Introduction to Statistical Computing in Scala - an Implementation of the K-Nearest Neighbors Classifier

Journal:	*IEEE Access*
Manuscript ID	Draft
Manuscript Type:	Original Manuscript
Date Submitted by the Author:	n/a
Complete List of Authors:	Tesileanu, Roxana; National Institute of Forest Research and Management (INCDS)
Keywords:	Machine learning algorithms, Classification algorithms, Diffusion processes
Subject Category Please select at least two subject categories that best reflect the scope of your manuscript:	Computers and information processing, Mathematics
Additional Manuscript Keywords:	learning pipeline for classification, Scala language

Introduction to Statistical Computing in Scala - an Implementation of the K-Nearest Neighbors classifier

Roxana Tesileanu, *Research Assistant, National Institute of Forest Research and Management (INCDS), Romania*
roxana.te@web.de

***Abstract*—Statistical computing in ecology evolves at a high speed, mainly because researchers have recognized the advantage of being able to design their algorithms according to their needs. The present paper introduces the implementation in Scala language of the k-Nearest Neighbors (kNN) classifier, which can be applied also on small datasets, a situation commonly encountered in ecological research, and discusses its possible role within a more complex learning pipeline for classification tasks.**

***Index Terms*—learning pipeline for classification, machine learning in ecology, diffusion processes, k-Nearest Neighbors classification, Scala language, Simple Build Tool (SBT), multivariate analyses**

I. Introduction

One of the drivers of the machine-learning progress is the great amount of data born within and gathered by networked and mobile computing systems which necessitates further processing in order to gain insights into the specific fields from which it originates [5]. The ”Big Data phenomenon” is real but, unfortunately, it can not be generalized to all research areas, especially some areas of ecological research which investigate systems which are by nature data-poor and will probably remain as such unless cost-intensive data collection projects are being proposed and financed. This by no means implies that such areas cannot take advantage of the progress of machine-learning and take the best out of the existing datasets. On the contrary, machine-learning, being a study field which ”sits at the crossroads of computer science, statistics and a variety of other disciplines concerned with automatic improvement over time, and inference and decision-making under uncertainty” [5], is welcomed in real-world environmental decision-making facing the need of proposing courses of action under the cloud of uncertainties and regarding issues characterized by multiple attributes [16]. Moreover, machine-learning is used not only in analyzing data from observational studies (which might benefit from larger time series datasets), but also from experimental studies (which can’t produce large amounts of data but benefit from an experimental design, delivering data which provides stronger inference about effects and causal relationships [8]) [3]. Indeed, a combination of machine-learning classifiers with the benefits of a MANOVA experimental design used for collecting training data, might produce useful research results, in ecological research. Thus, in order to be able to use the power of machine-learning algorithms adapted to small ecological datasets, a statistical package written in Scala is currently under development at INCDS.

The idea of adapting the use of machine-learning algorithms to real-world data requirements is not new. Hastie and Tibshirani [4], describe the Discriminant Adaptive Nearest Neighbor (DANN) procedure as a locally adaptive form of nearest neighbors classification using a modified linear discriminant analysis (LDA) procedure to estimate an effective metric for computing neighborhoods. The main idea of combining distance-based approaches with approaches based on the association between variables is driven by the fact that a multivariate dataset can be analysed in both ways [8]: calculating scores for the derived variables (components) for each object, and calculating dissimilarity measures (distances) for every possible pair of objects.

The present paper introduces the Scala functions used to run the kNN classification algorithm and then it delineates its role within a real-world classification pipeline for reducing error.

II. Running the kNN algorithm

In order to be able to run the kNN algorithm written using the Scala language, one should consider which tools to choose for the different components of a development system - which are according to Rehman and Paul [9]: the hardware platform, the operating system, editors, compilers and assemblers, debuggers, version control system and bug tracking. For implementing Scala projects, working with the Scala Build Tool (SBT) (http://www.scala-sbt.org/) condenses the list of the components of a development system but also requires a default structure of the project directory with the source files of user-defined packages located in the src subdirectory [11] following a standard pattern. This is the reason for which the source files of the scalaML package containing the kNN classifier are placed at the end of a chain of subdirectories within the project directory (see appendix A through E). Another advantage of using SBT for the statistical computing workflow, beside condensing the list of components of a development system, is the interactive use of the Scala REPL, a tool for evaluating expressions in Scala (https://docs.scala-lang.org/overviews/repl/overview.html), within a SBT session (for further details and references see [14]). In order to access the functions for the kNN classification, the package needs to be imported into the

current REPL session.

