राष्ट्रीय प्रौद्योगिकी संस्थान सिक्किम

NATIONAL INSTITUTE OF TECHNOLOGY SIKKIM

(An Institute of National Importance, Ministry of Education, Govt. of India)

Weekly Lab Assignment – I

Course Code: CS16201/CS18223/CS22203

1. Design a simplified rule-based system in the context of a smart home thermostat.
   Goal: To create a system that automatically adjusts the temperature based on specific rules.
   Sample Rules:
   - **Rule 1:** If the current time is between 6:00 AM and 8:00 AM and the external temperature is below 65°F, then set the thermostat to 70°F.
   - **Rule 2:** If the current time is between 8:00 AM and 5:00 PM and the external temperature is above 75°F, then set the thermostat to 72°F.
   - **Rule 3:** If the current time is between 5:00 PM and 6:00 AM and the external temperature is below 60°F, then set the thermostat to 65°F.
   - **Rule 4:** If the user is at home and prefers a warmer temperature, then set the thermostat to the user's preferred temperature.
   - **Rule 5:** If the user is away from home, then set the thermostat to an energy-saving temperature (e.g., 60°F in winter, 80°F in summer).
2. Design a simplified rule-based system in the context of a smart home lighting.
   Goal: To create a system that automatically adjusts the light of the house based on specific rules.
   Sample Rules:
   - **Rule 1:** Time of the day (Extract the current system time), dim lights during daytime and bright lights at night.
   - **Rule 2:** Number of people inside the home. If no one is at home, switch off the lights.
   - **Rule 3:** If it is past 11:00 PM, switch off the lights and switch it back on at 6:00 am in dim mode.
3. Write a Python function that simulates a simple MP Neuron. The function should take a list of non-binary inputs (age, salary, number of family members) and predict whether he/she will buy the house or not (binary output). It should return 1 if the sum of the inputs is greater than the threshold, and 0 otherwise. Generate 500 records (synthetic data) and use 70% of dataset to train the MP-Neuron model. Compute the test accuracy of the model on the test (30%) dataset.
4. Modify the basic MP Neuron model and design a Perceptron model to include weights for each input. The neuron should now take an additional argument - a list of weights and compute the weighted sum of the inputs. Determine the output based on whether this weighted sum exceeds the threshold.
5. Create a simple perceptron model to perform binary classification. The model should take two inputs, apply weights, add a bias (as the threshold), and use a step function as the activation function. Test this perceptron on a simple logic gate (AND, OR).

6. Manually adjust the weights and bias of a perceptron to model the XOR logic gate. Discuss why a single-layer perceptron cannot solve the XOR problem effectively. Implement the perceptron learning rule. Use this rule to train a perceptron on a simple dataset (e.g., a linearly separable dataset). Plot the decision boundary before and after training.
7. Investigate the impact of feature scaling and normalization on perceptron training. Use a dataset with varying scales and demonstrate how normalization affects the convergence of the perceptron.

Weekly Lab Assignment – II

Course Code: CS16201/CS18223/CS22203

1. Design a Multi-layer perceptron to make a binary decision on the following datasets. Use 70% of the dataset as training set and 30% as test set.
    a. Adult Census Income dataset: https://www.kaggle.com/datasets/uciml/adult-census-income/ - Predict whether the annual income of the person is >=50K or <50K
    b. Indian Liver Patient Records dataset: https://www.kaggle.com/datasets/uciml/indian-liver-patient-records - Predict whether person needs to be diagnosed or not?
    c. Titanic: Machine Learning from Disaster dataset: https://www.kaggle.com/c/titanic/data - Predict survival on the Titanic

2. Implement a Multi-Layer Perceptron (MLP) to classify handwritten digits from the MNIST dataset. However, restrict the implementation to not use any deep learning library such as TensorFlow or PyTorch, except for data loading. Specifically, you can use libraries or utilities to load and preprocess the MNIST dataset, but all aspects of building and training the neural network should be implemented without relying on external deep learning libraries. Experiment with different architectures by varying the number of hidden layers and neurons to observe their effects on classification performance.
3. Implement an MLP using a neural network library (such as TensorFlow or PyTorch) to classify handwritten digits from the MNIST dataset. Experiment with different numbers of hidden layers and neurons.
4. Modify the MLP from the previous question to use different activation functions (ReLU, Sigmoid, Tanh) in the hidden layers. Evaluate and compare the performance of each model on the same dataset (MNIST).
5. Explore the effect of different learning rates and optimization algorithms (SGD, Adam, RMSprop) on the training performance of an MLP. Use a dataset like CIFAR-10 for image classification. Comment on the generalization gap between the training and test accuracy.
6. Design a set of experiments to perform hyperparameter tuning on an MLP model. Focus on parameters such as the number of hidden layers, number of neurons per layer, learning rate, and batch size. Use tools like GridSearchCV or a library-specific solution for hyperparameter optimization.

राष्ट्रीय प्रौद्योगिकी संस्थान सिक्किम
NATIONAL INSTITUTE OF TECHNOLOGY SIKKIM
(An Institute of National Importance, Ministry of Education, Govt. of India)

Weekly Lab Assignment – III

Course Code: CS16201/CS18223/CS22203

1. Implement a CNN using a framework like TensorFlow or PyTorch to classify images from the CIFAR-10 dataset. Start with a simple architecture with one convolutional layer, one pooling layer, and one fully connected layer.

2. Modify the CNN model from the previous question by adding additional convolutional and pooling layers. Experiment with different architectures and observe the impact on model accuracy.

3. Implement data augmentation techniques and integrate them into the training process of your CNN model. Evaluate the impact of data augmentation on model performance using the CIFAR-10 dataset.

4. Conduct hyperparameter tuning for a CNN model on a chosen dataset. Focus on optimizing parameters like the number of filters, kernel size, learning rate, and epochs. Document the process and results.

5. Compare the performance of different CNN architectures (e.g., AlexNet, VGG, ResNet) on the same image classification task. Discuss the strengths and weaknesses of each architecture.

6. Implement a simple object detection framework using CNNs. Use a dataset like PASCAL VOC for training and evaluating the model. Discuss the challenges involved in adapting CNNs from image classification to object detection.