# Problem Set 3

Name:

Due Midnight on January 27, 2022

This problem set contains exercises related to neural networks and the NNGP. We use (T) to denote theory exercises and (C) to denote coding exercises. We indicate more involved problems with a *****, and these will be equivalent to solving 2 non-$*$ problems.

**Students should solve 4 problems. Note that $*$ problems are worth 2 normal problems, i.e. solving one $*$ problem means submitting a total 3 problems.**

## Fully Connected Neural Networks

The following problems depend on material in Lecture 4.

**Problem 1 (T).** Let $f_{\mathbf{w}}(x) = ABx$ denote a 1-hidden layer linear neural network with $A \in \mathbb{R}^{1\times k}, B \in \mathbb{R}^{d\times n}$.[1] Given a data set on $n$ samples $(X, y) \in \mathbb{R}^{d\times n} \times \mathbb{R}^{1\times n}$ and a learning rate $\eta > 0$, write out the gradient descent update rules for $A, B$ when $f$ is trained using the squared loss:

$$\mathcal{L}(\mathbf{w}) = \frac{1}{2}\|y - f_{\mathbf{w}}(X)\|_2^2 \ .$$

**Hint.** Start by computing the partial derivatives of the loss with respect to the parameters $A_i$ and then $B_{ij}$ for $i \in [k]$ and $j \in [d]$.[2] Double check that the update rule for $A$ should be of the form:

$$A^{(t+1)} = A^{(t)} + \eta \left(y - A^{(t)}B^{(t)}X\right) X^T {B^{(t)}}^T \ .$$

**Problem 2 (C, *).** For this problem, we will use torchvision data loaders to train a 1-hidden layer ReLU FC-net without bias terms on a standard image dataset. These data loaders serve as an extremely useful API for quickly downloading and running experiments on a variety of standard datasets.

(a) Use the corresponding training and test loaders for the CIFAR10 dataset to load and visualize sample images from CIFAR10 training and test data. You can use `matplotlib` and the `imshow` function to visualize images. Verify that there are $50,000$ training samples and $10,000$ test samples in CIFAR10. The features (images) will be $32\times 32$ color images (3 channels) and the labels will be numbers from 0 through 9 representing different objects. The following link is a useful reference for understanding this dataset.

(b) Train a 1 hidden layer ReLU fully connected network of width 128 to classify images from this dataset and report accuracy on the test set. Note that for training fully connected networks, we need to flatten the images to vectors of size $3 \times 32 \times 32 = 3072$. Also note that the labels come as integers. If you want to use mean squared error for training, you will need to convert these integers into 1-hot vectors (see Lecture 2). To extract the class label for computing accuracy, simply report the index of the predicted vector with maximum value (this is known as argmax).

**Remark.** While may find it useful to use the code from the following Jupyter notebook as a reference, I would highly recommend trying to write everything from scratch and see where you get stuck.

[1] Recall that $\mathbf{w}$ is a vector of all parameters in $A, B$.
[2] Here, we use the notation $[m]$ to denote the set of integers $\{1, 2, \ldots, m\}$.

**Problem 3 (C).** In this problem, we will analyze the effect of initialization on training linear autoencoders. Generate a dataset with one sample $z \sim \mathcal{N}(\mathbf{0}, I_{100\times 100})$.[3] Consider the network, $f_{\mathbf{w}}(x) = ABx$ where $A \in \mathbb{R}^{100\times k}, B \in \mathbb{R}^{k\times 100}$. You may use any width, $k > 1$, of your choice. In this problem, we will train $f$ to autoencode the data point $z$, i.e. we use gradient descent with learning rate $\eta$ to minimize the loss:

$$\mathcal{L}(\mathbf{w}) = \frac{1}{2}\|z - f_{\mathbf{w}}(z)\|_2^2 \ .$$

(a) Initialize $A_{ji}^{(0)} = B_{ij}^{(0)} = 0$ for $i \in [k], j \in [d]$ and train the network for 10 steps. What are the weights after training? Can you explain why?

(b) Initialize $A_i^{(0)} = 0$ and $B_{ij}^{(0)} \overset{i.i.d}{\sim} \mathcal{N}\left(0, \frac{1}{k}\right)$. Try training the network for 1000 steps with various step sizes $\eta \in \{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$. Is $A$ nonzero after training? Report the smallest training error you achieved and the corresponding learning rate.

(c) (Optional) Replace $z$ with an image from the CIFAR10 dataset, and train using the initialization scheme in (b). When you input a random test image into the network what does the output look like? Can you use the result of Problem 1 to prove that the output for any test sample should be a constant times the training sample?

