4.2

25 апреля 2016 г.

1 Задача 2

In [130]:
```
%matplotlib inline
import numpy as np
import math as mt
import matplotlib
import matplotlib.pyplot as plt
from pylab import *
from scipy.stats import *
from mpl_toolkits.mplot3d import Axes3D
```

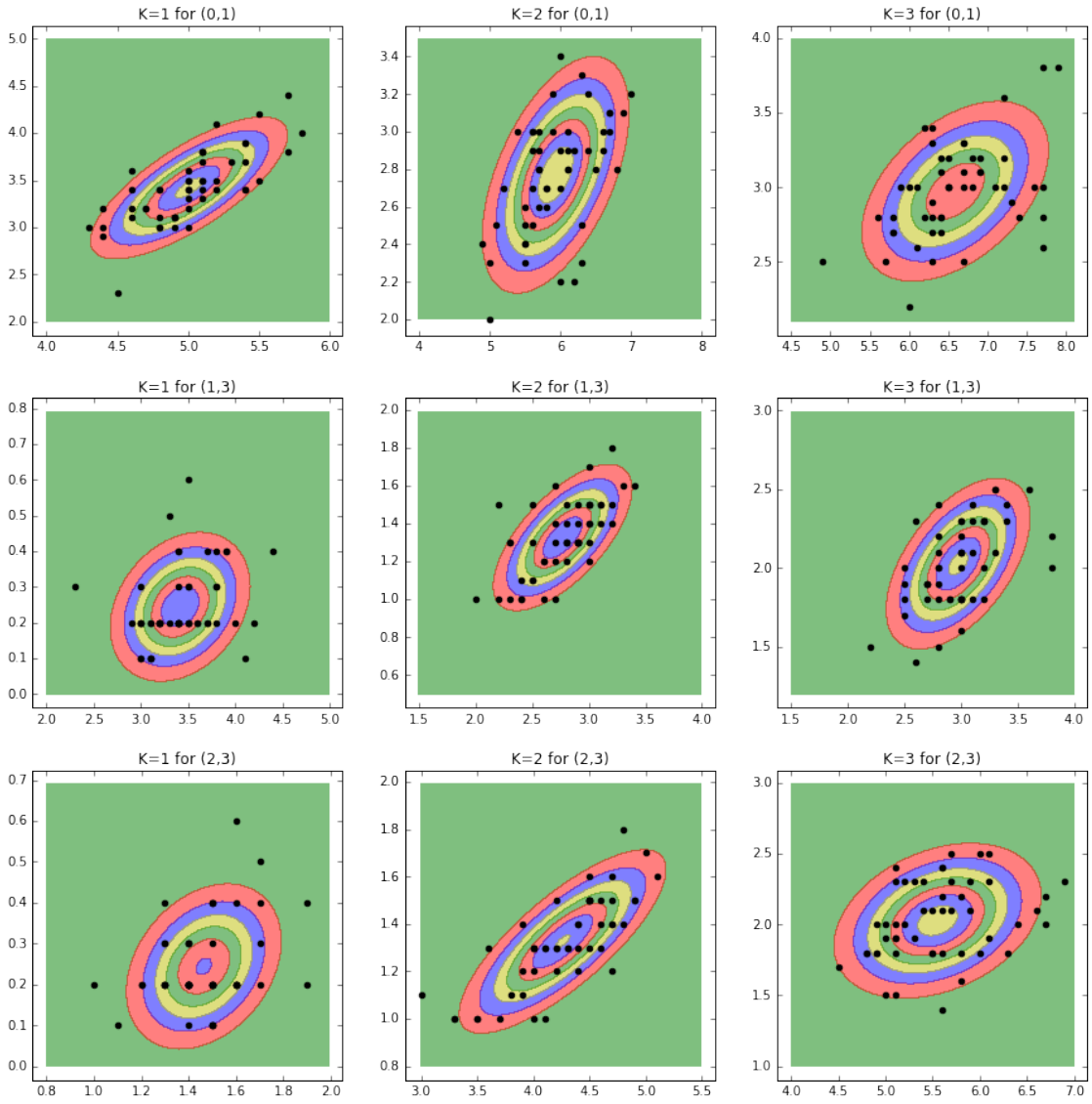
In [131]:
```
from sklearn.datasets import load_iris
data = load_iris()

def COV(x,y):
    return mean(x*y) - (mean(x)*mean(y))

means = []
sigmas = np.zeros((3,4,4))

for i in range(3):
    tmp = []
    for it in range(len(data['data'])):
        if data['target'][it] == i:
            tmp.append(data['data'][it])
    tmp = np.array(tmp)
    means.append([tmp[:,0].mean(),tmp[:,1].mean(),tmp[:,2].mean(),tmp[:,3].mean()])
    for kk in range(4):
        for jj in range(4):
            sigmas[i][kk][jj] = COV(tmp[:,kk],tmp[:,jj])
means = np.array(means)

print 'Матрицы ковариаций:'
print sigmas
print '\nВекторы средних:'
print means
```

```
Матрицы ковариаций:
[[[ 0.121764  0.098292  0.015816  0.010336]
```

```
 [ 0.098292  0.142276  0.011448  0.011208]
 [ 0.015816  0.011448  0.029504  0.005584]
 [ 0.010336  0.011208  0.005584  0.011264]]

[[ 0.261104  0.08348   0.17924   0.054664]
 [ 0.08348   0.0965    0.081     0.04038 ]
 [ 0.17924   0.081     0.2164    0.07164 ]
 [ 0.054664  0.04038   0.07164   0.038324]]

[[ 0.396256  0.091888  0.297224  0.048112]
 [ 0.091888  0.101924  0.069952  0.046676]
 [ 0.297224  0.069952  0.298496  0.047848]
 [ 0.048112  0.046676  0.047848  0.073924]]]

Векторы средних:
[[ 5.006  3.418  1.464  0.244]
 [ 5.936  2.77   4.26   1.326]
 [ 6.588  2.974  5.552  2.026]]
```

