DELFT UNIVERSITY OF TECHNOLOGY

MASTERS THESIS

Predicting Water Current based on Wind Data using a Neural Network.

Author:
Cor-Jan HEIJLEMA

Supervisor:
Prof. Dr. Ir. Geurt JONGBLOED
Dr. Lotte LINTMEIJER

A thesis submitted in fulfillment of the requirements for the degree of Master of Science

in the

Statistics Section
Delft Institute of Applied Mathematics

January 22, 2023

DELFT UNIVERSITY OF TECHNOLOGY

Abstract

Electrical Engineering, Mathematics and Computer Science
Delft Institute of Applied Mathematics

Master of Science

Predicting Water Current based on Wind Data using a Neural Network.

by Cor-Jan HEIJLEMA

The increased use of data and models in sports has revolutionized the way teams and athletes prepare and compete. This thesis presents a study that uses data in order to help the athletes improve their performance in the sport of sailing. This is done by analysing and utilising data, generated by sophisticated meteorological models. The data describes the dynamics of the wind and water in the designated race areas for the sailing regatta of the 2021 Tokyo Olympics. During this study we have attempted to unravel the relationship between the wind and the water such that an accurate prediction of the water current field could be made, given its corresponding wind field. By manipulating the data using principal component analysis, we were able to reduce the dimensionality of the data while still preserving most of its information. Using this dimension reduced data, we trained a multi-layer perceptron neural network to make water current predictions using wind data. The coefficient of determination of the predictions reached levels above 0.6 for some race areas. Even though the use case of this study was to predict water currents for the sailing regatta in Tokyo, the method can be used for many different applications.

Thesis Committee:

Chair:	Prof. Dr. Ir. Geurt Jongbloed
Company Supervisor:	Dr. Lotte Lintmeijer
Committee Member:	Prof. Dr. Ir. Martin van Gijzen

Contents

List of Figures

List of Tables

Chapter 1

Introduction

For achieving the best performance in sailing, the way a sailor interprets the wind and the water current is crucial. In order to be prepared in the best way to do this, sailors want to have as much information about the weather circumstances during their race as possible. This means that they would like to know the dynamics of the wind and of the current in a detailed way, so they can sail their boat in the best positions to benefit from these natural phenomena. The goal of this project is to develop a neural network, capable of predicting the near-surface water current based on wind data. More specifically we have done a case study in which we predict the water current in the race areas of the 2020 Olympic Games. This is done by training a neural network on data of which we reduced the dimensionality by applying principal component analysis.

The usage of scientific models in forecasting wind fields for sailing competitions has been going on for a long time already. During the Olympic Games of Sydney in 2000, numerical weather prediction models were used to provide weather information for the athletes and coaches as is mentioned in Spark and Connor, 2004.

In order to get a competitive advantage over other countries, the Dutch Sailing Federation committed to two things:

- Predicting the wind circumstances with a higher resolution in real-time.
- Predicting the water circumstances based on the wind information.

For this thesis, the focus was the latter of these two.
As part of the preparations for the Olympic Games, different parties have worked together to model detailed wind and current field data in Sagami Bay, based on actual wind measurements in the area. The wind and corresponding water current fields that are generated from these meteorological models form the foundation of our data-based research.

The report is organized as follows. In Chapter 2 more information is given about Sagami Bay and its complex meteorological circumstances. We also explain broadly the nuances of the used models that generated the wind and water data that is available to us, so we get a better understanding of what factors play a role in predicting the water current. We also give a simple overview of the available data itself. In Chapter 3, we start analysing the first part of the data: the wind fields. We look at whether the dimension of these wind fields can be reduced by analysing the velocity

and direction components of the wind fields and the relation between those components over the grid points considered. In Chapter 4, we take a closer look at the simulated water current fields. We want to see whether we can divide all simulated fields into a small number of regimes. We also apply principal component analysis to reduce the dimensionality of the data. Finally in Chapter 5 we train various neural networks on our reduced data to predict the water current. In Chapter 6 we discuss our findings and what steps could be considered for future developments.
In Appendix A we describe a tool that we created for the sailing federation during the 2020 Tokyo Olympic Games, that could provide insights into what current behaviour was to be expected based on the wind behaviour. Appendix B provides analysis results related to race areas that are not discussed in the report.

Chapter 2

Sagami Bay and The Data

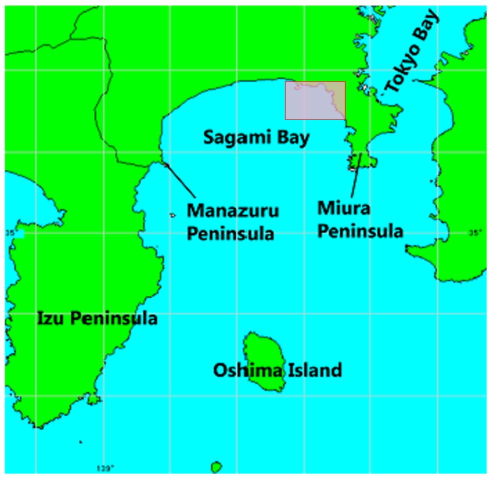
The goal of this chapter is to provide motivation and background information about the case study. In Section 2.1 we point out the meteorological complexities of Sagami Bay. In Section 2.2 the assumptions that were used for modeling the data are discussed. Section 2.3 provides a short overview of the format of the data is introduced.

2.1 Sagami Bay

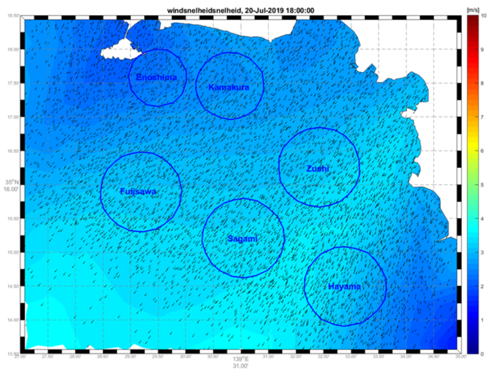
Sagami Bay is a coastal body of water, south of Tokyo. Figure 2.1a provides a simple map of the bay and its surroundings. There are several factors that cause the meteorological circumstances in the bay to be complex.
The first factor is the geology of the bay and the land surrounding it. 40 Kilometer to the west of the bay lies 3777 meter high Mount Fuji. On the west side of Sagami Bay, the Manazuru Peninsula is located. On the east side the bay is surrounded by the Miura Peninsula. South of the bay, where its entrance to the Pacific opens lies Oshima island with a mountain top of 700 meters high. These land masses interfere with the wind, creating perturbation.

Regarding the water, there are two ocean currents that come together near Sagami Bay. From the north the bay receives the Oyashio Current, while from the south the Kuroshio Current influences the flow of water as is described in Iwata and Matsuyama, 1989.

The Olympic races were held in the north-east of the bay. There were six different race areas which were all located on a relatively small area of the bay. Figure 2.1 shows where the races were held in the bay, and gives an overview of the six race areas.



(A) Sagami Bay. The pink area is the part of the bay in which the Olympic races were held.



(B) Race areas for Olympic races.

FIGURE 2.1: Grids of all race areas for the Olympic Games

2.2 Models

2.2.1 Wind Model

The wind model that produced the data used for this thesis is a complex meteorological model, called the “Weather Research and Forecasting (WRF) Model" which is described in Powers et al., 2017. The model is capable of doing real-time numerical weather predictions in a high resolution. Some applications include modeling the behaviour of wildland fire and forecasting solar energy. Using the WRF model, detailed wind fields of the bay and its surrounding were simulated. With the help of a supercomputer, a computer with extraordinary computational power, these simulations can also be made in real-time to help the sailors to get the best possible understanding of the wind during their race. Figure 2.2 shows one of these simulated wind fields of the Sagami Bay area. We will refer to this model as the ‘wind model’ throughout this report.

2.2.2 Water Model

Ocean currents are primarily driven by winds, water density and tides. Coastal and sea floor features influence the direction and speed of these currents. Friction between wind and the surface of a body of water, causes the water surface to be dragged along with the wind. Many currents in the ocean are so dominant, they make the ocean water flow in predictable ways along its surface. Some of them even flow for thousands of kilometers and can be hundreds of meters deep. The Kuroshio Current and the Oyashio Current, mentioned in Section 2.1 are two examples of such currents. These dominant currents are always there, created by three factors: global wind patterns, the rotation of the earth, and the shape of the ocean basins. Research shows that under high wind speeds (20-50 ms^{-1}) and in open water, the effect of the wind on the near-surface water currents is linear as is described in Chang et al., 2012. In the northwestern part of the Pacific Ocean, the ratio between the wind and current is around 2%. The effect on the direction of the water for these high wind speeds in the northwestern Pacific is constant; the surface direction is approximately 10 ° to the right of the wind direction. Unfortunately, as predictable as currents on

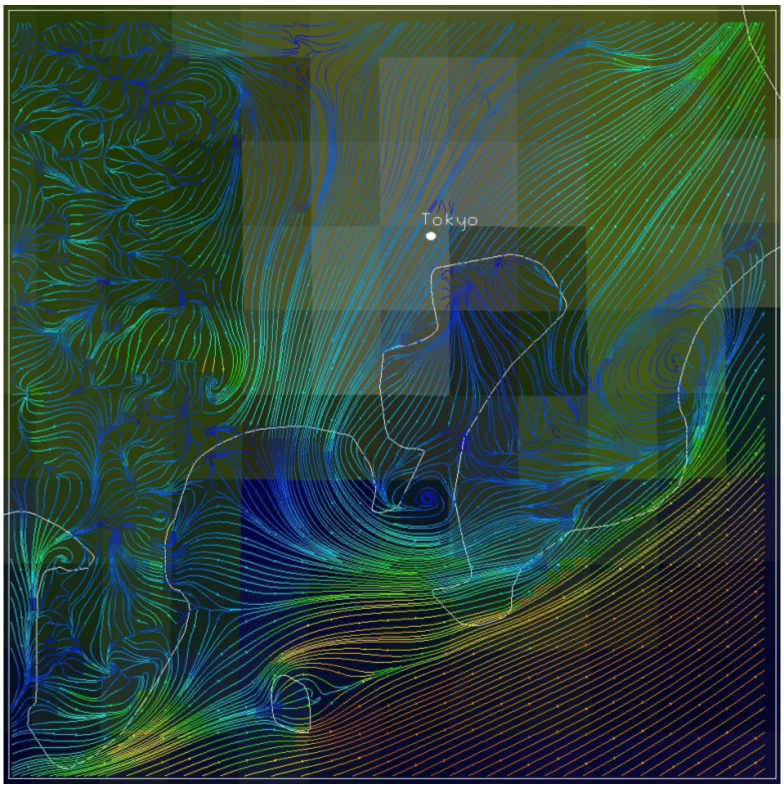


FIGURE 2.2: Complex wind field around Sagami Bay, modeled by meteorological model of Dr. Sukanta Basu, August 18th 5:00 local time. Lines represent wind, where the arrows show the direction and the colour represents the speed. Image: Dr. Sukanta Basu

the open water are this is not true for every body of water. The water current in coastal regions is much less predictable. In these areas, the wind is influenced by bodies of land, and differences in heat capacity between water and dry land. Often these influences cause the wind speed in these areas to be lower and less constant than on open sea. Figure 2.2 is exemplary of the difference in the complexity of wind in open water that generates predictable water currents, compared to wind along the coast where the wind is not as fast and influenced by the land. Besides the wind behaviour being less predictable, the tide is also a factor that in reality also plays a role in the water current in coastal areas.

The near-surface water current data used in this thesis stems from the FINEL2D model that was built by Svasek Hydraulics, an engineering firm specialised in water engineering Svasek, n.d. The FINEL2D model is a hydrodynamic water current model that can be used for modeling water current using various input factors. The model used for this thesis however, generated the data solely based the wind and the shape of the bay as the input parameters. The effect of the tide or salinity is thus neglected. We will refer to this model throughout the report as the 'water model'. Also when talking about 'water', we always mean the near-surface water.

Contrary to the wind model, the sailing federation cannot obtain results from the water model in real time. Hence it is the goal of this thesis project to create a neural network that can do so.

2.3 Data

The data set that was provided for this research consisted of pairs of simulated wind and water fields, representing the dynamics of the wind and water in and around Sagami Bay at a certain point in time. The wind fields were simulated using the WRF model, and the water fields were simulated with the Svasek model, using the corresponding wind fields as input. The total data represented 1200 wind and water field pairs, representing the respective dynamics at every hour, starting at 17th of July 2019 until September 4th of 2019.

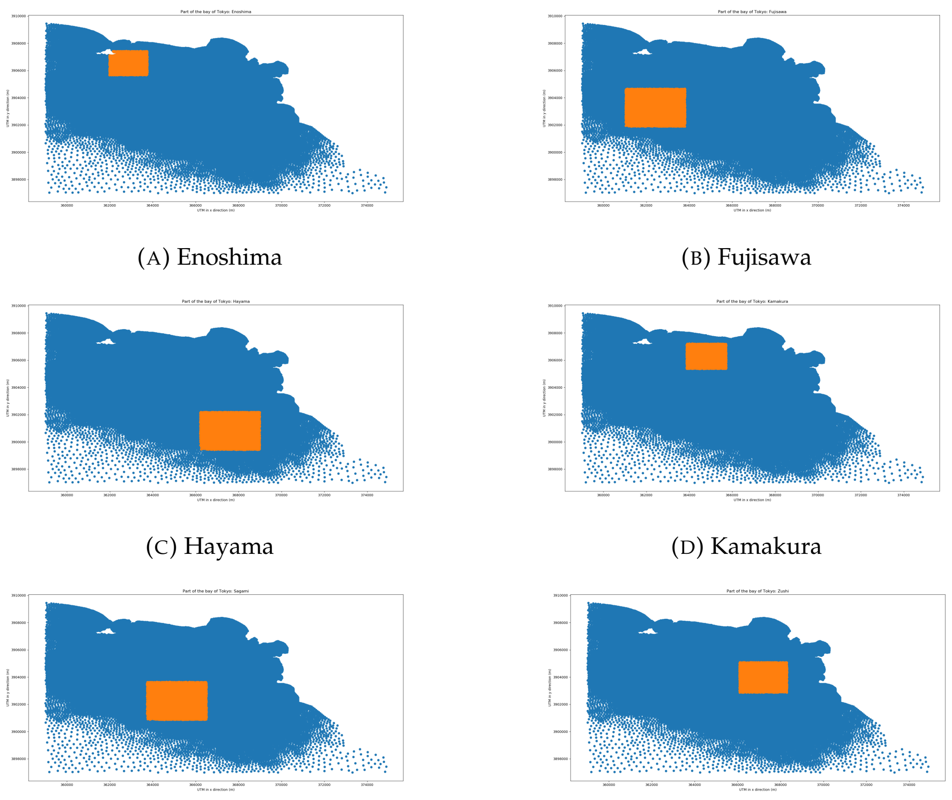
Each wind and water field consists of a grid with vector data describing the dynamics at these grid points. More precisely, every wind or water field is represented by a vector that contains the x and y component for the grid points in this field. The x component here represents the velocity of the wind or water respectively in the x direction (the longitude), while the y component represents the velocity in the y direction (the latitude). The first half of the vector contains the x components, while the second half contains the y components. Figure 2.3 shows the grid points of the area in which all sailing races were held. The different race areas are also represented in this Figure. As can be seen, the race areas vary in size and also consist of different numbers of grid points:

- Enoshima - 2674 gridpoints
- Fujisawa - 6542 gridpoints
- Hayama - 6450 gridpoints
- Kamakura - 2993 gridpoints
- Sagami - 6554 gridpoints
- Zushi - 4311 gridpoints

What can be seen from Figure 2.3 is that the density of the grid points decreases, as we look further away from the race areas. The area shown in blue, we will refer to as the 'Olympic Area' from now on. This area represents the north eastern part of Sagami Bay and contains all the race areas for the olympics. If we zoom out further, Figure 2.4 shows us the grid points representing the complete data, which we will refer to from now on as the 'Complete Bay'. Again we see that the density of the grid points is much lower outside the north of Sagami Bay. The number of grid points of these areas are as follows:

- Olympic Area - 69195 gridpoints
- Complete Bay - 80900 gridpoints

This indeed shows that, even though the olympic area is only a small part of the complete bay, most of the grid points are located in that area.



(A) Enoshima (B) Fujisawa

(C) Hayama (D) Kamakura

(E) Sagami (F) Zushi

FIGURE 2.3: Grids of all race areas Tokyo bay for the olympic games

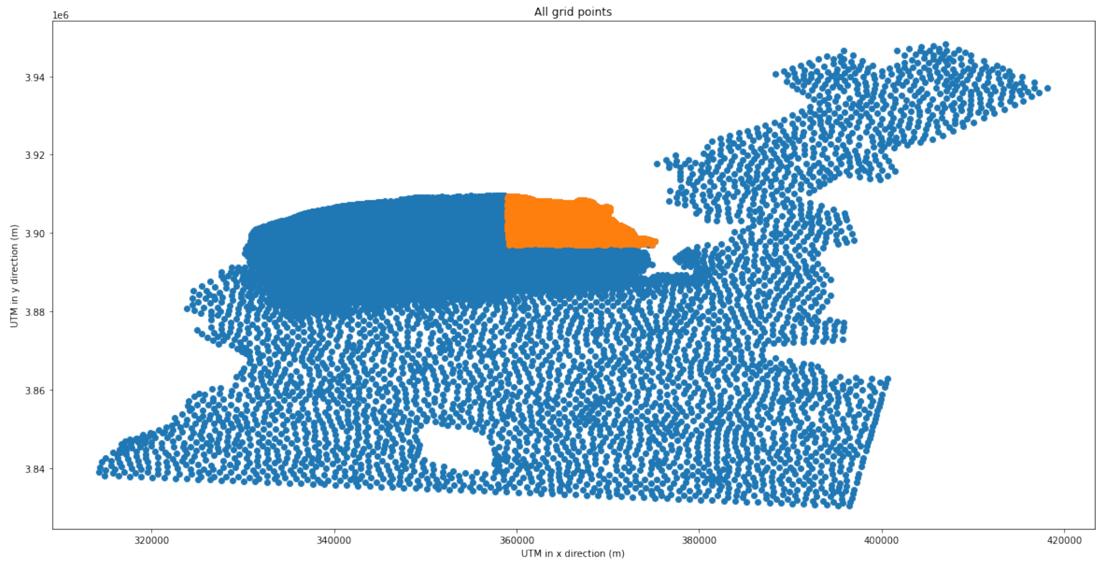


FIGURE 2.4: The complete set of grid points, representing one simulation of the state of the water current of the wind. The orange grid points represent the olympic area.

