第二章 k近邻算法

莉莉已经年过30仍然未婚，作为大龄剩女，父母逼婚甚急。莉莉使用在线约会网站寻找适合自己的约会对象已经有很长时间了，尽管网站会推荐不同的人选，但莉莉发现并不是每一个网站推荐的人选她都喜欢。经过一段时间的总结，莉莉将交往过的人分为三类：

- 不喜欢的人
- 魅力一般的人
- 极具魅力的人

此外，莉莉还收集了这些交往对象的一些数据信息，包括：

- 每年飞行多少公里
- 玩视频游戏所耗时间的百分比
- 每周消耗冰淇淋多少升

莉莉认为这些特征有助于匹配对象的分类。莉莉将每一位交往对象的特征及所属类别都存到了一个文件datingTestSet.txt中，她希望利用分类软件帮助她将网站新推荐的人选划分到确切的分类中，以提高约会成功的几率。

1. k近邻算法的原理

k近邻算法（k-Nearest Neighbor, kNN）是1967年由Cover和Hart提出的一种基本分类与回归方法。它的工作原理是：存在一个样本数据集合，也称为训练样本集，并且样本集中每个数据都存在标签，即我们知道样本集中每一个数据与所属分类的对应关系。输入没有标签的新数据后，将新的数据的每个特征与样本集中数据对应的特征进行比较，然后算法提取样本最相似数据（最近邻）的分类标签。

一般来说，我们只选择样本数据集中前k个最相似的数据，这就是k近邻算法中k的出处，通常k不大于20。最后，选择k个最相似数据中出现次数最多的分类，作为新数据的分类。

比如，电影可以按题材来进行分类，有爱情片、动作片等等，我们先假设只有这两种类型。然后选择电影中接吻镜头数目及打斗镜头数目来对电影分类。假设我们已经有了一个样本集，如表2-1所示。

表2-1 电影分类样本集

电影名称	打斗镜头数	接吻镜头数	电影类型
California Man	3	104	爱情片
He's Not Really into Dudes	2	100	爱情片
Beautiful Woman	1	81	爱情片
Kevin Longblade	101	10	动作片
Robo Slayer 3000	99	5	动作片
Amped II	98	2	动作片
新电影	18	90	?

