In [1]:
```
from matplotlib.colors import ListedColormap
from sklearn import datasets, metrics, neighbors
import sklearn.model_selection as ms

import numpy as np
```

In [3]:
```
%pylab inline
```

```
Populating the interactive namespace from numpy and matplotlib
```

In [22]:
```
digits = datasets.load_digits()
```

In [23]:
```
breast_cancer = datasets.load_breast_cancer()
```

Digits

In [45]:
```
digits
```

Out[45]:
```
{'DESCR': "Optical Recognition of Handwritten Digits Data Set\n=====
=============================================\n\nNotes\n-----\nData
Set Characteristics:\n    :Number of Instances: 5620\n    :Number of
Attributes: 64\n    :Attribute Information: 8x8 image of integer pix
els in the range 0..16.\n    :Missing Attribute Values: None\n    :C
reator: E. Alpaydin (alpaydin '@' boun.edu.tr)\n    :Date: July; 199
8\n\nThis is a copy of the test set of the UCI ML hand-written digit
s datasets\nhttp://archive.ics.uci.edu/ml/datasets/Optical+Recogniti
on+of+Handwritten+Digits\n\nThe data set contains images of hand-wri
tten digits: 10 classes where\neach class refers to a digit.\n\nPrep
rocessing programs made available by NIST were used to extract\nnorm
alized bitmaps of handwritten digits from a preprinted form. From a\
ntotal of 43 people, 30 contributed to the training set and differen
t 13\nto the test set. 32x32 bitmaps are divided into nonoverlapping
blocks of\n4x4 and the number of on pixels are counted in each block
. This generates\nan input matrix of 8x8 where each element is an in
teger in the range\n0..16. This reduces dimensionality and gives inv
ariance to small\ndistortions.\n\nFor info on NIST preprocessing rou
tines, see M. D. Garris, J. L. Blue, G.\nT. Candela, D. L. Dimmick,
J. Geist, P. J. Grother, S. A. Janet, and C.\nL. Wilson, NIST Form-B
ased Handprint Recognition System, NISTIR 5469,\n1994.\n\nReferences
\n----------\n  - C. Kaynak (1995) Methods of Combining Multiple Cla
ssifiers and Their\n    Applications to Handwritten Digit Recognitio
n, MSc Thesis, Institute of\n    Graduate Studies in Science and Eng
ineering, Bogazici University.\n  - E. Alpaydin, C. Kaynak (1998) Ca
scading Classifiers, Kybernetika.\n  - Ken Tang and Ponnuthurai N. S
uganthan and Xi Yao and A. Kai Qin.\n    Linear dimensionalityreduct
ion using relevance weighted LDA. School of\n    Electrical and Elec
tronic Engineering Nanyang Technological University.\n    2005.\n  -
```

```
Claudio Gentile. A New Approximate Maximal Margin Classification\n
Algorithm. NIPS. 2000.\n",
 'data': array([[  0.,   0.,   5., ...,   0.,   0.,   0.],
       [  0.,   0.,   0., ...,  10.,   0.,   0.],
       [  0.,   0.,   0., ...,  16.,   9.,   0.],
       ...,
       [  0.,   0.,   1., ...,   6.,   0.,   0.],
       [  0.,   0.,   2., ...,  12.,   0.,   0.],
       [  0.,   0.,  10., ...,  12.,   1.,   0.]]),
 'images': array([[[  0.,   0.,   5., ...,   1.,   0.,   0.],
        [  0.,   0.,  13., ...,  15.,   5.,   0.],
        [  0.,   3.,  15., ...,  11.,   8.,   0.],
        ...,
        [  0.,   4.,  11., ...,  12.,   7.,   0.],
        [  0.,   2.,  14., ...,  12.,   0.,   0.],
        [  0.,   0.,   6., ...,   0.,   0.,   0.]],

       [[  0.,   0.,   0., ...,   5.,   0.,   0.],
        [  0.,   0.,   0., ...,   9.,   0.,   0.],
        [  0.,   0.,   3., ...,   6.,   0.,   0.],
        ...,
        [  0.,   0.,   1., ...,   6.,   0.,   0.],
        [  0.,   0.,   1., ...,   6.,   0.,   0.],
        [  0.,   0.,   0., ...,  10.,   0.,   0.]],

       [[  0.,   0.,   0., ...,  12.,   0.,   0.],
        [  0.,   0.,   3., ...,  14.,   0.,   0.],
        [  0.,   0.,   8., ...,  16.,   0.,   0.],
        ...,
        [  0.,   9.,  16., ...,   0.,   0.,   0.],
        [  0.,   3.,  13., ...,  11.,   5.,   0.],
        [  0.,   0.,   0., ...,  16.,   9.,   0.]],

       ...,
       [[  0.,   0.,   1., ...,   1.,   0.,   0.],
        [  0.,   0.,  13., ...,   2.,   1.,   0.],
        [  0.,   0.,  16., ...,  16.,   5.,   0.],
        ...,
        [  0.,   0.,  16., ...,  15.,   0.,   0.],
        [  0.,   0.,  15., ...,  16.,   0.,   0.],
        [  0.,   0.,   2., ...,   6.,   0.,   0.]],

       [[  0.,   0.,   2., ...,   0.,   0.,   0.],
        [  0.,   0.,  14., ...,  15.,   1.,   0.],
        [  0.,   4.,  16., ...,  16.,   7.,   0.],
        ...,
        [  0.,   0.,   0., ...,  16.,   2.,   0.],
        [  0.,   0.,   4., ...,  16.,   2.,   0.],
        [  0.,   0.,   5., ...,  12.,   0.,   0.]],

       [[  0.,   0.,  10., ...,   1.,   0.,   0.],
        [  0.,   2.,  16., ...,   1.,   0.,   0.],
        [  0.,   0.,  15., ...,  15.,   0.,   0.],
        ...,
```

```
        [  0.,   4.,  16., ...,  16.,   6.,   0.],
        [  0.,   8.,  16., ...,  16.,   8.,   0.],
        [  0.,   1.,   8., ...,  12.,   1.,   0.]]]),
 'target': array([0, 1, 2, ..., 8, 9, 8]),
 'target_names': array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])}
```

