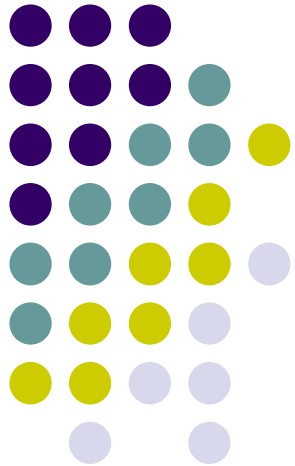
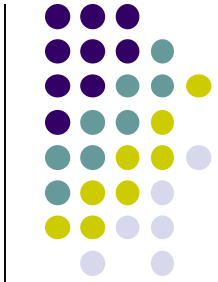
面向科学问题求解的编程实践



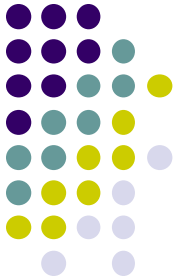
报告要求



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：

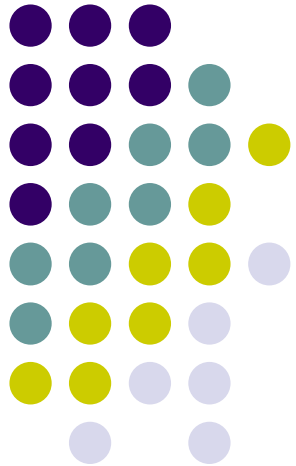
1. 报告文档(pdf格式，必需)
2. 源代码文件(必需)
3. 其他文件，如运行视频等，如果过大的文件可以发网上链接

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”

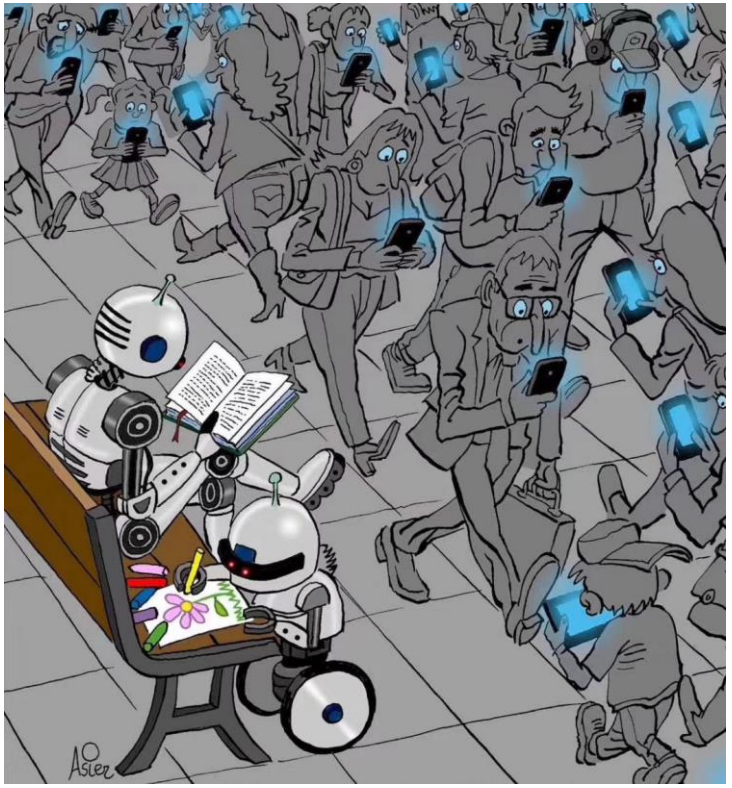
- 模型的表示
- 用于评估模型优度（性能）的目标函数
- 一种优化方法，通过学习找出一个模型，使得目标函数达到最优

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

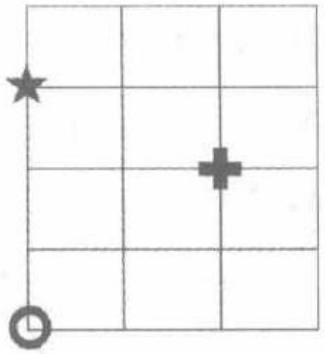
名称	产卵	鳞片	有毒	冷血	腿	爬行动物
眼镜蛇	是	有	有	是	0	是
响尾蛇	否	有	有	是	0	是
巨蚺	否	有	无	是	0	是
短吻鳄	是	有	无	是	4	是
箭毒蛙	是	无	有	否	4	否
鲑鱼	是	无	无	是	0	否
蟒蛇	是	无	无	是	0	是

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]
```

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

