Exerceise 21: Simple MNIST NN

Sergio Marín Sánchez

March 29, 2024

1 Exerceise 21: Simple MNIST NN

The following code is a simple neural network that classifies the MNIST dataset.

The next cell imports the necessary libraries and the dataset. It runs the neural network and prints some tests.

```
from simple_ml import *
```

```
Iteration:  0
[2 3 0 … 6 8 3] [1 5 1 … 7 6 9]
0.0713170731707317
Iteration:  10
[2 8 0 … 3 8 3] [1 5 1 … 7 6 9]
0.15117073170731707
Iteration:  20
[2 8 0 … 3 8 8] [1 5 1 … 7 6 9]
0.20624390243902438
Iteration:  30
[2 8 0 … 3 8 8] [1 5 1 … 7 6 9]
0.2491219512195122
Iteration:  40
[8 8 8 … 3 8 8] [1 5 1 … 7 6 9]
0.28785365853658534
Iteration:  50
[8 8 8 … 3 8 8] [1 5 1 … 7 6 9]
0.348780487804878
Iteration:  60
[8 8 8 … 3 6 8] [1 5 1 … 7 6 9]
0.4145853658536585
Iteration:  70
[8 8 8 … 4 6 8] [1 5 1 … 7 6 9]
0.48809756097560975
Iteration:  80
[1 8 1 … 4 6 8] [1 5 1 … 7 6 9]
0.550219512195122
Iteration:  90
[1 8 1 … 4 6 9] [1 5 1 … 7 6 9]
0.5970243902439024
```

```
Iteration:  100
[1 8 1 … 7 6 9] [1 5 1 … 7 6 9]
0.6325365853658537
Iteration:  110
[1 8 1 … 7 6 9] [1 5 1 … 7 6 9]
0.6596585365853659
Iteration:  120
[1 4 1 … 7 6 4] [1 5 1 … 7 6 9]
0.681609756097561
Iteration:  130
[1 4 1 … 7 6 4] [1 5 1 … 7 6 9]
0.6976341463414634
Iteration:  140
[1 4 1 … 7 6 4] [1 5 1 … 7 6 9]
0.7113170731707317
Iteration:  150
[1 4 1 … 7 6 4] [1 5 1 … 7 6 9]
0.7243414634146341
Iteration:  160
[1 4 1 … 7 6 4] [1 5 1 … 7 6 9]
0.7349512195121951
Iteration:  170
[1 4 1 … 7 6 4] [1 5 1 … 7 6 9]
0.7437560975609756
Iteration:  180
[1 4 1 … 7 6 4] [1 5 1 … 7 6 9]
0.7513414634146341
Iteration:  190
[1 4 1 … 7 6 4] [1 5 1 … 7 6 9]
0.7588536585365854
Iteration:  200
[1 4 1 … 7 6 4] [1 5 1 … 7 6 9]
0.7651951219512195
Iteration:  210
[1 4 1 … 7 6 4] [1 5 1 … 7 6 9]
0.7708780487804878
Iteration:  220
[1 4 1 … 7 6 4] [1 5 1 … 7 6 9]
0.7753414634146342
Iteration:  230
[1 4 1 … 7 6 4] [1 5 1 … 7 6 9]
0.779390243902439
Iteration:  240
[1 4 1 … 7 6 4] [1 5 1 … 7 6 9]
0.7837317073170732
Iteration:  250
[1 4 1 … 7 6 4] [1 5 1 … 7 6 9]
0.788
```

```
Iteration:  260
[1 4 1 … 7 6 4] [1 5 1 … 7 6 9]
0.7921707317073171
Iteration:  270
[1 4 1 … 7 6 4] [1 5 1 … 7 6 9]
0.7955853658536586
Iteration:  280
[1 4 1 … 7 6 4] [1 5 1 … 7 6 9]
0.7989756097560976
Iteration:  290
[1 4 1 … 7 6 4] [1 5 1 … 7 6 9]
0.8015365853658537
Iteration:  300
[1 4 1 … 7 6 4] [1 5 1 … 7 6 9]
0.8043658536585366
Iteration:  310
[1 4 1 … 7 6 4] [1 5 1 … 7 6 9]
0.8072439024390244
Iteration:  320
[1 4 1 … 7 6 4] [1 5 1 … 7 6 9]
0.809829268292683
Iteration:  330
[1 4 1 … 7 6 4] [1 5 1 … 7 6 9]
0.812219512195122
Iteration:  340
[1 4 1 … 7 6 4] [1 5 1 … 7 6 9]
0.8149268292682926
Iteration:  350
[1 4 1 … 7 6 4] [1 5 1 … 7 6 9]
0.8166829268292682
Iteration:  360
[1 4 1 … 7 6 4] [1 5 1 … 7 6 9]
0.8184878048780487
Iteration:  370
[1 4 1 … 7 6 4] [1 5 1 … 7 6 9]
0.8206585365853658
Iteration:  380
[1 4 1 … 7 6 4] [1 5 1 … 7 6 9]
0.8222195121951219
Iteration:  390
[1 4 1 … 7 6 4] [1 5 1 … 7 6 9]
0.8240731707317073
Iteration:  400
[1 4 1 … 7 6 4] [1 5 1 … 7 6 9]
0.8257073170731707
Iteration:  410
[1 4 1 … 7 6 4] [1 5 1 … 7 6 9]
0.8272926829268292
```

```
Iteration:  420
[1 4 1 … 7 6 4] [1 5 1 … 7 6 9]
0.8287560975609756
Iteration:  430
[1 4 1 … 7 6 4] [1 5 1 … 7 6 9]
0.8301219512195122
Iteration:  440
[1 4 1 … 7 6 4] [1 5 1 … 7 6 9]
0.8313414634146341
Iteration:  450
[1 4 1 … 7 6 4] [1 5 1 … 7 6 9]
0.8324146341463414
Iteration:  460
[1 4 1 … 7 6 4] [1 5 1 … 7 6 9]
0.8339268292682926
Iteration:  470
[1 4 1 … 7 6 4] [1 5 1 … 7 6 9]
0.8351219512195122
Iteration:  480
[1 4 1 … 7 6 4] [1 5 1 … 7 6 9]
0.836170731707317
Iteration:  490
[1 4 1 … 7 6 4] [1 5 1 … 7 6 9]
0.8374634146341463
Prediction:  [1]
Label:  1
Prediction:  [4]
Label:  5
Prediction:  [1]
Label:  1
Prediction:  [7]
Label:  7
```

```
i1 = im1__4__
print(call__make_predictions(i1))
```

```
[4]
```

```
in1, _ = call__make_predictions__adding__noise(i1, 5.0)
```

```
Original:  [4]
With noise added:  [4]
```

```
noise_scale = 5.0
while True:
    print(f'Checking noise scale: {noise_scale}')
    in1, prediction = call__make_predictions__adding__noise(i1, noise_scale)
    print('-' * 30)
```

