CS 577 — Deep Learning — Homework 2

Antonio Castañares Rodríguez, A20599898

09/12/2024

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.
- Turn in both your `.tex` file and the generated `.pdf` file.

1 Conditional Probability

[3] point(s) — part a: This exercise is designed to help you develop intuition for the concept of conditional probability by "creating one from scratch".

In mathematics, a *function* takes an input, let's call it x, to exactly one output, let's call it y. For example, if $w = 1/2$ and $b = 1/2$, then the function

$$y = \underbrace{f(x)}_{\text{(name of the function)}} = \underbrace{wx + b}_{\text{(what the function actually is)}} \tag{1}$$

assigns every input x to exactly one real number, namely $wx + b$. In this case, we say the dependency of y on x, i.e., the input-output relationship, is *deterministic*. Note in naming the function, we could have used g or h instead of f. There is nothing special about the letter "f" other than tradition (and also that the word "function" starts with that letter).

In the real world, it is often the case that the dependency of y on x has a degree of uncertainty. This notion is what *conditional probability* aims to capture:

$$\underbrace{p(y \mid x)}_{\text{(name of the conditional probability)}} = \underbrace{\qquad\qquad}_{\text{(what the conditional probability actually is)}} \qquad \text{(Intentionally blank)}$$

Just like for functions, there is nothing special about the letter p; we could've wrote $q(y \mid \mathbf{x})$ or $r(y \mid \mathbf{x})$. Now, why did we leave the right-hand-side blank? We've learned how to "create" functions since grade school mathematics. But most of us never learned the notation for how to do the same for conditional probability. Let's do that here.

Let $n > 0$ be an integer. Consider

$$y = wx + b + \epsilon$$

where w and b are non-random numbers and x is the input, but ϵ is random and defined via the following algorithm:

```
1. Initialize $\epsilon \leftarrow 0$
2. For $t = 1, \ldots, n$, do:
    1. Flip a fair coin
    2. If coin comes up head, then $\epsilon \leftarrow \epsilon + 1$
```

3. Else $\epsilon \leftarrow \epsilon - 1$

3. Return $\epsilon/\sqrt{n}$.

The above procedure defines a dependency of y on x that is random, and hence the input-output relationship is governed by a conditional probability. Let's call the conditional probability

$$\mathcal{S}(y; wx + b, n)$$

(the curly "S" is `\mathcal{S}`). When $n = 2$, we have

$$p(y \mid x) = \mathcal{S}(y; wx + b, 2) = \begin{cases} 1/4 & : \text{if } y = wx + b + \frac{2}{\sqrt{2}}, \\ 1/2 & : \text{if } y = wx + b \\ 1/4 & : \text{if } y = wx + b - \frac{2}{\sqrt{2}}, \\ 0 & : \text{otherwise.} \end{cases}$$

To see there, first note that there are 4 possible outcomes for the coin flip. There is only one outcome in which both coins come up heads, then $y = wx + b + \frac{2}{\sqrt{2}}$. Thus, the probability is 1/4. The same logic applies to the $y = wx + b - \frac{2}{\sqrt{2}}$ case. Now, there are two outcomes that results in $y = wx + b$: one coin comes up head while the other comes up tail. Hence the probability of this case is $2/4 = 1/2$. It is impossible for y to equal to any other values.

The exercise: calculate $p(y \mid x) = \mathcal{S}(y; wx + b, 3)$. *Hint: introduce a notation for denoting outcomes of the coin flips.*

Answer:
In one case, three coins will be heads, in another case three coins will be tail. All possible combinations of the three coins is 3! (six different cases). here are eight different scenarios, hence we have

$$p(y \mid x) = \mathcal{S}(y; wx + b, 3) = \begin{cases} 1/8 & : \text{if } y = wx + b + \frac{3}{\sqrt{3}}, \\ 3/8 & : \text{if } y = wx + b + \frac{1}{\sqrt{3}}, \\ 3/8 & : \text{if } y = wx + b - \frac{1}{\sqrt{3}}, \\ 1/8 & : \text{if } y = wx + b - \frac{3}{\sqrt{3}}, \\ 0 & : \text{otherwise.} \end{cases}$$

---

[2] point(s) — part b: This is a programming exercise. See `hw2.ipynb`

---

[2] point(s) — part c: This is a programming exercise. See `hw2.ipynb`

2 The Gaussian distribution

[1] point(s) — part a: Consider linear regression when the data live in the 2-dimensional space. That is, the data vectors $\mathbf{x}^{(i)} \in \mathbb{R}^2$ are 2-dimensional vectors, and the labels $y \in \mathbb{R}$ are numbers.

