Assignment 4 Report

Akshit Vakati Venkata MM23B009

April 12, 2024

1 Breaking The Spectre:

1.1 *Spectral Theorem:*[1]

Given any $n \times n$ **Hermitian matrix** A , there exists an $n \times n$ **Unitary matrix** U and an $n \times n$ **Diagonal matrix** of real values Λ , such that $A = U\Lambda U^H$

1.2 *Introduction:*

- Hermitian matrix

  A matrix B is said to be *Hermitian* if $B^H = B$. Where B^H is the *transpose conjugate* of B.

  For example:

  $$B = \begin{bmatrix} 1 & 2i & -3 & 4 \\ -2i & 5 & 0 & -6i \\ -3 & 0 & 7 & -8i \\ 4 & 6i & 8i & 9 \end{bmatrix}$$

  is an example of *Hermitian Matrix*

- Unitary Matrix

  In linear algebra, an invertible complex square matrix U is unitary if its matrix inverse U^{-1} equals its conjugate transpose U^*, that is, if

  $$U^*U = UU^* = I,$$

  where I is the identity matrix.

  For example:

  $$U = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix}$$

  is an example of *Unitary Matrix*

- Diagonal matrix

  A diagonal matrix is a matrix in which all off-diagonal entries are zero. That is, the matrix $D = (d_{ij})$ with n columns and n rows is diagonal if

  $$\forall i, j \in \{1, 2, ..., n\}, i \neq j \rightarrow d_{ij} = 0$$

  $$\begin{bmatrix} d_1 & 0 & \cdots & 0 \\ 0 & d_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & d_n \end{bmatrix}$$

[1] https://en.wikibooks.org/wiki/Linear_Algebra/Spectral_Theorem

- *How to calculate the eigenvalues of a given Matrix*

  Given a square matrix A, to find its eigenvalues, we solve the characteristic equation:

  $$\det(A - \lambda I) = 0$$

  where I is the identity matrix and λ is the eigenvalue.

  Expanding the determinant gives:

  $$\det(A - \lambda I) = \begin{vmatrix} a_{11} - \lambda & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} - \lambda & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} - \lambda \end{vmatrix}$$

  Calculate the determinant and set it equal to zero:

  $$\det(A - \lambda I) = 0$$

Now in the given Task:The columns of U are:

$$U = \begin{pmatrix} \vec{u}_1, & \vec{u}_2, & \ldots, & \vec{u}_n \end{pmatrix}$$

$$\text{where } \begin{pmatrix} \vec{u}_1, & \vec{u}_2, & \ldots, & \vec{u}_n \end{pmatrix} \text{ are the eigenvectors of } U.$$

and the diagonal entries of Λ are

$$\Lambda = \begin{pmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_n \end{pmatrix}$$

$$\text{where } \begin{pmatrix} \lambda_1, & \lambda_2, & \ldots, & \lambda_n \end{pmatrix} \text{ are the corresponding eigenvalues of } U.$$

- The given Matrix A can be decomposed into a spectrum of rank 1 projections.

$$A = \sum_{i=1}^{n} \lambda_i (\vec{u}_i \vec{u}_i^H)$$

1.3 Classifying the given matirx using eigen values

Table 1: Matrix Classifications

Matrix Type	Eigenvalue Properties
Hermitian Matrix	All real values.
Unitary Matrix	Modulus is 1 ($\lvert\lambda_i\rvert = 1$).
Positively Semidefinite Matrix	All eigenvalues of A are non-negative ($\lvert\lambda_i\rvert \leq 1$).
Positively Definite Matrix	All eigenvalues of A are strictly positive ($\lvert\lambda_i\rvert \geq 1$).
Normal Matrix	$AA^\dagger = A^\dagger A$ (orthogonal eigenvectors).

