ECE5984 – Applications of Machine Learning
Lecture 7 – Missing Values and Imputation

Creed Jones, PhD

Course Updates

- Quiz 2 on this Thursday February 10
  - Covers lectures 4-7
- At the end of the semester, I will replace your lowest quiz grade with your next lowest grade
- HW1 is due <u>tonight</u> at 11:59 PM
- Project Teams have been designated
  - Look under "People" in Canvas
  - Project 1 info next week

Quiz timeframe – you have two choices

- Option 1: 12 noon to 6 PM Eastern time
- Option 2: 9 PM to 3 AM (next day) Eastern time

- If you don't make a choice, you will be given Option 1
- <u>If you prefer Option 2, please send me an email by noon on Thursday!!!!</u>
  – Don't forget to put "5984" in the email subject line

- Your choice will remain for the rest of the semester (6 more quizzes)

Today's Objectives

Strategies for missing values

- Impute zeroes
- Impute column means
- Impute column medians
- Impute kNN weighted average
- Iterative imputation (experimental)
- Stratified imputation

- Some code

Even in fully valid data, we can have data problems to address; the big three are – missing values, outliers and irregular cardinality

- Missing values may represent unreported or irrelevant fields
  - A missing value in the income field cannot be assumed to mean INCOME=0

Consider the possible results when we inquire about a person's income:

- A given dollar amount
- Zero (the person truly has no income)
- Irrelevant for this example (the person is an infant, perhaps)
- Unknown, didn't ask (will show up as missing)
- Unknown, asked but they didn't answer (will show up as missing)
- Impossible values: $42 trillion or -$30K, for example (sign of an error)

What do we do?

Let’s consider what we might do with missing values in each column

Name	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	Target
39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	40	United-States	<=50K
50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	13	United-States	<=50K
38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0	40	United-States	<=50K

- Name – an ID field. If missing, perhaps generate a new one?
- Target – if missing, we can’t model with it, so discard the row.
- marital-status, education, workclass, occupation, race, sex, native-country – missing probably means “unknown”; perhaps replace with “unknown” as a new category?
- fnlwgt, education-num – numerics for which missing probably means “unknown”; good categories to impute a value
  - Perhaps impute mean or median of those with same occupation? And sex?
- capital-gain, capital-loss and hours-per-week – numeric for which missing probably means zero. (investigate further)

Possible ways to deal with missing values in feature columns (predictive variables)

1. Remove that feature from the dataset
   - Only if well over half of the values are missing
2. *Impute* the mean value of the data present
   - OK, but overly simplistic
3. Impute the mean value for an appropriate subset, using a categorical feature
   - Imagine a database of employees; if a salary is missing, replace it with the mean for other employees with the same job title
   - This is called stratified imputation
4. Replace it with zero
   - Sometimes, this is the right answer
5. More sophisticated techniques
   - SMOTE

Different variable types and roles have different methods for replacement of missing values

Field Type	Variable Type	If Any Missing Values	Delete Column?	Imputation methods					
ID	Any	replace with unique value	no						
Feature (Predictor)	Numeric	impute	if the "vast majority" are missing	zero	population mean	population median	kNN, SMOTE, et al	stratified mean or median	
	Ordinal (ordinal categorical)	impute				population median	kNN	stratified mode or median	
	Interval	impute				population median	kNN	stratified mode or median	
	Categorical (unordered categorical)	impute				population mode	kNN	stratified mode	"UNK“ (new category)
	Binary	impute				population mode	kNN	stratified mode	
	Text	leave blank or replace with "UNK"							
Target	Any	delete row	no						

Imputing zeroes only make sense when metadata or semantics tell you that’s what to do

- Here I am applying it to all columns
- This is not a typical approach

```
def trylinearfit(rawpred, target, imputer):
    imputer.fit(rawpred)
    newpred = imputer.transform(rawpred)
    xtrain, xtest, ytrain, ytest = \
        skms.train_test_split(newpred, target, test_size=0.3)
    model = sklm.LinearRegression()
    regr = model.fit(xtrain, ytrain)
    print("R-sq=", regression.score(xtrain, ytrain), \
        “;  MSE=", skmt.mean_squared_error(ytest, regr.predict(xtest)))

zeroimputer = skim.SimpleImputer(missing_values=np.nan, \
        strategy='constant', fill_value=0)
trylinearfit(predictors.copy(), target, zeroimputer)
```

