基于自适应随机梯度下降方法的非平衡数据分类

陶秉墨[1] 鲁淑霞[1,2]
(河北大学数学与信息科学学院 河北 保定 071002)
(河北省机器学习与计算智能重点实验室 河北 保定 071002)[2]

摘 要 对于不平衡数据分类问题,传统的随机梯度下降方法在求解一般的支持向量机问题时会产生一定的偏差,导致效果较差。自适应随机梯度下降算法定义了一个分布 p,在选择样例进行迭代更新时,其依据分布 p 而非依据均匀分布来选择样例,并且在优化问题中使用光滑绞链损失函数。对于不平衡的训练集,依据均匀分布选择样例时,数据的不平衡比率越大,多数类中的样例被选择的次数就越多,从而导致结果偏向少数类。分布 p 在很大程度上解决了这个问题。普通的随机梯度下降算法没有明确的停机准则,这导致何时停机成为一个很重要的问题,尤其是在大型数据集上进行训练时。以训练集或训练集的子集中的分类准确率为标准来设定停机准则,如果参数设定恰当,算法几乎可以在迭代的早期就停止,这种现象在大中型数据集上表现得尤为突出。在一些不平衡数据集上的实验证明了所提算法的有效性。

关键词 随机梯度下降,非均匀分布,停机准则,支持向量机,损失函数
中图法分类号 TP181 **文献标识码** A

Adaptive Stochastic Gradient Descent for Imbalanced Data Classification

TAO Bing-mo[1] LU Shu-xia[1,2]
(College of Mathematics and Information Science, Hebei University, Baoding, Hebei 071002, China)[1]
(Hebei Province Key Laboratory of Machine Learning and Computational Intelligence, Baoding, Hebei 071002, China)[2]

Abstract For imbalanced data classification, the performance of using traditional stochastic gradient descent for solving SVM problems is not very well. Adaptive stochastic gradient descent algorithm defines a distribution p instead of using uniform distribution to choose examples, and the smoothing hinge loss function is used in the optimization problem. Because of the training sets are imbalanced, using uniform distribution will cause the algorithm choose more majority class based on the imbalanced ratio. That would result the classifier bias towards the minority class. The distribution p largely overcomes this issue. When to stop the programs becomes an important problem, because the normal stochastic gradient descent algorithm does not have a stop criterion especially for large data sets. The stop criterion was setted according to the classification accuracy on the training sets or its subsets. This stop criterion could stop the programs very early especially for large data sets if the parameters are chosen properly. Some experiments on imbalanced data sets show that the proposed algorithm is effective.

Keywords Stochastic gradient descent, Nonuniform distribution, Stop criterion, Support vector machine, Loss function

1 引言

支持向量机[1](Support Vector Machine,SVM)是由 vapnik 等于 1995 年首次提出的。SVM 在解决小样本、非线性问题时表现出了许多优势。但是当训练数据集的规模较大时,SVM 目标函数的求解,尤其是带有核函数的非线性 SVM 问题的求解变得困难。Platt 等于 1998 年提出了序列最小最优化算法[2](Sequential Minimal Optimization,SMO),该算法并不是直接求解 SVM 的原始问题,而是求解其对偶问题。由于 SVM 的对偶问题是一个二次规划问题,该算法每次迭代时将原问题化为一个一元二次方程的子问题,从而可以直接求解,因此 SMO 具有较快的求解速度。SMO 算法在执行过程中的主要计算量并不在于子问题的求解,而是寻找最优的被优化坐标,而后者是与数据规模成正相关的。因此,当数据集的规模越来越大时,SMO 的求解速度会受到很大的影响。尤其是非线性 SVM 问题,由于引入了核函数,大规模的数据集会导致核矩阵的计算和存储开销很大,因此 SMO 的优势主要在于线性、数据稀疏问题。

近年来,坐标下降[3-4](Coordinate Descent,CD)算法和对偶坐标下降[5-7](Dual Coordinate Descent,DCD)算法被广泛研究。坐标下降算法与 SMO 类似,二者都是将要求解的最优化问题转化为一个更容易求解的子问题,从而利用迭代的策略来逼近原问题的最优值。值得指出的是,坐标下降算法在子问题也就是工作区的选取上比 SMO 更加灵活。通常情

本文受河北省自然科学基金(F2015201185)资助。
陶秉墨(1991-),男,硕士生,主要研究方向为机器学习,E-mail:281035569@qq.com;鲁淑霞(1966-),女,博士,教授,CCF 会员,主要研究方向为机器学习,E-mail:cmclusx@126.com(通信作者)。

况下，坐标下降算法是选定一个坐标之后，将其余变量视作常数，从而将目标多元函数转化为一元函数进行求解。块坐标下降算法[8](Block Coordinate Descent，BCD)将目标问题的变量按照一定策略划分为若干个子块，每次迭代更新时选定一个子块进行子问题求解。BCD 更重要的应用是在并行计算方面，通过某种策略将目标问题划分为子块之后，将每个子块再分配给各个计算节点进行并行计算，从每个节点返回的计算结果在各个节点之间交互之后，进行下一次的迭代。对偶坐标下降算法很显然是求解目标问题的对偶问题，该算法的优点在于在原问题和对偶问题之间定义了对偶残差的概念，从而可以利用该对偶残差的概念进行优化坐标的选取。对偶残差还可以作为停机准则，因为随着优化过程的进行，在满足一定的条件下，对偶残差会逐渐收敛至 0。

