FEDERAL UNIVERSITY OF PARANÁ

LIOR SPACH

NOVEL WAY TO COMPUTE COST AND SELECT PROTOTYPES IN OPTIMUM PATH FOREST CLASSIFIERS

CURITIBA

2017

LIOR SPACH

NOVEL WAY TO COMPUTE COST AND SELECT PROTOTYPES IN OPTIMUM PATH FOREST CLASSIFIERS

Undergraduate thesis presented to the thesis committee of Federal University of Paraná, as a requirement to obtain bachelor degree in Computer Science.
Advisor: Prof. Eduardo Jaques Spinosa

CURITIBA

2017

Abstract

Optimum path forest (OPF) has shown good results compared to traditional supervised classification algorithms like KNN, BC, SVM and ANN-MLP. Despite its success it is sensible to outliers.[3] The prototype selection method and the cost function are two essential components of OPF. They have direct impact in execution time and accuracy of the classifier and define how the decision boundary is created. Here we propose an alternative algorithm for these components, which better handles outliers and redefines the impact of connectivity on the decision boundary.

Contents

1 Introduction **2**

2 Supervised Classification Problem **4**

3 Optimum Path Forest (OPF) **6**

3.1 Training Phase 6

3.1.1 Prototype Selection 7

3.1.2 Prototype Competition 8

3.2 Classification Phase 9

3.2.1 f_{max} 9

3.2.2 $f_{percent}$ 9

4 Comparison **12**

4.1 Linear Separable 12

4.2 Outliers 13

5 Experiments **14**

5.1 Shape databases 14

5.2 Complex databases 15

6 Conclusion **17**

6.1 Future work 17

Chapter 1

Introduction

Supervised classification is the problem of associating a class to an object based on previously known set of objects for each class. For example, a physician may want to know if a patient is highly likely to have or not have cancer based on previous patients data. By comparing the medical test result of the patient with previous results of other patients who had cancer and who didn't have cancer, the physician could make an educated guess as to the likelihood of the current patient having cancer. The idea is to associate the patient, described by his test results, with one of two classes, "Has cancer" or "Doesn't have cancer".

This problem is of interest to many areas like speech recognition, spam detection, handwriting recognition, chemistry and bioinformatic, to cite a few.

Some well known and used solutions to the problem are K-nearest neighbors (KNN), Support Vector Machine (SVM), Bayesian Classifier (BC) and Multilayer Perceptron neural networks (ANN-MLP). A new solution was proposed in the years 2000's and is said to be faster then SVM and superior to the rest [1]. It has no parameters, has native multiclass support and has shown good results when compared to the other methods [2]. The new solution is called Optimum Path Forest.

Optimum Path Forest (OPF) is a proposed computational solution for the supervised classification problem. It shapes the prior knowledge in the form of a weighted complete graph, from which it then derives a set of trees to represent each class. A new object is then classified by finding the tree that offers it the best cost.

The OPF algorithm have two phases, first the data is reshaped as a graph and the trees for each class are computed. That is called the training phase. When a new sample arrives, the tree that offers it the best cost is found and the tree's class is associated to the sample, this is classification phase.

Two important components of OPF are the prototype selection method and the cost function. They have direct impact on the training and classification phases execution time complexity and accuracy. When proposed, OPF were paired with the MST method for prototype selection and f_{max} for cost function. In this work, we introduce two alternatives: A more intuitive solution for prototype selection method, called Nearest Neighbors (NN), and a new cost function called $f_{percent}$. NN provides same and sometimes better accuracy, while maintaining same execution time complexity. $f_{percent}$ is better when dealing with outliers and reinforces one of the main ideas of OPF, that is, connectivity.

This work focuses on the analysis of these two prototype selection methods and cost functions, comparing how they differ and change the classifier behaviour. Test on simple and complex databases are then used for experimentation.

The text is organized as follows. In chapter 2 the supervised classification problem is formalized. In chapter 3 OPF algorithm and its variations are explained and formalized. The differences between the variations are explored in chapter 4. In chapter 5, experiments results are shown and discussed. We conclude in chapter 6 by summarizing the important points and future work.

