CS598
Machine Learning in Computational Biology
(Lecture 3: Sequence - part 2)

Professor Jian Peng
Teaching Assistant: Rongda Zhu
Sequence data

- Protein/DNA sequence
- Generative and discriminative models for sequences
- Deep learning
Central Dogma

- From simple polymers to more complex polymers
- **DNA** stores the genetic information
- **RNA** is the intermediate molecule for protein synthesis
- **Proteins** carry out the biochemical functions in cell
Given a set of DNA/protein sequences with some biological significance, can we find a linear nucleotide pattern that is widespread in the set?
Hidden Markov Model (HMM)
Hidden Markov Model

\[ p(s(i)|s(i - 1)) \quad p(s(i + 1)|s(i)) \]

\[ p(x(i - 1)|s(i - 1)) \quad p(x(i)|s(i)) \quad p(x(i + 1)|s(i + 1)) \]
Hidden Markov Model

Assume we know both observed sequence \( (x) \) and the hidden state sequence \( (s) \), the probability of a M-letter sequence is:

\[
p(x, s) = p(s(1)) \prod_{i=2}^{M} p(s(i) | s(i - 1)) \prod_{i=1}^{M} p(x(i) | s(i))
\]

However, we don’t observe \( s \) in advance but want to know. This can be formulated as a problem to find the best \( s \) given the observed sequence \( x \):

\[
\arg \max_s p(s | x) \propto p(x, s)
\]
Variational Lower Bound

\[
\log p(x; \theta) = \log \sum_s p(x, s; \theta)
\]

\[
\geq \sum_s q(s) \log \frac{p(x, s; \theta)}{q(s)}
\]

\[
= - \sum_s q(s) \log q(s) + \sum_s q(s) \log p(x, s; \theta)
\]

\[
= - \sum_s q(s) \log \frac{q(s)}{p(s|x; \theta)} + \log p(x; \theta)
\]

\[
= -KL(q(s) || p(s|x; \theta)) + \log p(x; \theta)
\]

Kullback-Leibler Divergence

The bound is tight only when \( q(s) = p(s|x; \theta) \)
Optimizing the lower bound

Alternate optimization - Expectation Maximization

E-step

\[ q^t = \arg \max_q - \sum_s q(s) \log q(s) + \sum_s q(s) \log p(x, s; \theta^{t-1}) \]

M-step

\[ \theta^t = \arg \max_\theta \sum_s q^t(s) \log p(x, s; \theta) \]
TODO: derive the EM algorithm for HMMs
Protein structure levels

Primary Structure

Secondary Structure

Tertiary Structure

Quaternary Structure

(from MCB 2009 book)
Protein secondary structure
Protein data bank

Site Record Legend
- ○: BINDING SITE FOR RESIDUE S04 A 2 (SOFTWARE)
- ●: BINDING SITE FOR RESIDUE S04 A 1 (SOFTWARE)
- ●: BINDING SITE FOR RESIDUE S04 A 3 (SOFTWARE)
- ○: BINDING SITE FOR RESIDUE 03P A 1023 (SOFTWARE)

DSSP Legend
- T: turn
- E: beta strand
- G: 3/10-helix
- B: beta bridge
- S: bend
- H: alpha helix
Protein secondary structure prediction

\[ x \in \Sigma_{AA}^L \]

\[ y \in \Sigma_{SS}^L \]

\[ \Sigma_{AA} = \{AminoAcids\} \]

\[ \Sigma_{SS} = \{Helix, Beta, Coil\} \]
Other sequence labeling problems

Part of Speech

Article  Noun  Verb  Preposition  Article  Noun
The cat  sat  on  the  mat.

Handwriting Recognition

Foreign minister.  →  FOREIGN MINISTER.

Speech Recognition

THE SOUND OF
Discriminative models for sequence labeling

The goal is to build a distribution $p(y|x; \theta)$
Discriminative vs Generative Models

Verbally, generative models generate data.

\[ \{x, y\} \sim p(x, y; \theta) \]

Discriminative models make prediction from data

\[ y \sim p(y|x; \theta) \]
Sequence labeling as Classification

Combinatorial explosion when label sequences are long
First approximation

Assumption 1: individual labels are independent.

$$p(y|x; \theta) = \prod_{i=1}^{L} p(y(i)|x; \theta)$$
Naive Bayes Model

Assumption 2: Input data $x$ are generated from a distribution based on label $y$.

$$p(y(i)|x) = \frac{p(y(i)) \prod_{j=1}^{L} p(x(j)|y(i))}{\sum_{c \in \text{Class}} p(c) \prod_{j=1}^{L} p(x(j)|c)}$$
Naive Bayes Model

Assumption 2: Input data $x$ are generated from a distribution based on label $y$.

