PRML（Pattern Recognition And Machine Learning）读书会

第十四章 Combining Models

主讲人 网神

（新浪微博：@豆角茄子麻酱凉面）

QQ 群 177217565

读书会微信公众平台请扫描下面的二维码

网神(66707180) 18:57:18

大家好，今天我们讲一下第 14 章 combining models，这一章是联合模型，通过将多个模型以某种形式结合起来，可以获得比单个模型更好的预测效果。包括这几部分：

committees, 训练多个不同的模型，取其平均值作为最终预测值。

boosting: 是 committees 的特殊形式，顺序训练 L 个模型，每个模型的训练依赖前一个模型的训练结果。

决策树：不同模型负责输入变量的不同区间的预测，每个样本选择一个模型来预测，选择过程就像在树结构中从顶到叶子的遍历。

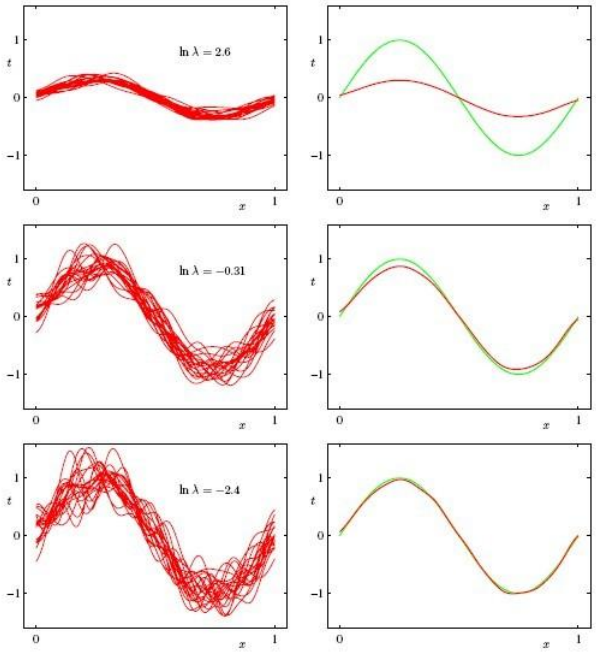
conditional mixture model 条件混合模型：引入概率机制来选择不同模型对某个样本做预测，相比决策树的硬性选择，要有很多优势。

本章主要介绍了这几种混合模型。讲之前，先明确一下混合模型与 Bayesian model averaging 的区别，贝叶斯模型平均是这样的：假设有 H 个不同模型 h，每个模型的先验概率是 p(h)，一个数据集的分布是：

$$p(\mathbf{X}) = \sum_{h=1}^{H} p(\mathbf{X}|h)p(h).$$

整个数据集 X 是由一个模型生成的，关于 h 的概率仅仅表示是由哪个模型来生成的 这件事的不确定性。而本章要讲的混合模型是数据集中，不同的数据点可能由不同模型生成。看后面讲到的内容就明白了。

首先看 committes，committes 是一大类，包括 boosting，首先将最简单的形式，就是讲多个模型的预测的平均值作为最后的预测。主要讲这么做的合理性，为什么这么做会提高预测性能。从频率角度的概念，bias-variance trade-off 可以解释，这个理论在 3.5 节讲过，我们把这个经典的图 copy 过来：



这个图大家都记得吧，左边一列是对多组数据分别训练得到一个模型，对应一条 sin 曲线，看左下角这个图，正则参数 lamda 取得比较小，得到一个 bias 很小，variance 很大的一个模型 。每条线的 variance 都很大，这样模型预测的错误就比较大，但是把这么多条曲线取一个平均值，得到右下角图上的红色线，红色线跟真实 sin 曲线也就是蓝色线 基本拟合。所以用平均之后模型来预测，variance 准确率就提高了很多，这是直观上来看，接下里从数学公式推导看下：

有一个数据集，用 bootstrap 方法构造 M 个不同的训练集 bootstrap 方法就是从数据集中随机选 N 个放到训练集中，做 M 次，就得到 M 个训练集，M 个训练集训练的到 M 个模型，用 $y_m(\mathbf{x})$ 表示，那么用 committees 方法，对于某个 x，最终预测值是:

$$y_{\mathrm{COM}}(\mathbf{x}) = \frac{1}{M}\sum_{m=1}^{M} y_m(\mathbf{x}).$$

我们来看这个预测值是如何比单个 $y_m(\mathbf{x})$ 预测值准确的，假设准确的预测模型是 h(x)，那么训练得到的 y(x)跟 h(x)的关系是：

$$y_m(\mathbf{x}) = h(\mathbf{x}) + \epsilon_m(\mathbf{x}).$$

后面那一项是模型的 error

ZealotMaster(850458544) 19:24:34

能使 error 趋近于 0 嘛？

网神(66707180) 19:25:13

模型越好越趋近于 0，但很难等于 0，这里 committes 方法就比单个模型更趋近于 0

ZealotMaster(850458544) 19:25:28

求证明

网神(66707180) 19:25:39

正在证明，平均平方和错误如下：

$$\mathbb{E}_{\mathbf{x}}\left[\{y_m(\mathbf{x}) - h(\mathbf{x})\}^2\right] = \mathbb{E}_{\mathbf{x}}\left[\epsilon_m(\mathbf{x})^2\right]$$

