

華東理工大學

# 模式识别大作业

题 目	_____ Otto Group 商品识别 _____
学 院	_____ 信息科学与工程 _____
专 业	_____ 控制科学与工程 _____
组 员	_____ 张贤益 (Y30180696) _____
	_____ 倪佳能 (Y30180667) _____
	_____ 闵佳峰 (Y30180665) _____
	_____ 朱思敏 (Y30180700) _____
	_____ 严爱俐 (Y30180687) _____
指导教师	_____ 赵海涛 _____

完成日期： 2018 年 12 月 15 日

# 目录

<b>1 项目介绍</b>	<b>2</b>
1.1 整体目标	2
1.2 问题关键	2
1.3 与课程联系	2
1.4 算法筛选	3
1.5 组内分工	3
<b>2 理论基础</b>	<b>4</b>
2.1 SVM	4
2.1.1 线性 SVM 与非线性 SVM	4
2.1.2 核技巧与常用核函数	4
2.2 神经网络	5
2.2.1 基本结构	5
2.2.2 分类实现	6
2.3 集成算法	6
2.3.1 bagging	7
2.3.2 随机森林	7
2.3.3 Boosting	9
2.3.3 Adaboost	10
2.3.4 Gradient boosting	11
<b>3 工具箱与框架</b>	<b>13</b>
3.1 LibSVM	13
3.2 Sklearn	13
<b>4 程序实现</b>	<b>15</b>
4.1 SVM in Matlab	15
4.2 Nerual Network in Python	18
4.3 RandomForest in Python	19
4.4 Adaboost in Python	21
4.5 Gradient Boosting in Python	22
<b>5 结果与讨论</b>	<b>24</b>
5.1 模型分析	24

5.1.1 支持向量机模型 .....	24
5.1.2 深度学习模型.....	24
5.1.3 集成学习模型.....	24
5.1.4 Adaboost 模型.....	25
5.1.5 Gradient Boosting 模型.....	25
5.2 Lintcode 分类结果 .....	26

# 1 项目介绍

近年来，大数据已成为热门话题。由于计算机技术和数据存储容量的快速发展，世界上所有科学，工程和商业领域的的数据量都在不断增长，迫切需要自动数据分析工具来准确检测大量数据中包含的知识，引入数据挖掘作为相应的解决方案。

数据挖掘吸引了很多人。为了改进他们的研究方法，数据科学家需要大量的实际数据。另一方面，公司需要更准确的模型来进行预测。数据预测竞赛通常是研究人员和公司的正确选择。本报告结合课上所学习的神经网络，SVM，Boosting 等算法，针对于 LintCode 上关于商品预测分类的项目进行数据分析实战演练。

## 1.1 整体目标

Otto Group 是世界上最大的电子商务公司之一，在全世界范围内，它每天会卖出数百万件商品。每件商品所属的类别分别是 Class\_1~ Class\_9。对于这家公司的来说，货物供给和需求分析是非常重要的信息。现给定一些商品的多个特征，需要设计一个算法模型来判断一个商品所属的类别。该项目的目标是将电子商务产品数据集上的分类学习模型应用于超过 200,000 种产品的 93 个特征，从而获得具有高准确度的预测模型，以识别未来的产品类别。

## 1.2 问题关键

鉴于有关电子商务产品的数据集包含超过 200,000 种产品的 93 种功能，该项目旨在建立一个能够区分 9 种主要产品类别的产品的预测模型。一些选定的分类学习模型将由包含每个产品的相应类别的数据集进行训练。为了在应用的模型之间进行比较，将使用交叉验证来评估准确度，然后将选择具有相对较高精度的模型。

## 1.3 与课程联系

该项目不仅有助于实现本次模式识别课程的目标，而且还将我们在课堂上学到的知识应用到实践中。该项目遵循课程目标，即学习模式识别中的高级知识和实施。作为模式识别中的重要组成部分，分类有许多现实世界的应用，如商业营销细分，互联网搜索结果分组等。在这个项目中，我们使用分类来区分主要类别

之间的电子商务产品，这直接帮助我们从小机器学习角度获得处理实际数据的实践技能。

## 1.4 算法筛选

在所有分类技术中，其中相当一部分在属性值类型和属性大小方面具有限制和偏好。此外，简单地在特定数据集上应用学习模型通常不会产生“最佳”分析结果。在本次报告中，我们在数据集上应用尽可能多的分类模型。在对所有应用模型的精度进行比较后，我们选择具有相对最高精度的模型。我们分别采用基本的机器学习分类模型（SVM），集成学习相关算法（Random Forest，Adaboost，Gradientboost）以及神经网络算法对模型进行分类预测。

## 1.5 组内分工

在这个项目中，我们尝试了 5 种分类模型，分别是：SVM，深度神经网络，随机森林，Adaboost，Gradient Boosting。

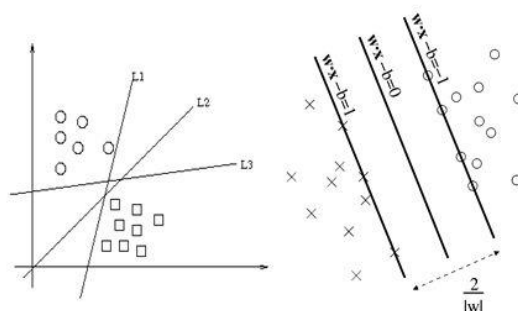
其中，闵佳峰负责 SVM 算法的算法研究与程序实现，严爱俐负责深度神经网络的算法研究与程序实现，倪佳能负责随机森林的算法研究与程序实现，朱思敏负责 Adaboost 的算法研究与程序实现，张贤益负责 Gradient Boosting 的算法研究与程序实现。本报告由本小组所有成员各自报告汇总而成。

本报告中的所有相关程序见：[https://github.com/Iker-zxy/Class-for-Pattern-Recognize/tree/master/Team%20Project Otto%20Group](https://github.com/Iker-zxy/Class-for-Pattern-Recognize/tree/master/Team%20Project%20Otto%20Group)

## 2 理论基础

### 2.1 SVM

SVM 即支持向量机，是一个有监督的学习模型，通常用来进行模式识别、分类以及回归分析。SVM 是一种典型的二分类模型，其主要思想为找到空间中的一个更够将所有数据样本划开的超平面，并且使得本本集中所有数据到这个超平面的距离最短。



