课程论文题目

李佩玮

摘要

常见的机器学习分类方法通常假设类别是无序的，但很多情况下，类别都自然地带有一定的顺序。例如：高，矮，很明显，高 > 矮。那么这种自然的顺序将被很多分类器忽视。

本文提出了一种方法，可以利用这种自然的顺序来进行排序，实验结果表明，利用了自然的顺序可以得到更好的分类效果。

关键词：有序划分；排序；决策树

1 引言

深度神经网络目前在计算机视觉和自然语言处理占据主导地位，这类算法通常有很好的表现，是因为神经网络有极强的特征提取能力。因此，通过表示学习的方式，实现有效的数据特征变换，是促使一个学习任务取得成功的关键因素。深度神经网络的模型并不擅长处理表格数据，主要原因是，表格数据特征不均匀，数据样本量不够大，极值较大等特点。对于表格数据，采用传统机器学习的方法会得到较好的结果。而基于表示学习思想的数值嵌入已被公认为是一种更有效的技术路线。

决策树是一个非常经典，非常传统的机器学习分类方法。传统的决策树算法是一种多分类算法。决策树通常用来处理表格数据。决策树算法采用树形结构，使用层层推理来实现最终的分类。决策树由下面几种元素构成：根节点：包含样本的全集，内部节点：对应特征属性测试，叶节点：代表决策的结果。预测时，在树的内部节点处用某一属性值进行判断，根据判断结果决定进入哪个分支节点，直到到达叶节点处，得到分类结果。这是一种基于 if-then-else 规则的有监督学习算法，决策树的这些规则通过训练得到，而不是人工制定的。决策树是最简单的机器学习算法，它易于实现，可解释性强，完全符合人类的直观思维，有着广泛的应用。

决策树学习的 3 个步骤：特征选择，决策树生成，决策树剪枝。特征选择决定了使用哪些特征来做判断。在训练数据集中，每个样本的属性可能有很多个，不同属性的作用有大有小。因而特征选择的作用就是筛选出跟分类结果相关性较高的特征，也就是分类能力较强的特征。在特征选择中通常使用的准则是：信息增益。选择好特征后，就从根节点触发，对节点计算所有特征的信息增益，选择信息增益最大的特征作为节点特征，根据该特征的不同取值建立子节点；对每个子节点使用相同的方式生成新的子节点，直到信息增益很小或者没有特征可以选择为止。剪枝的主要目的是对抗“过拟合”，通过主动去掉部分分支来降低过拟合的风险。

决策树通常假设类别是无序的，并且决策树算法可以输出预测类别的概率，因此本文的底层分类器选择了决策树。

统计学家区分可以在属性中表示的四个基本量，通常称为度量级别[1]。有四种测量类型: 标称量、序数量、间隔量和比率量。标称量和序数之间的区别在于，后者在它们可以假定的不同值之间表现出一种顺序。例如，序数属性可以表示外部温度的粗略分类，这些温度由值 Hot、Mild 和 Cool 表示。很明显，这些值之间是有顺序的，我们可以写成 Hot > Mild > Cool。

本文提出的方法是针对有序问题的，也就是说，用于具有有序类别的分类任务。

2 相关工作

本文提出的方法实际上是一个将多分类问题转换成二分类问题的方法。多分类的问题的二元分解有很多种方法，本节将一一介绍这些方法。

2.1 二元分解策略

在无序分类中，常用的二进制分解策略有三种，分别是一对一 (OVO)、一对所有 (OVA) 和多对多 (MVM)。这三种方法可以整合到用同一种编码矩阵[2]表示:

矩阵维数:k × h，其中 k 为多类分类问题 T_k 中的类数，h 的取值取决于具体的分解策略，表示分解后生成的二元分类问题 T_2 的个数;

每一行表示一个特定类的编码向量;

每一列显示每个预定义的类要么不参与分类，要么被定义为学习特定分类器的正类或负类;

每个元素 ele_{ij} 的取值范围为 1，−1,0。如果 ele_{ij} = 1，表示 c_i 作为学习第 j 个分类器的正类; 如果 ele_{ij} =−1，表示 c_i 作为负类来学习第 j 个分类器; 如果 ele_{ij} = 0，表示 c_i 不参与第 j 个分类器的学习。