In lecture 2, we saw that for linear regression, the relationship between the sample (input) $\mathbf{x}$ and the label (output) y is governed by

$$p(y \mid \mathbf{x}) = \mathcal{N}(y; \mathbf{w}^\top \mathbf{x} + b, \sigma^2) \tag{2}$$

where $\mathcal{N}(\,\cdot\,; \mu, \sigma^2)$ is the probability density function for a Gaussian random variable with mean μ and variance σ^2. The formula for $\mathcal{N}(\,\cdot\,; \mu, \sigma^2)$ is given by

$$\mathcal{N}(z; \mu, \sigma^2) := \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2}\frac{(z-\mu)^2}{\sigma^2}\right).$$

Let $\sigma = 1/\sqrt{2}$, $\mathbf{w} = \begin{bmatrix}0\\1\end{bmatrix}$ and $b = 3$. Compute $p(y \mid \mathbf{x})$ when $\mathbf{x} = \begin{bmatrix}1\\-2\end{bmatrix}$. The only unknown variable in your solution should be y. Type your answer in a single line of LaTeX code using fewer than 50 characters. You don't need to show your work.

Answer:

$$p(y \mid \begin{bmatrix}0\\1\end{bmatrix}) = \frac{1}{\sqrt{\pi}} \exp\left(-y^2 + 2y - 1\right)$$

---

[1] point(s) — part b: Plug in $y = 1$ and $y = -1/2$ into your answer above. Compute the numerical value of $\frac{p(1|\mathbf{x})}{p(-1/2|\mathbf{x})}$ and write the result below rounded to three decimal places (i.e., like 1.234)

Answer:

$$\frac{p(1 \mid \mathbf{x})}{p(-1/2 \mid \mathbf{x})} = \frac{0.564}{0.0595} = 9.478$$

---

[1] point(s) — part c: This is a programming exercise. See `hw2.ipynb`

---

[2] point(s) — part d: "$p(y \mid \mathbf{x}) = \mathcal{N}(y; \mathbf{w}^\top \mathbf{x} + b, \sigma^2)$" is what is known as a probability *density*: $p(y \mid \mathbf{x})$ is not the probability of observing exactly y, but rather the probability of observing a value *near* y. The number in the output cell of part c should be similar to the number in your answer in part b. Explain this why these two numbers are similar.

Answer:
In part a and b, the ratio between certain values of y ($y = 1$ and $y = -\frac{1}{2}$) is calculated using the Gaussian Probability Density Function (PDF).

On the other hand, random numbers which follows a normal distribution (see numpy.random.normal reference)are generating with the same mean and variance than part a and b in part c. After that, we estimate the previous ratio.

So, **we calculate directly the ratio using the Gaussian PDF (parts a and b)** and the same time, **we are generating random samples from the same conditions** (distribution, mean, variance) **in order to estimate the same ratio (part c)**. They are not equal because part c is a simulation.

3 A nonlinear regression problem

[3] point(s) — part a: Consider samples $x \in \mathbb{R}$ and labels $y \in \mathbb{R}$. Let $\boldsymbol{\theta} = \begin{bmatrix}w_1\\w_2\end{bmatrix}$ and let $f(x; \boldsymbol{\theta}) = \sin(w_1 x) + \cos(w_2 x)$. Calculate the gradient $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ of the function

$$J(\boldsymbol{\theta}) = J\left(\begin{bmatrix}w_1\\w_2\end{bmatrix}\right)^2 = \frac{1}{N}\sum_{i=1}^{N}(y^{(i)} - f(\mathbf{x}^{(i)}; \boldsymbol{\theta}))^2.$$

Answer:

$\frac{\partial J}{\partial w_1} = \frac{1}{N}\sum_{i=1}^{N} 2(y^{(i)} - \sin(w_1 x) - \cos(w_2 x)) \cdot (-x\cos(w_1 x)) \rightarrow$

$\mathbf{\frac{\partial J}{\partial w_1} = -\frac{2}{N}\sum_{i=1}^{N}(y^{(i)} - \sin(w_1 x) - \cos(w_2 x)) \cdot x\cos(w_1 x)}$

$\frac{\partial J}{\partial w_2} = \frac{1}{N}\sum_{i=1}^{N} 2(y^{(i)} - \sin(w_1 x) - \cos(w_2 x)) \cdot (x\sin(w_2 x)) \rightarrow$

$\mathbf{\frac{\partial J}{\partial w_2} = \frac{2}{N}\sum_{i=1}^{N}(y^{(i)} - \sin(w_1 x) - \cos(w_2 x)) \cdot x\sin(w_2 x)}$

---

[1] point(s) — part b: This is a programming exercise. See `hw2.ipynb`

---

[4] point(s) — part c: Run gradient descent for 100 steps with constant step size = 1/4. Compare two different initialization: initialization "A"

$$w_1 = -2.5, \quad w_2 = 3.56$$

and initialization "B"

$$w_1 = -2.5, \quad w_2 = 3.57.$$

Answer the following questions:

1. Which initialization led to better performance in terms of the training error?
2. Is one of the converged final parameter (nearly) optimal? How do you know?

