Data Driven VRP: A Neural Network Model to Learn Hidden Preferences for VRP

Jayanta Mandi
Data Analytics Laboratory, Vrije Universiteit Brussel, Belgium

Rocsildes Canoy
Data Analytics Laboratory, Vrije Universiteit Brussel, Belgium

Víctor Bucarey
Institute of Engineering Sciences, Universidad de O'Higgins, Rancagua, Chile

Tias Guns
Data Analytics Laboratory, Vrije Universiteit Brussel, Belgium
Department of Computer Science, KU Leuven, Belgium

Abstract

The traditional Capacitated Vehicle Routing Problem (CVRP) minimizes the total distance of the routes under the capacity constraints of the vehicles. But more often, the objective involves multiple criteria including not only the total distance of the tour but also other factors such as travel costs, travel time, and fuel consumption. Moreover, in reality, there are numerous implicit preferences ingrained in the minds of the route planners and the drivers. Drivers, for instance, have familiarity with certain neighborhoods and knowledge of the state of roads, and often consider the best places for rest and lunch breaks. This knowledge is difficult to formulate and balance when operational routing decisions have to be made.

This motivates us to learn the implicit preferences from past solutions and to incorporate these learned preferences in the optimization process. These preferences are in the form of arc probabilities, i.e., the more preferred a route is, the higher is the joint probability. The novelty of this work is the use of a neural network model to estimate the arc probabilities, which allows for additional features and automatic parameter estimation. This first requires identifying suitable features, neural architectures and loss functions, taking into account that there is typically few data available. We investigate the difference with a prior weighted Markov counting approach, and study the applicability of neural networks in this setting.

2012 ACM Subject Classification Computing methodologies → Planning and scheduling

Keywords and phrases Vehicle routing, Neural network, Preference learning

Digital Object Identifier 10.4230/LIPIcs.CP.2021.42

Supplementary Material *Software (Source Code and Anonymized Data)*: `https://github.com/JayMan91/CP2021-Data-Driven-VRP`

Funding Research supported by the FWO Flanders project Data-driven logistics (FWO-S007318N).

1 Introduction

Although the Vehicle Routing Problem (VRP) and its many variants have been extensively studied in the literature, the "theoretical optimal" solution often does not meet the expectations of the route planners and the drivers. This is because in real-life operations, the acceptability of a route is dependent not only on distance, travel time or fuel consumption, which have been studied in the literature, but also on multiple factors which are difficult to put in the objective function. A study by [2] has revealed that local drivers prefer routes that are not optimal in terms of travel time or cost. The drivers take into account several factors which are not in the objective function such as traffic congestion and availability of parking and fuel stations. This highlights the necessity of *preference-based* routing where the objective is to minimize the travel cost as perceived by the drivers and the planners. To put it another way, we can see it as maximizing the utility of the drivers and the planners.

In this work, we propose VRP solutions which are acceptable to the route planners and the drivers. We start from the setting studied in [1], which proposes the maximum likelihood routing. Maximum likelihood routing considers the transition probabilities between the stops as *revealed preferences* of the drivers and the planners and finds the maximum utility route by maximizing the joint transition probabilities. To estimate the transition probabilities, [1] uses a Markov counting approach which enumerates the past solutions for each realized route. Their approach uses only the past solutions for probabilities estimation but cannot make use of contextual features such as day of week. We extend their framework to use a neural network model to estimate the transition probabilities before finding the route by applying maximum likelihood routing. The motivation behind using the neural network model is to generate a better estimation of the cost vector by using historical as well as contextual information in the neural network model.

We start with a neural network model which is trained using both contextual information and past solutions. We also include the Markov prediction as a feature in the neural network and observe improvement in the solution quality. Finally, we choose a parsimonious architecture in order to avoid overfitting and with this we are able to outperform [1].

Contributions.

- We formulate the challenge of neural network-based learning of hidden preferences from moderately sized data, in a way that is compatible with existing VRP solvers.
- We investigate different features and architectures for such a neural network, more specifically arc-based linear models combined into per-node probabilistic estimates.
- We investigate how we can combine the Markov model and neural network, e.g., by considering the Markov predictions as an input to the neural network model.
- We propose two loss functions that allow for gradient-descent learning: one based on standard multi-class losses and another based on decision-focused learning that incorporate the VRP solving into the loss function.

2 Related Work

The VRP [3] has been studied with its many different variations. Traditional VRP minimizes a tangible objective such as operational costs [10], travel time [13], fuel consumption or carbon emission [26, 19]. Although multiple aspects of the assignment schedules of the drivers such as route balancing [14] have been studied, learning and optimizing drivers' preferences has recently received increasing attention.

The preferences of the drivers can be considered by including them in the objective function. This can be treated in a multi-objective VRP [11] setting, such as forming an objective function as a weighted sum or finding the set of Pareto optimal solutions based on standard multi-objective evolutionary algorithms [21]. However, the preferences of the drivers are implicit [22] and in most cases, explicit formalization of these preferences is not possible in practice.

Authors in [1] tackled the problem from a different perspective: they introduce a weighted Markov model to learn the preferences. This approach avoids the explicit specification of the preference constraints and the implicit sub-objectives. The Markovian model is built using preferences learned from past solutions, which the planners have constructed by modifying solutions given by off-the-self solvers. Contrary to their work, we use a neural network model to learn the drivers' preferences, allowing a more flexible and general framework.

Learning preferences for drivers have been focused mainly in the setting of one single origin and destination. TRIP [15] leverages past GPS data to learn drivers' preferences by comparing the ratios of the drivers' travel time to the average travel time, and with that, it generates routes that mimic the ones chosen by the drivers. The approach in [6] also deduces driving preferences from GPS traces and models them into the weights of a linear programming formulation, which is then optimized to generate new route suggestions. The authors in [8] are able to enhance the quality of the solutions by considering different routing preferences that vary depending on the contexts. While we do not provide an explicit representation of the preferences, we assume that the preferences can be expressed in terms of probabilities (or utilities) of the arcs in the graph.

