Information Processing Letters 126 (2017) 1–6

Contents lists available at ScienceDirect

Information Processing Letters

www.elsevier.com/locate/ipl

An incremental linear-time learning algorithm for the Optimum-Path Forest classifier

Moacir Ponti *, Mateus Riva

Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo – São Carlos, SP 13566-590, Brazil

ARTICLE INFO

Article history:
Received 23 November 2016
Received in revised form 2 May 2017
Accepted 11 May 2017
Available online 16 May 2017
Communicated by R. Uehara

Keywords:
OPF
Graph algorithms
Computational complexity
Minimum spanning tree
Machine learning

ABSTRACT

We present a classification method with incremental capabilities based on the Optimum-Path Forest classifier (OPF). The OPF considers instances as nodes of a fully-connected training graph, arc weights represent distances between two feature vectors. Our algorithm includes new instances in an OPF in linear-time, while keeping similar accuracies when compared with the original quadratic-time model.

1. Introduction

The optimum-path forest (OPF) classifier [1] is a classification method that can be used to build simple, multi-class and parameter independent classifiers. One possible drawback of using the OPF classifier in learning scenarios in which there is need to constantly update the model is its quadratic training time. Let a training set be composed of n examples, the OPF training algorithm runs in $O(n^2)$ time (note that all complexity in this paper relates to running time). Some efforts were made to mitigate this running time by using several OPF classifiers trained with ensembles of reduced training sets [2] and fusion using split sets using multi-threading [3]. Also, recent work developed strategies to speed-up the training algorithm by taking advantage of data structures such as [1,4]. However, an OPF-based method with incremental capabilities is still to be investigated, since sub-quadratic algorithms are important in many scenarios [5].

Incremental learning is a machine learning paradigm in which the classifier changes and adapts itself to include new examples that emerged after the initial construction of the classifier [6]. As such, an incremental-capable classifier has to start with an incomplete *a priori* dataset and include successive new data without the need to rebuild itself. In [4] the authors propose an alternative OPF algorithm which is more efficient in retraining the model, but their algorithm is not incremental. Also, the empirical evidence shows that the running time is still quadratic, although with a significantly smaller constant. In this paper we describe an algorithm that can include new examples individually (or in small batches) in a previously built model, which is a different objective when compared to [4] and [1]. In fact, we already used the improvements proposed by [1]. Therefore our new algorithm **does not compete**, but rather can be used as a complement for those variants.

Because OPF is based on the Image Foresting Transform for which there is a differential algorithm available (DIFT) [7], it would be a natural algorithm to try. However, DIFT is an image processing algorithm and includes all new pixels/nodes as prototypes, which would progressively con-

* Corresponding author.
E-mail address: ponti@usp.br (M. Ponti).

