Prédiction de trajectoires d’avions à l’aide de la régression floue.

Aircraft trajectory prediction using fuzzy regression.

Réalisé par GHASEMI HAMED Mohammad

Stage effectué au cours de la période du 15 Avril 2010 à fin Septembre 2010 au domaine R&D de la Direction de la Technique et de l’Innovation, Direction des Services de la Navigation Aérienne

Sous la direction de M. SERRURIER Mathieu
IRIT (équipe Raisonnements Plausibles, Décision, Méthodes de Preuves)

Résumé : Pour répondre à la demande croissante du trafic aérien, nous avons besoin de systèmes de gestion du trafic aérien qui soient capables d’anticiper automatiquement les éventuels conflits entre les trajectoires d’avions. Une grande partie de l’efficacité des solutions proposées par ces outils repose principalement sur l’efficacité de leur module de prédiction. Étant donné que ces prédictions seront effectuées au sol, nous ne disposons pas des données nécessaires pour faire une prédiction avec les équations physiques du vol. Donc nous sommes contraints d’employer des méthodes capables de fournir des résultats avec les meilleurs compromis entre l’exactitude et la l’imprécision sur les valeurs prédites. Pour simplifier la question, nous allons aborder le problème par la prédiction de l’altitude de l’avion dans sa phase de montée. Afin de trouver une solution, nous comparons la régression Imprécise, plus précisément, la régression Imprécise des K plus proches voisins, avec d’autres techniques de régression utilisés dans de telles situations.

Mots clés : régression, ensembles flous, trajectoire.
Remerciements

Je tiens à exprimer ma gratitude envers mon maître de stage, Mathieu Serrurier, enseignant chercheur à l’IRIT, qui m’a accepté comme comme stagiaire, m’a soutenu et m’a guidé pendant tout le long du stage.

Je tiens à remercier sincèrement David Gianazza, chef de l’équipe POM, pour tout l’aide qu’il m’a apporté pour rédiger ce rapport.

Je remercie Nicolas Durand, directeur adjoint de la DTI R&D, pour m’avoir accueilli au sein de l’équipe POM basée sur le site de la DTI R&D à Rangueil.

Merci à mon tuteur de stage Fabrice Evrard, pour la suivie de l’avancement de mon stage.

Je remercie Caroline Becker, Richard Alligier et Cyrile Allignol mes collègues à la DTI qui m’ont aider pendant la rédaction du rapport.

Mes remerciements vont enfin à l’ensemble de l’équipe POM présente au sein de la DTI, Jean Baptiste Laborde et Simon Marchal pour leur accueil, leur cordialité et leur aide.
# Contents

Remerciements i

Glossary v

Liste des tableaux vi

Table des figures vii

## I Context

I.1 Aircraft trajectory .................................................. 4
   I.1.1 Lateral profile .............................................. 5
   I.1.2 Vertical profile ............................................. 5
   I.1.3 The climb phase .............................................. 5
   I.1.4 The cruise phase .............................................. 5
   I.1.5 The descent phase ........................................... 7

I.2 Aircraft trajectory data ........................................... 8
   I.2.1 Data description ............................................. 8
   I.2.2 Data structure ............................................... 8

I.3 Summary in French .................................................. 16

## II Background

II.1 Regression .......................................................... 17
   II.1.1 Regression analysis: A little more .......................... 17
   II.1.2 Normal distribution of the error ............................ 18
   II.1.3 Linear Regression ............................................ 18
   II.1.4 Neural network ............................................... 20
   II.1.5 Support vector machine (SVM) ............................... 21

II.2 K-nearest neighbor (KNN) ........................................ 22
   II.2.1 KNN classification ........................................... 22
   II.2.2 KNN regression ............................................... 23

II.3 Fuzzy regression .................................................. 24
   II.3.1 Fuzzy set .................................................... 24
   II.3.2 Fuzzy regression methods ................................... 25
## Table des matières

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>II.3.3 Imprecise regression</td>
<td>26</td>
</tr>
<tr>
<td>II.4 Summary in French</td>
<td>27</td>
</tr>
<tr>
<td><strong>III Our approach</strong></td>
<td></td>
</tr>
<tr>
<td>III.1 KNN Regression</td>
<td>29</td>
</tr>
<tr>
<td>III.2 Distance function</td>
<td></td>
</tr>
<tr>
<td>III.2.1 Euclidean</td>
<td>30</td>
</tr>
<tr>
<td>III.2.2 Mahalanobis</td>
<td>31</td>
</tr>
<tr>
<td>III.2.3 Constrained distance</td>
<td>31</td>
</tr>
<tr>
<td>III.2.4 Constrained distance with relaxation</td>
<td>32</td>
</tr>
<tr>
<td>III.2.5 Area between trajectory (ABT)</td>
<td>33</td>
</tr>
<tr>
<td>III.2.6 Area between trajectory 2nd version (ABTv.2)</td>
<td>33</td>
</tr>
<tr>
<td>III.2.7 Hybrid ABT/Euc</td>
<td>35</td>
</tr>
<tr>
<td>III.3 Regression interval</td>
<td>35</td>
</tr>
<tr>
<td>III.4 KNN regression with Gaussian distribution</td>
<td>35</td>
</tr>
<tr>
<td>III.4.1 Global Gaussian distribution</td>
<td>36</td>
</tr>
<tr>
<td>III.4.2 Local Gaussian distribution</td>
<td>36</td>
</tr>
<tr>
<td>III.4.3 Proof of normality</td>
<td>37</td>
</tr>
<tr>
<td>III.5 KNN regression with fuzzy set</td>
<td>37</td>
</tr>
<tr>
<td>III.5.1 First optimization method</td>
<td>38</td>
</tr>
<tr>
<td>III.5.2 Second optimization method</td>
<td>39</td>
</tr>
<tr>
<td>III.6 Summary in French</td>
<td>42</td>
</tr>
<tr>
<td><strong>IV Experiments</strong></td>
<td></td>
</tr>
<tr>
<td>IV.1 Regression comparison</td>
<td>44</td>
</tr>
<tr>
<td>IV.1.1 Algorithm comparison</td>
<td>44</td>
</tr>
<tr>
<td>IV.1.2 Attribute selection</td>
<td>45</td>
</tr>
<tr>
<td>IV.2 KNN algorithm</td>
<td>47</td>
</tr>
<tr>
<td>IV.2.1 Chosing the value of $K$</td>
<td>48</td>
</tr>
<tr>
<td>IV.2.2 Choosing the distance functions</td>
<td>48</td>
</tr>
<tr>
<td>IV.2.3 Analysis of the normality distribution</td>
<td>49</td>
</tr>
<tr>
<td>IV.2.4 Fuzzy set prediction</td>
<td>50</td>
</tr>
<tr>
<td>IV.2.5 Choosing the interval type</td>
<td>56</td>
</tr>
<tr>
<td>IV.2.6 Conclusion</td>
<td>58</td>
</tr>
<tr>
<td>IV.3 Summary in French</td>
<td>59</td>
</tr>
<tr>
<td><strong>Conclusion</strong></td>
<td>62</td>
</tr>
</tbody>
</table>
Glossary

**MAE** : Mean Absolute Error.

**RMSE** : Root Mean Squared Error.

**ABT** : Area between trajectory.

**MLP** : Multilayer Perceptron.

**ANN** : Artificial Neural Networks.

**KNN** : K-nearest neighbor.

**SVM** : Support vector machine.

**ATM** : Air Traffic Management.

**ENAC** : École Nationale de l’Aviation Civile

**DGAC** : Direction Générale de l’Aviation Civile

**DSNA** : Direction des Services de la Navigation Aérienne
List of Tables

| IV.1 | Regression techniques. | 45 |
| IV.2 | Variable selection comparison. | 47 |
| IV.3 | KNN with Normalized Euclidean distance with different K values. | 48 |
| IV.4 | Constrained distance example. | 49 |
| IV.5 | Constrained distance with relaxation, using different parameters. | 50 |
| IV.6 | Different distance comparison. | 52 |
| IV.7 | The 95% intervals. | 52 |
| IV.8 | The 50% intervals. | 53 |
| IV.9 | Regression intervals for KNN with Constrained normalized Euclidean with relaxation with best set using a K=30. | 53 |
| IV.10 | Regression intervals for KNN with normalized Euclidean with best set using a K=30. | 53 |
# List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>I.1</td>
<td>Climb phase</td>
<td>6</td>
</tr>
<tr>
<td>I.2</td>
<td>Cruise phase</td>
<td>6</td>
</tr>
<tr>
<td>I.3</td>
<td>Descent phase</td>
<td>7</td>
</tr>
<tr>
<td>I.4</td>
<td>Altitude change</td>
<td>11</td>
</tr>
<tr>
<td>I.5</td>
<td>Distance change</td>
<td>12</td>
</tr>
<tr>
<td>I.6</td>
<td>Double predictor</td>
<td>13</td>
</tr>
<tr>
<td>I.7</td>
<td>Sample predictor</td>
<td>14</td>
</tr>
<tr>
<td>II.1</td>
<td>Example of linear regression with one independent variable.</td>
<td>19</td>
</tr>
<tr>
<td>II.2</td>
<td>A multilayer perceptron.</td>
<td>21</td>
</tr>
<tr>
<td>II.3</td>
<td>A trained SVM with samples from two classes.</td>
<td>22</td>
</tr>
<tr>
<td>II.4</td>
<td>Example of non-weighted KNN classification.</td>
<td>23</td>
</tr>
<tr>
<td>II.5</td>
<td>Example of weighted KNN regression.</td>
<td>24</td>
</tr>
<tr>
<td>II.6</td>
<td>Coolness of the air, represented in fuzzy.</td>
<td>25</td>
</tr>
<tr>
<td>III.1</td>
<td>ABT distance.</td>
<td>34</td>
</tr>
<tr>
<td>III.2</td>
<td>Global Gaussian distribution.</td>
<td>36</td>
</tr>
<tr>
<td>III.3</td>
<td>Non Gaussian distribution.</td>
<td>38</td>
</tr>
<tr>
<td>III.4</td>
<td>A trapezoidal fuzzy set.</td>
<td>39</td>
</tr>
<tr>
<td>IV.1</td>
<td>Kset density VS Gaussian density.</td>
<td>51</td>
</tr>
<tr>
<td>IV.2</td>
<td>Kset density VS Gaussian density, another Example.</td>
<td>51</td>
</tr>
<tr>
<td>IV.3</td>
<td>Comparing different intervals in their data representation.</td>
<td>54</td>
</tr>
<tr>
<td>IV.4</td>
<td>Possibilisty distribution VS Gaussian distribution.</td>
<td>55</td>
</tr>
<tr>
<td>IV.5</td>
<td>Possibilisty distribution VS Gaussian distribution, another Example.</td>
<td>55</td>
</tr>
<tr>
<td>IV.6</td>
<td>Comparing the predictive power, Fuzzy set VS Gaussian distribution.</td>
<td>56</td>
</tr>
<tr>
<td>IV.7</td>
<td>Interval sizes.</td>
<td>57</td>
</tr>
</tbody>
</table>
Abstract

To respond the increasing levels of air traffic demandes, we need Air Traffic Management (ATM) systems capable to automatically detect and solve the possible aircraft trajectory conflicts. The efficiency of these conflict solvers are mainly influenced by their trajectory prediction module. As these predictions are on the ground predictions, we don’t dispose of all the necessary data to use with the physics laws to predict the precise location of the aircraft. Thus we have to employ methods capable to provide results with the best trade off between the accuracy and the imprecision of the prediction values. To simplify our problem, we just focus on the prediction of the aircraft altitude in the climb phase. In order to find a solution, we compare the imprecise regression method more specifically, the KNN imprecise regression, with other regression techniques that could be used when dealing with such situations.
Introduction