For this part, only your response below will be graded. We will not take into consideration of the contents in the python notebook.

Answer:

1. **The initialization "B" led to better performance in terms of the training error**, as the MSE of the initialization "B" after 100 iterations is 0.0075086880887049725 compared to the MSE of the initialization "A" after 100 iterations, which is 1.0071568228127146.

$$\text{MSE (initialization "B")} < \text{MSE (initialization "A")}$$

MSE quantifies the difference between the true value and the predicted values generated by the model. For this reason, initialization "B" led to better performance, because it minimized the loss function better than initialization "A".

2. **The initialization "B" converged better than initialization "A" because its MSE is lower**. As I mentioned earlier, the MSE of the initialization "B" after 100 iterations is 0.0075086880887049725, being a value close to zero. If our MSE is zero, our model predict every single data point perfectly. However, data is often noisy, incomplete, or contains random variations that make it nearly impossible to achieve a MSE equal to zero. I obtained the MSE using the function J() given.

[10 bonus] point(s) — part d: Explain in as much quantitative detail as possible why these two initialization led to different results. If you wish to use figures, you may use
`\includegraphics[width=0.5\textwidth]{myfigure.png}`
Do not submit the figures separately.

For this part, only your response below will be graded. We will not take into consideration of the contents in the python notebook.

Answer:
As I mentioned previously, these two initializations led to the next results using gradient descent :

- Initialization "A": Its **MSE is 1.0071568228127146** after 100 iterations.
- Initialization "B": its **MSE is 0.0075086880887049725** after 100 iterations.

To understand why both values differ, **we must understand how our algorithm works.** Gradient descent converge to a minimum due to it "descend" through the function using negative gradients in each iteration. However, this algorithm may converge to a local minimum, especially when the loss function is non-convex. We can know if a function is non-convex observing its plot **(see figure 1)** or studying its second-derivative.

A non-convex function has local and global minimums. The main difference between them is that a global minimum is the smallest value of a function over its entire curve, while a local minimum is the smallest value of a function within a specific interval.

As I mentioned earlier, **this algorithm may converge to a local minimum and "get stuck" in one of them,** so it will not find a smaller value because it is in a local minimum. **This happens in the initialization A (see figure 2)**. Our algorithm "descends" though the function and gets stuck when it reaches a local minimum.

However, **the situation is totally different in the initialization B (see figure 3)**. It reaches a global minimum, which is why its MSE is smaller than the previous one.

In conclusion, **the convexity of our function** affects whether the algorithm converges to a local minimum or global minimum (the minimum will be global in convex function). If our function is non-convex, **our starting point is fundamental,** because if it is close to a local minimum our algorithm may be "pulled" toward that local minimum.

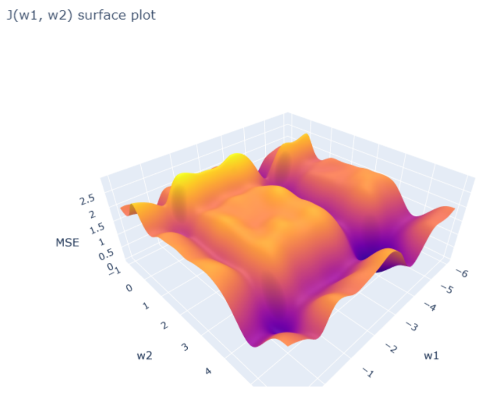

Figure 1: $sin(w_1 x) + cos(w_2 x)$

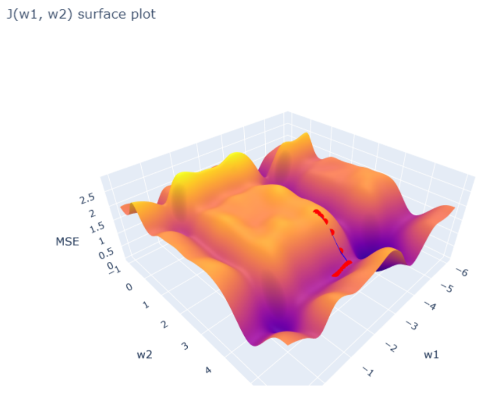

Figure 2: Descent for initialization A

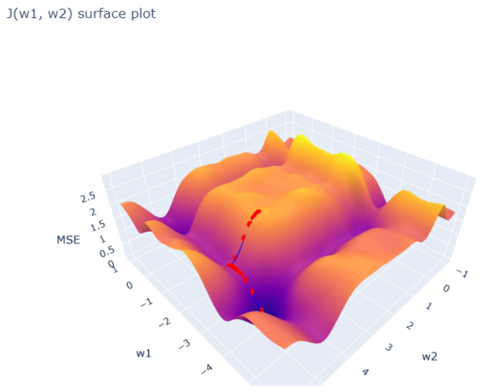

Figure 3: Descent for initialization B