

CS167: Machine Learning

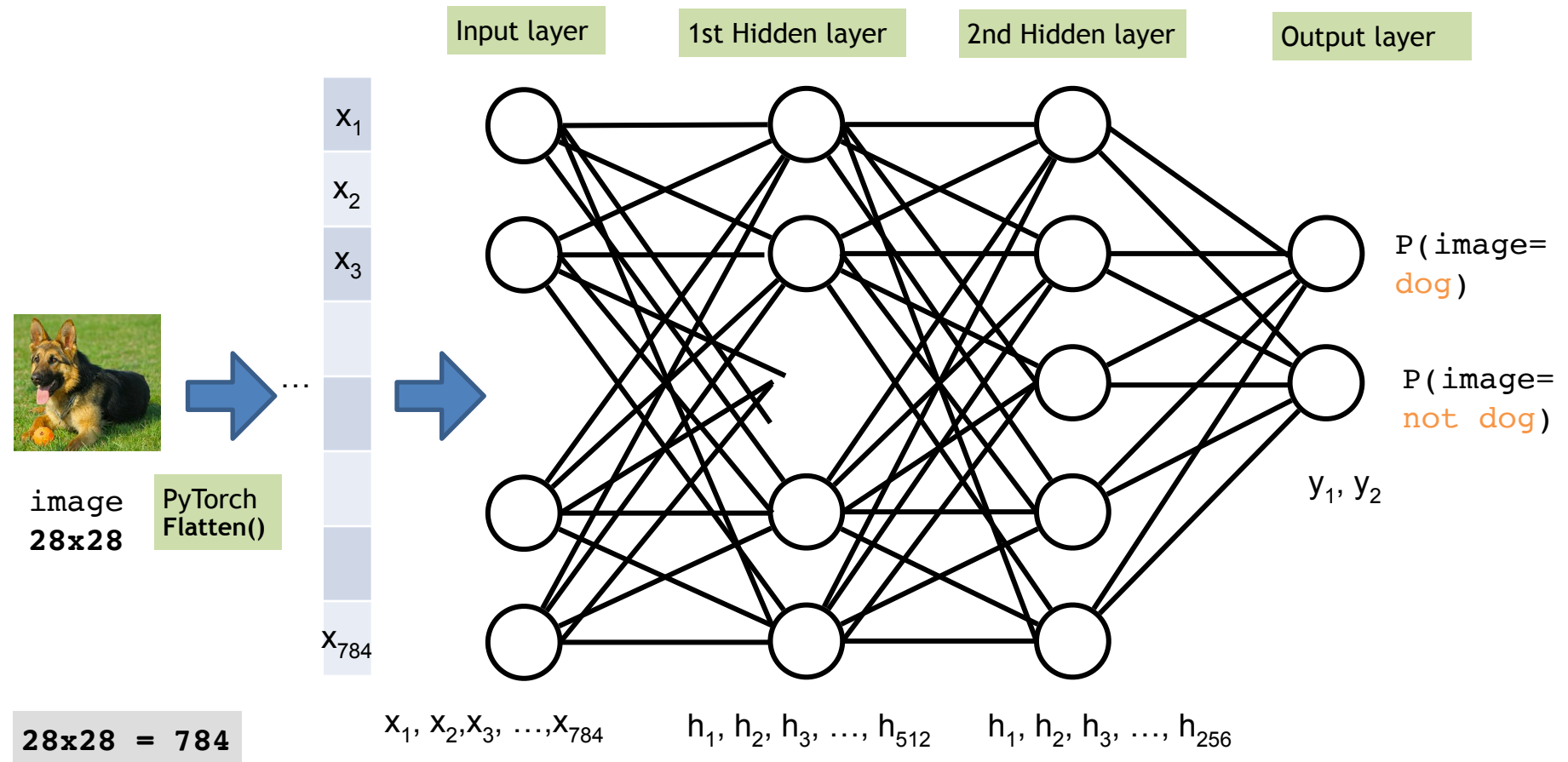
MLP Training
Convolutional Neural Network (CNN)

Monday, November 18th, 2024



Recap: Multilayer Perceptron (MLP)

- A multilayer perceptron is the simplest type of neural network. It consists of perceptrons (aka nodes, neurons) arranged in layers



Recap: List of PyTorch Functions We Need

- [nn.Linear\(\)](#)
creates the dense connections between two adjacent layers (*left layer* and *right layer*)
just provide **#neurons_left_layer** and **#neurons_right_layer**
 - [nn.ReLU\(\)](#)
 - [nn.Softmax\(\)](#)
 - [nn.flatten\(\)](#)
 - [nn.Sequential\(\)](#)
 - [nn.CrossEntropyLoss\(\)](#)
 - [torch.optim.SGD](#)
-
- Let's jump into the notebook for a detailed discussion.