La croissance rapide du trafic aérien augmente régulièrement la pression sur les systèmes ATM (Air Traffic Management). Les estimations montrent que d’ici 2020, la croissance du trafic aérien nous demandera des systèmes ATM capables de gérer deux fois plus d’avions qu’aujourd’hui, tout en fournissant un niveau de sécurité constant et un impact sur l’environnement plus faible[8]. L’utilisation de systèmes ATM automatisés ou semi-automatisés est l’une des approches qui permettraient d’atteindre ces objectifs. L’objectif des systèmes automatisés est d’anticiper de manière automatique les conflits possibles, au moins une partie, afin de proposer une solution au contrôleur ou, après validation de la solution par le contrôleur transmettre directement des ordres de contrôle à l’avion. Cependant, ce type de système nécessite d’être capable de prédire de manière automatique la trajectoire d’un avion. Sans modèle de prédiction efficace, il est impossible de créer des systèmes ATM automatisés ou semi-automatisés. Sans prévision fiable, il est impossible d’avoir des systèmes automatisés qui satisfassent les contraintes de sécurité. L’automatisation ou la semi-automatisation des systèmes ATM peut nous aider dans cette démarche. Actuellement, la Direction de la Technique et de l’Innovation (DTI) dispose d’un détecteur et solveur de trafic aérien. Il a été démontré, à l’aide de simulations, que cet outil est assez robuste pour être employé dans un système ATM, sous certaines hypothèses sur la prédiction de trajectoires. L’efficacité de ces outils dépend beaucoup du taux de faux conflits détectés (faux positifs) et du taux de véritables conflits non détectés (faux négatifs). Ces taux sont directement liés à la performance du module de prédiction utilisé dans le solveur. La prévision de trajectoire aérienne porte sur trois horizons temporels : la prévision à court terme (moins que 30 secondes), la prévision à moyen terme (dix à vingt minutes) et la prévision à long terme (actuellement pour les vols européens, de 4 à 5 heures) [11]. Afin de pouvoir automatiser la détection et la résolution de conflit et tout en garantissant un niveau satisfaisant de sécurité, nous nous intéressons dans ce rapport à la prédiction à moyen terme. Aujourd’hui, les modèles de prédiction utilisés sont basés sur une modélisation physique de l’avion et de son environnement. Les paramètres employés par ces modèles sont les suivants : le modèle de l’avion, le modèle météo et les intentions du vol. Le modèle d’avion comprend les données et les équations qui permettent de décrire la trajectoire de l’avion en vol. Le poids de l’avion est un paramètre critique, or il n’est pas fourni de manière précise par les compagnies aériennes. Le modèle météo est naturellement toujours imprécis et enfin, les intentions de vols ne sont presque jamais connues au sol. Au bord de l’avion, le pilote dispose d’un prédicteur de trajectoire avec lequel il ajuste les paramètres de vol afin de respecter les contraintes du vol. Ce prédicteur utilise le modèle à énergie totale pour prédire la trajectoire de l’avion. Comme le modèle a énergie totale utilise la masse, la vitesse et l’accélération de
l’avion, qui ne sont pas connues au sol, il ne peut pas être utilisé comme prédicteur sol. Cependant, nous disposons d’une importante masse de données décrivant des trajectoires. L’existence de ces données permet d’envisager d’utiliser des techniques issues de l’apprentissage artificiel afin de créer un modèle de prédiction. Dans ce travail, nous allons utiliser des méthodes de régression pour prédire la trajectoire de l’avion. Tenant compte du manque de données au sol, de leur imprécision, et de la nature même du processus d’apprentissage, il est illusoire de vouloir faire une prévision exacte. Ainsi, s’en tenir au résultat précis fourni par le modèle issu d’un algorithme d’apprentissage poserait d’importants problèmes de sécurité. Nous allons donc chercher à prédire, plutôt qu’une valeur précise, un intervalle qui devra contenir l’emplacement de l’avion dans la plupart des cas. Pour cela, nous allons utiliser la méthode des k plus proches voisins, qui permet d’obtenir un échantillon de données (donc, la qualité dépend de la fonction de distance utilisée) censé décrire des valeurs plausibles pour la prédiction. Il existe alors plusieurs façons de déterminer l’intervalle de prédiction. La plus utilisée consiste à considérer que la distribution suit une loi normale. On peut alors déterminer complètement cette distribution en calculant la moyenne et la variance de l’échantillon. En prenant le quantile 95% on peut espérer obtenir un intervalle qui contient la majorité des données tout en mettant de côté les valeurs aberrantes. Cette méthode n’est efficace que quand la distribution des données est Gaussienne. L’autre approche consiste à essayer de construire un ensemble flou qui décrit ces données. On parle alors de régression floue ou de régression imprécise. Cet ensemble flou correspond à une distribution de possibilité qui représide une famille de distribution de probabilité. L’utilisation de distribution de possibilité, même si elles sont moins précises que les distributions de probabilité, permet d’avoir une description fiable des données sans connaître a priori la forme de la distribution. Notre objectif est de vérifier si les valeurs issues des k plus proches voisins forment une distribution Gaussienne et nous allons ensuite voir si cette distribution ne peut pas être mieux décrite par une distribution de possibilité de possibilité. Pour cela, nous allons utiliser la méthode des K plus proches voisins combinée avec la méthode de régression imprécise. La méthode des K plus proches voisins repose beaucoup sur l’efficacité de la fonction de distance employée. Nous allons donc déterminer différentes fonctions de distance, certaines génériques et certaines spécialement conçues pour la prédiction de trajectoire, et comparer leurs efficacités sur les données. Ainsi, nous avons proposé deux nouvelles deux nouvelles fonctions de distance : les distances à contrainte avec relaxation et les distances basées sur la nature fonctionnelle des trajectoires. Ce texte est organisé de la manière suivante : dans le premier chapitre, nous décrivons la problématique de la prédiction de trajectoire aérienne. Nous y décrivons aussi la structure des données. Ensuite, nous présentons un état de l’art sur les méthodes de régression, puis nous verrons les ensembles flous et la régression imprécise. Le troisième chapitre décrit notre approche pour aborder cette problématique. Nous y verrons la régression avec les K plus proches voisins, les différentes fonctions de distances et enfin la régression avec les intervalles, notamment la régression imprécise. Le dernier chapitre comportera les résultats de nos tests sur des données réelles.
Chapter I

Context

I.1 Aircraft trajectory

We begin by describing briefly how trajectory is defined and where it is used, then we will explain in more detail what is our definition of trajectory.

The flight management system provides navigation, optimized route determination and flight planning. The aircraft’s en route guidance is usually composed by the following interrelated functions: flight planning, navigation, trajectory prediction, performance computations, and guidance.

The core of the Flight Management System (FMS) is the construction of the flight plan and the four-dimensional aircraft trajectory defined by the specified flight plan constraints and legs and the aircraft performance.

The flight planning and the trajectory prediction parts work together to supply the four-dimensional trajectory and merge them with all the appropriate trajectory information into a flight plan/profile.

The trajectory is composed of two parts — the lateral profile and the vertical profile. In spite of that, the lateral and vertical path are interdependent so that the ground speed is their coupling parameter. As shown in the following equations, since the aircraft has a flown speed schedule that is usually constant, CAS/mach speeds for climb and descent phases, the TAS (or ground speed) increases with altitude for the constant CAS portion and mildly decreases with altitude for the constant mach portion.

\[
mach = \sqrt{\left[\frac{1}{\delta}[1 + 0.2(CAS/661, 5)^2]^{3.5} - 1 + 1\right]^{0.286} - 1}
\]

\[
TAS = 661, 5 \times mach \times \sqrt{\theta}
\]

where

- \(CAS\) = calibrated airspeed in knots
- \(TAS\) = true airspeed in knots
- \(\delta\) = atmospheric pressure ratio (actual temperature / S.L. std. temperature)
I. CONTEXT

• \( \theta \) = atmospheric temperature ratio ( actual temperature/S.L. std. temperature )

The importance of the air speed’s change with altitude will become obvious in the lateral and vertical profile construction during ascending and descending flights.

I.1.1 Lateral profile

The lateral profile is the specified route by the FMS. It is composed of procedure legs, way points, hold patterns, etc, and that’s the FMS which computes all the turns and leg termination points according to how the aircraft should fly them.

The lateral profile is composed by straight and turn segments which begin and end at geographical points. It is difficult to compute all these segments because the turn transition distance and some leg termination points are calculated based on the predicted aircraft speed, wind, and altitude, which, themselves, are function of how much distance is available to climb and descend. Finally, the lateral profile computed by the FMS contains much more data than just straight lines connecting fixed way points. This is a complete prediction of the lateral path that the aircraft must fly in the trajectory.

I.1.2 Vertical profile

In the vertical profile we have the numerical integration of the aircraft energy balance equations including variable weight, speed, and altitude. The vertical profile is defined by the following manoeuvre types: unrestricted ascending and descending segments, restricted ascending and descending segments, level segments, and speed change segments.

I.1.3 The climb phase

The climb phase is typically defined by the segments shown in the figure I.1.

I.1.4 The cruise phase

It is a very simple vertical path, computed along the lateral path. It is normally composed of a climb speed to cruise speed acceleration or deceleration segment then a segment reaching the FMS computed top of descent. The cruise phase is normally calculated via several distances or time-based integration steps.

In contrary to the the climb and descent phases, the optimal cruise speeds slowly change with the changing weight of the aircraft which is caused by fuel burn.
I. Context

Figure I.1: Climb phase

Figure I.2: Cruise phase
I.1.5 The descent phase

The vertical descent phase (computed with the lateral path) can be composed of several vertical leg types, as shown in the figure I.3 [22].
I.2 Aircraft trajectory data

I.2.1 Data description

As we saw earlier for the aircraft trajectory prediction we need the following elements:

- Route description and the aircraft intention. (how the airplane will climb and descend)
- The aircraft performance model.
- Other elements like the earth model.

As we said earlier there exists a total-energy model[17] which gives us the true speed (TAS) of the aircraft to predict its trajectory in the next moment. In the meantime we need the aircraft’s mass, altitude, thrust (acting parallel to the aircraft velocity vector), aerodynamic drag, the velocity vector (which implies the orientation of the aircraft), and other parameters to compute the aircraft’s speed. It is obvious to see that at the moment of the prediction we don’t dispose of such data, for example the thrust acting parallel to the aircraft velocity vector, is a parameter that must be measured online and it is not possible to predict it without exact knowledge of the aircraft’s situation. As a result of this lack of data we have to use other method to predict the aircraft trajectory. Nowadays, we dispose of 3 minutes aircraft trajectory prediction by radar[11], but this is by far insufficient for the task of conflict detection. The data used in trajectory prediction are uncertain by nature, the earth model is just a prediction of how the weather is at the given altitude. For instance we have to know the atmospheric pressure at a given altitude, whereas we know that our estimation is always an imprecise. We can have the actual speed and altitude of the aircraft and typically these data are not precise. These imprecision change by certain number of parameters, for example the weather imprecision can be modelled but this is a difficult task and out of the scope of this text, therefore we will treat the problem of trajectory prediction as a prediction problem with precise data as input. Aircraft intentions are a set of instructions furnished to the trajectory engine. Based on the aircraft performance model, an instruction can be took into account or not because the trajectory engine is not capable to understand all the concepts[11]. The computation of the trajectory based on the aircraft intention requires some parameters that we miss some part like the weight of the aircraft, on the other hand the remaining parameters are not precise consequently we will ignore the aircraft intention in the first attempt to trajectory prediction.

For the task of prediction we will use machine learning algorithms. As rule of thumb we know that the simpler the model in training mode, the better result in production mode. Including the aircraft intention in the prediction models make them much more complex, thus at first, we will just use radar plots, earth model and aircraft performance models.

I.2.2 Data structure

In the first place we have to decide about our predictor type. Since we have to predict the trajectory at a given time, and due to the fact that a point at a given time represents an altitude, a latitude
and a longitude we have to employ regression algorithms to predict this three numbers. One can try to discretize the altitude, the latitude and the longitude values and use classification algorithm but the consequent problem that occurs is the fact that for the classifier algorithm, all the classes are equi-distant. For example the difference between the class of $1500\pm100$ and $1600\pm100$, is the same as the difference between the class of $1500\pm100$ and $2600\pm100$. This is a big problem because a miss-classification of the first type may be tolerated based on the predictor efficiency, but the second one is really an “absolute error” while for the algorithm, they are similar. After the selection of our strategy which will be supervised machine learning (regression algorithms are supervised), we have to select the data structure. The attributes set, The type of the attributes and the selection method for data used for the training and validation process, are the questions to respond before selecting our learning scheme. For the position of the aircraft we used the radar plots that were taken every 1 to 3 seconds. We had also an estimate of the wind and temperature on the given radar plot. The data field are the followings:

- **TIME**: time in second (from a given time, ex 1990) were the plot has been registered.
- **ID**: identifier of this flight.
- **LAT**: latitude of the aircraft.
- **LON**: latitude of the aircraft.
- **ALTITUDE**: altitude of the aircraft.
- **X\_CAUTRA**: $x'$ CAUTRA\(^1\) position of the of the aircraft.
- **Y\_CAUTRA**: $y'$ CAUTRA position of the of the aircraft.
- **ALTITUDE\_SMO**: atmospheric pressure at this point.
- **VIT\_X(kts)**: velocity of aircraft in X\_CAUTRA direction.
- **VIT\_Y(kts)**: velocity of aircraft in Y\_CAUTRA direction.
- **ROCD**: rate of climb or descent.
- **TAS**: true airspeed in knots
- **WIND\_X(kts)**: wind velocity in X\_CAUTRA direction.
- **WIND\_Y(kts)**: wind velocity in Y\_CAUTRA direction.
- **TEMP(C)**: temperature at this point.

---

\(^1\)Coordinateur AUtomatique du TRafic Aérien; it is a set of systems which manage aircraft trajectory informations, their distribution over the control positions, the correlation with radar pists, etc.
We can see that all these data fields are numerical. There are also other description of the flight at this point that can be obtained via the integration of these fields and other formula, for example we’ve seen above that we can obtain the CAS and MACH from the TAS, atmospheric pressure and atmospheric temperature.

Let’s reformulate the trajectory prediction problem like this: we know the current location of the aircraft and a limited part of its flight, just before its current location. We want to know where will be its position in \( t \) second after. Now we have an idea of what kind of input data we must give to our algorithm. Here, the location of the aircraft in each plot is defined by its altitude and its distance travelled from the airport. The current location of the aircraft with ten preceding locations is given to the algorithm and it has to predict the future location (altitude and distance traveled from the airport) of the aircraft at a given time. The time is discretized in interval of 15 seconds and the prediction is done from 1 to 40 point in future which means from fifteen second to ten minutes (40 \( \times \) 15s).

As you can see in the figure I.4, the point \( z_0 \) is the current altitude of the aircraft and the point \( z_T \) shows the future altitude of the aircraft at the time \( t_{\text{predict}} \). The point \( z_1 \) the nearest precedent point to \( z_0 \), in other word it represents the altitude of the aircraft at \( t_0 - 1 \). The \( Dz[i] \) represents the distance between the point \( z[i-1] \) and the point \( z[i] \). For instance the \( Dz_1 \) is the altitude that the aircraft travelled from \( z_1 \) to reach the \( z_0 \) altitude. The trajectory predictor must be capable to predict an altitude enough near to this point so that the prediction remains useful.

The figure I.5 shows the \( x[i] \) which is the aircraft’s travelled distance from the airport. Thus if the aircraft turns around the airport, its \( x \) at the next time plot will certainly increases, because it has travelled more distance whereas its distance from the airport at the next time plot may be the same. As we will see in the experiment chapter, the attributes describing \( x \) at different \( t \) improves our prediction, so it is always more interesting to use them.

Our data format has the following fields. time : time in second (from a given time, ex 1990) were the plot has been registered.

- \( t_{\text{predict}} \) : predicted time (form 1 to 40).
- \( z_T \) : altitude of the aircraft at the predicted time.
- \( x_T \) : distance of of the aircraft from the airport, at the predicted time.
- \( \text{vr} \) : relative velocity of wind.
- \( \text{esf} \) : energy share factor[17].
- \( \text{deltasisa} \) : difference of the atmospheric pressure from \( z_0 \) to \( z_T \).
Figure I.4: Altitude change

\[ Dz[i] = z[i-1] - z[i] \]

\[ Dz1 = z0 - z1 \]

\[ Dz9 = z8 - z9 \]
Figure I.5: Distance change

\[ D\dot{x}[i] = x[i-1] - x[i] \]

The traveled distance from the airport at the 3rd plot (t_predict) after x0.

The traveled distance from the airport at the 1st type (t_predict) plot after x0.
Figure I.6: Double predictor

- cas : CAS.
  \[ \text{deltaCAS} = \text{CAS}(t_0) - \text{CAS}(t_0 - \delta t) \]
- mach : MACH.
- \( \text{deltaMach} \) : difference of mach between the precedent calculated mach and the current.
  \[ \text{deltaMach} = \text{Mach}(\text{time of } z_0) - \text{Mach}(\text{time of } z_0) - (\text{difference of time}) \]
- \( Dz[i] \) : absolute difference of the \( z[i-1] \) and \( z[i] \).
- \( z_0 \) : altitude of the aircraft at the point in which we begin the prediction, the base point.
- \( x_0 \) : distance of the aircraft from the airport, at the point in which we begin the prediction, the base point.
- \( Dx[i] \) : absolute difference of the \( x[i-1] \) and \( x[i] \).

