Multi-view Attribute Weighted Naive Bayes

Huan Zhang, Liangxiao Jiang, Wenjun Zhang, and Chaoqun Li

摘要

朴素贝叶斯 (Naive Bayes, NB) 由于其简单、高效和有效的特点，一直是十大数据挖掘算法之一。为了削弱其属性条件独立假设，提出了许多改进。然而，它们都只关注原始属性视图，难以反映实际应用中的所有数据特征。为了更全面地描述数据特征，在本研究中，我们从原始属性构建了两个标签视图，并提出了一个新的模型，称为多视图属性加权朴素贝叶斯 (MAWNB)。在 MAWNB 中，我们首先构建了多个超父单依赖估计器 (SPODEs) 和随机树 (RTs)，然后利用它们依次对每个训练实例进行分类，并使用它们所有的预测类标签构建两个标签视图。接下来，为了避免属性冗余，我们通过最小化每个视图中的负条件对数似然 (CLL) 来优化每个类的每个属性值的权重。最后，融合三个视图估计的类成员概率来预测每个测试实例的类标签。大量的实验表明，MAWNB 的性能明显优于 NB 和所有其他现有的最先进的竞争对手。

关键词：朴素贝叶斯；属性加权；多视图学习；分类

1 引言

近几十年来，贝叶斯网络分类器 (BNC)，由于其明显的优势，如显式的模型可解释性和强大的模型表达能力，一直备受关注。在众多 BNC 中，朴素贝叶斯 (Naive Bayes, NB) 是最简单但相当有效的分类器，它假设给定类的所有属性都是完全独立的。

尽管结构简单，但 NB 已经表现出了惊人的性能，并一直是十大数据挖掘算法之一。然而，它的属性条件独立性假设在实际应用中很少成立。为了减轻这种不切实际的假设，提出了许多增强功能，可以大致分为六类: 结构扩展，属性选择，属性加权，实例选择，实例加权和微调。

尽管这些增强功能已经展示出了出色的分类性能，但据我们所知，所有这些增强都只关注原始属性视图，即专家定义的属性视图。事实上，真实的分类应用程序通常非常复杂，原始属性视图很难反映所有的数据特征。为了更全面地描述数据特征，在本研究中提出了一个从原始属性构建两个标签视图的通用框架，然后提出了一个称为多视图属性加权朴素贝叶斯 (MAWNB) 的新模型。

在 MAWNB 中，我们首先构建了多个超父单依赖估计器 (SPODEs) 和随机树 (RTs)，然后利用它们依次对每个训练实例进行分类并使用它们的所有预测类标签来构造两个标签视图。其次，为了避免属性冗余，我们通过最小化负条件对数似然 (CLL) 来优化每个类的每个属性值的权重，并在每个视图中建立属性加权 NB 模型。最后，我们通过构建的三个模型来融合估计的类成员概率，以预测每个测试实例的类标签。据我们所知，这是第一个通过多视图学习来改善 NB 的研究。

我们对来自加州大学欧文分校 (UCI) 存储库的 60 个基准数据集进行了综合实验。MAWNB 在分类精度和 ROC 曲线下面积 (area under The ROC curve, AUC) 方面的性能明显优于 NB 和现有的所有先进竞争对手，同时消融研究表明，MAWNB 中的每个部分都是提高性能的必要条件。

2 相关工作

在过去的几十年里，结构扩展和属性加权是缓解属性条件独立假设的两种有代表性的方法。

2.1 结构扩展

结构扩展是一种直接而强大的方法，它通过添加一些定向弧[1]显式地表示属性依赖关系。对于这种方法，如何找到确切的属性依赖关系是至关重要的。为了解决这一问题，提出了树增强朴素贝叶斯(TAN)[2],[3]，其中属性依赖关系用树形结构表示。在 TAN 中，类节点直接指向所有属性节点，每个属性节点最多只能有一个来自另一个属性节点的父节点。事实上，TAN 对于学习属性依赖关系是非常有效的，但同时 TAN 会导致相当大的计算成本。

为了降低计算成本，Webb 等[4]提出了一种称为平均单依赖估计器 (AODE) 的集成模型。AODE 依次将每个属性视为所有其他属性的父节点，从而构建多个超父单依赖估计器 (SPODE)。然后通过直接平均所有符合条件的 SPODE 的类成员概率来产生最终的预测结果。

除 AODE 模型外，Jiang 等[5]还提出了另一种模型——隐性朴素贝叶斯 (hidden naive Bayes, HNB)。在 HNB 中，为每个属性创建一个隐藏父节点，并且隐藏父节点具有显式语义。粗略地说，每个属性的隐藏父属性可以看作是来自所有其他属性的影响的聚合。综上所述，HNB 避免了学习最优结构的高计算复杂度，但仍然考虑了所有属性的影响。

