机器学习作业-SVM

软件 51 庞建业 2151601012

https://github.com/sherjy/Notes-RL

本次主要针对 SVM 分类方法的多种核方法的实现的学习和复习

SVM 分类方法简介

支持向量机（**support vector machines**，SVM）是一种二分类模型，是一种 **Supervised Learning**，是一个类似于逻辑回归的方法，用于对不同因素影响的某个结果的分类。
但逻辑回归主要采用的是 **sigmoid** 函数，**SVM** 有自己常用的核函数：**linear** 线性核、**rbf** 径向基、**poly** 多项式。

环境：

Ubuntu 16.04

Atom+Anaconda3.6

Kernel Trick

1.**linear**
简单的线性核函数，上文提到的香蕉和黄瓜的分类就是线性核函数的应用
主要应用于线性可分的情况

2.**rbf**
高斯核函数，可以实现高维投射

3.**poly**
多项式核函数也是可以实现低维到高维的映射，采用的频率并不是很高

SVM 关键是选取这些核函数的类型，主要有线性内核，多项式内核，径向基内核（**RBF**），**Sigmoid** 核。
这些函数中应用最广的应该就是 **RBF** 核了，无论是小样本还是大样本，高维还是低维等情况，**RBF** 核函数均适用，它相比其他的函数有一下优点：
1）RBF 核函数可以将一个样本映射到一个更高维的空间，而且线性核函数是 **RBF** 的一个特例，也就是说如果考虑使用 **RBF**，那么就没有必要考虑线性核函数了。
2）与多项式核函数相比，**RBF** 需要确定的参数要少，核函数参数的多少直接影响函数的复杂程度。另外，当多项式的阶数比较高时，核矩阵的元素值将趋于无穷大或无穷小，而 **RBF** 则在上，会减少数值的计算困难。
3）对于某些参数，**RBF** 和 **sigmoid** 具有相似的性能。

代码：

我们先看几个 demo
1.使用 SVM with liner kernel 的方法做 PCA 与 CCA 的多标签分类任务
见文件夹中 svm-多标签分类.py

```python
import numpy as np
import matplotlib.pyplot as plt

from sklearn.datasets import make_multilabel_classification
from sklearn.multiclass import OneVsRestClassifier
from sklearn.svm import SVC
from sklearn.preprocessing import LabelBinarizer
from sklearn.decomposition import PCA
from sklearn.cross_decomposition import CCA


def plot_hyperplane(clf, min_x, max_x, linestyle, label):
    w = clf.coef_[0]
    a = -w[0] / w[1]
    xx = np.linspace(min_x - 5, max_x + 5)
    yy = a * xx - (clf.intercept_[0]) / w[1]
    plt.plot(xx, yy, linestyle, label=label)

