Feature Selection

Udacity Capstone Proposal

Rasmus Nisted Velling

Introduction

Feature selection is at the heart of applied statistics and data science. In commercial as well as academic settings one of the greatest problems the analyst/scientist faces are often that of separating the (good) features, that provide explanatory value, from the poor, which mainly act as noise. And in 'big data' one will often face problems where the number of raw available features vastly exceeds what regular computers can handle within a reasonable amount of time. Thus, feature selection should be of interest to most people, who deal with applied statistics and data science on a higher level.

At the same time in many realworld settings, especially in businesses, a few number of data scientist and analyst are expected to solve problems across many domains like marketing & sales, operations, logistics, infrastructure, H&R and many more, while at the same time being required to deliver results fairly quickly and of a decent standard. In these types of settings flexibility and genericity become important model aspects, as the analysis and data science teams will want to deliver scalable and easily maintainable solutions to the problems posed by the surrounding business.

The NIPS 2003 Feature Selection challenge mimicked the above-mentioned requirements into one competition, where participating teams were asked to come up with one model (or framework) which was then implemented across five datasets. The final challenge score & ranking was then calculated as the average of the Balanced Error Rate (see evaluation metrics) on each dataset, thus, requiring the teams to come up with flexible and generic models, that where somehow able to distinguish useable features from non-useable.

Capstone motivation

Inspired by a specific, high dimensional, sparse data problem at work (which I was not able to use as Capstone Project due to legal reason), I wanted to investigate flexible and broad methods for dealing with high dimensional, sparse classification problems. And while researching on this topic, I stumbled across the NIPS 2003 Challenge, which I in many ways feel reflects a lot of the problems I, as a Data Analyst, deal with in my day-to-day work.

In this Capstone Project I will explore model methods and frameworks inspired by the NIPS 2003 Feature Selection Challenge applied to the five datasets made available for the same challenge. A key component of the challenge was that the teams were scored on how well a single model would perform across all five datasets, thus requiring model frameworks to be flexible and generic, and I will keep this dogma throughout this capstone as well.

Winning entries of the Challenge are described in the book *Feature Extraction: Foundations and Applications* (Guyon et al, 2006) which came to life as a product of the best entries in the NIPS 2003 Challenge. Many of the winning entries are coded in Matlab or C, so this in this Capstone project I will focus on exploring how close one can come to the winning entries using readily available and popular tools in Python.

Problem Statement & Solution

Problem

How well do python-implementations (using readily available tools like sklearn) of “winning” models and other popular algorithms perform on the NIPS 2003 Challenge datasets?

Solution

I will implement a series of classification models in python inspired by the winning entries from the NIPS 2003 Feature Selection Challenge and described in the book *Feature Extraction: Foundations and Applications* (Guyon et al, 2006), as well as models using more recent and/or other algorithms.

I will apply the same models to all five datasets (as done in the original NIPS 2003 Challenge), to test the genericity of the proposed solutions.

Dataset and inputs

The five datasets can be downloaded at http://clopinet.com/isabelle/Projects/NIPS2003/#challenge or individually at the UCI Machine Learning Repository (for instance http://archive.ics.uci.edu/ml/datasets/madelon).

The five data set are:

- **ARCENE**
  Non sparse. Features: 10 000. N train: 100. N valid: 100
  Official description: ARCENE was obtained by merging three mass-spectrometry datasets to obtain enough training and test data for a benchmark. The original features indicate the abundance of proteins in human sera having a given mass value. Based on those features one must separate cancer patients from healthy patients. We added a number of distractor feature called 'probes' having no predictive power. The order of the features and patterns were randomized.

- **DEXTER**
  Sparse Integer. Features: 20 000. N train: 300. N valid: 300
  Official description: The original data were formatted by Thorsten Joachims in the “bag-of-words” representation. There were 9947 features (of which 2562 are always zeros for all the examples) representing frequencies of occurrence of word stems in text. The task is to learn which Reuters articles are about 'corporate acquisitions'. We added a number of distractor feature called 'probes' having no predictive power. The order of the features and patterns were randomized.

- **DOROTHEA**
  Sparse binary. Features: 100 000. N train: 800. N valid: 350
  Official description: DOROTHEA is a drug discovery dataset. Chemical compounds represented by structural molecular features must be classified as active (binding to thrombin) or inactive.

- **GISETTE**
  Non sparse. Features: 5000. N train: 6000. N valid: 1000
  Official description: GISETTE is a handwritten digit recognition problem. The problem is to separate the highly confusible digits '4' and '9'.

- **MADELON**
  Non sparse. Features: 500. N train: 2000. N valid: 600
  Official description: MADELON is an artificial dataset containing data points grouped in 32 clusters placed on the vertices of a five dimensional hypercube and randomly labeled +1 or -1. The five dimensions constitute 5 informative features. 15 linear combinations of those features were added to form a set of 20 (redundant) informative features. Based on those 20 features one must separate the examples into the 2 classes (corresponding to the +-1 labels). We added a number of distractor feature called 'probes' having no predictive power. The order of the features and patterns were randomized.

