机器学习是什么，什么流程，需要哪些基础：

|  |
| --- |
| **可解决什么问题**：  1: 分类问题: 根据数据样本上抽取出的特征，判定其属于有限个类别中的哪一个  (**邮件识别**：垃圾或者正常，新闻分类)  2: 回归问题: 根据数据样本上抽取出的特征，预测一个连续值的结果  (电影票房，房价估计)  3:聚类问题: 根据数据样本上抽取出的特征，让样本抱抱团(相近/相关的样本在一团内)  (google 新闻分类)  基础：数学 ＋ 机器学习典型方法 ＋ 编程  **微积分**：理解微分的集合意义，优化理论，例如剃度下降，牛顿发，怎样调参数，怎样得到快的收敛速度，凸优化理论，KKT条件，对偶性等等，  **线性代数**：四个空间， 投影矩阵啊，最小二乘拟合的理解，PCA和SVD  概率论：ML,MAP,贝叶嘶，隐形马尔可夫模型，高斯混合模型  **典型的算法**：   1. 处理分类问题的常用算法包括：逻辑回归(工业界最常用)，支持向量机，随机森林，朴素贝叶斯(NLP中常用)，深度神经网络(视频、图片、语音等多媒体数据中使用)。 2. 处理回归问题的常用算法包括：线性回归，普通最小二乘回归（Ordinary Least Squares Regression），逐步回归（Stepwise Regression），多元自适应回归样条（Multivariate Adaptive Regression Splines） 3. 处理聚类问题的常用算法包括：K均值（K-means），基于密度聚类，LDA等等。 4. 降维的常用算法包括：主成分分析（PCA）,奇异值分解（SVD） 等。 5. 推荐系统的常用算法：协同过滤算法 6. 模型融合(model ensemble)和提升(boosting)的算法包括：bagging，adaboost，GBDT，GBRT 7. 其他很重要的算法包括：EM算法等等。   **机器学习项目的基本流程**：  1:**抽象成数学问题**：  我们明确我们可以获得什么样的数据，目标是一个分类还是回归或者是聚类的问题，如果都不是的话，如果划归为其中的某类问题  2:**获取数据**：对于数据，要注意以下几个点  1数据要有代表性，否则必然会过拟合  2考虑数据大小和内存的问题，估算出其对内存的消耗程度，判断训练过程中内存是否能够放得下。如果放不下就得考虑改进算法或者使用一些降维的技巧了。如果数据量实在太大，那就要考虑分布式了  3:**特征预处理和特征选择**：  筛选出显著特征、摒弃非显著特征，需要机器学习工程师反复理解业务。这对很多结果有决定性的影响。  特征选择好了，非常简单的算法也能得出良好、稳定的结果。这需要运用特征有效性分析的相关技术，如相关系数、卡方检验、平均互信息、条件熵、后验概率、逻辑回归权重等方法。  4:**训练模型和调优**  直到这一步才用到我们上面说的算法进行训练  5: **模型诊断**：确定模型调优的方向与思路  1:过拟合、欠拟合 判断是模型诊断中至关重要的一步。常见的方法如交叉验证，绘制学习曲线等。过拟合的基本调优思路是增加数据量，降低模型复杂度。欠拟合的基本调优思路是提高特征数量和质量，增加模型复杂度。  2: 误差分析 也是机器学习至关重要的步骤。通过观察误差样本，全面分析误差产生误差的原因:是参数的问题还是算法选择的问题，是特征的问题还是数据本身的问题  注意：诊断后的模型需要进行调优，调优后的新模型需要重新进行诊断，这是一个反复迭代不断逼近的过程，需要不断地尝试， 进而达到最优状态  6: **模型融合**：一般来说，模型融合后都能使得效果有一定提升。而且效果很好 |

朴素贝叶斯：

|  |
| --- |
| **为什么非常实用**：  首先，它是一个逆概问题，揭示了P(X|Y)和P(Y|X)的相反方向的条件概率的转换问题。  从贝叶斯公式的发现历史来看，其就是为了处理所谓“逆概”问题而诞生的。比如P(Y|X) 不能通过直接观测来得到结果，而P(X|Y) 却容易通过直接观测得到结果，就可以通过贝叶斯公式从间接地观测对象去推断不可直接观测的对象的情况。  例如：引申一步，基于样本特征去判断其所属标签的概率不好求，但是基于已经搜集好的打上标签的样本（有监督），却可以直接统计属于同一标签的样本内部各个特征的概率分布。因此贝叶斯方法的理论视角适用于一切分类问题的求解。 |