#### 2.1 确定性过程

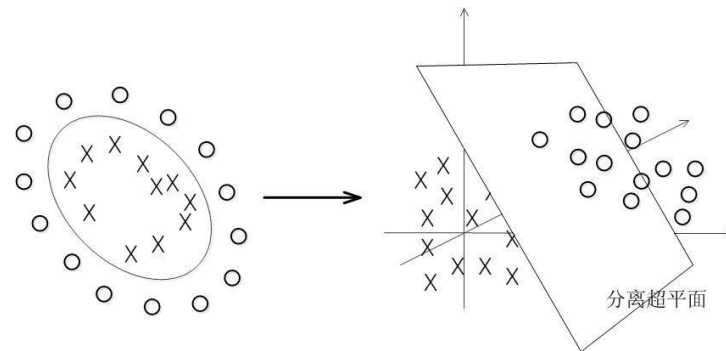
##### 2.1.1 线性 SVM 与非线性 SVM

支持向量机在高维或无限维空间中构造超平面或超平面集合，其可以用于分类、回归或其他任务。直观来说，分类边界距离最近的训练数据点越远越好，因为这样可以缩小分类器的泛化误差。尽管原始问题可能是在有限维空间中陈述的，但用于区分的集合在该空间中往往线性不可分。为此，有人提出将原有限维空间映射到维数高得多的空间中，在该空间中进行分离可能会更容易。为了保持计算负荷合理，人们选择适合该问题的核函数 $K(x, y)$ 来定义 SVM 方案使用的映射，以确保用原始空间中的变量可以很容易计算点积。高维空间中的超平面定义为与该空间中的某向量的点积是常数的点的集合。定义超平面的向量可以选择在数据基中出现的特征向量 $x_i$ 的图像的参数的线性组合。通过选择超平面，被映射到超平面上的特征空间中的点集 $x$ 由以下关系定义： $\sum_i \alpha_i K(x_i, x) = \text{constant}$  注意，如果随着 $y$  逐渐远离 $x$ ， $k(x, y)$ 变小，则求和中的每一项都是在衡量测试点  $x$  与对应的数据基点  $x_i$  的接近程度。这样，上述内核的总和可以用于衡量每个测试点相对于待分离的集合中的数据点的相对接近度。

##### 2.1.2 核技巧与常用核函数

对于线性不可分问题，SVM 可以通过一个非线性映射  $p$ ，把样本空间映射到一

个高维乃至无穷维的特征空间中（Hilbert 空间），使得在原来的样本空间中非线性可分的问题转化为在特征空间中的线性可分的问题。如图所示



## 2.2 核函数下的空间变换

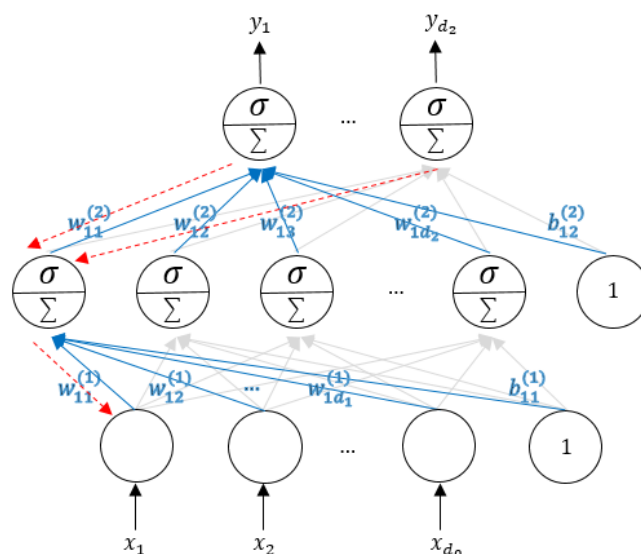
非线性映射  $p$  对应不同的核函数，常用的核函数有如下四种：

- ①线性核函数  $K(x, y) = x \cdot y$
- ②多项式核函数  $K(x, y) = [(x \cdot y) + 1]^d$
- ③高斯（RBF）核函数  $K(x, y) = \exp(-|x - y|^2/d^2)$
- ④sigmoid 核函数  $K(x, y) = \tanh(a(x \cdot y) + b)$

## 2.2 神经网络

在机器学习和认知科学领域，人工神经网络 (ANNs) 是受生物神经网络的启发而产生的一系列的统计学习算法，神经网络在所有的执行功能和并行的单位上的表现形式与生物神经网络类似，而不是在各模块分配的子任务有一个清晰的界定。

### 2.2.1 基本结构



## 2.3 神经网络基本结构

第一层是输入神经元，它通过突触向第二层神经元发送数据，然后通过更多的突触向第三层输出神经元发送数据。它的激活函数采用 sigmoid 函数，采用 BP 算法训练的多层前馈神经网络。越复杂的系统会有越多的神经元层，而这些神经元增加了系统的输入神经元层和输出神经元层。在神经突触上储存的参数称为“权重”，它是对计算中的数据给予不同的比重。

### 2.2.2 分类实现

不同的神经网络方法被运用于处理文档分类的问题。虽然有些文档分类问题是使用最简单的神经网络形式，这种形式被称为感知器，它只包括一个输入层和一个输出层，要建立其他更复杂的神经网络是在输入层和输出层之间增加一个隐藏层。一般来说，这些前馈网络至少由三层组成(即一个输入层、一个输出层和至少一个隐藏层)，并且利用反向传播作为学习机制。然而，这种相对比较老的感知机方法的表现性能依旧很好。

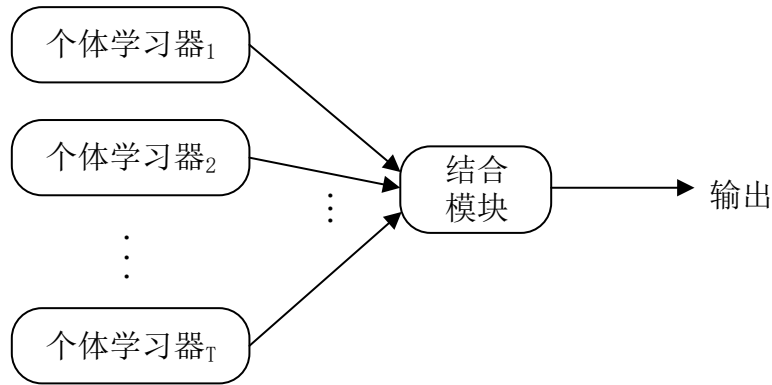
神经网络的优点是能够很好地处理噪声或矛盾数据。此外，某些类型的神经网络能够运用于处理模糊逻辑，但其必须从作为学习机制的反向传播转变为另一种反向传播。神经网络的高灵活的优点带来了计算成本高的缺点。另一个缺点是神经网络很难对结果作出解释。

## 2.3 集成算法

集成学习(ensemble learning)通过构建并结合多个学习器来完成学习任务，有时也被称为多分类器系统(multi-classifier system)、基于委员会的学习(committee-based learning)等。图 2-1 显示出集成学习的一般结构：先产生一组“个体学习器”(individual learner)，再用某种策略将它们结合起来。

根据个体学习器的生成方式，目前的集成学习方法大致可分为两大类，即个体学习器间存在强依赖关系、必须串行生成的序列化方法，以及个体学习器间不存在强依赖关系、可同时生成的并行化方法；前者的代表是 Boosting，后者的代表是 Bagging 和“随机森林”(Random Forest)。





2.4 集成学习示意图

### 2.3.1 bagging

Bagging 是并行式集成学习方法最著名的代表，它直接基于自助采样法 (bootstrap sampling)。给定包含  $m$  个样本的数据集，首先随机取出一个样本放入采样集中，再把该样本放回初始数据集，使得下次采样时该样本仍有可能被选中，这样，经过  $m$  次随机采样操作，可以得到含  $m$  个样本的采样集。初始训练集中有的样本在采样集里多次出现，有的则从未出现。

