Predicting water pump failure in Tanzania and optimising maintenance routes

Sam Kim
Harvard College
Cambridge, MA 02138
samuelkim@college.harvard.edu

Gareth Haslam
Harvard Extension School
Cambridge, MA 02138
haslam.gareth@gmail.com

Abstract

We describe MCMC techniques to model the functionality of water pumps in Tanzania and suggest an optimised route for their maintenance. Using field data describing various attributes of each pump, such as construction year, installer, and location, we describe a method to predict the current status of the pump from a set of three possibilities (functioning, functioning needs repair, and not functioning). We also optimize the route that a maintenance crew could take to repair damaged pumps in a version of the well–known, NP–hard, traveling salesman problem. We find acceptable solutions using stochastic metaheuristics. Code and more information are available at http://ghaslam.github.io/AM207/.

1 Introduction

Access to clean water is a fundamental need for people all over the world. Around 11% of the world's population currently lack such access due to either lack of natural resources or lack of infrastructure [1]. In Tanzania, despite the availability of water, only 33.5% have access to a piped source [2]. Many others depend on pumps for their supply. When these pumps fail, that can create serious problems for the people who depend on them. Previous work on predicting the failure of mechanical devices such as pumps has benefitted from the use of both statistical analysis as well as data obtained from direct observation of the machines, for example diagnostic vibration sensors [3]. As the popularity of machine learning and 'big data' continues to grow, researchers are also studying how failure can be predicted based on training data from large historical datasets, for example for rod pumps in the oil industry [4]. Liu's research used subject matter experts to first identify whether a pump was operating normally or about to fail and then trained their model to recognise the features in the data such as daily usage, that predicted the label.

We aim to use Bayesian Methods to assign one of three categorical labels indicating the functional status each pump (functioning, functioning needs repair, and not functioning). The Bayesion models will be built on training data, then used to predict the functional status for wells with unknown status. Then, we treat the repair of the damaged pumps as a constrained optimisation problem over the space of all possible routes that the maintenance crew could take to reach the damaged wells. We ignore the existing road infrastructure, terrain, height etc. and assume that all locations are equally accessible.

2 The Data

The data for this project was provided as part of a data science challenge run by DrivenData.org [5]. DrivenData aims to encourage data scientists from around the world to take part in online competitions to develop predictive statistical models that can address important problems in the fields of health, international development, and the environment. The data for the Tanzanian water pump competition was aggregated from the Tanzanian Ministry of Water by Taarifa, an open source platform for tracking infrastructure related issues [6]. The dataset contains records of 59,400 water pumps with information on a range of 39 parameters, such as pump construction year, installer, type of pump, water quality, population around the well, cost of water, quantity and others. These include a mix of categorical and numerical data, and there are also many missing values. Each pump is assigned a unique ID as well as precise coordinates for its locations. A subset of the parameters are shown in Table 1. The final column in Table 1 indicates the current status of the pump which the model will attempt to predict. The optimisation algo-

rithm will then attempt to find the route that reaches the damaged pumps in the s

In addition to the parameters given in the original dataset, we calculate some additional parameters such as age (based on the construction year), and relative remoteness (based on the average distance of the 5 nearest neighbours). Figure 1a shows a map of the physical geography of Tanzania and another showing the locations of the pumps. The high density of pumps appears as a solid blue colour but actually represents many individual points.

3 Modeling water well functionality

3.1 Model Classification

Bayesian methods are only compatible with numerical data and do not work directly with classification problems, so we need to convert the categorical data and classes into numerical data and labels.

For the classification, we can take two approaches:

1. Assume that the 3 labels, "non functional", "functional needs repair", and "functional" lie on a spectrum where the distance between "non functional" and "functional" is greater than the difference between "non functional" and "functional needs repair" or "functional needs repair" and "functional. We can then have the Bayesian model predict a number y, and then convert this to a label using a range, where the exact limits can be adjusted:

$$\text{functional: } 0.67 \leq y_i$$
$$\text{functional needs repair: } 0.33 < y_i < 0.67$$
$$\text{non functional: } y_i \leq 0.33$$

2. The assumption that the 3 labels lie on a linear spectrum is not necessarily a safe assumption, and the limits are set rather arbitrarily. A more natural and common method in machine learning is to build a classifier that decides between 2 classes. In this case, we can either build 3 one-versus-one classifiers or 3 one-versus-rest classifiers and use majority vote (or probabilities in the case of a tie) to make the final classification.

We use the first approach for its simplicity, although further work investigate the performance of the second approach, which is much more versatile at the cost of computational complexity.

