PROJECT REPORT

# ITERATIVE CLASSIFICATION ALGORITHM

by SAURABH KATKAR
CWID: A20320336
E-mail : skatkar@hawk.iit.edu

## 1. Abstract

Our project is an implementation of Iterative classification algorithm to classify text documents. The main motive behind using this type of algorithm is that it enable us to make appropriate predictions using the contents of the links and by taking advantage of the relations that exist between the links in order to classify data more accurately.. This report reviews text categorization and content based classification to learning algorithms. Then we propose an iterative approach using aggregates to test for relational information for text categorization. We test the text categorization with different data sets and support values. Accuracy is evaluated and efficiency of the algorithm is evaluated.

## 2. Introduction

Text categorization is a process that group text documents into one or more predefined categories. It has wide applications, such as email filtering, category classification for search engines and digital libraries. Classification of nodes or links in the network, discovery of links or nodes which are not yet observed or identification of essential nodes or links, are some of the research areas on networked data.
Basically there are two stages involved in text classification: Training stage and testing stage. In training stage, documents are preprocessed and are trained by a learning algorithm to generate the classifier. In testing stage, a validation of classifier is performed. There are many traditional learning algorithms to train the data, such as Decision trees, Naïve-Bayes (NB), Support Vector Machines (SVM), k-Nearest Neighbor (kNN), Neural Network (NNet), etc. We experiment using simple classification algorithm in an iterative fashion to improve classification accuracy by exploiting relational information in the data. The hypothesis underlying this approach is that if two objects are related, inferring something about one object can assist inferences about the other This forms the basis for our Iterative Classification Algorithm (ICA).

In this project, first of all, we evaluate the performance of content only classification, for different levels of content noise and on two datasets Citeseer and Cora, whose details are given below.. We show that when content and relations between links is present, using feature and node selection for iterative classification results in greater accuracy improvement.

## 3. Iterative Classification Algorithm

---

Iterative classification algorithm (ICA) is a successful approximate inference algorithm used for collective classification. Despite its simplicity, it performs sufficiently well or better than other collective classification algorithms, such as Gibbs Sampling .
To determine the label of a node, Iterative Classification Algorithm (ICA) assumes that all of the neighbors' attributes and labels of that node are already known. Then, it calculates the most likely label with a local classifier which uses node content and neighbors' labels. However, it is extremely rare to find a node with all of its neighbors labeled. So, ICA repeats the process iteratively until all of the label assignments become stabilized. One problem is all nodes do not have equal number of neighbors. This makes it hard to implement the local classifier which should take constant number of inputs. To overcome this difficulty, an aggregation operator such as count, mode or exists can be used. For example, count operator returns the number of occurrences of each label among the neighbors.

## 4. Advantage of ICA over Content Classification

---

Content based classification solves the classification problem using only the features (content) and labels of the given samples. In this type of classification the observed and unobserved samples are assumed to be drawn independently from an identical distribution. Connections or dependencies/relations between samples are not taken into consideration.
However, in addition to content, connectivity information is often available and connectivity can be an important factor in determining the node labels. For example, papers which are on a certain topic, usually cite or are cited by other papers on the same topic, people who are interested in a certain product have close friends who could be interested in the same product. Link based iterative classification takes into consideration the links between the objects in the given sample. This results in the improvement of the estimation performance of the classifiers. Attributes of objects and links together can be considered as the features of the nodes. However, when two linked samples are not yet classified, they require each other's labels to decide on their own label. This situation could get even more complicated when links create cycles in a vast network. ICA is thus proposed to overcome such problems.

## 5. Our Tools

---

### 5.1. The Python Ecosystem

With its mature scientific stack, Python is a growing contender in the landscape of text classificaiton. The scientific Python libraries used in this paper are:

- **NumPy**: Provides the ndarray data type to python, an efficient *n*-dimensional data representation for array-based numerical computation, similar to that used in Matlab (Van Der Walt et al., 2011). It handles efficient array persistence (input and output) and provides basic operations such as dot product. Most scientific Python libraries, including scikit-learn, use NumPy arrays as input and output data type.
- **SciPy**: Higher level mathematical functions that operate on ndarrays for a variety of domains including linear algebra, optimization and signal processing. SciPy is linked to compiled libraries to ensure high performances (BLAS, Arpack, and MKL for linear algebra and mathematical operations). Together, NumPy and SciPy provide a robust scientific environment for numerical computing and they are the elementary bricks that we use in all our algorithms.

