TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214 02/20 pp638-646
Volume 12, Number 6, December 2007

Semi-Supervised Additive Logistic Regression: A Gradient Descent Solution*

SONG Yangqiu (宋阳秋), CAI Qutang (蔡渠棠), NIE Feiping (聂飞平), ZHANG Changshui (张长水)**

**State Key Laboratory on Intelligent Technology and Systems,
Tsinghua National Laboratory for Information Science and Technology,
Department of Automation, Tsinghua University, Beijing 100084, China**

Abstract: This paper describes a semi-supervised regularized method for additive logistic regression. The graph regularization term of the combined functions is added to the original cost functional used in AdaBoost. This term constrains the learned function to be smooth on a graph. Then the gradient solution is computed with the advantage that the regularization parameter can be adaptively selected. Finally, the function step-size of each iteration can be computed using Newton-Raphson iteration. Experiments on benchmark data sets show that the algorithm gives better results than existing methods.

Key words: semi-supervised; Boosting; graph regularization

Introduction

Generalized additive models (GAMs) have been successfully used for nonlinear problems in statistics[1]. They generate solutions by a sum of functions $F(\boldsymbol{x})=\sum_{i=1}^{m} f_i(x^{(i)})$, which are based on the covariates of the features vector $\boldsymbol{x} = (x^{(1)}, x^{(2)}, ..., x^{(m)})^{\mathrm{T}}$. They are used to solve regression problems using the residual based Backfitting algorithm and to solve other generalized linear models (GLMs) using the Fisher scoring procedure (for the exponential family only) or the Newton-Raphson iteration based Backfitting algorithm[2].

GAMs can be more generally considered to be a set of functions of all the input features: $F(\boldsymbol{x})=\sum_{i=1}^{T}\beta_i f(\boldsymbol{x}, \gamma_i)$. If the "base functions" $f(\boldsymbol{x}, \gamma)$ are not easy to compute or there is a large dictionary of candidate over-complete base (i.e., for a classification tree with k terminal nodes, the number of functions[3] is bounded from $(np)^k \cdot 2^{k2}$), this procedure can be modified to a greedy forward stage-wise algorithm. With fixed $F_{t-1}(x)$, the algorithm finds the new function to be $F_t(x) = F_{t-1}(x)+\beta_t f_t(x)$. This approach is useful since the base function can be arbitrary, including both parametric and non-parametric methods for regression, classification, and also signal processing.

For the binary classification problem in machine learning, the well-known Boosting is equivalent to forward stage-wise additive modeling[1]. More specifically, the popular AdaBoost algorithm[4] can be seen as a gradient descent solution of the additive logistic regression model: $F(\boldsymbol{x})=\log\frac{P(y=1\mid \boldsymbol{x})}{P(y=-1\mid \boldsymbol{x})}$, where $P(y=1|\boldsymbol{x})$ is the probability of an input point belonging to a positive class. AdaBoost in essence minimizes the exponential cost functional of the margin[5,6]. The algorithm gives more attention to large negative margins,

Received: 2007-03-15; revised: 2007-07-10

* Supported by the Basic Research Foundation of Tsinghua National Laboratory for Information Science and Technology (TNList)

** To whom correspondence should be addressed.
E-mail: zcs@mail.tsinghua.edu.cn; Tel: 86-10-62796872

which results in large weights for the misclassification data.

To boost a "base learner" (or "weak learner") to deal with the partially labeled data, the margin of unlabeled data should be defined[7]. Bennet et al.[8] introduced "pseudo-classes" of unlabeled points. Their method, ASSEMBLE (Adaptive Semi-Supervised ensEMBLE) method, has two advantages that the "base learner" can be any supervised classifier and the adaptive step-sizes can be efficiently computed.

Regularization theory was originally developed to solve ill-posed problems. Recently, it has been extended to become an important part of statistical learning. Although Boosting performs well in generalization property, it still needs regularization[9]. Regularization for Boosting has been studied for years[10]. It has many forms, such as the $L1$ or $L2$ penalty terms in the inner product space of the hypothesis coefficients[6,11], a term in the reproducing kernel Hilbert space (RKHS) of the combined function[12], and a term based on graph analysis of the "base learner"[13].