照这样，可以采样出  $T$  个含  $m$  个训练样本的采样集，然后基于每个采样集训练出一个基学习器，再将这些基学习器进行结合，这就是 Bagging 的基本流程。在对预测输出进行结合时，Bagging 通常对分类任务使用简单投票法，对回归任务使用简单平均法。若分类预测时出现两个类收到同样票数的情形，则最简单的做法是随机选择一个，也可进一步考察学习器投票的置信度来确定最终胜者。

在 Bagging 的每轮随机采样中，训练集中大约有 36.8% 的数据没有被采样集采集到。对于这部分没采集到的数据，我们常常称之为袋外数据 (Out Of Bag, 简称 OOB)。这些数据没有参与训练集模型的拟合，因此可以用来检测模型的泛化能力。

### 2.3.2 随机森林

随机森林 (Random Forest, 简称 RF) 是 Bagging 的一个扩展变体。RF 在以决策树为基学习器构建 Bagging 集成的基础上，进一步在决策树的训练过程中引入了随机属性选择。具体来说，传统决策树在选择划分属性时是在当前结点的属性集合（假定有  $d$  个属性）中选择一个最优属性；而在 RF 中，对基决策树的每个结点，先从该结点的属性集合中随机选择一个包含  $k$  个属性的子集，然后再从这

个子集中选择一个最优属性用于划分。这里的参数  $k$  控制了随机性的引入程度：若令  $k=d$ ，则基决策树的构建与传统决策树相同；若令  $k=1$ ，则是随机选择一个属性用于划分；一般情况下，推荐值。

随机森林的收敛性与 Bagging 相似。如图 2-2 所示，随机森林的起始性能往往相对较差，特别是在集成中只包含一个基学习器时。这很容易理解，因为通过引入属性扰动，随机森林中个体学习器的性能往往有所降低。然而，随着个体学习器数目的增加，随机森林通常会收敛到更低的泛化误差。值得一提的是，随机森林的训练效率常优于 Bagging，因为在个体决策树的构建过程中，Bagging 使用的是“确定型”决策树，在选择划分属性时要对结点的所有属性进行考察，而随机森林使用的“随机型”决策树则只需考察一个属性子集。

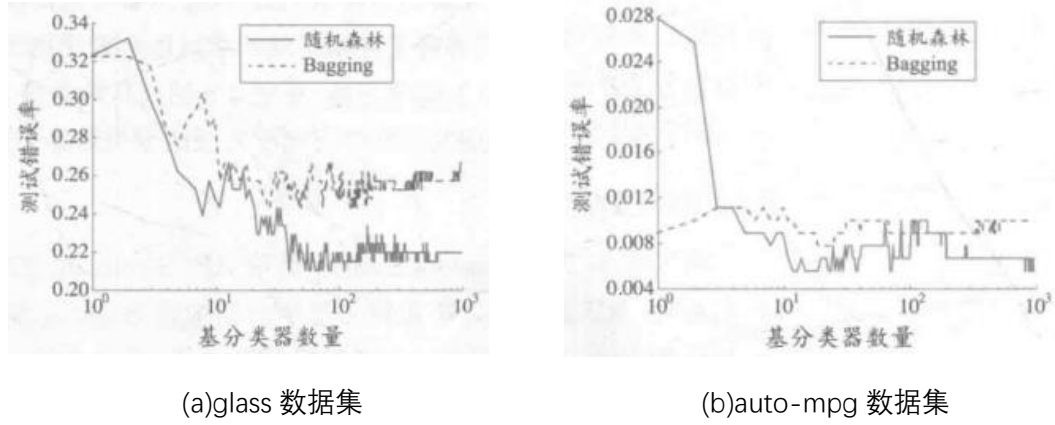


图 2.5 在两个 UCI 数据上，集成规模对随机森林与 Bagging 的影响

特征选择目前比较流行的方法是信息增益、增益率、基尼系数和卡方检验。随机森林采用的 CART 决策树基于基尼系数（GINI）选择特征，他的选择的的标准就是每个子节点达到最高的纯度，即落在子节点中的所有观察都属于同一个分类，此时基尼系数最小，纯度最高，不确定度最小。对于一般的决策树，假如总共有  $K$  类，样本属于第  $k$  类的概率为： $p_k$ ，则该概率分布的基尼指数为：

$$Gini(p) = \sum_{k=1}^K p_k(1-p_k) = 1 - \sum_{k=1}^K p_k^2 \quad (2-1)$$

基尼指数越大，说明不确定性就越大；基尼系数越小，不确定性越小，数据分割越彻底，越干净。对于 CART 树而言，GINI 系数为：

$$Gini(p) = 2p(1-p) \quad (2-1)$$

在历每个特征的每个分割点时，若使用特征  $A=a$ ，将集合  $D$  划分为两部

分，即满足  $A=a$  的样本集合  $D_1$ ，不满足  $A=a$  的样本集合  $D_2$ 。则在特征  $A=a$  的条件下  $D$  的基尼指数为：

$$Gini(D, A) = \frac{|D_1|}{|D|} Gini(D_1) + \frac{|D_2|}{|D|} Gini(D_2) \quad (2-3)$$

其中  $Gini(D, A)$  表示集合  $D$  的不确定性； $Gini(D_1)$  表示经过  $A=a$  分割后的集合  $D$  的不确定性。

随机森林中的每棵 CART 决策树都是通过不断遍历这棵树的特征子集的所有可能的分割点，寻找 GINI 系数最小的特征的分割点，将数据集分成两个子集，直至满足停止条件为止。

在抗过拟合的问题上，RF 对决策树做了改进。对于普通的决策树，我们将会在节点上所有的  $m$  个样本特征中选择一个最优的特征来做决策树的左右子树划分。但是 RF 的每个树，其实选用的特征是一部分，在这些少量特征中，选择一个最优的特征来做决策树的左右子树划分，将随机性的效果扩大，进一步增强了模型的泛化能力。

假设每棵树选取  $n$  个特征， $n$  越小，此时模型对于训练集的拟合程度会变差，偏差增加，但是会泛化能力更强，模型方差减小， $n$  越大则相反。在实际使用中，一般会将  $n$  的取值作为一个参数，通过开启 OOB 验证或使用交叉验证，不断调整参数以获取一个合适的  $n$  的值。

### 2.3.3 Boosting

Boosting 算法是一种常用的统计学习算法，应用广泛且有效。在分类问题中，它通过改变训练样本的权重，学习多个弱分类器，并将这些弱分类器进行线性组合，提高分类的性能。

Boosting 算法主要涉及到两个部分，加法模型和前向分步算法。加法模型意思就是强分类器由一系列弱分类器线性组合而成。一般组合形式如下：

$$F_M(x; P) = \sum_{m=1}^n \beta_m h(x; a_m) \quad (2-4)$$

其中， $h(x; a_m)$  就是多个弱分类器， $a_m$  是弱分类器学习到的最优参数， $\beta_m$  是弱学习在强分类器中所占比重， $P$  是所有  $a_m$  和  $\beta_m$  的组合。这些弱分类器进行线性组合就构成强分类器。