Since we have to predict \( xt \) and \( zt \) at the time \( t_{\text{predict}} \) and we know that the altitude changes much more in the climb phase than the distance travelled from the airport, the simplest way is to predict at first, \( xt \).
Once we have the $x_t$, we can in a second attempt predict the $z_t$. Another way to begin the trajectory prediction is to predict just the $z_t$ and once we have satisfactory results at this level we can expand our model to include also the $x_t$. In that case we won’t use $dx[i]$ and $x_t$ in the training set. We've tested this method and the results are available in the experiment chapter. In this chapter we've seen that our objective in the aircraft trajectory prediction is to do a prediction which diminish as much as possible the miss-detected conflicts. The aircraft trajectory is composed of a climb, cruise and descent phase. In all these phases there are laws which govern the speed, altitude and other parameters of the flight. First of all, even before the begin of flight, the aircraft’s trajectory is, computed in FMS. This is a theoretical path done based on the optimization of time and other constraints in FMS. Once the aircraft begin its flight we want to predict its location to detect the possible conflicts in a sector. It is too complex to use a machine learning algorithm with model that take into account all the available parameters which influence the trajectory prediction, so we try to restrict our parameters in a coherent and smaller subspace.

In the next chapter, we will take a look at the background of machine learning algorithms that can be used to build the trajectory predictor.

The trajectory prediction task involves three different phases (climb, cruise and descent) each with entirely different steps. For the first attempt, it was too time-consuming to make a model able to learn all phases, thus we’ve began by making a climb phase predictor. Accordingly to our precedent choice we filtered the training set by removing climbs with plateau. Plateau in a climb phase shows the intervention of an air traffic controller which is not possible to predict therefore
we don’t try to learn it. In any cases, our goal is the prediction of the aircraft’s trajectory not the air traffic controller’s intervention so as to help the air traffic controller to make the better choice.

We’ve used the database of CAUTRA system of Paris’s Orly airport. The CAUTRA system merges the received data from several radars so here, for each flight we have a position plot every 1 to 3 second. Finally these data have been locally smoothed with a linear model. The training set has also been restricted to one aircraft’s model. Since each aircraft performance model imposes certain restriction on its speed, optimal speed and other parameters, therefore the climb phase of two different model may be totally different, for example a Fokker 100 will not climb in the same manner as a Boeing 747, thus we’ve restricted our training to Airbus 320’s climb trajectory without plateau.
I.3 Summary in French

Aujourd'hui, les modèles de prédiction utilisés sont basés sur une modélisation physique de l'avion et de son environnement. Les paramètres employés par ces modèles sont les suivants : le modèle de l'avion, le modèle météo et les intentions du vol. Le modèle d'avion comprend les données et les équations qui permettent de décrire la trajectoire de l'avion en vol. Le poids de l'avion est un paramètre critique, or il n'est pas fourni de manière précise par les compagnies aériennes. Le modèle météo est naturellement toujours imprécis et enfin, les intentions de vols ne sont presque jamais connues au sol. Au bord de l’avion, le pilote dispose d’un prédicateur de trajectoire avec lequel il ajuste les paramètres de vol afin de respecter les contraintes du vol. Ce prédicateur utilise le modèle à énergie totale pour prédire la trajectoire de l’avion. Comme le modèle à énergie totale utilise la masse, la vitesse et l’accélération de l’avion, qui ne sont pas connues au sol, il ne peut pas être utilisé comme prédicateur sol. La trajectoire d’avion est constituée de deux profile: le profile latéral et le profile verticale. Le profile latérale est constitué de lignes droites et de des segments de virages. Elle décrit la trajectoire de l’avion sur la carte. Le profile verticale contient trois phases importantes : la montée, la croisière et la descente. La phase de montée, est elle même constituée de plusieurs petits phases. A la fin de cette phase on arrive à la fin de la montée et au début de la phase croisière. Dans la phase de croisière , l’avion vole à une altitude presque constante. Son altitude change légèrement parce qu’en consommant son carburant il devient plus léger, donc son altitude et sa vitesse de croisière qui sont corrélées changent légèrement dans cette phase. Il peut aussi changer d’altitude par rapport aux ordres de contrôle qui lui sont transmis quand il change de secteur. Dans la décente l’avion commence par le point du sommet de la descente jusqu’à boîter sur le sol. Quand au données de trajectoire en informatique, il faudra stocker tous les informations qui sont liées à un vol, qu’elles soient directement liées à l’avion, comme le modèle de l’avion ou qu’elles soient indirectement liées comme le modèle de météo. On a utilise des données provenant de plots de radar pris dans des intervalle de 1 à 3 secondes . Puis on a échantillonné un sous ensemble de plot qui décrit le le trajectoire de l’avions tous les 15 secondes. Si dans un pas de 15 secondes, on ne trouve pas de plot de radar, on a pris la moyenne entre le plot suivant et le plot précédent. Pour une prédiction complète, il faut prédire dans le profile vertical et dans le profile latéral. Pour ce rapport on a restreint nos recherche à une prédiction d’altitude dans phase de la montée. La prédiction du profile latérale dans la montée sera plus facile, parce que dans les montée , la vitesse vertical est bien supérieur à la vitesse latéral. Il ne faut pas oublier que ces deux prédiction sont corrélées, c’est d’ailleur pour cette raison que dans notre sous ensemble échantillonnée, on utilise la distance parcouru par l’avion de l’aéroport. Dans notre structure de donnees chaque instance modélise 10 pas de temps consécutifs de l’avion plus le pas de temps à prédire. Ce qui fait en tous 10 pas de temps consécutif et un 11ème utilisé pour l’apprentissage. Notre modèle de régression sera entrainé avec des prédicitions de 1 à 40 pas de temps. En d’autre terme, nous avons des instances de 11 pas dont les 10 premiers sont consécutifs et le 11ème pas, peut être un pas de 1 à 40 qu’on doit pouvoir apprendre la prédiction.
Chapter II

Background

According to the online dictionary Wordreference\(^1\) regression means "returning to a former state" and it is used in psychology and statistics. Since this text is about a Master thesis in computer science we will develop the statistical meaning of regression. In this chapter we begin by taking a look at regression and the analogy behind it. Then we will see some common methods of regression tested against our result set. Next, we will see fuzzy regression and KNN regression. These two concepts are the fundamental part of this text. Fuzzy regression and more precisely imprecise regression is the idea behind this work. We want to demonstrate that imprecise regression works better where we can’t obtain an enough precise result, that’s why conventionally we used Gaussian 95% quantile of the result so as to have an interval which contains most surely our response. Then we will take a deep look at KNN regression and different distance functions, because the main part of our experiments are developed using KNN regression.

II.1 Regression

In statistics, regression analysis are the techniques for analyzing relationship between several variables. The focus is to find the relationship between a dependent variable and one or more independent variables. In other word, the goal is to understand how the value of the dependent variable changes when we change one variable of the independent variables part and fix the rest. The estimated variable is a function of the independent variables called the regression function. One of the big area of its application is the prediction in machine learning. [3]

II.1.1 Regression analysis: A little more

Regression analysis is the prediction of a variable (or a set of variables) denoted by a function of independent variables, denoted X (a vector of \(x_i\)). \(Y=f(X)\) means that we have a regressor that predicts the value of Y based on the input vector X.

The regression "model" relates Y to a function of X and it includes the following variables.

\(^1\)On line dictionary http://www.wordreference.com.
II. BACKGROUND

- The unknown parameters denoted as it can be a scalar or a vector of length K.
- The independent variables, X.
- The dependent variable, Y.

\[ Y \approx f(X, \beta) \]

Based on the model type, the assumption for each model changes, for instance in linear regression models, we make the assumption that the regressors \( x_i \) are error-free, which means that they are not contaminated with measurement errors. Although not always realistic, removing the former results in a much more difficult errors-in-variables models. However for \( n \geq k \) (n data points of the form \((Y,X)\) ) is a common assumptions to be held for most classical approaches in regression analysis.

Until now, a lot of techniques were developed in regression analysis. When the regression function is defined in terms of a finite number of unknown parameters estimated from the data, it is a parametric regression like linear regression. Nonparametric regression function is defined by a set of functions, which may be infinite-dimensional. In these methods the predictor does not have a predetermined form, it is built by the information derived from the data like neural networks and SVMs. It is obvious that nonparametric regression requires a larger set of data because the data supplies the model structure and the model estimate.[1]

II.1.2 Normal distribution of the error

Generally in regression we minimize the sum of the squares of the errors. There are two reasons behind it, first of all Gauss and Legendre have shown that by minimizing the squared error, we can have a solvable linear system of equations (in case of linear regression). Second, if we minimizes the sum of the squares of the errors we will have a line which lies the means of distribution associated with the input variables, but this later occurs only, if the error distribution is Normal. When the data distribution is not normal, the regression function will draw us a curve or a line located inside the error distribution, which in contrary to a Normal distribution, is not the mean[18].

II.1.3 Linear Regression

In statistics, linear regression is the set of techniques, employed to model the relationship between a scalar variable y and a vector of input variables denoted X, in a way that the model depends linearly on the unknown parameters estimated from the data. This is called a “linear model”.

\[ y = X\beta + \varepsilon, \]

where
II. BACKGROUND

19

Figure II.1: Example of linear regression with one independent variable.

\[
y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}, \quad X = \begin{pmatrix} x'_1 \\ x'_2 \\ \vdots \\ x'_n \end{pmatrix} = \begin{pmatrix} x_{11} & \cdots & x_{1k} \\ x_{21} & \cdots & x_{2k} \\ \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{nk} \end{pmatrix}, \quad \beta = \begin{pmatrix} \beta_1 \\ \vdots \\ \beta_k \end{pmatrix}, \quad \varepsilon = \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{pmatrix}.
\]

- \( y_i \), is dependent variable.
- \( x_i \)'s, are called regressors.
- \( \beta \), is a k-dimensional parameter vector.
- \( \varepsilon_i \), is called the error term, disturbance term, or noise.

Two necessary assumptions are:

- The regressors in X must all be linearly independent. The design matrix X must have full column rank k. In other word, there must be enough data available compared to the number of parameters to be estimated \( n \geq k \).
- The regressors \( x_i \) are assumed to be error-free.
II.1.4 Neural network

A biological neuron, is the basic structure of our central nervous system. A group or groups of physically interconnected neurons forms what is called a neural network. It is the place that we store our information and we use when we are thinking about something. After the hypotheses of learning made by Donald Hebb in the late 1940s, Frank Rosenblatt invented the perceptron. The perceptron is the simplest kind of feed forward neural network: a linear classifier invented in 1957 at the Cornell Aeronautical Laboratory [19].

The perceptron is a binary classifier which maps a input vector $X$ of real values to an an output binary value $f(x)$.

$$f(x) = \begin{cases} 1 & \text{if } W \cdot X + b > 0 \\ 0 & \text{else} \end{cases}$$

The value of $f(x)$ is used to classify the input vector $X$ as positive or negative. As you can see, here we have a sign activation function. If the dot product of $W$ which is the weight vector and the input vector $X$ summed by $b$ is greater than zero, then the "activation function" is fired and the input instance is classified as positive otherwise it is classified as negative.

$$f(x) = \text{sign}(W \cdot X + b)$$

But in real-world problems, we want to learn something that is not binary nor linearly separable, Thus we will use more than one perceptron organized in many layers. Multilayer Perceptron (MLP) is a nonlinear classifier that maps a sets of input data onto a set of appropriate output with three or more layers of perceptron. The process of learning is done with the propagation of the error from the output node to the precedents layers until to the input nodes. This was just after the backpropagation algorithm of David E. Rumelhart, Geoffrey E. Hinton and Ronald J. Williams, that we had an applicable algorithm to use in the learning task of the MLP [4]. Each perceptron has a nonlinear activation function which must be differentiable $^2$.

The nonlinear activation function was developed to model the frequency of firing, of biological neurons in the brain. The two main activation functions currently used in applications are both sigmoid, and are described by

$$\phi(y_i) = \tanh(v_i) \text{ and } \phi(y_i) = (1 + e^{-v_i})^{-1},$$

Here $y_i$ is the output of the ith perceptron and $v_i$ is the weighted sum of the input vector.

Artificial Neural Networks (ANN) are applied in the current applications, however the MLP is not the unique type of ANN. Hopefield networks are ANN where the output of the node is fed back as their input. Self-organizing maps are ANN used in unsupervised learning to get discretized representation of the input space. Neural networks are used both in supervised and unsupervised learning, and in classification and regression.

$^2$The backpropagation requires differentiable activation function to propagate the error from the output node to the preceding layers [4]
II. BACKGROUND

II.1.5 Support vector machine (SVM)

SVM are a set of supervised learning methods employed in classification or regression. In classification, using the support vector machine we try to classify a data point in a n-dimensional vector, and we want to know whether we can separates such points with a n 1-dimensional hyperplane. There are a lot of hyperplanes that can classify the data. An option for the best hyperplane is the one that separates as more as possible the two classes. In SVM, we choose the hyperplane that represents the largest separation, or margin, between the two classes. In other word the one so that the distance from it to the nearest data point on each side is maximized. [21].

Given some training data $D$, a set of n points of the following form

$$D = \{(x_i, c_i) \mid x_i \in \mathbb{R}^n, c_i \in \{-1, 1\}\}_{i=1}^k$$

where the $c_i$ is either 1 or -1, indicates the class to which the point $x_i$ belongs. Each $x_i$ is a real vector of n-dimension. We are looking for the the maximum-margin hyperplane which divides the points with $c_i = 1$ from those with $c_i = -1$. Any hyperplane which separates these points can be written as the set of points $x$ satisfying

$$w \cdot x - b = 0,$$

Finally we will end by the following optimization problem.
Minimize (in $w, b$)

$$\|w\|$$

for any $i = 1, \ldots, k$

$$c_i(w \cdot x_i - b) \geq 1.$$
If the data is not linearly separable, the data is projected in higher dimension subspace and then classified using hyperplanes in the higher subspace \[21\]. A regression version of support vector machine exists which is suggested by Vladimir Vapnik. least squares are also applied to support vector machine, this version is called least squares support vector machine (LS-SVM) and is proposed by Suykens and Vandewalle \[12\].

II.2 K-nearest neighbor (KNN)

The k-nearest neighbor(KNN) algorithm is a classification method which classifies the input data based on closest training examples in the feature space. It is a local method, for the classification it doesn’t look the whole set, it just estimate the response, with the k instances which already exists in the training set. It doesn’t have a learning phase so you can find it in the lazy learning category of learning algorithms.

