https://mp.weixin.qq.com/s/mmyo1iaTxYdz9Y77LZVUuA

# 准备

机器学习是什么，人工智能的子类，深度学习的父类。

## 机器学习

使计算机改进或是适应他们的行为，从而使他们的行为更加准确。也就是通过数据中学习，从而在某项工作上做的更好。

引用王钰院士在2008年会议的一句话，假定W是给定世界的有限或者无限的所有对象的集合，Q是我们能够或得到的有限数据，Q是W的一个很小的真子集，机器学习就是根据世界的样本集来推算世界的模型，使得模型对于整体世界来说为真。

### 机器学习的两个驱动

神经网络，数据挖掘。

### 机器学习的分类

#### 监督学习

提供了包含正确回答的训练集，并以这个训练集为基础，算法进行泛化，直到对所有的可能输入都给出正确回答，这也称在范例中学习。

#### 无监督学习

没有提供正确回答，算法试图鉴别出输入之间的相似，从而将同样的输入归为一类，这种方法称密度学习。

#### 强化学习

介于监督和无监督之间，当答案不正确时，算法被告知，如何改正则不得而知，算法需要去探索，试验不同情况，直到得到正确答案，强化学习有时称为伴随评论家的学习，因为他只对答案评分，而不给出改进建议。

#### 进化学习

将生物学的进化看成一个学习过程，我们研究如何在计算机中对这一过程进行建模，采用适应度的概念，相当于对当前解答方案好坏程度的评分。（不是所有机器学习书籍都包含进化学习）

### 优点

泛化，对于未曾碰到的输入也能给出合理的输出。

### 监督学习

回归、分类。

### 机器学习过程：

1. 数据的收集和准备
2. 特征选择
3. 算法选择
4. 参数和模型选择
5. 训练
6. 评估

## 专业术语：

### 输入

输入向量x作为算法输入给出的数据

### 突触

wij是节点i和节点j之间的加权连接，类似于大脑中的突触，排列成矩阵W

### 输出

输出向量y，可以有n个维度

### 目标

目标向量t，有n个维度，监督学习所需要等待额外数据，提供了算法正在学习的“正确答案”

### 维度

输入向量的个数

### 激活函数

对于神经网络，g（·）是一种数学函数，描述神经元的激发和作为对加权输入的响应

### 误差

E是根据y和t计算网络不准确性的函数

### 权重空间

当我们的输入数据达到200维时，人类的限制使得我们无法看见，我们最多只能看到三维投影，而对于计算机可以抽象出200个相互正交的轴的超平面进行计算，神经网络的参数是将神经元连接到输入的一组权重值，如将神经元的权重视为一组坐标，即所谓的权重空间

### 维度灾难：

随着维度的增加，单位超球面的体积也在不断增加，2d中，单位超球面为圆，3d中则为球，而更高的维度便称为超球面，Vn = （2π/n）\*Vn-2，于是当n>2π时，体积开始缩小，因此可用数据减少，意味着我们需要更多的数据，当数据到达100维以上时，单位数据变得极小，进而需要更多的数据，从而造成维度灾难

**维度和体积的关系：**

|  |  |
| --- | --- |
| 维度 | 体积 |
| 1 | 2.0000 |
| 2 | 3.1416 |
| 3 | 4.1888 |
| 4 | 4.9348 |
| 5 | 5.2636 |
| 6 | 5.1677 |
| 7 | 4.7248 |
| 8 | 4.0587 |
| 9 | 3.2985 |
| 10 | 2.5502 |

### 机器学习算法测试

算法成功程度是预测和一直目标进行比较，对此我们需要一组新的数据，测试集。

当对算法进行训练时，过度的训练将会导致过拟合，即拟合曲线与数据完美拟合，但是失去了泛化能力，为检测过拟合我们需要用测试集进行验证，称为统计中的交叉验证，它是模型选择中的一部门：为模型选择正确的参数，以便尽可能的泛化。

数据的准备，我们需要三组数据集，训练算法的训练集，跟踪算法学习效果的验证集，用于产生最终结果的测试集，数据充足情况便执行50:25:25或60:20:20的划分，数据集分配应随机处理，当数据请核实板块，则采用流出方法或多折交叉验证。

混淆矩阵是检测结果是否良好的分类，制作一个方阵，其包含水平和垂直方向上所有可能的类，在（i，j）处的矩阵元素告诉我们在目标中有多少模式被放入类i中，主对角线上任何东西都是正确答案，主对角线元素之和除以所有元素的和，从而得到的百分比便是精度。

### 精度指标

真正例是被正确放入类1，假正例是被错误放入类1，而真反例是被正确放入类2，假反例是被错误放入类2。

|  |  |
| --- | --- |
| 真正例（TP） | 假正例（FP） |
| 假反例（FN） | 真反例（TN） |

* 敏感率=#TP/(#TP+#FN) 特异率=#TN/(#TN+#FP)
* 查准率=#TP/(#TP+#FP) 查全率=#TP/(#TP+#FN)
* F1 = 2\*(查准率\*查全率)/(查准率+查全率)

