Due Date : February 16th, 2019

Instructions

- *For all questions, show your work!*
- *Use a document preparation system such as LaTeX.*
- *Submit your answers electronically via Gradescope.*

Question 1 (4-4-4-2)**.** Using the following definition of the derivative and the definition of the Heaviside step function :

$$\frac{d}{dx}f(x) = \lim_{\epsilon \to 0} \frac{f(x+\epsilon) - f(x)}{\epsilon} \qquad H(x) = \begin{cases} 1 & \text{if } x > 0 \\ \frac{1}{2} & \text{if } x = 0 \\ 0 & \text{if } x < 0 \end{cases}$$

1. Show that the derivative of the rectified linear unit $g(x) = \max\{0, x\}$, **wherever it exists**, is equal to the Heaviside step function.
2. Give two alternative definitions of $g(x)$ using $H(x)$.
3. Show that $H(x)$ can be well approximated by the sigmoid function $\sigma(x) = \frac{1}{1+e^{-kx}}$ asymptotically (i.e for large k), where k is a parameter.

*4. Although the Heaviside step function is not differentiable, we can define its **distributional derivative**. For a function F, consider the functional $F[\phi] = \int_{\mathbb{R}} F(x)\phi(x)dx$, where ϕ is a smooth function (infinitely differentiable) with compact support ($\phi(x) = 0$ whenever $|x| \geq A$, for some $A > 0$).
   Show that whenever F is differentiable, $F'[\phi] = -\int_{\mathbb{R}} F(x)\phi'(x)dx$. Using this formula as a definition in the case of non-differentiable functions, show that $H'[\phi] = \phi(0)$. ($\delta[\phi] \doteq \phi(0)$ is known as the Dirac delta function.)

Answer 1. Write your answer here.

Question 2 (5-8-5-5)**.** Let x be an n-dimensional vector. Recall the softmax function : $S : \boldsymbol{x} \in \mathbb{R}^n \mapsto S(\boldsymbol{x}) \in \mathbb{R}^n$ such that $S(\boldsymbol{x})_i = \frac{e^{\boldsymbol{x}_i}}{\sum_j e^{\boldsymbol{x}_j}}$; the diagonal function : $\text{diag}(\boldsymbol{x})_{ij} = \boldsymbol{x}_i$ if $i = j$ and $\text{diag}(\boldsymbol{x})_{ij} = 0$ if $i \neq j$; and the Kronecker delta function : $\delta_{ij} = 1$ if $i = j$ and $\delta_{ij} = 0$ if $i \neq j$.

1. Show that the derivative of the softmax function is $\frac{dS(\boldsymbol{x})_i}{d\boldsymbol{x}_j} = S(\boldsymbol{x})_i \left(\delta_{ij} - S(\boldsymbol{x})_j\right)$.
2. Express the Jacobian matrix $\frac{\partial S(\boldsymbol{x})}{\partial \boldsymbol{x}}$ using matrix-vector notation. Use $\text{diag}(\cdot)$.
3. Compute the Jacobian of the sigmoid function $\sigma(\boldsymbol{x}) = 1/(1 + e^{-\boldsymbol{x}})$.
4. Let $\boldsymbol{y}$ and $\boldsymbol{x}$ be $n-$dimensional vectors related by $\boldsymbol{y} = f(\boldsymbol{x})$, L be an unspecified differentiable loss function. According to the chain rule of calculus, $\nabla_{\boldsymbol{x}} L = (\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}})^\top \nabla_{\boldsymbol{y}} L$, which takes up $\mathcal{O}(n^2)$ computational time in general. Show that if $f(\boldsymbol{x}) = \sigma(\boldsymbol{x})$ or $f(\boldsymbol{x}) = S(\boldsymbol{x})$, the above matrix-vector multiplication can be simplified to a $\mathcal{O}(n)$ operation.

Answer 2. Write your answer here.

Question 3 (3-3-3-3)**.** Recall the definition of the softmax function : $S(\boldsymbol{x})_i = e^{\boldsymbol{x}_i} / \sum_j e^{\boldsymbol{x}_j}$.

1. Show that softmax is translation-invariant, that is : $S(\boldsymbol{x}+c) = S(\boldsymbol{x})$, where c is a scalar constant.

2. Show that softmax is not invariant under scalar multiplication. Let $S_c(\boldsymbol{x}) = S(c\boldsymbol{x})$ where $c \geq 0$. What are the effects of taking c to be 0 and arbitrarily large ?
3. Let $\boldsymbol{x}$ be a 2-dimentional vector. One can represent a 2-class categorical probability using softmax $S(\boldsymbol{x})$. Show that $S(\boldsymbol{x})$ can be reparameterized using sigmoid function, i.e. $S(\boldsymbol{x}) = [\sigma(z), 1-\sigma(z)]^\top$ where z is a scalar function of $\boldsymbol{x}$.
4. Let $\boldsymbol{x}$ be a K-dimentional vector ($K \geq 2$). Show that $S(\boldsymbol{x})$ can be represented using $K-1$ parameters, i.e. $S(\boldsymbol{x}) = S([0, y_1, y_2, ..., y_{K-1}]^\top)$ where y_i is a scalar function of $\boldsymbol{x}$ for $i \in \{1, ..., K-1\}$.

