МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
КАФЕДРА КОМП'ЮТЕРНО-ІНТЕГРОВАНИХ КОМПЛЕКСІВ

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри
Віктор СИНЄГЛАЗОВ
"_____"_____2022 р.

КВАЛІФІКАЦІЙНА РОБОТА
(ПОЯСНЮВАЛЬНА ЗАПИСКА)

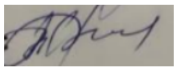
ВИПУСКНИКА ОСВІТНЬО-КВАЛІФІКАЦІЙНОГО РІВНЯ
"МАГІСТР"

Тема: Ансамблевий класифікатор на основі бустінгу

Виконавець: **Богдан ПЛОДИСТИЙ**

Керівник:  **д.т.н., професор Синєглазов В.М.**

Консультант з екологічної безпеки:  **к.т.н., доцент Явнюк А.А.**

Консультант з охорони праці:  **старший викладач Козлітін О.О.**

Нормоконтролер:  **к.т.н., професор Філяшкін М.**

Київ 2022

EDUCATION AND SCIENCE MINISTRY OF UKRAINE
NATIONAL AVIATION UNIVERSITY
DEPARTMENT OF COMPUTER INTEGRATED COMPLEXES

ADMIT TO DEFENSE
Head of department
Viktor SINEGLAZOV
“____” _____ 2022

QUALIFICATION WORK
(EXPLANATORY NOTE)

GRADUATE OF EDUCATION AND QUALIFICATION LEVEL
“MASTER”

THEME: Ensemble Classifier Based on Boosting

Executor: **Bogdan PLODISTYI**

Supervisor: **D.t.s., Professor Sineglazov V.M.**

Advisor on environmental protectio **Ph.D., Associate Professor Iavniuk A.A.**

Advisor on labor protection: **Senior Lecturer Kozlitin O.O.**

Norms inspector: **Ph.D., Professor Filyashkin M.K.**

Kyiv 2022

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет аеронавігації, електроніки та телекомунікацій
Кафедра авіаційних комп'ютерно - інтегрованих комплексів

Освітній ступінь: магістр

Спеціальність 151 "Автоматизація та комп'ютерно-інтегровані технології"

Освітньо-професійна програма "Комп'ютерно-інтегровані технологічні процеси і виробництва"

ЗАТВЕРДЖУЮ

Завідувач кафедри

Віктор СИНЄГЛАЗОВ

"____" _____2022 р.

ЗАВДАННЯ
на виконання дипломної роботи студента

Плодистого Богдана Олександровича

1. Тема роботи: Ансамблевий класифікатор на основі бустінгу.
2. Термін виконання роботи: з 19.08.2022р. до 15.11.2022р.
3. Вихідні дані до проекту (роботи): тип задачі – класифікація, тип навчання -напівкероване, структура нейронної мережі - ансамблева, бібліотеки: Tensor Flow, PyTorch та NumPy.
4. Зміст пояснювальної записки (перелік питань, що підлягають розробці): 1. Аналіз нейронних мереж з класифікатором на основі бустингу. 2. Структура та алгоритмізація глибокого напівсамостійного навчання з частковим залученням вчителя. 3. Структурно параметричний синтез ансамблю вибірки. 4. Проектування та комбінування алгоритмів XGboost та AdaBoost. 5. Побудова ансамблю та розробка інтерфейсу нейронної мережі з бустінгом. 6. Розробка метрики та аналізу результатів класифікатору.
5. Перелік обов'язкового графічного матеріалу: 1. Графіки алгоритмів ансамблю. 2. Відображення вибірок та статистичні таблиці. 3. Графіки точності та втрат ансамблю нейронної мережі.

6. Календарний план-графік

№ п/п	Завдання	Термін виконання	Відмітка про виконання
1	Формування мети та основних завдань побудови ансамблю	09.08.2022-26.08.2022	
2	Аналіз методів для бустінгу в нейронних мережах	26.08.2022-02.09.2022	
3	Теоретичний розгляд вирішення задачі напів самостійного навчання	02.09.2022-16.09.2022	
4	Розробка побудови блок-схеми алгоритмів з використанням AdaBoost	16.09.2022-23.09.2022	
5	Дослідження лінії управління безпілотним літальним апаратом	23.09.2022-07.10.2022	
6	Розробка програмного та підтримка апаратного забезпечення на базі мови програмування Python	07.10.2022-21.10.2022	
7	Впровадження метрики графічного інтерфейсу классифікації	21.10.2022-04.11.2022	
8	Підсумки роботи та підготовка презентації та программного матеріалу	04.11.2022-15.11.2022	

7. Консультанти зі спеціальних розділів

Розділ	Консультант (посада, П. І. Б.)	Дата, підпис	
		Завдання видав	Завдання прийняв
Охорона праці	Старший викладач, Олексій КОЗЛІТІН,		
Охорона навколишнього середовища	к.б.н., доцент, Андріан ЯВНЮК		

8. Дата видачі завдання _____

Керівник: _____ Віктор СИНЄГЛАЗОВ

(підпис)

Завдання прийняв до виконання: _____ Богдан ПЛОДИСТИЙ

(підпис)

NATIONAL AVIATION UNIVERSITY

Faculty of aeronavigation, electronics and telecommunications

Department of Aviation Computer Integrated Complexes

Educational level: master

Specialty 151 " Automation and computer-integrated technologies"

Educational and professional program "Computer-integrated technological processes and production"

APPROVED BY

Head of department

Victor SINEGLAZOV

"____" _____ 2022

Graduate Student's Diploma Thesis Assignment

Plodistyi Bohdan Oleksandrovych

1. The thesis title: Ensemble Classifier Based on Boosting.
2. The thesis to be complete between: from 19.08.2022 to 15.11.2022.
3. **Output data for the thesis:** software packages and Python 3.0 programming language, classifier based on Tensor Flow, PyTorch and NumPy libraries, neural network structural diagram, Weka application.
4. **The content of the explanatory note (the list of problems to be considered):** 1. Analysis of neural networks with boosting based classifier. 2. Structure and algorithmization of deep semi-supervised learning with partial teacher involvement. 3. Structurally parametric synthesis of ensemble sampling. 4. Design and combination of XGboost and AdaBoost algorithms. 5. Ensemble construction and development of neural network interface with boosting. 6. Development of metrics and analysis of classifier results.
5. List of compulsory graphic material: 1. Graphs of ensemble algorithms. 2. display of samples and statistical tables. 3. Graphs of accuracy and losses of the ensemble neural network.

6. Planned schedule:

№	Task	Execution term	Execution mark
1	Analysis of the relevance of the problem	09.08.2022-26.08.2022	
2	Analysis of characteristics of unmanned aerial vehicles and their application	26.08.2022-02.09.2022	
3	Research of information support of monitoring systems by unmanned aerial vehicles	02.09.2022-16.09.2022	
4	Research of monitoring and control subsystems of the ground control station	16.09.2022-23.09.2022	
5	Research of the unmanned aerial vehicle control line	23.09.2022-07.10.2022	
6	Development and research of the monitoring and control subsystem of the unmanned aerial vehicle	07.10.2022-21.10.2022	
7	Development of graphical user interface (operator)	21.10.2022-04.11.2022	
8	Conclusions on the work and preparation of presentation and handouts	04.11.2022-15.11.2022	

7. Special chapters' advisors

Chapter	Advisor (position, name)	Date, signature	
		Assignment issue date	Assignment accepted
Labor protection	Senior lecturer, Oleksiy KOZLITIN		
Environmental protection	Ph.D, Associate Professor, Andrian IANIUK		

8. Date of task receiving: _____

Diploma thesis supervisor: _____ Victor SINEGLAZOV

(signature)

Issued task accepted: _____ Bohdan PLODISTYI

(signature)

АНОТАЦІЯ

Пояснювальна записка до дипломної роботи «Ансамблевий классифікатор на основі бустінгу»: _____ с., ___ рис., _____ табл.

Ключові слова: АДА БУСТ, КОНТРОЛЬОВАНЕ НАВЧАННЯ, НАПІВКОНТРОЛЬОВАНЕ НАВЧАННЯ, АНСАМБЛЬ, БУСТІНГ, КЛАСИФІКАТОР, ПРОГРАМНИЙ ІНТЕРФЕЙС

Предмет дослідження - структурно-параметричний синтез ансамблю нейронних мереж з частковим залученням вчителя.

Мета роботи - отримання метрик та результатів вибірки ансамблевого підходу при побудові класифікаторів на основі використання нейронних мереж.

Метод дослідження - класифікація підходів та методів навчання на основі бустінгу, Ада Бусту з частковим залученням вчителя. Розробка програмного забезпечення з підключенням метрик. Включивши пари алгоритмів для побудови мічених та немічених вибірок, буде отримано набір результатів з додатковою інформацією для кожної нової вибірки та бустінгового методу класифікації.

За допомогою рішення та результатів алгоритму можна прискорити та вдосконалити завдання програмування з великим стеком вхідних даних. Це дозволить отримати унікальний швидкодіючий метод з різними типами параметрів.

Проект розробки включає нові пакети даних та бібліотеки для візуалізації програмної реалізації вибірки даних.

ANNOTATION

Explanatory note to the diploma work «Ensemble Classifier Based on Boosting»: _____ pages., ___ figure., _____ table.

Keywords: SSL, ADA BOOST, SUPERVISED, SEMI-SUPERVISED; ENSEMBLE, BOOSTING, CLASSIFICATOR, PROGRAMMING INTERFACE

The subject of research - Structural parametric synthesis of an ensemble of neural networks with partial teacher involvement.

The purpose of the work is to obtain the metrics and results of the ensemble approach sampling in the construction of classifiers based on the use of neural networks.

Research method - classification of approaches and methods of teaching based on boosting, Ada Boost with partial consideration of the teacher. Software development with metrics connection. By including pairs of algorithms to construct labeled and unlabeled samples, a set of results will be obtained with additional information for each new set and boosting classification method.

With solution and results algorithm can boost and improve programming tasks with large stack of data inputs. This can obtain a unique quick-stage method with different type of parameters.

The development project includes the latest data packages and libraries for better software visualization implementation of the sample data.

CONTENT

GLOSSARY

SSL – Semi Supervised Learning

UL – Unsupervised Learning

CNN – Convolutional Neural Network

AB – Ada Boost

AI – Artificial Intelligence

ML – Machine Learning

RPA – Robot-assisted Process Automation

LR – Logistic Regression

NN – Neural Network

SVM – Support Vector Machines

RF – Random Forest

KF – Kernel Factory

RNN – Residual Neural Network

SAB – Stochastic Adaptive Boosting

SQL – Structured Query Language

SSMB – Semi-Supervised Margin Boost

TGDS – Three-Gaussian Data Set

GB – Gradient Boost

ANN – Artificial Neural Network

EM - Expectation Maximization

TSVM - Transductive Support Vector Machines

PU – Positive and Unlabeled learning

TSNE - Distributed Stochastic Neighbor Embedding

INTRODUCTION

This paper considers the construction of a classifier based on neural networks, nowadays AI is a major global trend, as an element of AI, as a rule, an artificial neural network is used. One of the main tasks that solves the neural network is the problem of classification. For a neural network to become a tool, it must be trained. To train a neural network you must use a training sample. Since the marked training sample is expensive, the work uses semi-supervised learning, to solve the problem we use ensemble approach based on boosting.

Speaking of unlabeled data, we can move on to the topic of semi-supervised learning (SSL). This is due to the need to process hard-to-access, limited data. Despite many problems, the first algorithms with similar structures have proven successful on a number of basic tasks in applications, conducting functional testing experiments in AI testing. There are enough variations to choose marking, where training takes place on a different set of information, the possible validation eliminates the need for robust method comparison. Typical areas where this occurs are speech processing (due to slow transcription), text categorization.

Choosing labeled and unlabeled data to improve computational power leads to the conclusion that semi-supervised learning can be better than teacher-assisted learning. Also, it can be on an equal efficiency factor as supervised learning.

Neural networks represent global trends in the fields of language search, machine vision with great cost and efficiency. The use of "Hyper automation" allows the necessary tasks to be processed to introduce speedy and simplified task execution. Big data involves the introduction of multi-threading, something that large companies in the artificial intelligence industry are doing.

Data analysis is actively used in building applications, engineering as well as manufacturing architecture. This is done to learn certain actions that will help improve the profitability of the company in the future. Specialists include a couple of key areas of relevance between now and 2023:

- Robot-assisted process automation (RPA)

- Artificial intelligence and machine learning (AI\ML)

- Cognitive research in process automation.

- Using processes for point-to-point control in software (iBPMS)

The above-mentioned methods are actively used in pairs, as it can significantly improve the tasks of productivity, data processing, obtaining more accurate information. Machine learning solves many different problems: saving resources, speeding up decision-making for different areas of business, increased demand for innovation, such as self-managing bots.

Machine learning is also being implemented for smart data warehouses, information security, implying the use of mathematical models to support learning in various ways. The task of prediction is also opening up in new professions, such as neuro-design.

It is worth highlighting the current directions for building architectures, five directions can be taken as a basis:

- The Perceptron, or network with multiple layers, which is a set of transformations.

- Allocation of neural network memory for taking an arbitrary piece of data, especially recurrent networks.

- ResNet networks with accelerated access, which is in fact a simplified and improved version of a convolutional neural network.

- Artificial Intelligence with convolution is used to partially identify an overall picture or object, connecting it to the adjacent part speeding up processing.

CHAPTER 1. NEURAL NETWORKS AND THEIR CHARACTERISTICS.

1.1 Neural network structure.

The main prospect of the neural network is the ability to self-learn depending on the conditions, as a result of which it is possible to improve the performance. If we have an idealized case, then with each iterative pass the output algorithm should receive the necessary amount of knowledge from the environment in which it works.

The learning process is usually divided into many structures, it all depends on the chosen algorithm and learning stages. In this paper we investigate the accuracy with combining variations of the neural network.

Each self-learning unit has its own parameters that are rebuilt by simulating the environment in which the network is included. It is the way of building the parameters that determines how the training will be carried out.

We take into account the following standard for the sequence of training organization:

- The neural network includes "stimuli" from the external environment;
- Free parameters of the neural network can change as a result of this;
- When modifying the external structure, the neural network can respond to stimuli in a different way.

At the heart of learning, there is no unique algorithm for learning. By creating it and getting a certain set of data and parameters, you can highlight a certain advantage and goal with the results of work.

There are 4 main structures for training:

- Error correction based;
- competitive learning and Boltzmann learning;

- Hebbian learning;
- When using memory.

Construction of a neuron on the basis of k-unique computable node:

At the output node of the neural fence, you can pass the first rule that uses learning. This unit k, is triggered when controlling the signal vector y(n). Depending on the number of hidden neuron zones, data from the input vector will be obtained, which in turn are transmitted to the initial nodes. Discrete time is defined as n, or it is the ordinal number of the iterative pass of the process of setting the synaptic weight of neuron k.

To equalize the results of the expected output $d_k(n)$, it is necessary to obtain the output signal k, which in the final case is $y_k(n)$.

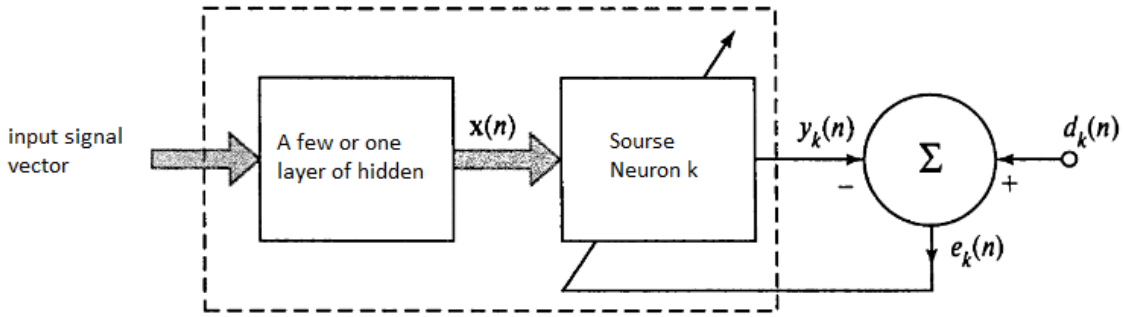


Figure 1.1 Block diagram for the output node of the neural network

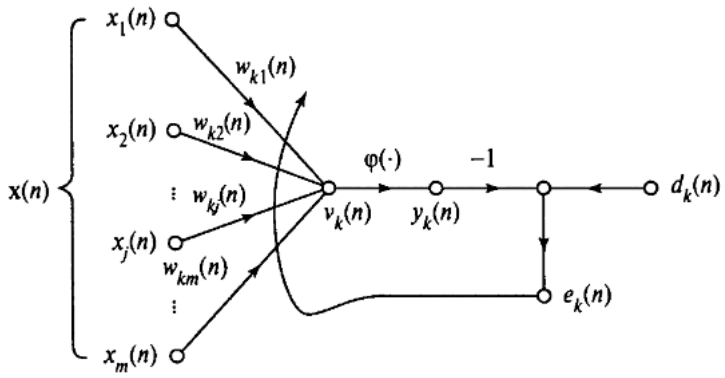


Figure 1.2 Illustration of the neuron signal transmission graph

In the case of memory-based learning, there is a need to store data in a repository where the examples for learning are properly located.

{(Yi, Di)}n, i=1

Yi is the input vector

Di - the corresponding signal at the output

Moreover, you can take a binary distribution for recognition or classification into several classes. From the basics of binarity, the variable outcomes can take the value 0 or -1 for the first class, and 0 for the last.

There are two basic rules for constructing this type of learning:

- Mutual inclusion of the parameter, to find the outer zone of the vector.

- In case of finding this zone of the trial vector, use the learning rule.

For the simplest case with memory-based use, the NNR rule is set - Nearest Neighbor Rule. The example that is closest to the test one is added to the location zone.