Chapter 2

Supervised Classification Problem

In the area of machine learning, there is an interest in making a computer categorize objects. The idea is to implement a function that takes a unknown object from a known domain, and based on previously given knowledge from the same domain, gives the best guess as to which category that object is part of.

The goal is to find a function that can correctly infer the category of new objects not seen before.

To follow the same naming convention of the machine learning community, the word *sample* is used to refer to an object and the word *class* when referring to a category.

A *sample* is represented by its feature vector, which for the scope of this work, will be defined as a sequence of real numbers $(v_1, v_3, v_4, ..., v_n) \in \Re^n$. The feature vector describes the sample. It can be understood as a collection of relevant measures that helps differentiate one sample from another. Choosing good metrics to create the feature vector is an important step in machine learning in order to obtain good classification accuracy. In this work, we assume the feature vectors were already given.

The *previous knowledge*, also known as *training samples*, is defined by a collection of samples $T = \{s_1, s_2, ...s_{|T|}\}$ and their respective classes $\lambda(s_1)$, $\lambda(s_2)$, ..., $\lambda(s_{|T|})$, where each s_i is bound to the same domain $\Re^n$.

The goal, is to use the *previous knowledge* to find function F, that takes a sample $x \in \Re^n$ and gives it's true class $\lambda(x)$.

$$F(x) = \lambda(x)$$

In most cases, when dealing with real world complex problems, function F can only be approximated. The reason could be, for example, having limited information in the training set or the features extracted for the samples not being representative enough.

To measure how good a classification function is, a *validation sample set*, where the classes are known, is used to compare the output of the function with the true class of each sample. A score is then derived.

Chapter 3

Optimum Path Forest (OPF)

Optimum Path Forest (OPF) is a framework to create graph based classifiers, for both supervised and unsupervised classification. Its based on viewing the samples as nodes in a graph, connected by edges defined by an adjacency relation and weighted by some distance metric.

In this work, only one version of OPF is used, which uses complete graph for adjacency relation and euclidean distance for distance metric. For simplicity, this configuration will be referred by the same acronym 'OPF'.

After shaping the samples as nodes in a complete graph, the first step is to identify the prototypes (i.e. the important nodes) using the prototype selection method. Then a process of competition starts where the prototypes try to conquer the other nodes, forming a collection of trees. These trees represent a trained classifier. That process describes the *training phase* of OPF.

When classifying a new sample, a similar process to the prototype competition is done. Each node offers a cost based computed by the cost function for the new sample. The one that offers the best cost conquers that sample, which then is classified as being of the same class. This is the *classification phase*.

3.1 Training Phase

A trained OPF classifier can be defined as a 4-tuple

$$(Z_1, V, P, L)$$

where $|V| = |P| = |L| = |Z_1|$ and:

- Z_1 is the set of **training samples** represented by their feature vectors;
- P is the **parent vector**, it represents parenthood and enables the representation of trees. $P(i)$ is the parent node of node i;
- L is the **label vector**, it represents the associated class for each node;

- V is the **cost vector**, it associates a cost $V(i)$ to each sample i.

The goal of the training phase is to derive V, P, and L from two inputs: the sample set Z_1 and λ, where $\lambda(i)$, is the real class of sample i, defined for all $i \in Z_1$. This phase has two steps:

1. Prototype Selections; (i.e. finding the nodes of importance)
2. Prototype Competition, which results in the forest.

3.1.1 Prototype Selection

Each sample will be thought of as being a node in a complete graph. The edges are weighed by the euclidean distance between the nodes.

There are different methods for finding the prototypes and the definition of what is a good prototype varies depending on the topology used. In this configuration, i.e. complete graph, the nodes considered important are the ones in the boundaries between classes.

To find the prototypes one of the following two methods can used.

Minimum spanning tree (MST)

This method is based on calculating the MST of the graph. All the nodes that have at least one neighbor of a different class in the MST are selected as prototypes. This is illustrated in figure 3.1.

