**1、引言**

**机器学习是什么？**

**Arthur Samuel**：在进行特定编程的情况下，给予计算机学习能力的领域。  
**Tom Mitchell**：一个程序被认为能从经验E中学习，解决任务T，达到性能度量值P，当且仅当，有了经验E后，经过P评判，程序在处理T时的性能有所提升。

#### 监督学习与无监督学习

根据训练数据是否有标记信息，机器学习任务大致分为两大类：**监督学习**和**非监督学习**，分类和回归是监督学习的代表，而聚类是非监督学习的代表。

**输入空间、特征空间、输出空间、假设空间**

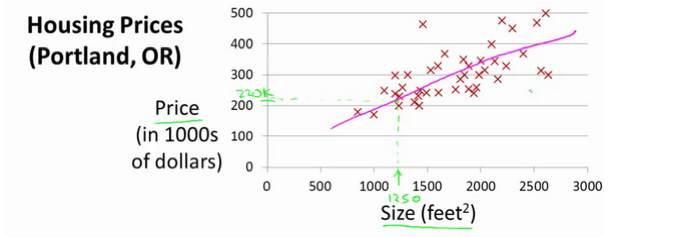
**输入空间**：在监督学习中，将输入所有可能取值的集合称为输入空间。  
**特征空间**：每个具体的输入是一个实例，通常由特征向量表示，所有特征向量存在的空间成为特征空间。有时输入空间和特征空间为相同的空间，有时为不同的空间，需要将实例从输入空间映射到输出空间。  
**输出空间**：在监督学习中，将输出所有可能取值的集合称为输出空间。  
**假设空间**：监督学习的目的在于学习一个由输入到输出的映射，这一映射由模型来表示。由输入空间到输出空间的映射的集合，称为假设空间。举个简单的例子，在一元线性回归中，假设空间即所有的直线y=ax+b组成的集合，我们的目标就是找到一条y=a'x+b'，使得损失最小。

**生成模型和判别模型**

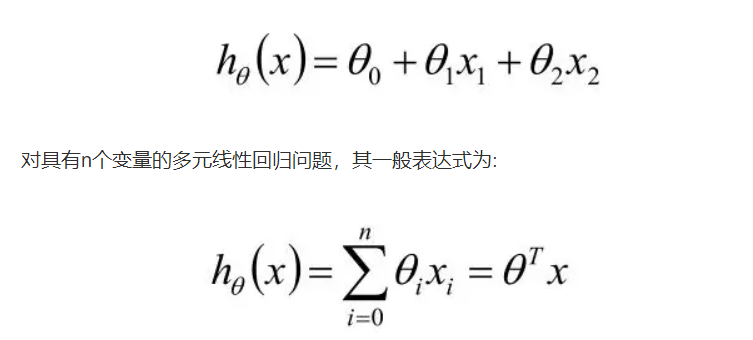
**生成模型**：生成模型由数据学习联合概率分布P(X,Y)，按后求出条件概率分布P(Y|X)，作为预测的模型。之所以被称为生成方法，是因为模型表示了给定输入X产生输出Y的关系。典型的模型有**朴素贝叶斯方法**和**隐马尔可夫模型的**。  
**判别模型**：判别模型由数据直接学习决策函数f(X),或者条件概率分布P(Y|X)。判别方法关心的是对给定的输入X，应该预测什么样的输出Y。典型的判别模型包括**k近邻算法**，**感知机**，**决策树**，**逻辑回归**，**支持向量机**等。

**2、线性回归**

**模型表示**

线性回归是最简单的机器学习模型，其假设输入和输出之间满足线性关系，假设我们想要通过房屋尺寸来预测房价，通过将收集到的数据绘制在二维坐标系中，我们总中拟合得到图中的直线：

以二元线性回归为例，其表达式如下：



#### 损失函数

线性回归的损失函数一般是**平方损失函数**

梯度下降**梯度下降(Gradient Descent)**是一个用来求函数最小值的算法，后面会有一张专门讲解梯度下降法及其改进算法，这里只是介绍一个基本的思想。  
梯度下降背后的思想是：开始时我们随机选择一个参数的组合，计算代价函数，然后我们寻找下一个能让代价函数值下降最多的参数组合。我们持续这么做直到找到一个局部最小值（**local minimum**），因为我们并没有尝试完所有的参数组合，所以不能确定我们得到的局部最小值是否便是全局最小值（**global minimum**），选择不同的初始参数组合，可能会找到不同的局部最小值。

具体的做法就是：**每一次都同时让所有的参数减去学习速率乘以损失函数的导数。**其中α是学习率（**learning rate**），它决定了我们沿着能让代价函数下降程度最大的方向向下迈出的步子有多大。

#### 除学习率外，使用梯度下降法时需要对特征进行标准化处理，以二元线性回归为例，如果两维的数值范围相差特别大，梯度下降算法需要非常多次的迭代才能收敛，解决的方法是尝试将所有特征的尺度都尽量缩放到-1到1之间：

# 3、梯度下降法

再进一步介绍算法之前，我们先来介绍一下梯度下降法的分类及对其的改进。

## 3.1 梯度下降法分类

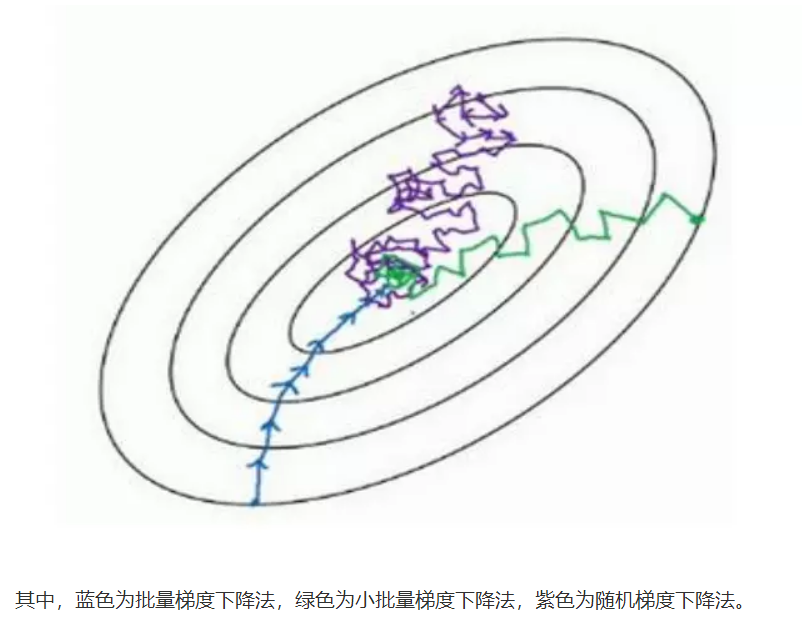
梯度下降法可以分为下面三种：

**批量梯度下降法（Batch Gradient Descent）**：批量梯度下降法，是梯度下降法最常用的形式，具体做法也就是在更新参数时使用所有的样本来进行更新。

**随机梯度下降法（Stochastic Gradient Descent）**：求梯度时没有用所有的m个样本的数据，而是仅仅选取一个样本j来求梯度。

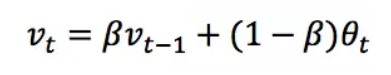
**小批量梯度下降法（Mini-batch Gradient Descent）**：小批量梯度下降法是批量梯度下降法和随机梯度下降法的折衷，也就是对于m个样本，我们采用x个样子来迭代。

