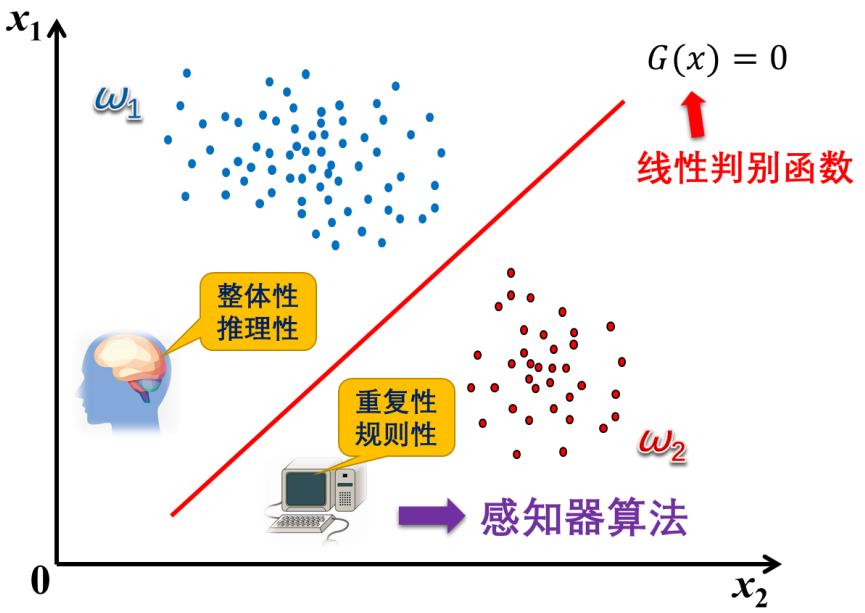
3.2 感知器算法

我们前面介绍了最简单的一种分类器形式——线性分类器，它试由线性判别函数及相应的分类决策规则构成的。

那这个线性判别函数是如何得到的呢？如果有一个模式识别问题，我们怎么才能设计出一个合适的线性分类器，使它能对问题中的样本正确地分类呢？这就是牵涉到线性分类器训练的问题。



就像我们前面尝试过的“模板匹配”方法一样，看起来不需要训练的模式识别算法，其实背后隐藏的都是需要利用人类的经验和观察力来提供分类决策依据，这不仅无法处理大量数据和高维度数据的问题，也无法应对问题的变化，针对各不相同的具体模式识别问题随机应变给出不同的分类器。所以，要得到一个最合适的分类器，需要把样本集中的数据一个个拿来处理，通过分析和计算，得到一个最佳的分类决策规则，才能够适应性地解决真正的模式识别问题。

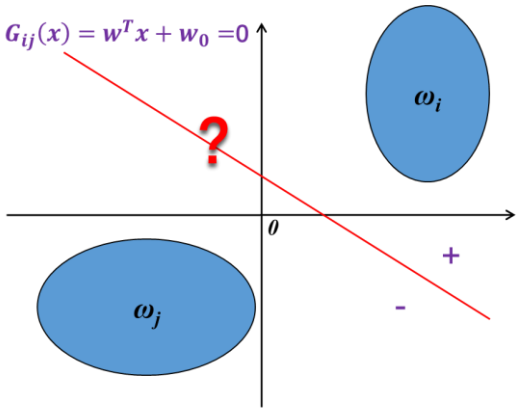
人类思维的优势在于逻辑性和整体直觉，人类不擅长的恰恰是重复性地去一个个处理大量的数据。而计算机的特长正是这种枯燥而机械的数据处理，只要我们给定明确的逻辑处理规则、计算公式和执行处理的详细步骤，计算机就可以勤勤恳恳地一步步完成数据处理任务，从而解决模式识别中的分类器训练问题。

所以，模式识别中任何分类器的设计，靠人工给定结果都是不靠谱的。而应该是由人类给定一个分类器的形式和架构，再设计一套有效的算法，让计算机能够通过执行这套算法自行寻找到最优的分类器参数，这也是从人的角度来说的分类器训练与从计算机角度来说的分类器学习含义完全相等的原因。

