机器学习 周志华&吴恩达

符号：《机器学习》前面之后拍照放上

缩写：s. t. – subject to约束条件

 tr - trace 其中A为矩阵，即矩阵对角线元素之和

e.g 例如，example

i.e 换而言之id est

i.i.d. 独立同分布independent and identically distributed (i.i.d.)，在概率统计理论中，如果变量序列或者其他随机变量有相同的概率分布，并且互相独立，那么这些随机变量是独立同分布。

周志华：

吴恩达：

如何修改算法可以区分是否真正理解学习算法的

看懂推导每一行，试着自己推导，是学习机器学习的好方法，同样适用于学习

待解决问题

1. 混合高斯模型
2. 矩阵求导

自己理解：

* 凸优化，如线性回归参数的方差为一个二次函数

1. 自己总结：

* 《李宏毅机器学习完整笔记》：<https://mp.weixin.qq.com/s/d4nh1a6uHcQ9dI213Jj5pg>
* 10门课程系统性学会机器学习：<https://mp.weixin.qq.com/s/bHehXzT-jYn-JIpS4zEnSA>

1. 第8课之前是算法基础，之后是理论基础、强化学习，主要是工具，有助于修改算法。
2. SVM和logistic回归问题是对EMR(经验风险最小化，empirical risk minimization）的近似。
3. 非凸性函数求解较困难，如对数函数。
4. 特征选择：找出相关度高的特征，即降低维数，可减小过拟合风险；
   1. 模型选择：模型表达式及特征次数；
5. 线性回归（linear regression）：运用最大似然数
6. 使用极大似然估计预测少量样本容易出现过拟合（overfitting，欠拟合-underfitting），但贝叶斯不会，因为减小了τ，系数接近于0
7. Logistics回归：在预测文本文件时很容易出现过拟合，文本文件特征较多，30000左右，使用高斯贝叶斯规范化，能减小过拟合风险
8. 不要过早优化算法，要在过程中根据实际需要不断优化，编程也是一样。

机器学习也会出现对次要的细节过度优化

垃圾邮件测试误差20%是不可接受的

误差太大，就会想着优化算法，如果不加分析乱尝试，多花3-6个月的情况也比较常见，可能方法：

* 使用更大的训练集 修正高方差
* 使用更小的特征集 修正高方差
* 使用更大的特征集 修正高偏差
* 从标题或文中寻找更好的特征集 修正高偏差
* 使用梯度下降迭代更多次 优化算法
* 尝试使用牛顿方法 优化算法
* 使用不同的值（参数） 优化目标函数
* 尝试SVM 优化目标函数

诊断是否是高方（偏差？）差的方法：

增大训练集，测试误差会减小，另外一般情况下，训练误差会增大，视频11集35min有图

诊断是否是高方差：

比较训练误差和测试差异的大小，差异很大表示方差高

提高更多的训练数据可以缩小他们的差距

吴恩达直升机项目（需进一步修正）：

1. 建立模拟器（气动等），使用计算机仿真
2. 设置成本函数
3. 根据手柄输入操作使用强化学习算法最小化成本函数

错误诊断方法：

* 如果在仿真器中能飞的很好，实物飞不好，说明模型有问题
* 使用优秀的人类飞行员在仿真器中和实际中飞直升机，默认人类飞行员能飞好，如果人类飞行员的成本函数比仿真和实物好，说明强化算法有问题；如果人类飞行员的成本函数比仿真和实物差，说明成本函数有问题

总之，要根据具体情况，设计合适的诊断方法寻找问题的本质，避免浪费无谓的时间，其中比较重要的一点是根据经验对问题形成直观的理解。当需要对数据进行预测时，即一个新的机器学习问题，先分析数据，最好用图表画出来，以下有两种建模方式：学术研究性，花较多时间在算法设计和选择上；商业运用型，根据已有算法和经验，快速建立模型，通过误差诊断，优化算法