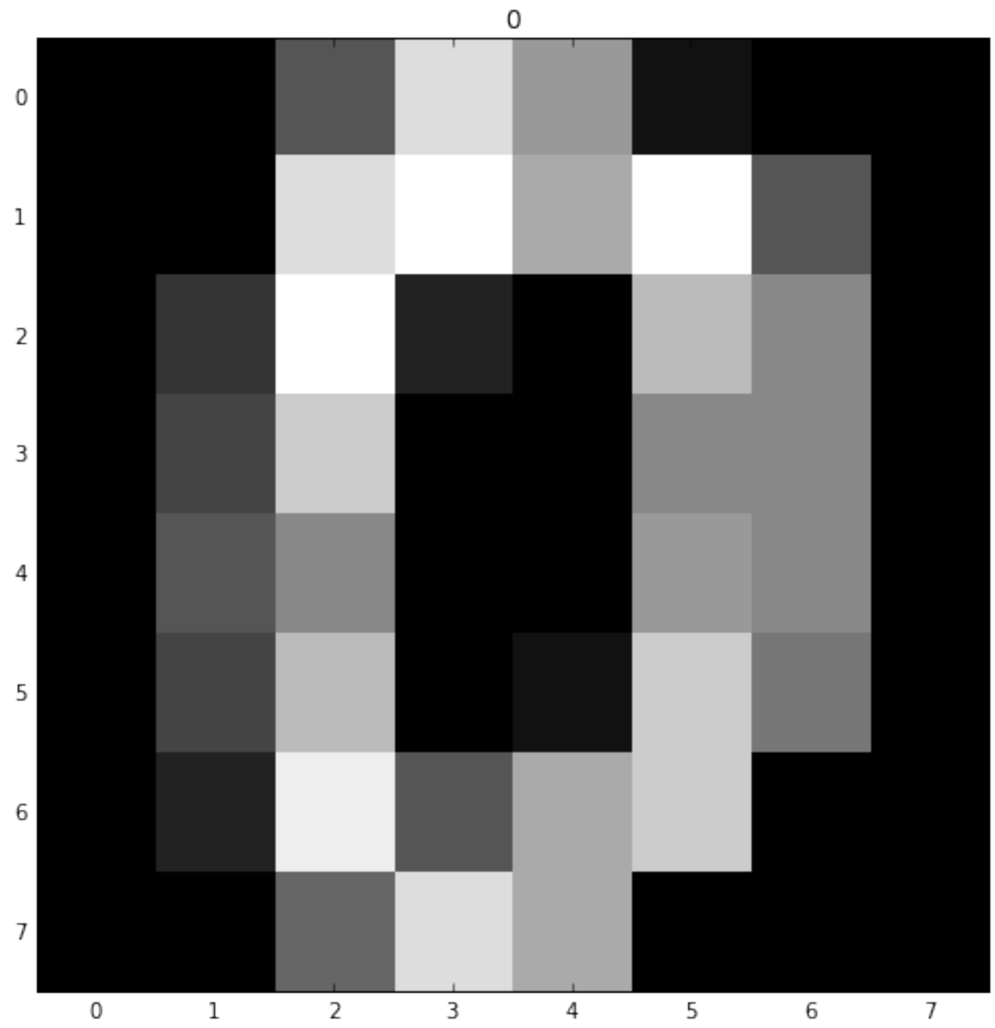
В датасете цифры:

In [46]:
```
plt.figure(figsize = (8, 8))

plt.imshow(digits.images[0], cmap = 'gray', interpolation = 'nearest')

plt.title(str(digits.target[0]))
```

Out[46]: <matplotlib.text.Text at 0x112f3c7d0>



In [47]:
```
digits.data[:50]
```

Out[47]:
```
array([[  0.,   0.,   5., ...,   0.,   0.,   0.],
       [  0.,   0.,   0., ...,  10.,   0.,   0.],
       [  0.,   0.,   0., ...,  16.,   9.,   0.],
       ...,
       [  0.,   0.,   0., ...,   6.,   0.,   0.],
       [  0.,   0.,   2., ...,   6.,   0.,   0.],
       [  0.,   0.,   1., ...,   8.,   0.,   0.]])
```

In [48]:
```
digits.target[:50]
```

Out[48]:
```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0
, 1, 2,
       3, 4, 5, 6, 7, 8, 9, 0, 9, 5, 5, 6, 5, 0, 9, 8, 9, 8, 4, 1, 7
, 7, 3,
       5, 1, 0, 0])
```

Breast_cancer

In [44]:
```
breast_cancer
```

Out[44]:
```
{'DESCR': 'Breast Cancer Wisconsin (Diagnostic) Database\n==========
===================================\n\nNotes\n-----\nData Set Charac
teristics:\n    :Number of Instances: 569\n\n    :Number of Attribut
es: 30 numeric, predictive attributes and the class\n\n    :Attribut
e Information:\n        - radius (mean of distances from center to p
oints on the perimeter)\n        - texture (standard deviation of gr
ay-scale values)\n        - perimeter\n        - area\n        - smo
othness (local variation in radius lengths)\n        - compactness (
perimeter^2 / area - 1.0)\n        - concavity (severity of concave 
portions of the contour)\n        - concave points (number of concav
e portions of the contour)\n        - symmetry \n        - fractal d
imension ("coastline approximation" - 1)\n\n        The mean, standa
rd error, and "worst" or largest (mean of the three\n        largest
values) of these features were computed for each image,\n        res
ulting in 30 features.  For instance, field 3 is Mean Radius, field\
n        13 is Radius SE, field 23 is Worst Radius.\n\n        - cla
ss:\n                - WDBC-Malignant\n                - WDBC-Benign
\n\n    :Summary Statistics:\n\n    =================================
===== ====== ======\n                                           Min 
Max\n    =================================== ====== ======\n    ra
dius (mean):                         6.981  28.11\n    texture (mean)
:                      9.71   39.28\n    perimeter (mean):          
43.79  188.5\n    area (mean):                          143.5  2501.
0\n    smoothness (mean):                     0.053  0.163\n    compa
ctness (mean):                  0.019  0.345\n    concavity (mean):
0.0    0.427\n    concave points (mean):                0.0    0.201
\n    symmetry (mean):                       0.106  0.304\n    fracta
l dimension (mean):              0.05   0.097\n    radius (standard e
rror):              0.112  2.873\n    texture (standard error):
```

```
0.36   4.885\n    perimeter (standard error):           0.757  21.98
\n    area (standard error):                6.802  542.2\n    smooth
ness (standard error):          0.002  0.031\n    compactness (stand
ard error):         0.002  0.135\n    concavity (standard error):
0.0    0.396\n    concave points (standard error):      0.0    0.053
\n    symmetry (standard error):            0.008  0.079\n    fracta
l dimension (standard error):   0.001  0.03\n    radius (worst):
7.93   36.04\n    texture (worst):                      12.02  49.54
\n    perimeter (worst):                    50.41  251.2\n    area (
worst):                         185.2  4254.0\n    smoothness (worst
):                  0.071  0.223\n    compactness (worst):
0.027  1.058\n    concavity (worst):                    0.0    1.252
\n    concave points (worst):               0.0    0.291\n    symmet
ry (worst):                     0.156  0.664\n    fractal dimension
(worst):            0.055  0.208\n    ==============================
======= ====== ======\n\n    :Missing Attribute Values: None\n\n
:Class Distribution: 212 - Malignant, 357 - Benign\n\n    :Creator:
Dr. William H. Wolberg, W. Nick Street, Olvi L. Mangasarian\n\n    :
Donor: Nick Street\n\n    :Date: November, 1995\n\nThis is a copy of
UCI ML Breast Cancer Wisconsin (Diagnostic) datasets.\nhttps://goo.g
l/U2Uwz2\n\nFeatures are computed from a digitized image of a fine n
eedle\naspirate (FNA) of a breast mass.  They describe\ncharacterist
ics of the cell nuclei present in the image.\n\nSeparating plane des
cribed above was obtained using\nMultisurface Method-Tree (MSM-T) [K
. P. Bennett, "Decision Tree\nConstruction Via Linear Programming."
Proceedings of the 4th\nMidwest Artificial Intelligence and Cognitiv
