本篇文章，我将从过来的人角度介绍下机器学习如何从入门到精通，这里我们谈经验，谈工具，更谈方法论。

## 1.入门

作为初入机器学习的小白，你可能除了一颗好奇的心和一番热血外什么都不没有，当然最好还是希望你能有线性代数、微积分和概率论的基础。你可能会心存顾虑：学过但忘了。不用担心，这种东西不用就会忘，但只要用到，学一学便会；或者说你可能真的没学过，这个也不用担心，只要你真的想学现在也来得及。

好了废话不多说，我们进入入门阶段的正题。入门阶段主要有三个任务：

1. **快速看完周志华的《西瓜书》**；
2. **看吴恩达 Coursera 上的《机器学习》**；
3. **调包跑算法。**

看完这个后可能大家会有很多不解或者很多疑惑。不着急，我们一个一个解答。

### 1.1 快速看完《西瓜书》

**问题一：为什么要选《机器学习》？**

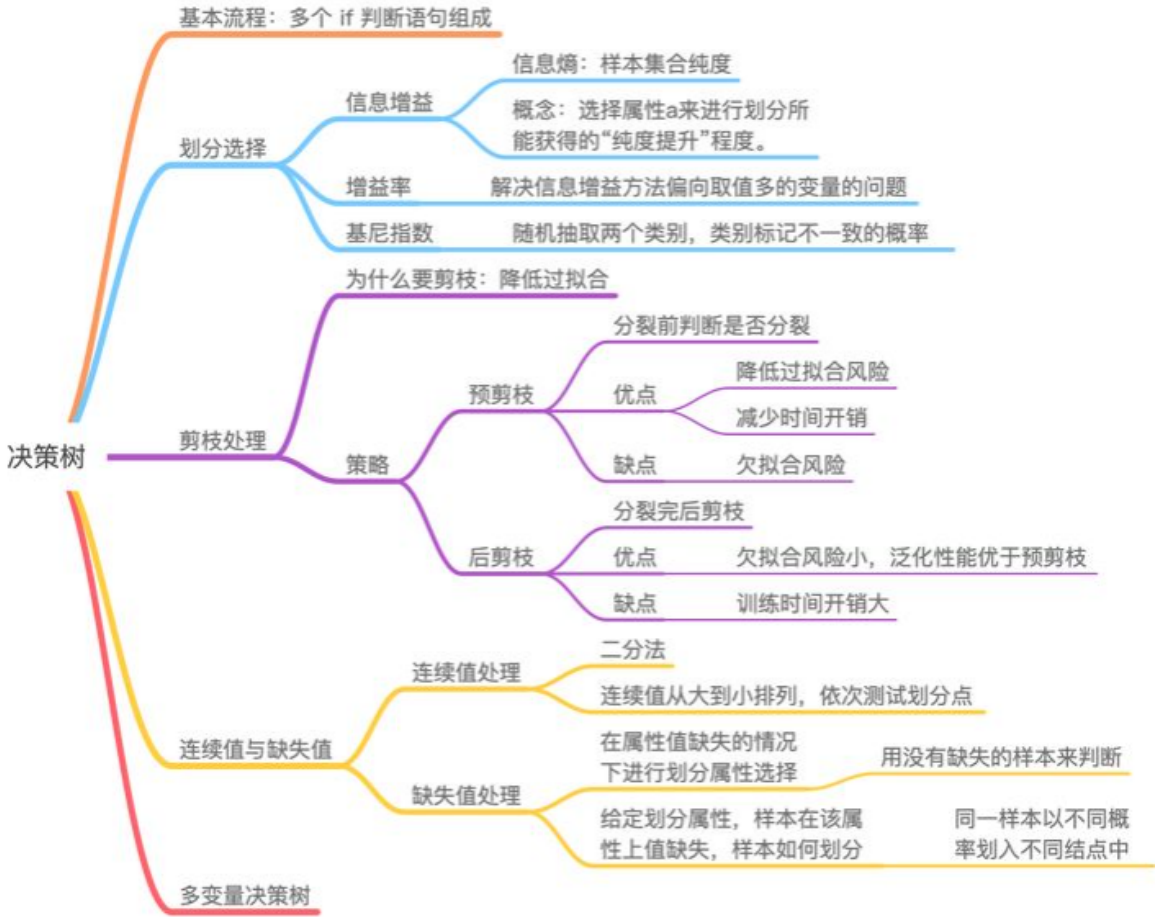
因为这本书真的很不错，作为入门书籍非常合适。同样有名的《统计学习方法》虽然也很不错，但是其对小白来说相对较难。虽然《西瓜书》上面也有公式，但我们在初学的时候太复杂的完全可以先跳过去，并且书中还配有西瓜的例子，可以很好的帮助同学们理解算法的工作过程。

