7_2

May 2, 2016

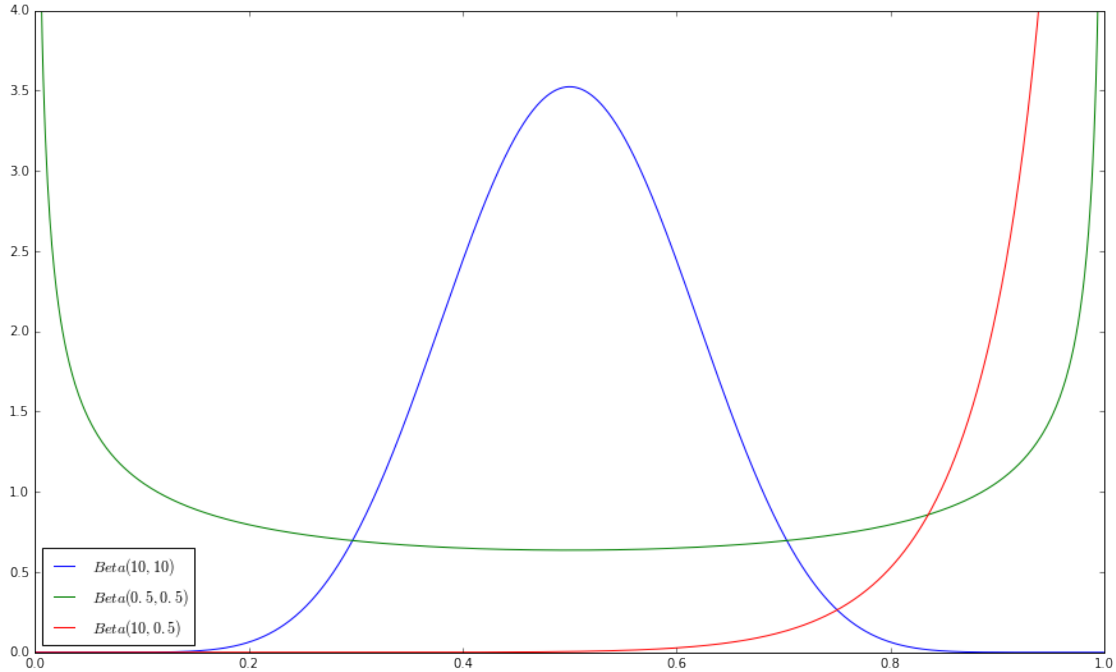
```
In [34]: import numpy as np
         import matplotlib.pyplot as plt
         import scipy.stats as stats
         %matplotlib inline
```

```
In [35]: grid = np.arange(0, 1, 0.001)
```

Для распределения $Bern(p)$ сопряженным является $Beta(\alpha, \beta)$. Рассмотрим три распределения, соответствующие “скорее честной”, “нечестной” и “нечестной, с перевесом в сторону герба” - $Beta(10, 10), Beta(0.5, 0.5)$ и $Beta(10, 0.5)$ соответственно.

```
In [180]: def fair_coin_dist(x):
              return stats.beta.pdf(x, 10, 10)
          def unfair_coin_dist(x):
              return stats.beta.pdf(x, 0.5, 0.5)
          def unfair_CoA_coin_dist(x):
              return stats.beta.pdf(x, 10, 0.5)
```

```
In [181]: plt.figure(figsize=(15, 9))
          params = (10, 10)
          plt.plot(grid, [fair_coin_dist(x) for x in grid], label=r'$Beta(10, 10)$')
          plt.plot(grid, [unfair_coin_dist(x) for x in grid], label=r'$Beta(0.5, 0.5)$')
          plt.plot(grid, [unfair_CoA_coin_dist(x) for x in grid], label=r'$Beta(10, 0.5)$')
          plt.ylim(0, 4)
          plt.legend(loc='best')
          plt.draw()
```



Если сопряженное распределение для $Bern(\theta) - Beta(\alpha, \beta)$, то байерсова оценка равна $\hat{\theta} = \frac{\alpha + n\overline{x}}{\alpha+\beta+n}$. Оценка максимального правдоподобия $\hat{\theta} = \overline{X}$.

