A Method for Detecting Convergence of Gradient Descent with Cauchy-Style Estimates

Galen Novello

February 12, 2017

Abstract

In this paper I propose a method for detecting the convergence of the method of gradient descent with "Cauchy style" estimates. Brief overviews of the method of gradient descent and Cauchy sequences are provided, followed by a discussion of how the data for this paper was generated and some initial data. The main result of the paper is the conjecture that the method of gradient descent has converged on iteration k if $\forall\ k - 4n \leq j, m \leq k$ the inequality $|\theta_{i_j} - \theta_{i_m}| < 5\alpha$ holds. Empirical evidence supporting this conjecture is presented and ideas to formalize and extend this research are given.

1 The Method of Gradient Descent

The method of gradient descent is a regression algorithm that works to minimize a cost function by using the gradient to iteratively adjust the model. A good introduction to the method as well as an outline of it's justification can be found in open source resources from Andrew Ng's Machine Learning Course, and it was these materials which guided me to creating the gradient descent algoritm used in this paper.

To produce a model some for quantity of interest y as a function of variables $x_1, ..., x_n$ the the method seeks coefficents $\theta_0, \theta_1, ..., \theta_n$ of the line

$$y = \theta_0 + \theta_1 x_1 + ... + \theta_n x_n.$$

The error of a model $\Theta = (\theta_0, \theta_1, \ldots, \theta_n)$ is given by the cost function

$$J = \frac{1}{2n} \sum (y - X \cdot \Theta)^2$$

where the sum is taken over all points $(y, x_1, x_2, \ldots, x_n)$ in training data set and for each point we set $X = (1, x_1, x_2, ...x_n)$ for notational convenience.

On each iterateration the method takes a hypothesis $\Theta = (\theta_0, \theta_1, \ldots, \theta_n)$ and computes the gradient of the cost function of using this hypothesis, DJ. A new hypothesis is then produced by taking $\Theta_{\text{new}} = \Theta - \alpha DJ$ where $\alpha \in \mathbb{R}$, know as the step size, is a suitable constant. In this way a sequence θ_{i_k} is produced for each coefficient θ_i and it is clear that convergence of the model is equivalent to convergence of each of these sequence pointwise.

2 Caucy Sequences

Cauchy Sequences are a well known tool for analyzing convergence of sequences. Most people first encounter them in the context of real numbers in an introductory course on real analysis or topology but the equivilence of being a Cauchy Sequence and being a convergent sequence is so important that it gets its own name and we call a topological space with such an equivalence complete. You can find rather exotic examples of complete spaces in the literature, but for the purposes of this conversation it will suffice to point out that metric spaces, and in particular $\mathbb{R}^n$, are complete.

A sequence θ_i is said to be a Cauchy sequence if for any $\epsilon > 0$ there is some N for which $|\theta_i - \theta_j| < \epsilon$ for all $i, j > N$. The idea of this paper is to adjust this definition in two ways to create a method to detect the convergence of the coefficients of gradient descent

- By changing the condition "for all $i, j > N$" to "for all i such that $k - k_0 < i < k$ " where k is the index of the most recent iteration and k_0 is a "window size" that depends on the number of variables in the model.
- By finding a bound on ϵ that will ensure (at least a high probability of) convergence of the model.

3 Initial Data, visual estimes of convergence time for different numbers of variables.

3.1 How pseudo random data for these tests is generated.

Convergence time of the method of gradient descent is affected by the spread of the data and the step size α. In fact, the method can diverge if α is chosen to be too large relative to the spread of the data. Data is generated for these test by first generating a vector, M, of constants uniformly from $[-1, 1]$ - this is the target function, the "answer" that the method will try and reconstruct.

Each data point $(y, x_1, x_2, ..., x_n)$ is then generated by picking each x_k uniformly from $[-1, 1]$ and computing y through the formula $y = M \cdot X$ where $X = (x_1, ..., x_n)$.