无论是 SMO，还是坐标下降或对偶坐标下降，这些算法的求解速度在很大程度上都受到训练集规模的影响。当计算设备有限时，很难处理大规模的问题。随机梯度下降[9-10](Stochastic Gradient Descent，SGD)算法作为梯度下降(Gradient Descent，CD)算法的一个变种，在处理大规模训练数据的问题上具有较好的表现。其原因是该算法采用随机梯度代替梯度来进行迭代更新，导致其每一次迭代的代价很小，并且不受训练集规模的影响。使用随机梯度而不是梯度的另一个优点是，当目标优化问题不是凸优化问题时，使用梯度下降的方法能否收敛到全局最优值受到初始点和步长的选取的影响。一旦迭代过程陷入到局部极值，将很难从局部极值中跳出。随机梯度下降算法可以在很大程度上解决这个问题，因为随机梯度下降并不是每一次都严格地朝着目标函数下降的方向迭代，所以当迭代过程陷入局部极值后，有很大概率跳出局部极值，从而避免了梯度下降所遇到的情况。

随机梯度下降因为没有一个明确的停机准则，所以在很多情况下设定迭代次数时需要一定的技巧。而上述其他算法则有很好的停机准则，比如 SMO 通过 KKT 条件来判别是否停机，对偶坐标下降使用对偶残差作为停机准则。我们将在下文的随机梯度下降算法中给出一个利用训练集分类准确率来判断是否停机的停机准则。

关于 SVM 求解的另一个问题是，当训练数据不平衡时如何有效地求解决策超平面。不平衡数据集的分类算法在现实世界中有着重要的意义。按照惯例，对于二分类问题，通常我们认为正类为少数类而负类为多数类，在现实的问题中，通常会更多地关注少数类。比如在疾病诊断、信用卡欺诈等领域，少数类往往是我们关注的重点。处理不平衡数据分类问题目前常用的有两种策略。第一种是对原算法进行一定的改进。比如上文提到的 SMO，CD，DCD 等算法都可以采用对训练样例加权的方法进行算法的调整。也就是说，赋予少数类比多数类更大的权重。模糊支持向量机[11](Fuzzy Support Vector Machine，FSVM)是 Lin 等于 2002 年提出的，这种算法在传统的支持向量机的基础上增加了模糊成员变量属性，为不同的样例赋予不同的权重，在处理不平衡数据和抗噪声、抗异常点方面具有良好的表现。在此基础上，Fan 等提出了基于熵的模糊支持向量机[12](Entropy-Based Fuzzy Support Vector Machine，EFSVM)，在 FSVM 的基础上给出了以熵为基础的明确的模糊隶属度，其在处理不平衡数据问题时有良好的表现。最大间隔分布机[13](Large Margin Distribution Machine，LDM)是周志华等于 2014 年提出的，这种算法最大化间隔分布而非传统的最大化最小间隔。理论研究表明，间隔分布作为衡量间隔的标准，比最小间隔更为合理。在此基础上，Cheng 等于 2016 年提出了损失敏感最大间隔分布机[14](Large Cost-Sensitive Distribution Machine，LCSDM)。该算法在目标函数中的间隔分布和损失函数项上同时加入了权重，因此在处理不平衡数据时对模型有较好的修正。上述方法均是给予训练样例不同的权重，从而在一定程度上解决了不平衡数据问题。加权的方法有一个明显的缺点，即模型求解过程中迭代不稳定，尤其是当训练数据的不平衡程度较大时。当数据不平衡程度较大时，为了得到较好的分类结果，必然会给少数类赋予很大的权重，因此少数类中单个样例对模型的影响很大。例如，用加权的随机梯度下降方法求解模型时，且当前迭代选择少数类样例时，由于其权重很大，一次迭代会对模型产生很大的影响从而影响迭代过程的稳定性。我们将在下文给出根据某种分布来选择样例的随机梯度下降算法，这种算法可以很好地解决模型迭代过程中的不稳定问题。解决不平衡数据的另一类方法是直接对训练数据进行预处理。SVM-SMOTE[15]是一种对少数类的过采样方法，该方法在分类算法执行前通过某种方式合成少数类数据以使数据达到平衡。相对于 SVM-SMOTE，还有一些对数据集欠采样[16-18]的方法，这些方法都是在训练数据之前对其进行一定的处理，使数据的不平衡程度降低。这些方法都在一定程度上改善了不平衡数据带来的问题，但是过采样增加了训练样例的个数，从而增加了计算量。欠采样删除了某些数据，很有可能丢失对分类有用的信息。

