任务一 线性回归算法梳理

学习内容：

1. **机器学习的一些概念**
2. **有监督:**

监督学习中有教师和监督者的概念，其主要功能是提供误差的精确度量（直接和输出值相比）。

在实际算法中，该功能由多组对应值（输入和期望输出）组成的训练集提供。基于训练集可以修正模型参数以减少全局损失函数。在每次迭代之后，如果算法足够灵活并且数据是相关的，则模型总体精度增加，并且预测值和期望值之间的差异变得接近于零。

1. **无监督**

无监督学习方法没有任何监督，而只是基于绝对误差的衡量。当需要对一组数据根据其相似度（或距离）进行分组（聚类）时，需要采用无监督学习方法。

在无法提供实际的监督数据时，强化学习使用基于环境提供的反馈来进行学习。在这种情况下，反馈得到的更多是定性的信息，并不能确定其误差的精确度量。

在强化学习中，这种反馈通常被称为**奖励（reward）（有时负面的反馈被定义为惩罚），**而了解在一个状态下执行某个行为是否是正面的非常有用。最有用的行为的顺序是必须学习策略，以便能够为得到高的计时和累积奖励做出最好的决策。

换句话说，一个动作可能是不完美的，但就整体策略而言，它必须能够提供最高的奖励。当处于经常动态变化的不确定环境时，无法实现对误差的精确测量，强化学习时非常有效的方法。

1. **泛化能力**

泛化能力（generalization ability）是指一个机器学习算法对于没有见过样本的识别能力。也叫做“举一反三”的能力，或者叫做学以致用的能力。

1. **过拟合、欠拟合（方差和偏差以及各自解决办法）**

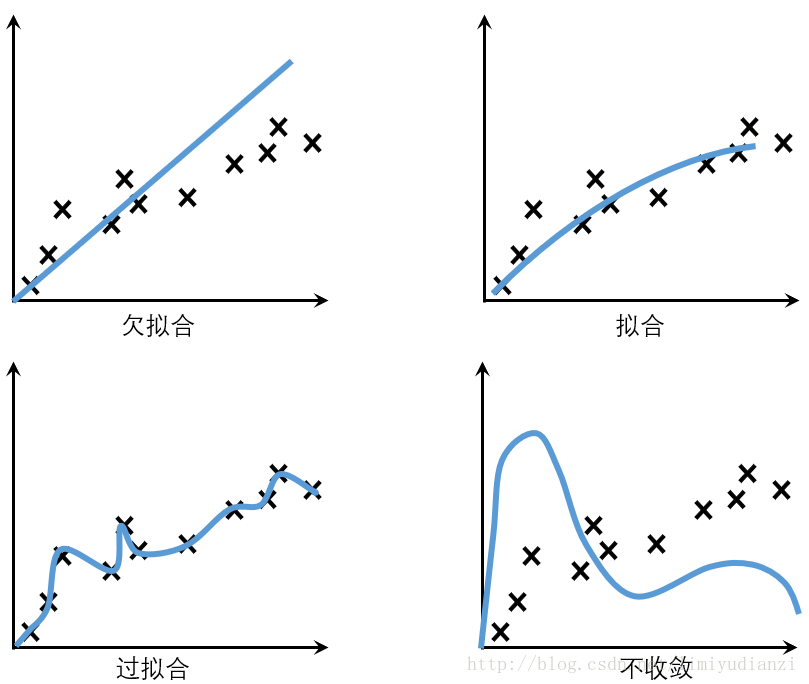
* 过拟合

过拟合是一种分类器会发生的现象，泛化能力可以理解为对分类器的一种性能的评价。过拟合通常可以理解为，模型的复杂度要高于实际的问题，所以就会导致模型死记硬背的记住，而没有理解背后的规律。

