Sequential Data Modeling - Conditional Random Fields

Graham Neubig
Nara Institute of Science and Technology (NAIST)

Reference:
An introduction to conditional random fields for relational learning.
Prediction Problems

Given $x$, predict $y$
Prediction Problems

Given \( x \), predict \( y \)

<table>
<thead>
<tr>
<th>A book review</th>
<th>Is it positive?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oh, man I love this book!</td>
<td>yes</td>
</tr>
<tr>
<td>This book is so boring...</td>
<td>no</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A tweet</th>
<th>Its language</th>
</tr>
</thead>
<tbody>
<tr>
<td>On the way to the park!</td>
<td>English</td>
</tr>
<tr>
<td>公園に行くなう！</td>
<td>Japanese</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A sentence</th>
<th>Its parts-of-speech</th>
</tr>
</thead>
<tbody>
<tr>
<td>I read a book</td>
<td>I read a book</td>
</tr>
</tbody>
</table>

Binary Prediction (2 choices)
Multi-class Prediction (several choices)
Structured Prediction (millions of choices)
Logistic Regression
Example we will use:

• Given **an introductory sentence from Wikipedia**

• Predict **whether the article is about a person**

Give

Gonso was a Sanron sect priest (754-827) in the late Nara and early Heian periods.

Predict

Yes!

Shichikuzan Chigogataki Fudomyoo is a historical site located at Magura, Maizuru City, Kyoto Prefecture.

No!

• This is **binary classification** (of course!)
Review: Linear Prediction Model

- Each element that helps us predict is a feature
  - contains “priest”
  - contains “site”
  - contains “(<#>-<#>)”
  - contains “Kyoto Prefecture”

- Each feature has a weight, positive if it indicates “yes”, and negative if it indicates “no”
  - \[ w_{\text{contains “priest”}} = 2 \]
  - \[ w_{\text{contains “site”}} = -3 \]
  - \[ w_{\text{contains “(<#>-<#>)”}} = 1 \]
  - \[ w_{\text{contains “Kyoto Prefecture”}} = -1 \]

- For a new example, sum the weights
  - Kuya (903-972) was a priest born in Kyoto Prefecture.
  - \[ 2 + -1 + 1 = 2 \]

- If the sum is at least 0: “yes”, otherwise: “no”
Review: Mathematical Formulation

\[
y = \text{sign}(w \cdot \varphi(x)) \\
= \text{sign}(\sum_{i=1}^{I} w_i \cdot \varphi_i(x))
\]

- **\(x\):** the input
- **\(\varphi(x)\):** vector of feature functions \(\{\varphi_1(x), \varphi_2(x), \ldots, \varphi_I(x)\}\)
- **\(w\):** the weight vector \(\{w_1, w_2, \ldots, w_I\}\)
- **\(y\):** the prediction, +1 if “yes”, -1 if “no”
  - (\(\text{sign}(v)\) is +1 if \(v \geq 0\), -1 otherwise)
Perceptron and Probabilities

- Sometimes we want the probability $P(y|x)$
- Estimating confidence in predictions
- Combining with other systems
- However, perceptron only gives us a prediction

$$y = \text{sign}(w \cdot \varphi(x))$$

In other words:

1. $P(y = 1|x) = 1$ if $w \cdot \varphi(x) \geq 0$
2. $P(y = 1|x) = 0$ if $w \cdot \varphi(x) < 0$
The Logistic Function

- The logistic function is a “softened” version of the function used in the perceptron

\[ P(y = 1 \mid x) = \frac{e^{w \cdot \phi(x)}}{1 + e^{w \cdot \phi(x)}} \]

- Can account for uncertainty
- Differentiable
Logistic Regression

- Train based on **conditional likelihood**
- Find the parameters \( \mathbf{w} \) that maximize the conditional likelihood of all answers \( y_i \) given the example \( x_i \)

\[
\hat{\mathbf{w}} = \arg\max_{\mathbf{w}} \prod_i P(y_i|x_i; \mathbf{w})
\]

- How do we solve this?
Review: Perceptron Training Algorithm

create map w
for / iterations
    for each labeled pair x, y in the data
        phi = create_features(x)
        y' = predict_one(w, phi)
        if y' != y
            w += y * phi