### 受试者工作曲线

y轴真正例率，x轴假正例率，线下区面积：AUC。

### 数据与概率的转换

通过贝叶斯法则处理联合概率P(C,X)和条件概率P(X|C)得出P(C|X)，MAP问题是训练数据中最可能的类是什么。将所有类的最终结果考虑在内的方法称为贝叶斯最优分类。

### 损失矩阵

指定类Ci被分为类Cj所涉及的风险。

### 基本统计概念

协方差，度量两个变量的依赖程度。

Cov（{xi},{yi}）=E({xi} – u)E({yi} – v)

### 权衡偏差与方差

偏差-方差困境：更复杂的模型不一定能产生更好的结果；模型糟糕可能由于两个原因，模型不准确而与数据不匹配，或者不精确而有极大的不稳定性。第一种情况称为偏差，第二种情况称为方差。

# 神经元、神经网络和线性判别

## 鲁棒性

鲁棒是Robust的音译，也就是健壮和强壮的意思。它是在异常和危险情况下系统生存的关键。比如说，计算机软件在输入错误、磁盘故障、网络过载或有意攻击情况下，能否不死机、不崩溃，就是该软件的鲁棒性。

## 神经网络

神经网络模仿的便是生物学中的神经网络，通过输入进而判定神经元激活否。

将一系列的神经元放置在一起，假设数据存在模式。通过神经元一些已知的样例，我们希望他能够发现这种模式，并且正确预测其他样例，称为模式识别。为了让神经网络能够学习，我们需要改变神经元的权重和阈值进而得到正确的结果，历史上的第一个神经网络——感知器。

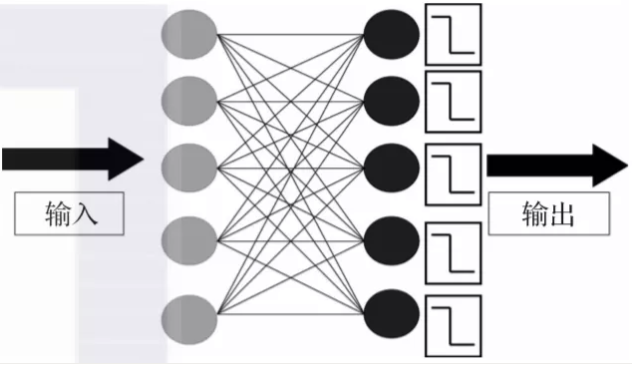
## Hebb法则

突触连接强度的变化和两个相连神经元激活得相关性成比例，如果两个神经元始终同时激活，那么他们之间连接的强度会变大，反之，如果两个神经元从来不同时激活，那么他们之间的连接会消失。也被成为长时效增强法则和神经可塑性。

## McCulloch和Pitts神经元

建模，一组输入加权wi相当于突触，一个加法器把输入信号相加（等价于收集电荷的细胞膜），一个激活函数，决定细胞对于当前的输入是否激活，输入乘于权重的和与阈值进行判断，大于则激活，否则抑制。局限性：现实中的神经元不给出单一的输出相应，而是给出一个点位序列，一种连续的方式给出分等级的输出。神经元不会根据电脑的时钟脉冲去顺序更新，而是随机的异步更新。

## 感知器



▲感知器神经网络

#### 权重更新规则

Wij <- Wij – n(yi – ti)\*xi

N为学习效率，过大会造成网络不稳定，过小会造成学习时间久；yi为神经元输出，ti为神经元目标，xi为神经元输入，Wij为权重。

#### 感知器学习算法

分为两部分，训练阶段和再现阶段。

初始化

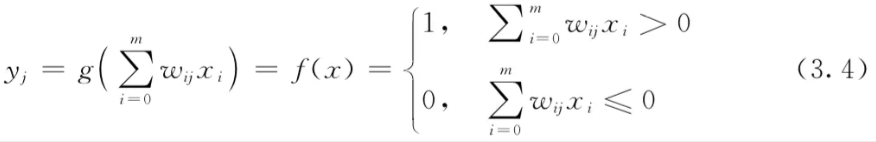
设置所有的权重wij为小的随机数（正或负都可）。

训练

对T次循环

对每一个输入向量：

利用激活函数g计算每一个神经元j的激活状态：

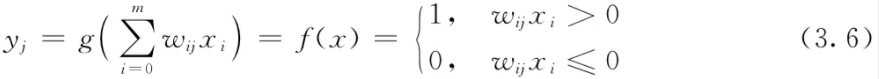


利用下式更新每一个权重：



再现

利用下式计算每一个神经元j的激活状态：



## 线性可分性

一条直线将神经元激活的和不激活的神经元划分开来，这条直线称为决策边界，也称为判别函数，在三维空间该决策边界为平面，更高维则为超平面。

## 感知器收敛定理

感知器以1/γ\*γ为界，其中γ为分离超平面与最接近的数据点之间的距离。

只要把数据映射到正确的维度空间，那么总是可以用一个线性函数来把两个类别区分开，为了较有效率的解决这个问题，有一整类的方法称为核分类器，也是支持向量机的基础。

## 数据项预处理

特征选择，我们每次去掉一个不同的特征，然后试着在所得的输入子集上训练分类器，看结果是否有所提高，如果去掉某一个特征能使得结果有所改进，那么久彻底去掉他，在尝试能否去掉其他的特征，这是一个测试输出与每一个特征的相关性的过于简单方法。