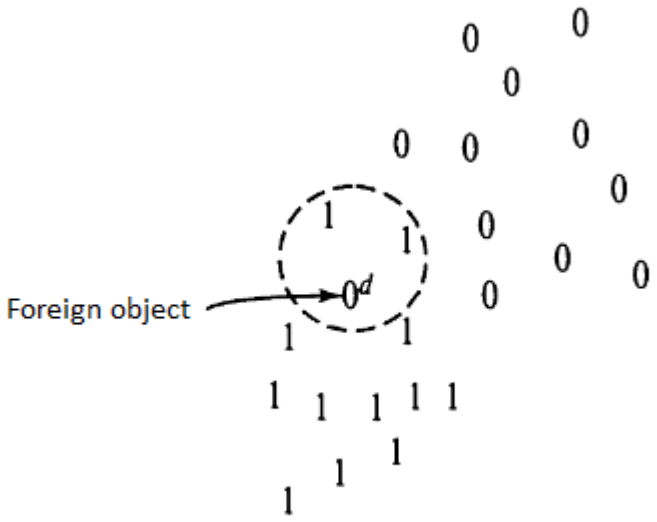


Figure 1.3 Graphical representation of the region with values 0, 1 and the trial vector d.

Hebb's doctrine can be stated as a certain axiom:

- If one point is permanently, or temporarily, close enough to another point to cause it to be excited, after a certain period of time, a modification of both or one point can be seen.

Knowing this, we can assume constant training based on this rule.

Below is a table 1.1 of Hebb's synapse:

Qualities	Meanings
Dependence on the time period mechanism	• Depending on the type of signal, you can find out the accuracy of the time difference of the signal, namely pre and post synapses.
Local mechanism	• Based on the case of a node of data that is gradually transmitted, we can assume that all elements of the system are in space-time proximity.
Interactivity of the mechanism	• By examining the limbs of the synapse, Hebb signals can be determined. Depending on the species, interactivity can be either static or deterministic.
Correlational mechanism	• As a condition for modifying this connection, to improve the algorithm, it is necessary to have the same response time on both signals

1.2 Classification of neural network training methods

Human vision is an active process by which we sequentially scan the optical environment. It is an intelligent, task-specific matrix that uses a small area with a large, low-resolution environment. We expect that future advances in image processing will come from systems that learn from one device to another and combines convolutional networks and SNNs that use reinforcement learning to decide where to look. To decide where to look. Systems combining deep learning and reinforcement learning are still in their infancy.

Systems using reinforcement learning are still in their infancy, but are already outperforming passive vision systems. Passive vision systems show impressive results in classification tasks leads to the ability to play a wide range of video games.

Natural language understanding is another area where deep learning will have a big impact in the coming years. This is another area where deep learning will have a major impact in the coming years. Expected systems using SNR to understand sentences or entire documents in the future will improve in the future if an electoral strategy is used paying attention one piece at a time.

Ultimately, there will be significant advances in artificial intelligence systems that combine representational learning:

- It enables rule-based manipulation of symbolic information.

- Have expressions through operations on large vectors.

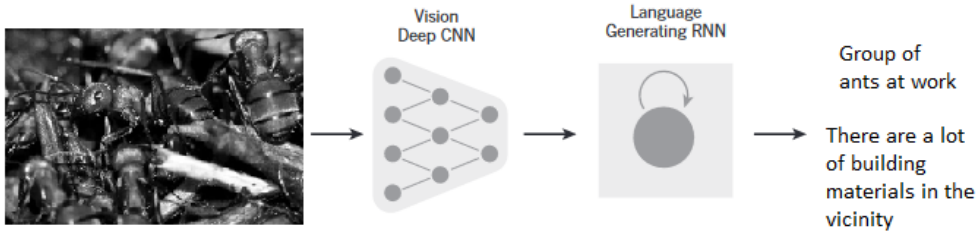


Figure 1.4 Neural network recognition process with logic output

One of the most popular strategies for improving classification performance is the use of data variety. Even if a particular learner performs better than other learners on many tasks.

In the case of the Random Forest, you still get:

- Algorithms for each new task. No algorithm is optimal for all possible data sets, so there is no such thing as a free lunch. Strategy

- Evaluating more than one algorithm and selecting the best performing one is called Single Best.

- It is one of the simplest but most reliable strategies and is therefore used industrially.

- Referenced strategies. However, it should be noted that Single Best is much more complex than simply selecting an algorithm.

- Implementing a data-driven ensemble technique. To be able to do this, a high level of experience is required to implement specific algorithms correctly.

The complexity of Single Best is increased by the fact that expert knowledge is required for all algorithms in the benchmark. Despite the difficulties associated with implementing the Single Best method, it is the preferred option for improving classification performance in industrial applications.

While the focus of this paper is on diversity generation mechanisms, an important part of ensemble design is the combination rule. In the literature, there are various fusion methods that depend on the results of the base classifiers. While there is a kind of voting is the most widely used method in classifier fusion, but it may not always be the best option. Since only the output of class labels is used, a lot of information is lost.

Confidences or posteriori probabilities contain the largest amount of information and reduces the generalization error. Therefore, in our hybrid design, we calibrate all our models to ensure that we have a measurement-level output for each base classifier. As a result, we can combine our classifiers using a weighted average. In this study, we restrict ourselves to linear combinations as we do not need to estimate the number of parameters there are a number of factors that need to be estimated to make the analysis manageable.

For the calculation of weights, we can resort to fixed rules or trained weights. While fixed rules offer very low time complexity and simplicity, their results are likely to be worse than trained rules. The simplest way to combine different metrics is to take the simple average. The performance of this simple method is usually close to that of the weighted average, but in many cases the weighted average can outperform the simple average.

Simple average in performance or authority-based weighting, the weight of each classifier is proportional to its performance on a validation set. This method usually produces better results, but at the cost of more computational effort.

1.3. Semi-supervised learning of neural networks

Most of the generated data is uncategorized or unlabeled, thereby making it difficult to use supervised approaches to automate applications like personal news filtering, email spam filtering, and document and image classification. Typically,

there is only a small amount of labeled data available, for example, based on which articles a user mark interesting, or which email he marks as spam, but there is a huge amount of data that has not been marked. As a result, there is an immense need for algorithms that can utilize the small amount of labeled data, combined with the large amount of unlabeled data to build efficient classification systems.

Existing semi-supervised classification algorithms may be classified into two categories based on their underlying assumptions. An algorithm is said to satisfy the manifold assumption if it utilizes the fact that the data lie on a low-

dimensional manifold in the input space. Usually, the underlying geometry of the data is captured by representing the data as a graph, with samples as the vertices, and the pairwise similarities between the samples as edge-weights.

Usually, machine learning has been divided into two categories. Into rules of supervised and unsupervised learning. Supervised learning finds a rule that can be used to predict the relationship between inputs and outputs. Finite instances in the form of input-output pairs, unsupervised learning in finding a structure of interest as the basis for a data set.

Mostly, SL has many Examples of training that produces a satisfactory learner. a topic generalization ability. Acquiring training is not trivial for the SL, who should comment on examples of enter the data with the appropriate labels. In many practical applications range from data mining to machine performance.

However, input data annotations are often difficult, costly and time consuming, and in particular have to be done manually by experts. On the other hand, there is often a large amount of undisclosed data available. In order to use undisclosed data, semi-Supervised learning has become the new paradigm combination with a large number of unscored points with a small number of annotated examples to create a better learner. Because SSL requires less human effort, only greater accuracy can be achieved, because it is based on unpublished data largely support SL has attracted the attention of the machine learning community.

So become the main solution obtained and develop an optimization algorithm taking into account that all the technical information needed. First, we present the initial setup initial unlabeled data labeling, training and the first classifier and a nonlinear function producing density estimate. Then, the semi-supervised boosting algorithm for binary classification and explaining this algorithm with an example commonly used component cost function AdaBoost, ASSEMBLE and many other boosting algorithms.

SSL is a learning paradigm associated with constructing models that use both labeled and unlabeled data. SSL methods can improve learning performance by using additional unlabeled instances compared to supervised learning algorithms, which can use only labeled data. It is easy to obtain SSL algorithms by extending supervised learning algorithms or unsupervised learning algorithms. SSL algorithms provide a way to explore the latent patterns from unlabeled examples, alleviating the need for a large number of labels. Depending on the key objective function of the systems, one may have a semi-supervised classification, a semi-supervised clustering, or a semi-supervised regression.

From that, following conditions are true:

- Semi-supervised classification. Given a training dataset that consists of both labeled instances and unlabeled instances, semi-supervised classification

aims to train a classifier from both the labeled and unlabeled data, such that it is better than the supervised classifier trained only on the labeled data.

- Semi-supervised clustering. Given a training dataset that consists of unlabeled instances, and some supervised information about the clusters, the goal of semi-supervised clustering is to obtain better clustering than the clustering from unlabeled data alone. Semi-supervised clustering is also known as constrained clustering.

- Semi-supervised regression. Given a training dataset that consists of both labeled instances and unlabeled instances, the goal of semi-supervised regression is to improve the performance of a regression algorithm from a regression algorithm with labeled data alone, which predicts a real-valued output instead of a class label.

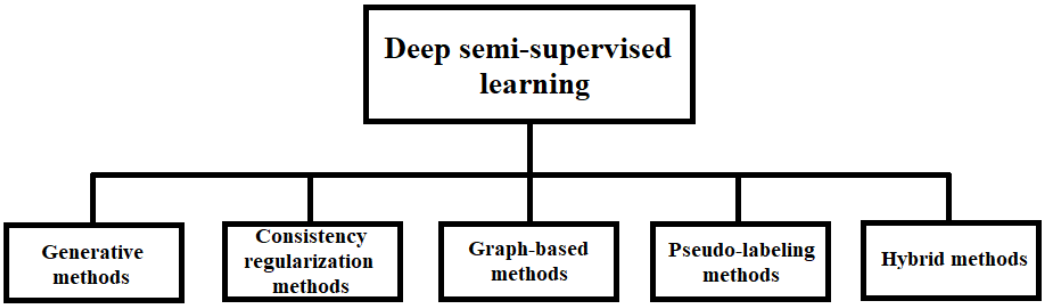


Figure 1.5 taxonomy of deep semi-supervised learning methods based on loss function and model design.

Looking at the classical SSL methods, generative models assume a model $p(x,y) = p(y)p(x|y)$, where the density function $p(x|y)$ is an identifiable distribution, for example, polynomial, Gaussian mixture distribution, etc., and the uncertainty is the parameters of $p(x|y)$. Generative models can be optimized by using iterative algorithms. This applies EM algorithm for classification. They compute the parameters of $p(x|y)$ and then classify unlabeled instances according to the Bayesian full probability formula. Moreover, generative models are harsh

on some assumptions. Once the hypothetical p(x|y) is poorly matched with the actual distribution, it can lead to classifier performance degradation.

A representative example following the low-density separation principle is Transductive Support Vector Machines (TSVMs). As regular SVMs, TSVMs optimize the gap between decision boundaries and data points, and then expand this gap based on the distance from unlabeled data to the decision margin. To address the corresponding non-convex optimization problem, a number of optimization algorithms have been proposed. For instance, a smooth loss function substitutes the hinge loss of the TSVM, and for the decision boundary in a low-density space, a gradient descent technique may be used.

Graph-based methods rely on the geometry of the data induced by both labeled and unlabeled examples. This geometry is represented by an empirical graph G = (V, E), where nodes V represent the training data points with

|V| = n and edges E represent similarities between the points. By exploiting the graph or manifold structure of data, it is possible to learn with very few labels to propagate information. For example, Label propagation is to predict the label

information of unlabeled nodes from labeled nodes. Each node label propagates to its neighbors according to the similarity.

At each step of node propagation, each node updates its label according to its neighbors' label information. In the label propagation label, the label of the labeled data is fixed so that it propagates the label to the unlabeled data. The label propagation method can be applied to deep learning. set in strong supervision. There are three types of weakly supervised data: incomplete supervised data, inexact supervised data, and inaccurate supervised data. Incomplete supervised data means only a subset of training data is labeled. In this case, representative approaches are SSL and domain adaptation. Inexact supervised data suggests that the labels of training examples are coarse-grained, e.g., in the scenario of multi-instance learning. Inaccurate supervised data means that the given labels are not always ground-truth, such as in the situation of label noise learning.

Positive and unlabeled (PU) learning is a variant of positive and negative binary classification, where the training data consists of positive samples and unlabeled samples. Each unlabeled instance can be either the positive and negative class. During the training procedure, only positive samples and unlabeled samples are available. We can think of PU learning as a special case of SSL.

Meta-learning. Meta-learning, also known as “learning to learn”, aims to learn new skills or adapt to new tasks rapidly with previous knowledge and a few trainings example. It is well known that a good machine learning model often requires a large number of samples for training. The meta-learning model is expected to adapt and generalize to new environments that have been encountered during the training process. The adaptation process is essentially a mini learning session that occurs during the test but has limited exposure to new task configurations. Eventually, the adapted model can be trained on various learning tasks and optimized on the distribution of functions, including potentially unseen tasks.

Self-supervised learning. Self-supervised learning has gained popularity due to its ability to prevent the expense of annotating large-scale datasets. It can leverage input data as supervision and use the learned feature representations for many downstream tasks. In this sense, self-supervised learning meets our expectations for efficient learning systems with fewer labels, fewer samples, or fewer trials. Since there is no manual label involved, self-supervised learning can be regarded as a branch of unsupervised learning.

1.4. Assumptions used to build networks with SSL

In the case of SSL, the implicit assumption is the example with the i.e., there are few labeled examples, but many unlabeled points in the training set. Unlabeled points in the training set. Therefore, it is very important it is likely that many unlabeled points have no labeled points at all in their neighborhood. Moreover,

defined in the point, suggests that we should consider the dis in labeling of unlabeled points.

In the clustering tends to be more reliable than in the case of the neighborhood of the unlabeled point in particular as the unlabeled point i is in a region with low data density. Consequently, all density-based clustering algorithms, can be used to group the training data into clusters.

There are some assumptions with explanations notation for the better understanding in the table 1.2 below:

Variable	Meaning
G	Generator
C	Classifier
R	Consistency constraint
g	A graph
X	Input space, for example $X = R^n$
Y	Regression: Y=R, output space classification: $y = \{y^1, y^2, \ldots, y^k\}$.
D_{ii}	The degree of node i
v	A node $v \in V$
X_L	Labeled dataset $x_i \in X$, $y_i \in Y$
X_U	Unlabeled dataset $x_i \in X$
L	Loss function
E	Expectation
H	Entropy

Z	Embedding matrix
A	The adjacency matrix of a graph
W	The weight matrix
W_{ij}	The weight associated with edge e_{ij}
D	The degree matrix of a graph
D_{ii}	The degree of node i
V	The set of vertices in a graph
Ɛ	The set of edges in a graph
Zv	An embedding for node v
S	Similarity matrix of a graph
D	Discriminator
$H_v^{(k)}$	Hidden embedding for node v in k_{th} layer
$S_{[u, v]}$	Similarity measurement between node u and v
$N_{(v)}$	The neighbors of a node v
$m_{N(v)}$	Message aggregated from node v's neighborhoods

And then a competition based on kinship would take place within each cluster other than its neighborhood for labeling unlabeled points. Largely also because it solves the sparsity problem of the labeled example because all unlabeled points of a cluster are unlabeled points in a cluster can be labeled because there is a labeled point in the cluster.

SSL aims to predict more accurately with the aid of unlabeled data than supervised learning that uses only labeled data. However, an essential prerequisite is that the data distribution should be under some assumptions. Otherwise, SSL may not improve supervised learning and may even degrade the prediction accuracy by misleading inferences.

Following and, the related assumptions in SSL include:

- Self-training assumption;
- Co-training assumption;
- Generative model assumption;
- Cluster assumption;
- Low-density separation;
- Manifold assumption.

Taking into account assumption of self-training, predictions of the self-training model, especially those with high confidence, tend to be correct. We can assume that when the hypothesis is satisfied, those high-confidence predictions are considered to be ground-truth. This can happen when classes form well-separated clusters.

Different reasonable assumptions lead to different combinations of labeled and unlabeled data, and accordingly, different algorithms are designed to take advantage of these combinations. For example, proposed co-training model, which works under the assumptions: instance x has two conditionally independent views, and each view is sufficient for a classification task. It’s the meaning of co-training.

Generally, in generative model it is assumed that data are generated from a mixture of distributions. When the number of mixed components, a prior $p(y)$ and a conditional distribution $p(x|y)$ are correct, data can be assumed to come from the mixed model. This assumption suggests that if the generative model is correct enough, we can establish a valid link between the distribution of unlabeled data and the category labels by $p(x, y) = p(y)p(x|y)$.

In cluster assumption two points x1 and x2 are in the same cluster, they should belong to the same category. This assumption refers to the fact that data in a single class tend to form a cluster, and when the data points can be connected by short curves that do not pass through any low-density regions, they belong to the same class cluster. According to this assumption, the decision boundary should not cross high-density areas but instead lie in low-density regions. Therefore, the learning algorithm can use a large amount of unlabeled data to adjust the classification boundary.

For the low-density separation case - decision boundary should be in a low-density region, not through a high-density area. The low-density separation assumption is closely related to the cluster assumption. We can consider the clustering assumption from another perspective by assuming that the class is separated by areas of low density. Since the decision boundary in a high-density region would cut a cluster into two different classes and within such a part would violate the cluster assumption.

Representing manifold assumption we know, that two points x1 and x2 are located in a local neighborhood in the low-dimensional manifold, they have similar class labels. This assumption reflects the local smoothness of the decision boundary. It is well known that one of the problems of machine learning algorithms is the curse of dimensionality. It is hard to estimate the actual data distribution when volume grows exponentially with the dimensions in high dimensional spaces. If the data lie on a low-dimensional manifold, the learning algorithms can avoid the curse of dimensionality and operate in the corresponding low-dimension space.

1.5 Semi-supervised learning classification of neural network approaches

Semi-supervised learning algorithms use not only the labeled data but also unlabeled data to construct a classifier. The goal of semi-supervised learning is to

use unlabeled instances and combine the information in the unlabeled data with the explicit classification information of labeled data for improving the classification performance. The main issue of semi-supervised learning is how to exploit information from the unlabeled data.

A number of different algorithms for semi-supervised learning have been presented, such as the Expectation Maximization based algorithms self-training, co-training, Transductive Support Vector Machine, Semi-Supervised SVM, graph-based methods, and boosting based semi-supervised learning methods. Self-training is a commonly used method to semi-supervised learning in many domains, such as Natural Language Processing and object detection and recognition. A self-training algorithm is an iterative method for semi-supervised learning, which wraps around a base learner. It uses its own predictions to assign labels to unlabeled data.

