基于 Fenchel 对偶的核 Logistic 回归并行学习算法

丁 朋 [1] 卿湘运 [1] 王行愚 [1]

摘 要 给出了一种大规模核 Logistic 回归的并行学习算法. 利用凸优化中的 Fenchel 对偶定理, 将核 Logistic 回归的优化原问题转换成对偶空间的优化问题, 再利用块更新迭代方法, 可以独立地在部分数据集上进行分类器训练. 设计了一个简单的客户机–服务器并行计算模式, 每个客户机对部分数据优化子问题, 在一次优化结束后, 服务器根据各客户机传递的信息修正各子问题目标函数. 在标准数据集的实验结果表明了基于 Fenchel 对偶的核 Logistic 回归并行学习算法的可行性.

关键词 核 Logistic 回归, Fenchel 对偶, 大规模机器学习, 凸优化

DOI 10.3724/SP.J.1004.2011.01145

A Parallel Learning Algorithm for Kernel Logistic Regression by Using Fenchel Duality

DING Peng[1] QING Xiang-Yun[1] WANG Xing-Yu[1]

Abstract A parallel learning algorithm for solving large scale kernel logistic regression problems is presented. Primal optimization problem for the kernel logistic regression is switched to the dual problem by using Fenchel duality theory in convex optimization. Then, learning the classifiers on subsets of training data can run independently when block-update methods are employed. A simple customer-server parallel computing mode is designed that each customer node learns a sub-problem for the subset of training data. Server node receives the messages passed by all customer nodes after one optimization iteration is end, followed by updating the objective functions of sub-problems. In comparison to non-parallel learning algorithms on standardized datasets, we obtain encouraging results.

Key words Kernel logistic regression (KLR), Fenchel duality, large-scale machine learning, convex optimization

Logistic 回归是统计分析、机器学习和数据挖掘领域一个经典的分类工具, 它的主要优点在于有统计理论基础, 并为探索性解释数据提供了一个有用的概率模型. 受到支持向量机 (Support vector machine, SVM) 取得广泛应用和巨大成功的影响, Logistic 回归非线性核化扩充之后的分类器称之为核 Logistic 回归 (Kernel logistic regression, KLR), 通过一个核函数使得原特征映射到一个高维或无穷维的特征向量, 分类器作用于此特征向量, 但不需要此特征向量的具体表达形式[1]. 与支持向量机相比较, KLR 的目标函数不是风险最小函数, 而是最大似然值, 因此 KLR 的解会产生类别归属的后验概率值, 同时 KLR 问题也是一个凸优化问题, 能够产生一个唯一的最优解. KLR 相当于将 SVM 的符号损失函数替换为二项分布的负对数似然函数, 两个损失函数在形式上类似[2], 但是 KLR 不再有支持向量的特性, 每个数据点对分类边界均有贡献.

研究者一般认为 KLR (也包括 Logistic 回归) 只适应于小规模问题, 对大规模问题很难学习训练, 目标函数优化速度慢. 如前所述, KLR 是一个凸优化问题, 应用梯度法如 Newton-Raphson 方法是常用的解决办法, 但是在每一次迭代中, Newton-Raphson 需要对一个 $m \times m$ 的矩阵进行求逆运算, 其中 m 表示数据点个数, 因此其复杂度为 $\mathrm{O}(m^3)$. 受到 Platt 等解决支持向量机二次规划问题的序贯最小优化算法 (Sequential minimal optimization, SMO) 的启发[3], Keerthi 等给出了核 Logistic 回归的快速对偶算法[4], 不需要对核矩阵进行任何矩阵运算, 类似解决 SVM 的 SMO 算法, 采用大小为 2 的工作集, 在每次迭代时计算代价最小, 但是当问题规模变大后, 收敛速度将很慢, 计算时间与数据规模的指数成正比, 约与 $m^{2.2}$ 成正比. 因此在利用 SMO 的优化包如 LIBSVM 软件包中利用了目标函数的二阶信息选择新的工作集. 但是由于核函数的限制, 对于大规模问题, 即使在软件实现中采用 LRU Cache 技术, 算法的速度也受到机器存储空间的限制, 甚至根本不能运行计算.

随着计算机集群技术及多核技术的发展, 大规模问题机器学习算法的并行实现成为了研究热点, 因此 NIPS (Neural information processing systems) 会议在 2009 年组织了一个 Workshop 专门讨论大规模机器学习的并行算法及其实现[5]. MapReduce 方法为大规模并行计算提供了一个框架, 各种机器学习算法的 MapReduce 实现平台 Mahout 也正在完善中[6]. SHOGUN 工具箱主要处理支持向量机的大规模学习问题[7]. 这些平台主要是将原算法进行适当的变化, 使得适合平台要求, 对系统构架和软件编程要求较高.

本文给出一个 KLR 的并行学习算法, 主要应用对偶分解的思想, 即将一个大的难解的原问题分解成若干个小的容易解的子问题, 再组合子问题的解得到原问题的解; 对于大规模问题的并行运算来说, 可将训练集分割成若干子集, 集群上的客户机对每个子集进行独立训练, 利用消息传递的机制, 在每一次训练结束服务器主机对各客户机传递的消息进行汇总, 再对各客户机提供修正子问题目标函数的消息, 迭代训练即可得到大规模问题的解. 由于核 Logistic 回归是一个凸优化问题, 应用 Fenchel 对偶定理对目标函数进行对偶分解, 可以得到全局最优解. 此类对偶分解的思想在离散马尔科夫随机场优化中已提出过[8], 但没有利用 Fenchel 对偶定理, 实质上虽然离散马尔科夫随机场优化的目标函数是非凸的, 但是每个子问题的 Fenchel 对偶是凸的, 因此也可以利用 Fenchel 对偶定理推出其算法. Hazan 等给出了支持向量机的并行分解算法[9], 利用 Fenchel 对偶定理得到了一个简单的分布式计算方法. 本文主要受上述工作的启发, 给出基于 Fenchel 对偶定理的核 Logistic 回归并行学习算法, 推导其训练过程, 并给出了实验方法, 实验结果验证了算法的有效性.