**问题二：为什么要快速看完?**

我这里强调快速，是因为快速真的很重要。因为如果战线拉得太长，如果一开始看的太细又读哪儿哪不懂，人是很容易有排斥心理的。最简单的一个例子，你想想你背英语单词的时候，多少次是从「abandon」开始背起的。每次快要期末考试了，或者四六级，又或者突然想学英语了，就拿起单词书来背，多少次是还没背到 「b」 就 abandon 了。出现这种情况的一大原因是成就感不足，由于在枯燥无味的知识海洋里没能得到**及时的正反馈激励自己继续前行**而出现了 abandon 的情况。特别是对于延迟满足感不强的同学来说，很容易在开始入门的时候就栽起了跟头。

**问题三：怎么快速看完？要有多块？**

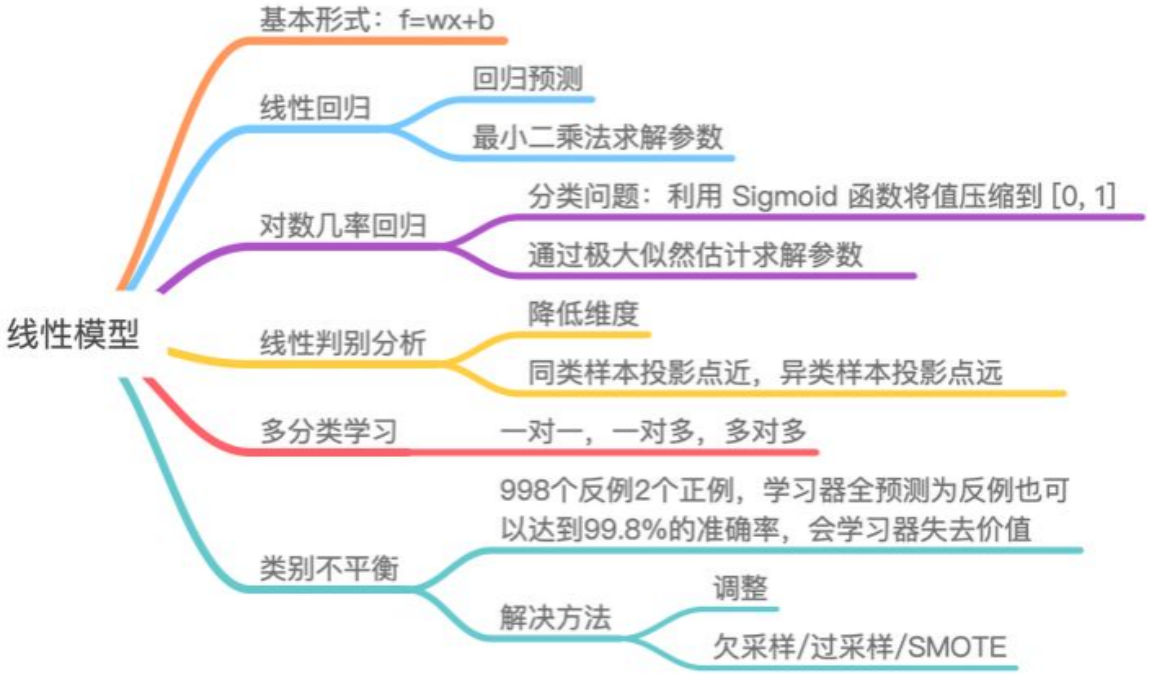
我们这里强调的快速阅读并不是说随便翻一番，更不是说「量子波动速读」（手动狗头）。我们阅读前要带着问题去读，比如说算法能用来干什么，算法的输入输出是什么等等。比较好的一种方法是**用思维导图来记录每一章的脉络或者主线**。以决策树为例，看完《西瓜书》后，我们可能有以下笔记：




通过记录思维导图的方式除了可以帮我们记笔记、理清自己思路外，还有两个非常重要的好处。其中一个是：**学完不至于感觉很虚**。试想一下，如果你快速看完一本书而不记笔记的话，是不是会有一种猪八戒吃人参果的感觉？（内心 OS：西瓜书讲的啥？）如果你有思维导图了，一看就知道决策树是基于多个 if 判断语句组成分类器（当然，也可能会知道决策树可以用于回归），有多种属性划分可以选择，在生成树的时候可以通过剪枝策略降低过拟合，处理连续值的时候有两种算法，处理缺失值的时候会遇到两个问题等等。