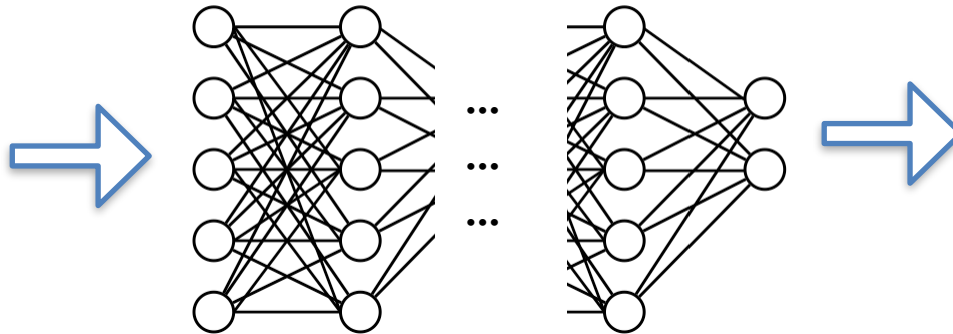
Today's Agenda

- Activity Poll on MLP
- Review: Gradient Descent and Stochastic Gradient Descent (SGD)
- Modular code for MLP training

Poll: MLP Summary

- Finish the MLP poll below:

<https://forms.gle/HDRw31snHAnLaQb17>



Today's Agenda

- Activity Poll on MLP
- Modular code for MLP training
- Review: Gradient Descent and Stochastic Gradient Descent (SGD)

Modular Code Multilayer Perceptron using MLP

A multilayer perceptron is the simplest type of neural network. It consists of perceptrons (aka nodes, neurons) arranged in layers. Create a network class with two methods:

- `init()`
- `forward()`

```
▶ import torch
  from torch import nn

# You can give any name to your new network, e.g., SimpleMLP.
# However, you have to mandatorily inherit from nn.Module to
# create your own network class. That way, you can access a lot of
# useful methods and attributes from the parent class nn.Module

class SimpleMLP(nn.Module):
    def __init__(self):
        super().__init__()
        # your network layer construction should take place here
        # ...
        # ...

    def forward(self, x):
        # your code for MLP forward pass should take place here
        # ...
        # ...
        return x
```

Dive into the Colab demo!

Today's Agenda

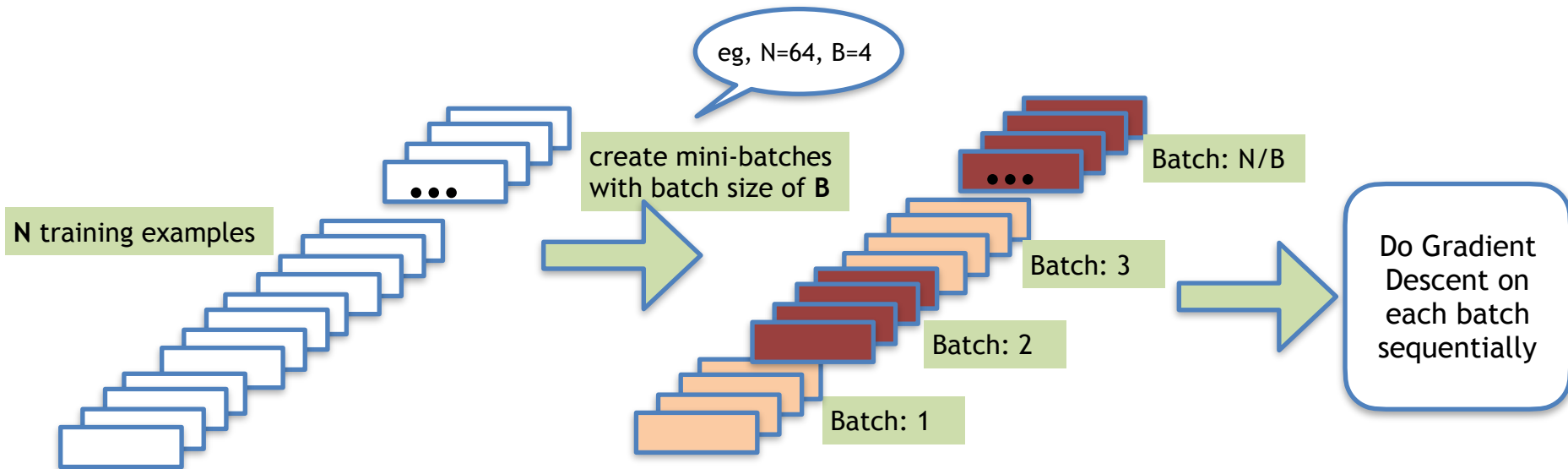
- Activity Poll on MLP
- Modular code for MLP training
- Review: Gradient Descent and Stochastic Gradient Descent (SGD)

