Training Deep Neural Networks

[YouTube Playlist]

Maziar Raissi

Assistant Professor

Department of Applied Mathematics

University of Colorado Boulder

maziar.raissi@colorado.edu

Boulder

Tabular Data

Iris Versicolor   Iris Setosa   Iris Virginica

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
5.5	2.5	4.0	1.3	versicolor
5.0	2.0	3.5	1.0	versicolor
6.7	2.5	5.8	1.8	virginica
5.6	3.0	4.5	1.5	versicolor
5.2	2.7	3.9	1.4	versicolor
5.0	3.5	1.3	0.3	setosa
6.4	2.7	5.3	1.9	virginica
6.4	2.8	5.6	2.2	virginica
5.1	3.8	1.6	0.2	setosa
5.1	3.7	1.5	0.4	setosa

– Number of observations: 150   – Number of variables: 5

The goal of statistics is to come up with a "model" that "explains" the data.
The goal of machine (deep) learning is to come up with an "algorithm" that "performs" well on some "test" data.

$x_i \in \mathbb{R}^d, y_i \in \{1, 2, \dots, k\}, i = 1, \dots, N \to$ training data

$x'_i \in \mathbb{R}^d, y'_i \in \{1, 2, \dots, k\}, i = 1, \dots, N' \to$ validation data

$x''_i \in \mathbb{R}^d, y''_i \in \{1, 2, \dots, k\}, i = 1, \dots, N'' \to$ test data

Performance metric (e.g., accuracy)

$\hat{y} = f_{\alpha,\beta}(x) \to$ prediction of model $f_{\alpha,\beta}$ at input x

$\alpha \to$ hyper-parameters of the model

$\beta \to$ parameters of the model

$$\frac{\sum_{i=1}^{N''} \mathbb{1}(\hat{y}''_i = y''_i)}{N''} \to \text{accuracy}$$

$\hat{y}''_i = f_{\alpha^*,\beta^*}(x''_i) \to$ predictions of the "optimal" model on the test data

Model/Algorithm

$$f_{\alpha,\beta}(x) = \arg \min_{j=1,\dots,k} p^{(j)}_{\alpha,\beta}(x)$$

$p^{(j)}_{\alpha,\beta}(x) \to j$-th element of the probability vector $p_{\alpha,\beta}(x) \in \mathbb{R}^k$

$\sum_{j=1}^k p^{(j)}_{\alpha,\beta}(x) = 1$ and $p^{(j)}_{\alpha,\beta}(x) \geq 0, \forall j = 1, \dots, k.$

Training

$$\mathcal{L}_\alpha(\beta) = -\sum_{i=1}^{N} \log p^{(y_i)}_{\alpha,\beta}(x_i) \to \text{loss function (negative log likelihood)}$$

$$\beta^* = \arg\min_{\beta} \mathcal{L}_\alpha(\beta) \to \text{given } \alpha \text{ (i.e., } \beta^* \text{ is a function of } \alpha)$$

Validation

$\hat{y}'_i = f_{\alpha,\beta^*(\alpha)}(x'_i) \to$ predictions of the model on the validation data

$$\alpha^* = \arg\min_{\alpha} \frac{\sum_{i=1}^{N'} \mathbb{1}(\hat{y}'_i = y'_i)}{N'}$$

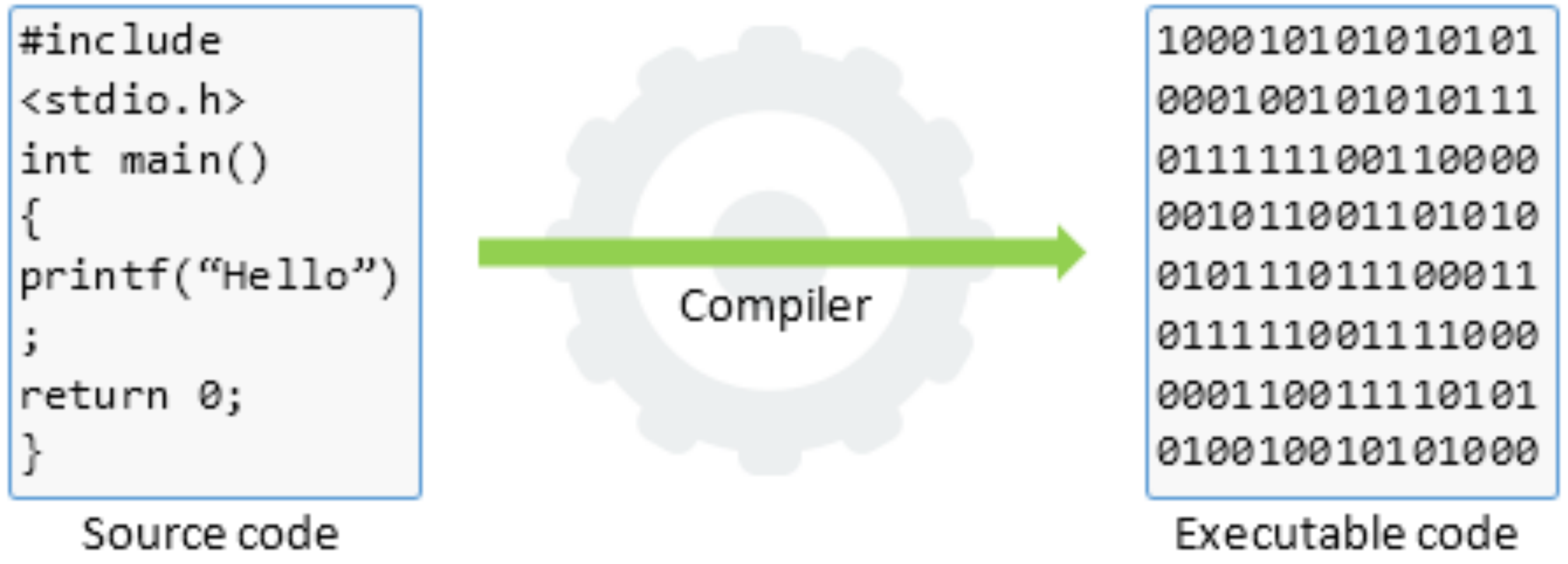
Multi-Layer Perceptron (MLP)

