The key of this dictionary represents the true class of the instances. For example, the first line represents results for instances whose true classification is 35 mpg. The value for each key is another dictionary that represents how our classifier classified the instances. For example, the line

```
    '15':   {'20': 3, '15': 4, '10': 1},
```

represents a test where 3 of the instances that were really 15mpg were misclassified as 20mpg, 4 were classified correctly as 15mpg, and 1 was classified incorrectly as 10mpg.

procedure to perform 10-fold cross-validation.

Finally, we need to write a procedure that will perform 10-fold cross-validation. That is, it builds 10 classifiers. Each classifier is trained on 9 of the buckets and tested on data from the remaining bucket.

```
def tenfold(bucketPrefix, dataFormat):
    results = {}
    for i in range(1, 11):
        c = Classifier(bucketPrefix, i, dataFormat)
        t = c.testBucket(bucketPrefix, i)
        for (key, value) in t.items():
            results.setdefault(key, {})
            for (ckey, cvalue) in value.items():
                results[key].setdefault(ckey, 0)
                results[key][ckey] += cvalue

    # now print results
    categories = list(results.keys())
    categories.sort()
    print(   "\n       Classified as: ")
    header =    "        "
    subheader = "      +"
    for category in categories:
        header += category + "   "
        subheader += "----+"
    print (header)
    print (subheader)
    total = 0.0
    correct = 0.0
```

```
    for category in categories:
        row = category + "    |"
        for c2 in categories:
            if c2 in results[category]:
                count = results[category][c2]
            else:
                count = 0
            row += " %2i |" % count
            total += count
            if c2 == category:
                correct += count
        print(row)
    print(subheader)
    print("\n%5.3f percent correct" %((correct * 100) / total))
    print("total of %i instances" % total)

tenfold("mpgData", "class     num    num    num    num    num    comment")
```

Running the program yields the following results:

```
       Classified as:
        10   15   20   25   30   35   40   45
      +----+----+----+----+----+----+----+----+
10    |  5 |  8 |  0 |  0 |  0 |  0 |  0 |  0 |
15    |  8 | 63 | 14 |  1 |  0 |  0 |  0 |  0 |
20    |  0 | 14 | 67 |  8 |  5 |  1 |  1 |  0 |
25    |  0 |  1 | 13 | 35 | 22 |  6 |  1 |  1 |
30    |  0 |  1 |  3 | 17 | 21 | 14 |  5 |  2 |
35    |  0 |  0 |  2 |  7 | 10 | 13 |  5 |  1 |
40    |  0 |  0 |  1 |  0 |  5 |  5 |  0 |  0 |
45    |  0 |  0 |  0 |  2 |  1 |  1 |  0 |  2 |
      +----+----+----+----+----+----+----+----+

52.551 percent correct
total of 392 instances
```

Kappa Statistic!

At the start of this chapter we mentioned some of the questions we might be interested in answering about a classifier including *How good is this classifier*. We also have been refining our evaluation methods and looked at 10-fold cross-validation and confusion matrices. In the example on the previous pages we determined that our classifier for predicted miles per gallon of selected car models was 53.316% accurate. But does 53.316% mean our classifier is good or not so good? To answer that question we are going to look at one more statistics, the Kappa Statistic.

The Kappa Statistic compares the performance of a classifier to that of a classifier that makes predictions based solely on chance. To show you how this works I will start with a simpler example than the mpg one and again return to the women athlete domain. Here are the results of a classifier in that domain:

	gymnast	basketball player	marathoner	TOTALS
gymnast	35	5	20	60
basketball player	0	88	12	100
marathoner	5	7	28	40
TOTALS	40	100	60	200

I also show the totals for the rows and columns. To determine the accuracy we sum the numbers on the diagonal (35 + 88 + 28 = 151) and divide by the total number of instances (200) to get 151 / 200 = .755

Now I am going to generate another confusion matrix that will represent the results of a random classifier (a classifier that makes random predictions). First, we are going to make a copy of the above table only containing the totals:

	gymnast	basketball player	marathoner	TOTALS
gymnast				60
basketball player				100
marathoner				40
TOTALS	40	100	60	200

Looking at the bottom row, we see that 50% of the time (100 instances out of 200) our classifier classifies an instance as “Basketball Player”, 20% of the time (40 instances out of 200) it classifies an instance as “gymnast” and 30% as “marathoner.”

Classifier:

gymnast: 20%
basketball player: 50%
marathoner: 30%

We are going to use these percentages to fill in the rest of the table. There were 60 total real gymnasts. Our random classifier will classify 20% of those as gymnasts. 20% of 60 is 12 so we put a 12 in the table. It will classify 50% as basketball players (or 30 of them) and 30% as marathoners.

	gymnast	basketball player	marathoner	TOTALS
gymnast	12	30	18	60
basketball player				100
marathoner				40
TOTALS	40	100	60	200

And we will continue in this way. There are 100 real basketball players. The random classifier will classify 20% of them (or 20) as gymnasts, 50% as basketball players and 30% as marathoners. And so on:

	gymnast	basketball player	marathoner	TOTALS
gymnast	12	30	18	60
basketball player	20	50	30	100
marathoner	8	20	12	40
TOTALS	40	100	60	200

To determine the accuracy of the random method we sum the numbers on the diagonal and divide by the total number of instances:

$$P(r) = \frac{12 + 50 + 12}{200} = \frac{74}{200} = .37$$

The Kappa Statistic will tell us how much better the real classifier is compared to this random one. The formula is

$$\kappa = \frac{P(c) - P(r)}{1 - P(r)}$$

where P(c) is the accuracy of the real classifier and P(r) is the accuracy of the random one. In this case the accuracy of the real classifier was .755 and that of the random one was .37 so

$$\kappa = \frac{.755 - .37}{1 - .37} = \frac{.385}{.63} = .61$$

How do we interpret that .61? Does that mean our classifier is poor, good, or great? Here is a chart that will help us interpret that number:

A commonly cited* scale on how to interpret Kappa

< 0:	less than chance performance
0.01-0.20	slightly good
0.21-0.40	fair performance
0.41-0.60	moderate performance
0.61-0.80	substantially good performance
0.81-1.00	near perfect performance

* Landis, JR, Koch, GG. 1977. The measurement of observer agreement for categorical data. Biometrics 33:159-74

sharpen your pencil

Suppose we developed a somewhat silly classifier that predicts the major of current university students based on how well they liked 10 movies. We have a data set of 600 students consisting of computer science (cs) majors, education majors (ed), English majors (eng) and psychology majors (psych). The confusion matrix is shown below. Can you compute the Kappa Statistic and interpret what that statistic means?

	predicted major				
	cs	ed	eng	psych	Total
cs	50	8	15	7	
ed	0	75	12	33	
eng	5	12	123	30	
psych	5	25	30	170	

accuracy = 0.697

solution

How good is our classifier? Can you compute the Kappa Statistic and interpret what that statistic means?

