Homework 1

Instructions: There are a total of 29 points for this homework. Each question is marked with the number of points it's worth. Some questions are **not graded**, including all bonus questions. You may either hand-draw or computer-generate plots as you find appropriate, but just make sure the important trends are clear.

Notation: We will use this math notation from the course webpage. For example, c is a scalar, $\mathbf{b}$ is a vector and $\mathbf{W}$ is a matrix. You are encouraged (though not forced) to follow this notation in a typeset submission, or to the best of your ability in a handwritten response—bolding vectors may be difficult :).

Math primer: A few questions use terms like "convexity", "discontinuity" and "differentiability". To solve the problems, only an informal understanding of these concepts is needed. For example, ReLU(x) is continuous because it can be drawn without lifting your pen from the paper. ReLU(x) is not differentiable everywhere since it has a "kink" at $x = 0$.

Approximation (14pt)

For this section, we consider functions represented by ReLU networks with a single real-valued input and a single real-valued output, unless otherwise specified. Let l denote the number of layers. For example, a network with $l = 2$ layers is written:

$$f(\mathbf{x}; \mathbf{W}_1, \mathbf{W}_2, \mathbf{b}_1, \mathbf{b}_2) = \mathbf{W}_2 \operatorname{ReLU}(\mathbf{W}_1 \mathbf{x} + \mathbf{b}_1) + \mathbf{b}_2,$$

where input $\mathbf{x}$ and output $f(\mathbf{x}; \mathbf{W}_1, \mathbf{W}_2, \mathbf{b}_1, \mathbf{b}_2)$ are scalars, unless otherwise specified (and l is the number of weight matrices).

1. **(1pt)** Consider a two-layer ReLU network with width k (i.e., k hidden neurons) with weight matrices $\mathbf{W}_1$, $\mathbf{W}_2$ and bias vectors $\mathbf{b}_1, \mathbf{b}_2$. $\mathbf{W}_1$ is a matrix in $\mathbb{R}^{k\times 1}$—write down the shapes of $\mathbf{W}_2, \mathbf{b}_1$ and $\mathbf{b}_2$.

2. **(1pt)** Write out the expression for a ReLU network with $l = 3$ layers.

3. **(1pt)** Answer yes or no: For a ReLU network with $l \geq 2$ layers, in general, is the network output convex with respect to the input $\mathbf{x}$? Is it concave?
   Convex: (yes/no)
   Concave: (yes/no)

4. **(5pt)** Consider a ReLU network with l layers, each of width k.

   (a) **(1pt)** How many discontinuities can the output have with respect to the input?

(b) **(1pt)** Choose one. As a function of the input, the output is always:
(A) linear, (B) piecewise-linear, or (C) polynomial.

(c) **(2pt)** In general, is the function differentiable at every input? If yes, why? If no:

i. What's the smallest number of input points at which the function can be non-differentiable?

ii. For $l = 2$ layers, what's the largest number of input points at which the function can be non-differentiable?

iii. For a general number of layers l, what's the largest number of input points at which the function can be non-differentiable? Choose one:
(A) Constant in l (B) Linear in l (C) polynomial in l (D) exponential in l.

Hint:

- It is hard to derive an exact answer, so focus on asymptotics.
- How are non-differentiable points related to linear regions?
- Each neuron is a separating hyperplane of the previous layer output.
- Assume that each linear region of the previous layer is divided (into two) by some hyperplane in the current layer. How does adding a layer affect the number of linear regions?

iv. Recall from lecture that a 2-layer network ($l = 2$) with sufficient width k is a universal approximator. Based on your answer to the previous questions, can deeper ReLU networks ($l > 2$) be more efficient (in terms of total number of neurons / hidden units) in approximating some functions?

(d) **(1pt)** In general, is the function differentiable at every input if a $\tanh$ nonlinearity is used instead of ReLU? If yes, why? If no, re-do the sub-points of part (c).

5. **(2pt)** For a 2-layer ReLU network with width 2 and *no* biases (i.e., $\mathbf{b}_1$ and $\mathbf{b}_2$ are all zeros), we aim to find $\mathbf{W}_1$ and $\mathbf{W}_2$ so that the corresponding network has different smoothness properties. For each case,

- If there exist $\mathbf{W}_1$ and $\mathbf{W}_2$ such that the corresponding property holds for input $x \in [-5, 5]$, provide an example of such $\mathbf{W}_1$ and $\mathbf{W}_2$ and, for that example, provide a plot of the function over $x \in [-5, 5]$.
- If there do not exist such $\mathbf{W}_1$ and $\mathbf{W}_2$, explain why.