2 随机梯度下降算法

支持向量机是一种非常流行且非常有效的机器学习算法。通常的支持向量机有原问题和对偶问题两种形式，本文重点考虑原问题的形式。给定一个训练集 $S=\{(x_i,y_i)\}_{i=1}^m$，$x_i\in R^n$ 且 $y_i\in\{+1,-1\}$，给出如下优化问题：

$$\min_w \frac{\lambda}{2}\|w\|^2+\frac{1}{m}\sum_{(x,y)\in S} l(w;(x,y)) \tag{1}$$

其中，

$$l(w;(x,y))=\max\{0,1-y\langle w,x\rangle\} \tag{2}$$

式(2)给出的是损失函数，而 $\frac{\lambda}{2}\|w\|^2$ 可以是目标函数的正则化项。随机梯度下降算法不同于梯度下降算法，梯度下降算法进行的每一次迭代更新所使用的梯度是用所有的训练样例计算出来的梯度。普通的梯度下降算法在进行每一次迭代的过程中都需要遍历整个训练集，导致它的计算量过于庞大，尤其是在训练集的规模很大时。另外，如果目标函数有很多个局部极小值，则普通的梯度下降算法很容易陷入局部极小值，从而无法求得全局最优。

随机梯度下降算法并不是计算梯度的精确值，而是用梯度的无偏估计来代替梯度。其目标函数不再是式(1)所给出的形式，而是如式(3)所示：

$$f(w;i_t)=\frac{\lambda}{2}\|w\|^2+l(w;(x_{i_t},y_{i_t})) \tag{3}$$

对式(3)求梯度可得式(4)：

$$\nabla_t=\lambda w_t-\mathbb{I}[y_{i_t}\langle w_t,x_{i_t}\rangle<1]y_{i_t}x_{i_t} \tag{4}$$

其中，$\mathbb{I}[y\langle w,x_i\rangle<1]$代表指示器函数，如果中括号内的判断语

句为真，则其值为 1，否则为 0。更新公式如式(5)、式(6)所示。

$$w_{t+1}=w_t-\eta_t\nabla_t \tag{5}$$

$$w_{t+1}=w_t+\eta_t\lambda w_t-\eta\, t\mathbb{I}[y_{i_t}\langle w_t,x_{i_t}\rangle<1]y_{i_t}x_{i_t} \tag{6}$$

其中，η_t 为步长。

3　自适应随机梯度下降算法

标准的支持向量机采用的是绞链损失函数，如式(2)所示，绞链损失函数的一个特点是该函数在某些点不可导。我们采用另外一种损失函数，如式(7)所示。在随机梯度下降算法的每一次迭代中，选择一个样例来进行目标函数的更新，这里按照某种分布 p 来随机选择样例 i_t，w 初始化为 0。如此，目标函数由式(8)给出，对式(8)求导可得式(9)。

$$\phi_\gamma(a)=\begin{cases}0, & a\geqslant 1\\ 1-a-\gamma/2, & a\leqslant 1-\lambda\\ 1/2\gamma(1-a)^2, & \text{otherwise}\end{cases} \tag{7}$$

$$\min_w\frac{\lambda}{2}\|w\|^2+\phi_\gamma(y_{i_t}\langle w,x_{i_t}\rangle) \tag{8}$$

$$\nabla_t=\begin{cases}\lambda w_t, & y_{i_t}\langle w_t,x_{i_t}\rangle\geqslant 1\\ \lambda w_t-1/\gamma(y_{i_t}\langle w_t,x_{i_t}\rangle-1), & 1-\gamma<y_{i_t}\langle w_t,x_{i_t}\rangle<1\\ \lambda w_t+1, & y_{i_t}\langle w_t,x_{i_t}\rangle\leqslant 1-\gamma\end{cases} \tag{9}$$

注意到 $w_{t+1}=w_t-\frac{1}{\lambda t}\nabla_t$，因此可以得到式(10)。这里给出了线性随机梯度下降的 w 的更新公式，我们重点关注带有核函数的随机梯度下降的更新公式。由式(10)可以得到式(11)。式(12)给出损失函数的导函数。

$$w_{t+1}=\begin{cases}(1-\frac{1}{t})w_t, & y_{i_t}\langle w_t,x_{i_t}\rangle\geqslant 1\\ (1-\frac{1}{t})w_t-\frac{1}{\lambda\gamma t}(y_{i_t}\langle w_t,x_{i_t}\rangle-1), & 1-\gamma<y_{i_t}\langle w_t,x_{i_t}\rangle<1\\ (1-\frac{1}{t})w_t+\frac{1}{\lambda t}, & y_{i_t}\langle w_t,x_{i_t}\rangle\leqslant 1-\gamma\end{cases} \tag{10}$$

$$w_{t+1}=(1-\frac{1}{t})w_t-\frac{1}{\lambda t}\varphi_t \tag{11}$$

$$\varphi_{i_t}=\begin{cases}0 & y_{i_t}\langle w_t,x_{i_t}\rangle\geqslant 1\\ \frac{1}{t}(y_{i_t}\langle w_t,x_{i_t}\rangle-1), & 1-\gamma<y_{i_t}\langle w_t,x_{i_t}\rangle<1\\ -1, & y_{i_t}\langle w_t,x_{i_t}\rangle\leqslant 1-\gamma\end{cases} \tag{12}$$