## 线性回归

回归问题是用一条线去拟合数据，而分类问题是寻找一条线来划分不同类别。回归方法，引入一个指示变量，它简单的标识每一个数据点所属的类别。现在问题就变成了用数据去预测指示变量，第二种方法是进行重复的回归，每一次对其中的一个类别，指示值为1代表样本属于该类别，0代表属于其他类别。

# 维度简约

## 特征选择法

根据一定的经验和规则，选取原有维度的一部分参与后续的计算和建模。用所选维度代替所有维度，与输出变量关联起来，整个过程不产生新维度。有四种思路：

* 经验法
* 测算法
* 基于统计分析的方法
* 机器学习算法

### ~~特征推导法~~

~~通过应用数据迁移，即通过可以用矩阵来描述的平移和旋转来改变图标的坐标系，从而用旧的特征推导出新的特征，因为他允许联合特征，并且坚定哪一个是有用的，哪一个没用~~

### ~~聚类法~~

~~把相似的数据点放一起，看能不能有更少的特征~~

## 基于特征转换的降维

将高维空间的数据点映射到低维空间中，映射后变量的特征来表示原有变量的总体特征。

独立成分分析ICA，主成分分析PCA，因子分析FA，线性判别分析LDA，局部线性嵌入LLE，核主成分分析Kernel PCA等。

### 主成分分析（PCA）

主成分的概念是数据中变化最大的方向。算法首先通过减去平均值来把数据集中，选择变化最大的方向并把它设为坐标轴，然后检查余下的变化并且找一个坐标轴使得它垂直于第一个并且覆盖尽可能多的变化。

不断重复这个方法直到找到所有可能的坐标轴。这样的结果就是所有的变量都是沿着直角坐标系的轴，并且协方差矩阵是对角的——每个新变量都与其他变量无关，而只与自己有关。一些变化非常小的轴可以去掉不影响数据的变化性。

#### 具体算法

1. 写成N个点Xi=(X1i，X2i，... xXi)作为行向量。
2. 把这些向量写成一个矩阵X(X将是N\*M阶矩阵)。
3. 通过减去每列的平均值来把数据中心化，并令变化好的矩阵为B。
4. 计算协方差阵C= 1/N \*B^TB。
5. 计算C的特征向量和特征值，即V^-1CV=D,其中V由C的特征向量组成，D是由特征值组成的M\*M阶对角矩阵。
6. 把D对角线上元素按降序排列，并对V的列向量做同样的排列。
7. 去掉那些小于η的特征值，剩下L维的数据。

### 因子分析

观察数据是否可以被少量不相关的因素或潜在的变量解释，目的用于发现独立因素和测量每一个因素固有的误差。

### 独立成分分析（ICA）

应用前提：数据信号源是独立且数据非高斯分布（最多一个成分高斯分布），观测信号源的数目不少于源信号数目。

统计成分是独立的，即对于E[bi,bj] = E[bi]E[bj]与及bi是不相关的。

盲源中的不同信号源相互独立时，盲源分离才与独立成分分析等价，ICA是盲源分离的一种特例。

## 基于特征组合的降维

建设性方法：通过迭代不断加入，测试每一个阶段的错误以了解某个特征加入时是否会发生变化。破坏性方法是去掉应用在决策树上的特征。

## 基于核的PCA算法

1. 选择核并且把它应用于距离矩阵从而得到矩阵K。
2. 计算K的特征值和特征向量。
3. 通过特征值的平方根标准化特征向量。
4. 保留与最大特征值对应的特征向量。

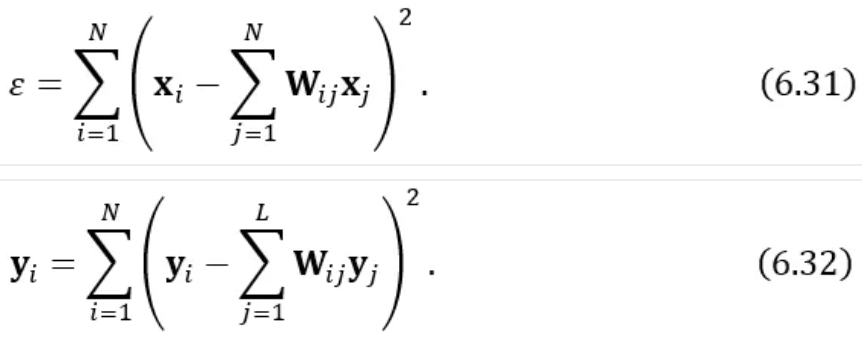
因为混合信号源很多时候是不独立的。

## 局部线性嵌入算法

找出每个点的邻近点(即前k个近的点):计算每对点间的距离。找到前k个小的距离。对于其他点，令Wij=0.对每个点xi:创建一个邻近点的位置表z,计算zi=zi-xi。

