14年有人做过一个实验[1]，比较在不同数据集上（121个），不同的分类器（179个）的实际效果。

论文题为：Do we Need Hundreds of Classifiers to Solve Real World Classification Problems?

写给懒得看的人：

**没有最好的分类器，只有最合适的分类器。**

随机森林平均来说最强，但也只在9.9%的数据集上拿到了第一，优点是鲜有短板。

SVM的平均水平紧随其后，在10.7%的数据集上拿到第一。

神经网络（13.2%）和boosting（~9%）表现不错。

**数据维度越高**，随机森林就比AdaBoost强越多，但是整体不及SVM[2]。

**数据量越大**，神经网络就越强。

### 近邻 (Nearest Neighbor)

典型的例子是KNN，它的思路就是——对于待判断的点，找到离它最近的几个数据点，根据它们的类型决定待判断点的类型。

它的特点是完全跟着数据走，没有数学模型可言。

适用情景：

需要一个特别容易解释的模型的时候。

比如需要向用户解释原因的推荐算法。

### 贝叶斯 (Bayesian)

典型的例子是Naive Bayes，核心思路是根据条件概率计算待判断点的类型。

是相对容易理解的一个模型，至今依然被垃圾邮件过滤器使用。

适用情景：

需要一个比较容易解释，而且不同维度之间相关性较小的模型的时候。

可以高效处理高维数据，虽然结果可能不尽如人意。

### 决策树 (Decision tree)

决策树的特点是它总是在沿着特征做切分。随着层层递进，这个划分会越来越细。

虽然生成的树不容易给用户看，但是数据分析的时候，通过观察树的上层结构，能够对分类器的核心思路有一个直观的感受。

举个简单的例子，当我们预测一个孩子的身高的时候，决策树的第一层可能是这个孩子的性别。男生走左边的树进行进一步预测，女生则走右边的树。这就说明性别对身高有很强的影响。

适用情景：

因为它能够生成清晰的基于特征(feature)选择不同预测结果的树状结构，数据分析师希望更好的理解手上的数据的时候往往可以使用决策树。

同时它也是相对容易被攻击的分类器[3]。这里的攻击是指人为的改变一些特征，使得分类器判断错误。常见于垃圾邮件躲避检测中。因为决策树最终在底层判断是基于单个条件的，攻击者往往只需要改变很少的特征就可以逃过监测。

受限于它的简单性，决策树更大的用处是作为一些更有用的算法的基石。

### 随机森林 (Random forest)

提到决策树就不得不提随机森林。顾名思义，森林就是很多树。

严格来说，随机森林其实算是一种集成算法。它首先随机选取不同的特征(feature)和训练样本(training sample)，生成大量的决策树，然后综合这些决策树的结果来进行最终的分类。

随机森林在现实分析中被大量使用，它相对于决策树，在准确性上有了很大的提升，同时一定程度上改善了决策树容易被攻击的特点。

适用情景：

**数据维度相对低（几十维），同时对准确性有较高要求时。**

因为不需要很多参数调整就可以达到不错的效果，基本上不知道用什么方法的时候都可以先试一下随机森林。

### SVM (Support vector machine)

SVM的核心思想就是找到不同类别之间的分界面，使得两类样本尽量落在面的两边，而且离分界面尽量远。

最早的SVM是平面的，局限很大。但是利用核函数(kernel function)，我们可以把平面投射(mapping)成曲面，进而大大提高SVM的适用范围。

提高之后的SVM同样被大量使用，在实际分类中展现了很优秀的正确率。

适用情景：

SVM在很多数据集上都有优秀的表现。

相对来说，SVM尽量保持与样本间距离的性质导致它抗攻击的能力更强。

和随机森林一样，这也是一个拿到数据就可以先尝试一下的算法。

### 逻辑斯蒂回归 (Logistic regression)

逻辑斯蒂回归这个名字太诡异了，我就叫它LR吧，反正讨论的是分类器，也没有别的方法叫LR。顾名思义，它其实是回归类方法的一个变体。

回归方法的核心就是为函数找到最合适的参数，使得函数的值和样本的值最接近。例如线性回归(Linear regression)就是对于函数f(x)=ax+b，找到最合适的a,b。