前向分步就是说在训练过程中，下一轮迭代产生的分类器是在上一轮的基础

上训练得来的。也就是可以写成这样的形式：

$$F_m(x) = F_{m-1}(x) + \beta_m h_m(x; a_m) \quad (2-5)$$

### 2.3.3 Adaboost

采用不同的损失函数，Boosting 算法衍生出了不同的类型。其中，Adaboost(Adaptive Boosting)算法就是损失函数为指数函数时的 Boosting 算法。

对于 Boosting 算法，存在以下两个问题：

- (1) 如何调整训练集，使得在训练集上训练的弱分类器得以进行；
- (2) 如何将训练得到的各个弱分类器联合起来形成强分类器。

针对以上两个问题，AdaBoost 算法进行了调整：

(1) 使用加权后选取的训练数据代替随机选取的训练样本，这样将训练的焦点集中在比较难分的训练数据样本上

(2) 将弱分类器联合起来，使用加权的投票机制代替平均投票机制。让分类效果好的弱分类器具有较大的权重，而分类效果差的分类器具有较小的权重。

下面具体介绍一下 Adaboost 算法。

假设给定一个二分类的训练数据集

$$T = \{(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\} \quad (2-6)$$

其中，每个样本点由实例与标记组成。实例  $x_i \in \chi \subseteq \mathbf{R}^n$ ，标记  $y_i \in Y = \{-1, +1\}$ ， $\chi$  是实例空间， $Y$  是标记集合。Adaboost 利用以下算法，从训练数据中学习一系列弱分类器或基本分类器，并将这些弱分类器线性组合成为一个强分类器。

输入：训练数据集  $T = \{(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\}$ ，其中  $x_i \in \chi \subseteq \mathbf{R}^n$ ， $y_i \in Y = \{-1, +1\}$ ；弱学习算法；

输出：最终分类器  $G(X)$ 。

- (1) 初始化训练数据的权值分布

$$D_1 = (w_{11}, \dots, w_{1i}, \dots, w_{1N}), w_{1i} = \frac{1}{N}, i = 1, 2, \dots, N \quad (2-7)$$

- (2) 对  $m = 1, 2, \dots, M$ 。

- (a) 使用具有权值分布  $D_m$  的训练数据集学习，得到基本分类器

$$G_m(x): \chi \rightarrow \{-1, +1\} \quad (2-8)$$

(b) 计算 $G_m(x)$ 在训练数据集上的分类误差率

$$e_m = \sum_{i=1}^N P(G_m(x_i) \neq y_i) = \sum_{i=1}^N w_{mi} I(G_m(x_i) \neq y_i) \quad (2-9)$$

(c) 计算 $G_m(x)$ 的系数

$$\alpha_m = \frac{1}{2} \log \frac{1 - e_m}{e_m} \quad (2-10)$$

(这里的对数是自然对数)。

(d) 更新训练数据集的权值分布

$$D_{m+1} = (w_{m+1,1}, \dots, w_{m+1,i}, \dots, w_{m+1,N}) \quad (2-11)$$

$$w_{m+1,i} = \frac{w_{mi}}{Z_m} \exp(-\alpha_m y_i G_m(x_i)), \quad i = 1, 2, \dots, N \quad (2-12)$$

这里,  $Z_m$ 是规范化因子

$$Z_m = \sum_{i=1}^N w_{mi} \exp(-\alpha_m y_i G_m(x_i)) \quad (2-13)$$

它使 $D_{m+1}$ 成为一个概率分布。

(3) 构建基本分类器的线性组合

$$f(x) = \sum_{m=1}^M \alpha_m G_m(x) \quad (2-14)$$

得到最终分类器

$$G(x) = \text{sign}(f(x)) = \text{sign}\left(\sum_{m=1}^M \alpha_m G_m(x)\right) \quad (2-15)$$

### 2.3.4 Gradient boosting

Boosting 的基本思想是通过某种方式使得每一轮基学习器在训练过程中更加关注上一轮学习错误的样本, 区别在于是采用何种方式。AdaBoost 采用的是增加上一轮学习错误样本的权重的策略, 而在 Gradient Boosting 中则将负梯度作为上一轮基学习器犯错的衡量指标, 在下一轮学习中通过拟合负梯度来纠正上一轮犯的错误。这里的关键问题是: 为什么通过拟合负梯度就能纠正上一轮的错误了? Gradient Boosting 的发明者给出的答案是: 函数空间的梯度下降。

Gradient Boosting 用于回归和分类问题的机器学习技术, 其以弱预测模型 (通常是决策树) 的集合的形式产生预测模型。它像其他增强方法一样以阶段方

式构建模型，并通过允许优化任意可微分损失函数来推广它们。

Gradient Boosting 的算法流程可以大致分为

1. 初始化:  $f_0(x) = \arg \min_{\gamma} \sum_{i=1}^N L(y_i, \gamma)$
2. for  $m=1$  to  $M$ :
  - (a) 计算负梯度:  $\tilde{y}_i = -\frac{\partial L(y_i, f_{m-1}(x_i))}{\partial f_{m-1}(x_i)}, \quad i = 1, 2 \dots N$
  - (b) 通过最小化平方误差, 用基学习器  $h_m(x)$  拟合  $\tilde{y}_i$ ,  $w_m = \arg \min_w \sum_{i=1}^N [\tilde{y}_i - h_m(x_i; w)]^2$
  - (c) 使用line search确定步长  $\rho_m$ , 以使  $L$  最小,  $\rho_m = \arg \min_{\rho} \sum_{i=1}^N L(y_i, f_{m-1}(x_i) + \rho h_m(x_i; w_m))$
  - (d)  $f_m(x) = f_{m-1}(x) + \rho_m h_m(x; w_m)$
3. 输出  $f_M(x)$

## 3 工具箱与框架

### 3.1 LibSVM

LibSVM 是台湾林智仁 (Chih-Jen Lin) 教授 2001 年开发的一套支持向量机的库，这套库运算速度还是挺快的，可以很方便的对数据做分类或回归。由于 LibSVM 程序小，运用灵活，输入参数少，并且是开源的，易于扩展，因此成为目前国内应用最多的 SVM 的库。

#### (1) LibSVM 的数据格式

Label 1:value 2:value ...

其中，Label 是类别的标识；1:value 表示该样本的第一个特征的数值。

#### (2) svm-scale 的用法

svm-scale 是用来对原始样本进行缩放的，范围可以自己定，一般是  $[0, 1]$  或  $[-1, 1]$ 。缩放的目的主要是：1、防止某个特征过大或过小，从而在训练中起的作用不平衡；2、为了提高计算速度。在核函数计算中，会用到内积运算或指数运算，不平衡的数据可能造成计算困难。

#### (3) svm-train 的用法

Svmtrain 可以对数据集进行交叉验证，也可以训练数据集，并获得 SVM 模型。

用法：svmtrain [options] training\_set\_file [model\_file]

### 3.2 Sklearn

Scikit learn 也简称 sklearn，是机器学习领域当中最知名的 python 模块之一。它包含了很多种机器学习的方式，从功能来分，包括：

- Classification 分类
- Regression 回归
- Clustering 非监督分类
- Dimensionality reduction 数据降维
- Model Selection 模型选择
- Preprocessing 数据预处理

