机器学习算法地图

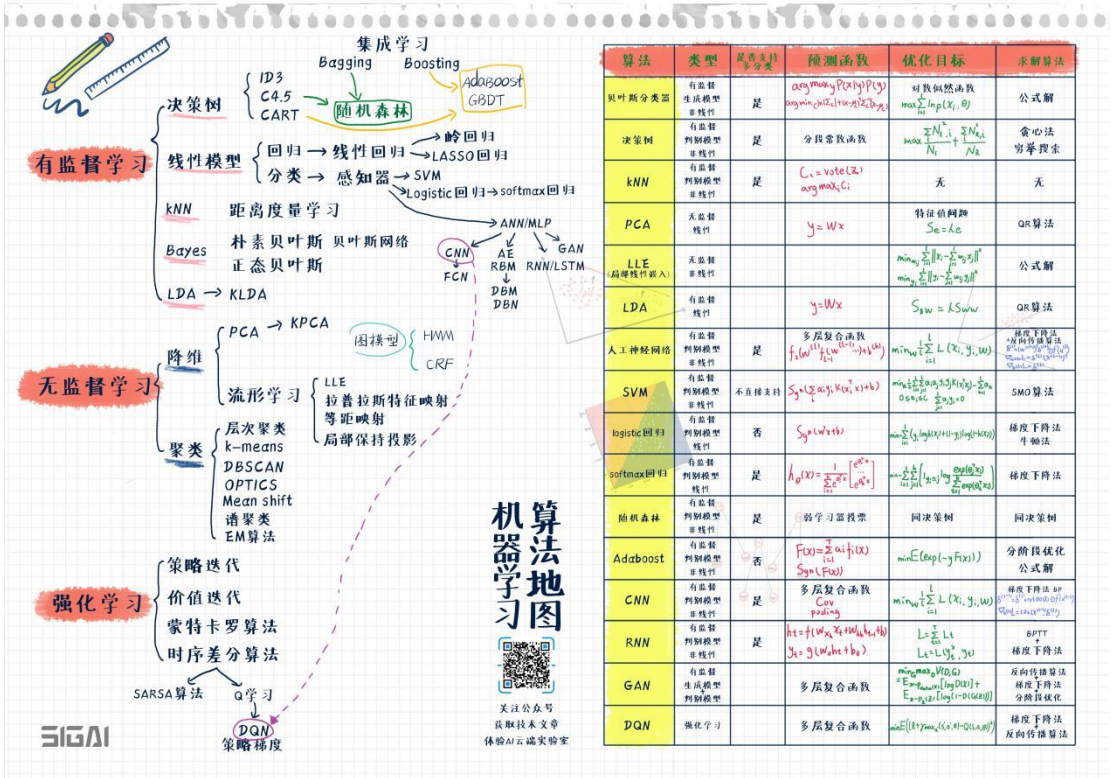
作者：SIGAI

2018．7．04

很多同学在学机器学习和深度学习的时候都有一个感受：所学的知识零散、不系统，缺乏整体感，这是普遍存在的一个问题。在这里，SIGAI 对常用的机器学习和深度学习算法进行了总结，整理出它们之间的关系，以及每种算法的核心点，各种算法之间的比较。由此形成了一张算法地图，以帮助大家更好的理解和记忆这些算法。

如果你对这张图感兴趣，可以关注 SIGAI 公众号，给公众号发消息，得到电子版的下载地址，用作电脑桌面是非常不错的，绝对有逼格！我们把这张图用精美的纸打印出来了，如果你想要纸质版的，也可以给我们的公众号发消息，我们会用快递发送给你（快递费自付），贴在墙上也是不错的！

下面先看这张图：



算法	类型	是否支持多分类	预测函数	优化目标	求解算法
贝叶斯分类器	有监督 生成模型 非线性	是	$\arg\max_y P(x\|y)P(y)$	对数似然函数 $\max\sum_{i=1}^{N}\ln p(x_i,\theta)$	公式解
决策树	有监督 判别模型 非线性	是	分段常数函数	$\max\frac{\sum N_{1,i}^2}{N_1}+\frac{\sum N_{2,i}^2}{N_2}$	贪心法 穷举搜索
kNN	有监督 判别模型 非线性	是	$C_x=\text{vote}(x)$ $\arg\max_x C_i$	无	无
PCA	无监督 线性		$y=Wx$	特征值阈值 $S_e=\lambda e$	QR算法
LLE（局部线性嵌入）	无监督 非线性			$\min_{w_{ij}}\sum_i\|x_i-\sum_j w_{ij}x_j\|^2$ $\min_{y_i}\sum_i\|y_i-\sum_j w_{ij}y_j\|^2$	公式解
LDA	有监督 线性		$y=Wx$	$S_{bw}=\lambda S_w w$	QR算法
人工神经网络	有监督 判别模型 非线性	是	多层复合函数 $f_3(w^{(3)}f_2(w^{(2)}f_1(w^{(1)}x+b^{(1)})+b^{(2)})+b^{(3)})$	$\min_w\frac{1}{2}\sum_{i=1}^{L}L(x_i,y_i,w)$	梯度下降法 +反向传播算法
SVM	有监督 判别模型 非线性	不直接支持	$\text{Sgn}(\sum_i^t a_i y_i K(x_i^T x)+b)$	$\min_w\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j K(x_i^T x_j)-\sum_{i=1}^{n}\alpha_i$ $0\le\alpha_i\le C$ $\sum_{i=1}^{n}\alpha_i y_i=0$	SMO算法
logistic回归	有监督 判别模型 线性	否	$\text{Sgn}(w^T x+b)$	$\min_w\sum_{i=1}^{n}(y_i\log h(x)+(1-y_i)\log(1-h(x)))$	梯度下降法 牛顿法
softmax回归	有监督 判别模型 线性	是	$h_\theta(x)=\frac{1}{\sum_{j=1}^{k}e^{\theta_j^T x}}\begin{bmatrix}e^{\theta_1^T x}\\ \vdots\\ e^{\theta_k^T x}\end{bmatrix}$	$\min_w-\frac{1}{m}\sum_{i=1}^{m}\sum_{j=1}^{k}\left[I_{\{y_i=j\}}\log\frac{\exp(\theta_j^T x_i)}{\sum_{l=1}^{k}\exp(\theta_l^T x_i)}\right]$	梯度下降法
随机森林	有监督 判别模型 非线性	是	弱学习器投票	同决策树	同决策树
Adaboost	有监督 判别模型 非线性	否	$F(x)=\sum_{i=1}^{T}a_i f_i(x)$ $\text{Sgn}(F(x))$	$\min E(\exp(-yF(x)))$	分阶段优化 公式解
CNN	有监督 判别模型 非线性	是	多层复合函数 Conv pooling	$\min_w\frac{1}{2}\sum_{i=1}^{L}L(x_i,y_i,w)$	梯度下降法 BP
RNN	有监督 判别模型 非线性	是	$h_t=f(W_{xh}x_t+W_{hh}h_{t-1}+b)$ $y_t=g(W_{ho}h_t+b_o)$	$L=\sum_{t=1}^{T}L_t$ $L_t=L(y_t',y_t)$	BPTT 梯度下降法
GAN	有监督 生成模型 判别模型		多层复合函数	$\min_G\max_D V(D,G)$ $=E_{x\sim p_{data}(x)}[\log D(x)]+$ $E_{z\sim p_z(z)}[\log(1-D(G(z)))]$	反向传播算法 梯度下降法 分阶段优化
DQN	强化学习		多层复合函数	$\min E[(R+\gamma\max_{a'}Q(s',a',\theta')-Q(s,a,\theta))^2]$	梯度下降法 反向传播算法

图的左半部分列出了常用的机器学习算法与它们之间的演化关系，分为有监督学习，无监督学习，强化学习 3 大类。右半部分列出了典型算法的总结比较，包括算法的核心点如类型，预测函数，求解的目标函数，求解算法。

理解和记忆这张图，对你系统化的掌握机器学习与深度学习会非常有帮助！

我们知道，整个机器学习算法可以分为有监督学习，无监督学习，强化学习 3 大类。除此之外还有半监督学习，但我们可以把它归到有监督学习中。算法的演变与发展大多在各个类的内部进行，但也可能会出现大类间的交叉，如深度强化学习就是深度神经网络与强化学习技术的结合。

根据样本数据是否带有标签值（label），可以将机器学习算法分成有监督学习和无监督学习两类。如果要识别 26 个英文字母图像，我们要将每张图像和它是哪个字符即其所属的类型对应起来，这个类型就是标签值。

有监督学习（supervised learning）的样本数据带有标签值，它从训练样本中学习得到一个模型，然后用这个模型对新的样本进行预测推断。它的样本由输入值 x 与标签值 y 组成：

$$(\mathrm{x}, y)$$

其中 x 为样本的特征向量，是模型的输入值；y 为标签值，是模型的输出值。标签值可以是整数也可以是实数，还可以是向量。有监督学习的目标是给定训练样本集，根据它确定映射函数：

$$y = f(\mathrm{x})$$

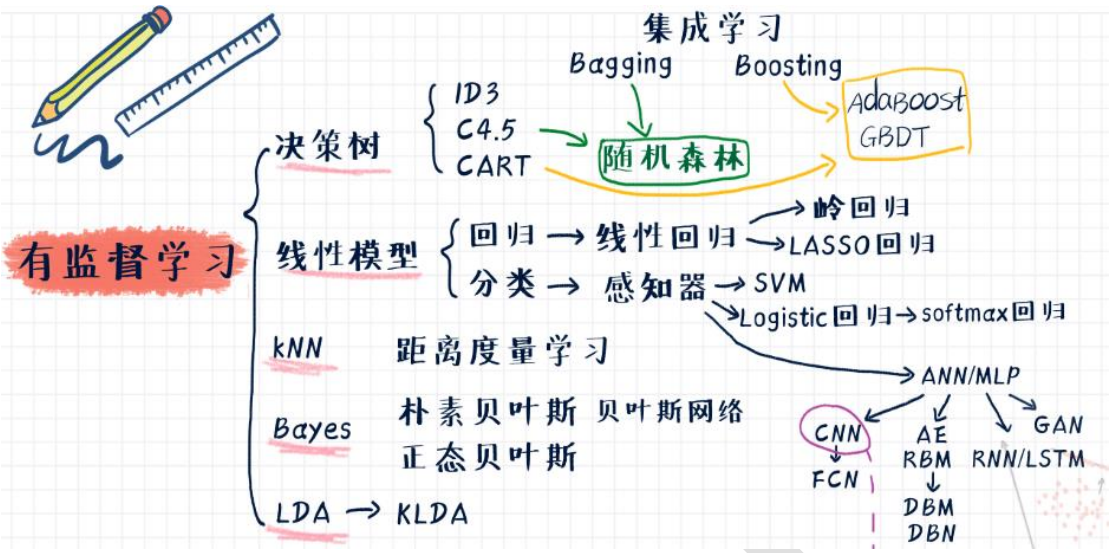
确定这个函数的依据是函数能够很好的解释训练样本，让函数输出值 f(x)与样本真实标签值 y 之间的误差最小化，或者让训练样本集的对数似然函数最大化。这里的训练样本数是有限的，而样本所有可能的取值集合在很多情况下是一个无限集，因此只能从中选取一部分样本参与训练。

日常生活中的很多机器学习应用，如垃圾邮件分类，手写文字识别，人脸识别，语音识别等都是有监督学习。这类问题需要先收集训练样本，对样本进行进行标注，用标注好的训练样本训模型，然后根据模型对新的样本进行预测。

无监督学习（unsupervised learning）对没有标签的样本进行分析，发现样本集的结构或者分布规律。无监督学习的典型代表是聚类和数据降维。

强化学习是一类特殊的机器学习算法，它根据输入数据确定要执行的动作，在这里。输入数据是环境参数。和有监督学习算法类似，这里也有训练过程中。在训练时，对于正确的动作做出奖励，对错误的动作做出惩罚，训练完成之后就用得到的模型进行预测。

在有监督学习算法中，我们列出了 5 个分支：



分别是决策树，贝叶斯，线性模型，kNN，LDA（线性判别分析），集成学习。LDA 也可以归类到线性模型中，但因为它是一种有监督的投影技术，我们单独列出。

决策树是一种基于规则的方法，它的规则是通过训练样本学习得到的，典型的代表是 ID3，C4.5，以及分类与回归树。

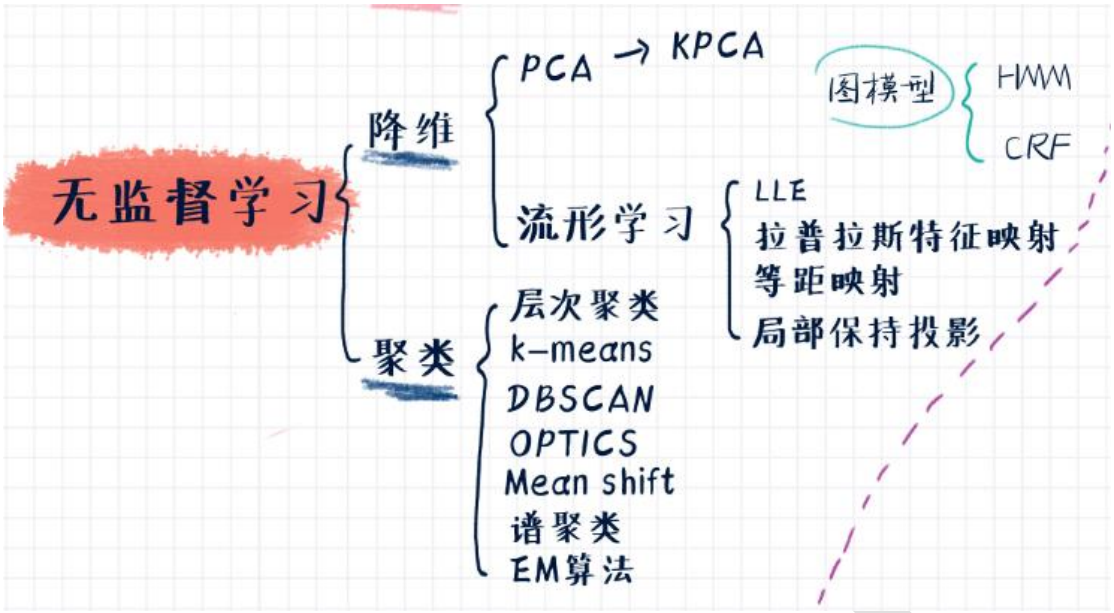
集成学习是机器学习中一类重要的算法，它通过将多个简单的模型进行集成，得到一个更强大的模型，简单的模型称为弱学习器。决策树与集成学习算法相结合，诞生了随机森林，Boosting 这两类算法（事实上，Boosting 算法的弱学习器不仅可以用决策树，还可以用其他算法）。

