To overcome the problem of getting stuck in local minima in machine/deep learning optimization, there are several techniques you can employ. One commonly used approach is to use advanced optimization algorithms that can help the model escape local minima and find better global minima.

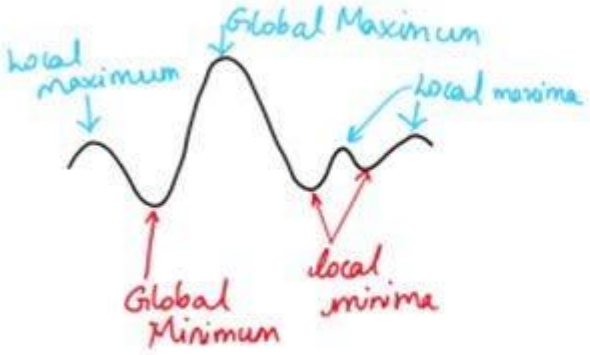


Figure (a)

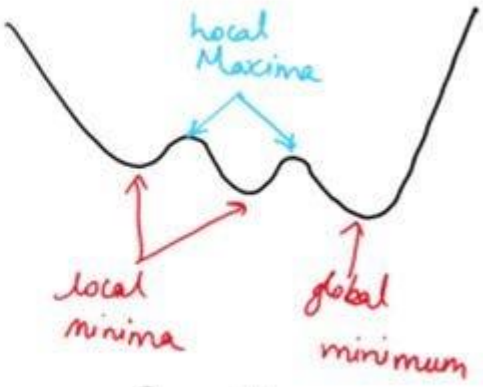
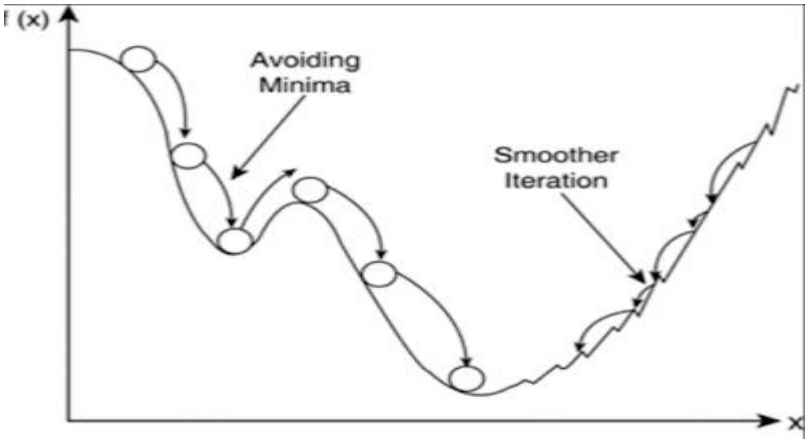
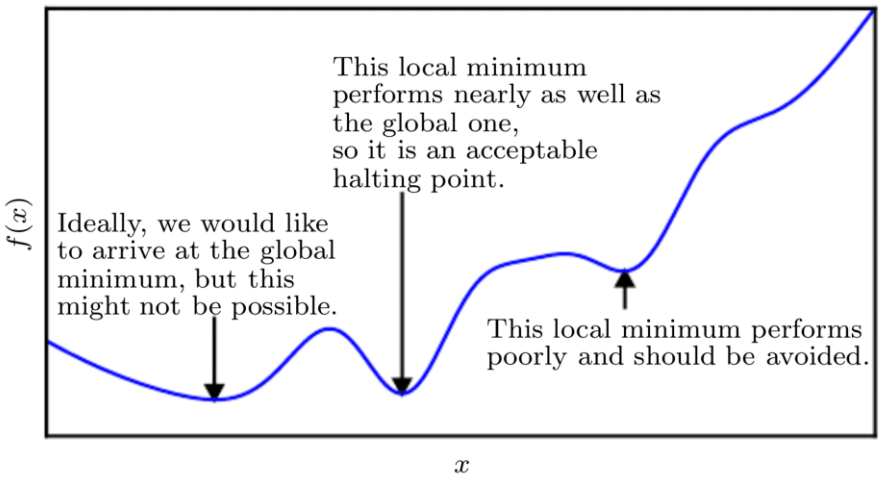


Figure (b)





Two techniques that are often employed are:

1. **Momentum Optimization**

Momentum optimization is a technique commonly used in machine learning to accelerate the convergence of gradient-based optimization algorithms. It helps overcome local minima and navigate through flat regions by accumulating a fraction of the past gradients' momentum. Mathematically, the momentum term is computed as follows:

```
velocity = beta * velocity - learning_rate * gradient
parameters += velocity
```

Here, the 'velocity' variable represents the accumulated momentum, 'beta' controls the contribution of past velocities, 'learning_rate' determines the

step size, and 'gradient' is the current gradient. (*I explained gradient descent and learning rate in one of my articles here*)

The momentum term acts as a weighted average of past gradients, allowing the optimization algorithm to have a memory of the direction it has been moving in previous steps. The effect of momentum is to increase the step size when the gradients consistently point in the same direction and decrease it when there is a high variance in the gradients' direction.

Let's consider an example to understand momentum optimization better. Suppose we have a simple optimization problem with a convex loss function that we want to minimize. We'll use a 2D surface plot to visualize the loss landscape.

Stochastic Gradient Descent

SGDR is a technique that periodically restarts the learning rate to help the model explore different areas of the optimization landscape. It updates the model parameters using small random subsets of the training data, which makes it computationally efficient compared to batch gradient descent. It involves cyclically varying the learning rate between low and high bounds. This cyclic learning rate schedule allows the model to jump out of shallow local minima. Mathematically, the learning rate is updated as follows:

```
parameters = parameters - learning_rate * gradient
```