CS294 Deep Reinforcement Learning HW4: MPC Model-Based RL

Yifat Amir

In [1]:
```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
```

Part 1

Fit a dynamics model to random data alone and use the learned dynamics model in your MPC controller to control the cheetah robot. Report your performance (copy/paste the log output into your report).

Iteration 0

AverageCost 121.468224013

StdCost 57.2858408949

MinimumCost 53.6009972585

MaximumCost 234.567366938

AverageReturn -66.0960621149

StdReturn 16.5012392242

MinimumReturn -98.4327335961

MaximumReturn -39.3160723071

Part 2

Run the full algorithm, including on-policy data aggregation, for 15 iterations. Make a graph of the performance (average return) at each iteration. How does performance change when the on-policy data is included?

I'm plotting only the first 6 iterations because my program is killed (9) consistently partway through iteration 7 (probably due to RAM). However, you can still see the improvement in the average return due to on-policy data.

In [2]:
```
# python main.py -n 15
```

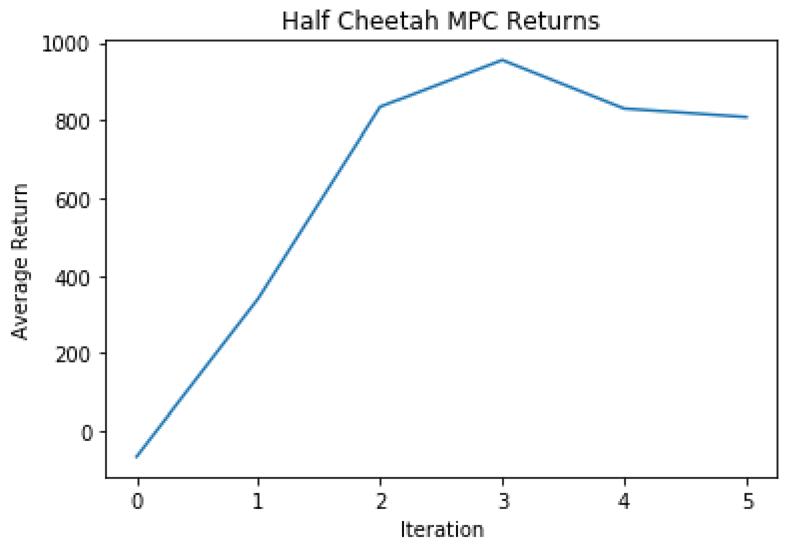
In [3]:
```
returns = np.array([[-55.48267077, -80.45197874, -52.65621207, -79.43407
671, -62.87969339,
 -98.4327336,  -57.90643994, -39.31607231, -57.1266877,  -77.27405592],
[ 367.6707783,   331.59849781,  340.56866406,  323.04083954,  335.528623
6,
  363.92776312,  347.00587687,  264.78432737,  393.87546613,  345.505925
72],
[ 693.91343055,  896.65851939,  897.74801041,  910.33378461,  842.536152
05,
  767.53420877,  901.6208748,   819.93068588,  822.1079186,   791.680131
74],
[  973.90002293,  1004.31786705,  1037.80711735,   967.5827035,    933.3
3556119,
   953.95272915,  1017.77948674,   895.31525909,   885.10568509,   879.9
8698304],
[ 824.88018926,  751.08901308,  820.39476689,  802.62657201,  812.547309
15,
  857.3520731,   884.34524805,  874.87008172,  807.22266434,  862.945757
16],
[ 813.95781192,  816.13460029,  771.81553034,  883.36952685,  884.557410
11,
  791.28056846,  800.65754863,  793.41804097,  792.37504429,  730.999163
84]])
```

In [4]:
```
avg_returns = [np.mean(row) for row in returns]
print(avg_returns)
```

```
[-66.096062115000009, 341.35067625199997, 834.40637167999989, 954.90834
15130001, 829.82736747600006, 807.85652456999992]
```

In [5]:
```
plt.plot(range(6), avg_returns)

plt.xlabel('Iteration')
plt.ylabel('Average Return')
plt.title('Half Cheetah MPC Returns')
plt.show()
```


Half Cheetah MPC Returns
Average Return
Iteration