Maximum Entropy (logistic regression) models

Sharon Goldwater
31 October 2016

Problems with approaches we’ve seen so far

Naive Bayes classifiers for text categorization, WSD, etc:

• independence assumptions between features can be problematic

PCFGs for parsing:

• context-free (structural) independence assumptions can be problematic
• difficult to model lexical effects (e.g., on PP attachment)

For parsing, more complex grammars can help

• parent annotation or otherwise splitting non-terminals (see J&M 14.5)
  – E.g., replace NP with subcategories NP’S, NP’VP, etc.

• lexicalized rules (see J&M 14.6-14.6.1)
  – E.g., replace NP with subcategories NP(Vinken), NP(cat), etc.

• but grammars get huge, parsers get slow, need very carefully designed smoothing... and we still can’t handle, e.g., unbounded dependencies.

• maybe try something completely different?

A different approach to modeling

• so far, all our models have been generative

• discriminative models can address some of the above issues (although they will introduce others)
Generative probabilistic models

- Model the joint probability \( P(\vec{x}, \vec{y}) \)
  - \( \vec{x} \): the observed variables (what we’ll see at test time).
  - \( \vec{y} \): the latent variables (not seen at test time; must predict).

<table>
<thead>
<tr>
<th>Model</th>
<th>( \vec{f} )</th>
<th>( \vec{y} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Naive Bayes</td>
<td>features</td>
<td>classes</td>
</tr>
<tr>
<td>HMM</td>
<td>words</td>
<td>tags</td>
</tr>
<tr>
<td>PCFG</td>
<td>words</td>
<td>tree</td>
</tr>
</tbody>
</table>

Generative models have a “generative story”

- a probabilistic process that describes how the data were created
  - Multiplying probabilities of each step gives us \( P(\vec{x}, \vec{y}) \).

- Naive Bayes: For each item \( i \) to be classified, (e.g., document)
  - Generate its class \( c_i \) (e.g., \textit{SPORT})
  - Generate its features \( f_{i1} \ldots f_{in} \) conditioned on \( c_i \) (e.g., ball, goal, Tuesday)

Result:

\[
P(\vec{c}, \vec{f}) = \prod_i P(c_i) \prod_j P(f_{ij}|c_i)
\]

Other generative stories

- HMM: For each position \( i \) in sentence,
  - Generate its tag \( t_i \) conditioned on previous tag \( t_{i-1} \)
  - Generate its word \( w_i \) conditioned on \( t_i \)

- PCFG:
  - Starting from S node, recursively generate children for each phrasal category \( c_i \) conditioned on \( c_i \), until all unexpanded nodes are pre-terminals (tags).
  - For each pre-terminal \( t_i \), generate a word \( w_i \) conditioned on \( t_i \).
Inference in generative models

- At test time, given only $\vec{x}$, infer $\vec{y}$ using Bayes’ rule:
  \[ P(\vec{y}|\vec{x}) = \frac{P(\vec{x}|\vec{y})P(\vec{y})}{P(\vec{x})} \]

- So, notice we actually model $P(\vec{x}, \vec{y})$ as $P(\vec{x}|\vec{y})P(\vec{y})$.
  - You can confirm this for each of the previous models.

Discriminative probabilistic models

- Model $P(\vec{y}|\vec{x})$ directly
- No model of $P(\vec{x}, \vec{y})$
- No generative story
- No Bayes’ rule

Discriminative models more broadly

- Trained to discriminate correct vs. wrong values of $\vec{y}$, given input $\vec{x}$.
- Need not be probabilistic.
- Examples: support vector machines, (some) neural networks, decision trees, nearest neighbor methods.
- Here, we consider only one method: Maximum Entropy (MaxEnt) models, which are probabilistic.

MaxEnt classifiers

- Used widely in many different fields, under many different names
- Most commonly, multinomial logistic regression
  - multinomial if more than two possible classes
  - otherwise (or if lazy) just logistic regression
- Also called: log-linear model, single neuron classifier, etc...
WSD as example classification task

- disambiguate three senses of the target word plant
  - $\vec{x}$ are the words and POS tags in the document the target word occurs in
  - $y$ is the latent sense. Assume three possibilities:

<table>
<thead>
<tr>
<th>$y$</th>
<th>sense</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Noun: a member of the plant kingdom</td>
</tr>
<tr>
<td>2</td>
<td>Verb: to place in the ground</td>
</tr>
<tr>
<td>3</td>
<td>Noun: a factory</td>
</tr>
</tbody>
</table>

Defining a MaxEnt model

- Define features $f_i(\vec{x}, y)$ that depend on both observed and latent variables.

\[
\begin{align*}
  f_1 : & \quad \text{POS(tgt)} = \text{NN} \& y = 1 \\
  f_2 : & \quad \text{POS(tgt)} = \text{NN} \& y = 2 \\
  f_3 : & \quad \text{preceding_word(tgt)} = \text{‘chemical’} \& y = 3 \\
  f_4 : & \quad \text{doc_contains(‘animal’)} \& y = 1
\end{align*}
\]

where $\text{tgt}$ is the target word

- Each feature $f_i$ has a real-valued weight $w_i$ (learned in training).
- $P(y|\vec{x})$ is a monotonic function of $\sum_i w_i f_i(\vec{x}, y)$: they go up/down together.

