Machine Learning

Ensemble Learning Methods

CE 5331 Machine Learning for Civil Engineers

Venki Uddameri, Ph.D. , P.E.

Recap

- What is Machine Learning
- How is it useful for Civil Engineers
- Overview of Machine Learning Methods
- Linear Regression
  - Bivariate
  - Regression interpretation
  - Multivariate
- Logistic Regression
  - Maximum likelihood estimation
  - Regularization (introduction)
- Naïve Bayesian Classifier
  - What is it
  - What makes it naïve
  - Bayes theorem
  - Prior, likelihood and posterior
- K-Nearest Neighbor
  - How does the algorithm work
  - Why is it a lazy learner
  - How to do regression and classification
- Introduction to Decision Trees
  - Fundamentals
  - Information Gain, Entropy and Gini Index
  - ID3 algorithm
  - Classification and Regression Trees (CART)
  - Multi-Adaptive Regression Splines (MARS)

Python – Introduction
Python – Functions
Python - Pandas
Python – np, scipy, statsmodels
Python – Scikit learn – linear, metrics
Python – Matplotlib, seaborn
Python – Mixed_Naive_Bayes
Python – scikit learn neighbors module

R – Classification and Regression Trees using rpart
R – Drawing trees using rpart.plot
R - Multiadaptive Regression Splines (MARS) using Earth Algorithm

Ensemble-based Methods

Ensemble-Based Methods

- So far we have built several individual models
  - Regression methods
  - Naïve Bayes
  - K Nearest Neighbors (KNN)
  - Tree-based methods
    - ID3; C4.5 and C5.0
    - CART
    - MARS
- Each model has its own strength and weakness

"All models are wrong some are useful"
George Box

Can a set of "wrong" models be combined to build a "better" model?

Ensemble Methods

- The idea of ensemble methods is to combine the results from multiple models to create a single prediction
  - Different models may predict well on different parts of data and not so well on other parts
  - A combined modeling scheme that incorporates multiple models can overcome each other's deficiencies
- Combined model will provide a 'better' prediction than any single model within the ensemble
  - This is the underlying premise of building ensemble models!!
- Building Ensemble models involves some important decisions
  - Which models to include within the ensemble?
  - How should the results from the ensemble methods be integrated?

Ensemble Methods - Advantages

- Improves overall performance
- Reduces predictive variances
  - This comes at increased bias
- Adds stability by reducing noise in predictions
- Minimizes sensitivity to starting parameters during the optimization process

The downside of ensemble modeling is increased computational costs

Model transparency is also affected by the use of ensemble methods

Ensemble modeling need not be restricted to using different models. We can also partition the dataset to create ensemble models

Ensemble Methods

- There are many ways we can develop ensemble methods
- Develop different models and combine them
  - A simple form of stacking
  - Use the same dataset
  - Use different datasets
- Resample the datasets and develop different models
  - Bootstrap aggregation (Bagging)
- Sequentially develop models by fitting new models to predict errors generated by the model at the previous step
  - Boosting
- Pass inputs from a sequence of models (lower level) to a higher-level function
  - General form of stacking

Simple Ensemble Methods – Voting Methods

- For classification Problems:
  - For each obtain probabilities corresponding to each class
    - Select the highest probability as the selected class
  - Repeat the above step for all selected models
  - The selected class is selected based on ‘Majority” rule
    - If a majority of the models select a particular class then that is the class predicted by the ensemble classifier
  - All models in the ensemble are assumed to have equal voting rights
    - Perhaps more votes to models which are better predictors?
- For regression problems:
  - An average value of all models
  - Perhaps a weighted average based on better predictors
    - The reciprocal of SSE (sum of square error) could be used for weighting

