Learning User Preferences in Matching for Ridesharing

Mojtaba Montazery and Nic Wilson
Insight Centre for Data Analytics, School of Computer Science and IT
University College Cork, Ireland
{*mojtaba.montazery, nic.wilson*}*@insight-centre.org*

Keywords: Dynamic Ridesharing, Preference Learning.

Abstract: Sharing car journeys can be very beneficial, since it can save travel costs, as well as reducing traffic congestion and pollution. The process of matching riders and drivers automatically at short notice, is referred to as dynamic ridesharing, which has attracted a lot of attention in recent years. In this paper, amongst the wide range of challenges in dynamic ridesharing, we consider the problem of ride-matching. While existing studies mainly consider fixed assignments of participants in the matching process, our main contribution is focused on the learning of the user preferences regarding the desirability of a choice of matching; this could then form an important component of a system that can generate robust matchings that maintain high user satisfaction, thus encouraging repeat usage of the system. An SVM inspired method is exploited which is able to learn a scoring function from a set of preferences; this function measures the predicted satisfaction degree of the user regarding specific matches. To the best of our knowledge, we are the first to present a model that is able to implicitly learn individual preferences of participants. Our experimental results, which are conducted on a real ridesharing data set, show the effectiveness of our approach.

1 INTRODUCTION

Ridesharing (also carpooling and lift-sharing) is a mode of transportation in which individual travellers share a vehicle for a trip. Increasing the number of travellers per vehicle trip by effective usage of spare car seats may of course enhance the efficiency of private transportation, and contribute to reducing traffic congestion, fuel consumption, and pollution. Moreover, ridesharing allows users to split travel costs such as fuel, toll, and parking fees with other individuals who have similar itineraries and time schedules. Conceptually, ridesharing is a system that can combine the flexibility and speed of private cars with the reduced cost of fixed-line systems such as buses or subways (Furuhata et al., 2013; Agatz et al., 2011).

Ridesharing is quite an old concept; it was first used in USA during World War II to conserve resources for the war. It reappeared as a result of the oil crisis in 1970s which led to the emergence of the first ridesharing algorithms. Nevertheless, ridesharing usage declined drastically between the 1970s and the 2000s due to the decrease in the price of fuel and vehicle ownership cost (Chan and Shaheen, 2012).

Furthermore, there are some challenges that have inhibited wide adoption of ridesharing. A few of the most important of those are listed as follows:

Riding with Strangers Surveys suggest little interest in sharing a ride with strangers because of personal safety concerns. This phenomenon is called *Stranger Danger* and could be alleviated by incorporation of social networks (Amey et al., 2011; Furuhata et al., 2013). (Chaube et al., 2010) conducted a survey among students of a university which shows that while only 7% of participants would accept rides from a stranger, 98% and 69% would accept rides from a friend and the friend of a friend, respectively.

Reliability of Service One of the largest behavioral challenges is the perception of low reliability in ridesharing arrangements; the parties may not necessarily follow through on the agreed-upon ride. For instance, if the driver has an unexpected appointment or emergency, the passenger may be left with no ridesharing option; or, from the other side, drivers might be required to wait because of a passenger being late (Amey et al., 2011).

Schedule Flexibility The lack of schedule flexibility has been one of the longest running challenges in ridesharing arrangements. Drivers and passengers often agree to relatively fixed schedules and meeting locations, not allowing much flexibility. It is

interesting to note that increased flexibility and increased reliability in ridesharing arrangements are often conflicting objectives (Amey et al., 2011).

Ride Matching Optimally matching riders and drivers –or at least getting a good match–is among most important challenges to be overcome. This can lead to a complicated optimization problem due to the large number of factors involved in the objective function. We will discuss this aspect of ridesharing further in Section 2.

Despite the above barriers to ridesharing, the demand for ridesharing services has increased again sharply in recent years, generating much interest along with media coverage (Saranow, 2006). This boost in ridesharing is mainly associated with a relatively new concept in ridesharing, *dynamic* or *real-time* ridesharing. Dynamic ridesharing refers to a system which supports an automatic ride-matching process between participants at short notice or even en-route (Agatz et al., 2012).

Technological advances, both hardware and software, are key enablers for dynamic ridesharing. The first influential fact is that the smartphones are becoming increasingly popular (Emarketer, 2014; Smith, 2015). As the first impact of smartphones on ridesharing, they provide an infrastructure on which a ridesharing application can run, replacing the old-fashioned, sometimes not so convenient, approaches such as phone or website. More importantly, smartphones are usually equipped with helpful communication capabilities, including global positioning system (GPS) (Zickuhr, 2012) and network connectivity (Duggan and Smith, 2013).

Dynamic ridesharing by its nature is able to ease some aspects of existing challenges in traditional ridesharing. For example, tracking participants by means of GPS could mitigate safety concerns or increase the reliability. In terms of flexibility, since dynamic ridesharing does not necessarily require long-term commitment, users have the option to request a trip sharing day-by-day or whenever they are pretty sure about their itinerary.

Even though the above advancement in technology could be beneficially available, ridesharing is still in short supply. In this study, we focus on the ride-matching problem which is central to the concept. However, we are mindful of the fact that there are a number of other challenges should be dealt to accomplish the ultimate success of ridesharing.

The rest of the paper is structured as follows. In Section 2, we model automated ride-matching problem and explain how user preferences could be considered in the matching process. In Section 3, a method to learn user preferences is described in detail. Section 4 evaluates the presented approach on a real ridesharing database. Finally, in Section 5, we summarize the main remarks and discuss some directions for future research.

