第 12 章 随机森林

第 12-13 章介绍**集成学习**(ensemble learning)的方法，也称为**组合学习**。

12.1 集成学习

集成学习的基本问题：如果有一些预测效果一般的**弱学习器**(weak learner)，能否以某种方式将它们组合起来，构成一个预测效果优良的**强学习器**(strong learner)？

集成学习的基本思路类似于“三个臭皮匠，顶个诸葛亮”（wisdom of crowds）。

用于集成学习的弱学习器，也称为**基学习器**(base learner)。

决策树则是最常见的基学习器。

给定数据与算法，则监督学习的估计结果是基本确定的(可能由于随机分组的不同而略有差异)，这可表示为

$$\text{数据} + \text{算法} = \text{结果} \qquad (12.1)$$

如此看来，怎么才能得到不同结果呢？

一种解决方法是，给定数据，但使用不同的算法，然后将所得的不同结果组合在一起；比如加权平均，而权重以交叉验证来确定。

以回归问题为例，假设共有三种不同的算法(比如线性回归、KNN 与决策树)，其预测结果分别为 $\hat{f}_1(\mathbf{x})$，$\hat{f}_2(\mathbf{x})$与 $\hat{f}_3(\mathbf{x})$；则集成算法的最终预测结果为

$$\hat{f}(\mathbf{x})=\alpha_1\hat{f}_1(\mathbf{x})+\alpha_2\hat{f}_2(\mathbf{x})+(1-\alpha_1-\alpha_2)\hat{f}_3(\mathbf{x}) \qquad (12.2)$$

其中，$\alpha_1\geq 0$，$\alpha_2\geq 0$，且$\alpha_1+\alpha_2\leq 1$。最优调节参数α_1与α_2可通过交叉验证来确定。

由于算法 $\hat{f}_1(\mathbf{x})$，$\hat{f}_2(\mathbf{x})$与 $\hat{f}_3(\mathbf{x})$皆为集成算法 $\hat{f}(\mathbf{x})$的特例，故 $\hat{f}(\mathbf{x})$的预测效果肯定优于(至少不差于)这三种单独的算法。

类似地，对于分类问题，则可将不同算法的预测结果，进行加权求和，按多数票规则投票；而投票权重可通过交叉验证确定。

集成学习的另一方式为，给定算法(比如以决策树的 CART 算法为基学习器)，然后“搅动数据”，得到不同的决策树模型，再组合在一起，即所谓“Perturb + Combine”的模式。

本章介绍的装袋法与随机森林即属于这种方式。

12.2 装袋法

Breiman(1996)提出**装袋法**(bootstrap aggregating，简记 bagging)。

装袋法使用“自助法”(bootstrap)来搅动数据。所谓自助法，是一种有放回的再抽样方法(resampling with replacement)。装袋法的具体步骤如下。

首先，对训练数据进行有放回的再抽样，得到B个自助样本(bootstrap samples)，比如$B=500$。其中，第b个自助样本可写为

$$\left\{\mathbf{x}_i^{*b}, y_i^{*b}\right\}_{i=1}^{n}, \quad b=1,\cdots,B \qquad (12.3)$$

其中，上标(superscript)的星号“*”表示这是自助样本。每个自助样本的样本容量均为n(保持样本容量不变)。

其次，根据自助样本$\left\{\mathbf{x}_i^{*b}, y_i^{*b}\right\}_{b=1}^{B}$估计$B$棵不同的决策树，不进行修枝。记其预测结果为

$$\left\{\hat{f}^{*b}(\mathbf{x})\right\}, \quad b=1,\cdots,B \qquad (12.4)$$

最后，对于回归树，将B棵决策树的预测结果进行平均：

$$\hat{f}_{bag}(\mathbf{x})=\frac{1}{B}\sum_{b=1}^{B}\hat{f}^{*b}(\mathbf{x}) \qquad (12.5)$$