也就是单个模型的期望 error 是：

$$\mathbb{E}_{\mathbf{x}}\left[\epsilon_m(\mathbf{x})^2\right]$$

如果用 M 个模型分别做预测，其平均错误是:

$$E_{\mathrm{AV}} = \frac{1}{M}\sum_{m-1}^{M} \mathbb{E}_{\mathbf{x}}\left[\epsilon_m(\mathbf{x})^2\right]$$

而如果用 committes 的结果来做预测，其期望错误是：

$$\begin{aligned} E_{\mathrm{COM}} &= \mathbb{E}_{\mathbf{x}}\left[\left\{\frac{1}{M}\sum_{m=1}^{M} y_m(\mathbf{x}) - h(\mathbf{x})\right\}^2\right] \\ &= \mathbb{E}_{\mathbf{x}}\left[\left\{\frac{1}{M}\sum_{m=1}^{M} \epsilon_m(\mathbf{x})\right\}^2\right] \end{aligned}$$

这个 $\frac{1}{M}$ 跑到了平方的里面，如果假设不同模型的 error 都是 0 均值，并且互不相关，也就是：

$$\begin{aligned} \mathbb{E}_{\mathbf{x}}\left[\epsilon_m(\mathbf{x})\right] &= 0 \\ \mathbb{E}_{\mathbf{x}}\left[\epsilon_m(\mathbf{x})\epsilon_l(\mathbf{x})\right] &= 0, \qquad m \neq l \end{aligned}$$

就可以得到：

$$E_{\mathrm{COM}} = \frac{1}{M}E_{\mathrm{AV}}.$$

在不同模型 error 互不相关的假设的下，committes 错误是单个模型 error 的 1/M，但实际上，不同模型

的 error 通常是相关的 ,因此 error 不会减少这么多 ,但肯定是小于单个模型的 error ,接下来讲 boosting，可以不考虑那个假设，取得实质的提高.boosting 应该是有不同的变种，其中最出名的就是 AdaBoost, adaptive boosting. 它可以将多个弱分类器结合，取得很好的预测效果，所谓弱分类器就是，只要比随即预测强一点，大概就是只要准确率超 50%就行吧，这是我的理解。

boosting 的思想是依次预测每个分类器，每个分类器训练时，对每个数据点加一个权重。训练完一个分类器模型，该模型分错的，再下一个模型训练时，增大权重；分对的，减少权重，具体的算法如下，我把整个算法帖出来，再逐步解释：

AdaBoost

1. Initialize the data weighting coefficients $\{w_n\}$ by setting $w_n^{(1)} = 1/N$ for $n = 1, \ldots, N$.
2. For $m = 1, \ldots, M$:
   (a) Fit a classifier $y_m(\mathbf{x})$ to the training data by minimizing the weighted error function

$$J_m = \sum_{n=1}^{N} w_n^{(m)} I(y_m(\mathbf{x}_n) \neq t_n) \tag{14.15}$$

   where $I(y_m(\mathbf{x}_n) \neq t_n)$ is the indicator function and equals 1 when $y_m(\mathbf{x}_n) \neq t_n$ and 0 otherwise.

   (b) Evaluate the quantities

$$\epsilon_m = \frac{\sum_{n=1}^{N} w_n^{(m)} I(y_m(\mathbf{x}_n) \neq t_n)}{\sum_{n=1}^{N} w_n^{(m)}} \tag{14.16}$$

   and then use these to evaluate

$$\alpha_m = \ln\left\{\frac{1-\epsilon_m}{\epsilon_m}\right\}. \tag{14.17}$$

   (c) Update the data weighting coefficients

$$w_n^{(m+1)} = w_n^{(m)} \exp\{\alpha_m I(y_m(\mathbf{x}_n) \neq t_n)\} \tag{14.18}$$

3. Make predictions using the final model, which is given by

$$Y_M(\mathbf{x}) = \text{sign}\left(\sum_{m=1}^{M} \alpha_m y_m(\mathbf{x})\right). \tag{14.19}$$