由式(10)、式(11)可知，随着 w 的不断更新，φ_i 在 w_{t+1} 中的系数如式(13)所示：

$$\frac{1}{\lambda i}\prod_{j=i+1}^{t}\frac{j-1}{j}=\frac{1}{\lambda t} \tag{13}$$

则有：

$$w_{t+1}=\frac{1}{\lambda t}\sum_{i=1}^{t}\varphi_i \tag{14}$$

其中，$\varphi(x_{i_t})$ 由式(12)给出。

如果 w_{t+1} 如式(5)所示，则可以仅通过内积来表示算法中的 $\varphi(x_{i_t})$，从而使用核函数：

$$w_{t+1}=\frac{1}{\lambda t}\sum_{j=1}^{m}\alpha_{t+1}[j]y_j\phi(x_j) \tag{15}$$

由于 φ_i 分 3 种情况，令 $\gamma=1$，在这种参数设定下可以看到，当 $y_{i_t}\langle w_t,x_{i_t}\rangle\geqslant 1$ 时，被选到的样例是分类正确的，并且处于支持超平面的两侧，因此它们是非支持向量，这时 α 应设定为 0。当 $1-\gamma<y_{i_t}\langle w_t,x_{i_t}\rangle<1$ 时，本文中由于 $\gamma=1$，因此 $0<y_{i_t}\langle w_t,x_{i_t}\rangle<1$，这时被选到的样例处于分类超平面与自己所在类别对应的支持超平面之间，且它们是分类正确，这时给对应的 α 的分量加 1。当 $y_{i_t}\langle w_t,x_{i_t}\rangle\leqslant 0$ 时，这时被选到的样例是被当前的超平面分错的点，显然这样的样例一定是支持向量，给予它们比第二种情况更大的权重，如给对应的 α 的分量增加 2。

此时，引入一个支持向量的退出机制。由上文可知，当 α 初始化为 0 以后，在 3 种情况中 α 要么保持 0 不变，要么增加某个整数，因此一旦 α 的某个分量由 0 变为某个正整数，它的这个分量就不再可能变回 0。从模型初期的迭代来看，这是不合理的。因为程序迭代初期可能的分类超平面是严重偏离最终的收敛的分类超平面的，所以在严重偏离的情况下，模型把某些本应该最终是非支持向量的样例判断为当前迭代这一步的支持向量。随后的实验表明，当训练样例的规模足够大时，这是没有影响的。因为假如训练样例有 5 万个，支持向量有 5000 个，这时多几个错误的支持向量对于模型的影响可忽略不计。但是，当训练集的规模很小时，这种现象的影响就不可忽略了。

自适应随机梯度下降算法如算法 1 所示。

算法 1　自适应随机梯度下降算法 ASGD

输入：最大迭代次数 T，训练集 S；计算分布 p(按照 3.1 节中的方式计算)；初始化 $\alpha=0$(α 是向量，维数和训练样例的个数相同)

输出：α

```
for i=1:1:T
  if 1-γ<y_{i_t}⟨w_t,x_{i_t}⟩<1
    α_{i_t}=α_{i_t}+1
  else y_{i_t}⟨w_t,x_{i_t}⟩≤1-γ
    α_{i_t}=α_{i_t}+2
end for
```

3.1　更新的分布 p 的计算

首先计算分布 p。若通常的随机梯度下降算法直接将不平衡数据集导入程序中运行，则会产生严重的偏差，从而导致分类结果非常差。究其原因，一个很重要的因素就是，数据的不平衡导致其中有一类数据的训练样例的个数远远少于另一类，如果仍旧按照均匀分布来选择样例进行更新，则必然导致多数类被选择的次数远远大于少数类。这里给出分布的目的，即在数据极不平衡的情况下，按照这个分布来选择样例进行更新，使两种样例被选择的次数大致持平。

对于一个随机选择的正类样例 x_+ 和负类样例 x_-，定义两个样例之间的距离：

$$d=\|x_+-x_-\|$$

定义正类样例的个数为 n_+，正类样例的集合为 N_+，负类样例的个数为 n_-，负类样例的集合为 N_-。一个正类样例 x_+ 到负类样例 x_- 的距离定义为：

$$d=\frac{1}{n_-}\sum_{x\in N_-}\|x_+-x\|$$

同理，一个负类样例 x_- 到正类样例 x_+ 的距离定义为：

$$d=\frac{1}{n_+}\sum_{x\in N_+}\|x_+-x\|$$

用 d_+^i 代表第 i 个正类样例到负类样例的距离，将 p_+ 定义为：

$$p_+=(\frac{1}{d_+^1},\frac{1}{d_+^2},\frac{1}{d_+^3},\cdots,\frac{1}{d_+^{n_+-1}},\frac{1}{d_+^{n_+}})$$

然后用 d^i_- 代表第 i 个负类样例到正类样例的距离，将 p_- 定义为：

$$p_+=(\frac{1}{d^1_-},\frac{1}{d^2_-},\frac{1}{d^3_-},\cdots,\frac{1}{d^{n_--1}_-},\frac{1}{d^{n_-}_-})$$