线性模型是最大的一个分支，从它最后衍生除了一些复杂的非线性模型。如果用于分类问题，最简单的线性模型是线性回归，加上 L2 和 L1 正则化项之后，分别得到岭回归和 LASSO 回归。对于分类问题，最简单的是感知器模型，从它衍生出了支持向量机，logistic 回归，神经网络 3 大分支。而神经网络又衍生出了各种不同的结构。包括自动编码器，受限玻尔兹曼机，卷积神经网络，循环神经网络，生成对抗网络等。当然，还有其他一些类型的神经网络，因为使用很少，所以在这里不列出。

kNN 算法基于模板匹配的思想，是最简单的一种机器学习算法，它依赖于距离定义，而距离同样可以由机器学习而得到，这就是距离度量学习。

贝叶斯也是有监督学习算法中的一个大分支，最简单的是贝叶斯分类器，更复杂的有贝叶斯网络。而贝叶斯分类器又有朴素贝叶斯和正态贝叶斯两种实现。

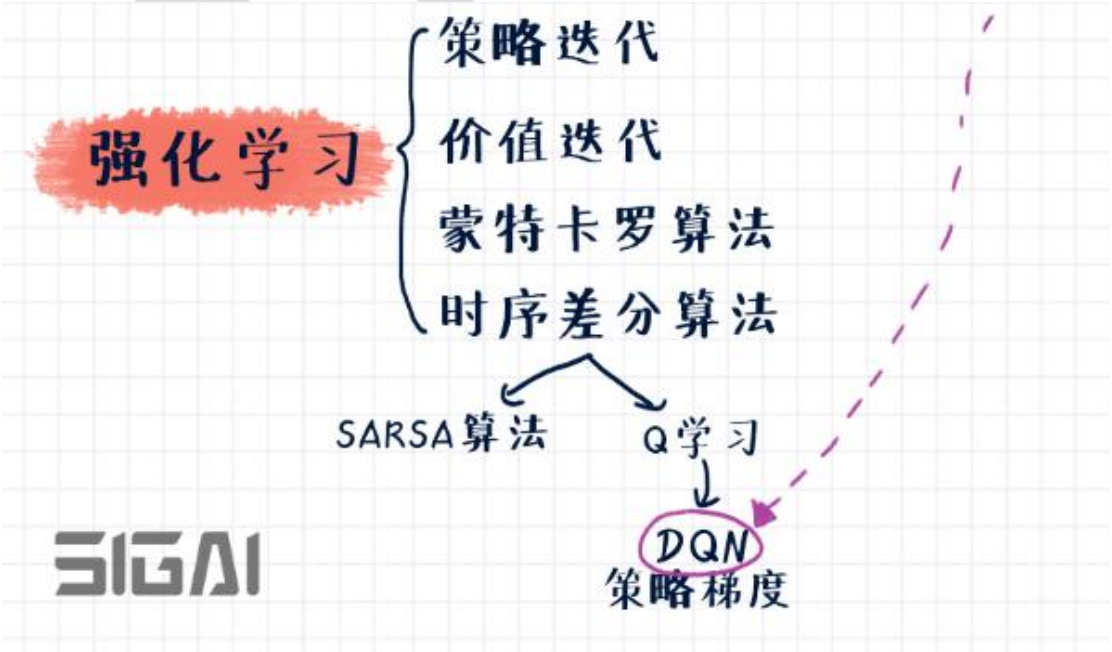
接下来说无监督学习，它可以分为数据降维算法和聚类算法两大类。演变关系如下图所示：



无监督的降维算法可以分为线性降维和非线性降维两大类。前者的典型代表是主成分分析（PCA），通过使用核技术，可以把它扩展为非线性的版本。流形学习是非线性降维技术的典型实现，代表性的算法有局部线性嵌入（LLE），拉普拉斯特征映射，等距映射，局部保持投影，它们都基于流形假设。流形假设不仅在降维算法中有用，在半监督学习、聚类算法中同样有使用。

聚类算法可以分为层次距离，基于质心的聚类，基于概率分布的距离，基于密度的聚类，基于图的聚类这几种类型。它们从不同的角度定义簇（cluster）。基于质心的聚类典型代表是 k 均值算法。基于概率分布的聚类典型代表是 EM 算法。基于密度的聚类典型代表是 DBSCAN 算法，OPTICS 算法，Mean shift 算法。基于图的聚类典型代表是谱聚类算法。

强化学习是机器学习中的一个特殊分支，用于决策、控制问题。这类算法的演变关系如下图所示：



整个强化学习的理论模型可以抽象成马尔可夫决策过程。核心任务是求解使得回报最大

的策略。如果直接用动态规划求解，则有策略迭代和价值迭代两类算法。他们都要求有精确的环境模型，即状态转移概率和奖励函数。如果做不到这一点，只能采用随机算法，典型的代表是蒙特卡罗算法和时序差分算法。强化学习与深度学习相结合，诞生了深度强化学习算法，典型代表是深度 Q 网络（DQN）以及策略梯度算法（策略梯度算法不仅可用神经网络作为策略函数的近似，还可以用其他函数）。

下面我们来分别介绍每种算法的核心知识点以及它们之间的关系。

有监督学习

先看有监督学习算法，它是当前实际应用中使用最广的机器学习算法。进一步可以分为分类问题与回归问题两大类。前面说过，有监督学习算法的预测函数为：

$$y = f\left(\mathrm{x}\right)$$

即根据输入数据 x 预测出输出数据 y。如果 y 是整数的类别编号，则称为分类问题；如果 y 是实数值，则为回归问题。

贝叶斯分类器

分类问题中样本的特征向量取值 x 与样本所属类型 y 具有因果关系。因为样本属于类型 y，所以具有特征值 x。分类器要做的则相反，是在已知样本的特征向量为 x 的条件下反推样本所属的类别 y。根据贝叶斯公式有：

$$p\left(y|\mathrm{x}\right) = \frac{p\left(\mathrm{x}|y\right)p\left(y\right)}{p\left(\mathrm{x}\right)}$$

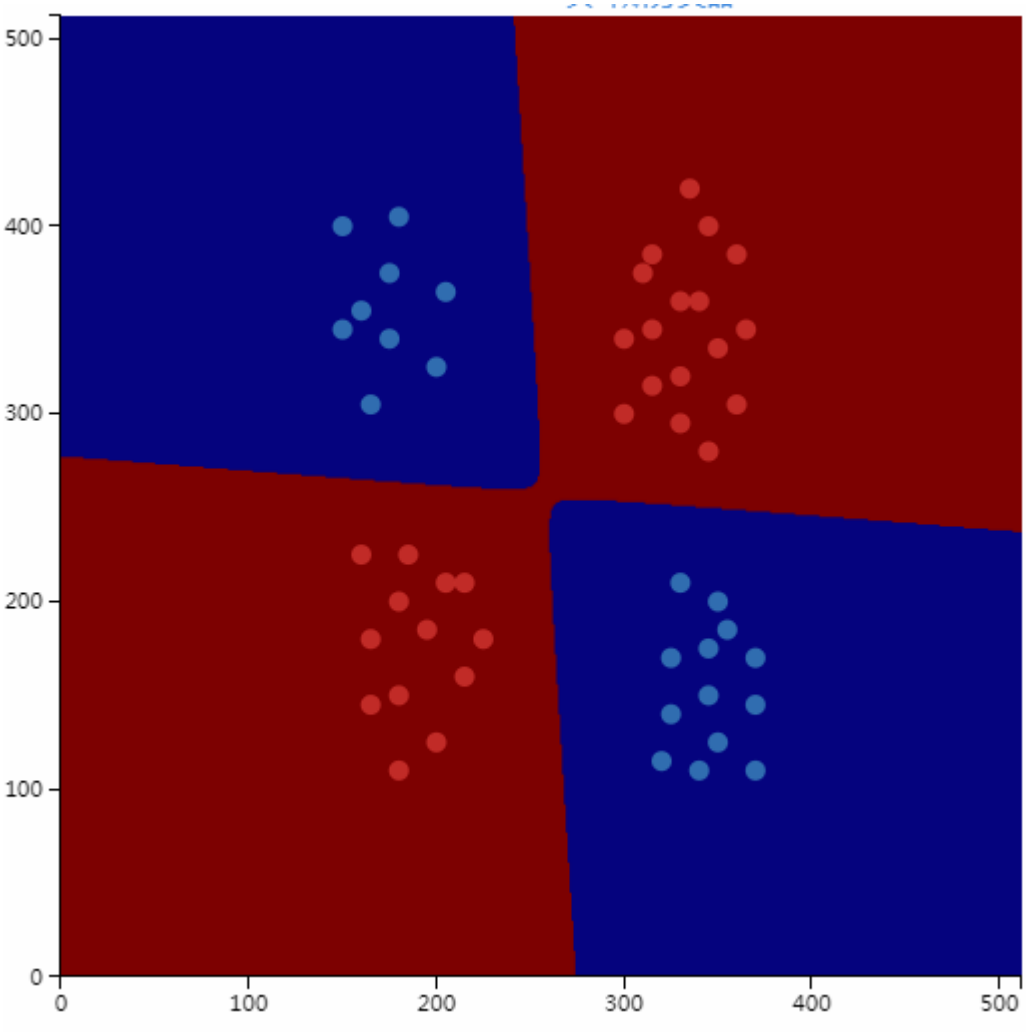
只要知道特征向量的概率分布 p(x)，每一类出现的概率 p(y)，以及每一类样本的条件概率 p(x|y)，就可以计算出样本属于每一类的概率 p(y|x)。如果只要确定类别，比较样本属于每一类的概率的大小，找出该值最大的那一类即可。因此可以忽略 p(x)，因为它对所有类都是一样的。简化后分类器的判别函数为：

$$\arg\max_y p\left(\mathrm{x}|y\right)p\left(y\right)$$

训练时的目标是确定 p(x|y)的参数，一般使用最大似然估计。如果假设样本特征向量的各个分量之间相互独立，则称为朴素贝叶斯分类器。如果假设特征向量 x 服从多维正态分布，则称为正态贝叶斯分类器。正态贝叶斯分类器的预测函数为：

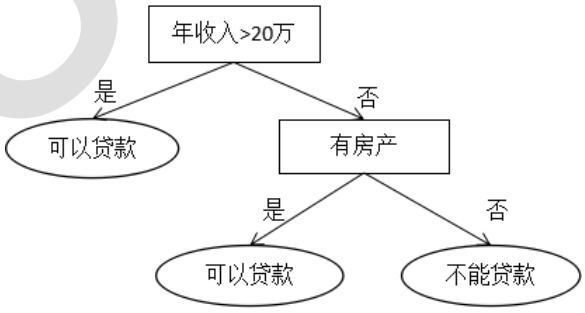
$$\min_c \ln\left(\left|\Sigma\right|_c\right) + \left(\left(\mathrm{x}-\mu_c\right)^{\mathrm{T}}\Sigma_c^{-1}\left(\mathrm{x}-\mu_c\right)\right)$$

贝叶斯分类器是一种生成模型，是非线性模型，它天然的支持多分类问题。下图是正态贝叶斯分类器对异或问题的分类结果（来自 SIGAI 云端实验室）：



决策树家族

决策树是基于规则的方法，它用一组嵌套的规则进行预测，在树的每个决策节点处，根据判断结果进入一个分支，反复执行这种操作直到到达叶子节点，得到决策结果。决策树的这些规则通过训练得到，而不是人工制定的。下图是决策树的一个例子：

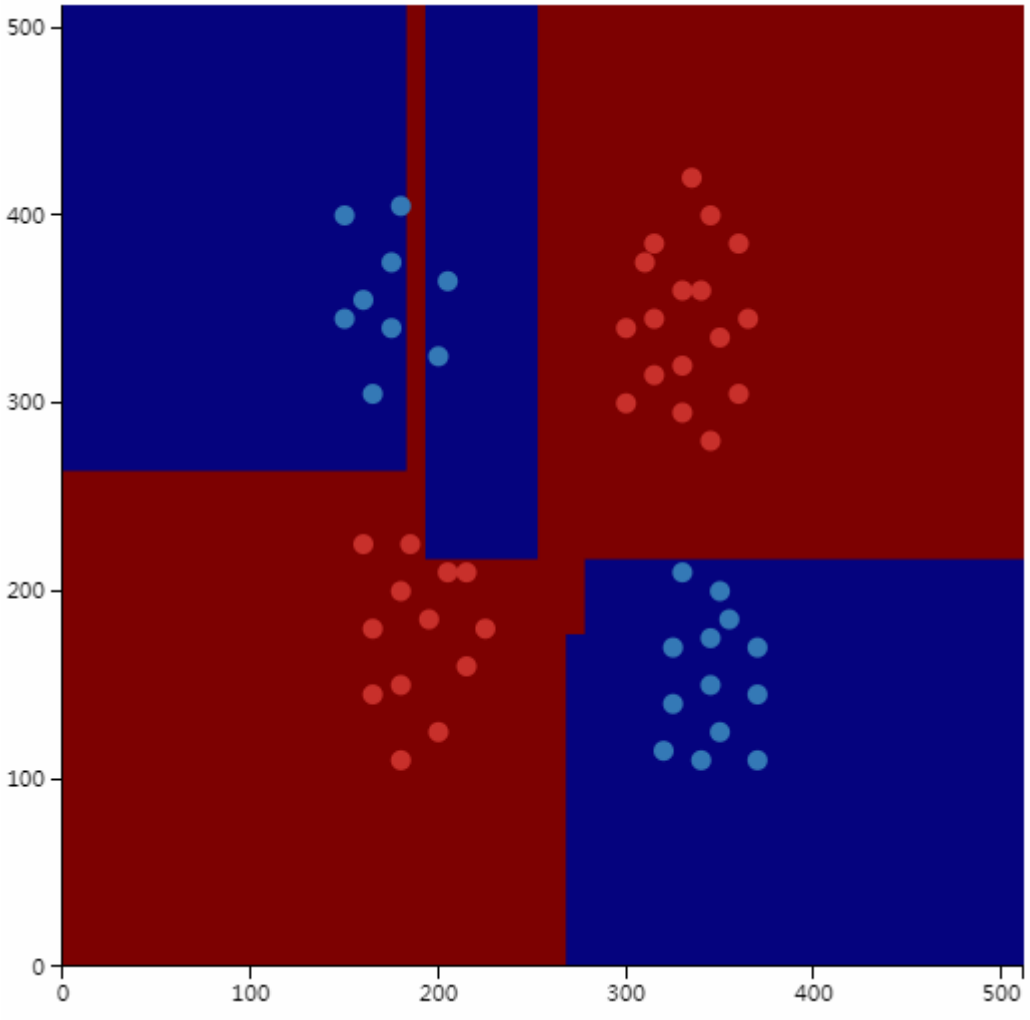


决策树是一种判别模型，也是非线性模型，天然支持多类分类问题。它既可以用于分类问题，也可以用于回归问题，具有很好的解释性，符合人类的思维习惯。常用的决策树有 ID3，C4.5，分类与回归树（CART）等。

分类树对应的映射函数是多维空间的分段线性划分，即用平行于各个坐标轴的超平面对空间进行切分；回归树的映射函数是一个分段常数函数。决策树是分段线性函数但不是线性函数，它具有非线性建模的能力。只要划分的足够细，分段常数函数可以逼近闭区间上任意

函数到任意指定精度，因此决策树在理论上可以对任意复杂度的数据进行分类或者回归。

下图是决策树进行空间划分的一个例子。在这里有红色和蓝色两类训练样本，用下面两条平行于坐标轴的直线可以将这两类样本分开（来自 SIGAI 云端实验室）：



这个划分方案对应的决策树如下图所示：

对于分类与回归树，训练每个节点时的目标是要让 Gini 不纯度最小化，这等价于让下面的值最大化：

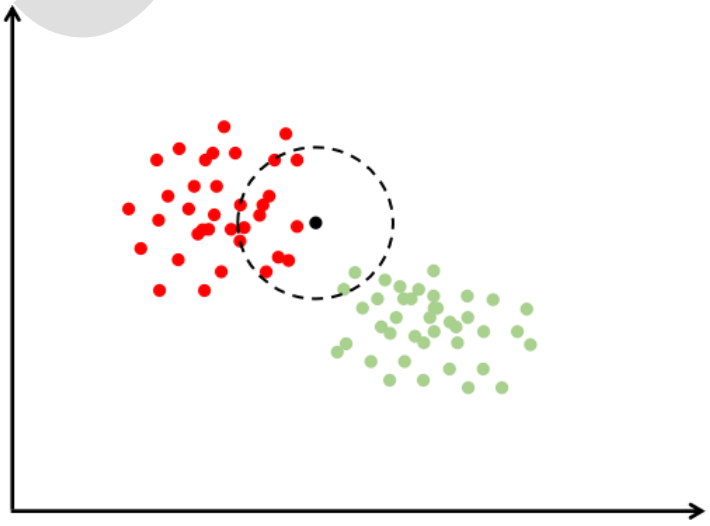
$$G = \frac{\sum_i N_{L,i}^2}{N_L} + \frac{\sum_i N_{R,i}^2}{N_R}$$

