Log-linear models and CRFs

[unused CRF+perceptron slides in this slidedeck too]

CS 585, Fall 2015 -- Oct. 1

Introduction to Natural Language Processing

http://people.cs.umass.edu/~brenocon/inlp2015/

Brendan O'Connor

UMASS CS

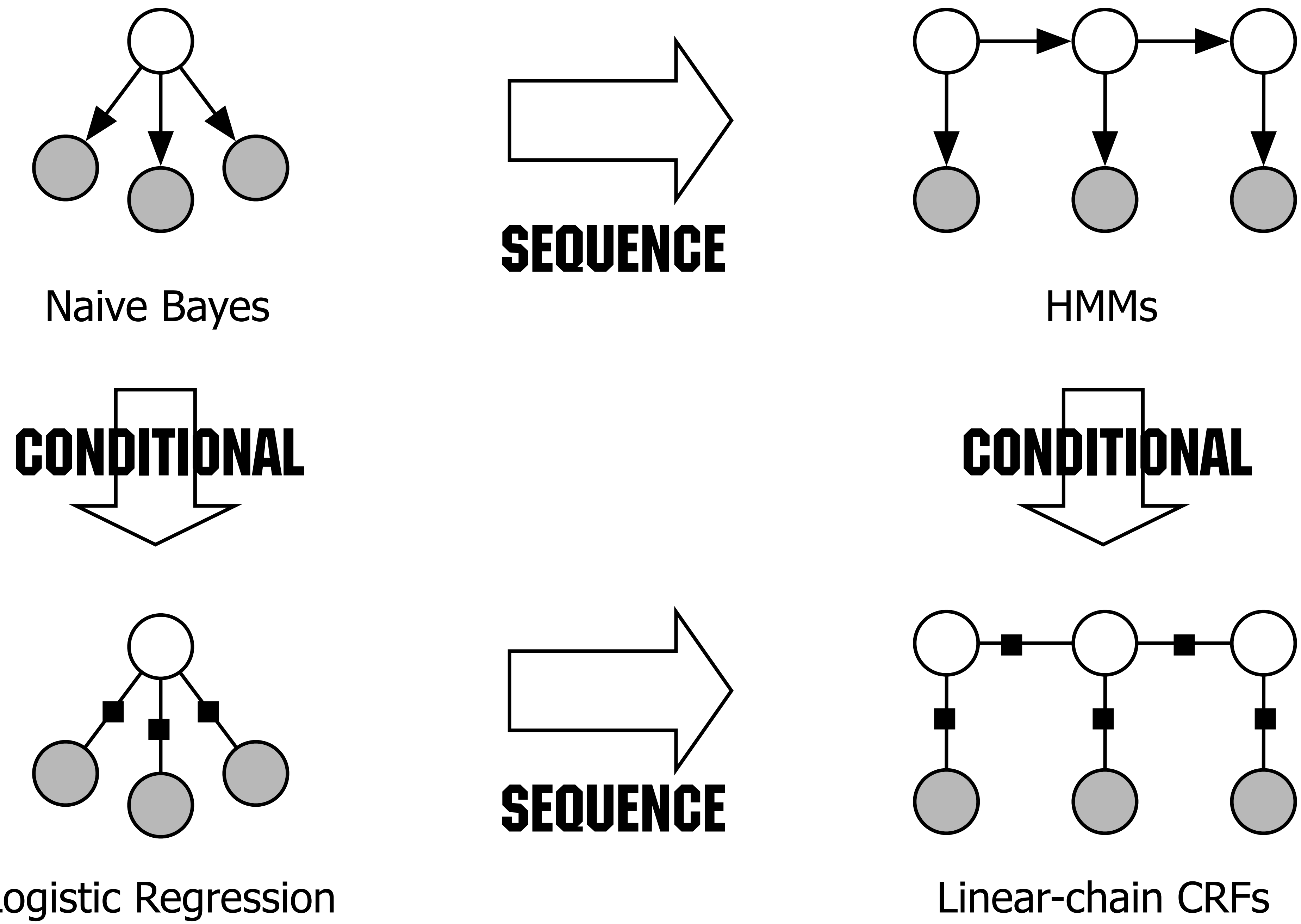
SCHOOL OF COMPUTER SCIENCE

Today

1. Motivation: we want features in our sequence model!
2. And how do we learn the parameters?

3. Outline
   1. Log-linear models
   2. Log-linear Sequence Models:
      1. Log-scale additive Viterbi
      2. Conditional Random Fields
   3. Learning: the Perceptron