In [132]:
```
coords = [[0,1], [1,3], [2,3]]  # Пары координат
means_for_3 = np.zeros((3,3,2))  # Сюда запишу три вектора средних для
                                 # каждой пары координат
sigma_for_3 = np.zeros((3,3,2,2))  # Сюда запишу три матрицы ковариации
                                   # для каждой компоненты для
                                   # каждой пары координат
Xs = [[[],[],[]],[[],[],[]],[[],[],[]]]  # Сюда запишу массивы для каждой из координат
                                         # и для каждой компоненты
for i in range(3):
    ii, jj = coords[i]  # Номера первой и второй координат
    it = 0
    for j in range(3):
        while it < len(data['data']) and data['target'][it] == j:
            Xs[i][j].append(data['data'][it][[ii,jj]])
            it += 1
    Xs[i] = np.array(Xs[i])

    for j in range(3):
        # Здесь считаю матрицу ковариции и вектор средних
        sigma_for_3[i][j][0][0] = mean(Xs[i][j][:,0]*Xs[i][j][:,0])\
            -(mean(Xs[i][j][:,0])*mean(Xs[i][j][:,0]))
        sigma_for_3[i][j][1][1] = mean(Xs[i][j][:,1]*Xs[i][j][:,1])\
            -(mean(Xs[i][j][:,1])*mean(Xs[i][j][:,1]))
        sigma_for_3[i][j][0][1] = mean(Xs[i][j][:,0]*Xs[i][j][:,1])\
            -(mean(Xs[i][j][:,0])*mean(Xs[i][j][:,1]))
        sigma_for_3[i][j][1][0] = sigma_for_3[i][j][0][1]

        means_for_3[i][j][0] = mean(Xs[i][j][:,0])
```

```
        means_for_3[i][j][1] = mean(Xs[i][j][:,1])

# Рисую сетку графиков
plt.close('all')
ax = []
f, ax = plt.subplots(3, 3)
f.set_figheight(15)
f.set_figwidth(15)

# Здесь записаны пределы построения сетки для расчета плотности
limits = [[[4,6,2,5],[4,8,2,3.5],[4.5,8.1,2.1,4]],
          [[2,5,0,0.8],[1.5,4,0.5,2],[1.5,4,1.2,3]],
          [[0.8,2,0,0.7],[3,5.5,0.8,2],[4,7,1,3]]]

# Рисую
for i in range(3):
    for j in range(3):
        x, y = np.mgrid[limits[i][j][0]:limits[i][j][1]:.01, \
                        limits[i][j][2]:limits[i][j][3]:.01]
        pos = np.empty(x.shape + (2,))
        pos[:, :, 0] = x; pos[:, :, 1] = y
        rv = multivariate_normal(means_for_3[i][j], sigma_for_3[i][j])

        ax[i][j].contourf(x, y, rv.pdf(pos), alpha=0.5, colors=('g','r','b','y'))
        ax[i][j].scatter(Xs[i][j][:,0],Xs[i][j][:,1], color='black')
        ax[i][j].set_title('K={} for ({},{})'.format(j+1,coords[i][0],\
                                                     coords[i][1]))
show()
```



2 $E(X|T \neq k)$

Считаю случайные векторы T и X независимыми и строю графики условных плотностей.