http://dx.doi.org/10.1016/j.ipl.2017.05.004
0020-0190/

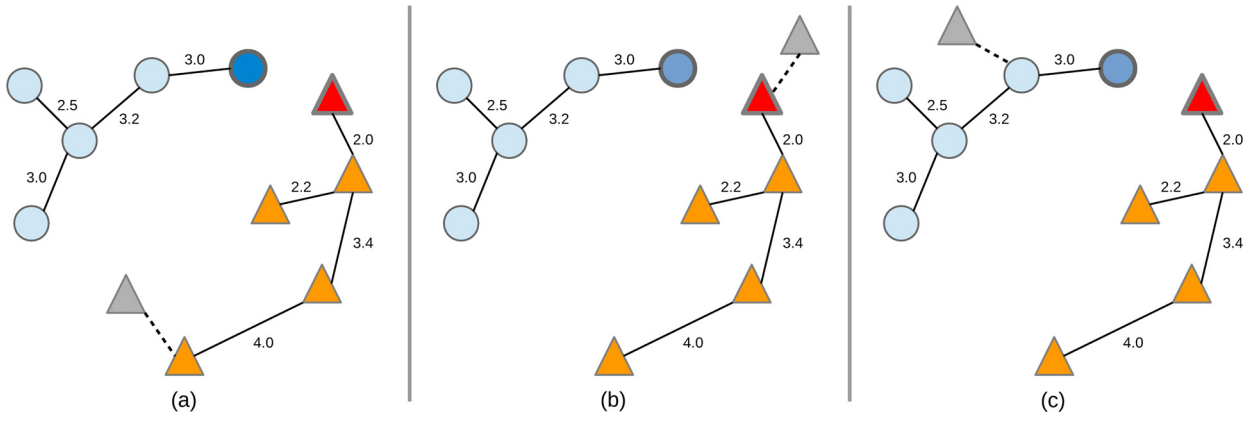

Fig. 1. OPF-Incremental cases when adding a new example (a grey triangle): (a) conquered by a tree of the same class through a non-prototype, (b) conquered by a tree of the same class through a prototype, (c) conquered by a tree of a distinct class.

vert the model into a 1-Nearest Neighbour classifier. Therefore we propose an alternative solution that maintains the connectivity properties of optimum-path trees.

Our OPF-Incremental (OPFI) is inspired in graph theory methods to update minimum spanning trees [8] and minimal length paths [9] in order to maintain the graph structure and thus the learning model. We assume there is an initial model trained with the original OPF training, and then perform several inclusions of new examples appearing over time. This is an important feature since models should be updated in an efficient way in order to comply with realistic scenarios. Our method will be useful everywhere the original OPF is useful, along with fulfilling incremental learning requirements.

2. OPF incremental (OPFI)

The optimum-path forest (OPF) classifier [1] interprets the instances (examples) as the nodes (vertices) of a graph. The edges connecting the vertices are defined by some adjacency relation between the examples, weighted by a distance function. It is expected that training examples from a given class will be connected by a path of nearby examples. Therefore the model that is learned by the algorithm is composed by several trees, each tree is a minimum spanning tree (MST) and the root of each tree is called prototype.

Our OPF incremental updates an initial model obtained by the original OPF training by using the minimum-spanning tree properties in the existing optimum-path forest. Provided this initial model, our algorithm is able to include a new instance in linear-time. Note that in incremental learning scenarios it is typical to start with an incomplete training set, often presenting a poor accuracy due to the lack of a sufficient sample.

Our solution works by first classifying the new example using the current model. Because the label of the classified example is known, it is possible to infer if it has been conquered by a tree of the same class (i.e. it was correctly classified) or not. We also know which node was responsible for the conquest, i.e. its **predecessor**, thus we have three possible cases:

1. **Predecessor belongs to the same class and is not a prototype**: the new example is inserted in the predecessor's tree, maintaining the properties of a minimum spanning tree.
2. **Predecessor belongs to the same class and is a prototype**: we must discover if the new example will take over as prototype. If so, the new prototype must reconquer the tree; otherwise, it is inserted in the tree as in the first case.
3. **Predecessor belongs to another class**: the new example and its predecessor become prototypes of a new tree. The new example will be root of an new tree; while the predecessor will begin a reconquest of its own tree, splitting it in two.

Fig. 1 illustrates the three cases when an element of the 'triangle' class is inserted in the OPF.

The classification and insertion of new elements is described on Algorithm 1 and shows the high-level solution described above.

Algorithm 1 OPF-Incremental insertion.

Require: a previously trained OPF model T with n vertices; new instances to be included $Z[1...b]$.

```
1: OPF_Classify(Z, T) // as in [1]
2: for i ← 1 to b (each new example) do
3:    if Z[i].label = Z[i].truelabel then
4:       if Z[i].pred is prototype then
5:          recheckPrototype(Z[i],Z[i].pred,T) // Algorithm 3
6:       else
7:          insertIntoMST(Z[i],Z[i].pred,T) // Algorithm 2
8:       end if
9:    else
10:      Z[i] becomes a prototype
11:      Z[i].pred becomes a prototype
12:      reconquest(Z[i].pred Z[i].pred, T) // Algorithm 4
13:   end if
14: end for
15: return T
```

The minimum spanning tree insertion function, described on Algorithm 2 is an adapted version of the minimum spanning tree updating algorithm proposed by [8]. The function for rechecking a prototype, described on Algorithm 3 takes the distance between the prototype and its