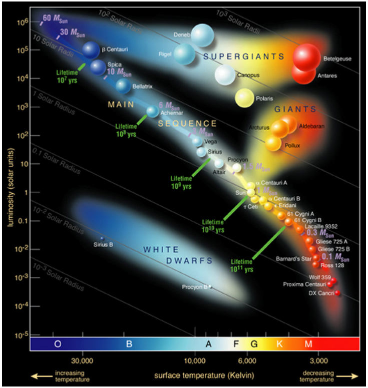
Using Multinomial Naive Bayes Classifiers at United Way of the Columbia-Willamette

bmarron

Sometimes classification is surprisingly straightforward ...

$$X = (x_1, x_2, x_3)$$

$$C = \{c_1, c_2, c_3, c_4\}$$



Hertzsprung-Russell diagram

...and sometimes not.

$X = (?)$

$C = \{?\}$

United Way's *Breaking the Cycle of Childhood Poverty* campaign wants to:

1. Identify combinations of programs and services that can lead to predictable outcomes of success.

2. Provide partner organizations with tools for in-house efficacy assessment of social service provisioning.

Recommended:

1. Write a proposal for a pilot project.

... a pilot project for the development of machine learning tools ...
... for use by social service organizations ...
... use supervised learning algorithms called Naive Bayes classifiers ...

Machine Learning Tools for
Social Service Providers Funded by
the United Way of the Columbia-Willamette

Bruce Marron, Portland State University

Alejandro Queral, United Way of the Columbia-Willamette

Multinomial Naive Bayes

2. Write a project plan document.

... what is to be done, how it will be done ...

... who will be involved, who will have access to the results ...

... ethical safeguards, transparency guarantees, and legal frameworks ...

... for the entire project life cycle ...

United Way of the Columbia-Willamette — MachineLearning Pilot Project MLPP1.20160516.0.1

Document Control Sheet

Document definition:

Document Title:	Project Plan for the Machine Learning Pilot Project
Document Number:	MLPP1
Main Author(s:)	Bruce Marron
Dissemination:	United Way of the Columbia-Willamette and its Partner Organizations

Version history:

Version	Date	Author	Summary of changes
0.1	16 May 2016	Bruce Marron	Document creation

Approval:

	Name	Date
Prepared	Bruce Marron	16 May 2016
Reviewed	Alejandro Queral	date
Approved	name	date

DRAFT

Multinomial Naive Bayes

3. Build and evaluate a set of Multinomial Naive Bayes classifiers.

...open source...

...readily available software...

...use Python-based, “MultinomialNB” from scikits-learn...

Naive Bayes Classifiers

Naive Bayes classifiers are supervised learning algorithms that apply Bayes' theorem with the "naive" assumption of independence between every pair of features.

Letting C_k stand for the class variable with k states and $\mathbf{x} = (x_1, \ldots, x_n)$ stand for the features (data), then the Naive Bayes assumption is,

$$p(x_i | x_{i+1}, \ldots, x_n, C_k) = p(x_i | C_k)$$

and a Naive Bayes classifier gives,

$$p(C_k|\mathbf{x}) = \frac{p(C_k)\ p(\mathbf{x}|C_k)}{p(\mathbf{x})}$$

$$P(C_k \mid x_1, \ldots, x_n) = \frac{P(C_k)P(x_1, \ldots x_n \mid C_k)}{P(x_1, \ldots, x_n)}$$

$$P(C_k \mid x_1, \ldots, x_n) = \frac{P(C_k)\prod_{i=1}^{n} P(x_i \mid C_k)}{P(x_1, \ldots, x_n)}$$

$$P(C_k \mid x_1, \ldots, x_n) \propto P(C_k)\prod_{i=1}^{n} P(x_i \mid C_k)$$

Class assignment is then taken as,

$$\hat{C}_k = \arg\max_{C_k} \left[P(C_k) \prod_{i=1}^{n} P(x_i \mid C_k)\right]$$

(i.e., the class with the highest probability given the data)

The Trial Run

Run MultinomialNB on Alpaydin's "OptDigits" database for the optical recognition of handwritten digits. Available from the UC Irvine Machine Learning Repository.

