**Forest Fire Prediction with Multilayer Perceptron**

Barışcan Tunalı

Big Data Masters Student

Istanbul Bahcesehir University,

bariscantunali@outlook.com

# ABSTRACT

In this paper, I will explain my study on Forest Fire prediction using Multilayer Perceptron and compare the results between experiments with different hidden neuron number.

The aim is to predict the burned area of forest fires, in the northeast region of Portugal, by using meteorological and other data which will be explained later. Main dataset has test set so I didn’t randomly seperate the test set.

# Categories and Subject Descriptors

[**Computing methodologies**]: Machine learning—*Machine learning algorithms; Ensemble methods; Perceptron; Simple Perceptron; Multilayer Neural Network*

**Keywords**

MLP, SLP, Machine Learning, Data Mining, Classification, Prediction, Forest Fire

# INTRODUCTION

One major environmental concern is the occurrence of forest fires (also called wildfires), which affect forest preservation, create economical and ecological damage and cause human suffering. Such phenomenon is due to multiple causes (e.g. human negligence and lightnings) and despite an increasing of state expenses to control this disaster, each year millions of forest hectares (ha) are destroyed all around the world. In particular, Portugal is highly affected by forest fires iFrom 1980 to 2005, over 2.7 million ha of forest area (equivalent to the Albania land area) have been destroyed. The 2003 and 2005 fire seasons were especially dramatic, affecting 4.6% and 3.1% of the territory, with 21 and 18 human deaths[1].

The problem was mainly regression problem. I tried to implement it by using multilayer perceptron. Since problem needs to be implemented in MLP, First we need to apply 1 to C coding to categorical values which are month and day of week.

The paper is organized as follows: In Section 2, we briefly describe the dataset and data preparation. Next, in Section 3, we explain our classification approach in detail. Experimental results and comparisons with other algorithms are provided in Section 4. Finally, Section 5 concludes the paper.

# DATASET AND FEATURES

Forest Fire dataset constists of 517 values, 262 Train,255 Test. I had 13 attributes initial, and 30 attributes after 1 to C coding, target was area.

My attributes are : [X], [Y], [Month], [Day], [FFMC], [DMC], [DC], [temp], [RH],[wind],[rain],[area]

Attribute name: max value – min value

[X] : Categorical : 1 to 9

[Y] : Categorical : 2 to 9

[Month] : Categorical : Months

[Day] : Categorical : Days

[FFMC] : 18.7 to 96.20

[DMC] : 1.1 to 291.3

[DC] : 7.9 to 860.6

[ISI]: 0.0 to 56.10

[temp] : 2.2 to 33.30

[RH] : 15.0 to 100

[wind] : 0.40 to 9.40

[rain] : 0.0 to 6.4

[area] : 0.00 to 1090.84

In order to make my program more flexible I created a class and handled all data operations, test train seperations. On presentation layer you only create class and call Train() function. After that you only access necessary train and test sets and send them to appropriate classification algorithms.

MultiLayerPerceptron = new MLP(); MultiLayerPerceptron.Train(DataPreppp.ForestFireTrainSet,i); MultiLayerPerceptron.Test(DataPreppp.ForestFireTestSet);

i means hidden neuron number.

Data preparation class structure as below :

public void Initialize()

{

ExtractData();

TransformData();

ForestFireOriginalSetTransformed = NormalizeData(ForestFireOriginalSetTransformed);

SeperateTestData(ForestFireOriginalSetTransformed);

}

*ExcractData()* : Dataset was in .csv format so I made a reader function that reads file, creates class, converts string values to appropriate type and bind them into class. Appends this class object into ForestFire List.

*TransformData()* : This function basicly adds ID into each object. Applies 1 to C coding to month and day.

*SeperateTestData()* : Gets normalized data and seperates flagged data as train and rest as train.

# METHOD

As I mentioned above, I used MLP to predict continious attribute.

My MLP class has 3 functions :

public void Train(List<ForestFire> TrainSet, int hiddenCount)

public void Test(List<ForestFire> TestSet)

private void InitializeWeights()

Train and Test are main public functions of the class. Train calls InitializeWeights which randomly assign weights at first.

I didn’t use any c# library or dll to implement MLP itself but I used lambda expressions and Math library to avoid code complexity.

# EXPERIMENTS

I implemented 10 experiments, which creates new classes with given number of (1 to 10) hidden neurons. We have always +1 neuron for both input and hidden layer. So results for these experiments listed below.