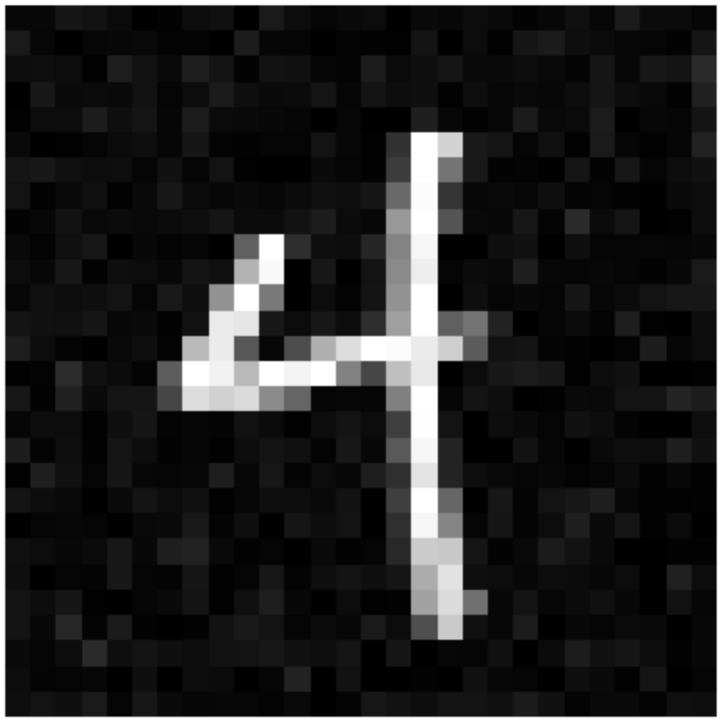
```
    if prediction != 4:
        break
    noise_scale += 5.0
```

```
Checking noise scale: 5.0
Original:  [4]
With noise added:  [4]
------------------------------
Checking noise scale: 10.0
Original:  [4]
With noise added:  [4]
------------------------------
Checking noise scale: 15.0
Original:  [4]
With noise added:  [9]
------------------------------
```

Show the image with noise

```
plt.figure()
plt.imshow(in1.reshape((28,28)), cmap='gray')
plt.axis('off')
```

```
(-0.5, 27.5, 27.5, -0.5)
```



Save the image with noise

```
cv2.imwrite('im1__4__noisy.pgm', in1.reshape((28,28)))
```

```
True
```

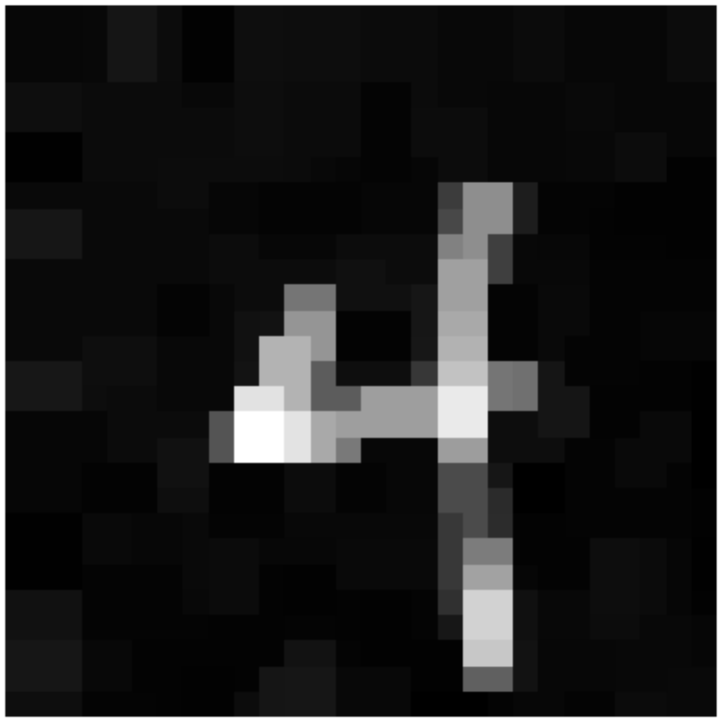
```
if1, _ = call__make_predictions__with__filtering(in1)
```

```
Before filtering:  [9]
After filtering:  [4]
```

Show the image with noise and filtering

```
plt.figure()
plt.imshow(if1.reshape((28,28)), cmap='gray')
plt.axis('off')
```

```
(-0.5, 27.5, 27.5, -0.5)
```



Save the image with noise and filtering

```
cv2.imwrite('im1__4__filtered.pgm', if1.reshape((28,28)))
```

```
True
```