研究的基础和趋势
理论计算机科学
卷。9 个，编号。now 3–4 (2014) 211–407 /
c 2014 C。Dwork 和 a。罗斯
the essence of knowledge DOI: 10.1561/0400000042 /

差别隐私的算法基础

辛西娅·德沃克亚伦·罗斯
美国宾夕法尼亚大学微软研究院，美国 dwork@microsoft.com
aaroth@cis.upenn.edu

内容

输入阀,交互式视讯,自变量(Independent Variable)

摘要

隐私保护数据分析问题有着跨越多个学科的悠久历史。随着关于个人的电子数据变得越来越详细，随着技术使这些数据的收集和管理变得越来越强大，对一个健壮的、有意义的和数学上严格的隐私定义的需

求增加了，同时需要一类计算丰富的算法来满足这个定义。差别隐私就是这样一个定义。

在激励和讨论了差别隐私的含义后，这本专著的优势在于致力于实现差别隐私的基本技术，以及这些技术在创造性组合中的应用，使用查询发布问题作为一个正在进行的例子。一个关键点是，通过重新思考计算目标，人们通常可以获得比用不同的私有实现有条不紊地替换非私有计算的每个步骤更好的结果。尽管有一些惊人强大的计算结果，但仍然存在根本性的限制——不仅仅是在差异隐私可以实现什么方面，而且在任何防止隐私完全崩溃的方法可以实现什么方面。实际上，这里讨论的所有算法都保持了针对任意计算能力的对手的不同隐私。某些算法是计算密集型的，其他算法是高效的。讨论了对手的计算复杂度和算法。

然后，我们从基础转向除了 queryrelease 之外的应用，讨论不同的机制设计和机器学习的私有方法。绝大多数关于差分私有算法的文献都考虑了一个单一的、静态的数据库，该数据库要经过许多分析。讨论了其他模型中的差异隐私，包括分布式数据库和数据流计算。

最后，我们注意到，这项工作旨在全面介绍差异隐私的问题和技术，但并不打算进行详尽的调查——到目前为止，在差异隐私方面有大量的工作，我们只能涵盖其中的一小部分。

C. Dwork 和 a。罗斯。*差别隐私的算法基础*。理论计算机科学的基础和趋势，第一卷。9 个，编号。第 3-4 页。211–407, 2014.

DOI: 10.1561/0400000042。

前言

隐私保护数据分析问题有着跨越多个学科的悠久历史。随着关于个人的电子数据变得越来越详细，随着技术使这些数据的收集和管理变得越来越强大，对一个健壮的、有意义的和数学上严格的隐私定义的需求增加了，同时需要一类计算丰富的算法来满足这个定义。*差别隐私就是这样一个定义*。

在激发和讨论了差异隐私的含义后，本书的优势在于致力于实现差异隐私的基本技术，以及这些技术在创造性组合中的应用(第 3-7 节)，使用查询释放问题作为一个持续的例子。一个关键点是，通过重新思考计算目标，人们通常可以获得比用不同的私有实现有条不紊地替换非私有计算的每个步骤更好的结果。

尽管有一些惊人的强大的计算结果，但仍然有一些基本的限制——不仅仅是用差别隐私可以实现什么，而且用任何防止隐私完全崩溃的方法都可以实现什么(第 8 节)。

实际上，本书中讨论的所有算法都保持了针对任意计算能力的对手的不同隐私。某些算法是计算密集型的，其他算法则是

高效。第 9 节讨论了对手的计算复杂度和算法。

在第 10 节和第 11 节中，我们将从基础转向除查询发布之外的应用，讨论机制设计和机器学习的不同私有方法。绝大多数关于差分私有算法的文献都考虑了一个单一的、静态的数据库，该数据库要经过许多分析。其他模型中的差异隐私，包括分布式数据库和数据流计算，将在第 12 节中讨论。

最后，我们注意到这本书是对差别隐私的问题和技术的全面介绍，但并不是详尽的调查——到目前为止，在差别隐私方面有大量的工作，我们只能涵盖其中的一小部分。

一

差别隐私的承诺

“差异隐私”描述了数据持有人或馆长对数据主体做出的承诺:“无论其他研究、数据集或信息源如何，您都不会因允许您的数据用于任何研究或分析而受到不利或其他影响。最好的情况是，差异私有数据库机制可以使机密数据广泛用于准确的数据分析，而无需求助于数据清理室、数据使用协议、数据保护计划或受限视图。尽管如此，数据效用最终会被消耗掉:信息恢复的基本法则指出，对太多问题过于准确的回答会以惊人的方式破坏隐私。差分隐私算法研究的目标是尽可能推迟这种必然性。

差异隐私解决了一个悖论，即在学习关于人群的有用信息的同时，对个人一无所知。医学数据库可能会告诉我们，吸烟会导致癌症，影响保险公司对吸烟者长期医疗费用的看法。吸烟者是否受到分析的伤害？也许——他的保险

如果保险公司知道他吸烟，保费可能会上涨。他也可能得到帮助——得知自己的健康风险后，他参加了戒烟计划。吸烟者的隐私被泄露了吗？研究后对他的了解肯定比以前多，但他的信息被“泄露”了吗？差异隐私会认为不是，理由是对吸烟者的影响是一样的，与他是否在研究中无关。影响吸烟者的是研究中得出的结论，而不是他在数据集中的存在与否。

差别隐私确保得出相同的结论，例如，吸烟导致癌症，而不管任何个人是选择进入还是选择退出数据集。具体来说，它确保任何输出序列(对查询的响应)都“本质上”同样可能发生，而与任何个人的存在与否无关。这里，概率被隐私机制(由数据管理员控制的东西)做出的随机选择所取代，术语“本质上”被参数 ε 捕获。更小的 ε 将产生更好的隐私(和更不准确的响应)。

差别隐私是一个定义，而不是算法。对于给定的计算任务 T 和给定的 ε 值，将有许多差分私有算法来以 ε 差分私有的方式实现 T。有些会比其他更准确。当 ε 很小时，为 T 找到一个高度精确的 ε 差分私有算法可能很困难，就像为特定的计算任务找到一个数值稳定的算法可能需要付出努力一样。

1.1 隐私保护数据分析

差异隐私是为隐私保护数据分析问题量身定制的隐私定义。我们简要地讨论了这个问题的其他方法的一些关注点。

*数据不能完全匿名并保持有用。*一般来说，数据越丰富，越有趣，越有用。这导致了“匿名化”和“删除个人可识别信息”的概念，人们希望

数据记录可以被抑制，剩余的记录可以被发布并用于分析。然而，数据的丰富性使得能够通过有时令人惊讶的字段或属性集合来“命名”个人，例如邮政编码、出生日期和性别的组合，或者甚至是三部电影的名称以及个人观看这些电影的大致日期。这种“命名”能力可用于链接攻击，将不同数据集中的“匿名”记录与非匿名记录进行匹配。因此，马萨诸塞州州长的医疗记录是通过将匿名医疗遭遇数据与(公开提供的)选民登记记录进行匹配来识别的，其观看历史被包含在网飞公布的匿

名电影记录集合中作为推荐竞赛的培训数据的网飞订户是通过与互联网电影数据库(IMDb)的链接来识别的。

差异隐私中和了链接攻击:由于差异隐私是数据访问机制的一个属性，并且与对手可用的辅助信息的存在与否无关，因此对 IMDb 的访问不会允许对其历史在网飞训练集中的人进行链接攻击，也不会允许对不在训练集中的人进行链接攻击。

重新识别"匿名"记录不是唯一的风险。"匿名化"数据记录的重新识别显然是不可取的，这不仅是因为重新识别本身肯定会揭示数据集中的成员关系，还因为记录可能包含泄露信息，如果与个人相关联，可能会造成损害。特定紧急护理中心在给定日期收集的医疗遭遇记录可能仅列出少量不同的投诉或诊断。邻居在所述日期访问该设施的附加信息给出了该邻居状况的相当窄的可能诊断范围。可能无法将特定记录与邻居匹配的事实为邻居提供了最低限度的隐私保护。

*对大型集合的查询不是保护性的。*关于特定个人的问题无法准确安全地回答，事实上，人们可能希望立即拒绝它们(如果识别它们在计算上可行的话)。强制查询覆盖大型集合并不是万能药，下面的差异攻击就说明了这一点。假设已知先生。x 在某个医学数据库中。综合起来，两个大问题的答案是"数据库中有多少人有镰状细胞特征？以及"数据库中有多少人，不叫 X，有镰状细胞特征？产生先生的镰状细胞状态。X.

*查询审核有问题。*人们可能会尝试审核查询和响应的顺序，目的是在根据历史记录回答当前查询会危及隐私的情况下阻止任何响应。例如，审计员可能在寻找会构成差异攻击的成对查询。这种方法有两个困难。首先，拒绝回答问题本身就有可能被披露。第二，查询审计可能在计算上不可行；事实上，如果查询语言足够丰富，甚至可能不存在用于判断一对查询是否构成差异攻击的算法过程。

*汇总统计信息不"安全"。*从某种意义上说，概要统计作为隐私解决方案概念的失败与刚才描述的差异攻击是直接相关的。汇总统计的其他问题包括针对数据库的各种重建攻击，其中每个人都有一个需要保护的"秘密位"。效用目标可能是允许，例如，形式为"有多少满足性质 P 的人具有秘密比特值 1？另一方面，对手的目标是显著增加猜测个人秘密的机会。第 8.1 节中描述的重构攻击表明，即使是线性数量的这

种类型的查询也很难保护:除非引入足够的不准确性，否则几乎所有的秘密位都可以重构。

发布汇总统计数据的风险的一个显著例证是，一种统计技术的应用，最初旨在确认或反驳法医混合样本中存在个人的脱氧核糖核酸，以裁定个人是否参与全基因组关联研究。根据人类基因组计划的网站，“单核苷酸多态性，或称 SNPs(发音为“SNPs”)，是 DNA

当基因组序列中单个核苷酸(A、T、C 或 G)发生改变时发生的序列变异。例如，一个单核苷酸多态性可能会将脱氧核糖核酸序列从 AAGGCTAA 改变为 ATGGCTAA。在这种情况下，我们说有两个等位基因:A 和 t。对于这样的单核苷酸多态性，我们可以问，给定一个特定的参考人群，两种可能的等位基因的频率分别是多少？给定参考人群中 SNPs 的等位基因频率，我们可以检查这些频率对于患有特定疾病的亚群(“病例”组)的差异，寻找与疾病相关的等位基因。因此，全基因组关联研究可能包含大量单核苷酸多态性病例组的等位基因频率。根据定义，这些等位基因频率只是聚集的统计数据，并且(错误的)假设是，由于这种聚集，它们保护了隐私。然而，给定个体的基因组数据，理论上可以确定该个体是否在病例组中(因此，是否患有该疾病)。作为回应，美国国立卫生研究院和韦尔科姆信托终止了公众对他们资助的研究的总频率数据的访问。

即使对于差别隐私来说，这也是一个具有挑战性的问题，因为所涉及的测量数量很大——几十万甚至一百万——并且在任何情况下群体中的个体数量都相对较少。

*“普通”的事实并不“好”。*如果随着时间的推移，一个数据主体被跟踪，揭示“普通”事实(如购买面包)可能会有问题。例如，考虑先生。t .年复一年地定期买面包，直到突然转而很少买面包。分析师可能会得出这样的结论。t 很可能被诊断出患有二型糖尿病病。分析师可能是正确的，也可能是不正确的；不管怎样，先生。t 受到伤害。

*“就几个。*在某些情况下，特定技术实际上可以为数据集的“典型”成员，或者更一般地说，“大多数”成员提供隐私保护。在这种情况下，人们经常听到这样的说法，即这项技术是足够的，因为它损害了“少数”参与者的隐私。抛开对离群者可能正是隐私最重要的人的担忧不谈，“少数”哲学本质上并非毫无价值:需要做出社会判断，权衡成本和

收益。一个与“少数人”哲学相一致的清晰的隐私定义还有待发展；然而，对于单个数据集来说，通过随机选择一个行子集并将其全部释放，可以实现“仅几个”隐私(Lemma 4.3，第 4 节)。描述可以在随机子样本上执行的统计分析质量的采样界限控制要发布的行数。当“只有少数人”的理念遭到拒绝时，差别隐私提供了一种选择。

1.2 书目注释

Sweeney [81]将选民登记记录与“匿名化”的医疗接触数据联系起来；纳拉亚南和什马蒂科夫对网飞公布的匿名排名数据进行了联动攻击[65]。法医组合中的现场工作应归功于荷马等人。[46]. 第一次重建袭击是由于迪努尔和尼西姆[18]。

2

基本条款

这一部分提出了差别隐私的正式定义，并列举了它的一些关键属性。

2.1 计算模型

我们假设存在一个可信的策展人，他在数据库 D 中保存个人数据，通常由 n 行组成。直觉是每一行都包含单个个体的数据，并且，仍然直

觉地说，隐私目标是同时保护每一行，同时允许对整个数据库进行统计分析。

在非交互式或离线模型中，策展人生成某种对象，如“合成数据库”、汇总统计数据集合或“净化数据库”等。发布后，馆长不再扮演任何角色，原始数据可能会被销毁。

查询是应用于数据库的函数。交互式或在线模型允许数据分析师自适应地询问查询，根据观察到的对先前查询的响应来决定下一步提出哪个查询。

可信的策展人可以被一组个人运行的协议所取代，使用加密技术来实现安全的多方协议，但是在大多数情况下，我们不会求助于加密假设。第 12 节描述了该模型和文献中研究的其他模型。

当预先知道所有的查询时，非交互模型应该给出最好的准确性，因为它能够在知道查询结构的情况下关联噪声。相比之下，当事先不知道关于查询的信息时，非交互模型带来了严峻的挑战，因为它必须为所有可能的查询提供答案。正如我们将看到的，为了确保隐私，甚至为了防止隐私灾难，准确性必然会随着提问的数量而恶化，并且为所有可能的问题提供准确的答案将是不可行的。

隐私机制或简称为机制，是一种算法，它将数据库、数据类型的宇宙 X(所有可能的数据库行的集合)、随机位以及可选的一组查询作为输入，并生成输出字符串。希望输出字符串能够被解码，以产生相对准确的查询答案，如果后者存在的话。如果没有出现任何查询，那么我们就处于非交互的情况，希望输出字符串可以被解释来提供未来查询的答案。

在某些情况下，我们可能要求输出字符串是一个合成数据库。这是从可能的数据库行的宇宙 X 中提取的多集。这种情况下的解码方法是在合成数据库上执行查询，然后应用某种简单的变换，例如乘以一个比例因子，以获得查询的真实答案的近似值。

2.2 定义私有数据分析

在数据分析环境中定义隐私的一种自然方法是要求分析师在分析完成后对数据集中任何个人的了解不超过她在分析开始前所了解的。通过以下方式正式确定这一目标也是很自然的

2.2。 定义私有数据分析

要求对手对个人的前视图和后视图(即。访问数据库之前和之后)不应该“太不一样”，或者对数据库的访问不应该“太多地”改变对手对任何个人的看法。然而，如果数据库教会了什么，这种隐私的概念是无法实现的。例如，假设对手的(不正确的)先验观点是每个人都有两只左脚。对统计数据库的访问告诉我们，几乎每个人都有一只左脚和一只

右脚。对手现在对任何给定的被调查者是否有两只左脚有着非常不同的看法。

定义隐私的“前/后”或“一无所知”方法的部分吸引力在于直觉，如果对个人一无所知，那么个人就不会受到分析的伤害。然而，“吸烟致癌”的例子表明这种直觉是有缺陷的；罪魁祸首是辅助信息(Mr。x烟)。

定义隐私的“什么都不是学来的”方法让人想起密码系统的语义安全性。粗略地说，语义安全是指从密文中对明文(未加密的消息)一无所知。也就是说，在看到密文之后知道的关于明文的任何事情在看到密文之前都是已知的。因此，如果有辅助信息表明密文是对“狗”或“猫”的加密，那么密文就不会泄露关于“狗”或“猫”中哪一个被加密的进一步信息。从形式上来说，这是通过比较窃听者猜测“狗”和“猫”中的哪一个已经加密的能力和所谓的对手模拟器的能力来建模的，对手模拟器拥有辅助信息，但无法访问密文，以猜测相同的事情。如果对于每一个窃听对手，以及所有的辅助信息(对手和模拟器都知道)，对手模拟器的猜测概率基本上与窃听者相同，那么系统就享有语义安全。当然，为了使系统有用，合法接收者必须能够正确解密消息；否则语义安全可以很容易地实现。

我们知道，在标准的计算假设下，语义安全的密码系统是存在的，那么为什么我们不能构建语义安全的私有数据库机制来产生查询的答案，同时保持单个行的秘密呢？

首先，这个类比并不完美:在语义安全的密码系统中，有三方:消息发送方(加密明文消息)、消息接收方(解密密文)和窃听者(由于无法了解明文的任何信息而感到沮丧，因为在发送明文之前她还不知道)。相比之下，在私人数据分析的环境中，只有两方:负责隐私机制的策展人(类似于发送者)和数据分析师，他们接收对查询的信息性响应(类似于消息接收者)，并试图挤出个人的隐私泄露信息(类似于窃听者)。因为合法接收者与窥探的对手是同一方，所以加密的类比是有缺陷的:拒绝对手的所有信息意味着拒绝数据分析师的所有信息。

其次，就像加密方案一样，我们要求隐私机制有用，这意味着它会教给分析师一些她以前不知道的东西。对手模拟器无法提供这种教学；也就是说，没有模拟器可以“预测”分析师学到了什么。因此，我们可以将数据库视为随机(不可预测)位的弱来源，从中我们可以提取

一些非常高质量的随机性来用作随机填充。这可以在加密技术中使用，在加密技术中，秘密消息被添加到随机值(“随机填充”)中，以产生一个字符串，该字符串在理论上隐藏了秘密。只有知道随机垫的人才能知道秘密；任何对 pad 一无所知的一方对这个秘密一无所知，无论他或她的计算能力如何。给定对数据库的访问权，分析师可以学习随机 pad，但是对手模拟器，没有给定对数据库的访问权，对 pad 一无所知。因此，给定使用随机 pad 加密秘密作为辅助信息，分析员可以解密秘密，但是对手模拟器对秘密一无所知。这在对手/分析者学习秘密的能力和能力之间产生了巨大的差异

对手模拟器做同样的事情，消除了任何类似语义安全的希望。

吸烟导致癌症的例子中的障碍和语义安全的希望都是辅助信息。显然，为了有意义，即使在“合理的”辅助知识的背景下，隐私保证也必须成立，但是将合理的辅助知识与任意的辅助知识分开是有问题的。例如，使用政府数据库的分析师可能是一家大型搜索引擎公司的员工。对于这样的人可以获得的辅助知识信息，有哪些“合理”的假设？

2.3 形式化差别隐私

我们将从差别隐私的技术定义开始，然后继续解释它。差异化隐私将通过流程提供隐私；特别是它会引入随机性。随机过程隐私的一个早期例子是随机反应，这是一种在社会科学中开发的技术，用于收集关于尴尬或非法行为的统计信息，通过拥有一个属性 p 来捕获。研究参与者被告知如下报告他们是否拥有财产 P:

1. 抛硬币。
2. 如果是反面，那就如实回答。
3. 如果正面，再抛第二枚硬币，正面回答“是”，反面回答“否”。

“隐私”来自对任何结果的似是而非的否认；特别是，如果拥有财产 P 相当于从事非法行为，即使是“是”的回答也不能证明有罪，因为无论被申请人是否真正拥有财产 P，这个答案都有至少 1/4 的概率出现。准确性来自对噪声产生过程的理解(从随机化中引入虚假的“是”和“否”答案):“是”答案的预期数量是不具有属性 P 的参与者数量的 1/4 倍加上具有属性 P 的参与者数量的 3/4。因此，如果 P 是具有属性 P 的参与者的真实分数，那么“是”答案的预期数量是(1/4)(1p)+(3/4)P =(1/4)+P/2. 因此，我们可以将 p 估计为回答“是”的分数的两倍减去 1/2，即 2((1/4)+p/2)-1/2。

随机化至关重要；更准确地说，无论辅助信息的所有当前甚至未来来源如何，包括其他数据库、研究、网站、在线社区、八卦、报纸、政府统计数据等，任何非琐碎的隐私保证都需要随机化。这是从一个简单的混合论证中得出的，我们现在画出草图。为了矛盾起见，假设