To summarize this chapter:

- Sagami Bay is a body of water from which we want to predict the near-surface current.

- We have simulated wind and water fields of Sagami Bay, generated by two different models. The bay is represented by a grid.
- The data consists of 1200 wind and 1200 water fields corresponding to each other. They represent the state of the wind and water current at every hour from part of the summer of 2019.
- Both wind and water simulations are represented by a vector that contains an x and a y component for every grid point. The wind or water at a subset of the bay can be obtained by using the x and y components of the corresponding grid points.

Chapter 3

Analysis of Wind Data

As mentioned in Chapter 2 the dynamics of a wind field are represented by a vector, containing the x and y components of the wind at the grid points located in the desired area. The desired area can be one of the race areas, any other area of interest, or simply the complete bay and its surroundings. One example of a race area is Hayama. In this area 6450 grid points are located. This means that a wind field in this area is represented by a vector of length 6450 x 2 = 12900, since there are two components for every grid point. A wind field covering all the grid points would be represented by a vector of length 80900 x 2 = 161800. Since we have a total of 1200 wind field simulations, our total wind data can be stored as a 1200 x 161800 matrix. Since the distance between grid points is small, especially in the race areas, we also know that there exist correlations within these wind fields. Because the wind in a specific grid point will be very similar to the wind in its adjacent grid points.
Because of these correlations between the grid points reducing the dimensionality of the data makes a lot of sense.

In this chapter we will reduce the dimensionality of the wind fields on 3 different scales, which are relevant as input for predicting the water current in the race areas. We start by looking at the wind fields on race area level, since these are the wind fields that directly influence the water at the locations where we want to predict the current. Secondly we will zoom out a little bit and look at the wind fields covering the full olympic area, as described in Chapter 2. Finally we look to simplify the wind fields covering the complete bay. These wind fields cover the whole of Sagami Bay, as well as Tokyo Bay and even part of the pacific ocean that is connected to Sagami Bay. There are two main benefits we get from reducing the dimensionality of the data. Firstly, this will increase the speed of the machine learning algorithms. Secondly, with a lower dimensionality, a lower number of training examples is required to strike a proper balance with the number of model parameters.

3.1 Dimensionality Reduction

In the next sections we will try to reduce the dimensionality of the wind fields for the different areas as mentioned earlier in the chapter.

3.1.1 Race Areas

As input for predicting the water current in the race areas, one of the options we consider is the wind directly above those areas. This is the wind that directly interacts with the water in the area, and thus should have a direct effect on the water.

It turns out that the method for reducing the dimensionality for the wind fields is the same for the different race areas. Therefore we will only discuss the results of one of the race areas (Hayama) in this chapter. Some results for the other race areas can be found in Appendix B.

From inspection we notice that, within the race areas, the simulated wind fields appear to be approximately constant. By constant, we mean that even though the wind fields vary over time, at each point in time, the direction and the speed of the wind, within the race area is approximately the same. If indeed the wind speed and direction in a race area are the same over all grid points in this area, the wind in a race area could be represented by one single vector, which represents the wind in all the grid points of the race area, making it possible to describe the wind field in the race area essentially with one two-dimensional vector. In this section we will analyse whether it is in fact reasonable to assume a constant wind field over the area.
To get a more intuitive understanding of the wind data we choose to transform the data into polar coordinates. This means that the data no longer consists of an x and a y component in every grid point, but rather a length component, which in our case represents a velocity, and an angle component, which represents the direction. This way we can compare the wind represented in the different grid points based on its speed and direction.

Figure 3.1 contains some randomly selected wind fields out of our data in the Hayama race area. In this Figure, the arrows represent the wind at the grid points. Here the direction of the arrows resembles the direction of the wind and the length of the arrows represents its speed. In addition to the arrows, the background colour gives an extra indication of the velocity of the wind over the different grid points. What stands out from this randomly selected sample is that, the arrows are very similar over the grid points, both in length as in direction. This holds for all four samples. The presumption that the lengths are the same is also confirmed by the background colour of the examples, which also looks constant in every example, indeed indicating a constant velocity in the field. Note that the variation of the colour has to do with scale. Choosing a much lower range of velocities might result in fluctuations in background colour. Hence inspection alone is not enough to assume that the wind is completely constant in any of these samples.
However, from looking at the samples, an idea originates. This idea is the following: if the wind over all grid points is very similar, we could assume this wind to be actually constant as a way to simplify the data. If we assume a constant wind over the area, we would only need one 2-D vector to represent the data in the whole area. For Hayama this would mean that we could reduce the dimensionality of the wind fields from 12900 to 2, thus drastically reducing the dimension of our data.

To check whether this assumption is viable, some analysis is required. To get a better idea of how similar the wind over the grid points is, we need to look at the variance of the data. The smaller the variance in the data is, the more justified it would be to reduce the data to one single vector.

In order to analyse the data we start off by introducing some notation. Let's number the grid points in a race area from 1 to N (for Hayama, $N = 6450$). We then have:

- ϕ_i : angle of wind at grid point i
- r_i : velocity of wind at grid point i

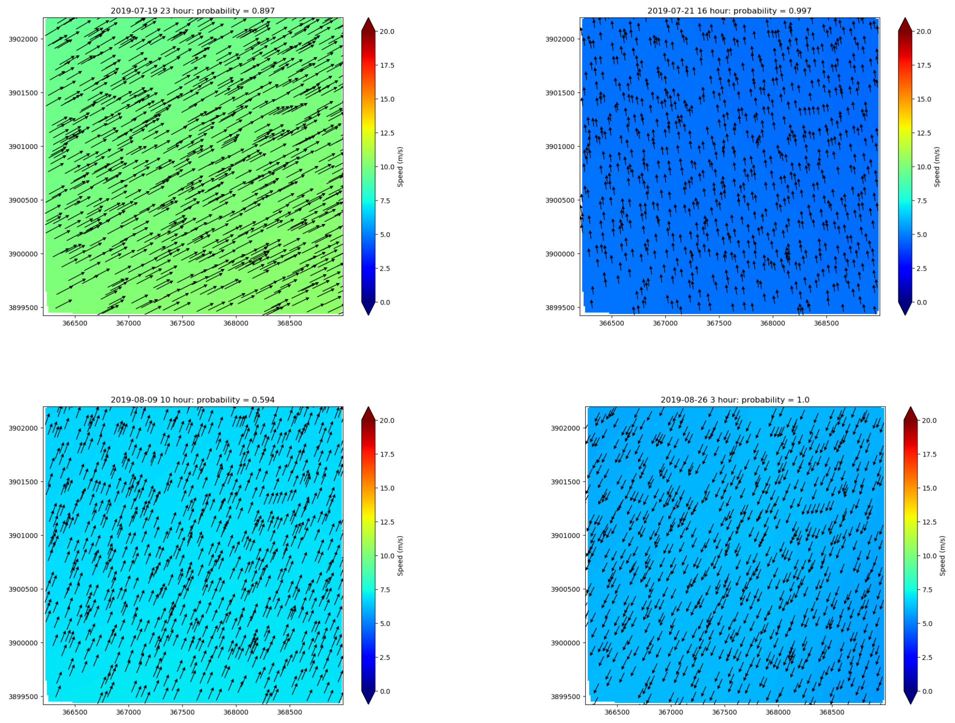


FIGURE 3.1: Examples of Wind Fields Hayama at different Time Stamps

For analysis purposes, we will describe each wind field as a sample set, in which each wind vector is independently drawn from a bivariate Gaussian distribution. This corresponds with the assumption that the wind is constant but contains measuring errors. We choose the parameters in the following way:

- $\mu_x = \arctan 2(\frac{1}{n}\sum_{i=1}^{n}\sin(\phi_i), \frac{1}{n}\sum_{i=1}^{n}\cos(\phi_i))$
- $\mu_y = \frac{1}{n}\sum_{i=1}^{n} r_i$
- $\sigma_x^2 = 1 - R$; where $R^2 = (\sum_{i=1}^{n}\sin(\phi_i))^2 + (\sum_{i=1}^{n}\cos(\phi_i))^2$
- $\sigma_y^2 = \frac{1}{n-1}\sum_i (r_i - \mu_y)^2$

Here μ_x is the circular mean of the angles and σ_x^2 is the circular variance of the angles. By choosing the parameters in this way, we assume a different distribution for every individual wind field. To assess the similarity in a wind field, a good visualisation is to plot an error ellipse. Figure 3.2 provides one example of such an ellipse. The Figure displays the wind at every grid point from Hayama. Figure 3.3 shows the ellipses of 50 randomly selected samples from the 1200 wind fields in the Hayama area. An error ellipse gives insight into three things:

- The variance over the y-axis, which in our case represents the speed. This is shown by the size of the ellipse along the y-axis.
- The variance over the x-axis, which in our case represents the direction. This is shown by the size of the ellipse along the x-axis.
- The correlation between the two variables. This is represented by the orientation of the ellipse.

These Figures give us an idea of how big the 'measurement errors' are that we mentioned earlier by the size of the ellipse. If the points are all very close to each other, the ellipse is small and thus the 'errors' are small, and we can indeed assume that the wind is constant. If the errors are big, the assumption that the wind is constant is less credible.

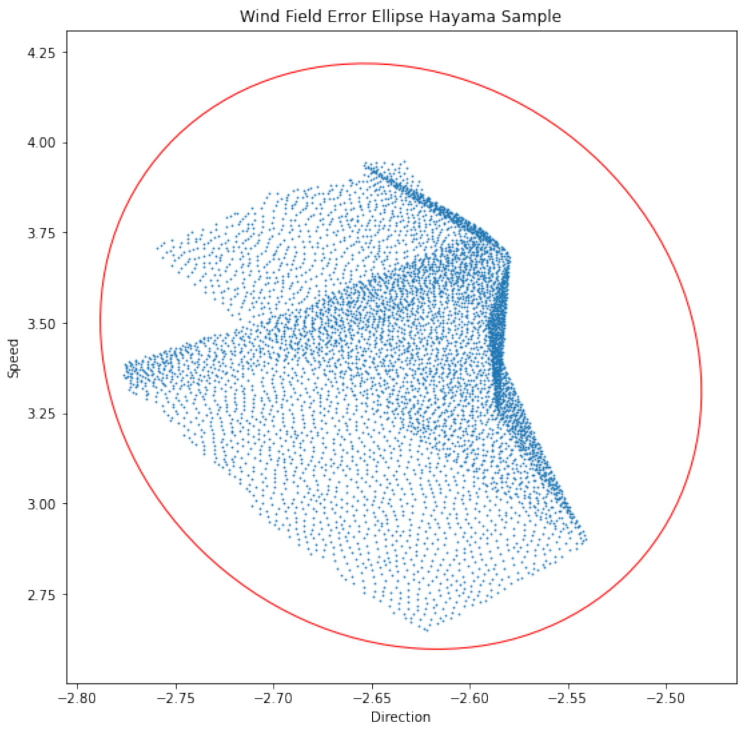


FIGURE 3.2: Error ellipse based on 98.9% Confidence Value.

From Figure 3.3 there are a few observations worth noting.

- Most of the samples are wind fields with angles between 1 and 2, i.e. wind from direction south-southwest
- There appears correlation between the velocity and the direction. However this correlation differs over the wind different wind fields.
- The higher the velocity of the wind, the less variance in the direction

The first observation will be discussed later on in Section 6.

The second observation tells us that there might exist correlation between the speed and the direction within a wind field. This can also clearly be seen in Figure 3.2, where a clear structure can be found in the data, opposed to an actual normally distributed data set. In reality, data measurements from neighbouring grid points are more similar, and the wind gradually changes over these grid points. Thus our assumption that the wind is constant and that the variance of the wind over the grid points is due to measurement errors is not correct. Also, there does not seem to be a specific correlation between the velocity and the direction of the wind that is uniform over all wind fields. In Figure 3.3, various shapes and orientations of the ellipses can be observed, hence implying different relations within the data.
To argue that the correlation within a wind field itself is independent of the direction

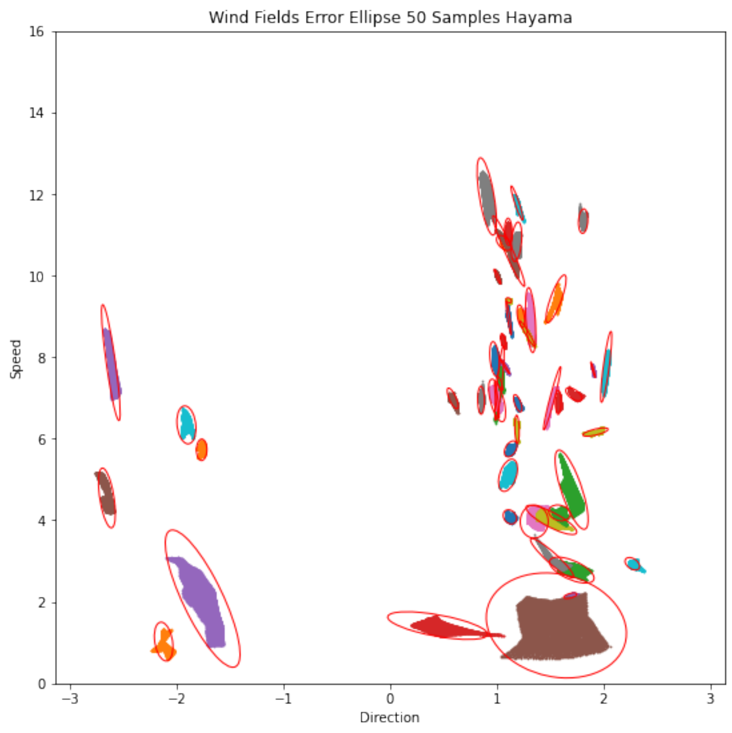


FIGURE 3.3: Error ellipses of 50 Wind Fields in Hayama. Every point cloud with corresponding ellipse represents a Wind Field.

and velocity of the wind, we plotted the correlation against the average velocity and against the average direction in Figure 3.4[1]. This plot suggests that indeed we will find no relation between the correlation within a wind field and the average direction or velocity of the wind.

Finally the third observation implies that for relatively weak winds, i.e. wind fields with low velocity, the direction varies a lot more than for winds with higher velocity. Figure 3.5 shows the relationship between the variance in the direction and the mean velocity of the wind fields. Indeed, we can see that only for wind fields with a low velocity, there exists variance in the direction. For winds with a mean velocity greater than 4 meters per second, the variance is already much lower and quickly decreases as the mean velocity increases. The argument that we can make here is that even though there are some wind fields that have quite some variance in terms of direction. All these wind fields have a very low velocity. And since these wind fields have a low velocity, their effect on the water current is very low. Hence, the direction, or even variance in direction for wind fields with very low velocity is not very relevant, as their influence is hardly affected by it.

In terms of the variance in velocity, we do not observe such a clear relationship. We do however see that the variance is very low for the majority of the wind fields. This can be seen in Figure 3.6. The relative variance, i.e. the coefficient of variation, of the velocity does however decrease when the mean velocity increases as can be seen in Figure 3.7. This means that relatively speaking, the wind becomes more constant as it gains in strength.

[1]For this plot we've only used wind fields in which the direction in all grid points was positive or in which the direction in all grid points was negative.

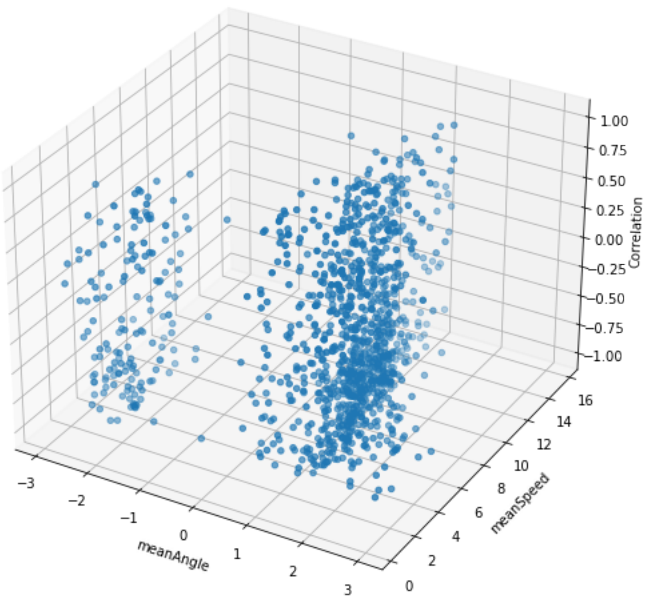


FIGURE 3.4: Correlation in each Wind Field plotted against its average velocity and average direction.

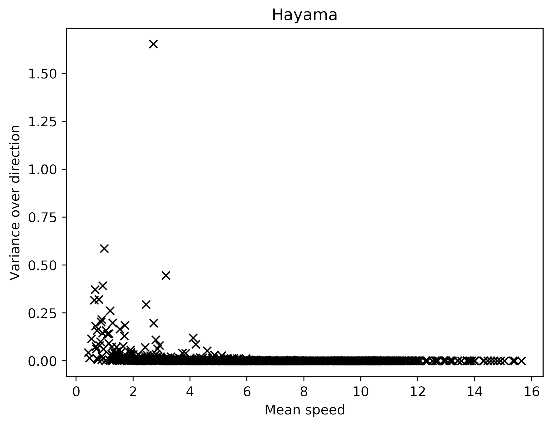


FIGURE 3.5: Variance in direction plotted against mean velocity of wind fields.

Even though there are multiple arguments that support the assumption of a constant wind, the assumption does not hold up in all cases. Also other questions will arise like, how would you determine the exact velocity and direction of this wind. The arguments do however lead to a conviction that reducing the dimensionality of the wind fields can be done without losing too much information. A popular mathematical technique for doing such reductions is using Principal Component Analysis, which we will refer to as 'PCA' from here on out.
The main idea behind PCA is to replace the original variables by derived variables, which are called the principal components and are linear combinations of the original variables as explained in Jolliffe, 2005. The principal components serve as a hierarchical coordinate system. This means that there is a certain hierarchy in the components. For the principal components, this hierarchy is based on the variance captured by the principal component. In case the first few principal components

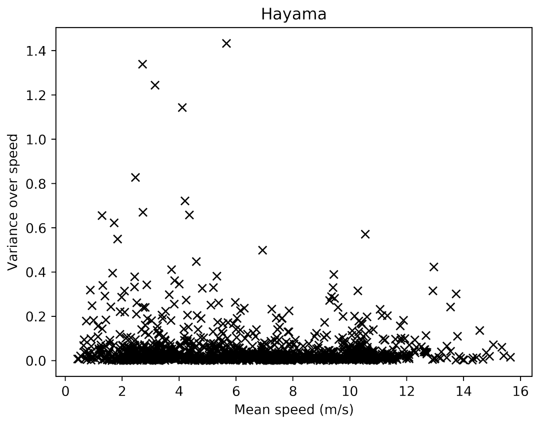


FIGURE 3.6: Variance in velocity plotted against mean velocity of wind fields.