1 核 Logistic 回归的并行学习算法

1.1 核 Logistic 回归

设 x 表示输入数据, z 表示 x 通过核映射得到的特征空间向量, 即 $z = \varphi(x)$. 利用再生核希尔伯特空间表示定理, 我们只需知道核函数的形式, 即 $K(x, \hat{x}) = \varphi(x) \cdot \varphi(\hat{x})$, 而对其映射关系不需知道. 并假设 $S = \{(x_i, y_i)\}_{i=1}^m$ 表示为一个训练集, 其中, x_i 是第 i 个输入数据, y_i 是对应于 x_i 的类标

收稿日期 2010-12-06 录用日期 2011-03-02
Manuscript received December 6, 2010; accepted March 2, 2011
国家自然科学基金 (61074113) 资助
Supported by National Natural Science Foundation of China (61074113)
1. 华东理工大学自动化系 上海 200237
1. Department of Automation, East China University of Science and Technology, Shanghai 200237

签, $y_i = 1$ 表示类 1, $y_i = -1$ 表示类 2. 设 $z_i = \varphi(x_i)$, KLR 学习问题转化为如下凸优化问题的求解:

$$\min_{w,b} \frac{C}{2}\|w\|_2^2 + \frac{1}{m}\sum_{i=1}^{m} loss((w,b);(x_i,y_i)) \tag{1}$$

$$loss((w,b);(x_i,y_i)) = \log(1+\exp(-y_i(w\cdot z_i+b)))$$

其中, C 为正则化参数, 需通过交叉验证等办法整定其参数值. 损失函数与二项分布的负对数似然函数一致. KLR 除了能给定分类规则, 还能估计分类的概率:

$$p(y=1|x) = \frac{1}{1+\exp(-(\varphi(x)\cdot w+b))} \tag{2}$$

1.2 Fenchel 对偶定理

标记函数 $f(x)$ 的有效域为 $dom(f) = \{x : f(x) < \infty\}$, $\varnothing$ 为空集, 函数 $f(x)$ 在 x 的梯度为 $\nabla f(x)$, 函数 $f(x)$ 在 x 的次微分为 $\partial f(x)$. 设 $f(x), h_1(x), \cdots, h_k(x)$ 为正常闭凸函数, 如果函数定义域的相对内点满足 $ri(dom(f)) \cap ri(dom(h_i)) \neq \varnothing$, 且问题的优化值是有限的, 则如下的原–对偶优化问题:

原问题:

$$\min_x f(x) + \sum_{i=1}^{k} h_i(x) \tag{3}$$

对偶问题:

$$\max_{\lambda} \left\{ -f^*\left(-\sum_{i=1}^{k}\lambda_i\right) - \sum_{i=1}^{k} h_i^*(\lambda) \right\} \tag{4}$$

没有对偶间隙, 并且存在原–对偶优化对. $(x^*,\ \lambda_i^*\ )$ 成为原–对偶优化对当且仅当 $-\sum_{i=1}^{k}\lambda_i^* \in \partial f(x^*)$ 和 $\lambda_i^* \in \partial h(x^*)$. 如果 $f(x)$ 是本质上严格凸函数, 且 $f(x)$ 的共轭函数 $f^*(\lambda) = \max_{x\in dom(f)}\{\lambda^{\mathrm{T}}x - f(x)\}$ 是有限的, 则最优解 $x^* = \nabla f^*(-\sum_{i=1}^{k}\lambda_i^*)$.

定理证明见文献 [10], 文献 [11] 的附录也给出了类似的证明. 主要思想是将原问题分解成若干个子问题, 其约束条件为每个子问题的解需一致, 利用最优化中拉格朗日因子原理, 将约束条件引入子问题, 再求子问题 Fenchel-Legendre 共轭函数的最优解, 即得到原问题的逼近下界解, 不断迭代则可得到原问题的最优解. 对于严格凸函数, 原问题最优解与对偶问题最优解一致.

1.3 基于 Fenchel 对偶定理的 KLR 并行算法

为了便于阅读计算过程, 首先不考虑偏移量, 即设 $b = 0$, 则得到如下优化问题:

$$\min_w \frac{C}{2}\|w\|_2^2 + \frac{1}{m}\sum_{i=1}^{m} loss(w;(x_i,y_i)) \tag{5}$$

$$loss(w;(x_i,y_i)) = \log(1+\exp(-y_i\cdot w\cdot x_i))$$

假定在并行计算时把训练集分割为 k 个数据子集, 并假设为 $S_1, \cdots, S_k$. S_j 为第 j 个数据集中的数据在 S 中位置信息的集合. $S_j \subset \{1,\cdots,m\}$, 并且 $S_j \neq \varnothing$, $\bigcup_j S_j = \{1,\cdots,m\}$. 同时, k 也等于客户端的数目, 在每个客户机上对数据子集进行训练. 因此原问题 (5) 可以转换为

$$\min_{w_1=w_2=\cdots=w_k} \frac{C}{2k}\sum_{j=1}^{k}\|w_j\|_2^2 + \frac{1}{m}\sum_{j=1}^{k}\sum_{i\in S_j} loss(w_j;(x_i,y_i)) \tag{6}$$

