Machine-Learning Models for Route Consolidation

WWT Artificial Intelligence Research & Development

APRIL | 2020

World Wide Technology
wwt.com

Table of Contents

Abstract

Finding common routes from a large set of individual trips is a difficult problem due to the natural complexity involved with nontrivial trips. Much of the current research for route consolidation has relied on clustering- or distance-based methods, along with ad-hoc rules for combining routes. We compare a more traditional physical-based method that uses clustering, graph theory and ad-hoc rules with a machine-learning method. In particular, the machine-learning method uses an autoencoder to reduce the number of trips in the dataset and find common or standard routes. The routes identified using the autoencoder are then post-processed using K-means clustering and Procrustes distance. We apply both methods to a mine haul truck trip dataset and show how the machine-learning method can largely replicate the results produced by the physical modeling method, thus providing a more generalizable alternative for route consolidation.

Business Justification

As the number of location-based services has increased, there has been an increase in the need to find common patterns or routes. While many location-based services focus on the problem of finding the quickest route, there are many practical benefits to identifying the most common route or creating standard routes. For example, identifying common traffic patterns or so-called “hot-routes” can have many practical benefits such as helping city planners and emergency personnel or providing a common set of routes to compare drivers. In general, “hot-routes” are defined as commonly traveled routes that are identified from a larger set of trips. This problem can be challenging for even medium-sized networks as the number of potential routes can increase exponentially with the number of potential segments (i.e., assuming routes are made up of multiple segments).

Much of the current literature regarding this problem considers a physical modeling approach using methods such as clustering (i.e., Li et al 2007, Froehlich and Krumm 2008, Su et al. 2013) or other similarity metrics (Wang et al. 2013). These clustering- or distance-based methods generally rely on physical-based features such as alignment, direction, speed, etc. In many cases, these hot-route methods use multi-step algorithms that involve *ad-hoc* rules that may lack generalization to a wider range of problems. To our knowledge, there has not been an attempt, to date, to apply machine learning to this problem. Using machine learning, in our case an autoencoder, we develop a potentially more generalizable methodology that can be applied to variety of route consolidation problems from disparate fields.

Two methods are explored here: a more traditional clustering algorithm paired with graph theory using ad-hoc rules (referred to as the *ad-hoc method*) and a machine-learning method using a multi-layer autoencoder with clustering and Procrustes distance (referred to as the *machine-learning method*).

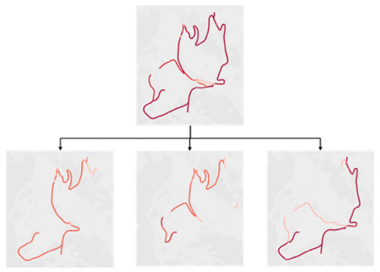
Although the methodology presented here can be applied to many different applications, we focus on a specific application related to mining routes. The analyzed data relies on GPS records for each trip taken by the truck of interest. Surface mines require haul truck operators to frequently travel between dump and shovel locations to transport raw rocks for further processing. Although the drivers are assigned where to go next, they do have some latitude to make diversions due to road closures or fueling needs. It is common for haul truck operators to rely on their experience and current observations of traffic to determine the routes they travel. One of the advantages of having standard routes in the mining industry is that the shovel and dump that trucks move between are constantly changing. Thus, by finding standard routes, drivers can accurately be compared and analyzed. Without this common language of standard routes, it is almost impossible to compare drivers in such a way that efficiency can be improved. This way, surface mining companies can optimize the transportation of materials and increase the daily output of processed minerals.

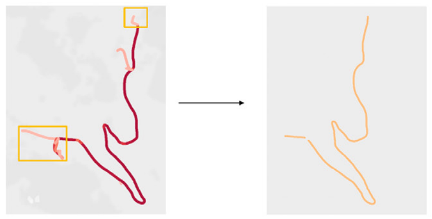
Methodology

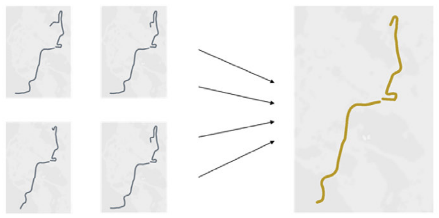
AD-HOC METHOD

Clustering algorithms use a similarity (or dissimilarity) metric to form clusters of similar data. Using pairwise clustering, we compare pairs of trips when finding route clusters. This clustering is only carried out on trips that have the same shovel and dump location (starting and ending location; see Figure 1 for an example) and follow a similar path. Note, a route is generated for each identified cluster of trips. Furthermore, routes identified by the clustering algorithm that are similar are also combined (see Figure 1). Next, we describe how this clustering method fits into the overall ad-hoc method algorithm.