Decision-focused learning [25, 5], which combines gradient-based neural network and optimization into a single framework, has recently received much attention in operations research. In this setup, the outputs of the neural network are fed into the optimization module as one of the inputs. The novelty of this approach is that it trains the neural network model while considering the objective value in the optimization problem. Decision-focused learning of submodular optimization problems, zero-sum games and SAT problems have been studied in [25, 17, 24] respectively. The approach proposed by [20] also combines a neural network model with any given optimization oracle via "implicit interpolation". Ours is the first work which uses this framework for learning preferences in the context of vehicle routing.

3 Preliminaries

3.1 Problem Description

In this work, we are interested in the route planning process of an actual small transportation company. The route planners in the company are responsible for organizing tours for a fleet of vehicles in order to deliver goods to the customers. Although they use a commercial route optimization software to produce routes that are optimal in terms of route length and travel time, they are hardly satisfied with the solutions. Solutions have to be modified to come up with a tour which is acceptable to the planners, drivers, and other stakeholders. In this way, the planners are implicitly optimizing the utilities of all those involved.

One way to approach this problem is to explicitly define the set of objectives. However, it is nearly impossible to model such personal preferences. We observe that the planners start from past realized routes because they require minimal modifications compared to the "theoretical optimal" routes. Therefore, in a way, the past solutions capture the preferences (or the utilities) of the planners and the drivers. Our objective in this work is to learn the latent preferences of the drivers and the route planners for vehicle routing using a neural network model and propose tours which are acceptable to the planners. More specifically, we will focus on learning preferences at the arc level. We consider the transition probabilities as *revealed preferences*. We use neural network to output the transition probabilities between every pair of nodes. The advantage of this formulation is that we can use the negative log probability in place of a traditional travel cost in any existing VRP solver.

Challenges. A machine learning model learns from training instances. In our case with the company data, each instance is realized in a day. However, due to functional and operational reasons, tours are not organized each day. Putting that into perspective, it would take more than 6 months to collect only 180 training instances. Consequently, we are not in a state to use a neural network model trained with thousands of training instances. Hence, we have to be particularly careful about using a neural network model with a large number of parameters, as such network is prone to overfitting with small data.

Another challenge in this case is that not all customers raise a demand request with equal frequency. In fact, some customers have daily requests, whereas others raise requests only once or twice in a month. Depending on which set of customers raise a request, there can be considerable changes in the tour. We also observe a weekly pattern in the tours, i.e., tours of one weekday are different from the other days but very similar to those from the same weekday of the previous weeks. Therefore, learning the weekly patterns from a limited number of weeks poses another challenge.

3.2 Formalization

We begin by formalizing the objective and the data structures. Formally, on a given day t, S^t is the set of stops to be served by a number m^t of homogeneous vehicles. We represent $S^t \doteq \{0, 1, \ldots, n\}$, where 0 represents the depot, and the other nodes represent the customers. Let A^t define the set of all arcs in S^t.

We call $\mathbf{x}^t$ a **routing** with respect to S^t with m^t homogeneous vehicles, if $\mathbf{x}^t$ contains a set of at most m^t tours in S^t with each tour starting from and ending at the depot 0 and each node in S^t is visited exactly once to satisfy its demand request. Additionally, a feasible routing should ensure that the total demand allocated to each vehicle does not exceed its capacity Q. Let $\mathcal{X}_{S^t}^{m^t}$ denote all feasible routings of m^t vehicles over S^t. The objective in standard CVRP is to minimize the total travel costs of the routing. We remark that the depot is fixed and always present but the set of stops S^t changes from one day to another as not all customers raise a demand request each day.

For learning the preferences from past data, we are given a dataset $\mathcal{H} = \{(S^t, z^t, \mathbf{X}^t)\}_{t=1}^T$. Each instance in the dataset is a tuple where t is a timestamp, S^t is the set of stops served at t, $\mathbf{X}^t$ denotes the actual preferred routing created by the planners, and z^t are feature variables such as the demand of each stop, the number of vehicles used, the day of the week, or some other known parameters. Hereafter, we will use the symbols without the suffix t to avoid notational complexity.

3.3 Transition Probabilities

Explicit specification of the preferences of the drivers and the planners would result in a complex model with a large number of parameters to tune. Instead, in this paper, we use the framework of [1], which captures the preferences of the route planners and the drivers using transition probabilities. In more formal terms, we learn a model which assigns probabilities to all the arcs within the network. Our hope is that these transition probabilities subsume the hidden preferences of the route planners and the drivers. Formally, we learn $\Pr(r|s)$ which denotes the probability of the next stop being r, conditional on the current stop s. We remark that the transition probability would be a function of some temporal and contextual attributes including but not limited to the traditional cost measures.

3.4 Maximum Likelihood Routing

Once the probabilities are learned, we follow the methodology of [1] to find the most *likely* routing from the set of all feasible routings. We call the routing with the highest probability the maximum likelihood routing (MLE routing). Formally,

$$\max_{x \in \mathcal{X}_S^m} \prod_{(s \to r) \in x} \Pr(r|s). \tag{1}$$

In order to identify the MLE routing, we solve an optimization problem whose feasible region is defined by the following standard CVRP constraints [23].

$$\sum_{r \in V,\, r \neq s} x_{sr} = 1 \qquad s \in S \tag{2}$$

$$\sum_{s \in V,\, s \neq r} x_{sr} = 1 \qquad r \in S \tag{3}$$

$$\sum_{r=1}^{n} x_{0r} = m \tag{4}$$

$$\text{if } x_{sr} = 1 \;\Rightarrow\; u_s + q_r = u_r \qquad (s,r) \in A : t \neq 0,\ s \neq 0 \tag{5}$$

$$q_s \leq u_s \leq Q \qquad s \in S \backslash \{0\} \tag{6}$$

$$x_{sr} \in \{0,1\} \qquad (s,r) \in A. \tag{7}$$

(2) and (3) ensure that each customer is served by exactly one vehicle. (5) performs subtour elimination. (6) ensures that the vehicle capacity is respected. We remark that in (4), we use the equality constraint because in practice, the company must use all the available vehicles. The only modification from the standard CVRP is that instead of minimizing the distance, we maximize the joint probability. To transform the product in the objective function into a sum, we consider log probabilities in the objective function and minimize the following:

$$\min_{x} \sum_{(s,r) \in A} -log\, \Pr(r|s) x_{sr} \tag{8}$$

In the subsequent discussions, the (s,r)-th entry of matrix P would contain $\Pr(r|s)$.