我们有一个非平凡的确定性算法。非平凡表示存在一个查询和两个在这个查询下产生不同输出的数据库。一次更改一行我们看到存在一对数据库，只是单个行的值不同，在这两个数据库上，相同的查询产生不同的输出。知道数据库是这两个几乎相同的数据库之一的对手会知道未知行中数据的价值。

因此，我们需要讨论随机化算法的输入和输出空间。在这本专著中，我们研究离散概率空间。有时我们将我们的算法描述为从连续分布中采样，但是这些应该总是以适当谨慎的方式离散到有限精度(参见下面的注释 2.1)。通常，具有域 A 和(离散)范围 B 的随机化算法将与从 A 到 B 上的概率单纯形的映射相关联，表示为∑(B):

定义 2.1(概率单纯形)。给定一个离散集 B，B 上的概率单形，表示为∞(B)定义为:

$$\Sigma(b) = \{x \in r \mid b \mid : \text{所有 I 和 Xxi 的 Xi} \geqslant 0 = 1\}$$
$$i=1$$

定义 2.2(随机化算法)。具有域 A 和离散范围 B 的随机化算法 M 与映射 M:A→∞(B)相关联。在输入 a ∈ A 时，算法 M 输出 M(a) = b，每个 b ∈ B 的概率为(M(a))b。概率空间在算法 m 的硬币翻转上。

我们会认为数据库 x 是来自宇宙 x 的记录的集合。用直方图来表示数据库通常会很方便:x ∈ N|X|，其中每个条目 xi 代表数据库 X 中 i ∈ X 类型的元素数量(我们稍微滥用了符号，让符号 N 表示所有非负整数的集合，包括零)。在该表示中，两个数据库 x 和 y 之间距离的自然度量将是它们的“1”距离:

定义 2.3(数据库之间的距离)。数据库 x 的 1 范数表示为 kxk1，定义为:

$$kxk1 = \sum_{i=1}^{|X|} |xi|。$$

两个数据库 x 和 y 之间的距离是 kx yk1

注意，kxk1 是数据库 x 大小的度量(即。它包含的记录数), kx yk1 是 x 和 y 之间有多少记录不同的度量。

数据库也可以由多组行(X 的元素)甚至有序的行列表来表示，这是集合的一种特殊情况，其中行号成为元素名称的一部分。在这种情况下，数据库之间的距离通常通过汉明距离来测量，即，。它们不同的行数。

然而，除非另有说明，我们将使用上述直方图表示法。(但是，请注意，即使直方图表示法在数学上更加方便，在实际实现中，多集表示法通常也会更加简洁)。

我们现在准备正式定义差异隐私，这将直观地保证随机化算法在相似的输入数据库上表现相似。

定义 2.4(差别隐私)。对于所有 S ⊆范围(m)和所有 x，y ∈ N|X|来说，具有域 N|X|的随机化算法 m 是(ε，δ)-差分私有 if，使得 kx yk1≤1:

$$\Pr[M(x) \in S] \leqslant \exp(\varepsilon)\Pr[M(y) \in S] + \delta,$$

其中概率空间在机制 m 的硬币翻转上方。

如果 δ = 0，我们说 M 是 ε-差分私有。

通常，我们对小于数据库大小中任何多项式倒数的 δ 值感兴趣。特别是，数量级为 1/kxk1 的 δ 值非常危险:它们允许通过公布少数数据库参与者的完整记录来“保护隐私”——准确地说是第 1 节中讨论的“少数”原则。

然而，即使 δ 可以忽略不计，在(ε，0)-和(ε，δ)-差异隐私之间也有理论上的区别。其中最主要的是量化顺序的转换。(ε，0)-差异隐私确保，对于机制 M(x)的每次运行，观察到的输出(几乎)同样可能同时在每个相邻数据库上观察到。相比之下(ε，δ)-差异隐私表示，对于每对相邻数据库 x，y，事后观察到的值 M(x)极不可能在数据库为 x 时比数据库为 y 时产生更多或更少。然而，给定一个输出 ξ∨M(x)，有可能找

到一个数据库 y，使得 ξ 比数据库为 x 时更有可能在 y 上产生。也就是说，分布 M(y)中 ξ 的质量可以显著大于其在分布 M(x)中的质量。数量

$$(\xi) \qquad \Pr[M(x) = \xi]$$

$$LM(x)kM(y) = \ln \qquad \Pr[M(y) = \xi]$$

对我们很重要；我们称之为观察 ξ 所导致的隐私损失。这种损失可能是正的(当一个事件更有可能在 x 下而不是在 y 下)，也可能是负的(当一个事件更有可能在 y 下而不是在 x 下)。正如我们将在引理 3.17 中看到的那样，(ε，δ)-差分隐私确保了对于所有相邻的 x，y，隐私损失的绝对值将被概率至少为 1δ 的 ε 所限制。一如既往，概率空间在机制 m 的硬币之上。

差分隐私不受后处理的影响:如果没有关于私有数据库的额外知识，数据分析师无法计算私有算法 M 的输出函数，并使其不那么差分隐私。也就是说，如果一种算法保护个人隐私，那么数据分析师就不能仅仅坐在角落里思考算法的输出，从而增加隐私损失——无论是在正式定义下，还是在任何直观意义上。形式上，具有(ε，δ)差分私有算法 M 的数据无关映射 f 的组成也是(ε，δ)差分私有的:

命题 2.1(后处理)。设 M : N|X| → R 是(ε，δ)-差分私有的随机化算法。设 f : R → R0 为任意随机映射。那么 f♀M:N | X |→R0 是(ε，δ)差分私有。

*证据。*我们证明了确定性函数 f : R → R0 的命题.结果如下，因为任何随机映射都可以分解成确定性函数的凸组合，差分私有机制的凸组合是差分私有的。

使用 kx yk1≤1 修复任何一对相邻数据库 x，y，并修复任何事件 S ⊆ R0。设 T = {r ∈ R : f(r) ∈ S}。然后我们有:

$$\begin{aligned}\Pr[f(M(x)) \in S] &= \Pr[M(x)) \in T] \\ &\leq \exp(\epsilon) \Pr[\mathcal{M}(y) \in T] + \delta \\ &= \exp(\epsilon) \Pr[f(\mathcal{M}(y)) \in S] + \delta\end{aligned}$$

这正是我们想要的。□

紧接着从定义 2.4 得出(ε，0)-差分隐私以一种直接的方式组成:两个(ε，0)差分私有机制的组成是(2ε，0)-差分私有。更一般地(定理 3.16)，“ε 和 δ 相加”:k 个差分私有机制的组成，其中 ith 机制是(εi，δI)-差分私有，对于 1 ⩽ i ⩽ k，是(Pi εi，Pi δi)差分私有。

(ε，0)-差分私有机制的组隐私也紧随定义 2.4，隐私保证的强度随着组的大小线性下降。

定理 2.2。对于大小为 k 的组，任何(ε，0)-差分私有机制 M 都是(kε，0)差分私有的。也就是说，对于所有 kx yk1⩽k 和所有 S ⊆范围(m)

$$\Pr[M(x) \in S] \leqslant \exp(k\varepsilon)\Pr[M(y) \in S],$$

其中概率空间在机制 m 的硬币翻转上方。

例如，这解决了包括多个家庭成员的调查中的隐私问题。

更一般地说，合成和群体隐私不是一回事，第 3.5.2 节(定理 3.20)中的改进合成界限在因子 k 的基础上得到了实质性的改进，即使在 δ = 0 时，也不能为群体隐私带来同样的收益。

2.3.1 差别隐私承诺了什么

经济学观点。差异隐私承诺保护个人免受由于他们的数据在私有数据库 x 中而可能面临的任何额外伤害，如果他们的数据不是 x 的一部分，他们就不会面临这些伤害。尽管一旦差异隐私机制 M 的结果 M(x)被发布，个人确实可能面临伤害，但差异隐私承诺伤害的概率不会因为他们选择参与而显著增加。这是一个非常实用的隐私定义，因为当一个人决定是否将她的数据纳入一个以不同的私人方式使用的数据库时，她考虑的正是这种差异:与她不参与造成伤害的概率相比，她参与造成伤害的概率。她无法控制数据库的其余内容。鉴于差别隐私的承诺，她确信，从未来伤害的角度来看，她应该对参与和不参与几乎无动于衷。如果有任何动机——从利他主义到金钱回报——差异化隐私可能

会说服她允许自己的数据被使用。这种直觉可以在效用理论的意义上被形式化，我们在这里简单描述一下。

考虑一个人，他对我们用 a 表示的所有可能的未来事件有任意的偏好。这些偏好由效用函数 ui 表示:A → R≥0，我们说个人 I 在 a ∈ A 发生的情况下经历效用 ui(a)。假设 x ∈ N|X|是包含个体的数据集是私有数据，M 是 ε-差分私有算法。设 y 是与 x 相同的数据集，除了它不包括个体 I 的数据(特别是 kx-yk1 = 1)，设 f:Range(M)→∞(A)是决定未来事件 A 分布的(任意)函数，条件是机制 M 的输出。通过对差异隐私的保证，以及命题 2.1 所保证的对任意后处理的弹性，我们拥有:

$$\text{ea}\infty f(M(x))[ui(a)]=\text{用户界面(a)} \quad \underset{a\in A}{X} \quad \text{镨} \quad \underset{f(M(x))}{[a]}$$

$$\leqslant X\ ui(a)\ \exp(\varepsilon) \quad \text{镨} \quad \underset{f(M(y))\ a\in A}{[a]}$$

$$= \exp(\varepsilon)Ea\vee f(M(y))[ui(a)]$$

同样的，

$$Ea\infty f(M(x))[ui(a)]\geqslant \exp((\varepsilon)Ea\infty f(M(y))[ui(a)]。$$

因此，通过保证 ε-差分隐私，数据分析师可以向个人保证，他预期的未来效用不会受到超过 exp(ε)∞(1+ε)因素的损害。请注意，这个承诺独立于个人的是效用函数 ui，同时适用于可能具有完全不同效用函数的多个个人。

2.3.2 什么样的差别隐私没有承诺

正如我们在吸烟导致癌症的例子中看到的那样，虽然差别隐私是一个极其强有力的保证，但它并不能保证无条件免受伤害。它也不会在以前不存在的地方创造隐私。更一般地说，差别隐私并不能保证一个人

认为自己的秘密会一直保密。它只是确保一个人对调查的参与本身不会被披露，也不会导致披露他对调查所做的任何具体贡献。从调查中得出的结论很可能反映了个人的统计信息。旨在发现特定疾病早期指标的健康调查可能会产生强有力的、甚至是决定性的结果；这些结论适用于特定的个人，并不是侵犯不同隐私的证据；该个人甚至可能没有参与调查(同样，无论该个人是否参与调查，差异隐私都确保以非常相似的概率获得这些结论性结果)。特别是，如果调查告诉我们，特定的私有属性与可公开观察的属性密切相关，这并不违反差异隐私，因为这种相同的相关性几乎以相同的概率被观察到，而与任何受访者的存在或不存在无关。

差别隐私的定性性质。引入并正式定义了差别隐私后，我们重新总结了它的关键优点。

1. *针对任意风险的保护，超越了针对重新识别的保护。*
2. *自动消除链接攻击，包括所有过去、现在和将来的数据集以及其他形式和来源的辅助信息。*
3. *隐私损失的量化。*差别隐私不是一个二元概念，有隐私损失的衡量标准。这允许不同技术之间的比较:对于隐私丢失的固定界限，哪种技术提供更好的准确性？对于固定精度，哪种技术提供更好的隐私？
4. *作文。*也许最关键的是，损失的量化还允许在多次计算中分析和控制累积的隐私损失。了解组合下差分私有机制的行为，能够从更简单的差分私有构建块中设计和分析复杂的差分私有算法。
5. *群体隐私。*差别隐私允许分析和控制家庭等群体遭受的隐私损失。
6. *后处理下的闭包差分隐私不受后处理的影响:如果没有关于私有数据库的额外知识，数据分析师就无法计算差分私有算法 M 的输出函数，也无法降低其差分私有性。*也就是说，无论是在形

式定义下，甚至是任何直观意义上，数据分析师都不能简单地坐在角落里思考算法的输出，而不管有什么辅助信息可用。

这些是差别隐私的信号属性。我们能证明一个相反的情况吗？也就是说，这些属性或其中的某个子集是否意味着不同的隐私？差别隐私在这些方面能被削弱，还能有意义吗？这些都是开放性的问题。

2.3.3 关于定义的最后评论

隐私的粒度。应仔细审查不同隐私的声明，以确定承诺隐私的粒度级别。差异隐私承诺即使数据库中的单个条目被修改，算法的行为也将大致保持不变。但是什么构成了数据库中的一个条目呢？例如，考虑一个采用图形形式的数据库。这样的数据库可能会对一个社交网络进行编码:每个人都由图中的一个顶点来表示，而人与人之间的友谊则由边来表示。

我们可以在对应于个体的粒度级别上考虑差异隐私:也就是说，我们可以要求差异隐私算法对图中任何顶点的添加或移除不敏感。这提供了强有力的隐私保证，但实际上可能比我们需要的更强。单个顶点的添加或移除毕竟可以在图中添加或移除多达 n 条边。根据我们希望从图中了解到的内容，对 n 边移除不敏感可能是一个无法满足的限制。

另一方面，我们可以在对应于边的粒度级别上考虑差异隐私，并要求我们的算法仅对图中单个或少量边的添加或移除不敏感。这当然是一个较弱的保证，但对于某些目的来说可能仍然足够。非正式地说，如果我们在单个边的水平上承诺 ε-差分隐私，那么任何数据分析师都不应该能够得出关于图中 $1/\varepsilon$ 边的任何子集存在的任何结论。在某些情况下，大量的社会联系可能不被视为敏感信息:例如，一个人可能不觉得有必要隐藏他的大多数联系是与他所在城市或工作场所的个人联系的事实，因为他生活和工作的地方是公共信息。另一方面，可能有少数社交联系人的存在是高度敏感的(例如，潜在的新雇主，或亲密的朋友)。在这种情况下，边缘隐私应该足以保护敏感信息，同时仍然允许比顶点隐私更全面的数据分析。边缘隐私将保护这样一个人的敏感信息，前提是他的朋友少于 $1/\varepsilon$。

作为另一个例子，可以设计不同的私人电影推荐系统，以在单部电影的“事件”级别保护训练集中的数据，隐藏任何单部电影的观看/评级，但不隐藏个人对牛仔西部片或戈尔的热情，或者在个人整个观看和评级历史的“用户”级别。

所有的小精灵都是一样的。当 ε 很小时，(ε，0)-差分隐私断言，对于所有相邻的数据库对 x，y 和所有输出 o，对手不能根据观察 o 来区分哪个是真正的数据库。当ε较小时，未能(ε，0)差分私有不一定是值得警惕的——例如，机制可能是(2ε，0)-差分私有。不同但很小的 epsilons 的隐私保证的性质非常相似。但是大价值是为了什么呢？未能(15，0)-有区别地私有仅仅表示存在相邻的数据库和输出 o，对于输出 o，观察 o 的概率分别取决于数据库是 x 还是 y，这是很大的。o 的输出可能非常不可能(这通过(ε，δ)-差分隐私来解决)；数据库 x 和 y 在“现实世界”中出现可能是非常不自然的；对手可能没有正确的辅助信息来识别已经发生了泄露输出；或者可能对数据库了解不够，无法确定它们的对称差的值。因此，就像一个弱密码系统可能只泄漏从消息的最低有效位到完整解密密钥的任何东西一样，不(ε，0)-或(ε，δ)-不同的私密性可能从实际上无意义的隐私泄露到整个数据库的完全泄露。大 ε 按照它自己的方式是大的。

一些额外的形式。我们的隐私机制 M 会经常取一些辅助参数 w 作为输入，除了数据库 x。例如，w 可以指定数据库 x 上的查询 qw，或者查询的集合 Qw。机制 M(w，x)可能(分别)以对 qw(x)或 qw 中的一些或所有查询的不同私有近似来响应。对于所有 δ ≥ 0，我们说一个机制 M(，)满足(ε，δ)-差分隐私，如果对于每一个 w，M(w)，，满足(ε，δ)-差分隐私。

可以包含在 w 中的参数的另一个例子是安全参数 κ，用于控制 δ = δ(κ)应该有多小。也就是说，对于所有 κ，M(κ)应该是(ε，δ(κ))差分私有的。通常，在本专著中，我们要求 δ 在 κ 中是一个可忽略的函数，即，。 $\delta = \kappa - \omega(1)$. 因此，我们认为 δ 在密码上很小，而 ε 通常被认为是一个适度小的常数。

在辅助参数 w 指定查询集合 Qw = {q : Xn → R}的情况下，我们将机制 M 称为大纲生成器。概要生成器输出一个(有差异的私有)概要 A，它可以用来计算 Qw 中所有查询的答案。也就是说，我们要求存在重构过程 R，使得对于指定查询 qv ∈ Qw 的每个输入 v，重构过程输出 R(A，v) ∈ R。典型地，我们将要求 M 以高概率产生概要 A，使得重建过程使用 A 来计算准确的答案。也就是说，对于所有或大部分(按某种分布加权)的查询 qv ∈ Qw，误差|R(A，v)-qv(x)|将是有界的。我们偶尔会滥用符号，并参考将实际查询 q(而不是它的某种表示 v)作为输入并输出 R(A，q)的重构过程。

提要的一个特例是合成数据库。顾名思义，合成数据库的行与原始数据库的行属于同一类型。合成数据库的一个优点是，它们可以使用分析师在原始数据库上使用的相同软件进行分析，从而不需要特殊的重建程序。

备注 2.1。由于浮点数实现的微妙性，在对实值机制(如拉普拉斯机制)进行编程时必须非常小心。否则，差异隐私会被破坏，因为数据库 x 上具有非零概率的输出，由于舍入，可能在相邻数据库 y 上具有零概率。这只是浮点的实现需要在差异隐私的背景下进行审查的一种方式，并不是唯一的。

2.4 书目注释

差异隐私的定义是由 Dwork 等人提出的。[23];这里和文献中使用的精确公式最早出现在[20]中，这要归功于 Dwork 和 McSherry。“差别隐私”一词是由迈克尔·施罗德创造的。语义安全的不可能性是由于 Dwork 和 Naor [25]。(ε，0)-差分私有机制的组成和组隐私在[23]中首次提出。

2.4。 书目注释

(ε，δ)-差异隐私的构成在[21]中首次提出(但参见附录 B 中的更正证明，由德沃克斯和雷[22]完成)。米罗诺夫观察到差分隐私易受浮点数不适当实现的影响，并提出了一种缓解措施[63]。

3

基本技术和合成定理

在回顾了一些概率工具之后，我们提出了拉普拉斯机制，它为实(向量)值查询提供了差分隐私。这种方法的应用自然会导致指数机制，这是一种从一组离散的候选输出中进行差异私有选择的方法。然后，我们分析由组成多个不同的私有机制所导致的累积隐私损失。最后，我们给出了一种方法——稀疏向量技术——用于私下报告潜在的大量计算的结果，前提是只有少数是“有意义的”。

在本节中，我们将描述一些最基本的差分隐私技术，我们将一次又一次地使用这些技术。这里描述的技术构成了我们将要开发的所有其他算法的基本构件。

3.1 有用的概率工具

以下浓度不等式通常会很有用。我们用易于使用的形式而不是最强烈的形式来陈述它们。

3.2。 随机反应

定理 3.1(加性切尔诺夫界)。设 X1，...,。*Xm 是有界的独立随机变量，使得所有 I 的 0 ⩽Xi ⩽1*。设 S = m1 Pmi=1 Xi 表示它们的平均值，设 E = E[S]表示它们的期望平均值。然后:

$$pr[S > +\varepsilon] \leqslant e\ 2\varepsilon 2$$

$$pr[S < -\varepsilon] \leqslant e\ 2\varepsilon 2$$

定理 3.2(乘法切尔诺夫界)。设 X1，...,。*Xm 是有界的独立随机变量，使得所有 I 的 0 ⩽Xi ⩽1*。设 S = m1 Pmi=1 Xi 表示它们的平均值，设 E = E[S]表示它们的期望平均值。然后:

$$pr[S > (1+\varepsilon)] \leqslant e\text{-}m\varepsilon 2/3$$

$$pr[S < (1\varepsilon)] \leqslant e\text{-}m\varepsilon 2/2$$

当我们没有独立的随机变量时，一切都不会丢失。我们仍然可以应用阿兹马的不平等:

定理 3.3(阿兹马不等式)。设 f 是 m 个随机变量 X1 的函数，...,。*每个 Xi 从集合 Ai 中取值，使得 E[f]有界*。让 ci 表示 Xi 对 f 的最大影响，。对于所有人工智能来说，一个$i' \in A_i$:

$$„X_{i-1}, X_i = a_i] - E[f/X_1, ..., = a_i']| \leq Xi\ 1\ 号，Xi{\cdot}ci$$

然后:

!

