



TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

JUHO UUSI-LUOMALAHTI
CLASSIFICATION OF WETLAND VEGETATION BASED ON TER-
RASAR-X DATA: COMPARISON OF METHODS

Diplomityö

Tarkastaja: Prof. Tarmo Lipping
Tarkastaja ja aihe hyväksytty
Talouden ja rakentamisen tiedekun-
nan tiedekuntaneuvoston kokouk-
sessa 5. lokakuuta 2016

TIIVISTELMÄ

JUHO UUSI-LUOMALAHTI: Classification of Wetland Vegetation Based on TerraSAR-X Data: Comparison of Methods

Tampereen teknillinen yliopisto

Diplomityö, 64 sivua, 6 liitesivua

Elokuu 2016

Tietotekniikan diplomi-insinöörin tutkinto-ohjelma

Pääaine: Ohjelmistotuotanto ja tiedonhallinta

Tarkastaja: Prof. Tarmo Lipping

Avainsanat: hahmontunnistusmenetelmät, kasvillisuuden luokittelu, SAR-tutkakuva, Random Forest, Support Vector Machine, Multilayer Perceptron ensemble

Tämän diplomityön tarkoituksena oli tutkia erilaisten hahmontunnistusmenetelmien soveltuvuutta vesikasvillisuuden ja maan pinnan luokitteluun SAR-satelliittitutkakuvista. Tutkakuvien automaattinen luokittelu mahdollistaa tehokkaan ja nopean maanpinnan ja kasvillisuuden kartoituksen ilman aikaa vieviä ja hitaita kenttä-tutkimuksia. SAR-tutkatekniikan avulla alueen monitorointi on mahdollista riippumatta kellonajasta tai pilvisyydestä toisin kuin käytettäessä näkyvän valon aallonpituusalueella toimivia passiivisia sensoreita. Diplomityö on osa Olkiluodon ydinvoimalan käytetyn polttoaineen loppusijoituksesta vastaavan Posiva Oy:n biosfäärin mallinnusprojektia.

Testiaineistona diplomityössä käytettiin Satakunnassa sijaitsevalta Poosjärveltä 4. heinäkuuta 2014 otettua SAR-satelliittitutkakuvaa. Poosjärvi on pieni 794 hehtaarin kokoinen järvi, joka sijaitsee noin 20 km Porin pohjoispuolella Pomarkun ja Noormarkun alueella. Poosjärven kasvillisuus koostuu pääosin saraheinistä, järvikortteesta, järvi-ruo'osta sekä vita- ja lummekasveista. Metsikkö on sekametsää. Luokittelussa käytettiin seitsemää eri luokkaa: järviruoko, järviruoko vedessä, järvikorte, vesi, puusto, saraheinä sekä ulpukka.

Diplomityössä tutkittiin neljän erityyppisen luokittimen soveltuvuutta tehtävään: Random Forest, Support Vector Machine, Multilayer Perceptron ensemble ja K-Nearest Neighbor. Luokittelussa käytetyt piirteet ovat peräisin kuvankäsittelymenetelmiä ja kaukokartoitusaineiston luokittelua käsittelevästä kirjallisuudesta. Lopputuloksena havaittiin, että erityisesti tekstuuripohjaisten piirteiden merkitys korostui luokittelun onnistumisen kannalta, ja että testatut luokittelumenetelmät osoittautuivat käyttökelpoisiksi kyseessä olevaan tehtävään. Paras luokittelutulos saavutettiin Multilayer Perceptron ensemble -luokittimella, joskin myös Random Forest ja Support Vector Machine -luokittelijat suoriutuivat hyvin. Avoimeksi kysymykseksi jäi, jakoivatko luokittelualgoritmit testiaineiston eri luokkiin maaperän, maanpinnan vai kasvillisuuden ominaisuuksien perusteella.

ABSTRACT

JUHO UUSI-LUOMALAHTI: Classification of Wetland Vegetation Based on TerraSAR-X Data: Comparison of Methods

Tampere University of Technology

Master of Science Thesis, 64 pages, 6 Appendix pages

August 2016

Master's Degree Programme in Information Technology

Major: Software Engineering

Examiner: Professor Tarmo Lipping

Keywords: pattern recognition, classification of vegetation, SAR-image, Random Forest, Support Vector Machine, Multilayer Perceptron ensemble

The aim of this Master's thesis was to study the feasibility of different pattern recognition techniques for classification of wetland vegetation and ground cover from SAR images. Use of pattern recognition and remote sensing techniques provides an efficient and fast way to survey large areas of earth's surface without time consuming and difficult field studies. Utilization of SAR-technology enables monitoring of the area regardless of time or weather conditions unlike passive radiometers operating in visible light spectrum. The thesis is part of the biosphere modelling project of Posiva Oy, the instance responsible for spent nuclear fuel disposal of the nuclear power plant in Olkiluoto.

The test data for the classification algorithms consisted of a SAR image obtained from Lake Poosjärvi on 4th of July 2014. Lake Poosjärvi is a small (area of 794 hectares) lake located in the western part of Finland in the Satakunta region. Forest type at Lake Poosjärvi is mainly mixed forest, and ground vegetation is dominated by sedges, water horsetail, water lilies, pondweeds and reed. Seven classes were used in classification: reed, reed in water, water horsetail, water, trees, sedge and yellow water lily.

In the thesis, the feasibility of four different classification algorithms was studied: Random Forest, Support Vector Machine, Multilayer Perceptron ensemble and K-Nearest Neighbor. Features used in classification were selected from image processing and remote sensing literature. It was found, that the classification methods used in the thesis were applicable to the task, and that texture-based features had significant positive impact on classification results. The best classification results were achieved using the Multilayer Perceptron ensemble classifier, though Random Forest and Support Vector Machine produced good results as well. The question whether the classifiers were classifying the test data according to the properties of soil, ground cover or vegetation, remains open.

ALKUSANAT

Tämän diplomityön käytännön osuus on tehty Tampereen teknillisen yliopiston Porin laitoksen signaalinkäsittelyn laboratoriossa kesän 2015 aikana. Diplomityössä tehty tutkimus liittyy Posiva Oy:n biosfäärin mallinnusprojektiin.

Haluan esittää kiitokseni työn ohjaajalle ja tarkastajalle professori Tarmo Lippingille mielenkiintoisen aihealueen tarjoamisesta diplomityön tutkimuskohteeksi. Kiitokseni esitän myöskin DI Teemu Kumpumäelle tieteellis-teknillisestä avusta ja konsultaatiosta liittyen diplomityön toteuttamiseen sekä Posiva Oy:lle projektin rahoituksesta. Kiitokset myös vanhemmilleni opintojeni aikaisesta taloudellisesta subventiosta, joka on pelastanut minut keskivaikealta aliravitsemukselta ja telttamajoitukselta. Lopuksi kiitokset TTY:n Porin laitokselle ja sen henkilökunnalle sujuvan ja joustavan opiskeluilmapiirin tarjoamisesta.

Huolimatta siitä, että tämän(kin) diplomityön teoria- ja kirjoitusosuutta on rakennettu kuin iisakin kirkkoa, on kyseinen vaihe viimein saatu kunniakkaaseen päätökseensä. Tämän positiivisen tapahtuman johdosta valmistumistodennäköisyyteni on kohonnut poikkeuksellisen korkeahkoon lukemaan.

Raumalla, 30.8.2016

Juho Uusi-Luomalahti

CONTENTS

1. INTRODUCTION	1
2. EARTH OBSERVATION WITH TERRASAR-X.....	3
2.1 Synthetic Aperture Radar Imaging.....	4
2.1.1 Imaging Geometry	7
2.1.2 Pulse Transmission and Receiving	8
2.1.3 SAR Signal Processing	9
2.1.4 Imaging Modes	11
2.1.5 Geometrical Distortions and Speckle Noise	12
2.1.6 SAR Data Preprocessing and Correction	14
2.2 TerraSAR-X Satellite Program	14
3. PATTERN RECOGNITION AND CLASSIFICATION METHODS IN REMOTE SENSING DATA ANALYSIS	17
3.1 Models.....	18
3.2 Features	19
3.3 Supervised and Unsupervised Learning.....	20
3.4 Overfitting and the Curse of Dimensionality	20
3.5 Evaluation of Classification Results	23
3.6 Ensemble Methods	24
3.6.1 Bagging	25
3.7 Decision Trees and Random Forest	26
3.7.1 Random Forest	28
3.8 Support Vector Machine	29
3.8.1 Soft-Margin Non-Linear SVM.....	30
3.9 Multilayer Perceptron Neural Network.....	32
3.9.1 Architecture.....	33
3.9.2 Training.....	36
3.10 K-Nearest Neighbor Classifier	38
4. FEATURE SET SELECTION AND COMPARISON OF CLASSIFICATION TECHNIQUES USING TERRASAR-X TEST DATA.....	40
4.1 Description of the Test Area and the Data	40
4.2 Feature Set Used in Classification	42
4.3 Visualization of Features.....	49
4.4 Comparison of Classification Techniques	52
4.4.1 Random Forest	53
4.4.2 Support Vector Machine	55
4.4.3 Multilayer Perceptron Neural Network Ensemble.....	58
4.4.4 K-Nearest Neighbor	60
5. CONCLUSIONS.....	63
BIBLIOGRAPHY	65

APPENDIX 1: RANDOM FOREST CLASSIFIER CONFUSION MATRIX AND STATISTICAL MEASURES FOR THE TEST DATA.....	67
APPENDIX 2: SVM CLASSIFIER CONFUSION MATRICES AND STATISTICAL MEASURES FOR THE TEST DATA	68
APPENDIX 3: MLP ENSEMBLE CLASSIFIER CONFUSION MATRIX AND STATISTICAL MEASURES FOR THE TEST DATA.....	70
APPENDIX 4: KNN CLASSIFIER CONFUSION MATRICES AND STATISTICAL MEASURES FOR THE TEST DATA	71

ABBREVIATIONS

AI	Artificial Intelligence
ANN	Artificial Neural Network
Bagging	Bootstrap Aggregating
DLR	German Aerospace Center (Deutsches Zentrum für Luft- und Raumfahrt)
DNF	Disjunctive Normal Form
GLCM	Gray-Level Co-Occurrence Matrix
GPS	Global Positioning System
KNN	K-Nearest Neighbor
LBP	Local Binary Pattern
LCT	Laser Communication Terminal
LiDAR	Light Detection and Ranging
ML	Machine Learning
MLP	Multilayer Perceptron
PCA	Principal Component Analysis
PR	Pattern Recognition
PRF	Pulse Repetition Frequency
PRI	Pulse Repetition Interval
RBF	Radial Basis Function
RF	Random Forest
SAR	Synthetic Aperture Radar
SVM	Support Vector Machine

1. INTRODUCTION

The aim of this thesis is to study the feasibility of pattern recognition techniques for classification of wetland vegetation and ground cover from remotely sensed SAR-images. Remote sensing is information collection from remote objects without making physical contact with them. Remote sensing is mainly referring to various space or airborne imaging techniques to survey the surface of the Earth. Co-operative use of remote sensing and pattern recognition provides an efficient and fast way to survey large areas of Earth's surface without time consuming and difficult field studies.

The field of remote sensing can be divided into active and passive remote sensing. In active remote sensing, imaged objects are illuminated with electromagnetic waves emitted from an airplane or satellite (radar or LiDAR systems). Advantage of active radar based remote sensing systems is that they are independent of time (night time operation is possible) and weather conditions (radar pulses can penetrate through clouds). In passive remote sensing, objects are observed by passively collecting electromagnetic waves (camera systems) scattered from objects. Passive remote sensing systems operating in visible or near visible light regions of the electromagnetic spectrum are prone for weather and lighting conditions.

In this thesis, an active remote sensing technique called SAR is utilized. SAR is an acronym for **S**ynthetic **A**perture **R**adar. In general, SAR is a pulse radar operating in the micro-wave region mounted on a moving platform like a satellite or an airplane. Contrary to ordinary radar systems, SAR is capable of producing fine resolution images from imaged objects by utilizing digital signal processing techniques. The result of SAR imaging is a 2D image, where brightness of each pixel corresponds to the intensity of backscattered radio wave pulse.

Another topic discussed in the thesis is machine learning and pattern recognition. Pattern recognition is a branch of machine learning, in which the goal is to discover regularities in the data in order to provide some useful information about the target the data was acquired from. A pattern recognition algorithm consists of a mathematical model and model parameters. The aim is to fit the model parameters to data in such a way that reliable predictions can be made for new unseen data in a particular task. Usually model parameters are tuned by using some heuristic search/optimization methods. Pattern recognition techniques have been utilized in very diverse range of applications: computer aided medical diagnosis, speech recognition, spam filters and optical character recognition, just to name a few.

In this thesis, four different pattern recognition algorithms are used to classify vegetation and ground cover from a SAR-image. The SAR-image was acquired by the TerraSAR-X satellite from Lake Poosjärvi. Lake Poosjärvi is a small (area of 794 hectares) lake located in the western part of Finland in the Satakunta region. The forest type at Lake Poosjärvi is mainly mixed forest, and ground vegetation is dominated by sedges, water horsetail, water lilies, pondweeds and reed. In the classification of ground cover and vegetation seven different classes were used: reed, reed in water, water horsetail, water, trees, sedge and yellow water lily.

The goal behind vegetation and ground cover classification is to be able to calculate biomass amounts automatically from the SAR image acquired from Lake Poosjärvi. Biomass calculation is a part of the biosphere modelling project of Posiva Oy, the instance responsible for spent nuclear fuel disposal of the nuclear power plant in Olkiluoto. Lake Poosjärvi is of interest since it is expected that a lake with similar conditions will be formed also in Olkiluoto in the vicinity of the spent nuclear fuel disposal cave as a result of post-glacial land uplift. The aim of biosphere modelling is to predict how much radioactive particles will end up into the ecosystem and human food chain in the case of hypothetical radioactive particle leakage from the disposal cave.

Contents of the thesis are organized as follows: the principle of SAR is described in chapter 2, the theory of pattern recognition and used pattern recognition techniques are discussed in chapter 3, the test area and data, features, parameters of classifiers and results are described in chapter 4, conclusions are represented in chapter 5.

2. EARTH OBSERVATION WITH TERRASAR-X

The earliest remote sensing applications for Earth surface observation were aerial photographs taken from airborne platforms like balloons or early airplanes. The first aerial photographs were taken from Paris in 1858 by balloonist named G. Tournachon. Although aerial photographs were available as early as 1858, the field of aerial photography remained quite undeveloped until World War 1, when military reconnaissance purposes started to drive the field forward. Next big leap in remote sensing technology was the development of satellites in the latter half of the 20th century. Before satellites, remote sensing of the Earth was mainly conducted locally with airplanes. Satellite technology moved perspective in remote sensing from local scale to global and even extra-terrestrial scale. For example, NASA Magellan spacecraft has provided topographic maps from Venus using synthetic aperture radar (SAR). The field of remote sensing has grown steadily, and nowadays remotely sensed data is sold also on commercial markets whereas in the early days of remote sensing data were only available for military and scientific purposes. [1]

Spaceborne remote sensing of the Earth's surface has its roots in military technology and NASA satellites launched in 1960s. First remote sensing satellites employed optical cameras and passive radiometers as imaging devices. In 1972, NASA launched the first Landsat satellite, which was capable of optical imaging using visible light and infrared spectral bands. The launch of Landsat 1 was followed by the series of launches of Landsat 2, Landsat 3 and Landsat 4 in 1975, 1978 and 1982, respectively. The most recent satellite, Landsat 8, was launched in 2013. Landsats are operating in visible light and infrared bands and, because of this, are easily disturbed by weather (clouds) and lighting conditions (lack of solar illumination). This is a common issue in all sensors operating at visible light wavelengths; operation of these satellites is restricted only for daytime and for good weather conditions. These limitations can be overcome by using active microwave radars instead of passive radiometers. Radar pulses with longer wavelength (microwaves) are able to penetrate through the cloud coverage, and active target illumination used in radar systems enables operation also during the night time. One of the commonly used imaging radar techniques is called *Synthetic Aperture Radar (SAR)*, which is very feasible especially on spaceborne platforms. [2]

The history of spaceborne civilian SAR-satellites begins on June 1978 when NASA launched the SEASAT satellite. The satellite operated only for 106 days until short-circuit in the satellite's electrical systems ended the mission on 10th of October 1978. Although the technology of the satellite was quite unsophisticated and it operated only for 106 days, it proved that the SAR-concept was feasible for future missions. SEASAT

was followed by SIR-A and SIR-B space shuttle missions in 1981 and 1984, respectively. The 1980s were mainly dominated by space shuttle based missions utilizing derivatives of the SEASAT radar. During 1980s, only one SAR-satellite, the Soviet 1870 SAR, was launched to orbit. The era of SAR-satellites actually started in 1990s, when five separate SAR-satellites were launched along with the interplanetary spacecraft Magellan SAR. Nowadays SAR technology has been utilized for more than 30 years in a large range of remote sensing applications like geoscience, climate research, environmental monitoring, 2D and 3D mapping, ice monitoring and oceanography. [2] [3]

2.1 Synthetic Aperture Radar Imaging

RADAR is an acronym for **RA**dio **D**etection **A**nd **R**anging. Radars are self-illuminating remote object-detection systems, which utilize radio- or microwave regions of the electromagnetic spectrum in their operation. Applications of modern radar systems are very diverse: military applications, outer space observation, meteorology, remote sensing, marine radars, ocean surveillance, geological ground observation etc. [4]

First radars were developed in 1920s for ship and aircraft detection purposes. First pulse radar systems were developed simultaneously in 1930s in the United States, Germany and Great Britain. History of SAR begins in 1951 when the “Doppler beam-sharpening” concept was invented by Carl Wiley of Goodyear Corporation. The first experimental airborne SAR-system was built in Goodyear research facility in Litchfield, Arizona. The system was bolted on a DC-3 aircraft, and it took its first flight in 1953. The first civilian spaceborne SAR-satellite, NASA SEASAT, was launched in 1978. [4][2]

The block diagram of a typical pulse radar system is illustrated in Figure 1. Generally, pulse radar systems consist of antenna, transmitter, receiver, data recorder and display device. The transmitter generates high power radio wave pulse, which is directed to the antenna. The antenna transmits the pulse towards an object to be observed. Some portion of the transmitted pulse is then backscattered from the observed object back to the antenna. This received pulse is directed from the antenna to the receiver, which converts it into a digital number. Received data is stored by the data recorder for further analysis, processing and display. By measuring the time difference between the transmitted pulse and the received echo, the distance between the radar and the illuminated object can be determined. The purpose of the switch in the system is to allow the use of a single antenna; it disconnects the receiver from the antenna during the pulse transmission. Otherwise, if the high power pulse were allowed to enter the receiver, the receiver could be damaged. [4]

In Synthetic Aperture Radar (SAR) systems, a pulse radar is attached to moving platform like an aircraft or a satellite to provide remote sensing images from the Earth. A typical SAR system consists of conventional pulse radar building blocks such as antenna, transmitter, receiver, data collection system and signal processor. The result from

SAR imaging is a two dimensional reflectivity image, where bright spots are identified as targets with high reflectivity (in the sense of radio waves), and dark spots as targets with low reflectivity. An example of remotely sensed SAR-image is shown in Figure 2. Major benefit of SAR imaging compared to optical sensors is its independence of weather and lighting conditions. Also, the resolution of SAR systems today is sufficient for high-precision applications and is comparable with optical imaging systems. For instance, TerraSAR-X satellite operating in spotlight mode can provide maximum resolution of up to 1 meter. [5]

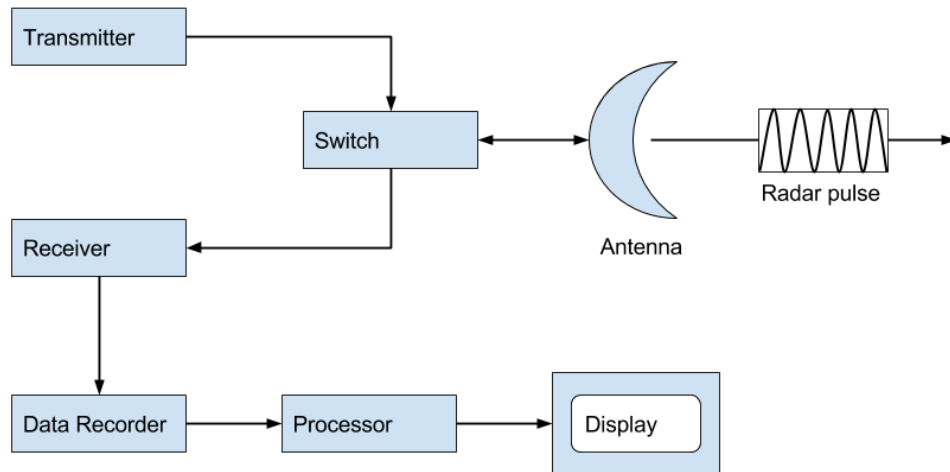


Figure 1 Block diagram of pulse radar system



Figure 2 First TerraSAR-X image: Tsimlyanskoye Impounding Reservoir in Russia [6]

In SAR, forward motion of an aircraft or a satellite is combined with signal processing in order to “synthesize” longer antennas than would be physically feasible to implement. This enables SAR to achieve good azimuth resolution with reasonable antenna lengths

independent of the altitude of the platform. This is based on the fact, that when the altitude of the platform is increased, the beam width on the ground is also increased, which further increases the illumination time T . These two factors balance each other resulting constant azimuth resolution independent of altitude. This property of SAR makes the technology feasible in high-altitude platforms like satellites. Practically, achievable azimuth resolution in SAR systems is approximately $\frac{1}{2}$ of the actual (real) antenna length. [4][5][7]

The principle of SAR is illustrated in Figure 3 Illustration of the SAR principleFigure 3: when the platform (satellite) is moving along its flight path, the SAR is simultaneously transmitting successive electromagnetic pulses to the ground. When the transmitted pulse reaches the ground, some portion of pulse energy is scattered back to the receiver. Magnitudes and phases of these backscattered pulses are stored into system memory and the SAR-image is computed based on this information. [4][5]

