FUNDAMENTALS OF MACHINE LEARNING

TUTORIAL 9

CSCI3320

Project: What Makes People Happy

Announcement

- The final project has been released
  - **Group project:** two students as a group
  - **Deadline:** 23:59 on May 2nd
  - **Programming language:** Python 3
  - **Task:** predict whether a person is happy
  - **Discussion Board:** Piazza
    piazza.com/cuhk.edu.hk/spring2017/csci3320/home

Project Introduction

- Predict whether a person is happy
  - Data preprocessing
    - Feature Extraction
    - Data transformation
  - Classification
    - Train the classifiers
    - Make the predictions
  - Visualization

The more accurate your prediction results are, the higher score you will obtain

Outline

- Data preprocessing
  - Extract Raw Feature Vectors
  - Data Transformation
- Classifiers
  - Logistic regression
  - Naïve Bayes
  - SVM
  - Random forest (an extension of decision tree, next tutorial)
- Visualization (next tutorial)

Data Preprocessing

- Extract Raw Feature Vectors
- Data Transformation

Raw Data

people may not provide answers for every question -> nan

UserID	YOB	Gender	Income	HouseholdStatus	EducationLevel	Party	Q124742	...
1	1938	Male	nan	Married (w/kids)	nan	Independent	No	...
2	1985	Female	$25,001 -$50,000	Single (no kids)	Master's Degree	Democrat	nan	...
5	1963	Male	over $150,000	Married (w/kids)	nan	nan	No	...
...	...	...	...	...	...	...	...	...
9480	nan	Female	nan	nan	nan	Independent	nan	...
9503	1945	Male	$25,001 -$50,000	Married (w/kids)	High School Diploma	Democrat	nan	...

110 columns, from UserID to votes

122771	Do/did you get most of your K-12 education in public school, or private school?	Public,Private
123464	Do you currently have a job that pays minimum wage?	Yes,No
123621	Are you currently employed in a full-time job?	Yes,No
124122	Did your parents fight in front of you?	Yes,No
124742	Do you have to personally interact with anyone that you really dislike on a daily basis?	Yes,No

just a recommendation. you don't have to

Hint: read file with `read_csv()` in package `pandas`

Extract Raw Feature Vectors

- The YOB, UserID, Happy, votes are already discrete numerical values
- We need to map the remaining 106 attributes to numerical values (how?)

map non-numerical values to numerical values
just map into numerical value. not necessarily have any meaning.
eg. income: 分段. but up to this point, we still cannot fit this data frame into estimator directly

just an example

	UserID	YOB	Gender	Income	HouseholdStatus	EducationLevel	Party	...
0	1	1938.0	0	nan	3	nan	0	...
1	2	1985.0	1	1	0	5	1	...
2	5	1963.0	0	5	3	nan	nan	...
...								
4617	9480	nan	1	nan	nan	nan	0	...
4618	9503	1945.0	0	1	3	1	1	...

Hint: you can use the `map()` function of `DataFrame` object in `pandas`
You can define own mapping method.

Data Transform

Goal:

Change raw feature vectors into a representation that is more suitable for the downstream estimators

Feature Binarization (Discretization)

- **Trick:** thresholding numerical features to get boolean values (or discrete values).
  - E.g., we can threshold the attribute "`Income`".

    eg. income. set some thresholds: above 50000 HKD -> 1; below 50000 ->0
    Can set more than one thresholds
- Useful for downstream probabilistic estimators.
  - E.g., [sklearn.neural_network.BernoulliRBM]
- A common trick for text processing
  - To simplify the probabilistic reasoning

Encoding Categorical Features

- **Motivation:** features are given as categorical not continuous values.
  - E.g., a person could have features
    - `["Male", "Female"]`
    - `["Democrat", "Republican", "Independent"]`
  - These feature can be easily coded as integers
    - `["Male", "Female"]→[0, 1]`
    - `["Democrat", "Republican", "Independent"]→[0, 1, 2]`
    - The a sample instance with `["Male", "Democrat"]` can be written as `[0, 0]`
  - Such representation can not be used directly with scikit-learn estimators (why?)
    - The estimators expect **continuous, ordered** input

because above, 0, 1, 2 -> they don't have specific meanings, maybe misleading
originally, the three types do not contain any ordering relationship 沒有大小比較
now since you use 0, 1, 2 ->you manually lay some ordering relationship onto the types.

How well an estimator can perform depends on how consistent your hypothesis (assumption) is with the reality.
Eg. You assume the data is normally distributed, and if it is indeed, the estimator can achieve high accuracy

Encoding Categorical Features

- **Motivation:** features are given as categorical not continuous values.
- **One possible solution:**
  - **one-of-K or one-hot encoding:** transform each categorical feature with m possible values into m binary features, with only one active
    - Interface in scikit-learn: [OneHotEncoder]

can use this interface or can also implement by yourself

Imputation of Missing Values

- **Problem:** many real world datasets contain **missing values**, which are **incompatible** with scikit-learn estimators
  - E.g., Blanks, NaNs, or other placeholders
- **Strategies:**
  - Basic way: discard entire rows and columns which contain the missing values
    - Simple. However, it may lose valuable data
  - Better method: infer missing data from known part of the data, i.e., impute the missing values

