תרגיל לסטודנטים: הרחבת רשת נוירונים בסיסית
תרגיל 2
הגשה 9.10.2024

רקע

בתרגיל זה, נבנה רשת נוירונים פשוטה לסיווג תמונות מאגר הנתונים **CIFAR-10**. וכן הנתונים מתרגיל 1. המטרה היא להבין לעומק את תהליכי forward pass ו-backward pass ולממשם מאפס באמצעות **Numpy** בלבד. בהמשך נרחיב את הרשת ונשווה ביצועים עם אפשרויות שונות.

מטרות\ סעיפי התרגיל

1. **הוספת שכבה חדשה**:

   הרחבת רשת הנוירונים להכללת שכבה נסתרת נוספת.

   מימוש פונקציות forward ו-backward הכוללות את השכבה החדשה.

2. **ניסוי והשוואה**:

   בחנו את הרשת עם וללא נירמול (`preprocess_data`) והשוו את הביצועים.

3. **הרחבה עצמית:**

   הציעו רעיון עצמי אחד להרחבה - למשל שינוי ההיפרפרמטרים.

4. **מאגר נתונים אחר**:

   נסו להריץ את אותה רשת על מאגר הנתונים מהתרגיל הראשון והשוו ביצועים.

5. **פונקציות אקטיבציה**:

   החליפו את ReLU בפונקציה Sigmoid והשוו את התוצאות. – עשו זאת עבור שני סטים של נתונים cifar10 וזה של תרגיל 1.

עזריאלי
מכללה אקדמית להנדסה
ירושלים

קוד התחלתי

```
import numpy as np
import os
import tarfile
import urllib.request

def download_and_extract_cifar10(data_dir="cifar10_data"):
    url = "https://www.cs.toronto.edu/~kriz/cifar-10-binary.tar.gz"
    file_name = os.path.join(data_dir, "cifar-10-binary.tar.gz")

    if not os.path.exists(data_dir):
        os.makedirs(data_dir)

    # Download CIFAR-10 dataset
    if not os.path.exists(file_name):
        print("Downloading CIFAR-10 dataset...")
        urllib.request.urlretrieve(url, file_name)
        print("Download complete!")

    # Extract the dataset
    if not os.path.exists(os.path.join(data_dir, "cifar-10-batches-bin")):
        print("Extracting CIFAR-10 dataset...")
        with tarfile.open(file_name, "r:gz") as tar:
            tar.extractall(path=data_dir)
        print("Extraction complete!")

def load_cifar10_batch(file_path):

    with open(file_path, 'rb') as f:
        data = np.frombuffer(f.read(), dtype=np.uint8)

    num_samples = data.shape[0] // 3073
    labels = data[0::3073]  # Extract labels
    images = data[1:].reshape(num_samples, 3, 32, 32)  # Image data

    return images, labels

def preprocess_data(images, normalize=True):

    if normalize:
        images = images / 255.0  # Normalize pixel values to [0, 1]
    return images.reshape(images.shape[0], -1)  # Flatten images
```

מחלקת רשת נוירונים

```
class SimpleNN:
    def __init__(self, input_size, hidden_size1, hidden_size2,
output_size):
        self.weights1 = np.random.randn(input_size, hidden_size1) * 0.01
        self.bias1 = np.zeros((1, hidden_size1))
        self.weights2 = np.random.randn(hidden_size1, hidden_size2) * 0.01
        self.bias2 = np.zeros((1, hidden_size2))
        self.weights3 = np.random.randn(hidden_size2, output_size) * 0.01
        self.bias3 = np.zeros((1, output_size))

    def forward(self, x):


        self.z1 = np.dot(x, self.weights1) + self.bias1
        self.a1 = np.maximum(0, self.z1)  # ReLU activation

        self.z2 = np.dot(self.a1, self.weights2) + self.bias2
        self.a2 = np.maximum(0, self.z2)  # ReLU activation

        self.z3 = np.dot(self.a2, self.weights3) + self.bias3
        return self.softmax(self.z3)

    def softmax(self, x):


        exp_x = np.exp(x - np.max(x, axis=1, keepdims=True))
        return exp_x / np.sum(exp_x, axis=1, keepdims=True)

    def compute_loss(self, y_pred, y_true):

        num_samples = y_true.shape[0]
        correct_log_probs = -np.log(y_pred[range(num_samples), y_true])
        return np.sum(correct_log_probs) / num_samples

    def backward(self, x, y_true, y_pred, learning_rate=0.01):

        num_samples = x.shape[0]

        # Gradient for output layer
        dz3 = y_pred
        dz3[range(num_samples), y_true] -= 1
        dz3 /= num_samples

        dw3 = np.dot(self.a2.T, dz3)
        db3 = np.sum(dz3, axis=0, keepdims=True)
```

```
# Gradient for second hidden layer
da2 = np.dot(dz3, self.weights3.T)
dz2 = da2 * (self.z2 > 0)

dw2 = np.dot(self.a1.T, dz2)
db2 = np.sum(dz2, axis=0, keepdims=True)

# Gradient for first hidden layer
da1 = np.dot(dz2, self.weights2.T)
dz1 = da1 * (self.z1 > 0)

dw1 = np.dot(x.T, dz1)
db1 = np.sum(dz1, axis=0, keepdims=True)

# Update weights and biases
self.weights1 -= learning_rate * dw1
self.bias1 -= learning_rate * db1
self.weights2 -= learning_rate * dw2
self.bias2 -= learning_rate * db2
self.weights3 -= learning_rate * dw3
self.bias3 -= learning_rate * db3
```

הנחיות לסטודנטים

1. **עבודה עצמאית**:
זכרו, המטרה היא לממש את הקוד בעצמכם ולא להתפתות להיעזר באינטרנט.
הבנת התהליך של forward pass ו-backward pass היא קריטית להבנה עמוקה של רשתות נוירונים.

2. **תיעוד הקוד**:
דאגו להוסיף הערות בעברית כהסבר בקובץ ה pdf שתצרפו עם התרגיל. הסבר בו את החישובים שנעשים בפוקציות `softmax` , `compute_loss` , ן `backward`

הגישו שני קבצים. 1. קוד מעודכן כקובץ py אשר מריץ אץ כל הנסיונות שלכם. 2. את הסברים בקובץ ה pdf מנומק ורשמו ת.ז + שם

בהצלחה!