OVO: h = k(k−1)⁄2。类 c_i 属于 T_k 定义为正类，类 c_j 属于 T_k 定义为负类，在每个二元分类任务 T_2 中，其中 i≠j。编码矩阵如图 1所示。显然，与 OVA 策略相比，OVO 策略产生了更多需要训练的分类器，但训练每个二元分类器[3]只需要两类样本。因此，为了比较两种策略的计算效率，有必要同时考虑分类器的数量和训练每个分类器所需的样本数量。

$$\begin{pmatrix} & f_1 & f_2 & f_3 & f_4 & f_5 & f_6 \\ c_1 & +1 & +1 & +1 & 0 & 0 & 0 \\ c_2 & -1 & 0 & 0 & +1 & +1 & 0 \\ c_3 & 0 & -1 & 0 & -1 & 0 & +1 \\ c_4 & 0 & 0 & -1 & 0 & -1 & -1 \end{pmatrix}$$

图 1: OVO

OVA: h = k。T_2 中定义的正类是 T_k 中预定义的类 c_i，而负类包括除了 c_i 之外的其他类。编码矩阵如图 2所示。在 OVA 策略中，ele_{ij} 的取值范围为 1，−1，每个二叉分类器的训练都涉及到所有的类。与 OVO 策略相比，OVA 策略更容易造成类不平衡[4]的问题，使相应的二元分类任务更加困难，影响方法的有效性。

$$\begin{pmatrix} & f_1 & f_2 & f_3 & f_4 \\ c_1 & +1 & -1 & -1 & -1 \\ c_2 & -1 & +1 & -1 & -1 \\ c_3 & -1 & -1 & +1 & -1 \\ c_4 & -1 & -1 & -1 & +1 \end{pmatrix}$$

图 2: OVA

MVM: 该策略可以看作是 OVO 和 OVA 的推广。每次合并几个类作为正类，合并另外几个类作为负类。确定 MVM 最终输出的最经典方法是错误纠正输出代码 (ECOC), 编码矩阵如图 3所示。

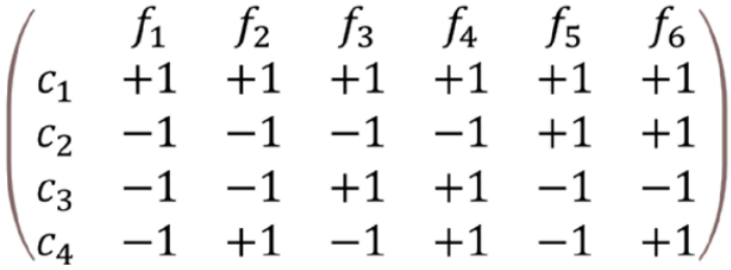

图 3: MVM

2.2 有序二元分解策略

与无顺序分类相比，序数分类的前提是类之间存在顺序关系。为了探索标签顺序信息对提高分类性能的有用性，在这些无序分解策略的基础上，提出了多种有序分解策略，如 one-vs-next[5]、one-vs-follower[6]、one-vs-previous[7]。编码矩阵如图 4图 5图 6所示。