1.4 How is the Diagonalization applied on matrix.

1. Let us consider a **Hermitian Matrix** A

$$A = \begin{bmatrix} 2+3i & 1-2i & -5 \\ 1+2i & 4 & 3-6i \\ -5 & 3+6i & 7 \end{bmatrix}$$

2. The characteristic polynomial of A is given by:

$$\det(A - \lambda I) = 0$$

3. Let U be the unitary matrix and the Λ be the diagonal matrix of eigenvectors

$$U = \begin{bmatrix} -0.2163 - 0.0396i & -0.1920 - 0.3667i & 0.3502 - 0.2974i \\ -0.0473 + 0.1458i & -0.9588 - 0.0809i & -0.2719 + 0.6361i \\ 0.9725 - 0.1213i & -0.2091 - 0.0749i & 0.4769 - 0.1726i \end{bmatrix}$$

$$\Lambda = \begin{bmatrix} 11.5127 + 0.8751i & 0 & 0 \\ 0 & -2.8781 + 0.5227i & 0 \\ 0 & 0 & 4.3654 - 0.3978i \end{bmatrix}$$

4. So the diagonalized form for A is :

$$A = U\Lambda U^{-1}$$

5. The given Matrix A can be decomposed into a spectrum OF PROJECTIONS.

$$A = \sum_{i=1}^{n} \lambda_i (\vec{u}_i \vec{u}_i^H)$$

2 Optimaze:

2.1 Introduction

Optimization methods are fundamental in various fields such as machine learning, statistics, and engineering. They are used to find the optimal solution to a problem by minimizing or maximizing an objective function. In this report, we will delve into first-order and second-order optimization methods, exploring their mathematical formulations, advantages, and applications.

2.2 First-Order Optimization

First-order optimization methods, also known as gradient-based methods, utilize the gradient of the objective function to iteratively update the parameters. The gradient points in the direction of the steepest ascent of the function. Thus, by moving in the *opposite direction of the gradient*, we can **minimize the function.** The general update rule for first-order methods is given by:

$$\theta^{(t+1)} = \theta^{(t)} - \alpha \nabla f(\theta^{(t)})$$

where $\theta^{(t)}$ represents the parameters at iteration t, α is the learning rate, and $\nabla f(\theta^{(t)})$ is the gradient of the objective function.

Common first-order optimization algorithms include:

- Gradient Descent
- Stochastic Gradient Descent (SGD)

For example:
The update rule for the given problem by gradient descent is the task to find the optimal equation of the plane $ax + by + cz = 1$,fitting the behavior of the system of points (x, y, z), to the best possible extent, which are (a, b, c).

In the following discussion:

1. A denotes matrix A, while b represents vector b.
2. The superscript t indicates the t-th time step.
3. θ^t denotes vector (a, b, c), and $\Delta\theta^t$ represents vector $(\Delta a, \Delta b, \Delta c)$.
   on considering the objective function as the mean squared error (MSE).

$$L = \frac{1}{n}\sum_{i=1}^{n}(ax_i + by_i + cz_i - 1)^2$$

4. **Gradient:** A vector pointing in the direction of the average steepest ascent of the mean squared error (MSE) of the square distances of the points from the plane, guiding optimization algorithms like gradient descent by indicating the average fastest increase in the MSE. Where Gradient can be calculated as follows for the given objective function:

$$\nabla L = \begin{bmatrix} \frac{\partial L}{\partial a} \\ \frac{\partial L}{\partial b} \\ \frac{\partial L}{\partial c} \end{bmatrix}$$

$$\frac{\partial L}{\partial a} = \frac{2}{n}\sum_{i=1}^{n}(ax_i + by_i + cz_i - 1)x_i$$

$$\frac{\partial L}{\partial b} = \frac{2}{n}\sum_{i=1}^{n}(ax_i + by_i + cz_i - 1)y_i$$

$$\frac{\partial L}{\partial c} = \frac{2}{n}\sum_{i=1}^{n}(ax_i + by_i + cz_i - 1)z_i$$

5. **Hessian Matrix:** A square matrix of second-order partial derivatives providing information about the average curvature of the mean squared error (MSE) of the square distances of the points from the plane around a specific point, crucial for determining the nature of critical points and guiding optimization algorithms.

