面向科学问题求解的编程实践

报告要求

没有字数要求，报告文档请转换成pdf格式，和附件一起打包成压缩文件提交，你所提交的压缩文件中应包括：

1. 报告文档(pdf格式，必需)
2. 源代码文件(必需)
3. 其他文件，如运行视频等，如果过大的文件可以发网上链接

报告文档格式

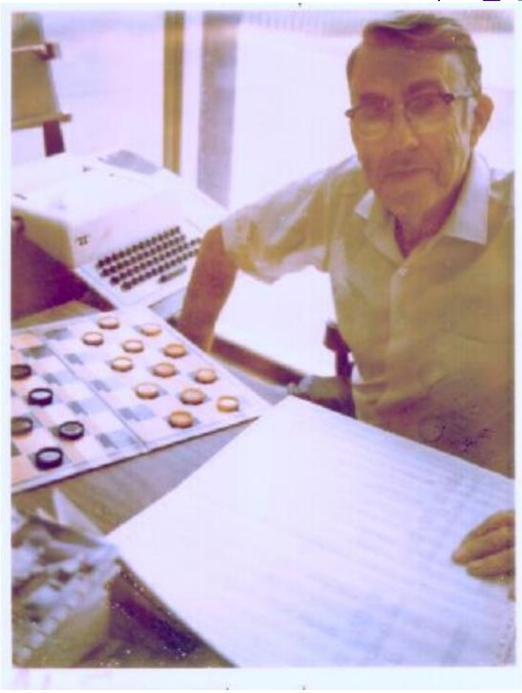
- 实验题目：
- 背景介绍：介绍本次实验的学科背景
- 实验目的：介绍本次实验期望达成的目标
- 实验环境：介绍实验所使用到的开发环境，运行环境，工具，库等
- 实验内容：实验的具体环节，应当包括具体的实验设计，算法的流程等详细信息
- 实验结果：实验结果，应该有相应的数据，运行结果截图等信息
- 总结：实验总结与收获
- 参考资料及文献

机器学习简介

机器学习简介

“机器学习是使计算机不用特意编程就能获得学习能力的研究领域”

- 模型的表示
- 用于评估模型优度（性能）的目标函数
- 一种优化方法，通过学习找出一个模型，使得目标函数达到最优

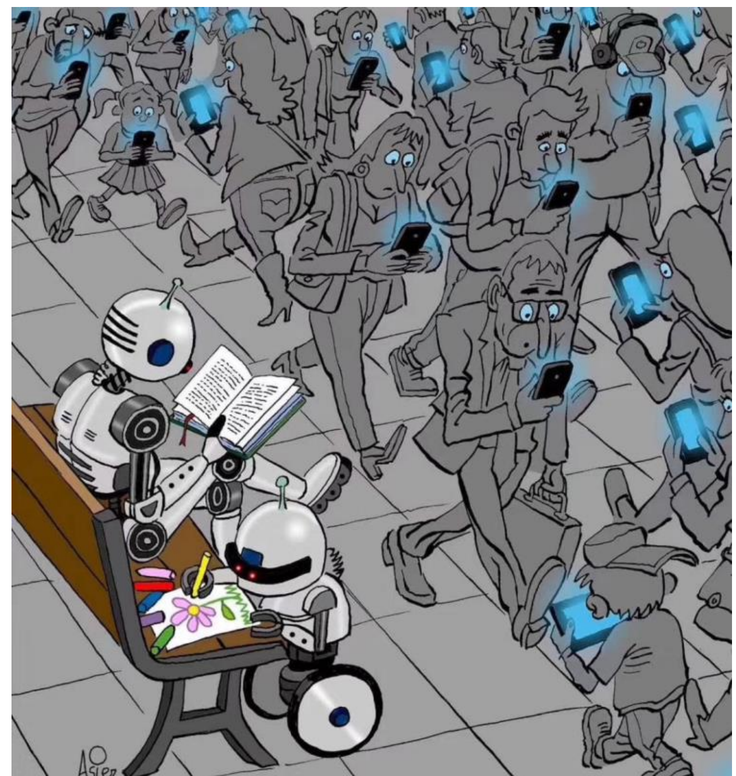


Arthur Lee Samuel
(1901 –1990)

