DS 211: Numerical Optimization

Lokesh Mohanty

October 17, 2022

Contents

1 Introduction

Optimization: objective -> function of decision variables

Data-driven modelling (machine learning) -> used where forward modelling is not practical

2 Linear Algebra

- functions: maps an input space to an output space
- scalar valued function: output is a scalar
- vector valued function: output is a vector
- countour map -> contour line(iso line) -> scalar representation for a vector (using a function)
- in a countour map, derivative always moves in the direction the function value increases
- finite derivative, finite volume, finite element methods -> methods to solve derivative numerically
- Tesselation: dividing a domain into a finite number of grid points
- Automatic differentiation
- Ax = λ x (eigen value decomposition)
- Av = σ u (singular value decomposition)
-
- Pseudo inverse -> S.V.D. (singular value decomposition) -> Moore-Penrose

- Matrix vector multiplication is a linear combination of matrix columns with the weights as the entries of the vector
- Matrix -> sum of eigen vector * eigen value -> E.V.D. (eigen value decomposition) -> dimension reduction
- Rectangular (Au = sigma v) -> S.V.D.

- Vector Norm -> L1($|x_1| + |x_2| + |x_3|$), L2($\sqrt{x_1^2 + x_2^2 + x_3^2}$), $Lp(\sqrt[p]{sumofpthpowerofallentries})Matrix$
$Frobeniusnorm(\sqrt{sumofsquaresofallentries})$

3 Optimization

Standard ML algorithm: output = function (input/features), function is not known error/loss = |output' - output|, output' is computed using the trained model (function)

Convex function

4 Unconstrained optimization algorithms

4.1 Empirical risk minimization method

4.2 Forward Modelling

4.2.1 Cholesky Factorization

4.2.2 QR Factorization

4.2.3 SVD Factorization

4.3 Backtracking (step length selection)

Rules: wolfe, goldstein

4.4 Iterative Algorithms

4.4.1 Line Search Method

Choose a direction and then the length to move

1. Gradient Descent (1st order)
   (a) Steepest Descent Algorithm

$$P_k = \frac{-\nabla f(\theta^k)}{||\nabla f(\theta^k)||}$$

(b) Batch Gradient Descent

- BGD: batch gradient descent
- mini BGD: mini batch gradient descent
- SGD: stochastic gradient descent

(c) Faster Optimizers

i. Momentum Optimization (momentum based)
ii. Nesterov Accelerated Gradient (momentum based)
iii. Ada Grad (adaptive learn based)
iv. RMS Prop (adaptive learn based)
v. ADAM, Adamax, Nadam, Adam W (adaptive moment acceleration)

2. Newton Method (2nd order)

   Newtons direction:

$$P_k = -\nabla^2 f_k^{-1} \nabla f_k$$

3. Quasi-Newton Method (2nd order)

4. Conjugate Gradient

$$x_{k+1} = x_k + \alpha_k p_k$$

   Coordinate Descent

   (a) p_k to be one of the coordinate

   (b)

$$\alpha : \min f(x_k + \alpha p_k)$$

   Problem 1: minimize f(x), $f(x) = 4x_1^2 + x_2^2$, (Separable function) Let $x^{(1)} = (-1, -1)$ and $p^{(1)} = (1, 0)$

$$\implies x^{(2)} = (-1+\alpha, -1) \implies f(x^{(1)}) = 4(-1+\alpha)^2+(-1)^2, \frac{\partial f(x)}{\partial \alpha} = 8(-1+\alpha) = 0 \implies \alpha = 1 =$$

   Problem 2: minimize f(x), $f(x) = 4x_1^2 + x_2^2 - 2x_1x_2$, (Non Separable function) Let $x^{(1)} = (-1, -1)$ and $p^{(1)} = (1, 0)$

$$\implies x^{(2)} = (-1+\alpha, -1) \implies f(x^{(1)}) = 4(-1+\alpha)^2+(-1)^2-2(-1+\alpha)(-1), \frac{\partial f(x)}{\partial \alpha} = 8(-1+\alpha)+2$$

   for non separable funcitons coordinate descent doesn't converge easily hence we need to go along the H-conjugate directions to converge fast.

Conjugate Gradient:

$$f(x) = \frac{1}{2}x^T Hx + c^T x + d$$

$$x = x^{(0)} + \sum_{1=0}^{n-1} \alpha^{(i)} p^{(i)}$$

$$\psi(\alpha) = \frac{1}{2}(x^{(0)} + \alpha^{(0)}p^{(0)})^T H(x^{(0)} + \alpha^{(0)}p^{(0)}) + c^T(x^{(0)}(x^{(0)} + \alpha^{(0)}p^{(0)})$$

$$\psi(\alpha) = \frac{1}{2}\alpha^T P^T HP\alpha + (Hx^{(0)} + c)^T P\alpha + \frac{1}{2}x^{(0)} + c^T x^{(0)}$$

A convex quadratic functions can be minimized in atmost n steps provided we search along the conjugate direction. It converges much faster if the eigen values are clustered together.

