# Perception learning感知机学习

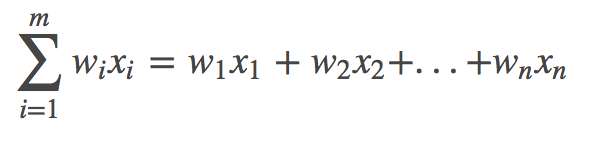
在机器学习中，**感知机**（perceptron）是二分类的线性分类模型，属于监督学习算法。输入为实例的特征向量，输出为实例的类别（取+1和-1）。感知机对应于输入空间中将实例划分为两类的分离超平面。感知机旨在求出该超平面，为求得超平面导入了基于误分类的损失函数，利用梯度下降法 对损失函数进行最优化（最优化）。感知机的学习算法具有简单而易于实现的优点，分为原始形式和对偶形式。感知机预测是用学习得到的感知机模型对新的实例进行预测的，因此属于判别模型。感知机由**Rosenblatt**于1957年提出的，是**神经网络**和**支持向量机**的基础。

## ****定义****

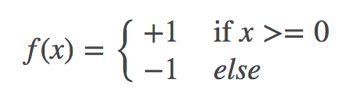
假设输入空间(特征向量)为*X*⊆*Rn*，输出空间为Y={-1, +1}。输入*x*∈*X*表示实例的特征向量，对应于输入空间的点；输出y∈Y表示示例的类别。由输入空间到输出空间的函数为

*f*(*x*)=*sign*(*w*·*x*+*b*)

称为感知机。其中，参数w叫做权值向量**weight**，b称为偏置**bias**。*w*·*x*表示w和x的**点积**



sign为符号函数，即



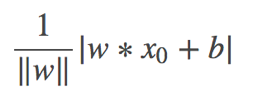
在二分类问题中，*f*(*x*)的值（+1或-1）用于分类*x*为正样本（+1）还是负样本（-1）。感知机是一种线性分类模型，属于判别模型。我们需要做的就是找到一个最佳的满足*w*⋅*x*+*b*=0的w和b值，即分离超平面（separating hyperplane）。

## ****学习策略****

**核心：极小化损失函数。**

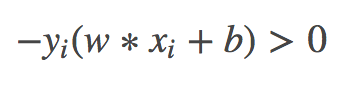
如果训练集是可分的，感知机的学习目的是求得一个能将训练集正实例点和负实例点完全分开的分离超平面。为了找到这样一个平面（或超平面），即确定感知机模型参数w和b，我们采用的是损失函数，同时并将损失函数极小化。