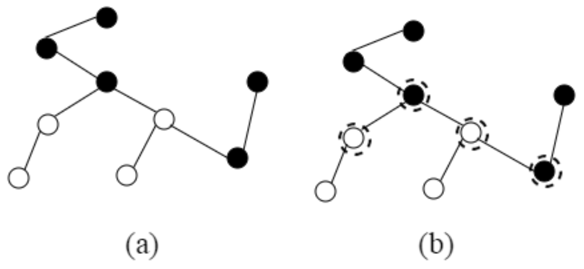

Figure 3.1: (a) Minimum spanning tree. The black and white nodes represent different classes. (b) Nodes selected to be the prototypes are the ones with at least one neighbor of a different class.

Nearest Neighbor (NN)

In this method each node select $m - 1$ other nodes as prototypes, where m is the number of classes. Given a training node set Z, set of classes $C = (c_1, c_2, c_3, ..., c_m)$ we define $Z(c_i)$ as the subset of Z that contains all nodes of class c_i. The collection of nodes selected as prototypes by a node n is defined by:

$$\bigcup_{c_i \in C - \{\lambda(n)\}} \{w\} \ where \ d(n,w) \leq d(n,u), \forall u \in Z(c_i), w \in Z(c_i)$$

where $\lambda(n) \in C$ is the class of node n.

So the complete set of prototypes is:

$$\bigcup_{\forall n \in Z} \bigcup_{c_i \in C - \{\lambda(n)\}} \{w\} \ where \ d(n,w) \leq d(n,u), \forall u \in Z(c_i), w \in Z(c_i)$$

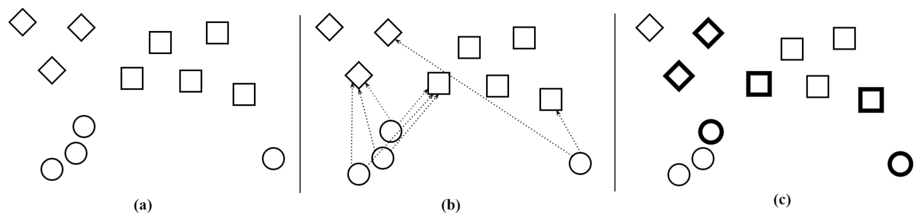

Figure 3.2: (a) Set of nodes, of three classes; (b) Nodes selected to be prototypes by nodes of class circle; (c) nodes selected to be prototypes at the end of the process.

3.1.2 Prototype Competition

Here, each node will offer the others a cost, starting by the prototypes. This step is defined in algorithm 1. Lines 1-8 is initialization. Priority queue Q will initially contain all prototypes. Since the prototypes have the minimum cost, they will be the first to be removed from Q. For each removal the removed sample will offer a cost to all other samples. (line 11, see figure 3.3) The cost offered is the maximum between its own cost and the distance between them. If they are of different class however, the cost offered is $+\infty$. That avoids nodes of a class conquering nodes from another. (line 10-13) If a better cost is offered to a sample, the offerer is set as parent of that sample, represented by vector P, and that sample will be inserted in Q. It is inserted in Q to reanalyze if it can offer better cost to other samples with its updated cost, this guarantees optimum costs in the end of the process.

Notice that samples offer a $+\infty$ cost to samples of different class. (lines 10-13)

As a result of this process, we obtain an optimum path forest, where each tree is rooted in a prototype. The triple cost vector, parent vector and label vector defines our trained OPF classifier.

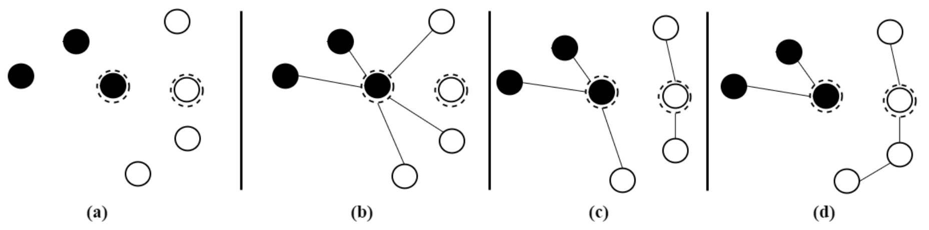

Figure 3.3: (a) The two prototypes are in the priority queue Q. The black and white nodes represent different classes. (b) The black prototype is removed from Q first and offers a cost to other nodes. (c) The white prototype is removed from Q and offers its cost to other nodes. (d) After all nodes are removed from Q.