* 欠拟合

欠拟合（under-fitting）是和过拟合相对的现象，可以说是模型复杂度较低，没法很好的学习到数据背后的规律。

不收敛一般形容一些基于梯度下降法的模型，收敛是指这个算法有能力找到局部或全局最小值，（比如找到使得预测的标签和真实的标签最相近的值，也就是二者距离的最小值），从而得到一个问题的最优解。如果说一个机器学习算法的效果和瞎蒙的差不多，那么基本可以说这个算法没有收敛，也就是根本没有去学习。



1. **偏差和方差**

通过使用偏差和方差这两个参数衡量模型的拟合程度。我们通过这两个参数的大小，以及相应的图像来简单的判断模型是否欠拟合、过拟合问题。并以此来判断我们是否需要选择：增加训练集数量、修改某些参数、增减特征数量等操作，来优化改进算法。

判断结论

1. 高偏差：模型存在欠拟合问题
2. 高方差：模型存在过拟合问题

机器学习中：数据分为3部分：训练集、验证集（验证方法有流出法、交叉验证法、自助法）、测试集。

* 偏差

偏差，度量了学习算法的期望预测与真实结果的偏离程度，即刻画了学习算法本身的拟合能力。偏差的定义式如下：

* 方差

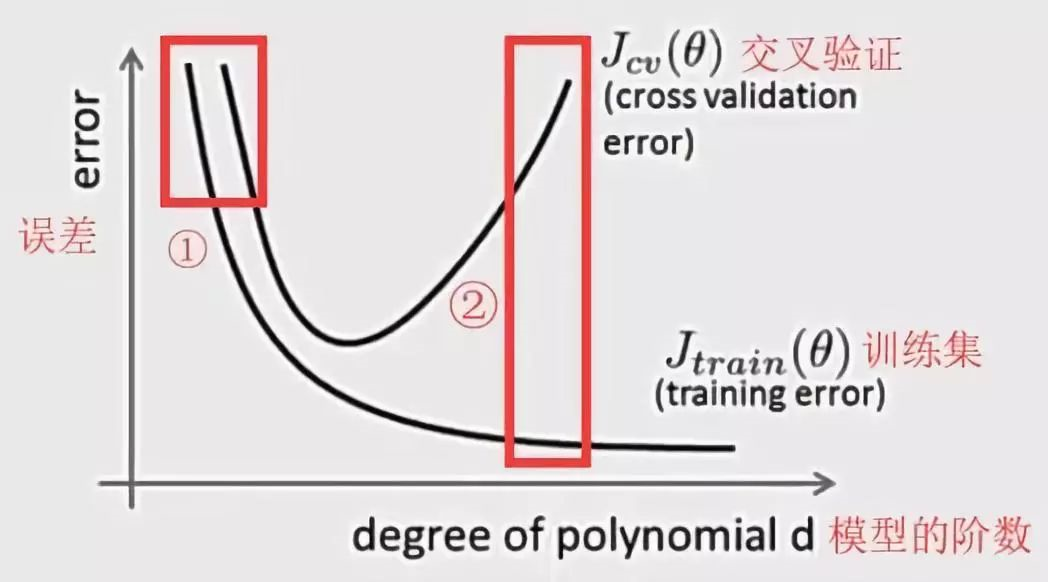
方差，指一个特定训练集训练得到的函数，与所有训练集得到平均函数的差的平方再取期望，

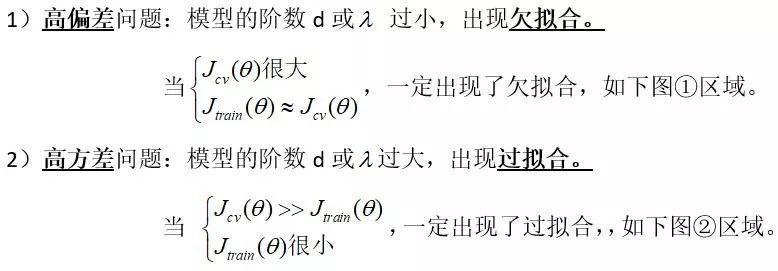
方差，度量了同样大小的训练集的变动所导致的学习性能的变化，即刻画了数据扰动所造成的影响。方差表示所有模型构建的预测函数，与真实函数的差别有多大。

