超并行机器学习与海量数据挖掘

贾冬雨 孙脩然

实验一

命令	想法	Accuracy
train	L2R_L2LOSS_SVC_DUAL	96.49%
train -w1 1 -w-1 3	不平衡	95.96%
train -s 4	不同的 svm,MCSVM_CS	96.45%
train -B 1	增加 bias	96.46%

- 由于训练样本中 A 类样 本数量少于非 A 类,我们思考认为调节对不同类别样本的罚值权重或许可以对平衡问题 有所帮助,由于 A 类和非 A 类的比例为 1:3 左右。 即对于 A 类问题,罚值权重为 3, 非 A 类罚值权重为 1,希望这种方式可以部分解决非平衡问题。然而实验结果告诉我们 并不是这样。

修改liblinear 源代码

- 修改liblinear源代码,直接输出预测值,而不是分类结果.
- 修改 liblinear 源代码,使之能够读入两个文件,并在内部 随机的对输入样本进行重排序.
- 使用MPI多进程方案分别训练每个子问题,制作并行训练。
- minmax 预测方法直接通过多个模型生成预测结果.不需要存储每个模型的中间结果。

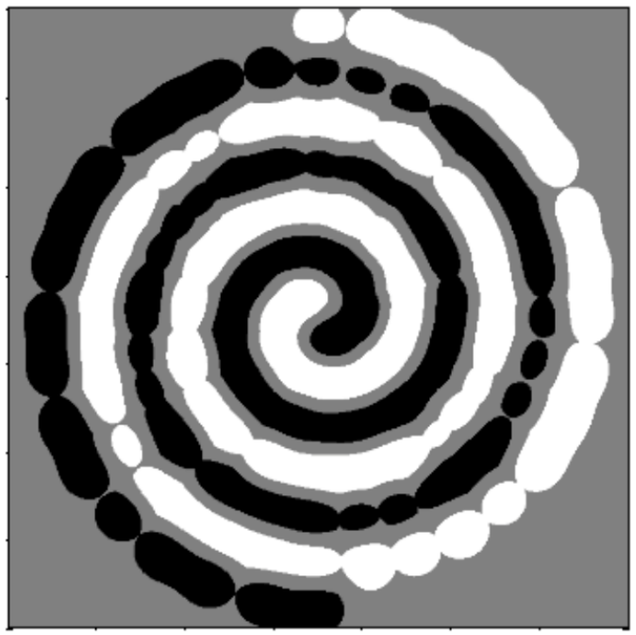
最大最小模块化网络

- 完全先验
- A类先验,其他随机
- A类随机,其他先验
- 引入重叠

因为第二问中随机结果偶然的出现了98%的正确率，所以我们决定实验随机和先验结合的情况。

引入重叠

Boundaries of M^3-SVM Using Three Methods



random

Y-axis

Y-axis & overlap*

*overlap means two subsets share the training data along the y-axis

分组与结果

A is split to 3 groups.

[0,46]

[47,61]

[62,63]

B&D is split to 7 groups:

B[0,21]；B[22,27]；

B[28,32][42,44]；

B[41]；B[60]；B[65]；

B[61-64][66-82]D；

C is split to 2 groups:

C[1,8]

C[9,30]

