Problem2-metropolis

April 9, 2021

1 Metropolis algorithm example

Here we look at the Metropolis-Hastings algorithm, which is a Markov-Chain Monte Carlo (MCMC) technique.

1.0.1 Swig it, compile it, add it to the path

[1]:
```
! swig -c++ -python swig/metropolis.i
! python swig/setup_metropolis.py build_ext --inplace
```

```
running build_ext
building '_metropolis' extension
aarch64-linux-gnu-gcc -pthread -DNDEBUG -g -fwrapv -O2 -Wall -g -fstack-
protector-strong -Wformat -Werror=format-security -Wdate-time
-D_FORTIFY_SOURCE=2 -fPIC -I/usr/include/python3.7m -c swig/metropolis_wrap.cxx
-o build/temp.linux-aarch64-3.7/swig/metropolis_wrap.o -I./ -std=c++11 -O3
aarch64-linux-gnu-g++ -pthread -shared -Wl,-O1 -Wl,-Bsymbolic-functions
-Wl,-z,relro -Wl,-z,relro -g -fstack-protector-strong -Wformat -Werror=format-
security -Wdate-time -D_FORTIFY_SOURCE=2 build/temp.linux-
aarch64-3.7/swig/metropolis_wrap.o -o /home/pi/tensorflow-probability-
ChanceStarr/RandomNumbers/_metropolis.cpython-37m-aarch64-linux-gnu.so
```

[2]:
```
import sys
import os
sys.path.append( os.path.abspath("swig") )
```

[3]:
```
import metropolis
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
```

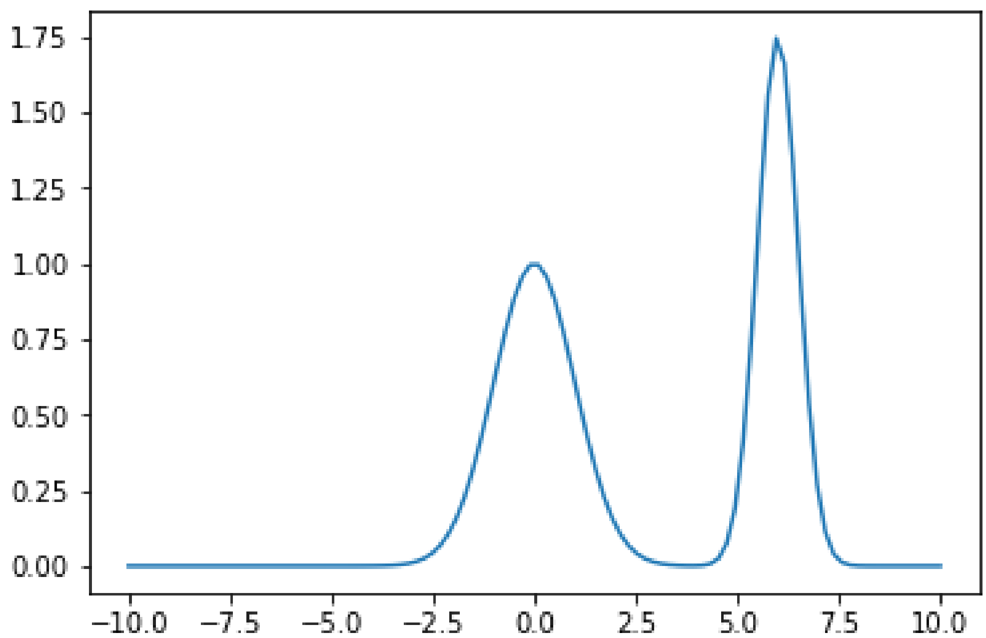
1.1 Probability distribution

Make the probability distribution equal to a sum of Gaussians.

[4]:
```
A = [1., 1.75]
sigma = [1.0, 0.5]
center = [0.0, 6.0]
```

```
g = metropolis.gaussianD( A, sigma, center )

gvalsi = []
gxvals = np.linspace(-10,10,100)
for x in gxvals:
    gvalsi.append( g(x) )
gvals = np.array(gvalsi)
```

