EECS 738 – Machine Learning

LAB 2

Naïve Bayes and KNN

Announcements

- You can submit your code both in .ipynb and .py formats
- Both formats are provided in the lab
- A training and a testing dataset are provided to complete the code
- The code is going to be graded based on the provided train/test datasets **AND** a private dataset
- For EECS 738: Please try to use LaTeX or some other text editor, if you are scanning your homework submit one original copy to me

Lab Overview

Please note that the following slides are very BASIC definitions and concepts to help you do the lab. They are included here only for a quick review and for reference.

Lab Overview

- In this lab you are asked to complete the Naïve Bayes and K-Nearest Neighbors codes
- Please review these two algorithms and make sure you know what the attribute value, class value, target, labels, etc. are
- Make sure you understand and implement each step of the algorithms according to the instructions mentioned in the code
- Refer to the book and the links provided for each concept for more details and examples

Naïve Bayes Review

- Bayes theorem provides a way of calculating the posterior probability, P(c|x), from P(c), P(x), and P(x|c).
- Naive Bayes classifier assume that the effect of the value of a predictor (x) on a given class (c) is independent of the values of other predictors.
- This assumption is called class conditional independence.

Naïve Bayes Review

Likelihood

Class Prior Probability

$$P(c \mid x) = \frac{P(x \mid c)\,P(c)}{P(x)}$$

Posterior Probability

Predictor Prior Probability

$$P(c \mid \mathrm{X}) = P(x_1 \mid c) \times P(x_2 \mid c) \times \cdots \times P(x_n \mid c) \times P(c)$$

Naïve Bayes Review

- $P(c|x)$ is the posterior probability of ***class (target)*** given ***predictor (attribute).***
- $P(c)$ is the prior probability of class.
- $P(x|c)$ is the likelihood which is the probability of predictor given class.
- $P(x)$ is the prior probability of *predictor*.

- The posterior probability can be calculated by first, constructing a frequency table for each attribute against the target.
- Then, transforming the frequency tables to likelihood tables
- Finally use the Naive Bayesian equation to calculate the posterior probability for each class.
- The class with the highest posterior probability is the outcome of prediction.

http://www.saedsayad.com/naive_bayesian.htm

Naïve Bayes Example

Frequency Table		Play Golf	
		Yes	No
Outlook	Sunny	3	2
	Overcast	4	0
	Rainy	2	3

$$P(x \mid c) = P(Sunny \mid Yes) = 3/9 = 0.33$$

Likelihood Table		Play Golf		
		Yes	No	
Outlook	Sunny	3/9	2/5	5/14
	Overcast	4/9	0/5	4/14
	Rainy	2/9	3/5	5/14
		9/14	5/14	

$$P(x) = P(Sunny) = 5/14 = 0.36$$

$$P(c) = P(Yes) = 9/14 = 0.64$$

Posterior Probability: $P(c \mid x) = P(Yes \mid Sunny) = 0.33 \times 0.64 \div 0.36 = 0.60$

http://www.saedsayad.com/naive_bayesian.htm

Naïve Bayes Example

- P(sneezing,builder|flu) = P(sneezing|flu)P(builder|flu)

$$P(\text{flu} \mid \text{sneezing,builder}) = \frac{P(\text{flu})P(\text{sneezing,builder} \mid \text{flu})}{P(\text{sneezing,builder})}$$

- General form:

$$P(C|A_1, A_2 ... A_n) = \frac{(\prod_{i=1}^{n} P(A_i \mid C))\,P(C)}{P(A_1, A_2 ... A_n)}$$

http://users.sussex.ac.uk/~christ/crs/ml/lec02b.html

Normal or Gaussian Distribution Formula

- We can use the distribution of the numerical variable to have a good guess of the frequency
- For example:

one common practice is to assume normal distributions for numerical variables

- The probability density function for the normal distribution is defined by two parameters (mean and standard deviation).