自然常数e：

|  |
| --- |
| ex =1+ x+ x2/2! + x3/3! + …  这个式子可以得到常数e的范围，可以得到possion分布的pdf. |

判别式模型和生成式模型(Discrimitive & Generative)：

|  |
| --- |
| 两者都是分类器的基本概念，但是对于输入x,标签y,  判别式模型（Discrimitive）: 产生条件概率模型，p(y|x)  生成式模型(generative): 产生联合概率模型,p(x,y)  判别式模型 ->生成式模型,但是反过来不行  常见的判别式：  Logistic Regression  SVM  Traditional Neural Network  Nearest Neighbor  CRF  Boosting  Linear Discriminant Analysis  常见的生成式：  Gaussians  Naïve Bayes  Mixtures of Multinomials  Mixtures of Gaussians  Mixtures of Experts  HMMs |

Constraint optimization：

|  |
| --- |
| 约束条件h(x) = 0 , f(x)<0: 注意h(x)必须是线性的，即仿射函数，不然无法解。但是f(x)没有约束。  凸函数的充要条件：对于定义域内的每个点，Hessian matrix是正定的。  对于一般的约束优化的问题，KKT条件只是必要条件，但是对于凸优化问题，kkt就是充要条件。凸优化问题的转化是问题的核心。工业上常采用内点发来求解。  常见的凸优化问题： LP(linear programming) , QCQP，有详细的解答。  对偶问题：拉格朗日函数是凸函数，如果是凸的，那么原问题和对偶问题的gap就是0.  两种情况下求解hessian matrix:  1: 判断局部最小值的时候，当剃度为0，hessian是正定的  2:判断一个函数是否是凸函数，当f二阶可导，并且hessian matrix是正定的。 |

Kkt条件：

|  |
| --- |
| 不等式约束： ci(x)<= 0  等式约束：hi(x)=0  广义拉格朗日函数(generized lagrange function):  L = f(x) + sum(ai\*ci(x)) + sum(bi\*hi(x)), 其中ai >= 0 约束不等条件。  KKT条件是产生极值点的一阶必要条件：  1: L 关于a\* 的偏导为0；  2: L 关于b\* 的偏导为0；  3: L 关于x\* 的偏导为0；  4: 不等式约束要被激活： 即ai\*ci(x\*) = 0 且ai\* >= 0; ci(x\*) <= 0  5:hi(x\*) = 0 |

**特征工**程：

|  |
| --- |
| **特征** => 数据中抽取出来的对结果预测有用的信息。  **什么是好的特征**：  1:好特征意味着更强的灵活性  2: 好特征意味着需要简单的模型  3:好特征意味着更好的结果  数据**采集**的考虑的几点，理解业务是最重要的：  1: 哪些数据对最后结果预测有帮助  2: 数据我们能采集到吗？（例如，年龄）  3: 线上实时计算是否快捷  例如商品推荐，我们可以从三个方面考虑采集信息：  1: 店家的，评分，发货速度等等  2: 商品的，评分，多少人买，点击率（ctr）  3: 用户的埋点，用户的历史购买记录，购买的max和min等等  数据的**存储**化格式：  1: 例如时间戳  2: 大多数情况下，需要关联很多hive表和hdfs文件  数据的**清洗**：对业务理解的核心：  1: 简单的intuitive的判断： 例如人的身高是3m+, 一个人花了几万买了牙膏牙刷  2: 通过组合统计属性判定：例如号称在美国的用户但是其ip一直是大陆，一批买篮球鞋样本群体中有85%是女性  3: 补齐其相应的省值  数据的**采样**：  1:为什么： 很多情况下，正负样本是不平衡的，例如某些疾病的患者与健康人  例如logistic regression是正负样本敏感的  2: 采样的方法：  a: 随机采样（不太好，可能具有偏向性），分层采样（对不同年龄段采样）  b: 正样本>负样本，并且量都挺大，downsampling  c: 正样本>负样本， 且量不大，建议最好采集更多的数据，oversampling等等。  数据的**特征处理**：  1: 归一化，统计max,min,mean,std  2: 离散化，落在哪个分段，特征向量都是一样的，例如数据范围是1-10，对应的特征唯独是10唯，那么我们数据落在哪个区间哪个纬度就是1，其他是0.  3: one-hot编码：根据某一个纬度的变量来进行向多类进行高纬映射，例如某一纬度有三种取值对应用[1 0 0] [0 1 0] [0 0 1],  one-hot 编码的问题就是无法反应相应的权重问题，可以用histogram来统计。  4: hash处理：  **一维**的词带扩充和**n-gram**：  例如一维就是统计每一个词的频率，而二维是将多个词相关联，  “我讨厌你”:二维的词带包含 “我讨厌”, 这样更具有区分度和推荐度  一般搜素引擎的都是多维的词带 |