- Training set has 3823 cases
- Test set has 1797 cases
- Each case has a feature vector with
  - 63 measured values ranging from 0-16 (whole digits)
  - a single class attribute corresponding to the natural number set, $N = \{0, 1, 2, ..., 9\}$

The Data

`tr_d` = (3823 x 64) (`n_cases, n_features`)
`te_d` = (1787 x 64) (`n_cases, n_features`)

Head of `tr_d`:

```
0,1,6,15,12,1,0,0,0,7,16,6,6,10,0,0,0,8,16,2,...,14,7,1,0,0,0
0,0,10,16,6,0,0,0,0,7,16,8,16,5,0,0,0,11,16,...,16,15,3,0,0,0
0,0,8,15,16,13,0,0,0,1,11,9,11,16,1,0,0,0,0,...,14,0,0,0,0,7
0,0,0,3,11,16,0,0,0,0,5,16,11,13,7,0,0,3,15,...,1,15,2,0,0,4
0,0,5,14,4,0,0,0,0,0,13,8,0,0,0,0,0,3,14,4,...,12,14,7,0,0,6
```

Multinomial Naive Bayes

/home/bmarron/Desktop/MNBayes1.py PrintDate: May 29, 2016 1

United Way MLPP: MultinomialNB Optical Digits Example

```python
# -*- coding: utf-8 -*-


"""
Title:                  Sample MNBayes1
Project Descriptor:     Machine Learning Pilot Project (United Way)
Project ID:             2016SoE013 (STAT_570_Consulting)
Record:
Author:                 bmarron
Origin Date:            28 May 2016
"""

"""
This script provides an example of the use of the Multinomial Bayes
algorithm as applied to the Optical Recognition of Handwritten Digits
dataset. The original source of the dataset:
        E. Alpaydin, C. Kaynak
        Department of Computer Engineering
        Bogazici University, 80815 Istanbul Turkey
        alpaydin@boun.edu.tr
        July 1998

The dataset is available at UC Irivine Machine Learning Repository.
"""

"""
Important websites:
    #UCI Machine Learning Repository
http://archive.ics.uci.edu/ml/

    #Python machine learning tools
http://scikit-learn.org/
"""

#%% Import packages


import numpy as np
import cPickle
import urllib
import csv
import operator
import pylab as pl
import pandas as pd
from time import time
from sklearn import cross_validation
from sklearn.utils.fixes import bincount
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.naive_bayes import MultinomialNB
from sklearn.svm import SVC
from sklearn.svm import LinearSVC

#%%

#Download and save the raw, UCI opttical digits data and data info sheet

'''
http://archive.ics.uci.edu/ml/datasets/Optical+Recognition+of+Handwritten+Digits
http://archive.ics.uci.edu/ml/machine-learning-databases/optdigits/optdigits.tra
http://archive.ics.uci.edu/ml/machine-learning-databases/optdigits/optdigits.tes
'''
```

/home/bmarron/Desktop/MNBayes1.py PrintDate: May 29, 2016 2

```python

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=.5,
    random_state=random_state)

#%% Confirm data content with info sheet about training dataset

#Data sheet: 0=376, 1=389, 2=380, 9=382


test0 = np.where(raw_optdigits_tr[:, 64] == 0)
test0 = list(test0[0])

test1 = np.where(raw_optdigits_tr[:, 64] == 1)
test1 = list(test1[0])

test2 = np.where(raw_optdigits_tr[:, 64] == 2)
test2 = list(test2[0])

test9 = np.where(raw_optdigits_tr[:, 64] == 9)
test9 = list(test9[0])

print "zeros: %s" %(len(test0))
print "ones: %s" %(len(test1))
print "twos: %s" %(len(test2))
print "nines: %s" %(len(test9))

#%% Split training dataset(tr_d) and test dataset(te_d) into features (X)

#and class (y)


tr_d_X, tr_d_y = raw_optdigits_tr[:,0:64], raw_optdigits_tr[:, 64]
te_d_X, te_d_y = raw_optdigits_te[:,0:64], raw_optdigits_te[:, 64]


#%% Save the split (processed) data


# Save tr_d_X

with open("tr_d_X.pkl", 'wb') as f:
    cPickle.dump(tr_d_X, f, protocol=2)

# Save tr_d_y

with open("tr_d_y.pkl", 'wb') as f:
    cPickle.dump(tr_d_y, f, protocol=2)

# Save te_d_X

with open("te_d_X.pkl", 'wb') as f:
    cPickle.dump(te_d_X, f, protocol=2)

# Save te_d_y

with open("te_d_y.pkl", 'wb') as f:
    cPickle.dump(te_d_y, f, protocol=2)

#%%

