Recall: HMMs

- **Input** $x = (x_1, ..., x_n)$  
  **Output** $y = (y_1, ..., y_n)$

$$P(y, x) = P(y_1) \prod_{i=2}^n P(y_i | y_{i-1}) \prod_{i=1}^n P(x_i | y_i)$$

- **Training**: maximum likelihood estimation (with smoothing)
- **Inference problem**: $\arg\max_y P(y|x) = \arg\max_y \frac{P(y, x)}{P(x)}$
- Exponentially many possible $y$ here!
- **Viterbi**:  
  $$\text{score}_i(s) = \max_{y_i} P(s | y_{i-1}) P(x_i | s) \text{score}_{i-1}(y_{i-1})$$

This Lecture

- Generative vs. discriminative models
- CRFs for sequence modeling
- Named entity recognition (NER)
- Structured SVM
- (if time) Beam search

Named Entity Recognition

- **BIO tagset**: begin, inside, outside
- POS tagging is a plausible generative model of language — NER with this vanilla tag set is not
- What’s different about modeling $P(y|x)$ directly vs. $P(x, y)$ and computing the posterior later?
What does the model say when both lights are red?
- \( P(b,r,r) = \frac{1}{7} \times \frac{1}{7} \times 1 \) = 1/49 = 4/28
- \( P(w,r,r) = \frac{6}{7} \times \frac{1}{2} \times \frac{1}{2} \times \frac{1}{2} \) = 6/28 = 6/28
- \( P(w|r,r) = 6/10! \)

- Lights are working — wrong!

What if \( P(b) \) were 1/2 instead of 1/7 (the NB estimate)?
- \( P(b,r,r) = \frac{1}{2} \times \frac{1}{7} \times \frac{1}{7} \times 1 \) = 1/2 = 4/8
- \( P(w,r,r) = \frac{1}{2} \times \frac{6}{7} \times \frac{1}{2} \times \frac{1}{2} \times \frac{1}{2} \) = 1/8 = 1/8
- \( P(w|r,r) = 1/5! \)

- Lights are broken — correct! Data likelihood is lower but posterior \( P(y|x) \) is more accurate
Conditional Random Fields

- HMMs are expressible as Bayes nets (factor graphs)

\[ P(y, x) = P(y_1)P(x_1|y_1)P(y_2|y_1)P(x_2|y_2) \ldots \]

- CRFs: discriminative models with the following globally-normalized form:

\[ P(y|x) = \frac{1}{Z} \prod_k \exp(\phi_k(x,y)) \]

\[ Z = \sum_y \exp(\sum_k \phi_k(x,y)) \]

\[ \text{normalizer} \]

- Naive Bayes : logistic regression :: HMMs : CRFs

local vs. global normalization <-> generative vs. discriminative

- How do we max over y? Intractable in general — can we fix this?

Locally normalized model: each factor is a probability distribution that normalizes

\[ P(y|x) \propto \prod_k \exp(\phi_k(x,y)) \]

Sequential CRFs

- HMMs: \( P(y, x) = P(y_1)P(x_1|y_1)P(y_2|y_1)P(x_2|y_2) \ldots \)

- CRFs: \( P(y|x) \propto \prod_k \exp(\phi_k(x,y)) \)

\[ P(y|x) \propto \exp(\phi_o(y_1)) \prod_{i=2}^{n} \exp(\phi_t(y_{i-1}, y_i)) \prod_{i=1}^{n} \exp(\phi_e(x_i, y_i)) \]

We condition on x, so every variable can depend on all of x

x can’t depend arbitrarily on y in a generative model — would make inference hard

x token index — lets us look at current word
Sequential CRFs:

\[
P(y|\mathbf{x}) = \frac{1}{Z} \prod_{i=2}^{n} \exp(\phi_t(y_{i-1}, y_i)) \prod_{i=1}^{n} \exp(\phi_c(y_i, i, \mathbf{x}))
\]

...in fact, we typically don’t show \(\mathbf{x}\) at all.

Don’t include initial distribution, can bake into other factors.