For semi-supervised learning[14,15], the regularization term based on the graph Laplacian is often used[16-19]. It is intuitive that every point's label should be similar to the labels of points in its local neighborhood. Therefore, besides maximizing the minimal margins, a smoother combined function with respect to the known labeled and unlabeled points would be very helpful. The graph Laplacian operator regularizes the function to be smooth on the constructed graph. Moreover, it can approximate to the Laplace-Beltrami operator on manifold which measures the smoothness of the defined function on the manifold[20].

In this paper, we propose a semi-supervised regularized additive logistic regression algorithm. The framework contains the graph regularization term of the combined functions together with the cost functional of the margin. Thus, the solution is the negative gradient direction in each iteration. The algorithm has several advantages:

- The algorithm includes a clear explanation of the solutions to the two functions. The first term minimizes the exponential negative margin cost functional. The second term corrects the current label of each point based on their neighbor information and smooths the combined function. Both are easily computed using defined pseudo-classes on their respective distributions.
- The solution gives an adaptive regularization parameter. This parameter guarantees that the cost functional of the margin decreases in each iteration. With the definition of the two types of pseudo-classes, the function step-size for each iteration can be computed using a Newton-Raphson iteration.
- The regularization result can be seen as a form of the shrinkage strategy[1]. Therefore, the additive model both reduces the margin cost and smooths the functions.

1 Regularized Semi-Supervised Additive Logistic Regression

For the semi-supervised learning problem, assume that the learner is learning from a set of observed examples $D=\{X, y\}$. Suppose that there are l labeled points and u unlabeled points. Then, the observed input points can be written as $X = \{X_L, X_U\}$, where $X_L = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_l\}$ and $X_U = \{\boldsymbol{x}_{l+1}, \boldsymbol{x}_{l+2}, ... , \boldsymbol{x}_{l+u}\}$. Each point $\boldsymbol{x} \in \mathcal{X} \subseteq \mathbf{R}^m$ is an m dimensional vector. In the two-class case, the label $y \in \mathcal{Y}$ can only be the binary values $\{1, -1\}$. The observed label set is $y = y_L = \{y_1, y_2, ..., y_l\}$. For labeled points, $\{X_L, y_L\}$ are randomly generated according to some unknown probability $p(x, y)$. For unlabeled points, X_U are randomly generated from the marginal probability $p(\boldsymbol{x})$.

In the additive logistic regression framework, the additive model has the form:

$$F_T(\boldsymbol{x}) = \sum_{t=1}^{T} c_t f_t(\boldsymbol{x}) \tag{1}$$

where $f_t(\boldsymbol{x})$: $\mathcal{X} \to \{1,-1\}$ is the "base learner" (hypotheses), t is the iteration number, and c_t is the weighting coefficient of the function f_t. The function $F(\boldsymbol{x})$ which minimizes the exponential criterion

$$J(F) = \mathrm{E}(\mathrm{e}^{-yF(\boldsymbol{x})}) \tag{2}$$

is the symmetric logistic transform of $P(y = 1|\boldsymbol{x})$[5].

1.1 Semi-supervised regularization framework

The general semi-supervised regularization framework on the combined function is

$$F^*(\boldsymbol{x}) = \arg\min_{F \in \mathcal{F}} \int_{\mathcal{X}\times\mathcal{Y}} V(y, F(\boldsymbol{x}))\mathrm{d}p(\boldsymbol{x}, y) + \lambda \| F \|_I^2 \tag{3}$$

where λ is a parameter that controls the tradeoff between the two terms. $\int_{\mathcal{X}\times\mathcal{Y}} V(F(\boldsymbol{x}), y)\mathrm{d}p(\boldsymbol{x}, y)$ is the expected risk. Many functions can be used for loss