Aren Ashlock
Department of Computer Science
Iowa State University

April 4, 2024

HW5 — Neural Network

1 Neural Network

1. For each problem below, state which dimensions d, d_H, d_o are determined by the problem, and state the value of each of those dimensions. Similarly, state whether the activation function $g_H(\cdot)$ or $g_o(\cdot)$ is determined by the problem, and if it is determined, state what it should be.

   (a) One wants a neural network that takes a 32×32 RGB-format image and determines which alphanumeric letter (from 'a' through 'z' and '0' through '9') the image depicts.
   Answer: $d = 32 \times 32 \times 3 = 3072, d_H$ is not determined by the problem, $d_o = 26 + 10 = 36$.
   $g_o(\cdot)$ is determined by the problem and should be the softmax activation function.

   (b) Suppose that you are presented with a paragraph of 128 tokens given by a writer. One wants a network that determines whether the writer is happy or sad.
   Answer: $d = 128, d_H$ is not determined by the problem, $d_o = 2$.
   $g_o(\cdot)$ is determined by the problem and should be the sigmoid activation function.

   (c) You want a neural network that predicts the future GPS coordinate pair of a watch given 20 past GPS coordinate pairs.
   Answer: $d = 20 \times 2 = 40, d_H$ is not determined by the problem, $d_o = 2$.
   $g_o(\cdot)$ is determined by the problem and should be the linear activation function (since it is continuous and could be negative).

2. Design a MLP neural network to solve the house price prediction problem (using the same data set we have been using for the first half of the semester). Take all 6 X features as the input and the house price as the output. Use no more than 3 hidden layers. Each hidden layer can have no more than 30 units. Use ReLU as the activation function.

   (a) Show me your code (data loading & normalization, network structure, and training). Use the "NeuralNetwork.ipynb" as an example.

```
# Get the data for the house prices
real_estate_data = pd.read_csv("data/Real_estate.csv")
df_real_estate_data = pd.DataFrame(real_estate_data)

# Data normalization
min_max_scaler = preprocessing.MinMaxScaler()
data = min_max_scaler.fit_transform(df_real_estate_data)
scaled_df = pd.DataFrame(data, columns=df_real_estate_data.columns)

# Get the X tensor (feature inputs) and y tensor (actual cost)
X_tensor = torch.tensor(np.array(scaled_df.iloc[:, 1:-1]), dtype=torch.float)
.unsqueeze(1)
y_tensor = torch.tensor(np.array(scaled_df.iloc[:, -1]), dtype=torch.float)
.unsqueeze(1)

# Create TensorDataset and DataLoader for mini-batch processing
```

```
dataset = TensorDataset(X_tensor, y_tensor)
dataloader = DataLoader(dataset, batch_size=10, shuffle=True)

model = nn.Sequential(
    nn.Flatten(),
    nn.Linear(6, 30),
    nn.ReLU(),
    nn.Linear(30, 30),
    nn.ReLU(),
    nn.Linear(30, 1)
)

criterion = nn.MSELoss()
optimizer = optim.SGD(model.parameters(), lr=0.001)

# Number of epochs (iterations over the entire dataset)
num_epochs = 100

for epoch in range(num_epochs):
    for inputs, targets in dataloader:
        # Forward pass
        outputs = model(inputs)
        loss = criterion(outputs, targets)

        # Backward pass and optimize
        optimizer.zero_grad()  # Clear existing gradients
        loss.backward()  # Compute gradients
        optimizer.step()  # Update parameters
```

(b) How many hidden layers does your MLP model have? How many units does each hidden layer have?

Answer: There are 2 hidden layers, with 30 units each

(c) What is your learning rate? What is your batch size? How many training epochs did you have?

Answer: Learning rate = 0.001, Batch size = 10, Epochs = 100

(d) How many parameters does your MLP model have? Why? Show me the calculation process.

Answer: The input layer has 0 parameters. Then, the first hidden layer has $30*(6+1) = 210$, the second hidden layer has $30*(30+1) = 930$, and the output layer has $1*(30+1) = 31$. Combining all of these, the MLP model has 1171 parameters.