[5]:
```
plt.plot(gxvals, gvals)
plt.show()
```



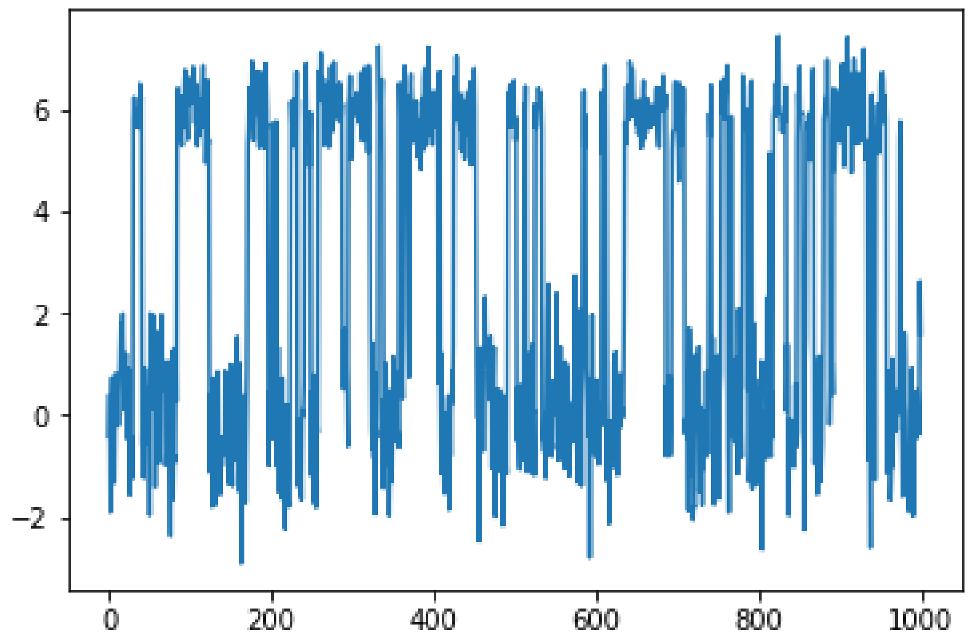
1.2 Run Metropolis-Hastings

[6]:
```
x0 = 0.0
delta = 1.0
nskip = 1000

m = metropolis.metropolisD( g, x0, delta, nskip, False )
xvals = []

nmcsteps = 1000
for i in range(nmcsteps):
    m.monte_carlo_step()
    xvals.append( m.get() )
```

1.3 Plot the time series of the "walker"

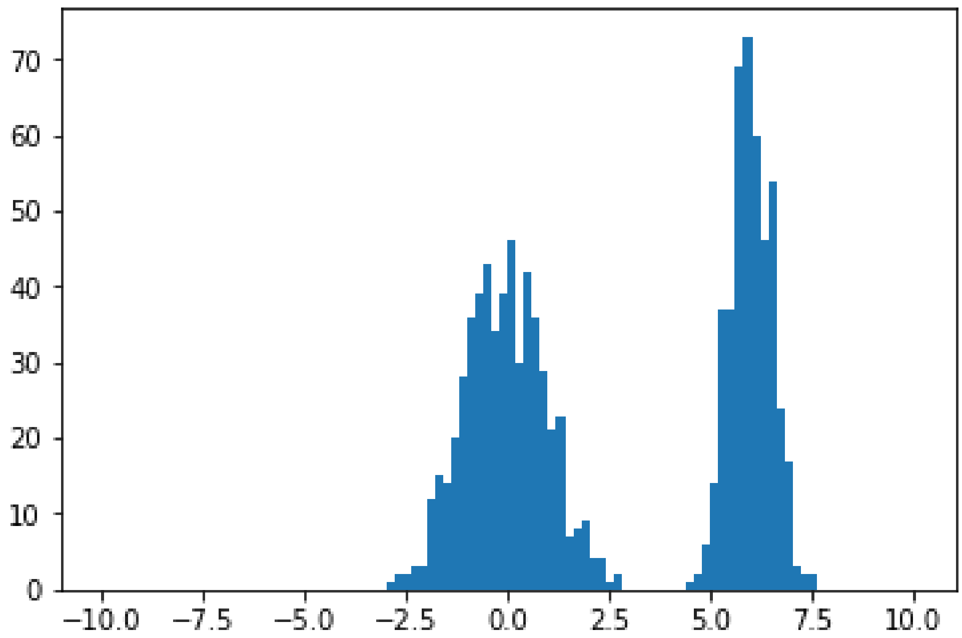
[7]:
```python
plt.plot(xvals)
```

[7]: [<matplotlib.lines.Line2D at 0x7f4be05f28>]



1.4 Plot the distribution that MH arrives at

[8]:
```python
res = plt.hist( xvals, bins=100, range=(-10,10) )
plt.show()
```



2 Problem 2 - TensorFlow

Below tensorflow and tensorflow-probability are used to calculate the Metropolis algorithm.