**Problem 4 (C, *).** In this problem, we will empirically investigate the effect of attractors arising in over-parameterized nonlinear autoencoders. Select one image, $z$, from the CIFAR10 dataset and reshape it to a vector of size 3072. Consider the network, $f_{\mathbf{w}}(x) = A\phi(Bx)$ where $A \in \mathbb{R}^{3072\times 1024}, B \in \mathbb{R}^{1024\times 3072}$. In this problem, we will begin by training $f$ to autoencode the data point $z$, i.e. we use gradient descent with learning rate $\eta$ to minimize the loss:

$$\mathcal{L}(\mathbf{w}) = \frac{1}{2}\|z - f_{\mathbf{w}}(z)\|_2^2 \ .$$

(a) Using the default initialization scheme in PyTorch, train the network using gradient descent to near zero training mean squared error (error less than $10^{-5}$).

(b) Visualize the output of the network upon inputting a random training image, test image, random noise, and the all zeros vector. In particular, it will be useful to reshape the output into an image of shape $(3, 32, 32)$ for such visualizations. You may also want to try using the `visdom` library, which makes it easy to visualize images in your browser using the `vis.image` command.

(c) Repeat steps (a) and (b) above when training on 2 images, $z_1, z_2$. How does the output look when you input a test image?

(d) Let $x$ denote a given test image. Let $f$ denote the network trained in (c) and let $f^{(r)}(x)$ denote application of the network $r$ times to $x$, i.e. $f^{(2)}(x) = f(f(x))$. Visualize the following as images: $f(x)$, $f^{(2)}(x)$ , $f^{(20)}(x)$.

(e) Consider $x_1 = z_1 + 10^{-3}\epsilon$ and $x_2 = z_2 + 10^{-3}\epsilon$ where $\epsilon \in \mathcal{N}(0, I_{3072\times 3072})$. Visualize the following as images: $f(x_1)$, $f(x_2)$, $f^{(10)}(x_1)$, and $f^{(10)}(x_2)$.

## Neural Network Gaussian Processes, Dual Activations, and Over-parameterization

The following problems depend on material in Lecture 5.

**Problems 5-7 make use of the definition of the dual activation, which is recalled below.**

**Dual Activations.** Let $\xi \in [-1, 1]$. Recall that for a given activation $\phi(x) : \mathbb{R} \to \mathbb{R}$, the dual activation $\check{\phi} : [-1, 1] \to \mathbb{R}$ is defined by:

$$\check{\phi}(\xi) = \mathbb{E}_{(u,v)\sim\mathcal{N}(\mathbf{0},\Lambda)}[\phi(u)\phi(v)] \quad ; \quad \Lambda = \begin{bmatrix} 1 & \xi \\ \xi & 1 \end{bmatrix}$$

[3]Note that this is a vector in 100 dimensions where each coordinate $z_i$ follows a standard normal distribution.

**Problem 5 (T).** Prove that the dual activation of $\phi(x) = x$ is $\check{\phi}(\xi) = \xi$.

**Hint:** This can be done by directly evaluating the double integral in the expectation. First try integrating over the variable $u$ by completing the square and using facts regarding Gaussian integrals.

**Problem 6 (T, *).** Prove that the dual activation of $\phi(x) = e^{-C^2+Cx}$ is $\check{\phi}(\xi) = e^{-C^2+C^2\xi}$.
**Hint:** Here are two potential approaches:

(1) Directly compute the double integral:

$$\mathbb{E}_{(u,v)\sim\mathcal{N}(\mathbf{0},\Lambda)}[\phi(u)\phi(v)] = \frac{1}{2\pi\sqrt{1-\xi^2}} \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} e^{-C^2+Cu}e^{-C^2+Cv}e^{-\frac{u^2+v^2-2uv\xi}{2(1-\xi^2)}}dudv$$

This integral can be evaluated by using the technique from Lecture 1 and Homework 1. Namely, complete the square in the integrand (for $u$ first and then $v$) and use the fact that the density of the Gaussian has integral 1.

(2) Use the facts that the dual commutes with differentiation, i.e. $(\check{\phi'}) = \check{\phi}'$, that $\check{\phi}(1) = 1$, and that the dual of $a\phi(x)$ is $a^2\check{\phi}$ for constant $a \in \mathbb{R}$. Solving the corresponding differential equation in $\check{\phi}$ gives the result.

**Problem 7 (T, *).** Prove that the dual activation of $\phi(x) = \sin(x)$ is $\check{\phi}(\xi) = e^{-1}\sinh(\xi)$.
**Hint:** Again, we provide two possible approaches.

(1) Use the fact that $\sin(x) = \frac{1}{2i}(e^{ix} - e^{-ix})$ and use the derivations similar to that of Problem 1. In particular, it may be useful to compute:

$$\mathbb{E}_{(u,v)\sim\mathcal{N}(\mathbf{0},\Lambda)}[e^{C_1u}e^{C_2v}] = \frac{1}{2\pi\sqrt{1-\xi^2}} \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} e^{C_1u}e^{C_2v}e^{-\frac{u^2+v^2-2uv\xi}{2(1-\xi^2)}}dudv \quad ;$$

for constants $C_1, C_2$.