Xm) c2i

i=1

定理 3.4(斯特林近似)。*n！*可以近似为

—— √/

2nπ(n/e)n:

—— —— √ √ v

n ^ 1/(12n+1) n 1/(12n)

2nπ(n/e) e < n！ < *2nπ(n/e) e* 。

3.2 随机反应

让我们回忆一下第 2 节中描述的简单随机反应机制，用于评估尴尬或违法的频率

行为。让 XYZ 成为这样的活动。面对“过去一周你在 XYZ 订婚了吗？被告被指示执行以下步骤:

1. 抛硬币。
2. 如果是反面，那就如实回答。
3. 如果正面，再抛第二枚硬币，正面回答“是”，反面回答“否”。

随机化回答背后的直觉是，它提供了“似是而非的否认”。例如，可能会给出“是”的回答，因为第一次和第二次抛硬币都是正面，概率为 1/4.换句话说，隐私是通过过程获得的，没有“好”或“坏”的回应。获得答复的过程会影响对答复的合理解释。正如下一个声明所显示的，随机反应是不同的隐私。

权利要求 3.5。上述随机化回答的版本是
(ln，0)-差分私有。

*证据。*搞定一个回答者。一个案例分析表明，Pr[Response = Yes|Truth = Yes] = 3/4.具体来说，当真值为“是”时，如果第一枚硬币正面朝上(概率 1/2)或第一枚和第二枚正面朝上(概率 1/4)，结果将为“是”，而 Pr[Response = Yes | true = No]= 1/4(第一枚正面朝上，第二枚反面朝上；概率 1/4)。将类似的推理应用于“否”答案的情况，我们得到:

$$\frac{\text{公关[回应=是|真相=是]}}{\text{公关[回应=是|真相=否]}} = \frac{3/4}{1/4} = 3.$$

公关[回应=否|真相=否]
公关[回应=否|真相=是]

实数。决定我们能多准确地回答这类问题的一个重要参数是它们的灵敏度:

定义 3.1(` 1-敏感性)。函数 f 的 1-灵敏度:
N|X| → Rk 为:

$$\Delta f = \max_{x,\, y \in N|X|} KF(x)f(y)k1。$$

kx yk1 = 1

函数 f 的灵敏度 1 捕捉了在最坏的情况下单个个体的数据可以改变函数 f 的幅度，因此，直观地，我们必须引入响应中的不确定性，以便隐藏单个个体的参与。事实上，我们将把这种直觉形式化:一个函数的灵敏度给出了一个上限，即为了保护隐私，我们必须扰动它的输出多少。一种噪声分布自然会导致不同的隐私。

定义 3.2(拉普拉斯分布)。标度为 b 的拉普拉斯分布(以 0 为中心)是概率密度函数分布:

一 |x|
__ __搭接(x|b) = / exp − / 。
2b b

这个分布的方差是 σ2 = 2b2。我们有时会写 Lap(b)来表示带有标度 b 的拉普拉斯分布，有时会滥用符号并简单地写 Lap(b)来表示随机变量 X ∞Lap(b)。

拉普拉斯分布是指数分布的对称形式。

我们现在将定义拉普拉斯机制。顾名思义，拉普拉斯机制将简单地计算 f，并用从拉普拉斯分布中提取的噪声扰动每个坐标。噪声的比例将被校准到 f 的灵敏度(除以 ε)。定义 3.3(拉普拉斯机制)。给定任何函数 f：N|X| → Rk，拉普拉斯机制定义为:

Yk)

易的身份证在哪里。从 Lap(↭f/ε)中提取的随机变量。

定理 3.6。拉普拉斯机制保留了(ε，0)-差分隐私。

*证据。*设 x ∈ N|X|和 y ∈ N|X|为 kx yk1⩽1，设 f()为某种函数 f : N|X| → Rk。设 px 表示 ML(x，f，ε)的概率密度函数，设 py 表示 ML(y，f，ε)的概率密度函数。我们在任意点 z ∈ Rk 比较两者

□*exp(−ε|f(xΔ)fi−zi|)*□

_____ / =

py(z)i=1 □exp(−ε|f(yΔ)fi−zi|)□

_____

_____

_____

_____约克ε(| f(y)I zi | | f(x)I zi |)

= 经历

f

i=1

约克 ε| f(x)I f(y)I |

_____≤经历 /

f

i=1

$$= \exp\left(\cdot \|f(x) \quad f(y)\ _1 \right.$$

ε – k

f

≤*exp(ε)*，

其中第一个不等式源自三角形不等式，最后一个源自灵敏度的定义以及 kx yk1≤1 的事实。□p/pxy((ZZ))≥exp((ε)遵循对称性。□

示例 3.1(计数查询)。计数查询是“数据库中有多少元素满足属性 P？我们将一次又一次地返回这些查询，有时以这种纯形式，有时以分数形式(“数据库中元素的分数是多少...?")。有时使用权重(线性查询)，有时使用稍微复杂的形式(例如，。将 h : N|X| → [0，1]应用于数据库中的每个元素，并对结果求和)。计数是一种极其强大的原始。它捕获了统计查询学习模型中所有可学习的内容，以及许多标准的数据挖掘任务和基本统计信息。由于计数查询的灵敏度为 1(单个个体的添加或删除最多可以改变 1 个计数)，定理 3.6 的直接结果是，通过添加缩放到 1/ε 的噪声，即通过添加从 Lap(1/ε)提取的噪声，可以为计数查询实现(ε，0)差分隐私。预期失真或误差为 1/ε，与数据库大小无关。

m 计数查询的固定但任意的列表可以被视为向量值查询。在没有关于查询集合的任何进一步信息的情况下，这个向量值查询的敏感度的最坏情况界限是 m，因为单个个体可能会改变每个计数。在这种情况下(ε，0)-可以通过向每个查询的真实答案添加缩放到 m/ε 的噪声来实现差异隐私。

我们有时将响应大量(可能是任意的)查询的问题称为查询释放问题。

示例 3.2(直方图查询)。在特殊(但常见)的情况下，查询在结构上是不相交的，我们可以做得更好——我们不必让噪声随着查询的数量而增加。直方图查询就是一个例子。在这种类型的查询中，宇宙 N|X|被划分成单元，查询询问每个单元中有多少数据库元素。由于单元格是不相交的，单个数据库元素的添加或移除会影响恰好一个单元格中的计数，并且与该单元格的差值以 1 为界，因此直方图查询的敏感度为 1，并且可以通过将 Lap(1/ε)中的独立绘制添加到每个单元格中的真实计数来回答。

为了理解拉普拉斯机制对于一般查询的准确性，我们使用以下有用的事实:

事实 3.7。如果 Y∞搭接(b)，则:

pr[| Y |≥t b]= exp(t)。

这个事实，连同一个并界，给了我们一个拉普拉斯机制精度的简单界:

定理 3.8。设 f : N|X| → Rk，设 y = ML(x，f()，ε)。那么∀δ∑(0，1):

k∏f

$$(x) - y\|_\infty \geq \ln \quad \frac{}{\delta} \quad \cdot \quad \frac{}{\varepsilon} \quad \leq \delta$$ 复兴信贷银行

*证据。*我们有:

k∏f " k∏f #

– —— – —— Pr kf= Pr max|Yi| ≥ ln·

I∏k δ ε

k∏f

_ ——≤ k Pr |Yi| ≥ ln

δ ε

δ

_ = k /

k

= δ

其中倒数第二个不等式来自于每个 Yi∨Lap(↮f/ε)和 Fact 3.7。□

示例 3.3(名字)。假设我们希望从一万个潜在名字的列表中计算出哪些名字在 2010 年人口普查参与者中最常见。这个问题可以表示为查询 f : N|X| → R10000.这是一个直方图查询，所以灵敏度∏f = 1，因为每个人最多只能有一个名字。利用上述定理，我们看到，我们可以同时计算所有 10，000 个具有(1，0)-差分隐私的名称的频率，并且在概率为 95%的情况下，任何估计都不会偏离超过 ln(10000/.05) ≈ 12.2 的加性误差.对于一个拥有 300，000，000 人口的国家来说，这是一个相当低的误差！

不同的私人选择。示例 3.3 中的任务是一个不同的私人选择:结果空间是离散的，任务是产生一个“最佳”答案，在这种情况下是人口最多的直方图单元。

示例 3.4(最常见的医疗状况)。假设我们希望知道在一组受访者的病史中哪种疾病(大约)最常见，那么这组问题就是，对于所考虑的每种疾病，该个体是否曾经接受过这种疾病的诊断。由于个人可以经历许多情况，这组问题的敏感性可能很高。尽管如此，正如我们接下来所描述的，这个任务可以通过在每个计数中添加 Lap(1/ε)噪声来解决(注意噪声的小尺度，它与条件的总数无关)。至关重要的是，m 嘈杂计数本身不会被释放(尽管“获胜”计数可以在没有额外隐私成本的情况下被释放)。

报告噪音最大。考虑以下简单算法来确定 m 个计数查询中哪个具有最高值:将独立生成的拉普拉斯噪声 Lap(1/ε)添加到每个计数中，并返回最大噪声计数的索引(我们忽略平局的可能性)。将此算法称为报告最大噪声。

请注意“报告噪声最大值”算法中起作用的“信息最小化”原则:不是释放所有噪声计数并允许分析师找到最大值及其索引，而是仅公开与最大值对应的索引。因为一个人的数据可以影响所有的计数，计数的向量有很高的灵敏度，特别是 f = m，如果我们想用拉普拉斯机制释放所有的计数，就需要更多的噪声。

权利要求 3.9。报告最大噪声算法是(ε，0)-差分私有。

*证据。*固定 D = D0 ∨{ a }。设 c 分别为 c0，表示数据库为 D 时的计数向量，分别为 D0.我们使用两个属性:

1. *计数的单调性。*对所有$\in [m], c_j \geq c'_j$人来说；和
2. *利普希茨物业。*对所有的$\in [m], 1 + c'_j \geq c_j$来说。

修复任何 I∏m。我们将从上面和下面限制我被 D 和 D0 选中的概率比。

固定 R1，从[Lap(1/ε)]m1 中抽取，用于除第 I 个计数以外的所有噪声计数。我们将独立地为每个 r-I 进行辩论。我们使用符号 Pr[i|ξ]来表示报告噪声最大算法的输出是 I 的概率，以 ξ 为条件。

我们首先论证 Pr[i|D，r-I]≤eεPr[I | D0，r-I]。规定

$$r\infty = \min:ci+ri > CJ+rj \ \forall j \ 6 = I。$$

(美国)罗得岛州(Rhode Island),剩余收入(residual income)

请注意，在固定 ri 的情况下，当且仅当 ri≥r∞时，当数据库为 D 时，I 将是输出(argmax 噪声计数)。

对于所有 1 ⩽ j 6= i ⩽ m，我们有:

$$ci+r\infty > CJ+rj$$

$$(1 + c_i') + r^* \geq c_i + r^* > c_j + r_j \geq c_j' + \text{rj}$$
$$_i' + (r^* + 1) > c_j' + r_j 。$$

因此，如果 ri≥r∞+1，那么当数据库为 D0，噪声矢量为(ri，ri)时，ith 计数将是最大值。下面的概率超过了 ri∞Lap(1/ε)的选择。

Pr[ri≥1+r ∫]≥eεPr[ri≥r ∫]= eεPr[I | D，r I]≥Pr[I | D0，r I]≥Pr[ri≥1+r ∫]≥eεPr[ri≥r ∫]= eεPr[I | D，r I]，乘以 eε 后，得出我们想要显示的结果:

Pr[i|D，r-I]≤eεPr[I | D0，r-I]。

我们现在认为 Pr[i|D0，r-I]≤eεPr[I | D，r-I]。规定

$$= \min : c_i' + r_i > c_j' + r_j \ \forall j \neq \text{一世}。$$

(美国)罗得岛州(Rhode Island),剩余收入(residual income)

请注意，在固定 ri 的情况下，当且仅当 ri≥r∞时，当数据库为 D0 时，I 将是输出(argmax 噪声计数)。

对于所有 1 ≤ j 6= i ≤ m，我们有:

$$_i' + r^* > c_j' + \text{河}$$
$$1 + c_i' + r^* > 1 + c_j' + \text{rj}$$
$$\text{c}^{\prime}_i + (r^* + 1) > (1 + c_j') + \text{rj}$$
$$\text{河} + (r^* + 1) \geq c_i' + (r^* + 1) > (1 + c_j') + r_j \geq c_j + r_j 。$$

因此，如果 ri≥r∞+1，那么 I 将是具有随机性(ri，ri)的数据库 D 的输出(argmax 噪声计数)。因此，随着概率取代了 ri 的选择:

Pr[i|D，r I]≥Pr[ri≥r ∫ +1]≥eεPr[ri≥r ∫]= eεPr[I | D0，r I]，

在乘以 ε 后，得出我们想要展示的结果:

Pr[i|D0，r-I]≤eεPr[I | D，r-I]。□

3.4 指数机制

在“最常见的名称”和“最常见的状况”的例子中，我们使用拉普拉斯噪声估计计数，并报告噪声最大值。在这两个例子中，响应的效用与产生的噪声值直接相关；也就是说，名称或条件的受欢迎程度是在与噪声大小相同的尺度和单位下适当测量的。

指数机制是为我们希望选择“最佳”响应的情况而设计的，但是在计算量中直接添加噪声会完全破坏其价值，例如在拍卖中设置价格，目标是最大化收入，而在最佳价格中添加少量正噪声(以保护出价的隐私)会显著减少最终收入。

例 3.5(南瓜。假设我们有丰富的南瓜供应和四个投标人:A，F，I，K，其中 A，F，I 每个投标 1.00 美元，K 投标 3.01 美元.最优价格是多少？在 3.01 美元时，收入为 3.01 美元，在 3.00 美元时，收入为 1.00 美元，但在 3.02 美元时，收入为零！

指数机制是用任意实用程序(和任意非数字范围)回答查询的自然构件，同时保留了差异隐私。给定任意范围 R，指数机制是相对于某种效用函数 u：N|X| × R → R 来定义的，它将数据库/输出对映射到效用得分。直观地说，对于一个固定的数据库 x，用户更喜欢该机制输出一些具有最大可能效用分数的 R 元素。注意，当我们谈论效用得分 u：N|X| × R → R 的敏感度时，我们只关心 u 相对于其数据库参数的敏感度；它的范围参数可以任意敏感:

$$\Delta u \equiv \max_{r \in \mathcal{R}} \max_{x,y:\|x-y\|_1 \leq 1} |u(x\ ,\ r)-u(y\ ,\ r)|。$$

指数机制背后的直觉是以与 exp(εu(x，R)/∈u)成比例的概率输出每个可能的 r ∈ R，因此隐私损失大约为:

$$\ln \frac{\exp(\varepsilon u(x,r)/\Delta u)}{\exp(\varepsilon u(y,r)/\Delta u)} = \varepsilon[u(x,r)-u(y,r)]/\Delta u) \leq \varepsilon.$$

这种直观的观点忽略了规范化术语的一些影响，当数据库中增加一个人导致一些元素 r ∈ R 的效用降低而另一些元素增加时，就会出现这种影响。接下来定义的实际机制保留了隐私预算的一半，用于规范化条款的更改。

定义 3.4(指数机制)。指数机制 ME(x，u，R)选择并输出一个元素 r ∈ R ——概率正比于 exp(εu/2∞(x，ru))。

指数机制可以在大的任意域上定义复杂的分布，因此当问题的自然参数中 u 的范围极大时，可能无法有效地实现指数机制。

回到南瓜的例子，数据库 x 上的价格 p 的效用只是当价格为 p 并且需求曲线如 x 所描述时获得的利润。潜在价格的范围独立于实际出价是很重要的。否则，在一个数据集中存在非零权重的价格，而在相邻数据集中存在零权重的价格，这违反了差别隐私。

定理 3.10。指数机制保留了(ε，0)差分隐私。

*证据。*为了清楚起见，我们假设指数机制的范围 R 是有限的，但这不是必须的。如同所有不同的隐私证明一样，我们考虑实例化的概率比 *3.4。 指数机制*

在两个相邻的数据库 x ∈ N|X|和 y ∈ N|X|上输出一些元素 r ∈ R(即。kx yk1⩽1)。

$$\frac{\Pr[\mathcal{M}_E(x,u,R)=r]}{\Pr[\mathcal{M}_E(y,u,R)=r]} = \frac{\frac{\exp(\frac{\varepsilon u(x,r)}{2\Delta u})}{\sum_{r^0\in R}\exp(\frac{\varepsilon u(x,r^0)}{2\Delta u})}}{\frac{\exp(\frac{\varepsilon u(y,r)}{2\Delta u})}{\sum_{r^0\in R}\exp(\frac{\varepsilon u(y,r^0)}{2\Delta u})}}$$

$$= \frac{\exp(\frac{\varepsilon u(x,r)}{2\Delta u})}{\exp(\frac{\varepsilon u(y,r)}{2\Delta u})} \cdot \frac{\sum_{r^0\in R}\exp(\frac{\varepsilon u(y,r^0)}{2\Delta u})}{\sum_{r^0\in R}\exp(\frac{\varepsilon u(x,r^0)}{2\Delta u})}$$

$$= \exp\frac{\varepsilon(u(x,r^0)-u(y,r^0))}{2\Delta u}$$

$$\frac{\qquad}{\qquad} \quad \text{ppr0∈rexp(exp(εuεu2Δ2Δ((x,ry,ruu00))))}$$

$$r0\in R$$

$$\text{P exp(εu2Δ(x,ru0))}$$

$$\varepsilon \qquad \varepsilon \qquad r0\in R$$

$$\leq \exp\frac{}{2}\cdot\exp\frac{}{2}\cdot \text{Pr0∈R exp(εu2Δ(x,ru0))}$$

$$= \exp(\varepsilon)。$$

_____类似地，Pr[/Pr[MMEE((x，uy，u)= RR]]≥exp(-ε)由对称性决定。□

指数机制通常可以提供强大的效用保证，因为当结果的质量分数下降时，它会以指数方式迅速降低结果。对于给定的数据库 X 和给定的效用测度 u : N|X| × R → R，让 OPTu(x) = maxr∈R u(x，R)表示任意元素 r∈R 相对于数据库 X 的最大效用得分。我们将限制指数机制返回 R 的“好”元素的概率，其中好将以 OPTu(x)来衡量。结果是，返回的元素 R 的效用分数比 OPTu(x)差 0 倍以上的可能性极小(∏u/ε)log | R |)。

定理 3.11。修复一个数据库 x，让 ROPT = {r ∈ R : u(x，r) =

OPTu(x)}表示 R 中获得效用得分 OPTu(x)的元素集。然后:

—— ——— ME(x，u，R))≤OPTu(x)2∏u ln | R |+t≤e-t

Pr u(

ε |ROPT

证据。

——————————— /

R | exp(εc/2∏u)Pr[u(ME(x，u，R)) ≤c] ≤/

| ROPT | exp(εOptu(x)/2∏u)

$$= \frac{|\mathcal{R}|}{\exp\left(\varepsilon(c - \mathrm{OPT}_u \right)_{(十)}}$$

——— ——— / / 。

|ROPT 2∏u

不等式来自于观察到每个 r ∈ R 与 u(x，r) ≤ c 最多有未归一化的概率质量 exp(εc/2∏u)，因此这种“坏”元素 R 的整个集合最多有总的未归一化的概率质量| R | exp(εc/2∏u)。相比之下，我们知道至少存在|ROPT| ≥ 1 个 u(x，r) = OPTu(x)的元素，因此存在非归一化概率质量 exp(εOPTu(x)/2∏u)，因此这是归一化项的下界。

该定理源于插入 c 的适当值。□

由于我们总是有|ROPT| ≥ 1，我们可以更普遍地利用下面的简单推论:

