Georgia Tech College of Computing

Supervised Learning for Motion Prediction

CS8803: AI for Robotics
Final Project Report

Team Name	TEAM 4 – “Team Atlanta”		
Semester	Spring 2016		
Team Members	Eran Medan Mark Trinquero Thomas Wyrick		
Date	April 2016	Version	1.0

Table of Contents

Overview of Problem Space

The goal of this project is to apply the knowledge and techniques gained throughout the semester to a real-world data problem and help students further explore the practical side of robotics. A combination of machine learning and artificial intelligence algorithms will be explored as part of *Team Atlanta's* approach.

In the real world, predictions must be made based on unstructured data with unknown or missing information sets. Oftentimes senor data is noisy, thus adding further complexity to the problem. For this project – the team was tasked with predicting future paths of motion for a NEXBUG Nano bot based on 60 seconds (1800 fames) of historical 'training' data.

The NEXBUG Nano bot uses the physics of vibration to propel forward and explore its environment. Due to the nature of the bot's design, forward motion is sporadic and difficult to predict. When encountering an obstacle the bot will move to a new path due to its persistent random behavior.

Team Background

Team Members:

Eran Medan **Thomas Wyrick** **Mark Trinquero**



"Team Atlanta" (Team 4) is comprised of all Atlanta based OMSCS students, which allowed us the opportunity to meet both virtually and in-person during the course of the project. All three team members have full-time work experience within the technology industry, and come from a variety of sectors and backgrounds.

This mix helped bring a collection of skill sets to the table and offer a unique perspective to the team's overall approach and project solution.

In order to collaborate throughout the project stages, the team utilized google hangouts to touch base virtually. This allowed the group an opportunity during the workweek to discuss general updates, potential ideas or approaches, and deep dives into AI specific algorithms.

Problem Statement

Each project team will be provided 10 videos, each 62 seconds long recorded at 30 frames per second. The goal of the project is to predict the robot's positions during the final 2 seconds based on data points from the first 60 seconds of each video. The team seeks to use an approach that provides the greatest accuracy in predicting future positions based on knowledge and techniques we have learned throughout the course.

Overview of Initial Ideas & Project Approaches:

The team kicked off the project by each making an initial pass, and then re-grouping to discuss and brainstorm various approaches the team could take. These discussions were very beneficial in helping identify team skill sets and break up project work accordingly.

A snapshot of ideas discussed during the beginning project stages has been summarized below:

- Branches of Machine Learning
    - Supervised learning -
    - Unsupervised learning
    - Re-enforcement learning
- Artificial Intelligence Approaches from CS6601
    - Localization (Partial and Kalman Filters)
    - Search, Planning, Control, and Smoothing
    - Robot Motion (Continuous and Controlled)
- GT OMSCS Course Concepts and Prior Knowledge/Techniques
    - KNN (regression)
    - K-Means (clustering)
    - Decision Tree (regression)
    - Random Forests, Extra Trees, Multiple Trees
    - Data Enrichment (adding velocity, trajectory, etc. to our provided data set based on running set of 'k' points)
    - Adding decay rate into accuracy predictions due to inherent noise

Note: detailed analysis into a sub-set of approaches that the team evaluated as part of our final approach can be found in the body this document. The team utilized Python's scikit-learn library heavily throughout analysis and in the development of our final solution.

Enrichment of Provided Data:

