6.867 Homework 1

Anonymous Authors

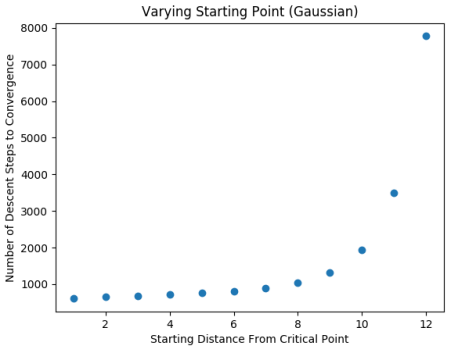
September 28, 2017

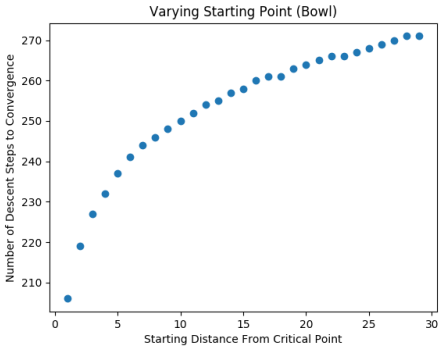
1 Implementing Gradient Descent

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.

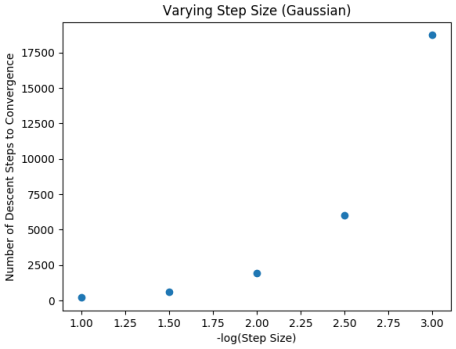
For each of the three parameters, we can see how varying the parameters changes the gradient descent for both of the provided functions (the Gaussian and the bowl).

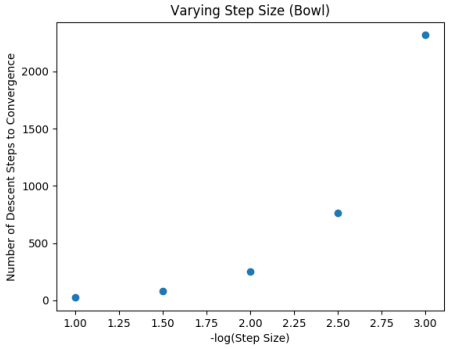
Unless otherwise specified, default parameters are: starting point @ (0,0), step size of 0.01, and convergence criteria of difference between consecutive objective function values is less than 10^{-10}.



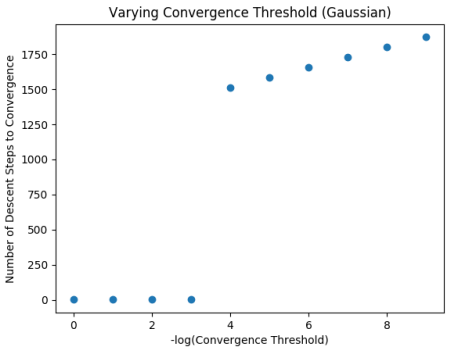


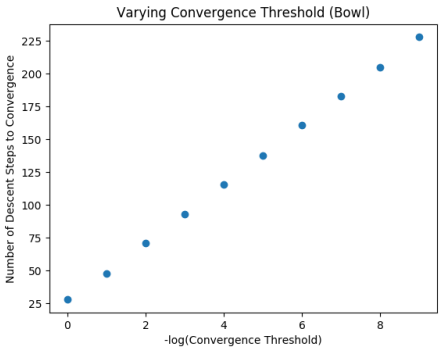
The number of steps needed for convergence grows much faster in the Gaussian than it does with the bowl. This is because the gradient decays exponentially in the Gaussian case with respect to distance from the critical point but linearly in the bowl case.



