Naïve Bayes Classifier

葉建華

jhyeh@mail.au.edu.tw

http://jhyeh.csie.au.edu.tw/

Hard Decisions vs. Soft Decisions

- Answer: "Which class does this data instance belong to?"
  - That's a hard answer
- How about "best guess" by probabilistic estimation?

Decision by Naïve Bayes

- Pros: Works with a small amount of data, handles multiple classes
- Cons: Sensitive to how the input data is prepared
- Works with: Nominal values

Probability Distribution

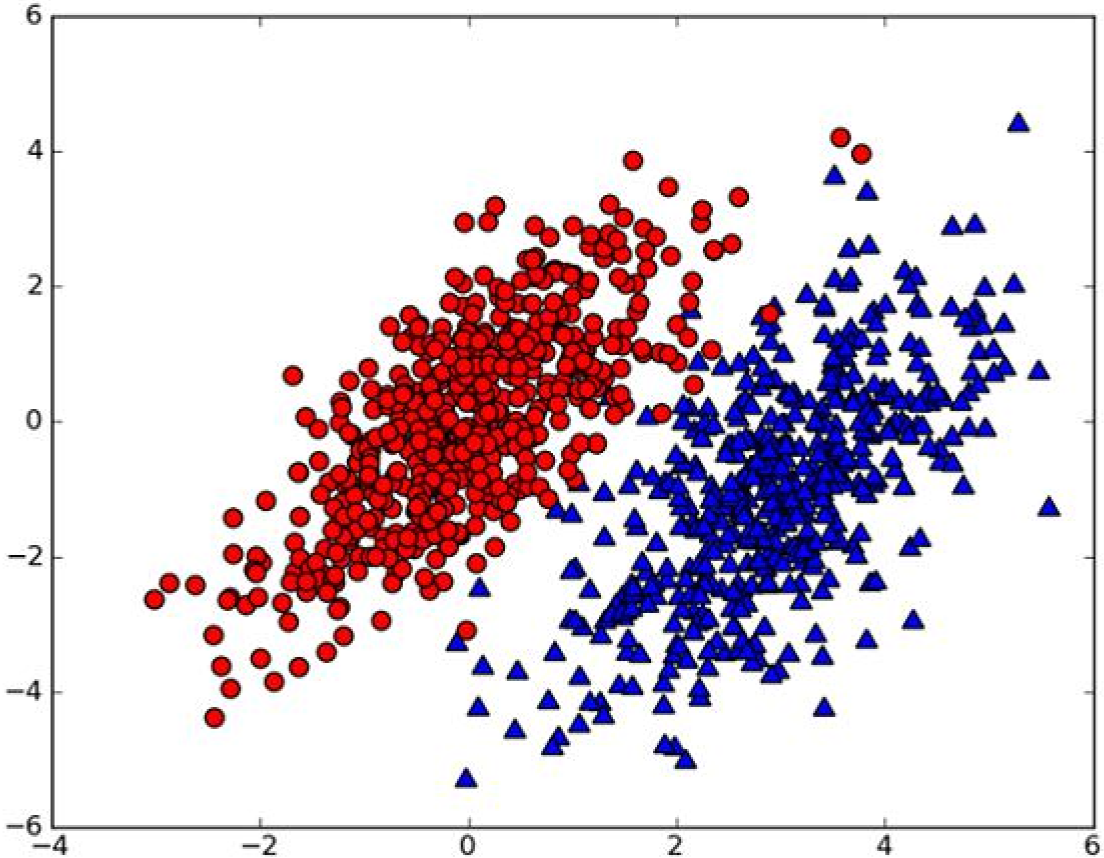


Figure 4.1 Two probability distribu-tions with known parameters describing the distribution

If p1(x, y) > p2(x, y), then the class is 1.

If p2(x, y) > p1(x, y), then the class is 2.