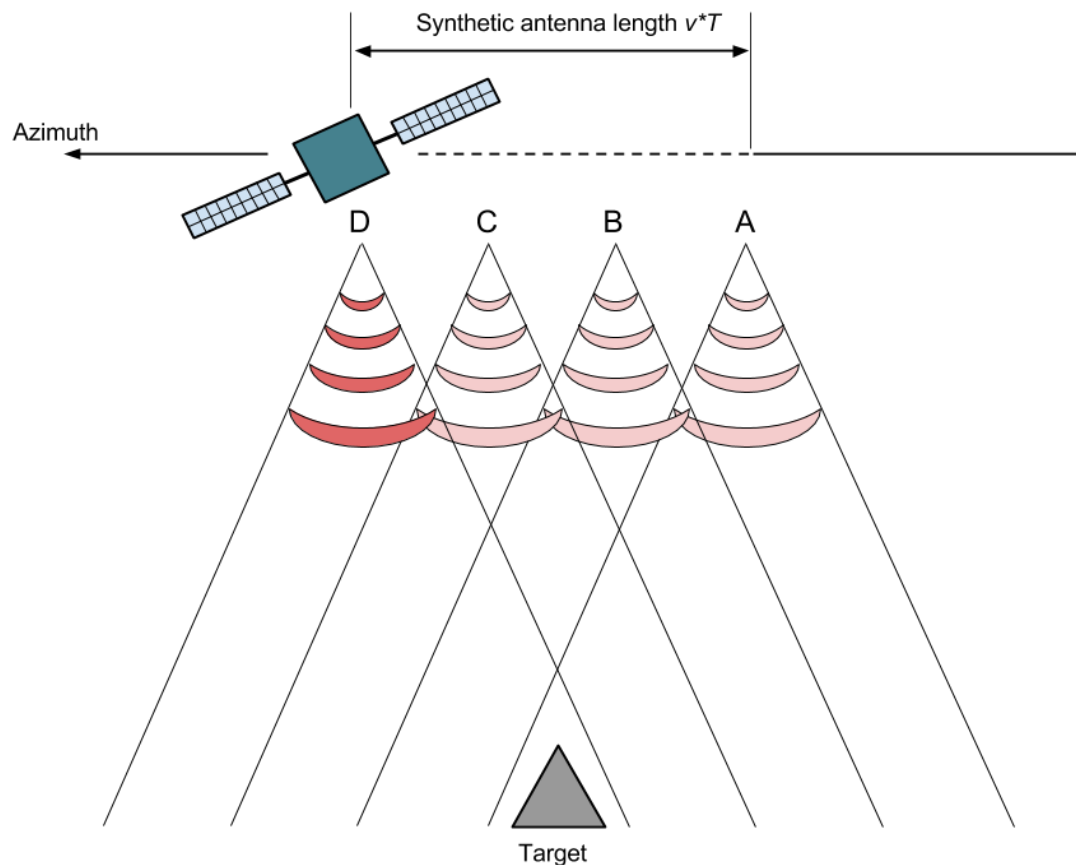


Figure 3 Illustration of the SAR principle

In Figure 3, during the time period T , amplitude and phase information of received pulses are stored in positions A-D. This information is further used with the aid of signal processing to form an image of the target. The time period T is called illumination time, i.e., the time during which the target is within the radar beam. The length of the synthetic antenna is $v \cdot T$, where v is the velocity of the platform. [4][5]

2.1.1 Imaging Geometry

The imaging geometry of SAR is illustrated in Figure 4. The platform is moving at height H with velocity v_{SAR} . The antenna beam is aimed to the ground with an angle of incidence θ_0 . The antenna of SAR is arranged perpendicular to the flight direction, which is referred as azimuth direction. The imaging direction is called as slant range direction (R_0). The area illuminated on the ground by the antenna beam is called the antenna footprint. During the forward movement of the platform, the antenna footprint is swiping a strip on the ground, called radar swath. In airborne SAR systems, the radar swath width is typically from a few kilometers to 20 km. In spaceborne systems, the swath width ranges from 30 km to 500 km. [8]

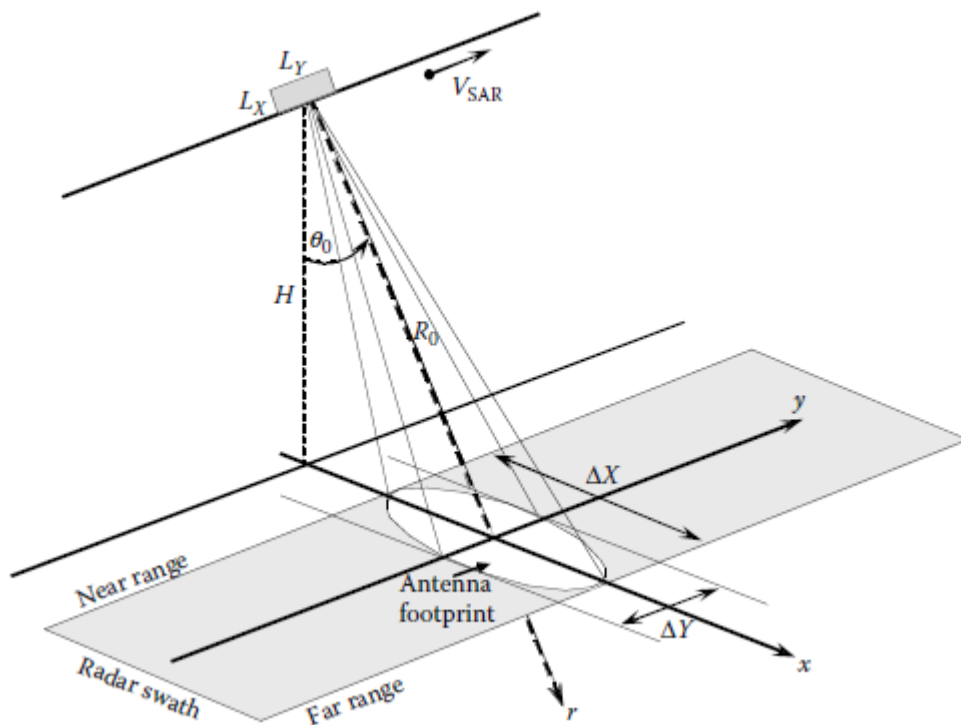


Figure 4 SAR imaging geometry [8]

An important aspect having great influence on SAR imaging quality is the spatial resolution of the system. The spatial resolution defines the ability of the SAR system to distinguish two closely positioned scatterers on the ground. Slant-range resolution of the SAR system is denoted as $\delta_r = \frac{c}{2B}$, which is inversely proportional to the system bandwidth B (c is speed of light). Azimuth resolution, denoted as δ_a , is half of the actual (real) antenna length according to $\delta_a = \frac{d_a}{2}$, where d_a is the antenna length. The equation for azimuth resolution implies that the shorter the antenna, the better the azimuth resolution. This counterintuitive result is based on the fact that radar beam width on the ground increases when the length of the real antenna decreases. This further means

longer illumination time T and longer synthetic antenna length, which increases the azimuth resolution. [5]

2.1.2 Pulse Transmission and Receiving

SAR-systems are operating in the microwave region of the electromagnetic spectrum. This region is commonly divided into 7 bands listed in the Table 1 Commonly used frequency bands for SAR systems [5]Table 1.

Table 1 Commonly used frequency bands for SAR systems [5]

Frequency Band	Ka	Ku	X	C	S	L	P
Frequency [GHz]	40-25	17.6-12	12-7.5	7.5-3.75	3.75-2	2-1	0.5-0.25
Wavelength [cm]	0.75-1.2	1.7-2.5	2.5-4	4-8	8-15	15-30	60-120

For example, different bands can be used in the following applications:

- **P- and L-band** - foliage penetration, subsurface imaging, biomass estimation
- **L-, C-, S- and X-band** - agriculture, ocean, ice or subsidence monitoring
- **X- and Ku-band** - snow monitoring
- **X- and Ka-band** - very high-resolution imaging.

The most frequently used bands in common applications are the L-, C- and X-bands. As a rule of thumb, the longer the wavelength, the better the penetration of the radar pulse through the atmosphere. [5]

In radar systems, short pulse time is preferable, because it improves the slant range resolution. Unfortunately, short pulse time also decreases the average power, which has negative impact to signal detectability and measurement precision. These two conflicting requirements cannot be fulfilled at the same time in radar systems employing standard sinusoidal pulse form; improvement in resolution always decreases the average power. Due to that, so-called *chirp* pulse is used; it offers signal detectability of long-pulse radar system as well as good slant range resolution. The chirp signal is a frequency modulated constant amplitude pulse and its instantaneous frequency is usually varied linearly according to $f_i = kt$, where k is the chirp rate. Bandwidth B of a chirp pulse is denoted as $B = k\tau$. Illustration of the chirp pulse is shown in Figure 5Virhe. **Vitteen lähde ei löytynyt.** Chirp pulses are transmitted from radar in constant time intervals. Pulse transmission is followed by the *echo window* time, when the radar listens backscattered echoes from the transmitted pulse. This interval is called Pulse Repetition Interval (PRI) and it is measured in seconds. The reciprocal of PRI is the Pulse Repetition Frequency (PRF) measured in Hz. [4][5]

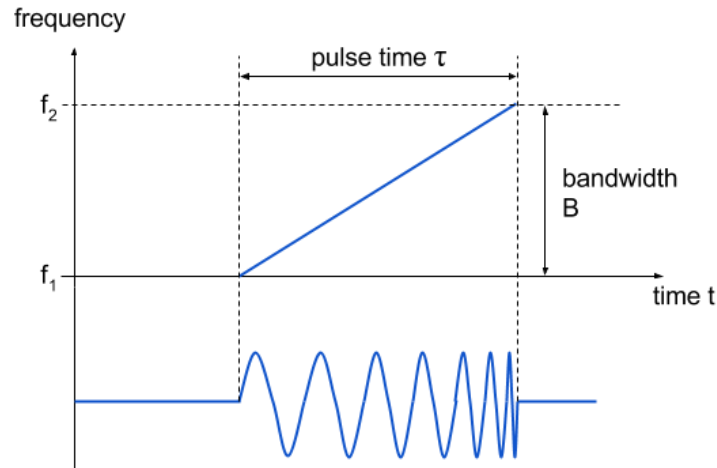


Figure 5 Chirp pulse

The amplitude and phase of the backscattered pulses are highly dependent on the geometrical and electrical properties of an imaged object, as well as SAR system parameters such as frequency, polarization and incidence angle of emitted electromagnetic waves. For instance, calm sea surface appears dark (low intensity) in SAR images since most of the pulse energy is reflected away. On the other hand, many manmade objects, like high rising buildings, tend to appear as very bright spots in the SAR image due to the double-bounce reflection effect. [4][5][8]

2.1.3 SAR Signal Processing

Signal processor is the heart of a SAR system; the synthetic antenna is formed by using digital signal processing techniques. The received radar echoes are stored in a 2-dimensional complex valued data matrix, where real parts represent magnitude and imaginary parts phase information. Columns of the matrix correspond to range direction and rows to azimuth direction. The raw SAR-data in the matrix are transformed into human interpretable form with signal processing. In Figure 6, a raw unprocessed SAR-data matrix is illustrated; the information from an imaged target is spread out both in range and azimuth directions. [2][5]

Two main approaches exist for SAR processing: the Two-dimension Algorithm and the Range Doppler Processing Algorithm. The difference between the two is that the Two-dimensional Algorithm processes range and azimuth data simultaneously while the Range Doppler Algorithm handles them separately. The Range Doppler Algorithm is more commonly used since it is computationally more efficient due to utilization of the Fast Fourier Transform (FFT). [5]

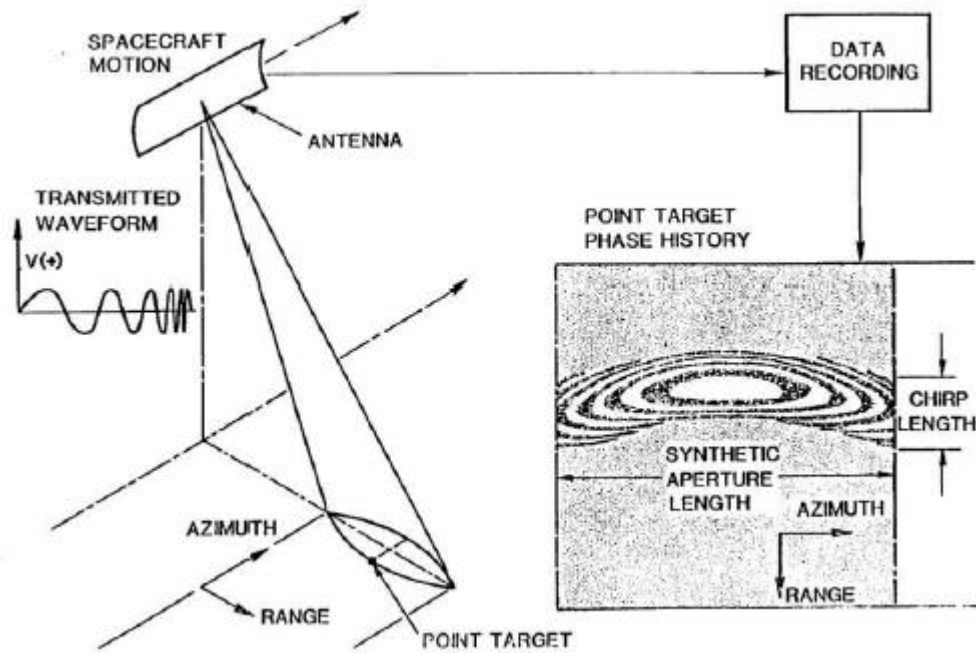


Figure 6 SAR point target echo [2]

SAR signal processing with the Range Doppler Algorithm consists of two matched filter operations: range compression and azimuth compression (Figure 7 **Virhe. Viitteen lähdeä ei löytynyt.**). The first step, range compression, transforms transmitted chirp signal into a short pulse. This results in range compressed image, which reveals only the information about relative range distances between the radar and the ground objects. The second step, the azimuth compression, compresses the signals in the azimuth direction resulting in the final SAR image. All these operations are computed as multiplications in the frequency domain instead of convolutions in the time domain to decrease computational load of the algorithm (utilization of FFT). The effect of range and azimuth compression on raw SAR data is shown in Figure 8. [4][5]

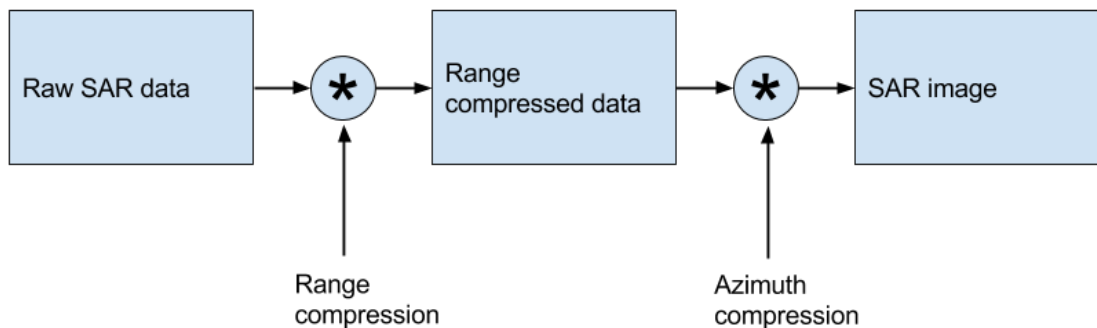


Figure 7 SAR processing steps with the Range Doppler Algorithm

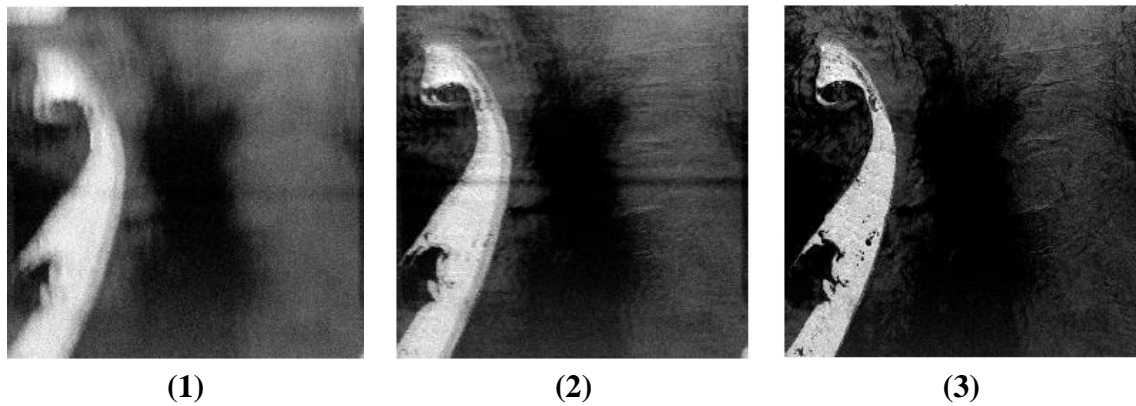


Figure 8 SAR images taken on 15th of April 1994 from the tip of Cape Cod. Image (1) illustrates raw SAR data where pulse energy is spread out both in range and azimuth directions, (2) presents the image after the range compression, (3) presents fully processed image (after both range and azimuth compression) [2]

2.1.4 Imaging Modes

Modern SAR systems are capable of producing remote sensing imagery in different imaging modes by steering the radar beam. Imaging mode selection is a tradeoff between the resolution and the size of the imaged area; the better the resolution, the smaller is the imaged area. Three commonly used imaging modes, Stripmap, ScanSAR and Spotlight, are illustrated in Figure 9. [5]

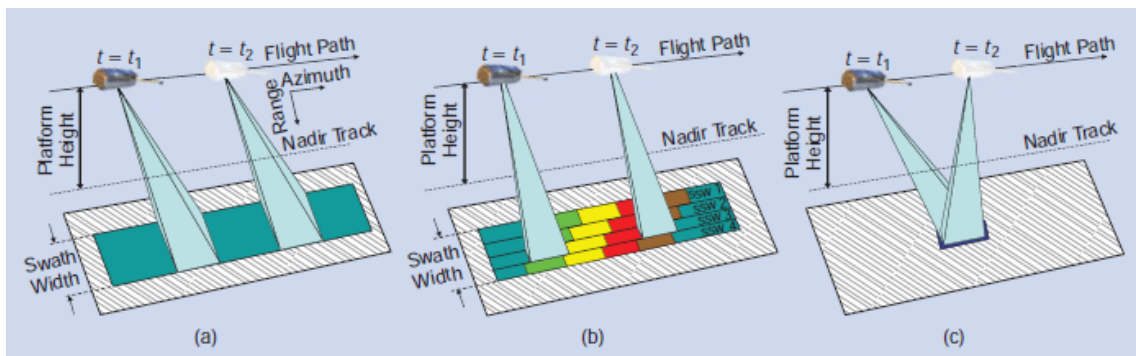


Figure 9 SAR imaging modes. (a) Stripmap, (b) ScanSAR, (c) Spotlight [5]

The basic imaging mode is the Stripmap mode shown in Figure 9 (a). In the Stripmap mode, the radar beam is scanning a single continuous strip along the flight path of the platform. Figure 9 (b) illustrates the ScanSAR mode which can be utilized if larger imaged area is required. In the ScanSAR mode, the radar beam is steered in order to produce multiple sub-swaths resulting in larger total area than in the Stripmap mode. Wider swath in the ScanSAR mode is achieved at the expense of resolution. If higher precision is required, the Spotlight mode shown in Figure 9 (c) offers the best resolution among all the three imaging modes. In the Spotlight mode, the radar beam is steered to a single fixed point to achieve long illumination time, resulting in increased synthetic aperture

length and higher resolution. The disadvantage of the Spotlight mode is smaller imaged area since only a single spot is imaged instead of continuous swath. [5]

2.1.5 Geometrical Distortions and Speckle Noise

Geometrical distortions are distortions in SAR images induced by the side-looking imaging geometry of SAR. These distortions occur because SAR is measuring distance between the radar and ground objects in slant range direction instead of true horizontal direction (ground range direction). Three specific sources of distortions are called *foreshortening*, *layover* and *radar shadow*. These distortions are illustrated in Figure 10, Figure 11 and Figure 12, respectively. [8]

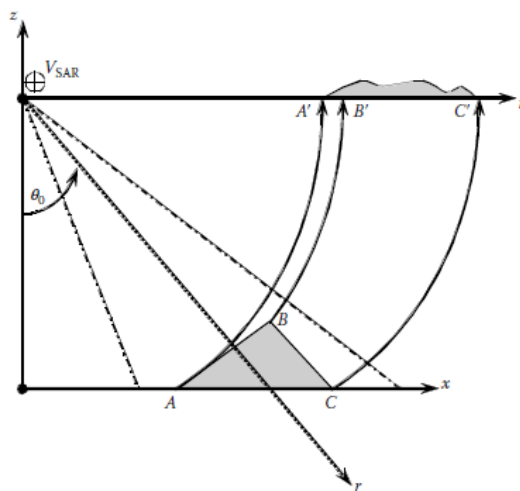


Figure 10 Foreshortening distortion [8]

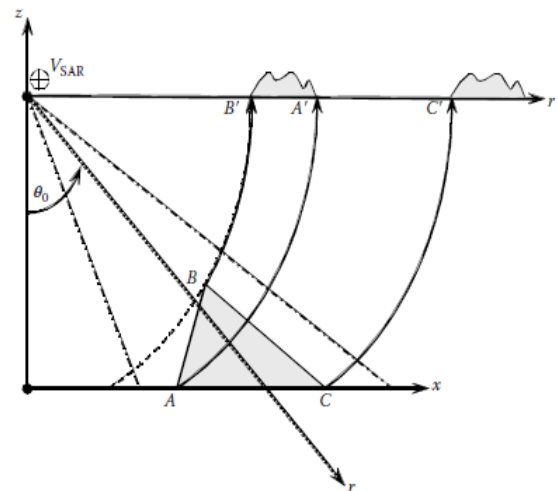


Figure 11 Layover distortion [8]

The foreshortening distortion is illustrated in Figure 10. Foreshortening is a dominant distortion in mountainous areas and it occurs especially with steep-looking spaceborne SAR sensors. In foreshortening distortion, the distance between two points located on a fore slope of a mountain appears to be smaller than it actually is. In Figure 10, all three points A , B and C are equally spaced in x -direction, but in SAR image, A' and B' appear to be considerably closer to each other than B' and C' . [8]

The layover distortion is illustrated in Figure 11. In the layover distortion, the ordering of objects in SAR image is reversed compared to true ordering of objects on the ground. The influence of this phenomenon can be seen from Figure 11, where the horizontal ordering of A and B is reversed in the SAR image ($AB \rightarrow B'A'$). This occurs in the case of very steep slope, because the top of the mountain is closer to the SAR sensor than the objects on the ground. [8]

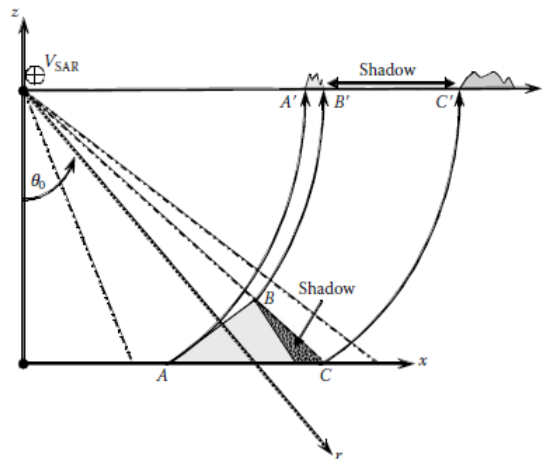


Figure 12 Radar shadow [8]

The radar shadow distortion is illustrated in Figure 12. In the radar shadow distortion, some parts of the Earth's surface are "shadowed"; these regions appear as zero signal (system noise is present) in SAR image. The radar shadow distortion occurs because radar cannot "see" behind the slopes that are in steeper angle than the SAR sensor depression angle θ_0 . In Figure 12, the segment between B and C appears in SAR image as shadow between points B' and C' . [8]

Another source of errors in SAR-images is the *speckle noise* illustrated in Figure 13. In fact, speckle is not a noise; it is a physical measurement of the resolution cell structure. Speckle cannot be reduced by increasing the transmit power since its variance increases with intensity. Speckle arises when there exist multiple randomly distributed scatterers within a single resolution cell (Figure 13). The signal scattered back to the antenna is a coherent sum of these multiple scatterings. Speckle phenomenon results strong fluctuations in amplitudes and phases from one resolution cell to another appearing as grainy texture in SAR images. Because of speckle, amplitudes and phases in SAR images are not deterministic; they follow exponential and uniform distribution, respectively. [5]

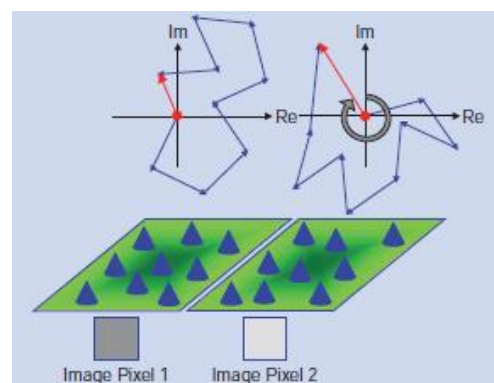


Figure 13 Speckle. The two squares represent resolution cells. In each resolution cell there exists random distribution of scatterers, which results variation in amplitudes and phases from pixel to pixel. [5]

2.1.6 SAR Data Preprocessing and Correction

SAR images obtained from different imaging directions are not comparable without radiometric calibration; pixel intensity values differ between images taken with different imaging angles. In order to make the images compatible with each other, radiometric calibration is conducted.

