Tracking Time-Varying Hidden Faults using Stochastic Gradient Descent

John C. Platt, Emre Kıcıman, David A. Maltz
Microsoft Research, 1 Microsoft Way, Redmond WA 98052
{jplatt, emrek, dmaltz}@microsoft.com
http://research.microsoft.com/{~jplatt, ~emrek, ~dmaltz}

Summary

In computer systems, hidden faults can stochastically produce observable symptoms. These faults can be transient and a symptom can be produced by multiple possible faults. We introduce an algorithm that takes a time series of the presence or absence of observed symptoms and produces a current estimate of the probability over faults. We use this algorithm to isolate problems on the Internet that interfere with Microsoft's web servers delivering content. The algorithm is very scalable: we have done analysis of 1 billion observed HTTP requests, with over 10,000 possible hidden faults.

The Problem

Consider the following mathematical model. You have N hidden coins, each will come up heads with some probability, q_j. Each of these coins corresponds to a hidden fault. We cannot observe the outcomes of the coins: the coins come in bags, where the same coin can lie in more than one bag. For each set of coin flips, someone will look in one bag of coins and tell us whether at least one of the coins is heads, or whether all of them are tails. Observing the bag corresponds to observing a symptom, where the symptom can have multiple possible underlying causes. The results of different bags are reported, repeatedly. How should we estimate the underlying probability of coin flips, q_j?

The Algorithm

We estimate the parameters q_j with MAP estimation: $\max_{\vec{q}} \log P\left(\vec{q}|\vec{d}\right) = \max_{\vec{q}} \left[\log P\left(\vec{d}|\vec{q}\right) + \log P(\vec{q})\right]$. We use a noisy-OR model for each bag of coins, then assume that the bag observations, d_i, follow a binomial distribution governed by the probability of each bag being true, p_i. We use a beta prior for the coin flips. We then find $\max_{\vec{q}} \sum_i d_i \log p_i + (1-d_i)\log(1-p_i) + \sum_j a_j \log q_j + b_j \log\left(1-q_j\right)$, where $p_i = 1 - \prod_{j\in B_i}(1-q_i)$; B_i are the coins in the ith bag; and a_j and b_j depend on the parameters of the beta distribution. For numerical stability, we estimate the log odds, z_j, of a coin flip: $q_j = 1/(1+e^{-z_j})$.

We decided to use stochastic (on-line) gradient descent (SGD) to find the q_j that maximize the posterior probability $P\left(\vec{q}|\vec{d}\right)$. An unbiased estimate, G_{ij}, of the gradient w.r.t. z_j is estimated by taking the gradient of the observation likelihood with respect to one observation i, and dividing the gradient of the prior by L, the number of observations:

$$G_{ij} = \left.\frac{q_j(d_i - p_i)}{p_i}\right|_{j\in B\_i} + \frac{1}{L}\left(a_j - \left(a_j + b_j\right) q_j\right).$$

In system applications, the observations are noisy and frequent. We use SGD with momentum in order to smooth the estimates: it is important to maintain low-noise estimates of q_j in order to compute a good gradient G_{ij}. The update is then $S_{ij} \leftarrow \alpha S_{ij} + (1-\alpha)G_{ij},\ z_j \leftarrow z_j + \eta S_{ij}$.

Discussion

One strong advantage of SGD over other optimization algorithms is that it provides an on-line estimate of the underlying fault probabilities, . In systems applications, the true underlying fault probabilities change over time and tracking these changes is critical to the operators managing the system. SGD leverages a nice property of systems applications: symptom observations are plentiful, with many observations per second. We track the fluctuating fault probabilities by updating the fault probabilities once per symptom observation, and then report the current estimate.

Note that this problem is different from the standard medical diagnosis problem, which is commonly phrased as finding the posterior marginals of a fault vector given *one* observation of a set of symptoms. In our problem, we are flooded with symptoms: we need a fast algorithm to exploit multiple measurements. We also can exploit these multiple measurements to accurately determine the underlying marginal posterior probabilities.

The Application

As a web content provider, Microsoft wants to provide content reliably to our customers. This does not always happen: an HTTP connection to Microsoft can fail if any of a number of faults exists. For example: a specific Microsoft server may have a problem; the client software accessing the page may be a robot or worm with a broken HTTP implementation; or the client's Internet service provider (called an Autonomous System (AS)) may have lost connectivity to Microsoft because of a routing problem on the Internet.

We use SGD to dynamically determine the cause of failed HTTP connections. Each HTTP connection is modeled as a bag of coins (potential faults). These faults include the server identity, the browser type, and the client's AS. Each HTTP connection either succeeds or fails (corresponding to an outcome of a bag of coins). We update the probability of each fault once for every successful or failed HTTP connection. We use a step size of 0.1 and a momentum of 0.999 and zero strength on the prior.

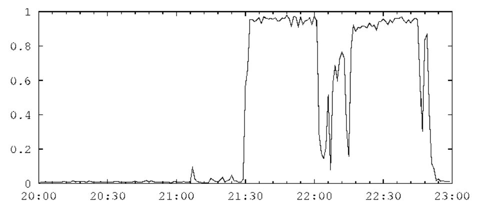
Experimental Results

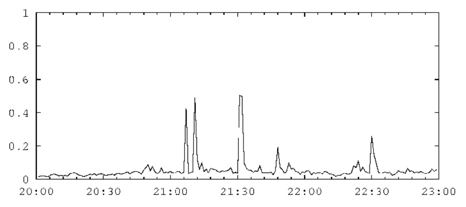
First, we compared SGD to a full numerical optimization of the data log likelihood (with BFGS), to check for optimization quality and speed. We created a vector of 200 faults, each with occurrence probability drawn from a Beta(0.05,1) distribution. We then created 100,000 observations, each connected at random to the 200 faults with noisy-OR links of probability 0.05. The RMSE of the MAP estimate to the true occurrence probabilities and the elapsed time is shown below. For this experiment, the code is written in Matlab.

	Guess the prior	**BFGS**	**SGD, 1 epoch**	**SGD, 20 epochs**
RMSE	0.1637	0.0039	0.0178	0.0122
Elapsed time (sec)	0	1956	7.4	147

The full optimization is more accurate, but between 13 and 264 times slower, depending on the number of epochs (passes through the data) allowed for SGD. Note that SGD and BFGS both accurately estimate the high probability faults, but SGD rarely provides an estimate lower than 7×10^{-3}, while BFGS produces estimates as low as 10^{-7}.

We tested SGD on a day when MSN had observed many failed connections. SGD localized the major problem to 2 ASes, in the same geographical area (see plots below), which accounted for 95% of the HTTP problems. The left figure shows the fault probability of one of the 2 ASes that caused the major problem, while the right figure is for a third AS that had intermittent problems throughout the day.


1
0.8
0.6
0.4
0.2
0
20:00
20:30
21:00
21:30
22:00
22:30
23:00


1
0.8
0.6
0.4
0.2
0
20:00
20:30
21:00
21:30
22:00
22:30
23:00