2.2 属性加权

虽然结构扩展是有效的，但学习最优结构仍然是相当困难的。近年来，许多专家关注的是另一种改进 NB 的方法，即 (通用) 属性加权。在这种方法中，每个属性根据其预测能力[6]在 0 到 1 之间分配一个连续的权重。现有的属性加权方法可以分为两大类: 过滤器和包装器。过滤器使用一般数据特征来学习属性权重，将其视为构建分类器之前的预处理步骤。包装器根据构建的分类器的性能反馈迭代优化属性权重。通常，包装器可能更准确，但很耗时。

2.2.1 过滤器

对于过滤器来说，如何计算每个属性的权重是最关键的问题。为了解决这一问题，Hall[7]提出了基于决策树的属性加权 NB (DTAWNB) 模型。在 DTAWNB 中，首先从随机样本的训练实例构建未修剪决策树的集合，然后为每个属性分配与其在决策树中的最小深度成反比的权重。

除此之外，Lee 等[6]提出了 Kullback-Leibler (KL) 基于度量的属性加权 NB (KLAWNB) 模型。在 KLAWNB 中，属性的权重是根据属性提供给类变量的全部信息估计的，这些信息是通过 KL 度量计算出来的。最近，Jiang 等[8]提出了一种基于相关性的特征 (属性) 加权 NB (CFWNB) 模型，该模型同时考虑了属性类相关性和属性与属性的冗余性。

2.2.2 包装器

对于包装器来说，如何寻找最优的权重向量是最重要的问题。为了解决这一问题，Zaidi 等[9]提出了一种减轻 NB 独立性假设的加权属性模型 (WANBIA(CLL))，该模型使用梯度下降搜索通过最小化负条件对数似然 (CLL) 来优化属性权重。随后，Jiang[10]将其扩展为特定于类的属性，为每个类区分地为每个属性分配一个特定的权重，并将它们的模型表示为 CAWNB(CLL)。最近，Zhang 等[11]进一步提出了另一种名为类特定属性值加权 NB (CAVWNB(CLL)) 的模型，将权重向量展开为权重矩阵，从而为每个类的每个属性值赋予一个特定的权重。

3 本文方法

从上面的描述我们可以得出结论，几乎所有现有的方法都只是关注如何使用原始属性视图来缓解 NB 中的属性条件独立假设。然而，大多数实际应用程序相当复杂，在这些场景中，类标签通常依赖于数据特征的许多不同方面。尽管原始属性视图中的每个手动提取的属性都可以从单个方面反映一些数据特征，但在实际应用程序中，手动提取所有所需的数据特征是非常困难的。

因此，提取一些高级属性来更全面地描绘数据特征是有意义和必要的。为此，在本研究中，我们提出了一种新的分类框架，即多视图属性加权。在该框架中，首先添加了一个多视图构造模块来从原始属性中生成一些高级属性，然后添加了一个多视图加权模块来避免每个视图中的属性冗余，最后添加了一个多视图融合模块来融合它们估计的类成员概率。在一般框架的基础上，我们进一步提出了一种新的多视图属性加权朴素贝叶斯 (MAWNB) 模型，具体描述如下。