定义集合:

$$\mathcal{H} = \left\{ \bar{w} = (w_1,\cdots,w_k) \in \mathbf{R}^{nk} : w_1 = \cdots = w_k \right\}$$

及一个指示函数 $\delta_{\mathcal{H}}(\bar{w})$. 当 $\bar{w} \in \mathcal{H}$ 时, $\delta_{\mathcal{H}}(\bar{w}) = 0$; 当 $\bar{w} \notin \mathcal{H}$ 时, $\delta_{\mathcal{H}}(\bar{w}) = \infty$. 问题 (6) 则可写成关于 $\bar{w}$ 及指示函数的新的优化问题:

$$\min_{\bar{w}} \frac{C}{2k}\|\bar{w}\|_2^2 + h(\bar{w}) + \delta_{\mathcal{H}}(\bar{w}) \tag{7}$$

其中, $h(\bar{w}) = \frac{1}{m}\sum_{j=1}^{k}\sum_{i\in S_j} loss(w_j;(x_i,y_i))$. 因为函数 $f(\bar{w}) = \frac{C}{2k}\|\bar{w}\|_2^2$, $h(\bar{w})$ 和指示函数 $\delta_{\mathcal{H}}(\bar{w})$ 均为正常闭凸函数, 且有 $ri(dom(f)) \cap ri(dom(h)) \neq \varnothing$ 和 $ri(dom(f)) \cap ri(dom(\delta_{\mathcal{H}})) \neq \varnothing$, 优化值有限, 因此可利用 Fenchel 对偶定理, 得到原问题 (7) 的对偶问题:

$$\max_{\lambda} \left\{ -f^*\left(-\sum_{i=1}^{2}\lambda_i\right) - h^*(\lambda_1) - \delta_{\mathcal{H}}^*(\lambda_2) \right\} \tag{8}$$

令 $\bar{\lambda} = \lambda_1$, $\bar{\mu} = \lambda_2$, 则对偶问题 (8) 可写成如下形式:

$$\max_{\bar{\lambda},\bar{\mu}} \{-f^*(-(\bar{\lambda}+\bar{\mu})) - h^*(\bar{\lambda}) - \delta_{\mathcal{H}}^*(\bar{\mu})\} \tag{9}$$

利用 $f(x) = \frac{C}{2}x^2$ 的共轭 $g(\lambda) = \frac{1}{2C}\lambda^2$ 这一性质, 式 (9) 可写为

$$\max_{\bar{\lambda},\bar{\mu}} -\frac{k}{2C}\sum_{j=1}^{k}\|\lambda_j+\mu_j\|_2^2 - \sum_{j=1}^{k} h_j^*(\lambda_j) - \delta_{\mathcal{H}}^*(\bar{\mu}) \tag{10}$$

利用块更新迭代算法时, 假定只优化部分变量, 其余变量保持不变. 因此在本算法中假定在第 t 次迭代优化 λ_j 时, 其余的 λ 变量和 $\bar{\mu}$ 变量保持不变, 再一次利用 Fenchel 对偶理论:

$$\begin{aligned}
\lambda_j^{(t)} &= \arg\max_{\lambda} -\frac{k}{2C}\left\|\lambda+\mu_j^{(t-1)}\right\|_2^2 - h_j^*(\lambda) = \\
&\arg\max_{\lambda} -\frac{k}{2C}\left\|\lambda+\mu_j^{(t-1)}\right\|_2^2 - \max_{w}(w^{\mathrm{T}}\lambda - h_j(w)) = \\
&\arg\min_{w}\max_{\lambda} -\frac{k}{2C}\left\|\lambda+\mu_j^{(t-1)}\right\|_2^2 - w^{\mathrm{T}}\lambda + h_j(w) = \\
&\arg\min_{w}\left(\max_{\lambda}\lambda^{\mathrm{T}}(-w) - \frac{k}{2C}\left\|\lambda+\mu_j^{(t-1)}\right\|_2^2\right) + \\
&h_j(w) = \arg\min_{w}\frac{C}{2k}\|w\|_2^2 + w^{\mathrm{T}}\mu_j^{(t-1)} + \\
&\frac{1}{m}\sum_{i\in S_j} loss(w;(x_i,y_i)) = P_j(\mu_j^{(t-1)}) = w_j^{(t)}
\end{aligned} \tag{11}$$

因此 $\lambda_j^{(t)}$ 和 $w_j^{(t)}$ 构成原–对偶优化对, 设 $f_1(\lambda) = h_j^*(\lambda)$, $f_2(\lambda) = -\frac{k}{2C}\left\|\lambda+\mu_j^{(t-1)}\right\|_2^2$, 根据 Fenchel 对偶理论 (见文献 [12] 第 434 页), $w_j^{(t)}$ 和 $\lambda_j^{(t)}$ 满足 Lagrangian 优化条件, 有

$$w_j^{(t)} = \arg\min_{w}\frac{C}{2k}\|w\|_2^2 + w^{\mathrm{T}}\mu_j{}^{(t-1)} + w^{\mathrm{T}}\lambda_j^{(t)} \tag{12}$$

