Question 1
Please give short answers (1-3 sentences) to the following questions.

(a) What is the *curse of dimensionality*?

Very large feature spaces (i.e., large number of features) for which we might not have enough data to adequately train a machine learning model.

(b) What is *overfitting* and how can we avoid it?

Overfitting is the risk of highly flexible (complicated) models.
We can avoid overfitting by obtaining more training samples, reducing the number of features, or performing regularization.

(c) It is often assumed that the magnitude (i.e., $|w_1|$, $|w_2|$, ..., , $|w_D|$) of the linear regression coefficients $\mathbf{w} = [w_1, \ldots, w_D]$, where $y = \mathbf{w}^T\mathbf{x}$, indicates the importance of the corresponding features. Describe a situation where this may not be true.

The input features might not have the same range, e.g., x_1 values are between $[0, 100]$, while x_2 values are between $[0, 1]$.

Question 2
Please select the correct answer(s) to the following questions. Multiple answers could be correct.

(a) Which are possible hyper-parameters?

A. The learning rate α of gradient descent

B. The regularization term λ in linear regression

C. The degree of polynomial in non-linear regression with a polynomial function

D. The weights $\mathbf{w}$ in non-linear regression

A, B, and C are the correct answers. The weight vector $\mathbf{w}$ is a parameter that is learned during training.

(b) Suppose we perform linear regression, but we do not assume that all features are equally predictive of the outcome of interest. Which of the following can we do to address this?

A. $l1$-regularization

B. $l2$-regularization

C. non-linear regression

A and B are the correct answers, since they will help minimize the effect of non-useful features.

(c) Which of the following machine learning methods always have one unique optimum, regardless of the data?

A. K-Nearest Neighbor

B. Linear perceptron

C. Linear regression

D. Non-linear regression using the non-linear space $\mathbf{x} \in \mathbb{R}^2 \rightarrow \mathbf{z}$, $\mathbf{z} = [x_1, x_2, x_1^2, x_2^2]$

C and D are the correct answers, since the Hessian of the optimization function (i.e., RSS error) is positive semi-definite.

(d) Which of the following statements is/are correct?

A. The gradient of a function $f(\mathbf{x})$ is the a vector pointing in the direction of the steepest slope of f at that point

B. The second-order derivative of the optimization function is used to determine if we have reached an optimal point

C. The gradient descent method can be used to maximize a function

D. Numerical search methods are useful only when we cannot find a closed-form solution to the optimization problem

A and C are correct. Regarding item B, the first-order derivative (i.e., gradient vector) determines whether we have reached an optimal point. Regarding item D, numerical methods can be used when we have a closed-form solution, but the latter might be computationally expensive (e.g., see computational cost of closed-form solution for linear regression).

(e) Adding more polynomial terms in a non-linear regression model

A. Increases the risk of overfitting

B. Yields reduced error on the training samples

C. Provides a more complicated model

D. Can be reliably learned if we use adequate number of training samples

All the above are correct.

Question 3
Assume a non-linear regression, whose output $y \in \mathbb{R}$ is predicted from the input $x \in \mathbb{R}$ as follows:

$$y = f(x) = bx^3 + c$$

The goal of the non-linear regression is to learn the weights $b, c \in \mathbb{R}$ from the training data $\mathcal{D} = \{(x_1, y_1), \ldots, (x_N, y_N)\}$.
Hint: For the following, you can use the formula: $(z_1 + z_2 + z_3)^2 = z_1^2 + z_2^2 + z_3^2 + 2z_1z_2 + 2z_2z_3 + 2z_1z_3$

(a) Write the residual sum of squares (RSS) error between the actual and predicted outcomes.

$RSS(b, c) = \sum_{n=1}^{N} (y_n - f(x_n))^2 = \sum_{n=1}^{N} \left(y_n - bx_n^3 - c\right)^2$

(b) Assuming that c is constant, derive the gradient descent formula for updating the weight $b \in \mathbb{R}$ so that the RRS error (as obtained from question a) is minimized with respect to b. Give the batch update formula and the stochastic update for a random sample (x_m, y_m). Set up your formulas so that the learning rate parameter is $\nu(k) > 0$, in which k is the iteration index.

$\frac{\vartheta RSS(b,c)}{\vartheta b} = 2\sum_{n=1}^{N} \left(y_n - bx_n^3 - c\right)(-x_n^3) = -2\sum_{n=1}^{N} x_n^3 \left(y_n - bx_n^3 - c\right)$

Batch: $b(k+1) := b(k) - \nu(k)\frac{\vartheta RSS(b,c)}{\vartheta b} \Rightarrow b(k+1) := b(k) + \nu(k)\sum_{n=1}^{N} x_n^3 \left(y_n - b(k) \cdot x_n^3 - c\right)$

Stochastic: $b(k+1) := b(k) + \nu(k) \cdot x_m^3 \left(y_m - b(k) \cdot x_m^3 - c\right)$

Note: In the above, we have indicated the value of b from the previous iteration as $b(k)$ (instead of writing only b).

