二、Learning to Answer Yes/No 二元分类

2.1 Perceptron Hypothesis Set 感知器的假设空间

还是银行发信用卡的例子，银行可能掌握了用户的各种属性，如年龄，年薪，工作年限，负债情况等等，这些属性可以作为上面提到的样本输入$\boldsymbol{X = \{x1, x2, x3, x4\}^T}$的向量属性值。但是这样还是无法进行机器学习，因为我们还需要另一个输入，即假设空间$\mathbf{H}$。假设空间该如何表示呢？本节提出了一种表示方式，这种学习的模型称之为**感知器（Perceptron）**。

这种假设空间的思想就类似考试给的成绩，对每一题给一个特定的分数，即权重，说白了就是给输入向量的每个属性乘以一个加权值$\boldsymbol{w_i}$，在设计一个及格线，即所谓的阈值或者叫**门槛值（threshold）**，如果加权求和的分数大于这个及格线就叫及格了，即对应的输出值为1，小于这个及格线成为不及格，对应的输出值为-1。其中h(x)∈H，如公式2-1所示 $\boldsymbol{h(x) = sign(\sum_{i=1}^{d} w_i x_i - threshold)}$

公式 2-1

其中sign括号中所包含的内容大于0时，取+1；小于0时，取-1。

此时可以对h(x)做一些数学上的简化，注意这仅仅是一种数学表示方式的简化，如公式2-2所示。

$$
\begin{aligned}
\boldsymbol{h(x)} &= \boldsymbol{sign([\sum_{i=1}^{d} - threshold])} \\
&= \boldsymbol{sign([\sum_{i=1}^{d} + (-threshold * 1)])} \\
&= \boldsymbol{sign([\sum_{i=1}^{d} + (w_0 * x_0)])} \\
&= \boldsymbol{sign([\sum_{i=0}^{d} w_i x_i])} \\
&= \boldsymbol{sign(W^T \cdot X)}
\end{aligned}
$$

公式2-2

如上所示，将阈值的负数表示为权值向量中的一项，用$\boldsymbol{w_0}$ 表示，而对应权值分量$\boldsymbol{w_0}$的输入分量则被默认为1，用 $\boldsymbol{x_0}$ 最终将公式简化为两个向量内积的形式，其中T表示转置。

这里必须说明一个问题，就是不同h(x) 对应着不同的向量，即可以说假设空间H就是向量$\boldsymbol{W}$的取值范围。

这么描述还是很抽象，因此引入一种方式就是使用图像（或者可以说是几何）来更形象更具体的来说明以上函数。（这里说点题外话，由于二元函数和三元函数可以使用几何图像来一一对应，用几何的方式更直观的表示函数的意义，方便大家理解，这在以后的章节中会不断使用）

为了理解的方便将输入向量的**维度限制为两个**，即h函数可以表示成公式2-3。 $\boldsymbol{h(x) = sign(w_0 + w_1 x_1 + w_2 x_2)}$

公式2-3

将输入向量对应于一个二维平面上的点（如果向量的维度更高，对应于一个高维空间中的点）。

输出y（在分类问题中又称作标签，label）使用○表示+1，×表示-1。

假设h对应一条条的直线（如果在输入向量是高维空间的话，则对应于一个超平面）这里不止一条，不同的权值向量$\boldsymbol{W}$对应不同的直线，因为sign是以0为分界线的函数，所以可以设$\boldsymbol{w_0 + w_1 x_1 + w_2 x_2 = 0}$，该式恰是一条直线的表示。