These are all **log-linear** models



[from Sutton&McCallum reading]

[OLD SLIDE, OLD NOTATION]

- Input document **d** (a string...)
- Engineer a <u>feature function</u>, f(d), to generate <u>feature vector</u> ***x***

$$f(d) \longrightarrow x$$

f(d) = (
Count of "happy",
(Count of "happy") / (Length of doc),
log(1 + count of "happy"),
Count of "not happy",
Count of words in my pre-specified word list, "positive words according to my favorite psychological theory",
Count of "of the",
Length of document,
...
)

Typically these use <u>feature templates</u>:
Generate many features at once

for each word w:
- ${w}_count
- ${w}_log_1_plus_count
- ${w}_with_NOT_before_it_count
- ....

- Not just word counts. Anything that might be useful!
- <u>Feature engineering</u>: when you spend a lot of trying and testing new features. Very important for effective classifiers!! This is a place to put linguistics in.

[OLD SLIDE, OLD NOTATION]

Classification: LogReg (I)

- compute **features** (x's)
- given **weights** (betas)
- compute the **dot product**

$$z = \sum_{i=0}^{|X|} \beta_i x_i$$

Decision rule:

z > 0 -> Decide y*=POS

z <= 0 -> Decide y*=NEG

Log-linear models

- The form will generalize to multiclass and sequences...
  - x: Text Data
  - y: Proposed class
  - θ: Feature weights (model parameters)
  - f(x,y): Feature extractor, produces feature vector

dot product notation:

$$Goodness(y) = \sum_i \theta_i f_i(x, y) \qquad \equiv \theta^\mathsf{T} f(x, y)$$

$$p(y|x) \propto \exp G(y) \quad \Leftrightarrow \quad \log p(y|x) = C + G(y)$$

Decision rule: $\arg\max_{y^*} G(y^*)$

NB and LogReg can be expressed in this form...

Log-linear notation

$G(y) = \beta^T f(x,y)$

f(x,y) based on these feature templates:
key: (class=y AND word=w)
value: count of w

β

```
{"POS_The": +0.01,
 "NEG_The": -0.01,
 "POS_awesome": +2.2,
 "NEG_awesome": -2.2,
...}
```

f(x, POS)

```
{"POS_The": 3,
 "POS_awesome": 7,
 "POS_quizzical": 0,
...}
```

f(x, NEG)

```
{"NEG_The": 3,
 "NEG_awesome”: 7,
...}
```

$\beta^T f(x,POS) = ...$

$\beta^T f(x,NEG) = ...$

Log-linear models

- The form will generalize to multiclass and sequences...
  - x: Text Data
  - y: Proposed class
  - θ: Feature weights (model parameters)
  - f(x,y): Feature extractor, produces feature vector

dot product notation:

$$Goodness(y) = \sum_i \theta_i f_i(x, y) \qquad \equiv \theta^\mathsf{T} f(x, y)$$

$$p(y|x) \propto \exp G(y) \quad \Leftrightarrow \quad \log p(y|x) = C + G(y)$$

Decision rule: $\arg\max_{y^*} G(y^*)$

NB and LogReg can be expressed in this form...

Log-linear models

- The form will generalize to multiclass and sequences...
  - x: Text Data
  - y: Proposed class **or Proposed SEQUENCE**
  - θ: Feature weights (model parameters)
  - f(x,y): Feature extractor, produces feature vector

dot product notation:

$$Goodness(y) = \sum_i \theta_i f_i(x, y) \qquad \equiv \theta^\mathsf{T} f(x, y)$$

$$p(y|x) \propto \exp G(y) \quad \Leftrightarrow \quad \log p(y|x) = C + G(y)$$

Decision rule: $\arg\max_{y^*} G(y^*)$

NB and LogReg can be expressed in this form...