3.1 MAWNB 总体框架

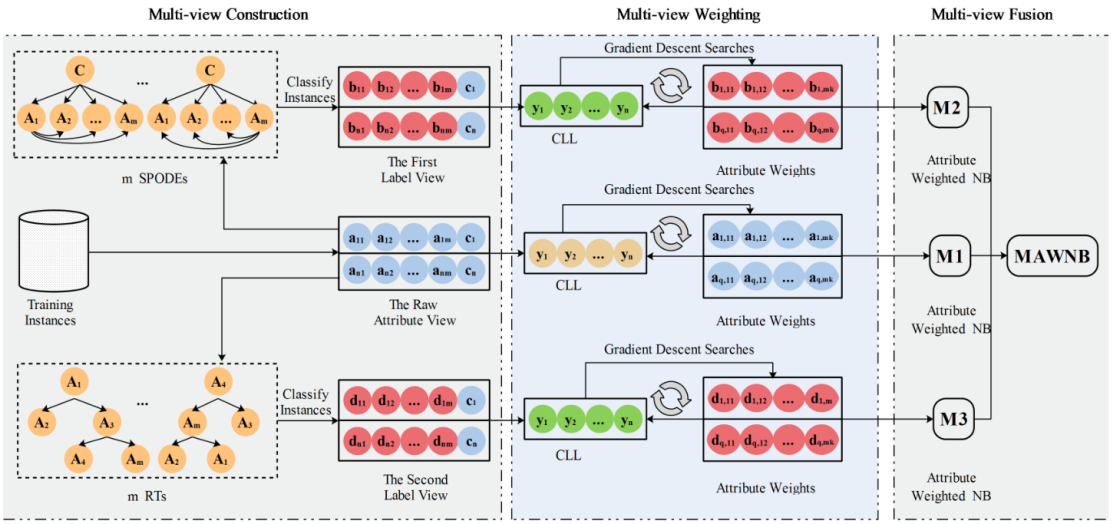

图 1: MAWNB 总体架构

MAWNB 的总体框架如图 1所示。从图 1 中我们不难发现，MAWNB 主要包括三个模块: 多视图构造、多视图加权和多视图融合。在多视图构造模块中，第 i 个实例的原始属性视图由 $\langle a_{i1}, a_{i2}, \cdots, a_{im}, c_i \rangle$ 表示。然后，利用原始属性视图分别构建 m 个超父单依赖估计器 (SPODEs) 和 m 个随机树 (RTs)。然后依次对每个训练实例进行分类，所有预测的类标签构造两个标签视图。对于第 i 个实例，两个标签视图可以用 $\langle b_{i1}, b_{i2}, \cdots, b_{im}, c_i \rangle$ 以及 $\langle d_{i1}, d_{i2}, \cdots, d_{im}, c_i \rangle$ 表示。在多视图加权模块中，利用梯度下降搜索，使负对数似然最小化，为每个类的每个属性值赋予特定的权重，然后建立三个属性加权 NB 模型，分别表示为 M1、M2 和 M3。在多视图融合模块中，融合三个模型估计的类成员概率，得到每个测试实例的预测类标签。在下面的小节中，我们将分别描述这三个模块的详细流程。

3.2 多视图构造

如上所述，我们的 MAWNB 由一个属性视图和两个标签视图组成。其中，属性视图只是由原始的属性组成，而两个标签视图是由自学习过程构建的。首先，如图 1左上角矩形所示，每个属性依次被视为所有其他属性的父节点，以建立由 **S** 表示的 m 个超父单依赖估计器（SPODE），然后利用每个建立的 SPODE 依次对每个训练实例进行分类，其所有预测的类标签构建第一个标签视图。具体来说，对于第 j 个 SPODE Sj，我们使用公式 (1) 来预测第 i 个训练实例的类标签，并使用公式 (2) 来构建第

一标签视图。

$$c_1^j(\mathbf{y_i}) = \arg\max_{c \in C} P(a_j, c) \prod_{r=1, r \neq j}^{m} P(a_r \mid a_j, c), \tag{1}$$

$$b_{ij} = c_1^j(\mathbf{y_i}), \tag{2}$$

其中 $c_1^j(\mathbf{y_i})$ 是 Sj 对第 i 个训练实例 $\mathbf{y_i}$ 的预测类标签，b_{ij} 是第一个标签视图中 $\mathbf{y_i}$ 的第 j 个构造属性值。$P(a_j, c)$ 为先验概率，$P(a_r \mid a_j, c)$ 为条件概率，分别由公式 (3) 和公式 (4) 估计。

$$P(a_j, c) = \frac{\sum_{i=1}^{n} \delta(a_{ij}, a_j)\, \delta(c_i, c) + 1}{n + n_j * q}, \tag{3}$$

$$P(a_r \mid a_j, c) = \frac{\sum_{i=1}^{n} \delta(a_{ir}, a_r)\, \delta(a_{ij}, a_j)\, \delta(c_i, c) + 1}{\sum_{i=1}^{n} \delta(a_{ij}, a_j)\, \delta(c_i, c) + n_r}, \tag{4}$$

其中 n_j, n_r 分别是属性 A_j 和 A_r 的值的数量。

现在，已经构造了第一个标签视图。由于每个 SPODE 都有不同的父节点，我们认为每个 SPODE 都可以从其独特的角度反映一些特定的数据特征。但是，SPODE 的基分类器仍然是 BNC，受集成学习的启发，基分类器应该是多样化和互补的。在一些数据集上，BNC 的分类性能没有决策树的分类性能那么好。因此，为了使基分类器具有多样性和互补性，同时更全面地描绘数据特征，我们进一步构建 m 棵随机树 (RTs)，用 **R** 表示，构建第二个标签视图，如图 1左下角矩形所示。

为了构建每个 RT，在每个分区中，我们首先从所有原始属性中随机选择 $\log_2 m$ 个属性，然后以信息增益作为选择度量递归地分割训练实例，定义如下:

$$\text{Gain}(T, A_j) = \text{Ent}(T) - \sum_{v=1}^{n_j} \frac{|T_v|}{|T|} \text{Ent}(T_v), \tag{5}$$