Bayesian Decision Theory

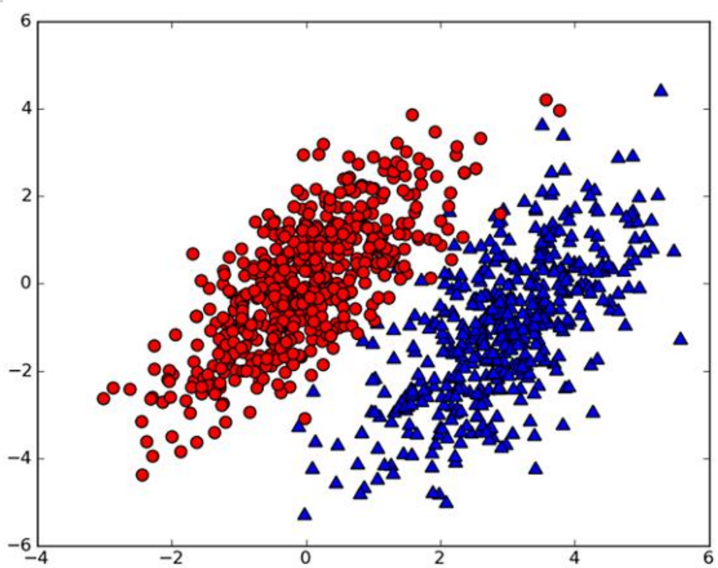
- Bayesian decision theory just choosing the decision with the highest probability
- From chapter 1~3, you can
  - Use kNN from chapter 1, and do 1,000 distance calculations
  - Use decision trees from chapter 2, and make a split of the data once along the x-axis and once along the y-axis
  - Compute the probability of each class, and compare them

Bayes' Rule

- Conditional Probability: P(A|B)
- In our case, P(x|c), x=data, c=class
- Bayes' Rule (貝式定理)

$$p(c|x) = \frac{p(x|c)\,p(c)}{p(x)}$$

Recall the Example…



$$p(c_i|x,y) = \frac{p(x,y|c_i)p(c_i)}{p(x,y)}$$

If $P(c_1|x, y) > P(c_2|x, y)$, the class is c_1.

If $P(c_1|x, y) < P(c_2|x, y)$, the class is c_2.

Document Classification

- Look at the documents by the words used
- Treat the presence or absence of each word as a feature
- What will you get?
- Two more assumptions
  - Feature independence
    - Statistics tells us that if we need N samples for one feature, we need N^M for M features, which grows very fast!
    - Assuming feature independence will reduce to MxN
  - Every feature is equally important

General Approach

General approach to naïve Bayes

1. Collect: Any method. We'll use RSS feeds in this chapter.
2. Prepare: Numeric or Boolean values are needed.
3. Analyze: With many features, plotting features isn't helpful. Looking at histograms is a better idea.
4. Train: Calculate the conditional probabilities of the independent features.
5. Test: Calculate the error rate.
6. Use: One common application of naïve Bayes is document classification. You can use naïve Bayes in any classification setting. It doesn't have to be text.

Creating Data

Listing 4.1 Word list to vector function

```
def loadDataSet():
    postingList=[['my', 'dog', 'has', 'flea', \
                  'problems', 'help', 'please'],
                 ['maybe', 'not', 'take', 'him', \
                  'to', 'dog', 'park', 'stupid'],
                 ['my', 'dalmation', 'is', 'so', 'cute', \
                   'I', 'love', 'him'],
                 ['stop', 'posting', 'stupid', 'worthless', 'garbage'],
                 ['mr', 'licks', 'ate', 'my', 'steak', 'how',\
                   'to', 'stop', 'him'],
                 ['quit', 'buying', 'worthless', 'dog', 'food', 'stupid']]
    classVec = [0,1,0,1,0,1]    #1 is abusive, 0 not
    return postingList,classVec

def createVocabList(dataSet):
    vocabSet = set([])
    for document in dataSet:
        vocabSet = vocabSet | set(document)
    return list(vocabSet)

def setOfWords2Vec(vocabList, inputSet):
    returnVec = [0]*len(vocabList)
    for word in inputSet:
        if word in vocabList:
            returnVec[vocabList.index(word)] = 1
        else: print "the word: %s is not in my Vocabulary!" % word
    return returnVec
```

1 Create an empty set

2 Create the union of two sets

3 Create a vector of all 0s

Calculating Probabilities

$$p(c_i|w) = \frac{p(w|c_i)p(c_i)}{p(w)}$$

Count the number of documents in each class
for every training document:
  for each class:
    if a token appears in the document → increment the count for that token
    increment the count for tokens
  for each class:
   for each token:
      divide the token count by the total token count to get conditional probabilities
  return conditional probabilities for each class