Q1: How to train a naive bayes model?
Q2: How to make predictions?
Any caveats of this model?
Caveat 1: individual labels are **NOT** independent

Labels on neighboring positions are often identical.

Possible:
- Beta Beta Beta Beta
- Helix Helix Helix Helix

Impossible:
- Helix Beta Helix
- Beta Helix Beta
How to capture the sequential relationship of labels?
Markov Models

Have we seen this model before?

\[ p(y(i + 1) | y(i)) \]
**Hidden Markov Models**

\[ p(s(i)|s(i-1)) \quad p(s(i+1)|s(i)) \]

\[ p(x(i-1)|s(i-1)) \quad p(x(i)|s(i)) \quad p(x(i+1)|s(i+1)) \]
Hidden Markov Models

Assumption: hidden state sequence is output label sequence

\[ p(x, s) = p(s(1)) \prod_{i=2}^{M} p(s(i)|s(i-1)) \prod_{i=1}^{M} p(x(i)|s(i)) \]

Q1: How to train this model?
Q2: How to make predictions?
Comparison (accuracy%) on protein secondary structure prediction

<table>
<thead>
<tr>
<th>Method</th>
<th>Accuracy %</th>
</tr>
</thead>
<tbody>
<tr>
<td>Naive Bayes</td>
<td>57.5</td>
</tr>
<tr>
<td>HMM</td>
<td>72.5</td>
</tr>
</tbody>
</table>
Build discriminative models from generative models

Generative models generate data.
\[
\{x, y\} \sim p(x, y; \theta)
\]

Discriminative models make prediction from data
\[
p(y|x) = \frac{p(x|y)p(y)}{\sum_y p(x|y)p(y)}
\]

Bayes’ Theorem
Caveat 2: the generative assumption
The generative assumption

\[ Y \rightarrow X? \]

\[
\begin{array}{cccccccc}
Y & & & & & & & \\
X & NKEI LDEAYVMAS VDNPHVCRLLGICLTSTVQLITQLMPFGCLLDYVREHKDNIGSQYLL & & & & & & & \\
\end{array}
\]

\[
\begin{array}{cccccccc}
Y & Article & Noun & Verb & Preposition & Article & Noun & \\
X & The & cat & sat & on & the & mat. & \\
\end{array}
\]

\[
\begin{array}{cccccccc}
Y & FOREIGN MINISTER. & \\
X & Foreign & Minister. & \\
\end{array}
\]
Discriminative vs Generative Models

Generative models generate data.

\[
\{x, y\} \sim p(x, y; \theta)
\]

Discriminative models make prediction from data

\[
y \sim p(y|x; \theta)
\]
From Naive Bayes Model to Logistic Regression

\[
p(y|x) = \frac{p(x|y)p(y)}{\sum_y p(x|y)p(y)}
\]

Distributional assumptions

\[
p(y|x; \theta) = \frac{\exp(\theta_y^T \phi(x))}{\sum_{y' \in label} \exp(\theta_{y'}^T \phi(x))}
\]

\(\phi(x)\): feature map function

\(\theta_y\) : label-specific weights
Feature map: Beyond simple observation

Flexibility: add complicated or high-order interactions

\( \phi(\text{NKEILDEAYVMASVDNPHVCRLLGICL}) \)
Training a Logistic Regression Model

\[
p(y|x; \theta) = \frac{\exp(\theta_y^T \phi(x))}{\sum_{y' \in \text{label}} \exp(\theta_{y'}^T \phi(x))}
\]

Maximum likelihood: no closed-form solution

Gradient ascent

1. \( \nabla_{\theta_y} \log p(y|x; \theta) = \phi(x) \delta(y = y') - \sum_{y' \in \text{label}} \phi(x)p(y'|x; \theta) \)

2. \( \theta^{\text{new}} = \theta^{\text{old}} + \alpha \nabla_{\theta} \log p(y|x; \theta) \)
TODO: Gradient methods