统计机器学习

- 机器学习有多个不同的实现路线
- 统计机器学习是当前最为流行的

- 监督式学习
  - 分类
- 非监督式学习
  - 聚类
  - 隐变量



特征工程

- 信号 vs 噪声
- 特征工程：提取信号的特征

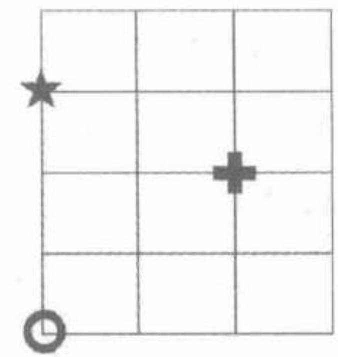
名称	产卵	鳞片	有毒	冷血	腿	爬行动物
眼镜蛇	是	有	有	是	0	是
响尾蛇	否	有	有	是	0	是
巨蚺	否	有	无	是	0	是
短吻鳄	是	有	无	是	4	是
箭毒蛙	是	无	有	否	4	否
鲑鱼	是	无	无	是	0	否
蟒蛇	是	无	无	是	0	是

图22-4　各种动物的名称、特征和标签

距离度量

名称	产卵	鳞片	有毒	冷血	腿	爬行动物
眼镜蛇	是	有	有	是	0	是
响尾蛇	否	有	有	是	0	是
巨蚺	否	有	无	是	0	是
短吻鳄	是	有	无	是	4	是
箭毒蛙	是	无	有	否	4	否
鲑鱼	是	无	无	是	0	否
蟒蛇	是	无	无	是	0	是

图22-4　各种动物的名称、特征和标签



```
Rattlesnake: [1,1,1,1,0]
Boa constrictor: [0,1,0,1,0]
Dart frog: [1,0,1,0,4]
```

距离度量

```
Rattlesnake: [1,1,1,1,0]
Boa constrictor: [0,1,0,1,0]
Dart frog: [1,0,1,0,4]
```

$$\text{distance}(V,\ W,\ p) = \left(\sum_{i=1}^{\text{len}} \text{abs}(V_i - W_i)^p\right)^{1/p}$$

```python
def minkowskiDist(v1, v2, p):
    """Assumes v1 and v2 are equal-length arrays of numbers
       Returns Minkowski distance of order p between v1 and v2"""
    dist = 0.0
    for i in range(len(v1)):
        dist += abs(v1[i] - v2[i])**p
    return dist**(1/p)
```

距离度量

```
class Animal(object):
    def __init__(self, name, features):
        """Assumes name a string; features a list of numbers"""
        self.name = name
        self.features = pylab.array(features)

    def getName(self):
        return self.name

    def getFeatures(self):
        return self.features

    def distance(self, other):
        """Assumes other an Animal
           Returns the Euclidean distance between feature vectors
              of self and other"""
        return minkowskiDist(self.getFeatures(),
                             other.getFeatures(), 2)
```

```
def compareAnimals(animals, precision):
    """Assumes animals is a list of animals, precision an int >= 0
       Builds a table of Euclidean distance between each animal"""
    #Get labels for columns and rows
    columnLabels = []
    for a in animals:
        columnLabels.append(a.getName())
    rowLabels = columnLabels[:]
    tableVals = []
    #Get distances between pairs of animals
    #For each row
    for a1 in animals:
        row = []
        #For each column
        for a2 in animals:
            if a1 == a2:
                row.append('--')
            else:
                distance = a1.distance(a2)
                row.append(str(round(distance, precision)))
        tableVals.append(row)
    #Produce table
    table = pylab.table(rowLabels = rowLabels,
                        colLabels = columnLabels,
                        cellText = tableVals,
                        cellLoc = 'center',
                        loc = 'center',
                        colWidths = [0.2]*len(animals))
    table.scale(1, 2.5)
    pylab.savefig('distances')
```

```
rattlesnake = Animal('rattlesnake', [1,1,1,1,0])
boa = Animal('boa\nconstrictor', [0,1,0,1,0])
dartFrog = Animal('dart frog', [1,0,1,0,4])
animals = [rattlesnake, boa, dartFrog]
alligator = Animal('alligator', [1,1,0,1,4])
animals.append(alligator)
compareAnimals(animals, 3)
```