推论 3.12。修复数据库 x，我们有:

$$(x) - \frac{\ln}{\varepsilon}$$ ME(x，u，R))≤OPTu 2∏u(ln(| R |)+t)≤e t Pr u(

从定理 3.11 和推论 3.12 的证明中可以看出，指数机制特别容易分析。

示例 3.6(两者中的最佳)。考虑一个简单的问题，即确定两种医学状况 A 和 B 中哪一种更常见。假设两个真实计数对于条件 A 为 0，对于条件 b 为 c > 0。我们的效用概念将与实际计数联系在一起，因此计数越大，效用越高，u = 1.因此，A 的效用为 0，B 的效用为 c。使用指数机制

我们可以立即应用推论 3.12，看到观察(错误)结果 A 的概率最多为 2e-c(ε/(2∏u))= 2e-cε/2。

分析报告噪声最大值似乎更复杂，因为它需要了解当加到 A 计数上的噪声为正，加到 B 计数上的噪声为负时(概率为 1/4)的情况下会发生什么。

如果向数据集中添加元素不会导致函数值降低，则函数在数据集中是单调的。计数查询是单调的；向一群买家提供固定价格获得的收入也是如此。

考虑“报告单侧噪声 arg-max”机制，在单调效用的情况下，该机制将噪声添加到从参数为 ε/∏u 的单侧指数分布提取的每个潜在输出的效用中，或者在非单调效用的情况下，添加到参数为 ε/2∏u 的效用中，并报告结果 Arg-Max。

利用这种算法，其隐私证明几乎与 Report Noisy Max 的相同(但是当效用是非单调时损失了因子 2)，我们立即在上面的示例 3.6 中获得结果 A 在 c 中是指数的(ε/∏u)= cε 比结果 b 更不可能被选择。

定理 3.13。当参数 ε/2∏u 为-differential private 时，报告单侧噪声参数-Max。

备注 3.1。当用拉普拉斯噪声或指数噪声实例化时，报告噪声最大值与指数机制具有相似的保证，但导致不同的分布。事实证明，用 Gumbel 分布实例化报告噪声最大值会产生一种算法，该算法从指数机制分布中精确采样。这个事实是机器学习中的民间传说，被称为“甘贝尔最大诡计”。

3.5 合成定理

现在，我们已经有了几个构建模块来设计不同的私有算法，了解如何将它们结合起来设计更复杂的算法是很重要的。为了使用这些工具，我们希望两个差分私有算法的组合本身是差分私有的。事实上，正如我们将看到的，情况就是这样。当然，参数 ε 和 δ 必然会降级——考虑使用拉普拉斯机制重复计算相同的统计量，每次缩放以给出 ε-差分隐

私。该机制的每个实例给出的答案的平均值最终会收敛到统计数据的真实值，因此我们无法避免我们的隐私保证的强度会随着重复使用而降低。在这一节中，我们给出了一些定理，说明了当差分专用子程序被组合时，参数 ε 和 δ 是如何组成的。

让我们首先从一个简单的热身开始:我们将看到(ε1，0)-差分私有算法和(ε2，0)-差分私有算法的独立使用，当放在一起时，是(ε1 + ε2，0)-差分私有的。

定理 3.14。设 M1 : N|X| → R1 为 ε1-差分私有算法，M2 : N|X| → R2 为 ε2-差分私有算法。那么它们的组合，通过映射定义为 M1，2:N | X |→ R1×R2:m1，2(x) = (M1(x)，M2(x))就是 ε1+ε2-差分私有。

*证据。*设 X，y ∈ N|X|为 kx yk1≤1.修正任意(r1，r2) ∈ R1 × R2.然后:

$$\frac{\Pr[M1，2(x) = (r1，r2)]}{\Pr[M1，2(y) = (r1，r2)]} = \frac{\Pr[M1(x) = r1]\Pr[M2(x) = r2]}{\Pr[M1(y) = r1]\Pr[M2(y) = r2]}$$

$$\Pr[M1(x) = r1]\Pr[M2(x) = r1] = \Pr[M1(y) = r1] \Big/ \Pr[M2(y) = r1]$$

$$\leqslant exp(\varepsilon 1)exp(\varepsilon 2)$$

$$= \exp(\varepsilon 1 + \varepsilon 2)$$

$\frac{\Pr[M1，2(x)=(r1，r2)]}{\Pr[M1，2(y)=(r1，R2)]}$根据对称性，≥exp((ε1+ε2))。□

合成定理可以反复应用，得到如下推论:

推论 3.15。设 Mi : N|X| → Ri 为 I∏k 的(εi，0)-差分私有算法。那么如果 M[k] : N|X| → Qki=1 Ri 定义为

Mk(x))，那么 M[k]就是(Pki=1 εi，0)-差分私有。

这个定理推广到(ε，δ)-微分隐私的证明见附录 B:

定理 3.16。设 Mi：N|X| → Ri 是 IΠk 的(εi，δI)-差分私有算法。那么如果 M[k]：N|X| → Qki=1 Ri 定义为 M[k](x) = (M1(x)，...,。*Mk(x))，那么 M[k]就是(Pki=1 εi，Pki=1 δi)差分私有。*

合成是"自动"的，这是差异隐私的一个优点，因为数据库管理员无需特别努力就能获得界限。

3.5.1 组成:一些技术细节

在本节的剩余部分，我们将证明一个更复杂的合成定理。为此，我们需要一些定义和引理，根据分布之间的距离度量来重新表述差分隐私。在下面的分数量中，如果分母为零，那么我们定义分数的值为无穷大(记数器将始终为正)。

定义 3.5(KL-散度)。从同一区域取值的两个随机变量 Y 和 Z 之间的 KL-散度或相对熵定义为:

$$D(Y\|Z) = \mathbb{E}_{y\sim Y}\left[\ln \frac{\Pr[Y=y]}{\Pr[Z=y]}\right].$$

已知 D(Y kZ) ⩾ 0，当且仅当 Y 和 Z 同分布时等式成立。然而，D 不是对称的，不满足三角不等式，甚至可以是无限的，特别是当 Supp(Y)不包含在 Supp(Z)中时。

定义 3.6(最大散度)。从同一域取值的两个随机变量 Y 和 Z 之间的最大散度定义为:

$$D_\infty(Y\|Z) = \max_{S\subseteq \mathrm{Supp}(Y)} \left[\ln \frac{\Pr[Y\in S]}{\Pr[Z\in S]}\right].$$

Y 和 Z 之间的 δ-近似最大散度定义为:

$$D_\infty^{\delta}(Y\|Z) = \max_{S\subseteq \mathrm{Supp}(Y),\ \Pr[Y\in S]>\delta} \left[\ln \frac{\Pr[Y\in S]-\delta}{\Pr[Z\in S]}\right].$$

S⊆Supp(Y):Pr[Y ∈S]≥ δ　　Pr[Z ∈ S]

备注 3.2。注意，机制 M 是

1. *ε-差分私有当且仅当在每两个相邻数据库 x 和 y 上，D∞(M(x)kM(y)) ≤ε 和 D∞(M(y)Km(x))≤ε；而且是*
2. (ε，δ)-在每两个相邻的数据库 x，y 上有差别的私有当且仅当:D｜δ(M(x)Km(y))≤ ε 和 D｜δ(M(y)k M(x))≤ ε 。

另一个有用的距离度量是两个随机变量 Y 和 Z 之间的统计距离，定义为

$$\sum(Y，Z) = \overset{极好的}{\underset{S}{最大}}| Pr[Y∈S]—Pr[Z∈S]|。$$

我们说当∑(Y，Z) ≤ δ 时，Y 和 Z 是 δ-闭的。

根据精确的最大散度和统计距离，我们将使用近似最大散度的以下表述:

引理 3.17。

1. *D｜δ(Y kZ)≤ε 当且仅当存在随机变量 Y^0，使得∞(Y，Y^0)≤δ 且 D｜(Y^0kZ)≤ε 。*
2. 当且仅当存在随机变量 Y^0，Z0 使得∞(Y，Y^0)≤ δ/(eε+1)、∞(Z，Z0) ≤ δ/(eε + 1)和 D∞(Y^0kz 0)≤ ε 时，我们同时具有 D｜δ(Y kZ)≤ ε 和 D｜δ(ZkY)≤ ε 。

证据。对于第 1 部分，假设存在 Y^0δ-接近 Y，这样 *D∞(Y kZ) ≤ε*。那么对于每一个 S，

Pr[Y ∈ S] ≤ Pr[Y 0 ∈ S] + δ ≤ eε Pr[Z ∈ S] + δ，进而

D｜δ(Y kZ)≤ ε 。

反之，假设 D | δ(Y kZ)≤ ε 。让 S = {y : Pr[Y = y] > eε Pr[Z = y]}。然后

x(Pr[Y = Y]—eεPr[Z = Y])= Pr[Y∈S]—eεPr[Z∈S]≤ δ 。

y∈S

而且，如果我们让 T = {y : Pr[Y = y] < Pr[Z = y]}，那么我们有

X X
(Pr[Z = Y]—Pr[Y = Y])=(Pr[Y = Y]—Pr[Z = Y])
y∈Ty/∈T

≥X(Pr[Y = Y]—Pr[Z = Y])
y∈S

≥X(Pr[Y = Y]—εPr[Z = Y])/
y∈S

因此，我们可以通过降低 S 上的概率和提高 T 上的概率来从 Y 中获得 Y 0，以满足:

1. 对于所有 y ∈ S，Pr[Y 0 = y] = eε Pr[Z = y] < Pr[Y = y]。
2. 对于所有 y ∈ T，Pr[Y = y] ≤ Pr[Y 0 = y] ≤ Pr[Z = y]。
3. 对于所有 y /∈ S ∪ T，Pr[Y 0 = y] = Pr[Y = y] ≤ eε Pr[Z = y]。

然后经检验 D∞(Y 0kZ) ≤ ε，和

∑(Y，Y ^ 0)= Pr[Y∈S]—Pr[Y ^ 0∈S]= Pr[Y∈S]—eεPr[Z∈S]≤ δ 。

我们现在证明第二部分。假设存在随机变量 Y 0 和 Z0。然后，对于每一组 S，

Pr[Y ∈ S] ≤ Pr[Y 0 ∈ S] + δ
$e^{\varepsilon} + 1$
$\frac{\delta}{/e\varepsilon + 1} \leq e\varepsilon\ Pr[Z0 \in S] +$

$$\leqslant e\varepsilon \Pr[Z \in S] + \frac{\delta}{e\varepsilon + 1}\Big) + \frac{\delta}{e\varepsilon + 1}$$

$$= e\varepsilon \Pr[Z \in S] + \delta。$$

因此，D∠δ(Y kZ)≤ ε ，并且通过对称性，D∠δ(ZkY)≤ ε 。

相反，给定 Y 和 Z，使得 D｜δ(Y kZ)≤ ε 和 D｜δ(ZkY)≤ ε ，我们类似地进行第 1 部分。但是，我们不是简单地降低 S 上 Y 的概率质量来获得 Y 0，消除与 eε Z 的差距，而是也增加了 S 上 Z 的概率质量。具体来说，对于每个 y ∈ S，我们将采取

$$\Pr[Y\,0 = y] = e\varepsilon \Pr[Z0 = y]$$

$$\frac{e\varepsilon}{1 + e\varepsilon} = e\varepsilon / (\Pr[Y = y] + \Pr[Z = y])$$

$$\sum n/e\varepsilon Pr[Z = Y]，\ Pr[Y = y]]。$$

这也意味着对于 y ∈ S，我们有:

$$\frac{\Pr[Y = Y] — \Pr[Y\,0 = Y]}{} = \Pr[Z0 = Y]\Pr[Z = Y]\Pr[/Y = Y]e\varepsilon Pr[Z = Y]，\ e\varepsilon + 1$$

因此

$$\text{def } X \quad 0\alpha = \sum_{y \in S} \Pr[Y = Y] — \Pr[Y = Y]$$

$$= \sum_{y \in S} \Pr[Z0 = y] — \Pr[Z = y]$$

$$\Pr[Y \in S]e\varepsilon Pr[Z \in S]$$

$$= \frac{\delta}{e^{\varepsilon}+1} \leqslant \delta \cdot e\varepsilon + 1$$

类似地，在集合 S0 = {y : Pr[Z = y] > eε Pr[Y = y]}上，我们可以减少 Z 的概率质量，增加 Y 的概率质量，总共增加一些 α0 ⩽ δ/(eε + 1)，这样对于每个 y ∈ S0，我们都有 Pr[Z0 = y] = eε Pr[Y 0 = y]。

如果 α = α0，那么我们可以对所有的 y /∈ S ∪ S0 取 Pr[Z0 = y] = Pr[Z = y]和 Pr[Y 0 = y] = Pr[Y = y]，给出 D∈(Y kZ)⩽ ε 和∈(Y，Y 0)=∈(Z，Z0) = α。如果 α 6= α0，假设 α > α0，那么我们仍然需要在 S ∪ S0 之外的点上增加 Y 0 的概率质量，并减少 Z0 的质量，总共为 β=α-α0，以确保概率和为 1.也就是说，如果我们试图取上面定义的“质量函数”Pr[Y 0 = y]和 Pr[Z0 = y]，那么虽然我们确实有这样的性质，即对于每个 Y，Pr[Y 0 = y] ⩽ eε Pr[Z0 = y]和 Pr[Z0 = y] ⩽ eε Pr[Y 0 = y]，我们也有 Py Pr[Y 0 = Y]= 1β 和 Py Pr[Z0 = y] = 1 + β。然而，这意味着如果我们让 R = {y : Pr[Y 0 = y] < Pr[Z0 = y]}，那么

$$\Pr[Y' = y]) \geq \text{X}^{\text{公关}}[\text{Z0} = \text{y}]^{\text{公关}}[\text{Y X}^{\text{公关}}[\text{Z0} = \text{y}]^{\text{公关}}[\text{Y 0} = \text{y}] = 2\beta\text{。}$$

$y \in R$ y

因此，我们可以将 R 中点上的 Y ^ 0 的概率质量总共增加 β，将 R 中点上的 Z0 的概率质量总共减少 β，同时保留对于所有 y ∈ R，Pr[Y ^ 0 = Y] ⩽Pr[Z0 = Y]的性质。得到的 Y ^ 0 和 Z0 具有我们想要的性质:D∞(Y ^ 0，Z0) ⩽ ε 和∞(Y，Y ^ 0)，∞(Z，Z0) ⩽ α。□

引理 3.18。假设随机变量 Y 和 Z 满足 D∞(Y kZ) ⩽ ε，D∞(ZkY) ⩽ ε。那么 D(Y kZ)⩽ ε (eε1)。

*证据。*我们知道对于任何 Y 和 Z，都是 D(Y kZ) ⩾ 0 的情况

(通过“对数和不等式”)，因此它足以约束 D(Y kZ) + D(ZkY)。我们得到:

D(Y kZ) ≤ D(Y kZ) + D(ZkY)

$$X \quad \Pr[Y = y] \quad \Pr[Z = y]$$

$$\frac{\Pr[Y = y]}{\Pr[Z = y]} \quad \frac{\Pr[Z = y]}{\Pr[Y = y]} = \Pr[Y = y] \ln \quad +年$$

$$\Pr[Z = Y]+(\Pr[Z = Y]—\Pr[Y = Y])\ln/ \quad \frac{\Pr[Z = y]}{\Pr[Y = y]}$$

$$\leq X[0+| \Pr[Z = Y]—\Pr[Y = Y]|\varepsilon]$$

$$= \varepsilon X[\max\{\Pr[Y = y], \Pr[Z = y]\}$$

—最小值{Pr[Y = y]，Pr[Z = y]}]

$$\leq \varepsilon X[(e\varepsilon 1)\min\{\Pr[Y = Y], \Pr[Z = y]\}]$$

≤ ε (eε1)。□

引理 3.19(阿兹马不等式)。让 C1，...,。*Ck 是实值随机变量，使得对于每一个 I江k]，Pr[|Ci| ≤ α] = 1，对于每一个(c1，...,。ci-1)∈ Supp(C1，...,。ci-1)，我们有*

$$\text{ci 1 = ci 1]} \leq \beta 。$$

那么对于每个 z > 0，我们有

$$\_ \text{“k} \quad \surd / \# \quad 2$$

$$\Pr XCi > k\beta + z \quad k\alpha \leq e\ z/2。$$

i=1

3.5.2 高级作曲

除了让参数降级更慢之外，我们希望我们的定理能够处理更复杂的合成形式。然而，在我们开始之前，我们必须讨论一下我们所说的作文到底是什么意思。我们希望我们的定义涵盖以下两个有趣的场景:

1. 在同一数据库上重复使用不同的私有算法。这既允许多次重复使用相同的机制，也允许从任意私有构建块中模块化构建不同的私有算法。
2. 在不同的数据库上重复使用不同的私有算法，尽管这些数据库可能包含与同一个人相关的信息。这使我们能够推断出一个人的累积隐私损失，他的数据可能分布在多个数据集上，每个数据集可能以不同的隐私方式独立使用。由于新数据库一直在创建，而对手实际上可能会影响这些新数据库的构成，这与重复查询单个固定数据库是一个根本不同的问题。

我们希望对合成进行建模，其中对手可以自适应地影响输入到未来机制的数据库，以及对这些机制的查询。让 F 成为一族数据库访问机制。(例如，F 可以是所有 ε-差分私有机制的集合。对于概率对手 A，我们考虑两个实验，实验 0 和实验 1，定义如下。

家庭 F 和对手 A 的实验 b:

k:

1. *a 输出两个相邻的数据库 x0i 和 x1i，一个机制 Mi ∈ F 和参数 wi。*
2. *a 接收 yi ∈R Mi(wi，xi，b)。*

我们允许上面的对手 A 在整个实验中是有状态的，因此它可以根据先前机制的输出自适应地选择数据库、机制和参数。我们把 A 对实验的看法定义为 A 的掷硬币和所有的机械输出(y1，...,。*yk)*。(xji、Mi 和 wi 都可以在此基础上重建。

出于直觉，考虑一个对手，他总是选择 x0i 来保存 Bob 的数据，而 x1i 的不同之处仅在于 Bob 的数据被删除了。那么实验 0 可以被认为是“真实世界”，鲍勃允许他的数据在许多数据发布中使用，而实验 1 是一个“理想世界”，这些数据发布的结果不依赖于鲍勃的数据。我们对隐私的定义仍然要求这两个实验彼此“接近”，就像差异隐私的定义所要求的那样。对 Bob 的直观保证是，考虑到所有 k 机制的输出，对手“分不清”Bob 的数据是否被使用过。

定义 3.7。我们说，如果对于每个对手 A，我们有 D∞(V 0kV 1) ⩽ ε，其中 V b 表示以上 k 重合成实验 b 中 A 的视图，那么在 k 重自适应合成下，数据库访问机制家族 F 满足 ε-差分隐私。

(ε，δ)-k 倍自适应合成下的差分隐私反而要求 D｜δ(V 0kV 1)⩽ ε 。

定理 3.20(高级构图)。对于所有 ε，δ，δ0 ⩾ 0，k 倍自适应合成下的(ε，δ)-差分私有机制类满足(ε0，kδ+δ0)-差分隐私:

ε 0 = q2k ln(1/δ0)ε+kε(eε1)。

*证据。*对手 A 的视图由形式为 v = (r，y1，...,。*yk)，其中 r 是 A 和 y1 的硬币投掷，...,。yk 是 M1 机制的输出，...,。Mk。*让

*B = {v : Pr[V 0 = v] > eε0 Pr[V 1 = v]}。*我们将证明 Pr[V 0 ∈ B] ⩽ δ，因此对于每个集合 S，我们有 Pr[V 0∈S]⩽Pr[V 0∈B]+Pr[V 0∈(S \ B)]⩽ δ +eε0 Pr[V 1∈S]。

这相当于说 D｜δ(V 0kV 1)⩽ ε 0。

它仍然显示 Pr[V 0 ∈ B] ⩽ δ。设随机变量 V = 0

„Y_k^0) denote the view of *A* in Experiment 0 and $V^1 = (R^1, Y_1^1, \ldots Yk0)$*表示实验 0 中 A 的视图，V = 1(R1，Y11，…,。Yk1)实验 1 中 A 的视图.那么对于固定视图 v = (r，y1，…,yk)，我们有*

Pr[V 0 = v] !
_____ ln/
Pr[V 1 = v]

yi0 1 = yi 1] !

yi1 1 = yi 1]

k 0 = y
Pr[Y
yi0 1 = yi 1] !

