CS 679: Natural Language Processing

Lectures #8: Maximum Entropy Models

This work is licensed under a Creative Commons Attribution-Share Alike 3.0 Unported License.

Objectives

- Understand how a “vote” can be used in a family of high-powered classifiers
- Introduce Maximum Entropy (MaxEnt) classifiers
- Lay the groundwork for the idea of “Feature Engineering”
- Introduce undirected graphical models
- Prepare for structured prediction tasks

Approaches to Deriving MaxEnt

1. Start with exponential form and maximize Data Likelihood
   - Employ straight-forward differential calculus
   - Leads to constraints
   - Today’s lecture
2. Start with Occam’s Razor, Entropy, Constraints
   - Employ the calculus of variations (Lagrange multipliers)
   - Leads to exponential form
   - Read the Berger tutorial, linked from schedule as optional reading
3. Start with statistics and regression
   - Leads to Logistic Regression
4. Start with the perceptron and back-propagation
   - In stochastic gradient descent, the update is an approximation of the derivative (gradient).

Truth about Vote

- Rather than:
  \[
  \text{vote}(c \mid \text{serve} \mid d) = \sum_i \lambda_i c f_i(d)
  \]

- Actually:
  \[
  \text{vote}(c \mid \text{serve} \mid d) = \sum_i \lambda_i c f_i(d, c)
  \]

Example: feature \(f_i(d, c)\) indicates whether “context-word:jail” exists in \(d\) and \(c\) == “prison sense”

Such features allow for direct dependence on the class label.

Assume Exponential Form

- Strategy:
  - Define the form of the model
  - Define an objective function (opposite of loss function)
  - Maximize the objective
- Make the votes into a probability distribution:
  \[
  \text{P}(c \mid d, \lambda) = \frac{\exp \sum_i \lambda_i c f_i(d, c)}{\sum_c \exp \sum_i \lambda_i c f_i(d, c)}
  \]

  - Makes votes positive.
  - Normalizes votes.

- For weights \(\lambda\), we get a conditional distribution \(\text{P}(c \mid d, \lambda)\)
- This distribution is known by various names:
  - Log-linear, MaxEnt, logistic regression, Gibbs

1. Start with our Vote

- For a word \(w\), we take a weighted vote for each class (in this case, a word sense):
  \[
  \text{vote}(c \mid \text{serve} \mid \text{context of document } d) = \sum_i \lambda_i c f_i(d)
  \]

<table>
<thead>
<tr>
<th>Feature</th>
<th>Food</th>
<th>Jeff</th>
<th>Serve</th>
</tr>
</thead>
<tbody>
<tr>
<td>context jail</td>
<td>1.0</td>
<td>0.3</td>
<td>0.5</td>
</tr>
<tr>
<td>radiant/UF</td>
<td>2.0</td>
<td>0.7</td>
<td>0.3</td>
</tr>
<tr>
<td>object head meals</td>
<td>1.0</td>
<td>1.5</td>
<td>1.5</td>
</tr>
<tr>
<td>object head years</td>
<td>1.5</td>
<td>1.2</td>
<td>1.7</td>
</tr>
<tr>
<td>TOTAL</td>
<td>-2.5</td>
<td>-0.7</td>
<td>2.8</td>
</tr>
</tbody>
</table>
Choose the Parameters

- We want to choose parameters that maximize the (log) conditional likelihood of the labeled data:
  \[
  L(\lambda) = \log P(\text{all labels} \mid \text{all documents}, \lambda) = \log P(c_1, c_2, \ldots, c_n \mid d_1, d_2, \ldots, d_m, \lambda)
  \]

- Assume the data instances are i.i.d. (independent and identically distributed)
  \[
  L(\lambda) = \log \prod_{k=1}^{m} P(c_k \mid d_k, \lambda) = \sum_{k=1}^{m} \log P(c_k \mid d_k, \lambda)
  \]

The Likelihood Function

- Assuming fixed data, the (log) conditional likelihood is a function of the parameters \(\lambda\):
  \[
  L(\lambda) = \log \prod_{i, j} P(c_i \mid d_j, \lambda)
  \]
- If there aren’t many values of class \(c\), it’s easy to calculate:
  \[
  L(\lambda) = \sum_{i, j} \log \exp \left( \frac{\lambda(c_i) f(d_j, c)}{\theta} \right)
  \]
- We can separate this into two components:
  \[
  L(\lambda) = \sum_{i, j} \lambda(c_i) f(d_j, c) - \sum_{i, j} \log \left( \sum_{c} \exp \left( \frac{\lambda(c_i) f(d_j, c)}{\theta} \right) \right)
  \]

Maximizing the Likelihood

- How does the likelihood \(L(\lambda)\) change when we tweak \(\lambda(c)\) (infinitesimally)?
  \[
  \frac{\partial}{\partial \lambda(c)} \log \prod_{k=1}^{m} P(c_k \mid d_k, \lambda) = \sum_{k=1}^{m} \left( \frac{\lambda(c_k) f(d_k, c)}{\theta} \right) - \sum_{k=1}^{m} \left( \frac{\lambda(c_k) f(d_k, c)}{\theta} \right)
  \]
- To maximize, set the derivative to 0.
- Will lead us to data-driven constraints!

