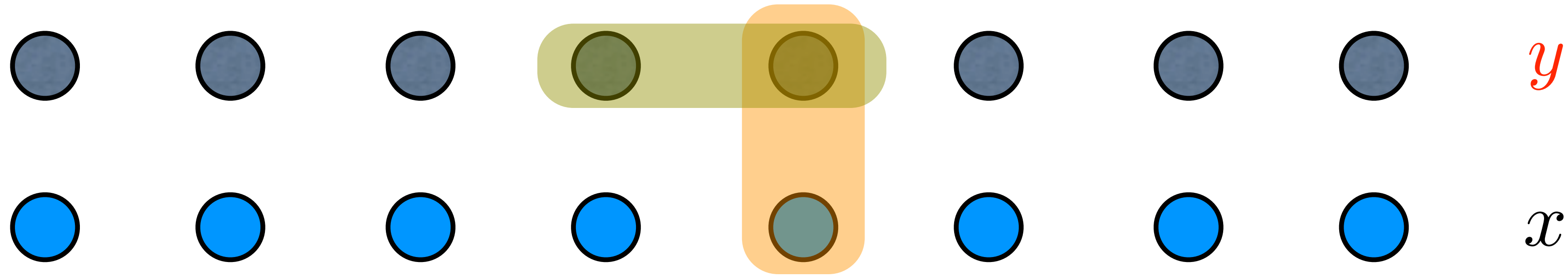
- the inference (argmax) must be efficient
  - either the search space GEN(x) is small, or factored
  - features must be local to y (but can be global to x)
    - e.g. bigram tagger, but look at all input words (cf. CRFs)



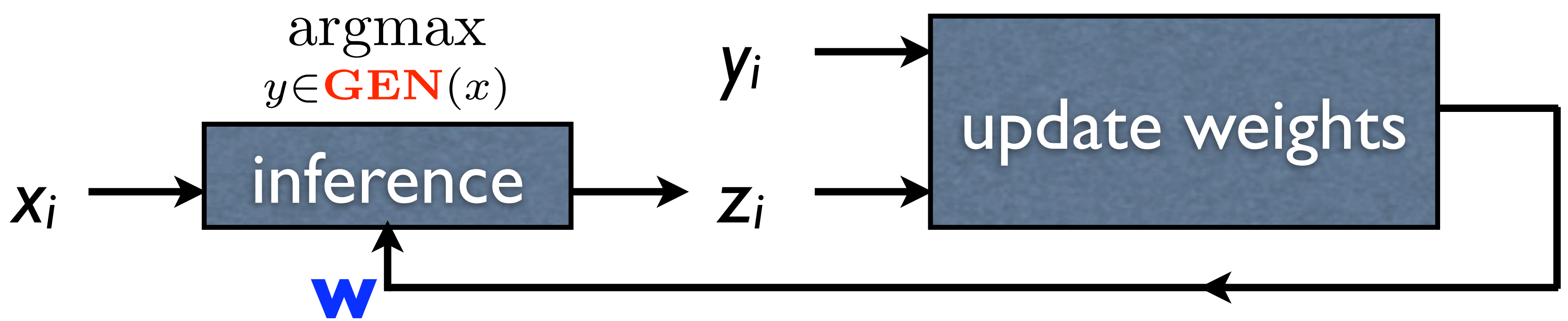
Efficiency vs. Expressiveness



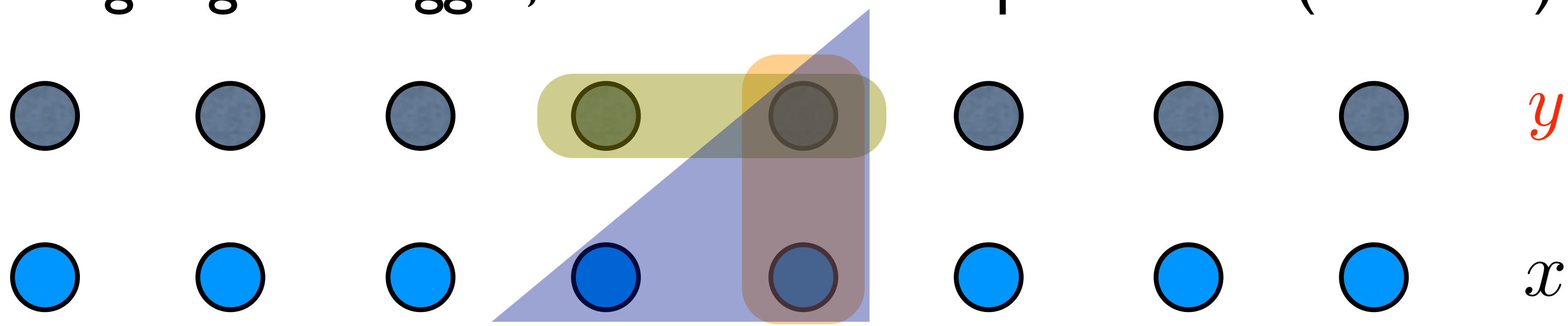
- the inference (argmax) must be efficient
  - either the search space GEN(x) is small, or factored
  - features must be local to y (but can be global to x)
    - e.g. bigram tagger, but look at all input words (cf. CRFs)