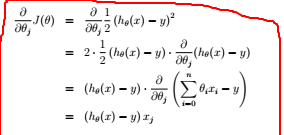
1. 通常需要花费1/3甚至一半的时间设计诊断方法。
2. K-means算法：多次随机初始化运算取最优结果可避免局部最优；
3. （me）算法核心：监督学习时，一般最小化方差来优化参数，通常使用梯度下降；无监督学习时，一般最大化似然性，经常使用高斯混合模型，假设存在标记z，往往不能使导数为0直接求出参数解析解，而是使用EM算法循环求解，E为期望步，求期望，M为最大化步，令导数为0最大化似然性，求出对于参数。
4. PCA(Principal Component Analysis)主成分分析：先判断特征是否需要压缩降维，勿乱用，
5. 批梯度下降（batch gradient descent）：通过求偏导使假设h与y值方差接近于0，即最小化方差，从而使算法收敛，注意局部最优问题及解决方法。一下是几个主要推导公式，加入为x的n次方也类似。

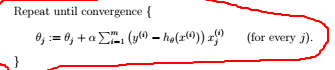
 假设h：

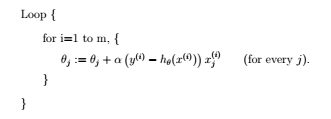
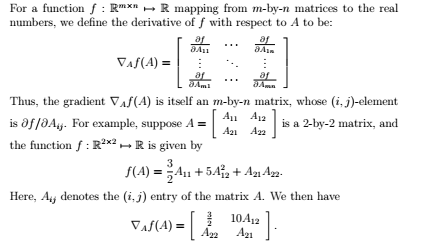
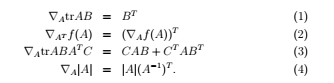
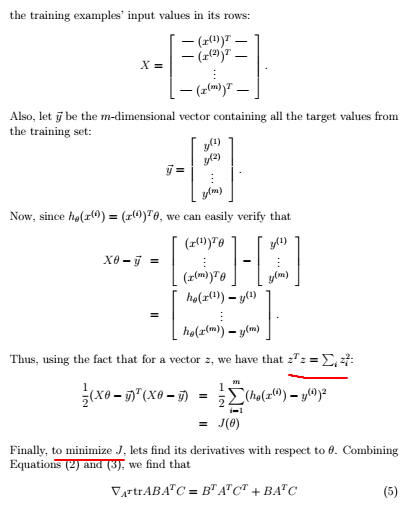
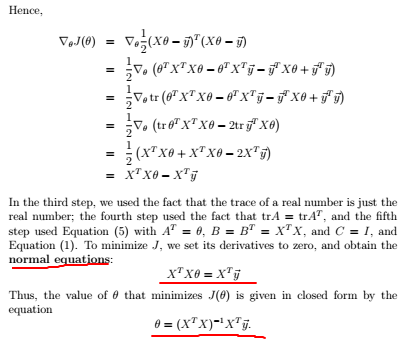
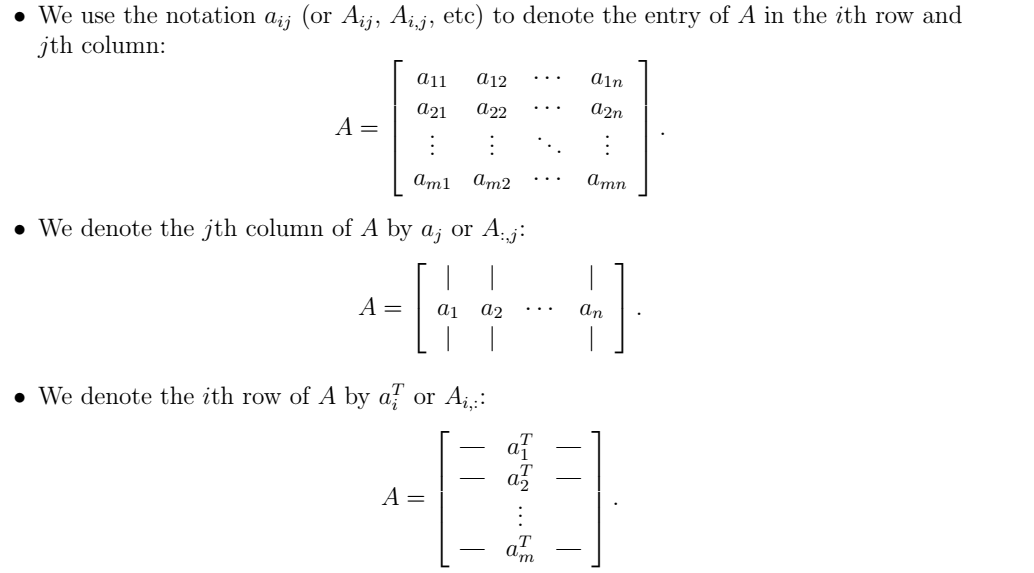
 h和y方差表达式：