II.2.1 KNN classification

Originally KNN was made for classification. In KNN classification, we look at the K nearest instance and we choose the class that has the majority. Let’s assume that \( X \) is a \( d \)-dimensional input instance to be classified:

\[
X = \{ x_1, \ldots, x_d \}
\]

We have \( C \) classes as the response value

\[
Y \in Y_1, \ldots, Y_c
\]

And the training set is composed of \( n \) instances
II. BACKGROUND

Figure II.4: Example of non-weighted KNN classification.

for any \( i \in 1, \ldots, n \) \( X_i = \{ x_{i1}, \ldots, x_{id} \} \)
\[ \text{Kset} = \{ \text{instances in K nearest neighbors of X } \} \]
\[ K(c) = \{ X_i \mid (X_i, Y_c) \in \text{Kset} \} \]

\[ f(x) = \arg \max_y \sum_{X_i \in K(c)} d(X, X_i)^{-1} \]

where \( d(X, X_i) \) is the distance between the input X and the instance \( X_i \) located in Kset. This was the weighted version of the KNN where each instance is weighted by the inverse of its distance to X. To obtain the non-weighted version we have just to replace the \( d(X, X_i)^{-1} \) by 1.

The figure II.4, shows an example of non-weighted KNN classification. The X instance (green circle) should be classified either to the class of blue squares or to the class of red triangles.

If \( K = 3 \) it is classified as red triangle else if \( K = 5 \) it is classified as blue square (3 squares vs. 2 triangles inside the outer circle).

II.2.2 KNN regression

The regression version of KNN is just like finding a weighted average of the K-nearest neighbor. Since we are looking for a real value, weighting takes a more important role in the regression. Let’s formalize this method:

\[ f(x) = \frac{\sum_{X_i \in \text{Kset}} W_i \cdot Y_i}{\sum_{X_i \in \text{Kset}} W_i} \]

Where \( W_i = d(X, X_i)^{-1} \)

In the figure II.5, we can see an example of weighted KNN regression using \( K=5 \), where the response value is a weighted regression of the 5 instances inside the circle.
II.3 Fuzzy regression

Although statistical regression has many applications, in the following cases, it is not the best solution to adopt.

- Small number of observations
- Arduous to verify the assumptions of distribution
- Ambiguity in the relationship between the independents and the response variables
- Uncertainty in degree of the event
- Inaccuracy and bias introduced by linearisation

As a result, the statistical regression model is not useful in the above cases. In contrary fuzzy regression, is a nonparametric method, which can be employed in estimating the relationships among variables where we dispose of limited and imprecise data, and the variables are interacting in an uncertain, qualitative, and fuzzy way. In this section we begin to see the definition of the fuzzy sets, then we take a brief view on the most common fuzzy regression methods and at last we end up by describing imprecise regression.

II.3.1 Fuzzy set

Computers do not reason as humans do. Computers "reason" when they are working with binary data that represents precise facts but humans reason with vague assertions and facts that involve
II. BACKGROUND

imprecision and uncertainty like: "He is young", "the air is cool" or "your are strong". Fuzzy set is an effort in computer science to represents the vagueness and uncertainty that humans have tendency to associate with facts.

In 1965, Lotfi A. Zadeh introduced the notion of fuzzy set as an extension of classical notion of set. In classical set theory, each element has a binary membership degree, it either belongs to a set or does not belong to it. However, in fuzzy sets, membership degree of elements to a set is between 0 and 1 [23]. It is like in real world that objects may have imprecise defined criteria of membership, For instance a 30-years old person may be a member of the set little young , young, a bit old and old. In contrary, in classical set theory the set old and young are disjoint and the former example isn’t possible, however in fuzzy set a 30 years-old individual doesn’t belong to the set young with a membership degree of 1, but it membership degree is between 0 and 1.

Formally, a fuzzy set is represented as a pair \((A,m)\) like below:

\[ A \rightarrow [0, 1], \text{where} \ A \text{is a set and} \ m \text{and for each} \ x \in A \text{we call} m(x), \text{the grade of membership of} \ x \text{in} \ (A,m). \]

Fuzzy set have a wide domain of application, they are useful where the information is incomplete or imprecise, like in : social science, industrial engineering, economics, geography, computer engineering and etc. In our context we represents the result of a trajectory prediction as a fuzzy set.

II.3.2 Fuzzy regression methods

Fuzzy regression for data may be done with three categories of variables. The first type is when the independent and the dependent variables are crisp data. In this case just the relationship

Figure II.6: Coolness of the air, represented in fuzzy.
between the regressor variables and the response variable is fuzzy. In the second case the input data is crisp but the output variable is fuzzy and the third one this is where both input and output data are fuzzy.

In the case of fuzzy regression with fuzzy response variable, there is Diamond’s method which is a based on a extension of the least square fuzzy error minimization.[6]. In [18] the authors say that this approach is not applicable on crisp input and output variables because it will be the same as the conventional regression. Another approach while always treating fuzzy response variable with crisp data, is to find the optimum interval with the upper and lower bounds calculated separately and take this interval as the possibilistic output. This method is very sensitive to outliers [18].

Recently there have been some works on ”imprecise regression” which is the main subject of this master thesis, and we will discuss it in the next paragraph.

II.3.3 Imprecise regression

Knowing that the representation of the example is an incomplete representation of the world, searching for a precise model when dealing with precise data may not be the best solution. Recently, H. Prade and M. Serrurier [18] have introduced the concept of imprecise regression to overcome the problem of methods with illusory precision. Hence with a crisp set of input data we will look for a model that is as precise as possible with a faithful description of the data. [18]

Learning is usually biased by uncertainty caused by noisy data and description language. Epistemic uncertainty is used to handle the description language uncertainty and probabilistic methods can handle the noise. Imprecise regression describe the data by fuzzy set which represents a family of probability distributions. Fuzzy sets are also a well known tool for representing the epistemic uncertainty[18].

Imprecise regression allows us to represent the imprecision of the model by taking into account the incompleteness of the information caused by the data and the representation of the hypothesis. A regression data set is a set of pairs \((\overrightarrow{x}_i, y_i)\), \(1 \leq i \leq m\) where \(\overrightarrow{x}_i \in X\) is a vector of \(n\) input variables and \(y_i \in \mathbb{R}\) is the real output variable. An imprecise fuzzy function \(F\) is from \(X\) to \((\mathbb{R} \rightarrow [0, 1])\) and associate a distribution of possible output values to the input vector \(\overrightarrow{x}\). In imprecise regression we search the fuzzy function \(F(\overrightarrow{x})\) that maximize the function below:

\[
\text{Info}(F) = -\sum_{i=1}^{m} \pi_i(y_i) \ast \log\left(\frac{\text{Area}(\pi_i) + \text{Area}(A(\pi_i(y_i)))}{2 \ast \text{Area}(\pi_{\text{max}})}\right) \tag{II.1}
\]

Using this evaluation function we will maximize the accuracy of the imprecise function and its precision and it ensure a trade-off between accuracy and precision of the model[18]. In this master thesis we want to demonstrate that fuzzy set found using the former evaluation function, may fit better our data set than regression with Gaussian assumption. In the experiment part we will study numerical results.
II.4 Summary in French

La régression est une méthode statistique très utilisée pour analyser la relation entre une variable par rapport à une ou plusieurs autres variables. On essaye de trouver une estimation des paramètres inconnus du modèle de régression avec un ajustement mathématique d’un modèle spécifié. Cet ajustement se fait en fonction des données récoltées. Donc la précision des données récoltées a une forte influence sur l’efficacité du modèle de régression. La méthode des ” moindres carrés ” est l’une des méthodes les plus employées pour obtenir un modèle estimé. Sous certaines hypothèses, les estimateurs trouvée en employant la méthode des moindres carrés sont les meilleurs estimateurs parmi lesquelles on peut trouver les estimateurs linéaires et non biaisés. L’une de ces hypothèses est la distribution normale des erreur de prévision. Si les erreurs de la prévision sont distribuées de façon Gaussiennes, en optimisant les moindres carres, on arrivera à trouver un modèle qui prédira la moyenne des erreurs. Mais il se trouve que dans certaine cas cette distribution n’est pas gaussienne cette méthode ne s’avère pas la meilleure.

Nous avons également vu que nous étions pas capable de faire une précision exacte et nous cherchions des méthodes pour prendre en compte cette incertitude ou imprécision. Donc, il faudra prédire une intervalle qui contiendra dans la plupart des cas l’emplacement de l’avion. Puisque dans les méthodes de régression, nous estimons les moindres carres, l’hypothèse de la loi normale sur cette intervalle est toujours la plus avantageuse. Mais si l’emplacement de l’avion dans cette intervalle ne suit pas une loi de probabilité normale et on ne peut pas trouver une autre loi de probabilité déjà connu, alors même si nous avons une intervalle dont les avions sont dans la plupart des cas situées dedans, On arrivera quand même pas a exprimer notre incertitude sur cet espace. La régression Imprécise est une méthode de régression capables de fournir des résultats avec les meilleurs compromis entre l’exactitude et la l’incertitude sur les valeurs prédites. Cette approche est utilisée quand la distribution des données proviens d’une famille de distribution et non d’une seule loi de distribution. La régression imprécise utilise les ensemble flous pour associer un degré de incertitude a l’intervalles de la prévision. Il existe aussi d’autre approches de régression pour prédire des intervalles. Certaines approches utilise juste une intervalle, d’autre les ensemble flous. Il y a aussi une approche qui applique les moindres carres aux ensembles flous. Comme nous avons vu précédemment les moindres carres sont plus avantageux pour des distribution gaussienne que les provenant d’une famille de distribution ou d’une loi non normale. La régression elle même comporte plusieurs méthodes.

La régression linéaire est l’une des méthodes les plus utilisées quand la variable à prédire est distribuée linéairement. Les réseaux de neurones utilisent plusieurs couche de neurones artificielles pour faire la régression. Jusqu’à la découverte de l’algorithme de rétropropagation , on ne pouvait pas les utiliser pour pour une régression non-linéaire. Cette algorithme propage les erreur de sortie sur les poids des sorties des couche précédant jusqu’à au poids des entrées de la couche d’entrée. Les vecteur a vaste marge (SVM) s sont une autre méthode très utilisées dans la régression non-lineaire. Ils sont une généralisation des classifiants linéaires. Dans les SVMs on traite des problèmes de discrimination non-linéaire, en reformulant le problème de classement comme un problème d’optimisation quadratique. Dans les SVMs, on choisie la frontière de séparation en maximisant la marge, qui elle est définie comme étant la distance entre la frontière de séparation et les instances les plus proches. En cherchant la frontière séparatrice optimale,
on reformule le problème de la classification en un problème d’optimisation quadratique, pour lequel il existe des algorithmes connus. SVM a été d’abord crée pour la classification et puis adapté à la régression. Les méthodes de régressions que nous avons vue sont tous des méthodes globales. Quand les ensembles de données sont denses, on peut aussi employer les méthodes locales. La régression avec les K plus proches voisins est une méthode de régression locale. Elle est utilisée quand les données sont assez dense et l’ensemble d’apprentissage peut décrire la distribution de l’espace à apprendre. Dans le chapitre suivant nous verrons avec plus de détail, cette approche. Après avoir choisi la méthode de régression nous devrons pouvoir trouver une approche pour traiter le problème de l’incertitude des résultats. La régression imprécise avec les K plus proches voisins est la méthode que nous avons utilise dans ce travail. Nous avons aussi fait des tests dans lesquelles on a comparé ces différentes méthodes. Nous verrons les résultats des tests dans le dernier chapitre.
Chapter III

Our approach

In this chapter we focus more on what we developed and tested during this master thesis. We used KNN as our regression algorithm, and concentrated our effort to find a distance function which gives better regression results. We introduced two new distance functions named area between trajectory and constrained distance. This chapter is organized as follow: we begin with a survey on distance function notion, then we will see different distance functions and finally we end up by exploring different ways of finding regression intervals.

III.1 KNN Regression

The details of the KNN regression algorithm has been previously shown in II.5. It is important to say that like other local methods, this algorithm can be used when we have dense training set which is our case. To find the nearest neighbours of the input data, KNN must to use a distance function and this function has a large impact on the effectiveness of the algorithm. Using the same algorithm parameter but with a better distance function which effectively expresses the distance between instances, we can have much better results and that is what we will see later in the experiments chapter.

III.2 Distance function

The distance notion has a deep impact on the effectiveness of a machine learning algorithm. A distance function is a function that expresses the difference of instances in the feature space. This difference can also be bounded. \( D : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R} \), \( X_i, X_j \in \mathcal{X} \)

If the result value of the distance function is positive only it is a distance metric \(^1\) and obeys the following properties:

- **Isolation**: \( D(X_i, X_j) = 0 \iff X_i = X_j \)
- **Symmetry**: \( D(X_i, X_j) = D(X_j, X_i) \)

\(^1\)We use the term distance function for both distance function and distance metric.
• Triangular equality: \( D(X_i, X_j) + D(X_j, X_z) \geq D(X_i, X_z) \)

It is not the case for all distance functions. Kernel functions are widely used as distance functions. A kernel function is defined like below:

\[
K(X_i, X_j) = \langle \phi(X_i), \phi(X_j) \rangle
\]

Where \( \langle a, b \rangle \) is the inner product of the vector a and b and \( \phi : X \rightarrow \phi(X) \) is a mapping from X to an inner product feature space [10].

Although metric distance functions (distance metrics) are widely used, they don’t give robust results as they fail outliers. Jacobs et al. have shown that robust distance functions on outliers do not respect the triangular inequality [13]. Furthermore Tyversky has shown in its wide judgement on human similarity judgement that it often violates both the symmetry and the triangular inequality [5] and finally in [16] it is shown that to obtain an optimal distance function for KNN classification in object discrimination task, we have to use functions that violate the “isolation” property. Therefore for different domain, application specific distance functions which fail to obeys the mentioned metric properties have been designed. For further study you can see a list of them in the section I.2.6 of [10].

In the next paragraphs we will distance functions which are used in the experiment chapter of this text, It is not without interest to stress that, beside conventional distance functions or application-specific distance functions, there are some research done to learn the distance functions for machine learning problems [10] [7].

### III.2.1 Euclidean

Euclidean distance is the most used distance function (also a distance metric), It is also known as \( L_2 \) distance, it is defined by

\[
D_{Euc}(X_i, X_j) = \sqrt{\sum_{k=1}^{d} (x_{ik} - x_{jk})^2}
\]

This version of Euclidean doesn’t put any difference between variables, it is a good distance metric when we have non-correlated variables but when the variables are correlated it is not really effective. Let’s see an example: We have three independent variables \( x_1, x_2, x_3 \) and \( x_4 \), where just \( x_1 \) and \( x_2 \) are correlated, and a response variable \( y \). Since \( x_1 \) and \( x_2 \) are correlated a change in \( x_1 \) will cause a change in \( x_2 \). When using Euclidean distance function we take account of the change in \( x_1 \) and \( x_2 \) in the same manner that we take account of the changes in \( x_3 \) and \( x_4 \), but it is not correct. Because a change in \( x_1 \) and \( x_2 \) might be because of the change in just \( x_1 \) or just \( x_2 \) but a change in \( x_3 \) and \( x_4 \) means that the two variables \( x_3 \) and \( x_4 \) changed independently.