```
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)
```

	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

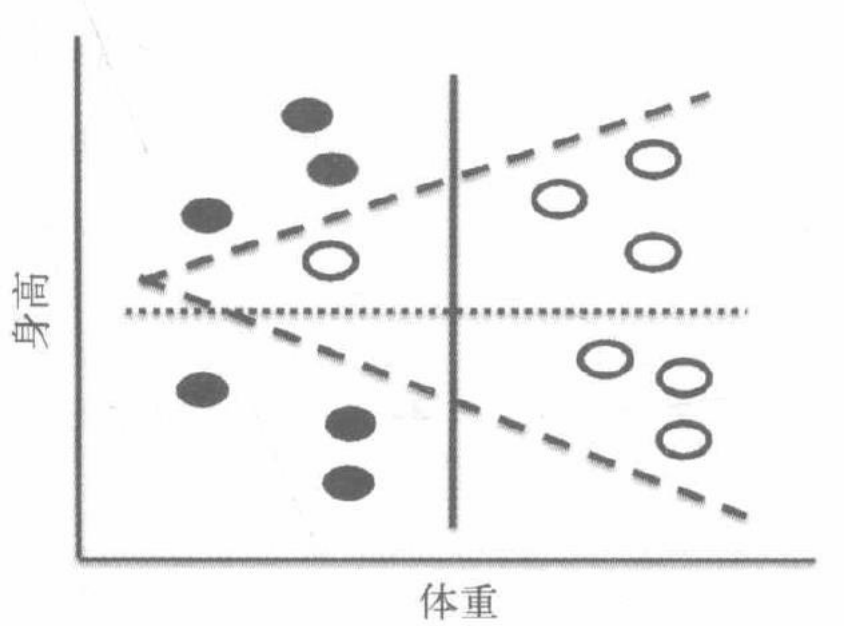


图23-1　身高、体重和衬衫类型

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

```
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
```

```
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
```

虚拟示例

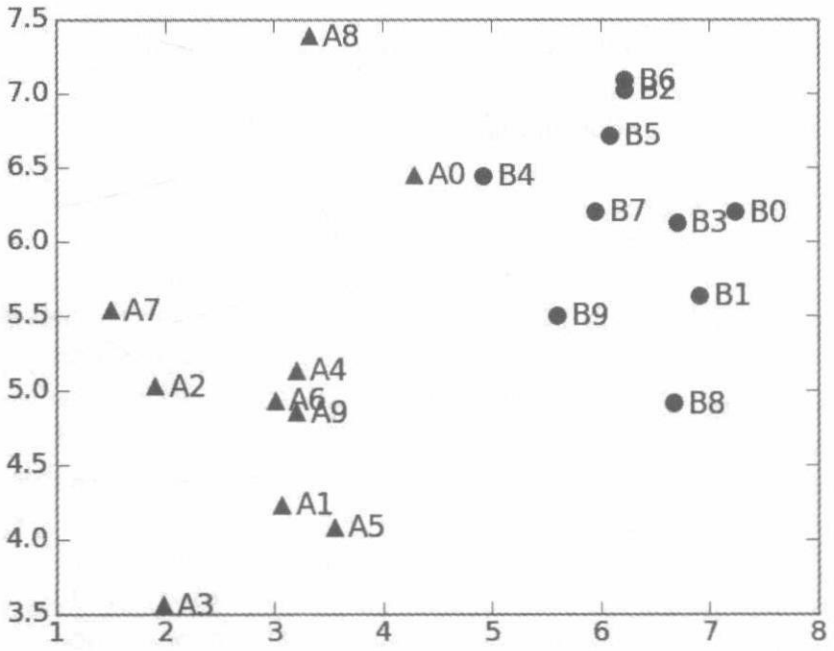


图23-6　来自两种分布的实例

```
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

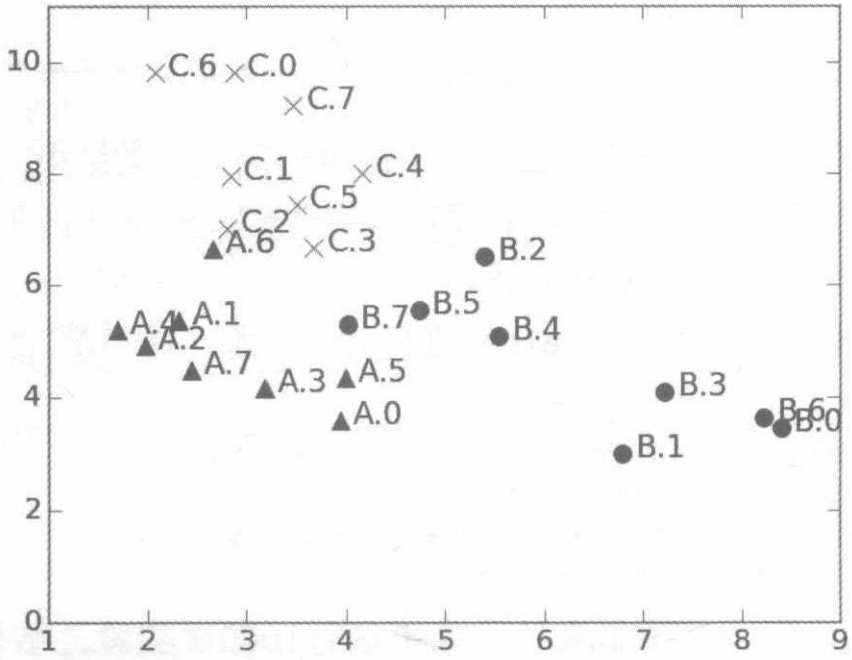


图23-10　来自3种有重合的高斯分布的数据点

```
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
```

