203-5in

簡潔データ構造を用いた高速かつ省メモリな木カーネルの学習

Fast and Memory-Efficient Kernel Learning for Trees with Succinct Data Structures

木村 大翼　　鹿島 久嗣
Daisuke Kimura　　Hisashi Kashima

東京大学大学院情報理工学系研究科
Graduate School of Information Science and Technology, The University of Tokyo

The kernel method is a promising approach to analyzing structured data such as sequences, trees, and graphs and Support Vector Machine (SVM) is a powerful classifier for kernel methods. However, applying SVM to large-scale data is challenging because the time complexity of SVM training scales quadratically with the number of training data. In this paper we propose a fast and memory-efficient SVM training algorithm for a tree kernel by utilizing cutting plane algorithm and XBW, which is a succinct data structure for a labeled tree. The time complexity of our proposed algorithm scales linearly with the number of training data and the space complexity is asymptotic to the information-theoretic lower bound of labeled trees.

1. はじめに

1.1 構造データに対するカーネル法

現実世界において様々なデータが配列, 木構造, あるいはグラフ構造といった非ベクトル型のデータとして表現される. 例えば配列データとしては, バイオインフォマティクス [Durbin 98] の分野における DNA や RNA や自然言語処理 [Manning 99] におけるテキストがあり, 木構造としては同分野の構文解析木や Web データの HTML や XML [Abiteboul 00] といった半構造データがある. さらにグラフ構造 [Kramer 01] として表現されるものとしては化合物がある. これら配列, 木構造, グラフ構造といったいわゆる構造データを対象とした解析手法が近年盛んに研究されている.

これら構造データを対象とした解析手法における有望なアプローチの一つにカーネル法 [Schölkopf 02] がある. カーネル法はカーネル関数と呼ばれる 2 つのデータの内積を経由してデータにアクセスする. データの内積はある種の類似度と捉えることができるので, データ間に何らかの類似度が定めることによって非ベクトル型のデータにもカーネル法を用いることができる. 構造データに対するカーネル関数の設計の代表的な枠組みとして畳み込みカーネル [Haussler 99] がある. 畳み込みカーネルにおいては構造データは陰に部分構造に分解され, カーネル関数はこれら部分構造間のカーネル関数の和として定義される. 構造データを対象としたカーネル関数の設計においては, 部分構造の表現力とカーネル関数の効率的な計算アルゴリズムのバランスを上手くとることが必要不可欠であり, 様々なカーネル関数が文字列 [Lodhi 02, Leslie 02], 木構造 [Collins 01, Kashima 02, Aiolli 09], グラフ構造 [Kashima 03, Gärtner 03] などに対して提案されてきた. これらはカーネル法の代表的な学習器であるサポートベクトルマシン (SVM) [Vapnik 95] に用いられ, 上記の自然言語処理やバイオインフォマティクスなどの分野において高い予測精度を示し大きな成功を収めている.

1.2 カーネル関数を用いた SVM 学習の問題点

カーネル関数と SVM を組み合わせることで得られた学習器が高い予測精度を示す一方, その学習には一般に $O(n^2)$ の計算量が必要である. ここで n は訓練データの数である. そのためカーネル関数を用いた SVM 学習は大規模データに対して用いることが非常に困難であるという問題点がある. この問題に対処するために様々な計算アルゴリズムが提案されてきた [Joachims 99, Platt 99, Joachims 06, Shalev-Shwartz 07, Hsieh 08, Yu 08, Severyn 11].

本論文では特に木構造に対するカーネル関数を用いた場合の SVM 学習の高速化を扱い, Severyn と Moschitti らの枠組み [Severyn 11] に着目する. 彼らは Collins らの木カーネル [Collins 01] を用いた SVM 学習問題に対し, 切断法を用いることで双対問題を高速に計算するアルゴリズムを提案した. また木構造が有する構造を利用し, 予め木構造をまとめて DAG 化することによってさらなる高速化を行った. しかしながら理論的な最悪計算量は依然として $O(n^2)$ である.