Multilayer Perceptron with 0,99 Momentum Rate, 1000 Epoch, 4,31712474106579E-07 Training Coefficient, 30 Input Neurons, 1 Hidden Neurons

6 249

MeanError : -0,643106739115649

SumOfSquareError : 593,638938652063

SumOfSquareErrorMean : 2,32799583785123

SumOfError : -163,992218474491

ErrorStdDev : 1,38362189920341

MeanValue : 0,107339734532049

SumOfSquareValue : 491,128757137718

SumOfSquareValueMean : 1,92599512603026

SumOfValue : 27,3716323056724

ValueStdDev : 1,38364493545882

MeanOutput : 0,750446473647698

SumOfSquareOutput : 143,608395609215

SumOfSquareOutputMean : 0,563170178859668

SumOfOutput : 191,363850780163

OutputStdDev : 0,000518699723036826

Multilayer Perceptron with 0,99 Momentum Rate, 1000 Epoch, 4,31712474106579E-07 Training Coefficient, 30 Input Neurons, 2 Hidden Neurons

6 249

MeanError : -0,560165350319629

SumOfSquareError : 568,175511675244

SumOfSquareErrorMean : 2,22813926147155

SumOfError : -142,842164331506

ErrorStdDev : 1,38360183643013

MeanValue : 0,107339734532049

SumOfSquareValue : 491,128757137718

SumOfSquareValueMean : 1,92599512603026

SumOfValue : 27,3716323056724

ValueStdDev : 1,38364493545882

MeanOutput : 0,667505084851678

SumOfSquareOutput : 113,618816690663

SumOfSquareOutputMean : 0,445563987022207

SumOfOutput : 170,213796637178

OutputStdDev : 0,000974022259280432

Multilayer Perceptron with 0,99 Momentum Rate, 1000 Epoch, 4,31712474106579E-07 Training Coefficient, 30 Input Neurons, 3 Hidden Neurons

6 249

MeanError : -0,518817252517106

SumOfSquareError : 556,787543860864

SumOfSquareErrorMean : 2,18348056416025

SumOfError : -132,298399391862

ErrorStdDev : 1,38358563979641

MeanValue : 0,107339734532049

SumOfSquareValue : 491,128757137718

SumOfSquareValueMean : 1,92599512603026

SumOfValue : 27,3716323056724

ValueStdDev : 1,38364493545882

MeanOutput : 0,626156987049154

SumOfSquareOutput : 99,9789666380202

SumOfSquareOutputMean : 0,392074378972628

SumOfOutput : 159,670031697534

OutputStdDev : 0,00134407669184965

Multilayer Perceptron with 0,99 Momentum Rate, 1000 Epoch, 4,31712474106579E-07 Training Coefficient, 30 Input Neurons, 4 Hidden Neurons

5 250

MeanError : -0,494088554645274

SumOfSquareError : 550,390666161174

SumOfSquareErrorMean : 2,1583947692595

SumOfError : -125,992581434545

ErrorStdDev : 1,38357192419767

MeanValue : 0,107339734532049

SumOfSquareValue : 491,128757137718

SumOfSquareValueMean : 1,92599512603026

SumOfValue : 27,3716323056724

ValueStdDev : 1,38364493545882

MeanOutput : 0,601428289177323

SumOfSquareOutput : 92,238278691939

SumOfSquareOutputMean : 0,361718739968388

SumOfOutput : 153,364213740217

OutputStdDev : 0,0016592002971948

Multilayer Perceptron with 0,99 Momentum Rate, 1000 Epoch, 4,31712474106579E-07 Training Coefficient, 30 Input Neurons, 5 Hidden Neurons

6 249

MeanError : -0,47766067976543

SumOfSquareError : 546,311427150176

SumOfSquareErrorMean : 2,1423977535301

SumOfError : -121,803473340185

ErrorStdDev : 1,38355991143721

MeanValue : 0,107339734532049

SumOfSquareValue : 491,128757137718

SumOfSquareValueMean : 1,92599512603026

SumOfValue : 27,3716323056724

ValueStdDev : 1,38364493545882

MeanOutput : 0,585000414297479

SumOfSquareOutput : 87,268454902992

SumOfSquareOutputMean : 0,342229234913694

SumOfOutput : 149,175105645857

OutputStdDev : 0,00193653956126269

Multilayer Perceptron with 0,99 Momentum Rate, 1000 Epoch, 4,31712474106579E-07 Training Coefficient, 30 Input Neurons, 6 Hidden Neurons

7 248

MeanError : -0,465970912223553

SumOfSquareError : 543,490966368776

SumOfSquareErrorMean : 2,13133712301481

SumOfError : -118,822582617006

ErrorStdDev : 1,38354914331814

MeanValue : 0,107339734532049

SumOfSquareValue : 491,128757137718

SumOfSquareValueMean : 1,92599512603026

SumOfValue : 27,3716323056724

ValueStdDev : 1,38364493545882

MeanOutput : 0,573310646755602

SumOfSquareOutput : 83,8159186952456

SumOfSquareOutputMean : 0,328689877236257

SumOfOutput : 146,194214922678

OutputStdDev : 0,00218621886621996

Multilayer Perceptron with 0,99 Momentum Rate, 1000 Epoch, 4,31712474106579E-07 Training Coefficient, 30 Input Neurons, 7 Hidden Neurons

