**韩淑婷周报**

目录

[一 第一周2019年9月29日 2](#_Toc26740160)

[1 论文研究 2](#_Toc26740161)

[2 Linux学习 3](#_Toc26740162)

[二、第二周2019年10月6日 5](#_Toc26740163)

[1 论文研究 5](#_Toc26740164)

[三、第三周2019年10月13日 7](#_Toc26740165)

[1 论文研究 7](#_Toc26740166)

[四、第四周2019年10月20日 9](#_Toc26740167)

[1 实验进展 9](#_Toc26740168)

[五、第五周2019年10月27日 10](#_Toc26740169)

[1 实验方面 10](#_Toc26740170)

[1 其他方面 11](#_Toc26740171)

[六、第六周2019年11月3日 12](#_Toc26740172)

[1 实验方面 12](#_Toc26740173)

[2 论文方面 12](#_Toc26740174)

[3 其他方面 13](#_Toc26740175)

[七、第七周2019年11月10日 14](#_Toc26740176)

[1讨论班前和后 14](#_Toc26740177)

[2 读书和准备考试 14](#_Toc26740178)

[3 论文方面 14](#_Toc26740179)

[八、第八周2019年11月17日 15](#_Toc26740180)

[1实验数据 15](#_Toc26740181)

[2 其他方面 16](#_Toc26740182)

[九、第九周2019年11月24日 17](#_Toc26740183)

[1 实验方面 17](#_Toc26740184)

[十、第十周2019年12月1日 19](#_Toc26740185)

[1 实验方面 19](#_Toc26740186)

[十一、第十一周2019年12月8日 21](#_Toc26740187)

[1 算法 21](#_Toc26740188)

[2 测试 22](#_Toc26740189)

### 一 第一周2019年9月29日

## 1 论文研究

**1.1 文献翻译：**

Assessing Heuristic Machine Learning Explanations with Model Counting

用模型计数评估启发式机器学习解释

摘要

机器学习（ML）模型广泛应用于金融、医学、教育等领域的决策过程。在这些领域，ML的结果对人类有直接的影响，比如，用ML评估一个人是否能获得贷款或者一个人是否能从监狱释放。然而我们不能盲目地依赖黑盒ML模型，我们需要去解释ML做出的决策。这推动了一系列ML解释系统的发展，其中包括LIME和它的后继者ANCHOR。由于现有的工具创造出来的机器学习解释都具有启发性，因此有必要对其进行验证。我们提出了一种基于SAT的方法来评估ANCHOR产生的解释的质量。我们将一个训练过的ML模型和一个给定预测的解释进行编码并用作命题公式。然后通过使用最新的近似模型计数器，我们用求出的解的个数来评估所提供的解释的质量。

介绍

近些年ML的发展在很大程度上说明了AI的影响和社会意义。ML的应用领域快速扩展，包括安全至上的一些设置（如智能汽车）和直接影响人类的一些领域（如经济、医学、教育和司法）。提高ML模型置信度的工作不仅包括验证模型属性还包括在ML模型的操作不能被人类直观解释的情况下，解释模型的预测。可解释的AI（XAI）的一般领域既针对自然可解释的ML模型（包括解释树和解释集）的发展，也针对ML模型被视为黑盒的环境（例如神经网络）中的解释计算、分类器的集合，等等。

最著名的XAI方法是基于启发式的，并在特征值空间没有被完全分析的意义上提供了所谓的局部解释。在最近提出的许多计算局部解释的方法中，LIME和它的后继者ANCHOR事两个成功的实例。然而，由于LIME和ANCHOR都是基于启发式的，那么一个自然的问题就产生了：基于启发式的解释在实践中有多精准？本文重点关注ANCHOR（因为它相对于LIME有改进），并提出了一种新颖的方法来评估用于计算局部解释的启发式方法的质量。对于每个计算出的解释，ANCHOR给出一种质量的度量，或者称为解释的估计精度，即解释适用及预测匹配的实例的百分比。从ML模型的编码、ANCHOR计算得到的解释以及目标预测开始，本文建议使用（近似）模型计数来评估解释的实际精度。具体来说，本文考虑了二值化神经网络（BNN），利用了最近提出的针对BNN的命题编码，并在知名数据集上对ANCHOR的质量进行了评估。正如我们所展示的，ANCHOR的解释质量可能有很大的不同，这表明对有些数据集来说，ANCHOR的解释相当准确，但对于另一些数据集来说，ANCHOR的解释可能相当不准确。我们得到的有点出乎意料的实验结论表明XAI需要更多更正式的研究。