**模型选择和优化**：

|  |
| --- |
| 模型的几个问题：  过拟合：overfitting, high variance  欠拟合： underfitting, high bias  可以以多项式回归来visualize!  关于三种模型的training accuracy 和validation的accuracy的比较：    一般在工业上过拟合的例子比较多，即test 和validation 的准确率差距比较大。  解决方法：  **过拟合**：  1: 找更多数据来学习  2: 增大正则化系数  3: 减少特征个数  注意：不能以将维解决过拟合问题  **欠拟合**：  1: 找更多的特征  2: 减少正则化系数  **模型的融合**：  1: 群众的力量是伟大的：例如，bagging, random forest  2:一万小时定律,重复的迭代和训练： 例如， Adaboost（每次分配给分错的样本更高的权重，这样就可以逐渐减少错误率）, 逐步增强树(Gradient Boosting Tree)  boosting中有可能是过拟合的，需要具体的准则来控制。Boosting的一种核心思想就是尽量用线性模型来模拟非线性的模型。 |

**关于信息论基础**：

|  |
| --- |
| 自信息： i(x) = -log(p(x))， 对不确定性的测量  熵： 平均不确定性的测量，即维信息量的期望    不确定性越大，熵值越大，如果为定值，熵为0  Notice:对于函数-xlogx, 在（0，1）间的分布为：    因此如果要取到最大熵模型，我们得到的概率分布要尽可能均匀。  互信息(mutual information)：收信者收到信息x后，对y的不确定性的减少量。  即i(y,x) = i(y) – i(y|x) = log(p(y|x)/p(y)) = i(x) – i(x|y)  平均互信息：    **条件熵和联合熵 && 平均互信息**：  条件熵：    联合熵：      **交叉熵**：衡量概率分布的差异性    logistic回归中就是已交叉熵为代价函数  **相对熵（KL Divergence）**: 也是衡量两个概率的分布差异性    **各个熵之间的关系（重要！！！！！！）**：      **信息论和机器学习关系**：  ： |

**EM 算法 && GMM 模型**：

|  |
| --- |
| EM 算法的理论基础：目标函数是最大化似然函数，通过引入隐变量z,使得对数似然函数内有加法，不便于求解最大值，    基础是jensen不等式（函数的期望 >= 期望的函数）    为了使等号成立，    因此，整体框架就是：    **Notice**: 为了更好理解上述的Q 和 Z, 我们可以以一个例子来理解，即GMM的一个例子，例如我们的样本点使一系列的身高，样本中包括男性和女性，都服从  N(μ1,σ1)和N(μ2,σ2),其中四个参数都是未知的，目标使判定身高是属于男性还是女性。  这里，z 就表示样本点是男性还是女性，对应相应的先验概率，而Q(i,z)就是样本点和现在类别的联合概率分布（归一化后的结果），即得到了：    这样，带入EM 算法的框架，并且对上述求偏倒为0，求解得到相应的迭代方程是： |

**推荐系统**：

|  |
| --- |
| **为什么？**：  对用户而言：  －找到好玩的东西  －帮助决策  －发现新鲜事物  对商家而言：  － 提供个性化服务，提高信任度和粘性  － 增加营收  不同的标准：  协同过滤的算法：两种不同类型的处理方式，但是都是求相应的相似度。  例如：对于矩阵U, 大小是n\*m, 代表Uij代表的是第i个用户对第j个商品的评分，注意U是一个稀疏矩阵（因为并不是一个用户都评论了所有的的商品）  Item-based:处理items 间的向量，即列向量。  User\_based:处理用户间的向量，即行向量。    user－based数据量比较庞大，而且这种推荐是不稳定的，因为人对item的评价是会改变的，相对而言item-based比较好一点。  **Cosine 相似度**和**pearson相似度对比**：简单的说，后者在牵着的基础上减去了各纬度的均值，这样做的目的就是消除每个用户对不同商品的偏差，以至于得到更好相似度的测量。典型的例子就是如果某个用户对评分很苛刻，那么其评分就会很低，而另一位用户整体评分又很高，这样如果用cosine相似度就会偏差很大，但是pearson相似度却没有。 |

**决策树**：

|  |
| --- |
| ID3(信息增益):  C4.5(信息增益率):  CART(gini 系数): |

**随机森林**：

|  |
| --- |
| **随机森林的使用原因**：  决策书的泛化能力较弱，其与交叉验证的关系较紧密。  **bootstrap**: 有放回的抽样办法，指依靠自己的资源，称为自助法  **bagging(bootstrap aggregation)**:从样本集中有放回的选出n个样本，对这些n个样本建立分类器，然后投票决定分类的结果。  随机森林就是决策树和bagging的结合。  **Adboost:** adaptive boost, 对不同的重要性的采取不同的权重，  因此adboost就是更新sign(sum(wi\*ci)), 其中wi为权重，ci 为相应的分类器,每次更新我们就更加关心上次分类错误的样本，因此权重更大一点。  注意上述是样本的权重，不是分类器的权重，在分类错的样本上我们加大权重，我们就类似的增多这样的样本。  Adboost会产生多个分类器，每个有不同的权重。  **GBDT:** gradient boosting decision tree,  **GDBT:** gradient decent decision tree.  随机森林过拟合的概率较低，adboost可能产生过拟合，GDBT不仅可以解决分类问题，也可以解决回归问题。  **怎样理解随机森林的随机**：  1: 在样本集中用bootstrap采样（有放回的抽样）n个样本  2: 从所有的属性中随机选择k个属性，建立cart决策树 |

