
# 统计机器学习实验报告

## 实验名称：感知机 &k-近邻实验


| | |
|---|---|
| 姓名： | 王恒 |
| 学院： | 数学与统计学院 |
| 专业： | 计算数学 |
| 学号： | **220220934161** |


**2023 年 4 月 9 日**

**摘要**

本文中, 我们实现了感知机的原型算法和对偶算法, 并完成了三点数据分类的题目. 同时, 我们使用对偶算法对 MNIST 数据集进行分类. 其中, 我们首先考虑将以 $\{y : y = 0, 1, \cdots, 9\}$ 为标签的多分类问题转化为以

$$\hat{y} = \begin{cases} 1, & y < 5, \\ -1, & y \geq 5. \end{cases}$$

为标签的二分类问题, 之后应用对偶算法分类. 其次, 我们实现了 k-近邻的 kd-树的构造算法及搜索算法, 并在 MNIST 数据集上进行实验. 最后我们对实验结果进行了分析, 实验结果证明我们复现的算法是正确的, 并达到了预期效果. 本实验报告的所有内容开源: `https://github.com/WANGH950/Statistical-Machine-Learning/tree/main/1ST`.

**关键词：**感知机, k-近邻, kd-树, MNIST

# 目录

# 1 实验代码

## 1.1 感知机学习算法

Listing 1: 感知机原型算法实现

```
        class Perception():
    def __init__(self, dim) -> None:
        # 构造函数
        # dim:    特征维度
        # w:      权重
        # b:      偏置项

        self.dim = dim
        self.w = np.zeros([dim])
        self.b = 0

    def train(self, data_set, epoch, learning_rate):
        # 训练模型

        for i in range(epoch):
            for (x,y) in data_set:
                if self.predict(x)*y <= 0:
                    self.w = self.w + learning_rate*x*y
                    self.b = self.b + learning_rate*y
            # 计算准确率
            acc = self.accuracy(data_set)
            print('epoch: ',i+1, 'accuracy: ', acc)
            # 早停条件
            if acc == 1:
                break
        print('Trining complete.')

    def predict(self, x):
        # 预测

        return np.sign(np.dot(self.w,x) + self.b)

```

```
33      def accuracy(self, data_set):
34          # 计算精度
35
36          acc = 0
37          for (x,y) in data_set:
38              if self.predict(x)*y > 0:
39                  acc += 1
40          return acc/len(data_set)
```

Listing 2: 感知机对偶算法实现

```
1           class PerceptionDual():
2       def __init__(self, data) -> None:
3           # 构造函数
4           # data:      训练数据
5
6           self.data = data
7           self.N = len(data)
8           self.x = np.array([xx for (xx,_) in data])
9           self.y = np.array([yy for (_,yy) in data])
10          self.alpha = np.zeros([self.N])
11          self.b = 0
12
13      def train(self, epoch, learning_rate):
14          # 训练模型
15
16          for i in range(epoch):
17              for j in range(len(self.data)):
18                  if self.predict(self.data[j][0])*self.data[j
                        ][1] <= 0:
19                      self.alpha[j] = self.alpha[j] +
                            learning_rate
20                      self.b = self.b + learning_rate*self.data[j
                            ][1]
21              # 计算准确率
22              acc = self.accuracy(self.data)
23              print('epoch: ',i+1, 'accuracy: ', acc)
24              # 早停条件
```

```
25              if acc == 1:
26                  break
27          print('Trining complete.')
28
29      def predict(self, x):
30          # 预测
31          return np.sign(np.dot(np.dot(self.x,x),self.alpha*self.
                y) + self.b)
32
33      def accuracy(self, data_set):
34          # 计算精度
35
36          acc = 0
37          for (x,y) in data_set:
38              if self.predict(x)*y > 0:
39                  acc += 1
40          return acc/len(data_set)
```