Another problem of Euclidean distance and other non-normalized distance function, is that they are influenced by the range of the variables. To overcome this problem we use the formula below to normalize the data set :

\[
\text{Norm}(x_i) = \frac{x_i - x_{\text{max}}}{x_{\text{max}} - x_{\text{min}}}
\]
III. Our approach

Where $x_{\text{max}}$ and $x_{\text{min}}$ are respectively, the maximum and minimum of the variable $x$ in our dataset. So a variable with a range from 0 to 1 has the same impact than a variable which its range vary from 0 to 1000, and Finally we can weight the the variables to differentiate between variable and let some variables have more impact than others on the distance function.

III.2.2 Mahalanobis

This distance function, is based on correlations between variables of the by which the different pattern can be identified. It is a generalization of the Euclidean distance in way that it is scale-invariant and takes into account the correlations of the data set. It measures the distance of any two vectors, when assuming that they have the same data distribution as origin.

$$D_{\text{Mah}}(X_i, X_j) = \sqrt{\sum_{k=1}^{d} \sum_{l=1}^{d} x_{ik} \sigma_{lk}^{-1} x_{jl}}$$

Where $\sigma$ is the covariance matrix. If the covariance matrix is the identity matrix then the Mahalanobis distance is the same as the Euclidean distance and if it is diagonal we have a normalized euclidean distance. [10].

Mahalanobis distance has wide application in cluster analysis and classification. It is used instead of the Euclidean distance where the data has a non spherically symmetric distribution. In linear regression models Mahalanobis distance is used to detect outliers because points with greater Mahalanobis distance from the rest of population of data points have a greater influence on the coefficients of the regression equation.

III.2.3 Constrained distance

We introduce this distance function to use when we don’t have a ”good” distance function but we dispose of enough domain specific information about the variables and their correlation with the output variable, it is not without interest to constrain one or more independent variables. Once the neighborhood of the input data is filtered with these constraints the distance function can be used to extract the nearest neighbors.

First the feature space is filtered like the below:

$$\psi = \{ Z \mid \text{for each } \alpha \in \rho, \ \alpha(Z) = \text{true} \}$$

where $\rho$ is a set of function which describe a constraint on the feature space.

Then we find the Kset

$$Kset = \{ \text{instances located in } \psi \text{ that are in K nearest neighbors of } X \}.$$ Now that we have the Kset we can continue with our distance function in the same way as in KNN regression.
III.2.4 Constrained distance with relaxation

The distance function above may be useful in some area where the data set are dense. But in some cases we do not dispose of enough instances to provide the K nearest neighbors, so there are input instances for which we can’t find neighbors satisfying the constraint. The key to this problem is to relax the constraints. If one have enough domain knowledge to prioritize the constraint to relax, it might be the best solution, otherwise we suggest two approaches: first to relax all the constraints with the same ratio, second to relax the constraint that has the greatest mean ratio of violation. The relaxation is done in each nearest neighbors search until we reach to the K number or a portion of the K number.

Algorithm 1

Find the K-nearest neighbors of the input
1: round ← 0
2: repeat
3: count ← 0
4: for all i in training set do
5: if satisfyConstraints($X_i$) and count $\leq K$ then
6: Add $X_i$ to KSet
7: count ← count + 1
8: else if count $> K$ then
9: remove the farthest from KSet
10: end if
11: end for
12: round ← round + 1
13: until $KSetSize < k \times percentFit$ and $percentFit \neq -1$ and round $< 30$
14: if $KSetSize < k \times percentFit$ then
15: KSet ← conventional KNN Search
16: end if

In the algorithm 1 you can see how the Kset is filled with the nearest neighbors. The repeat loop is done until we reach the percentFit. The percentFit is a real number between 0 and 1 which means the percent of K that satisfies our constraints which can be accepted. In our algorithm we end the repeat loop after 30 iterations, but it depends on the dataset. Finally if we don’t find the percentFit of the K we just do a conventional KNN search to find the K nearest neighbors. In the algorithm 2, the function which is responsible to check the satisfaction of the instances is described. In line 9 you can see the term $\frac{|X_i - X_j| - (relaxValue + adjustPas)}{relaxvalue}$ which describes the ratio of violation of the current constraint in the loop, when comparing the $i^{th}$ attribute of the 2 instances $X_i$ and $X_j$. Since in each nearest neighbors search in the algorithm 1 we need to relax more the selected constraint, as you can see in line 5 of the algorithm 2, we use the count value in the calculation of the value of ”adjustPas”. 
Algorithm 2 Check if the instance satisfies the constraints

1: \( \text{ret} \leftarrow \text{true} \)
2: \textbf{for all} \ relaxValue \text{ in all constraints } \textbf{do}
3: \( \text{adjustPas} \leftarrow 0 \)
4: \textbf{if} relax the constraint \textbf{then}
5: \( \text{adjustPas} \leftarrow 0.25 \times \text{count} \times \text{relaxValue} \)
6: \textbf{end if}
7: \textbf{if} \( |X_{ii} - X_{ji}| > \text{relaxValue} + \text{adjustPas} \) \textbf{then}
8: \( \text{oldViolationValue} \leftarrow \text{the old sum of violation value for the current constraint} \)
9: \( \text{newViolationValue} \leftarrow \frac{|X_{ii} - X_{ji}| - (\text{relaxValue} + \text{adjustPas})}{\text{relaxvalue}} \)
10: \( \text{violIndices} \leftarrow \text{store the (oldViolationValue + newViolationValue) with the index of the current constraint} \)
11: \( \text{ret} \leftarrow \text{false} \)
12: \textbf{end if}
13: \textbf{end for}
14: \text{return ret}

III.2.5 Area between trajectory (ABT)

As with various domain, we can define domain-specific distance functions. Here we introduce the ABT distance function which calculates the area between two trajectories, then we use it as a distance to differentiate the two trajectory instances. It is calculated as follow:

\( t-9, ... t0, t \) is our time serie and we dispose of \( z_{t[i]}, x_{t[i]} \) where \( i \in \{-9, ..., 0\} \)

\[
D_{ABT} = \sum_{i \in \{-9, ..., 1\}} \text{area}(T_{cor}[i]_{x1}, T_{cor}[i + 1]_{x1}, T_{cor}[i]_{x2}, T_{cor}[i + 1]_{x2})
\]

Where \( T_{cor}[i]_{x1} \) is the coordinate \((x, z)\) of the instance \( x1 \) at the moment \( i \), so \( (x_{t[i]}, z_{t[i]}) \) and the \text{area}(a, b, c, d) function, is a calculation function which calculates the area between the four points \( a, b, c, d \) in a two- dimensional space.

In the figure III.1 you can see an example of ABT distance, the total area between the 10 data points (t-9 ... t0) of the two trajectories are calculated.

This distance function can also be normalized, or work just with normalized z or just normalized \( x \) variables.

III.2.6 Area between trajectory 2nd version (ABTv.2)

This distance calculates the area between the trajectory with a slightly different mode. It doesn’t take into account the altitude and the \( x \) (distance traveled from the airport) of t-9, in consequent we subtract the altitude of t-9 from the altitude of all the next data points. So after the subtraction operation, the altitude of t-9 will be 0 and the t-8’s will be \( Dz9 \), and so on, this will be the same case for \( x \).
Figure III.1: ABT distance.
III. Our approach

III.2.7 Hybrid ABT/Euc

ABT distance function is just a linear combination of t-9 to t0’s locations, however in each data point we have more variables than these location variables. Since they give us some information, we will calculate the Euclidean distance of these resting variables and create a hybrid ABT/Euc distance which takes into account all the variables. In this distance function we sum the result of the ABT distance with the Euclidean distance of all the non-location variables.

\[ D_{Hybrid}(X_i, X_j) = D_{ABT}(X_i, X_j) + D_{Euc}(K_i, K_j) \]

Where \( K = X - \{z0, \ldots, z0, zt, x0, \ldots, x0, xt\} \) variables.

III.3 Regression interval

The goal of using a regression interval in our prediction task is to calculate the aircraft’s location at \( t_{predict} \). As we’ve seen before we’re not able to obtain precise results, therefore we have to estimate the most probable or possible interval which includes our aircraft location. This is also the goal of fuzzy regression.

Let’s take a look at the distribution of the k-nearest neighbors of an input instance. Due to the fact that, in local methods (in this case KNN) we make the assumption that the estimated data point must be similar to its nearest neighbors, the regression interval must be the interval which is the more representative to reflect the probability or the possibility of the nearest neighbors distribution.

**Algorithm 3** Calculate the regression interval in KNN

1: Kset ← Find the K-nearest neighbors of the input
2: finalValue ← KNNregression(Kset)
3: regressionInterval ← buildInterval(Kset,finalValue)

In this section we will take a look on two different methods of calculating regression intervals, the first one uses Normal distribution and the second one employs fuzzy sets. The regression interval is calculated in the function call “buildInterval(Kset,finalValue)” in 3rd line of algorithm 3 listed above.

III.4 KNN regression with Gaussian distribution

In statistics, the Gaussian distribution \( N(\mu, \sigma^2) \) is modeled by two parameters, mean \( \mu \) and variance \( \sigma^2 \). After obtaining these parameters, we can obtain the \( \alpha \) quantile. Here \( \alpha \) is 0.95, thus the 95% quantile, is the interval where we can find 95% of the data when the distribution is Gaussian.

\[ P(\mu - t \sigma \leq X \leq \mu + t \sigma) = \Phi(t) - \Phi(-t) = \Phi(t) - (1 - \Phi(t)) = 2 \Phi(t) - 1 \]
where

\[ P(\mu - t \sigma \leq X \leq \mu + t \sigma) = \alpha, \text{and } \alpha \in ]0, 1[ , \]

which is equivalent to \[ 2 \Phi(t) - 1 = \alpha, \text{where } \Phi(t) = \frac{\alpha + 1}{2}, \]

if we take \( t = 2 \), \[ P(\mu - 2 \sigma \leq X \leq \mu + 2 \sigma) \approx 0.9544 \]

In case of assumption of a Gaussian distribution of the K-nearest neighbors around the input instance, how do we calculate the \( \mu \) and the \( \sigma^2 \)? Actually there are two way to calculate these parameters: globally and locally. We will see them in the next two paragraphs.

### III.4.1 Global Gaussian distribution

The average value of \( \mu \) and \( \sigma^2 \) for all the instances in the validation set are calculated. Then we use these two constants in regression to take the 95% interval. In the figure III.2 you can see that the regression interval is the same everywhere in the feature space.

### III.4.2 Local Gaussian distribution

Since it is not guaranteed that the nearest neighbors will be distributed identically over the feature space, the average value of the \( \mu \) may be too big somewhere and somewhere else it may be too small. To overcome this issue, we can calculate these parameters locally. As shown in algorithm 4, when doing a KNN regression, we use the found k-nearest neighbors in the regression part to
calculate the mean and the variance of the distribution. Once we’ve found these values, in 5\textsuperscript{th} line of the algorithm, we build the local regression interval. For more details about the formula see section III.4.

It is important to note that this interval is the 95% Gaussian quantile of one input instance’s nearest neighbors. It is a local regression interval and we have to calculate it for each input instance.

**Algorithm 4** Local regression interval in KNN

1: Kset ← Find the K-nearest neighbors of the input
2: finalValue ← KNNregression(Kset)
3: mean ← avg(Kset)
4: sigma ← calculate the standard deviation of Kset.
5: regressionInterval ← \((\text{mean} - 2 \times \text{sigma}, \text{mean} + 2 \times \text{sigma})\)

III.4.3 Proof of normality

We are talking about the 95% quantile without knowing even if our data distribution is Gaussian or not. We can do many types of normality tests to verify this assumption. One of them is the following: in the test data set, each instance contains the input parameters plus the real result. For all the instances in the test set, we make a KNN regression, then we look at the 95% quantile of the Kset\(^2\) and check if the real result is located inside it. If our data distribution were Gaussian, we will have near 95% of test set instances which have their real result located in the Kset’s 95% quantile. During the test phase we can also look at the average 95% quantile, and compare it with the required interval which contains 95% of the data distribution. In the experiments chapter we will have a closer sight on these tests.

There exists also statistical normality tests which are used to determine whether a data set is well-modeled by a normal distribution or not. In the experiments chapter, we’ve used the Shapiro-Wilk test [20] on two instances of our dataset to compare them with a normal distribution.

III.5 KNN regression with fuzzy set

In some cases, like in III.3, we can see that the normal assumption about the nearest neighbors distribution might not be the best one. Hence we will try to find the fuzzy set that can the best fit our data set. Here, we have chosen imprecise regression as our regression method, so we will try to find the fuzzy set that maximize the function described in III.5, this fuzzy set will be the regression interval of our input data.

\(^2\)The set containing the K nearest neighbors of an input instance.
III. Our approach

It is important to notice that, like all the mentioned regression interval calculation methods, we are executing the third line of the algorithm 3 to find our regression interval. Imprecise regression assigns a fuzzy set to each input instance. This fuzzy set is a trapezoidal shaped figure drawn over an interval.

As you can see in the figure III.4, the fuzzy set is a trapezoid with four vertices named a, b, c, and d. The (a, d) interval is called the support which should contain all the data (Kset) without the outliers and the (b, c) interval is called the core. We want to assign values included in Kset to a, b, c and d which maximize the evaluation function described before in . The optimization step includes, a maximization of the equation II.1 for each $x_i$. So for each instance, we must iterate over all the possible values of $y_i$, and maximize the equation II.1. Since this equation contains non derivable functions like $\text{Area}(A_{(\pi_i(y_i))})^3$, it is not possible to use the Newton or the BFGS method. Thus, to solve this optimization problem, we used two different approaches which didn’t make any use of the derivation in their maximization process.

III.5.1 First optimization method

In our first method we chose the initial values for a, b, c and d so as to begin with an interval that will certainly includes their optimum values. In this problem the current state is the values affected to a, b, c and d and the neighborhood is the other state which is the result of changing the value of one, to four vertices $^4\pm\alpha$, where $\alpha$ is a decreasing value initiated by $\frac{d-a}{15}\times\mu$.

