Regularization

September 12, 2024

1 Comparing Neural Networks with and without Regularization on MNIST

1.1 Introduction

We will compare the performance of a simple neural network on the MNIST dataset, with and without the application of L2 regularization. Regularization is a technique used to improve the generalization of machine learning models, helping them perform better on unseen data by preventing overfitting.

We will: - Train a neural network without regularization. - Train a neural network with L2 regularization. - Compare the results in terms of accuracy and loss.

1.2 Loading and Preprocessing the Data

[ ]:
```
import numpy as np
import tensorflow as tf
from tensorflow.keras.datasets import mnist
from tensorflow.keras.utils import to_categorical
import matplotlib.pyplot as plt

# Load the MNIST dataset
(x_train, y_train), (x_test, y_test) = mnist.load_data()

# Normalize the images to [0, 1]
x_train, x_test = x_train / 255.0, x_test / 255.0

# Convert labels to one-hot encoded format
y_train = to_categorical(y_train, 10)
y_test = to_categorical(y_test, 10)

# Show the shape of the data
print(f'Training data shape: {x_train.shape}')
print(f'Test data shape: {x_test.shape}')
```

1.3 Defining the Neural Network Models

We will create two neural network models: - **Model 1**: Without regularization. - **Model 2**: With L2 regularization.

```
[ ]: from tensorflow.keras.models import Sequential
     from tensorflow.keras.layers import Dense, Flatten
     from tensorflow.keras.regularizers import l2

     # Define a function to create the neural network model
     def create_model(with_regularization=False):
         if with_regularization:
             regularizer = l2(0.001)  # L2 regularization with lambda = 0.001
         else:
             regularizer = None

         model = Sequential([
             Flatten(input_shape=(28, 28)),
             Dense(128, activation='relu', kernel_regularizer=regularizer),
             Dense(64, activation='relu', kernel_regularizer=regularizer),
             Dense(10, activation='softmax')
         ])
         return model

     # Create models
     model_no_reg = create_model(with_regularization=False)
     model_with_reg = create_model(with_regularization=True)
```

1.4 Training the Models

Next, we will compile and train both models. We will use the Adam optimizer and categorical crossentropy loss function.

```
[ ]: # Compile the models
     model_no_reg.compile(optimizer='adam', loss='categorical_crossentropy',␣
      ↪metrics=['accuracy'])
     model_with_reg.compile(optimizer='adam', loss='categorical_crossentropy',␣
      ↪metrics=['accuracy'])

     # Train the model without regularization
     history_no_reg = model_no_reg.fit(x_train, y_train, epochs=10, batch_size=64,␣
      ↪validation_split=0.2, verbose=1)

     # Train the model with L2 regularization
     history_with_reg = model_with_reg.fit(x_train, y_train, epochs=10,␣
      ↪batch_size=64, validation_split=0.2, verbose=1)
```

1.5 Evaluating the Models

After training, we will evaluate both models on the test set to compare their performance.

[ ]:
```
# Evaluate both models on the test set
test_loss_no_reg, test_acc_no_reg = model_no_reg.evaluate(x_test, y_test,␣
 ↪verbose=0)
test_loss_with_reg, test_acc_with_reg = model_with_reg.evaluate(x_test, y_test,␣
 ↪verbose=0)

print(f"Test accuracy without regularization: {test_acc_no_reg:.4f}")
print(f"Test accuracy with L2 regularization: {test_acc_with_reg:.4f}")
```

1.6 Visualizing Results

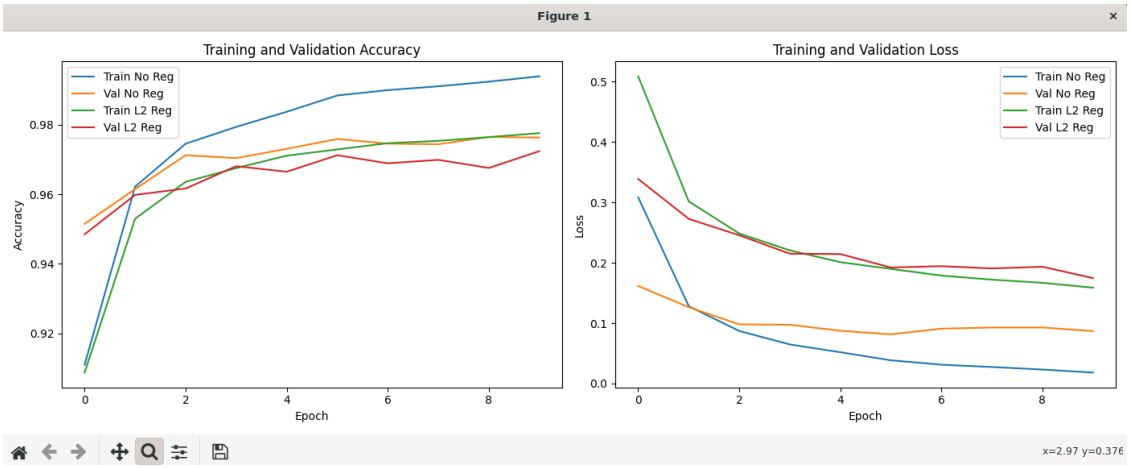
We will plot the training and validation accuracy and loss for both models to observe the effects of regularization.

[ ]:
```
# Plot training and validation accuracy and loss
plt.figure(figsize=(14, 5))

# Accuracy plots
plt.subplot(1, 2, 1)
plt.plot(history_no_reg.history['accuracy'], label='Train No Reg')
plt.plot(history_no_reg.history['val_accuracy'], label='Val No Reg')
plt.plot(history_with_reg.history['accuracy'], label='Train L2 Reg')
plt.plot(history_with_reg.history['val_accuracy'], label='Val L2 Reg')
plt.title('Training and Validation Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()

# Loss plots
plt.subplot(1, 2, 2)
plt.plot(history_no_reg.history['loss'], label='Train No Reg')
plt.plot(history_no_reg.history['val_loss'], label='Val No Reg')
plt.plot(history_with_reg.history['loss'], label='Train L2 Reg')
plt.plot(history_with_reg.history['val_loss'], label='Val L2 Reg')
plt.title('Training and Validation Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()

plt.tight_layout()
plt.show()
```



1.7 Conclusion

From the results, we can observe that the model with L2 regularization shows reduced validation loss compared to the model without regularization, showing better generalization and recognition to new or unseen data. Regularization also showed a slight decrease in accuracy on the trained models due to the restraint on overfitting which allowed for the better recognition on new data.