得到最优解时, 上式右边函数的一阶微分为 0, 经过简单的运算得出求 $\lambda_j^{(t)}$ 的表达式为

$$\lambda_j^{(t)} = -\mu_j^{(t-1)} - \frac{C}{k}P_j(\mu_j^{(t-1)}) \tag{13}$$

$\bar{\mu}^{(t)}$ 的推导过程类似, 假定 λ 固定不变,

$$\begin{aligned}\overline{\mu}^{(t)} &= \arg\max_{\overline{\mu}} -\frac{k}{2C}\left\|\lambda^{(t)}+\overline{\mu}\right\|_2^2 - \delta^*_{\mathcal{H}}(\overline{\mu}) = \\ &\arg\max_{\overline{\mu}} -\frac{k}{2C}\left\|\lambda^{(t)}+\overline{\mu}\right\|_2^2 - \max_{\overline{w}\in\mathcal{H}} \overline{w}^{\mathrm{T}}\overline{\mu} = \\ &\arg\min_{\overline{w}\in\mathcal{H}}\left(\max_{\overline{\mu}} \overline{\mu}^{\mathrm{T}}(-\overline{w}) - \frac{k}{2C}\left\|\lambda^{(t)}+\overline{\mu}\right\|_2^2\right) = \\ &\arg\min_{\overline{w}\in\mathcal{H}} \frac{C}{2k}\|\overline{w}\|_2^2 + \overline{w}^{\mathrm{T}}\lambda^{(t)} = \\ &\arg\min_{w} \frac{C}{2}\|w\|_2^2 + w^{\mathrm{T}}\left(\sum_{j=1}^{k}\lambda_j^{(t)}\right) = \\ &P(\lambda^{(t)}) = w^{(t)}\end{aligned} \tag{14}$$

其中利用了指示函数的共轭是集合 $\mathcal{H}$ 的支撑函数这一性质, $\overline{\mu}^{(t)}$ 和 $w^{(t)}$ 构成原–对偶优化对. 由式 (14) 倒数第三步可以得出 w 的表达式:

$$w^{(t)} = -\frac{1}{C}\sum_{j=1}^{k}\lambda_j^{(t)} \tag{15}$$

设 $f_1(\overline{\mu}) = \delta^*_{\mathcal{H}}(\overline{\mu})$, $f_2(\overline{\mu}) = -\frac{k}{2C}\left\|\lambda^{(t)}+\overline{\mu}\right\|_2^2$, 根据 Fenchel 对偶理论及 Lagrangian 优化条件, 可得到 $\mu_j^{(t)}$ 的计算结果为

$$\mu_j^{(t)} = -\lambda_j^{(t)} + \frac{1}{k}\sum_{l=1}^{k}\lambda_l^{(t)} \tag{16}$$

从式 (13) 和 (15) 可以看出, 经过两次 Fenchel 对偶变换, 一方面可以将主问题转换为若干个子问题的解, 另一方面其优化运算还是在主空间进行, 避免求损失函数的共轭对偶.

当偏移量不为 0 时, 为便于块更新算法, 目标函数改写为

$$\begin{aligned}\min_{w_1=w_2=\cdots=w_k}\quad & \frac{C}{2k}\sum_{j=1}^{k}\|w_j\|_2^2 + \frac{C\varepsilon}{2}b^2 + \\ & \frac{1}{m}\sum_{j=1}^{k}\sum_{i\in S_j} loss(w_j;(x_i,y_i))\end{aligned} \tag{17}$$

其中, ε 是个任意小的数值. 同样应用上述方法得出偏移量不为 0 时的子问题:

$$\begin{aligned}P_j(d,d_{n+1}) = \quad & \min_{w,b}\frac{C}{2k}\|w\|_2^2 + \frac{C\varepsilon}{2k}b^2 + \\ & (w,b)^{\mathrm{T}}\cdot(d,d_{n+1}) + \\ & \frac{1}{m}\sum_{i\in S_j} loss((w,b);(x_i,y_i))\end{aligned} \tag{18}$$

向量 d, d_{n+1} 的作用就是作为服务器和客户机之间的消息, 修正各个客户机端对训练子集的学习结果.

设 $g(\varepsilon) = \log(1+e^{\varepsilon})$, 上述问题的 Lagrangian 表达为

$$\begin{aligned}L = & \frac{C}{2k}\|w\|_2^2 + \frac{C\varepsilon}{2k}b^2 + (w,b)^{\mathrm{T}}\cdot(d,d_{n+1}) + \\ & \frac{1}{m}\sum_{i\in S_j} g(\xi_i) + \sum_i \alpha_i\left[-\xi_i - y_i(w\cdot z_i - b)\right]\end{aligned} \tag{19}$$

由 KKT 条件可以得出:

$$\nabla_w L = \frac{C}{k}w + d - \sum_i \alpha_i y_i z_i = 0 \tag{20}$$

$$\frac{\partial L}{\partial b} = d_{n+1} + \sum_i \alpha_i y_i = 0 \tag{21}$$

(忽略 ε 项)

$$\frac{\partial L}{\partial \xi_i} = \frac{1}{m}g'(\xi_i) - \alpha_i = 0 \tag{22}$$

由式 (20) 和 (22) 可以看出: w, ξ_i 都是 α_i 的函数, 并且

$$\begin{aligned}w &= \frac{k}{C}(\sum_i \alpha_i y_i z_i - d) \\ \xi_i(\alpha_i) &= g'^{-1}(m\alpha_i)\end{aligned} \tag{23}$$

假设 $\delta = m\alpha_i$. 因为 ξ_i 是 α_i 的函数, 所以考虑下面这个函数:

$$G(\delta) = \delta\xi_i - g(\xi_i)$$

注意到:

$$\frac{\mathrm{d}G}{\mathrm{d}\delta} = \xi_i + \delta\frac{\mathrm{d}\xi_i}{\mathrm{d}\delta} - \frac{\mathrm{d}g}{\mathrm{d}\delta} = \xi_i = g'^{-1}(\delta)$$

