6.867 Homework 1

Anonymous Authors

September 28, 2017

1 Implementing Gradient Descent

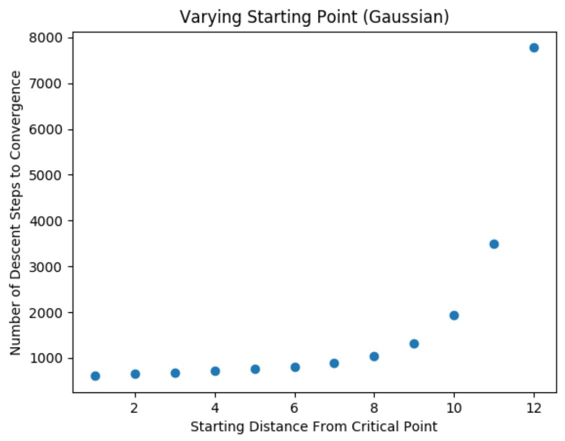
In any gradient descent algorithm, the main hyperparameters we have to tune are the initial point we start the gradient descent from, the step size, and the convergence criteria.

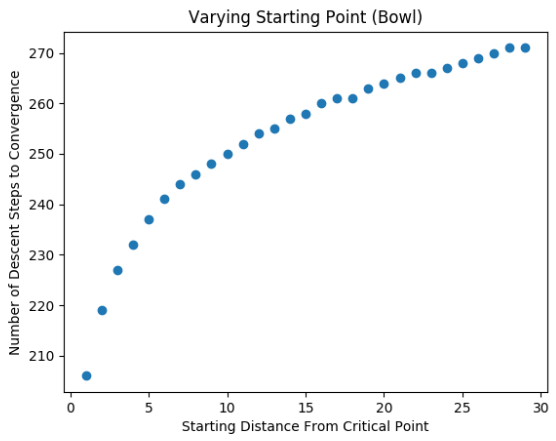
- An incorrect initial guess could lead to getting stuck at a local min without ever reaching the global minumum
- A step size that is too large can shoot past the minumum or go back and forth without ever reaching the critical point. Conversely, a step size that is too small can make the algorithm take far too long to converge.
- A convergence criteria that is too lax can result in a sub-optimal stopping point, while a convergence criteria that is too strict can result in the algorithm taking too long.

For each of the three parameters, we can see how varying the parameters changes the gradient descent for both of the provided functions (the Gaussian and the bowl).

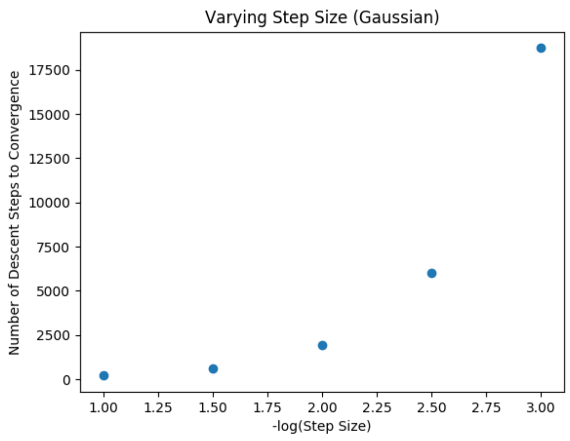
Default parameters are:

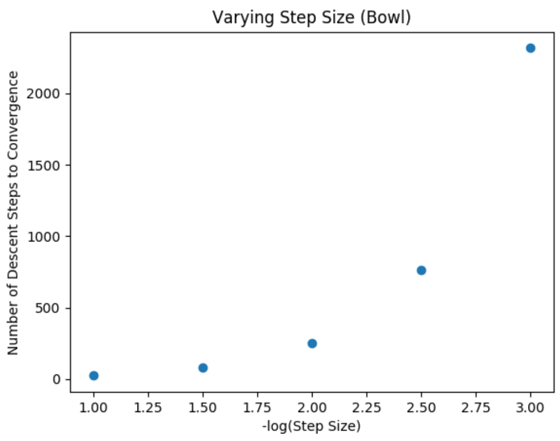
- Starting Point: (0,0)
- Step Size: 0.01
- Convergence Criteria: Difference between consecutive objective function values is less than 10^{-10}, max of 20,000 iterations