Feature Independence Assumption

$$p(w_0, w_1, w_2 .. w_N | c_i)$$

will result in

$$p(w_0|c_i)p(w_1|c_i)p(w_2|c_i)...p(w_N|c_i)$$

Training Function

Listing 4.2 Naïve Bayes classifier training function

```
def trainNB0(trainMatrix,trainCategory):
    numTrainDocs = len(trainMatrix)
    numWords = len(trainMatrix[0])
    pAbusive = sum(trainCategory)/float(numTrainDocs)
    p0Num = zeros(numWords); p1Num = zeros(numWords)
    p0Denom = 0.0; p1Denom = 0.0
    for i in range(numTrainDocs):
        if trainCategory[i] == 1:
            p1Num += trainMatrix[i]
            p1Denom += sum(trainMatrix[i])
        else:
            p0Num += trainMatrix[i]
            p0Denom += sum(trainMatrix[i])
    p1Vect = p1Num/p1Denom          #change to log()
    p0Vect = p0Num/p0Denom          #change to log()
    return p0Vect,p1Vect,pAbusive
```

1 Initialize probabilities

2 Vector addition

3 Element-wise division

Problems of Training Function

- Problem 1: Easy to get 0 because of zeros()

$$p(w_0|c_i)p(w_1|c_i)p(w_2|c_i)...p(w_N|c_i)$$

```
p0Num = ones(numWords); p1Num = ones(numWords)
p0Denom = 2.0; p1Denom = 2.0
```

- Problem 2: underflow
  - Small number(probability) multiplication can easily underflow
  - Solution: log function. ln(a*b) = ln(a)+ln(b)
  - Both underflow and round-off problems avoided

```
p1Vect = log(p1Num/p1Denom)
p0Vect = log(p0Num/p0Denom)
```

