CSC 503/SENG 474: Assignment 2

Due on: Sunday, Nov 6th at 23:59 PST
Where: Brightspace (`https://bright.uvic.ca/d2l/home/244947`)

Instructions:

- You must complete this assignment *entirely* on your own. In other words, you should come up with the solution *yourself*, write the code *yourself,* conduct the experiments *yourself*, analyze the results *yourself*, and finally, write it all solely by *yourself*. The university policies on academic dishonesty (a.k.a. cheating) will be taken very seriously.

- This does not mean that you need to go to a cave and self-isolate while preparing the assignment. You are allowed to have high-level discussions with your classmates about the course material. You are also more than welcome to use Piazza or come to office hours and ask questions. If in doubt, ask!— we are here to help.

- If you are still stuck, you can use books and *published* online material (i.e., material that has a fixed URL). However, **you must *explicitly* credit all sources**. **You are also not allowed to copy-paste online materials**. This includes "slightly" adapting the online code to fit your problem. Woe to you if you are caught doing this!

  - Why "if stuck"? Assignments are designed to develop your practical ML skills and make you strong. If you do the assignments well, the project will feel like a piece of cake. So, give your best. But, on the other hand, do not waste a whole week on a single question: if you are stuck on a question for a few days, ask (us) for help!

- If you cannot make it until the deadline, you can use **a maximum of two grace days** per assignment. They are not free, though: **each grace day comes with the 30% mark penalty** (so submitting on Monday evening would reduce your score by 30%; submitting on Tuesday would reduce it by 60%). No other accommodations will be provided unless explicitly approved by the instructor at least 7 days before the deadline.

- **These assignments are supposed to be really hard! Start early!** You will need *at least* two weeks to complete them!

  - If you do not feel challenged enough, please let me know, and I'll think of something.

- Remember: **you will need to gather at least one-third of all points during the assignments to pass the course. If you don't, you will get an F!**

- Make sure to follow the technical requirements outlined below. TAs can (and will) take 50% off your grade if you disregard some of them.

- Be sure that your answers are clear and easy for TAs to understand. They can penalize you if your solutions lack clarity or are convoluted (in a non-algebraic way), even if they are nominally correct.

- We will try to grade your assignments within seven (7) days of the initial submission deadline.

- If you think there is a problem with your grade, you have one week to raise concern after the grades go public. Grading TAs will be holding office hours during those seven days to address any such problems. After that, your grade is set in stone.

Technical matters:

- You must type up your analysis and solutions *electronically* and submit them as a self-containing Jupyter notebook. Jupyter notebooks can contain code, its output, and images. They can also be used to type math and proofs in LaTeX mode.
  - You ***must*** use LaTeX mode to type formulas. Typing `a^2=sqrt(3)+b1` is a pretty good way to lose 50% of your grade for no good reason.
- Each problem should be submitted as a separate file.
- Each file should be named `VNumber_SurnameInitial_N.ipynb`, where `N` is two digit-padded problem number.
  - **Correct:** `V00000000_SmithJ_05.ipynb`.
  - **Incorrect:** `JohnSmith_V12345 Problem 1.ipynb`, `prob1.pdf` etc.
- Zip all `ipynb` files and submit them as `assignment1.zip` to the Brightspace. Do not put Jupyter files into a directory—they have to be in the root directory of the submitted ZIP file. Do not submit RAR, TAR, 7zip, SHAR and whatnot; just use good ol' ZIP. Do not include other files.
  - You can validate your submission by downloading `https://gist.github.com/inumanag/56ae2ca04b7357530c3aeb2b63699fed` and running it as follows: `python3 validate.py hw1.zip`. This script will complain if you violate submission rules. Use it to polish your submission—if you submit something that the script does not like, you risk losing 50% of your grade.
- The first cell of each Jupyter notebook must start with your name and V number. See the attached notebook for the details.
- Your notebook should be organized sequentially according to the problem statement. Use sections (with the appropriate numbers and labels) within the notebook. Figures and relevant code should be placed in the proper location in the document.
- Notebook code ***must be runnable***! Ideally, all answers will be the output of a code cell.
- Make sure that all images are *embedded* within the notebook (i.e., we cannot see an image that points to `C:\WIN311\THISSU~1.BMP`).
- You must use **Python 3** to complete the assignments. Feel free to use NumPy and `pandas` as you find it fit. Use SciPy, `scikit-learn`, and other non-standard libraries **only** when explicitly allowed to do so.
- Your first executable cell should set the random seed to 1337 to ensure the reproducibility of your results. For Numpy/SciPy and `pandas`, use `numpy.random.seed(1337)`; otherwise, use `random.seed(1337)`.

- Document your code! Use either Markdown cells or Python comments to let us know what you have done!
- Finally, **be concise**! We do not appreciate long essays that amount to basically nothing.

This assignment consists of 4 problems. Some are intended only for graduate students (those taking CSC 503), and are labelled as such. Some contain bonus sections: you can use bonus points to improve your overall homework score. Bonus points cannot be transferred to other assignments or the final project. Any graduate-level problem counts as a bonus problem for the undergraduate students.

Some problems are purposefully open-ended. Whatever you think a correct answer is, make sure to support it with code and data.

Problem 1. The American Handwriting [4 points]

Over the years, the U.S. National Institute of Standards and Technology collected digital images of the digits written by high school students and the U.S. Census Bureau employees. These images serve as the basis of the extremely popular MNIST dataset commonly used to benchmark machine learning classifiers.