根据约束条件计算令等式（6.31）最小的权矩阵W:计算局部协方差C=ZZ^T，其中Z是zi组成的矩阵。利用CW=I计算W,其中I是N\*N单位矩阵。对于非邻近点，令Wij=0。

对W/∑W设置其他元素计算使得等式(6.32)最小的低维向量 yi:创建M=(I-W)T(I-W).计算M的特征值和特征向量。根据特征值的大小给特征向量排序。对应于第q小的特征值，将向量y的第q行设置为第q+1 个特征向量(忽略特征值为0)



## 多维标度算法

1. 计算由每对点平方相似度组成的矩阵D, Dij=|xi-xj|.计算J=IN – 1/N (IN是N\*N单位矩阵，N是数据点个数)。
2. 计算B=-1/2JDJ^T.
3. 找到B的L个最大的特征值入i,，以及相对应的特征向量ei。
4. 用特征值组成对角矩阵V并且用特征向量组成矩阵P的列向量。
5. 计算嵌入x=pv^0.5

## ISOMAP算法

1. 创建所有点对之间的距离
2. 确定每个点的邻近点，并做成一个权重表G
3. 通过找最短路径估计测地距离dG
4. 把经典MDS算法用于一系列dG

# 概率学习

## 期望最大算法（EM）

额外加入位置变量，通过这些变量最大化函数。

## 高斯混合模型的期望最大算法

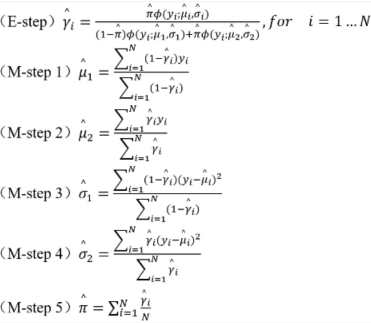
初始化

设置是从数据集中随机选出来的值

设置（这里是整个数据集的平均值）

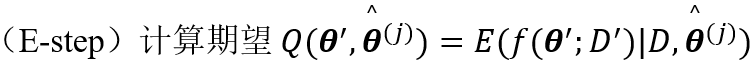
设置=0.5

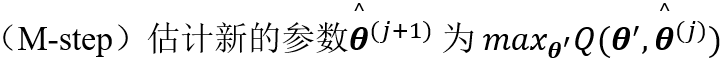
迭代直到收敛：



## 通常的期望最大化算法

1. 初始化
2. 猜测参数E:\Library\Document\WeChat Files\ziqi_forsaken\FileStorage\Temp\19d7b1cc10dc3fb207b537f2936124d5
3. 迭代直到收敛：





## 信息准则

除了通过模型选择确定停止学习的时间，前期采用验证集思想，而信息准则则是确定一些方法从而期待这个训练过的模型可以表现的多好。

* 艾卡信息准则：AIC = ln（C）-k
* 贝叶斯信息准则：BIC = 2ln（C）-klnN

K是模型中参数的数目，N是训练样本的数量，C是模型的最大似然。以上两种方法都是奥卡姆剃刀的一种形式。

## 奥卡姆剃刀

如无必要，勿增实体，即简单有效原理。

## 最近邻法

如果没有一个描述数据的模型，那么最好的事情就是观察相似的数据并且把他们选择成同一类。

## 核平滑法

用一个核（一堆点的权重函数）来根据输入的距离来决定每一个数据点有多少权重。当两个核都会对离当前输入更近的点给出更高的权重，而当他们离当前输入点越远时，权重会光滑的减少为0，权重通过λ来具体化。

## KD-Tree

在一个时刻选择一个维度并且将它分裂成两个，从而创建一颗二进制树，并且让一条直线通过这个维度里点的坐标的中位数。这与决策树的差别不大。数据点作为树的树叶。

制作树与通常的二进制树的方法基本相同：我们定义一个地方来分裂成两种选择——左边和右边， 然后沿着它们向下。可以很自然地想到用递归的方法来写算法。

选择在哪分裂和如何分裂使得KD-Tree是不同的。在每一步只有一个维度分裂，分裂的地方是通过计算那一维度的点的中位数得到的，并且在那画一条直线。通常，选择哪一个维度分裂要么通过不同的选择要么随机选择。

算法向下搜索可能的维度是基于到目前为止树的深度，所以在二维里，它要么是水平的要么是垂直的分裂。组成这个方法的核心是简单地选代选取分裂的函数，找到那个坐标的中位数的值，并且根据那个值来分裂点。