对于损失函数的选择，我们采用的是误分类点到超平面的距离（可以自己推算一下，这里采用的是几何间距，就是点到直线的距离）：

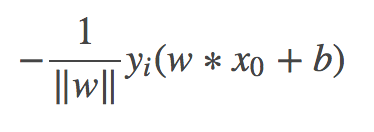


其中||*w*||是*L*2范数。

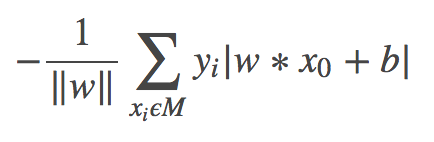
对于误分类点(*xi*,*yi*)来说：



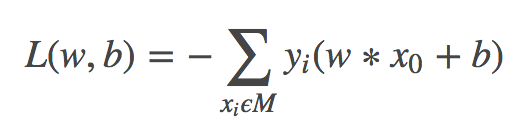
误分类点到超平面的距离为：



那么，所有点到超平面的总距离为：

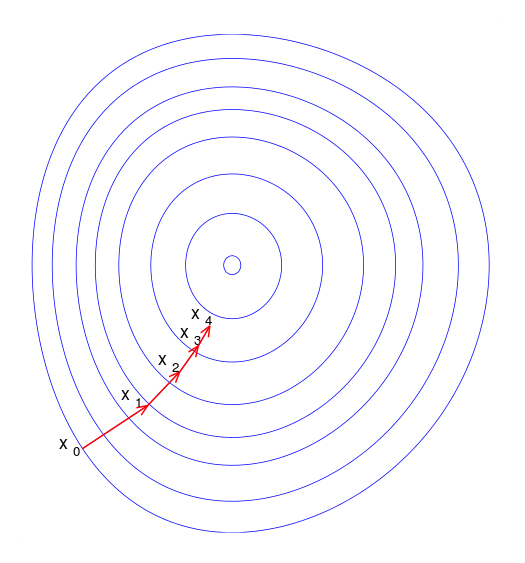


不考虑1||*w*||,就得到感知机的损失函数了。

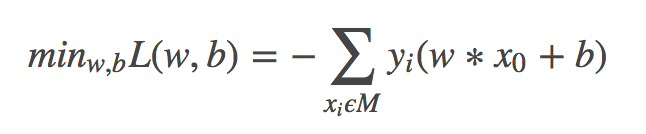


其中M为误分类的集合。这个损失函数就是感知机学习的**经验风险函数**。可以看出，随时函数*L*(*w*,*b*)是非负的。**如果没有误分类点，则损失函数的值为0，而且误分类点越少，误分类点距离超平面就越近，损失函数值就越小**。同时，损失函数*L*(*w*,*b*)是连续可导函数。

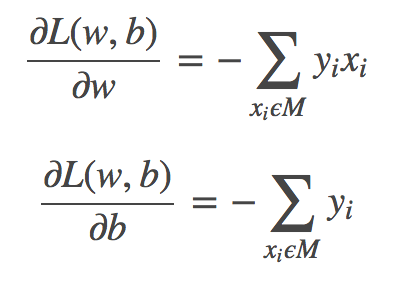
## ****学习算法****

感知机学习转变成求解损失函数*L*(*w*,*b*)的最优化问题。最优化的方法是随机梯度下降法（stochastic gradient descent），这里采用的就是该方法。关于梯度下降的详细内容，参考[wikipedia Gradient descent](https://en.wikipedia.org/wiki/Gradient_descent)。下面给出一个简单的梯度下降的可视化图：  


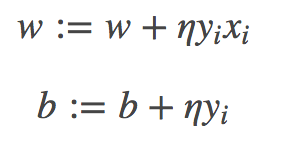
上图就是随机梯度下降法一步一步达到最优值的过程，说明一下，梯度下降其实是局部最优。感知机学习算法本身是误分类驱动的，因此我们采用随机梯度下降法。首先，任选一个超平面*w*0和*b*0，然后使用梯度下降法不断地**极小化目标函数**



极小化过程不是一次使M中所有误分类点的梯度下降，而是一次随机的选取一个误分类点使其梯度下降。使用的规则为 *θ*:=*θ*−*α*∇*θℓ*(*θ*)，其中α是步长，∇θℓ(θ)是梯度。假设误分类点集合M是固定的，那么损失函数L(w,b)的梯度通过偏导计算：



