Where's my Ferry?

Support Vector Machines for Modeling Ferry Tardiness

Kyle Wenholz, advised by Professor Brad Richards

March 31, 2013

Contents

1 Introduction

Ferries (as in **Figure 1** present an interesting opportunity to examine a complex traffic system filled with data and affecting the lives of many individuals. This paper will discuss a new approach to predicting the timeliness of ferries using a support vector machine. An overview of how support vector machines work and their complications in real world use will introduce the discussion of how this powerful construct presents a powerful and intuitive model for tackling the massive data associated with ferries. A functioning model is developed to examine the complexities of this approach, the practicality, and as a means of exploring some of the data. Additionally, the hypothesis that the addition of weather information improves accuracy of the model is explored.

Figure 1: A prototypical WSDOT ferry on its route.

We will attempt to predict the tardiness of ferries, where tardy is defined as no less than three minutes later than the initial (at day's beginning) scheduled arrival or departure time. This definition of tardiness was chosen to allow for a non-trivial number of tardy events (more than 10% in the data) and to remain practical: i.e. someone could use the restroom in three minutes time or otherwise make use of that information. The Washington State Department of Transportation's ferry system in the Pacific Northwest served as the source for all data. As will be discussed later on, this ferry system is notoriously on time with over 95% of trips being on schedule (within one minute) [5]. This system is of particular interest for its sheer volume of passengers: 22 million per year [9]. The volume lends itself well to this project's goals of using data mining.

2 Preliminaries

Since the goal of the project was to use a large data set for machine learning and to develop a model for possibly predict future ferry tardiness, two approaches popular in available implementations and the literature were initially evaluated: artificial neural networks (ANNs) and support vector machines (SVMs). These two approaches fit into the category of **supervised learning**: labeled (known) data is fed into an algorithm that learns to make better predictions through comparing its own predictions to the labeled data set. ANN and SVM are often considered to be very similar in the problems they attempt to tackle, especially since each can be used as a linear classifier: a function taking in some object and determining a class to which it belongs. Rather than considering tardiness as a spectrum of degrees, we approach tardiness as a binary feature: late or not. This conceptually simplifies our problem. If artificial neural networks and support vector machines can both solve classification problems, however, which are we supposed to choose?

2.1 ANN and SVM

Byvatov et al examine the differences of the two approaches for classifying drugs [2]. While their model certainly isn't for a ferry system, their discussion suggests data sets with many features (an aspect of the ferry model we discuss later) are better suited for SVM, and they also find little difference in the predictions of ANN and SVM. Most importantly, their conclusion, and that of their references, is the two approaches are complementary. Each tool catches different cases, but the results are comparable in each approach, and SVM may even require less "tweaking". They concede that SVMs are generally easier to conceptualize, lending a sense of confidence to the work.

Support vector machines are more intuitively motivated. Consider mapping the features of a ferry trip (time of departure, weather, boat name, etc.) into a coordinate plane. Now, if we do this for all the points in our training set, we can also attach labels to the points (late or not, as in **Figure 2**). In two dimensions, we could try drawing a line through the late and not-late points to separate them. This line is simply a hyperplane in higher dimensions, and SVM has its theoretical underpinnings based on the idea of drawing this hyperplane optimally. Apart from the possible ease of use and understanding, there are also examples of SVM being used in similar traffic prediction systems, providing a baseline comparison and the final reason for choosing SVM over artificial neural networks.