Answer 3. Write your answer here.

Question 4 (15)**.** Consider a 2-layer neural network $y : \mathbb{R}^D \to \mathbb{R}^K$ of the form :

$$y(x, \Theta, \sigma)_k = \sum_{j=1}^{M} \omega_{kj}^{(2)} \sigma \left(\sum_{i=1}^{D} \omega_{ji}^{(1)} x_i + \omega_{j0}^{(1)} \right) + \omega_{k0}^{(2)}$$

for $1 \leq k \leq K$, with parameters $\Theta = (\omega^{(1)}, \omega^{(2)})$ and logistic sigmoid activation function σ. Show that there exists an equivalent network of the same form, with parameters $\Theta' = (\tilde{\omega}^{(1)}, \tilde{\omega}^{(2)})$ and tanh activation function, such that $y(x, \Theta', \tanh) = y(x, \Theta, \sigma)$ for all $x \in \mathbb{R}^D$, and express Θ' as a function of Θ.

Answer 4. Write your answer here.

Question 5 (2-2-2-2)**.** Given $N \in \mathbb{Z}^+$, we want to show that for any $f : \mathbb{R}^n \to \mathbb{R}^m$ and any sample set $\mathcal{S} \subset \mathbb{R}^n$ of size N, there is a set of parameters for a two-layer network such that the output $y(\boldsymbol{x})$ matches $f(\boldsymbol{x})$ for all $\boldsymbol{x} \in \mathcal{S}$. That is, we want to interpolate f with y on any finite set of samples $\mathcal{S}$.

1. Write the generic form of the function $y : \mathbb{R}^n \to \mathbb{R}^m$ defined by a 2-layer network with $N-1$ hidden units, with linear output and activation function ϕ, in terms of its weights and biases $(\boldsymbol{W}^{(1)}, \boldsymbol{b}^{(1)})$ and $(\boldsymbol{W}^{(2)}, \boldsymbol{b}^{(2)})$.
2. In what follows, we will restrict $\boldsymbol{W}^{(1)}$ to be $\boldsymbol{W}^{(1)} = [\boldsymbol{w}, \cdots, \boldsymbol{w}]^T$ for some $\boldsymbol{w} \in \mathbb{R}^n$ (so the rows of $\boldsymbol{W}^{(1)}$ are all the same). Show that the interpolation problem on the sample set $\mathcal{S} = \{\boldsymbol{x}^{(1)}, \cdots \boldsymbol{x}^{(N)}\} \subset \mathbb{R}^n$ can be reduced to solving a matrix equation : $\boldsymbol{M}\tilde{\boldsymbol{W}}^{(2)} = \boldsymbol{F}$, where $\tilde{\boldsymbol{W}}^{(2)}$ and $\boldsymbol{F}$ are both $N \times m$, given by

$$\tilde{\boldsymbol{W}}^{(2)} = [\boldsymbol{W}^{(2)}, \boldsymbol{b}^{(2)}]^\top \qquad \boldsymbol{F} = [f(\boldsymbol{x}^{(1)}), \cdots, f(\boldsymbol{x}^{(N)})]^\top$$

   Express the $N \times N$ matrix $\boldsymbol{M}$ in terms of $\boldsymbol{w}$, $\boldsymbol{b}^{(1)}$, ϕ and $\boldsymbol{x}^{(i)}$.

*3. **Proof with Relu activation.** Assume $\boldsymbol{x}^{(i)}$ are all distinct. Choose $\boldsymbol{w}$ such that $\boldsymbol{w}^\top \boldsymbol{x}^{(i)}$ are also all distinct (Try to prove the existence of such a $\boldsymbol{w}$, although this is not required for the assignment - See Assignment 0). Set $\boldsymbol{b}_j^{(1)} = -\boldsymbol{w}^\top \boldsymbol{x}^{(j)} + \epsilon$, where $\epsilon > 0$. Find a value of ϵ such that $\boldsymbol{M}$ is triangular with non-zero diagonal elements. Conclude. (Hint : assume an ordering of $\boldsymbol{w}^\top \boldsymbol{x}^{(i)}$.)

