2K2-2

k-最近傍グラフの分割による Extreme Multi-label 分類器の学習

Extreme Multi-label Learning via k-nearest Neighbor Graph Partitioning

田頭 幸浩 *1
Yukihiro Tagami

*1ヤフー株式会社
Yahoo Japan Corporation

Web scale classification problems, such as Web page tagging and E-commerce product recommendation, are typically regarded as multi-label classification with an extremely large number of labels. In this paper, we propose GPT, which is a novel tree-based approach for extreme multi-label learning. GPT recursively splits a feature space with a hyperplane at each internal node, considering approximate k-nearest neighbor graph based on the label vectors. We learn the linear binary classifiers using a simple optimization procedure. Experimental results demonstrate the effectiveness of our proposed method.

1. 導入

本稿では，ラベル集合が非常に大きなマルチラベル分類問題（extreme multi-label classification）に取り組む．この問題の具体例としては，Wikipedia の新規ページに対して，カテゴリ集合の中からいくつかの該当するものをタグ付けするタスクが考えられる．また，他の例として，ウェブユーザーに対して，その閲覧ウェブページや検索キーワードを用いて，オンライン広告の候補集合から，関連性のある部分集合を選択するタスクがあげられる．

マルチラベル分類問題のデータセットを $\mathcal{D} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^N$ と表す．ここで N はサンプルの数，$\boldsymbol{x}_i \in \mathbb{R}^M$ は M 次元の特徴量ベクトル，$\boldsymbol{y}_i \in \mathbb{R}^L$ は対応する L 次元のラベルベクトルである．i 番目のサンプルが j 番目のラベルを持つ時，ラベルベクトル $\boldsymbol{y}_i$ の j 次元目は $y_{ij} = 1$ となり，そうでなければ $y_{ij} = 0$ となる．なお，本稿で想定するラベルの種類数 L は 10^4 から 10^6 程度である（表 1 を参照）．マルチラベル学習の目的は，与えられた特徴量ベクトルからラベルベクトルを正しく予測する，分類器 $f : \mathbb{R}^M \to \{0,1\}^L$ を構築することである．

従来の，ラベルごとに独立した二値分類器を構築する，一対他（one-versus-rest）のアプローチでは，ラベル数と同数の二値分類器を学習する必要がある．さらに，このアプローチの予測時には，それぞれのテストサンプルごとに，全ての二値分類器を適用しなくてはならない．そのため，ラベル数が増えるにつれ，この単純な手法では，学習時と予測時の両方において計算コストが高くなってしまう．

この問題に対処するため，これまでにも，いくつかの手法が提案されてきた [Prabhu 14, Jain 16, Jasinska 16]. FastXML [Prabhu 14] は木構造ベースのマルチラベル分類手法である．この手法では，初期化の方法を変更し，複数の木を作成することで，最終的な予測精度を向上させる（アンサンブル学習）．それぞれの木の中間ノードは線形分類器を保持しており，これにより特徴量空間を分割している．この線形分類器は，ランキング指標である nDCG をベースとした損失関数を最適化することで，学習が行われる．予測時には，テストサンプルはルートノードから分類木を辿り，到達した葉ノードに含まれる学習データの，経験ラベル分布を用いて予測を行う．十分にバランスされた木の深さは $O(\log N)$ となり，予測時にテストサンプルに対して適用する線形分類器の数もこれに等しい．この数は通常，全ラベル数 L と比較して非常に小さいため，上記の一対他の手法と比較して，高速な予測が可能である．

連絡先: 田頭 幸浩，yutagami@yahoo-corp.jp

Algorithm 1 一つの分類木の学習

Require: 学習データ: $\mathcal{D}_{train} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^N$