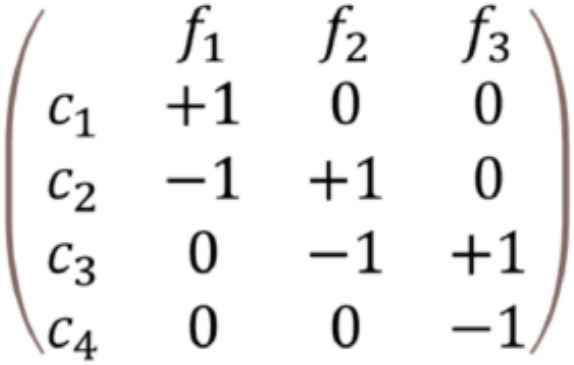

图 4: one-vs-next

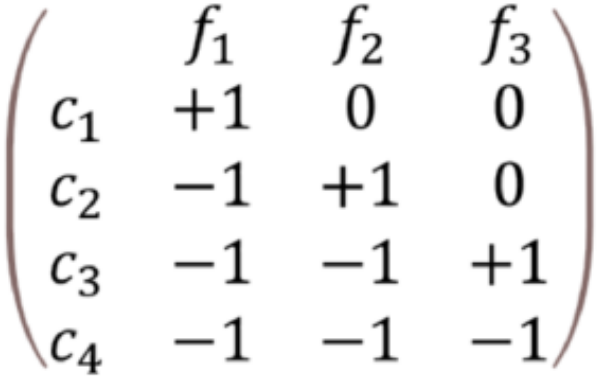

图 5: one-vs-followers

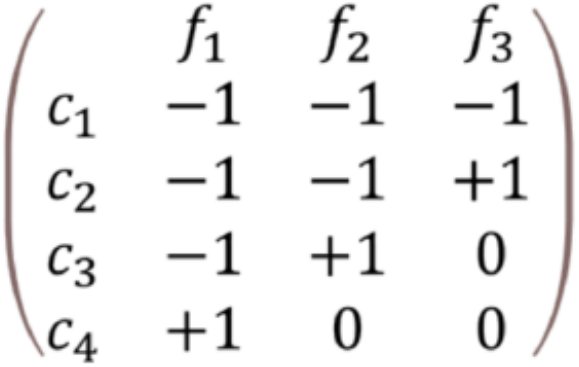

图 6: one-vs-previous

由于假设类之间存在顺序关系，因此每个类在序中的次序记为 r。

在 one-vs-next 策略的设定下，生成的二进制分类器的数量为 h = k−1。为学习第 j 个二分类器定义的正类为次序 r = j 的类，负类为次序 r = j + 1 的类。

在 one-vs-follower 策略的设置中，生成的二进制分类器的数量为 h = k−1。学习第 j 个二分类器定义的正类为次序 r = j 的类，负类由次序 r > j 的类合并得到。

在 one-vs-previous 策略的设置中，生成的二进制分类器的数量为 h = k−1。用于训练第 j 个分类器的正类定义为次序 r = k−j + 1 的类，负类由次序 r < k−j + 1 的类合并得到。

one-vs-next 策略可以看作是有序的 OVO 策略，one-vs-follower 和 one-vs-previous 策略都可以看作是有序的 OVR 策略。此外，通过上述任何一种有序分解策略进行二进制分解后，需要训练的二进制分类器的数量 h 为 k−1。

因此，有序分解的时间复杂度一般较低。

3 本文方法

3.1 本文方法概述

本文的方法其实就是 OrderedPartitions。在 OrderedPartitions 策略的设置中，生成的二进制分类器的数量为 h = k−1。训练第 j 个二分类器定义的正类由所有次序为 r≤j 的类合并而来，负类由所有秩为 r > j 的类合并而来。编码矩阵如下图 8所示。可以看出 OrderedPartitions 策略可以看作是一个有序的 mvm 策略。

$$\begin{pmatrix} & f_1 & f_2 & f_3 \\ c_1 & +1 & +1 & +1 \\ c_2 & -1 & +1 & +1 \\ c_3 & -1 & -1 & +1 \\ c_4 & -1 & -1 & -1 \end{pmatrix}$$