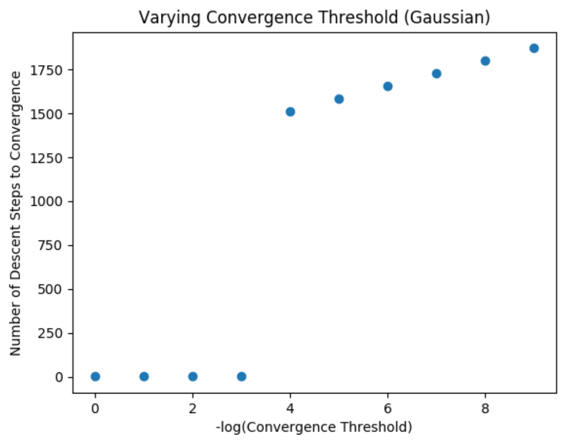


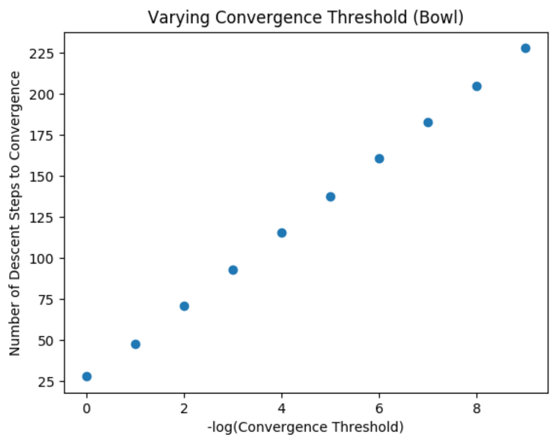
The number of steps needed for convergence grows much faster in the Gaussian than it does with the bowl. This is because the gradient decays exponentially in the Gaussian case with respect to distance from the critical point but linearly in the bowl case.



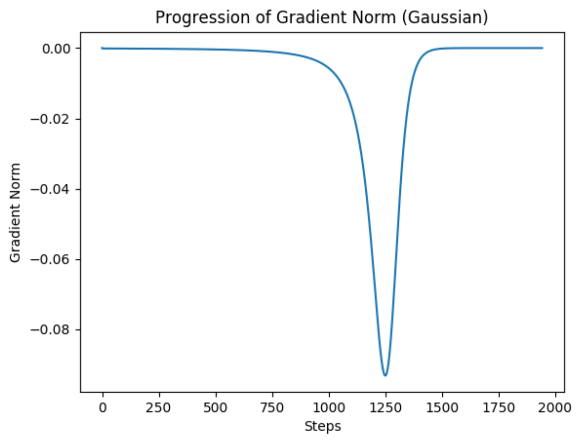


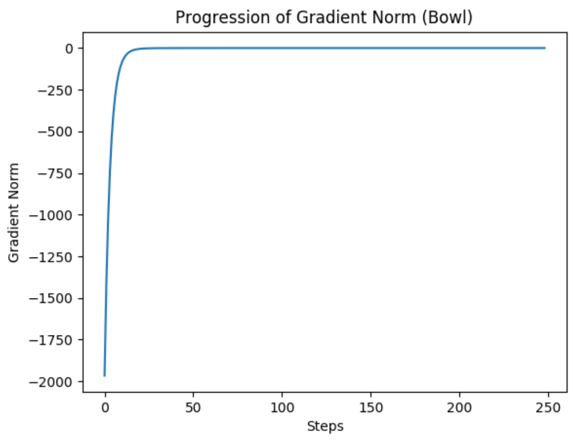
In both cases, step size eventually grows linearly with the number of steps needed until convergence. However, in some cases where the step size is extremely large (not shown), the function may not converge at all.





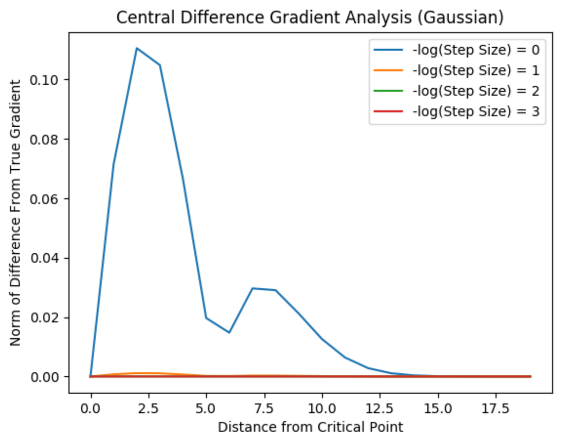
Here, as the convergence threshold grows, so to does the number of steps needed to converge.

