Entire Regularization Paths for Graph Data

Koji Tsuda KOJI.TSUDA@TUEBINGEN.MPG.DE
Max Planck Institute for Biological Cybernetics, Spemannstr. 38, 72076 Tübingen, Germany

Abstract

Graph data such as chemical compounds and XML documents are getting more common in many application domains. A main difficulty of graph data processing lies in the intrinsic high dimensionality of graphs, namely, when a graph is represented as a binary feature vector of indicators of all possible subgraph patterns, the dimensionality gets too large for usual statistical methods. We propose an efficient method to select a small number of salient patterns by regularization path tracking. The generation of useless patterns is minimized by progressive extension of the search space.

1. Introduction

Graphs are general and powerful data structures that can be used to represent diverse kinds of objects. Much of the real world data is represented not as vectors, but as graphs including sequences and trees, for example, biological sequences, semi-structured texts such as HTML and XML, chemical compounds, RNA secondary structures, and so forth. In supervised learning for graph data, one can rely on the similarity measures derived from graph alignment or graph kernels. However, one drawback is that the features used in learning are implicitly defined, and the derived prediction rules are hard to interpret. Another approach is based on *graph mining*, where a set of small graphs (i.e., patterns) is used to represent a graph (Figure 1). The graph mining approach is especially popular in chemoinformatics, where the task is to predict activity values of chemical compounds (Kazius et al., 2006), because it is crucial to understand which parts of the compound contribute to its activity.

For interpretability of prediction rules, it is required to select a small number of salient patterns necessary for a given learning task. A naive approach is to first use a frequent substructure mining algorithm such as gSpan (Yan & Han, 2002) or Gaston (Nijssen & Kok, 2004) to collect frequently appearing patterns, and then apply a conventional feature selection algorithm. Actually, this approach was employed by Kazius et al. (2006), where a linear support vector machine is used for feature selection. A frequent substructure mining algorithm has two parameters: minimum support (*minsup*) and maximum pattern size (*maxpat*). The output of graph mining is the set of all patterns satisfying the following two constraints: (1) the number of edges is less than *maxpat*, and (2) the pattern is included in more than *minsup* graphs in the database.

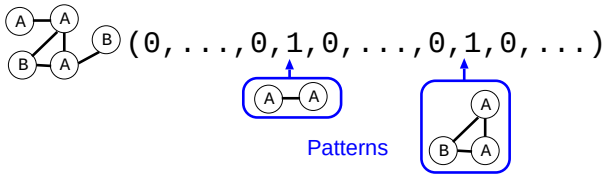

Figure 1. Feature space based on subgraph patterns. The feature vector consists of binary pattern indicators.

In principle, salient features should be selected from the full set of patterns, i.e., the set of all subgraphs of the graphs in the database. However, in the naive approach, one needs to restrict the pattern set a priori using *minsup* and *maxpat* constraints, because it takes a prohibitive amount of time to enumerate the full pattern set. Too much restriction can result in the loss of informative patterns and poor prediction accuracy. Another practical problem is that appropriate values of the parameters are data-dependent, and one has to devise a reasonable procedure to set them.

In this paper, we propose an efficient algorithm for pattern selection based on a regularization path tracking algorithm, called LAR-LASSO (Efron et al., 2004). LAR-LASSO is a forward feature selection algorithm that adds or removes a numerical feature in each step. Instead of numerical features, a pattern is added or removed in our case. The central issue is how to implement the two kinds of feature searching steps contained in the algorithm. We present a method based on the data structure called DFS Code Tree (Yan &

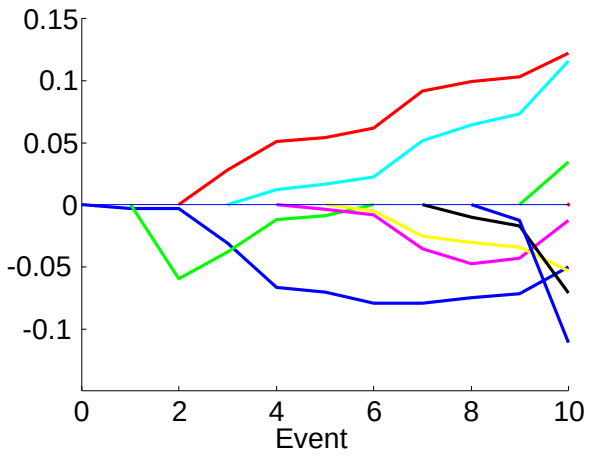

Figure 2. Solution paths for the EDKB dataset. Each curve shows the evolution of the weight parameter of a pattern. A curve can be terminated by an exclusion event. For example, the green curve emerging from the 2nd event disappears at the 7th event.