决策树训练求解时采用了枚举搜索和贪婪法的思想，找到的不一定是结构最优的树。如果想要了解决策树的原理，请阅读 SIGAI 之前的公众号文章“理解决策树”。

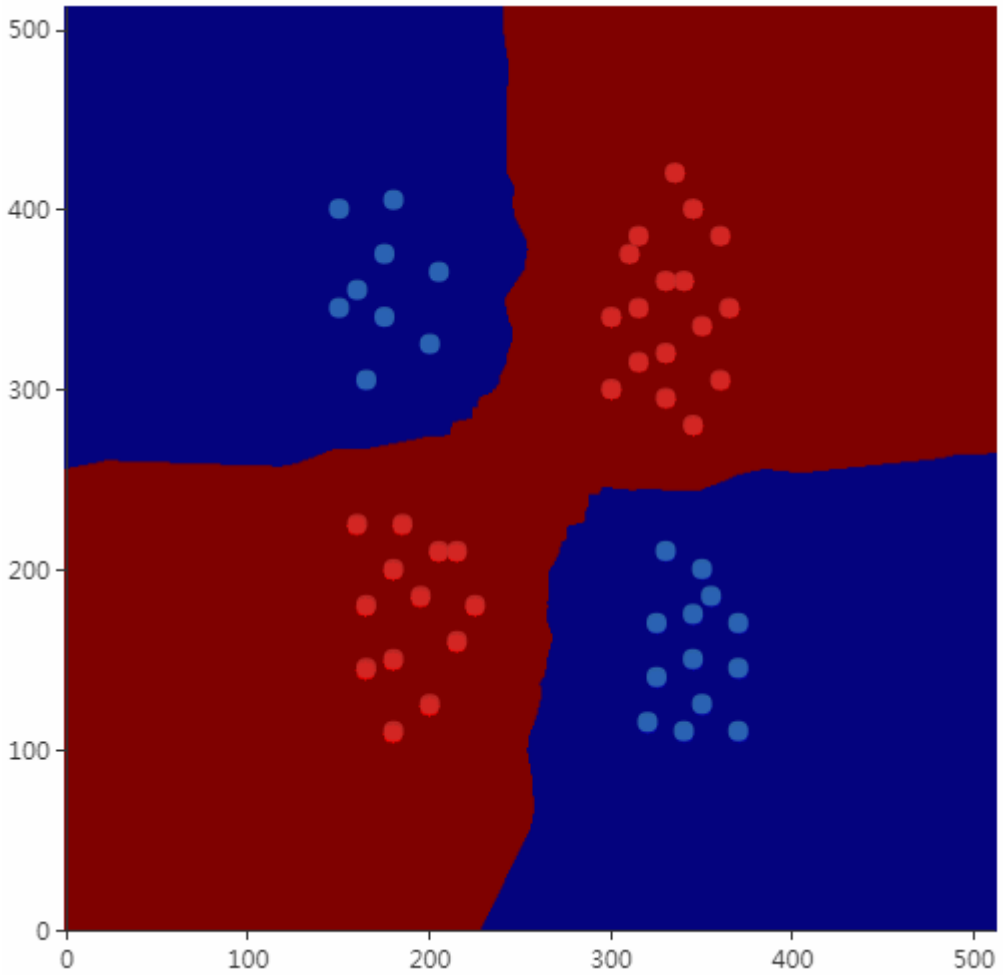
kNN 算法

kNN 算法基于以下思想：要确定一个样本的类别，可以计算它与所有训练样本的距离，然后找出和该样本最接近的 k 个样本，统计这些样本的类别进行投票，票数最多的那个类就是分类结果。因为直接比较样本和训练样本的距离，kNN 算法也被称为基于实例的算法，这实际上是一种模板匹配的思想。

下图是使用 k 近邻思想进行分类的一个例子：



在上图中有红色和绿色两类样本。对于待分类样本即图中的黑色点，我们寻找离该样本最近的一部分训练样本，在图中是以这个矩形样本为圆心的某一圆范围内的所有样本。然后统计这些样本所属的类别，在这里红色点有 12 个，圆形有 2 个，因此把这个样本判定为红色这一类。上面的例子是二分类的情况，我们可以推广到多类，k 近邻算法天然支持多类分类问题，它是一种判别模型，也是非线性模型。下图是 kNN 算法对异或问题的分类结果（来自 SIGAI 云端实验室）：



kNN 算法依赖于样本距离值，常用的距离有欧氏距离，Mahalanobis 距离等。这些距离定义中的参数可以通过学习得到，如 Mahalanobis 距离中的矩阵 S，这称为距离度量学习。

线性模型家族

线性模型的预测函数是线性函数，既可以用于分类问题，也可以用于回归问题，这是机器学习算法中的一个庞大家族。从线性模型中衍生出了多种机器学习算法，对于分类问题，有岭回归，LASSO 回归；对于分类问题，有支持向量机，logistic 回归，softmax 回归，人工神经网络（多层感知器模型），以及后续的各种深度神经网络。

对于分类问题，线性模型的预测函数为：

$$\mathrm{sgn}\left(\mathrm{w}^{\mathrm{T}}\mathrm{x}+b\right)$$

其中 sgn 是符号函数。最简单的线性分类器是感知器算法，它甚至无法解决经典的异或

问题，不具有太多的实用价值。

对于回归问题，线性模型的预测函数为：

$$\mathrm{w}^{\mathrm{T}}\mathrm{x}+b$$

训练时的目标是最小化均方误差：

$$\min\frac{1}{2l}\sum_{i=1}^{l}\left(\mathrm{w}^{\mathrm{T}}\mathrm{x}_i-y_i\right)^2$$

可以证明，这是一个凸优化问题，可以得到全局极小值。求解时可以采用梯度下降法或者牛顿法。

岭回归是线性回归的 L2 正则化版本，训练时求解的问题为：

$$\min_{\mathrm{w}}\frac{1}{2l}\sum_{i=1}^{l}\left(\mathrm{w}^{\mathrm{T}}\mathrm{x}_i-y_i\right)^2+\lambda\left\|\mathrm{w}\right\|_2^2$$

如果系数 $\lambda>0$，这个问题是一个严格凸优化问题，可用用梯度下降法，牛顿法求解。

LASSO 回归是线性回归的 L1 正则化版本，训练时求解的问题为：

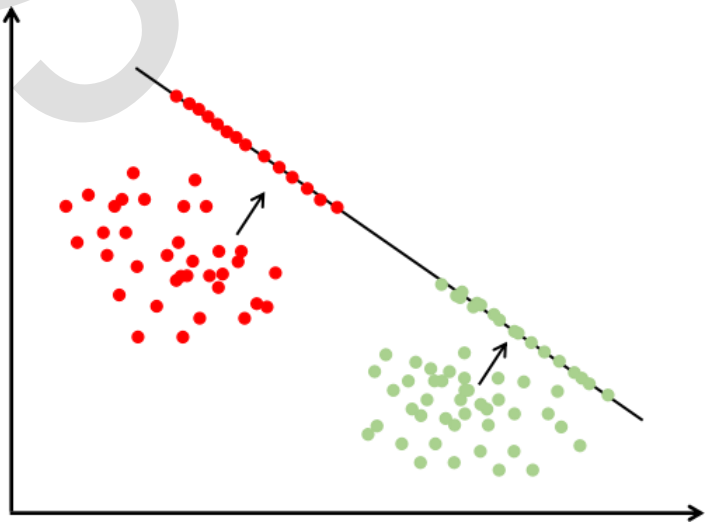
$$\min_{\mathrm{w}}\frac{1}{2l}\sum_{i=1}^{l}\left(\mathrm{w}^{\mathrm{T}}\mathrm{x}_i-y_i\right)^2+\lambda\left\|\mathrm{w}\right\|_1$$

同样的，这是一个凸优化问题，可以用梯度下降法和牛顿法求解。

线性判别分析（LDA）是一种有监督的线性投影技术，它寻找向低维空间的投影矩阵 W，样本的特征向量 x 经过投影之后得到的新向量 y：

$$\mathrm{y}=\mathrm{Wx}$$

投影的目标是同一类样投影后的结果向量差异尽可能小，不同类的样本差异尽可能大。直观来看，就是经过这个投影之后同一类的样本进来聚集在一起，不同类的样本尽可能离得远。下图是这种投影的示意图：



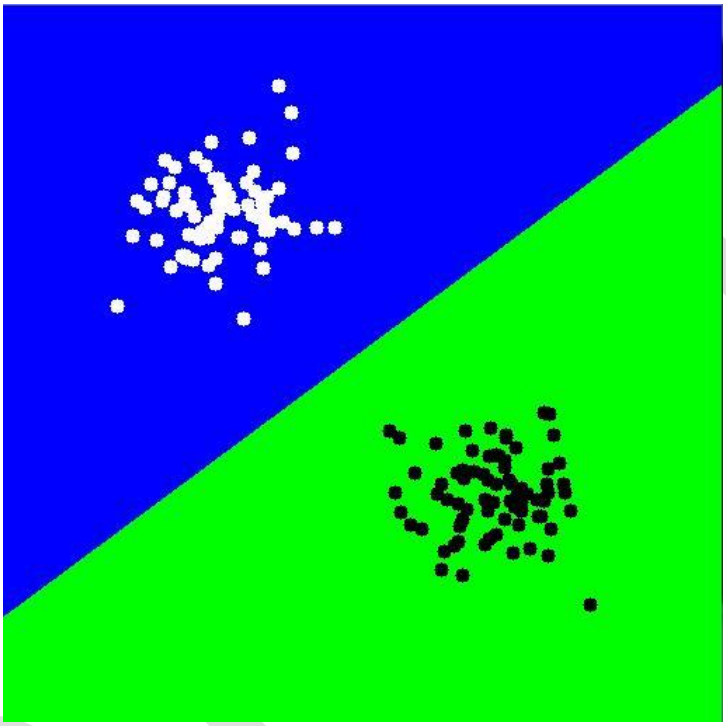
训练时的求解目标是最大化类间差异与类内差异的比值：

$$\max L(\mathrm{W}) = \frac{\mathrm{tr}\left(\mathrm{W}^{\mathrm{T}}\mathrm{S}_B\mathrm{W}\right)}{\mathrm{tr}\left(\mathrm{W}^{\mathrm{T}}\mathrm{S}_W\mathrm{W}\right)}$$

最后归结为求解矩阵的特征值和特征向量：

$$\mathrm{S}_B\mathrm{w}_i = \lambda_i\mathrm{S}_W\mathrm{w}_i$$

如果我们要将向量投影到 c-1 维，则挑选出最大的 c-1 个特征值以及它们对应的特征向量，组成矩阵 W。线性判别分析不能直接用于分类问题，它只是完成投影，投影之后还需要用其他算法进行分类，如 kNN。下图是 LDA 降维之后用最小距离分类器分类的结果：



从这张图可以看出，决策面是直线。LDA 是一种线性模型，也是判别模型，只能用于分类问题。

logistic 回归即对数几率回归，它的名字虽然叫“回归”，但却是一种用于二分类问题的分类算法，它用 sigmoid 函数估计出样本属于某一类的概率。这种算法可以看做是对线性分类器的改进。

预测函数为：

$$h(\mathrm{x}) = \frac{1}{1+\exp\left(-\mathrm{w}^{\mathrm{T}}\mathrm{x}\right)}$$

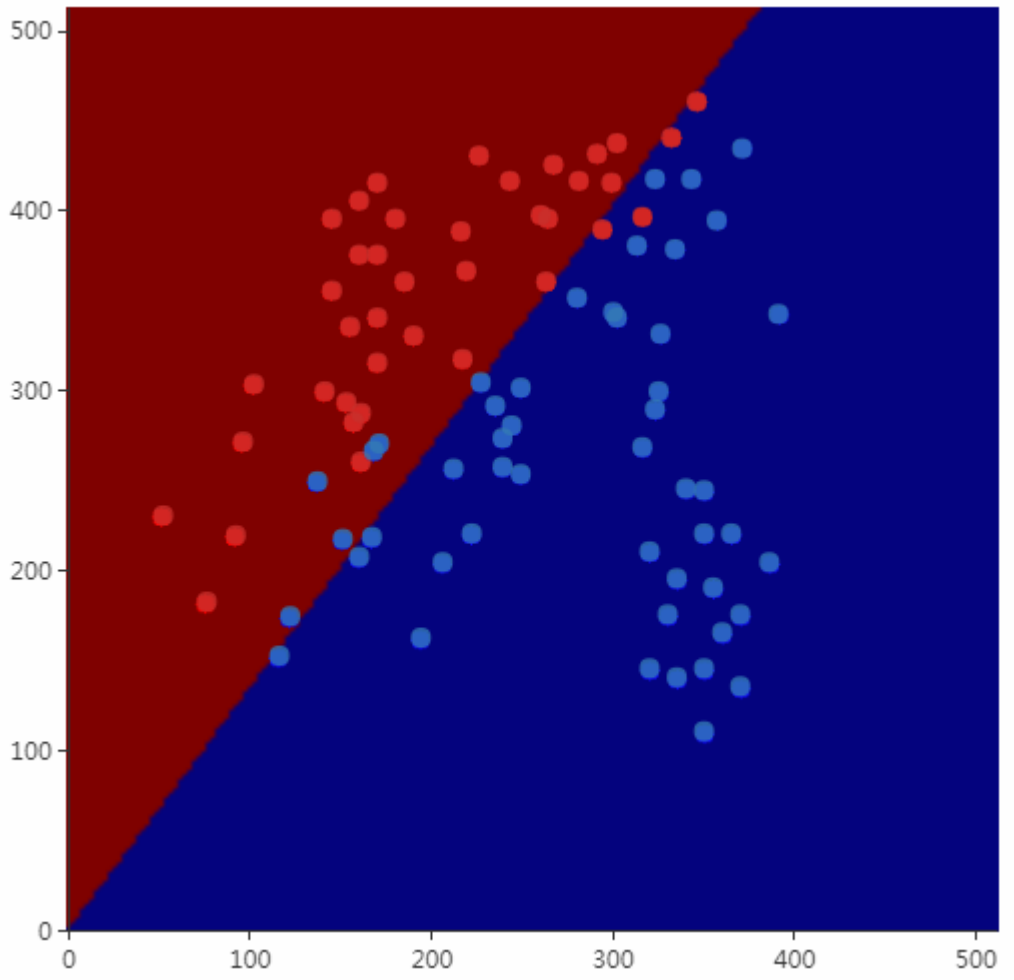
其中 w 为线性映射权向量，由训练算法确定。训练时的优化目标是最大化对数似然函数：

$$\max_{\mathrm{w}} \sum_{i=1}^{l}\left(y_i \log h(\mathrm{x}_i) + (1-y_i)\log\left(1-h(\mathrm{x}_i)\right)\right)$$