HMMs and CRFs can be expressed in this form...

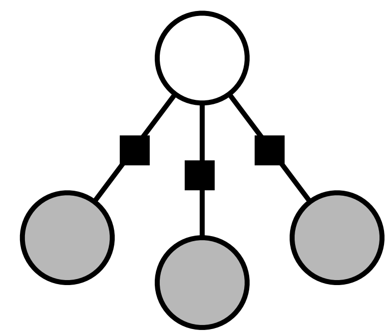
Today

1. Motivation: we want features in our sequence model!
2. And how do we learn the parameters?

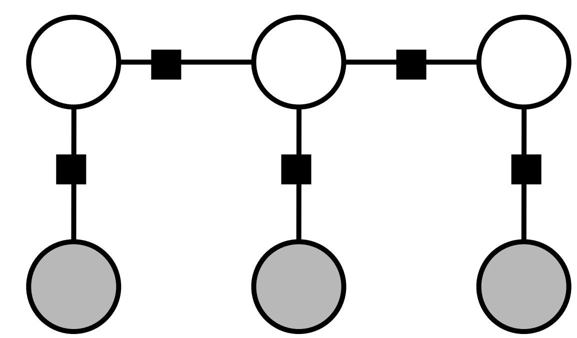
3. Outline
   1. Log-linear models
   2. **Log-linear Sequence Models:**
      1. Log-scale additive Viterbi
      2. Conditional Random Fields
   3. Learning: the Perceptron

CRF motivation: best of both worlds

- Want info from **features**
  - Is this the first token in the sentence?
  - Second? Third? Last? Next to last?
  - Word to left? Right?
  - Last 3 letters of this word? Last 3 letters of word on left? On right?
  - Is this word capitalized? Does it contain punctuation?

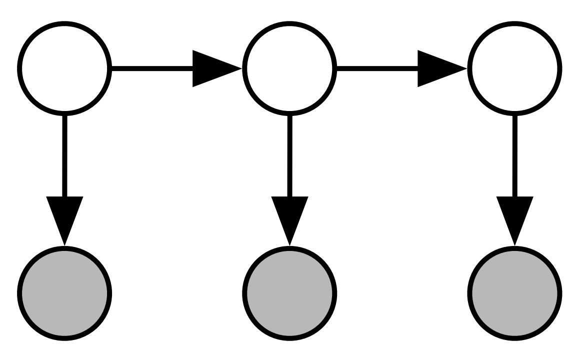


Logistic Regression



Linear-chain CRFs

- Want info from **POS Context**
  - What tags are left/right?
- Need joint decoding (Viterbi)