(sub-) Gradient methods:
1. Basic principle of convex optimization
2. Subgradient descent

(quasi-) Newton’s method:
1. Newton’s method: use second-order info
2. LBFGS method: fast approximation

Online method:
1. Stochastic gradient descent
2. Learning rate
3. Adagrad
Naive Bayes Model as Logistic Regression

\[
p(y|x) = \frac{p(y) \prod_i p(x(i)|y)}{\sum_{y'} p(y') \prod_i p(x(i)|y')}
= \frac{\exp[\log p(y) + \sum_i \log p(x(i)|y)]}{\sum_{y'} \exp[\log p(y') + \sum_i \log p(x(i)|y')]} 
\]

If we choose \( \theta_y = [\log p(y), \{\log p(x(i) = a|y)\}_{a,i}]^T \)
and \( \phi(x) = [1, \{\delta(x(i) = a)\}_{a,i}]^T \)

then, the naive bayes model is just a non-optimal logistic regression model.
Joint vs conditional models

1. A **joint** model gives probabilities of both input and hidden (or output) and tries to maximize the joint likelihood.
   - Normally, it is easy to train: simply count frequencies
   - Its objective is not directly related to classification accuracy

2. A **conditional** model takes input data and models only the probability of the output label.
   - It aims at maximizing the conditional probability.
   - More closely related to classification accuracy
   - Harder to train
Comparison (accuracy%) on protein secondary structure prediction

<table>
<thead>
<tr>
<th></th>
<th>Naive Bayes</th>
<th>Logistic Regression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy (%)</td>
<td>57.5</td>
<td>80</td>
</tr>
</tbody>
</table>
TODO: LR and the maximum entropy principle
Connecting LR models together

\[ p(y(i)|x; \theta) \propto \exp(\theta^T_{y(i)} \phi(x)) \]

Any ideas?
Adding edge features

\[ \psi(y(i), y(i + 1)) \]

\( \phi(x, y(i)) \)
\( \phi(x, y(i + 1)) \)

Edge feature: Combination of neighboring labels

Node feature: Combination of label and local input
Examples

Edge feature: Correlation between neighboring labels

\[ \psi(y(i), y(i + 1)) \]
Examples

Edge feature: Correlation between neighboring labels

\[ \psi(y(i), y(i + 1)) \]
Examples

Node feature: Correlation between label and local input

\[ \phi(x, y(i)) \]
Examples

Node feature: Correlation between label and local input

\[ \phi(x, y(i)) \]
Conditional Random Fields

\[ p(y|x; \theta, \mu) \propto \exp \left[ \sum_i \theta^T \phi(x, y(i)) + \mu^T \psi(y(i), y(i + 1)) \right] \]

Node potential \quad Edge potential
Conditional Random Fields

\[ p(y|x; \theta, \mu) = \frac{1}{Z_x} \exp\left[ \sum_i \theta^T \phi(x, y(i)) + \mu^T \psi(y(i), y(i + 1)) \right] \]

Normalization factor:

\[ Z_x = \sum_y \exp\left[ \sum_i \theta^T \phi(x, y(i)) + \mu^T \psi(y(i), y(i + 1)) \right] \]

Question 1: How to compute the normalization factor?

Question 2: How to make predictions? \( \arg \max_y p(y|x; \theta; \mu) \)
TODO: How to train CRF models?
Comparison (accuracy%) on protein secondary structure prediction

- Naive Bayes: 57.5%
- Logistic Regression: 65%
- HMM: 72.5%
- CRF: 80%
Comparison (accuracy%) on handwriting recognition (OCR)

Structured Prediction

Naive Bayes | Logistic Regression | HMM | CRF
---|---|---|---
40 | 52.5 | 65 | 77.5
Can we do better?

protein secondary structure prediction

<table>
<thead>
<tr>
<th>Method</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Naive Bayes</td>
<td>65</td>
</tr>
<tr>
<td>Logistic Regression</td>
<td>65</td>
</tr>
<tr>
<td>HMM</td>
<td>70</td>
</tr>
<tr>
<td>CRF</td>
<td>75</td>
</tr>
<tr>
<td>Deep Learning</td>
<td>80</td>
</tr>
</tbody>
</table>
Can we do better?

handwriting recognition

Naive Bayes  Logistic Regression  HMM  CRF  Deep Learning