Han, 2002). Especially, an effective tree pruning condition is proposed, which results in efficient discovery of optimal patterns. Our algorithm draws the entire regularization path in a very high dimensional space of pattern indicators.

2. Path Tracking Algorithms

The entire regularization path (ERP) algorithms offer a principled way of feature selection (Efron et al., 2004; Rosset & Zhu, 2003). The idea is to trace the solution vector of the L1-norm regularized regression algorithm LASSO (Tibshirani, 1996), as the regularization parameter moves from infinity to zero.

Denote by $X \in \Re^{n \times d}$ the design matrix and by $\boldsymbol{y} \in \Re^n$ the target variables. A learning problem with the L1 regularization is written as

$$\boldsymbol{\beta}(\lambda) = \operatorname*{argmin}_{\boldsymbol{\beta}} L(\boldsymbol{y}, X\boldsymbol{\beta}) + \lambda \|\boldsymbol{\beta}\|_1. \tag{1}$$

where $\boldsymbol{\beta} \in \Re^d$ is a weight vector. In this paper, we assume that the loss function is quadratic[1],

$$L(\boldsymbol{y}, X\boldsymbol{\beta}) = \frac{1}{2} \sum_i (y_i - \boldsymbol{\beta}^\top \boldsymbol{x}_i)^2. \tag{2}$$

Due to the sparsity inducing property of the L1 norm, most of the weights are exactly zero. Denote by $\mathcal{A}$ the active set, i.e., the set of indices of nonzero weights in $\boldsymbol{\beta}$. If the regularization strength λ is set to infinity, the active set is empty. As λ is decreased towards zero, the number of elements in the active set gradually increases to d. This trajectory $\boldsymbol{\beta}(\lambda)$ is called the regularization path. Efron et al. (Efron et al., 2004) pointed out that this path is *piecewise linear* if the loss function is piecewise quadratic. Therefore, to draw the entire path, one does not need to take infinitesimally small steps in λ.

The active set changes itself as the regularization parameter decreases. At a given parameter value, either of the two events can happen: (1) inclusion of a feature to the active set, (2) exclusion of a feature from the active set. The ERP algorithm captures each event of feature inclusion and exclusion so that the entire path of solutions is obtained.

To solve the LASSO problem (1) with a fixed regularization parameter, one needs to solve a quadratic program. In machine learning literature, it is often the case that a grid search is performed to find the optimal regularization parameter in terms of, e.g., the cross validation error. To obtain a better regularization parameter, one has to repeat parameter learnings to try many candidate values, which is quite time consuming. By using the path tracking algorithm, one can find exactly the best regularization parameter in a relatively small computational cost.

The path tracking algorithm is shown in Algorithm 1. The derivative of the loss in the pseudo code is described as

$$\nabla L(\boldsymbol{\beta}) = -\sum_{i=1}^{n} (y_i - \boldsymbol{\beta}^\top \boldsymbol{x}_i)\boldsymbol{x}_i. \tag{3}$$

First of all, the first element of the active feature set $\mathcal{A}$ is found by the initial search and the initial direction vector $\boldsymbol{\gamma}$ is set (lines 1 and 2). Next, one has to determine the step length d. As the weight $\boldsymbol{\beta}$ is moved to the direction of $\boldsymbol{\gamma}$, one of the following two events can occur: (1) inclusion of a new feature, or (2) exclusion of an existing feature. The search at line 4 is performed to determine the step size that the first inclusion event occurs. Due to the Karush-Kuhn-Tucker (KKT) condition, every element in the active set $\mathcal{A}$ has the same gradient value (Rosset & Zhu, 2003),

$$\nabla L(\beta + d\gamma)_j = \nabla L(\beta + d\gamma)_{j'}, \quad \forall j, j' \in \mathcal{A},$$

where $\nabla L(\cdot)_j$ denotes the derivative with respect to the j-th variable. This constant is denoted as $\nabla L(\beta + d\gamma)_{\mathcal{A}}$. The direction $\boldsymbol{\gamma}$ is called an equiangular direction.

[1]To apply our algorithm to other loss functions, it is necessary to extend the algorithm considerably. Especially, we need a new pruning condition for the search tree (see Theorem 1).

Table 1. Number of nodes in the search tree and computation time against the maximum pattern size constraint (maxpat) for the CPDB dataset. The minimum support constraint is not used. The computation time is measured on a standard 3GHz PC with 1GB memory. The last row shows the computational cost when no constraints are imposed. In this case, the results of the naive method are not available due to memory overflow.