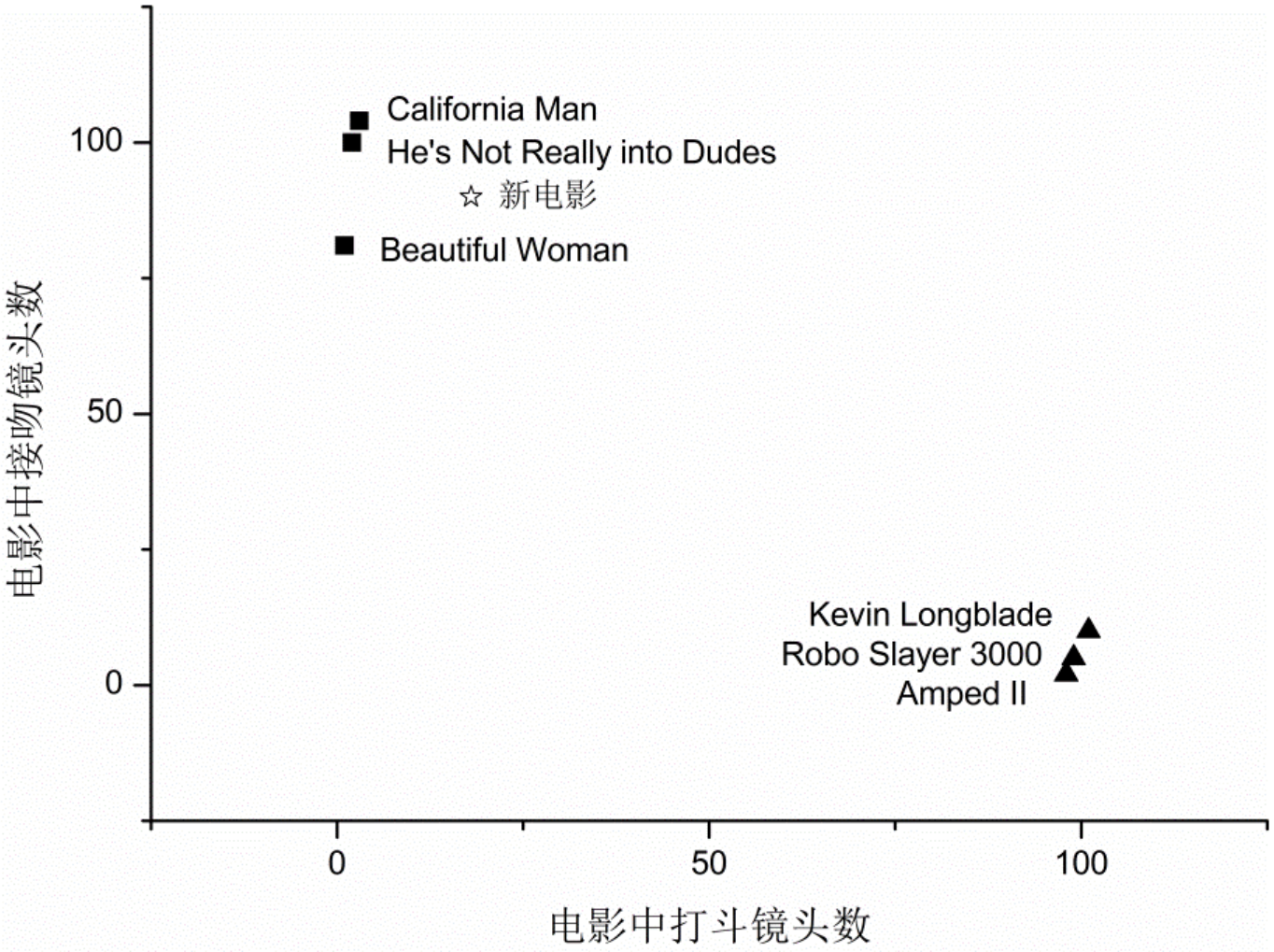

图2-1 使用打斗和接吻镜头数来分类电影

我们计算一下新电影与每部电影之间的距离，按欧氏距离计算，即两个点A（x_1, y_1）与B（x_2, y_2）之间的距离d为：

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

新电影与样本集中各部电影之间的距离如表2-2所示。假定k=3，则三个最靠近的电影全部是爱情片，因此判定新电影为爱情片。

表2-2 新电影与样本集中各部电影之间的距离

电影名称	与新电影的距离	电影类型
California Man	20.5	爱情片
He's Not Really into Dudes	18.7	爱情片
Beautiful Woman	19.2	爱情片
Kevin Longblade	115.3	动作片
Robo Slayer 3000	117.4	动作片
Amped II	118.9	动作片

2. kNN算法的实现

1）定义函数load从文件中读入数据。
2）定义class kNN实现分类
3）在主程序中进行测试

3. 模型的评估

我们训练得到一个模型的目的是对“新样本”进行预测，一个模型适用于新样本的能力称为“泛化”（generalization）能力。分类错误的样本数占样本总数的比例称为“错误率”（error rate），即如果在m个样本中有a个样本分类错误，则错误率为E=a/m，相应的，1-a/m称为“精度”（accuracy）。我们把学习器在训练集上的误差称为“训练误差”（training error），在新样本上的误差称为“泛化误差”（generalization error）。

3.1 评估方法

为了评估一个学习器的泛化能力，我们可以使用一个“测试集”（testing set）来考察学习器对新样本的判别能力，以测试集上的“测试误差”（testing error）作为泛化误差的近似。
从样本集产生训练集和测试集的常用方法：

1）留出法（hold-out）

训练/测试集的划分要尽可能保持数据分布的一致性。
单次使用留出法得到的估计结果往往不够稳定可靠，通常要多次随机划分取平均值作为评估结果。

2）交叉验证法（cross validation）

K折交叉验证（k-fold cross validation），常用的k是10，5等。
训练集（training set ），
验证集（validation set），
测试集（testing set）

3.2 性能度量

对学习器的泛化能力进行评估，需要一个评价标准，这就是性能度量（performance measure）。模型的“好坏”是相对的，哪个模型更好，不仅取决于算法和数据，还决定于任务需求。

1) 错误率与精度

错误率与精度是分类任务中最常用的两种性能度量。

但对偏斜（skewed）样本，即其中类别分布严重不均衡的样本，错误率和精度并不是一个好的评估指标。

2) 查准率、查全率与F1

例如对于癌症预测问题，我们更关心的是：

- 预测阳性病例中有多少是真正的阳性？
- 真正的阳性病倒中有多少被预测出来？

这就是查准率(precision)与查全率(recall)的概念。

以二分类问题为例，可将样本按真实类别与学习器预测类别划分为

- 真阳性（true positive）
- 假阳性（false positive）
- 真阴性（true negative）
- 假阴性（false negative）

四种情形的样本数分别用TP、FP、TN、FN表示，显然有TP+FP+TN+FN=样本总数。分类结果的“混淆矩阵”（confusion matrix）如表2-3所示。