方差的定义式如下：

* 使用偏差、方差判断拟合程度

我们使用训练集和交叉验证集的代价函数，来判断模型是否存在欠拟合、过拟合等问题。

使用下面的图像来判断：



1. **交叉验证**

它的基本思想就是将原始数据（dataset）进行分组，一部分做为训练集来训练模型，另一部分做为测试集来评价模型。

交叉验证用于评估模型的预测性能，尤其是训练好的模型在新数据上的表现，可以在一定程度上减小过拟合；还可以从有限的数据中获取尽可能多的有效信息。

主要方法：

1. 留出法 （holdout cross validation

在机器学习任务中，拿到数据后，我们首先会将原始数据集随机分为三部分：训练集、验证集和测试集。 训练集用于训练模型，验证集用于模型的参数选择配置，测试集对于模型来说是未知数据，用于评估模型的泛化能力。

2.k 折交叉验证（k-fold cross validation）加以改进：

k 折交叉验证通过对 k 个不同分组训练的结果进行平均来减少方差，因此模型的性能对数据的划分就不那么敏感。

k 折交叉验证通过对 k 个不同分组训练的结果进行平均来减少方差，因此模型的性能对数据的划分就不那么敏感。

第一步，不重复抽样将原始数据随机分为 k 份。

第二步，每一次挑选其中 1 份作为测试集，剩余 k-1 份作为训练集用于模型训练。

第三步，重复第二步 k 次，这样每个子集都有一次机会作为测试集，其余机会作为训练集。

在每个训练集上训练后得到一个模型，

用这个模型在相应的测试集上测试，计算并保存模型的评估指标，

第四步，计算 k 组测试结果的平均值作为模型精度的估计，并作为当前 k 折交叉验证下模型的性能指标。

k 一般取 10，

数据量小的时候，k 可以设大一点，这样训练集占整体比例就比较大，不过同时训练的模型个数也增多。 数据量大的时候，k 可以设小一点。

3.留一法

当 k＝m 即样本总数时，叫做留一法（Leave one out cross validation），每次的测试集都只有一个样本，要进行 m 次训练和预测。 这个方法用于训练的数据只比整体数据集少了一个样本，因此最接近原始样本的分布。 但是训练复杂度增加了，因为模型的数量与原始数据样本数量相同。 一般在数据缺乏时使用。

此外：

多次 k 折交叉验证再求均值，例如：10 次 10 折交叉验证，以求更精确一点。

划分时有多种方法，例如对非平衡数据可以用分层采样，就是在每一份子集中都保持和原始数据集相同的类别比例。模型训练过程的所有步骤，包括模型选择，特征选择等都是在单个折叠 fold 中独立执行的。

还有一种比较特殊的交叉验证方式，Bootstrapping： 通过自助采样法，即在含有 m 个样本的数据集中，每次随机挑选一个样本，再放回到数据集中，再随机挑选一个样本，这样有放回地进行抽样 m 次，组成了新的数据集作为训练集。

这里会有重复多次的样本，也会有一次都没有出现的样本，原数据集中大概有 36.8% 的样本不会出现在新组数据集中。优点是训练集的样本总数和原数据集一样都是 m，并且仍有约 1/3 的数据不被训练而可以作为测试集。 缺点是这样产生的训练集的数据分布和原数据集的不一样了，会引入估计偏差。