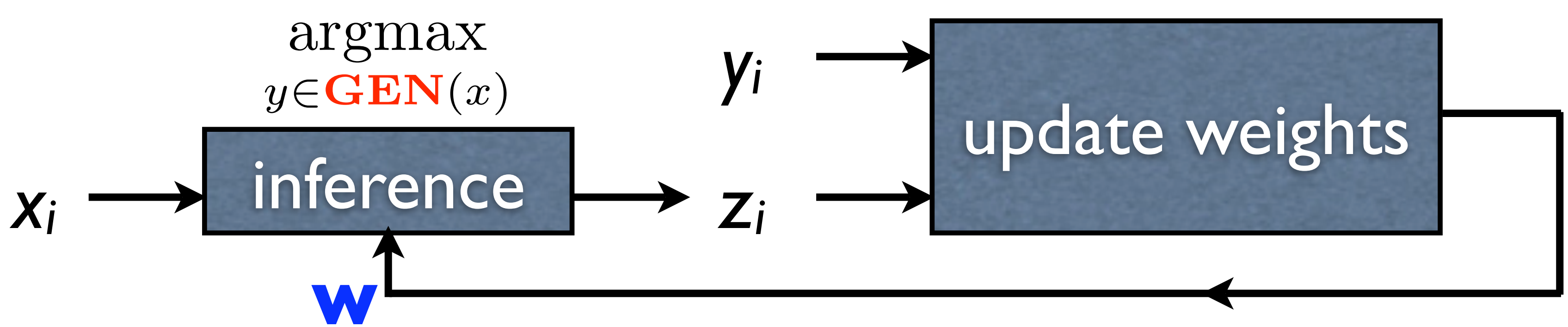
Efficiency vs. Expressiveness



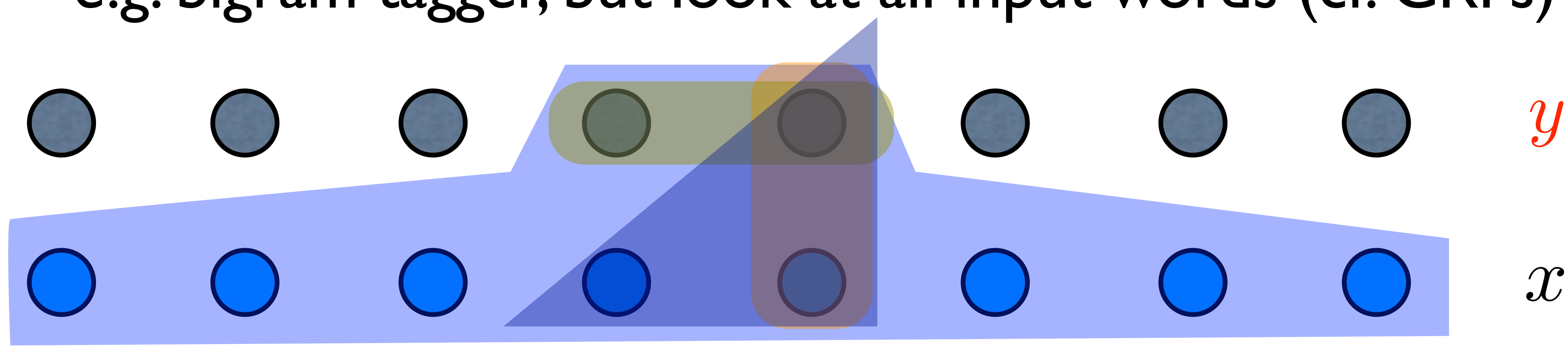
- the inference (argmax) must be efficient
  - either the search space GEN(x) is small, or factored
  - features must be local to y (but can be global to x)
    - e.g. bigram tagger, but look at all input words (cf. CRFs)



