**Instructions:** You need not necessarily attempt the questions in a linear order :-)

1. **(1 Mark)** Prove Pythagoras' Theorem.

2. A model is said to be identifiable if a sufficiently large training set can rule out all but one setting of the model's parameters (in other words, you will be able to find a unique solution which minimizes a cost function). Models which have hidden variables (for example, deep neural networks containing hidden layers) are often not identifiable because we can obtain equivalent models (which give the same output) using a simple trick.
   (a) **(2 Marks)** Suggest such a method for deep neural networks containing hidden layers with sigmoid activations ?
   (b) **(1 Mark)** Given a network with $m$ hidden layers and $n$ sigmoid neurons per layer how many equivalent models will you get using the above method ?
   (c) **(2 Marks)** If the deep neural network contains relu activations then suggest a method which will give you infinite equivalent models (provided that the cost function depends only on the model's output)
   (d) **(1 Mark)** The above discussion suggests that the objective function of a deep neural network with hidden variables could have multiple local minima (multiple solutions). In other words the objective function is non-convex. Argue why this kind of non-convexity resulting from the discussion above is not a problem.

3. Consider an overcomplete autoencoder containing one hidden layer with logistic activations. We discussed that overcomplete autoencoders could suffer from the problem of overfitting. In particular, they can simply learn to copy the input to some units in the hidden layer and then copy these units to the output layer. Typically, adding some regularization solves the above problem.
   (a) **(1 Mark)** Justify why one would use an overcomplete autoencoder instead of an undercomplete autoencoder ?
   (b) **(1 Mark)** Typically, it is observed that using L2 regularization in this case helps in preventing overfitting. On further investigation, we find that the weights connecting the hidden layer to the output layer are actually very large in the absence of regularization. Comment on the corresponding observation for the weights connecting the input layer to the hidden layer. Would they be large or small ?
   (c) **(2 Marks)** Explain your answer.

4. We saw two kinds of oscillations in gradient descent based approaches. The first kind resulting from the aggressive nature of momentum and nag (because of which they end up taking a lot of u-turns). The second type of oscillations were seen in the stochastic versions of these algorithms because of greedy estimates. This question is about the second type of oscillations.

(a) **(1 Mark)** All other things remaining equal, would momentum based stochastic gradient descent have more or fewer oscillations than vanilla stochastic gradient descent.

(b) **(2 Marks)** Explain your answer.

5. We saw that any boolean function of $n$ variables can be represented by a neural network containing an input layer with $n$ neurons, one hidden layer with $2^n$ neurons and one output layer with 1 neuron.

   (a) **(2 Marks)** Suggest a scheme using which any function of $n$ variables can be represented by a neural network containing an input layer with $n$ neurons, one hidden layer with at most $2^{n-1}$ neurons, an optional second hidden layer with at most 1 neuron and one output layer with 1 neuron? (Hint: any boolean function can be represented as a sum of product of its variables)

   (b) **(1 Mark)** Using the above scheme, draw a neural network which can represent the XOR function of 3 variables using an input layer with 3 neurons, one hidden layer with at most 4 neurons and one output layer with 1 neuron?

6. **(1 Mark)** Prove that the solution to the following optimization problem is given by the dominant eigen vector of A:

$$\begin{aligned} \max_x \quad & x^\top A x \\ s.t. \quad & ||x||_2 = 1 \end{aligned}$$

7. **(2 Marks)** Consider the standard least-squares linear regression problem : $Ax = b$, where $x$ is an unknown which needs to be learned. The solution to this problem is given by $x = (A^\top A)^{-1}(A^\top b)$. We will assume that $A^\top A$ is invertible and $A$ is a full rank matrix. Computing the inverse of $A^\top A$ could still be expensive. Assume that you have access to a magic box which can compute all the eigen vectors and eigen values of a matrix in O(1) time (hence the name magic box). Using this magic box suggest a way of computing the product $(A^\top A)^{-1}A^\top$?

8. Stochastic Gradient Descent is agnostic to the importance of individual features of the input data. The update for each dimension uses the *same* learning rate. In situations where we have sparse data, we notice that the sparse features get fewer updates and if the feature turns out to be important to the task at hand, we might not be optimal w.r.t. the updates. AdaGrad solves this problem by letting us make dimension specific updates. Specifically, the update equation for AgaGrad is given by

$$\begin{aligned} v_t &= v_{t-1} + (\nabla w_t)^2 \\ u &= -\frac{\eta}{\sqrt{v_{t-1} + \epsilon}} \nabla w_t \\ w_t &= w_{t-1} + u \end{aligned}$$

where $w$ is one specific dimension and $v_t$ is the history of gradients w.r.t. $w$ and $\eta$ is some constant learning rate.

(a) **(1 Mark)** The denominator $v_{t-1} + \epsilon$ is a strictly non-decreasing function. This ends up killing the rate of update for frequent parameters. Modify the update equation for $v_t$ to circumvent this issue ?