三者到达极小值的路径大致可以表示为下图：

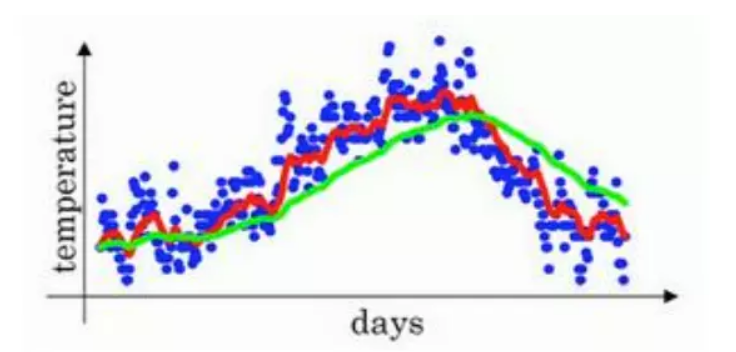


## 3.2 梯度下降法改进

#### 指数加权平均数

指数加权平均数是对趋势的一种刻画，计算公式如下：

在上式中，vt是t时刻的指数加权平均值，v0 = 0，θt是第t时刻的实际值。β是一个参数。简单的理解，β代表了vt刻画的是多少天的平均值，如果β太小，那么vt越接近于θt，如果β很大，那么vt就能代表一个更长时间的平均值。

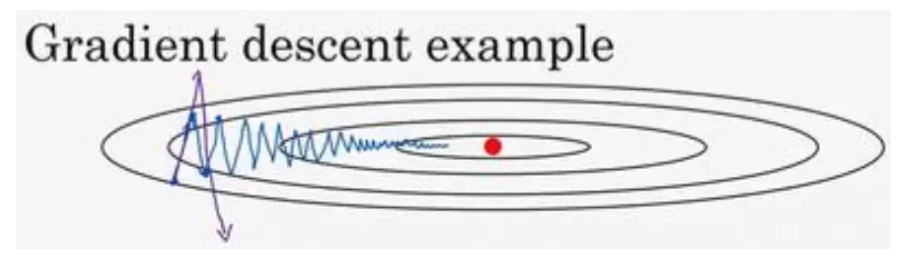
大体上，**vt大概是1/(1-β)天的平均值**，即当β=0.1时，我们可以认为vt代表了近10天的平均值。β=0.02时，我们可以认为vt代表了近50天的平均值，如下图所示，红色表示的是β=0.1时vt的变化，绿色表示的是β=0.02时vt的变化：

在指数加权平均的过程中，还有一个很重要的概念，叫做**偏差修正(Bias correction)**,可以看到，如果v0=0，那么v1 = (1-β)\*θ1，特别地，如果β=0.9，那么  
v1 = 0.1 \* θ1，这样导致v1会特别小，需要进行一定程度的修正， 具体的修正方法是对vt除以( 1-β^t)，当t特别小的时候，可以起到较为明显的修正效果，但是当t变大时，分母接近于1，基本没有进行修正。

在机器学习中，在计算指数加权平均数的大部分时候，大家不在乎执行偏差修正，因为大部分人宁愿熬过初始时期，拿到具有偏差的估测，然后继续计算下去。如果你关心初始时期的偏差，在刚开始计算指数加权移动平均数的时需要进行偏差修正。

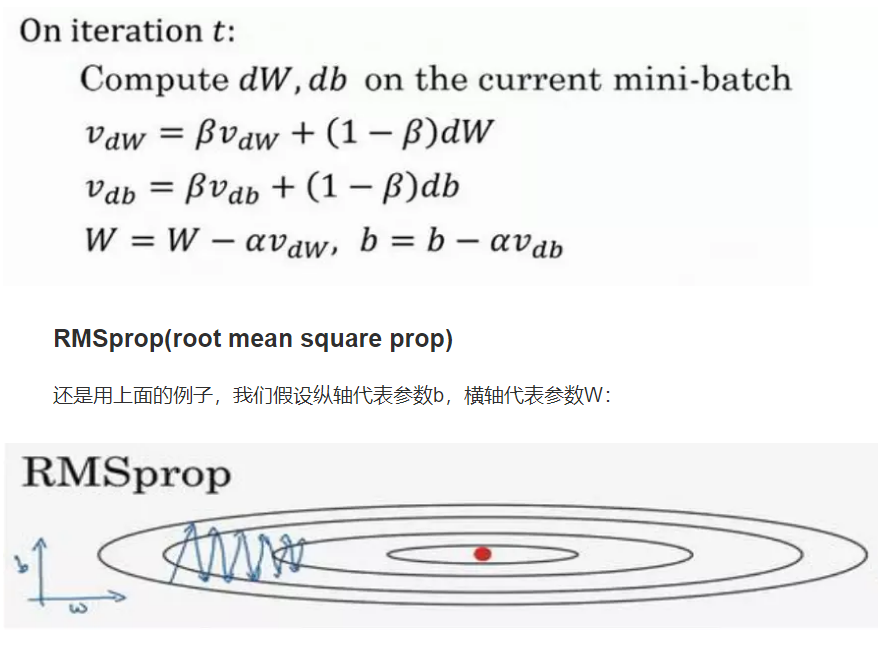
**动量梯度下降法(Momentum)**

**动量梯度下降法**基本的想法就是计算梯度的指数加权平均数，并利用该梯度 更新你的权重。假设我们使用梯度下降法更新参数的路径如下图中蓝色的线所示：



上图中的路径上下波动减慢了梯度下降法的速度，如果我们用更大的学习率，可能导致这种波动进一步加大，变成图中的紫色的线。因此，你希望在纵轴上学习慢一点，消除这种上下的摆动，而在横轴上，你希望快速从左向右移动，移动到最小值处。