$T \leftarrow$ new tree
T.root $\leftarrow$ GrowTree($\mathcal{D}_{train}$)
return T

procedure GrowTree($\mathcal{D}$)
  $n \leftarrow$ new node
  if $|\mathcal{D}| < MaxInLeaf$ **then** ▷ 葉ノードの作成
    $n.\boldsymbol{y} \leftarrow$ empirical label distribution of $\mathcal{D}$
  else
    $(\mathcal{D}_{\text{left}}, \mathcal{D}_{\text{right}}, \boldsymbol{w}_n) \leftarrow$ SplitNode($\mathcal{D}$)
    $n.\boldsymbol{w} \leftarrow \boldsymbol{w}_n$
    n.left $\leftarrow$ GrowTree($\mathcal{D}_{\text{left}}$)
    n.right $\leftarrow$ GrowTree($\mathcal{D}_{\text{right}}$)
  end if
  return n
end procedure

procedure SplitNode($\mathcal{D}$)
  $G \leftarrow$ ConstructApproximateKNNG($\mathcal{D}$)
  $\boldsymbol{w} \leftarrow$ LearnHyperplane($G, \mathcal{D}$)
  $\mathcal{D}_{\text{left}} \leftarrow \{(\boldsymbol{x}_i, \boldsymbol{y}_i) \in \mathcal{D} : \boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_i > 0\}$
  $\mathcal{D}_{\text{right}} \leftarrow \{(\boldsymbol{x}_i, \boldsymbol{y}_i) \in \mathcal{D} : \boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_i \leq 0\}$
  return $(\mathcal{D}_{\text{left}}, \mathcal{D}_{\text{right}}, \boldsymbol{w})$
end procedure

2. 提案手法

この章では，提案手法について述べる．本稿では，先に報告した手法の概要 [Tagami 17] に加え，より詳細な説明を行う．

上記のように，FastXML はテストサンプルが到達した葉ノードに含まれる学習データの，経験ラベル分布を用いて予

Algorithm 2 一つの分類木を用いた予測

Require: テストサンプル: $\boldsymbol{x}_t$

$n \leftarrow T$.root ▷ 木のルートノードから開始
while n is not a leaf node **do**
  $\boldsymbol{w}_n \leftarrow n.\boldsymbol{w}$
  if $\boldsymbol{w}_n^{\mathrm{T}}\boldsymbol{x}_t > 0$ **then**
    $n \leftarrow n$.left ▷ 左の子ノードをたどる
  else
    $n \leftarrow n$.right ▷ 右の子ノードをたどる
  end if
end while
return $n.\boldsymbol{y}$

測を行う．この予測手順は，葉ノードに対応する特徴量の部分空間内の，全ての学習データを用いた k-近傍法による分類（k-nearest neighbor classifier）とみなすことができる．この観点から提案手法では，分類木の各中間ノードで，可能な限り，ラベル空間における k-最近傍点を保持したままバランスよく学習データを分割することを目指す．言い換えると，ラベル空間における k-最近傍グラフ（k-nearest neighbor graph; KNNG）の分割を繰り返すことにより，分類木を構築する．それゆえ，提案手法を “graph partitioning tree” (GPT) と呼ぶ．

GPT の学習の概要を Algorithm 1 に示す．先に述べたように，GPT は木構造の各ノードで線形分類器を学習し，その分類器により学習データを分割し，木を成長させていく．ノードに含まれる学習データの数が閾値を下回ると，そのノードを葉ノードとし，分割を終了する．以下では，分類木の各ノードに紐づく線形分類器の学習方法について述べる．まず，k-最近傍グラフを作成し，そのグラフを用いて分離超平面を学習する．

以下では，説明のため，分類木の中間ノード n に対応する，学習データのインデックス集合を $\mathcal{I}_n \subset \{1, 2, \ldots, N\}$ と表現する．

分類木の中間ノード n において，ラベルベクトルを用いた k-最近傍グラフを作成する．このグラフの頂点は，1 つの学習サンプルに対応している．j 番目のサンプル点が，i 番目のサンプルのラベル空間における k-最近傍点の集合に含まれる時，i 番目の頂点から j 番目の頂点に有向枝（エッジ）が結ばれる．本稿では，ラベル空間における i 番目のサンプルに対する k-最近傍点の集合を，正規化したラベルベクトル $\boldsymbol{y}/|\boldsymbol{y}|$ の内積を用いて，以下のように定義する．

$$\mathcal{N}_{\boldsymbol{Y}_n}^{(i)} := \underset{S:S\subset\mathcal{I}_n,|S|=k,i\notin S}{\arg\max} \sum_{j\in S} \frac{\boldsymbol{y}_i^{\mathrm{T}}\boldsymbol{y}_j}{|\boldsymbol{y}_i||\boldsymbol{y}_j|}$$