```

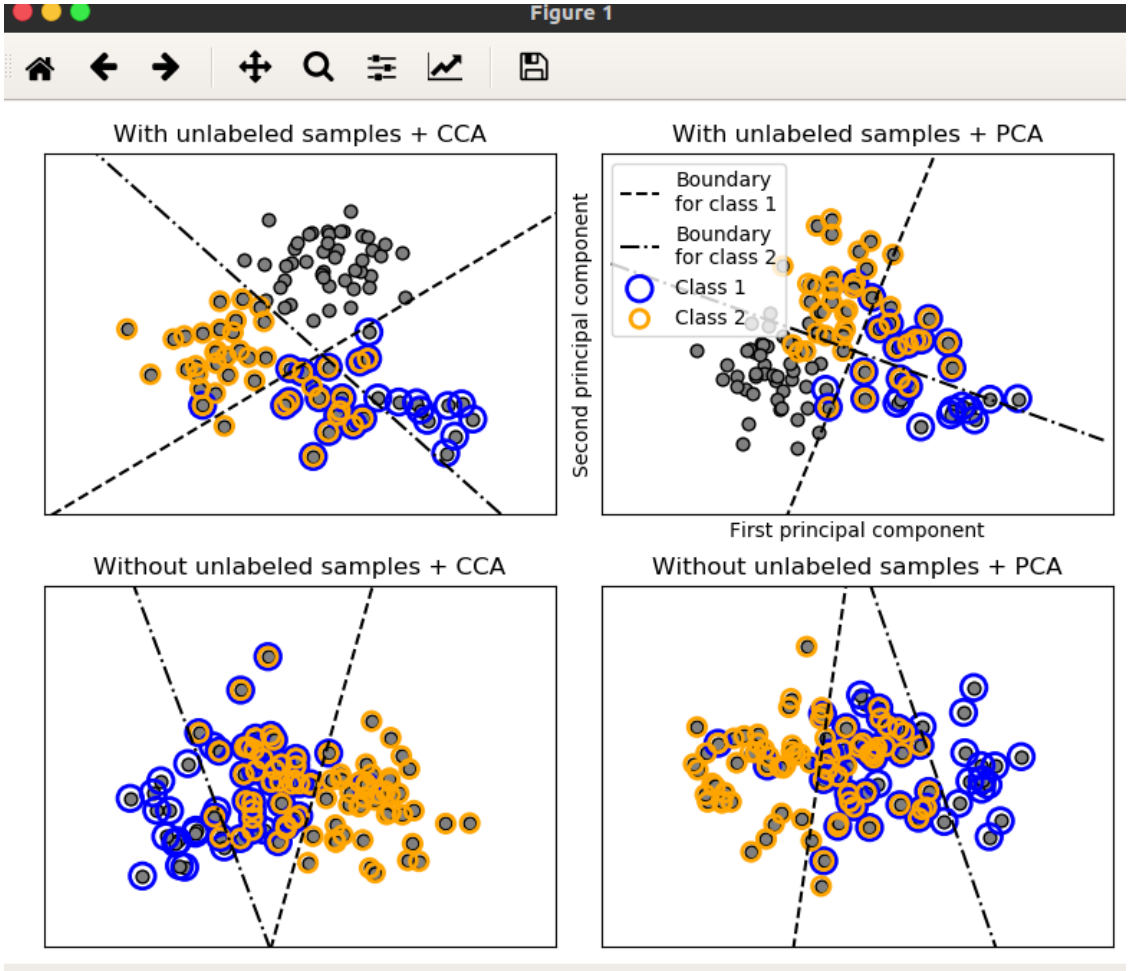
Combo
Max 5511

Linter

Severity	Provider	Description	Line

Atom Runner: M4.py

Atom Runner: svm-多标签分类.py



备注：

上面是使用线性核的方法 `sklearn.multiclass.OneVsRestClassifier` 做 **PCA** 与 **CCA** 的多标签分类任务，也是处理大规模数据集的最简单的思想和方法
其中
PCA(principal component analysis) 就是我们熟知的主成分分析方法，一般我用来它来做线性模型的多元分析，可以把一个比较多、有大量 **feature** 的数据集提取成只有几个 **main feature** 的思想，**但是有一点，提取出的 feature 不一定能还原成原来的标签，**是一个降维操作。
CCA(canonical correlation analysis) 就是典范对应分析，做的任务就是关联度分析。

2. 对点的权重进行改变，使用 SVM 进行带权分类
见文件夹中 svm-带权分类.py

```
def plot_decision_function(classifier, sample_weight, axis, title):
    xx, yy = np.meshgrid(np.linspace(-4, 5, 500), np.linspace(-4, 5, 500))
    Z = classifier.decision_function(np.c_[xx.ravel(), yy.ravel()])
    Z = Z.reshape(xx.shape)

    axis.contourf(xx, yy, Z, alpha=0.75, cmap=plt.cm.bone)
    axis.scatter(X[:, 0], X[:, 1], c=y, s=100 * sample_weight, alpha=0.9,
                 cmap=plt.cm.bone, edgecolors='black')

    axis.axis('off')
    axis.set_title(title)


np.random.seed(0)
X = np.r_[np.random.randn(10, 2) + [1, 1], np.random.randn(10, 2)]
y = [1] * 10 + [-1] * 10
sample_weight_last_ten = abs(np.random.randn(len(X)))
sample_weight_constant = np.ones(len(X))
sample_weight_last_ten[15:] *= 5
sample_weight_last_ten[9] *= 15
```

