Problem Set 4 **Oke**

CEE 616: Probabilistic Machine Learning *11.06.2025*

Due Nov 20, 2025 at 11:59PM. Submit on Canvas.

Submission instructions

There are two options for submission:

1. JupyterLab Notebook. Please name your notebook as follows:
   `<lastname>-<firstname>-PS4.ipynb`

2. R/Python/MATLAB script *and* PDF document with supporting responses. Your PDF should have complete responses to all the questions (including all the required plots). Your script should be clearly commented, producing all the results and plots you show in your PDF document. The filenames should be in a similar format as described above.

Whenever datasets are provided, be sure to leave the relative path and filenames as originally given in order to ensure that your scripts will run properly. For instance, here, all the data sets are found in the `data` folder. So, when you call `read.csv()`, use the relative path, e.g. `data/Default.csv`. This way, when you submit your work, you need not include the data. I will have the exact same folder and will be able to run all the scripts as the same path will be referenced.

Problem 1 *Neural networks for structured data—regression (18 pts; 5 pts EC)*

(a) Develop a feed-forward neural network model (Model 1) to predict the average house value based on the 8 features in the California housing dataset. Note that the target variable in the scikit-learn implementation is in units of $100,000. As a starting point, you may use the model in `L3a-Regression-ANN.ipynb` for this problem. (Use the `validation_split` argument in your `fit()` call so that the validation loss is obtained from a further partitioning of the training data.)

- **(i)** Summarize or plot the structure of your ANN. *[2]*
- **(ii)** Show the plot of training/validation loss vs. epoch. Explain how you might use this plot to prevent overfitting the model. *[4]*
- **(iii)** Report the performance (e.g. MSE) of your model on the left-out test set. Plot the predictions (`y_pred`) against the observations (`y_true`). *[3]*

(b) Create two alternative ANNs by modifying the hyperparameters (e.g. number of dense layers, number of neurons in each dense layer, activation function, etc.). Call these "Model 2" and "Model 3," respectively.

- **(i)** Briefly discuss how you arrived at Models 2 and 3 (e.g. What were your considerations? What did you explore?) *[2]*
- **(ii)** Summarize or plot the structure of both Models 2 and 3. *[2]*
- **(iii)** Show the plot of training/validation loss vs. epoch for each of these models. *[2]*

(c) Compare the performance of all three models on the left-out test set using a table and one or more plots. State the best-performing model. *[3]*

Extra Credit: Use one of the Keras tuners[1] to find the best hyperparameters within a specified range. Report on any improvements to the best model you identified in part (c). *[5]*

Problem 2 *Neural networks for structured data—classification (8 pts)*

Estimate a neural network to predict the `Default` class (Yes/No) using all three variables `student`, `balance` and `income` in the `data/Default.csv` file. Summarize your model. Report training/validation loss/accuracy plots (or other metrics, e.g. precision/recall), test performance, etc.

Problem 3 *Neural networks for image classification (18 pts; 4pts EC)*

(a) Develop a convolutional neural network to predict handwritten digits from the MNIST dataset. You may use `L3c-CNN-MNIST.ipynb` as a starting point. Your model should be substantially different from the example shown in class. Perform a few experiments on some of the hyperparameters and *optimizers* to arrive at your final model. In particular discuss the impact of the optimizers you implemented in your modeling experiments. *[6]*

(b) Plot the trajectory of the training and validation loss and accuracy. At which epoch do you observe the optimal state of the network? *[3]*

(c) Obtain a classification report and discuss the performance of your model. *[3]*

(d) Plot the confusion matrix using a heat map. *[2]*

(e) Regularization approaches such as dropout can significantly improve the performance of a neural network. Insert one or more dropout layers following each dense layer. (Experiment with rates from 0.1 through 0.5.) Comment on the impact on model performance. *[4]*

Problem 4 *Kernel exploration (15 pts)*

Stationary kernels are Mercer kernels whose values only depend on the elementwise differences between inputs $\boldsymbol{r} = \boldsymbol{x} - \boldsymbol{x}'$. Thus, we can write:

$$\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') = \mathcal{K}(\boldsymbol{r}) \tag{1}$$

Now, we will explore a few stationary kernels in 1D:

(a) Squared exponential (SE) kernel *[5]*

$$\mathcal{K}(r; \sigma^2) = \sigma^2 \exp\left(-\frac{r^2}{2\ell^2}\right) \tag{2}$$

where σ^2 is the variance amplitude parameter and $e \ll$ is the bandwidth or length scale parameter. Plot the SE kernel in the domain $r \in [-5, 5]$, $r \in \mathbb{R}$, for various ℓ and σ^2 values (you can plot these on the same or different axes; but consolidate as much as possible for clarity). How do σ^2 and ℓ affect the shape of the kernel?

(b) Periodic or exponential sine squared kernel *[5]*

$$\mathcal{K}(r; \ell, p) = \sigma^2 \exp\left[-\frac{2}{\ell^2} \sin^2\left(\pi \frac{r}{p}\right)\right] \tag{3}$$

where σ^2 is the amplitude, p is the period and ℓ the length scale. Explore and desribe the impacts of p and ℓ on the periodic kernel function. Generate plots supporting your description. For simplicity, you can assume $\sigma^2 = 1$.

(c) Cosine kernel *[5]*

$$\mathcal{K}(r; p) = \cos\left[2\pi \frac{r}{p}\right] \tag{4}$$

where p is the period. Explore and desribe the impacts of p on the cosine kernel function. Generate plots supporting your description. For simplicity, you can assume $\sigma^2 = 1$.

Problem 5 *Support vector machines (16 pts; 6 pts EC)*

In this problem, you will fit a support vector machine (SVM) in order to predict if a credit card customer will default on their payment based on their income, balance and student status[2].

(a) Convert the `student` variable into an appropriate type (e.g. `category`, `numeric`). *[1]*

(b) Randomly split your dataset into a training set and a test set. *[1]*

(c) Now, find the best model (within reason, and using a feasible approach) for the following kernel choices:

- **i.** linear *[3]*
- **ii.** polynomial *[3]*
- **iii.** radial *[3]*

Hint: Select reasonable parameter ranges within which to perform a grid search.

(d) Compare the performance of the best models for the three kernel choices, using the test set. Tabulate (or plot) the performance metrics. Which of the approaches gives the best performance? (Do not simply consider the accuracy; also observe the precision, recall, AUC, etc) Provide ample justification for your response. *[5]*

Extra Credit: Generate 2D plots the decision boundaries for the models in part (c), using the `income` and `balance` features. *[4]*

[1] Keras tuners: https://keras-team.github.io/keras-tuner/documentation/tuners/
[2] If using the `student` variable proves problematic, you may leave it out of your model.