なお，S は要素数が k であるインデックスの集合を表し，$|\boldsymbol{y}_i| = \sum_j y_{ij}$ はサンプルに紐づくラベルの個数を表す．

ラベルベクトル $\boldsymbol{y}$ は通常，疎なベクトルであるため，最近傍点の集合 $\mathcal{N}_{\boldsymbol{Y}_n}^{(i)}$ は転置インデックスを用いて効率的に検索することが可能である．その際，j 番目のラベルを持つデータの数 n_j を用いて，全てのデータ点に対して最近傍点の集合を検索する計算コストは $\sum_{j=1}^{L} n_j(n_j-1)/2$ で見積もられる．しかしながら，ほぼ全てのサンプルに対して紐づくラベルがある場合，n_j は全データ数 N に近づき，上記の計算コストは $\mathcal{O}(N^2)$ に近づく．本稿では，このような多くのサンプルに対して紐づくラベルをヘッドラベルと呼び，逆に少数のサンプルにのみ紐づくラベルをテイルラベルと呼ぶ．言い換えると，ヘッドラベルの n_j は大きく，テイルラベルの n_j は小さい．上記の計算コストの問題に対処するため，本稿ではテイルラベルにのみ着目する．閾値パラメータ n_{th} を用いて，$n_j < n_{th}$ を満たすラベルのみを用いて近似最近傍点の集合を検索する．この単純な近似により，ルートノードに近いノードでは紐づく学習データ数が多いため，テイルラベルのみを用いる．一方，分割を繰り返して葉ノードに近づくと，紐づく学習データの量も減り，全てのラベルを用いて k-最近傍グラフを構築することになる．

上記の手順により得られた $\tilde{\mathcal{N}}_{\boldsymbol{Y}_n}^{(i)}$ を用いて近似 k-最近傍グラフを構築したのち，最小グラフカットを求めることにより，線形分類器の学習を行う．ただし，一般的な最小グラフカット問題 [Shi 00] とは異なり，未知のテストデータに対する予測を行うため，予測時にも利用可能なルールにより，グラフを分割する必要がある．そこで，確率的 k-means クラスタリング [Bottou 95] と同様に，各 i 番目のサンプルに対して，以下の目的関数を逐次最大化することで，分離超平面 $\boldsymbol{w}_n$ を学習する．

$$\sum_{j\in\tilde{\mathcal{N}}_{\boldsymbol{Y}_n}^{(i)}} \log\sigma(c_i\boldsymbol{w}_n^{\mathrm{T}}\boldsymbol{x}_j) + \sum_{j\in S_n^-} \log\sigma(-c_i\boldsymbol{w}_n^{\mathrm{T}}\boldsymbol{x}_j) - \lambda|\boldsymbol{w}_n|_1$$

ここで $\sigma(z) = 1/(1+\exp(-z))$ はシグモイド関数，$S_n^- \subset \mathcal{I}_n$ は中間ノード n に紐づく学習データからランダムに選択したインデックスの集合，λ は正則化パラメータである．c_i は i 番目のサンプルが現在，超平面のどちら側にあるかを表す変数であり，$\boldsymbol{w}_n^{\mathrm{T}}\boldsymbol{x}_i > 0$ の時 $c_i = +1$ であり，そうでなければ $c_i = -1$ を取る．

上記の式の第一項は，i 番目のサンプルとそのラベル空間での近傍点が，分離超平面の同じ側に配置されることを目的としている．逆に，第二項は，全体からランダムに選択された点が，i 番目のサンプルとは異なる側に分類されることを目指している．これは，分類面の片側に多くのサンプルが配置されないようにし，バランス良くデータが分割されることを目的としている．第三項は L_1 正則化項であり，$\boldsymbol{w}_n$ が疎なベクトルで表現されることを意図している．これにより，モデルサイズを小さくする効果がある．