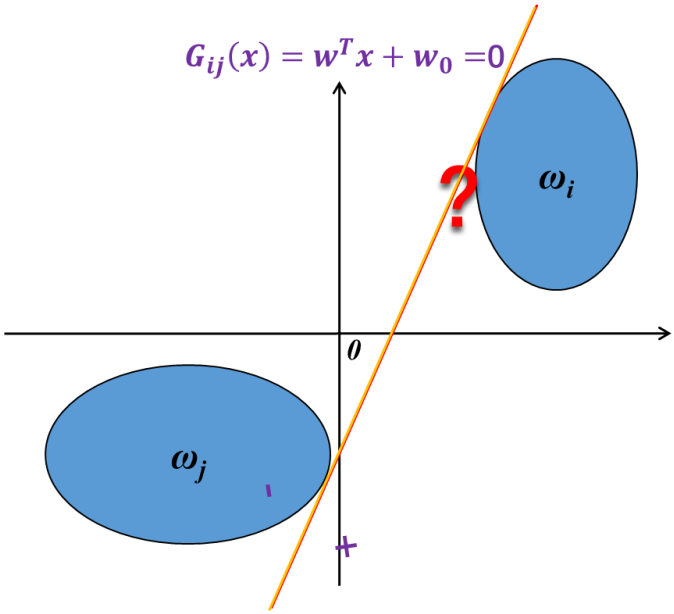
3.2.1 线性分类器训练的一般思路

在介绍感知器训练算法之前，我们先来看一看学习分类器训练的一般思路。如果给定了一个包含两类样本的样本集，那么训练一个线性分类器，就是希望计算机能够依据样本集中的数据，自行寻找到一条分类决策边界，也就是确定能把

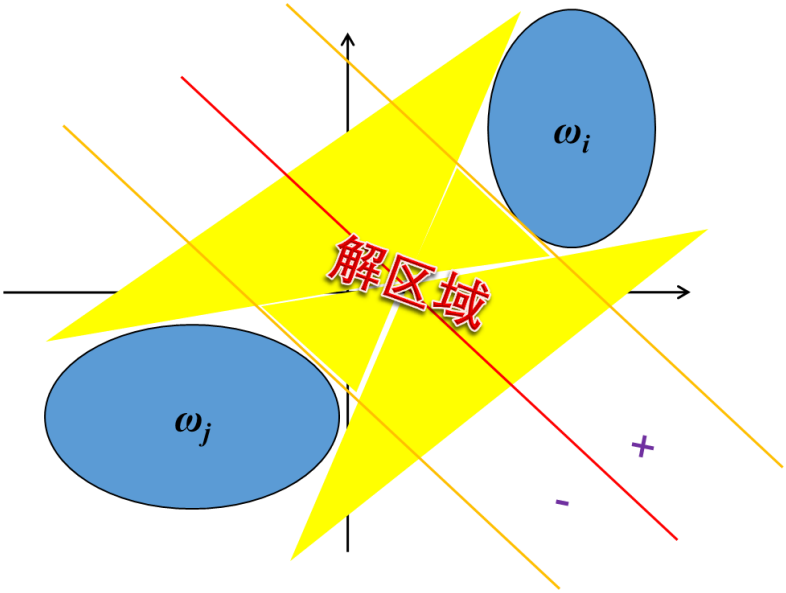
两类样本正确分类的权向量 w 和偏置量 w_0。



给定训练集 → w =?, w_0 =?



我们再来看看如果给定了训练集，那么最终能求到的线性判别函数会在什么位置呢？显然，$G_{ij}(x)=0$ 的分类决策边界，可以旋转，直到与两类样本的分布区域相切，这代表不同的权向量 w；还可以平移，知道两类样本的分布区域相切，这代表不同的偏置量 w_0。因此，最终的解不是唯一解，而是位于一个区域内，称为解区域。



那么，求解一个线性分类器的过程，就是按照某种准则，找到解区域中一个最优解的过程，其一般的思路是：

1. 首先设定一个标量的准则函数（Criterion Function），使其值能够代表解的优劣程度，准则函数值越小，说明解越符合要求，越好；

2. 通过寻找准则函数的极小值，就能找到最优的一个解，使准则函数 取得极小值的增广权向量 ，就是最优解；

在求解线性分类器的过程中，既要求权向量 w，又要求偏置量 w_0，比较麻烦，可不可以进行一些规范，简化数据和计算的表达形式呢？

首先，我们看一下要求的线性判别函数，它的形式是

$$G_{ij}(x)=wTx+w_0$$

其中 x 是 n 维的样本特征向量，w 是 n 维的权向量，w_0 是偏置量。为将 w_0 统一纳入求解过程，对 x 和 w 都进行增广，变为 n+1 维。此时，判别函数也就变为了

$$\begin{cases} G_{ij}(\boldsymbol{x}^{(1)})=\boldsymbol{w}^T\boldsymbol{x}^{(1)}>0 \\ G_{ij}(\boldsymbol{x}^{(2)})=\boldsymbol{w}^T\boldsymbol{x}^{(2)}>0 \\ \vdots \\ G_{ij}(\boldsymbol{x}^{(l_i)})=\boldsymbol{w}^T\boldsymbol{x}^{(l_i)}>0 \\ G_{ij}(\boldsymbol{y}^{(1)})=\boldsymbol{w}^T\boldsymbol{y}^{(1)}<0 \\ G_{ij}(\boldsymbol{y}^{(2)})=\boldsymbol{w}^T\boldsymbol{y}^{(2)}<0 \\ \vdots \\ G_{ij}(\boldsymbol{y}^{(l_j)})=\boldsymbol{w}^T\boldsymbol{y}^{(l_j)}<0 \end{cases}$$