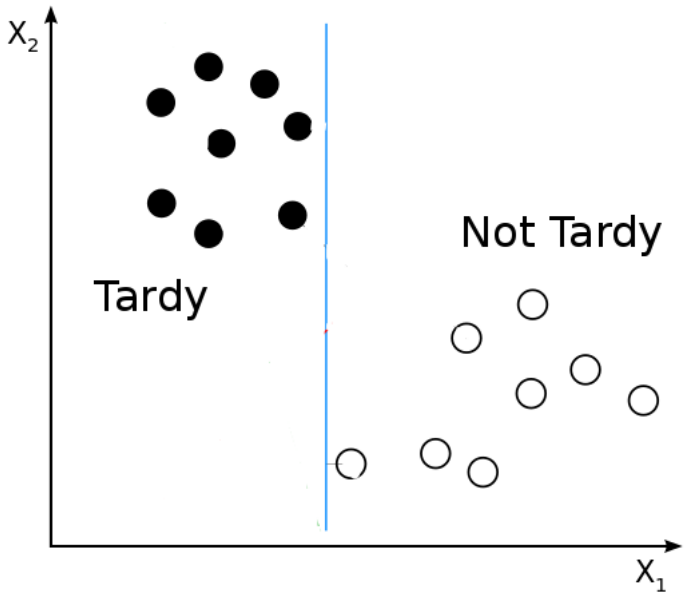

Figure 2: A simple example of a separating line for tardiness (i.e. hyperplane). Each data point (or instance) has features X_1 and X_2.

Smith et al uses SVM to predict the timeliness of airline traffic in conjunction with weather patterns [8]. Their goal is to predict the likelihood of requiring a "ground delay program": shuffling traffic to account for tardiness in the schedule. The model they use involves traffic flow management programs to estimate the effects of capacity on an air traffic system, and the SVM is trained on labeled data to produce a function predicting the need of a ground delay program and/or an actual delay. Their system was found to be 78% accurate in predicting need of a ground delay program and 83% accurate in predicting a delay. Air traffic is much like the ferry system given its sensitivity to weather, attempt to adhere to a strict schedule, and heavy use. Due to these similarities, Smith's access to resources like the AMPL supercomputer, and his group's expert knowledge, we use 80% as the target accuracy in for the ferry model.

It is also noteworthy that SVMs can be used for regression analysis in large data sets with many

variables [3]. While this paper will not cover this use of SVM, the possibility of performing such an analysis with the same tool was one reason in choosing support vector machines.

2.2 Basics of SVM

A brief overview of the principles behind support vector machines and their use will help to lay a basis for the process of the project. As stated earlier, SVMs are used to find a linear classifier (some function) for a set of data. In this project, we seek a way to separate ferry trips that will be three minutes past their scheduled time (late) or less (on time). To find the linear classifier, an SVM uses a feature vector, F, to describe the trip. An example F may have entries for

$$F = \begin{bmatrix} EstimatedArrival \\ BoatName \\ Temperature \\ WindSpeed \end{bmatrix}.$$

This feature vector needs to be entirely numerical, so categorical variables like boat name must be encoded as numbers. It is possible to assign boat names to different values, but most often, it is best to expand the feature like a bit vector. If there were boat names *SS Minnow, Death Star,* and *Millennium Falcon*, then we would have

$$F = \begin{bmatrix} EstimatedArrival \\ SS\_Minnow \\ Death\_Star \\ Millennium\_Falcon \\ Temperature \\ WindSpeed \end{bmatrix}$$

where the entries for boat names are a 1 if this trip has that name and 0 otherwise. For reasons detailed in [3] this is considered best practice.

Along with feature vectors for every trip, an SVM uses a weight vector, w, that it is the backbone for the linear classifier. Thus, this w is what the SVM trains. To determine which class a feature vector corresponds to the dot product of F and w is taken, with negative values corresponding to late and positive values to on time. While training on the labeled data, however, the SVM adjusts w whenever it predicts incorrectly. The amount to adjust can vary based on the implementation, but over time w improves on its ability to predict the class of any instance in the training set.

An optimal solution to w would form a hyperplane separating the instances perfectly by class. In **Figure 3** is an example of how an SVM might train a weight vector (represented graphically as a line) over a set of training data. The final vector, H_3, is considered optimal because it maximizes the distance between itself and the nearest points of each class. In the example, it is possible to separate the data with a line, but in the real world, and our ferry problem, this is rarely the case.