真实例子：动物分类

- 草食动物、肉食动物、杂食动物

```
Bear, Cow, Deer, Elk, Fur seal, Grey seal, Lion, Sea lion
3 herbivores, 4 carnivores, 1 omnivores

Badger, Cougar, Dog, Fox, Guinea pig, Human, Jaguar, Kangaroo, Mink,
Mole, Mouse, Pig, Porcupine, Rabbit, Raccoon, Rat, Red bat, Skunk,
Squirrel, Wolf, Woodchuck
4 herbivores, 9 carnivores, 8 omnivores

Moose
1 herbivores, 0 carnivores, 0 omnivores
```

Clustering without scaling

Bear, Cow, Deer, Elk, Fur seal, Grey seal, Lion, Sea lion
3 herbivores, 4 carnivores, 1 omnivores

Badger, Cougar, Dog, Fox, Guinea pig, Human, Jaguar, Kangaroo, Mink, Mole, Mouse, Pig, Porcupine, Rabbit, Raccoon, Rat, Red bat, Skunk, Squirrel, Wolf, Woodchuck
4 herbivores, 9 carnivores, 8 omnivores

Moose
1 herbivores, 0 carnivores, 0 omnivores

Clustering with z-scaling

Badger, Bear, Cougar, Dog, Fox, Fur seal, Grey seal, Human, Jaguar, Lion, Mink, Mole, Pig, Raccoon, Red bat, Sea lion, Skunk, Wolf
0 herbivores, 13 carnivores, 5 omnivores

Guinea pig, Kangaroo, Mouse, Porcupine, Rabbit, Rat, Squirrel, Woodchuck
4 herbivores, 0 carnivores, 4 omnivores

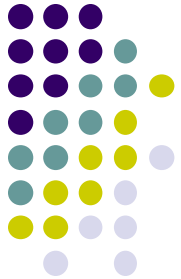
Cow, Deer, Elk, Moose
4 herbivores, 0 carnivores, 0 omnivores

Clustering with i-scaling

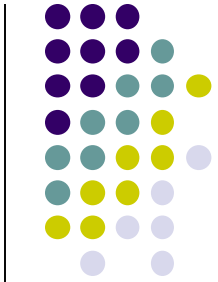
Cow, Deer, Elk, Moose
4 herbivores, 0 carnivores, 0 omnivores

Badger, Bear, Cougar, Dog, Fox, Fur seal, Grey seal, Human, Jaguar, Lion, Mink, Mole, Pig, Raccoon, Red bat, Sea lion, Skunk, Wolf
0 herbivores, 13 carnivores, 5 omnivores

Guinea pig, Kangaroo, Mouse, Porcupine, Rabbit, Rat, Squirrel, Woodchuck
4 herbivores, 0 carnivores, 4 omnivores

