Mathematical Foundations of Deep Neural Networks, M1407.001200
E. Ryu
Fall 2022

Homework 10
Due 5pm, Wednesday, November 16, 2022

Problem 1: *Log-derivative trick for VAE.* Let $Z \in \mathbb{R}^k$ be a random variable. Let $q_\phi(z)$ be a probability density function for all $\phi \in \mathbb{R}^p$. Assume $q_\phi(z)$ is differentiable in ϕ for all fixed $z \in \mathbb{R}^k$. Let $h \colon \mathbb{R}^k \to \mathbb{R}$ satisfy $h(z) > 0$ for all $z \in \mathbb{R}^k$. Assume that the order of integration and differentiation can be swapped. Show

$$\nabla_\phi \mathbb{E}_{Z\sim q_\phi(z)} \left[\log\left(\frac{h(Z)}{q_\phi(Z)}\right)\right] = \mathbb{E}_{Z\sim q_\phi(z)} \left[(\nabla_\phi \log q_\phi(Z)) \log\left(\frac{h(Z)}{q_\phi(Z)}\right)\right].$$

Hint. Since $q_\phi(z)$ is a probability density function,

$$\int \nabla_\phi q_\phi(z)\, dz = \nabla_\phi \int q_\phi(z)\, dz = \nabla_\phi 1 = 0.$$

Problem 2: *Projected gradient method.* Consider the optimization problem

$$\begin{array}{ll} \underset{x\in\mathbb{R}^n}{\text{minimize}} & f(x) \\ \text{subject to} & x \in C, \end{array}$$

where $C \subset \mathbb{R}^n$. Constrained optimization problems of this type can be solved with the *projected gradient method*

$$x^{k+1} = \Pi_C(x^k - \alpha \nabla f(x^k)),$$

where Π_C is the projection onto C. The projection of $y \in \mathbb{R}^n$ onto $C \subseteq \mathbb{R}^n$ is defined as the point in C that is closest to y:

$$\Pi_C(y) = \underset{x\in C}{\operatorname{argmin}} \|x - y\|^2.$$

For the particular set

$$C = \{x \in \mathbb{R}^2 \,|\, x_1 = a,\, 0 \le x_2 \le 1\},$$

where $a \in \mathbb{R}$, show that

$$\Pi_C(y) = \begin{bmatrix} a \\ \min\{\max\{y_2, 0\}, 1\} \end{bmatrix},$$

where $y = (y_1, y_2)$.

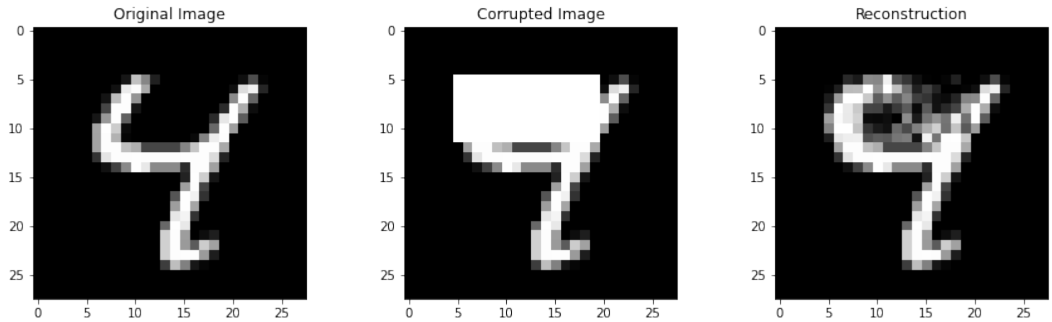

Figure 1: The original, corrupted, and inpainted MNIST image.

Problem 3: *Image inpainting with flow models.* Assume we have a trained flow model that we use to evaluate the likelihood function p. (Since we will not further train or update the flow model, we supress the network parameter θ and write p rather than p_θ.) The starter code `flow_inpainting.py` loads a NICE flow model pre-trained on the MNIST dataset saved in `nice.pt`. Let $X_{\text{true}} \in \mathbb{R}^{28\times 28}$ be an MNIST image with pixel intensities normalized to be in $[0,1]$. Let $M = \{0,1\}^{28\times 28}$ be a binary mask. We measure $M \odot X_{\text{true}}$, where $\odot$ denotes elementwise multiplication, and the goal is to inpaint the missing information $(1-M)\odot X_{\text{true}}$, where $1 - M \in \{0,1\}^{28\times 28}$ is the inverted mask. (See Figure 1.) Perform inpainting by solving the following constrained maximum likelihood estimation problem

$$\begin{array}{ll} \underset{X\in\mathbb{R}^{28\times 28}}{\text{minimize}} & -\log p(X) \\ \text{subject to} & M \odot X = M \odot X_{\text{true}} \\ & 0 \leq X \leq 1, \end{array}$$

where $0 \leq X \leq 1$ is enforced elementwise. Use the projected gradient method with learning rate 10^{-3} and 300 iterations.

Hint. Represent the optimization variable with

```
X = image.clone().requires_grad_(True)
```

while preserving `image`, the tensor containing the corrupted image. When manipulating `X` in the projection step, manipulate `X.data` rather than `X` itself so that the computation graph is not altered by the projection step. Use `clamp(...)` to enforce the $0 \leq X \leq 1$ constraint.

Remark. The optimization problem can be interpreted as finding the most likely reconstruction consistent with the measurements.

Remark. The NICE paper [2] obtains better inpainting results by using a learning rate scheduler (iteration-dependent stepsize) and adding noise to escape from local minima.