3.2 Classification Phase

Now we want to define the classification function F, where F takes two parameters: a trained OPF classifier represented by the 4-tuple (Z_1, V, P, L) and a sample t yet to be classified. The output of the function is the predicted class for t.

F can be defined as:

$$F((Z_1, V, P, L), t) = L(n) \ where \ f_{cost}(n, t) \leq f_{cost}(a, t), \forall a \in Z_1, n \in Z_1$$

The algorithm therefore is simple: Calculate the cost each node in the Z_1 offers, take the node which offer the lowest (n) and associate that nodes class $(L(n))$ as the predicted class.

What still need to be defined is the cost function, f_{cost}. Two cost functions are proposed. The first one, f_{max} is originally introduced with OPF [1]. The second cost function, $f_{percent}$, is introduced here.

3.2.1 f_{max}

This cost function can be defined as the biggest distance in the path from a node to its prototype.

$$f_{max}(n, t) = \begin{cases} max\{d(n, t), f_{max}(P(n), n)\} & n \ is \ not \ prototype \\ d(n, t) & otherwise \end{cases}$$

A more intuitive idea of f_{max} is to imagine we connect the sample to be classified t to a node n by an edge and find the biggest edge value from it to the prototype. This is represented in figure 3.4.

3.2.2 $f_{percent}$

The idea of this cost function is that the nodes selected as prototypes are in a region of uncertainty, while nodes that are not selected as prototypes are

Algorithm 1 Prototype Competition

Input: Training set Z_1 λ rotulated, prototype set $S \subset Z_1$ and distance function d.
Output: Parent vector P, cost vector V and label vector L.
Auxiliary: variable cst, priority queue Q.

```
 1: for all $t \in Z_1$ do
 2:     $P(t) \leftarrow nil$
 3:     $V(t) \leftarrow +\infty$
 4: for all $s \in S$ do
 5:     $P(s) \leftarrow nil$ ,$V(s) \leftarrow 0$ ,$L(s) \leftarrow \lambda(s)$
 6:     Insert $s$ in $Q$ with cost $V(s)$
 7: while $Q$ is not empty do
 8:     Remove $s$ from $Q$ where $V(s)$ is minimum.
 9:     for all $t \in Z_1$ where $s \neq t$ and $V(t) > V(s)$ do
10:         if $\lambda(s)! = \lambda(t)$ then
11:             $cst \leftarrow max\{V(s), d(s,t)\}$
12:         else
13:             $cst \leftarrow +\infty$
14:         if $cst < V(t)$ then
15:             if $V(t) \neq +\infty$ then remove $t$ from $Q$
16:             $P(t) \leftarrow s$
17:             $L(t) \leftarrow L(s)$
18:             $V(t) \leftarrow cst$
19:             Insert $t$ in $Q$ with cost $V(t)$
```

in a more certain region. The farthest the node is from the prototype, that is, the more nodes in between them, the more power this node will have on classification.

Different from f_{max}, which is a way to decided which value to select (which edge), the $f_{percent}$ is a modifier applied after calculating the distance.