1.3 本論文の貢献

本論文では部分パス木カーネル [Kimura 12] を用いた SVM 学習問題に対し, Severyn らの切断法と簡潔データ構造 XBW を用いることで高速かつ省メモリな計算アルゴリズムを提案する. この木カーネルは Collins らの木カーネルを含む既存の木カーネルと同等以上の予測精度をもつことが示されている [Kimura 12]. 本論文の貢献を以下にまとめる.

1. 既存の木カーネルと同等以上の予測精度をもつ部分パス木カーネルの SVM 学習問題に対し, 提案アルゴリズムの時間計算量はデータ数について線形である.
2. 高速化のために必要なメモリはラベル付き木の情報論的下限に漸近するため極めて省メモリである.
3. 簡潔データ構造を機械学習アルゴリズムに取り入れる初の試みである. 簡潔データ構造は大規模データを圧縮したまま高速に扱うことが可能なため, 大規模データを扱う機械学習に対する有望なアプローチとなりうる.

2. 部分パス木カーネル

本論文では木構造はラベル付きの根付き木 T を考える. T において各ノードのラベルはラベル集合 Σ から成り, 枝にラベルはないものとする. また T のノードの数とラベル集合の大きさをそれぞれ $|T|, |\Sigma|$ とする.

本論文では特に Kimura らによって提案された部分パス木カーネル [Kimura 12] を扱う. 部分パスとは木の根から各葉

連絡先: Daisuke_Kimura@mist.i.u-tokyo.ac.jp

までのパスの部分文字列として定義される. 図 1(a) のラベル付き木に対する部分パスを図 1(b) に示す. この部分パスを用いて 2 つの木 T_1 と T_2 間における部分パス木カーネルは以下のように定義される.

$$K(T_1, T_2) \equiv \sum_{p \in P} \lambda^{|p|} \mathrm{num}(T_{1_p}) \mathrm{num}(T_{2_p}) \quad (1)$$

ここで P は T_1 と T_2 における全部分パス集合であり, $\mathrm{num}(T_{1_p})$ と $\mathrm{num}(T_{2_p})$ はそれぞれ T_1 と T_2 における部分パス $p \in P$ の出現回数である. また λ $(0 < \lambda \leq 1)$ は減衰パラメータである.

後のために式 (1) を以下のように表現する.

$$K(T_1, T_2) \equiv \sum_{l \in node(T_1)} \sum_{p \in P_{T_1 l}} \lambda^{|p|} \mathrm{num}(T_{2_p}) \delta_{T_1 l p} \quad (2)$$

ここで $node(T)$ は木 T のノード集合であり, P_{Tl} は木 T のノード l から始まる部分パス集合である. また δ_{Tlp} は木 T においてノード l から始まる部分のパス p が存在すれば 1 を, そうでなければ 0 をとる関数である.

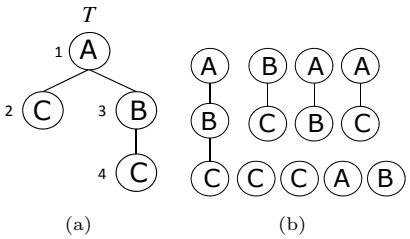

図 1: (a) ラベル付き木. (b) Kimura らの部分パス.

3. 簡潔データ構造

簡潔データ構造とは, データの大きさが情報論的下限に漸近しつつ, 索引を利用して高速な操作が可能であるデータ構造である.

定義 1 *(情報論的下限)*
あるデータ構造がとりえる種類が L であるときに $\lg L$ *bits* を情報論的下限と呼ぶ.

情報論的下限はあるデータ構造を表現する際にこれ以上小さくすることができない下限を示している. 例えば, 長さ n のバイナリベクトル $B = \{0,1\}^n$ のとりえる種類は 2^n であるから, B の情報論的下限は n bits となる.

