# Regression with Gradient Descent

October 1, 2015

## 1

First, we discuss variations on gradient descent, including analytic gradient descent, finite difference gradient descent, and Matlab's `fminunc`, which includes some variations such as adaptive step size, using a quadratic approximation instead of linear approximation, and using a 2-dimensional subspace to reduce computational complexity. First, we'll examine the analytic gradient descent, which relies on having an analytic gradient expression at all points in space.

We'll refer to three example functions: the $n$-dimensional quadratic "bowl", $Q_n$, the $n$-dimensional inverted Gaussian (centered at $\mu = 0$, with $\Sigma = \mathbf{I}_n$), $N_n$, and the $n$-dimensional sum of sin's, $S_n$. These are defined as follows (leaving off the normalization constant on the inverted Gaussian for simplicity):

$$Q_n = \|\mathbf{x}\|^2$$

$$N_n = -\exp\left(-\frac{1}{2}\|\mathbf{x}\|^2\right)$$

$$S_n = \sum_{i=1}^{n} \sin(x_n)$$

Next, we benchmark the gradient descent variations on these two functions (seeding with an initial guess of $(1, 1, \ldots)$):

Table 1: Iterations (step = .1, threshold = .001)

| $n$ | Analytic Gradient $Q_n$ | $N_n$ | $S_n$ | Finite Differences $Q_n$ | $N_n$ | $S_n$ | `fminunc` $Q_n$ | $N_n$ | $S_n$ |
|---|---|---|---|---|---|---|---|---|---|
| 2 | 16 | 33 | 46 | 16 | 33 | 46 | 6 | 15 | 18 |
| 5 | 18 | 60 | 50 | 18 | 60 | 50 | 12 | 48 | 36 |
| 20 | 21 | 1 | 57 | 21 | 1 | 57 | 42 | 21 | 126 |
| 100 | 25 | 1 | 64 | 25 | 1 | 64 | 202 | 101 | 606 |

Focusing on just $S_n$, and leaving the step size, starting point, and threshold the same unless otherwise noted, we may also examine the effects of each variable (defaulting to $n = 5$:

Table 2: Examining the effect of other variables

| Step Size $\Delta$ | Iterations | Starting Point $x_0$ | Iterations | Threshold $\delta$ | Iterations |
|---|---|---|---|---|---|
| 2.0 | 177 | 0.001 | 49 | 0.01 | 39 |
| 1.0 | 5 | 0.01 | 49 | 0.001 | 50 |
| 0.1 | 61 | 0.1 | 50 | 0.0001 | 61 |
| 0.01 | 503 | 1.0 | 61 | 0.00001 | 72 |

We notice that the finite differences and analytic methods yield the exact same number of iterations for all functions. This can be explained by examining the diagonal of the Hessian matrix for

each of the functions - in all cases, the values are small, indicating low curvature, and thus that the function can be well-approximated by the linear finite difference approximation. Therefore, when examining the effects of other variables, we only present the numbers using the finite-difference gradient (the analytic gradient produces identical numbers). We may draw a few conclusions by examining the effects of other variables. As we decrease the step size, it takes longer to converge. However, increasing the step size beyond 1 causes the iteration to overshoot, resulting in an overall higher number of required iterations. Moving the starting point further from the minimum increases the number of iterations, but only slightly, because the gradient increases, which increases the step size proportionally. Finally, requiring a stricter convergence criteria via a smaller threshold increases the number of iterations, but only by a constant amount, because the gradient again ensures the step size is in proportion to the distance from the minimum.

By plotting the difference between the finite differences approximation, we can see the accuracy achieved. Here, we again use $\delta = .01$. Figures 1 through 3 are plots of $(\nabla f(\mathbf{x}) - \tilde{\nabla} f(\mathbf{x}))/f(\mathbf{x})$ for $x = (0, 1)$.
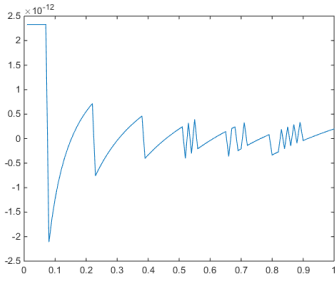


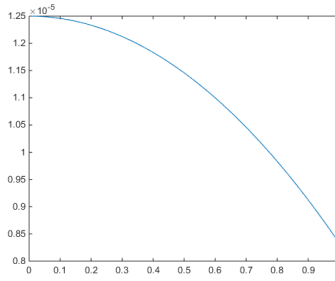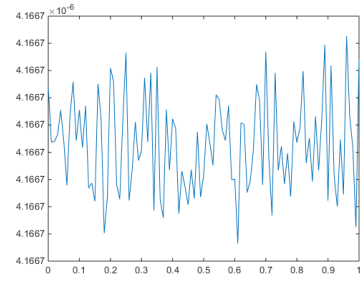Figure 1: $Q_1$, scale of $10^{-12}$      Figure 2: $N_1$, scale of $10^{-5}$      Figure 3: $S_1$, scale of $10^{-6}$

Given the scales of the plots, it is clear that the finite differences method is quite accurate for these functions over these distance scales. As noted, this is generally true when the diagonal entries of the Hessian matrix are small.

## 2

Another way we can benchmark our gradient descent method is to perform basis function regression on some given sample data points. Given, say, 10 data points $X = x^{(1:10)}$ that are generated from a noisy distribution from $y = \sin(2\pi x)$, we'll try to estimate the generating function based on (a) the values at the data points $Y = y^{(1:10)}$ and (b) a basis set of functions.
We can begin by considering a simple polynomial basis; i.e. $\phi(x) = [1, x, x^2, \ldots, x^M]$ where $M$ is the degree of the polynomial we are considering. We wish to minimize our parameter $\theta$, which has a well-known analytic solution:
$$\theta = (\Phi^T \Phi)^{-1} \Phi^T y$$
given that
$$\Phi = somematrixhere$$
while limited to our polynomial basis. We can easily test this by plotting our data $(X, Y)$ with our learned model $\theta^T \Phi(X)$ for varying values of $M$:
We note that we can't have $M > 10$, or else we have too many parameters to adjust for the data points that we have. We can adjust for this using the *Moore-Penrose inverse*, but we will not address it in detail here.
However, we can generalize our basis functions to more than simply polynomials in $x$. In this way, we can introduce trigonometric bases, exponential bases, etc. to give us more flexibility in choosing

the parameter $\theta$. Our ultimate goal is the minimize the sum of the squared errors, or $SSE(\theta)$, equal to

$$\text{SSE}(\theta) = \sum_{i=1}^{n} (\theta^T \phi(x^{(i)}) - y^{(i)})^2$$

. We can minimize this by using gradient descent, which we discussed earlier. We can then confirm our calculations analytically by computing the derivative of $\text{SSE}(\theta)$:

$$\frac{\partial}{\partial \theta} \left( \sum_{i=1}^{n} (\theta^T \phi(x^{(i)}) - y^{(i)})^2 \right) = 2 \sum_{i=1}^{n} \phi(x^{(i)})(\theta^T \phi(x^{(i)}) - y^{(i)})$$