	Naive		Progressive	
maxpat	#nodes	time	#nodes	time
5	2142	0.51	1171	0.38
6	5717	1.39	2111	0.67
7	14309	3.79	3614	1.36
8	33862	9.93	4936	1.90
9	75814	25.64	6605	2.42
10	161858	65.60	7961	2.80
11	332553	164.20	8613	3.00
12	665202	397.95	8857	3.12
13	1302273	931.61	8964	3.11
∞	N/A	N/A	9088	3.15

We also need to determine the step size d_2 that the first exclusion event occurs (line 5). If $d_1 < d_2$, then the next event is inclusion, otherwise exclusion. Then, the actual step is taken (line 7), the active set is updated (lines 8,9), a new direction is set for a new iteration.

3. Main Search

In applying LAR-LASSO to graph data, one has to solve two kinds of pattern search problems: the initial search (line 1) and the main search (line 4). The main search is called many times while the initial search is done only once.

Let us define some notations. Given a graph database $\mathcal{G} = \{G_i\}_{i=1}^n$, let $\mathcal{T}$ denote the set of all patterns, i.e., the set of all subgraphs included in at least one graph in $\mathcal{G}$. Then, each graph G_i is encoded as a feature vector $\boldsymbol{x}_i = (x_{it})_{t\in T}$,

$$x_{it} = I(t \subseteq G_i),$$

where $t \subseteq G_i$ denotes that t is a subgraph of G_i, and $I(\cdot)$ is 1 if the condition inside is true and 0 otherwise.

In the main search problem, we need to find the optimal pattern t that attains the minimum value of d_t, implicitly defined as

$$|\nabla L(\boldsymbol{\beta} + d_t\boldsymbol{\gamma})_t| = |\nabla L(\boldsymbol{\beta} + d_t\boldsymbol{\gamma})_{\mathcal{A}}|. \tag{4}$$

The right hand side is equal for any element in $\mathcal{A}$ due to the KKT condition. The above equation is rewritten as

$$|\sum_{i=1}^n w_i x_{it} - d_t \sum_{i=1}^n v_i x_{it}| = |\rho_0 - d_t\eta_0| \tag{5}$$

where

$$w_i = (y_i - \boldsymbol{\beta}^\top \boldsymbol{x}_i), \quad v_i = \boldsymbol{\gamma}^\top \boldsymbol{x}_i,$$

and $\rho_0 = \sum_i w_i x_{ik}$, $\eta_0 = \sum_i v_i x_{ik}$ for any $k \in \mathcal{A}$. The equation (5) has the solution,

$$d_t = \min_+ \left\{ \frac{\rho_0 - \sum_i w_i x_{it}}{\eta_0 - \sum_i v_i x_{it}}, \quad \frac{\rho_0 + \sum_i w_i x_{it}}{\eta_0 + \sum_i v_i x_{it}} \right\}. \tag{6}$$

where min_+ stands for the operation of taking minimum among strictly positive elements.

3.1. Search Algorithm

Our search strategy requires a canonical search space in which a whole set of patterns are enumerated without duplication. As the search space, we adopt the DFS code tree (Yan & Han, 2002). The basic idea of the DFS code tree is to organize patterns as a tree, where a child node has a supergraph of the parent's pattern. A pattern is represented as a text string called the DFS (depth first search) code. The patterns are enumerated by generating the tree from the root to leaves using a recursive algorithm. To avoid duplications, node generation is systematically done by rightmost extensions.

For efficient search, it is important to minimize the size of the search space. To this aim, *tree pruning* is crucially important: Suppose the search tree is generated up to the pattern t and denote by d_t^* the minimum d_t among the ones observed so far. If it is guaranteed that $d_{t'}$ of any supergraph t' is not smaller than d_t^*, we can avoid the generation of downstream nodes without losing the optimal pattern.

We propose the following pruning condition.

Theorem 1. *Let us define*

$$b_{\boldsymbol{w}} = \max\left\{ \sum_{w_i<0} |w_i| x_{it}, \sum_{w_i>0} |w_i| x_{it} \right\}.$$

If the following condition is satisfied,

$$b_{\boldsymbol{w}} + d_t^* b_{\boldsymbol{v}} < |\rho_0| - d_t^*|\eta_0|, \tag{7}$$

the inequality $d_{t'} > d_t^*$ holds for any t' such that $t \subset t'$.