従来のデータ圧縮の目的がデータの大きさを小さくすることであり, データを利用する際にデータをすべて復元する必要がある. 一方で簡潔データ構造はデータの大きさを小さく保ちつつ, データを利用する際には復元を必要とせずに高速な操作を行うことが可能である. 簡潔データ構造は Jacobson [Jacobson 88] によって初めて提案され, その後順序集合 [Raman 02], 文字列 [Grossi 03], 順序木 [Sadakane 10], ラベル付き木 [Ferragina 05] など様々な離散データに対して提案されている.

3.1 XBW

本節ではラベル付き木に対する簡潔データ構造である XBW [Ferragina 05] について述べる. XBW は文字列に対する可逆変換である BW 変換 [Burrows 94] をラベル付き木に対して拡張したものと見なすことができ, 5. 章で提案する高速かつ省メモリな計算アルゴリズムにおいて重要な役割を果たす.

ラベル付き木において各ノードから根までのラベルを繋げた文字列を木の接尾辞と呼ぶ. これらの各接尾辞を辞書順にソートする. ソート後の各接尾辞における先頭ノードの子ノードを並べたものが XBW である. この変換は $O(|T|)$ で行うことができる. 図 1(a) に対する接尾辞と XBW をそれぞれ図に示す.

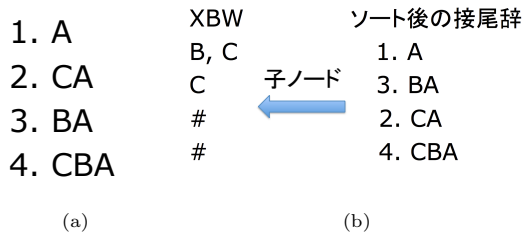

図 2: (a) 図 1(a) に対する接尾辞. (b) 図 1(a) に対する XBW とソート後の接尾辞.

XBW はラベル付き木 T に対する情報論的下限である $2|T| + |T|\ln|\Sigma|$ bits に漸近する大きさで保持することができる. また高速な部分パスの検索が可能であり, 木 T に含まれる長さ m の部分パスの個数を $O(m\ln|\Sigma|)$ 時間で求めることができる. これは従来の木構造に対する接尾辞木 [Shibuya 03] や接尾辞配列 [Kimura 12] と同じオーダーの計算時間であり, これら従来のデータ構造に比べて圧倒的に省メモリである.

4. 切断法による SVM 学習

本章では [Severyn 11] に基づいた切断法を用いた SVM 学習アルゴリズムについて述べる. 双対側での定式化と学習アルゴリズムについて述べた後に計算上のボトルネックを示す.

4.1 切断法による SVM 学習アルゴリズム

[Severyn 11] における切断法は以下に示す Joachims による主問題側での 1-slack SVM に基づいている.

$$\begin{aligned} &\underset{\boldsymbol{w}, \xi \geq 0}{\text{minimize}} \ \frac{1}{2}\|\boldsymbol{w}\|^2 + C\xi \\ &s.t. \ \forall \boldsymbol{c} \in \{0,1\}^n : \xi \geq \frac{1}{n} \sum_{i=1}^{n} c_i (1 - y_i \boldsymbol{w} \cdot \phi(\boldsymbol{x}_i)) \end{aligned} \quad (3)$$

ここで n 次元ベクトル $\boldsymbol{c} = (c_1, ..., c_n) \in \{0,1\}^n$ は通常の n-slack SVM における各制約式 $y_i(\boldsymbol{w} \cdot \phi(\boldsymbol{x}_i)) \geq 1 - \xi_i$ が含まれるか否かを示す役割を果たしている.