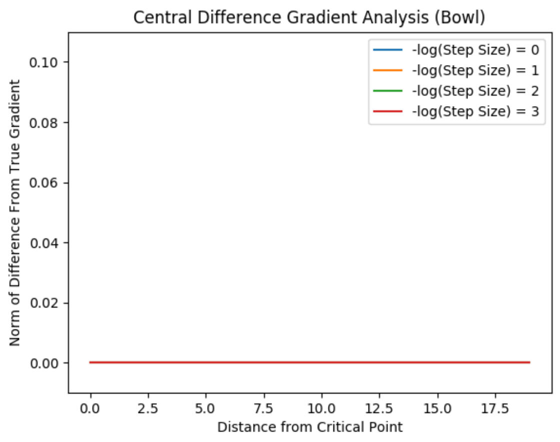




As mentioned before, the gradient norm of the Gaussian starts off very small when the starting point is far away from the critical point. Around step 1000, it starts to reach the steep part of the Gaussian, at which point it quickly reaches convergence. For the bowl, the gradient norm starts off high and quickly decreases and converges.

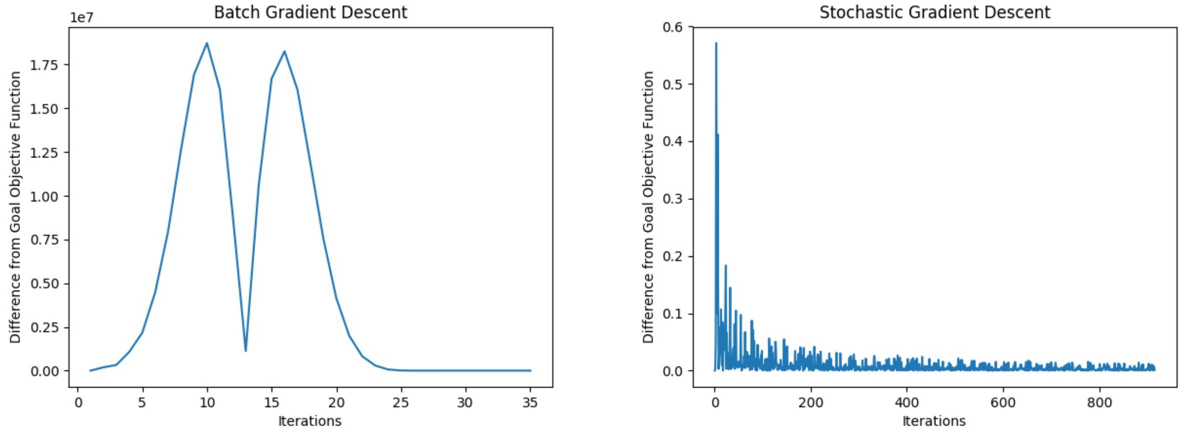
To perform gradient descent on functions that do not have a clean, closed-form gradient, we can approximate the gradient using central differences. Here, we analyze how the performance of the central differences approximation varies based on the step size used for the difference approximation.





When the step size is large, we see some errors in the approximate gradient norm. However, once we reach step sizes of 10^{-2} or smaller, the errors vanish. For the bowl, we have no error in our gradient approximation regardless of step size because the gradient of the bowl is linear. Thus, approximating the gradient of the bowl with the average of two points gives us the exact gradient.

Often, when running gradient descent, it is computationally expensive to calculate the total error. We can use Stochastic Gradient Descent as an alternative: each time we calculate the least square error, we only use one sample to calculate the error. This results in a much faster overall computation at the expense of some accuracy. We can compare the two algorithms by calculating the gradient descent of the least square error with similar convergence criteria.



It takes many more iterations for SGD to converge than batch descent. Furthermore, batch descent can converge on a more accurate answer (due to the randomness of SGD throwing things off). However, SGD runs much faster, even while taking more iterations. In practice, mini-batch SGD is often used as the best of both worlds.