The modifier value is an integer number, that represent a percentage. The modifier value of node n can be defined as:

$$M(n) = \begin{cases} 5\% & n \textit{ is prototype} \\ -2\% * P_{num}(n) & \textit{otherwise} \end{cases}$$

Where $P_{num}(n)$ is the number of nodes n is away from its prototype in the tree. It can be defined as:

$$P_{num}(n) = \begin{cases} 0 & n \textit{ is prototype} \\ 1 + P_{num}(P(n)) & \textit{otherwise} \end{cases}$$

When offering a cost to a new node t the distance is calculated and the modifier is applied on top. So the cost a node n offers to t is defined by

$$f_{percent}(n,t) = d(n,t) + d(n,t) * max\{M(n), -50\%\}$$

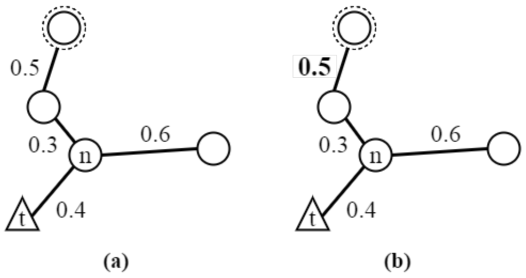

Figure 3.4: f_{max} (a) Starting situation where n is offering a cost to t. (b) By f_{max}, the cost is the biggest distance from t to the prototype.

Notice we limit the modifier to a minimum of -50%.

Figure 3.5 show how the nodes modifier will look like.

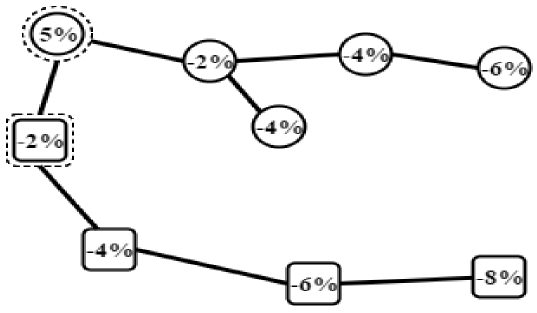

Figure 3.5: $f_{percent}$ Two trees of two classes, circle and retangle, modifier is shown inside each node.

As efficiency goes, the modifier of each node can be calculated after the prototype competition phase and be stored in vector V in $O(n)$. In other words, there is no relevant efficiency difference between using f_{max} and $f_{percent}$.

Chapter 4

Comparison

This chapter shows the differences between the different methods and cost functions for two situations: (a) linear separable, and (b) outliers.

All methods behave similarly for non-linear situations, since the overlapping areas will be mostly composed of prototypes, while showing clear differences in (a) and (b).

To compare the solutions, we go through the two cases, showing how the prototypes selections differ, and how the cost functions creates the boundary for each set of prototypes.

4.1 Linear Separable

In a linear separable problem, what we want is to have a well define border. The nodes of importance are the ones close to the border, not the internal ones.

The usual result of applying NN to a linear problem is to have most or all border nodes as prototypes. In such situation, applying f_{max} or $f_{percent}$ is irrelevant. If a classification node is internal, the same classification will happen, if it is in the border, nodes of same power(as both are prototypes) will try to conquer it, leaving it to be conquered by the nearest prototype, whatever the cost function is. So using NN gives us a normal linear separation between classes for both cost functions.

In contrast, when using MST, except special cases, some nodes in the border are selected as prototypes, and the rest is connected to these prototypes, forming a structure where the prototypes form a bridge between classes in the MST.

The frontier nodes that are not prototype are the ones that change the decision boundary. Here one concept of OPF is visible, connectivity. The idea is that the more connected a class is, the more classification power it has.

When node cost is computed by f_{max}, what is really being done is to set a minimum cost for that node. It will not be able to offer less than the minimum when doing classification. The higher the minimum is, the less classification power the node has. The way this is supposed to show connectivity is that if the frontier nodes are too sparse, their power will be decreased.

This has problems though. Firstly, it is unlikely that a node in the frontier will be far from its prototype (because that edge cost will not be desired in the MST), what happens more often is it being selected as prototype as well. When the frontier node is not far from its prototype, the minimum set for its cost is not strong enough to make difference.

In $f_{percent}$, the power is inverted. Prototypes receive a penalty and non-prototype frontier nodes receive a boost. We known that only closest nodes to another class are selected as prototypes. The idea of $f_{percent}$ is that those nodes, the prototypes, are too uncertain and therefore should have less power in classification. If it has other nodes connected to it, that means more certainty, so those nodes receive a boost in power.