[9]:
```
import numpy as np
import tensorflow.compat.v2 as tf
import tensorflow_probability as tfp
tf.enable_v2_behavior()

tfd = tfp.distributions

dtype = np.float32

target = tfd.Normal(loc=dtype(0), scale=dtype(1))

samples = tfp.mcmc.sample_chain(
  num_results=1000,
  current_state=dtype(1),
  kernel=tfp.mcmc.RandomWalkMetropolis(target.log_prob),
  num_burnin_steps=500,
  trace_fn=None,
  seed=42)

sample_mean = tf.math.reduce_mean(samples, axis=0)
sample_std = tf.sqrt(
```

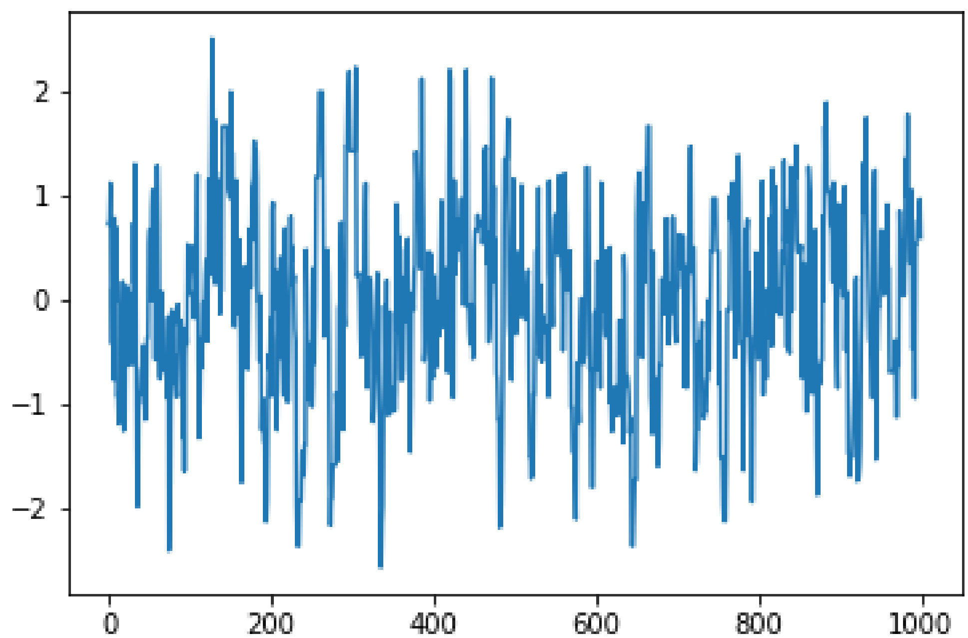
```
    tf.math.reduce_mean(
        tf.math.squared_difference(samples, sample_mean),
        axis=0))

print('Estimated mean: {}'.format(sample_mean))
print('Estimated standard deviation: {}'.format(sample_std))
```

```
Estimated mean: -0.06041654199361801
Estimated standard deviation: 0.9357634782791138
```

2.1 Plot the time series of the "walker"

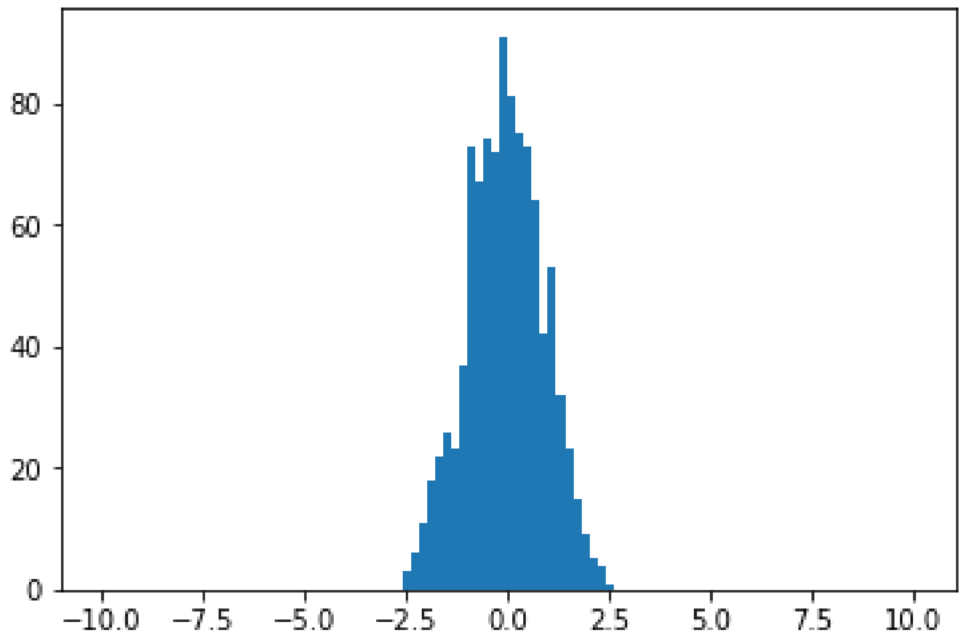
[10]:
```
plt.plot(samples)
```

[10]: [<matplotlib.lines.Line2D at 0x7f32410da0>]



2.2 Plot the distribution that MH arrives at

[11]:
```
res = plt.hist( samples, bins=100, range=(-10,10) )
plt.show()
```



80
60
40
20
0
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0


```
[ ]:
```