从 API 模块来分，包括以下几类，等等：

- 聚类: sklearn.cluster
- KNN最近邻: sklearn.neighbors
- logistic regression逻辑回归:  
sklearn.linear\_model.LogisticRegression
- svm支持向量机: sklearn.svm
- Naive Bayes朴素贝叶斯: sklearn.naive\_bayes
- Decision Tree决策树: sklearn.tree
- Neural network神经网络: sklearn.neural\_network

Sklearn官网提供了一个流程图，见图2-3，学习Sklearn时，需要先了解一下模型方法，然后根据图中的判断条件选择的算法。

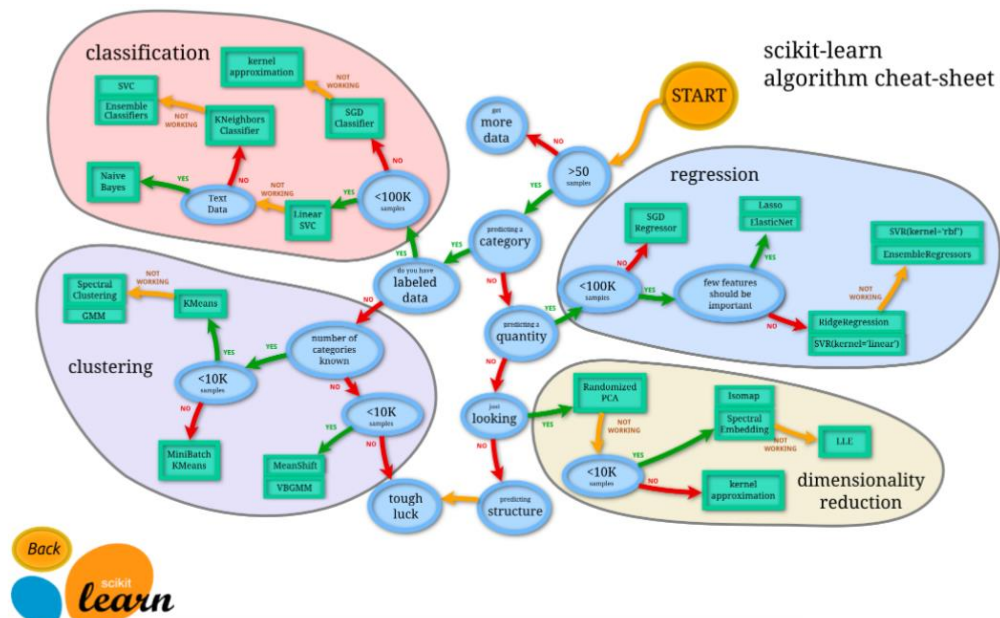


图 3-1 sklearn 包含的算法流程图



## 4 程序实现

【程序】[https://github.com/Iker-zxy/Class-for-Pattern-Recognize/tree/master/Team%20Project\\_Otto%20Group](https://github.com/Iker-zxy/Class-for-Pattern-Recognize/tree/master/Team%20Project_Otto%20Group)

### 4.1 SVM in Matlab

```
1 % 数据预处理
2 % 首先从train.csv中获取训练样本和类别，然后将同类别的样本和类别集中，得到有序的训练集，最后，对训练集作归一化处理。
3 [~, ~, rawFile] = xlsread('D:\Software\MATLAB\Project\SVM\libsvm-master\matlab\train.csv');
4 rawData = cell2mat(rawFile(2:end,2:(end-1))); %原始数据集
5 rawTargets = char(rawFile(2:end,end)); %原始数据集的类别
6 [m, n] = size(rawData); %m为样本数, n为特征数
7 data = zeros(m, n); %有序数据集
8 targets = []; %有序数据集的类别
9 c = 9; %类别总数9
10 for i = 1:c
11     str = strcat('class_', num2str(i));
12     IndexArray{i} = findStrInArray(rawTargets, str); %将相同类别的数据放在一个元胞中
13 end
14 k = 0;
15 for i = 1:c
16     for j = 1:length(IndexArray{i})
17         k = k + 1;
18         data(k,:) = rawData(IndexArray{i}(j),:); %对原始数据集进行排序
19     end
20     targets = [targets; i * ones(length(IndexArray{i}),1)]; %对每个类别贴标签
21 end
22 [stdData, PS] = mapstd(data',0,1); %将样本归一化—均值为零，方差为一
23 stdData = stdData'; %归一化样本
24
25 % 使用交叉验证和网格搜索得到最优的c和g
26 % 高斯核函数的主要参数有惩罚系数c以及参数g，首先将c和g设定在较大范围，如 $2^{(-10)} \sim 2^{10}$ ，并且使参数以较大步距增加，如每次迭代后参数乘以2，然后调用svmtrain函数分别在不同参数下进行交叉验证，最后绘制验证成功率的等高线图，选出图中验证成功率较高的区域，缩小c和g的范围，并缩小参数的步距，重新进行交叉验证.....上述步骤循环往复，最终获取最高验证成功率对应的c和g。
27 function [bestp,bestc,bestg,pArray] =
28 GridSVMcg(train_label,train_data,cmin,cmax,gmin,gmax,v,cstep,gstep,accstep)
29 % [bestacc,bestc,bestg,cg] =
30 SVMcgPP(train_label,train,cmin,cmax,gmin,gmax,v,cstep,gstep,accstep)
31 %
32 % train_label:训练集标签.要求与libsvm工具箱中要求一致.
33 % train:训练集.要求与libsvm工具箱中要求一致.
34 % cmin:惩罚参数c的变化范围的最小值(取以2为底的对数后),即  $c_{min} = 2^{(cmin)}$ .默认为 -5
35 % cmax:惩罚参数c的变化范围的最大值(取以2为底的对数后),即  $c_{max} = 2^{(cmax)}$ .默认为 5
36 % gmin:参数g的变化范围的最小值(取以2为底的对数后),即  $g_{min} = 2^{(gmin)}$ .默认为 -5
37 % gmax:参数g的变化范围的最小值(取以2为底的对数后),即  $g_{min} = 2^{(gmax)}$ .默认为 5
38 %
39 % v:cross validation的参数,即给测试集分为几部分进行cross validation.默认为 3
40 % cstep:参数c步进的大小.默认为 1
41 % gstep:参数g步进的大小.默认为 1
42 % accstep:最后显示准确率图时的步进大小.默认为 1.5
43
44 % 对缺省参数进行初始化
```

```

43 if nargin < 3
44     accstep = 1.5;
45     v = 3;
46     cstep = 1;
47     gstep = 1;
48     gmax = 5;
49     gmin = -5;
50     cmax = 5;
51     cmin = -5;
52 elseif nargin < 4
53     accstep = 1.5;
54     v = 3;
55     cstep = 1;
56     gstep = 1;
57     gmax = 5;
58     gmin = -5;
59     cmax = 5;
60 elseif nargin < 5
61     accstep = 1.5;
62     v = 3;
63     cstep = 1;
64     gstep = 1;
65     gmax = 5;
66     gmin = -5;
67 elseif nargin < 6
68     accstep = 1.5;
69     v = 3;
70     cstep = 1;
71     gstep = 1;
72     gmax = 5;
73 elseif nargin < 7
74     accstep = 1.5;
75     v = 3;
76     cstep = 1;
77     gstep = 1;
78 elseif nargin < 8
79     accstep = 1.5;
80     cstep = 1;
81     gstep = 1;
82 elseif nargin < 10
83     accstep = 1.5;
84 end
85
86 % 获取c和g的组合
87 cArray = cmin:cstep:cmax;
88 gArray = gmin:gstep:gmax;
89 m = length(cArray);    %c数组的长度
90 n = length(gArray);    %g数组的长度
91 A = zeros(m * n, 2);   %c和g的组合
92 pArray = [];
93 k = 1;
94 for i = 1:m
95     for j = 1:n