During the training stage of our model, we also need to convert the given labels into numbers. Assuming that we use the first approach above for prediction, we use a similar scheme:

$$y_i = \begin{cases} 0 & \text{non functional} \\ 0.5 & \text{functional needs repair} \\ 1 & \text{functional} \end{cases}$$

3.2 Categorical data

Many of the data features are categorical data that do not have any numerical interpretation, so we cannot convert them the same way that we converted our label. For example, the "installer" feature has labels such as "UNICEF," "Roman," and "Artisan." To deal with these, we use one-of-k representation in which we add a new feature column for every unique value. For example, we would add the columns "installer=UNICEF," "installer=Roman," and "installer=Artisan." The "installer=UNICEF" would be 1 if that data's "installer" was "UNICEF," and 0 otherwise. So if there are k unique values for a particular categorical feature, then for each data point, we add on a vector of length k which has a single 1 and $k-1$ 0s.

3.3 The model

We model y_i using a logistic function that constrains the range to $0 < y_i < 1$, and the parameter for the logistic function is controlled by the features and weights, which are also parameters that we need to calculate. Our model comes from the assumption that there are certain factors impacting decay rate, and so the functionality is dependent on this decay rate. We then select features from the data which we have a prior belief to have a strong influence on functionality, choose priors on their distributions and construct a Metropolis-Hastings Sampler to select the optimum value for each parameter. New predictions of functionality can then be made from this model.

We also add on a term for noise, ϵ_i, which we assume is Gaussian noise controlled by the standard deviation σ.

$$y_i = \frac{1}{1+e^{\alpha_i}} + \sigma\epsilon_i$$
$$\alpha_i = \beta_0 + \beta_1 x_{i,1} + \beta_2 x_{i,2} + ... + \beta_n x_{i,n}$$

There are n features and $n+2$ parameters ($n+1$ βs and σ). The parameters are found by sampling from the posterior:

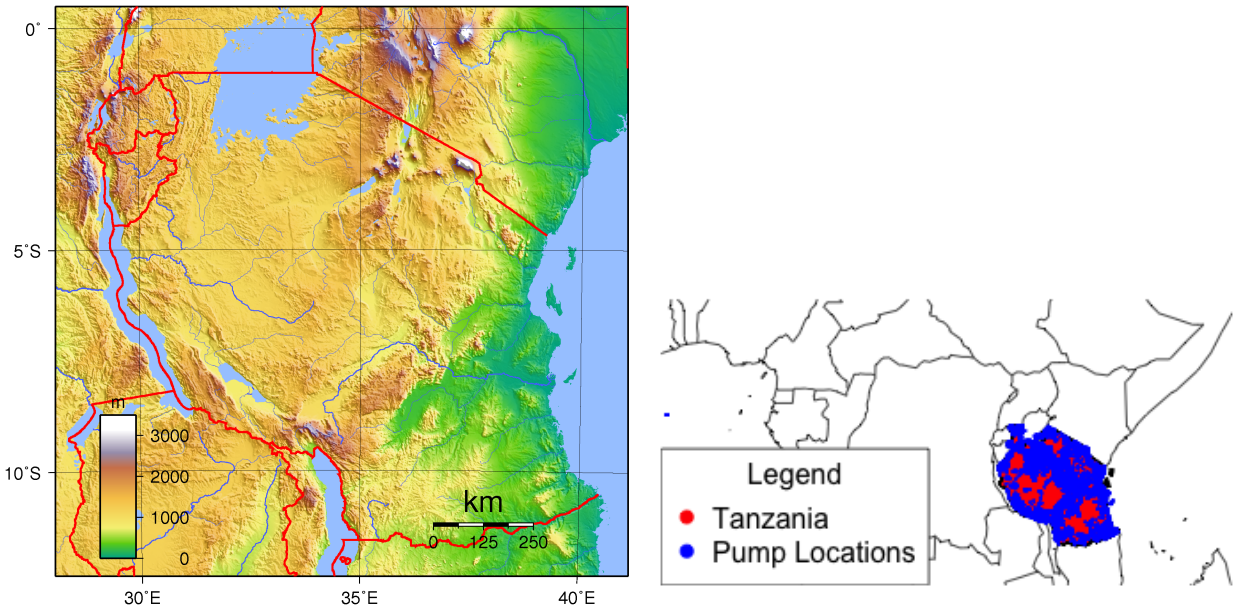
$$p(Y, \Theta) = p(Y|\Theta)p(\Theta)$$

4 Prediction Results

4.1 Results

Figure 2 shows the trace plot of the model parameters with blockwise updating and componentwise updating. These are after the burn-in period. Blockwise updating has an acceptance rate of nearly 0 because we are updating 8 parameters at a time. Componentwise updating has a much higher acceptance rate, although at a huge time cost.