2 AUTOMATED RIDE-MATCHING

The essential element of dynamic ridesharing is the automation of the trip matching process, which allows trips to be arranged at short notice with minimal effort from participants. This means that a system helps riders and drivers to find suitable matches and facilitates the communication between participants (Hwang et al., 2006; Agatz et al., 2012).

In order to model the matching problem, two disjoint types of ridesharing request are considered: a set of requests in which the owner of the request are drivers (D), and requests created by riders (R). Hence, all trip requests could be represented by the set $\mathcal{S} = R \cup D$. Then, ridesharing requests are represented as a bipartite graph $G = (D, R, E)$, with E denoting the edges of the graph. This setting is extendable for the case when some participants are flexible with being driver or rider.

This graph becomes a weighted graph by assigning a weight c_{ij} to the edge (S_i, S_j), where $S_i, S_j \in \mathcal{S}$. Generally speaking, c_{ij} quantifies how much is gained by matching S_i and S_j. This weight could be a composition of an assortment of factors, related to the overall system, or to individual travellers.

For finding optimal matchings, one approach popular in the literature is solving an optimization problem in which the sum of the benefits from proposed matching should be maximized (Agatz et al., 2011). To do this, a binary decision variable x_{ij} is introduced that would be 1 when the match (S_i, S_j) is proposed, and 0 if not. Then, the objective function to be maximized is $\sum_{i,j} x_{ij} c_{ij}$. After running the solver, a fixed schedule is proposed to users as the optimal solution.

However, this approach neglects a crucial requirement of a practical system, that is, getting users confirmation before fixing a ride-share for them. Although earlier we emphasised the concept of automation in ride-matching which attempts to minimize users' efforts, we believe that it couldn't be fully automatic. In fact, it is hard in practice to convince users to share their ride with somebody without their final agreement.

For this reason, in this study, we suggest a novel attitude towards ride-matching problems, by looking at the problem as a recommendation system rather than an optimization problem. In this setting, the sys-

tem just recommends a set of best possible matchings to each individual with respect to the weights c_{ij}. Note that not only does c_{ij} take into account participants desires, but also it could incorporate the overall system benefits.

In terms of the system's benefits–which ultimately could result in less pollution, traffic congestion etc.– – two measures are often mentioned in the related studies; the saved travel distance (d_{ij} obtained from the match (S_i, S_j)) and the saved travel time (t_{ij}) (Furuhata et al., 2013; Agatz et al., 2012).

The second aspect of c_{ij} is how much the two individuals, who involved in the matching (S_i and S_j), are happy with that matching. While positional and temporal elements of a ride sharing usually play a more important role in forming users' satisfaction degree, users' social preferences such as the other party's reputation and gender, smoking habit and so forth are also suggested as a relevant component (Ghoseiri et al., 2011).

The common strategy for finding arcs' weights (c_{ij}) is making a composition of the system's benefits (e.g., $c_{ij} = d_{ij} + t_{ij}$) and posing some constraints which expresses users' limitations (Herbawi and Weber, 2012; Agatz et al., 2011). Nevertheless, there are two main shortcomings have not been fully addressed by adoption of this strategy:

Only Hard Constraints The fact is that users might sacrifice some of their desires in favour of another ones. For instance, a rider who likes smoking in the car may decide to share his trip in a non-smoking vehicle due to other favourable conditions. Therefore, having soft constraints instead of hard ones may often be more plausible. Posing soft constraints could be rephrased as considering users' preferences, the term that is mainly used in this paper.

Eliciting Complicated Parameters In order to set constraints, there is a prerequisite to ask users to specify several parameters explicitly; such as earliest possible departure time, latest possible arrival time and maximum excess travel time (Baldacci et al., 2004; Agatz et al., 2011; Amey et al., 2011). However, elicitation of such parameters for every trip is quite impractical in real world situations. As an example, users can simply state their departure time, but finding how much they are flexible with that time is not a straightforward task. Moreover, some participants may be hesitant or unwilling to disclose certain preferences for privacy reasons.

Our solution for attempting to overcome the aforementioned issues is learning each user's preferences based on his (or her) past ridesharing records. For instance, we can learn from his previous choices that a particular user is normally more flexible with time than location. We will discuss more about this subject in Section 2.1. As a result of learning weights, besides modeling soft constraints in form of users' preferences, there is no need to directly ask those cumbersome parameters from participants anymore.

Thus, an arc's weight could be the combination of the system's benefits and those learned weights. To mathematically illustrate, assume $c_{ij} = (w_{ij} \cdot w_{ji}) \cdot (d_{ij} + t_{ij})$ where w_{ij} indicates how much sharing the ride with the user j is favourable for the user i, and the converse for w_{ji}. The weights w_{ij} and w_{ji} are learned from the previous records of the users i and j, respectively.

2.1 Learning Weights

In this section, the mechanism of learning users' preferences from the previous ridesharing records will be characterized. To give an intuitive sense, we start with an example.

Example 1. Assume there is a user who has a daily trip at a specific time (e.g., every day at 8 : 30 from the location A to B). This trip request is denoted by S_1.

On Day 1, the system recommends three matching opportunities, namely S_2, S_3 and S_4. The user rejects S_2, accepts S_3 and leaves S_4 without any response. Ignoring of a case may potentially mean that the user had been tentative about it.

On the second day, two recommendations, S_5 and S_6, are suggested to the user. S_5 is rejected and S_6 is accepted. While the user was waiting for the response of the other party in S_6, he receives another suggestion, S_7; then, the user cancels his previous offer (to S_6) and accepts S_7.

On the third day, the system has found two feasible matches which are S_8 and S_9. The goal is to evaluate how much these two opportunities are desirable for the user. Using the notation of the previous section, we would like to find $w_{1,8}$ and $w_{1,9}$ which are finally incorporated in the calculation of $c_{1,8}$ and $c_{1,9}$, respectively. Table 1 summarizes this example.

