Uma versão não-interativa do k-NN sobre dados cifrados

Hilder V. L. Pereira, Diego F. Aranha

Instituto de Computação – Universidade Estadual de Campinas (Unicamp)
Av. Albert Einstein, 1251 – CEP 13083-852, Campinas/SP – Brazil

hilder@lasca.ic.unicamp.br, dfaranha@ic.unicamp.br

Abstract. *Machine learning tasks typically require large amounts of sensitive data to be shared, which is notoriously intrusive in terms of privacy. Outsourcing this computation to the cloud requires the server to be trusted, introducing a non-realistic security assumption and high risk of abuse or data breaches. In this paper, we propose a non-interactive privacy-preserving version of the k-NN classifier, combining order-preserving encryption and homomorphic encryption. According to our experiments, the privacy-preserving variant achieves the same accuracy as the conventional k-NN classifier, but considerably impacts the original performance. The performance penalty is however still viable for practical use when the security properties provided by the approach are examined in detail. In particular, the cloud server does not need to be trusted beyond correct execution of the protocol and computes the algorithm over encrypted data and encrypted classes, never learning the real dataset values, the number of classes, the query vectors or their classification.*

Resumo. *Tarefas de aprendizagem de máquina normalmente exigem que grandes quantidades de dados sensíveis sejam compartilhados, o que é notoriamente intrusivo em termos de privacidade. Terceirizar esta computação para a nuvem requer que o servidor seja confiável, introduzindo um requisito de segurança não realista. Neste trabalho, propomos uma versão do classificador k-NN que pode ser executada na nuvem sobre dados cifrados de uma forma não interativa, combinando cifração que preserva a ordem e criptografia homomórfica. De acordo com nossos experimentos, a versão sobre dados cifrados alcança a mesma precisão que a convencional, mas com impactos consideráveis no desempenho original. Contudo, a penalidade de desempenho não é proibitiva e a solução continua viável para uso prático, quando as propriedades de segurança adicionais fornecidas são examinadas em detalhe. Em particular, o servidor em nuvem não precisa ser confiável para além da execução correta do protocolo e, como todos os dados recebidos são cifrados, o servidor não aprende os valores do conjunto de dados, o número de classes, os vetores a serem classificados nem as classes a eles atribuídas.*

1. Introdução

Com corporações e governos cada vez mais intrusivos na suas coletas de dados e esforços de vigilância, e também com os recorrentes vazamentos de dados observados nos últimos anos, o paradigma de computação em nuvem enfrenta um novo desafio para permanecer como o modelo de computação vigente. Os baixos custos operacionais e a alta disponibilidade de capacidade de armazenamento e de poder computacional podem não parecer tão

atraentes depois que os riscos de terceirizar a computação e o armazenamento de dados são considerados, especialmente no caso de aplicações sensíveis. Por exemplo, não há garantias formais de que o provedor dos serviços em nuvem não está se comportando de forma abusiva ou intrusiva, ou até mesmo que a infra-estrutura se encontra devidamente protegida contra ataques externos. Diferentes regimes jurídicos e influência governamental introduzem outras complicações para o problema. O impacto financeiro a longo prazo da atual crise de confiança na computação em nuvem é estimado entre 35 e 180 bilhões de dólares nos EUA apenas em 2016 [Miller 2014].

Uma solução proposta para amenizar estas questões consiste na *computação sobre os dados cifrados*. Neste modelo, os dados são cifrados com esquemas que preservam algumas propriedades dos textos claros (originalmente chamados de *homomorfismos privados* [Rivest et al. 1978]), o que permite que algumas operações ainda sejam realizadas no domínio cifrado. Construções que fornecem esse recurso e suportam um número arbitrário de operações (adições e multiplicações) são classificados como *criptografia totalmente homomórfica* e, geralmente, introduzem uma penalidade de desempenho tão grande que torna suas aplicações práticas inviáveis. A *criptografia parcialmente homomórfica ou homomórfica em nível* impõe um limite no número e nos tipos de operações que podem ser realizadas sobre dados cifrados, mas apresenta desempenho muito superior. No entanto, exigem uma reformulação dos algoritmos a serem executados homomorficamente para que estes satisfaçam as restrições impostas por tais esquemas criptográficos.

Tradicionalmente, computação sobre dados cifrados já havia sido aplicada à contagem de votos de forma secreta em eleições eletrônicas, sem a necessidade de revelar os votos que sendo somados [Hirt and Sako 2000], mas esquemas de criptografia homomórfica modernos podem permitir uma série de aplicações e serviços interessantes nos anos futros, como processamento do genoma de forma privativa ou coleta de dados de saúde e diagnósticos que preservam a privacidade dos pacientes [Naehrig et al. 2011]. Mineração de dados [Lindell and Pinkas 2009] e aprendizagem de máquina [Dowlin et al. 2016] são aplicações naturais, pois são serviços já oferecidos na nuvem por empresas como Amazon[1], Google[2] e Microsoft[3]. Tais serviços envolvem quantidades consideráveis de dados a serem compartilhados e manipulados em plataformas não confiáveis.

Alguns trabalhos anteriores na literatura têm tratado do problema de executar algoritmos de classificação sobre dados cifrados. O problema é fundamental em si mesmo e uma solução para ele pode ser usada como sub-rotina de vários algoritmos de aprendizagem de máquina. Um algoritmo de classificação utiliza dados já classificados manualmente (conjunto de treinamento) para inferir a qual classe de dados uma nova amostra pertence. Tal amostra é chamada de vetor de consulta. [Graepel et al. 2013] adaptaram algoritmos de classificação simples para trabalhar sobre dados cifrados usando criptografia homomórfica, enquanto [Bost et al. 2014] consideraram classificadores mais complexos, como árvores de decisão e inferência Bayesiana para aplicações médicas. Outros autores projetaram protocolos para realizar agrupamentos (*clustering*) sobre dados cifrados [Jha et al. 2005] baseando-se em soluções da computação multi-parte (portanto, interativas).

[1] `https://aws.amazon.com/machine-learning/`
[2] `https://cloud.google.com/products/machine-learning/`
[3] `https://azure.microsoft.com/en-us/services/machine-learning/`

A principal contribuição deste trabalho é uma versão do classificador k-NN que preserva a privacidade, sendo executado sobre dados cifrados usando criptografia homomórfica e criptografia que preserva a ordem. Os protocolos são não-interativos, ou seja, os clientes e a nuvem não interagem durante a classificação, e, portanto, são ideais para o cenário de computação em nuvem, no qual o armazenamento e o processamento são delegados de um agente com poucos recursos computacionais para servidores poderosos. Os resultados experimentais demonstram que não há perdas significativas de precisão e a análise de segurança é feita no modelo semi-honesto. Até onde vai o conhecimento dos autores, esta é a primeira versão não-interativa e compatível com computação em nuvem do classificador k-NN sobre dados cifrados.

Este artigo é organizado da seguinte forma: na Seção 2, o classificador k-NN convencional é revisitado e o problema de classificação que preserva a privacidade é definido. Na Seção 3, os conceitos de criptografia homomórfica e de criptografia que preserva a ordem são apresentados. Os algoritmos propostos são discutidos na Seção 4. A Seção 5 apresenta os resultados experimentais comparando, sobre seis diferentes bases de dados, a versão tradicional com as versão que preserva a privacidade. Na Seção 6, as propriedades de segurança oferecidas pelo esquema são discutidas brevemente. Os trabalhos relacionados são discutidos na Seção 7 e as conclusões se encontram na última seção.

2. Definição do problema

Nesta seção, define-se o problema de classificação que o k-NN foi projetado para resolver e discute-se uma variante desse problema levando em conta a privacidade e o modelo de processamento centralizado na nuvem.

2.1. O classificador *k*-NN

O classificador *k-Nearest Neighbor* (k -NN), traduzido normalmente como k vizinhos mais próximos, é um método supervisionado não-paramétrico que serve para classificar uma instância com base nas classes de seus vizinhos mais próximos [Alpaydin 2004, Altman 1992]. Cada instância é representada por um vetor em $\mathbb{R}^p$ (ou seja, tem p atributos) e um rótulo associado, chamado de classe da instância. Um vetor de consulta é uma instância que ainda não tem um rótulo e que deve ser classificada pelo k-NN usando as n instâncias já classificadas.

O classificador k-NN tem um parâmetro k, que é inteiro positivo que representa o número de vizinhos mais próximos considerados ao classificar uma nova instância. Também há uma função d de $\mathbb{R}^p \times \mathbb{R}^p$ para $\mathbb{R}$, que determina a distância entre os duas instâncias (a distância euclidiana é frequentemente utilizada).

Para classificar uma nova instância $x \in \mathbb{R}^p$, o classificador k-NN executa o procedimento a seguir:

1. Encontra os k vizinhos mais próximos entre todas as instâncias classificadas $u_1, u_2, ..., u_n$, ou seja, as k instâncias cujas distâncias $d(u_i, x)$ são as menores.
2. Seleciona a classe mais frequente entre as classes dos k vizinhos mais próximos e a atribui à x.

Há também uma variante do k-NN conhecido como k-NN *ponderado pela distância*, ou simplesmente k-NN *com pesos*. Nesta versão, ao invés de simplesmente

atribuir a classe mais frequente entre os k vizinhos mais próximos, o inverso da distância é usado como o peso do voto de cada um desses k vizinhos.

2.2. k-NN privativo na nuvem

O problema *k-NN não interativo que preserva a privacidade* é definido como o problema de processar o k-NN em uma plataforma não confiável sem a ajuda do proprietário dos dados e sem revelar os valores das instâncias, suas classes, os vetores a serem classificados e as classes atribuídas para eles. Por plataforma não confiável, entende-se uma terceira parte que retém os dados de alguma forma, mas não é confiável sob o ponto de vista do proprietário dos dados.

O cenário considerado neste trabalho supõe um modelo de cliente e servidor, no qual o proprietário dos dados é o cliente e o servidor é um provedor de serviços em nuvem. O cliente pretende armazenar os dados na nuvem e processá-los de uma forma *não-interativa*, isto é, a nuvem deve se comunicar com o cliente apenas para receber os dados e os vetores a serem classificados, mas não deve se comunicar com o cliente durante o processamento. Também assume-se que o cliente tem recursos computacionais restritos, como processamento e espaço de armazenamento.

O outro cenário possível é o de processamento distribuído, no qual o cliente interage com a nuvem (ou com as outras partes envolvidas) recebendo e processando dados durante a fase de treinamento ou de classificação. Vale ressaltar que o modelo não interativo é mais conveniente para o cliente e é o modelo esperado quando se consideram serviços em nuvem.

3. Conceitos Fundamentais

Nesta seção, os dois esquemas de criptografia usados para garantir a propriedade de preservação da privacidade do classificador proposto são apresentados.

3.1. Criptografia que preserva a ordem

Criptografia que preserva a ordem (OPE[4]) é um esquema de criptografia simétrico e determinístico, cuja função de cifração preserva a ordenação numérica dos textos claros [Boldyreva et al. 2009]. Em outras palavras, dados dois textos claros m_1 e m_2, e seus textos cifrados correspondentes, c_1 e c_2, cifrados com um esquema OPE, a seguinte implicação é verdadeira:

$$m_1 \leq m_2 \Rightarrow c_1 \leq c_2.$$

Como esse tipo de esquema trabalha sobre conjuntos finitos de valores numéricos, é suficiente descrevê-los usando o conjunto $D = \{1, 2, ..., M\}$ como o espaço de textos claros (ou domínio) e o conjunto $R = \{1, 2, ..., N\}$ como o espaço de criptogramas (ou contradomínio). Então, o esquema OPE usado aqui é parametrizado por dois inteiros positivos M e N, que representam o número possível de textos claros e de textos cifrados, respectivamente. O esquema é apresentado a seguir:

$\mathrm{OPE.keyGen}(M, N)$ é um algoritmo não determinístico que recebe os parâmetros M e N com $N > M$ e devolve a chave secreta $\mathcal{K}$.

[4]Do inglês *Order-preserving encryption*

OPE. $\mathrm{enc}(\mathcal{K}, m)$ é um algoritmo que cifra um valor $m \in D$ usando a chave $\mathcal{K}$ e devolve um criptograma $c \in R$.

OPE. $\mathrm{dec}(\mathcal{K}, c)$ é um algoritmo que decifra um criptograma $c \in R$ usando a chave $\mathcal{K}$ e devolve o texto claro correspondente $m \in D$.

Para um vetor $w = (w_1, ..., w_p)$, define-se a cifração componente a componente:

$$\text{OPE. enc}(\mathcal{K}, w) = (\text{OPE. enc}(\mathcal{K}, w_1), ..., \text{OPE. enc}(\mathcal{K}, w_p)).$$

Define-se da mesma forma a decifração de um vetor.

O espaço formado pelos vetores cifrados é chamado de *espaço cifrado*. Se um esquema OPE é usado para cifrar vetores, então a ordem é mantida em cada eixo. Portanto, é muito provável que cada vetor mantenha a sua vizinhança no espaço cifrado e esse fato é usado para fazer com que a nuvem consiga encontrar os vizinhos mais próximos de um vetor cifrado.

3.2. Criptografia Homomórfica

Um esquema criptográfico é homomórfico para uma operação $\star$ se é equivalente executar esta operação sobre textos claros ou sobre mensagens cifradas. Por exemplo, considerando a operação de adição, a soma de dois criptogramas cifrados por um esquema homomórfico gera uma terceira mensagem cifrada que corresponde à soma dos dois textos claros correspondentes.

O número de operações que podem ser realizadas *em sequência* sobre textos cifrados é geralmente limitada. Depois de aplicar mais operações do que o limite suportado pelo esquema, o texto cifrado não pode ser corretamente decifrado. Por exemplo, se o esquema é homomórfico para a multiplicação e suporta apenas dois produtos em sequência, então, dadas as mensagens cifradas c_1, c_2, c_3, e c_4, é possível avaliar produtos da forma $c_i \cdot c_j$ e $c_i \cdot c_j \cdot c_l$ (para $i, j, l \in \{1, 2, 3, 4\}$), mas não $c_1 \cdot c_2 \cdot c_3 \cdot c_4$, pois este último emprega três produtos em sequência.

Tais esquemas de criptografia são chamados de *parcialmente homomórficos* (SHE[5]). Esquemas totalmente homomórficos (FHE[6]) atuais não têm tais limitações, mas introduzem uma penalidade de desempenho tão grande que torna o seu uso proibitivo na prática [Naehrig et al. 2011].

Neste trabalho, o esquema SHE utilizado foi o YASHE, que pode ser descrito em termos das seguintes funções:

HE. $\mathrm{keyGen}(\mathbf{X})$ é o algoritmo que recebe o conjunto de parâmetros X e devolve a chave secreta sk e a chave pública pk.

HE. $\mathrm{enc}(\mathrm{pk}, m)$ é o algoritmo que cifra uma mensagem m usando a chave pública pk e devolve um texto cifrado c.

HE. $\mathrm{dec}(\mathrm{sk}, c)$ é o algoritmo que decifra um criptograma c usando a chave secreta sk e devolve o texto claro m.

HE. $\mathrm{add}(c_1, c_2)$ é o algoritmo que soma dois textos cifrados c_1 e c_2, devolvendo um terceiro texto cifrado que corresponde à cifração da soma $m_1 + m_2$.

[5]Do inglês *Somewhat Homomorphic Encryption*.

[6]Do inglês *Fully Homomorphic Encryption*.

$\mathrm{HE.prod}(c_1, c_2)$ é o algoritmo que multiplica dois textos cifrados c_1 e c_2 e devolve um terceiro criptograma que corresponde à cifração do produto $m_1 \cdot m_2$.

O algoritmo $\mathrm{HE.keyGen}$ recebe um parâmetro de segurança λ; dois inteiros t e n que determinam o espaço de mensagens, i.e. o anel $R = \mathbb{Z}_t[x]/\langle\Phi_n(x)\rangle$, onde $\Phi_n(x)$ é o n-ésimo polinômio ciclotomico; duas distribuições de probabilidade discretas e limitadas χ_e e χ_k sobre R; e dois inteiros positivos q e w. O anel R pode ser visto como um conjunto de polinômios de grau menor ou igual ao grau de $\Phi_n(x)$ e com coeficientes em $\mathbb{Z}_t = \{0, 1, ..., t-1\}$. As distribuições χ_e e χ_k são usadas para amostragem de ruídos (polinômios com coeficientes pequenos) que são usados na geração de chave e na cifração. A distribuição Gaussiana discreta é normalmente escolhida para a χ_e e uma distribuição sobre polinômios com coeficientes pequenos (normalmente, coeficientes em $\{-1, 0, 1\}$) é escolhida para χ_k. Há também um parâmetro implícito L, a profundidade multiplicativa suportada pelo esquema. Ao fixar L e t, pode-se derivar os outros parâmetros, pois eles têm que respeitar algumas restrições para garantir a corretude e segurança do esquema.

4. k-NN não-interativo sobre dados cifrados

A versão privativa proposta para o k-NN pode ser dividida em quatro rotinas principais: inicialização, requisição, processamento e resposta.

- Codificação: essa função recebe a classe de uma instância $l \in \mathbb{N}$ e devolve o monômio x^l.
- Inicialização: o cliente tem n vetores $u_1, .., u_n \in \mathbb{R}^p$ e suas respectivas classes $l_1, ..., l_p \in \mathbb{N}$. Ou seja, cada vetor u_i é rotulado com um inteiro não negativo l_i. Esta sub-rotina cifra os vetores u_i usando o esquema OPE e codifica cada classe como um monômio para então cifrar usando o esquema de HE, conforme mostrado no Algoritmo 1. Note que esta etapa de inicialização é executada pelo cliente.

Algorithm 1 Inicialização

Entrada: $(u_1, .., u_n) \in \mathbb{R}^{n \times p}$, $(l_1, ..., l_n) \in \mathbb{N}^n$

$K = \mathrm{OPE.keyGen}(M, N, s)$
$(\mathrm{sk}, \mathrm{pk}) = \mathrm{HE.keyGen}(\text{ parâmetros })$
for $i = 1$ **to** n **do**
  $v_i = \mathrm{OPE.enc}(K, u_i)$
  $p(x) = \mathrm{encode}(l_i)$
  $c_i = \mathrm{HE.enc}(\mathrm{pk}, p(x))$
end for
Enviar $(v_1, v_2, .., v_n)$ e $(c_1, c_2, .., c_n)$ para o servidor na nuvem.

- Requisição: o cliente tem um vetor $w \in \mathbb{R}^p$ para ser classificado. Este vetor é cifrado utilizando o esquema OPE e então submetido para um servidor na nuvem.
- Processamento: A nuvem classifica um vetor cifrado y usando os vetores cifrados $v_1, v_2, .., v_n$ e suas respectivas classes (também cifradas) executando o Algoritmo 3. A função `indexesMinDistances` devolve os índices das k menores distâncias. Por exemplo, se o primeiro, o terceiro e o sétimo vetores fossem os

Algorithm 2 Requisição

Entrada: $w \in \mathbb{R}^p$

$y = \text{OPE.enc}(w)$
Escolher algum $k \in \mathbb{N}^*$
Enviar y e k para o servidor na nuvem.

três vizinhos mais próximos de y, então $d_1, d_3,$ e d_7 seriam as menores distâncias e essa função devolveria $(1, 3, 7)$.

Algorithm 3 Processamento

Entrada: vetor cifrado y, $k \in \mathbb{N}^*$

for $i = 1$ to n **do**
  $d_i = ||v_i - y||$
end for
$(i_1, .., i_k) =$ indexesMinDistances$(d_1, .., d_n)$
$class_y = c_{i_1}$
for $j = 2$ to k **do**
  $l = i_j$
  $class_y = \text{HE.add}(class_y, c_l)$
end for
Retornar $class_y$

- Resposta: o cliente recebe um texto cifrado $class_y$, decifra-o e extrai a classe do vetor y:

Algorithm 4 Resposta

Entrada: $class_y$

$c(x) = \text{HE.dec}(\text{sk}, class_y)$
$a_i =$ maxCoefficient$(c(x))$
Atribuir a classe i

A função funciona corretamente porque quando o servidor soma as classes cifradas no Algoritmo 4, está de fato gerando um polinômio que diz quantas vezes cada classe apareceu entre os k vetores mais próximos, pois a i-ésima classe é representada por x^i e a soma das classes resulta em um polinômio da forma $a_1x^1 + a_2x^2 + ... + a_sx^s$, onde cada coeficiente a_i representa o número de vezes que a i-ésima classe apareceu. Além disso, como as classes são cifradas usando um esquema de criptografia homomórfica, a nuvem pode somar classes cifradas e o criptograma resultante será decifrado para a soma das classes, como esperado.

5. Resultados experimentais

A versão privativa proposta para o k-NN foi implementada usando o esquema OPE apresentado em [Boldyreva et al. 2009] e o esquema homomórfico chamado

Tabela 1. Conjuntos de dados usados, número de instâncias (amostras no conjunto) e de atributos (variáveis de interesse).

CONJUNTO DE DADOS	INSTÂNCIAS	ATRIBUTOS
IRIS	150	4
WINE	178	13
CLIMATE MODEL (CM)	540	18
CREDIT APPROVAL (CA)	690	15
ABALONE	4177	8
WALL-FOLLOWING ROBOT (WFR)	5456	24

YASHE [Bos et al. 2013]. Ficam indicadas ao leitor as implementações públicas disponíveis em [Lepoint and Naehrig 2014] e [Dowlin et al. 2015]. A versão privativa foi avaliada usando conjuntos de dados do repositório de bases de dados para aprendizagem de máquina, da UCI[7]. Os conjuntos de dados são descritos na Tabela 1.

Todos os testes foram executados em uma máquina equipada com um processador Intel Xeon 2.6GHz, 30GB de memória RAM e o sistema operacional GNU/Linux Debian 8.0. É importante ressaltar que o consumo de memória ficou abaixo de 1GB durante toda a coleta dos resultados experimentais. A versão proposta do k-NN privativo foi implementada em C++ e compilada usando o GCC 4.9.2 fornecido pelo sistema operacional. A *flag* de otimização `-O3` foi usada. Para comparação, o k-NN implementado na biblioteca *Scikit Learn* do Python[8] foi usado como o k-NN convencional.

Os experimentos consistiram no processamento dos conjuntos de dados usando o k-NN convencional e a versão sobre dados cifrados, e na comparação das acurácias de classificação resultantes. Os resultados estão resumidos na Tabela 2. A tabela tem cinco colunas que representam o número de vizinhos mais próximos considerado ($k \in \{1, 3, 5, 7, 9\}$). A fim de verificar como os parâmetros do esquema OPE podem influenciar na precisão do k-NN, os conjuntos de dados foram cifrados usando vários pares de valores (M, N) (lembre-se que M e N determinam os tamanhos dos espaços de texto claro e de texto cifrado, respectivamente). Como não foram observadas diferenças significativas para as diversas combinações de parâmetros, são apresentados apenas os os resultados das execuções usando $(M, N) = (2^{32}, 2^{40})$. Como esperado, a versão que preserva a privacidade conservou a precisão do classificador original para quase todas as amostras.

O criptosistema YASHE foi instanciado com $\lambda = 80$, $w = 64$ e fixando parâmetros L e t para gerar n e q como descrito em [Lepoint and Naehrig 2014]. Como não é necessário realizar multiplicações homomórficas t deve ser um número maior do que o número máximo de vizinhos a ser considerado, é preciso escolher $t > k$ para lidar com o caso em que os k vizinhos mais próximos têm a mesma classe. Nesta situação, o valor 1 será somado ao mesmo coeficiente de polinômio a ser devolvido como classe pelo Algoritmo 3, fazendo com que o coeficiente se torne zero, já que as operações no anel R são tomadas módulo t. Portanto, foram escolhidos os parâmetros do YASHE como $L = 0$ e $t = 10$, o que resultou em $n = 541$ e $q = 33554467$ (um primo com 25 bits).

[7]UCI Repository: `https://archive.ics.uci.edu/ml/`
[8]`http://scikit-learn.org`

Tabela 2. Comparação das precisões entre a versão convencional do k-NN (CONV) e a versão privativa (PRIV) instanciada usando $M = 2^{32}$ e $N = 2^{40}$ como parâmetros do esquema OPE. Nenhum perda significativa de precisão foi observada na versão privativa.

	$k=1$		$k=3$		$k=5$		$k=7$		$k=9$	
BASE	CONV	PRIV	CONV	PRIV	CONV	PRIV	CONV	PRIV	CONV	PRIV
IRIS	0.960	0.960	0.980	0.980	0.960	0.961	0.960	0.960	0.960	0.970
WINE	0.847	0.830	0.796	0.796	0.779	0.779	0.796	0.780	0.745	0.730
CM	0.895	0.896	0.928	0.928	0.934	0.934	0.923	0.923	0.917	0.913
CA	0.633	0.619	0.685	0.685	0.680	0.680	0.746	0.746	0.746	0.737
WFR	0.883	0.882	0.875	0.875	0.868	0.869	0.855	0.855	0.837	0.837
ABALONE	0.591	0.583	0.612	0.618	0.621	0.620	0.628	0.630	0.628	0.627

Tabela 3. Comparação dos tempos de execução, em milissegundos, para classificar uma instância usando $k = 3$

BASE	CONVENCIONAL	PRIVATIVO
IRIS	0.020	1.30838
WINE	0.017	1.85090
CLIMATE	0.032	6.92744
CREDIT	0.044	7.66866
ABALONE	0.168	57.0700
WFR	0.180	83.0032

Uma comparação dos tempos de execução para classificar uma única amostra, usando $k = 3$, é mostrada na Tabela 3. Ressaltamos que mudanças no valor de k têm pouco efeito sobre os tempos de execução. Os conjuntos de dados foram divididos em um conjunto de treinamento contendo $\frac{2}{3}$ dos dados e um conjunto teste com o $\frac{1}{3}$ restante. Depois, a versão privativa foi executada 10 vezes e o tempo médio para classificar o conjunto foi computado. Foi realizada a mesma experiência na implementação convencional.

Para as bases testadas, a versão convencional do k-NN foi no máximo cerca de 400 vezes mais rápida que a versão sobre dados cifrados. Mesmo assim, as soluções propostas neste trabalho ainda podem ser consideradas eficientes se comparadas com soluções interativas, pois os custos de comunicação entre nuvem e cliente podem introduzir latências muito maiores dos que os tempos de execução observados.

A Tabela 4 apresenta os tempos de execução para cifrar cada conjunto de dados, incluindo conjuntos de treinamento e de teste. Ou seja, esses tempos correspondem à execução do procedimento de inicialização e consulta, sem considerar o custo de enviar os vetores para a nuvem (última operação do Algoritmo 2). Cifrar os conjuntos de teste é muitas vezes mais rápido do que cifrar o conjunto de treinamento porque, neste passo, não é necessário usar o esquema HE, que é mais lento do que o OPE.

Os vetores cifrados são representados por vetores de inteiros cujas entradas têm $\log_2(N)$ bits e, devido à nossa escolha de parâmetros, as classes cifradas são representadas por polinômios de grau até 540, com cada coeficiente tendo 16 bits. O comprimento,

Tabela 4. Comparação dos tempos de execução em segundos para cifrar os conjuntos de dados.

BASE	TESTE	TREINAMENTO
IRIS	0.214	0.037
WINE	0.621	0.154
CLIMATE	11.27	24.91
CREDIT	0.553	1.91
ABALONE	1.36	7.43
WFR	6.618	16.56

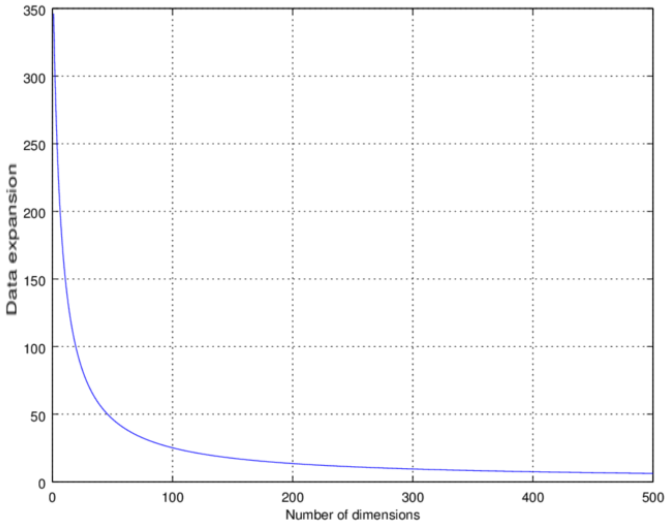

Figura 1. Expansão de dados como uma função do número de dimensões considerando os parâmetros M e N fixos

em bits, das entradas dos vetores originais (textos claros) é $\log_2(M)$ e as classes são representados por números inteiros. Portanto, considerando um conjunto de dados com n vetores P-dimensionais, a expansão de dados, definida como o tamanho máximo em bits dos dados cifrados sobre o tamanho máximo em bits de dados em texto claro é:

$$\frac{np\log_2 N + n540 \cdot 16}{np\log M + n32} = \frac{p\log_2 N + 8640}{p\log M + 32}$$

Note que o número de instâncias não afeta a expansão para armazenamento de dados e que os valores M e N têm pouco impacto na expansão, pois contribuem em escala logarítmica. Conforme o número de dimensões cresce, o quociente vai se aproximando de um. A Figura 1 mostra o efeito de variar p considerando $M = 2^{32}$ e $N = 2^{64}$ fixos.

6. Análise de Segurança

Assume-se neste trabalho que o servidor na nuvem satisfaz o modelo comumente chamado de *honesto, mas curioso* [Graepel et al. 2013], o que significa que o servidor

deve seguir o protocolo e executar o k-NN como esperado, devolvendo a resposta correta, mas pode tentar aprender informações durante a execução ou mais tarde tentar extrair informações dos criptogramas armazenados.

O servidor que executa o k-NN não aprende quais são os valores de qualquer coordenada de qualquer um dos vetores que ele recebe, incluindo os vetores de consulta, e também não aprende as classes associadas aos vetores. Como o esquema de criptografia homomórfica usado para cifrar as classes é não determinístico, o servidor não consegue determinar sequer quantas classes diferentes existem entre as classes cifradas. Como operações sobre mensagens cifradas resultam em outro texto cifrado bem formado, a classe atribuída ao vetor de consulta também não pode ser descoberta pelo servidor de nuvem. Por outro lado, a nuvem sabe o número de vetores e quantas entradas cada vetor tem por simples exame da quantidade de criptogramas recebidos.

Apesar disso, o esquema determinístico OPE introduz uma desvantagem: se a nuvem possui informações sobre a semântica das dimensões (qual variável é representada por cada componente dos vetores), e se também possui um conjunto de dados que está fortemente correlacionada com os dados cifrados, pode ser possível fazer ataques de inferência com base em análise de frequência [Naveed et al. 2015]. Por isso, nossas soluções são apropriadas apenas para a execução do k-NN sobre bases de dados privadas, sem informação pública sobre a distribuição dos seus dados.

7. Trabalhos relacionados

Vários trabalhos acadêmicos já estudaram problemas relacionados à execução do k-NN conservando a privacidade. No entanto, as soluções foram fornecidas para cenários diferentes, pois eram interativas, envolvendo processamento distribuído entre agentes com poder de computação similar; ou tratavam versões mais simples do problema, focando-se apenas na busca dos k vizinhos mais próximos e ignorando a etapa de classificação. Como resultado, a nossa proposta apresenta melhora qualitativa sobre esses trabalhos, fornecendo funcionalidades adicionais e garantias de privacidade correspondentes.

Os autores de [Zhan et al. 2005] consideraram um cenário conhecido como dados verticalmente particionados, em que cada uma de várias partes detém um conjunto de atributos das mesmas instâncias e querem realizar o k-NN na concatenação de seus conjuntos de dados. Eles propuseram um protocolo interativo no qual cada uma das partes precisa calcular as distâncias entre as instâncias em sua própria partição e um vetor de consulta e então combinar essas distâncias usando um esquema de criptografia homomórfica aditivo, com técnicas de perturbação aleatória, para encontrar os k vizinhos mais próximos. A etapa de classificação é finalmente executada localmente por cada uma das partes.

Em [Xiong et al. 2006, Xiong et al. 2007] os autores assumem que os proprietários de vários dados, cada um com uma base de dados própria, colaboram através da execução de um protocolo distribuído para executar o k-NN sem revelar uns aos outros os seus próprios dados. A classificação de uma nova instância é executada por cada usuário em seu próprio banco de dados e, em seguida, um protocolo distribuído seguro é usado para classificar a instância com base nos k vizinhos mais próximos de cada base de dados. Isso significa que o vetor de consulta é revelado e o processo é interativo, com carga de processamento para cada parte envolvida.

No artigo [Choi et al. 2014], os autores apresentam três métodos para encontrar os k vizinhos mais próximos preservando a privacidade dos dados, mas não abordam o problema de classificação. Além disso, os três métodos são interativos. É interessante notar que, mesmo que encontrar os k vizinhos mais próximos seja o passo principal do classificador k-NN, uma solução apenas para esse etapa implica em o cliente ter que armazenar, pelo menos, uma tabela que relacione os vetores no conjunto de dados e suas respectivas classificações, e implica também a necessidade de classificar os vetores de consulta localmente, depois de receber os k vizinhos mais próximos. Portanto, esse tipo de solução não é apropriada com o modelo de computação em nuvem.

O trabalho [Elmehdwi et al. 2014] considera um cenário diferente: o proprietário cifra os dados e os envia para um primeiro servidor, e envia a chave secreta para um segundo servidor. Assim, qualquer pessoa autorizada é capaz de enviar uma nova instância para o primeiro servidor, que executa um protocolo distribuído com o segundo servidor (este pode decifrar alguns dados nesse processo), e, finalmente, o primeiro servidor devolve os k vizinhos mais próximos. Observe que, novamente, a classificação não é executada, porque os autores estavam apenas interessados em encontrar os vizinhos mais próximos. Além disso, mesmo que o cliente não tenha que processar os dados, este método requer um servidor confiável para armazenar a chave privada, e esse servidor de confiança atua como o cliente no cenário de processamento distribuído. Basear-se em uma terceira parte confiável, naturalmente, apresenta um risco substancial adicional.

Os autores de [Zhu et al. 2013] propõem um cenário em que o proprietário dos dados os cifra e os envia para a nuvem, e então, outros usuários podem fazer requisições ao servidor em nuvem para obter, de forma segura, os vizinhos de vetores consulta arbitrários. A segurança do esquema é garantida graças a um protocolo interativo executado entre os usuários que querem realizar a consulta e o proprietário dos dados: esse protocolo gera uma chave usada para cifrar os vetores consulta e que não é apta a decifrar os vetores cifrados e armazenados na nuvem. Sendo assim, o proprietário dos dados ainda precisa participar da etapa de processamento, mesmo que seja apenas para gerar chaves, logo, esse protocolo pode ser classificado como interativo. Além disso, a classificação não é feita, apenas os vizinhos são encontrados.

Outra abordagem é proposta em [Wong et al. 2009], onde um novo esquema de criptografia chamado de criptografia que preserva o produto escalar é também proposto. Com esse esquema é possível encontrar os k vetores mais próximos sem a necessidade de um processo interativo, pois o servidor pode calcular produtos internos entre os vetores do conjunto de dados através do cálculo do produto interno dos vetores cifrados, determinando assim os vetores mais próximos do vetor de consulta. No entanto, novamente, os autores se preocuparam apenas com a tarefa de encontrar os vizinhos mais próximos, não com o problema de classificação. Além disso, um esquema de criptografia criado *ad hoc* para esta tarefa carece da extensa análise de segurança já realizada sobre sistemas criptográficos mais genéricos e bem estabelecidos.

8. Conclusão

Este trabalho apresenta uma versão não-interativa e privativa do classificador k-NN, e constatou-se por meio de avaliações experimentais que é suficientemente eficiente e precisa para ser viável na prática. O protocolo proposto combina criptografia homomórfica

e criptografia que preserva a ordem e é aplicável para a execução de classificações de instâncias usando dados cifrados armazenados na nuvem. Segundo consta, esta é a primeira proposta do classificador k-NN sobre dados cifrados que pode ser executado de uma forma não-interativa.

Como discutido na seção 3.2, o esquema criptográfico homomórfico empregado utiliza como espaço de textos claros um conjunto de polinômios, mas pode ser facilmente substituído por esquemas mais gerais, bastando apenas codificar cada classe i como um vetor binário com uma única entrada diferente de zero na posição i e adaptar os algoritmos para trabalhar com vetores ao invés de polinômios.

Além disso, se um cliente e um provedor de serviços em nuvem já utilizam qualquer protocolo para encontrar vizinhos mais próximos (por exemplo, usando outras primitivas criptográficas em vez de OPE ou executando algum algoritmo interativo), podem então usar um esquema homomórfico e as técnicas apresentadas aqui para obter uma classe a partir das outras classes.

Como trabalho futuro, possíveis melhorias para o k-NN aqui apresentado podem envolver técnicas de ofuscação de dados e de perturbação para obter propriedades de segurança mais fortes contra ataques de inferência, preservando ao mesmo tempo precisão e eficiência. Pretende-se também estender a versão apresentada aqui para uma versão do k-NN ponderada pelas distâncias.

Referências

Alpaydin, E. (2004). *Introduction to Machine Learning*. The MIT Press.

Altman, N. S. (1992). An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 46(3):175–185.

Boldyreva, A., Chenette, N., Lee, Y., and O'Neill, A. (2009). Order-Preserving Symmetric Encryption. In *Proceedings of the 28th Annual International Conference on Advances in Cryptology: The Theory and Applications of Cryptographic Techniques*, EUROCRYPT '09, pages 224–241, Berlin, Heidelberg. Springer-Verlag.

Bos, J. W., Lauter, K., Loftus, J., and Naehrig, M. (2013). *Improved Security for a Ring-Based Fully Homomorphic Encryption Scheme*, pages 45–64. Springer Berlin Heidelberg, Berlin, Heidelberg.

Bost, R., Popa, R. A., Tu, S., and Goldwasser, S. (2014). Machine learning classification over encrypted data. Technical report, Cryptology ePrint Archive, Report 2014/331, 2014. http://eprint. iacr. org.

Choi, S., Ghinita, G., Lim, H.-S., and Bertino, E. (2014). Secure knn query processing in untrusted cloud environments. *Knowledge and Data Engineering, IEEE Transactions on*, 26(11):2818–2831.

Dowlin, N., Gilad-Bachrach, R., Laine, K., Lauter, K., Naehrig, M., and Wernsing, J. (2015). Manual for using homomorphic encryption for bioinformatics.

Dowlin, N., Gilad-Bachrach, R., Laine, K., Lauter, K., Naehrig, M., and Wernsing, J. (2016). Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. Technical Report MSR-TR-2016-3.

Elmehdwi, Y., Samanthula, B. K., and Jiang, W. (2014). Secure k-nearest neighbor query over encrypted data in outsourced environments. In *Data Engineering (ICDE), 2014 IEEE 30th International Conference on*, pages 664–675. IEEE.

Graepel, T., Lauter, K., and Naehrig, M. (2013). Ml confidential: Machine learning on encrypted data. In *Information Security and Cryptology (ICISC)*, pages 1–21. Springer.

Hirt, M. and Sako, K. (2000). Efficient receipt-free voting based on homomorphic encryption. In *Advances in Cryptology – EUROCRYPT*, pages 539–556. Springer.

Jha, S., Kruger, L., and McDaniel, P. (2005). Privacy preserving clustering. In *Computer Security (ESORICS)*, pages 397–417. Springer.

Lepoint, T. and Naehrig, M. (2014). A comparison of the homomorphic encryption schemes FV and YASHE. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 8469 LNCS:318–335.

Lindell, Y. and Pinkas, B. (2009). Secure multiparty computation for privacy-preserving data mining. *Journal of Privacy and Confidentiality*, 1(1):5.

Miller, C. C. (2014). Revelations of N.S.A. spying cost U.S. tech companies. *The New York Times*. http://www.nytimes.com/2014/03/22/business/fallout-from-snowden-hurting-bottom-line-of-tech-companies.html (Acessed February 04, 2016).

Naehrig, M., Lauter, K., and Vaikuntanathan, V. (2011). Can homomorphic encryption be practical? In *Proceedings of the 3rd ACM workshop on Cloud Computing Security*, pages 113–124. ACM.

Naveed, M., Kamara, S., and Wright, C. V. (2015). Inference Attacks on Property-Preserving Encrypted Databases. *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security - CCS '15*, pages 644–655.

Rivest, R. L., Adleman, L., and Dertouzos, M. L. (1978). On data banks and privacy homomorphisms. *Foundations of secure computation*, 4(11):169–180.

Wong, W. K., Cheung, D. W.-l., Kao, B., and Mamoulis, N. (2009). Secure kNN computation on encrypted databases. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*, pages 139–152. ACM.

Xiong, L., Chitti, S., and Liu, L. (2006). K nearest neighbor classification across multiple private databases. In *Proceedings of the 15th ACM International Conference on Information and Knowledge Management*, pages 840–841.

Xiong, L., Chitti, S., and Liu, L. (2007). Mining multiple private databases using a knn classifier. In *Proceedings of the 2007 ACM Symposium on Applied Computing*, SAC '07, pages 435–440, New York, NY, USA. ACM.

Zhan, J., Chang, L., and Matwin, S. (2005). Privacy preserving k-nearest neighbor classification. *International Journal of Network Security*.

Zhu, Y., Xu, R., and Takagi, T. (2013). Secure k-nn computation on encrypted cloud data without sharing key with query users. In *Proceedings of the 2013 International Workshop on Security in Cloud Computing*, Cloud Computing '13, pages 55–60, New York, NY, USA. ACM.