**1.2 个人总结**

由于ML模型是黑盒，结果是否是有意义有待解释，因此产生了ML解释系统。针对ML解释系统质量的高低，本文提出一种基于用模型计数的方法来评估。本实验利用BNN对ANCHOR进行评估，得到的实验结果是：ANCHOR的表现不稳定，对于某些数据集来说，ANCHOR得到的解释很精确，对于另一些数据集而言，ANCHOR得到的解释不精确。这代表XAI领域还需要更多更深入的研究。

**1.3 ANCHOR解读**

Anchors: High-precision model-agnostic explanations

ANCHOR：高精度模型解释器

文中介绍了一种新颖的模型无关的解释复杂机器学习模型的系统，该系统使用高精度的规则进行解释，称它为Anchors。Anchors使用局部的充分条件来解释模型行为。首先，设计了算法来有效地生成这样的规则；然后设计了多组实验，针对各种复杂模型和不同领域，来验证Anchors的可扩展性；最后，通过user study来说明，Anchors能允许用户预测模型的行为，并且用户的预测精确度比通过其他解释模型或无解释模型的情况要高。

复杂的机器学习模型确实带来了高准确率，但是也使得模型对于用户来说是一个黑盒，而用户对理解模型行为的需求越来越关注，使得可解释的机器学习开始盛行。可解释的机器学习分为全局可解释的模型和局部可解释的模型。全局可解释的模型一般是特别设计的，局部可解释模型一般是与模型无关的。

**1.4 BNN解读**

Binarized Neural Networks: Training Neural Networks with Weights and Activations Constrained to +1 or −1

二值化神经网络：训练权重和激活度限制为+1或-1的神经网络

通过研究论文对BNN（二值化神经网络）进行理解。论文中提出一种方法训练BNN，在运行时权重和激活值均二值化，训练时进行梯度计算，用Torch7 和 Theano框架做了实验，证明了可以在MNIST, CIFAR-10 和SVHN数据集上训练BNN，效果不错。文中还证明了在前向传播过程，BNN极大地减少了内存消耗（大小和访问次数），用位移操作代替多数计算操作。而且二值化CNN导致卷积核的重复。文献认为在专用的硬件上可以减少60%的时间复杂度。最后，文中写了一个二进制矩阵乘法GPU核，预计在运行MNISTBNN时比未经优化的核快七倍，并且保证其分类精度。决定式的二值化有两种方法。第一种是：正数是1，负数是-1。第二种是：随机式的二值化，对x计算一个概率p，当p大于一个阙值（计算机随机产生）时为+1，否则为-1。第二种方法虽然看起来比第一种更合理，但是在实现时却有一个问题，那就是每次生成随机数会非常耗时，所以除了在文献试验中训练时的一些激活值一般使用第一种方法。

## 2 Linux学习

我参考《鸟哥的Linux私房菜》一书对Linux系统进行学习，在实体机上安装了Linux发行版Ubuntu 19.04。主要学习了在Linux系统下，对于文件的处理；文件权限和用户权限的管理；软件包的管理——包括从源码到apt（在虚拟机上安装CentOS使用yum进行软件包管理）的在线管理；学习了Linux下文件系统的管理，包括分区和挂载等；学习了在Linux发行版下一些Shell的基础。

### 二、第二周2019年10月6日

## 1 论文研究

**1.1 基数约束的CNF**

Towards an Optimal CNF Encoding of Boolean Cardinality Constraints

布尔基数约束的最优CNF编码

基数约束的CNF是指对CNF中命题变量的数量作出一些限制。常见的形式有：n个变量的CNF中，至少或者至多有k个变量可以被赋值为真。文中提出了一个统一的框架来解决这类问题。文中提出两种改进现有结果的新编码，一个只需要7n子句和2n个附加变量，另一个要求O(n\*k)个子句。其优点是：仅通过单位传播就可以在线性时间内检测到不一致性。此外文中还证明了任何此类编码所需子句数的线性下界。

1.2 **ML解释：基于启发式的白盒方法**

Deep inside convolutional networks: Visualising image classification models and saliency maps

本文提出了图像分类的可视化模型，这一模型使用了深度卷积网络（Convnet）。基于分别计算分类得分的梯度，我们考虑了两种可视化技术：第一种生成了一张图可以最大化图像分类的得分，由Convnet捕获。第二种对于给定的图片和类，计算其显著性图。我们的研究表明，这种显著图可以用在弱监督目标分割上（使用卷积网络）。最后，我们建立了基于梯度的卷积网络与反卷积神经网络的联系。

