Structured Output Prediction: Generative Models

CS6780 – Advanced Machine Learning
Spring 2015
Thorsten Joachims
Cornell University

Reading:
Murphy 17.3, 17.4, 17.5.1

Structured Output Prediction

• Supervised Learning from Examples
  – Find function from input space \( X \) to output space \( Y \)
  \[
h: X \rightarrow Y
\]
such that the prediction error is low.

• Typical
  – Output space is just a single number
    • Classification: \(-1, +1\)
    • Regression: some real number

• General
  – Predict outputs that are complex objects

Examples of Complex Output Spaces

• Natural Language Parsing
  – Given a sequence of words \( x \), predict the parse tree \( y \).
  – Dependencies from structural constraints, since \( y \) has to be a tree.

\[ x \rightarrow y \]

\[ \text{The dog chased the cat} \rightarrow \text{VP} \]

\[ \text{Det} \rightarrow \text{N} \]

\[ \text{V} \rightarrow \text{Det} \rightarrow \text{N} \]

Examples of Complex Output Spaces

• Multi-Label Classification
  – Given a (bag-of-words) document \( x \), predict a set of labels \( y \).
  – Dependencies between labels from correlations between labels (“iraq” and “oil” in newswire corpus)

\[ x \rightarrow y \]

\[ \text{Due to the continued violence in Baghdad, the oil price is expected to further increase.} \]

\[ \text{OPEC officials met with …} \]

\[ \text{antarctica, benelux, germany, iraq, oil, coal, trade, acquisitions} \]

Examples of Complex Output Spaces

• Noun-Phrase Co-reference
  – Given a set of noun phrases \( x \), predict a clustering \( y \).
  – Structural dependencies, since prediction has to be an equivalence relation.
  – Correlation dependencies from interactions.

\[ x \rightarrow y \]

\[ \text{The policeman fed the cat. He did not know that he was late.} \]

\[ \text{The cat is called Peter.} \]

Examples of Complex Output Spaces

• Scene Recognition
  – Given a 3D point cloud with RGB from Kinect camera
  – Segment into volumes
  – Geometric dependencies between segments (e.g. monitor usually close to keyboard)
Part-of-Speech Tagging Task

- Assign the correct part of speech (word class) to each word in a document
  - "The/DT planet/NN Jupiter/NNP and/CC its/PRP moons/NNS are/VBP in/IN effect/NN a/DT mini-solar/JJ system/NN, and/CC Jupiter/NNP itself/PRP is/VBZ often/RB called/VBN a/DT star/NN that/IN never/RB caught/VBN fire/NN."
- Needed as an initial processing step for a number of language technology applications
  - Information extraction
  - Answer extraction in QA
  - Base step in identifying syntactic phrases for IR systems
  - Critical for word-sense disambiguation (WordNet apps)
  - ...

Why is POS Tagging Hard?

- Ambiguity
  - He will race/VB the car.
  - When will the race/NN end?
  - I bank/VB at CFCU.
  - Go to the bank/NN!
- Average of ~2 parts of speech for each word
  - The number of tags used by different systems varies a lot. Some systems use < 20 tags, while others use > 400.

The POS Learning Problem

- Example

<table>
<thead>
<tr>
<th>sentence</th>
<th>POS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. (bank, at, CFCU)</td>
<td>(PRP, V, PREP, N)</td>
</tr>
<tr>
<td>2. (Go, to, the, bank)</td>
<td>(V, PREP, DET, N)</td>
</tr>
</tbody>
</table>

Markov Model

- Definition
  - Set of States: \( s_1, ..., s_k \)
  - Start probabilities: \( P(S_1 = s) \)
  - Transition probabilities: \( P(S_i = s | S_{i-1} = s') \)
- Random walk on graph
  - Start in state \( s \) with probability \( P(S_1 = s) \)
  - Move to next state with probability \( P(S_i = s | S_{i-1} = s') \)
- Assumptions
  - Limited dependence: Next state depends only on previous state, but no other state (i.e. first order Markov model)
  - Stationary: \( P(S_i = s | S_{i-1} = s') \) is the same for all \( i \)

Hidden Markov Model for POS Tagging

- States
  - Think about as nodes of a graph
  - One for each POS tag
  - Special start state (and maybe end state)
- Transitions
  - Think about as directed edges in a graph
  - Edges have transition probabilities
- Output
  - Each state also produces a word of the sequence
  - Sentence is generated by a walk through the graph

Hidden Markov Model

- States: \( y \in \{s_1, ..., s_k\} \)
- Outputs symbols: \( x \in \{o_1, ..., o_m\} \)
- Starting probability \( P(Y_1 = y_2) \)
  - Specifies where the sequence starts
- Transition probability \( P(Y_i = y_i | Y_{i-1} = y_{i-1}) \)
  - Probability that one state succeeds another
- Output/Emission probability \( P(X_i = x_i | Y_i = y_i) \)
  - Probability that word is generated in this state