4.2 Outliers

In both NN and MST methods, the outlier will be selected as a prototype, together with a few nodes of the other class.

When dealing with outliers, $f_{percent}$ outperforms f_{max}. The reason for this is that an outlier already receives a percentage penalty by being prototype. Also it is surrounded by non-prototype nodes which are boosted.

Chapter 5

Experiments

In this section the OPF algorithm is tested with different databases, results are shown and an analysis is presented. The results are split in 4 cases: MST + f_{max}, MST + $f_{percent}$, NN + f_{max} and NN + $f_{percent}$.

5.1 Shape databases

First, we test with databases known as shapes[1]. They are two dimensional databases with different decision boundaries. The test method used was 4-fold cross-validation, with each fold having similar class ratio. The bases are shown in figure 5.1. The results are shown in table 5.1.

Table 5.1: Mean accuracy for shapes databases using 4-fold cross-validation

Database Name	**MST**+f_{max}	**MST**+$f_{percent}$	**NN**+f_{max}	**NN**+$f_{percent}$
Flame	100.0%	99.59%	100%	99.59%
D31	96.42%	96.48%	95.97%	96.55%
Aggregation	99.74%	99.74%	99.74%	99.74%
Compound	97.5%	97.75%	97.5%	97.75%
Spiral	100.0%	100.0%	100.0%	100.0%
R15	99.5%	99.5%	99.5%	99.5%
Pathbased	98.98%	99.32%	98.98%	98.98%
Jain	100.0%	100.0%	100.0%	100.0%

The result were similar through all methods. What is of interest here is that all new proposed solutions ($f_{percent}$ and nearest neighbors prototype selection method) are capable of solving simple problems.

[1]can be found at:

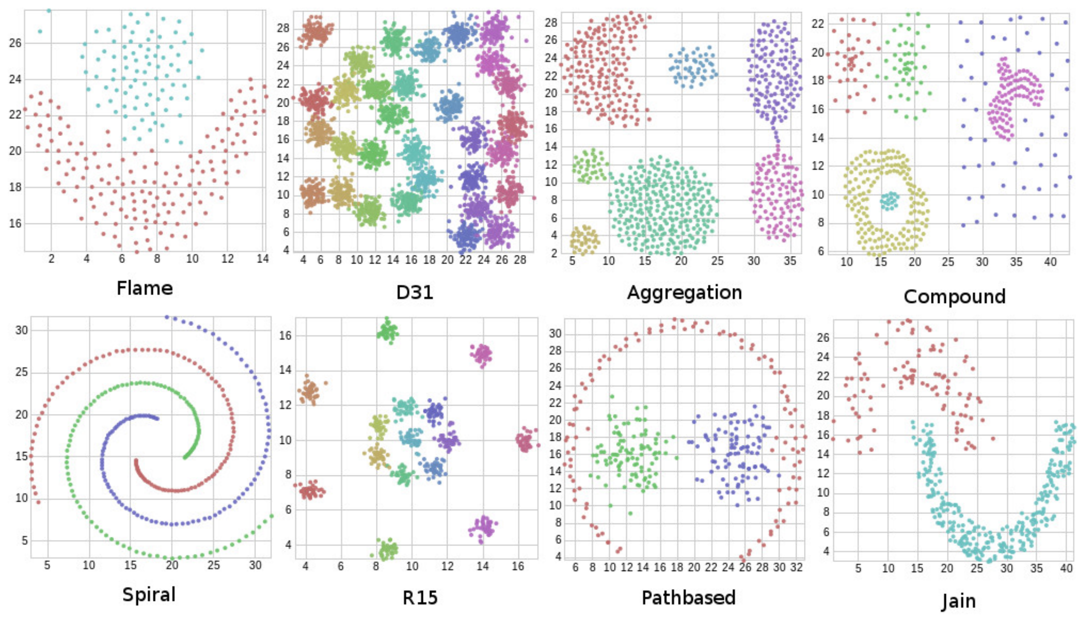

Figure 5.1: View of shapes databases. Colors represent different classes.