对 p_+ 和 p_- 进行归一化，即使 $\sum_{i=1}^{n_+}\frac{1}{d^i_+}=1$ 和 $\sum_{i=1}^{n_-}\frac{1}{d^i_-}=1$ 成立。然后，$p=[\frac{1}{2}p_+,\frac{1}{2}p_-]$。

分布 p 按照一类样例与另一类样例的距离来决定被选择到的概率的大小，本文选取了距离的倒数，因此距离越远，被选择的概率越小，距离越近，被选择的概率越大；且正负样例被选择的概率各为 50%，因此并不存在均匀分布中选择样例极不平衡的情形。

3.2 停机准则的选取

传统的随机梯度下降方法没有一个明确的停机准则。本文给出了一个简单的根据训练集上的分类准确率和支持向量的变化数的停机准则。

在非常流行的序列最小最优化算法中(SMO)，每迭代一次支持向量的集合有可能增加，随着迭代次数的增加，支持向量的数量不断增加。当然，对于本文中式(1)给出的原问题，λ 就是一个控制正则化项和损失函数项的均衡参数。当 λ 的取值很小时，可以认为损失函数项的系数很大，即模型对于误差的容忍度很低，此时支持向量的个数相对较少。无论 λ 如何取值，支持向量的数量总存在一个极限值，最差的情形是所有的向量都是支持向量。也就是说，当迭代进行到一定次数时，支持向量数量的增加速度开始明显下降，直至收敛，即基本不再增加。关于停机准则的详细讨论将在实验部分给出。

4 实验与结果

实验分为两个部分：1)在小型数据集上对算法进行实验，并且与其他算法进行比较；2)在相对较大的数据集上进行实验，并给出分析。

(1)小型数据集上的实验

在第一部分实验所用到的数据集的信息如表 1 所列，表 1 中的数据来源于 KEEL 的不平衡数据。

表 1 第一部分实验所用到的数据信息

数据名称	样例个数	正类样例个数	负类样例个数	特征数	不平衡比率
Glass1	214	76	138	9	1.82
haberman	306	81	225	3	2.78
pima	768	268	500	8	1.87
Vehicle1	846	217	629	18	2.90
Vehicle2	846	218	628	18	2.88
Vehicle3	846	212	634	18	2.99
Cleveland0vs4	173	13	160	13	12.31
Yeast0256vs3789	1004	99	905	8	9.14
Yeast0359vs78	506	50	456	8	9.12
Abalone19	4174	32	4142	8	129.44
Yeast1458vs7	693	30	663	8	22.10
Yeast2vs8	482	20	462	8	23.10

由于本部分的数据集规模较小，一般的算法训练过程所占用的时间都非常短，因此这里只关注最后的分类准确率。本文采用 G-means 的方法来计算分类准确率，所有的结果都是五折交叉验证的平均值。

基于表 1 中的数据的运行结果如表 2 所列。表 2 对 3 个算法进行了比较，它们分别是本文提出的随机梯度下降算法、Entropy EFSVM 算法和 SVM-SMOTE 算法。从运行的结果来看，本文的随机梯度下降算法在 12 组数据中有大约半数的准确率是最优的，这说明本文算法具有较高的准确率。

表 2 第一部分实验的运行结果

数据名称	ASGD	Entropy FSVM	SVM-SMOTE
Glass1	67.11±7.45	71.40±3.70	49.01±5.10
haberman	67.00±1.60	67.71±6.27	62.00±4.70
pima	70.97±1.58	70.90±3.43	67.41±4.28
Vehicle1	66.40±2.56	66.76±4.21	57.37±4.44
Vehicle2	74.53±3.49	74.53±5.94	65.33±4.25
Vehicle3	69.16±3.93	66.84±2.91	54.97±1.99
Cleveland0vs4	98.83±1.67	74.66±15.21	0
Yeast0256vs3789	74.68±6.98	80.32±4.37	76.86±5.01
Yeast0359vs78	68.67±2.59	71.32±7.81	45.95±4.72
Abalone19	71.45±8.11	56.58±10.93	70.87±4.13
Yeast1458vs7	66.25±5.03	70.38±6.02	55.47±11.57
Yeast2vs8	83.23±3.92	83.26±4.80	69.65±19.65

表 3 第二部分实验所用的数据信息

数据名称	样例个数	正类样例个数	负类样例个数	特征数	不平衡比率
w1atrain	47272	1407	45865	300	32.60
w1atest	2477	72	2405	300	33.40
w2atrain	46279	1372	44907	300	32.73
w2atest	3470	107	3363	300	31.43
w3atrain	44837	1336	43501	300	32.56
w3atest	4912	143	4769	300	33.35
w4atrain	42383	1263	41120	300	32.56
w4atest	7366	216	7150	300	33.10
w5atrain	39861	1198	38663	300	32.27
w5atest	9888	281	9607	300	34.19
w6atrain	32561	954	31607	300	33.13
w6atest	17188	525	16663	300	31.74
jicnntrain	91701	8712	82989	22	9.53
jicnntest	49990	4853	45137	22	9.30