LR拟合的就不是线性函数了，它拟合的是一个概率学中的函数，f(x)的值这时候就反映了样本属于这个类的概率。

适用情景：

LR同样是很多分类算法的基础组件，它的好处是输出值自然地落在0到1之间，并且有概率意义。

因为它本质上是一个线性的分类器，所以处理不好特征之间相关的情况。

虽然效果一般，却胜在模型清晰，背后的概率学经得住推敲。它拟合出来的参数就代表了每一个特征(feature)对结果的影响。也是一个理解数据的好工具。

### 判别分析 (Discriminant analysis)

判别分析主要是统计那边在用，所以我也不是很熟悉，临时找统计系的闺蜜补了补课。这里就现学现卖了。

判别分析的典型例子是线性判别分析(Linear discriminant analysis)，简称LDA。

（这里注意不要和隐含狄利克雷分布(Latent Dirichlet allocation)弄混，虽然都叫LDA但说的不是一件事。）

LDA的核心思想是把高维的样本投射(project)到低维上，如果要分成两类，就投射到一维。要分三类就投射到二维平面上。这样的投射当然有很多种不同的方式，LDA投射的标准就是让同类的样本尽量靠近，而不同类的尽量分开。对于未来要预测的样本，用同样的方式投射之后就可以轻易地分辨类别了。

使用情景：

判别分析适用于高维数据需要降维的情况，自带降维功能使得我们能方便地观察样本分布。它的正确性有数学公式可以证明，所以同样是很经得住推敲的方式。

但是它的分类准确率往往不是很高，所以不是统计系的人就把它作为降维工具用吧。

同时注意它是假定样本成正态分布的，所以那种同心圆形的数据就不要尝试了。

### 神经网络 (Neural network)

神经网络现在是火得不行啊。它的核心思路是利用训练样本(training sample)来逐渐地完善参数。还是举个例子预测身高的例子，如果输入的特征中有一个是性别（1:男；0:女），而输出的特征是身高（1:高；0:矮）。那么当训练样本是一个个子高的男生的时候，在神经网络中，从“男”到“高”的路线就会被强化。同理，如果来了一个个子高的女生，那从“女”到“高”的路线就会被强化。

最终神经网络的哪些路线比较强，就由我们的样本所决定。

神经网络的优势在于，它可以有很多很多层。如果输入输出是直接连接的，那它和LR就没有什么区别。但是通过大量中间层的引入，它就能够捕捉很多输入特征之间的关系。卷积神经网络有很经典的不同层的可视化展示(visulization)，我这里就不赘述了。

神经网络的提出其实很早了，但是它的准确率依赖于庞大的训练集，原本受限于计算机的速度，分类效果一直不如随机森林和SVM这种经典算法。

使用情景：

数据量庞大，参数之间存在内在联系的时候。

当然现在神经网络不只是一个分类器，它还可以用来生成数据，用来做降维，这些就不在这里讨论了。

### Rule-based methods

它里面典型的算法是C5.0 Rules，一个基于决策树的变体。因为决策树毕竟是树状结构，理解上还是有一定难度。所以它把决策树的结果提取出来，形成一个一个两三个条件组成的小规则。

使用情景：

它的准确度比决策树稍低，很少见人用。大概需要提供明确小规则来解释决定的时候才会用吧。

### 提升算法（Boosting）

接下来讲的一系列模型，都属于集成学习算法(Ensemble Learning)，基于一个核心理念：三个臭皮匠，顶个诸葛亮。

翻译过来就是：当我们把多个较弱的分类器结合起来的时候，它的结果会比一个强的分类器更

典型的例子是AdaBoost。

AdaBoost的实现是一个渐进的过程，从一个最基础的分类器开始，每次寻找一个最能解决当前错误样本的分类器。用加权取和(weighted sum)的方式把这个新分类器结合进已有的分类器中。