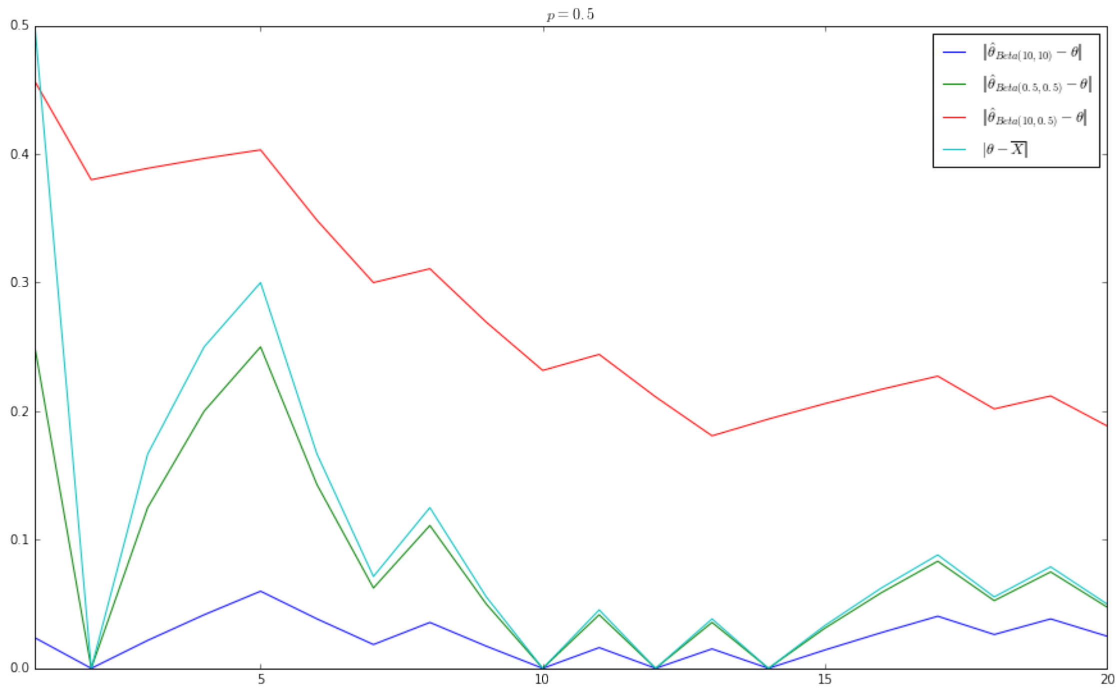
```
In [130]: def estimation(a, b, sample):
              return (a + np.sum(sample)) / (a + b + len(sample));
          def get_data(p, sample, N, a, b):
              return np.abs([(p - estimation(a, b, np.array(sample[:n])))
                                     for n in range(1, N + 1)])
          def maximum_likelihood_data(p, sample, N):
              return np.abs([p - np.average(sample[:n]) for n in range(1, N + 1)])
```

```
In [196]: N = 20
          def draw_plot(p):
              sample = stats.binom.rvs(1, p, size=N)
              grid = np.arange(1, 21, 1)
              plt.figure(figsize=(15, 9))
              plt.title(r'$p={:}$'.format(p))

              data_fair = get_data(p, sample, N, 10, 10)
              data_unfair = get_data(p, sample, N, 0.5, 0.5)
              data_CoA_unfair = get_data(p, sample, N, 10, 0.5)
              data_mle = maximum_likelihood_data(p, sample, N)

              plt.plot(grid, data_fair, label=r'|$\hat{\theta}_{Beta(10, 10)} - \theta$|')
              plt.plot(grid, data_unfair, label=r'|$\hat{\theta}_{Beta(0.5, 0.5)} - \theta$|')
              plt.plot(grid, data_CoA_unfair, label=r'|$\hat{\theta}_{Beta(10, 0.5)} - \theta$|')
              plt.plot(grid, data_mle, label=r'$|\theta - \overline{X}$|')
              plt.legend(loc='best')

              plt.xlim(1, 20)
```