(2)大型数据集上的实验

第二部分实验在较大的数据集上运行算法，由于受到机器硬件的限制，一般的处理不平衡数据的算法无法运行，因此与普通的带有高斯核的随机梯度下降进行比较。首先进行基本的收敛性分析。

在 w6a 数据上进行 150000 次迭代的结果如图 1 所示。

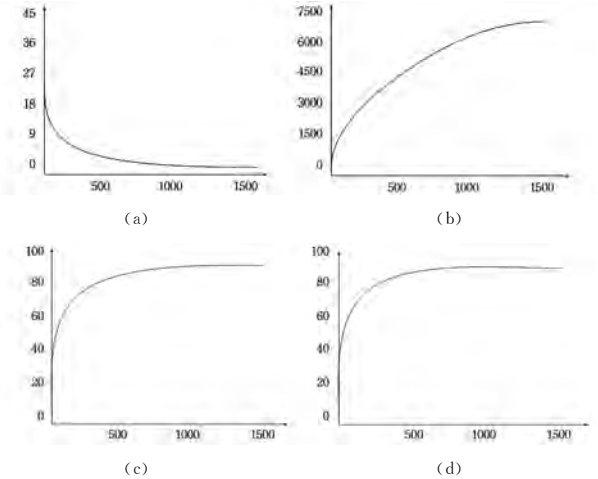

图 1 w6a 数据集上 15 万次迭代时的运行结果

图 1 中横坐标的一个单位代表 100 次迭代。图 1(a)代表每 100 次迭代支持向量的增量，图 1(b)代表支持向量的累计数，图 1(c)代表训练集上的正类样例的分类准确率，图 1(d)代表训练集上的负类样例的分类准确率。从图 1(a)和

图 1(b)可以看出，在大约进行到 7 万次迭代时，图 1(a)中的每 100 次迭代支持向量的增加数趋于 0，而图 1(b)中的支持向量的累计数明显趋于平缓。在进行到 15 万次迭代时，图 1(a)中的数量完全趋于 0，而图 1(b)中的支持向量的累计数基本不再增加。这里可以得出一个结论，如果按照支持向量的增加来评判是否收敛，那么对于本例中的数据集，需要进行 15 万次迭代才可以找到所有的支持向量。

然而，从图 1(c)和图 1(d)可以看出，训练集上的分类准确率的收敛远远早于支持向量累计数的收敛。图 1(c)中正类的分类准确率在 2 万次迭代后基本稳定，图 1(d)中负类的分类准确率在 2 万次迭代后也基本稳定。

因此由图 1 可以看到，在大约 2 万次迭代之后，模型的稳定性已经较好，而在 2 万～15 万的迭代，支持向量的数量虽然还有显著的增加，但是对于模型的影响较小。

在 w6a 数据上进行 2 万次迭代后的结果如图 2 所示。

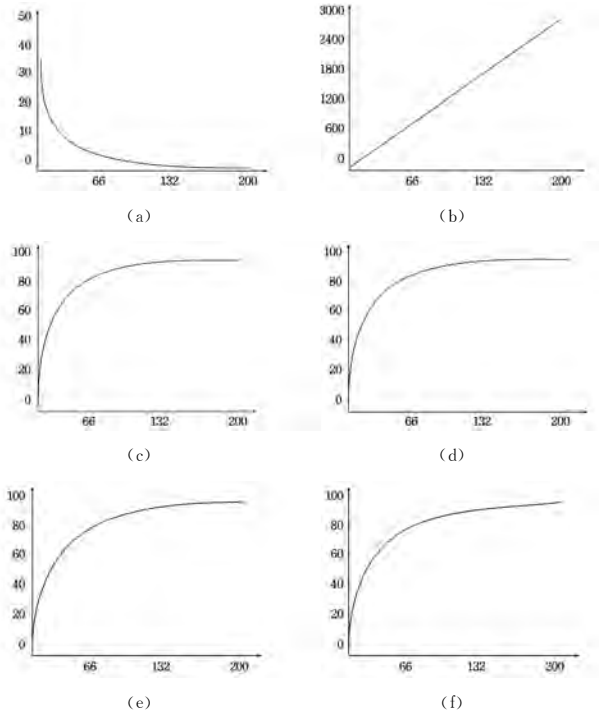

图 2　w6a 数据集上 2 万次迭代时的运行结果

图 2 中横坐标的一个单位代表 100 次迭代。图 2(a)是每 100 次迭代支持向量的增加数，图 2(b)是支持向量的累计数，图 2(c)是训练集上的正类样例的分类准确率，图 2(d)是测试集上的正类样例的分类准确率，图 2(e)是训练集上的负类样例的分类准确率，图 2(f)是测试集上的负类样例的分类准确率。从 2 万次迭代的结果可以看出，相比于支持向量的累计数，分类准确率在 2 万次迭代内就已经收敛。

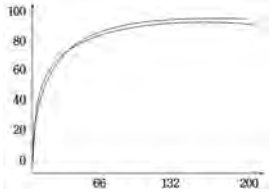

图 3　图 2(c)和图 2(d)的叠加对比

将图 2(c)和图 2(d)叠加，从而得到图 3，图 3 中的一个单位代表 100 次迭代。从图 3 可以看到，在 5000 次迭代以内，训练集和测试集的分类准确率大致是贴合的，但是超过 5000 次迭代后，两条曲线开始分离。另外，训练集上的准确率明显高于测试集上的准确率，这说明迭代次数的增加产生了一定的过拟合。