3.5 Transition Probability Estimation by Markov Counting

The goal of the Markov Counting approach is to estimate all the conditional probabilities $\Pr(r|s)$ over the set of all stops in the data: $S^{\text{all}} = \bigcup_t S^t$. From conditional probability theory, we have:

$$\Pr(r|s) = \frac{\Pr(s \to r)}{\Pr(s)}, \tag{9}$$

where $\Pr(s) = \sum_u \Pr(s \to u)$. By defining the frequency of a transition $(s \to r)$ in the historical dataset $\mathcal{H}$ as $f_{sr} = \sum_t [\![(s \to r \in \mathbf{X}^t)]\!]$, where $[\![\cdot]\!]$ equals 1 if the statement inside the bracket is true and 0 otherwise, the conditional probabilities from the dataset can be estimated by:

$$\Pr(r|s) = \frac{f_{sr}}{\sum_u f_{su}}. \tag{10}$$

We point out that with this formulation, we can solve the standard CVRP which minimizes the distance if we replace $\Pr(r|s)$ by a *distance-based probability* $\Pr_{dist}(r|s)$:

$$\Pr_{dist}(r|s) = \frac{e^{-d_{sr}}}{\sum_u e^{-d_{su}}}. \tag{11}$$

The transition probability matrix construction algorithm presented in [1] makes use of *weighing schemes,* where a variable weight w_t is defined for each historical instance in $\mathcal{H}$. This weight varies according to the properties of the tuple $(S^t, z^t, \mathbf{X}^t)$. Giving varying weights to each historical instance affects the way the transition frequencies are counted, hence each weighing scheme results in a different transition matrix. Exponential weighing is one of the most used scheme, where instances far-off in the past receive decaying weights. Therefore, in our experiments, we will compare our approach with the Markov model with the exponential weighing scheme.

4 Learning the Transition Probabilities Using Neural Network

One limitation of the Markov counting approach introduced in section 3.5 is that it only uses past data to arrive at the probabilities. We want the transition probabilities to be a function of other attributes such as the day of week and the distances between stops, among others. This is the motivation behind using a neural network model.

Neural networks are made of interconnected units called neurons. A single neuron takes a series of inputs $z_0, ..., z_n$ and returns an output o as a function of the inputs $o = f(\sum w_i z_i)$, or in matrix form $o = f(Wz)$, where f is an activation function and w_i's are the weights. Many choices for the activation function exist – *sigmoid*, *ReLU*, *tanh* are some of the widely used activation functions. A network consists of several layers, and multiple neurons are stacked in each layer where the inputs are connected to each neuron. The output of the layer can be conveniently described in matrix form as $o = f(Wz)$. Here, each row of W corresponds to each neuron. The dimension of output o is controlled by the dimension of matrix W. In a multilayer network, the subsequent layers use the outputs of the preceding layers as inputs. Obviously, the designer has the option to transform the output between two layers.

A multilayer neural network is considered as a universal function approximator [16], which tries to learn the functional relationship between the output and the input. To do so, the parameters of the neural network must be learned using training data. This is done by backpropagating the loss between the predicted output and the target output. During backpropagation, the derivative of the final loss with respect to the weights is computed and then the weights are updated by gradient descent. The choice of the loss function is dependent on the problem at hand. For a multiclass classification approach, categorical cross entropy loss is the preferred choice.

We propose to learn the transition probabilities between the stops from the historical data using a neural network. For a single day t, we have $(S^t, z^t, \mathbf{X}^t)$ as explained in section 3.5.

Feature variables. We want the predicted probabilities to be a function of the feature variables. Different types of features can be considered: time-lagged temporal features, features related to the set of stops to be served (S^t), the distance between the stops and contextual features such as day of week, number of vehicles. The motivation behind using the time-lagged solutions as features is to learn from past solutions. We define the *look back period* (L) as the maximum number of past observations considered in our model. The motivation of this look back period is two-fold: 1. it allows us to model the fact that past observations lose their relevance over time, and 2. to avoid problems of over-fitting due to lack of observations. We can also consider the output of the Markov counting model as a feature, as it subsumes past information. Moreover, this can be computed easily on the fly.

From this discussion, it is evident that some of the features are specific to an arc. This includes the time-lagged features, the distances and the Markov probabilities. On the other hand, features such as day of week and stops to be served are the same across all the stops. All of these are considered as an input to the neural network in Figure 1.

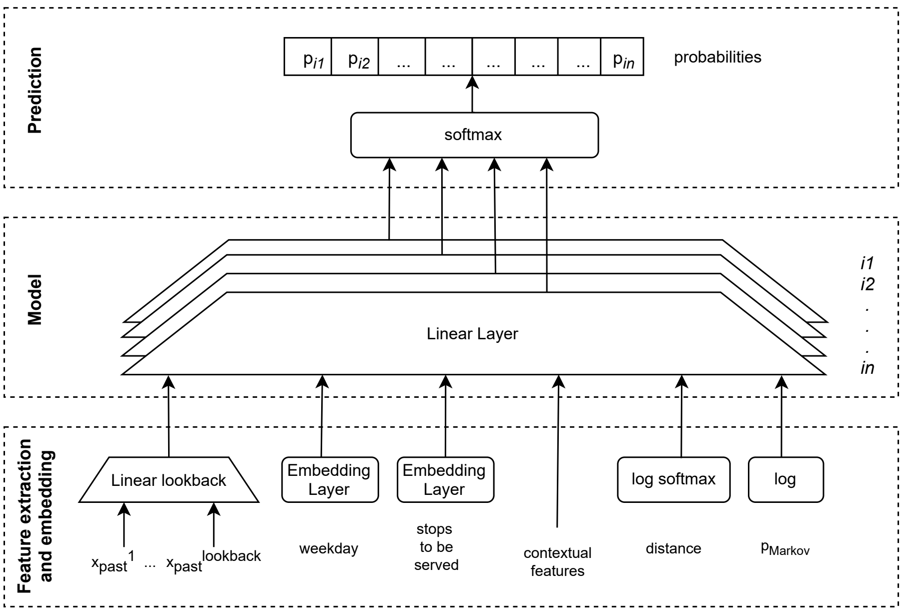

Figure 1 Neural network architecture to estimate transition probbailities from a source node s.

