2010 CS229 Final Project

BATTERY MANAGEMENT IN DATACETNER USING MACHINE LEARNING

Hyun Goo Kang

1. Introduction

To process billions of web pages and serve millions of its users, Google runs a huge number of machines in its datacenters. And a large number of those machines fail every day. While those failures can cause Google's services a lot of troubles, Google does many things to keep its services stable. The Google File System (GFS), for example, replicates its data across machines to tolerate various hardware failures [1].

At a lower level, Google has come up with another brilliant way to make things reliable; our servers run with backup batteries. Power failure is very common in datacenters and it can cause machines to suddenly go down. When an entire datacenter experiences a short glitch or unstable electricity, a common example, all its machines may go down and all the services running on those machines would become unavailable for a while. Google would not only lose all the revenues it could make during the down-time, but would also hurt its reputation on reliability badly – serious money.

Batteries can keep those machines up during the power failure. Even when power do not come back in time, backup batteries would give applications enough time to prepare for a clean shutdown and prevent data corruptions. As I mentioned earlier, it can save some serious money.

Unfortunately, batteries are not very reliable. When a short power failure happens, a number of batteries may fail to keep their machines alive and it would cause some service disruption – battery failure. When batteries are not managed at all, battery failure rate can go up pretty high and make things quite unstable. In order to keep battery failure rate low, we replace bad batteries with the new ones to keep them fresh. Unfortunately, it is very difficult to know which battery is bad. And replacing batteries recklessly would waste a lot of good batteries and money.

In this paper, I propose a battery management scheme using machine learning, Machine learning models will find batteries that look bad, and we will replace them to keep our batteries operating. This knowledge would allow us to remove smaller number of batteries and while keeping the failure rate at a low level. The goal of this management scheme is to keep our datacenters healthy while minimizing the cost of battery replacements. To find a good middle point, we bring in cost estimation models.

Please note that most of the numbers and details about batteries and datacenters at Google will be omitted in this paper, for confidentiality reasons. This paper has been reviewed by my peers at Google before being submitted.

2. Cost Estimation Models

a) Battery Failure Cost

Battery failure cost can be estimated as Fig. 1 on the next page. When battery failure rate is below certain level (region a), applications will not experience any noticeable hick-ups as Google's services, such as GFS, are built fault-tolerant. A number of machines will go down for some time, but its impact would be negligible. When the failure rate goes above certain level (region b), however, services would start experiencing bad disruptions. Let's call it 'catastrophe threshold'. When 20% of machines go down, for example, 0.8% of 3-times replicated data on GFS would become unavailable [1]. This means a lot of users accessing the data would be affected by the failure. Considering the revenue Google is making with its services, such disruptions will cost Google HUGE money both in

