

Topic 6 (Pt 1) :

Python Machine Learning for NLP



What is machine learning?

- In a nutshell, machine learning is basically **learning from data**
- Machine learning **automatically learns the relations from certain features in our data**
- Two of the most common high level problems that we can solve with machine learning are :
 - **Classification (supervised ML)** : given the classes, decides what class to put something in (labelled data)
 - **Clustering (unsupervised ML)** : decides what things to put in similar group whose members are similar in some way (unlabelled data) with classes unknown

How does NLP fits in?

- When the **datasets** are in the form of **texts** (e.g: emails, tweets, n-grams, named entities)
- Text and language are things that we as humans understand really well, but computers understand less well.
- We have to **transform a language into things that computers can understand** – which, in the general case, would be numbers.
- Attempt to **make the computer understand the gist of the context** of the given texts
 - understanding & applying the patterns to future things

Types of machine learning

- Supervised learning

- Machine **learning** task of inferring a function from labeled data (past experience) consisting of a set of training **examples**.
- Each example is a pair of an input object (ie: a vector) and a desired output value (target) **used to predict values of class attribute**

- Unsupervised learning

- Machine **learning** task of inferring a function to describe hidden structure from "unlabeled" data (a class, label or category is not included in the observations).

- Semi-supervised learning

- A class of **supervised learning** tasks that also make use of unlabeled data for training – a small amount of labeled data with a large amount of unlabeled data

Supervised vs Unsupervised Learning

Supervised	Unsupervised
<ul style="list-style-type: none">• known number of classes• based on a training set• used to classify future observations	<ul style="list-style-type: none">• unknown number of classes• no prior knowledge• used to understand (explore) data

Classification vs Clustering

Criteria	Classification	Clustering
Prior Knowledge of classes	Yes	No
Use case	Classify new sample into known classes	Suggest groups based on patterns in data
Algorithms	Decision Trees, Bayesian classifiers	K-means, Expectation Maximization
Data Needs	Labeled samples from a set of classes	Unlabeled samples

Examples : Supervised Learning Problem

- An emergency room in a hospital measures 17 variables (e.g., blood pressure, age, etc) of newly admitted patients.
- **A decision is needed:** whether to put a new patient in an intensive-care unit.
- Due to the high cost of ICU, those patients who may survive less than a month are given higher priority.
- **Problem:** to predict **high-risk patients** and discriminate them from **low-risk patients**.

Examples : Unsupervised Learning Problem

- Business case:
 - Inventory categorization:
 - Group inventory by sales activity
 - Group inventory by manufacturing metrics

Python Scientific & Machine Learning Packages

- Numpy (“Num Pie”)
- Scipy (“Sigh Pie”)
- NLTK
- Scikit-learn (“Sigh Kit-Learn”)

Numpy & Scipy

- Numpy (“Num Pie”)
 - NumPy is the fundamental package for scientific computing with Python. It contains among other things:
 - a powerful N-dimensional array object
 - sophisticated (broadcasting) functions
 - tools for integrating C/C++ and Fortran code
 - useful linear algebra, Fourier transform, and random number capabilities
- Scipy (“Sigh Pie”)
 - Open-source software for mathematics, science, and engineering.
 - The SciPy library depends on NumPy, which provides convenient and fast N-dimensional array manipulation.
 - The SciPy library is built to work with NumPy arrays, and provides many user-friendly and efficient numerical routines such as routines for numerical integration and optimization.




Installing Numpy & Scipy

- Make sure you are currently at your Python directory **where 'pip' is located** before installing the packages
- Use 'pip' for Numpy (all OS)

```
C:\Users\RAGS 12-006-0006\AppData\Local\Programs\Python\Python35-32\Lib\site-packages>pip install numpy
```

- Use 'pip' for Scipy (Mac & Linux)
 - > pip install scipy
- Get the source file (*.zip) from the following link:

- <https://github.com/scipy/scipy/releases>
- Download
scipy-0.19.0.zip
- Extract the zip file to the directory
site-packages in
Python3.X

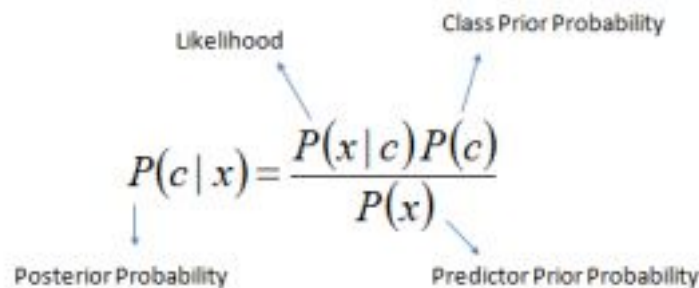
Downloads	
 Changelog	312 KB
 README	55.6 KB
 scipy-0.19.0.tar.gz	13.3 MB
 scipy-0.19.0.tar.xz	8.35 MB
 scipy-0.19.0.zip	14.6 MB
 Source code (zip)	
 Source code (tar.gz)	

Text Classification

- Goals:
 - i. How to identify certain features of language data that are salient(distinguished) for classification?
 - ii. How to construct a machine learning model to perform language processing tasks automatically?
 - iii. What can we learn about a particular language from these models?

Naïve Bayes Classification

- Based on Bayes theorem
- Assumption of independence among predictors.
- A Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature
- Perform well in case of categorical input variables compared to numerical variable(s).



