Efficient Classification with Conjunctive Features

Naoki Yoshinaga and Masaru Kitsuregawa / Institute of Industrial Science, University of Tokyo

Efficiency-accuracy dilemma in feature-based classifiers

**Conjunctive features** (e.g., n-grams) play an important role in obtaining accurate classifiers in many NLP tasks. Ex) Log-linear model (LLM) w/ binary features

\[ p(y | x) = \frac{1}{Z(x)} \exp \sum_{i} w_i f_i(x, y), \quad w_i \in \mathbb{R} \]

\[ f_p(x, y) = \wedge_{i \in P} f_i(x, y); \text{conjunctive feature} \]

But if we use many conjunctive features, the classification becomes very slow.

\[ O(|x|) \Rightarrow O(x^d) \] (or less) conjunctions of primitive features in \( x \)

Current approaches

There are two ways to speeding up the classification with conjunctive features.

- **Polynomial kernel** implicitly expresses conjunctive features, but its computation is heavy in practice
  \[ O(|SV| |x|); |SV| >> |x| \] [1].

- **L1-regularized LLM** shrinks the feature space [2], but frequent features are likely to survive.

Efficient classifier with feature sequence trie

We speed up a classifier trained with many conjunctive features.

**Idea:**

Pre-compute weight \( W_c \) for some feature vector \( x_c \) and use it to obtain weight \( W \) of \( x \supset x_c \)

\[ W(x \supset x_c) = \left( \sum_{i \in C} w_i \right) \cdot \prod_{i \in C} f_i(x) \]

\[ W(x \supset x_c) = \left( \sum_{i \in C} w_i \right) \cdot \prod_{i \in C} f_i(x) \]

Obtain source feature vectors from actual data, sort features in them by frequency, and store weights of the prefix feature vectors into a trie.

**Solution:**

Obtain source feature vectors from actual data, sort features in them by frequency, and store weights of the prefix feature vectors into a trie.

Time complexity:

\[ O(|x_c| + |x^d| - |x^d_+|) \] remaining weights (sparse)

Evaluation: dependency parsing

Results of Japanese dependency parsing

<table>
<thead>
<tr>
<th>Model</th>
<th>SVM</th>
<th>L1-LLM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conj. degree</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>( F^d )</td>
<td># feat.</td>
</tr>
<tr>
<td></td>
<td>( x^d )</td>
<td>ave. feat.</td>
</tr>
<tr>
<td>Dep. Acc. (%)</td>
<td>88.29</td>
<td>90.93</td>
</tr>
<tr>
<td>Kernel</td>
<td>baseline</td>
<td>0.003</td>
</tr>
<tr>
<td>Parse [ms/sent.</td>
<td>0.015</td>
<td>0.093</td>
</tr>
</tbody>
</table>

- Kyoto Corpus; Sassano’s Shift/reduce parser
- 3,258,313 sentences (news article) were parsed to build fstrie (weight calculation took 1 hour)

Classification time as a function of fstrie size

- Feature vectors are likely to follow Zipf’ law (cf. [3])
- The nodes in fstrie are pruned according to their probability and impact on computation reduction

What’s next?

- Evaluation in other tasks (try the code at http://www.tkl.iis.u-tokyo.ac.jp/~ynaga/)
- Application to structured prediction (CRF)