这是一个凸优化问题，可以得到全局最优解，求解时可以采用梯度下降法或者牛顿法。分类时的判断规则为：

$$\mathrm{sgn}\left(\mathrm{w}^{\mathrm{T}}\mathrm{x}\right)$$

logistic 回归是一种判别模型，也是线性模型，它只支持二分类问题。下图是用 logistic 回归进行分类的结果（来自 SIGAI 云端实验室）：



从上图可以看到，分界面是一条直线，这也说明了它是一个线性模型。

logistic 回归只能用于二分类问题，将它进行推广可以得到处理多类分类问题的 softmax 回归。softmax 回归按照下面的公式估计一个样本属于每一类的概率：

$$h_{\theta}\left(\mathrm{x}\right)=\frac{1}{\sum_{i=1}^{k}e^{\theta_i^{\mathrm{T}}\mathrm{x}}}\begin{bmatrix}e^{\theta_1^{\mathrm{T}}\mathrm{x}}\\ \cdots\\ e^{\theta_k^{\mathrm{T}}\mathrm{x}}\end{bmatrix}$$

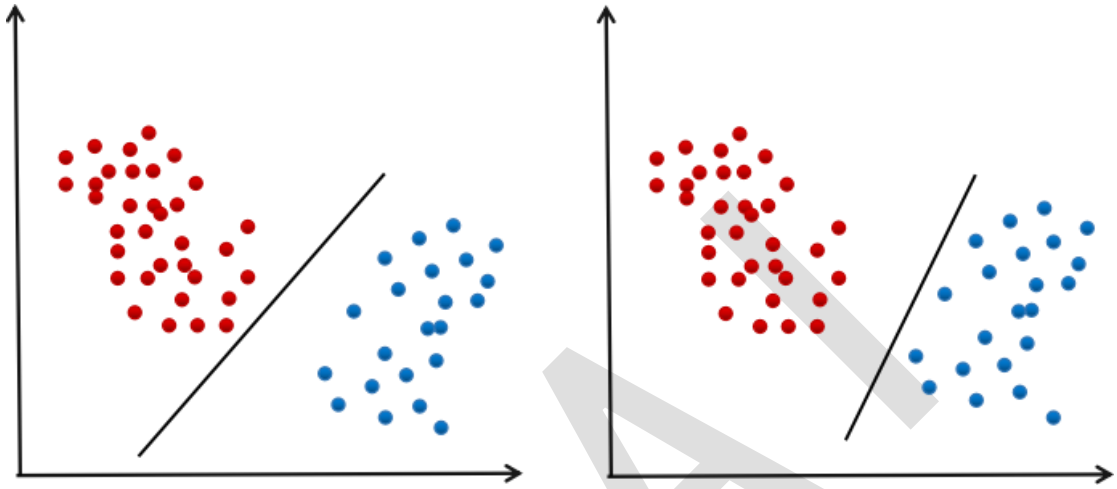
模型的输出为一个 k 维向量，其元素之和为 1，每一个分量为样本属于该类的概率。训练时的损失函数定义为：

$$\min_{\theta}L\left(\theta\right)=-\sum_{i=1}^{l}\sum_{j=1}^{k}\left(1_{y_i=j}\log\frac{\exp\left(\theta_j^{\mathrm{T}}\mathrm{x}_i\right)}{\sum_{t=1}^{k}\exp\left(\theta_t^{\mathrm{T}}\mathrm{x}_i\right)}\right)$$

上式是对 logistic 回归损失函数的推广。这个损失函数是凸函数，可以采用梯度下降法求解。Softmax 回归是一种判别模型，也是线性模型，它支持多分类问题。

支持向量机

支持向量机是对线性分类器的改进，加上了最大化分类间隔的约束，另外还使用了核技术，通过非线性核解决非线性问题。一般情况下，给定一组训练样本可以得到不止一个可行的线性分类器，下图就是一个例子：



在上图中两条直线都可以将两类样本分开。问题是：在多个可行的线性分类器中，什么样的分类器是好的？为了得到好的泛化性能，分类平面应该不偏向于任何一类，并且离两个类的样本都尽可能的远。这种最大化分类间隔的目标就是支持向量机的基本思想。支持向量机在训练时优化的目标是让训练样本尽量分类正确，而且决策面离两类样本尽可能远。原问题带有太多的不等式约束，一般转化为对偶问题求解，使用拉格朗日对偶，加上核函数之后，优化的对偶问题为：

$$\min_{\alpha} \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l}\alpha_i\alpha_j y_i y_j K\left(\mathrm{x}_i^{\mathrm{T}}\mathrm{x}_j\right)-\sum_{k=1}^{l}\alpha_k$$

$$0 \le \alpha_i \le C$$

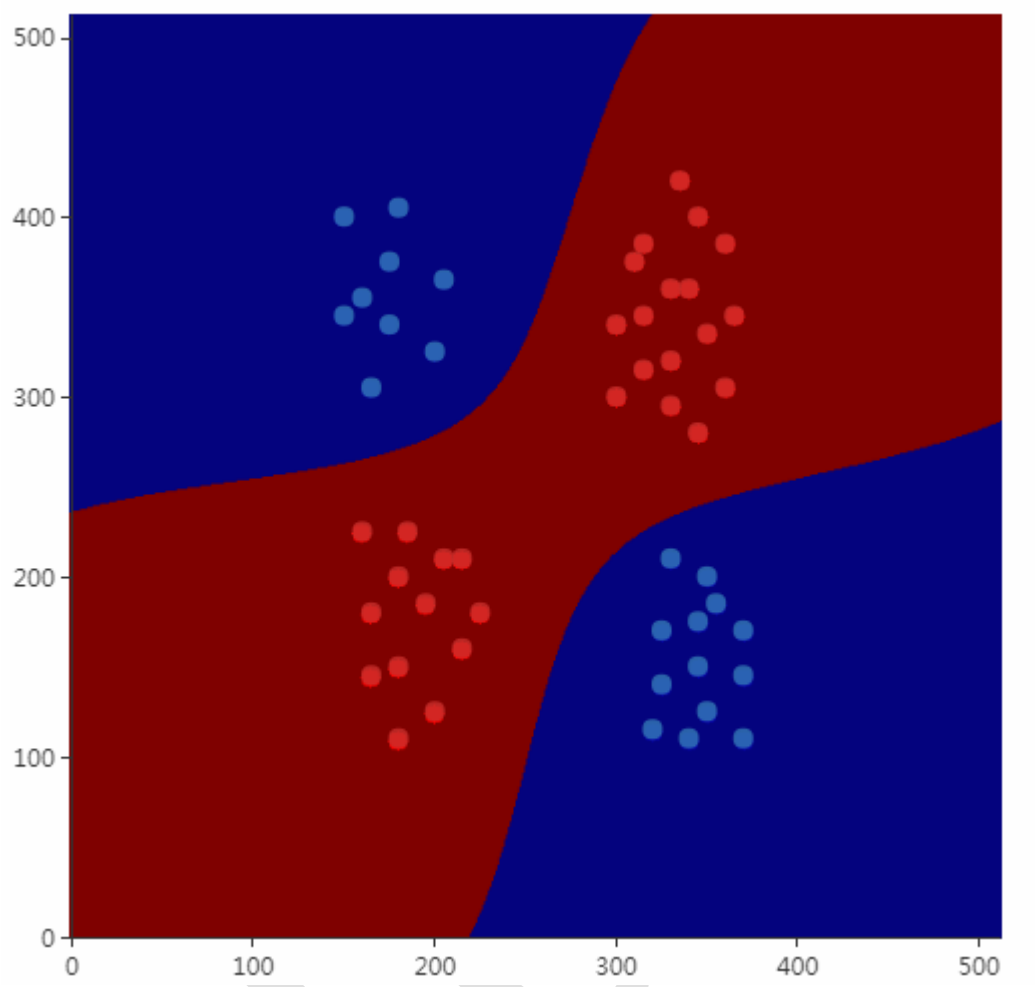
$$\sum_{j=1}^{l}\alpha_j y_j = 0$$

预测函数为：

$$\operatorname{sgn}\left(\sum_{i=1}^{l}\alpha_i y_i K\left(\mathrm{x}_i^{\mathrm{T}}\mathrm{x}\right)+b\right)$$

这是一个凸优化问题，可以得到全局最优解，求解时一般采用 SMO 算法，这是一种分治法，每次挑选出两个变量进行优化，对这两个变量的优化问题求公式解。优化变量的选择使用了 KKT 条件。

支持向量机是一种判别模型，既支持分类问题，也支持回归问题，如果使用非线性核，则是一种非线性模型，这从它的预测函数也可以看出来。标准的支持向量机只能解决二分类问题，通过多个二分类器组合，可以解决多分类问题，另外一种思路是直接构造多类的损失函数来解决多分类问题。下图是用支持向量机对异或问题进行分类的结果（来自 SIGAI 云端实验室）：



神经网络

人工神经网络是一种仿生方法，受启发于人脑的神经网络。从数学上看，它本质上是一个多层复合函数。如果使用 sigmoid 作为激活函数，它的单个神经元就是 logistic 回归。假设神经网络的输入是 n 维向量 x，输出是 m 维向量 y，它实现了如向量到向量的映射：

$$\mathbb{R}^n \to \mathbb{R}^m$$

将这个函数记为：

$$\mathrm{y} = h(\mathrm{x})$$

神经网络第 l 层的变换写成矩阵和向量形式为：

$$\mathrm{u}^{(l)} = \mathrm{W}^{(l)}\mathrm{x}^{(l-1)} + \mathrm{b}^{(l)}$$

$$\mathrm{x}^{(l)} = f\left(\mathrm{u}^{(l)}\right)$$

如果采用欧氏距离，训练时的优化目标为：

$$\min_W L(W) = \frac{1}{2m}\sum_{i=1}^{m}\left\|h(\mathrm{x}_i) - \mathrm{y}_i\right\|^2$$

这不是一个凸优化问题，因此不能保证得到全局极小值。可以采用梯度下降法求解，因为是一个复合函数，需要对各层的权重与偏置求导，采用了反向传播算法，它从多元函数求

导的链式法则导出。误差项的计算公式为，对于输出层：

$$\delta^{(n_l)} = \left(\mathrm{x}^{(n_l)} - \mathrm{y}\right) \odot f'\left(\mathrm{u}^{(n_l)}\right)$$

对于隐含层：

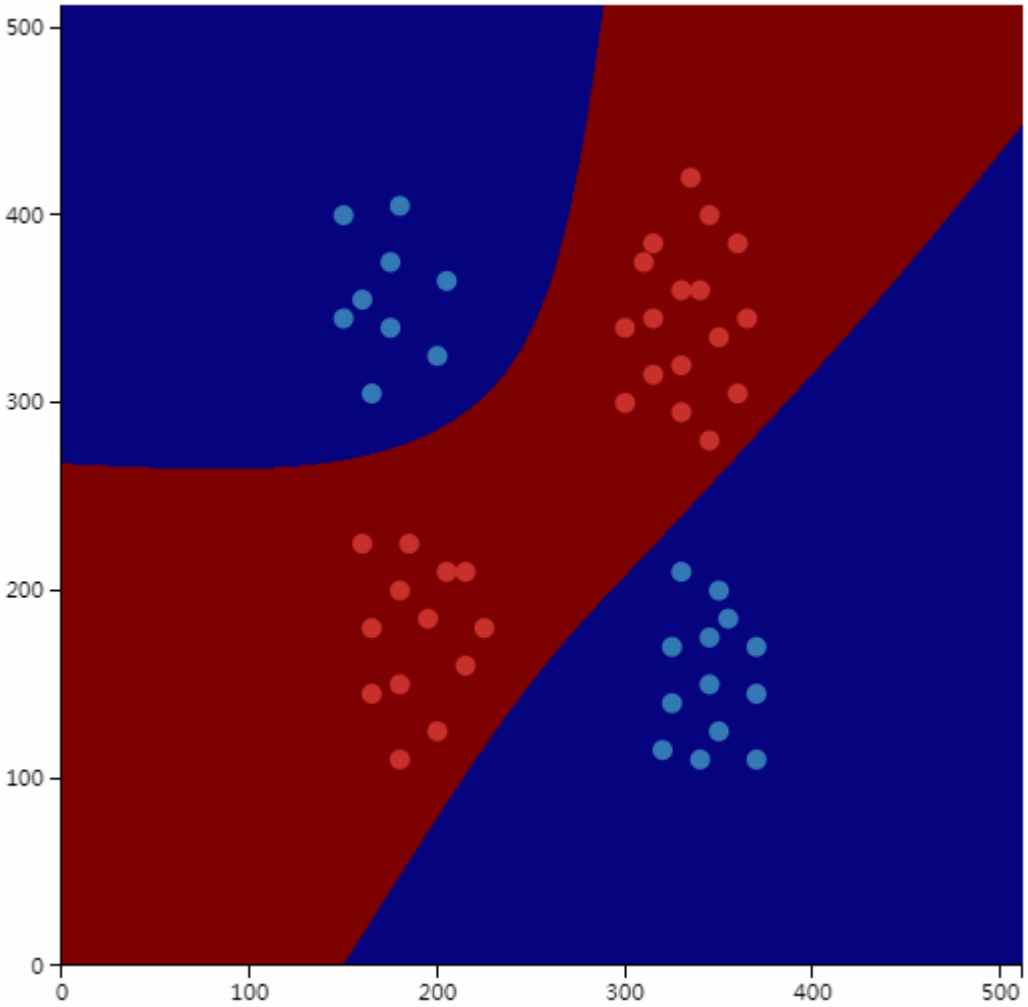
$$\delta^{(l)} = \left(\mathrm{W}^{(l+1)}\right)^{\mathrm{T}} \delta^{(l+1)} \odot f'\left(\mathrm{u}^{(l)}\right)$$

根据误差项可以得到权重和偏置的梯度值：

$$\nabla_{\mathrm{w}^{(l)}} L = \delta^{(l)} \left(\mathrm{x}^{(l-1)}\right)^{\mathrm{T}}$$

$$\nabla_{\mathrm{b}^{(l)}} L = \delta^{(l)}$$

然后用梯度下降法更新。神经网络是一个判别模型，并且是非线性模型，它既支持分类问题，也支持回归问题，并且支持多分类问题。下图是用神经网络对异或问题的分类结果（来自 SIGAI 云端实验室）：



深度神经网络家族

深度神经网络是对多层感知器模型的进一步发展，它可以完成自动的特征提取，端到端的训练，是深度学习的核心技术。可以分为自动编码器，受限玻尔兹曼机，卷积神经网络，循环神经网络，生成对抗网络这几种类型。

自动编码器用一个单层或者多层神经网络对输入数据进行映射，得到输出向量，作为从输入数据提取出的特征。在这种框架中，神经网络的前半部分称为编码器，用于从原始输入数据中提取特征；后半部分称为解码器，训练时根据提取的特征重构原始数据，它只用于训练阶段。

训练时的做法是先经过编码器得到编码后的向量，然后再通过解码器得到解码后的向量，用解码后的向量和原始输入向量计算误差。如果编码器的映射函数为 h，解码器的映射函数为 g，训练时优化的目标函数为：

$$\min \frac{1}{2l}\sum_{i=1}^{l}\left\|\mathrm{x}_i - g_{\theta'}\left(h_\theta\left(\mathrm{x}_i\right)\right)\right\|_2^2$$

在这里同样采用梯度下降法和反向传播算法。自动编码器的改进型有去噪自动编码器，收缩自动编码器，变分自动编码器，稀疏编码等。

单个自动编码器只能进行一层特征提取，可以将多个自动编码器组合起来使用，得到一种称为层叠编码器的结构。层叠自动编编码器由多个自动动编码器串联组成，能够逐层提取输入数据的特征，在此过程中逐层降低输入数据的维度，将高维的输入数据转化成低维的特征。

受限玻尔兹曼机由 Hinton 等人提出，是一种生成式随机神经网络，这是一种概率模型。在这种模型中，神经元的状态值是以随机的方式确定的，而不像之前介绍的神经网络那样是确定性的。