線形分類器 $\boldsymbol{w}_n$ の学習は，AdaGrad [Duchi 11] により学習率を調整した FTRL-Proximal アルゴリズム [McMahan 11] を用いる．

GPT の予測の概要を Algorithm 2 に示す．分類木の各中間ノードに紐づく線形分類器を用いて，どちらの子ノードに進むかを決定する．テストサンプルが葉ノードにたどり着くと，その葉ノードに紐づいたラベルベクトルを用いて予測を行う．複数の木を用いる場合は，それぞれの分類木から得られたラベルベクトルを平均して最終的な予測を行う．

3. 実験

この章では，提案手法の評価実験を行う．まず，データセットと実験設定，比較手法について述べ，その後，実験結果を示す．

3.1 データセット

提案手法を評価するため，5 つの大規模なマルチラベル分類問題のデータセットを用いた．これらのデータセットは，Extreme Classification Repository [Bhatia 16] で配布されたものであり，あらかじめ前処理がされ，学習データとテストデータに分割されている．データセットの統計値を表 1 にまとめた．

表 1: データセットの統計値

データセット	学習データ数 N	テストデータ数 N_{test}	特徴量数 M	ラベル数 L	ラベルごとの平均サンプル数	サンプルごとの平均ラベル数
AmazonCat-13K	1,186,239	306,782	203,882	13,330	448.57	448.57
Wiki10-31K	14,146	6,616	101,938	30,938	8.52	18.64
Delicious-200K	196,606	100,095	782,585	205,443	72.29	75.54
WikiLSHTC-325K	1,778,351	587,084	1,617,899	325,056	17.46	3.19
Amazon-670K	490,449	153,025	135,909	670,091	3.99	5.45

3.2 実験設定と比較手法

提案手法である GPT との比較手法として，木構造ベースの手法である，FastXML [Prabhu 14]，PfastereXML [Jain 16]，PLT [Jasinska 16] を用いた．

手法の評価指標としては，extreme マルチラベル分類やランキング問題で広く用いられている，precision@k ($k \in \{1,3,5\}$) を用いた．

$$\mathrm{P@}k := \frac{1}{kN_{test}} \sum_{i=1}^{N_{test}} \sum_{l=1}^{k} y_{i\pi(l)}$$

なお，$\pi(k) = j$ は，j 番目のラベルが，予測スコア上位 k 番目にあることを表す．

GPT の学習には，全てのデータセットに対して，以下の共通のハイパーパラメータを用いた．学習器の数: 50（FastXML と PfastreXML のデフォルト値），近似最近傍点を検索する際の閾値パラメータ: $n_{th} = 50$，判別面 $\boldsymbol{w}_n$ の学習に用いる近似最近傍点およびランダムサンプル点の数: $|\tilde{\mathcal{N}}^{(i)}_{\mathbf{Y}_n}| = |S^-_n| = 10$，判別面 $\boldsymbol{w}_n$ の最適化時のエポック数: 10，AdaGrad の初期学習率: $\eta_0 = 0.1$，L_1 正則化パラメータ: $\lambda = 4$，葉ノード作成の閾値パラメータ: 10.

FastXML と PfastreXML に対しては，それらの著者が公開している C++の実装 [Bhatia 16] を用いて実験を行った．その際には，著者が指定したハイパーパラメータを用いた．我々の実験では，評価指標として通常の（重み付けされていない）precision@k を用いたため，PfastreXML の傾向スコア（propensity score）としては全てのラベルに対して同じ値を設定した．PLT の結果に対しては，論文中 [Jasinska 16] で報告された，それぞれのデータセットに対してハイパーパラメータをチューニングした結果を用いて比較を行った．

3.3 実験結果

実験結果を表 2 に示す．**太字**で表している値が，全ての手法の中で最良の結果である．提案した GPT は比較した木構造ベースの手法の中で，ほぼ全てのデータセットに対して最良の結果を示した．例えば，WikiLSHTC-325K データセットにおいては，二番目に最良の結果を示した PfastreXML に対して，約 5%高い P@1 を示した．このデータセットにおいて，C++で実装された GPT の予測時間は，1 つの CPU スレッド上で，1 サンプルあたり 0.9 ミリ秒であった．