第一步，初始化每个数据点的权重为 1/N.接下来依次训练 M 个分类器，每个分类器训练时，最小化加权的错误函数(14.15)，错误函数看上面贴的算法，从这个错误函数可以看出，权重相同时，尽量让更多的 x 分类正确，权重不同时，优先让权重大的 x 分类正确，训练完一个模型后，式(14.16)计算 ϵ_m ，既分类错误的样本的加权比例. 然后式(14.17)计算:

$$\alpha_m = \ln\left\{\frac{1-\epsilon_m}{\epsilon_m}\right\}$$

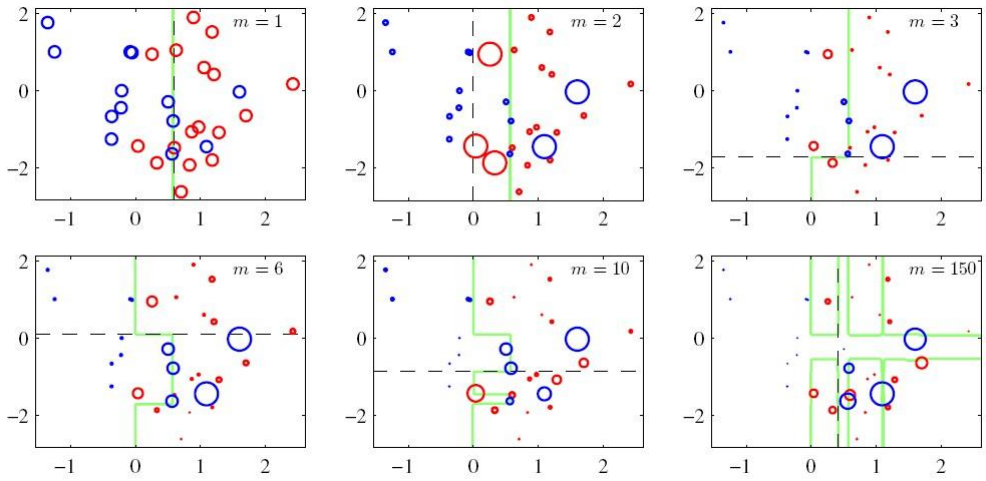
只要分类器准确率大于 50%， ϵ_m 就小于 0.5, α_m 就大于 0。而且 ϵ_m 越小(既对应的分类器准确率越高)， α_m 就越大，然后用 α_m 更新每个数据点的权重，即式(14.18)：

$$w_n^{(m+1)} = w_n^{(m)} \exp\{\alpha_m I(y_m(\mathbf{x}_n) \neq t_n)\} \qquad (14.18)$$

可以看出，对于分类错误的数据点，α_m 大于 0，所以 exp(a)就大于 1，所以权重变大。但是从这个式子看不出，对于分类正确的样本，权重变小。这个式子表明，分类正确的样本，其权重不变 ，因为 exp(0)=1. 这是个疑问。如此循环，训练完所有模型，最后用式(14.19)做预测:

$$Y_M(\mathbf{x}) = \text{sign}\left(\sum_{m=1}^{M} \alpha_m y_m(\mathbf{x})\right)$$

从上面过程可以看出，如果训练集合中某个样本在逐步训练每个模型时，一直被分错，他的权重就会一直变大，最后对应的 α_m 也越来越大，下面看一个图例：



图中有蓝红两类样本 ，分类器是单个的平行于轴线的阈值 ，第一个分类器(m=1)把大部分点分对了，但有 2 个蓝点，3 个红点不对，m=2 时，这几个错的就变大了，圈越大，对应其权重越大 ，后面的分类器似乎是专门为了这个几个错分点而在努力工作，经过 150 个分类器，右下角那个图的分割线已经很乱了，看不出到底对不对 ，应该是都已经分对了吧。

网神(66707180) 19:59:59

@ZealotMaster 不知道是否明白点了，大家有啥问题？

ZealotMaster(850458544) 20:00:14

嗯，清晰一些了，这个也涉及 over fitting 嘛？感觉 m=150 好乱……

苦瓜炒鸡蛋(852383636) 20:02:23

是过拟合

网神(66707180) 20:02:25

不同的分割线，也就是不同的模型，是主要针对不同的点的，针对哪些点，由模型组合时的系数 α_m 来影响。

苦瓜炒鸡蛋(852383636) 20:04:50

这是韩家炜 那个数据挖掘书的那一段：

"How does boosting compare with bagging?" Because of the way boosting focuses on the misclassified tuples, it risks overfitting the resulting composite model to such data. Therefore, sometimes the resulting "boosted" model may be less accurate than a single model derived from the same data. Bagging is less susceptible to model overfitting. While both can significantly improve accuracy in comparison to a single model, boosting tends to achieve greater accuracy.