受限玻尔兹曼机的数据分为可见变量和隐变量两种类型，并定义了它们之间的概率关系。可见变量是神经网络的输入数据，如图像；隐变量是从输入数据中提取的特征。在受限玻尔兹曼机中，可见变量和隐藏变量都是二元变量，即其取值只能为 0 或 1，整个神经网络是一个二部图。

可见节点用向量表示为 v，隐藏节点用向量表示为 h。任意可见节点和隐藏节点之间都有边连接。(v, h)的联合概率服从玻尔兹曼分布，联合概率定义为：

$$p\left(\mathrm{v},\mathrm{h}\right)=\frac{1}{Z_\theta}\exp\left(-E_\theta\left(\mathrm{v},\mathrm{h}\right)\right)=\frac{1}{Z_\theta}\exp\left(\mathrm{v}^{\mathrm{T}}\mathrm{W}\mathrm{h}+\mathrm{b}^{\mathrm{T}}\mathrm{v}+\mathrm{d}^{\mathrm{T}}\mathrm{h}\right)$$

训练时迭代更新权重参数直至网络收敛，这种方法称为 Contrastive Divergence。

和自动编码器类似，可以将多个受限玻尔兹曼机层叠加起来使用，在种结构称为深度玻尔兹曼机（Deep Boltzmann Machine），简称 DBM。通过多层的受限玻尔兹曼机，可以完成数据在不同层次上的特征提取和抽象。

在 DBM 中，所有层的节点之间的连接关系是无向的，如果我们限制某些层之间的连接关系为有向的，就得到了另外一种结构，称为深信度网络（Deep Belief Network），简称 DBN。在 DBN 中，靠近输入层的各个层之间的连接关系是有向的，是贝叶斯置信网；靠近输出层的各个层之间的连接关系是无向的，是受限玻尔兹曼机。

在所有深度学习框架中，卷积神经网络应用最为广泛，在机器视觉等具有空间结构的数据问题上取得了成功。标准的卷积神经网络由卷积层，池化层，全连接层构成。可以看做是权重共享的全连接神经网络。

训练时同样采用梯度下降法和反向传播算法。对于卷积层，根据误差项计算卷积核梯度的计算公式为：

$$\nabla_{\mathrm{K}^{(l)}} L = \mathrm{conv}\left(\mathrm{X}^{(l-1)}, \delta^{(l)}\right)$$

卷层误差项的递推公式为：

$$\delta^{(l-1)} = \delta^{(l)} * \mathrm{rot180}\left(\mathrm{K}\right) \odot f^{'}\left(\mathrm{u}^{(l-1)}\right)$$

也可以用矩阵乘法来实现卷积，这种做法更容易理解，可以方便的计算出对卷积核的梯度值。

循环神经网络是仅次于卷积神经网络的第二大深度神经网络结构，在语音识别、自然语言处理等问题上取得了成功。循环神经网络具有记忆功能，用于时间序列数据预测。循环层实现的映射为：

$$\mathrm{h}_t = f\left(\mathrm{W}_{xh}\mathrm{x}_t + \mathrm{W}_{hh}\mathrm{h}_{t-1} + \mathrm{b}\right)$$

输出层实现的映射为：

$$\mathrm{y}_t = g\left(\mathrm{W}_o\mathrm{h}_t + \mathrm{b}_o\right)$$

对单个样本，训练时的损失函数为各个时刻的损失函数之和：

$$L = \sum_{t=1}^{T} L_t$$

这里的反向传播算法称为 BPTT（Back Propagation Through Time），在时间轴上进行反向传播。误差项的递推计算公式为：

$$\delta_t = \left(\left(\mathrm{W}_o\right)^{\mathrm{T}}\left(\left(\nabla_{\mathrm{y}_t^*} L_t\right) \odot g^{'}\left(\mathrm{v}_t\right)\right)\right) \odot f^{'}\left(\mathrm{u}_t\right) + \left(\mathrm{W}_{hh}\right)^{\mathrm{T}} \delta_{t+1} \odot f^{'}\left(\mathrm{u}_t\right)$$

$$\delta_T = \left(\mathrm{W}_o\right)^{\mathrm{T}}\left(\nabla_{\mathrm{v}_T} L\right) \odot f^{'}\left(\mathrm{u}_T\right) = \left(\mathrm{W}_o\right)^{\mathrm{T}}\left(\left(\nabla_{\mathrm{y}_T^*} L\right) \odot g^{'}\left(\mathrm{v}_T\right)\right) \odot f^{'}\left(\mathrm{u}_T\right)$$

根据误差项计算权重和偏置的公式为：

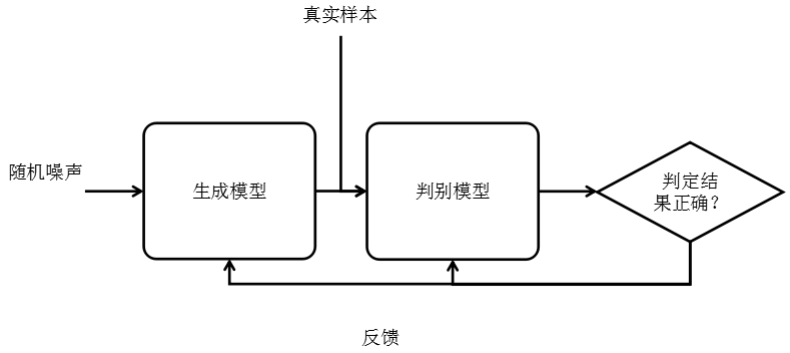
$$\nabla_{\mathrm{W}_{hh}} L = \sum_{t=1}^{T}\left(\nabla_{\mathrm{u}_t} L\right)\mathrm{h}_{t-1}^{\mathrm{T}} = \sum_{t=1}^{T} \delta_t \mathrm{h}_{t-1}^{\mathrm{T}}$$

$$\nabla_{\mathrm{W}_{xh}} L = \sum_{t=1}^{T}\left(\nabla_{\mathrm{u}_t} L\right)\mathrm{x}_t^{\mathrm{T}} = \sum_{t=1}^{T} \delta_t \mathrm{x}_t^{\mathrm{T}}$$

$$\nabla_{\mathrm{b}_h} L = \sum_{t=1}^{T} \nabla_{\mathrm{u}_t} L = \sum_{t=1}^{T} \delta_t$$

生成对抗网络（Generative Adversarial Network，简称 GAN）是用机器学习的方法来解决数据生成问题的一种框架，它的目标是生成服从某种随机分布的数据，由 Goodfellow 在 2014 年提出。 这种模型能够找出样本数据内部的概率分布规律，并根据这种规律产生出新的数据。

整个框架由一个生成模型和一个判别模型组成。生成模型用于学习真实数据的概率分布，并生成符合这种分布的数据；判别模型的任务是判断一个输入数据是来自于真实数据集还是由生成模型生成的。在训练时，通过两个模型之间不断的竞争，从而分别提高这两个模型的生成能力和判别能力。



生成模型的输入是随机噪声 z，输出是产生的数据 G(z)。判别模型的输入是真实样本，或者生成网络生成的数据，得到的是它们的分类结果 D(x)。

训练的目标是让判别模型能够最大程度的正确区分真实样本和生成模型生成的样本；同时要让生成模型使生成的样本尽可能的和真实样本相似。即：判别模型要尽可能将真实样本判定为真实样本，将生成模型产生的样本判定为生成样本；生成模型要尽量让判别模型将自己生成的样本判定为真实样本。基于以上 3 个要求，对于生成模型生成的样本，要最小化如下目标函数：

$$\log\left(1-D\left(G\left(\mathrm{z}\right)\right)\right)$$

这意味着如果生成模型生成的样本和真实样本越接近，被判别模型判断为真实样本的概率就越大，即 $D\left(G\left(\mathrm{z}\right)\right)$ 的值越接近于 1，目标函数的值越小。另外还要考虑真实的样本，对真实样本要尽量将它判别成 1。这样要优化的目标函数定义为：

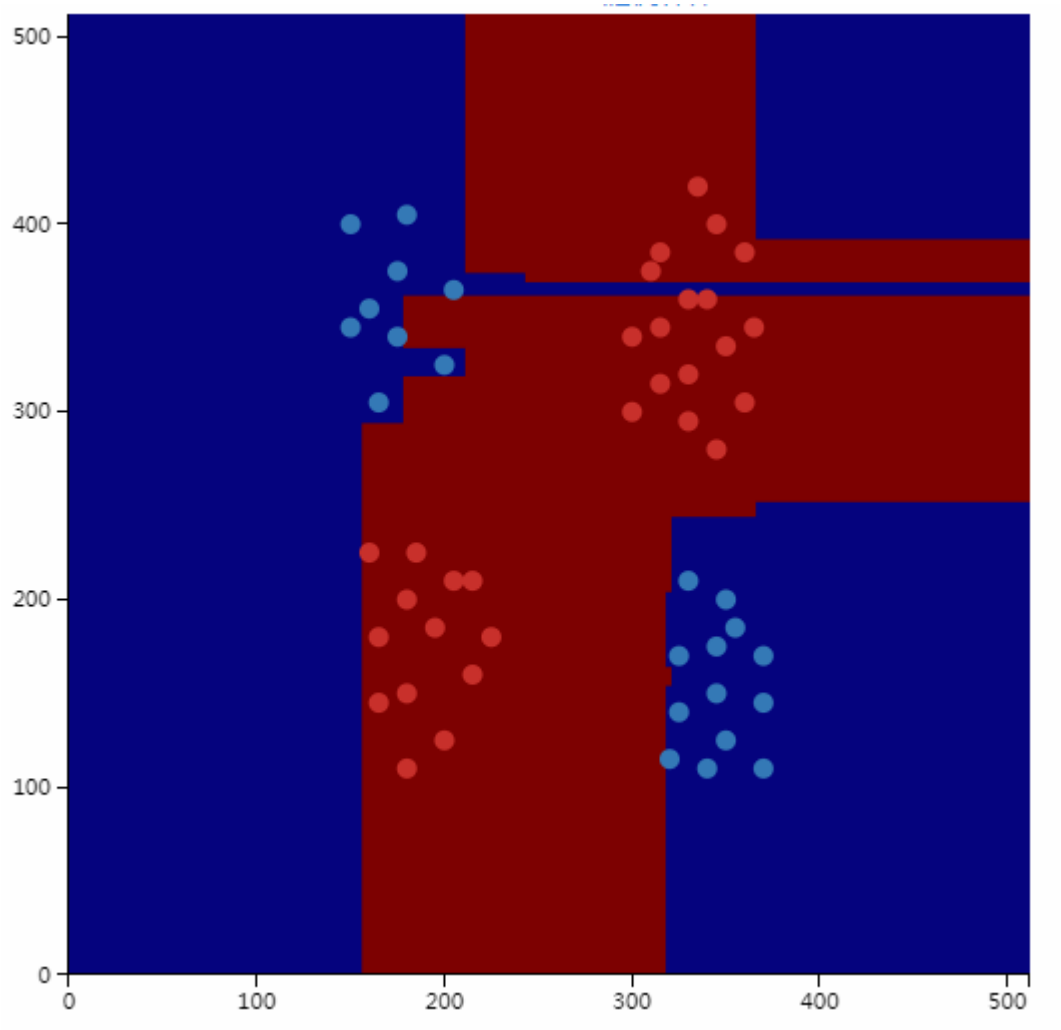
$$\min_G \max_D V\left(D,G\right)=\mathrm{E}_{\mathrm{x}\sim p_{data}(\mathrm{x})}\left[\log D\left(\mathrm{x}\right)\right]+\mathrm{E}_{\mathrm{z}\sim p_z(\mathrm{z})}\left[\log\left(1-D\left(G\left(\mathrm{z}\right)\right)\right)\right]$$

在这里判别模型和生成模型是目标函数的自变量，它们的参数是要优化的变量。求解时采用了交替优化的策略，先固定住生成网络，训练判别网络；然后固定住判别网络，训练生成网络。每个网络的训练都采用梯度下降法和反向传播算法。

集成学习家族

集成学习（ensemble learning）是一类机器学习算法，它通过多个模型的组合形成一个精度更高的模型，参与组合的模型称为弱学习器（weak learner）。在预测时使用这些弱学习器模型联合进行预测；训练时需要用训练样本集依次训练出这些弱学习器。随机森林和 AdaBoost 算法是这类算法的典型代表。

随机森林由多棵决策树组成。用多棵决策树联合预测可以提高模型的精度，这些决策树用对训练样本集随机抽样构造出样本集训练得到。由于训练样本集由随机抽样构造，因此称为随机森林。随机森林不仅对训练样本进行抽样，还对特征向量的分量随机抽样，在训练决策树时，每次分裂时只使用一部分抽样的特征分量作为候选特征进行分裂。下图是随机森林对异或问题的分类结果（来自 SIGAI 云端实验室）：



对应的随机森林如下图所示：

随机森林是一种判别模型，也是一种非线性模型，它既支持分类问题，也支持回归问题，并且支持多分类问题，有很好的解释性。

Boosting 算法也是一种集成学习算法。它的分类器由多个弱分类器组成，预测时用每个弱分类器分别进行预测，然后投票得到结果；训练时依次训练每个弱分类器，在这里和随机森林采用了不同的策略，不是对样本进行随机抽样构造训练集，而是重点关注被前面的弱分类器错分的样本。弱分类器是很简单的分类器，它计算量小且精度不用太高。

AdaBoost 算法由 Freund 等人提出，是 Boosting 算法的一种实现版本。在最早的版本中，这种方法的弱分类器带有权重，分类器的预测结果为弱分类器预测结果的加权和。训练时训练样本具有权重，并且会在训练过程中动态调整，被前面的弱分类器错分的样本会加大权重，因此算法会关注难分的样本。