# 支持向量机

## 支持向量机（SVM）

当前现代机器学习中最流行的算法之一，其在大小合理的数据集上经常提供比其他机器学习算法更好的分类性能。

## 支持向量

在每个类中距离分类线最近的那些点则被称为支持向量。

如果有一个非线性可分数据集，那么就不能满足所有数据点的约束条件，解决方法是引入一些松弛变量η>=0。

## 选择核

任何一个对称函数如果是正定的，都可以用来做核。这就是Mercer定理的结果，Mercer定理也指出一些核旋绕到一起的结果可能是另一个核。

## 支持向量机算法

1. 初始化：

对于指定的内核和内核参数，计算数据之间距离的内核

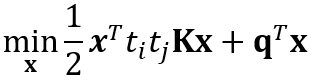
这里主要的工作是计算K=XX^T。

对于线性内核，返回K，对于多项式的次数d，返回1/σ 8 K^d。

对于RBF核，计算K=exp(-(x-x')^2/2σ\*σ。

1. 训练：

将约束集组装为要求解的矩阵：

约束于E:\Library\Document\WeChat Files\ziqi_forsaken\FileStorage\Temp\b115b0b4a901663eaddc17da1447aaa2

将这些矩阵传递给求解器。

将文持向量标识为距高最近点一定距离内的向量,并处理其余的训练集。

用公式(8.10)计算b^\*

## 分类

对于给定的测试数据z，使用支持向量对相关内核的数据进行分类，计算测试数据与支持向量的内积，进行分类，返回标记或值。

# 优化和搜索

## 下山法

朝哪个方向移动才能下降尽可能快：

* 采用线性搜索，知道方向，那么久沿着他一直走，直到最小值，这仅是直线的搜索；
* 信赖域，通过二次型建立函数的局部模型并且找到这个局部模型的最小值。由于我们不知道防线，因此可以采用贪婪选择法并且在每个点都沿着下降最快的方向走，这就是所知的最速下降，它意味着pk=-▽f（xk）。最速下降基于函数中的泰勒展开，这是一种根据导数近似函数值的方法。

## Lenenberg-Marquardt算法

给定一个初始点X0

当J^Tr(x)>公差并且没有超出最大迭代次数:

重复:

用线性最小二乘法算出(J^TJ+vI)dx=一J^Tr中的dx。令Xnew=x+dx。

计算实际减少和预测减少的比例:

实际=||f(x)- f(xnew)||

预测=▽f^T(x)\*xnew-x

p=实际/预测

如果0<p<0.25:

接受:x=Xnew。

或者如果p>0.25:

接受: x=Xnew。

增加信赖城大小(减少v)。

或者:

拒绝。

减少信赖域大小(增加v)。

一直到x被更新或超出跌代的最大次数

## 共轭梯度

二维空间中，如下图所示，一步沿x轴方向，另一部沿y方向，这样足以足以达到最小值。而在n维空间我们应该走n步以达到最小值，它尝试在线性情况下实现这个想法，但是我们通常感兴趣的非线性情况下，只需要多一点迭代就可以达到最小。

▲左边：如果方向之间是相互正交的并且步长是正确的，每一个维度只需要走一步，这里走了两步。右边：在椭圆上共轭的方向不是相互正交的。

具体算法：

给一个初始点x0和停止参数ε,令p0=-▽f(x)。

设置Pnew=P0

当Pnew>εεp0时:

用牛顿-拉夫森迭代法计算αkP

当ααdp>ε2时:

α=-(▽f(x)^T p)(p^T H(x)p)

x=x+αP

dp=P^TP

评价▽f(xnew)。

计算βn+1-更新p←▽f(xnew)+βk+1p。

检查及重新启动。

## 搜索的三种基本方法

### 穷举法

* 检查所有方法，保证找到全局最优

### 贪婪搜索

* 整个系统只找一条路，在每一步都找局部最优解。所以对于TSP,任意选择第-个城市，然后不断重复选择和当前所在城市最近并且没有访问过的城市，直到走完所有城市。它的计算量非常小，只有O(NlogN),但它并不保证能找到最优解，并且我们无法预测它找到的解决方案如何，有可能很糟糕。

### 爬山法

* 爬山法的基本想法是通过对当前解决方案的局部搜索，选择任一个选项来改善结果，进行局部搜索时做出的选择来自于一个移动集(moveset)。它描述了当前解决方案如何被改变从而用来产生新的解决方案。所以如果我们想象在2D欧几里得空间中移动，可能的移动就是东、南、西、北。

对于TSP，爬山法要先任意选一个解决方案， 然后调换其中一对城市的顺序，看看总的旅行距离是否减少。当交换的对数达到预先给定的数时，或找不到一个调换可以改善相对于预先给定的长度的结果时停止算法。