e Science Society,\npp. 97-101, 1992], a classification method which
uses linear\nprogramming to construct a decision tree.  Relevant fea
tures\nwere selected using an exhaustive search in the space of 1-4\
nfeatures and 1-3 separating planes.\n\nThe actual linear program us
ed to obtain the separating plane\nin the 3-dimensional space is tha
t described in:\n[K. P. Bennett and O. L. Mangasarian: "Robust Linea
r\nProgramming Discrimination of Two Linearly Inseparable Sets",\nOp
timization Methods and Software 1, 1992, 23-34].\n\nThis database is
also available through the UW CS ftp server:\n\nftp ftp.cs.wisc.edu\
ncd math-prog/cpo-dataset/machine-learn/WDBC/\n\nReferences\n-------
---\n   - W.N. Street, W.H. Wolberg and O.L. Mangasarian. Nuclear fe
ature extraction \n     for breast tumor diagnosis. IS&T/SPIE 1993 I
nternational Symposium on \n     Electronic Imaging: Science and Tec
hnology, volume 1905, pages 861-870,\n     San Jose, CA, 1993.\n   -
O.L. Mangasarian, W.N. Street and W.H. Wolberg. Breast cancer diagno
sis and \n     prognosis via linear programming. Operations Research
, 43(4), pages 570-577, \n     July-August 1995.\n   - W.H. Wolberg,
W.N. Street, and O.L. Mangasarian. Machine learning techniques\n
to diagnose breast cancer from fine-needle aspirates. Cancer Letters
77 (1994) \n     163-171.\n',
 'data': array([[  1.79900000e+01,   1.03800000e+01,   1.22800000e+0
2, ...,
           2.65400000e-01,   4.60100000e-01,   1.18900000e-01],
        [  2.05700000e+01,   1.77700000e+01,   1.32900000e+02, ...,
           1.86000000e-01,   2.75000000e-01,   8.90200000e-02],
        [  1.96900000e+01,   2.12500000e+01,   1.30000000e+02, ...,
           2.43000000e-01,   3.61300000e-01,   8.75800000e-02],
        ...,
```

```
        [  1.66000000e+01,   2.80800000e+01,   1.08300000e+02, ...,
           1.41800000e-01,   2.21800000e-01,   7.82000000e-02],
        [  2.06000000e+01,   2.93300000e+01,   1.40100000e+02, ...,
           2.65000000e-01,   4.08700000e-01,   1.24000000e-01],
        [  7.76000000e+00,   2.45400000e+01,   4.79200000e+01, ...,
           0.00000000e+00,   2.87100000e-01,   7.03900000e-02]]),
 'feature_names': array(['mean radius', 'mean texture', 'mean perime
ter', 'mean area',
        'mean smoothness', 'mean compactness', 'mean concavity',
        'mean concave points', 'mean symmetry', 'mean fractal dimens
ion',
        'radius error', 'texture error', 'perimeter error', 'area er
ror',
        'smoothness error', 'compactness error', 'concavity error',
        'concave points error', 'symmetry error', 'fractal dimension
error',
        'worst radius', 'worst texture', 'worst perimeter', 'worst a
rea',
        'worst smoothness', 'worst compactness', 'worst concavity',
        'worst concave points', 'worst symmetry', 'worst fractal dim
ension'],
       dtype='|S23'),
 'target': array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 1, 1, 1, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
0, 0, 0,
        1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0,
1, 1, 1,
        1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1,
1, 1, 0,
        1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1,
1, 1, 1,
        1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0,
0, 1, 1,
        0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1,
1, 1, 1,
        0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0,
0, 0, 1,
        0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0,
1, 0, 1,
        0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1,
1, 1, 0,
        0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1,
0, 1, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0,
1, 0, 1,
        1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 0, 1,
        1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1,
1, 1, 0,
        1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1,
1, 0, 1,
        1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
```

```
0, 0, 1,
        0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1,
1, 0, 1,
        1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1,
1, 1, 1,
        0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0,
1, 0, 1,
        1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1,
1, 1, 1,
        0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0,
1, 1, 1,
        1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1,
0, 1, 1,
        1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1,
1, 1, 1,
        1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1]),
 'target_names': array(['malignant', 'benign'],
      dtype='|S9')}
```

