CS4487 - Machine Learning

Lecture 2b - Naive Bayes Classifier

Dr. Antoni B. Chan

Dept. of Computer Science, City University of Hong Kong

Outline

1. Naive Bayes Gaussian Classifier - Iris dataset
2. Naive Bayes Spam Classifier - Spam dataset

Naive Bayes Classifier

- How to deal with multiple features?
  - e.g., $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$
- **Naive Bayes assumption**
  - assume each feature dimension is modeled independently.
    - e.g., for 2 dimensions, $p(x_1, x_2|y) = p(x_1|y)p(x_2|y)$
    - accumulates evidence from each feature dimension:
      - $\log p(x_1, x_2|y) = \log p(x_1|y) + \log p(x_2|y)$
  - allows us to model each dimension of the observation with a simple univariate distribution.
- **Example: Gaussian classifier**
  - We will consider the 2-dimensional iris data shown in the beginning of lecture.

Setup Python

In [1]:
```
%matplotlib inline
import IPython.core.display
# setup output image format (Chrome works best)
IPython.core.display.set_matplotlib_formats("svg")
import matplotlib.pyplot as plt
import matplotlib
from numpy import *
from sklearn import *
from scipy import stats
random.seed(100)          # specify a seed so results are reproducible
```

```
/anaconda3/lib/python3.5/importlib/_bootstrap.py:222: RuntimeWarning: numpy.
dtype size changed, may indicate binary incompatibility. Expected 96, got 88
  return f(*args, **kwds)
/anaconda3/lib/python3.5/importlib/_bootstrap.py:222: RuntimeWarning: numpy.
dtype size changed, may indicate binary incompatibility. Expected 96, got 88
  return f(*args, **kwds)
/anaconda3/lib/python3.5/importlib/_bootstrap.py:222: RuntimeWarning: numpy.
dtype size changed, may indicate binary incompatibility. Expected 96, got 88
  return f(*args, **kwds)
/anaconda3/lib/python3.5/importlib/_bootstrap.py:222: RuntimeWarning: numpy.
dtype size changed, may indicate binary incompatibility. Expected 96, got 88
  return f(*args, **kwds)
/anaconda3/lib/python3.5/site-packages/sklearn/cross_validation.py:41: Depre
cationWarning: This module was deprecated in version 0.18 in favor of the mo
del_selection module into which all the refactored classes and functions are
moved. Also note that the interface of the new CV iterators are different fr
om that of this module. This module will be removed in 0.20.
  "This module will be removed in 0.20.", DeprecationWarning)
/anaconda3/lib/python3.5/site-packages/sklearn/ensemble/weight_boosting.py:2
9: DeprecationWarning: numpy.core.umath_tests is an internal NumPy module an
d should not be imported. It will be removed in a future NumPy release.
  from numpy.core.umath_tests import inner1d
/anaconda3/lib/python3.5/site-packages/sklearn/grid_search.py:42: Deprecatio
nWarning: This module was deprecated in version 0.18 in favor of the model_s
election module into which all the refactored classes and functions are move
d. This module will be removed in 0.20.
  DeprecationWarning)
/anaconda3/lib/python3.5/site-packages/sklearn/learning_curve.py:22: Depreca
tionWarning: This module was deprecated in version 0.18 in favor of the mode
l_selection module into which all the functions are moved. This module will
be removed in 0.20
  DeprecationWarning)
```

Load data

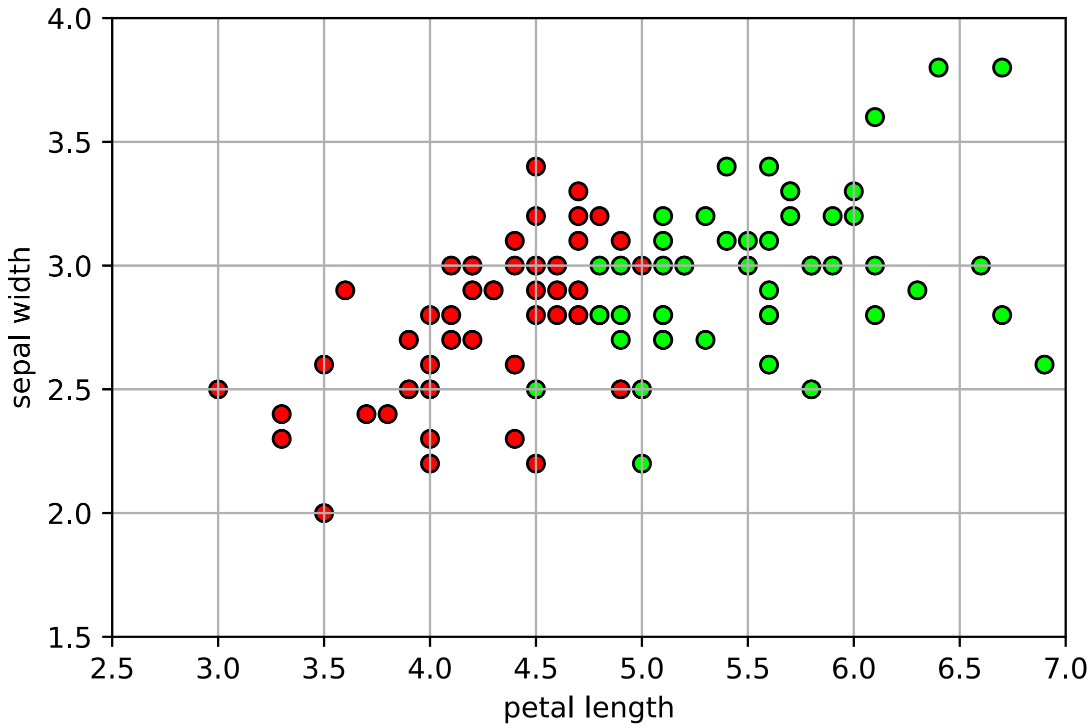
```
In [2]: # load iris data each row is (petal length, sepal width, class)
        irisdata = loadtxt('iris2.csv', delimiter=',', skiprows=1)

        X = irisdata[:,0:2]  # the first two columns are features (petal length, sepal w
        idth)
        Y = irisdata[:,2]    # the third column is the class label (versicolor=1, virgin
        ica=2)

        print(X.shape)
```

```
(100, 2)
```