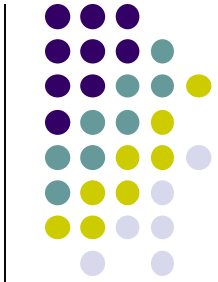


分类方法



- 分类器评价
- 问题：预测跑步者的性别
- 方法
  - K最近邻方法
  - 基于回归的分类器
- 例子：从“泰坦尼克”号生还

分类器评价



- 线性回归中多项式的阶数
  - 能够非常好的拟合现有数据
  - 能够对未知数据做出好的预测
- 训练集 vs 测试集
  - 训练误差
  - 泛化误差

准确度

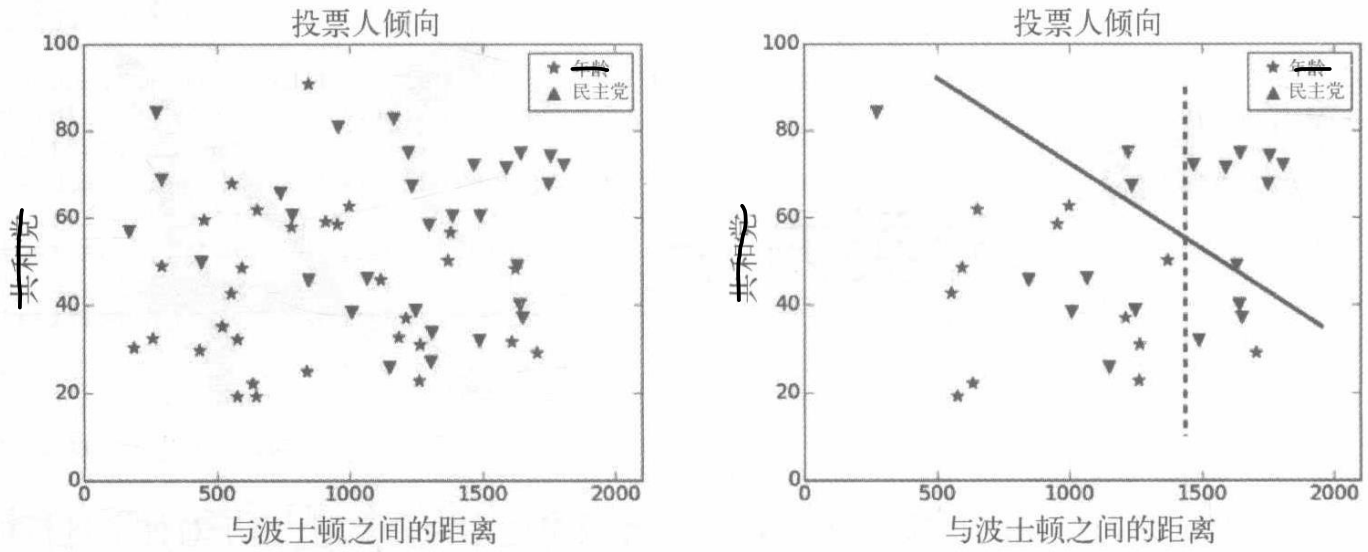


图24-1　绘制投票人倾向

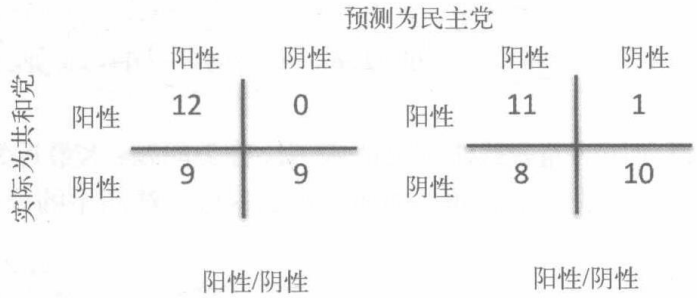


图24-2　混淆矩阵

$$\text{准确度} = \frac{\text{真阳性} + \text{真阴性}}{\text{真阳性} + \text{真阴性} + \text{假阳性} + \text{假阴性}}$$