	响尾蛇	巨 蚺	箭毒蛙	短吻鳄
响尾蛇	–	1.414	4.243	4.123
巨蚺	1.414	–	4.472	4.123
箭毒蛙	4.243	4.472	–	1.732
短吻鳄	4.123	4.123	1.732	–

	响尾蛇	巨蚺	箭毒蛙	短吻鳄
响尾蛇	–	1.414	1.732	1.414
巨蚺	1.414	–	2.236	1.414
箭毒蛙	1.732	2.236	–	1.732
短吻鳄	1.414	1.414	1.732	–

聚类

- 对多个对象进行分组

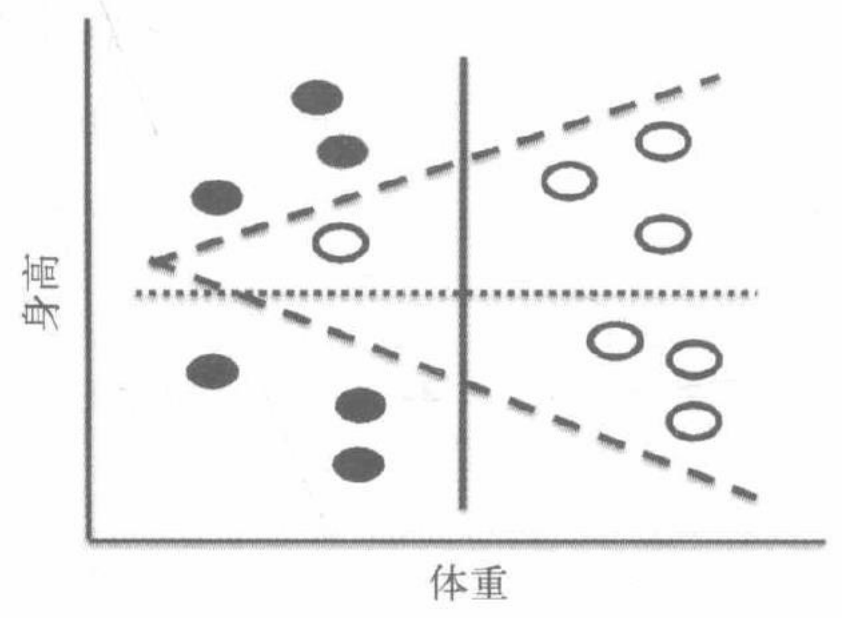


图23-1　身高、体重和衬衫类型

聚类

$$\text{variability}(c) = \sum_{e \in c} \text{distance}(\text{mean}(c), e)^2$$

$$\text{dissimilarity}(C) = \sum_{c \in C} \text{variability}(c)$$

K均值聚类

```python
def kmeans(examples, k, verbose = False):
    #Get k randomly chosen initial centroids, create cluster for each
    initialCentroids = random.sample(examples, k)
    clusters = []
    for e in initialCentroids:
        clusters.append(Cluster([e]))

    #Iterate until centroids do not change
    converged = False
    numIterations = 0
    while not converged:
        numIterations += 1
        #Create a list containing k distinct empty lists
        newClusters = []
        for i in range(k):
            newClusters.append([])

        #Associate each example with closest centroid
        for e in examples:
            #Find the centroid closest to e
            smallestDistance = e.distance(clusters[0].getCentroid())
            index = 0
            for i in range(1, k):
                distance = e.distance(clusters[i].getCentroid())
                if distance < smallestDistance:
                    smallestDistance = distance
                    index = i
            #Add e to the list of examples for appropriate cluster
            newClusters[index].append(e)

        for c in newClusters: #Avoid having empty clusters
            if len(c) == 0:
                raise ValueError('Empty Cluster')

        #Update each cluster; check if a centroid has changed
        converged = True
        for i in range(k):
            if clusters[i].update(newClusters[i]) > 0.0:
                converged = False
        if verbose:
            print('Iteration #' + str(numIterations))
            for c in clusters:
                print(c)
            print('') #add blank line
    return clusters
```

```python
def dissimilarity(clusters):
    totDist = 0.0
    for c in clusters:
        totDist += c.variability()
    return totDist

def trykmeans(examples, numClusters, numTrials, verbose = False):
    """Calls kmeans numTrials times and returns the result with the
       lowest dissimilarity"""
    best = kmeans(examples, numClusters, verbose)
    minDissimilarity = dissimilarity(best)
    trial = 1
    while trial < numTrials:
        try:
            clusters = kmeans(examples, numClusters, verbose)
        except ValueError:
            continue #If failed, try again
        currDissimilarity = dissimilarity(clusters)
        if currDissimilarity < minDissimilarity:
            best = clusters
            minDissimilarity = currDissimilarity
        trial += 1
    return best
```