此种方法不是很常用，除非数据量真的很少

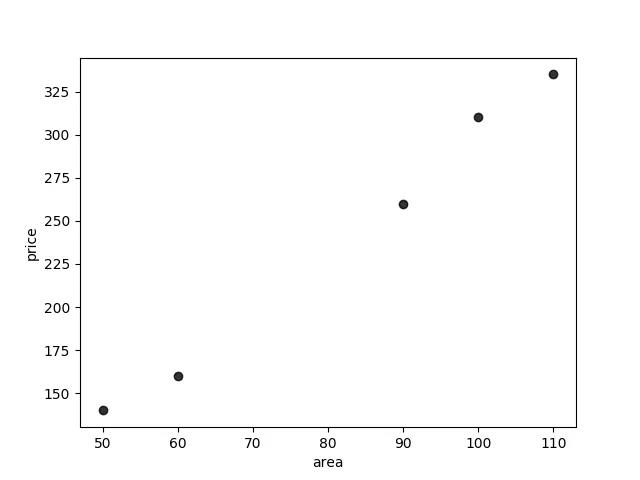
1. **线性回归的原理**

线性回归的模型形如：

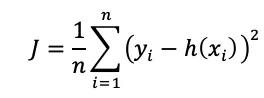


线性回归得出的模型不一定是一条直线，在只有一个变量的时候，模型是平面中的一条直线；有两个变量的时候，模型是空间中的一个平面；有更多变量时，模型将是更高维的。

线性回归模型有很好的可解释性，可以从权重W直接看出每个特征对结果的影响程度。

线性回归适用于X和y之间存在线性关系的数据集，可以使用计算机辅助画出散点图来观察是否存在线性关系。例如我们假设房屋价格和房屋面积之间存在某种线性关系，画出散点图如下图所示。

看起来这些点分布在一条直线附近，我们尝试使用一条直线来拟合数据，使所有点到直线的距离之和最小。实际上，线性回归中通常使用残差平方和，即点到直线的平行于y轴的距离而不用垂线距离，残差平方和除以样本量n就是均方误差。均方误差作为线性回归模型的代价函数(cost function)。使所有点到直线的距离之和最小，就是使均方误差最小化，这个方法叫做最小二乘法。

代价函数：

其中，



下面求使J最小的W和b：

1.偏导数法

偏导数法是非常麻烦的，需要一个一个地计算w。为了方便，这里以单变量线性回归为例。

2.正规方程法

正规方程使用矩阵运算，可以一次求出W向量。但是当变量(feature)个数大于数据个数时，会导致xTx不可逆，这时候就不能用此方法了。

使用正规方程法，如果希望得到的模型带有偏置项b，就要先给数据集X增加全为1的一列，这样才会把b包含在W中；如果不添加，那么模型是强制过原点的。

3.梯度下降法

这里的代价函数J的海森矩阵H是半正定的，因此J一定有全局最小值，所以也可以使用梯度下降法来求解。梯度下降法是一种迭代解法，不仅可以求解最小二乘问题，也适用于其它代价函数的问题。但是需要设置学习率α，α设置的过大或过小，都不能很好地训练出模型，而且梯度下降法需要对数据集进行特征缩放。一般会在数据集特别大的时候或者xTx不可逆的时候使用梯度下降法。

1. **线性回归的损失函数、代价函数、目标函数**

代价函数（有的地方也叫损失函数，Loss Function）在机器学习中的每一种算法中都很重要，因为训练模型的过程就是优化代价函数的过程，代价函数对每个参数的偏导数就是梯度下降中提到的梯度，防止过拟合时添加的正则化项也是加在代价函数后面的。在学习相关算法的过程中，对代价函数的理解也在不断的加深，在此做一个小结。

假设有训练样本(x, y)，模型为h，参数为θ。h(θ) = θTx（θT表示θ的转置）。

（1）概况来讲，任何能够衡量模型预测出来的值h(θ)与真实值y之间的差异的函数都可以叫做代价函数C(θ)，如果有多个样本，则可以将所有代价函数的取值求均值，记做J(θ)。因此很容易就可以得出以下关于代价函数的性质：