复杂模型

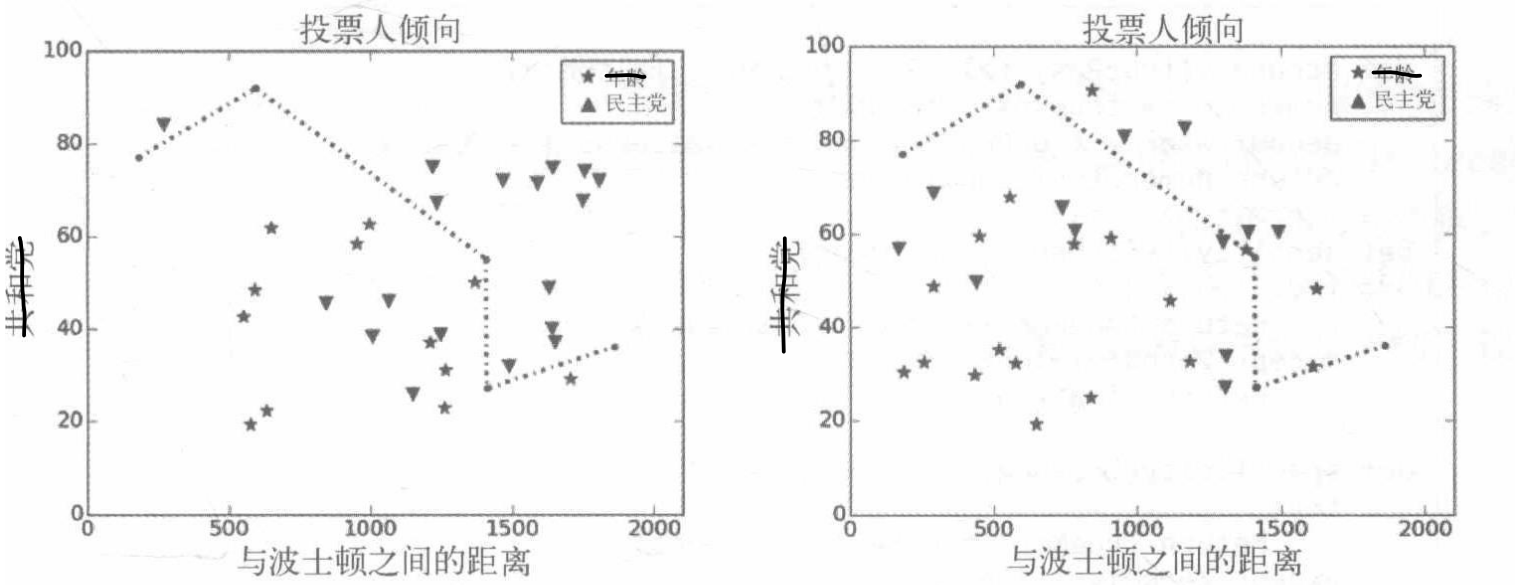


图24-3　更复杂的模型

类别不平衡

例：某种疾病的发病率0.1%

- 灵敏度（召回率）
- 特异度（精确度）

$$\text{灵敏度} = \frac{\text{真阳性}}{\text{真阳性} + \text{假阴性}}$$

$$\text{特异度} = \frac{\text{真阴性}}{\text{真阴性} + \text{假阳性}}$$

$$\text{阳性预测值} = \frac{\text{真阳性}}{\text{真阳性} + \text{假阳性}}$$

$$\text{阴性预测值} = \frac{\text{真阴性}}{\text{真阴性} + \text{假阴性}}$$

```
def accuracy(truePos, falsePos, trueNeg, falseNeg):
    numerator = truePos + trueNeg
    denominator = truePos + trueNeg + falsePos + falseNeg
    return numerator/denominator

def sensitivity(truePos, falseNeg):
    try:
        return truePos/(truePos + falseNeg)
    except ZeroDivisionError:
        return float('nan')

def specificity(trueNeg, falsePos):
    try:
        return trueNeg/(trueNeg + falsePos)
    except ZeroDivisionError:
        return float('nan')

def posPredVal(truePos, falsePos):
    try:
        return truePos/(truePos + falsePos)
    except ZeroDivisionError:
        return float('nan')

def negPredVal(trueNeg, falseNeg):
    try:
        return trueNeg/(trueNeg + falseNeg)
    except ZeroDivisionError:
        return float('nan')

def getStats(truePos, falsePos, trueNeg, falseNeg, toPrint = True):
    accur = accuracy(truePos, falsePos, trueNeg, falseNeg)
    sens = sensitivity(truePos, falseNeg)
    spec = specificity(trueNeg, falsePos)
    ppv = posPredVal(truePos, falsePos)
    if toPrint:
        print(' Accuracy =', round(accur, 3))
        print(' Sensitivity =', round(sens, 3))
        print(' Specificity =', round(spec, 3))
        print(' Pos. Pred. Val. =', round(ppv, 3))
    return (accur, sens, spec, ppv)
```

预测跑步者的性别

- 跑步者
  - 标签（性别）
  - 特征向量（年龄、完赛时间）

- 训练集（80%）
- 测试集（20%）

```
def getBMData(filename):
    """Read the contents of the given file. Assumes the file
    in a comma-separated format, with 6 elements in each entry:
    0. Name (string), 1. Gender (string), 2. Age (int)
    3. Division (int), 4. Country (string), 5. Overall time (float)
    Returns: dict containing a list for each of the 6 variables."""

    data = {}
    f = open(filename)
    line = f.readline()
    data['name'], data['gender'], data['age'] = [], [], []
    data['division'], data['country'], data['time'] = [], [], []
    while line != '':
        split = line.split(',')
        data['name'].append(split[0])
        data['gender'].append(split[1])
        data['age'].append(int(split[2]))
        data['division'].append(int(split[3]))
        data['country'].append(split[4])
        data['time'].append(float(split[5][:-1])) #remove \n
        line = f.readline()
    f.close()
    return data
```

预测跑步者的性别

- 训练集（80%）
- 测试集（20%）

```
#Figure 24.5
class Runner(object):
    def __init__(self, gender, age, time):
        self.featureVec = (age, time)
        self.label = gender

    def featureDist(self, other):
        dist = 0.0
        for i in range(len(self.featureVec)):
            dist += abs(self.featureVec[i] - other.featureVec[i])**2
        return dist**0.5

    def getTime(self):
        return self.featureVec[1]
    def getAge(self):
        return self.featureVec[0]
    def getLabel(self):
        return self.label
    def getFeatures(self):
        return self.featureVec

    def __str__(self):
        return str(self.getAge()) + ', ' + str(self.getTime())\
               + ', ' + self.label

def buildMarathonExamples(fileName):
    data = getBMData(fileName)
    examples = []
    for i in range(len(data['age'])):
        a = Runner(data['gender'][i], data['age'][i],
                   data['time'][i])
        examples.append(a)
    return examples

def divide80_20(examples):
    sampleIndices = random.sample(range(len(examples)),
                                  len(examples)//5)
    trainingSet, testSet = [], []
    for i in range(len(examples)):
        if i in sampleIndices:
            testSet.append(examples[i])
        else:
            trainingSet.append(examples[i])
    return trainingSet, testSet
```

