# MODULE I

# DETERMINISTIC OPTIMIZATION

By: Dr. A K Yadav (9911375598)
Dept of CSE, ASET, AUUP

# Module I - Topics to be covered



- Overview on Modern Cryptography
- Number Theory
- Probability and Information Theory
- Classical Cryptosystems
- Cryptanalysis of Classical Cryptosystems
- Shannon's Theory

# Recommended Reading




- Geof H. Givens and Jennifer A. Hoeting, "Computational Statistics", 2nd Edition, Wiley
- Christian P. Robert George Casella, "Monte Carlo Statistical Methods", Springer
- Boyd and Vandenberghe, "Convex Optimization", Cambridge University Press

# Objective and Learning Outcome



**Objective:**

- To learn about applications of Optimization algorithms
- Broad working knowledge of modern computational statistics
- Practical understanding of how and why existing methods work
- Enabling effective use of modern statistical methods
- To explore the use of approach for solving real life problems

**Learning Outcomes**

After completion of the course, you will be able to:

- Basic understanding of computational statistics.
- Optimization Techniques understanding.
- Able to understand Moneo Carlo Methods.
- Able to understand Markov Methods.

# Module Assessment and Self Work



**Module Assessment**

- Quiz
- Assignment

**PSDA (Self Work)**

- Minor Experiment
- Group Discussion
- Case study

# Newton-Raphson

# Newton-Raphson



- It is used for finding the root of the continuous, differentiable function.
- If we know, the root we are looking for is near the point $x = x_0$
- The Newton's method state that the better choice of the root is
- $x_1 = x_0 - f(x_0)/f'(x_0)$
- This can be generalized as
- $x_{n+1} = x_n - f(x_n)/f'(x_n)$

# MODULE I

# DETERMINISTIC OPTIMIZATION

By: Dr. A K Yadav (9911375598)
Dept of CSE, ASET, AUUP

# Conjugate Gradients



- The conjugate gradient is a method for solution of linear equations $Ax = b$
- $A$ matrix must be positive-definite.
- The method is often implemented as an iterative algorithm
- The method is applicable to sparse systems that are too large to be handled by a direct implementation or other direct methods
- Large sparse systems often arise when numerically solving partial differential equations or optimization problems
- The conjugate gradient method can also be used to solve unconstrained optimization problems such as energy minimization

- Suppose we want to solve the system of linear equations
- $Ax = b$
- Where $x$ is vector and $A$ is a symmetric, positive-definite and real valued matrix of size $n \times n$. $b$ is also known.
- We want to find out the value of vector $x$

# Algorithm



```
function x = conjgrad(A, b, x)
    r = b - A * x;
    p = r;
    rsold = r' * r;
    for i = 1:length(b)
        Ap = A * p;
        alpha = rsold / (p' * Ap);
        x = x + alpha * p;
        r = r - alpha * Ap;
        rsnew = r' * r;
        if sqrt(rsnew) < 1e-10
            break
        end
        p = r + (rsnew / rsold) * p;
        rsold = rsnew;
        end
    end
```

# APPLIED PROBABILITY AND STATISTICS

By: Dr. A K Yadav (9911375598)
Dept of CSE, ASET, AUUP

# Quasi-Newton methods



- The discrete Newton method strategy for numerically approximating the Hessian by $M^{(t)}$ is a computationally burdensome one.
- At each step, $M^{(t)}$ is wholly updated by calculating a new discrete difference for each element.
- A more efficient approach can be designed, based on the direction of the most recent step.
- When $x^{(t)}$ is updated to $x^{(t+1)} = x^{(t)} + h^{(t)}$, the opportunity is presented to learn about the curvature of $g$ in the direction of $h^{(t)}$ near $x^{(t)}$.
- Then $M^{(t)}$ can be efficiently updated to incorporate this information
- To do this, we must abandon the componentwise discrete-difference approximation to $\boldsymbol{g''}$ used in the discrete Newton method.