这是一种基于启发式的白盒方法，通常用在计算机视觉领域。

**1.3 LIME解读**

“Why Should I Trust You?” Explaining the Predictions of Any Classifier

我为什么信任你？解释任意分类器的预测

尽管被广泛采用，机器学习模型仍然大多是黑匣子。然而，了解预测背后的原因对于评估模型中的信任非常重要。如果一个人计划基于预测采取行动，或者在选择是否部署新模型时，信任是至关重要的。这种理解进一步提供了对模型的洞察，可以用来将不可信的模型或预测转化为可信的模型或预测。本文中，我们提出了一种新的解释技术LIME，它通过在预测周围局部地学习一个可解释的模型，以可解释和可靠的方式解释任何分类器的预测。我们进一步提出了一种解释模型的方法，通过以非冗余的方式呈现具有代表性的个体预测及其解释，将任务定义为子模块优化问题。我们通过解释文本（如随机森林）和图像分类（如神经网络）的不同模型来展示这些方法的灵活性。解释的有效性通过新的实验来证明，无论是模拟实验还是人类实验。我们的解释在各种需要信任的场景中增强了用户的能力：决定是否应该信任一个预测，在模型之间进行选择，改进一个不可信的分类器，以及检测为什么不应该信任一个分类器。

我们在建立模型的时候，经常会思考我们的模型是不是够稳定，会不会出现样本偏差效应， p>>N时候会不会过拟合？我们检查模型稳定，我们进行一些cross-validation来看看各项评估指标方差大不大。 可是如果样本一开始因为采样偏差导致样本有偏，导致模型和实际情况有差异，这个就不太好评估了。同样，p>>N也会有类似的问题，尤其在文本挖掘领域。一般情况，如果特征不是很多的话，尤其像logistic regression这样的模型，我们会把模型权重给打印出来看看，看看训练出的模型结果，是否和人的经验吻合。下面是LIME文章中提到一个文本分类的例子，预测一段文本是无神论相关的还是基督徒相关的。文中分类器预测结果这篇文本是无神论相关的，可是主要区分特征却与人的经验十分不吻合的，这样的模型是不能让人信服的，当我们把这几个特征删除后，预测结果又反向了。我们可以通过人工构建一些由这些特征组成的文本来加入到预测实验中，会大大降低模型性能。

LIME是Local Interpretable Model-Agnostic Explanations的缩写。LIME的目的是试图解释模型在预测样本上的行为，这种解释是可被理解的，并且这种解释是模型无关的，不需要深入到模型内部。作者提出的方法一种局部方法，非全局的，在每个预测样本附近随机采样产生一些样本。选取K个特征，我们可以在采样的样本以及这K个特征上，做加权回归模型。回归模型输出的K个特征以及权重，就是分类器对预测样本的解释。

**1.4 ANCHOR解读**

Anchors: High-precision model-agnostic explanations

ANCHOR：高精度模型解释器

可解释的核心是：用户能足够理解模型的行为，且能精确地预测模型对于样本的预测结果。大多数的局部可解释模型，都是使用一个线性模型去拟合模型的局部行为，这样线性模型能给出样本中不同特征的相对重要性。但是，由于线性模型拟合的是局部的结果，对于一个未知样本，不能确定线性模型的结果是否适用于该样本（即不确定该样本是否在局部范围内），这也就是上文提到的“覆盖度”，线性模型的覆盖度是不确定的。这样就会导致低用户精确度（用户预测模型行为的精确度）。本文提出新的模型无关的可解释模型，基于if-then的规则，我们称它为Anchors。

Anchors给出的解释是模型在局部行为的充分条件，也就是说，若模型满足该条件，则模型一定（大概率）会给出某种分类。局部可解释模型的思想为：当原始模型很复杂以至于难以给出简洁的解释时，聚焦于单个样本的预测来做出解释是可行的。接下来我们在不同的数据集（表格，文本，图片等）及不同的模型（情感分类，词性分类，文本生成）上做了实验来验证Anchors, 并进行了user study来说明Anchors对用户理解模型行为的贡献。

### 三、第三周2019年10月13日

## 1 论文研究

本周我主要对研讨班上学习的论文进行整理，第一篇论文的研究内容主要与机器学习在SAT问题中的应用有关，其余论文是关于强化学习和深度学习算法的改进。

CrystalBall: Gazing in the Black Box of SAT Solving

现代SAT求解器通过复杂的启发式方法实现了可伸缩性和鲁棒性，这些启发式方法难以理解和解释。 因此，新算法洞察力的发展主要局限于专家直觉，而对新洞察力的评估则仅限于根据求解程序的运行时间或代理求解程序的运行时间的性能度量。 在这种情况下，是否有可能开发一种框架，以提供对SAT求解器执行的白盒访问，从而可以帮助SAT求解器开发人员和用户综合现代SAT求解器的算法启发式算法？