In simple ensemble models the number of models are arbitrarily fixed

The models are often not of the same type

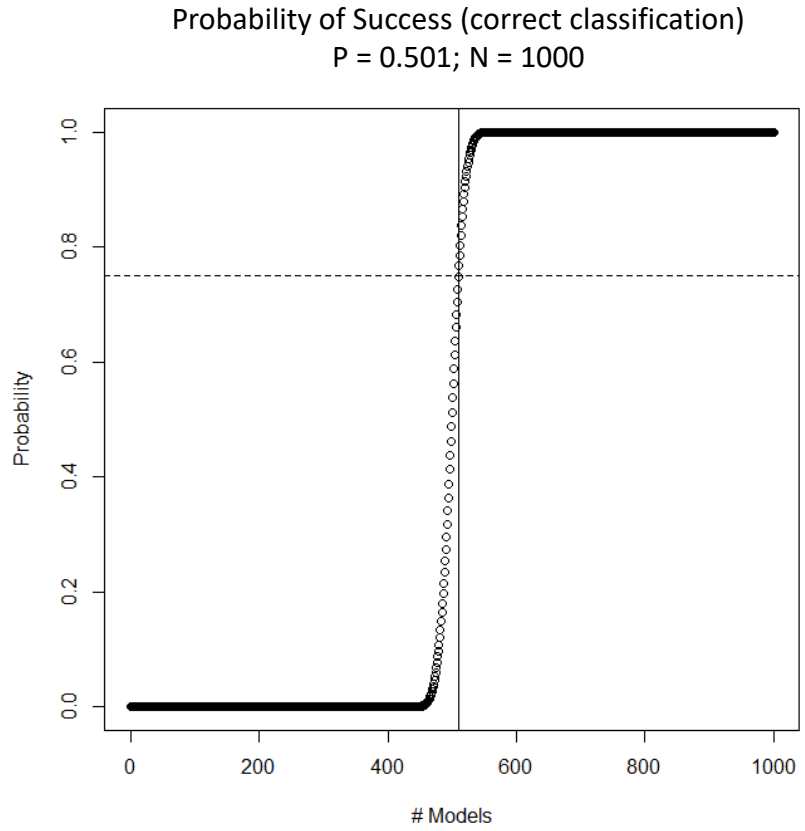
The models generally use the same input dataset

The models can use different inputs

A majority vote classifier is also called a hard voting classifier

Why Voting Classifiers Work

- Consider the probability of providing a correct classification of a model is 0.501
  - Slightly better than random guessing
- Let us take 1000 such models
- The cumulative probability of a set of models providing correct classification can be computed using Bernoulli distribution
  - Assumes models behave independently, correlations among models changes this estimate



Probability of 510 models providing correct classification is ~ 0.75

Illustrative Example

- Predicting damage to culverts in Texas
- Create an ensemble of Logistic Regression, CART and KNN classifier to predict satisfactory and Unsatisfactory states

Satisfactory

Unsatisfactory

Code	Meaning	Description
9	Excellent	As new
8	Very Good	No problems noted.
7	Good	Some minor problems.
6	Satisfactory	Structural elements show some minor deterioration.
5	Fair	All primary structural elements are sound but may have minor section loss, cracking, spalling or scour.
4	Poor	Advanced section loss, deterioration, spalling or scour.
3	Serious	Loss of section, deterioration, spalling or scour has seriously affected primary structural components. Local failures are possible. Fatigue cracks in steel or shear cracks in concrete may be present.
2	Critical	Advanced deterioration of primary structural elements. Fatigue cracks in steel or shear cracks in concrete may be present or scour may have removed substructure support. Unless closely monitored it may be necessary to close the bridge until corrective action is taken.
1	Imminent Failure	Major deterioration or section loss present in critical structural components or obvious vertical or horizontal movement affecting structure stability. Bridge is closed to traffic but with corrective action may put back in light service.
0	Failed	Out of service, beyond corrective action.

Source: United States Department of Transportation. Recording and Coding Guide for the Structure Inventory and Appraisal of the Nation's Bridges. Washington, D.C., 1995, page 38.

Illustrative Example

- VotingClassifier class is available in sklearn.ensemble module
  - https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.VotingClassifier.html
- Steps for running a voting classifier
  - Read data
  - Split into training and testing
  - Instantiate objects for various models of interest
    - They need to be in the sklearn form but not necessarily from sklearn library
    - They are the estimators to be used
  - Fit the voting classifier model
    - Hard Voting is based on classification performed the individual classifier
    - Soft Voting is based on predicted probabilities (Class that has highest average probability) – needs predict_proba() method
  - Perform accuracy tests (on the testing dataset)
  - Fit individual classifiers of the ensemble and compare accuracy metrics

sklearn.ensemble.VotingClassifier

class sklearn.ensemble.**VotingClassifier**(*estimators, voting='hard', weights=None, n_jobs=None, flatten_transform=True*) [source]

Soft Voting/Majority Rule classifier for unfitted estimators.

New in version 0.17.

Read more in the User Guide.

Parameters:

estimators : *list of (str, estimator) tuples*
Invoking the `fit` method on the `VotingClassifier` will fit clones of those original estimators that will be stored in the class attribute `self.estimators_`. An estimator can be set to `'drop'` using `set_params`.

Deprecated since version 0.22: Using `None` to drop an estimator is deprecated in 0.22 and support will be dropped in 0.24. Use the string `'drop'` instead.

voting : *str, {'hard', 'soft'} (default='hard')*
If 'hard', uses predicted class labels for majority rule voting. Else if 'soft', predicts the class label based on the argmax of the sums of the predicted probabilities, which is recommended for an ensemble of well-calibrated classifiers.

weights : *array-like, shape (n_classifiers,), optional (default=`None`)*
Sequence of weights (`float` or `int`) to weight the occurrences of predicted class labels (`hard` voting) or class probabilities before averaging (`soft` voting). Uses uniform weights if `None`.

n_jobs : *int or None, optional (default=None)*
The number of jobs to run in parallel for `fit`. `None` means 1 unless in a `joblib.parallel_backend` context. `-1` means using all processors. See Glossary for more details.

flatten_transform : *bool, optional (default=True)*
Affects shape of transform output only when voting='soft' If voting='soft' and flatten_transform=True, transform method returns matrix with shape (n_samples, n_classifiers * n_classes). If flatten_transform=False, it returns (n_classifiers, n_samples, n_classes).