```

The Results

```
Testbenching a MultinomialNB classifier...
MultinomialNB(alpha=0.03, class_prior=None, fit_prior=True)
done in 0.016397s
Predicting the outcomes of the testing set
done in 0.006701s
```

Accuracy: 0.89
$$\left(\frac{TP + TN}{\Sigma Total Population}\right)$$

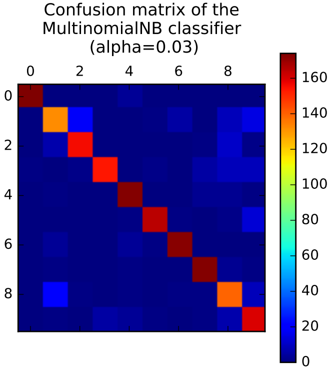
Recall: 0.73-0.98
$$\left(\frac{TP}{(TP + FN)}\right)$$

Precision:0.76-0.99
$$\left(\frac{TP}{(TP + FP)}\right)$$

Multinomial Naive Bayes

Confusion matrix:

	0	1	2	3	4	5	6	7	8	9
0	174	0	0	0	4	0	0	0	0	0
1	0	132	19	0	0	1	6	0	9	15
2	0	7	156	0	0	0	0	1	11	2
3	1	0	2	154	0	2	0	6	9	9
4	0	1	0	0	173	0	0	3	3	1
5	0	0	0	0	1	165	1	0	2	13
6	0	4	0	0	4	1	172	0	0	0
7	0	1	0	0	1	0	0	173	3	1
8	0	21	1	0	1	0	1	1	140	9
9	0	1	0	6	4	1	0	1	7	160

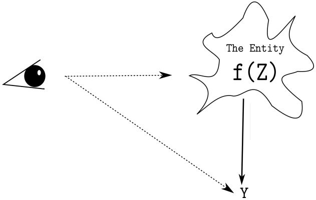


Next Steps

1. Finalize the Machine Learning Pilot Project proposal document.
2. Finalize the Project Plan document.
3. Obtain a test dataset from Metropolitan Family Services.
4. Run a Multinomial NB classifier.
5. Evaluate and share results.
6. Move to Phase II.

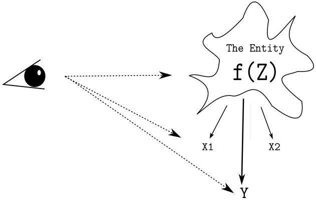
Thanks!

Scientific Inference: From reality to models and back again



- We observe an entity in Nature that we suspect generates non-random patterns of information
- Our states of knowledge about the causal relationships and processes, $f(\cdot)$, that are operating as well as about the inputs, Z, are limited; often severely
- We assume that some observable outcome, Y, is causally related to the entity as $f(Z) \Longrightarrow \{Y\}$

Scientific inference: From reality to models and back again



- We assume that some observable and measurable attributes (data), $\{X1, X2\}$ are logically related to the entity's internal processes as, $\{X1, X2\}|f(Z)$
- Lacking full knowledge of the entity's processes, we use a probability model and consider $X1, X2, Y$ as random variables with a joint probability distribution function
- Lacking complete datasets, we accept sampled datasets
- We make inductive inferences from the sampled datasets back to $f(Z)$ by assuming sampling distributions, evaluating our prior knowledge, and using the (weaker) syllogisms of plausible reasoning coupled with probability theory