因此每条边的一边为正的，而另一边为表示为负的。

最终得到的图像如图2-1所示。

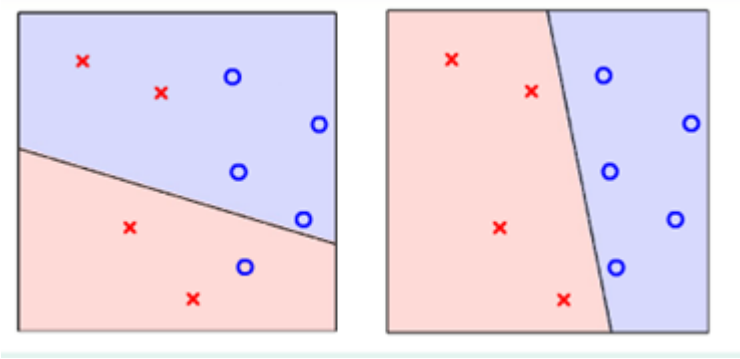


图2-1 感知器在维度为2时的几何表示

因此这里将感知器作为一条二元线性分类器（linear (binary) classifiers）。

2.2 Perceptron Learning Algorithm (PLA) 感知器学习算法

在第一章中，我们介绍过一个机器学习模型由两部分组成，而上一节仅仅介绍了它其中的一部分即假设空间H如何表示。

本节我们将更详细的介绍感知器的算法A，即如何从假设空间中找到一个近似未知目标函数f的最好假设g(x)。

问题是，我们如何找到这个g(x)呢？

首先考虑，g(x)和目标函数f越接近越好，但问题是我们不知道f（如果知道了就不需要学习了）

但是我们知道些什么呢？知道的是样本输入x在f(x)作用下得到的标记y。

所以如果我们能使得g(x)在所有的样本输入中都能够得到跟f函数作用过输入得到的输出一样的话，我们认为这时的g是不错的。（在后面的章节还会在这种思想的基础上更深入的讨论这一问题）

但是问题又来了，假设空间H的函数h(x)有无数种表示，即向量w有无数种取值。（如在二元输入时，假设空间对于在二维平面上的直线，在那个空间中可以画出无数条直线）

面对这无数多种情况，我们又该如何求解？

我们想到一个简单的方式，就是一步一步的修正错误的分类，在二维平面中可以想象成一条初始的直线，在经过不断的纠正它的错误（就是旋转平移之类的）使得最终的结果可以达到希望的效果。

还要在重复上一节中已经得到的一个结论，在感知器模型中，每一个假设函数h都对应一个权值向量。因此我们要做的就是不断修正这个权值向量使得最接近目标函数f。

PLA介绍

首先我们在设置初始 w_0（注意此处是向量不是向量的分量！），比如设置为0向量，然后使用训练样本来将权值向量修正的更接近目标函数f。其修正步骤如下：

将权值向量的修正次数表示为t，t=0,1,2,...

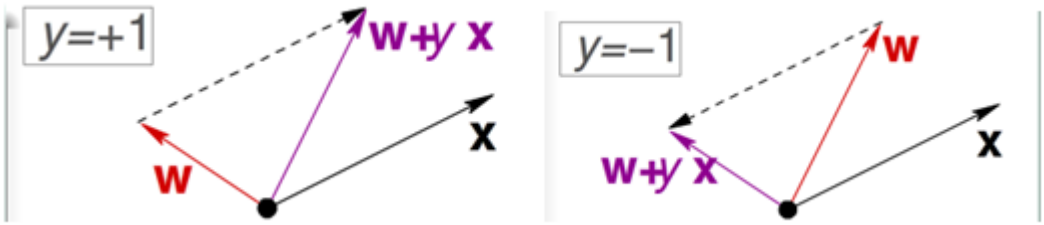
在何种情况下需要修正向量 呢？如公式2-4所示。 $sign(w_i^T \cdot X_{n(t)})$

其中训练样本 $(x_n^T \cdot y_{n(t)})$，$x_{n(t)}$ 为在t次时使用的输入向量x，而 $y_{n(t)}$ 为在t次时的标记量

该公式2-4的意思就是在t次时，选择的权值向量，有一个训练 $(x_n^T \cdot y_{n(t)})$ 样本使得在经过 $h_t(x)$（即 w）假设计算的得到的标签与f(x)得到的标签不一致。

在这种情况下就需要对权值向量进行修改，使它符合条件。修改的公式如公式2-5所示。 $w_{t+1} = w_t + y_{n(t)} \cdot x_{n(t)}$