Attributes:

estimators_ : *list of classifiers*
The collection of fitted sub-estimators as defined in `estimators` that are not 'drop'.

For Ensemble Regressors see - https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.VotingRegressor.html

Python Code

```
# Step 1: Load Libraries
import os
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression # Logistic regression
from sklearn.neighbors import KNeighborsClassifier #KNN classifier
import sklearn.ensemble as ens
from sklearn import metrics
from sklearn import tree
```

```
# Step 2: Change working directory
dir = 'D:\\Dropbox\\000CE5333Machine Learning\\Week7EnsembleLEarning\\Code'
os.chdir(dir)

# Read the dataset
a = pd.read_csv('TXculvertdata.csv') # read our dataset
features = ['SVCYR','ADT','Reconst','PTRUCK'] # INPUT DATA FEATURES
X = a[features] # DATAFRAME OF INPUT FEATURES
SVCYR2 = a['SVCYR']**2  # Add SVCYR square to the dataset
X['SVCYR2'] = SVCYR2  # CALCULATE THE SQUARE OF AGE
Y = a['Culvert_Damage'] # ADD IT TO THE INPUT FEATURE DATAFRAME
```

```
# Step 3: Split into training and testing data
X_train,X_test,y_train,y_test=train_test_split(X,Y,test_size=0.30,random_state=10)
```

```
# Step 4: instantiate the model object (using the default parameters)
lr = LogisticRegression(C=10**9)
knn = KNeighborsClassifier(n_neighbors=3)
nbc = mnb.MixedNB(categorical_features=[2])
```

```
# Step 5: Instantiate Voting Classifier, Fit it to training data and assess fit on testing data
lr = LogisticRegression(C=10**9)
knn = KNeighborsClassifier(n_neighbors=3)
nbc = tree.DecisionTreeClassifier(max_depth=3)
vote_clf = ens.VotingClassifier(estimators=[('lr',lr),('cart',nbc),('knn',knn)],
                                voting='hard')
vote_clf.fit(X_train,y_train)
```

```
# Step 6: Test Accuracy of individual classifiers within the ensemble on testing #
dataset
clf.lr = lr.fit(X_train,y_train)
y_pred = clf.lr.predict(X_test)
acc = metrics.accuracy_score(y_test,y_pred)
round(acc,4)

clf.knn = knn.fit(X_train,y_train)
y_pred = clf.knn.predict(X_test)
acc = metrics.accuracy_score(y_test,y_pred)
round(acc,4)

clf.nbc = nbc.fit(X_train,y_train)
y_pred = clf.nbc.predict(X_test)
acc = metrics.accuracy_score(y_test,y_pred)
round(acc,4)
```

Results

- The Voting classifier has better accuracy then all individual models considered within the ensemble
  - Does a much better job in estimating ‘unsatisfactory’ states

Model	Accuracy
Logistic	0.6517
CART	0.6564
KNN	0.6236
Ensemble	0.6691

Ensemble Model

	Predicted	
Obs	0	1
0	2758	747
1	1234	1247

LR Model

	Predicted	
Obs	0	1
0	2922	583
1	1502	979

Weighting Ensemble Models - Optimization

- The weighting of a set of models within an ensemble can be viewed as an optimization problem
- Let there be k models each having a known sum of squares error then
- An optimization model can be set up to obtain unknown weights

Weight of the i^{th} model

Square Error of j^{th} data point

Minimize

$$\sum_{i=1}^{K} w_i \sum_{j=1}^{N} (o_j - p_{i,j})^2$$

Subject to:

$$\sum_{i=1}^{K} w_i = 1$$

Weights imply relative importance of selected models

$$0 \leq w_i \leq 1$$

Non-Negativity

The optimization model is linear if only weights are estimated and nonlinear if both weights and model parameters are estimated

Voting Ensemble Methods

- Easy to understand
- Models of different types can be mixed
- Weighting of the models can be made objective
- The selection of models can be arbitrary
- The predictions depend upon the models selected within the ensemble
- The accuracy depends upon the number of models selected and the predictive accuracy of individual models
- Improvements over individual classifiers depends upon correlation between the classifiers
  - Similarity in methods, dataset, features

You Should Know

- What are ensemble models
  - Why are they advantageous
  - How can they be constructed
- Why ensemble methods are likely to perform better than individual models
- Simple Ensemble method or Voting method
- How to implement it in scikit learn
- What factors are likely to influence their performance
  - Relative to individual models