### 5.2. Scikit-learn

Scikit-learn (Pedregosa et al., 2011) is a general purpose machine learning library written in Python. It provides efficient implementations of state-of-the-art algorithms, accessible to non-machine learning experts, and reusable across scientific disciplines and application fields. It also takes advantage of Python interactivity and modularity to supply fast and easy prototyping. There is a variety of other learning packages. For instance, in Python, PyBrain (Schaul et al., 2010) is best at neural networks and reinforcement learning approaches, but its models are fairly black box, and do not match our need to interpret the results. Beyond Python, Weka (Hall et al., 2009) is a rich machine learning framework written in Java, however, it is more oriented toward data mining.

Some higher level frameworks provides full pipeline to apply machine learning techniques to neuroimaging. PyMVPA (Hanke et al., 2009) is a Python packaging that does data preparation, loading and analysis, as well as result visualization. It performs multi-variate pattern analysis and can make use of external tools such as R, scikit-learn or Shogun (Sonnenburg et al., 2010). PRoNTo (Schrouff et al., 2013) is written in Matlab and can easily interface with SPM but does not propose many machine learning algorithms. Here, rather than full-blown neuroimaging analysis pipelines, we discuss lower-level patterns that break down how neuroimaging data is input to scikit-learn and processed with it. Indeed, the breadth of machine learning techniques in scikit-learn and the variety of possible applications are too wide to be fully exposed in a high-level interface. Note that a package like PyMVPA that can rely on scikit-learn for neuroimaging data analysis implements similar patterns behind its high-level interface.

### 5.3. Scikit-learn concepts

In *scikit-learn*, all objects and algorithms accept input data in the form of 2-dimensional arrays of size samples × features. This convention makes it generic and domain-independent. Scikit-learn objects share a uniform set of methods that depends on their purpose: *estimators* can fit models from data, *predictors* can make predictions on new data and *transformers* convert data from one representation to another.

- **Estimator**. The *estimator* interface, the core of the library, exposes a fit method for learning model parameters from training data. All supervised and unsupervised learning algorithms (e.g., for classification, regression or clustering) are available as objects implementing this interface. Machine learning tasks such as feature selection or dimensionality reduction are also provided as estimators.
- **Predictor**. A *predictor* is an estimator with a predict method that takes an input array X_test and makes predictions for each sample in it. We denote this input parameter "X_test" in order to emphasize that predict generalizes to new data. In the case of supervised learning estimators, this method typically returns the predicted labels or values computed from the estimated model.

When testing an estimator, one needs a reliable metric to evaluate its performance. Using the same data for training and testing is not acceptable because it leads to overly confident model performance, a phenomenon also known as *overfitting*. **Cross-validation** is a technique that allows one to reliably evaluate an estimator on a given dataset. It consists in iteratively fitting the estimator on a fraction of the data, called *training set*, and testing it on the left-out unseen data, called *test set*. Several strategies exists to partition the data. For example, $k$-fold cross-validation consists in dividing (randomly or not) the samples in $k$ subsets: each subset is then used once as testing set while the others $k - 1$ subsets are used to train the estimator. This is one of the simplest and most widely used cross-validation strategies. The parameter $k$ is commonly set to 5 or 10.
For a given model, the scores on the various test sets can be averaged to give a quantitative score to assess how good the model is. Maximizing this cross-validation score offers a principled way to set hyperparameters and allows to choose between different models. This procedure is known as *model selection*.

## 6. Datasets and Loading a Networked Dataset

---

We used the CiteSeer and CoRA scientific publication and web page datasets in the experiments. Both datasets are downloaded from the Statistical Relational Learning Group web site (http://linqs.cs.umd.edu/projects/projects/lbc/index.html) at the University of Maryland.

### 6.1 CiteSeer:

CiteSeer data set consists of information on 3312 scientific papers. Every paper in CiteSeer also cites or is cited by at least one other paper in the data set. There are 3703 unique words that are contained at least 10 times in these papers. We parse this dataset using python to determine the number of classes, links and graph that exists between different features in the dataset. There are 6 classes assigned to the papers according to their topics. Just as in the CoRA dataset, word, class and cites and cited by information are given in two

separate files. Total number of connections between the papers is 4732. There are on an average 2.77 links per paper.

**6.2 CoRA:**
CoRA data set consists of information on 2708 Machine Learning papers. Every paper in CoRA cites or is cited by at least one other paper in the data set. There are 1433 unique words that are contained at least 10 times in these papers. We parse this dataset using python to determine the number of classes, links and graph that exists between different features in the dataset. There are 7 classes assigned to the papers according to their topics. For each paper, whether or not it contains a specific word, which class it belongs to, which papers it cites and which papers it is cited by are known. Citation connections and paper features (class and included words) are contained in two seperate files. Total number of connections between the papers is 5429. There are on an average 4.01 links per paper.