从直觉上理解这个公式相对困难，我们还是将它化成一个几何图形，更准确的说法变成向量加减的形式去理解它，如图2-2所示。



公式2-5的几何解释

图2-2a中是在本身标记为+1时，权值向量和输入向量的内积为负数，对权值向量略作修改，加上一个标记y和输入向量的乘积，得到一个新的权值向量，可以看出新的权值向量和输入向量的相乘之后符合了标记的要求。

图2-2b中是在本身标记为-1时，权值向量和输入向量的内积为正数，对权值向量略作修改，加上一个标记y和输入向量的乘积，得到一个新的权值向量，可以看出新的权值向量和输入向量的相乘之后符合了标记的要求。

如此这般的重复查找错误样本和修改加权向量，直到再也找不到可以使公式2-4成立的样本为止，此时得到的加权向量，即为我们想要的最终g。

描述了上面内容之后，你很可能有一个疑问就如何查找错误样本点，或者如何确定没有错误的点了。

一个简单的方式就是将训练样本编号，从1到n，整个训练样本就有n个点。以按从1到n的顺序不断查找错误点，如果没有错就自动的用下一个样本点继续查找，当从1到n这n个样本点都没有产生错误时，算法即结束得到g。将这种方式的算法叫做Cyclic PLA。

这时候就又出来几个新的问题，第一，这个算法一定会找到一个能使所有的样本都不符合（即都被分对了类）的情况吗？就是这个算法会不会停止？第二个问题这个算法找到的真的是最好的g吗？看起来好像只是在训练样本中才符合这一性质，如果出现新的样本的话又会如何呢？

第一个问题下一小节将进行介绍，而其他问题会在后面的章节中讨论。

2.3 Guarantee of PLA PLA算法可行的保障

PLA算法只有在满足训练样本是线性可分（linear separable）的情况下才可以停止。

什么是线性可分呢？简单的说就是存在一条直线能将两类样本点完全分开。

如图2-3所示。

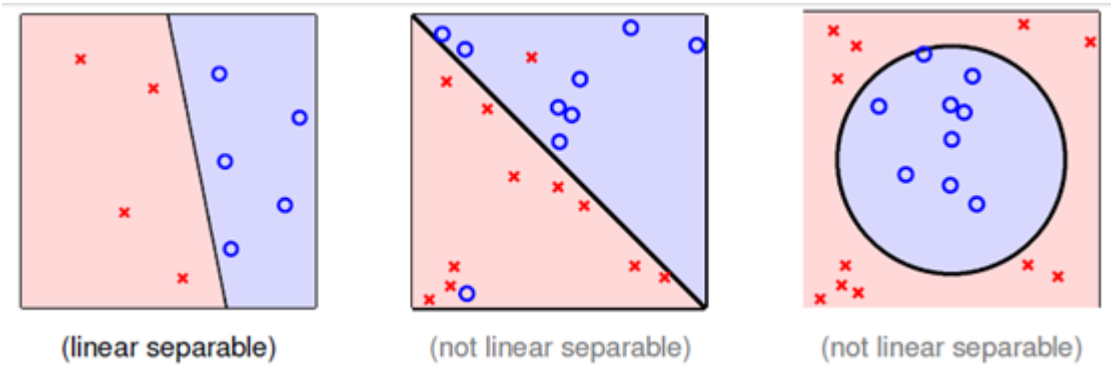


图2-3 线性可分与线性不可分

其中最左边的为线性可分的训练样本，而右边两个图形为线性不可分的两种情况，这两种情况会在后面的章节一一解释。

我们需要证明在线性可分的情况下，权值向量在经过一段时间的修正会停止，即t次修正会有一个上界。

首先我们考虑是否每次修正都可以使得权值向量w_r变得更好，就是是否会更接近未知的目标函数所表示的向量。有了这个思路，我们先假设目标函数的权值向量为w_f，可以求解出两个向量相似度的度量方式有很多，其中比较常用的一种方式就是求两个向量的内积，于是我们对w_r和w_f做内积。其中T表示为停止时的次数。

直接使用这两个向量做内积，其内积越大并不能代表这两个向量越接近，因为向量本身的变长也可以导致这一现象。因此我们需要求解的是这两个向量做归一化（就是各自除以自身的L1范式得到单位向量）之后的内积，这时它俩的内积有了上界即为1，如公式2-6所示 $\frac{w_f^T w_T}{||w_f^T|| \cdot ||w_T||}$