View data

```
In [3]: # a colormap for making the scatter plot: class 1 will be red, class 2 will be g
        reen
        mycmap = matplotlib.colors.LinearSegmentedColormap.from_list('mycmap', ["#FF0000
        ", "#FFFFFF", "#00FF00"])

        axbox = [2.5, 7, 1.5, 4] # common axis range

        # a function for setting a common plot
        def irisaxis():
            plt.xlabel('petal length'); plt.ylabel('sepal width')
            plt.axis([2.5, 7, 1.5, 4]); plt.grid(True)
```

```
In [4]: # show the data
        plt.figure()
        plt.scatter(X[:,0], X[:,1], c=Y, cmap=mycmap, edgecolors='k')
        # c is the color value, drawn from colormap mycmap
        irisaxis()
```



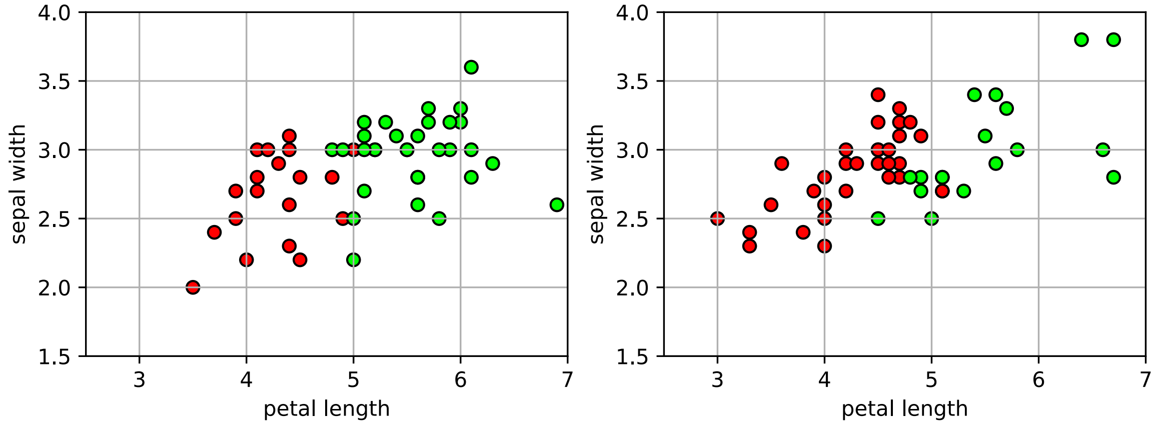
Split training/test data

- We will select 50% of the data for training, and 50% for testing
  - use `model_selection` module
    - `train_test_split` - give the percentage for training and testing.
    - `StratifiedShuffleSplit` - also preserves the percentage of examples for each class.

In [5]:
```
# randomly split data into 50% train and 50% test set
trainX, testX, trainY, testY = \
  model_selection.train_test_split(X, Y,
    train_size=0.5, test_size=0.5, random_state=4487)

print(trainX.shape)
print(testX.shape)
```

```
(50, 2)
(50, 2)
```

In [6]:
```
# view train & test data
plt.figure(figsize=(9,3))

plt.subplot(1,2,1)    # put two subplots in the same figure
# scatter plot - Y value selects the color
plt.scatter(trainX[:,0], trainX[:,1], c=trainY, cmap=mycmap, edgecolors='k')
irisaxis()

plt.subplot(1,2,2)
plt.scatter(testX[:,0], testX[:,1], c=testY, cmap=mycmap, edgecolors='k')
irisaxis()
```



Learn Gaussian NB model

- treat each dimension as an independent Gaussian

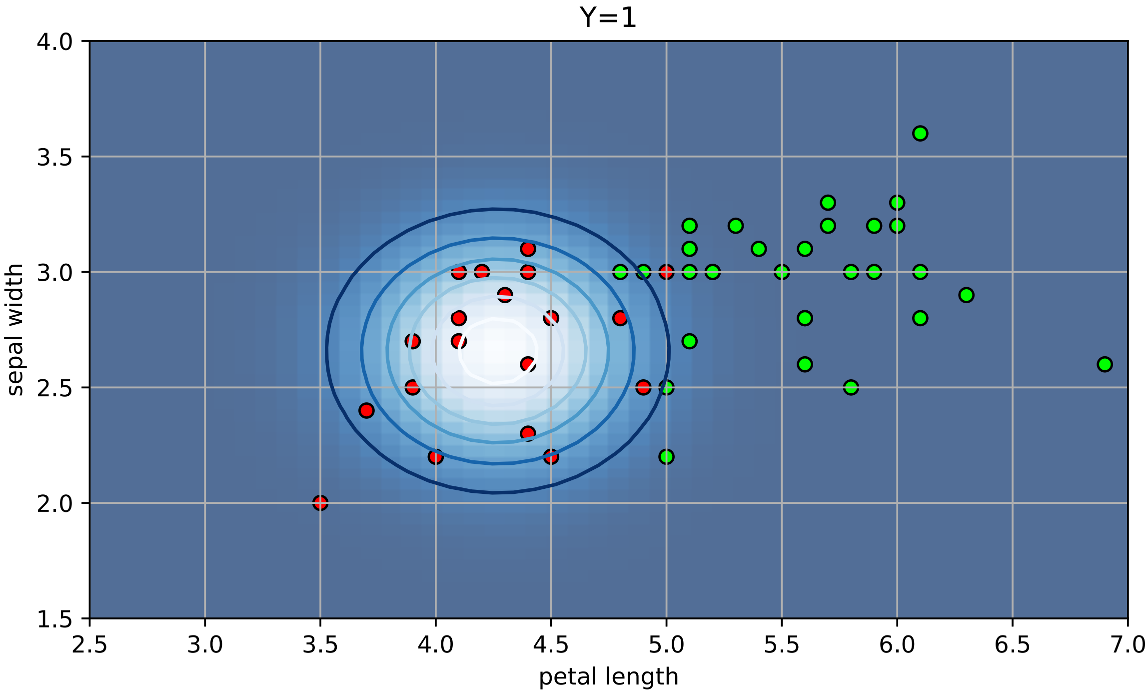
In [7]:
```
# get the NB Gaussian model from sklearn
model = naive_bayes.GaussianNB()

# fit the model to training data
model.fit(trainX, trainY)

# see the parameters
print("class prior: ",  model.class_prior_)
print("class 1 mean: ", model.theta_[0,:])
print("class 1 var:  ", model.sigma_[0,:])
print("class 2 mean: ", model.theta_[1,:])
print("class 2 var:  ", model.sigma_[1,:])
```

