Ejercicios3

June 21, 2024

1 Ejercicios sobre diferenciación automática

1.1 1. Modo forward

Realiza la gráfica computacional para la función $f : \mathbb{R}^2 \to \mathbb{R}^2$ definida por:

$$f(x) = \begin{pmatrix} \ln(x_1^2) + 2x_2 \\ \cos\left(\ln(x_1^2) + x_2^2\right) \end{pmatrix}$$

Y obtén el gradiente de la función utilizando el modo forward. Ejemplifica con la entrada $x = (2, 5)$.

1.2 2. Modo reverse

A partir de la gráfica computacional dada con los nodos Linear, Softmax y CrossEntropy, crea la gráfica computacional para el problema de la regresión logística:

$$f(x) = \sigma(wx + b)$$

Donde σ es la función softmax y w y b los parámetros a aprender. Cálcula el gradiente utilizando la función objetivo de entropía cruzada y actualiza los pesos por medio del método de ascenso por gradiente:

$$\theta \leftarrow \theta - \eta \nabla \mathcal{J}(\theta)$$

Donde η es la taza de aprendizaje.

Utiliza el dataset de Iris de sklearn, y recuerda realizar la separación de 70-30 en los datos de entrenamiento y evaluación. Asimismo, evalúa utilizando classification_report.

[1]:
```
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn.datasets import load_iris
from tqdm import tqdm
from nn import *
import matplotlib.pyplot as plt

data = load_iris()
x = data.data
y = data.target
```

```
dim = len(data.feature_names)
classes = len(data.target_names)

x_train, x_test, y_train, y_test = train_test_split(x,y, test_size=0.3)
```

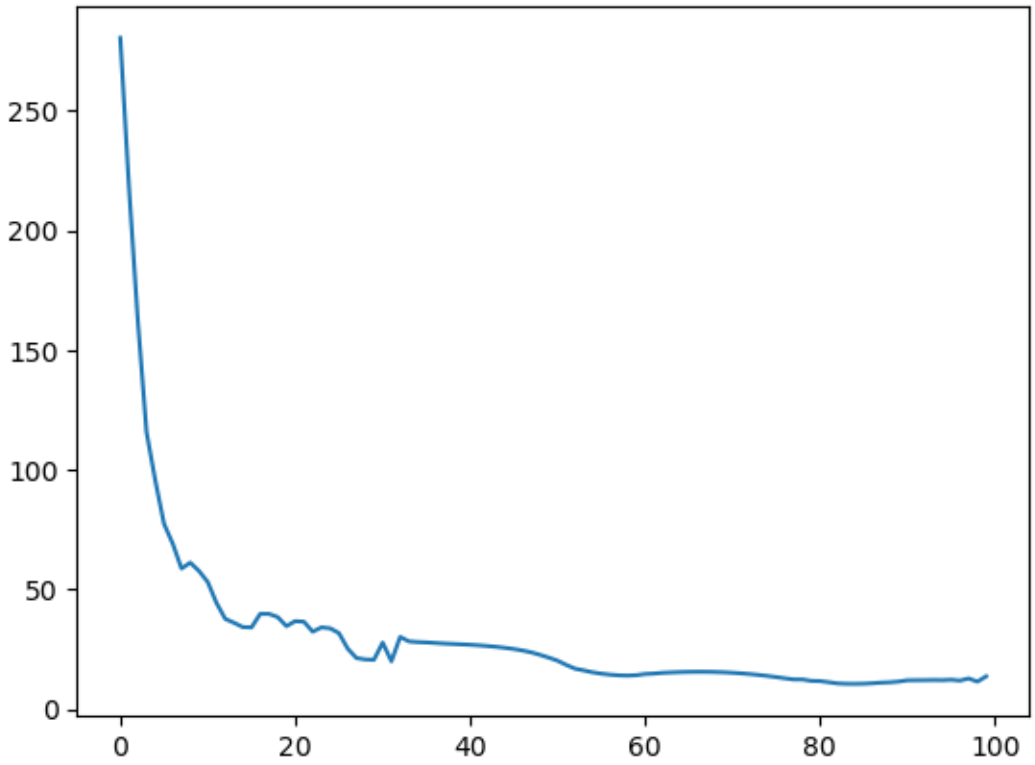
[2]:
```
lin1 = Linear(dim, classes)
soft = Softmax()
risk = CrossEntropy()
```

[3]:
```
epochs = 100
lr = 0.1
total_loss = []
for t in tqdm(range(epochs)):
    epoch_loss = 0
    for x_i, y_i in zip(x_train,y_train):
        pred = soft(lin1(x_i))
        loss = risk(pred, y_i)

        loss.backward()
        lin1.w -= lr*lin1.grad
        lin1.b -= lr*lin1.grad_b
        epoch_loss += loss.value
    total_loss.append(epoch_loss)
```

```
100%|                    | 100/100 [00:00<00:00, 316.75it/s]
```

[4]:
```
plt.plot(total_loss)
plt.show()
```



[5]:
```
test_pred = [soft(lin1(x_i)).argmax() for x_i in x_test]

print(classification_report(test_pred, y_test))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        13
           1       0.95      1.00      0.97        18
           2       1.00      0.93      0.96        14

    accuracy                           0.98        45
   macro avg       0.98      0.98      0.98        45
weighted avg       0.98      0.98      0.98        45
```