The main difficulty in self-training is to find a set of high-confidence predictions of unlabeled data. Although for many domains decision tree classifiers produce good classifiers, they provide poor probability estimates. The reason is that the sample size at the leaves is almost always small, and all instances at a leaf get the same probability. The probability estimate is simply the proportion of the majority class at the leaf of a (pruned) decision tree. A trained decision tree indeed uses the absolute class frequencies of each leaf of the tree as follows:

$$p(k|x) = \frac{K}{N}$$

Then, a set of newly-labeled data, which we call a set of high-confidence predictions, are selected to be added to the training set for the next iterations. The performance of the self-training algorithm strongly depends on the selected newly-labeled data at each iteration of the training procedure.

First, we briefly review the online boosting tracking system, which is based on online boosting for feature selection, and replaced by our proposed online SemiBoost algorithm. The basic idea is to formulate the tracking as a binary

classification problem between the foreground object to be tracked and the local background. where K is the number of instances of the class k out of N instances at a leaf. However, these probabilities are based on very few data points, due to the fragmentation of data over the decision tree.

Assuming that the object is detected in the first frame, the initial classification is performed by taking positive samples of the object and randomly selected negative samples of the background. The tracking loop consists of the following steps. From I to I+1 the classifier is evaluated pixel by pixel in the local environment. The classifier gives a response corresponding to the likelihood ratio.

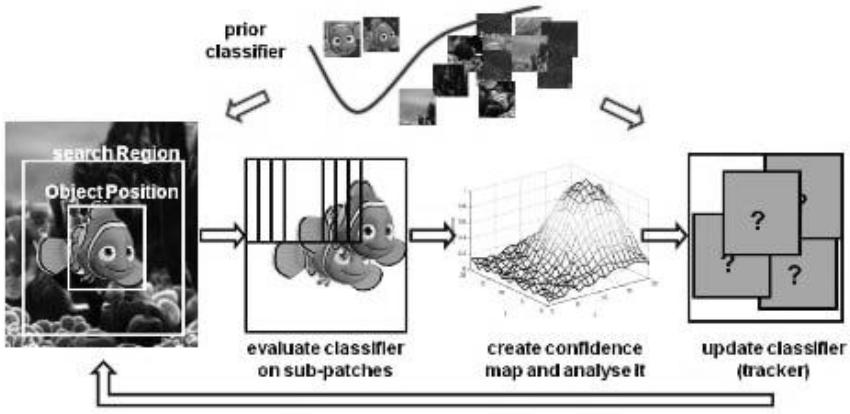


Figure 1.6 Presentation of the object according to a certain time in the tracking process

In short, we extend the analysis of decision trees to ensembles of decision trees. An ensemble combines many possibly weak classifiers, hopefully into a single strong classifier. Ensemble methods differ according to the base learner and the way the classifiers are combined. Examples of ensemble methods are batch method, boosting, random forest method and random subspace method.

In general, self-training is a wrapper algorithm, and is hard to analyze. However, for specific base classifiers, theoretical analysis is feasible, for example showed that the algorithm minimizes an upper-bound on a new definition of cross entropy based on a specific instantiation of the Bregman distance. In this paper, we focus on using a decision tree learner as the base learner in self-training. We

show that improving the probability estimation of the decision trees will improve the performance of a self-training algorithm.

Semi-supervised assumptions (SSAs) hold for the data distribution. As summarized in, there are three fundamental SSAs: semi-supervised smoothness, cluster and manifold assumptions.

The semi-supervised smoothing assumption states that if two points in a high-density region are close, then their corresponding labels should be the same or consistent. The cluster assumption is described as follow: if points are located in the same cluster, they are likely to belong to the same class. In other words, the decision boundary is likely to lie in a low data-density region, which is also referred to as the low density separation assumption. The manifold assumption states that the high-dimensional data lies on a low-dimensional manifold whose properties ensure more accurate density estimate and or more appropriate similarity measures.

To work on the aforementioned SSAs, regularization has been employed in SSL to exploit unlabeled data. A number of regularization methods have beenproposed based on cluster or smoothness assumption, which exploits unlabeled data to regularize the decision boundary and therefore affects the selection of learning hypotheses. Working on cluster or smoothness assumption, most of regularization methods are naturally inductive.

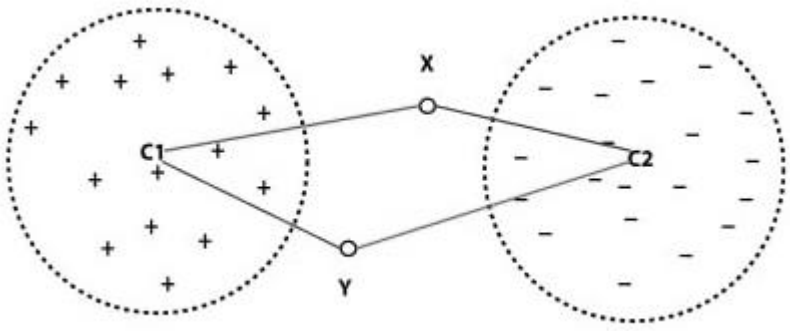


Figure 1.7 Distance of unlabeled examples

On the other hand, the manifold assumption has also been applied for regularization where the geometric structure behind labeled and unlabeled data is explored with a graph-based representation. In such a representation, examples are expressed as the vertices and the pairwise similarity between examples is described as a weighted edge. Thus, graph-based algorithms make good use of the manifold structure to propagate the known label information over the graph for labeling all nodes. In nature, most of such graph-based regularization algorithms are transductive although they can be converted into inductive algorithms with the out-of-sample extension.

As a generic ensemble learning framework, boosting works via sequentially constructing a linear combination of base learners, which appears remarkably successful for SL. Boosting has been extended to SSL with different strategies. Semi-supervised MarginBoost and ASSEMBLE were proposed by introducing the “pseudo-class” or the “pseudo-label” concepts to an unlabeled point so that unlabeled points can be treated as same as labeled examples in the boosting procedure.

In essence, such extensions work in a self-training like style; the unlabeled points are assigned pseudo-class labels based on the constructed ensemble learner so far, and in turn those pseudo-class labels will be used to find a new learner to be added to the ensemble. As pointed out in, such algorithms attempt to minimize both labeled and unlabeled margin cost only. Thus, a hypothesis can be very certain about the classification of unlabeled points with very low margin cost even though these unlabeled points are not classified correctly.

From single decision trees to ensembles of decision trees, in particular the Random Subspace Method and Random Forest. In this case, probability is estimated by combining the predictions of multiple trees. However, if the trees in the ensemble suffer from poor probability estimation, the ensemble learner will not benefit much from self-training on unlabeled data. Using the modified decision tree learners as the base learner for the ensemble will improve the

performance of self-training with the ensemble classifier as the base learner. The results of the experiments on the several benchmark datasets confirm this.

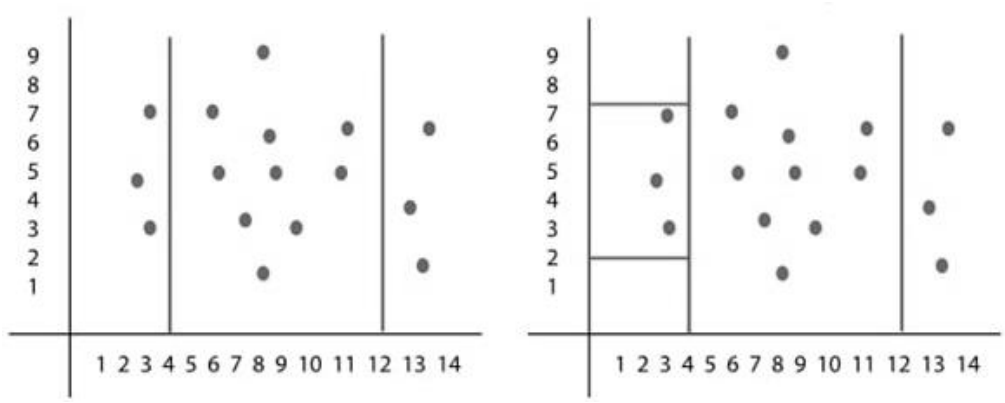


Figure 1.8 Standart and Grafted decision trees examples.

We show that these modifications do not produce better performance when used on the labeled data only, but they do benefit more from the unlabeled data in self-training.

The modifications that we consider are Naive Bayes Tree, a combination of no-pruning and Laplace correction, grafting, and using a distance-based measure. We then extend this improvement to algorithms for ensembles of decision trees and we show that the ensemble learner gives an extra improvement over the adapted decision tree learners.

The performance of the self-training algorithm strongly depends on the selected newly-labeled data at each iteration of the training procedure. This selection strategy is based on confidence in the predictions and therefore it is vital to self-training that the confidence of prediction, which we will call here probability estimation, is measured correctly.

There is a difference between learning algorithms that output a probability distribution, neural networks, logistic regression, margin-based classifiers, and algorithms that are normally seen as only outputting a classification model, like decision trees. Most of the current approaches to self-training utilize the first kind of learning algorithms as the base learner.

Summarizing the first section, it was possible to classify methods with the partial involvement of the teacher, assumptions and understand the basic component of classical methods or ancient teachings. The next section will deal with AdaBoost, MarginBoost methods, and the concept of boosting.

CHAPTER 2. NEURAL NETWORK TRAINING WITH PARTIAL TEACHER INVOLVEMENT.

2.1 Problem statement and data sampling.

The basis for starting to process data in the areas of data assembly, pattern detection and machine learning will require detailed consideration of models and datasets.

The concept of a dataset includes future vectors, among which, each vector is a description of an object using its characteristics. For example, consider a synthetic three-Gaussians dataset.

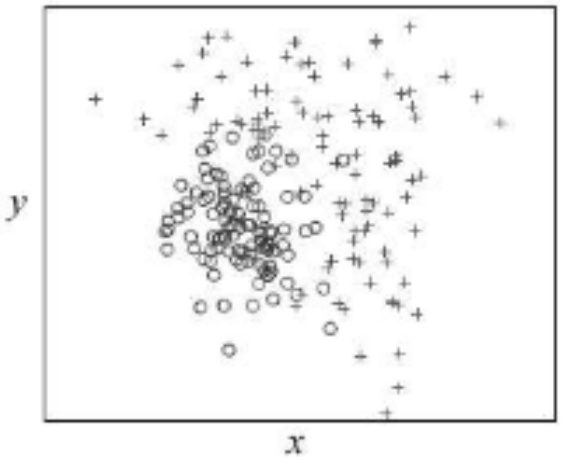


Figure 2.1 The synthetic three-Gaussians data set

The data structure is usually presented in the form of a prediction model, and as a model with a study or dataset construction. For example, a support vector machine, a neural network or a decision tree.

The basic process of model emergence from the data is referred to as learning or training, using a learning algorithm. There are different learning settings through which the most basic are supervised learning and unsupervised learning.

There are different parameters for learning, the most important of which are supervised learning and unsupervised learning. Predicting the value of a target or an undefined instance is the goal in supervised learning. Predictor is a unique name for a learned model.

Crosses and circles need to be marked to define the data assembly and the task of the predictor is to determine the unknown values of the instance.

A similar parsing is called classification with a classifier.

Having numerical values in the data sample, it is possible to redefine the coordinates by regression with a learning model called fitted regression model. The collection of data for training in all cases will be referred to as an example.

In the case of binary classification definitions, finding positive and negative points is used to highlight markers. Unsupervised learning involves some data or input to investigate the internal distribution.

Clustering, is a concept that reveals the achievement of a cluster structure of data points.

The ensemble has a number of advantages in operation:

- With tagged and untagged data, every classification algorithm that has weight membership can be enhanced.

- Using different classes, from two, to multi-classes, it is possible to determine the marginal value involving unspecified data.

- SSMBoost can only be used with a specific step dimension. Assemble, on the other hand, has a matched step-size, each regular student is weighted in the right ensemble supervision methods.

- Speaking for acceleration, unlabelled data with a significant number of classifier pair reductions can be seen.

- It is well used in practice with hardware acceleration, which improves performance and data processing.

- By running experiments with first-pass, one can see the efficiency that can refine a particular ensemble, AdaBoost loses out on performance with the same starting number of pairs of learners.

2.2 Classification of approaches

For comparison with a single learning unit, ensemble methods try to construct a collection of learners with a consequent combination of them. There are a couple of similar definitions for ensemble: commite-based learning, and multiple classifier systems.

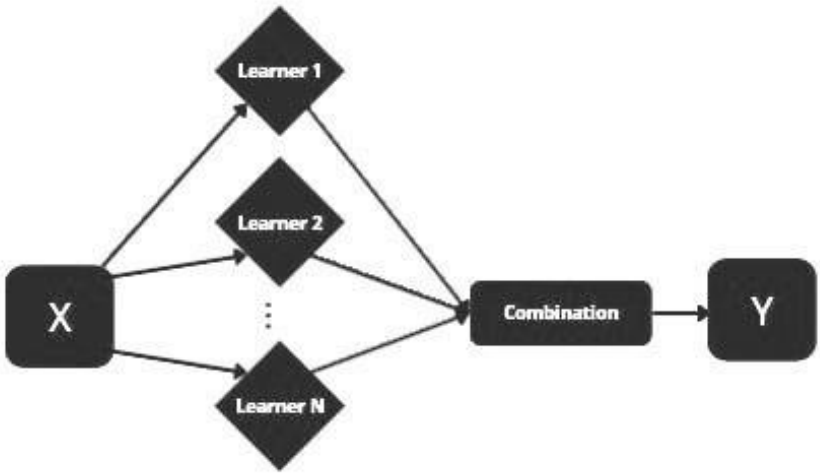


Figure 2.2 Architecture of a common ensemble

There are three key approaches to problem-solving in ensemble methods: ensembles of weak learners, combining classifiers, mixture of experts.

Combining classifiers are usually used to define patterns.

Following machine learning can highlight the acceleration of the data sampling process, in building powerful algorithms to increase performance with the method of ensembles of weak learners. It is from this method that AdaBoost and Bagging were subsequently described.

By expounding on the topic of weak learners we can redefine weak learners into strong learners.

The third method moves into the neural network section. Using combinations and joins of parametric models, groups of rules "mixture of experts" helps to obtain an optimal solution to the problem.

This is shown through errors, noise level, and average in the Hansen and Salamon study.

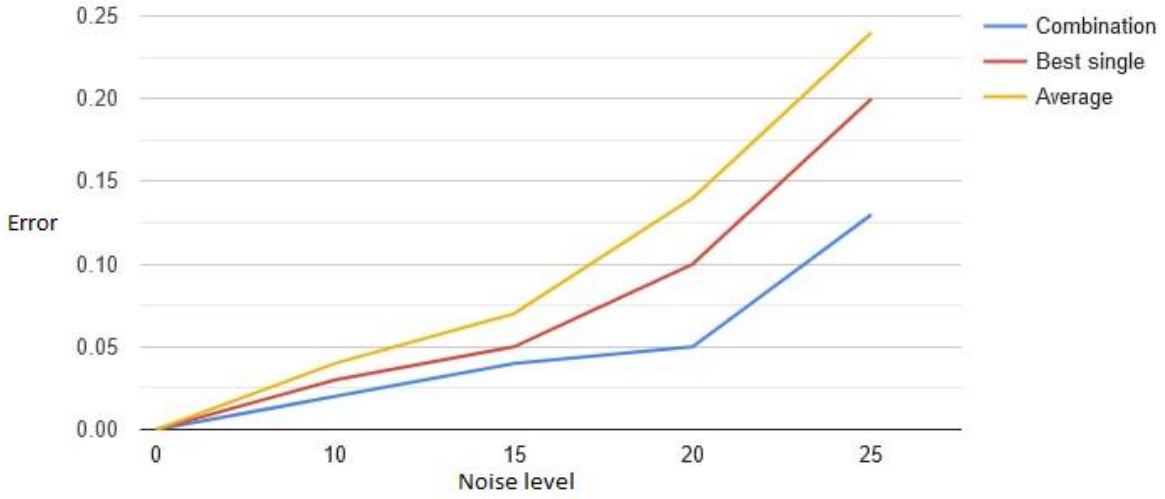


Figure 2.3 Illustration of the Combining method with better results

This method was more accurate than single purpose. Errors and noise level is lower in such case. This makes it clear that there is an advantage in using the combinatorial method.

There are a number of algorithms for solving different problems in an ensemble. Each of them has its pros and cons, the combination includes a number of advantages for boosting tasks.

These algorithms include: AdaBoost, Logistics Regression, Neural Network, Support Vector Machines, Random Forest and Kernel Factory, k-nearest neighbor.

The Lasso approach - is used to define the concept of logistic regression. The scientist attributes a constraint to the sum of the full values of the coefficients. In the output from this the coefficients are reduced to zero. We process the shrinkage value through cross validation. Different data packages are used for this task, e.g., glmnet. The variable α is overridden by a value of one to detect lasso

properties, thus allowing the function to find and compute the sequence λ by setting the value of nlambda to 100.

To create a Random Forest view, two values must be set: the total amount of variables to be processed for each partitioning and the number of trees in the ensemble. Revisiting Breyman's advice and finding the number of values equal to the square root of the allowable value of the predictors and using a large number of trees up to 500. The processing of the Random Forest type of data uses a library of appropriate name.

The first applications of the boosting phenomenon can be seen as an affinity to deterministic weighting. Stochastic boosting, is a boosting that enhances single algorithms by adding randomness as a mandatory part of the process. The two non-negotiable values are the number of terminal connections in the underlying classifiers and the number of iterations.

By setting the maximum number of connections to eight, setting the maximum tree depth to three values, we can achieve appropriate recommendations. In addition, the number of iterations is set to 500. Stochastic boosting can be seen in the implementation using the Ada library.

Support vector machines are commonly used in binary classification with high margin classifiers that select areas of different classes with the required hyperplane at the maximum margin. Minimum distance between two unknown’s parameters to the classification hyperplane.

The purpose of the k-nearest neighbor method is to rule that every input pair of objects is similar to the output pair. This briefly describes lazy learning, without a direct learning process with an easy way to store the set. During the learning process, the condition appears that the test dataset is similar to the dataset close to the learning instance. Then, a preferential class is determined, through k instances. Based on the regression, the test dataset is assigned to the mean of the k instances.