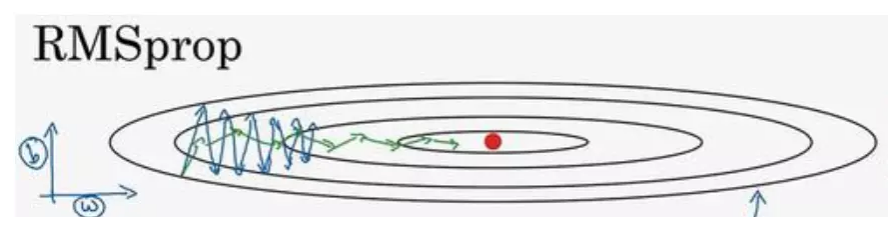
我们之前介绍的指数加权平均数，可以反映近一段时间的趋势，在纵轴上的上下波动，使得平均值接近于0，而在横轴方向的平均值仍然较大。利用这种思想，Momentum的计算过程如下：



#### RMSprop(root mean square prop)

还是用上面的例子，我们假设纵轴代表参数b，横轴代表参数W：

所以，你想减缓𝑏方向的学习，即纵轴方向，同时加快，至少不是减缓横轴方向的学习， RMSprop 算法可以实现这一点，将趋势变成下面的绿色的线：



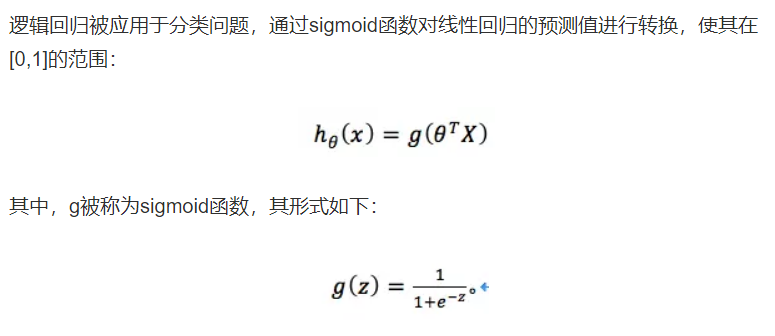
#### Adam(Adaptive Moment Estimation)

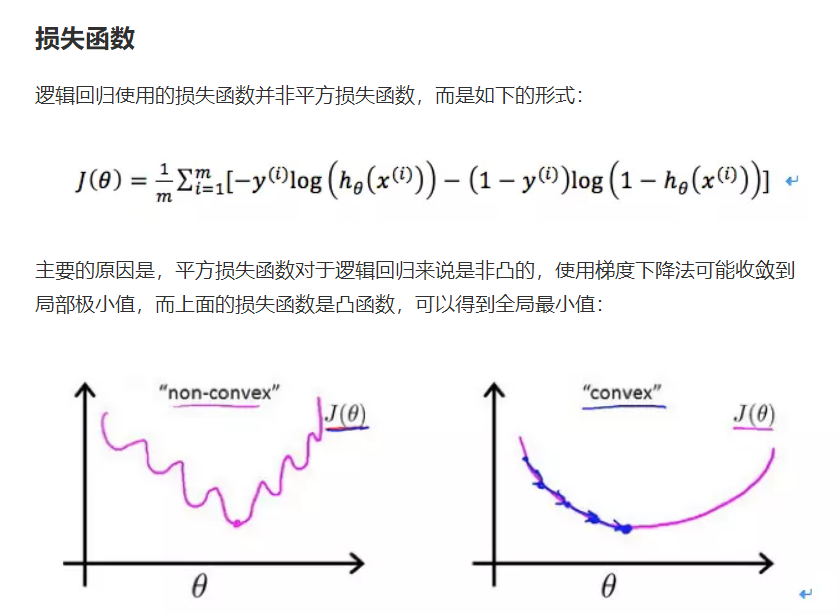
Adam 优化算法基本上就是将 Momentum 和 RMSprop 结合在一起，它使用的是经过偏差修正后的指数加权平均数：