本文就设计了一个这样的框架称为CrystalBall。 将现代冲突驱动子句学习（CDCL）求解器视为针对不同任务（例如分支，子句内存管理和重新启动）的分类器和回归器的组合，专注于派生分类器以保留或丢弃学习的子句。与基于机器学习的最新技术不同，CrystalBall采用监督学习，并使用从单个SAT求解器运行中提取的大量千兆字节数据进行预测分析。我们将现代CDCL求解器视为针对不同任务的分类器和回归器的组合，例如分支（分支到哪个变量），子句内存管理（哪些学习的子句保留在内存中以及哪些子句抛出），重新启动（终止时） 分支并重新启动等等。为了更深入地了解基础分类器，第一步，我们建立了一个框架，以在求解阶段为SAT求解器提供白盒访问。 我们设想，这样的框架将允许最终用户对他们的试探法的性能获得深入的数据驱动的理解，并帮助他们设计更好的试探法。 我们并不是要取代专家的直觉，而是提出一种“在环专家”的方法，在此方法中，CrystalBall会为专家提供统计上合理的可解释分类器。

因此本文是利用机器学习对SAT算法进行改进，具体做法是将sat算法中分支、二元传播、矛盾分析等处理过程中的一些数据作为特征保存在数据库中，根据这些保存的特征作为输入进行模型训练，并利用得到的结果生成新的学习子句加入到CNF子句集中。

Deep Exploration via Bootstrapped DQN

科学探索仍然是强化学习（RL）的主要挑战。 诸如“贪婪”之类的常见探索抖动策略不会进行时间扩展（或深层）探索； 这可能导致数据需求成倍增长。 但是，在复杂的环境中，大多数用于统计有效的RL的算法在计算上都不易处理。 随机值函数提供了一种有前景的方法来进行广义的科学探索，但是现有算法与非线性参数化值函数不兼容。 作为解决此类情况的第一步，本文开发了自举DQN。 我们证明自举DQN可以将深度探索与深度神经网络相结合，从而比任何抖动策略都以指数级的速度更快地学习。 在Arcade学习环境中，自举DQN可以大大提高大多数游戏的学习速度和累积性能。

TD OR NOT TD: ANALYZING THE ROLE OF TEMPORAL DIFFERENCING IN DEEP REINFORCEMENT LEARNING

我们对强化学习（RL）的理解是由几十年前使用表格表示法和线性函数逼近器获得的理论和经验结果形成的。 这些结果表明，使用时间差分（TD）的RL方法优于直接蒙特卡洛估计（MC）。 这些结果如何在处理感知上复杂的环境和深度非线性模型的深度RL中保持呢？ 在本文中，我们使用专门设计的环境来控制TD在现代深度RL中的作用，这些环境控制着影响绩效的特定因素，例如奖励稀疏性，奖励延迟和任务的感知复杂性。 将TD与无限水平MC进行比较时，我们能够在现代环境中重现经典结果。 但是我们也发现，即使奖励稀少或延迟，有限水平MC也不比TD低。 这使得MC在深度RL中成为TD的可行替代方案。

NOISY NETWORKS FOR EXPLORATION

我们介绍了NoisyNet，这是一种深度强化学习代理，其权重添加了参数噪声，并表明代理策略的诱发随机性可用于辅助高效探索。 通过梯度下降以及其余网络权重来学习噪声参数。 NoisyNet易于实现并且几乎没有计算开销。 我们发现，用NoisyNet代替A3C，DQN和Dueling代理人（分别为熵奖励和-贪婪）的常规探索启发法，可以在许多Atari游戏中获得更高的分数，在某些情况下，它可以将代理人从亚人类提升为超人类性能。

Evolution Strategies as a Scalable Alternative to Reinforcement Learning

我们探索使用进化策略（ES）（一种黑盒优化算法）来替代流行的基于MDP的RL技术（例如Qlearning和Policy Gradients）。 在MuJoCo和Atari上进行的实验表明，ES是一种可行的解决方案策略，可以很好地扩展可用CPU的数量：通过使用基于公共随机数的新颖通信策略，我们的ES实施仅需要通信标量，因此可以扩展到一千多名并行工作者。 这使我们能够在10分钟的训练中解决10分钟内的3D人形行走问题，并在大多数Atari游戏中获得竞争性结果。 此外，我们着重介绍了ES作为黑盒优化技术的一些优点：它对动作频率和延迟的奖励是不变的，可以忍受极长的视野，并且不需要时间折现或价值函数逼近。

