TTIC 31230, Fundamentals of Deep Learning

David McAllester, Winter 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.

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

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 Cogntive Toolkit
- Chainer

  ⋮

- EDF (Educational Framework in Python/NumPy for earlier versions of 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 $y \in \{y_1, \ldots, y_k\}$.

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), $\ldots$, (x_{N-1}, y_{N-1}) drawn IID from the population.

A Multi Layer Perceptron (MLP)

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

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

$$\sigma(u) = \frac{1}{1+e^{-u}} \qquad , \sigma(m) = P(y|m) \text{ for margin } m.$$

other common activation functions are

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

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 $-\ln P_\Phi(y|x)$.

$$h = \sigma\left(W^0 x - b^0\right)$$

$$s = \sigma\left(W^1 h - b^1\right)$$

$$P[\hat{y}] = \operatorname*{softmax}_{\hat{y}}\; s[\hat{y}]$$

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

Source Code

$$h = \sigma\left(W^0 x - b^0\right)$$

$$s = \sigma\left(W^1 h - b^1\right)$$

$$P[\hat{y}] = \operatorname*{softmax}_{\hat{y}}\ s[\hat{y}]$$

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

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

Source Code

$$h = \sigma\left(W^0 x - b^0\right)$$

$$s = \sigma\left(W^1 h - b^1\right)$$

$$P_\Phi[\hat{y}] = \operatorname*{softmax}_{\hat{y}} \; s[\hat{y}]$$

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

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)

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

1. Randomly Initialize Φ (initialization is important and must be done with care).

2. Repeat until "converged":

   - draw $z \sim$ Train at random.

   - Φ `-=` $\eta \nabla_\Phi\, \mathcal{L}_\Phi(z)$

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.

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

END