Imputing column means can be appropriate, depending on the distribution of the feature

- This approach is better than dropping rows and often the most achievable

```
def trylinearfit(rawpred, target, imputer):
    imputer.fit(rawpred)
    newpred = imputer.transform(rawpred)
    xtrain, xtest, ytrain, ytest = \
        skms.train_test_split(newpred, target, test_size=0.3)
    model = sklm.LinearRegression()
    regr = model.fit(xtrain, ytrain)
    print("R-sq=", regression.score(xtrain, ytrain), \
        “;  MSE=", skmt.mean_squared_error(ytest, regr.predict(xtest)))

meanimputer = skim.SimpleImputer(missing_values=np.nan, strategy='mean')
trylinearfit(predictors.copy(), target, meanimputer)
```

Imputing column *medians* is often looked at as superior because outliers in the column will have no effect on the value

- If no stratifying information is available, I will use this for columns in which imputing zero makes no sense

```
def trylinearfit(rawpred, target, imputer):
    imputer.fit(rawpred)
    newpred = imputer.transform(rawpred)
    xtrain, xtest, ytrain, ytest = \
        skms.train_test_split(newpred, target, test_size=0.3)
    model = sklm.LinearRegression()
    regr = model.fit(xtrain, ytrain)
    print("R-sq=", regression.score(xtrain, ytrain), \
        “;  MSE=", skmt.mean_squared_error(ytest, regr.predict(xtest)))

medianimputer = skim.SimpleImputer(missing_values=np.nan,
strategy='median')
trylinearfit(predictors.copy(), target, medianimputer)
```

sklearn has a function to impute the weighted average of the nearest neighbor – this is similar in spirit to the SMOTE method (but no randomness)

- In limited experimentation, I have not seen this method to be revolutionary
- But I want to try it some more

```
def trylinearfit(rawpred, target, imputer):
    imputer.fit(rawpred)
    newpred = imputer.transform(rawpred)
    xtrain, xtest, ytrain, ytest = \
        skms.train_test_split(newpred, target, test_size=0.3)
    model = sklm.LinearRegression()
    regr = model.fit(xtrain, ytrain)
    print("R-sq=", regression.score(xtrain, ytrain), \
        “;  MSE=", skmt.mean_squared_error(ytest, regr.predict(xtest)))

knnimputer = skim.KNNImputer(missing_values=np.nan, n_neighbors=5,
weights='distance')
trylinearfit(predictors.copy(), target, knnimputer)
```

sklearn has an experimental function to perform iterative imputation – it’s still a work in progress

- In limited experimentation, I have not seen this method to be revolutionary

- But I want to try it some more

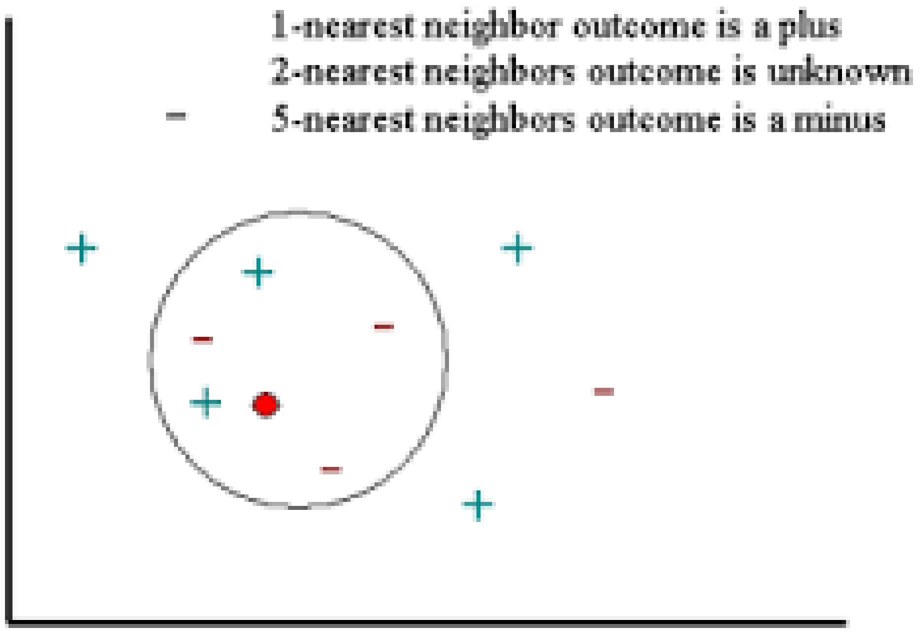
```
def trylinearfit(rawpred, target, imputer):
    imputer.fit(rawpred)
    newpred = imputer.transform(rawpred)
    xtrain, xtest, ytrain, ytest = \
        skms.train_test_split(newpred, target, test_size=0.3)
    model = sklm.LinearRegression()
    regr = model.fit(xtrain, ytrain)
    print("R-sq=", regression.score(xtrain, ytrain), \
        “;  MSE=", skmt.mean_squared_error(ytest, regr.predict(xtest)))

iterativeimputer = skim.IterativeImputer(missing_values=np.nan)
trylinearfit(predictors.copy(), target, iterativeimputer)
```