[Severyn 11] は式 (3) にラグランジェの未定乗数法を用いることで双対側において,

$$\begin{aligned} &\underset{\boldsymbol{\alpha} \geq 0}{\text{maxmize}} \ \boldsymbol{h}\top\boldsymbol{\alpha} - \frac{1}{2}\boldsymbol{\alpha}\top G\boldsymbol{\alpha} \\ &s.t. \ \boldsymbol{\alpha}\top\mathbf{1} \leq C \end{aligned} \quad (4)$$

と定式化を行い, 式 (4) に対する切断法によるアルゴリズムを提案した (Algorithm 1). ここで $G_{ij} = \boldsymbol{g}^{(i)} \cdot \boldsymbol{g}^{(j)}$, $h_i = d^{(i)}$, $d^{(i)} = \frac{1}{n}\sum_{k=1}^{n} c_{ki}$, $\boldsymbol{g}^{(i)} = -\frac{1}{n}\sum_{k=1}^{n} c_{ki} y_k \phi(\boldsymbol{x}_k)$ である.

$d^{(i)}$ と $\boldsymbol{g}^{(i)}$ はそれぞれ, Algorithm 1 の各ステップにおいて追加される Cutting Plane $(d^{(i)}, \boldsymbol{g}^{(i)})$ を定義するバイアスと劣勾配である (Algorithm 1 の 11, 12 行目).

Algorithm 1 は, 初め制約が全くない状態で最適化を行う (Algorithm 1 の 2 行目). 次に現在の解において最も制約を破っている Cutting Plane を追加する (Algorithm 1 の 8 – 13 行目). その後, 新たに加えた制約と一つ前の制約のペナルティの差を比較し, その差が ε 以内なら終了し, (Algorithm 1 の 15 行目), そうでなければ終了条件を満たすまでアルゴリズムを繰り返す. 定理 1 が示すようにこのアルゴリズムの繰り返しの数は訓練データの数 n に依存しない.

Algorithm 1 切断法による SVM 学習アルゴリズム (双対側)

1: **Input:** $(\boldsymbol{x}_1, y_1)$,...,$(\boldsymbol{x}_n, y_n)$, C, ε
2: $S \leftarrow \emptyset; t = 0;$
3: **repeat**
4: **新たな制約でグラム行列 G を更新**
5: $\boldsymbol{\alpha} \leftarrow$ 式 (4) によって更新
6: $\xi = \frac{1}{C}(\boldsymbol{h}^\top \boldsymbol{\alpha} - \frac{1}{2}\boldsymbol{\alpha}^\top G\boldsymbol{\alpha})$
7: $\boldsymbol{w} = -\sum_{j=1}^{t} \alpha_j \boldsymbol{g}^{(j)}$
/* Cutting Plane を見つける */
8: **for** $i = 1$ to n **do**
9: $c_i \leftarrow \begin{cases} 1 & y_i(\boldsymbol{w} \cdot \phi(\boldsymbol{x}_i) \leq 1) \\ 0 & otherwise \end{cases}$
10: **end for**
11: $d^{(t)} = \frac{1}{n}\sum_{i=1}^{n} c_i$
12: $\boldsymbol{g}^{(t)} = -\frac{1}{n}\sum_{i=1}^{n} c_i y_i \phi(\boldsymbol{x}_i)$
/* 制約を新たに一つ加える */
13: $S \leftarrow S \bigcup (d^{(t)}, \boldsymbol{g}^t)$
14: $t = t + 1$
15: **until** $d^{(t)} + \boldsymbol{w} \cdot \boldsymbol{g}^{(t)} \leq \xi + \varepsilon$
16: **return** $\boldsymbol{\alpha}$

定理 1 *(Algorithm1 の終了までの繰り返しの数 [Severyn 11])*
$R = max_{1 \leq i \leq n} \|\phi(\boldsymbol{x}_i)\|$ とする. *Algorithm 1* は高々

$$max\{\frac{2}{\varepsilon}, \frac{8CR^2}{\varepsilon^2}\}$$

回の繰り返しで終了し, その数はデータ数 n に依存しない.