\( \Rightarrow \) Every output-state sequence has a probability

\[
P(x, y) = P(x_1, ..., x_n, y_1, ..., y_l) \prod_{i=2}^{l} P(x_i | y_i) P(y_i | y_{i-1}) = P(y_1) P(x_1 | y_1) \prod_{i=2}^{l} P(x_i | y_i) P(y_i | y_{i-1})
\]
Estimating the Probabilities

- Fully observed data:
  - input/output sequence pairs
  
  \[ P(Y_i = a | Y_{i-1} = b) = \frac{\text{# of times state } a \text{ follows state } b}{\text{# of times state } b \text{ occurs}} \]
  
  \[ P(X_i = a | Y_i = b) = \frac{\text{# of times output } a \text{ is observed in state } b}{\text{# of times state } b \text{ occurs}} \]

- Smoothing the estimates:
  - See Naïve Bayes for text classification

- Partially observed data (\( Y_i \) unknown):
  - Expectation-Maximization (EM)

\[
\begin{align*}
P(X_i = a | Y_i = b) & = \frac{\text{# of times output } a \text{ is observed in state } b}{\text{# of times state } b \text{ occurs}} \\
\end{align*}
\]

HMM Prediction (Decoding)

Question: What is the most likely state sequence given an output sequence?

\[
y^* = \arg\max_{y \in \{y_1, \ldots, y_l\}} P(x_1, \ldots, x_l, y_1, \ldots, y_l) = \arg\max_{y \in \{y_1, \ldots, y_l\}} \left\{ P(y_1) \prod_{i=2}^{l} P(x_i | y_i) P(y_i | y_{i-1}) \right\}
\]

Going on a trip

- Deal: trip to any 3 cities in Germany -> Italy -> Spain for one low low price

Going on a trip

- Deal: trip to any 3 cities in Germany -> Italy -> Spain for one low low price

<table>
<thead>
<tr>
<th>Country</th>
<th>City options</th>
</tr>
</thead>
<tbody>
<tr>
<td>Germany</td>
<td>Berlin/Munich/Witten</td>
</tr>
<tr>
<td>Italy</td>
<td>Rome/Venice/Milan</td>
</tr>
<tr>
<td>Spain</td>
<td>Madrid/Barcelona/Malaga</td>
</tr>
</tbody>
</table>

Going on a trip

- Deal:
  - Each city \( i \) has an attractiveness score \( c_i \in [0, 10] \)
  - Each flight has a comfort score \( f_{i,j} \in [0, 10] \)
  - Find the best trip!

Going on a trip

- Deal:
  - Each city \( i \) has an attractiveness score \( c_i \in [0, 10] \)
  - Each flight has a comfort score \( f_{i,j} \in [0, 10] \)
  - Find the best trip!
Viterbi Algorithm for Decoding

- Efficiently compute most likely sequence
  \[ \hat{y} = \arg\max_{y \in \{y_1, \ldots, y_l\}} \left\{ P(y_1) \prod_{i=2}^{l} P(x_i | y_i) P(y_i | y_{i-1}) \right\} \]

- Viterbi Algorithm:
  \[ \delta_y(1) = P(Y_1 = y) P(X_1 = x_1 | Y_1 = y) \]
  \[ \delta_y(i+1) = \max_{y \in \{y_1, \ldots, y_k\}} \delta_y(i) P(Y_{i+1} = y | Y_i = y) \]
  \[ P(Y_1) = \begin{array}{c|c|c|c|c|c|c} DET & 0.3 & DET & 0.3 & DET & 0.3 & DET \\
  PRP & 0.3 & PRP & 0.3 & PRP & 0.3 & PRP \\
  N & 0.1 & N & 0.1 & N & 0.1 & N \\
  PREP & 0.1 & PREP & 0.1 & PREP & 0.1 & PREP \\
  V & 0.2 & V & 0.2 & V & 0.2 & V \end{array} \]
  \[ P(Y_i | Y_{i-1}) = \begin{array}{c|c|c|c|c|c|c} DET & 0.01 & DET & 0.01 & DET & 0.01 & DET \\
  PRP & 0.01 & PRP & 0.01 & PRP & 0.01 & PRP \\
  N & 0.48 & N & 0.48 & N & 0.48 & N \\
  PREP & 0.01 & PREP & 0.01 & PREP & 0.01 & PREP \\
  V & 0.01 & V & 0.01 & V & 0.01 & V \end{array} \]

Example:

Starting from the Viterbi example provided, we can see the probabilities associated with each word in the sentence. The Viterbi algorithm helps in decoding the most likely sequence of words by considering the probabilities of transitioning from one word to another. The table above shows the probabilities for each word type, such as DET (Determiner), PRP (Preposition), N (Noun), PREP (Preposition), and V (Verb). The Viterbi algorithm calculates the most probable sequence of these words based on the transition probabilities and emission probabilities.