### 四、第四周2019年10月20日

## 1 实验进展

* 1. 对启发式算法的优化

对启发式策略进行优化，提高所有的exactly-one约束中变量的得分，使他们更容易被选择和赋值。对大量测试用例进行测试，测试结果显示当exactly-one约束中变量的得分增加值设为10时，程序的运行时间最短，效率提高的最多。对于积分增加值设为10时，效率提高最明显，还有待探究原因。

* 1. 对识别exactly-one约束算法的优化

我在源程序中添加测试用例中的exactly-one约束识别的算法。改进后的程序在运行含exactly-one约束很多的测试用例时，运行时间明显减少，运行效率明显提高。识别并标识exactly-one约束后，我们是否要把exactly-one约束包含的二元文字对进行删除？对此我用测试用例进行实验并比较删除前后的运行效率。我发现删除后运行时间反而更长，效率反而更低。我分析是因为exactly-one约束本身蕴含的二元文字对能帮助我们快速剪枝，如果将他们全部删去，有些反而会在矛盾分析中重新学习出来，因此导致运行时间变长，运行效率变慢。

* 1. 目前效率

经过测试用例的测试，目前程序运行效率比C2D快一个数量级，但是仍与D4有微小差距。

我仍尝试从别的方面继续提高编译器效率，争取在不久的将来，能实现在运行时间上少于D4，或者达到在运行某一类测试用例时效率比D4高。

### 五、第五周2019年10月27日

## 1 实验方面

* 1. 对更多测试用例进行测试，发现上周得到的结果是不对的。对于已经识别出的exactly-one约束，其蕴含的二元子句是否删除呢？测试了不同的测试用例集我发现，对于变量和子句数目越多、规模越大的测试用例，删掉二元子句会显得更高效。

图一 是否删除二元子句对比

上图横坐标是测试用例编号，纵坐标是运行时间单位毫秒。

图中靠上的蓝色线是不删除二元子句的运行结果，下面的橘色线是删除后运行的结果。删除后的运行时间更低，运行效率更高。规模越大的测试用例效果越明显。

图二 效率对比C2D和D4

上图蓝色柱是C2D，橘色柱是我所优化后的编译器，灰色柱是D4。纵坐标是运行时间单位毫秒。在上图测试用例中，我的编译器和D4都比C2D快很多。但对于规模特别大的测试用例（变量和子句数上千），我的编译器运行时间就会和C2D在一个数量级，与D4相比差很多（图上为列出）。因此目前我优化的编译器在处理规模较小的测试用例中更有优势。

图三 效率对比D4

图三是对图二对比D4的放大，橘色柱是我优化的编译器，灰色是D4。目前我的结果与D4还有些差距。

## 1 其他方面

学习了程序分析技术并写了一些总结。

解决了工作站之前插上网线连不上网的问题。

### 六、第六周2019年11月3日

## 1 实验方面

选取不同领域的测试用例对实验进行了更多的测试。

我所优化的编译器暂时用HST表示，下面是测试数据的记录。

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例/时间ms | C2D | D4 | HST |
| BMSk3-96 | 401 | 19.059 | 31.761 |
| BMSk3-241 | 390 | 13.781 | 30.454 |
| CBSk3-1 | 408 | 9.338 | 17.804 |
| CBSk3-170 | 383 | 4.288 | 14.266 |
| SW100-8-LP6-3 | 36192 | 1545 | 14564 |
| Urf100-39 | 449 | 11.238 | 17.212 |
| Urf100-334 | 406 | 11.332 | 15.536 |
| RTIk3-1 | 392 | 9.484 | 10.134 |
| RTIk3-314 | 410 | 3.772 | 7.641 |
| AIM50-3-YES-4 | 230 | 3.868 | 0.776 |
| AIM200-2-YES-3 | 710 | 0.712 | 2.542 |

## 2 论文方面

关于近似模型计数的一些论文

A scalable approximate model counter

近似模型计数是在给定的公差和置信度内对满足要求的数量进行计数的方法，它是精确模型计数的一种实用替代方法，然而基本上仅在理论上研究了近似模型计数。目前近似模型计数实现仅适用于DNF公式。本文介绍一种新颖的算法以及一种参考实现，它是CNF公式的第一个可扩展的近似模型计数器。 该算法通过向SAT求解器发出多项式调用来工作。本文提出的ApproxMC可缩放到具有成千上万个变量的公式，ApproxMC以高置信度报告接近准确计数的边界，并且在对于无法计算精确模型计数太大的情况下也成功以小公差和高置信度报告边界。

Taming the Curse of Dimensionality: Discrete Integration by Hashing and Optimization