它的好处是自带了特征选择（feature selection），只使用在训练集中发现有效的特征(feature)。这样就降低了分类时需要计算的特征数量，也在一定程度上解决了高维数据难以理解的问题。

最经典的AdaBoost实现中，它的每一个弱分类器其实就是一个决策树。这就是之前为什么说决策树是各种算法的基石。

使用情景：

好的Boosting算法，它的准确性不逊于随机森林。虽然在[1]的实验中只有一个挤进前十，但是实际使用中它还是很强的。因为自带特征选择（feature selection）所以对新手很友好，是一个“不知道用什么就试一下它吧”的算法。

### 装袋算法（Bagging）

同样是弱分类器组合的思路，相对于Boosting，其实Bagging更好理解。它首先随机地抽取训练集（training set），以之为基础训练多个弱分类器。然后通过取平均，或者投票(voting)的方式决定最终的分类结果。

因为它随机选取训练集的特点，Bagging可以一定程度上避免过渡拟合(overfit)。

在[1]中，最强的Bagging算法是基于SVM的。如果用定义不那么严格的话，随机森林也算是Bagging的一种。

使用情景：

相较于经典的必使算法，Bagging使用的人更少一些。一部分的原因是Bagging的效果和参数的选择关系比较大，用默认参数往往没有很好的效果。

虽然调对参数结果会比决策树和LR好，但是模型也变得复杂了，没事有特别的原因就别用它了。

### Stacking

这个我是真不知道中文怎么说了。它所做的是在多个分类器的结果上，再套一个新的分类器。

这个新的分类器就基于弱分类器的分析结果，加上训练标签(training label)进行训练。一般这最后一层用的是LR。

Stacking在[1]里面的表现不好，可能是因为增加的一层分类器引入了更多的参数，也可能是因为有过渡拟合(overfit)的现象。

使用情景：

没事就别用了。

（修订：

[@庄岩](https://www.zhihu.com/people/7ca4a0bcf00428d17e73e7ad0bd51e4d)

提醒说stacking在数据挖掘竞赛的网站kaggle上很火，相信参数调得好的话还是对结果能有帮助的。

[http://blog.kaggle.com/2016/12/27/a-kagglers-guide-to-model-stacking-in-practice/](https://link.zhihu.com/?target=http%3A//blog.kaggle.com/2016/12/27/a-kagglers-guide-to-model-stacking-in-practice/)

这篇文章很好地介绍了stacking的好处。在kaggle这种一点点提升就意味着名次不同的场合下，stacking还是很有效的，但是对于一般商用，它所带来的提升就很难值回额外的复杂度了。）

### 多专家模型（Mixture of Experts）

最近这个模型还挺流行的，主要是用来合并神经网络的分类结果。我也不是很熟，对神经网络感兴趣，而且训练集异质性（heterogeneity）比较强的话可以研究一下这个。

**讲到这里分类器其实基本说完了。讲一下问题里面其他一些名词吧。**

**最大熵模型 (Maximum entropy model)**

最大熵模型本身不是分类器，它一般是用来判断模型预测结果的好坏的。

对于它来说，分类器预测是相当于是：针对样本，给每个类一个出现概率。比如说样本的特征是：性别男。我的分类器可能就给出了下面这样一个概率：高（60%），矮（40%）。

而如果这个样本真的是高的，那我们就得了一个分数60%。最大熵模型的目标就是让这些分数的乘积尽量大。

LR其实就是使用最大熵模型作为优化目标的一个算法[4]。

**EM**

就像最大熵模型一样，EM不是分类器，而是一个思路。很多算法都是基于这个思路实现的。

@刘奕驰 已经讲得很清楚了，我就不多说了。

**隐马尔科夫 (Hidden Markov model)**

这是一个基于序列的预测方法，核心思想就是通过上一个（或几个）状态预测下一个状态。

之所以叫“隐”马尔科夫是因为它的设定是状态本身我们是看不到的，我们只能根据状态生成的结果序列来学习可能的状态。

适用场景：

可以用于序列的预测，可以用来生成序列。