_____

yi1 1 = yi 1]
我
k
def X
咦)。
i=1

, ,y_{i-1}) we condition on $R^0 = r$, $Y_1^0 = y_1$,*...yi 1)我们以 R0 = r，Y10 = y1 为条件，...,。*

Yi0) = ci(r，y1，...,yi 1，Y0)。一旦前缀被固定，下一对数据库 x0i 和 x1i、机制 Mi 和由 A 输出的参数 wi 也被确定(在实验 0 和 1 中)。于是 Yi0 按照 Mi(wi，x0i)分布。此外，对于任何值 yi，我们都有

Pr[M (w，x0i) = yi] ! *我 我*
yi 1，yi) = ln Pr[/Mi 1) = yi] 。
(wi，xi

通过 ε-差分隐私，这受到 ε 的限制。我们也可以这样推理:

yi 1，yi)|

≤最大{D∞(Mi(wi，x0i)kMi(wi，x1i))，

$^{1}_{i}$) D∞(Mi(wi，x kMi(wi，x0i))} =

ε。

根据引理 3.18，我们有:

„Yi0)|R0 = r,Y10 = y1,...Yi0)|R0 = r，Y10 = y1，...,。$^{0}_{i}$) yi0 1 =

yi 1]= D(Mi(wi，x KMi(wi，x1i))≤ ε (eε1)。

因此，我们可以将阿兹马不等式应用于随机变量 Ci =
yi0),α=ε，β=εεε0，z = p2ln(1/δ),推导出
" #
Pr[V 0 ∈ B] = Pr XCi > ε0< e z2/2 =δ，
我
如你所愿。

为了将证明扩展到(ε，δ)-差分私有机制的组成，对于 δ > 0，我们使用来自引理 3.17(第 2 部分)的近似最大散度的特征来将分析简化为与(ε，0)-不可区分序列的情况相同的情况。具体来说，使用引理 3.17，第 2 部分，对于对手 A 选择的每个差分私有机制和统计距离的三角形不等式，得出 V 0 是 kδ-接近随机变量 W = (R，Z1，...,。*Zk)使得对于每个前缀 r，y1，...,。*$_{1} = Y_1^1 = y_1$ *yi-1，如果我们以 R = R1 = r，Z 为条件，...,。子*$_{-1} = Y_{i-1}^1 = y_{i-1}$*，则认为 D∞(ZikYi1) ≤ε，D∞(Yi1kZi) ≤ε。*

这足以表明 D｜δ0(WkV 1)≤ ε 0.由于 V 0 是 kδ-接近 W，引理 3.17，第 1 部分给出了 Dδ0+kδ(V 0kW) ≤ ε0。□

一个直接而有用的推论告诉我们，如果我们希望确保给定 ε0，δ0 的(ε0，kδ+δ0)-差分隐私，那么对于 k 个机制中的每一个，ε 都是安全的选择。

推论 3.21。给定目标隐私参数 0 < ε0 < 1 和 δ0 > 0，以确保 k 个机制上的(ε0，kδ + δ0)累积隐私损失，每个机制是(ε，δ)-差分隐私就足够了，其中

$$\varepsilon\, 0\, \varepsilon = /. \qquad \overline{\qquad\qquad\qquad}$$
$$2p2k \ln(1/\delta 0)$$

*证据。*定理 3.20 告诉我们组成将为(ε∑，kδ + δ0)

全 δ0，其中 ε∞= p2k ln(1/δ0)ε+kε2。当 ε0 < 1 时，我们得到 ε∞≤ε0 为理想值。□

请注意，上面的推论给出了一个粗略的指南，说明如何设置 ε 以在合成下获得所需的隐私参数。当人们关心优化常数时(在处理实际实现时就是这样)，ε 可以通过直接引用合成定理来设置得更紧。

例 3.7。假设在鲍勃的一生中，他是 k = 10，000 (ε0，0)-差异私有数据库的成员。假设这些数据库之间没有协调——任何给定数据库的管理员甚至可能不知道其他数据库的存在——那么 ε0 的值应该是多少，以便在鲍勃的一生中，他的累积隐私损失以 ε = 1 为界，概率至少为 1e 32？定理 3.20 指出，取 δ0 = e32，ε0 ≤ 1/801 就足够了.假设不同的专用数据库之间没有协调，这对于任意的对手来说基本上是最优的。

那么，我们能以不平凡的准确性回答多少个问题呢？在一个大小为 n 的数据库中，如果误差为 o(n)量级，那么我们可以说精度是不小的。定理 3.20 说，对于 ε 和 δ 的固定值，有可能以非平凡的精度回答接近 n2 的计数查询。类似地，一个人可以回答接近 n 个查询，同时仍然拥有

$$\_\ \checkmark/$$

噪声 o(n) —即小于采样误差的噪声。我们将看到，有可能显著改善这些结果，在某些情况下，甚至可以处理指数数量的只有噪声的查询

$$\_\ \checkmark/$$

略大于 n，通过协调添加到各个响应中的噪声。事实证明，这种协调是必不可少的:没有协调，高级合成定理中的界限几乎是紧密的。

3.5.3 拉普拉斯与高斯

添加拉普拉斯噪声的替代方法是添加高斯噪声。在这种情况下，我们不是将噪声缩放到“1 灵敏度”,而是缩放到“2 灵敏度”:

定义 3.8(` 2-敏感性)。函数 f 的 2-灵敏度:
N|X| → Rk 为:

↮2(f)= 最大 KF(x)f(y)k2。
x，y∈N|X|
kx yk1 = 1

参数为 b 的高斯机制在 k 个坐标的每一个中添加方差为 b 的零均值高斯噪声。以下定理在附录 a 中得到证明。

定理 3.22。设 ε∞(0，1)是任意的。对于 c2 > 2ln(1.25/δ)，参数 σ≥c∏2(f)/ε 的高斯机制是(ε，δ)差分私有的。

高斯噪声的优点之一是，为保护隐私而添加的噪声与其他噪声源属于同一类型；此外，两个高斯的和是高斯的，因此隐私机制对统计分析的影响可能更容易理解和纠正。

这两种机制在合成情况下会产生相同的累积损失，因此即使每个单独计算的隐私保证较弱，许多计算的累积效应也是可比的。同样，如果 δ 足够大(例如，。很小，实际上我们永远不会经历担保的弱点。

也就是说，相对于我们对拉普拉斯噪声的体验，高斯噪声在理论上有一个缺点。考虑报告噪声最大值(带拉普拉斯噪声)，在这种情况下，每个候选输出在数据库 x 上的质量分数与其邻居 y 上的质量分数相同。与候选输出的数量无关，该机制产生(ε，0)-差分隐私。相反，如果我们使用高斯噪声并报告最大值，并且如果候选数比 1/δ 大，那么我们将准确地选择具有大高斯噪声的事件——概率小于 δ 的噪声。当我们离高斯的尾部如此之远时，我们不再能保证观测值在一个 e ε 因子内，这个因子在 x 上出现的可能性和在 y 上出现的可能性一样大。

3.5.4 作文评语

分析合成情况下累积隐私损失的能力让我们了解到一个差异私有数据库的世界能够提供什么。
一些观察是有序的。

弱量化。假设对手总是选择 x0i 来保存 Bob 的数据，而 x1i 是同一个数据库，但是删除了 Bob 的数据。定理 3.20，通过适当选择参数，告诉我们一个对手——包括一个知道或者甚至选择(！数据库对—在确定 b∞{ 0，1}的值方面没有什么优势。这是一个固有的弱量化。我们可以确保对手不太可能区分现实和任何特定的选择，但我们不能同时确保所有选择。如果有无数个数据库，但鲍勃只是其中 10，000 个数据库中的一员，那么我们不能同时保护鲍勃的缺席免受所有亿万个数据库减去一万个数据库的影响。这类似于(ε，δ)-差分隐私定义中的量化，其中我们预先固定一对相邻的数据库，并认为这两个数据库的输出几乎同样可能。

人类和鬼魂。直观地说，每条记录只有少量位的(，0)差异私有数据库的保护性不如包含我们整个病史的相同选择的差异私有数据库。那么，从什么意义上来说，我们的主要隐私措施，告诉我们同样的事情，关于数据库，根本不同的复杂性和敏感性的数据存储？答案在于组成定理。想象一个有两种生物居住的世界:鬼魂和人类。这两种类型的生物行为相同，以相同的方式与他人互动，以相同的方式写作、学习、工作、大笑、恋爱、哭泣、繁殖、生病、康复和衰老。唯一的区别是鬼魂在

数据库，而人类有。隐私对手的目标是确定给定的 50 岁“目标”是鬼还是人。事实上，对手有整整50年的时间来这么做。对手不需要保持被动，例如，她可以组织临床试验并登记她选择的患者，她可以创建人类来填充数据库，有效地创建最坏情况(为了隐私)的数据库，她可以在 25 岁时将目标暴露在化学物质中，并在 35 岁时再次暴露，等等。她可以知道目标的所有信息，这些信息可能会被输入任何数据库。如果目标是人类，她可以知道目标会在哪个数据库里。合成定理告诉我们，每个数据库的隐私保证——无论数据类型、复杂性和敏感性如何——都为人/鬼位提供了相当的保护。

3.6 稀疏向量技术

拉普拉斯机制可以用于回答自适应选择的低灵敏度查询，并且我们从我们的合成定理中知道，隐私参数与回答的查询数量(或其平方根)成比例地降低。不幸的是，经常会发生这样的情况:我们有大量的问题要回答——太多了，无法使用独立的扰动技术产生合理的隐私保证，即使有 3.5 节的高级合成定理.然而，在某些情况下，我们只关心高于某个阈值的查询的身份。在这种情况下，我们可以通过丢弃明显低于阈值的查询的数字答案，并仅报告它们确实低于阈值，来超越天真的分析。(如果我们选择这样做，我们将能够以很少的额外成本获得高于阈值的查询的数值)。这类似于我们在第 3.3 节的“报告最大噪声”机制中所做的，实际上迭代该算法或指数机制将是非交互式或离线情况下的一个选项。

在本节中，我们将展示如何在在线设置中分析这种方法。技术很简单——添加噪声并仅报告噪声值是否超过阈值——我们的重点是分析，显示隐私仅随着实际高于阈值的查询数量而降低，而不是随着查询总数而降低。如果我们知道高于阈值的查询集比查询总数小得多，也就是说，如果答案向量是稀疏的，这将是一个巨大的节约。

更详细地说，我们将考虑一系列事件——每个查询一个——如果在数据库上评估的查询超过给定的(已知的，公共的)阈值，就会发生这些事件。我们的目标是发布一个位向量，指示每个事件是否已经发生。

当每个查询出现时，该机制将计算一个有噪声的响应，将其与(众所周知的)阈值进行比较，如果超过阈值，则揭示这一事实。出于隐私证明(定理 3.24)中的技术原因，该算法在阈值 t 的噪声版本 T7 下工作。虽然 T 是公开的，但嘈杂的版本不是。

下面的分析不会导致每个可能的查询的隐私损失，只会导致接近或高于阈值的查询值的隐私成本。

设定。让 m 表示灵敏度 1 查询的总数，可以自适应地选择。不失一般性，有一个预先固定的单一阈值 T(或者，每个查询可以有自己的阈值，但结果不变)。我们将向查询值添加噪声，并将结果与 t 进行比较。肯定的结果意味着有噪声的查询值超过了阈值。我们预计少量噪声值会超过阈值，我们只释放阈值以上的噪声值。算法将在其停止条件下使用 c。

我们将首先分析算法在 c = 1 次超阈值查询后停止的情况，并表明无论查询的总序列有多长，该算法都是-差分私有的。然后我们将使用我们的合成定理分析 c > 1 的情况，并导出(，0)和(，δ)微分隐私的界。

我们首先认为，专门针对只有一个阈值以上查询的情况的算法是私有且准确的。

算法 1 输入是私有数据库 D，自适应选择的敏感度 1 查询流 f1，..。和阈值 t。输出是响应流 a1，...

阈值以上(D，{fi}，T，)

设 T˘= T+Lap2。对于每个

查询，如果 fi(D)+vI≥

T7，则让 vI =$(\frac{4}{\epsilon})$圈，然后

输出 ai = >。

停下来。

其他

输出 ai = ⊥.

如果…就会结束
结束于

定理 3.23。Threshold 以上是(，0)-差分私有。

*证据。*修复任意两个相邻的数据库 D 和 D0.让 A 表示代表 Threshold 以上(D，{fi}，T)输出的随机变量，让 A0 表示代表 Threshold 以上(D0，{fi}，T，)输出的随机变量。该算法的输出是这些随机变量 a ∈ {>,⊥}k 的某种实现，并且具有对于所有 i < k，ai = ⊥和 ak = >的形式。算法内部有两种类型的随机变量:噪声阈值 T7 和对 k 个查询中每一个的扰动，{vi}ki=1.对于下面的分析，我们将固定 v1 的(任意)值，...,。 *ˇk-1，取 vk 和 t 的随机性的概率。*定义以下表示任何查询 f1 的最大噪声值的量，...,。*D 上评估的 fk 1:*

$$g(D) = \max_{i<k}(fi(D) + vi)$$

在下文中，我们将滥用符号并将 Pr[T\ u = t]写成 t 处评估的 T\ u 的 pdf 的简写形式(类似于 vk)，并写入 1[x]来表示事件 x 的指示函数。注意，固定 v1 的值，...,。 *ˇk-1(使 g(D)成为确定性量)，我们有:*

Pr[A = A]= Pr[T > g].(D)和 fk(D)+vk≥T7]
t，v\u kt，v\u k

= Pr[tˇ(g(D)，fk(D) + vk]]
t，v\u k

Z ∞ Z ∞ = Pr[vk = v]
−∞ −∞

pr[T\ = t]1[t ∈(g(D)，fk(D) + v]]dvdt

。

= *

我们现在改变变量。定义:

$$\check{v} = v + g(D)g(D0) + fk(D0)fk(D)\check{t} = t + g(D)g(D0)$$

请注意，对于任何 D、D0、| v\u v |≤2 和| t\u t |≤1。这是因为每个查询 fi(D)都是 1 敏感的，因此数量 g(D)也是 1 敏感的。应用变量的这种变化，我们有:

Z∨Z∨

$$\Sigma = Pr[vk = \check{v}]Pr[\check{t} = t\check{]}1\ [(\ t + g(D) - g(D0))\ \Sigma\ \check{}$$

$$\Sigma(g(D)，\ fk(D0) + v + g(D) - g(D0)]]dvdt$$

Z∨Z∨

$$=Pr[vk = \check{v}]Pr[\check{t} = t\check{]}1[(t\Sigma(g(D0)，\ fk(D0) + v]]dvdt$$

$$\leq \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \exp(\epsilon/2)\Pr[\nu_k = v] \cdot \exp(\epsilon/2)\Pr[\hat{T} = t]\mathbf{1}[(t \in (g(D'), f_k(D') + v]] \bullet dvdt$$

= exp(Pr)[T > g\ (D0)和 fk(D0)+vk≥T7]

t，v\u k

= exp() Pr [A0 = a]

t，v\u k

其中不等式来源于我们对| v-v |和| t-t |的限制以及拉普拉斯分布的 pdf 形式。□

定义 3.9(准确性)。我们会说输出答案流 a1 的算法，...,。∈ *{>,⊥}*响应于 k 个查询 f1 的流，...,。fk 相对于阈值 T 是(α，β)-精确的，如果除了概率至多为 β 之外，算法不会在 fk 之前停止，并且对于所有 αi = >:*

$$fi(D) \geq T\alpha$$

对于所有人工智能= ⊥:

fi(D) ⩽*T + α*。

算法 1 会出什么问题？噪声阈值 T7 可能离 T 很远，比如说| T7 | T | >α。此外，一个小的计数 fi(D)< Tα 可能会有如此多的噪声加入其中，以至于它被报告为高于阈值(即使当阈值接近正确时)，而一个大的计数 fi(D) > T + α 可以被报告为低于阈值。所有这些都发生在概率指数很小的 α 中。总之，我们可能对噪声阈值的选择有问题，或者我们可能对一个或多个单独的噪声值 vi 有问题。当然，我们可能有两种错误，所以在下面的分析中，我们为每种类型分配 α/2。

fk 使得| ｛ I ＜ k:fi(D)⩾T α ｝ | = 0(即。唯一接近高于阈值的查询可能是最后一个查询)，高于阈值(D，{fi}，T，)对于以下情况是准确的(α，β):

$$8(\log k + \log(2/\beta))\ \alpha = \ /.$$

证据。注意，如果我们能证明，除了概率至多为 β:

$$\max_{i\in[k]} |\nu_i| + |T - T| \leq \alpha \text{ 。}$$

如果是这种情况，那么对于任何 ai = >，我们都有:

fi(D)+vI⩾tˇ⩾T | Ttˇ|

或者换句话说:

fi(D)⩾T | T TJ | |vI | ⩾Tα

类似地，对于任何 ai = ⊥，我们有:

fi(D)< TJ⩽T+| T TJ |+|vI | ⩽T+α

对于任何 I < k:fi(d)< t-α< t |vI | | t-t |,我们也会有这样的结果，因此:fi(d)+vI≤t ',这意味着 ai = ⊥. 因此，在 k 个查询被回答之前，算法不会停止。

我们现在完成证明。

回想一下，如果 Y∞Lap(b)，那么:Pr[| Y |≥t b]= exp(t)。因此，我们有:

$$\Pr[|T - \hat{T}| \geq \frac{\alpha}{2}] = \exp\left(-\frac{\epsilon\alpha}{4}\right.$$

将这个量设置为至多 β/2，我们发现我们需要 α ≥ 4log(2/β)

同样，受工会约束，我们有:

$$\alpha$$

$$\_\_ \text{pr}[\max |vI| \geq \alpha/2] \leq k \exp/I \in k] \quad 8$$

将这个量设置为至多 β/2，我们发现我们需要 α ≥ 8(log(2/β)+logk)

/ 这两种说法结合起来证明了这个定理。□

我们现在展示如何使用组合来处理多个“阈值以上”的查询。

稀疏算法可以这样理解:当查询进入时，它会重复调用 Threshold 以上。每当报告一个高于阈值的查询时，算法只需在一个新的高于阈值的实例上重新启动剩余的查询流。它在重新启动超过阈值 c 次后停止(即。在高于阈值 c 的查询出现之后)。Threshold 的每个实例化都是(，0)-私有的，因此组合定理适用。

定理 3.25。稀疏是(，δ)-差分私有。

*证据。*我们观察到稀疏完全等同于以下过程:我们对查询流{fi}设置运行 Threshold 以上(D，{fi}，T，0)

c，如果 δ= 0；

0

= p/ 1，否则。 8cln δ

算法 2 输入是私有数据库 D，自适应选择的敏感度 1 查询流 f1，..。
阈值 T 和截止点 C。输出是答案流 a1，...

Sparse(*D, {fᵢ}, T, c, , δ*)

$\frac{2c}{}$ $\overline{32c \ln \frac{1}{\delta}}$

如果 δ = 0，让 σ =。**否则让 σ =**

让 t0 = T+Lap(σ)让计数= 0 对
于我做的每个查询，让 vi =
Lap(2σ)如果 fi(D)+vI≥t0 计
数，则输出 ai = >。
让计数=计数+1。
让 T 计数= T + Lap(σ) else

输出 ai = ⊥. **如果**
计数≥ c，则结
束，然后暂停。如
果...就会结束
结束于

使用由 AboveThreshold 提供的答案。当 Threshold 以上暂停时(在 1 次 Threshold 以上查询之后)，我们只需在剩余的流上重新启动稀疏(D，{fi}，T，0)，并以这种方式继续，直到我们重新启动 threshold 以上 c 次。在阈值以上的第三次重启停止后，我们也停止。我们已经证明了 Threshold 以上(D，{fi}，T，0)是(0，0)差分私有的。最后，根据高级合成定理(定理 3.20)，0 = p/1-差分私有算法的 c 应用是(，δ)

8cln

δ

0 = /c 差分私有算法的差分私有和 c 应用根据需要是(，0)-私有的。□

通过再次观察到稀疏只由对 Threshold 以上的 c 调用组成，仍然需要证明稀疏的准确性。我们注意到，如果对 Threshold 以上的这些调用中的每一个都是(α，β/c)-精确的，那么稀疏将是(α，β)-精确的。

fk 使得 L(T)≦| { I:fi(D)≥T α } |≤c，如果 δ > 0，稀疏是(α，β)精确的:

(lnk + ln 2βc)q512cln 1δ α =。

如果 δ = 0，稀疏表示(α，β)精确到:

8c(lnk + ln(2c/β))

α =/

*证据。*我们简单地应用定理 3.24，将 β 设定为 β/c，并且

p/ 1 和/c，分别取决于 δ > 0 还是 δ = 0。□

8cln δ

最后，我们给出了一个稀疏版本，它实际上输出了上述阈值查询的数值，我们可以在精度损失恒定的情况下做到这一点。我们称这个算法为数值解析，它只是一个带有拉普拉斯机制的稀疏组合。不是输出向量 a ∈ {>,⊥}*，而是输出向量 a ∈ (R ∪ {⊥})*.