First, we sum all the columns:

	cs	**ed**	**eng**	**psych**	**TOTAL**
SUM	60	120	180	240	600
%	10%	20%	30%	40%	100%

Next, we construct the confusion matrix for the random classifier

predicted major

	cs	ed	eng	psych	Total
cs	8	16	24	32	80
ed	12	24	36	48	120
eng	17	34	51	68	170
psych	23	46	69	92	230
Total	60	120	180	240	600

The accuracy of this random classifier is:
(8 + 24 + 51 + 92) / 600 = (175 / 600) = 0.292

solution continued

So the accuracy of our classifier P(c) is 0.697
and that of the random classifier P(r) is 292

The Kappa Statistic is

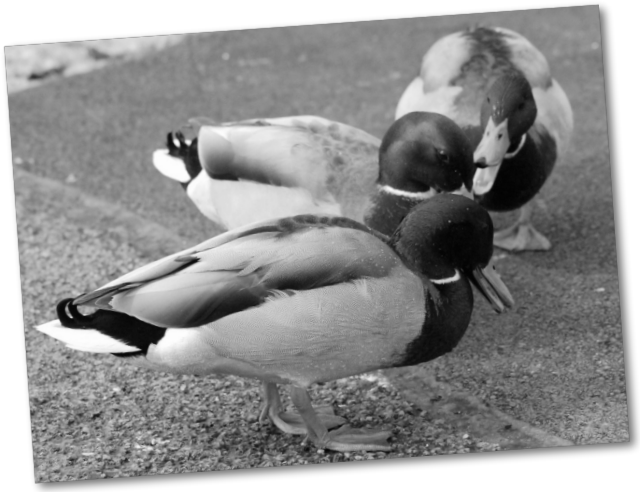
$$\kappa = \frac{P(c) - P(r)}{1 - P(r)}$$

$$\kappa = \frac{0.697 - 0.292}{1 - 0.292} = \frac{0.405}{0.708} = 0.572$$

This suggests our algorithm performs moderately well.

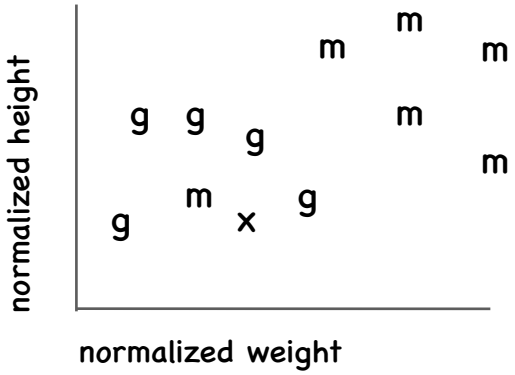
Improvements to the Nearest Neighbor Algorithm!

One trivial example of a classifier is the **Rote Classifier**, which just memorizes the entire training set and only classifies an instance if that instance exactly matches one in the training set. If we only evaluated classifiers on instances in the training data, the Rote Classifier would always be 100% accurate. In real life, the rote classifier is not a good choice because there will be instances we want to classify that are not in the training set. You can view the nearest neighbor algorithm we have been working with as an extension of the rote classifier. Instead of requiring exact matches we are looking at instances that are close matches. Pang-Ning Tan, Michael Steinbach, and Vipin Kumar in their data mining textbook [1] call this the *If it walks like a duck, quacks like a duck, and looks like a duck, then it's probably a duck* approach.



One problem with the nearest neighbor algorithm occurs when we have outliers. Let me explain what I mean by that. And let us return, yet again, to the women athlete domain; this time only looking at gymnasts and marathoners. Suppose we have a particularly short and lightweight marathoner. In diagram form, this data might be represented as on the next page, where *m* indicates 'marathoner' and *g*, 'gymnast.

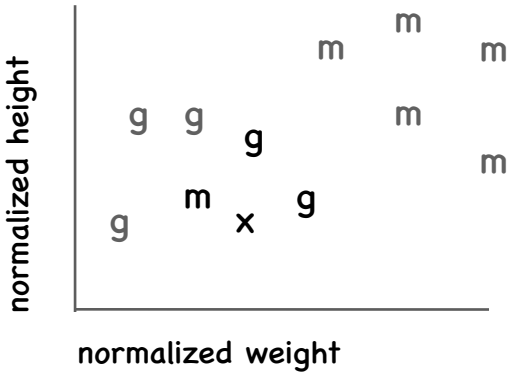
[1] Introduction to Data Mining. 2005. Addison-Wesley



We can see that short lightweight marathoner as the sole *m* in the group of *g*'s. Suppose *x* is an instance we would like to classify. Its nearest neighbor is that outlier *m,* so it would get classified as a marathoner. If we just eyeballed the diagram we would say that *x* is most likely a gymnast since it appears to be in the group of gymnasts.

kNN

One way to improve our current nearest neighbor approach is instead of looking at one nearest neighbor we look at a number of nearest neighbors—k nearest neighbors or kNN. Each neighbor will get a vote and the algorithm will predict that the instance will belong to the class with the highest number of votes. For example, suppose we are using three nearest neighbors ($k = 3$). In that case we have 2 votes for *gymnast* and one for *marathoner,* so we would predict *x* is a gymnast:



So when we are trying to predict a **discrete class** (marathoners, gymnasts, or basketball players, for example) we can use this voting method. The class with the most votes will be the one assigned to the instance. If there is a tie the predicted class will be selected randomly from the classes that are tied. When we are trying to predict a **numeric value** like how many stars a person will give the band *Funky Meters* we can apportion influence from the nearest neighbors to compute a distance-weighted value. Let me parse that out a bit more. Suppose we are trying to predict how well Ben will like *Funky Meters* and Ben's three closest neighbors are Sally, Tara, and Jade. Here are their distances from Ben and their ratings for *Funky Meters*.