网神(66707180) 20:04:56

嗯，这章后面也讲到了 boosting 对某些错分的样本反应太大，一些异常点会对模型造成很大的影响，接下来讲 boosting 的错误函数，我们仔细看下对 boosting 错误函数的分析，boosting 最初用统计学习理论来分析器泛化错误的边界 bound，但后来发现这个 bound 太松，没有意义。实际性能比这个理论边界好得多，后来用指数错误函数来表示。指数错误函数如下：

$$E = \sum_{n=1}^{N} \exp\{-t_n f_m(\mathbf{x}_n)\}$$

其中 N 是 N 个样本点，$f_m(\mathbf{x})$是多个线性分类器的线性组合:

$$f_m(\mathbf{x}) = \frac{1}{2}\sum_{l=1}^{m} \alpha_l y_l(\mathbf{x})$$

$t_n \in \{-1, 1\}$是分类的目标值。我们的目标是训练系数α_l和分类器$y_l(\mathbf{x})$的参数，使 E 最小。

最小化 E 的方法，是先只针对一个分类器进行最小化，而固定其他分类器的参数，例如我们固定 $y_1(\mathbf{x}), \ldots, y_{m-1}(\mathbf{x})$和其系数 $\alpha_1, \ldots, \alpha_{m-1}$，只针对$\alpha_m$ and $y_m(\mathbf{x})$做优化，这样错误函数 E 可以改写为：

$$\begin{aligned} E &= \sum_{n=1}^{N} \exp\left\{-t_n f_{m-1}(\mathbf{x}_n) - \frac{1}{2} t_n \alpha_m y_m(\mathbf{x}_n)\right\} \\ &= \sum_{n=1}^{N} w_n^{(m)} \exp\left\{-\frac{1}{2} t_n \alpha_m y_m(\mathbf{x}_n)\right\} \end{aligned}$$

也就是把固定的分类器的部分都当做一个常量：$w_n^{(m)} = \exp\{-t_n f_{m-1}(\mathbf{x}_n)\}$，只保留$y_m(\mathbf{x})$相关的部分。我们用$\mathcal{T}_m$表示$y_m(\mathbf{x})$分对的数据集，$\mathcal{M}_m$表示分错的数据集，E 又可以写成如下形式：

$$\begin{aligned} E &= e^{-\alpha_m/2} \sum_{n \in \mathcal{T}_m} w_n^{(m)} + e^{\alpha_m/2} \sum_{n \in \mathcal{M}_m} w_n^{(m)} \\ &= (e^{\alpha_m/2} - e^{-\alpha_m/2}) \sum_{n=1}^{N} w_n^{(m)} I(y_m(\mathbf{x}_n) \neq t_n) + e^{-\alpha_m/2} \sum_{n=1}^{N} w_n^{(m)}. \end{aligned}$$

上式中，因为将数据分成两部分，也就确定了t_n个$y_m(\mathbf{x}_n)$是否相等，所以消去了这两个变量

看起来清爽点了：

$$= (e^{\alpha_m/2} - e^{-\alpha_m/2}) \sum_{n=1}^{N} w_n^{(m)} I(y_m(\mathbf{x}_n) \neq t_n) + e^{-\alpha_m/2} \sum_{n=1}^{N} w_n^{(m)}. \quad (14.23)$$

这里面后一项是常量，前一项就跟前面 boosting 的算法里所用的错误函数：

$J_m = \sum_{n=1}^{N} w_n^{(m)} I(y_m(\mathbf{x}_n) \neq t_n)$ 形式一样了。

上面是将：

$$= \quad (e^{\alpha_m/2} - e^{-\alpha_m/2}) \sum_{n=1}^{N} w_n^{(m)} I(y_m(\mathbf{x}_n) \neq t_n) + e^{-\alpha_m/2} \sum_{n=1}^{N} w_n^{(m)}.$$

(14.23

对$y_m(\mathbf{x}_n)$ 做最小化得出的结论,即指数错误函数就是 boosting 里求单个模型时所用的错误函数.类似，

也可以得到指数错误函数里的α_m就是 boosting 里的ϵ_m ，确定了α_m and $y_m(\mathbf{x})$,根据

$$\begin{aligned} E &= \sum_{n=1}^{N} \exp\left\{-t_n f_{m-1}(\mathbf{x}_n) - \frac{1}{2} t_n \alpha_m y_m(\mathbf{x}_n)\right\} \\ &= \sum_{n=1}^{N} w_n^{(m)} \exp\left\{-\frac{1}{2} t_n \alpha_m y_m(\mathbf{x}_n)\right\} \end{aligned} \qquad (14.22$$

可以得到，更新 w 的方法：

$$w_n^{(m+1)} = w_n^{(m)} \exp\left\{-\frac{1}{2} t_n \alpha_m y_m(\mathbf{x}_n)\right\}.$$