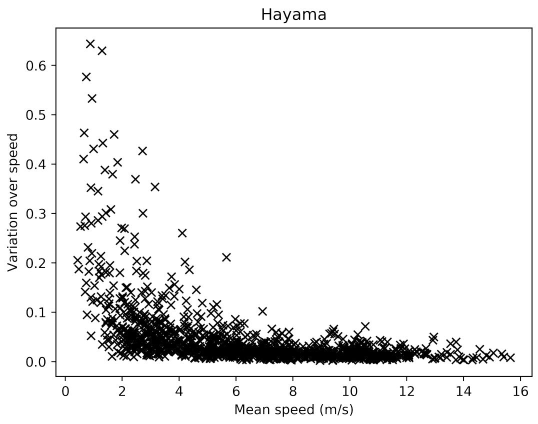


FIGURE 3.7: Relative variance in velocity plotted against mean velocity of wind fields.

contain enough of the information, it is possible to use a smaller amount of variables to represent most of the data. By doing this we can preserve a large percentage of the variability of the original variables, while the number of variables is significantly smaller. Suppose we have N samples called $x_1, x_2, \ldots, x_N$, where each x_i has k entries. Our data can then be represented as an N x k matrix X. The process of transforming data using principal components is described in Jolliffe and Cadima, 2016. There are several ways to execute the technique. One way is by using the following steps:

1. Start off by calculating the mean of all of our samples, i.e. the mean of all rows of X. $m_X = \frac{1}{N}\sum_{i=1}^{N} x_i$. This is the mean vector, as all our rows are vectors. We create a matrix with N rows, where every row is exactly m_X. Let's call this matrix $\hat{X}$.

2. Create a new matrix in which we store the 'mean centered data'. $B = X - \hat{X}$. Now the center of the distribution creating the wind fields is at the origin.

3. Construct the sample variance-covariance matrix C of the rows of B, so $C = \frac{1}{N}B^T B$.

4. Compute the real, nonnegative eigenvalues and eigenvectors of matrix C. Recall that for an eigenvector v_i and its eigenvalue λ_i the equation $Cv_i = \lambda_i v_i$

holds. We then construct a matrix V containing all the eigenvectors of C, where the vectors are sorted based on the magnitude of their corresponding eigenvalues. This means that the first row of V is the eigenvector that corresponds with the highest eigenvalue, followed by the eigenvector corresponding to the second highest eigenvalue etc. The eigenvalue decomposition leads to the following equation: $CV = VD$, where V is the matrix containing the eigenvectors in order of magnitude as just described and D is a diagonal matrix containing the corresponding eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_K$.

5. We can now create the principal components. These are the new variables that are linear combinations of our initial variables. They are uncorrelated and are ordered by the amount of information they contain from the original variables. The principal components can be found by constructing matrix $T = BV$. The rows of matrix T represent the principal components. The values in each row are the loadings of the columns for that specific principal component. These loadings represent the correlation between the original variables and the principal components. The eigenvalue corresponding to the principal component tells us about the captured variance by this principal component i.e. the amount of information from the original variables. The percentage captured by principal component 1 for example is calculated by $\frac{\lambda_1}{\sum_{j=1}^{K} \lambda_j}$, where $\lambda_1, \lambda_2, \ldots, \lambda_N$ are the eigenvalues ordered by magnitude. If we would like to use $d \leq K$ principal components to describe our data, the variance captured can be calculated similarly: $\frac{\sum_{j=1}^{d} \lambda_j}{\sum_{j=1}^{N} \lambda_j}$

6. Using matrix T we can express our original data vectors $x_1, x_2, \ldots, x_N$ in terms of the principal components. The transformed vectors, y_i are obtained in the following way: $y_i = Tx_i$. The entries of the transformed vectors y_i represent the position of the field in the new coordinate system i.e. the coordinates. These new coordinates are referred to as the scores.

7. To transform the vectors back to the original space, we use $\hat{x}_i = T^T y_i + m_X$. Note that the projected vector, $\hat{x}_i$ is not equal to the original vector x_i unless we use all principal components. If we use less principal components, information will be lost, as is explained by the captured variance of the principal components. The mean square error between the projected vectors after PCA and the original vectors is equal to the sum of the eigenvalues corresponding to the principal components that were not used. Note that $\hat{x}_i$ is nothing more than the average of all samples (m_X) plus the scores corresponding to the principal components for sample i (y_i) multiplied with their respective loadings (which are stored in matrix T).

We start off by discovering how PCA would work in the case where the assumption we originally made is true. This assumption stated that the wind is actually constant (with measurement errors). Therefore we now look at a dataset in which this constantness holds. The dataset we use for this is based on measurements of historic winds of one single location. By applying this measured wind to all grid points of the wind field, we create a constant wind field over the Hayama race area. This dataset has the same features of our original dataset except for the values, which are now actually constant.
If we apply PCA to this dataset (the constant wind fields), we find that there are only 2 principal components needed to capture 100% of the variance. This is expected, since the wind is constant over the whole field, and a constant wind can be

described using only 2 dimensions. The captured variance can be seen in Figure 3.9. What we have done here by using PCA, is reduce the data of an entire 12900 dimensional wind field to 2 dimensions while still preserving all the information.

If we look at the loadings for the first two components we find something that makes sense intuitively. The first principal component has a loading score of -0.005 for the first 6450 entries (the x components), and a loading score of -0.01 for the second 6450 entries (the y components). This tells us that all x components are equally correlated with this principal component, and all y components are also equally correlated with this principal component. This makes sense since the x components are all the same and the y components are all the same. For the second principal component we observe a similar result, where the loading of all x components is the same and the loading of all y components is the same as well.

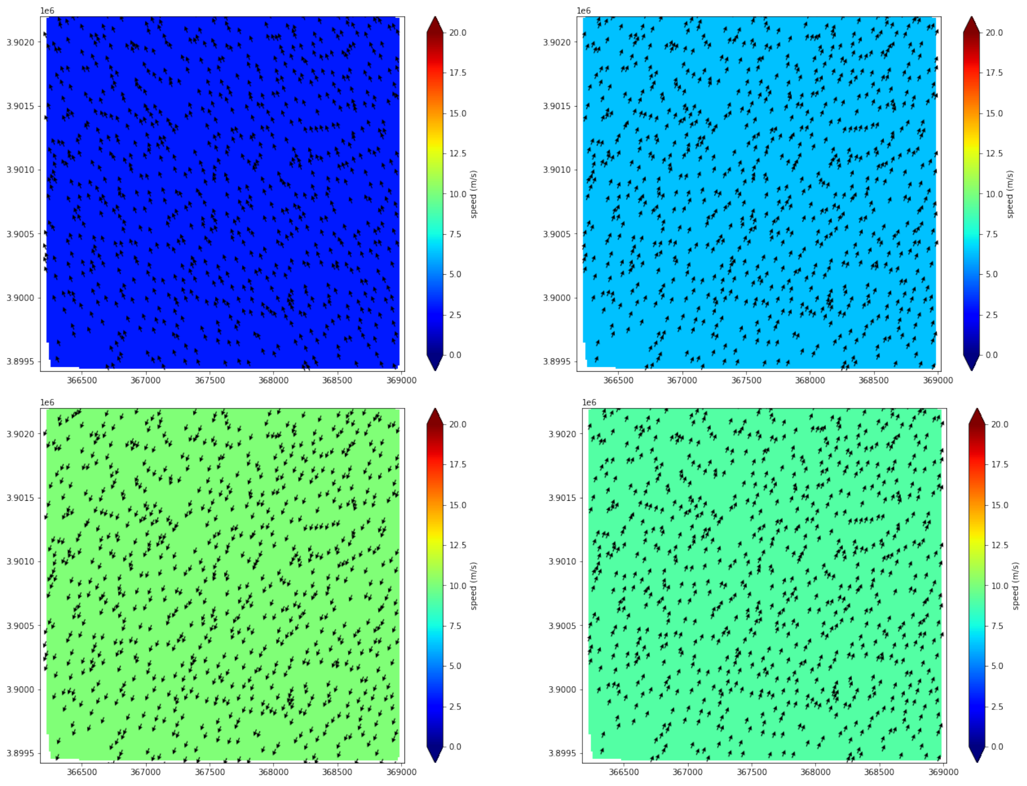


FIGURE 3.8: Constant wind fields at Hayama. Generated by applying a measured 2D wind value over all grid points.

Now that we have seen the result of applying PCA on data that actually describes constant wind fields, we want to apply it on our actual data.

After applying PCA on the data we find out that the first two principal components capture 99.6% of the variance as is shown in Figure 3.10. Throughout this report we choose to use 99% as the required threshold for the amount of explained variance, in order to represent the data as well as possible, while being able to reduce dimensionality. Hence the 99.6% is sufficient for reducing the dimensionality of the wind fields in Hayama to 2 by using these components. Therefore we can conclude that, by using PCA, we achieve our goal of reducing the dataset containing 12900-dimensional wind fields to a dataset of 2-dimensional wind fields.

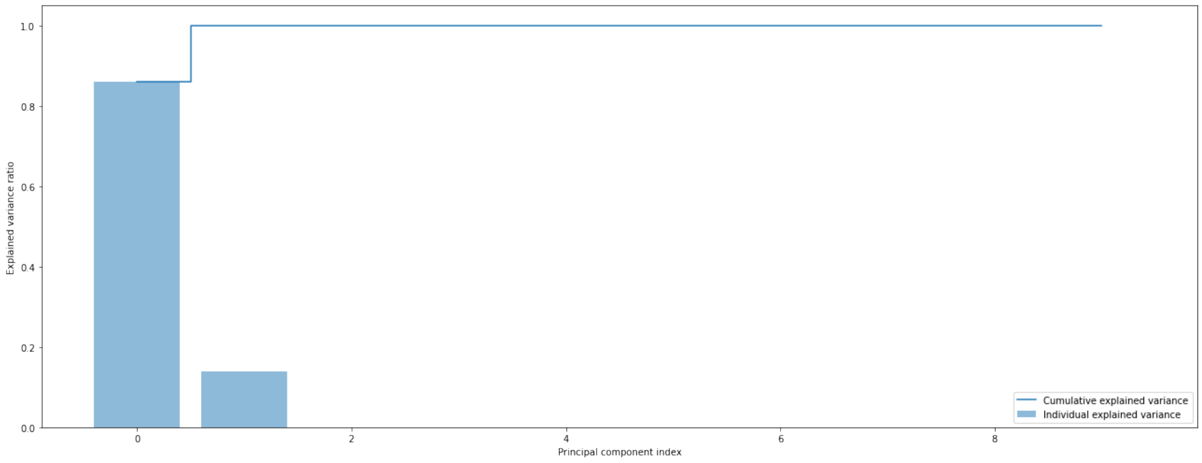


FIGURE 3.9: Captured variance by the principal components for constant wind fields in Hayama.

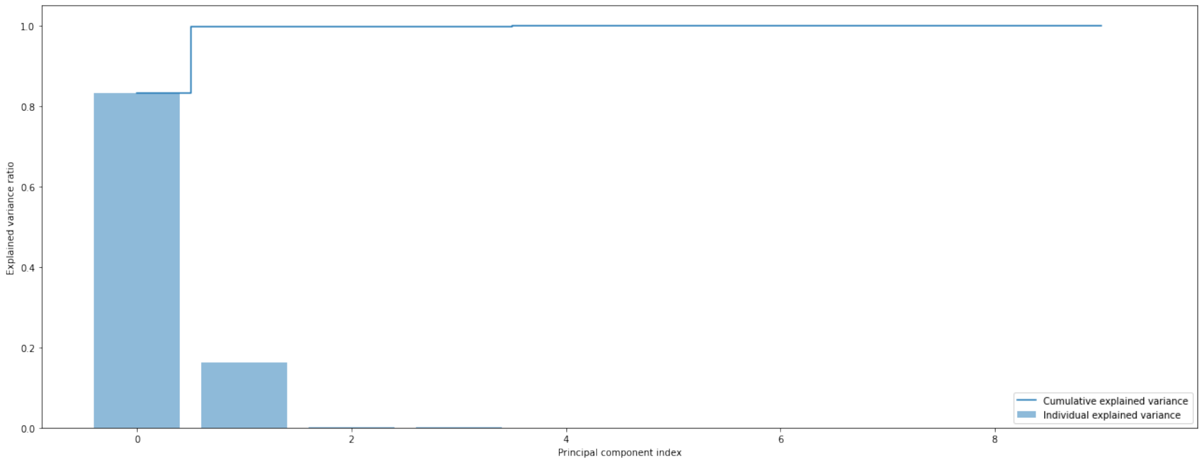


FIGURE 3.10: Captured variance by the principal components for the simulated wind fields in Hayama.

The loadings of the first two principal components show something similar with the loadings for the constant wind data set. For the first principal component, which explains about 80% of the variance, the loadings of the x components are very similar and the loading scores of the y components are very similar. For the x components, the loadings for PC1 are all between -0.0066 and -0.0059. For the y components the loading scores for PC1 are all between -0.0104 and -0.0108. Even though these loading scores are not exactly the same for all x components and all y components they are very similar. This indicates that each grid point contributes (almost) the same to PC1. For PC2 we find the same type of results, where the x components of the grid points contribute equally and the y components of the grid points contribute equally.

The loadings can be seen as the weight given to each original variable when calculating the principal component. In our case the original variables of course are the x and y components of the grid points of the wind field. If all x components have the same weight, this means that we actually use their average. The same goes for the y components. The fact that the average of the x components and the average of the *y* components already provide enough information to explain more than 99% of the variance is another indication that the field must be (almost) constant as we already suspected at the start of this chapter.

3.1.2 Olympic Area

In this section we look to reduce the dimensionality of the Olympic Area as is described in Chapter 2. This area is a lot bigger than any single one of the race areas, since it already contains all of them. The idea that we can reduce the wind fields in this area to a 2-Dimensional vector is therefore a bit far fetched.
Applying PCA however is still a good idea to reduce the dimensionality of these wind fields. The aim is to still explain 99% of the variance of the data. Figure 3.11 shows the explained variance by the first 10 components. We see that the result is still very similar to the PCA result of Hayama and even of the constant wind fields of Hayama. This indicates that the wind in this area does not vary heavily over the grid points. It turns out that with 6 components we achieve our goal of capturing more than 99% of the variance. Thus as input for our Neural Network it suffices to use these 6 components for every wind field, instead of the 138390 dimensional vectors that originally describe these fields.

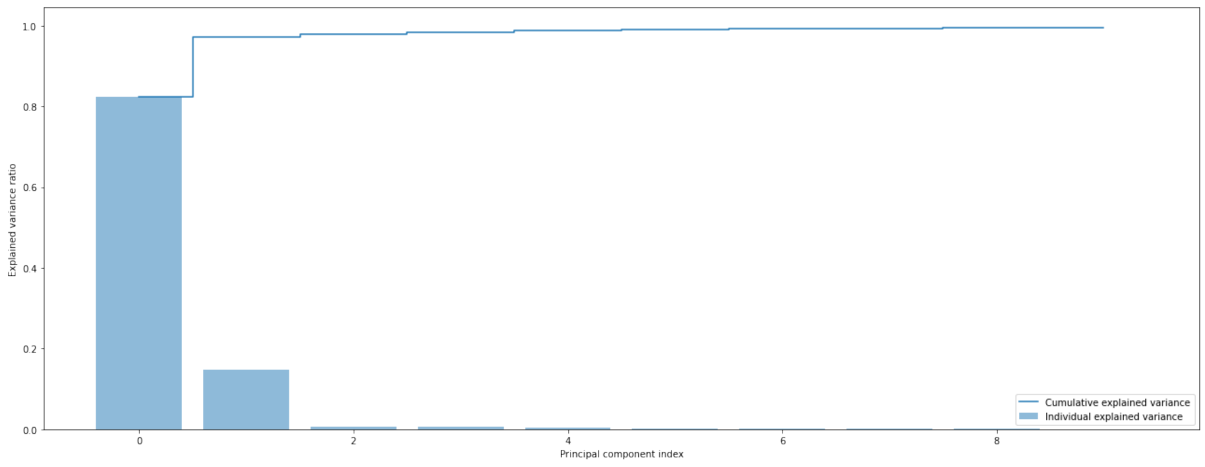


FIGURE 3.11: Captured variance by the principal components for the simulated wind fields in the Olympic Ara.

3.1.3 Complete Wind Fields

In this section we apply PCA to the full simulated wind fields i.e. the wind fields formed using all the grid points. As mentioned in 2 already, this means that these wind fields contain not only Sagami Bay, but also Tokyo Bay and even part of the pacific ocean. As can be seen in Figure 2.2, the wind seems to vary a lot over this large area.

Figure 3.12 shows the results of applying PCA to this dataset. Indeed we find out that a lot more principal components are needed to explain 99% of the variance of these large wind fields. The exact number of principal components necessary turns out to be 25.
What stands out from Figure 3.12 however is the fact that still the majority of the variance is explained by the first principal components. To understand this we

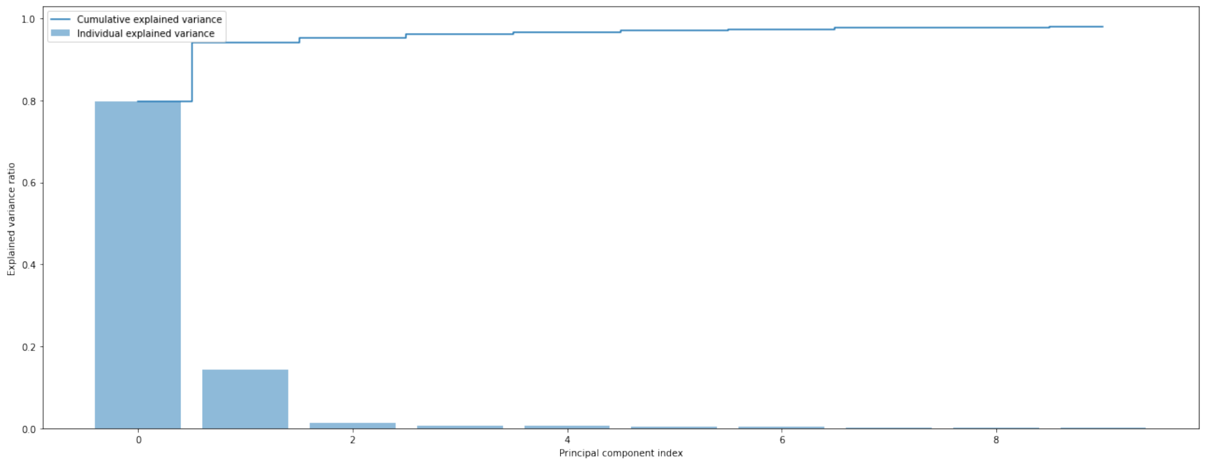


FIGURE 3.12: Captured variance by the principal components for the simulated wind fields over the whole area.

should first note the difference in density of the grid points over the complete bay. As mentioned in Chapter 2 already, the complete wind fields contain 80900 gridpoints, while the olympic area, which is significantly smaller as can be seen in Figure 2.4, contains 69195 gridpoints. Even in Figure 2.3 it is already apparent that there is a discrepancy in density of the grid points. The highest density is clearly where the race areas are located.