对于每种算法来说，代价函数不是唯一的；

代价函数是参数θ的函数；

总的代价函数J(θ)可以用来评价模型的好坏，代价函数越小说明模型和参数越符合训练样本(x, y)；

J(θ)是一个标量；

（2）当我们确定了模型h，后面做的所有事情就是训练模型的参数θ。那么什么时候模型的训练才能结束呢？这时候也涉及到代价函数，由于代价函数是用来衡量模型好坏的，我们的目标当然是得到最好的模型（也就是最符合训练样本(x, y)的模型）。因此训练参数的过程就是不断改变θ，从而得到更小的J(θ)的过程。理想情况下，当我们取到代价函数J的最小值时，就得到了最优的参数θ，记为：

minθJ(θ)minθJ(θ)

例如，J(θ) = 0，表示我们的模型完美的拟合了观察的数据，没有任何误差。

（3）在优化参数θ的过程中，最常用的方法是梯度下降，这里的梯度就是代价函数J(θ)对θ1, θ2, ..., θn的偏导数。由于需要求偏导，我们可以得到另一个关于代价函数的性质：

选择代价函数时，最好挑选对参数θ可微的函数（全微分存在，偏导数一定存在）

2. 代价函数的常见形式

经过上面的描述，一个好的代价函数需要满足两个最基本的要求：能够评价模型的准确性，对参数θ可微。

**2.1 均方误差**

在线性回归中，最常用的是**均方误差**(Mean squared error)，具体形式为：

J(θ0,θ1)=12m∑i=1m(y^(i)−y(i))2=12m∑i=1m(hθ(x(i))−y(i))2J(θ0,θ1)=12m∑i=1m(y^(i)−y(i))2=12m∑i=1m(hθ(x(i))−y(i))2

m：训练样本的个数；

hθ(x)：用参数θ和x预测出来的y值；

y：原训练样本中的y值，也就是标准答案

上角标(i)：第i个样本

**2.2 交叉熵**

在逻辑回归中，最常用的是代价函数是**交叉熵**(Cross Entropy)，交叉熵是一个常见的代价函数，在神经网络中也会用到。下面是《神经网络与深度学习》一书对交叉熵的解释：

交叉熵是对「出乎意料」（译者注：原文使用suprise）的度量。神经元的目标是去计算函数y, 且y=y(x)。但是我们让它取而代之计算函数a, 且a=a(x)。假设我们把a当作y等于1的概率，1−a是y等于0的概率。那么，交叉熵衡量的是我们在知道y的真实值时的平均「出乎意料」程度。当输出是我们期望的值，我们的「出乎意料」程度比较低；当输出不是我们期望的，我们的「出乎意料」程度就比较高。

在1948年，克劳德·艾尔伍德·香农将热力学的熵，引入到信息论，因此它又被称为香农熵(Shannon Entropy)，它是香农信息量(Shannon Information Content, SIC)的期望。香农信息量用来度量不确定性的大小：一个事件的香农信息量等于0，表示该事件的发生不会给我们提供任何新的信息，例如确定性的事件，发生的概率是1，发生了也不会引起任何惊讶；当不可能事件发生时，香农信息量为无穷大，这表示给我们提供了无穷多的新信息，并且使我们无限的惊讶。更多解释可以看[这里](http://blog.csdn.net/rtygbwwwerr/article/details/50778098" \t "_blank)。

J(θ)=−1m[∑i=1m(y(i)loghθ(x(i))+(1−y(i))log(1−hθ(x(i)))]J(θ)=−1m[∑i=1m(y(i)log⁡hθ(x(i))+(1−y(i))log⁡(1−hθ(x(i)))]

符号说明同上

**2.3 神经网络中的代价函数**

学习过神经网络后，发现逻辑回归其实是神经网络的一种特例（没有隐藏层的神经网络）。因此神经网络中的代价函数与逻辑回归中的代价函数非常相似：

J(θ)=−1m[∑i=1m∑k=1K(y(i)kloghθ(x(i))+(1−y(i)k)log(1−(hθ(x(i)))k)]J(θ)=−1m[∑i=1m∑k=1K(yk(i)log⁡hθ(x(i))+(1−yk(i))log⁡(1−(hθ(x(i)))k)]