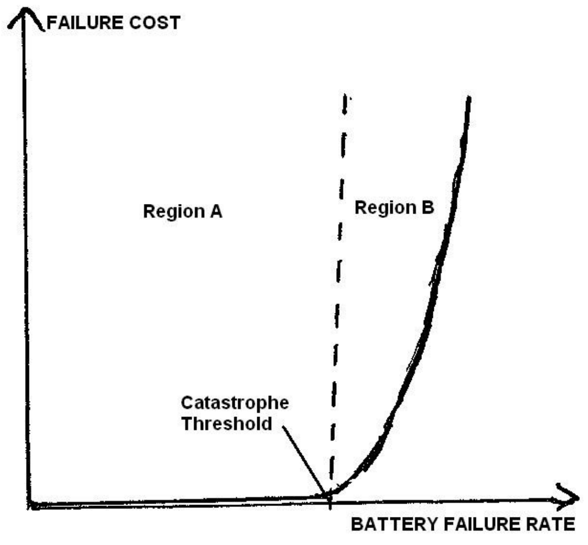


Figure 1. Failure cost vs. failure rate

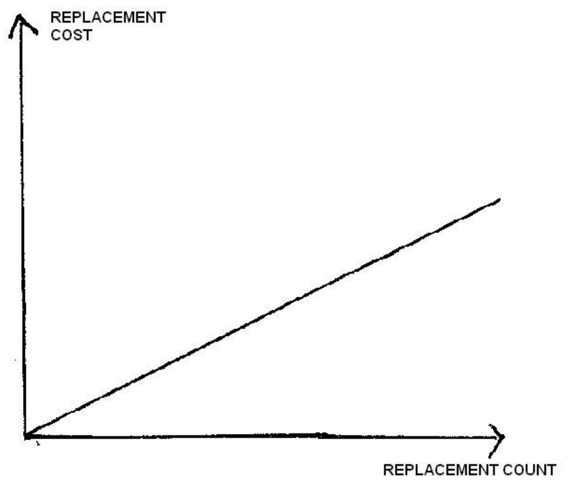


Figure 2. Replacement cost vs. replacement rate

short-term and long-term. Also, the impact of battery failures would grow exponentially with failure rate. Batteries should be managed properly so that its failure rate would NEVER exceed the catastrophe threshold.

b) Battery Replacement Cost

Battery replacement cost can be estimated as Fig. 2. Replacing a battery can cost us the followings: 1) cost of a new battery, 2) cost of replacement labor and 3) machine down time during replacement. At most cases, these costs should remain near-constant per replacement. Here we assume we do not replace batteries at a rate that could cause serious down-time.

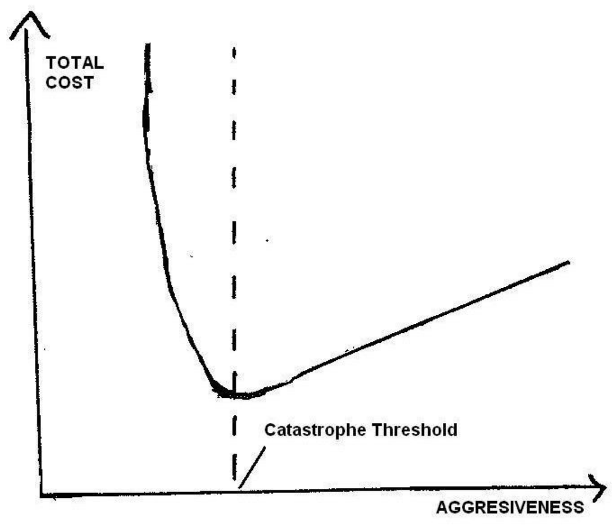


Figure 3. Total cost vs. replacement scheme

c) Overall Cost

Replacing batteries too conservatively would cause battery failure cost to go up exponentially, while replacing them too aggressively would cause unnecessary replacement cost (fig. 3). In order to minimize the overall cost, we would have to replace batteries at an appropriate level so that we would always keep the failure rate below the catastrophe threshold level, while keeping the replacement cost at the minimum. So far, smart engineers at Google have come up with some replacement heuristics to keep the failure rate below the threshold.

However, we can further reduce the cost by using a more accurate battery failure prediction model. Assuming that batteries would become bad at a constant rate (say X batteries/day), we would have to replace X bad batteries every day to keep our failure rate low. It would give the following cost calculation:

By having a more accurate model (with higher p), we can reduce the total cost. Having a model with lower variance would also help us keep the failure rate stable.

In this paper, I propose the following management scheme. From the battery test results, we train a machine learning model that predicts the whether a battery is likely to keep a machine up. Once we have an accurate model, we look at our batteries and replace the ones that are more likely to fail than others. The replacement-triggering threshold should be tweaked adaptively, so that we would not replace too little nor too many batteries in our datacenter. We can estimate how bad our batteries are from recent battery test results (i.e. test failure %) and adapt our replacement threshold to that.

3. Battery Modeling

In order to build the battery management scheme, we first need to build a battery failure prediction model. First of all, let’s represent the status of a battery as “B”. B should represent the internal (chemical and physical) status of a battery and it should determine how the battery will operate at a given circumstance. Unfortunately, this is a hidden variable - it cannot be measured.

Instead, we would like to look at some values we can measure from outside. Voltage and charge current of a battery can be easily measured (Fig. 4) and they should be determined by battery status (B) and other variables such as surrounding temperature and fullness (how fully charged a battery is) of a battery. We simplify these and represent them as the followings:

V: Voltage, I: Charge Current, %: Fullness
T: Temperature around the battery

$$V = Fv(B, T, \%$$
$$I = Fi(B, T, \%)$$

Another data we have is battery test results. At Google, we periodically run load tests on batteries and store the results (= whether a battery managed to keep a machine up during the test). Here, we define a