2.3 Kernel mapping

In **Figure 4** is an example of data we can't separate. Most real world data is like this: inseparable by a hyperplane. The problem this creates is that we're no longer looking for a linear classifier. A nonlinear classifier is required, a significantly more difficult problem. Since no optimal hyperplane exists, it is difficult to formulate how long an SVM "look" for the hyperplane. We might consider looking for

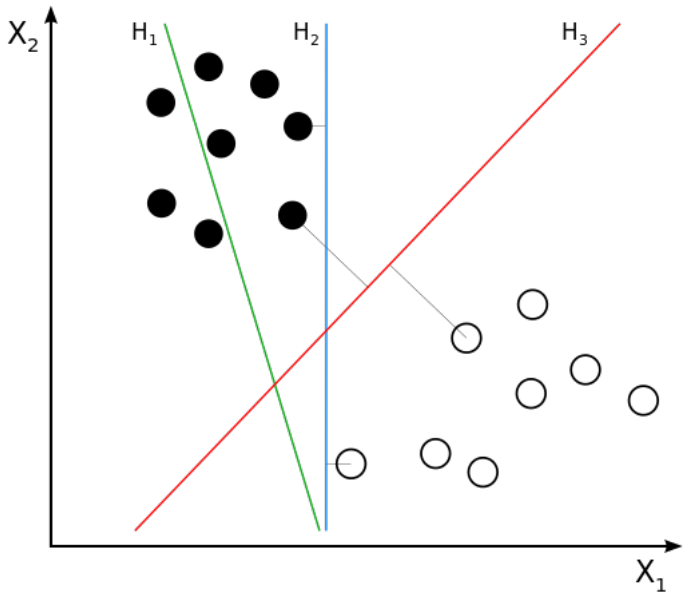

Figure 3: As in **Figure 2**, we are trying to separate labeled points with a line. An SVM does this iteratively, meaning line H_1 would be the first pass, H_2 the result of refitting, and H_3 the final classifier produced by the SVM.

something within reasonable bounds, but the kernel trick does away with this problem and brings us back to looking for a linear classifier [1].

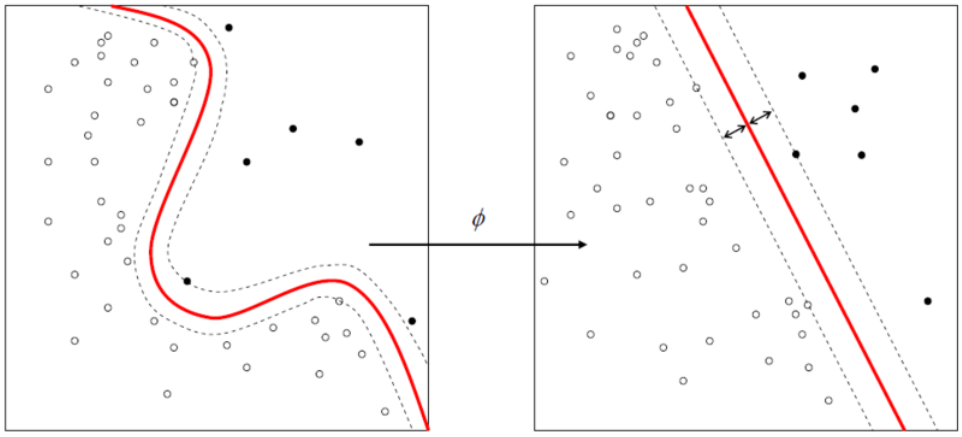

Figure 4: Separating the data in the first part requires a mapping into a higher (often much) dimension. This mapping makes it possible for a learning algorithm like SVM to find at least some separating hyperplane.

The kernel trick maps the instances of data into a higher dimensional space (often much higher), and the SVM works to find a separating plane in the new space. The example in **Figure 4** gives an idea of what the mapping would do to the data. The kernel function used in this project is a Gaussian radial basis function and was chosen for its popularity and since it is built into the LibSVM library [3] (discussed in the next section).

Using the kernel trick requires a small addition to the training process. Optimal parameters for the kernel function need to be found before fitting with the SVM. This is done by an exhaustive grid search:

1. The user supplies a space of parameters to try.

2. Within this space, a search grid is created.

3. Each grid square is "tried" by using parameters within to transform the data and then run the SVM on a subset of the training data.