我们观察到 NumericSparse 是私有的:

定理 3.27。NumericSparse 是(，δ)-差分私有。

*证据。*请注意，如果 δ = 0，numeric parse(D，{fi}，T，c，，0)只是稀疏(D，{fi}，T，c，$\frac{8}{9}$，0)的自适应合成，以及带有隐私参数(0，δ) = ($\frac{1}{9}$，0)

的拉普拉斯机制。如果 δ > 0，**那么** numeric parse(D，{fi}，T，c，，0)**是稀疏**(D，{fi}，T，c，$\frac{\sqrt{512}}{\sqrt{512}+1}$，$\delta^{(\epsilon',\delta)} = (\frac{1}{\sqrt{512}+1}$ 2)**与带有隐私参数，**δ2)**的拉普拉斯机制的组合。**因此，数字解析的隐私来自简单的组合。□

为了讨论准确性，我们必须定义我们所指的机制的准确性，该机制响应一系列数值查询输出流 a∞(r∪{⊥})*:

算法 3 输入是私有数据库 D，自适应选择的敏感度 1 查询流 f1，..。阈值 T 和截止点 C。输出是答案流 a1，...

数字解析(D，{fi}，T，c，，，δ)

如果 δ = 0，让 1 ← $\frac{8}{9}$, 2 ← $\frac{2}{9}$. **否则让**$^1 = \frac{\sqrt{512}}{\sqrt{512}+1}\epsilon, \epsilon_2 = \frac{2}{\sqrt{512}+1}$

如果 δ = 0，让 σ() = 2c。**否则让**$\sigma(\epsilon) =$

让 t0 = T+Lap(σ(1))让计数= 0 对于我做的每个查询，让 vI = Lap$(2\sigma(\epsilon_1))$**如果 fi(D)+vI⩾t0 计数，那么让 υI←Lap**$(\sigma(\epsilon_2))$**输出 ai = fi(D) + υi。让计数=计数+1。**

让 T 计数= T +圈

16log|X|log|Q| + 4log	1	(4.2)	年底
β 最大值| f(x)-f(y)|≤ f∈Q ε kxk1 等效地，对于任何具有 x		⍰.	。 975
16log|X|log|Q| + 4log β1 kxk1 ⩾	越南	证据。	(4.3)

随机反应是由于华纳[84](比差别隐私早了40年！ ?咨询团 柬埔寨	麦克谢里和米罗诺夫得到了类似的证明。	/	1978
四	/	释放带有相关错误的线性查询	1979
在这一节中，我们考虑了解决查询释放问题的算法，与简单使用拉普拉斯机制的组合相比，该算法具有更好的准确性。	66	回应“你已经知道大概的答案了，因为你刚刚问了我几乎完全一样的问题。	进行的
p|Q|ln(1/δ)。 边界冲突	越南	形式 f : X → [0，1]，并应用于数据库 X 返回数据库上查询的总和或平均值(我们将两者都考虑，这取决于哪一个更便于分析)。	1981
f(x) = /Xxi f(x i)。	当我们认为线性查询返回和值时，我们将它们称为非规范化线性查询，并说它们取值:	|X|	1992
每当我们声明一个界限时，从上下文中应该很清楚我们说的是规范化查询还是非规范化查询，因为它们取的值在非常不同的范围内。	我们将展示两种技术，分别用于线下和线上案例。	u(x，y) =最大| f(x)-f(y)|。	1988

请注意，使用此定义，线性查询的灵敏度Π f≤1。 后面几节将讨论任意低敏感度查询。			
在线机制，尽管事先不知道整个查询集，但将获得与离线机制相同的准确性，并且将是稀疏向量技术的直接应用。	在我们继续介绍这些机制之前，我们将给出一个有用的线性查询类的例子。	1992	1993
在本节中，我们给出了一个基于使用指数机制对小型数据库进行采样的算法。	算法 4 小型数据库机制	SmallDB(x，Q，ε，α)	进行的
u(x，y) =最大| f(x)-f(y)| f∈Q	指数机制下的采样和输出 y ∈ R	/	2017
黎巴嫩联合国维和行动	证据。	/	进行的
	换句话说，我们将展示对于任何线性查询的集合	Q 而对于任何数据库 x，都有一个大小为 kyk1 = log|Q|的“小”数据库 y	进行的
引理 4.3(采样界限)。	log|Q| kyk1 = /α 2	这样:	进行的
|X|	一	f(y) = /Xyi f(χi) = /1 Xm f(Xi)。	进行的
|X|	X xj	英[法(Xi)] = /f(χj) = f(x)，	进行的

1 个 Xm
— E[f(y)] =
E[f(Xi)] = f(x)。
m**end if**
i=1

---

pr[| f(y)f(x)| >α]≤2e 2mα2。 We will say that an algorithm which outputs a stream of answers $a_1,\ldots,$ (R {}) in response to a stream of k queries $f_1,\ldots,f_k$ is (α,β)-accurate with respect to a threshold T if except with probability at most β, the algorithm does not halt before f_k, and for all a_i R:

取所有线性查询 f ∈ Q 上的并集，我们
得到:

2

Pr 最大值| f(y)f(x)| >α≤2 | Q | e 2α。

Theorem 3.28. For any sequence of k queries $f_1,\ldots f \in Q$ such that $L(T) \equiv |\{i : f_i(D) \geq T - \alpha\}| \leq c$, if $\delta > 0$, NumericSparse is (α,β) accurate for:

$$\alpha = \frac{(\ln k + \ln \frac{4c}{\beta})\sqrt{c \ln \frac{2}{\delta}}(\sqrt{512}+1)}{}.$$

定

理 4.2 的证明简单地来自于观察到 R 包

含所有大小为 log/α|Q|2 的数据库。

设 Q 为任意一类线性查询。
$\max_{f \in \mathcal{Q}} |f(x) - f(y)| \leq \alpha +$

ε *kxk1*Accuracy requires two conditions: first, that for all a_i = ⍰: $f_i(D) \leq T + \alpha$. This holds with probability $1 - \beta/2$ by the accuracy theorem for Sparse. Next, for all a_i⍰ R, it requires $|f_i(D) - a_i| \leq \alpha$. This holds for with probability $1-\beta/2$ by the accuracy of the Laplace mechanism. □

证据。应用指数机制的效用界限 This accuracy is equal, up to constants and a factor of log*k*, to the accuracy we would get, given the same privacy guarantee, if we knew the identities of these large above-threshold queries ahead of time, and answered them with the Laplace mechanism. That is, the sparse vector technique allowed us to fish out the identities of these large queries almost "for free", paying only logarithmically for the irrelevant queries. This is the same guarantee that we could have gotten by trying to find the large queries with the exponential mechanism and then answering them with the Laplace mechanism. This algorithm, however, is trivial to run, and crucially, allows us to choose our queries adaptively.

(定理 3.11)当∏u = kx1k 1 且 OPTq(D) ⩽ α(由定理 4.2 得出)时，我们发现:

Randomized Response is due to Warner [84] (predating differential privacy by four decades!). The Laplace mechanism is due to Dwork et al. [23]. The exponential mechanism was invented by McSherry and Talwar [60]. Theorem 3.16 (simple composition) was claimed in [21]; the proof appearing in Appendix B is due to Dwork and Lei [22];

3.7.我们通过(1)指出 R 满足|R| ≤ |X|log|Q|/α2，完成了证明，其中 R 是所有数据库的集合，并且

一 The material in Sections 3.5.1 and 3.5.2 is taken almost verbatim from Dwork et al. [32]. Prior to [32] composition was modeled informally, much as we did for the simple composition bounds. For specific mechanisms applied on a single database, there are “evolution of confidence” arguments due to Dinur, Dwork, and Nissim [18, 31], (which pre-date the definition of differential privacy) showing that the privacy parameter in*√ k*-fold com-

最后，我们现在可以陈述 SmallDB 的效用定理。Theorem 3.20 generalizes those arguments to arbitrary differentially private mechanisms,

定理 4.5。通过适当选择 α，让 y 作为 SmallDB(x，Q，ε，)的数据库输出，我们可以确保概率 1β:The sparse vector technique is an abstraction of a technique that was introduced, by Dwork, Naor, Reingold, Rothblum, and Vadhan [28] (indicator vectors in the proof of Lemma 4.4). It has subsequently found wide use (e.g. by Roth and Roughgarden [74], Dwork, Naor, Pitassi, and Rothblum [26], and Hardt and Rothblum [44]). In our presentation of the technique, the proof of Theorem 3.23 is due to Salil Vadhan.

4

Releasing Linear Queries with Correlated Error

One of the most fundamental primitives in private data analysis is the ability to answer numeric valued queries on a dataset. In the last section, we began to see tools that would allow us to do this by adding independently drawn noise to the query answers. In this section, we continue this study, and see that by instead adding carefully correlated noise, we can gain the ability to privately answer vastly more queries to high accuracy. Here, we see two specific mechanisms for solving this problem, which we will generalize in the next section.

In this section, we consider algorithms for solving the *query release* problem with better accuracy than we would get by simply using compositions of the Laplace mechanism. The improvements are possible because the set of queries is handled as a whole — even in the online setting! — permitting the noise on individual queries to be correlated. To immediately see that something along these lines might be possible, consider the pair of queries in the differencing attack described in Section 1: "How many people in the database have the sickle cell trait?" and "How many people, not named X, in the database have the sickle cell trait?" Suppose a mechanism answers the first question using the Laplace mechanism and then, when the second question is posed,

responds "You already know the approximate answer, because you just asked me almost the exact same question." This coordinated response to the pair of questions incurs no more privacy loss than either question would do taken in isolation, so a (small) privacy savings has been achieved.

The query release problem is quite natural: given a class of queries Q over the database, we wish to release some answer a_i for each query $f_i \in Q$ such that the error $\max_i |a_i - f_i(x)|$ is as low as possible, while still preserving differential privacy.[1] Recall that for any family of low sensitivity queries, we

[1] It is the privacy constraint that makes the problem interesting. Without this constraint, the query release problem is trivially and optimally solved by just outputting exact answers for every query.

can apply the Laplace mechanism, which adds fresh, independent, noise to the answer to each query. Unfortunately, at a fixed privacy level, for (,0)-privacy guarantees, the magnitude of the noise that we must add with the Laplace mechanism scales with $|Q|$ because this is the rate at which the sensitivity of the combined queries may grow. Similarly, for $(,\delta)$-privacy guarantees, the noise scales with

---

$\sqrt{|Q|\ln(1/\delta)}$. For example, suppose that our class of queries Q consists only of many copies of the same query: $f_i = f$ for all i. If we use the Laplace mechanism to release the answers, it will add independent noise, and so each a_i will be an independent random variable with mean $f(x)$. Clearly, in this regime, the noise rate must grow with $|Q|$ since otherwise the average of the a_i will converge to the true value $f(x)$, which would be a privacy violation. However, in this case, because $f_i = f$ for all i, it would make more sense to approximate f only once with $a \approx f(x)$ and release $a_i = a$ for all i. In this case, the noise rate would not have to scale with $|Q|$ at all. In this section, we aim to design algorithms that are much more accurate than the Laplace mechanism (with error that scales with $\log|Q|$) by adding non-independent noise as a function of the set of queries.

Recall that our universe is $X = \{\chi_1,\chi_2,\ldots,\chi_{|X|}\}$ and that databases are represented by histograms in $\mathbb{N}^{|X|}$. A *linear query* is simply a counting query, but generalized to take values in the interval [0,1] rather than only boolean values. Specifically, a linear query f takes the

form $f : X \to [0,1]$, and applied to a database x returns either the *sum* or *average* value of the query on the database (we will think of both, depending on which is more convenient for the analysis). When we think of linear queries as returning *average* values, we will refer to them as *normalized* linear queries, and say that they take value:

$$\frac{1}{|x|}$$

$$\frac{}{1/3kxk_{1\ i=1}}$$

ε α 3

概率为 16:maxf ∈Q | f(x)f(y)| ⩽ α 。

证据。

. Note that normalized linear queries take values in [0,1], whereas un-normalized queries take values in $[0, kxk_1]$.

将该量设置为至多 α，并求解 kxk1 得到(4.3)。求解 α 产率(4.4)。

请注意，该定理指出，对于固定的 α 和 ε，即使 δ = 0，也有可能回答数据库大小中几乎指数级的许多查询。 这和拉普拉斯机制形成对比，我们直接用它来回答线性查询，只能回答线性很多。 mechanism with range equal to the set of all *small* databases y and quality function $u(x,y)$ equal to minus the maximum approximation error incurred by querying y to obtain an approximation for $f(x)$:

然而，我们可以简单地通过乘以 kxk1 得到非规范化查询的相应界限：

定理 4.6(非规范化查询的准确性定理)。

2/ |X|log|Q| + 4logβ1 1/3We don't actually care that the potential output databases are small, only that they are not too numerous: their number plays a role in the proof of utility, which is an immediate application of the utility theorem for the exponential mechanism (Theorem 3.11). More specifically, if the total number of potential outputs is not too numerous then, in particular, the total number of low-utility outputs is not too

numerous, and therefore the ratio of bad outputs to good outputs (there is at least one) is not too large.

16logAs a result, privacy will be immediate, but utility will require a proof. The key will be to argue that, even for a very large set of counting queries, few queries are “significant”; that is, significant queries will be sparse. As with the sparse vector algorithms, we can scale noise according to the number of significant queries, with little dependence on the total number of queries.

最大| f(x)-f(y)|≤kxk1⍰/⍰。

Example 4.1. 然而，这类查询表现出非常简单的结构，这使得它可以被小型数据库很好地近似。通过考虑我们的查询类的更精细的结构，我们将能够给出差分私有机制的界限，该机制在简单采样界限(Lemma 4.3)的基础上有所改进，并且即使对于双指数大类的查询也可以是非平凡的.3 我们在这里不会完全开发这些界限，而是将陈述更简单的计数查询类的几个结果。just those that ask about subsets of features S of size $|S| = k$ for some fixed k. This class of queries $Q_k = \{f_S : S ⍰ \{1,...,d\}, |S| = k\}$ has size $_k^d$.

定义 4.1(粉碎)。一类计数查询 q 粉碎点 S⊆X 的集合如果对于每一个 T ⊆ S，存在一个 f ∈ Q，使得{x ∈ S : f(x) = 1} = T。*也就是说，如果对于 S 的 2 | S | 个子集中的每一个，Q 中有某种函数将这些元素精确地标记为正，并且不将 S \ T 中的任何元素标记为正，那么 Q 就会粉碎 S。*

请注意，q 要粉碎 s，必须是|Q| ≥ 2|S|的情况，因为 q 必须包含每个子集 T⊆S 的函数 f。

定义 4.2(Vapnik–Chervonenkis(VC)维度)。如果存在基数为|S| = d 的集合 S ⊆ X，那么计数查询 q 的集合具有 VC 维 d，使得 q 粉碎 s，并且 q 不粉碎任何基数为 d+1 的集合.我们可以用 VC-DIM(Q)来表示这个量。This is important because it will allow us to simultaneously guarantee, for all sufficiently large databases, that there is at least *one* database in the range of the exponential mechanism that well approximates x on queries in

Q, and that there are not *too many* databases in the range to dissipate the probability mass placed on this "good" database.

再次考虑在域 X = R 上定义的范围[0，∞]上的一维区间类。
我们观察到有限概念类的 VC 维永远不能太大。

引理 4.7。 N^|X| : *kyk*1 $= \frac{\log|Q|}{\alpha^2}\}$
证据。

事实证明，在我们的小数据库机制范围内，我们可以用术语 VC-DIM(Q)来代替术语 log|Q。

定理 4.8。对于任何有限类线性查询 Q，如果 R = {y ∈ N|X| : kyk ∈ O/VC-DIMα2 (Q)}那么对于所有 x ∈ N|X|，存在一个 y ∈ R，使得:

作为这个定理的结果，我们得到了定理 4.5 的模拟，用 VC 维作为查询类复杂度的度量:

定理 4.9。.设 y 为 SmallDB 输出的数据库(x，Q，ε，)。

 |X|VC-DIM(Q) +对数 б1 1/3对数 sm is simply an instantiation of the exponential mechanism. Therefore, privacy follows from Theorem 3.10.

□

 εkxk1But first, we must justify our choice of range $R = \{y \in \mathbb{N}^{|X|} : \|y\|_1 = \log_{\alpha}|Q|_2\}$, the set of all databases of size $\log|Q|/\alpha_2$.

Theorem 4.2. 等效地，对于任何具有 x

$$\max_{f \in \mathcal{Q}} |f(x) - f(y)| \leq \alpha$$

kxk1 ≥ O /εα3

概率为 1б:maxf ∈Q | f(x)f(y)| ≤ α 。
一个类似的(虽然更麻烦)查询复杂性度量，即“脂肪粉碎维度”，定义了一类线性查询的复杂性，而不是简单地计算查询。

4.2 一种在线机制:私有乘法权重

我们现在将给出一种机制，用于回答在线到达并可以交互选择的查询。
该算法将是稀疏向量算法(可以自适应地回答阈值查询)和指数梯度下降算法的简单组合，用于在线学习线性预测器。

后一种算法也被称为对冲或更一般的乘法权重技术。

such that:

$$\max_{f \in \mathcal{Q}} |f(x) - f(y)| \leq \alpha$$

也就是说，让 Σ([X])表示集合[|X|]上的概率分布集合，我们有 XΣ([X])。 h probability $x_j/\|x\|_1$, and let y be the database containing elements $X_1,...,X_m$. Now fix any $f \in Q$ and consider the quantity $f(y)$. We have:

请注意，我们始终可以缩放数据库以具有此属性，而无需更改任何线性查询的规范化值。

/

算法 5 乘法权重更新规则。

/

MW(xt，ft，vt):如果 vt < ft(xt)，则

让 rt = ft else

假设 rt = 1 英尺

(即。X_i)] = ——*f*(χ_j) = *f*(*x*),

更新:对于所有 I Σ[| X |]Let

x\ ti+1 = exp (-ηrt[I])xti

t+1

———— /*m*

xti+1 = P | X | JX = 1ˇIxˇTJ+1

输出 xt+1。

/

定理 4.10。 修复一类线性查询 Q 和一个数据库 x ∈

Σ([X])，让 x1Σ([X])描述 X 上的均匀分布:x1i = 1/|X|对于所有 I。

| ft(x)-ft(XT)| >α，以及

| ft(x)-vt | <α。

那一定是:□

4log|X|

Proposition 4.4. L ≤ 1 + / . Let *y* be the database output by SmallDB(*x,Q,ε,α*). Then with probability 1 − *β*:

$$\max_{f\in\mathcal{Q}} |f(x) - f(y)| \leq \alpha + \quad 2\left(\frac{\log|\mathcal{X}|\log|\mathcal{Q}|}{\alpha^2} + \log\left(\frac{1}{\beta}\right.\right.$$

请注意，如果我们证明这个定理，我们将已经证明，对于序列中的最后一个数据库 xL+1，它必须是对于所有 f∈Q:| f(x)f(Xl+1)| ≤ α，否则有可能扩展序列，与极大性相矛盾。

定理 4.10 是通过跟踪一个势函数 ψ，测量假设数据库 xt 在时间 t 和真实数据库 d 之间的相似性来证明的。

*潜在功能开始时不会太大。*al mechanism

势函数总是非负的。

这三个事实将迫使我们得出结论，更新回合不能太多。

现在让我们开始分析收敛定理的证明。

证据。

L > 4/logα2|X|。□

我们定义我们的潜在函数如下。

Theorem 4.5. By the appropriate choice of α, letting y be the database output by SmallDB($x,Q,\varepsilon,\frac{\alpha}{2}$), we can ensure that with probability $1-\beta$:

=− xx[i]ηrt[i] -log xexp(−ηrt[j])xtj

i=1 i=j

|X|

=-ηHRT，Xi-log xexp(−ηrt[j])xtj

j=1

|X|

≥ηHRT，Xi-log xxtj(1+η2-ηrt[j])

j=1

引理 4.12。

ψtψt+1≥ηHRT，xti HRT，Xiη2

第一个不等式源于这样一个事实:

exp((ηrt[j])≤1ηrt[j]+η2(rt[j])2≤1ηrt[j]+η2。

第二个不等式来自于对数(1 + y) ≤ y 的事实 t for fixed α and ε, even with $\delta = 0$, it is possible to answer almost *exponentially* many queries

in the size of the database.[2] This is in contrast to the Laplace mechanism, when we use it directly to answer linear queries, which can only answer *linearly* many.