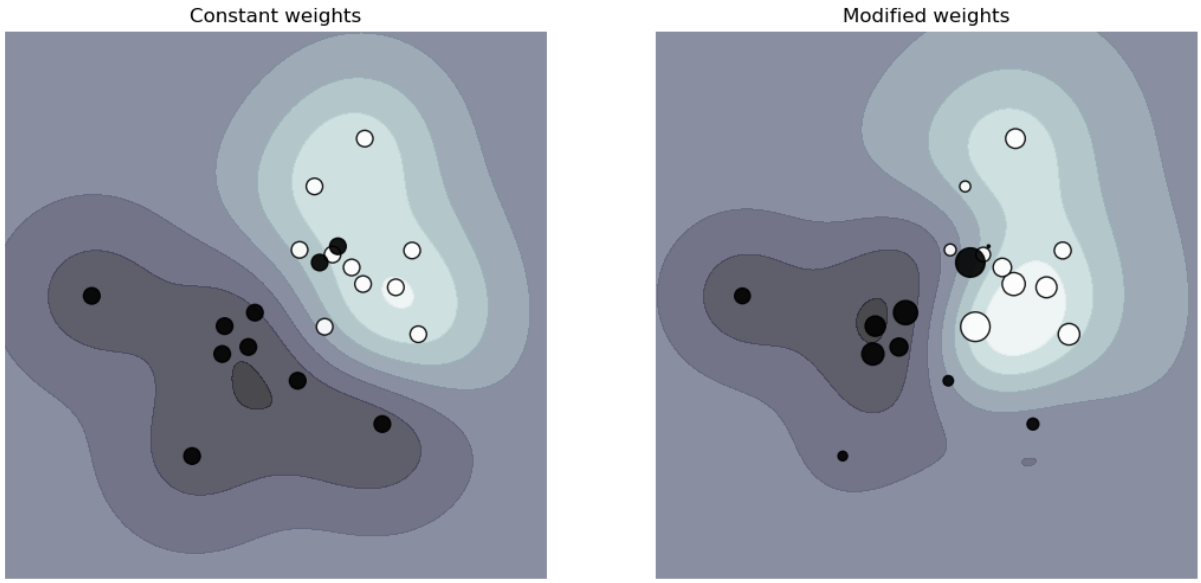
Combo
Max 5511

Linter

Severity	Provider	Description	Line

Atom Runner: M4.py

Atom Runner: svm-带权分类.py



备注：

在这个 demo 中，对点的权重进行不等设定，也就是对参数 C 进行更改，其中点越大，代表权重越大，在画分类边界时可以很明显看到扭曲的现象，**说明这种权重变化对分类结果是很敏感的。**

