CSCI-GA.2565 — Homework 3

your NetID here

Version 1.1

Instructions.

- **Due date.** Homework is due **Wednesday, March 19, at noon EST**.
- **Gradescope submission.** Everyone must submit individually at gradescope under `hw3` and `hw3code`: `hw3code` is just python code, whereas `hw3` contains everything else. For clarity, problem parts are annotated with where the corresponding submissions go.
  - **Submitting `hw3`.** `hw3` must be submitted as a single PDF file, and typeset in some way, for instance using LaTeX, Markdown, Google Docs, MS Word; you can even use an OCR package (or a modern multi-modal LLM) to convert handwriting to LaTeXand then clean it up for submission. Graders reserve the right to award zero points for solutions they consider illegible.
  - **Submitting `hw3code`.** Only upload the two python files `hw3.py` and `hw3_utils.py`; don't upload a zip file or additional files.
- **Consulting LLMs and friends.** You may discuss with your peers and you may use LLMs. *However,* you are strongly advised to make a serious attempt on all problems alone, and if you consult anyone, make a serious attempt to understand the solution alone afterwards. You must document credit assignment in a special final question in the homework.
- **Evaluation.** We reserve the right to give a 0 to a submission which violates the intent of the assignment and is morally equivalent to a blank response.
  - `hw3code:` your grade is what the autograder gives you; note that you may re-submit as many times as you like until the deadline. However, we may reduce your auto-graded score if your solution simply hard-codes answers.
  - `hw3:` you can receive 0 points for a blank solution, an illegible solution, or a solution which does not correctly mark problem parts with boxes in the gradescope interface (equivalent to illegibility). All other solutions receive full points, *however* the graders do leave feedback so please check afterwards even if you received a perfect score.
- **Regrades.** Use the grade scope interface.
- **Late days.** We track 3 late days across the semester per student.
- **Library routines.** Coding problems come with suggested "library routines"; we include these to reduce your time fishing around APIs, but you are free to use other APIs.

Version history.

1.0. Initial version.

1.1. Included [`hw3`] and [`hw3code`] tags.

1. On initialization.

Consider a 2-layer network

$$f(\boldsymbol{x}; \boldsymbol{W}, \boldsymbol{v}) = \sum_{j=1}^{m} v_j \sigma\left(\langle \boldsymbol{w}_j, \boldsymbol{x} \rangle\right),$$

where $\boldsymbol{x} \in \mathbb{R}^d$, $\boldsymbol{W} \in \mathbb{R}^{m \times d}$ with rows $\boldsymbol{w}_j^\top$, and $\boldsymbol{v} \in \mathbb{R}^m$. For simplicity, the network has a single output, and bias terms are omitted.

Given a data example $(\boldsymbol{x}, y)$ and a loss function ℓ, consider the empirical risk

$$\widehat{\mathcal{R}}(\boldsymbol{W}, \boldsymbol{v}) = \ell\left(f(\boldsymbol{x}; \boldsymbol{W}, \boldsymbol{v}), y\right).$$

Only a single data example will be considered in this problem; the same analysis extends to multiple examples by taking averages.

(a) [hw3] For each $1 \le j \le m$, derive $\partial \widehat{\mathcal{R}} / \partial v_j$ and $\partial \widehat{\mathcal{R}} / \partial \boldsymbol{w}_j$.

(b) [hw3] Consider gradient descent which starts from some $\boldsymbol{W}^{(0)}$ and $\boldsymbol{v}^{(0)}$, and at step $t \ge 0$, updates the weights for each $1 \le j \le m$ as follows:

$$\boldsymbol{w}_j^{(t+1)} = \boldsymbol{w}_j^{(t)} - \eta \frac{\partial \widehat{\mathcal{R}}}{\partial \boldsymbol{w}_j^{(t)}}, \qquad \text{and} \qquad v_j^{(t+1)} = v_j^{(t)} - \eta \frac{\partial \widehat{\mathcal{R}}}{\partial v_j^{(t)}}.$$

Suppose there exists two hidden units $p, q \in \{1, 2, \ldots, m\}$ such that $\boldsymbol{w}_p^{(0)} = \boldsymbol{w}_q^{(0)}$ and $v_p^{(0)} = v_q^{(0)}$. Prove by induction that for any step $t \ge 0$, it holds that $\boldsymbol{w}_p^{(t)} = \boldsymbol{w}_q^{(t)}$ and $v_p^{(t)} = v_q^{(t)}$.