In [49]:
```
breast_cancer.data[:50]
```

Out[49]:
```
array([[  1.79900000e+01,   1.03800000e+01,   1.22800000e+02, ...,
          2.65400000e-01,   4.60100000e-01,   1.18900000e-01],
       [  2.05700000e+01,   1.77700000e+01,   1.32900000e+02, ...,
          1.86000000e-01,   2.75000000e-01,   8.90200000e-02],
       [  1.96900000e+01,   2.12500000e+01,   1.30000000e+02, ...,
          2.43000000e-01,   3.61300000e-01,   8.75800000e-02],
       ...,
       [  1.31700000e+01,   1.86600000e+01,   8.59800000e+01, ...,
          2.08800000e-01,   3.90000000e-01,   1.17900000e-01],
       [  1.20500000e+01,   1.46300000e+01,   7.80400000e+01, ...,
          6.54800000e-02,   2.74700000e-01,   8.30100000e-02],
       [  1.34900000e+01,   2.23000000e+01,   8.69100000e+01, ...,
          1.28200000e-01,   2.87100000e-01,   6.91700000e-02]])
```

In [50]:
```
breast_cancer.target[:50]
```

Out[50]:
```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1
, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0
, 0, 0,
       1, 0, 1, 1])
```

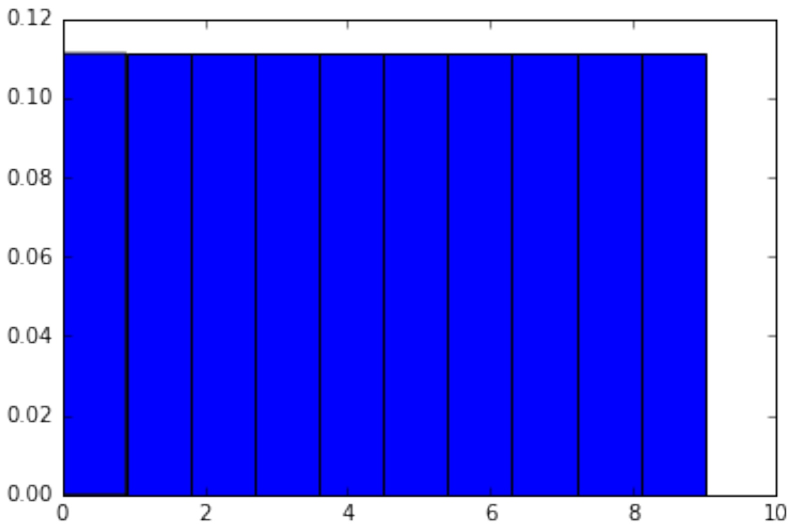
Детальный анализ данных

Посмотрим более детально на структуру данных

In [60]:
```
from collections import Counter
```

In [69]:
```
plt.hist(list(Counter(digits.target)), normed=True)
```

Out[69]:
```
(array([ 0.11111111,  0.11111111,  0.11111111,  0.11111111,  0.11111
111,
         0.11111111,  0.11111111,  0.11111111,  0.11111111,  0.11111
111]),
 array([ 0. ,  0.9,  1.8,  2.7,  3.6,  4.5,  5.4,  6.3,  7.2,  8.1,
9. ]),
 <a list of 10 Patch objects>)
```



Классы полностью сбалансированы

Data представляет из себя вектор-строку записанных подряд битов изображения:

In [137]:
```
print digits.data[3]
print digits.images[3]
```

```
[  0.   0.   7.  15.  13.   1.   0.   0.   0.   8.  13.   6.  15.
4.   0.
   0.   0.   2.   1.  13.  13.   0.   0.   0.   0.   0.   2.  15.  1
1.   1.
   0.   0.   0.   0.   0.   1.  12.  12.   1.   0.   0.   0.   0.
0.   1.
  10.   8.   0.   0.   0.   8.   4.   5.  14.   9.   0.   0.   0.
7.  13.
  13.   9.   0.   0.]
[[  0.   0.   7.  15.  13.   1.   0.   0.]
 [  0.   8.  13.   6.  15.   4.   0.   0.]
 [  0.   2.   1.  13.  13.   0.   0.   0.]
 [  0.   0.   2.  15.  11.   1.   0.   0.]
 [  0.   0.   0.   1.  12.  12.   1.   0.]
 [  0.   0.   0.   0.   1.  10.   8.   0.]
 [  0.   0.   8.   4.   5.  14.   9.   0.]
 [  0.   0.   7.  13.  13.   9.   0.   0.]]
```

Поскольку исходов у digits более двух, и все они дискретны, то среди BernoulliNB, Поскольку исходов у digits более двух, и все они дискретны, то среди BernoulliNB, MultinomialNB и GaussianNB лучшую оценку будет давать MultinomialNB

In [138]:
```
Counter(breast_cancer.target)
```

Out[138]:
```
Counter({0: 212, 1: 357})
```

Классы немного несбалансированы, более того 'самый плохой' estimator, который будет всегда говорить нет (0), будет иметь точность:

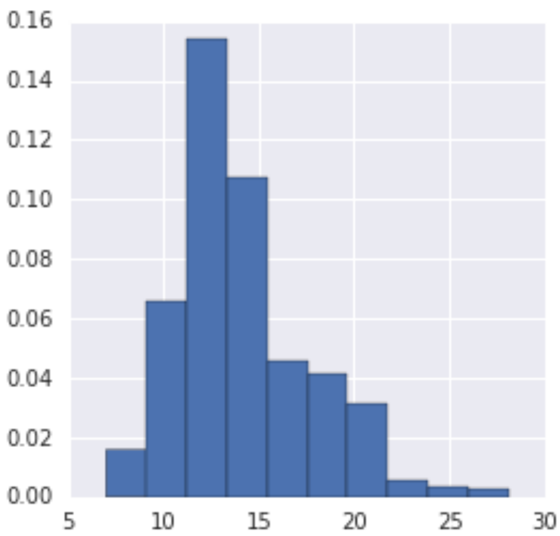
In [139]:
```
Counter(breast_cancer.target)[0] / float(Counter(breast_cancer.target)
```

Out[139]:
```
0.5938375350140056
```

Учитывая, что многие данные breast_cancer непрерывные, то среди BernoulliNB, MultinomialNB и GaussianNB лучшую оценку будет давать GaussianNB

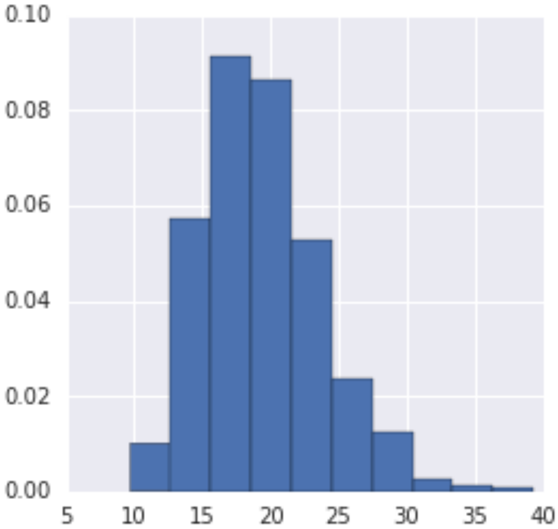
In [140]:
```
plt.figure(figsize=(4, 4))
hist(breast_cancer.data[:, 0], normed=True)
```

Out[140]:
```
(array([ 0.01580383,  0.06571067,  0.15387941,  0.1072997 ,  0.04574
793,
         0.04158903,  0.03160766,  0.00582246,  0.00332712,  0.00249
534]),
 array([  6.981 ,   9.0939,  11.2068,  13.3197,  15.4326,  17.5455,
         19.6584,  21.7713,  23.8842,  25.9971,  28.11  ]),
 <a list of 10 Patch objects>)
```