作业代码：

使用多种核函数对 test.txt 文件中的点进行分类

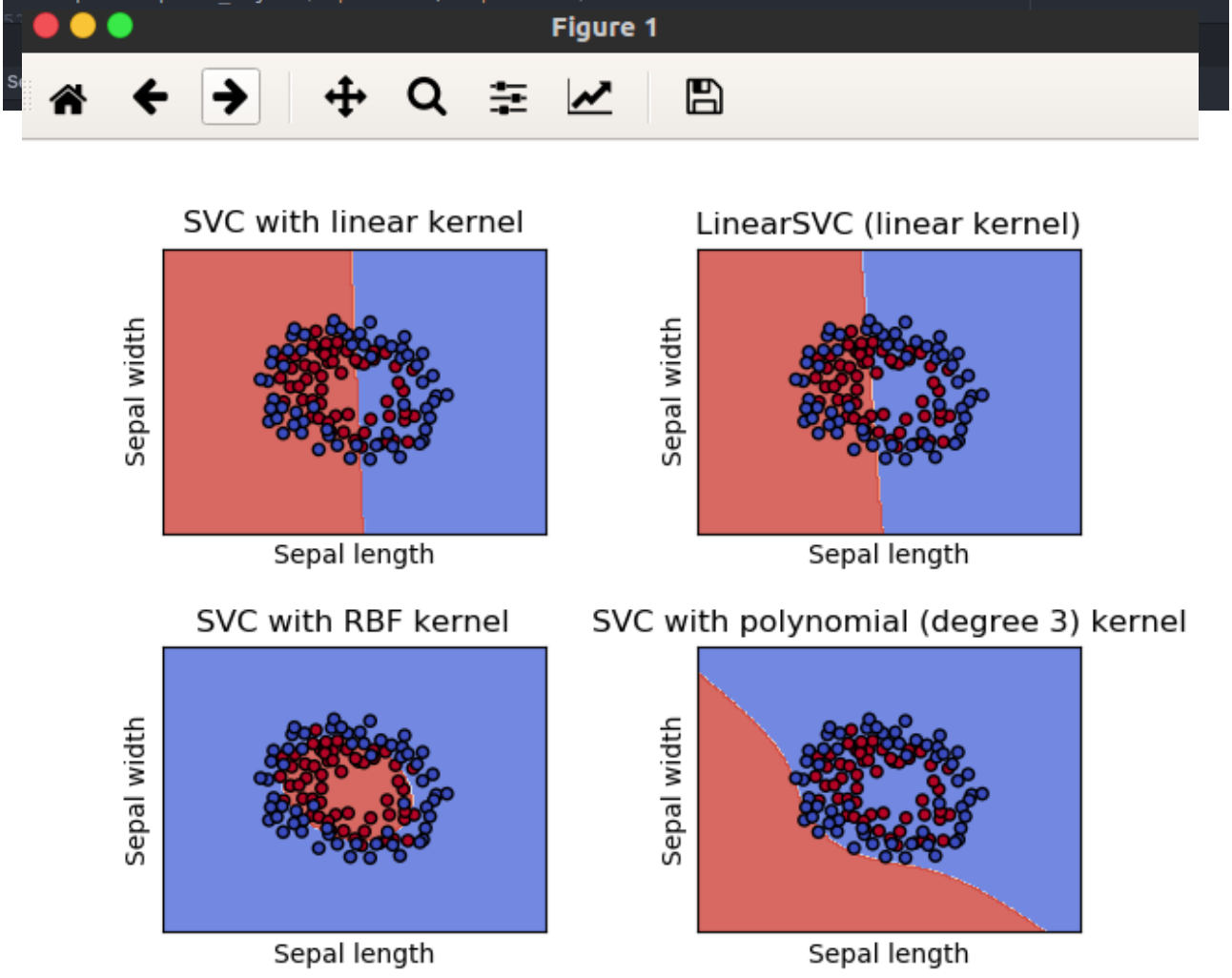
见文件夹中 M4_1.py

```
data = np.loadtxt("/home/ysera/桌面/新建文件夹/ml_homework_4/ml_homework_4/test.txt", delimiter=",")
X = np.array(data[:,:2])
y=np.array(data[:,2])

C = 1.0
models = (svm.SVC(kernel='linear', C=C),
          svm.LinearSVC(C=C),
          svm.SVC(kernel='rbf', gamma=0.7, C=C),
          svm.SVC(kernel='poly', degree=3, C=C))
models = (clf.fit(X, y) for clf in models)


titles = ('SVC with linear kernel',
          'LinearSVC (linear kernel)',
          'SVC with RBF kernel',
          'SVC with polynomial (degree 3) kernel')

fig, sub = plt.subplots(2, 2)
plt.subplots_adjust(wspace=0.4, hspace=0.4)
```



为了对比，在可视化的时候四个子图拼在一起比较

```
titles = ('SVC with linear kernel',
          'LinearSVC (linear kernel)',
          'SVC with RBF kernel',
          'SVC with polynomial (degree 3) kernel')
```

其中 我使用了：

带线性核的 SVC

线性 SVC

带 RBF 核的 SVC

带多项式核的 SVC

```
C = 1.0
models = (svm.SVC(kernel='linear', C=C),
          svm.LinearSVC(C=C),
          svm.SVC(kernel='rbf', gamma=0.7, C=C),
          svm.SVC(kernel='poly', degree=3, C=C))
models = (clf.fit(X, y) for clf in models)
```

参数调整如上，可以看到 RBF 确实如在我一开始比较中的适应多种场景，最常用的方法，处理这种线性不可分的问题，而多项式我们设置为 3 次多项式（高阶的效果越差），处理这种问题效果不是很好。