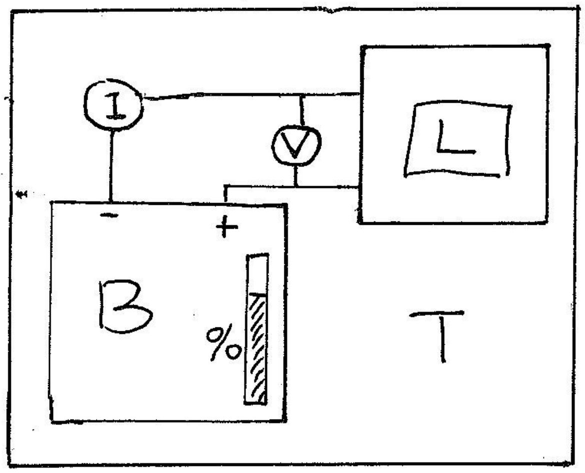


Figure 4. Battery Model

variable P: probability that a battery would pass a test (= keep its machine up). This should be determined by the status of the battery and other factors, such as load and surrounding temperature. Higher load on a machine would discharge a battery faster and increase its failure probability. Temperature is also known to affect a battery's performance [2]. We make the following assumption on P:

L = load running on its machine

$$P = Fp(B, T, L, \%)$$

One thing to notify here is that we keep our batteries full most of the times. To simplify the equations, we assume % to be full all the time and remove the variable from the equations:

$$V = Fv'(B, T), I = Fi'(B,T)$$
$$P = Fp'(B, T, L)$$

Later when we train our machine learning models from our test results, we can enforce this assumption by leaving out the tests done with not-fully-charged batteries - where batteries have been newly inserted or discharged in the near past.

Unfortunately, we still have B, a hidden variable, in our equation. We need make a bold yet reasonable assumption to replace B with measurable values, V and I. Let's assume that B can be estimated from V, I and T:

$$B \sim Fb(V, I, T)$$

This leads to:

$$P = Fp'(B, T, L) \sim Fp'(Fb'(V, I, T), T, L)$$
$$= Fx (V, I, T, L)$$

With this assumption, we are going to build a machine learning model to predict P from V, I, T and L.

4. Model Training

a) Choosing a model

One problem with the model we just proposed is that in contains external variables, T and L that are not related to a battery’s status. This makes things trickier, as we would not want to replace a battery when it has no problem but is likely to fail due to high temperature and high load. We would have to exclude and normalize such external factors and replace the batteries that are more likely to fail in general.

As it would be very difficult to achieve this with a classification model, I decided to use regression models that would predict the battery failure probability. With an accurate regression model, we should be able to compare a battery at a certain situation against how an average battery would perform under the same external condition. This should give us a good estimate on how bad a battery is (i.e. <a battery's failure rate> / <avg failure rate under the same condition>). From these, we can simply replace the top N% of the batteries with the highest "badness" estimate.

b) Battery Data

To train a model, we create a data instance from each battery test result. We use voltage, charge-current and temperature values measured right before the test as input features. Load of a machine could not be used as a feature as its value changes very fast over time and is unpredictable.

Measurement data is not stable at all. There were a lot of meaningless and flawed measurement data and I had to carefully filter them out. For the time-series data, such as V, I and T, I created a number of features by calculating average, hi/lo, median and std deviation of the data over intervals, with different interval sizes.

c) Model Selection

For each battery type, I trained logistic regression and support vector regression models with grid-search parameter selection and forward-search feature selection techniques [3,4]. Linear, polynomial and RBF kernels were used for SVR. Features have been scaled be improve the models. To find the right parameters and features, parameters and features have been tested on a small (500~1000 examples) dataset with equal positive and negative data sizes. For validation, 5-fold CV was used. SVR with linear and polynomial kernels were dropped, as they ran much slower than RBF kernel, showing similar accuracy. Different battery types show different characteristics and different features were selected for battery types.

Model \ Battery Type	Type A	Type B
Logistic Regression	68.15%	74.78%
SVR w/o feature selection	74.0%	83.6%
SVR w/ feature selection	72.5%	82.2%

Table 1. 5-fold cross-validation accuracies. Logistic regression was done with 4000 data samples and SVR was trained with 1000 data samples.

d) Applying the model

As services have been designed fault tolerant and can live with some failures, false-negative rate (not catching some bad batteries) is not our major concern. Instead, we would like to reduce false-positive rate.