$$H = \begin{bmatrix} \frac{\partial^2 L}{\partial a^2} & \frac{\partial^2 L}{\partial a \partial b} & \frac{\partial^2 L}{\partial a \partial c} \\ \frac{\partial^2 L}{\partial a \partial b} & \frac{\partial^2 L}{\partial b^2} & \frac{\partial^2 L}{\partial b \partial c} \\ \frac{\partial^2 L}{\partial a \partial c} & \frac{\partial^2 L}{\partial b \partial c} & \frac{\partial^2 L}{\partial c^2} \end{bmatrix}$$

$$\frac{\partial^2 L}{\partial a^2} = \frac{2}{n} \sum_{i=1}^{n} x_i^2$$

$$\frac{\partial^2 L}{\partial b^2} = \frac{2}{n} \sum_{i=1}^{n} y_i^2$$

$$\frac{\partial^2 L}{\partial c^2} = \frac{2}{n} \sum_{i=1}^{n} z_i^2$$

$$\frac{\partial^2 L}{\partial a \partial b} = \frac{\partial^2 L}{\partial b \partial a} = \frac{2}{n} \sum_{i=1}^{n} x_i y_i$$

$$\frac{\partial^2 L}{\partial a \partial c} = \frac{\partial^2 L}{\partial c \partial a} = \frac{2}{n} \sum_{i=1}^{n} x_i z_i$$

$$\frac{\partial^2 L}{\partial b \partial c} = \frac{\partial^2 L}{\partial c \partial b} = \frac{2}{n} \sum_{i=1}^{n} y_i z_i$$

6. Update the parameters θ^t using the following update rule:

$$\theta^{t+1} = \theta^t - \Delta\theta^t$$

7. $\Delta\theta^t$ is calculated using Newton's steepest descent method$\Delta\theta^t = H^{-1} g^t$
where H is the Hessian and g is the Gradient, which are: $H_{ij} = \frac{\partial^2 L}{\partial \theta_i \partial \theta_j}, \quad g_i = \frac{\partial L}{\partial \theta_i}$

8. Keep track of the error at each step, computed similarly to the objective function. On Repeating the following steps until this error is less than a threshold, say δ. Then we are printing the optimal parameters with the number of steps (epochs) taken for the solution to converge to θ^*.

2.3 Second-Order Optimization

Second-order optimization methods utilize not only the gradient but also the Hessian matrix of the objective function. The Hessian matrix contains second-order partial derivatives and provides information about the curvature of the function. By incorporating information about the curvature, second-order methods can converge faster than first-order methods.

Common second-order optimization algorithms include:

- Newton's Method :

  The general update rule for second-order methods is given by:

$$\theta^{(t+1)} = \theta^{(t)} - \alpha(\nabla^2 f(\theta^{(t)}))^{-1} \nabla f(\theta^{(t)})$$

  where $\nabla^2 f(\theta^{(t)})$ represents the Hessian matrix of the objective function.

- Quasi-Newton Methods (e.g., BFGS, L-BFGS):

  Quasi-Newton uses inverse Hessian estimation to compute each update iteration. There are a few estimation algorithms available but the most well-known is the Broyden-Fletcher-GolfarbShanno (BFGS) algorithm. The estimate inverse Hessian, Q for BFGS is computed at each iteration with the following equation, The update rules for the parameter vector θ and the approximation of the inverse Hessian matrix H in the Quasi-Newton method are given by:

$$d\theta = -\alpha H \nabla f(\theta)$$
$$dH = \frac{\Delta y \Delta y^T}{\Delta y^T \Delta s} - \frac{H \Delta s \Delta s^T H}{\Delta s^T H \Delta s}$$

2.4 Conclusion

In conclusion, both first-order and second-order optimization methods are powerful tools for minimizing or maximizing objective functions. Here are key considerations:

- First-order methods, like gradient descent, are computationally efficient and easy to implement, making them suitable for large-scale problems.
- However, they may converge slowly, especially in complex or non-convex scenarios.
- Second-order methods, such as Newton's method, converge faster due to their incorporation of curvature information.
- Yet, they require more computational resources to compute and store second-order derivatives or approximations.
- The choice between methods depends on factors like dataset size, computational resources, and desired convergence rate, balancing efficiency with speed.

For further reading and detailed mathematical formulations, refer to the following sources:

1. Review of second-order optimization techniques in artificial neural networks back propagation [2].
2. Boyd, S., & Vandenberghe, L. (2004). *Convex Optimization*. Cambridge University Press.
3. Nocedal, J., & Wright, S. J. (2006). *Numerical Optimization*. Springer.

[2] https://iopscience.iop.org/article/10.1088/1757-899X/495/1/012003/pdf