其中 A_j 是拆分属性，n_j 是 A_j 值的数量，$Ent(T)$ 称为熵，描述给定实例集的纯度，T_v 是父节点的第 v 个子节点上的实例的子集。

与第一个标签视图类似，在构建 m 个 RTs 后，利用它们依次对每个训练实例进行分类，构建第二个标签视图。具体来说，对于第 j 个 RT Rj，我们用公式 (6) 预测第 i 个训练实例的类标签，用公式 (7) 构造第二个标签视图。

$$c_2^j(\mathbf{y_i}) = \arg\max_{c \in C} \frac{\sum_{i=1}^{l} \delta(c_i, c)}{l}, \tag{6}$$

$$d_{ij} = c_2^j(\mathbf{y_i}), \tag{7}$$

其中 $c_2^j(\mathbf{y_i})$ 为 $\mathbf{y}$ 的预测类标签，l 为 $\mathbf{y}$ 所在叶节点中的训练实例数，d_{ij} 为 $\mathbf{y}$ 在第二个标签视图中构造的第 j 个属性值。

这样，我们构建了两个标签视图，我们认为它们可以更全面地描绘数据特征，主要基于以下三个

原因:1) 我们假设，如果两个实例被分类器分类到同一个类标签中，它们可能具有一些不容易从单个原始属性中找到的高级公共数据特征。因此，我们认为我们构建的单个标签属性比单个原始属性可以提供更有用的分类信息。2) 每个实例的标签视图是通过多个分类器的预测结果进行拼接得到的，即使单个分类器可能会出现一些分类偏差，总的标签视图仍然可以在一定程度上弥补单个分类器的分类偏差。换句话说，标签视图本身可以被视为高级属性的集合。因此，我们认为总标签视图可以从不同方面反映许多数据特征。3) 不同类型的基分类器本身也可以描绘不同的数据特征。因此，在我们的多视图构建模块中，我们选择了两种基分类器 (NB 和决策树) 来构建两种不同的标签视图。总之，多视图构建模块中的所有这些策略都保证了数据特征被尽可能完整地提取出来。

3.3 多视图加权

现在已经构造了三个视图来描述数据特征。一方面，属性越多意味着对数据的描述越全面，但另一方面，这三种视图都存在属性冗余的风险。为了避免属性冗余，在多视图加权模块中，通过最小化负 CLL 来优化每个类的每个属性值的权重，从而在每个视图中建立属性加权 NB 模型。在每个模型中，将每个属性值的权重纳入条件概率的指数部分，则测试实例 x 属于每个类的类隶属概率可由式 (8) 预测:

$$P(c \mid \mathbf{x}) = \frac{P(c) \prod_{j=1}^{m} P\left(a_{jk} \mid c\right)^{w_{c,jk}}}{\sum_{c' \in C} P\left(c'\right) \prod_{j=1}^{m} P\left(a_{jk} \mid c'\right)^{w_{c,jk}}}, \tag{8}$$

其中 $w_{c,jk}$ 表示特定类 c 的第 J 个属性的第 K 个值的权重。

使用特定于类的属性值加权而不是一般属性加权的原因如下:1) 对于原始属性视图，在许多现实应用中，每个类标签对每个属性值有不同的依赖关系。但一般的属性权重可能会使这些依赖产生偏差，因为不同的类别标签和不同的属性值具有相同的权重。2) 对于两种标签视图，这种偏差更加明显，因为在标签视图中，类标签自然与属性值高度相关，而不是与属性高度相关。而一般的属性权重完全忽略了这些差异，只是对不同的属性值给予相同的权重。综上所述，特定类别的属性值加权确实是必要的，而且比一般属性加权更可取。

下一个重要问题是如何学习特定于类的属性值权重。我们也使用 L-BFGS-M 优化程序学习每个类的每个属性值的权重，并使用 $l2$ 正则化来避免过拟合。对于目标函数，在本研究中，我们只选择最小化负 CLL，但是当我们改变目标函数以最小化 MSE 时，我们也可以得到类似的结论。正如我们从图 1中间矩形的多视图权重模块中看到的，所有三个视图都利用梯度下降搜索来学习它们的最佳权重。为了简单起见，这里我们只是在原始属性视图中描述了详细的优化过程，两个标签视图中的优化过程是相似的。首先，我们将每个类 $w_{c,jk}$ 的每个属性值的权重初始化为 1.0，然后利用梯度下降搜索通过最小化负 CLL 迭代更新特定于类的属性值的权重。目标函数由式 (9) 定义。

$$-CLL(\mathbf{w}) = -\log P(C \mid T, \mathbf{w}) + \lambda \left\| \mathbf{w} - \mathbf{w}_{\text{one}} \right\|^2 \tag{9}$$

$$= -\sum_{i=1}^{n} \log P\left(c_i \mid \mathbf{y_i}, \mathbf{w}\right) + \lambda \left\| \mathbf{w} - \mathbf{w_{one}} \right\|^2 \tag{10}$$

$$= -\sum_{i=1}^{n} \log \frac{P\left(c, \mathbf{y_i}; \mathbf{w}\right)}{\sum_{c'} \left(P\left(c', \mathbf{y_i}; \mathbf{w}\right)\right)} + \lambda \left\| \mathbf{w} - \mathbf{w}_{\text{one}} \right\|^2, \tag{11}$$