(e) Is your trained model better than, the same with, or worse than the `Multiple Linear Regression` solution you had from your previous homework submission? Why?

Answer: I am confident that is is better than the `Multiple Linear Regression` solution. I cannot show why since I didn't complete that section from HW3, but if I had the R^2 value, I would compare it with the R^2 from this model and the one with a bigger value is better. (I know this isn't acceptable, but it's the best I can do...)

3. Take the "MNIST.ipynb" as a start point:

(a) Execute the given code using torch and `nn.Sequential` model to train a neural network. What is your testing accuracy? (Just tell me a number, no need to show the code, this is for your own convenience to make sure you can run the code)

Answer: 94.05%

(b) Modify the code to train the same model with the test data, and evaluate the accuracy using the training data. What is your accuracy? Show me your modifications (i.e., only the lines of programs that are different from my given code).

```
# Training loop
# (changed from train_loader to test_loader)
for images, labels in test_loader:
# Testing loop - Calculate accuracy
# (changed from test_loader to train_loader)
for images, labels in train_loader:
```

Answer: 86.74%

(c) Consider the following modification of the data set: for every hand written digit image, suppose the corresponding digit is $i \in \{0, ..., 9\}$, change its label (y) to $i\%2$ (i.e., mod(i,2)).

- Show me your code that modifies `y_train` and `y_test` to align with the data set modifications.

  ```
  # Insert in read_images_loader function before return
  for i in range(size):
              labels[i] = labels[i] % 2
  ```

- Using the same MLP provided in the "MNIST.ipynb", what minimum change should you make to have the model work with the modified data set (i.e., what is your number of outputs)?
  Answer: The number of outputs should be 2 $\rightarrow$ change the final nn.Linear(256, 10) to nn.Linear(256, 2)
- What are the one hot encoded outcomes of the labels in the modified data set?
  Answer: $[1, 0]$ maps to 0 and $[0, 1]$ maps to 1. So, we have $0, 2, 4, 6, 8 = [1, 0]$ and $1, 3, 5, 7, 9 = [0, 1]$.

(d) Modify your train data set such that **there are only 10 images left with the label being 3, and 10 images left with the label being 9.**

- Show me your code that makes the above modification.

  ```
  def modified_read_images_labels(self, images_filepath, labels_filepath):
          # Same stuff as the read_images_labels code...
          # (except the return)

          # ---------------- New stuff ----------------

          # empty arrays for new training data
          modifiedLabels = []
          modifiedImages = []

          # counters to ensure limit of images with labels 3 and 9
          label3 = 0
          label9 = 0

          # modify the data to meet parameters
          for i in range(len(labels)):
              if(labels[i] == 3):
                  if(label3 < 10):
                      modifiedLabels.append(labels[i])
                      modifiedImages.append(images[i])
                      label3 += 1
              elif(labels[i] == 9):
                  if(label9 < 10):
                      modifiedLabels.append(labels[i])
                      modifiedImages.append(images[i])
                      label9 += 1
              else:
  ```

```
                modifiedLabels.append(labels[i])
                modifiedImages.append(images[i])

        return modifiedImages, modifiedLabels

    def load_data(self):
        # Only changed the line below to be the new function I made
        x_train, y_train = self.modified_read_images_labels(self.training_images
```

- Using the same MLP provided in the "MNIST.ipynb", and train the model with the modified data set. Evaluate your model's performance with the testing data set. Your model should be performing worse than the original model. **Describe** what changes you could make to improve the model's performance given the modified data set? Undoing your above data set modifications cannot be a solution. Do not include your code, but you are welcome to use experiments or other analyses to help deriving your answer to this question.
  Answer: My solution gives me an accuracy of 76.19%, which is much worse than with the original data. I originally had an incorrect solution since my MLP was performing better (accuracy was 95.25%). This happened when I was modifying both the training and testing data sets, so that is my proposed change to improve the model's performance.

Submitted by Aren Ashlock on April 4, 2024.