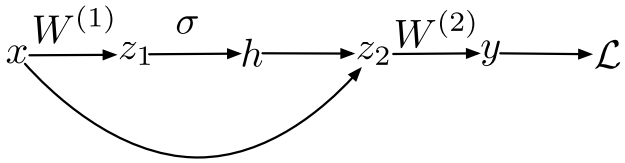
HOMEWORK 3

CSC2515 Fall 2022

- **Deadline:** Tuesday, November 22, 2022 at **19:59**.
- **Submission:** You need to submit two files through MarkUs. One is a PDF file including all your answers and plots. The other is a source file (Python script or Jupyter Notebook) that reproduces your answers. You can produce the file however you like (e.g. LaTeX, Microsoft Word, etc) as long as it is readable. Points will be deducted if we have a hard time reading your solutions or understanding the structure of your code. If the code does not run, you may lose most/all of your points for that question. Your report must include all the relevant figures and graphs. We may not look at your code or Jupyter Notebook.
- **Late Submission:** 10% of the marks will be deducted for each day late, up to a maximum of 3 days. After that, no submissions will be accepted.
- **Collaboration:** You can discuss the assignment with up to two other students (group of three). You can work on the code together. But each of you need to **write your homework report individually**. You must mention the name of your collaborators clearly in the report and the source code.

1. Backpropagation – 20pts.

The goal of this exercise is to help you practice how backpropagation works. We consider a simple variation of the feedforward fully-connected network. In the usual feedforward fully-connected network, each layer is connected to its previous layer. The main difference here is that the second hidden layers is connected to the input too. The computation graph and how each computation is performed is as follows:



$$
\begin{aligned}
z_1 &= W^{(1)}x \quad \text{with } x \in \mathbb{R}^d \\
h &= \sigma(z_1) \quad \text{with } h \in \mathbb{R}^d \\
z_2 &= h + x \\
y &= W^{(2)}z_2 \\
\mathcal{L} &= \frac{1}{2}(y-t)^2 \text{ with } t \in \mathbb{R}.
\end{aligned}
$$

Here σ is the activation function, and you can assume that it is differentiable. Answer the following questions:

(a) **[4pt]** Determine the dimensions of $W^{(1)}$, $W^{(2)}$, z_1, and z_2.

(b) **[2pt]** Calculate the number of parameters in this network, as a function of d. You need to show how you get to the solution.

(c) **[14pt]** Compute the gradient of loss $\mathcal{L}$ with respect to all variables. That is, compute

- $\bar{y} = \frac{\partial \mathcal{L}}{\partial y} = \dots$
- $\bar{W}^{(2)} = \frac{\partial \mathcal{L}}{\partial W^{(2)}} = \dots$
- $\bar{z}_2 = \dots$
- $\bar{h} = \dots$
- $\bar{z}_1 = \dots$
- $\bar{W}^{(1)} = \dots$
- $\bar{x} = \dots$

You need to show all the steps and simplify the solution.

2. Multi-Class Logistic Regression – 10pts.

The goal of this exercise is to verify the formula on Slide 96 of Lecture 3. Consider

$$\begin{aligned}\mathbf{z} &= W\mathbf{x}\\ \mathbf{y} &= \mathrm{softmax}(\mathbf{z})\\ \mathcal{L}_{\mathrm{CE}}(\mathbf{t},\mathbf{y}) &= -\mathbf{t}^\top \log \mathbf{y} = -\sum_{k=1}^{K} t_k \log y_k\end{aligned}$$

Note that if $x \in \mathbb{R}^d$, the dimension of W is $K \times d$. We denote its k-th row by $\mathbf{w}_k$. The vector $\mathbf{y}$ is a function of W and x. And the output $\mathbf{t}$ is a one-hot encoding of the output.