Problem 4: *Ingredients of Glow [1].* Let

$$A = PL(U + \mathrm{diag}(s)) \in \mathbb{R}^{C\times C},$$

where $P \in \mathbb{R}^{C\times C}$ is a permutation matrix, $L \in \mathbb{R}^{C\times C}$ is a lower triangular matrix with unit diagonals, $U \in \mathbb{R}^{C\times C}$ is upper triangular with zero diagonals, and $s \in \mathbb{R}^C$. To clarify, $L_{ii} = 1$ for $i = 1, \dots, C$, $L_{ij} = 0$ for $1 \le i < j \le C$, and $U_{ij} = 0$ for $1 \le j \le i \le C$.

(a) Let $f_1(x) = Ax$. Show

$$\log\left|\frac{\partial f_1}{\partial x}\right| = \sum_{i=1}^{C} \log|s_i|.$$

(b) Given $h\colon \mathbb{R}^{a\times b\times c} \to \mathbb{R}^{a\times b\times c}$, define

$$\left|\frac{\partial h(X)}{\partial X}\right| = \left|\frac{\partial(h(X).\mathrm{reshape}(abc))}{\partial(X.\mathrm{reshape}(abc))}\right|,$$

i.e., we define the absolute value of the Jacobian determinant with the input and output tensors vectorized. Note that the reshape operation, which maps elements from the tensor in $\mathbb{R}^{a\times b\times c}$ to the elements of the vector in $\mathbb{R}^{abc}$, is not unique. Show that the definition of $\left|\frac{\partial h(X)}{\partial X}\right|$ does not depend on the specific choice of reshape.

(c) Let $f_2(X \mid P, L, U, s)$ be the 1×1 convolution from $\mathbb{R}^{C\times m\times n}$ to $\mathbb{R}^{C\times m\times n}$ with filter $w \in \mathbb{R}^{C\times C\times 1\times 1}$ defined as

$$w_{i,j,1,1} = A_{i,j}, \qquad \text{for } i = 1, \dots, C,\ j = 1, \dots, C.$$

So $X \in \mathbb{R}^{C\times m\times n}$ and $f_2(X \mid P, L, U, s) \in \mathbb{R}^{C\times m\times n}$. (Assume the batch size is 1.) Show

$$\log\left|\frac{\partial f_2(X \mid P, L, U, s)}{\partial X}\right| = mn\sum_{i=1}^{C} \log|s_i|.$$

(d) Consider the following coupling layer from $X \in \mathbb{R}^{2C\times m\times n}$ to $Z \in \mathbb{R}^{2C\times m\times n}$:

$$\begin{aligned} Z_{1:C,:,:} &= X_{1:C,:,:} \\ Z_{C+1:2C,:,:} &= f_2(X_{C+1:2C,:,:} | P, L(X_{1:C,:,:}), U(X_{1:C,:,:}), s(X_{1:C,:,:})), \end{aligned}$$

where P is a fixed permutation matrix, $L(\cdot)$ outputs lower triangular matrices with unit diagonals in $\mathbb{R}^{C\times C}$, $U(\cdot)$ outputs upper triangular matrices with zero diagonals in $\mathbb{R}^{C\times C}$, and $s(\cdot) \in \mathbb{R}^C$. Show

$$\log\left|\frac{\partial Z}{\partial X}\right| = mn\sum_{i=1}^{C} \log|s_i|.$$

Remark. Given any $A \in \mathbb{R}^{n\times n}$, a decomposition $A = PL(U + \mathrm{diag}(s))$ can be computed via the so-called PLU factorization, which performs steps analogous to Gaussian elimination.

Problem 5: *Gambler's ruin.* You are a gambler at a casino with a starting balance of 100$. You will play a game in which you bet 1$ every game. With probability 18/37, you win and collect 2$ (so you make a 1$ profit). With probability 19/37, you lose and collect no money. You play until you reach a balance of 0$ or 200$ or until you play 600 games. Write a Monte Carlo simulation with importance sampling to estimate the probability that you leave the casino with 200$. Specifically, simulate playing up to 600 games until you reach the balance of 0$ or 200$ and repeat this $N = 3000$ times.

Hint. Regardless of the outcome, simulate $K = 600$ games. The outcomes of the games form a sequence of Bernoulli random variables with probability mass function

$$f(X_1, \ldots, X_K) = \prod_{i=1}^{K} p^{X_i}(1-p)^{(1-X_i)}$$

and $p = 18/37$. For the sampling distribution, also use a sequence of Bernoulli random variables with probability mass function

$$g(Y_1, \ldots, Y_K) = \prod_{i=1}^{K} q^{Y_i}(1-q)^{(1-Y_i)}$$

but with $q > p$. Try using $q = 0.55$.

Hint. The answer is approximately 2×10^{-6}. Submit Python code that produces this answer.

Problem 6: Solve

$$\begin{array}{ll} \underset{\mu,\sigma\in\mathbb{R}}{\text{minimize}} & \mathbb{E}_{X\sim\mathcal{N}(\mu,\sigma^2)}[X\sin(X)] + \frac{1}{2}(\mu-1)^2 + \sigma - \log\sigma \\ \text{subject to} & \sigma > 0 \end{array}$$

using SGD combined with

(a) the log-derivative trick and

(b) the reparameterization trick.

Hint. Use the change of variables $\sigma = e^{\tau}$ to remove the constraint $\sigma > 0$.

Clarification. Implement SGD in Python and submit the code.

References

[1] D. P. Kingma and P. Dhariwal, Glow: Generative flow with invertible 1x1 convolutions, *NeurIPS*, 2018.

[2] L. Dinh, D. Krueger, and Y. Bengio, NICE: Non-linear independent components estimation, *ICLR Workshop*, 2015.