我们用C++ stl 中的 map 来统计不同标号的出现数目，并且累加结果。

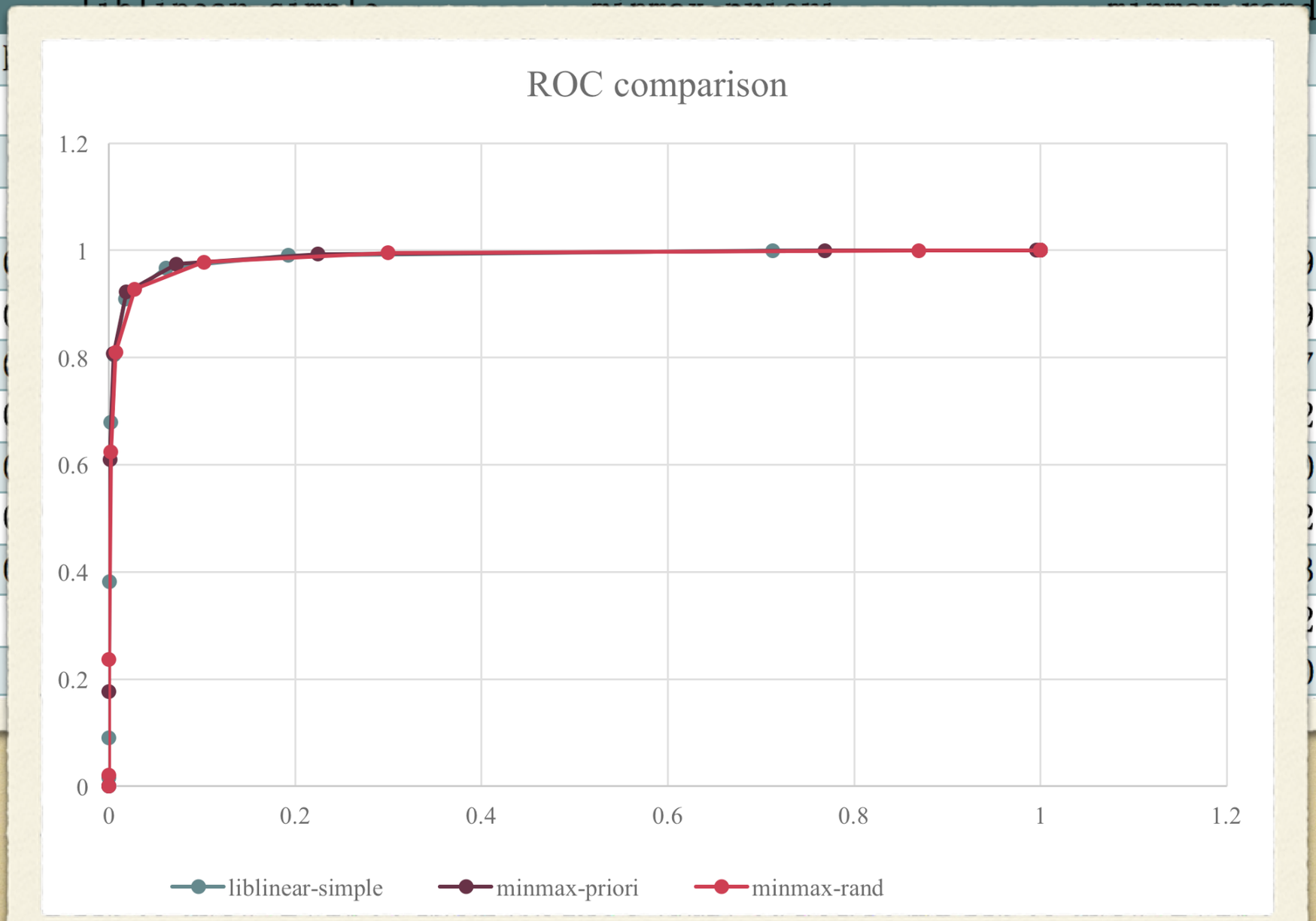
类别	Accuracy	时间	F1
A 类先验，其他随机	96.4881%	7.56s	0.926068
A 类随机，其他先验	96.1282%	7.24s	0.921407
完全先验	96.7025%	7.45s	0.931236
完全随机	96.1573%	7.36s	0.921207
引入重叠（少）	96.7634%	7.96s	0.932554
引入重叠（多）	96.6866%	9.59s	0.931216

首先，我们把单独标号数目较大者拿出来，单独算作一组，然后把连续的标号分为一组.

我们对边界部分的重叠了 200 个左右的样本,可以看 到升还是了准确率和 F1 值。当引入重叠较多时,我们重叠了 1000 个左右的样本,准 确率轻微下降了。原因可能是对于引入重叠较多的情况下,子问题规模变大,复杂度上 升,线性方法刻画难度上升导致的。而对于少量的重叠,可以较好地处理边界情况而导 致准确度上升。

ROC

thd			
−8			1
−6			1
−4			1
−2			9672131
−1			5628415
−0.5			7923497
0			27759563
0.5			09508197
1			24043716
2			36830601
4			21420765
6			00010929



训练时间分析

类别	第一次执行时间（s）	第二次执行时间(s)	第三次执行时间(s)	二三次平均时间(s)
串行（1 个子问题）	8.98	6.01	6.12	6.065
串行（27 个子问题）	20.76	16.63	17.16	16.895
并行（随机 27 个子问题）	13.2	7.336	7.23	7.283

- 由于我们目前的计算机体系结构，反复命中可以大大提升程序执行效率，因为数据都在内存或者cache里面。每次运行时间可能不一样，第一次时间往往较长，我们采用第二第三次的测试时间平均值分析。
- 并行化程度来看，在27个子问题的程序中，并行程序的执行速度大约是串行的两倍，这也许是因为虚拟机仅有两个核的缘故。

LMS & Perceptron

修改了 liblinear 源代码, . 这样只用调用参数 –s 14 和 –s 15 就可 以使用 Perceptron 和 LMS 分类器.