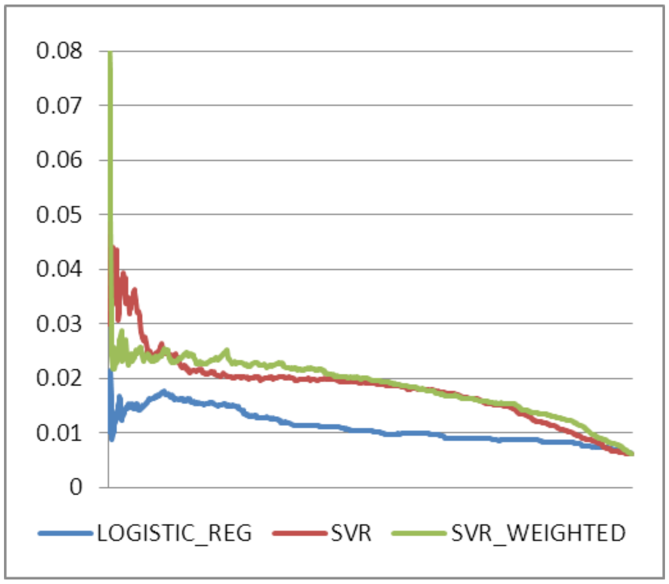
As batteries are mostly good, we have a lot more negative (pass) instances than positive (fail) instances. This makes our models very conservative and increases false-positive rate. To address this, I weighted positive cases by a weight parameter "w" and which was roughly optimized with data distribution.

Here I would like to show that our machine model performs better than the heuristics we have set up before. One of our heuristics shows the following accuracy:

	Type A		Type B	
	Recall	Prec	Recall	Prec
Heuristics	0.02%	2.2%	0.11%	32%
SVR	0.02%	5.8%	0.11%	100%
	18%	2.2%	23%	32%

Table 2. Recall and Precision of positive class (fails)

With the same precision or recall, our model shows a lot better coverage/ accuracy. We can also compare our own machine learning models and see how weighting influences false-positive rate. Fig 5 shows that SVR performs much better than logistic regression in all cases. For battery type B, our model has shown quite astounding prediction result, surpassing our former heuristics by far. Even for type A, we have been able to predict with 2~3x more accuracy than mere random selection. This would save us 2~3x good batteries!



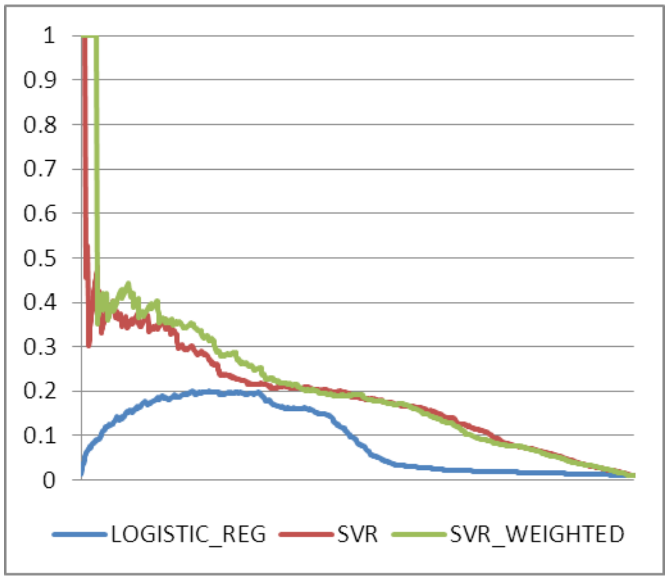


Figure 5. Precission recall curves for battery type A and battery type B.

5. Conclusion and future work

We have been able to predict more accurately than ever with machine learning models. Having an accurate prediction model would help us keep machines more stable and also reduce the number of good batteries being replaced. It would be interesting to extend the models into a concrete replacement schemes in our future work.

References

[1] Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung, The Google File System, 19th ACM Symposium on Operating Systems Principles, October 2003

[2] Anton Andersson, Battery Lifetime Modelling, Umeå University, January 2006

[3] Chih-Chung Chang and Chih-Jen Lin, LIBSVM : a library for support vector machines, 2001. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm

[4] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. LIBLINEAR: A library for large linear classification Journal of Machine Learning Research 9(2008), 1871-1874.