对于分类树，则将B棵决策树的预测结果进行多数投票(majority vote)，得票最多的类别胜出：

$$\hat{f}_{bag}(\mathbf{x}) = \underset{y\in\{1,\cdots,K\}}{\operatorname{argmax}} \sum_{b=1}^{B} I\left(y = \hat{f}^{*b}(\mathbf{x})\right) \qquad (12.6)$$

其中，假设$y \in \{1,\cdots,K\}$，共分为K类。$I(\cdot)$为示性函数。

通过多数票，将许多弱分类器进行组合，即所谓“combining many weaker classifiers to produce a powerful committee”。

由于装袋法把很多自助样本(bootstrap samples)的估计结果汇总(aggregating)，故名“bootstrap aggregating”。

12.3 装袋法的原理

由于 bagging 将很多决策树进行平均，故可降低估计量的方差(variance)，从而提高模型的预测准确率。

例如，假设随机变量$\{z_1,\cdots,z_n\}$为独立同分布(iid)，而方差为σ^2，则样本均值$\overline{z}=\frac{1}{n}\sum_{i=1}^{n}z_i$的方差可缩小$n$倍：

$$\mathrm{Var}(\bar{z}) = \mathrm{Var}\left(\frac{1}{n}\sum_{i=1}^{n} z_i\right) = \frac{1}{n^2}\sum_{i=1}^{n}\mathrm{Var}(z_i) = \frac{\sigma^2}{n} \qquad (12.7)$$

其中，由于$\{z_1, \cdots, z_n\}$相互独立，故上式中的协方差均为 0，而所有$\mathrm{Var}(z_i) = \sigma^2$(同分布)。

另一方面，由于装袋法并不修枝，而让决策树尽情生长，故可降低每棵决策树的偏差(bias)；然后，通过 bagging 来控制方差(variance)。

在一定条件下，装袋法可以同时降低偏差与方差，故可减小均方误差(MSE)或总误差。

Breiman(1996)使用若干标准的机器学习数据集，发现与单棵决策树相比，bagging 可显著降低测试误差。

应该在什么情况下使用装袋法？

Bagging 特别适用于方差较大的不稳定估计量(unstable estimators)，比如决策树、线性回归的变量子集选择(subset selection)等。

对于不稳定估计量，当样本数据轻微扰动时(比如换一个样本，或使用子样本)，则可能引起估计结果较大的变化。

比如，当数据轻微变化时，所得决策树结构可能发生较大改变。

又比如，当用线性回归进行变量子集选择时，所选变量也对样本数据比较敏感，并不稳健。

另一方面，bagging 对于方差较小的稳定估计量(stable estimators)，比如 KNN，则作用不大。对于 KNN 而言，即使搅动数据，一般也不会对附近 K 近邻的平均结果产生明显影响。

由于单棵决策树为分段常值函数(piecewise constant function)，故并不连续。

将许多决策树进行平均，则可得到更为光滑的回归函数或决策边界，从而提高预测性能。

以摩托车撞击实验数据集 mcycle 为例(参见第 10 章)。

首先，导入本章所需模块：

```
In [1]: import numpy as np
   ...: import pandas as pd
   ...: import matplotlib.pyplot as plt
   ...: import seaborn as sns
   ...: from sklearn.model_selection import
             train_test_split
   ...: from sklearn.model_selection import KFold,
             StratifiedKFold
```

```
...: from sklearn.model_selection import
         GridSearchCV
...: from sklearn.metrics import mean_squared_error
...: from sklearn.linear_model import
         LinearRegression
...: from sklearn.linear_model import
         LogisticRegression
...: from sklearn.tree import DecisionTreeClassifier
...: from sklearn.tree import DecisionTreeRegressor
...: from sklearn.ensemble import BaggingClassifier
...: from sklearn.ensemble import BaggingRegressor
...: from sklearn.ensemble import
         RandomForestClassifier
```

```
...: from sklearn.ensemble import
         RandomForestRegressor
...: from sklearn.datasets import load_iris,
         load_boston
...: from sklearn.metrics import cohen_kappa_score
...: from sklearn.metrics import plot_roc_curve
...: from sklearn.inspection import
         plot_partial_dependence
...: from mlxtend.plotting import
         plot_decision_regions
```