• In other words
  • Try to classify each training example
  • Every time we make a mistake, update the weights
Stochastic Gradient Descent

- Online training algorithm for probabilistic models (including logistic regression)

```
create map w
for / iterations
    for each labeled pair x, y in the data
        w += α * dP(y|x)/dw
```

- In other words
  - For every training example, calculate the gradient (the direction that will increase the probability of y)
  - Move in that direction, multiplied by learning rate α
Gradient of the Logistic Function

- Take the derivative of the probability

\[
\frac{d}{dw} P(y = 1|x) = \frac{d}{dw} \frac{e^{w \cdot \varphi(x)}}{1 + e^{w \cdot \varphi(x)}} = \varphi(x) \frac{e^{w \cdot \varphi(x)}}{(1 + e^{w \cdot \varphi(x)})^2}
\]

\[
\frac{d}{dw} P(y = -1|x) = \frac{d}{dw} \left(1 - \frac{e^{w \cdot \varphi(x)}}{1 + e^{w \cdot \varphi(x)}}\right) = -\varphi(x) \frac{e^{w \cdot \varphi(x)}}{(1 + e^{w \cdot \varphi(x)})^2}
\]
Example: Initial Update

- Set $\alpha=1$, initialize $w=0$

$x = \text{A site, located in Maizuru, Kyoto} \quad y = -1$

$$w \cdot \varphi(x) = 0 \quad \frac{d}{dw} P(y=-1|x) = -\frac{e^0}{(1+e^0)^2} \varphi(x)$$

$$= -0.25 \varphi(x)$$

$$w \leftarrow w + -0.25 \varphi(x)$$

- $W_{\text{unigram "Maizuru"}} = -0.25$
- $W_{\text{unigram ","}} = -0.5$
- $W_{\text{unigram "in"}} = -0.25$
- $W_{\text{unigram "Kyoto"}} = -0.25$
- $W_{\text{unigram "A"}} = -0.25$
- $W_{\text{unigram "site"}} = -0.25$
- $W_{\text{unigram "located"}} = -0.25$
**Example: Second Update**

\[ x = \text{Shoken, monk born in Kyoto} \quad y = 1 \]

\[
\mathbf{w} \cdot \varphi(x) = -1
\]

\[
\frac{d}{d \mathbf{w}} P(y = 1 | x) = \frac{e^1}{(1 + e^1)^2} \varphi(x)
\]

\[
= 0.196 \varphi(x)
\]

\[
\mathbf{w} \leftarrow \mathbf{w} + 0.196 \varphi(x)
\]

\[\begin{align*}
W_{\text{unigram “Maizuru”}} &= -0.25 \\
W_{\text{unigram “,”}} &= -0.304 \\
W_{\text{unigram “in”}} &= -0.054 \\
W_{\text{unigram “Kyoto”}} &= -0.054 \\
W_{\text{unigram “A”}} &= -0.25 \\
W_{\text{unigram “site”}} &= -0.25 \\
W_{\text{unigram “located”}} &= -0.25 \\
W_{\text{unigram “Shoken”}} &= 0.196 \\
W_{\text{unigram “monk”}} &= 0.196 \\
W_{\text{unigram “born”}} &= 0.196
\end{align*}\]
SGD Learning Rate?

- How to set the learning rate $\alpha$?
- Usually decay over time:

$$\alpha = \frac{1}{C + t}$$

parameter \hspace{1cm} number of samples
Other Online Gradient-based Methods

- **Problem:** SGD can be instable or slow depending on the learning rate

- **Other options:**
  - **SGD+Momentum:** Move in the direction of the rolling average of past updates
  - **AdaGrad:** Normalizes the gradient by the variance of past updates
  - **AdaDelta:** Similar to AdaGrad, but focuses only on recent gradients

Discussion (1)
Discussion Time (1)

• Think of a problem that is related to your research that you can solve with a classifier.

• In this problem:
  • What is the input X?
  • What is the output Y?
  • What is the advantage of having a probability estimate? What can you do with it?

• Please discuss:
  • For 5 minutes in small groups.
  • I will ask for a few groups to answer at the end of the 5 minutes (in English or Japanese).
Calculating Optimal Sequences, Probabilities
Sequence Likelihood

• Logistic regression considered probability of $y \in \{-1, +1\}$

$P(y|x)$

• What if we want to consider probability of a sequence?

