**1. Which Linear Regression algorithm can we use if we have a training set with millions of features?**

The million degrees of freedom, which also means the number of independent variables in a set, leads to the development of a complex model. A simple linear regression model used to fit data with millions of degrees of freedom would perform poorly as it will underfit. In such cases, the complexity of the simple linear regression model has to be increased to generalize well on the subject data, hence the need for regularization. Regularized linear regression models like Ridge Regression or Lasso Regression are commonly used. These methods introduce regularization terms that help control the model complexity and prevent overfitting. Ridge Regression, in particular, is often favoured when dealing with high-dimensional datasets as it can handle a large number of features effectively. Also, a polynomial regression model with an optimal degree of freedom (Via the elbow method or cross-validation ) can be used.

**2. Can the Gradient Descent Algorithm get stuck in a local minimum when training a linear regression model?**

No, the Gradient Descent Algorithm used for training a linear regression model does not get stuck in a local minimum. Gradient descent produces a convex-shaped graph (cost function for linear regression is a convex function) which only has one global optimum. Therefore, it cannot get stuck in a local minimum. Gradient descent iteratively adjusts the model parameters in the direction of the steepest descent of the cost function, eventually converging to the global minimum. It is important to keep in mind that some gradient descent variants, such as stochastic gradient descent (SGD), might exhibit fluctuations and slow-paced convergence because randomization is introduced by utilizing subsets of the training data at each iteration. Randomness has been integrated into stochastic gradient descent and mini-batch gradient descent. This indicates that they can converge close to the global optimum, but they generally don't converge. The learning rate hyperparameter can be progressively reduced to aid in their convergence.

**3. Do all Gradient Descent Algorithms lead to the same model if they are running for the same no of epochs?**

While all gradient descent algorithms seek to minimize the cost function, if they are run for the same number of epochs, the final models they converge to could not be the same. This variance can be brought about by differences in learning rates, parameter initialization, or the use of various gradient descent variants (such as batch gradient descent and stochastic gradient descent). These variables may have an impact on the optimization process and may result in somewhat different models.

**4. If you are doing a batch gradient descent and you are monitoring the validation error at every epoch. If the validation error is constantly increasing what can be the problem? How to fix that?**

This issue can be addressed by taking the following steps:

* Decrease the learning rate: The learning rate determines the step size taken during each parameter update. If the learning rate is too high, it can lead to overshooting the optimal solution. By reducing the learning rate, you can help the model converge to a better solution.
* Regularization: Introduce regularization techniques like Ridge Regression or Lasso Regression, which add a penalty term to the cost function. Regularization discourages the model from relying too heavily on any particular feature and helps prevent overfitting.
* Feature selection: If the validation error is increasing, it may indicate that some of the features are not contributing meaningfully to the model's performance. Consider performing feature selection techniques to identify and remove irrelevant or redundant features, which can help improve the model's generalization ability.
* Increase training data: If possible, try to gather more training data to provide the model with a larger and more diverse set of examples to learn from. This can help the model better capture the underlying patterns in the data and reduce overfitting.