The term boosting was first proposed to define the change of weak learners to strong learners. When the strong learner is close to perfect performance, the weak learner has almost no noticeable advantage at the start.

It is assumed that every weak participant is capable of turning into a strong one. Counting is easy, but getting the desired result with training is difficult. The basic principle of operation does not include difficulty in the process. Initially, positive and negative instances are classified. As an example, weak learners identify a binary distribution problem in the data sample.

At location Y, we can consider training the instances with distribution V and the ground-truth function f. Assuming that the total area is divided into 3 parts Y_1, Y_2, Y_3, each part is equal to 1/3 of the total distribution and randomly assumes 50% of the classification error of the weak learner problem. It is necessary to eliminate as much as possible the errors of the classification problem by having only weak learners in the first two areas and 1/3 of the classification errors in the third area. In this zone we can label the weak classifier as J_1, it is logical that this value is not desirable in the subsequent work.

The main point of the work is to change the errors from the J_1 variable. We can redefine V' from V, for obviousness in the initial blunders from J and focus on the value of Y_3. The next step is to train the classifier J_2 from V. Understanding that J_2 is identically weak in the classifier, we can say that it has incorrect values in zone Y_2 and correct values in zones Y_1 and Y_3. By crossing the values of J_1 and J_2 the resulting classifier will have the correct classifications in zone Y_1, and possibly some errors in the other locations.

To find more evidence in combined classifier errors we need to get a new distribution V" and learn a new classifier j_3 from the distribution. Apply, that zones X_2 and X_3 have correct classifications. Approved, that in a place of $Y_{1,}$ Y_2 and Y_3 we know each of two classifiers to make needed classifications. It is observed that the last learners focus on the errors of the first learners.

Base procedure for boosting will be described in a few steps:

Data on input: sample distribution – V; the basis of learning algorithm – L; quantity of a learning iterations - Q;

1. $V_1 = V$ (for the process of distribution initialization)

2. for t = 1…….Q:

3. $J_t = L(V_t)$; (than we find a weak leaner with training from V_t)

4. $E_t = P_{x\sim Vt}(j_t(x) \neq f(x))$; (find the error of j_t)

5. V_{t+1} = Adding distribution(V_t, E_t)

Output data: H(x) = mixed output ($\{J_1(x),\ldots,J_t(x)\}$)

Based on the experiment done, it can be said that the bussing works with a data set of distributions into zones with a combination of elements within zones to make further assumptions.

MarginBoost is a stage-wise procedure corresponds to a gradient descent of a cost functional based on a decreasing function of the margin, in the space of linear combinations of base classifiers. This new method enhances work based on a direct plug-in extension of AdaBoost in the sense that all the ingredients of the gradient algorithm such as the gradient direction and the stopping rule are defined from the expression of the new cost function. Moreover, while the algorithm has been tested using the mixtures of models, SMMBoost is designed to combine any base classifiers that deals with both labeled and unlabeled data.

2.3 Learning algorithms for semi-supervised learning neural networks

Based on the achievements of the past algorithm, one can well see the uncertainty in such variables: distribution adjustment and combination of output values.

The AdaBoost algorithm, is considered one of the most successful for speeding up processing. By including consideration of binary classification in the -1 to +1 area, the output of the algorithm can be achieved through an exponential loss minimization function:

$l_{exp}(j \mid V) = E_{x\sim}V[e^{-f(x)j(x)}]$

Input data: data set $V = \{(y_1, r_1), (y_2,r_2),\ldots,(y_m, r_m)\}$; mane learning algorithm L; quantity of a learning iterations – Q.

1. $V_1(x) = 1/m$. (to find the weight L)
2. for $t = 1,\ldots,T$:
3. $j_t = L(V, V_t)$;
4. $e_t = P_{x\sim Vt}(j_t(x) \neq f(x))$; (find the error of j_t)
5. if $e_t > 0{,}5$ then break
6. $a_t = ½ \ln(1-e_t / e_t)$; (find the value of the j_t weight)
7. $V_{t+1}(x) = (V_t(x) / Z_t) * \{\exp(-a_t)$ if $j_t(x) = f(x)\}$; $\{\exp(a_t)$ if $j_t(x) \neq f(x)\}$ $= V_t(x)\exp(-a_t f(x) j_t(x)/ Z_t$ (Z_t can describe as a factor which enables V_{t+1} to the distribution stage)

Output data: $H(x) = \text{sign}(\Sigma^T_{t=1} a_t j_t(x))$

We can use summarize weighted combination for weak learners:

$H(x) = \Sigma^T_{t=1} a_t j_t(x)$

For using an exponential loss formula obtained a new basic and simple one, for getting minimalization of classification error. Main goal is to minimize the J value, often partial derivative of the loss for each x is equal to zero.

$de^{-f(x)J(x)}_{dJ(x)} = -f(x)e^{-f(x)J(x)} = -e^{-J(x)}P(f(x) = 1 \mid x) + e^{J(x)}P(f(x) = -1 \mid x) = 0$

After getting an answer we obtain:

$J(x) = ½ \ln \{P(f(x) = 1 \mid x)\}$;

$\{P(f(x) = -1 \mid x)\}$ and other,

$\text{sign}(J(x) = \text{sign}\{1/2 \ln P(f(x) = 1 \mid x\}; \{P(f(x) = -1 \mid x\} = \{1, P(f(x) = 1|x) > P(f(x) = -1 \mid x)\}; \{-1, P = (f(x) = 1 \mid x) < P(f(x) = -1 \mid x)\} = \text{argmax}\, P(f(x) = y \mid x);$ $y \in \{-1;1\}$

Furthermore, introduce that sign(J(x)) can get the Bayes error rate. In this example of algorithm was one of the cases was ignored – $P(f(x) = 1|x) = P(f(x) = -1|x)$. Also, known that exponential loss is minimized, the classification error too. If we replaced the non-differentiable classification problem, we can get better optimization point.

J created one-by-one with iteratively generating J_j and a_t. Seen that first classifier j_1 can be represented as activation of the weak learning algorithm with the new distribution. So, parameter j_t is generated near the distribution V_t, it's weight a_t also determined as $a_t j_t$ with exponential loss and minimizing.

$L_{exp}(a_t h_t \mid V_t) = E_{x\sim Vt}[e^{-f(x)atht(x)}] = E_{x\sim Vt}[e^{-at\|}(f(x) = j_t(x)) + e^{at\|} f(x) \neq J_t(x))] = e^{-at}P_{x\sim Vt}(f(x) = j_t(x)) + e^{at}P_{x\sim Vt}(f(x) \neq J_t(x)) = e^{-at}(1-e_t) + e^{at}e^t;$

Where $e_t = P_{x\sim Vt}(j_x(x) \neq f(x))$. So, if we get the needed a_t, exponential loss will be equal to zero:

$Dl_{exp}(a_t j_t \mid V_t) / da_t = -e^{-at}(1-e_t) + e^{at}et = 0;$

$a_t = ½ \ln(1-e_t/e_t)$

After combining processes one of equation with weak classifiers with weights can combined as H_{t-1}. With the help of AdaBoost adjusting the sample distribution for the next iteration, main algorithm can get output data with a weak classifier J_t with refusing a few mistakes of H_{t-1}. To minimize the exponential loss, it needs to see the ideal classifier j_t that corrects full obtain mistakes of H_{t-1}:

$L_{exp}(H_{t-1} + h_t \mid V) = E_{x\sim V}[e^{-f(x)(Ht-1(x)+ht(x))}] = E_{x\sim V}[e^{-f(x)Ht-1(x)}e^{-f(x)jt(x)}]$

In addition, Taylor formula for the case of $e^{-f(x)jt(x)}$, the exponential loss can be described with approximation by:

$L_{exp}(H_{t-1} + j_t \mid V) \approx E_{x\sim V}[e^{-f(x)Jt-1(x)}(1-f(x)j_t(x) + f(x)^2j_t(x)^2/2)] = E_{x\sim V}[e^{-f(x)Jt-1(x)}(1-f(x)j_t(x)+1/2)]$

From the task, we know $f(x)^2 = 1$ and $J_t(x)^2 = 1$;

Main classifier J_t is:

$J_t(x) = \text{argmin}_h lexp(J_{t-1} + j \mid V) = \text{argmin}_h E_{x\sim V}[e^{-f(x)Jt-1(x)}(1-f(x)j(x)+1/2)] =$

$=\text{argmax}_h E_{x\sim V}[e^{-f(x)jt-1(x)}f(x)j(x)]=\text{argmin}_h E_{x\sim V}[E_{x\sim V}{}^{e-f(x)Jt-1(x)}{}_{[e-f(x)Jt-1(x)]}f(x)j(x)]$

Knowing that $E_{x\sim V}[e^{-f(x)Jt-1(x)}]$ is a stable quantity.

Describe a distribution V_t as:

$V_t(x) = V(x)e^{-f(x)Jt-1(x)} / E_{x\sim V}[E^{-f(x)Jt-1(x)}]$

Throw the math representative form of upper formula we can obtain:

$J_t(x) = \text{argmax}_h E_{x\sim V}[{}^{e-f(x)Jt-1(x)}{}_{[E-f(x)Jt-1(x)]}f(x)j(x)] = \text{argmax}_h E_{x\sim Vt}[f(x)j(x)]$

We can write that in appropriate way: $f(x)j_t(x) = 1\text{-}2^{\|} (f(x) \neq j_t(x))$, normal classifier was $j_t(x) = \text{argmin}_h E_{x\sim Vt}[^{\|}(f(x) \neq j(x))]$.

Moreover, the perfect j_t minimize the classification error with the distribution V_t. We observe, that weaker learner trained under V_t and 0.5 classification error due to V_t. Describing the relation between V_t and V_{t+1}:

$$V_{t+1}(x) = V(x)e^{-f(x)Jt(x)}{}_{Ex\sim V[e-f(x)Jt(x)]} = V(x)e^{-f(x)Jt\text{-}1(x)e0f(x)atjt(x)}{}_{Ex\sim V[e-f(x)Jt(x)]} =$$

$$= Vt(x)e^{-f(x)atjt(x)}{}_{Ex\sim V[e-f(x)Jt(x)]}{}^{Ex\sim V[e-f(x)Jt\text{-}1(x)]}$$

From the formula above AdaBoost update with the sample distribution. That's how this algorithm works and can be learnable with specific distribution. The aim is to re-weighting with training examples in every iteration per one pass.

On the other side, without handling weighted training parameters, usually used the meaning of re-sampling and it gives sampling training parameters in every iteration per one iteration, including needed distribution. It applies main task.

Both of two variants include a part of performance which can be unique and not be clear between them at all. Re-sampling can produce other option for Boosting with restart. Every iteration of AdaBoost, program check for each learner and know that he better than random guess. Having an optimal number of rounds T AdaBoost will be early-refused far in a case of each T. Including even on multiclass problems.

If we have base learner which cannot be on pass check, re-sampling removed them, continue with a new pair of a generated base learners. So, the AdaBoost provides solving and avoiding for the early-termination tasks.

Starting from parsing the description of the previous working algorithm, you can see the advantages of processing it in a dataset. Having a coordinate axis in two projections, it is possible to parse 4 points where y(i) = f(zi), a unique point label. In this way, one can see the XOR problem. Most classes (positive and negative) cannot be crossed by a linear classifier.

Parsing the problem with the basic learning algorithm one can see the solution of eight different functions. By training the data under the required distribution, and having the output with the smallest error. If more than one function with the smallest error is found, one random function will be selected. All of the examples discussed below can include both classes.

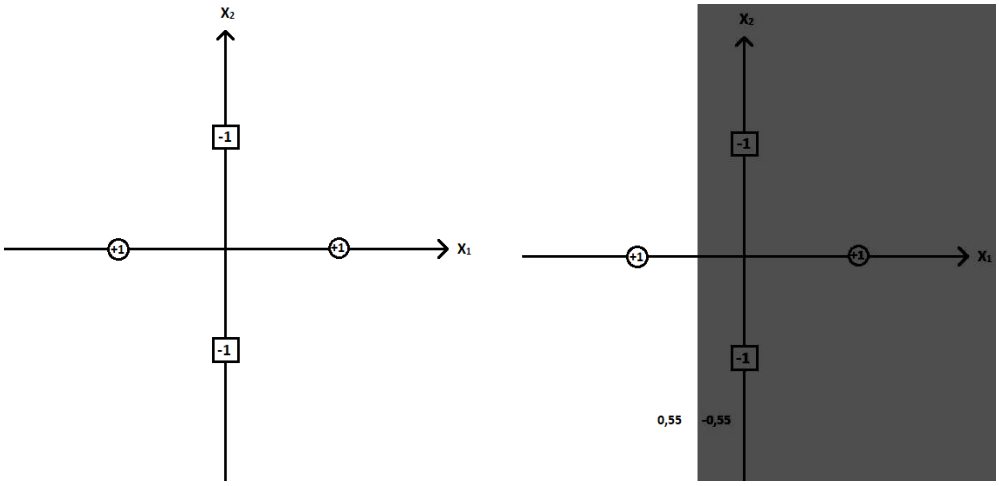


Figure 2.4 The XOR data and 1st iteration

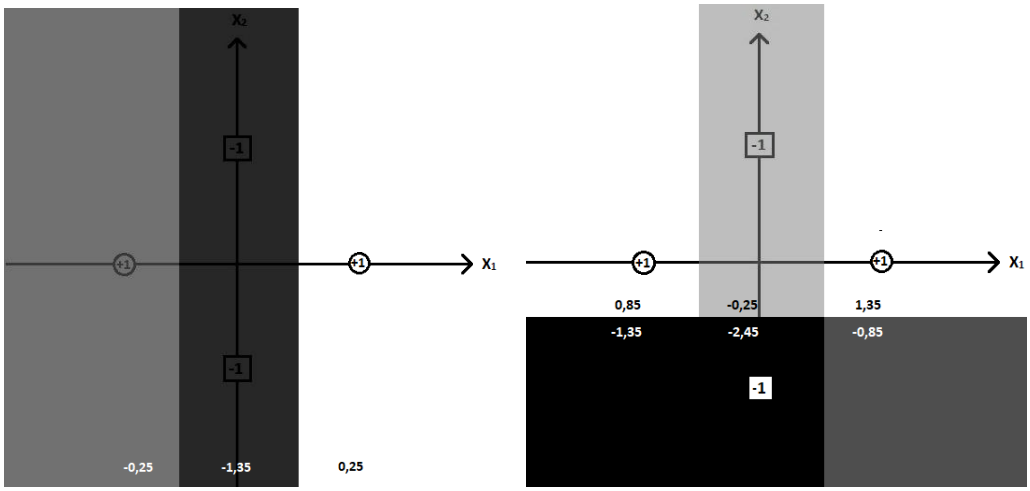


Figure 2.5: 2nd iteration and 3rd iteration

It is necessary to connect the trained algorithm to the data. Given that j_2, j_3, j_5 and j_8 have the smallest qualification error of 0.25, we can consider - j_2 as the classifier. Parameters x_1, x_2 are part of x in the first and last dimension.

$$j_1(x) = \begin{cases} +1, if\ (x1 > -0.5) \\ -1,\ otherwise \end{cases} \qquad j_2(x) = \begin{cases} -1, if\ (x1 > -0.5) \\ +1,\ otherwise \end{cases}$$

$$j_3(x) = \begin{cases} +1, if\ (x1 > +0.5) \\ -1,\ otherwise \end{cases} \qquad j_4(x) = \begin{cases} -1, if\ (x1 > +0.5) \\ +1,\ otherwise \end{cases}$$

$$j_5(x) = \begin{cases} +1, if\ (x2 > -0.5) \\ -1,\ otherwise \end{cases} \qquad j_6(x) = \begin{cases} -1, if\ (x2 > -0.5) \\ +1,\ otherwise \end{cases}$$

$$j_7(x) = \begin{cases} +1, if\ (x2 > +0.5) \\ -1,\ otherwise \end{cases} \qquad j_8(x) = \begin{cases} -1, if\ (x2 > +0.5) \\ +1,\ otherwise \end{cases}$$