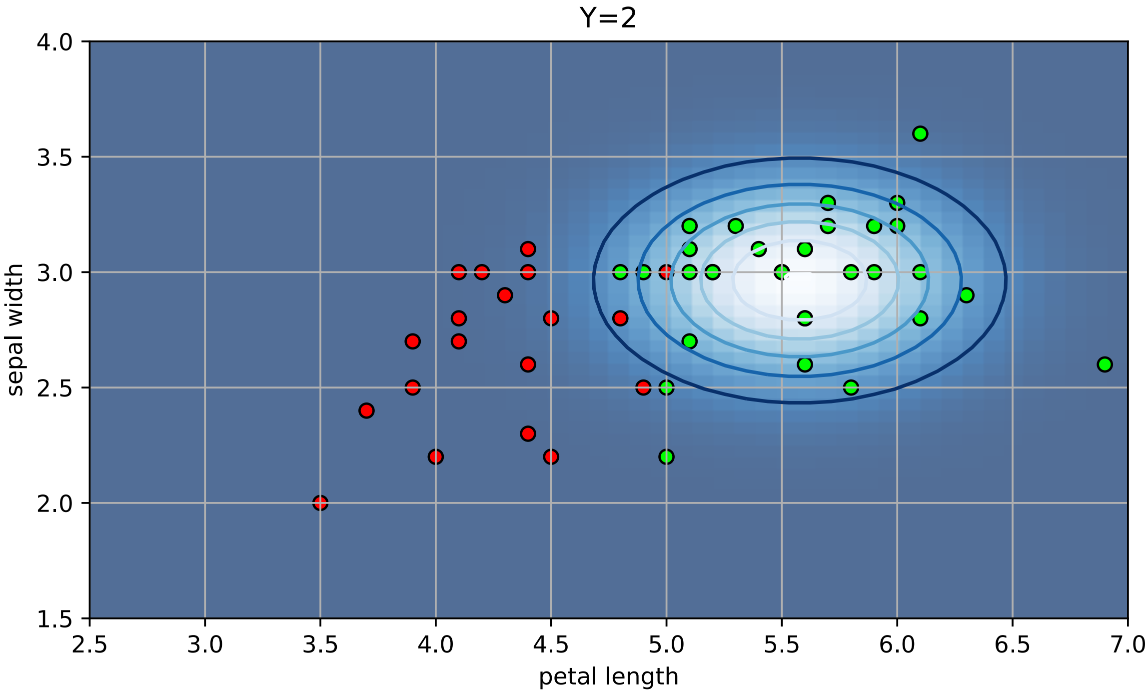
```
class prior:  [0.38 0.62]
class 1 mean:  [4.26842105 2.65789474]
class 1 var:   [0.14426593 0.09927978]
class 2 mean:  [5.57741935 2.96451613]
class 2 var:   [0.22045786 0.07777315]
```

- View 2d class conditionals:
  - $p(x_1, x_2|y = c) = p(x_1|y = c)p(x_2|y = c)$

In [9]:
```
ccd
```

Out[9]:

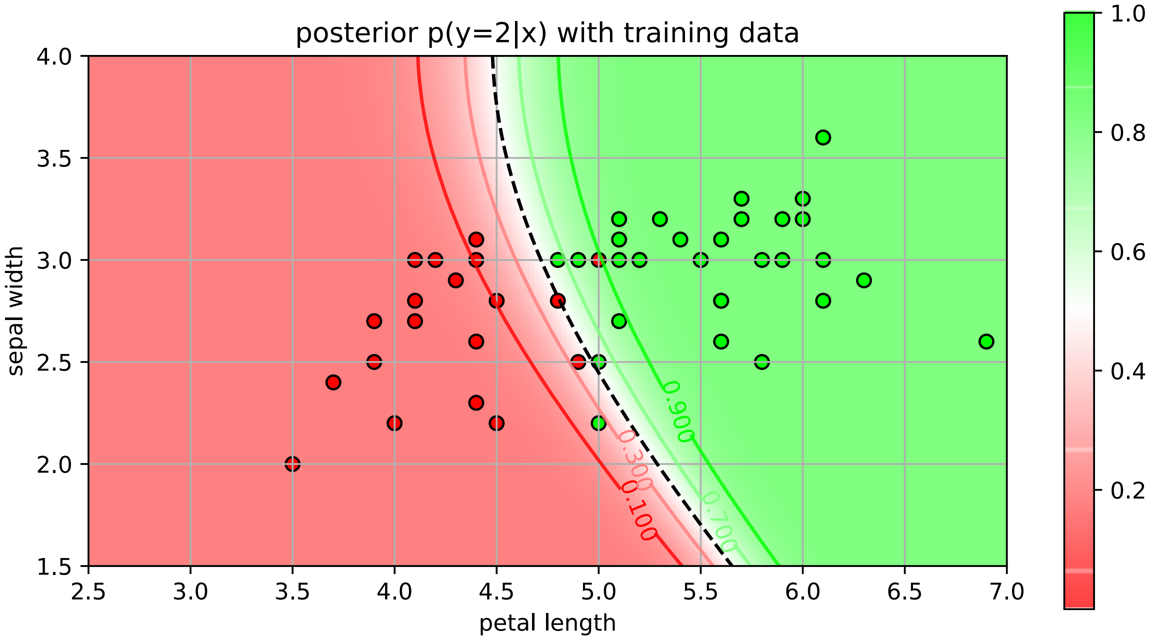




View the Posterior

In [12]:
```
pfig
```

Out[12]:



Evaluate on the test set

In [13]:
```
# predict from the model
predY = model.predict(testX)
print("pred: ", predY)
print("true: ", testY)

# calculate accuracy
acc = metrics.accuracy_score(testY, predY)
print("accuracy=", acc)
```