Efficiency vs. Expressiveness



- the inference (argmax) must be efficient
  - either the search space GEN(x) is small, or factored
  - features must be local to y (but can be global to x)
    - e.g. bigram tagger, but look at all input words (cf. CRFs)



What about tree-to-string?

Averaged Perceptron

Inputs: Training set (x_i, y_i) for $i = 1 \ldots n$

Initialization: $\mathbf{W}_0 = 0$

Define: $F(x) = \text{argmax}_{y \in \mathbf{GEN}(x)} \mathbf{\Phi}(x, y) \cdot \mathbf{W}$

Algorithm: For $t = 1 \ldots T$, $i = 1 \ldots n$

$z_i = F(x_i)$

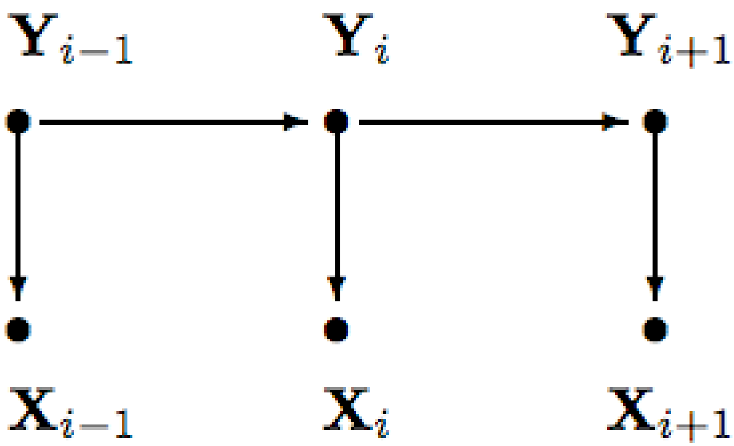
If $(z_i \neq y_i)$ $\mathbf{W}_{j+1} \leftarrow \mathbf{W}_j + \mathbf{\Phi}(x_i, y_i) - \mathbf{\Phi}(x_i, z_i)$

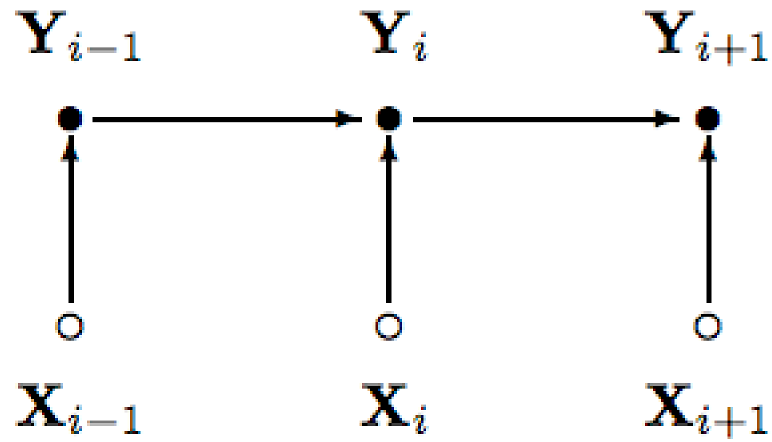
Output: Parameters $\mathbf{W} = \sum_j \mathbf{W}_j$

- more stable and accurate results
- approximation of voted perceptron (Freund & Schapire, 1999)