集成受维数诅咒的影响，随着问题维数的增长，集成很快变得棘手。 我们提出了一种随机算法，该算法以高概率给出了在指数大集合上定义的一般离散积分的常数因子近似值。 该算法仅依赖于解决离散实例优化问题的少量实例，这些实例受用作哈希函数的随机生成的奇偶约束的约束。 作为一个应用程序，我们证明了通过少量的MAP查询，我们可以有效地逼近离散图形模型的分区功能，该模型又可以用于边际计算或模型选择。

## 3 其他方面

研读论文准备讲下周二11月5日的讨论班。

专业选修课临近结课考试，对程序设计分析、基于模型的诊断技术、数据挖掘、面向对象的数据库原理四门课进行复习和整理。

### 七、第七周2019年11月10日

## 1讨论班前和后

周一周二准备研讨班要讲的的论文，主要是研究论文和制作PPT。

研讨班结束后我对研讨班上提出的一些问题进行重新思索，尝试运行了anchor给出的源代码并得到了正确的结果。对于我讲的论文中一些残余问题，例如实验结果的图像为什么是上升的，我再次学习后认为是因为参与到模型计数中的anchor解释数目越多，解释的精度就会越高。因为单个解释是片面的，越多的解释能够覆盖更多的情况，越能描述这个模型，因此图像是单调递增的。

## 2 读书和准备考试

通过阅读周志华《机器学习》一书对机器学习进行学习。了解了机器学习领域的一些基本术语，假设空间，归纳偏好，模型评估与选择，线性模型等内容。这本书预计还需要两周读完。

下周还有两门课要结课考试，所以本周也花了一些时间对学习内容进行了复习。

## 3 论文方面

本周对新的论文也进行了一些学习。Scikit-learn使用一致的，面向任务的界面公开了各种机器学习算法，包括受监督的和不受监督的，因此可以轻松比较给定应用程序的方法。 由于它依赖于科学的Python生态系统，因此可以轻松地将其集成到传统统计数据分析范围之外的应用程序中。Scikit-learn利用这种丰富的环境来提供许多著名的机器学习算法的最新实现，同时保持与Python语言紧密集成的易于使用的界面。 这满足了软件和网络行业以及计算机科学以外的领域（例如生物学或物理学领域）的非专家对统计数据分析日益增长的需求。而CopyVIZ是开源的图形可视化软件。图形可视化是将结构信息表示为抽象图和网络图的一种方式。它在网络、生物信息学、软件工程、数据库和网页设计、机器学习以及其他技术领域的视觉界面中有重要应用。

### 八、第八周2019年11月17日

## 1实验数据

改进前和改进后的数据，对于含有很多exactly-one约束的测试用例，改进后的效率有提升，运行规模越大的测试用例提升越明显，部分测试用例效率提升甚至达到了百分之四十。表格中没有标颜色的表示改进后比改进前效率高，黄色表示的是改进后效率有明显的提升，灰色表示的是改进后比改进前效率低。

但对于含有很少exactly-one约束的测试用例，改进后的时间效率反而会比改进前降低一些。因为用来识别exactly-one约束的时间并没有在后续搜索中得到补偿。

数据单位是毫秒。

测试还没有结束，这只是一部分数据。

|  |  |  |
| --- | --- | --- |
| 测试用例 | 改进前 | 改进后 |
| Flat50-2 | 12.881 | 11.218 |
| Flat50-6 | 36.16 | 23.493 |
| Flat50-9 | 25.379 | 21.969 |
| Flat50-11 | 15.247 | 11.177 |
| Flat50-13 | 37.928 | 35.733 |
| Flat50-16 | 12.366 | 10.738 |
| Flat50-20 | 30.44 | 26.232 |
| Flat50-24 | 10.752 | 12.351 |
| Flat50-31 | 27.214 | 19.472 |
| Flat50-47 | 12.032 | 13.273 |
|  |  |  |
| Flat100-1 | 35.564 | 24.252 |
| Flat100-4 | 83.242 | 64.797 |
| Flat100-10 | 667.77 | 839.413 |
| Flat100-19 | 74.758 | 66.241 |
| Flat100-28 | 42.868 | 30.649 |
| Flat100-39 | 40.534 | 30.274 |
| Flat100-51 | 35.033 | 17.037 |
| Flat100-61 | 61.597 | 75.751 |
|  |  |  |
|  |  |  |
| Flat200-2 | 5256.33 | 3224.33 |
| Flat200-10 | 661.222 | 584.318 |
| Flat200-11 | 877.49 | 365.633 |
| Flat200-16 | 23526.2 | 13714 |
| Flat200-24 | 335.177 | 454.452 |
| Flat200-30 | 1431.33 | 1402.98 |
| Flat200-40 | 663.34 | 591.964 |
| Flat200-44 | 1767.27 | 1135.06 |
| Flat200-51 | 675.773 | 669.532 |
| Flat200-58 | 12171.2 | 8983.9 |
|  |  |  |
| Uf75-02 | 10.11 | 12.987 |
| Uf75-09 | 13.526 | 14.878 |
| Uf75-016 | 1.938 | 2.251 |
| Uf75-023 | 3.582 | 4.151 |
| Uf75-030 | 4.271 | 4.542 |
| Uf75-037 | 4.02 | 4.691 |
| Uf75-044 | 2.756 | 3.209 |
| Uf75-051 | 3.575 | 4.146 |
| Uf75-058 | 8.787 | 10.323 |
| Uf75-059 | 5.859 | 6.787 |
|  |  |  |
| Uf150-020 | 101.735 | 126.625 |
| Uf150-024 | 90.186 | 109.425 |
| Uf150-046 | 228.22 | 288.472 |
| Uf150-048 | 761.723 | 999.844 |
| Uf150-059 | 58.487 | 62.399 |
| Uf150-061 | 79.452 | 99.631 |
| Uf150-065 | 102.657 | 123.652 |
| Uf150-075 | 139.422 | 173.478 |
| Uf150-081 | 101.375 | 122.498 |
| Uf150-089 | 534.17 | 678.584 |
| Uf150-094 | 82.476 | 97.132 |
|  |  |  |