The second thing we should understand is that a principal component explains a certain percentage of the total variance. The total variance is the trace of the covariance matrix of the samples. In other words, it is the sum over all variables of the variance of each individual variable, where the variables as we know are x and y components of the grid points in our area.

To illustrate why the first and second principal component are still able to explain so much of the variance of the complete wind fields let's compare the total variance of the wind fields in the olympic area to the total variance of the complete wind fields. The total variance is calculated by summing the variance over each individual variable:

$$\sum_{i=1}^{N} \left(\frac{\sum_{j=1}^{M} x_j - \overline{x_i}}{M-1} \right)$$

In this equation, N is the number of variables, which is in our case the number of grid points and M is the number of samples. $\overline{x_i}$ is the mean for each grid point i over all samples.

- Total variance of the wind fields in the olympic area: 1671753.46
- Total variance of the complete wind fields: 1977648.19

As we can see, most variance is caused by the data in the olympic area. So even though the area of the complete wind fields is multiple times the size of the olympic area, due to the number of grid points, the total variance only increases by 18%.

Now if we look at the loadings of the first principal component, we observe a much higher variety. The values for the x components have loading scores ranging between -0.0025 and -0.0001. For the y components, the loading scores for PC1 range between -0.0034 and -0.0002. This means that not all points are weighed the same anymore for PC1 as was the case earlier in Hayama. However it is still the case that 82% of the variance originates from the olympic area that contains all race areas. And we have seen that the variance from that area can be very well explained with two principal components. The additional principal components are however very important in describing the wind outside the olympic area, which might also influence the water current in the race areas.

Chapter 4

Water Current Analysis

The aim of this chapter is to decide upon a dataset which we will use as output to train our neural network on. Since the original current data is very high-dimensional, and this is undesirable for our neural network, we would like to find an alternative. Data dimensionality can be reduced in multiple ways. In this chapter we discuss two methods to reduce data dimensionality. First, the data fields have been categorized using clustering. Second, PCA techniques have been used, as described in Chapter 3, to reduce the dimensionality of the data. The idea behind clustering the data is to investigate whether the water currents can be divided into a small number of categories, in which the current fields behave similarly. If this is the case, only knowing the category would be enough information for the sailors to understand the behaviour of the current. This would mean that our neural network would only need to predict the correct category for the current field, depending on the wind input. We will use clustering to see whether it is possible to divide the water fields into these categories. If clustering turns out to be succesful a classification model will be trained. If we decide to go with the PCA reduced data, a regression model will be trained. Throughout this chapter we will give analysis results for the Hayama area, as these results are very similar for all individual race areas. The results for the other areas can be found in Appendix B.
To provide better insight in the water current fields, some typical examples are shown in Figure 4.1. As can be seen from these examples, the water fields contain a lot more variance in terms of speed and direction than the typical wind fields we have shown in Chapter 3.

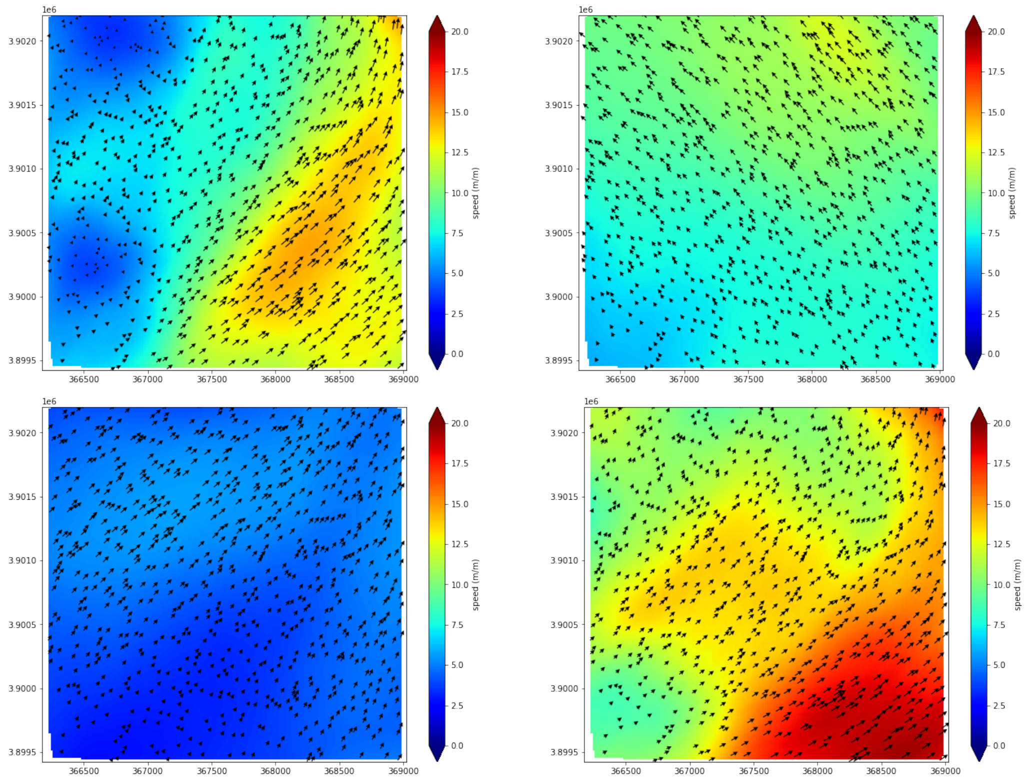


FIGURE 4.1: Examples of Water Current Fields of the Hayama race area at different Time Stamps

4.1 Clustering

Using clustering, we will answer the question whether there is a small number of clusters in which the water fields are very similar. If this is the case, our Neural Network should only need to predict the cluster number instead of the complete current field.

Clustering is a well known method used to categorise data by attempting to find clusters which consist of groups of similar data points. It is an unsupervised machine learning technique, which means that the data is not categorised beforehand. This makes sense for our problem, since we don't know beforehand whether there is a certain set of categories, and if so which field belongs to which category. By using clustering, an algorithm will assign a label to each data point, based on an optimization criterion. What we do need to give as an input is the number of categories (clusters). There exist many clustering techniques, but for this research the K-means clustering method, which is described in Likas, Vlassis, and Verbeek, 2003 is used. This technique is one of the simplest clustering algorithms, yet it is an effective way of creating clusters and assigning data points to it, which is what we aim to achieve. K-means clustering is intuitively easy to understand, which helps in interpreting the results as well. The method divides the data into K distinct categories based on the coordinates of the samples, which in our case are the x and y components of the water current in the grid points of Hayama. More formally: we have a data set $\{x_1, \ldots, x_n\}$ consisting of samples, which in our case are current fields. The goal of the algorithm is to partition the set into K disjoint subsets (clusters) $C_1, \ldots, C_k$ in such a way that the total variation within these clusters is minimized. This is often done by looking at the squared Euclidean distances between the cluster center m_i and the samples within that cluster. This is the criterion:

$$\arg\min_{m,C} \sum_{i=1}^{n} ||x_i - m_{C(x_i)}||^2$$

where $m_{C(x_i)}$ represents the cluster mean belonging to the cluster to which x_i is attributed.

Figure 4.2 shows a 2-D example of this total variation, where 3 clusters are used on one-dimensional samples.

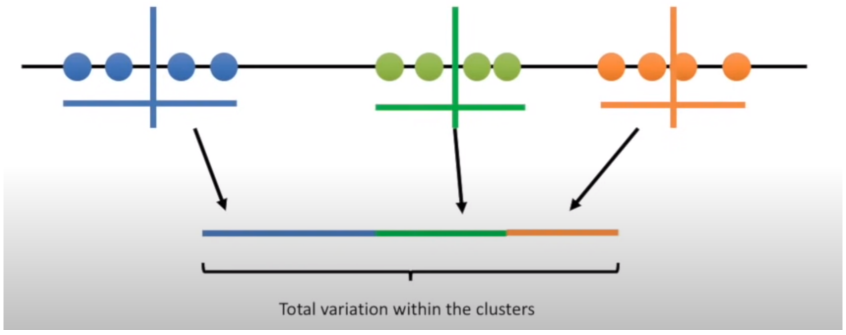


FIGURE 4.2: 2-D Example of total variation within 3 clusters.

If we can find distinct clusters, we find categories which we can contribute to our data. The lower the variation between the samples belonging to the same cluster, the more likely it is that we can actually use the category to characterize the current fields. A way to define the fields by a category could be for example by using the average of the samples of this field.

The first step in our clustering analysis is to find out whether there exists a number K of distinct clusters, such that every sample clearly belongs to one of these clusters and clearly not to any of the other ones. A good way to find this number K is by using the 'elbow-method'. To use this method we start by defining 'distortion'. The distortion value is the sum of the squared (Euclidian) distances between the samples and their respective cluster centers. To illustrate this method, Figure 4.3 shows the distortion values for the number of clusters for an unrelated data set that is very suitable for clustering. To determine the optimal number of clusters, we have to select the value of K at the 'elbow', i.e. the point after which the distortion starts decreasing in a linear fashion. For the data set belonging to Figure 4.3 this value is 3. What this Figure illustrates is that there is a clear reduction in distortion going from 2 to 3 clusters, while there is no clear reduction anymore when increasing the number of clusters.

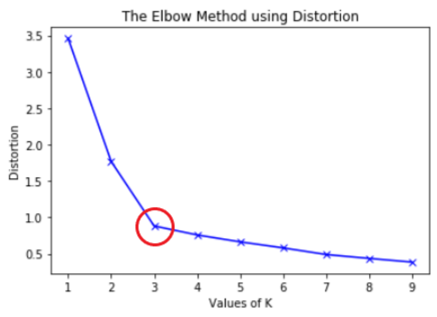


FIGURE 4.3: Example of elbow method on unrelated data set. The red circle clearly indicates the 'elbow' of the graph.

Figure 4.4 shows the distortion against the number of clusters for our actual Hayama current data. As there was a clear 'elbow' in the example shown in Figure 4.3, this is clearly not the case in our actual data. This is an indication that there does not exist a clear number of clusters that would make sense. We can also see that for our actual data the distortion values are very high. This is not a good sign, as this indicates that the data is on average not very similar to their respective cluster centers. For comparison: the average squared distance of the two current fields with the average of the two fields, shown in Figure 4.5 is approximately $0.5e6$. Since these two current fields are very clearly not similar, the fact that the distortion values shown in Figure 4.4 are higher only illustrates that the data cannot effectively be divided in different clusters.

As an extra analysis for choosing the optimal number of clusters we can look at so called 'silhouette values' for the number of clusters. These values give an indication of how close the samples are to their respective cluster center compared to other cluster centers. This is seen as a measure of how well the samples are classified. A silhouette value can be assigned to each individual sample i and is defined in the following way:

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}; a(i), b(i) \leq 0$$

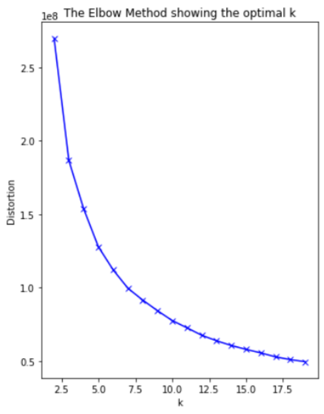


FIGURE 4.4: Elbow method on Hayama current data.

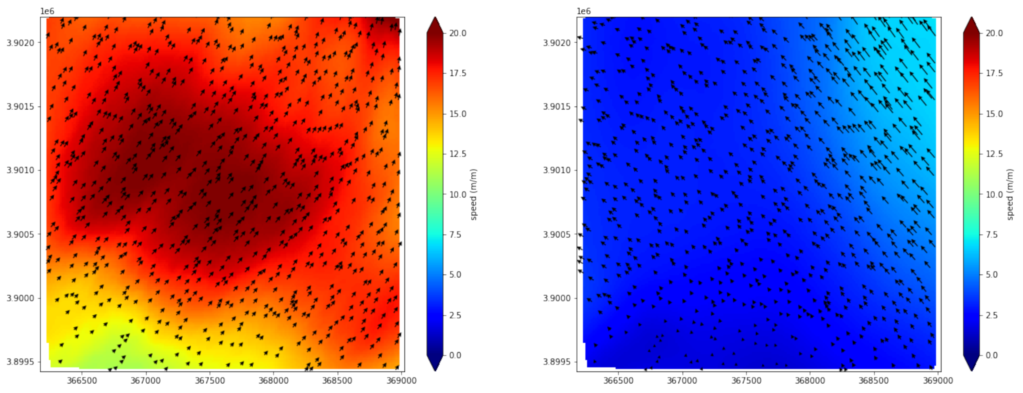


FIGURE 4.5: Two current fields that are clearly not similar to each other.

Where $b(i)$ is the average distance between sample i and all samples belonging to the closest (again Euclidean distance) other cluster. $a(i)$ is the average distance between sample i and all other samples belonging to the same cluster. The silhouette value $s(i)$ can take values between -1 and 1. If the clusters are very distinct, which is what we want, the Euclidean distance between sample i and the other samples in its own cluster is very small compared to the Euclidean distance with the samples in the 'closest' other cluster. In that case $a(i)$ would be small and $b(i)$ would be big, resulting in a silhouette value close to 1, which would be the best value. The closer the value gets to -1 the worse the classification has been, since in case of a negative silhouette value, the sample is actually on average closer to sapmles in another cluster than to the samples of its own cluster. In Figure 4.6 we have plotted the average silhouette values over all samples for different numbers of clusters. What we can see from the silhouette plot is that the best values are actually achieved when we would use 2 or 3 clusters. If we use more clusters, we actually get worse average silhouette scores. This indicates that there is no clear distinction between the samples from different clusters if we use so many. On the other hand we have already seen that the distortion value is very high if we would only use 2 or 3 clusters. To illustrate this: using 3 clusters the distortion value is about 1.8e08. This means that the average distance between a sample and its cluster mean equals $\frac{1.8e08}{1200} = 0.15e06$. This is not much smaller than the distance between the two current fields shown in Figure 4.5. Using 2 clusters, the average distance is even bigger. This means that we would not be able to characterize current fields by taking the average of its cluster,

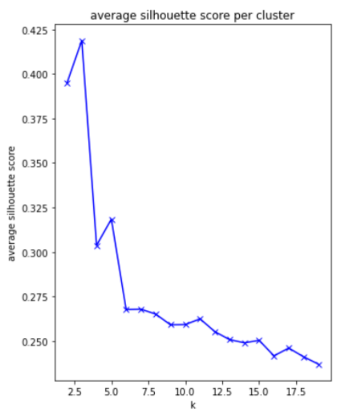


FIGURE 4.6: Average silhouette values on Hayama current data clustering.

since the samples differ a lot within these clusters.

We can conclude that the idea of characterizing the current fields by a category is not suitable for our data based on the elbow-method, the silhouette values and the distortion values.

4.2 PCA

As the K-means cluster technique did not provide clear current field categories, principal component analysis is used to reduce data dimensionality. The principal component analysis technique we apply on the current data is the same as we have already described in detail in Chapter 3. To retain as much detail in the current fields as possible, just as for the wind fields, we would like to capture at least 99% of the variance of our data with our principal components. Whereas we would only need 2 principal components to capture 99% of the variance in the wind fields in Hayama, for the current fields this number is 10. Intuitively this makes sense, since Figure 4.1 already displays that the current can differ a lot within a single field. Figure 4.7 shows the variance captured by the first 10 principal components.

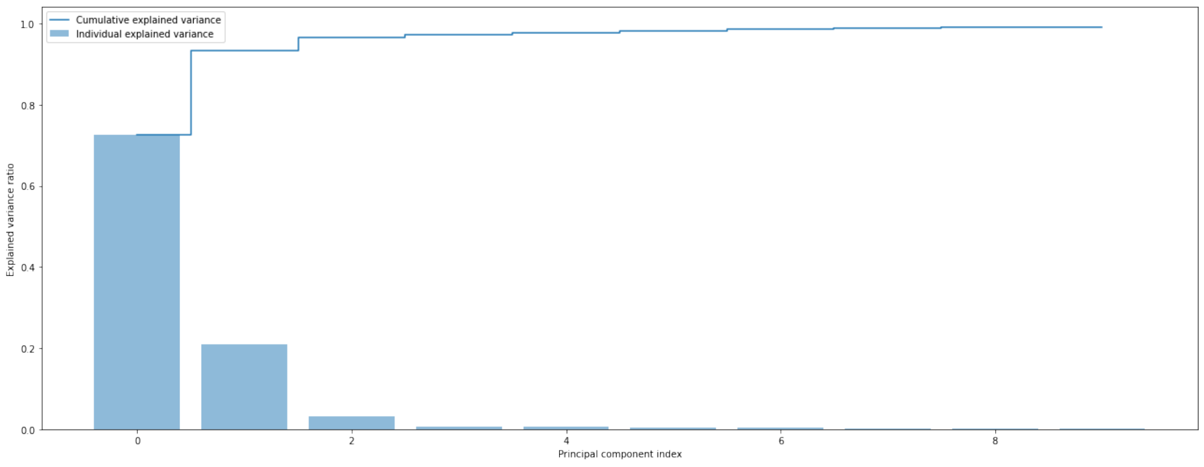


FIGURE 4.7: Captured variance by the principal components for the simulated current fields in the Hayama race area.

In Chapter 3 we have already discussed that by using principal component analysis we transform our data from its original space where our data is respresented by the x and y components of the grid points in our area, to a new space, in which we express the data by the principal components. For the wind fields, the goal was to do this transformation so that we could reduce the dimensionality of our data while still retaining a high percentage of its variance. For the current must do the same thing, in order to be able to make predictions fast. The principal components can help us create a faster, simpler neural network. However, we should remember that we want to predict current fields for the sailors and not just 10 principal components. Therefore the extra step that we need to take for the current data is that we should be able to transform the principal components back to actual current fields i.e. vectors that contain the x and y components of the grid points of the race area.