Recall that the k-th component of $\mathbf{y}$ is

$$y_k = \mathrm{softmax}(z_1, \ldots, z_K)_k = \frac{\exp(z_k)}{\sum_{k'=1}^{K} \exp(z_{k'})}.$$

(a) **[5pt]** Compute

$$\frac{\partial y_k}{\partial z_{k'}},$$

for any $k, k' = 1, \ldots, K$ (note that k and k' may or may not be the same). Try to write it in a compact form (no $\exp(\cdots)$ would be needed).

(b) **[5pt]** Compute

$$\frac{\partial \mathcal{L}_{\mathrm{CE}}(\mathbf{t}, \mathbf{y}(\mathbf{x}; W))}{\partial \mathbf{w}_k}.$$

You need to show all the derivations in order to get the full mark. The final solution alone will not give you any mark, as it is already shown on the slide.

3. Some Interpretations of Regularization – 20pts.

We have learned about regularization and how we can use it to control the effective complexity of the hypothesis space. A regularizer allows us to work with a large hypothesis space while avoid overfitting. Previously, we have seen a constraint optimization/geometric interpretation of two commonly used regularizers based on the ℓ_1 and ℓ_2 norms. This exercise helps you gain a better understanding of regularization by providing two new perspectives.

Consider a linear model that takes an input $\mathbf{x} \in \mathbb{R}^d$, maps to the feature space by $\phi : \mathbb{R}^d \to \mathbb{R}^p$, and produces output

$$f(\mathbf{x}; \mathbf{w}) = \phi(\mathbf{x})^\top \mathbf{w}. \tag{3.1}$$

Clearly, $\mathbf{w} \in \mathbb{R}^p$.

Let us consider the regression problem. We are given a dataset $\mathcal{D} = \{(\mathbf{x}^{(i)}, t^{(i)})\}_{i=1}^N$ with $t^{(i)} \in \mathbb{R}$. The regularized least squares objective is

$$\min_{\mathbf{w}} \frac{1}{N} \sum_{i=1}^{N} \left| \phi(\mathbf{x}^{(i)})^\top \mathbf{w} - t^{(i)} \right|^2 + \lambda \mathcal{R}(\mathbf{w}).$$

When $\mathcal{R}(\mathbf{w}) = \|\mathbf{w}\|_2^2$, we get the Ridge regression, and when $\mathcal{R}(\mathbf{w}) = \|\mathbf{w}\|_1$, we get Lasso.

3.1. *Regularizing the Smoothness – 12pt.*

One interpretation of regularization is that it induces an inductive bias over smoother solutions, that is, it encourages the solution of the regularized objective to be smooth. There are different ways to define smoothness. A natural way is to define smoothness based on the derivative of the function w.r.t. its input $\mathbf{x}$.

To see this concretely, consider two functions

$$g_1(x) = \sin(2\pi x), \qquad g_2(x) = \sin(20\pi x), \tag{3.2}$$

defined over $x \in [0, 1]$. Both are bounded in $[-1, +1]$, but the former varies slower than the latter as x changes. This can be seen by taking the derivative of these functions. We get $g_1'(x) = 2\pi \cos(2\pi x)$ and $g_2'(x) = 20\pi \cos(20\pi x)$. The derivative of the first function is bounded between -2π and 2π, while the latter is bounded between -20π and 20π. Comparing these two derivatives validates our intuition that g_2 is less smooth.

The derivatives g_1' and g_2' are both functions of x. If we want to obtain a single number that quantifies the smoothness of g_1 and g_2, we have several choices. A reasonable choice is to look at the average squared magnitude of their derivative over the input domain. That is, we compute

$$\int_0^1 |g'(x)|^2 \mathrm{d}x,$$

for both g_1 and g_2. Soon you shall compute this yourself.