model 存储方式保持与原来一致，从而保证兼容。

LMS Learning Rule

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha \nabla F(\mathbf{x})\Big|_{\mathbf{x}=\mathbf{x}_k}$$

$$\mathbf{x}_{k+1} = \mathbf{x}_k + 2\alpha e(k)\mathbf{z}(k)$$

$${}_1\mathbf{w}(k+1) = {}_1\mathbf{w}(k) + 2\alpha e(k)\mathbf{p}(k)$$

$$b(k+1) = b(k) + 2\alpha e(k)$$

Perceptron Learning Rule

$${}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} + e\mathbf{p} = {}_1\mathbf{w}^{old} + (t-a)\mathbf{p}$$

$$b^{new} = b^{old} + e$$

Sequential Mode VS Batch Mode

种类	Accuracy
Perceptron Sequential	89.3%
Perceptron Batch	95.7947%
LMS Sequential	88.1%
LMS Batch	93.831%

- 我们刚开始使用的是Sequential mode ,后来发现准确率不高，改用Batchmode 后大大提升了准确率。
- LMS Sequential 之所以这么低并不全是Sequential mode 决定的，而是由于参数选取不合理，学习系数选取不合理导致了收敛过慢。
- LMS 比Perceptron 低是因为学习系数选取较小，其收敛不到位导致的，并不是LMS 本身比较差