因为 $g(\varepsilon) = \log(1+\mathrm{e}^{\varepsilon})$, 所以有 $g'(\xi) = \mathrm{e}^{\xi}/(1+\mathrm{e}^{\xi})$. 函数 g' 是单调、可微的, g' 的反函数 g'^{-1} 的定义域是 (0, 1), 因此 $\xi_i(\alpha_i)$ 的定义域为 $(0, 1/m)$. 如果 g 是凸函数, 则 G 也是凸函数. 因此有如下函数:

$$\begin{aligned}G(\delta) &= \delta\log\delta + (1-\delta)\log(1-\delta) \\ G'(\delta) &= \log(\frac{\delta}{1-\delta}) \\ G''(\delta) &= \frac{1}{\delta(1-\delta)}\end{aligned}$$

现在使用 Wolfe 对偶理论可以写出式 (18) 的对偶形式为

$$\begin{aligned}P_j^*(d) = & \min_{0\le\alpha\le 1/m} f(\alpha) = \frac{k}{2C}\alpha^{\mathrm{T}}Q_j^{\mathrm{T}}Q_j\alpha + \\ & \frac{1}{m}\sum_{i\in S_j} G(m\alpha_i) - \frac{k}{C}d^{\mathrm{T}}Q_j\alpha \\ \text{s.t.}\quad & d_{n+1} + \sum_i \alpha_i y_i = 0,\ \forall\, i\in S_j\end{aligned} \tag{24}$$

其中, Q_j 是由第 j 个训练子集中 $y_i z_i$ 向量组成的矩阵, 矩阵每一列为 $y_i z_i$, $i\in S_j$. 利用核映射的方法, $\tilde{K} = Q_j^{\mathrm{T}}Q_j$ 矩阵的每个元素 $\tilde{k}(r,s)$ 可以用 Mercer 核表示 $\tilde{k}(r,s) = y_r y_s k(x_r,x_s)$, $k(x_r,x_s)$ 为 Mercer 核; 同样根据式 (23) 也可将第 j 个客户机要解决的子问题的 w_j 表示成核的形式:

$$w_j = \frac{k}{C}(Q_j\alpha_j - d) \tag{25}$$

因此子问题的原–对偶优化解满足如下关系:

$$P_j(d) = \frac{k}{C}(Q_j P_j^*(d) - d) \tag{26}$$

利用 d_j 代替 μ_j, 将上式代入式 (13), 得到:

$$\lambda_j^{(t)} = -Q_j\alpha_j \tag{27}$$

再将此式代入式 (16), 可得:

$$d_j^{(t)} = Q_j\alpha_j^{(t)} - \frac{1}{k}\sum_{l=1}^{k}Q_l\alpha_l^{(t)} \tag{28}$$

因此行向量 $\tilde{\boldsymbol{L}} = d^{\mathrm{T}}Q_j$ 也可用 Mercer 核表示, 其表达式比较复杂, 第 t 次迭代时的第 j 个训练子集的行向量 $\tilde{\boldsymbol{L}}^{(t)}$ 的第 r 个元素可以表示为

$$\tilde{\boldsymbol{L}}^{(t)}(r) = \underbrace{\sum_{s=1}^{c_j} \alpha_s^{(t)} y_r y_s K(x_r, x_s)}_{\text{(a)}} - \underbrace{\frac{1}{k}\sum_{i=1}^{m} \alpha_i^{(t)} y_r y_i K(x_r, x_i)}_{\text{(b)}} \tag{29}$$

其中, x_r, x_s 表示第 j 个训练子集的数据, c_j 表示第 j 个训练子集数据个数, x_i 表示整个数据集的数据. 在各客户机解优化问题 (24) 后将对偶优化问题的权值 α 传递给服务器端, 由服务器计算式 (29), 再将计算结果传递给各客户机, 再解新的优化问题. 实际实现时, 为加快计算速度, 式 (29) 的 (a) 部分可在各客户机计算, 式 (29) 的 (b) 部分由服务器计算, 两者可以同时计算, 服务器端计算完 (b) 部分后再传送给各客户机. 当 $k = 1$ 时, 也即在一台机器上运行全部数据集时, 为零向量, 因此式 (24) 等同于 KLR 的非并行解法, 如文献 [4] 所述. 而在采用本文提出的并行算法时, 由于将大的数据集分割成若干个小的独立的数据集, 且在各独立的客户机上运行, 需要由服务器给客户机传递其他客户机子问题解的信息, 以修正子问题的解, 保证得到主问题的全局优化解.

算法 1. 基于 Fenchel 对偶的 KLR 并行求解算法

步骤 1. 各客户机并行优化子问题:

$$\alpha_j^{(t)} = P_j^*(d_j^{(t-1)})$$

$$b_j^{(t)} = P_j^b(d_{j,n+1}^{(t-1)})$$

对每个子问题 (18) 我们解其对偶优化问题 (24), 优化结束将 α 传送给服务器.