$P(Y|X)$
Calculating Multi-class Probabilities

- Each sequence has its own feature vector

<table>
<thead>
<tr>
<th>Feature Vector</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>time flies N V</td>
<td>φ_{T,&lt;S&gt;,N} = 1, φ_{T,N,V} = 1, φ_{T,V,&lt;S&gt;} = 1, φ_{E,N,time} = 1, φ_{E,V,flies} = 1</td>
</tr>
<tr>
<td>time flies V N</td>
<td>φ_{T,&lt;S&gt;,V} = 1, φ_{T,V,N} = 1, φ_{T,N,&lt;S&gt;} = 1, φ_{E,V,time} = 1, φ_{E,N,flies} = 1</td>
</tr>
<tr>
<td>time flies N N</td>
<td>φ_{T,&lt;S&gt;,N} = 1, φ_{T,N,N} = 1, φ_{T,N,&lt;S&gt;} = 1, φ_{E,N,time} = 1, φ_{E,N,flies} = 1</td>
</tr>
<tr>
<td>time flies V V</td>
<td>φ_{T,&lt;S&gt;,V} = 1, φ_{T,V,V} = 1, φ_{T,V,&lt;S&gt;} = 1, φ_{E,V,time} = 1, φ_{E,V,flies} = 1</td>
</tr>
</tbody>
</table>

- Use weights for each feature to calculate scores

<table>
<thead>
<tr>
<th>Weight</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>w_{T,&lt;S&gt;,N}</td>
<td>1</td>
</tr>
<tr>
<td>w_{T,V,&lt;S&gt;}</td>
<td>1</td>
</tr>
<tr>
<td>w_{E,N,time}</td>
<td>1</td>
</tr>
<tr>
<td>( \varphi ) (time flies N V) * w = 3</td>
<td>( \varphi ) (time flies V N) * w = 0</td>
</tr>
<tr>
<td>( \varphi ) (time flies N N) * w = 2</td>
<td>( \varphi ) (time flies V V) * w = 1</td>
</tr>
</tbody>
</table>
The Softmax Function

- Turn into probabilities by taking exponent and normalizing (the Softmax function)

\[
P(Y|X) = \frac{e^{w \cdot \varphi(Y, X)}}{\sum_{\hat{Y}} e^{w \cdot \varphi(\hat{Y}, X)}}
\]

- Take the exponent and normalize

\[
\begin{align*}
\exp(\varphi(\text{time flies}_{\text{N V}}) \cdot w) &= 20.08 \\
\exp(\varphi(\text{time flies}_{\text{V N}}) \cdot w) &= 1.00 \\
\exp(\varphi(\text{time flies}_{\text{N N}}) \cdot w) &= 7.39 \\
\exp(\varphi(\text{time flies}_{\text{V V}}) \cdot w) &= 2.72
\end{align*}
\]

\[
\begin{align*}
P(\text{N V} | \text{time flies}) &= .6437 \\
P(\text{V N} | \text{time flies}) &= .0320 \\
P(\text{N N} | \text{time flies}) &= .2369 \\
P(\text{V V} | \text{time flies}) &= .0872
\end{align*}
\]
Calculating Edge Features

- Like perceptron, can calculate features for each edge

\[
\begin{align*}
\varphi_{E,N,\text{time}} &= 1 \\
\varphi_{T,<S>,N} &= 1 \\
\varphi_{E,N,\text{flies}} &= 1 \\
\varphi_{T,N,N} &= 1 \\
\varphi_{E,V,\text{time}} &= 1 \\
\varphi_{T,<S>,V} &= 1 \\
\varphi_{E,V,\text{flies}} &= 1 \\
\varphi_{T,V,N} &= 1 \\
\varphi_{E,V,\text{flies}} &= 1 \\
\varphi_{T,N,V} &= 1 \\
\varphi_{E,V,\text{flies}} &= 1 \\
\varphi_{T,V,\text{flies}} &= 1 \\
\varphi_{T,V,<S>} &= 1 \\
\varphi_{T,V,<S>} &= 1 \\
\varphi_{T,V,<S>} &= 1
\end{align*}
\]
Calculating Edge Probabilities