All datasets have integer inputs between 0 and 999 or as binary 0 or 1. And all datasets have had random variables with no explanatory power added. No columns names/labels have been given to either datasets, as the challenge is to extract explanatory power with out any preliminary knowledge of the data.

Binary outcome
It is worth noting that all target variables are encoded as +1 or -1.

Benchmark model & evaluation metrics

Evaluation metrics
The NIPS 2003 Feature Selection Challenge used as primary scoring metrics the *Balanced Error Rate*, defined as:

BER = 0.5*(**b**/(**a**+**b**) + **c**/(**c**+**d**))

Where a, b, c and d can be obtained from the classifier confusion matrix as follows:

		Prediction	
		Class -1	*Class +1*
Truth	*Class -1*	**a**	**b**
	Class +1	**C**	**d**

Benchmark models
I propose the following benchmark models

1. Super Naïve benchmark
   This model comes in two versions predicting either all 1’s or all negative 1’s. This model will in either case yield a BER of 0.5 (the ‘perfect’ model would yield a BER of 0.0, a model getting everything exactly wrong gives 1.0)
2. Simple PCA-based Logistic Regression
   PCA decomposition is a popular tool (with reason) for decomposing high dimensional data, and logistic regression is at the base of binary classification. I propose at benchmark-model where the input data is decomposed using PCA, and then a logisitic regression is applied on top of these components.

Example of benchmark PCA Logistic Regression function in Python

```
def benchmark_pca_logistic(data):
    # make PCA Logistic reg
    pca = PCA(n_components=5).fit(data['X_train'])
    Xpca_train = pca.transform(data['X_train'])
    Xpca_valid = pca.transform(data['X_valid'])

    # Basic Logistic regression
    clf = LogisticRegression(
        random_state=0, solver='sag', multi_class='ovr',
        max_iter=1000).fit(Xpca_train, data['y_train'])
    y_valid_hat = clf.predict(Xpca_valid)
    y_train_hat = clf.predict(Xpca_train)

    BER = src.ber(y=data['y_valid'].tolist(), y_hat=y_valid_hat.tolist())
    return BER
```

3. Challenge results
   All original entries, as well as post-challenge entries, can be found online as an archived website at: https://web.archive.org/web/20130502021704/http://www.nipsfsc.ecs.soton.ac.uk/results/
   Unfortunately, it is not possible to make new submission, as the original site has been taking down, and thus it is not possible for me to get a test-score for my models. Thus I will be comparing my validation scores, with validation scores from the Challenge website, although this is not the best benchmark, but it is feasible.

Example of a final ranking, including the PCA Logistic Regression shown above, and an out-of-the-box XGBOOST classifier (actual results)

BER by model and dataset

Model	ARCENE	DEXTER	DOROTHEA	GISETTE	MADELON	Total Score
NIPS Winner, Validation score	0.07	0.05	0.05	0.02	0.07	0.05
model_xgboost	0.29	0.10	0.30	0.03	0.23	0.19
benchmark_pca_logistic	0.30	0.32	0.40	0.15	0.41	0.31
benchmark_all_neg1	0.50	0.50	0.50	0.50	0.50	0.50
benchmark_all_pls1	0.50	0.50	0.50	0.50	0.50	0.50

Project outline & design

The project will go through the following steps:

- **Introduction, background and problem statement**
- **NIPS 2003: Theory & methods**
  Brief review of some of the best methods described in the book based on the NIPS 2003 Challenge: *Feature Extraction: Foundations and Applications* (Guyon et al, 2006), and my considerations as to how these can be implemented in python.
- **Data loading**
  Downloading the data from http://clopinet.com/isabelle/Projects/NIPS2003/ and loading the data as numpy arrays, including very basic data cleaning. The data comes fairly ready for processing.
- **Data exploration**
  Exploring datasets, features and target variables
- **Model pipeline and scoring**
  Build a framework for simple execution of multiple proposed models on multiples dataset as well

as gathering up relevant benchmark and scoring metrics for final comparison.
Algorithms intended to be used include (but not limited to):

a. PCA + Logistic regression
b. Decision Trees & Random Forests from sklearn
c. XGBoost
d. Keras deep networks
e. SVM from sklearn

The

- **Comparison with NIPS 2003 Challenge results**
  Comparing my proposed models' scores to those of the NIPS 2003 Challenge.
- **Concluding remarks**

All the datasets are readily available at the UCI Machine Learning Repository (https://archive.ics.uci.edu/ml), as well as on the challenge-committee's website (http://clopinet.com/isabelle/Projects/NIPS2003/). All the dataset contain a training set and a validation set with both input and target variables given. The datasets also contain a test set, where labels are withheld for future competition purposes. Thus I will optimizing for a cross-validation error on the trainingsets, and leaving the validation sets out untill the end. Therefor my reported errors will not compare directly to the rankings in the competition, but should still give an indication as to how well my proposed techniques perform.

My code is (still work in progress) available at: https://github.com/rasmusvelling/udacity_capstone