4.2 切断法による SVM 学習アルゴリズムのボトルネック

本節では前節で述べた切断法による SVM 学習アルゴリズムの計算上のボトルネックについて述べる. Algorithm 1 におけるボトルネックは 8-10 行目の Cutting Plane を求める計算である. 8-10 行目において $\boldsymbol{w} \cdot \phi(\boldsymbol{x}_i)$ はカーネル関数を用いることで

$$\boldsymbol{w} \cdot \phi(\boldsymbol{x}_i) = \sum_{k=1}^{n}(\sum_{j=1}^{t} \frac{1}{n}\alpha_j c_{kj} y_k) K(\boldsymbol{x}_k, \boldsymbol{x}_i) \quad (5)$$

と展開することができる. この式を直接計算すると $O(n)$ 回のカーネル関数の計算が必要となり, この計算をすべてのデータ $\boldsymbol{x}_i (i = 1, , , n)$ に対して行う必要があるため, Cutting Plane を求める計算量は $O(n^2)$ となる. そのため Algorithm 1 の計算量も $O(n^2)$ となる.

この問題に対し, [Severyn 11] は式 (5) のカーネル関数の値を個別に計算するのではなく, 各ステップにおける Cutting Plane を DAG 化してまとめることによって共通する部分の計算を省略することで高速化を図った. しかしながら理論的な最悪計算量は依然として $O(n^2)$ であった.

5. 提案アルゴリズム

本章では 2. 章で述べた部分パス木カーネルに対し, 前節の Algorithm 1 と XBW を用いることでデータ数 n について線形時間であり, なおかつ省メモリであるアルゴリズムを提案する. まず部分パス木カーネルの場合, 式 (5) がデータ $\boldsymbol{x}_i$ と各 Cutting Plane に含まれるデータに共通する部分パスをすべて求める問題に帰着されることを示す. その後帰着後の問題が XBW を用いることでデータ数 n に依存しない計算量で解くことが可能であり, Algorithm 1 がデータ数について線形時間で実行できることを示す.

部分パス木カーネルの場合, 式 (5) は式 (2) を代入することで,

$$\sum_{k=1}^{n}(\sum_{j=1}^{t} \frac{1}{n}\alpha_j c_{kj} y_k) K(\boldsymbol{x}_k, \boldsymbol{x}_i)$$
$$= \sum_{j=1}^{t} \alpha_j \left\{ \sum_{l \in node(x_1)} \sum_{p \in P_{x_1 l}} (\sum_{k=1}^{n} \beta_{kjp} \mathrm{num}(x_k p)) \delta_{x_1 lp} \right\} \quad (6)$$

と展開することができる. ここで $\beta_{kjp} = \frac{1}{n} c_{kj} y_k \lambda^{|p|}$ である. 式 (6) は $\boldsymbol{x}_i$ と j ステップ目の Cutting Plane に含まれるデータとデータ $\boldsymbol{x}_i$ 間の共通する部分パスを求め, それらを t ステップ分足し合わせたものだと見なすことができる. 定理 2 が示すように, この式は XBW を用いることでデータ数 n に依存しない計算量で計算することができる.

定理 2 *(XBW を用いた高速計算)*
木 $\boldsymbol{x}_i (i = 1, ..., n)$ に対して $\tilde{x} = \frac{1}{n}\sum_{i=1}^{n} |\boldsymbol{x}_i|$ とする. 式 *(6)* は *XBW* を用いることによって高々 $O(t\tilde{x}^2)$ の時間計算量, $2n\tilde{x} + n\tilde{x}\ln|\Sigma|$ *bits* に漸近する空間計算量で計算することができる. 特に時間計算量は訓練データ数に依存しない.

証明 1 *(定理 2 の証明)* 余白の都合上省略.

定理 3 *(提案アルゴリズムの時間計算量)*
提案アルゴリズムの時間計算量は訓練データの数について線形である.

証明 2 *(定理 3 の証明)*
定理 *1* より, *Algorithm 1* の繰り返しの回数 $t*$ はデータ数 n に依存しない定数である. よって 4 行目から 15 行における計算量が $O(n)$ であることを示せばよい.

