CS 577 — Deep Learning — Homework 1

Antonio Castañares Rodríguez, A20599898

08/29/2024

Read these instructions carefully:

- In the LaTeX source code, type your answer in between "`%%% BEGIN ANSWER`" and "`%%% END ANSWER`". For advanced LaTeXusers, you can use your custom macros if you wish by placing them between "`%%% BEGIN MACROS`" and "`%%% END MACROS`" in the header. Do not modify anything else.
- There is a PDF file `latex_symbols.pdf` which lists LaTeX symbols that you will need later.
- The first section "Introduction to Latex (not graded)" is a tutorial. **It is not graded**. Feel free to skip it.
- The second section titled "LaTeX Problems" **will be graded**. You will recreate content from the slides in Lecture 1. You will receive full credit as long as your answer is sufficiently similar visually.
- The third section "Perceptron Problem" **will be graded**.
- When typing mathematics, you can use either the inline mode or display mode (see below). It is up to you. Please make the choice that results in good readability.
- Turn in both your `.tex` file and the generated `.pdf` file.

1 Introduction to Latex (not graded)

[**0**] **points — inline mode:** Mathematics goes inside a pair of single dollar sign like this: "`$...$`". For example, "`$\pi \approx 3.14$`" renders as "$\pi \approx 3.14$"

Write the formula for the circumference C of the circle with radius r

Answer:
$C = 2\pi r$

[**0**] **points — display mode:** Mathematics goes inside a pair of double dollar signs for display mode like this: "`$$...$$`". For example, "`$$\sin(\theta)^2 + \cos(\theta)^2 = 1$$`" renders as

$$\sin(\theta)^2 + \cos(\theta)^2 = 1$$

Type in display mode a function f of x and y defined (you should use colon ":" followed by an equal sign, i.e., :=, for function definition) by $\sin(x)^2$ plus $\cos(y)^2$.

Answer:

$$f(x, y) := \sin(x)^2 + \cos(y)^2$$

[**0**] **points — superscript:** You saw that the caret symbol `^` is for superscript. Now, try taking x to the 1/3 power by directly typing `x^` followed by `1/3`...

Answer:

$$x^1/3$$

If you did this exactly, you probably noticed that it's not quite right. Let's fix that by surrounding the `1/3` with `{...}`.

Answer:

$$x^{1/3}$$

[**0**] **points — subscript:** Subscripts in LaTeX are created using the underscore symbol `_`. For example, typing `x_i` will produce x_i. Now, try typing x with the subscript "100".

Answer:

$$x_{100}$$

[**0**] **points — summation notation:** The command `\sum` is used to produce the summation symbol in LaTeX. For example, `\sum_{...}^{...}` produces $\sum_{...}^{...}$. Now, write the sum of 1 squared, 2 squared, ..., n squared using summation notation.

Answer:

$$1^2 + 2^2 + ... + n^2 = \sum_{i=1}^{N} i^2$$

[**0**] **points — blackboard font:** The set of real numbers is typically written using the blackboard font in LaTeX. To use this font, surround the symbol with `\mathbb{...}`. Now, write the set of real numbers (capital R in blackboard font).

Answer:

$$\mathbb{R}$$

[**0**] **points — minimization notation:** Minimization in LaTeX is similar to summation, except there is no superscript. Now, write the notation for "minimization of x squared where x is in the set of the reals". The symbol for "belongs" is `\in`.

Answer:

$$\min x^2 \in \mathbb{R}x^2$$

---

[0] points — bold symbols: To type a bold symbol in LaTeX, use the command `\mathbf{...}`, where the `...` is replaced with your symbol. Now, write X (uppercase) in bold.

Answer:

$$\mathbf{X}$$

For bolding Greek symbols, you need to use `\bm{...}` instead of `\mathbf{...}`. Write "theta" in bold.

Answer:

$$\boldsymbol{\theta}$$

---

[0] points — uppercase Greek letters: To type an uppercase Greek letter in LaTeX, simply write the name of the letter with an initial capital letter, like `\Theta` for Θ. Note that not all Greek letters have an uppercase version in LaTeX (e.g., there is no capital "alpha"). Now, type the uppercase version of delta.

Answer:

$$\Delta$$

---

[0] points — gradient symbol: Other commands or symbols in LaTeX can also be subscripted or superscripted. The symbol for the gradient is `\nabla`. Now, type the symbol for the partial derivative with respect to bold **x** (just the "nabla" and the "x" part).

Answer:

$$\nabla x$$

---

[0] points — fractions: Fractions in LaTeX are written using the command `\frac{numerator}{denominator}`. Now, write Newton's law of universal gravitation: F is the force of gravity, G is the gravitational constant, m_1 and m_2 are the masses of the objects, and r is the distance between the objects.

Answer:

$$F = G\frac{m_1 m_2}{r^2}$$

1.1 Linear algebra

[**0**] **points — matrices:** You can write matrices in LaTeX using the `bmatrix` environment. For example, the matrix

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

is written as:

```
\begin{bmatrix}
a & b \\
c & d
\end{bmatrix}
```

Now, write down the formula for the inverse of this matrix.