我们再来看求解的目标，也就是要求两类中所有样本都能够被正确地分类，即以下不等式方程组成立。我们可以看到，这些不等式中，对于第 i 类样本是判别函数值大于 0，对于第 j 类样本是判别函数值小于 0。大于小于混在一起，也不方便处理。我们可以把位于分类决策边界负侧的第 j 类的所有样本的特征向量乘以-1，则求解目标就统一变为了大于 0 的不等式方程组形式，即要求找到满足 Gij(x)=wTx>0 的判别函数。

3.2.2 感知器算法的原理

要学习感知器算法，需要先了解感知器模型。

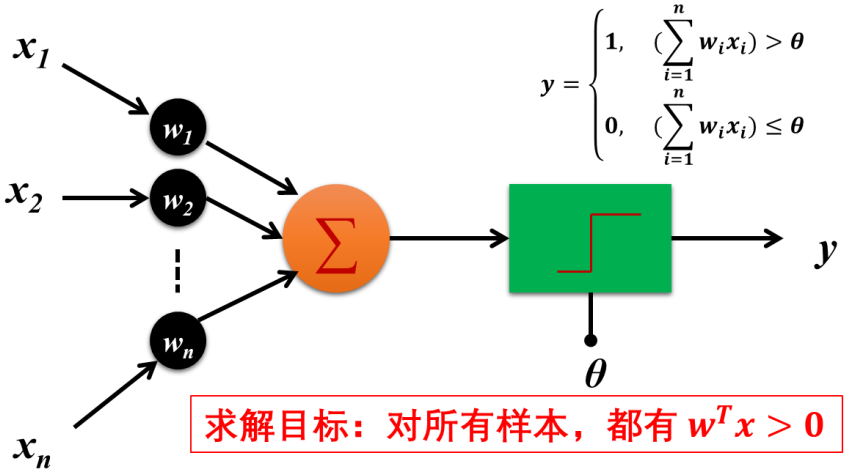
感知器模型是一种神经元模型。而神经元模型是根据神经生理学的研究，所提出的模拟神经元功能的数学模型。你可能了解过，生理解剖发现，神经元是一种有输入输出的细胞，它的输入端称为“树突”，可以输入神经刺激；输出端称为“轴突”，可以传导出神经冲动。一个神经元的轴突连接到其他神经元的树突，这种神经元间的相互连接，就构成了一个神经元网络，可以实现复杂的信息处理功能。一个神经元一般具有多个树突，树突所搜集到的输入信号，经过神经元的集中处理，如果达到一定的激发阈值，就会激活轴突的输出。

感知器（Perceptron）作为一种神经元模型，是美国计算机科学家罗森布拉特（F. Rosenblatt）在 1957 年提出的。它具有多路的输入和单路的输出，将所有输入信号加权求和后与一个阈值相比较，如果大于阈值，则神经元输出为 1，小于等于阈值，神经元输出为 0。因此，感知器是一种非常简单的神经元模型，它没有反馈和内部状态，只是依靠输入信号是否超过阈值来决定是否激活神经元的输出。

很明显，如果把感知器的输入信号看作是一个待识别样本的特征向量，那么感知器的数学模型就构成了一个典型的线性分类器，可以做出非常明确的二分类决策。因此，感知器模型不仅是一个仿生神经元模型，也是一个可实现的线性分类器模型。如果我们能够通过样本集，使感知器学习到输入权重值和输出的阈值，我们也就可以针对具体的模式识别问题，训练出一个可以实现线性分类的分类器。

感知器模型虽然简单，但它在模拟人脑生理特性来实现人工智能方面具有开创性贡献，为此，IEEE 的计算智能学会（Computational Intelligence Society）于 2004 年专门设立了罗森布拉特奖，用于奖励那些在仿生智能计算领域做出杰出贡献的科学家，获奖人中包括人工智能领域中鼎鼎大名的支持向量机算法提出者 Vapnik，和深度学习算法的开创人 Hinton。

感知器是一个通过输入加权和与阈值的比较来觉得是否激活输出的神经元模型，这是一个线性分类器，输入的权构成了线性分类决策边界的权向量，激活输出的阈值 θ 就是分类决策边界的偏置量 w_0。所以，经过数据规范化以后，得到了统一的求解目标，即$\boldsymbol{w^T x > 0}$



感知器算法设定准则函数的依据很简单，就是最终分类器要能正确分类所有的样本，所以，感知器算法的准则函数 J 设定为所有错分样本的判别函数值之和，再乘以-1.因此，只要存在错分样本，准则函数值就是大于 0 的，只有当所有样本都正确地被分类了，准则函数才能取得极小值 0.