TerraSAR-X data are delivered in binary format. The data are stored in a folder structure, which includes separate files for metadata (xml) and the radar data. The radar data are in complex form consisting of real and imaginary parts. Processing of the radar data is implemented with Sentinel-1 toolbox, which is a dedicated program for TerraSAR-X data calibration. Sentinel-1 toolbox accepts complex data only in polar form, thus polar form conversion must be made prior to importing the data into Sentinel-1.

Terrain correction is calculated in Sentinel-1 according to the following equation:

$$\sigma^0 = \beta_0 \sin \theta_{loc}$$

where θ_{loc} is the angle between the radar pulse and the ground (terrain model required), and

$$\beta_0 = k_s |DN|^2$$

where DN is the digital number (magnitude of a complex number), and k_s is the calibration coefficient given by the satellite operator. Intensity of the radar echo is denoted by σ^0 . Calculated radar echo intensities, σ^0 , are exported into geotiff (.tif) file format. Interpolation of the data is handled in Sentinel-1 concurrently during exporting. Different interpolation methods, like Nearest Neighbour (NN), for example, are available.

[9]

2.2 TerraSAR-X Satellite Program

TerraSAR-X is a German satellite program implemented in public-private cooperation between German Aerospace Center (DLR) and EADS Astrium GmbH. The objective of the satellite program is to supply high-quality remote sensing radar data both for public and private sector institutions. The DLR is responsible for operation and development of the satellite system as well as receiving, archiving and processing of the radar data. On the other hand, EADS Astrium is responsible for marketing and commercial utilization of the satellite data and products. In exchange for commercial utilization rights, EADS Astrium is taking part in operation costs and has funded the development of the satellite.

TerraSAR-X satellite was launched from the Russian spaceport Baikonur on June 15th, 2007. It was fully operational at the beginning of 2008. Since that, TerraSAR-X has inspected the surface of the Earth from its 514 km height orbit and supplied information for both scientific and commercial purposes. The orbit is selected in such a way, that the satellite is moving along the day-night boundary of the Earth, and thus always keeps the same side towards the sun. In this way optimal power supply is guaranteed for the satellite. All the regions of the Earth are imaged during the 11 day cycle. Using different imaging angles, certain points on the Earth can be targeted within 2 to 4 days.

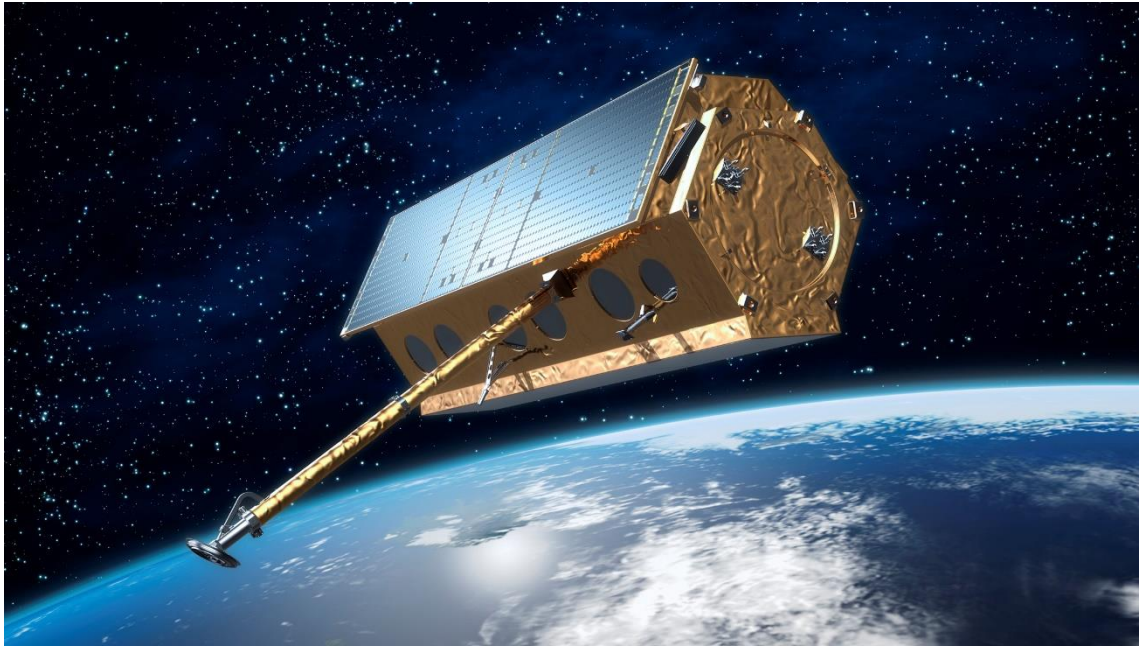


Figure 14 TerraSAR-X satellite [10]

TerraSAR-X is 5 m long and 2.4 m wide, and it weights approximately 1.3 tons. On one side of its hexagonal shaped frame (Figure 14 TerraSAR-X satelliteFigure 14), it is carrying 5 m long and 80 cm wide radar antenna. Energy required for the satellite's operation is generated with gallium arsenide solar cells, which are fitted on another side of the satellite's frame in the same manner as the radar antenna. Position control of the satellite is handled with star sensors and a GPS receiver. Technical specifications of TerraSAR-X can be seen from Table 2.

The primary payload of the satellite is the Synthetic Aperture Radar (SAR), which is operating in the X-band at the frequency of 9.65 GHz (wavelength of ~3 cm). The radar antenna consists of 384 transmit/receive modules, and it can operate in H (horizontal) and V (vertical) polarizations. The radar beam can be controlled by 0.75° in the flight direction and by 20° in the direction perpendicular to the flight direction. The data gathered are stored in 256 Gbit mass storage system, and it is transmitted to the ground station via 300 Mbit/s downlink. Three different imaging modes are available:

- **Spotlight Mode:** 5-10 by 10 kilometres area is recorded with a maximum resolution of up to 1 meter.
- **Stripmap Mode:** 30 kilometres wide and maximum 1500 kilometres long strip is recorded with a resolution of 3 meters.
- **ScanSAR mode:** 100 kilometres wide and maximum 1500 kilometres long strip is recorded with a resolution of 18 metres.

Table 2 Technical specifications of TerraSAR-X

TerraSAR-X	
Launch	June 15th, 2007
Site	Baikonur, Kazakhstan
Launch vehicle	DNEPR-1
Orbit height	514 kilometers
Inclination	97.44 degree
Satellite mass	approx. 1.230 kilogram
Satellite size	5 meters height by 2.4 meters diameter
Radar frequency	9.65 GHz
Energy consumption	800 watt (Average)
Mission operation	German Space Operations Center, Oberpfaffenhofen
Satellite command	DLR Ground Station, Weilheim
Data reception	DLR Ground Station, Neustrelitz

In addition to the Synthetic Aperture Radar, the primary payload, two experimental devices are also mounted on the satellite: the Laser Communication Terminal (LCT) and the IGOR dual frequency GPS receiver. LCT is a device used to experiment rapid optical data transfer in space between the two satellites TerraSAR-X and NFIRE. IGOR is a device used for determination of satellite's orbit with an accuracy of a few centimetres.[6]

3. PATTERN RECOGNITION AND CLASSIFICATION METHODS IN REMOTE SENSING DATA ANALYSIS

Pattern recognition (PR) is a branch of machine learning (ML). Roots of machine learning are in the field of artificial intelligence (AI), which began to develop in 1950s, when the first digital computers were invented. Increased computational power due to the invention of digital computers made it possible for scientists to start exploring computationally demanding AI algorithms, which were earlier out of human labor scope. The first AI algorithms employed symbolic methods or simple linear models like perceptrons. In 1980s, ML and PR began to drift apart from the field of AI, because researchers in the AI started to focus more on logical approaches, whereas the field of ML and PR moved towards solving practical problems with statistical and probabilistic methods. [11][12][13]

The goal of pattern recognition is to discover regularities in the data in order to provide useful information about the phenomenon the data is acquired from. PR techniques have been utilized in a very diverse range of applications: computer aided medical diagnosis, speech recognition, spam filters and optical character recognition, just to name a few. Although PR, ML and AI are considered nowadays separate fields, clear discrimination between them is difficult in many cases, since many overlapping methods are still used. In general, PR and ML are focusing more on solving specific practical problems, like image classification and face recognition, whereas the aim of AI is to develop “ideally intelligent” machines, like self-driving cars or chess playing algorithms. [11][12][13]

The aim in pattern recognition is to fit a mathematical model to the data at hand in such a way that some performance measure, such as correct classification rate, for example, is maximized. A PR model can be described as a function $\mathbf{y}(\mathbf{x})$, which takes \mathbf{x} as an input vector and maps it to an output vector \mathbf{y} . When establishing a pattern recognition system, usually the first step is to select a suitable model for the given task. The next step is to find model parameters having the best fit to the data, i.e., to find a parameter set, which enables the system to make accurate predictions for new unseen data. It is said that the system is capable to *generalize*, when it is able to make precise predictions for unseen data. [14]

Mitchell describes the machine learning property of a computer program in the following way: “A computer program is said to learn from experience E with respect to some

class of tasks T and performance measure P , if its performance at tasks in T , as measured by P , improves with experience E .” [15]

As an example, the task T could be to classify oranges and bananas according to their color, in which case the experience E could be the labelled data set of fruits and their colors $\langle \text{fruit}, \text{color} \rangle$, and performance P could be the percentage of correctly classified fruits. In PR, this labelled data set E is called the training data set. The training data set is used to find model parameters giving the best performance with respect to P , i.e., to enable the system to generalize. The ability to generalize is essential for PR systems, since only minor fraction of all possible input data vectors can be included into the training data set. [14]

3.1 Models

Models play the central role in machine learning and pattern recognition applications since model selection has very significant impact on how well the system is able to perform. The model parameters are usually defined using some numerical and heuristic methods, because mathematically exact solutions are not available for most of the problems. This means that it is not guaranteed that the best parameters (global optimum) are ever found for a given problem. Fortunately, this is more theoretical than a practical issue, and machine learning techniques are still very feasible in practical problems. [16]

Large number of different machine learning algorithms exist and they can be grouped in many different ways. One approach is to divide them into three categories: *geometric models*, *probabilistic models*, and *logical models*. [16]

Geometric models are constructed using geometrical concepts such as lines, planes and distances. One great advantage of geometric models is that they are easy to visualize and interpret. Of course, the interpretation of geometric models is strongly dependent on the dimensionality of the data; two or three dimensions are easy to understand for human brains, but higher dimensions are causing great difficulties to our imagination. However, despite our limited capacity to understand high-dimensional spaces, they occur often in machine learning applications. Some examples of geometric models are the nearest-neighbor classifier, linear models and the K-means clustering algorithm. [16]

Probabilistic models are based on theories derived from the field of probability and statistics. Statistical pattern recognition methods are assuming that there exists some random process, which is generating the data according to some unknown probability distribution. The goal is to find out more information about this unknown distribution using probabilistic models and the data. Gaussian mixture models, the naïve Bayesian classifier and Bayesian networks are examples of probabilistic models. [16]

Logical models are employing rules and logical deduction to make predictions. They are more algorithmic in nature than probabilistic and geometric models, deriving a lot of inspiration from computer sciences and engineering. The advantage of logical models is that the rules they are using are easy to understand. For instance, the rules can be organized into a tree-like structure, which is easy to interpret for humans. Some very popular logical models are decision trees and random forests. [16]

For all machine learning models there exist two universal characteristics: bias and variance. The bias of a model defines how expressive and flexible it is. The variance defines how sensitive the model is for small fluctuations in the training data. For example, high-bias and low-variance models, such as linear classifiers, are robust to random variations in the training data, but they suffer from lack of expressivity. On the other hand, low-bias and high-variance models, such as neural networks, for example, are capable to model very complex problems, but they suffer from high variance; small variations in the training data can result in totally different models. Therefore, model selection is always a tradeoff between bias and variance. This phenomenon is called *bias-variance dilemma*. [16]

3.2 Features

Features are important aspect in machine learning applications. They mainly define how successful a certain machine learning application can be, since classification performance of a model is strongly related to features. A good feature set contains information that makes it easy to discriminate between classes. More precisely, features are mappings $f_i = X \rightarrow \mathcal{F}_i$ from input space X to the feature domain \mathcal{F}_i . Common feature domains are real or integer numbers, but feature domains like Booleans or colors can also be used. Features can be manipulated in many different ways; domain of a feature can be changed via discretization, optimization can be utilized to select a good feature subset or transformations such as principal component analysis can be applied. [16]

Features are divided into three groups based on their domain:

- *Quantitative features* (continuous) are mappings into real numbers.
- *Ordinal features* are mappings into some totally ordered set, such as characters or strings. Although ordering is included in this feature type, it does not need to have numerical scale.
- *Categorical features* are mappings into a set, which does not have either ordering or scale. For example, Boolean features are categorical.

Different models are able to handle different feature types. For instance, decision trees can deal with all types of features, Naïve Bayes classifier is able to handle only categorical features (ordinal features are treated as categorical ones), and many geometrical

models, like the K-means algorithm with Euclidean metric, can only handle quantitative features. [16]

Sometimes it is necessary to reduce the number of features because some PR models are sensitive to high number of features (high dimensional data). Feature reduction serves several purposes: it avoids overfitting and it may increase interpretability of the data in some cases. The need for feature reduction is dependent on the classification algorithm used and the number of features. For example, some classification algorithms, like decision trees, have built-in functionality for feature selection, and therefore no external feature reduction methods are required. Feature reduction can be implemented either by selecting a good subset of features from the original feature set or by applying some transformation like Principal Component Analysis (PCA) to the original feature space. In both cases, the aim is to represent the data in reduced number of dimensions. [13]

3.3 Supervised and Unsupervised Learning

Depending on the type of the training data, two learning paradigms exist in the field of machine learning: *supervised learning* and *unsupervised learning*. In supervised learning, all training patterns are labelled; each training pattern has a desired output value attached to it. If the output values are discrete (classes), the learning task is called *classification*. If the output values are continuous, the task is called *regression*. On the other hand, in unsupervised learning the training data consists of patterns without desired output values. The goal of unsupervised learning is to find some underlying structures within the data set. *Clustering* is an unsupervised learning method, the objective of which is to form clusters of similar input patterns. *Density estimation* is another unsupervised method which is used to determine the distribution of the data. Unsupervised learning techniques can also be utilized for visualization purposes to project high-dimensional data down to two or three dimensions (like in the case of Self Organizing Maps, for example). [14]

3.4 Overfitting and the Curse of Dimensionality

Overfitting and *the curse of dimensionality* are two closely related issues in machine learning. Both of them are dealing with the chronic problem of the lack of training data. Overfitting occurs when the model has too many parameters to be adjusted and too few training data. The curse of dimensionality deals with the fact that high-dimensional spaces tend to be very sparse, since exponentially increasing amount of training data is required to fill the space when the dimensionality increases. So, when the dimensionality of the feature space is high, lack of training data is present, and overfitting might occur.

Overfitting is a phenomenon, which arises when the classifier adapts to noise properties of the training data. Overfitting leads to poor generalization performance, and thus low

prediction accuracy for unseen samples. Usually overfitting occurs if the classifier is too complex (too many parameters to adjust) compared to the amount of training data available. The opposite behavior is also possible; underfitting occurs when the classifier is too rigid. In this case the classifier cannot model the structure in the data. Overfitting and underfitting are illustrated in Figure 15 and 16, respectively. Figure 15 [13]

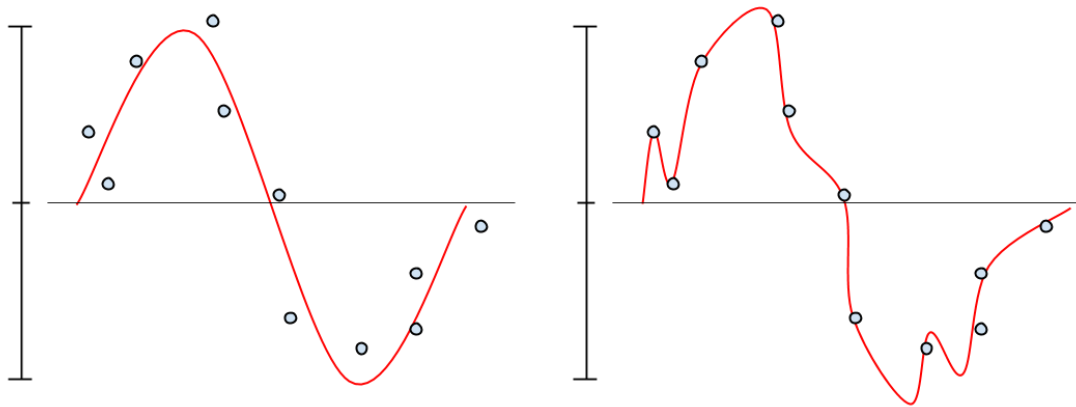


Figure 15 Overfitting. Model on the left hand side has a good fit to the data contrary to the model on the right, which has adapted to the noise in the data.

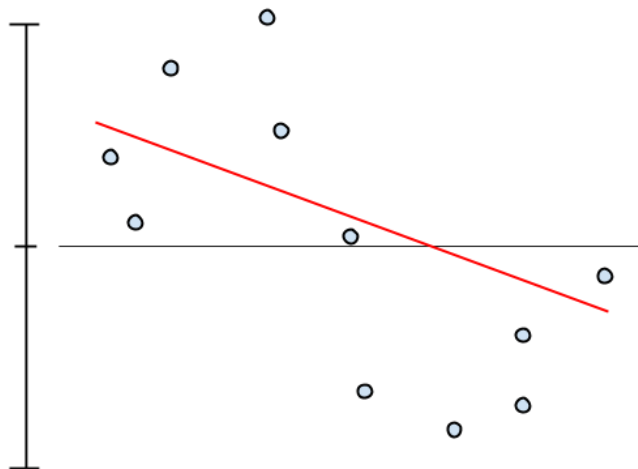


Figure 16 Underfitting. The model cannot capture the structure in the data since it is too rigid.

The number of available training patterns has strong influence on the classifier's tendency to overfit. The more training data are available, the more complex classifiers can be used without the fear of overfitting. Figure 17 shows how the number of available training patterns affects the classifiers tendency to overfit. [14]

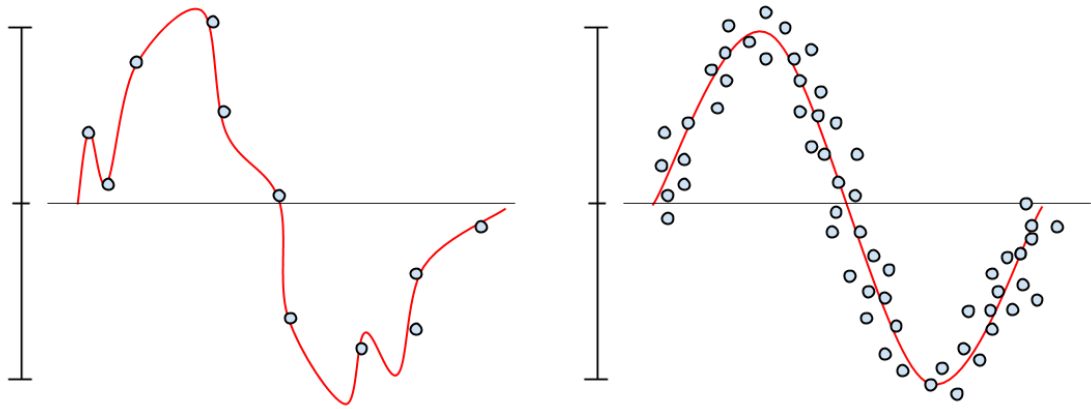


Figure 17 Overfitting. Models in both pictures share the same complexity, but the number of training data points differs. On the left hand side, the model is overfitting due to the lack of training data. On the right hand side, enough training data is available and the model has good and smooth fit to the data.