As explained in Chapter 3 the fields we obtain when transforming the principal component data back to the original space are not an exact copy of the original field. The error between the original field and the field we get after transforming the data back from the principal components depends on the number of principal components used and the amount of variance they capture. More precisely, the mean square error between the two vectors is equal to the sum of the eigenvalues corresponding to the principal components not used. In Figure 4.8 we see three plots of the same current field to illustrate the difference between the original data and data transformed back from principal components. The first plot shows the field based on the original data. The second and third plot show the field represented by the vector that is the result of transforming the data back from 9 principal components and 2 principal components respectively. Indeed we see that using 9 principal components we can do a better job at recreating the original data than we can with 2 principal components, even though the 2 principal components already capture 93.3% of the total variance.

To get a more quantitative understanding of the accuracy, besides the visualisation shown here, we can compare the current fields after the PCA operations, with the original current fields by constructig an accuracy score.
A commonly used method to compare two vectors is by looking at the magnitude of the delta vector, i.e. the difference between the original current field vector: $V_{original}$ and the current field after PCA: V_{PCA}. The magnitude of the delta vector will then be:

$$\sum_{i=1}^{N}(V_{\text{original}_i} - V_{PCA_i})^2$$

This results in an absolute number that gives an indication of how 'close' the prediction vector is to the original vector.
To get a better understanding of what this number actually means, we should make it relative to the magnitude of the original vector. It also turns out that there is a positive correlation of 0.582 between the magnitude of the delta vector and the magnitude of the original vector, implying that a higher magnitude of the original vector causes the difference to be higher as well. We look at the relative difference by dividing the magnitude of the delta vector by the magnitude of the original vector.

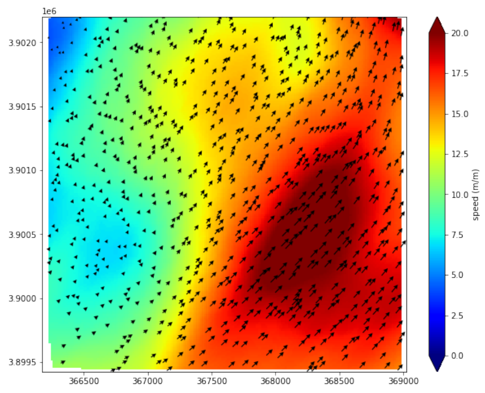
Here are some statistics from this method of measuring accuracy:

What stands out the most from these statistics, is the huge improvement of using 9 principal components compared to only using the first 2. We can interpret the

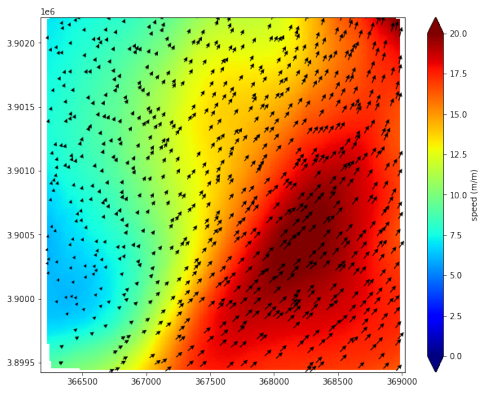
	2 Principal Components	
	Magnitude of Delta Vector	Relative Magnitude of Delta Vector
Maximum	221773.25	1.984
Minimum	1246.18	0.0022
Mean	26852.64	0.1036
Median	19526.37	0.0473
Standard Deviation	24856.09	0.1793

	10 Principal Components	
	Magnitude of Delta Vector	Relative Magnitude of Delta Vector
Maximum	27681	0.1811
Minimum	258.34	0.0004
Mean	4147.79	0.0162
Median	3139.13	0.0072
Standard Deviation	3687.99	0.0249

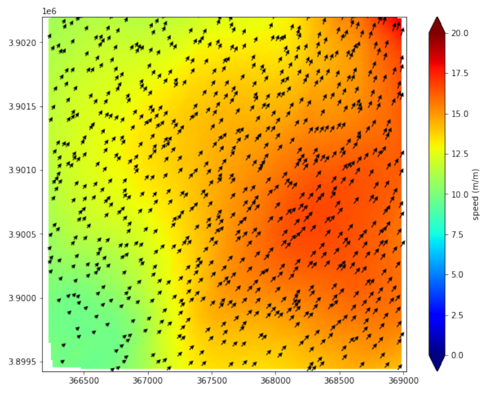
relative magnitude of the delta vector as a percentage, hence we see that, with using the 10 principal components, the mean 'error' is 1.62% and the median error is only 0.72%.



(A) Original current field



(B) Recreation of current field using 9 principal components.



(C) Recreation of current field using 2 principal components.

FIGURE 4.8: Example of a current field and its recreation using principle components.

If we only use the first two components, we can also confirm that there are no clear clusters, as we already suspected from our analysis before. Figure 4.9 shows a scatter plot containing all these 2-dimensional representations of the current fields, where the scores of the first component are represented by the x-axis and the scores of the second component are represented by the y-axis.

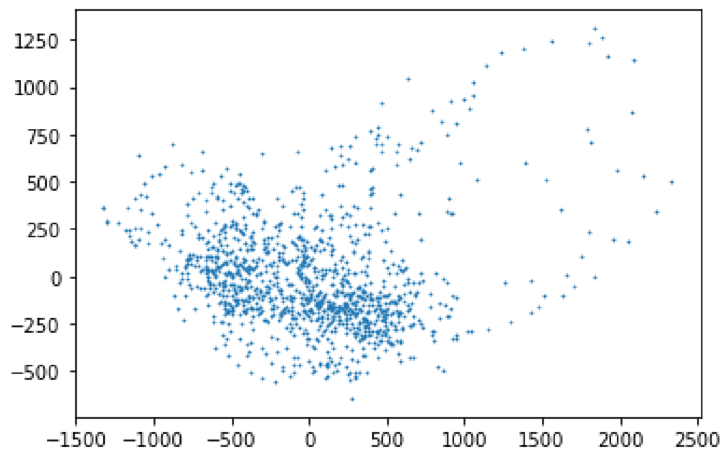


FIGURE 4.9: All current fields represented by the first two principal components.

We can conclude that we can greatly reduce the dimensionality of the current data while still being able to retain most of the information by using principal component analysis. We will use the reduced datasets as output to train our neural network on in the next chapter. We also know that we are capable to transform the reduced data back to a current field that very much represents the actual field.

Chapter 5

Using Neural Networks to Predict Change in the Current Field

In this chapter we describe how we trained a neural network on our dimensionality reduced data i.e. the original data reduced to its principle components by the PCA we described in Chapters 3 and 4. We start by giving some general background knowledge about how neural networks operate. Then we point out the differences between different types of neural networks. After that we specify the different setups we will use. In the end we discuss the results.

The goal of this research can be summarized by finding a function $f(\boldsymbol{x}) = \boldsymbol{y}$, where $\boldsymbol{x}$ is wind data and $\boldsymbol{y}$ is current data, which is the data that we want to predict. We do already have 1200 of these wind and current data pairs: $\boldsymbol{x_1}, \ldots, \boldsymbol{x_{1200}}$ and $\boldsymbol{y_1}, \ldots, \boldsymbol{y_{1200}}$. The function f that returns $\boldsymbol{y_n}$ when the input is $\boldsymbol{x_n}$ we do not have yet. If we have this function f then given a new wind field x_{new}, we can return $f(\boldsymbol{x}_{\text{new}}) = \boldsymbol{y}_{\text{new}}$, which is the predicted current field, based on this new wind data. To find this function we will train a neural network on our data.

An artificial neural network, or neural network for short, is a supervised learning algorithm that learns a function $f(\cdot) : R^m \longrightarrow R^n$. In our case, this function has as input the reduced wind fields and as output the reduced water fields. The name 'neural network' stems from the way the algorithm works, as it attempts to mimic the way the human nervous system works. A neural network is a structure of nodes and connections between those nodes. The nodes in the network are able to receive input signals, to process them and to send output signals (Zhang, 2018). By doing this, neural networks are capable of learning complicated nonlinear relationships from sets of training examples. The nodes can be thought of as neurons, and the connections between the nodes can be thought of as synapses. Hence the comparison with the human nervous system. Figure 5.1 gives an example of the structure of a small neural network. As can be seen, the network consists of three 'types' of layers. The first layer is called the input layer. These neurons represent the input we provide for the network: the wind field. The final layer is the output layer. These neurons represent the output we provide for the network: the water field. The layers in between are called the hidden layers.

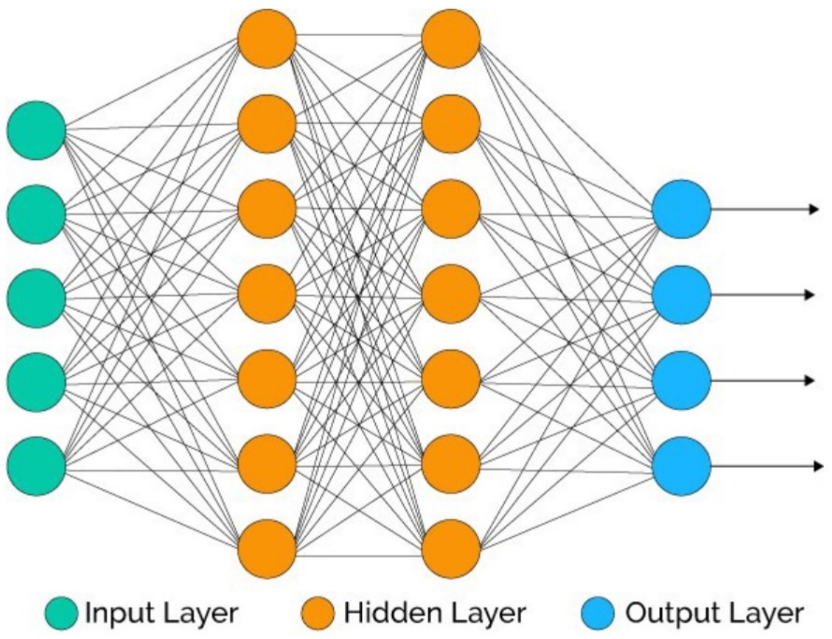


FIGURE 5.1: Structure of a neural network that contains 2 hidden layers.

5.1 The Method

In short a neural network works as follows: The network can be best described from left to right, keep in mind the example network shown in Figure 5.1. The input nodes, which are the nodes of the first layer or the 'input layer', contain the input we provide for our network. These inputs are then sent to all the nodes in the next layer via the connections. As is also shown in Figure 5.1, every node is connected to every other node in the next layer. The number of nodes in each hidden layer, as well as the number of hidden layers can be changed to improve the performance of the model. These connections adjust the values when sending them to the next node by multiplying them by a weight. These new weighted values are then summed and a bias is added as well. This results in one value, which is the input of the node in the next layer. For the next part, note that we use the notation x_i for just one value of the input vector, as opposed to $\boldsymbol{x_i}$ which we use for the entire sample vector. In the example shown in Figure 5.1, where there are 5 input nodes, the input of a node in the second layer, or the first hidden layer would be: $V = w_1 x_{i_1} + w_2 x_2 + w_3 x_3 + w_4 x_4 + w_5 x_5 + bias$. Where w_i is the weight given to the input value x_i. The node receiving this value contains an 'activation function' f. This function again adjusts the received value to a new value $f(V)$. This process is shown in Figure 5.2.

The newly created value from the node is again sent on via connections to the next layer of nodes, again adjusting its value when sending it to these connected nodes. This process repeats itself until the values arrive at the output layer. The weights and biases used in the connections are the parameters of the network. These parameters are estimated during the training of the network using a technique called backpropagation. An elaborate explanation on how this technique works can be found in Hecht-Nielsen, 1992. The idea behind the technique is the following: We start by simply choosing random values for the parameters. We then use our training data to see how well the network performs using these randomly chosen parameters

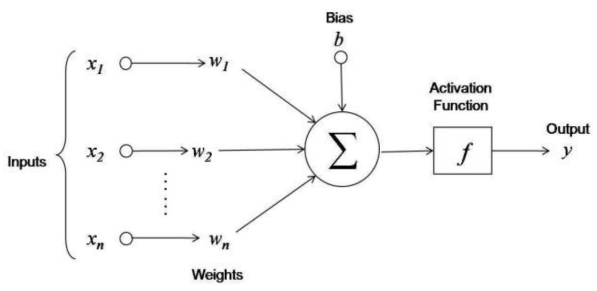


FIGURE 5.2: How the input data is adjusted by the connections and a node in the next layer.

by comparing the output of the network, i.e. the *predicted* values with our actual output i.e. the *observed* values. We have seen already, that the output of the network is a result of the function $f(\boldsymbol{x})$, where $\boldsymbol{x}$ is the input and $f(\cdot)$ represents the result of all operations on the input. As mentioned earlier these operations consist of applying weights and biases as well as applying activation functions on which we will elaborate more later on. The predicted values, corresponding to the input values we feed to the network are then compared with the observed values. The difference between the predicted and the observed values is measured by the sum of the squared residuals. The value of the sum of the squared residuals, or SSR for short can be written down as a function:

$$\mathrm{SSR}(\boldsymbol{p}) = \sum_{i=1}^{k} |\mathrm{Observed}_i - \mathrm{Predicted}_i(\boldsymbol{p})|^2$$

where k is the number of observed data points used for training the neural network and $\boldsymbol{p}$ is the vector containing the parameters of the model. This is our cost function $c(\boldsymbol{p})$, which tells us the 'cost' of the difference between the predicted value and the actual value. The input of this function are all the parameters of the model i.e. the weights and biases, while the output is a single value.

5.1.1 Weights and Biases

Since we want to optimize the performance of the network, we would like to find the minimum of the cost function, by changing the weights and biases. To update our initial estimates and get improved values for the parameters, we apply an iterative technique called *gradient descent* which is described in Hecht-Nielsen, 1992. This technique uses the partial derivatives of the SSR with respect to the different parameters to improve the the value of c. This process is repeated a number of times until a certain criterion is reached, usually the criterion is that the improvements per iteration are smaller than a certain predetermined value. The technique works in a way where it can update the values of all parameters in one iteration. When working with a large set of training data, *stochastic gradient descent* is a technique that might be preferred. For this technique only a randomly selected sample of the data is used to estimate the gradient instead of the total set of the data. By selecting only a sample set, we are able to speed up the process significantly. Now that we have discussed the weights and biases of the network, in the next section we will address the activation function of the nodes.

There are some issues that may occur during the optimization process. One of the issues is, that instead of finding the global minimum of the cost function i.e. the optimum, we end up in a local minimum. To increase the chance of finding the global minimum we should optimize the cost function multiple times, each time starting with differently chosen parameters. Another issue that could occur is when the convergence to the minimum goes so slow that the improvements per iteration are smaller than our criterion causing the algorithm to stop. This can be prevented by lowering this criterion.

5.1.2 Activation function

An *activation function* is a function that transforms an input signal into an output signal. When a value is received by a hidden node in the network, it is used as an input for the activation function of this node. The outcome of this function is the output of the node which is then sent forward via its connections to the next layer in the network. The function derives its name from its use in the past, where a node would be 'activated' or not. If the activation function would output a value of 1, the node would be activated, if the output would be 0, the node would not be activated. The 1 or 0 value can be a good choice when using binary classifiers: is this a cat yes or no?
For binary classifiers usually a threshold function would be used as activation function:

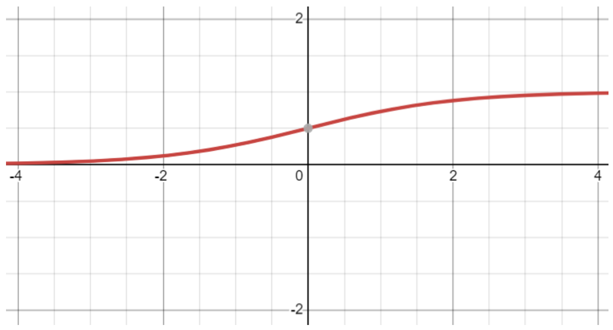
$$f(\mathrm{u}) = 1 \text{ if } \mathrm{u} > a; 0 \text{ otherwise}$$

where a would be the threshold.
For nontrivial problems however, more complex activation functions are used. Without the use of activation functions, a neural network would simply be a polynomial function of degree 1. The complexity of such a function is limited. To solve more complex, nonlinear problems activation functions are required.

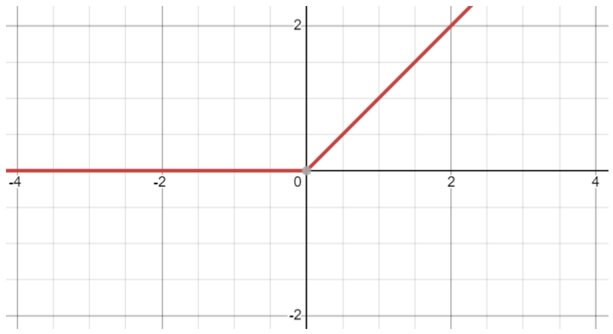
We will discuss the three most commonly used activation functions in this section, which are:

- Sigmoid; $f(x) = \frac{1}{1+e^{-x}}$
- Rectified linear unit (ReLu); $f(x) = max\{0, x\}$
- Hyperbolic Tangent: $tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$

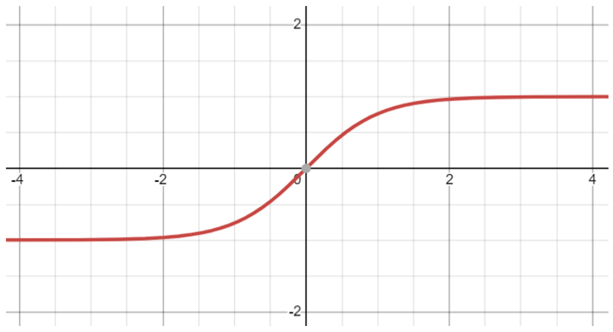
A plot of these functions can be seen in Figure 5.3 More details about these activation functions are described in Sharma, Sharma, and Athaiya, 2017. QUOTE: "There is no thumb rule for selecting any activation function but the choice of activation function is context dependent, i.e it depends on the task that is to be accomplished. Different Activation Functions have both advantages and disadvantages of their own and it depends on the type of system that we are designing."



(A) Sigmoid function: $f(x) = \frac{1}{1+e^{-x}}$



(B) ReLu function: $f(x) = max\{0, x\}$



(C) Hyperbolic tangent: $tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$

FIGURE 5.3: Activation functions.