So, training the kernel parameters requires training the SVM on very small sets of data many times to find the best parameters in a space provided by the user.

Implementing all of the functionality of a SVM and kernel function would require expert knowledge and a thorough amount of validation. Fortunately, libraries to work with SVMs and kernel functions already exist.

2.4 LibSVM

LibSVM [3] from Chih-Chung Chang and Chih-Jen Lin provides the SVM implementation used. This library has bindings to many languages, is reasonably straightforward to use from the command-line, is well tested, and has an acceptable amount of documentation. Using this implementation of an SVM allowed a great deal of time to be saved and provided a higher degree of certainty in the results.

3 Predicting tardiness through data

Support vector machines require two sets of labeled data to find a linear classifier: training and testing. The training data is fed into the SVM as a means of training the weight vector and finding a linear classifier, and the testing data is used after this process to check the accuracy of the SVM's product. It is crucial that these two sets be disjoint. Otherwise, the SVM is basically cheating by being tested on data it already learned from. There is no clear indicator for how much data is needed to successfully train an SVM, but too little means the SVM can't generalize and too much can cause overfitting. Overfitting is where an SVM is trained on too much data of similar qualities so that it fails to accurately classify data of different qualities. This roughly corresponds to finding patterns which only apply to a small subset of the real instances. So, it often comes down to the engineer to determine the appropriate amounts of data for the training and testing sets based on experience, experimentation, and availability.

3.1 Washington State's ferries

The Washington State Department of Transportation (WSDOT) runs the ferry system in the Pacific Northwest (since 1951) over a stretch of more than 130 miles, from Victoria, British Columbia to Point Defiance in Tacoma [6]. The system serves over 22 million passengers per year in about $160,000$ sailings [9]. We will refer to one sailing (from one terminal to another) as a trip. **Figure 5** gives the map of the WSDOT's 20 terminals and 10 routes, lending perspective to the size of the entire system.

The system is interesting for more than just its traffic. Several of the routes are rather windy looking, suggesting they may be more susceptible to weather and traffic delays. Some terminals, furthermore, are a hub for several routes. We might conjecture these terminals to be susceptible to pileups in traffic. These and other complexities of the system make it worthwhile in pursuing the robust model provided by an SVM. Fortunately, the complexities and heavy use of the system seem to have convinced the WSDOT to keep thorough records of the traffic.

To effectively use an SVM, we need enough data we can experiment with and it needs to be in a uniform, accurate state. Originally, a web scraper was written to grab information from the WSDOT VesselWatch [7] page, but this was made obsolete when a request for information was eventually fulfilled. The WSDOT provided, upon request, a record of approximately $340,000$ ferry trips from October of

Figure 5: The Washington State Department of Transportation ferry system. 20 terminals, making 10 routes, serve users from Point Defiance in Tacoma up to Victoria in British Columbia [7].

Vessel	Departing	Arriving	Sched Depart	Actual Depart
Kittitas	Mukilteo	Clinton	9/1/2010 0:00	9/1/2010 0:01
Sealth	Vashon	Southworth	9/1/2010 0:05	9/1/2010 0:06
Tacoma	Colman	Bainbridge	9/1/2010 0:15	9/1/2010 0:19

Initial ETA	Actual Arrival	Date	Route Name
9/1/2010 0:13	9/1/2010 0:14	9/1/2010	Mukilteo - Clinton
9/1/2010 0:16	9/1/2010 0:19	9/1/2010	Southworth - Vashon
9/1/2010 0:45	9/1/2010 0:51	9/1/2010	Seattle - Bainbridge Island

Figure 6: The header of the data provided from the WSDOT (broken into two tables to fit page widths). There are over 340,000 lines in the original file.

2010 to October of 2012. The data came as a comma separated value file with the nine fields. An example is shown in **Figure 6**.