```

```

96     A(k, 1) = cArray(i);
97     A(k, 2) = gArray(j);
98     k = k + 1;
99     end
100 end
101
102 % 对训练集进行交叉验证, 从中选择交叉验证概率最高, c最小的c和g组合
103 bestc = 0;
104 bestg = 0;
105 bestp = 0;
106 b = 2; %指数的底
107 MyPar = parpool; %打开并行处理池
108 parfor i = 1:m * n
109     cmd = ['-v ', num2str(v), ' -c ', num2str( b ^ A(i, 1) ), ' -g ', num2str( b ^ A(i,
110 2)), ' -t 2', ' -w1 12', ' -w2 1', ' -w3 2', ' -h 0']; %高斯核函数
111     pArray(i) = svmtrain(train_label, train_data, cmd); %交叉验证概率计算
112 end
113 delete(MyPar) %计算完成后关闭并行处理池
114 % parfor i = 1:m
115 %     for j = 1:n
116 %         cmd = ['-v ', num2str(v), ' -c ', num2str( basenum^X(i,j) ), ' -g ', num2str(
117 basenum^Y(i,j) )];
118 %         cg(i,j) = svmtrain(train_label, train, cmd);
119 %     end
120 % end
121 %     if cg(i,j) > bestacc
122 %         bestacc = cg(i,j);
123 %         bestc = basenum^X(i,j);
124 %         bestg = basenum^Y(i,j);
125 %     end
126 %     if ( cg(i,j) == bestacc && bestc > basenum^X(i,j) )
127 %         bestacc = cg(i,j);
128 %         bestc = basenum^X(i,j);
129 %         bestg = basenum^Y(i,j);
130 %     end
131 [bestp, I] = max(pArray);
132 bestc = b ^ A(I, 1);
133 bestg = b ^ A(I, 2);
134 [X, Y] = meshgrid(cmin:cstep:cmax, gmin:gstep:gmax); %获取网格坐标
135 pMatrix = reshape(pArray, [n, m]); %重新排列交叉验证概率数组, 获得矩阵形式
136
137 % 绘制不同c和g的等高线图
138 [C, h] = contour(X, Y, pMatrix, 50:accstep:100); %绘制交叉验证概率矩阵中60%~100%的等
139 高线图
140 xlabel('log2c', 'FontSize', 10);
141 ylabel('log2g', 'FontSize', 10);
142 grid on;
143
144 % 模型预测
145 % 得到最优的参数c和g后, 可以通过这两个参数获得训练模型, 然后调用svmpredict函数对测试集进行预测并
146 输出概率。

```

```

145 %预测测试集
146 [bestp,bestc,bestg,cg] = GridSVMcg(trainTargets, trainData, 4, 12, -4, 4, 3, 1,
1); %获得最优的c和g
147 %[bestcvaccuracy,bestc,bestg,psosvm_option] = psosvmcg(trainTargets, trainData); %
粒子群算法寻找最优的c和g
148 cmd = ['-c ',num2str(bestc),' -g ',num2str(bestg), ' -w1 12', ' -w2 1', ' -w3 2'];
149 model = svmtrain(trainTargets, trainData, cmd); %获得最优训练模型
150 [predict_label, accuracy, dec_values] = svmpredict(testTargets, testData, model);
%测试集预测

```

## 4.2 Nerual Network in Python

```

1 # 导入相关包
2 import keras
3 import pandas as pd
4 import numpy as np
5 from keras.layers import Dense, Activation, Flatten, Convolution2D, Dropout,
MaxPooling2D,BatchNormalization
6 from keras.optimizers import SGD, Adadelta, Adagrad
7 from keras.models import Sequential
8 import tensorflow as tf
9 from keras import backend as K
10 #
11 np.random.seed(42)
12 session_conf = tf.ConfigProto(intra_op_parallelism_threads=1,
inter_op_parallelism_threads=1)
13 tf.set_random_seed(42)
14 sess = tf.Session(graph=tf.get_default_graph(), config=session_conf)
15 K.set_session(sess)
16
17 # 数据预处理 把训练集的分类情况转化成one-hot向量的函数定义
18 def to_one_hot(y):
19     y_temp = np.zeros([y.shape[0], 10]) # 转换为one-hot向量
20     for i in range(y.shape[0]):
21         y_temp[i, y[i]] = 1
22     return y_temp
23
24 # 读取数据
25 train_data =
pd.read_csv(open("D:/Machine_learning/1.Dataset/5.Otto_group/train.csv"))
26 test_data =
pd.read_csv(open("D:/Machine_learning/1.Dataset/5.Otto_group/test.csv"))
27 train_y_raw = train_data["target"]
28 x_label = []
29 for i in range(1,94):
30     x_label.append("feat_%s"%(i))
31 train_x = np.array(train_data[x_label])
32 test_x = np.array(test_data[x_label])
33
34 # 将train_y中形如class_1的数据转换成one-hot向量, 9维
35 train_y = np.zeros([len(train_y_raw),9])
36 for i in range(len(train_y_raw)):
37     label_data = int(train_y_raw[i][-1]) # 取最后一个字就行

```



```

38     train_y[i, label_data-1] = 1
39 print(train_x.shape, train_y.shape, test_x.shape) # (49502, 93) (49502, 9) (12376,
93)
40
41 # 用Keras搭建神经网络模型
42 model = Sequential()
43 model.add(Dense(128, input_shape=(93,), activation="relu"))
44 model.add(Dense(64, activation="relu"))
45 model.add(Dense(32, activation="relu"))
46 model.add(Dense(16, activation="relu"))
47 model.add(Dense(9))
48 model.add(Activation('softmax'))
49 model.summary()
50 model.compile(loss='mean_squared_logarithmic_error', optimizer='adadelta', metrics=
['accuracy'])
51 model.fit(x=train_x, y=train_y, batch_size=2048, nb_epoch=300, verbose=1)
52
53 # 预测答案
54 test_y = model.predict(test_x)
55 print(test_y.shape)
56 answer =
pd.read_csv(open("D:/Machine_learning/1.Dataset/5.Otto_group/sampleSubmission.csv")
)
57 class_list =
["class_1", "class_2", "class_3", "class_4", "class_5", "class_6", "class_7", "class_8", "c
lass_9"]
58 answer[class_list] = answer[class_list].astype(float)
59
60 # 存入答案
61 j = 0
62 for class_name in class_list:
63     answer[class_name] = test_y[:, j]
64     j += 1
65 answer.to_csv("D:/Machine_learning/1.Dataset/5.Otto_group/sampleSubmission.csv", ind
ex=False) # 不要保存索引列