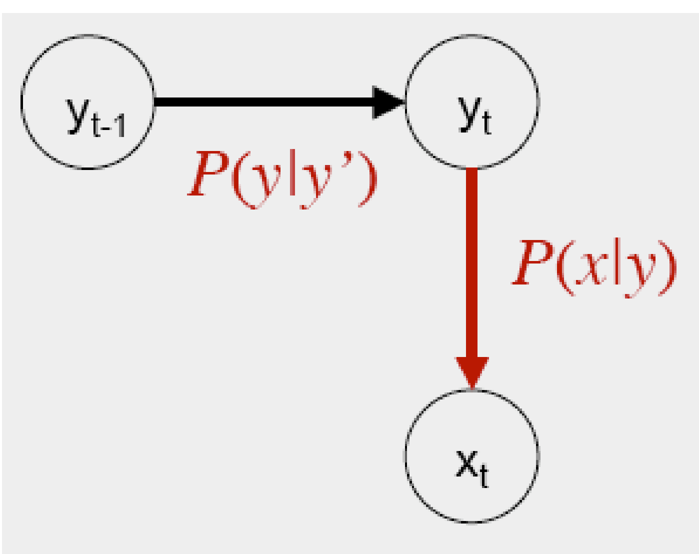
Comparison with Other Models

from HMM to MEMM

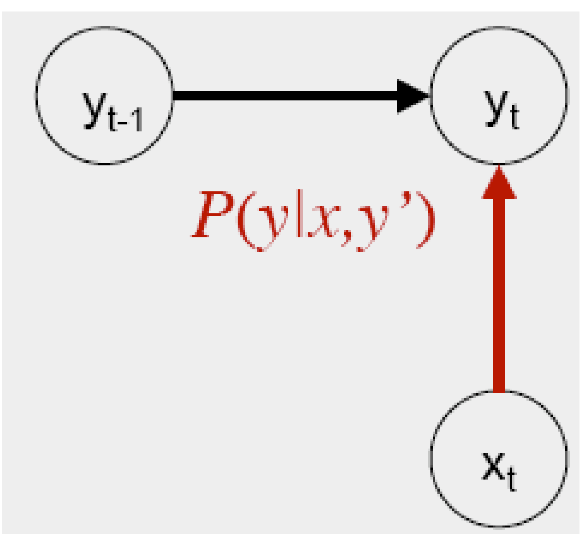




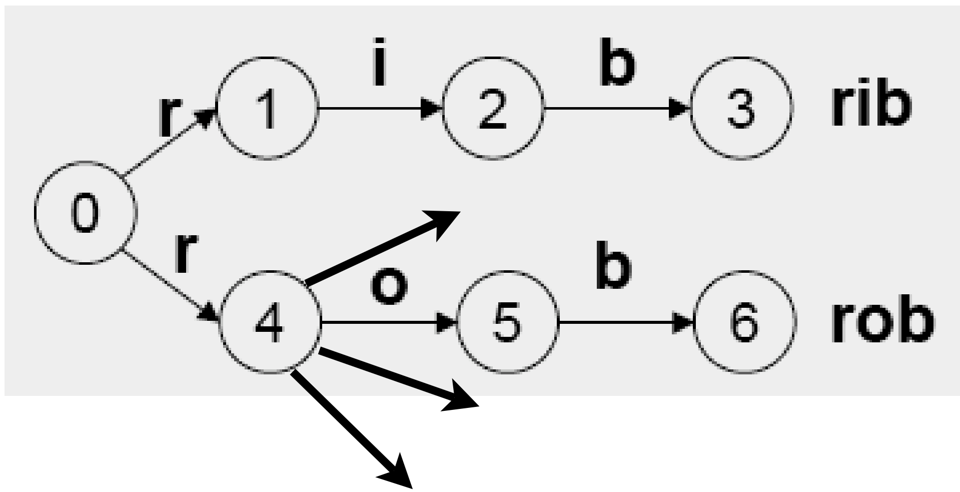
HMM: joint distribution

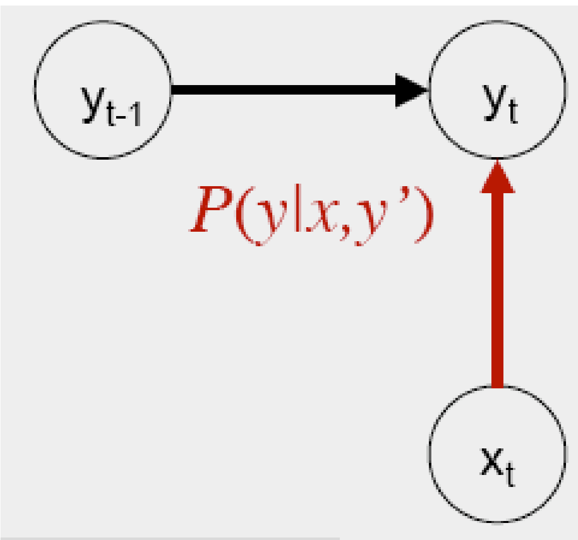


MEMM: locally normalized (per-state conditional)