5.2 Training the Neural Network

5.2.1 Setups

The neural network we use to predict the current fields is a Multi-layer perceptron regressor. This is the type of artificial neural network as is described earlier in this chapter. In order to get the best model, we try out several setups to see which choices have a positive impact on the performance of the network.
We will vary over the following things:

- The Input Data; For the input data we will experiment with the wind fields that are reduced in dimensionality as a result of principal component analysis, as described in Chapter 3. We will use the reduced fields of the race area itself, which are 2-dimensional (Wind Race Area). We will use the reduced wind fields of the olympic area, which are 6-dimensional (Wind Olympic Area). And finally we will also use the reduced fields of the complete area, which consists are 25-dimensional (Wind Complete Area).
- The activation function; We will test the results of the network using the ReLu function as well as the hyperbolic tangent function and the sigmoid function.
- The Hidden Layers/Nodes; To get a good result we will try out different combinations of number of hidden layers and number of nodes per hidden layer.

In Panchal et al., 2011 some rule-of-thumb methods are mentioned for determining the number of hidden nodes in a network:

- The number of hidden neurons should be between the size of the input layer and the size of the output layer
- The number of hidden neurons should be 2/3 the size of the input layer, plus the size of the output layer
- The number of hidden neurons should be less than twice the size of the input layer

However, Panchal et al., 2011 mentions, ultimately the selection of the architecture for a neural network comes down to trial and error. Usually, by increasing the number of hidden neurons and or the number of hidden layers the performance of the model will increase. However, this is not granted and over complicating can lead to over fitting.

We will try to find the best model in the following way:

1. We start training models using the different input data and activation functions we described earlier in the chapter.
2. To determine the number of hidden nodes in the network, we start with one layer, and a number of nodes in the layer based on the rules of thumb we just described.
3. We then increase the number of hidden neurons and check for improvement.
4. If the performance of the model does not improve, we add an extra hidden layer.

5. In practice, a more trial and error approach is used, where we improve the model based on the results by earlier versions of the model.

If in the end we find a number of models that perform similarly, we will use

5.2.2 Scoring Method

To judge how well the neural network performs we use the *coefficient of determination* of the prediction: R^2 as a measure of performance. This coefficient is defined as:

$$R^2 = \left(1 - \frac{SS_{\text{res}}}{SS_{\text{tot}}}\right)$$

where SS_{res} is the residual sum of squares, and SS_{tot} is the total sum of squares:

$$SS_{\text{res}}(\boldsymbol{p}) = \sum_{i=1}^{m}(||\text{observation}_i - \text{prediction}_i(\boldsymbol{p})||^2)$$
$$SS_{\text{tot}} = \sum_{i=1}^{m}(||\text{observation}_i - \text{mean of observations 1 to } m||^2)$$

Here we test on m observations. The best possible score equals 1, which occurs when SS_{res} is 0 i.e. the predictions are 100% accurate. A baseline model that will always predict the mean of the observations would get a score of 0. It is also possible to get a negative score for R^2, in case the model performs worse than the baseline model.

Cross-Validation.

Since we only possess 1200 data points, we will construct the models using all these data points. To evaluate the performance of these models, we would actually like to see how the model performs on new data i.e. data on which the model was not trained. We can estimate the performance of the model by using a technique called leave one out cross-validation. The idea behind this is as follows:
Instead of training the model on the complete set of the 1200 data points, we leave out one point, and train the model on the remaining 1199. This way the model will be very similar to the case in which the model would be trained on all data, but we can test how it performs on unseen data, namely the one data point that we left out. We can then repeat this process while leaving out another data point. In fact, we can repeat this process 1200 times, each time leaving out a different data point in the training set, on which we will then test its performance. For every repetition of this process we train a model that very well represents the actual model, since we only leave out 1 data point in the training set, and we can see how it performs on 'new' data. By averaging out the performance value over all 1200 repetitions, we can get a good idea of how accurate our model is.

This is the theoretical idea behind cross-validation. In reality, training those 1200 models takes a long time, so instead of using 1199 points to train the model, we could also use less data points as long as the model still represents the performance of the actual model well. In practice I have chosen to train 10 models by splitting up the data into 10 disjoint subsets. This is called 10-fold cross-validation. Each of the 10 models is then trained on 9 of these subsets, containing a total of 1080 datapoints, and then tested on the remaining 120. By taking the average of the performance of these 10 models, a good indication of the performance of the actual model is achieved.

5.3 Results

In this section we provide the scores for the predictive performance of the models we have trained. These scores are based on the R^2 value as described in the previous section. By using cross-validation we make sure that the values in this section are good representations of the true performance of the neural networks we train. The results shown in this section are the average of the R^2 values using the cross validation as described in the previous section.

To get an idea of how we should improve the model by adjusting the number of hidden nodes and layers, we start off by training neural networks for all three activation functions using the same number of hidden neurons and hidden layers. These numbers are the following:

Layer 1	Layer 2	Layer 3
10	0	0
25	0	0
50	0	0
100	0	0
250	0	0
100	100	0
250	250	0
100	100	100
250	250	250

TABLE 5.1: Setups for number of hidden nodes and hidden layers in initial neural networks

These numbers were initially chosen to get an understanding of how the performance of the model would improve by increasing the number of nodes and the number of layers.

As already mentioned at the start of Chapter 5, the input for the neural networks are the dimensionality reduced wind fields, i.e. principal components, over the respective areas (race area, olympic area, complete area). The outputs of the network are the dimensionality reduced current fields in the respective race areas. Let's take the dimensionality reduced wind fields for the race area as an example. In Chapter 3, we have seen that we can reduce the wind fields over these race areas to a two dimensional vector that still represents the original wind data in a good way. Thus to train a neural network on this data, the input of the network consists of two nodes. In Chapter 4 we have seen that the current fields in the race area can be reduced by applying PCA, to a 10-dimensional vector. This means that our neural network will be a function $f(\cdot) : R^2 \longrightarrow R^10$. If we use the olympic area as input the function will be a function $f(\cdot) : R^6 \longrightarrow R^10$ and if we use the complete area as input the function will be a function $f(\cdot) : R^{25} \longrightarrow R^10$.

The scores for the models trained on these setups of hidden nodes and hidden layers are shown in Section 5.3.1. Based on these results we have an idea of how we can improve the models to perform even better. The results of these improved models are discussed in Section 5.3.2

5.3.1 Initial Results

In the first type we use only one hidden layer, while in the second type we will use two hidden layers. Within each type we vary the input and number of hidden neurons to find out what should be the input and how many hidden neurons should be used. For all models we use stochastic gradient descent to train the model.

Recall that the neural network is trained on the principal components of the data, i.e. the inputs are the principle components of the wind fields and the outputs are the principle components of the corresponding current fields, hence the scores apply to the principal components and not the complete data sets. What follows are the 10-fold cross-validated scores as described before.

ReLu Activation Function

The performance of the models using the ReLu activation function are shown in Table 5.2.

#Hidden Neurons	Race Area	Olympic Area	Complete Area
10	0.245	0.222	0.235
25	0.250	0.234	0.292
50	0.249	0.226	0.313
100	0.254	0.244	0.311
250	0.252	0.247	0.291
{100,100}	0.263	0.212	0.345
{250,250}	0.268	0.207	0.420
{100,100,100}	0.229	0.226	0.391
{250,250,250}	0.267	0.163	0.482

TABLE 5.2: Coefficients of determination for using the ReLu activation function. Three different input data using various number of hidden neurons.

Hyperbolic Tangent Activation Function

The performance of the models using the Hyperbolic Tangent activation function are shown in Table 5.3

Sigmoid Activation Function

The performance of the models using the Hyperbolic Tangent activation function are shown in Table 5.4

#Hidden Neurons	Race Area	Olympic Area	Complete Area
10	0.103	0.107	0.088
25	0.182	0.165	0.147
50	0.199	0.198	0.187
100	0.217	0.232	0.231
250	0.250	0.394	0.293
{100,100}	0.233	0.256	0.189
{250,250}	0.236	0.474	0.251
{100,100,100}	0.187	0.305	0.201
{250,250,250}	0.244	0.334	0.267

TABLE 5.3: Coefficients of determination for using the hyperbolic tangent activation function. Three different input data using various number of hidden neurons.

#Hidden Neurons	Race Area	Olympic Area	Complete Area
10	0.059	0.048	0.054
25	0.125	0.103	0.097
50	0.180	0.149	0.164
100	0.244	0.175	0.223
250	0.242	0.198	0.277
{100,100}	0.177	0.164	0.186
{250,250}	0.211	0.201	0.315
{100,100,100}	0.173	0.129	0.163
{250,250,250}	0.219	0.145	0.303

TABLE 5.4: Coefficients of determination for using the Sigmoid activation function. Three different input data using various number of hidden neurons.

5.3.2 Final Models

Based on the results of the models with the standard number of nodes and layers, as shown in Figure 5.1, some important observations can be made. First of all, it is clear from all models that the wind describing the Olympic Area does not serve as a better input than the 2-dimensional Race Area data. The performance of the models is actually worse when using the Olympic Area as input for all different setups, as can be seen in Tables 5.2 5.3 and 5.4. It is also clear that using the wind from the Complete Area is the best input for all models. Further more do we observe the best results for the chosen numbers of neurons using the ReLu activation function. Therefore we have chosen to focus on models that use the ReLu activation and use the wind fields of the Complete Area as input. The initial results also indicate that both increasing the number of hidden neurons as well as the hidden layers improves the performance of the model. For these reasons the final models we will test consist of the numbers of hidden neurons and layers are shown in Table 5.5.

Results

The results of the models trained on the neural networks with the structures described in Table 5.5 can be found in Table 5.6.
We can conclude from Table 5.6 that the model using 3 hidden layers containing 1000 hidden neurons each performs the best out of the final models. We know that the

Layer 1	Layer 2	Layer 3	Layer 4	Layer 5
500	500	500		
500	500	500	500	
500	500	500	500	500
1000	1000	1000		
1000	1000	1000	1000	
1000	1000	1000	1000	1000

TABLE 5.5: Setups for number of hidden nodes and hidden layers in initial neural networks

Layer 1	Layer 2	Layer 3	Layer 4	Layer 5	Coefficient of Determination
500	500	500			0.586
500	500	500	500		0.605
500	500	500	500	500	0.607
1000	1000	1000			0.640
1000	1000	1000	1000		0.635
1000	1000	1000	1000	1000	0.625

TABLE 5.6: Performances of the models using ReLu activation function and Complete Area wind input. The Layers describe the number of hidden nodes used in the respective models.

model has predictive qualities since the coefficient of determination is 0.64.

To get an intuitive feeling of the results of the model, Figure 5.4 shows 5 randomly selected current fields that are predicted by the model based on their respective reduced wind fields of the complete area, compared to the actual current fields belonging to those wind fields. To make this prediction more valuable, we trained the model on all samples, except the 5 samples we predict here, so that we indeed find out how well the model performs on 'new' data, i.e. data on which the model has not been trained. Recall that the neural network predicts the principal components of a water current field, hence we should also transform this predicted data back to the original space of the water current data, using the loadings of the principal components.

To quantify the accuracy in a different way than the coefficient of determination, we will also again look at the relative magnitude of the delta vector, as we have previously done in Chapter 4. In this case we will compare the predicted current data, which is transformed back to the original space, with the original current data. These are the main statistics for using the relative magnitude:

	Relative Magnitude of Delta Vector
Maximum	0.2568
Minimum	0.0006
Mean	0.0197
Median	0.0083
Standard Deviation	0.0318

We can see that the average 'relative error' is 1.97% and the median relative error is 0.83%.

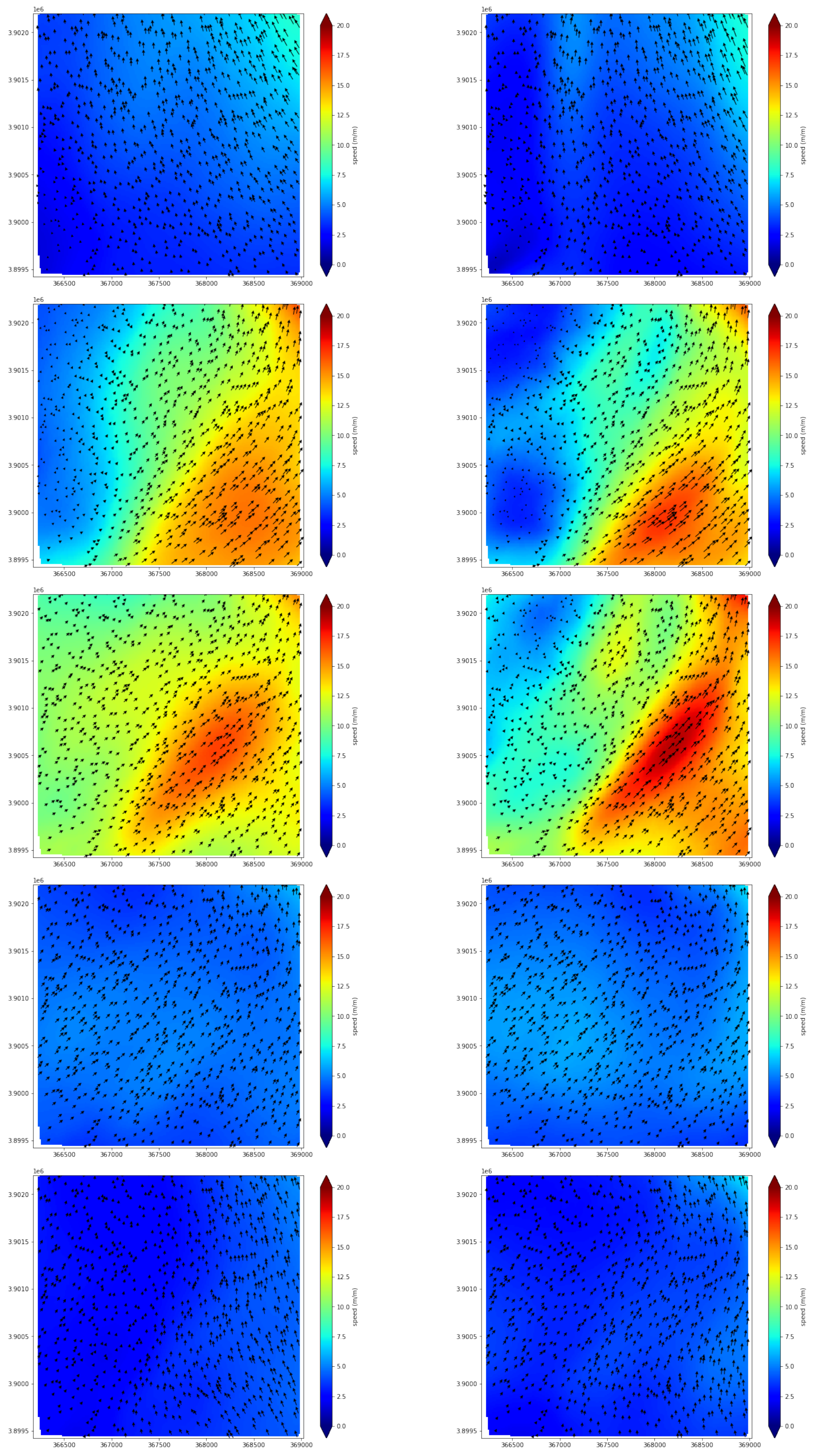


FIGURE 5.4: Predictions of our model for randomly selected water current fields on the left, vs the actual water current fields on the right.

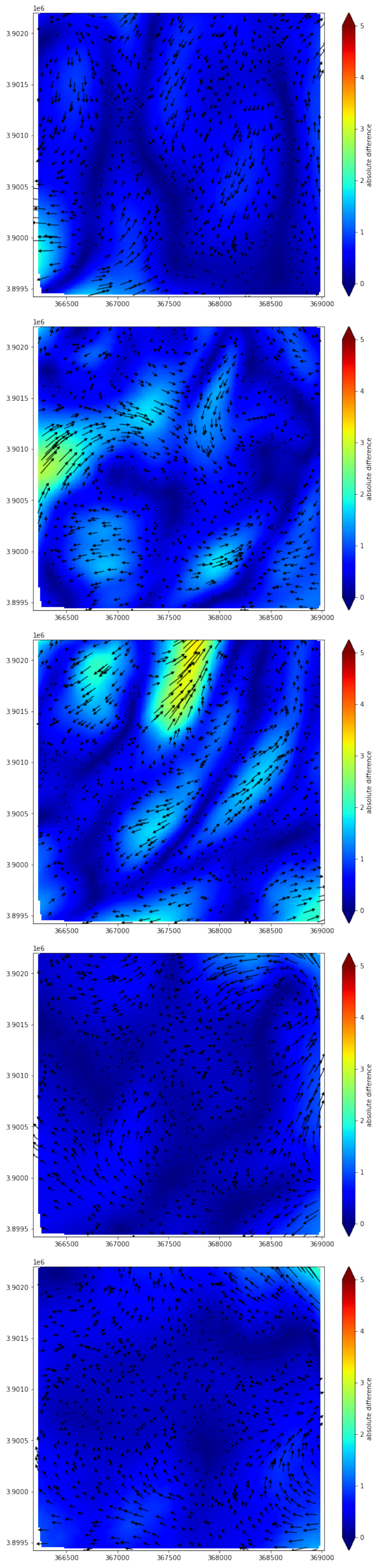


FIGURE 5.5: Visualisation of where the predicted field differs from the actual field for water current fields of Figure 5.4

5.4 Without the use of PCA

To see whether using PCA is justified, the idea was for this section to compare the performance of our model with a model that we trained on the wind and current data without using PCA.

We will not go over all the different variations concerning the number of hidden neurons or the activation function. Instead we will compare the best performing model that we tested, using the best input: the neural network using the ReLu activation function with (1000,1000,1000) hidden neurons i.e. the network with three hidden layers that each consist of 1000 neurons. The input will be the wind field describing the complete bay.
Even though the structure of the hidden layers remains the same, the input layer and the output layer are different in size from when we use principal components. Using the principal components the input layer consisted of 25 neurons and the output layer of only 10. Using the non-reduced data the size of the input layer will be 161800 and the length of the output layer will be 12900.

Unfortunately, while the model using PCA took 22 minutes to be trained, the model using the complete, non-reduced, data was not able to be completed as it ran into timeout errors. Perhaps using more powerful computers the model could be trained, but for this research we were unable to compare the two.
We can however compare the number of weights and biases between the two different choices:

- PCA: (25*1000 + 1000) + (1000*1000 + 1000) + (1000*1000 + 1000) + (1000*10 + 10) = 2038010
- No PCA: (161800*1000 + 1000) + (1000*1000 + 1000) + (1000*1000 + 1000) + (1000*12900 + 12900) = 176715900

This means that without using PCA the model contains 176700000/2035000 $\approx$ 87 as many parameters. Since every parameter needs to be optimized, fitting the model will be significantly more complicated without using PCA.