```

### 4.3 RandomForest in Python

```

1 # -*-Otto Group 商品识别_随机森林（数据预处理）-*-
2 # -*-加载训练数据集函数loadTrainSet()-*-
3 # -*-加载测试数据集函数loadTestSet()-*-
4 # -*-评估函数evaluation(label, pred_label)-*-
5 # ----- 答案存入csv文件saveResult(testlabel, filename = "submission.csv") ----- #
6 import csv
7 import random
8 import numpy as np
9
10 # ----- 加载训练数据集 ----- #
11 def loadTrainSet():
12     traindata = []
13     trainlabel = []
14     table = {"class_1":1, "class_2":2, "class_3":3, "class_4":4, "class_5":5,
15             "class_6":6, "class_7":7, "class_8":8, "class_9":9}

```

```

16     with open("train.csv") as f:
17         rows = csv.reader(f)
18         next(rows)
19         for row in rows:                                #row为train.csv文件[序号+每个商品的93个
feat+label](95)
20             l = []
21             for i in range(1,94):
22                 l.append(int(row[i]))                    #l中存入第n个商品的93个feat
23             traindata.append(l)                          #存入train所有商品的feat
24             trainlabel.append(table.get(row[-1]))        #存入train所有商品的label(class
num改为num)
25         f.close()
26
27     traindata = np.array(traindata,dtype="float")
28     trainlabel = np.array(trainlabel,dtype="int")
29     #数据标准化(零均值, 归一化)
30     mean = traindata.mean(axis=0)                        #求均值
31     std = traindata.std(axis=0)                          #求标准差
32     traindata = (traindata - mean)/std
33     x_range = range(len(trainlabel))
34     #打乱数据顺序
35     randomIndex = [i for i in x_range]                   #长度为49502
36     random.shuffle(randomIndex)                          #0-49501顺序打乱
37     traindata = traindata[randomIndex]
38     trainlabel = trainlabel[randomIndex]
39     return traindata,trainlabel
40
41 # ----- 加载测试数据集 ----- #
42 def loadTestSet():
43     testdata = []
44     with open("test.csv") as f:
45         rows = csv.reader(f)
46         next(rows)
47         for row in rows:
48             l = []
49             for i in range(1,94):
50                 l.append(int(row[i]))
51             testdata.append(l)
52     f.close()
53     testdata = np.array(testdata,dtype="float")
54     #数据标准化(零均值, 归一化)
55     mean = testdata.mean(axis=0)
56     std = testdata.std(axis=0)
57     testdata = (testdata - mean)/std
58     return testdata
59
60
61 # ----- 评估函数 ----- #
62 def evaluation(label,pred_label):
63     num = len(label)
64     logloss = 0.0
65     for i in range(num):
66         p = max(min(pred_label[i][label[i]-1],1-10**(-15)),10**(-15))

```

```

67     logloss += np.log(p)
68     logloss = -1*logloss/num
69     return logloss
70
71
72 # ----- 答案存入csv文件 ----- #
73 def saveResult(testlabel,filename = "submission.csv"):
74     with open(filename,'w',newline='') as myFile:
75         mywriter=csv.writer(myFile)
76         byte_temp=
77         ['id','class_1','class_2','class_3','class_4','class_5','class_6','class_7','class_
78         8','class_9']
79         mywriter.writerow(byte_temp)
80         id_num = 1
81         for eachlabel in testlabel:
82             l = []
83             l.append(id_num)
84             l.extend(eachlabel)
85             mywriter.writerow(l)
86             id_num += 1

```

## 4.4 Adaboost in Python

```

1  # 导入包
2  import pandas as pd
3  import numpy as np
4  from sklearn.cross_validation import train_test_split
5  from sklearn.ensemble import AdaBoostClassifier
6  from sklearn.tree import DecisionTreeClassifier
7  #数据的导入
8  data_train=pd.read_csv('train.csv')           #读取训练数据
9  data_test=pd.read_csv('test.csv')             #读取测试数据
10 Y=data_train.target                          #读取训练数据中的target字段输出
11 #对数据进行预处理
12 data_train.dropna(axis=0,how='any',subset=['target'],inplace=True) #删除Score为空值
    的数据行
13 X=data_train.drop(['target','id'], axis=1)
14 X=pd.get_dummies(X)                          #用
    pd.get_dummies方法进行one-hot编码
15 X=X.apply(lambda x:(x-np.min(x))/(np.max(x)-np.min(x))) #min-max标准化 (Min-Max
    Normalization), 也称为离差标准化, 是对原始数据的线性变换, 使结果值映射到[0-1]
16 test_X=data_test.drop(['id'], axis=1)
17 test_X=pd.get_dummies(test_X)
18 test_X=test_X.apply(lambda x:(x-np.min(x))/(np.max(x)-np.min(x)))
19
20 X=np.array(X)
21 X=pd.DataFrame(X)    #加这两行是为了让dataframe的列名变为按序排列的数字, 为什么用原来的列名总会
    出现使得X^T*X为奇异矩阵
22 test_X=np.array(test_X)
23 test_X=pd.DataFrame(test_X)
24 X_total=pd.concat([X,test_X])                #将训练测试集合并起来, 统一进行数据处理
25 cols=[x for i,x in enumerate(X_total.columns) if X_total.iat[0,i]==0] #得到全为0的列
26 X_total=X_total.drop(cols,axis=1)            #删掉全为0的列

```

```

27 X=X_total[0:49502]
28 test_X=X_total[49502:61878]
29
30 # 将数据分为训练集和校验集
31 train_X, valid_X, train_Y, valid_Y = train_test_split(X.values, Y.values,
32 test_size=0.25)
33
34 bdt_real = AdaBoostClassifier(
35     DecisionTreeClassifier(max_depth=8),
36     n_estimators=3000,
37     learning_rate=0.01)
38
39 #bdt_real.fit(X.values, Y.values)
40 bdt_real.fit(train_X, train_Y)
41
42 Y_pred=bdt_real.predict_proba(test_X)
43
44 answer = pd.read_csv(open("./sampleSubmission.csv"))
45 class_list =
46 ["class_1","class_2","class_3","class_4","class_5","class_6","class_7","class_8","c
47 lass_9"]
48 answer[class_list] = answer[class_list].astype(float)
49
50 # 将答案放进去
51 j = 0
52 for class_name in class_list:
53     answer[class_name] = Y_pred[:, j]
54     j += 1
55 answer.to_csv("./submission.csv",index=False) # 不要保存索引列

```

## 4.5 Gradient Boosting in Python

```

1 # 导入包
2 import numpy as np
3 import pandas as pd
4 from sklearn.ensemble import GradientBoostingClassifier, RandomForestClassifier,
5 AdaBoostClassifier, BaggingClassifier
6 import datetime
7 # 导入数据
8 def now():
9     return '_'.join(str(datetime.datetime.now()).split())
10 df_train = pd.read_csv('/home/xyz/Documents/Otto_Group_data/train.csv')
11 df_test = pd.read_csv('/home/xyz/Documents/Otto_Group_data/test.csv')
12 # 定义相关函数
13 def convert_class_label_to_int(class_label):
14     return int(class_label[6:]) - 1
15 def find_means_stds(features_array):
16     means = [np.mean(features_array[:, i]) for i in range(features_array.shape[1])]
17     stds = [np.std(features_array[:, i]) for i in range(features_array.shape[1])]
18     return {'means': means, 'stds': stds}
19 def z_score_feature(feature_slice, mean, std):
20     return (feature_slice - mean) / std