The discussion of smoothness so far has been for the scalar input x. In that case, the derivative is a scalar-valued function. We can extend this idea to multivariate functions $g : \mathbb{R}^d \to \mathbb{R}$. In that case, instead of g', we have $\frac{\partial g(\mathbf{x})}{\partial \mathbf{x}}$, and instead of the squared magnitude $|g'(x)|^2$ as the integrand, we have

$$\left\| \frac{\partial g(\mathbf{x})}{\partial \mathbf{x}} \right\|_2^2.$$

After this introduction, you are ready to define and study the smoothness-based regularizer.

(a) **[1pt]** Compute the partial derivative of the linear model (3.1) w.r.t. its input $\mathbf{x} \in \mathbb{R}^d$:

$$\frac{\partial f(\mathbf{x}; \mathbf{w})}{\partial \mathbf{x}}.$$

Note that for a fixed $\mathbf{x}$ and $\mathbf{w}$, the dimension should be d.

(b) **[1pt]** Compute

$$\left\| \frac{\partial f(\mathbf{x}; \mathbf{w})}{\partial \mathbf{x}} \right\|_2^2.$$

As a sanity check, this is a real-valued function, that is, a function from $(\mathbf{x}, \mathbf{w}) \to \mathbb{R}$.

(c) **[2pt]** Define the smoothness regularizer for function $f(\mathbf{x}; \mathbf{w})$ (3.1) as

$$\mathcal{R}(\mathbf{w}) = \int_{\mathcal{X}} \left\| \frac{\partial f(\mathbf{x}; \mathbf{w})}{\partial \mathbf{x}} \right\|_2^2 \mathrm{d}\mathbf{x}.$$

Compute and simplify this expression. Your solution should factorize to terms depending on $\mathbf{w}$ and a term that is independent of $\mathbf{w}$. The latter is in the form of integral. The solution should not be a function of $\mathbf{x}$ anymore.

(d) **[2pt]** If $\phi(\mathbf{x}) = \mathbf{x}$ (identity feature map), write down $\mathcal{R}(\mathbf{w})$ in terms of $\mathbf{w}$ alone. How is it related to the regularizers that we have seen so far in the course? You should assume that $\mathcal{X} = [0, 1]^d$, the d-dimensional cube with side length of 1.

(e) **[2pt]** For the rest of this subsection, consider $\mathcal{X} = [0, 1]$, so the input $\mathbf{x} = x$ is a scalar. As a practice for the next question, compute $\int_0^1 |g_1'(x)|^2 \mathrm{d}x$ and $\int_0^1 |g_2'(x)|^2 \mathrm{d}x$ with g_1 and g_2 defined earlier (3.2). You need to show the major steps of the computation.
Hint: The integral of $\int_0^1 \cos^2(2\pi k x)\mathrm{d}x$ *is equal to* $\frac{1}{2}$*, for any* $k = 1, 2, 3, \ldots$.

(f) **[2pt]** Consider the following feature map based on the Fourier bases:

$$\phi(x) = [\sin(1 \times 2\pi x), \cdots, \sin(k \times 2\pi x), \cdots, \sin(p \times 2\pi x)]^\top.$$

Note that $\phi(x) \in \mathbb{R}^p$. Compute $\mathcal{R}(\mathbf{w})$ for this feature map. You should be able to get a simple form that does not have any matrices involved.
Hint: You may need to compute integrals of the form $\int_0^1 \cos(2\pi k x)\cos(2\pi k' x)\mathrm{d}x$ *with* $k, k' = 1, 2, 3, \ldots,$ *and* $k' \neq k$*. The value of such an integral is zero.*

(g) **[2pt]** What is the interpretation of the regularizer in the previous step? Can you relate it to the smoothness of the basis functions $\sin(k \times 2\pi x)$?