5.2 Complex databases

In the second test we use OPF in known databases, more specifically, the **shuttle database**[2], **Image Segmentation database**[3], the **Pen-Based Recognition of Handwritten Digits database**[4] and the **Breast Cancer Wisconsin database**[5]. The objective here is show its efficacy with more complex problems.

For the **Shuttle**, **Image Segmentation** and **Handwritten digits** datasets, the author predefined training and testing sets were used. The shuttle has 43500 training samples and 14500 test samples, 9 attributes, 7 classes where class 1 encompass 80% of samples. The image segmentation dataset has 2310 samples, 210 training, 2100 testing samples, 7 classes with similar density in both sets and 19 attributes per sample. The handwritten digits dataset has 10992 samples, 7494 training, 3498 testing, 16 attributes per sample and 10 classes with similar density in both sets. Accuracy results for these bases are shown in table 5.2.

The **Breast Cancer** dataset has 2 classes, 9 attributes per sample, 699 samples from which 16 were removed for containing partial data (therefore only 683 samples were used). To test, 4-fold cross-validation was applied, maintaining the class ratio similar between folds. The mean accuracy and number of errors

[2]https://archive.ics.uci.edu/ml/datasets/Statlog+(Shuttle)
[3]https://archive.ics.uci.edu/ml/datasets/Image+Segmentation
[4]https://archive.ics.uci.edu/ml/datasets/Pen-Based+Recognition+of+Handwritten+Digits
[5]https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Original)

are reported in table 5.2.

Table 5.2: Accuracy and amount of incorrectly classified samples for complex datasets

Database Name	**MST**+f_{max}		**MST**+$f_{percent}$		**NN**+f_{max}		**NN**+$f_{percent}$	
	Accuracy	Wrong	Accuracy	Wrong	Accuracy	Wrong	Accuracy	Wrong
Shuttle	99.87%	19	99.88%	18	99.88%	17	99.88%	18
Image Segmentation	86.57%	282	87.76%	257	87.47%	263	87.71%	258
Handwritten digits	97.77%	78	96.37%	127	97.74%	79	96.94%	107
Breast Cancer	95.02%	8.5	95.89%	7	95.01%	8.5	95.89%	7

In both Image Segmentation and Breast Cancer databases, there is a slight better performance by $f_{percent}$ over f_{max}. That result is inverse for Handwritten Digits database. Both prototype selection method show similar results for all bases, the one exception being when paired with $f_{percent}$ on Handwritten Digits database, where the result is better.

Chapter 6

Conclusion

We presented the two alternatives algorithms, one for prototype selection and one for cost function. A comparison was made between the methods, arguing that the proposed methods are better at handling outliers and showing how it creates a different decision boundary.

Than experimentation was done to analyze its performance in real problems, showing mostly similar results for the cost functions and a slight better accuracy for nearest neighbors with f_{max}.

6.1 Future work

Although $f_{percent}$ decision boundary is good for outliers, it seem to have a negative impact on other cases. An idea is to set a mean boost to all nodes in the tree, instead strong boost to some and weak to others. Prototype cost modifier should not change, for maintaining control over outliers.

Nearest Neighbors (NN) has shown better results compared to the MST method. It may be interesting to make a more in depth comparison and maybe propose a new version that better harness its strengths.

OPF has many components that can be rethought. One point we didn't touch was the prototype competition. Improvements here mean forming better trees, which has direct impact on the cost functions behaviour, so that is interesting to explore.

Bibliography

[1] Alexandre X. Falcão João P. Papa. doutorado papa. 2009.

[2] Alexandre X. Falcão João P. Papa and Celso T. N. Suzuki. Supervised pattern classification based on optimum-path forest. *Intl. Journal of Imaging Systems and Technology*, 2009.

[3] R. Souza, R. Lotufo, and L. Rittner. A comparison between optimum-path forest and k-nearest neighbors classifiers. In *2012 25th SIBGRAPI Conference on Graphics, Patterns and Images*, pages 260–267, Aug 2012.