Reyhane Shahrokhian 99521361

HomeWork1 of DeepLearning Course

Dr. DavoodAbadi

Question 1:

1-1: to classify news in this question we can use bayes classification. We should compute the probability of both classes(0, 1) and see that it belongs to which class.

Test1:

Class 0:

$P(C = 0 \mid t) = P(C = 0) \cdot P(t_1 \mid 0) \cdot P(t_2 \mid 0) \cdot P(t_3 \mid 0) \cdot P(t_4 \mid 0) = \frac{1}{2}\times\frac{2}{9}\times\frac{2}{9}\times\frac{2}{9}\times\frac{1}{9} \simeq 0.000609$

Class 1:

$P(C = 1 \mid t) = P(C = 1) \cdot P(t_1 \mid 1) \cdot P(t_2 \mid 1) \cdot P(t_3 \mid 1) \cdot P(t_4 \mid 1) = \frac{1}{2}\times\frac{1}{8}\times\frac{1}{8}\times\frac{1}{8}\times\frac{2}{8} \simeq 0.000244$

⇒ so test1 belongs to class 0.

Test2:

Class 0:

$P(C = 0 \mid t) = P(C = 0) \cdot P(t_1 \mid 0) \cdot P(t_2 \mid 0) \cdot P(t_3 \mid 0) \cdot P(t_4 \mid 0) \cdot P(t_5 \mid 0) = \frac{1}{2}\times\frac{2}{9}\times\frac{2}{9}\times\frac{2}{9}\times\frac{1}{9}\times 0 = 0$

Class 1:

$P(C = 1 \mid t) = P(C = 1) \cdot P(t_1 \mid 1) \cdot P(t_2 \mid 1) \cdot P(t_3 \mid 1) \cdot P(t_4 \mid 1) \cdot P(t_5 \mid 1) = \frac{1}{2}\times\frac{1}{8}\times\frac{1}{8}\times\frac{1}{8}\times\frac{2}{8}\times 0 = 0$

⇒ probability of both classes computed as 0 due to the non-existence of one of the word in our train vocabulary.

1-2: To solve the problem that occurs for test2 in 1-1, we can use laplace smoothing to avoid 0 in the multiplication. We assume alpha as 1.

Everything with this method is the same but the value of P(t_i| x) should be computed as:

$$P(t_i \mid x) = \frac{count\ t_i\ in\ class\ x + alpha}{total\ word\ count + alpha \times vocab-size}$$

Class 0:

$total\ word\ count = 9$, $vocab - size = 6$

$P(C = 0 \mid t) = P(C = 0) \cdot P(t_1 \mid 0) \cdot P(t_2 \mid 0) \cdot P(t_3 \mid 0) \cdot P(t_4 \mid 0) \cdot P(t_5 \mid 0) = \frac{1}{2} \times \frac{3}{15} \times \frac{3}{15} \times \frac{3}{15} \times \frac{2}{15} \times \frac{1}{15} = 0.0000355$

Class 1:

$total\ word\ count = 8$, $vocab - size = 6$

$P(C = 1 \mid t) = P(C = 1) \cdot P(t_1 \mid 1) \cdot P(t_2 \mid 1) \cdot P(t_3 \mid 1) \cdot P(t_4 \mid 1) \cdot P(t_5 \mid 1) = \frac{1}{2} \times \frac{2}{14} \times \frac{2}{14} \times \frac{2}{14} \times \frac{3}{14} \times \frac{1}{14} = 0.0000223$

⇒ so this test belongs to class 0.

Question 3:

Probit model is a type of regression for binary classification. And we have:

$P(Y = 1 \mid X) = \phi(X^T\beta)$, $P(Y = 0 \mid X) = 1 - \phi(X^T\beta)$

In which β is a vector of coefficients.

⇒ likelihood for single observation:

$$L(\beta; x_i, y_i) = \phi(x_i^T\beta)^{y_i} \cdot (1 - \phi(x_i^T\beta))^{(1-y_i)}$$

⇒ likelihood for entire sample, we have to product all of the single likelihoods:

$$L(\beta; X, Y) = \prod_{i=1}^{n} [\phi(x_i^T\beta)^{y_i} \cdot (1 - \phi(x_i^T\beta))^{(1-y_i)}]$$

⇒ to compute -log-likelihood, we should take natural logarithm(ln):

$$- \text{log-likelihood} = - \sum_{i=1}^{n} [y_i \ln \phi(x_i^T\beta) + (1 - y_i) \ln(1 - \phi(x_i^T\beta))]$$

As I didn't know anything about probit regression, I've used some information from wikipedia.

Question 4:

4-1:

- If we don't add activation functions, it actually doesn't matter how many layers you added; the output would still be a linear combination of the inputs.
- The network could learn complex patterns and representations in our data with non-linear activation functions.
- With activation functions, problems with vanishing gradients are solved and it allows gradients to flow efficiently.