Gaussian Distribution Formula

		Humidity	*Mean*	*StDev*
Play Golf	yes	86 96 80 65 70 80 70 90 75	79.1	10.2
	no	85 90 70 95 91	86.2	9.7

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i$$ Mean

$$\sigma = \left[\frac{1}{n-1}\sum_{i=1}^{n}(x_i-\mu)^2\right]^{0.5}$$ Standard deviation

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$ Normal distribution

$$P(\text{humidity} = 74 \mid \text{play} = \text{yes}) = \frac{1}{\sqrt{2\pi}(10.2)} e^{-\frac{(74-79.1)^2}{2(10.2)^2}} = 0.0344$$

$$P(\text{humidity} = 74 \mid \text{play} = \text{no}) = \frac{1}{\sqrt{2\pi}(9.7)} e^{-\frac{(74-86.2)^2}{2(9.7)^2}} = 0.0187$$

http://www.saedsayad.com/naive_bayesian.htm

Naïve Bayes Summary

Once again, for each known class value:

- Calculate probabilities for each attribute, conditional on the class value.
- Use the product rule to obtain a joint conditional probability for the attributes.
- Use Bayes rule to derive conditional probabilities for the class variable.

Once this has been done for all class values, output the class with the highest probability.

KNN Summary

researchgate: Pseudocode for KNN

k-Nearest Neighbor
Classify $(\mathbf{X}, \mathbf{Y}, x)$ // $\mathbf{X}$: training data, $\mathbf{Y}$: class labels of $\mathbf{X}$, x: unknown sample
for $i = 1$ **to** m **do**
  Compute distance $d(\mathbf{X}_i, x)$
end for
Compute set I containing indices for the k smallest distances $d(\mathbf{X}_i, x)$.
return majority label for $\{\mathbf{Y}_i$ where $i \in I\}$

Confusion Matrix

- Each column of the matrix represents the instances in a predicted class while each row represents the instances in an actual class (or vice versa)

		Predicted		
		Cat	**Dog**	**Rabbit**
Actual class	**Cat**	5	3	0
	Dog	2	3	1
	Rabbit	0	2	11

- Of the 8 actual cats, the system predicted that three were dogs, and of the six dogs, it predicted that one was a rabbit and two were cats.

- https://en.wikipedia.org/wiki/Confusion_matrix

NB/KNN Confusion Matrix

For any data set we used to test the model we can build a confusion matrix:

- Counts of examples with:
- class label ω_j that are classified with a label α_i

model

target		$\alpha = 1$	$\alpha = 0$
	$\omega = 1$	140	20
	$\omega = 0$	17	54

https://people.cs.pitt.edu/~milos/courses/cs2750-Spring04/lectures/class12.pdf

NB/KNN Confusion Matrix

For any data set we used to test the model we can build a confusion matrix:

model

		$\alpha = 1$	$\alpha = 0$
target	$\omega = 1$	140	20
	$\omega = 0$	17	54

agreement

Error: ?

Accuracy of NB/KNN Confusion Matrix

		PREDICTED CLASS	
		Class=Yes	Class=No
ACTUAL CLASS	Class=Yes	a (TP)	b (FN)
	Class=No	c (FP)	d (TN)

- **Most widely-used metric:**

$$\text{Accuracy} = \frac{a+d}{a+b+c+d} = \frac{TP+TN}{TP+TN+FP+FN}$$

http://wwwis.win.tue.nl/~tcalders/teaching/datamining09/slides/DM09-02-Classification.pdf

Confusion Matrix: A Python Suggestion

```
>>> a
array([foo, foo, foo, foo, bar, bar,
       bar, bar, foo, foo, foo], dtype=object)
>>> b
array([one, one, one, two, one, one,
       one, two, two, two, one], dtype=object)
>>> c
array([dull, dull, shiny, dull, dull, shiny,
       shiny, dull, shiny, shiny, shiny], dtype=object)
```

```
>>> crosstab(a, [b, c], rownames=['a'], colnames=['b', 'c'])
b     one          two
c     dull  shiny  dull  shiny
a
bar   1     2      1     0
foo   2     2      1     2
```

http://pandas.pydata.org/pandas-docs/stable/generated/pandas.crosstab.html

NB/KNN Handling the Dataset

- The data should be: ‘dataframe’ type
- Example:

```
In [33]: from sklearn import datasets
         import numpy as np
         iris = datasets.load_iris()
         df = pd.DataFrame(data=np.c_[iris['data'], iris['target']],
                            columns=iris['feature_names'] + ['target'])
         df.head()
```

Out[33]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
0	5.1	3.5	1.4	0.2	0.0
1	4.9	3.0	1.4	0.2	0.0
2	4.7	3.2	1.3	0.2	0.0
3	4.6	3.1	1.5	0.2	0.0
4	5.0	3.6	1.4	0.2	0.0