Figure 2.5 Bringing up the learning algorithm on eight functions

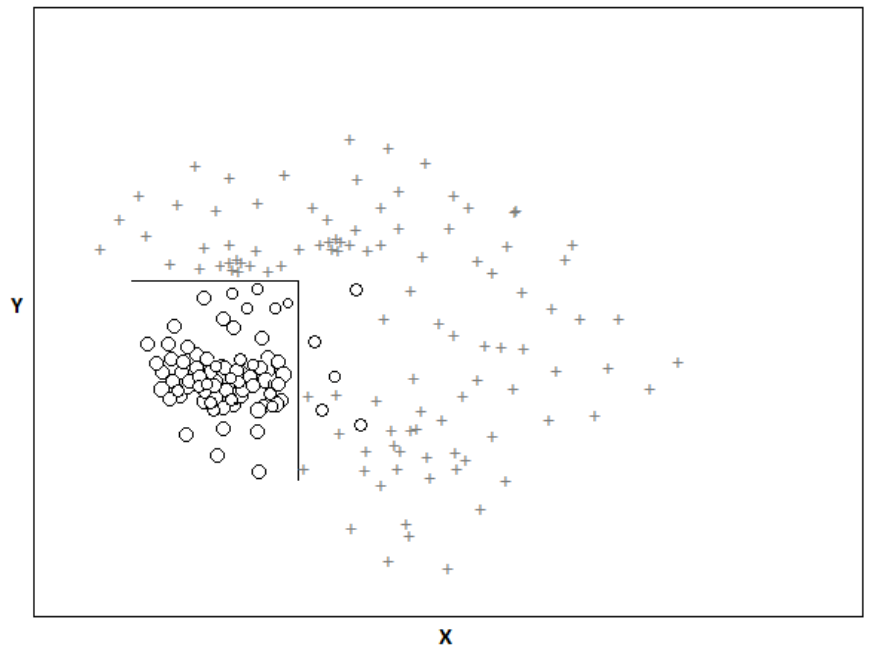


Figure 2.6 Single decision tree

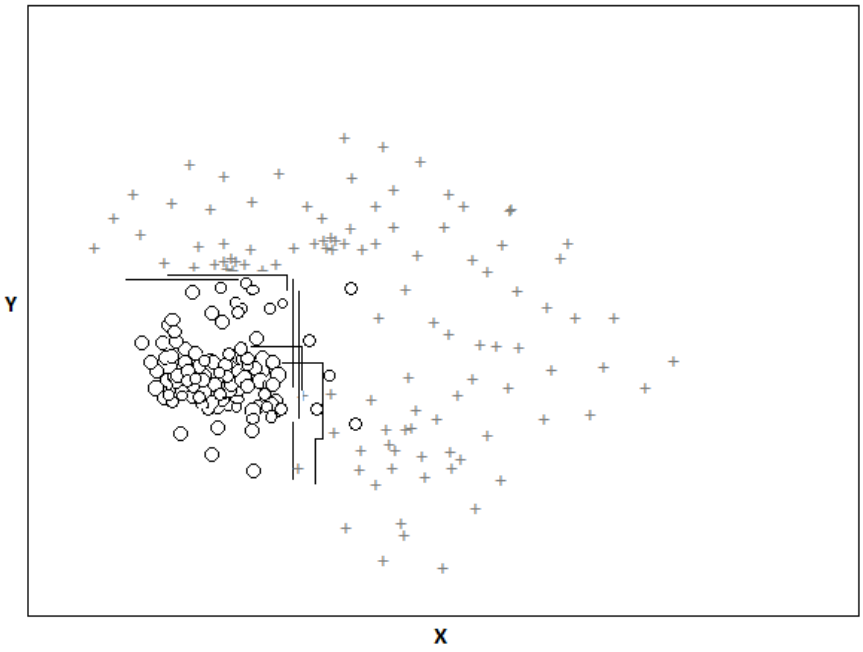


Figure 2.7 AdaBoost usage

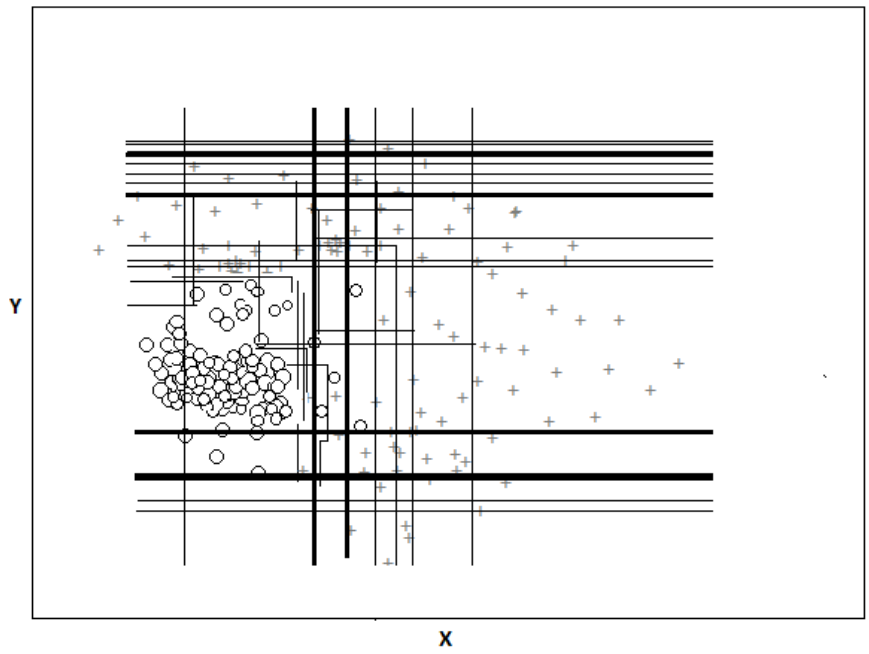


Figure 2.8 Three-Gaussian data set usage with AdaBoost decision trees

From the decision boundaries 0.25 if a fourth half of error from the first figures. The weight of j_2 is 0.5ln 3 is equal to 0.55, so describing an image with the classification weight on the black and grey sides we can see the values of 0.55 and -0.55.

On figure 2.3 Parameter z_1 is higher, we can see other iteration with rounds of the main algorithm on the invoking stage. In each case of j_3, j_5 and j_8 parameters have smallest error with taking a supposed parameter j_3 with weight 0.8.

Next figure shows combination of parameters j_2 and j_3 using grey levels. The weight of z_2 increasing during parameters j_5 and j_8 have small error. Against taken j_5 as supposed parameter and optimal weight that saw on picture 2.4 with combined classification parameters j_2, j_3 and j_5 accordingly.

The last step is to distribute the classification: z_1 and z_2 are positive weights. Other cases are negative sets of weights. You can see the correct placement of the weights. AdaBoost produced a non-linear classifier with an error of zero.

The main drawback of the algorithm is that it is overloaded in its data processing, which significantly reduces performance in some cases.

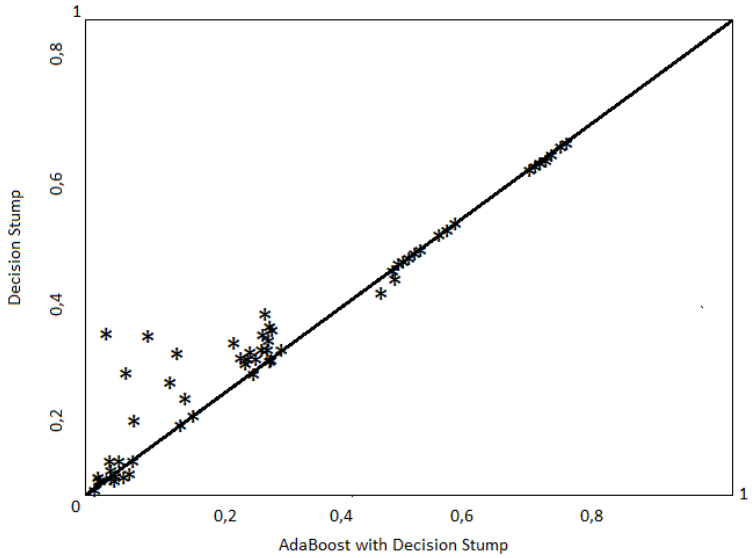


Figure 2.9 Decision Stump results

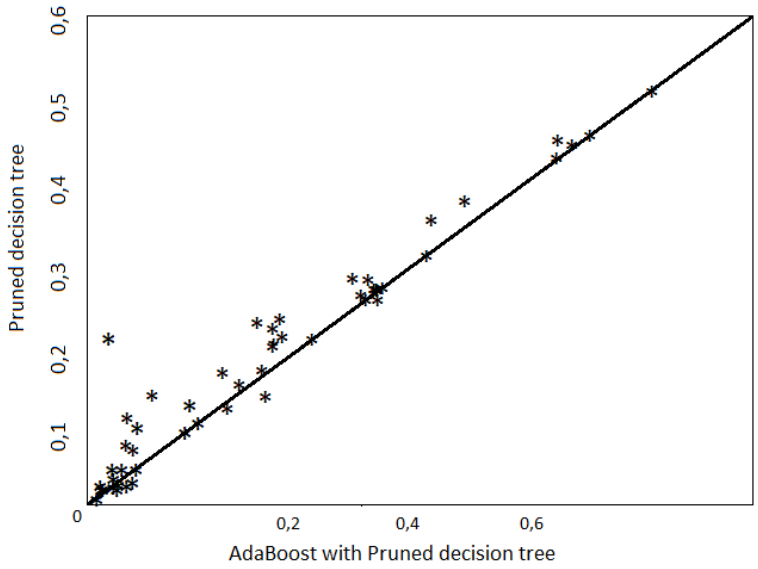


Figure 2.10: Pruned decision tree results

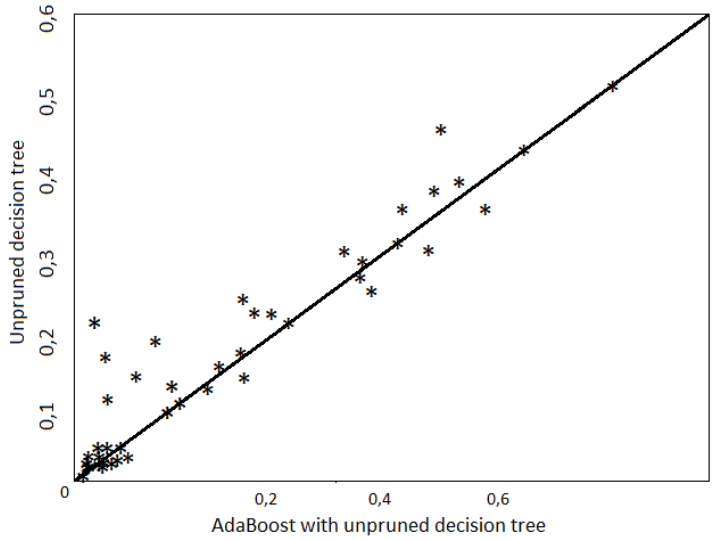


Figure 2.11: Unpruned decision tree results

Including the latest results of the AdaBoost method, we can consider the available range of classifications of SSA, let's construct table 2.1:

T – transductive property;

I – inductive property;

Group	Approach	Summary	T/I
Manifold Assumption	Label Propagation	Graph-based; Maximize label consistency using Graph Laplacian	T
	Min-Cuts	Edge-weight based graph-partitioning algorithm constraining nodes with same label to be in same partition	T
	MRFs, GRFs	Markov random field and Gaussian random field models	T
	LDS	TSVM trained on a dimensionality reduced data using graph-based kernel	T
	SGT	Classification cost minimized with a Laplacian regularizer	T
	LapSVM	SVM with Laplacian regularization	I
Cluster Assumption	Co-training	Maximizes predictor consistency among two distinct feature views	I
	Self-training	Assumes pseudo-labels as true labels and retrains the model	I
	SSMB	Maximizes pseudo-margin using boosting	I
	ASSEMBLE	Maximizes pseudo-margin using boosting	I
	Mixture of Experts	EM based model-fitting of mixture models	I
	EM-Naive Bayes	EM based model-fitting of Naive Bayes	I
	TSVM, S3VM	Margin maximization using density of unlabeled data	I
	Gaussian processes	Bayesian discriminative model	I
Manifold & Cluster As-sumptions	SemiBoost	Boosting with a graph Laplacian inspired regularization	I

To summarize, in the next section we can consider several algorithms involving ensemble, for processing semi-guided learning on a sample. By selecting the correct ensemble with an iterative pass, you can view the performance and quality of the algorithm's response.

CHAPTER 3. STRUCTURALLY PARAMETRIC SYNTHESIS OF AN ENSEMBLE OF NEURAL NETWORKS.

3.1 An ensemble approach in the construction of neural network-based classifiers.

Develop a Hybrid Ensemble consisting of six sub-ensembles: Bagged Logistic Regression, Random Forest, Kernel Factory, Bagged Support Vector Machines, Stochastic Boosting, and Bagged Neural Networks. We test the algorithm on eleven data sets using five times twofold cross-validation. The Hybrid Ensemble significantly and consistently outperforms the Single Best sub-ensemble on all data sets when the authority method or Self Organizing Migrating Algorithm is used for weight estimation.

Analyses also indicate that the Hybrid Ensemble yields increasingly important classification improvements with increasingly difficult tasks. To the best of our knowledge this study is the first to assess the added value of algorithm-induced diversity over and above data-induced diversity in ensemble design.

While data perturbation techniques have flourished in recent years, the use of algorithm variation has remained largely unexplored. Although several papers recognized that combining multiple inductive biases is an effective way to create diversity, only few scholars have used this strategy so far. Furthermore, even less authors recognized that the integration of both schemes could be beneficial in improving the ensemble accuracy. Summarizes the literature by including a set of ground-breaking, pivotal papers that use data variation and algorithm variation.

Let’s describe the Hybrid Diversity Generation Strategies on table 3.1:

Data set	Data	Algorithm	Algorithm → Data	Data → Algorithm
Breiman (1996)	x			
Freund and Schapire (1996)	x			
Ho (1998)	x			
Breiman (2001)	x			
Michalski et al. (1994)		x		
Woods et al. (1997)		x		
Wang et al. (2000)		x		
Tsoumakas et al. (2005)		x		
Canuto et al. (2007)		x		
Menahem et al. (2009)		x		
Nascimento and Coelho (2009)	x	x	x	
This study	x	x		x

Depend on the output that is received from the base classifiers. While some form of (weighted) voting has become the most frequently used method in classifier fusion, it is not necessarily the best option. A lot of information is lost as it only uses class label output. Confidences or posteriori probabilities contain the highest amount of information and are able to reduce the generalization error. Therefore, in our hybrid design we calibrate all of our models to ensure we have measurement level output for every base classifier. As a result, we can fuse our classifiers using weighted averaging. In this study we restrict ourselves to linear combinations because we want to limit the number of parameters that needs to be estimated in order to keep the analysis tractable.

To calculate the weights, one can make use of fixed rules or trained weights. While fixed rules have a very small-time complexity and provide simplicity, their result is expected to be worse than that of the trained ones. The easiest way to combine the different measures is to take the simple average. The performance of this simple method is often close to that of the weighted average but in most cases, a weighted average is able to outperform the simple average. In performance- or authority- based weighting the weight of each classifier is set proportional to its performance on a validation set.

This method typically performs better albeit at the cost of more computational effort. Weights can be trained by either a statistical method or a general-purpose solver. The latter category has the advantage that the objective function can be chosen freely to fit the application and that it is more likely to find global optima for the parameters. Although not the primary focus of this study, we tried to benchmark as many methods as possible in each category. Characteristics of base classifiers will represent in table 3.2

Algorithm	Main features	Objective function	Optimization method
Ensemble of Neural Networks	Identical to Logistic Regression if no hidden layer and if logistic activation function is used. Semi/non parametric. Hidden layer(s). Bagging.	Conditional log loss	Quasi Newton Method
Ensemble of Support Vector Machines	Maximum margin seperating hyperplane and soft margin. Kernels. Bagging.	Hinge loss	Constrained quadratic programming
Ensemble of Logistic Regressions with Lasso	Linear combination of features. Logistic link function. Bagging.	Penalized log loss	Newton method
Random Forest	Parallel- independently built trees. Random feature selection. Bagging.	No explicit global loss function. Locally: entropy=log loss	Greedy search
Kernel Factory	Kernels. Base classifiers are Random Forests. Random Forest handles remaining non-linearities from kernels.	No explicit global loss function. Locally: entropy=log loss	Greedy search
Stochastic AdaBoost	Iterative- dependently built trees. Focus on poorly classified instances. Hybrid bagging and boosting.	Exponential loss	Gradient descent

Base classifier parameter tuning is performed by cross-validation using X train, Y train, X validate, and Y validate. The combiners that are tuned use Y' validate and Y validate.

Input:

• x = predictor variables

• y = response variable with class labels {0,1}

• combine = one of the following combination methods {GA, DEA, GSA, MALSC, PSO, SOMA, TSA, NNBL, GINNLS, LHNNLS, AUTH, MEAN}

• member parameters=parameters of base classifiers

• combination parameters=parameters of combiners

Classifier Generation:

Randomly divide x into X train (50% of instances) and X validate (50%)

Make the same split for y: Y train and Y validate

Algorithms ← (LR, RF, AB, KF, NN, SV)

for Algorithms do

Tuned Parameters ← tune (X train, Y train, X validate, Y validate)

Classifiers ← train (X train, Y train, Tuned Parameters)

Y' validate ← predict (Classifiers, X validate)

Calibrators ← train calibrator (Y' validate, Y validate)

Y' validate ← calibrate (Calibrators, Y' validate)

Evaluations ← evaluate (Y validate, Y validate)

Classifiers ← train (X, Y, Tuned Parameters)

End.

Classifier combination:

if combine one of {GA, DEA, GSA, MALSC, PSO, SOMA, TSA, NNBL, GONNLS, LHNNLS}

then

weights ← optimize classifier weights (Y' validate, Y validate)

else if combine == AUTH then

weights ← evaluations/sum(evaluations)

else if combine == MEAN then

weights ← (1/6,1/6,1/6,1/6,1/6,1/6)

Result: Calibrators, Classifiers, Weights

Semi-supervised learning deals with methods for exploiting the unlabeled data in addition to the labeled data to improve performance on the classification task. Semi-supervised learning has been the topic of four different Neural Information Processing Workshops. Ensemble methods such as AdaBoost work by iteratively using a base learning mechanism to construct a classifier to improve the ensemble classifier and then adding the classifier to the current ensemble with an appropriate scalar multiplier (the step-size). It is well known that such algorithms are performing gradient descent of an error function in function space. Depending on the measure of quality of the classifier, different criteria are produced for choosing the base classifier and assigning the step-size.

The advantages of Ensemble:

• Any weight-sensitive classification algorithm can be boosted using labeled and unlabeled data;

• Unlabeled data can be assimilated into margin-cost based ensemble algorithms for both two-class and multi-class problems;

• Ensemble can efficiently exploit the adaptive step-sizes used to weight each base learner within existing supervised ensemble methods. SSMBoost is practically limited to fixed step-sizes;

• Ensemble can exploit unlabeled data to reduce the number of classifiers needed in the ensemble therefore speeding up learning.

• Computational results show the approach is effective on a number of test problems, producing more accurate ensembles than AdaBoost using the same number of base learners.

Determine unlabeled data we must come up with a mechanism for defining the margin associated with unlabeled data points. While the strategy is in general applicable to many different margin cost functions, we focus on the one used in AdaBoost.

Let the base classifiers be fj (x): Rn → [1, −1] where fj is the j_{th} classifier in the ensemble.

Let the labeled training data, L, be the n-dimensional points $x_1, \ldots, x`$ with know labels, $y_1, \ldots, y_l$.

For now, assume the problem has two classes $y_i = 1$ or −1. A multi-class extension is discussed in later sections. The ensemble classifier F (x) is formed from a linear combination of the J base classifiers:

$$F(x) = \Sigma e^{-yiF(xi)} \quad (3.1)$$

$F(x) = \Sigma^{j}_{i=1} w_j f_j(x)$, where w_j is the weighting term for the j_{th} classifier. For labeled data points the margin is $y_i F(x_i)$. AdaBoost performs gradient descent in function space in order to minimize an exponential margin cost function.

U – unlabeled data.