Chapter 6

Conclusion

From the research we can conclude that it is possible to make meaningful predictions about water current based on wind data using a neural network. Even when using big datasets, these predictions can be made rather fast, by using principal component analysis to reduce the dimensionality of your data, while still delivering promising results.

The results show that using a rectified linear unit (ReLu) as activation function, will provide the most accurate predictions. This is the case in all the race areas, however it is too early to conclude that this would also be the case for other datasets that represent wind and water fields in other places in the world. As can also be observed from Chapter 5 and Appendix A, the model does not perform equally well on all different race areas, even though it is trained specifically on the areas own data, using the same setups for all of them.

The model that we trained that delivered the highest accuracy, made use of three hidden layers of neurons, all consisting of 1000 nodes. However, after more trial and error a different combination of layers and nodes could possibly perform better. Using a more powerful computer, these combinations can be tested to train new models.
In general, to pick these parameters, there are two main things that should be taken into account:

- Overfitting
- Complexity versus accuracy

By overfitting, one could find a very specific combination of layers and nodes that performs very well on the given dataset, but will perform worse than average on other datasets. As can be seen from the results in 5 as well as the results in Appendix A, there is not one model configuration that is clearly superior. It can also be observed that in different race areas, different model configurations are preferred. Hence the required generality of the model should be considered when constructing the model.

Something else that is observed from the results in Appendix A, is that the accuracy is negatively correlated with the number of principal components used for the current data i.e. the more principal components used to describe the current field the lower the accuracy of the model.

As mentioned earlier already, in general, adding more nodes will improve the predictive quality of the model. However, one should consider whether the added complexity of the model, is worth the improved performance. By complicating the model, it will take longer to be trained, and will take longer to predict. Depending on the computing power and the relation between added nodes and improved performance, one should find a balance in this.

Another conclusion we can draw, is that using a larger area for the wind, boosts the performance of the model significantly. Intuitively, it also makes sense that not only the wind directly above the water area has influence, but also the wind around the area as this also influences the water around the area. Hence, even though the wind directly above the area might be the same for two data fields, the current can still differ significantly.

In the end, the model can definitely be applied to predict water currents, once it is properly trained on and configured for the underlying data set. This means that most likely, for every data set, a different number of layers and nodes, and possibly even a different activation function should be used to achieve optimal results. And those optimal results will also vary in how ‘optimal’ they are for every data set. Therefore one should be very aware of the accuracy of the model, when using it for predicting.

Appendix A

Figures related to other race areas.

A.1 Enoshima

In this section we highlight the results of training our model on the data of the Enoshima race area. This is the area with the lowest number of gridpoints, which also helps to explain that with only 7 principal components, we already capture 99% of the variance of the current fields.

Enoshima might have some of the most complex surroundings, as it is located next to an island. Because of this the variance of the wind velocity, which can be seen in Figure A.3, was slightly higher than in the other areas. However, this did not impact the performance of the model, as its accuracy scored the second highest for Enoshima, out of all the other race areas.

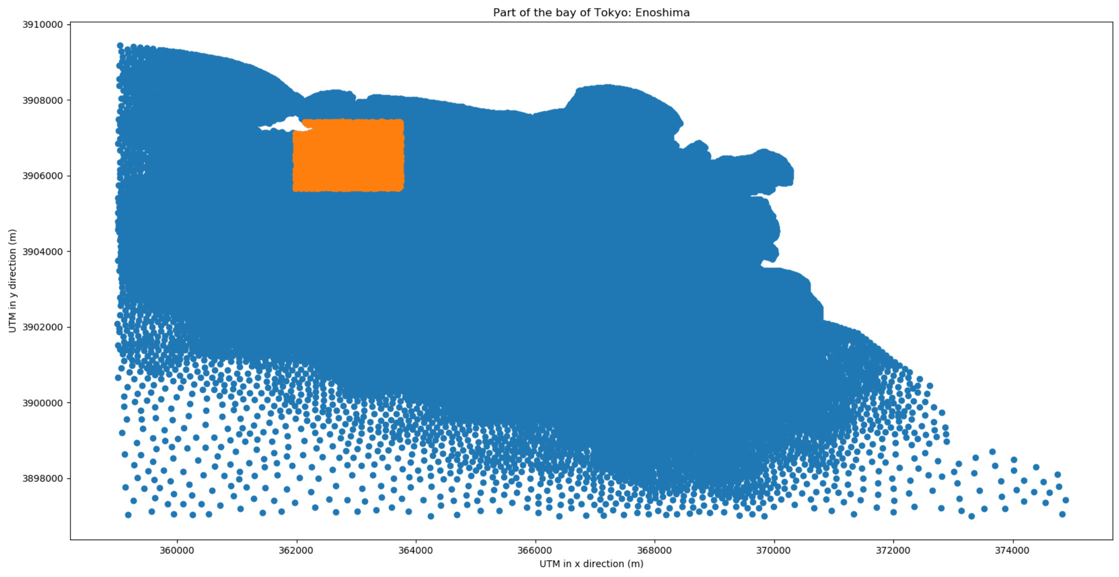


FIGURE A.1: Grid points of race area Enoshima highlighted

Principal components required to capture >99% of the variance:

- Wind: 2 Principal Components
- Water: 7 Principal Components

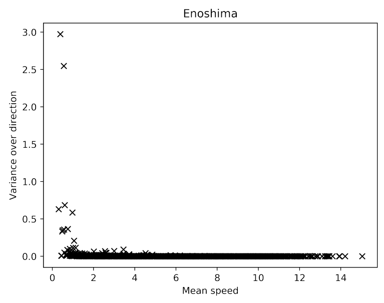


FIGURE A.2: Variance in direction plotted against mean velocity of wind fields.

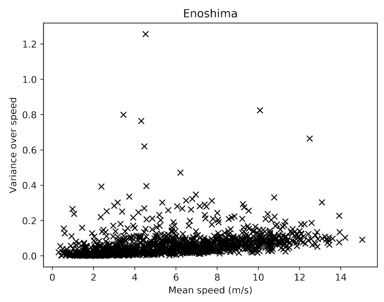


FIGURE A.3: Variance in velocity plotted against mean velocity of wind fields.

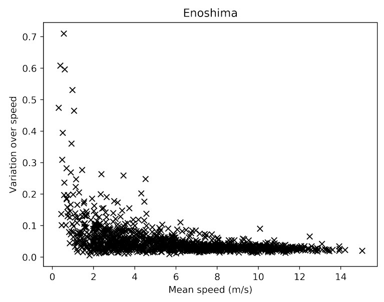


FIGURE A.4: Relative variance in velocity plotted against mean velocity of wind fields.

#Hidden Neurons	Hyperbolic Tangent	Sigmoid	ReLu
10	0.198	0.117	0.307
25	0.273	0.222	0.310
50	0.301	0.298	0.308
100	0.319	0.329	0.308
250	0.323	0.324	0.312
{100,100}	0.303	0.285	0.331
{250,250}	0.295	0.288	0.330
{100,100,100}	0.256	0.244	0.325
{250,250,250}	0.254	0.278	0.320

TABLE A.1: Coefficients of determination for using the principal components of the wind field of the race area, using various numbers of hidden neurons.

#Hidden Neurons	Hyperbolic Tangent	Sigmoid	ReLu
10	0.151	0.098	0.327
25	0.226	0.183	0.350
50	0.268	0.246	0.355
100	0.305	0.276	0.379
250	0.354	0.332	0.399
{100,100}	0.269	0.296	0.425
{250,250}	0.328	0.401	0.506
{100,100,100}	0.262	0.279	0.479
{250,250,250}	0.304	0.374	0.569

TABLE A.2: Coefficients of determination for using the principal components of the wind field of complete bay, using various numbers of hidden neurons.

#Hidden Neurons	
500	0.420
1000	0.438
{500,500}	0.562
{1000,1000}	0.611
{500,500,500}	0.602
{1000,1000,1000}	0.624

TABLE A.3: Coefficients of determination for using the principal components of the wind field of complete bay, using the ReLu activation function for a various number of hidden neurons.

A.2 Fujisawa

In this section we highlight the results of training our model on the data of the Fujisawa race area. This area, together with the Hayama and Sagami race areas, contains the most grid points. This helps explain why Fujisawa required the most number of principal components to capture 99% of the variance of the data. This is the most out of all the race areas.

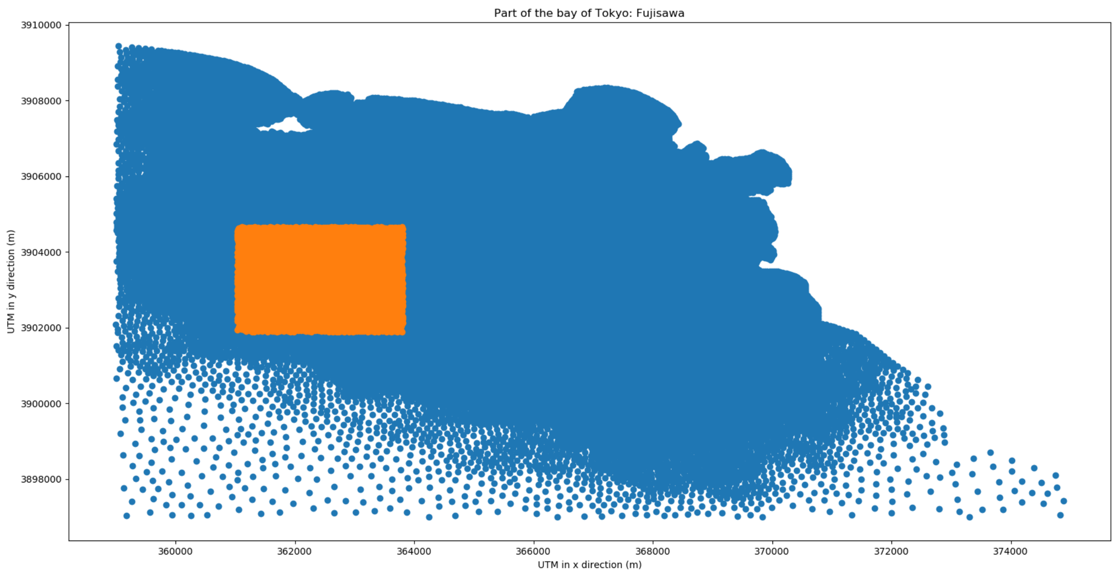


FIGURE A.5: Grid points of race area Fujisawa highlighted

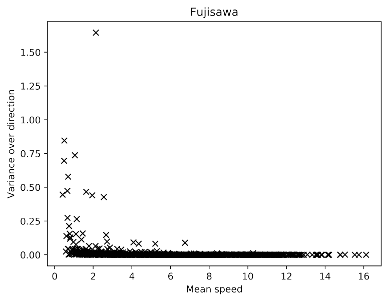


FIGURE A.6: Variance in direction plotted against mean velocity of wind fields.

Principal components required to capture >99% of the variance:

- Wind: 2 Principal Components
- Water: 13 Principal Components

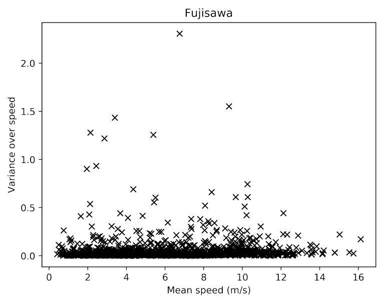


FIGURE A.7: Variance in velocity plotted against mean velocity of wind fields.

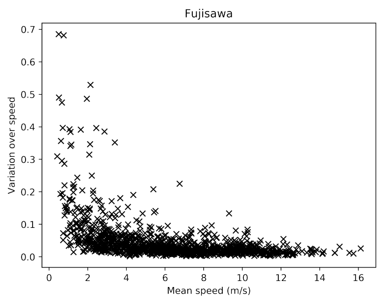


FIGURE A.8: Relative variance in velocity plotted against mean velocity of wind fields.

#Hidden Neurons	Hyperbolic Tangent	Sigmoid	ReLu
10	0.071	0.047	0.145
25	0.107	0.093	0.162
50	0.129	0.134	0.166
100	0.134	0.145	0.162
250	0.159	0.148	0.166
{100,100}	0.119	0.130	0.169
{250,250}	0.172	0.136	0.176
{100,100,100}	0.113	0.119	0.146
{250,250,250}	0.138	0.130	0.151

TABLE A.4: Coefficients of determination for using the principal components of the wind field of the race area, using various numbers of hidden neurons.

#Hidden Neurons	Hyperbolic Tangent	Sigmoid	ReLu
10	0.069	0.034	0.156
25	0.099	0.078	0.194
50	0.118	0.108	0.189
100	0.142	0.140	0.173
250	0.185	0.171	0.135
{100,100}	0.119	0.119	0.211
{250,250}	0.168	0.201	0.272
{100,100,100}	0.114	0.102	0.238
{250,250,250}	0.156	0.208	0.391

TABLE A.5: Coefficients of determination for using the principal components of the wind field of complete bay, using various numbers of hidden neurons.

#Hidden Neurons	
500	0.177
1000	0.233
{500,500}	0.380
{1000,1000}	0.457
{500,500,500}	0.469
{1000,1000,1000}	0.561

TABLE A.6: Coefficients of determination for using the principal components of the wind field of complete bay, using the ReLu activation function for a various number of hidden neurons.

A.3 Kamakura

In this section we highlight the results of training our model on the data of the Kamakura race area. This area, together with the Enoshima area, is one of the race areas with the lowest amount of grid points. In order to capture more than 99% of the variance of the data, 8 principal components were required. The model scored the highest accuracy on this race area. The above average constancy of the wind could have played a role in this.

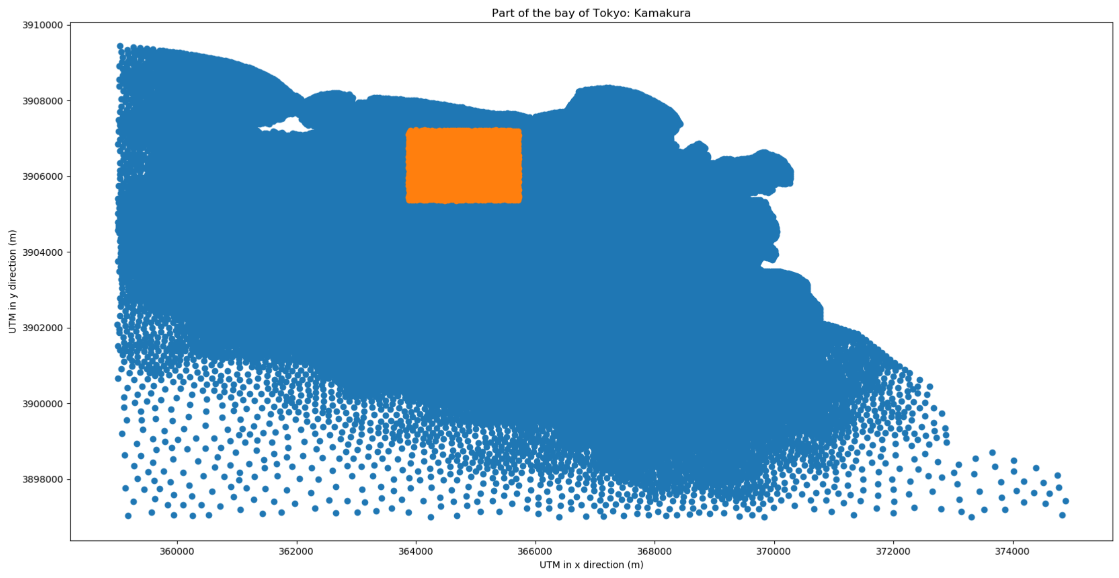


FIGURE A.9: Grid points of race area Kamakura highlighted

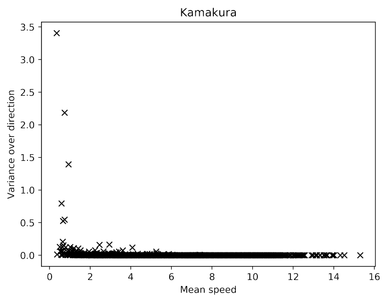


FIGURE A.10: Variance in direction plotted against mean velocity of wind fields.

Principal components required to capture >99% of the variance:

- Wind: 2 Principal Components
- Water: 8 Principal Components

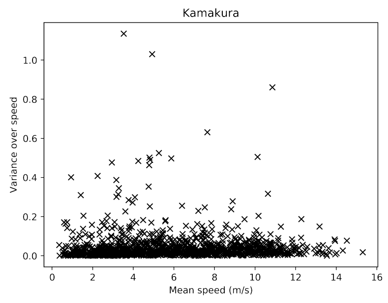


FIGURE A.11: Variance in velocity plotted against mean velocity of wind fields.

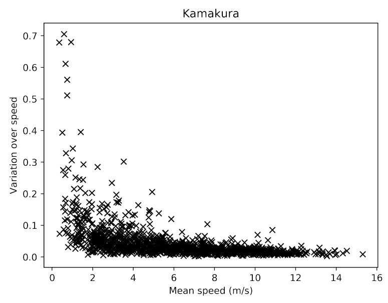


FIGURE A.12: Relative variance in velocity plotted against mean velocity of wind fields.

#Hidden Neurons	Hyperbolic Tangent	Sigmoid	ReLu
10	0.205	0.146	0.295
25	0.264	0.231	0.279
50	0.286	0.295	0.288
100	0.306	0.310	0.319
250	0.300	0.311	0.326
{100,100}	0.293	0.291	0.304
{250,250}	0.319	0.300	0.309
{100,100,100}	0.233	0.262	0.298
{250,250,250}	0.252	0.275	0.277

TABLE A.7: Coefficients of determination for using the principal components of the wind field of the race area, using various numbers of hidden neurons.

#Hidden Neurons	Hyperbolic Tangent	Sigmoid	ReLu
10	0.159	0.136	0.285
25	0.236	0.215	0.354
50	0.273	0.260	0.386
100	0.311	0.292	0.373
250	0.368	0.352	0.399
{100,100}	0.294	0.325	0.428
{250,250}	0.366	0.442	0.508
{100,100,100}	0.283	0.276	0.481
{250,250,250}	0.323	0.380	0.574

TABLE A.8: Coefficients of determination for using the principal components of the wind field of complete bay, using various numbers of hidden neurons.

#Hidden Neurons	
500	0.424
1000	0.481
{500,500}	0.562
{1000,1000}	0.625
{500,500,500}	0.621
{1000,1000,1000}	0.663

TABLE A.9: Coefficients of determination for using the principal components of the wind field of complete bay, using the ReLu activation function for a various number of hidden neurons.

A.4 Sagami

In this section we highlight the results of training our model on the data of the Sagami race area. This area is very similar to the Fujisawa area in many ways. They are located very close to each other, both relatively far from the land and they have around the same number of grid points. This could explain why the performance of the model is so similar on both these areas.