这里之所以多了一层求和项，是因为神经网络的输出一般都不是单一的值，K表示在多分类中的类型数

1. **优化方法（梯度下降法、牛顿法、拟牛顿法等）**

1)梯度下降法

梯度下降法实现简单，当目标函数是凸函数时，梯度下降法的解是全局解。一般情况下，其解不保证是全局最优解，梯度下降法的速度也未必是最快的。

梯度下降法的优化思想：用当前位置负梯度方向作为搜索方向，因为该方向为当前位置的最快下降方向，所以也被称为是”最速下降法“。最速下降法越接近目标值，步长越小，前进越慢。

缺点：

靠近极小值时收敛速度减慢，求解需要很多次的迭代；

直线搜索时可能会产生一些问题；

可能会“之字形”地下降。

2)牛顿法

牛顿法最大的特点就在于它的收敛速度很快。

优点：二阶收敛，收敛速度快；

缺点：

牛顿法是一种迭代算法，每一步都需要求解目标函数的Hessian矩阵的逆矩阵，计算比较复杂。

牛顿法收敛速度为二阶，对于正定二次函数一步迭代即达最优解。

牛顿法是局部收敛的，当初始点选择不当时，往往导致不收敛；

二阶海塞矩阵必须可逆，否则算法进行困难。

关于牛顿法和梯度下降法的效率对比：

从本质上去看，牛顿法是二阶收敛，梯度下降是一阶收敛，所以牛顿法就更快。如果更通俗地说的话，比如你想找一条最短的路径走到一个盆地的最底部，梯度下降法每次只从你当前所处位置选一个坡度最大的方向走一步，牛顿法在选择方向时，不仅会考虑坡度是否够大，还会考虑你走了一步之后，坡度是否会变得更大。所以，可以说牛顿法比梯度下降法看得更远一点，能更快地走到最底部。（牛顿法目光更加长远，所以少走弯路；相对而言，梯度下降法只考虑了局部的最优，没有全局思想。）

根据wiki上的解释，从几何上说，牛顿法就是用一个二次曲面去拟合你当前所处位置的局部曲面，而梯度下降法是用一个平面去拟合当前的局部曲面，通常情况下，二次曲面的拟合会比平面更好，所以牛顿法选择的下降路径会更符合真实的最优下降路径。

3)拟牛顿法

拟牛顿法的本质思想是改善牛顿法每次需要求解复杂的Hessian矩阵的逆矩阵的缺陷，它使用正定矩阵来近似Hessian矩阵的逆，从而简化了运算的复杂度。

拟牛顿法和最速下降法一样只要求每一步迭代时知道目标函数的梯度。通过测量梯度的变化，构造一个目标函数的模型使之足以产生超线性收敛性。这类方法大大优于最速下降法，尤其对于困难的问题。另外，因为拟牛顿法不需要二阶导数的信息，所以有时比牛顿法更为有效。如今，优化软件中包含了大量的拟牛顿算法用来解决无约束，约束，和大规模的优化问题。

1. **线性回归的评估指标**

对于机器学习的两个基本问题分类和回归的评价方式有所不同，分类问题一般通过分类准确率、召回率、F1值、ROC/AUC等手段进行模型的评估。对于回归问题，该如何评价？  这里简要列举部分评估方法。

1）残差估计

总体思想是计算实际值与预测值间的差值简称残差。从而实现对回归模型的评估，一般可以画出残差图，进行分析评估、估计模型的异常值、同时还可以检查模型是否是线性的、以及误差是否随机分布。可以看出，残差随机分布在0值附近，总体离0值越近说明回归模型的拟合效果越好。对于偏离较大的残差点，可以认为是异常值点。

2）均方误差(Mean Squared Error, MSE)

均方误差是线性模型拟合过程中，最小化误差平方和(SSE)代价函数的平均值。MSE可以用于不同模型的比较，或是通过网格搜索进行参数调优，以及交叉验证等。

3)决定系数

可以看做是MSE的标准化版本，用于更好地解释模型的性能。换句话说，决定系数是模型捕获相应反差的分数。