Solutions - Practical Lesson 8

Matteo Sani
matteosan1@gmail.com

November 19, 2019

1 Solutions

1.1 Exercises

1.1.1 Exercise 8.3

Taking as example the pricing NN trained on call, try to price put options.

Solution To adapt the code to predict put prices instead of calls it is enough to replace in the `bs_simulation.py` the function call to the call pricer (much easier to do than to explain):

[ ]:
```
import numpy as np
np.random.seed(1000)
import matplotlib.pyplot as plt
import csv
from finmarkets import put

def gen_paths(S0, r, sigmas, T, M, I):
    dt = float(T) / M
    paths = np.zeros((M + 1, I), np.float64)
    paths[0] = S0
    for t in range(1, M + 1):
        for i in range(len(sigmas)):
            rand = np.random.standard_normal()
            paths[t, i] = paths[t - 1, i] * np.exp((r - 0.5 * sigmas[i] ** 2) *␣
 ↪dt +
                                                   sigmas[i] * np.sqrt(dt) * rand)
    return paths

S0 = 100                                    # initial stock price                ␣
 ↪
K = [100]#[80, 90, 100, 110, 120]           # strike price                       ␣
 ↪
r = 0.01                                    # risk-free interest rate            ␣
 ↪
sigmas = [s/100. for s in range(15, 55, 5) ]# volatility in market             ␣
 ↪
```

```
T = [1] #0.25, 0.5, 0.75, 1, 2, 3]          # time in years
 ↪
M = 365*max(T)                              # number of steps within each
 ↪simulation
I = len(sigmas)                             # number of simulations
 ↪


paths = gen_paths(S0, r, sigmas, max(T), M, I)

plt.ion()
plt.plot(paths)
plt.grid(True)
plt.xlabel('days')
plt.ylabel('$S^{i}_{t}$')
plt.show()
plt.savefig("underlyings.png")

with open("bs_training.csv", mode='w') as f:
    writer = csv.writer(f, delimiter=",", quotechar='"', quoting=csv.
 ↪QUOTE_MINIMAL)
    for sims in range(I):
        for i, S_t in enumerate(paths[:, sims]):
            for t in T:
                M = t * 365
                if i >= M:
                    continue
                for k in K:
                    put_price = put(S_t, k, r, sigmas[sims], t)
                    writer.writerow([sigmas[sims], k, t, S_t, put_price])


S0 = 100
K = 100
r = 0.01
sigmas = [0.25]
T = 1
M = 365*T
I = 1

paths = gen_paths(S0, r, sigmas, T, M, I)
with open("bs_testing.csv", mode='w') as f:
    writer = csv.writer(f, delimiter=",", quotechar='"', quoting=csv.
 ↪QUOTE_MINIMAL)
    for i, S_t in enumerate(paths[:, 0]):
        put_price = put(S_t, K,  r, sigmas[0], T)
        writer.writerow([sigmas[0], K, T, S_t, put_price])
```

```
S0 = 100
K = 95
r = 0.01
sigmas = [0.20]
T = 3
M = 365*T
I = 1

paths = gen_paths(S0, r, sigmas, T, M, I)

with open("bs_testing_off.csv", mode='w') as f:
    writer = csv.writer(f, delimiter=",", quotechar='"', quoting=csv.
 ↪QUOTE_MINIMAL)
    for i, S_t in enumerate(paths[:, 0]):
        put_price = put(S_t, K,  r, sigmas[0], T)
        writer.writerow([sigmas[0], K, T, S_t, put_price])
```