 梯度下降优化公式，其中成为步长，自己手动设置，过小收敛时间长，过大可能跨过最小值

对某个参数求导，无论y-h还是h-y计算结果是一样的：



回归目标是让h收敛于y，即方差接近于0，注意这每次都要遍历一次样本集所有方差，计算量会很大：

1. 增量式梯度下降（stochastic gradient descent or incremental  
   gradient descent）：也叫随机梯度下降，与上面公式不一样，去掉累加符号，只用上一次参数进行更新。大规模数据时比批梯度下降收敛快的多，但收敛精度没那么高，可能在最小值附近徘徊，过程有可能往高处走，但总趋势是下降的。
2. 矩阵求偏导，注意求导符号，方程求完导后变为矩阵
   * + 1. Trace推导公式：
       2. 两个合并有：
3. 最小二乘法（least square revisited）：和批梯度下降原理一样，但通过公式直接求出，运用时直接用画红线公式就好了，足以X，矩阵格式。
   1. 参考
4. 非参数算法（non-parametric algorithm）：The term “non-parametric” (roughly) refers to the fact that the amount of stuff we need to keep in order to represent the

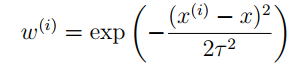
hypothesis h grows linearly with the size of the training set.

如：局部加权回归（Locally weighted linear regression，LWLR），之所以称为非参数，是因为每一次预测都会重新遍历训练集计算，而不是通过一次计算确定所有参数，直接通过参数预测。在直升机和自动驾驶上有运用

最小化

输出 θT x

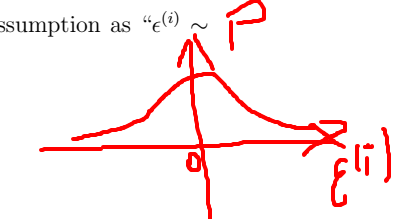
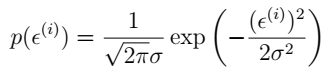
其中一般选用钟型核，与高斯分布类似，但不称为高斯核：

称为波长参数，由自己选定，越小越窄，可通过控制其大小，改变欠拟合和过拟合情况；

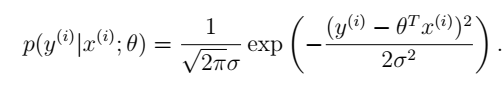
x为想要预测的点，称为查询点，每次预测都需要重新计算拟合，为了提高巨大数据集计算速度，查询Andrew Moore on trees

1. 证明线性回归最小化方差的正确性，可使用用如下式子分析：

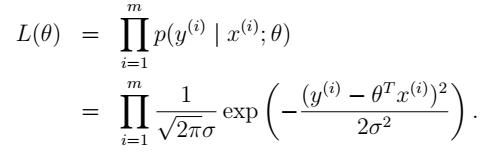
其中，θT为未知的参数，为误差项，可能由于漏选重要特征或随机噪声（孤立和同分布因素）造成，通常假设它服从一种分布，大多数情况会符合高斯分布，且是独立同分布的，即每一个样本服从相同高斯分布、相互独立，然后通过最大化似然函数（换句话说就是最大化当前样本出现的概率）求出θT