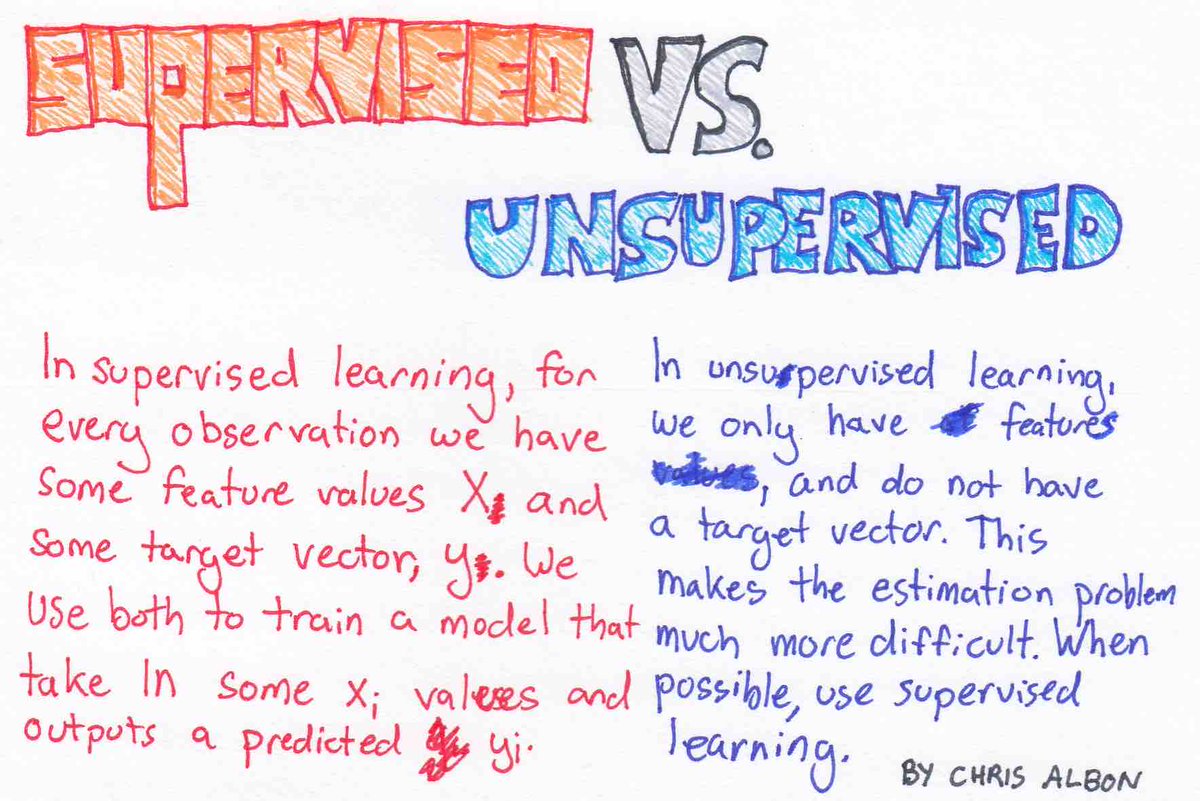
然后，随机选取一个误分类点，根据上面的规则，计算新的*w*,*b*，然后进行更新：



其中*η*是步长，大于0小于1，在统计学习中称之为学习率（learning rate）。这样，通过迭代可以期待损失函数*L*(*w*,*b*)不断减小，直至为0.

参考资料：<https://blog.csdn.net/dream_angel_z/article/details/48915561>

# Supervised vs Unsupervised监督和无监督



有两种类型的学习 - 受监督和无监督。

有监督学习 –

在有监督学习中，我们有一些训练数据。使用这些训练数据，我们创建了一个预测模型，可用于未来预测结果。最终目标是建立一个称为假设函数的预测函数f（x）。 “学习”使用数学算法对该函数进行优化，以便给定输入数据x关于某个域，它将能够预测一些有趣的值f（x）。例如 - 房子的市场价格。属于此类别的一些算法是 - 回归，分类 - KNN，决策树，朴素湾，支持向量机。

示例 - 假设您提供了装满不同水果的篮子，您的任务是对它们进行分组。假设这些水果是苹果，香蕉，西瓜，橙子。既然你熟悉水果，你可以很容易地将它们分成苹果，香蕉，西瓜，橙子等各种类别。这项活动被称为训练数据。现在，如果您获得不同类似的水果，您将在未来使用此学习体验。这种类型的算法被称为分类。

无监督学习 -

与受监督学习不同，在无监督情况下，我们拥有带有输入x的训练数据集，但没有任何目标值或我们可以预测的内容。此信息缺失。这种学习的目标是在数据集内找出一组类似的例子。这种算法被称为聚类。以上例。现在让我们说，如果你再次提供了装满水果的桶，但这次你不知道那些水果。你以前从未见过他们。然后你将使用身体特征将他们放在不同的组中。比方说颜色，大小等，这称为聚类。因此，无监督学习任务涉及识别数据内的关系。这里没有提供培训示例。算法的例子是 - 用K-means聚类。

参考资料：<https://www.linkedin.com/pulse/machine-learning-supervised-vs-unsupervised-amit-sinha>

# Overfit vs Underfit

# 

过拟合：

1）简单理解就是训练样本的得到的输出和期望输出基本一致，但是测试样本输出和测试样本的期望输出相差却很大 。

2）为了得到一致假设而使假设变得过度复杂称为过拟合。想像某种学习算法产生了一个过拟合的分类器，这个分类器能够百分之百的正确分类样本数据（即再拿样本中的文档来给它，它绝对不会分错），但也就为了能够对样本完全正确的分类，使得它的构造如此精细复杂，规则如此严格，以至于任何与样本数据稍有不同的文档它全都认为不属于这个类别！