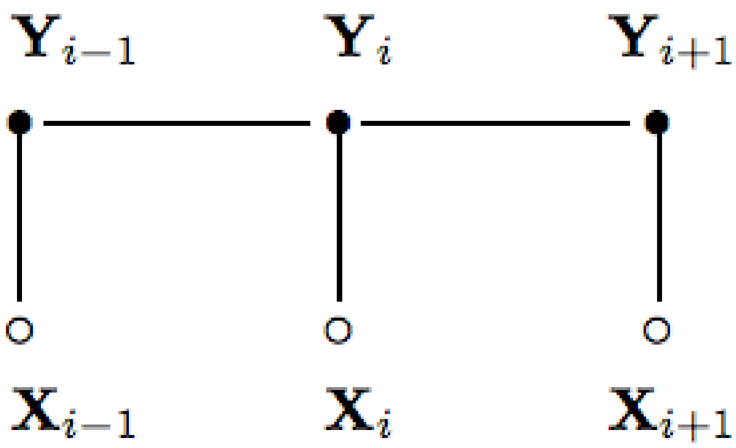
Label Bias Problem





- bias towards states with fewer outgoing transitions
- a problem with all locally normalized models

Conditional Random Fields



- globally normalized (no label bias problem)
- but training requires *expected* features counts
  - (related to the fractional counts in EM)
  - need to use Inside-Outside algorithm (sum)
- Perceptron just needs Viterbi (max)

Experiments

Experiments: Tagging

- (almost) identical features from (Ratnaparkhi, 1996)
  - trigram tagger: current tag t_i, previous tags t_{i-1}, t_{i-2}
  - current word w_i and its spelling features
  - surrounding words w_{i-1} w_{i+1} w_{i-2} w_{i+2}..

Method	Error rate/%	Numits
Perc, avg, cc=0	**2.93**	10
Perc, noavg, cc=0	3.68	20
Perc, avg, cc=5	3.03	6
Perc, noavg, cc=5	4.04	17
ME, cc=0	3.4	100
ME, cc=5	**3.28**	200

Experiments: NP Chunking

- B-I-O scheme

Rockwell International Corp.
B I I
's Tulsa unit said it signed
B I I O B O
a tentative agreement ...
B I I

- features:
  - unigram model
  - surrounding words and POS tags

Current word	w_i	$\& \ t_i$
Previous word	w_{i-1}	$\& \ t_i$
Word two back	w_{i-2}	$\& \ t_i$
Next word	w_{i+1}	$\& \ t_i$
Word two ahead	w_{i+2}	$\& \ t_i$
Bigram features	w_{i-2}, w_{i-1}	$\& \ t_i$
	w_{i-1}, w_i	$\& \ t_i$
	w_i, w_{i+1}	$\& \ t_i$
	w_{i+1}, w_{i+2}	$\& \ t_i$
Current tag	p_i	$\& \ t_i$
Previous tag	p_{i-1}	$\& \ t_i$
Tag two back	p_{i-2}	$\& \ t_i$
Next tag	p_{i+1}	$\& \ t_i$
Tag two ahead	p_{i+2}	$\& \ t_i$
Bigram tag features	p_{i-2}, p_{i-1}	$\& \ t_i$
	p_{i-1}, p_i	$\& \ t_i$
	p_i, p_{i+1}	$\& \ t_i$
	p_{i+1}, p_{i+2}	$\& \ t_i$
Trigram tag features	p_{i-2}, p_{i-1}, p_i	$\& \ t_i$
	p_{i-1}, p_i, p_{i+1}	$\& \ t_i$
	p_i, p_{i+1}, p_{i+2}	$\& \ t_i$

Experiments: NP Chunking

- results

Method	F-Measure	Numits
Perceptron, avg, cc=0	**93.53**	13
Perceptron, noavg, cc=0	93.04	35
Perceptron, avg, cc=5	93.33	9
Perceptron, noavg, cc=5	91.88	39
Max-ent, cc=0	92.34	900
Max-ent, cc=5	**92.65**	200