(a) **(1pt)** The function is linear.

(b) **(1pt)** The function has 2 non-differentiable points.

(c) **(BONUS; 0pt)** The function is convex (and not linear).

(d) **(BONUS; 0pt)** The function is neither convex nor concave.

6. **(BONUS; non-convexity of neural networks; 0pt)** Consider a 2-layer ReLU network. Plot an example to show that the network output is not guaranteed to be convex (or concave) w.r.t. **network parameters**. You can pick the network width (although 2 suffices).

   In particular, find a fixed input and a linear path in parameter space, and plot the network output with that fixed input while varying parameters along the linear path. The plot should be neither convex nor concave.

7. **(4pt) Logic gate ReLU networks**

   Hint: $\mathrm{ReLU}(x)$ non-linearity is like a "branching" operation at $x = 0$. Can you find a set of weights such that the desired decision boundaries correspond to zero inputs to ReLU's?

   (a) **(OR gate; 2pt)** Construct a 2-*layer width*-2 ReLU network with 2-*dimensional inputs in* $\mathbb{R}^2$ such that:

   $$f(\mathbf{x}; \mathbf{W}_1, \mathbf{W}_2, \mathbf{b}_1, \mathbf{b}_2) > 0 \iff \mathbf{x}_1 > 0 \text{ OR } \mathbf{x}_2 > 0 \tag{1}$$

   Write out the algebraic formula of f with explicit $\mathbf{W}_1, \mathbf{W}_2, \mathbf{b}_1, \mathbf{b}_2$.

   Hint: Reminder, the input is not boolean.

   (b) **(XOR gate; 2pt)** Construct a ReLU network with *at most* 3 *layers, each with width at most* 4 and 2-*dimensional inputs in* $\mathbb{R}^2$ such that:

   $$\begin{aligned} f(\mathbf{x}; \mathbf{W}_1, \mathbf{W}_2, \mathbf{b}_1, \mathbf{b}_2) > 0 \iff & (\mathbf{x}_1 < 0 \text{ AND } \mathbf{x}_2 > 0) \text{ OR} \\ & (\mathbf{x}_1 > 0 \text{ AND } \mathbf{x}_2 < 0) \end{aligned} \tag{2}$$

   Write out the algebraic formula of f with explicit weight matrices and bias vectors.

   Hint: This is a more challenging problem. Think about how to implement an AND gate.

   (c) **(BONUS; NAND gate and functional completeness; 0pt)** Write down a ReLU network that implements the NAND gate. What does this tell you about the possibility of representing any boolean function with ReLU networks.

Homework 1

Backpropagation (3pt)

8. **(3pt)** Let $\mathbf{W}$ denote a $d \times d$ real matrix, and consider the following system of equations:

$$\mathbf{y} = \mathbf{W}\mathbf{x} \tag{3}$$
$$\mathbf{u} = \mathrm{ReLU}(\mathbf{y}) \tag{4}$$
$$\mathbf{v} = \mathbf{u} + \mathbf{W}\mathbf{u} \tag{5}$$
$$\mathcal{L} = \tfrac{1}{2}\|\mathbf{v}\|_2^2. \tag{6}$$

Note that $\mathbf{x}, \mathbf{y}, \mathbf{u}$ and $\mathbf{v}$ must all be vectors in $\mathbb{R}^d$ for these equations to make sense. Since $\|\mathbf{v}\|_2^2$ denotes the standard squared Euclidean norm of vector $\mathbf{v}$, $\mathcal{L}$ is a scalar.

(a) **(1pt)** Show that $\dfrac{\partial \mathcal{L}}{\partial \mathbf{W}_{ij}} = \displaystyle\sum_{m=1}^{d} \mathbf{v}_m \cdot \frac{\partial \mathbf{v}_m}{\partial \mathbf{W}_{ij}}$.