Answer:

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}^{-1}$$

[**0**] **points — big matrices:** You can write a big matrix in LaTeX using dots for representing a sequence of elements. Here's an example of a large matrix:

$$\begin{bmatrix} x_{11} & \cdots & x_{1N} \\ \vdots & \ddots & \vdots \\ x_{N1} & \cdots & x_{NN} \end{bmatrix}$$

In this example, `\cdots` represents horizontal (centered) dots, `\vdots` represents vertical dots, and `\ddots` represents diagonal dots.

Now, create a column vector with $x_1, \ldots, x_N$ and a row vector with $x_1, \ldots, x_N$. Hint: You only need `&` but not `\\` for the row vector.

Answer:

$$\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{pmatrix} \begin{pmatrix} x_1 & x_2 & \cdots & x_N \end{pmatrix}$$

1.2 Lists

[**0**] **points — nested lists:** In LaTeX, you can create a nested bulleted list using the `itemize` environment like this:

```
\begin{itemize}
    \item A
    \item B
    \begin{itemize}
        \item X
        \item Y
        \item Z
    \end{itemize}
```

```
    \item C
\end{itemize}
```

This will produce the following nested list:

- A
- B
  - X
  - Y
  - Z
- C

To create a numbered list, use the `enumerate` environment instead of `itemize`. Nested lists can be a hacky way to write pseudo algorithms (with code indentation) in LaTeX. Now, write the pseudocode to compute the n-th Fibonacci number using nested numbered list.

> **Answer:**
>
> 1. If input is 0 or 1 or 2.
>
>   1.1 Return Incorrect input, 0 or 1 respectively.
>
> 2. Return the sum of the two previous number of this input.

2 LaTeX Problems

Note: The following problems refer to slides from Lecture 01.

[2] points — Empirical risk minimization: Write the empirical risk minimization (ERM) on slide number "27/73" (ignore my handwritten notes) of . Consult the `latex_symbols.pdf` page 4 for the matrix transpose symbol.

> **Answer:**
>
> $$min_{\theta \in \Theta} J(\boldsymbol{\theta}) := \frac{1}{N} \sum_{i=1}^{N} L(f(x_i; \boldsymbol{\theta}), y_i)$$

---

[2] points — Gradient: Write the gradient on slide number "29/73" (ignore my handwritten notes). Consult the `latex_symbols.pdf` page 4 for the partial derivative symbol.

> **Answer:**
>
> $$\nabla_x f(\mathbf{x}) = [\frac{\partial f}{\partial x_1} f(\mathbf{x}) \cdots \frac{\partial f}{\partial x_d} f(\mathbf{x})]^T$$

---

[3] points — Stochastic gradient descent algorithm: Write the content on slide number "42/73" (ignore my handwritten notes) starting from "Let...". (Ignore the slide numbers in your answer.) The curly brackets are `\{...\}` and the left arrow is `\gets`.

Answer:
Let $e_k > 0$ be learning rates, $k = 1, 2, \cdots$

- Initialize $\boldsymbol{\theta}$
- While not converged (k = iteration counter):
  - Select m samples $\{\mathbf{x}^{(1)}, \cdots, \mathbf{x}^{(m)}\}$ and matching labels $\{y^{(1)}, \cdots, y^{(i)}\}$
  - Compute gradient $\mathbf{g} \leftarrow \nabla_{\boldsymbol{\theta}} \frac{1}{m} \sum_{i=1}^{m} L(f(\mathbf{x}^{(i)}, \boldsymbol{\theta}), y^{(i)}$
  - Compute update $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - e_k g$

---

[3] points — Perceptron algorithm: Write the content inside the "Perceptron update" blue box on slide number "53/73" (ignore my handwritten notes). (Just the pseudocode. Ignore the title text of the box.)

Answer:
Input: $(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \cdots$, time T

1. Initialize $\mathbf{w} = \mathbf{0}$.
2. For $t = 1, 2, \cdots, T$

   2.1 If $y^{(t)}\mathbf{w}^{(t)}\mathbf{x}^{(t)} > 0$, then $\mathbf{w} \leftarrow \mathbf{w}$,

   2.2 Else, then $\mathbf{w} \leftarrow \mathbf{w} + y^{(t)}\mathbf{x}^{(t)}$.

Output: $\mathbf{w}$

3 Perceptron Problem

[10] points — Perceptron calculation: In this problem, you will run the Perceptron update for the following dataset for $T = 8$ iterations. It is possible for the Perceptron to converge before all T iterations. In that case, you may declare that "the perceptron converges after this iteration" and stop.