如果数据本身呈现二次型，故用一条二次曲线拟合会更好。但普通的PLS程序只提供线性方程供拟合之用。这就产生拟合不足即“欠拟合”现象，从而在预报时要造成偏差。如果我们用人工神经网络拟合，则因为三层人工神经网络拟合能力极强，有能力拟合任何函数。如果拟合彻底，就会连实验数据点分布不均匀，实验数据的误差等等“噪声”都按最小二乘判据拟合进数学模型。这当然也会造成预报的偏差。这就是“过拟合”的一个实例了。

参考资料：<https://blog.csdn.net/cindysuna/article/details/50057579>

# The Argument For Parametric Models

# 

# 当数据生成函数大致匹配参数概率分布时，我们可以将我们的计算仅限于其参数。这使得仅使用少量信息就知道很多。

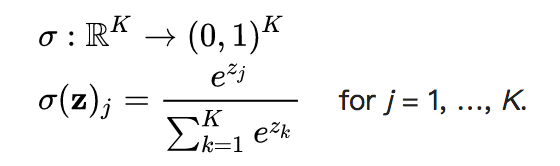
许多概率分布的灵活性意味着如果选择分布族(舉例：正態跟學生T)通常不是问题

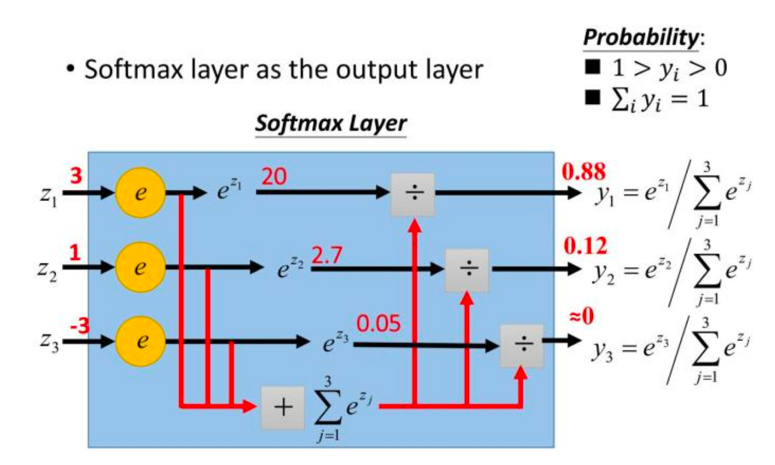
然而，范围匹配是重要的。例如，如果我们想要一个概率，我们就不应该选择输出数大于1的概率分布。

# Softmax Normalization

softmax函数的本质就是将一个K 维的任意实数向量压缩（映射）成另一个K

维的实数向量，其中向量中的每个元素取值都介于（0，1）之间。





舉例：

如果我们输入[1，2，3，4，1，2，3]，那么softmax是[0.024，0.064，0.175，0.475，0.024，0.064，0.175]。输出的大部分重量都是原始输入中的'4'。这是该功能通常用于：突出显示最大值并抑制显着低于最大值的值。但是请注意：softmax不是尺度不变的，所以如果输入[0.1,0.2,0.3,0.4,0.1,0.2,0.3]（其总和为1.6），softmax将是[0.125,0.138,0.1513,0.169,0.125， 0.138，0.153]。这表明，对于0到1之间的值，softmax事实上不重视最大值（注意，0.169不仅小于0.475，它也小于初始值0.4）。

Softmax标准化是一种减少数据中极值或异常值的影响的方法，无需从数据集中移除它们。给出离群数据非常有用，我们希望将其包含在数据集中，同时仍将数据的重要性保留在均值的标准差内。

# 

参考资料：

<https://en.wikipedia.org/wiki/Softmax_function#Softmax_normalization>

<https://blog.csdn.net/u014422406/article/details/52805924>

<https://www.zhihu.com/question/23765351>

{\displaystyle \sigma (\mathbf {z} )\_{j}={\frac {e^{z\_{j}}}{\sum \_{k=1}^{K}e^{z\_{k}}}}}