乍一看公式2-6完全无从下手，w_f^T是未知目标向量，w_T是终止时的向量，也是一个未知向量，因此思路就是将其中一个未知量消除，消除w_f^T的可能性不大，因此选择消除w_T在公式中的不确定性，如公式2-7所示是解决归一化之前两个向量内积的问题。

$$\begin{aligned} W_f^T &\geq W_f^T(W_{T-1} + Y_{n(T-1)}X_{n(T-1)}) \\ &= W_f^T W_{T-2} + Y_{n(T-1)}W_f^T X_{n(T-1)} \end{aligned}$$

取所有样本中$Y_{n(T-1)}W_{n(T-1)}$的最小乘积(因为W_f^T是在线性可分情况下的目标函数，所以所有的$Y_{n(T-1)}W_f^T X_{n(T-1)}$必定大于等于0)

$$\begin{aligned} &\geq W_f^T W_{T-1} + \min_n Y_n W_f^T X_n \\ &\geq W_f^T W_{T-2} + 2\min_n Y_n W_f^T X_n \end{aligned}$$

进行迭代

$$\geq W_f^T W_0 + T\min_n Y_n W_f^T X_n$$

又因为初始值设置为0向量,因此$W_0 = 0$

$$= T\min_n Y_n W_f^T X_n$$

除了W_T不容易确定之外，W_T的L1范式$||W_T||$也不容易得出，如公式2-8是求解L1范式的不等式，其思想如公式2-7。

$$\begin{aligned}||W_T|| &= ||W_{T-1} + Y_{n(T-1)}X_{n(T-1)}|| \\ &= ||W_{T-1}||^2 + ||Y_{n(T-1)}X_{n(T-1)}||^2 + 2Y_{n(T-1)}W_{T-1}X_{n(T-1)}\end{aligned}$$

因为只有在犯错的情况下才会进行改变，那什么时候是犯错，就是在公式2-4成立的情况，即$sign(W_t^T \cdot X_{n(t)})$，该公式等价于$Y_{n(t)} \cdot X_{n(t)} \leq 0$，因此如下2-8

$$\begin{aligned}&\leq ||W_{T-1}||^2 + ||Y_{n(T-1)}X_{n(T-1)}||^2 + 0 \\ &\leq ||W_{T-1}||^2 + \max||X_n||^2 \\ &\leq \ldots \leq ||W_0||^2 + T\max_n||X_n||^2 \\ &= T\max_n||X_n||^2\end{aligned}$$

通过公式2-7和公式2-8可以将公式2-6写成如公式2-9，如下式所示。

$$\begin{aligned}\frac{W_f^T W_T}{||W_f^T|| \cdot ||W_T||} &\geq \frac{T\min_n Y_n W_f^T X_n}{W_f^T\sqrt{T\max_n||X_n||^2}} \\ &= \sqrt{T}\frac{\min_n Y_n \frac{W_f^T}{||W_f^T||}X_n}{\sqrt{\max_n||X_n||^2}} \\ &= \sqrt{T} \cdot C\end{aligned}$$

将公式2-9中的常数设置为C，该公式如公式2-10所示。

$$\frac{\min_n Y_n \frac{W_f^T}{||W_f^T||}X_n}{\sqrt{\max_n||X_n||^2}}$$

可以看出权值向量和目标函数内积会以的速度不断的增长，但是这种增长不是没有限制的，它最多只能等于1。

因此有以下结论，如公式2-11所示。

$$\begin{aligned}1 &\geq \frac{w_f^T w_T}{||w_f^T|| \cdot ||w_T||} \\ &\geq \sqrt{T}\frac{\min_n Y_n \frac{W_f^T}{||W_f^T||}X_n}{\sqrt{\max_n||X_n||^2}}\end{aligned}$$