Architecture of the neural network. Our goal is to build a network that estimates the transition probabilities for each arc $(s, r) \in A$. We have $|A| \times |A|$ distance features, $L \times |A| \times |A|$ time lagged features and $|K|$ number of contextual features. Because of limited data, our aim is to build a parsimonious network with as few parameters as possible.

Authors in [1] propose a linear combination of Markov probabilities (Eq. 10) and distance-based probabilities (Eq. 11), $P^t = \omega P^t_{Markov} + (1 - \omega) P^t_{dist}$. Essentially, this approach considers these two factors while computing the probabilities. Furthermore, this can be extended to more number of feature variables in general. The advantage of using a neural network is that it learns the weights of the linear combination itself.

The final layer of the proposed architecture in Figure 1 performs this linear combination. This linear layer outputs unnormalized scores and a softmax operation upon these would result in the probabilities. As the outputs are unnormalized scores, so should be the inputs and that is why we log-transform the Markov probabilities. The distance based probabilities are arrived by considering the softmax of the distances from the source stop s.

We treat the categorical features such as day of week and stops to be served, by passing them to embedding layers before feeding them to the linear layer. The embedding layer computes dense vector representations of the categorical variables. The numerical feature variables such as the number of vehicles are passed directly to the final layer.

We treat the time-lagged solutions between (s, r) by considering linear combinations of L previous solutions of (s, r). The first linear layer considers past solutions of the look back period as inputs and its output is a score based on that. Thus, the stop which has been chosen more often as the next stop, would be assigned a higher score. We remark that there are other ways to treat time-lagged variables such as LSTM [7], but they might be prone to overfitting because they have a large number of parameters.

Algorithm 1 Transition probability estimation from source stop s.

```
Input :
  - Historical solutions up to day T: $\{(S^t, z^t, \mathbf{X}^t)\}_{t=1}^T$
  - $s$: the source stop
active days ⟵ list of days where $s$ is active
Model.initialize();
for $t$ in active days do
    $x^t_{(s)} \leftarrow \mathbf{X}^t[s,:]$ // target solution
    $Past^t_{(s)} \leftarrow \{\mathbf{X}^{t'}[s,:] | t' \in$ active days[-(look back period):] $\}$
    if length(active days) < look back period then
        Fill the remaining days with equiprobable probabilities vector
    end
    $\mathcal{H}^t \leftarrow \left(z^t_{(s)}, Past^t_{(s)}\right)$
    for training epochs do
        $\hat{P}^t[s,:] \leftarrow$ Model.Predict($\mathcal{H}^t$)
        $L \leftarrow$ Cross Entropy($\hat{P}^t[s,:], x^t_{(s)}$)
        Update Model by backpropagating $\nabla_{\hat{P}^t}(L)$
    end
end
```

The final linear layer outputs a score between s and a destination node. So essentially, there are $|A|$ number of such linear layers. The use of neural network gives us the flexibility to use separate weights for the linear layers of every destination node. Obviously, this may result in overfitting as the number of parameters increase.

Other than the contextual features, which are the same for all (s, r), the other inputs to the linear layer are specific to (s, r) and so is the output. We use a **separate model for each source node** s and each of them generates transition probabilities from the source node to all the other nodes.

Algorithm. Algorithm 1 proposes a training scheme for estimating probabilities from a single source stop s to the others. We use all the features including the time-lagged solutions day t, to estimate the transition probabilities on day t. This past data ($\text{Past}^t_{(s)}$) is obtained by extracting the corresponding row from the incidence matrix. As the training is for s, we only consider the next stop visited after it in the past. Hence, we formulate it as a multiclass classification problem, where the classes are the possible next stops and the objective is to classify them correctly. We use different models for different stops and while training the model for a stop s, we only consider past days where s was served. For any stop that does not have enough past data, to fill the look back period, the **remainder of the look back period are filled with uniform distribution** over the set of possible next stops.

Loss function. We formulate the learning problem as a multiclass classification task. The classification problem is to identify the next stop after s. While training we do not consider any VRP constraints, i.e., the transition probabilities of all the stops can be nonzero regardless of whether they are active or not. Once the neural network predicts the transition probability vector $\mathbf{p}_{(s)}$, we compute the cross entropy loss with respect to the actual solution $\mathbf{x}_{(s)}$.

$$L(\mathbf{p}_{(s)}, \mathbf{x}_{(s)}) = -\sum_{u \in V} x_{su} \log(p_{su}) \tag{12}$$

Algorithm 2 Evaluation of Maximum Likelihood Routing.

```
Input: $S^{T+1}, z^{T+1}, \{\mathbf{X}^t\}_{t=1}^{T+1}$
1  for each stop s in $S^{T+1}$ do
2      active days ⟵ list of days s is active
3      Model ← Algorithm 1 ($\{(S^t, z^t, \mathbf{X}^t)\}_{t=1}^{T}$, s) // Model training
4      if length(active days) $\geq$ look back period then
5          $Past^T_{(s)} \leftarrow \{\mathbf{X}^{T'}[s,:] | T' \in$ active days[-(look back period):] }
6      end
7      else
8          $Past^T_{(s)} \leftarrow \{\mathbf{X}^{T'}[s,:] | T' \in$ active days }
9      end
10     $\mathcal{H}^{T+1} \leftarrow \left( z^{T+1}_{(s)}, Past^{T+1}_{(s)} \right)$
11     Model ⟵ Model dictionary [s]
12     $\hat{P}^{T+1}$[s,:] ⟵ Model.predict($\mathcal{H}^{T+1}$)
13 end
14 MLE Routing ($-log(P^{T+1})$); Compare with $X^{T+1}$
```

Finally this loss is backpropagated to update the parameters of the neural network. Algorithm 2 shows how we utilize the estimated transition probabilities to come up with the maximum likelihood solutions. Once we train the models for each stop using the data available until day T, we use it for routing on day $T + 1$. To do so, first we estimate the transition probabilities for each stop using the trained models. The (s, r)-th entry of the matrix $\hat{P}$ contains the estimated transition probability of going from stop s stop r. Then using the estimated transition probability matrix $\hat{P}$, we solve the maximum likelihood routing problem.