又因为 $t_n y_m(\mathbf{x}_n) = 1 - 2I(y_m(\mathbf{x}_n) \neq t_n)$ 有：

$w_n^{(m+1)} = w_n^{(m)} \exp(-\alpha_m/2) \exp\{\alpha_m I(y_m(\mathbf{x}_n) \neq t_n)\}$ 这又跟 boosting 里更新数据点权重的方法以一致。

总之，就是想说明，用指数错误函数可以描述 boosting 的 error 分析,接下来看看指数错误函数的性质，再贴一下指数错误函数的形式：

$$E = \sum_{n=1}^{N} \exp\{-t_n f_m(\mathbf{x}_n)\}$$

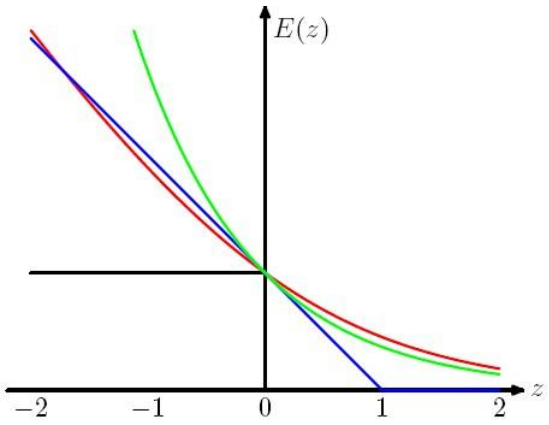
$$f_m(\mathbf{x}) = \frac{1}{2} \sum_{l=1}^{m} \alpha_l y_l(\mathbf{x})$$

其期望 error 是：

$$\mathbb{E}_{\mathbf{x},t}\left[\exp\{-t y(\mathbf{x})\}\right] = \sum_t \int \exp\{-t y(\mathbf{x})\} p(t|\mathbf{x}) p(\mathbf{x})\, \mathrm{d}\mathbf{x}.$$

然后最所有的 y(x)做 variational minimization,得到：$y(\mathbf{x}) = \frac{1}{2} \ln\left\{\frac{p(t=1|\mathbf{x})}{p(t=-1|\mathbf{x})}\right\}$

这是 half the log-odds ，看下指数错误函数的图形：



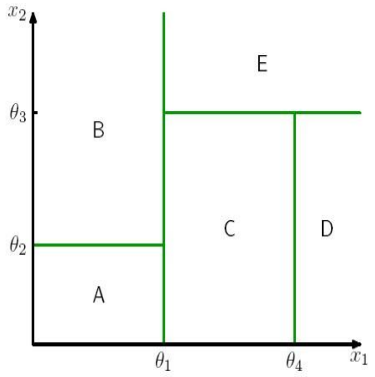
绿色的线是指数错误函数 $z = ty(\mathbf{x})$,可以看到，对于分错的情况，既 z<0 时，绿色的线呈指数趋势上升，所以异常点会对训练结果的影响很大。图中红色的线是 corss-entropy 错误，是逻辑分类中用的错误函数 ,对分错的情况 ,随错误变大 ,其函数值基本是线性增加的 ,蓝色线是 svm 用的错误函数 ,叫 hinge 函数。

==========

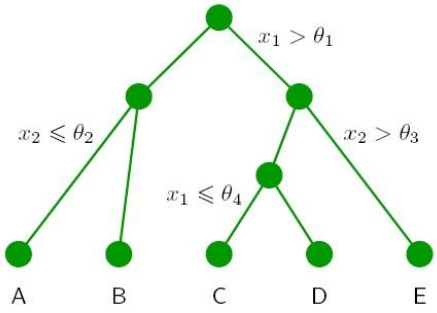
大家有没有要讨论的？公式比较多，需要推导一下才能理解。接下来讲决策树和条件混合模型。决策树概念比较简单，容易想象是什么样子的，可以认为决策树是多个模型，每个模型负责 x 的一个区间的预测，通过树形结构来寻找某个 x 属于哪个区间，从而得到预测值。

决策树有多个算法比较出名，ID3, C4.5, CART，书上以 CART 为例讲的。

CART 叫 classification and regression trees 先看一个图示：



这个二维输入空间，被划分成 5 个区间，每个区间的类别分别是 A-E，它的决策树如下，很简单明了：



决策树在一些领域应用比较多，最主要的原因是，他有很好的可解释性。那如何训练得到一个合适的决策树呢？也就是如何决定需要哪些节点，节点上选择什么变量作为判断依据，以及判断的阈值是多大。

首先错误函数是残差的平方和，而树的结构则用贪心策略来演化。

开始只有一个节点，也就是根节点，对应整个输入空间，然后将空间划分为 2，在多个划分选择之中，选择使残差最小的那个划分，书上提到这个区间划分以及划分点阈值的选择，是用穷举的方法。然后对不同的叶子节点再分别划分子区间。这样树不停长大，何时停止增加节点呢？简单的方法是当残差小于某个值的时候。但经验发现经常再往多做一些划分后，残差又可以大幅降低