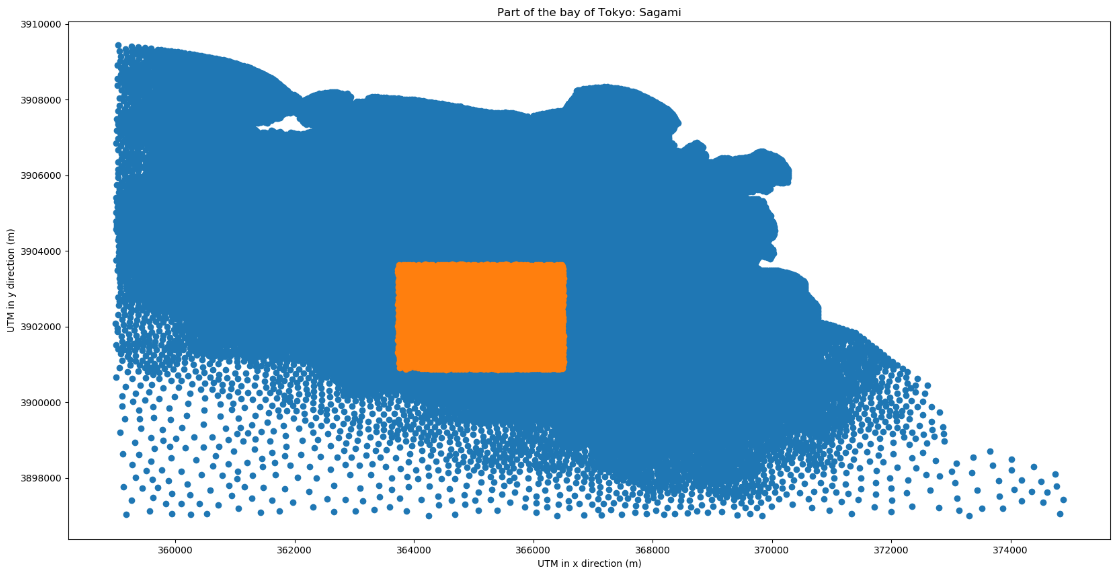


FIGURE A.13: Grid points of race area Sagami highlighted

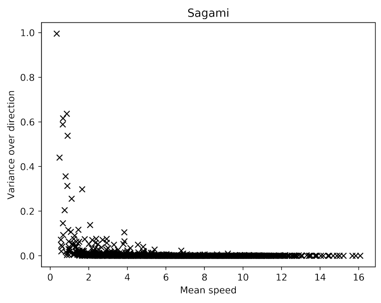


FIGURE A.14: Variance in direction plotted against mean velocity of wind fields.

Principal components required to capture >99% of the variance:

- Wind: 2 Principal Components
- Water: 12 Principal Components

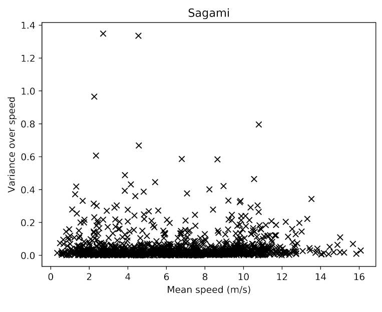


FIGURE A.15: Variance in velocity plotted against mean velocity of wind fields.

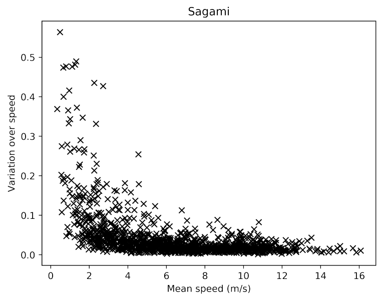


FIGURE A.16: Relative variance in velocity plotted against mean velocity of wind fields.

#Hidden Neurons	Hyperbolic Tangent	Sigmoid	ReLu
10	0.086	0.054	0.165
25	0.122	0.108	0.183
50	0.150	0.145	0.177
100	0.150	0.162	0.183
250	0.165	0.169	0.184
{100,100}	0.136	0.135	0.160
{250,250}	0.171	0.141	0.191
{100,100,100}	0.123	0.121	0.160
{250,250,250}	0.134	0.142	0.162

TABLE A.10: Coefficients of determination for using the principal components of the wind field of the race area, using various numbers of hidden neurons.

#Hidden Neurons	Hyperbolic Tangent	Sigmoid	ReLu
10	0.065	0.045	0.170
25	0.098	0.082	0.183
50	0.130	0.124	0.204
100	0.162	0.156	0.199
250	0.202	0.193	0.210
{100,100}	0.137	0.136	0.234
{250,250}	0.175	0.239	0.314
{100,100,100}	0.135	0.089	0.277
{250,250,250}	0.183	0.196	0.400

TABLE A.11: Coefficients of determination for using the principal components of the wind field of complete bay, using various numbers of hidden neurons.

#Hidden Neurons	
500	0.188
1000	0.232
{500,500}	0.395
{1000,1000}	0.450
{500,500,500}	0.496
{1000,1000,1000}	0.559

TABLE A.12: Coefficients of determination for using the principal components of the wind field of complete bay, using the ReLu activation function for a various number of hidden neurons.

A.5 Zushi

In this section we highlight the results of training our model on the data of the Zushi race area. This is the area where the accuracy of the final model was the lowest. Interestingly enough, the accuracy of the model with only one single hidden layer, containing 500 hidden neurons performed better here than on Fujisawa and Sagami, but the added layers added relatively less value here.

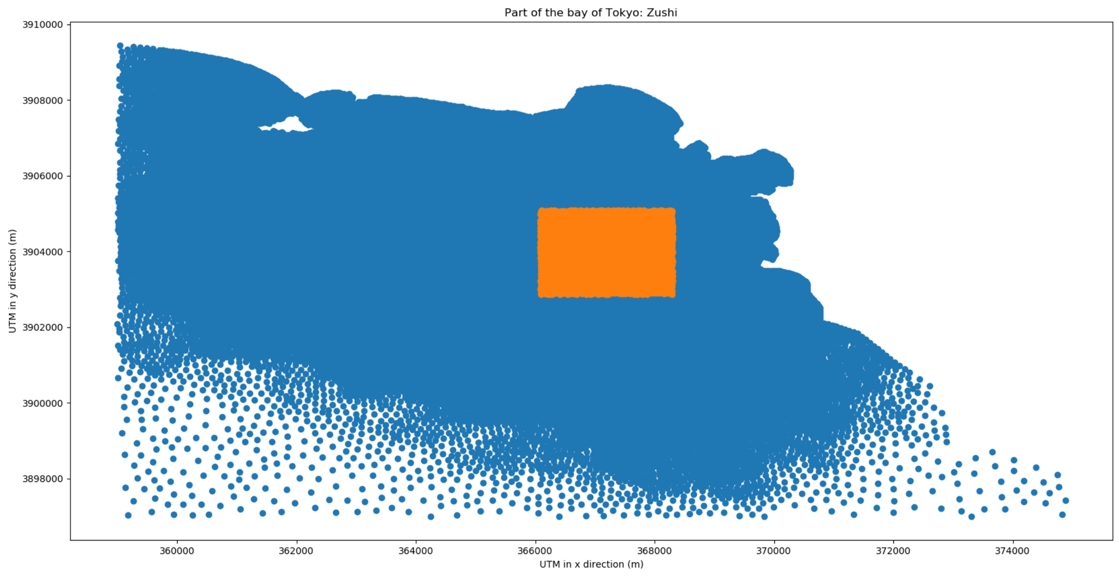


FIGURE A.17: Grid points of race area Zushi highlighted

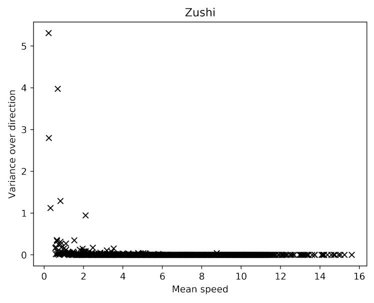


FIGURE A.18: Variance in direction plotted against mean velocity of wind fields.

Principal components required to capture >99% of the variance:

- Wind: 2 Principal Components
- Water: 10 Principal Components

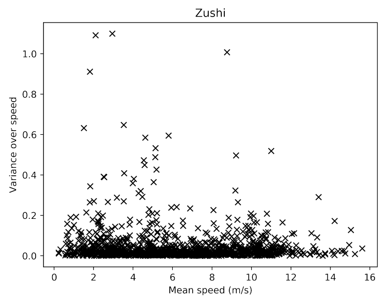


FIGURE A.19: Variance in velocity plotted against mean velocity of wind fields.

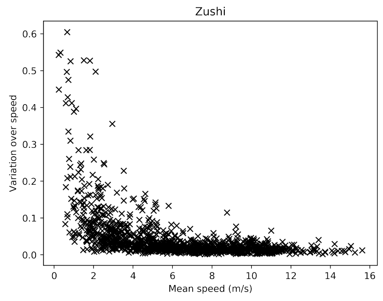


FIGURE A.20: Relative variance in velocity plotted against mean velocity of wind fields.

#Hidden Neurons	Hyperbolic Tangent	Sigmoid	ReLu
10	0.109	0.087	0.240
25	0.144	0.166	0.252
50	0.213	0.209	0.258
100	0.226	0.243	0.260
250	0.261	0.246	0.262
{100,100}	0.216	0.201	0.263
{250,250}	0.268	0.227	0.258
{100,100,100}	0.206	0.178	0.237
{250,250,250}	0.231	0.187	0.217

TABLE A.13: Coefficients of determination for using the principal components of the wind field of the race area, using various numbers of hidden neurons.

#Hidden Neurons	Hyperbolic Tangent	Sigmoid	ReLu
10	0.109	0.081	0.235
25	0.165	0.147	0.268
50	0.200	0.199	0.269
100	0.239	0.237	0.246
250	0.288	0.271	0.237
{100,100}	0.210	0.217	0.271
{250,250}	0.262	0.310	0.329
{100,100,100}	0.187	0.181	0.301
{250,250,250}	0.251	0.292	0.399

TABLE A.14: Coefficients of determination for using the principal components of the wind field of complete bay, using various numbers of hidden neurons.

#Hidden Neurons	
500	0.257
1000	0.285
{500,500}	0.383
{1000,1000}	0.454
{500,500,500}	0.464
{1000,1000,1000}	0.507

TABLE A.15: Coefficients of determination for using the principal components of the wind field of complete bay, using the ReLu activation function for a various number of hidden neurons.

Appendix B

Code

B.1 Code

```
#Import Packages:

#Importing Data
import pickle as pc
#Importing functions for statistics
from statistics import *
from numpy import *
#Principal Component Analysis
from sklearn.decomposition import PCA
#Generating random samples
import random
#Neural Network
from sklearn.neural_network import MLPRegressor
from sklearn.datasets import make_regression
from sklearn.model_selection import train_test_split
#Cross Validation
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
#Plotting
import matplotlib.pyplot as plt
import scipy.interpolate as interpolate


#Functions:

#Import the data --> recall that the data is an array where every row is a wind/current fie
#This also includes the coordinates
def select_area(area):
# Getting back the objects:
    with open("./areadata/" + area + '.pkl', 'rb') as f:  # Python 3: open(..., 'rb')
        wind_data, c_data, wind_polar, c_polar, testdata, xy_sel = pc.load(f)

    #fig1 = windplot(testdata)
    #display(HTML(fig1.to_html()))

    return(testdata, c_data, wind_data, wind_polar, xy_sel)
```

```
#Function to plot the data
def create_field(fig, ax, df, coords, form, vmin=0, vmax=20, unit = "speed m/s"):## spl

    ## step 1: make grid with equal steps based on coords
    xr = np.arange(coords[:,0].min(), coords[:,0].max(), 15)
    yr = np.arange(coords[:,1].min(), coords[:,1].max(), 15)
    xx,yy = np.meshgrid(xr,yr,copy=True, sparse=False, indexing='xy')

    ## create figure
    if form ==2:
        ll = int(len(df))
        X = df[0:int((ll/2))] # velocity in x
        Y = df[int(ll/2):int(len(df))] # velocity in y
        EE   =np.sqrt(X**2+Y**2) # magnitude of the velocity (speed)
        ex=X/EE # eenheidsvector
        ey=Y/EE #*eenheidsvector

        ## step 2: interpolate x,y and EE over the grid (griddata in matlab)
        grid_z0 = interpolate.griddata(coords, EE, (xx, yy), method='cubic')
        grid_z1 = interpolate.griddata(coords, ex, (xx, yy), method='cubic')
        grid_z2 = interpolate.griddata(coords, ey, (xx, yy), method='cubic')

    else:
        grid_z0 = interpolate.griddata(coords, df, (xx, yy), method='cubic')


    ## step 3 create plot
    cmap = plt.cm.get_cmap("jet")
    ax.axis('equal')

    cm   = ax.pcolormesh(xr,yr, grid_z0, cmap=cmap, vmin = vmin, vmax=vmax)
    fig.colorbar(cm, extend='both', label= unit, ax=ax)

    if form ==2:
        skip=(slice(None,None,8))
        ax.quiver(coords[skip,0], coords[skip,1], X[skip],Y[skip], units='xy', color =


#Training the Neural Network

#Loading data
#Race area ---> Options: ['Enoshima', 'Fujisawa', 'Hayama', 'Kamakura', "Sagami", "Zush
#xy_sel represent the corresponding coordinates
area = 'Hayama'
[testdata, c_data, wind_data, wind_polar, xy_sel] = select_area(area)
#Full bay
with open('fullBay.pkl', 'rb') as f:
    date, fullCurrent, fullWind, fullCurrentPCA, fullWindPCA = pc.load(f)
#Olympic area
with open('olympicArea.pkl', 'rb') as f:
    date2, olyCurrent, olyWind, olyCurrentPCA, olyWindPCA = pc.load(f)



#Filter on 2019 data
c_data2019 = c_data[1395:,:]
wind_data2019 = wind_data[1395:,:]

#PCA Current
pca_cur = PCA(n_components = 10)
pca_c_data = pca_cur.fit_transform(c_data2019)

#PCA Wind
pca_wind = PCA(n_components = 2)
pca_wind_data = pca_wind.fit_transform(wind_data2019)

#Select your data, here we use the principal components data of the wind fields of the comp
#and the current field of the race area

X = fullWindPCA
y = pca_c_data

#You can deselect some (random) samples to check out how well these are predicted
#X_train = np.delete(X.copy(), randomlist, 0)
#y_train = np.delete(y.copy(), randomlist, 0)

X_train = X.copy()
y_train = y.copy()
#Using cross validation to measure the accuracy  ---> Here we use 10-fold cross validation
cv = KFold(n_splits = 10, random_state = 1, shuffle = True)
#You can use different setups ---> Here we use 3 hidden layers of size 1000 and the ReLu ac
regr = MLPRegressor(random_state = 10, max_iter = 5000, hidden_layer_sizes = (1000,1000,100
scores = cross_val_score(regr, X, y, cv = cv, n_jobs = -1)

print('Accuracy: %.3f (%.3f)' % (mean(scores), std(scores)))

#Training the model without using cross validation:
regr = MLPRegressor(random_state = 10, max_iter = 5000, hidden_layer_sizes = (1000,1000,100


########

#Using the Neural network to make prediction: ---> In this case field 382 from the array
pred_pca_field = regr.predict(X[382,:].reshape(1,-1))
#Transforming the predicted principal components back to the original space
pred_full_field = pca_cur.inverse_transform(pred_pca_field)
#Plotting the field
fig1=plt.figure(figsize=(10,8))
fig1.clf()
data = pred_full_field[0]
ax1 = fig1.add_subplot(111)
```

```
create_field(fig1,ax1, data, xy_sel, form=2, vmin=0, vmax=20, unit = "speed (m/m)")

########

#Extra Analysis

#Determining the number of principal components ---> In this case for the current data
pca = PCA()
test = pca.fit(c_data2019)
fig = plt.figure()
fig.set_size_inches(18, 7)
exp_var_pca = test.explained_variance_ratio_[0:10]
cum_sum_eigenvalues = np.cumsum(exp_var_pca)
plt.bar(range(0,len(exp_var_pca)), exp_var_pca, alpha=0.5, align='center', label='Indiv
plt.step(range(0,len(cum_sum_eigenvalues)), cum_sum_eigenvalues, where='mid',label='Cum
plt.ylabel('Explained variance ratio')
plt.xlabel('Principal component index')
plt.legend(loc='best')
plt.tight_layout()
plt.show()


#Choose number of components based on explained variance ---> In this case if explained
idx = np.where(np.cumsum(test.explained_variance_ratio_)>.99)
print("Number of necessary principal components is " + str(idx[0][0] + 1))
```

Bibliography

Chang, Y-C et al. (2012). “Observed near-surface currents under high wind speeds”. In: Journal of Geophysical Research: Oceans 117.C11.

Hecht-Nielsen, Robert (1992). “Theory of the backpropagation neural network”. In: Neural networks for perception. Elsevier, pp. 65–93.

Iwata, Shizuo and Masaji Matsuyama (1989). “Surface circulation in Sagami Bay: the response to variations of the Kuroshio axis”. In: Journal of Oceanography 45.5, pp. 310–320.

Jolliffe, Ian (2005). “Principal component analysis”. In: Encyclopedia of statistics in behavioral science.

Jolliffe, Ian T and Jorge Cadima (2016). “Principal component analysis: a review and recent developments”. In: Philosophical Transactions of the Royal Society A: Mathematical, Physical an 374.2065, p. 20150202.

Likas, Aristidis, Nikos Vlassis, and Jakob J Verbeek (2003). “The global k-means clustering algorithm”. In: Pattern recognition 36.2, pp. 451–461.

Panchal, Gaurang et al. (2011). “Behaviour analysis of multilayer perceptrons with multiple hidden neurons and hidden layers”. In: International Journal of Computer Theory and Engine 3.2, pp. 332–337.

Powers, Jordan G et al. (2017). “The weather research and forecasting model: Overview, system efforts, and future directions”. In: Bulletin of the American Meteorological Society 98.8, pp. 1717–1737.

Sharma, Sagar, Simone Sharma, and Anidhya Athaiya (2017). “Activation functions in neural networks”. In: towards data science 6.12, pp. 310–316.

Spark, Elly and Gregory J Connor (2004). “Wind forecasting for the sailing events at the Sydney 2000 Olympic and Paralympic Games”. In: Weather and forecasting 19.2, pp. 181–199.

Svasek (n.d.). FINEL Model. URL: https://www.svasek.nl/model-research/finel/.

Zhang, Zhihua (2018). “Artificial neural network”. In: Multivariate time series analysis in climate and envi Springer, pp. 1–35.