另一个好处是：**帮助自己迭代知识**。我们知道产品是需要迭代的，知识当然也是要迭代的。这也是为什么我们要分为入门、进阶和精通三步走的一大原因。罗马不是一日建成的，一口也吃不成胖子。

当然作为初学者，我们的思维导图可能没有那么详细，比如说看完《西瓜书》线性分类那一章，我们可能只有：




不多这样也没有关系，知识是不断迭代的，思维导图只是辅助我们学习的一个工具。

至于时间，大家可以自己把握，我是花了不到一个星期看完，每天 3～4 个小时的样子。

## 1.2 吴恩达 Coursera 上的《Machine Learning》

**问题一：为什么是这门课?**

这里要注意，是 Coursera 的《Machine Learning》，不是 CS229 的《Machine Learning》。推荐原因有以下几点：

- 内容基础，虽然是英文，但是有中文字幕；
- 质量很棒，12 万个评分，平均得分 4.9，要知道满分才 5 分；
- 每节课时间不长，且课后都有小练习；
- 除了自己记笔记外，还可以再网上找到其他同学上课记的笔记，可以作为自己笔记的补充；
- 《任务一》「太爽了」，可以稍微痛苦点，跟着吴恩达推推公式，加深理解，为进阶做准备。

可能有同学还会推荐其他老师的课，甚至手机了一堆资料。我只想说，别这样。我们要拒绝仓鼠病（仓鼠喜欢囤东西），东西太多会分散我们的注意力，还会让我们有一种感觉学了好多的错觉。Less is more，学精品就行。

**问题二：这门课也要快速吗?**

千万别快！一定要认真做笔记（或思维导图）！做完笔记还要去查看别人的笔记，补充下自己遗漏的点。对待好的课程一定要怀着虔诚的心态去看，千万别浪费第一次看视频的机会，后面再看可能就没那么有耐心了。

**问题三：那我要花多久时间?**

总共十一门课程，没门课程都有预估时间，加上整理笔记看其他人的笔记，自己可以预估一下。建议在两个星期以内完成。

这门课虽然是英文的，而且还有公式，但是由于吴恩达老师讲的很好，所以其难度是比我们看西瓜书要低的。

## 1.3 调包跑算法

**问题一：为什么要调包跑算法?**

原因三有：

- **提高感性认识**：我记得刚学计算机的时候，老师特别喜欢对我们说一句话：自己写一写、跑一跑、感受一下。只有动手操作一下，才能切身感受；
- **提高代码能力**：计算机是一门工程性很强的学科，不仅要学理论知识，更要有不错的代码实现能力；
- **成就感**：大家不觉得能够「预测」未知的事情本身就是一件很有成就感的事吗?

**问题二：数据从哪儿来?**

数据集来源太多了，但还是那句话：不要有仓鼠病，我们需要什么数据去拿什么数据即可。比较建议的数据源有 Kaggle 和阿里天池，可以去看看那些入门级的竞赛，跑跑里面的数据集。

**问题三：跑哪些算法?**

既然是调包，大部分都会用到 Sklearn，里面的算法都可以试一试，能跑通一个，再换另一个的速度也挺快的，最好覆盖常用的算法如：LR、SVM、Random Forest 等。

**问题四：花多久时间?**

这个要看自己的基础了，如果代码能力还行的话，一天就能搞定了，如果代码能力不 ok，可能要两三天。如果代码能力不行的话，记得去看别人写好的代码，而不是自己瞎捣鼓。

## 1.4 总结

总结一下入门阶段三个任务的**目的**：

1. 《西瓜书》：了解并认识算法，对机器学习有个大概的了解；
2. 《Machine Learning》：在吴恩达老师的带领下推推公式，加深对算法的理解，为进阶学习做准备；
3. 调包跑算法：对算法有一个感性的认识，并提高动手能力。

