Ahern-Dissertation-AppendixA

November 22, 2015

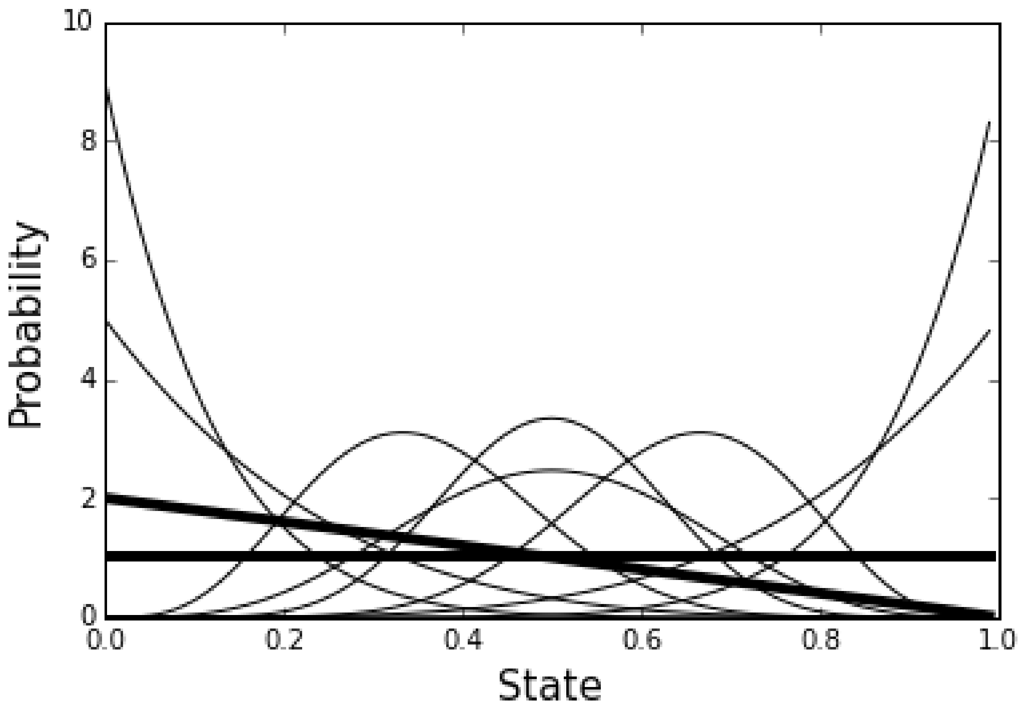
0.1 Definitions

```
In [1]: from sympy import *
        import numpy as np

In [2]: states = symbols('t0:10') # Sufficient for up to ten messages
        messages = symbols('m0:10')
        actions = symbols('a0:10')
        t, b = symbols('t b', positive=True)

In [3]: def U_S(state, action, bias):
                # return 0 - (action - state - bias)**2 # Crawford & Sobel 1982
                return  1 - (action - state - (1 - state)*bias)**2

        def U_R(state, action):
            # return 0 - (action - state - bias)**2 # Crawford & Sobel 1982
            return  1 - (action - state)**2
```

0.2 Beta Distribution

0.3 Visualize

```
In [4]: from scipy.stats import beta, uniform
        import matplotlib.pyplot as plt
        %matplotlib inline
        import numpy as np

In [13]: hfont = {'fontname':'Helvetica'}
         x = np.arange (0, 1, 0.01)
         for alpha_var in range(1,10, 4):
             for beta_var in range(1,10, 4):
                     y = beta.pdf(x,alpha_var,beta_var)
                     plt.plot(x,y, 'k')
         plt.plot(x, beta.pdf(x,1,2), linewidth=4, color='k')
         plt.plot(x, beta.pdf(x,1,1), linewidth=4, color='k')
         plt.ylabel("Probability", fontsize=15, **hfont)
         plt.xlabel("State", fontsize=15, **hfont)
         plt.savefig('beta-distribution.png', fontsize=15, **hfont)
         plt.show()
```




```
In [6]: from sympy.stats import Uniform, Beta, density, E, sample, P
        from sympy import symbols

In [7]: from sympy import *

In [2]: from sympy import *
        t, m, a, a_0, a_1, m_0, m_1, b = symbols('t m a a_0 a_1 m_0 m_1 b')
        t_star = symbols('t_star')

In [14]: from sympy import *
         from sympy.stats import Uniform, density, Beta, E

In [15]: X = Beta("x", 1,2)

         part1 = 1 - (actions[0] - t - (1-t)*b)**2
         part2 = 1 - (actions[1] - t - (1-t)*b)**2

         full_S = integrate(part1*density(X)(t).evalf(), (t, 0, states[0])) + integrate(part2*density(X
         full_R = full_S.subs(b, 0)

In [17]: t0_sol = Eq(solve(diff(full_S, states[0]), states[0])[0], states[0])
         print t0_sol

0.25*(-a0 - a1 + 4.0*b - sqrt((a0 + a1 - 2.0)**2) - 2.0)/(b - 1.0) == t0

In [18]: a0_sol = Eq(solve(diff(full_R, actions[0]), actions[0])[0], actions[0])
         print a0_sol

0.333333333333333*t0*(2.0*t0 - 3.0)/(t0 - 2.0) == a0
```

```
In [19]: a1_sol = Eq(solve(diff(full_R, actions[1]), actions[1])[0], actions[1])
         print a1_sol
```

```
0.666666666666667*t0 + 0.333333333333333 == a1
```

```
In [20]: result = solve([t0_sol, a0_sol, a1_sol], [states[0], actions[0], actions[1]])
```

```
In [21]: print result
```

```
[((9.0*b - sqrt(9.0*b**2 - 18.0*b + 5.0) - 3.0)/(6.0*b - 2.0), 0.333333333333333*(-3.0 + 2.0*(9.0*b - s
```