To incorporate unlabeled data, we must define the margin of an unlabeled data point. We do not know the class y_i for unlabeled data points. Note that for labeled data points, the margin, $y_i F(x_i)$, is positive if the point is correctly classified and negative if the point is wrongly classified. An unlabeled point is never right or wrong.

So, we define the margin for an unlabeled data point x_i to allow the same margin to be used for both supervised and unsupervised data we can introduce the concept of a pseudo-class. The pseudo-class of an unlabeled data point x_i is defined as $y_i = \text{sign}(F(x_i))$.

The margin then is $y_i F(x_i)$ where y_i is the known class label if x_i is labeled or the pseudo-class if x_i is unlabeled. The introduction of the pseudo-class is the critical difference between our approach and the independently developed SSMBoost. By introducing pseudo-classes, we can show that our Adaptive Semi-Supervised Ensemble method, which corresponds to the intuitive semi-supervised ensemble algorithm, maximizes the margins of both the labeled and unlabeled points in function space. As noted above, given that the margin for labeled points is $y_i F(x_i)$, we can define the margin for unlabeled data points as $|F(x_i)|$. Using these values, we can then define a margin cost function that incorporates both labeled and unlabeled data.

The Essemble cost function for AdaBoost is:

$$C(F) = \sum_{i \in labeled} \alpha_i e^{-y_i F(x_i)} + \sum_{j \in unlabeled} \alpha_j e^{-|F(x_j)|} \quad (3.2)$$

In general, the Essemble cost function for any margin cost function, M, is

$$C(F) = \sum_{i \in labeled} \alpha_i M\left(-y_i F(x_i)\right) + \sum_{j \in unlabeled} \alpha_j \left(-|F(x_j)|\right) \quad (3.3)$$

The terms α_i and α_j are used to weight the labeled and unlabeled data so that we could, for example, choose to weight the margins associated with unlabeled data points as counting only 40% as much as the margins for labeled data points.

To create a practical descent-based algorithm we build on the AnyBoost approach. Recall the AnyBoost algorithm from:

1. Let $F_0(x) := 0$;

2. for t: = 0 to T do;

3. Let f_{t+1}: = L (F_t, $-\nabla C(F_t)$);

4. if $-<\nabla C(F_t), f_{t+1}>\leq 0$ then;

5. return Ft;

6. end if;

7. Choose w_{t+1}

8. Let F_{t+1}: = Ft + $_{wt+1}f_{t+1}$

9. end for

10. return F_{T+1}

3.2 A review of neural network ensemble combining-based approaches

The boosting algorithm within the generic margin cost functional framework for boosting. Boosting is treated as a greedy yet stage-wise functional minimization procedure where each stage seeks a function from a given subspace so that combining it with those functions already found in the same way can lead to the greatest reduction in terms of a cost functional defined based on training examples. Since our algorithm is within the generic margin cost functional framework developed for generic yet abstract boosting algorithms, it allows a range of various margin cost functions to be applied.

To facilitate our boosting learning, we also come up with an initialization setting based on clustering analysis. It is worth stating that our algorithm is developed for binary classification tasks but easily extended to cope with multiclass classification tasks via the one-against-rest scheme although this treatment might be less efficient than those methods developed very recently for

multi-class boosting without the use of binary decomposition. Extensive experiments demonstrate that algorithm yields favorite results for benchmark and real-world classification tasks in comparison to many state-of-the-art SSL algorithms including semi-supervised boosting algorithms.

The generic form of an ensemble learner constructed by boosting is the voted combination of base learners, sign [F(x)]. F(x) is the linear combination of base learners as follows:

$$F(x) = \sum_t W_t f_t(x) \quad (3.4)$$

For binary classification, f_t: $X \rightarrow \{+1, -1\}$ are base classifiers and $w_t \in R$, are weights for linear combination.

Given a training set of |L| labeled examples, L = {(x1, y1), …, (x,y|L|)}, generated according to a distribution, boosting finds out F(x) so that $P(F(x) \neq y)$ on this distribution is minimized. In reality, the distribution is unknown and a training set L is available only. Thus, boosting would find F(x) by minimizing a margin cost functional defined on the training set L:

$$C(F) = \frac{1}{|L|} \Sigma\, C[y_i\, F(x_i)], \quad (3.5)$$

where C: $R \rightarrow R$ is a non-negative and monotonically decreasing cost function. In $y_i F(x_i)$ is the margin of an example, $i \in L$, with respect to F (x).

3.3. Synthesis of algorithm in the construction of an ensemble of neural networks with partial teacher involvement based on boosting

1. Let $w_o(i) = l/l$, i = 1, ..., l;

2. Let $g_o(x) = 0$;

3. For t = 1 ... T (gradient descent);

4. Learn a gradient direction $h_{t+1} \in H$ with a high value of:

$$J_t^s = \Sigma_i \in_s w_t(i) y_y h_{t+1}(x_i) \quad (3.6)$$

5. Apply the stopping rule: if $J_t^s \leq \Sigma_i \in_s w_t(i) y_i g_t(x_i)$ then return g_t else go on;

6. Choose a step-length for the obtained direction by a line-search or by fixing it as a constant ∈;

7. Add the new direction to obtain: $g_{t+1} = \frac{|at| gt+at-1ht+1}{|at+1|}$ (3.7)

8. Fix the weight distribution: $W_{t+1} = \frac{c\prime(p(gt+1(xi),yi))}{\Sigma i \in s\ c'(p(gt+1(xj),yi))}$ (3.8)

So, in that way we can synthesis one of the needed boosting algorithms.

3.4 Research results

Taking a dataset with text, we can visualize the value: build a graph (heat map) that shows the correction sign between themselves and with the target variable (markups). Coding structures will be in next chapter.

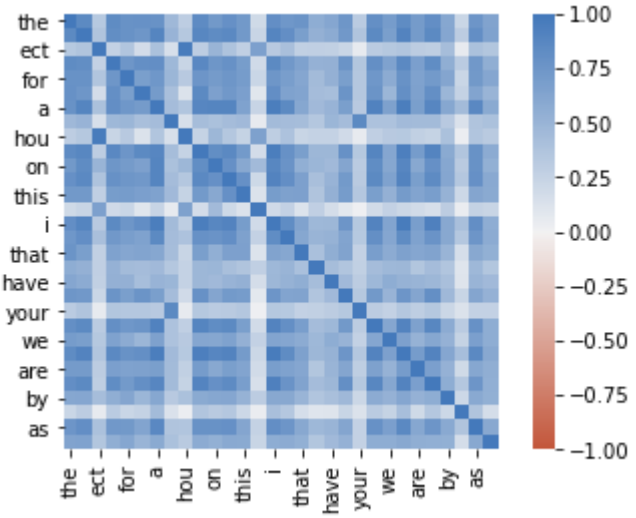


Figure 3.1 Heat map with correction sign

Constructing histograms of the distribution of the label boxplot-and the attribute and the target variable:

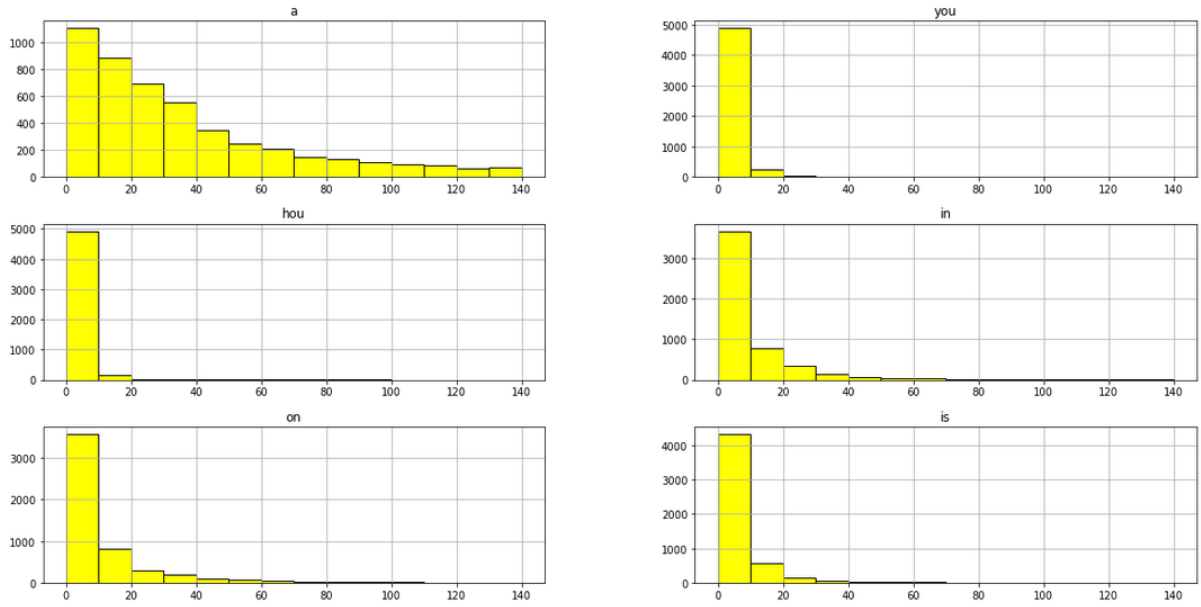


Figure 3.2 Boxplot with the target variables

From that we can build prediction change graphs with the 0 and 1 values:

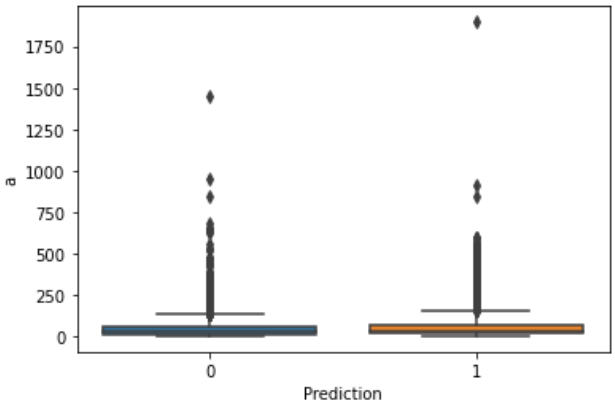


Figure 3.3 Boxplot with prediction value for ‘a’ case

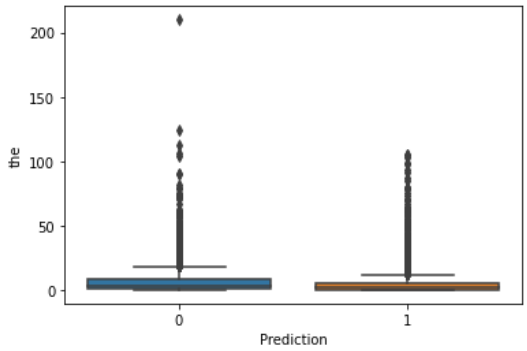


Figure 3.4 Boxplot with prediction value for ‘the’ case

After normalizing the data, we can obtain a training test split, in our case this dataset has 5172 rows x 3000 columns. That we have length in 5172 values.

	the	to	ect	and	for	of	a	you	hou	in	...	enhancements	connevey	jay	valued	lay	infrastructure	military	all
0	0.000000	0.000000	0.094072	0.000000	0.000000	0.000000	0.188144	0.000000	0.000000	0.000000	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
1	0.023453	0.038111	0.070358	0.017590	0.017590	0.005863	0.299023	0.002932	0.079153	0.052769	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
2	0.000000	0.000000	0.049326	0.000000	0.000000	0.000000	0.394611	0.000000	0.000000	0.197305	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
3	0.000000	0.030395	0.133737	0.000000	0.030395	0.006079	0.310026	0.012158	0.060789	0.006079	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
4	0.041197	0.035312	0.100050	0.005885	0.029427	0.011771	0.335462	0.000000	0.052968	0.017656	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
5167	0.024624	0.024624	0.024624	0.036936	0.000000	0.000000	0.393982	0.000000	0.000000	0.061560	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
5168	0.075673	0.058376	0.023783	0.004324	0.012973	0.010810	0.326475	0.008648	0.006486	0.049728	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
5169	0.000000	0.000000	0.032756	0.032756	0.000000	0.000000	0.360317	0.000000	0.000000	0.032756	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
5170	0.016025	0.056086	0.008012	0.000000	0.016025	0.008012	0.224345	0.016025	0.000000	0.064099	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
5171	0.052734	0.057528	0.011985	0.002397	0.014382	0.011985	0.354759	0.019176	0.004794	0.055131	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	

5172 rows × 3000 columns

Figure 3.5 Normalized data structure

Our task of this work is to train classifiers and use ensemble with different algorithms, first result with confusion matrix will be for kNN method:

```
Num neighbors: 1, Error rate = 0.10821256038647344
Num neighbors: 2, Error rate = 0.11400966183574879
Num neighbors: 3, Error rate = 0.13333333333333333
Num neighbors: 4, Error rate = 0.13140096618357489
Num neighbors: 5, Error rate = 0.13429951690821257
Num neighbors: 6, Error rate = 0.13623188405797101
Num neighbors: 7, Error rate = 0.1468599033816425
Num neighbors: 8, Error rate = 0.13719806763285025
Num neighbors: 9, Error rate = 0.14299516908212562
Num neighbors: 10, Error rate = 0.1468599033816425
Num neighbors: 11, Error rate = 0.14879227053140096
Num neighbors: 12, Error rate = 0.14782608695652175
Num neighbors: 13, Error rate = 0.15265700483091788
Num neighbors: 14, Error rate = 0.1565217391304348
```

Best result with 1 neighbor

```
Confusion matrix:
[[688  62]
 [ 50 235]]
Classification report:
              precision    recall  f1-score   support

           0       0.93      0.92      0.92       750
           1       0.79      0.82      0.81       285

    accuracy                           0.89      1035
   macro avg       0.86      0.87      0.87      1035
weighted avg       0.89      0.89      0.89      1035
```

Figure 3.6 kNN method results and confusion matrix

Taken a decision tree classifier with X and Y training fit we testing dataset to obtain precision and accuracy for the statistical purpose.

```
Confusion matrix:
[[705  45]
 [ 52 233]]
Classification report:
              precision    recall  f1-score   support

           0       0.93      0.94      0.94       750
           1       0.84      0.82      0.83       285

    accuracy                           0.91      1035
   macro avg       0.88      0.88      0.88      1035
weighted avg       0.91      0.91      0.91      1035
```

Figure 3.7 Decision tree classifier results

Let’s get an image throw the plot function by a programming way. Here we can see the graph of decision tree:

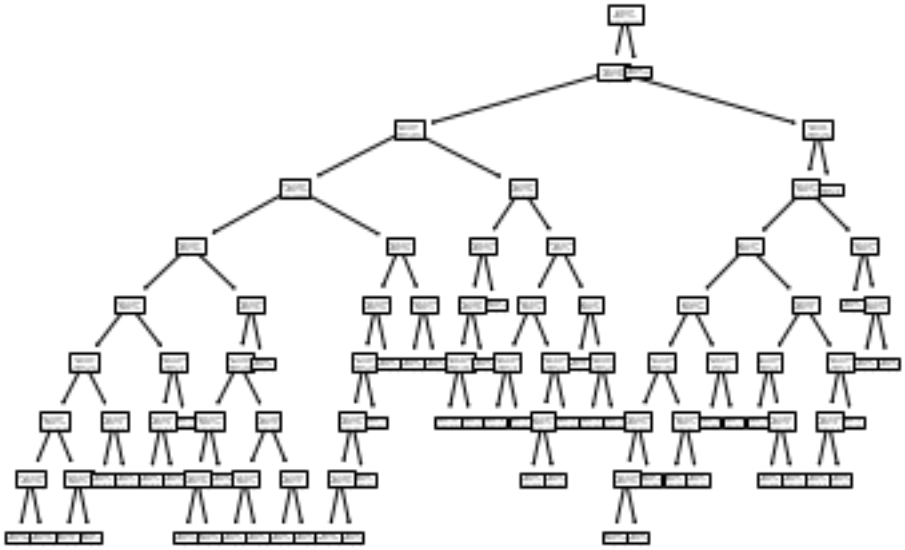


Figure 3.8 Decision tree graph image plot

For the SVM algorithm we can take Linear Kernel, Polynomial Kernel, RBF Kernel and Sigmoid Kernel methods. From this we obtain classificational report with accuracy, weighted and macro avg parameters.