3.2. *Bayesian Interpretation of Regularizer – 8pt.*

We have already seen a statistical interpretation of the squared error loss. It is the MLE estimate with the Gaussian noise model:

$$\mathbb{P}\left\{ y^{(i)} \mid \mathbf{x}^{(i)}, \mathbf{w} \right\} = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{ -\frac{1}{2\sigma^2} \left(y^{(i)} - \mathbf{w}^\top \mathbf{x}^{(i)} \right)^2 \right\}. \tag{3.3}$$

The MLE computes the parameter $\mathbf{w}$ that maximizes the likelihood:

$$\hat{\mathbf{w}}_{\mathrm{MLE}} \leftarrow \operatorname*{argmax}_{\mathbf{w}} \mathbb{P}\{\mathcal{D} | \mathbf{w}\},$$

for data $\mathcal{D} = \{(\mathbf{x}^{(i)}, t^{(i)})\}_{i=1}^N$.

In the Bayesian approach to parameter estimation, we consider the prior over parameters $\mathbf{w}$, and report the posterior $\mathbb{P}\{\mathbf{w}|\mathcal{D}\}$. This is not a single estimated vector $\hat{\mathbf{w}}$ (also referred to as a point estimate), but a distribution over parameters. As an approximation to the Bayesian approach, we can find the mode (or maximum) of the posterior as a single point estimate. This is called the *maximum a posteriori (MAP)* estimate, which is[1]

$$\hat{\mathbf{w}}_{\text{MAP}} \leftarrow \operatorname*{argmax}_{\mathbf{w}} \mathbb{P}\{\mathbf{w}|\mathcal{D}\}. \tag{3.4}$$

According to the Bayes' theorem, we have

$$\mathbb{P}\{\mathbf{w}|\mathcal{D}\} = \frac{\mathbb{P}\{\mathcal{D}|\mathbf{w}\}\mathbb{P}\{\mathbf{w}\}}{\mathbb{P}\{\mathcal{D}\}}.$$

As $\mathbb{P}\{\mathcal{D}\}$ does not depend on $\mathbf{w}$, the optimization problem (3.4) is equivalent to

$$\hat{\mathbf{w}}_{\text{MAP}} \leftarrow \operatorname*{argmax}_{\mathbf{w}} \mathbb{P}\{\mathcal{D}|\mathbf{w}\}\mathbb{P}\{\mathbf{w}\}. \tag{3.5}$$

The prior distribution $\mathbb{P}\{\mathbf{w}\}$ specifies our preference or knowledge about the solution of the estimate *before* seeing the data.

(a) **[2pt]** Suppose that our prior distribution for the parameters $\mathbf{w}$ is a Gaussian distribution with mean zero and covariance $\Sigma_{\mathbf{w}}$, that is

$$\mathbb{P}\{\mathbf{w}\} = \frac{1}{(2\pi)^{p/2}\sqrt{\det(\Sigma_{\mathbf{w}})}} \exp\left(-\frac{1}{2}\mathbf{w}^\top \Sigma_{\mathbf{w}}^{-1}\mathbf{w}\right). \tag{3.6}$$

Consider the same Gaussian noise model for data (3.3) with N data points. Derive the MAP estimate $\hat{\mathbf{w}}_{\text{MAP}}$.
You need to specify enough detail for your derivations. Simplify the solution as much as possible, and remove terms that do not affect the optimization problem. Your solution should be similar to one of the regularized estimators that we had seen before.
Note: Do not confuse the Gaussian distribution of the noise model (3.3) and the Gaussian distribution over the prior (3.6). Soon you will see that they do not both need to be Gaussian.

(b) **[2pt]** Consider $\Sigma_{\mathbf{w}} = \sigma_{\mathbf{w}}^2\mathbf{I}$, where $\mathbf{I}$ is the identity matrix. Use this choice in your answer to the previous question, and simplify.

(c) **[2pt]** We can consider other prior distributions as well. Let us consider the Laplace distribution

$$\mathbb{P}\{\mathbf{w}\} = \frac{1}{2b}\exp\left(-\frac{\|\mathbf{w}\|_1}{b}\right),$$

with $b > 0$. What is its MAP estimate?