Classification with MaxEnt

- Choose the class that has highest probability according to

\[
P(y|\vec{x}) = \frac{1}{Z} \exp \left( \sum_i w_i f_i(\vec{x}, y) \right)
\]

where

- $\exp(x) = e^x$ (the monotonic function)
- $\sum_i w_i f_i$ is the dot product of vectors $\vec{w}$ and $\vec{f}$, also written $\vec{w} \cdot \vec{f}$.
- The normalization constant $Z = \sum_y \exp(\sum_i w_i f_i(\vec{x}, y'))$

Which features are active?

- Example doc: 

  

  \[
  \begin{align*}
    P(y = 1|\vec{x}) & \quad \text{will have} \quad f_1, f_4 = 1 \quad \text{and} \quad f_2, f_3 = 0 \\
    P(y = 2|\vec{x}) & \quad f_2 = 1 \quad f_1, f_3, f_4 = 0 \\
    P(y = 3|\vec{x}) & \quad f_3 = 1 \quad f_1, f_2, f_4 = 0
  \end{align*}
  \]

- Notice that zero-valued features have no effect on the final probability
- Other features will be multiplied by their weights, summed, then $\exp$. 

Sharon Goldwater  MaxEnt models  31 October 2016
### Feature weights: intuition

- $P(y|x)$ increases when $\sum_i w_i f_i(x, y)$ increases.

- Using the same binary features of the target word $tgt$:
  
  - $f_1$: $\text{POS}(tgt) = \text{NN} \& y = 1$
  - $f_2$: $\text{POS}(tgt) = \text{NN} \& y = 2$
  - $f_3$: $\text{preceding\_word}(tgt) = \text{chemical} \& y = 3$
  - $f_4$: $\text{doc\_contains('animal')} \& y = 1$

- For senses $\{1: \text{member of plant kingdom}; 2: \text{put in ground}; 3: \text{factory} \}$, which weights are likely to be positive? Negative?

### Feature templates

In practice, features are usually defined using **templates**

- $\text{POS}(tgt) = t \& y$
- $\text{preceding\_word}(tgt) = w \& y$
- $\text{doc\_contains}(w) \& y$

- instantiate with all possible POSs $t$ or words $w$ and classes $y$
- usually filter out features occurring very few times
- templates can also define real-valued or integer-valued features

- NLP tasks often have a few templates, but 1000s or 10000s of features

### Training the model

- Given annotated data, choose weights that make the labels most probable under the model.

- That is, given items $x^{(1)} \ldots x^{(N)}$ with labels $y^{(1)} \ldots y^{(N)}$, choose
  
  $$\hat{w} = \arg\max_{\bar{w}} \sum_j \log P(y^{(j)}|x^{(j)})$$

- called **conditional maximum likelihood estimation** (CMLE)

- Like MLE, CMLE will overfit, so we use tricks (**regularization**) to avoid that.
The downside to MaxEnt models

- Supervised MLE in generative models is easy: compute counts and normalize.
- Supervised CMLE in MaxEnt model not so easy
  - requires multiple iterations over the data to gradually improve weights (using gradient ascent).
  - each iteration computes $P(y^{(j)}|x^{(j)})$ for all $j$ and each possible $y^{(j)}$.
  - not too bad when there are only a handful of classes $y$. BUT...

MaxEnt for parsing

- Now we have $y =$ parse tree, $x =$ sentence.
- Features can mirror parent-annotated/lexicalized PCFG:
  - counts of each CFG rule used in $y$
  - pairs of words in head-head dependency relations in $y$
  - each word in $x$ with its parent and grandparent categories in $y$.
- Note these are no longer binary features.

Global features

- Features can also capture global structure. Ex (Charniak and Johnson, 2005):
  - length difference of coordinated conjuncts
  - $\text{cat} = c$ & $\text{len} = \ell$ & $\text{end} \_\text{sent} \& \text{before} \_\text{punc}$
  - i.e., are heavy (long) noun phrases at the end of the sentence?

  $$\text{I gave him the book about NLP}$$
  vs
  $$\text{I gave the book about NLP to him}$$

Global features
Features for parsing

- Altogether, Charniak and Johnson (2005) use 13 feature templates
- with a total of 1,148,697 features
- and that is after filtering out features occurring fewer than five times

Training a MaxEnt parser (naively)

For each iteration $i$
  For each sentence $x$
    For each parse tree $y$
      extract features
      compute probabilities

$\Leftarrow 10s$
$\Leftarrow 10,000s$
$\Leftarrow 1000s-10,000s$
$\Leftarrow \text{millions}$

ACK!
### Discriminative re-ranking

- Instead of learning to discriminate best parse from all parses, discriminate from other plausible parses.

- Use generative probabilistic parser to produce $n$-best list of parses.

- Train MaxEnt model to re-rank the list.
  - now only (say) $n = 50$ parses instead of thousands.
  - One important feature not mentioned earlier: the log prob of the parse under the generative model!

### Parser performance

- $F_1$-measure (from precision/recall on constituents) on WSJ test set:
  - standard PCFG $\sim$80% \(^1\)
  - lexicalized PCFG (Charniak, 2000) 89.7%
  - re-ranked LPCFG (Charniak and Johnson, 2005) 91.0%

\(^1\)Figure from (Charniak, 1996): assumes POS tags as input

### Summary of model types

- Probabilistic generative models:
  - model $P(x, y)$ by assuming a generative process
  - easy to train but independence assumptions can be problematic

- Probabilistic discriminative models:
  - model $P(y|x)$ only, have no generative process
  - can incorporate arbitrary local/global features, including correlated ones
  - much slower to train; may require pre-filtering of alternatives ($n$-best list)

- Both kinds of model are currently in use for many NLP tasks.
  - Classification, parsing (discussed here)
  - Sequence models: e.g., taggers that incorporate spelling features.
References