*4. **Proof with sigmoid-like activations**. Assume ϕ is continuous, bounded, $\phi(-\infty) = 0$ and $\phi(0) > 0$. Decompose $\boldsymbol{w}$ as $\boldsymbol{w} = \lambda \boldsymbol{u}$. Set $\boldsymbol{b}_j^{(1)} = -\lambda \boldsymbol{u}^\top \boldsymbol{x}^{(j)}$. Fixing $\boldsymbol{u}$, show that $\lim_{\lambda \to +\infty} \boldsymbol{M}$ is triangular with non-zero diagonal elements. Conclude. (Note that doing so preserves the distinctness of $\boldsymbol{w}^\top \boldsymbol{x}^{(i)}$.)

Answer 5. Write your answer here.

Question 6 (6)**.** Compute the *full*, *valid*, and *same* convolution (with kernel flipping) for the following 1D matrices : $\big[1, 2, 3, 4\big] * \big[1, 0, 2\big]$

Answer 6. Write your answer here.

Question 7 (5-5)**.** Consider a convolutional neural network. Assume the input is a colorful image of size 256×256 in the RGB representation. The first layer convolves 64 8×8 kernels with the input, using a stride of 2 and no padding. The second layer downsamples the output of the first layer with a 5×5 non-overlapping max pooling. The third layer convolves 128 4×4 kernels with a stride of 1 and a zero-padding of size 1 on each border.

1. What is the dimensionality (scalar) of the output of the last layer ?
2. Not including the biases, how many parameters are needed for the last layer ?

Answer 7. Write your answer here.

Question 8 (4-4-4)**.** Assume we are given data of size $3 \times 64 \times 64$. In what follows, provide the correct configuration of a convolutional neural network layer that satisfies the specified assumption. Answer with the window size of kernel (k), stride (s), padding (p), and dilation (d, with convention $d = 0$ for no dilation). Use square windows only (e.g. same k for both width and height).

1. The output shape of the first layer is $(64, 32, 32)$.
   (a) Assume $k = 8$ without dilation.
   (b) Assume $d = 6$, and $s = 2$.
2. The output shape of the second layer is $(64, 8, 8)$. Assume $p = 0$ and $d = 0$.
   (a) Specify k and s for pooling with non-overlapping window.
   (b) What is output shape if $k = 8$ and $s = 4$ instead ?
3. The output shape of the last layer is $(128, 4, 4)$.
   (a) Assume we are not using padding or dilation.
   (b) Assume $d = 1$, $p = 2$.
   (c) Assume $p = 1$, $d = 0$.

Answer 8. Write your answer here.

Due Date: February 16th, 2019
Instructions

- *For all questions, show your work!*
- *Submit your code and your report (pdf) electronically via the course Gradescope page. If you use the Jupyter notebook, please export it as pdf and submit via Gradescope.*
- *You should push your code to a Github repository, and include the link to the repository in your report!*

Problem 1

In this problem, we will build a Multilayer Perceptron (MLP) and train it on the MNIST handwritten digit dataset [1].

Building the Model [35] Consider an MLP with two hidden layers with h^1 and h^2 hidden units. For the MNIST dataset, the number of features of the input data h^0 is 784. The output of the neural network is parameterized by a softmax of $h^3 = 10$ classes.

1. Build an MLP and choose the values of h^1 and h^2 such that the total number of parameters (including biases) falls within the range of [0.5M, 1.0M].
2. Implement the forward and backward propagation of the MLP in numpy without using any of the deep learning frameworks that provides automatic differentiation. Use the class structure provided here[2].
3. Train the MLP using the probability loss (*cross entropy*) as training criterion. We minimize this criterion to optimize the model parameters using *stochastic gradient descent*.

In the following sub-questions, please specify the *model architecture* (number of hidden units per layer, and the total number of parameters), the *nonlinearity* chosen as neuron activation, *learning rate*, *mini-batch size*.

Initialization [10] In this sub-question, we consider different initial values for the weight parameters. Set the biases to be zeros, and consider the following settings for the weight parameters:

- **Zero**: all weight parameters are initialized to be zeros (like biases).

[1] `http://yann.lecun.com/exdb/mnist/` - Use the standard train/valid/test splits such as the one provided *here*.
[2] `https://github.com/CW-Huang/IFT6135H19_assignment/blob/master/assignment1.ipynb`

- **Normal**: sample the initial weight values from a standard Normal distribution; $w_{i,j} \sim \mathcal{N}(w_{i,j}; 0, 1)$.
- **Glorot**: sample the initial weight values from a uniform distribution; $w^l_{i,j} \sim \mathcal{U}(w^l_{i,j}; -d^l, d^l)$ where $d^l = \sqrt{\frac{6}{h^{l-1}+h^l}}$.

1. Train the model for 10 epochs [3] using the initialization methods above and record the average loss measured on the training data at the end of each epoch (10 values for each setup).
2. Compare the three setups by plotting the losses against the training time (epoch) and comment on the result.