所以通常的做法是，先构造一个足够大的树，使叶子节点多到某个数量，然后再剪枝，合并的原则是使叶子尽量减少，同时残差又不要增大。

也就是最小化这个值：$$C(T) = \sum_{\tau=1}^{|T|} Q_\tau(T) + \lambda|T|$$ 其中：$$Q_\tau(T) = \sum_{\mathbf{x}_n \in \mathcal{R}_\tau} \{t_n - y_\tau\}^2 .$$

y_τ 是某个叶子负责的那个区间里的所有数据点的预测值的平均值：$$y_\tau = \frac{1}{N_\tau} \sum_{\mathbf{x}_n \in \mathcal{R}_\tau} t_n$$

t_n 是某个数据点的预测值，$|T|$ 是叶子的总数，λ 控制残差和叶子数的 trade-off，这是剪枝的依据。

以上是针对回归说的，对于分类，剪枝依据仍然是：

$$C(T) = \sum_{\tau=1}^{|T|} Q_\tau(T) + \lambda|T|$$

只是 Q(T)不再是残差的平方和，而是 cross-entropy 或 Gini index

交叉熵的计算是:

$$Q_\tau(T) = \sum_{k=1}^{K} p_{\tau k} \ln p_{\tau k}$$

$p_{\tau k}$ 是第τ个叶子，也就是第τ个区间里的数据点，被赋予类别 k 的比例。

Gini index 计算是：

$$Q_\tau(T) = \sum_{k=1}^{K} p_{\tau k} \left(1 - p_{\tau k}\right).$$

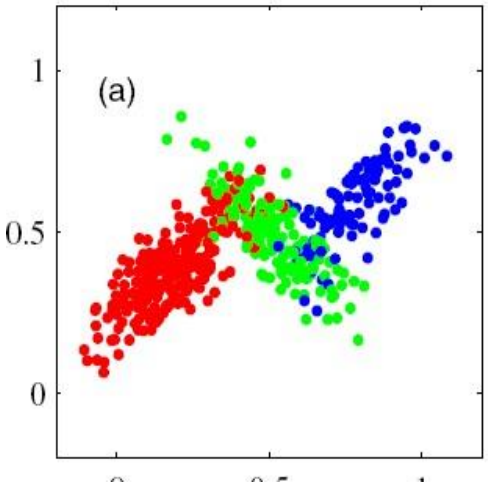
决策树的优点是直观，可解释性好。缺点是每个节点对输入空间的划分都是根据某个维度来划分的，在坐标空间里看，就是平行于某个轴来划分的，其次，决策树的划分是硬性的，在回归问题里，硬性划分更明显。

==========

决策树就讲这么多 ,接下来是 conditional mixture models 条件混合模型。条件混合模型 ,我的理解是，将不同的模型依概率来结合，这部分讲了线性回归的条件混合，逻辑回归的条件混合，和更一般性的混合方法 mixture of experts，首先看线性回归的混合。

第 9 章讲过高斯混合模型，其模型是这样的：$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k).$$

这是用多个高斯密度分布来拟合输入空间，就像这种 x 的分布：



线性回归混合的实现，可以把这个高斯混合分布扩展成条件高斯分布，模型如下：

$$p(t|\boldsymbol{\theta}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(t|\mathbf{w}_k^{\mathrm{T}}\boldsymbol{\phi}, \beta^{-1})$$

这里面，有 K 个线性回归$\mathbf{w}_k^{\mathrm{T}}\boldsymbol{\phi}$，其权重参数是$\mathbf{W}k$，将多个线性回归的预测值确定的概率联合起来，得到最终预测值的概率分布。模型中的参数$\boldsymbol{\theta}$包括$\mathbf{W} = \{\mathbf{w}_k\}\boldsymbol{\pi} = \{\pi_k\}, \text{ and } \beta$三部分，接下来看如何训练得到这些参数，总体思路是用第 9 章介绍的 EM 算法，引入一个隐藏变量$\mathbf{Z} = \{\mathbf{z}_n\}$，每个训练样本点对应一个$\mathbf{Z}_n$，$\mathbf{Z}_n$是一个 K 维的二元向量，$z_{nk} \in \{0, 1\}$,如果第 k 个模型负责生成第 n 个数据点，则$z_{nk}$等于 1，否则等于 0，这样，我们可以写出 log 似然函数：

$$\ln p(\mathbf{t}, \mathbf{Z}|\boldsymbol{\theta}) = \sum_{n=1}^{N}\sum_{k=1}^{K} z_{nk} \ln \left\{\pi_k \mathcal{N}(t_n|\mathbf{w}_k^{\mathrm{T}}\boldsymbol{\phi}_n, \beta^{-1})\right\}$$