At first glance, the problem might seem to be a classification problem because the supervision (labels) is in the form of classes (i.e., Acceptance, Rejection etc.). However, a closer look suggests that learning a classifier is not a correct choice. The first reason is that there is a logical order between classes in the current case (e.g., Acceptance has a highest value and Rejection the lowest value), whereas classification is often applied when there is no ordering between class

Table 1: The scenario described in Example 1 is summarized here. The user's responses are Acceptance(A), Cancellation(C), Ignoring(I) and Rejection(R).

Day 1	Day 2	Day 3 (Today)
(S_2,R)	(S_5,R)	$S_8(w_{1,8}=?)$
(S_3,A)	(S_6,C)	$S_9(w_{1,9}=?)$
(S_4,I)	(S_7,A)	

labels, such as classifying fruits into three categories, apple, pear and orange. Secondly, a classifier will predict a class label whereas here, a scalar value is required.

If each class label is replaced by an appropriate number which keeps the natural ordering, the problem could be considered as a regression problem. For instance, the number 1 might be assigned for Acceptance, 0.66 for Cancellation, 0.33 for Ignoring and 0 for Rejection.

In spite of the fact that learning a regressor has none of those defects mentioned about classification, we believe that it still suffers from the following flaws:

- The first point is that the user's response to a recommended opportunity not only depends on the properties of that particular case, but also depends on other rival opportunities. Taking Example 1 to illustrate, it couldn't be said that if S_3 existed in the suggestion list on the day 2, it would certainly be accepted again, because the user accepted it in presence of S_2 and S_4 which does not guarantee its acceptance when S_6 and S_7 are available.
- Two class labels do not necessarily have the same desirability distance for all instances. Consider Example 1; assigning value numbers to classes as described above suggests that the difference between S_2 and S_3 from the first day, and S_5 and S_7 from the second day are both 1. However, the only thing that is known is that the user preferred S_3 to S_2 and S_7 to S_5, not the extent of the difference.

To address the above issues, we suggest considering the supervision in the form of qualitative preferences. This means that instead of assigning value numbers, sets of preferences among alternatives are derived from the user's choices. The following preferences set could be formed for Example 1:

$$\lambda = \{(S_3 \succ S_4), (S_4 \succ S_2), (S_7 \succ S_6), (S_6 \succ S_5)\}.$$

In the next section, a model will be proposed to learn a scoring function by making use of this kind of preferences set.

3 LEARNING MODEL: SVPL

3.1 Basic Formulation

As described above, the primary capability of the method developing in this section should be learning a scoring function from a set of preferences, expressed between several alternatives (i.e., preferences between suggested ridesharing trips that are derived from the user's feedback). This scoring function can generate a value number for each alternative (i.e., ridesharing trip opportunity), measuring the expected desirability degree of that alternative for the user.

Mathematically speaking, the set of alternatives is expressed as $\mathcal{X} = \{X_k | k \in \{1,\dots,m\},\ X_k \in \mathbb{R}^d\}$ in which d is the dimension of the features vector (with each feature representing a different property of the trip). We formulate the preferences set as $\lambda = \{(X_i \succ Y_i) | i \in \{1,\dots,n\},\ X_i, Y_i \in \mathcal{X}\}$. Then, the goal is finding a function $f(X) : \mathbb{R}^d \to \mathbb{R}$ which maps a features vector to a scalar value and is in agreement with the preferences set λ, i.e., if $X_i \succ Y_i$ is in λ then we aim to have $f(X_i) > f(Y_i)$. In order to achieve this goal, we developed a derivation of conventional SVM which eventually turns out to be similar to the approach proposed in (Joachims, 2002) as SVMRank.

In the rest of this section, the notation λ_i (corresponding to $(X_i \succ Y_i)$) represents the *preference point* in $\mathbb{R}^d$, given by $\lambda_i = X_i - Y_i$. We say that preferences points are consistent if there is a scoring function f satisfies the following condition:

$$f(\lambda_i) > 0 \ \ \forall i. \tag{1}$$

This condition comes from the fact that for all preferences points, the condition $f(X_i) > f(Y_i)$ should be true. Here, in Section 3.1, we assume that all preferences points are consistent, meaning there is at least one scoring function able to satisfy Eq. (1). In Section 3.2, we consider the inconsistent case.

Another assumption being made in this section is that the scoring function is a linear function. So, there is a vector ω with d components such that

$$f(X) = \omega \cdot X. \tag{2}$$

We will consider the case of a non-linear scoring function in Section 3.3.

With respect to these two assumptions, it can be said that if h is the hyperplane $f(X) = 0$ (which is a line in $d = 2$ case), then the hyperplane should include the origin, and all preferences points should be in the associated positive open half-space, given by h. This condition is mathematically written as:

$$\omega \cdot \lambda_i > 0 \ \ \forall i, \tag{3}$$

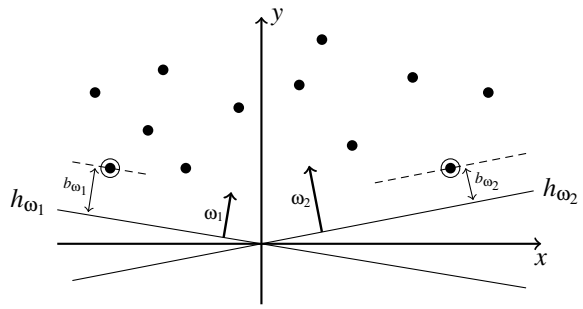

Figure 1: Two samples of plausible hyperplanes (h_{ω_1} and h_{ω_2}) and their associated normal vectors (ω_1 and ω_2) for a set of consistent preferences points are illustrated. b_{ω_1} and b_{ω_2} are the margins of h_{ω_1} and h_{ω_2}, respectively.

where ω is the normal vector to the hyperplane h. Two examples of plausible hyperplanes (h_{ω_1} and h_{ω_2}) for a consistent set of preferences points are illustrated in Figure 1.