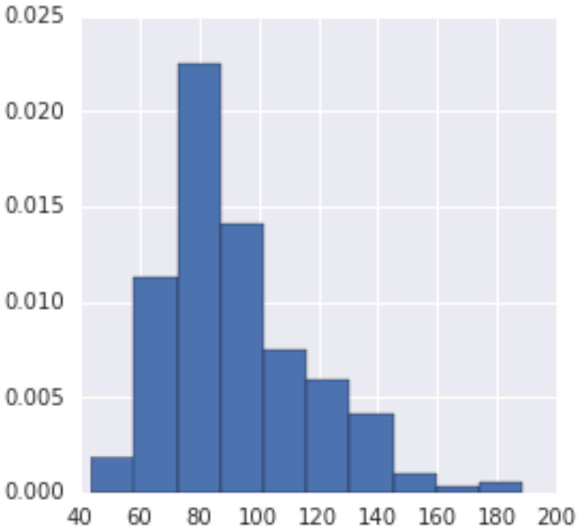
In [141]:
```
plt.figure(figsize=(4, 4))
hist(breast_cancer.data[:, 1], normed=True)
```

Out[141]:
```
(array([ 0.01010381,  0.05705683,  0.09152867,  0.08617959,  0.05289
644,
         0.02377368,  0.01248118,  0.00237737,  0.00118868,  0.00059
434]),
 array([  9.71 ,  12.667,  15.624,  18.581,  21.538,  24.495,  27.45
2,
         30.409,  33.366,  36.323,  39.28 ]),
 <a list of 10 Patch objects>)
```



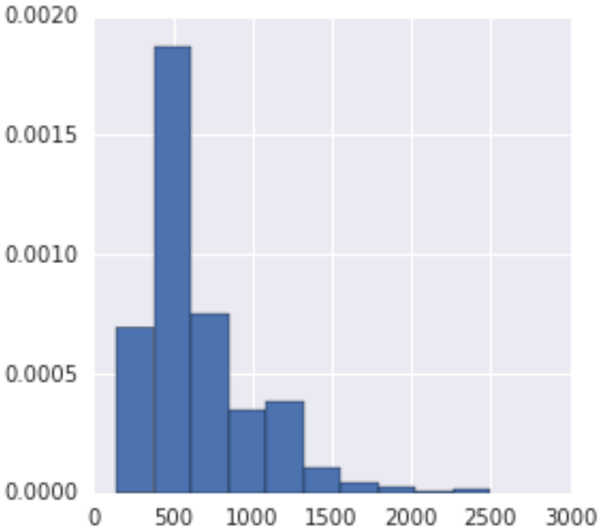
In [142]:
```
plt.figure(figsize=(4, 4))
hist(breast_cancer.data[:, 2], normed=True)
```

Out[142]:
```
(array([ 0.00182172,  0.01129463,  0.02246782,  0.01408793,  0.00752
976,
         0.00595094,  0.00412922,  0.00097158,  0.00036434,  0.00048
579]),
 array([  43.79 ,   58.261,   72.732,   87.203,  101.674,  116.145,
         130.616,  145.087,  159.558,  174.029,  188.5  ]),
 <a list of 10 Patch objects>)
```



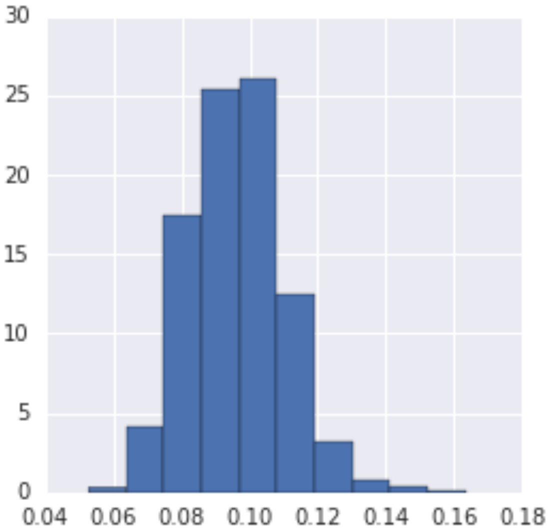
In [143]:
```
plt.figure(figsize=(4, 4))
hist(breast_cancer.data[:, 3], normed=True)
```

Out[143]:
```
(array([  6.93296457e-04,   1.87115495e-03,   7.52934862e-04,
          3.50375629e-04,   3.80194831e-04,   1.04367209e-04,
          4.47288037e-05,   2.23644018e-05,   7.45480061e-06,
          1.49096012e-05]),
 array([  143.5 ,   379.25,   615.  ,   850.75,  1086.5 ,  1322.25,
         1558.  ,  1793.75,  2029.5 ,  2265.25,  2501.  ]),
 <a list of 10 Patch objects>)
```