Adam 算法结合了 Momentum 和 RMSprop 梯度下降法，并且是一种极其常用的学习算法，被证明能有效适用于不同神经网络，适用于广泛的结构。

# 4、逻辑回归

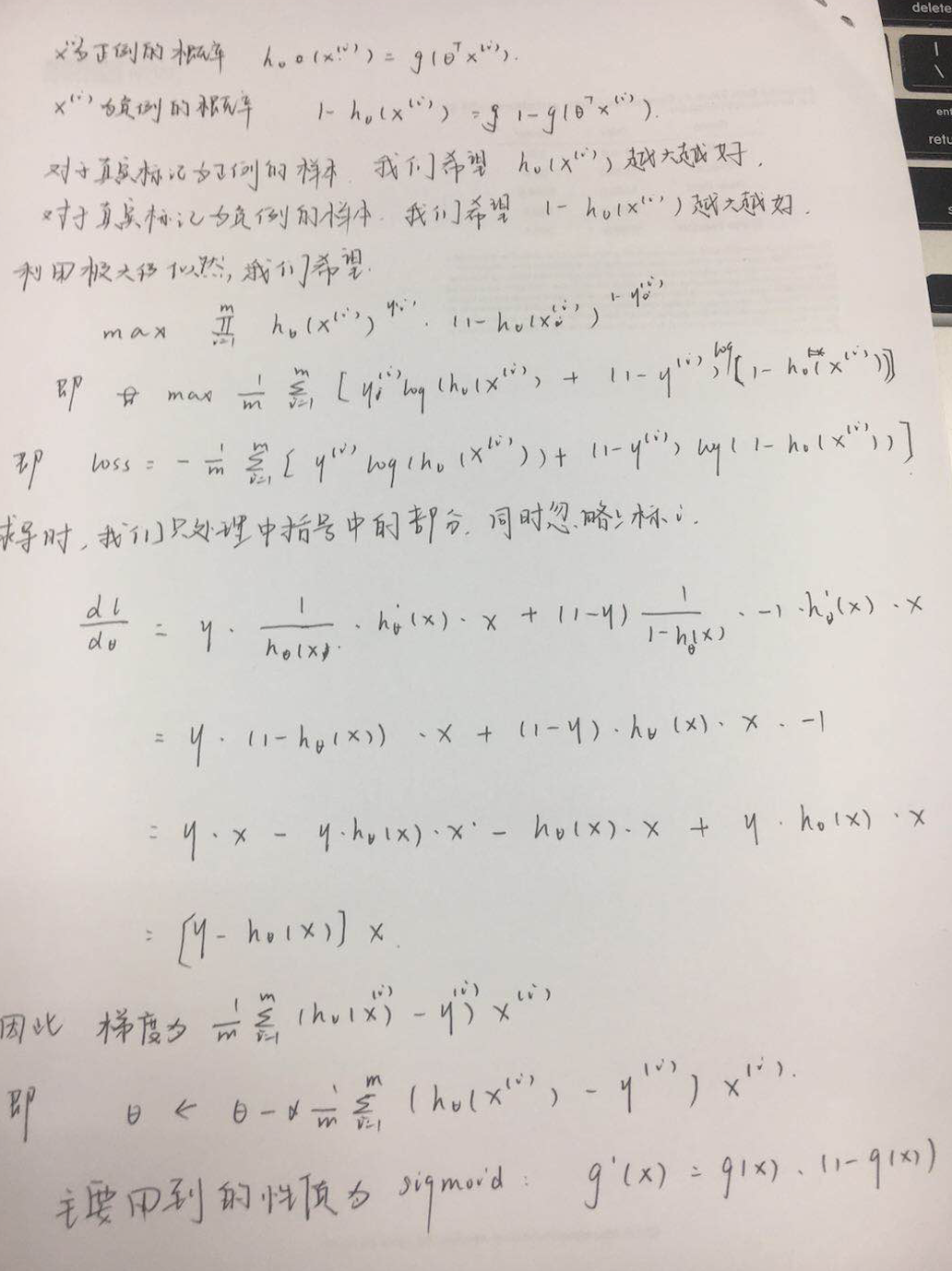
#### 基本原理





#### 过程推导

逻辑回归的推导是面试中经常会问到的问题，通过梯度下降法进行推导时，我们用到的主要性质时sigmoid函数的导数性质：g'(x) = g(x)(1-g(x))，一定要牢记于心。下图是推导过程：

**逻辑回归为什么会选择sigmoid函数呢**，总结有以下几方面吧:  
1、函数连续，单调递增  
2、求导方便  
3、输出范围为(0,1)，可以用作输出层，结果可以表示概率  
4、抑制两头，对中间细微变化敏感，对分类有利。

#### 解决多分类问题

逻辑回归还可以用于多分类，我们常用的策略被称为**one vs all**。

假设我们有三个类，如下图所示：

# https://upload-images.jianshu.io/upload_images/4155986-5658e25d28af190b.png?imageMogr2/auto-orient/采用one vs all策略时，有多少类别，我们就要训练几个分类器。每次选择一个类别作为正例，其他所有类别的样本作为负例，训练一个分类器：https://upload-images.jianshu.io/upload_images/4155986-30f0b411ffda7807.png?imageMogr2/auto-orient/5、过拟合和正则化

## 5.1 过拟合

机器学习的目的是使学到的模型不仅对已知的数据而且对未知的数据都能有很好的预测能力。当损失函数给定时，模型在训练集上的损失被称为**训练误差**，在测试集上的损失被称为**测试误差**。

如果模型不能很好的适应训练集，会造成**欠拟合(underfit)**。相反，如果模型过于强调拟合原始数据，会导致对未知数据拟合很差，这种情况被称为**过拟合(overfit)**。

看下面的例子：

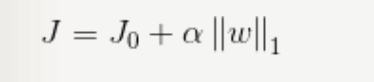
## https://upload-images.jianshu.io/upload_images/4155986-564fb46a9fb2df37.png?imageMogr2/auto-orient/5.2 正则化

正则化的目的是对过大的参数进行一定的惩罚，降低其对模型的影响，使我们能够得到一个较为简单的模型。这符合**奥卡姆剃刀**的原理，即在所有可能选择的模型中，能够很好地解释已知数据并且十分简单的才是最好的模型。

我们常用的正则化有**L1正则化**和**L2正则化**。

#### L1正则化

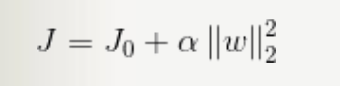
L1正则化即在损失函数的基础上增加参数的1范数：

我们常说，**L1正则化具有参数选择**的作用，直观从图像理解，如下

从贝叶斯角度看，当参数的先验概率符合**拉普拉斯分布**时，最大化后验概率可以得到添加1范数的损失函数。

#### L2正则化

L2正则化即在损失函数的基础上增加参数的2范数：

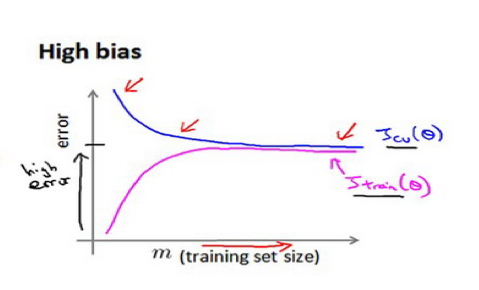
L2正则化具有**权重衰减**的功能：从贝叶斯角度看，当参数的先验概率符合**高斯分布**时，最大化后验概率可以得到添加2范数的损失函数。

# 6、方差vs偏差

当你运行一个学习算法时，如果这个算法的表现不理想，那么多半是出现两种情况：要么是偏差比较大，要么是方差比较大。换句话说，出现的情况要么是欠拟合，要么是过拟合问题。那么这两种情况，哪个和偏差有关，哪个和方差有关，或者是不是和两个都有关？

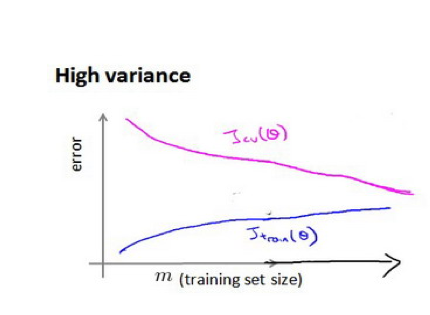
## 6.1 偏差(Bias)

偏差基本对应于欠拟合问题，其表现是模型在训练集和验证集上的误差都比较大，随着数据集的增加，模型在训练集和验证集上的误差表现如下：

在面临高偏差时，我们应该尝试以下的方法：**尝试获得更多的特征，尝试增加多项式特征，尝试减少正则化程度λ**。

## 6.2 方差(Variance)

方差问题对应于过拟合问题，其表现是模型在训练集上误差比较小，而在验证集上的误差远远高于训练集。另一个解释方差问题的角度是，对于同一个形式的模型(比如都是四次回归)，针对不同的训练集，其拟合得到的参数相差很大。随着数据集的增加，模型在训练集和验证集上的误差表现如下：