Wouldn't it be a good idea to play with that dataset? To do so, install Keras and load the MNIST dataset as follows:

```
from keras.datasets import mnist
(train_X, train_y), (test_X, test_y) = mnist.load_data()
```

Now, let's see how we can use neural networks to classify these images.

1. [**Derivative; 0.5 points**] Calculate the gradient of the softmax function:

$$f(\mathbf{x})_i = \frac{e_i^x}{\sum_j e_j^x}$$

2. [**Simple; 1 points**] Implement a simple one-layered neural network from scratch (using only NumPy). The implementation should include the forward propagation (i.e. prediction), and the backpropagation-powered gradient descent for training the network. Feel free to select the number of nodes in the hidden layer yourself (it must be, however, greater than 10; the recommended value is 128). Each hidden node should use the sigmoid activation function. The output layer should use the softmax activation function to produce the final 10 values (probabilities of each digit). You can use either classical or stochastic gradient descent. Learning rate and the number of iterations are also up to you; you can experiment with $\{0.002, 0.02, 0.2\}$ to get a sense of the best learning rate.
   Plot the network's accuracy (or error) on test data as the number of iterations increases. Does it keep raising (falling) with the number of gradient descent iterations?

3. [**Keras; 0.5 points**] Now use Keras to model and train the exact same network. Faster? Slower? Better?

4. [**ReLU; 0.5 points**] This time, use the ReLU activation functions instead of the sigmoids. What happens?

5. [**Dropout; 0.5 points**] Now add two hidden layers. You should end up with a three-layer deep neural network. Use Keras to model the network and add dropout to each layer. Furthermore, use L2 regularization for the training objective. Use cross-validation to select the best regularization and dropout hyperparameters. How much improvement did you get?

6. [**Convolution; 0.5 points**] Use Keras to model and train a simple convolutional network with one convolutional layer, one pooling layer, and one fully connected layer. You are free to pick any hyperparameters you want: play with the data and provide some justification behind your hyperparameter selection. How much improvement did you get, if any?

7. [**Mugshots; 0.5 points**] Display nine images of your choice that were consistently misclassified by all of the above models (if there are no such images, pick at least those that were misclassified most of the time). You can plot an image via `matplotlib.pyplot.imshow(image, cmap='gray')`. Would you be able to classify those images yourself or not? Why?

It goes without saying: for each model, provide the corresponding training and test errors with the metrics of your choice.

Bonus [0.5 points]: Use the leaky ReLU activation function for your artisan neural network you made from scratch ([Simple]). Any improvements?

Problem 2. Support machines [2 points]

We have seen in class how to use support vector machines to perform binary classification. Let's now see how they work in practice by playing with the `sklearn`'s SVM implementation.

1. [**Hard; 0.5 points**] Select all images corresponding to ones and sevens from the MNIST dataset and train a hard-margin SVM classifier that classifies if an image depicts 1 or 7. How good is it? Can it even be done?

2. [**Soft; 0.5 points**] Now train a soft-margin SVM classifier for the same problem. Use cross-validation to select the best value of C. Did you achieve better results?

3. [**Kernel; 0.5 points**] Try the following kernels with the best soft-margin model and see which one is the best:

   (a) polynomial kernel of various degrees;

   (b) Gaussian kernel with various values of σ (also known as *radial basis function* kernel); and

   (c) linear kernel.

So far, we've only used SVMs for binary classification. But did you know that you can use them for multiclass classification as well? One common way to do it is by using all-vs-all (AVA) classification trick: train $n(n-1)$ binary classifiers $f_{i,j}$ that distinguish between the pairs of classes i and j (if you have, of course, n different classes). Then, you can classify an example x as:

$$f(x) = \arg\max_i \sum_j f_{i,j}(x).$$

4. [**AVA; 0.5 points**] Use the best-performing SVM model thus far to implement the multiclass AVA classification for all digits at once. Once done, plot the confusion matrix (heatmap) of all n^2 classifiers.

Again, for each model, make sure to provide the training and test errors in the metrics of your choice.

Problem 3. Pigs, begone! [2 points]

Let's get rid of those annoying spam texts! First, get the email dataset from `http://www.aueb.gr/users/ion/data/enron-spam/preprocessed/enron5.tar.gz`. This dataset consists of short emails (each email is a .txt file) placed in the label directory (spam or ham, where ham is obviously not spam).

Represent each message as a bag of words, and use these bags of words to train your Na ive Bayes classifier to predict if an email is a spam or not.

Use 70/30% split for the training/test data. How good is your simple classifier?

Problem 4. Uber-SVM [2 points; only for CSC 503]

Sometimes it is worth using different slack variables for different classes in the soft-margin SVM formulation. For example, we will use this trick if the datasets are not balanced or if it is more important to correctly classify one class instead of the other. More formally, for a binary classification problem between two classes + and −, we would like to optimize the following function:

$$\min_{w,b} \frac{1}{2}\|w\|^2 + C_+ \sum_{i:\, y_i \text{ is } +} \xi_i + C_- \sum_{i:\, y_i \text{ is } -} \xi_i$$

such that $y_i(w^\top x_i + b) \geq 1 - \xi_i$ and $\xi_i \geq 0$ for any i. Your job is to derive the Lagrangian dual formulation of this problem.