- However, it is possible to retain a type of secant condition based on differences.
- Specifically, a secant condition holds for $M^{(t+1)}$ if
  - $g'(x^{(t+1)}) - g'(x^{(t)}) = M^{(t+1)}(x^{(t+1)} - x^{(t)})$
- This condition suggests that we need a method to generate $M^{(t+1)}$ from $M^{(t)}$ in a manner that requires less calculations and satisfies the above equation.

# Applied Probability and Statistics

Module-1, Lecture-4

By: Dr. A K Yadav (9911375598)
Dept of CSE, ASET, AUUP

# Univariate Fisher scoring



- As we know $I(\theta)$ can be approximated by $-l''(\theta)$ in likelihood inference.
- In optimization of $g$ in an MLE problem, $-l''(\theta)$ cab be replaced by $I(\theta)$ in Newton update.
- This yields an updating increment of $h(t) = l'(\theta^{(t)})/I(\theta^{(t)})$ where $I(\theta^{(t)})$ is the expected Fisher information evaluated at $\theta^{(t)}$.
- So the updating equation is $\theta^{(t+1)} = \theta^{(t)} + l'(\theta^{(t)})I(\theta^{(t)})^{-1}$
- This approach is called Fisher scoring.

# Univariate Fisher scoring



- Fisher scoring and Newton's method both have the same asymptotic properties
- But for individual problems one may be computationally or analytically easier than the other.
- Generally, Fisher scoring works better in the beginning to make rapid improvements
- While Newton's method works better for refinement near the end.

# Multivariate Fisher scoring



- To use the Newton's method update, we again approximate $g(x^*)$ by the quadratic Taylor series expansion

$$g(\mathbf{x}^*) = g(\mathbf{x}^{(t)}) + (\mathbf{x}^* - \mathbf{x}^{(t)})^{\mathrm{T}}\mathbf{g}'(\mathbf{x}^{(t)}) + \frac{1}{2}(\mathbf{x}^* - \mathbf{x}^{(t)})^{\mathrm{T}}\mathbf{g}''(\mathbf{x}^{(t)})(\mathbf{x}^* - \mathbf{x}^{(t)})$$

- and maximize this quadratic function with respect to $x^*$ to find the next iterate.
- Setting the gradient of the right-hand side equal to zero yields

$$\mathbf{g}'(\mathbf{x}^{(t)}) + \mathbf{g}''(\mathbf{x}^{(t)})(\mathbf{x}^* - \mathbf{x}^{(t)}) = 0$$

This provides the update

$$\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} - \mathbf{g}''(\mathbf{x}^{(t)})^{-1}\mathbf{g}'(\mathbf{x}^{(t)})$$

- The multivariate Newton increment is $h^{(t)} = -g''(x^{(t)})^{-1}g'(x^{(t)})$
- As in the univariate case, in MLE problems we may replace the observed information at $\theta^{(t)}$ with $I(\theta^{(t)})$, the expected Fisher information at $\theta^{(t)}$.
- This yields the multivariate Fisher scoring approach with update given by $\theta^{(t+1)} = \theta^{(t)} + I(\theta^{(t)})^{-1}I'(\theta^{(t)})$
- This method is asymptotically equivalent to Newton's method.

# EM algorithm



- The EM algorithm iteratively seeks to maximize $L(\theta|x)$ with respect to $\theta$
- Let $\theta^{(t)}$ denote the estimated maximizer at iteration $t$, for $t = 0,1, \ldots$
- Define $Q(\theta|\theta^{(t)})$ to be the expectation of the joint log likelihood for the complete data, conditional on the observed data $X = x$

$$
\begin{aligned}
Q(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t)}) &= E\left\{\log L(\boldsymbol{\theta}|\mathbf{Y}) \;\middle|\; \mathbf{x}, \boldsymbol{\theta}^{(t)}\right\} \\
&= E\left\{\log f_{\mathbf{Y}}(\mathbf{y}|\boldsymbol{\theta}) \;\middle|\; \mathbf{x}, \boldsymbol{\theta}^{(t)}\right\} \\
&= \int \left[\log f_{\mathbf{Y}}(\mathbf{y}|\boldsymbol{\theta})\right] f_{\mathbf{Z}|\mathbf{X}}(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta}^{(t)})\, d\mathbf{z},
\end{aligned}
$$