Stratified imputation (in this case on the year) is generally the most promising but is not built-in to sklearn (at least as far as I can tell)

```
yearcol = rawpredictors[:,0]
newpred = np.zeros((0,0))
newtarg = np.zeros((0,0))
for year in distinctyears:
    index = np.where(yearcol == year)
    thispred = predictors[index]
    if (thispred.shape[0] == 0):
        break
    medimputer = skim.SimpleImputer(missing_values=np.nan, strategy='median')
    medimputer.fit(thispred)
    newthispred = medimputer.transform(thispred)
    if (len(newpred) == 0):
        newpred = newthispred.copy()
        newtarg = target[index]
    else:
        newpred = np.append(newpred, newthispred, axis=0)
        newtarg = np.append(newtarg, target[index], axis=0)
```

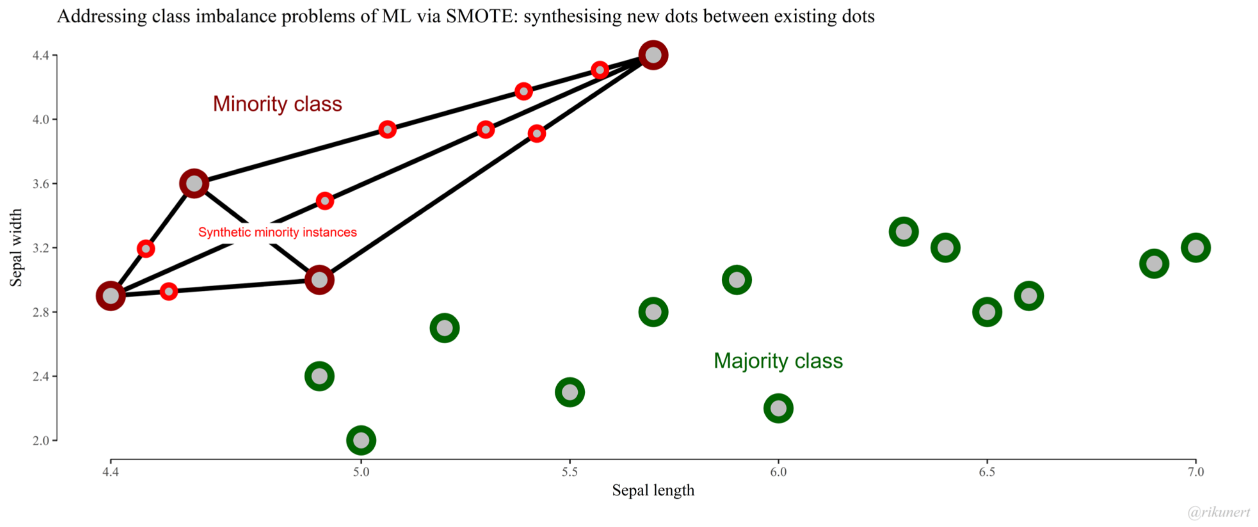
kNN Imputation replaces a missing value with the value of the nearest neighbor(s) as determined by the features present



- Imagine a dataset with predictors X, Y and Z
- If X is missing for a given sample, determine the closest sample in the Y-Z plane for which there is an X value
  - Impute that X value for the missing
- This is Nearest-Neighbor
- For kNN, find the k-nearest neighbors having an X value and aggregate their X values to impute
  - Mean, median, etc.