Pre-conditioned Conjugate Gradient (PCG): minimize

$$f(x) = \frac{1}{2}||Ax - b||^2_{grid} = \frac{1}{2}(Ax - b)^T(Ax - b)$$

to find x. Here f(x) is called the residual

$$\min f(x) = \frac{1}{2}x^T Hx + c^T x$$

->

$$\nabla f(x) = Hx + c = 0 \implies Hx^* = -c$$

Residual is $r_k = Hx_k - Hx^*$

$P_i HP_j = \{0, i \neq j; NZ, i = j\}$

Search Direction: $x_{k+1} = x_k - \alpha_k r_k$ Conjugate Gradient: $x_{k+1} = x_k + \alpha_k[-r_k + \beta_k P_{k-1}]$ $x_{k+1} = x_k + \alpha_k P_k$, $P_k = -r_k + \beta_k P_{k-1}$

$\phi(\alpha) = \frac{1}{2}(x_k + \alpha_k P_k)^T H(x_k + \alpha_k P_k) + c^T(x_k + \alpha_k P_k)$ $\min_\alpha \phi(\alpha)$ -> $\frac{\partial \phi}{\partial \alpha} = 0 \implies \alpha_k = \frac{-c^T p_k - P_k^T Hx_k}{P_k^T HP_k} = \frac{-(c+Hx_k)^T P_k}{P_k} = \frac{-r_k^T P_k}{P_k^T HP_k}$

$$P_{k-1}^T Hp_k = 0 \implies P_{k-1}^T H(-r_k + \beta_k P_{k-1}) = 0 \implies \beta_k = \frac{P_{k-1}^T Hr_k}{P_{k-1}^T HP_{k-1}}$$

Conjugate Gradient Algorithm: $P_0 = -r_0$, steepest descent Given x_0, $r_0 = Hx_0 + c$ while $||r_k|| > \epsilon_{thresold}$

(a) $\alpha_k = \frac{-r_k^T P_k}{P_k^T H P_k}$, [Exact min of f along P_k]

(b) $x_{k+1} = x_k + \alpha_k P_k$

(c) $r_{k+1} = Hx_{k+1} + c$

(d) $\beta_{k+1} = \frac{P_{k-1}^T H r_k}{P_{k-1}^T H P_{k-1}} = \frac{r_{k+1}^T H P_k}{P_k^T H P_k}$

(e) $P_{k+1} = -r_{k+1} + \beta_{k+1} P_k$

(f) $k + k + 1$

end while

- White noise
- Red noise

Pre-conditioned Conjugate Gradient: $Hx = -c$ (when small changes in c causes large changes in x i.e., unstable) -> $M^{-1}H = -M^{-1}c$, M: SPD and invertible

$k(M^{-1}H) << k(H)$ or $M^{-1}H$ has ev clusters for easier computation

4.4.2 Trust Region Method

Define a trust region and an approximation for the function in the trust region which can easily be minimized to find the direction

4.4.3 Derivative Free Method

1. Evolution Method (genetic)
2. Bayesian Method (probabilistic)

5 Machine Learning

5.1 Fitting the model

- the data is divided into batches and the model is trained for every batch
- epochs: number of times the model is trained with the same training set

5.2 Backpropagation

5.2.1 Algorithm runs for one mini-batch at a time

1. All mini-batches that result in a pass through the entire dataset is called an epoch
2.

5.2.2 Forward pass

1. Mini-batch is passed to the input layer, which sends it to the first hidden layer
2. Output from each neuron of the hidden layer is calculated and passed to the next layer (for every instance in a mini-batch)
3. Process is repeated until the mini-batch reaches the output layer
4. This process is similar to prediction, but now all intermediate results are saved

5.2.3 Loss calculation

Use a loss function to quantify the error of prediction with respect to the ground truth

5.2.4 Backward pass (Reverse pass)

1. Apply chain rule and calculate how sensitive is the error to each prarmeters in all layers by working backwards from the output layer to the input layer
2. Evaluate the gradient for the mini-batch loss with respect to the parameters in the backward pass

5.2.5 Gardient descent step

Apply the SGD algorithm (or ts variant) to update the parameters

5.2.6 Iterate

5.3 Example

5.3.1 Autodiff

$f(x, y) = x^2y + y + 2$

5.3.2 Autodiff for Logistic Regression

$f(w_1, w_2, b) = ylog(\sigma(x_1w_1 + x_2w_2 + b)) - (1 - y)log(1 - \sigma(x_1w_1 + x_2w_2 + b))$

6 Notes

BLAS: Basic Linear Algebra Set sklearn -> scipy -> BLAS -> FORTRAN -> Machine Language

7 Tasks [0/3]

- ☐ Read NW chapter 3
- ☐ Finish Assignment 1
- ☐ Revise whats taught till now