- The last equation emphasizes that $Z$ is the only random part of $Y$ once we are given $X = x$
- EM is initiated from $\theta^{(0)}$ then alternates between two steps:
- E for expectation and M for maximization.
- The algorithm is summarized as:

1. **E step:** Compute $Q(\theta|\theta^{(t)})$
2. **M step:** Maximize $Q(\theta|\theta^{(t)})$ with respect to θ and set $\theta^{(t+1)}$ equal to the maximizer of $Q$
3. Return to the **E step** unless a stopping criterion has been met

- Stopping criteria for optimization may be based upon:
- $(\theta^{(t+1)} - \theta^{(t)})^T(\theta^{(t+1)} - \theta^{(t)})$ or $|Q(\theta^{(t+1)}|\theta^{(t)}) - Q(\theta^{(t)}|\theta^{(t)})|$

# EM Variants



- Improving the E Step
- Improving the M Step
- Acceleration Methods:
  - Aitken Acceleration
  - Quasi-Newton Acceleration

# Applied Probability and Statistics

Module-1, Lecture-5

By: Dr. A K Yadav (9911375598)
Dept of CSE, ASET, AUUP

# LU Decomposition



- Suppose we can write the matrix $A_{NN}$ as a product of two matrices
- $A = LU$-Eq. 1 where $L$ is lower triangular and $U$ upper triangular

$$\begin{bmatrix} \alpha_{11} & 0 & 0 & 0 \\ \alpha_{21} & \alpha_{22} & 0 & 0 \\ \alpha_{31} & \alpha_{32} & \alpha_{33} & 0 \\ \alpha_{41} & \alpha_{42} & \alpha_{43} & \alpha_{44} \end{bmatrix} \cdot \begin{bmatrix} \beta_{11} & \beta_{12} & \beta_{13} & \beta_{14} \\ 0 & \beta_{22} & \beta_{23} & \beta_{24} \\ 0 & 0 & \beta_{33} & \beta_{34} \\ 0 & 0 & 0 & \beta_{44} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix}$$

- We can use a decomposition to solve the linear set $Ax = b$ Eq. 2
- $Ax = LUx = Ly = b$ Eq. 3
- First solving for the vector $y$ such that $Ly = b$ Eq. 4
- Then solving $Ux = y$ Eq. 5

- What is the advantage of breaking up one linear set into two successive ones?
- Eq. 4 can be solved by forward substitution as follows
- $y_1 = \frac{b_1}{\alpha_{11}}$ and $y_i = \frac{1}{\alpha_{ii}}[b_i - \sum_{j=1}^{i-1} \alpha_{ij} y_j]$ $i = 2,3, \ldots, N$ Eq. 6
- After that Eq. 5 can be solved by back substitution
- $x_{N=} \frac{y_N}{\beta_{NN}}$ and $x_i = \frac{1}{\beta_{ii}}[y_i - \sum_{j=i+1}^{N} \beta_{ij} x_j]$ $i = N, N-1, \ldots, 1$ Eq. 7

- How can we solve for $L$ and $U$, given $A$?
- First, we write out the $i, j^{th}$ component of Eq. 1
- $\alpha_{i1}\beta_{1j} + \cdots + \alpha_{ij}\beta_{ij} + \cdots + \alpha_{iN}\beta_{Nj} = a_{ij}$
- The number of terms in the sum depends on whether $i$ is smaller or $j$.
- Three cases will be there:
- $i < j$: $\alpha_{i1}\beta_{1j} + \alpha_{i2}\beta_{2j} \ldots + \alpha_{ii}\beta_{ij} = a_{ij}$ *Eq. 8*
- $i = j$: $\alpha_{i1}\beta_{1j} + \alpha_{i2}\beta_{2j} \ldots + \alpha_{ii}\beta_{jj} = a_{ij}$ *Eq. 9*
- $i > j$: $\alpha_{i1}\beta_{1j} + \alpha_{i2}\beta_{2j} \ldots + \alpha_{ij}\beta_{jj} = a_{ij}$ *Eq. 10*