User	Distance	Rating
Sally	5	4
Tara	10	5
Jade	15	5

So Sally was closest to Ben and she gave *Funky Meters* a 4. Because I want the rating of the closest person to be weighed more heavily in the final value than the other neighbors, the first step we will do is to convert the distance measure to make it so that the larger the number the closer that person is. We can do this by computing the inverse of the distance (that is, 1 over the distance). So the inverse of Sally's distance of 5 is

$$\frac{1}{5} = 0.2$$

User	Inverse Distance	Rating
Sally	0.2	4
Tara	0.1	5
Jade	0.067	5

Now I am going to divide each of those inverse distances by the sum of all the inverse distances. The sum of the inverse distances is 0.2 + 0.1 + 0.067 = 0.367.

User	Influence	Rating
Sally	0.545	4
Tara	0.272	5
Jade	0.183	5

We should notice two things. First, that the sum of the influence values totals 1. The second thing to notice is that with the original distance numbers Sally was twice as close to Ben as Tara was, and that is preserved in the final numbers were Sally has twice the influence as

Tara does. Finally we are going to multiple each person's influence and rating and sum the results:

predicted Score for Ben

$$= 0.545 \times 4 + 0.272 \times 5 + 0.183 \times 5$$

$$= 2.18 + 1.36 + 0.915 = 4.455$$

sharpen your pencil

I am wondering how well Sofia will like the jazz pianist Hiromi. What is the predicted value given the following data using the k nearest neighbor algorithm with k = 3.?

person	distance from Sofia	rating for Hiromi
Gabriela	4	3
Ethan	8	3
Jayden	10	5

sharpen your pencil - solution

the first thing to do is to compute the inverse (1 over the distance) of each distance:

Person	Inverse Distance	Rating
Gabriela	1/4 = 0.25	3
Ethan	1/8 = 0.125	3
Jayden	1/10 = 0.1	5

The sum of the inverse distances is 0.475. Next I am going to compute the influence of each person by dividing the inverse distance by the sum of each distance

Person	Influence	Rating
Gabriela	0.526	3
Ethan	0.263	3
Jayden	0.211	5

Finally, I multiply the influence by the rating and sum the results:

$$= (0.526 \times 3) + (0.263 \times 3) + (0.211 \times 5)$$

$$= 1.578 + 0.789 + 1.055 = 3.422$$

A new dataset and a challenge!

It is time to look at a new dataset, the Pima Indians Diabetes Data Set developed by the United States National Institute of Diabetes and Digestive and Kidney Diseases.

Astonishingly, over 30% of Pima people develop diabetes. In contrast, the diabetes rate in the United States is 8.3% and in China it is 4.2%.

Each instance in the dataset represents information about a Pima woman over the age of 21 and belonged to one of two classes: a person who developed diabetes within five years, or a person that did not. There are eight attributes:

attributes:

1. Number of times pregnant
2. Plasma glucose concentration
3. Diastolic blood pressure (mm Hg)
4. Triceps skin fold thickness (mm)
5. 2-Hour serum insulin (mu U/ml)
6. Body mass index (weight in kg/(height in m)^2)
7. Diabetes pedigree function
8. Age (years)

Here is an example of the data (the last column represents the class—0=no diabetes; 1=diabetes):

2	99	52	15	94	24.6	0.637	21	0
3	83	58	31	18	34.3	0.336	25	0
5	139	80	35	160	31.6	0.361	25	1
3	170	64	37	225	34.5	0.356	30	1

So, for example, the first woman has had 2 children, has a plasma glucose concentration of 99, a diastolic blood pressure of 52 and so on.

code it - part 1

There are two files on our website. pimaSmall.zip is a zip file containing 100 instances of the data divided into 10 files (buckets). pima.zip is a zip file containing 393 instances. When I used the pimaSmall data with the nearest neighbor classifier we built in the previous chapter using 10-fold cross-validation I got these results:

```
       Classified as:
         0   1
       +----+----+
0      | 45 | 14 |
1      | 27 | 14 |
       +----+----+

59.000 percent correct
total of 100 instances
```

Hint: The python function heapq.nsmallest(n, list) will return a list with the n smallest items.

Here is your task:

Download the classifier code from our website and implement the kNN algorithm. Let us change the initializer method of the class to add another argument, k:

```
def __init__(self, bucketPrefix, testBucketNumber, dataFormat, k):
```

The method signature should look like `def knn(self, itemVector):` It should make use of self.k (remember to set that value in the init method) and return the class (in this Pima Cancer dataset case '0' or '1'). You should also modify the procedure tenfold to pass k to the initializer.

code it - answer

My modification to __init___ was simply:

```
def __init__(self, bucketPrefix, testBucketNumber, dataFormat, k):
        self.k = k
        ...
```

My knn method was

```
def knn(self, itemVector):
    """returns the predicted class of itemVector using k
    Nearest Neighbors"""
    # changed from min to heapq.nsmallest to get the
    # k closest neighbors
    neighbors = heapq.nsmallest(self.k,
                            [(self.manhattan(itemVector, item[1]), item)
                            for item in self.data])
    # each neighbor gets a vote
    results = {}
    for neighbor in neighbors:
        theClass = neighbor[1][0]
        results.setdefault(theClass, 0)
        results[theClass] += 1
    resultList = sorted([(i[1], i[0]) for i in results.items()],
                        reverse=True)
    #get all the classes that have the maximum votes
    maxVotes = resultList[0][0]
    possibleAnswers = [i[1] for i in resultList if i[0] == maxVotes]
    # randomly select one of the classes that received the max votes
    answer = random.choice(possibleAnswers)
    return( answer)
```

My slight modification to tenfold was:

```
def tenfold(bucketPrefix, dataFormat, k):
    results = {}
    for i in range(1, 11):
        c = Classifier(bucketPrefix, i, dataFormat, k)
```

...