If we take x as one of the four vertices value, in the next state it can change to x, x + $\alpha$ or x – $\alpha$. We have four vertices and each vertex can have 3 different values, so we have $3^4$ = 81 states. One of the calculated state is our current state, thus the neighborhood has 80 states.

Our neighborhood function in algorithm 5 finds all the 80 states, then we evaluate them and choose the state that maximize the evaluation function. These operations are done until the loop is terminated. The constant values for the loop termination are chosen practically. This method

\begin{itemize}
\item For a fixed instance ($x_i$), the function $\text{Area}(A_{(\pi_i(y_i))})$ is not continue over the $y_i$ axe.
\item The vertices are a, b, c, d.
\end{itemize}
III. Our approach

III.5.2 Second optimization method

Our second optimization method was the famous simulated annealing optimization method. Simulated annealing [14] is a meta-heuristic method developed for optimization problems. This method is inspired from a well-known physical phenomenon, coming from metallurgy. Let us consider a function $E: s \mapsto \mathbb{R}$ to be minimized, and representing the energy of a statistical mechanical system in a given state $s$. The probability for the system to go from the state $s$ to the state $s'$ at the temperature $T$ is given by the Boltzman-Gibbs distribution $P(s) = e^{-(E(s') - E(s))/kT}$ where $k$ is the Boltzmann constant. For high values of $T$, all states have a high probability to be accepted. On the opposite side, when $T$ is close to zero, only states improving the current minimization of the function will be accepted. The convergence to the global optima is granted when using a logarithmic decrease of the temperature. Given an initial temperature $T_0$, the temperature at step $t$ of the algorithm is $T_t = r \times T_{t-1}$ for $t \geq 1$ and $0 < r < 1$. With this method, we are not sure to find the global minimum, but we can expect to find at least a local one better than the one computed by a greedy algorithm. The values of temperatures $T_0$, $T_{\text{min}}$, and $r$ are critical and depend on the problem we want to solve. The initial state can be chosen randomly, since a “good” initial state will be forgotten” at a high temperature.

In this approach we have the same initialization part and the same neighborhood function, but we use the simulated annealing optimization method to find the global maximum. Simulated annealing tries to find the global minimum but we implemented it to find the global maximum.
Algorithm 5 First method of optimization

1: \( a \leftarrow \text{mean} - 12.5 \times \text{sigma} \)
2: \( b \leftarrow \text{mean} - 12.5 \times \text{sigma} \)
3: \( c \leftarrow \text{mean} + 12.5 \times \text{sigma} \)
4: \( d \leftarrow \text{mean} + 12.5 \times \text{sigma} \)
5: \( \alpha \leftarrow \frac{d-a}{15 \times \mu} \)
6: \( \text{iterationnum} \leftarrow 10000 \)
7: \( \text{repeat} \)
8: \( \text{neighbourhood} \leftarrow \text{all the 80 states resulted of changing the value of a and/or b and/or c and/or d by } \pm \alpha \)
9: \( \text{for all} \text{ nextState in neighbourhood do} \)
10: \( \text{if} \ \text{eval(nextState)} \geq \text{eval(currentState)} \text{ then} \)
11: \( \text{currentState} \leftarrow \text{nextState} \)
12: \( \text{end if} \)
13: \( \text{end for} \)
14: \( \alpha \leftarrow \text{decrease } (\alpha) \)
15: \( \text{until } i = 0 < \text{iterationnum and } \alpha > \frac{\text{mean}}{100000} \)
16: \( \text{bestState} \leftarrow \text{currentState} \)

In the 10th line of the algorithm 6 we use simulated annealing with cooling by plateau to find the local maximum. In the local maximum search, we use 800 plateau and in each plateau we iterate 1000 times. The temperature is cooled by a coefficient of 0.99 in each plateau. If after 35 consecutive plateaus, there is no better solution found or the \( \alpha \) used in neighbourhood function has reached 0.0001 of its initial value in the local maximum search, we quit the current local search. We do these local maximum searches “imax” times, and each time we begin by a different initial state to find a global maximum near to the real global maximum. Simulated annealing is slow but converges asymptotically to a global maximum. This is why we chose it as our optimization method.

Once the fuzzy set maximizing the imprecise evaluation function is found, we can use it as our ”regression set”. This technique is a different way to find an interval for the regression result. In the experiments chapter we will have numerical comparison of these different approaches of regression.
Algorithm 6 Second method of optimization

1: search ← true
2: i ← 0
3: lastBestFound ← 0
4:imax ← 20
5: stateSpace ← neighborhood function
6: currentState ← random state in stateSpace { initial state }
7: bestState ← currentState { initialize best state with initial state }
8: while search and i <imax do

9: \[ t\text{init} \leftarrow \frac{\text{deltachange}}{\text{movement} \times \log(\text{movement})} \] { takes a new initial temperature by doing some movements and using the sum of the change in evaluation function in the formula below }

10: currentState ← findLocalMaximum( tinit, currentState) { finds the local maximum using the simulated annealing method }

{ save the maximum if it is the best one }
11: if \( \text{eval(currentState)} > \text{eval(bestState)} \) then
12: bestState ← currentState
13: lastBestFound ← i
14: end if

{ if this doesn’t change for a while we break }
15: if \( i - \text{lastBestFound} > 8 \) then
16: search ← false
17: end if
18: i ← i +1
19: end while

20: stateSpace ← neighborhood function
21: currentState ← random state in stateSpace { find a new random state for the next round }
III. Our Approach

III.6 Summary in French

Ce chapitre a pour but de décrire le travail théorique et pratique que nous avons effectué pendant cette mémoire. Nous avions décidé d’appliquer une méthode de régression pour prédire la trajectoire de l’avion. On a vu aussi qu’il fallait avoir une approche d’incertitude ou probabiliste qui sache gérer la notion de l’imprécision caché dans notre problématique. Comme nous l’avions annoncé à la fin de la deuxième chapitre, nous allons développer notre sujet autour de la méthode des K plus proches voisins. En suite nous allons utiliser les ensembles flous pour décrire l’intervalle de distribution. K plus proche voisin et une méthode de classification qui a été adaptée pour la régression. Nous avons implémenté une version pondérée de la régression des K plus proche voisins. Le choix du K est un des plus importantes choix à faire dans ce problème. Dans la classification un K plus grand réduit le bruit sur les données mais réduit aussi la limite entre les classes, quand à la régression on doit être plus attentif par qu’on travaille avec des valeurs absolues. La fonction de distance utilisé dans cette méthode a beaucoup d’influence sur son efficacité. C’est avec cette fonction qu’elle trouve le voisinage de la valeur à prédire. Une fois le voisinage trouvé, elle prend comme résultat, la moyenne pondérée sur l’ensemble de la variable dépendante (la variable à prédire) du voisinage. On voie aussi que la densité de l’espace de l’apprentissage joue beaucoup sur la précision des résultats. La distance Euclidienne est la distance la plus utilisée. On a l’habitude de la normaliser car sans normalisation, les attributs qui ont des valeur plus grandes pèseront plus dans la distance alors que ça ne devrait pas être comme ça. La distance Mahalanobis projette les données dans une autre espace, non-corrélée, puis elle y calcule la distance des instances. Nous avons aussi proposé deux fonctions de distance. La première prend la surface entre les deux trajectoires comme la valeur de leur distance. L’idée est de poser des contraintes sur le voisinage est puis dans le voisinage appliquer une fonction la distance souhaitée, par exemple la distance Euclidienne. La relaxation viens du fait que dans certains cas, les données ne sont pas assez denses et le voisinage qui satisfasse toutes les contraintes est très petite voire même vide, donc dans les situations pareilles, on pense à relaxer les contraintes. Les contraintes, sont normalement définies par l’étude des données et elle portent normalement sur les variables qui sont les plus corrélées à la variable dépendante.

L’incertitude sur l’intervalle à prédire est explorée par deux façons. La méthode Gaussienne prend la quantile de la Gaussienne 95% , centrée à la moyenne et elle assume que la distribution dans l’intervalle de prédiction est Gaussienne. Dans cette méthode, nous avons le choix entre une hypothèse globale et une hypothèse locale. Dans l’hypothèse globale, on considère que les données sont reparties sur toute l’espace de manière presque identique, ce qui nous mène à choisir une intervale Gaussienne qui sera pareille pour toutes les instances à prédire. L’hypothèse locale vient du fait que nous faisons l’hypothèse que les données ne sont pas distribuées de façon identiques, dont on prendra une intervale Gaussienne qui sera propre à chaque instance à prédire. La deuxième approches est l’approche de la régression imprécise qui trouve une distribution de possibilité sur l’intervalle de la prédiction. La distribution de possibilité de la régression imprécise doit être calculée en maximisant le résultat d’une formule qui porte sur la distribution de la variable dépendante. Nous avons expliqué en détail, la raison pour laquelle cette fonction n’est pas continue, donc pas dérivable partout. Sachant ce fait, nous ne pouvons pas faire une optimisation
avec la méthodes de Newton ou la méthode de BFGS. Nous avons implémenté deux méthodes d’optimisation. La première étant une approche locale, est plus rapide mais ne garantit pas les maximums locaux. La deuxième est le recuit simulé. Il est plus lent mais converge asymptotiquement au maximums locaux. Dans le chapitre suivant nous allons voir les résultats des tests de normalité effectuées pour la preuve de la distribution normale des données. On verra aussi la comparaison des intervalles de prédiction et leur degrés de ressemblance avec l’hypothèse avec laquelle elles ont été calculées. Nous pourront ensuite mieux choisir la méthode la plus efficace pour la prédiction de trajectoire.
Chapter IV

Experiments

In this chapter we will see some numerical results of the tests done during this master thesis to demonstrate the objective mentioned in the introduction. This chapter begins by reviewing what we did on "Attribute selection", then we see the result of the normality-test, KNN algorithms applied with different parameters and we finish by taking a look on comparing KNN with other regression techniques. In these tests we implemented some algorithms in Java or we used already implemented algorithms of WEKA\(^1\). We separated the data into a training set named "TS" containing 34360 instances and a validation set of 7960 instances called VS. All the results \(^2\) demonstrated in this chapter are obtained with of our models trained on the training set TS and validated using the validation set VS. We didn’t use cross validation because we had validation sets containing enough instances to verify the efficiency of our models.

It is important to mention that, in the final days of this research, we found that it is much more efficient to separate the data sets by \(t_{\text{predict}}\), like \(TS_1\) and \(VS_1\) for \(t_{\text{predict}}\) equal of 1, and so on until 40. Next when employing the KNN regression for an instance with \(t_{\text{predict}}\) equal to \(i\), we will use the \(TS_i\) as our training set and \(VS_i\) as our validation set. We also found that it is more efficient to predict the attribute \(\delta zt (zt - z0)\), instead of the attribute \(zt\). We didn’t had the time to do again all tests, so we call these data sets, "best set", thus in the experiments, when we mention using "best set", we would refer to the regression type that we explained in this paragraph.

IV.1 Regression comparison

IV.1.1 Algorithm comparison

In the table IV.1, we can see different regression method tested with our dataset. These results are obtained with TS and VS. Our multilayer perceptron has three layers, contains 15 nodes in

---

\(^1\)Weka is an open source software which collects machine learning algorithms for data mining tasks. For more information see "http://www.cs.waikato.ac.nz/ml/weka/".

\(^2\)Mean absolute error, root mean squared error and etc.
IV. Experiments

the hidden layer with sigmoid functions and was trained 50000 times with TS using a learning rate of 0.15. The KNN regression result is the best result that we’ve found using KNN and it is explained in more detail in IV.2.1. The SVM\(^3\) that we used has a kernel using radial basis function with \(\nu = 0.5, \gamma = 0.0, \epsilon = 0.001\). As you can see neural networks is currently the best solution. We think that the SVM model might get better result and this is because of our misconfiguration of its model.

<table>
<thead>
<tr>
<th>Method</th>
<th>MAE</th>
<th>RMSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>MLP</td>
<td>560</td>
<td>760</td>
</tr>
<tr>
<td>Linear regression</td>
<td>830</td>
<td>1100</td>
</tr>
<tr>
<td>KNN regression</td>
<td>780</td>
<td>1030</td>
</tr>
<tr>
<td>KNN regression with best set</td>
<td>670</td>
<td>930</td>
</tr>
<tr>
<td>SVM</td>
<td>2750</td>
<td>3450</td>
</tr>
</tbody>
</table>

Table IV.1: Regression techniques.

IV.1.2 Attribute selection

The whole available attribute’s set is described in section I.2.2 and it will be called the ”standard” attribute set. For selecting a set of consistent variables we used two groups of attributes selection algorithm in WEKA. In attribute selection methods we use a search and an evaluation algorithm to find the subset of attributes which works best for prediction. The evaluator is used to value each subset of attributes and the search method determines what kind of search to perform.

Method description

Evaluation methods:

- CfsSubsetEval : this evaluator assigns a value to a subset of attributes by taking into account the individual predictive ability of each variable along with the degree of redundancy between them [9].

- Wrapper Subset Evaluator : Wrapper methods evaluate possible attribute sets with a selected learning scheme. This evaluator is used to find the best set suitable for a pre-selected learning scheme. The accuracy of the learning scheme for a set of attributes is determined by the result of cross validation with the learning scheme [15].

Search methods:

---

\(^3\)Using WEKA with this classifier : ”weka.classifiers.functions.LibSVM -S 4 -K 2 -D 3 -G 0.5 -R 0.0 -N 0.5 -M 40.0 -C 1.0 -E 0.001 -P 0.1”
- **BestFirst**: It performs greedy hill-climbing augmented with a backtracking facility to search the space of attribute subsets.

- **Greedy Stepwise**: Searches through the space of attribute subsets by performing a greedy forward or backward search.

- **Genetic search**: It uses the genetic algorithm described in [2].

- **Exhaustive Search**: It starts from the empty set of attributes and do an exhaustive search through the space of attribute subsets.