まず定理 *2* より 9 行目は $O(t\tilde{x}^2)$ で計算できるので 8 行目から 10 行目までの *Cutting Plane* を求める計算は $O(n)$ で行うことができる. 4 行目におけるグラム行列 G の更新は $G_{it} = \boldsymbol{g}^{(i)} \cdot \boldsymbol{g}^{(t)}$ の定義より, 一つ前のステップにおける式 *(5)* の結果を直接利用することで定数時間で行うことができる. また 5 行目において式 *(4)* は高々サイズが $t * \times t*$ のヘシアンをもつ二次計画問題であるから, 同様にデータ数 n に依存しない. 7 行目と 12 行目は陽に計算する必要がない. 15 行目の計算は 4 行目と同様に式 *(5)* の結果を利用することで定数時間で実行することができる. 以上より各ステップにおける計算量は $O(n)$ となる.

6. 関連研究

SVM の高速化に関しては様々な研究が行われてきた．まずカーネル関数を線形カーネルに限定し, 主問題側で SVM の最適化問題を解くことによって高速化を実現したものとして [Joachims 06, Shalev-Shwartz 07] などが挙げられる. [Joachims 06] は切断法, [Shalev-Shwartz 07] は確率的勾配法に基づいた最適化アルゴリズムをそれぞれ提案した. しかしながらこれらのアルゴリズムは線形カーネルに限定されているため, 本論文で扱った木カーネルなど構造データにに対するカーネル関数に適用するこてはできない. 次に分解法に基づいたアルゴリズムとして [Joachims 99, Platt 99] が挙げられる. 分解法は双対問題において, 一度に全体を最適化するのではなく小さな部分問題を繰り返し解くことによって最適化を行う手法である. また構造データに対するカーネル関数に対し, 切断法とサンプリングを組み合わせることによって高速化を試みたものとして [Yu 08, Severyn 11] が挙げられる.

7. まとめ

本論文では部分パス木カーネルを用いた SVM 学習問題に対して切断法とラベル付き木に対する簡潔データ構造である XBW を用いた高速かつ省メモリである学習アルゴリズムを提案した. 提案アルゴリズムにおいては, 新たに加える制約を求める際の計算がカーネル関数の線形和であり, 木構造のマッチング問題に帰着されることに着目し XBW によって効率的に計算できることを示した. 提案アルゴリズムの時間計算量ははデータ数に対して線形であり, またカーネル関数の線形和を高速に求める際に必要な空間計算量は訓練データのラベル付き木の情報論的下限に漸近し, 極めて高速かつ省メモリである.

今後の方針としては提案アルゴリズムと既存の SVM 学習アルゴリズムの比較を行い, 実験的にも高速であることを示すこと, 文字列, 木, グラフに対する他のカーネルに対しても本論文の枠組みを利用することによって高速かつ省メモリであるアルゴリズムを構築することが考えられる.

参考文献

[Abiteboul 00] Abiteboul, S., Buneman, P., and Suciu, D.: *Data on the Web: from relations to semistructured data and XML*, Morgan Kaufmann (2000)

[Aiolli 09] Aiolli, F., Martino, G. D. S., and Sperduti, A.: Route Kernels for Trees, in *ICML*, pp. 17–24 (2009)

[Burrows 94] Burrows, M., Wheeler, D. J., Burrows, M., and Wheeler, D. J.: A block-sorting lossless data compression algorithm, Technical report (1994)

[Collins 01] Collins, M. and Duffy, N.: Convolution Kernels for Natural Language, in *NIPS*, pp. 625–632 (2001)

[Durbin 98] Durbin, R., Eddy, S., Krogh, A., and Mitchison, G.: *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*, Cambridge University Press (1998)

[Ferragina 05] Ferragina, P., Luccio, F., Manzini, G., and Muthukrishnan, S.: Structuring Labeled Trees for Optimal Succinctness, and beyond, in *FOCS*, pp. 184–196 (2005)