The *margin* of a hyperplane, denoted by b, is defined as the distance from the hyperplane to the closest preference point; for the hyperplane h_ω, it could be found by:

$$b_\omega = \frac{\min_i \omega \cdot \lambda_i}{\|\omega\|}, \tag{4}$$

where $\|\omega\|$ is the Euclidean norm of ω ($\|\omega\|^2 = \omega \cdot \omega$).

Based on the principal idea in SVM (Burges, 1998), among all hyperplanes that can meet the condition stated in Equation (3), we look for the hyperplane that produces the largest margin, since it seems reasonable that for the preferences points, the maximum marginal space from the condition boundary is desirable. Thus, in order to find the optimal ω (say ω^*) and consequently $f(X)$, we need to solve an optimization problem as follows:

$$\omega^* = \arg\max_\omega \frac{\min_i \omega \cdot \lambda_i}{\|\omega\|} \tag{5a}$$

subject to

$$\omega \cdot \lambda_i > 0 \ \ \forall i. \tag{5b}$$

Let $a_\omega = \min_i \omega \cdot \lambda_i$ where according to Eq. (5b) a_ω should be a positive number. Then we have:

$$\omega^* = \arg\max_\omega \frac{a_\omega}{\|\omega\|} \tag{6a}$$

subject to

$$\omega \cdot \lambda_i \geq a_\omega > 0 \qquad \forall i. \tag{6b}$$

Let $\omega' = \frac{\omega}{a_\omega}$, where of course ω' like ω is a normal vector to the hyperplane h_ω, because the positive scalar value a_ω just affects the magnitude of the vector not its direction. As a result, the problem could be simplified by replacement of ω with the term $a_\omega\omega'$. That means the objective function (Eq. 6a) will be:

$$\frac{a_\omega}{\|\omega\|} = \frac{a_\omega}{a_\omega\|\omega'\|} = \frac{1}{\|\omega'\|}$$

and the constraint (Eq. 6b) is modified like this:

$$\begin{aligned} \omega \cdot \lambda_i &\geq a_\omega \\ a_\omega\omega' \cdot \lambda_i &\geq a_\omega \\ \omega' \cdot \lambda_i &\geq 1 \end{aligned}$$

Hence, the problem is reformulated as the following:

$$\omega'^* = \arg\max_{\omega'} \frac{1}{\|\omega'\|} \tag{7a}$$

subject to

$$\omega' \cdot \lambda_i \geq 1 \ \ \forall i. \tag{7b}$$

Note that ω'^* found from Eq. (7) is not necessarily equivalent with ω^* found from Eq. (6), but as described above, both give an equivalent hyperplane ($h_\omega \equiv h_{\omega'}$). So, without loss of generality, we can use the symbol ω instead of ω' in Eq. (7).

To have a more standard form, the arrangement of the problem is reformulated as a Minimization problem in this manner:

$$\omega^* = \arg\min_\omega \frac{1}{2}\|\omega\|^2 \tag{8a}$$

subject to

$$1 - \omega.\lambda_i \leq 0 \qquad \forall i \tag{8b}$$

Now, this is a convex quadratic programming problem, because the objective function is itself convex, and those points which satisfy the constraints are also from a convex set (a set of linear constraints define a convex set).

In Section 3.3, more explanation will be given as to why we are interested in the form of problem formulation (both objective function and constraints) in which preferences points only appear in the form of pairwise dot products. To fulfill this demand, we switch to the *Lagrangian Dual Problem*.

For our problem, the *Lagrangian function* which is basically obtained by augmenting the objective function with a weighted sum of the constraint functions, is like this:

$$\mathcal{L}(\omega, \mu) = \frac{1}{2}\|\omega\|^2 + \sum_{i=1}^{n} \mu_i \ (1 - \omega.\lambda_i) \tag{9}$$

where μ_i is referred as the *Lagrange multiplier* associated with the i^{th} inequality constraint $1 - \omega.\lambda_i \leq 0$ (Boyd and Vandenberghe, 2004, Chapter 5).

The optimization problem stated in Eq. (8) is called the primal form; where the Lagrangian dual form is formulated in this fashion:

$$(\omega^*, \mu^*) = \arg\max_{\mu} \inf_{\omega} \mathcal{L}(\omega, \mu) \tag{10a}$$

subject to

$$\mu_i \geq 0 \qquad \forall i. \tag{10b}$$

Because the primal form is convex and Slater's condition holds, we say the *strong duality* holds for our problem (Boyd and Vandenberghe, 2004, Sec. 5.2.3). This means that the optimal value for the dual problem equals the optimal value for the primal form. As a consequence, solving the Lagrangian dual form (Eq. 10) is equivalent to solving the primal form of the problem (Eq. 8).

As stated, strong duality is obtained for our problem, and also the objective function is differentiable. Regarding these two criteria, an optimal value must satisfy the *Karush Kuhn Tucker (KKT)* conditions (Boyd and Vandenberghe, 2004, Sec. 5.5.3). Here, we just exploit the *stationary* condition of KKT which states:

$$\frac{\partial \mathcal{L}}{\partial \omega} = 0 \quad \text{i.e., } \omega = \sum_{i=1}^{n} \mu_i \lambda_i \tag{11}$$

We substitute equality constraint (11) into Eq. (9) which leads to a remodeled Eq. (10), in the form that we are interested in:

$$\mu^* = \arg\max_{\mu} \mathcal{L} = \sum_{i=1}^{n} \mu_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \mu_i \mu_j \lambda_i \cdot \lambda_j \tag{12a}$$

subject to

$$\mu_i \geq 0 \qquad \forall i. \tag{12b}$$

At this stage, we have just a simple positivity condition on μ_i which is more manageable than Eq. (8b) and more importantly, the preferences points are shown just in the form of dot products ($\lambda_i \cdot \lambda_j$).