载入数据文件 mcycle，并察看前 5 个观测值：

```
In [2]: mcycle = pd.read_csv('mcycle.csv')
   ...: mcycle.head()
Out[2]:
   times  accel
0    2.4    0.0
1    2.6   -1.3
2    3.2   -2.7
3    3.6    0.0
4    4.0   -2.7
```

取出数据矩阵 X 与响应向量 y：

```
In [3]: X = np.array(mcycle.times).reshape(-1, 1)
   ...: y = mcycle.accel
```

其中，使用 reshape(-1, 1)方法，将向量 np.array(mcycle.times)变为133×1的矩阵 X。

作为参照系照，先通过交叉验证选择最优决策树，并进行样本内预测：

```
In [4]: model = DecisionTreeRegressor(random_state=123)
   ...: path = model.cost_complexity_pruning_path(X, y)
   ...: param_grid = {'ccp_alpha': path.ccp_alphas}
   ...: kfold = KFold(n_splits=10, shuffle=True,
                      random_state=1)
   ...: model =
GridSearchCV(DecisionTreeRegressor(random_state=123),
                           param_grid, cv=kfold)
   ...: pred_tree = model.fit(X, y).predict(X)
In [5]: sns.scatterplot(x='times', y='accel',
                        data=mcycle, alpha=0.6)
   ...: plt.plot(X, pred_tree, 'b')
   ...: plt.title('Single Tree Estimation')
```