You can download this code at guidetodatamining.com. Remember, this is just one way to implement this method, and it is not necessarily the best way.

code it - part 2

Which makes the most difference? Having more data (comparing the results from pimaSmall and pima) or having a better algorithm (comparing k=1 to k=3)?

code it - results!

Here are my accuracy results (k=1 is the nearest neighbor algorithm from the last chapter):

	pimaSmall	pima
k=1	59.00%	71.247%
k=3	61.00%	72.519%

So it seems that roughly tripling the amount of data increases the accuracy much more than improving the algorithm does.

sharpen your pencil

Hmm. 72.519% seems like pretty good accuracy but is it? Compute the Kappa Statistic to find out:

	no diabetes	diabetes
no diabetes	219	44
diabetes	64	66

Performance:

- ☐ slightly good
- ☐ fair
- ☐ moderate
- ☐ substantially good
- ☐ near perfect

sharpen your pencil – answer

	no diabetes	diabetes	TOTAL
no diabetes	219	44	263
diabetes	64	66	130
TOTAL	283	110	393
ratio	0.7201	0.2799	

random (r) classifier:

	no diabetes	diabetes
no diabetes	189.39	73.61
diabetes	93.61	36.39

accuracy

$$p(r) = \frac{189.39 + 36.39}{393}$$
$$= .5745$$

$$\kappa = \frac{P(c) - P(r)}{1 - P(r)} = \frac{.72519 - .5745}{1 - .5745} = \frac{.15069}{.4255} = .35415$$

Only fair performance

More data, better algorithms & a broken bus

Several years ago I was at a conference in Mexico City. This conference was a bit unusual in that it alternated between a day of presentations and a day of touring (the Monarch Butterflies, Inca ruins, etc). The days of touring involved riding long distances on a bus and the bus had a tendency to break down. As a result, a bunch of us PhD types spend a good deal of time standing at the side of road talking to one another as the bus was being attended to. These roadside exchanges were the highpoint of the conference for me. One of the people I talked to was a person named Eric Brill. Eric Brill is famous for developing what is called the Brill tagger, which does part-of-speech tagging. Similar to what we have been doing in the last few chapters, the Brill tagger classifies data—in this case, it classifies words by their part of speech (noun, verb, etc.). The algorithm Brill came up with was significantly better than its predecessors (and as a result Brill became famous in natural language processing circles). At the side of that Mexican road, I got to talking with Eric Brill about improving the performance of algorithms. His view is that you get more of an improvement by getting more data for the training set, than you would by improving the algorithm. In fact, he felt that if he kept the original part-of-speech tagging algorithm and just increased the size of the training data, the improvement would exceed that of his famous algorithm. Although, he said, you cannot get a PhD for just collecting more data, but you can for developing an algorithm with marginally improved performance!

Here's another example. In various machine translation competitions, Google always places at the top. Granted that Google has a large number of very bright people developing great algorithms, much of Google's dominance is due to its enormous training sets it acquired from web.

更多数据 ⇨ Més dades ⇨ More data

This isn't to say that you shouldn't pick the best algorithm for the job. As we have already seen picking a good algorithm makes a significant difference. However, if you are trying to solve a practical problem (rather than publish a research paper) it might not be worth your while to spend a lot of time researching and tweaking algorithms. You will perhaps get more bang for your buck—or a better return on your time—if you concentrate on getting more data.

With that nod toward the importance of data, I will continue my path of introducing new algorithms.

People have used kNN classifiers for

recommending items at Amazon
assessing consumer credit risk
classifying land cover using image analysis
recognizing faces
classifying the gender of people in images
recommending web pages
recommending vacation packages

Naïve Bayes

Let us return yet again to our women athlete example. Suppose I ask you what sport Brittney Griner participates in (gymnastics, marathon running, or basketball) and I tell you she is 6 foot 8 inches and weighs 207 pounds. I imagine you would say basketball and if I ask you how confident you feel about your decision I imagine you would say something along the lines of "pretty darn confident."

Now I ask you what sport Heather Zurich (pictured on the right) plays. She is 6 foot 1 and weighs 176 pounds. Here I am less certain how you will answer. You might say 'basketball' and I ask you how confident you are about your prediction. You probably are less confident than you were about your prediction for Brittney Griner. She could be a tall marathon runner.

Finally, I ask you about what sport Yumiko Hara participates in; she is 5 foot 4 inches tall and weighs 95 pounds. Let's say you say 'gymnastics' and I ask how confident you feel about your decision. You will probably say something along the lines of "not too confident." A number of marathon runner have similar heights and weights.

With the nearest neighbor algorithms, it is difficult to quantify confidence about a classification. With classification methods based on probability—

Bayesian methods—we can not only make a classification but we can make probabilistic classifications—this athlete is 80% likely to be a basketball player, this patient has a 40% chance of getting diabetes in the next five years, the probability of rain in Las Cruces in the next 24 hours is 10%.

Nearest Neighbor approaches are called **lazy learners**. They are called this because when we give them a set of training data, they just basically save—or remember—the set. Each time it classifies an instance, it goes through the entire training dataset. If we have a 100,000 music tracks in our training data, it goes through the entire 100,000 tracks each time it classifies an instance.

Bayesian methods are called **eager learners**. When given a training set eager learners immediately analyze the data and build a model. When it wants to classify an instance it uses this internal model. Eager learners tend to classify instances faster than lazy learners.

The ability to make probabilistic classifications, and the fact that they are eager learners are two advantages of Bayesian methods.

Probability

I am assuming you have some basic knowledge of probability. I flip a coin; what is the probably of it beings a 'heads'? I roll a 6 sided fair die, what is the probability that I roll a '1'? that sort of thing. I tell you I picked a random 19 year old and have you tell me the probability of that person being female and without doing any research you say 50%. These are examples of what is called prior probability and is denoted $P(h)$—the probability of hypothesis h.

So for a coin:

P(heads) = 0.5

For a six sided dice, the probability of rolling a '1':

P(1) = 1/6

If I have an equal number of 19 yr. old male and females →

P(female) = .5

Suppose I give you some additional information about that 19 yr. old—the person is a student at the Frank Lloyd Wright School of Architecture in Arizona. You do a quick Google search, see that the student body is 86% female and revise your estimate of the likelihood of the person being female to 86%.