The Derivative I: Numerator

For some feature \(j\) and some class \(c\),

\[
\frac{\partial}{\partial \lambda(c)} N(\lambda) = \frac{\partial}{\partial \lambda(c)} \sum_{i, j} \lambda(c) f(d_j, c) = \sum_{i, j} \lambda(c) f(d_j, c) - \sum_{i, j} \lambda(c) f(d_j, c)
\]

\[
= \sum_{k \in c} f_j(d_k, c)
\]

E.g.: we observed the word “fork” \(j\) near the “food” sense \(c\) of “serve” 3 times (twice in one example and once in another).

The Derivative II: Denominator

For some feature \(j\) and some class \(c\),

\[
\frac{\partial}{\partial \lambda(c)} M(\lambda) = \frac{\partial}{\partial \lambda(c)} \sum_{i, j} \log \sum_{c'} \exp \left( \frac{\lambda(c') f(d_j, c')}{\theta} \right) = \sum_{i, j} \log \sum_{c'} \exp \left( \frac{\lambda(c') f(d_j, c')}{\theta} \right) - \sum_{i, j} \log \sum_{c'} \exp \left( \frac{\lambda(c') f(d_j, c')}{\theta} \right)
\]

\[
= \text{expected (predicted) count of feature } j \text{ in data instances with label } c \text{ according to model } \lambda
\]

Where does this lead us?

\[
\frac{\partial \log P(c \mid d, \lambda)}{\partial \lambda(c)} = \text{empirical count}(f_j, c) \cdot \text{predicted count}(f_j, c, \lambda)
\]

- The optimal parameters are the ones for which each feature’s predicted count equals its empirical count. The optimal distribution is:
  - Always unique (but parameters may not be unique)
  - Always exists (if feature counts come from actual data).

E.g.: the count for feature \(f_j = \text{“context” word jail”} \) and \(c = \text{“prison sense”}:\)

\[
\text{empirical} = 1 \quad \text{predicted} = 1.2
\]
How to Train

- Initialize weights to starting value: $\mathbf{w} = [1]$
- Iterate:
  - Given current weights $\mathbf{w}$, calculate the conditional likelihood of the data (the function to optimize):
    \[ L(\mathbf{w}) = \sum_{i=1}^{N} \log p(x_i | d, \mathbf{w}) \]
  - Compute the gradient of the likelihood at current weights $\mathbf{w}$, i.e., derivative with respect to each feature weight:
    \[ \frac{d}{d\mathbf{w}} L(\mathbf{w}) = \text{empirical count}(f, c) - \text{predicted count}(f, c, \mathbf{w}) \]
  - Optimize the gradient with proven methods to compute new weight vector $\mathbf{w}'$:
    - Adaptive stochastic gradient descent. E.g., Adagrad
  - If stopping criterion is met (e.g., $\Delta \mathbf{w}$ is small), then stop.

Convex Optimization

- This Likelihood function is provably convex!

How to Classify with MaxEnt

$$ \hat{c} = \arg \max_c p(c | d, \mathbf{w}) $$

$$ = \arg \max_c \frac{\exp \sum_c \lambda_c f(c, d)}{\sum_c \exp \sum_c \lambda_c f(c, d)} $$

$$ = \arg \max_c \exp \sum_c \lambda_c f(c, d) $$

$$ = \arg \max_c \lambda_c f(c, d) $$

$$ = \arg \max_c \lambda_c f(c, d) = \arg \max_c \text{vote}(c | d, \mathbf{w}) $$

Primary Differences

Naïve-Bayes
- Trained to maximize joint likelihood of data & classes: $P(C, D)$.
- Features assumed to supply independent evidence.
- Feature weights can be set independently, (usually by counting and smoothing). - Quick!
- Not much knowledge required.

Maxent
- Trained to maximize the conditional likelihood of classes given data: $P(C | D)$.
- Feature weights take feature dependense into account.
- Feature weights must be mutually estimated. Takes time!
- Good compromise between empiricism and rationalism.
- High bias, low variance, esp. when data is sparse.

Why Conditional Models?

Consider this metaphor:
Which would you rather take: a multiple-choice exam or an essay exam?
- Joint, generative models are like essay questions.
- Conditional models are like multiple-choice questions.
- Discuss!

Next

- More insight into MaxEnt
- Connection to Naïve Bayes, Neural Nets
- Discussion of Feature Engineering
- Undirected graphical models
MaxEnt Model (Other notation)

- **Training**
  
  \[
  w = \arg \min_{w \in \mathbb{R}^N} \mathcal{F}(w) = \arg \min_{w \in \mathbb{R}^N} \frac{1}{m} \sum_{i=1}^{m} \log p_{y_i|x_i}
  \]

  \[
  p_{y|x} = \frac{1}{Z(x)} \exp(w^T \Phi(x,y))
  \]

  \[
  Z(x) = \sum_{y \in \mathcal{Y}} \exp(w^T \Phi(x,y))
  \]

- **Classification**
  
  \[
  y = \arg \max_{y \in \mathcal{Y}} p_{y|x} = \arg \max_{y \in \mathcal{Y}} w^T \Phi(x,y)
  \]