[Gärtner 03] Gärtner, T., Flach, P., and Wrobel, S.: On Graph Kernels: Hardness Results and Efficient Alternatives, in *COLT* (2003)

[Grossi 03] Grossi, R., Gupta, A., and Vitter, J. S.: High-order entropy-compressed text indexes, in *SODA*, pp. 841–850 (2003)

[Haussler 99] Haussler, D.: Convolution Kernels on Discrete Structures, Technical Report UCSC-CRL-99-10, University of Calfornia in Santa Cruz (1999)

[Hsieh 08] Hsieh, C.-J., Chang, K.-W., Lin, C.-J., Keerthi, S. S., and Sundararajan, S.: A dual coordinate descent method for large-scale linear SVM, in *ICML*, pp. 408–415 (2008)

[Jacobson 88] Jacobson, G. J.: *Succinct static data structures*, PhD thesis (1988)

[Joachims 99] Joachims, T.: Advances in kernel methods, chapter Making large-scale support vector machine learning practical, pp. 169–184, MIT Press (1999)

[Joachims 06] Joachims, T.: Training linear SVMs in linear time, in *KDD*, pp. 217–226 (2006)

[Kashima 02] Kashima, H. and Koyanagi, T.: Kernels for Semi-Structured Data, in *ICML*, pp. 291–298 (2002)

[Kashima 03] Kashima, H., Tsuda, K., and Inokuchi, A.: Marginalized Kernels between Labeled Graphs, in *ICML*, pp. 321–328 (2003)

[Kimura 12] Kimura, D. and Kashima, H.: Fast Computation of Subpath Kernel for Trees, in *ICML*, pp. 393–400 (2012)

[Kramer 01] Kramer, S. and De Raedt, L.: Feature Construction with Version Spaces for Biochemical Application, in *ICML*, pp. 258–265 (2001)

[Leslie 02] Leslie, C., Eskin, E., and Noble, W. S.: The spectrum kernel: a string kernel for SVM protein classification, in *PSB*, pp. 564–575 (2002)

[Lodhi 02] Lodhi, H., Saunders, C., Shawe-Taylor, J., Cristianini, N., and Watkins, C.: Text classification using String Kernels, *J. Mach. Learn. Res.*, Vol. 2, pp. 419–444 (2002)

[Manning 99] Manning, C. D. and Schütze, H.: *Foundations of Statistical Natural Language Processing*, The MIT Press (1999)

[Platt 99] Platt, J. C.: Advances in kernel methods, chapter Fast training of support vector machines using sequential minimal optimization, pp. 185–208, MIT Press (1999)

[Raman 02] Raman, R., Raman, V., and Rao, S. S.: Succinct indexable dictionaries with applications to encoding k-ary trees and multisets, in *SODA*, pp. 233–242 (2002)

[Sadakane 10] Sadakane, K. and Navarro, G.: Fully-functional succinct trees, in *SODA*, pp. 134–149 (2010)

[Schölkopf 02] Schölkopf, B. and Smola, A. J.: *Learning with Kernels*, MIT Press (2002)

[Severyn 11] Severyn, A. and Moschitti, A.: Fast Support Vector Machines for Structural Kernels, in *ECML PKDD*, pp. 175–190 (2011)

[Shalev-Shwartz 07] Shalev-Shwartz, S., Singer, Y., and Srebro, N.: Pegasos: Primal Estimated sub-GrAdient SOlver for SVM, in *ICML*, pp. 807–814 (2007)

[Shibuya 03] Shibuya, T.: Constructing the Suffix Tree of a Tree with a Large Alphabet, *IEICE Trans*, Vol. 86, No. 5, pp. 1061–1066 (2003)

[Vapnik 95] Vapnik, V. N.: *The nature of statistical learning theory*, Springer-Verlag New York, Inc. (1995)

[Yu 08] Yu, C.-N. J. and Joachims, T.: Training structural svms with kernels using sampled cuts, in *KDD*, pp. 794–802 (2008)