其中 w 表示属性权向量，w_{one} 也是一个与 w 大小相同且元素都等于 1.0 的矩阵， 是一个超参数，设为 1.0。

3.4 多视图融合

现在，我们基于原始属性视图和两个标签视图建立了三个属性加权 NB 模型 M1、M2 和 M3。由于三种模型从不同的角度描述数据特征，在多视图融合阶段，我们直接对其估计的类隶属概率进行平均，并将每个测试实例的预测类标签定义为类隶属概率最大的类，其表示方式如下:

$$c(\mathbf{x}) = \arg\max_{c \in C} \left(P(c \mid \mathbf{x})_{M1} + P(c \mid \mathbf{x})_{M2} + P(c \mid \mathbf{x})_{M3}\right). \tag{12}$$

总之，我们的 MAWNB 的整个学习算法可以分为训练 (MAWNB-Training) 和分类 (MAWNBClassification) 算法。

4 复现细节

4.1 与已有开源代码对比

本文的复现工作没有参考任何相关源代码，在此明确申明。

4.2 算法伪代码

Procedure 1 MAWNB-Training(T)

Input: Training dataset

Output: **S**-the built m SPODEs, **R**-the built m RTs, $M1$-built model by the raw attribute view, $M2$-the built model by the first label view, $M3$-the built model by the second label view

for each attribute j(j = 1, 2, $\bullet\bullet\bullet$, m) **do**
  Build a SPODE it S_j by regarding the jth attribute as the parent node
end
for each attribute j(j = 1, 2, $\bullet\bullet\bullet$, m) **do**
  Selecting $log_2 m$ attributes from the raw attribute view randomly
  Build a RT R_j using the information gain selection measure
end
for each instance $\mathbf{y_i}$(i = 1,2,...,n) **do**
  for each SPODE model S_j(j = 1,2,...,m) **do**
    Predict the attribute value $b_i j$ of $\mathbf{y_i}$ in the first label view using S_j by NB
  end
end
for each instance $\mathbf{y_i}$(i = 1,2,...,n) **do**
  for each RT model R_j(j = 1,2,...,m) **do**
    Predict the attribute value $d_i j$ of $\mathbf{y_i}$ in the second label view using R_j by NB
  end
end
Build three attribute weighted NB models M1,M2 and M3 by L-BFGS-M
return S, **R**, $M1$, $M2$, $M3$

Procedure 2 MAWNB-Classification(**x**, **S**, **R**, $M1$, $M2$, $M3$)

Input: Test instance, **S**-the built m SPODEs, **R**-the built m RTs, $M1$-the built model by the raw attribute view, $M2$-the built model by the first label view, $M3$-the built model by the second label view

Output: c(**x**)-the predicted class label of **x**

for each SPODE model $S_j(j = 1,2,...,m)$ **do**

| Predict the attribute value b_j of **x** in the first label view using S_j by NB

end

for each RT model $R_j(j = 1,2,...,m)$ **do**

| Predict the attribute value d_j of **x** in the second label view using R_j by NB

end

Use the built $M1$, $M2$ and $M3$ to estimate the class-membership probabilities of **x** respectively

Fuse the estimated class-membership probabilities of three built models by Eq.12

return c(**x**)

4.3 关键代码展示

我们把数据集中的数据经过差值填补后进行离散化处理，将处理好的数据输入我们的程序。先将离散化的数据转化为不小于 0 的整数，然后对数据进行标准化，再分出测试集与训练集。再将训练集输入给我们的模型 MAWNB。

图 2为实现原始属性的代码。我们把原始的数据用朴素贝叶斯分类器分类，再采用 L-BFGS-M 算法得到每个属性值对应的权重，并保存下来。

```
def raw_view(self, x_train, y_train, k):
    x1 = pd.DataFrame(x_train, columns=self.features)
    y1 = pd.DataFrame(y_train, columns=['class'])
    self.nb_m1 = Naive_Bayes()
    self.nb_m1.nb_fit(x1, y1)
    m = x_train.shape[1]
    x_train = self.nb_m1.pred(x1)
    mlp = neural_network.MLPClassifier(hidden_layer_sizes=(m * k), solver='lbfgs')

    mlp.fit(x_train, y_train)

    return mlp.coefs_[0]
```