The route consolidation ad-hoc algorithm involves five steps. Initially, a historic set of 6,575 haul truck trips are processed so that each trip has a connected path. This involves using Dijkstra's algorithm from graph theory by defining a starting and ending point and finding the optimal path. The algorithm then separates out groups of trips that have large differences using a pairwise modes clustering algorithm to produced routes (as described above). Trips are matched to a route if they share at least 60 percent of the same segments from a defined route (produced by the clustering algorithm). Furthermore, only routes with 50 or more matching trips are retained. For each collection of trips from the same shovel and dump location set, the algorithm determines the most common route taken based on the number of times each road segment was traveled. Once a route is determined for a particular pair of shovel and dump locations, the final route is determined after two post-processing steps. First, terminal road segments are left out since they have a high amount of variation (see Figure 1). Finally, when there are gaps near intersections, routes are sutured via Dijkstra's algorithm (see Figure 1).


(a): Separating trips by shovel and dump locations


(b): Removing terminal road segments


(c): Combining similar routes into a single route

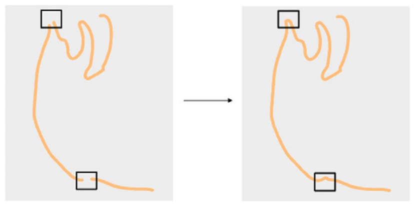

(d): Suturing routes at complex intersections

FIGURE 1: Various operations used in the route consolidation algorithm.

MACHINE-LEARNING METHOD

The goal of the developed machine-learning method is to use an autoencoder for dimension reduction on a large collection of trips to find common routes. Furthermore, autoencoders are a type of nonlinear dimension reduction technique that have gained popularity with the advancement of deep learning. Autoencoders are defined by a neural network structure where both the input and output of the model are the same. The goal of an autoencoder is to learn a mapping from the input to the output. That is, the hidden states of an autoencoder encode information about the input in a reduced space. This encoding is possible since the hidden states have a much smaller dimension than the input/output (dataset of trips). Due to this smaller dimension, the hidden-states are forced to capture the most salient or variable information in the raw data.

Regarding the route application, the autoencoder developed here reduces the number of trips, with the hidden states of the autoencoder representing common or typical routes. The input/output data is formed by converting each route to a hot-vector using ones and zeros. That is, each element of the hot-vector represents a potential segment from an overall trip, if a particular segment is passed over during a particular trip, that segment is coded with a one, segments that are not passed over are coded with zeros.

Critically, unlike the ad-hoc method described above, the machine-learning method is not given any information about order or starting/ending points of the route. For example, the ad-hoc algorithm creates routes based on shovel and dump locations (starting and ending locations) and uses Dijkstra's algorithm to create continuous routes from a defined starting position. Thus, the machine-learning algorithms uses less physical modeling and determines routes purely from the raw data.

Further, autoencoders can be made arbitrarily deep by adding additional hidden layers to the model. Deep autoencoders are useful for complex problems in which the process of interest needs to be learned slowly in a generalizable way. Image processing problems often employ deep autoencoders to learn complex features. As shown below, we test a variety of different autoencoders, including a deep version. Generally, the middle-hidden layer is extracted from an autoencoder as the compressed or dimension reduced version of the raw data.

Once the middle-hidden layer from the autoencoder has been extracted, we perform K-means clustering on these hidden units. As described in Friedman, Hastie and Tibshirani (2001), in high dimensions, local neighbors do not exist, this is generally referred to as the curse-of-dimensionality. Thus, applying the clustering to the hidden units of the autoencoder is more appropriate than clustering the high-dimensional raw data. The results below also suggest that using the clustering produces more coherent routes than just using the routes suggested by the autoencoder.

Since the autoencoder lacks any knowledge of the order of the routes, it is possible to get unrealistic routes that may have large gaps (i.e., routes that are missing segments that make the route physically impossible). To address this issue, we post-process or bias-correct the routes towards the true observed routes. There is a similar idea in the climate literature of bias correcting climate model output towards observed data. This process involves using Procrustes distance (Friedman, Hastie, and Tibshirani 2001) to find observed routes that are most similar to the routes produced by the clustered autoencoder's hidden units. Procrustes distance is a method for translating, scaling and rotating two shapes to minimize the distance between the two shapes. Here, the routes can be thought of as two-dimensional shapes. Therefore, comparing routes can be thought of as comparing shapes, and thus, Procrustes distance can be applied to this problem.