如何求解呢？感知器算法采用了数值优化中经典的梯度下降法，即从一个任意的初始权向量 w_0 出发，沿准则函数值下降最快的方向，也就是负梯度方向对权向量进行一步步修正，直到获得全局最优解为止。即第 k+1 步是在第 k 步获得的权向量基础上进行递推，得到第 k+1 步的权向量。其中 ρ 是每一次递推的调整步长。

我们把感知器算法的准则函数定义代入，就可以求得准则函数第 k 步时对 w(k)的梯度，正好是所有错分样本的和乘以-1。因此感知器算法的权向量递推公式为$\boldsymbol{w(k+1) = w(k) + \rho(k+1)\sum_{x\in X_0} x}$，

即每一步把当前被错分的样本加起来，在调整步长的控制下，去修正权向量。要特别注意的是，修正的方向是“+”，这是因为负梯度和准则函数梯度中的负号相抵消的原因。

感知器算法的具体步骤为：

a、 设定初始权向量 $\boldsymbol{w}(0)$，$k=0$；

b、 对训练样本集的所有规范化增广特征向量进行分类，将分类错误的样本（即不满足 $G_{ij}(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} > 0$ 的样本）放入集合 X_0 中；

c、 修正权向量：$\boldsymbol{w}(k+1) = \boldsymbol{w}(k) + \rho(k+1)\sum_{\boldsymbol{x}\in X_0} \boldsymbol{x}$

d、 $k = k+1$，返回步骤 b，直至所有样本都能被正确分类为止。

也可采取单样本修正的算法，步骤为：

a、 设定初始权向量 $\boldsymbol{w}(0)$，$k=0$；

b、 从训练样本集中顺序抽取一个样本，将其规范化增广特征向量 $\boldsymbol{x}$ 代入到判别函数中计算；

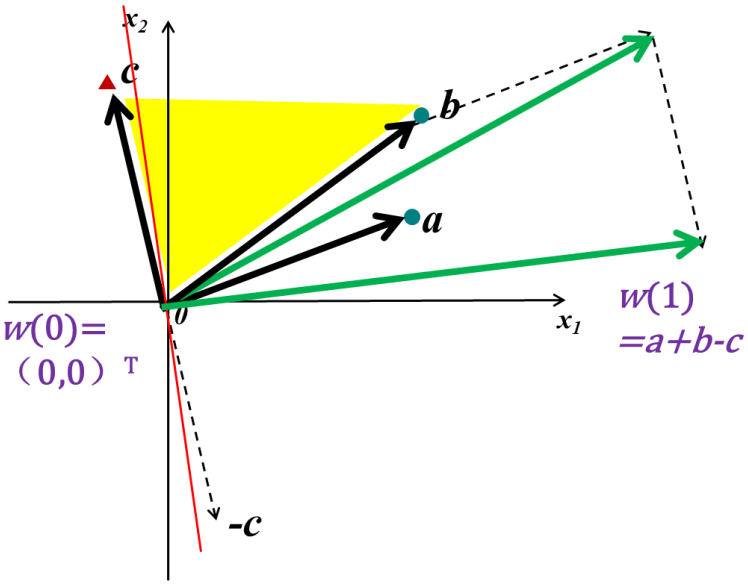
c、 若分类正确（即 $G_{ij}(\boldsymbol{x})=\boldsymbol{w}^T\boldsymbol{x}>0$）返回到步骤 b，抽取下一个样本；