 高斯分布如下：

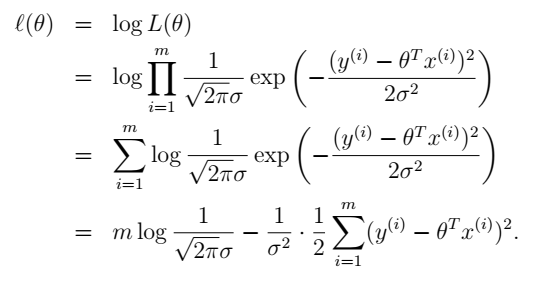
则有，注意式中；号，其前为自变量后为参数，若为，号，则前后为自变量

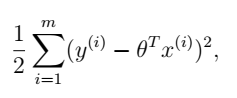


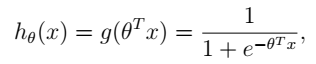
求似然函数：

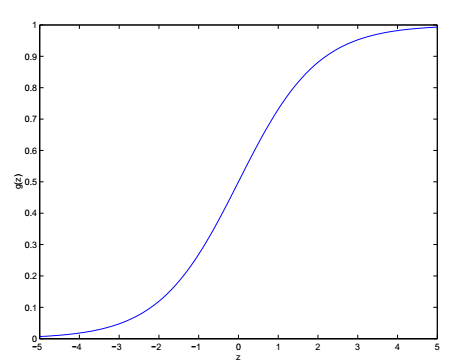


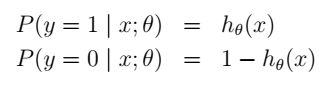
两边大于0，可同时求对数：



可见，最大化似然函数，就是最小化方差

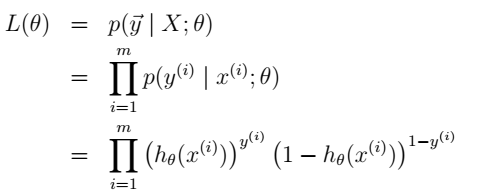
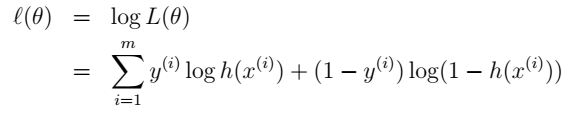
1. Logistic（对数几率）回归：通过使用如下公式作为假设模型，再假设样本是独立的，然后通过最大化似然数求得，其中会用到梯度上升，梯度上升和下降在很多学习算法中都会用到，要学会自己推导，活学活用。

其图像如图，呈S(sigmoid)型：

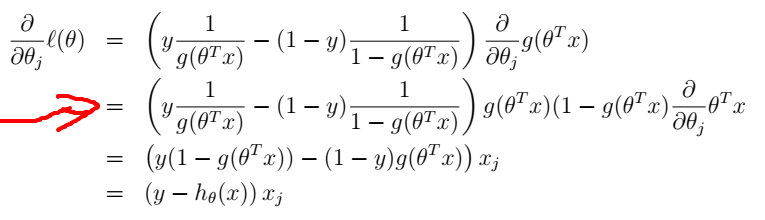
因而可假设如下，对于每个x，要么是或的伯努利分布：

可将两式合并为，可通过画图并将y=0，1的情况验证带入下式验证

样本似然函数为：

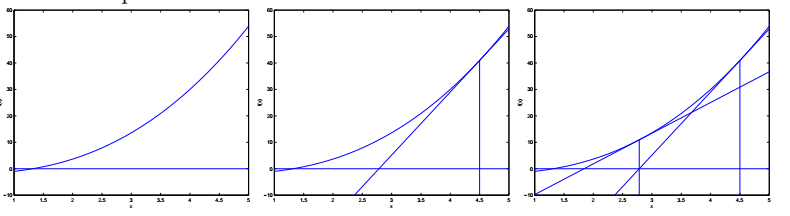
两边求对数得：

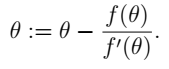
最大化对数似然性，显然，很难直接通过求解析解求出，所以可通过梯度上升方法求出最大值，联想s型图片左右移动显然恰好分类正确处p会得到一个最大值，可想到似然函数是一个开口向下的的凸函数，有空证明