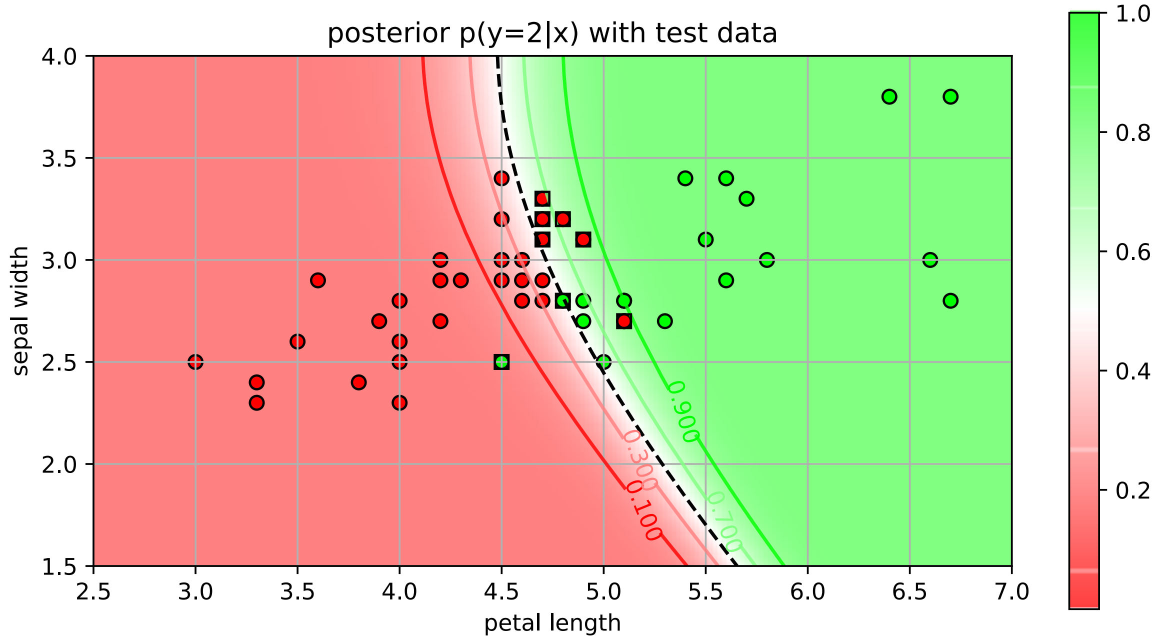
```
pred:  [2. 2. 1. 2. 1. 2. 1. 1. 1. 2. 2. 2. 2. 2. 2. 1. 2. 2. 1. 2. 1. 1. 1.
1.
 1. 1. 1. 1. 2. 1. 2. 1. 2. 2. 1. 1. 1. 1. 1. 1. 2. 1. 2. 2. 2. 1. 2. 2.
 1. 1.]
true:  [2. 2. 1. 2. 1. 2. 1. 1. 1. 1. 1. 2. 2. 2. 2. 1. 2. 2. 1. 1. 1. 1. 2.
1.
 1. 1. 1. 1. 2. 1. 2. 1. 2. 2. 1. 1. 2. 1. 1. 1. 1. 1. 1. 2. 2. 1. 1. 2.
 1. 1.]
accuracy= 0.84
```

Viewing test results

In [15]:

```
tfig
```

Out[15]:



Example: Naive Bayes Spam Classifier

- Goal: given an input email, predict whether it is spam or not
  - input: text string

> A home based business opportunity is knocking at your door. Donít be rude and let this chance go by. You can earn a great income and find your financial life transformed. Learn more Here. To Your Success. Work From Home Finder Experts

  - output: spam, not spam (or ham)

Text Document Representation

- Text document is a string!
  - we need to pick a suitable representation.
- **Bag-of-Words (BoW) model**
  - Let $\mathcal{V} = \{w_1, w_2, \cdots w_V\}$ be a list of V words (called a **vocabulary**).
  - represent a text document as a vector $\mathbf{x} \in \mathbb{R}^V$.
    - each entry x_j represents the number of times word w_j appears in the document.
- **Example:**
  - Document: "This is a test document"
  - Vocabulary: $\mathcal{V} = \{\text{"this"}, \text{"test"}, \text{"spam"}, \text{"foo"}\}$
  - Vector representation: $\mathbf{x} = [1, 1, 0, 0]$

- **NOTE:**
  - the order of the words is not used!
  - rearranging words leads to the same representation!
- Example:
  - "this is spam" $\rightarrow \mathbf{x} = [1, 0, 1, 0]$
  - "is this spam" $\rightarrow \mathbf{x} = [1, 0, 1, 0]$
- This is why it is called "bag-of-words"

Steps to make BoW

1. Build a vocabulary $\mathcal{V}$
   - remove *stopwords*
     - the most common words that provide little information
     - examples: "the", "a", "on"
   - convert to all lower case
2. Calculate the vector for each document
   - count the occurence of each word in the vocabulary

In [16]:
```
# Load text data from directories
#   each sub-directory contains text files for 1 class
textdata = datasets.load_files("email", encoding="utf8", decode_error="replace")

# target names
print("class names = ", textdata.target_names)
print("classes = ", unique(textdata.target))
print("num samples = ", len(textdata.target))
```

```
class names =  ['ham', 'spam']
classes =  [0 1]
num samples =  50
```

In [17]:
```
# look at first sample
print("Sample 1 is Class " + str(textdata.target[0]) + \
  "(" + textdata.target_names[textdata.target[0]] + ")")
print("---")
print(textdata.data[0])
```

```
Sample 1 is Class 1(spam)
---
Get Up to 75% OFF at Online WatchesStore

Discount Watches for All Famous Brands

* Watches: aRolexBvlgari, Dior, Hermes, Oris, Cartier, AP and more brands
* Louis Vuitton Bags & Wallets
* Gucci Bags
* Tiffany & Co Jewerly

Enjoy a full 1 year WARRANTY
Shipment via reputable courier: FEDEX, UPS, DHL and EMS Speedpost
You will 100% recieve your order
```

In [18]:
```
# randomly split data into 50% train and 50% test set
traintext, testtext, trainY, testY = \
  model_selection.train_test_split(textdata.data, textdata.target,
  train_size=0.5, test_size=0.5, random_state=11)

print(len(traintext))
print(len(testtext))
```

```
25
25
```

In [19]:

```python
# setup the document vectorizer to make BoW
# - use english stop words
# - only use the most frequent 100 words in the dataset
cntvect = feature_extraction.text.CountVectorizer(stop_words='english', max_feat
ures=100)

# create the vocabulary, and return the document vectors
# NOTE: we only use the training data!
trainX = cntvect.fit_transform(traintext)

# calculate vectors for the test data
testX  = cntvect.transform(testtext)

# print the vocabulary
# - (key,value) pairs correspond to (word,vector index)
print(cntvect.vocabulary_)
```

```
{'day': 31, 'close': 21, 'brand': 18, 'thanks': 90, 'chief': 19, 'phone': 71
, 'microsoft': 61, 'york': 99, 'wilson': 96, 'new': 66, 'frank': 41, 'accoun
t': 11, 'john': 52, 'windows': 97, 'states': 88, 'republic': 79, 'financial'
: 38, 'status': 89, 'commented': 24, 'days': 32, 'today': 91, 'man': 58, 'na
mes': 64, 'just': 53, 'send': 85, 'let': 55, 'nations': 65, 'pricing': 73, '
number': 68, '30mg': 8, '000': 1, 'wilmott': 95, '30': 7, 'provide': 74, 'en
abled': 35, 'release': 77, 'email': 34, 'mandelbrot': 59, 'nigeria': 67, '70
': 10, 'compensation': 25, 'required': 80, 'contact': 26, 'details': 33, 'fe
deral': 37, 'following': 39, 'funds': 44, 'representative': 78, 'inheritance
': 50, 'benoit': 17, 'forward': 40, 'info': 47, 'mr': 63, '60': 9, 'receive'
: 75, 'united': 92, 'inform': 48, 'current': 29, 'office': 69, 'soon': 86, '
extended': 36, '2010': 6, 'freeviagra': 42, 'source': 87, 'model': 62, '20':
5, 'scifinance': 84, 'address': 12, 'wheeler': 94, 'codeine': 23, 'choice':
20, '10': 2, 'gpu': 46, 'information': 49, 'ryan': 83, 'right': 81, 'going':
45, 'code': 22, '00': 0, 'working': 98, 'mg': 60, 'risk': 82, 'country': 28,
'fund': 43, 'andrew': 15, 'payment': 70, 'adobe': 13, 'bank': 16, 'cost': 27
, 'major': 56, 'regards': 76, 'interesting': 51, 'make': 57, 'watson': 93, '
15mg': 4, 'pills': 72, 'david': 30, '120': 3, 'agaliofu': 14, 'know': 54}
```