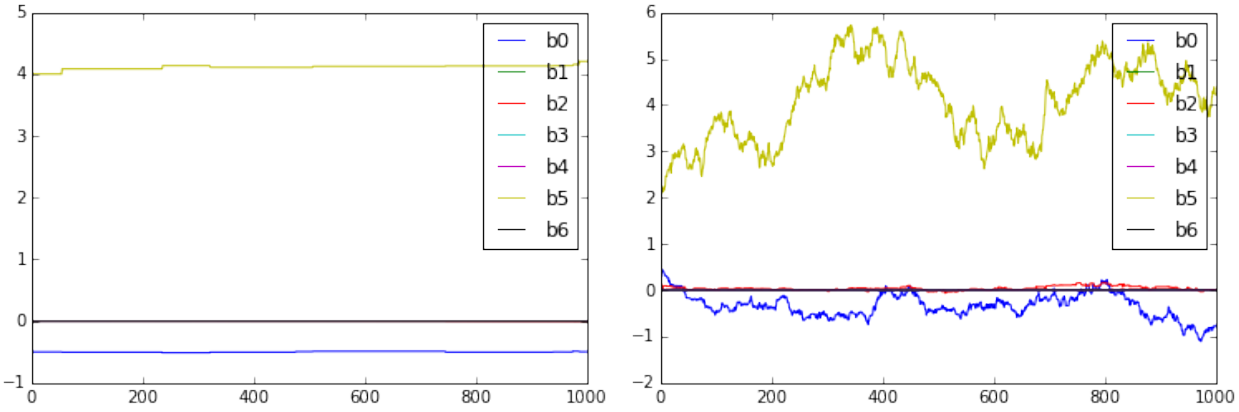
The parameters chosen are "longitude," "latitude," "age" (2015-"construction_year"), "gps_height," "quantity=dry," and "population."



(a) Physical geography of Tanzania.

(b) Distribution of water pumps.

Figure 1: Geography of Tanzania and distribution of water pumps.



(a) Blockwise updating

(b) Componentwise updating

Figure 2: Traceplot of model parameters, using MH sampler.

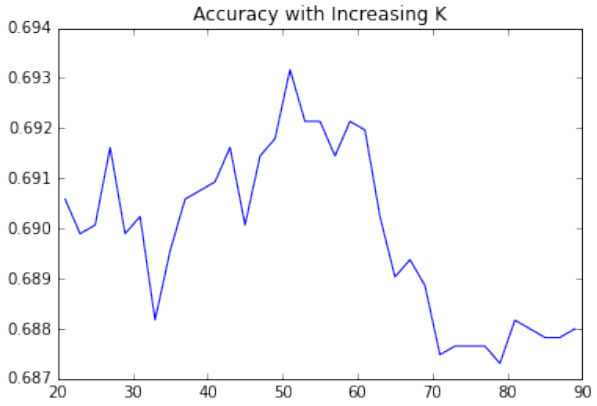
We see that β_5 has significant predictive power since this is the farthest from 0. This corresponds to the "quantity=dry" feature. β_0 is the constant offset, which we would expect to be non-zero.

By fine-tuning the step parameters and carefully choosing features, we reach a cross-validation prediction accuracy of 60.6%.

4.2 Comparison to machine learning approaches

Here we compare our methods with other common machine learning methods.

The k-Nearest Neighbors algorithm (also known as the k-NN algorithm) is a machine learning algorithm that can be applied to classification problems. There is no training aspect to the algorithm. Predictions are made by taking the k nearest data points to the desired data point in feature space, and taking the majority of their labels. The labels can also be weighted by $1/d$, where d is the distance from the desired data point to the labeled data point. We apply the k-NN algorithm using only the longitude and latitude data, so we are picking the geographically nearest neighbors. Accuracy in cross-validation is shown in the below figure as a function of k. We see that it reaches 69% prediction accuracy.



Decision trees are a machine learning method in which a tree of rules is built. One traverses down the tree, and each node represents a rule based on the data's features. The leaves are the final decision. The nodes are built based on which features have the most predictive power for classification. Random forests are an extension of decision trees in which many trees are trained, and stochasticity is introduced to make the trees different, which decreases bias. We show in randomforest_explore.ipynb that random forests on a small subset of features can reach an accuracy of 79%. Further experimentation and more sophisticated feature extraction can reach prediction accuracies of 83% on this data.