**预计时间**：3～4 个星期。

第一阶段所用到的：

- **工具**：《西瓜书》；《统计学习方法》；思维导图；做笔记推荐 Markdown 或者手写笔记；
- **方法论**：及时正反馈；拒绝仓鼠病。

PS：如果有多余的时间也可以看一下《集体智慧编程》，这本书很简单并且有很完整的代码实现，可以根据自己的喜好选择阅读还是动手实践。

## 2.进阶

完成入门阶段的学习后，我们成功从机器学习小白晋升到机器学习入门，在看到 SVM，LR 之类的词时再也不会陌生了。但是革命尚未成功，同志仍需努力。

那在进阶阶段，我们学什么呢？以及如何学习呢？

我们来看下进阶学习的任务：

1. **学习《统计学习方法》《西瓜书》**；
2. **学习书本以外的常用算法**；
3. **学习特征工程，打比赛。**

同样的，我们来解答下同学们的疑惑。

## 2.1 学习《统计学习方法》《西瓜书》

**问题一：为什么又学习《西瓜书》？**

因为我们第一遍学的不仔细，我们第一遍只是简单的过一遍了。尚且，好的书不应该只看一遍。要相信每次阅读都会给我带来不一样的体验。

**问题二：两本谁先谁后?**

建议先读《统计学习方法》（建议看第二版），并以《统计学习方法》的目录来重新整理自己的知识体系，以《西瓜书》为辅，《统计学习方法》看到什么算法了，就去《西瓜书》里面看一看，补充一下。这样做的原因很简单：以《统计学习方法》的目录构建我们的知识体系更加合理。

**问题三：该怎么去学习?**

这个阶段不能再像入门阶段那样看到复杂的公式就跳过了。但是我也知道很多同学的数学基础不好，而书中的部分公式又有很多跳过的步骤，读起来比较困难。这时候我们就要借用群体智慧了：

- 首先，我们要学会利用 CSDN/知乎等专业的平台，这些平台上的那些高赞高阅读量的博文都是经过时间的筛选出来的高质量文章，很多很不错的博文写的比书上写得更加有利于人理解，当然也可以看看一些论文，非常经典比如说：吴信东的《数据挖掘十大算法》；
- 其次，github 上有一个比较有名的《南瓜书》，这本书是对《西瓜书》中公式推导的补充，周志华老师也推荐过；
- 最后，利用好我们的思维导图，在阅读书或博客的时候，记得完善自己的笔记，这个真的很重要!

另外，《统计学习方法》比较厚，特别是第二版的，我们要学会取舍，要分清哪些算法需要看，哪些算法可以放一放。比如说，我以后是想做推荐、广告 CTR、风控之类的事情，那我们就可以把 MCMC、LDA 之类的算法先放一放，而把有限的时间放在 LR、SVM、决策树上。

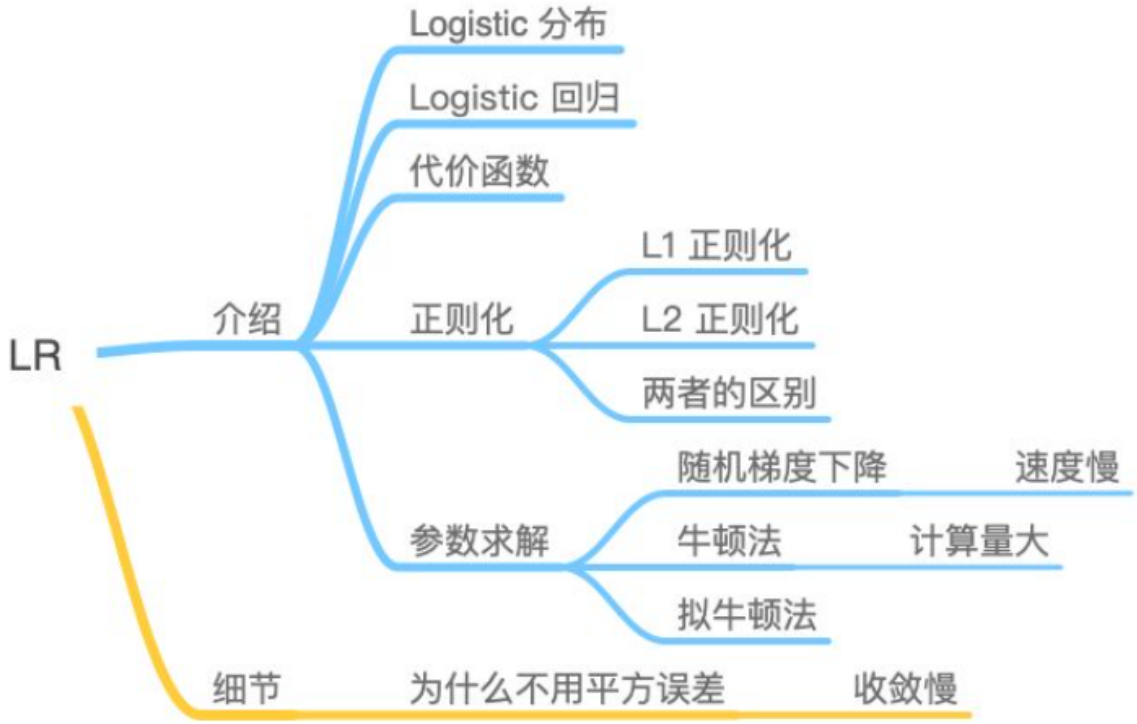
学习的时候涉及到大量的数学原理，缺啥补啥即可，千万别放下机器学习去学数学去了。不是说数学不重要，而是我们要明白，每个阶段的目标和任务，目标导向性有助于我们学习。