4-2: As long as that non-linear function meets certain criteria, it can be used. The criteria is as follow:

- It should be continuous.
- It should be differentiable.
- It should be monotonic.
- It is better to be used to be bounded within a specific range.

But it is common to use sigmoid, softmax, tanh, ReLU, Leaky ReLU, etc.

Question 5:

5-1:

- Sigmoid:

  Formula: sigmoid(x) = $\frac{1}{1 + exp(-x)}$

  + Sigmoid maps input x to a value between 0 and 1. It is mostly used in binary classification problems.

  - The problem is that it suffers from the vanishing gradient problem for deep networks and outputs are not centered around zero, which can prevent convergence. Also the output is always positive which causes zig zag path in weights optimization.

- Softmax:

  Formula: softmax(x_i) = $\frac{exp(x_i)}{\sum_{j=1}^{q} exp(x_j)}$

  + Softmax is primarily used in multi-class classification problems. It transforms a vector of scores into a probability distribution over multiple classes so the sum of the probabilities for all classes is equal to 1.

  - As it is generalization of sigmoid, It also suffers from the vanishing gradient problem for deep networks

- ReLU:

  Formula: ReLu(x) = $max(0,\ x)$

  + ReLU is a simple and popular choice. It outputs 'x' if 'x' is positive and 0 otherwise. It's widely used in deep learning models, especially convolutional neural networks (CNNs).

It is simple and computationally efficient, leading to faster training and can save memory and computation. ReLU optimization is really easy as it is similar to linear units. It solves the vanishing gradient problem for positive inputs.

- One negative point is that it prones to the "dying ReLU" problem when neurons output zero for all inputs, effectively stopping their contribution to the learning process. Also as it is not zero-centered, in some situations it may slow down convergence.

- Tanh:

  Formula: Tanh$(x) = \frac{1 - exp(-2x)}{1 + exp(-2x)}$

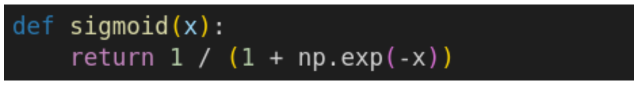
  + Tanh is similar to the sigmoid but maps inputs to the range [-1, 1]. It is often used in scenarios where data should be centered around zero, which can help with faster convergence in some models. It is smooth and differentiable, which makes it suitable for gradient-based optimization.

  - As I mentioned it is similar to sigmoid, so it suffers from vanishing gradient problems for deep networks, although less so compared to sigmoid. It's also expensive to be computed due to exp.

I've searched for pros and cons of tanh and I read some blogs then I write what I get, so I can't insert specific links here:)

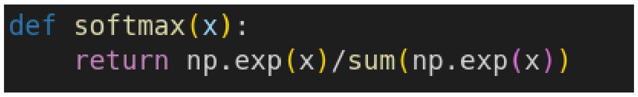
5-2: Activation functions implemented based on their Formula above.

- Sigmoid:



```python
def sigmoid(x):
    return 1 / (1 + np.exp(-x))
```

- Softmax:



```python
def softmax(x):
    return np.exp(x)/sum(np.exp(x))
```

- ReLU:

```
def relu(x):
    return np.maximum(x, 0)
```

- Leaky ReLU:

```
def leaky_relu(x):
    return np.maximum(x, 0.01 * x)
```

  Formula for this wasn't mentioned in the last part of the question. It is:

  Leaky_ReLU$(x) = max(0.01x,\ x)$

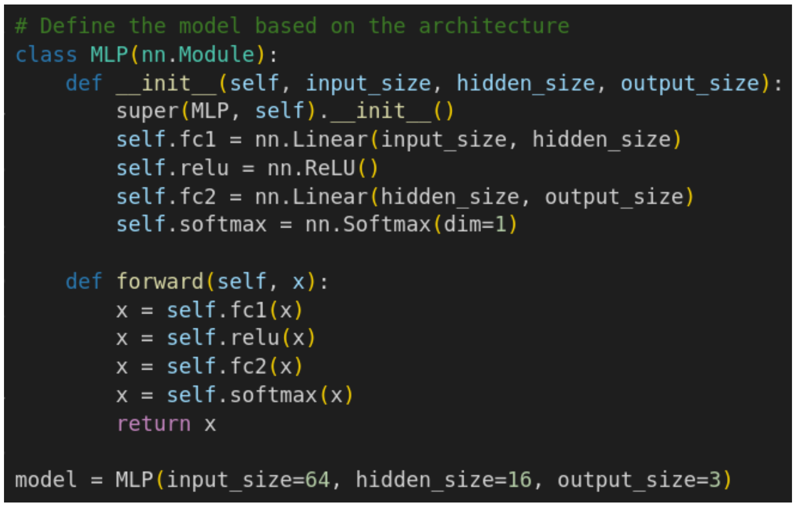
- Tanh:

```
def tanh(x):
    return (1 - np.exp(-2 * x)) / ( 1 + np.exp(-2 * x))
```

5-3: It can be implemented with one hidden layer because the images are not too complex, so the MLP doesn't need more hidden layers.