对求偏导得：

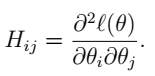
最后得出随机梯度上升规则

1. 牛顿方法（Nweton method）：参数通常初始化为0，收敛速度非常快，不同算法的收敛速度会有很大差异，对于几十个到几百个样本，十几次迭代就能收敛，比梯度上升快得多，对于批梯度上升，缺点：重新计算一次Hessian矩阵，其是一个n\*n维或n+1\*n+1维矩阵，对于几千个特征情况计算量会很大，但对于特征少的计算会很快。推导如下：

为了使函数，可以采用如下图逼近，每次迭代在当前点做切线，然后以切线零点作为下一次迭代起点

根据斜率的特点可推得更新公式为，可尝试图中画出x、f和倒数为负的情况，发现都是适用下面公式的：

因为实际情况，经常是一阶导数（导数为0点就是极大值和极小值处），且往往是多维的，所以延伸到多维和二阶倒数，有：

注意其中梯度和Hessian矩阵，梯度求导在上面有提到，另外，H矩阵如下，可联系高数求二阶偏导数公式格式：

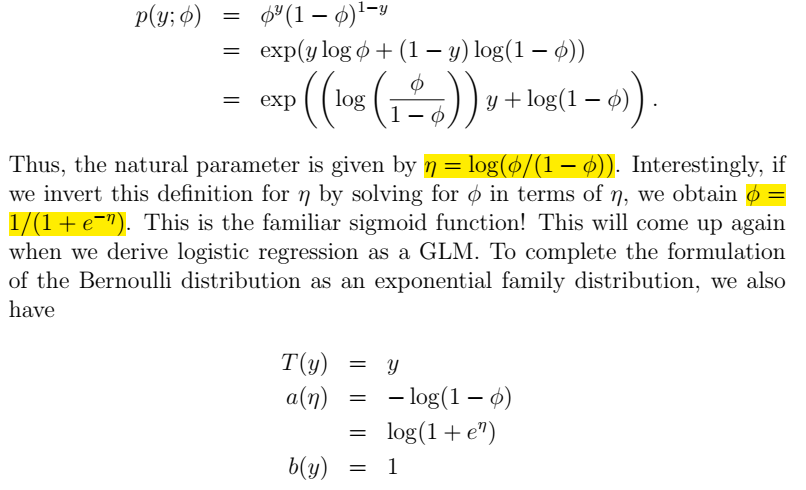
1. 广义线性模型Generalized Linear Models (GLMs)：正则响应函数是什么？？为什么化成这个？Softmax回归？

伯努利分布和高斯分布都属于广义线性模型，都可以化为指数族。模型如下：

其中，

η is called the natural parameter（自然参数） (also called the canonical parameter) of the distribution; T (y) is the sufficient statistic (充分统计量)(for the distributions we consider, it will often be the case that T (y) = y); and a(η) is the log partition function. The quantity e-a(η) essentially plays the role of a normalization constant, that makes sure the distribution p(y; η) sums/integrates over y to 1.

一般，同一种分布的b(y)、T(y)、a(η)是固定的，可通过改变ηT改变分布的概率密度

如，伯努利分布转化步骤如下：

1. 概率论知识理解：

大数定律理解：当一个时间客观存在概率一定时，如果测试次数足够多，且各测试相互独立，则该时间在测试中出现的概率趋近于客观概率

后验概率（方程左侧）：P(c|x)给定测试属性，预测测试样本的分类；先验概率：P（c）由样本集直接得出某一样本的概率；条件概率：P(x|c)某一类c中，x的联合概率分布

1. 生成学习算法（Generative Learning algorithms）：根据不同类样本建立不同模型，如，将A类提出来建立模型，再将B类提出来建立模型。之前的都属于判别式学习算法（discriminative learning algorithms）。怎么建立模型？先验概率后验概率反过来不成立？一个邮件关键词条（特征）可能有50000个，若只判断有无，则有2^50000种可能，选取特征的方法，遍历邮件所有词，遍历之前邮件，查找出现过3次以上的词。laplace平滑，+1。
2. 朴素贝叶斯（）适合文本分类
3. 感知器学习算+法（perceptron learning algorithm）：公式如下：