然后用 EM 算法结合最大似然估计来求各个参数，EM 算法首先选择所有参数的初始值，假设是$\boldsymbol{\theta}^{\mathrm{old}}$

在 E 步根据这些参数值，可以得到模型 k 相对于数据点 n 的后验概率：

$$\gamma_{nk} = \mathbb{E}[z_{nk}] = p(k|\boldsymbol{\phi}_n, \boldsymbol{\theta}^{\mathrm{old}}) = \frac{\pi_k \mathcal{N}(t_n|\mathbf{w}_k^{\mathrm{T}}\boldsymbol{\phi}_n, \beta^{-1})}{\sum_j \pi_j \mathcal{N}(t_n|\mathbf{w}_j^{\mathrm{T}}\boldsymbol{\phi}_n, \beta^{-1})}.$$

书上提高一个词 ,这个后验概率又叫做 responsibilities ,大概是这个数据点 n 由模型 k 负责生成的概率吧，有了这个 responsibilities，就可以计算似然函数的期望值了，如下：

$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{\mathrm{old}}) = \mathbb{E}_{\mathbf{Z}}\left[\ln p(\mathbf{t}, \mathbf{Z}|\boldsymbol{\theta})\right] = \sum_{n=1}^{N}\sum_{k=1}^{K} \gamma_{nk} \left\{\ln \pi_k + \ln \mathcal{N}(t_n|\mathbf{w}_k^{\mathrm{T}}\boldsymbol{\phi}_n, \beta^{-1})\right\}$$

在 EM 的 M 步，我们令γ_{nk}为固定值，最大化这个期望值$Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{\mathrm{old}})$,从而求得新的参数$\boldsymbol{\theta}$

首先看 π_k ， π_k 是各个模型的混合权重系数，满足 $\sum_k \pi_k = 1$ ，用拉格朗日乘子法，可以求得:

$$\pi_k = \frac{1}{N}\sum_{n=1}^{N}\gamma_{nk}.$$

接下来求线性回归的参数 $\mathbf{W}k$ ，将似然函数期望值的式子里的高斯分布展开，可以得到如下式子：

$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{old}}) = \sum_{n=1}^{N}\gamma_{nk}\left\{-\frac{\beta}{2}\left(t_n - \mathbf{w}_k^{\mathrm{T}}\boldsymbol{\phi}_n\right)^2\right\} + \text{const}$$

要求第 k 个模型的 $\mathbf{W}k$ ，其他模型的 W 都对其没有影响，可以统统归做后面的 const，这是因为 log 似然函数，每个模型之间是相加的关系，一求导数，其他模型的项就都消去了。上面的式子是一个加权的最小平方损失函数，每个数据点对应一个权重 $\beta\gamma_{nk}$，这个权重可以看做是数据点的有效精度，将这个式子对 $\mathbf{W}k$ 求导，可以得到：

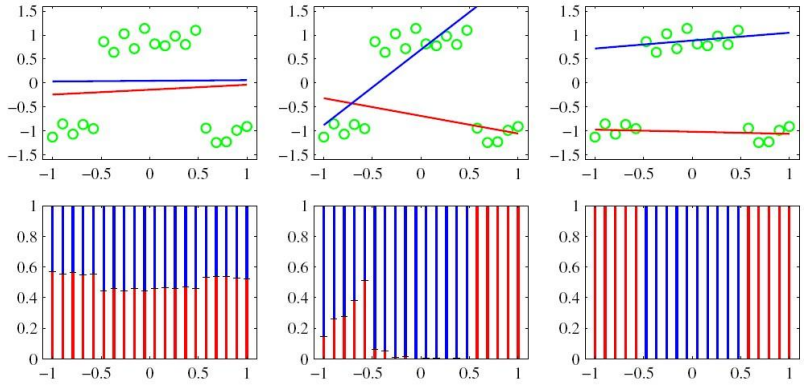
$$0 = \sum_{n=1}^{N}\gamma_{nk}\left(t_n - \mathbf{w}_k^{\mathrm{T}}\boldsymbol{\phi}_n\right)\boldsymbol{\phi}_n$$

最后求得 $\mathbf{w}_k = \left(\boldsymbol{\Phi}^{\mathrm{T}}\mathbf{R}_k\boldsymbol{\Phi}\right)^{-1}\boldsymbol{\Phi}^{\mathrm{T}}\mathbf{R}_k\mathbf{t}.$

