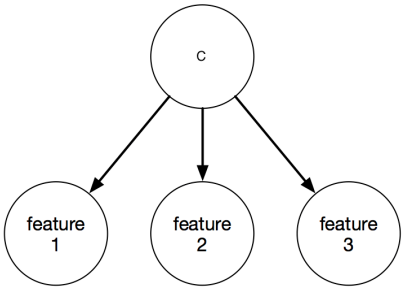
Project 2: Naïve Bayes



"All models are wrong, but some are useful."
-- George E.P. Box

Introduction

Aðalbrandr is visiting St. Louis from Norway and has been having the hardest time distinguishing boys and girls because of the weird American names like Jack and Jane. This has been causing lots of problems for Aðalbrandr when he goes on dates. When he heard that Washington University has a Machine Learning class, he asked that we help him identify the gender of a person based on their name to the best of our ability. In this project, you will implement Naïve Bayes to predict if a name is male or female. But before you get started, you need to update your code on whatever computer you want to use.

Simply call:

```
svn update
```

The code for this project (`project2`) consists of several files, some of which you will need to read and understand in order to complete the assignment, and some of which you can ignore.

Files you'll edit:

`partners.txt`	If you work in a group, this file should contain the two wustlkeys of you and your group partner. These should be in two separate lines (the first two lines). There should be nothing else in this file. Please make sure that your partner also puts your wustlkey in his/her `partners.txt` file, as **project partnerships must be reciprocal**. If you don't have a partner then just leave the file alone.
`naivebayesPXY.m`	Computes P(X|Y)
`naivebayesPY.m`	Computes P(Y)
`naivebayes.m`	Computes log probability log P(Y|X = x1)
`naivebayesCL.m`	Turns the Naïve Bayes output into a linear classifier
`classifyLinear.m`	Makes predictions with a linear classifier.

Files you might want to look at:

`name2features.py`	Converts a name to a feature vector. You do not need to edit this,

	but you may want to!
`genTrainFeatures.m`	Calls name2features.py.
`whoareyou.m`	Allows you to test your linear classifier interactively.
`example_tests.m`	Describes several unit tests to find obvious bugs in your implementation.
`boys.train`	The training examples for boys names.
`girls.train`	The training examples for girls names.

How to submit: You can commit your code with subversion, with the command line

```
svn commit -m "some insightful comment"
```

where you should substitute "some meaningful comment" with something that describes what you did. You can submit as often as you want until the deadline. Please be aware that the last submission determines your grade.

Evaluation: Your code will be autograded for technical correctness. Please *do not* change the names of any provided functions or classes within the code, or you will wreak havoc on the autograder. If your program outputs too much on the console the autograder will stop doing it's job (so make sure you terminate statements with a semi-colon). However, the correctness of your implementation -- not the autograder's output -- will be the final judge of your score. If necessary, we will review and grade assignments individually to ensure that you receive due credit for your work.

Academic Dishonesty: We will be checking your code against other submissions in the class for logical redundancy. If you copy someone else's code and submit it with minor changes, we will know. These cheat detectors are quite hard to fool, so please don't try. We trust you all to submit your own work only; *please* don't let us down. If you do, we will pursue the strongest consequences available to us.

Getting Help: You are not alone! If you find yourself stuck on something, contact the course TAs for help. Office hours and Piazza are there for your support; please use them. If you can't make our office hours, let us know and we will schedule more. We want these projects to be rewarding and instructional, not frustrating and demoralizing. But, we don't know when or how to help unless you ask.

Of boys and girls

Take a look at the files `girls.train` and `boys.train`. For example with the unix command

```
cat girls.train
```

```
...
Addisyn
Danika
Emilee
Aurora
Julianna
Sophia
Kaylyn
Litzy
Hadassah
```

Believe it or not, these are all more or less common girl names. The problem with the current file is that the names are in plain text, which makes it hard for a machine learning algorithm to do anything useful with them. We therefore need to transform them into some vector format, where each name becomes a vector that represents a point in some high dimensional input space.