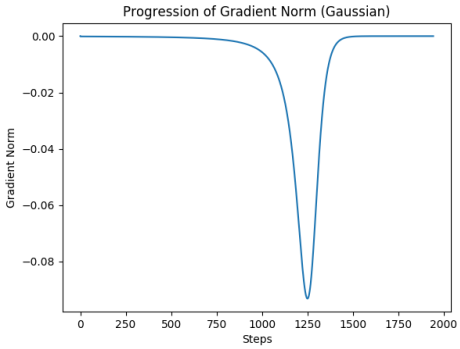


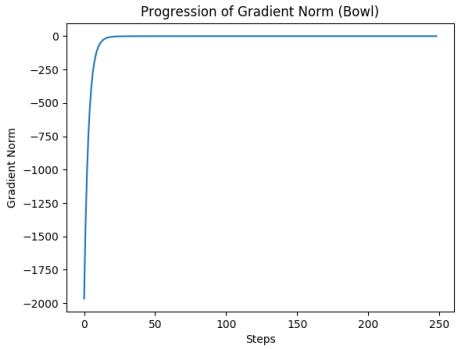
In both cases, step size eventually grows linearly with the number of steps needed until convergence. However, in some cases where the step size is extremely large (not shown), the function may not converge at all.





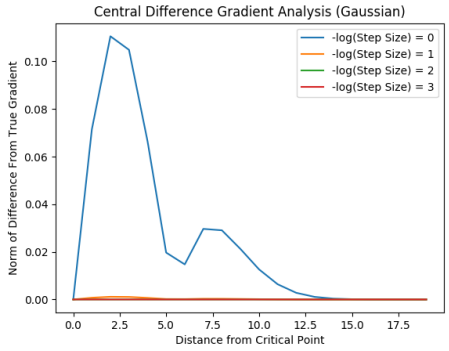
Here, as the convergence threshold grows, so to does the number of steps needed to converge. Intuitively, this makes sense–the more accurate the gradient descent needs to be, the more iterations we need to converge.

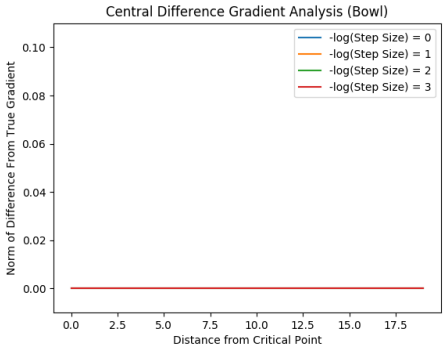




As mentioned before, the gradient norm of the Gaussian starts off very small when the starting point is far away from the critical point. Around step 1000, it starts to reach the steep part of the Gaussian, at which point it quickly reaches convergence. For the bowl, the gradient norm starts off high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

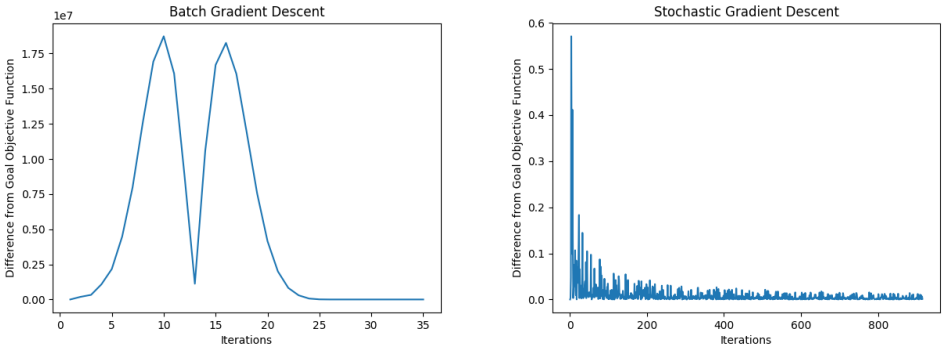
To perform gradient descent on functions that do not have a clean, closed-form gradient, we can approximate the gradient using central differences. Here, we analyze how the performance of the central differences approximation varies based on the step size used for the difference approximation.





When the step size is larger than 10^{-2}, we start to see errors in the Gaussian. For the bowl, we have no error in our gradient approximation regardless of step size because the gradient of the bowl is linear; because of this, our averaging approximation acutally gives us the exact answer.

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

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

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

We ran regressions on the data using a few different degrees ($M = 0, 1, 3, 10$).[1] Below are plots of the resulting polynomial functions, compared to the given data and the true function:

[1] See the Appendix for the numerical values of the weights.