```



```

20 # 数据标签处理
21 renamed_labels = [convert_class_label_to_int(i) for i in df_train['target'].values]
22
23 feature_columns = ['feat_' + str(i + 1) for i in range(93)]
24 df_train[renamed_labels] = renamed_labels
25 mstdict = find_means_stds(df_train[feature_columns].values)
26 for i in range(93):
27     df_train['feat_' + str(i + 1)] = z_score_feature(df_train['feat_' + str(i +
28     1)].values, mstdict['means'][i],
29                                                         mstdict['stds'][i])
30
31 for i in range(93):
32     df_test['feat_' + str(i + 1)] = z_score_feature(df_test['feat_' + str(i +
33     1)].values, mstdict['means'][i],
34                                                         mstdict['stds'][i])
35
36 # 模型建立
37 clf = GradientBoostingClassifier(n_estimators=700, max_depth=7, max_features=20,
38     learning_rate=0.03)
39 clf.fit(df_train[feature_columns].values, df_train[renamed_labels].values)
40 labels_pr = clf.predict_proba(df_test[feature_columns].values)
41 labels_pr_tr = clf.predict_proba(df_train[feature_columns].values)
42 now_name = now()
43 # 模型预测
44 predict_dict = {'id': df_test['id'].values}
45 for i in range(9):
46     predict_dict['class_' + str(i + 1)] = labels_pr[:, i]
47 df_sub = pd.DataFrame(predict_dict)
48 # 输出结果
49 df_sub.to_csv('/home/xyz/Documents/Otto_Group_data/submission2.csv', index = False)

```

## 5 结果与讨论

### 5.1 模型分析

#### 5.1.1 支持向量机模型

支持向量机（SVM）是一组用于分类，回归和异常值检测的监督学习方法。

支持向量机的优点是：

- （1） 在高维空间有效。
- （2） 在尺寸数大于样本数的情况下仍然有效。
- （3） 在决策函数中使用训练点的子集（称为支持向量），因此它也具有内存效率。
- （4） 通用：可以为决策功能指定不同的内核功能。 提供了通用内核，但也可以指定自定义内核。

支持向量机的缺点包括：

- （1） 如果要素数量远远大于样本数量，则该方法可能会导致性能下降。
- （2） SVM 不直接提供概率估计，这些是使用昂贵的五倍交叉验证计算的（请参阅下面的分数和概率）。

#### 5.1.2 深度神经网络模型

深度学习是基于学习数据表示的更广泛的机器学习方法系列的一部分。观察（例如，图像）可以以许多方式表示，例如每个像素的强度值矢量，或者以更抽象的方式表示为一组边缘，特定形状的区域等。一些表示使得从示例学习任务（例如，面部识别）更容易。深度学习的承诺之一是用无人监督或半监督特征的有效算法取代手工制作的特征学习和分层特征提取。本项目的深度学习基于多层前馈人工神经网络，该网络使用反向传播进行随机梯度下降训练。网络可以包含大量隐藏层，这些隐藏层由具有 softmax，整流和最大激活功能的神经元组成。自适应学习速率，速率退火，动量训练，丢失，L1 或 L2 正则化，检查点和网格搜索等高级功能可实现高预测精度。每个计算节点通过多线程（异步）在其本地数据上训练全局模型参数的副本，并通过网络上的模型平均周期性地为全局模型做出贡献。

#### 5.1.3 随机森林模型

为了达到更好的准确性，我们决定尝试集成学习将几个模型混合在一起。在统计学和机器学习中，集成学习方法使用多种学习算法来获得比从任何组成学习算法获得的更好的预测性能。与统计力学中的统计集合（通常是无限的）不同，机器学习集合仅指具体有限的替代模型集，但通常允许在这些备选之间存在更灵活的结构。对于随机森林来说，整体中的每棵树都是从训练集中用替换（即自举样本）绘制的样本构建的。此外，在构建树期间拆分节点时，所选的拆分不再是所有要素中的最佳拆分。相反，拾取的拆分是功能的随机子集中的最佳拆分。由于这种随机性，森林的偏差通常略微增加（相对于单个非随机树的偏差），但由于平均，其方差也减小，通常不仅补偿偏差的增加，因此产生整体更好的模型。

#### 5.1.4 Adaboost 模型

采用不同的损失函数，Boosting 算法衍生出了不同的类型。其中，Adaboost (Adaptive Boosting) 算法就是损失函数为指数函数时的 Boosting 算法。

Adaboost 的主要优点有：

- (1) Adaboost 作为分类器时，分类精度很高；
- (2) 在 Adaboost 的框架下，可以使用各种回归分类模型来构建弱学习器，非常灵活；

- (3) 作为简单的二元分类器时，构造简单，结果可理解；

- (4) 不容易发生过拟合。

Adaboost 的主要缺点有：

对异常样本敏感，异常样本在迭代中可能会获得较高的权重，影响最终的强学习器的预测准确性。

#### 5.1.5 Gradient Boosting 模型

Gradient Boosting 是用于回归问题的机器学习技术，其以弱预测模型的集合的形式产生预测模型，通常是 20 个决策树。它像其他增强方法一样以阶段方式构建模型，并通过允许优化任意可微分损失函数来推广它们。梯度增强方法也可以通过将它们减少到具有合适的损失函数的回归来用于分类问题。

## 5.2 Lintcode 分类结果

在这个项目中，我们尝试了 5 种分类模型（包括整体模型），提交的结果从最开始的 6.23 持续降低到 0.47。我们使用的模型如下：SVM，深度神经网络，随机森林，Adaboost，Gradient Boosting。我们可以获得的最好效果是 Gradient Boosting 的 0.47，因此，我们可以得出结论，即使使用最佳参数，一个纯粹的训练模型也很难获得高精度。对于树模型，参数中树的数量越多，它可以获得的精度越高。但是，为了打破瓶颈，使用集成学习算法是必要的但是耗时。在 Lintcode 上分类效果较好的排名如图所示。


















#	Team Name	Score	Member	Entries	Last
1	SinceJune	0.00000		18	a year ago
2	seagullbird	0.00000		2	a year ago
3	Darkrail	0.00000		2	a year ago
4	jetrunner	0.00000		5	a year ago
5	ChengliangGao	0.00000		2	a year ago
6	jxt	0.00070		1	a year ago
7	HaoliZhang	0.00846		4	a year ago
8	eric074	0.01768		7	a year ago
9	bingege	0.09836		19	a year ago
10	Light@HLi	0.24349		3	a year ago
11	Iker	0.47774		5	5 hours ago
12	blingbling37	0.52158		8	2 days ago
13	Call_me_Xuyifeng	0.52809		10	a day ago
14	cannonni	0.55332		30	2 days ago
15	g7MjP5XtN	0.57771		14	in 8 hours
16	ifreewolf	0.58855		7	5 days ago
17	longest	0.59156		30	11 hours ago

图 5-1 Lintcode 上的提交结果