Solving this new form of the problem gives optimal values for the Lagrange multipliers (μ^*); we then can use Equation (11) to find ω^* as well.

3.2 Handling Inconsistencies

So far, we assumed that there exists at least one hyperplane such that all preferences points are placed in the positive open half-space of it. However, it is plausible that in many practical cases this assumption may result in finding no feasible solution. To handle inconsistencies, we develop the basic model by reformulating the initial constraint (8b) such that it could be violated with an appropriate amount of cost.

For this purpose, the constraint (8b) is rewritten as $1 - \omega.\lambda_i \leq \xi_i$ where ξ_i represents the cost of violating the i^{th} constraint and obviously should be non-negative ($\xi_i \geq 0$). Then, the objective function is augmented by the term $C \sum_{i=1}^{n} \xi_i$ to reveal the effect of costs, where C is a constant parameter to be chosen by the user. The parameter C scales the impact of inconsistent points; a larger C corresponds to assigning a higher penalty to errors. Thus, the primal form of the problem becomes the following:

$$\omega^* = \arg\min_{\omega, \xi_i} \frac{1}{2}\|\omega\|^2 + C \sum_{i=1}^{n} \xi_i \tag{13a}$$

subject to

$$1 - \omega.\lambda_i - \xi_i \leq 0 \qquad \forall i, \tag{13b}$$

$$\xi_i \geq 0 \qquad \forall i. \tag{13c}$$

This optimization problem (Eq. 13) is equivalent to *Optimization Problem 1* (Ranking SVM) proposed in (Joachims, 2002).

Because we have a new set of constraints ($\xi_i \geq 0$), a new set of positive Lagrange multipliers $\alpha_i (i = 1, ..., n)$ are introduced. The Lagrangian changes as follows:

$$\mathcal{L} = \frac{1}{2}\|\omega\|^2 + C \sum_{i=1}^{n} \xi_i + \sum_{i=1}^{n} \mu_i \left(1 - \omega.\lambda_i - \xi_i\right) - \sum_{i=1}^{n} \alpha_i \xi_i \tag{14}$$

The stationary KKT condition entails the new equality constraints:

$$\frac{\partial \mathcal{L}}{\partial \xi_i} = 0 \quad \text{i.e., } \alpha_i = C - \mu_i \qquad \forall i. \tag{15}$$

By substituting $C - \mu_i$ for α_i in Equations (14), the Lagrangian dual form of the problem becomes:

$$(\omega^*, \mu^*) = \arg\max_{\mu} \inf_{\omega} \mathcal{L} = \frac{1}{2}\|\omega\|^2 + \sum_{i=1}^{n} \mu_i \left(1 - \omega.\lambda_i\right) \tag{16a}$$

subject to

$$\mu_i \geq 0 \qquad \forall i, \tag{16b}$$

$$\alpha_i \geq 0 \quad \text{i.e., } C - \mu_i \geq 0 \qquad \forall i. \tag{16c}$$

As in the previous section, replacing ω with $\sum_{i=1}^{n} \mu_i \lambda_i$ gives us the simpler form of the problem:

$$\mu^* = \arg\max_{\mu_i} \mathcal{L} = \sum_{i=1}^{n} \mu_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \mu_i \mu_j \lambda_i \cdot \lambda_j \tag{17a}$$

subject to

$$0 \leq \mu_i \leq C \qquad \forall i. \tag{17b}$$

Since neither the ξ_i nor its associated Lagrange multiplier (α_i) appear in the objective function, this format of the problem is very similar to the one without introducing costs (Eq. 12), except μ_i is constrained by the upper bound value of C.

3.3 Non-Linear Scoring Function

Remember that we assumed that the scoring function (with its associated hyperplane) is a linear function. Conversely, a non-linear scoring function might be a better choice in some real-world applications. To deal with this matter, in this section, the model is developed further to cover the non-linear case as well.

The *Kernel function* concept (Aizerman et al., 1964) is a widely-used trick for pattern recognition problems which can be also exploited for our case. The idea comes from this fact that a set of non linearly-representable data could be linearly managed if they mapped to a higher dimension. To do the mapping, we assume a function Φ of the form:

$$\Phi : \mathbb{R}^d \rightarrow \mathcal{H}, \tag{18}$$

where $\mathcal{H}$ denotes a higher dimensional space than d-dimension. If all preferences points are mapped into $\mathcal{H}$ by making use of Φ, the Lagrangian function will be:

$$\mathcal{L} = \sum_{i=1}^{n} \mu_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \mu_i \mu_j \Phi(\lambda_i) \cdot \Phi(\lambda_j). \tag{19}$$

A deeper look into the process of computing $\mathcal{L}$ (Eq. 19) reveals that, even though $\mathcal{L}$ is still a scalar value, the optimization problem performs a dot product operation ($\Phi(\lambda_i) \cdot \Phi(\lambda_j)$) in the high dimensional space which is computationally expensive.

Principally, a kernel is a function which operates in the lower dimension ($K(X,Y) : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$), but yields an identical result to the dot product of mapped vectors in the higher dimension ($K(X,Y) = \Phi(X) \cdot \Phi(Y)$).