The column headings are all fairly straightforward, but it is worth noting the scheduled and initial values, for departure and arrival respectively, are the times found on the schedule for the beginning of the day. With the way the data is given, we separate out the times into departures and arrivals, creating two separate scheduling problems. This way it is possible to try predicting arrivals with the added information of departure time (a seemingly easier problem).

Before feeding the data into the SVM, it was necessary to transform times and categorical variables into numeric values. This was done through several Python scripts: using seconds past January 1, 1970 for the times and expanding the categoricals out as many variables functioning like a bit vector. Additionally, and a far more difficult problem, we joined this ferry data with weather data obtained from the National Oceanic and Atmospheric Administration (NOAA).

3.2 Weather and final SVM format

Retrieving accurate and full historical weather data presented an unexpected challenge. Most weather sites are focused on future weather and make historical data available only through programming APIs

WBAN	Date	Time	StationType
SkyCondition	SkyConditionFlag	Visibility	VisibilityFlag
WeatherType	WeatherTypeFlag	DryBulbFarenheit	DryBulbFarenheitFlag
DryBulbCelsius	DryBulbCelsiusFlag	WetBulbFarenheit	WetBulbFarenheitFlag
WetBulbCelsius	WetBulbCelsiusFlag	DewPointFarenheit	DewPointFarenheitFlag
DewPointCelsius	DewPointCelsiusFlag	RelativeHumidity	RelativeHumidityFlag
WindSpeed	WindSpeedFlag	WindDirection	WindDirectionFlag
ValueForWindCharacter	ValueForWindCharacterFlag	StationPressure	StationPressureFlag
PressureTendency	PressureTendencyFlag	PressureChange	PressureChangeFlag
SeaLevelPressure	SeaLevelPressureFlag	RecordType	RecordTypeFlag
HourlyPrecip	HourlyPrecipFlag	Altimeter	AltimeterFlag

$$94274, 20110101, 0053, 12, CLR, , 10.00, , , , 25, , -3.9, , 22, , -5.4, ,$$

$$16, , -8.9, , 69, , 0, , 000, , , , 29.69, , 1, , 002, , 30.04, , AA, , ,$$

$$, 30.03,$$

Figure 7: An example line from a weather data file preceded by the many, but thankfully explicit, column headings.

which may require a paid subscription. Fortunately, NOAA provides (by subscription) a portal for retrieving weather data by station. Stations record over 20 different features at some point every hour. It seems that some stations experience outages or are unable to record on some instruments every so often; because of this, we used the station with the seemingly most complete data set in the area. This turned out to be a station near the Tacoma Narrows Bridge. The data was downloaded in files by month for the range of October 2010 to October 2012 from a quality controlled store on NOAA's site [4]. Only about $1,000$ of the more than $24,000$ weather entries had to be removed for null values in the wind data. Unfortunately, the station is not a central point, but it seemed close enough at the time.

In **Figure 7** is an example of what the weather data looks like in its raw form. Note there are some categorical variables, but nearly everything is a numeric. The categorical variables weren't utilized in the final model, since they can be inferred from the other data. Many of the columns are simply blank, due to the stations lack of equipment for certain measures. The measures for dry bulb, wet bulb, dew point, wind speed, wind direction, and station pressure were used in the final model. There was very little missing data for any of these, and none of them directly imply another. Humidity may seem to be missing, but it can be calculated by the dry bulb and wet bulb temperatures.

We avoided using multiple stations to keep the joining of ferry and weather data simpler: since all ferries matched with one station, we just found the weather recording closest (within at least an hour) of the actual time reported on the ferry trip. Of the over $340,000$ ferry trips, only about $13,000$ had to be removed for lack of weather data. With a data set so large, this only constituted 4%, an acceptable loss in our case. To join the data, a simple Python script searched the weather data for a nearest time to join with each ferry trip, and then the departure and arrival information were separated to place each as an individual event in separate files. It was then straightforward to calculate the label (late or not) for each event by subtracting the appropriate times.