在面临高偏差时，我们应该采取下面的方法：**获得更多的训练实例，尝试减少特征的数量，尝试增加正则化程度λ**

# 7、支持向量机SVM



<http://blog.pluskid.org/?page_id=683>

## 7.6 LR和SVM的区别

总结了几点LR与SVM的区别，和大家分享：  
1、LR可以输出属于每一类别的概率，SVM则不行  
2、LR是基于概率最大化推导的，而SVM是基于最大化几何间隔推导的  
3、SVM的决策超平面只有少量的支撑向量决定，而LR所有的样本都参与决策面的更新，所以SVM对异常数据并不敏感，LR更加敏感  
4、SVM依赖数据表达的距离测度，所以需要先对数据进行标准化处理，但是LR不需要。

# 8、朴素贝叶斯

朴素贝叶斯是基于贝叶斯定理与特征条件独立假设的分类方法。

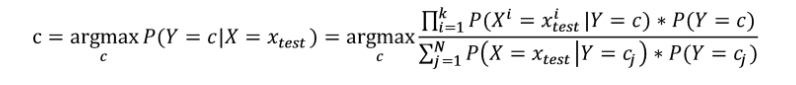
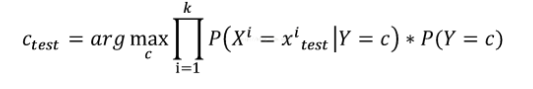
## 8.1 贝叶斯定理

贝叶斯定理是关于随机事件A和B的条件概率的定理，形式如下：

https://upload-images.jianshu.io/upload_images/4155986-a8132b1ec93c71e9.png?imageMogr2/auto-orient/

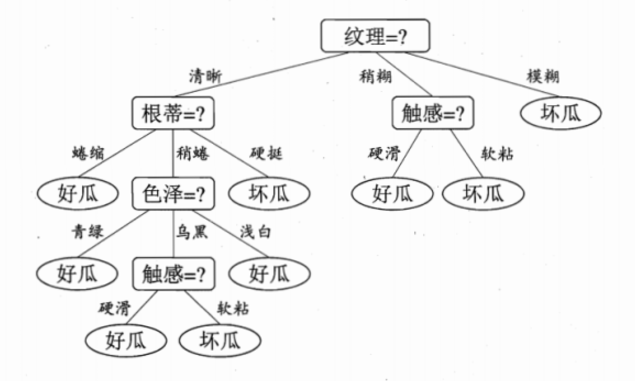
## 8.2 朴素贝叶斯分类

朴素贝叶斯分类的基本思想是：给出待分类项，求解在此项出现的条件下其他各个类别的出现的概率，哪个概率较大就认为待分类项属于哪个类别，用贝叶斯定理表示为(这里的上标表示一维特征)：

分母对于所有的c都是相同的，因此可以省去，故有：我们要做的就是统计对每一个类别来说，每一维特征每个特征出现的频率。频率有可能出现0的情况，我们需要进行\*拉普拉斯平滑操作。

**拉普拉斯平滑**：就是对每类别下所有划分的计数加1，这样如果训练样本集数量充分大时，并不会对结果产生影响，并且解决了频率为0的尴尬局面。

# 9、决策树方法

决策树（Decision Tree）是数据挖掘中一种基本的分类和回归方法，它呈树形结构，在分类问题中，表示基于特征对实例进行分类的过程，可以认为是if−then规则的集合，也可认为是定义在特征空间与类空间上的条件概率分布。下图是一个简单的决策树示例：

决策树模型的主要优点是模型具有可读性，分类速度快。在学习时，利用训练数据，根据损失函数最小化原则建立决策树模型；而在预测时，对新的数据，利用决策树模型进行分类。主要的决策树算法有ID3算法、C4.5算法和CART算法。

一个决策树的学习过程包括三个步骤：特征选择、决策树的生成以及决策树的修剪。

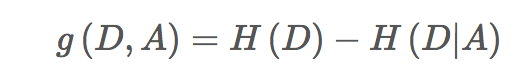
## 9.1 特征选择

**熵**：在信息论与概率统计中，熵表示随机变量不确定性的度量。

**条件熵**H(Y|X)表示在已知随机变量X的条件下随机变量Y的不确定性：

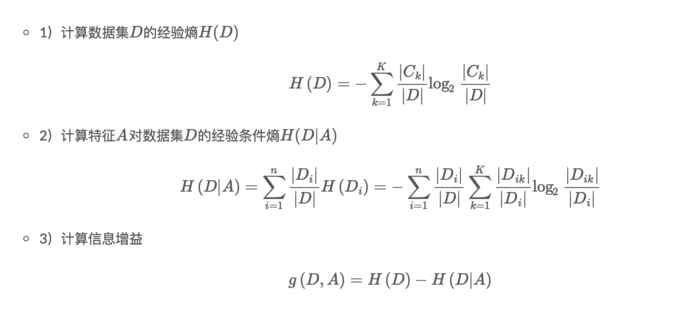
**信息增益**

信息增益表示得知特征X的信息而使得类Y的信息的不确定性减少的程度。信息增益大的特征具有更强的分类能力。特征A对训练数据集D的信息增益g(D,A)，定义为集合D的经验熵H(D)与特征A给定条件下D的经验条件熵H(D|A)之差，即：