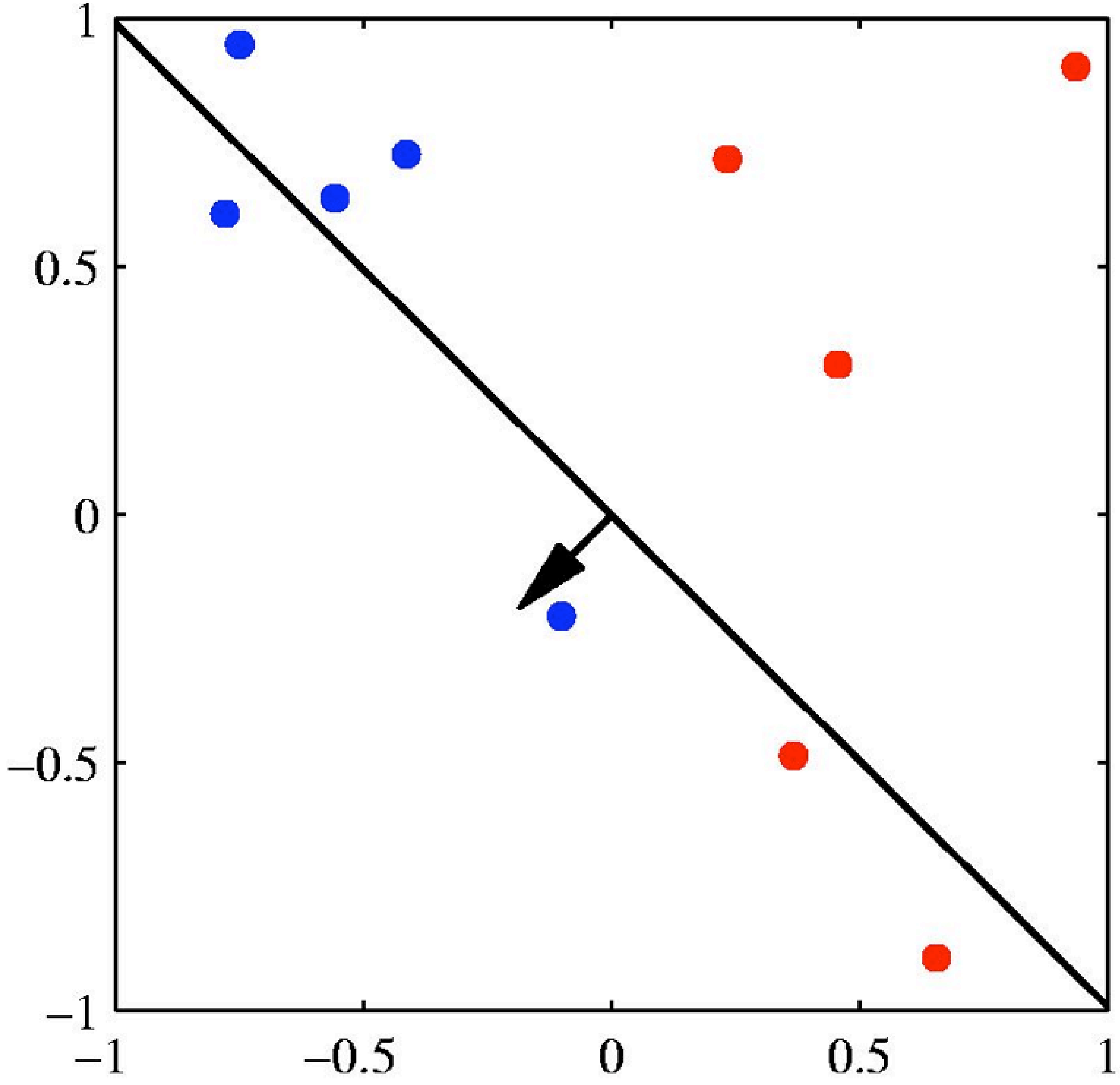
- (Sha and Pereira, 2003) trigram tagger
  - voted perceptron: 94.09% vs. CRF: 94.38%