### Methods result

#### Subset selection methods:

- **1** - Attribute subset evaluator: CfsSubsetEval  
  Search method: BestFirst in backward direction, stale search after 5 node expansions  
  Resulted variables: t\_predit, deltaisa, deltaCas, mach, deltaMach, Dx9, Dx8, Dx7

- **2** - Attribute subset evaluator: CfsSubsetEval  
  Search method: Greedy Stepwise  
  Resulted variables: t\_predit, esf, deltaisa, deltaCas, mach, deltaMach, xt

- **3** - Attribute subset evaluator: CfsSubsetEval  
  Search method: Genetic search  
  Resulted variables: t\_predit, deltaisa, deltaCas, mach, Dx9, Dx8, Dx3, Dx2, Dx1

- **4** - Attribute subset evaluator: CfsSubsetEval  
  Search method: Exhaustive Search  
  Resulted variables: t\_predit, deltaisa, deltaCas, mach, deltaMach, Dx9, Dx8, Dx7

#### Wrapper methods:

- **5** - Attribute subset evaluator: Wrapper Subset Evaluator  
  Learning scheme: weka.classifiers.functions.LinearRegression  
  Search method: Best first Search in backward direction, stale search after 5 node expansions  
  Resulted variables: t\_predit, deltaisa, cas, deltaCas, mach, deltaMach, Dz9, Dz8, Dz6, Dz4, Dz3, Dz2, Dz1, z0, Dx9, Dx8, Dx7, Dx6, Dx5, Dx3, Dx1, x0
IV. EXPERIMENTS

- Attribute subset evaluator: Wrapper Subset Evaluator
  Learning scheme: weka.classifiers.functions.MultilayerPerceptron
  Scheme options: -L 0.25 -M 0.2 -N 1000 -V 0 -S 1 -E 20 -H a,
  Search method: Best first Search in backward direction, stale search after 5 node expansions
  Resulted variables: tpredit, vr, esf, cas, deltaCas, mach, deltaMach, Dz9, Dz8, Dz7, Dz6, Dz5, Dz4, Dz3, Dz2, Dz1, z0, Dx9, Dx8, Dx7, Dx6, Dx5, Dx3, Dx1, x0, xt

Attribute selection conclusion

In the table IV.2 we will compare the standard set and the resulting sets from the experiments in IV.1.2. We used Multilayer perceptron (MLP) and linear regression methods to test the different attribute sets. We will use the symbol "ref: x" to make reference to the xth experiment, for example "ref: 6" means that we use the attributes set which resulted from running the 6th experiment.

In the table IV.2 the first and obvious result is that MLP is more efficient than linear regression.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>MAE</th>
<th>RMSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>MLP, trained 50000 times, Standard set</td>
<td>560</td>
<td>760</td>
</tr>
<tr>
<td>MLP, trained 50000 times, ref: 6</td>
<td>590</td>
<td>800</td>
</tr>
<tr>
<td>MLP, trained 50000 times, ref: 6-xt</td>
<td>1000</td>
<td>1540</td>
</tr>
<tr>
<td>Linear regression, ref: 5</td>
<td>720</td>
<td>960</td>
</tr>
<tr>
<td>Linear regression, Standard set</td>
<td>830</td>
<td>1100</td>
</tr>
</tbody>
</table>

Table IV.2: Variable selection comparison.

We’ve tested MLP and linear regression with the attribute set found by their own wrapper subset method. We can see that by reducing the attributes subset in linear regression to the result of its wrapper subset method, we obtain better result, however this is not the case for MLP. We’ve also tested MLP without the variable ”xt”, we can observe that this variable plays an important role in the prediction efficiency. After considering this table we decided to keep the standard set as our attribute set.

IV.2 KNN algorithm

In this section we will show the numerical results of different KNN implementations on the standard attributes set. You can assume that all the tests are done using TS4 and VS5 otherwise we will mention the training set and the validation set. We begin by choosing the K, then we

---

4Our training set with the standard attribute set containing 34360 instances.
5Our validation set with the standard attribute set containing 7960 instances.
describe what distance function we will choose and after demonstrating that we are working with non Gaussian data, we will see why in our case, it is better to use imprecise regression.

**IV.2.1 Chosing the value of $K$**

In the table IV.3 we show the experiments of KNN with normalized Euclidean distance with different $K$s. We’ve used normalized Euclidean distance because it is the most used distance function.

<table>
<thead>
<tr>
<th>$K$</th>
<th>MAE</th>
<th>RMSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>1000</td>
<td>1315</td>
</tr>
<tr>
<td>10</td>
<td>965</td>
<td>1280</td>
</tr>
<tr>
<td>10 (using the best set)</td>
<td>695</td>
<td>920</td>
</tr>
<tr>
<td>15</td>
<td>960</td>
<td>1265</td>
</tr>
<tr>
<td>20</td>
<td>955</td>
<td>1255</td>
</tr>
<tr>
<td>25</td>
<td>945</td>
<td>1245</td>
</tr>
<tr>
<td>30</td>
<td>940</td>
<td>1240</td>
</tr>
<tr>
<td>30 (using the best set)</td>
<td>675</td>
<td>915</td>
</tr>
</tbody>
</table>

Table IV.3: KNN with Normalized Euclidean distance with different K values.

As we can see, the result of $K$ equal to 10 is almost the same as where the $K$ value is 20 or 30. Thus in our distance tests we’ve used a $K$ equal to 10 to have a faster KNN algorithm and when comparing the distributions we’ve employed a $K$ equal to 20 and 30 to have a more populated distribution.

**IV.2.2 Choosing the distance functions**

**Different Constraint distance functions**

In this part we will see different distance functions. The distance used is Normalized Euclidean and the $K$ used in the KNN is 10. The constraint attributes used in tables IV.5 and IV.4 are obtained through observing the relationships between attributes in the training set. These mentioned tables are just a brief review of our experiments to justify our constraint selection. In the table IV.4 we can see the results of constrained distance. This distance gives better MAE and RMSE than the simple normalized Euclidean distance function. However, this distance function doesn’t work for all instances. Since our training set is not enough dense, KNN combined with this distance function, fails to find the K nearest neighbours for all the input instances. Thus we need to relax some constraints.

In the table IV.5 we can see constrained distance with relaxation\(^6\) function experimented with different parameters. In some cases, the previous distance function had better results, but

---

\(^6\)See the algorithm 1.
<table>
<thead>
<tr>
<th>Number</th>
<th>Constraints</th>
<th>MAE</th>
<th>RMSE</th>
<th>K=0 instances</th>
<th>K=0 percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>None</td>
<td>965</td>
<td>1280</td>
<td>0</td>
<td>0%</td>
</tr>
<tr>
<td>2</td>
<td>z0=500, x0=5</td>
<td>785</td>
<td>1085</td>
<td>320</td>
<td>4.2%</td>
</tr>
<tr>
<td>3</td>
<td>z0=500, dx9=0.015</td>
<td>775</td>
<td>1040</td>
<td>1480</td>
<td>18.6%</td>
</tr>
<tr>
<td>4</td>
<td>z0=500, x0=5, dx9=0.015</td>
<td>775</td>
<td>1070</td>
<td>2840</td>
<td>35.6%</td>
</tr>
<tr>
<td>5</td>
<td>z0=1000</td>
<td>816</td>
<td>1105</td>
<td>1080</td>
<td>13.5%</td>
</tr>
<tr>
<td>6</td>
<td>z0=1000, dx9=0.015, dx1=0.3</td>
<td>785</td>
<td>1075</td>
<td>1080</td>
<td>13.5%</td>
</tr>
</tbody>
</table>

Table IV.4: Constrained distance example.

This gain is achieved by neglecting some instances. Since the neglected instances are located in sparser locations in the training set subspace, the result of their KNN regression may be worse than other instances. Finally we’ve ended up by selecting the constraint ”z0=500, x0=5” because it improves our MAE on the best set from 695 to 625 which means 10.0% of improvement in MAE, it improves also the RMSE from 925 to 975 which means 5.4% of improvement in RMSE, and it is quite simple. The reason behind the success of these constraints, can also be justified by other way. The combination of ”z0=500” and ”x0=5” regroup a set of instances which belongs to similar aircraft locations in different trajectories. They describe aircraft that are near to each other in altitude and had travelled the same distance from the airport (the turns done by the aircraft are taken into account through the attribute “x” which is the travelled distance from the airport at the time ”t0”). In the table IV.6 we compare different distance functions. These tests were made with a random set of 160 instances from our standard validation set. We can see that the Mahalanobis distance function is not suitable to use in conjunction with the constrained method. The Mahalanobis distance function projects the input space into a non-correlated space where our constrained method looses its utility. This lost of utility is because constrained distance method, tries to exploit the correlation between some independent attributes with the response variable and when we apply it to a non-correlated set, it gives worse results.

IV.2.3 Analysis of the normality distribution

Experience objectives

As said in III.4.3 we will compare the mean of 95% quantile of different K, with the mean of Gaussian quantile needed to have 95% of the data located in it. We call this later interval, the ”alpha quantile”. We will have the same comparison for the 50% quantile and similarly, the mean of Gaussian quantile needed to to have 50% of the data located in it will be called the ”beta quantile”. We will also see what percent of the real data is located in the Gaussian 95% and 50% quantiles. We find it useful to underline that the data distribution for which we are calculating these Gaussian notions is the Kset of KNN regressions done with TS and VS, for more information see III.4.

\(^7\)Kset is the set of K nearest neighbours found for an input instance in a KNN regression.
Table IV.5: Constrained distance with relaxation, using different parameters.

<table>
<thead>
<tr>
<th>Constraints</th>
<th>Minimum MAE</th>
<th>MAE</th>
<th>Maximum MAE</th>
<th>RMSE</th>
<th>K = 0 percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>0</td>
<td>40</td>
<td>965</td>
<td>1110</td>
<td>1280</td>
</tr>
<tr>
<td>best set,None</td>
<td>0</td>
<td>40</td>
<td>965</td>
<td>1110</td>
<td>1280</td>
</tr>
<tr>
<td>z0=500</td>
<td>0</td>
<td>805</td>
<td>4700</td>
<td>1055</td>
<td>0</td>
</tr>
<tr>
<td>z0=1000</td>
<td>0</td>
<td>855</td>
<td>5080</td>
<td>1115</td>
<td>0</td>
</tr>
<tr>
<td>z0=1500</td>
<td>0</td>
<td>880</td>
<td>4450</td>
<td>1155</td>
<td>0</td>
</tr>
<tr>
<td>z0=500, dx9=0.015</td>
<td>0</td>
<td>795</td>
<td>5700</td>
<td>1080</td>
<td>0</td>
</tr>
<tr>
<td>z0=500, dx1=0.3</td>
<td>0</td>
<td>805</td>
<td>4700</td>
<td>1055</td>
<td>0</td>
</tr>
<tr>
<td>best set, z0=500, x0=5</td>
<td>46</td>
<td>625</td>
<td>1095</td>
<td>875</td>
<td>0</td>
</tr>
<tr>
<td>z0=500, x0=5</td>
<td>0</td>
<td>780</td>
<td>4900</td>
<td>1030</td>
<td>0</td>
</tr>
<tr>
<td>z0=500, xt=20</td>
<td>0</td>
<td>805</td>
<td>4700</td>
<td>1050</td>
<td>0</td>
</tr>
<tr>
<td>z0=500, x0=5, dx9=0.015</td>
<td>0</td>
<td>930</td>
<td>5700</td>
<td>1250</td>
<td>0</td>
</tr>
<tr>
<td>z0=500, x0=5, dx1=0.3</td>
<td>0</td>
<td>800</td>
<td>4900</td>
<td>1055</td>
<td>0</td>
</tr>
<tr>
<td>z0=500, x0=5, xt=20</td>
<td>0</td>
<td>800</td>
<td>4900</td>
<td>1055</td>
<td>0</td>
</tr>
<tr>
<td>z0=500, x0=5, dx9=0.015, dx1=0.3</td>
<td>0</td>
<td>930</td>
<td>5700</td>
<td>1250</td>
<td>0</td>
</tr>
</tbody>
</table>

Experience results

The tests done here are performed using KNN regression\(^8\) with normalized\(^9\) Euclidean distance. As you can see in the tables IV.7 and IV.8, the Kset distribution is not Gaussian. In table IV.7, with a K equal to 30, the alpha quantile is nearly 1.4 times the 95% Gaussian quantile. In the same row you can see that we must take a 99.3% Gaussian quantile to have 95% of the Kset data distribution. We’ve shown by the two figures IV.2 and IV.1, that the Kset data distribution is not Gaussian, these figures compare the density of a Kset of 30 represented in red, with a Gaussian density having the same mean and variance represented by green. It is important to note that the Kset of these two figure didn’t pass the Shapiro-Wilk normality test\(^10\).

IV.2.4 Fuzzy set prediction

In section III.3 we saw three different methods of finding a regression interval: Global Gaussian, Local Gaussian and fuzzy set which in our case is calculated using imprecise regression method.

\(^8\)We always means Weighted KNN regression when we talk about KNN regression.
\(^9\)Normalization is done based on the data range of the training set.
\(^10\)Shapiro-Wilk normality test using an error risk of 5%.
IV. EXPERIMENTS

Figure IV.1: Kset density VS Gaussian density.

Density distribution of a Kset of 30 (in red), compared to a Gaussian distribution having the same mean and variance.

Figure IV.2: Kset density VS Gaussian density, another Example.

Density distribution of a Kset of 30 (in red), compared to a Gaussian distribution having the same mean and variance.
<table>
<thead>
<tr>
<th>Distance</th>
<th>constraints</th>
<th>MAE</th>
<th>RMSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normalized Euclidean</td>
<td>None</td>
<td>965</td>
<td>1280</td>
</tr>
<tr>
<td>Normalized Euclidean with best set</td>
<td>None</td>
<td>695</td>
<td>925</td>
</tr>
<tr>
<td>Constrained Normalized Euclidean with relaxation</td>
<td>$z_0=500, x_0=5$</td>
<td>780</td>
<td>1030</td>
</tr>
<tr>
<td>Constrained Normalized Euclidean with relaxation using best set</td>
<td>$z_0=500, x_0=5$</td>
<td>625</td>
<td>875</td>
</tr>
<tr>
<td>Mahalanobis</td>
<td>None</td>
<td>9202</td>
<td>18975</td>
</tr>
<tr>
<td>Constrained Mahalanobis with relaxation</td>
<td>$z_0=500, x_0=5$</td>
<td>10150</td>
<td>24310</td>
</tr>
<tr>
<td>ABT</td>
<td>None</td>
<td>5765</td>
<td>6770</td>
</tr>
<tr>
<td>ABT without $z_0$</td>
<td>None</td>
<td>5990</td>
<td>7000</td>
</tr>
<tr>
<td>Constrained ABT with relaxation</td>
<td>$z_0=500, x_0=5$</td>
<td>6080</td>
<td>7090</td>
</tr>
<tr>
<td>Constrained ”ABT without $z_0” with relaxation”</td>
<td>$z_0=500, x_0=5$</td>
<td>6350</td>
<td>6370</td>
</tr>
<tr>
<td>ABT V.2</td>
<td>None</td>
<td>920</td>
<td>1110</td>
</tr>
<tr>
<td>Constrained ABT V.2 with relaxation</td>
<td>$z_0=500, x_0=5$</td>
<td>844</td>
<td>955</td>
</tr>
</tbody>
</table>

Table IV.6: Different distance comparison.