$$p_{\alpha,\beta}(x) = \text{softmax}(W \underbrace{\text{ReLU}(Vx + a)}_{=:h} + b)$$

Boulder

An overview of gradient descent optimization algorithms

YouTube Playlist



Deep Learning: An algorithm that writes an algorithm

Source Code: Data (examples/experiences)
Compiler: Deep Learning
Executable Code: Deployable Model

Deep: Function Compositions $f_L \circ f_{L-1} \circ \ldots f_2 \circ f_1$
Learning: Loss Function, Back-propagation, and Gradient Descent

$$\min_\theta L(\theta)$$

$L(\theta) \approx J(\theta) \rightarrow$ noisy estimate of the objective function (e.g., due to mini-batching)

Stochastic Gradient Descent

$$\theta_{t+1} = \theta_t - \gamma \nabla_\theta J(\theta_t)$$

$J : \mathbb{R}^d \rightarrow \mathbb{R}$
$\nabla_\theta J : \mathbb{R}^d \rightarrow \mathbb{R}^d$
one back-propagation (fast)

$\nabla_\theta J : \mathbb{R}^d \rightarrow \mathbb{R}^d$
$\nabla^2_\theta J : \mathbb{R}^d \rightarrow \mathbb{R}^{d \times d}$
d back-propagations (slow)

Momentum

$$v_t = \gamma v_{t-1} + \eta \nabla_\theta J(\theta_t)$$
$$\theta_{t+1} = \theta_t - v_t$$

Nesterov Accelerated Gradient (NAG)

$$v_t = \gamma v_{t-1} + \eta \nabla_\theta J(\theta_t - \gamma v_{t-1})$$

Adagrad

$$g_t = \nabla_\theta J(\theta_t)$$
$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{G_t + \epsilon}} \odot g_t$$
$$G_t = \sum_{\tau=1}^{t} g_\tau \odot g_\tau$$

Adadelta

$$E[g^2]_t = \gamma E[g^2]_{t-1} + (1-\gamma) g_t^2$$
$$\theta_{t+1} = \theta_t + \Delta\theta_t$$
$$\Delta\theta_t = -\frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} g_t$$

units don't match

$$\Delta\theta_t = -\frac{RMS[\Delta\theta]_{t-1}}{RMS[g]_t} g_t$$

RMSprop

$$E[g^2]_t = 0.9 E[g^2]_{t-1} + 0.1 g_t^2$$
$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} g_t$$

Adam

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t$$
$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2$$

$$\left.\begin{aligned} \hat{m}_t &= \frac{m_t}{1-\beta_1^t} \\ \hat{v}_t &= \frac{v_t}{1-\beta_2^t} \end{aligned}\right\} \text{bias-corrected}$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t$$

AdaMax

$$u_t = \max(\beta_2 u_{t-1}, |g_t|) \qquad \theta_{t+1} = \theta_t - \frac{\eta}{u_t} \hat{m}_t$$

Nadam

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} (\beta_1 \boxed{\hat{m}_t} + \frac{(1-\beta_1) g_t}{1-\beta_1^t})$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} (\beta_1 \hat{m}_{t-1} + \frac{(1-\beta_1) g_t}{1-\beta_1^t}) \rightarrow \text{Adam}$$

Ruder, Sebastian. "An overview of gradient descent optimization algorithms." *arXiv preprint arXiv:1609.04747* (2016).

Questions?

YouTube Playlist