LibLinear类说明文档

郭沛 2015.05.18

LibLinear类是对台湾大学Chih-Jen Lin博士实验室开源的liblinear代码进行打包封装而来的，主要的改进是屏蔽了很多实现细节，只把最主要的功能暴露给用户，同时与OpenCV的Mat格式进行兼容。

训练接口

```
void LibLinear::train(Mat &trainingdataMat, Mat &labelsMat, parameter &param);
```

trainingdataMat是一个M*N的矩阵，包含M个训练样本，按照行排列，每个样本有N维特征。labelsMat是一个M*1的列向量，每一行是对应样本的标签。param是训练参数。

如果训练样本和标签值为float*形式的裸数据，请使用下面的接口:

```
void LibLinear::train(const float *data, const float *label,
           int nSamples, int nFeatures, parameter &param);
```

data指向训练数据，是一个nSample*nFeatures大小的一维float数组，nSamples代表样本数，nFeatures代表特征数。label指向测试数据，是一个nSamples大小的一维float数组。param是训练参数。

训练参数的构造方法如下:

```
static parameter LinearParam::construct_param();
```

这是默认的构造函数，使用编号为1的solver，即带L2约束与L2损失函数的SVM。在大多数情况下，该参数表现的很好，如果感觉速度比较慢，可以换成编号为2的solver。可以用下面的构造函数实现:

```
static parameter LinearParam::construct_param(int solver_type);
```

该接口只需要通过solver_type指定不同的solver编号，剩下的参数将由默认参数填充。solver共有八种选择（0~7），具体的选择方法见FAQ。如果你对parameter的结构很了解，想做更多的自定义，请使用下面的接口:

```
static parameter LinearParam::construct_param(int solver_type,
    double eps,
    double C,
    int nr_weight,
    int *weight_label,
    double *weight,
    double p);
```

值得注意的是，parameter内分配的weight_label和weight空间应该由用户自己回收，请使用下面的接口：

```
static void LinearParam::destroy_param(parameter *param);
```

保存模型请用：

```
void LibLinear::save_model(string model_file_name);
```

model_file_name是模型文件的名字。

加载训练模型请用：

```
void LibLinear::load_model(string model_file_name);
```

model_file_name是模型文件的名字。

测试接口

```
double LibLinear::predict(Mat &SampleMat);
```

该函数用于预测单个训练样本的标签值。SampleMat为输入的1*N的行向量，代表单一样本的N维特征，输出为该样本的预测标签值。如果需要预测多个训练样本的标签值，请使用：

```
void LibLinear::predict(Mat &SamplesMat, Mat &OutputMat);
```

其中，SamplesMat为输入的测试样本，是一个M*N的矩阵，每行代表一个训练样本，每个样本特征为N维。OutputMat是一个M*1的矩阵，每行代表对应样本的预测标签值。
在实际中，我们通常需要将一个向量形式的样本，通过分类器转化为一个标量，这个标量既可以是样本到分类面的距离，即Sigma(w*x)，又可以代表样本的分类概率。实际上，在liblinear

中样本分类概率就是通过样本到分类面距离通过sigmoid函数得到的。同时，liblinear的一个局限在于，只有选择solver 0或者7，即逻辑回归的分类器，才可以输出预测概率值，因此，推荐使用样本到分类面距离这个标量。具体的接口如下：

```
double LibLinear::predict_values(Mat &SampleMat, Mat &ValueMat);
```

其中，SamplesMat是输入的待预测的样本，是1*N的行向量，ValueMat是返回的该样本到分类面的距离，是1*K的矩阵，K根据分类标签数确定，对于二分类，则只返回一个距离，正是我们想要的标量值。返回值是样本的分类标签。

如果待测试的数据是float*形式的裸数据，请使用下面的接口：

```
double LibLinear::predict_values(float *sample, int nFeatures, float *value);
```

其中，sample为待测试的样本，是nFeature大小的一维float数组。样本到分类面的距离将会保存到value中。返回值是样本的分类标签。

如果想要得到分类的概率，请使用下面的接口：

```
double LibLinear::predict_probabilities(Mat &SampleMat, Mat &ProbMat);
```

SamplesMat是输入的待预测的样本，是1*N的行向量，ProbMat是返回的该样本的分类概率，是1*K的矩阵，K根据分类标签数确定，对于二分类，则只有一个距离，正是我们想要的标量值。注意，该接口只有在选择solver 0或者7时正常得到概率值，其他情况下ProbMat为0。返回值是样本的分类标签。

其它接口

如果想要输出模型文件中的分类权重，请使用如下接口：

```
void LibLinear::get_w(double *val);
```

注意的是，val需要由用户自己开辟并释放空间。

默认的析构函数可能不会及时释放内存空间，影响程序的正常运行。因此，推荐使用如下的函数进行内存释放：

```
void LibLinear::release();
```

FAQ

Q：在什么情况下使用liblinear?
A：当样本的维度比较高时（例如上千到上万），可以使用liblinear，它的优势在于保证精度的前提下，大大提高训练和预测的速度。

Q：libsvm和liblinear的比较?
A：举例说明，在new20数据上进行实验，样本维数为62061，训练样本15935个，测试样本3993个。libsvm使用线性核，训练时间为2m26s，测试时间为34s，测试精度为84.0%。liblinear采用默认的solver 1，训练时间为2.9s，测试时间为1.2s，测试精度为85.4%。liblinear速度快了30倍。

Q：liblinear训练参数如何选择?
A：liblinear提供了8种分类的solver，它们的区别主要在于损失函数和约束函数的不同。下面介绍推荐的参数选择方法：首先选择默认solver 1， 如果速度慢，可以选择solver 2。如果想要输出分类概率，只能选择solver 0 或者 7。同时应该注意过大的C以及未经过归一化的样本都可能导致训练速度变慢。
这8种solver如下：

- 0 – 带L2约束的逻辑回归（主问题）
- 1 – 带L2约束与L2损失函数的SVM（对偶问题）
- 2 – 带L2约束与L2损失函数的SVM（主问题）
- 3 – 带L2约束与L1损失函数的SVM（对偶问题）
- 4 – Crammer and Singer的SVM算法
- 5 – 带L1约束与L2损失函数的SVM
- 6 – 带L1约束的逻辑回归
- 7 – 带L2约束的逻辑回归（对偶问题）

在new20数据的例子中，不同solver的速度和精度比较如下：

solver种类	**S0**	**S1**	**S2**	**S3**	**S4**	**S5**	**S6**	**S7**
训练时间	7.9s	2.9s	6.9s	2.8s	2.5s	6.4s	5.8s	3.7s
测试时间	1.5s	1.2s	1.5s	0.9s	1.0s	0.6s	0.6s	1.5s
测试精度	82.9%	85.4%	85.4%	85.1%	85.2%	82.0%	75.9%	82.9%

可见不同solver之间的性能相差不大。带L1约束的solver会给w带来稀疏性，可以用来挑选特征。

Q：有没有Demo程序？

A：LibLinearObj工程的main.cpp中，包含了上述所有接口的演示。该程序中，正样本全部为y轴左侧，负样本全部为y轴右侧，所以分类面应该是y轴。