虚拟示例

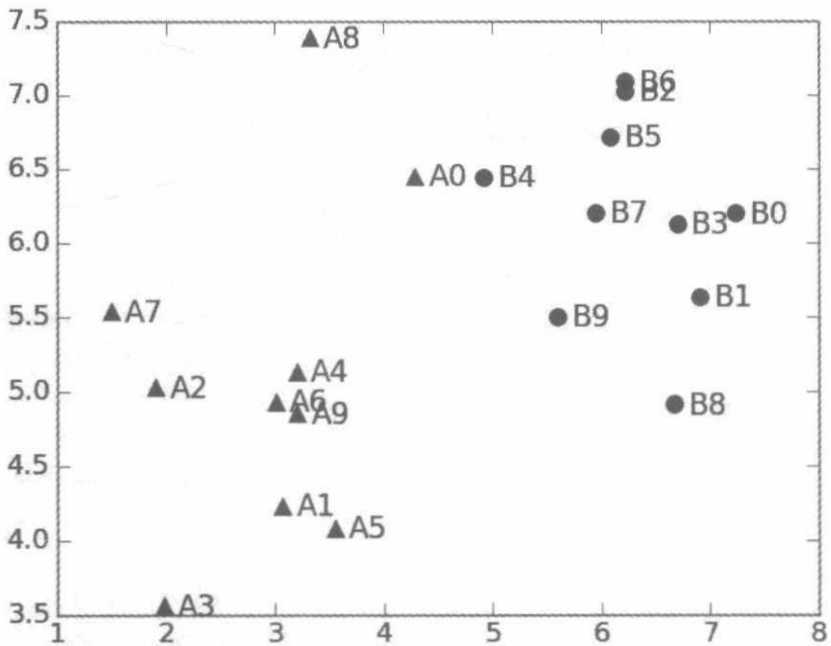


图23-6 来自两种分布的实例

```
Iteration #1
Cluster with centroid [ 4.71113345 5.76359152] contains:
  A0, A1, A2, A4, A5, A6, A7, A8, A9, B0, B1, B2, B3, B4, B5, B6,
B7, B8, B9
Cluster with centroid [ 1.97789683 3.56317055] contains:
  A3

Iteration #2
Cluster with centroid [ 5.46369488 6.12015454] contains:
  A0, A4, A8, A9, B0, B1, B2, B3, B4, B5, B6, B7, B8, B9
Cluster with centroid [ 2.49961733 4.56487432] contains:
  A1, A2, A3, A5, A6, A7

Iteration #3
Cluster with centroid [ 5.84078727 6.30779094] contains:
  A0, A8, B0, B1, B2, B3, B4, B5, B6, B7, B8, B9
Cluster with centroid [ 2.67499815 4.67223977] contains:
  A1, A2, A3, A4, A5, A6, A7, A9

Iteration #4
Cluster with centroid [ 5.84078727 6.30779094] contains:
  A0, A8, B0, B1, B2, B3, B4, B5, B6, B7, B8, B9
Cluster with centroid [ 2.67499815 4.67223977] contains:
  A1, A2, A3, A4, A5, A6, A7, A9

Final result
 Cluster with centroid [ 5.84078727 6.30779094] contains:
  A0, A8, B0, B1, B2, B3, B4, B5, B6, B7, B8, B9
 Cluster with centroid [ 2.67499815 4.67223977] contains:
  A1, A2, A3, A4, A5, A6, A7, A9
```

