Optimization:
To do gradient descent, PyTorch provides optimization packages. This package contains standard optimizers.

PyTorch has a torch.optim package that is used to implement various optimization algorithms. Most common optimizers used today like SGD, Adam, ReLu etc are already available within this package. It also has the robustness to add more sophisticated optimizers as per your need.

In order to use torch.optim we will have to construct an optimizer object which will store the current state and will update the parameters of the optimizers based on the computed gradients.

In order the initialize a optimizer this is the syntax to be followed -

CODE IN GOOGLE COLAB

Optimization algorithms that are already available in the torch.optim package are:
Adadelta - https://arxiv.org/abs/1212.5701

Adagrad - http://jmlr.org/papers/v12/duchi11a.html

Adam - https://arxiv.org/abs/1412.6980

Adamax - https://arxiv.org/abs/1412.6980

ASGD - http://dl.acm.org/citation.cfm?id=131098

LBFGS -

RMSprop - Proposed by G. Hinton in his course.
The centered version first appears in Generating Sequences With Recurrent Neural Networks.

SGD - http://www.cs.toronto.edu/~hinton/absps/momentum.pdf

These algorithms can be used with just one line of code. The syntax for using these algorithms and their parameter definition can be found in PyTorch official documentation here - https://pytorch.org/docs/stable/optim.html#algorithms

Adjusting learning_rate:
What is learning rate and why it needs to be adjusted?

PyTorch has a very good documentation explaining how to adjust the learning rate in PyTorch. The documentation can be found here -
https://pytorch.org/docs/stable/optim.html#how-to-adjust-learning-rate