就像贪婪算法一样，我们无法预测结果将会怎样：有可能找到全局最优解，也有可能陷在第一个局部最大值上， 从而并不定能找到全局最优解，爬山法的核心循环仅仅是调换对城市， 并且仅当它使得距离变小时才保留调换。

## 模拟退火算法

开始时选择一个任意的很高的温度T，之后我们将随机选择状态并且改变它们的值，监视系统变化前后的能量。如果能量变低了，系统就会喜欢这种解决方法，所以我们接受这个变化。目前为止，这和梯度下降法很像。

然而，如果能量不变低，我们仍然考虑是否接受这个解决方法，并且接受的概率是exp((Ebefore- Eafter)/T)。这叫作波尔兹曼分布(Boltzmann distribution)。注意到Ebefore Eafter 是负的，所以我们定义的概率是合理的。偶尔接受这个不好的状态是因为我们可能找到的是局部最小，并且通过接受这个能量更多的状态，我们可以逃离出这个区域。

在重复上述方法几次后，我们采用一个退火时间表以便降低温度并且使得该方法能延续下去直到温度达到0。由于温度变低，所以接受任一个特殊的较高的能量状态的机会就会变少。最常用的退火时间表是T(l+1)=cT(t)，这里0<c<1(更加常用的是0.8<c<1)。需要减慢退火的速度以允许更多的搜索。

## 四种算法TSP结果比较

第一行为最好的解决方案和距离，第二行为运行时间，城市为10个。

Exhaustive search((1, 5, 10, 6, 3, 9, 2, 4, 8, 7, 0), 4.18)1781.0613Greedy search((3, 9, 2, 6, 10, 5, 1, 8, 4, 7, 0]), 4.49)0.0057Hill Climbing((7, 9, 6, 2, 4, 0, 3, 8, 1, 5, 10]), 7.00)0.4572Simulated Annealing((10, 1, 6, 9, 8, 0, 5, 2, 4, 7, 3]), 8.95)0.0065

# 进化学习

## 遗传算法（GA）

模拟进化是如何搜索的，通过改变基因来改变个体的适应度。

GA使用字符串（类似染色体的作用），字符串中的每个元素都是从某些字母表中选择，字母表中的值通常是二进制的相当于等位基因，对于解决方法，将被变为一个字符串，然后我们随机生产字符串作为初始种群。

评价适应度可以被看成一个预测，它作用于一个字符串并且返回一个值，它是遗传算法中唯一因具体问题不同而不同的部分。最好的字符串有最高的适应值，当解决方案不好时，适应度也随之下降，GA工作于由种群组成的字符串，然后评价每个字符串的适应度，并进行培养。

产生后代的常用3种方法：

### 联赛选择

* 反复从种群中挑选四个字符串，替换并将最适合的两个字符串放人交配池中。

### 截断选择

* 仅按比例f挑出适应度最好的一-部分并且忽略其他的。比如，f= 0.5经常被使用，所以前50%的字符串被放入交配池，并且被等可能地选择。这很显然是一个非常简单的实施方法，但是它限制了算法探索的数量，使得GA偏向于进化。

### 适应度比例选择

* 最好的方法是按概率选择字符串，每个字符串被选择的概率与它们的适应度成比例。通常采用的函数是(对于字符串a)：

这里F^α是适应度，如果适应度不是正值，则F需要在整个过程中被exp⁡(sF)替换，这里s是选择强度(selection strength)参数，并且你会意识到这个等式作为第4章的softmax激活）：

## 遗传算子产生

最常用时实现方法是在字符串中随机找一个点，在这个点之前的部分用字符串1的，而在交叉点之后，用字符串2的剩下部分。我们实际上产生了两个后代，第2个是由字符串2的第一部分和字符串1的第二部分组成的。这个方式称为单点交叉，显然，它的扩展是多点交叉。

最极端的情形是均匀交叉，它的字符串中的每一个元素都随机地选自与他的父母，下图展示了三中类型的交叉法：

▲交叉算子的不同形式。（a）单点交叉。随机选择字符串中的一个位置，然后用字符串1的第一部分和字符串2的第二部分组成后代。（b）多点交叉。选择多个点，后代的生成方式和前面一样。（c）均匀交叉。每个元素都随机的选自于它的父母。

对当前最好的字符串实现进化通过编译算子实现，字符串中每个元素的值以概率p（通常很小）改变。

## 3. 基本遗传算法

初始化

进过我们选的字母表产生N个长为L的字符事。

学习

生成一个(开始是空的)新的种群。

重复：

        通过适应度在当前种群中选择两个字符串。

        重组它们产生两个新的字符串。

        让后代变异。

        要么把两个后代加到种群中，要么使用精英法和比赛法

        保持记录种群中最好的字符串。

直到新种群中产生N个字符串

可选性地，使用精英法从父代中挑选最合适的字符串，并替换子代中的一些其他字符串。

追踪新种群中最好的字符串。

用新种群代替当前种群

直到到达停止标准。

## 4. 使用遗传算法进行图着色

把方案编码成字符串，选择合适的适应度函数，选择合适的遗传算子。

## 5. 与采样结合的进化学习

最基础的版本是我们熟知的基于种群的增长学习算法(PBIL).该算法非常简单，像基本的GA一样，它采用一个二进制字母表，但不保留种群，而是利用一个概率向来给出每一个元素是0或1的概率。