- Calculate scores, and take exponent

\[
\begin{align*}
\text{time} & \quad e^{w\phi}=1.00 \\
N & \quad e^{w\phi}=7.39 \\
P & =0.881 \\
V & \quad e^{w\phi}=1.00 \\
\text{flies} & \quad P =0.237 \\
N & \quad e^{w\phi}=1.00 \\
V & \quad e^{w\phi}=1.00 \\
& \quad P =0.644 \\
\langle S \rangle & \quad e^{w\phi}=2.72 \\
\text{V} & \quad P =0.731 \\
\langle /S \rangle & \quad e^{w\phi}=1.00 \\
\end{align*}
\]

- This is now the same form as the HMM

- Can use the Viterbi algorithm
- Calculate probabilities using forward-backward
Review of Forward-Backward

- $e^{w\varphi} = 7.39$
  - $f = 7.39$
  - $b = 3.72$
  - $P = 0.881$

- $e^{w\varphi} = 1.00$
  - $P = 0.237$

- $e^{w\varphi} = 7.39 + 1.00$
  - $P = 0.269$

- $e^{w\varphi} = 8.39 * 1.00 * 1.00 / 31.21$
  - $P = 0.032$

- $e^{w\varphi} = 1.00 * 1.00 * 1.00 / 31.21$
  - $P = 0.087$

- $e^{w\varphi} = 1.00 * 1.00 * 2.72 / 31.21$
  - $P = 0.032$

- $e^{w\varphi} = 8.39 * 2.72 * 1.00 / 31.21$
  - $P = 0.032$

- $e^{w\varphi} = 8.39 * 1.00 * 2.72 / 31.21$
  - $P = 0.032$

- $e^{w\varphi} = 1.00 * 1.00 * 1.00 / 31.21$
  - $P = 0.032$
Conditional Random Fields

“An Introduction to Conditional Random Fields for Relational Learning”
Charles Sutton and Andrew McCallum
Maximizing CRF Likelihood

- Want to maximize the likelihood for sequences

\[ \hat{\mathbf{w}} = \arg\max_{\mathbf{w}} \prod_i P(Y_i | X_i ; \mathbf{w}) \]

\[ P(Y | X) = \frac{e^{\mathbf{w} \cdot \varphi(Y, X)}}{\sum \tilde{Y} e^{\mathbf{w} \cdot \varphi(\tilde{Y}, X)}} \]

- For convenience, we consider the log likelihood

\[ \log P(Y | X) = \mathbf{w} \cdot \varphi(Y, X) - \log \sum \tilde{Y} e^{\mathbf{w} \cdot \varphi(\tilde{Y}, X)} \]

- Want to find gradient for stochastic gradient descent

\[ \frac{d}{d \mathbf{w}} \log P(Y | X) \]
Deriving a CRF Gradient:

\[
\log P(Y | X) = w \cdot \varphi(Y, X) - \log \sum \tilde{Y} e^{w \cdot \varphi(\tilde{Y}, X)}
\]

\[
= w \cdot \varphi(Y, X) - \log Z
\]

\[
\frac{d}{dw} \log P(Y | X) = \varphi(Y, X) - \frac{d}{dw} \log \sum \tilde{Y} e^{w \cdot \varphi(\tilde{Y}, X)}
\]

\[
= \varphi(Y, X) - \frac{1}{Z} \sum \tilde{Y} \frac{d}{dw} e^{w \cdot \varphi(\tilde{Y}, X)}
\]

\[
= \varphi(Y, X) - \sum \tilde{Y} \frac{e^{w \cdot \varphi(\tilde{Y}, X)}}{Z} \varphi(\tilde{Y}, X)
\]

\[
= \varphi(Y, X) - \sum \tilde{Y} P(\tilde{Y} | X) \varphi(\tilde{Y}, X)
\]
In Other Words...