This we denote as $P(h|D)$ —the probability of the hypothesis h given some data D. For example:

P(female | attends Frank Lloyd Wright School) = 0.86

which we could read as "The probability the person is female given that person attends the Frank Lloyd Wright School is 0.86

The formula is

$$P(A \mid B) = \frac{P(A \cap B)}{P(B)}$$

An example.

In the following table I list some people and the types of laptops and phones they have:

name	laptop	phone
Kate	PC	Android
Tom	PC	Android
Harry	PC	Android
Annika	Mac	iPhone
Naomi	Mac	Android
Joe	Mac	iPhone
Chakotay	Mac	iPhone
Neelix	Mac	Android
Kes	PC	iPhone
B'Elanna	Mac	iPhone

What is the probability that a randomly selected person uses an iPhone?

There are 5 iPhone users out of 10 total users so

$$P(iPhone) = \frac{5}{10} = 0.5$$

What is the probability that a randomly selected person uses an iPhone given that person uses a Mac laptop?

$$P(iPhone \mid mac) = \frac{P(mac \cap iPhone)}{P(mac)}$$

First, there are 4 people who use both a Mac and an iPhone:

$$P(mac \cap iPhone) = \frac{4}{10} = 0.4$$

and the probability of a random person using a mac is

$$P(mac) = \frac{6}{10} = 0.6$$

So the probability of that some person uses an iPhone given that person uses a Mac is

$$P(iPhone \mid mac) = \frac{0.4}{0.6} = 0.667$$

That is the formal definition of posterior probability. Sometimes when we implement this we just use raw counts:

$$P(iPhone|mac) = \frac{number\ of\ people\ who\ use\ a\ mac\ and\ an\ iPhone}{number\ of\ people\ who\ use\ a\ mac}$$

$$P(iPhone \mid mac) = \frac{4}{6} = 0.667$$

sharpen your pencil

What's the probability of a person owning a mac given that they own an iPhone

i.e., P(mac|iPhone)?

tip

If you feel you need practice with basic probabilities please see the links to tutorials at guidetodatamining.com.

sharpen your pencil – solution

What's the probability of a person owning a mac given that they own an iphone

i.e., P(mac|iPhone)?

$$P(mac \mid iPhone) = \frac{P(iPhone \cap mac)}{P(iPhone)}$$

$$= \frac{0.4}{0.5} = 0.8$$

Some terms:

$P(h)$, the probability that some hypothesis h is true, is called the **prior probability** of h. Before we have any evidence, the probability of a person owning a Mac is 0.6 (the evidence might be knowing that the person also owns an iPhone).

P(h|d) is called the **posterior probability** of h. After we observe some data d what is the probability of h? For example, after we observe that a person owns an iPhone, what is the probability of that same person owning a Mac? It is also called **conditional probability**.

In our quest to build a Bayesian Classifier we will need two additional probabilities, $P(D)$ and $P(D|h)$. To explain these consider the following example.

Microsoft Shopping Cart

Did you know that Microsoft makes smart grocery store shopping carts? Yep, they do. Well, actually, Microsoft has contracted with a company called Chaotic Moon to develop them. Chaotic Moon's slogan is *We are smarter than you. We are more creative than you.* You can decide whether they are arrogant, cheeky, or something else. Anyway, the cart combines a shopping cart with a Windows 8 tablet, a Kinect, a Bluetooth speaker (so the cart can talk to you), and a mobile robotics platform (so the cart can follow you around the store).

You come in with your grocery store loyalty card. The cart recognizes you. It has recorded all previous purchases (as well as the purchases of everyone else in the store).

Suppose the cart software wants to determine whether to show you a targeted ad for Japanese Sensha Green Tea. It only wants to show that ad if you are likely to purchase the tea.

The cart system has accumulated the small dataset shown on the next page from other shoppers

P(D) is the probability that some training data will be observed. For example, looking on the next page we see that the probability that the zip code will be 88005 is 5/10 or 0.5.

P(88005) = 0.5

P(D|h) is the probability that some data value holds given the hypothesis. For example, the probability of the zip code being 88005 given that the person bough Sencha Green Tea or *P(88005|Sencha Tea)*.

Customer ID	Zipcode	bought organic produce?	bought Sencha green tea?
1	88005	Yes	Yes
2	88001	No	No
3	88001	Yes	Yes
4	88005	No	No
5	88003	Yes	No
6	88005	No	Yes
7	88005	No	No
8	88001	No	No
9	88005	Yes	Yes
10	88003	Yes	Yes

Zipcodes are a set of postal codes used in the U.S.

In this case we are looking at all the instances where the person bought Sensha Tea. There are 5 such instances. Of those, 3 are with the 88005 zip code.

$$P(88005 \mid SenchaTea) = \frac{3}{5} = 0.6$$

sharpen your pencil

What's the probability of the zip code being 88005 given that the person did not buy Sencha tea?

sharpen your pencil – solution

What's the probability of the zip code being 88005 given that the person did not buy Sencha tea?

There are 5 occurrences of a person not buying Sencha tea. Of those, 2 lived in the 88005 zip code. So

$$P(88005 \mid \neg SenchaTea) = \frac{2}{5} = 0.4$$

That ¬ symbol means 'not'.

sharpen your pencil

This is key to understanding the rest of the chapter so let us practice just a bit more.

1. What is the probability of a person being in the 88001 zipcode (without knowing anything else)?

2. What is the probability of a person being in the 88001 zipcode knowing that they bought Sencha tea?

3. What is the probability of a person being in the 88001 zipcode knowing that they did not buy Sencha tea?

sharpen your pencil – solution

This is key to understanding the rest of the chapter so let us practice just a bit more.