图 7: OrderedPartitions

本文提出的方法是对 k 类问题构造 k×(k-1)/2 个分类器。由于类别是有序的，我们可以将每个类别视为一个划分点，将连续的值划分开来。如图 8所示。

在每一次的预测中，会有一些不可见的类的实例。概率的计算方法如下公式所示。

$$Pr(V_1) = 1 - Pr(Target > V_1)$$

$$Pr(V_i) = Pr(Target > V_{i-1}) \times (1 - Pr(Target > V_i)), 1 < i < k$$

$$Pr(V_k) = Pr(Targe > V_{k-1})$$

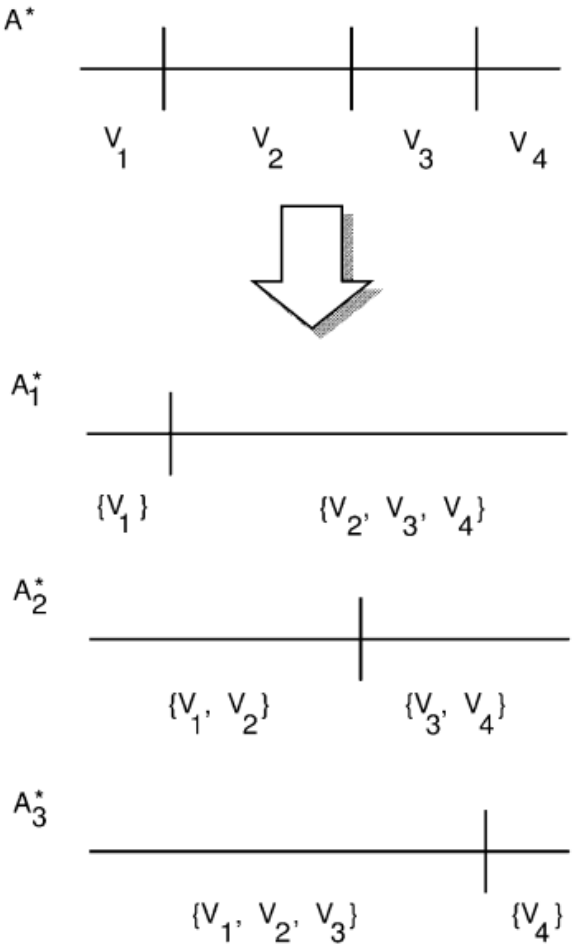

图 8: 方法示意图

3.2 数据集选取

本研究提出的算法是基于决策树的，因此数据集均为表格数据。由于本研究提出的算法是针对类别为有序数据的，但是并没有那么多类别有序的数据集。于是，作者选取了离散数据类型的数据集，来自己定义有序的类别。也就是说，把用于回归问题的数据集经过一些数据处理，用于分类问题。实验中选取的数据集与原文中的用到的数据集是一样的，但是原文中用到的某些数据集没有找到，实验中就没有使用了。

4 复现细节

首先，定义一个名为 ordered_partitions 的函数，接受两个参数：data 和 partitions。data 是需要进行分区的数据类别，partitions 是一个存储分区结果的列表。

```
def ordered_partitions(data, partitions=[]):
```

接下来，在函数内部，通过判断 data 是否为空，来决定是否继续进行分区：

```
 if not data:
        yield partitions
    else:
```

如果 data 为空，说明分区已经完成，可以通过 yield 语句返回最终的分区结果：

```
        yield partitions
```

如果 data 不为空，则需要继续进行分区。这时，需要通过一个循环来枚举所有可能的分区方案：

```
        for i in range(len(data)):
```

在每一次循环中，可以从 data 中选取一个数据项，并将其加入到当前的分区结果中：

```
            sub_data = data[:i] + data[i+1:]
            sub_partitions = partitions + [data[i]]
```

然后，使用递归算法继续对剩余的数据进行分区：

```
            yield from ordered_partitions(sub_data, sub_partitions)
```

可以使用以下代码调用 ordered_partitions 函数，并获得最终的分区结果：

```
data = [1, 2, 3, 4]
for partitions in ordered_partitions(data):
    print(partitions)
```

输出结果如下：

```
[[1], [2], [3], [4]]
[[1], [2], [3, 4]]
[[1], [2, 3], [4]]
[[1], [2, 3, 4]]
[[1, 2], [3], [4]]
[[1, 2], [3, 4]]
[[1, 2, 3], [4]]
[[1, 2, 3, 4]]
```

可以对生成的结果进行遍历，如果分区的大小等于 2，则将其分为两个类别；如果分区的大小不等于 2，则不进行任何操作。

使用了网格搜索的方式进行调参。其中，搜索了两个参数，决策树剪枝置信度门槛和每个叶子节点的最小实例数。每个叶子节点的最小实例数从 3 到 10，每次增加 1。剪枝置信度从 0.1 到 0.5，每次增加 0.5。经过了若干次实验后，综合实验的结果。最终将剪枝置信度确定为 0.25，将每个叶子结点的最小实例数确定为 0.25。

4.1 与已有开源代码对比

没有参考任何相关源代码。

4.2 实验环境搭建

实验使用了 Python 和 Java 两个编程语言。

其中 Python 语言主要用作数据处理。将连续的数值划分为离散的三组，如：高、中、低。先使用 Python 将数据集读入，然后进行一番处理，处理完之后再把数据用“.csv”的格式存在外存中。于是就可以用 Java 来读取处理好的数据集。