## 2 其他方面

复习和准备考试，目前有三门选修课已经结课。

继续阅读了《机器学习》这本书。

### 九、第九周2019年11月24日

## 1 实验方面

对某些不含exactly-one约束的测试用例进行预处理，一种思路是随机判断m个长子句是不是exactly-one约束，如果他们都不是，那么我们认为这个测试用例包含exactly-one约束的概率低，并结束对这个测试用例的判断。第二种思路是计算长子句是exactly-one所需要的二元子句的个数，与测试用例中包含的二元子句数进行对比，如果相差的数量级太大，我们也结束对这个测试用例的判断。

算法：

Bool judge=false; //judge=false不进行exactly- one识别，反之进行识别

itr=begin;

if(itr->exact)

judge=true；

else

{

int num\_bi\_judge=(itr->len-1) \* (itr->len)/2; //二元文字对个数

for (unsigned x=0; x<itr->len-1; x++)

{

for (unsigned y=x+1; y<itr->len; y++)

{

unsigned m=LIT\_NEG ( itr->lits[x] );

unsigned n=LIT\_NEG ( itr->lits[y] );

vector< unsigned >cmpp=binary\_clauses[m];

vector< unsigned >::iterator cmp = find(cmpp.begin(),cmpp.end(),n);

if(cmp==cmp.end())

{

goto judgefalse; //如果不是exactly-one约束，退出循环

}

else

num\_bi\_judge--;

}

}

judge=! num\_bi\_judge; //是exactly-one约束，judge=true

itr->exact=judge;

}

对改进后的算法再次进行测试，并与之前的测试数据对比。

算法对于不含或含很少的exactly-one约束的测试用例时间和改进前相差无几。

表格中时间单位为毫秒；改进后2与改进前的时间差距保持在零点几毫秒，与程序每次测试的实验误差在一个数量级，基本可以忽略不计。

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例 | 改进前 | 改进后1 | **改进后2** |
| Uf75-02 | 10.11 | 12.987 | 10.526 |
| Uf75-09 | 13.526 | 14.878 | 13.494 |
| Uf75-016 | 1.938 | 2.251 | 1.959 |
| Uf75-023 | 3.582 | 4.151 | 3.611 |
| Uf75-030 | 4.271 | 4.542 | 3.932 |
| Uf75-037 | 4.02 | 4.691 | 4.082 |
| Uf75-044 | 2.756 | 3.209 | 2.781 |
| Uf75-051 | 3.575 | 4.146 | 3.622 |
| Uf75-058 | 8.787 | 10.323 | 8.898 |
| Uf75-059 | 5.859 | 6.787 | 6.005 |
|  |  |  |  |
| Uf150-020 | 101.735 | 126.625 | 101.384 |
| Uf150-024 | 90.186 | 109.425 | 90.821 |
| Uf150-046 | 228.22 | 288.472 | 230.096 |
| Uf150-048 | 761.723 | 999.844 | 775.627 |
| Uf150-059 | 58.487 | 62.399 | 57.615 |
| Uf150-061 | 79.452 | 99.631 | 80.094 |
| Uf150-065 | 102.657 | 123.652 | 103.064 |
| Uf150-075 | 139.422 | 173.478 | 141.899 |
| Uf150-081 | 101.375 | 122.498 | 102.108 |
|  |  |  |  |