5 Decision Focused Learning

The approaches proposed so far consider the prediction of the transition probabilities and the VRP optimization separately. Such approaches can be viewed as two-stage approaches [4], where a neural network model is separately trained to estimate the unknown coefficients of an optimization problem.

One drawback of such a two-stage approach is the neural network model fails to incorporate information from the optimization problem. As the neural network model is trained without regard for the downstream optimization problem, the loss function fails to consider the impact of the predicted coefficients on the final objective value of the optimization problem.

Decision focused learning approaches [5, 25], on the other hand, consider how effective the predicted values are to solve the optimization problem and is trained with respect to the *optimization task loss* rather than a prediction loss such as cross entropy loss.

In Algorithm 3 we show our implementation of the decision focused learning approach for this problem. We implement the methodology of [20] to differentiate a combinatorial optimization problem with linear objective.

Algorithm 3 Decision Focused Learning Algorithm.

```
   Input : Historic solutions till days T: ${(s^t, z^t, \mathbf{X}^t)}_{t=1}^T$
1  for training epochs do
2      for $t$ in 1 to $T$ do
3          for each stop s in $S^{T+1}$ do
4              $x^t_{(s)} \leftarrow \mathbf{X}^t[s,:]$
5              $Past^t_{(s)} \leftarrow \{\mathbf{X}^{t'}[s,:] | t' \in \textit{active days} \wedge |Past^t_{(s)}| \leq \text{lookback period}\}$
6              $\mathcal{H}^t \leftarrow \left(z^t_{(s)}, Past^t_{(s)}\right)$
7              $\hat{P}^t[s,:] \leftarrow \text{Model}^{(s)}.Predict(\mathcal{H}^t)$
8          end
9          $\hat{\pi} \leftarrow -log(\hat{P}^t)$
10         $\hat{X}^t \leftarrow \text{MLE Routing}(\pi)$
11         $L \leftarrow sum(ReLU(X^t - \hat{X}^t))$
12         $\tilde{\pi} \leftarrow \hat{\pi} - \lambda \frac{dL}{d\hat{X}^t}$
13         $\tilde{X} \leftarrow \text{MLE Routing}(\tilde{\pi})$
14         $\nabla_\pi(L) \leftarrow -\frac{1}{\lambda}[\hat{X}^t - \tilde{X}]$
15         for each stop s in $S^{T+1}$ do
16             $\text{Model}^{(s)}.\text{backpropagate}\left(\nabla_\pi(L)[s,:]\right)$
17         end
18     end
19 end
```

They consider an optimization problem $\min_{X \in \mathcal{X}} f(\pi, X)$ with a linear objective. $\hat{X}^*(\hat{\pi})$ is the solution by using predicted $\hat{\pi}$ and the final optimization task loss is $L(X^*(\pi), \hat{X}^*(\hat{\pi}))$. The gradient of this task loss with respect to $\hat{\pi}$ is the following

$$\nabla_{\hat{\pi}} L(X^*(\pi), \hat{X}^*(\hat{\pi})) = -\frac{1}{\lambda}[\hat{X}^*(\hat{\pi}) - \tilde{X}^*(\tilde{\pi})] \tag{13}$$

where $\tilde{\pi}$ is a perturbation around the predicted $\hat{\pi}$, given by

$$\tilde{\pi} = \hat{\pi} + \lambda \frac{dL(X^*(\pi), \hat{X}^*(\hat{\pi}))}{d\hat{X}^*(\hat{\pi})} \tag{14}$$

In our setting, π is the matrix of negative log probability vectors i.e. $\pi = -\log(P)$, and X is the resulted routing. The final task is to minimize the difference between the actual route and the proposed route. So a suitable choice for the *task loss* is to consider arc difference, the number of arcs present in the actual solution but not in the predicted solution. Formally,

$$L(X^*(\pi), \hat{X}^*(\hat{\pi})) = Sum(ReLU(X^*(\pi) - \hat{X}^*(\hat{\pi}))) = \sum_{(i,j) \in \dim(X)} \max\left(x_{ij} - \hat{x}_{ij}, 0\right) \tag{15}$$

here *Sum* is the summation of all the elements of the matrix. The derivative of L can be computed s follows

$$\frac{dL(X^*(\pi), \hat{X}^*(\hat{\pi}))}{d\hat{X}^*(\hat{\pi})} = \begin{cases} -1 \text{ if } \hat{x}_{ij} < x_{ij} \\ 0 \text{ otherwise} \end{cases} \quad \forall (i,j) \in \dim(\hat{X}^*) \tag{16}$$

If we consider a squared loss instead of the ReLU, we replace $\max\big(x_{ij} - \hat{x}_{ij}, 0\big)$ with $\big(x_{ij} - \hat{x}_{ij}\big)^2$. In this case the derivative would be

$$\frac{dL(X^*(\pi), \hat{X}^*(\hat{\pi}))}{d\hat{X}^*(\hat{\pi})} = \begin{cases} -2 \text{ if } \hat{x}_{ij} < x_{ij} \\ 2 \text{ if } \hat{x}_{ij} > x_{ij} \\ 0 \text{ otherwise} \end{cases} \quad \forall (i,j) \in \dim(\hat{X}^*) \tag{17}$$

Intuitively, if $i \to j$ is present in X, but not in $\hat{X}$, then we lower π_{ij} by λ and with this generate a new solution with. A scaled difference between these two solutions is the gradient with respect to π.