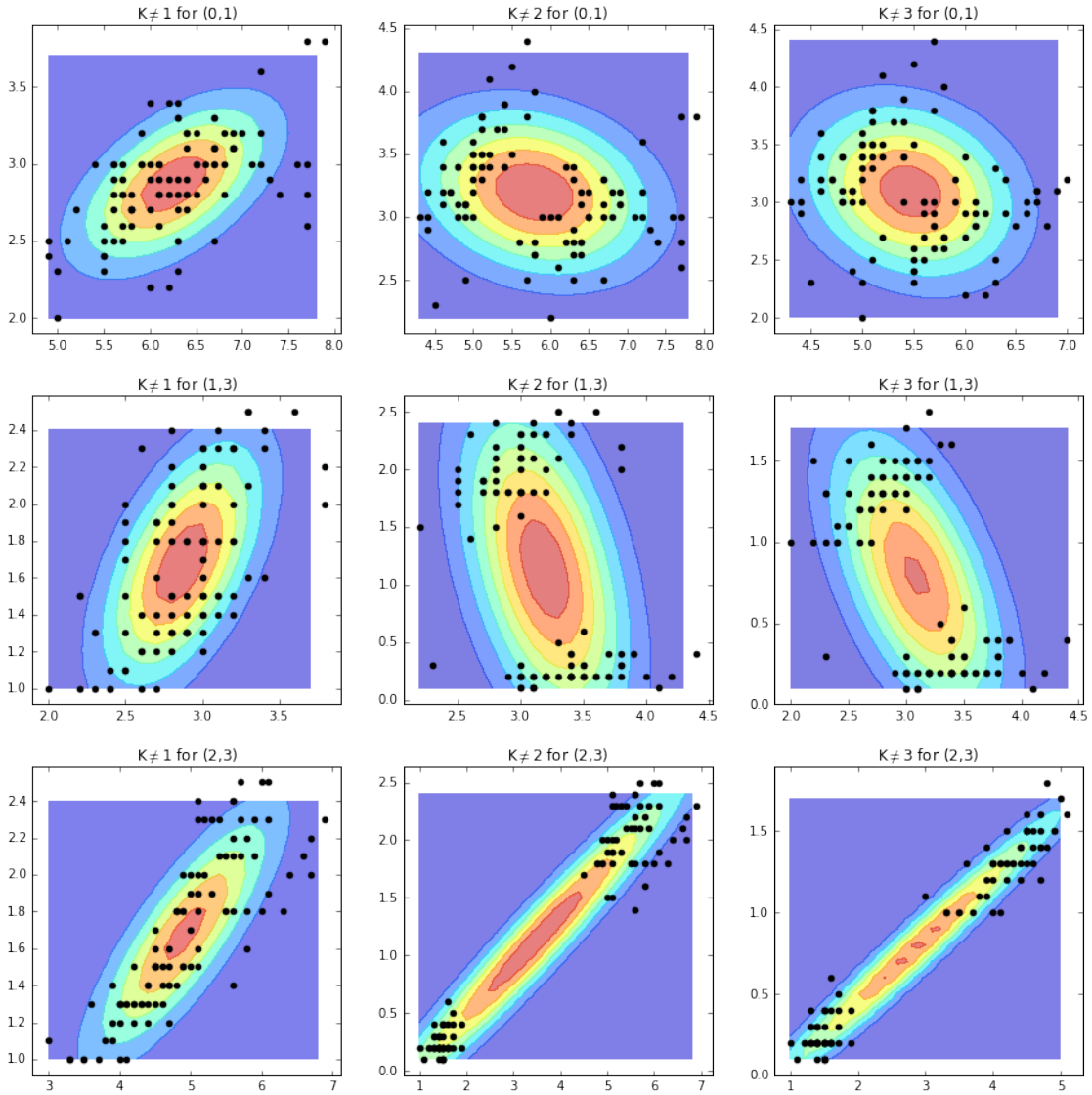
In [133]:
```
means = []
sigmas=[]
for i in range(3):
    tmp = []
    for it in range(len(data['data'])):
        if data['target'][it] != i:
            tmp.append(data['data'][it])
    tmp = np.array(tmp)
```

```
        means.append([mean(tmp[:,0]), mean(tmp[:,1]), mean(tmp[:,2]), mean(tmp[:,3])])
        sigmas.append(cov(tmp, rowvar=0))

    sigmas = np.array(sigmas)
    means = np.array(means)

    print 'Матрицы ковариаций:'
    print sigmas
    print '\nВекторы средних:'
    print means
```

```
Матрицы ковариаций:
[[[ 0.43934949  0.12215758  0.45336162  0.1671596 ]
  [ 0.12215758  0.11072323  0.14279596  0.08002828]
  [ 0.45336162  0.14279596  0.6815798   0.28873131]
  [ 0.1671596   0.08002828  0.28873131  0.18042828]]

 [[ 0.89362727 -0.08132525  1.79123636  0.74141919]
  [-0.08132525  0.17311515 -0.4172404  -0.17056566]
  [ 1.79123636 -0.4172404   4.38579394  1.86658586]
  [ 0.74141919 -0.17056566  1.86658586  0.84492424]]

 [[ 0.41177677 -0.06037778  0.75514949  0.28693434]
  [-0.06037778  0.2266303  -0.41083636 -0.151     ]
  [ 0.75514949 -0.41083636  2.09833939  0.8029596 ]
  [ 0.28693434 -0.151       0.8029596   0.32068182]]]

Векторы средних:
[[ 6.262  2.872  4.906  1.676]
 [ 5.797  3.196  3.508  1.135]
 [ 5.471  3.094  2.862  0.785]]
```

```
In [134]: coords = [[0,1], [1,3], [2,3]]  # Пары координат
          means_for_3 = np.zeros((3,3,2))  # Сюда запишу три вектора средних для
                                           # каждой пары координат
          sigma_for_3 = np.zeros((3,3,2,2))  # Сюда запишу три матрицы ковариации
                                             # для каждой компоненты для
                                             # каждой пары координат
          Xs = [[[],[],[]],[[],[],[]],[[],[],[]]]  # Сюда запишу массивы для каждой из координат