The kNN classifier reads CSV and text files for classification tasks, uses basic vector operations for handling variables and objects, runs the kNN algorithm on the data and computes the error rate of the kNN classifier.

A. Basic vector operations

Because most statistical algorithms manipulate datasets which are actually collections of vectors, manipulating vectors is an essential task. The scalaML package offers this functionality, using the Scala main types while developing machine-learning algorithms. This makes the process of synchronizing types in further development stages easier. The vector operations are found in the BasicVectorOP.scala source file (see appendix A) and include: vector addition, vector subtraction, elementwise multiplication and dot product for any two vectors of type Array[Double], and matrix multiplication for any two matrices of type Array[Array[Double]], where each Array[Double] represents an observation (object), and all elements with the same index of inside arrays (i.e. all elements with the same index inside of all Array[Double] within the Array[Array[Double]]) represent a column (i.e. a variable) of the multivariate dataset [2], [7], [12], [15], [1].

The documentation of the kNN classifier offeres worked examples of functions' application in REPL (see [13]).

B. Reading files for classification tasks

The ReadFile.scala source file (appendix B) of the scalaML package contains the function used to read CSV and text files for classification tasks. It returns a tuple with the information needed for the kNN algorithm (a data matrix of type Vector[Array[Double]], the data labels in the form of a Vector[Int], and the used classes in the form of a Range[Int]) [3], [7], [12], [1] (see [13] for details).

C. KNN classification function

The kNN classification function (see appendix C) implements in Scala the pseudocode of the kNN algorithm [3], [7], [12], [1]. It starts by calculating the Euclidean distance between a new object (which is going to be classified by the algorithm) and every object of the dataset, it sorts the distances in increasing order, it then takes k objects with the lowest distances and finds the majority class among those k objects, returning it as the prediction for the class of the new object [3]. The documentation of the kNN classifier gives a thorough example of its use (see [13]). The variables can also be scaled from 0 to 1 before using the kNN classifier, if a normalization is required. For this purpose a normalizing function can be used (see appendix D) [8], [3], [7], [1].

D. Testing framework

In order to test the accuracy of the kNN classifier, an error rate can be computed using the testing function provided in the TestFrameWorkClassif.scala source file (see appendix E) [3], [7], [1]. The error rate is given by the total number of errors (misclassified observations) divided by the total number of tested observations (see [13] for details on its use).

III. The learning pipeline for reducing error

The machine-learning workflow for classification tasks for high-dimensional datasets (i.e. more than three variables recorded for each object) generally consists of the following steps: data collection, reading the data in R, Scala, or other programming language used for statistical computation, analyzing the data using multivariate analyses like (MANOVA, PCA, cluster analysis, etc.), applying and testing the classification algorithm (kNN, Bayes classifier, etc.), deciding on and argumenting the choice of the most appropriate model. Real-world workflows are itterative, requiring a dynamic configuration of the used algorithms and eventually the application of different algorithms to compare the results. Such workflows are called in machine-learning practice "learning pipelines" [6].

The present paper proposes a combination of the MANOVA experimental design and analysis, with a further data classification step using the kNN classifier. The background idea is that, if two or more groups differ in their group centroids, they are, globally speaking, suitable for classification tasks. If, not, then we probably miss the variables which make a difference, or, we don't really have two different populations, and further classifications are possible but miss their point. In the first case, we further investigate (add new variables or remove current variables), and if the MANOVA test indicates group differences, we are able to find a first decision boundary using the linear combination with the highest eigenvalue.