d、 若分类错误（即不满足 $G_{ij}(\boldsymbol{x})=\boldsymbol{w}^T\boldsymbol{x}>0$），修正权向量：

$$\boldsymbol{w}(k+1)=\boldsymbol{w}(k)+\rho(k+1)\boldsymbol{x}$$

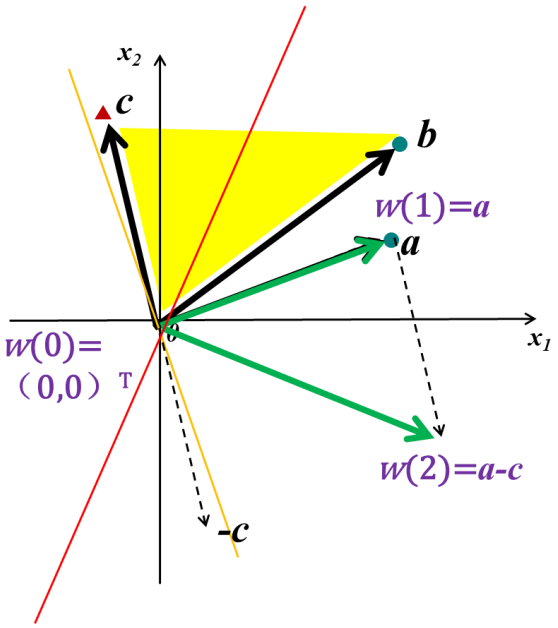
e、 返回到步骤 b，抽取下一个样本；直至训练样本集中所有样本均被正确分类。

我们来直观地看一下感知器算法的过程。



如果在二维特征空间中有 3 个样本，分别属于两个类。如果我们考虑训练得到一个偏置量为 0 的线性分类器，显然分类决策边界会通过原点，而解区域是如图这样的黄色区域。假设我们取初始权向量为 0 向量，那么显然所有的样本都将被错分。所以，如果取调整补充为 1 的话，下一步的 w(1)，就等于 0 向量加上所有样本的向量和。特别要注意的是，为什么样本 c 不是直接与 a、b 相加呢？因为我们要对负侧的样本进行规范化，在分类器训练的过程中，实际使用的是负侧样本乘以-1 以后的向量。可以看到，感知器算法确实成功地求得了线性分类器的解，可以把所有样本正确地分类。

感知器算法还有另一种变形，它不是在每一步递推时将所有被错分的样本都找出来，再将其向量和用于修正权向量，而是每次处理 1 个样本，如果发现分类错误，就用这一个被错分的样本来修正权向量。



假设我们仍然取初始权向量为 0 向量，那么开始依次处理 abc 三个样本。因为初始权向量是 0 向量，所以 a 是被错分的，修正的结果 w(1)就是 a，此时得到了一个线性判别函数。在用这个线性判别函数处理样本 b 时，b 是被正确分类的，所以权向量无需修正。接着在处理样本 c 时，样本 c 是被当前的判别函数错分的，因此需要用 c 来修正权向量，得到 w（2）。此时的 w(2)能够正确分类所有样本，所以就是最优解。

我们可以发现这种算法流程求得的最优解，与前一种算法流程求得的最优解并不相同。那么，如何评判哪一个解更优呢？感知器算法真的能找到最优的权向量吗？

事实上，这个两个解都是最优解，因为感知器算法进行求解的准则函数只与是否能够将所有样本都正确分类有关。只要没有错分样本，最终的解落入解区域内，那么，从感知器算法的准则函数来说，就都取得了极小值，得到了算法的最优解。而在解区域中是否还能优中选优，感知器算法就无能为力了。

因此，一个寻优的过程中，评价方式所觉得的目标设定高低，直接决定了我们最终能得到什么样的结果。之所以我们经常强调要立鸿鹄志、志存高远，与此是完全相同的道理。

3.2.3 感知器算法的学习速率

感知器算法通过设定所有错分样本的判别函数值之和的负值，推导除了权向量调整的递推公式，就是这两种形式。

$$w(k+1)=w(k)+\rho(k+1)\sum_{x\in X_0}x$$

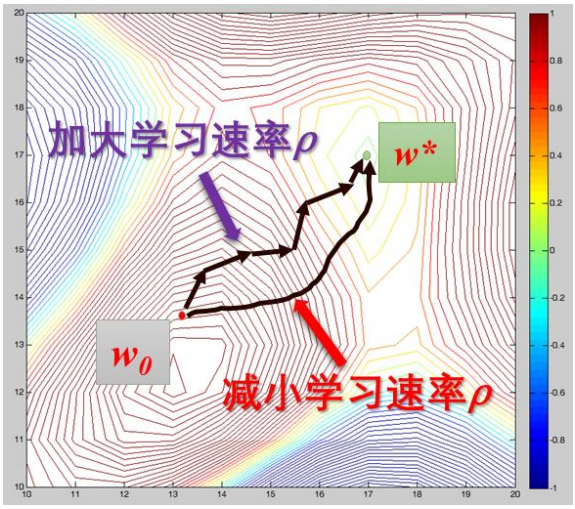
or

$$w(k+1)=w(k)+\rho(k+1)x, x\in X_0$$

学习速率

在递推公式中，ρ 是调整步长，它的实质是梯度下降法中，每一步向当前负梯度方向调整的大小因子，也就是分类器学习到最优权向量的速度。因此，当我们在线性分类器的学习过程中使用梯度下降法时，调整步长 ρ 也称为“学习速率”。

ρ 的取值越大，求解的速度越快，但求解路径越不光滑，求解的精度越差，容易过冲甚至振荡； ρ 的取值越小，求解的速度越慢，但求解精度越高。因此，只要使用梯度下降法来训练分类器，其求解精度和速度之间是存在矛盾的。