另外，由于 sigmoid 比较重要，所以单另拉出来实现一遍并画图：

见文件夹中 M4_2.py

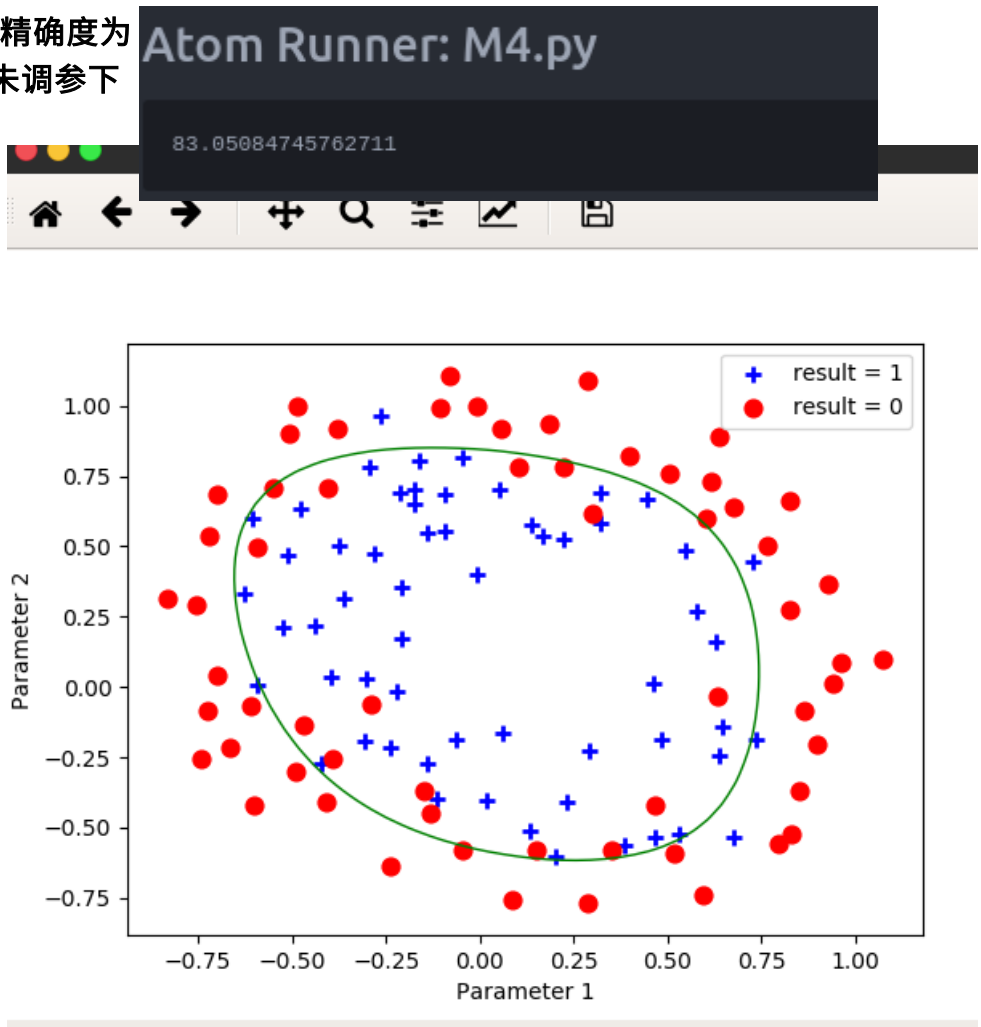
```
from numpy import *
import matplotlib.pyplot as plt
import numpy as np
from scipy.optimize import minimize

def loadDataSet():
    data = loadtxt("/home/ysera/桌面/新建文件夹/ml_homework_4/ml_homework_4/test.txt", delimit
    y = np.c_[data[:, 2]]
    X = data[:, 0:2]
    return data,X,y

def map_feature(x1, x2):
    x1.shape =(x1.size,1)
    x2.shape =(x2.size,1)
    degree =6
    mapped_fea = ones(shape=(x1[:,0].size,1))
    for i in range(1, degree +1):
        for j in range(i +1):
            r =(x1 **(i - j))*(x2 ** j)
            mapped_fea = append(mapped_fea, r, axis=1)
```