Following the joining of weather and ferry data and the labeling, it is necessary scale the data between -1 and 1, since this is considered an SVM best practice to make no feature stand out by sheer magnitude [**?**]. LibSVM performs this transformation quickly and is also useful for separating out the training data from the testing data. We used it to randomly sample about $290,000$ data points for training and $30,000$ for testing. The sampling was random to the extent that both sets of data contained

a roughly equal distribution of late and not-late ferry trips. This is an incredibly important feature to preserve, since we want to both learn about tardiness and check if the linear classifier produced is able to predict both scenarios. With separated data in hand, we proceeded to train a linear classifier using the LibSVM package.

4 Results

The results for the project exist in three parts. As described before, the primary goal is to model the ferry system and answer the question of whether or not using an SVM to train a linear classifier is useful. This leads to the first two phases which involve a first pass and then a serious refining of the methodology. As a benchmark, we use approximately 90% accurate on the test data to be comparable with Smith et al [8], since like that group, we wish to see if using SVM is a practical method for determining ferry tardiness. Recall that a secondary goal is to find which features matter in determining tardiness. While this goal is asking questions best answered by a regression analysis, we will attempt a crude approximation through empirical methods. The final component is a small experiment to test interesting phenomena arising in the general results. This latter part is possibly the most interesting of the project, but we begin with the first attempt to use LibSVM.

4.1 The first pass

We began by determining whether or not a ferry is on time for its departure, leaving arrival for after the process was worked out. A file with all of the labeled training instances was fed into LibSVM, using the default parameters of the kernel. The process took an entire night running on low end hardware, but the results were disappointing to say the least.

After running on the test data, the linear classifier only yielded 60% accuracy. This is significantly beneath the benchmark. The time to run the process was also surprising and led to a more cautious approach for the rest of the work. To improve the performance of the classifier, LibSVM provides an exhaustive guide [**?**] for beginners in the perils of simply "running it". The directions weren't followed on the first pass because it was hoped the process would be much simpler just using the default parameters.

Having learned some lessons, the LibSVM guide was followed to the letter on a second pass. The most significant change to the process was training on the parameters for the kernel function. As mentioned before LibSVM provides code to perform an exhaustive search for the best fit parameters to use in the function. The subsets for this training came from the training data.

4.2 General results

4.3 The problem with weather

5 Closing thoughts

References

[1] A Aizerman, Emmanuel M Braverman, and LI Rozoner, *Theoretical foundations of the potential function method in pattern recognition learning*, Automation and remote control **25** (1964), 821–837.

[2] Evgeny Byvatov, Uli Fechner, Jens Sadowski, and Gisbert Schneider, *Comparison of support vector machine and artificial neural network systems for drug/nondrug classification*, Journal of Chemical Information and Computer Sciences **43** (2003), no. 6, 1882–1889.

[3] Chih-Chung Chang and Chih-Jen Lin, *Libsvm: a library for support vector machines*, ACM Transactions on Intelligent Systems and Technology (TIST) **2** (2011), no. 3, 27.

[4] National Oceanic and Atmospheric Administration, *NCDC: Quality Controlled Local Climatological Data*, `http://cdo.ncdc.noaa.gov/qclcd/QCLCD`, October 2012.

[5] Washington State Department of Transportation, *A Comparison of Operational Performance: Washington State Ferries to Ferry Operators Worldwide*, `http://www.wsdot.wa.gov/Research/Reports/700/750.1.htm`, November 2012.

[6] ———, *Washington State Ferries Our Fleet*, `http://www.wsdot.wa.gov/Ferries/yourwsf/ourfleet/`, October 2012.

[7] ———, *WSDOT - Ferries - VesselWatch*, `http://www.wsdot.com/ferries/vesselwatch/Default.aspx`, January 2012.

[8] David A Smith, Lance Sherry, and G Donohue, *Decision support tool for predicting aircraft arrival rates, ground delay programs, and airport delays from weather forcasts*, International Conference on Research in Air Transportation (ICRAT-2008), 2008.

[9] Washington State Department of Transportation, *WSF Traffic Statistics*, `http://www.wsdot.wa.gov/ferries/traffic_stats/`, October 2012.