Stochastic Gradient Descent (SGD)

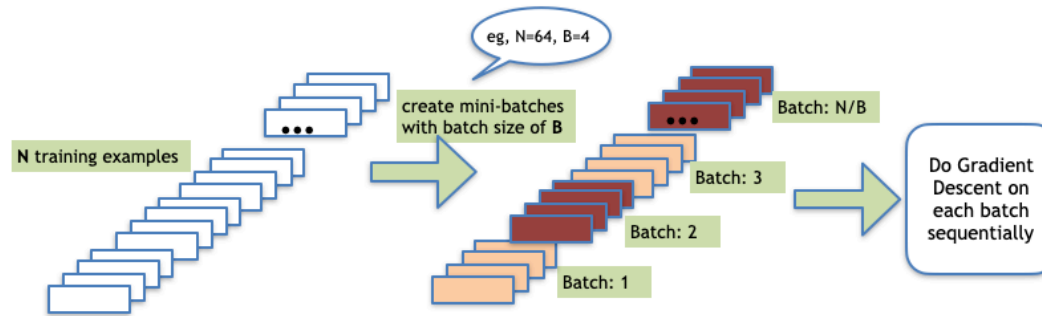
- Keep doing the **Gradient Descent**, but instead of using all the training samples, *use small subset of training samples* picked randomly when computing the **gradient vector**
 - divide the entire training data into **mini batches**
 - calculate the **gradient vector** based on that batch $\nabla E(\mathbf{w})$
 - adjust (or update) the values of the weights based on the **gradient vector** to that batch

$$\mathbf{w}^{\text{new}} = \mathbf{w}^{\text{old}} - \eta \nabla E(\mathbf{w})$$

Stochastic Gradient Descent (SGD)



In PyTorch: Stochastic Gradient Descent (SGD)



PyTorch Implementation

```
learning_rate = 1e-3
batch_size    = 4 # If the total sample size is 64, setting batch_size=4 will divide the data into 64/4=16 mini-batches of tensors
epochs       = 10
# let's use SGD optimization algorithm for training our model
optimizer    = torch.optim.SGD(mlp_model.parameters(), lr=learning_rate)
```

Optimization, as we have discussed earlier, is a process of adjusting model parameters to reduce model error in each training step. PyTorch provides a selection of optimization algorithms in the [torch.optim](#) package. Some of them are as follows:

- [torch.optim.SGD](#)
- [torch.optim.Adam](#)
- [torch.optim.RMSprop](#)

In addition to selecting the optimizer, we can also select the hyperparameters which are referred to as adjustable parameters crucial for controlling the model optimization process. You can influence the training and convergence of the model by tweaking these hyperparameters:

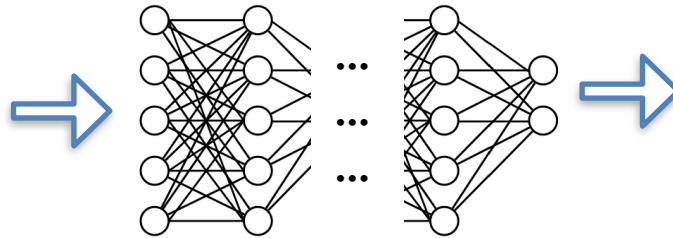
- **epochs:** denotes the number of iterations over the dataset
- **batch size:** represents the quantity of data samples in each iteration propagated through the network before updating the parameters
- **learning rate:** determines the extent of parameter updates made at each batch/epoch

Today's Agenda

- Activity Poll on MLP
- Modular code for MLP training
- Review: Gradient Descent and Stochastic Gradient Descent (SGD)
- **Convolutional Neural Network (CNN): another type of neural network**

What's Next?

- A **multilayer perceptron (MLP)** is the simplest type of neural network. It consists of perceptrons (aka nodes, neurons) arranged in layers



- A **multilayer perceptron (MLP)** is just the tip of the iceberg; plenty of other neural network variants exist.