LibSVM for Python 使用

经历手写SVM的惨烈教训(还是太年轻)之后，我决定使用工具箱/第三方库

Python

libsvm的GitHub仓库

LibSVM是开源的SVM实现，支持C， C++， Java，Python ， R 和 Matlab 等, 这里选择使用Python版本。

安装LibSVM

将LibSVM仓库的所有内容放入Python的包目录\Lib\site-packages或者工程目录中。

在libsvm根目录和python子目录下中分别新建名为 `__init__.py` 的空文件，这两个空文件将标识所在的目录为python包可以直接导入。

允许草民吐槽一下各种Blog里切换根目录的奇怪的解决方案：这个和这个

因为经常使用svm，所以草民将libsvm包放入\Lib\site-packages目录下。在Python交互环境或在任意脚本中都可以使用 `import libsvm.python` 来使用libsvm的python接口。

使用LibSVM

LibSVM的使用非常简单，只需调用有限的接口

示例1：

```
from libsvm.python.svmutil import *
from libsvm.python.svm import *

y, x = [1,-1], [{1:1, 2:1}, {1:-1,2:-1}]
prob  = svm_problem(y, x)
param = svm_parameter('-t 0 -c 4 -b 1')
model = svm_train(prob, param)
yt = [1]
xt = [{1:1, 2:1}]
p_label, p_acc, p_val = svm_predict(yt, xt, model)
print(p_label)
```

输出结果：

```
optimization finished, #iter = 1
nu = 0.062500
obj = -0.250000, rho = 0.000000
nSV = 2, nBSV = 0
Total nSV = 2
test:
Model supports probability estimates, but disabled in predicton.
Accuracy = 100% (1/1) (classification)
[1.0]
```

在SVM数据中下载train1.txt和test1.txt。

LibSVM可以在文件中读取训练数据，这样便于大规模数据的使用。

示例：

```
from libsvm.python.svmutil import *
from libsvm.python.svm import *

y, x = svm_read_problem('train1.txt')
yt, xt = svm_read_problem('test1.txt')
model = svm_train(y, x )
print('test:')
p_label, p_acc, p_val = svm_predict(yt[200:202], xt[200:202], model)
print(p_label)
```

可以看到输出：

```
optimization finished, #iter = 5371
nu = 0.606150
obj = -1061.528918, rho = -0.495266
nSV = 3053, nBSV = 722
Total nSV = 3053
test:
Accuracy = 40.809% (907/2225) (classification)
```

LibSVM接口

训练数据格式

libsvm的训练数据格式如下：

```
<label> <index1>:<value1> <index2>:<value2> ...
```

示例：

```
1 1:2.927699e+01 2:1.072510e+02 3:1.149632e-01 4:1.077885e+02
```

主要类型

- `svm_problem`

保存定义SVM模型的训练数据

- `svm_parameter`

存储训练SVM模型所需的各种参数

- `svm_model`

完成训练的SVM模型

- `svm_node`

模型中一个特征的值，只包含一个整数索引和一个浮点值属性。

主要接口：

- `svm_problem(y, x)`

由训练数据y,x创建svm_problem对象

- `svm_train()`

svm_train有3个重载：

```
model = svm_train(y, x [, 'training_options'])
model = svm_train(prob [, 'training_options'])
model = svm_train(prob, param)
```

用于训练svm_model模型

- `svm_parameter(cmd)

创建svm_parameter对象，参数为字符串。

示例：

```
param = svm_parameter('-t 0 -c 4 -b 1')
```

- `svm_predict()`

调用语法：

```
p_labs, p_acc, p_vals = svm_predict(y, x, model [,'predicting_options'])
```

参数：

`y` 测试数据的标签

`x` 测试数据的输入向量

`model` 为训练好的SVM模型。

返回值：

`p_labs` 是存储预测标签的列表。

`p_acc` 存储了预测的精确度，均值和回归的平方相关系数。

`p_vals` 在指定参数'-b 1'时将返回判定系数(判定的可靠程度)。

这个函数不仅是测试用的接口，也是应用状态下进行分类的接口。比较奇葩的是需要输入测试标签y才能进行预测，因为y不影响预测结果可以用0向量代替。

- `svm_read_problem`

读取LibSVM格式的训练数据：

```
y, x = svm_read_problem('data.txt')
```

- `svm_save_model`

将训练好的svm_model存储到文件中：

```
svm_save_model('model_file', model)
```

model_file的内容：

```
svm_type c_svc
kernel_type linear
nr_class 2
total_sv 2
rho 0
label 1 -1
probA 0.693147
probB 2.3919e-16
nr_sv 1 1
SV
0.25 1:1 2:1
-0.25 1:-1 2:-1
```