图 2: 实现原始属性视图

图 3为实现第一个标签视图的代码。在使用 SPODE 做分类的时候，我发现使用 SPODE 算法的效率太低了，于是我尝试采用了 BernoulliNB 算法。BernoulliNB 算法的时间复杂度较低，使得整个算法的效率提高。BernoulliNB 也是一个基于 NBC 的分类器，最终跑出来的实验效果与 SPODE 差不多。

```
def first_view(self, x_train, y_train, k):
    m = x_train.shape[1]
    x = []
    self.S = []
    for i in range(m):
        clf = BernoulliNB()#由于SPODE运行速度太慢，这里改成了BernoulliNB
        #BernoulliNB也是一个基于NBC的分类器，实验效果与SPODE差不多
        clf.fit(x_train,y_train)
        t = clf.predict(x_train)
        t = np.array(t)

        x.append(t)
        self.S.append(clf)
    x_first = x[0].reshape(-1, 1)

    for i in range(1, m):
        x_first = np.concatenate([x_first, x[i].reshape(-1, 1)], 1)


    x1 = pd.DataFrame(x_first, columns=self.features)
    y1 = pd.DataFrame(y_train, columns=['class'])
    self.nb_m2 = Naive_Bayes()
    self.nb_m2.nb_fit(x1, y1)
    x_first = self.nb_m2.pred(x1)

    mlp = neural_network.MLPClassifier(hidden_layer_sizes=(m * k), solver='lbfgs') #max_iter=100
    mlp.fit(x_first, y_train)
    return mlp.coefs_[0]
```

图 3: 实现第一个标签视图

图 4为实现第二个标签视图的代码。在这里原论文中对于 RTs 的描述是在所有原始属性中随机选取其中的 $log_2 m$ 个属性，但是在实验中我发现，随机选取了 $log_2 m$ 个属性后对于实验结果并没有任何增益，甚至在有些数据集中会导致预测的准确率下降。因此在实验中我们没有对这个参数进行设置，而是直接选取了该参数的默认值。

```
def second_view(self, x_train, y_train, k):
    x = []
    m = x_train.shape[1]
    self.R = []
    for i in range(m):
        clf = RandomForestClassifier()#max_features='log2'
        clf.fit(x_train, y_train)
        rt_pred = clf.predict(x_train)
        self.R.append(clf)
        x.append(rt_pred)
    x_second = x[0].reshape(-1, 1)
    for i in range(1, m):
        x_second = np.concatenate([x_second, x[i].reshape(-1, 1)], 1)

    self.nb_m3 = Naive_Bayes()
    x2 = pd.DataFrame(x_second, columns=self.features)
    y2 = pd.DataFrame(y_train, columns=['class'])
    self.nb_m3.nb_fit(x2, y2)

    x_second = self.nb_m3.pred(x2)
    mlp = neural_network.MLPClassifier(hidden_layer_sizes=(m * k), solver='lbfgs') #max_iter=100
    mlp.fit(x_second, y_train)

    return mlp.coefs_[0]
```

图 4: 实现第二个标签视图

4.4 创新点

近年来，许多专家关注的是另一种改进 NB 的方法，即属性加权。在这种方法中，每个属性根据其预测能力在 0 到 1 之间分配一个连续的权重。目前最先进的贝叶斯算法，例如 CAWNB(CLL)，WANBIA(CLL)，CFWNB，DTAWNB 和 KLAWNB 都是通过属性加权来减轻属性条件独立假设带来的限制。而我们的 MA 截至目前为止，这是第一个通过多视图学习来改善 NB 的研究。该论文从一个全新的角度去改进贝叶斯分类器，并取得了很好的成果。

5 实验结果分析

本部分将会对实验结果进行分析，对实验内容进行说明，对实验结果进行分析描述。

5.1 实验数据集

我们首先在 Waikato Environment for Knowledge Analysis (WEKA) 平台中使用无监督属性过滤器 ReplaceMissingValues 替换所有缺失的属性值。然后应用 WEKA 平台中的无监督过滤器 Discrealize 将数值属性离散为标称属性。以 iris 数据集为例，处理之前如图 5所示：

sepal length	sepal width	petal length	petal width	class
5.1	3.5	1.4	0.2	Iris-setosa
4.9	3	1.4	0.2	Iris-setosa
4.7	3.2	1.3	0.2	Iris-setosa
4.6	3.1	1.5	0.2	Iris-setosa
5	3.6	1.4	0.2	Iris-setosa
5.4	3.9	1.7	0.4	Iris-setosa
4.6	3.4	1.4	0.3	Iris-setosa
5	3.4	1.5	0.2	Iris-setosa
4.4	2.9	1.4	0.2	Iris-setosa
4.9	3.1	1.5	0.1	Iris-setosa
5.4	3.7	1.5	0.2	Iris-setosa
4.8	3.4	1.6	0.2	Iris-setosa
4.8	3	1.4	0.1	Iris-setosa
4.3	3	1.1	0.1	Iris-setosa
5.8	4	1.2	0.2	Iris-setosa
5.7	4.4	1.5	0.4	Iris-setosa
5.4	3.9	1.3	0.4	Iris-setosa
5.1	3.5	1.4	0.3	Iris-setosa
5.7	3.8	1.7	0.3	Iris-setosa
5.1	3.8	1.5	0.3	Iris-setosa
5.4	3.4	1.7	0.2	Iris-setosa
5.1	3.7	1.5	0.4	Iris-setosa

