Labs
Machine Learning Course
Fall 2022

EPFL
School of Computer and Communication Sciences
Nicolas Flammarion & Martin Jaggi
www.epfl.ch/labs/mlo/machine-learning-cs-433

Problem Set 08, Nov 10, 2022
(Neural Networks & PyTorch Introduction)

Goals. The goals of this exercise are to:

- Introduce you to the PyTorch deep learning framework.
- Explore the representational capacity of neural networks by approximating 2d functions.
- Train a fully connected neural network to classify digits using PyTorch.
- Explore the effects of weight initialization on activation and gradient magnitudes.
- Mathematically analyze basic components of neural network training.

Problem 1 (PyTorch Introduction and Neural Networks):

The accompaning Jupyter Notebook contains a brief introduction to PyTorch along with three neural network exercises. We recommend running the notebook on **Google Colab** which provides you with a free GPU and does not require installing any packages.

1. Open the colab link for the lab 8:
   `https://colab.research.google.com/github/epfml/ML_course/blob/master/labs/ex08/template/ex08.ipynb`
2. To save your progress, click on *"File > Save a Copy in Drive"* to get your own copy of the notebook.
3. Click 'connect' on top right to make the notebook executable (or 'open in playground').
4. Work your way through the introduction and exercises.

Alternatively you can download the notebook from GitHub and install PyTorch locally, see the instructions on **pytorch.org**.

Additional Tutorials: If you plan on using PyTorch in your own projects, we recommend additionally going through the official tutorials after the exercise session:

- Deep Learning with PyTorch: a 60-minute Blitz
- Learning PyTorch with Examples

Problem 2 (Variance Preserving Weight Initialization for ReLUs):

When training neural networks it is desirable to keep the variance of activations roughly constant across layers. Let's assume we have $y = \mathbf{x}^\top \mathbf{w}$ where $\mathbf{x} = \text{ReLU}(\mathbf{z})$, $\mathbf{z} \in \mathbb{R}^d$ and $\mathbf{w} \in \mathbb{R}^d$. Further assume that all elements $\{w_i\}_{i=1}^d$ and $\{z_i\}_{i=1}^d$ are independent with $w_i \sim \mathcal{N}(0, \sigma)$ and $z_i \sim \mathcal{N}(0, 1)$.

Derive $\text{Var}[y]$ as a function of d and σ i.e. how should we set σ to have $\text{Var}[y] = 1$.

Hint: Remember the law of total expecation i.e. $\text{E}_A[A] = \text{E}_B[\text{E}_A[A|B]]$

Problem 3 (Softmax Cross Entropy):

In the notebook exercises we performed multiclass classification using softmax-cross-entropy as our loss. The softmax of a vector $\mathbf{x} = [x_1, ..., x_d]^\top$ is a vector $\mathbf{z} = [z_1, ..., z_d]^\top$ with:

$$z_k = \frac{\exp(x_k)}{\sum_{i=1}^{d} \exp(x_i)} \tag{1}$$

The label y is an integer denoting the target class. To turn y into a probability distribution for use with cross-entropy, we use one-hot encoding:

$$\text{onehot}(y) = \mathbf{y} = [y_1, ..., y_d]^\top \text{ where } y_k = \begin{cases} 1, & \text{if } k = y \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

The cross-entropy is given by:

$$H(\mathbf{y}, \mathbf{z}) = -\sum_{i=1}^{d} y_i \ln(z_i) \tag{3}$$

We ask you to do the following:

1. Equation 1 potentially computes $\exp$ of large positive numbers which is numerically unstable. Modify Eq. 1 to avoid positive numbers in $\exp$. Hint: Use $\max_j(x_j)$.
2. Derive $\frac{\partial H(\mathbf{y},\mathbf{z})}{\partial x_j}$. You may assume that $\mathbf{y}$ is a one-hot vector.
3. What values of x_i minimize the softmax-cross-entropy loss? To avoid complications, practitioners sometimes use a trick called label smoothing where $\mathbf{y}$ is replaced by $\hat{\mathbf{y}} = (1 - \epsilon)\mathbf{y} + \frac{\epsilon}{d}\mathbf{1}$ for some small value e.g. $\epsilon = 0.1$.

Problem 4 (Computation and Memory Requirements of Neural Networks):

Let's consider a fully connected neural network with L layers in total, all of width K. The input is a mini-batch of size $n \times K$. For this exercise we will only consider the matrix multiplications, ignoring biases and activation functions.

- How many multiplications are needed in a forward pass (inference)?
- How many multiplications are needed in a forward + backward pass (training)?
- How much memory is needed for an inference forward pass? The memory needed is the sum of the memory needed for activations and weights. Assume activations are deleted as soon as they are no longer required.
- How much memory is needed for a training forward + backward pass? Note that during training we additionally use the forward activations for the computation of the derivatives.