步骤 2. 服务器根据各客户机传递的消息修正更新各客户机子优化问题:

$$\bar{d}_j^{(t)} = Q_j\alpha_j^{(t)} - \frac{1}{k}\sum_{l=1}^{k} Q_l{\alpha_l}^{(t)}$$

由于引入了核映射, 所以这一步实质上是如上所述计算式 (29);

$$d_{j,n+1}^{t} = d_{j,n+1}^{t-1} - \frac{1}{k}\sum_{l=1}^{k} d_{j,n+1}^{t-1} + \frac{C\varepsilon}{k} b_j^{(t)} - \frac{C\varepsilon}{k^2}\sum_{l=1}^{k} b_l^{(t)}$$

$$d_j^{(t)} = \left(\bar{d}_j^{(t)}, d_{j,n+1}^{t}\right)$$

步骤 3. 迭代执行步骤 1 和 2, 直至满足算法结束条件.

1.4 算法终止条件及子问题的 SMO 解

为了算法的完整性, 类似文献 [4] 给出子优化问题 (24) 的终止条件、偏移参数的解法和子问题的 SMO 解.

问题 (24) 的 Lagrangian 表达为

$$\overline{L} = f(\alpha) - \beta\left(\sum_i \alpha_i y_i + d_{n+1}\right) \tag{30}$$

与文献 [4] 的主要不同之处是优化的目标函数发生了变化, 但其 SMO 推导步骤基本相似.

定义:

$$F_r = w(\alpha)\cdot z_r = \frac{k}{C}\sum_{s\in S_j}\alpha_s y_s k(x_r, x_s) - \frac{k}{C}\left(\left(1-\frac{1}{k}\right)\sum_{s\in S_j} y_s k(x_r, x_s)\right) \tag{31}$$

$$H_r = F_r + y_r G'(m\alpha_r) \tag{32}$$

从而对偶问题的最优条件为

$$\frac{\partial \overline{L}}{\partial \alpha_r} = (H_r - \beta)y_r = 0, \quad \forall r \in S_j \tag{33}$$

定义:

$$b_{\mathrm{up}} = \max_r H_r$$
$$i_{\mathrm{up}} = \arg\max_r H_r$$
$$b_{\mathrm{low}} = \min_r H_r$$
$$i_{\mathrm{low}} = \arg\min_r H_r$$

那么由最优条件可以得出问题 (24) 的终止条件为

$$b_{\mathrm{low}} = b_{\mathrm{up}}$$

即问题 (24) 的求解过程中, 只要

$$H_r \neq H_s \tag{34}$$

问题 (24) 就会一直在求解.

现在假设 (r, s) 满足式 (34), 调节 α_r, α_s 使式 (24) 函数 f 的值在下降. 定义:

$$\tilde{\alpha}_r = \alpha_r + \frac{t}{y_r},\ \tilde{\alpha}_s = \alpha_s - \frac{t}{y_s},\ \tilde{\alpha}_k(t) = \alpha_k, \quad \forall k \neq r, s$$

从而 $\phi(t) = f(\tilde{\alpha}(t))$.

由于 $f(\alpha)$ 是凸函数, ϕ 也是严格的凸函数. 并且 ϕ 的定义域为

$$\{t : \tilde{\alpha}(t) \in A\}\ \left(A = \left\{\alpha : 0 < \alpha_r < \frac{1}{m} \quad \forall r\right\}\right)$$

因此, $\phi^{'}(t) = H_r - H_s$. 选择合适的 t, $t^* = \arg\min_t \phi(t)$, 并设 $\alpha_{\mathrm{new}} = \tilde{\alpha}_k(t^*)$.

对于终止条件, 可改为

$$b_{\mathrm{low}} \geq b_{\mathrm{up}} - 2\tau \tag{35}$$

其中, τ 是精度参数. 当终止条件满足后, 可通过下式求出 b: $b = (b_{\mathrm{low}} + b_{\mathrm{up}})/2$. 由此, 得出子问题的 SMO 解.

算法 2. 子问题的 SMO 解

步骤 1. 选择合适的初始值 $\alpha^0 \in A$, $u = 0$;

步骤 2. 如果满足 (35), 终止; 如果不满足, $\alpha = \alpha^u$, 选择 $r = i_{\mathrm{up}}$, $s = i_{\mathrm{low}}$, 利用 Newton-Raphson 算法解 $t^* = \arg\min_t \phi(t)$;

步骤 3. 令 $\alpha^{u+1} = \alpha_{\mathrm{new}}$, $u = u + 1$, 返回步骤 2.

2　实验

2.1　实验设置

为了验证算法性能, 我们搭建了一个简单的客户机–服务器局域网, 见图 1 所示, 采用星形网络拓扑结构. 交换机为港湾 Hammer 3550-24, 传输速度 10/100 MB 自适应. 受实验条件的限制, 各台客户机的配置不一样, 最低配置的客户机内存仅为 256 M, CPU 主频 1.6 G, 最高配置的客户机内存为 1.5 G, 其 CPU 主频为 2.4 G, 服务器为联想万全服务器. 虽然对大规模实验有一定的限制, 但也凸显本算法对网络配置等要求较低的优点. 以下实验中, 单机运行实验均在配置最好的机器上运行.

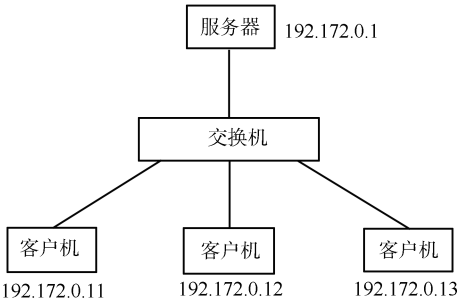

图 1　网络拓扑结构

Fig. 1　Topological structure of network

搭建的网络采用 Linux 操作系统. 选取 ssh 作为可信赖的服务, 局域网各节点之间用 ssh 登录. 编辑各节点之间的/etc/hosts 文件, 配置各节点名称及 IP 地址 (如图 1 所示的 IP 地址). 安装 NFS 文件系统, 将分割的数据集文件传输至各节点. 全部程序采用 Linux 下的 C++ 语言实现, 通过配置脚本文件实现网络之间的通信和任务批处理等工作. 并行计算时通过 socket 编程实现服务器和客户机之间的通信. 核 Logistic 回归的单机实现程序来自 http://www-ai.cs.uni-dortmund.de/SOFTWARE/MYKLR/, 此程序实现了文献 [4] 提出的算法.

