|  |  |  |
| --- | --- | --- |
| 3주차 선형시스템 보고서 | | |
| 제출일 : 2024년 03월 25일 | | 작성자 : 이준용 |
| 구분 | 내용 | |
| 학습 범위와 내용 | 3주차 온라인 강의 내용 | |
| 정리 내용 | **수식을 넣기가 힘들어 수식관련 개념은 개인적으로 공부하였고 보고서는 개념위주로 정리했습니다.**  **2.1 SOLUTION FOR A SYSTEM OF LINEAR EQUATIONS**  **2.1.1 The Nonsingular Case (M = N)**  "The Nonsingular Case (M = N)"은 선형 방정식 시스템의 해가 비특이한(non-singular) 경우를 나타냅니다. 여기서 M은 방정식의 수를 나타내고 N은 미지수의 수를 나타냅니다.  선형 방정식 시스템에서 비특이한 경우란, 해가 유일하게 존재하고 미지수의 수와 방정식의 수가 동일한 경우를 의미합니다. 이 경우, 시스템은 하나의 해를 가지며, 행렬 형태로 나타내면 행렬의 랭크(rank)가 최대가 됩니다.  이러한 비특이한 경우에 대한 솔루션은 다양한 방법으로 얻을 수 있습니다. 가우스 소거법(Gaussian elimination), 역행렬을 이용한 방법, 또는 행렬의 랭크와 영공간(null space)을 이용한 방법 등이 있습니다. 비특이한 경우에는 일반적으로 해를 구하는 과정이 수월합니다.  **2.1.2 The Underdetermined Case (M < N): Minimum-Norm Solution**  선형 방정식 시스템이 미지수의 수보다 방정식의 수가 적은 경우를 다룹니다. 이 경우에는 해가 무수히 많을 수 있습니다. 일반적으로, 이러한 경우에는 무한한 수의 해를 가지며, 최소노름(minimum-norm) 해가 특히 중요한 역할을 합니다. 최소노름 해는 모든 해 중에서 노름(norm)이 가장 작은 해를 나타냅니다. 이는 일종의 최적해로 볼 수 있습니다.  **2.1.3 The Overdetermined Case (M > N): LSE Solution**  선형 방정식 시스템이 미지수의 수보다 방정식의 수가 많은 경우를 다룹니다. 이 경우에는 일반적으로 해가 존재하지 않거나, 해가 무수히 많을 수 있습니다. 따라서, 모든 방정식을 만족하는 해를 찾는 것이 불가능할 수 있습니다. 이런 경우에는 최소자승해(Least Squares Estimation, LSE)를 찾는 것이 일반적으로 사용됩니다. 최소자승해는 주어진 방정식에 가장 적합한 해를 나타냅니다. 이것은 방정식을 가장 잘 근사하는 해를 찾는 것으로, 오차를 최소화하는 해를 의미합니다.  **2.1.4 RLSE (Recursive Least-Squares Estimation)**  RLSE(Recursive Least-Squares Estimation)는 실시간으로 발생하는 데이터를 기반으로 모델을 업데이트하는 데 사용되는 재귀적인 최소자승 추정 기법입니다. 이 기법은 시간에 따라 데이터가 동적으로 변할 때 유용하게 사용됩니다.  RLSE는 일련의 관측치가 순차적으로 도착하는 상황에서 모델 파라미터를 조정하는 방법으로, 예측 모델의 파라미터를 업데이트하기 위해 최신 데이터를 이용합니다. 이를 통해 모델이 데이터에 더 잘 적응할 수 있도록 합니다.  일반적으로 RLSE는 다음과 같은 단계를 따릅니다:  1.초기화: 모델 파라미터를 초기화합니다.  2.측정값 수신: 새로운 관측치(데이터)가 도착합니다.  3.예측: 현재 모델 파라미터를 기반으로 다음 관측치를 예측합니다.  4.측정: 실제 관측치를 측정합니다.  5.업데이트: 측정치와 예측치 간의 차이를 최소화하는 방향으로 모델 파라미터를 업데이트합니다.  6.다음 관측치 기다리기: 다음 데이터를 기다립니다.  이러한 단계는 지속적으로 반복되며, 시간이 지남에 따라 모델은 더 많은 데이터를 수용하고 예측 능력을 향상시킵니다.  **2.2 SOLVING A SYSTEM OF LINEAR EQUATIONS**  **2.2.16 Gauss Elimination**  Gaussian Elimination(가우스 소거법)은 선형 방정식 시스템을 해결하는 고전적인 방법 중 하나입니다. 이 방법은 행렬을 사용하여 선형 방정식을 간단한 형태로 변환하여 해를 구하는데 사용됩니다.  가우스 소거법의 주요 단계는 다음과 같습니다:  확장된 행렬 구성: 선형 방정식 시스템을 행렬 형태로 나타냅니다. 이를 위해 행렬은 계수 행렬과 상수 벡터로 분리됩니다.  