- $\mathbf{x}^{(1)} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ "east", $y^{(1)} = +1$
- $\mathbf{x}^{(2)} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$ "northeast", $y^{(2)} = +1$
- $\mathbf{x}^{(3)} = \begin{bmatrix} 0 \\ 2 \end{bmatrix}$ "north", $y^{(3)} = -1$
- $\mathbf{x}^{(4)} = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$ "northwest", $y^{(4)} = -1$
- $\mathbf{x}^{(5)} = \begin{bmatrix} -1 \\ 0 \end{bmatrix}$ "west", $y^{(5)} = -1$
- $\mathbf{x}^{(6)} = \begin{bmatrix} -1 \\ -1 \end{bmatrix}$ "southwest", $y^{(6)} = -1$
- $\mathbf{x}^{(7)} = \begin{bmatrix} 0 \\ -1 \end{bmatrix}$ "south", $y^{(7)} = +1$
- $\mathbf{x}^{(8)} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$ "southeast", $y^{(8)} = +1$

Write your answer in the box below. Show all your calculations. Go through each iteration thoroughly. Make sure that you address

- What is $\mathbf{w}$ at the beginning/end of an iteration?
- What is the calculation that goes into checking the "If/Else" statement?

Answer:
Being:

$$\mathbf{X} = [[1,0],[1,1],[0,2],[-1,1],[-1,0],[-1,-1],[0,-1],[1,-1]]$$

$$\mathbf{y} = [1,1,-1,-1,-1,-1,1,1]$$

and using the Perception algorithm, we have the next iterations:

1. **Case else: w initial** $= \begin{bmatrix}0\\0\end{bmatrix} \to 1 \cdot \begin{bmatrix}0 & 0\end{bmatrix} \cdot \begin{bmatrix}1\\0\end{bmatrix} = 0 \to \mathbf{w_f} = \begin{bmatrix}0\\0\end{bmatrix} + 1 \cdot \begin{bmatrix}1\\0\end{bmatrix} = \begin{bmatrix}1\\0\end{bmatrix}$
2. **Case if: w initial** $= \begin{bmatrix}1\\0\end{bmatrix} \to 1 \cdot \begin{bmatrix}1 & 0\end{bmatrix} \cdot \begin{bmatrix}1\\1\end{bmatrix} = 1 > 0 \to \mathbf{w_f} = \mathbf{w_i}$
3. **Case else: w initial** $= \begin{bmatrix}1\\0\end{bmatrix} \to (-1) \cdot \begin{bmatrix}1 & 0\end{bmatrix} \cdot \begin{bmatrix}0\\2\end{bmatrix} = 0 \to \mathbf{w_f} = \begin{bmatrix}1\\0\end{bmatrix} + (-1) \cdot \begin{bmatrix}0\\2\end{bmatrix} = \begin{bmatrix}1\\-2\end{bmatrix}$
4. **Case if: w initial** $= \begin{bmatrix}1\\-2\end{bmatrix} \to (-1) \cdot \begin{bmatrix}1 & -2\end{bmatrix} \cdot \begin{bmatrix}-1\\1\end{bmatrix} = 3 > 0 \to \mathbf{w_f} = \mathbf{w_i}$
5. **Case if: w initial** $= \begin{bmatrix}1\\-2\end{bmatrix} \to (-1) \cdot \begin{bmatrix}1 & -2\end{bmatrix} \cdot \begin{bmatrix}-1\\0\end{bmatrix} = 1 > 0 \to \mathbf{w_f} = \mathbf{w_i}$
6. **Case else: w initial** $= \begin{bmatrix}1\\-2\end{bmatrix} \to (-1) \cdot \begin{bmatrix}1 & -2\end{bmatrix} \cdot \begin{bmatrix}-1\\-1\end{bmatrix} = -1 < 0 \to \mathbf{w_f} = \begin{bmatrix}1\\-2\end{bmatrix} + (-1) \cdot \begin{bmatrix}-1\\-1\end{bmatrix} = \begin{bmatrix}2\\-1\end{bmatrix}$
7. **Case if: w initial** $= \begin{bmatrix}2\\-1\end{bmatrix} \to 1 \cdot \begin{bmatrix}2 & -1\end{bmatrix} \cdot \begin{bmatrix}0\\-1\end{bmatrix} = 1 > 0 \to \mathbf{w_f} = \mathbf{w_i}$
8. **Case if: w initial** $= \begin{bmatrix}2\\-1\end{bmatrix} \to 1 \cdot \begin{bmatrix}2 & -1\end{bmatrix} * \begin{bmatrix}1\\-1\end{bmatrix} = 3 > 0 \to \mathbf{w_f} = \mathbf{w_i}$

Python code:

```python
# Imports
import numpy as np

# Algorithm
def perceptron_update(X,y,T):
  w = np.zeros((1,2))
  for t in range(T):
    if y[t] * (np.dot(w,X[t])) <= 0:
      print("Caso else")
      w = w + y[t]*X[t]
    else:
      print("Caso if")
    print(w)

# Main
X = np.array([[1,0],[1,1],[0,2],[-1,1],[-1,0],[-1,-1],[0,-1],[1,-1]])
y = np.array([1,1,-1,-1,-1,-1,1,1])
T = 8

perceptron_update(X,y,T)
```