The *curse of dimensionality* is a phenomenon occurring in high dimensional spaces, i.e., when the dimensionality of the feature space is high. The origin of the problem is that the number of required training data points grows exponentially with the dimensionality of the space. The curse of dimensionality is illustrated in Figure 18; when $D=1$, only three “cubes” are required to cover the space; when $D=3$, already 27 cubes are required. This exponential growth continues as more dimensions are added. [14]

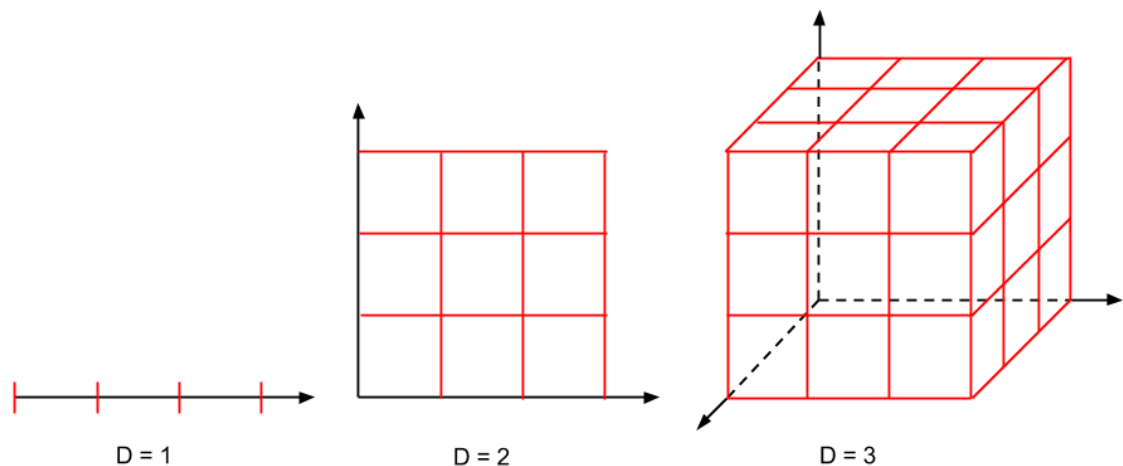


Figure 18 The curse of dimensionality

High-dimensional feature spaces are common in practice, and therefore the curse of dimensionality has considerable negative influence in machine learning applications. Many problems arise due to the curse of dimensionality: when the space becomes sparse, overfitting is more likely to occur; distance measures lose their meaning in high dimensional spaces, because all points are equally far away from others. Also, our everyday intuitions from the three-dimensional space are not always generalized to high-

dimensional spaces and thus some counter-intuitive phenomena may occur. For example, in high dimensional spaces, most of the volume of a sphere is concentrated in a thin shell near the surface. The same phenomenon applies to high-dimensional Gaussian distributions; most of the probability mass of the distribution is concentrated in a thin shell. [14]

3.5 Evaluation of Classification Results

Evaluation of the classification results is an important step in the design process of a pattern recognition system. Evaluation of results gives us information about the classifier's performance and enables to analyze and compare the performance of different classification algorithms. [13]

Holdout method is the simplest approach for the evaluation of classifier performance. In the holdout method, the data are divided randomly into two separate (not overlapping) sets: training and test sets. The training set is used only for training the classifier whereas the test set is used only for evaluation of the classifier performance. The test set is kept totally separate from the training set, because over-optimistic results are obtained if the performance is evaluated over the same set as used in the training. The holdout method is computationally efficient and simple to implement, but it gives pessimistically biased error estimates. [13]

Cross-validation is another method for performance evaluation. In cross-validation, the classifier is trained with a training set containing $n - 1$ samples and the performance is evaluated using the remaining sample. This maneuver is repeated n times until all samples are tested. Cross-validation is computationally inefficient for large data sets, since n classifiers should be trained in order to calculate the classification performance.[13]

V-fold cross-validation is a combination of these two methods. In this approach, the data set is divided into v subsets. The classifier is trained using $v - 1$ subsets and tested using the remaining subset. This procedure is repeated until all subsets are tested. V-fold cross-validation is a compromise between the holdout method and the original cross-validation; it gives reduced bias compared to the holdout method and it is less computationally demanding than cross-validation. [13]

Confusion matrix is a tool for numerical assessment of classifier performance. Confusion matrix is a square matrix calculated for the classified test set. It shows relationships between the actual and predicted classes. Diagonal entries of the confusion matrix represent correctly classified samples. Each column in the confusion matrix represents samples belonging to the actual class and each row represents samples in a predicted class. Example of a confusion matrix can be seen from Table 3. [17][13]

Table 3 Example of confusion matrix for the case of three classes

Predicted classes	Actual classes		
	Cat	Dog	Fish
Cat	5	3	0
Dog	2	3	1
Fish	0	2	11

Different performance metrics shown in Table 4 can be derived from the confusion matrix. The following abbreviations are used:

- TP , true positive hits
- TN , true negative hits
- FP , false positive hits
- FN , false negative hits

Table 4 Confusion matrix performance metrics. $P = TP + FN$, $N = FP + TN$

Accuracy (Acc)	$\frac{TP + TN}{P + N}$
Error rate (E)	$1 - Acc$
False positive rate (fpr)	$\frac{FP}{N}$
True positive rate (tpr)	$\frac{TP}{P}$
Precision ($Prec$)	$\frac{TP}{TP + FP}$
Sensitivity ($Sens$, identical to tpr)	$\frac{TP}{P}$
Specificity ($Spec$)	$\frac{TN}{N}$

3.6 Ensemble Methods

Ensemble methods are techniques to improve classification performance by combining the outputs of multiple classifiers, called base classifiers, instead of using a single classifier. Combinations of many base classifiers are called as *model ensembles*. Ensemble methods are very effective techniques, and they often outperform other methods. The

cost for higher performance is increased algorithmic and model complexity, as well as increased computation times. [16]

Theoretical foundations of ensemble methods are based on computational learning theory and statistics. A widely known phenomenon in statistics is that averaging measurements leads to more robust and reliable estimates, because the influence of random fluctuations in single measurements is reduced. This holds also in the case of model ensembles. If the ensemble of slightly different base classifiers is trained based on the same training data, it is possible to reduce random fluctuations in individual base classifiers in a similar way. An important requirement for model ensembles is that some degree of diversity should be present among the base classifiers. Otherwise, if the models were too similar, combining the classification results of multiple base classifiers would not offer any improvements over a single classifier. [16]

There are numerous ways to construct a model ensemble. For example, base classifiers could be arranged in either parallel or serial fashion, learning algorithms for individual base classifiers could be common or dissimilar, feature spaces of base classifiers could be the same or different etc. However, all classifiers employing ensemble methods share two characteristics:

- multiple base classifiers are constructed based on training data
- predictions of these classifiers are combined in some way.

Often predictions of the base classifiers are combined simply by averaging or majority voting, but more complex combining techniques, like tree-based approaches, for example, exist as well. [13] [16]

Two common ensemble methods are *bagging* and *boosting*. Boosting is a useful technique with robust and high-biased classifiers (like linear classifiers), and it is primarily a bias-reduction technique. Bagging, on the other hand, is mainly a variance-reduction technique, and it is commonly used with instable, high-variance and low-biased classifiers, like decision trees or neural networks.[16]

3.6.1 Bagging

Bagging (bootstrap aggregating) is an ensemble technique that creates multiple base classifiers by picking random samples from the original training data set. These random samples are called *bootstrap samples*. This procedure creates multiple overlapping replicates (bootstrap samples) from the training data and base classifiers are trained using these replicates.

Bootstrap samples are drawn uniformly with replacement from the original training data set. This sampling method leads to the situation where some data points appear multiple times in the bootstrap sample and some of them are missing. If the size of an original

data set is n , the size of a bootstrap sample is n' and if $n = n'$, then for large n each bootstrap sample contains approximately 63% of unique points from the original training data set. These differences between bootstrap samples introduce required diversity among the base classifiers.

Bagging is particularly useful with low-bias and high-variance classifiers like neural networks and decision trees, since bagging is mainly a variance-reduction technique. Reduced variance will lead to higher performance and more robust classification results. If the base classifiers have already low variance, then bagging may not provide any significant improvements.

[13]

3.7 Decision Trees and Random Forest

Decision trees are popular supervised machine learning tools. Wide range of machine learning problems like classification, ranking, probability estimation, regression, and clustering can be solved using tree models. They are expressive and their interpretability for humans is relatively good. Decision trees are able to handle all types of features (quantitative, ordinal, and categorical). Good expression capability of decision trees is based on the fact that decision trees correspond to *DNF* (*disjunctive normal form*) expressions. All logical expressions can be written in DNF form, therefore decision trees are maximally expressive. A negative effect induced by the high expressivity is their tendency to overfit easily. [15][16]

An example of a simple decision tree is illustrated in Figure 19. The top most node, “Color”, is called the *root node*. All blue nodes are called *internal nodes* and they are representing some features describing the data set. Edges exiting from the internal nodes are labelled with values of that particular feature. For example, the feature located at the root node has two possible values, “green” and “blue”. This set of exiting edges is called a *split*; it is splitting the feature space according to the values located at the exiting edges. The outermost nodes, colored in red and without any successors, are called *leaf nodes*. The leaf nodes are responsible for associating class labels for input vectors fed into the decision tree. When an input vector is classified using a decision tree classifier, the tree is traversed from top to bottom, and at each internal node one feature is tested at a time. Traversing is continued until we end up at a leaf node, where class label is associated for the input vector. [16]

Decision trees are grown (trained) from top to bottom by successively partitioning the feature space into smaller chunks. Each split in the decision tree performs partitioning of the feature space into two parts (hyperplanes parallel to the coordinate axes). The goal is that after each split, less mixing between classes is present, i.e., the data become

more pure after each split. This mixing between classes is measured using *impurity measures*.

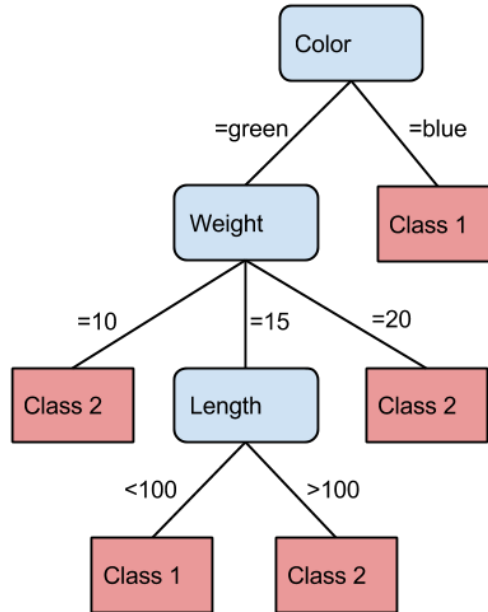


Figure 19 A simple decision tree

Common impurity measures are:

- minority class: $\min(\rho, \rho - 1)$
- Gini index: $2\rho(1 - \rho)$
- entropy: $-\rho \log_2 \rho - (1 - \rho) \log_2(1 - \rho)$

where $\rho = \frac{n^\oplus}{n^\oplus + n^\ominus}$, n^\oplus and n^\ominus are numbers of positive and negative samples in the training data, respectively.

During the training process of a decision tree, it should be decided after each split whether to continue splitting or make the current node a leaf node. This is done by defining a threshold value for the particular impurity measure; if impurity is below the threshold, the current node is made a leaf node; if it is above, splitting shall continue. Trees have tendency to overfit easily, therefore the training process should be terminated before the tree has been grown to a full length. The right moment for stopping growing the tree is when the classification performance over the validation set starts to decrease. Another, yet not so common, approach for preventing overfitting is first to grow the tree to its full length (allow it to overfit) and then afterwards post-prune the tree. [13]

3.7.1 Random Forest

Random forest (RF), illustrated in Figure 20, is a classifier ensemble built up from multiple parallel decision trees. Since decision trees are high-variance and low-bias classifiers, the bagging ensemble technique is utilized. In the RF classifier, an input vector is fed to all decision trees belonging to the forest, and predictions from each individual decision tree are computed. The predictions are combined using majority voting to produce the final output. Trees in the random forest are grown to their full lengths and allowed to overfit to the training data.

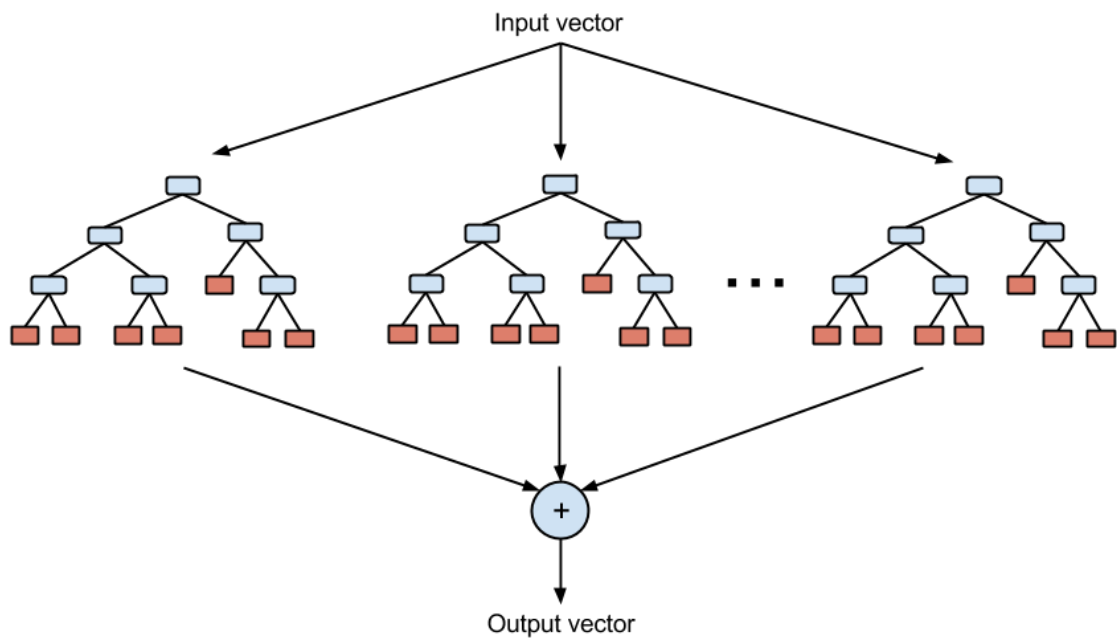


Figure 20 Random forest classifier

According to the bagging ensemble technique, each decision tree in the random forest is constructed using slightly varying bootstrap sample. This introduces required diversity among the base classifiers. In addition to bagging, the *random subspace method* is utilized in tree construction. In the random subspace method, if M is the total number of features, then at each split, a feature is selected for the node from a random subset of size $m \ll M$ features. The reason for this maneuver is to reduce correlation between the trees; if there exist features which are very strong predictors for the output, these features are likely to be selected at the top of most of the trees. This increases correlation between the trees, which has negative impact on the classification performance due to decreased diversity.

[17][13]

3.8 Support Vector Machine

Support Vector Machines (SVM) are linear algorithms for classification and regression. The principle of SVM is to construct separating hyperplane between the classes in such a way that the margin of separation is maximized. This so-called structural risk minimization provides a good generalization performance in SVMs despite the fact that no problem-domain knowledge is involved during the training. This characteristic has made SVMs very popular, since not much parameter tuning is required in order to achieve relatively good classification results.

Originally SVMs are linear and binary classifiers, therefore some extensions have to be made in order to handle non-linear and multiclass data. For multiclass classification, either one-against-one or one-against-all approaches are used to convert a binary SVM into a multiclass classifier. For non-linear data, so called *kernel trick* is used; the data are mapped into a higher-dimensional space where the optimal separating hyperplane is constructed. Kernel trick is based on the *Cover's theorem*, which states that the original data can be mapped into a new high-dimensional space, where the data is linearly separable with high probability.

In order to construct a separating hyperplane between classes, only a subset of training samples, called support vectors, are required. These support vectors define completely the separating hyperplane as is shown in the Figure 21.

[13][18]

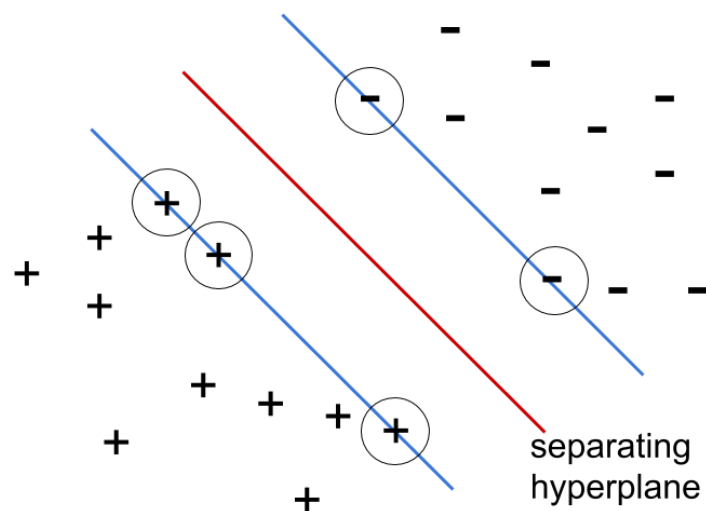


Figure 21 Data points marked with ring are called support vectors

3.8.1 Soft-Margin Non-Linear SVM

In practical applications, perfectly linearly separable data are unusual. Therefore, it is impossible to construct a hyperplane, which could separate all the data without errors. To solve this problem, the so-called soft-margin SVM is used. In soft-margin SVM, some of the data points are allowed to overlap the margin and to be misclassified. Figure 22 illustrates the soft-margin SVM. [13]

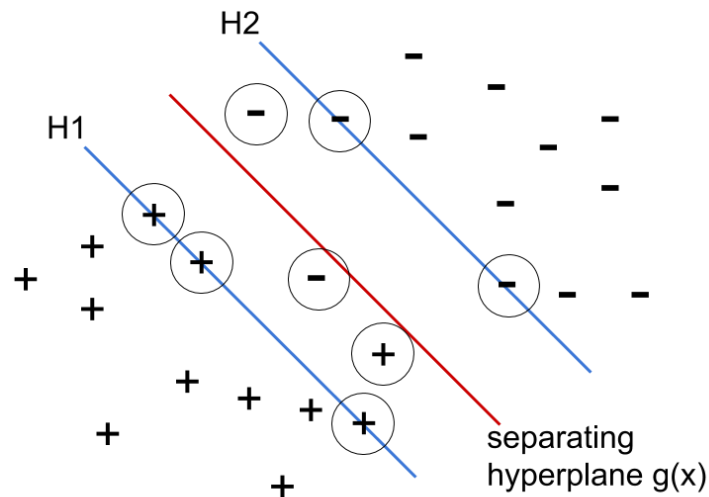


Figure 22 Soft-margin SVM, data points marked with circles are the support vectors

In the following description of soft-margin SVM, the training data set is denoted by $\{\mathbf{x}_i, i = 1, \dots, n\}$ and the corresponding binary class labels by $\mathbf{y}_i = \pm 1$. The discriminant function $g(\mathbf{x})$, defining the separating hyperplane is expressed as

$$g(\mathbf{x}) = \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}) + w_0 \quad (1)$$

where function $\boldsymbol{\phi}()$ is some nonlinear function mapping an input vector \mathbf{x} into a transformed feature space $\boldsymbol{\phi}(\mathbf{x})$. The aim of this is to make the input vectors linearly separable in the $\boldsymbol{\phi}$ space.

During the SVM training procedure, optimal values for \mathbf{w} and w_0 are found in such a way that the margin of separation between classes is maximized.

Class labels for input patterns \mathbf{x}_i are assigned according to the decision rule:

$$\begin{aligned} g(\mathbf{x}_i) > 0 &\Rightarrow \mathbf{y}_i = +1 \\ g(\mathbf{x}_i) < 0 &\Rightarrow \mathbf{y}_i = -1 \end{aligned} \quad (2)$$

Canonical hyperplanes $H1$ and $H2$ (Figure 22) are defined using equations:

$$\begin{aligned}
H1: \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}) + w_0 &= +1 \\
H2: \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}) + w_0 &= -1
\end{aligned} \tag{3}$$

The perpendicular distance between the canonical hyperplanes $H1$ and $H2$ and the separating hyperplane $g(\mathbf{x})$ is $1/|\mathbf{w}|$, hence the total margin is $2/|\mathbf{w}|$. Therefore, in order to maximize the margin of separation, $|\mathbf{w}|$ must be minimized. All points that lie inside the margin or on the canonical hyperplanes, are called support vectors (Figure 22).

The cost function to be minimized is

$$\frac{1}{2} \mathbf{w}^T \mathbf{w} + C \sum_i \xi_i \tag{4}$$

subject to the following constraints

$$\begin{aligned}
\mathbf{y}_i(\mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}_i) + w_0) &\geq 1 - \xi_i & i = 1, \dots, n \\
\xi_i &\geq 0 & i = 1, \dots, n
\end{aligned} \tag{5}$$

Slack variables ξ_i allow some of the data points to overlap the margin. The regularization term $C \sum_i \xi_i$ in Equation 4 is controlling smoothness of the decision surface. Changing the value of the regularization parameter C will change the number of data points allowed to lie inside the margin. The lower the value of C , the less penalty is given for the data points inside the margin, and thus the smoother the decision surface becomes.

The method of Lagrange multipliers is used for solving this kind of constrained optimization problem. Dual form of the Lagrangian is

$$L_D = \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j \boldsymbol{\phi}^T(\mathbf{x}_i) \boldsymbol{\phi}(\mathbf{x}_j) \tag{6}$$

subject to the constraints

$$\begin{aligned}
\sum_{i=1}^n \alpha_i y_i &= 0 \\
0 &\leq \alpha_i \leq C
\end{aligned} \tag{7}$$

This constrained optimization problem can be solved by using numerical quadratic programming solvers. Lagrangian multipliers α_i are obtained as a solution to the problem. The weight vector \mathbf{w} is solved from the equation:

$$\mathbf{w} = \sum_{i \in SV} \alpha_i y_i \boldsymbol{\phi}(\mathbf{x}_i) \quad (8)$$

Substituting the weight vector (Equation 8) into the linear discriminant function (Equation 1) we get

$$g(\mathbf{x}) = \sum_{i \in SV} \alpha_i y_i \boldsymbol{\phi}^T(\mathbf{x}_i) \boldsymbol{\phi}(\mathbf{x}) + w_0 \quad (9)$$

where

$$w_0 = \frac{1}{n_{\widehat{SV}}} \left\{ \sum_{i \in \widehat{SV}} y_i - \sum_{i \in SV, j \in \widehat{SV}} \alpha_i y_i \boldsymbol{\phi}^T(\mathbf{x}_i) \boldsymbol{\phi}(\mathbf{x}_j) \right\}. \quad (10)$$

SV is the set of all support vectors and \widehat{SV} is the set of support vectors lying on the canonical hyperplanes (boundaries of the margin). A new input vector \mathbf{x} is assigned a class label according to the sign of Equation 9 in the same manner as is stated in the decision rule (Equation 2).

Because inner products $\boldsymbol{\phi}^T(\mathbf{x})\boldsymbol{\phi}(\mathbf{y})$ are appearing in both Lagrangian L_D (Equation 6) and in discriminant function $g(\mathbf{x})$ (Equation 9), the inner products can be replaced by kernel functions:

$$K(\mathbf{x}, \mathbf{y}) = \boldsymbol{\phi}^T(\mathbf{x})\boldsymbol{\phi}(\mathbf{y}) \quad (11)$$

This is called the *kernel trick*. When the kernels are used, explicit calculation of transformation $\boldsymbol{\phi}(\mathbf{x})$ can be avoided and instead of that, it is possible to operate directly in the original input space. This reduces the computational load of SVMs. In theory, limitations exist for acceptable kernel functions; they must satisfy *Mercer's conditions*. The most commonly used kernel functions are polynomial, Gaussian and sigmoid kernels.

During the construction of an SVM classifier, kernel type and kernel parameters have to be selected, as well as the value for the regularization parameter C . Often these parameters are defined using the grid-search approach, as in LIBSVM C++ software implementation.

[13][18]

3.9 Multilayer Perceptron Neural Network

Artificial neural networks (ANN) are biologically inspired learning systems, which are partially imitating the functionality of neuronal connections in the brain. Although functionality of ANNs has biological background, in practical applications many simplifica-

tions are made compared to real biological neural systems. Despite the simplified nature of ANNs, they are very efficient methods for solving many practical problems. For example, ANNs have been utilized in vehicle control, speech recognition, face identification, target recognition and image compression. ANNs are defined as adaptive systems, capable of adapting to their environment by learning from experience. Knowledge acquired during the learning process is stored into neuronal connections of the network.

The most common, and the most successful ANN model is the *multilayer perceptron (MLP)*. MLPs are universal function approximators, which means that they are capable of learning any non-linear continuous function. Mathematical foundations of this characteristic are based on the *universal approximation theorem*. The universal approximation theorem states that MLP using a single hidden layer is capable of approximating any continuous function with arbitrary high accuracy. Although the theory states that single hidden layer is enough to approximate any continuous function, it does not state that single hidden layer is always the most optimal choice. The number of required hidden layers is dependent on the problem at hand, and sometimes only one hidden layer is impractical in the sense of learning time and generalization performance.