**问题五：学到什么程度?**

我觉得进阶学习的程度至少要包括：

1. 对需要学习的算法，掌握其数学原理，学会公式推导，掌握算法的优缺点；
2. 对不需要学习的算法，了解算法的作用，使用场景；
3. 掌握算法细节，比如说 SVM 的 SMO 和核函数；K-means 的调优与改进，如何确定 k 值，如何证明其收敛性；LR 的参数更新方法，L1 和 L2 两种正则化，参数求解如何优化等；

以 LR 为例，学完后的思维导图可以是这样的：




虽然看起来内容没之前多，但仔细看的话，内容深度还是和入门阶段的有明显区别的。

**问题六：大概花多久?**

这个要根据每个人选择的算法个数而定，平均一个算法可以花两到四天去学习掌握。

## 2.2 学习书本以外的常用算法

**问题一：有哪些算法?**

比较有名的有 Boosting 三兄弟： XGBoost、LightGBM 和 CatBoost；

其次，FM、FFM 及其衍生的算法也很有名；

**问题二：为什么要学习?**

首先，这些算法是比较常用的算法，性能好，鲁棒性强，经常用于各大小公司，和各大比赛（Kaggle、阿里天池等）；

其次，我们可以学一学这些算法的思想，有助于我们算法调优，比如说我们用 XGBoost 的时候，因为学习了 LightGBM 所以知道对连续特征离散化有助于 XGBoost 提升泛化性；

**问题三：如何学习?**

首先，建议先去看高质量的博文，对三个算法有个大概的了解后再去看论文，这样做的好处在于读完博客可以对算法有个大概的了解，再去看论文也更容易入手，此外原汁原味的论文读起来也是一种享受；

其次，不建议每个都学，学有余力的同学可以多学一点，时间紧张的东西挑重点的来学，要懂得取舍。

## 2.3 打比赛

**问题一：为什么要打比赛?**

首先，打比赛是我们从理论通往实践的重要一步，比赛的数据相对工作后的数据来说会干净很多，在从理论过渡到实际工作的过程中有很大的帮助。

其次，「没有免费的午餐」，即没有算法能完美的解决所有的问题。通过打比赛，我们可以加深对算法的理解，熟悉算法的擅长领域。

最后，打比赛的时候，我们还会学到机器学习中一个非常重要的领域——特征工程。我们常说，特征决定上限，而算法只是去逼近这个上界，从这句话就能看出特征的重要性。所以只会算法是远远不够的，还要学会做特征工程。而且，不同的算法可能需要不同的特征工程，这也会加深我们对算法的理解。

**问题二：如何学习?**

首先，强烈推荐 Kaggle，因为 Kaggle 上有很多大佬分享的经验帖子，不仅有 baseline 算法，还会有数据分析、特征工程的经验和心路历程，非常建议大家去学习实践；

其次，特征工程是一个比较偏经验的，作为刚接触特征工程的同学来说，可以一遍打比赛，一遍看看相关书籍和博客：有一些比较有名的特征工程书《特征工程入门与实践》，还有很多同学整理的特征工程相关博客也可以看看；

最后，还是那句话，记得整理笔记！因为博客和书籍中有很多重复的内容，只有认真梳理了后才能形成我们自己的学习体系。

**问题三：预计时间**

短则看几本书看看 Demo 跑跑算法几个星期就搞定了，长则打几场比赛需要大半年的时间。

## 2.4 总结

总结一下进阶阶段三个任务的**目的**：

1. **学习《统计学习方法》《西瓜书》**：通过推公式加深对算法的理解，掌握算法细节；
2. **学习书本以外的常用算法**：对书本内容的一个补充，学习和掌握目前比较流行的算法；
3. **打比赛**：将算法学习从理论过渡到实践，同时学习特征工程。