- Input layer: The input layer should have one neuron for each pixel in the image causing 64 neurons. Each neuron will represent the grayscale value of a pixel.
- Hidden layer: The number of neurons is not fixed and can be adjusted through experimentation, however I tried 32 and 16; And the result with 16 neurons was good. The ReLU is the activation function of this layer.it is commonly used in hidden layers due to its ability to introduce non-linearity.
- Output layer: The number of neurons in the output layer should be equal to the number of classes, which is 3 in this case. The Softmax activation is used because it converts the output values into class probabilities, which is suitable for multi-class classification.
- Loss function: Categorical cross-entropy is a common choice for multi-class classification tasks.

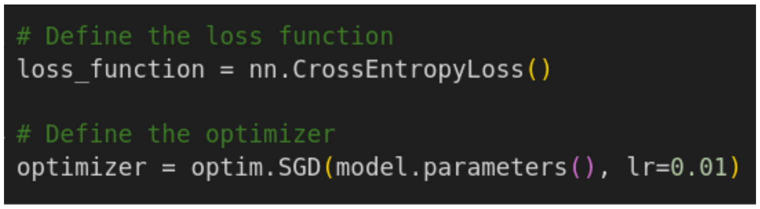
5-4: To implement the architecture represented in the last part, first I define a transformation to resize and normalize the images. Then I create train data and label tensors. After that, I define the model, loss function and optimizer based on my explanation on 5.3. At the end it is the time to train the model; for which I choose 1000 as the number of epochs. The loss decreased form 1.096 to 0.88 which means the model is working:)



```python
# Define the model based on the architecture
class MLP(nn.Module):
    def __init__(self, input_size, hidden_size, output_size):
        super(MLP, self).__init__()
        self.fc1 = nn.Linear(input_size, hidden_size)
        self.relu = nn.ReLU()
        self.fc2 = nn.Linear(hidden_size, output_size)
        self.softmax = nn.Softmax(dim=1)

    def forward(self, x):
        x = self.fc1(x)
        x = self.relu(x)
        x = self.fc2(x)
        x = self.softmax(x)
        return x

model = MLP(input_size=64, hidden_size=16, output_size=3)
```



```python
# Define the loss function
loss_function = nn.CrossEntropyLoss()

# Define the optimizer
optimizer = optim.SGD(model.parameters(), lr=0.01)
```

Question 6:

The problem is the combination of ReLU and Sigmoid activations. ReLU ensures that the output is always positive, and when followed by Sigmoid, the resulting values tend to be greater than or equal to 0.5. According to the threshold, the model consistently predicts class 1.These settings result in an ineffective model.

Question 7:

7-1: Machine learning is a more general approach to teach machines how to learn from data, however deep learning is a subset of machine learning that focuses on neural networks with multiple layers.

The key difference is that deep learning algorithms can automatically discover and extract complex patterns and representations from data, whereas traditional machine learning methods require human intervention to design and engineer features. This ability to learn complex representations is why deep learning has excelled in tasks like image and speech recognition, natural language processing, and other data-rich domains, however it often requires substantial computational power.

7-2: In deep neural networks, layers closer to the input capture lower-level features, while layers deeper in the network capture higher-level features; Therefore, the 11th layer is likely to have learned more abstract and complex representations compared to the 7th layer but if the classification task is relatively simple and the features required for classification are not highly abstract, the 7th layer may be sufficient for achieving good results.

Yet, deeper layers in a neural network are more prone to overfitting, especially if the network is very deep and the dataset is not sufficiently large. Thus, the 7th layer may be less prone to overfitting compared to the 11th layer.

Anyway, the best approach is often determined through experimenting with both configurations (using the 7th and 11th layers as the final classification layer) and checking which one performs better on a validation dataset.

7-3:Ultimately, the choice between deeper and wider networks depends on the problem's complexity, the available computational resources, and some other factors. If the problem has complex patterns in the data, and we have sufficient data and computational resources, deeper networks may be more effective. But, For problems where the relationships in the data are not highly hierarchical and we have limited computational resources, wider networks with more neurons per layer may be a better choice.

I got help from ChatGPT for this question:)

7-4:

Pros:

- Each additional layer can capture increasingly abstract and complex features from the data, enabling the network to learn more intricate patterns.
- Deeper networks can achieve higher predictive accuracy and better generalization to unseen data, especially for complex tasks.
- Pre-trained deep networks with many layers, such as convolutional neural networks (CNNs) and transformer models, have proven effective for transfer learning with fine-tuning these models on new tasks with small amounts of task-specific data.

Cons:

- Very deep networks can suffer from vanishing and exploding gradient problems during training, which can lead to slow convergence.
- Deeper networks are more prone to overfitting, especially when the dataset is small.
- Deeper networks often require larger datasets to generalize effectively.
- Deeper networks require more computational resources, which can make training more time-consuming and resource-intensive.