表2-3 分类结果混淆矩阵

真实类别	预测类别	
	阳性	阴性
阳性	TP（真阳性）	FN（假阴性）
阴性	FP（假阳性）	TN（真阴性）

查准率（P）和查全率（R）分别定义为：

$$P = \frac{TP}{TP + FP} \qquad R = \frac{TP}{TP + FN}$$

查准率和查全率是一对矛盾的量，一般来说，查准率高时查全率往往偏低，而查全率高时查准率往往偏低。

假设一个模型A的查准率为0.9，查全率为0.4，另一个模型B的查准率为0.7，查全率为0.6，那么哪个模型更好呢？

一个综合考虑查准率与查全率的指标是F1：

$$\frac{1}{\mathrm{F1}}=\frac{1}{2}\left(\frac{1}{\mathrm{P}}+\frac{1}{\mathrm{R}}\right)\rightarrow \mathrm{F1}=\frac{2\mathrm{PR}}{\mathrm{P}+\mathrm{R}}$$

F1是P与R的调和平均值。

对一些具体的应用，对查准率和查全率的重视程序会有不同。

比如在商品推荐系统中，为了尽可能少打扰用户，更希望推荐内容确是用户感兴趣的，此时查准率更重要。

而在逃犯信息检索系统中，更希望尽可能少漏掉逃犯，此时查全率更重要。

F1度量的更一般形式F_β，可以兼顾到对查准率与查全率的不同偏好:

$$\frac{1}{F_\beta}=\frac{1}{1+\beta^2}\left(\frac{1}{P}+\frac{\beta^2}{R}\right)\rightarrow F_\beta=\frac{(1+\beta^2)PR}{\beta^2P+R}$$

其中β>0表示查全率对查准率的相对重要性，β=1时退化为标准的F1，β>1时查全率更重要，β<1时查准率更重要。

4. 利用kNN算法改进约会网站配对效果

取k=3，以所有样本为训练数据，同样以所有样本为测试数据，看看预测精度如何。程序输出结果为:

预测精度: 0.885

这个精度已经比较高，有没有办法继续提高精度呢？

4.1 归一化

数值归一化在机器学习领域是很常用的，尤其是当不同的特征取值相差比较大的时候。

归一化常用的方法是将不同属性值都缩放到相同的特定区间，比如，利用下面的公式就可以将任意取值范围的特征值转化为0到1区间内的值：

$$newValue = (oldValue - min)/(max - min)$$

其中min和max分别是该属性取值中的最小值和最大值。另外一种常用的归一化方法称为Z-Score标准化(Z-Score standardization)或均值归一化(mean normalization)，它采用下面的公式：

$$newValue = (oldValue - mean)/delta$$

其中mean和delta分别为该属性所有数据的均值和方差。经过标准化，该属性数据将呈现标准正态分布，均值为0、方差为1。

先对约会网站的数据进行归一化操作，然后再利用kNN算法测试一下其预测效果，输出结果为：

预测精度: 0.973

4.2 kNN算法评估与k值选择

1) 留出法评估
2) 交叉验证法评估
3) 模型的应用

5. kNN算法的关键要素

kNN算法依据距离选择最近的k个邻居来决定新样本的类别，那么距离的计算方法与k值的选择无疑会影响预测结果及模型准确性。

5.1 距离的计算

两个点X与Y的坐标为：$X=(x_1,x_2,\cdots,x_n)$，$Y=(y_1,y_2,\cdots,y_n)$

闵可夫斯基距离定义为：

$$\left(\sum_{i=1}^{n}|x_i-y_i|^p\right)^{1/p}$$

欧氏距离即为闵可夫斯基距离当p=2时的特例。

当p=1时，常称为曼哈顿距离：

$$\sum_{i=1}^{n}|x_i-y_i|$$

当 p 趋于无穷大时，称为切比雪夫距离，它是各个坐标距离的最大值：

$$\lim_{p\to\infty}\left(\sum_{i=1}^{n}|x_i-y_i|^p\right)^{1/p}=\max_i|x_i-y_i|$$

5.2 k值的选择

k值的不同也会对模型的预测准确率造成影响，如图2-4所示。

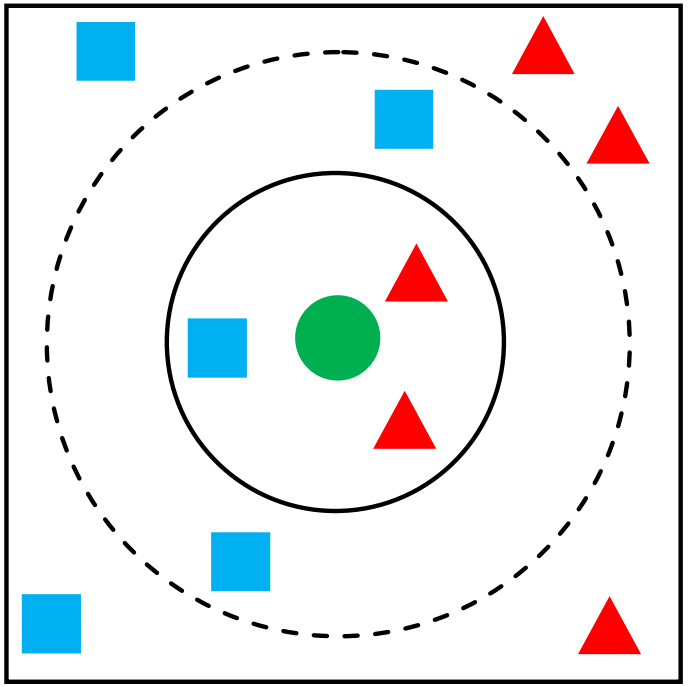

图2-4 kNN算法图示

（绿色为待分类点，红色三角与蓝色方块分别代表两个类别）

一般来说:

- 较小的*k*值，整体模型变得复杂，容易发生过拟合。
- 较大的*k*值，模型变得简单，容易欠拟合。

在实际应用中，*k*值一般取一个比较小的数值，或者采用交叉验证法来选择最优的*k*值。

6. kNN算法的优缺点

优点：

- 简单，易于理解，易于实现，无需训练；
- 对异常值不敏感；
- 适合于多分类问题。

缺点：

- 当样本不平衡时，如一个类的样本容量很大，而其他类样本容量很小时，容易产生误分类。
- 计算量较大，对每一个新样本都要计算它与全体已知样本的距离。

7. kNN算法的发展

为了克服kNN算法的缺点，人们提出了一些修正方法以提高kNN算法的预测准确率及运行效率。

- 增加近邻的权重，距离越近则权重越高。
- 指定一定的距离为半径，以该半径内的近邻来确定新样本的分类。

为了解决kNN算法计算量大的问题：

- KD树(KD Tree) 算法，把距离信息保存在一棵树里，在计算之前从树中查询距离信息，尽量避免重新计算。
- Ball树(Ball Tree)算法，基于KD树进一步优化算法性能。

8. 利用scikit-learn模块实现kNN算法

8.1 scikit-learn中的数据归一化

在scikit-learn程序包的preprocessing模块中实现了MinMaxScaler类和StandardScaler类用于数据的归一化。

MinMaxScaler类将各属性值缩放到特定的区间，其用法：

MinMaxScaler(feature_range = (0, 1), copy = True)

MinMaxScaler类中的常用方法有：

- fit(X, y = None)
- transform(X)
- fit_transform(X, y = None)
- inverse_transform(X)

StandardScaler类将各属性值标准化到均值为0方差为1的标准正态分布，其用法：

StandardScaler(copy = True, with_mean = True, with_std = True)

StandardScaler类的常用方法有：

- fit(X, y = None)
- transform(X)
- fit_transform(X, y = None)
- inverse_transform(X)

我们看到StandardScaler类与MinMaxScaler类的方法差不多，这是Scikit-learn的特点，很多机器学习方法都具有相似的方法，如fit, predict, score等。

8.2 scikit-learn中的kNN算法分类器

在scikit-learn程序包的neighbors模块中，实现了:

- KNeighborsClassifier，k近邻算法分类器
- RadiusNeighborsClassifier，固定半径的分类器
- NearestNeighbors，无监督学习的近邻搜索器
- KNeighborsRegressor ，k近邻回归器

KNeighborsClassifier类的用法：

KNeighborsClassifier(n_neighbors=5, weights='uniform',

algorithm='auto', leaf_size=30, p=2, metric='minkowski',

metric_params=None, n_jobs=1, **kwargs)

KNeighborsClassifier类的常用方法包括：

- fit(X, y): 利用属性矩阵X及标签向量y对分类器进行训练。
- kneighbors(X = None, n_neighbors = None, return_distance = True): 找到点X的n_neighbors个近邻。
- predict(X): 预测X的类别。
- score(X, y, sample_weight = None): 返回分类器用于测试集X及其标签向量y的平均准确率。

8.3 scikit-learn中的数据分割

在scikit-learn中的model_selection模块实现了train_test_split函数，用于将样本集分割为训练集与测试集，其用法如下：

train_test_split(X, y, test_size = 0.25, train_size = None, random_state = None, shuffle = True, stratify = None)

在model_selection模块中实现了多种数据分割的类：

- KFold用于K折交叉验证
- LeaveOneOut用于留一法交叉验证
- RepeatedKFold用于重复K折交叉验证

KFold类的用法如下：

KFold(n_splits = 3, shuffle = False, random_state = None)

KFold类通过方法split(X, y = None)将样本集分割为训练集及测试集，返回训练集和测试集的索引。

在model_selection模块中还实现了cross_val_score函数利用交叉验证对模型评分，其用法如下:

cross_val_score(estimator, X, y=None, groups=None, scoring=None, cv=None, n_jobs=None, verbose=0, fit_params=None, pre_dispatch='2*n_jobs', error_score='raise-deprecating')

8.4 利用scikit-learn实现约会网站问题的kNN算法