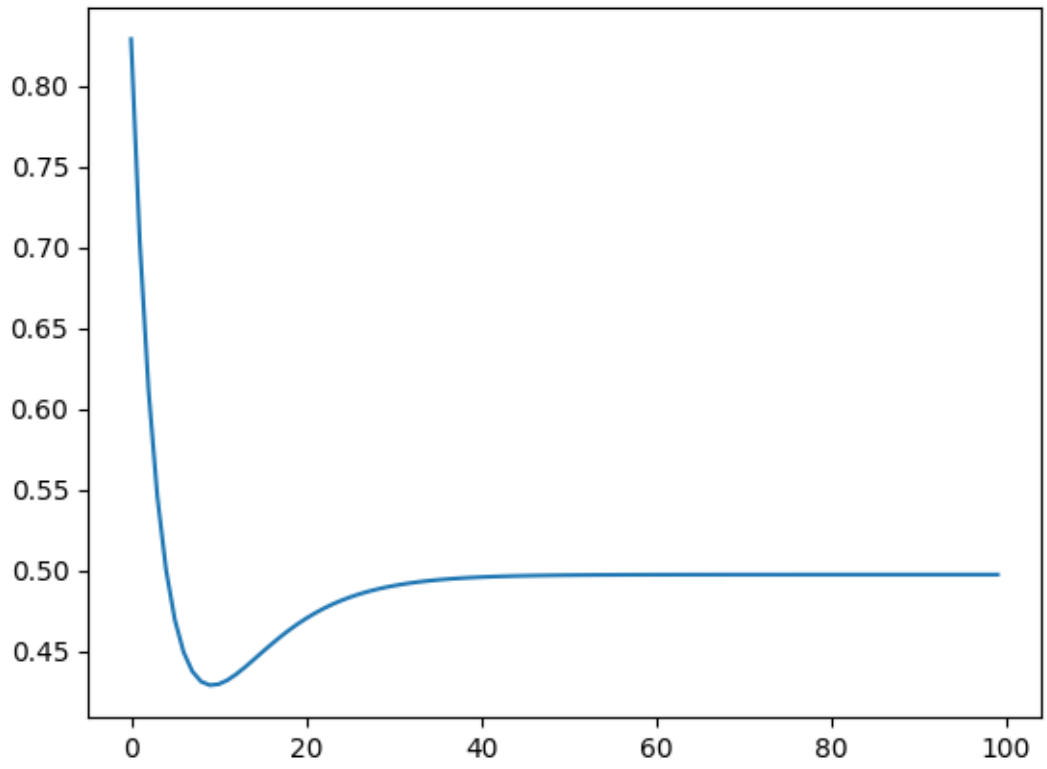
Convergence of the algorithm is affected if the interval $[-1, 1]$ is changed in either case. It is the suspicion of the author that the ratio of the lengths of these intervals is relevant to choosing the step size α, and perhaps it is worth noting that in this case that ratio is 1.

A set of 500 data points is generated in this way, 50 of them are removed for a final test set and the remaining 450 form the training set for the data. α is choosen as .01.

3.2 Data on the rate of convergence at randomness level blank for different numbers of variables

The convegence of the algorithm can be visualized by plotting the values of the θ_i vs the number of iterations. For example, the convergence of θ_1 for a 9 variable gradient

descent algorithm is given



With this particular model convregence can further be determined by the target function to the model which in this case gave 7 digits of accuracy after 100 iterations ($\theta_{real} = 0.48810335753783995$, $\theta_{predicted} = 0.48810336800164367$). From the graph it is also clear that almost as good an approximation would have been obtained with half as many (or fewer) iterations.

Running 30 test for each number of variables from 2 to 11 and analyzing the convergence of θ_1 (the algorithm is symmetric in θ_i so there is no harm in this) leads to the following estimates for the number of iterations to by which we expect convegence to begin.

# of vars	estimate
2	does not converge with this alpha
3	10
4	12
5	15
6	20
7	25
8	30
9	35
10	40
11	40

These are, of course rough estimates, based on a small sample, but they appear to indicate a pretty linear growth and lead to the initial hypothesis to look for convergence after $4n$ iterations where n is the number of variables.

Another look at the graph shows that the distance between the local min on the curve and the value to which it converges is less than .1. This was true for all 270 models that converged, so we expect that if all point start to tend to be within .1 of eachother the curve is on it's way to convergence. In this case α is .01 so our bound of .1 can be though of as 10α. We give a silightly more conservative initial estimate: that we can choose ϵ to be 5 α.

Putting it together we conjecture that the method of gradient descent has converged on iteration k if

$$\forall\ k - 4n \leq j, m \leq k, \quad |\theta_{i_j} - \theta_{i_m}| < 5\alpha$$

4 The Cost of Implementing the Condition

To implement this condition it will at worst be necessary to keep a matrix of the n+1 values of each coefficient over 4n iterations and checking them which means the cost of implementing the proceduce is on the order n^2 checks per iteration in particular this is CONSTANT with respect to the size of the training set. Each iteration of the gradient descent method requires computations at least on the order of n times the number of data points in the training set, so unless the number of variables is greater than the number of data points (which probably wouldn't converge anyway), implementing the test required above will lower the computational cost of running the algorithm (significantly).

5 Empirical Data

To gauge the effectiveness of this method to detect convergence 30 more simulations for each number of variables from 3 to 11 were performed, and the result of the Gradient Descent after 100 iterations and after the Cauchy cutoff were compared. On average the theta from the Cauchy estimate agreed with the real value of the coefficient to within .0001 for low numbers of variables and .001 for larger. The following table gives an idea of the averge number of computations for caucy convergence and to what accuracy:

# of vars	average iterations to converge	Cauchy accuracy
3	15	.0001
4	21	.001
5	27	.001
6	33	.001
7	38	.001
8	45	.001
9	49	.001
10	55	.001
11	65	.001

The accuracy of the coefficients from 100 iterations continued to be within .00001 at 11 variables. So there was a loss of a couple places of accuracy, but about a 33% savings in computation. Probably not a bad trade off in most situations.

The entirety of the data referred to above as well as the source code used to generate it and a copy of this paper can be found on GitHub .

6 Proposals for Future Research

- Refine Estimate of computation savings
- Provide analytic justification/refinement of estimates for k_0 and ϵ.
- Make precise connection between size of alpha, intensity of randomness, and convergence