(d) **[2pt]** How are the previous MAP estimates (with Gaussian and Laplace priors) related to the regularized estimators we had seen before?

[1] We have not discussed the Bayesian approach by the time this homework is released, but your current knowledge is enough to go through the required derivations for the rest of this exercise.

4. Handwritten Digit Classification – 50 points.

For this question you will build classifiers to label images of handwritten digits. Each image is 8 by 8 pixels and is represented as a vector of dimension 64 by listing all the pixel values in raster scan order. The images are grayscale and the pixel values are between 0 and 1. The labels y are $0, 1, 2, \cdots, 9$ corresponding to which character was written in the image. There are 700 training cases and 400 test cases for each digit; they can be found in a3digits.zip.

Starter code written in Python is provided to help you load the data. A skeleton is also provided for each question that you should use to format your solution. You are welcome to use any other programming language, as long as your code is functional and modular.

Please note that if you are asked to report/compute quantities these should be clearly displayed in your written report. It is not sufficient to simply print these as an output of your code. The same applies to plots and figures.

0. **[2pt]** Load the data and plot the means for each of the digit classes in the training data (include these in your report). Given that each image is a vector of size 64, the mean will be a vector of size 64 which needs to be reshaped as an 8×8 2D array to be rendered as an image. Plot all 10 means side by side using the same scale.

4.1. *K-NN Classifier – 10pt.*

1. **[5pt]** Build a simple K nearest neighbour classifier using Euclidean distance on the raw pixel data.
   (a) For $K = 1$ report the train and test classification accuracy.
   (b) For $K = 15$ report the train and test classification accuracy.
2. **[1pt]** For $K > 1$, K-NN might encounter ties that need to be broken in order to make a decision. Choose any (reasonable) method you prefer and explain it briefly in your report.
3. **[4pt]** Use 10 fold cross validation to find the optimal K in the 1-15 range. You may use the KFold implementation in sklearn. Report this value of K along with the train, validation, and test set classification accuracies, averaged across folds where applicable.

4.2. *Classifiers comparison – 24pt.* In this section, you will design three different classifiers for the provided hand-written digits data set. You are free to implement your own classifier or to use any package you prefer as long as you will provide readable modular code. We recommend exploring well-known packages such as PyTorch, TensorFlow and scikit-learn. Here is the list of classifiers you are to implement

1. **MLP - Neural Network Classifier – 8pt.** Design a Multi-Layer Perceptron Neural Network. We are asking for a fully connected network with one-hot encoding output. Your input layer needs one unit for each pixel; given that your are distinguishing among 10 classes, the output layer will need 10 units. Other than the input and the output layers, you are free to deign the best network that provide the least possible error rate, taking overfitting into consideration.

2. **SVM classifier – 8pt.** Design an SVM classifier. We briefly covered a few kernels in the class. Since you are using external packages, you have a pretty good chance to try kernel beyond what described. Also, there are useful grid search tools that helps you reach the optimal set of hyper-parameters
3. **AdaBoost Classifier – 8pt.** You will have a chance to turn a weak-learner into a strong performing classifier. Again, you have total freedom of the architecture as long as you provide original, readable and modular code.

4.3. *Model Comparison – 14pt.* Briefly summarize the performance of each model, including the K-NN model with the optimal K you found. This will be a good chance to study different ways to measuring a classifier performance. Measuring MSE or error rate is not enough, you need to provide, at least, the following metrics for each classifier:

1. ROC (Receiver Operating Characteristics) curve
2. Confusion Matrix
3. Accuracy
4. Precision
5. Recall

While we did not cover all these metrics in class, it is a very good idea to read and understand what each one measures. A good starting point is perhaps this Wikipedia link Receiver operating characteristic.

Which classifier performed best? Which performed worst? Did this match your expectations? It is important to show good understanding of why a certain classifier did better in the provided dataset. What can you say about the computational cost of each classifier? You need to provide detailed commentary on the results.