实验数据均来自 LIBSVM 网站 (www.csie.ntu. edu.tw/cjlin/libsvm/) 提供的二类分类数据, 数据名称和属性如表 1 所示.

表 1　实验数据属性

Table 1　Properties of the data sets

数据集	特征维数	训练样本个数	测试样本个数
a4a	123	4 781	27 780
w4a	300	7 366	42 383
w5a	300	9 888	39 861
w6a	300	17 188	32 561
w7a	300	24 692	25 057
a8a	123	22 692	9 865
a9a	123	32 561	16 281

2.2　性能评价指标

为阐述算法的并行性, KLR 的概率输出结果暂不做复杂度 (Perplexity) 等指标的计算, 而是输出二值分类结果. 如前所述, KLR 虽具有概率分类结果的优点, 但是不具有稀疏解的特性, 因此计算复杂度较 SVM 高许多, 特别是在利用 SMO 求解时, 对于稍大规模的问题, KLR 的 SMO 求解非常慢, 实验发现其计算时间并不简单地如文献 [4] 所述, 即计算时间同 $m^{2.2}$ 成正比, 而是以更大的指数增长. 因此不同于 SVM 的并行求解算法, 算法运行时间的增益不是简单地与客户机数成正比的关系, 而经常出现的情况是在可接受的时间里可解和不可解的问题. 虽然所取的数据集规模属于中等, 但是对于单机求解其计算时间仍是难以忍受的, 所以没有取规模很大的数据集, 一方面是因为难以同非并行算法进行比较, 另一方面也受实验条件的限制.

所有实验采用径向基核, 核宽度为 1, 正则化因子 $C = 0.00001$. 以下实验是在三台客户机 (即 $k = 3$) 和一台服务器上运行, 训练样本均分成三个小规模的数据集, 每台客户机只对分配给它的数据进行训练. 单机运行 KLR 时实验是在性能配置最优的机器上进行的.

本研究主要与单机运行 KLR 算法进行比较, 其主要比较的指标有:

1) 目标函数值

$$\frac{1}{2C}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y_i y_j K(x_i,x_j)+\frac{1}{m}\sum_{i=1}^{m}\log\left(1+\mathrm{e}^{-\frac{1}{C}\sum\limits_{j=1}^{m} y_i y_j \alpha_j K(x_i,x_j)-b y_i}\right) \tag{36}$$

目标函数值越小, 说明优化性能更佳.

2) 运行时间

对于运行时间超过 1 小时的实验结果用 "–" 表示, 而不计算最终运行结果, 主要的原因是其运行时间已不具备可比性, 不需要再运行下去.

3) 训练样本正确分类率和测试样本正确分类率

通过判断下式的符号来决定输入样本 x_i 的类别:

$$\frac{1}{C}\sum_{j=1}^{m} y_j\alpha_j K(x_j,x_i)+b \tag{37}$$

值得一提的是, 本研究主要研究算法的可并行性及在运行时间上的增益, 因此固定模型参数, 而没有采用交叉验证等方法通过优化模型参数的方法来提高分类正确率.

2.3　实验结果

2.3.1　在标准数据集上的并行计算实验结果

在 6 个标准数据集上的并行计算实验结果如表 2 所示. 由于部分数据集单机运行时间超过 1 小时尚未结束, 因此没有给出实验结果, 而本研究给出的并行算法能在短时间内结束各数据集的计算, 因此在运行时间上明显占优, 虽然没有对参数进行整定, 但就其绝对分类结果也还不错.

2.3.2　客户机数目固定、训练数据集规模递增的并行计算结果

接下来的实验是在 a9a 数据集上进行的, 主要研究在客户机数目固定为 3、训练数据集规模递增时运行时间情况, 实验结果如图 2 所示. 从图 2 可以看出, 在客户机数目固定的情况下并行算法运行时间与数据集规模成线性关系.

表 2 并行和非并行 KLR 在 6 个标准数据集上的计算代价和泛化性能比较
Table 2 Computational costs and generalization performance comparison of parallel and non-parallel KLR on the six benchmark datasets

数据集	函数目标值		运行时间 (s)		训练样本分类正确率 (%)		测试样本分类正确率 (%)	
	单机	并行	单机	并行	单机	并行	单机	并行
a4a	0.6931	0.5439	19	6	87.7	93.46	84.69	89.72
w4a	0.3447	0.5502	960	18	93.76	91.23	86.47	80.91
w5a	–	0.1117	–	37	–	89.52	–	78.78
w6a	–	0.2795	–	85	–	92.36	–	89.30
w7a	–	0.3649	–	504	–	89.94	–	81.38
a8a	–	0.3856	–	135	–	89.37	–	81.39

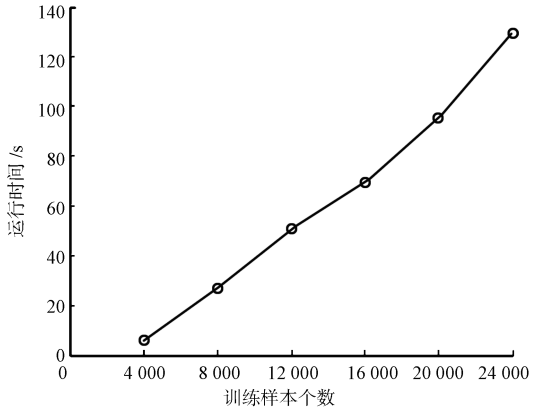

图 2 a9a 数据集训练样本递增时的运行时间
Fig. 2 Running time with varying scales of the a9a dataset

2.3.3 客户机数目递增的并行计算结果

我们也初步研究在客户机数目递增时算法的运行时间. 取 a9a 数据集的 12 000 个训练样本来进行实验, 实验结果如表 3 所示.