Thus, the post-processing procedure involves taking each autoencoder's hidden cluster and finding the trip from the 6,575 observed trips that has the minimum Procrustes distance. Once this trip has been identified, we replace the autoencoder's hidden cluster of interest with the identified observed route. Next, we reduce the high-dimensional set of observed routes by removing any routes that share more than 60 percent of the segments from the identified route. This ensures that the algorithm is finding a diverse set of routes and not continually picking the same route. This procedure is carried out in an iterative fashion as described in Algorithm 1.

Algorithm 1 Procrustes Post-processing Algorithm

```
for $i = 1, 2, \ldots, n_{routes}$ do
    1. Find the $ObservedRoute^*_j$, from all possible observed routes ($ObservedRoute^*_{1:N}$) that has the
    minimum Procrustes distance to $AutoencoderRoute_i$
    2. Set $UpdatedRoute_i = ObservedRoute^*_j$
    3. Find all of the routes in $ObservedRoute^*_{1:N}$ that share at least 60% of the segments from
    $UpdatedRoute_i$, call this set of routes $OverlapRoutes_i$
    4. Take out all of the routes from $OverlapRoutes_i$ that appear in $ObservedRoute_{1:N^*}$ and update
    $ObservedRoute_{1:N^*}$
end for
```

Implementation and Results

Both methods described above are evaluated on 6,575 historic haul truck trips. These trips are all so-called loaded trips, which are defined as trips where the truck is loaded with material. These routes come from a five-day period at a single mine. Additionally, both methods were carried out using personal laptops due to the relatively small number of routes. The machine-learning method was also deployed on an NVIDIA Tesla V100 GPU in the cloud using Amazon Web Services (AWS). The GPU version of the model only took 55 seconds to run and was an order of magnitude faster than the version deployed on a personal laptop. As previously mentioned, each of the 6,575 trips are pre-processed into hot-vectors of size 152 (for the 152 segments traveled across during the five days of interest) for the autoencoder. The original 6,575 trips can be seen in Figure 2.

HAUL TRUCK ROUTE RESULTS

The autoencoder is trained in Python with TensorFlow using 150 epochs and a batch size of 5. The final autoencoder model consists of five hidden layers of the following sizes: 300, 60, 12, 60, 300, such that the middle-hidden layer has 12 hidden-units. Each of the internal layers uses the Relu activation function, while the output layer uses the sigmoidal activation due to the binary form of the data (i.e., binary hot-vectors). Thus the 6,575 original trips have been reduced to 12 routes. We experimented with architectures that consisted of fewer and larger numbers of layers and found that 5-layers generally produced the most coherent routes.

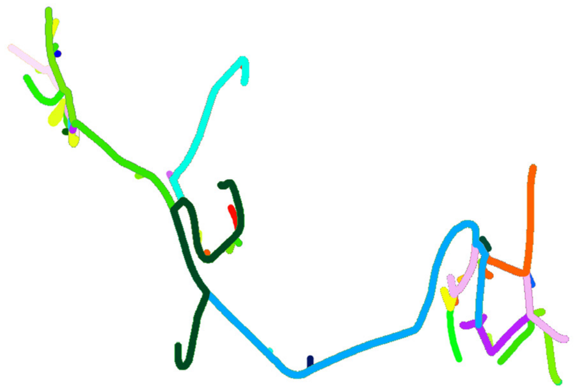


FIGURE 2:

Historic haul trips (6,575) over a five-day period at a surface mine. Note, due to the number of trips we have not adjusted this plot for overlapping trips.

Since the Relu activation function is used with the extracted middle layer of the autoencoder, the hidden units will have values between zero and infinity. Recall, the routes are defined by binary hot-vectors. Thus, a thresholding technique is used to map the hidden units back to the original binary space. In particular, for each hidden unit, all of the values above the 85th percentile are assigned ones, while the remaining values are assigned zeros. We found that this threshold value largely matched the lengths of the observed routes. Note, the thresholding is applied after the K-means clustering, although in theory, it could be applied before. The routes found by the autoencoder can be found in Figure 3. We have applied a post-processing procedure for the sake of visualization, to each route figure, so that overlapping routes can be seen. This post-processing creates so-called subway routes and adds a small level of noise to the visualized routes. Clearly the autoencoder is filtering out some of the noise in the data and finding salient routes, especially when comparing the autoencoder routes to the raw trips in Figure 2. Unfortunately, some of the routes are physically disconnected involving some large gaps.