The diagram shows the equation $P(c|x) = \frac{P(x|c)P(c)}{P(x)}$ with arrows pointing from labels to the terms in the equation. 'Likelihood' points to $P(x|c)$, 'Class Prior Probability' points to $P(c)$, 'Posterior Probability' points to $P(c|x)$, and 'Predictor Prior Probability' points to $P(x)$.

$$P(c|x) = \frac{P(x|c)P(c)}{P(x)}$$

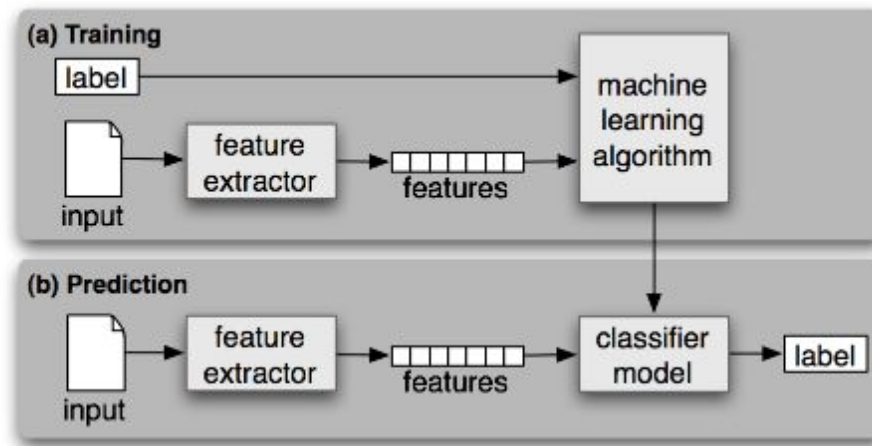
Labels in the diagram:

- Likelihood (points to $P(x|c)$)
- Class Prior Probability (points to $P(c)$)
- Posterior Probability (points to $P(c|x)$)
- Predictor Prior Probability (points to $P(x)$)

$$P(c|X) = P(x_1|c) \times P(x_2|c) \times \dots \times P(x_n|c) \times P(c)$$

Naïve Bayes Classification

- Framework used by supervised classification :



- During training, a feature extractor is used to convert each input value to a feature set. These feature sets, which capture the basic information about each input that should be used to classify
- Pairs of feature sets and labels are fed into the machine learning algorithm to generate a model.
- During prediction, the same feature extractor is used to convert unseen inputs to feature sets. These feature sets are then fed into the model, which generates predicted labels.

Identifying Features in a Dataset

- A feature is basically some kind of transformation of our input into a **numeric feature**, or just **a number that our machine can learn from**.
- We want to get a set of numbers, or a feature as we call it, that represents our dataset
- An example of a *feature* in text in Part of Speech (POS) category
- When more features are added, a model becomes slower to train because it takes more time to learn these patterns

Naïve Bayes Text Classification with NLTK

- Problem 1 : Classifying gender based on German names
 - Data Preparation
 - Data Pre-processing
 - Identifying features
 - Deciding what **features** of the input are relevant, and how to **encode** those features.
 - Male and female names have some distinctive characteristics. Names ending in *a*, *e* and *i* are likely to be female, names ending in *k*, *o*, *r*, *s* and *t* are likely to be male

Defining features

- We start by looking at the final letter of a given name. The following **feature extractor** function builds a **dictionary** containing relevant information about a given name

```
>>> def gender_feature(word):  
    return {'last_letter': word[-1]}  
  
>>> gender_feature('Shrek')  
{ 'last_letter': 'k' }
```

- After defining a feature extractor, we prepare a list of examples and corresponding class labels.

```
>>> from nltk.corpus import names  
>>> labeled_names = [(name, 'male') for name in names.words('male.txt')] +  
... [(name, 'female') for name in names.words('female.txt')]  
>>> import random  
>>> random.shuffle(labeled_names)
```

Data Preparation

- Use the feature extractor to process the *names* data, and divide the resulting list of feature sets into training and testing set
- Training set is used to train new “Naïve Bayes” classifier”

```
>>> featuresets = [(gender_features(n), gender) for (n, gender) in labeled_names]
>>> train_set, test_set = featuresets[500:], featuresets[:500]
>>> classifier = nltk.NaiveBayesClassifier.train(train_set)
```

Testing the Data

- Test the classifier with some names that did not appear the training data (e.g: names of character from “The Matrix” movie which are names in year 2199 or ypur own names which are not Germans)

```
>>> classifier.classify(gender_features('Neo'))  
'male'  
>>> classifier.classify(gender_features('Trinity'))  
'female'
```

Evaluating Results

- Evaluate classifier on larger test set(accuracy)

```
>>> print(nltk.classify.accuracy(classifier, test_set))  
0.77
```

- Examine the classifier to determine which features it found most effective/informative for distinguishing the names' genders

```
>>> classifier.show_most_informative_features(5)  
Most Informative Features  
      last_letter = 'a'           female : male   =   33.2 : 1.0  
      last_letter = 'k'           male  : female =   32.6 : 1.0  
      last_letter = 'p'           male  : female =   19.7 : 1.0  
      last_letter = 'v'           male  : female =   18.6 : 1.0  
      last_letter = 'f'           male  : female =   17.3 : 1.0
```

- Names in the training set that end in "a" are female 33 times more often than they are male, but names that end in "k" are male 32 times more often than they are female.
- These ratios are known as **likelihood ratios**, and can be useful for comparing different feature-outcome relationships

Classifying names other than German

- Collect/find a list of 100 names (50 males & 50 females) that you wish to classify other than German (e.g, Korean, Malay, Arab, Italian, Indonesian names, etc..)
- Save your list of names into 2 files using filenames **other than male.txt and female.txt** (you may name it male1.txt or female1.txt to avoid replacing the existing files for German names in nltk folder). Place the files into the **folder containing your nltk_data**:
>> C:/.../users/username/appdata/roaming/nltk_data/corpora/names
- Define a function to extract the features for the classifier to contain 1 or more beginning/ending letters of the names
- Repeat the steps in Slide 17 – 20