[13][14][15][18]

3.9.1 Architecture

MLP consists of an *input layer*, one or more *hidden layers* and an *output layer*. Figure 23 illustrates the architecture of the MLP. The input signal is fed to the input layer of the network, from where it propagates through the network on layer-by-layer basis. When the signal has reached the output layer, the result of computation can be read from the output neurons. The most common approach to interconnect the neurons of an MLP is to connect each neuron from the previous layer to all the neurons on the next layer (*fully connected*).

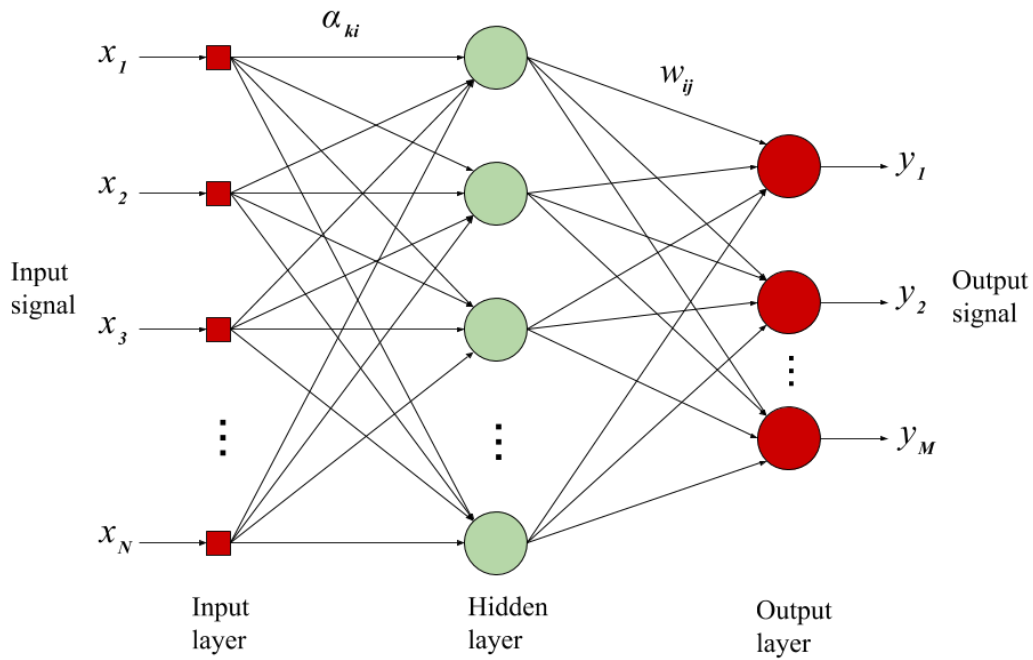


Figure 23 Architecture of a multilayer perceptron

MLP projects an input vector \mathbf{x} from an original input space to a nonlinear hidden space formed by the outputs of the hidden neurons. After this transformation, linear combination of outputs from the hidden neurons are calculated at the output layer. The MLP model is defined by the following equation:

$$y_j(\mathbf{x}) = \sum_{i=1}^L w_{ij} \phi \left(\sum_{k=1}^N \alpha_{ki} x_k + \alpha_{0i} \right) + w_{0j} \quad (12)$$

where

- indices i and j refer to the i -th hidden neuron and the j -th output neuron, respectively
- α_{0i} and w_{0j} are bias terms
- an input vector $\mathbf{x} \in \mathbb{R}^N$
- $\phi()$ is a nonlinear activation function.

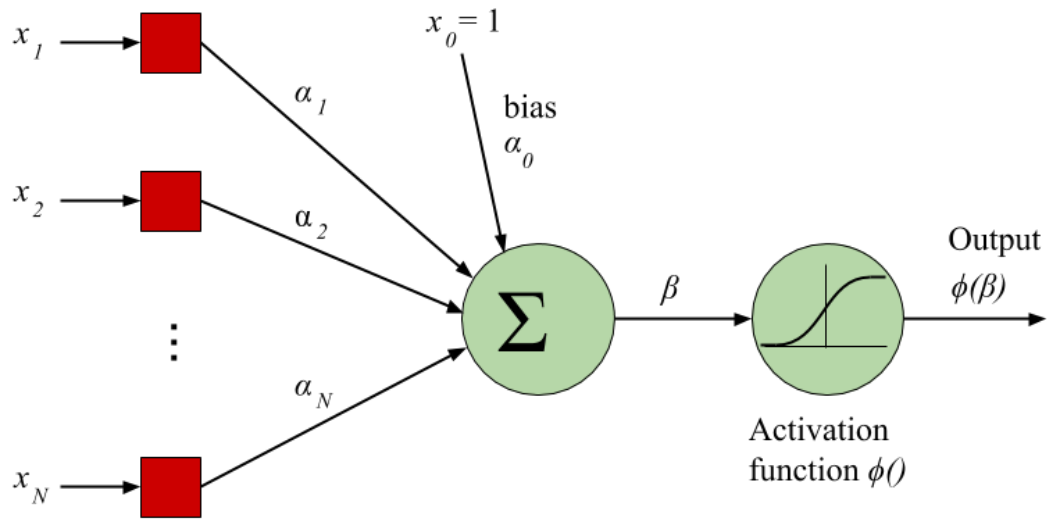


Figure 24 Nonlinear hidden unit

The hidden neurons, as the name implies, are hidden computation units inside the network. Their task is to extract meaningful features from the data, and to transform the data from the original input space to the hidden space, where the data is more easily separable. Figure 24 illustrates how the signal is flowing through a single hidden unit (neuron). In the first step, scalar product $\beta = \boldsymbol{\alpha}^T \mathbf{x} + \alpha_0$ is computed by projecting the input vector \mathbf{x} on the weight vector $\boldsymbol{\alpha}$. Next, nonlinear transformation $\phi(\beta)$ is applied. After that, the signal is propagated to the next layer of the network (to the output layer or the next hidden layer). The function $\phi()$ is called an activation function and its purpose is to add nonlinearity to the network. The most commonly used activation functions are the sigmoid function:

$$\phi(\beta) = \frac{1}{1 + e^{-\beta}} \quad (13)$$

and the hyperbolic tangent:

$$\phi(\beta) = \tanh(\beta). \quad (14)$$

Some requirements apply for the activation functions; they must be differentiable and nonlinear. If the activation functions were not differentiable, training algorithms utilizing derivatives could not be used. If the activation functions were linear, the whole network could be reduced to ordinary linear single-layer perceptron and nonlinear decision surfaces could not be created. MLP's ability to learn complex nonlinear decision boundaries is based on the nonlinearity of hidden neurons, thus nonlinear neurons are crucial for MLPs.

[13][15][18]

3.9.2 Training

The aim of the MLP training process is to define which features in the training data should be represented by the hidden neurons of the network. This is achieved by adjusting network weights in such a way that the network's classification error is minimized. A popular training technique for adjusting network's weights is the *error back-propagation algorithm*. It is based on the error-correction learning rule, which reduces the classification error of the network in a step-by-step fashion.

The error back-propagation algorithm consists of two distinct steps; a forward pass and a backward pass. In the forward pass, the training pattern is applied to the input layer, and is propagated through the network in order to compute the network's actual response to a particular training pattern. When the actual response is known, the error signal can be calculated by subtracting the network's actual response from the desired response for the particular training pattern. In the backward pass, the error signal is propagated backwards through the network on layer-by-layer basis, and the weights of the network are adjusted in order to move the actual response closer to the desired response.

A commonly used error measure for MLPs is:

$$E = \frac{1}{2} \sum_{n \in N} \sum_{j \in \text{outputs}} [t_j(n) - y_j(n)]^2 \quad (15)$$

where N is the set of all training patterns, *output* is a set of all output neurons of the network, and $t_j(n)$ and $y_j(n)$ are the target value and the actual response of the output neuron j to the training pattern n , respectively. The goal of the training process is to adjust the network's weights in such a way that error E is minimized.

Weights w_{sr} between neurons s and r are updated according to the *delta rule*:

$$w_{sr} = w_{sr} + \Delta w_{sr} \quad (16)$$

$$\Delta w_{sr} = \eta \delta_s(n) x'_s(n) \quad (17)$$

where $x'_s(n)$ is the input signal to the neuron s , and η is the learning rate parameter. The term $\delta_s(n)$ denotes local gradient which depends on whether the neuron is located in the hidden or in the output layer of the network.

For the neuron in the output layer the local gradient is defined as

$$\delta_s(n) = e_j(n) \phi'_s(v_s(n)) \quad (18)$$

and for the hidden layer neuron

$$\delta_s(n) = \phi'_s(v_s(n)) \sum_q \delta_q(n) w_{qs}(n) \quad (19)$$

where

- $\phi'_s()$ is the first order derivative of the activation function
- $v_s(n)$ is the induced local field (weighted sum of all inputs to the neuron) of neuron s for the training pattern n
- $e_j(n)$ is the error signal of the output neuron j for the training pattern n , $e_j(n) = t_j(n) - y_j(n)$
- $\delta_q(n)$ is the local gradient of the neuron q , which is lying directly on the next layer from the neuron s
- $w_{qs}(n)$ is the weight between the neuron q and s .

The weight adjusting method represented above is utilizing simple gradient descent search over the weight space; the weight update step is always performed in the direction of the deepest descent of the gradient. This approach is suffering from slow convergence and therefore more sophisticated optimization techniques, like Quasi-Newton and Conjugate Gradient methods, have been developed in order to make the convergence faster. Another issue in MLP training is that the error surface contains many local minima. Because of backpropagation algorithm's iterative and numerical nature, it is possible to get trapped in one of these local minima having negative influence on classification performance. Although it is not guaranteed that the global minimum is ever found, studies have shown that in practice the backpropagation algorithm is a very feasible way to train MLPs, and the problem of local minima is more theoretical than a practical issue.

Since MLP is a very flexible machine learning model, it is very prone to overfitting. Overtraining of a network will lead to adaptation to noise properties of the training data, and thus poor generalization accuracy. The most common way to avoid overtraining the network is the cross-validation approach. In cross-validation, the training data are divided into two separate sets; training and validation sets. The training set is used to tune the network weights using the backpropagation algorithm, meanwhile the validation set is used only for monitoring the classification error E . The training algorithm is stopped when the lowest classification error is reached over the validation set.

[15][18][13]

3.10 K-Nearest Neighbor Classifier

K-nearest neighbor (KNN) algorithm is one of the simplest machine learning techniques. It is a memory based (sometimes also called lazy learning) technique for classification and regression. The KNN algorithm stores training examples explicitly into the memory in order to classify input patterns. The principle of the KNN-classifier is to find k nearest training patterns for an input vector \mathbf{x} using some distance metric. The input pattern \mathbf{x} is assigned to a majority class among the k training patterns. Figure 25 illustrates the KNN-classifier in a two class case when $k=8$.

Training of KNN classifier is fast because everything required is just to load the data into memory. On the other hand, classifying a new input pattern is slow, because pairwise distances have to be calculated between all points in the data set. For large data sets this kind of linear search is too slow and more advanced data structures, like clustering or kd-trees, for storing the training data have to be used.

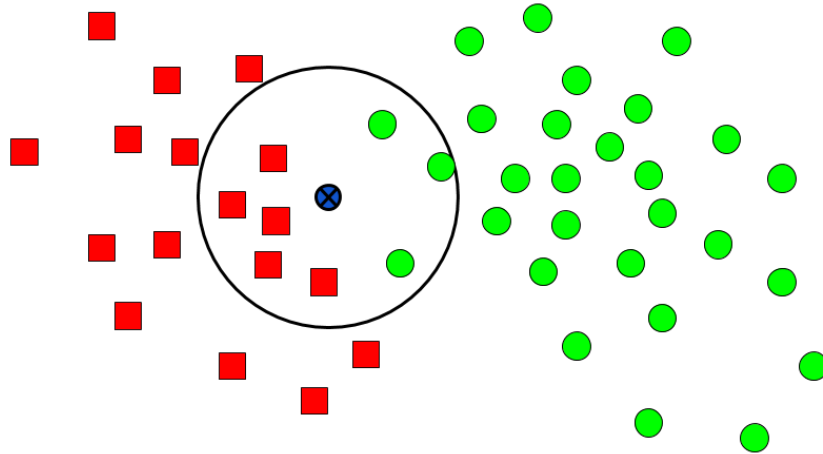


Figure 25 K-Nearest Neighbor classifier, $k=8$

An important aspect of the KNN classifier is the selection of the distance metric. The choice of the distance metric can have significant impact on classification performance. The most common distance metric used in the KNN-algorithm is the *Minkowski distance*:

$$d_p(\mathbf{x}, \mathbf{y}) = \left(\sum_{j=1}^d |x_j - y_j|^p \right)^{\frac{1}{p}} \quad (20)$$

where $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$.

The Euclidean distance (2-norm)

$$d_2(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{j=1}^d (x_j - y_j)^2} \quad (21)$$

and the *Manhattan distance* (*1-norm*)

$$d_1(\mathbf{x}, \mathbf{y}) = \sum_{j=1}^d |x_j - y_j| \quad (22)$$

are the two most common special cases of the Minkovski distance metric. As a remark, Minkowski distances with $p < 1$ are not considered as distance metrics, since they violate the *triangle inequality* rule.

The implementation of the KNN-algorithm is relatively straightforward; only two parameters have to be selected; the distance metric and the number of neighbors, k . The number of neighbors influences the sensitivity of the classifier; the greater the k , the less sensitive the classifier is to noise. Large k makes also transition zones between the classes fuzzier, and hence more mixing might occur.

Although at the first sight the KNN-algorithm might seem as a very easy and fast classification method, it has several drawbacks. It is suffering from the curse-of-dimensionality; in high-dimensional spaces pairwise distances between points tend to be nearly equal making use of distance metrics meaningless. To overcome this issue, different dimensionality reduction methods, like feature selection or transformation, have to be used. This again increases complexity of the pattern recognition system. Another drawback of KNN was already mentioned above; it cannot handle large datasets without more advanced data structures for storing the training data. Usually a lot of tweaking is required in order to use KNN classifier in complex pattern recognition tasks.

[16][15][13]

4. FEATURE SET SELECTION AND COMPARISON OF CLASSIFICATION TECHNIQUES USING TERRASAR-X TEST DATA

In this chapter, the feature set and classification algorithms used in the thesis are described. The task at hand was to study the feasibility of different feature types and pattern recognition techniques for ground cover and vegetation classification from spaceborne SAR-images. Monitoring of environmental changes by physically visiting the area at interest is usually slow and time consuming process, especially in the case of rough terrain. Using pattern recognition and remote sensing techniques for environment monitoring could significantly ease this process, since large areas can be mapped efficiently without physical attendance in the field.

In this thesis, 4 different classifier types with 5 separate feature groups were tested. The tested classifiers were Random Forest, Support Vector Machine, Neural Network and K-Nearest Neighbor. The feature set used in classification included the following feature groups: Basic features, Textural features, Statistical features, features based on the Gray-Level Co-Occurrence Matrix and Local Binary Patterns. Classification performance of the algorithms were tested over all combinations formed from these feature groups, resulting in total number of 31 different feature group combinations. Data processing was done using Matlab built-in tools and external software libraries (*libSVM* and *R randomForest* package)

4.1 Description of the Test Area and the Data

Test data for classification algorithms consisted of a SAR image obtained from Lake Poosjärvi (61.633864°, 21.906526°) at 16:12 UTC on 4th of July 2014. The SAR image was taken with TerraSAR-X satellite operating in the Spotlight mode (X-band) using dual polarizations (HH, VV). Cross-polarizations (VH, HV) were not used. The image was acquired with ascending orbit and incidence angle of 44.17°. The same test data have been previously used in the paper by Kumpumäki and Lipping [19].

Lake Poosjärvi is a small (area of 794 hectares) lake located in the western part of Finland in the Satakunta region. It lies approximately 20 km north from the city of Pori (Figure 26). The forest type at Lake Poosjärvi is mainly mixed forest, and ground vegetation is dominated by sedges, water horsetail, water lilies, pondweeds and reed. [20]



Figure 26 Location of Lake Poosjärvi (red circle)

Vegetation was divided into 7 classes:

- | | |
|--------------------|----------------------|
| 1. reed | 5. trees |
| 2. water horsetail | 6. sedge |
| 3. reed in water | 7. yellow water lily |
| 4. water | |

Reeds growing in water and on dry land were separated into two classes since their radar echoes were significantly different, which might have had caused mixing with other classes. Ground truth data around Lake Poosjärvi were available from the areas shown in Figure 27.

SAR-data preprocessing included geometrical correction and calibration using Sentinel-1 toolbox delivered by the satellite operator. Original irregularly sampled data were interpolated into 1x1 m raster using the nearest neighbor interpolation scheme. This maneuver caused some duplicates to the data, since the resolution of the raster was selected high enough to preserve all the original data points.

Speckle noise filtering was not used for the SAR-image, since experiments showed better classification results for non-filtered data. In experiments, filtering was conducted with 2D Gaussian (sigma=1.75) weighted bootstrap filter in 7x7 pixel sliding window. However, non-filtered data resulted in better classification performance for all classifiers and therefore filtering was not used. Higher performance for non-filtered data might occurred because the filtering smoothed and removed important textural information from the image, and secondly, the applied classifiers (RF, NN, SVM) are not very sensitive to the noise in the data. Also, speckle is not a noise in the traditional sense; it is a

physical measurement of contents inside a resolution cell, and therefore it might carry some information about imaged areas.

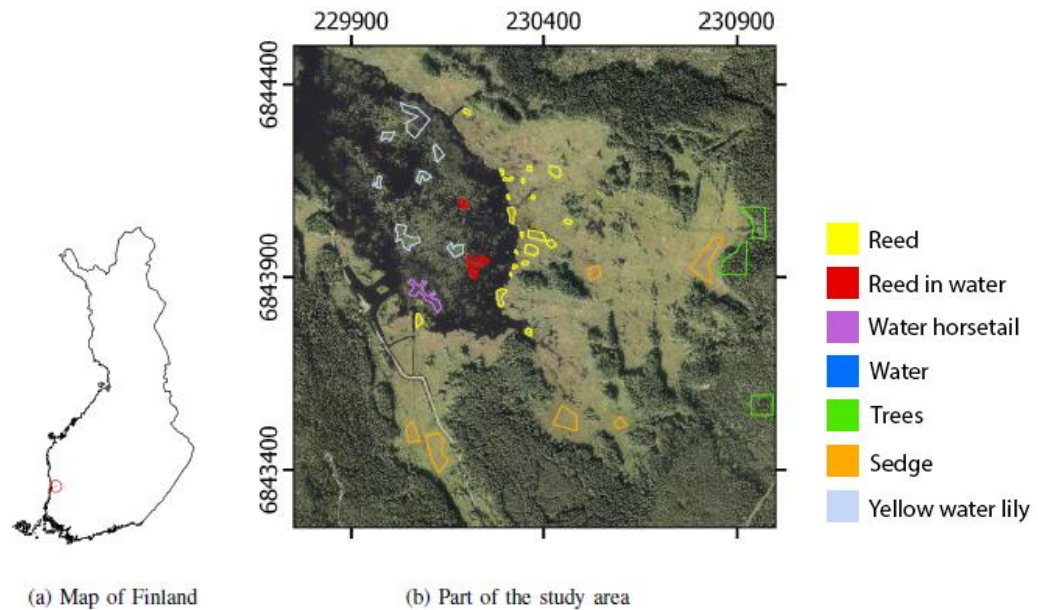


Figure 27 Ground truth data areas around Lake Poosjärvi [19]

The ground truth data was divided randomly into training and test set. The training set consisted of 2500 samples for each individual class, with the exception of 1491 samples for reed in water. The test set included 250 samples for each class.

4.2 Feature Set Used in Classification

Five different groups of features were used in the thesis: Basic features, Textural features, Statistical features, features based on the Gray-Level Co-Occurrence Matrix (GLCM) and Local Binary Patterns (LBP). This resulted in total number of 57 features. Although GLCM and LBP are considered as textural features, they were arranged as separate feature groups, because testing their impact on classification performance was one of the objectives of the study. Also, it should be noted that the names of the feature groups are suggestive. For example, statistical operators are involved also in other than statistical feature group.

The initial feature set was selected based on what has been proposed in the literature. Features in the initial feature set were tested by trial-and-error; if a particular feature had positive impact on classification performance, it was included to final feature set. For instance, in addition to GLCM and LBP, fractal dimension (box counting method) and edge detection based textural features were initially included, but no improvement was observed, and thus they were left out from the final feature set.

In the following list all the features used in the classification are defined. The operator M_{size} denotes a boxcar filter of size $size$, and B_{stat} denotes a 2D Gaussian (sigma=1.75) weighted bootstrap of statistic $stat$ in 7x7 pixel sliding window.

Basic features

The following features were included in the Basic features group:

1. $B_{mean}\left(\frac{|HH|}{|HH|+|VV|}\right)$
2. $B_{mean}(|HH| * |VV|)$
3. $\sqrt{B_{mean}(|HH|^2)}$
4. $\sqrt{B_{mean}(|VV|^2)}$
5. $|\Delta|$, where $\Delta = \frac{E(HH*VV^*)}{\sqrt{E(|HH|^2)}\sqrt{E(|VV|^2)}}$
6. $|\arg \Delta|$
7. $M_{3x3}(|HH + VV|)$
8. $M_{3x3}(|HH - VV|)$
9. $M_{3x3}(|HH| + |VV|)$
10. $M_{3x3}(|HH| - |VV|)$
11. $|HH|$
12. $|VV|$

Textural features

The following features were included in the Textural features group:

13. $B_{std}(|HH + VV|)$
14. $B_{std}(|HH - VV|)$
15. $B_{mean}(|HH + VV|)$
16. $B_{mean}(|HH - VV|)$
17. $B_{mean}(|HH + VV|^2)$
18. $B_{mean}(|HH - VV|^2)$
19. *difference of variance of |HH + VV| in 7x7 window*
20. *difference of variance of |HH - VV| in 7x7 window*
21. *Entropy(|HH + VV|) over round window of radius 9 pixels*
22. *Entropy(|HH - VV|) over round window of radius 9 pixels*
23. *Entropy(|HH| + |VV|) over round window of radius 9 pixels*
24. $M_{3x3}(| |HH + VV| - M_{7x7}(|HH + VV|) |)$
25. $M_{3x3}(| |HH - VV| - M_{7x7}(|HH - VV|) |)$

Entropy is calculated using the following equation:

$$-\sum_{i=1}^{256} hist(i) * \log_2 hist(i) \quad (23)$$

where $hist(i)$ means sample count of the i -th bin in the histogram derived from the 8-bit intensity image. [21]

Statistical features

The following features were included in the Statistical features group:

- 26. $Median_{15 \times 15}(|HH + VV|)$
- 27. $Median_{15 \times 15}(|HH - VV|)$
- 28. $25\text{ percentile}_{15 \times 15}(|HH + VV|)$
- 29. $25\text{ percentile}_{15 \times 15}(|HH - VV|)$
- 30. $75\text{ percentile}_{15 \times 15}(|HH + VV|)$
- 31. $75\text{ percentile}_{15 \times 15}(|HH - VV|)$
- 32. $Range_{15 \times 15}(|HH - VV|)$
- 33. $Range_{15 \times 15}(|HH - VV|)$

Operator $Stat_{size}$ is a statistic $Stat$ calculated in a sliding window of size $size$.

