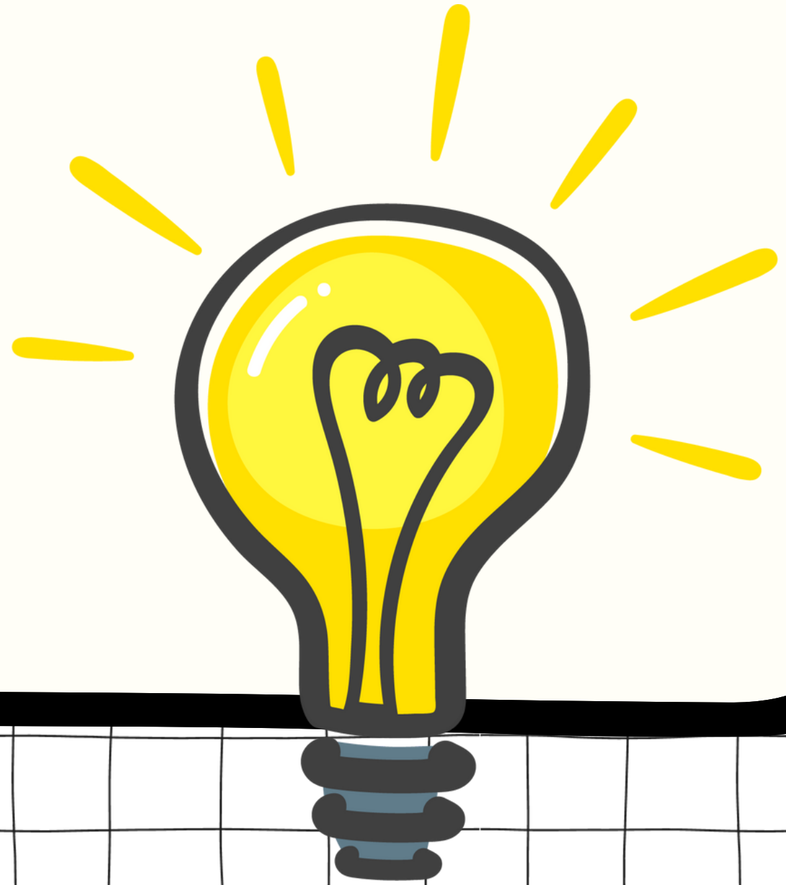
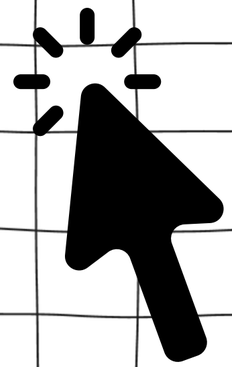
Deep Neural Network for MNIST Classification

Name : Ha Thanh Nga
Class : BA62
Subject : Big Data Analytics



Contents



Data Import :

- Import the necessary libraries and packages.
- Load the dataset, specifying whether it's a TensorFlow dataset and set as_supervised True : load a dataset from Tf dataset (2-tuple structure [input , target])
- Use with_info to gather information about the dataset, such as its version, features, and a sample of the data.

Data Preprocessing:

- Perform data preprocessing, including splitting the data into training and test sets.
- Create a validation dataset, typically by taking a small portion (e.g., 10%) from the training data.
- Cast the number of validation samples to tf.int64 data type for consistency.
- Scale the data to a specific range (e.g., 0 to 1) using float32 data type.
- Use a map function to apply preprocessing to the entire dataset efficiently.

Batching and Shuffling:

- Batch data into smaller subsets, determining the batch size. Common choices include stochastic gradient descent (batch size = 1) and mini-batch gradient descent (e.g., batch size = 100).
- Shuffle the data to present it in a random order, a crucial step for effective training.
- Set the buffer size for shuffling, considering the size of dataset and computational constraints.

Model Outline:

- Define the architecture of machine learning model, specifying the input and output sizes.
- Create hidden layers with appropriate sizes and activation functions, using tools like Keras Sequential.
- Flatten data.
- Apply dense layers to compute dot products and add biases.
- Choose the appropriate activation function based on problem (e.g., linear for regression, sigmoid for binary classification, softmax for multiclass classification).

Optimize and choose the appropriate loss function :

- Configure the model for training by compiling it.
- Select an optimizer, such as Adam, for efficient gradient-based optimization.
- Choose a loss function suitable for your problem (e.g., binary cross-entropy, categorical cross-entropy).
- Define metrics to monitor model performance, often including accuracy.

Training the Model & Improvements

- Specify the number of epochs for training, determining how many times the model will see the entire dataset.
- Train the model by fitting it to the training data, using the validation dataset to monitor performance.
- Adjust the model architecture as needed, including changing the number of hidden layers or their sizes, to improve performance.

Model Evaluation

- After training, evaluate the model's performance using the test dataset.
- Measure loss and relevant metrics, ensuring the model generalizes well to unseen data.
- Prevent overfitting by analyzing the evaluation results.

Thank you