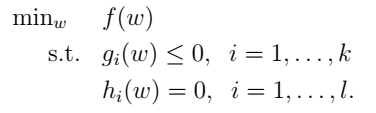
更新公式如下：

其中，0为阈值，为步长，为权值

1. 神经网络：神经网络由感知机构成，相当于一个个神经元。一个感知器只能进行线性分类，不能解决异或问题，若改成输入层、隐层、输出层三层结构，则可进行非线性划分，这也是典型的神经网络结构。如何确定隐层神经元个数是个未决问题，实际运用中常使用“试错法”确定。另外，神经网络存在严重优化问题，存在多个局部最优解，相对于神经网络，有些人更倾向于支持向量机，因为它更高效和无需定制。

训练神经网络的经典方法是误差的逆向传播（errorBackPropagation，BP），具体参见周志华《机器学习》P101.

1. 相关知识点：

* 拉格朗日乘数：使用拉格朗日乘数对等式约束和不等式约束进行对偶，等式约束较简单，求导为0即可，主要讨论不等式约束和等式约束，形式如下：
* 时间复杂度：（https://www.jianshu.com/p/f4cca5ce055a）定义： 存在常数 c，使得当 N >= c 时 T(N) <= f(N)，表示为 T(n) = O(f(n)) 。

算法的时间复杂度，用来度量算法的运行时间，记作: T(n) = O(f(n))。它表示随着 输入大小n 的增大，算法执行需要的时间的增长速度可以用 f(n) 来描述。

综合起来：如果一个算法的执行次数是 T(n)，那么只保留最高次项，同时忽略最高项的系数后得到函数 f(n)，此时算法的时间复杂度就是 O(f(n))。为了方便描述，下文称此为 大O推导法。如：T（n）=3n^3+2n^2+3n+4,T(n)= O(n^3)

时间复杂度分析的基本策略是：从内(如最内层循环)向外分析，从最深层开始分析。如果遇到函数调用，要深入函数进行分析。

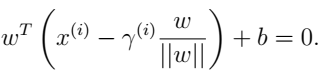
* 坐标上升算法：
* SMO算法：最小指的是改变最少的alpha值
* 应用：可以使用高斯核和多项式和做图像识别，效果和最好的神经网络相当；氨基酸序列分类
* 动态规划算法

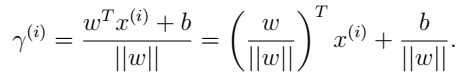
1. 支持向量机（support vector machine SVMS）:目标是找出一个超平面（n-1维），使与超平面相聚最近（最坏情况）的两个不同类的距离最大。经过优化可以用到维数很高的特征算法中，甚至无限维；超平面表达式如下：

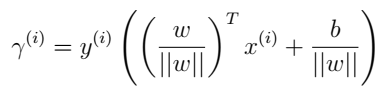
其中，w为超平面法向量，b为位移项。

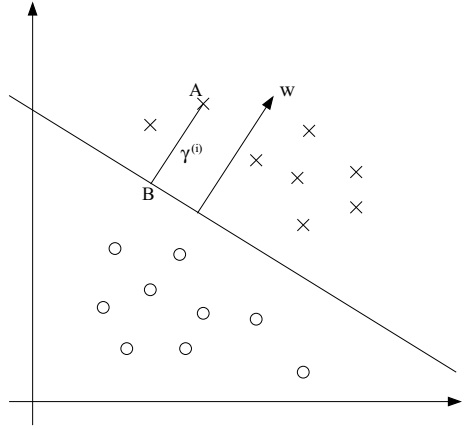
用w=2w，b=2b代入上式，可看成将超平面按比例缩小，因而，超平面参数可以按比例缩放，进行归一化处理