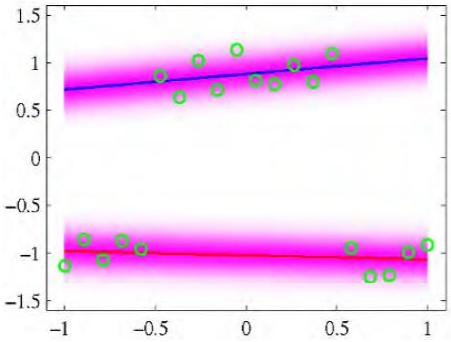
其中 $\mathbf{R}_k = \text{diag}(\gamma_{nk})$ ，同样，对 β 求导，得到： $\frac{1}{\beta} = \frac{1}{N}\sum_{n=1}^{N}\sum_{k=1}^{K}\gamma_{nk}\left(t_n - \mathbf{w}_k^{\mathrm{T}}\boldsymbol{\phi}_n\right)^2$

这样的到了所有的参数的新值，再重复 E 步和 M 步，迭代求得满意的最终的参数值。

接下来看一个线性回归混合模型 EM 求参数的图示，两条直线对应两个线性回归模型用 EM 方法迭代求参数，两条直线最终拟合两部分数据点，中间和右边分别是经过了 30 轮和 50 轮迭代，下面那一排，表示每一轮里，每个数据点对应的 responsibilities，也就是类 k 对于数据点 n 的后验概率：



最终求得的混合模型图示：



接下来讲逻辑回归混合模型，逻辑回归模型本身就定义了一个目标值的概率，所以可以直接用逻辑回归本身作为概率混合模型的组件，其模型如下：

$$p(t|\phi,\boldsymbol{\theta}) = \sum_{k=1}^{K} \pi_k y_k^t \left[1-y_k\right]^{1-t}$$

其中$y_k = \sigma\left(\mathbf{w}_k^{\mathrm{T}}\phi\right)$，这里面的参数$\boldsymbol{\theta}$包括$\{\pi_k\}$ and $\{\mathbf{w}_k\}$两部分。求参数的方法也是用EM，这里就不细讲了，要提一下的是在 M 步，似然函数不是一个 closed-form 的形式，不能直接求导来的出参数，需要用 IRLS(iterative reweighted least squares)等迭代方法来求，下图是一个逻辑回归混合模型的训练的图示，左图是两个类别的数据，蓝色和红色表示实际的分布，中间图是用一个逻辑回归来拟合的模型，右图是用两个逻辑回归混合模型拟合的图形：

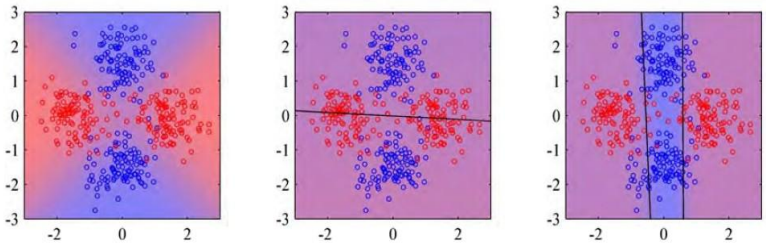


Figure 14.10 Illustration of a mixture of logistic regression models. The left plot shows data points drawn from two classes denoted red and blue, in which the background colour (which varies from pure red to pure blue) denotes the true probability of the class label. The centre plot shows the result of fitting a single logistic regression model using maximum likelihood, in which the background colour denotes the corresponding probability of the class label. Because the colour is a near-uniform purple, we see that the model assigns a probability of around 0.5 to each of the classes over most of input space. The right plot shows the result of fitting a mixture of two logistic regression models, which now gives much higher probability to the correct labels for many of the points in the blue class.

接下来讲 mixtures of experts，mixture of experts 是更一般化的混合模型，前面讲的两个混合模型，其混合系数是固定值，我们可以让混合系数是输入 x 的函数即：

$$p(\mathbf{t}|\mathbf{x}) = \sum_{k=1}^{K} \pi_k(\mathbf{x}) p_k(\mathbf{t}|\mathbf{x}).$$

系数$\pi_k(\mathbf{x})$叫做 gating 函数，组成模型 $p_k(\mathbf{t}|\mathbf{x})$叫做 experts，,每个 experts 可以对输入空间的不同区域建模，对不同区域的输入进行预测，而 gating 函数则决定一个 experts 该负责哪个区域。

gating 函数满足 $0 \leqslant \pi_k(\mathbf{x}) \leqslant 1$ and $\sum_k \pi_k(\mathbf{x}) = 1$，因此可以用 softmax 函数来作为 gating 函数，如果 expert 函数也是线性函数，则整个模型就可以用 EM 算法来确定。在 M 步，可能需要用 IRLS，来迭代求解，这个模型仍然有局限性，更一般化的模型是 hierarchical mixture of experts 也就是混合模型的每个组件又可以是一个混合模型。好了就讲这么多吧，书上第 14 章的内容都讲完了。