1. What is the probability of a person being in the 88001 zipcode (without knowing anything else)?

   There are 10 total entries in our database and only 3 of them are from 88001 so P(88001) is 0.3

2. What is the probability of a person being in the 88001 zipcode knowing that they bought Sencha tea?
   There are 5 instances of buying Sencha tea and only 1 of them is from the 88001 zipcode so

$$P(88001 \mid SenchaTea) = \frac{1}{5} = 0.2$$

3. What is the probability of a person being in the 88001 zipcode knowing that they did not buy Sencha tea?
   There are 5 instances of not buying Sencha tea and 2 of them are from the 88001 zipcode:

$$P(88001 \mid \neg SenchaTea) = \frac{2}{5} = 0.4$$

Bayes Theorem

Bayes Theorem describes the relationship between $P(h)$, $P(h|D)$, $P(D)$, and $P(D|h)$:

$$P(h \mid D) = \frac{P(D \mid h)P(h)}{P(D)}$$

This theorem is the cornerstone of all Bayesian methods. Usually in data mining we use this theorem to decide among alternative hypotheses. Given the evidence, is the person a gymnast, marathoner, or basketball player. Given the evidence, will this person buy Sencha tea, or not. To decide among alternatives we compute the probability for each hypothesis. For example,

We want to display an ad for Sencha Tea on our smart shopping cart display only if we think that person is likely to buy the tea. We know that person lives in the 88005 zipcode.

There are two competing hypotheses:

The person will buy Sencha tea.
We compute P(buySenchaTea|88005)

The person will not buy Sencha tea.
We compute P(¬buySenchaTea|88005)

We pick the hypothesis with the highest probability!

So if P(buySenchaTea|88005) = 0.6 and

P(¬buySenchaTea|88005) = 0.4

So it is more likely that the person will buy the tea so we will display the ad.

Suppose we work for an electronics store and we have three sales flyers in email form. One flyer features a laptop, another features a desktop and the final flyer a tablet. Based on what we know about each customer we will email that customer the flyer that will most likely generate a sale. For example, I may know that a customer lives in the 88005 zipcode, that she has a college age daughter living at home, and that she goes to yoga class. Should I send her the flyer with the laptop, desktop, or tablet?

Let D represent all that I know about that customer:

- lives in 88005 zipcode
- has college age daughter
- goes to yoga class

My hypotheses are which flyer is the best: laptop, desktop, tablet. So I compute:

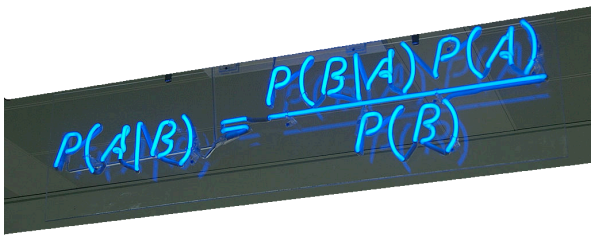
$$P(laptop \mid D) = \frac{P(D \mid laptop)P(laptop)}{P(D)}$$

$$P(desktop \mid D) = \frac{P(D \mid desktop)P(desktop)}{P(D)}$$

$$P(tablet \mid D) = \frac{P(D \mid tablet)P(tablet)}{P(D)}$$

And pick the hypothesis with the highest probability.

More abstractly, in a classification task we have a number of possible hypotheses: h_1, h_2, ...h_n. These hypotheses are the different categories of our task (for example, basketball players, marathoners, gymnasts, or 'will get diabetes', 'will not get diabetes').



$$P(h_1 \mid D) = \frac{P(D \mid h_1)P(h_1)}{P(D)} \quad , \quad P(h_2 \mid D) = \frac{P(D \mid h_2)P(h_2)}{P(D)}$$

$$\ldots \quad P(h_n \mid D) = \frac{P(D \mid h_n)P(h_n)}{P(D)}$$

Once we compute all these probabilities, we will pick the hypothesis with the highest probability. This is called **the maximum a posteriori hypothesis**, or h_{MAP}.

We can translate that English description of calculating the maximum a posteriori hypothesis into the following formula:

$$h_{MAP} = \arg\max_{h \in H} P(h \mid D)$$

H is the set of all the hypotheses. So $h \in H$ means "for every hypothesis in the set of hypotheses." The full formula means something like "for every hypothesis in the set of hypotheses compute $P(h|D)$ and pick the hypothesis with the largest probability." Using Bayes Theorem we can convert that formula to:

$$h_{MAP} = \arg\max_{h \in H} \frac{P(D \mid h)P(h)}{P(D)}$$

So for every hypothesis we are going to compute:

$$\frac{P(D \mid h)P(h)}{P(D)}$$

You might notice that for all these calculations, the denominators are identical—$P(D)$. Thus, they are independent of the hypotheses. If a specific hypothesis has the max probability with the formula used above, it will still be the largest if we did not divide all the hypotheses by $P(D)$. If our goal is to find the most likely hypothesis, we can simplify our calculations:

$$h_{MAP} = \arg\max_{h \in H} P(D \mid h)P(h)$$

To see how this works, we will use an example from Tom M. Mitchell's book, *Machine Learning*. Tom Mitchell is chair of the Machine Learning Department at Carnegie Mellon University. He is a great researcher and an extremely nice guy. On to the example from the book. Consider a medical domain where we want to determine whether a patient has a particular kind of cancer or not. We know that only 0.8% of the people in the U.S. have this form of cancer. There is a simple blood test we can do that will help us determine whether someone has it. The test is a binary one—it comes back either POS or NEG. When the disease is present the test returns a correct POS result 98% of the time; it returns a correct NEG result 97% of the time in cases when the disease is not present.

Our hypotheses:

- The patient has the particular cancer
- The patient does not have that particular cancer.

sharpen your pencil

Let's translate what I wrote above into probability notation. Please match up the English statements below with their associated notations and write in the probabilities. If there is no English statement matching a probability, please write one.

We know that only 0.8% of the people in the U.S. have this form of cancer.

$P(POS|cancer)$ = _____

$P(POS|\neg cancer)$ = _____

When the disease is present the test returns a correct POS result 98% of the time;

$P(cancer)$ = _____

$P(\neg cancer)$ = _____

$P(NEG|cancer)$ = _____

it returns a correct NEG result 97% of the time in cases when the disease is not present

$P(NEG|\neg cancer)$ = _____

sharpen your pencil – solution

We know that only 0.8% of the people in the U.S. have this form of cancer.