- To get the gradient we:

\[
\frac{d}{d \mathbf{w}} \log P(Y | X) = \varphi(Y, X) - \sum_{\tilde{Y}} P(\tilde{Y} | X) \varphi(\tilde{Y}, X)
\]

add the correct feature vector

subtract the expectation of the features
Sequential Data Modeling – Conditional Random Fields

Example

\[
\begin{align*}
\phi_{T,<S>,N} &= 1, \quad \phi_{T,N,V} = 1, \quad \phi_{T,V,<S>} = 1, \quad \phi_{E,N,\text{time}} = 1, \quad \phi_{E,V,\text{flies}} = 1 \quad \text{P} = 0.644 \\
\phi_{T,<S>,V} &= 1, \quad \phi_{T,V,N} = 1, \quad \phi_{T,N,<S>} = 1, \quad \phi_{E,V,\text{time}} = 1, \quad \phi_{E,N,\text{flies}} = 1 \quad \text{P} = 0.032 \\
\phi_{T,<S>,N} &= 1, \quad \phi_{T,N,N} = 1, \quad \phi_{T,N,<S>} = 1, \quad \phi_{E,N,\text{time}} = 1, \quad \phi_{E,N,\text{flies}} = 1 \quad \text{P} = 0.237 \\
\phi_{T,<S>,V} &= 1, \quad \phi_{T,V,V} = 1, \quad \phi_{T,V,<S>} = 1, \quad \phi_{E,V,\text{time}} = 1, \quad \phi_{E,V,\text{flies}} = 1 \quad \text{P} = 0.087 \\
\end{align*}
\]

\[
\begin{align*}
\phi_{T,<S>,N}, \phi_{E,N,\text{time}} &= 1 - 0.644 - 0.237 = 0.119 \\
\phi_{T,N,V} &= 1 - 0.644 = 0.356 \\
\phi_{T,<S>,V}, \phi_{E,V,\text{time}} &= 0 - 0.032 - 0.087 = -0.119 \\
\phi_{T,V,N} &= 0 - 0.032 = -0.032 \\
\phi_{T,V,<S>}, \phi_{E,V,\text{flies}} &= 1 - 0.644 - 0.087 = 0.269 \\
\phi_{T,N,N} &= 0 - 0.237 = -0.237 \\
\phi_{T,N,<S>}, \phi_{E,V,\text{flies}} &= 0 - 0.032 - 0.237 = -0.269 \\
\phi_{T,V,V} &= 0 - 0.087 = -0.087 \\
\end{align*}
\]
Combinatorial Explosion

- **Problem!**: The number of hypotheses is exponential.

\[
\frac{d}{d \mathbf{w}} \log P(Y | X) = \varphi(Y, X) - \sum_{\mathbf{\tilde{Y}}} P(\mathbf{\tilde{Y}} | X) \varphi(\mathbf{\tilde{Y}}, X)
\]

\[O(T^{|X|})\]

\[T = \text{number of tags}\]
Calculate Feature Expectations using Edge Probabilities!

- If we know the edge probabilities, just multiply them!

\[
\begin{align*}
\phi_{T,<S>,N}, \phi_{E,N,time} &= 1 - .881 = .119 \\
\phi_{T,<S>,V}, \phi_{E,V,time} &= 0 - .119 = -.119
\end{align*}
\]

Same answer as when we explicitly expand all \( Y \)!

\[
\begin{align*}
\phi_{T,<S>,N}, \phi_{E,N,time} &= 1 - .644 - .237 = .119 \\
\phi_{T,<S>,V}, \phi_{E,V,time} &= 0 - .032 - .087 = -.119
\end{align*}
\]
CRF Training Procedure

- Can perform stochastic gradient descent, like logistic regression

```plaintext
create map w
for I iterations
    for each labeled pair X, Y in the data
        gradient = \( \varphi(Y, X) \)
        calculate \( e^{\varphi(y, x)w} \) for each edge
        run forward-backward algorithm to get \( P(edge) \)
        for each edge
            gradient -= \( P(edge) \)*\( \varphi(edge) \)
        w += \( \alpha \) * gradient
```

- Only major difference is gradient calculation
- Learning rate \( \alpha \)
Learning Algorithms
Batch Learning

- **Online Learning**: Update after each example

```plaintext
Online Stochastic Gradient Descent
create map w
for / iterations
    for each labeled pair x, y in the data
        w += α * dP(y|x)/dw
```