Hyperparameter Search [10] From now on, use the Glorot initialization method.

1. Find out a combination of hyper-parameters (model architecture, learning rate, nonlinearity, etc.) such that the average accuracy rate on the validation set ($r^{(valid)}$) is at least 97%.
2. Report the hyper-parameters you tried and the corresponding $r^{(valid)}$.

Validate Gradients using Finite Difference [15] The finite difference gradient approximation of a scalar function $x \in \mathbb{R} \mapsto f(x) \in \mathbb{R}$, of precision ϵ, is defined as $\frac{f(x+\epsilon)-f(x-\epsilon)}{2\epsilon}$. Consider the first layer weights of the MLP you built in the previous section, as a vector $\theta = (\theta_1, \ldots, \theta_m)$. We are interested in approximating the gradient of the loss function L, evaluated using **one** training sample, at the end of training, with respect to $\theta_{1:p}$, the first $p = \min(10, m)$ elements of θ, using finite differences.

1. Evaluate the finite difference gradients $\nabla^N \in \mathbb{R}^p$ using $\epsilon = \frac{1}{N}$ for different values of N

$$\nabla^N_i = \frac{L(\theta_1, \ldots, \theta_{i-1}, \theta_i + \epsilon, \theta_{i+1}, \ldots, \theta_p) - L(\theta_1, \ldots, \theta_{i-1}, \theta_i - \epsilon, \theta_{i+1}, \ldots, \theta_p)}{2\epsilon}$$

   Use at least 5 values of N from the set $\{k10^i : \ i \in \{0, \ldots, 5\}, \ k \in \{1, 5\}\}$.
2. Plot the maximum difference between the true gradient and the finite difference gradient ($\max_{1 \le i \le p} |\nabla^N_i - \frac{\partial L}{\partial \theta_i}|$) as a function of N. Comment on the result.

[3] One epoch is one pass through the whole training set.

Problem 2

Convolutional Networks: Many techniques correspond to incorporating certain prior knowledge of the structure of the data into the parameterization of the model. Convolution operation, for example, is designed for visual imagery.

Instructions: [25] For this part of the assignment we will train a convolutional network on MNIST for 10 epochs using your favorite deep learning frameworks such as Pytorch of Tensorflow. Plot the train and valid errors at the end of each epoch for the model.

1. Come up with a CNN architecture with more or less similar number of parameters as MLP trained in Problem 1 and describe it.
2. Compare the performances of CNN vs MLP. Comment.

You could take reference from the architecture mentioned here[4].

Problem 3

Dogs vs. Cats is an InclassKaggle challenge for image classification. Using what you have learned from the class and from previous problems, make an attempt at training the best CNN classifier for this task. Use the link here[5] to download the data and access the competition. Make sure to read the description and the rules thoroughly.

- The data is already preprocessed and ready to use, but you are free to add more layers of preprocessing if you wish. You may for example do your own cropping. Any extra preprocessing step should be mentioned in your report. You are not allowed to use any external data sources, but you can use usual data augmentation techniques. Please make sure to mention them. You are responsible for splitting the train set to an actual train and validation sets yourself.
- In addition to Kaggle submissions, we require that you submit your code through Gradescope, along with your report. We will run your code and make sure we can recreate your submission.
- You cannot use things that we have not covered in the class, directly from the deep learning library you are using, such as BatchNorm/WeightNorm/LayerNorm layers, regularization techniques (including dropout), and optimizers such as ADAM, "unless you implement them yourself".

[4] https://github.com/MaximumEntropy/welcome_tutorials/tree/pytorch/pytorch
[5] https://www.kaggle.com/c/ift6135h19

- This problem is meant to give you a chance to play with the methods seen in class and apply them to a real classification task. Show your efforts by recording what you have tried on the report (even things that did not lead to high scores).
- Your mark will be a combination of the accuracy score you obtained on the private leaderboard (more details on Kaggle) and the quality of your report. Your report should include the answers to the following questions:

Important: The Kaggle submission deadline is 1 day before the assignment deadline.

1. [20 score/5 description] Describe the architecture (number of layers, filter sizes, pooling, etc.), and report the number of parameters. You can take inspiration from some modern deep neural network architectures such as the VGG networks to improve the performance.
2. [15] Plot the training error and validation error curves, along with the training and validation losses. Comment on them. What techniques (you did not implement) could be useful to improve the validation performance. How does your validation performance compare to the test set performance (that you can only get in Kaggle).
3. [15] Compare different hyperparameter settings and report the final results of performance on your validation set. Aside from quantitative results, also include some visual analysis such as visualizing the feature maps or kernels, or showing examples where the images are (a) clearly misclassified and (b) where the classifier predicts around 50% on both classes. Explain your observation and/or suggest any improvements you think may help.