6 Experimental Evaluation

6.1 Evaluation Criteria

We are interested in how the MLE routing solutions differ from the used routes. To do so, we evaluate the performance using the following two evaluation measures.

Arc Difference (AD). measures the number of arcs traveled in the actual solution but not in the MLE routing solution. It is calculated by taking the set difference of the arc sets of the test and predicted solutions. The percentage is computed by dividing AD by the total number of arcs in the whole routing.

Route Difference (RD). indicates the number of stops that were incorrectly assigned to a different route. Intuitively, RD may be interpreted as an estimate of how many moves between routes are necessary when modifying the predicted MLE solution to match the actual routing. To compute RD, the pair of routes with the smallest difference in stops is greedily selected without replacement. The total number of incorrectly assigned stops is considered as RD. The percentage is computed by dividing RD by the total number of stops in the whole routing.

We also present the cross entropy (CE) loss as the neural network models are trained with respect to this criterion.

6.2 Data Description

For empirical evaluation[1], we use actual historical data from a logistics company to compare the performance of our proposed approaches against the Markov model presented in [1]. The data consists of 201 daily routings collected in a span of 39 weeks. It has 73 unique customers, each representing a node other than the depot. In each instance, an average of 31 stops are serviced by an average of 8 vehicles. We group the instances by day of the week, giving us an average of 29 instances per weekday. In training and testing the models, we used a 75%-25% split while ensuring that we have exactly 7 testing instances per weekday. We use a rolling window model for valuation, where the lookback period remains fixed and counts backwards from the most recent observation.

In Table 3, we present the percentage AD and RD of the Markov approaches on the test instances for each day of week. The Markov (allday) approach arrives on the transition

[1] The code and the anonymised data are available at `https://github.com/JayMan91/CP2021-Data-Driven-VRP`.

probability by considering all past days. On the contrary, the Markov (weekday) approach considers only those past instances which occurred on the same day of week as the evaluation instances. Both the approaches use Eq. (10) to compute the probabilities. We can see in Table 3 that Markov (allday) performs better on the weekdays, but its result worsens on the weekends. Due to the operational characteristics of the company, the number of customers, number of available vehicles, and hence the routing decisions tend to be highly dependent on the day-of-week. So there is a strong influence of the day on the probabilities. Probably, this is why [1] preferred the Markov (weekday) approach. The motivation behind the neural network approach is that it can consider the day of week as feature, so that we do not have to compute the probability separately for each day. Moreover, other feature variables can easily be passed into the neural network.

In our experiment we consider the following feature variables to predict the transition probabilities– a. day of week, b. the set of stops to be served, c. distance between the stops, d. number of available vehicles, e. routings used in the past, f. transition probabilities computed by Markov (weekday).

6.3 Experimental Results

In this section, we will address the following research questions

- Choice of the feature variables and the network architecture in a systematic way
- Compare the quality of predictions of the neural network trained with respect to the CE loss with that of Markov counting approach
- The effectiveness of a decision focused approach, which trains the network to directly minimize AD

6.3.1 Choice of Network Architecture and Feature Variables

As mentioned in Section 4, there are many choices for the network architecture and because of limited data we are careful to avoid overfitting.[2] In Table 1, we first present the impact of feature variables on the quality of predictions. We show the cross entropy loss on both training and test data and AD and RD on test data. We point out that the network presented in Figure 1 results in the lowest training loss among them. On the other hand, a network only with Markov probabilities as input lowers test loss and lower AD, RD. A network without the time-lagged data has even lower CE loss on the test instances and lowest AD and RD suggesting that past information is already subsumed in the Markov probabilities, making the time-lagged information redundant. The Markov probabilities along with the contextual information seem to be the right choice for the feature variables.

In the lower section of Table 1, we present two alternative architecture choices. The first one replaces the linear layer of the lagged solutions with an LSTM. The second one has different weights for different destination stops in the final layer in Figure 1.

It shows that using different weights for different destination stops results in lowest training CE loss. But this model clearly overfits, as the performance is poor on the test data. The LSTM model seems to improve on CE loss but not on AD and RD measures. So overall, the model **without the past data** results in lowest CE loss as well as lowest AD and RD. This model has the Markov probabilities as input, and the past information carried by this probability.

[2] We use Pytorch [18] and Gurobi [9] for neural network and VRP models respectively.

Table 1 Investigation into feature variables and architectures.

Model	Training CE	Test CE	AD	AD (%)	RD	RD(%)
Experiment on feature variables						
Neural Net	**2.14**	1.10	6.27	19.80	4.57	18.04
Neural Net (without past data)	2.48	**1.04**	**5.68**	**18.04**	**4.30**	**17.02**
Neural Net (without weekday)	**2.14**	1.09	6.24	19.75	4.59	18.03
Neural Net (without stop information)	2.20	1.13	6.28	19.86	4.56	17.97
Neural Net (without distance)	2.20	1.10	6.18	19.54	4.46	17.62
Neural Net (without Markov probabilities)	2.43	1.49	7.99	26.26	5.32	21.38
Neural Net (only Markov probabilities)	2.58	1.07	5.95	18.85	**4.29**	**16.93**
Experiment on architecture choice						
LSTM	2.22	**1.01**	6.35	20.10	4.49	17.75
Linear Layer different for each stop	**1.37**	1.82	7.21	22.81	4.74	18.57

6.3.2 Neural Network Predictions

Table 2 Comparison of Neural Network with Markov Counting (Actual Distance is 413 km.)