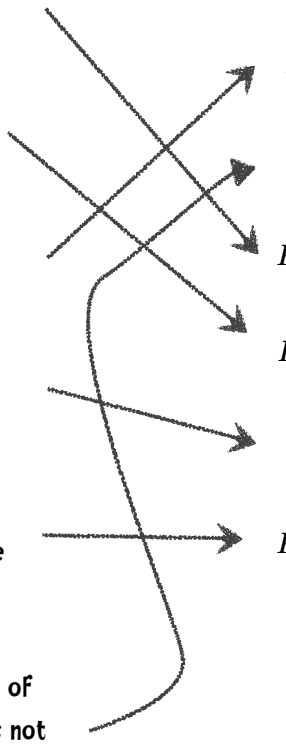
99.2% of people don't have this cancer

When the disease is present the test returns a correct POS result 98% of the time;

When the disease is present the test returns a incorrect NEG result 2% of

it returns a correct NEG result 97% of the time in cases when the disease is not present

it returns an incorrect POS result 3% of the time in cases when the disease is not present



$P(POS|cancer) = 0.98$

$P(POS|\neg cancer) = 0.03$

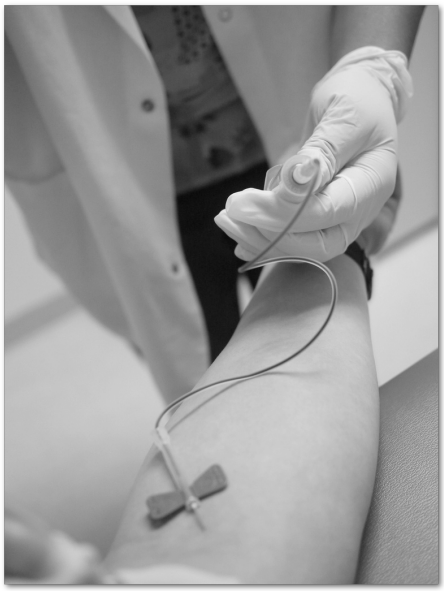
$P(cancer) = 0.008$

$P(\neg cancer) = 0.992$

$P(NEG|cancer) = 0.02$

$P(NEG|\neg cancer) = 0.97$

sharpen your pencil – solution



Suppose Ann, comes into the doctor's office

A blood test for cancer is given and the test result is POS.

This is not looking good for Ann. After all, the test is 98% accurate.

Using Bayes Theorem determine whether it is more likely that Ann has cancer or that she does not.

$P(cancer) = 0.008$

$P(\neg cancer) = 0.992$

$P(POS|cancer) = 0.98$

$P(POS|\neg cancer) = 0.03$

$P(NEG|cancer) = 0.02$

$P(NEG|\neg cancer) = 0.97$

sharpen your pencil – solution

Suppose Ann, comes into the doctor's office
A blood test for the cancer is given and the test result is POS.

This is not looking good for Ann. After all, the test is 98% accurate.

Using Bayes Theorem determine whether it is more likely that Ann has cancer or that she does not.

We are finding the maximum a posteriori probability:

$P(POS \mid cancer)P(cancer) = .98(.008) = .0078$

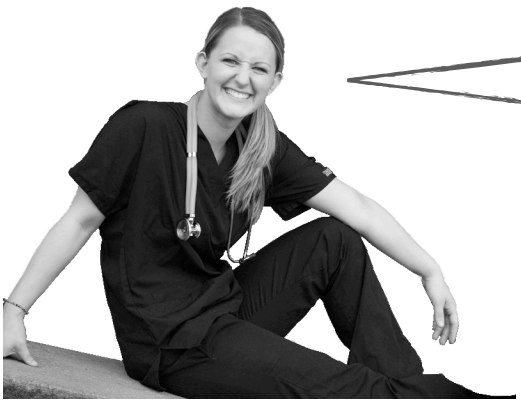
$P(POS \mid \neg\ cancer)\ P(\neg\ cancer) = .03(.992) = .0298$

We select hMAP and classify the patient as not having cancer.

If we want to know the exact probability we can normalize these values by having them sum to 1:

$$P(cancer \mid POS) = \frac{0.0078}{0.0078 + 0.0298} = 0.21$$

Ann has a 21% chance of having cancer.



You may think "That just doesn't make sense. After all, the test is 98% accurate, but yet you re telling me Ann is most likely not to have cancer. "

You are in good company. 85% of medical doctors get the answer wrong as well.

I just didn't make that 85% number up. See, among others,

Casscells, W., Schoenberger, A., and Grayboys, T. (1978): "Interpretation by physicians of clinical laboratory results." N Engl J Med. 299:999-1001.

Gigerenzer, Gerd and Hoffrage, Ulrich (1995): "How to improve Bayesian reasoning without instruction: Frequency formats." Psychological Review. 102: 684-704.

Eddy, David M. (1982): "Probabilistic reasoning in clinical medicine: Problems and opportunities." In D. Kahneman, P. Slovic, and A. Tversky, eds, Judgement under uncertainty: Heuristics and biases. Cambridge University Press, Cambridge, UK.

Here is why the results seem so counterintuitive. Most people see the statistic that 98% of the people who have this particular cancer will have a positive test result and also conclude that 98% of the people who have a positive test result have this particular cancer. This fails to take into account that this cancer affects only 0.8% of the population. Let's say we give the test to everyone in a city of 1 million people. That means that 8,000 people have cancer and 992,000 do not. First, let's consider giving the test to the 8,000 people with cancer. We know that 98% of the time when we give the test to people with cancer the test correctly returns a positive result. So 7,840 people have a correct positive result and 160 of those people with cancer have an incorrect negative result. Now let's turn to the 992,000 people without cancer. When we give the test to them, 97% of the time we get a correct negative result so (992,000 * 0.97) or 962,240 of them have a correct negative result and 30,000 have an incorrect positive result. I have summarized these results on the following page.

	positive test result	negative test result
people with cancer	7,840	160
people without cancer	30,000	962,240

Now, consider Ann getting a positive test result and the data in the 'positive test result' column. 30,000 of the people with a positive test result had no cancer while only 7,840 of them had cancer. So it seems probable that Ann does not have cancer.

> Still don't get it?
> Don't worry. Many people don't.
> After more practice you will gain a better understanding.

Why do we need Bayes Theorem?

Yet again, Bayes Theorem is

$$P(h \mid D) = \frac{P(D \mid h)P(h)}{P(D)}$$

Let us return to the shopping cart example presented earlier. In that example, we obtained the information on the right from customers.