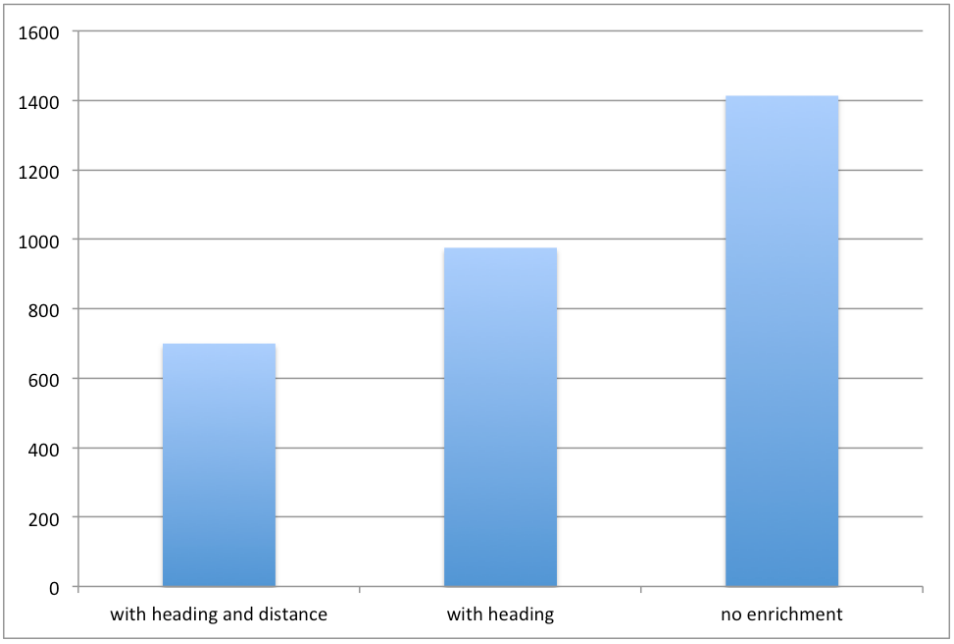
Using the provided training and testing data sets – the team decided to explore 'enhancing' or 'enriching ' the data by adding additional extrapolated attributes. The enriched data sets produced mixed results depending on the algorithm and approach in question. The team's takeaway was sometimes it is a trade-off between over-fitting vs. providing additional information to a model. The team also explored isolating particular attributes to see how and what affect various combinations of data enrichment affected prediction accuracy.

Some of the data enrichment approaches the team evaluated have been listed below:

- Trajectory/Heading – running average of the set of last 'n' data points
  - N = 3 data points
  - N = 7 data points
- Total Distance – cumulative distance of last 'n' data points
  - N = 3 data points
  - N = 7 data points

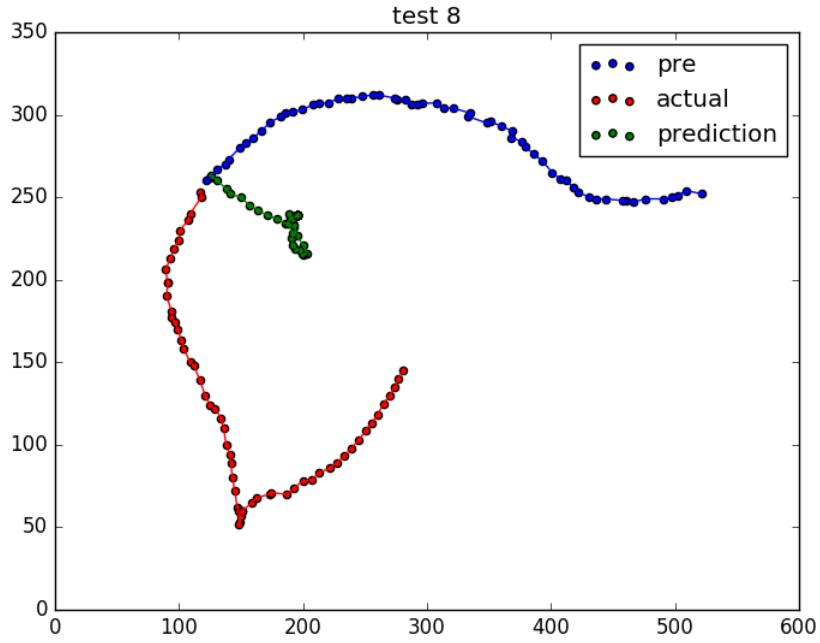
To the team's surprise, the only enrichment that improved the team's results was adding the Euclidian distance and heading. Adding running averages of 'N' data points showed inconsistent results for different test files and different algorithms. Therefore they were not considered as part of the team's final solution. Plots of each approach with un-enriched and enriched data have been provided on the following page for reference and support of above conclusions.

