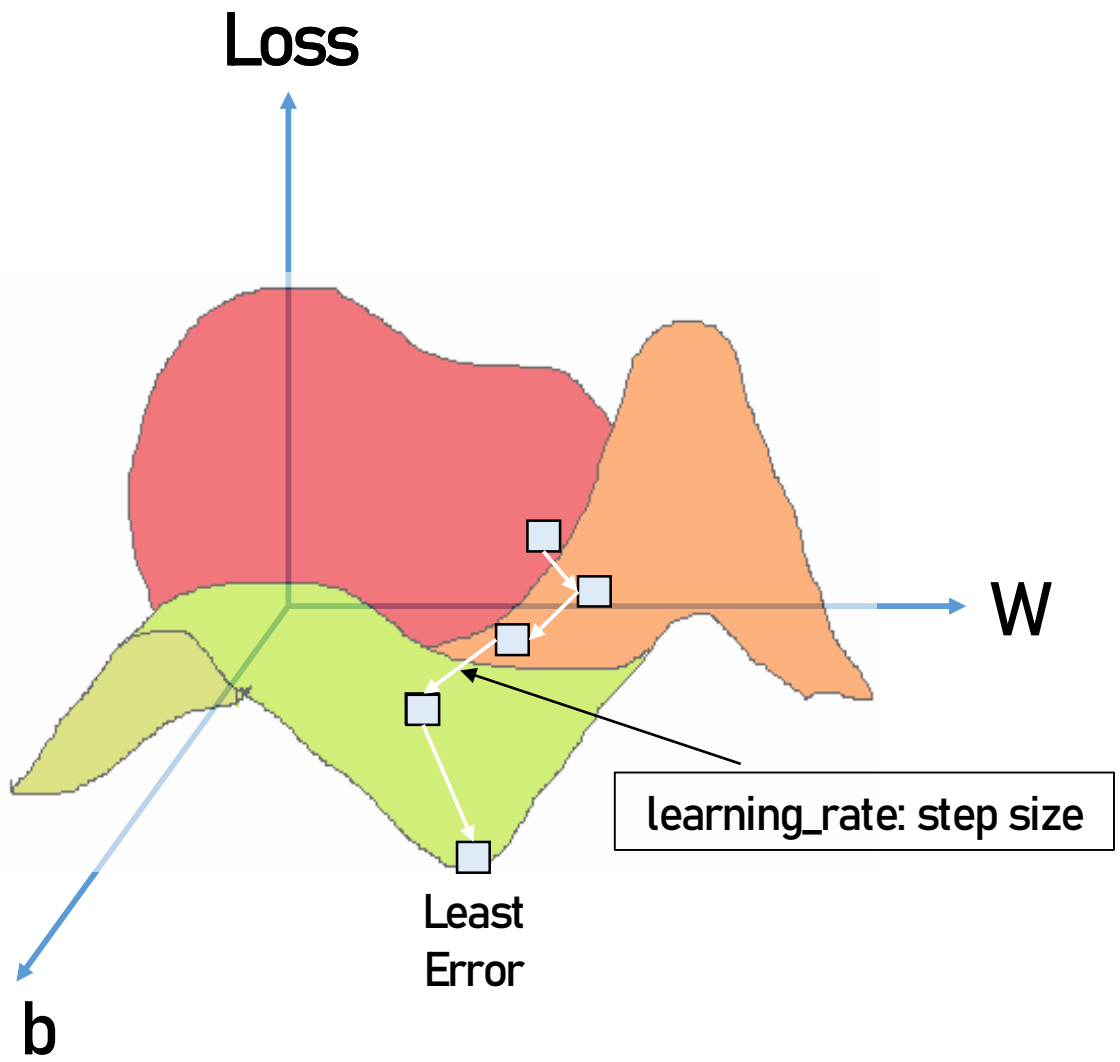
Optimizers

Training Process

- Initialize all the weights in the network before training
- Iterate over the dataset and pass one row after another through the network
- After each row is passed, do the following:
  - Calculate the loss
  - Change weights & biases to minimize the loss
  - The process of calculating gradients which are used to change the weights is called "back-propagation"
- Iterate through the dataset "n" number of times
- "n" is called the number of epochs

Visualizing Gradient Descent



$$Parameters^{t+1} = Parameters^{t} - learning\_rate * Gradient(\Theta^{t})$$

Change each parameter value by deducting the respective gradient multiplied by the learning rate

Till now we have manually used the gradients change parameter values

But in a real world scenario that is impossible since the neural networks might have millions of parameters

Optimizer does this job!!

✓ Construct an object of Optimizer

- ✓ Construct an object of Optimizer
- ✓ Perform calculations & call backward()

- ✓ Construct an object of Optimizer
- ✓ Perform calculations & call backward()
- ✓ Perform optimizer.step()

Optimizers

torch.optim.SGD

torch.optim.Adam

torch.optim.Adagrad

$$Parameters^{t+1} = Parameters^{t} - learning\_rate * Gradient(\Theta^{t})$$

SGD Optimizer

Change each parameter value by deducting the respective gradient multiplied by the learning rate

Optimizers

torch.optim.SGD

torch.optim.Adam

torch.optim.Adagrad