如下下图所示，为A样本到超平面的距离，投影到超平面上坐标可表示为：

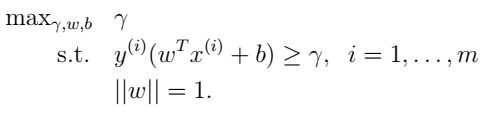
有：

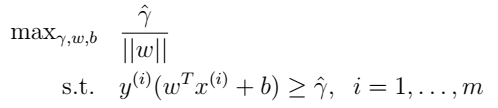
可得：

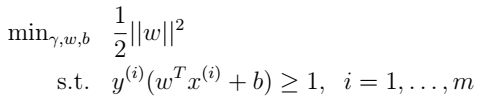
考虑{+1，-1}类有：



可通过以下方法获得最优划分平面：



其中，jjwjj = 1约束确保函数边距等于几何边距（？？），所以我们也有保证所有的几何利润至少γ。因此，解决这个问题将导致(w;b)具有相对于训练集。如果我们能解决上面的优化问题，我们就完成了。但是,\jjwjj = 1”约束是一个讨厌的(非凸的)约束，这个问题肯定存在没有任何格式，我们可以插入到标准优化软件解决。所以，让我们试着把这个问题转化成一个更好的问题。考虑:

通过放缩，可令与超平面平行的上下两个超平面刚好经过两个类最近2个样本，且，则可得到SVM更新方法及约束条件

理解得比较模糊，有空再搞清楚

1. 经验风险最小化（empirical risk minimization (ERM)）：使训练误差最小

方差、偏差和泛化误差的概念（自己理解，需进一步理解，在吴恩达视频后面讲到后实战P152）及其与模型复杂度（次数，一般的模型越复杂拟合程度越高）的关系：偏差指在一次模型训练中，拟合曲线与样本的偏差，拟合程度越高偏差越小；方差指大小相同的训练集训练出来后的拟合曲线之间的方差，拟合程度越高，不同训练集（测试集）测试结果差异越大；泛化误差则是模型实际预测时的误差，一般在方差和偏差折中的时泛化误差最小，左右通常称为欠拟合和过拟合

训练误差（使用m个样本训练时产生误差和的平均值）是对一般误差（泛化误差？一次实际预测出错的概率？）的近似

样本复杂度：为了达到一个特定的错误的界所需要的样本数

1. 聚类的用途：图像处理、数据分析、用图片建立3D模型、电磁大脑、语音处理；强化学习：自动驾驶直升机、小车、机械狗等

数学符号读法：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 大写 | 小写 | 英文注音 | 国际音标注音 | 中文注音 |
| Α | α | alpha | alfa | 阿耳法 |
| Β | β | beta | beta | 贝塔 |
| Γ | γ | gamma | gamma | 伽马 |
| Δ | δ | deta | delta | 德耳塔 |
| Ε | ε | epsilon | epsilon | 艾普西隆 |
| Ζ | ζ | zeta | zeta | 截塔 |
| Η | η | eta | eta | 艾塔 |
| Θ | θ | theta | θita | 西塔 |
| Ι | ι | iota | iota | 约塔 |
| Κ | κ | kappa | kappa | 卡帕 |
| ∧ | λ | lambda | lambda | 兰姆达 |
| Μ | μ | mu | miu | 缪 |
| Ν | ν | nu | niu | 纽 |
| Ξ | ξ | xi | ksi | 可塞 |
| Ο | ο | omicron | omikron | 奥密可戎 |
| ∏ | π | pi | pai | 派 |
| Ρ | ρ | rho | rou | 柔 |
| ∑ | σ | sigma | sigma | 西格马 |
| Τ | τ | tau | tau | 套 |
| Υ | υ | upsilon | jupsilon | 衣普西隆 |
| Φ | φ | phi | fai | 斐 |
| Χ | χ | chi | khai | 喜 |
| Ψ | ψ | psi | psai | 普西 |
| Ω | ω | omega | omiga | 欧米伽 |
| 积分 | 符号 | Sum |  |  |
|  |  |  |  |  |