表 3 a9a 数据集在客户机数目递增时算法的运行时间
Table 3 Running time with varying customer nodes on the a9a dataset

客户端节点数	$k = 1$	$k = 2$	$k = 3$
运行时间 (秒)	–	68	51

当 $k = 1$ 时, 即单机运行全部数据, 运行时间超过 1 小时, 客户机数目递增, 由于客户机数目不多且网络通信需花费一定时间, 因此运行时间减少不是很显著. 取 a9a 数据集的 24 000 个样本进行训练, $k = 2$ 时算法运行时间也超过 1 小时, 因此没有给出比较结果. 该实验也说明一个问题, 由于 KLR 解的非稀疏性导致 SMO 求解的困难, 因此在单机上运行 KLR 的子问题规模不能过大, 需要增加更多的客户机数, 将大的训练数据集分割成多个更小的数据集进行训练, 才能使得问题可快速求解, 也说明了并行训练 KLR 的必要性.

3 结论与展望

本文给出了基于 Fenchel 对偶的核 Logistic 回归的并行学习算法, 将一个大的数据集划分成若干个小的数据集, 推导出了基于小数据集上的子问题优化目标函数和优化算法, 在每个客户机上对子问题学习训练结束之后向服务器传送权值和部分中间结果, 服务器再修正各子问题优化目标函数, 经迭代训练得到问题的全局最优解.

接下来的主要工作围绕如下两方面展开: 1) SMO 训练速度还是很慢, 可考虑再对每个子问题应用 Fenchel 对偶定理进行在线学习, 相当于对本算法取极端情况, 每个样本对应于一个客户机; 2) 针对多类核 logistic 回归给出并行学习算法.

References

1 Jaakkola T S, Haussler D. Probabilistic kernel regression models. In: Proceedings of the 7th International Workshop on Artificial Intelligence and Statistics. Florida, USA: Morgan Kaufmann Publishers, 1999. 1−9

2 Zhu J, Hastie T. Kernel logistic regression and the import vector machine. *Journal of Computational and Graphical Statistics*, 2005, **14**(1): 185−205

3 Platt J C. Fast training of support vector machines using sequential minimal optimization. Advances in Kernel Methods. Cambridge, USA: The MIT Press, 1999. 185−208

4 Keerthi S S, Duan K B, Shevade S K, Poo A N. A fast dual algorithm for kernel logistic regression. *Machine Learning*, 2005, **61**(1−3): 151−165

5 Large-scale machine learning: parallelism and massive datasets [Online], available: http://www.select.cs.cmu.edu/meetings/biglearn09/#xu, December 11, 2009

6 Gillick D, Faria A, Denero J. Map reduce: distributed computing for machine learning [Online], available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi = 10.1.1.111.9204, October 10, 2010

7 Sonnenburg S, Ratsch G, Henschel S, Widmer C, Behr J, Zien A, Bona F, Binder A, Gehl C, Franc V. Shogun-a large scale machine learning toolbox. *Journal of Machine Learning Research*, 2010, **11**: 1799−1802

8 Komodakis N, Pragios N, Tziritas G. MRF-optimization via dual decomposition: message-passing revisited. In: Proceedings of the 11th International Conference on Computer Vision. Rio de Janeiro, Brazil: IEEE, 2007. 1−8

9 Hazan T, Man A, Shashua A. A parallel decomposition solver for SVM: distributed dual ascend using fenchel duality. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Anchorage, USA: IEEE, 2008. 1−8

10 Hazan T, Shashua A. Norm-product belief propagation: primal-dual message passing for approximate inference. *IEEE Transactions on Information Theory*, 2010, **56**(12): 6294−6316

11 Shalev-Shwartz S, Kakade S M. Mind the duality gap: logarithmic regret algorithms for online optimization. *Advances in Neural Information Processing Systems*. Cambridge: The MIT Press, 2009. 1457−1464

12 Bertsekas D P, Nedic A, Ozdaglar A E. *Convex Analysis and Optimization*. Beijing: Tsinghua University Press, 2006. 421−446

丁 朋 华东理工大学硕士研究生. 主要研究方向为机器学习和模式识别. E-mail: pding85@126.com
(**DING Peng** Master student at East China University of Science and Technology. His research interest covers machine learning and pattern recognition.)

卿湘运 华东理工大学讲师. 主要研究方向为机器学习和模式识别. 本文通信作者. E-mail: xytsing@yahoo.com.cn
(**QING Xiang-Yun** Lecturer at East China University of Science and Technology. His research interest covers machine learning and pattern recognition. Corresponding author of this paper.)

王行愚 华东理工大学教授. 主要研究方向为智能控制和模式识别. E-mail: xywang@ecust.edu.cn
(**WANG Xing-Yu** Professor at East China University of Science and Technology. His research interest covers intelligent control and pattern recognition.)