5 实验结果分析

实验中，基本分类器选择了 C4.5 决策树，决策树有一个超参数为是否剪枝，剪枝的决策树和未剪枝的决策树进行比较，通常，经过剪枝可以更好地防止过拟合，所以实验一开始，我就选择了剪枝的决策树。对于每个数据集，通过将目标值分别离散到 3、5 和 10 个区间来创建三个不同的版本。这样做是为了评估不同数量的类对方案的相对性能的影响。在比较中加入了 C4.5-1PC，以确定性能的总体差异是否由于将多类问题转化为几个两类问题而造成的。本研究要证明的是差异的原因是有序的转换：将多类问题转化为几个两类问题。

目标值离散到 3 个区间的实验结果如下图 9所示。

	C4.5-ORD	C4.5	C4.5-1PC
Abalone	62.96%	63.63%	63.45%
DeltaAilerons	63.18%	63.81%	62.73%
Delta Elevators	79.82%	80.12%	79.86%
Friedman Artificial	71.45%	71.43%	71.25%
Kinematics of Robot Arm	63.42%	63.54%	63.87%
Computer Activity (1)	75.03%	74.25%	75.24%
Computer Activity (2)	84.15%	83.61%	84.69%
Auto MPG	83.02%	83.92%	84.22%
Auto Price	82.20%	84.65%	85.22%
BostosHousing	93.42%	93.79%	93.08%
Diabetes	38.14%	52.09%	43.26%
Machine CPU	70.29%	72.68%	73.16%
Servo	69.88%	75.21%	70.72%
Wisconsin Breast Cancer	48.40%	41.24%	50.05%
Pumadyn Domain (1)	60.49%	61.33%	60.89%
Pumadyn Domain (2)	56.66%	56.01%	57.65%
Bank Domain (1)	75.01%	74.80%	74.94%
Bank Domain (2)	58.69%	59.03%	59.36%
California Housing	62.83%	63.52%	63.37%
Stocks Domain	89.35%	88.88%	89.75%

图 9: 目标值离散到 3 个区间的实验结果示意

原文中的实验结果如下图 10所示。

Dataset	C4.5-ORD	C4.5	C4.5-1PC
Abalone	66.07±0.26	63.90±0.24 •	65.91±0.34
Ailerons	75.37±0.31	74.78±0.37 •	73.79±0.34 •
Delta Ailerons	80.54±0.26	80.34±0.17	80.9 ±0.17 ○
Elevators	64.43±0.17	62.22±0.25 •	63.63±0.36 •
Delta Elevators	70.83±0.22	69.89±0.26 •	70.73±0.27
2D Planes	86.61±0.04	86.61±0.05	86.52±0.09 •
Pole Telecomm	95.90±0.12	95.64±0.10 •	95.58±0.1 •
Friedman Artificial	80.78±0.09	80.23±0.14 •	80.3 ±0.18 •
MV Artificial	99.51±0.02	99.53±0.02	99.55±0.03 ○
Kinematics of Robot Arm	64.37±0.37	63.76±0.24 •	64.88±0.38 ○
Computer Activity (1)	79.04±0.39	78.44±0.43 •	78.38±0.31 •
Computer Activity (2)	81.02±0.42	80.76±0.31	80.21±0.4 •
Census Domain (1)	70.53±0.17	69.58±0.27 •	70.95±0.18 ○
Census Domain (2)	69.53±0.21	68.19±0.28 •	69.61±0.23
Auto MPG	78.81±1.26	78.12±1.14	79.16±1.32
Auto Price	86.25±1.38	85.36±1.60	85.87±1.27
Boston Housing	75.52±1.07	74.83±1.72	74.79±1.18
Diabetes	54.45±2.61	51.00±3.91	54.45±2.61
Pyrimidines	56.89±3.98	50.11±2.95 •	55.12±3.31
Triazines	54.47±3.28	54.18±3.20	54.11±2.84
Machine CPU	73.88±2.39	71.85±2.44	74.26±2.77
Servo	77.24±1.16	75.56±1.26 •	79.22±1.27 ○
Wisconsin Breast Cancer	35.98±2.33	37.40±3.23	35.71±2.11
Pumadyn Domain (1)	66.71±0.37	66.02±0.28 •	67.37±0.32 ○
Pumadyn Domain (2)	78.83±0.15	77.65±0.41 •	77.64±0.3 •
Bank Domain (1)	85.89±0.28	86.02±0.28	85.61±0.11 •
Bank Domain (2)	57.15±0.45	55.84±0.38 •	53.94±0.26 •
California Housing	78.94±0.11	79.02±0.17	79.42±0.16 ○
Stocks Domain	91.74±0.53	91.24±0.53	91.77±0.45