That is exactly what the Python script `name2features.py` does.

```
#!/usr/bin/python
import sys

def hashfeatures(name):
  B=128; # number of dimensions in our feature space
  v=[0]*B; # initialize the vector to be all-zeros
  name=name.lower() # make all letters lower case
  # hash prefixes & suffixes
  for m in range(3):
    featurestring='prefix'+name[0:min(m+1,len(name))]
    v[hash(featurestring) % B]=1
    featurestring='suffix'+name[-1:-min(m+2,len(name)+1):-1]
    v[hash(featurestring) % B]=1
  return v

for name in sys.stdin:
  print ','.join(map(str,hashfeatures(name.strip())))
```

You can call this function (in unix/linux/Mac OSX) from the Shell (**not the Octave/MATLAB shell**) with `cat girls.train | python name2features.py`. It reads every name in the girls.train file and converts it into a 128-dimensional feature vector.

Can you figure out what the features are? (Understanding how these features are constructed will help you later on in the competition.)

We have provided you with a MATLAB function `genTrainFeatures.m`, which calls this Python script, transforms the names into features and loads them into Octave. You can call

```
[x,y]=genTrainFeatures()
```

to load in the features and the labels of all boys and girls names.

The Naïve Bayes Classifier

The Naïve Bayes classifier is a linear classifier based on Bayes Rule. The following questions will ask you to finish these functions in a pre-defined order.
Remember: as a general rule, you should avoid tight loops at all cost.

(a) Estimate the class probability P(Y) in **`naivebayesPY.m`** . This should return the probability that a sample in the training set is positive or negative, independent of its features.

(b) Estimate the conditional probabilities P(X|Y) in **`naivebayesPXY.m`** . Use a **multinomial** distribution as model. This will return the probability vectors for all features given a class label.

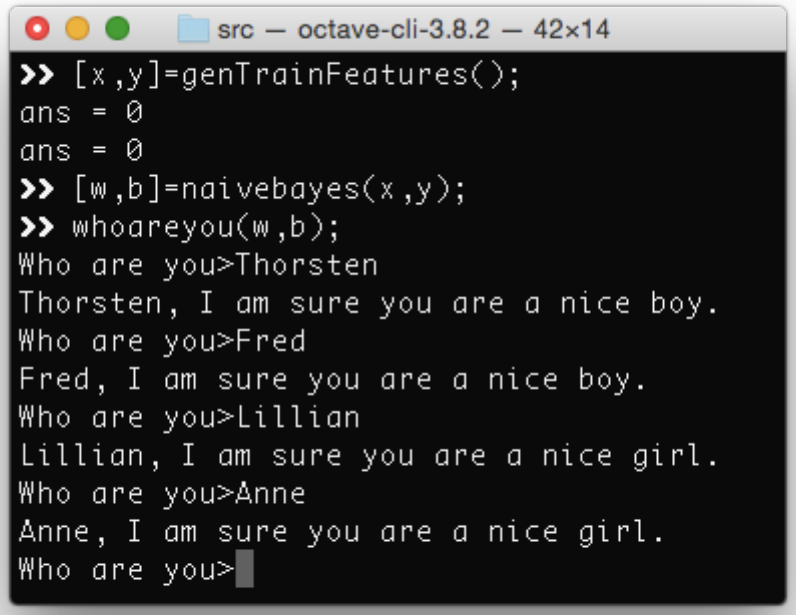
(c) Solve for the log ratio, $\log\left(\frac{P(Y=1|X)}{P(Y=-1|X)}\right)$, using Bayes Rule. Implement this in **`naivebayes.m`**.

(d) Naïve Bayes can also be written as a linear classifier. Implement this in **`naivebayesCL.m`**

(e) Implement **`classifyLinear.m`** that applies a linear weight vector and bias to a set of input vectors and outputs their predictions. (You can use your answer from the previous project.) You can now test your training error with

```
>> [x,y]=genTrainFeatures();
>> [w,b]=naivebayesCL(x,y);
>> preds=classifyLinear(x,w,b);
>> trainingerror=sum(preds~=y)/length(y)
trainingerror =  0.2367
```

Once your Naïve Bayes code is implemented, you can try the function **`whoareyou.m`** to test names interactively.



Hints

Tests. To test your code you can run `example_tests.m`, which describes and partially implements several unit tests. Feel free to implement your own tests to better debug and test your implementation. Those tests are a subset of what we will use in the audograder to grade your submission.

Feature Extraction (Quality Evaluation)

30% of the grade for your project2 submission will be assigned by how well your classifier performs on a secret test set. If you want to improve your classifier modify `name2features.py`. The autograder will use your Python script to extract features and train your classifier on the same names training set.

Credits: Project adapted from Kilian Weinberger (Thanks for sharing!).