K最近邻方法

- 计算距离
- 找出最近的K个实例
- 根据K个标签进行标记

```
examples = buildMarathonExamples('bm_results2012.txt')
training, testSet = divide80_20(examples)
truePos, falsePos, trueNeg, falseNeg =\
         KNearestClassify(training, testSet, 'M', 9)
getStats(truePos, falsePos, trueNeg, falseNeg)
```

```
Accuracy = 0.65
Sensitivity = 0.715
Specificity = 0.563
Pos. Pred. Val. = 0.684
```

```
#Figure 24.6
def findKNearest(example, exampleSet, k):
    kNearest, distances = [], []
    #Build lists containing first k examples and their distances
    for i in range(k):
        kNearest.append(exampleSet[i])
        distances.append(example.featureDist(exampleSet[i]))
    maxDist = max(distances) #Get maximum distance
    #Look at examples not yet considered
    for e in exampleSet[k:]:
        dist = example.featureDist(e)
        if dist < maxDist:
            #replace farther neighbor by this one
            maxIndex = distances.index(maxDist)
            kNearest[maxIndex] = e
            distances[maxIndex] = dist
            maxDist = max(distances)
    return kNearest, distances

def KNearestClassify(training, testSet, label, k):
    """Assumes training and testSet lists of examples, k an int
       Uses a k-nearest neighbor classifier to predict
         whether each example in testSet has the given label
       Returns number of true positives, false positives,
         true negatives, and false negatives"""
    truePos, falsePos, trueNeg, falseNeg = 0, 0, 0, 0
    for e in testSet:
        nearest, distances = findKNearest(e, training, k)
        #conduct vote
        numMatch = 0
        for i in range(len(nearest)):
            if nearest[i].getLabel() == label:
                numMatch += 1
        if numMatch > k//2: #guess label
            if e.getLabel() == label:
                truePos += 1
            else:
                falsePos += 1
        else: #guess not label
            if e.getLabel() != label:
                trueNeg += 1
            else:
                falseNeg += 1
    return truePos, falsePos, trueNeg, falseNeg
```