Linear Kernel:

```
Confusion matrix:
[[702  48]
 [ 53 232]]
Classification report:
              precision    recall  f1-score   support

           0       0.93      0.94      0.93       750
           1       0.83      0.81      0.82       285

    accuracy                           0.90      1035
   macro avg       0.88      0.88      0.88      1035
weighted avg       0.90      0.90      0.90      1035
```

Figure 3.9 Linear Kernel results

Polynomial Kernel:

```
Confusion matrix:
[[702  48]
 [ 53 232]]
Classification report:
              precision    recall  f1-score   support

           0       0.93      0.94      0.93       750
           1       0.83      0.81      0.82       285

    accuracy                           0.90      1035
   macro avg       0.88      0.88      0.88      1035
weighted avg       0.90      0.90      0.90      1035
```

Figure 3.10 Polynomial Kernel results

RBF Kernel:

```
Confusion matrix:
[[702  48]
 [ 53 232]]
Classification report:
              precision    recall  f1-score   support

           0       0.93      0.94      0.93       750
           1       0.83      0.81      0.82       285

    accuracy                           0.90      1035
   macro avg       0.88      0.88      0.88      1035
weighted avg       0.90      0.90      0.90      1035
```

Figure 3.11 RBF Kernel results

Sigmoid Kernel:

```
Confusion matrix:
[[702  48]
 [ 53 232]]
Classification report:
              precision    recall  f1-score   support

           0       0.93      0.94      0.93       750
           1       0.83      0.81      0.82       285

    accuracy                           0.90      1035
   macro avg       0.88      0.88      0.88      1035
weighted avg       0.90      0.90      0.90      1035
```

Figure 3.12 Sigmoid Kernel results

Including last 4 data from Kernel methods, we can produce SVM Gridsearch for prediction results:

```
Fitting 5 folds for each of 9 candidates, totalling 45 fits
[CV 1/5] END .....................C=1, gamma=1, kernel=rbf; total time=  11.9s
[CV 2/5] END .....................C=1, gamma=1, kernel=rbf; total time=  12.3s
[CV 3/5] END .....................C=1, gamma=1, kernel=rbf; total time=  12.0s
[CV 4/5] END .....................C=1, gamma=1, kernel=rbf; total time=  14.2s
[CV 5/5] END .....................C=1, gamma=1, kernel=rbf; total time=  15.2s
[CV 1/5] END ...................C=1, gamma=0.1, kernel=rbf; total time=  21.6s
[CV 2/5] END ...................C=1, gamma=0.1, kernel=rbf; total time=  21.9s
[CV 4/5] END .................C=100, gamma=0.1, kernel=rbf; total time=   9.1s
[CV 5/5] END .................C=100, gamma=0.1, kernel=rbf; total time=   8.8s
[CV 1/5] END ................C=100, gamma=0.01, kernel=rbf; total time=  14.4s
[CV 2/5] END ................C=100, gamma=0.01, kernel=rbf; total time=  14.1s
[CV 3/5] END ................C=100, gamma=0.01, kernel=rbf; total time=  14.0s
[CV 4/5] END ................C=100, gamma=0.01, kernel=rbf; total time=  14.2s
[CV 5/5] END ................C=100, gamma=0.01, kernel=rbf; total time=  14.1s
```

Figure 3.13 Start and end of the grid prediction iterations

Final confusion matrix with accuracy for SVM algorithm:

```
Confusion matrix:
[[731  19]
 [ 11 274]]
Classification report:
              precision    recall  f1-score   support

           0       0.99      0.97      0.98       750
           1       0.94      0.96      0.95       285

    accuracy                           0.97      1035
   macro avg       0.96      0.97      0.96      1035
weighted avg       0.97      0.97      0.97      1035
```

Figure 3.14 SVM methods confusion matrix results

Futhermore, Random Forest method include almost the same result:

```
Confusion matrix:
[[737  13]
 [ 17 268]]
Classification report:
              precision    recall  f1-score   support

           0       0.98      0.98      0.98       750
           1       0.95      0.94      0.95       285

    accuracy                           0.97      1035
   macro avg       0.97      0.96      0.96      1035
weighted avg       0.97      0.97      0.97      1035
```

Figure 3.15 Random Forest results

With the value of depth = 16, and estimations number = 256 we can do a Random Forest Gridsearch:

```
Confusion matrix:
[[737  13]
 [ 23 262]]
Classification report:
              precision    recall  f1-score   support

           0       0.97      0.98      0.98       750
           1       0.95      0.92      0.94       285

    accuracy                           0.97      1035
   macro avg       0.96      0.95      0.96      1035
weighted avg       0.97      0.97      0.97      1035
```

Figure 3.16 Random Forest Gridsearch results

Taken the last algorithm, Ada Boost:

```
Confusion matrix:
[[717  33]
 [ 23 262]]
Classification report:
              precision    recall  f1-score   support

           0       0.97      0.96      0.96       750
           1       0.89      0.92      0.90       285

    accuracy                           0.95      1035
   macro avg       0.93      0.94      0.93      1035
weighted avg       0.95      0.95      0.95      1035
```

Figure 3.17 Ada Boost confusion matrix results

Ada Boost Gridsearch with 200 numbers of estimators and 1 value of learning rate:

```
Confusion matrix:
[[728  22]
 [ 18 267]]
Classification report:
              precision    recall  f1-score   support

           0       0.98      0.97      0.97       750
           1       0.92      0.94      0.93       285

    accuracy                           0.96      1035
   macro avg       0.95      0.95      0.95      1035
weighted avg       0.96      0.96      0.96      1035
```

Figure 3.18 Ada Boost Gridsearch results

To see the results on a graph we need to construct fully connected feed-forward network:

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 128)               384128

 dense_1 (Dense)             (None, 128)               16512

 dense_2 (Dense)             (None, 1)                 129

=================================================================
Total params: 400,769
Trainable params: 400,769
Non-trainable params: 0
```

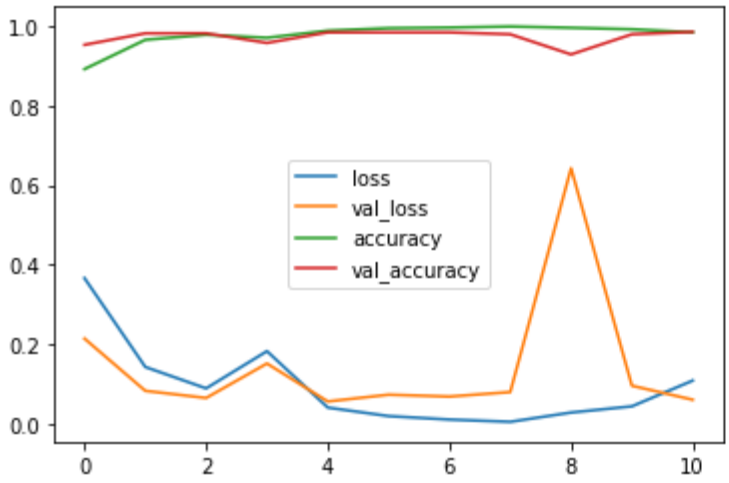
Figure 3.19 Sequential model of fully connected feed-forward network and parameters

So, this way we take results with 15 epochs and all needed values:

```
Epoch 1/15
207/207 [==============================] - 2s 7ms/step - loss: 0.3655 - accuracy: 0.8918 - val_loss: 0.2133 - val_accuracy: 0.9529
Epoch 2/15
207/207 [==============================] - 1s 6ms/step - loss: 0.1418 - accuracy: 0.9655 - val_loss: 0.0818 - val_accuracy: 0.9819
Epoch 3/15
207/207 [==============================] - 1s 6ms/step - loss: 0.0880 - accuracy: 0.9782 - val_loss: 0.0638 - val_accuracy: 0.9819
Epoch 4/15
207/207 [==============================] - 1s 6ms/step - loss: 0.1816 - accuracy: 0.9707 - val_loss: 0.1506 - val_accuracy: 0.9577
Epoch 5/15
207/207 [==============================] - 1s 5ms/step - loss: 0.0396 - accuracy: 0.9885 - val_loss: 0.0550 - val_accuracy: 0.9843
Epoch 6/15
207/207 [==============================] - 1s 5ms/step - loss: 0.0182 - accuracy: 0.9949 - val_loss: 0.0721 - val_accuracy: 0.9843
Epoch 7/15
207/207 [==============================] - 1s 5ms/step - loss: 0.0095 - accuracy: 0.9964 - val_loss: 0.0674 - val_accuracy: 0.9843
Epoch 8/15
207/207 [==============================] - 1s 5ms/step - loss: 0.0038 - accuracy: 0.9991 - val_loss: 0.0787 - val_accuracy: 0.9795
Epoch 9/15
207/207 [==============================] - 1s 4ms/step - loss: 0.0272 - accuracy: 0.9958 - val_loss: 0.6421 - val_accuracy: 0.9287
Epoch 10/15
207/207 [==============================] - 1s 4ms/step - loss: 0.0428 - accuracy: 0.9921 - val_loss: 0.0947 - val_accuracy: 0.9795
Epoch 11/15
207/207 [==============================] - 1s 5ms/step - loss: 0.1076 - accuracy: 0.9843 - val_loss: 0.0594 - val_accuracy: 0.9855
```

Figure 3.20 Epoch representation with losses and accuracy

Final accuracy after 15 epochs with graph:




```
33/33 [==============================] - 0s 3ms/step - loss: 0.0449 - accuracy: 0.9845
Test score: 0.04490905627608299
Test accuracy: 0.9845410585403442
```

Figure 3.21 Final accuracy and graph for AdaBoost ensemble

In addition, evaluate results from RNN and CNN:

```
Epoch 1/15
68/68 [==============================] - 184s 3s/step - loss: 4.1375 - accuracy: 0.1413 - val_loss: 2.2795 - val_accuracy: 0.1400
Epoch 2/15
68/68 [==============================] - 160s 2s/step - loss: 2.2338 - accuracy: 0.1896 - val_loss: 2.2762 - val_accuracy: 0.1000
Epoch 3/15
68/68 [==============================] - 161s 2s/step - loss: 2.1580 - accuracy: 0.2156 - val_loss: 2.2602 - val_accuracy: 0.1500
Epoch 4/15
68/68 [==============================] - 172s 3s/step - loss: 2.0761 - accuracy: 0.2751 - val_loss: 2.2648 - val_accuracy: 0.1600
Epoch 5/15
68/68 [==============================] - 169s 2s/step - loss: 1.7886 - accuracy: 0.3783 - val_loss: 2.2480 - val_accuracy: 0.2500
Epoch 6/15
68/68 [==============================] - 168s 2s/step - loss: 1.3927 - accuracy: 0.5474 - val_loss: 2.6817 - val_accuracy: 0.2200
Epoch 7/15
68/68 [==============================] - 162s 2s/step - loss: 0.9444 - accuracy: 0.6905 - val_loss: 2.5404 - val_accuracy: 0.2600
Epoch 8/15
68/68 [==============================] - 158s 2s/step - loss: 0.6994 - accuracy: 0.7965 - val_loss: 2.9724 - val_accuracy: 0.2200
Epoch 9/15
68/68 [==============================] - 159s 2s/step - loss: 0.4188 - accuracy: 0.8727 - val_loss: 3.3843 - val_accuracy: 0.2600
Epoch 10/15
68/68 [==============================] - 160s 2s/step - loss: 0.3277 - accuracy: 0.8996 - val_loss: 3.3578 - val_accuracy: 0.2100
Epoch 11/15
68/68 [==============================] - 159s 2s/step - loss: 0.2923 - accuracy: 0.9294 - val_loss: 2.9347 - val_accuracy: 0.2400
Epoch 12/15
68/68 [==============================] - 158s 2s/step - loss: 0.2086 - accuracy: 0.9368 - val_loss: 3.9335 - val_accuracy: 0.2400
Epoch 13/15
68/68 [==============================] - 158s 2s/step - loss: 0.1959 - accuracy: 0.9414 - val_loss: 3.7110 - val_accuracy: 0.2200
Epoch 14/15
68/68 [==============================] - 158s 2s/step - loss: 0.1723 - accuracy: 0.9507 - val_loss: 3.4075 - val_accuracy: 0.2300
Epoch 15/15
68/68 [==============================] - 159s 2s/step - loss: 0.1364 - accuracy: 0.9554 - val_loss: 3.9858 - val_accuracy: 0.2900
```

Figure 3.22 CNN parameter results with 15 epochs

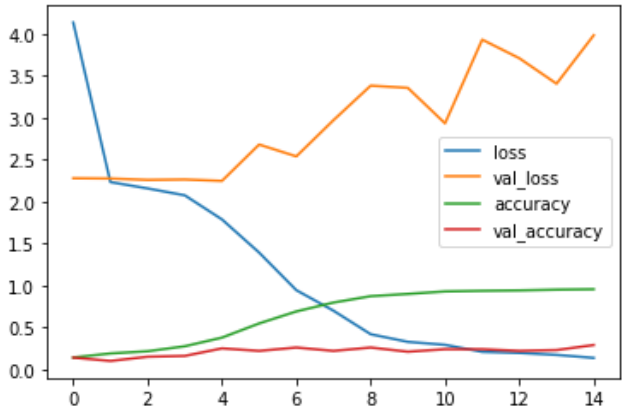


Figure 3.23 Graphical representation of CNN parameters results

At the ending, we can obtain embedded model, to achieve better results in accuracy:

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_4 (Embedding)      (None, None, 64)          640000

dense_14 (Dense)             (None, None, 64)          4160

dense_15 (Dense)             (None, None, 64)          4160

bidirectional_2 (Bidirectio  (None, None, 64)          6208
nal)

bidirectional_3 (Bidirectio  (None, 32)                2592
nal)

dense_16 (Dense)             (None, 1)                 33

=================================================================
Total params: 657,153
Trainable params: 657,153
Non-trainable params: 0
```

Figure 3.24 Embedded model with parameters

From the sequential fitting we can obtain best accuracy value:

```
Epoch 1/10
279/279 [==============================] - 19s 57ms/step - loss: 0.1668 - accuracy: 0.9370
Epoch 2/10
279/279 [==============================] - 18s 64ms/step - loss: 0.0387 - accuracy: 0.9901
Epoch 3/10
279/279 [==============================] - 16s 58ms/step - loss: 0.0121 - accuracy: 0.9980
Epoch 4/10
279/279 [==============================] - 16s 57ms/step - loss: 0.0107 - accuracy: 0.99661s - los - ETA: 0s - loss: 0
Epoch 5/10
279/279 [==============================] - 17s 62ms/step - loss: 0.0021 - accuracy: 0.9993
Epoch 6/10
279/279 [==============================] - 18s 65ms/step - loss: 5.9325e-04 - accuracy: 1.0000
Epoch 7/10
279/279 [==============================] - 16s 59ms/step - loss: 3.8283e-04 - accuracy: 1.0000
Epoch 8/10
279/279 [==============================] - 15s 54ms/step - loss: 2.8130e-04 - accuracy: 1.0000
Epoch 9/10
279/279 [==============================] - 16s 56ms/step - loss: 2.1350e-04 - accuracy: 1.0000
Epoch 10/10
279/279 [==============================] - 15s 54ms/step - loss: 1.6823e-04 - accuracy: 1.0000
```

Figure 3.25 Accuracy results after last 10 epochs

We can represent the 1 and 0 values from all neural dataset:

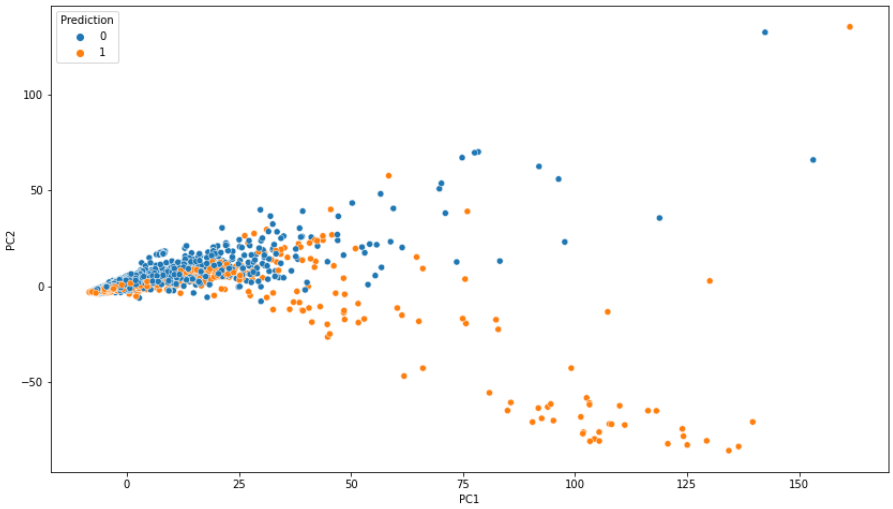


Figure 3.26 Prediction points representation

Let’s find distributed stochastic neighbor embedding using TSNE-1 and TSNE-2 dataset columns:

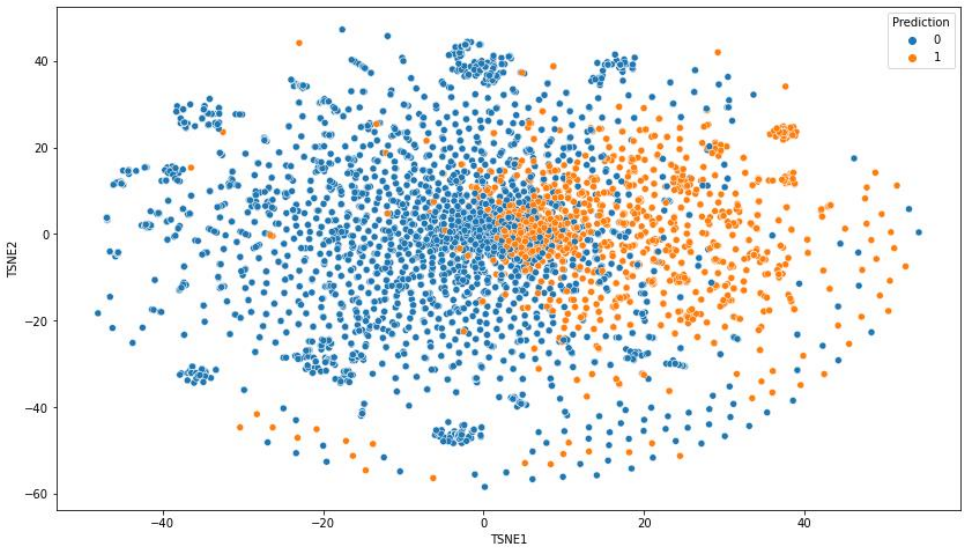


Figure 3.27 TSNE results from two sets

The EM expectation maximization graph:

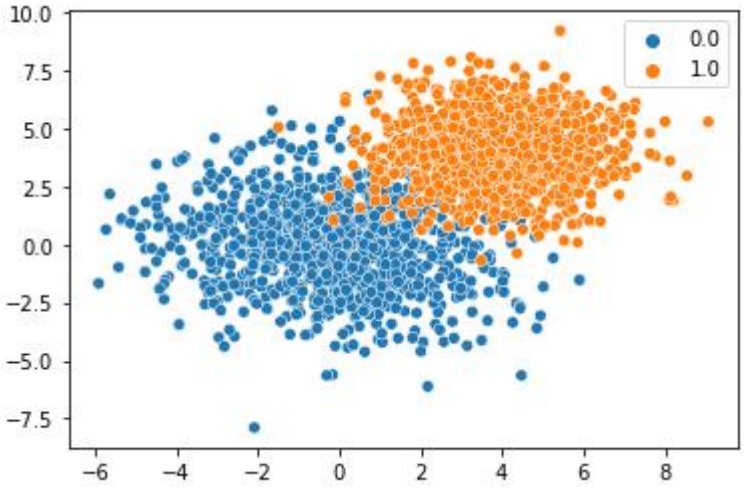


Figure 3.28 EM representation

Obtaining latest accuracy from classification confusion matrix:

```
Confusion matrix:
[[950   0]
 [ 22 143]]
Classification report:
              precision    recall  f1-score   support

           0       0.98      1.00      0.99       950
           1       1.00      0.87      0.93       165

    accuracy                           0.98      1115
   macro avg       0.99      0.93      0.96      1115
weighted avg       0.98      0.98      0.98      1115
```

Figure 3.29

So, after a set of epochs and ensembles of different algorithms: AdaBoost, SVM, Kernel Factory we can sort results and know that accuracy = 0,98.