**预计时间**：这个时间波动会比较大，但无论花多长时间，这个阶段能获取到的知识都是非常多的。

**工具**：《统计学习方法》；《西瓜书》；《南瓜书》；《特征工程入门与实践》；博客；思维导图；论文；

**方法论**：目标导向性；学会取舍；当然也要记得正反馈。

## 3.精通

这一阶段就不太好谈了，因为目前我也在这个阶段里遨游，但我个人角度来说，有以下几个任务：

1. **深入了解算法细节，对比各大算法差异**；
2. **学习算法在工业中的应用，看源代码**；
3. **看论文**；
4. **写博客。**

同理，我们一个一个的来解释。

## 3.1 深入了解算法细节，对比各大算法差异

**问题一：我都会推公式了，还不够细吗?**

首先，我们说的推公式很多时候都是看得懂公式，而不是知道为什么要这么推，即知其然而不知其所以然。比如说，我们会用拉格朗日乘子法来求解 SVM 的对偶问题，但是我们可能不知道拉格朗日乘子法原本是求解灯饰优化问题的，而不是 SVM 中的不等式优化问题，如果从将拉格朗日乘子法从不等式优化扩展到等式优化呢?

除此之外还有：为什么一定要转换成对偶问题呢？为什么要把 SVM 从 minmaxf(x) 转化到 maxminf(x) 呢？原来的公式不香吗？所以推公式只是加深了对算法的理解，这个阶段的的学习还要多问为什么。为什么要这么做，为什么问多了之后可能我们就知道了：首先转化为对偶问题后可以简化问题复杂度，原来是和样本维度和样本数量有关的问题，现在只和样本数量有关了；其次转换为对偶问题可以方便的引入核函数，将数据从低维映射到高维，更容易求得最优解。。

其次，很多看似很简单的东西，如我们在进阶阶段学习到的：LR 引入 L1 和 L2 范式可以通过约束权重值来减少过拟合风险。但我们有没有想过为什么权重小就能减少过拟合风险？为什么 L1 和 L2 范式有效?

到这个阶段后，我们可能就知道了：降低权重是为了降低算法的复杂度，防止出现变量值波动引起的预测精度问题。我们也会知道，L1 和 L2 范式之所以有效，是因为引入了「零均值的拉普拉斯分布」和「零均值的正态分布」这种先验知识。