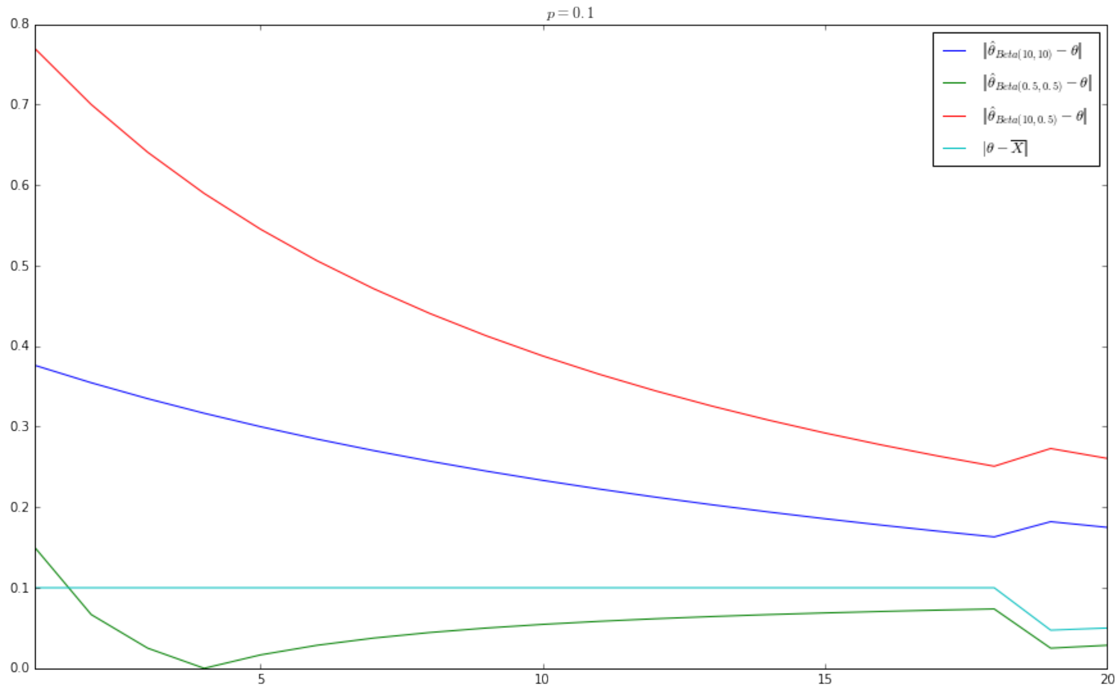
```
plt.show()
```

0.0.1 Построим графики отклонения оценок от истиного значения p

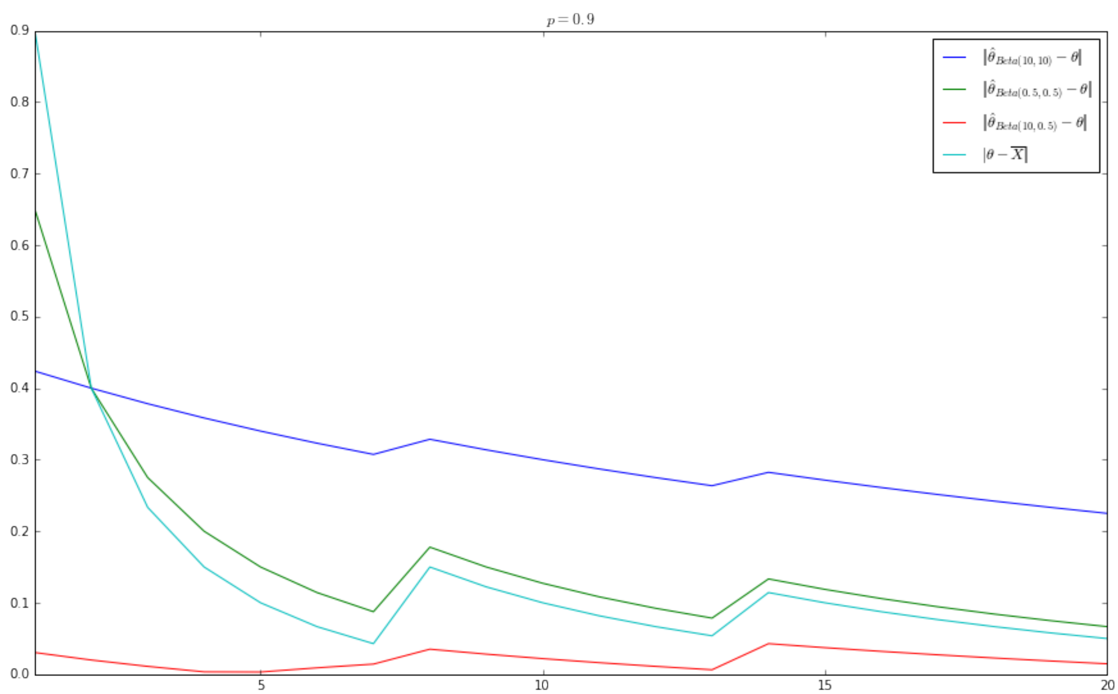
```
In [197]: draw_plot(0.5)
```




```
In [198]: draw_plot(0.1)
```




```
In [200]: draw_plot(0.9)
```



0.0.2 Вывод.

При значении $p = 0.1$ лучше всего показывает оценка 'скорее нечестной монеты' и макс. правдоподобия. При $p = 0.5$ - оценка 'скорее честной монеты', а при $p = 0.9$ - оценка 'нечестной со сдвигом в сторону герба' и макс. правдоподобия

```
In [ ]:
```