Logistic regression for graph classification

Keywords: logistic regression, graph mining

Nino Shervashidze NINO.SHERVASHIDZE@TUEBINGEN.MPG.DE
Laboratoire d'Informatique de Paris 6, Université Pierre et Marie Curie, Paris, France

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

Abstract

In this paper we deal with graph classification. We propose a new algorithm for performing sparse logistic regression on graphs. Sparsity is required for the reason of interpretability, which is often necessary in domains such as bioinformatics or chemoinformatics. Our method is comparable in accuracy with other methods of graph classification and produces probabilistic output in addition.

1. Introduction

One of the fields where graphs are very commonly used as a means of representation is chemoinformatics, where small molecules can be represented as graphs. Here it is a central issue to predict chemical activities of molecules based on their structure, which is called Quantitative Structure-Activity Relationship (QSAR) Analysis. In this task, it is important to be able to identify a small number of molecular patterns which are responsible for the considered chemical activity: if the set of these patterns is not small, then chemists will not be able to interpret the obtained result. It is often desirable as well to be able to produce a posteriori probability that a molecule will reveal a certain activity. Graph classification methods such as gBoost (Saigo et al., 2008) have been successfully applied on most chemical benchmark datasets, but they do not provide probabilistic output which would be advantageous for a number of applications. Such applications include rejection threshold setting (Chow, 1970), measuring prediction confidence, or covariate shift adaptation (Bickel et al., 2007). We attempt to rectify the situation by proposing an algorithm for performing sparse logistic regression for graph classification.

NIPS 2008 workshop on "Structured Input, Sructured Output" (SISO 2008)

2. Logistic regression on graphs

In this section we expose the logistic regression problem and our solution in a graph classification setting. We are looking for a graph classification method which will *(i)* take into account all substructures occurring a dataset of graphs, *(ii)* find a small set of discriminative substructures, *(iii)* give the a posteriori probability of each graph belonging to each class.

A well-studied method for estimating a posteriori probabilities is logistic regression. In this section we present a column generation algorithm for realising sparse logistic regression on graphs.

In this work we restrict ourselves to connected non-oriented labeled graphs, but the results can be easily extended to more general cases. First we introduce some definitions used in the paper.

2.1. Preliminaries

Definition 1 (Labeled connected graph) *A labeled graph is represented as a 4-tuple $G = (V, E, \mathcal{L}, l)$, where V is a set of vertices, $E \subseteq V \times V$ is a set of edges, $\mathcal{L}$ is a set of labels, and $l : V \cup E \to \mathcal{L}$ is a mapping that assigns labels to the vertices and edges. A labeled connected graph is a labeled graph such that there exists a path between any pair of vertices.*

Definition 2 (Subgraph) *Let $G' = (V', E', \mathcal{L}', l')$ and $G = (V, E, \mathcal{L}, l)$ be labeled connected graphs. G' is a subgraph of G ($G' \subseteq G$) if the following conditions are satisfied: (1) $V' \subseteq V$, (2) $E' \subseteq E$, (3) $\mathcal{L}' \subseteq \mathcal{L}$, (4) $\forall v' \subseteq V', l(v') = l'(v')$ and (5) $\forall e' \subseteq E', l(e') = l'(e')$.*

In graph classification the goal is to learn a decision rule from training examples $\{(G_i, y_i)\}_{i=1}^{l}$, where G_i is a graph and $y_i \in \{-1, +1\}$ is the corresponding class label. Let $\mathcal{S}$ be the set of all subgraphs that occur in at least one training example. Then any graph G_i can be represented as a $|\mathcal{S}|$-dimensional binary vector $\boldsymbol{x}_i$,

$$\boldsymbol{x}_{is} = I(s \subseteq G_i),\ \forall s \in \mathcal{S},$$

where $I(\pi)$ is an indicator function giving 1 if π is true and 0 otherwise.

Every subgraph s is associated with two functions, also called hypotheses or features, which map all $\boldsymbol{x}_i$ to some value in $\{-1, +1\}$:

$$h(\boldsymbol{x}_i, s, 1) = 2\boldsymbol{x}_{is} - 1$$

and its complementary,

$$h(\boldsymbol{x}_i, s, -1) = -2\boldsymbol{x}_{is} + 1.$$

Given these simple features, the aim of the classifier will be to select a small discriminative set of these and find optimal coefficients for respective functions in order to build a decision rule of good quality. For the sake of simplicity, we will suppose that the $T = 2|\mathcal{S}|$ hypotheses are numbered and we will refer to them as h_j, $j \in \{1, \ldots, T\}$.