                                                   # и для каждой компоненты
          for i in range(3):
              ii, jj = coords[i]  # Номера первой и второй координат
              it = 0
              for j in range(3):
                  for it in range(len(data['data'])):
                      if data['target'][it] != j:
                          Xs[i][j].append(data['data'][it][[ii,jj]])
```

```
    Xs[i] = np.array(Xs[i])

    for j in range(3):
        # Здесь считаю матрицу ковариции и вектор средних
        sigma_for_3[i][j][0][0] = mean(Xs[i][j][:,0]*Xs[i][j][:,0])\
            -(mean(Xs[i][j][:,0])*mean(Xs[i][j][:,0]))
        sigma_for_3[i][j][1][1] = mean(Xs[i][j][:,1]*Xs[i][j][:,1])\
            -(mean(Xs[i][j][:,1])*mean(Xs[i][j][:,1]))
        sigma_for_3[i][j][0][1] = mean(Xs[i][j][:,0]*Xs[i][j][:,1])\
            -(mean(Xs[i][j][:,0])*mean(Xs[i][j][:,1]))
        sigma_for_3[i][j][1][0] = sigma_for_3[i][j][0][1]

        means_for_3[i][j][0] = mean(Xs[i][j][:,0])
        means_for_3[i][j][1] = mean(Xs[i][j][:,1])

# Рисую сетку графиков
plt.close('all')
ax = []
f, ax = plt.subplots(3, 3)
f.set_figheight(15)
f.set_figwidth(15)

# Рисую
for i in range(3):
    for j in range(3):
        x, y = np.mgrid[min(Xs[i][j][:,0]):max(Xs[i][j][:,0]):0.1, \
                        min(Xs[i][j][:,1]):max(Xs[i][j][:,1]):0.1]
        pos = np.empty(x.shape + (2,))
        pos[:, :, 0] = x; pos[:, :, 1] = y
        rv = multivariate_normal(means_for_3[i][j], sigma_for_3[i][j])

        ax[i][j].contourf(x, y, rv.pdf(pos), alpha=0.5)
        ax[i][j].scatter(Xs[i][j][:,0],Xs[i][j][:,1], color='black')
        ax[i][j].set_title('K$\\neq${} for ({},{})'.format(j+1,coords[i][0],\
                                                coords[i][1]))
show()
```