전진 소거(Forward Elimination): 확장된 행렬을 삼각 행렬(upper triangular matrix)로 변환합니다. 이를 위해 각 단계에서 주 대각선 상의 원소를 피봇(pivot)으로 선택하고, 해당 열의 다른 행들을 조합하여 그 아래의 원소를 0으로 만듭니다.  후진 대입(Back Substitution): 변환된 삼각 행렬을 사용하여 변수를 하나씩 역으로 대체하여 해를 찾습니다. 이는 마지막 행부터 시작하여 유도적으로 계산됩니다.  가우스 소거법은 일반적으로 단계별로 진행되며, 해결하려는 선형 방정식의 수와 미지수의 수에 따라 복잡성이 달라집니다. 또한, 가우스 소거법은 소거 과정에서 부동소수점 연산을 사용하므로, 특히 큰 행렬이나 수치적으로 불안정한 경우 반올림 오차에 주의해야 합니다.  **2.2.2 Partial Pivoting**  Partial pivoting(부분 피벗팅)은 가우스 소거법(Gaussian Elimination)을 사용하여 선형 방정식 시스템을 해결하는 과정에서 사용되는 기법 중 하나입니다. 이 기법은 주로 행렬의 특정 요소를 선택하는 방법을 의미합니다.  부분 피벗팅의 주요 목적은 피벗 선택 시에 최대한의 정밀도를 유지하면서 피벗을 선택하는 것입니다. 일반적으로 가우스 소거법에서는 피벗을 현재 열의 가장 큰 값을 가진 행으로 선택합니다. 그러나 이러한 방식은 어떤 경우에는 문제가 될 수 있습니다.  예를 들어, 주어진 행렬이 특정한 패턴이나 특성을 가지고 있을 때, 피벗이 선택되는 과정에서 반영된 오차가 증폭될 수 있습니다. 이러한 상황에서 부분 피벗팅이 유용하게 사용됩니다.  부분 피벗팅의 작동 방식은 다음과 같습니다:  현재 열에서 가장 큰 절대값을 가진 피벗을 찾습니다.  찾은 피벗을 현재 열의 첫 번째 행과 교환합니다.  이러한 교환은 현재 열에서의 부분 피벗팅 과정을 완료합니다.  이제 가우스 소거법을 계속 진행합니다.  부분 피벗팅은 계산의 안정성을 향상시키고 반올림 오차를 줄여주므로, 수치적으로 안정적인 해를 얻는 데 도움이 됩니다. 따라서, 선형 방정식 시스템의 해를 구할 때 가우스 소거법을 적용할 때는 부분 피벗팅을 고려하는 것이 일반적으로 좋은 방법입니다.  **2.2.3 Gauss–Jordan Elimination**  Gauss-Jordan Elimination(가우스-요르단 소거법)은 가우스 소거법(Gaussian Elimination)의 확장된 형태로, 선형 방정식 시스템을 해결하는 또 다른 방법입니다. 이 방법은 행렬을 기반으로 하며, 삼각 행렬(upper triangular matrix)이 아닌 행렬을 최소 행사다리꼴(Minimal Row Echelon Form)로 변환합니다.  가우스-요르단 소거법의 주요 특징은 다음과 같습니다:  전진 소거(Forward Elimination): 가우스 소거법과 마찬가지로, 행렬을 삼각 행렬의 형태로 변환합니다. 이 단계에서 주 대각선 상의 원소를 피봇(pivot)으로 선택하고, 해당 열의 다른 행들을 조합하여 그 아래의 원소를 0으로 만듭니다.  후진 소거(Backward Elimination): 전진 소거가 완료되면, 행렬을 최소 행사다리꼴(Minimal Row Echelon Form)로 변환하기 위해 후진 소거 단계를 수행합니다. 이 단계에서는 각 피봇의 위쪽의 원소를 0으로 만들기 위해 각 행을 다시 조합합니다.  단위 행렬로 변환: 후진 소거가 완료되면, 각 피봇의 위치에 대응하는 열을 기준으로 피봇 아래의 모든 원소가 0이 되므로, 최소 행사다리꼴 형태가 됩니다. 이 단계에서 각 피봇 위치에 대응하는 행은 단위 행렬의 형태를 가지게 됩니다.  가우스-요르단 소거법은 행렬의 해를 구하기 위해 반복적인 단계를 사용하며, 최종 결과는 각 변수의 값을 포함하는 해를 제공합니다.  **2.3 INVERSE MATRIX**  Inverse Matrix(역행렬)은 주어진 정방 행렬(square matrix)과 곱하여 항등 행렬(identity matrix)을 얻는 행렬을 의미합니다.  행렬 \* 역행렬 = 역행렬 \* 행렬 = 항등행렬  역행렬을 구하는 것은 선형 대수학에서 중요한 주제 중 하나이며, 다양한 응용 분야에서 사용됩니다. 역행렬을 구하는 방법 중 가장 일반적인 방법은 가우스-조르단 소거법(Gauss-Jordan elimination)을 사용하는 것입니다. 이 방법은 주어진 행렬을 삼각 행렬로 변환한 후, 다시 역행렬을 구합니다.  그러나 역행렬이 존재하기 위해서는 몇 가지 조건이 필요합니다:  1. 