## 7. Experimental Setup and Results

---

**7.1 Content Based Classification**
In this type of classification, we take the dataset and partition it into a training set and test set. The training data is then converted to a set of vectors of features. The statistical properties of the features of the training set is then used to build a model.
For the purpose of classification we use the following classifiers;

**Support Vector Machines**
These are supervised learning models with associated learning algorithms that analyze data and recognize patterns, used for classification and regression analysis. Given a set of training examples an SVM training algorithm builds a model that assigns new examples into one category or the other, making it a non-probabilistic binary linear classifier.

**Multinomial Naive Bayes Classifier**
A naive Bayes classifier is a simple probabilistic classifier based on applying Bayes' theorem with strong (naive) independence assumptions. With a multinomial event model, samples (feature vectors) represent the frequencies with which certain events have been generated by a multinomial. This is the event model typically used for document classification; the feature values are then term frequencies. The likelihood of observing a feature vector (histogram) *F* is given by

$$p(F|C) = \frac{(\sum_i F_i)!}{\prod_i F_i!} \prod_i {p_i}^{F_i}$$

**Logistic Regression**

Logistic Regression is a discriminative model that determines conditional class probabilities without modelling the marginal distribution of features. Logistic regression classifier can also be thought of as a one layer multilayer perceptron with its own special training algorithm.

**Decision Tree Classifier**

In decision tree classification, a decision tree can be used to visually and explicitly represent decisions and decision making. Decision Tree Classifier is a simple and widely used classification technique. It applies a straightforward idea to solve the classification problem. Decision Tree Classifier poses a series of carefully crafted questions about the attributes of the test record. Each time time it receive an answer, a follow-up question is asked until a conclusion about the class label of the record is reached.

Using these classifiers, we extract the features of the training data and ask the model to predict the most likely outcome based on the labels. The accuracy score of each of the model is determined and evaluated.

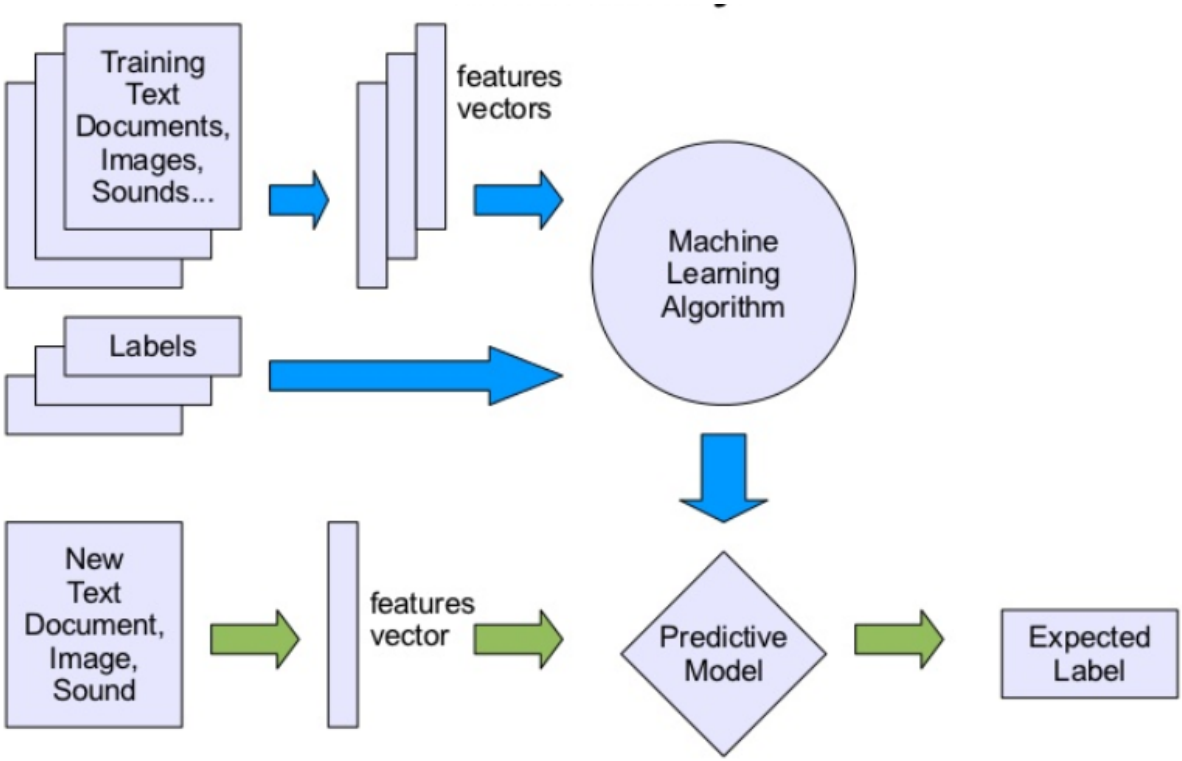


*fig: Content-Only text classification*

Table I.