4. Experiments

First, we illustrate how our method works using the estrogen receptor dataset from the Endocrine Disruptors Knowledge Base[2]. It contains 131 chemical compounds suspected to work as environmental hormones,

[2] http://edkb.fda.gov

Algorithm 1 The LAR-LASSO algorithm

$\boldsymbol{\beta} = 0$, $\mathcal{A} = \text{argmax}_j |\nabla L(\boldsymbol{\beta})|_j$. ▷ Initial Search
$\boldsymbol{\gamma}_{\mathcal{A}} = -\text{sgn}(\nabla L(\boldsymbol{\beta}))_{\mathcal{A}}$, $\boldsymbol{\gamma}_{\mathcal{A}^C} = 0$.
while $\max |L(\boldsymbol{\beta})| > 0$ **do**
  $d_1 = \min\{d > 0 : |\nabla L(\boldsymbol{\beta} + d\boldsymbol{\gamma})_j| = |\nabla L(\boldsymbol{\beta} + d\boldsymbol{\gamma})_{\mathcal{A}}|, j \notin \mathcal{A}\}$ ▷ Main Search
  $d_2 = \min\{d > 0 : (\boldsymbol{\beta} + \boldsymbol{\gamma})_j = 0, j \in \mathcal{A}\}$.
  Find step length: $d = \min(d_1, d_2)$
  Take step: $\boldsymbol{\beta} \leftarrow \boldsymbol{\beta} + d\boldsymbol{\gamma}$
  If $d = d_1$ then add the variable attaining the minimum to $\mathcal{A}$.
  If $d = d_2$ then remove the variables such that $\beta_j = 0$ from $\mathcal{A}$.
  $C = \frac{1}{2}\sum_i \boldsymbol{x}_{\mathcal{A},i}\boldsymbol{x}_{\mathcal{A},i}^{\top}$
  $\boldsymbol{\gamma}_{\mathcal{A}} = C^{-1}\text{sgn}(\boldsymbol{\beta}_{\mathcal{A}}), \boldsymbol{\gamma}_{\mathcal{A}^C} = 0$.
end while

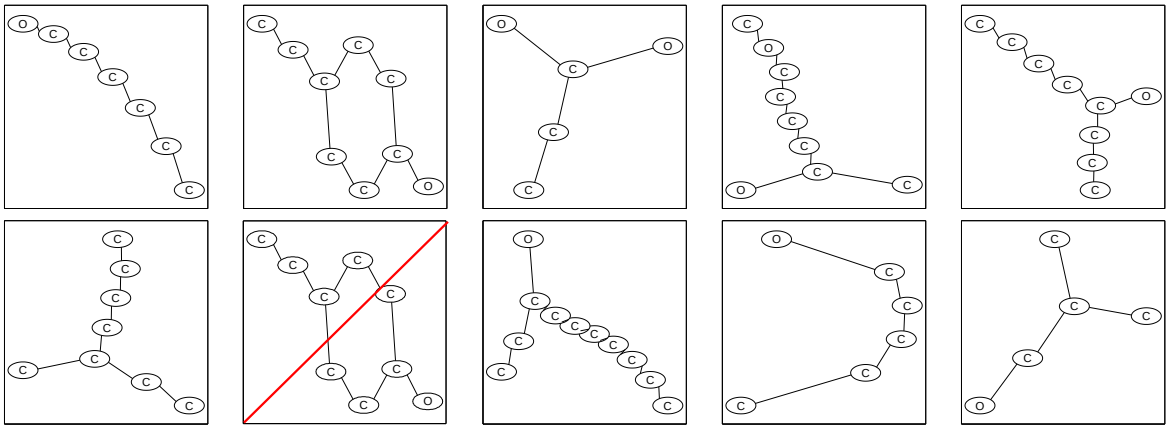

Figure 3. Patterns included or excluded in the first 10 events for the EDKB dataset. The 7th event (indicated by a red line) is an exclusion event.

and our task is to predict its toxicity. The first 10 events of LAR-LASSO are shown in Figure 3. As it is just the beginning of the regularization path, most events are inclusions. However, within this short period, an exclusion happens at the 7th event. The evolution of weight parameters is shown in Figure 2.

The computation time and the size of search space are summarized in Table 1. We compared our method with the naive method which constructs the whole feature space first and then LARS/LASSO is applied.

References

Efron, B., Hastie, T., Johnstone, I., & Tibshirani, R. (2004). Least angle regression. *Ann. Statist.*, *32*, 407–499.

Kazius, J., Nijssen, S., Kok, J., & Ijzerman, T. B. A. (2006). Substructure mining using elaborate chemical representation. *J. Chem. Inf. Model.*, *46*, 597–605.

Nijssen, S., & Kok, J. (2004). A quickstart in frequent structure mining can make a difference. *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 647–652). New York: ACM Press.

Rosset, S., & Zhu, J. (2003). *Piecewise linear regularized solution paths* (Technical Report). Stanford University.

Tibshirani, R. (1996). Regression shrinkage and selection via LASSO. *J. R. Stat. Soc. Ser. B Stat. Methodol.*, *58*, 267–288.

Yan, X., & Han, J. (2002). gspan: Graph-based substructure pattern mining. *Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM)* (pp. 721–724). IEEE Computer Society.