Due to the above property of the kernel function, the term $\Phi(\lambda_i) \cdot \Phi(\lambda_j)$ can be simply replaced in Eq. (19) with an appropriate kernel. The great advantage of such a replacement is that the complexity of the optimization problem remains only dependent on the dimensionality of d (the preferences points space) and not of $\mathcal{H}$.

Note that this simplification happens without even explicitly stating the Φ function because the problem has been formulated in terms of dot product of points, the property that we insisted to have in the previous sections. So, the problem is rewritten as follows:

$$\mu^* = \arg\max_{\mu} \mathcal{L} = \sum_{i=1}^{n} \mu_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \mu_i \mu_j K(\lambda_i, \lambda_j) \tag{20a}$$

subject to

$$0 \leq \mu_i \leq C \qquad \forall i \tag{20b}$$

There still remains one point that should be considered. Solving this optimization problem only gives μ^* where after all, ω^* is required. The Eq. (11) cannot be used to attain ω anymore, because after the mapping, ω will live in $\mathcal{H}$; that is:

$$\omega = \sum_{i=1}^{n} \mu_i \Phi(\lambda_\mathbf{i}) \tag{21}$$

However, recall that finding ω^* is just an intermediate goal to achieve the scoring function. Therefore, referring back to the scoring function in Equation (2) and using Equation (21) for eliminating ω, brings the following form of the scoring function:

$$\begin{aligned} f(\mathbf{X}) &= \omega \cdot \Phi(\mathbf{X}) \\ &= (\sum_{i=1}^{n} \mu_i \Phi(\lambda_\mathbf{i})) \cdot \Phi(\mathbf{X}) \\ &= \sum_{i=1}^{n} \mu_i (\Phi(\lambda_\mathbf{i}) \cdot \Phi(\mathbf{X})) \end{aligned}$$

Importantly, this result contains points only in the format of dot products; where incorporating a kernel function gives:

$$f(\mathbf{X}) = \sum_{i=1}^{n} \mu_i \, K(\lambda_\mathbf{i}, \mathbf{X}) \tag{22}$$

Equations (20) and (22) form the method, which is referred to as *SVPL* (Support Vector Preferences Learner) in our experiments. As seen, the input parameters should be chosen in SVPL are C and the kernel function's parameters if it has any.

4 EXPERIMENTS

4.1 Data Repository

The experiments make use of a subset of a year's worth of real ridesharing records (from Apr 2014 to Apr 2015), provided by Carma[1] (formerly Avego). Carma is a software company currently offering a dynamic ride-share application for internet-enabled mobile phones. In addition to their App, any individual can create his own App, powered by the Carma's free API which uses Carma's live data.

The process of Carma's application is briefly explained so as to facilitate the understanding of the structure of the provided database. Users can request a ridesharing as a driver, rider or both (flexible to be rider or driver). Once a ridesharing request is created, the user could ask the system to find appropriate matching opportunities. Then, a suggestion list is shown on the user's screen. For each suggested

[1] https://carmacarpool.com/

item, the user can accept, reject or just leave it with no response. Note that accepting an opportunity at this stage does not mean the ridesharing will certainly happen, because it also needs the confirmation of the other party; so, it is observed in the database that for a single trip a user may have several accepted items. While the system is waiting to get the second confirmation, the user can cancel an initially accepted item. It should be pointed out that the cancellation of a specific item differs from the cancellation of the whole trip request; the former often happens because of the greater attractiveness of a new item, and the latter is for the time when the user does not want to share the ride for the created trip anymore.

According to the above mechanism, any ride-matching record could be associated with a class label which is among these four: *Accepted*, *Cancelled*, *Ignored* or *Rejected*. As explained earlier, a class label comes from the response of the *main* user towards sharing the ride with the *target* user.

The second element of a ride-matching record is a vector of features, built from the personal information of both the main and the target user and the properties of their trips. Before starting the learning process, normalizing the features' spaces is an essential requirement due to the fact that margin based methods are known to be sensitive to the way features are scaled (Ben-Hur and Weston, 2010). Because of that, features are scaled to an identical range, i.e., $[0,1]$. The extracted features from the provided database are listed here:

- Positional Component: Expressing how much the pick-up and drop-off locations will be suitable for the main user.
- Temporal Component: Expressing how much the departure time will be appropriate for the main user.
- Gender Component: Indicating whether the target user is of the same gender (1) or the opposite (0).
- Has Image: it considers whether the target user's profile contains his/her picture or not.
- Average Rating: At the end of a ridesharing experience, users can rate the other party. This feature holds the average rate that the target user has gained from previous trips.
- Is Favourite: As well as the rating, a user can mark a particular user as a favourite person at the end of ridesharing. This feature shows whether the target user is among individuals who are marked as a favourite of the main user or not.
- Previous Rating: If the main user and the target user have had a previous ridesharing experience with each other, this feature shows the rating that the main user has given to the target user.

Considering the fact that a separate model will be individually learned for each user, and with the purpose of having an appropriate number of records, the experiments are run for those users who have at least 50 records. In the provided data set, 44 users met this condition, where the best user in terms of number of records had 402 records. The total number of records were 5676.

4.2 Results

Besides SVPL, using a Regression based method is another viable approach for learning weights, as explained in Section 2.1. Thus, in this section, the performance of SVPL is compared with a Regression based method as a competitor.

These two methods need different formats for the input data, though they both eventually produce a scoring function. For the regression method, the main user responses (class labels) are converted to scalar values, and for SVPL, a set of preferences among ride-matching records is created (Section 2.1 illustrates how).