在 w6a 数据上进行 5000 次迭代所显示的结果如图 4 所示。

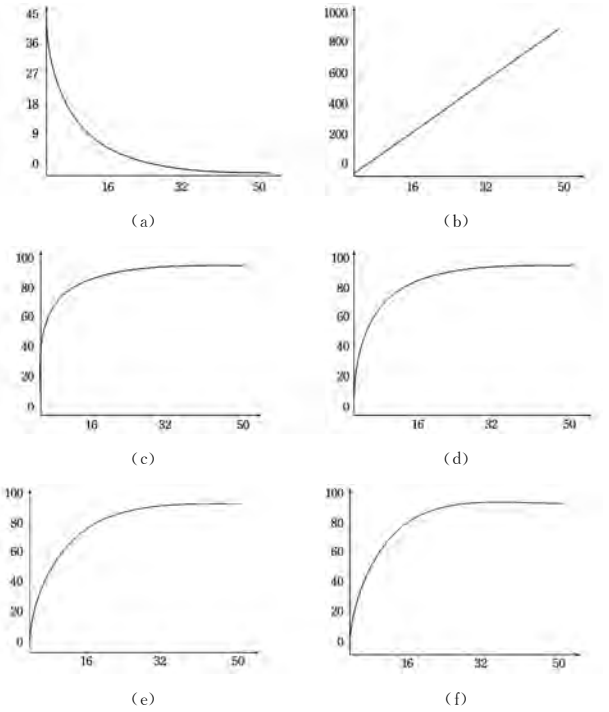

图 4　w6a 数据集上 5000 次迭代的运行结果

图 4 中横坐标的一个单位代表 100 次迭代。图 4(a)是每 100 次迭代支持向量的增量，图 4(b)是支持向量的累计数，图 4(c)是训练集上的正类样例的分类准确率，图 4(d)是测试集上的正类样例的分类准确率，图 4(e)是训练集上的负类样例的分类准确率，图 4(f)是测试集上的负类样例的分类准确率。从图 4(a)和图 4(b)可以看出，在到达 5000 次迭代时，支持向量的累计数远远没有收敛，仍处于快速上升的阶段。从图 4(c)、图 4(d)可以看出，准确率在迭代次数很少时就已经频繁到达峰值。将图 4(c)和图 4(e)累加，从而得到图 5。

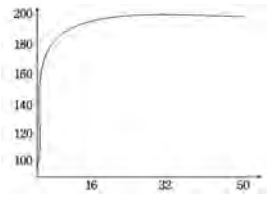

图 5　w6a 数据集上 5000 次迭代时正负样例的分类准确率之和

从图 5 可以看到，在大约 1500 次迭代时两类分类准确率的总和就已经基本稳定，不再增加。

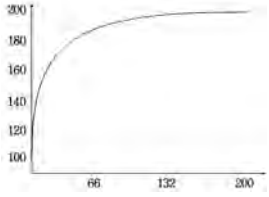

图 6　w6a 数据集上 2 万次迭代时正负样例的分类准确率之和

图 6 是 2 万次迭代的正负样例在训练集上的分类准确率的和。从图中可以看到，在不到 4000 次迭代时，准确率基本达到峰值。

因此,在随机梯度下降算法中采用支持向量的变化数的大小来控制程序在哪一阶段结束,如果执行者并不在乎程序的运算量,而是追求完全的收敛,那么可以将支持向量的变化数设置得小一些,如此,停机时模型就会处于非常稳定的状态。如果执行者输入较大的数据集,若想尽量快地训练模型,那么通过模型在训练集上的分类准确率来进行停机,而把支持向量的变化数设置得尽量大,这样在基本不影响精度的情况下,可以用尽量少的迭代次数来停机。本文使用训练集上的分类准确率来停机的原因是,在迭代初期,训练集上的分类准确率和测试集上的分类准确率是基本贴合的。

表 4 列出了本文算法和普通随机梯度下降算法在这一部分数据集上的分类准确率信息。

表 4 第二部分实验的运行结果

数据	ASGD			SGD			迭代数与训练集规模的比值/%
	正类准确率	负类准确率	几何平均	正类准确率	负类准确率	几何平均	
w1a	87.50	87.73	87.62	35.75	91.40	57.16	5.71
w2a	88.79	87.07	87.92	55.30	89.60	70.39	8.43
w3a	91.61	85.66	88.58	63.84	90.59	76.05	10.71
w4a	79.63	87.12	83.29	39.89	84.29	57.99	1.89
w5a	88.97	85.98	87.46	45.60	92.75	65.03	8.53
w6a	88.76	88.15	88.46	39.20	90.44	59.54	10.44
ijcnn	85.66	87.37	86.51	45.20	86.28	62.45	10.03