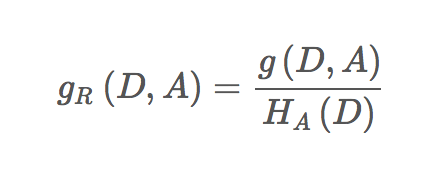


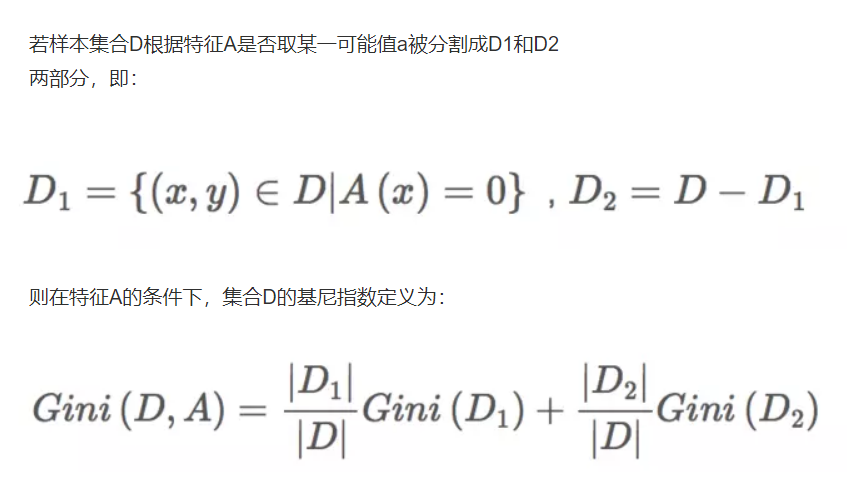
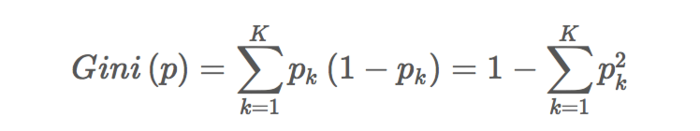
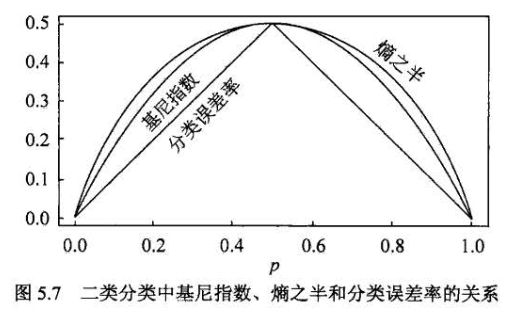
一般地，熵H(Y)与条件熵H(Y|X)之差称为互信息。决策树学习中的信息增益等价于训练数据集中类与特征的互信息。

设训练数据集为D，|D|表示其样本容量，即样本个数。设有K  
个类Ck,k=1,2,⋅⋅⋅,K,根据特征A的取值将D划分为n个子集D1,D2,⋅⋅⋅,Dn，记子集Di中属于类Ck的样本的集合为Dik。  
则信息增益的计算过程如下：

**信息增益比**

以信息增益作为划分训练数据集的特征，存在偏向于选择取值较多的特征的问题。使用信息增益比可以对这一问题进行校正。  
信息增益比表示特征A对训练数据集D的信息增益比。gR(D,A)定义为其信息增益g(D,A)与训练数据集D关于特征A的值的熵HA(D)之比，即：

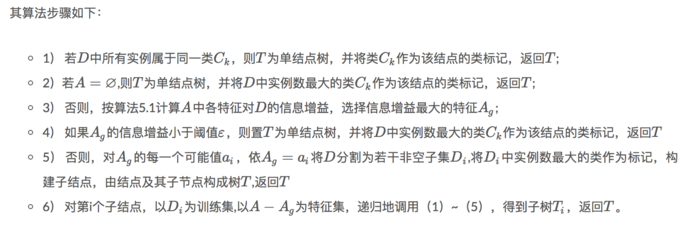


**基尼系数**  
分类问题中，假设有K个类，样本点属于第k类的概率为pk，则概率分布的基尼系数定义为：基尼系数Gini(D)表示集合D的不确定性，表示经A=a分割后集合D的不确定性。基尼系数越大，样本集合的不确定性越大，与熵类似。  
从下图可以看出基尼指数和熵之半的曲线很接近，都可以近似地代表分类误差率。

**9.2 决策树的生成**

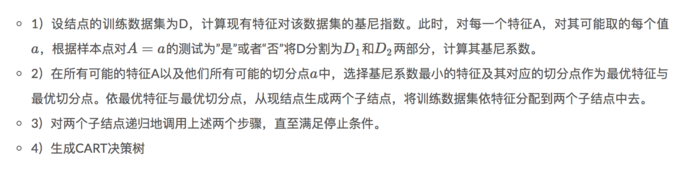
**ID3算法**  
ID3算法的核心是在决策树各个结点上应用信息增益准则选择特征，递归地建构决策树。

具体的算法步骤如下图：

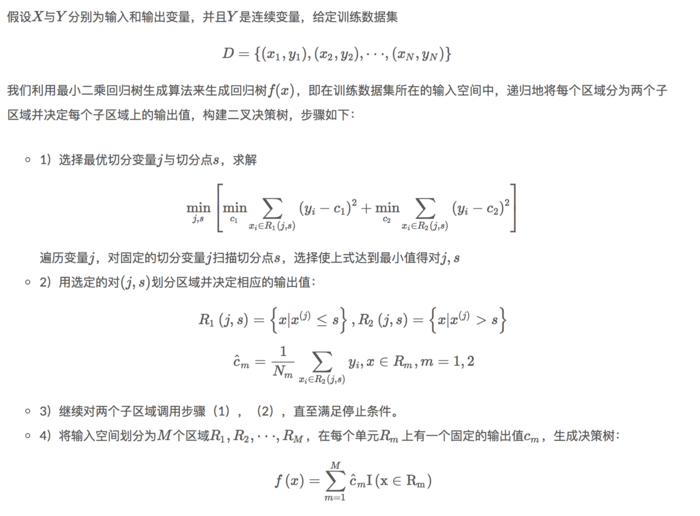
**C4.5**  
与ID3算法相似，C4.5算法对ID3算法进行了改进，C4.5在生成的过程中，用信息增益比来选择特征

**CART**  
分类树与回归树（classification and regression tree，CART）模型（Breiman）既可用于分类也可用于回归。

对分类树用基尼系数（Gini index）最小化准则，进行特征选择，生成二叉树，其具体步骤如下：

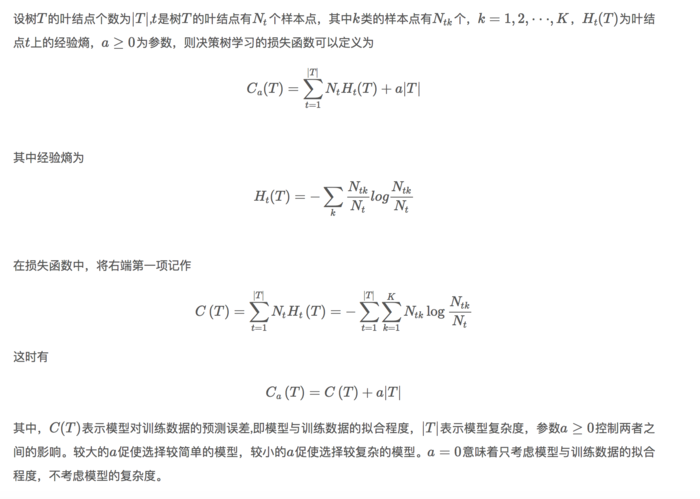


回归树：



## 9.3决策树剪枝

**剪枝**  
决策树的过拟合指的是学习时过多地考虑如何提高对训练数据的正确分类，从而构建出过于复杂的决策树。解决过拟合的办法是考虑决策树的复杂度，对已生成的决策树进行简化，即剪枝（从已生成的树上裁剪调一些子树或叶结点，并将其根结点或父结点作为新的叶结点，从而简化分类树模型）。  
下图展示了决策树剪枝的过程：



