Mathematical Foundations of Deep Neural Networks, M1407.001200
E. Ryu
Fall 2024

Homework 3
Due 5pm, Monday, March 25, 2024

Problem 1: *3-layer MLP to fit a univariate function.* Consider the univariate function

$$f_\star(x) = (x-2)\cos(4x).$$

Let $f_\theta(x)$ be a 3-layer MLP with the sigmoid activation function, i.e., $\sigma(x) = (1+e^{-x})^{-1}$. Given data X_i generated as IID unit Gaussians and corresponding labels $Y_i = f_\star(X_i)$ for $i = 1, \dots, N$, define the loss function

$$\mathcal{L}(\theta) = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{2}(f_\theta(X_i) - Y_i)^2.$$

Use PyTorch to train f_θ with minibatch shuffled cyclic SGD applied to

$$\underset{\theta\in\mathbb{R}^p}{\text{minimize}} \quad \mathcal{L}(\theta).$$

Use layer widths $(n_0, n_1, n_2, n_3) = (1, 64, 64, 1)$, total epochs $K = 1000$ (this is epochs, not iterations), stepsize $\alpha = 0.1$, batch size $B = 128$, and number of data points $N = 512$. (Use the starter code `threelayerSGD.py`.) For all three layers, initialize the weights as IID unit Gaussians and biases with the (deterministic) value 0.03. Plot the final trained function with $f_{\theta_K}(x)$ as a function of x.

Hint. For initialization, do something like

```
model.l1.weight.data = torch.normal(0, 1, model.l1.weight.shape)
model.l1.bias.data = torch.full(model.l1.bias.shape, 0.03)
```

Hint. For the squared loss, use `nn.MSELoss()`.

Problem 2: *Deep learning operates under $p \gg N$.* In the previous problem, how many trainable parameters are in the 3-layer MLP? Repeat the previous problem with the training labels

```
y_train = f_true(X_train) + torch.normal(0, 0.5, X_train.shape)
```

How are the results affected?

Remark. From a classical statistical perspective, it is surprising that large neural networks with more parameters p than the number of data points N do not "overfit", even in the presence of label noise. In fact, most of deep learning operates under the regime where there are more unknowns (trainable parameters) than data points. We will revisit this issue later in our discussion of the bias-variance tradeoff and the double descent phenomenon.

Problem 3: *Basic properties of the CE loss.* Define the *cross entropy loss* as

$$\ell^{\mathrm{CE}}(f, y) = -\log\left(\frac{\exp(f_y)}{\sum_{j=1}^{k}\exp(f_j)}\right),$$

where $f \in \mathbb{R}^k$ and $y \in \{1, \ldots, k\}$.

(a) Show that $0 < \ell^{\mathrm{CE}}(f, y) < \infty$.

(b) Let e_i be the i-th unit vector, i.e, e_i is the one-hot vector with value 1 is the i-th coordinate and 0's for all other coordinates. Show that $\ell^{\mathrm{CE}}(\lambda e_y, y) \to 0$ as $\lambda \to \infty$.

Problem 4: *Derivative of max.* Let

$$f(x) = \max\{f_1(x), \ldots, f_N(x)\},$$

where $f_1, \ldots, f_N$ are differentiable univariate functions. Show that if $I = \operatorname{argmax}_{1,\ldots,N}\{f_i(x)\}$ is unique at $x \in \mathbb{R}$ (the maximum is attained by only one function at a given x), then f is differentiable at x and

$$\frac{d}{dx}f(x) = \frac{d}{dx}f_I(x)$$

at x.

Problem 5: *Basic properties of activation functions.* Prove the following basic facts about some commonly used activation functions.

(a) *Idempotence of ReLU.* The ReLU activation $\sigma(z) = \max\{0, z\}$ is *idempotent*, i.e.,

$$\sigma(\sigma(z)) = \sigma(z), \qquad \forall z \in \mathbb{R}.$$

(b) *Softplus.* The *softplus* function $\sigma(z) = \log(1 + e^z)$ is considered a smooth alternative of ReLU. Show that softplus has Lipschitz continuous derivatives while ReLU does not.