7 248

MeanError : -0,457239075476423

SumOfSquareError : 541,428405104195

SumOfSquareErrorMean : 2,12324864746743

SumOfError : -116,595964246488

ErrorStdDev : 1,38353932915725

MeanValue : 0,107339734532049

SumOfSquareValue : 491,128757137718

SumOfSquareValueMean : 1,92599512603026

SumOfValue : 27,3716323056724

ValueStdDev : 1,38364493545882

MeanOutput : 0,564578810008472

SumOfSquareOutput : 81,2825411561096

SumOfSquareOutputMean : 0,318755063357292

SumOfOutput : 143,96759655216

OutputStdDev : 0,00241467320988687

Multilayer Perceptron with 0,99 Momentum Rate, 1000 Epoch, 4,31712474106579E-07 Training Coefficient, 30 Input Neurons, 8 Hidden Neurons

7 248

MeanError : -0,450476847673017

SumOfSquareError : 539,856778556855

SumOfSquareErrorMean : 2,11708540610531

SumOfError : -114,871596156619

ErrorStdDev : 1,38353027282235

MeanValue : 0,107339734532049

SumOfSquareValue : 491,128757137718

SumOfSquareValueMean : 1,92599512603026

SumOfValue : 27,3716323056724

ValueStdDev : 1,38364493545882

MeanOutput : 0,557816582205065

SumOfSquareOutput : 79,3473903266411

SumOfSquareOutputMean : 0,311166236575063

SumOfOutput : 142,243228462292

OutputStdDev : 0,00262625058267343

Multilayer Perceptron with 0,99 Momentum Rate, 1000 Epoch, 4,31712474106579E-07 Training Coefficient, 30 Input Neurons, 9 Hidden Neurons

6 249

MeanError : -0,445091644096824

SumOfSquareError : 538,621006358931

SumOfSquareErrorMean : 2,11223924062326

SumOfError : -113,49836924469

ErrorStdDev : 1,38352183538188

MeanValue : 0,107339734532049

SumOfSquareValue : 491,128757137718

SumOfSquareValueMean : 1,92599512603026

SumOfValue : 27,3716323056724

ValueStdDev : 1,38364493545882

MeanOutput : 0,552431378628872

SumOfSquareOutput : 77,8230428323886

SumOfSquareOutputMean : 0,305188403264269

SumOfOutput : 140,870001550362

OutputStdDev : 0,00282403443182171

Multilayer Perceptron with 0,99 Momentum Rate, 1000 Epoch, 4,31712474106579E-07 Training Coefficient, 30 Input Neurons, 10 Hidden Neurons

6 249

MeanError : -0,440706632390743

SumOfSquareError : 537,624937328541

SumOfSquareErrorMean : 2,10833308756291

SumOfError : -112,380191259639

ErrorStdDev : 1,38351391454142

MeanValue : 0,107339734532049

SumOfSquareValue : 491,128757137718

SumOfSquareValueMean : 1,92599512603026

SumOfValue : 27,3716323056724

ValueStdDev : 1,38364493545882

MeanOutput : 0,548046366922792

SumOfSquareOutput : 76,5927899577

SumOfSquareOutputMean : 0,300363882187059

SumOfOutput : 139,751823565312

OutputStdDev : 0,00301029729226025

# CONCLUSIONS

The dataset was very difficult. MLP couldn’t converge. Eventhough we standardized the data, MLP weights couldn’t find a way to converge.

All weights göne infinity after few epocs. So I standardized all weights as Sum-Of-Weights-are-1, after every epoch I reassigned new values so weights couldn’t go infinity.

But despite every counter measure errors are still very high compared to output. E.g with 10 neuron average output is 0.1 but average output is 0.5. Its 5 times than expected.

I also used momentum but it didn’t changed the result significantly.

You can see there are some numbers right after experiment parameters (like 6-249). Its my basic measure other than error. 6 means correctly predicted and 249 means false predicted. I calculate it with this formula :

if (Math.Abs(output) <= Math.Abs(ForestFireTrain.area) \* 1.2 && Math.Abs(output) >= Math.Abs(ForestFireTrain.area) \* 0.8)

CorrectCount++;

else

FalseCount++;

Finally the data didn’t converged and didn’t fit the formulation. And results was still meaningless even for 10000 epocs.

# REFERENCES

1. [Cortez and Morais, 2007] P. Cortez and A. Morais. A Data Mining Approach to Predict Forest Fires using Meteorological Data. In J. Neves, M. F. Santos and J. Machado Eds., New Trends in Artificial Intelligence, Proceedings of the 13th EPIA 2007 - Portuguese Conference on Artificial Intelligence, December, Guimarães, Portugal, pp. 512-523, 2007. APPIA, ISBN-13 978-989-95618-0-9. Available at: [[Web Link](http://www.dsi.uminho.pt/~pcortez/fires.pdf)]