由于机器硬件的限制,大部分算法由于表 3 中的数据量过大而无法运行,因此与普通的随机梯度下降算法进行比较。从表 4 中的 SGD 一栏可以看出,我们所选择的所有训练数据在该算法下都是偏向少数类的,进而导致少数类的分类准确率为 0。而本文提出的改进的随机梯度下降明显没有偏向任何一类。表 4 中迭代数与训练集规模的比值是指停机时的迭代次数除以训练集中的样例个数。从这一列数据可以看出,我们的算法在大多数据集合上大约 10%就停机了,并且较多明显小于 10%。这说明停机的时间确实是在迭代的初期。

结束语 本文算法使用一个分布 p 来代替原来的均匀分布以选择样例,实验证明,这种分布确实能够修正原始算法偏向少数类的问题,并且无论训练数据的不平衡比率如何波动,结果都是很稳定的。也就是说,本文算法对于训练集的不平衡比率不敏感,而且这种分布是嵌入式的。只要算法是"随机"(stochastic)的,该分布都可以直接嵌入到其他算法之中。其次,本文算法使用的损失函数是光滑化的绞链损失函数,这种函数不存在不可导的点,并且还有一个参数 γ。这意味着对于不同的 γ 的选取,可以给不同的支持向量对应的 α 的分量以不同的权重,提升算法的灵活性。对于适当的 γ 的选取,可以进一步提高分类准确率。再次,本文以训练集的分类准确率的使用作为停机的标准,使得程序在迭代早期就可以停止,并且不会降低分类器的泛化能力。最后,本文中的算法保留了随机梯度下降的一个最大的优势,即它相对较低的计算复杂度,这就导致运行时间相对较短。这对于处理较大规模的数据是一个很大的优势。

参 考 文 献

[1] CORTES C, VAPNIK V. Support-vector networks [J]. Machine Learning, 1995, 20(3): 273-297.

[2] PLATT J C. Sequential Minimal Optimization : A Fast Algorithm for Training Support Vector Machines [J]. Technical Report, 1998, 208(1): 212-223.

[3] WRIGHT S J. Coordinate Descent Algorithms [J]. Mathematical Programming, 2015, 151(1): 3-34.

[4] NESTEROV Y, STICH S U. Efficiency of the Accelerated Coordinate Descent Method on Structured Optimization Problems [J]. Core Discussion Papers, 2016, 27(1): 110-123.

[5] SHALEV-SHWARTZ S, ZHANG T. Accelerated Proximal Stochastic Dual Coordinate Ascent for regularized Loss Minimization [J]. Mathematical Programming, 2016, 155(1/2): 105-145.

[6] SHALEV-SHWATRZ S, ZHANG T. Stochastic Dual Coordinate Ascent Methods for Regularized Loss Minimization [J]. Journal of Machine learning Research, 2012, 14(1): 2013.

[7] CSIBA Q, ZHENG Q, RICHTARIK Q. Stochastic Dual Coordinate Ascent with Adaptive Probabilities [C] // International Conference on Machine Learning. 2015: 674-683.

[8] WANG X, ZHANG W, YAN J, et al. On the Flexibility of Block Coordinate Descent for Large-Scale Optimization [J]. Neurocomputing, 2018, 272(10): 471-480.

[9] JOHNSON R, ZHANG T. Accelerating Stochastic Gradient Using Predictive Variance Reduction [C] // International Conference on Neural Information Processing Systems. 2013: 315-232.

[10] SHALEV-SHWARTZ S, SINGER Y. Primal Estimated Subgradient Solver for SVM [J]. Mathematical Programming, 2011, 127(1): 3-30.

[11] LIN C F, WANG S D. Fuzzy support vector machines [J]. IEEE Trans. Neural Network, 2002, 13(2): 464-471.

[12] FAN Q, WANG Z, LI D, et al. Entropy-based fuzzy support vector machine for imbalanced data-sets [J]. Knowledge-Based Systems, 2017, 115(1): 87-89.

[13] ZHANG T, ZHOU Z H. Large margin distribution machine [C] // Acm Sigkdd International Conference on Knowledge Discovery & Data Mining. 2014: 313-322.

[14] CHENG F, ZHANG J, WEN C, et al. Large Cost-Sensitive Margin Distribution Machine for Imbalanced Data Classification [J]. Neruocomputing, 2016, 224(8): 45-57.

[15] CHAWLA N V, BOWYER K W, HALL L O, et al. SMOTE: synthetic minority over-sampling technique [J]. Journal of Artificial Intelligence Research, 2002, 16(1): 321-357.

[16] GALAR M, BARRENECHEA E, HERRERA F. EUSBoost: Enhancing ensembles for highly imbalanced data-sets by evolutionary under-sampling [J]. Pattern Recognition, 2013, 46(12): 3460-3471.

[17] LIU X Y, WU J, ZHOU Z H. Exploratory Under-Sampling for Class-Imbalanced Learning [J]. IEEE Transactions on Systems Man & Cybernetics Part B Cybernetics A publication of the IEEE Systems Man & Cybernetics Society, 2009, 39(2): 539-550.

[18] KUBAT M, MATWIN S. Addressing the Curse of Imbalanced Training Sets: One-Sided Selection [C] // International Conference on Machine Learning. 2012: 179-186.