Remark: as a result, if the neural network is initialized symmetrically, then such a symmetry may persist during gradient descent, and thus the representation power of the network will be limited.

(c) [hw3] Random initialization is a good way to break symmetry. Moreover, proper random initialization also preserves the squared norm of the input, as formalized below.

First consider the identity activation $\sigma(z) = z$. For each $1 \le j \le m$ and $1 \le k \le d$, initialize $w_{j,k}^{(0)} \sim \mathcal{N}(0, 1/m)$ (i.e., normal distribution with mean $\mu = 0$ and variance $\sigma^2 = 1/m$). Prove that

$$\mathbb{E}\left[\left\|\boldsymbol{W}^{(0)} \boldsymbol{x}\right\|_2^2\right] = \|\boldsymbol{x}\|_2^2.$$

Remark: This is similar to `torch.nn.init.kaiming_normal_()`.

Next consider the ReLU activation $\sigma_r(z) = \max\{0, z\}$. For each $1 \le j \le m$ and $1 \le k \le d$, initialize $w_{j,k}^{(0)} \sim \mathcal{N}(0, 2/m)$. Prove that

$$\mathbb{E}\left[\left\|\sigma_r(\boldsymbol{W}^{(0)} \boldsymbol{x})\right\|_2^2\right] = \|\boldsymbol{x}\|_2^2.$$

Hint: linear combinations of Gaussians are again Gaussian! For the second part (with ReLU), consider the symmetry of a Gaussian around 0.

Solution.

2. ResNet.

In this problem, you will implement a simplified ResNet. You do not need to change arguments which are not mentioned here (but you of course could try and see what happens).

(a) [hw3code] Implement a class `Block`, which is a building block of ResNet. It is described in (He et al., 2016) Figure 2.

The input to `Block` is of shape (N, C, H, W), where N denotes the batch size, C denotes the number of channels, and H and W are the height and width of each channel. For each data example $\boldsymbol{x}$ with shape (C, H, W), the output of `block` is

$$\texttt{Block}(\boldsymbol{x}) = \sigma_r\left(\boldsymbol{x} + f(\boldsymbol{x})\right),$$

where σ_r denotes the ReLU activation, and $f(\boldsymbol{x})$ also has shape (C, H, W) and thus can be added to $\boldsymbol{x}$. In detail, f contains the following layers.

i. A `Conv2d` with C input channels, C output channels, kernel size 3, stride 1, padding 1, and no bias term.
ii. A `BatchNorm2d` with C features.
iii. A ReLU layer.
iv. Another `Conv2d` with the same arguments as i above.
v. Another `BatchNorm2d` with C features.

Because 3×3 kernels and padding 1 are used, the convolutional layers do not change the shape of each channel. Moreover, the number of channels are also kept unchanged. Therefore $f(\boldsymbol{x})$ does have the same shape as $\boldsymbol{x}$.

Also, implement the option to use SiLU instead of ReLU, and `LayerNorm` instead of `BatchNorm2d`.

Additional instructions are given in doscstrings in `hw3.py`.

(b) [hw3] Explain why a `Conv2d` layer does not need a bias term if it is followed by a `BatchNorm2d` layer.

(c) [hw3code] Implement a (shallow) `ResNet` consists of the following parts:

i. A `Conv2d` with 1 input channel, C output channels, kernel size 3, stride 2, padding 1, and no bias term.
ii. A `BatchNorm2d` with C features.
iii. A ReLU layer.
iv. A `MaxPool2d` with kernel size 2.
v. A `Block` with C channels.
vi. An `AdaptiveAvgPool2d` which for each channel takes the average of all elements.
vii. A `Linear` with C inputs and 10 outputs.

Also, implement the option to use SiLU instead of ReLU, and `LayerNorm` instead of `BatchNorm2d`.

Additional instructions are given in doscstrings in `hw3.py`.

(d) [hw3code] Implement `fit_and_validate` for use in the next part. Please do not shuffle the inputs when batching in this part! The utility function `loss_batch` will be useful. See the docstrings in `hw3.py` and `hw3_util.py` for details.