2.2. Dual formulation of logistic regression

Here we review logistic regression and then we give its dual formulation which can be efficiently combined with subgraph mining.

Logistic regression is a well-studied classification technique. It assumes that y's are generated stochastically as a function of $\boldsymbol{x}$'s and the logarithm of the ratio of the conditional distributions is linear in features of $\boldsymbol{x}$:

$$\log \frac{Pr(y = +1|\boldsymbol{x})}{Pr(y = -1|\boldsymbol{x})} = \sum_{j=1}^{T} \alpha_j h_j(\boldsymbol{x}),$$

α_j being arbitrary real numbers.

This assumption is true for a large number of distributions. Given the last equation, we can express the a posteriori probability $Pr(y|\boldsymbol{x})$ as a simple function of the linear combination $\sum_{j=1}^{T} -\alpha_j h_j(\boldsymbol{x})$, which we will denote $H_\alpha(\boldsymbol{x})$:

$$Pr(y|\boldsymbol{x}) = \frac{1}{1 + e^{-yH_\alpha(\boldsymbol{x})}}.$$

Then the likelihood of the labels is given by

$$\prod_{i=1}^{n} \frac{1}{1 + e^{-y_i H_\alpha(\boldsymbol{x}_i)}}$$

and maximizing this likelihood is equivalent to minimizing its negative logarithm

$$\sum_{i=1}^{n} log(1 + exp(-y_i H_\alpha(\boldsymbol{x}_i))). \qquad (1)$$

As it was said in the introduction, we require sparsity for the reason of interpretability. A well-studied technique for achieving sparsity of a solution is the minimization of a $L1$ norm of the parameter vector α (Tibshirani, 1996), which is equivalent to imposing a Laplace prior over these parameters (Williams, 1995). As we have mentioned earlier, the hypothesis set is complementary-closed, meaning that if a hypothesis h exists, then $-h$ exists as well. Given this property, we can restrict the parameters to non-negativity. Consequently the $L1$ norm will correspond to a simple sum, without absolute value.

Hence, the problem has the following form:

$$\begin{aligned} &\min_{\alpha_j} \sum_{j=1}^{T} \alpha_j + C \sum_{i=1}^{n} g(\xi_i) \\ &\text{subject to}\ \alpha_j \geq 0,\ \forall j \in \{1, \ldots, T\} \qquad (2) \\ &\qquad \xi_i = -y_i \sum_{j=1}^{T} \alpha_j h_j(\boldsymbol{x}_i),\ \forall i \in \{1, \ldots, n\}, \end{aligned}$$

where $g(x) = \log(1 + \exp(x))$ and $C > 0$ is a $L1$ regularization parameter (corresponding to the scale parameter of the Laplace distribution over parameters α_j).

The problem is convex, but in this form it is impossible to solve because of the prohibitive number of variables. For this reason we switch to the corresponding dual problem

$$\begin{aligned} &\max_{\lambda_i}\ -\sum_{i=1}^{n} G(\frac{\lambda_i}{C}) \\ &\text{subject to}\ \sum_{i=1}^{n} \lambda_i y_i h_j(\boldsymbol{x}_i) \leq 1,\ \forall j \in \{1, \ldots, T\}, \end{aligned} \qquad (3)$$

where G represents the Legendre transformation of g, corresponding to binary entropy: $G(x) = x \log x + (1 - x)\log(1 - x)$. After solving the dual problem, the primal solution will be obtained from the Lagrange multipliers.

The dual problem has few variables, but a huge number of constraints. We propose to solve this problem using a column generation technique: beginning with an empty set of constraints, at each iteration the most violated constraint will be identified and added to the constraint set. The most violated constraint is found by the subgraph mining procedure, which is explained in detail in (Saigo et al., 2008). Each time when a new hypothesis is added, the optimal solution is updated by solving the dual problem subject to the updated set of constraints. If there is no more violated constraint, then the current solution is optimal. It is intuitive to suppose that subgraphs found in last iterations will have a negligible impact on the final solution. Therefore we can early-stop the algorithm by introducing a tolerance parameter $\epsilon > 0$ and replacing the stopping criterion by

$$\sum_{i=1}^{n} \lambda_i y_i h_j(\boldsymbol{x}_i) > 1 + \epsilon. \tag{4}$$

We have not yet obtained bounds on the primal objective function in case of early-stopping. However, our experiments show that the difference between primal objective values obtained with $\epsilon = 0$ and $\epsilon = 0.01$ is systematically less than 10^{-6}.

