1. (a) Formulation of Linear regression as maximization of a likelihood function :

In linear regression we are seeking those values of theta where P(data | theta) is maximum.
For computational purposes we are considering negative log-likelyhood function so, Now we will minimize this function.

$$NLL(\theta) = -\sum_{i=1}^{N} \log p(y_i | x_i, \theta)$$

$$= -\sum_{i=1}^{N} \log \left[\left(\frac{1}{2\pi\sigma^2} \right)^{1/2} \exp\left(\frac{-1}{2\sigma^2} (y_i - \beta^T x_i)^2 \right) \right]$$

$$= -\sum_{i=1}^{N} \frac{1}{2} \log\left(\frac{1}{2\pi\sigma^2} \right) - \frac{1}{2\sigma^2} \sum_{i=1}^{N} (y_i - \beta^T x_i)^2$$

$$= -\frac{N}{2} \log\left(\frac{1}{2\pi\sigma^2} \right) - \frac{1}{2\sigma^2} RSS(\beta)$$

Where RSS is the residual sum of squares. Now we have to minimize the RSS term. If we go on differentiating this RSS w.r.t. β and equal it to zero, we get that

$$\hat{\beta}_{OLS} = (X^T X)^{-1} X^T Y$$

This matrix gives the same parameters that we have got through gradient descent or any other optimization technique.

(b) We can make a linear regression model for classification, treating 0 and 1 as outcomes. And this model will fit the best hyperplane that minimizes the distances between the points.

But this model is not a good choice to choose as there is no meaningful thresold at which we can distinguish one class from the other. More problem arises when we have multiple classes as output.

2. In stochastic gradient descent, we take a minibatch for one step of tuning parameters and if we are taking momentum as well then we update our parameters as follows

$$V_{dw} = \beta V_{dw} + (1-\beta) dw$$
$$V_{db} = \beta V_{db} + (1-\beta) db$$

where initial valre of V_{dw} and V_{db} is 0.

$$w = w - \text{learning\_rate} \times V_{dw}$$
$$b = b - \text{learning\_rate} \times V_{db}$$

Intuition : For intuition, suppose that w and b are two parameters of the network and cosider that if we go along w direction then we are going towards local optimum and b direction is perpendicular to that path.

Now we will consider dw and db collectively as acceleration. So, at any stage acceleration can have any direction but the net momentum towards the local optimum is always will be there but what we are doing in this technique is that we are cancelling out momentum in perpendicular direction to this path by taking exponential average of all the gradients (dw, db) till that stage. Hence our path will have less movement in unwanted direction and we will achieve local optimum earlier than the gradient descent method.

3. (a) In batch gradient descent, if we have a lot of data, then one step of updating parameters takes lot of time because we require a lot of computations even in vector implementation.

If we take 1 datapoint at a time, then we will not profit of vector implementation because then we are only applying only loopes. Hence to make our process fast and effective we take minibatch size in between 1 and total batch size.

Efficiency and robustness comparison
This method usually add noise to the gradient term and makes a zig-zag path towards the local optimum with a speedy convergence.
But in deep-neural network, we require ~~a~~ ~~do~~ many iterations so, overall this method is time efficient than the batch gradient descent.

(b) If we initialize all the weights to the same valve, then all the hidden unit in a hidden layer are symmetric to each other and will calculate the same valve after each update step. Now at each layer, our network will work as if it has one neuron in it So, Our network will not be able detect all complex features of the input and accuracy will be very low.

(c) Regularization minimizes overfitting by injecting weight term in the loss so, for reducing loss, now we have small weights compared to that without regularization. And If we have smaller weighs, then our model will be more smoother so, it will change it's shape slowly as ~~[illegible]~~ input changes.

Other ways to reduce over-fitting :
Data Augmentation :- In this method we manipulate our data and add it to the input. For ~~[illegible]~~ example we can crop the image, flip the image. This results in more data and has more variety in it.

Early stopping : While ~~training~~ validating the data we get a point after which loss increases so, we take this point as our model for testing.

(d) In batch normalization, we normalize each layer of our network before applying activation to it. By this we achieve time-efficiency in training. This method also has slight regularization effect. By this, we have small inputs to each layer so, it increases size of gradient descent. Because it centers data around 0, then we don't have much deviation when we are taking a different data, and this means that we do not have to train our model for different data or we can say that variation in these models after training different data will be less.

4. $f = 3$ No. of parameters $= ((3 \times 3) \times 3) \times 8$

$il = 3$ $= 216$

$ol = 8$

$p = 1$ $ip \sim N1 \times N2 \times 3$

$s = 2$ $op \sim \left[\frac{N1+1}{2}\right] \times \left[\frac{N2+1}{2}\right] \times 8$

Where $[\cdot]$ is greatest integer function.