Gray-Level Co-Occurrence Matrix (GLCM)

The Gray-Level Co-Occurrence Matrix (GLCM) is a textural feature extraction technique. The idea behind the GLCM is to count occurrence frequencies of gray tone values of neighboring pixel pairs in the image. If the image contains large homogenous areas (i.e. coarse textures), then it is assumed that neighboring pixels represent relatively similar intensity values. This causes high values in the GLCM to concentrate on its main diagonal. On the other hand, if the image contains very fine-grained texture (i.e., large fluctuations in intensity values within a small region), the intensity values of neighboring pixels represent larger variation, and thus high values are spread out from the GLCM main diagonal. As an extreme case, if pixel values of the image were selected randomly, the values in the GLCM would represent uniform distribution. [17]

GLCM itself is a matrix the entries of which represent occurrence frequencies of two pairs of gray level intensities for certain direction and distance d between the pixels. Practically speaking, only 4 different directions are used: 0° , 45° , 90° and 135° . Figure 28 illustrates an example of GLCM computation for 4x4 image with 4 different gray level intensities. For instance, in Figure 28 (c), the value in cell (2,3) represents how many times gray level (2,3) appears in horizontal direction with distance $d=1$. Common practice is to normalize frequency values into the range [0,1] to avoid scaling effects. [17]

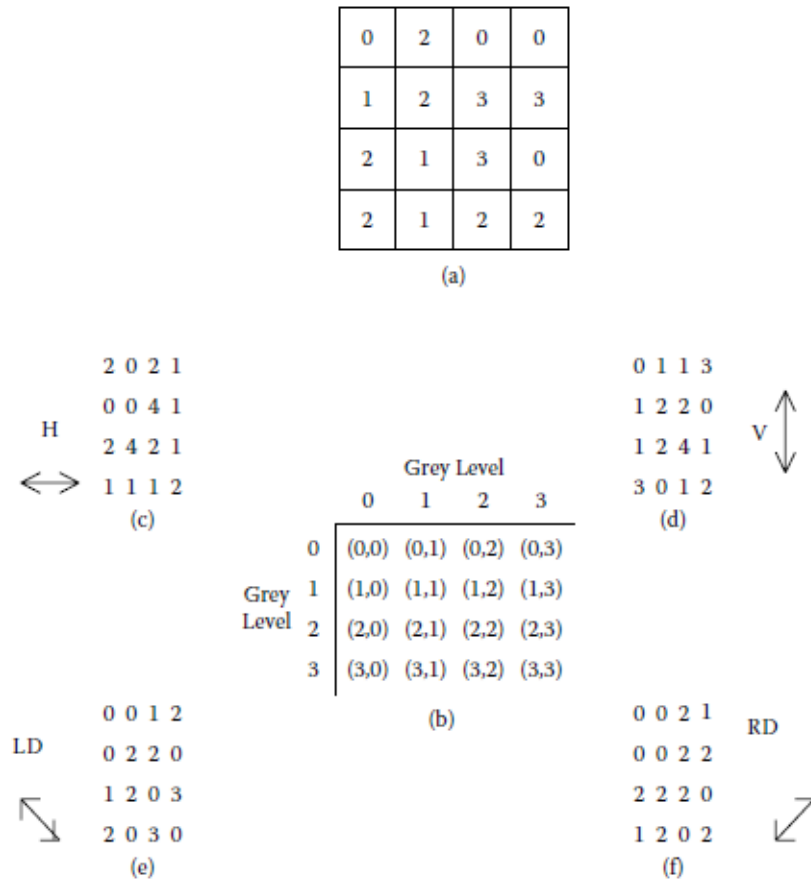


Figure 28 Illustration of GLCM, (a) is a gray-tone image with range of 4 gray level values, (b) is the matrix of different gray level combinations for neighboring pixels, (c) – (f) are GLCMs calculated from the image (a) for four directions with angles 0° , 45° , 90° , 135° , and with distance $d=1$ [17]

Wide range of different texture measures can be calculated from GLCMs. In his original paper Haralick [22] proposed to use 14 different texture measures for each direction, then to calculate mean and range for each measure, and to use these as inputs to a classification algorithm. This procedure is not widely adopted and usually much lower number of texture measures is used to avoid highly correlated input data for a classifier. Texture measures can be made rotation invariant by taking the average of the four directions. In this thesis, four different GLCM texture measures were used: [17][23]

1. Mean [23]

$$\sum_{i,j} ip(i,j) \quad (24)$$

2. Contrast [24]

$$\sum_{i,j} |i - j|^2 p(i,j) \quad (25)$$

3. Correlation [24]

$$\sum_{i,j} \frac{(i - \mu_i)(j - \mu_j)p(i,j)}{\sigma_i \sigma_j} \quad (26)$$

4. Energy [24]

$$\sum_{i,j} p(i,j)^2 \quad (27)$$

where

- i and j refer to GLCM rows and columns, respectively
- $p(i,j)$ is a value of an element in the j -th row and i -th column of the GLCM
- μ is the mean value of the GLCM
- σ_i and σ_j are row- and column-wise variances, respectively

GLCMs were calculated for $|HH + VV|$ polarization combination in 15x15 sliding window using the following parameters:

- four directions with angles of 0° , 45° , 90° and 135°
- distance of 1 pixel between neighbors
- quantization to 20 gray levels \Rightarrow size of GLCM is 20x20.

This parameter set resulted in 4 different GLCMs (one for each direction). For each of the GLCMs, four texture measures with distance $d = 1$ were calculated: Mean, Contrast, Correlation and Energy. Figure 29 illustrates the GLCM neighborhood. All texture measures were made rotation invariant by taking the average value over the four directions.

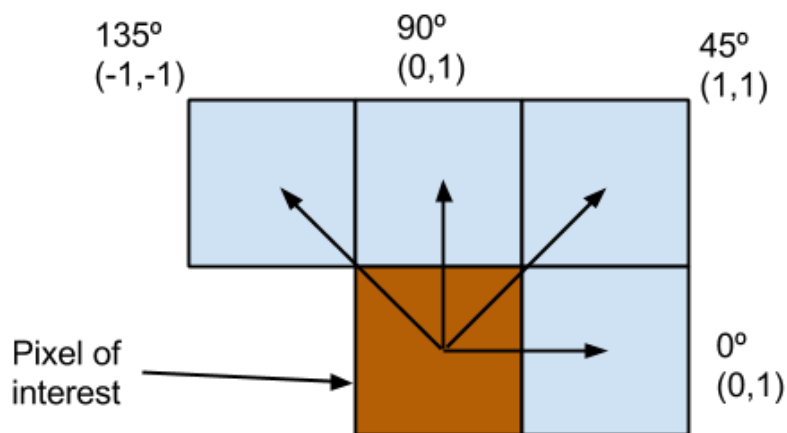


Figure 29 GLCM neighborhood

Local Binary Patterns (LBP)

Local Binary Pattern (LBP) is simple yet effective method for texture feature extraction. It is fast to compute, invariant to monotonic grayscale changes and it can be made rotation invariant if desired. In LBP, the analysis window is divided into square cells, where the center pixel is compared to its neighboring pixels. The commonly used circular neighborhood is shown in Figure 30.

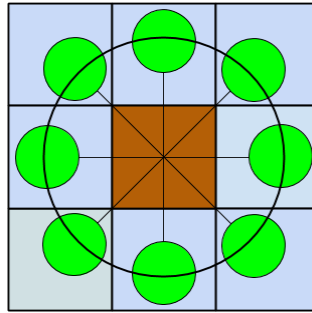


Figure 30 LBP circular 3x3 neighborhood

Figure 31 presents an example of LBP computation. All neighboring pixels are compared to the center pixel; if the center pixel is greater than the neighboring pixel, the pixel is denoted with 0; if the center pixel is equal or smaller than the neighboring pixel, the pixel is denoted with 1. In Figure 31, the resulting binary values (on the right-hand side) correspond to binary sequence 11110000. The neighborhood can be traversed either clockwise or counter-clockwise and the starting point can be chosen freely as long as the same procedure is applied in all analysis windows. The *LBP code*, denoted by $LBP_{P,R}$, is calculated according to the following equation:

$$LBP_{P,R} = \sum_{p=0}^{P-1} s(g_p - g_c) 2^p \quad (28)$$

where P is the number of sampling points, R is the radius of the neighborhood, g_p is the p -th neighboring pixel, g_c is the center pixel, and $s(z)$ is the thresholding function

$$s(z) = \begin{cases} 1, & z \geq 0 \\ 0, & z < 0 \end{cases} \quad (30)$$

If a sampling point does not fall at integer coordinates, bilinear interpolation is used.

The resulting LBP code for the example shown in the Figure 31 is:

$$LBP_{8,1} = 1 * 2^0 + 1 * 2^1 + 1 * 2^2 + 1 * 2^3 + 0 * 2^4 + 0 * 2^5 + 0 * 2^6 + 0 * 2^7 = 15$$

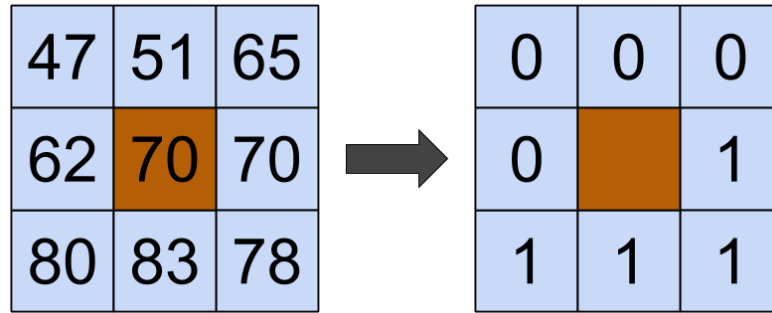


Figure 31 Example of LBP computation. The block on the left side represents a single cell in the analysis window. The block on the right side shows the resulting binary values.

After the computation of LBP codes for all cells in the analysis window, the LBP histogram is created. The LBP histogram is a feature describing the texture within the analysis window. In the LBP histogram, each histogram bin corresponds to single LBP code value. This means, that there exist as many bins in the histogram as there are different LBP codes. For instance, the histogram for the (8,R) neighborhood would include $2^8 = 256$ bins.

In many cases, an extension called *uniform patterns* are used to reduce the length of the LBP histogram and to make it rotation invariant. A uniform pattern is defined by counting the number of bitwise transitions of LBP bit sequence when it is traversed circularly; if the bit sequence includes at most 2 bitwise transitions from 0 to 1 or vice versa, it is defined as a uniform pattern. For example, the bit sequences 00000000 (0 transitions), 00110000 (2 transitions) and 11100111 (2 transitions) are uniform whereas the sequences 11001001 (4 transitions) and 01010010 (6 transitions) are not. Figure 32 illustrates all the 58 different uniform patterns for the (8, R) neighborhood. In the LBP histogram, every uniform pattern has its own individual bin whereas all non-uniform patterns are assigned to a single bin.

Uniform patterns are employed when LBP histograms are made rotation invariant; rotation invariance is achieved by circularly rotating each uniform pattern to its minimum value. For example, patterns 10000011, 11100000 and 00111000 are just different rotations of the same “minimum value pattern” 00000111. In rotation invariant LBP histogram, all uniform patterns showing this behavior are assigned to the same histogram bin, i.e., all the rows in Figure 32 are representing single bin. [25]

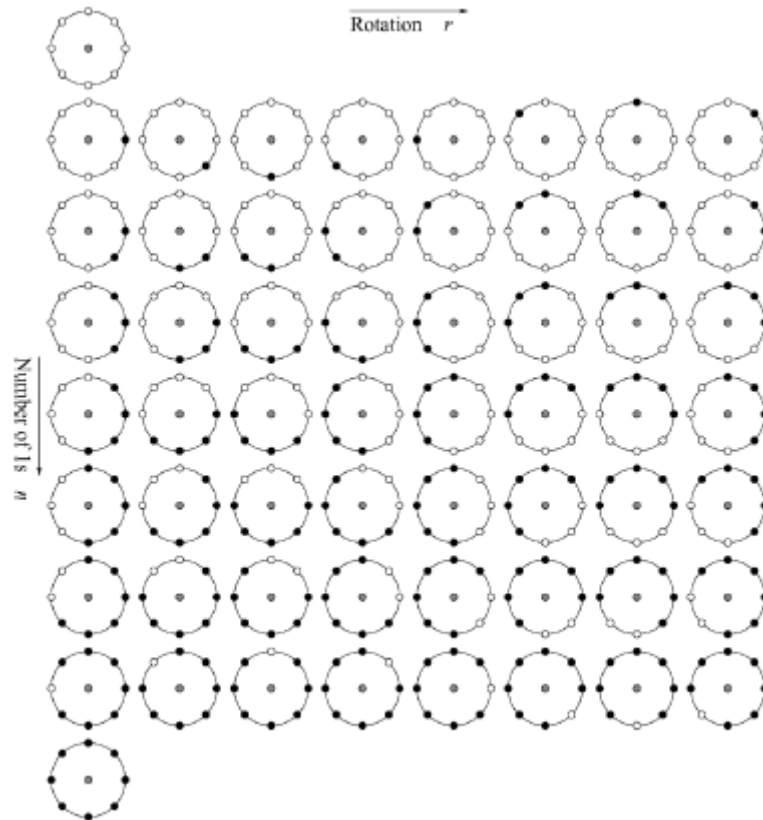


Figure 32 Different uniform patterns for the $(8, R)$ neighborhood [25]

In this thesis, $|HH + VV|$ and $|HH - VV|$ polarization combinations were used for LBP features. Circular $(8,1)$ neighborhood, shown in Figure 30, with uniform and rotation invariant patterns was used, which resulted in 2 LBP histograms with 10 bins in each.

4.3 Visualization of Features

Features were visualized by calculating the so-called feature fingerprints for each class. Feature fingerprint is an image showing distributions of features within a single class. Feature fingerprint consists of feature histograms (showing distribution of values of a particular feature) stacked one after another in the x-direction. Distribution of values of a particular feature is shown along the y-direction; the darker the color, the more values are concentrated around the certain value. Feature fingerprints for each class are shown in Figures 33-39. Values along the x-axis correspond to individual features numbered as follows:

- 1-12 Basic features
- 13-25 Textural features
- 26-33 Statistical features
- 34-37 GLCM features
- 38-57 LBP histogram features

Prior to plotting the fingerprints, the data were normalized by dividing by standard deviation and subtracting the mean. LBP features appear discontinuous in the fingerprints since they already represent discrete histograms.

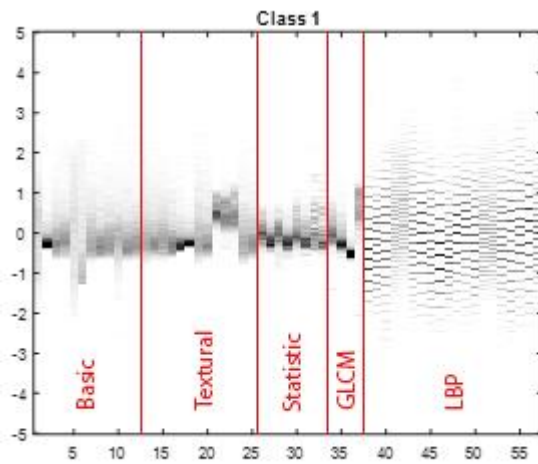


Figure 33 Feature fingerprint for reed (class 1)

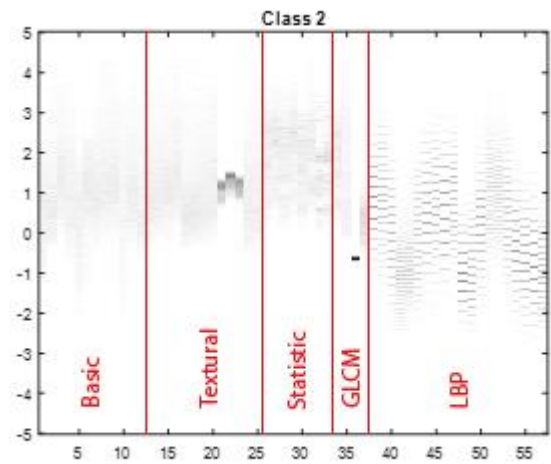


Figure 34 Feature fingerprint for water horsetail (class 2)

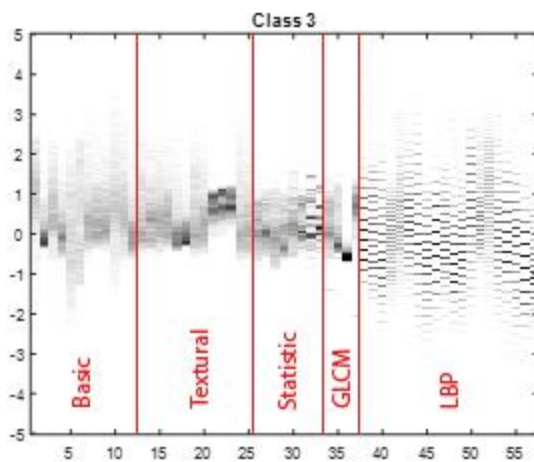


Figure 35 Feature fingerprint for reed in water (class 3)

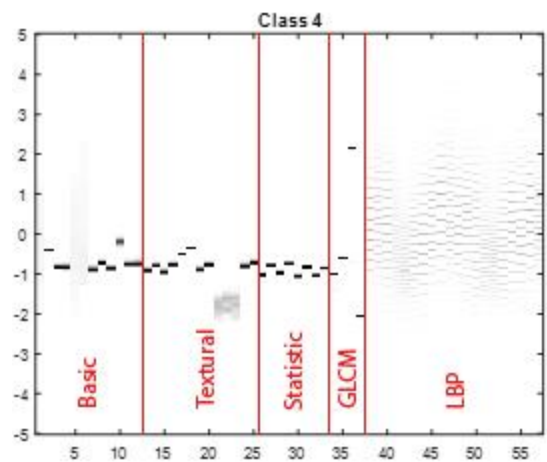


Figure 36 Feature fingerprint for water (class 4)

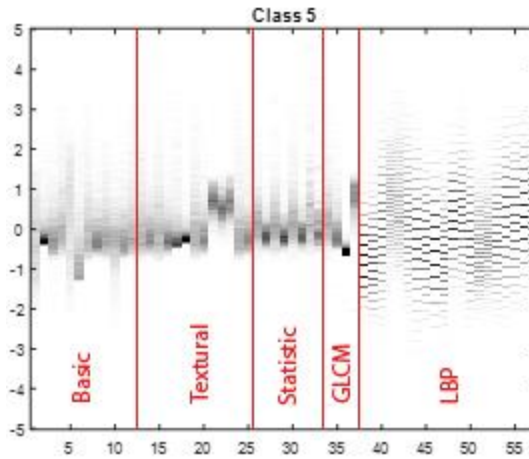


Figure 37 Feature fingerprint for *trees* (class 5)

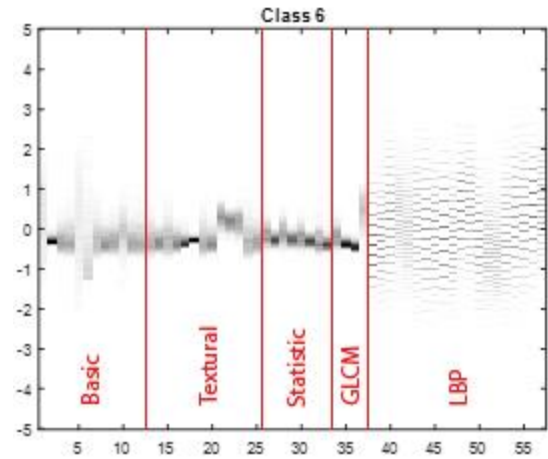


Figure 38 Feature fingerprint for *sedge* (class 6)

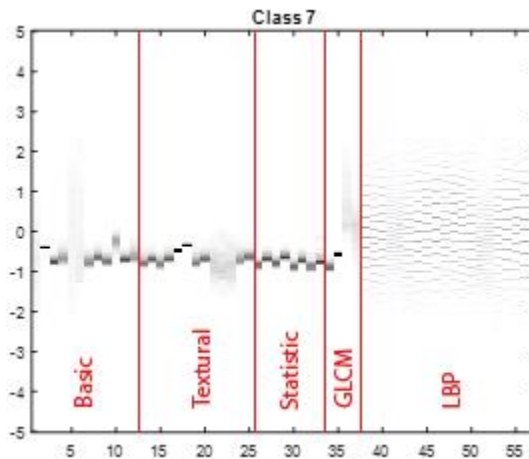


Figure 39 Feature fingerprint for *yellow water lily* (class 7)

Although precise visualization of high-dimensional data is impossible in 2 or 3 dimensions, some sort of rough estimation, in this case using feature fingerprints, can be made to get an overall picture. By visually inspecting the fingerprints, some clear and notable differences between classes can be noted. However, this method is only 2D approximation of 57 dimensional feature space, thus final conclusions cannot be made about the data structure and distribution in the feature space.

Water (class 4) and yellow water lily (class 7) are easy to recognize, since their radar echoes are very strong and distinct; the values in the fingerprints are highly concentrated. Yellow water lily is a plant growing in shallow waters, with its leaves floating on the water surface, thus causing its fingerprint to be very similar to water. Because of floating leaves of yellow water lily, its scattering properties are slightly different compared to water, and due to that, more variance is involved in the fingerprint. Sedge

(class 6) shares also similarities with water and yellow water lily. At Lake Poosjärvi, sedge is growing mainly in moist soil and shallow water, causing its fingerprint to contain similar patterns as water and yellow water lily. Among these 3 classes, the sedge class represents the highest variance. Notable differences between fingerprints of water, yellow water lily and sedge occur in GLCM and Textural feature groups. From classification viewpoint, presence of water seems to be an important characteristic for separating yellow water lily and sedge from other classes. Although reed (class 1), water horsetail (class 2) and reed in water (class 3) are also wetland plants, other characteristics than the presence of water are dominating their radar echoes.

Other two classes having similar fingerprints are reed (class 1) and trees (class 5). Similarities might be caused by thick leaf coverage present in both of the cases; large reed areas and forests are inducing similar radar echoes. Due to that, all classification algorithms occasionally mixed between reed and trees. Reed in water (class 3) has also similarities with reed and trees; Basic, Textural and GLCM feature groups appear to be similar whereas Statistical and LBP groups are different.

The fingerprint of water horsetail (class 2) is unique compared to other classes. Areas dominated by water horsetail resemble large fields of thin, upward pointing tubes rising from water. This physical property causes the radar pulse to backscatter from water horsetail in a very special way.

Lastly, it should be noted that the analysis represented above is suggestive based on visual comparison of fingerprints and on the knowledge gathered during the field studies at Lake Poosjärvi. More sophisticated methods than visual inspection of fingerprints should be employed in order to get more comprehensive view on the issue. For example, visual comparison of the LBP features between the classes is difficult and reliable conclusions cannot be made. Still, the LBP feature group was one of the best performing feature groups alone.