The decision boundary ideally separates the different groups. Though, variables overlap in the decision space, forming a "transition zone". This is where the residual variance comes from. KNN can then be further applied to investigate the situation occurring in the neighborhood of the objects located in the transition zone, and the best techniques to do this logically appear to be distance-based (i.e. based on dissimilarity measures between objects). Now, the question is how we delineate the transition zone? We can check the objects which produce the MANOVA residual (analysis of the residual) [8]. For high residuals we choose a higher k number, and for low residuals we choose a lower k number. This approach could represent a helpful investigation pipeline for diffusion processes, possibly requiring the use of alternative dissimilarity measures identifying the correct direction for neighbor selection, and certainly needs further investigation. Nevertheless, it promises a high practical relevance for environmental data analysis.

IV. Conclusion

Scala has already been adopted for developing machine-learning algorithms and is one of the languages in which Spark's MLlib (Spark's library for machine learning functions) is written. Its main functionality is to offer parallel algorithms which run well on data clusters, so it is used for analysing large data sets. Some classic algorithms are not included because they were not designed for parallel platforms [6].

For ecological research purposes, being able to run machine-learning algorithms on small datasets is essential. As a result, a package for statistical computing written in Scala is currently under development at INCDS. In addition, this offers the opportunity of exploring new learning pipelines advancing the field of machine-learning research. One such example is combining the MANOVA experimental design and analysis with the use of the kNN classifier, for adjusting the decision boundary in classification tasks involving multivariate datasets.

APPENDIX A
SOURCE FILE - BASICVECTOROP.SCALA

Please follow the link to the source file:
https://github.com/RoxanaTesileanu/multivariate_analyses/blob/master/DeepLearning/src/main/scala/com/mai/scalaML/BasicVectorOP.scala

APPENDIX B
SOURCE FILE - READFILE.SCALA

Please follow the link to the source file:
https://github.com/RoxanaTesileanu/multivariate_analyses/blob/master/DeepLearning/src/main/scala/com/mai/scalaML/ReadFile.scala

APPENDIX C
SOURCE FILE - KNN.SCALA

Please follow the link to the source file:
https://github.com/RoxanaTesileanu/multivariate_analyses/blob/master/DeepLearning/src/main/scala/com/mai/scalaML/kNN.scala

APPENDIX D
SOURCE FILE - AUTONORM.SCALA

Please follow the link to the source file:
https://github.com/RoxanaTesileanu/multivariate_analyses/blob/master/DeepLearning/src/main/scala/com/mai/scalaML/AutoNorm.scala

APPENDIX E
SOURCE FILE - TESTFRAMEWORKCLASSIF.SCALA

Please follow the link to the source file:
https://github.com/RoxanaTesileanu/multivariate_analyses/blob/master/DeepLearning/src/main/scala/com/mai/scalaML/TestFrameWorkClassif.scala

ACKNOWLEDGMENT

The author would like to thank Jeff Druce and Mike Reposa from CRA for very useful learning tips and Georgeta Ionescu from INCDS for giving me the financial support needed. The present document was edited using the LaTeX document class IEEEtran developed by Michael Shell [10].

NOTE

This paper is part of the project "Experimenting with Scala and R for multivariate analyses", which is stored on my GitHub profile under the following link: https://github.com/RoxanaTesileanu/multivariate_analyses. Further documentation on using the kNN classifier [13], and on using Linux as a development platform for Scala projects [14] can be found under the following link: https://www.researchgate.net/profile/Roxana_Tesileanu/publications