强分类器的计算公式为：

$$F(\mathrm{x})=\sum_{t=1}^{T}\alpha_t f_t(\mathrm{x})$$

分类时的判定规则为：

$$\mathrm{sgn}\left(F(\mathrm{x})\right)$$

训练目标是最小化指数损失函数：

$$L\left(y,F(\mathrm{x})\right)=\exp\left(-yF(\mathrm{x})\right)$$

求解时采用了分阶段优化的策略，先把弱分类器的权重当做常数，优化弱分类器。得到弱分类器之后，再优化它的权重。弱分类器的权重计算公式为：

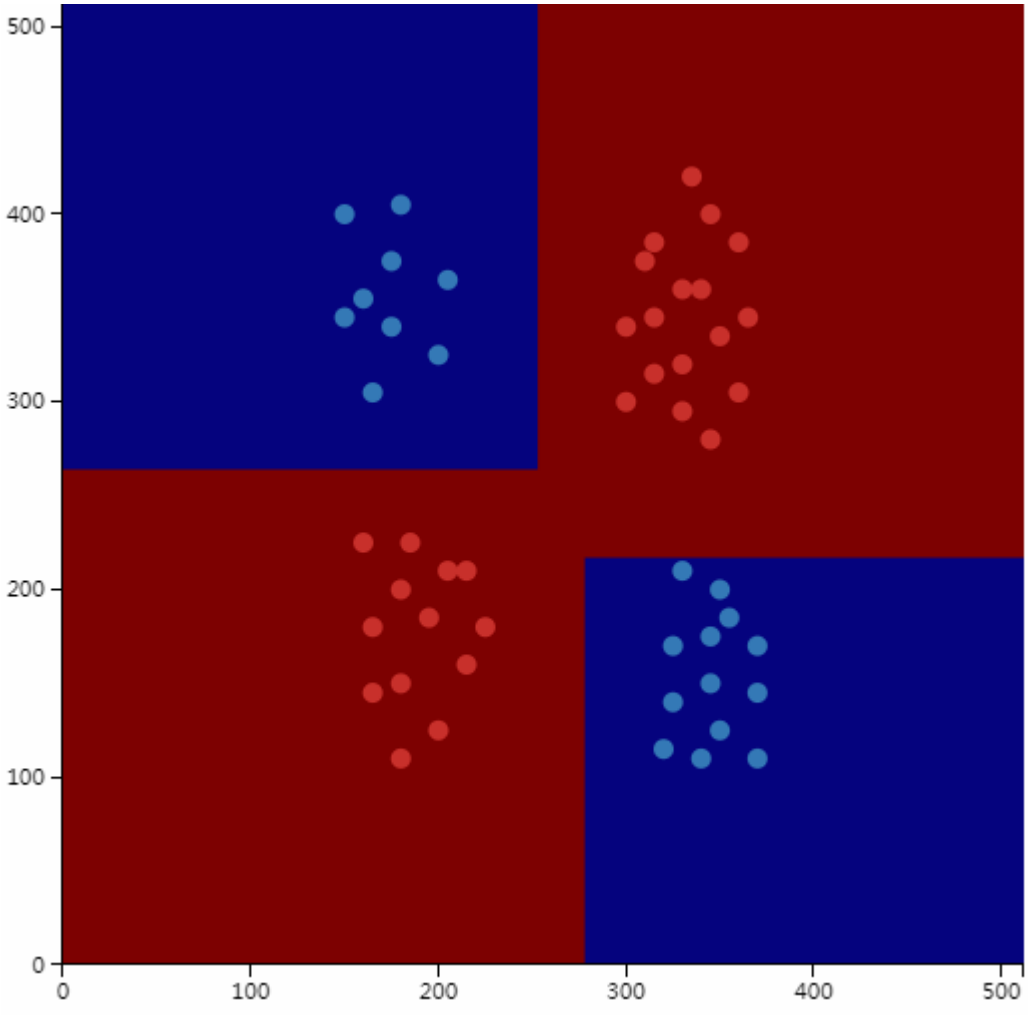
$$\alpha_t=\frac{1}{2}\log\left((1-e_t)/e_t\right)$$

样本权重的更新公式为：

$$w_i^t=w_i^{t-1}\exp\left(-y_i\alpha_t f_t(\mathrm{x}_i)\right)/Z_t$$

AdaBoost 算法的原则是关注之前被错分的样本，准确率高的弱分类器有更大的权重。

AdaBoost 算法是一个判别模型，也是非线性模型，它只支持二分类问题。下图是用 AdaBoost 算法对异或问题的分类结果（来自 SIGAI 云端实验室）：



无监督学习

相对于有监督学习，无监督学习的研究进展更为缓慢，算法也相对较少。无监督学习可以分为聚类和降维两大类，下面分别介绍。

聚类算法

聚类属于无监督学习问题，其目标是将样本集划分成多个类，保证同一类的样本之间尽量相似，不同类的样本之间尽量不同。这些类被称为簇（cluster）。与有监督的分类算法不同，聚类算法没有训练过程，直接完成对一组样本的划分从而确定每个样本的类别归属。我们一般将距离算法分为层次距离，基于质心的聚类，基于密度的聚类，基于概率分布的聚类，基于图的聚类这几种类型，它们从不同的角度定义簇。

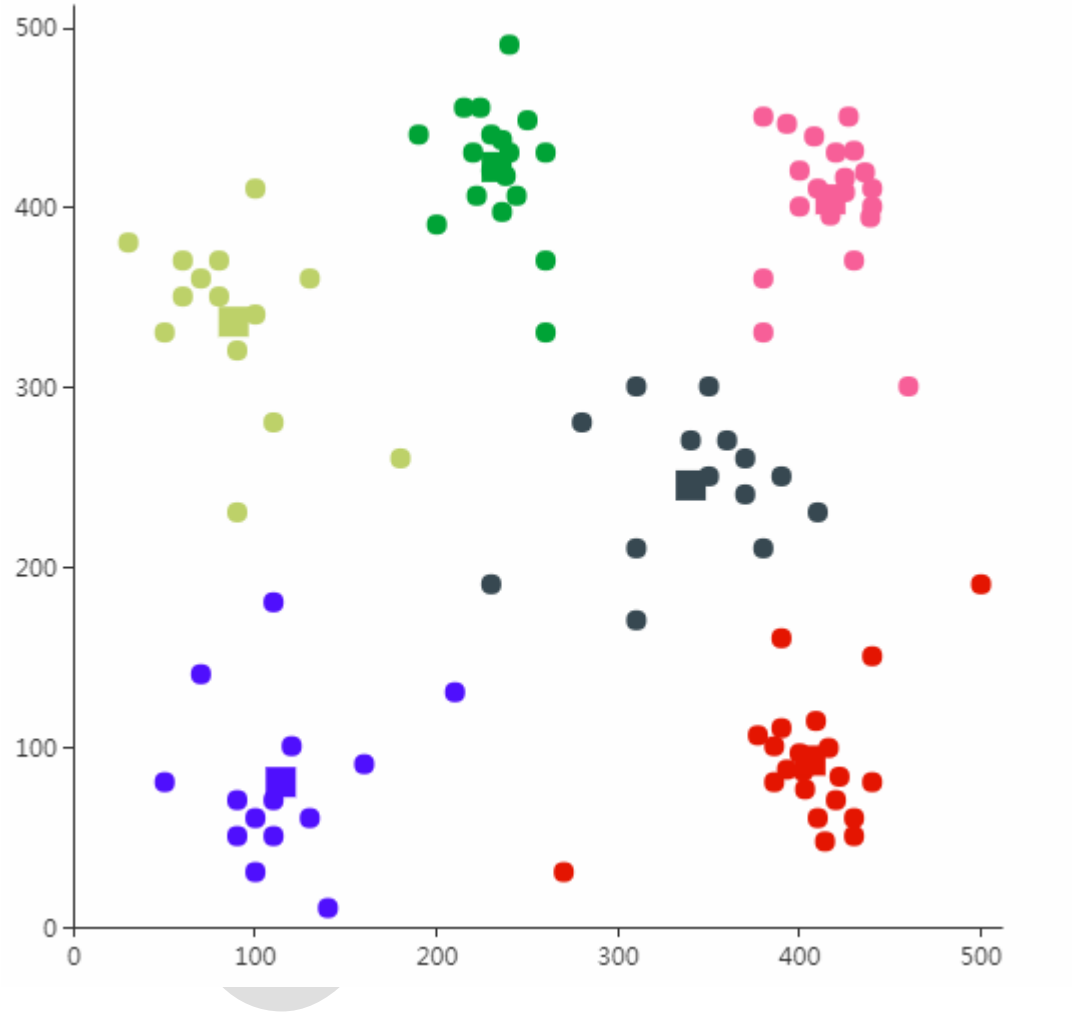
k 均值算法是一种被广为用于实际问题的聚类算法。它将样本划分成 k 个类，参数 k 由人工设定。算法将每个样本分配到离它最近的那个类中心所属的类，而类中心的确定又依赖于样本的分配方案。假设样本集有 l 个样本，给定参数 k 的值，算法将这些样本划分成 k 个集合：

$$S = \{S_1, ..., S_k\}$$

最优分配方案是如下最优化问题的解：

$$\min_s \sum_{i=1}^{k} \sum_{x \in S_i} \|x - \mu_i\|^2$$

其中 μ_i 为类中心向量。这个问题是 NP 难问题，不易求得全局最优解，只能近似求解。实现时采用迭代法近似求解，只能保证收敛的局部最优解处。每次迭代时，首先计算所有样本离各个类中心的距离，然后将其分配到最近的那个类；接下来再根据这种分配方案更新类中心向量。下图为 k 均值算法的聚类结果（来自 SIGAI 云端实验室）：



基于概率分布的算法假设每一个簇的样本服从相同的概率分布，这是一种生成模型。经常使用的是多维正态分布，如果服从这种分布，则为高斯混合模型，在求解时一般采用 EM 算法。

EM 算法是一种迭代法，其目标是求解似然函数或后验概率的极值，而样本中具有无法观测的隐含变量 z。例如有一批样本分属于 3 个类，每个类的样本都服从正态分布，均值和协方差未知，并且每个样本属于哪个类也是未知的，需要在这种情况下估计出每个正态分布的均值和协方差。算法在实现时分为两步：

E 步，基于当前的参数估计值 θ_i，计算在给定 x 时对 z 的条件概率的数学期望：

$$Q_i(z_i) = p(z_i | x_i; \theta)$$

M 步，求解如下极值问题，更新 θ 的值：

$$\theta = \arg\max_{\theta} \sum_{i} \sum_{z_i} Q_i(z_i) \log \frac{p(x_i, z_i; \theta)}{Q_i(z_i)}$$

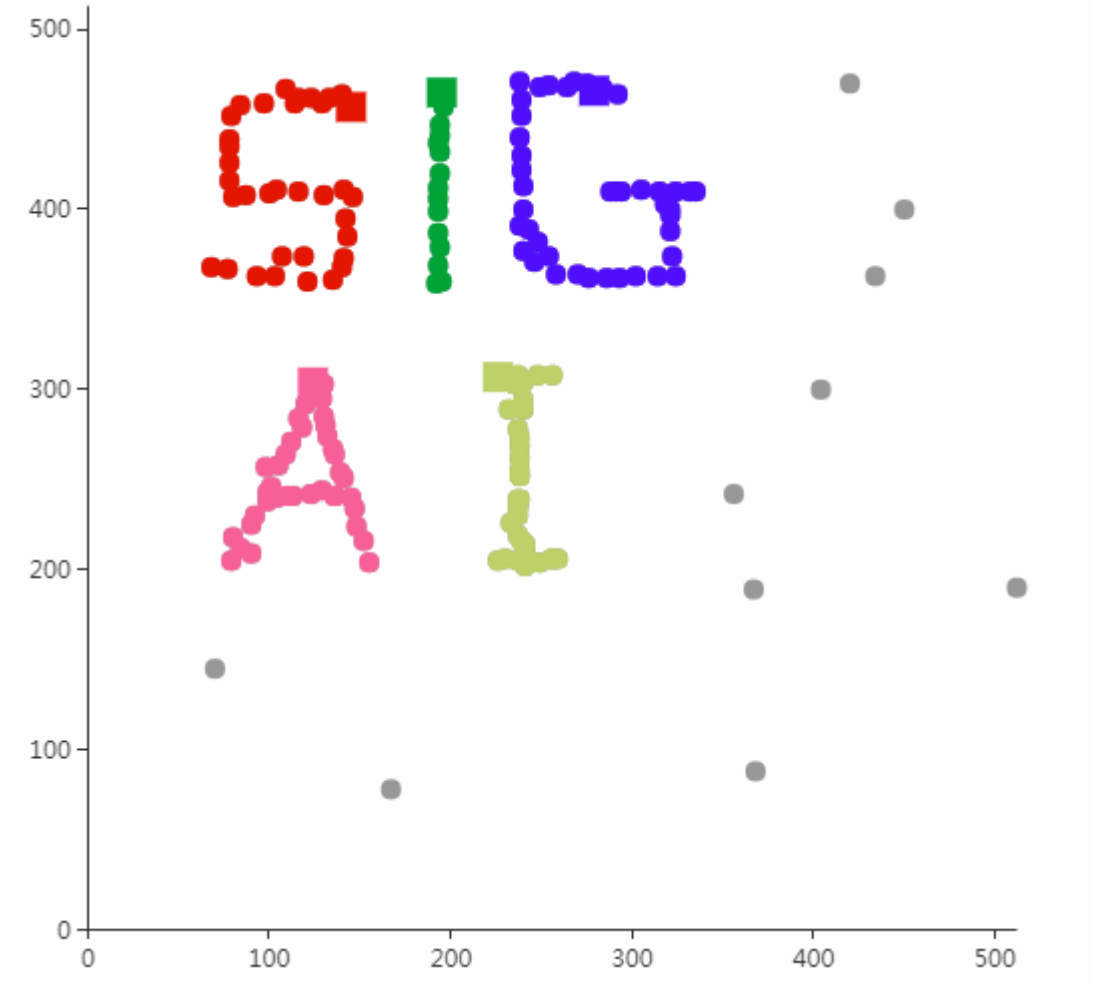
实现时 Q_i 可以按照下面的公式计算：

$$Q_i(z_i) = \frac{p(x_i, z_i; \theta)}{\sum_{z} p(x_i, z; \theta)}$$

迭代终止的判定规则是相邻两次函数值之差小于指定阈值。

DBSCAN 算法是一种基于密度的算法，对噪声鲁棒。它将簇定义为样本密集的区域，算法从一个种子样本开始，反复向密集的区域生长，直至到达边界。

算法首先找出核心点，即周围样本非常密集的那些样本点。然后从某一核心点出发，不断向密度可达的区域扩张，得到一个包含核心点和边界点的最大区域，这个区域中任意两点密度相连。下图是 DBSCAN 算法的聚类结果（来自 SIGAI 云端实验室）：



OPTICS 算法是对 DBSCAN 算法的改进，对参数更不敏感。它不直接生成簇，而是对本进行排序，这种排序包含了聚类信息。

均值漂移（Mean Shift）算基于核密度估计技术，是一种寻找概率密度函数极值点的算法。在用于聚类任务时，它寻找概率密度函数的极大值点，即样本分别最密集的位置，以此得到簇。

基于图的算法把样本数据看成图的顶点，通过数据点之间的距离构造边，形成带权图。通过图的切割实现聚类，即将图切分成多个子图，这些子图就是对应的簇。基于图的聚类算法典型的代表是谱聚类算法。谱聚类算法首先构造数据的邻接图，得到图的拉普拉斯矩阵。接下来对矩阵进行特征值分解，通过特征值和特征向量构造出簇。

数据降维

在有些应用中，向量的维数非常高。以图像数据为例，对于高度和宽度分别为 100 像素的图像，如果将所有像素值拼接起来形成一个向量，这个向量的维数是 10000。另外向量的各个分量之间可能存在相关性。直接将向量送入机器学习算法中处理效率会很低，也影响算法的精度。为了可视化显示数据，我们也需要把向量变换到低维空间中。