图 5: 原始的 iris 数据集

经过 WEKA 平台处理之后，我们的数据集如图 6所示：

'sepal length'	'sepal width'	'petal length'	'petal width'	class
'\'(5.02-5.38]\"	'\'(3.44-3.68]\"	'\'(-inf-1.59]\"	'\'(-inf-0.34]\"	Iris-setosa
'\'(4.66-5.02]\"	'\'(2.96-3.2]\"	'\'(-inf-1.59]\"	'\'(-inf-0.34]\"	Iris-setosa
'\'(4.66-5.02]\"	'\'(2.96-3.2]\"	'\'(-inf-1.59]\"	'\'(-inf-0.34]\"	Iris-setosa
'\'(-inf-4.66]\"	'\'(2.96-3.2]\"	'\'(-inf-1.59]\"	'\'(-inf-0.34]\"	Iris-setosa
'\'(4.66-5.02]\"	'\'(3.44-3.68]\"	'\'(-inf-1.59]\"	'\'(-inf-0.34]\"	Iris-setosa
'\'(5.38-5.74]\"	'\'(3.68-3.92]\"	'\'(1.59-2.18]\"	'\'(0.34-0.58]\"	Iris-setosa
'\'(-inf-4.66]\"	'\'(3.2-3.44]\"	'\'(-inf-1.59]\"	'\'(-inf-0.34]\"	Iris-setosa
'\'(4.66-5.02]\"	'\'(3.2-3.44]\"	'\'(-inf-1.59]\"	'\'(-inf-0.34]\"	Iris-setosa
'\'(-inf-4.66]\"	'\'(2.72-2.96]\"	'\'(-inf-1.59]\"	'\'(-inf-0.34]\"	Iris-setosa
'\'(4.66-5.02]\"	'\'(2.96-3.2]\"	'\'(-inf-1.59]\"	'\'(-inf-0.34]\"	Iris-setosa
'\'(5.38-5.74]\"	'\'(3.68-3.92]\"	'\'(-inf-1.59]\"	'\'(-inf-0.34]\"	Iris-setosa
'\'(4.66-5.02]\"	'\'(3.2-3.44]\"	'\'(1.59-2.18]\"	'\'(-inf-0.34]\"	Iris-setosa
'\'(4.66-5.02]\"	'\'(2.96-3.2]\"	'\'(-inf-1.59]\"	'\'(-inf-0.34]\"	Iris-setosa
'\'(-inf-4.66]\"	'\'(2.96-3.2]\"	'\'(-inf-1.59]\"	'\'(-inf-0.34]\"	Iris-setosa
'\'(5.74-6.1]\"	'\'(3.92-4.16]\"	'\'(-inf-1.59]\"	'\'(-inf-0.34]\"	Iris-setosa
'\'(5.38-5.74]\"	'\'(4.16-inf)\"	'\'(-inf-1.59]\"	'\'(0.34-0.58]\"	Iris-setosa
'\'(5.38-5.74]\"	'\'(3.68-3.92]\"	'\'(-inf-1.59]\"	'\'(0.34-0.58]\"	Iris-setosa
'\'(5.02-5.38]\"	'\'(3.44-3.68]\"	'\'(-inf-1.59]\"	'\'(-inf-0.34]\"	Iris-setosa
'\'(5.38-5.74]\"	'\'(3.68-3.92]\"	'\'(1.59-2.18]\"	'\'(-inf-0.34]\"	Iris-setosa
'\'(5.02-5.38]\"	'\'(3.68-3.92]\"	'\'(-inf-1.59]\"	'\'(-inf-0.34]\"	Iris-setosa
'\'(5.38-5.74]\"	'\'(3.2-3.44]\"	'\'(1.59-2.18]\"	'\'(-inf-0.34]\"	Iris-setosa
'\'(5.02-5.38]\"	'\'(3.68-3.92]\"	'\'(-inf-1.59]\"	'\'(0.34-0.58]\"	Iris-setosa