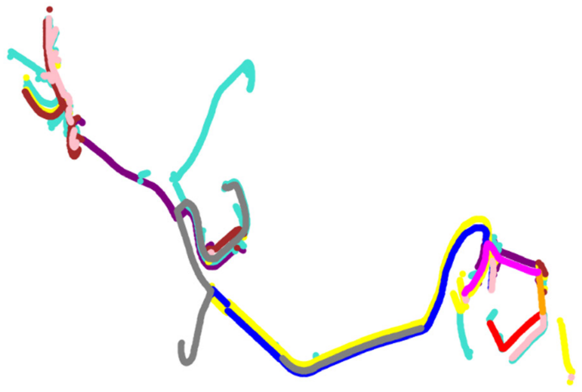


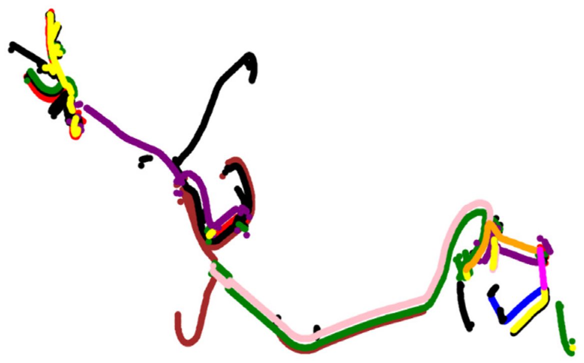
FIGURE 3:

Routes for the five-layer autoencoder before any post-processing.

Next, the K-means clustering algorithm is applied to the routes found by the autoencoder. The routes produced by the clustering have slightly more coherence than the original autoencoder routes, but they still have a variety of physically unrealistic characteristics. Using Algorithm 1 and Procrustes distance we can correct these unrealistic features.

FIGURE 4:

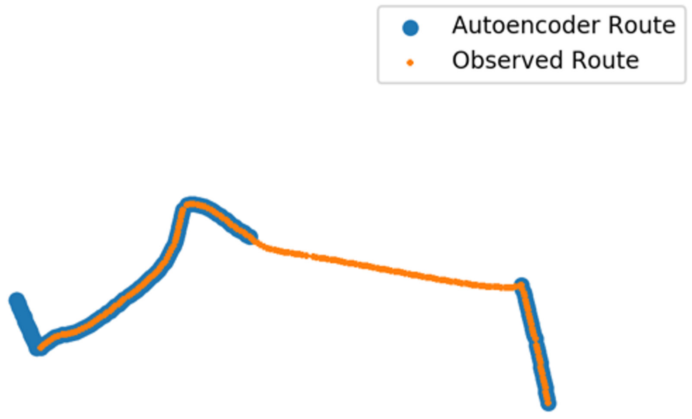
Routes after the K-means clustering is applied to the hidden-units from the autoencoder.



An example of post-processing the clustered autoencoder routes can be found in Figure 5. The example in Figure 5 shows an autoencoder route that is clearly not realistic from a physical perspective, while the matched route (in blue), found from the raw data using Procrustes distance, both matches the overall shape of the autoencoder route and is, by definition, physically feasible. Using Algorithm 1 (defined above) all of the clustered autoencoder routes from Figure 4 are post-processed leading to the final set of routes for the machine learning method shown in Figure 6.

FIGURE 5:

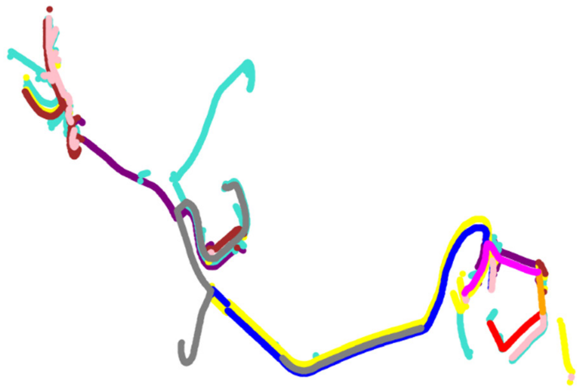
An example of using Procrustes distance to post-process the clustered autoencoder routes.



Clearly the final routes produced by the machine learning method are all physically realistic and match many of the routes found using the ad-hoc method (as shown in Figure 7).

FIGURE 6:

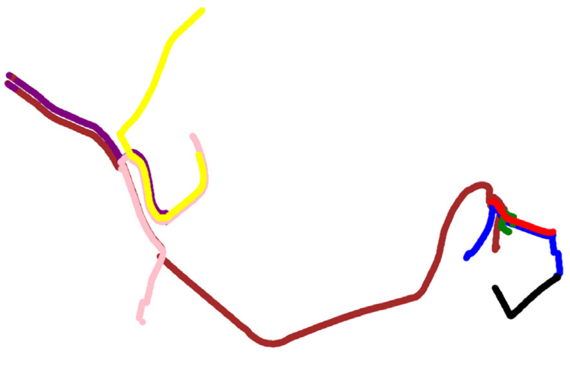
Final routes produced by the machine learning method after post-processing.