主成分分析（principal component analysis，简称 PCA）是一种数据降维和去除相关性的方法，它通过线性变换将向量投影到低维空间。对向量进行投影就是让向量左乘一个矩阵得到结果向量，这也是线性代数中讲述的线性变换：

$$\mathrm{y} = \mathrm{W}\mathrm{x}$$

降维要确保的是在低维空间中的投影能很好的近似表达原始向量，即重构误差最小化。下图是主分量投影示意图：

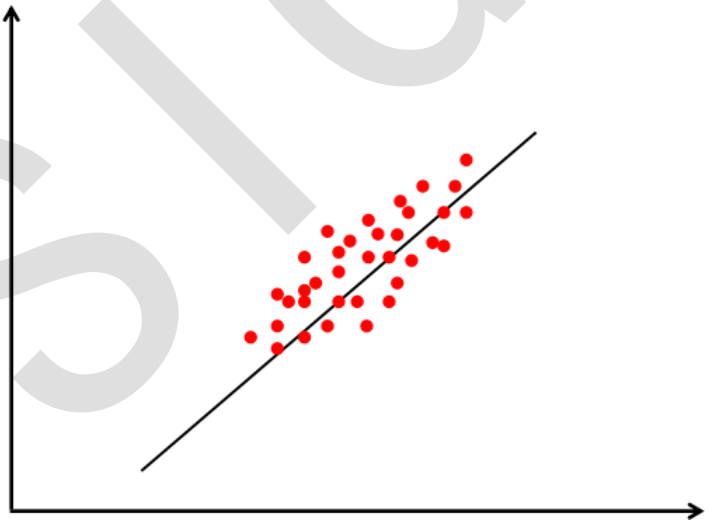


图 7.1　主分量投影示意图

在上图中样本用红色的点表示，倾斜的直线是它们的主要变化方向。将数据投影到这条直线上即完成数据的降维，把数据从 2 维降为 1 维。

寻找投影矩阵时要优化的目标是使得重构误差最下化：

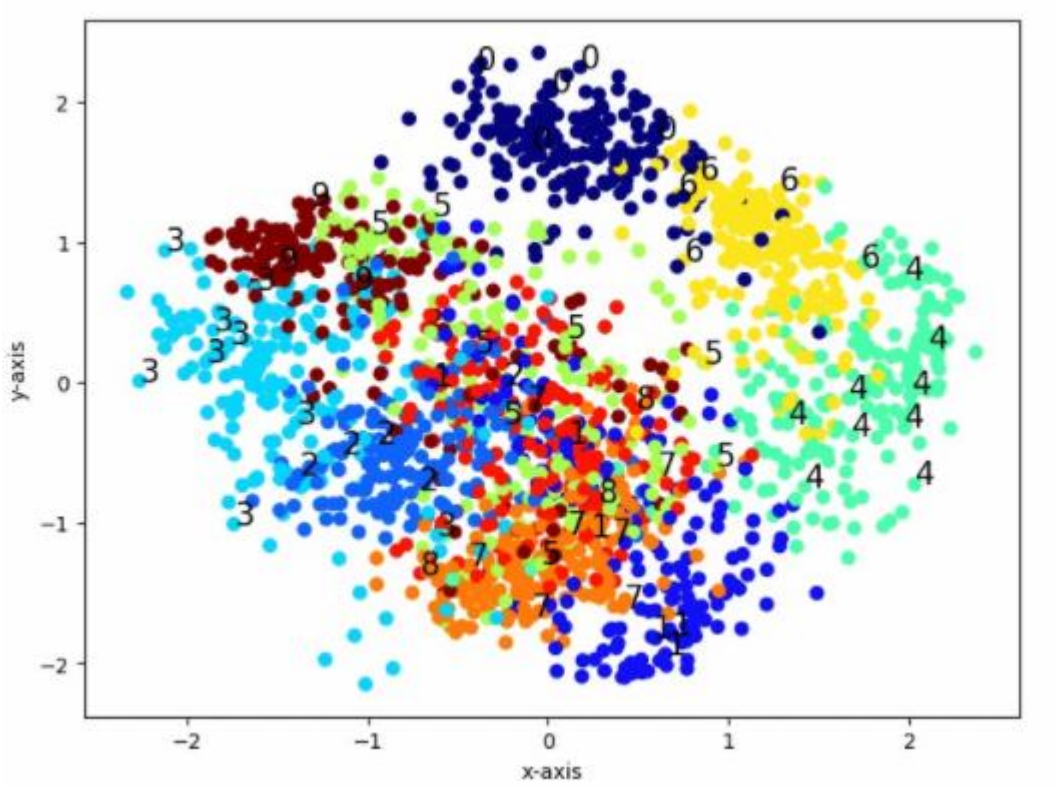
$$L = \sum_{i=1}^{n} \left\| \mathrm{m} + \sum_{j=1}^{d'} a_{ij} \mathrm{e}_j - \mathrm{x}_i \right\|^2$$

使得该函数取最小值的 e_j 为散度矩阵最大的 d' 个特征值对应的单位长度特征向量。即

求解下面的优化问题：

$$\min_{W} -\mathrm{tr}\left(W^{T}SW\right)$$
$$W^{T}W = I$$

矩阵 W 的列 e_j 是我们要求解的基向量。散度矩阵是实对称矩阵，因此属于不同特征值的特征向量是正交的。下图是主成分分析对手写数字图像的降维结果（来自 SIGAI 云端实验室）：



虽然都是线性投影算法，但主成分分析和线性判别分析有本质的不同，前者是无监督的，后者是有监督的，计算过程中使用了类别标签信息。

主成分分析是一种线性降维技术，对于高度非线性的数据具有局限性，而在实际应用中很多时候数据是非线性的。此时可以采用非线性降维技术，流形学习（manifold learning）是非线性降维技术的典型代表

流形是微分几何中的一个概念，它是高维空间中的几何结构，即空间中的点构成的集合，可以简单的理解成二维空间的曲线，三维空间的曲面在更高维空间的推广。下图是三维空间中的一个流形，这是一个卷曲的曲面：

假设有一个 N 维空间中的流形 M，即：

$$M \subset \mathbb{R}^{N}$$

流形学习降维要实现的是如下映射：

$$M \to \mathbb{R}^{n}$$

其中 $n \ll N$ 。即将 N 维空间中流形 M 上的点映射为 n 为空间中的点

局部线性嵌入（简称 LLE）将高维数据投影到低维空间中，并保持数据点之间的局部线性关系。其核心思想是每个点都可以由与它相近的多个点的线性组合来近似，投影到低维空间之后要保持这种线性重构关系，并且有相同的重构系数。

每个数据点和它的邻居位于或者接近于流形的一个局部线性片段上，即可以用它邻居点的线性组合来重构，组合系数刻画了这些局部面片的几何特性：

$$\mathrm{x}_i \approx \sum_j w_{ij} \mathrm{x}_j$$

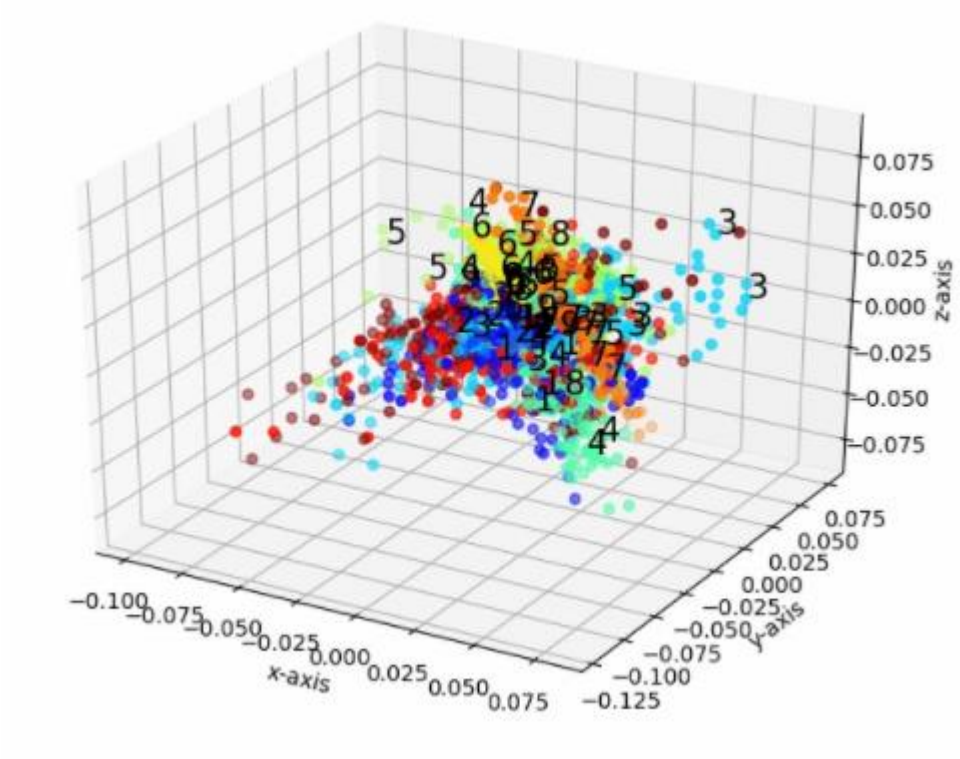
权重系数通最小化下面的重构误差确定：

$$\min_{w_{ij}} \sum_{i=1}^{l} \left\| \mathrm{x}_i - \sum_{j=1}^{l} w_{ij} \mathrm{x}_j \right\|^2$$

假设非线性映射将向量从 D 维空间的 x 映射为 d 维空间的 y。每个点在 d 维空间中的坐标由下面的最优化问题确定：

$$\min_{\mathrm{y}_i} \sum_{i=1}^{l} \left\| \mathrm{y}_i - \sum_{j=1}^{l} w_{ij} \mathrm{y}_j \right\|^2$$

这里的权重和上一个优化问题的值相同，在前面已经得到。下图为用 LLE 算法将手写数字图像投影到 3 维空间后的结果（来自 SIGAI 云端实验室）：



等距映射（Isomap）使用了微分几何中测地线的思想，它希望数据在向低维空间映射之后能够保持流形上的测地线距离。

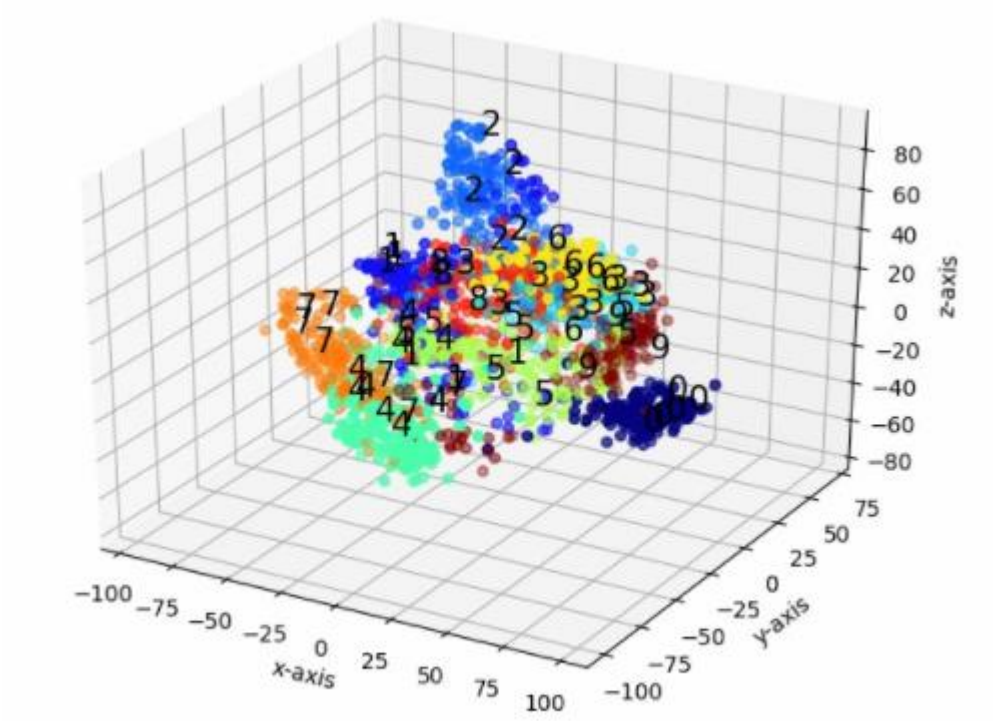
测地线源自于大地测量学，是指地球上任意两点之间在球面上的最短路径。在三维空间中两点之间的最短距离是它们之间线段的长度，但如果要沿着地球表面走，最短距离就是测地线的长度，因为我们不可能从地球内部穿过去。算法计算任意两个样本之间的测地距离，然后根据这个距离构造距离矩阵。最后通过距离矩阵求解优化问题完成数据的降维，降维之

后的数据保留了原始数据点之间的距离信息。

降维过求解如下最优化问题实现：

$$\min_{\mathrm{y}} \sum_{i=1}^{N}\sum_{j=1}^{N}\left(d_G(i,j)-\left\|\mathrm{y}_i-\mathrm{y}_j\right\|\right)^2$$

这个目标函数的意义是向量降维之后任意两点之间的距离要尽量的接近在原始空间中这两点之间的最短路径长度，因此可以认为降维尽量保留了数据点之间的测地距离信息。下图是等距映射对手写数字图像降维后的结果（来自 SIGAI 云端实验室）：



强化学习

强化学习是一类特殊的机器学习算法，如果说有监督学习和无监督学习是要根据预测函数来确定输出标签信息或者聚类类别、降维后的向量，则强化学习算法是要根据当前的状态确定要执行的动作。

强化学习与有监督学习和无监督学习的目标不同，它借鉴于行为主义心理学。算法要解决的问题是智能体在环境中怎样执行动作以获得最大的累计奖励。对于自动行驶的汽车，强化学习算法控制汽车的动作，保证安全行驶。智能体指强化学习算法，环境是类似车辆当前状态与路况这样的由若干参数构成的系统，奖励是我们期望得到的结果，如汽车正确的在路面上行驶而不发生事故。

很多控制、决策问题都可以抽象成这种模型。和有监督学习不同，这里没有标签值作为监督信号，系统只会给算法执行的动作一个评分反馈，这种反馈一般还具有延迟性，当前的动作所产生的后果在未来才会完全得到，另外未来还具有随机性。

强化学习要解决的问题可以抽象成马尔可夫决策过程（Markov Decision Process，简称 MDP）。马尔可夫过程的特点是系统下一个时刻的状态由当前时刻的状态决定，与更早的时刻无关。与马尔可夫过程不同的是，在 MDP 中系智能体可以执行动作，从而改变自己和环

境的状态，并且得到惩罚或奖励。马尔可夫决策过程可以表示成一个五元组：

$$\{S, A, P_a, R_a, \gamma\}$$

其中 S 和 A 分别为状态和动作的集合。假设 t 时刻状态为 s_t，智能体执行动作 a，下一时刻进入状态 s_{t+1}。下一时刻的状态由当前状态以及当前采取的动作决定，是一个随机性变量，这一状态转移的概率为：

$$p_a\left(s, s^{'}\right) = p\left(s_{t+1} = s^{'} \middle| s_t = s, a_t = a\right)$$

这是当前状态为 s 时行动作 a，下一时刻进入状态 $s^{'}$ 的条件概率。这个公式表明下一时刻的状态与更早时刻的状态和动作无关，状态转换具有马尔可夫性。有一种特殊的状态叫做终止状态（也称为吸收状态），到达该状态之后不会再进入其他后续状态。对于围棋，终止状态是一局的结束。

执行动作之后，智能体会收到一个立即回报：

$$R_a\left(s, s^{'}\right)$$

立即回报和当前状态、当前采取的动作以及下一时刻进入的状态有关。在每个时刻 t，智能体选择一个动作 a_t 执行，之后进入下一状态 s_{t+1}，环境给出回报值。智能体从某一初始状态开始，每个时刻选择一个动作执行，然后进入下一个状态，得到一个回报，如此反复：

$$s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} s_2 \xrightarrow{a_2} s_3 \cdots$$

问题的核心是执行动作的策略，它可以抽象成一个函数 π，定义了在每种状态时要选择执行的动作。这个函数在状态 s 所选择的动作为：

$$a = \pi(s)$$

这是确定性策略。对于确定性策略，在每种状态下智能体要执行的动作是唯一的。另外还有随机性策略，智能体在一种状态下可以执行的动作有多种，策略函数给出的是执行每种动作的概率：

$$\pi(a|s) = p(a|s)$$

即按概率从各种动作中选择一种执行。策略只与当前所处的状态有关，于历史时间无关，在不同时刻对于同一个状态所执行的策略是相同的。

强化学习的目标是要达到我们的某种预期，当前执行动作的结果会影响系统后续的状态，因此需要确定动作在未来是否能够得到好的回报，这种回报具有延迟性。对于围棋，当前走的一步棋一般不会马上结束，但会影响后续的棋局，需要使得未来赢的概率最大化，而未来又具有随机性，这为确定一个正确的决策带来了困难。