表 2: 実験結果

データセット		GPT	FastXML	PfastreXML	PLT
AmazonCat-13K	P@1	0.9084	**0.9310**	0.8994	0.9147
	P@3	0.7676	**0.7818**	0.7724	0.7584
	P@5	0.6255	0.6338	**0.6353**	0.6102
Wiki10-31K	P@1	**0.8476**	0.8295	0.8263	0.8434
	P@3	**0.7322**	0.6756	0.6874	0.7234
	P@5	**0.6320**	0.5770	0.6006	0.6272
Delicious-200K	P@1	**0.4746**	0.4320	0.3762	0.4537
	P@3	**0.4165**	0.3868	0.3562	0.3894
	P@5	**0.3871**	0.3621	0.3403	0.3588
WikiLSHTC-325K	P@1	**0.6336**	0.4975	0.5810	0.4567
	P@3	**0.3997**	0.3310	0.3761	0.2913
	P@5	**0.2906**	0.2445	0.2769	0.2195
Amazon-670K	P@1	**0.4236**	0.3697	0.3919	0.3665
	P@3	**0.3725**	0.3332	0.3584	0.3212
	P@5	**0.3384**	0.3053	0.3321	0.2885

4. まとめと今後の課題

本稿ではラベルの種類数が非常に多いマルチラベル分類問題に対して，木構造ベースの分類手法である GPT を提案した．GPT は，ラベルベクトルの k-最近傍グラフを分割することにより，分類木を構築する．グラフの分割には，シンプルな逐次最適化手法を用いており，実装も非常に容易である．複数の大規模なデータセットを用いた実験により，GPT と他の木構造ベースの分類手法を比較し，GPT の有用性を示した．

本稿では，既存手法と予測精度の観点における比較を行ったが，その他にもモデルサイズや予測速度，分類木のバランス度合いなど，更なる分析が求められる．また，今回は木構造ベースの手法に絞って比較を行ったが，その他のアプローチをとる手法との比較も今後の課題である．

5. 謝辞

本研究を進めるにあたり，さまざまなコメントをいただいた，鹿島久嗣教授および鹿島研究室の皆さま，ヤフー株式会社の同僚の皆さまに感謝いたします．

参考文献

[Bhatia 16] Bhatia, K., Jain, H., Prabhu, Y., and Varma, M.: The Extreme Classification Repository, `https://manikvarma.github.io/downloads/XC/XMLRepository.html` (2016), Last Accessed: January 15, 2017

[Bottou 95] Bottou, L., Bengio, Y., et al.: Convergence properties of the k-means algorithms (1995)

[Duchi 11] Duchi, J., Hazan, E., and Singer, Y.: Adaptive Subgradient Methods for Online Learning and Stochastic Optimization, *J. Mach. Learn. Res.* (2011)

[Jain 16] Jain, H., Prabhu, Y., and Varma, M.: Extreme Multi-label Loss Functions for Recommendation, Tag-

ging, Ranking & Other Missing Label Applications, in *KDD* (2016)

[Jasinska 16] Jasinska, K., Dembczyński, K., Busa-Fekete, R., Pfannschmidt, K., Klerx, T., and Hüllermeier, E.: Extreme F-Measure Maximization using Sparse Probability Estimates, in *ICML* (2016)

[McMahan 11] McMahan, H. B.: Follow-the-Regularized-Leader and Mirror Descent: Equivalence Theorems and L1 Regularization., in *AISTATS* (2011)

[Prabhu 14] Prabhu, Y. and Varma, M.: FastXML: A Fast, Accurate and Stable Tree-classifier for Extreme Multi-label Learning, in *KDD* (2014)

[Shi 00] Shi, J. and Malik, J.: Normalized cuts and image segmentation, *IEEE Transactions on pattern analysis and machine intelligence* (2000)

[Tagami 17] Tagami, Y.: Learning Extreme Multi-label Tree-classifier via Nearest Neighbor Graph Partitioning, in *WWW Companion* (2017)