4.4 Comparison of Classification Techniques

Four different classification algorithms were tested: Support Vector Machine (SVM), Random Forest (RF), Multilayer perceptron neural network ensemble (MLP ensemble) and K-Nearest Neighbor (KNN). The KNN classifier was included despite the well-known fact, that KNN is not famous for its robustness, especially when operating with high-dimensional data. KNN was included for pure curiosity; to see how it manages against more sophisticated classification algorithms.

Performance of the classifiers was evaluated using 10-fold cross-validation. Average classification accuracies and corresponding standard deviations were calculated between the folds. In addition to numerical evaluation, visual inspection was used to get better view about the generalization performance of the classifiers. Visual inspection was con-

ducted by comparing classified SAR-images with aerial photographs taken from the area to recognize possible flaws in classification. Visual inspection was necessary, since the correlation between test and training sets caused over-optimistic classification accuracies. For example, KNN classifiers showed very high accuracies of 0.95-0.98, but visual inspection revealed relatively poor generalization performance for all KNNs. Correlation between the training and test sets was caused by the fact that both of the sets were sampled from the same ground truth data areas, i.e. ground truth data areas for training and test sets were not spatially separated enough. This caused classifiers to classify the points inside the ground truth data areas almost perfectly leading to very high classification accuracies. Spatially separated areas for test and training sets would have been required in order to get better view on the generalization performance of the classifiers. Unfortunately, collection of ground truth data is expensive and time consuming, and not enough time was available to build perfect training and test sets.

The training set included 2500 points (1491 for reed in water) and the test set 250 points for each class. The confusion matrix and statistics were calculated for each classifier for the best performing feature group combination. Input data for the SVM and KNN classifiers were normalized before training and classification. Normalization of the data was conducted by subtracting the mean and dividing by the standard deviation. All classified SAR-images were post-processed using mode filter in 5x5 sliding window.

The importance of different feature groups was studied by calculating classification accuracies and standard deviations (10-fold cross validation) for all combinations of all the 5 feature groups. These 5 feature groups were used, because computation of classification accuracies for all combinations of the 57 individual features would have been too heavy.

Tables for classification accuracies and standard deviations for tested classifiers are represented in the following sections. In these tables, number 1 indicates that the particular feature group was included in a combination. Bolded font indicates the highest accuracy among combinations including some certain number of feature groups. Bold italics indicates the overall highest accuracy for the certain classifier. If more than one combination achieved the same highest accuracy value, the combination having the lowest standard deviation was selected.

4.4.1 Random Forest

The random forest classifier was implemented using the R randomForest package through the Matlab interface. The number of trees in the forest was 750, and in each tree, 70% subspace sampling was used. Bootstrap samples were drawn without replacement and the majority voting scheme was used to combine predictions of individual decision trees.

Classification accuracies and standard deviations for the random forest classifier using different feature group combinations are shown in Table 5.

Table 5 Classification accuracies and standard deviations for different feature group combinations for the random forest classifier

Basic (1)	Text (2)	Stat (3)	GLCM (4)	LBP (5)	Acc.	σ
1	0	0	0	0	0,671	0,0167
0	1	0	0	0	0,825	0,0206
0	0	1	0	0	0,954	0,0066
0	0	0	1	0	0,697	0,0134
0	0	0	0	1	0,809	0,0189
1	1	0	0	0	0,849	0,0143
1	0	1	0	0	0,943	0,0084
1	0	0	1	0	0,817	0,0206
1	0	0	0	1	0,868	0,0177
0	1	1	0	0	0,958	0,0111
0	1	0	1	0	0,896	0,0200
0	1	0	0	1	0,927	0,0183
0	0	1	1	0	0,951	0,0079
0	0	1	0	1	0,962	0,0055
0	0	0	1	1	0,920	0,0104
1	1	1	0	0	0,958	0,0055
1	1	0	1	0	0,904	0,0185
1	1	0	0	1	0,926	0,0144
1	0	1	1	0	0,943	0,0083
1	0	1	0	1	0,962	0,0072
1	0	0	1	1	0,921	0,0121
0	1	1	1	0	0,958	0,0111
0	1	1	0	1	0,972	0,0082
0	1	0	1	1	0,950	0,0168
0	0	1	1	1	0,962	0,0051
1	1	1	1	0	0,960	0,0058
1	1	1	0	1	0,971	0,0038
1	1	0	1	1	0,951	0,0102
1	0	1	1	1	0,962	0,0074
0	1	1	1	1	0,972	0,0084
1	1	1	1	1	0,972	0,0044

The highest classification accuracy of 0.972 was obtained by using all feature groups (1,2,3,4,5). By using only a single feature group, Statistical (3) features gave the highest accuracy of 0.954. Among the combinations including two feature groups, the highest accuracy of 0.962 was achieved using Statistical (3) and LBP (5) features. Among the combinations including three feature groups, the highest accuracy of 0.972, which was

already the overall highest accuracy, was achieved using Textural (2), Statistical (3) and LBP (5) features. By adding more groups, classification accuracy did not improve from this value, and the only effect was that Basic (1) feature group halved the standard deviation. The effect of GLCM features was neutral.

The confusion matrix and statistics for the feature set (1,2,3,4,5) can be seen from Appendix 1. The classified SAR image using the random forest classifier is shown in Figure 40.

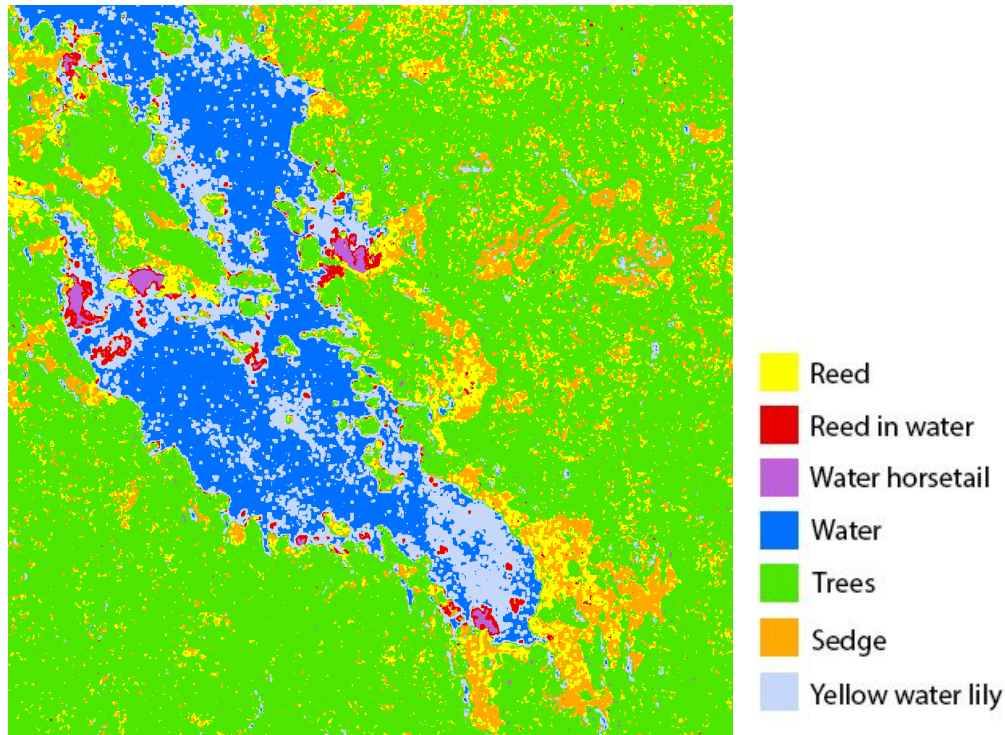


Figure 40 Classified SAR image using the random forest classifier with the full feature set (1,2,3,4,5)

4.4.2 Support Vector Machine

Two SVMs with different kernels were tested; one with radial basis function (RBF) kernel and one with cubic polynomial kernel. RBF kernel SVM was implemented using the libSVM library through Matlab interface. Cubic polynomial kernel SVM utilized Matlab's built-in SVM toolbox. Equations for the kernel functions are:

- radial basis function kernel [26]:

$$K(\mathbf{x}, \mathbf{y}) = \exp(-\gamma \|\mathbf{x} - \mathbf{y}\|^2), \gamma > 0 \quad (31)$$

- cubic polynomial kernel [27]:

$$K(\mathbf{x}, \mathbf{y}) = (1 + \mathbf{x}^T \mathbf{y})^3 \quad (32)$$

Since SVM is originally a binary classifier, it was converted to handle multiclass cases using one-vs-one voting approach. Optimal values for the RBF kernel parameter γ and soft margin control parameter C were obtained by grid search.

SVM classification accuracies for different feature group combinations are presented in Table 6.

Table 6 Classification accuracies and standard deviations for the support vector machine classifiers

Basic(1)	Text (2)	Stat (3)	GLCM(4)	LBP (5)	Cubic kernel		RBF kernel	
					Acc.	σ	Acc.	σ
1	0	0	0	0	0,631	0,0216	0,629	0,0158
0	1	0	0	0	0,801	0,0172	0,788	0,0196
0	0	1	0	0	0,810	0,0230	0,790	0,0094
0	0	0	1	0	0,538	0,0382	0,656	0,0062
0	0	0	0	1	0,851	0,0138	0,808	0,0255
1	1	0	0	0	0,829	0,0153	0,799	0,0188
1	0	1	0	0	0,840	0,0157	0,791	0,0180
1	0	0	1	0	0,776	0,0181	0,752	0,0175
1	0	0	0	1	0,913	0,0097	0,876	0,0183
0	1	1	0	0	0,924	0,0197	0,875	0,0248
0	1	0	1	0	0,880	0,0192	0,840	0,0200
0	1	0	0	1	0,961	0,0042	0,936	0,0134
0	0	1	1	0	0,867	0,0174	0,798	0,0149
0	0	1	0	1	0,956	0,0078	0,933	0,0115
0	0	0	1	1	0,945	0,0053	0,925	0,0080
1	1	1	0	0	0,919	0,0130	0,878	0,0164
1	1	0	1	0	0,891	0,0120	0,846	0,0177
1	1	0	0	1	0,964	0,0051	0,934	0,0132
1	0	1	1	0	0,854	0,0161	0,803	0,0151
1	0	1	0	1	0,955	0,0046	0,928	0,0075
1	0	0	1	1	0,949	0,0057	0,919	0,0083
0	1	1	1	0	0,936	0,0193	0,881	0,0206
0	1	1	0	1	0,982	0,0024	0,960	0,0079
0	1	0	1	1	0,975	0,0028	0,951	0,0088
0	0	1	1	1	0,962	0,0051	0,934	0,0076
1	1	1	1	0	0,928	0,0113	0,884	0,0168
1	1	1	0	1	0,982	0,0031	0,959	0,0055
1	1	0	1	1	0,975	0,0038	0,947	0,0089
1	0	1	1	1	0,959	0,0055	0,928	0,0064
0	1	1	1	1	0,984	0,0026	0,961	0,0082
1	1	1	1	1	0,983	0,0031	0,960	0,0058

The highest classification accuracies of 0.984 and 0.961 for cubic kernel and RBF kernel SVMs, respectively, were achieved by using feature groups 2, 3, 4 and 5. The best single feature group was LBP (5) having accuracy of 0.851 for cubic kernel and 0.808 for RBF kernel. Textural (2) and LBP (5) features had the best performance among the combinations including two feature groups, having accuracies of 0.961 and 0.936 for cubic and RBF kernel SVMs, respectively. Textural (2), Statistical (3) and LBP (5) features performed the most successfully within the combinations including three feature groups, having accuracies of 0.982 for cubic and 0.960 for RBF kernel. By adding more features into this combination, no significant improvements in classification accuracy or standard deviation was achieved. The influence of adding Basic (1) and GLCM (4) features was almost neutral. In all cases, performance of the cubic kernel SVM was slightly better than the RBF kernel SVM. However, the classified SAR images did not reveal any relevant differences between these two classifiers.

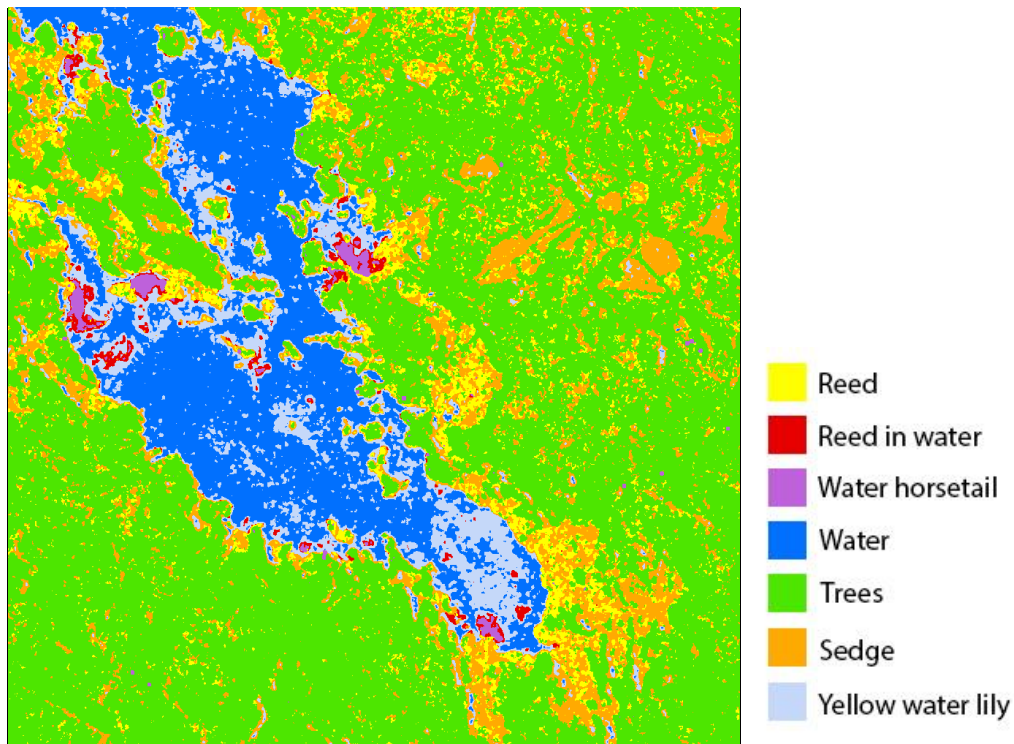


Figure 41 Classified SAR image using RBF kernel SVM and the feature set (2,3,4,5)

Confusion matrices and statistics for the SVMs using the feature set (2,3,4,5) can be seen from Appendix 2. SAR image classification results for cubic and RBF kernel SVMs are shown in Figure 41 and 49, respectively.

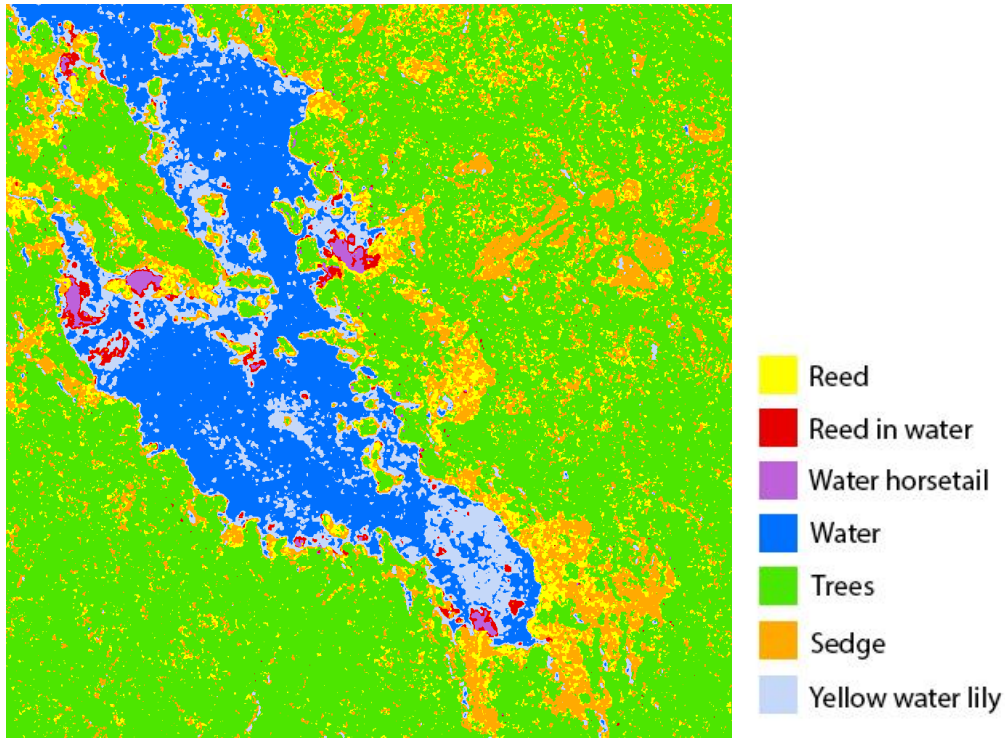


Figure 42 Classified SAR image using cubic kernel SVM and the feature set (2,3,4,5)

4.4.3 Multilayer Perceptron Neural Network Ensemble

The multilayer perceptron (MLP) neural network (NN) ensemble classifier was implemented using Matlab's Neural Network Toolbox and applying the bagging ensemble technique (see section 3.6). The ensemble consisted of 50 fully connected multilayer perceptrons as base classifiers. Each neural network was trained using a bootstrap sample (bagging) drawn uniformly with replacement from the original training data set. Predictions of multiple base classifiers were combined using the majority voting scheme. Training of networks was conducted using the scaled conjugate gradient back-propagation algorithm.

All base classifiers shared the same architecture; 30 neurons in the first hidden layer and 37 neurons in the second. This architecture was defined by applying brute force search over possible hidden layer configurations and the best performing configuration was selected. The activation function used in hidden layer neurons was tansig (hyperbolic tangent sigmoid transfer function) [28]:

$$\text{tansig}(x) = \frac{2}{1 + e^{-2x}} - 1 \quad (33)$$

and in output neurons softmax (soft max transfer function) [29]:

$$\text{softmax}(x)_j = \frac{e^{x_j}}{\sum_{k=1}^K e^{x_k}}, \quad \text{for } j = 1, \dots, K \quad (33)$$

where K is the number of dimensions in the input vector x .

Classification accuracies of the NN ensemble using different feature group combinations are presented in Table 7.

Table 7 Classification accuracies and standard deviations for different feature group combinations using the NN ensemble classifier

Basic (1)	Text (2)	Stat (3)	GLCM (4)	LBP (5)	Acc.	σ
1	0	0	0	0	0,595	0,0125
0	1	0	0	0	0,807	0,0170
0	0	1	0	0	0,825	0,0154
0	0	0	1	0	0,656	0,0099
0	0	0	0	1	0,857	0,0144
1	1	0	0	0	0,841	0,0136
1	0	1	0	0	0,830	0,0101
1	0	0	1	0	0,788	0,0164
1	0	0	0	1	0,884	0,0087
0	1	1	0	0	0,923	0,0085
0	1	0	1	0	0,894	0,0136
0	1	0	0	1	0,965	0,0053
0	0	1	1	0	0,867	0,0122
0	0	1	0	1	0,959	0,0074
0	0	0	1	1	0,950	0,0060
1	1	1	0	0	0,925	0,0089
1	1	0	1	0	0,900	0,0093
1	1	0	0	1	0,963	0,0077
1	0	1	1	0	0,860	0,0099
1	0	1	0	1	0,955	0,0086
1	0	0	1	1	0,948	0,0089
0	1	1	1	0	0,935	0,0063
0	1	1	0	1	0,980	0,0046
0	1	0	1	1	0,976	0,0021
0	0	1	1	1	0,962	0,0080
1	1	1	1	0	0,932	0,0069
1	1	1	0	1	0,977	0,0074
1	1	0	1	1	0,976	0,0065
1	0	1	1	1	0,957	0,0106
0	1	1	1	1	0,980	0,0041
1	1	1	1	1	0,980	0,0050

The best result of 0.980 was achieved by using feature groups 2, 3, 4 and 5. The LBP (5) was the best single feature group having accuracy of 0.857. Textural (2) and LBP (5) features had the best performance among the combinations including two feature groups, having accuracy of 0.965. Textural (2), Statistical (3) and LBP (5) features per-

formed the most successfully among the combinations including three feature groups having the accuracy of 0.980. By adding Basic (1) and/or GLCM (4) features to this combination, no significant improvement in accuracy or standard deviation were achieved.

Confusion matrix and statistics for the feature set (2,3,4,5) can be seen from Appendix 3. SAR image classification result for the MLP ensemble is shown in Figure 43.

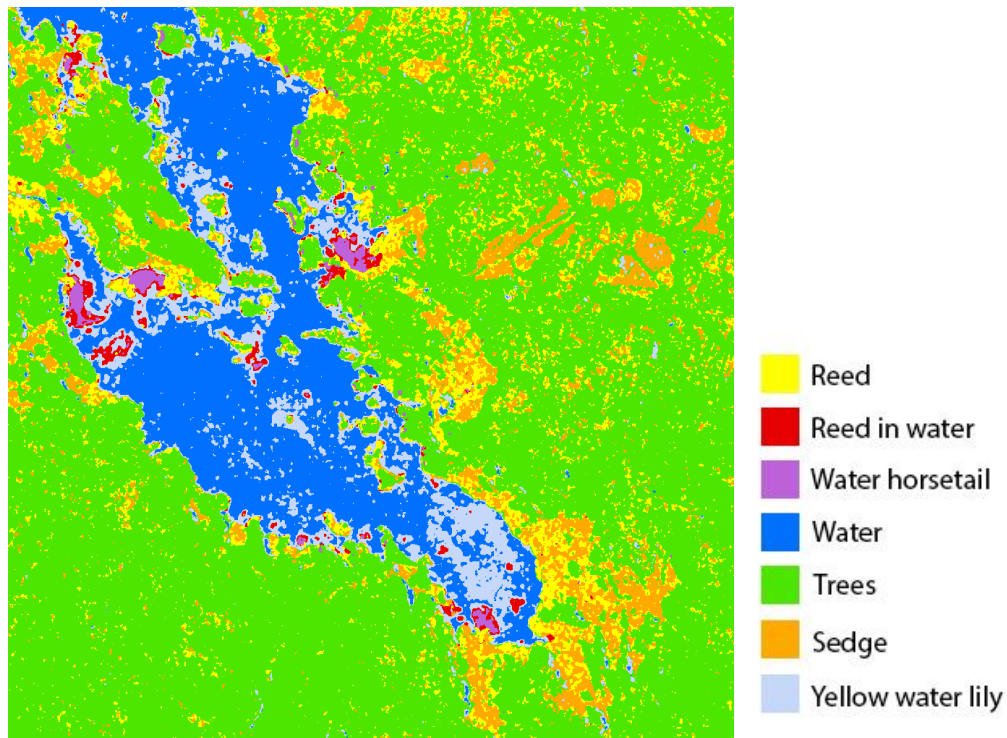


Figure 43 Classified SAR image using the MLP ensemble and the feature set (2,3,4,5)

By visually comparing the classified SAR images, it can be noted that the performance of the MLP ensemble is the best among all classifiers. This can be seen especially in the areas of water and trees; the least mixing is occurring when using the MLP ensemble. Ordinary single MLP neural networks were also tested, but classification results were significantly lower than using the MLP ensemble.