(c) *Equivalence of tanh and sigmoid.* Let $\sigma(z) = (1 + e^{-z})^{-1}$ be the sigmoid function and let $\rho(z) = (1 - e^{-2z})/(1 + e^{-2z})$ be the tanh function. Show that the two activation functions are equivalent in the sense that MLPs built with them are equivalent: given $L > 1$, $A_1, \ldots, A_L$, and $b_1, \ldots, b_L$, there are $C_1, \ldots, C_L$ and $d_1, \ldots d_L$ such that

$$\begin{aligned} y_L &= A_L y_{L-1} + b_L \\ y_{L-1} &= \sigma(A_{L-1}y_{L-2} + b_{L-1}) \\ &\vdots \\ y_2 &= \sigma(A_2 y_1 + b_2) \\ y_1 &= \sigma(A_1 x + b_1), \end{aligned}$$	$$\begin{aligned} y_L &= C_L y_{L-1} + d_L \\ y_{L-1} &= \rho(C_{L-1}y_{L-2} + d_{L-1}) \\ &\vdots \\ y_2 &= \rho(C_2 y_1 + d_2) \\ y_1 &= \rho(C_1 x + d_1), \end{aligned}$$

represent identical $x \mapsto y_L$ mappings, and vice versa. Here, $x \in \mathbb{R}^{n_0}$, $A_\ell, C_\ell \in \mathbb{R}^{n_\ell \times n_{\ell-1}}$, $b_\ell, d_\ell \in \mathbb{R}^{n_\ell}$ for $\ell = 1, \ldots, L$,and σ is applied element-wise.

Remark. The "equivalence" of part (c) should not be understood to mean there is no practical difference between the two activation functions. As we will discuss in later in this course, how one initializes neural network parameters is important. When standard initializations are used, tanh is often easier to train compared to sigmoid, due to the fact that the output of tanh is zero-centered.

Problem 6: *Vanishing gradients.* Consider the 2-layer neural network

$$f_\theta(x) = u^\intercal \sigma(ax+b) = \sum_{j=1}^{p} u_j \sigma(a_j x + b_j),$$

where $x \in \mathbb{R}$ and $a, b, u \in \mathbb{R}^p$. Let σ be the ReLU activation function. Using the data $X_1, \dots, X_N \in \mathbb{R}$ and labels $Y_1, \dots, Y_N \in \mathcal{Y}$, we train the neural network by solving

$$\underset{\theta \in \mathbb{R}^{3p}}{\text{minimize}} \quad \tfrac{1}{N} \textstyle\sum_{i=1}^{N} \ell(f_\theta(X_i), Y_i)$$

with SGD. We assume $\ell(x, y)$ is differentiable in x. Assume the j-th ReLU output is "dead" at initialization in the sense that $a_j^0 X_i + b_j^0 < 0$ for all $i = 1, \dots, N$. Show that j-th ReLU output remains dead throughout the training.

Remark. The term "vanishing gradients" refers both to the circumstance where the gradient exactly vanishes (as in this problem) and to the circumstance where the gradient becomes extremely small but not zero.

Problem 7: *Leaky ReLU.* The *leaky ReLU* activation function [1] is defined as

$$\sigma(z) = \begin{cases} z & \text{for } z \geq 0 \\ \alpha z & \text{otherwise,} \end{cases}$$

where α is a fixed parameter (α is not trained) often set to $\alpha = 0.01$. Show that leaky ReLU, instead of ReLU, is used in the previous problem, the gradient no longer exactly vanishes.

References

[1] A. L. Maas, A. Y. Hannun, and A. Y. Ng, Rectifier nonlinearities improve neural network acoustic models. *ICML Workshop on Deep Learning for Audio, Speech, and Language Processing*, 2013.