3 Классифицирую пространство

4 Использую классификатор $k = \arg\max_k p_{(X|I\{T=k\})}(x|1)$.

```
In [135]: def COV(x,y):  # Функция для рассчета ковариации
              return mean(x*y) - (mean(x)*mean(y))

          means = []  # Сюда запишу векторы средних для каждой компоненты k=1,2,3
          sigmas = np.zeros((3,4,4))  # А сюда - матрицы ковариаций

          # Рассчитываю - Записываю
          for i in range(3):
```

```
            tmp = []
            for it in range(len(data['data'])):
                if data['target'][it] == i:
                    tmp.append(data['data'][it])
            tmp = np.array(tmp)
            means.append([tmp[:,0].mean(),tmp[:,1].mean(),tmp[:,2].mean(),tmp[:,3].mean()])
            for kk in range(4):
                for jj in range(4):
                    sigmas[i][kk][jj] = COV(tmp[:,kk],tmp[:,jj])
        means = np.array(means)

        print 'Матрицы ковариаций:'
        print sigmas
        print '\nВекторы средних:'
        print means
        print '\n'
```

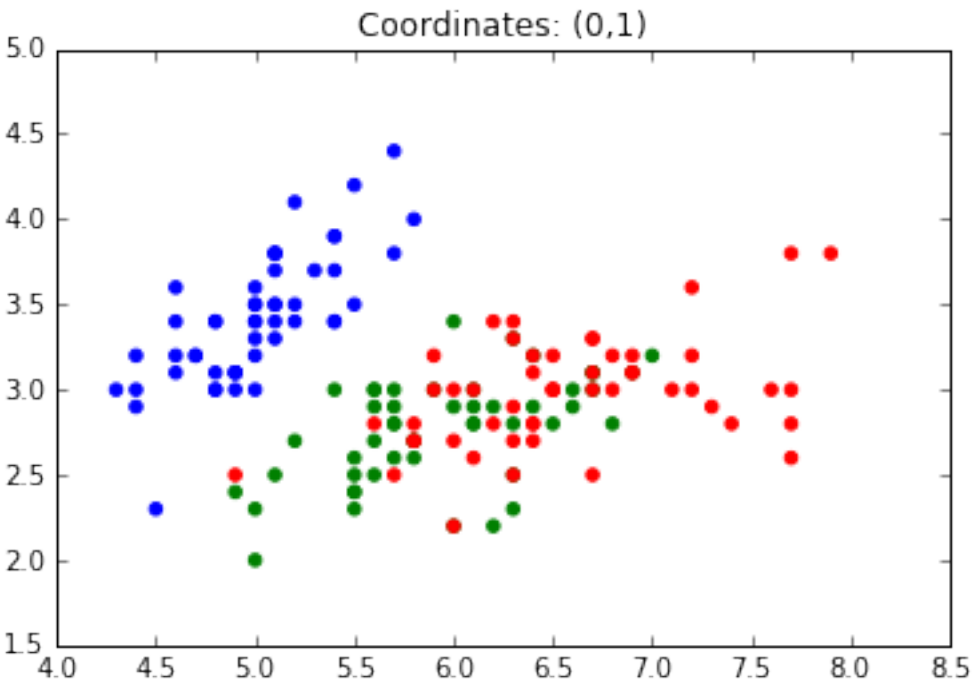
```
Матрицы ковариаций:
[[[ 0.121764  0.098292  0.015816  0.010336]
  [ 0.098292  0.142276  0.011448  0.011208]
  [ 0.015816  0.011448  0.029504  0.005584]
  [ 0.010336  0.011208  0.005584  0.011264]]

 [[ 0.261104  0.08348   0.17924   0.054664]
  [ 0.08348   0.0965    0.081     0.04038 ]
  [ 0.17924   0.081     0.2164    0.07164 ]
  [ 0.054664  0.04038   0.07164   0.038324]]

 [[ 0.396256  0.091888  0.297224  0.048112]
  [ 0.091888  0.101924  0.069952  0.046676]
  [ 0.297224  0.069952  0.298496  0.047848]
  [ 0.048112  0.046676  0.047848  0.073924]]]

Векторы средних:
[[ 5.006  3.418  1.464  0.244]
 [ 5.936  2.77   4.26   1.326]
 [ 6.588  2.974  5.552  2.026]]
```

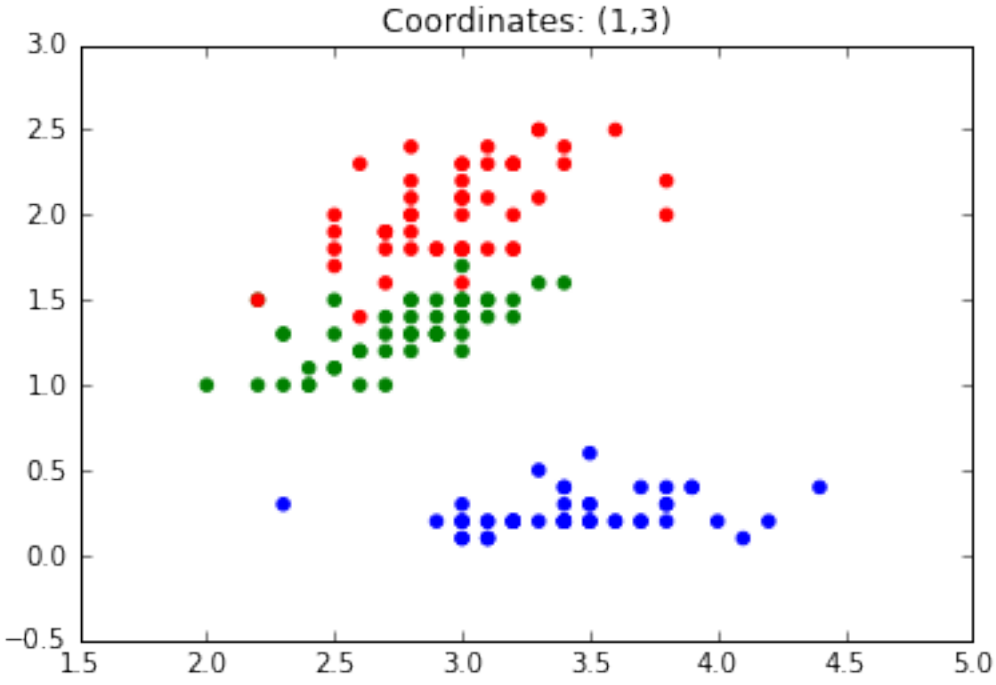
In [139]:
```
def ArgMax(num):
    mymax = -1
    k=0
    for ii in range(3):
        tmp = multivariate_normal.pdf(data['data'][num], means[ii], sigmas[ii])
        if tmp > mymax:
            k = ii
```

