

TTIC 31230, Fundamentals of Deep Learning

David McAllester, Autumn 2020

Deep Learning Frameworks

Deep Learning Frameworks

A framework provides a high level language for writing models $P_{\Phi}(y|x)$.

A framework compiles a model into an optimization algorithm.

$$\Phi^* \approx \underset{\Phi}{\operatorname{argmin}} E_{(x,y) \sim \text{Train}} - \ln P_{\Phi}(y|x)$$

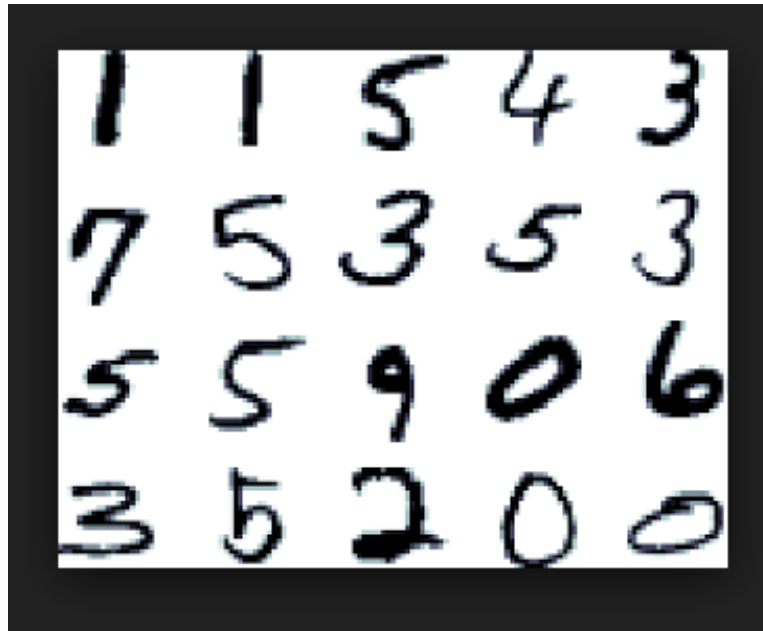
A framework also typically provides support for managing large training sets and pre-trained model parameter values (also called “models”).

Some Frameworks

- PyTorch
- Tensorflow
- Keras
- Microsoft Cognitive Toolkit
- Chainer
- \vdots
- EDF (Educational Framework in Python/NumPy used for the first problem set in this class).

An Example

We consider the problem of taking an input x (such as an image of a hand written digit) and classifying it into some small number of classes (such as the digits 0 through 9) using a multi layer perceptron (MLP).



Multiclass Classification

Assume a population distribution on pairs (x, y) for $x \in \mathbb{R}^d$ and $y \in \{y_1, \dots, y_k\}$.

For MNIST x is a 28×28 image which we take to be a 784 dimensional vector giving $x \in \mathbb{R}^{784}$.

For MNIST $k = 10$.

Let Train be a sample $(x_0, y_0), \dots, (x_{N-1}, y_{N-1})$ drawn IID from the population.

A Multi Layer Perceptron (MLP)

$$\begin{aligned}\mathbf{h} &= \sigma \left(W^0 \mathbf{x} - b^0 \right) \\ \mathbf{s} &= \sigma \left(W^1 \mathbf{h} - b^1 \right) \\ P_{\Phi}[\hat{y}] &= \underset{\hat{y}}{\text{softmax}} \mathbf{s}[\hat{y}]\end{aligned}$$

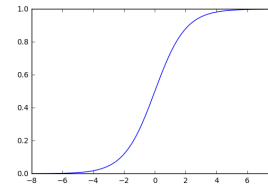
W^1 and W^2 are matrices. b_1 and b_2 are vectors.

σ is a scalar-to-scalar activation function applied to each component of a vector.

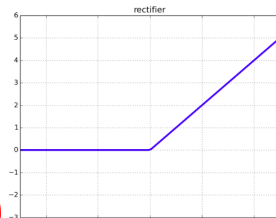
Activation Functions

An activation function $\sigma : \mathbb{R} \rightarrow \mathbb{R}$ (scalar-to-scalar) is applied to each component of a vector.

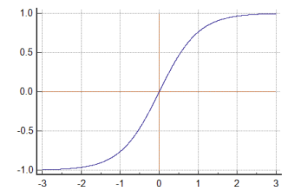
sigmoid: $\sigma(u) = \frac{1}{1+e^{-u}}$



other common activation functions are



$\text{ReLU}(u) = \max(0, u)$, $\tanh(u) = 2\sigma(2u) - 1$



The Framework Source Code

The source code is a sequence of assignment statements taking as input a training point, typically $\langle x, y \rangle$, and outputs a loss value \mathcal{L} , typically $-\ln P_{\Phi}(y|x)$.

$$\textcolor{red}{h} = \sigma \left(W^0 \textcolor{red}{x} - b^0 \right)$$

$$\textcolor{red}{s} = \sigma \left(W^1 \textcolor{red}{h} - b^1 \right)$$

$$\textcolor{red}{P}[\hat{y}] = \underset{\hat{y}}{\text{softmax}} \textcolor{red}{s}[\hat{y}]$$

$$\mathcal{L} = -\ln P[y]$$

Source Code

$$\textcolor{red}{h} = \sigma \left(W^0 \textcolor{red}{x} - b^0 \right)$$

$$\textcolor{red}{s} = \sigma \left(W^1 \textcolor{red}{h} - b^1 \right)$$

$$\textcolor{red}{P}[\hat{y}] = \underset{\hat{y}}{\text{softmax}} \textcolor{red}{s}[\hat{y}]$$

$$\mathcal{L} = -\ln P[y]$$

The source code is sometimes called a **computational graph**.
I prefer to call it the source code.

Source Code

$$\textcolor{red}{h} = \sigma \left(W^0 \textcolor{red}{x} - b^0 \right)$$

$$\textcolor{red}{s} = \sigma \left(W^1 \textcolor{red}{h} - b^1 \right)$$

$$\textcolor{red}{P}_\Phi[\hat{y}] = \underset{\hat{y}}{\text{softmax}} \textcolor{red}{s}[\hat{y}]$$

$$\mathcal{L} = -\ln P[y]$$

The framework automatically computes $\nabla_\Phi \mathcal{L}_\Phi(\langle x, y \rangle)$ where $\Phi = (W^0, b^0, W^1, b^1)$.

Frameworks Automate Stochastic Gradient Descent (SGD)

$$\Phi^* = \underset{\Phi}{\operatorname{argmin}} E_{z \sim \text{Train}} \mathcal{L}_{\Phi}(z)$$

1. Randomly Initialize Φ (initialization is important and must be done with care).
2. Repeat until “converged”:
 - draw $z \sim \text{Train}$ at random.
 - $\Phi \leftarrow \Phi - \eta \nabla_{\Phi} \mathcal{L}_{\Phi}(z)$

Epochs

In practice we cycle through the training data visiting each training pair once.

One pass through the training data is called an **Epoch**.

Summary

A framework provides a high level language for defining $\mathcal{L}_\Phi(z)$.

The framework compiles the source code for $\mathcal{L}_\Phi(z)$ into an optimization algorithm.

$$\Phi^* = \underset{\Phi}{\operatorname{argmin}} E_{z \sim \text{Train}} \mathcal{L}_\Phi(z)$$

END