In [20]:

```python
# define a function for prettier output...
def showVocab(vocab, counts=None):
    "print out the vocabulary. if counts specified, then only print the words w/
non-zero entries"
    allwords = list(vocab.keys())
    allwords.sort()  # sort vocabulary by index
    wordlist = []
    for word in allwords:
        ind = vocab[word]
        if (counts is None):
            mystr = "{:3d}. {}".format(ind, word)
        elif (counts[0,ind]>0):
            mystr = "{:3d}. ({:0.4f}) {}".format(ind, counts[0,ind], word)
        else:
            continue  # skip it
        wordlist.append(mystr)

    # print 2 columns
    it = iter(wordlist)
    for i in it:
```

```
        print('{:<30}{}'.format(i, next(it)))

# show the vocabulary
showVocab(cntvect.vocabulary_)
```

```
  0. 00                             1. 000
  2. 10                             3. 120
  4. 15mg                           5. 20
  6. 2010                           7. 30
  8. 30mg                           9. 60
 10. 70                            11. account
 12. address                       13. adobe
 14. agaliofu                      15. andrew
 16. bank                          17. benoit
 18. brand                         19. chief
 20. choice                        21. close
 22. code                          23. codeine
 24. commented                     25. compensation
 26. contact                       27. cost
 28. country                       29. current
 30. david                         31. day
 32. days                          33. details
 34. email                         35. enabled
 36. extended                      37. federal
 38. financial                     39. following
 40. forward                       41. frank
 42. freeviagra                    43. fund
 44. funds                         45. going
 46. gpu                           47. info
 48. inform                        49. information
 50. inheritance                   51. interesting
 52. john                          53. just
 54. know                          55. let
 56. major                         57. make
 58. man                           59. mandelbrot
 60. mg                            61. microsoft
 62. model                         63. mr
 64. names                         65. nations
 66. new                           67. nigeria
 68. number                        69. office
 70. payment                       71. phone
 72. pills                         73. pricing
 74. provide                       75. receive
 76. regards                       77. release
 78. representative                79. republic
 80. required                      81. right
 82. risk                          83. ryan
 84. scifinance                    85. send
 86. soon                          87. source
 88. states                        89. status
 90. thanks                        91. today
 92. united                        93. watson
 94. wheeler                       95. wilmott
 96. wilson                        97. windows
 98. working                       99. york
```

In [21]:

```python
# show a document vector
# - only the non-zero entries are printed
print(trainX[0])
```

```
  (0, 40)	1
  (0, 44)	1
  (0, 80)	1
  (0, 49)	1
  (0, 12)	1
  (0, 68)	1
  (0, 71)	1
  (0, 26)	2
  (0, 64)	1
  (0, 47)	1
  (0, 74)	1
  (0, 50)	1
  (0, 16)	1
  (0, 70)	5
  (0, 88)	1
  (0, 78)	1
  (0, 65)	1
  (0, 92)	2
  (0, 41)	2
  (0, 52)	2
  (0, 63)	2
  (0, 31)	2
```

In [22]:
```
# show the actual words
showVocab(cntvect.vocabulary_, trainX[0])
print("---")
print(traintext[0])
```

```
 12. (1.0000) address          16. (1.0000) bank
 26. (2.0000) contact          31. (2.0000) day
 40. (1.0000) forward          41. (2.0000) frank
 44. (1.0000) funds            47. (1.0000) info
 49. (1.0000) information      50. (1.0000) inheritance
 52. (2.0000) john             63. (2.0000) mr
 64. (1.0000) names            65. (1.0000) nations
 68. (1.0000) number           70. (5.0000) payment
 71. (1.0000) phone            74. (1.0000) provide
 78. (1.0000) representative   80. (1.0000) required
 88. (1.0000) states           92. (2.0000) united
---
Attn:Good Day,

Compliment of the day to you, my name is Mr John Frank Harmon a UNITED
NATIONS Representative here in UNITED STATES this year 2014 last
payment quarter for all outstanding payment from World Bank on overdue
contracts,inheritance and all other payment has commenced,you are to
provide the below info asap so that the payment processing can start
off.

Your full names
Contact phone number
Contact Address

The above information is required so as to go through your payment
file and start the processing of this long and overdue funds.

looking forward to hearing from you

Mr John Frank Harmon
```

Naive Bayes model for Boolean vectors

- Model each word independently
  - absence/presence of a word w_j in document
  - Bernoulli distribution
    - present: $p(x_j = 1|y) = \pi_j$
    - absent: $p(x_j = 0|y) = 1 - \pi_j$
  - MLE parameters: $\pi_j = N_j/N$,
    - N_j is the number of documents in class y that contain word j.
    - N is the number of documents in class y.