# Crout's algorithm



- Set $\alpha_{ii} = 1,\ i = 1, \ldots, N$
- For each $j = 1, 2, 3, \ldots, N$ do these two procedures: First, for $i = 1, 2, \ldots, j$, use Eq 8 and 9 to solve for $\beta_{ij}$, namely

$$\beta_{ij} = a_{ij} - \sum_{k=1}^{i-1} \alpha_{ik}\beta_{kj}. \quad \text{Eq. 12}$$

(When $i = 1$ in Eq 12 the summation term is taken to mean zero.) Second, for $i = j + 1, j + 2, \ldots, N$ use Eq. 10 to solve for $\alpha_{ij}$, namely

$$\alpha_{ij} = \frac{1}{\beta_{jj}} \left( a_{ij} - \sum_{k=1}^{j-1} \alpha_{ik}\beta_{kj} \right). \quad \text{Eq 13}$$

Be sure to do both procedures before going on to the next $j$.

- How to Find the Inverse of $N \times N$ Matrix?
- Compute the determinant of the given matrix
- Calculate the determinant of $\mathrm{N} - 1 \times N - 1$ minor matrices
- Formulate the matrix of cofactors
- Take the transpose of the cofactor matrix to get the adjugate matrix
- Divide each term of the adjugate matrix by the determinant
- $A^{-1} = \frac{1}{\det(A)} adj(A)$

# Inverse of Matrix



- $AA^{-1} = I$
- $AB = I$ where $B = A^{-1}$
- We know $A = LU$, we have $I$ and we must find out $B$
- Find out $B$ column wise
- $LUB = I$
- $UB = Y$
- $LY = I$

# Applied Probability and Statistics

Module-1, Lecture-6

By: Dr. A K Yadav (9911375598)
Dept of CSE, ASET, AUUP

# Cholesky Decomposition



- The Cholesky decomposition or Cholesky factorization is a decomposition of a symmetric, positive-definite matrix into the product of a lower triangular matrix and its conjugate transpose.
- The Cholesky decomposition is roughly twice as efficient as the $LU$ decomposition for solving systems of linear equations.
- Suppose we can write the matrix $A_{NN}$ as a product of two matrices that is $A = LL^T$ Eq. 1 where $L$ is lower triangular and $L^T$ is its transpose.

- We can use a decomposition to solve the linear set $Ax = b$ Eq. 2
- $AX = LL^T X = L(L^T X) = LY = B$ Eq.3
- First solving for the vector $Y$ such that $LY = B$ Eq. 4
- Then solving $L^T X = Y$ Eq. 5
- $L_{ii} = \sqrt{A_{ii} - \sum_{k=1}^{i} L_{ik}^2}$

# Applied Probability and Statistics

Module-1, Lecture-7

By: Dr. A K Yadav (9911375598)
Dept of CSE, ASET, AUUP

# Low Rank Updates for the Cholesky Decomposition



Suppose $A \in \mathbf{R}^{n\times n}$ is nonsingular, $u, v \in \mathbf{R}^n$ with $1 + v^T A^{-1}u \neq 0$, and we want to solve two sets of linear equations

$$Ax = b, \qquad (A + uv^T)\tilde{x} = b.$$

The solution $\tilde{x}$ of the second system is called a *rank-one update* of $x$. The matrix inversion lemma allows us to calculate the rank-one update $\tilde{x}$ very cheaply, once we have computed $x$. We have

$$\begin{aligned} \tilde{x} &= (A + uv^T)^{-1}b \\ &= (A^{-1} - \frac{1}{1 + v^T A^{-1}u} A^{-1}uv^T A^{-1})b \\ &= x - \frac{v^T x}{1 + v^T A^{-1}u} A^{-1}u. \end{aligned}$$

We can therefore solve both systems by factoring $A$, computing $x = A^{-1}b$ and $w = A^{-1}u$, and then evaluating

$$\tilde{x} = x - \frac{v^T x}{1 + v^T w} w.$$