(e) [hw3] Using `fit_and_validate()`, train a `ResNet` with 16 channels on the data given by `hw3_utils.torch_digits()`, using the cross entropy loss and SGD with learning rate 0.005 and batch size 16, for 30 epochs. Plot the epochs vs training and validation cross entropy losses. Since there is some inconsistency due to random initialization, try 3 runs and have 3 plots. Repeat this for each combination of ReLU/SiLU and `BatchNorm2d`/`LayerNorm`, for a total of 12 plots. Include these 12 plots in your written submission.

Do you notice any significant differences/improvements between the different combinations of activation functions and normalization layers? Include at least one observation in your written submission.

Solution.

3. RBF kernel and nearest neighbors.

(a) [hw3] Recall that given data examples $((\boldsymbol{x}_i, y_i))_{i=1}^n$ and an optimal dual solution $(\hat{\alpha}_i)_{i=1}^n$, the RBF kernel SVM makes a prediction as follows:

$$f_\sigma(\boldsymbol{x}) = \sum_{i=1}^n \hat{\alpha}_i y_i \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{x}_i\|_2^2}{2\sigma^2}\right) = \sum_{i \in S} \hat{\alpha}_i y_i \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{x}_i\|_2^2}{2\sigma^2}\right),$$

where $S \subset \{1, 2, \ldots, n\}$ is the set of indices of support vectors.

Given an input $\boldsymbol{x}$, let $T := \arg\min_{i \in S} \|\boldsymbol{x} - \boldsymbol{x}_i\|_2$ denote the set of closest support vectors to $\boldsymbol{x}$, and let $\rho := \min_{i \in S} \|\boldsymbol{x} - \boldsymbol{x}_i\|_2$ denote this smallest distance. (In other words, $T := \{i \in S : \|\boldsymbol{x} - \boldsymbol{x}_i\| = \rho\}$.) Prove that

$$\lim_{\sigma \to 0} \frac{f_\sigma(\boldsymbol{x})}{\exp\left(-\rho^2/2\sigma^2\right)} = \sum_{i \in T} \hat{\alpha}_i y_i.$$

Remark: in other words, when the bandwidth σ becomes small enough, RBF kernel SVM is almost the 1-nearest neighbor predictor with the set of support vectors as the training set.

(b) [hw3] Consider the XOR dataset:

$$\begin{aligned}
\boldsymbol{x}_1 &= (+1, +1), \quad y_1 = +1, \\
\boldsymbol{x}_2 &= (-1, +1), \quad y_2 = -1, \\
\boldsymbol{x}_3 &= (-1, -1), \quad y_3 = +1, \\
\boldsymbol{x}_4 &= (+1, -1), \quad y_4 = -1.
\end{aligned}$$

Verify that $\hat{\boldsymbol{\alpha}} = (1/\alpha, 1/\alpha, 1/\alpha, 1/\alpha)$ is an optimal dual solution to the RBF kernel SVM, where

$$\alpha = \left(1 - \exp\left(-\frac{\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|_2^2}{2\sigma^2}\right)\right)^2 = \left(1 - \exp\left(-\frac{2}{\sigma^2}\right)\right)^2 > 0.$$

Hint: prove that the gradient of the dual function is $\mathbf{0}$ at $\hat{\boldsymbol{\alpha}}$. Since the dual function is concave, and $\hat{\boldsymbol{\alpha}} > \mathbf{0}$, it follows that $\hat{\boldsymbol{\alpha}}$ is an optimal dual solution.

Remark: in other words, all four data examples are mapped to support vectors in the reproducing kernel Hilbert space. In light of (a), when σ is small enough, $f_\sigma(\boldsymbol{x})$ is almost the 1-nearest neighbor predictor on the XOR dataset. In fact, it is also true for large σ, due to the symmetry of the XOR data.

Solution.

4. LLM Use and Other Sources.

[hw3] Please document, in detail, all your sources, including include LLMs, friends, internet resources, etc. For example:

1a. I asked my friend, then I found a different way to derive the same solution.

1b. ChatGPT 4o solved the problem in one shot, but then I rewrote it once one paper, and a few days later tried to re-derive an answer from scratch.

1c. I accidentally found this via a google search, and had trouble forgetting the answer I found, but still typed it from scratch without copy-paste.

1d. ...

⋮

4. I used my solution to hw1 problem 5 to write this answer.

Solution.

References

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.