起初，向量的每一个值都是0.5，所以每一个元素有相等的机会变成0或1，之后通过从分布指定的向量中取样来构建群体,并计算群体中每个成员的适合度。

我们使用这个种群中的一个子集(通常只有前两个适应度最高的向量)和一个学习速率p来更新概率向量，学习速率通常被设置为0.005(这里best和second代表种群中最好的和第二好的成员):p= pX(1 - η)+ η(best十second)/2。

之后丢弃这些种群，并且利用更新的概率向量重新取样来产生新的种群，算法的核心就是简单地利用适应度最高的两个字符串和更多的向量来寻找新的字符串。

# 07 强化学习

强化学习是状态(state)或情形(situation)与动作(action)之间的映射，目的是最大化一些数值形式的奖赏(reward)。也就是说，算法知道当前的输人(状态)，以及它可能做的一些事(动作)，目的是最大化奖赏。进行学习的智能体和环境之间有着明显的区别，环境是智能体完成动作的地方，也是产生状态和奖赏的地方。

## 1. 马尔可夫决策过程

马尔可夫性：当前的状态对于计算奖赏提供了足够的信息而不需要以前的状态信息，一个具有马尔可夫性的强化学习成为马尔可夫决策过程。它意味着基于以前的经历，我们只需要知道当前的状态和动作就可以计算下一步奖赏的近似，以及下一步的状态。

## 2. 值

强化学习尝试决定选择哪一个动作来最大化未来的期望奖赏，这个期望奖赏就是值，可以考虑当前状态，对所有采取的动作进行平均，让策略来自己解决这个问题，即状态值函数，或者考虑当前状态和可能采取的动作即动作值函数。

## 3. O-learning算法

初始化

对于所有的s和a, 设置Q(s, a)为一个很小的随机數。

重复:

初始化s。

用目前的策略选择动作a。

重复：

        使用ε-greedy或者其他策略来选择动作a。

        采取动作a并得到奖赏r。

        采样新的状态s’

        更新Q(s, a)←Q(s, a)+u(r+γmaxa’ (s', a')-Q(s, a))

        设置s←s'

应用到当前情节的每一步。直到没有更多的情节。

## 4. Sarsa算法

初始化

对于所有的s和a,设置Q(s, a)为一个很小的随机数。

重复:

初始化s。

用当前策略选择动作

重复:

        实行动作a并得到奖赏r

        采样新的状态s'

        用当前策略选择动作a

        更新Q(s, a)<-Q(s, a)+u(r+yYQ(s',a')-Q(s,a)).

        s<-s’,a<-a’

应用到当前情节的每一步

直到没有更多的情节。

### 两种算法的相同

都是bootstrap方法，因为他们都是从对正确答案很少的估计开始，并且在算法进行过程中不断迭代。

### 不同

两个算法一开始都没有环境的任何信息，因此会利用ε-greedy策略随机探索。然而，随着时间的推移，两个算法所产生的决策出现了很大的不同。

产生不同的主要原因是Q-learning总是尝试跟着最优的路径，也就是最短的路，这使它离悬崖很近。并且，ε-greedy也意味着有时将会不可避免地翻倒。通过对比的方式，sarsa 算法将会收敛到一个非常安全的路线，它远离悬崖，即使走的路线很长。

sarsa 算法产生了一个非常安全的路线，因为在它的Q的估计中包含了关于动作选择的信息，而Q-learning生成了一条冒险但更短的路线。哪种路线更好由你决定，并且依赖于跌落悬崖的后果有多么严重。

# 08 树的学习

决策树的主要思想是从树根开始，把分类任务按顺序分类成一个选择，一步步进行到叶子节点最终得到分类的结果，树结构可以表示成if-then规则的集合，适合应用于规则归纳系统。

## 1. ID3

在决策树下一次分类是，对于每一个特征，通过计算真个训练集的熵减少来选择特征，这成为信息增益，描述为整个集合的熵减去每一个特定特征被选择后的熵减去每一个特定特征被选中后的熵。

### 具体算法

如果所有的样本都具有同一标记:返回标记为该类标记的叶子节点。

否则，如果没有剩余特征用于测试:返回标记为最常见标记的叶子节点，

否则:使用公式选择S中具有最大信息增益的特征户作为下一个节点。为每一个特征户的可能取值f增加一个分支。对于每个分支:计算除去F后的每一个特征的Sf，使用Sf递归调用算法，计算目前样本集合的信息增益。