K最近邻方法

- 距离的计算

- 对训练数据进行采样
  - 有效减少计算量
- K值的确定
  - 奇数

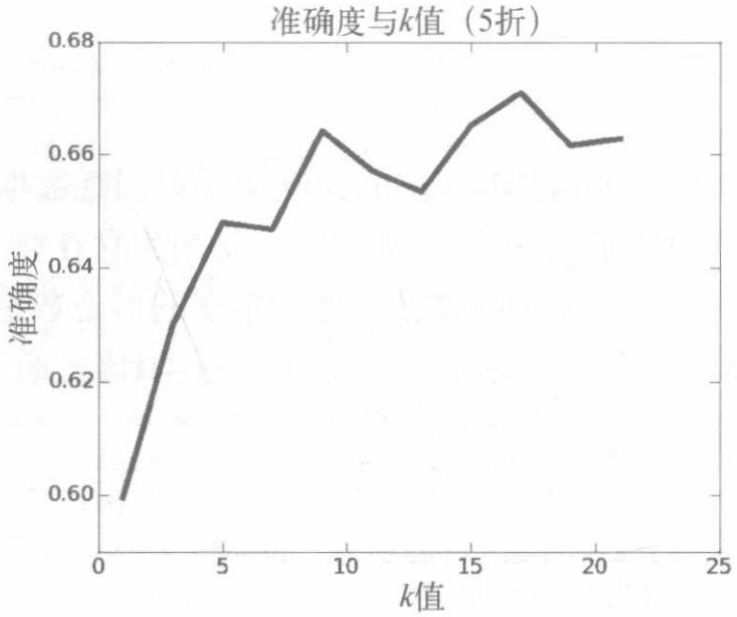


图24-9　选择k值

基于回归的分类器

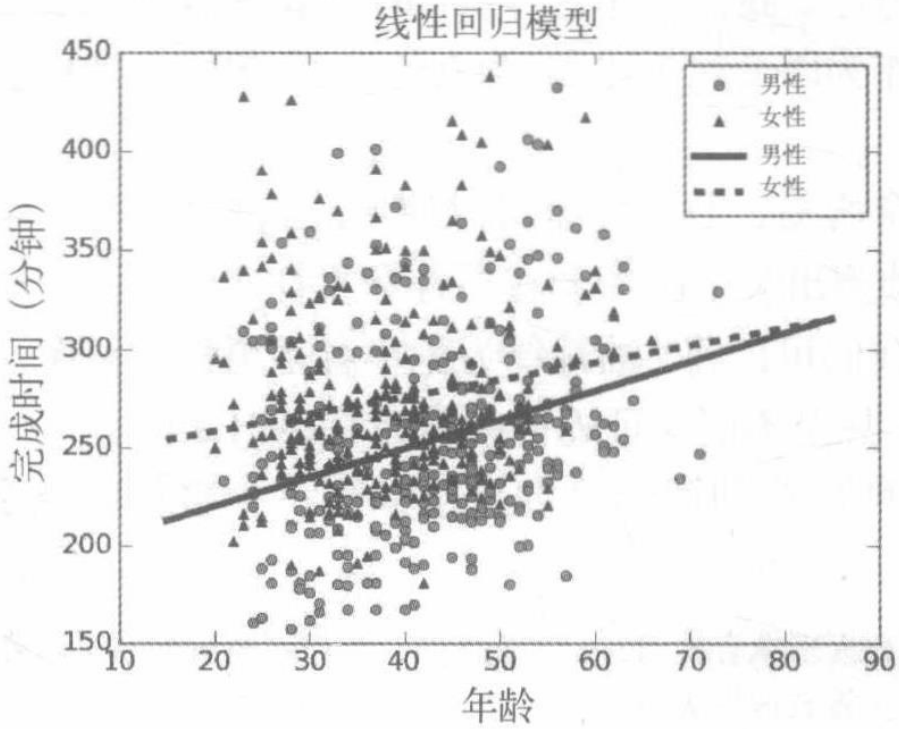


图24-11　男性和女性的线性回归模型

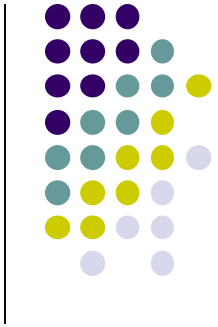
准确度=0.616
灵敏度=0.682
特异度=0.529
阳性预测值=0.657

Logistic回归模型

- 直接预测一个事件的概率

- https://scikit-learn.org/stable/

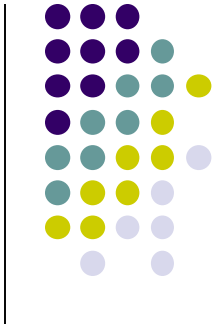
  - Simple and efficient tools for predictive data analysis
  - Accessible to everybody, and reusable in various contexts
  - Built on NumPy, SciPy, and matplotlib
  - Open source, commercially usable - BSD license



```
import sklearn.linear_model

featureVecs, labels = [], []
for i in range(25000): #create 4 examples in each iteration
    featureVecs.append([random.gauss(0, 0.5), random.gauss(0, 0.5),
                        random.random()])
    labels.append('A')
    featureVecs.append([random.gauss(0, 0.5), random.gauss(2, 0.),
                        random.random()])
    labels.append('B')
    featureVecs.append([random.gauss(2, 0.5), random.gauss(0, 0.5),
                        random.random()])
    labels.append('C')
    featureVecs.append([random.gauss(2, 0.5), random.gauss(2, 0.5),
                        random.random()])
    labels.append('D')
model = sklearn.linear_model.LogisticRegression().fit(featureVecs,
                                                      labels)
print('model.classes_ =', model.classes_)
for i in range(len(model.coef_)):
    print('For label', model.classes_[i],
          'feature weights =', model.coef_[i])
print('[0, 0] probs =', model.predict_proba([[0, 0, 1]])[0])
print('[0, 2] probs =', model.predict_proba([[0, 2, 2]])[0])
print('[2, 0] probs =', model.predict_proba([[2, 0, 3]])[0])
print('[2, 2] probs =', model.predict_proba([[2, 2, 4]])[0])
```

```
model.classes_ = ['A' 'B' 'C' 'D']
For label A feature weights = [-4.65720783 -4.38351299 -0.00722845]
For label B feature weights = [-5.17036683 5.82391837 0.04706108]
For label C feature weights = [ 3.95940539 -3.97854738 -0.04480206]
For label D feature weights = [ 4.37529465 5.40639909 -0.09434664]
[0, 0] probs = [ 9.90019074e-01 4.66294343e-04 9.51434182e-03
2.90294956e-07]
[0, 2] probs = [ 8.72562747e-03 9.78468475e-01 3.18006160e-06
1.28027180e-02]
[2, 0] probs = [ 5.22466887e-03 1.69995686e-08 9.93218655e-01
1.55665885e-03]
[2, 2] probs = [ 7.88542473e-07 1.97601741e-03 7.99527347e-03
9.90027921e-01]
```



```
#Figure 24.14
featureVecs, labels = [], []
for i in range(20000):
    featureVecs.append([random.gauss(0, 0.5), random.gauss(0, 0.5)])
    labels.append('A')
    featureVecs.append([random.gauss(2, 0.5), random.gauss(2, 0.5)])
    labels.append('D')
model = sklearn.linear_model.LogisticRegression().fit(featureVecs,
                                                      labels)
print('model.coef =', model.coef_)
print('[0, 0] probs =', model.predict_proba([[0, 0]])[0])
print('[0, 2] probs =', model.predict_proba([[0, 2]])[0])
print('[2, 0] probs =', model.predict_proba([[2, 0]])[0])
print('[2, 2] probs =', model.predict_proba([[2, 2]])[0])
```

```
model.coef = [[ 5.79284554 5.68893473]]
[0, 0] probs = [ 9.99988836e-01   1.11643397e-05]
[0, 2] probs = [ 0.50622598 0.49377402]
[2, 0] probs = [ 0.45439797 0.54560203]
[2, 2] probs = [ 9.53257749e-06   9.99990467e-01]
```