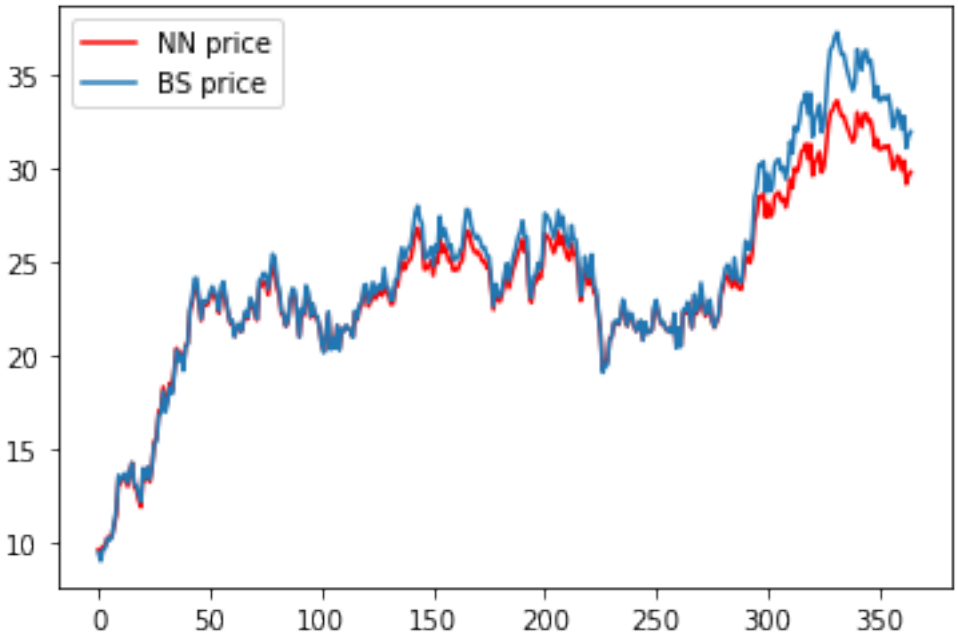
Then it is enough to run on the generated samples, `bs_training.py` and `bs_testing.py` to train and check the performance of the new NN.

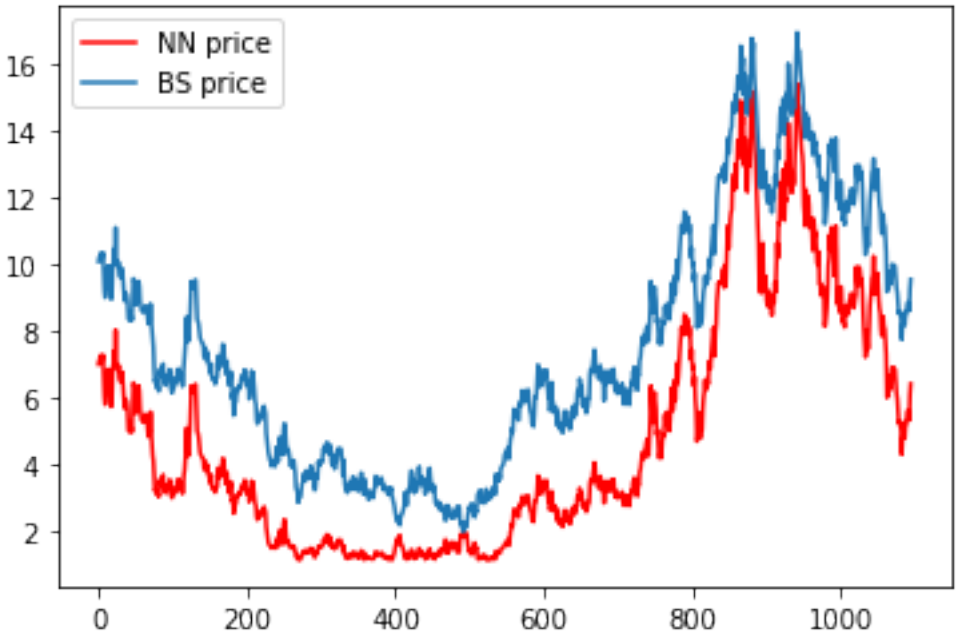
[1]:
```
import bs_training
```

```
Using TensorFlow backend.

Epoch 1/1000
2919/2919 [==============================] - 0s 55us/step - loss: 376.4286 -
mse: 376.4286 - mae: 15.1232
Epoch 500/1000
2919/2919 [==============================] - 0s 12us/step - loss: 0.1357 - mse:
0.1357 - mae: 0.2707
Epoch 1000/1000
2919/2919 [==============================] - 0s 10us/step - loss: 0.0521 - mse:
0.0521 - mae: 0.1590
2919/2919 [==============================] - 0s 25us/step
Test: [0.07155836318307195, 0.071558378636837, 0.2163873016834259]
```

[2]:
```
import bs_testing
```






```
<Figure size 432x288 with 0 Axes>
```

From this simple exercise with some degree of symmetry between the two problems it is clear that it is not enough to reuse some NN without specific study on the problem under investigation.