An implementation of RDF-Graph Kernels for Spark

Dennis Kubitza [*]

Maximilian Radomsky [†]

February 12, 2018

Project References on Github

Lab Report [1]

Abstract

Machine learning paradigms strongly depend on the specific structure of the observed and unobserved Data. For the big goal of promoting Machine Learning on Structured Data like Resouce Description Frameworks, with its schema-free structure, one class of Algorithms is naturally well suited: Kernel Based Algorithms. We follow the examinations of Lösch et al. (2012) and implement their proposed Graph-Kernels for the usage in Apache Spark, especially for further usage in the Semantic Analytics Stack (SANSA). Our implementation combines different approaches from Graph Combinatorics, Data-Mining and Big Data Analysis to ensure scalability in storage and computational performance.

[*]**Email: denn_kubi@freenet.de**, Rudolf-Breitscheid-Str.1 40595 Düsseldorf, Germany

[†]**Email: maxradomskyy@gmail.com,**

[1]as Part of the Examination of Modul 4223, Master of Computer Science, University of Bonn

Contents

1 Introduction

While each every Machine Learning Task is defined by its Input Set, it's Set of valid models and the expected behaviour of the learning Agent, some algorithms exist that solve problems under such general assumtpions that almost any Data-dependet Problem can be reduced to fit their requirements. Kernel-based Machine learning methods don't require any specific structure for the Data, solomly that a scalar valued function exists, suitable for summarizing a Observation or Subobservation as a single value. We call such a function a kernel Functions. Before stating a mathematic exact definition, applyable to even the most general settings we are taking a look on some examples where Kernel-Based Machine learning is applied to RDF.

Kernel Examples

- XXX
- XXX
- XXX

2 Problem Statement

A definition of Kernels, that is suitably general and applyable to all Machine Learning Algorithms, but still mathematically precise can be found in (Shawe-Taylor and Cristianini, 2004). In the Context of RDF-Graphs it is possible to reformulate this definition, as Lösch et al. (2012) did, bridging machine learning to feature-representation models for structured Data.

Definition 1 *Let $\mathscr{X}$ be a Subgraph of an RDF-Graph and let $\phi : \mathscr{X} \Rightarrow \mathbb{R}^k$ be a feature representation. A kernel Function is given by*

$$k(x, y) =< \phi(x), \phi(y) >_H$$

, where $< \cdot, \cdot >_H$ extends $\mathbb{R}^k$ to a Hilbert space.

In the case of RDF-Data Lösch et al. (2012) listed four different Kernels that are in particular suitable, as they may scale to both local and global features and compute Kernels independently from the type of reference or literals given. This is due to the used feature representation, where the existance of certain characteristic subgraphs are identified with Indicator-Variables in the Feature Representation of two Subgraphs of Interest. In the follwing Paragraphs we will explain them and possible problems / computation approaches, we want to consider in our implementations.

2.1 Walk Kernel

The path kernel corresponds to a weighted sum of the cardinality of walks up to a length l, or more formally:

$$(\kappa_{l,\lambda})(G_1, G_2) = \sum_{i=1}^{l} \lambda^i \big|\{p | p \in walks_i(G_1 \cap G_2)\}\big|$$

Lösch et al. (2012). Concerning the implementation it should be very easy to paralize the this process, as the addition of an Edge to a Walk is independent of its preceeding sequence of Edges. We decided to XXX . For further details we refer to **??**.

2.2 Path Kernel

The Path Kernel uses the same principle as the Walk Kernel, but counts the number of paths.

$$(\kappa_{l,\lambda})(G_1, G_2) = \sum_{i=1}^{l} \lambda^i \big|\{p|p \in paths_i(G_1 \cap G_2)\}\big|$$

Lösch et al. (2012). As it is now not possible to use the path independence, as before, we also need to alter the the approach. Most favorable without the need of accessing previous calculations steps, e.g to check if we already visited a single node.

2.3 Full Subtree Kernel

Already Lösch et al. (2012) mentioned that the computation of a Intersection Graph might get costly. They therefore proposed to limit the Calculations of Kernels, not on arbitrary Subgraphs, but only on certain Subgraphs wich can be identified with a certain central entitiy. This enables a replacement of the Intersection Graph with other suitable structures. One of them is the so called Intersection Tree:

Definition 2 *XXX*

We can obtain the Full Subtree Kernel by XXX to the Intersection Tree of two entities:

Definition 3 *XXX*

The main target of computational power is now the Intersection Graph. While **?** propose an easy Algorithm to generate them, they did not consider the nessecarity of parralelization. In **??** we provide a parralized Version of the Algorithm together with interesting parts of its computation. The final enumeration of Full Subtrees can easily be paralized, as each connected subgraph of a Tree is neccesarily again a Tree.

2.4 Partial Subtree Kernel

Like the Full Subtree Kernel, the Partial Subtree Kernel is defined by XXX on the Intersection Tree of two Entities:

Definition 4 *XXX*

Psychic Costs and Human Capital Investment I also shed new light o

References

Lösch, U., Bloehdorn, S., and Rettinger, A. (2012). Graph kernels for rdf data. In Simperl, E., Cimiano, P., Polleres, A., Corcho, O., and Presutti, V., editors, *The Semantic Web: Research and Applications*, pages 134–148. Springer Berlin Heidelberg, Berlin, Heidelberg.

Shawe-Taylor, J. and Cristianini, N. (2004). *Kernel methods for pattern analysis*. Cambridge university press.