Sequential CRFs:

\[
P(y|\mathbf{x}) = \frac{1}{Z} \prod_{i=2}^{n} \exp(\phi_t(y_{i-1}, y_i)) \prod_{i=1}^{n} \exp(\phi_c(y_i, i, \mathbf{x}))
\]

Computing (arg)maxes:

\[
P(y|\mathbf{x}) = \frac{1}{Z} \prod_{i=2}^{n} \exp(\phi_t(y_{i-1}, y_i)) \prod_{i=1}^{n} \exp(\phi_c(y_i, i, \mathbf{x}))
\]

\[
\text{argmax}_y P(y|\mathbf{x}): \text{can use Viterbi exactly as in HMM case}
\]

\[
\max_{y_1, \ldots, y_n} e^{\phi_t(y_{n-1}, y_n)} e^{\phi_c(y_n, n, \mathbf{x})} \cdots e^{\phi_t(y_2, 2, \mathbf{x})} e^{\phi_t(y_1, y_2)} e^{\phi_c(y_1, 1, \mathbf{x})}
\]

\[
= \max_{y_1} e^{\phi_t(y_{n-1}, y_n)} e^{\phi_c(y_n, n, \mathbf{x})} \cdots e^{\phi_c(y_2, 2, \mathbf{x})} e^{\phi_c(y_1, 1, \mathbf{x})}
\]

\[
= \max_{y_1} \max_{y_2} e^{\phi_t(y_{n-1}, y_n)} e^{\phi_c(y_n, n, \mathbf{x})} \cdots \max_{y_n} e^{\phi_t(y_2, 2, \mathbf{x})} e^{\phi_c(y_1, 1, \mathbf{x})}
\]

\[
\exp(\phi_t(y_{i-1}, y_i)) \text{and } \exp(\phi_c(y_i, i, \mathbf{x})) \text{ play the role of the Ps now, same dynamic program}
\]

Computing Marginals:

\[
P(y|\mathbf{x}) = \frac{1}{Z} \prod_{i=2}^{n} \exp(\phi_t(y_{i-1}, y_i)) \prod_{i=1}^{n} \exp(\phi_c(y_i, i, \mathbf{x}))
\]

\[
\text{Normalizing constant } Z = \sum_{\mathbf{y}} \prod_{i=2}^{n} \exp(\phi_t(y_{i-1}, y_i)) \prod_{i=1}^{n} \exp(\phi_c(y_i, i, \mathbf{x}))
\]

Analogous to \(P(\mathbf{x})\) for HMMs.

For both HMMs and CRFs:

\[
P(y_i = s|\mathbf{x}) = \frac{\text{forward}_i(s) \text{backward}_i(s)}{Z}\text{ for CRFs, } P(\mathbf{x})\text{ for HMMs}
\]

for HMMs;

\[
P(y_i = s, \mathbf{x}) = \frac{\text{forward}_i(s) \text{backward}_i(s)}{Z}\text{ for CRFs, } P(\mathbf{x})\text{ for HMMs}
\]

Inference in General CRFs:

\[
\text{Can do inference in any tree-structured CRF}
\]

\[
\text{Sum-product algorithm: generalization of forward-backward to arbitrary tree-structured graphs}
\]

We’ll come back to this in a few lectures when we deal with other kinds of graphs.
Feature Functions

\[ P(y|x) = \frac{1}{Z} \prod_{i=2}^{n} \exp(\phi_t(y_{i-1}, y_i)) \prod_{i=1}^{n} \exp(\phi_e(y_i, i, x)) \]

- \( \phi_t \) can have sophisticated features! Generally look like linear models
  \[ \phi_e(y_i, i, x) = w^\top f_e(y_i, i, x) \]
  \[ \phi_t(y_{i-1}, y_i) = w^\top f_t(y_{i-1}, y_i) \]
- Log-linear model — structurally like logistic regression!

Training CRFs

\[ P(y|x) = \frac{1}{Z} \prod_{i=2}^{n} \exp(\phi_t(y_{i-1}, y_i)) \prod_{i=1}^{n} \exp(\phi_e(y_i, i, x)) \]

- Assume \( \phi_t \) and \( \phi_e \) are both linear feature functions \( w^\top f(\text{args}) \)

\[ \mathcal{L}(y^*, x) = \log P(y^*|x) = \sum_{i=2}^{n} w^\top f_t(y_{i-1}^*, y_i^*) + \sum_{i=1}^{n} w^\top f_e(x_i, y_i^*) - \log Z \]

- Gradient is gold features minus expected features under model, like in LR

\[ \frac{\partial}{\partial w_j} \mathcal{L}(y^*, x) = \sum_{i=2}^{n} f_{t,j}(y_{i-1}^*, y_i^*) + \sum_{i=1}^{n} f_{e,j}(x_i, y_i^*) - \mathbb{E}_x \left[ \sum_{i=2}^{n} f_{t,j}(y_{i-1}, y_i) + \sum_{i=1}^{n} f_{e,j}(x_i, y_i) \right] \]