总之，这个阶段的任务就是要多问自己为什么。为什么是这样的？为什么这样行？那样不行吗？只有多问自己为什么，带着辩证的思维去学习，才能更加深刻的了解算法细节。

**问题二：为什么要对比各大算法的差异?**

独木不成林，我们要将学习到的零零散散的知识从点练成线，再将线拼接成面，甚至堆积成三维立体。

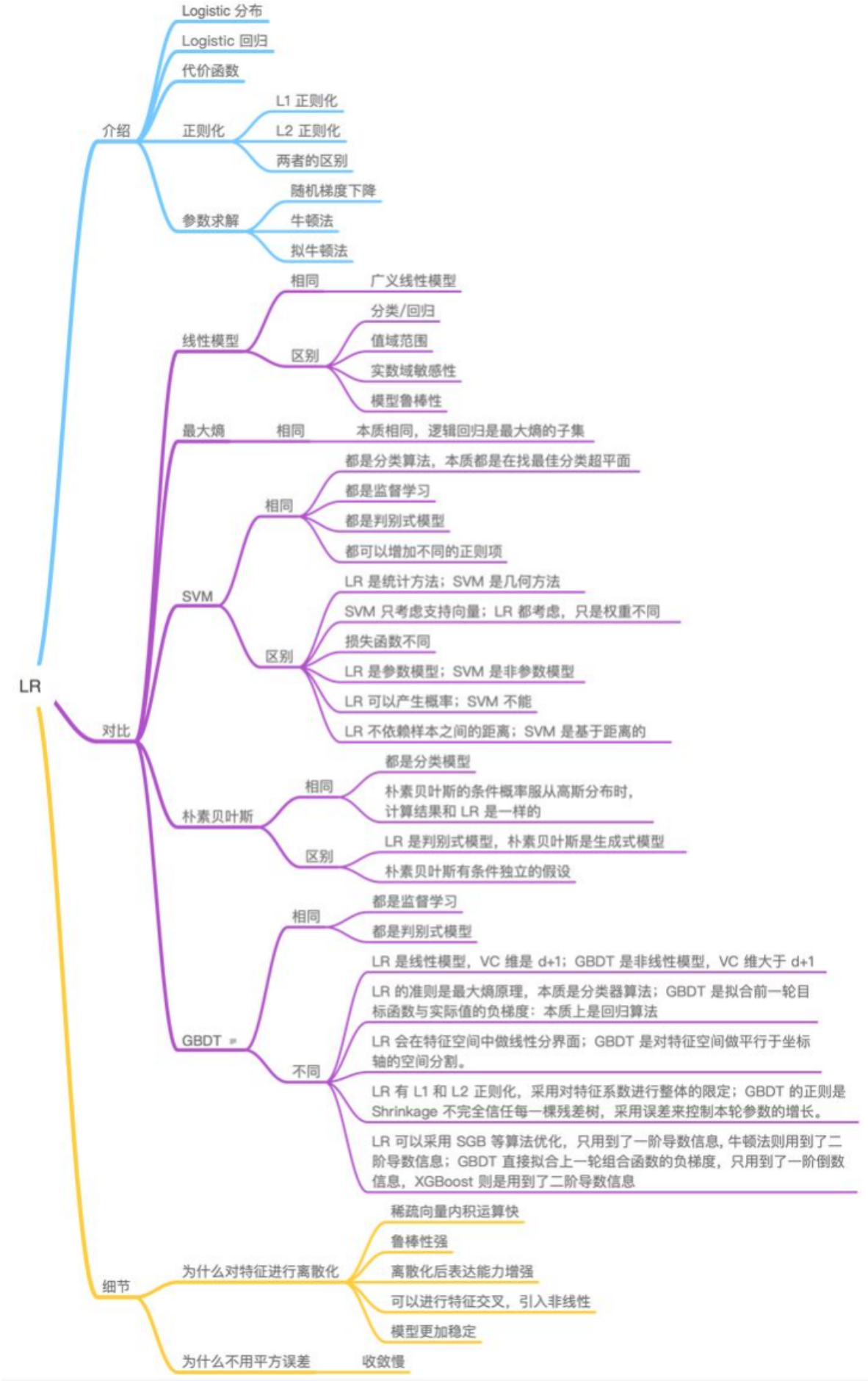
在进阶学习的过程中，我们知道 XGBoost 比 LR 效果要好，SVM 就算数学理论很完备工业界还是不常用。但我们不知道其为什么？为什么 XGBoost 比 LR 好？为什么 SVM 不常用？LR 和 贝叶斯为什么会应用于不同的领域？为什么「没有免费的午餐」？

了解不同算法之间的差异，及优缺点，比说 LR 与 SVM 的区别与联系、LR 与最大熵算法的区别与联系等等。

通过学习各大算法的差异与联系，可以帮助我们更加深刻的了解各类算法。

**问题三：要学习到什么程度?**

这个不好说，精通阶段仁者见仁智者见智，以 LR 为例，我觉得至少应该学到这种阶段：


LR
介绍
Logistic 分布
Logistic 回归
代价函数
正则化
L1 正则化
L2 正则化
两者的区别
参数求解
随机梯度下降
牛顿法
拟牛顿法
对比
线性模型
相同
广义线性模型
区别
分类/回归
值域范围
实数域敏感性
模型鲁棒性
最大熵
相同
本质相同，逻辑回归是最大熵的子集
SVM
相同
都是分类算法，本质都是在找最佳分类超平面
都是监督学习
都是判别式模型
都可以增加不同的正则项
区别
LR 是统计方法；SVM 是几何方法
SVM 只考虑支持向量；LR 都考虑，只是权重不同
损失函数不同
LR 是参数模型；SVM 是非参数模型
LR 可以产生概率；SVM 不能
LR 不依赖样本之间的距离；SVM 是基于距离的
朴素贝叶斯
相同
都是分类模型
朴素贝叶斯的条件概率服从高斯分布时，
计算结果和 LR 是一样的
区别
LR 是判别式模型，朴素贝叶斯是生成式模型
朴素贝叶斯有条件独立的假设
GBDT
相同
都是监督学习
都是判别式模型
不同
LR 是线性模型，VC 维是 d+1；GBDT 是非线性模型，VC 维大于 d+1
LR 的准则是最大熵原理，本质是分类器算法；GBDT 是拟合前一轮目
标函数与实际值的负梯度：本质上是回归算法
LR 会在特征空间中做线性分界面；GBDT 是对特征空间做平行于坐标
轴的空间分割。
LR 有 L1 和 L2 正则化，采用对特征系数进行整体的限定；GBDT 的正则是
Shrinkage 不完全信任每一棵残差树，采用误差来控制本轮参数的增长。
LR 可以采用 SGB 等算法优化，只用到了一阶导数信息，牛顿法则用到了二
阶导数信息；GBDT 直接拟合上一轮组合函数的负梯度，只用到了一阶倒数
信息，XGBoost 则是用到了二阶导数信息
细节
为什么对特征进行离散化
稀疏向量内积运算快
鲁棒性强
离散化后表达能力增强
可以进行特征交叉，引入非线性
模型更加稳定
为什么不用平方误差
收敛慢