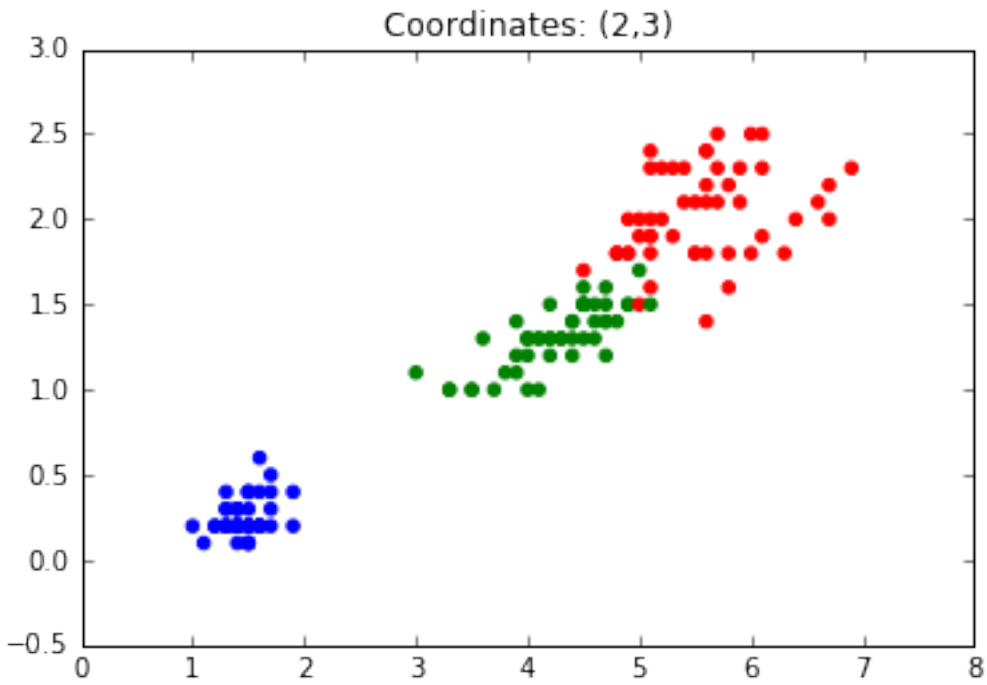
```
            mymax = tmp
    return k

# Применяю функцию ArgMax для всей выборки ирисов
classified = np.array([ArgMax(i) for i in range(len(data['data']))])
print 'Сравнение вектора classified с метками:'
print classified == data['target']

st = Counter(classified == data['target']).most_common()
print '\n'
print "Процент ошибок: {}%".format(float(st[1][1])/float(st[0][1]) * 100)
```

```
Сравнение вектора classified с метками:
[ True  True  True  True  True  True  True  True  True  True  True  True
  True  True  True  True  True  True  True  True  True  True  True  True
  True  True  True  True  True  True  True  True  True  True  True  True
  True  True  True  True  True  True  True  True  True  True  True  True
  True  True  True  True  True  True  True  True  True  True  True  True
  True  True  True  True  True  True  True  True  True  True False  True
  True  True  True  True  True  True  True  True  True  True  True False
  True  True  True  True  True  True  True  True  True  True  True  True
  True  True  True  True  True  True  True  True  True  True  True  True
  True  True  True  True  True  True  True  True  True  True  True  True
  True  True  True  True  True  True  True  True  True  True  True  True
  True False  True  True  True  True  True  True  True  True  True  True
  True  True  True  True  True  True]


Процент ошибок: 2.04081632653%
```

In [140]:
```
# Рисую
cols = ['b','g','r']

Xs = [[],[],[]]

for i in range(3):
    ii, jj = coords[i]  # Номера первой и второй координат
    Xs[i] = data['data'][:,[ii,jj]]
Xs = np.array(Xs)

# Массив цветов, в которые нужно покрасить элементы
colors = [cols[i] for i in classified]

for i in range(3):
    figure()
    plt.scatter(Xs[i][:,0],Xs[i][:,1], color=colors)
    title('Coordinates: ({},{})'.format(coords[i][0], coords[i][1]))
    show()
```


Coordinates: (0,1)
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
4.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0
8.5


Coordinates: (1,3)
3.0
2.5
2.0
1.5
1.0
0.5
0.0
−0.5
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0



5 Использую классификатор $k = \arg\min_k p_{(X|I\{T\neq k\})}(x|1)$.

```
In [141]: def COV(x,y):  # Функция для рассчета ковариации
              return mean(x*y) - (mean(x)*mean(y))

          means = []  # Сюда запишу векторы средних для каждой компоненты k=1,2,3
          sigmas = np.zeros((3,4,4))  # А сюда - матрицы ковариаций

          # Рассчитываю - Записываю
          for i in range(3):
              tmp = []
              for it in range(len(data['data'])):
                  if data['target'][it] != i:
                      tmp.append(data['data'][it])
              tmp = np.array(tmp)
              means.append([tmp[:,0].mean(),tmp[:,1].mean(),tmp[:,2].mean(),tmp[:,3].mean()])
              for kk in range(4):
                  for jj in range(4):
                      sigmas[i][kk][jj] = COV(tmp[:,kk],tmp[:,jj])
          means = np.array(means)

          print 'Матрицы ковариаций:'
          print sigmas
```

```
        print '\nВекторы средних:'
        print means
        print '\n'
```

```
Матрицы ковариаций:
[[[ 0.434956  0.120936  0.448828  0.165488]
  [ 0.120936  0.109616  0.141368  0.079228]
  [ 0.448828  0.141368  0.674764  0.285844]
  [ 0.165488  0.079228  0.285844  0.178624]]

 [[ 0.884691 -0.080512  1.773324  0.734005]
  [-0.080512  0.171384 -0.413068 -0.16886 ]
  [ 1.773324 -0.413068  4.341936  1.84792 ]
  [ 0.734005 -0.16886   1.84792   0.836475]]

 [[ 0.407659 -0.059774  0.747598  0.284065]
  [-0.059774  0.224364 -0.406728 -0.14949 ]
  [ 0.747598 -0.406728  2.077356  0.79493 ]
  [ 0.284065 -0.14949   0.79493   0.317475]]]

Векторы средних:
[[ 6.262  2.872  4.906  1.676]
 [ 5.797  3.196  3.508  1.135]
 [ 5.471  3.094  2.862  0.785]]
```