Say we know a customer lives in the 88005 zipcode and our two competing hypotheses are that they will buy Sencha tea or they will not. So:

$P(h_1|D) = P(buySenchaTea|88005)$

and

$P(h_2|D) = P(\neg\ buySenchaTea|88005)$

Customer ID	Zipcode	bought organic produce?	bought Sencha green tea?
1	88005	Yes	Yes
2	88001	No	No
3	88001	Yes	Yes
4	88005	No	No
5	88003	Yes	No
6	88005	No	Yes
7	88005	No	No
8	88001	No	No
9	88005	Yes	Yes
10	88003	Yes	Yes

In this case you may wonder why we need to compute

$$\frac{P(88005 \mid buySenchaTea)P(buySenchaTea)}{P(88005)}$$

when we can just as easily compute $P(buySenchaTea|88005)$ directly from the data in the table. In this simple case you would be correct but for many real world problems it is very difficult to compute $P(h|D)$ directly.

Consider the previous medical example where we were interested in determining whether a person had cancer or not given that a certain test returned a positive result.

$$P(cancer \mid POS) \approx P(POS \mid cancer)P(cancer)$$

$$P(\neg cancer \mid POS) \approx P(POS \mid \neg cancer)P(\neg cancer)$$

It is relatively easy to compute the items on the right hand side. We can estimate $P(POS|cancer)$ by giving the cancer test to a representative sample of people with cancer and $P(POS|\neg\ cancer)$ by giving the test to a sample of people without cancer. $P(cancer)$ seems like a statistic that would be available on government websites and $P(\neg\ cancer)$ is simply

$$1 - P(cancer)$$

However, computing $P(cancer|POS)$ directly would be significantly more challenging. This is asking us to determine the probability that when we give the test to a random average person in the entire population and the test result is POS then that person has cancer. To do this we want a representative sample of the population but since only 0.8% of people have cancer a sample size of 1,000 people would only have 8 people with cancer—far too few to feel that our counts are representative of the population as a whole. So we would need an extremely large sample size. So Bayes Theorem provides a strategy for computing $P(h|D)$ when it is hard to do so directly.

Naïve Bayes

Most of the time we have more evidence than just a single piece of data. In the Sencha tea example we had two types of evidence: zip code and whether the person purchased organic food. To compute the probability of an hypothesis given all the evidence, we simply multiply the individual probabilities. In this example

Code:

tea = Person buy Sencha tea

¬ tea = Person does not buy Sencha tea

P(88005|tea) = probability that a person lives in the 88005 zipcode given that person bought Sencha tea.

etc.

Customer ID	Zipcode	bought organic produce?	bought Sencha green tea?
1	88005	Yes	Yes
2	88001	No	No
3	88001	Yes	Yes
4	88005	No	No
5	88003	Yes	No
6	88005	No	Yes
7	88005	No	No
8	88001	No	No
9	88005	Yes	Yes
10	88003	Yes	Yes

We would like to know whether a person who lives in the 88005 zipcode and bought organic produce will likely buy tea:

$P(tea|88005 \ \& \ organic)$ and for that we simply multiply the probabilities:

$P(tea|88005 \ \& \ organic) = P(88005 \mid tea)\ P(organic \mid tea)\ P(tea) = .6(.8)(.5) = .24$

$P(\neg tea|88005 \ \& \ organic) = P(88005 \mid \neg tea)\ P(organic \mid \neg tea)\ P(\neg tea) = .4(.25)(.5) = .05$

So a person who lives in the trendy 88005 zip code area and buys organic food is more likely to buy Sencha Green tea than not. So let's display the Green Tea ad on the shopping cart display!

Here's how Stephen Baker describes the smart shopping cart technology:

> ... here's what shopping with one of these carts might feel like. You grab a cart on the way in and swipe your loyalty card. The welcome screen pops up with a shopping list. It's based on patterns of your last purchases. Milk, eggs, zucchini, whatever. Smart systems might provide you with the quickest route to each item. Or perhaps they'll allow you to edit the list, to tell it, for example, never to promote cauliflower or salted peanuts again. This is simple stuff. But according to Accenture's studies, shoppers forget an average of 11 percent of the items they intend to buy. If stores can effectively remind us of what we want, it means fewer midnight runs to the convenience store for us and more sales for them.
>
> Baker. 2008. P49.

The Numerati

I've mentioned this book by Stephen Baker several times. I highly encourage you to read this book. The paperback is only $10 and it is a good late night read.

i100 i500

Let's say we are trying to help iHealth, a company that sells wearable exercise monitors that compete with the Nike Fuel and the Fitbit Flex. iHealth sells two models that increase in functionality: the i100 and the i500:

iHealth100:
heart rate, GPS (to compute miles per hour, etc), wifi to automatically connect to iHealth website to upload data.

iHealth500:
i100 features + pulse oximetry (oxygen in blood) + free 3G connection to iHealth website

They sell these online and they hired us to come up with a recommendation system for their customers. To get data to build our system when someone buys a monitor, we ask them to fill out the questionnaire. Each question in the questionnaire relates to an attribute. First, we ask them what their main reason is for starting an exercise program and have them select among three options: health, appearance or both. We ask them what their current exercise level is: sedentary, moderate, or active. We ask them how motivated they are: moderate or aggressive. And finally we ask them if they are comfortable with using technological devices. Our results are as follows.

Main Interest	Current Exercise Level	How Motivated	Comfortable with tech. Devices?	Model #
both	sedentary	moderate	yes	i100
both	sedentary	moderate	no	i100
health	sedentary	moderate	yes	i500
appearance	active	moderate	yes	i500
appearance	moderate	aggressive	yes	i500
appearance	moderate	aggressive	no	i100
health	moderate	aggressive	no	i500
both	active	moderate	yes	i100
both	moderate	aggressive	yes	i500
appearance	active	aggressive	yes	i500
both	active	aggressive	no	i500
health	active	moderate	no	i500
health	sedentary	aggressive	yes	i500
appearance	active	moderate	no	i100
health	sedentary	moderate	no	i100

sharpen your pencil

Using the naïve Bayes method, which model would you recommend to a person whose
main interest is health
current exercise level is moderate
is moderately motivated
and is comfortable with technological devices

Turn the page if you need a hint!