$\rho(k+1)$ 的选择

在递推公式中，ρ 是调整步长，它的实质是梯度下降法中，每一步向当前负梯度方向调整的大小因子，也就是分类器学习到最优权向量的速度。因此，当我们在线性分类器的学习过程中使用梯度下降法时，调整步长 ρ 也称为“学习速率”。

ρ 的取值越大，求解的速度越快，但求解路径越不光滑，求解的精度越差，容易过冲甚至振荡； ρ 的取值越小，求解的速度越慢，但求解精度越高。因此，只要使用梯度下降法来训练分类器，其求解精度和速度之间是存在矛盾的。为在算法求解的速度和精度之间取得平衡，感知器算法中的学习速率有各种不同的设定方法：

- **固定值**

  即 $\rho(k+1)$ 选择固定的非负数；

- **绝对修正**

  在单样本修正算法中，为保证分类错误的样本在对权向量进行一次修正后能正确分类，需要满足

$$\boldsymbol{w}^T(k+1)\boldsymbol{x}>0$$

  代入递推修正公式 $\boldsymbol{w}(k+1)=\boldsymbol{w}(k)+\rho(k+1)\boldsymbol{x}$，得

$$\boldsymbol{w}^T(k+1)\boldsymbol{x}=\boldsymbol{w}^T(k)\boldsymbol{x}+\rho(k+1)\boldsymbol{x}^T\boldsymbol{x}>0$$

  即

$$\rho(k+1)>\frac{\left|\boldsymbol{w}^T(k)\boldsymbol{x}\right|}{\boldsymbol{x}^T\boldsymbol{x}}$$

  满足该条件的 $\rho(k+1)$ 称为绝对修正因子。

- **部分修正**

若取

$$\rho(k+1)=\lambda\frac{\left|\boldsymbol{w}^T(k)\boldsymbol{x}\right|}{\boldsymbol{x}^T\boldsymbol{x}},\quad 0<\lambda\le 2$$

则称为部分修正。

- **变步长法**

可以取按照某种规律逐步减少的 $\rho(k+1)$，使得算法开始时收敛较快，接近最优解时收敛速度变慢，以提高求解的精度。比较常用的变步长法是取

$$\rho(k+1)=\lambda\frac{1}{k},\quad \lambda>0$$

- **最优步长法**

在每一步时，通过求准则函数

$$J(\boldsymbol{w}(k+1))$$

对于不同的 $\rho(k+1)$ 可以取得的最小值，来确定最优的 $\rho(k+1)$。

该方法的问题在于：相比采用小步长带来的递推次数增加，每步求最优步长会带来更大的计算量。

感知器算法是历史上首个有效的有监督学习算法，虽然今天看起来过于简单，但罗森布拉特在 1957 年首次提出时确实引起了人工智能界的轰动，并且对后续的人工智能、特别是基于神经网络的机器学习算法产生了深远的影响。

至于为什么略显粗糙的一个算法会有这么重要的意义，我们在之后的课程中将为您做深入的分析！

3.2.4 感知器算法的深入分析

感知器算法是一种模拟神经元功能的机器学习算法。它采用了能够实现一个线性分类器的感知器模型，并通过设定错分样本判别函数和的准则函数，采用梯度下降法得到了非常简单的递推式算法流程，能够对线性可分问题找到正确分类所有特别的解。

感知器算法虽然看起来很简单，并且也只能找到没有错分的线性分类器解，不能再解区域中继续寻找最优的解，但是，它仍然在人工智能的发展历史上占据了重要的地位，说它是当今热火朝天的“深度学习”的鼻祖，也完全当之无愧。究其原因，是因为感知器算法在简单的模型的求解过程中，已经包含了以机器学习中的许多重要思想和算法原理，为后世的人工智能发展奠定了基础。下面，我们就从学习规则、收敛性、优化方法等方面来做一下分析。

最初的人工智能研究，重点在于通过“设计”使机器具有类似人的智能，从而能够处理推理、概念理解、抽象思维甚至创造等任务。一直到上世纪 50 年代初，人们才意识到，人工智能最重要的能力是“学习”的能力，就是计算机能够自主地去学习知识，学习处理事务的能力。

学习的基本规则有多种，总体上可以分为有监督学习和无监督学习两类。

有监督学习的依据就是学习到的模型的输出与真实输出之间的误差，对模型参数的调整方向，就是去减少这种误差，从而使学习到的模型逼近真实的模型。如果要调整的模型参数就是输入加权求和的权向量 w，那么通用的误差反馈学习

