1. **认识机器学习**
   1. **什么是机器学习**

机器学习，顾名思义，是通过机器对于目前任务进行大量学习并基于此可以正常地执行任务的过程。我们可以简单地将机器学习总结为*基于大量的数据去找寻到一个解决某件事务的函数*。因为对于任何机器要处理的事务，都是给定一个输入，通过机器得到输出。就比如说我们想要进行语音识别，那我们就可以输入一段语音，通过机器内部的复杂函数关系，从而得出结果，输出文字。

当然，这里说的函数不是我们通常日常计算的函数，而是从输入到输出的一个或多个复杂的映射，这样的函数对于我们人类来说难以解出，所以说这就是机器学习其存在的意义。

* 1. **机器学习的类型**

上面说到，机器学习的主要工作原理就是根据事务来找到最适合的函数，那么既然是函数，他就需要输入和输出。更具不同的输入输出的情况，我们将机器学习进行了如下分类：

1. 当我们的**输出为一个或多个标量（数字）**时，那这个机器学习方法我们就可以称为**回归（Regression）**。这种方法的例子就比如根据目前的气象指数来预测接下来的各项天气知数等等。
2. 当我们的**输出为从多个选项中的某个特定选项**，那这样的机器学习方法我们就可以称为**分类（Classification）**。就比如说，我们输入一张猫的照片，然后给定的答案区间就是猫和狗，要让机器根据自己的判断输出正确的答案。

还有一种情况，不同于上面的任何情况，而是**让机器创造一些具有结构的东西（create something with structure(image,document)）**，让机器学会自己创造事务，这样的情况称为**结构化学习（Structured learning）**。最好的例子就是去年很火的“AI自动续写文章”之类的。

* 1. **如何去实现机器学习（简单步骤）**

上面已经简单地介绍了机器学习的定义与分类，下面就是一个实现机器学习的简单例子。我们知道机器学习就是从事务与大量的数据中提炼出一个函数。对于一个事务，他会受到很多因素的影响，所以说他的结果不会和日常数学学习中的函数那么稳定，所以说我们实现机器学习的问题就可以转化为去找最适合函数的问题。

首先第一步，就是要对这个函数进行一个预估与猜测。我们可以设这个函数为一个简单的函数 得到一个**模型（Model）**，这样形如的函数我们可以称之为**线性模型（Lineal model）**。这个函数，我们把这个函数中已知量称为**特征f（Feature）**，与特征相连的未知量称为**权重w（Weight）**，单独的未知量称为**偏差/偏置b（Bias），**自己设置的参数称为**超参数（Hyperparameters）。**

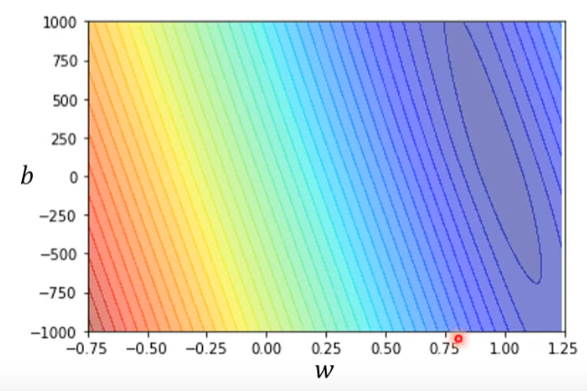
第二步，根据上一步假设得到的模型定义一个损失（loss,is a function of parameters）函数L(b,w),其参数均为模型函数中的未知量，函数的输入就是weight和bias的值，函数的输出值用来判断对应的函数是否合适。

损失函数的计算需要基于模型的训练数据，就是将训练数据输入值输入到模型函数得到的预估值与真实情况的输出值进行比较得到，多次值得平均值（或残差或其他方法）即为该损失函数得值。计算函数可以简化为：Loss: L = ; = (一些写成 = 为真实输出值)。其中，真实数据被称为标签（Label）。

上述方法中e = 的方法称为**平均绝对误差mean absolute error(MAE)**。

还有e = 的方法称为**均方误差（最小二乘法原理）mean square error(MSE)。**而当计算值与实际值均为概率分布，则需要用到**交叉熵方法Cross-entropy）**。

第三步就是优化函数。上一步得到的Loss值是反映函数中w和b值得取值是否合适，其中Loss的值越大，说明越不合适，Loss的值越小说明越合适。根据不同的w和b的值，我们可以得到一张二维等高线图，称为**误差面（Error Surface）**。在图像中，越红区域代表Loss越大，越蓝区域表示Loss越小。



我们的优化步骤就是找到一个最佳的w和b使得L最小，这样的一组w和b我们称之为。在这个步骤中，我们唯一会用到的方法称为**梯度下降法（Gradient Descent）**。

最简单的情况就是参数只有一个w时，当w为不同数值时，对应的L也不同，w与L构建出来的误差面为一条曲线。我们首先需要随机选取一个w的值为，接着我们需要计算在该点L的微分，即计算的值（一维曲线上的表示即曲线在该点的切线斜率，）。根据几何定义，我们可以得出，当该微分值为负，则说明曲线左高右低；当该微分为正，则说明曲线左低右高。w的值往曲线低趋势的地方根据步伐（间隔）变换，重复多次。步伐的设定与两方面因素有关，**一是该点斜率大小，二是与学习速率有关**。设**η为机器学习的学习速率,是一个超参数。**该点斜率越大，步伐就可以大一点，η设置的比较大，那么步伐改变就越快；反之也成立。所以说步伐就可以近似用**step =** 的值来表示，那么下一个微分点就为 。