### 决策树复杂度

假设树是近似平衡的，那么每个节点的成本包括搜索d个可能的特征（尽管每个层级减少1，但这不会影响O(·)符号的复杂性），然后计算每个分裂的数据集的信息赠与，这需要花费O（dnlogn），其中n为该节点上数据及的大小，对于根节点，n=N，并且如果树是平衡的，则在树的每个阶段将n除于2。在树种的大约logN层级上对此求和，得到计算成本O(dN^2logN)。

# 09 委员会决策：集成学习

下图为集成学习的基本思想，给定一个相对简单的二类分类问题和一些学习器，一个学习器的椭圆覆盖数据的一个子集，组合多个椭圆给出相当复杂的决策边界。

▲通过组合许多简单的分类器（这里简单地将椭圆决策边界放在数据上），决策边界可以变得更加复杂，使得正例与圆圈难以分离。

## 1. AdaBoost（自适应提升）

每次迭代中，一个新的分类器在训练集上训练，而训集中的每-个数据点在每一步迭代时都会调整权重，改变权重的根据是数据点被之前的分类器成功分类的难度。一开始， 这些权重都被初始化为1/N,其中N是训练集中点的个数然后，每次迭代时，用所有被错分的点的权重之和作为误差函数ε。

对于错误分类的点，其权重更新乘子为a=(1-ε)/ ε。对于正确分类的点，其权重不变。接着在整个数层集上做归一化(这是降低被正确分类的数据点的重要性的有效方法)。在设定的迭代次数结束之后训练终止，或者当所有的数据点都被正确分类后训练终止，或者一个点的权重大于最大可用权重的一半时训练也终止。

**具体算法：**

初始化所有的权值为1/N，其中N为数据点的个数

当(t<T,最大迭代次数)：

在上训练分类器，得到数据点的假设

计算训练误差

设置

使用如下公式更新权值：

其中Zn为标准化常量

返回标记为最普通类标的叶子节点

## 2. 随机森林

如果一棵树是好的，那么许多树木应该更好，只要他们有足够的变化。

## 3. 基本的随机森林训练算法

对于每N个树:

创建一个训练集的bootstrap样本。

使用这个bootstrap样本训练决策树。

在决策树的每一个节点，随机选择m个特征，然后只在这些特征集合中计算信息增益(或者基尼不纯度)，选择最优的一个。

重复过程直到决策树完成。

## 4. 专家混合算法

对于每一个专家：

计算属于每一个可能的类别的输入的概率，通过如下公式计算（其中w\_i是对于每个分类器的权重）：

对于树上的每个门控网络：

计算：

传递一个输入到下一层门（这里的和是和该门相关的输入上的和）：

# 10 无监督学习

无监督学习在不知道数据点属于这一类而那些数据点属于另一类的情况下找到数据中相似输入的簇。

## 1. k-means算法

初始化

选择一个k值。

在输入空间中选择k个随机位置。

将簇中心μj,安排到这些位置。

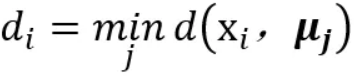
学习

重复:

对每一个数据点Xi:

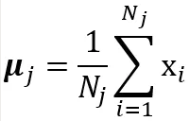
计算到每一个簇中心的距离。

用下面的距离将数据点安排到最近的簇中心。



对每个簇中心:

将中心的位置移到这个簇中点的均值处(Nj是簇j中点的个数):



直到簇中心停止移动

使用

对每个测试点:

计算到每个簇中心的距离。

用下面的距离将数据点安排到最近的簇中心

**2. 在线k-Means算法**

初始化选择一个值k,它与输出节点的数目有关。用小的随机值来初始化权重。学习归一化数据以便所有的点都在单位球上。重复:        对每一个数据点:        计算所有节点的激活。        选出激活最高的那个节点作为胜利者。        用公式更新权重。直到迭代的次数超过阈值。使用对于每个测试点：        计算所有节点的激活        选择激活最高的节点作为胜利者

**3. 自组织特征映射算法**

初始化  
选择大小（神经元数目）和映射的维度d或者        随机选择权重向量的值使得它们都是不同的OR        设置权值来增加数据的前d个主成分的方向学习重复        对每一个数据点：            用权重和输入间的欧氏距离的最小值来选择最匹配的神经元，        用下面的公式来更新最匹配节点的权重向量：        这里η(t)是学习效率        其他的神经元用下面的公式更新权重向量：        这里是邻居节点的学习效率，而是邻居函数，它决定是否每个神经元应该是胜利神经元的邻居（所以h=1是邻居，h=0不是邻居）        减少学习效率并且调整邻居函数，一般通过，这里0≤α≤1决定大小下降的速度，k是算法已经运行的迭代次数，k\_max是算法停止的迭代次数。相同的公式被用于学习效率(η,ηn)和邻居函数直到映射停止改变或超出了最大迭代的次数使用对每个测试点：        用权重和输入间的欧氏距离的最小值来选择最匹配的神经元n\_b：