Along the wide spectrum of regression methods, the SVM regression method is chosen. This choice serves two purposes. First, since both rival methods have a similar foundation in their learning mechanism, the merit of using the preferences set rather than scalar values could be assessed more precisely. Secondly, the same input parameters (e.g., cost error, the kernel's parameters etc.) have been used to tune models in order to have a fair comparison.

In terms of choosing the kernel function, although there are many kernels available in the literature, and devising a new kernel by meeting *Mercer's Condition* is possible, we just simply utilize three well-known kernels, listed here:

- $K(\mathbf{x},\mathbf{y}) = \mathbf{x}\cdot\mathbf{y}$: The linear kernel which is equivalent to the non-kernel based formulation (Eq. 17).
- $K(\mathbf{x},\mathbf{y}) = (\gamma\mathbf{x}\cdot\mathbf{y}+r)^p$: The polynomial kernel of degree p; here, we assume $p=2$, $\gamma=1$ and $r=0$.
- $K(\mathbf{x},\mathbf{y}) = e^{-\|\mathbf{x}-\mathbf{y}\|^2/2\sigma^2}$: This is the Gaussian radial basis function (RBF), a typical choice for the kernel function. The parameter σ is tuned as an input parameter.

For each user, the ride-matching records are sorted in ascending order of the creation time, and then split into two parts. The first part includes 80% of the records and is used to learn the model. At the end of learning stage, models can predict, for each record,

Table 2: Five ride-matching records which are used in Example 2.

	Ground Truth		SVPL Prediction		Regression Prediction	
	Labels	Ranking	Value	Ranking	Value	Ranking
S_1	C	2	0.52	2	0.71	3
S_2	I	3	0.35	4	0.28	5
S_3	I	3	0.31	5	0.36	4
S_4	A	1	0.77	1	0.91	1
S_5	R	4	0.43	3	0.75	2

a scalar value which expresses the goodness of that record for the main user.

The second part of the data is used for the testing stage. From this data, a weak order (a kind of ranking in which incomparable items are assigned with the same rank) with respect to class labels could be derived. This ranking is used as the ground truth for testing. From the other side, the predicted scalar values for records produced by learned models suggest a total order (ordinal ranking) between records. Thereafter, *C-Index* (or concordance *C*) as one of standard ranking assessment measures, is exploited so as to find the accuracy of models (Fürnkranz and Hüllermeier, 2010). It is is calculated as follows:

$$\text{C-Index}(r,\hat{r}) = \frac{\kappa}{\kappa+\bar{\kappa}}$$

where κ is the number of correctly ranked pairs and $\bar{\kappa}$ (Kendall tau distance) is the number of incorrectly ranked pairs. This is illustrated by the following example.

Example 2. Assume there are 5 ride-matching records to test the learned models. Table 2 includes the user's responses to these records, derived partial ranking as the ground truth and predicted scalar values/ranking by using SVPL and Regression. According to the given information, there are two pairs that are incorrectly ranked for SVPL ((S_2,S_5) and (S_3,S_5)). So, the C-Index for SVPL is $\frac{7}{9} = 0.77$. For the Regression method, Kendall tau distance is 3 due to three wrong ranked pairs ((S_1,S_5), (S_2,S_5) and (S_3,S_5)). Hence, the C-Index is $\frac{6}{9} = 0.66$. Note that S_1 and S_3 are incomparable here.

In order to adjust the input parameters, we use the *Grid Search* algorithm for hyper-parameter optimization (Bergstra and Bengio, 2012). Roughly speaking, in the Grid Search, input parameters are tuned for each rival method individually to yield the best performance for that particular method. So, the accuracy of models are compared at their optimal point.

Each run of the learning and testing phases for all those 44 users takes less than a couple of minutes, making use of a computer facilitated by a Core i7 2.60 GHz processor and 8 GB RAM memory. In our experiments, simply averaging the accuracy of a model among users is used to compute the overall accuracy of the model in each run. Figure 2 depicts the accuracy of models for those three aforementioned kernels.

As seen in the figure, SVPL outperforms the regression method overall where the difference between two methods is most notable in the linear case. The accuracy of Regression increases by utilizing the RBF kernel, while it is not the case for SVPL. We should note that kernel methods are supposed to ease representation of points in the features space whilst incurring computational expense; that could mean if a model has almost the same accuracy in a non-kernel and kernel based formulation, the non-kernel one is preferred.

5 CONCLUDING REMARKS

In this paper, we have presented novel aspects of the potential automatic ride-matching system, making use of user preferences. Unlike the prevalent systems in ridesharing, this approach gives freedom to participants in the choice of ridesharing partner; it is supported by the fact that the user may disapprove of the assigned matching which is found from the optimization of the whole system. A model similar to SVMRank(Joachims, 2002) in form, SVPL, was discussed in Section 3 with the ability of learning user preferences in a natural way in order to finally contribute in the suggestion of a menu of good choices to the user. Moreover, despite providing the flexibility in the matching process to deviate from the optimal solution, learning user preferences softens the traditional way of setting hard constraints, and remove the need for eliciting some complex parameters from participants.

