大家好

Hello, everyone.

我是Yupeng Zhang 来自马里兰大学

I am Yupeng Zhang, from the University of Maryland.

今天 我要讲解我们撰写的论文

Today, I'm going to present our paper:

安全机器学习：一种可扩展隐私保护机器学习系统

SecureML: A System for Scalable Privacy-Preserving Machine Learning.

这个工作由我和来自VISA研究院的Payman共同完成

This is a joint work with Payman from VISA research.

现今 机器学习已被用在各个领域中 且引发了各个领域的变革

Machine learning is used everywhere these days, and always lead to breakthroughs in a lot of areas.

例如 机器学习可以用于图像处理、语音识别、异常检测、甚至下围棋

For example, it can be used for image processing, speech recognition, fraud detection and even playing the game Go.

机器学习之所以在实际中的应用效果如此之好

And the reason why it performs so well in practice

是因为我们使用了大量的数据来训练机器学习模型

is because we use a large amount of data to train these machine learning models, OK?

虽然机器学习引发了变革 此技术也引入了安全问题

And despite of these breakthroughs, it also leads to security issues,

我们应该如何保护数据的隐私性？

namely how do we protect the privacy of our data?

这里需要澄清一点 这里所指的隐私性与前面讲座中的隐私性不太相同

So, to clarify this, this privacy is something different from the previous talk.

这里我们考虑的是用于训练机器学习模型的数据 如何保证这些数据的隐私性

Here we are considering the privacy of the data used for the training purpose,

毕竟作为终端用户 我们不想向公司分享我们的数据 让它们可以运行机器学习算法

because as the end user, we do not want to share our sensitive information with these companies for them to run the machine learning algorithms, OK?

我们应该如何解决这个问题？

So, how do we solve this problem?

隐私保护机器学习为这类安全问题提供了一个解决方案

Privacy-preserving machine learning provides a solution for this security issue.

它允许公司执行相同的机器学习算法 但不需要得知用户的实际数据

It enables a company to still perform the same machine learning algorithm without knowing the underlying content of users’ data.

这样一来 用户仍然可以获得机器学习算法带来的益处

So, in this way, we can still gain the benefits of machine learning

同时可以实现数据隐私保护、不将数据泄露给公司

while protecting our data privacy, without leaking the information to the companies.

隐私保护机器学习研究领域已经有了很多的前置学术成果

And there are a lot of great prior works in the literature of privacy-preserving machine learning.

幻灯片上列举了其中一些论文

And some of them are listed on the slide.

这是一个非常前沿 进展速度很快的研究领域

And this is a very active line of research.

在我们的论文中 我们聚焦于下述安全模型：双服务器模型

So, in particular, in our paper, we focus on the following security model, called two-server model.

在这个模型中 我们假定两个服务器分别属于两个不同的公司

In this model, we assume there are two servers that belong to two different companies.

两个服务器不会实施共谋攻击

And they do not collude with each other.

作为终端用户 我们首先将数据拆分成两个分享值 分别将分享值发送给两个服务器

As an end-user, we will first split our data into two shares, and send each share to one server.

这样 单一服务器无法得到原始数据任何信息 因为它只能得到其中一个分享值

In this way, a single server cannot learn anything about the original data, because he only has access to a single share.

随后 两个服务器互相交互 执行两方安全计算 生成机器学习模型

After that, the two servers interact with each other to perform a two party secure computation, and generate the model, OK?

这一安全模型的优点在于

And the benefits of this security model is that,

首先 此模型将多方安全计算过程归约为两方安全计算过程 以此大幅提高计算效率

first, it reduces the problem from multi-party computation to two-party computation, which improves the efficiency significantly.

其次 上传数据后 用户即可处于离线状态 模型训练过程中用户不需要与服务器交互

second, the users can go offline after submitting the data, and do not need to interact with the server during the training.

再有 此模型也可以解决下述问题：两个公司想共同训练模型 但不想将数据分享给对方

In addition, it also captures the scenario where the two companies want to jointly train a model without sharing their data with each other.

很多前置学术成果都使用了这一安全模型

And the model is widely used in many prior works.

本篇论文的主要贡献是

The contribution of our paper is that,

在此安全模型下 我们提出了新的协议 支持隐私保护线性回归、逻辑回归、和神经网络

under this security model, we propose new protocols for privacy preserving linear regression, logistic regression, and neural networks.

特别地 我们综合使用了秘密分享、预计算三元组代数运算、以及混淆电路技术

In particular, we are using a combination of secure sharing and arithmetic on shared values with pre-computed triplets, plus garbled circuits.

从实现角度看 我们的系统与前置工作相比 效率有了量级上的提升

On the implementation side, we show that our system is orders of magnitude faster than prior work,

我们的系统支持大数据集模型训练 可以支持百万量级的数据集 五千个特征

and scales to large datasets with up to a million records and 5,000 features.

本次讲座 我们主要关注线性回归和逻辑回归

And in this talk, I'm going to focus on linear regression and logistic regression.

相关技术可以推广到神经网络模型训练中

And the technique can be generalized to train neural networks.

大家可以阅读我们的论文 以了解更多的技术细节

And you can refer to our paper for more details about it.

首先 什么是线性回归？

So, first, what is linear regression?

假设我们将数据点和与之相关的结果值画在图中 如幻灯片所示

Imagine we draw the data points and their corresponding values are playing like this.

线性回归要根据图中的点 尝试拟合出一条与点尽可能吻合的线

A linear regression is trying to fit a line across these data points, OK?

形式化地讲 输入是数据-值对x和y

More formally speaking, the input is the data-value pairs of x and y.

这里x可以是一个向量 我们也称x为特征值

Here x can be a vector, and is also known as features.

y是一个单值 我们也称y为标签值

And y is a single value for value, and also known as a label.

输出的模型w是一个系数向量 其维度大小与输入x的维度大小相同

The output model w is defined by a vector of coefficients that has the same dimension with the input x,

我们要求w和x的内积结果应该与y值非常近似

such that the inner product between w and x should be approximately equal to the value y.

实际上 模型定义了x和y之间的线性关系

This defines a linear relationship between x and y.

为了训练模型 我们这里引入随机梯度下降算法 简称SGD算法

To train the model, here I am introducing an algorithm called stochastic gradient descent, SGD.

我们把这个问题看成一个最优化问题

We view the problem as an optimization problem,

尝试找到一个最佳模型w 使得y\*和y的距离取得最小值

trying to find the best model w to minimize the distance between y\* and y.

算法首先将w初始化到一个随机位置上

And it begins by initializing w at a random location.

随后 算法从数据集中选取一个随机样本

And then, select a random sample from the data set.

算法根据当前模型w计算预测值 并将结果与正确的标签值进行比较

It computes the prediction based on the current model w, compare it with the correct label,

比较结果将告诉算法 应该往哪个方向移动 以得到最优解

and this will tells you which direction to move towards the optimal location.

可以证明 如果重复上述步骤 最终模型会收敛到最优位置

And it can be proved that if you do it repeatedly, eventually the model will converge to the optimum location.

这就是SGD算法

This is SGD.

对于线性回归这一特殊算法 更新函数可以用幻灯片上的公式表示

And in particular, for linear regression, the update function is this formula as shown on the slide.

正如大家所看到的 更新公式非常简单

As you can see, this update formula is very simple.

更新公式只涉及到乘法运算和减法运算

It only involves several modifications and subtractions.

因此 一种很自然地实现隐私保护线性回归的方法是

Therefore, a natural way to do privacy-preserving linear regression is that

直接将秘密分享与分享值代数运算直接应用到线性回归算法上

how about we just apply the secret sharing and arithmetic on shared values directly?

这应该可以解决隐私问题

That should solve the problem.

整个场景描述如下

And the scenario is that following.

用户首先将数据和标签值进行秘密分享

Users first secret shared data and values.

服务器随机初始化模型 同样对模型执行秘密分享

And the servers initialize the model w randomly, and also secret shared them.

随后 我们直接应用预计算的三元组 一遍又一遍地在分享值上执行更新函数

After that, we simply run this formula again and again using arithmetic on secret shared values with pre-computed triplets.

这就能解决问题了

And problem solved.

但这里有一个很大的问题

But there's a big issue here.

因为秘密分享和分享值代数运算只能在整数域上执行 例如在模质数下的整数域执行

Because secret sharing and this arithmetic only works on integer fields, for example integers modulo large prime,

但只有当参数带小数时 线性回归和SGD算法才能正确执行

but the linear regression and SGD only makes sense if the numbers are represented by decimal numbers.

我们如何在整数域上实现带小数的运算？

How can we support decimal number operations?

这就引出了我们的第一个贡献

This leads to our first contribution.

我们给出了一种方法 可以在整数域上直接执行带小数乘法

We show a way to support decimal multiplications directly in integer fields.

具体思想如下

And the idea is that following.

考虑存在两个幻灯片上所示的带小数点的数 我们知道这两个数对应的明文

Think about there are two decimal numbers like this. And you know the plaintext of then.

随后 在不丧失计算精度的条件下将两个数相乘

Then, you multiply them together by keeping the full accuracy.

我们可以得到结果c 其中c的小数部分为原来的两倍

You will get a number c that double with the size of the decimal part, right?

这里我们假定整数部分足够大 不会超过有限域的范围

And here, I'm assuming the integer part is big enough and does not grow.

如果不考虑小数点 小数部分的乘法运算和整数部分的乘法运算完全一致 这很不错

So, this step is exactly the same as integer multiplications by removing the decimal point, which is good,

问题在于c会变长

but the size of c is keep growing.

如此计算下去 c的长度会越来越长 最终超过整数域的范围 最终导致溢出

Sooner or later, it will be larger than the size of the field, and you have overflow.

解决此问题的一种直观方法是进行截断 即直接扔掉c的小数点最后几位

So, a natural way to solve this is by truncation, which means simply cut last significant bits of c, throw them away, OK?

这样一来 c的长度就和a、b相同了

In this way, c has the same size of a and b.

这一方法称为定点乘法

And this is known as fixed point multiplication.

在论文中我们证明：可以在分享值上应用相同的截断技巧

So, in our paper, we further show that this truncation trick still works even if the numbers are secret shared.

具体来说 这里我们有两个服务器上分享的a和b

In particular, here we have a and b secret shared between the two servers,

也就是说 这两个数分别被一个大整数域上的大随机数所遮盖

which means they're masked by a large random number in a much larger field.

随后 我们应用预计算三元组执行乘法操作 得到c的分享值 即c\_0和c\_1

And then, you perform multiplication using the pre-computed triplets, which gives you the shares of the result c,

c\_0和c\_1中编码了全精度乘法计算结果

c\_0 and c\_1 that encodes the number with full accuracy.

随后 两个服务器分别独立地对c\_0和c\_1进行截断 此过程不引入任何通信开销

After that, you truncate each individual share independently without any communication.

我们证明截断后 应用两个分享值仍然可以以很高的概率恢复出定点乘法的计算结果

And we show that after truncation, these two shares can reconstruct to the exact same answer of fixed point multiplication

只不过计算结果的最后一位小数上会增加一个非常小的误差值

plus a small error on the last decimal bit with high probability.

这就是我们的技术方案

That's our technique.

应用此技术方案 回到协议层面上 每一次乘法运算中 我们都对结果分享值简单截断

With this technique, back to the protocol, after every multiplication, we simply need to perform truncations on the shares.

这样就完成了整个隐私保护线性规划协议的实现

And that completes the whole protocol for privacy-preserving linear regression, OK?

这里向大家展示截断技术的应用效果

And to show the effect of our technique,

由于此技术只在计算结果的最后一个比特中引入了非常小的噪声

as we are only introducing a very small error on the last decimal bit,

因此 整个计算过程的执行时间几乎和在明文上应用小数执行整个计算过程的时间相同

it behaved almost exactly the same as running the training on the plaintext data using decimal numbers.

具体来说 在各种不同的场景下 我们所提出的技术要比定点乘法混淆电路快4-8倍

And in particular, our technique is 4-8 times faster than fixed point multiplication garbled circuits in different settings.

线性回归部分就这些内容

That's all for linear regression.

下一部分 逻辑回归

Next, logistic regression.

逻辑回归主要用在分类问题上

Logistic regression works for classification algorithms,

我们要尝试将数据分成两个类型

where you are trying to divide your data point into two categories.

形式化地讲 逻辑回归中的数据-标签值对和线性回归相同

And formally speaking, now the data-value pairs is as before.

但在逻辑回归中 y是一个比特值 取值为0或者1 分别表示两种分类结果

And now, y should be binary, either 0 or 1, representing two categories.

逻辑回归和线性回归的区别是 我们要在内积结果上进一步执行一个额外的函数f

And the difference between logistic regression and linear regression is that we want to further apply an extra function f on top of the inner product.

此函数f一般称为激活函数

And this f is known as the activation function.

逻辑回归中的激活函数f为1/(1+e^(-u))

For logistic regression, the f is in this form, 1/(1+e^(-u)).

函数图像如幻灯片所示

And the shape is shown in the plot.

我们仍然可以使用SGD算法来训练模型

To train a model, we still can use SGD algorithm.

令人惊讶的是 逻辑回归的更新函数几乎与线性回归完全相同

And surprisingly, the update function is almost the same as linear regression,

唯一的区别是我们要在内积结果上额外调用一次激活函数f

except that we apply this extra f on top of the inner product.

更新函数的其它地方都与线性回归完全相同

Everything else is the same as linear regression,

这意味着如果我们可以通过安全多方计算的方式计算函数f

which means if we have a way to compute this function f in secure computation,

我们就可以把这个计算过程应用到原始线性回归协议中 即可实现逻辑回归

we can add it to the original protocol for linear regression and that's it.

但事实证明 实现这一步骤会面临巨大的挑战

But it turns out this is quite challenging,

因为函数f涉及到精确到小数的自然对数求幂

because the function f involves decimal exponentiation with a natural base.

如何实现此计算过程？

How can we compute this?

传统方法是应用所谓的多项式近似方式

A traditional way is through something called polynomial approximation.

幻灯片给出了10阶多项式近似激活函数f的图像

And here's the curve using a degree 10 polynomial.

正如大家所看到的 近似图像与逻辑回归激活函数非常接近

As you can see it's quite close to the logistic function.

但是通过安全计算方式实现近似函数的计算 会引入较大的计算开销

But it introduced a big overhead to evaluate this polynomial in secure computation,

因为我们至少需要执行10次乘法计算 才能完成10阶近似多项式的计算过程

because you need to use at least 10 modifications to compute a degree 10 polynomial.

在我们的论文中 我们重点考虑 激活函数的作用究竟是什么

Instead in our paper, we think about what we need from this activation function.

因为我们要解决的是分类问题 我们实际需要的是一个值域为[0,1]的激活函数

What we really need is something bounded between 0 and 1, because it's working for classification problems.

并且此函数在0点附近应该大幅递增

And it's increasing in the middle.

那么 我们能不能用这样一个函数作为激活函数？

So, how about we use this function, OK?

我们证明 把此函数作为激活函数 训练出的模型准确性和原始逻辑回归函数准确性相同

We show that if using this function as activation function, we can almost get the same accuracy as the original logistic functions,

但更重要的是 我们可以应用混淆电路高效地通过安全多方计算方式实现此激活函数

but more importantly, it is very efficient to evaluate in secure computation using garbled circuit.

此激活函数只涉及到减法运算和与0比大小 后者本质上是查看最高位比特值是否为0

It only involves the subtraction and compared to 0, which means to take the most significant bit.

因此 这引出了我们论文的另一个贡献

Therefore, this brings us to the next contribution.

我们提出了一个新的概念：适用于安全多方计算的激活函数

We bring off a concept of secure computation friendly activation function.

我们不再通过已有方法近似计算激活函数

Instead of approximating existing activation functions,

我们后退一步 思考我们到底需要满足何种条件的激活函数

well, we just take a step back, think about what properties we need from these functions,

随后 我们尝试提出一个新的激活函数 其可以高效地通过多方安全计算的方式实现

and try to come up with new ones that can be efficiently computed in secure computations.

回到协议中来 我前面也讲到 我们只需要执行和线性回归相同的协议

Back to the protocol, as I promised before, you just run the exact same protocol for linear regression,

在计算内积结果后 我们转换到混淆电路上计算激活函数的结果 再切换回原始协议中

after the inner product, we switch to garbled circuit to evaluate this function, and then switch back to run the original protocol.

这就是隐私保护逻辑回归的完整协议了

And that's all for privacy-preserving logistic regression.

我们还在论文中引入了一些其它的优化方法

In addition, in our paper, we also introduced other optimizations,

如向量化 即所有计算过程都可以用矩阵形式表示

like vectorization, which means everything can be represented in a matrix form.

这种方式可以大幅提高计算效率

And this can improve the efficiency significantly.

进一步 这一技术可以推广到神经网络训练中

And also, the techniques generalized to neural networks.

最后 我给大家讲解我们的实验结果

Finally, I want to show you some experimental results.

我们在包含10万条数据、每条数据包含500个特征的数据集上进行了实验

So here, I'm showing you the results for 100,000 records and 500 features in each record.

我们的协议可以很自然地分为2个阶段

Our protocol can be naturally divided into two phases.

第一个阶段是与数据无关的离线阶段 此阶段要生成乘法三元组

The first phase is the data independent offline phase, to generate these multiplication triplets.

第二个阶段是在线阶段 此阶段要训练算法

And the second phase is the online phase to train the algorithm.

在局域网环境下 网络带宽为1.2GB/s 网络时延为0.17ms

In a LAN network, the bandwidth is 1.2 GB/s. And the delay is 0.17ms.

协议的离线阶段需要花费大约400秒

And the offline phase of our protocol takes around 400 seconds.

协议的在线阶段执行速度非常快

And the online phase of our protocol is extremely fast.

只需要花费1.4秒 之比明文数据训练慢2倍

It only takes 1.4s, which is only twice slower compared to plaintext training.

在广域网环境下 网络带宽为9MB/s 网络时延为72ms

In WAN network, where the bandwidth is 9MB/S, and the delay is 72ms,

离线阶段大约要花费9000秒 在线阶段要花费141秒

it takes around 9,000 seconds for offline phase, and 141 seconds for online phase.

即使在广域网环境下 我们系统的执行效率也要比前置工作快54倍

Even in the WAN setting, the performance of our system is 54x faster than systems in prior work.

进一步 我们观察到离线阶段是我们系统的性能瓶颈

Moreover, we observe that the offline phase is the bottleneck in our systems.

我们进一步提出了一个替代方案 在用户的帮助下生成乘法三元组

So, we further provide an alternative way to generate these triplets with the help of the users.

应用这一方案 我们可以大幅降低离线阶段的时间开销 稍稍提高在线阶段的时间开销

In this way, we can significantly reduce the offline time while increasing the online time a little bit.

不过此方案会减弱系统的安全模型

In addition, this also weakens the security model.

如果应用此方案 我们需要进一步假设客户端不与任意一个服务器实施共谋攻击

Here, we further assume that the clients cannot collude with any of the two servers, OK?

接下来是逻辑回归的实验结果

Next, results for logistic regression.

正如我前面所讲到的那样 我们协议的一大优势在于

As I presented before, the good thing about our protocol is that

与线性回归相比 逻辑回归不会额外使用更多的预计算乘法三元组

the logistic regression consumes no extra pre-computed multiplication triplets compared to linear regression.

因此 逻辑回归的离线阶段时间消耗与线性回归离线阶段的时间消耗完全相同

Therefore the offline phase is exactly the same as linear regression.

在线阶段 我们需要进一步执行一次混淆电路 并引入一次额外的信息交互

In the online phase, we need to evaluate an extra garbled circuit, and introduce an extra round of interaction,

额外增加的时间消耗如幻灯片所示

which increase the online time by this much.

总消耗时间与线性回归仍处在同一个量级

It's still on the same order compared to linear regression.

据我们所知 这是在此安全模型下第一个实现隐私保护逻辑回归的研究成果

To the best of our knowledge, we are the first to implement a system for privacy-preserving logistic regression under this security model.

此系统可以支持100万条数据集 每条数据集包含5000个特征

I can scale to up to a million records with 5,000 features in each record.

最后是神经网络

Finally, neural networks.

我们事先了包含2个隐藏层 每层包含128个神经元的神经网络

We implemented the neural network with two hidden layers and 128 neurons each.

这里我们直接给出端到端的性能测试结果 综合考虑了在线阶段和离线阶段

And here I am presenting the end-to-end performance including both online and offline.

在局域网环境下 训练此神经网络的时间大约为25,000秒

It takes 25,000 seconds to train this neural network in the LAN setting,

训练所消耗的时间是明文训练所消耗时间的35倍

which has a 35 times overhead compared to plaintext training.

在广域网环境下 性能会变得更加糟糕

In the WAN setting, the performance is much worse.

对于此量级的数据集 总训练时间约为200,000秒

For this size of data set, it approximately will take 200,000 seconds in total.

最后是总结

To sum up.

本文提出了一个新的协议 实现隐私保护线性回归、逻辑回归、神经网络

In this paper, we present new protocols for privacy-preserving linear regression, logistic regression, and neural networks.

特别地 我们引入了一种新的方法 可以直接在整数域下实现带小数乘法

In particular, we introduced a way to support decimal multiplication directly on integer fields.

我们提出了一个容易通过安全多方计算方式实现的激活函数

And we propose secure computation friendly activation functions.

我们引入了向量化优化方式

And we introduced vectorization.

从实现角度看 我们系统的执行效率比前置工作高几个数量级 可以支持大数据集训练

On the implementation side, our system is orders of magnitude faster than prior work, and can scale to large data sets.

这就是我讲座的全部内容了 谢谢大家

That's all of my talk, thank you.

我们台下有两个麦克风 听众可以用麦克风提问

We have two mics for questions from the audience.

让我来问个问题吧

So, let me ask a question.

在SGD算法下 你们给出了三个技术提高了安全多方计算场景下SGD算法的效率

So, with three principal techniques for adapting this task to the to the SGD setting,

你是否可以介绍一下 每个技术分别对算法提供了多大的优化量？

can you break down which technique was contributed what to the protocol?

我们在论文中给出了详细的基准测试结果

OK, we have a very detailed micro benchmarking in the paper.

简单总结一下 与通用方案相比 每个独立的技术都将算法速度提高了10倍左右

And to summarize a little bit, each one individually introduced approximately 10 times speed up compared to the generic approach.

但需要把这几个技术组合起来使用

But all these things must be used together.

例如 带小数乘法运算不能用在混淆电路上

For example, this decimal thing cannot be used with garbled circuits.

你只能在分享值代数计算过程中应用带小数乘法计算的相关优化技术

And you must be integrated into this arithmetic on shared values.

把各技术综合起来 算法的总执行效率会提高好几个量级

And in total, we have speed up of several orders in magnitude.

非常感谢

Thank you.

你好 我是John Percival 来自罗切斯特大学

Hi, John Percival, University of Rochester.

我有个很简单的问题

Just a simple question.

在广域网实验中 你们是通过仿真完成的实验 还是在真实的广域网环境下完成的实验？

For your experimental setup, for the wide area network experiment, is that simulation or did you actually use a real wide area network?

我们在亚马逊的机器上实现了我们的方案

Yeah, actually I implemented on the Amazon machine.

一台机器位于美国东海岸 一台机器位于美国西海岸

One is located in US east coast, and the other one is located in US west coast.

因此这是个真实的实验 没有仿真过程

And it's the real experiment, no simulation.

好的 谢谢

OK, thank you.

好的 我们再次对演讲者表示感谢

OK, let’s thank the speaker again.