○, • statistically significant improvement or degradation

图 10: 原文中目标值离散到 3 个区间的实验结果示意

该算法对于 Diabetes 数据集的准确率表现地很低。原因是这个数据集很小，只有一百多行数据。设定的超参数：叶子节点的最少实例树对于该数据集来说太大了，因此对于该数据集生成的决策树较小，达不到较为准确的分类效果。

实验一共用到了 20 个数据集。对于将目标值离散到 3 个区间的实验，在 4 个数据集上，C4.5-ORD 的表现比 C4.5-1PC 好。在 7 个数据集上，C4.5-ORD 的表现比 C4.5 好。

对于将目标值离散到 5 个区间的实验，在 16 个数据集上，C4.5-ORD 的表现比 C4.5-1PC 好。在 7 个数据集上，C4.5-ORD 的表现比 C4.5 好。

对于将目标值离散到 10 个区间的实验，在 18 个数据集上，C4.5-ORD 的表现比 C4.5-1PC 好。在 7 个数据集上，C4.5-ORD 的表现比 C4.5 好。

目标值离散到 5 个区间的实验结果如下图 11所示。

	C4.5-ORD	C4.5	C4.5-1PC
Abalone	45.77%	46.33%	39.81%
DeltaAilerons	47.85%	48.74%	46.09%
Delta Elevators	65.02%	65.60%	63.90%
Friedman Artificial	62.92%	62.06%	57.82%
Kinematics of Robot Arm	40.90%	40.51%	35.35%
Computer Activity (1)	63.06%	60.55%	61.27%
Computer Activity (2)	80.74%	80.35%	79.89%
Auto MPG	77.64%	80.25%	77.84%
Auto Price	57.23%	59.50%	57.36%
BostosHousing	88.04%	88.06%	86.23%
Diabetes	20.70%	31.63%	16.28%
Machine CPU	70.19%	65.02%	60.19%
Servo	56.41%	60.96%	47.25%
Wisconsin Breast Cancer	27.78%	31.75%	18.51%
Pumadyn Domain (1)	49.36%	49.42%	46.92%
Pumadyn Domain (2)	57.08%	56.93%	57.26%
Bank Domain (1)	69.09%	69.74%	70.20%
Bank Domain (2)	38.39%	38.99%	34.61%
California Housing	50.84%	51.68%	46.79%
Stocks Domain	80.49%	80.03%	79.43%