此步骤何时停止取决于两种情况。一是设定循环的次数，循环到达次数后即停止；二是不断循环直到找到使得的值为0的点停止。但是第二种方法有时只会找到**局部最优解（Local Minima）,**而不能找到**全局的最优解（Global Minima）**（情况近似于极小值与最小值的关系）。

我们将梯度下降的方法推广到更加广泛的范围去，即模型的参数有多个，比如w和b时，我们就可以类比上面的方法来进行数据的处理。

跟单独一个参数不同的时，我们的误差面变成了二维的表格，那我们就需要计算在(,)位置时L分别对w和b的微分（也就是偏微分），即计算 和 的值，然后在w和b方向分别以**，**的步伐向和方向移动，重复执行至停止。

上面得到模型的三个步骤我们称之为**训练（training）**，是我们根据已有的大量数据得到模型的过程，我们接下来要做的就是使用这个模型进行使用和预测。

但在使用过程中我们可能会发现这个函数会达不到我们的预期，那么我们就需要对我们的模型进行更进一步的修改，结合我们的**领域知识（Domain knowledge）**，对模型进行更深层次的修改。

一般的事务通常不会是一种简单的线性关系即不一定为线性模型（Lineal model）。线性模型太过于简单，我们需要更加复杂的模型，线性模型有一些限制称为**模型偏置（Model bias），即无法表述真实的情况**。针对复杂的情况，我们可以将复杂情况简化为**常数b加上多段分段函数**的方法来表示**（constant + sum of set of lineal models），这样的函数被称为弹性函数或复杂函数。**对于不同的输入值，通过分段函数可以实现更加灵活与真实的输出，所加上的常数b为**函数与y轴的交点**。针对函数为曲线的情况，我们可以使用直线来进行拟合，点取得越多，越贴合曲线。

针对其中的一条分段函数，我们可以使用**sigmoid函数（）**来进行拟合。这个函数为**逻辑回归（Logistic regression）**的一般形式，就是高中生物里面的S型曲线，我们需要去逼近的直线称之为**Hard sigmoid函数**。当趋近于无穷的时候，y趋近于c。通常Hard sigmoid函数会比较难地去直接实现，但如果可以得到比较准确的且简单的Hard sigmoid函数，我们可以不用sigmoid函数去进行拟合。

每个Hard sigmoid函数都可以拆分为两个**矫正线性单元（Rectified Lineal Unit）**组成的，简称为ReLU。每个ReLU可以写为，由一条水平线和斜线组成。ReLU和sigmoid函数在机器学习中属于**激活函数（Activation function）**。

针对于sigmoid函数，我们可以调整其中的c、w、b三个参数实现不同的曲线。其中，c越大，函数最值越大，整体函数变高；w改变会；b改变会使曲线左右移动。

所以说复杂函数中的分段函数都是由sigmoid函数进行拟合的，只是每个函数的c、w、b的取值不同。所以说这个复杂的函数我们可以写成：。

通常，我们会使用多个数据作为函数的输入，在线性模型中我们可以使用进行表示多个输入。在复杂函数中，我们可以使用来针对多个x输入的情况。每个i循环代表一个分段函数，每段i中的b和w都是相同的。但如果使用Hard sigmoid函数表示的话，我们就可以写成**。**

//

我们可以将所有的未知参数集合成一个向量，我们将其命名为θ。我们进行复杂函数的损失函数的计算就可以计算L(θ)。具体的方法和上面线性模型求损失函数的方法一样，代入数据x值球的平均差作为L。求出来的θ可以使得L的值越小，说明越合适。我们就需要进行梯度下降的方法，任选出θ值为，求出向量，就是一般定义上的**梯度（Gradient），即可以表示为，**其数学含义就是将中的所有成员对L进行微分。而更新的方向和步伐我们可以表示为**，即原向量减去乘以参数η的梯度得到新的向量。**其中中的上标表示梯度下降的次数，下标表示向量中参数的位置。上面的写法可以简化为。上述方法一直重复，停止规则和上述简单函数的梯度下降规则一致。

一般我们针对大量的数据会进行分组，n个数量的数据为**一批（batch），称为B，**其中n为超参数。在每次梯度下降的过程中，我们会使用不同B的梯度g去进行操作。所有的batch都使用一次我们称作一个**纪元（Epoch）**，每一次更新参数称作**更新（Update）**。更新次数等于数据总量除以每个B的数量。

针对已经进行过sigmoid或ReLU的某段分段函数我们可以继续进行sigmoid和ReLU,但是会增加更多的参数，增加更多的层数。每次使用sigmoid或ReLU会生成一个**神经（Neuron）**，多次使用可以形成**神经网络（Neural Network）。**因为此种方法可以生成多层神经元，我们就可以称之为**深度学习（deep learning）。**

但值得注意的是，不一定是sigmoid或ReLU使用地越多越好，有时候多次使用会产生过拟合（Overfitting）的问题，使得符合原本数据但预测结果不准确。