<table>
<thead>
<tr>
<th>$K$</th>
<th>Percentage of data located in Gaussian 95%</th>
<th>Gaussian 95% quantile size</th>
<th>Gaussian quantile needed to obtain the alpha quantile</th>
<th>Alpha quantile size</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>54.2%</td>
<td>1730</td>
<td>99.6%</td>
<td>6040</td>
</tr>
<tr>
<td>10</td>
<td>73.2%</td>
<td>2870</td>
<td>99.1%</td>
<td>6175</td>
</tr>
<tr>
<td>15</td>
<td>78.5%</td>
<td>3510</td>
<td>99.9%</td>
<td>6215</td>
</tr>
<tr>
<td>20</td>
<td>82.6%</td>
<td>3940</td>
<td>99.7%</td>
<td>6230</td>
</tr>
<tr>
<td>25</td>
<td>85.0%</td>
<td>4275</td>
<td>99.5%</td>
<td>6210</td>
</tr>
<tr>
<td>30</td>
<td>86.9%</td>
<td>4570</td>
<td>99.3%</td>
<td>6330</td>
</tr>
</tbody>
</table>

Table IV.7: The 95% intervals.

In these tests, we’ve used the best with a $K = 30$. The employed distance functions are the Constrained normalized Euclidean distance with relaxation ("$z_0=500, x_0=5$" as constraint) and the normalized Euclidean distance. As we saw earlier, our data set is not Gaussian, so employing the globally Gaussian method which consists of using an interval of approximately $4 \times \sigma$ would not be reasonable. Thus we will compare the local Gaussian method with the imprecise regression method.

**Average results**

In the tables IV.9 and IV.10, we’ve put the average results for the following interval types: Local Gaussian 95% quantile, the A-D interval of the found possibility distribution by imprecise regression and the alpha to reach A-D. The alpha to reach A-D, is the Gaussian quantile needed
IV. EXPERIMENTS

<table>
<thead>
<tr>
<th>K</th>
<th>Percentage of data located in Gaussian 50%</th>
<th>Gaussian quantile size</th>
<th>Gaussian quantile needed to obtain the beta quantile</th>
<th>Beta quantile size</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>23.4%</td>
<td>595</td>
<td>92.1%</td>
<td>1530</td>
</tr>
<tr>
<td>10</td>
<td>36.7%</td>
<td>990</td>
<td>70.2%</td>
<td>1455</td>
</tr>
<tr>
<td>15</td>
<td>43.5%</td>
<td>1210</td>
<td>59.4%</td>
<td>1450</td>
</tr>
<tr>
<td>20</td>
<td>47.0%</td>
<td>1355</td>
<td>53.4%</td>
<td>1450</td>
</tr>
<tr>
<td>25</td>
<td>51.0%</td>
<td>1470</td>
<td>49.0%</td>
<td>1440</td>
</tr>
<tr>
<td>30</td>
<td>53.9%</td>
<td>1570</td>
<td>44.7%</td>
<td>1405</td>
</tr>
</tbody>
</table>

Table IV.8: The 50% intervals.

to obtain the same percentage of $zt$ \(^{11}\) inclusion in Kset as the A-D interval. For example if an A-D interval had a $zt$ inclusion in Kset equal to 80%, The alpha to reach A-D is the Gaussian quantile needed to obtain 80% of $zt$ inclusion in Kset.

<table>
<thead>
<tr>
<th>Interval type</th>
<th>Interval size</th>
<th>$zt$ inclusion in Kset</th>
<th>Kset coverage by the interval</th>
</tr>
</thead>
<tbody>
<tr>
<td>Local Gaussian 95% quantile</td>
<td>2600</td>
<td>83.1%</td>
<td>94.3%</td>
</tr>
<tr>
<td>Alpha to reach A-D</td>
<td>2190</td>
<td>76.5%</td>
<td>76.7%</td>
</tr>
<tr>
<td>A-D interval</td>
<td>2635</td>
<td>76.5%</td>
<td>97.7%</td>
</tr>
</tbody>
</table>

Table IV.9: Regression intervals for KNN with Constrained normalized Euclidean with relaxation with best set using a K=30.

<table>
<thead>
<tr>
<th>Interval type</th>
<th>Interval size</th>
<th>$zt$ inclusion in Kset</th>
<th>Kset coverage by the interval</th>
</tr>
</thead>
<tbody>
<tr>
<td>Local Gaussian 95% quantile</td>
<td>2205</td>
<td>71.1%</td>
<td>95.5%</td>
</tr>
<tr>
<td>Alpha to reach A-D</td>
<td>2090</td>
<td>71.9%</td>
<td>71%</td>
</tr>
<tr>
<td>A-D interval</td>
<td>2240</td>
<td>71.9%</td>
<td>98.3%</td>
</tr>
</tbody>
</table>

Table IV.10: Regression intervals for KNN with normalized Euclidean with best set using a K=30.

**Fuzzy set for data representation**

In the tables IV.9 and IV.10, we saw that the Local Gaussian 95% quantile is a nearly equal to the A-D interval but it contains less percent of the Kset. As you can also see in the figure IV.3 the A-D interval always represents better the Kset’s data distribution than the Local Gaussian 95% quantile. The alpha to reach A-D, has the same $zt$ inclusion in K-set than the A-D interval, however as you can see in the figure IV.3, the possibility distribution is much more efficient in the data distribution representation, than the ”alpha to reach A-D”’s interval.

\(^{11}\)The right value of the attribute to be predicted.
In the figures IV.4 and IV.5, we have shown the difference between a Gaussian 95% quantile and a possibility distribution. These figures compare the Gaussian 95% quantile drawn with a green line to the blue trapezoid which is the possibility distribution. The black point is the predicted value. The figures are drawn using two samples of the tests done in the table IV.9. You can see that the fuzzy set represents better the data distribution than the Gaussian distribution, and the predicted point is in core the part of the fuzzy set where the possibility is one. In IV.5, the data is uniformly distributed and you can see that in the same way, the possibility distribution has its core equal to its support. It means that every instance in the fuzzy set has a possibility of one to occur, and when we look at our histogram we can see that this is consistent.
IV. EXPERIMENTS

Figure IV.4: Possibility distribution VS Gaussian distribution.

Figure IV.5: Possibility distribution VS Gaussian distribution, another Example.
Figure IV.6: Comparing the predictive power, Fuzzy set VS Gaussian distribution.

**Fuzzy set for prediction interval**

In the figure IV.6 the percent of the \( z_t \) inclusion in K-set for different interval have been shown. When using the normalized Euclidean distance, the A-D interval, has a better average and appears more efficient for data the regression interval. However when using the constrained distance, the local Gaussian 95% interval improves significantly in the prediction power. When we look at the figure IV.7, shows the interval sizes, we can understand that one major reason of this improvement is due to the fact that the interval sizes are larger with the constrained distance than the other one. The other point to stress is that the distribution of the \( z_t \) over the prediction intervals, changes by selecting another distance function. For a normalized Euclidean distance the A-D interval has a better average prediction result, but for the constrained distance the predictive power of the Gaussian distribution becomes more dominant.

**IV.2.5 Choosing the interval type**

We saw earlier that for the distribution’s representation task, the possibility distribution appears significantly more efficient than the local Gaussian distribution method. However here, our goal is to find the interval that with an smaller size, has a better prediction result. Thus in this moment and based on the above results, we first suggest to use the Constrained normalized Euclidean distance with relaxation combined with the local Gaussian ’s interval. Otherwise we suggest to use the same distance function combined with the alpha for A-D interval which has the same predictive efficiency that the A-D interval, but it is in average 15% smaller than the other ones.
Figure IV.7: Interval sizes.
IV.2.6 Conclusion

We saw earlier that for the distribution’s representation task, the possibility distribution appears significantly more efficient than the local Gaussian distribution method. The test have also shown that our distance functions are not really suitable for this problem. Thus we suggest to search for distance functions that minimize the distance between the prediction interval and the right value to be predicted. The KNN regression is not also the most efficient solution, because our data set in not enough dense to provide us the sufficient neighbourhood around each input instance. So we it would be better to use other regression methods or to improve our data set’s density.
IV.3 Summary in French

Dans le chapitres "Context " nous avons vue la définition de la trajectoire d’avion et la structure des données qu’on a utilisée pour notre problématique. Dans le chapitre " Background " on a fait l’état de l’art sur la régression et les méthodes de gestion de l’incertitude pour les régressions non-précises. Le chapitre précédant on a expliqué avec plus de détails ceux que nous avons décidé de développer et les raisons pour lesquelles nous les avons choisis. Dans ce chapitre nous avons publié les résultats des tests sur nos données. La plupart de ces tests ont été fait en utilisant l’ensemble nommé " TS " (une ensemble des données contenant 34360 instances ) comme l’ensemble d’apprentissage et ils ont été valide par l’ensemble "VS" (une ensemble des données contenant 7960 instances ). Parfois on a utilisé des sous-ensembles de chaque ensemble d’apprentissage et de validation , pour l’apprentissage et la validation respectivement. Au derniers jours de ce mémoire on a constaté qu’il fallait diviser les ensembles par le pas de temps à prédir et qu’il fallait prédir la différence entre l’altitude courante et l’attitude à l’instant à prédir et non la valeur absolue de l’altitude à prédir. A cause du manque de temps nous avons pas pu refaire tous les tests et nous avons juste ajouté les résultats des tests effectués avec la façon qu’on vient de décrire. Nous n’avons pas utiliser la validation croisée car nous disposions de suffisamment de donne pour valider les modèles.

On a commence par une comparaison des méthodes de prédiction avec la régression avec les K plus proches voisins combiné avec la fonction de distance Euclidienne avec contraintes et relaxation et on a constaté que les réseaux de neurone avait les meilleur résultats de prédiction. Comme les résultats des SVMs avaient apparus vraiment inefficace, nous pensons qu’ils ont été mal configurés. Ensuite dans la sélection des attributs nous avons décidé de garder notre ensemble d’attribut car les résultats montraient qu’ils comportaient tous des informations sur les trajectoires dont on ne pouvaient pas toujours retrouver dans les autres attributs. Pour les K plus proches voisins, le choix du K a été le suivant: puisque l’efficacité de la régression des K plus proches voisins avec la distance Euclidienne normalisée, est presque pareille à partir de 10 à 30, donc nous avons fait nos tests avec une valeur de K égale à 10 et pour les tests de distributions nous avons choisi un K de 30, car elle peut mieux décrire une distribution. Parmi les différentes fonctions de distance, la distance Euclidienne contrainte avec relaxation s’est montrée la plus efficace. En étudiant la normalité des données, on a constaté que l’intervalle de prédiction n’avais pas toujours une distribution Gaussienne et qu’elles entaient mieux représentées par les ensembles floue trouvées par la régression Imprécise. En regardant les pourcentage des valeurs prédites dans l’intervalle de la régression, on a observe qu’en utilisant la distance Euclidienne normalisée l’intervalle floue a presque la même capacité de prédiction, sachant que sa taille est très proche à taille de l’intervalle de la quantile Gaussienne locale 95%, alors qu’elle représente mieux la distribution des données. Quand à la distance Euclidienne contrainte avec relaxation, la situation change, cette distance distribue les instances d’une façon que l’erreur entre l’intervalle et la valeur prédite est plus Gaussienne qu’avant, donc l’intervalle Gaussienne a plus de capacité de prédiction, mais elle est toujours moins apte à représenter l’intervalle des données. Sachant que c’est la capacité à prédir qui est la plus prioritaire pour notre problématique, donc avec les résultats courants nous proposons la méthode de l’intervalle de la quantile de la Gaussienne locale 95% avec la fonction de distance avec contraintes. Nous avons vue que notre fonction
de distance n’était pas très efficace et on propose de faire plus de recherche sur les fonctions de distances. Nous avons vu aussi que la méthode des K plus proches voisins a des résultats moins performants que les autres méthodes de régression donc nous suggérons de changer de méthode tout en restant sur la régression Imprécise.
Conclusion

In this report, we saw what is the trajectory prediction problem and why we have to use regression algorithms. In the context chapter we demonstrated that we dispose of imprecise data and even with the most efficient regression algorithms we are unable to get precise and accurate results. As we accepted that we have to take account of the uncertainty on our prediction results, we decided to have the best trade off between uncertainty and accuracy. The later goal depends mainly on our data set and our regression method. Imprecise regression was the method that we have selected to reach the best trade-off between imprecision and accuracy in our results. However this method is well suited for non-normal data distributions so we demonstrated in our experiments chapter that our data distribution is not Gaussian. For the choice of the regression method we choose KNN regression, thus we introduced the KNN weighted imprecise regression. It is obvious that the KNN efficiency is significantly influenced by its distance function, so we’ve experimented the most common distance functions. We have introduced also the constrained distance function with relaxation which had the most efficient results in our method.

While validating our results with our model, the experiments showed that when using KNN regression, the data distribution of the neighborhoods found around the instances to predict are not Gaussian. Then we saw that possibility distribution resulting from the imprecise approach represents better the data distribution of the neighborhood, than the famous Gaussian 95% quantile. Experiments demonstrated, that the possibility distribution’s supports had almost equal values to Gaussian 95% quantiles, but they cover more the neighborhood distribution than the normal 95% quantile. This later statement, confirms that possibility distribution represents better the neighborhood distribution than the Gaussian 95% quantile. When comparing the percentage of the instance’s real values inside the prediction intervals (the prediction power of the distributions), we saw that when using the normalized Euclidean distance function, the Gaussian 95% quantile had almost the same prediction result, but when using our introduced distance function, its prediction efficiency became better than the Fuzzy set. This is all about our distance function, because the distance of the neighborhood interval from the response value, the neighborhood error (found by our introduced distance function), has a data distribution more similar to the Gaussian one than our possibility distribution. Our prediction result is also influenced by our regression technique. Since our data set in not enough dense to provide us the sufficient neighborhood around each input instance, It would be better to use other regression methods or to improve our data set’s density.

The next step could be to search a better distance function which will minimize the distances.
between the right values to predict and the found prediction intervals. It could be found by exploring different distance functions and also by changing our \( K \) values to see the other possible distributions. Finally we also saw that, KNN weighted regression, might not be the best option for the trajectory prediction problem, thus we could explore the imprecise regression method with more efficient regression techniques.
Bibliography


distances: Image retrieval and class representation. *IEEE Transactions on Pattern Analysis
and Machine Intelligence (PAMI)*, 2000.


[16] Shyjan Mahamud and Martial Hebert. Minimum risk distance measure for object recogni-

CONTROL, 2009.


[19] Frank Rosenblatt. The perceptron: A probabilistic model for information storage and orga-
nization in the brain. *Cornell Aeronautical Laboratory, Psychological Review v65, No. 6,