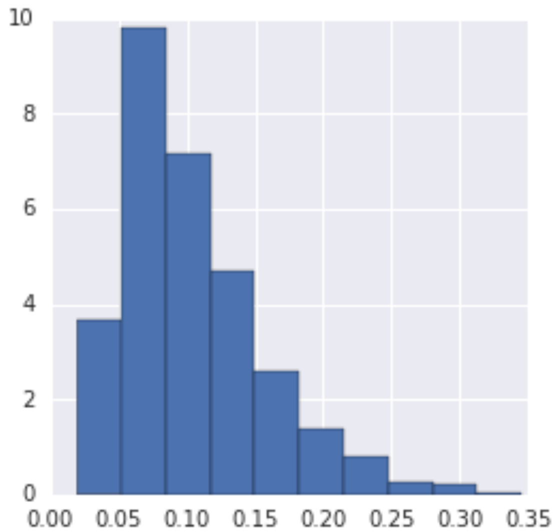
In [144]:
```
plt.figure(figsize=(4, 4))
hist(breast_cancer.data[:, 4], normed=True)
```

Out[144]:
```
(array([  0.31731863,   4.12514222,  17.45252477,  25.38549057,
         26.02012784,  12.53408597,   3.17318632,   0.79329658,
          0.31731863,   0.15865932]),
 array([ 0.05263 ,  0.063707,  0.074784,  0.085861,  0.096938,  0.10
8015,
         0.119092,  0.130169,  0.141246,  0.152323,  0.1634  ]),
 <a list of 10 Patch objects>)
```



In [145]:
```
plt.figure(figsize=(4, 4))
hist(breast_cancer.data[:, 5], normed=True)
```

Out[145]:
```
(array([ 3.66566188,  9.81103621,  7.16960338,  4.68989093,  2.58752
603,
         1.4015766 ,  0.80860189,  0.26953396,  0.21562717,  0.05390
679]),
 array([ 0.01938 ,  0.051982,  0.084584,  0.117186,  0.149788,  0.18
239 ,
         0.214992,  0.247594,  0.280196,  0.312798,  0.3454  ]),
 <a list of 10 Patch objects>)
```



Как видим, многие данные имеют распределение, похожее на нормальное

Сравнение оценок

In [146]:
```
from sklearn.naive_bayes import BernoulliNB, MultinomialNB, GaussianNB
estimators = [BernoulliNB(), MultinomialNB(), GaussianNB()]
for estimator in estimators:
    digits_result = ms.cross_val_score(estimator, digits.data, digits.t
    bc_result = ms.cross_val_score(estimator, breast_cancer.data, breas
    print(u'Оценка с помощью %s' % estimator)
    print(u'Для датасета цифр точность равна: %f, точность оценки рака
    print(u'---')
```

```
Оценка с помощью BernoulliNB(alpha=1.0, binarize=0.0, class_prior=No
ne, fit_prior=True)
Для датасета цифр точность равна: 0.825824, точность оценки рака гру
ди: 0.627420
---
Оценка с помощью MultinomialNB(alpha=1.0, class_prior=None, fit_prio
r=True)
Для датасета цифр точность равна: 0.870877, точность оценки рака гру
ди: 0.894579
---
Оценка с помощью GaussianNB(priors=None)
Для датасета цифр точность равна: 0.818600, точность оценки рака гру
ди: 0.936749
---
```

Как видно, оценка с помощью Бернулли очень плохо показала себя на тесте с недискретными данными, а теоретические догадки совпали на практике.

Ответы на вопросы из задания

1) Каким получилось максимальное качество классификации на датасете breast_cancer?

In [147]:
```
0.936749
```

Out[147]: 0.936749

2) Каким получилось максимальное качество классификации на датасете digits?

In [148]:
```
0.870877
```

Out[148]: 0.870877

3) Какие утверждения из приведенных ниже верны?

(a) На вещественных признаках лучше всего сработал наивный байесовский классификатор с распределением Бернулли

(b) На вещественных признаках лучше всего сработал наивный байесовский классификатор с мультиномиальным распределением

(c) Мультиномиальное распределение лучше показало себя на выборке с целыми неотрицательными значениями признаков

(d) На вещественных признаках лучше всего сработало нормальное распределение

(c), (d)

In [ ]:

```

```