图 6: 处理后的 iris 数据集

5.2 实验结果

在本次实验中，我们在三个数据集上进行了对比实验，我们将 MAWNB 与其相关的竞争对手——标准朴素贝叶斯 the standard Naive Bayes(NB) 和随机森林 Random Forest(RF) 进行了比较。实验结果如表 1所示：

表 1: MAWNB 与 RF 和 NB 的分类精度比较

Dataset	MAWNB	RF	NB	论文
iris	95.56%	95.33%	94.67%	95.40%
kr-vs-kp	98.44%	99.28%	95.57%	98.45%
agaricus-lepiota	84.91%	99.28%	87.89%	91.87%

从实验结果可以看出，我复现的 MAWNB 在一些数据集上表现良好，如 iris 数据集，预测的准确率超过了 RF 和 NB，并略微超过原论文中的数据；但是在另外一些数据集上就表现一般，如 agaricus-lepiota 数据集，在该数据集上预测的准确率较低，低于 RF 和 NB，并且也低于原论文中对于该数据集的预测准确率；在 kr-vs-kp 数据上，我的复现的结果高于 NB，虽然低于 RF，但是与原论文中给出的数据基本持平。

总的来说，我复现的 MAWNB 预测结果表现良好，虽然不及原论文的结果优秀，但是也做到了接近原论文的数据。在与其他算法的准确率进行对比的时候，也基本做到了与原论文给出的结果一致。

6 总结与展望

论文针对原始属性视图难以反映所有数据特征的问题，提出了一种新的多视图属性加权朴素贝叶斯模型 (MAWNB)，该模型首先构造两个标签视图，然后优化每个视图中每个类的每个属性值的权重，最后融合估计的类隶属概率来预测每个测试实例的类标签。这项工作提供了一个新颖且有吸引力的模型，可以应用于广泛的现实应用，例如新冠肺炎诊断和缺陷分类。

在复现的过程中我也遇到了很多问题，比如在选取函数参数的时候，需要反复运行代码进行试验，才能确定一个比较好的参数的取值。在对照论文进行复现的时候，也遇到了很多细节上的问题，我就按照原文所说结合自己的理解和所学知识，进行复现。在一些函数的选取上，也遇到了很多问题，我反复查找资料学习。由于所学知识以及经验有限，我的算法还不完善，存在一些不足，比如实验结果不稳定，一些细节没有复现到位等问题。在复现写代码的过程中，我获益良多，学习到了很多知识提高了动手能力和自学能力，也学习到了很多算法和机器学习的方法。

此外，在复现的过程中，我发现生成 SPOED 视图和多视图加权模块的时间复杂度较高，也许可以使用过滤器来降低多视图加权模块的时间复杂度。对于最后的融合模块，如果对每个视图的预测结果施加以对应的权重再融合，也许最后的预测效果会更好。这两点或许可以作为未来进一步的研究方向。

参考文献

[1] DUAN Z, WANG L, CHEN S, et al. Instance-based weighting filter for superparent one-dependence estimators[J]. Knowledge-Based Systems, 2020, 203: 106085.

[2] FRIEDMAN N, GEIGER D, GOLDSZMIDT M. Bayesian network classifiers[J]. Machine learning, 1997, 29(2): 131-163.

[3] KEOGH E J, PAZZANI M J. Learning augmented Bayesian classifiers: A comparison of distribution-based and classification-based approaches.[C]//AIStats. 1999.

[4] WEBB G I, BOUGHTON J R, WANG Z. Not so naive Bayes: aggregating one-dependence estimators [J]. Machine learning, 2005, 58(1): 5-24.

[5] JIANG L, ZHANG H, CAI Z. A novel bayes model: Hidden naive bayes[J]. IEEE Transactions on knowledge and data engineering, 2008, 21(10): 1361-1371.

[6] LEE C H, GUTIERREZ F, DOU D. Calculating feature weights in naive bayes with kullback-leibler measure[C]//2011 IEEE 11th International Conference on data mining. 2011: 1146-1151.

[7] HALL M. A decision tree-based attribute weighting filter for naive Bayes[C]//International conference on innovative techniques and applications of artificial intelligence. 2007: 59-70.

[8] JIANG L, ZHANG L, LI C, et al. A correlation-based feature weighting filter for naive bayes[J]. IEEE transactions on knowledge and data engineering, 2018, 31(2): 201-213.

[9] ZAIDI N A, CERQUIDES J, CARMAN M J, et al. Alleviating naive Bayes attribute independence assumption by attribute weighting[J]., 2013.

[10] JIANG L, ZHANG L, YU L, et al. Class-specific attribute weighted naive Bayes[J]. Pattern recognition, 2019, 88: 321-330.

[11] ZHANG H, JIANG L, YU L. Class-specific attribute value weighting for Naive Bayes[J]. Information Sciences, 2020, 508: 260-274.