(b) **(1 Mark)** The units of the update $u$ is inconsistent with the units of $w$. Explain why ? (you can ignore $\epsilon$ for now)

(c) **(2 Marks)** Suggest a modification to the update to make the dimensions consistent while remaining faithful to the original goal of adapting the learning rate for each dimension. (Hint: Think along the lines of momentum method)

9. We intend to solve the following equation

$$Ax = b$$

The solution $x^* = A^{-1}b$, can be potentially expensive to compute because of the inversion of the matrix $A$. We consider a method called Jacobi iterations and start by writing $A = D + E$ where $D$ is a diagonal matrix with entries from the diagonal of $A$ and $E$ is matrix of non-diagonal entries of $A$.

$$\begin{aligned} Dx &= -Ex + b \\ x &= -D^{-1}Ex + D^{-1}b \\ x &= Lx + m \quad \text{where} \quad L = -D^{-1}E \quad m = D^{-1}b \end{aligned}$$

Assume that if $\lambda_d$ is the dominant eigen value of $L$ then $|\lambda_d| < 1$.

(a) **(2 Marks)** Construct an iterative update procedure to solve the above problem. (Write an update equation for $x_i$ )

(b) **(1 Mark)** What is the condition for convergence ? (In terms of $x_i$, $x_{i+1}$)

(c) **(1 Mark)** Prove that the true solution $x^*$ is obtained on convergence.

(d) **(1 Mark)** From part (b) argue that the procedure is independent of the starting point $x_0$.

10. Batch normalization(BN) layer normalizes the inputs to zero mean and unit variance. A batch normalization layer takes in activations at a layer $l$ and maps them to normalized activations. These normalized activations form the input to the next layer. Consider a mini-batch $B$ with $m$ examples. Let us consider one such activation $x$ in the layer $l$ taking values $x_1, x_2, \ldots, x_m$. We normalize these to

$$\hat{x}_i = \frac{x_i - \mu_{\mathbf{B}}}{\sqrt{\sigma_{\mathbf{B}}{}^2 + \epsilon}}$$

where $\mu_{\mathbf{B}}$ and $\sigma_{\mathbf{B}}$ are the mean and standard deviation for batch $B$ and $\epsilon$ is some small constant (used for numerical stability). The corresponding output of the batch normalization layer $y_i$ for the $i$-th example is given by

$$y_i = \gamma \hat{x}_i + \beta$$

(a) **(1 Mark)** What are the parameter(s) of the batch normalization layer ?

(b) **(1 Mark)** Assuming that we have the gradients of the loss function w.r.t. $y_i$, *i.e.*, $\frac{\partial \mathscr{L}}{\partial y_i} \quad \forall i \in \{1, \ldots, m\}$ is known, derive the expression for the gradient of the loss function w.r.t. the parameter(s) of the batch normalization layer.

(c) **(3 Marks)** Derive the expression for the derivative of the loss function w.r.t. $x_i$.

11. Let $d_0, \ldots, d_{n-1}$ be the search directions used in Conjugate gradient descent. Recall that these search directions are $H$-orthogonal and hence are linearly independent. Also recall that $\alpha_{i-1}$ is the learning rate used in the update rule $u_{i+1} = u_i + \alpha_i d_i$ and $\nabla u_i = He_i$

(a) **(1 Mark)** Starting with $e_i = \sum_{j=0}^{i-1} \delta_j d_j$, prove that, $d_k^\top \nabla u_i = 0 \quad \forall k < i$.

(b) **(2 Marks)** Notice that the above condition is true irrespective of how we construct $d_i$'s as long as they are $H$-orthogonal. Now suppose we construct $d_i$'s using the following procedure

$$d_0 = \nabla u_0 \tag{1}$$

$$d_i = \nabla u_i - \sum_{k=0}^{i-1} \beta_{ik} d_k \quad \forall i > 0 \tag{2}$$

Show that, $\nabla u_k^\top \nabla u_i = 0 \quad \forall k \neq i$

(c) **(3 Marks)** Essentially what we've done above is used $\nabla u_0, \ldots, \nabla u_{n-1}$ as the basis for constructing $d_i$'s. Given the procedure for constructing search directions as given in part (b), prove that

$$\beta_{ij} = \frac{1}{\alpha_{i-1}} \frac{\nabla u_i^\top \nabla u_i}{d_{i-1}^\top H d_{i-1}} \quad i = j + 1$$

$$\beta_{ij} = 0 \quad i > j + 1$$

12. Consider the function $f(\theta) = f(x, y, z) = x^2 + y^2 + z^2 - 8$

(a) **(0.25 Mark)** Compute the gradient of the function at $\theta_0 = \{1, -1, 1\}$.

(b) **(0.25 Mark)** Compute the Hessian of the function at $\theta_0 = \{1, -1, 1\}$.

(c) **(0.5 Mark)** Starting from $\theta_0 = \{1, -1, 1\}$ apply one step of Newton's method to obtain $\theta_1$.

13. **(.100 Marks)** Starting from first principles can you prove Stein's Lemma ? ;-)