Bayesian modeling is probably not the best method to use for this type of problem. It assumes that the classification can be described by some sort of model, and we would have to guess this model ahead of time. Decision trees and random forests do not assume any sort of relation between the variable features and the labels, and merely seek out the variable features with the highest predictive power. Second, there are a lot of parameters in the MH sampler that need to be tuned, including initial guesses for the parameters and step sizes. Additionally, Bayesian modeling takes extremely long to train, on the order of hours on a laptop, assuming that the initial guesses for the parameters and step sizes are set correctly. Training k-NN and random forests take on the order of minutes.

5 Optimizing humanitarian aid delivery

In the second part of this project, we use the temporal/geospatial conflict occurence model in the first section as both inspiration for the aid delivery analogy and also as a source of randomly sampled data points representing geospatially distributed conflicts.

5.1 The traveling salesman problem

One question of particular interest is how to route emergency aid to locations where it is needed. For concreteness, let's postulate a Red Cross medical or food supply caravan that originates from the organization's in-country headquarters. This caravan wishes to visit all n emergent locations in order to deliver needed supplies. They wish to do so in the most efficient manner possible.

This is the traveling salesman problem (TSP), an optimization problem that is quite well known. It was first described in 1932 by Karl Menger (shortly after his year here at Harvard as a visiting lecturer) and has been studied extensively ever since.[7] Here is the traditional convex optimization specification of the problem:[8]

$$\min \sum_{i=0}^{n} \sum_{j \neq i, j=0}^{n} c_{ij} x_{ij}$$

s.t.

$$\begin{aligned}
& x_{ij} \in \{0,1\} && i,j = 0, \cdots, n \\
& \sum_{i=0, i \neq j}^{n} x_{ij} = 1 && j = 0, \cdots, n \\
& \sum_{j=0, j \neq i}^{n} x_{ij} = 1 && i = 0, \cdots, n \\
& u_i - u_j + n x_{ij} \leq n - 1 && 1 \leq i \neq j \leq n
\end{aligned}$$

As is clear from the constraints, this is an integer linear program (ILP) where:

- x_{ij} is a binary decision variable indicating whether we go from location i to location j.

- c_{ij} is the distance[1] between location i and location j.
- The objective function is the sum of the distances for routes that we decide to take.
- The final constraint ensures that all locations are visited once and only once.

The problem, of course, is that brute force solution of the TSP is $\mathcal{O}(n!)$. Traditional, deterministic algorithm approaches such as branch-and-bound or branch-and-cut are still impractical for larger numbers of nodes. In many cases, exhaustive search for global optimality is not even particularly helpful as long as the solution found is good enough. We will use simulated annealing (SA) to get acceptable solutions to the TSP.

Figure **??** shows a sample draw of conflict data (the blue points), and a near-optimal TSP route found through 50,000 iterations of simulated annealing.

5.2 Packing the aid truck — the Knapsack Problem

We extend the TSP into a multi-objective optimization problem where *the contents of the aid trucks* also have an optimization component. Therein lies the knapsack problem: subject to a volume or weight constraint, and given that different locations might have very different needs such as food, vaccinations, or emergent medical supplies, *which supplies do we pack on the trucks*?

Here's the unbounded[2] version of the knapsack problem:

$$\begin{aligned} &\max \sum_{i=1}^{n} v_i x_i \\ &\text{s.t.} \\ &\quad x_i \in \mathbb{Z} \\ &\quad x_i \geq 0 \\ &\quad \sum_{i=1}^{n} w_i x_i \leq W \end{aligned}$$

In this formulation:

- x_i is a zero or positive integer decision variable indicating how many units of item i we load on the truck.
- v_i is the utility we get from bringing along item i.
- w_i is the weight of item i.
- W is the maximum weight the truck can carry.

5.3 A brief detour for modeling assumptions

Before we can optimize this aid delivery mechanism, we will need to decide a way to model humanitarian aid needs at a given conflict.

Let us assume that there are K distinct types of humanitarian aid to be delivered. (Without loss of generality, we will use three categories for all of our examples — perhaps we can think of them food aid, first aid supplies, and medicines for concreteness.) We can model each conflict's aid needs as

$$\boldsymbol{x} \sim \mathrm{Dir}(\boldsymbol{\alpha})$$

where $\boldsymbol{\alpha}$ parameterizes the distribution to generate vectors of length K representing the relative proportions of needs.[9] For example, in our three category example we might draw the vector $(0.11,\ 0.66,\ 0.23)$ for a certain conflict, meaning that 11% of the aid needed at this conflict is food aid, 66% is first aid supplies, and 23% is medicines. Now that we know the proportions for the given conflict, how might we turn this unitless vector into absolute amounts?