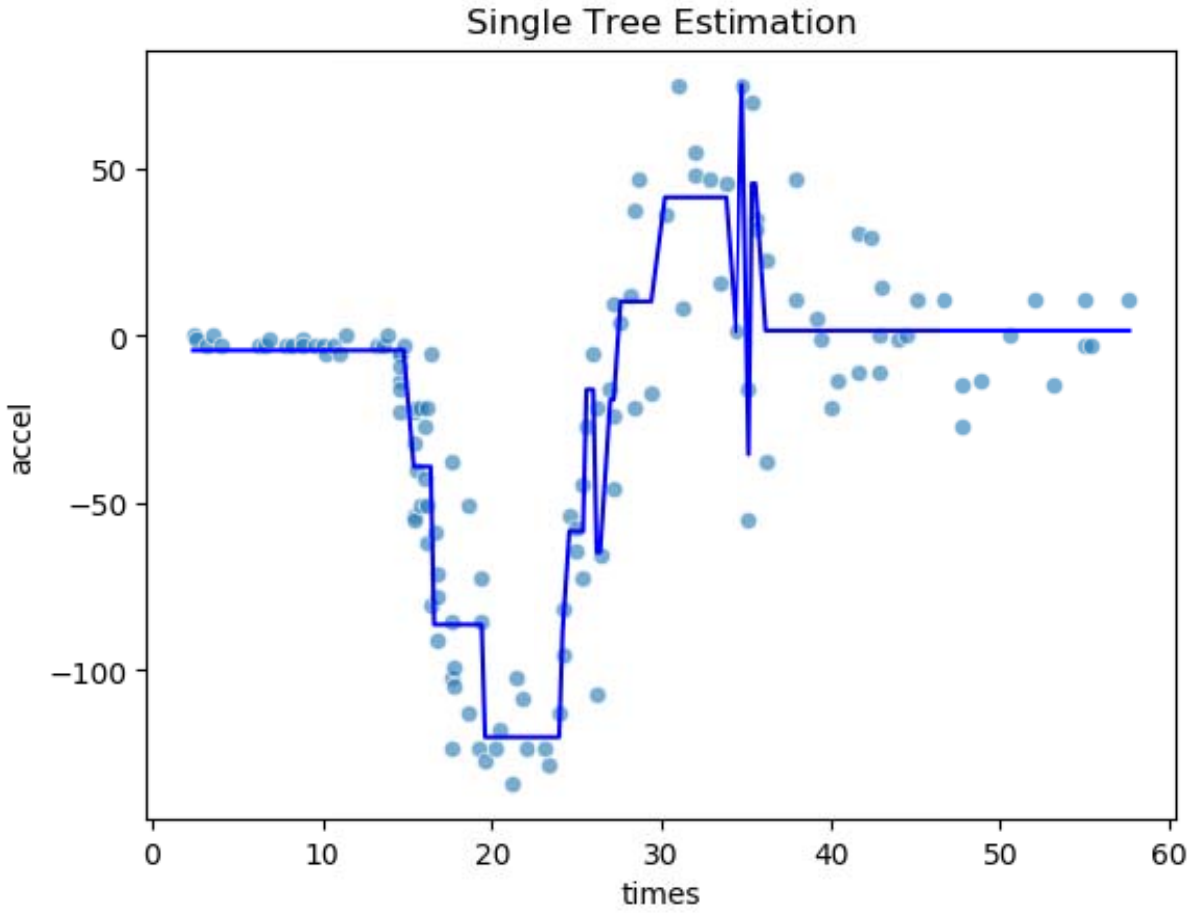

图 12.1 单棵决策树所估计的回归函数

从图 12.1 可见，单棵决策树所估计的回归函数为“阶梯函数”(step function)，并不连续。

虽然回归树抓住了数据变化的主要特征，但在细节上比较粗糙，存在“欠拟合”(underfit)。

下面使用 sklearn 的 `BaggingRegressor` 类作装袋法估计，并进行样本内预测：

```
In [6]: model = BaggingRegressor(base_estimator=
                DecisionTreeRegressor(random_state=123),
                      n_estimators=500, random_state=0)
   ...: pred_bag = model.fit(X, y).predict(X)
```

其中，参数“base_estimator=DecisionTreeRegressor (random_state=123)”指定决策树为基学习器；参数“n_estimators=500”表示共有 500 棵决策树。

```
In [7]: sns.scatterplot(x='times', y='accel',
                        data=mcycle, alpha=0.6)
   ...: plt.plot(X, pred_bag, 'b')
   ...: plt.title('Bagging Estimation')
```