4.4.4 K-Nearest Neighbor

KNN classifiers were implemented using Matlab's built-in KNN toolbox. Three KNN classifiers with k values of 1, 3 and 6 were tested. All three KNN classifiers used the same city block (Manhattan) distance metric [30]:

$$D(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^N |x_i - y_i| \quad (34)$$

Since the original KNN implementation is suffering from long search times during classification, training examples were organized into kd-tree to ease the search.

Classification accuracies and standard deviations for the KNN classifiers with different k values are shown in Table 8.

Table 8 Classification accuracies and standard deviations for the KNN classifiers

Basic (1)	Text (2)	Stat (3)	GLCM (4)	LBP (5)	k = 1		k = 3		k = 6	
					Acc.	σ	Acc.	σ	Acc.	σ
1	0	0	0	0	0,584	0,0073	0,580	0,0106	0,590	0,0129
0	1	0	0	0	0,799	0,0116	0,794	0,0165	0,794	0,0186
0	0	1	0	0	0,941	0,0090	0,917	0,0099	0,901	0,0096
0	0	0	1	0	0,642	0,0146	0,653	0,0157	0,658	0,0153
0	0	0	0	1	0,916	0,0065	0,886	0,0091	0,862	0,0084
1	1	0	0	0	0,803	0,0158	0,790	0,0148	0,780	0,0140
1	0	1	0	0	0,843	0,0108	0,824	0,0121	0,808	0,0178
1	0	0	1	0	0,762	0,0126	0,751	0,0177	0,749	0,0125
1	0	0	0	1	0,937	0,0089	0,914	0,0083	0,891	0,0124
0	1	1	0	0	0,944	0,0108	0,929	0,0159	0,915	0,0169
0	1	0	1	0	0,868	0,0136	0,858	0,0215	0,848	0,0242
0	1	0	0	1	0,966	0,0073	0,953	0,0083	0,932	0,0123
0	0	1	1	0	0,933	0,0059	0,911	0,0093	0,893	0,0098
0	0	1	0	1	0,969	0,0052	0,955	0,0050	0,937	0,0095
0	0	0	1	1	0,958	0,0046	0,941	0,0041	0,923	0,0077
1	1	1	0	0	0,899	0,0127	0,881	0,0131	0,870	0,0191
1	1	0	1	0	0,853	0,0135	0,832	0,0167	0,826	0,0151
1	1	0	0	1	0,959	0,0035	0,944	0,0086	0,923	0,0117
1	0	1	1	0	0,866	0,0065	0,847	0,0111	0,839	0,0129
1	0	1	0	1	0,964	0,0042	0,947	0,0052	0,932	0,0030
1	0	0	1	1	0,956	0,0043	0,939	0,0047	0,921	0,0064
0	1	1	1	0	0,947	0,0079	0,933	0,0120	0,917	0,0155
0	1	1	0	1	0,980	0,0037	0,969	0,0038	0,953	0,0055
0	1	0	1	1	0,973	0,0047	0,962	0,0053	0,944	0,0084
0	0	1	1	1	0,975	0,0036	0,962	0,0052	0,947	0,0078
1	1	1	1	0	0,910	0,0090	0,890	0,0074	0,878	0,0153
1	1	1	0	1	0,971	0,0029	0,960	0,0067	0,945	0,0078
1	1	0	1	1	0,966	0,0035	0,953	0,0075	0,933	0,0100
1	0	1	1	1	0,971	0,0029	0,955	0,0041	0,941	0,0034
0	1	1	1	1	0,982	0,0027	0,972	0,0036	0,956	0,0063
1	1	1	1	1	0,974	0,0039	0,962	0,0057	0,947	0,0074

The highest classification accuracies of 0.982, 0.972, and 0.956 for k values 1, 3 and 6, respectively, were achieved with feature groups 2, 3, 4 and 5. The Statistical (3) feature

group was the best single group having accuracies of 0.941, 0.917 and 0.901 for k values 1,3 and 6, respectively. Statistical (3) and LBP (5) features had the best performance among the combinations including two feature groups, having accuracies of 0.969, 0.955 and 0.937 ($k=1,3,6$). Textural (2), Statistical (3) and LBP (5) features produced the best results among the combinations including three feature groups, having accuracies of 0.980, 0.969 and 0.953 ($k=1,3,6$). By using these three feature groups, the classification accuracies were almost as high as the overall highest values, and only extremely small improvements were provided by adding more feature groups into this combination. The effect resulted by adding the GLCM (4) features was neutral and the effect of the Basic (1) features was actually slightly negative.

Confusion matrix and statistics for the KNN classifiers using the feature set (2,3,4,5) can be seen from Appendix 4. SAR image classification result for the KNN ($k=1$) classifier using the feature set (2,3,4,5) is shown in Figure 44.

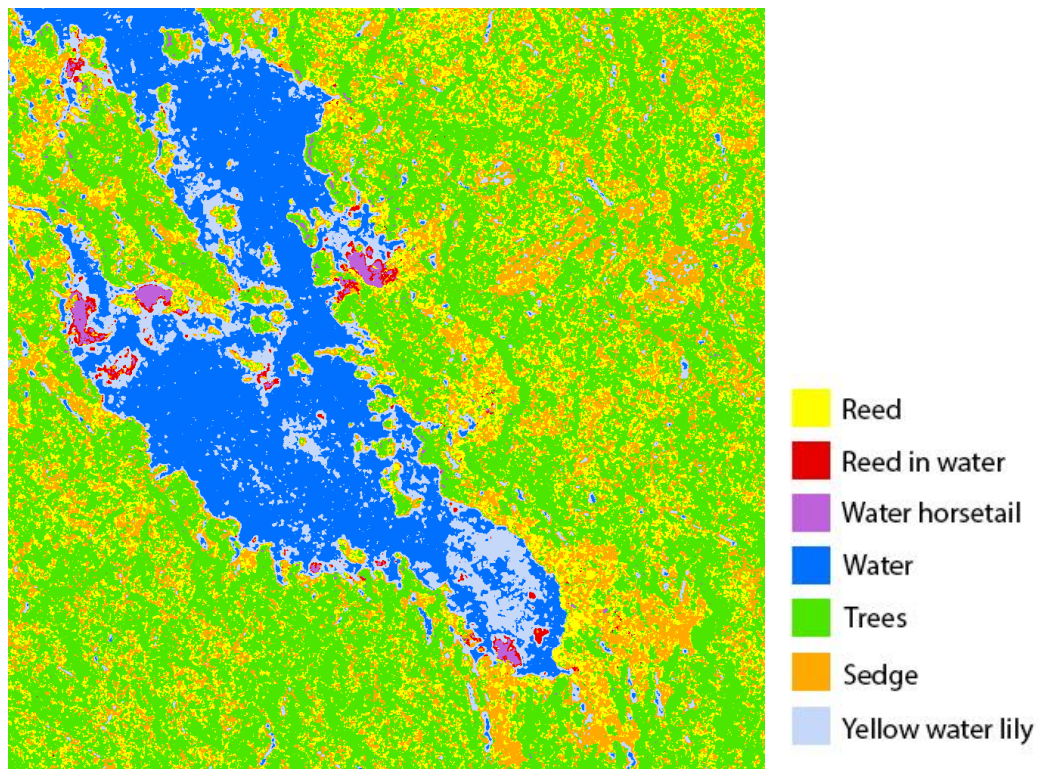


Figure 44 Classified SAR image using the best KNN classifier ($k=1$) and the feature set (2,3,4,5)

Although the KNN classifiers produced very high classification accuracies, their generalization performance was very low. This can be seen clearly from Figure 44, in which a lot of mixing occurs especially in the forest areas.

5. CONCLUSIONS

Selection of good and reasonable feature set had very strong influence on the performance of the classifiers. Since only HH and VV polarizations were available (no cross-polarizations), polarimetric methods could not be utilized. It was found that the importance of textural features was emphasized in this case. Experiments showed that among all the tested classifiers the most important feature groups were Textural, Statistical and LBP groups. Despite the name of Statistical feature group, it is also measuring textural information since the statistics are computed in 15x15 sliding window. The least important feature groups were Basic and GLCM, which had only very minor impact on classification performance. It was also discovered that larger sliding windows (15x15) resulted better performance than smaller ones (7x7, 9x9).

All the tested classifiers resulted very high classification accuracies for the test data. This was caused by the fact that the test and training sets were not enough spatially separated, which caused correlation between the sets. In other words, the test and training sets were sampled from regions not distant enough from each other. More spatially separated areas for test and training sets would have been required in order to get better view on the generalization performance of the classifiers. Unfortunately, collection of ground truth data is expensive and time consuming, and not enough time was available to build complete training and test sets.

Since all classifiers resulted almost equally high classification accuracies, discrimination between them was made by visually comparing the classification results. The best performing classifier was the MLP ensemble with the classification accuracy of 0.980. Although cubic kernel SVM and KNN ($k=1$) classifiers resulted almost equal classification accuracies (0.984 and 0.982), it could be clearly seen from the classified SAR-images that the least mixing occurs with the MLP ensemble. Classification results of the cubic kernel SVM, RBF kernel SVM and RF were almost identical and no notable differences could be observed. Although the best results were obtained with the MLP ensemble, it was also the most complex classifier, and a lot of trials and tweaking was required in order to get it function properly. If the same amount of time had been invested into RF or SVM classifiers, equally good results might have been achieved. The worst performing classifier was KNN. Although KNN classifiers resulted in high classification accuracies, their generalization capability was very low. This can be clearly seen from the SAR image classified using the KNN; a lot of mixing is occurs especially in the areas covered by trees. In general, all classifiers were mixing more or less between the reed (class 1), trees (class 5) and sedge (class 6).

Although the test data set used in the experiments was not complete, the results showed that the methods used in the thesis are feasible for classification of vegetation from the spaceborne SAR images. Especially the RF classifier is promising since it is computationally less demanding than the SVM or MLP ensemble. However, the best results were obtained with the MLP ensemble, thus more studies are required in order to reveal the real potential of the RF. Another issue which requires further studies is whether the classes are detected based on the properties of the vegetation, soil or ground cover.

BIBLIOGRAPHY

- [1] Wikipedia, Remote Sensing History, 2016. Available: http://en.wikipedia.org/wiki/Remote_sensing#History. [Accessed: 14-Apr-2016].
- [2] C. R. Jackson and J. R. Apel, Synthetic Aperture Radar Marine User's Manual, U.S. Department of Commerce, 2004.
- [3] Wikipedia, Seasat. Available: <http://en.wikipedia.org/wiki/Seasat>. [Accessed: 14-Apr-2016].
- [4] Y. K. Chan and V. C. Koo, An Introduction to Synthetic Aperture Radar (SAR), Prog. Electromagn. Res. B, vol. 2, no. 6, pp. 27–60, 2008.
- [5] A. Moreira, P. Prats, M. Younis, G. Krieger, I. Hajnsek, and K. Papathanassiou, A Tutorial on Synthetic Aperture Radar, IEEE Geosci. Remote Sens. Mag., no. March, pp. 1–43, 2013.
- [6] Deutschen Zentrums für Luft- und Raumfahrt, TerraSAR-X Brochure, 2009. Available: http://www.dlr.de/eo/Portaldata/64/Resources/dokumente/TSX_brosch.pdf, [Accessed: 29-Mar-2016].
- [7] C. Wolff, Radar Tutorial. Available: <http://www.radartutorial.eu/>, [Accessed: 26-Apr-2016].
- [8] J.-S. Lee and E. Pottier, Polarimetric Radar Imaging, CRC Press, 2009.
- [9] AIRBUS Defence & Space, Radiometric Calibration of TerraSAR-X Data. Available: http://www2.geo-airbusds.com/files/pmedia/public/r465_9_tsx-x-itd-tn-0049-radiometric_calculations_i3.00.pdf, [Accessed: 23-May-2016].
- [10] Deutschen Zentrums für Luft- und Raumfahrt, DLR Portal. Available: <http://www.dlr.de/>. [Accessed: 25-May-2016].
- [11] Wikipedia, Artificial Intelligence, 2015. Available: http://en.wikipedia.org/wiki/Artificial_intelligence. [Accessed: 20-Aug-2015].
- [12] Wikipedia, Machine Learning, 2015. Available: http://en.wikipedia.org/wiki/Machine_learning. [Accessed: 20-Aug-2015].
- [13] A. R. Webb and K. D. Copsey, Statistical Pattern Recognition, John Wiley & Sons, Ltd., 2011.
- [14] C. M. Bishop, Pattern Recognition and Machine Learning, Springer Science+Business Media, LLC, 2006.
- [15] T. M. Mitchell, Machine Learning, McGraw-Hill Science/Engineering/Math, 1997.
- [16] P. Flach, Machine Learning: The Art and Science of Algorithms that Make Sense of Data, Cambridge University Press, 2012.
- [17] B. Tso and P. M. Mather, Classification Methods for Remotely Sensed Data, CRC Press, 2009.
- [18] S. Haykin, Neural Networks - A Comprehensive Foundation, Prentice Hall International, Inc, 1999.
- [19] T. Kumpumäki and T. Lipping, Transformation and Texture Based Features in TerraSAR-X Data Classification for Environmental Monitoring, Proc IEEE International Geoscience and Remote Sensing Symposium, IGARS2015, Milan,

- Italy, July 26-31, pp. 3278-3281, 2015.
- [20] Ympäristöhallinto, Natura 2000 -alueet, Poosjärvi. Available: [http://www.ymparisto.fi/fi-FI/Luonto/Suojelualueet/Natura_2000_alueet/Poosjarvi\(5451\)](http://www.ymparisto.fi/fi-FI/Luonto/Suojelualueet/Natura_2000_alueet/Poosjarvi(5451)). [Accessed: 26-May-2016].
- [21] Mathworks, Matlab Entropy Function Documentation, 2015, Available: <http://se.mathworks.com/help/images/ref/entropy.html>. [Accessed: 10-Aug-2015].
- [22] R. M. Haralick, K. Shanmugam, and I. Dinstein, Textural Features for Image Classification, *IEEE Trans. Syst. Man Cybern.*, vol. 3, pp. 610–620, 1973.
- [23] M. Hall-Beyer, GLCM Texture Tutorial, University of Calgary, 2015. Available: http://www.fp.ucalgary.ca/mhallbey/glcm_mean.htm. [Accessed: 10-Aug-2015].
- [24] Mathworks, Matlab Graycoprops Function Documentation, 2015. Available: <http://se.mathworks.com/help/images/ref/graycoprops.html>. [Accessed: 10-Aug-2015].
- [25] G. Zhao and T. Ahonen, Rotation Invariant Image and Video Description with Local Binary Pattern Features, *IEEE Trans. Image Process.*, vol. 21, no. 4, pp. 1–13, 2010.
- [26] C.-W. Hsu, C.-J. Lin, and C.-C. Chang, A Practical Guide to Support Vector Classification, *BJU Int.*, vol. 101, no. 1, pp. 1396–400, 2008.
- [27] Mathworks, Support Vector Machines Documentation, 2015.
- [28] Mathworks, Tansig Function Documentation. Available: <http://se.mathworks.com/help/nnet/ref/tansig.html>. [Accessed: 11-Aug-2016].
- [29] Mathworks, Softmax Function Documentation. Available: <http://se.mathworks.com/help/nnet/ref/softmax.html>. [Accessed: 11-Aug-2016].
- [30] Mathworks, Classification Using Nearest Neighbor Documentation, 2015. Available: <http://se.mathworks.com/help/stats/classification-using-nearest-neighbors.html>. [Accessed: 13-Aug-2015].

APPENDIX 1: RANDOM FOREST CLASSIFIER CONFUSION MATRIX AND STATISTICAL MEASURES FOR THE TEST DATA

Table 9 Confusion matrix for the RF classifier using the feature group 1,2,3,4,5 (accuracy of 0.979)

Predicted classes	Actual classes						
	1	2	3	4	5	6	7
1	232	0	1	0	3	11	0
2	2	250	0	0	0	0	0
3	0	0	249	0	0	0	0
4	0	0	0	250	0	0	1
5	4	0	2	0	246	1	0
6	14	0	1	0	1	238	0
7	0	0	0	0	0	0	249

Table 10 Statistical measures for the confusion matrix

	Classes						
	1	2	3	4	5	6	7
True Pos	232	250	249	250	246	238	249
False Pos	14	0	0	1	5	16	0
False Neg	18	0	1	0	4	12	1
True Neg	1486	1500	1500	1499	1495	1484	1500
Precision	0.943	1.000	1.000	0.996	0.980	0.937	1.000
Sensitivity	0.928	1.000	0.996	1.000	0.984	0.952	0.996
Specificity	0.991	1.000	1.000	0.999	0.997	0.989	1.000

APPENDIX 2: SVM CLASSIFIER CONFUSION MATRICES AND STATISTICAL MEASURES FOR THE TEST DATA

Table 11 Confusion matrix for the RBF SVM using the feature group 2,3,4,5 (accuracy 0.963)

Predicted classes	Actual classes						
	1	2	3	4	5	6	7
1	228	0	2	0	12	24	0
2	0	250	0	0	0	0	0
3	0	0	250	0	0	0	0
4	0	0	0	250	0	0	0
5	1	0	0	0	232	1	0
6	21	0	0	0	6	225	0
7	0	0	0	0	0	0	250

Table 12 Statistical measures for the RBF SVM confusion matrix

	Classes						
	1	2	3	4	5	6	7
True Pos	228	250	250	250	232	225	250
False Pos	36	0	0	0	2	27	0
False Neg	22	0	0	0	18	25	0
True Neg	1464	1500	1500	1500	1498	1473	1500
Precision	0.864	1.000	1.000	1.000	0.991	0.893	1.000
Sensitivity	0.912	1.000	1.000	1.000	0.928	0.900	1.000
Specificity	0.976	1.000	1.000	1.000	0.999	0.982	1.000

Table 13 Confusion matrix for the cubic SVM using the feature group 2,3,4,5 (accuracy 0.986)

Predicted classes	Actual classes						
	1	2	3	4	5	6	7
1	244	0	0	0	8	7	0
2	0	250	0	0	0	0	0
3	0	0	249	0	0	0	0
4	0	0	0	250	0	0	0
5	1	0	0	0	240	1	0
6	5	0	1	0	2	242	0
7	0	0	0	0	0	0	250

Table 14 Statistical measures for the cubic SCM confusion matrix

	Classes						
	1	2	3	4	5	6	7
True Pos	244	250	249	250	240	242	250
False Pos	15	0	0	0	2	8	0
False Neg	6	0	1	0	10	8	0
True Neg	1485	1500	1500	1500	1498	1492	1500
Precision	0.942	1.000	1.000	1.000	0.992	0.968	1.000
Sensitivity	0.940	1.000	0.996	1.000	0.840	0.916	1.000
Specificity	0.990	1.000	1.000	1.000	0.999	0.995	1.000

APPENDIX 3: MLP ENSEMBLE CLASSIFIER CONFUSION MATRIX AND STATISTICAL MEASURES FOR THE TEST DATA

Table 15 Confusion matrix for the MLP ensemble using the feature group 2,3,4,5 (accuracy 0.969)

Predicted classes	Actual classes						
	1	2	3	4	5	6	7
1	229	0	1	0	8	19	0
2	0	250	1	0	0	0	0
3	0	0	246	0	2	0	0
4	0	0	0	250	0	0	0
5	3	0	0	0	239	0	0
6	18	0	1	0	3	231	0
7	0	0	1	0	0	0	250

Table 16 Statistical measures for MLP ensemble confusion matrix

	Classes						
	1	2	3	4	5	6	7
True Pos	229	250	246	250	239	231	250
False Pos	28	1	0	0	3	22	1
False Neg	21	0	4	0	11	19	0
True Neg	1472	1499	1500	1500	1497	1478	1499
Precision	0.891	0.996	1.000	1.000	0.988	0.913	0.996
Sensitivity	0.916	1.000	0.984	1.000	0.956	0.924	1.000
Specificity	0.981	0.999	1.000	1.000	0.998	0.985	0.999

APPENDIX 4: KNN CLASSIFIER CONFUSION MATRICES AND STATISTICAL MEASURES FOR THE TEST DATA

Table 17 Confusion matrix for the KNN ($k=1$) using the feature group 2,3,4,5 (accuracy 0.985)

Predicted classes	Actual classes						
	1	2	3	4	5	6	7
1	250	0	0	0	12	4	0
2	0	250	0	0	0	0	0
3	0	0	250	0	1	0	0
4	0	0	0	250	0	0	0
5	0	0	0	0	229	1	0
6	0	0	0	0	8	245	0
7	0	0	0	0	0	0	250

Table 18 Statistical measures for the KNN ($k=1$) confusion matrix

	Classes						
True Pos	250	250	250	250	229	245	250
False Pos	16	0	1	0	1	8	0
False Neg	0	0	0	0	21	5	0
True Neg	1484	1500	1499	1500	1499	1492	1500
Precision	0.940	1.000	0.996	1.000	0.996	0.968	1.000
Sensitivity	1.000	1.000	1.000	1.000	0.916	0.980	1.000
Specificity	0.989	1.000	0.999	1.000	0.999	0.995	1.000

Table 19 Confusion matrix for the KNN ($k=3$) using the feature group 2,3,4,5 (accuracy 0.973)

Predicted classes	Actual classes						
	1	2	3	4	5	6	7
1	249	0	0	0	26	10	0
2	0	250	0	0	0	0	0
3	0	0	250	0	4	0	0
4	0	0	0	250	0	0	0
5	0	0	0	0	215	1	0
6	1	0	0	0	9	239	0
7	0	0	0	0	0	0	250

Table 20 Statistical measures for the KNN ($k=3$) confusion matrix

	Classes						
True Pos	249	250	250	250	215	239	250
False Pos	36	0	0	0	1	10	0
False Neg	1	0	0	0	35	11	0
True Neg	1464	1500	1500	1500	1499	1490	1500
Precision	0.874	1.000	1.000	1.000	0.995	0.960	1.000
Sensitivity	0.996	1.000	1.000	1.000	0.860	0.956	1.000
Specificity	0.976	1.000	1.000	1.000	0.999	0.993	1.000

Table 21 Confusion matrix for the KNN ($k=6$) using the feature group 2,3,4,5 (accuracy 0.962)

Predicted classes	Actual classes						
	1	2	3	4	5	6	7
1	250	0	0	0	38	15	0
2	0	250	0	0	0	0	0
3	0	0	250	0	5	1	0
4	0	0	0	250	0	0	0
5	0	0	0	0	201	2	0
6	0	0	0	0	11	233	0
7	0	0	0	0	0	0	250

Table 22 Statistical measures for the KNN ($k=6$) confusion matrix

	Classes						
True Pos	250	250	250	250	201	233	250
False Pos	53	0	0	0	2	11	0
False Neg	0	0	0	0	49	17	0
True Neg	1447	1500	1500	1500	1498	1489	1500
Precision	0.825	1.000	1.000	1.000	0.990	0.955	1.000
Sensitivity	1.000	1.000	1.000	1.000	0.804	0.932	1.000
Specificity	0.965	1.000	1.000	1.000	0.999	0.993	1.000