2 Linear Basis Function Regression

2.1 Closed-form Solution with Polynomial Basis

Consider the basis function

$$\Phi_M(x) = [\phi_0(x), \ldots, \phi_M(x)],$$

where $\phi_k(x) = x^k$. Applying Φ_M to $\mathbf{X}$ gives the desired basis change, so we have the generalized linear model

$$\mathbf{Y} = \Phi_M(\mathbf{X}) \cdot \boldsymbol{\beta} + \boldsymbol{\epsilon},$$

which has the close-form solution (from first order conditions on the least squares error[1]),

$$\hat{\boldsymbol{\beta}} = (\Phi_M(\mathbf{X})^T \Phi_M(\mathbf{X}))^{-1} \Phi_M(\mathbf{X})^T \mathbf{Y}$$

where $\hat{\boldsymbol{\beta}}$ is the maximum-likelihood estimator of the regression coefficients.

We fit polynomial regressions on the data via this method with a few different max degrees ($M = 0, 1, 3, 10$).[2] Below are plots of the resulting polynomial functions, compared to the original data and source function:

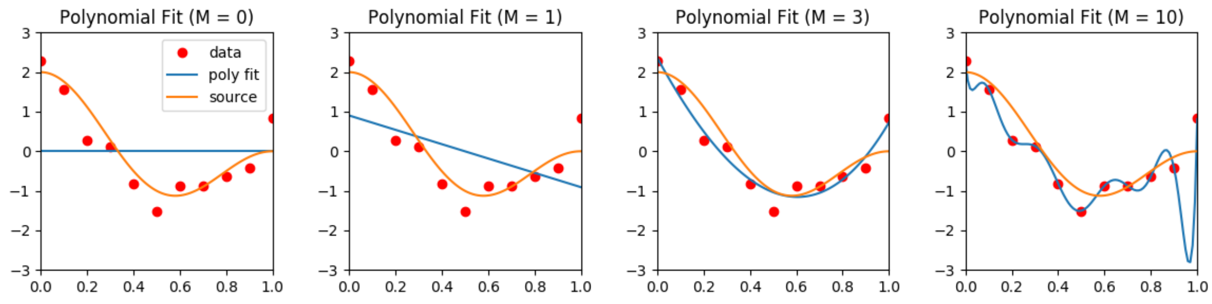


Figure 7: M-Degree Polynomial Fit

[1]See section 2.2 for a derivation of the gradient vector.
[2]See the appendix for the regression coefficients.

It appears that $M = 3$ gives the closest interpolation of the true function. $M = 0, 1$ give a poor fit as the models do not have sufficient degrees of freedom, while $M = 10$ has too many paramters and leads to overfitting.

2.2 Closed-form Gradient of Squared Error

We take a second approach to finding the least squares estimators (which also happen to be maximum likelihood estimators). The residual sum of squares for weight vector β is given by

$$loss(\boldsymbol{\beta}) = |(\boldsymbol{Y} - \Phi_M(\boldsymbol{X})\boldsymbol{\beta})|^2.$$

Differentiating the error function with respect to β gives the gradient function

$$\begin{aligned}\frac{\partial}{\partial \boldsymbol{\beta}}|(\boldsymbol{Y} - \Phi_M(\boldsymbol{X})\boldsymbol{\beta})|^2 &= \frac{\partial}{\partial \boldsymbol{\beta}}[(\boldsymbol{Y} - \Phi_M(\boldsymbol{X})\boldsymbol{\beta})^T \cdot (\boldsymbol{Y} - \Phi_M(\boldsymbol{X})\boldsymbol{\beta})] \\ &= \frac{\partial}{\partial \boldsymbol{\beta}}[-(\Phi_M(\boldsymbol{X})\boldsymbol{\beta})^T\boldsymbol{Y} - \boldsymbol{Y}^T(\Phi_M(\boldsymbol{X})\boldsymbol{\beta}) + \boldsymbol{\beta}^T\Phi_M(\boldsymbol{X})^T \cdot (\Phi_M(\boldsymbol{X})\boldsymbol{\beta})] \\ &= -2 \cdot \Phi_M(\boldsymbol{X})^T\boldsymbol{Y} + 2 \cdot \Phi_M(\boldsymbol{X})^T\Phi_M(\boldsymbol{X})\boldsymbol{\beta}.\end{aligned}$$

2.3 Gradient Descent Solution with Polynomial Basis

2.4 Closed-form Solution with Cosine Function Basis

3 Ridge Regression

4 Sparsity and Lasso

5 Appendix