High Level Plot of the effect of various enrichments on performance:



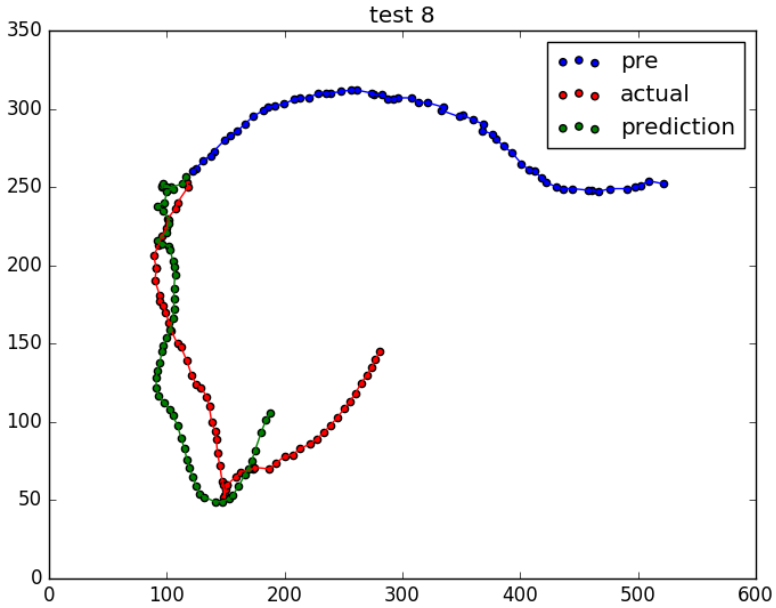
Un-Enriched Data Plot:

Using **KNN**, predicting the next point based on the previous (un-enriched data)



Enriched Data Plot:

Using **KNN**, predicting a single point based on enriched data (added velocity and heading)



Evaluation of Selected Approaches & Algorithms:

Summary of Approaches The Team Used

1. K-Nearest Neighbors (KNN) with enriched velocity and heading, single point prediction – This approach provided the best error rate however when looking at the visual plots the predicted and actual path often diverge quickly. We expected this as the error noise is increasing and each prediction is less certain than the previous.
2. Extremely Randomized Trees (Extra Trees) with enriched velocity and heading, single point prediction – this approach provided worse results on the test data, more investigation is required to explain the discrepancy
3. Both of the above with additional enriched data – as mentioned before, adding sliding averages for heading and velocity surprisingly did not improve either algorithm's results.
4. Extra Trees and KNN predicting the next 60 based on a sliding window of size k, where we tried k of sizes 5, 10, 20, 30, and 60. The team tested with and without enriched data. The results were very interesting
    a. Visually (see diagram below) the plots seem a little "closer" to the actual path
    b. Enriching the data seemed to have much less effect than method #1 and #2
    c. The error rate in most cases was close to the mean error rate we got in #1 (Which is the approach we used eventually for the submission). We chose to keep #1 as our final approach simply because we tested it more. Since the mean error of this approach was not significantly better than #1, and considering the time constraints, the team thought it was preferable not to significantly change our submission code in the last minute without very good reason.

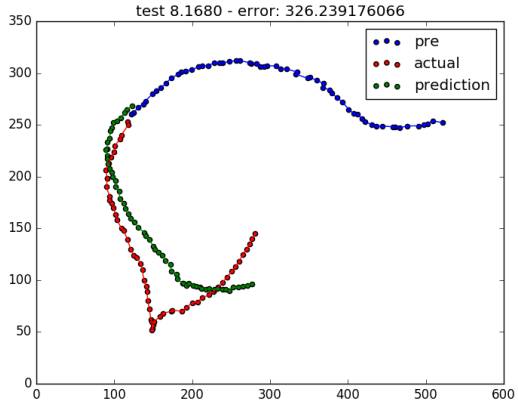


Fig A: approach #4 – using 60 samples sliding window, Extra Trees regression, predicting the next 60 (0 to 60 trained on 61 to 120, 1 to 61 trained on 62 to 121, n to n+60 trained on (n+1) to (n+1)+60 for n = 0 to training set size – 60)

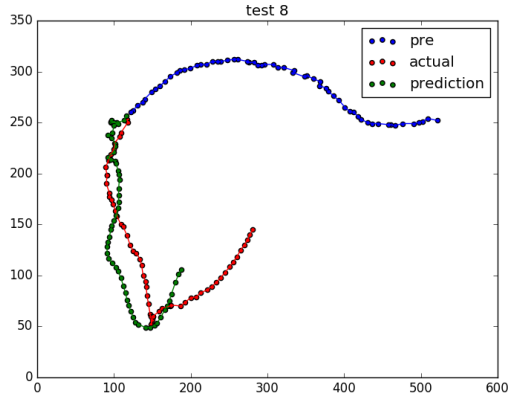


Fig B: approach #1 – using single point predicting the next, KNN regression.