Log Function Characteristic

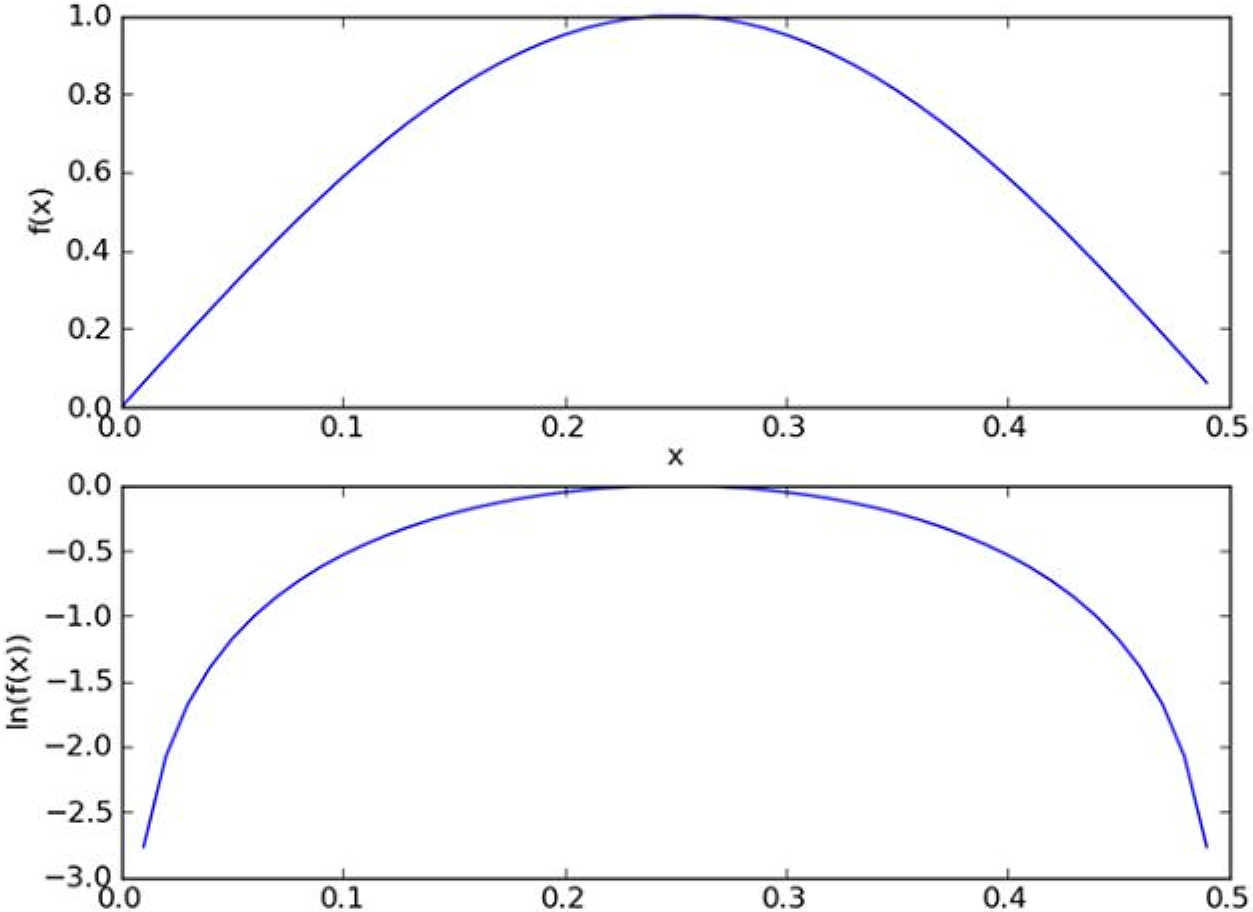


Figure 4.4 Arbitrary functions `f(x)` and `ln(f(x))` increasing together. This shows that the natural log of a function can be used in place of a function when you're interested in finding the maximum value of that function.

Naïve Bayes Classification

Listing 4.3 Naïve Bayes classify function

```
def classifyNB(vec2Classify, p0Vec, p1Vec, pClass1):
    p1 = sum(vec2Classify * p1Vec) + log(pClass1)
    p0 = sum(vec2Classify * p0Vec) + log(1.0 - pClass1)
    if p1 > p0:
        return 1
    else:
        return 0

def testingNB():
    listOPosts,listClasses = loadDataSet()
    myVocabList = createVocabList(listOPosts)
    trainMat=[]
    for postinDoc in listOPosts:
        trainMat.append(setOfWords2Vec(myVocabList, postinDoc))
        p0V,p1V,pAb = trainNB0(array(trainMat),array(listClasses))
        testEntry = ['love', 'my', 'dalmation']
        thisDoc = array(setOfWords2Vec(myVocabList, testEntry))
        print testEntry,'classified as: ',classifyNB(thisDoc,p0V,p1V,pAb)
        testEntry = ['stupid', 'garbage']
        thisDoc = array(setOfWords2Vec(myVocabList, testEntry))
        print testEntry,'classified as: ',classifyNB(thisDoc,p0V,p1V,pAb)
```

1 Element-wise multiplication

Set-of-words vs. Bag-of-words

- Set-of-words model only record whether the words present or not
- Bag-of-words model record the occurrences of the words

Listing 4.4 Naïve Bayes bag-of-words model

```
def bagOfWords2VecMN(vocabList, inputSet):
    returnVec = [0]*len(vocabList)
    for word in inputSet:
        if word in vocabList:
            returnVec[vocabList.index(word)] += 1
    return returnVec
```

Example: Spam Mail Classification

Example: using naïve Bayes to classify email

1. Collect: Text files provided.
2. Prepare: Parse text into token vectors.
3. Analyze: Inspect the tokens to make sure parsing was done correctly.
4. Train: Use `trainNB0()` that we created earlier.
5. Test: Use `classifyNB()` and create a new testing function to calculate the error rate over a set of documents.
6. Use: Build a complete program that will classify a group of documents and print misclassified documents to the screen.

Cross Validation

- Hold-out cross validation
  - Randomly selecting a portion of our data for the training set and a portion for the test set
  - Do multiple times and take the average error rate

Example: Reveal Local Attitudes from Personal Ads

- Import RSS feeds
- Should consider removal of stop words
- Read the textbook for further references

```
>>> bayes.getTopWords(ny,sf)
the error rate is:  0.2
SF**SF**SF**SF**SF**SF**SF**SF*:
love
time
will
there
hit
send
francisco
female
NY**NY**NY**NY**NY**NY**NY**NY*:
friend
people
will
single
sex
female
night
420
relationship
play
hope
```