REFERENCES

[1] Scala Standard Library 2.12.0, 2003.
[2] Paul Dawkins. *Paul's notes on linear algebra*. Math Tutorial Lamar University. Lamar University, 2005.
[3] Peter Harrington. *Machine learning in action*. Manning Publications Co., Shelter Island, 2012.
[4] Trevor Hastie and Robert Tibshirani. Discriminant Adaptive Nearest Neighbor Classification and Regression. *IEEE Trans. Pattern Anal. Match. Intell.*, 18(6):607–616.
[5] M. I. Jordan and T. M. Mitchell. Machine learning: trends, perspectives, and prospects. 349(6245):255–260, July 2015.
[6] Holden Karau, Andy Konwinski, Patrick Wendell, and Matei Zaharia. *Learning Spark*. O'Reilly, Sebastopol, 2015.
[7] Martin Odersky, Lex Spoon, and Bill Venners. *Programming in Scala*. Artima, Walnut Creek, second edition, 2010.
[8] Gerry Quinn and Michael Keough. *Experimental design and data analysis for biologists*. Cambridge University Press, Cambridge, 2002.
[9] Rafeeq Ur Rehman and Christopher Paul. *The Linux Development Platform*. Pearson, 2003.
[10] Michael Shell. How to use the IEEEtran LaTeX class. *Journal of LaTeX class files*, 14(8).
[11] Joshua Suereth and Matthew Farwell. *SBT in action: the simple Scala Build Tool*. Manning Publications Co., Shelter Island, 2016.
[12] Jason Swartz. *Learning Scala*. O'Reilly, Sebastopol, 2015.
[13] Roxana Tesileanu. Introduction to statistical computing in Scala: an application of the k-Nearest Neighbors classifier, 2017.
[14] Roxana Tesileanu. Using Linux as a development platform for Scala projects, 2017.
[15] Andrew Trask. *Grokking deep learning*. Manning Publications Co., Shelter Island, 2017.
[16] Arild Vatn and Daniel W. Bromley. Choices without prices without apologies. *Journal of Environmental Econimics and Management*, 26:129–148, 1994.

Roxana Tesileanu graduated Forestry and Environmental Sciences (Dipl. Forstwirtin, German equivalent to BSc. plus MSc.) in 2008 at Albert-Ludwigs Universitaet in Freiburg i. Br., Germany, and works since 2014 as research assistant at INCDS Romania. She is specialized in environmental statistics and bioinformatics.

europass Curriculum vitae

PERSONAL INFORMATION

Roxana Tesileanu

str. Unirii, nr. 103, 505600 Sacele (Romania)

(+40)731207004

roxana.te@web.de

www.researchgate.net/profile/Roxana_Tesileanu https://github.com/RoxanaTesileanu

WORK EXPERIENCE

15/07/2004–30/09/2007 — Student Assistant (wissenschaftliche Hilfskraft)
Institute of Soil Science, Albert-Ludwigs Universitaet Freiburg, Freiburg i. Br. (Germany)

01/10/2008–14/01/2009 — Internship
Swiss Federal for Forest and Landscape Research WSL, Economics and Social Sciences Research Units, Birmensdorf (Switzerland)

01/06/2009–28/02/2010 — scientific-technical assistant
Schwz. Eidg. Agroscope, Changins Station, Nyon (Switzerland)

01/04/2010–01/09/2013 — Sales and marketing department manager
Feinmoebel Freiburg GmbH, Freiburg i. Br. (Germany)

08/05/2014–Present — Research Assistant
National Institute of Forest Research (INCDS) , Brasov Station, Bucharest (Romania)

EDUCATION AND TRAINING

01/09/2001–30/07/2003 — Allgemeine Hochschulreife
Goethe Gymnasium, Freiburg i. Br. (Germany)

01/10/2003–30/06/2008 — Dipl. Forstwirtin
Albert-Ludwigs Universitaet, Freiburg (Germany)

PERSONAL SKILLS

Mother tongue(s): Romanian

Other language(s):

	UNDERSTANDING		SPEAKING		WRITING
	Listening	Reading	Spoken interaction	Spoken production	
German	C2	C2	C2	C1	C2
English	C1	C1	C1	C1	C1
French	B2	B2	B2	B2	B2

Levels: A1 and A2: Basic user - B1 and B2: Independent user - C1 and C2: Proficient user
Common European Framework of Reference for Languages

europass Curriculum vitae Roxana Tesileanu

Job-related skills

- advanced skills in classical statistical data analysis
- good use of the R statistical environment
- good use of the Python packages for scientific data analysis
- good use of the Scala language; a machine-learning package in Scala is currently under development (https://github.com/RoxanaTesileanu/multivariate_analyses/tree/master/DeepLearning/src/main/scala/com/mai/scalaML)
- initial exposure to Bayesian data analysis with the Figaro probabilistic programming language
- interdisciplinary working skills

Digital competence

SELF-ASSESSMENT				
Information processing	Communication	Content creation	Safety	Problem solving
Proficient user	Proficient user	Proficient user	Proficient user	Proficient user

Digital competences - Self-assessment grid