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

$$p(y \mid x) = \frac{1}{Z(x)} \exp \underbrace{\sum_i w_i f_i(x, y)}_{\text{Weight}}, \quad w_i \in \Re$$

$f_F(x, y) = \wedge_{f_i \in F} f_i(x, y)$: *conjunctive feature*

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

$O(|x|) \Rightarrow O(|\underline{x^d}|)$ *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-reguralized 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 $x \supset x_c$

ex) $x = \{f_1, f_2, f_3, f_4\}$ $(x^2 = \{f_1, f_2, f_3, f_4, f_{1,2}, f_{1,3}, f_{1,4}, f_{2,3}, f_{2,4}, f_{3,4}\}), x_c = \{f_1, f_2, f_3\}$

weight vector

			w_4
		w_3	$w_{3,4}$
	w_2	$w_{2,3}$	$w_{2,4}$
w_1	$w_{1,2}$	$w_{1,3}$	$w_{1,4}$

$W_{\{1,2,3\}}$

weight computation

$$W = w_1 + w_2 + w_{1,2} + w_3 + w_{2,3} + w_{1,3} + w_4 + w_{3,4} + w_{2,4} + w_{1,4}$$

w/ pre-computed weight $W_{\{1,2,3\}}$

$$W = \boxed{W_{\{1,2,3\}}} + w_4 + w_{3,4} + w_{2,4} + w_{1,4}$$

 # features checked 10 > 7 (= $\boxed{3}$ + 4)

Issue:
- ✓ Which x_c should we pre-compute weight for?
- ✓ How to find an optimal x_c for input x?

actual data in the task → *source feature vectors* $\{f_1, f_2\}$, $\{f_1, f_3, f_4\}$, $\{f_1, f_3, f_5\}$, $\{f_2, f_6\}$ → *feature sequence trie (fstrie)*

$\{f_1, f_3\} \updownarrow W_{\{1,3\}}$; root; f_1, f_2; f_2, f_3, f_6; f_4, f_5; *frequent (coarse)* ↓ *rare (fine)*

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: $O(|x_c| + \underbrace{|x^d| - |x_c^d|}_{\text{remaining weights (sparse)}})$

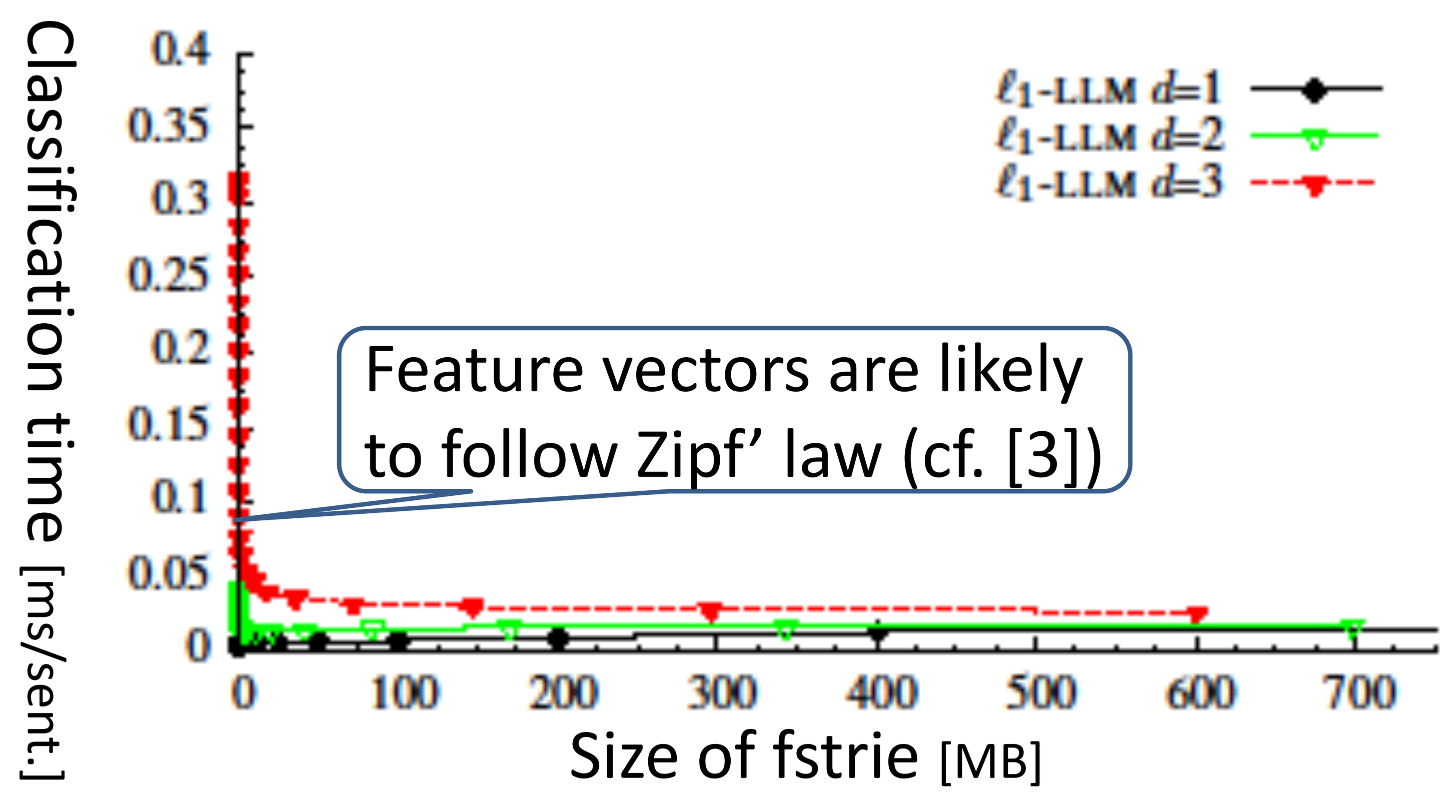
Evaluation: dependency parsing

Results of Japanese dependency parsing

Model		SVM		L1-LLM	
Conj. degree		1	3	1	3
$\lvert F^d \rvert$ # feat. ($\times 10^3$)		39.7	26194.4	9.3	129.5
$\lvert x^d \rvert$ ave. # feat.		27.3	3286.7	26.5	2088.3
Dep. Acc. (%)		88.29	90.93	88.22	90.71
Classify [ms./sent.]	kernel	13.480	10.945	NA	NA
	baseline	0.003	0.345	0.004	0.314
	w/ fstrie	NA	0.079	NA	0.027
Parse [ms./sent.]		0.015	0.093	0.016	0.040

- ✓ 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



* 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)

References

[1] Kudo and Matsumoto: Fast method for kernel-based text analysis. ACL '03.

[2] Gao et al. A comparative study of parameter estimation methods for statistical natural language processing. ACL '07

[3] Chuang et al. Power-law relationship and self-similarity in the itemset support distribution: analysis and applications. The VLDB Journal.