RRC Summer Sessions 2020 **(Due: 2nd June 2020)**

DL Assignment #1

Instructor: Shashank Srikanth

Submission Policy: Read all the instructions below carefully before you start working on the assignment, and before you make a submission.

- This is a theory based assignment with an optional coding question. There will be another coding assignment on implementing deep neural networks.
- Solve the questions on paper. Email the latex submission/photos of your answer sheets in a zipped folder to the instructor.
- Ensure that submitted assignment is your original work. Please do not copy any part from any source including your friends and seniors. No harm in using the internet as long as you understand the solution and attempted it on your own.

Problem 1: MLE & MAP

(a) Show that the MLE and the MAP distribution is the same in case of uniform prior.

(b) Find the parameters of a Gaussian distribution using MLE. Is the mean μ and sigma σ estimated using MLE unbiased?

Problem 2: Gradient Descent

(a) Write the Taylor series expansion of the cost function J in terms of its derivatives (eg. Jacobian and Hessian).

(b) Assuming strong convexity, show that the gradient descent will always decrease the function value at each step. [Use the equation in part (a) for the same]

(c) Determine the optimal learning rate at each gradient descent iteration assuming a second order approximation of the cost function.

(d) Derive the Newton's update rule for optimization of cost functions.

Problem 3: Linear and Logistic Regression

(a) It is often advised to do feature scaling before linear regression. Explain the intuition behind the same.

(b) Show that the cost function below for logistic regression is non convex.

$$\text{cost}\left(h_\theta\left(x^{(i)}\right), y^{(i)}\right) = \frac{1}{2}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2$$

(c) Show that the other formulation of logistic regression cost function shown in class is convex.

$$\begin{aligned} J(\theta) &= \frac{1}{m}\sum_{i=1}^{m} \text{cost}\left(h_\theta\left(x^{(i)}\right), y^{(i)}\right) \\ &= -\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)} \log h_\theta\left(x^{(i)}\right) + \left(1 - y^{(i)}\right) \log\left(1 - h_\theta\left(x^{(i)}\right)\right)\right] \end{aligned}$$

Problem 4: Vector and Tensor Derivatives

(a) Find $\frac{\partial y}{\partial x}$ and $\frac{\partial y}{\partial Q}$, if $y = x^T Q x$. It is given that $x \in \mathbb{R}^N$ and $Q \in \mathbb{R}^{NxN}$.

(b) Derive the closed form solution of linear regression with L2 Norm using the matrix derivatives method. [Hint: Write the cost function and take its derivative in a manner similar to the least squares solution]

(c) Derive the derivative of the softmax function with respect to its inputs.

Problem 5: Neural Networks and Backpropagation

(a) Neural networks can be used to represent simple logic gates such as binary AND & OR. Define a single layer neural network (its weights and activation function) to represent the above two functions (AND and OR). [Hint: ReLU or Signum function can be used as activation functions in this case]

(b) A single layer neural network cannot represent certain complex gates such as XOR and XNOR. We need a two layer neural network with non-linearities to define such functions. Define a 2 layer neural network with non linear activation functions like Signum/ReLU that can represent the XOR and XNOR gates.
* This can give you an intuition behind why we need more layers to approximate highly non-linear functions.

(c) Find the derivative of $f(x; W)$ with respect to x and w. The function is defined in slide 30 of the vector and tensor derivatives slide on GitHub.

(d) Solve the problem given in the 31st slide of the vector and tensor derivatives slide on GitHub.

(e) [Optional coding question] Read tutorials on how to define your own autograd functions in PyTorch. Implement the forward and backward pass for your own custom ReLU activation function and train a linear/logistic regression model on the same.

Problem 5: Convolutional Neural Networks

(a) If the previous layer has size JxK, and a filter of size MxN is applied with stride s and zero-padding of width P, what will be the size of the resulting convolutional layer?

(b) If max pooling with filter size F and stride s is applied to a layer of size JxK, what will be the size of the resulting (downsampled) layer?

(c) Can fully connected layers be represented using convolution layers itself? If yes, show how?

(d) [Additional Reading - (No need to attempt)]: Read the original ResNet paper by He et al. Explain the intuition behind the use of residual/skip connections.

(e) [Additional Reading - (No need to attempt)]: Read about one-shot learning for face recognition and other tasks. Explain the benefits of the triplet loss function used in such scenarios.