Imputation of Missing Values

- **Better method:** infer missing data from known part of the data, i.e., impute the missing values
  - Replace missing data with some statistic values
    - mean, median, or most frequent
    - Interface in scikit-learn: **sklearn.preprocessing.Imputer**
  - Use interpolation method

    general method but may not be useful in our project
    - Random replacement, Lagrange's polynomial interpolation, Newton's interpolation
  - Use modeling method

    general method but may not be useful in our project
    - Regression, Naïve Bayes, decision tree

Project Tasks

- Implement a function to transform the raw data into a numerical matrix

```
def transform(filename):
    # your code here
    return {'data':data,'target':target}
```

- Implement a function to impute the missing value

```
def fill_missing(X, strategy, isClassified):
    # your code here
    return X_full
```

Other Preprocessing Methods

- Standardization
- Normalization
- Generating polynomial features
- Custom methods

Standardization

- **Motivation:** Many machine learning estimators assume the individual features have Gaussian distribution (*zero mean and unit variance*)
  - RBF kernel in SVM
  - L1 and l2 regularizers of linear models
- **Solution:**
  - Ignore the shape of the distribution
  - Center it by removing the mean value of each feature,
  - Scale it by dividing by their standard deviation

    if this sd is very small, it may cause problems. If it is 0.00000000001, the number itself is not precise, and the calculation is not precise.
- API in scikit-learn: sklearn.proprocessing.scale

Standardization – Scale in a Range

- **Motivation:**
  - Robustness to very small standard deviations of features
  - Preserve zero entries in sparse data

if originally there are many 0s in this sparse matrix, when you perform standardization,
you will minus mean values for each entry, then all 0s will become non-zero values.
the computation memory required becomes extremely large all of a sudden.

- **Solution:**
  - Scale features to lie between a give minimum and maximum value
    - Lie between 0 to 1: sklearn.preprocessing.MinMaxScaler
    - Lie between -1 to 1: sklearn.preprocessing.MaxAbsScaler
      - Divide by maximum value in each feature
      - For data already centered at zero, or **sparse data**

Standardization – Scale with Outliers

if there are no outliers, you don't need to scale with outliers

- **Motivation:** outliers can often influence the sample mean/variance in a negative way.
- Solution:
  - Use more robust estimates for the center and range of data
    - Mean → Median
    - Deviation → Quantile range
  - API in scikit-learn: sklearn.preprocessing.RobustScaler

Normalization

- Scale individual samples to have unit norm
  - L1 norm
  - L2 norm
  - ……
- Why useful?
  - Quadratic form such as dot-product
  - Kernel functions which quantify the similarity of any pair of samples

Generating polynomial features

- Motivation: add complexity to the model

  increase complexity of the model, may result in overfitting (disadvantage)
- Solution:
  - Consider non-linear features
    - Use polynomial features $x \rightarrow x^2, \cdots, x^n$
  - API in scikit-learn:
    sklearn.preprocessing.PolynomialFeatures

Summary: the need for transformation depends on the model you are using.

Classification

- After data processing, you can feed the obtained data into the classifiers
  - Logistic Regression a linear model
  - Naïve Bayes
  - Support Vector Machine (SVM)
  - Random Forest (next tutorial)
- Make predictions with the trained classifiers

Logistic Regression

- A linear model for classification
- Object function when use L2 penalty

$$\min_{w,c} \frac{1}{2} w^T w + C \sum_{i=1}^{n} \log(\exp(-y_i(X_i^T w + c)) + 1).$$

- C is a parameter, you can find a "best" C via cross validation
- **Tasks:**
  - Train a logistic regression classifier in scikit-learn
  - Implement your own logistic regression classifier

Naïve Bayes

- **Assumption:** independence between every pair of features.
  - Naïve Bayes theorem: given a class variable y and a dependent feature vector $\boldsymbol{x} = (x_1, \cdots, x_n)$, we have

$$P(y|x_1, \cdots, x_n) = \frac{P(y)P(x_1, \cdots, x_n|y)}{P(x_1, \cdots, x_n)}$$

  - Good at text classification and spam filtering
- **Tasks:**
  - Choose one of the three Naïve Bayes classifiers in scikit-learn: GaussianNB, MultinomialNB, BernoulliNB and train the classifier
  - Implement your own Naïve Bayes classifier

Support Vector Machine

- Support vector machines (SVMs) are a set of supervised learning methods
  - classification,
  - regression
  - outliers detection.
- Advantages:
  - Effective in high dimensional spaces.
  - Memory efficient: support vectors.
  - Versatile: different Kernel functions:
    - Linear kernel
    - Polynomial kernel
    - Gaussian (RBF) kernel
    - Self-defined kernel functions
- **Tasks:**
  - Train a SVM classifier

Image Preprocessing

- A fixed size for each image.
- Keep only the grey level for all pixels
- Normalize the contrast of your images
- Try to work on a gradient map
- …..

Search the Internet for image preprocessing tutorials

References

- Preprocess:
  - **Data preprocess in scikit-learn:** http://scikit-learn.org/stable/modules/preprocessing.html
  - Kotsiantis, S. B., D. Kanellopoulos, and P. E. Pintelas. "*Data preprocessing for supervised leaning.*" *International Journal of Computer Science* (2006).
- Classifiers:
  - **Logistic regression**: http://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  - **Naïve Bayes**: http://scikit-learn.org/stable/modules/naive_bayes.html
  - **SVM**: http://scikit-learn.org/stable/modules/svm.html#svm-classification
  - **Random forest**: http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html