## 3.2 学习工业应用，看源码

**问题一：为什么要学习工业应用?**

因为我们大多数人最终还是要去工业界的，而工业界的思维和我们在学校的思维是有差距的。

首先，工业界更加强调落地，而非精度。平常我们打比赛、发论文之类的，都把算法搞的非常复杂，但是在工业界面对千万级甚至亿级的数据，太复杂的算法不利于我们算法落地。所以工业界是需要在精度和算力中取一个平衡点的；

其次，工业界看中速度，算法一定要快。这个时候算法如何并行化实现就显得尤为重要。比较通用的方法是：数据并行和特征并行，大家可以自行了解下。

**问题二：为什么要看源码?**

看源码一来是为了加深自己对算法的理解，二来是学习下算法在落地时的优化策略或是一些小Trick，比如说如何排序？用 O(nlogn)的快排吗？如何采样？如何加权采样？随机数怎么算？等等问题。这些都是需要从源码中学到的。

## 3.3 看论文

这个应该不用问为什么了吧，很多先进的机器学习/深度学习算法都来源于论文，稍微老一点的类似GDBT+LR、GDBT+FM 这种组合（GDBT 可以换成 XGBoost），年轻一点的有 DIN、DIEN 等。

看论文有助于我们了解最新的算法的模型，跟上前沿的研究轨迹。

## 3.4 写博客

其实写博客进阶的时候也可以写，甚至入门的时候也可以写。无非就是把笔记整理一下发出去。但需要注意，我们这里说的博客是写给别人看的，如果是写给自己看的随便你怎么写。但现在你的目标受众是其他读者，你就要本着负责任的心态，把你的博文写好。

这个写好博客说的是，至少要让读者明白你在说什么吧。很多会觉得麻烦，但其实不是，你之前的学习是一个输入，消化了多少你并不知道，而写博客是一种输出，你写出来多少就代表你会了多少。

有同学可能要说了：」我会但我写不出，我是茶壶里煮饺子。「

抱歉，千万别这么想，其实你就是不会，有这种思想很大程度上是因为自己眼高手低，没有脚踏实地的原因。

另外，写博客也是整理思路的一个过程，可以很好的帮助我们学习整理归纳，加深自己对算法的了解。

### 3.5 总结

最后，我们来总结下精通阶段的几个任务：

1. **深入了解算法细节，对比各大算法差异**：将点串成面，加深算法理解；
2. **学习工业应用，看源码**：了解工业应用和算法落地时的优化，加深算法理解；
3. **看论文**：了解前沿算法；
4. **写博客**：梳理知识，加深算法了解。

**工具**：源码；博客；论文；

**方法论**：以点串线带面；知识的输入输出；

## 4.学习资料推荐

贪心学院 aijiaoai《机器学习》，这个课程覆盖了 16 大经典算法讲解，20 个实操案例，8 大项目作业。并且是玩游戏通关的轻松学习模式，让你在 2 个月的时间快速入门机器学习。

入门：

第一步：第一遍西瓜书：一周半线上完成。

进度：1，2周

- 做思维导图。

算法是干什么的，有哪些应用同时搜索，算法的输入输出。一些基本的特征整理。

第二步：ng coursera上面的《machine learning》。

进度：3，4周。

认真做笔记or思维导图。推导笔记一定要自己写一遍。+思维导图。

第三步：掉包跑算法。测试测试，和ng的课程一起就行。用kaggle or 阿里天池的数据源，跑一跑就行。

进阶：

第一步：统计学方法（主）+西瓜书（辅）。前者梳理脉络，可以有步骤的选择着读。

进度：两个月左右。之后就准备学校的复习的相关事宜了。从明天开始做起吧。

统计学方法，有些可以不用看