(c) Assuming that b is constant, derive the gradient descent formula for updating the weight $c \in \mathbb{R}$ so that the RRS error (as obtained from question a) is minimized with respect to c. Give the batch update formula and the mini-batch update for a random subsample of the training data $\mathcal{D}_s \in \mathcal{D}$. Set up your formulas so that the learning rate parameter is $\nu(k) > 0$, in which k is the iteration index.

$\frac{\vartheta RSS(b,c)}{\vartheta c} = 2\sum_{n=1}^{N} \left(y_n - bx_n^3 - c\right)(-1) = -2\sum_{n=1}^{N} \left(y_n - bx_n^3 - c\right)$

Batch: $c(k+1) := c(k) - \nu(k)\frac{\vartheta RSS(b,c)}{\vartheta c} \Rightarrow c(k+1) := c(k) + \nu(k)\sum_{n=1}^{N} \left(y_n - b \cdot x_n^3 - c(k)\right)$

Mini-batch: $c(k+1) := c(k) + \nu(k)\sum_{(x_m,y_m)\in\mathcal{D}_s} \left(y_m - b \cdot x_m^3 - c(k)\right)$

(d) We now apply $l2$-norm regularization for the aforementioned non-linear regression in terms of both weights $b, c \in \mathbb{R}$. Write the expression of the evaluation criterion for the non-linear regression with regularization, including the RSS error and regularization term, assuming $\lambda > 0$ as the model complexity for both b and c.

$RSS(b, c) = \sum_{n=1}^{N} \left(y_n - bx_n^3 - c\right)^2 + \lambda b^2 + \lambda c^2$

(e) Derive the gradient descent formula for jointly updating weights $b, c \in \mathbb{R}$ so that the optimization criterion of the regularized non-linear regression (as obtained from question d) is

minimized jointly with respect to b and c. Give the batch update formula using the learning rate parameter $\nu(k) > 0$, in which k is the iteration index.
Hint: Assume a weight vector $\mathbf{w} = [b, c]^T \in \mathbb{R}^2$.

$\frac{\vartheta RSS(b,c)}{\vartheta b} = -2\sum_{n=1}^{N} x_n^3 \left(y_n - bx_n^3 - c\right) + 2\lambda b$

$\frac{\vartheta RSS(b,c)}{\vartheta c} = -2\sum_{n=1}^{N} \left(y_n - bx_n^3 - c\right) + 2\lambda c$

$$\begin{bmatrix} b(k+1) \\ c(k+1) \end{bmatrix} = \begin{bmatrix} b(k) \\ c(k) \end{bmatrix} - \nu(k) \begin{bmatrix} -2\sum_{n=1}^{N} x_n^3 \left(y_n - b(k)x_n^3 - c(k)\right) + 2\lambda b(k) \\ -2\sum_{n=1}^{N} \left(y_n - b(k)x_n^3 - c(k)\right) + 2\lambda c(k) \end{bmatrix}$$

Question 4
Assume that you have a set of training and test data, $\mathbf{X^{train}} \in \mathbb{R}^{D\times N_1}$ and $\mathbf{X^{test}} \in \mathbb{R}^{D\times N_2}$, respectively, with corresponding label vectors $\mathbf{y^{train}} \in \mathbb{R}^{N_1}$ and $\mathbf{y^{test}} \in \mathbb{R}^{N_2}$. You would like to perform classification using logistic regression and use cross-validation on the training set to determine the best value of the learning rate $\alpha = \{0.001, 0.01, 0.1\}$ for the gradient descent optimization of logistic regression. Please provide a basic pseudocode to do this. In the pseudocode, please also include the last evaluation step after having identified the best learning rate α based on the test data.
Hint: You can assume a function *pred*=LR(X,y,Z,α), which provides a decision for test samples Z using training data X, training labels y and gradient descent learning rate α, and a function *acc*=ComputeAcc(*pred*,*lab*), which computes the classification accuracy between predicted class *pred* and actual labels *lab*.

Split $\left\{\mathbf{X^{train}}, \mathbf{y^{train}}\right\}$ into S equal parts $\left\{\mathbf{X_S^{train}}, \mathbf{y_S^{train}}\right\}_{s=1}^{S}$

$A = [0.001,\ 0.01,\ 0.1]$

$CVAcc = []$ (vector to store the accuracies from the cross-validation)

for $\alpha \in A$:
    $E = []$ (vector to store accuracies within each fold)
    for $s \in 1, \ldots, S$:
        pred=LR($\mathbf{X^{train}} \setminus \mathbf{X_S^{train}}$, $\mathbf{y^{train}} \setminus \mathbf{y_S^{train}}$, $\mathbf{X_S^{train}}$, α)
        $E(s)$=ComputeAcc(pred,$\mathbf{y_S^{train}}$)
    $CVAcc = [CVAcc, \frac{1}{S}\left(E(1) + \ldots E(S)\right)]$ (storing average accuracy from all folds for α)

$\alpha^{best} = A(arg\max(CVAcc))$ (obtaining the best learning rate)

pred=LR($\mathbf{X^{train}}$,$\mathbf{y^{train}}$,$\mathbf{X^{test}}$,α^{best})

acc=ComputeAcc(pred,$\mathbf{y^{test}}$)