(2) Use the fact that $\sin(x) = \frac{1}{2i}(e^{ix} - e^{-ix})$ and then use the closed form for the moment generating function of a Gaussian from Lecture 1.

**Problems 8 and 9 depend on the results (e.g. Theorem 1) from Section 3 of Lecture 5.**

**Problem 8 (T).** Prove the following facts about the dual activation, $\check{\phi} : [-1, 1] \to \mathbb{R}$:

1. $(\check{a\phi}) = a^2\check{\phi}$ for any constant $a \in \mathbb{R}$.
2. $\check{\phi}(\xi)$ is convex and non-decreasing for $\xi \in [0, 1]$.

**Problem 9 (T).** Let $\phi(x) = \sin(x)$ denote the ReLU activation. For $x, \tilde{x} \in \mathcal{S}^{d-1}$ (e.g. $x, \tilde{x}$ on the unit sphere) with $\xi = x^T\tilde{x}$, the dual activation is given by:

$$\check{\phi}(\xi) = e^{-1}\sinh(\xi).$$

Use this to compute the dual activation of $\psi(x) = \cos(x)$.

**Problem 10 (T).** Complete the proof of Proposition 1 in Lecture 4. Namely, let $w \sim \mathcal{N}(\mathbf{0}, I_d)$, $x, \tilde{x} \in \mathbb{R}^d$. Prove that if $u = w^Tx$, $v = w^T\tilde{x}$, then:

$$\mathbb{E}[(u, v)] = (0, 0) \quad ; \quad \mathrm{Cov}((u, v)) = x^T\tilde{x}$$

**For Problems 11 and 12, utilize the following setting:**

For $d = 200, n = 100$, generate data $X \in \mathbb{R}^{n \times d}$ with entries drawn i.i.d. from a standard normal distribution. Normalize $X$ such that each row of $X$ has norm 1. Let $f : \mathbb{R}^d \to \mathbb{R}$ be the function $f(x) = \frac{1}{d}\sum_{i=1}^{d} \sin x_i$, and let $y = f(X) \in \mathbb{R}^n$ where $f(X)_j = f(X_{j,:}^T)$ (i.e. $f$ is applied to each row of $X$). Lastly, for $n_t = 5000$, generate test samples $X_t \in \mathbb{R}^{n_t \times d}$ where the entries are drawn i.i.d. from a standard normal distribution and test labels $y_t = f(X_t) \in \mathbb{R}^{n_t}$. Normalize $X_t$ such that each row of $X_t$ has norm 1. **Make sure to set the random seed in numpy so that experiments are reproducible.**

**Problem 11 (C).** Report the test MSE of using kernel regression to fit the data exactly (i.e. via the `solve` function) with the following NNGP kernels of 1 hidden layer networks:

1. The normalized NNGP of a network with ReLU activation: $\check{\phi}(\xi) = \frac{1}{\pi}\left(\xi(\pi - \arccos(\xi)) + \sqrt{1-\xi^2}\right)$.
2. The normalized NNGP of a network with sine activation: $\check{\phi}(\xi) = \frac{2e}{e^2-1}\sinh(\xi)$.
3. The normalized NNGP of a network with erf activation: $\check{\phi}(\xi) = \frac{1}{\arcsin\left(\frac{2}{3}\right)}\arcsin\left(\frac{2\xi}{3}\right)$.

Compare the above MSEs with the test MSE of solving kernel regression with the Laplace kernel $e^{-L\|x-\tilde{x}\|_2}$ with $L = \frac{1}{200}$.

**Problem 12 (C, *).** Let $f(x) = \frac{\sqrt{2}}{\sqrt{k}}A\phi(Bx)$ denote a 1 hidden layer neural network with $A \in \mathbb{R}^{1 \times k}$, $B \in \mathbb{R}^{k \times d}$, and $\phi : \mathbb{R} \to \mathbb{R}$ be the element-wise ReLU activation (i.e. $\phi(z) = \max(0, z)$). In this problem, we consider the case where the parameters $B_{ij} \overset{i.i.d.}{\sim} \mathcal{N}(0,1)$ and are fixed. For $d = 200$ and $k \in [1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024]$, plot $k$ vs. the test MSE when training only the last layer $A$ to fit the data $(X, y)$. On the same plot, present the test MSE of solving kernel regression with the normalized NNGP of the ReLU network given by

$$\check{\phi}(\xi) = \frac{1}{\pi}\left(\xi(\pi - \arccos(\xi)) + \sqrt{1-\xi^2}\right).$$

How does the test error of the NNGP (corresponding to $k \to \infty$) compare with that of the corresponding finite width models?

## Course Challenge

**Problem 13 (C, *).** Benchmark fully connected networks (architecture of your choosing) and kernel regression with the ReLU NNGP of a 1-hidden layer network on the course challenge.