	CE loss	Arc Difference (AD)		Route Difference (RD)		Distance (km.)
		Absolute	Percent	Absolute	Percent	
Markov (allday)	2.77	10.33	35.69	6.29	25.75	424
Markov (weekday)	2.44	5.86	18.55	4.39	17.26	418
Neural Net	**1.04**	**5.68**	**18.04**	**4.30**	**17.02**	414
Conventional VRP	11.90	21.47	73.14	11.65	46.93	**366**

The last section suggests to consider a network without the lagged variables for this task. Next, we compare the quality of predictions of this model shown to that of Markov counting approach. We present the average of CE loss, AD, RD between the actual solutions and generated solution on test instances in Table 2. We also present the distance of the solutions of these approaches. We point out in Table 2 that a neural network model results in lower CE loss, which is expected as the model is trained with that objective. Moreover, we also observe lower AD and RD with this model. We also present results of a conventional VRP algorithm, which is the best in terms of total distance covered, but clearly very far off from the preferred solution. Table 3 presents this comparison in more detail, where we evaluate for each day of week separately. Although we do not need to train the neural network separately

Table 3 Daywise Analysis of Arc Difference and Route Difference.

	Arc Difference(%)			Route Difference(%)		
	Markov (allday)	Markov (weekday)	Neural Net	Markov (allday)	Markov (weekday)	Neural Net
Monday	51.53	**23.62**	**23.62**	27.98	21.75	**21.25**
Tuesday	**24.96**	25.61	25.82	29.15	**28.87**	30.85
Wednesday	**19.61**	21.30	20.48	17.96	15.12	**14.61**
Thursday	24.89	22.86	**21.17**	19.75	18.63	**17.08**
Friday	19.18	19.38	**18.08**	13.74	13.19	**11.54**
Saturday	51.59	**0.00**	**0.00**	25.21	**0.00**	**0.00**
Sunday	58.09	**17.08**	17.11	46.48	**23.23**	23.82
Overall	35.69	18.55	**18.04**	25.75	17.26	**17.02**

for each day of week, by considering the Markov probabilities as inputs, it is able to generate predictions which result in lower AD and RD. This demonstrates the advantage of the neural network approach, which can consider multiple inputs, contextual as well as temporal, in a single framework.

Figures 2 to 6 illustrate our approach for one instance. Figures 3 and 4 present the learned transition probabilities and Figures 5 and 6 show the MLE routing of Markov weekday and neural net respectively.

6.3.3 Decision Focused Learning

Next, we experiment with the decision focused learning approach introduced in section 5. We use the same neural network architecture but trained with arc difference as the loss function, and the loss backpropagted through a corresponding subgradient (Eq. (16)). We present the

Table 4 AD and RD with Decision Focused Learning.

	CE loss (test)	Arc Difference (AD)		Route Difference (RD)		Distance (km.)
		Absolute	Percent	Absolute	Percent	
Relu loss	2.77	13.76	45.96	9.51	38.54	434
Squared loss	3.97	13.31	44.27	9.10	37.14	436

solution quality of this approach in Table 4. We can see, it fails to generate lower AD and RD on the test instances. In fact, we observe AD reducing on training instances but not on test instances, suggesting a case of overfitting. Only 152 instances is not enough to train a complex model like this. So the poor quality can be attributed to limited amount of data.

7 Conclusion

We presented a neural network model which learns the transition probabilities between stops in a CVRP setting. With these transition probabilities, we solve the MLE routing problem instead of the conventional VRP. The resulting solution is able to mimic the solution preferred by the route planners and drivers. In this way, we are able to include the preferences of the planners and the drivers implicitly in the VRP solution.

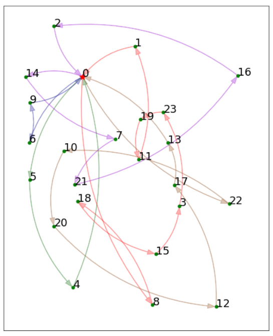

Figure 2 Human-made solution
.

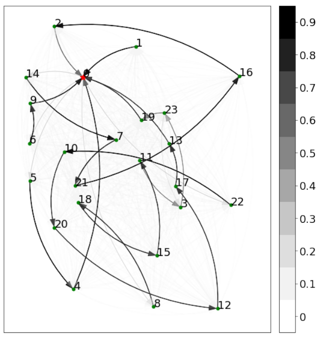

Figure 3 Learned probabilities (Markov)
.

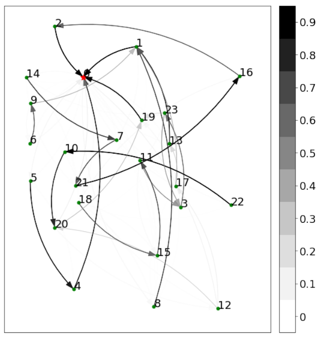

Figure 4 Learned probabilities (NN)
.

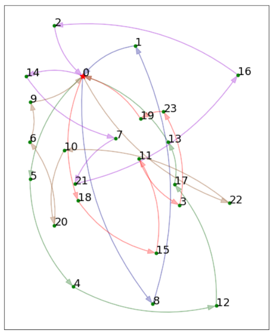

Figure 5 Markov solution.

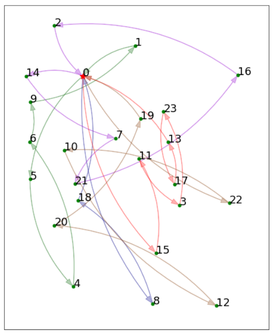

Figure 6 NN solution.

We extend on the work of [1]. The novelty of our approach is to use a neural network model to estimate the probabilities. Key developments are the use of an arc-based architecture to control the number of trainable parameters, and the identification of the standard cross-entropy classification loss as a suitable (and cheap to compute) proxy loss for training. This leads us to develop a general framework for such problem setting, which has the flexibility of taking contextual features including the output of [1] into consideration. By considering the contextual features in a principled way, our approach marginally outperforms [1], emphasising the advantage of a generic approach.

We also use a decision focused learning approach which directly trains the neural network to minimize the final objective of minimizing the difference between the generated solution and the preferred solution. Although this approach considers the structure of the VRP optimization problem, our results show that it fails to generate good quality solutions in the test data. We believe it is due to the limited number of training instances.

Our methodology relies on the presence of recurrent stops in our training set. Future research will aim to extend our methodology to learn preferences over non-recurring stops. It will also be interesting to investigate loss functions that include the structure of the CVRP tackling the challenges of the scalability.