**支持向量机**：

|  |
| --- |
| SVM vs LR 的主要区别:  1: 对于损失函数， svm 对于margin边界以外得’点’，我们认为损失是0， 但是LR 仍然给了一些小的损失值， 因为sigmoid 函数将R上的值映射到(0,1),注意是开区间，因此我们不可能取到0和1， 于是损失函数  -ylog(z) – (1-y)log(1-z) 对于分类正确的样本仍然有一些小的损失值  2: svm 的损失函数有一个约束条件， LR并没有  3: SVM的目标函数并不是损失函数，而LR的目标函数是损失函数  4: svm 不能给出概率的结果，lr 可以  5: lr 的可解释性较强  6: svm 对正则项给出了几何解释，有约束条件的正则， 因此svm是robust的。  7: svm 处理非线性问题上比lr 要快，采用核函数会更好更快  三种分类：  1: hard margin, 线性可分  2: soft margin, 线性不可分  3: kernel function, 核函数找到映射高唯的空间  转化为对偶问题：  max min f(x,lambda) <= min max f(x,lambda)  证明就几行。  但是为什么转化为对偶问题，就简化计算了呢？  因为对偶问题的维度就是1， 而原来的问题维度是d , 实际上都要求偏导，并且对偶函数可以找到支持向量，可以大大简化计算的复杂度。 |

**朴素贝爷斯模型**：

|  |
| --- |
| 贝耶斯模型的一些问题（用垃圾邮箱判断为例）：  1: 对于训练数据集，首先要提取词语，然后统计每一个词在训练集的概率  2: 贝耶斯是逆概率问题，对于 p(‘垃圾’ ｜ ‘某词语’) 不好计算，但是 p(‘某词语’| ‘垃圾‘) 可以从训练样本得到，因此采用naïve bayes 方法  3: 对于分完词的概率计算问题， p(‘词语1’,’词语2’,… ‘词语n’|’垃圾’) =p(词语1’|’垃圾’) \*p(词语2’|’垃圾’)\* … \* p(‘词语n’|’垃圾’), 或者理解这些词语为是条件概率下的独立  4: 平滑问题， 我们可能遇到训练集上不存在的词语在test集上存在了，这样通过贝耶斯我们只能得到了0， 显然不符合世纪情况。 因此需要平滑  多项式平滑：对于重复的词语我们重复计算  贝努力平滑：对于重复的词语我们只计算一次  还有拉普拉斯平滑因子等等  一些tricks:  1: 因为分子是小概率事件的相乘，我们可以转换为log， 这样就是乘法就变成了加法  2: 对于二分类方法（比如判断是否是垃圾邮件， 判定用户是否是男用户）， 我们可以求p(‘垃圾’| ‘邮件’ )／p(‘正常’／ ‘邮件’)， 若大于1，判定为垃圾邮件。  但是对于多分类问题，此种方法不适合了，需要求出后验概率然后再比较，最后分类。  3: 对于某些问题，我们可以选择字典里的top k的词语， 例如5万维， 对于大部分的词语频率很低，那么我们忽律这部分的词语，这样做可能产生误报，但是漏掉垃圾邮件的概率很低，工程上使用较多  4: 对于不同位置的词语，加上不同的权重， 例如标题的权重较大  一些评价：  1: 同样使用与多分类  2: 再分布独立的条件下，效果甚至超过lr  3: 速度较快  4: 对于离散数据的处理，效果不错，但是对于连续的值不是很好  5: 朴素贝耶斯是弱分类器， 不适合bagging 和 boosting.  Naïve Bayes 和 Logistic Regression的异同：  1: 两者都是产生概率，LR 是基于loss function 最优化取到的结果， NR 是基于统计跳过loss function来得最终的概率。  2: NB 比 LR 多了独立分布的假设，  3: 一个是判别模型(LR)，一个是生成模型(NB)  贝耶斯网络的问题： 补充 |

**关于偏度和峰度**：

|  |
| --- |
| Skew: 偏度， 是关于原点的三点距，衡量数据的平衡成都，如果大于0，表示中位数大于均值  Kurl: 峰读， 是关于原点的四点距，衡量数据分布的陡峭程度，若越大，说明越陡峭。 |