| Classifier | Accuracy(Citeseer) | Accuracy(Cora) |
|---|---|---|
| Decision Tree Classifier | 74.9754661433 % | 93.1683899557 % |

| | | |
|---|---|---|
| Logistic Regression | 16.781157998 % | 30.2067946824 % |
| MultinomialNB | 50.6378802748 % | 71.3072378139 % |
| SVM | 56.0353287537 % | 88.8478581979 % |

*table1: Accuracy of Classification Prediction*

**7.2 Sampling**

Data sets are split into training and validation sets. We use k-fold cross-validation for creating these partitions. In k-fold cross-validation nodes are splited into random partitions k times. Out of these k partitions, k-1 partitions are used as training data, while the remaining one is left for validation. Validation is performed k times to ensure that each subsample is used for validation exactly once and also to have an errorbar on the validation accuracy. When the number of links per node is low, k-fold sampling generates nearly disconnected graphs. k has been chosen to be 5 in our experiments. This ensures that because of high number of folds, the generated subset is interconnected and balanced. Selected subset is used as the test data while remaining nodes compose the train subset. Results print out the mean of cross validated results.

Table II.

| Classifier | Citeseer Score | Cora Score | Citeseer Result | Cora Result |
|---|---|---|---|---|
| Decision Tree Classifier | 1.0 | 1.0 | 125.2242809 | 103.9353745 |
| Logistic Regression | 0.20833333 | 0.3017997231 | 16.93083555 | 5.985510128 |
| MultinomialNB | 0.703431373 | 0.7231195201 | 108.9002484 | 33.60935100 |

*table2: Results of 5-fold Cross Validation*

**7.3 Iterative Classification**

To determine the label of a node, Iterative Classification Algorithm (ICA) assumes that all of the neighbors' attributes and labels of that node are already known. Then, it calculates the most likely label with a local classifier which uses node content and neighbors' labels.

Iterative Classification Algorithm (ICA) is applied on the content and the relations that exist between the contents in order to predict the labels based on labels that are already known. This most likely label is then calculated with different local classifiers which use the node content and neighbours' labels. The different relational features can be represented using aggregation operators such as count, mode and exists.

The algorithm for ICA is given below. In pseudo code, OA represents observed attributes(content) of the samples. Y represents the unlabeled data and yi stands for temporary label assignment of sample Yi. g is the result of local classifier which shows the probability of getting label yi for sample Yi. O is the random ordering of nodes in every step of the iteration.

**Iterative Classification Algorithm (ICA)**
**for all** $Yi \in Y$ do
    Compute $g(y_i|v_{N(Yi);w})$ only using $OA(Yi|x_{N(Yi)})$
    $\forall y_i \in C$:
    **Set** $y_i = \arg\max_y g(y|v_{N(Yi)})$
**end for**
**repeat**
Generate Ordering O over nodes Y
**for all** $Yi \in O$ **do**
    Compute $g(y_i|v_{N(Yi);w})$ $\forall y_i \in C$:
    $y_i = \arg\max_y g(y|v_{N(Yi)})$
**end for**
**until** labels are stabilized

As shown above, the Iterative Classification Algorithm (ICA) starts with a bootstrapping to assign initial and temporary labels to all nodes by using only the content features of the nodes. Then, it starts iterating and updating labels according to the both relational and content features

## 8. Conclusion and Future Work

In this paper, we have analyzed the performance of text classification using scikit-learn to find accuracy of the prediction of different classifiers. We also performed 5 fold cross-validation on the data to determine how the result generalized into a statistical dataset. We implemented the Iterative Classification Algorithm for improving the accuracy obtained with only content based classification.
Our future works could include extending the iterative procedure for prediction of multiple object types by simply combining the results of multiple classifiers. Each classifier would make use of the dynamic attributes filled in through the efforts of the other classifiers. In this sense the classifiers would collaborate with each other to improve accuracies for both classification tasks.

## References


1. Jennifer Neville, David Jensen. *Iterative Classification in Relational Data.* Knowledge Discovery Laboratory. Department of Computer Science. University of Massachusetts.
2. Sofus A. Macskassy, Foster Provost. *A Simple Relational Classifier.* NYU Stern School of Business.
3. Qing Lu, Lise Getoor. *Link Based Classification.* Department of Computer Science. University of Maryland, College Park.
4. Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Gallagher, Tina Eliassi-Rad. *Collective Classification in Network Data.*
5. Baris Senliol, Atakan Aral, Zehra Cataltepe. *Feature Selection for Collective Classification*