Note also that in this discussion, it has been most convenient to think about normalized queries. However, we can get the corresponding bounds for unnormalized queries simply by multiplying by kxk_1:

Theorem 4.6 (Accuracy theorem for un-normalized queries)**.** By the appropriate choice of α, letting y be the database output by SmallDB($x,Q,\varepsilon,\frac{\alpha}{2}$), we can ensure that with probability $1 - \beta$:

$$\max |f(x) - f(y)| \leq kxk_1 \sqrt{\frac{16\log\left(\frac{2|X|\log|Q| + 4\log\frac{1}{\beta}}{\alpha}\right)^{1/3}}{\varepsilon}}. \quad (4.4)$$

More Refined Bounds. We proved that *every* set of linear queries Q has a collection of databases of size at most $|X|^{\log|Q|/\alpha_2}$ that wellapproximates every database x with respect to Q with error at most α. This is often an over-estimate however, since it completely ignores the structure of the queries. For example, if Q simply contains the same query repeated over and over again, each time in a different guise, then there is no reason that the size of the range of the exponential mechanism should grow with $|Q|$. Similarly, there may even be classes of queries Q that have *infinite* cardinality, but nevertheless are well approximated by small databases. For example, queries that correspond to asking whether a point lies within a

[2] Specifically, solving for k we find that the mechanism can answer k queries for:

$$\alpha_3 kxk_1\, k \leq \exp\left(\frac{O}{\log|X|}\right).$$

[3] In fact, our complexity measure for a class of queries can be finite even for *infinite* classes of queries, but here we are dealing with queries over a finite universe, so there do not exist infinitely many distinct queries.

given interval on the real line form an infinitely large class Q, since there are uncountably many intervals on the real line. Nevertheless, this class of queries exhibits very simple structure that causes it to be well approximated by small databases. By considering more refined structure of our query classes, we will be able to give bounds for differentially private mechanisms which improve over the simple sampling bounds (Lemma 4.3) and can be non-trivial even for doubly exponentially large classes of queries.[3] We will not fully develop these bounds here, but will instead state several results for the simpler class of *counting queries*. Recall that a counting query $f : X \to \{0,1\}$ maps database points to boolean values, rather than any value in the interval $[0,1]$ as linear queries do.

Definition 4.1 (Shattering)**.** A class of counting queries Q *shatters* a collection of points $S \subseteq X$ if for every $T \subseteq S$, there exists an $f \in Q$ such that $\{x \in S : f(x) = 1\} = T$. That is, Q shatters S if for every one of the $2^{|S|}$ subsets T of S, there is some function in Q that labels exactly those elements as positive, and does not label any of the elements in $S \setminus T$ as positive.

Note that for Q to shatter S it must be the case that $|Q| \geq 2^{|S|}$ since Q must contain a function f for each subset $T \subseteq S$. We can now define our complexity measure for counting queries.

Definition 4.2 (Vapnik–Chervonenkis (VC) Dimension)**.** A collection of counting queries Q has VC-dimension d if there exists some set $S \subseteq X$ of cardinality $|S| = d$ such that Q shatters S, and Q does not shatter any set of cardinality $d+1$. We can denote this quantity by VC-DIM(Q).

Consider again the class of 1-dimensional intervals on the range $[0,\infty]$ defined over the domain $X = \mathrm{R}$. The function $f_{a,b}$ corresponding to the interval $[a,b]$ is defined such that $f_{a,b}(x) = 1$ if and only if $x \in [a,b]$. This is an infinite class of queries, but its VC-dimension is 2. For any pair of distinct points $x < y$, there is an interval that contains neither point ($a,b < x$), an interval that contains both points ($a < x < y < b$), and an interval that contains each of the points but not the other ($a < x < b < y$ and $x < a < y < b$). However, for any 3 distinct points $x < y < z$, there is no interval $[a,b]$ such that $f_{a,b}[x] = f_{a,b}[z] = 1$ but $f_{a,b}[y] = 0$.

We observe that the VC-dimension of a finite concept class can never be too large.

Lemma 4.7. For any finite class Q, VC-DIM(Q) ≤ log$|Q|$.

Proof. If VC-DIM(Q) = d then Q shatters some set of items $S \subseteq X$ of cardinality $|S| = d$. But by the definition of shattering, since S has 2^d distinct subsets, Q must have at least 2^d distinct functions in it. □

It will turn out that we can essentially replace the term log$|Q|$ with the term VC-DIM(Q) in our bounds for the SmallDB mechanism. By the previous lemma, this is can only be an improvement for finite classes Q.

Theorem 4.8. For any finite class of linear queries Q, if $R = \{y \in \mathbb{N}^{|X|} : \|y\|_1 = O(\frac{\text{VC-DIM}(Q)}{\alpha^2})\}$ then for all $x \in \mathbb{N}^{|X|}$, there exists a $y \in R$ such that:

$$\max_{f \in Q} |f(x) - f(y)| \leq \alpha$$

As a result of this theorem, we get the analogue of Theorem 4.5 with VC-dimension as our measure of query class complexity:

Theorem 4.9. Let y be the database output by SmallDB($x, Q, \varepsilon, \frac{\alpha}{2}$).
Then with probability $1 - \beta$:

$$\max_{f \in Q} |f(x) - f(y)| \leq O\left(\frac{|X| \cdot \text{VC-DIM}(Q) + \log(1/\beta) \cdot \log(1/\alpha)}{\varepsilon \|x\|_1}\right)$$

Equivalently, for any database x with

$$\|x\|_1 \geq O\left(\frac{|X| \cdot \text{VC-DIM}(Q) + \log(1/\beta) \cdot \log(1/\alpha)}{\varepsilon \alpha^3}\right)$$

with probability $1 - \beta$: $\max_{f \in Q} |f(x) - f(y)| \leq \alpha$.

An analogous (although more cumbersome) measure of query complexity, the "Fat Shattering Dimension," defines the complexity of a

class of linear queries, as opposed to simply counting queries. The Fat Shattering Dimension controls the size of the smallest "α-net" (Definition 5.2 in Section 5) for a class of linear queries Q as VC-dimension does for counting queries. This measure can similarly be used to give more refined bounds for mechanisms designed to privately release linear queries.

4.2 An online mechanism: private multiplicative weights

We will now give a mechanism for answering queries that arrive online and may be interactively chosen. The algorithm will be a simple combination of the sparse vector algorithm (which can answer threshold queries adaptively), and the exponentiated gradient descent algorithm for learning linear predictors online.

This latter algorithm is also known as Hedge or more generally the multiplicative weights technique. The idea is the following: When we view the database $D \in \mathbb{N}^{|X|}$ as a histogram and are interested only in linear queries (i.e., linear functions of this histogram), then we can view the problem of answering linear queries as the problem of learning the linear function D that defines the query answers $\langle D,q\rangle$, given

a query $q \in [0,1]^{|X|}$. If the learning algorithm only needs to access the data using privacy-preserving queries, then rather than having a privacy cost that grows with the number of queries we would like to answer, we can have a privacy cost that grows only with the number of queries the learning algorithm needs to make. The "multiplicative weights" algorithm which we present next is a classical example of such a learning algorithm: it can learn any linear predictor by making only a small number of queries. It maintains at all times a current "hypothesis predictor," and accesses the data only by requiring examples of queries on which its hypothesis predictor differs from the (true) private database by a large amount. Its guarantee is that it will always learn the target linear function up to small error, given only a small number of such examples. How can we find these examples? The sparse vector algorithm that we saw in the previous section allows us to do this on the fly, while paying for only those examples that have high error on the current multiplicative weights hypothesis. As queries come in, we ask whether the true answer to the query differs substantially from the answer to the query on the current multiplicative weights hypothesis. Note that this is a threshold query of the type handled by the sparse vector technique. If the answer is "no" — i.e., the difference, or error, is "below threshold," — then we can respond to the query using the publicly known hypothesis predictor, and have no further privacy loss. If the answer is "yes," meaning that the currently known hypothesis predictor gives rise to an error that is above threshold, then we have found an example appropriate to update our learning algorithm. Because "above threshold" answers correspond exactly to queries needed to update our learning algorithm, the total privacy cost depends only on the learning rate of the algorithm, and not on the total number of queries that we answer.

First we give the multiplicative weights update rule and prove a theorem about its convergence in the language of answering linear queries. It will be convenient to think of databases x as being probability distributions over the data universe X. That is, letting $\Delta([X])$ denote the set of probability distributions over the set $[|X|]$, we have $x \in \Delta([X])$.

Note that we can always scale a database to have this property without changing the normalized value of any linear query.

Algorithm 5 The Multiplicative Weights (MW) Update Rule. It is instantiated with a parameter $\eta \leq 1$. In the following analysis, we will take $\eta = \alpha/2$, where α is the parameter specifying our target accuracy.

MW(x^t, f_t, v_t): **if** $v_t < f_t(x^t)$

then

Let $r_t = f_t$ **else**

Let $r_t = 1 - f_t$

(i.e., for all χ_i, $r_t(\chi_i) = 1 - f_t[\chi_i]$) **end if**

Update: For all i ⍰ [/X/] Let

$$x\hat{}^{t}_{i}{}^{+1} = \exp(-\eta r_t[i]) \cdot x^t_i$$

$$t+1$$

$$/_{|j}X=1\hat{}\,i\,x\hat{}\,tj+1$$

剩下的证据现在很容易就能看出来。

Theorem 4.10. | vt ft(x)| <α。

因此，当且仅当 vt < ft(xt)时，ft(x) < ft(xt)。特别是，如果 ft(XT)ft(x)≥ α，rt = ft，如果 ft(x)ft(XT)≥ α，rt = 1 ft。 因此，通过引理 4.12 和更新规则中描述的 η = α/2 的选择，

1. *hr，xti-HRT，xi*。
2. *最后我们知道:*

α 2

| X | L /.

四

求解，我们发现:L || 2 | X |。

我们现在可以将乘法权重更新规则与 NumericSparse 算法结合起来，给出一个交互式查询释放机制。出于技术原因，这可以通过询问 f(x)-f(XT)(无绝对值)和 f(XT)-f(x)来完成。回想一下 numericsparse 算法的响应是⊥或某个超过 t 的(正)值。*我们在回答中使用助记符 E 来强调查询询问的是一个错误。*If the update rule is invoked only when the difference $|f(x) - f(x^t)|$ is truly large (Theorem 4.10, condition 1), and if the approximations $f(x)+\lambda_t$ are sufficiently accurate (Theorem 4.10, condition 2), then we can apply the theorem to conclude that updates are not so numerous (because L is not so large) *and* the resulting x^{L+1} gives accurate answers to all queries in Q (because no distinguishing query remains).

定理 4.13。在线乘法权重机制(通过数字解析)是(，0)-差分私有的。We will show:

1. 算法 6 在线乘法权重机制(通过 NumericSparse)将私有数据库 x、隐私参数 δ、准确度参数 α 和 β 以及可以从一类查询 q 中自适应选择的线性查询流{fi}作为输入。
2. 通过 numericparse/(x，{f }，，δ，α，β)在线 MW
3.

让 T ← /||x||1

如果…就会结束

用查询流{fi0}初始化 numeric parse(x，{fi0}，T，c，，，δ)，输出答案流 Ei。 $|v_t - f_t(x)| < \alpha$ cannot have

设 t ←0，设 x0 Σ([X])满足 x0i = 1/|X| 为所有 i ∈ [|X|]。

对于每个查询 fi doRecall that we here view the database as a probability distribution — i.e., we assume $kxk_1 = 1$. Of course this does not require actually modifying the real database. The potential function that we use is the relative entropy, or KL divergence, between x and x^t (when viewed as probability distributions):

设 f20i 1()= fi()-fi(XT)。

如果 e2i 1 =⊥，E2i = ⊥，那么

让 ai = fi(xt) else

如果 E2i 1∈R，则

Proposition 4.11. 设 ai = fi(XT)+E2i 1 else

*设 ai = fi(XT)-E2i 结束 if*Divergence) is always a non-negative quantity, by the log-sum inequality, which states that if $a_1,...,a_n$ and $b_1,...,b_n$ are non-negative numbers, then

设 xt+1 = MW(xt，fi，ai)设 t ← t +

1。*i i i*

结束于

/Noting that x is a probability distribution, we see that this quantity is maximized when $x[1] = 1$ and $x[i] = 0$ for all $i > 1$, giving $\Psi_i = \log|X|$. □

We will now argue that at each step, the potential function drops by at least $\alpha^2/4$. 证据。这直接源于 numericspare 的隐私分析，因为 OnlineMW 算法只通过 numericspare 访问数据库。

非正式地说，在线乘法权重机制的效用证明(通过 numericsplash)使用 numericsplash 的效用定理(定理 3.28)得出结论，乘法权重更新规则很有可能仅在查询 ft 是真正有区别的查询时被调用，这意味着| fi(x)-ft(XT)|是“大的”，并且所发布的 fi(x)的噪声近似值是“精确的”。

定理 4.14。*对于δ = 0，概率至少为 1β，对于所有查询 fi，在线乘法权重机制(通过 NumericSparse)返回一个答案 ai，使得任何 α 的| fi(x)ai | ⩽3α，从而:*

*32log|X| |X| α ≥*at $^{P_{|X|}}{}_{i=1}$
x[*i*] = 1.

|X| ! */X/*36log|x|log(|q|)+log32log|x|/3||x||12/31

考虑到这一点，我们可以! /3

|X|

32log|X|

α =

α 2||x||1

4.2. An online mechanism: private multiplicative weights 85

注意，对于δ = 0 的情况，有了这个值，我们得到T = 2α。

假设我们处于这种高(1β)概率情况。

此外，仍然通过稀疏向量算法的精度特性，

E2i-1、e2i 中最多有一个值为⊥；

对于不触发更新的所有 I(ai = fi(XT))，我们有| fi(x)-fi(XT)|≤T+α= 3α；和

对于触发更新的所有 I，我们有| fi(x)-ai |≤α (定理 4.10，条件 2)。

优化上述 α 的表达式并去除归一化因子，我们发现 OnlineMW 机制可以以 3α 的精度回答每个线性查询，除了概率 β:

这与 SmallDB 机制相当。 □

α ||x||1

再次优化 α 的上述表达式并去除归一化因子，我们发现 OnlineMW 机制可以以 3α 的精度回答每个线性查询，除了概率 β，对于:

就在一个镜头里，所以没有机会利用构图。

私有乘法权重算法的准确性依赖于几个参数，值得进一步讨论。

4.3 书目注释

本节给出的离线查询释放机制来自 Blum 等人。

$$\geq -\eta hr_t, xi - \log \sum x_j^t (1 + \eta^2 - \eta r_t[j]) \square$$

本节给出的在线查询发布机制来自 Hardt 和 Rothblum [44]。

5

/

一般化

/

The second inequality follows from the fact that log(1 + *y*) ≤ *y* for

y > −1. □

88By the conditions of the database/query sequence (described in the hypothesis for Theorem 4.10 above), for every *t*,

1. *5.1*。 通过 α 网的机制
2. *选择这个网的一个元素的指数机制，其中质量函数最小化网的元素在 Q 中的查询的最大误差。*

这些查询会在在线算法中产生更新步骤。

在接下来的部分中，我们将讨论查询类 q 的数据结构。

从某类数据结构 D 中为一类查询 Q 提取的数据结构 D 隐含地被赋予了一个求值函数 Eval ： D ×Q → R，利用这个函数我们可以在 D 上求值 Q 中的任何查询。

5.1 通过 α 网的机制

给定一组查询 Q，我们定义 α 网如下:

定义 5.2(α-网)。

$\max_{f \in \mathcal{Q}} |f(x) - f(y)| \leq \alpha$.

This completes the proof. □

我们写 Nα(Q)来表示 Q 的所有 α 网集合中最小基数的 α 网。For (,0) privacy, we essentially (with somewhat worse constants) recover the bound for SmallDB. For (,*δ*)-differential privacy, we obtain better bounds, by virtue of being able to use the composition theorem. The queries to NumericSparse are asking whether the magnitude of the error given by estimating $f_i(x)$ by applying f_i to the current approximation x^t to x is above an appropriately chosen threshold T, that is, they are asking if $|f(x) - f(x^t)|$ is large. For technical reasons this is done by asking about $f(x)-f(x^t)$ (without the absolute value) and about $f(x^t) - f(x)$. Recall that the NumericSparse algorithm responds with either ⍰ or some (positive) value exceeding T. We use the mnemonic E for the responses to emphasize that the query is asking about an error.

Theorem 4.13. The Online Multiplicative Weights Mechanism (via NumericSparse) is (,0)-differentially private.

也就是说，对于每个可能的数据库 x，对于 Q 中的所有查询，都存在一个"看起来"像 x 的 α 网成员，直到误差容限为 α。It outputs a stream of answers $\{a_i\}$.

小的 α 网对我们来说将是有用的，因为当与指数机制配对时，它们将直接导致以高精度回答查询的机制。

$$(2+32\sqrt{2})\sqrt{c \log \tfrac{2}{\delta}}(\log k + \log \tfrac{4c}{\beta})$$

网络机制(x，Q，ε，α)

设 R ← Nα(Q)

设 q : N|X| × R → R 定义为:

q(x，y) =最大| f(x)-f(y)| f∈Q

指数机制下的采样和输出 y ∈ R

Let $f_2{}^0{}_{i-1}(\cdot) = f_i(\cdot) - f_i(x^t)$. **Let** $f_2{}^0{}_i(\cdot) =$

$f_i(x^t) - f_i(\cdot)$

命题 5.1。 *Net 机制是(ε，0)差分私有的。*

证据。

同样，我们可以通过对指数机制的分析来开始理解网络机制的效用保证:

命题 5.2。 *设 Q 为任何一类灵敏度为 1/kxk1 的查询。*

2log(|Nα(Q)|) + log 1

Let $x^{t+1} = MW(x^t, f_i, a_i)$ **Let** $t \leftarrow$

$t + 1$. **.**

证据。

---

*t 前。*f∈Q εkxk1□

插入 t = logβ1 就完成了证明。/

Theorem 4.14. 因此，我们可以看到，函数集合 Q 的|Nα(Q)|上的上界立即给出了差分私有机制可以同时为类 Q 中的所有函数提供的精度上界。

这正是我们在 4.1 节中所做的，我们看到关键量是 Q 的 VC 维，当 Q 是一类线性查询时。

5.2 迭代构造机制

非正式地，数据库更新算法维护一系列数据结构 D2 D1，...

定义 5.3(数据库更新顺序)。

In our case $c = 4\log|X|/\alpha^2$ and $k = 2|Q|$, and we have been normalizing, which reduces α by a factor of $||x||_1$. D1 = U(⊥，，），

，$^2||x||_1$

对于每个 t = 1，2，..。，

对于每个 t = 1，2，..。 l 1，Dt+1 = U(Dt，ft，vt)。Thus, f_i, a_i form a valid pair of query/value updates as required in the hypothesis of Theorem 4.10 and so, by that theorem, there can be at most $c = {}^4\_\_\_\_\_^{\log_{\alpha^2}|X|}$ such update steps.

