模式识别课后作业六

2020E8013282019 李一帆

January 8, 2021

1 简述题

1.1 问题 1

问题描述：有 N 个样本 $x_1, ..., x_N$，每个样本维数 D，希望将样本维数降到 K，请给出 PCA 算法计算过程。

解答：首先计算数据的均值 $\bar{x}$：

$$\bar{x} = \frac{1}{N}\sum_{n=1}^{N} x_n$$

然后计算样本的协方差矩阵 S。

$$S = \frac{1}{N}\sum_{n=1}^{N} (x_n - \bar{x})(x_n - \bar{x})^T$$

对协方差矩阵 S（$D \times D$）进行特征值分解，取前 K 个特征值对应的特征向量，分别记作 $\boldsymbol{u}_1, \boldsymbol{u}_2, ..., \boldsymbol{u}_K$ （其中 $\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_{K-1} \geq \lambda_K$），将映射矩阵记作 $U = [u_1, u_2, ..., u_K]$ 大小为 $D \times K$。

最后每个样本 x_n 的映射可以表示为 $z_n = U^T x_n$。

1.2 问题 2

问题描述：根据自己的理解简述结构风险最小化与 VC 维。

解答：结构风险最小化指的是对于未知数据有一个较低的错误率，即具有较好的泛化性能。VC 维指在一个 n 维空间内，模型一定能够分类的数据点的数量，例如 2 维平面的 VC 维是 3，主要用来衡量模型的复杂程度。

1.3 问题 3

问题描述：推导出 Hard-Margin SVM 的优化目标。

解答：

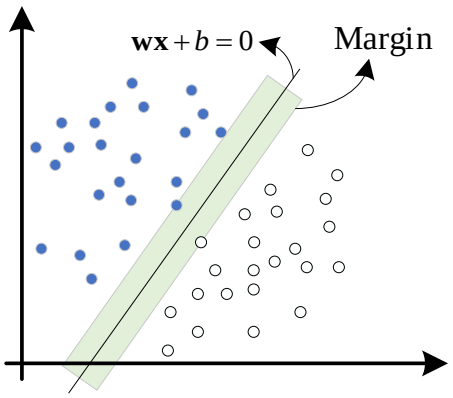

Figure 1: 硬间隔 SVM.

记数据点到分界面之间的距离为 $d(x) = \frac{|wx+b|}{\sqrt{\sum_{i=1}^{d}|w_i|^2}}$。则 margin 可以定义为

$$margin = \arg\min_{x_i \in D} d(x) = \arg\min_{x_i \in D} \frac{|b + wx_i|}{\sqrt{\sum_{i=1}^{d}|w_i|^2}}$$

SVM 的目标是使得 margin 最大的情况下，获得参数 w 和 b，因此目标函数可以写作如下形式。

$$\arg\max_{w,b} margin = \arg\max_{w,b} \arg\min_{x_i \in D} \frac{|b + wx_i|}{\sqrt{\sum_{i=1}^{d}|w_i|^2}}$$

约束条件为所有样本点都分对，即 $(wx_i + b)y_i \geq 0, \forall x_i \in D$。

令 $|wx_i + b| \geq 1$，则 $(wx_i + b)y_i \geq 1, \forall x_i \in D$。margin 可以表示为如下形式。

$$\arg\min_{x_i \in D} \frac{|wx_i + b|}{\sqrt{\sum_{i=1}^{d} w_i^2}} \geq \arg\min_{x_i \in D} \frac{1}{\sqrt{\sum_{i=1}^{d} w_i^2}} = \frac{1}{\sqrt{\sum_{i=1}^{d} w_i^2}}$$

因此原目标函数可以转换为如下形式。

$$\arg\max_{w,b} margin = \arg\max_{w,b} \arg\min_{x_i \in D} \frac{|b + wx_i|}{\sqrt{\sum_{i=1}^{d}|w_i|^2}} = \arg\min_{w,b} \sum_{i=1}^{d} w_i^2$$

约束条件为 $(wx_i + b)y_i \geq 0, \forall x_i \in D$。

1.4 问题 4

问题描述：请解释 Hinge Loss 在 SVM 中的意义。

解答：Hinge Loss 用来描述对分类错误的惩罚。如果分类正确则不会有惩罚；如果分类错误，则根据分类错误时的函数距离进行惩罚，距离越远则惩罚力度越大。

1.5 问题 5

问题描述：简述核方法的基本原理。

解答：核方法主要是将低维线性不可分数据映射到高维，使其变得线性可分。实际在应用的时候是将数据内积利用核函数进行映射，使得映射后的数据点线性可分。

2 计算机编程

问题描述：从 MNIST 数据集中任意选择两类，对其进行 SVM 分类，可调用现有的 SVM 工具如 LIBSVM，展示超参数 C 以及核函数参数 gamma 的选择过程。

解答：这里选择 RBF 核对 MNIST 数据集中的 0，1 手写数字进行分类。训练集为 MNIST 训练集的 0，1 手写数字，测试集为 MNIST 测试集的 0，1 手写数字（一共选择 2115 个数字）。Figure 2 为测试的结果。

Gamma / C	0.00001	0.000001	0.0000001	0.00000001
1	86.52482270	99.85815603	53.6643026	53.6643026
2	87.61229314	99.85815603	53.6643026	53.6643026
3	87.61229314	99.85815603	53.6643026	53.6643026
4	87.61229314	99.85815603	53.6643026	53.6643026
5	87.61229314	99.85815603	53.6643026	53.6643026
6	87.61229314	99.85815603	53.6643026	53.6643026
7	87.61229314	99.85815603	53.6643026	53.6643026
8	87.61229314	99.85815603	53.6643026	53.6643026
9	87.61229314	99.85815603	53.6643026	53.6643026

Figure 2: 测试结果.