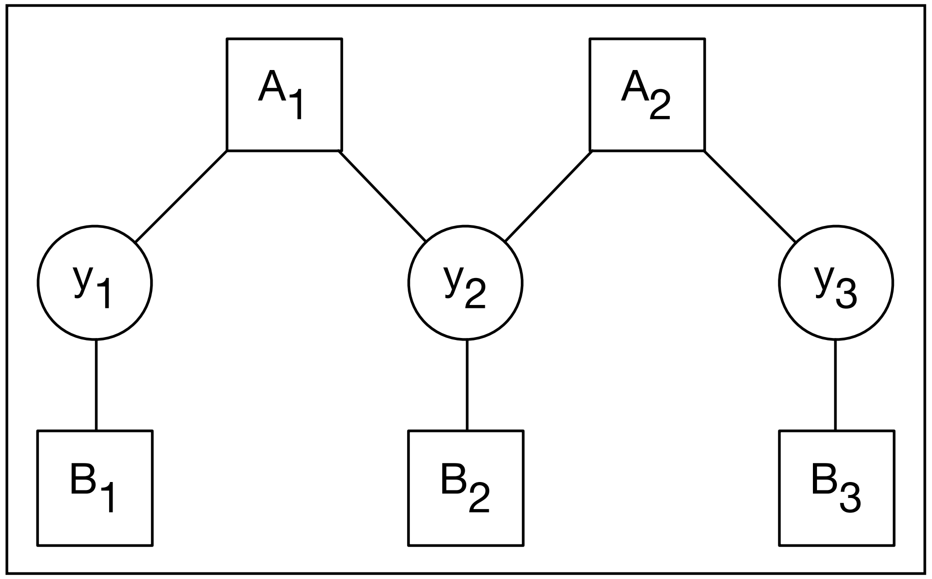


HMMs

From HMM to CRF

1. An HMM is a type of log-linear model with "transition" and "emission" features.
2. Do **discriminative learning**: Instead of learning the weights as simple conditional probabilities .... learn them to make high-accuracy sequence predictions
   [The *structured perceptron*: predict the entire sequence (Viterbi), then update weights where there are errors.]
3. Throw in lots more features!

HMM as factor graph



$$p(y, w) = \prod_t p(y_{t+1}|y_t)p(w_t|y_t)$$

$$\log p(y, w) = \sum_t \log p(w_t|y_t) + \log p(y_{t+1}|y_t)$$

$G(y)$ ↑ ***goodness***

$B_t(y_t)$ ↑ ***emission factor score***

$A(y_t, y_{t+1})$ ↑ ***transition factor score***

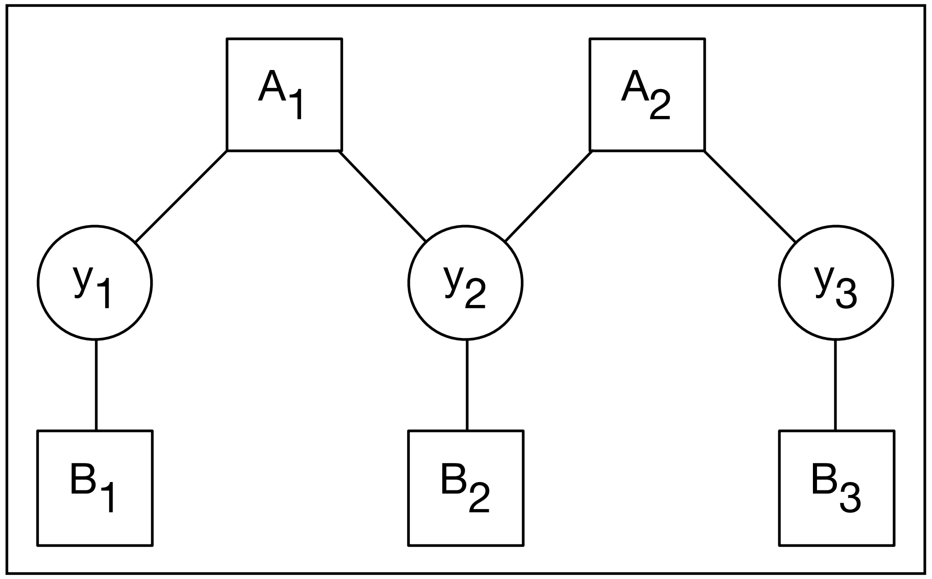
You can do Viterbi with these log-scale factor scores.

"Additive Viterbi" let's call it?

-- See Exercise! --

- stopped here on 10/1

HMM as log-linear



$$p(y, w) = \prod_t p(y_{t+1}|y_t)p(w_t|y_t)$$

$$\log p(y, w) = \sum_t \log p(y_t|w_t) + \log p(y_{t+1}|y_t)$$

$G(y)$
goodness

$B_t(y_t)$
emission factor score

$A(y_t, y_{t+1})$
transition factor score

$$\text{G(y)} = \sum_t \left[\sum_{k \in K} \sum_{w \in V} \mu_{w,k} 1\{y_t = k \wedge w_t = w\} + \sum_{k,j \in K} \lambda_{j,k} 1\{y_t = j \wedge y_{t+1} = k\} \right]$$

$$= \sum_t \sum_{i \in \text{allfeats}} \theta_i f_{t,i}(y_t, y_{t+1}, w_t)$$

$$= \sum_{i \in \text{allfeats}} \theta_i f_i(y_t, y_{t+1}, w_t)$$

[~ SM eq 1.13, 1.14]

CRF

- prob dist over whole sequence

$$\log p(\vec{y}|\vec{x}) = C + \theta^{\mathsf{T}} \vec{f}(\vec{x}, \vec{y})$$

- linear chain CRF:

$$\vec{f}(\vec{x}, \vec{y}) = \sum_t \vec{f}_t(\vec{x}, y_t, y_{t+1})$$

- its feature functions decompose over functions of neighboring tags.
- advantages
  - 1. why just word identity features? add many more!
  - 2. can train it to optimize accuracy of sequences (discriminative learning)

- is there a terrible bug in sutton&mccallum? there's no sum over *t* in these equations!

We can write (1.13) more compactly by introducing the concept of *feature functions*, just as we did for logistic regression in (1.7). Each feature function has the form $f_k(y_t, y_{t-1}, x_t)$. In order to duplicate (1.13), there needs to be one feature $f_{ij}(y, y', x) = \mathbf{1}_{\{y=i\}}\mathbf{1}_{\{y'=j\}}$ for each transition (i, j) and one feature $f_{io}(y, y', x) = \mathbf{1}_{\{y=i\}}\mathbf{1}_{\{x=o\}}$ for each state-observation pair (i, o). Then we can write an HMM as:

$$p(\mathbf{y}, \mathbf{x}) = \frac{1}{Z} \exp\left\{\sum_{k=1}^{K} \lambda_k f_k(y_t, y_{t-1}, x_t)\right\}. \qquad (1.14)$$

Definition 1.1
Let Y, X be random vectors, $\Lambda = \{\lambda_k\} \in \Re^K$ be a parameter vector, and $\{f_k(y, y', \mathbf{x}_t)\}_{k=1}^K$ be a set of real-valued feature functions. Then a *linear-chain conditional random field* is a distribution $p(\mathbf{y}|\mathbf{x})$ that takes the form

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \exp\left\{\sum_{k=1}^{K} \lambda_k f_k(y_t, y_{t-1}, \mathbf{x}_t)\right\}, \qquad (1.16)$$

Today

1. Motivation: we want features in our sequence model!
2. And how do we learn the parameters?

3. Outline
   1. Log-linear models
   2. Log-linear Sequence Models:
      1. Log-scale additive Viterbi
      2. Conditional Random Fields
   3. **Learning: the Perceptron**

Perceptron learning algorithm

- For ~10 iterations
  - For each (x,y) in dataset
    - PREDICT

$$y^* = \arg\max_{y'} \theta^\mathsf{T} f(x, y')$$

    - IF y=y*, do nothing
    - ELSE update weights

$$\theta := \theta + r[f(x,y) - f(x,y^*)]$$

learning rate constant
e.g. r=0.01

Features for
TRUE label

Features for
PREDICTED label

Update rule

y=POS
x="this awesome movie ..."
Make mistake: y*=NEG

learning rate e.g. r=0.01 → Features for TRUE label → Features for PREDICTED label →

$$\theta := \theta + r[f(x,y) - f(x,y^*)]$$

	POS_awesome	POS_this	POS_oof	....	NEG_awesome	NEG_this	NEG_oof	....
f(x, POS) =	1	1	0	....	0	0	0	....
f(x, NEG) =	0	0	0	....	1	1	0	....
f(x, POS) – f(x, NEG) =	+1	+1	0	....	-1	-1	0	....

Update rule

learning rate
e.g. r=0.01

Features for
TRUE label

Features for
PREDICTED label

$$\theta := \theta + r[f(x, y) - f(x, y^*)]$$

For binary features...

For each feature j in true y but not predicted y*:

$$\theta_j := \theta_j + (r) f_j(x, y)$$

For each feature j not in true y, but in predicted y*:

$$\theta_j := \theta_j - (r) f_j(x, y)$$

Perceptron issues -- for next time

- Does it converge? (sometimes, but generally no)
  - Solution: the **averaged perceptron**
    Take weight vectors every once in a while and average them
- Can you regularize it? No... just averaging...
- By the way ... there's also *likelihood* training out there