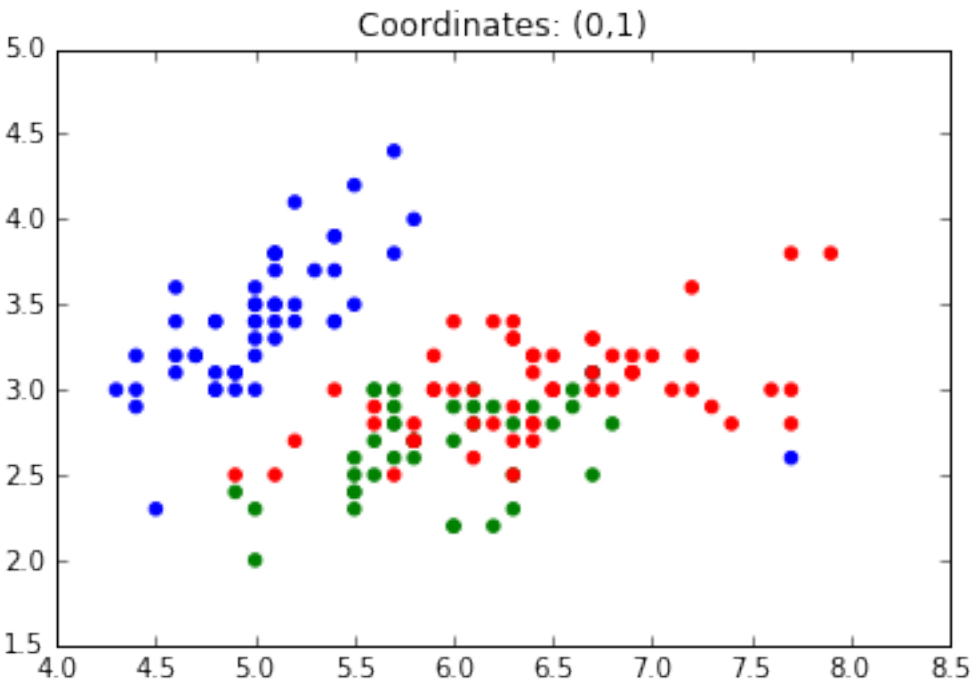
In [142]:
```
def ArgMin(num):
    mymax = 1000000
    k=0
    for ii in range(3):
        tmp = multivariate_normal.pdf(data['data'][num], means[ii], sigmas[ii])
        if tmp < mymax:
            k = ii
            mymax = tmp
    return k

# Применяю функцию ArgMin для всей выборки ирисов
classified = np.array([ArgMin(i) for i in range(len(data['data']))])
print 'Сравнение вектора classified с метками:'
print classified == data['target']

st = Counter(classified == data['target']).most_common()
print '\n'
print "Процент ошибок: {}%".format(float(st[1][1])/float(st[0][1]) * 100)
```

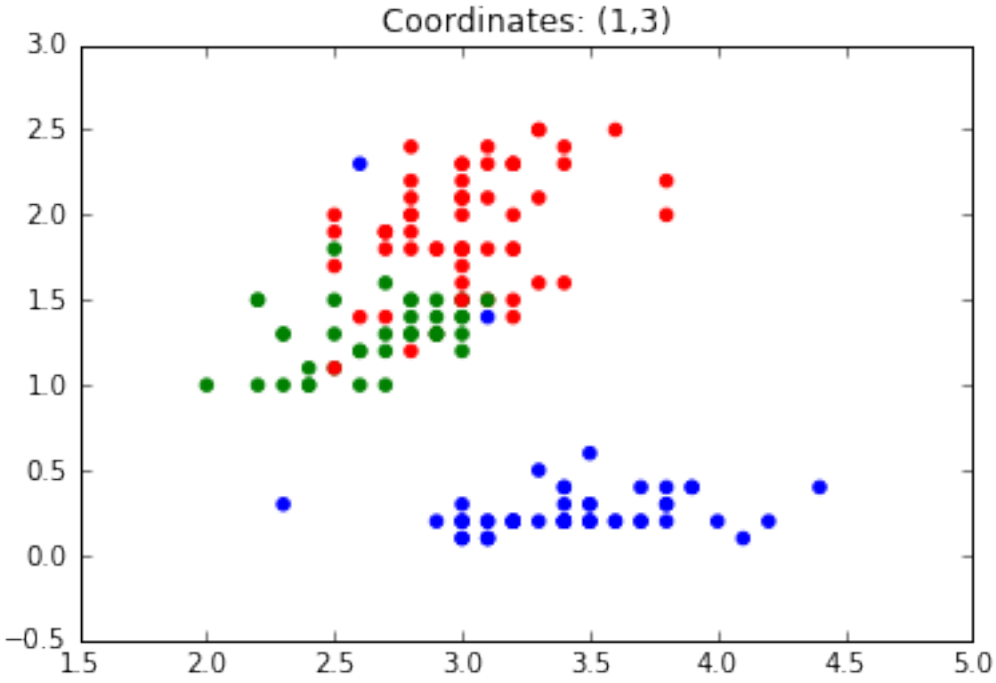
```
Сравнение вектора classified с метками:
[ True  True  True  True  True  True  True  True  True  True  True  True
```