图 11: 目标值离散到 5 个区间的实验结果示意

总得来说，本实验结果和原文中的实验结果，所表现出来的准确率差距不大。但是本实验结果表现出的准确率较低于原文中的实验结果。

原文中的实验结果如下图 12所示。

Dataset	C4.5-ORD	C4.5	C4.5-1PC
Abalone	47.86±0.45	46.34±0.73 •	49.55±0.65 ○
Ailerons	59.24±0.30	56.97±0.35 •	55.58±0.34 •
Delta Ailerons	55.76±0.34	55.54±0.50	56.77±0.15 ○
Elevators	50.32±0.24	47.76±0.29 •	50.72±0.33 ○
Delta Elevators	49.78±0.31	47.63±0.42 •	50.34±0.29 ○
2D Planes	75.37±0.10	75.37±0.06	75.29±0.07
Pole Telecomm	95.05±0.12	95.05±0.10	94.94±0.07
Friedman Artificial	66.50±0.19	64.83±0.18 •	64.01±0.13 •
MV Artificial	99.19±0.04	99.20±0.04	99.19±0.02
Kinematics of Robot Arm	47.28±0.34	43.69±0.41 •	42.58±0.70 •
Computer Activity (1)	65.96±0.38	63.75±0.32 •	65.03±0.32 •
Computer Activity (2)	68.69±0.47	66.80±0.47 •	66.76±0.44 •
Census Domain (1)	53.32±0.22	50.20±0.36 •	51.46±0.34 •
Census Domain (2)	51.49±0.26	48.96±0.33 •	50.97±0.21 •
Auto MPG	59.74±0.98	59.58±1.86	59.46±1.25
Auto Price	66.80±2.20	62.39±2.71 •	63.50±1.43 •
Boston Housing	60.97±1.41	59.34±1.49 •	59.70±1.65
Diabetes	29.00±3.14	26.55±5.21	33.80±2.63 ○
Pyrimidines	43.27±2.85	42.27±3.51	42.68±2.78
Triazines	40.03±2.51	38.90±3.07	37.14±2.40 •
Machine CPU	58.10±1.32	56.78±2.78	56.62±2.43
Servo	52.45±1.60	55.16±2.09 ○	49.82±1.65 •
Wisconsin Breast Cancer	21.36±2.04	22.92±3.48	21.71±1.40
Pumadyn Domain (1)	49.83±0.30	46.04±0.43 •	48.28±0.34 •
Pumadyn Domain (2)	65.75±0.35	62.67±0.42 •	63.51±0.37 •
Bank Domain (1)	74.04±0.24	73.14±0.31 •	73.27±0.36 •
Bank Domain (2)	38.68±0.52	37.37±0.59 •	36.01±0.22 •
California Housing	64.83±0.24	63.30±0.18 •	64.36±0.18 •
Stocks Domain	86.85±0.67	87.05±0.88	85.19±0.53 •

○, • statistically significant improvement or degradation

图 12: 原文中目标值离散到 5 个区间的实验结果示意

目标值离散到 10 个区间的实验结果如下图 13所示。

	C4.5-ORD	C4.5	C4.5-1PC
Abalone	25.85%	27.78%	16.04%
DeltaAilerons	26.09%	29.61%	19.40%
Delta Elevators	65.57%	66.27%	63.29%
Friedman Artificial	31.58%	31.75%	21.41%
Kinematics of Robot Arm	15.71%	19.56%	10.23%
Computer Activity (1)	47.18%	43.92%	35.35%
Computer Activity (2)	74.12%	73.69%	71.86%
Auto MPG	72.39%	73.54%	60.58%
Auto Price	32.96%	31.13%	6.29%
BostosHousing	71.58%	74.07%	75.16%
Diabetes	0.47%	1.16%	3.26%
Machine CPU	52.78%	46.99%	27.66%
Servo	51.26%	56.77%	25.99%
Wisconsin Breast Cancer	18.87%	29.69%	10.31%
Pumadyn Domain (1)	26.45%	26.25%	17.05%
Pumadyn Domain (2)	44.76%	42.37%	27.68%
Bank Domain (1)	54.09%	55.63%	40.58%
Bank Domain (2)	43.84%	42.29%	35.92%
California Housing	33.19%	34.74%	22.39%
Stocks Domain	61.45%	61.92%	48.88%

图 13: 目标值离散到 10 个区间的实验结果示意

可以很明显地看出来，随着数据分区的数量变多，实验结果的准确率是有很明显的下降的。这一点也很好解释，分类中的不可见的类的概率是通过上文中的公式计算得出的。不可见的类越多，计算得出的误差也会增大。