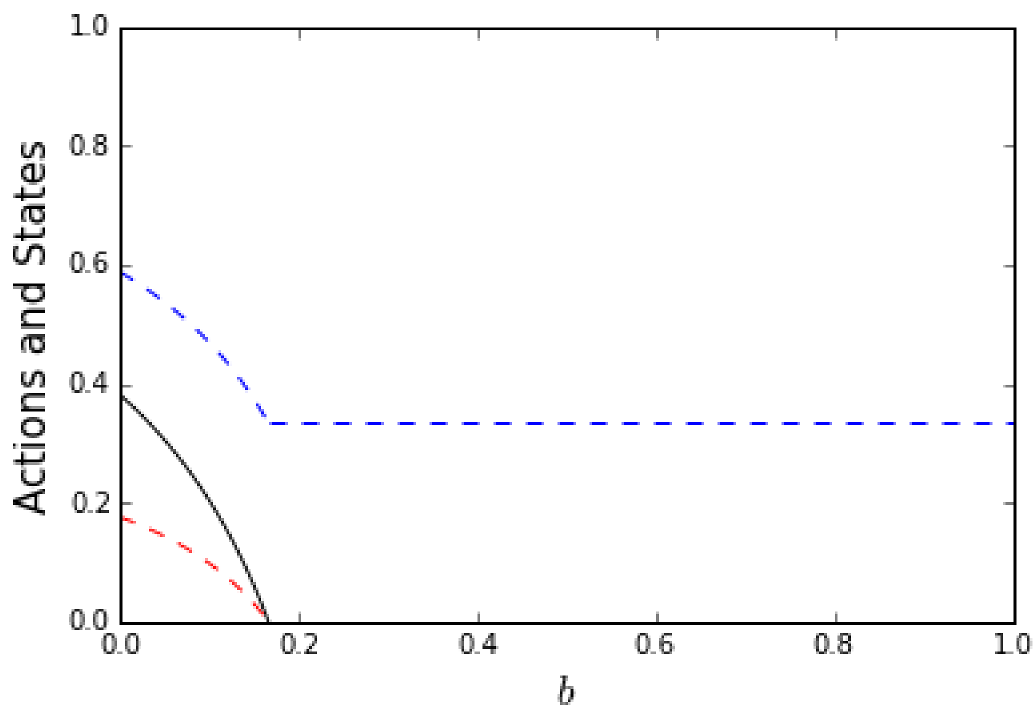
```
In [54]: #result = [((9.0*b - sqrt(9.0*b**2 - 18.0*b + 5.0) - 3.0)/(6.0*b - 2.0), 0.333333333333333*(-3
```

```
In [29]: result[1]
```

Out[29]:

$$\left(\frac{1}{6.0b-2.0}\left(9.0b+\sqrt{9.0b^{2}-18.0b+5.0}-3.0\right),\quad \frac{0.333333333333333\left(-3.0+\frac{1}{6.0b-2.0}\left(18.0b+2.0\sqrt{9.0b^{2}-18.0b+5.0}-6.0\right)\right)\left(9.0b+\sqrt{9.0b^{2}-18.0b+5.0}-3.0\right)}{\left(-2.0+\frac{1}{6.0b-2.0}\left(9.0b+(9.0b^{2}-18.0b+5.0)^{0.5}-3.0\right)\right)(6.0b-2.0)},\quad 0.333333333333333+\frac{1}{6.0b-2.0}\left(6.0b+0.666666666666667\sqrt{9.0b^{2}-18.0b+5.0}-2.0\right)\right)$$

```
In [33]: x = np.linspace(0,1/6.0, num=100)

         plt.plot(x, [result[1][0].subs(b, value).evalf() for value in x], 'k')
         plt.plot(x, [result[1][1].subs(b, value).evalf() for value in x], 'r', linestyle='--')
         plt.plot(x, [result[1][2].subs(b, value).evalf() for value in x], 'b', linestyle='--')
         plt.axhline(1/3.0, 1/6.0, 1, color='b', ls='--')
         plt.ylim(0,1)
         plt.xlim(0,1)
         plt.xlabel(r"$b$", fontsize=15, **hfont)
         plt.ylabel("Actions and States", fontsize=15, **hfont)
         plt.savefig("sol2-beta.png")
         plt.show()
```



```
In [23]: solve(result[1][0], b)
```

Out[23]:

$$[0.1666666666666667]$$