Algorithm 1 Column generation for logistic regression

```
Input: $\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_l, y_l)\}$, $C$, $\epsilon$
Initialize: $j \leftarrow 1$, $CTS \leftarrow \emptyset$, $\lambda_i \leftarrow \frac{C}{2}\ \forall i \in \{1, \ldots, l\}$
    {CTS – set of constraints}
 1: loop
 2:     $h_j \leftarrow h$ maximizing $\sum_{i=1}^{l} \lambda_i y_i h(\boldsymbol{x}_i)$ found by
        the graph mining procedure
 3:     if $\sum_{i=1}^{l} \lambda_i y_i h_j(\boldsymbol{x}_i) < 1 + \epsilon$ then
 4:         break: $T \leftarrow j - 1$, return $H_\alpha = \sum_{j=1}^{T} \alpha_j h_j$
 5:     end if
 6:     $CTS \leftarrow CTS \cup \{\sum_{i=1}^{l} \lambda_i y_i h_j(\boldsymbol{x}_i) \leq 1\}$
 7:     Solve (3) subject to $CTS$
 8:     for $k \in \{1, \ldots, j\}$ do
 9:         $\alpha_k \leftarrow$ Lagrange multiplier associated with the
            $k$'th constraint
10:     end for
11:     $j \leftarrow j + 1$
12: end loop
```

3. Experiments

We have done several experiments to compare the performances of logistic regression with those of gBoost (Saigo et al., 2008) on several chemical datasets. The datasets BZR, DHFR, ER and CPDB contain 300 to 700 graphs each, the average number of nodes and edges does not exceed 41 and 44 respectively, and the labels are binary in all cases. We used 10-fold cross-validation for choosing the C parameter of sparse logistic regression and the ν parameter of gBoost. The possible values were $\{0.25, 0.7, 2, 5, 15, 40, 100\}$ and $\{0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35\}$ for C and ν respectively. We used three evaluation criteria: accuracy (acc), which is the ratio between the number of correctly labeled examples and the number of examples; area under the ROC curve (AUC); the number of subgraphs with non-zero coefficients in the final solution (#sg). Table 1 summarizes obtained results.

Table 1. Comparison of logistic regression (LR) with gBoost (gB)

	BZR		DHFR		ER		CPDB	
	LR	gB	LR	gB	LR	gB	LR	gB
acc.	0.75	**0.78**	**0.86**	0.82	0.82	0.82	**0.80**	0.78
AUC	0.80	**0.81**	**0.89**	0.88	**0.83**	0.79	**0.86**	0.82
#sg	**126**	521	**143**	295	**228**	445	**86**	326

The results show that logistic regression and gBoost have comparable performance in terms of accuracy. Logistic regression seems to produce sparser solutions, which is a good point.

The runtime is comparable for the two methods. For our logistic regression method, as well as for gBoost, graph mining is the most time-consuming part.

4. Conclusion

In this work we considered graph classification. Our contribution is to combine logistic regression with a subgraph mining algorithm by proposing a column generation algorithm (inspired by gBoost) to realise logistic regression on graphs. This algorithm has comparable accuracy with gBoost and in addition it gives probabilistic output.

The proposed method is only applicable to small graphs due to high computational cost. Consequently, a path to follow would be to search for a way to scale up the method. In our opinion, in this direction it would be interesting to consider the projected gradient method instead of column generation: we believe that this would allow to reduce the number of iterations, i.e. the number of graph mining procedure calls.

References

Bickel, S., Brückner, M., & Scheffer, T. (2007). Discriminative learning for differing training and test distributions. *Proceedings of the 24th International Conference on Machine Learning* (pp. 81–88).

Chow, C. K. (1970). On optimum recognition error and reject tradeoff. *IEEE Transactions on Information Theory*, *IT-16*, 41–46.

Saigo, H., Nowozin, S., Kadowaki, T., Kudo, T., & Tsuda, K. (2008). gboost: A mathematical programming approach to graph classification and regression. *Machine Learning (accepted).*

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B*, *58*, 267–288.

Williams, P. (1995). Bayesian regularisation and pruning using a laplace prior. *Neural Computation*, *7*, 117–143.