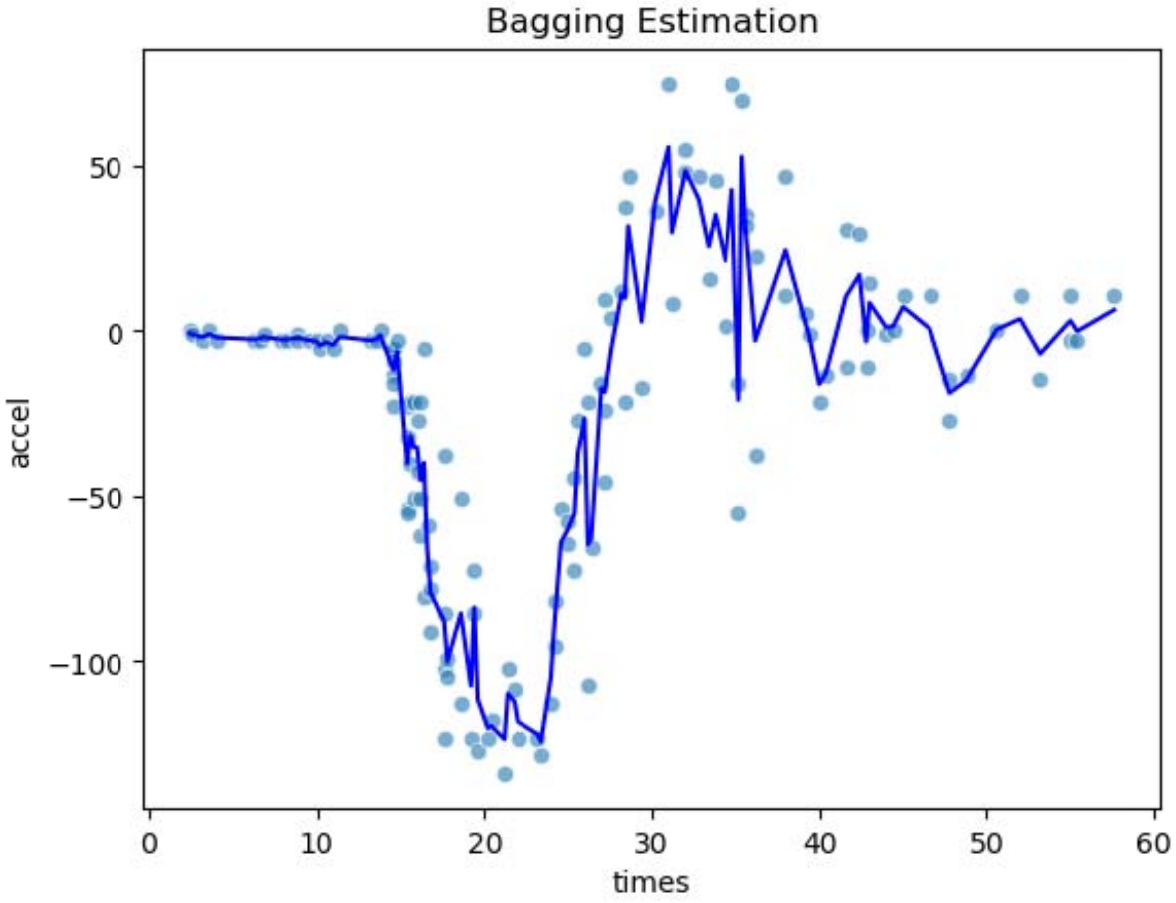

图 12.2 装袋法所估计的回归函数

从图 12.2 可见，虽然 bagging 所估计的回归函数依然不太光滑，但已经比单棵决策树更为贴近数据，拟合效果更好。

在此例中， bagging 所估函数之所以不太光滑，可能是由于样本容量太小，仅 133 个观测值。

12.4 袋外误差

Bagging 还提供了估计测试误差的简便方法，可以不用测试集(test set)或交叉验证(CV)。

由于进行有放回的再抽样，每棵树都有些观测值未使用，称为**袋外观测值**(Out-of-Bag observations，简记 OOB 观测值)，可作为测试集。

对于任意观测值$\mathbf{x}_i$，它未出现于大约$B/3$的决策树，故对于这些决策树而言是 OOB 观测值。

将这些(未用到$\mathbf{x}_i$)决策树的预测结果进行平均(或多数投票)，记为对第i个观测值的袋外预测值(OOB prediction for the ith observation)$\hat{y}_{i,OOB}$。由此，可得到对所有观测值的袋外预测值$\left\{\hat{y}_{i,OOB}\right\}_{i=1}^{n}$。

将袋外预测值$\hat{y}_{i,OOB}$与实际观测值y_i对比，即可得到**袋外误差**(out-of-bag error)。

对于回归问题，袋外误差为**袋外均方误差**(OOB MSE)：

$$\text{MSE}_{OOB} \equiv \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_{i,OOB} - y_i)^2 \qquad (12.8)$$

对于分类问题，袋外误差则为**袋外错误率**(OOB error rate)：

$$\text{Err}_{OOB} \equiv \frac{1}{n}\sum_{i=1}^{n} I(\hat{y}_{i,OOB} \neq y_i) \qquad (12.9)$$

如果自助样本的数目B足够大，则袋外误差与“留一交叉验证误差”(Leave-One-Out Cross-Validation error，简记 LOOCV)几乎等价。