In a similar manner to part (a), one may derive the following additional relations:

- $\dfrac{\partial \mathbf{v}_m}{\partial \mathbf{W}_{ij}} = \dfrac{\partial \mathbf{u}_m}{\partial \mathbf{W}_{ij}} + \delta_{im} \cdot u_j + \displaystyle\sum_{l=1}^{d} \mathbf{W}_{ml} \frac{\partial \mathbf{u}_l}{\partial \mathbf{W}_{ij}}$.
- $\dfrac{\partial \mathbf{y}_k}{\partial \mathbf{W}_{ij}} = \delta_{ik}\mathbf{x}_j$,

  where the "Kronecker delta" is given by $\delta_{ik} = \begin{cases} 1 \text{ if } i = k; \\ 0 \text{ if } i \neq k. \end{cases}$
- $\dfrac{\partial \mathbf{u}_l}{\partial \mathbf{y}_k} = \delta_{lk} \cdot \Theta(\mathbf{y}_k)$,

  where Θ denotes the Heaviside step function given by $\Theta(\mathbf{y}_k) = \begin{cases} 1 \text{ if } \mathbf{y}_k \geq 0; \\ 0 \text{ if } \mathbf{y}_k < 0. \end{cases}$

(b) **(2pt)** Let $\frac{\partial \mathcal{L}}{\partial \mathbf{W}}$ denote the matrix with entries $(\frac{\partial \mathcal{L}}{\partial \mathbf{W}})_{ij} = \frac{\partial \mathcal{L}}{\partial \mathbf{W}_{ij}}$. Using the given relations and your answer to part (a), show that:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}} = \mathbf{v} \otimes \mathbf{u} + \mathrm{diag}(\Theta(\mathbf{y}))(\mathbf{I} + \mathbf{W}^\top)\mathbf{v} \otimes \mathbf{x},$$

where $\otimes$ is the outer product and diag shapes its input into a diagonal matrix.

The advantage of this kind of expression for $\frac{\partial \mathcal{L}}{\partial \mathbf{W}}$ is that it is easy to code up in PyTorch, and makes efficient use of matrix multiplication primitives, which have highly optimized, parallelized implementations on GPU.

PyTorch (0pt)

9. **(NOT GRADED; 0pt)** Complete PyTorch tutorial colab notebooks here. Before proceeding with the following section, you should at least complete the notebooks ("Tensor Arithmetic" and "Network Modules").

CIFAR-10 Classification (12pt)

In this section, we are going to work on this colab notebook to train a network for classifying a handwritten digit dataset, CIFAR-10 [Krizhevsky, 2009, Torralba et al., 2008].

The following questions 9-15 are stated in detail in the colab notebook. Please include your added lines of code, text output, and any plots to them in the same pdf submission.

To download a plot from colab, hold `shift` while you right click on the image.

Note: A lot of skeleton code is provided to you already. Make sure to read through and understand them. We will provide *less* skeleton code in future assignments as you get more used to deep learning code structures.

10. **(Building neural networks; 4pt)** Complete the incomplete `forward` and `backward` definitions of the module classes, each using ≤ 5 lines of code. We expect this question to take more time, as you are being asked to derive and implement the backward pass for multiple components. **Hint**: Recall, for linear layers, the forward pass takes the form: $out = \mathbf{W}\mathbf{x} + \mathbf{b}$, and the backward pass requires us to know $\frac{dL}{dout}$, $\frac{dL}{d\mathbf{x}}$, $\frac{dL}{d\mathbf{b}}$. ReLU and Loss layers can be similarly computed.

    (a) **(1pt)** `Linear.forward`

    (b) **(1pt)** `Linear.backward`

    (c) **(1pt)** `ReLU.backward`

    (d) **(1pt)** `CrossEntropyLoss.backward`

11. **(Training loop; 2pt)** Complete the missing parts in `train_epoch` and `evaluate` functions, each using ≤ 5 lines of code.

12. **(Training curve 1; 2pt)** Train a model for 30 epochs and plot the learning curves. Comment on any interesting observation from the plot.

13. **(Universal approximation; 1pt)** Neural networks are universal approximators, which means that we can always find a network that fits the training set. Why do you think that we didn't get perfect training accuracy? Write down some ideas for improving training accuracy. Is a perfect training accuracy all we need?

14. **(Data augmentation; 1pt)** Write code to create training and validation datasets, where **only the training set** has a random cropping augmentation (specifications in colab). Visualize the effect of the augmentation.

    Provide both your code (≤ 5 lines) and your visualization.

15. **(Training curve 2; 2pt)** Train a model on the new training dataset and plot the learning curves. Comment on any difference you observe from the previous curves.

References

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

Antonio Torralba, Rob Fergus, and William T Freeman. 80 million tiny images: A large data set for nonparametric object and scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2008.