这里我们手写 sigmoid 函数和 loss function，对它的精确度 accuracy 进行计算

可以看到精确度为83.1%，未调参下



分类效果还是可以的

SVM 相关小结

SVM 分类器适合的数据集：

SVM 在小样本训练集上能够得到比其它算法好很多的结果。支持向量机之所以成为目前最常用，效果最好的分类器之一，在于其优秀的泛化能力，这是是因为其本身的优化目标是结构化风险最小，而不是经验风险最小，因此，通过 margin 的概念，得到对数据分布的结构化描述，因此减低了对数据规模和数据分布的要求。而大规模数据上，并没有实验和理论证明表明 svm 会差于其它分类器，也许只是相对其它分类器而言，领先的幅度没有那么高而已。

feature 数据要归一化，svm 对数据范围的变化比较敏感

- 优先考虑 RBF 作为核函数，因为 RBF 核的一种特例就是线性核
- 使用 Grid-search 来搜索超参数，同时可以使用交叉验证来判断当前使用的超参数是否为最优，比如对于 RBF 核的 SVM 来说就有 C 和 γ 两种参数要获取，那么可以尝试 C 取值 2e-5, 2e-3,...2e15，γ 取值 2e-15, 2e-13，...2e3。两者排列组合一对一对的试，用交叉验证来看那一对的准确度高，然后根据自己的精度需求，不断的细化搜索区间，找到更好的一对参数。
- 用 3 中找到的最优超参数来训练模型。

凸优化：

凸优化问题就是去寻找凸函数的极值。一般在没有任何约束条件下，求函数极值一般是求偏导，然后令偏导为 0，求得参数的值。这是最简单的情况。但是 SVM 优化目标函数是有不等式约束条件的。隐马尔可夫模型遇到过有等式约束的条件下，求函数的极值问题，使用的是拉格朗日乘子法，等式约束条件下的优化问题已经知道方法了，但是 SVM 中的优化函数是带不等式约束条件，使用对偶问题用 KKT 求解。

备注：

```
svc模型中参数的讲解:
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape=None, degree=3, gamma='auto', kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)

SVC参数解释
 (1) C: 目标函数的惩罚系数C，用来平衡分类间隔margin和错分样本的，default C = 1.0;
 (2) kernel: 参数选择有RBF, Linear, Poly, Sigmoid, 默认的是"RBF";
 (3) degree: if you choose 'Poly' in param 2, this is effective, degree决定了多项式的最高次
幂;
 (4) gamma: 核函数的系数('Poly', 'RBF' and 'Sigmoid'), 默认是gamma = 1 / n_features;
 (5) coef0: 核函数中的独立项，'RBF' and 'Poly'有效;
 (6) probablity: 可能性估计是否使用(true or false);
 (7) shrinking: 是否进行启发式;
 (8) tol (default = 1e - 3) : svm结束标准的精度;
 (9) cache_size: 制定训练所需要的内存（以MB为单位）;
 (10) class_weight: 每个类所占据的权重，不同的类设置不同的惩罚参数C, 缺省的话自适应;
 (11) verbose: 跟多线程有关，不大明白啥意思具体;
 (12) max_iter: 最大迭代次数，default = 1,  if max_iter = -1, no limited;
 (13) decision_function_shape :  'ovo' 一对一, 'ovr' 多对多  or None 无, default=None
 (14) random_state : 用于概率估计的数据重排时的伪随机数生成器的种子。
 ps: 7,8,9一般不考虑。
```

SVM 优点及局限性

一、 SVM 优点

1、解决小样本下机器学习问题。

2、解决非线性问题。

3、无局部极小值问题。（相对于神经网络等算法）

4、可以很好的处理高维数据集。

5、泛化能力比较强。

二、SVM 缺点

1、对于核函数的高维映射解释力不强，尤其是径向基函数。

2、对缺失数据敏感。

三、SVM 应用领域文本分类、图像识别、主要二分类领域

参考

[1] svm 可视化工具
https://github.com/karpathy/svmjs

[2] 格式提取
https://blog.csdn.net/csj664103736/article/details/72828584

[3] SVM
https://www.zhihu.com/question/21094489/answer/86273196

[4] SVM 实用经验
A Practical Guide to Support Vector Classification
Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin
Department of Computer Science
National Taiwan University, Taipei 106, Taiwan
http://www.csie.ntu.edu.tw/~cjlin
Initial version: 2003 Last updated: May 19, 2016
https://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf