



(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2025-0031333
(43) 공개일자 2025년03월07일

(51) 국제특허분류(Int. Cl.)
H04L 9/00 (2022.01) G06F 17/16 (2006.01)
(52) CPC특허분류
H04L 9/008 (2013.01)
G06F 17/16 (2013.01)
(21) 출원번호 10-2023-0112637
(22) 출원일자 2023년08월28일
심사청구일자 2023년08월28일

(71) 출원인
고려대학교 산학협력단
서울특별시 성북구 안암로 145, 고려대학교 (안암
동5가)
(72) 발명자
안태민
서울특별시 성북구 고려대로12길 66, 102호
이강훈
서울특별시 동작구 동작대로39길 22, 115-301
(뒷면에 계속)
(74) 대리인
김등용

전체 청구항 수 : 총 4 항

(54) 발명의 명칭 역행렬 기법을 이용하여 암호화된 데이터에 대한 역행렬을 생성하는 장치 및 방법

(57) 요약

역행렬 기법을 이용하여 암호화된 데이터에 대한 역행렬을 생성하는 장치 및 방법이 개시된다. 상기 역행렬 생성 방법은 적어도 프로세서(processor)를 포함하는 컴퓨팅 장치에 의해 수행되고, 행렬(A)을 수신하는 단계, 제1 알고리즘을 수행하여 상기 행렬(A)의 역행렬(B_r)을 생성하는 단계, 및 상기 역행렬(B_r)을 출력하는 단계를 포함한다.

대표도 - 도1

Algorithm 1 Proposed Inverse Matrix Method

```

1: Input:  $n \times n$  invertible matrix  $A$ , iteration number  $r$ 
2: Output: approximate inverse matrix  $B_r$ 
3:  $\lambda_1 \leftarrow$  a dominant eigenvalue or trace of  $A^T A$ 
4:  $B_0 \leftarrow \frac{1}{\lambda_1} I_{n \times n}$ 
5:  $\bar{A}_0 \leftarrow I_{n \times n} - \frac{1}{\lambda_1} A^T A$ 
6: for  $i = 1$  to  $r$  do
7:    $B_i \leftarrow B_{i-1} (I_{n \times n} + \bar{A}_{i-1})$ 
8:    $\bar{A}_i \leftarrow \bar{A}_{i-1}^2$ 
9: end for
10:  $B_r \leftarrow B_r A^T$ 
  
```

(72) 발명자

유준수

서울특별시 송파구 백제고분로36나길 28-1, 3층

윤지원

서울특별시 용산구 한강대로43길 8, 102동 1401호

이 발명을 지원한 국가연구개발사업

과제고유번호 1711193441

과제번호 2021-0-00558-003

부처명 과학기술정보통신부

과제관리(전문)기관명 정보통신기획평가원

연구사업명 정보보호핵심원천기술개발(R&D, 정보화)

연구과제명 동형암호 기술을 활용한 국가통계 분석 시스템 개발

기 여 율 1/1

과제수행기관명 고려대학교산학협력단

연구기간 2023.01.01 ~ 2023.12.31

명세서

청구범위

청구항 1

적어도 프로세서(processor)를 포함하는 컴퓨팅 장치에 의해 수행되는, 역행렬 생성 방법에 있어서, 행렬(A)를 수신하는 단계;

제1 알고리즘을 수행하여 상기 행렬(A)의 역행렬(B_r)을 생성하는 단계; 및

상기 역행렬(B_r)을 출력하는 단계를 포함하는 역행렬 생성 방법.

[제1 알고리즘]

Algorithm 1 Proposed Inverse Matrix Method

```

1: Input:  $n \times n$  invertible matrix  $A$ , iteration number  $r$ 
2: Output: approximate inverse matrix  $B_r$ 
3:  $\lambda_1 \leftarrow$  a dominant eigenvalue or trace of  $A^T A$ 
4:  $B_0 \leftarrow \frac{1}{\lambda_1} I_{n \times n}$ 
5:  $\bar{A}_0 \leftarrow I_{n \times n} - \frac{1}{\lambda_1} A^T A$ 
6: for  $i = 1$  to  $r$  do
7:    $B_i \leftarrow B_{i-1} (I_{n \times n} + \bar{A}_{i-1})$ 
8:    $\bar{A}_i \leftarrow \bar{A}_{i-1}^2$ 
9: end for
10:  $B_r \leftarrow B_r A^T$ 
    
```

청구항 2

제1항에 있어서,

상기 행렬(A)은 동형암호(Homomorphic Encryption, HE) 기법으로 암호화된 행렬인, 역행렬 생성 방법.

청구항 3

제2항에 있어서,

상기 제1 알고리즘에 포함되는 덧셈 연산과 곱셈 연산은 암호문 상에서 수행되는, 역행렬 생성 방법.

청구항 4

제2항에 있어서,

상기 동형암호 기법은 CKKS(Cheon-Kim-Kim-Song) 동형암호 기법인, 역행렬 생성 방법.

발명의 설명

기술 분야

[0001]

본 발명은 사용자의 데이터에 대한 역행렬 생성 방법에 관한 것으로, 특히 동형암호(Homomorphic Encryption, HE)로 암호화된 암호문에 대한 역행렬을 생성하는 장치 및 방법에 관한 것이다.

배경 기술

- [0002] 동형암호(Homomorphic Encryption, HE)는 기본적으로 평문에서 연산하는 것보다 암호문 상에서 매우 많은 연산량을 요구한다. 특히, 동형암호에서의 곱셈은 덧셈에서보다 더 복잡하고 정교하게 설계되어 있고, 덧셈에서 연산할 때보다 훨씬 더 많은 시간과 잡음 증가를 일으킨다.
- [0003] 평문에서는 역행렬을 보통 가우스 소거법(Gaussian elimination)을 이용하여 계산한다. 하지만, 일부 동형암호에서는 덧셈과 곱셈만이 지원되기 때문에, 많은 나눗셈이 필요한 가우스 소거법은 사용하기 힘들다. 따라서 동형암호에서는 역행렬을 구할 때 주로 뉴턴 방법(Newton method)을 사용한다. 뉴턴 방법은 덧셈과 곱셈으로만 연산이 가능하기 때문에 동형암호에서는 가우스 소거법보다 더 효과적이다.
- [0004] 뉴턴 방법은 근사적인 역행렬을 구하는 방법으로 연산값이 역행렬에 수렴하는 데 (더하기, 곱하기로 이루어진) 특정 연산들을 반복하여야 한다. 이때, 반복하는 횟수를 반복 횟수(iteration number)라고 했을 때, 상당한 반복 횟수가 필요하다. 따라서 동형암호에서 역행렬을 구하는 것은 많은 곱하기 연산과 많은 연산 시간을 필요로 한다.
- [0005] 일부 동형암호에서는 초기에 깊이(depth)라는 파라미터를 설정하여 암호문 당 곱하기 횟수를 지정한다. 깊이(depth)를 높게 설정할수록 암호문 사이즈가 커져서 더하기와 곱하기 연산 시간이 늘어난다.
- [0006] 본 발명에서 제안하는 새로운 역행렬 방법을 사용하면 이러한 깊이(depth)를 훨씬 적게 설정할 수 있고, 그에 따라 역행렬을 구하는 데 필요한 시간을 많이 줄일 수 있다.

선행기술문헌

특허문헌

- [0007] (특허문헌 0001) 대한민국 공개특허 제2022-0096223호 (2022.07.07. 공개)
(특허문헌 0002) 대한민국 공개특허 제2022-0120410호 (2022.08.30. 공개)
(특허문헌 0003) 대한민국 공개특허 제2021-0129573호 (2021.10.28. 공개)
(특허문헌 0004) 대한민국 공개특허 제2019-0005540호 (2019.01.16. 공개)

발명의 내용

해결하려는 과제

- [0008] 본 발명이 이루고자 하는 기술적인 과제는 역행렬 기법을 이용하여 암호화된 데이터에 대한 역행렬을 생성하는 장치 및 방법을 제공하는 것이다.

과제의 해결 수단

- [0009] 본 발명의 일 실시예에 따른 역행렬 생성 방법은 적어도 프로세서(processor)를 포함하는 컴퓨팅 장치에 의해 수행되고, 행렬(A)을 수신하는 단계, 제1 알고리즘을 수행하여 상기 행렬(A)의 역행렬(B_1)을 생성하는 단계, 및 상기 역행렬(B_1)을 출력하는 단계를 포함한다.

발명의 효과

- [0010] 본 발명의 실시예들에 의한 경우, 생성된 역행렬을 이용하여 프라이버시를 보존하면서 데이터를 분석하는 영역에서 사용 가능하다.
- [0011] 또한, 의료, 금융, 위치와 관련된 민감한 데이터를 동형암호를 이용해 암호화된 상태에서 처리하여 사용자의 프라이버시를 보존할 수 있다.
- [0012] 또한, 동형암호에서 역행렬을 구하거나, 역행렬을 포함하는 연산으로 데이터를 분석하는 영역에서 사용 가능하다.

- [0013] 또한, 동형암호에서 역행렬을 계산해야 하는 경우 기존보다 더 빠르고 효율적으로 학습할 수 있다.
- [0014] 또한, 동형암호에서 역행렬을 포함한 기술을 사용하여 데이터를 분석할 경우 기존보다 더 빠르고 효율적으로 학습할 수 있다.

도면의 간단한 설명

- [0015] 본 발명의 상세한 설명에서 인용되는 도면을 보다 충분히 이해하기 위하여 각 도면의 상세한 설명이 제공된다.
- 도 1은 본 발명의 일 실시예에 따른 역행렬 생성 방법을 나타내는 제1 알고리즘을 도시한다.
- 도 2는 종래의 역행렬 생성 방법을 나타내는 제2 알고리즘을 도시한다.

발명을 실시하기 위한 구체적인 내용

- [0016] 본 명세서에 개시되어 있는 본 발명의 개념에 따른 실시예들에 대해서 특정한 구조적 또는 기능적 설명들은 단지 본 발명의 개념에 따른 실시예들을 설명하기 위한 목적으로 예시된 것으로서, 본 발명의 개념에 따른 실시예들은 다양한 형태로 실시될 수 있으며 본 명세서에 설명된 실시예들에 한정되지 않는다.
- [0017] 본 발명의 개념에 따른 실시예들은 다양한 변경들을 가할 수 있고 여러 가지 형태들을 가질 수 있으므로 실시예들을 도면에 예시하고 본 명세서에서 상세하게 설명하고자 한다. 그러나, 이는 본 발명의 개념에 따른 실시예들을 특정한 개시 형태들에 대해 한정하려는 것이 아니며, 본 발명의 사상 및 기술 범위에 포함되는 모든 변경, 균등물, 또는 대체물을 포함한다.
- [0018] 제1 또는 제2 등의 용어는 다양한 구성 요소들을 설명하는데 사용될 수 있지만, 상기 구성 요소들은 상기 용어들에 의해 한정되어서는 안 된다. 상기 용어들은 하나의 구성 요소를 다른 구성 요소로부터 구별하는 목적으로만, 예컨대 본 발명의 개념에 따른 권리 범위로부터 벗어나지 않은 채, 제1 구성 요소는 제2 구성 요소로 명명될 수 있고 유사하게 제2 구성 요소는 제1 구성 요소로도 명명될 수 있다.
- [0019] 어떤 구성 요소가 다른 구성 요소에 "연결되어" 있다거나 "접속되어" 있다고 언급된 때에는, 그 다른 구성 요소에 직접적으로 연결되어 있거나 또는 접속되어 있을 수도 있지만, 중간에 다른 구성 요소가 존재할 수도 있다고 이해되어야 할 것이다. 반면에, 어떤 구성 요소가 다른 구성 요소에 "직접 연결되어" 있다거나 "직접 접속되어" 있다고 언급된 때에는 중간에 다른 구성 요소가 존재하지 않는 것으로 이해되어야 할 것이다. 구성 요소들 간의 관계를 설명하는 다른 표현들, 즉 "~사이에"와 "바로 ~사이에" 또는 "~에 이웃하는"과 "~에 직접 이웃하는" 등도 마찬가지로 해석되어야 한다.
- [0020] 본 명세서에서 사용한 용어는 단지 특정한 실시예를 설명하기 위해 사용된 것으로서, 본 발명을 한정하려는 의도가 아니다. 단수의 표현은 문맥상 명백하게 다르게 뜻하지 않는 한, 복수의 표현을 포함한다. 본 명세서에서, "포함하다" 또는 "가지다" 등의 용어는 본 명세서에 기재된 특징, 숫자, 단계, 동작, 구성 요소, 부분품 또는 이들을 조합한 것이 존재함을 지정하려는 것이지, 하나 또는 그 이상의 다른 특징들이나 숫자, 단계, 동작, 구성 요소, 부분품 또는 이들을 조합한 것들의 존재 또는 부가 가능성을 미리 배제하지 않는 것으로 이해되어야 한다.
- [0021] 다르게 정의되지 않는 한, 기술적이거나 과학적인 용어를 포함해서 여기서 사용되는 모든 용어들은 본 발명이 속하는 기술 분야에서 통상의 지식을 가진 자에 의해 일반적으로 이해되는 것과 동일한 의미를 가진다. 일반적으로 사용되는 사전에 정의되어 있는 것과 같은 용어들은 관련 기술의 문맥상 가지는 의미와 일치하는 의미를 갖는 것으로 해석되어야 하며, 본 명세서에서 명백하게 정의하지 않는 한, 이상적이거나 과도하게 형식적인 의미로 해석되지 않는다.
- [0022] 이하, 본 명세서에 첨부된 도면들을 참조하여 본 발명의 실시예들을 상세히 설명한다. 그러나, 특허출원의 범위가 이러한 실시예들에 의해 제한되거나 한정되는 것은 아니다. 각 도면에 제시된 동일한 참조 부호는 동일한 부재를 나타낸다.
- [0023] 우선, 동형암호(Homomorphic Encryption, HE)에 대해 설명한다.
- [0024] (1) 동형암호의 특성
- [0025] 동형암호는 LWE(Learning With Error) 문제를 기반으로 하는 양자 내성 암호이다.
- [0026] 동형암호가 기존 암호와 다른 점은 암호문 간 연산(Eval) 스텝을 추가로 갖는다는 것이다. 기존 암호는 보안 레

벨에 따른 파라미터/키 생성, 암호화, 복호화를 기반으로 전송 상태에서의 보안을 고려하고, 종단에서의 데이터 유출에 대한 모델을 고려하지 않았다. 동형암호는 연산(Eval) 단계를 통해 암호문 간 연산 기능을 제공하는데, 대표적으로 암호문 간 덧셈과 곱셈을 제공한다. 따라서, 연산(Eval) 단계를 이용하면, 암호문 간 연산이 가능하여 클라우드에서 암호문을 복호화하지 않고 암호문 간 연산을 통해 사용자가 원하는 연산을 동형적으로 만들 수 있다.

[0027] 동형암호는 매 암호문 간 연산 시에 잡음(noise)이 추가된다. 잡음을 통해 암호문의 보안(security)이 유지되지만, 일정 수준의 잡음 레벨을 넘게 되면, 복호화 시 정확한 결과를 얻지 못하는 문제가 있다(correctness). 따라서, 암호문 내에 잡음이 증가하는 정도를 잘 추적하고, 이에 따른 파라미터를 잘 설정하는 것이 중요하다.

[0028] 동형암호는 일반적으로 다음의 단계와 기능을 갖는다.

[0029] - ParamGen(λ): 보안 레벨(λ)에 따른 각종 파라미터(Params)를 생성한다. 파라미터의 예시로는 잡음의 분산(α), LWE 샘플 차원 등이 있다.

[0030] - KeyGen(Params): 파라미터가 정해지면 각종 키를 생성한다. 여기서 생성하는 키는 개인키 sk, 연산키 evk가 있다. 개인키는 암호화와 복호화할 때 사용되고, 연산키는 암호문 간 연산 시에 사용된다.

[0031] - Encrypt(pk, μ): 메시지 μ 를 개인키 sk로 암호화하여 LWE 암호문 c를 생성한다.

[0032] - Decrypt(sk, c): 암호문 c를 개인키 sk로 복호화하여 메시지 u를 얻는다.

[0033] - Eval(evk, c_1 , c_2): 암호문 c_1 과 c_2 를 연산키 evk를 이용해 연산한다. 일반적으로 연산(Eval)은 암호문 간 덧셈과 곱셈이 있다.

[0034] (2) 유한(Leveled) 동형암호 vs 완전(Fully) 동형암호

[0035] 동형암호에서 발생하는 잡음 문제를 해결하기 위해, 동형암호는 잡음을 제거하는 부트스트랩(bootstrapping) 기능을 제공한다. 부트스트랩을 통해 잡음을 줄여 추가적인 암호문 간 연산을 수행할 수 있는 장점이 있다. 하지만, 부트스트랩 과정은 동형암호에서 상당히 무거운 연산이므로, 부트스트랩 기능은 웬만하면 사용하지 않는다.

[0036] 유한 동형암호는 부트스트랩 과정을 생략하고 정해진 회로 사이즈에 대해서 파라미터를 잘 설정해서 회로를 설계하는 방법이다. 일반적으로 완전 동형암호보다 회로 연산 시간이 빠르다.

[0037] 완전 동형암호는 고정된 파라미터를 기반으로 하므로 파라미터 설정에 고민할 필요가 없다. 일반적으로 회로의 깊이(depth)가 깊은 것들에 적용된다. 깊이(depth)는 암호문 당 곱하기 횟수를 뜻한다.

[0038] (3) 대표적인 동형암호 기법

[0039] 동형암호 스키마는 여러 가지 존재한다. 대표적으로 CKKS(Cheon-Kim-Kim-Song) 그리고 TFHE(Fast Fully Homomorphic Encryption over the Torus) 기법으로 나눌 수 있다. CKKS 기법은 실수 영역에서 효과적으로 연산을 가능하게 하고, 근삿값을 결과로 제공한다. 더하기와 곱하기 연산을 제공한다. TFHE 기법은 동형암호 기본 게이트(NAND, OR, AND)를 설계하여, 임의의 연산을 동형암호 게이트들로 설계할 수 있도록 제공한다. 본 기술은 대표적인 완전 동형암호인 CKKS 기법에 초점을 맞추었다. CKKS는 깊이(depth) 파라미터가 존재하고 깊이(depth)를 최대한 줄이는 것이 중요하다.

[0040] 이하에서는, 기존의 역행렬 방법(Newton method)과 본 발명에서 제안하는 새로운 역행렬 방법을 설명한다. 본 발명에서 제안하는 역행렬 생성 방법은 적어도 프로세서(processor) 및/또는 메모리(memory)를 포함하는 컴퓨팅 장치에 의해 수행될 수 있다. 따라서, 역행렬 생성 방법을 이루는 단계들 중 적어도 일부는 컴퓨팅 장치에 포함되는 프로세서의 동작으로 이해될 수 있고, 컴퓨팅 장치는 역행렬 생성 장치로 명명될 수도 있다. 컴퓨팅 장치는 PC(Personal Computer), 서버(server), 태블릿 PC, 랩탑 컴퓨터 등을 포함할 수 있다.

[0041] 가역 정사각 행렬 A에 대한 역행렬을 뉴턴 방법으로 구해보자. α 를 AA^T 의 절댓값이 제일 큰 고유값의 역수라고 하자. 뉴턴 방법은 행렬 수열 $\{X_k\}_{k \geq 0}$ 에 대해 다음과 같은 점화식을 X_k 가 A^{-1} 로 수렴할 때까지 계산한다.

[0042]
$$X_0 = \alpha A^T \text{ and } X_{k+1} = X_k (2I - AX_k)$$

[0043] 이번에는 가역 정사각 행렬 A에 대한 역행렬을 새로운 역행렬 방법으로 구해보자. β 를 $A^T A$ 의 절댓값이 제일 큰

고유값의 역수라고 하자. 새로운 역행렬 방법은 행렬 수열 $\{Y_k\}_{k \geq 0}$ 에 대해 다음과 같은 점화식을 Y_k 가 $(A^T A)^{-1}$ 로 수렴할 때까지 계산한다.

$$Y_0 = \beta I, A' = I - \beta A^T A \text{ and } Y_{k+1} = Y_k (I + A'^{2^k})$$

그 다음, $Y_k A^T$ 를 계산하면 A^{-1} 를 얻을 수 있다.

뉴턴 방법의 수렴 증명은 잘 알려져 있으므로 생략하고 새로운 역행렬 방법을 사용했을 때 역행렬로 수렴하는 것을 증명한다. 위에 점화식을 이용하여 다음과 같은 식을 얻을 수 있다.

$$Y_k = \beta(I + A')(I + A'^2) \cdots (I + A'^{2^{k-1}}) = (A^T A)^{-1} (I - A'^{2^k})$$

여기서 A' 의 스펙트럼 반경(spectral radius)이 1보다 작으면 증명이 완성된다. 스펙트럼 반경이 1보다 작으면 $\lim_{k \rightarrow \infty} A'^k = 0$ 이기 때문이다. 스펙트럼 반경은 고유값의 절댓값의 상한을 뜻한다. 따라서, 모든 고유값의 절댓값이 1보다 작으면 된다. $A^T A$ 는 양의 정부호 행렬이기 때문에 모든 고유값이 양수이다. A' 의 고유값은 $1 - \beta \times (A^T A \text{의 고유값})$ 이고 β 는 $A^T A$ 의 절댓값이 제일 큰 고유값의 역수이기 때문에 A' 의 고유값들의 절댓값은 항상 1보다 작다. 따라서 Y_k 는 $(A^T A)^{-1}$ 로 수렴한다. 즉, $Y_k A^T$ 는 A^{-1} 로 수렴한다.

이하에서는, 본 발명에서 제안하는 새로운 역행렬 방법이 더 효율적인 이유를 설명한다.

위에 역행렬 방법들에서 절댓값이 제일 큰 고유값을 사용한다고 했는데, 사실 동형암호에서는 고유값을 구하기 어렵기 때문에, 절댓값이 제일 큰 고유값 대신 행렬의 트레이스(trace)를 사용한다. 트레이스는 행렬의 대각성분들의 합을 뜻하고, 행렬이 정사각 행렬일 경우 모든 고유값들의 합과 같다. 위에 역행렬 방법들은 양의 정부호 행렬들의 절댓값이 제일 큰 고유값들을 사용하는데, 이 경우 트레이스는 절댓값이 제일 큰 고유값보다 항상 크다. 따라서 트레이스를 대신 사용해도 수렴성은 유지된다.

다음은 새로운 역행렬 방법이 뉴턴 방법과 수렴 속도가 동일하다는 증명이다. 이때, 둘 다 트레이스를 대신 사용하는 것으로 가정한다. 물론, 둘 다 절댓값이 제일 큰 고유값을 사용한다고 해도 같게 수렴한다. 이에 대한 증명은 원리가 동일하기에 생략한다.

상술한 뉴턴 방법에서의 X_k 는 다음과 같이 표현할 수 있다.

$$X_k = A^{-1} (I - (I - \alpha A A^T)^{2^k})$$

여기서, α 는 $A A^T$ 의 트레이스의 역수이다.

새로운 역행렬 방법에서의 Y_k 는 다음과 같이 표현할 수 있다.

$$Y_k = (A^T A)^{-1} (I - (I - \beta A^T A)^{2^k})$$

여기서 β 는 $A^T A$ 의 트레이스의 역수이다.

둘 다 역행렬로 수렴하려면 각각 $\|A^{-1} - X_k\|$, $\|(A^T A)^{-1} - Y_k\|$ 가 0이 되어야 한다.

$A^T A$ 와 $A A^T$ 의 고유값은 동일하므로 $A^T A$ 와 $A A^T$ 의 고유값을 크기 순서대로 $\lambda_1, \dots, \lambda_n (> 0)$ 라 하자. 따라서, 다음과 같이 계산할 수 있다.

$$\|A^{-1} - X_k\| \leq \|A^{-1}\| \|I - \alpha A A^T\|^{2^k} = \|A^{-1}\| \left(1 - \frac{\lambda_n}{\lambda_1 + \dots + \lambda_n}\right)^{2^k} = \|A^{-1}\| (1 - \delta_1)^{2^k}$$

$$\begin{aligned} \|(A^T A)^{-1} - Y_k\| &\leq \|(A^T A)^{-1}\| \|I - \beta A^T A\|^{2^k} = \|(A^T A)^{-1}\| \left(1 - \frac{\lambda_n}{\lambda_1 + \dots + \lambda_n}\right)^{2^k} \\ &= \|(A^T A)^{-1}\| (1 - \delta_2)^{2^k} \end{aligned}$$

- [0062] 여기서, $\|A^{-1}\|$, $\|A^T A^{-1}\|$ 에서의 오차가 조금 있을 수 있지만, $0 < \delta_1 = \delta_2 < 1$ 이기 때문에, 수렴 속도가 동일하다고 할 수 있다. 즉, 새로운 역행렬 방법과 뉴턴 방법은 수렴 속도가 동일하다.
- [0063] 본 발명에서 제안하는 역행렬 생성 방법은 도 1에 도시된 제1 알고리즘(Algorithm 1)으로 나타낼 수 있고, 기존의 뉴턴 방법은 도 2에 도시된 제2 알고리즘(Algorithm 2)으로 나타낼 수 있다.
- [0064] 각각 알고리즘에서 수렴에 필요한 반복 횟수 r 에 대해서 반복 연산해야 하는 식을 살펴보자. 각각 식에서 곱하기 2번, 더하기(빼기) 1번이 필요한 것을 알 수 있다. 즉, 두 방법의 연산 횟수는 동일하다.
- [0065] 하지만, 두 방법을 동형암호(예컨대, CKKS 동형암호)로 구현한다고 했을 때, 깊이(depth)에서 차이가 있다. 한 번의 행렬 곱셈을 하는 데에 깊이(depth)가 1만큼 소모된다고 가정하자. 본 발명에서 제안하는 방법 식에서 B_0 와 $\overline{A_0}$ 가 깊이(depth) 0인 상태로 시작한다고 하자.
- [0066] $B_1 = B_0 + B_0 \overline{A_0}$ 이므로 B_1 의 깊이(depth)는 1이 되고, $\overline{A_1} = \overline{A_0} \times \overline{A_0}$ 이므로 $\overline{A_1}$ 의 깊이(depth)는 1이 된다. 따라서, 한 번의 반복 횟수마다 곱하기는 2번 사용하지만 깊이(depth)는 1만 소모가 된다.
- [0067] 이번에는 기존의 뉴턴 방법 식에서 B_0 와 A 가 깊이(depth) 0인 상태로 시작한다고 하자. $B_1 = 2B_0 - B_0 \times A \times B_0$ 이므로 B_1 의 깊이(depth)는 2가 된다. 따라서, 뉴턴 방법에서는 한 번의 반복 횟수마다 곱하기가 2번 사용되고 깊이(depth)도 2가 소모된다.
- [0068] 결국, 본 발명에서 제안하는 역행렬 생성 방법은 기존의 뉴턴 방법과 반복 횟수와 연산 횟수는 거의 동일하지만 깊이(depth)를 거의 2배 정도 적게 설정할 수 있으므로 더 빠르고 효율적으로 역행렬을 구할 수 있다.
- [0069] 상술한 새로운 역행렬 기법은 기존의 뉴턴 방법보다 더 빠르고 효율적이다. 따라서, CKKS와 같이 더하기와 곱하기 연산을 지원하고 깊이(depth) 개념이 존재하는 동형암호에서 역행렬을 계산할 때, 뉴턴 방법 대신 새로운 역행렬 방법을 사용하면 더 빠르고 효율적으로 계산할 수 있다. 예를 들어, 역행렬이 필요한 기계학습(linear regression, linear discriminant analysis 등)을 동형암호로 구현할 때 유용하게 이용될 수 있다.
- [0070] 상술한 역행렬 생성 방법이 컴퓨팅 장치에 의해 수행되는 경우, 컴퓨팅 장치는 주어진 행렬을 대상으로 제1 알고리즘을 수행함으로써 주어진 행렬에 대한 역행렬을 생성할 수 있다.
- [0071] 보다 구체적으로, 역행렬 생성 방법은 소정의 데이터, 예컨대, $n \times n$ (n 은 임의의 자연수)의 대상 행렬(A), 및 반복 횟수(r , r 은 임의의 자연수)를 수신하고, 제1 알고리즘을 수행함으로써 행렬(A)에 대한 역행렬(B_r)을 생성할 수 있다. 대상 행렬(A)는 컴퓨팅 장치에 미리 저장되어 있을 수 있다. 다른 예로, 대상 행렬(A)는 소정의 외부 장치(예컨대, 사용자 단말)로부터, 유무선 통신망을 통하여, 수신되는 것도 가능하다. 구체적인 예로, 컴퓨팅 장치는 외부 장치로부터 암호문에 대한 연산을 위탁받을 수 있고, 위탁 연산 과정에서 임의의 행렬에 대한 역행렬을 생성하는 과정이 필요할 수 있다. 본 발명에서 제안하는 역행렬 생성 방법은, 요구되는 역행렬을 생성함으로써, 위탁 연산을 수행할 수 있다. 역행렬이 생성된 후, 컴퓨팅 장치는 생성된 역행렬을 외부 장치로 송신하거나, 위탁 연산의 결과를 외부 장치로 송신할 수 있다.
- [0072] 또한, 대상 행렬은 동형암호 등의 기법을 통해 암호화된 행렬을 의미할 수 있다. 암호화된 행렬이라 함은 행렬의 각 원소가 평문이 아닌 암호문인 것을 의미한다. 따라서, 컴퓨팅 장치는 암호문 간의 연산(덧셈 및/또는 곱셈)을 지원하는 동형암호 기법을 이용하여 제1 알고리즘에 포함되는 연산들을 수행함으로써 대상 행렬에 대한 역행렬을 생성할 수 있다. 이때, 생성되는 역행렬 또한 암호화된 상태일 수 있다.
- [0073] 이상에서 설명된 장치는 하드웨어 구성 요소, 소프트웨어 구성 요소, 및/또는 하드웨어 구성 요소 및 소프트웨어 구성 요소의 집합으로 구현될 수 있다. 예를 들어, 실시예들에서 설명된 장치 및 구성 요소는, 예를 들어, 프로세서, 컨트롤러, ALU(Arithmetic Logic Unit), 디지털 신호 프로세서(Digital Signal Processor), 마이크로컴퓨터, FPA(Field Programmable array), PLU(Programmable Logic Unit), 마이크로프로세서, 또는 명령(instruction)을 실행하고 응답할 수 있는 다른 어떠한 장치와 같이, 하나 이상의 범용 컴퓨터 또는 특수 목적 컴퓨터를 이용하여 구현될 수 있다. 처리 장치는 운영 체제(Operation System, OS) 및 상기 운영 체제 상에서 수행되는 하나 이상의 소프트웨어 애플리케이션을 수행할 수 있다. 또한, 처리 장치는 소프트웨어의 실행에 응답하여, 데이터를 접근, 저장, 조작, 처리 및 생성할 수도 있다. 이해의 편의를 위하여, 처리 장치는 하나가 사용되는 것으로 설명된 경우도 있지만, 해당 기술 분야에서 통상의 지식을 가진 자는, 처리 장치가 복수 개의 처

리 요소(Processing Element) 및/또는 복수 유형의 처리 요소를 포함할 수 있음을 알 수 있다. 예를 들어, 처리 장치는 복수 개의 프로세서 또는 하나의 프로세서 및 하나의 컨트롤러를 포함할 수 있다. 또한, 병렬 프로세서(Parallel Processor)와 같은, 다른 처리 구성(Processing Configuration)도 가능하다.

[0074] 소프트웨어는 컴퓨터 프로그램(Computer Program), 코드(Code), 명령(Instruction), 또는 이들 중 하나 이상의 조합을 포함할 수 있으며, 원하는 대로 동작하도록 처리 장치를 구성하거나 독립적으로 또는 결합적으로(Collectively) 처리 장치를 명령할 수 있다. 소프트웨어 및/또는 데이터는, 처리 장치에 의하여 해석되거나 처리 장치에 명령 또는 데이터를 제공하기 위하여, 어떤 유형의 기계, 구성 요소(Component), 물리적 장치, 가상 장치(Virtual Equipment), 컴퓨터 저장 매체 또는 장치, 또는 전송되는 신호 파(Signal Wave)에 영구적으로, 또는 일시적으로 구체화(Embody)될 수 있다. 소프트웨어는 네트워크로 연결된 컴퓨터 시스템 상에 분산되어서, 분산된 방법으로 저장되거나 실행될 수도 있다. 소프트웨어 및 데이터는 하나 이상의 컴퓨터 판독 가능 기록 매체에 저장될 수 있다.

[0075] 실시예에 따른 방법은 다양한 컴퓨터 수단을 통하여 수행될 수 있는 프로그램 명령 형태로 구현되어 컴퓨터 판독 가능 매체에 기록될 수 있다. 상기 컴퓨터 판독 가능 매체는 프로그램 명령, 데이터 파일, 데이터 구조 등을 단독으로 또는 조합하여 포함할 수 있다. 상기 매체에 기록되는 프로그램 명령은 실시예를 위하여 특별히 설계되고 구성된 것들이거나 컴퓨터 소프트웨어 당업자에게 공지되어 사용 가능한 것일 수도 있다. 컴퓨터 판독 가능 기록 매체의 예에는 하드 디스크, 플로피 디스크 및 자기 테이프와 같은 자기 매체(Magnetic Media), CD-ROM, DVD와 같은 광기록 매체(Optical Media), 플롭티컬 디스크(Floptical Disk)와 같은 자기-광 매체(Magneto-optical Media), 롬(ROM), 램(RAM), 플래시 메모리 등과 같은 프로그램 명령을 저장하고 수행하도록 특별히 구성된 하드웨어 장치가 포함된다. 프로그램 명령의 예에는 컴파일러에 의해 만들어지는 것과 같은 기계어 코드뿐만 아니라 인터프리터 등을 사용해서 컴퓨터에 의해서 실행될 수 있는 고급 언어 코드를 포함한다. 상기된 하드웨어 장치는 실시예의 동작을 수행하기 위해 하나 이상의 소프트웨어 모듈로서 작동하도록 구성될 수 있으며, 그 역도 마찬가지이다.

[0076] 본 발명은 도면에 도시된 실시예를 참고로 설명되었으나 이는 예시적인 것에 불과하며, 본 기술 분야의 통상의 지식을 가진 자라면 이로부터 다양한 변형 및 균등한 타 실시예가 가능하다는 점을 이해할 것이다. 예를 들어, 설명된 기술들이 설명된 방법과 다른 순서로 수행되거나, 및/또는 설명된 시스템, 구조, 장치, 회로 등의 구성 요소들이 설명된 방법과 다른 형태로 결합 또는 조합되거나, 다른 구성 요소 또는 균등물에 의하여 대치되거나 치환되더라도 적절한 결과가 달성될 수 있다. 따라서, 본 발명의 진정한 기술적 보호 범위는 첨부된 등록청구범위의 기술적 사상에 의해 정해져야 할 것이다.

도면

도면1

Algorithm 1 Proposed Inverse Matrix Method

```

1: Input:  $n \times n$  invertible matrix  $A$ , iteration number  $r$ 
2: Output: approximate inverse matrix  $B_r$ 
3:  $\lambda_1 \leftarrow$  a dominant eigenvalue or trace of  $A^T A$ 
4:  $B_0 \leftarrow \frac{1}{\lambda_1} I_{n \times n}$ 
5:  $\bar{A}_0 \leftarrow I_{n \times n} - \frac{1}{\lambda_1} A^T A$ 
6: for  $i = 1$  to  $r$  do
7:    $B_i \leftarrow B_{i-1} (I_{n \times n} + \bar{A}_{i-1})$ 
8:    $\bar{A}_i \leftarrow \bar{A}_{i-1}^2$ 
9: end for
10:  $B_r \leftarrow B_r A^T$ 

```

도면2

Algorithm 2 Newton's Matrix Inverse Method

- 1: **Input:** $n \times n$ invertible matrix $\mathbf{A}$, iteration number r
 - 2: **Output:** approximate inverse matrix $\mathbf{B}_r$
 - 3: $\lambda_1 \leftarrow$ a dominant eigenvalue or trace of $\mathbf{A}\mathbf{A}^T$
 - 4: $\mathbf{B}_0 \leftarrow \frac{1}{\lambda_1} \mathbf{A}^T$
 - 5: **for** $i = 1$ **to** r **do**
 - 6: $\mathbf{B}_i \leftarrow \mathbf{B}_{i-1}(2\mathbf{I}_{n \times n} - \mathbf{A}\mathbf{B}_{i-1})$
 - 7: **end for**
-