(Note: Fig. B illustrates the team's final approach for project submission)

Implementation of Selected Approach

Traditionally Supervised Learning is used in the form of given a set of N training examples of the form $(x_0, y_0) \ldots (x_N, y_N)$ such that x_i is the feature vector for the i-th example and y_i is the corresponding known true value. The learner is then trained on the function $f: X \rightarrow Y$. Predictions can then be made in form of $\hat{Y} = f(\hat{X})$, where $\hat{X}$ is new unseen input by the learner.

For our input space, $X_N = (robot - x_N,\ robot - y_N,\ heading_N,\ velocity_N)$. The output space Y is defined, $Y_N = (robot - x_{N+1},\ robot - y_{N+1})$. For training our model we are using the provided 20 min of training data as well as the 60 sec of data provided at test time. While the requirement is to predict 60 time steps (2 sec at 30 frames/sec) ahead, we only trained the leaner to predict one-step ahead. So to predict 60 steps ahead, we are predicting one-step ahead from the last known step in the test data. We are then using our output as input for our learner for the remaining 59 steps.

This is displayed below assuming a test data set of size N:

1. Initialize: $\hat{y}_N = f(\hat{x}_N)$
2. While $i < 60$:
    a. $\hat{y}_{N+i+1} = f(\hat{y}_{N+i})$

The reason for using the one-step look ahead versus 60 was to reduce the amount of dimensions needed for our learner. Bellman's Curse of Dimensionality states in order for a model to converge to the true value of smooth function, the volume of data needed to support the model grows exponentially with the number of dimensions. By minimizing our dimensions, we believe we have maximized the utility of our training data.

Areas for Further Exploration:

Due to the time constraints and juggling of parallel commitments at work, the team was unable to explore every possible solution. Some ideas for areas of improvement and further exploration have been summarized below:

- HMM Recognition: possible use of Hidden Markov Models for better prediction based on current Markov state
- Adding Decay Rate: to compensate for compounding on inherent noise in data
- Boosting: as an effort to reduce bias and improve accuracy predictions
- Further Enrichment of Data: testing with running sub-sets of 'n' points, velocity, trajectory, smoothing

Guiding Principles

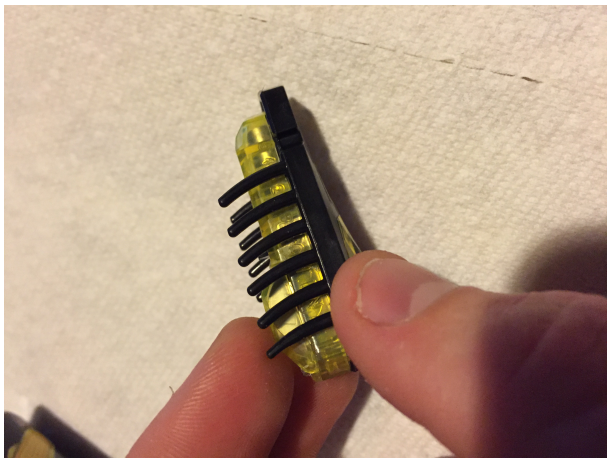
- Utilize CS8803 course concepts in solution design
- Evaluate multiple approaches to compare results
- Leverage external libraries as needed
- Avoid hardcoding

Assumptions

- Input data will be provided as per project specs
- It is acceptable to not use computer vision techniques, and read parsed location data from provided .txt files
- Programs will be executed on a standard deployment platform (provided VM)
- Any and all external libraries will be specified inside the team's readme.txt file. Any associated commands for installing/ running these libraries will also be included.
- Tools and methods from Python's scikit-learn library (http://scikit-learn.org/stable/) for machine learning are considered acceptable for the scope of this project.

Appendix:

As part of the project, the team decided to purchase a HEXBUG nano to get the chance to experience the project 'data' in real life and also play around with the technology. See below for a few images of the HEXBUG, and also its interactions with a team member's cat.



Resources Consulted and Works Cited:

The team utilized Python's scikit-learn library (http://scikit-learn.org/stable/) for machine learning aspects in our approach. Links to further documentation for some of the methods used have been provided below:

1. http://scikit-learn.org/stable/auto_examples/neighbors/plot_regression.html
2. http://scikit-learn.org/stable/supervised_learning.html
3. http://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeRegressor.html
4. http://scikit-learn.org/stable/auto_examples/plot_multioutput_face_completion.html
5. http://www.stat.ucla.edu/~sabatti/statarray/textr/node5.html