References

1 Rocsildes Canoy and Tias Guns. Vehicle routing by learning from historical solutions. In *Principles and Practice of Constraint Programming - 25th International Conference, CP 2019, Stamford, CT, USA, September 30 - October 4, 2019, Proceedings*, volume 11802, pages 54–70. Springer, 2019.

2 Vaida Ceikute and Christian S Jensen. Routing service quality–local driver behavior versus routing services. In *2013 IEEE 14th International Conference on Mobile Data Management*, volume 1, pages 97–106. IEEE, 2013.

3 George B Dantzig and John H Ramser. The truck dispatching problem. *Management science*, 6(1):80–91, 1959.

4 Emir Demirovic, Peter J. Stuckey, James Bailey, Jeffrey Chan, Chris Leckie, Kotagiri Ramamohanarao, and Tias Guns. An investigation into prediction + optimisation for the knapsack problem. In Louis-Martin Rousseau and Kostas Stergiou, editors, *Integration of Constraint Programming, Artificial Intelligence, and Operations Research - 16th International Conference, CPAIOR 2019, Thessaloniki, Greece, June 4-7, 2019, Proceedings*, volume 11494 of *Lecture Notes in Computer Science*, pages 241–257. Springer, 2019.

5 Adam N Elmachtoub and Paul Grigas. Smart “predict, then optimize”. *Management Science*, 2021.

6 Stefan Funke, Sören Laue, and Sabine Storandt. Deducing individual driving preferences for user-aware navigation. In *Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 1–9, 2016.

7 Klaus Greff, Rupesh Kumar Srivastava, Jan Koutník, Bas R. Steunebrink, and Jürgen Schmidhuber. LSTM: A search space odyssey. *IEEE Trans. Neural Networks Learn. Syst.*, 28(10):2222–2232, 2017.

8 Chenjuan Guo, Bin Yang, Jilin Hu, Christian S Jensen, and Lu Chen. Context-aware, preference-based vehicle routing. *The VLDB Journal*, pages 1–22, 2020.

9 LLC Gurobi Optimization. Gurobi optimizer reference manual, 2020. URL: `http://www.gurobi.com`.

10 Xiangpei Hu, Zheng Wang, Minfang Huang, and Amy Z. Zeng. A computer-enabled solution procedure for food wholesalers’ distribution decision in cities with a circular transportation infrastructure. *Computers & Operations Research*, 36(7):2201–2209, 2009.

11 Nicolas Jozefowiez, Frédéric Semet, and El-Ghazali Talbi. Multi-objective vehicle routing problems. *European journal of operational research*, 189(2):293–309, 2008.

12 Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. `arXiv:1412.6980`.

13 Christophe Lecluyse, Tom Van Woensel, and Herbert Peremans. Vehicle routing with stochastic time-dependent travel times. *4OR*, 7(4):363–377, 2009.

14 Tzong-Ru Lee and Ji-Hwa Ueng. A study of vehicle routing problems with load-balancing. *International Journal of Physical Distribution & Logistics Management*, 1999.

15 Julia Letchner, John Krumm, and Eric Horvitz. Trip router with individualized preferences (trip): incorporating personalization into route planning. In *IAAI’06 Proceedings of the 18th conference on Innovative applications of artificial intelligence - Volume 2*, pages 1795–1800, 2006.

16 Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

17 Chun Kai Ling, Fei Fang, and J. Zico Kolter. What game are we playing? end-to-end learning in normal and extensive form games. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*, pages 396–402. ijcai.org, 2018.

18 Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pages 8024–8035, 2019.

19 Yong Peng and Xiaofeng Wang. Research on a vehicle routing schedule to reduce fuel consumption. In *2009 International Conference on Measuring Technology and Mechatronics Automation*, volume 3, pages 825–827, 2009.

20 Marin Vlastelica Pogancic, Anselm Paulus, Vít Musil, Georg Martius, and Michal Rolinek. Differentiation of blackbox combinatorial solvers. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*, 2020.

21 J. David Schaffer. Multiple objective optimization with vector evaluated genetic algorithms. In *Proceedings of the 1st International Conference on Genetic Algorithms*, pages 93–100, 1985.

22 Tomer Toledo, Yichen Sun, Katherine Rosa, Moshe Ben-Akiva, Kate Flanagan, Ricardo Sanchez, and Erika Spissu. Decision-making process and factors affecting truck routing. In *Freight Transport Modelling*. Emerald Group Publishing Limited, 2013.

23 P Toth and D Vigo. The family of vehicle routing problem. *Vehicle Routing: Problems, Methods, and Applications*, pages 1–23, 2014.

24 Po-Wei Wang, Priya L. Donti, Bryan Wilder, and J. Zico Kolter. Satnet: Bridging deep learning and logical reasoning using a differentiable satisfiability solver. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 6545–6554, 2019.

25 Bryan Wilder, Bistra Dilkina, and Milind Tambe. Melding the data-decisions pipeline: Decision-focused learning for combinatorial optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 1658–1665, 2019.

26 Yiyong Xiao, Qiuhong Zhao, Ikou Kaku, and Yuchun Xu. Development of a fuel consumption optimization model for the capacitated vehicle routing problem. *Computers & Operations Research*, 39(7):1419–1431, 2012.

A Experiment Setup

We use Pytorch [18] and Gurobi [9] for neural network and VRP models respectively. We use Adam optimizer [12] implementation of Pytorch. The hyperparameters of each setup is detailed below.

Table 5 Hyperparameters Configuration (For all experiments the embedding dimension of weekday and stop feature are 6 and 40 respectively).

	Learning rate	Epochs
Neural Net	0.1	50
Neural Net (without past data)	0.1	100
Neural Net (without weekday)	0.1	50
Neural Net (without stop information)	0.1	100
Neural Net (without distance)	0.1	100
Neural Net (without Markov probabilities)	0.1	100
Neural Net (only Markov probabilities)	0.1	100
LSTM	0.1	50
Linear Layer different for stops	0.01	100