A natural extension would be if the preferences set (pairwise trade-offs) derived from the user's feedback could be associated with some degree of uncertainty. For instance, if it is said that the user proba-

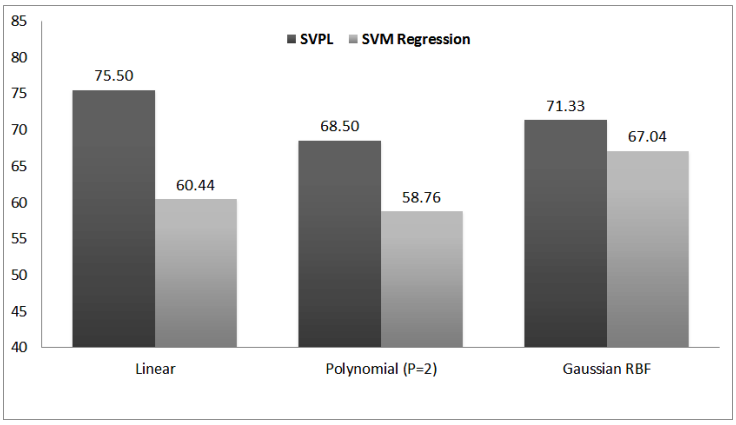

Figure 2: The accuracy of models when linear formulation, polynomial kernel and gaussian kernel are chosen.

bly prefers the case A to B, with certainty degree of 0.7. We plan to study the uncertainty in preferences in our future works. Other interesting subjects related to this paper that could be addressed in future works, include handling nominal and unknown values in the features space, online learning, and utilizing a general model for users who still do not have enough records for learning a dedicated model.

6 ACKNOWLEDGMENT

This publication has emanated from research supported in part by a research grant from Science Foundation Ireland (SFI) under Grant Number SFI/12/RC/2289, and also supported by Carma (http://carmacarpool.com).

REFERENCES

Agatz, N., Erera, A., Savelsbergh, M., and Wang, X. (2012). Optimization for dynamic ride-sharing: A review. *European Journal of Operational Research*, 223(2):295 – 303.

Agatz, N. A., Erera, A. L., Savelsbergh, M. W., and Wang, X. (2011). Dynamic ride-sharing: A simulation study in metro atlanta. *Transportation Research Part B: Methodological*, 45(9):1450–1464.

Aiolli, F. and Sperduti, A. (2011). A preference optimization based unifying framework for supervised learning problems. In Fürnkranz, J. and Hüllermeier, E., editors, *Preference Learning*, pages 19–42. Springer Berlin Heidelberg.

Aizerman, A., Braverman, E. M., and Rozoner, L. I. (1964). Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control*, 25:821–837.

Amey, A., Attanucci, J., and Mishalani, R. (2011). Real-time ridesharing the opportunities and challenges of utilizing mobile phone technology to improve rideshare services. *Transportation Research Record: Journal of the Transportation Research Board*, 2217(1):103–110.

Armant, V. and Brown, K. N. (2014). Minimizing the driving distance in ride sharing systems. In *Tools with Artificial Intelligence (ICTAI), 2014 IEEE 26th International Conference on*, pages 568–575. IEEE.

Baldacci, R., Maniezzo, V., and Mingozzi, A. (2004). An exact method for the car pooling problem based on lagrangean column generation. *Operations Research*, 52(3):422–439.

Ben-Hur, A. and Weston, J. (2010). A users guide to support vector machines. In Carugo, O. and Eisenhaber, F., editors, *Data Mining Techniques for the Life Sciences*, volume 609 of *Methods in Molecular Biology*, pages 223–239. Humana Press.

Bergstra, J. and Bengio, Y. (2012). Random search for hyper-parameter optimization. *The Journal of Machine Learning Research*, 13(1):281–305.

Boyd, S. and Vandenberghe, L. (2004). *Convex optimization*. Cambridge university press.

Burges, C. J. (1998). A tutorial on support vector machines for pattern recognition. *Data mining and knowledge discovery*, 2(2):121–167.

Chan, N. D. and Shaheen, S. A. (2012). Ridesharing in north america: Past, present, and future. *Transport Reviews*, 32(1):93–112.

Chaube, V., Kavanaugh, A. L., and Perez-Quinones, M. A. (2010). Leveraging social networks to embed trust in rideshare programs. In *System Sci-*

ences (HICSS), 2010 43rd Hawaii International Conference on, pages 1–8. IEEE.

Duggan, M. and Smith, A. (2013). Cell internet use 2013. Pew Research Center, http://www.pewinternet.org/2013/09/16/cell-internet-use-2013/.

Emarketer (2014). 2 billion consumers worldwide to get smart(phones) by 2016. http://www.emarketer.com/Article/2-Billion-Consumers-Worldwide-Smartphones-by-2016/1011694.

Fürnkranz, J. and Hüllermeier, E. (2010). *Preference learning*. Springer.

Furuhata, M., Dessouky, M., Ordóñez, F., Brunet, M.-E., Wang, X., and Koenig, S. (2013). Ridesharing: The state-of-the-art and future directions. *Transportation Research Part B: Methodological*, 57:28–46.

Ghoseiri, K., Haghani, A. E., and Hamedi, M. (2011). *Real-time rideshare matching problem*.

Herbawi, W. M. and Weber, M. (2012). A genetic and insertion heuristic algorithm for solving the dynamic ridematching problem with time windows. In *Proceedings of the 14th annual conference on Genetic and evolutionary computation*, pages 385–392. ACM.

Hwang, M., Kemp, J., Lerner-Lam, E., Neuerburg, N., and Okunieff, P. (2006). Advanced public transportation systems: state of the art update 2006. Technical report.

Joachims, T. (2002). Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 133–142. ACM.

Saranow, J. (2006). Carpooling for grown-ups—high gas prices, new services give ride-sharing a boost; rating your fellow rider. *Wall Street Journal*.

Simonin, G. and O'Sullivan, B. (2014). Optimisation for the ride-sharing problem: a complexity-based approach. In *ECAI*, pages 831–836.

Smith, A. (2015). Us smartphone use in 2015. Pew Research Center, http://www.pewinternet.org/2015/04/01/us-smartphone-use-in-2015/.

Zickuhr, K. (2012). Three-quarters of smartphone owners use location-based services. Pew Research Center, http://www.pewinternet.org/2012/05/11/three-quarters-of-smartphone-owners-use-location-based-services/.