- Class-conditional distribution

$$p(x_1, \cdots, x_V | y = \text{spam}) = \prod_{j=1}^{V} p(x_j | y = \text{spam})$$

$$\log p(x_1, \cdots, x_V | y = \text{spam}) = \sum_{j=1}^{V} \log p(x_j | y = \text{spam})$$

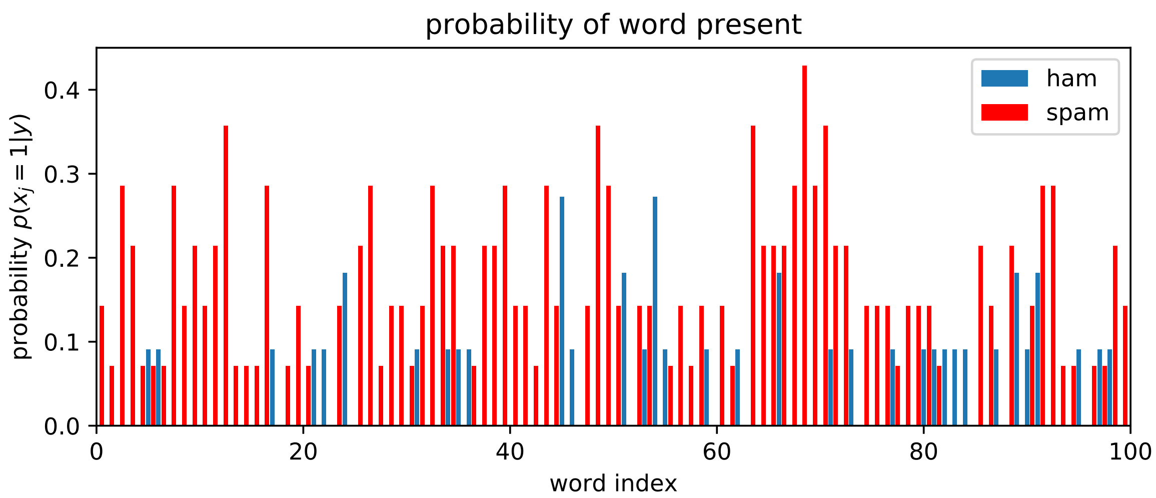
- for a document, the log-probabilities of the words being in a spam message adds.
  - accumulate evidence over all words in the document.
  - more words that are associated with spam --> more likely the document is spam

In [23]:
```
# fit the NB Bernoulli model.
# the model automatically converts count vector into binary vector
bmodel = naive_bayes.BernoulliNB(alpha=0.0)
bmodel.fit(trainX, trainY)
```

```
/anaconda3/lib/python3.5/site-packages/sklearn/naive_bayes.py:472: UserWarni
ng: alpha too small will result in numeric errors, setting alpha = 1.0e-10
  'setting alpha = %.1e' % _ALPHA_MIN)
```

Out[23]:
```
BernoulliNB(alpha=0.0, binarize=0.0, class_prior=None, fit_prior=True)
```

In [24]:
```
# a function for plotting the word probabilities
def plotWordProb(model):
    xr = arange(0,len(cntvect.vocabulary_))
    plt.figure(figsize=(8,3))
    plt.bar(xr, exp(model.feature_log_prob_[0,:]), width=0.4, label=textdata.tar
get_names[0])
    plt.bar(xr+0.5, exp(model.feature_log_prob_[1,:]), width=0.4, color='r', lab
el=textdata.target_names[1])
    plt.legend()
    plt.xlabel('word index'); plt.ylabel('probability $p(x_j=1|y)$');
    plt.xlim(min(xr), max(xr)+1)

# make plot
plotWordProb(bmodel)
plt.title('probability of word present');
```



In [25]:
```
# prediction
predY = bmodel.predict(testX)
print("predictions: ", predY)
print("actual:      ", testY)

# calculate accuracy
acc = metrics.accuracy_score(testY, predY)
print(acc)
```

```
predictions:  [0 1 1 1 1 1 1 1 0 1 0 0 0 0 0 0 0 1 1 0 0 0 1 1 1]
actual:       [0 1 1 1 0 1 1 0 0 1 0 0 1 0 0 0 0 0 1 1 0 1 0 0 1]
0.68
```

In [26]:
```
# show examples of misclassified
inds = where(predY != testY)
print(inds)
for i in inds[0]:
    print("---- true={}, pred={}".format(testY[i], predY[i]))
    print(testtext[i])
```

```
(array([ 4,  7, 12, 17, 19, 21, 22, 23]),)
---- true=0, pred=1
LinkedIn

Julius O requested to add you as a connection on LinkedIn:

Hi Peter.

Looking forward to the book!


Accept  View invitation from Julius O

---- true=0, pred=1
Hi Peter,

The hotels are the ones that rent out the tent. They are all lined up on the
hotel grounds : )) So much for being one with nature, more like being one wi
th a couple dozen tour groups and nature.
I have about 100M of pictures from that trip. I can go through them and get
you jpgs of my favorite scenic pictures.

Where are you and Jocelyn now? New York? Will you come to Tokyo for Chinese
New Year? Perhaps to see the two of you then. I will go to Thailand for wint
er holiday to see my mom : )

Take care,
D

---- true=1, pred=0
Get Up to 75% OFF at Online WatchesStore

Discount Watches for All Famous Brands

* Watches: aRolexBvlgari, Dior, Hermes, Oris, Cartier, AP and more brands
* Louis Vuitton Bags & Wallets
```

* Gucci Bags
* Tiffany & Co Jewerly

Enjoy a full 1 year WARRANTY
Shipment via reputable courier: FEDEX, UPS, DHL and EMS Speedpost
You will 100% recieve your order
Save Up to 75% OFF Quality Watches
---- true=0, pred=1
This e-mail was sent from a notification-only address that cannot accept inc
oming e-mail. Please do not reply to this message.

Thank you for your online reservation. The store you selected has located th
e item you requested and has placed it on hold in your name. Please note tha
t all items are held for 1 day. Please note store prices may differ from th
ose online.

If you have questions or need assistance with your reservation, please conta
ct the store at the phone number listed below. You can also access store inf
ormation, such as store hours and location, on the web at http://www.borders
.com/online/store/StoreDetailView_98.
---- true=1, pred=0
Get Up to 75% OFF at Online WatchesStore

Discount Watches for All Famous Brands

* Watches: aRolexBvlgari, Dior, Hermes, Oris, Cartier, AP and more brands
* Louis Vuitton Bags & Wallets
* Gucci Bags
* Tiffany & Co Jewerly

Enjoy a full 1 year WARRANTY
Shipment via reputable courier: FEDEX, UPS, DHL and EMS Speedpost
You will 100% recieve your order
---- true=1, pred=0
Get Up to 75% OFF at Online WatchesStore

Discount Watches for All Famous Brands

* Watches: aRolexBvlgari, Dior, Hermes, Oris, Cartier, AP and more brands
* Louis Vuitton Bags & Wallets
* Gucci Bags
* Tiffany & Co Jewerly

Enjoy a full 1 year WARRANTY
Shipment via reputable courier: FEDEX, UPS, DHL and EMS Speedpost
You will 100% recieve your order
---- true=0, pred=1
Ok I will be there by 10:00 at the latest.
---- true=0, pred=1
Hello,