求解得到公式2-12的结论。

$$T \leq \frac{\max_n||X_n||^2}{\min_n Y_n \frac{W_f^T}{||W_f^T||}X_n}$$

将公式2-12中的值分别使用简单的数字符号代替，如公式2-13和公式2-14所示。

$$R^2 = \max_n ||X_n||^2$$

$$\rho = (\min_n \frac{W_f^T}{||W_f^T||} X_n)^2$$

从公式2-12中就可以看出T是有上界，即在线性可分的情况下PLA算法最终会停止，找到一个最接近目标函数的假设函数g。

2.4 Non-Separable Data 线性不可分的数据

上一节的阐述PLA这个算法一定会停下来这一结论，是建立在存在一个目标函数，可以将所有的数据点都线性分开这个假设的基础之上。对于一堆复杂的数据，如何能确定它一定是线性可分的？比如一个PLA算法运行了很长时间仍然没有停止，此时存在两种可能性，

> 一是该数据集是线性可分的，但是还没有运行结束；

> 另一种，压根就不存在一条直线可以将数据集分开，就是压根这个算法就不会终止。

假如是后者又该如何处理？

首先还是要解释下为什么会出现后者，此种情况出现的概率大吗？

出现不可分的一种可能是从未知目标函数中产生的训练样本存在噪音（noise），如录入样本时有人工的错误等情况导致数据本身不正确，使得最终本可以线性可分的样本集变得线性不可分了，如图2-4所示。

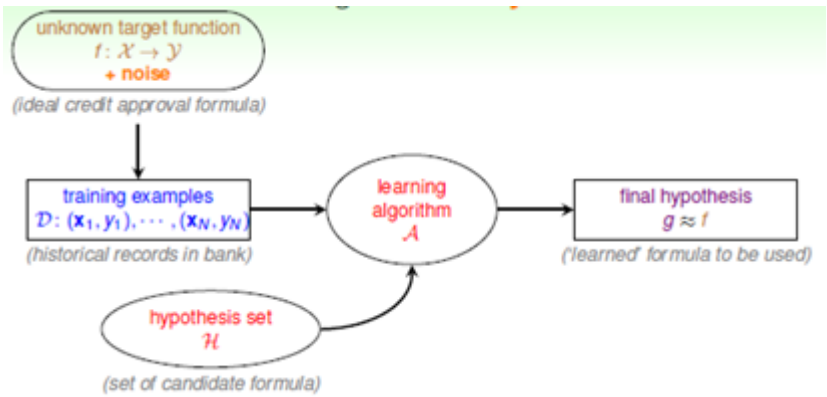


图2-4 加入噪音的机器学习流程图

而噪音占整个数据集的比例一般不会太大，如图2-5所示。这种情况下我们又该如何计算出最佳的假设g呢？

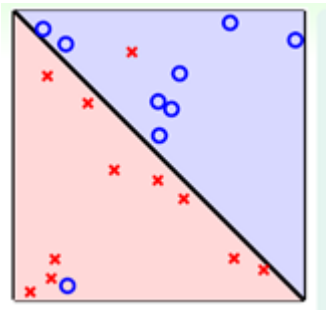


图2-5 存在噪音时线性不可分的情况

一种新的思路是找出犯错最少的权值向量 ，如公式2-15所示。 $$W_g = \arg\min_{w} \sum_{n=1}^{N} sign(W^T X_n) \neq Y_n$$

其中表示当满足条件时输出1，否则为0。但是这个公式在数学上是NP难问题，我们无法直接求解，于是我们需要找出一种近似的算法来求解这个问题。

这里介绍一个叫pocket的算法，它的本质是一种贪心算法，做一简单的介绍：

1. 也是随机的初始化一个权值向量
2. 随机的使用n个点中的一个点去发现是否有错误（此处与cyclic PLA使用的循环方式有所不同，不是按顺序一个一个的查看是否符合条件，而是在**n个点中随机的抽取**，这种方式可以增加其寻找最优解的速度）
3. 和PLA一样使用公式2-5进行修正.
4. 如果有了修正，则计算出刚刚修正过的权值向量和上一个权值向量到底谁犯的错误比较少，将少的保留重复第2步到第4步的动作。

假如很长时间都没有新的权值向量比当前的权值向量犯错更少，则返回该向量作为函数g。