对于样本容量很大或变量很多的大数据，尤其方便使用袋外误差(可节省运算时间)，作为对测试误差的估计。

对于回归问题，还可根据 OOB 均方误差计算准R^2(pseudo R^2)，即在响应变量y的样本方差中，有多大比例可由模型解释：

$$\text{Pseudo } R^2 \equiv \frac{\widehat{\text{Var}}(y) - \text{MSE}_{OOB}}{\widehat{\text{Var}}(y)} = 1 - \frac{\text{MSE}_{OOB}}{\widehat{\text{Var}}(y)} \quad (12.10)$$

其中，$\widehat{\text{Var}}(y)$为响应变量y的样本方差。

12.5 随机森林

如果$\{z_1,\cdots,z_n\}$为 iid，则其样本均值$\overline{z}$的方差可缩小n倍。

但如果$\{z_1,\cdots,z_n\}$之间相关，则样本均值$\overline{z}$的方差一般不会缩小n倍。

“三个臭皮匠，顶个诸葛亮”的成立有前提条件，即希望这些“臭皮匠”互不相关，才更有互补性。

在使用装袋法时，由于所用算法完全相同(比如都是 CART 算法)，而每棵决策树所用自助样本都来自同样的原始数据，故存在比较高的相关性。

如果想进一步提高 bagging 的预测性能，则必须降低这些决策树之间的相关性。

如何使决策树之间更不相关(decorrelate)？

Bagging 的决策树之间相关性强，也是因为使用相同的特征变量$\mathbf{x}_i$。

Breiman(2001b)大胆地提出**随机森林**(Random Forest)：

在 bagging 的基础上(依然使用自助样本)，在决策树的每个节点进行分裂时，仅随机选取部分变量(比如m个变量)作为候选的分裂变量。

比如，在一个节点随机选择m个变量作为候选的分裂变量(其余$(p-m)$个变量则不用)，而在下一节点再次随机选择(可能不相同的)m个变量作为候选的分裂变量，以此类推；然后对随机森林中的每棵决策树都如此操作。