### 十、第十周2019年12月1日

## 1 实验方面

（一）用矩阵方法实现exactly-one约束的识别：

首先是矩阵的构造

bool binaryClause[max\_var\*2+2][max\_var\*2+2]={false}; //初始化矩阵

for (itr = begin; itr < end; itr++)

{

if (clause\_seen[itr - begin])

clause\_seen[itr - begin] = false;

else if (itr->len == 2)

{

Add\_Binary\_Clause\_Naive(itr->lits[0], itr->lits[1]);

binaryClause[itr->lits[0]][itr->lits[1]]=true; //矩阵对应位置设置为1

binaryClause[itr->lits[1]][itr->lits[0]]=true;

//jici++;

if (itr->exact == 1)

{

Add\_Binary\_Clause\_Naive(LIT\_NEG(itr->lits[0]), LIT\_NEG(itr->lits[1]));

jici++;

}

}

else

long\_clauses.push\_back(Clause(itr->lits, itr->len, itr->exact));

}

接下来是识别函数

itr = begin;

if (itr->exact)

judge = true;

else

{

int num\_bi\_judge = (itr->len - 1) \* (itr->len) / 2;

for (unsigned x = 0; x < itr->len - 1; x++)

{

for (unsigned y = x + 1; y < itr->len; y++)

{

unsigned m = LIT\_NEG(itr->lits[x]);

unsigned n = LIT\_NEG(itr->lits[y]);

if(binaryClause[m][n]) //矩阵对应位置是否为1

{

num\_bi\_judge--;

}

else{

goto zoule;

}

}

}

}

（二）已标记的exactly-one约束等价展开为多个子句对的集合：

if(itr->exact)

{

for (unsigned x = 0; x < itr->len - 1; x++)

{

for (unsigned y = x + 1; y < itr->len; y++)

{

unsigned m = LIT\_NEG(itr->lits[x]);

unsigned n = LIT\_NEG(itr->lits[y]);

Add\_Binary\_Clause\_Naive(m, n);

itr->exact=false;

}

}

}

实验数据还没有测试完，这次的周报暂时先没有写出实验结果。

### 十一、第十一周2019年12月8日

## 1 算法

着色问题flat测试用例集的exactly-one约束是展开表示的，我写了一个算法，将其输出为exactly-one约束用e显式标识的形式。

<?php

$file = file('flat30-14.cnf');

$m=$n=$s=0;

for($i = 0;$i<count($file);$i++){

if($file[$i][0]=='c'||$file[$i][0]=='p'){

$s++;

continue;

}

elseif ($file[$i][0]!='-') {

$in = strlen($file[$i]);

$file[$i]='e '.substr($file[$i],0,$in-1)."\n";

$m++;

}

}

$m=$m\*3;

$i=0;

while($n!=$m){

if($file[$i][0]=='-') {

array\_splice($file,$i,1);

$n++;

continue;

}

$i++;

}

$i=0;

while(true){

if($file[$i][0]=='p'){

if(preg\_match('/\d+/',$file[$i],$arr)){

$file[$i]='p cnf'.' '.($arr[0]).' '.(count($file)-$s)."\n";

}

break;

}

$i++;

}

file\_put\_contents('1.txt',$file);

例如

-1 -2 0

-1 -3 0

-2 -3 0

1 2 3 0

-4 -5 0

-4 -6 0

-5 -6 0

4 5 6 0

-7 -8 0

-7 -9 0

-8 -9 0

7 8 9 0

-10 -11 0

-10 -12 0

-11 -12 0

10 11 12 0

-13 -14 0

-13 -15 0

-14 -15 0

经过处理后变为：（仅列出了某个测试用例的局部）

e 1 2 3 0

e 4 5 6 0

e 7 8 9 0

e 10 11 12 0

e 13 14 15 0

## 2 测试

用处理过的显式标示的测试用例进行测试，分别记录用展开方式处理e-o约束和本文中直接处理e-o约束的运行时间。仅测量了部分数据。

|  |  |  |
| --- | --- | --- |
| 测试用例 | 展开处理e-o约束 | 直接处理e-o约束 |
| Flat50-6 | 30.019 | 15.601 |
| Flat50-94 | 12.985 | 11.509 |
| Flat100-8 | 92.534 | 83.104 |
| Flat100-94 | 94.819 | 86.327 |
| Flat100-51 | 34.331 | 19.011 |
| Flat200-10 | 4460.07 | 3871.75 |
| Flat200-44 | 1758.33 | 1168.47 |