# 10、集成学习方法

集成学习的主要思想是利用一定的手段学习出多个分类器，而且这多个分类器要求是弱分类器，然后将多个分类器进行组合公共预测。核心思想就是如何训练处多个弱分类器以及如何将这些弱分类器进行组合。关于集成学习，大致可以分为三类：**Bagging**、**Boosting**和**Stacking**。

本章我们只介绍集成学习方法的基本思想，而具体的实现方法大家可以根据链接自行学习。

## 10.1 Bagging

Bagging即套袋法，其算法过程如下：

1、从原始样本集中抽取训练集.每轮从原始样本集中使用Bootstraping的方法抽取n个训练样本（在训练集中，有些样本可能被多次抽取到，而有些样本可能一次都没有被抽中）.共进行k轮抽取，得到k个训练集.（k个训练集相互独立）  
2、每次使用一个训练集得到一个模型，k个训练集共得到k个模型.（注：根据具体问题采用不同的分类或回归方法，如决策树、神经网络等）  
3、对分类问题：将上步得到的k个模型采用投票的方式得到分类结果；对回归问题，计算上述模型的均值作为最后的结果.

我们常见的Bagging算法是随机森林算法，关于随机森林算法的细节，可以参考博客：[https://www.jianshu.com/p/8f99592658f2](https://www.jianshu.com/p/8f99592658f2" \t "_blank)

## 10.2 Boosting

Boosting是一族可将弱学习器提升为强学习器的算法。关于Boosting的两个核心问题：  
1、在每一轮如何改变训练数据的权值或概率分布？  
通过提高那些在前一轮被弱分类器分错样例的权值，减小前一轮分对样本的权值，而误分的样本在后续受到更多的关注.  
2、通过什么方式来组合弱分类器？  
通过加法模型将弱分类器进行线性组合，比如AdaBoost通过加权多数表决的方式，即增大错误率小的分类器的权值，同时减小错误率较大的分类器的权值。而提升树通过拟合残差的方式逐步减小残差，将每一步生成的模型叠加得到最终模型。

我们常见的Boosting算法有**AdaBoost**，**梯度提升决策树GBDT**，**XgBoost**以及**LightGBM**。大家可以根据下面的参考资料进行学习：

**AdaBoost**：[https://www.jianshu.com/p/f2017cc696e6](https://www.jianshu.com/p/f2017cc696e6" \t "_blank)  
**GBDT**：[https://www.jianshu.com/p/c32af083be5b](https://www.jianshu.com/p/c32af083be5b" \t "_blank)  
**Xgboost**：原论文：[https://arxiv.org/pdf/1603.02754v1.pdf](https://arxiv.org/pdf/1603.02754v1.pdf" \t "_blank)  
博客：[https://blog.csdn.net/github\_38414650/article/details/76061893](https://blog.csdn.net/github_38414650/article/details/76061893" \t "_blank)  
**LightGBM**：LightGBM 中文文档：[http://lightgbm.apachecn.org/cn/latest/index.html](http://lightgbm.apachecn.org/cn/latest/index.html" \t "_blank)

## 10.2 Boosting

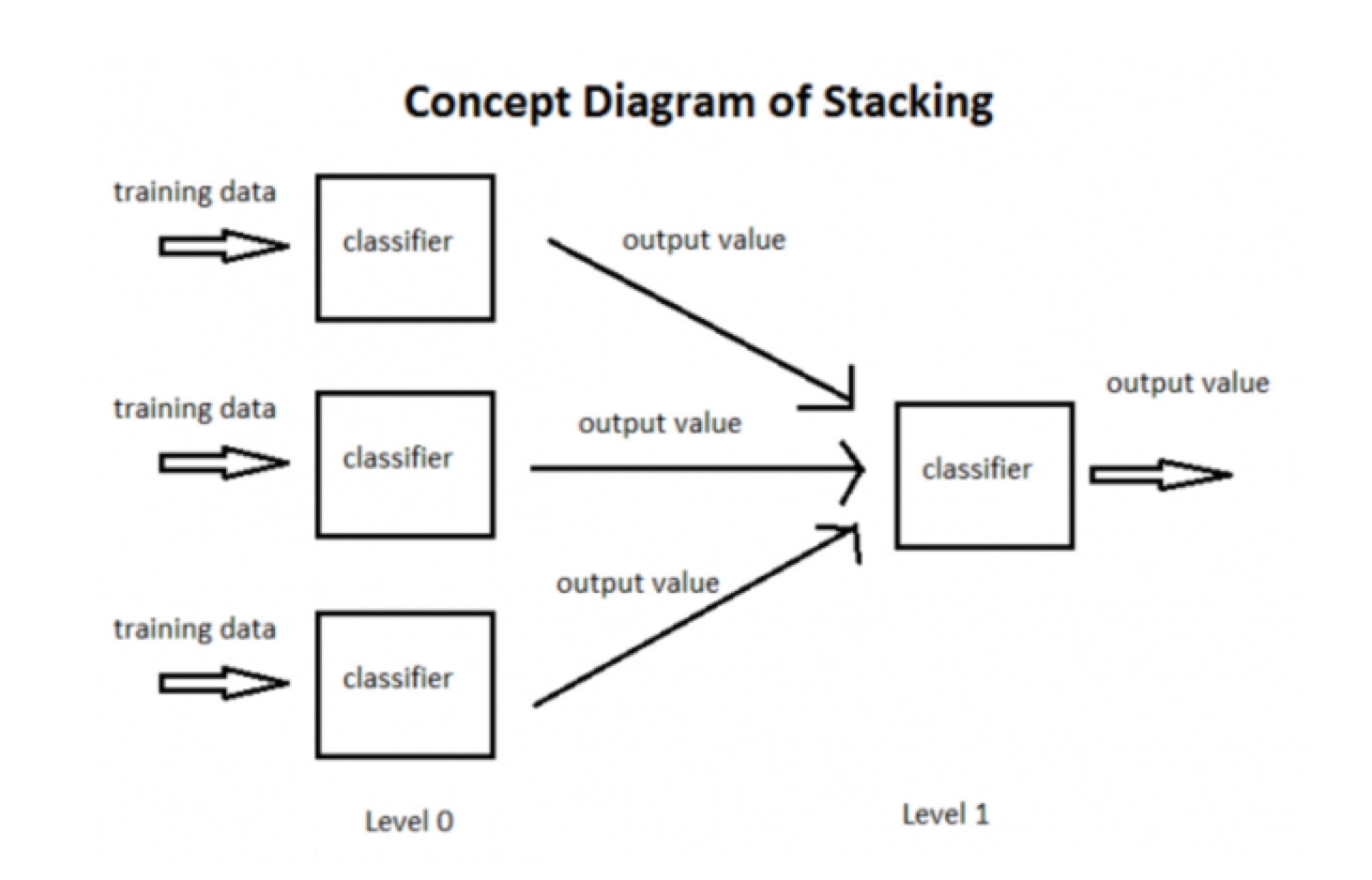
Boosting是一族可将弱学习器提升为强学习器的算法。关于Boosting的两个核心问题：  
1、在每一轮如何改变训练数据的权值或概率分布？  
通过提高那些在前一轮被弱分类器分错样例的权值，减小前一轮分对样本的权值，而误分的样本在后续受到更多的关注.  
2、通过什么方式来组合弱分类器？  
通过加法模型将弱分类器进行线性组合，比如AdaBoost通过加权多数表决的方式，即增大错误率小的分类器的权值，同时减小错误率较大的分类器的权值。而提升树通过拟合残差的方式逐步减小残差，将每一步生成的模型叠加得到最终模型。