- **Batch Learning**: Update after all examples

```plaintext
Batch Stochastic Gradient Descent
create map w
for / iterations
    for each labeled pair x, y in the data
        gradient += α * dP(y|x)/dw
        w += gradient
```
Batch Learning Algorithms: Newton/Quasi-Newton Methods

- **Newton-Raphson Method:**
  - Choose how far to update using the second-order derivatives (the Hessian matrix)
  - Faster convergence, but $|w^*|^2 |w|$ time and memory
- **Limited Memory Broyden-Fletcher-Goldfarb-Shanno algorithm (L-BFGS):**
  - Guesses second-order derivatives from first-order
  - Most widely used?
- **More information:**
Online Learning vs. Batch Learning

- **Online:**
  - In general, simpler mathematical derivation
  - Often converges faster
- **Batch:**
  - More stable (does not change based on order)
Discussion (2)
Discussion Time (2)

- Think of a downstream task that we can use POS tags (or other sequence labels) for.

- In this problem:
  - How can we use the probabilities provided by CRFs in the downstream task?

- Please discuss:
  - For 5 minutes in small groups.
  - I will ask for a few groups to answer at the end of the 5 minutes (in English or Japanese).
Regularization
Cannot Distinguish Between Large and Small Classifiers

• For these examples:

<table>
<thead>
<tr>
<th>Classifier 1</th>
<th>Classifier 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>he +3</td>
<td>bird -1</td>
</tr>
<tr>
<td>saw -5</td>
<td>robbery +1</td>
</tr>
<tr>
<td>a +0.5</td>
<td></td>
</tr>
<tr>
<td>bird -1</td>
<td></td>
</tr>
<tr>
<td>robbery +1</td>
<td></td>
</tr>
<tr>
<td>in +5</td>
<td></td>
</tr>
<tr>
<td>the -3</td>
<td></td>
</tr>
<tr>
<td>park -2</td>
<td></td>
</tr>
</tbody>
</table>

Which classifier is better?
Sequential Data Modeling – Conditional Random Fields

Cannot Distinguish Between Large and Small Classifiers

- For these examples:

<table>
<thead>
<tr>
<th>Classifier 1</th>
<th>Classifier 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>he +3</td>
<td>bird -1</td>
</tr>
<tr>
<td>saw -5</td>
<td>robbery +1</td>
</tr>
<tr>
<td>a +0.5</td>
<td></td>
</tr>
<tr>
<td>bird -1</td>
<td></td>
</tr>
<tr>
<td>robbery +1</td>
<td></td>
</tr>
<tr>
<td>in +5</td>
<td></td>
</tr>
<tr>
<td>the -3</td>
<td></td>
</tr>
<tr>
<td>park -2</td>
<td></td>
</tr>
</tbody>
</table>

- Which classifier is better?

**Classifier 1**
- he saw a bird in the park: -1
- he saw a robbery in the park: +1

**Classifier 2**
- bird: -1
- robbery: +1

Probably classifier 2! It doesn't use irrelevant information.
Regularization

• A penalty on adding extra weights

• L2 regularization:
  • Big penalty on large weights, small penalty on small weights
  • High accuracy

• L1 regularization:
  • Uniform increase whether large or small
  • Will cause many weights to become zero → small model
Regularization in Logistic Regression/CRF

- To do so in logistic regression/CRF, we add the penalty to the log likelihood (for the whole corpus)

\[
\hat{w} = \arg\max_w \left( \log \prod_i P(Y_i | X_i; w) \right) - c \sum_{w \in w} w^2
\]

- \(c\) adjusts the strength of the regularization
  - smaller: more freedom to fit the data
  - larger: less freedom to fit the data, better generalization

- \(L1\) also used, slightly more difficult to optimize
  - Online learning: Sub-gradient descent
  - Batch learning: OWL-QN
Conclusion
Conclusion

- **Logistic regression** is a probabilistic classifier
- **Conditional random fields** are probabilistic structured discriminative prediction models
- Can be trained using
  - Online stochastic gradient descent (like perceptron)
  - Batch learning using a method such as L-BFGS
- **Regularization** can help solve problems of overfitting

“An Introduction to Conditional Random Fields for Relational Learning”
Charles Sutton and Andrew McCallum
Thank You!