选择策略的目标是按照这个策略执行后，在各个时刻的累计回报值最大化，即未来的预期回报。按照某一策略执行的累计回报定义为：

$$\sum_{t=0}^{+\infty} \gamma^t R_{a_t}\left(s_t, s_{t+1}\right)$$

这里假设状态转移概率以及每个时刻的回报是已知的，算法要寻找最佳策略来最大化上面的累计回报。

如果每次执行一个动作进入的下一个状态是确定的，则可以直接用上面的累计回报计算公式。如果执行完动作后进入的下一个状态是随机的，则需要计算各种情况的数学期望。为此定义状态价值函数的概念，它是在某个状态 s 下，按照策略 π 执行动作，累计回报的数学期望。状态价值函数的计算公式为：

$$V_\pi(s)=\sum_{s'} p_{\pi(s)}(s,s')\left(R_{\pi(s)}(s,s')+\gamma V_\pi(s')\right)$$

动作价值函数是智能体按照策略 π 执行，在状态 s 时执行具体的动作 a 后的预期回报，计算公式为：

$$Q_\pi(s,a)=\sum_{s'} p_a(s,s')\left(R_a(s,s')+\gamma V_\pi(s')\right)$$

除了指定初始状态与策略之外，还指定了在状态 s 时执行的动作 a。这个函数衡量的是给定某一策略，在某一状态时执行各种动作的价值。

给定一个策略，可以用动态规划算法计算它的状态价值函数，即策略评估（Policy Evaluation）。在每种状态下执行的动作有多种可能，需要对各个动作计算数学期望。按照定义，状态价值函数的计算公式为：

$$V_\pi(s)=\sum_a \pi(a|s)\sum_{s'} p_{\pi(s)}(s,s')\left(R_{\pi(s)}(s,s')+\gamma V_\pi(s')\right)$$

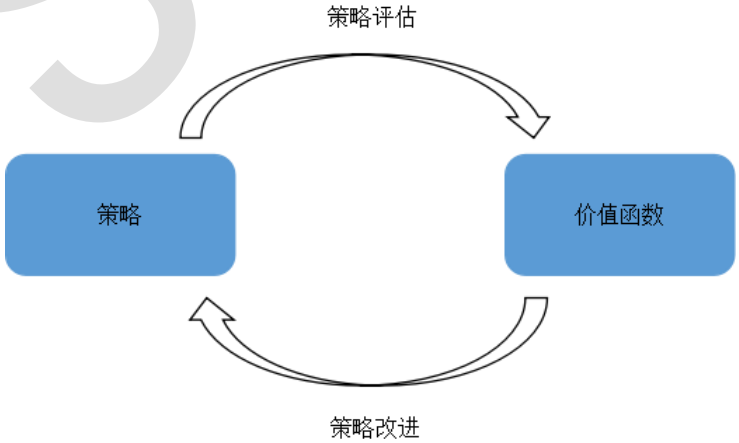
求解时使用迭代法，首先为所有状态的价值函数设置初始值，然后用公式更新所有状态的价值函数，第 k 次迭代时的更新公式为：

$$V_{k+1}(s)=\sum_a \pi(a|s)\sum_{s'} p_a(s,s')\left(R_a(s,s')+\gamma V_k(s')\right)$$

算法最后会收敛到真实的价值函数值。

策略评估的目的是为了得到更好的策略，即策略改进。策略改进通过按照某种规则对当前策略进行调整，得到更好的策略。

策略迭代是策略评估和策略改进的结合。从一个初始策略开始，不断的改进这个策略达到最优解。每次迭代时首先用策略估计一个策略的状态价值函数，然后根据策略改进方案调整该策略，再计算新策略的状态价值函数，如此反复直到收敛。这一过程如下图所示：



在策略迭代算法中，策略评估的计算量很大，需要多次扫描所有状态并不断的更新状态价值函数。实际上不需要知道状态价值函数的精确值也能迭代到最优策略，值迭代就是其中的一种方法。

根据贝尔曼最优化原理，如果一个策略是最优策略，整体最优的解局部一定也最优，因此最优策略可以被分解成两部分：从状态 s 到 $s^{'}$ 采用了最优动作，在状态 $s^{'}$ 是采用的策略也是最优的。根据这一原理，每次选择当前回报和未来回报之和最大的动作，价值迭代的更新公式为：

$$V_{k+1}\left(s\right)=\max_a \sum_{s^{'}} p_a\left(s,s^{'}\right)\left(R_a\left(s,s^{'}\right)+\gamma V_k\left(s^{'}\right)\right)$$

策略迭代算法和价值迭代算法虽然都可以得到理论上的最优解，但是它们的计算过程依赖于状态转移概率以及回报函数。对于很多应用场景，我们无法得到准确的状态模型和回报函数。因此前面介绍的这两种算法在实际问题中使用价值有限。

对于无法建立精确的环境模型的问题，我们只能根据一些状态、动作、回报值序列样本进行计算，估计出价值函数和最优策略。基本思想是按照某种策略尝试执行不同的动作，观察得到的回报，然后进行改进。

蒙特卡洛算法和时序差分算法是解决这这类问题的两种方法。蒙特卡洛算法是一种随机数值算法，它通过使用随机数来近似解决某些难以直接求解的问题。在强化学习中，蒙特卡洛算法可以根据样本得到状态价值函数以及动作价值函数的估计值，用于近似数学期望值。

在上面的例子中，样本是一些随机的点，在用于计算强化学习的价值函数时，样本是一些片段。在这里先定义片段（episode）的概念，它是从某一状态开始，执行一些动作，直到终止状态为止的一个完整的状态和动作序列，这类似于循环神经网络中的时间序列样本。蒙特卡洛算法从这些片段样本中学习，估算出状态价值函数和动作价值函数。实现时的做法非常简单：

按照一个策略执行，得到一个状态和回报序列，即片段。多次执行，得到多个片段。接下来根据这些片段样本估计出价值函数。

蒙特卡洛算法需要使用完整的片段进行计算，这在有些问题中是不现实的，尤其是对于没有终止状态的问题。时序差分算法（Temporal Difference learning，简称 TD 学习）在执行一个动作之后就进行价值函数估计，无需使用包括终止状态的完整片段，它结合了蒙特卡洛算法与动态规划算法的思想。与蒙特卡洛算法一样，TD 算法无需依赖状态转移概率，直接采样计算。TD 算法用贝尔曼方程估计价值函数的值，然后构造更新项。迭代更新公式为：

$$V\left(s\right)=V\left(s\right)+\alpha\left(R+\gamma V\left(s^{'}\right)-V\left(s\right)\right)$$

算法用当前动作的立即回报值与下一状态当前的状态价值函数估计值之和构造更新项，更新本状态的价值函数：

$$R+\gamma V\left(s^{'}\right)$$

在上式中没有使用状态转移概率，而是和蒙特卡洛算法一样随机产生一些样本来进行计算，因此称为无模型的算法。用于估计状态价值函数时，算法的输入为策略，输出为该策略的状态值函数。

Sarsa 算法用于估计给定策略下的动作价值函数，同样是每次执行一个动作之后就进行更新。它的迭代更新公式为：

$$Q\left(s,a\right)=Q\left(s,a\right)+\alpha\left(R+\gamma Q\left(s^{'},a^{'}\right)-Q\left(s,a\right)\right)$$

由于更新值的构造使用了$\{s,a,R,s',a'\}$这 5 个变量，因此被命名为 Sarsa 算法。根据所有状态-动作对的价值函数可以得到最优策略。

Q 学习算法估计每个动作价值函数的最大值，通过迭代可以直接找到价值函数的极值，从而确定最优策略，类似于价值迭代算法的思想。

$$Q(s,a)=Q(s,a)+\alpha\left(R+\gamma\max_{a'}Q(s',a')-Q(s,a)\right)$$

实现时需要根据当前的动作价值函数的估计值为每个状态选择一个动作来执行，这里有两种方案。第一种方案是随机选择一个动作，这称为探索（exploration）。第二种方案是根据当前的动作函数值选择一个价值最大的动作执行：

$$a=\max_{a'}Q(s,a')$$

这称为利用（exploitation）。第三种方案是二前两者的结合，即ε-贪心策略。执行完动作之后，进入状态s'，然后寻找状态s'下所有动作的价值函数的极大值，构造更新项。算法最终会收敛到动作价值函数的最优值。用于预测时，在每个状态下选择函数值最大的动作执行，这就是最优策略，具体实现时同样可以采用ε-贪心策略。

前面介绍的算法只能用于状态和动作的集合是有限的离散基且状态和动作数量较少的情况，状态和动作需要人工预先设计。实际应用中的场景可能会很复杂，很难定义出离散的状态；即使能够定义，数量也非常大，无法用数组存储。用一个函数来逼近价值函数或策略函数成为解决这个问题的一种思路，函数的输入是原始的状态数据，函数的输出是价值函数值或策略函数值。

在有监督学习中，我们用神经网络来实现分类或回归函数，同样的，也可以用神经网络可来拟合强化学习中的价值函数和策略函数，这就是深度强化学习的基本思想。在这里，神经网络被用于从原始数据如图像中直接预测出函数值。

在 Q 学习中用表格存储动作价值函数的值，如果状态和动作太多这个表将非常大，在某些应用中也无法列举出所有的状态形成有限的状态集合。解决这个问题的方法是用一个函数来近似价值函数，深度 Q 学习用神经网络来近似动作价值函数。网络的输入是状态，输出是各种动作的价值函数值。下面用一个例子进行说明。算法要实现自动驾驶，将当前场景的图像作为状态，神经网络的输入是这种图像，输出是每个动作对应的 Q 函数值，这里的动作是左转，右转，刹车，加油门等。显然，神经网络输出层的尺寸与动作数相等。

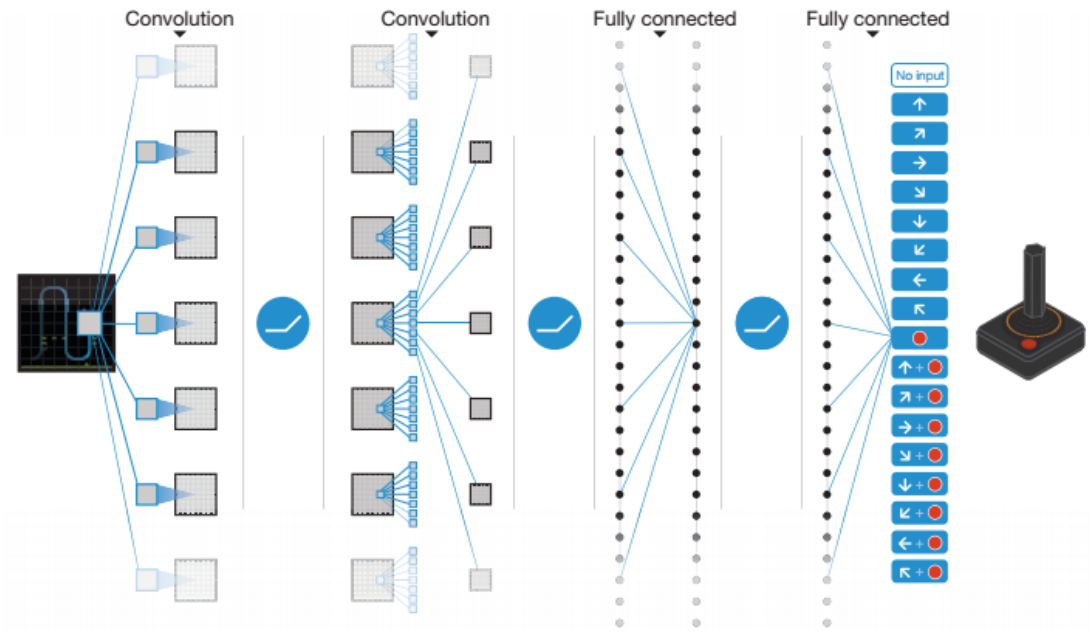
DeepMind 提出了一种用深度 Q 解决 Atari 游戏的方法，使用卷积神经网络拟合 Q 函数，称为深度 Q 网络（简称 DQN）。网络的输入为经过处理后游戏图像画面，原始的画面是 210x160 的彩色图像，每个像素的值为[0, 255]之间的整数，所有可能的状态数为：

$$256^{210\times160\times3}$$

这个规模的矩阵无法直接用表格存储。网络的输出值是在输入状态下执行每个动作的 Q 函数值，在这里有 18 个值，代表游戏中的 18 种动作。神经网络用于近似最优 Q 函数：

$$Q(s,a,\theta)\approx Q_{\pi}(s,a)$$

其中θ是网络的参数。网络结构如下图所示：



关键问题是训练样本标签值的与损失函数的设计。这里的目标是逼近最优策略的 Q 函数值，因此可以采用 Q 学习的做法。损失函数用神经网络的输出值与 Q 学习每次迭代时的更新值构造，定义为：

$$L(\theta)=\mathrm{E}\left(\left(R+\gamma\max_{a'}\left(s',a',\theta\right)-Q(s,a,\theta)\right)^2\right)$$

在这里采用了欧氏距离损失，是神经网络的输出值与 Q 函数估计值之间的误差，与 Q 学习中的更新项相同。另一个问题是如何得到训练样本，和 Q 学习类似，可以通过执行动作来生成样本。实现时，用当前的神经网络进行预测，得到所有动作的价值函数，然后按照策略选择一个动作执行，得到下一个状态以及回报值，以此作为训练样本。

这里还使用了经验回放（Experience Replay）技术。神经网络要求训练样本之间独立同分布，而 Atari 游戏的训练样本是一个时间序列，前后具有相关性。解决这个问题的做法是经验池，将样本存储在一个集合中，然后从中随机采样得到每次迭代所用的训练样本。

深度 Q 学习基于动作价值函数，它用神经网络拟合 Q 函数的最优值，通过函数值间接得到最优策略。如果动作集合是连续的或维数很高，这种方法将面临问题。例如算法要控制机器人在 x 和 y 方向上移动，每个方向上的移动距离是[-1.-, +1.0]之间的实数，移动距离无法穷举出来离散化成动作集合，因此无法使用基于价值函数的方法。此时可以让神经网络根据输入的状态直接输出 x 和 y 方向的移动距离，从而解决连续性动作问题。

策略梯度（Policy Gradient）算法[6]是这种思想的典型代表，策略函数网络的输入是图像之类的原始数据。策略函数根据这个输入状态直接预测出要执行的动作：

$$a=\pi\left(s;\theta\right)$$

其中 θ 是神经网络的参数。对于随机性策略，神经网络的输出是执行每种动作的概率值：

$$\pi\left(a|s;\theta\right)=p\left(a|s;\theta\right)$$

这是一种更为端到端的方法，神经网络的映射定义了在给定状态的条件下执行每种动作的概率，根据这些概率值进行采样可以得到要执行的动作。对于离散的动作，神经网络的输出层神经元数量等于动作数，输出值为执行每个动作的概率。对于连续型动作，神经网络的

输出值为高斯分布的均值和方差，动作服从此分布。

这里的关键问题是构造训练样本和优化目标函数，在这两个问题解决之后剩下的就是标准的神经网络训练过程。在样本生成问题上，策略梯度算法采用的做法和 DQN 类似，用神经网络当前的参数对输入状态进行预测，根据网络的输出结果确定出要执行的动作，接下来执行这个动作，得到训练样本，并根据反馈结果调整网络的参数。如果最后导致负面的回报，则更新网络的参数使得在面临这种输入时执行此动作的概率降低；否则加大这个动作的执行概率。策略梯度算法在优化目标上和深度 Q 学习不同，深度 Q 学习是逼近最优策略的 Q 函数，而策略梯度算法是通过最大化回报而逼近最优策略。

推荐文章

往期文章汇总

[31] 理解计算：从√2到AlphaGo ——第2季 神经计算的历史背景 SIGAI 2018.7.4

本文为SIGAI原创