For that reason, let's assign each conflict a scaled size $s \in [1, 10]$ based on the number of casualties (a proxy for the severity of the conflict). We can use this size scalar to turn our proportion vector into a vector of absolute needs.

It should be noted that **both of these modeling methods for proportions and size are "plug-and-play"** — because of purposely designed loose coupling in our model, these methods could trivially be replaced by a different method of calculating or predicting the needs of each conflict. For example, if an independent model was used to calculate each of K needs based on the features of each conflict, those quantities could easily be plugged in to this model. Ultimately, the only quantities that our TSP/Knapsack model needs is an $n \times K$ matrix of aid needs for n cities and K categories of aid.

5.4 A new objective function to integrate TSP and Knapsack

For the vanilla TSP, we simply try to minimize the total distance. Now that we are adding a new objective, we will need to integrate the two into a coherent **loss function**. Here is the function we will actually try to minimize in the combined TSP/Knapsack:

$$L(\boldsymbol{x}) = \text{total distance} + \text{sum of squared aid shortfalls}$$

[1] In our application, we deal with geospatial data on a large enough scale that the Euclidean distance is actually very imprecise. In order to model distances over the planet's surface, we use the Haversine formula.

[2] Often, this problem is formulated such that you can only bring one of each item, but that does not make sense in our application. Rather, we want to be able to bring as many types of each type of aid as we think necessary, and we'll assume that as many as desired are available to load on the trucks before starting out from HQ.

The effect of squaring aid shortfalls acts as a weight, causing greater importance to be placed on minimizing this aspect of the problem first. Proposals wherein aid shortfalls occur are heavily penalized. As we will see in later graphs, once the SA algorithm is able to avoid all shortfalls and the concurrent massive loss function penalties, a much slower descent begins to take place wherein the distance is slowly optimized. See figure **??** for a depiction of this phenomenon.

5.5 Implementing the Knapsack aspect

Figure **??** shows the same draw of cities as in figure **??**, this time factoring in limited carrying capacity for aid supplies on the aid delivery mechanism and using our new loss function. As we can see, the huge penalty incurred when supplies run out quickly induces the simulated annealing algorithm to converge on a solution with multiple stops at HQ to reload.

Figure **??** uses some uniformly distributed points on the $[0, 50]$ plane to demonstrate how the proposed TSP/Knapsack routes converge as the number of iterations increases.

5.6 Finding the optimal site for the resupply location

Our initial assumption was that the HQ was located in the capital city of Kampala. However, we should ask whether our HQ could be more conveniently located. We can answer this question by treating the reload location as another parameter and continuing to sample HQ locations using SA. Figure **??** shows the TSP/Knapsack optimized once again, this time using a the optimal HQ location, while **??** compares the loss function as each method converges to its best possible configuration.

Conclusions

In each part of this problem, analytical solutions either do not exist (e.g. in the distribution of events) or are computationally infeasible (e.g. in the TSP/Knapsack optimizations). We found that using a metaheuristic such as SA converged on robust solutions in relatively short order. In the future, we would like to formulate our loss function based on real world data based on refugee locations and aid distribution requirements; our methodology would not change, but the solutions would be more useful for predictive tasks. Additionally, the separate models could be fit and incorporated which realistically model how much of each type of aid is needed at each conflict location. Future research might also include adding many more constraints or twists to the problem, and trying different stochastic optimization techniques such as genetic algorithms, Tabu search, or ant colony optimization.

References

[1] UN Water, "World Water Day Day 2013 - Facts and Figures," *www.unwater.org*, Checked 8th May 2015.

[2] J. Morisset, "Tanzania: Water is life, but access remains a problem," *blogs.worldbank.org*, September 2012.

[3] J. Nakamura, "Predicting time-to-failure of industrial machines with temporal data mining," Master's thesis, University of Washington, Dept. of Computing and Software Systems, Tacoma, WA, 2007.

[4] Y. Liu, *Applying A Data Driven Approach To Failure Prediction For Rod Pump Artificial Lift Systems*. PhD thesis, USC, Dept. of Electrical Engineering, Los Angeles, CA, 2013.

[5] *http://www.drivendata.org/about/*, Checked 8th May 2015.

[6] *http://taarifa.org*, Checked 8th May 2015.

[7] K. Menger, "Das botenproblem," *Ergebnisse eines Mathematischen Kolloquiums*, vol. 2, pp. 11–12, 1932.

[8] W. Winston, *Operations Research: Applications and Algorithms*. Thomson Brooks/Cole, 2004.

[9] K. P. Murphy, *Machine Learning: a Probabilistic Perspective*. Cambridge, MA: MIT Press, 2012.