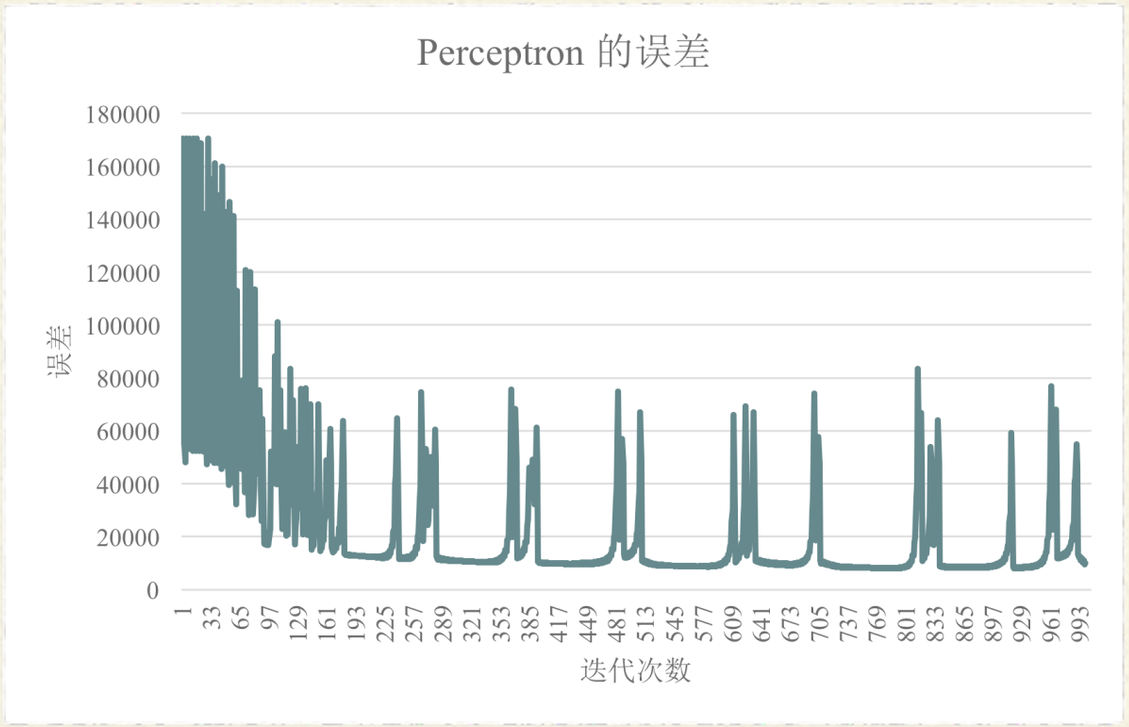
结果分析

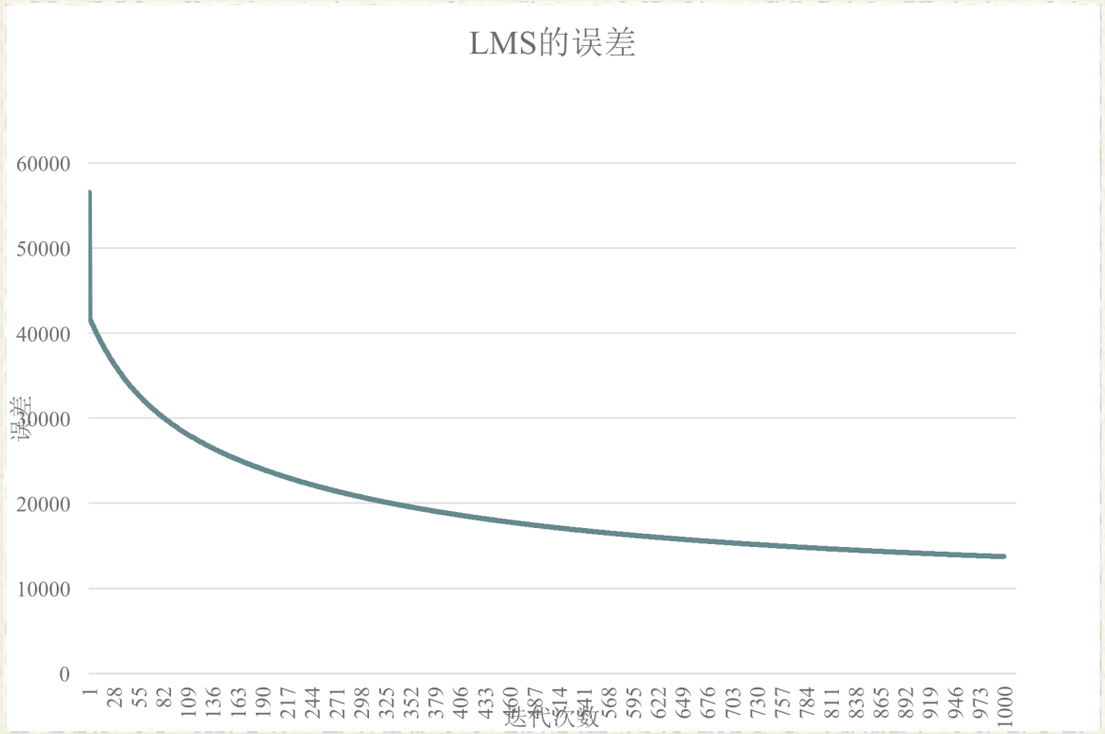
种类	子问题数	Accuracy	时间(s)
LMS	1	93.831%	32.36
Perceptron	1	95.7947%	19.5
LMS(随机)	27	94.4609%	77.8
Perceptron（随机）	27	95.8741%	30.07
LMS（先验）	27	95.6677%	101.57
Perceptron（先验）	27	96.3558%	32.97

- 先验>随机>单一问题， Perceptron 好于LMS, 这和我们经验中的结果是不一样的。分析肯能有以下几种可能。一是我们的步长选取不合适，导致收敛速度慢或者总是过冲震荡。二是我们选取得迭代次数不够多，三是我们选取的误差上限过大。

- 时间上Lms 大于同组Perceptron 时间。

迭代次数分析





- 在200次左右，Perceptron 就已经收敛到较好水平。LMS 实际上在1000次时还有降低误差的余地。
- 可以进一步调大学习系数，来弥补收敛过慢的问题
- Perceptron 中由于没有学习系数的概念，每次修改可能会对结果影响很大，导致结果不稳定。

MLP分类器

- 5个hidden node
- 1个output node
- 5001维输入
- 激活函数sigmoid

$$\begin{pmatrix} Weight \\ correction \\ \Delta w_{ij}(n) \end{pmatrix} = \begin{pmatrix} Learning \\ parameter \\ \eta \end{pmatrix} \begin{pmatrix} Local \\ gradient \\ \delta_j(n) \end{pmatrix} \begin{pmatrix} Input\ signal \\ of\ neuron\ j \\ y_i(n) \end{pmatrix}$$

- The local gradient is given by

$$\delta_j(n) = e_j(n)\varphi_j'(v_j(n)) \tag{4.14}$$

when the neuron j is in the output layer.

- In the hidden layer, the local gradient is

$$\delta_j(n) = \varphi_j'(v_j(n)) \sum_k \delta_k(n) w_{kj}(n) \tag{4.24}$$

computed recursively from the local gradients of the following layer, back-propagating error

MLP

类别	任务数	时间	准确率
MLP	1	108.2s	82.52%
MLP 并行随机	27	245.4s	85.21%
MLP 并行先验	27	236.23s	86.35%

- 然而由于问题是分线性的，一次迭代就要花费数十秒时间，给我们的训练带来很大难度，仅仅迭代10次。
- 如果全部猜非A，正确率75%，所以提升并不明显,由于训练时间比较长，我们没有进行次数更多的迭代，另外学习参数的选取也仅仅是根据结果的人为判断，没有科学的依据，所以准确率不是很尽如人意。但是更多的迭代次数或许能够带来更高的准确率。