Other NLP Applications

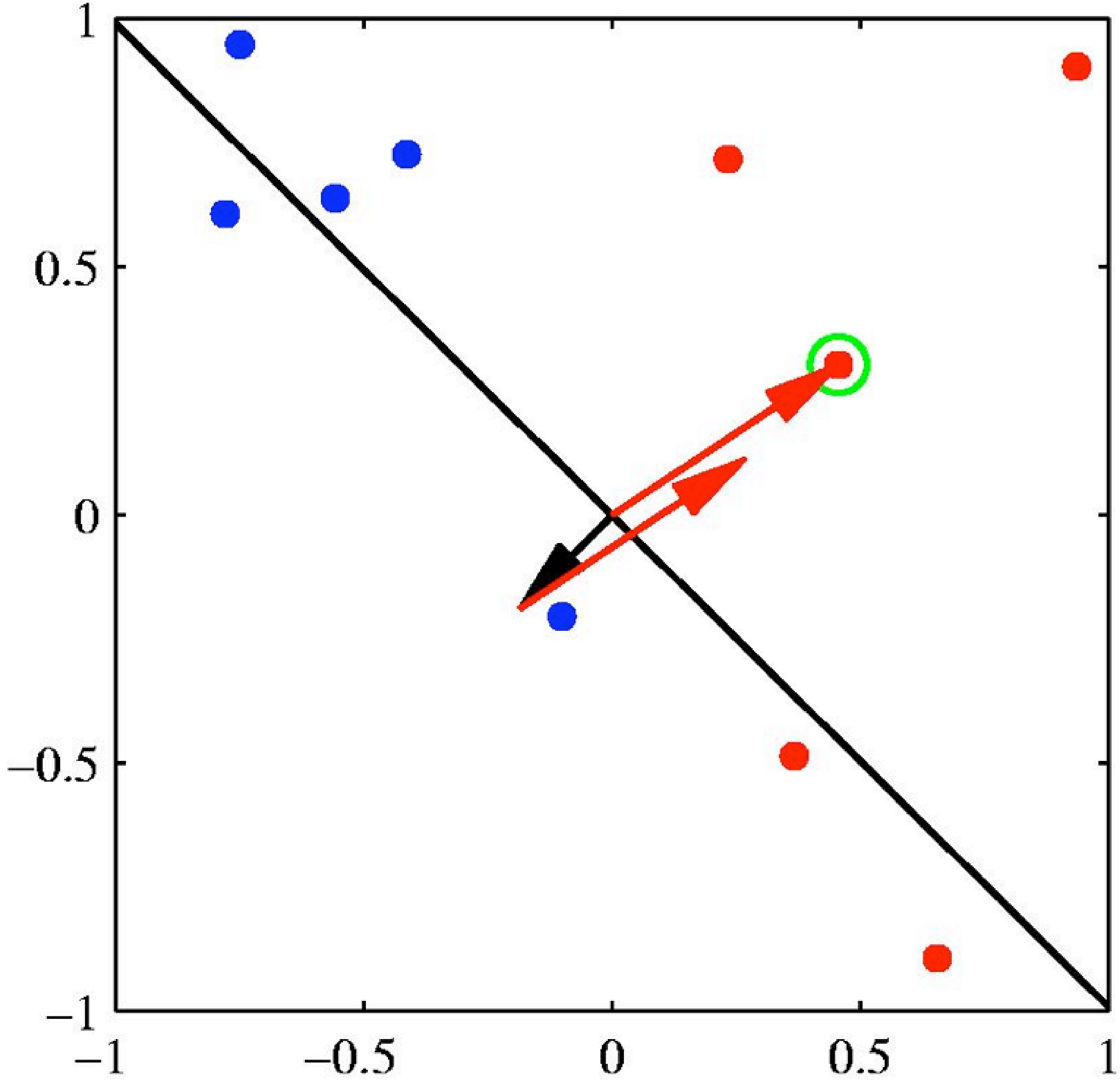
- dependency parsing (McDonald et al., 2005)
- parse reranking (Collins)
- phrase-based translation (Liang et al., 2006)
- word segmentation
- ... and many many more ...

