ECE 445 (ML for ENGG): Mini Jupyter Exercise #3

Waheed U. Bajwa (waheed.bajwa@rutgers.edu)

Gradient Descent

Consider the bivariate function $f : \mathbb{R}^2 \to \mathbb{R}$ that is defined as follows:

$$f(\mathbf{w}) = (w_1^2 + w_2 - 11)^2 + (w_1 + w_2^2 - 7)^2.$$

Provide an implementation of gradient descent that minimizes this function while adhering to the following requirements:

- You cannot use any optimization-related functions from `numpy`, `scipy`, etc. Your implementation must compute the gradient $\nabla f(\mathbf{w})$ numerically; in other words, you cannot analytically evaluate the gradient.
- Your implementation should declare that it has found a minimum $\mathbf{w}^*$ if (and when) $\|\nabla f(\mathbf{w}^*)\|_2$ falls below 10^{-12}. Your implementation should declare *failure* if it cannot find $\mathbf{w}^*$ within 10,000 iterations.

1. Initialize gradient descent from $\mathbf{w}^0 = \begin{bmatrix} 0 & -4 \end{bmatrix}^T$.
   - Run the algorithm with step size $\gamma = 0.005$ and, if the algorithm converges, output $\mathbf{w}^*$ and the number of iterations it took the algorithm to converge.
   - Run the algorithm with step size $\gamma = 0.01$ and, if the algorithm converges, output $\mathbf{w}^*$ and the number of iterations it took the algorithm to converge.
   - Comment on the convergence speed of gradient descent for this problem as a function of the step size.
2. Run gradient descent with step size $\gamma = 0.01$ for four different initializations: (i) $\mathbf{w}^0 = \begin{bmatrix} 0 & -4 \end{bmatrix}^T$; (ii) $\mathbf{w}^0 = \begin{bmatrix} 0.5 & -4 \end{bmatrix}^T$; (iii) $\mathbf{w}^0 = \begin{bmatrix} 0 & 4 \end{bmatrix}^T$; and (iv) $\mathbf{w}^0 = \begin{bmatrix} 0.5 & 4 \end{bmatrix}^T$.
   - Compare the solutions returned in each one of the four runs; are all the solutions the same? Based on the information available to you from these runs, are the solutions local minima or global minima?
   - Generate a contour plot of $f(\mathbf{w})$ over the region $[-5, 5] \times [-5, 5]$ using 100 contour lines and overlay on this plot the four gradient descent solution paths corresponding to the four different initializations. Refer to the following figure corresponding to $f(\mathbf{w}) = 0.2w_1^2 + w_2^2$ and a single initialization of $\mathbf{w}^0 = \begin{bmatrix} 4 & 4 \end{bmatrix}^T$ for reference purposes.

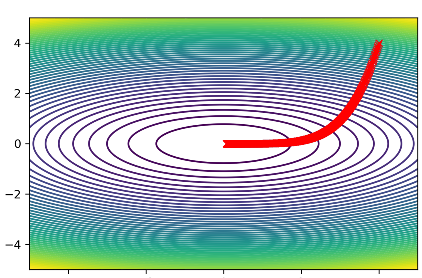

Figure 1: An illustrative contour plot overlayed with the gradient descent solution path.

Parameter Estimation

1. Use the `scipy.stats.multivariate_normal` module within the `scipy` package to generate N realizations of a Gaussian random vector $\mathbf{X} \in \mathbb{R}^5$ with mean vector $\boldsymbol{\mu} = \mathbb{E}[\mathbf{X}] = \begin{bmatrix} -1 & 0 & 4 & 1 & 0.5 \end{bmatrix}^T$ and covariance matrix $\mathbf{C} = 2\mathbf{I}$, where $\mathbf{I} \in \mathbb{R}^{5\times 5}$ denote the identity matrix. Here, we are interested in the following values of N: $N = 10^j, j = 1, \ldots, 6$.
   - Obtain an estimate $\widehat{\boldsymbol{\mu}}_N$ of the mean vector $\boldsymbol{\mu}$ of $\mathbf{X}$ for each value of N as follows: $\widehat{\boldsymbol{\mu}}_N = \frac{1}{N}\sum_{n=1}^{N} \mathbf{x}_n$, where $\mathbf{x}_n$ denotes the nth realization of $\mathbf{X}$.
   - Compute the instantaneous error between $\widehat{\boldsymbol{\mu}}_N$ and the actual mean as follows: $e_N = \|\widehat{\boldsymbol{\mu}}_N - \boldsymbol{\mu}\|_2^2$.
   - Provide a log-log plot of e_N as a function of N; comment on the relationship between e_N and N.