Simple Minimization Formulation

Dr. Kalidas Y.

By the end of this lecture, you will understand how to craft a simple minimization function

Loss function minimization

- *A function* that *outputs a real number*
- That function takes as *input some parameters*
- By *searching for parameter values*
- Use a *smart search*
  - Gradient descent
  - Newton-Raphson
  - Other
- That loss function should capture *some meaning of the domain*
- There can be *several local minimia*
- *Loss surface* may not be smooth

Key Steps

- STEP 1: Input is a vector (e.g. w vector)
  - If it is a matrix, flatten to a vector and input
  - *It is an array of numbers*
  - *It corresponds to parameters*
- STEP 2: Output is a *number*
- STEP 3: Choose a *minimization method*
  - There are a dozen minimization methods (aka *search methods*)
  - For example, for Gradient Descent – You need to supply gradient function (aka Jacobian) $\nabla L(w)$
  - You may not need to supply gradient function
    - Nelder-Mead's method
    - Variants of Secant method
    - Popular – BFGS method (Broyden-Fletcher-Goldfarb-Shanno)

Revisiting the linear regression formulation

- Example of squared error: $L(w) = (w \cdot x - y)^2$
  - Here x and y are given and therefore constant
  - w is parameter vector
- x is a vector of k dimensions (k x 1 matrix)
- w is a vector of k dimensions (k x 1 matrix)
- x.w is the dot product of the x and w vectors
- y is the given actual value
- On the right side, squared error is calculated

Other non-trivial examples… To find inverse of a matrix?

- Given: Let A be a square matrix
- Task: We need to find $B = A^{-1}$ inverse of A
- Constraint: We need to use Loss function minimization formulation

- $L(B) = norm\big(vector(BA - I)\big) = 0$

An example of low dimensional transformation

(1)

Let input be k dimensional

There are N data points

The X matrix *is of shape* (N x k)

(2)

Task: Each and every k dimensional point we need to transform to become a 2 dimensional point

(3)

Let Q be the transformation matrix *of shape* (k x 2)

New $x \vdash Qx$

The transformation is $X_{N \times k}\, Q_{k \times 2} = X'_{N \times 2}$

(4)

Constraint: It should satisfy some constraints… such as *pairwise similarities/distances be maintained*

(5)

Loss function: Design a loss function and minimize it!

We need: $XX^T = X'X'^T$

Minimum difference in pairwise similarities

: $XX^T - XQ(XQ)^T = XX^T - X(QQ^T)X^T$

(6)

Write a program to code for loss function

$L(Q) = ||vec(XX^T - X(QQ^T)X^T)||_2$