Theory

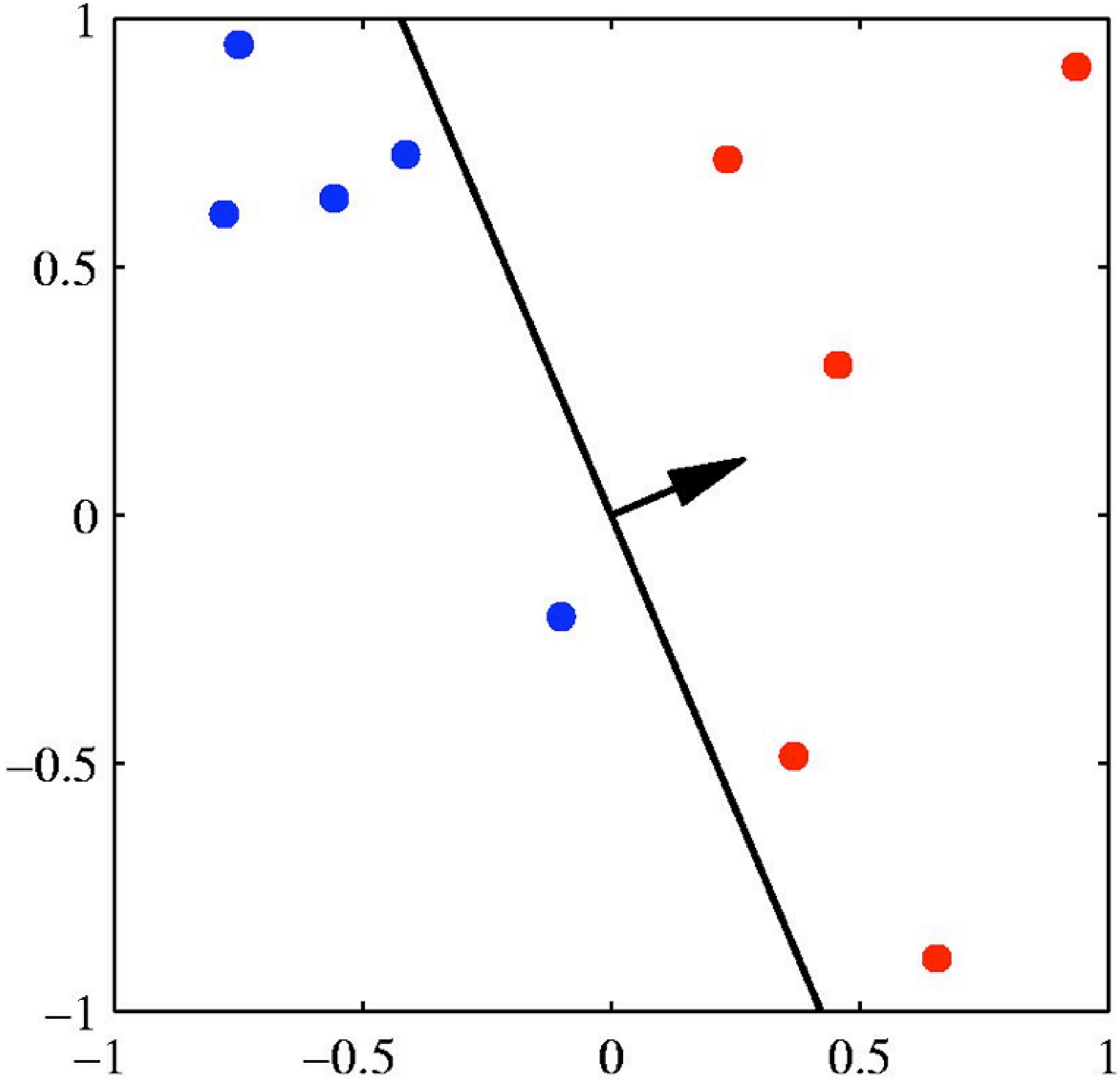
Vanilla Perceptron



Vanilla Perceptron



Vanilla Perceptron



Convergence Theorem

- Data is separable if and only if perceptron converges
  - number of updates is bounded by $(R/\gamma)^2$
  - γ is the margin; $R = \max_i \| \mathrm{x}_i \|$
- This result generalizes to structured perceptron

$$R = \max_i \|\Phi(x_i, y_i) - \Phi(x_i, z_i)\|$$

- Also in the paper: theorems for non-separable cases and generalization bounds

Conclusion

- a very simple framework that can work with many structured problems and that works very well
  - all you need is (fast) 1-best inference
  - much simpler than CRFs and SVMs
  - can be applied to parsing, translation, etc.
- generalization bounds depend on separability
  - not the (exponential) size of the search space
- extensions: MIRA, k-best MIRA, ...
- major limitation: only local features