- `svm_load_model`

读取存储在文件中的svm_model:

```
model = svm_load_model('model_file')
```

调整SVM参数

LibSVM在训练和预测过程中需要一系列参数来调整控制。

svm_train的参数：

- `-s` SVM的类型(svm_type)
  - 0 -- C-SVC(默认)

    使用惩罚因子(Cost)的处理噪声的多分类器
  - 1 -- nu-SVC(多分类器)

    按照错误样本比例处理噪声的多分类器
  - 2 -- one-class SVM

    一类支持向量机，可参见"SVDD"的相关内容
  - 3 -- epsilon-SVR(回归)

    epsilon支持向量回归
  - 4 -- nu-SVR(回归)

- `-t` 核函数类型(kernel_type)
  - 0 -- linear(线性核):

    `u'*v`
  - 1 -- polynomial(多项式核):

    `(gamma*u'*v + coef0)^degree`
  - 2 -- radial basis function(RBF,径向基核/高斯核):

    `exp(-gamma*|u-v|^2)`
  - 3 -- sigmoid(S型核):

    `tanh(gamma*u'*v + coef0)`
  - 4 -- precomputed kernel(预计算核)：

    核矩阵存储在 `training_set_file` 中

下面是调整SVM或核函数中参数的选项：

- `-d` 调整核函数的degree参数，默认为3
- `-g` 调整核函数的gamma参数，默认为 `1/num_features`
- `-r` 调整核函数的coef0参数，默认为 `0`
- `-c` 调整C-SVC, epsilon-SVR 和 nu-SVR中的Cost参数，默认为 `1`
- `-n` 调整nu-SVC, one-class SVM 和 nu-SVR中的错误率nu参数，默认为 `0.5`
- `-p` 调整epsilon-SVR的loss function中的epsilon参数，默认 `0.1`
- `-m` 调整内缓冲区大小,以MB为单位，默认 `100`
- `-e` 调整终止判据，默认 `0.001`
- `-wi` 调整C-SVC中第i个特征的Cost参数

调整算法功能的选项：

- `-b` 是否估算正确概率,取值0 - 1，默认为 `0`
- `-h` 是否使用收缩启发式算法(shrinking heuristics),取值0 - 1，默认为 `0`
- `-v` 交叉校验
- `-q` 静默模式

Matlab

LibSVM的Matlab接口用法类似，Matlab丰富的标准工具箱提供了各种方便。

Statistic Tools工具箱提供了svmtrain和svmclassify函数进行SVM分类。

```
traindata = [0 1; -1 0; 2 2; 3 3; -2 -1;-4.5 -4; 2 -1; -1 -3];
group = [1 1 -1 -1 1 1 -1 -1]';
testdata = [5 2;3 1;-4 -3];
svm_struct = svmtrain(traindata,group);
Group = svmclassify(svm_struct,testdata);
```

svmtrain接受traindata和group两个参数，traindata以一行表示一个样本，group是与traindata中样本对应的分类结果，用1和-1表示。

svmtrain返回一个存储了训练好的svm所需的参数的结构体svm_struct。

svmclassify接受svm_struct和以一行表示一个样本的testdata，并以1和-1列向量的形式返回分类结果。

Keep working, we will find a way out. This is Finley, welcome to join us.

分类: MachineLearning

-Finley-
关注 - 8
粉丝 - 20
+加关注

1 推荐　0 反对

« 上一篇：支持向量机原理

» 下一篇：朴素贝叶斯分类器及Python实现

posted @ 2016-03-28 16:07 -Finley- 阅读(2577) 评论(0) 编辑 收藏

刷新评论　刷新页面　返回顶部

注册用户登录后才能发表评论，请 登录 或 注册，访问网站首页。

最新IT新闻:

- 新改变：在搜索引擎中直接提交网址到Google索引中
- 都有哪些改变？Windows 10 Creators Update详细上手
- 微软收购软件容器提供商Deis
- 外卖决战：“掉队者”百度左右战局？
- 13张美国首次登月照：从训练到回收再到游行庆祝

» 更多新闻...

最新知识库文章:

- 如何打好前端游击战
- 技术文章的阅读姿势
- 马拉松式学习与技术人员的成长性
- 程序员的“认知失调”
- 为什么有的人工作多年还是老样子

» 更多知识库文章...