Since you are an owner of at least one Google Groups group that uses the cus
tomized welcome message, pages or files, we are writing to inform you that w
e will no longer be supporting these features starting February 2011. We mad
e this decision so that we can focus on improving the core functionalities o
f Google Groups -- mailing lists and forum discussions. Instead of these fe
atures, we encourage you to use products that are designed specifically for

```
file storage and page creation, such as Google Docs and Google Sites.

For example, you can easily create your pages on Google Sites and share the
site (http://www.google.com/support/sites/bin/answer.py?hl=en&answer=174623)
with the members of your group. You can also store your files on the site by
attaching files to pages (http://www.google.com/support/sites/bin/answer.py?
hl=en&answer=90563) on the site. If you�re just looking for a place to uplo
ad your files so that your group members can download them, we suggest you t
ry Google Docs. You can upload files (http://docs.google.com/support/bin/ans
wer.py?hl=en&answer=50092) and share access with either a group (http://docs
.google.com/support/bin/answer.py?hl=en&answer=66343) or an individual (http
://docs.google.com/support/bin/answer.py?hl=en&answer=86152), assigning eith
er edit or download only access to the files.

you have received this mandatory email service announcement to update you ab
out important changes to Google Groups.
```

Smoothing

- Some words are not present in any documents for a given class.
  - $N_j = 0$, and thus $\pi_j = 0$.
    - i.e., the document in the class **definitely** will not contain the word.
    - can be a problem since we simply may not have seem an example with that word.
- Smoothed MLE
  - add a smoothing parameter α that adds a "virtual" count
  - parameter: $\pi_j = (N_j + \alpha)/(N + 2\alpha)$,
  - this is called *Laplace smoothing*
- In general, *regularizing* or *smoothing* of the estimate helps to prevent *overfitting* of the parameters.

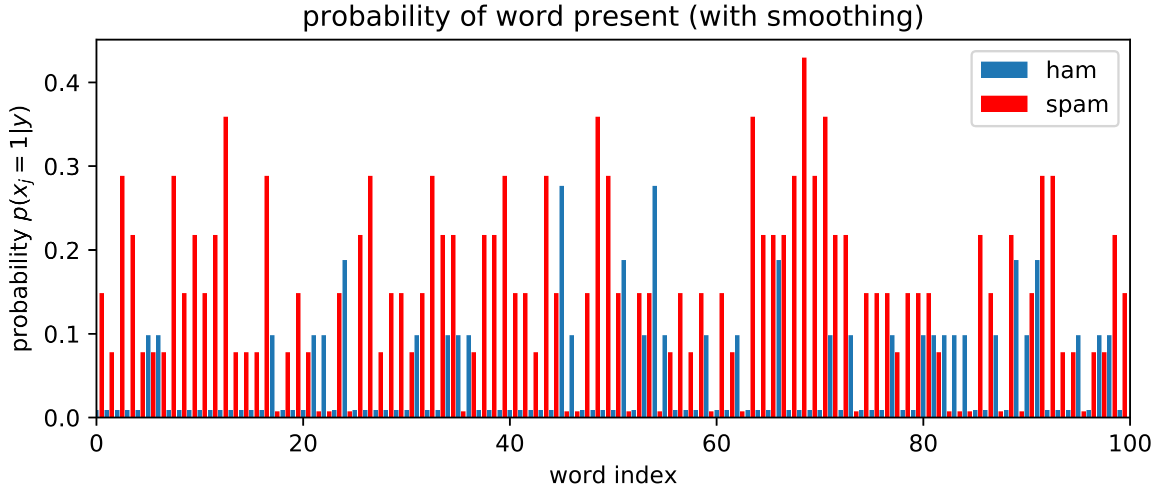
In [27]:

```
# fit the NB Bernoulli model w/ smoothing (0.1)
bmodels = naive_bayes.BernoulliNB(alpha=0.1)
bmodels.fit(trainX, trainY)
```

Out[27]:

```
BernoulliNB(alpha=0.1, binarize=0.0, class_prior=None, fit_prior=True)
```

In [28]:
```
# make plot
plotWordProb(bmodels)
plt.title('probability of word present (with smoothing)');
# note the small probabilities are all slightly above 0.
```



In [29]:
```
# prediction
predY = bmodels.predict(testX)
print("predictions: ", predY)
print("actual:      ", testY)

# calculate accuracy
acc = metrics.accuracy_score(testY, predY)
print(acc)
# a little better!
```

```
predictions:  [0 1 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 1]
actual:       [0 1 1 1 0 1 1 0 0 1 0 0 1 0 0 0 0 0 1 1 0 1 0 0 1]
0.72
```

Most informative words

- The most informative words are those with high probability of being in one class, and low probability of being in other classes.
  - e.g., For class 1, find large values of $\log p(w_j|y=1) - \log p(w_j|y=0)$

In [30]:
```
# get the word names
fnames = asarray(cntvect.get_feature_names())
# coef_ contains the scores for each word
# (higher means more informative)
# sort the coefficients in ascending order, and take the 10 largest.
tmp = argsort(bmodel.coef_[0])[-10:]

for i in tmp:
    print("{:3d}. {:15s} ({:.5f})".format(i, fnames[i], bmodel.coef_[0][i]))
```

```
 16. bank            (-1.25276)
 67. nigeria         (-1.25276)
 26. contact         (-1.25276)
 69. office          (-1.25276)
 32. days            (-1.25276)
 48. inform          (-1.02962)
 70. payment         (-1.02962)
 63. mr              (-1.02962)
 12. address         (-1.02962)
 68. number          (-0.84730)
```

Naive Bayes for Count Vectors

- Now we consider using the number of times each word appears in the document D.
- Two ways to create a document vector x based on the word counts.
- **Term-Frequency (TF)**
  - handles documents with different lengths (number of words).
  - normalize the count to a frequency, by dividing by the number of words in the document.
    - $x_j = \frac{w_j}{|D|}$
      - w_j is the number of times word j appears in the document
      - $|D|$ is the number of words in the document.