Additionally find LDA accuracy arrays from all methods:

```
array([[0.12478774, 0.75555864, 0.11965362],
       [0.12682139, 0.11286744, 0.76031118],
       [0.16823723, 0.48915762, 0.34260514],
       [0.13131801, 0.74204858, 0.12663341]])


array([[0.75655209, 0.17673878, 0.06670913],
       [0.24446442, 0.68344126, 0.07209433],
       [0.44018391, 0.10129395, 0.45852214],
       [0.85362806, 0.07210483, 0.07426711]])
```

Figure 3.30 Accuracy array results with LDA module

In next chapter we describe the user interface and coding structures from this work. All results will get using Python 3.0, tensor flow, NumPy and additional libraries. Best accuracy was 0.98.

CHAPTER 4. PYTHON BASED SOFTWARE DEVELOPMENT

4.1 Implementation of the program's lock scheme

The scheme was implemented using the built-in modules of Pandas, NumPy, MatPlotlib, Seaborn, PIL, and sklearn libraries. By reading the SQL database, opening the CSV file, the database with the set of occurring phrases was processed.

The paper also used graphs to display the metrics and data collection of the required methods and algorithmic approaches of ensemble boosting.

Using programming packages:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

from sklearn.preprocessing import normalize
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.svm import SVC
from sklearn.ensemble import AdaBoostClassifier, RandomForestClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import classification_report, confusion_matrix

%matplotlib inline
```

Figure 4.1 First part of needed program libraries and modules for solving the problem

Also, we have second part with better graphical representation in other .py file:

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import tensorflow as tf
import string

from tensorflow import keras

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import classification_report, confusion_matrix

from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.layers import Dense, Conv2D, Flatten, MaxPooling2D, Dropout
from tensorflow.keras.callbacks import EarlyStopping

from nltk.corpus import stopwords

%matplotlib inline
```

Figure 4.2 Graphical representation of second part of needed modules

Programming interface include taskbars, debugger, consoles, error point measure line, progress bar, program window and selected files with databases. Other frames represent representative work of tensor flow libraries.

4.2 User interface with selection of the desired algorithmisation

CHAPTER 5. ENVIRONMENTAL PROTECTION.

5.1 Recommendations for the use of computing power for environmental protection.

Addressing today's global challenges such as biodiversity loss, global change and increasing demand for ecosystem services requires better environmental forecasting. The increasing availability and computational power of data is facilitating the development of quantitative methods in ecology. However, a flexible methodological framework is needed to apply these advances to environmental forecasting. Deep Learning (DL) is a branch of Machine Learning (ML) that is rapidly gaining popularity but has not yet been widely applied in ecology. It involves training Deep Neural Networks (DNNs), which are artificial neural networks composed of multiple layers and a large number of neurons. This paper presents an example (with code and data) of designing, training and using DNNs for environmental prediction. Using the example of bark beetle occurrence in coniferous forests, the authors show that DNNs can very well predict both short-term and local risk of infestation and long-term dynamics at larger scales. The paper also shows that DNNs are superior to standard methods in predicting bark beetle dynamics and have great potential for developing a comprehensive forecasting system in this area.

Machine learning is a family of computer algorithms designed to identify patterns in complex, often nonlinear, data and build accurate predictive models from that data. Compared to classical statistical approaches, such as regression, machine learning focuses on identifying and describing complex relationships and has predictive capabilities in parameter estimation and confidence intervals. Machine learning, at the intersection of computer science and statistics and the core of artificial intelligence and data science, is a rapidly growing field.

Deep learning is a relatively new area of ML. The main DL tool is the deep neural network (DNN). It is based on artificial neural networks (ANNs) invented in the middle of the last century. Essentially, DL is a set of methods by which large (more neurons) and deeper (more layers) neural networks can be trained. These networks are made possible by the development of improved algorithms for optimizing the weights that connect neurons (e.g., stochastic gradient descent), more available processing power, and more training data. Although these improvements may seem minor, today's DNNs not only outperform their simpler predecessors, but often outperform other ML approaches in standard tests of prediction accuracy.

Until a decade ago, ML was hardly used in ecology, but in recent years its popularity has skyrocketed. However, the potential of ML is far from being fully exploited, and there are currently very few applications of deep learning in ecology (Figure S1). The aim of this work is to contribute to the wider dissemination of deep learning in ecology by demonstrating its potential in prediction. As an example, the authors of this paper chose the task of predicting bark beetle outbreaks in conifer-dominated forests.

Convolutional neural network (CNN) is a type of neural network often used for pattern recognition in image or time series data. In addition to the full link layers, the CNN includes convolution and fusion layers. In the convolutional processing layer, filters are applied sequentially to all parts of the input data, using the same weights. Subsequent design layers combine the output data from the coalescing layers, making them less sensitive to small shifts and distortions in the data. A series of such convolution and aggregation layers allows for the extraction of fairly high-quality properties from the data.

During DNN training, the weights are iteratively updated to minimize the prediction error. To measure the prediction accuracy on new input data, the data is divided into a training set and a test set. The details of the network architecture, such as the size of the network, the choice of specific layer types, and the

parameters of the training process, largely determine the prediction accuracy and are generally task-specific. Section S2 of the Appendix provides guidance on DNN design and training, as well as practical considerations for DNN applications.

5.2 Using neural networks to track diseases and their factors

Deep learning is well suited for generalizing beyond experimental data, which is essential for making predictions about applied ecological problems. A particular strength of DL is its ability to achieve a high level of abstraction using raw data. Furthermore, deep learning can help improve traditional environmental modelling methods for forecasting tasks. Another promising approach is combining deep learning with process-based models to achieve a better understanding of environmental processes.

Machine learning in general, and DNNs in particular, are often criticized for their "black box" nature - it is difficult to intuitively interpret a trained model and its weights. As a result, more traditional models continue to be an important tool for improving understanding, especially of causal relationships in nature. It should be noted, however, that these classical approaches make a priori assumptions about the pattern of the data that may not reflect the actual relationship between cause and effect. Although traditional models provide more understandable results and have a more rigorous basis for hypothesis testing, they often do not reflect reality as accurately. Machine learning can more accurately describe cause and effect relationships without making a priori assumptions.

DNNs work particularly well with environmental data because they can effectively combine different types of data (e.g., images, numeric and categorical variables). Furthermore, their hierarchical multi-layered structure reflects the fact that ecosystems are often governed by multiple processes in hierarchical relationships. In this study, DNNs outperformed all other methods for modeling

the dynamics of the bark beetle epidemic (with the exception of the random forest algorithm).

Methods based on logical rules make it possible to take into account different aspects (semantic, structural, punctuation) of individual words and the language itself, but their application encounters a number of problems:

- A particular collection of various linguistic rules that must take into account very different constructive linguistic properties. This aspect requires a team of linguists.

- The narrow scope of the set of rules is due to the following.

The format of writing various messages on the Internet differs slightly from the accepted norms of the Russian language in literary form. Messages posted on social networks are characterized by the presence of punctuation and spelling errors, the presence of various types of verbal errors and jargon, strange punctuation marks, and the use of special symbols and graphics to strengthen the emotional distance of the text.

- Attachment to the language of the analyzed text is always related to unique linguistic structures are unique and cannot be transferred and applied to other languages. The use of a rule-based approach can only be effective if the text being analyzed is grammatically correct and if the various constructions used in the language being analyzed are covered by a corpus of rules.

5.3 Using self-learning systems to compute statistics of environmental damage

Creating a fully-fledged artificial intelligence, similar to the human brain, is an incredibly difficult task for scientists. They have not yet achieved it, but there is already a lot of development and research in this area.

A lot of energy is required to train an artificial intelligence. And it will grow over time. However, researchers say that even at the current level, the process of creating artificial intelligence causes a lot of damage to the environment.

To verify this, the scientists ran four different machine learning programs on a single graphics processor. They then measured the amount of energy consumed by each program. The researchers knew how much energy was consumed and also calculated how much carbon dioxide was produced.

The more complex the programs were, the more power they required and the more energy they consumed. At the same time, the amount of harmful emissions increased. It was found, for example, that the machine learning process emitted five times more carbon dioxide equivalents than the lifetime of a car.

Environmental engineers develop methods to solve environmental problems. They participate in local and global environmental efforts, such as air and water pollution control, recycling and waste management.

Depending on the focus of their work, their duties may include collecting soil or groundwater samples and analyzing them for contamination, designing municipal wastewater and industrial waste treatment systems, analyzing scientific data, investigating controversial projects, and conducting quality control studies. They may provide legal or financial advice on procedures, equipment or environmental problems. They can study the effects of large-scale problems such as acid rain, global warming and ozone depletion and try to minimise them.

They will work with other engineers and scientists to solve large-scale problems. Teamwork is almost always part of any engineer's daily routine, but it is particularly important for environmental engineers, who often work alongside civil, mechanical and other engineers.

CHAPTER 6. LABOR PROTECTION

6.1 Analysis of the use of software in the workplace

The engineers created a convolutional neural network and taught it to recognize people in streaming video, identify details of equipment - helmets, vests, cables - and types of production spaces. In the trial version, the system records and responds to the three most common worker behavior scenarios:

Whether the worker is wearing a helmet on his or her head - the workplace norm;

Whether the worker is wearing the hood of the work coat over his or her helmet - this is strictly prohibited;

Whether the worker is tied up with a rope - mandatory when working at high altitudes.

A common problem in industrial machine learning projects is that, due to the novelty of the subject and the sporadic nature of implementations, there are no initial sets of templates for training neural networks. We had to develop and mark from scratch a reference dataset containing 45 sequences of positive and negative scenarios of production personnel behavior.

The video stream is processed in three steps. First, we filter the frames in which no people are present. Then, the parts of the video in which the system recognizes people are passed to a convolutional neural network. The network identifies the person by the markings, and identifies the elements of holding: a helmet on the head or a rope on the torso. The algorithm, which uses the reference vector method, then compares the subject's image to the patterns in the database. If the frame contains violations, the system sends a notification according to regulations.

The mask RNN was used for image segmentation. This frame can recognize all defined classes of objects and select the objects in the frame. The neural network was trained using a mapping script, which is optimal when working with a limited dataset and we were not burdened with collecting statistics about the work: how many employees are on site, which departments, which locations do they spend more time.

The final version was built with robust video stream analysis, object recognition and behavior-based classification. The accuracy is 82-95 percent. Our pilot project produced excellent results in the testing phase and now the customer is continuing the testing. We will keep an eye on further developments as the road from pilot project to industrial solution is very long.

Here is an example from our practice: the manager of a wood processing company is concerned about a manufacturing defect - a deformation of a wooden beam, the quality of which has been checked by a special employee. We propose to solve this problem by introducing machine learning and computer vision algorithms to automatically detect such defects.

We place a special device with an optical sensor at a certain point in the manufacturing process that films the process, and a special software algorithm analyzes the video and automatically identifies the beam profile. If the sample is curved, the device sends a signal to the inversion and classification mechanism.

The new project inversion mechanism will result in an entry in the database or CRM: how many problems were detected during the working day, at which location and at what time. The customer can now improve their financial performance and calculate the financial benefits of automation.

6.2 Electrostatics, electronic components and safety when working with computer technology

Computer should be ergonomically designed and equipped so that the information displayed can be read safely and comfortably under the conditions of use:

- Computer shall be designed so that the computer housing, when fixed in a particular position, can be rotated horizontally and vertically so that the screen is visible from the front;

- Computer should be designed with soft, muted colours and diffused light, i.e. light scattered in all directions;

- Computer housing shall have an opaque surface with a coefficient of reflection of (0,4-0,6) and no shiny reflective parts;

- Computer shall be designed so that brightness and contrast are adjustable.

Documentation shall be available on the operation of new (improved) computers.

Workstations shall be positioned so that the computer screen faces sideways towards the light apertures (except for peripheral workstations), so that natural light is mainly from the left.

Artificial lighting in computer rooms shall be provided by a general, uniform lighting system. In production, administration and public areas where documents are primarily handled, a combined lighting system shall be used (in addition to general lighting, local luminaires shall be installed to illuminate the document area).

The lighting shall not cause glare on the screen surface. Glare from direct light sources and from light sources reflecting from work surfaces (screens, desks, keyboards, etc.) shall be reduced by appropriate selection of the type of luminaires and by the arrangement of workstations in relation to natural and artificial light sources.

Luminaires used for local lighting shall be fitted with an anti-glare reflector with an angle of at least 40°.

LB fluorescent tubes and compact fluorescent tubes shall be used as light sources for artificial lighting. Metal halide lamps may be used for indirect lighting in industrial, administrative and public buildings. Incandescent lamps, including halogen lamps, may be used in local luminaires.

Electronic ballast luminaires with parabolic reflector optics (hereinafter referred to as EBG) are used for lighting computer rooms. Multi-lamp luminaires may be used with EBGs having the same number of front and rear branches.

6.3 Failure systematics and hazards in neural network operation

Today's fourth industrial revolution is generally focused on automation through technology and intelligent systems, with applications in many areas, including smart healthcare, business intelligence, smart cities, and smart cybersecurity.

The power of deep learning approaches has increased dramatically across a wide range of applications, particularly in security technology as an excellent solution for exploring complex architectures using high-dimensional data. Thus, deep-learning techniques can play a key role in creating data-driven intelligent systems that meet today's needs due to their excellent capabilities to learn from previous data. Consequently, DL can transform the world and people's daily lives through the power of automation and experiential learning.

Consequently, DL technology is relevant to artificial intelligence, machine learning, and data science with advanced analytics, which are well-known areas of computer science, especially in modern smart computing. In what follows, we first review the place of deep learning in artificial intelligence and how DL technology relates to this area of computer technology.

6.4 Use of machine learning for occupational safety prevention

Machine learning systems packaged as predictive analytics systems are available on the market that use data from past events and various variables to predict future trends. Maximizing the use of predictive analytics systems in your organization requires careful input of historical incident data, as well as a variety of other sources of safety analytics data, such as safety audits, surveys, and integrated personnel data related to the context of the situation and environment.

The data is then analyzed and used to predict future behavior and alert management before an accident occurs. Your organization's predictive analytics system should use data based on a variety of factors.

In addition, occupational safety affects your company's reputation and, therefore, its ability to attract the best employees, contracts and investors.

6.5 Computer room fire safety

From a financial perspective alone, it's easy to see why finding new technologies to reduce workplace accidents and increase productivity is essential, and why leveraging machine learning with predictive data analytics is the next big step in developing a proactive safety culture. The safety professionals of the future can rely on machine learning and predictive analytics as a key part of their profession to provide up-to-date information for preventive maintenance, reducing injuries and developing new safety practices and policies.

If your organization has a safety-focused culture with a focus on continuous process improvement, you are ready to take the next step and consider implementing machine learning and predictive data analytics as part of your organization's operations.

CONCLUTIONS

1. The necessity of using machine learning with the partial involvement of teacher ssl, to create an intelligent classifier is substantiated.
2. Proposed mathematical model of neural network on the basis of boosting, which makes it possible to increase the accuracy of solving the problem of classification.
3. Developed an algorithm for machine learning with partial teacher involvement based on boosting.
4. Computer program for building a neural network based on booming is developed. Learning algorithms for neural networks with SSL were supplied.
5. Synthesis of an algorithm for building an ensemble of neural networks with SSL on boosting stage was performed.
6. The new algorithm is developed by means of the software and the block diagram of the program with the additional user interface is introduced.

REFERENCE

1. N. Avinash, Neural Network Models in R, 2019. [Online]. Available: www.datacamp.com/community/tutorials/neural-network-models-r

2. D. Luke, What Is an Artificial Neural Network? Here's Everything You Need to Know, 2019. [Online]. Available: www.digitaltrends.com/cool-tech/what-is-an-artificial-neural-network/

3. D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," nature, vol. 323, no. 6088, pp. 533–536, 1986.

4. B. Jason, A Gentle Introduction to Computer Vision, 2019. [Online]. Available: www.machinelearningmastery.com/what-is-computer-vision/

5. Technopedia, What Is Computer Vision? - Definition from Techopedia, 2019. [Online]. Available: www.techopedia.com/definition/32309/computer-vision

6. S. J. Prince, Computer vision: models, learning, and inference. Cambridge University Press, 2012.

7. T. Celik and H. Kusetogullari, "Solar-powered automated road surveillance system for speed violation detection," IEEE Transactions on Industrial Electronics, vol. 57, no. 9, pp. 3216–3227, 2009

8. Z. Deng, H. Sun, S. Zhou, J. Zhao, L. Lei, and H. Zou, "Multi-scale object detection in remote sensing imagery with convolutional neural networks," ISPRS journal of photogrammetry and remote sensing, vol. 145, pp. 3–22, 2018.

9. D. Xiao, F. Shan, Z. Li, B. T. Le, X. Liu, and X. Li, "A target detection model based on improved tiny-yolov3 under the environment of mining truck," IEEE Access, vol. 7, pp. 123 757–123 764, 2019.

10. S. Smriti, Computer Vision Makes Autonomous Vehicles Intelli1gent and Reliable, 2019. [Online]. Available: www.analyticsinsight.net/computer-vision-makes-autonomous-vehicles-intelligent-and-reliable/

11. G. Lewis, "Object detection for autonomous vehicles," 2014.

12. V. Marco, What Is Machine Learning? A Definition, 2017. [Online]. Available: www.expertsystem.com/machine-learning-definition/

13. I. Goodfellow, Y. Bengio, and A. Courville, Deep learning. Book in preparation for MIT Press, 2016. [Online]. Available: http://www.deeplearningbook.org
14. B. Volodymyr, Machine Learning Algorithms: 4 Types You Should Know,2018. [Online].Available:www.theappsolutions.com/blog/development/machine-learning-algorithm-types/
15. S. Ray, Essentials of machine learning algorithms (with python and r codes), 2017. [Online]. Available: http://www.analyticsvidhya.com/blog/2015/08/common-machine-learning-algorithms
16. C. M. Bishop, Pattern recognition and machine learning. springer, 2006.