规则就是：

$$\boldsymbol{w(k+1)=w(k)+\rho(k+1)r\big(y(x)-\hat{y}(x)\big)x}$$

其中，y(x)是样本 x 对应的真实输出，y^(x)则是当前模型的实际输出。r 是输出误差的一个函数，称为学习信号，代表误差如何反馈影响权向量的调整。对于二分类问题，如果考虑输出就是分类决策结果，则输出就只有 0，1 两个值。感知器算法直接把输出误差值作为学习信号，而样本数据规范化后，分类正确时，输出误差位 0；分类错误时，输出误差位 1，因为此时 y(x)应该永远输出 1。所以，感知器算法的权向量递推调整算法就是：

$$\boldsymbol{w(k+1)=w(k)+\rho(k+1)x, x\in X_0}$$

和前面我们用梯度法得到的算法时一致的。

感知器算法： $\boldsymbol{r\big(y(x),\hat{y}(x)\big)=y(x)-\hat{y}(x)}$

感知器算法正因为是首个采用误差反馈学习规则来是实现的机器学习算法，它模拟人类学习的试错过程，不是希望计算机能够一次学会复杂的逻辑推理，而是通过一个形式确定但是参数不确定的模型来逐步逼近真实存在的客观规律，所以对后世产生了深远的影响。一直到现在热门的深度学习，从根本上仍然是以有监督的误差反馈学习为核心，当然，其分类器的模型和学习信号的设计，又比感知器有了更多的进步。

以上只是从误差反馈学习的角度对感知器算法进行了粗略的分析。如果要从数学上更严密地说明误差反馈学习规则，就必须得提到损失函数（Loss Function）。

损失函数针对一个机器学习算法，如果学习到一个输入输出映射模型，那么对一个样本输入，模型得到的输出与该样本对应的真实输出之间的差别，就称为损失函数。损失函数是对单个样本来说的，如果对于具有一定分布的样本整体的损失函数的数学期望，就称为代价函数（Cost Function）。而机器学习的目标，就是通过不断优化模型的参数，来找到最小的代价函数。

$$L(y,\hat{y})=\begin{cases}1, y=\hat{y}\\ 0, y\neq\hat{y}\end{cases}$$ 连续? 可导? ➡ 代理损失函数

对于二分类问题，如果我们只考虑分类结果是否正确，那么损失函数就可以设定为只有 0，1 两个值，称为 0-1 损失函数。有了损失函数和代价函数，我们就可以用最常用的梯度下降法，来求解代价函数最小的分类器最优解。

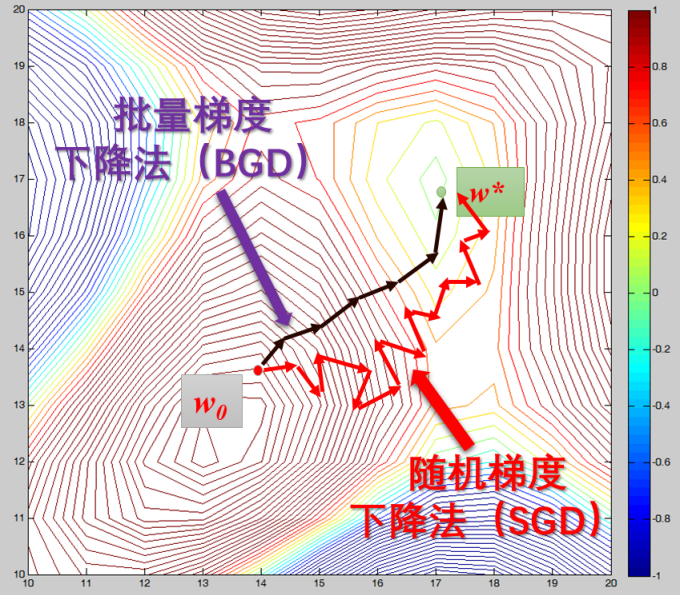
当然，0-1 损失函数虽然简单，但是无法反映能够正确分类所有样本的解中哪个解更好，更重要的是，0-1 损失是阶跃函数形式，在使用梯度下降法是无法求导。所以，我们也常常使用能够包容损失函数的结果，又能连续可求导地反映每一个解的优劣的损失函数，称为 0-1 损失函数的代理损失函数，您可以课后阅读课程提供的参考资料，看看都有哪些代理损失函数，以及它们都被用在了哪些算法中。

误差反馈学习一般采用梯度下降法来逐步调整模型参数，感知器算法也是这样。但是，在梯度下降的优化方法里，也有两种不同的算法思路，我们一起来看一看。

代价函数取得极小值，是我们基于误差反馈学习规则的模型参数优化目标。当采用梯度下降法时，应当是找到当前代价函数对模型参数的负梯度方向，也就

是代价函数值下降最快的方向来调整模型参数。注意：此时需要将所有样本集中的样本对应的损失函数值都计算出来，才能得到代价函数值，也才能求得代价函数对模型参数的负梯度，这就称为“批量梯度法（BGD）”。