- **Term-Frequency Inverse Document Frequency (TF-IDF)**
  - some words are common among many documents
    - common words are less informative because they appear in both classes.
  - **inverse document frequency (IDF)** - measure rarity of each word
    - $IDF(j) = \log \frac{N}{N_j}$
      - N is the number of documents.
      - N_j is the number of documents with word j.
    - IDF is:
      - 0 when a word is common to all documents
      - large value when the word appears in few documents
  - **TF-IDF vector:** downscale words that are common in many documents
    - multiply TF and IDF terms
    - $x_j = \frac{w_j}{|D|} \log \frac{N}{N_j}$

In [31]:

```
# TF-IDF representation
# (For TF, pass use_idf=False)
tf_trans = feature_extraction.text.TfidfTransformer(use_idf=True, norm='l1')
# 'l1' - entries sum to 1

# setup the TF-IDF representation, and transform the training set
trainXtf = tf_trans.fit_transform(trainX)

# transform the test set
testXtf = tf_trans.transform(testX)

print(trainXtf[0])
```

```
  (0, 31)	0.06423955966673985
  (0, 63)	0.055169680057832474
  (0, 52)	0.07067474166045794
  (0, 41)	0.07067474166045794
  (0, 92)	0.059248044723895436
  (0, 65)	0.03211977983336992
  (0, 78)	0.03533737083022897
  (0, 88)	0.03211977983336992
  (0, 70)	0.13792420014458118
  (0, 16)	0.029624022361947718
  (0, 50)	0.03533737083022897
  (0, 74)	0.03533737083022897
  (0, 47)	0.03533737083022897
  (0, 64)	0.03211977983336992
  (0, 26)	0.059248044723895436
  (0, 71)	0.029624022361947718
  (0, 68)	0.025860735932526902
  (0, 12)	0.027584840028916237
  (0, 49)	0.029624022361947718
  (0, 80)	0.03211977983336992
  (0, 44)	0.03533737083022897
  (0, 40)	0.03533737083022897
```

In [32]:

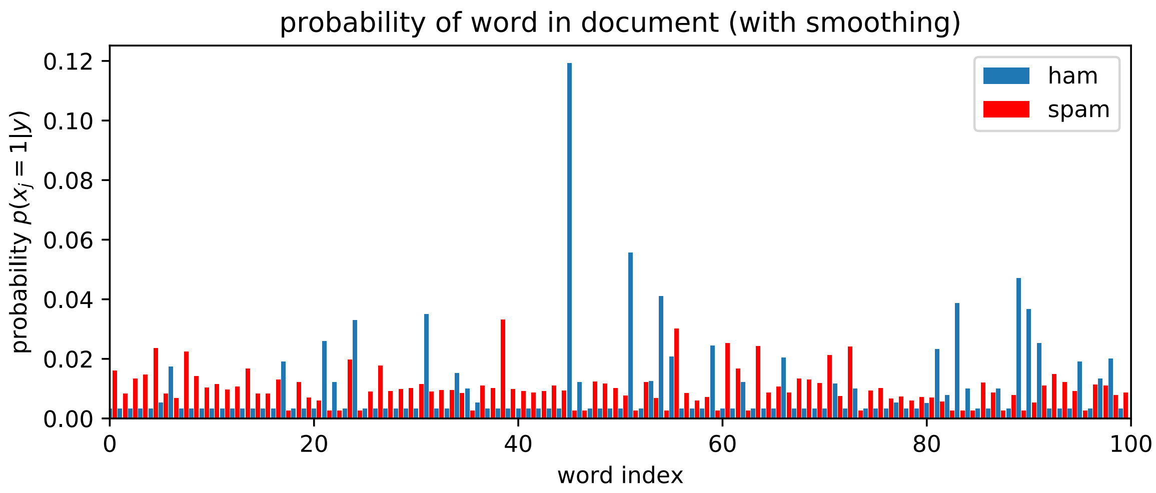
```
showVocab(cntvect.vocabulary_, trainXtf[0])
```

```
 12. (0.0276) address           16. (0.0296) bank
 26. (0.0592) contact           31. (0.0642) day
 40. (0.0353) forward           41. (0.0707) frank
 44. (0.0353) funds             47. (0.0353) info
 49. (0.0296) information       50. (0.0353) inheritance
 52. (0.0707) john              63. (0.0552) mr
 64. (0.0321) names             65. (0.0321) nations
 68. (0.0259) number            70. (0.1379) payment
 71. (0.0296) phone             74. (0.0353) provide
 78. (0.0353) representative    80. (0.0321) required
 88. (0.0321) states            92. (0.0592) united
```

Naive Bayes Multinomial

- TF or TF-IDF representation
  - Document word vector $\mathbf{x}$
    - x_j is the frequency of word j occuring in the document.
    - vector $\mathbf{x}$ sums to 1, i.e. $\sum_j x_j = 1$.
- Use a multinomial distribution as the class conditional
  - based on the frequency that a word appears in a document of a class.

In [33]:
```
# fit a multinomial model (with smoothing)
mmodel_tf = naive_bayes.MultinomialNB(alpha=0.05)
mmodel_tf.fit(trainXtf, trainY)

# show the word probabilites
plotWordProb(mmodel_tf)
plt.title('probability of word in document (with smoothing)');
```



In [34]:
```
# prediction
predYtf = mmodel_tf.predict(testXtf)
print("prediction: ", predYtf)
print("actual:     ", testY)

# calculate accuracy
acc = metrics.accuracy_score(testY, predYtf)
print(acc)
```

```
prediction:  [0 1 1 1 1 1 1 1 0 1 0 0 1 1 1 0 0 1 1 1 1 1 1 1 1]
actual:      [0 1 1 1 0 1 1 0 0 1 0 0 1 0 0 0 0 0 1 1 0 1 0 0 1]
0.68
```

In [35]:
```python
# most informative words for TF-IDF
fnames = asarray(cntvect.get_feature_names())
tmp = argsort(mmodel_tf.coef_[0])[-10:]
for i in tmp:
    print("{:3d}. {:15s} ({:.5f})".format(i, fnames[i], mmodel_tf.coef_[0][i]))
```

```
 26. contact         (-4.03474)
 23. codeine         (-3.92301)
 70. payment         (-3.85287)
  7. 30              (-3.80167)
  4. 15mg            (-3.74496)
 72. pills           (-3.72604)
 63. mr              (-3.71758)
 60. mg              (-3.68161)
 55. let             (-3.49983)
 38. financial       (-3.40804)
```

Summary

- **Generative classification model**
  - estimate probability distributions of features generated from each class.
  - given feature observation predict class with largest posterior probability.
- **Advantages:**
  - works with small amount of data.
  - works with multiple classes.
- **Disadvantages:**
  - accuracy depends on selecting an appropriate probability distribution.
    - if the probability distribution doesn't model the data well, then accuracy might be bad.

- **Other text preprocessing**
  - *Stemming*
    - convert related words into a common root word
    - example: testing, tests --> "test"
    - see NLTK toolbox (http://www.nltk.org (http://www.nltk.org))
  - *Lemmatisation*
    - similar to stemming
    - groups inflections of word together (gone, going, went -> go)
    - see NLTK
  - Removing numbers and punctuation.

- **Other word models**
  - *N-grams*
    - similar to BoW except look at pairs of consecutive words (or N consecutive words in general)
  - *word vectors*
    - each word is a real vector, where direction indicates the "concept"
    - words about similar things point in the same direction
    - adding and subtracting word vectors yield new word vectors