我们注意到，对于我们考虑的所有数据库更新算法，近似答案 vt 仅用于确定 ft(x)-ft(Dt)的符号，这是要求 ft(x) (vt)的估计误差小于 α 的动机。

1. 定义 5.4(数据库更新算法)。
2. 注意，T(α)-数据库更新算法的定义意味着，如果 U 是 T(α)-数据库更新算法，那么给定任何极大(U，x，Q，α，U)-数据库更新序列，最终的数据库
3. 也就是说，T(α)数据库更新规则的目标是生成最大数据库更新序列，最大数据库更新序列中的最终数据结构必然编码每个查询 f ∈ Q 的近似答案。

既然我们已经定义了数据库更新算法，我们可以注意到，我们在定理 4.10 中真正证明的是乘法权重算法是一个 T(α)-数据库更新算法，用于 T(α) = 4log|X|/α2。

> 在继续之前，让我们对什么是数据库更新算法建立一些直觉。

数据库更新算法和在线学习算法:我们注意到数据库更新算法本质上是错误界模型中的在线学习算法。

学习理论的背景。

不难看出，有限类函数 c 总是存在错误有界的在线学习算法。

除了布尔标签之外，我们还可以将数据库更新算法视为错误限制模型中的在线学习算法:这里，到达的示例是查询(可能以对抗顺序出现)。

Theorem 4.15. 一个用于类 Q 的数据库更新算法将与一个相应的识别器一起使用，该识别器的工作是输出一个在真实数据库 x 和假设 Dt 上表现不同的函数，也就是指出一个错误。

$$\alpha \geq \frac{(2+32\sqrt{2})\cdot\sqrt{\log|X|\log\frac{2}{\delta}}\ \log|Q|+\log\ \frac{32\log|X|}{\alpha^2\beta}}{}$$

, (3α except with probability β, for:

请注意，先验地，差异私有识别器是比差异私有释放算法更弱的对象:识别器仅在集合 Q 中找到具有近似最大值的查询，而释放算法必须找到 Q 中每个查询的答案。

我们将首先分析集成电路算法，然后用特定的识别器和数据库更新算法实例化它。接下来是一个形式分析，但是对于该机制的直觉很简单:我们简单地运行迭代数据库构建算法来构建一个假设，该假设相对于查询 q 近似匹配 x。如果在每一轮中，我们的识别器成功地找到一个假设数据库和真实数据库之间存在高度差异的查询，那么我们的数据库更新算法将输出一个相对于 qβ-精确的数据库。如果识别器没有找到这样的查询，那么肯定是没有这样的查询，并且我们的数据库更新算法已经了解了关于感兴趣的查询的准确假设！因此，隐私将遵循我们的合成定理。nd a guarantee that no more than $c = O(\log|X|/\alpha^2)$ queries will be above a threshold of $T = O(\alpha)$, for any α. (The queries we ask are always: " How much does the real answer differ from the predicted answer of the current multiplicative weights hypothesis." The answers to these questions both give us the true answers to the queries, as well as instructions how to update the learning algorithm appropriately when a query is above threshold.) Together, this leads us to set the threshold to be $O(\alpha)$, where α is the expression that satisfies: $\alpha = O(\log|X|\log k/(kxk_1\alpha^2))$. This minimizes the two sources of error: error from the sparse vector technique, and error from failing to update the multiplicative weights hypothesis.

4.3 Bibliographical notes

The offline query release mechanism given in this section is from Blum et al. [8], which gave bounds in terms of the VC-Dimension of the query class (Theorem 4.9). The generalization to fat shattering dimension is given in [72].

4.3. Bibliographical notes

The online query release mechanism given in this section is from Hardt and Rothblum [44]. This mechanism uses the classic multiplicative weights update method, for which Arora, Hazan and Kale give an excellent survey [1]. Slightly improved bounds for the private multiplicative weights mechanism were given by Gupta et al. [39], and the analysis here follows the presentation from [39].

5

Generalizations

In this section we generalize the query release algorithms of the previous section. As a result, we get bounds for arbitrary low sensitivity queries (not just linear queries), as well as new bounds for linear queries. These generalizations also shed some light on a connection between query release and machine learning.

The SmallDB offline query release mechanism in Section 4 is a special case of what we call the *net mechanism*. We saw that both mechanisms in that section yield *synthetic databases*, which provide a convenient means for approximating the value of any query in Q on the private database: just evaluate the query on the synthetic database and take the result as the noisy answer. More generally, a mechanism can produce a *data structure* of arbitrary form, that, together with a fixed, public, algorithm (independent of the database) provides a method for approximating the values of queries.

The Net mechanism is a straightforward generalization of the SmallDB mechanism: First, fix, independent of the actual database, an α-net of data structures such that evaluation of any query in Q using the released data structure gives a good (within an additive α error) estimate of the value of the query on the private database. Next, apply

the exponential mechanism to choose an element of this net, where the quality function minimizes the maximum error, over the queries in Q, for the elements of the net.

We also generalize the online multiplicative weights algorithm so that we can instantiate it with any other *online learning algorithm* for learning a database with respect to a set of queries. We note that such a mechanism can be run either online, or offline, where the set of queries to be asked to the "online" mechanism is instead selected using a "private distinguisher," which identifies queries on which the current hypothesis of the learner differs substantially from the real database. These are queries that would have yielded an update step in the online algorithm. A "distinguisher" turns out to be equivalent to an agnostic learning algorithm, which sheds light on a source of hardness for efficient query release mechanisms.

In the following sections, we will discuss *data structures* for classes of queries Q.

Definition 5.1. A data structure D drawn from some class of data structures D for a class of queries Q is implicitly endowed with an evaluation function Eval : $D \times Q \to \mathbb{R}$ with which we can evaluate any query in Q on D. However, to avoid being encumbered by notation, we will write simply $f(D)$ to denote Eval(D,f) when the meaning is clear from context.

5.1 Mechanisms via α-nets

Given a collection of queries Q, we define an α-net as follows:

Definition 5.2 (α-net)**.** An α-net of data structures with respect to a class of queries Q is a set $N \subseteq \mathbb{N}^{|X|}$ such that for all $x \in \mathbb{N}^{|X|}$, there exists an element of the α-net $y \in N$ such that:

$$\max_{f \in Q} |f(x) - f(y)| \leq \alpha \,.$$

We write $N_\alpha(Q)$ to denote an α-net of minimum cardinality among the set of all α-nets for Q.

That is, for every possible database x, there exists a member of the α-net that "looks like" x with respect to all queries in Q, up to an error tolerance of α.

Small α-nets will be useful for us, because when paired with the exponential mechanism, they will lead directly to mechanisms for answering queries with high accuracy. Given a class of functions Q, we will define an instantiation of the exponential mechanism known as the *Net* mechanism. We first observe that the Net mechanism preserves ε-differential privacy.

Algorithm 7 The Net Mechanism

NetMechanism(x,Q,ε,α)

Let $R \leftarrow N_\alpha(Q)$

Let $q : \mathrm{N}^{|X|} \times R \rightarrow \mathrm{R}$ be defined to be:

$$q(x,y) = -\max |f(x) - f(y)|$$

Sample And Output $y \in R$ with the exponential mechanism $M_E(x,q,R)$

Proposition 5.1. The Net mechanism is $(\varepsilon,0)$ differentially private.

Proof. The Net mechanism is simply an instantiation of the exponential mechanism. Therefore, privacy follows from Theorem 3.10. □

We may similarly call on our analysis of the exponential mechanism to begin understanding the utility guarantees of the Net mechanism:

Proposition 5.2. Let Q be any class of sensitivity $1/k \times k_1$ queries. Let y be the database output by NetMechanism(x,Q,ε,α). Then with probability $1 - \beta$:

$$2\log(|N_\alpha(Q)|) + \log \frac{1}{}$$

$$\max_{f \in \mathcal{Q}} |f(x) - f(y)| \leq \alpha + \frac{\overline{\beta}}{\varepsilon \|x\|_1}.$$

Proof. By applying Theorem 3.11 and noting that $S(q) = \|x\|_1$, and that $\mathrm{OPT}_q(D) \leq \alpha$ by the definition of an α-net, we find:

$$\max_{f \in \mathcal{Q}} |f(x) - f(y)| \geq \alpha + \frac{2}{\varepsilon \|x\|_1} \left(\log\left(|\mathcal{N}_\alpha(\mathcal{Q})|\right) + t\right) \leq \mathrm{Pr}e^{-t}.$$

Plugging in $t = \log \frac{1}{\delta}$ completes the proof. □

We can therefore see that an upper bound on $|N_\alpha(Q)|$ for a collection of functions Q immediately gives an upper bound on the accuracy that a differentially private mechanism can provide simultaneously for *all* functions in the class Q.

This is exactly what we did in Section 4.1, where we saw that the key quantity is the VC-dimension of Q, when Q is a class of linear queries.

5.2 The iterative construction mechanism

In this section, we derive an offline generalization of the private multiplicative weights algorithm, which can be instantiated with any properly defined learning algorithm. Informally, a database update algorithm maintains a sequence of data structures $D^1, D^2, \ldots$ that give increasingly good approximations to the input database x (in a sense that depends on the database update algorithm). Moreover, these mechanisms produce the next data structure in the sequence by considering only one query f that *distinguishes* the real database in the sense that $f(D^t)$ differs significantly from $f(x)$. The algorithm in this section shows that, up to small factors, solving the query-release problem in a differentially private manner is equivalent to solving the simpler *learning* or *distinguishing* problem in a differentially private manner: given a private distinguishing algorithm and a non-private database update algorithm, we get a corresponding private

release algorithm. We can plug in the exponential mechanism as a canonical private distinguisher, and the multiplicative weights algorithm as a generic database update algorithm for the general linear query setting, but more efficient distinguishers are possible in special cases.

Syntactically, we will consider functions of the form $U : D \times Q \times \mathrm{R} \to D$, where D represents a class of data structures on which queries in Q can be evaluated. The inputs to U are a data structure in D, which represents the current data structure D^t; a query f, which represents the distinguishing query, and may be restricted to a certain set Q; and also a real number, which estimates $f(x)$. Formally, we define a *database update sequence*, to capture the sequence of inputs to U used to generate the database sequence $D^1, D^2, \ldots$.

Definition 5.3 (Database Update Sequence)**.** Let $x \in \mathrm{N}^{|X|}$ be any database and let $(D^t, f_t, v_t)_{t=1,\ldots,L} \in (D \times Q \times \mathrm{R})^L$ be a sequence of tuples. We say the sequence is a (U,x,Q,α,T)-*database update sequence* if it satisfies the following properties:

1. $D^1 = U(\in, \cdot, \cdot)$,
2. for every $t = 1,2,\ldots,L$, $\left| f_t(x) - f_t(D^t) \right| \geq \alpha$,
3. for every $t = 1,2,\ldots,L$, $|f_t(x) - v_t| < \alpha$,
4. and for every $t = 1,2,\ldots,L-1$, $D^{t+1} = U(D^t, f_t, v_t)$.

We note that for all of the database update algorithms we consider, the approximate answer v_t is used only to determine the *sign* of $f_t(x) - f_t(D^t)$, which is the motivation for requiring that the estimate of $f_t(x)$ (v_t) have error smaller than α. The main measure of efficiency we're interested in from a database update algorithm is the maximum number of updates we need to perform before the database D^t approximates x well with respect to the queries in Q. To this end we define a database update algorithm as follows:

Definition 5.4 (Database Update Algorithm)**.** Let $U : D \times Q \times \mathrm{R} \to D$ be an update rule and let $T : \mathrm{R} \to \mathrm{R}$ be a function. We say U is a $T(\alpha)$*database*

update algorithm for query class Q if for every database $x \in \mathbb{N}^{|X|}$, every (U,x,Q,α,L)-database update sequence satisfies $L \leq T(\alpha)$.

Note that the definition of a $T(\alpha)$-database update algorithm implies that if U is a $T(\alpha)$-database update algorithm, then given any maximal (U,x,Q,α,U)-database update sequence, the final database D^L must satisfy $\max_{f \in Q} |f(x) - f(D^L)| \leq \alpha$ or else there would exist another query satisfying property 2 of Definition 5.3, and thus there would exist a $(U,x,Q,\alpha,L+1)$-database update sequence, contradicting maximality. That is, the goal of a $T(\alpha)$ database update rule is to generate a maximal database update sequence, and the final data structure in a maximal database update sequence necessarily encodes the approximate answers to every query $f \in Q$.

Now that we have defined database update algorithms, we can remark that what we really proved in Theorem 4.10 was that the Multiplicative Weights algorithm is a $T(\alpha)$-database update algorithm for $T(\alpha) = 4\log|X|/\alpha^2$.

Before we go on, let us build some intuition for what a database update algorithm is. A $T(\alpha)$-database update algorithm begins with some initial guess D^1[4] about what the true database x looks like. Because this guess is not based on any information, it is quite likely that D^1 and x bear little resemblance, and that there is some $f \in Q$ that is able to distinguish between these two databases by at least α: that is, that $f(x)$ and $f(D^1)$ differ in value by at least α. What a database update algorithm does is to update its hypothesis D^t given evidence that its current hypothesis D^{t-1} is incorrect: at each stage, it takes as input some query in Q which distinguishes its current hypothesis from the true database, and then it outputs a new hypothesis. The parameter $T(\alpha)$ is an upper bound on the number of times that the

[4] Imagine that the database update algorithm is attempting to sculpt x out of a block of clay. Initially, its sculpture D^1 bears no resemblance to the true database: it is simply a block of clay. However, a helpful distinguisher points out to the sculptor places in which the clay juts out much farther than the true target database: the sculptor dutifully pats down those bumps. If the distinguisher always finds large protrusions, of magnitude at least α, the sculpture will be finished soon, and the distinguisher's time will not be wasted!

database update algorithm will have to update its hypothesis: it is a promise that after at most $T(\alpha)$ distinguishing queries have been provided, the algorithm will finally have produced a hypothesis that looks like the true database with respect to Q, at least up to error α.[1] For a database update algorithm, smaller bounds $T(\alpha)$ are more desirable.

Database Update Algorithms and Online Learning Algorithms: We remark that database update algorithms are essentially *online learning algorithms* in the *mistake bound model*. In the setting of online learning, unlabeled examples arrive in some arbitrary order, and the learning algorithm must attempt to label them.

Background from Learning Theory. In the *mistake bound model of learning*, labeled examples $(x_i, y_i) \in X \times \{0,1\}$ arrive one at a time, in a potentially adversarial order. At time i, the learning algorithm A observes x_i, and must make a prediction $\hat{y}_i$ about the label for x_i. It then sees the true label y_i, and is said to *make a mistake* if its prediction was wrong: i.e., if $y_i = \hat{y}_i$. A learning algorithm A for a class of functions C is said to have a mistake bound of M, if for all $f \in C$, and for all adversarially selected sequences of examples $(x_1, f(x_1)), \ldots, (x_i, f(x_i)), \ldots$, A never makes more than M mistakes. Without loss of generality, we can think of such a learning algorithm as maintaining some hypothesis $\hat{f} : X \to \{0,1\}$ at all times, and updating it only when it makes a mistake. The adversary in this model is quite powerful — it can choose the sequence of labeled examples adaptively, knowing the current hypothesis of the learning algorithm, and its entire history of predictions. Hence, learning algorithms that have finite mistake bounds can be useful in extremely general settings.

It is not hard to see that mistake bounded online learning algorithms always exist for finite classes of functions C. Consider, for example, the *halving algorithm*. The halving algorithm initially maintains a set S of functions from C consistent with the examples that it has seen so far: Initially $S = C$. Whenever a new unlabeled example arrives, it predicts according to the majority vote of its consistent hypotheses: that is, it

predicts label 1 whenever $|\{f \in S : f(x_i) = 1\}| \geq |S|/2$. Whenever it makes a mistake on an example x_i, it updates S by removing any inconsistent function: $S \leftarrow \{f \in S : f(x_i) = y_i\}$. Note that whenever it makes a mistake, the size of S is cut in half! So long as all examples are labeled by *some* function $f \in C$, there is at least one function $f \in C$ that is never removed from S. Hence, the halving algorithm has a mistake bound of $\log|C|$.

Generalizing beyond boolean labels, we can view database update algorithms as online learning algorithms in the mistake bound model: here, examples that arrive are the queries (which may come in adversarial order). The labels are the approximate values of the queries when evaluated on the database. The database update algorithm hypothesis D^t makes a *mistake* on query f if $|f(D^t) - f(x)| \geq \alpha$, in which case we learn the label of f (that is, v_t) and allow the database update algorithm to update the hypothesis. Saying that an algorithm U is a $T(\alpha)$-database update algorithm is akin to saying that it has a mistake bound of $T(\alpha)$: no adversarially chosen sequence of queries can ever cause it to make more than $T(\alpha)$-mistakes. Indeed, the database update algorithms that we will see are taken from the online learning literature. The multiplicative weights mechanism is based on an online learning algorithm known as *Hedge*, which we have already discussed. The Median Mechanism (later in this section) is based on the *Halving Algorithm*, and the Perceptron algorithm is based (coincidentally) on an algorithm known as *Perceptron*. We won't discuss Perceptron here, but it operates by making *additive* updates, rather than the multiplicative updates used by multiplicative weights.

A database update algorithm for a class Q will be useful together with a corresponding *distinguisher*, whose job is to output a function that behaves differently on the true database x and the hypothesis D^t, that is, to point out a mistake.

Definition 5.5 (($F(\varepsilon),\gamma$)-Private Distinguisher)**.** Let Q be a set of queries, let $\gamma \geq 0$ and let $F(\varepsilon) : \mathrm{R} \rightarrow \mathrm{R}$ be a function. An algorithm $\mathrm{Distinguish}_\varepsilon : \mathrm{N}^{|X|} \times D \rightarrow Q$ is an ($F(\varepsilon),\gamma$)-Private Distinguisher for Q if for every setting of the privacy parameter ε, on every pair of inputs $x \in \mathrm{N}^{|X|}$, $D \in D$ it is (ε,0)-differentially

private with respect to x and it outputs an $f \in Q$ such that $|f(x) - f(D)| \geq \max_{f \in Q} |f(x) - f(D)| - F(\varepsilon)$ with probability at least $1 - \gamma$.

Remark 5.1. In machine learning, the goal is to find a function $f : X \to \{0,1\}$ from a class of functions Q that *best labels* a collection of labeled examples $(x_1,y_1),\ldots,(x_m,y_m) \in X \times \{0,1\}$. (Examples $(x,0)$ are known as *negative examples*, and examples $(x,1)$ are known as *positive examples*). Each example x_i has a *true* label y_i, and a function f *correctly labels* x_i if $f(x_i) = y_i$. An *agnostic learning algorithm* for a class Q is an algorithm that can find the function in Q that labels all of the data points approximately as well as the best function in Q, even if no function in Q can perfectly label them. Note that equivalently, an agnostic learning algorithm is one that maximizes the number of positive examples labeled 1 minus the number of negative examples labeled 1. Phrased in this way, we can see that a *distinguisher* as defined above is just an agnostic learning algorithm: just imagine that x contains all of the "positive" examples, and that y contains all of the "negative examples." (Note that it is ok of x and y are not disjoint — in the learning problem, the same example can occur with both a positive and a negative label, since agnostic learning does not require that any function perfectly label every example.) Finally, note also that for classes of linear queries Q, a distinguisher is simply an optimization algorithm. Because for linear queries f, $f(x) - f(y) = f(x - y)$, a distinguisher simply seeks to find $\operatorname{argmax}_{f \in Q} |f(x - y)|$.

Note that, *a priori*, a differentially private distinguisher is a weaker object than a differentially private release algorithm: A distinguisher merely finds a query in a set Q with the approximately largest value, whereas a release algorithm must find the answer to every query in Q. In the algorithm that follows, however, we reduce release to optimization.

We will first analyze the IC algorithm, and then instantiate it with a specific distinguisher and database update algorithm. What follows is a formal analysis, but the intuition for the mechanism is simple: we simply run the iterative database construction algorithm to construct a hypothesis that approximately matches x with respect to the queries Q. If at each round

our distinguisher succeeds in finding a query that has high discrepancy between the hypothesis database and the true database, then our database update algorithm will output a database that is β-accurate with respect to Q. If the distinguisher ever fails to find such a query, then it must be that there are no such queries, and our database update algorithm has already learned an accurate hypothesis with respect to the queries of interest! This requires at most T iterations, and so we access the data only $2T$ times using $(\varepsilon_0, 0)$-differentially private methods (running the given distinguisher, and then checking its answer with the Laplace mechanism). Privacy will therefore follow from our composition theorems.