행렬은 정방 행렬(square matrix)이어야 합니다. 즉, 행과 열의 개수가 동일해야 합니다.  2. 행렬의 행렬식(determinant)이 0이 아니어야 합니다. 이는 행렬이 특이(non-singular)해야 한다는 것을 의미합니다. 특이하지 않은 행렬은 역행렬을 가집니다.  역행렬은 다양한 응용에서 사용되며, 예를 들어 선형 방정식의 해를 구하거나, 최소 자승 문제(least squares problem)의 해를 찾는 등의 계산에 활용됩니다. 또한, 역행렬은 행렬 방정식을 해결하는 데 사용되는 데 있어서 중요한 도구입니다.  **2.4 DECOMPOSITION (FACTORIZATION)**  **2.4.1 LU Decomposition (Factorization): Triangularization**  LU Decomposition은 행렬을 하삼각 행렬(Lower Triangular Matrix)과 상삼각 행렬(Upper Triangular Matrix)의 곱으로 분해하는 행렬 분해 방법입니다. 이것은 대부분 가우스 소거법과 관련이 있으며, 선형 방정식 시스템의 해를 구하는 데 사용됩니다.  LU Decomposition의 기본 아이디어는 다음과 같습니다:  1. 주어진 행렬 A 를 하삼각 행렬 L 과 상삼각 행렬 U 의 곱으로 분해합니다. 즉, A = LU  2. L은 단위 행렬(Identity matrix)에 가까운 하삼각 행렬이고, U 는 상삼각 행렬입니다.  3. 이렇게 하면 행렬 A를 L 과 U의 곱으로 나타낼 수 있습니다.  4. 이 분해된 L과 U를 사용하여 선형 방정식 시스템의 해를 효율적으로 계산할 수 있습니다.  LU Decomposition은 계산 효율성을 향상시키는 데 유용하며, 특히 동일한 계수 행렬에 대해 여러 개의 선형 방정식 시스템을 푸는 경우에 효율적입니다. 또한, LU Decomposition은 행렬의 역행렬을 구하는 데도 사용될 수 있습니다.  Triangularization(삼각화)은 행렬을 삼각 행렬로 변환하는 프로세스를 나타냅니다. LU Decomposition에서는 행렬을 하삼각 행렬과 상삼각 행렬로 분해하므로, 삼각화라는 용어가 사용됩니다. 이 과정은 가우스 소거법을 사용하여 주어진 행렬을 삼각 행렬로 변환하는 것을 의미합니다. 이렇게 하면 행렬의 해를 더 쉽게 계산할 수 있습니다.  **2.4.2 Other Decomposition (Factorization): Cholesky, QR, and SVD**  LU 분해 외에도, 수치 선형 대수학에서 널리 사용되는 여러 가지 다른 행렬 분해(팩터화) 방법이 있습니다. 그 중 중요한 세 가지는 쇼레스키(Cholesky) 분해, QR 분해 및 특이값 분해(SVD)입니다.   1. **쇼레스키 분해**: 쇼레스키 분해는 대칭이고 양의 정부호 행렬에 적용됩니다. 이 방법은 대칭 양의 정부호 행렬을 하삼각 행렬과 그 공액 전치행렬의 곱으로 분해합니다. 수학적으로, 행렬 *A*가 대칭 양의 정부호 행렬이라면, *A*=*LLT*와 같이 분해될 수 있습니다. 여기서 *L*은 하삼각 행렬입니다. 이 분해는 선형 방정식의 시스템을 해결하는 데, 최적화 문제를 해결하는 데, 그리고 상관관계 있는 무작위 변수를 생성하는 데 사용됩니다. 2. **QR 분해**: QR 분해는 행렬을 직교 행렬 *Q*와 상삼각 행렬 *R*의 곱으로 분해합니다. 수학적으로, *A*가 선형 독립인 열을 가진 *m*×*n* 행렬이라면, *A*=*QR*로 분해됩니다. 여기서 *Q*는 *m*×*m* 직교 행렬이고 *R*은 *m*×*n* 상삼각 행렬입니다. QR 분해는 최소 자승 문제를 해결하는 데, 고유값 계산에 사용되며, 수치 최적화에 사용됩니다. 3. **특이값 분해 (SVD)**: 특이값 분해는 행렬을 세 행렬의 곱으로 분해합니다. 이 세 행렬은 좌 특이 행렬 *U*, 대각 특이값 행렬 ΣΣ, 그리고 우 특이 행렬 *VT*입니다. 주어진 *m*×*n* 행렬 *A*에 대해, SVD는 *A*=*U*Σ*VT*로 분해됩니다. 여기서 *U*는 *m*×*m* 직교 행렬, Σ는 *m*×*n* 대각 행렬이고, *VT*는 *n*×*n* 직교 행렬입니다. 특이값 분해는 차원 축소, 이미지 압축, 데이터 근사화, 최소 자승 문제 해결 등의 다양한 응용 프로그램에 사용됩니다. | |
| 질문 내용 | 1. **SVD(Singular Value Decomposition)는 왜 이미지 압축과 같은 응용 분야에서 효과적으로 사용되는가요?** | |