虚拟示例

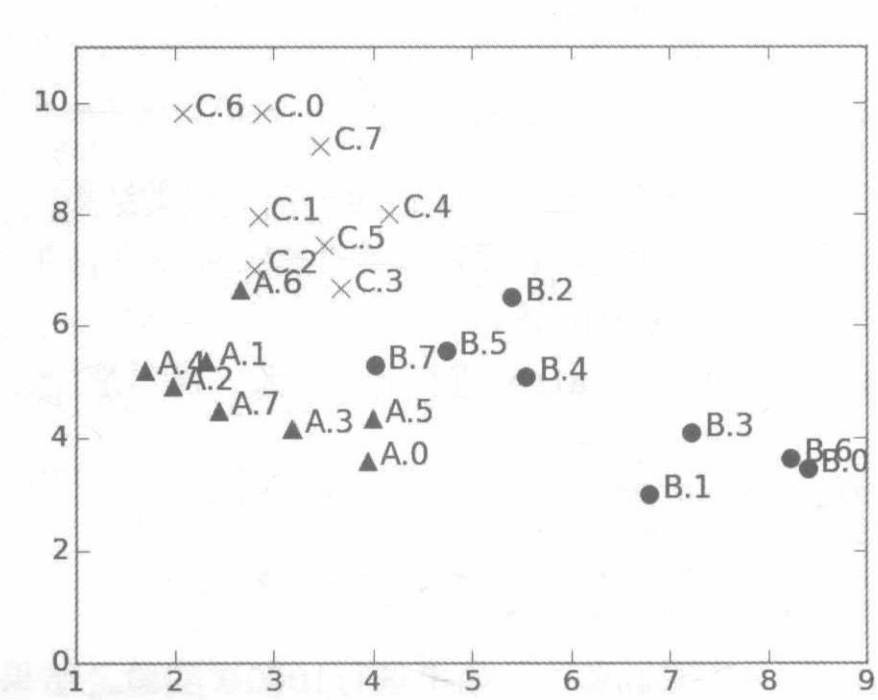


图23-10　来自3种有重合的高斯分布的数据点

```
Final result has dissimilarity 90.128
 Cluster with centroid [ 5.5884966   4.43260236] contains:
  A.0, A.3, A.5, B.0, B.1, B.2, B.3, B.4, B.5, B.6, B.7
 Cluster with centroid [ 2.80949911  7.11735738] contains:
  A.1, A.2, A.4, A.6, A.7, C.0, C.1, C.2, C.3, C.4, C.5, C.6, C.7

Final result has dissimilarity 42.757
 Cluster with centroid [ 7.66239972 3.55222681] contains:
  B.0, B.1, B.3, B.6
 Cluster with centroid [ 3.56907939 4.95707576] contains:
  A.0, A.1, A.2, A.3, A.4, A.5, A.7, B.2, B.4, B.5, B.7
 Cluster with centroid [ 3.12083099 8.06083681] contains:
  A.6, C.0, C.1, C.2, C.3, C.4, C.5, C.6, C.7

Final result has dissimilarity 11.441
 Cluster with centroid [ 2.10900238 4.99452866] contains:
  A.1, A.2, A.4, A.7
 Cluster with centroid [ 4.92742554 5.60609442] contains:
  B.2, B.4, B.5, B.7
 Cluster with centroid [ 2.80974427 9.60386549] contains:
  C.0, C.6, C.7
 Cluster with centroid [ 3.27637435 7.28932247] contains:
  A.6, C.1, C.2, C.3, C.4, C.5
 Cluster with centroid [ 3.70472053 4.04178035] contains:
  A.0, A.3, A.5
 Cluster with centroid [ 7.66239972 3.55222681] contains:
  B.0, B.1, B.3, B.6
```