批量梯度法的每一次对模型参数的调整，都朝向代价函数值减小的方向，因此，寻优路径相对式比较平滑的。但是当样本集中的样本数量非常多时，每一步模型参数的调整，都要将所有样本的损失函数计算一遍，整个算法的计算量是非常巨大的。由此，可以采用一个简化的方法，就是每次只对随机抽取的一个样本计算损失函数值，并且按损失函数对模型参数的负梯度方向调整模型参数，这种方法称为“随机梯度法（SGD）”。

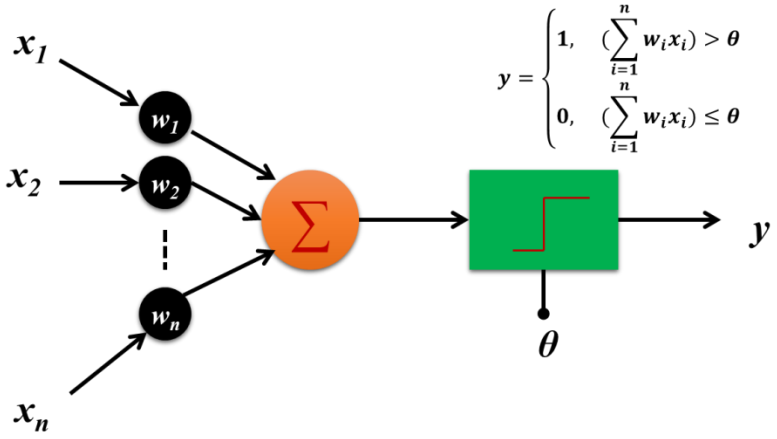


显然，随机梯度法在每一次模型参数调整时，不一定会朝着整体代价函数值减少的方向去前进，而是可能随机振荡，甚至最终不会确保收敛到极值点。但是，可以证明，随机梯度下降法会以很高的概率收敛于极值点，更重要的是，随机梯度法的计算量比批量梯度法的计算量大大地减少了，使得学习速度加快了很多，还可以边学习边加入新的样本，实现在线学习。这也是随机梯度法得到了非常广泛使用的原因，一直到现在的深度学习算法中，随机梯度下降法也是进行模型训练不可或缺的算法之一。

我们回顾一下感知器算法的两种权向量更新方法，有没有发现其中一个是批量梯度法的结果，一个是随机梯度法的结果？所以。为什么我们说感知器算法中蕴含了许多非常重要的原理和方法，由此也可见一斑。

感知器算法既然有这么多开创性的贡献，并且在上个世纪末就被罗森布拉特提出来了，为什么人工智能研究没有追随其思想并发扬光大，而是在其几十年之后才开始蓬勃发展呢？这就不能不提到它的缺陷，而且是致命的缺陷了。

我们回顾一下感知器模型，它的显著特征是一个线性分类器，它的最大缺陷就是无法解决线性不可分问题。



在 1957 年罗森布拉特（他是一名实验心理学家）提出感知器算法之后，引起了巨大的轰动，他本人也极力推广他的模型，并于 1960 年开发除了采用感知器算法的专用计算机 Mark I，能够实现一些简单图形和字母的识别。

随后，大量的商业投资涌入以感知器为核心的人工智能研究，社会公众也将罗森布拉特看作是将改变世界的天才。这引起了一些长期从事人工智能研究的学者的警惕。1969 年，这些学者中的重量级人物马文 •明斯基（Marvin Lee Minsky）和西蒙 • 派珀特博士（Seymour Papert，MIT 媒体实验室和 LOGO 编程语言的创立者）出版了一本名为《感知器》的书，用异或问题这一线性分类器典型的线性不可分问题，强烈地质疑了感知器的合理性，并连带着把通过人工神经元网络来实现人工智能的道路（称为连接主义）完全封堵，直接导致人工智能的研究进入寒冬。

当时看来，不能解决线性不可分问题确实是感知器算法的致命缺陷，甚至在明斯基等人的打击下罗森布拉特真的被“致了命”，于 1971 年 43 岁生日的当天，溺水而亡。但是，如果我们放大科技发展史的尺度，我们会发现明斯基等人对神经网络的否定，与后来神经网络算法特别是深度学习算法的成功并不相符，也就是，大师做了错误的判断。

回顾这段历史，能给我们的启示很多。首先当然是“满招损，谦受益”，如果罗森布拉特当时不是那么高调，社会不是那么捧杀，资本不是那么疯狂，踏踏实实地研究，神经网络技术可能早就得到了更大的发展，面对当前已经有些疯狂的对人工智能的巨大社会投入，也需要有更清醒的头脑。第二，权威也有看走眼的时候，不迷信人，只相信科学真理，如果认为方向正确，就持之以恒研究下去，终究也会有拨乱反正的一天。