我们常见的Boosting算法有**AdaBoost**，**梯度提升决策树GBDT**，**XgBoost**以及**LightGBM**。大家可以根据下面的参考资料进行学习：

**AdaBoost**：[https://www.jianshu.com/p/f2017cc696e6](https://www.jianshu.com/p/f2017cc696e6" \t "_blank)  
**GBDT**：[https://www.jianshu.com/p/c32af083be5b](https://www.jianshu.com/p/c32af083be5b" \t "_blank)  
**Xgboost**：原论文：[https://arxiv.org/pdf/1603.02754v1.pdf](https://arxiv.org/pdf/1603.02754v1.pdf" \t "_blank)  
博客：[https://blog.csdn.net/github\_38414650/article/details/76061893](https://blog.csdn.net/github_38414650/article/details/76061893" \t "_blank)  
**LightGBM**：LightGBM 中文文档：[http://lightgbm.apachecn.org/cn/latest/index.html](http://lightgbm.apachecn.org/cn/latest/index.html" \t "_blank)

## 10.4 Stacking

stacking 就是当用初始训练数据学习出若干个基学习器后，将这几个学习器的预测结果作为新的训练集，来学习一个新的学习器。



具体的原理以及python的实现例子可以参考文章：<https://www.jianshu.com/p/3d2bd58908d0>

# 11、总结一下机器学习中的损失函数

## 11.1 0-1损失函数

0-1损失是指，预测值和目标值不相等为1，否则为0：

## https://upload-images.jianshu.io/upload_images/4155986-252991bffbd834a6.png?imageMogr2/auto-orient/

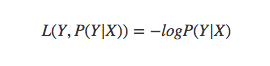
## 11.2 绝对值损失函数

绝对值损失函数为：

https://upload-images.jianshu.io/upload_images/4155986-6964814bb2daf72c.png?imageMogr2/auto-orient/

## 11.3 log对数损失函数

逻辑回归的损失函数就是对数损失函数，log损失函数的标准形式：



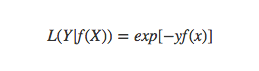
## 11.4 平方损失函数

回归问题中经常使用平方损失函数：

https://upload-images.jianshu.io/upload_images/4155986-3f53be0f8321e992.png?imageMogr2/auto-orient/

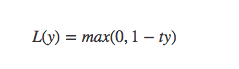
## 11.5 指数损失函数

AdaBoost就是一指数损失函数为损失函数的。 指数损失函数的标准形式：



## 11.6 Hinge损失函数

SVM中使用的是Hinge损失函数：



# 参考资料：

1、线性回归原理和实现基本认识：<https://blog.csdn.net/lisi1129/article/details/68925799>  
2、从贝叶斯角度看L1及L2正则化：  
<https://www.jianshu.com/p/4d562f2c06b8>  
3、梯度下降（Gradient Descent）小结：[https://www.cnblogs.com/pinard/p/5970503.html](https://www.cnblogs.com/pinard/p/5970503.html" \t "_blank)  
4、L1正则化及推导：[https://www.jianshu.com/p/2f60e672d4f0](https://www.jianshu.com/p/2f60e672d4f0" \t "_blank)  
5、支持向量机系列：[http://blog.pluskid.org/?page\_id=683](http://blog.pluskid.org/?page_id=683" \t "_blank)  
6、合页损失函数的理解：[https://blog.csdn.net/lz\_peter/article/details/79614556](https://blog.csdn.net/lz_peter/article/details/79614556" \t "_blank)  
7、李航统计学习方法——算法3朴素贝叶斯法：[https://www.cnblogs.com/bethansy/p/7435740.html](https://www.cnblogs.com/bethansy/p/7435740.html" \t "_blank)  
8、机器学习-决策树：[https://www.jianshu.com/p/8c4a3ef74589](https://www.jianshu.com/p/8c4a3ef74589" \t "_blank)  
9、集成学习系列(七)-Stacking原理及Python实现：[https://www.jianshu.com/p/3d2bd58908d0](https://www.jianshu.com/p/3d2bd58908d0" \t "_blank)  
10、集成学习—boosting和bagging异同[https://www.cnblogs.com/dudumiaomiao/p/6361777.html](https://www.cnblogs.com/dudumiaomiao/p/6361777.html" \t "_blank)  
11、集成学习系列(一)-随机森林算法：[https://www.jianshu.com/p/8f99592658f2](https://www.jianshu.com/p/8f99592658f2" \t "_blank)  
12、集成学习系列(二)-AdaBoost算法原理：  
<https://www.jianshu.com/p/f2017cc696e6>  
13、集成学习系列(五)-GBDT(梯度提升决策树)：  
<https://www.jianshu.com/p/c32af083be5b>  
14、Xgboost论文：[https://arxiv.org/pdf/1603.02754v1.pdf](https://arxiv.org/pdf/1603.02754v1.pdf" \t "_blank)  
15、通俗、有逻辑的写一篇说下Xgboost的原理：[https://blog.csdn.net/github\_38414650/article/details/76061893](https://blog.csdn.net/github_38414650/article/details/76061893" \t "_blank)  
16、LightGBM 中文文档：<http://lightgbm.apachecn.org/cn/latest/index.html>  
17、机器学习总结（一）：常见的损失函数  
<https://blog.csdn.net/weixin_37933986/article/details/68488339>