SMOTE: Synthetic Minority Over-sampling Technique replaces missing values with a value randomly distributed between the values of like samples

- SMOTE determines the nearest neighbors in modeling space to an instance with a missing value (in the non-missing dimensions)
- It then replaces the missing value with a randomly weighted sum of the existing values of the neighbors for that attribute
- SMOTE is also used to create synthetic samples to correct class imbalance
- https://www.jair.org/index.php/jair/article/download/10302/24590



Journal of Artificial Intelligence Research 16 (2002) 321–357 Submitted 09/01; published 06/02

VIRGINIA TECH.

BRADLEY DEPARTMENT OF ELECTRICAL & COMPUTER ENGINEERING

SMOTE: Synthetic Minority Over-sampling Technique

Nitesh V. Chawla CHAWLA@CSEE.USF.EDU
Department of Computer Science and Engineering, ENB 118
University of South Florida
4202 E. Fowler Ave.
Tampa, FL 33620-5399, USA

Kevin W. Bowyer KWB@CSE.ND.EDU
Department of Computer Science and Engineering
384 Fitzpatrick Hall
University of Notre Dame
Notre Dame, IN 46556, USA

Lawrence O. Hall HALL@CSEE.USF.EDU
Department of Computer Science and Engineering, ENB 118
University of South Florida
4202 E. Fowler Ave.
Tampa, FL 33620-5399, USA

W. Philip Kegelmeyer WPK@CALIFORNIA.SANDIA.GOV
Sandia National Laboratories
Biosystems Research Department, P.O. Box 969, MS 9951
Livermore, CA, 94551-0969, USA

Abstract

An approach to the construction of classifiers from imbalanced datasets is described. A dataset is imbalanced if the classification categories are not approximately equally represented. Often real-world data sets are predominately composed of "normal" examples with only a small percentage of "abnormal" or "interesting" examples. It is also the case that the cost of misclassifying an abnormal (interesting) example as a normal example is often much higher than the cost of the reverse error. Under-sampling of the majority (normal) class has been proposed as a good means of increasing the sensitivity of a classifier to the minority class. This paper shows that a combination of our method of over-sampling the minority (abnormal) class and under-sampling the majority (normal) class can achieve better classifier performance (in ROC space) than only under-sampling the majority class.

This paper is on Canvas

Take a look at it

There are a couple of good implementations of SMOTE in the freely available ML libraries for Python

- SMOTE for Imbalanced Classification with Python, by Jason Brownlee
- Nice tutorial
- https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/

- A package called SMOTE is available from David Sanchez
- https://github.com/chupati/smote

I experimented with several of these methods, using linear regression on the BaseballSalariesShort dataset

```
Raw File C:/Data/Baseball/BattingSalariesShort.xlsx is of size (2000, 24)
File C:/Data/Baseball/BattingSalariesShort.xlsx is of size (1244, 24)
Method=Impute zero, training set R-sq= 0.12968,  test set MSE=1.140704e+13
Method=Impute mean, training set R-sq= 0.13169,  test set MSE=1.136093e+13
Method=Impute median, training set R-sq= 0.13048,  test set MSE=1.137889e+13
Method=KNN imputation, training set R-sq= 0.13779,  test set MSE=1.137482e+13
Method=Iterative imputation, training set R-sq= 0.13843,  test set MSE=1.139793e+13
Method=Stratified imputation, training set R-sq= 0.14192,  test set MSE=1.162313e+13
```

Note: results may vary due to the randomness of splitting the datasets

LET'S LOOK AT SOME CODE

Today’s Objectives

Strategies for missing values

- Impute zeroes
- Impute column means
- Impute column medians
- Impute kNN weighted average
- Iterative imputation (experimental)
- Stratified imputation

- Some code