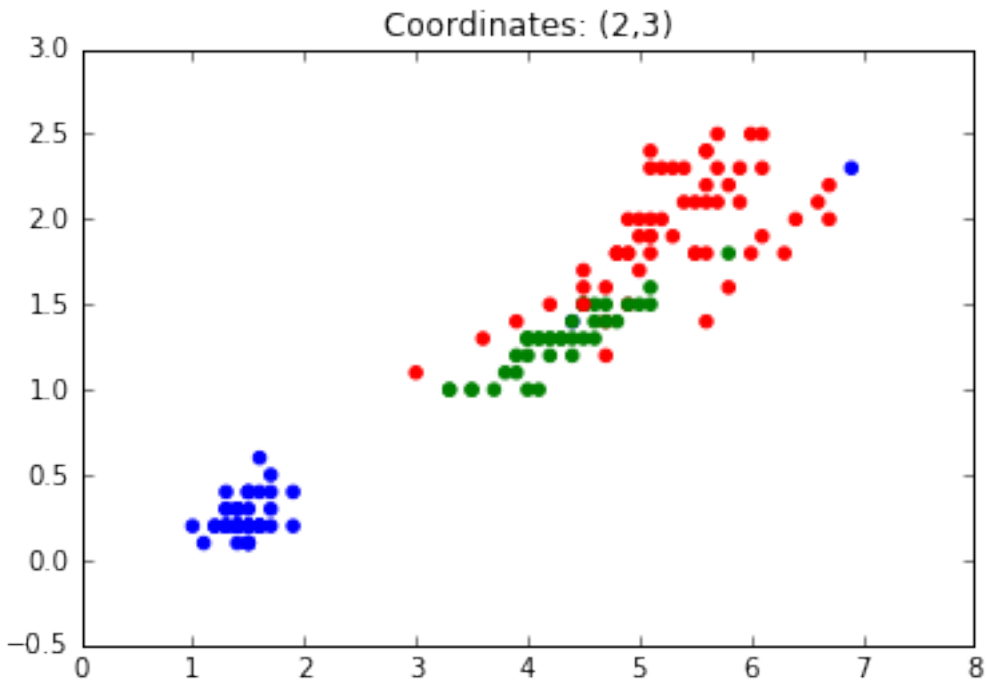
```
 True  True  True  True  True  True  True  True  True  True  True  True
 True  True  True  True  True  True  True  True  True  True  True  True
 True  True  True  True  True  True  True  True  True  True  True  True
 True  True False False False  True  True  True False  True  True False
 True False  True  True False False  True  True  True  True False  True
 True False  True  True  True False  True  True  True  True  True  True
False False  True  True  True  True  True  True  True  True  True  True
 True  True False  True  True  True  True  True  True  True  True  True
False  True  True  True  True  True  True  True  True  True False False
 True  True  True  True  True  True  True  True  True  True  True  True
 True False  True  True  True  True  True  True  True  True  True  True
 True  True  True  True  True  True]
```

Процент ошибок: 13.6363636364%

In [143]:
```
# Рисую
cols = ['b','g','r']

Xs = [[],[],[]]

for i in range(3):
    ii, jj = coords[i]  # Номера первой и второй координат
    Xs[i] = data['data'][:,[ii,jj]]
Xs = np.array(Xs)

# Массив цветов, в которые нужно покрасить элементы
colors = [cols[i] for i in classified]

for i in range(3):
    figure()
    plt.scatter(Xs[i][:,0],Xs[i][:,1], color=colors)
    title('Coordinates: ({},{})'.format(coords[i][0], coords[i][1]))
    show()
```


Coordinates: (0,1)
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
4.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0
8.5


Coordinates: (1,3)
3.0
2.5
2.0
1.5
1.0
0.5
0.0
−0.5
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0



В итоге, мне удалось классифицировать четырехмерное пространство векторов. И наиболее точным классификатором оказался argmax с процентом ошибок в 2% против классификатора argmin - 13%.