由图中可以看出在用 RBF 核作为核函数时，Gamma 的选择对于结果影响较为明显。在 Gamma 为 0.000001 时，手写数字识别的准确率最高。

A 附录——代码 (Python)

```
from svmutil import *
from svm import *
import os
import numpy as np
import gzip


def load_data(data_file):
    files = ['train-labels-idx1-ubyte.gz', 'train-images-idx3-ubyte.gz',
             't10k-labels-idx1-ubyte.gz', 't10k-images-idx3-ubyte.gz']
    paths = []
    for file in files:
        paths.append(os.path.join(data_file, file))

    with gzip.open(paths[0], 'rb') as lbpath:
        y_train = np.frombuffer(
            lbpath.read(), np.uint8, offset=8)

    with gzip.open(paths[1], 'rb') as imgpath:
        x_train = np.frombuffer(
            imgpath.read(), np.uint8, offset=16).reshape(len(y_train), 28, 28)

    with gzip.open(paths[2], 'rb') as lbpath:
        y_test = np.frombuffer(
            lbpath.read(), np.uint8, offset=8)

    with gzip.open(paths[3], 'rb') as imgpath:
        x_test = np.frombuffer(
            imgpath.read(), np.uint8, offset=16).reshape(len(y_test), 28, 28)
    return (x_train, y_train), (x_test, y_test)
```

```
31
32
33      def make_data_input(default=[0, 1]):
34          x_train = []
35          y_train = []
36          x_test = []
37          y_test = []
38          (train_images, train_labels), (test_images, test_labels) =
               load_data('MNIST/')
39          for k, image in enumerate(train_images):
40              if int(train_labels[k]) in default:
41                  image_flatten = image.reshape(28 * 28, 1)
42                  x_img = {i: int(image_flatten[i]) for i in range(28
                        * 28)}
43                  x_train.append(x_img)
44                  y_train.append(int(train_labels[k]))
45
46          for k, image in enumerate(test_images):
47              if int(test_labels[k]) in default:
48                  image_flatten = image.reshape(28 * 28, 1)
49                  x_img = {i: int(image_flatten[i]) for i in range(28
                        * 28)}
50                  x_test.append(x_img)
51                  y_test.append(int(test_labels[k]))
52          return (x_train, y_train, x_test, y_test)
53
54
55      def make_data_file(default=[0, 1]):
56          (train_images, train_labels), (test_images, test_labels) =
               load_data('MNIST/')
57          with open('train_mnist.txt', 'w') as file:
58              for k, image in enumerate(train_images):
59                  if int(train_labels[k]) in default:
60                      image_flatten = image.reshape(28 * 28, 1)
61                      x_img = str(train_labels[k]) + ' '
62                      for i in range(1, 28 * 28 + 1):
```

```
63                          x_img += str(i) + ':' + str(image_flatten[i
                                - 1][0]) + ' '
64                      x_img.strip(' ')
65                      x_img += '\n'
66                      file.write(x_img)
67
68          with open('test_mnist.txt', 'w') as file:
69              for k, image in enumerate(test_images):
70                  if int(train_labels[k]) in default:
71                      image_flatten = image.reshape(28 * 28, 1)
72                      x_img = str(test_labels[k]) + ' '
73                      for i in range(1, 28 * 28 + 1):
74                          x_img += str(i) + ':' + str(image_flatten[i
                                - 1][0]) + ' '
75                      x_img.strip(' ')
76                      x_img += '\n'
77                      file.write(x_img)
78
79
80      if __name__ == '__main__':
81          #######Make dataset#######
82          # make_data_file()
83          x_train, y_train, x_test, y_test = make_data_input()
84
85          cs = range(1, 11)
86          gammas = [0.00001, 0.000001, 0.0000001, 0.00000001]
87          accs = np.zeros([len(cs), len(gammas)], dtype='float')
88          prob = svm_problem(y_train, x_train)
89          for i, c in enumerate(cs):
90              for j, gamma in enumerate(gammas):
91                  parameter = '-t 2 -c %d -g %f' % (c, gamma)
92                  param = svm_parameter(parameter)
93                  m = svm_train(prob, param)
94                  p_label, p_acc, p_val = svm_predict(y_test, x_test,
                        m)
95                  accs[i, j] = p_acc[0]
```

```
96          print(accs)
```