Training CRFs

- \( \mathcal{L}(y^*, x) = \log P(y^*|x) = \sum_{i=2}^{n} w^\top f_t(y_{i-1}^*, y_i^*) + \sum_{i=1}^{n} w^\top f_e(x_i, y_i^*) - \log Z \)

Implementation Tips

- Often many features but only a few are active on a single sentence even across many different labels
- Maintain the gradient as a sparse vector for efficiency
  - Counter in `utils.py` is a way to do this

How to compute expectations?
- Forward-backward helps you compute \( P(y_i = s| x) \)
- Take weighted sum over all features at all tags and positions
- Transition features: need to compute \( P(y_i = s_1, y_{i+1} = s_2| x) \) using forward-backward as well
- ...but you can build a pretty good system without transition features
Basic Features for NER

\[ P(y|x) = \frac{1}{Z} \prod_{i=2}^{n} \exp(\phi_t(y_{i-1}, y_i)) \prod_{i=1}^{n} \exp(\phi_e(y_i, i, x)) \]

Barack Obama will travel to Hangzhou today for the G20 meeting.

Transitions: \( f_t(y_{i-1}, y_i) = \text{Ind}[y_{i-1} & y_i] \)

Emissions: \( f_e(y_6, 6, x) = \text{Ind}[\text{B-LOC & Current word = Hangzhou}] \)
\( \text{Ind}[\text{B-LOC & Prev word = to}] \)

Features for NER

- Word features
- Capitalization
- Word shape
- Prefixes/suffixes
- Lexical indicators
- Context features
  - Words before/after
  - Tags before/after
- Word clusters
- Gazetteers

Nonlocal Features

The news agency Tanjug reported on the outcome of the meeting.

The delegation met the president at the airport, Tanjug said.

- Various ways to capture this information — we’ll talk about this in a few lectures

Finkel and Manning (2008), Ratimov and Roth (2009)
**Semi-Markov Models**

- Barack Obama will travel to Hangzhou today for the G20 meeting.

  - **PER**  O  **LOC**  O  **ORG**  O

- Chunk-level prediction rather than token-level BIO
- y is a set of touching spans of the sentence
- Viterbi looks like looping over all spans that could lead to a given point
- Pros: features can look at whole span at once
- Cons: there’s an extra factor of n during inference

  

  \[ \text{Sarawagi and Cohen (2004)} \]

**Evaluating NER**

- Barack Obama will travel to Hangzhou today for the G20 meeting.

  - **B-PER**  **I-PER**  O  O  **B-LOC**  O  O  **B-ORG**  O  O

- Prediction of all Os still gets 66% accuracy on this example!
- What we really want to know: how many named entity chunk predictions did we get right?
- Precision: of the ones we predicted, how many are right?
- Recall: of the gold named entities, how many did we find?
- F-measure: harmonic mean of these two
- Partial credit? Typically no but more complex metrics exist

**Evaluating NER**

- Barack Obama will travel to Hangzhou today for the G20 meeting.

  - **B-PER**  **I-PER**  O  O  **B-LOC**  O  O  **B-ORG**  O  O

- More correct: ROC curve
- Measure the area under the curve as a way of evaluating the system holistically