对于回归树，一般建议随机选取$m=p/3$个变量(p为特征变量的个数)。对于决策树，则一般建议$m=\sqrt{p}$。

这种做法称为**随机特征选择**(random feature selection)，其目的是为了降低决策树之间的相关性。

随机特征选择也称为**列子抽样**(column subsampling)，因为在数据框的二维表结构中，每一列都对应于一个特征变量。

随机森林作预测的方式与装袋法相同，也是对所有决策树的预测结果进行平均(回归问题)，或多数投票(分类问题)。

在决策树的每个节点，随机森林仅使用少部分变量。这种做法似乎有些疯狂，因为大多数变量被暂时遗弃，未使用全部信息，无疑会增大偏差(bias)。

由于不同节点被强迫使用不同的变量进行分裂，故可降低不同决策树之间的相关性，从而减小方差(variance)。

在一次节点分裂中未选为候选变量的特征变量，以后依然有机会选上；或在其他决策树选上。

在偏差与方差的权衡中，随机森林以牺牲少量偏差为代价，可以换取方差的更大幅下降，从而降低均方误差(或总误差)。

装袋法为随机森林的特例。对于随机森林，如果令$m = p$，则为装袋法。

在每个节点，装袋法均将所有变量作为候选的分裂变量。由于使用所有特征变量进行分裂，故偏差较小，但不同决策树之间相关性较强，导致方差较大。

在另一极端，如果m很小，虽然决策树之间很不相关，可降低方差，但由于每次用于分裂的变量太少，导致偏差较大。

随机森林最重要的调节参数就是m；在文献中一般称为“*mtry*”，而 sklearn 则称为“`max_features`”。

调节参数m的选择，涉及偏差与方差的权衡，可通过袋外误差或交叉验证来确定，即选择一个最优参数m，使得袋外误差或交叉验证误差最小化。

另一方面，对于随机森林，自助样本的数目B，则不太重要。通常选择$B=500$或更多。

由于随机森林使用B棵决策树的预测结果进行平均或多数投票，故增加B并不会导致过拟合(这由大数定律所保证)。

一般选择足够大的B，使得袋外误差或交叉验证误差不再随着B增大而下降即可。

12.6 变量重要性

与单棵决策树相比，随机森林包含很多决策树(比如 500 棵决策树的平均)，故无法像单棵决策树那样进行解释。

对于随机森林，应如何度量“变量重要性”(variable importance)？

由于在每次节点分裂，仅使用一个变量，故容易区分每个变量的贡献，考察该分裂变量使得残差平方和(或基尼指数)下降多少。

对于每个变量，在随机森林的每棵决策树，可度量由该变量所导致的分裂准则函数(残差平方和或基尼指数)之下降幅度。

针对此下降幅度，对每棵决策树进行平均，即为对该变量重要性的度量。将每个特征变量的重要性依次排序画图，即为**变量重要性图**(variable importance plot)。

12.7 偏依赖图

我们依然感兴趣于每个变量对于y的边际效应(marginal effects)。

对于特征向量$\mathbf{x}=(x_1\ x_2\cdots x_p)'$，假设$y=f(\mathbf{x})$，但函数$f(\cdot)$无解析表达式(比如随机森林)。

不失一般性，考虑第 1 个特征变量x_1对y的边际效应：

$$\frac{\partial y}{\partial x_1}=\frac{\partial f(x_1,x_2,\cdots,x_p)}{\partial x_1} \qquad (12.11)$$

边际效应$\frac{\partial y}{\partial x_1}$依赖于其他变量$(x_2,\cdots,x_p)$的取值，一般来说不是常数(除非是线性模型)。

考虑在函数$y=f(x_1,x_2,\cdots,x_p)$中，将其他变量$(x_2,\cdots,x_p)$对于y的影响通过积分平均掉：

$$\phi(x_1)\equiv \mathrm{E}_{x_2,\cdots,x_p}\ f(x_1,x_2,\cdots,x_p) \qquad (12.12)$$

其中，期望算子$\mathrm{E}_{x_2,\cdots,x_p}(\cdot)$对变量$(x_2,\cdots,x_p)$求期望，故在上式右边已将$(x_2,\cdots,x_p)$积分积掉，故所得结果$\phi(x_1)$只是$x_1$的函数。

由于$f(\cdot)$无解析表达式，一般很难直接计算此期望。

使用统计学方法，以样本均值替代总体均值$\mathrm{E}_{x_2,\cdots,x_p}(\cdot)$可得：

$$\hat{\phi}(x_1) \equiv \frac{1}{n}\sum_{i=1}^{n} f(x_1, x_{i2}, \cdots, x_{ip}) \qquad (12.13)$$

任意给定x_1，均可计算$\hat{\phi}(x_1)$，并可画出$(x_1, \hat{\phi}(x_1))$的图像，称为**偏依赖图**(Partial Dependence Plot)。

偏依赖图适用于任何 $f(\cdot)$ 无解析表达式的黑箱(black box)方法。

也可同时考虑多个变量对于 y 的偏影响。

比如，响应变量 y 对于 (x_1, x_2) 的偏依赖可定义为

$$\hat{\phi}(x_1, x_2) \equiv \frac{1}{n}\sum_{i=1}^{n} f(x_1, x_2, x_{i3}, \cdots, x_{ip}) \qquad (12.14)$$

12.8 回归问题的随机森林 Python 案例

对于回归问题，我们以波士顿房价数据 `boston`，演示随机森林的 Python 操作。

* 详见教材，以及配套 Python 程序（现场演示）。

12.9 分类问题的随机森林 Python 案例

作为分类问题的随机森林案例，我们使用 UCI Machine Learning Repository 的声呐数据 Sonar 来演示。

响应变量为 Class，表示声呐回音来自“金属筒”(Metal Cylinder，记为 M)还是“岩石”(Rock，记为 R)。

特征变量共 60 个(V1-V60)，表示在不同角度(aspect angle)与频道(frequency band)下，声呐反射信号的能量。研究目的是为了区分这些声呐信号究竟来自金属还是岩石。

* 详见教材，以及配套 Python 程序（现场演示）。