原文中的实验结果如下图 14所示。

Dataset	C4.5-ORD	C4.5	C4.5-1PC
Abalone	29.48±0.36	26.68±0.61 •	27.43±0.58 •
Ailerone	39.36±0.33	36.64±0.37 •	35.76±0.32 •
Delta Ailerons	43.06±0.37	41.31±0.61 •	43.30±0.30
Elevators	31.41±0.34	28.58±0.35 •	29.77±0.38 •
Delta Elevators	39.86±0.33	36.9 ±0.44 •	41.44±0.23 ○
2D Planes	54.95±0.14	53.00±0.14 •	51.25±0.32 •
Pole Telecomm	91.18±0.08	90.85±0.14 •	89.34±0.15 •
Friedman Artificial	44.55±0.17	41.06±0.10 •	32.79±0.38 •
MV Artificial	98.11±0.03	98.17±0.05	97.36±0.06 •
Kinematics of Robot Arm	25.83±0.36	24.39±0.28 •	20.37±0.38 •
Computer Activity (1)	45.61±0.38	42.20±0.46 •	42.12±0.34 •
Computer Activity (2)	48.77±0.55	45.58±0.65 •	45.60±0.48 •
Census (1)	32.58±0.18	30.5 ±0.23 •	29.00±0.21 •
Census (2)	31.51±0.27	29.33±0.22 •	28.95±0.19 •
Auto MPG	36.55±1.18	39.63±1.70 ○	24.65±1.09 •
Auto Price	48.40±2.01	36.82±2.40 •	34.37±3.01 •
Boston Housing	42.66±1.00	38.61±1.25 •	35.93±1.65 •
Diabetes	14.55±3.12	23.00±3.12 ○	19.80±2.06 ○
Pyrimidines	19.39±2.95	23.89±2.69	15.93±1.83 •
Triazines	20.51±1.21	16.50±1.94 •	12.22±1.74 •
Machine CPU	36.35±2.47	36.23±1.48	30.59±1.93 •
Servo	34.57±0.98	34.60±1.47	13.18±0.03 •
Wisconsin Breast Cancer	10.73±1.91	11.24±3.15	11.33±1.24
Pumadyn Domain (1)	26.49±0.38	23.69±0.61 •	16.6 ±0.43 •
Pumadyn Domain (2)	45.76±0.44	41.87±0.42 •	41.21±0.54 •
Bank Domain (1)	52.76±0.37	49.57±0.44 •	43.58±0.82 •
Bank Domain (2)	25.51±0.42	24.20±0.33 •	24.06±0.39 •
California Housing	44.77±0.28	42.67±0.32 •	39.47±0.27 •
Stocks Domain	74.83±1.33	72.69±0.76 •	72.09±0.70 •

○, • statistically significant improvement or degradation

图 14: 原文中目标值离散到 5 个区间的实验结果示意

6 总结与展望

本实验证明，利用类别中排序信息可以得到更好的排序效果。

本实验是基于自带顺序的类别上。未来，希望可以将本算法运用到本身没有顺序的类别上。找到类别在特征空间中的存在的顺序，利用这种虚拟的排序，使用本算法，预期会得到更好的排序效果。

参考文献

[1] WITTEN I H, FRANK E, GELLER J. Data mining[J/OL]. ACM SIGMOD Record, 2002, 31: 76-77. https://dl.acm.org/doi/10.1145/507338.507355. DOI: 10.1145/507338.507355.

[2] EALLWEIN E L A, SCHAPIRE R E. Reducing Multiclass to Binary: A Unifying Approach for Margin

Classifiers Yoram Singer[J]. Journal of Machine Learning Research, 2000, 1: 113-141.

[3] HSU C W, LIN C J. A Comparison of Methods for Multiclass Support Vector Machines[J]. IEEE TRANSACTIONS ON NEURAL NETWORKS, 2002, 13: 415.

[4] WANG X, NIU Y. New one-versus-all ν-SVM solving intra−inter class imbalance with extended manifold regularization and localized relative maximum margin[J]. Neurocomputing, 2013, 115: 106-121. DOI: 10.1016/J.NEUCOM.2013.02.002.

[5] KWON Y S, HAN I, LEE K C. Ordinal Pairwise Partitioning (OPP) Approach to Neural Networks Training in Bond rating[J/OL]., 1997. https://onlinelibrary.wiley.com/doi/10.1002/. DOI: 10.1002/(SICI)1099-1174(199703)6:1.

[6] KIM K J, AHN H. A corporate credit rating model using multi-class support vector machines with an ordinal pairwise partitioning approach[J]. Computers & Operations Research, 2012, 39: 1800-1811. DOI: 10.1016/J.COR.2011.06.023.

[7] WU H, LU H, MA S. A practical SVM-based algorithm for ordinal regression in image retrieval[J/OL]. Proceedings of the ACM International Multimedia Conference and Exhibition, 2003: 612-621. https://dl.acm.org/doi/10.1145/957013.957144. DOI: 10.1145/957013.957144.