**How well do NER systems do?**

<table>
<thead>
<tr>
<th>System</th>
<th>Resources Used</th>
<th>$F_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>+ LBJ-NER</td>
<td>Wikipedia, Nonlocal Features, Word-class Model</td>
<td>90.80</td>
</tr>
<tr>
<td>- (Suzuki and Isozaki, 2008) Semi-supervised on 1G-word unlabeled data</td>
<td>89.92</td>
<td></td>
</tr>
<tr>
<td>- (Ando and Zhang, 2005) Semi-supervised on 27M-word unlabeled data</td>
<td>89.31</td>
<td></td>
</tr>
<tr>
<td>- (Kazama and Torisawa, 2007a)</td>
<td></td>
<td>88.02</td>
</tr>
<tr>
<td>- (Kazama and Torisawa, 2007b) Non-local Features</td>
<td>87.24</td>
<td></td>
</tr>
<tr>
<td>+ (Finkel et al., 2005) Non-local Features</td>
<td>86.86</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Model</th>
<th>$F_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Collobert et al. (2011)*</td>
<td>89.59</td>
</tr>
<tr>
<td>Lin and Wu (2009)</td>
<td>83.78</td>
</tr>
<tr>
<td>Lin and Wu (2009)*</td>
<td>90.90</td>
</tr>
<tr>
<td>Huang et al. (2015)*</td>
<td>90.10</td>
</tr>
<tr>
<td>Passos et al. (2014)</td>
<td>90.05</td>
</tr>
<tr>
<td>Passos et al. (2014)*</td>
<td>90.90</td>
</tr>
<tr>
<td>Luo et al. (2015)* + gaz</td>
<td>89.9</td>
</tr>
<tr>
<td>Luo et al. (2015)* + gaz + linking</td>
<td>91.2</td>
</tr>
<tr>
<td>Chiu and Nichols (2015)</td>
<td>90.69</td>
</tr>
<tr>
<td>Chiu and Nichols (2015)*</td>
<td>90.77</td>
</tr>
<tr>
<td>LSTM-CRF (no char)</td>
<td>90.20</td>
</tr>
<tr>
<td>LSTM-CRF</td>
<td>90.94</td>
</tr>
<tr>
<td>S-LSTM (no char)</td>
<td>87.96</td>
</tr>
<tr>
<td>S-LSTM</td>
<td>90.33</td>
</tr>
</tbody>
</table>

Ratinov and Roth (2009) Lample et al. (2016)
Structured SVM

- CRF: \( \log P(y|x) \propto \sum_{i=2}^{n} w^T f_i(y_{i-1}, y_i) + \sum_{i=1}^{n} w^T f_e(x_i, y_i) \)

- We can formulate an SVM using the same features

\[
w^T f(x, y) = \sum_{i=2}^{n} w^T f_i(y_{i-1}, y_i) + \sum_{i=1}^{n} w^T f_e(x_i, y_i)
\]

Minimize \( \lambda \|w\|^2 + \sum_{j=1}^{m} \xi_j \)

s.t. \( \forall j \xi_j \geq 0 \)

\( \forall j \forall y \in Y \ w^T f(x_j, y_j^*) \geq w^T f(x_j, y) + \ell(y, y_j^*) - \xi_j \)

- Exponentially large state space! Use Viterbi for loss-augmented decode
- Same as normal Viterbi but boost wrong labels’ scores by 1 (if using Hamming loss)
- Only need Viterbi, not forward-backward…hmm…

Viterbi Time Complexity

<table>
<thead>
<tr>
<th></th>
<th>VBD</th>
<th>VBN</th>
<th>VB</th>
<th>VBZ</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NNP</td>
<td>NNS</td>
<td>NN</td>
<td>NNS</td>
</tr>
</tbody>
</table>

Fed raises interest rates 0.5 percent

- n word sentence, s tags to consider — what is the time complexity?
- \( O(ns^2) \) — s is ~40 for POS, n is ~20

- Many tags are totally implausible
- Can any of these be:
  - Determiners?
  - Prepositions?
  - Adjectives?
- Features quickly eliminate many outcomes from consideration — don’t need to consider these going forward
Beam Search

- Maintain a beam of $k$ plausible states at the current timestep
- Expand all states, only keep $k$ top hypotheses at new state

- VBD +1.2
- NNP +0.9
- VBN +0.7
- NN +0.3

Fed raises

- O(nks) time complexity with beam size of $k$

How good is beam search?

- Big enough beam size: always exact! Usually works well even with smaller beams
- What’s the case when $k=1$?
- How about when there’s no transition model?
  - Depends on the strength of nonlocal interactions — we’ll come back to this later!

Implementation Tips for CRFs

- Caching is your friend! Cache feature vectors especially
- Try to reduce redundant computation, e.g. if you compute both the gradient and the objective value, don’t rerun the dynamic program
- Exploit sparsity in feature vectors where possible. The weight vector needs to be stored explicitly, but all features and gradients are typically faster to handle sparsely
- Think about your data structures: if things are too slow

Debugging Tips for CRFs

- Hard to know whether inference, learning, or the model is broken!
- Compute the objective — is optimization working?
  - **Inference**: check gradient computation (most likely place for bug)
    - Are expectations being computed correctly? Do probabilities normalize / expectations look reasonable?
  - **Learning**: are you applying the gradient correctly?
- If objective is going down but model performance is bad:
  - **Inference**: check performance if you decode the training set
  - **Model**: if dev set performance is bad: work on features more!
Next Time

- Unsupervised sequence modeling
- Writing tips as you prepare your report