Overall, the two methods share many of the same routes, with the machine-learning method tending to have slightly longer routes and covering more overall area than the ad-hoc method. Considering the lack of underlying knowledge concerning the order or starting/ending locations of the routes, the machine-learning method manages to mostly match the ad-hoc method. Recall, the ad-hoc method performs route consolidation for individual starting and ending locations (i.e., shovel and dump locations). Therefore, it is possible that the ad-hoc routes contain more common starting and ending locations than the machine learning method. The number of routes is also a tuning parameter in the machine-learning algorithm (i.e., the number of clusters), which is selected here to match the ad-hoc method as closely as possible.

FIGURE 7:

The final routes produced by the ad-hoc method.



Conclusion

We have presented two distinct methods for route consolidation using both physical modeling and machine learning, respectively. While taking very different approaches, the two methods tend to give similar routes. These results demonstrate the potential to apply machine learning to route consolidation. Critically, at each of the three steps involved in the machine-learning method, the routes continually improve. Furthermore, the machine-learning method is generalizable and can provide a more general way of performing route consolidation without using domain specific rules. Although the use case here focused on mine routes, it is easy to imagine how this method could be applied to a variety of alternative route consolidation problems in other fields. For example, this method could be used to suggest popular hiking or bike routes to tourists or those unfamiliar with a certain area or non-physical processes such as internet traffic routing.

There are a variety of ways in which the machine-learning method could be improved. The most obvious improvement might include information about the order of the trips through either a data or modeling mechanism. For example, one potential path would be to apply a long short-term memory (LSTM) model to the problem. LSTM models are neural network models with an explicit mechanism for learning dependent sequences or dynamics. In natural language processing (NLP), LSTMs are often used to predict the next word in a sentence. A similar idea could be applied to the problem of route consolidation, by treating each of the segments as words and the trips as sentences.

Furthermore, it might also be possible to improve the results of the autoencoder by explicitly including information about the trips in the input/output data of the model. For this to be successful the model would have to learn starting and ending locations that go together in a way that produces sensible routes. Using such a method may produce routes that do not need to be post-processed towards the observed data.

References

Li, Xiaolei, et al. "Traffic density-based discovery of hot routes in road networks." *International Symposium on Spatial and Temporal Databases*. Springer, Berlin, Heidelberg, 2007.

Su, Han, et al. "Calibrating trajectory data for similarity-based analysis." *Proceedings of the 2013 ACM SIGMOD international conference on management of data*. 2013.

Wang, Haozhou, et al. "An effectiveness study on trajectory similarity measures." *Proceedings of the Twenty-Fourth Australasian Database Conference-Volume 137*. Australian Computer Society, Inc., 2013.

Froehlich, Jon, and John Krumm. *Route prediction from trip observations*. No. 2008-01-0201. SAE Technical Paper, 2008.

Friedman, Jerome, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*. Vol. 1. No. 10. New York: Springer series in statistics, 2001.

World Wide Technology

ARTIFICIAL INTELLIGENCE
RESEARCH & DEVELOPMENT

Learn more about WWT's Artificial Intelligence Research & Development at:
https://www.wwt.com/artificial-intelligence-research-and-development

ABOUT WWT ARTIFICIAL INTELLIGENCE RESEARCH & DEVELOPMENT

The Artificial Intelligence Research & Development program at WWT is an applied research initiative focused on investigating the one to three-year horizon of the artificial intelligence space. The AI R&D program conducts internal projects grounded in WWT's deep understanding of industry use cases and produces reusable components and white papers to share with customers and the AI community.

Through strategic partnerships, cutting-edge data science and ML Ops skills, and ability to test and learn in the Advanced Technology Center and public cloud, the AI R&D team advances WWT's knowledge of the AI and ML space, thus allowing WWT to be on the bleeding edge and remain strategic advisors to businesses in their AI needs.

ABOUT WWT

Founded in 1990, WWT has grown to become a global technology solution provider with nearly $12 billion in annual revenue. With thousands of IT engineers, hundreds of application developers and unmatched labs for testing and deploying technology at scale, WWT helps customers bridge the gap between IT and business. By bridging leading technology companies together in a physical yet virtualized environment through its Advanced Technology Center, WWT integrates individually impressive technologies to produce game-changing solutions.

Based in St. Louis, WWT employs more than 6,000 employees and operates over 4 million square feet of warehousing, distribution and integration space in more than 20 facilities throughout the world.