马拉松例子

```
def applyModel(model, testSet, label, prob = 0.5):
    #Create vector containing feature vectors for all test examples
    testFeatureVecs = [e.getFeatures() for e in testSet]
    probs = model.predict_proba(testFeatureVecs)
    truePos, falsePos, trueNeg, falseNeg = 0, 0, 0, 0
    for i in range(len(probs)):
        if probs[i][1] > prob:
            if testSet[i].getLabel() == label:
                truePos += 1
            else:
                falsePos += 1
        else:
            if testSet[i].getLabel() != label:
                trueNeg += 1
            else:
                falseNeg += 1
    return truePos, falsePos, trueNeg, falseNeg
```

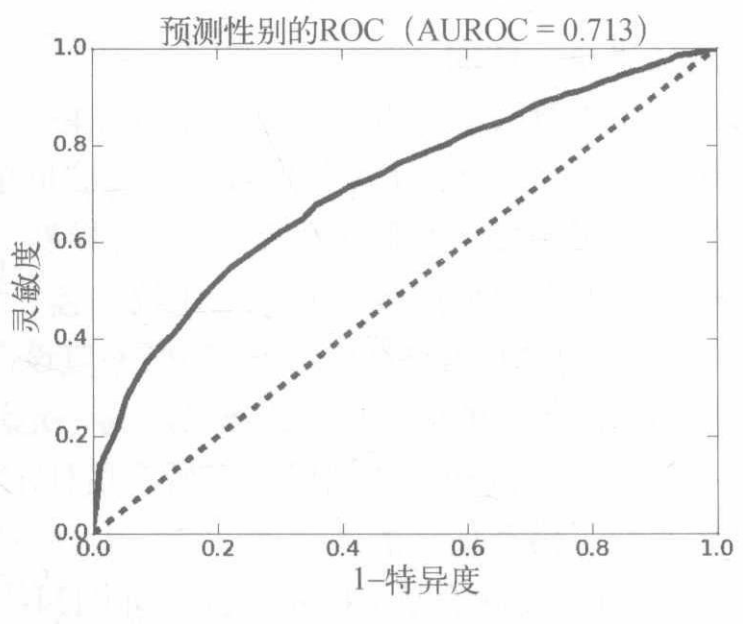
```
Feature weights for label M: age = 0.055, time = -0.011
 Accuracy = 0.635
 Sensitivity = 0.831
 Specificity = 0.377
 Pos. Pred. Val. = 0.638
```

```
examples = buildMarathonExamples('bm_results2012.txt')
training, test = divide80_20(examples)

featureVecs, labels = [], []
for e in training:
    featureVecs.append([e.getAge(), e.getTime()])
    labels.append(e.getLabel())
model = sklearn.linear_model.LogisticRegression().fit(featureVecs,
                                                      labels)
print('Feature weights for label M:',
      'age =', str(round(model.coef_[0][0], 3)) + ',',
      'time =', round(model.coef_[0][1], 3))
truePos, falsePos, trueNeg, falseNeg = \
                    applyModel(model, test, 'M', 0.5)
getStats(truePos, falsePos, trueNeg, falseNeg)
```

ROC曲线

```
def buildROC(model, testSet, label, title, plot = True):
    xVals, yVals = [], []
    p = 0.0
    while p <= 1.0:
        truePos, falsePos, trueNeg, falseNeg =\
                                applyModel(model, testSet, label, p)
        xVals.append(1.0 - specificity(trueNeg, falsePos))
        yVals.append(sensitivity(truePos, falseNeg))
        p += 0.01
    auroc = sklearn.metrics.auc(xVals, yVals, True)
    if plot:
        pylab.plot(xVals, yVals)
        pylab.plot([0,1], [0,1,], '--')
        pylab.title(title + ' (AUROC = '\
                    + str(round(auroc, 3)) + ')')
        pylab.xlabel('1 - Specificity')
        pylab.ylabel('Sensitivity')
    return auroc
```



buildROC(model, test, 'M', 'ROC for Predicting Gender')

从“泰坦尼克”号生还

- 1912年4月15日清晨，沉没在北大西洋
- 1046位乘客信息
  - 舱位等级、年龄、性别、是否生还、姓名