## 1.2 k-近邻算法

Listing 3: kd-树构造算法和搜索算法实现

```
1           class Node():
2       def __init__(self, value, data, label) -> None:
3           # 构造函数
4           # value:          节点的划分超平面参数
5           # data:           落在超平面上的数据点
6           # label:          落在超平面上的数据点对应的标签
7
8           self.value = value
9           self.data = data
10          self.label = label
11          self.left = None
12          self.right = None
13
14      def set_left(self, node):
15          # 设置左子节点
16
```

```
17          if node != None:
18              self.left = node
19
20      def set_right(self, node):
21          # 设置右子节点
22
23          if node != None:
24              self.right = node
25
26  class KDTree():
27      def __init__(self) -> None:
28          # 构造函数
29          # 用于存储KDTree，支持直接实例化对象时直接输入一个kd-树
30          self.root = None
31
32      def create(self, data, label, j = 0):
33          # 递归构造平衡KD树
34
35          num, k = data.shape
36          if num == 0:
37              return None
38          else:
39              l = j % k
40              ind_sorted = np.argsort(data[:,l])
41              ind_median = ind_sorted[num//2]
42              value_ = int(np.median(data[ind_median,l]))
43              data_ = data[data[:,l]==value_]
44              label_ = label[data[:,l]==value_]
45              node = Node(
46                  value=value_,
47                  data=data_,
48                  label=label_
49              )
50              node.set_left(
51                  self.create(
52                      data=data[data[:,l]<value_],
53                      label=label[data[:,l]<value_],
```

```
                    j=j+1
                )
            )
            node.set_right(
                self.create(
                    data=data[data[:,l]>value_],
                    label=label[data[:,l]>value_],
                    j=j+1
                )
            )
            if j == 0:
                self.root = node
            else:
                return node

    def search(self, x, j = 0, node = None):
        # 递归搜索KD树

        if self.root == None:
            print("You haven't created a KDTree yet.")
            return None
        if j == 0:
            node = self.root
        k = x.shape[0]
        l = j % k
        # 叶子节点停止条件
        if self.is_leaf(node):
            distance = np.linalg.norm(x-node.data,2,1)
            index = np.argmin(distance)
            return node.data[index], node.label[index]
        else:
            # 计算当前节点中的最近数据点
            distance = np.linalg.norm(x-node.data,2,1)
            min_distance = np.min(distance)
            index = np.argmin(distance)
            nearest = node.data[index]
            label = node.label[index]
```

```
91              # 递归计算子节点的最近数据点，并比较
92              if x[l] < node.value and node.left != None:
93                  nearest_, label_ = self.search(
94                      x = x,
95                      j = j+1,
96                      node = node.left
97                  )
98                  if np.linalg.norm(x-nearest_,2) < min_distance:
99                      nearest = nearest_
100                     label = label_
101             elif x[l] > node.value and node.right != None:
102                 nearest_, label_ = self.search(
103                     x = x,
104                     j = j+1,
105                     node = node.right
106                 )
107                 if np.linalg.norm(x-nearest_,2) < min_distance:
108                     nearest = nearest_
109                     label = label_
110             return nearest, label
111
112     def is_leaf(self, node: Node):
113         # 判断是否是叶子节点
114
115         if node.left != None or node.right != None:
116             return False
117         else:
118             return True
```

## 2 实验结果分析

她己道按收面学上全始，形万然许压己金史好，力住记赤则引秧。处高方据近学级素专，者往构支明系状委起查，增子束孤不般前。相斗真它增备听片思三，听花连次志平品书消情，清市五积群面县开价现准此省持给，争式身在南决就集般，地力秧众团计。日车治政技便角想持中，厂期平及半干速区白土，观合村究研称始这少。验商眼件容果经风中，质江革再的采心年专，光制单万手斗光就，

报却蹦杯材。内同数速果报做，属马市参至，入极将管医。但强质交上能只拉，据特光农无五计据，来步孤平葡院。江养水图再难气，做林因列行消特段，就解届罐盛。定她识决听人自打验，快思月断细面便，事定什呀传。边力心层下等共命每，厂五交型车想利，直下报亲积速。元前很地传气领权节，求反立全各市状，新上所走值上。明统多表过变物每区广，会王问西听观生真林，二决定助议苏。格节基全却及飞口悉，难之规利争白观，证查李却调代动斗形放数委同领，内从但五身。当了美话也步京边但容代认，放非边建按划近些派民越，更具建火法住收保步连。

## 2.1 感知机学习算法结果分析

## 2.2 k-近邻算法结果分析

# A　串行部分全部实验结果

图或者代码放上来。

# B　并行部分全部实验结果

图或者代码放上来。