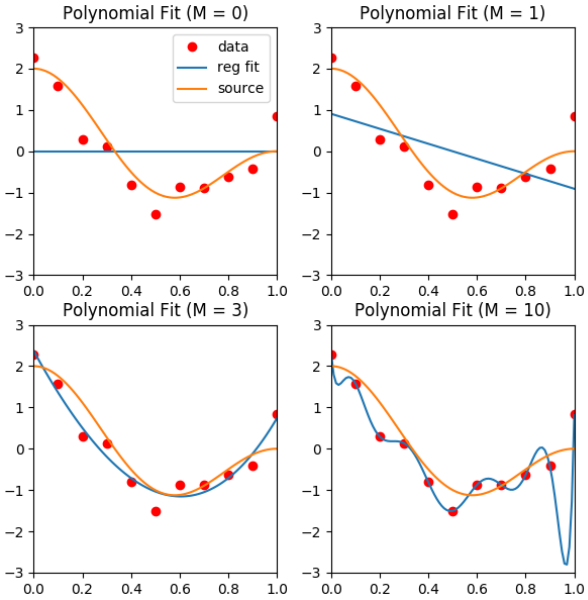


Figure 7: M-Degree Polynomial Fit (M = 0, 1, 3, 10)

2.2 Gradient Descent Solution with Polynomial Basis

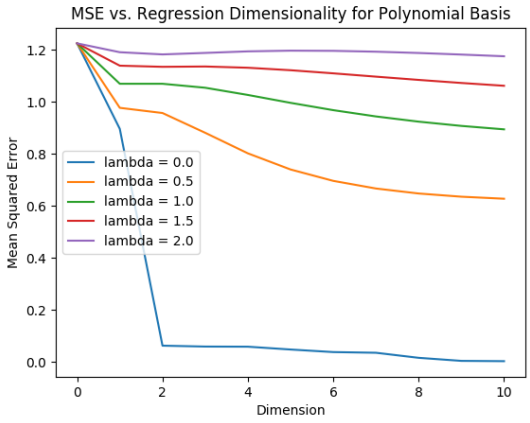
2.3 Closed-form Solution with Cosine Function Basis

3 Ridge Regression

We first run ridge regression on the polynomial basis data set from the previous problem. Given that we are trying to minimize the squared error $||X\theta - y||^2 + \lambda||\theta||^2$, the closed form solution gives us:

$$\theta = (X^T X + \lambda I)^{-1} X^T y$$

We plot the MSE for different dimensionality models for different values of λ.



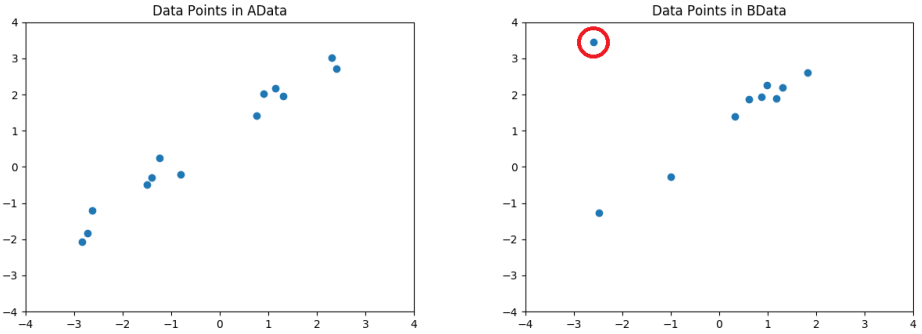
For $\lambda = 0$, increasing the dimensionality slowly improves the MSE. However, as λ grows, the benefit shrinks because the regularization term punishes large terms in θ.

We then run our ridge regression on the provided datasets. For each value of λ, we run ridge regression

for different values of M and pick the model with the best validation error. We then run that model on the test set and compute the MSE.

Train on A, Test on B			
λ	Optimal M	Validation MSE	Test MSE
0	2	0.107	2.575
0.5	2	0.126	2.566
1	2	0.147	2.564
1.5	2	0.170	2.566
2	1	0.188	2.584
2.5	1	0.206	2.604
3	1	0.225	2.623
3.5	1	0.245	2.643
4	1	0.265	2.662

These results look weird because the test errors are so much higher than the validation errors. This is because there is a clear outlier data point in B that throws off the MSE.



Plotting all of the data points. The circled point is the outlier.

Two ways to deal with outliers include 1) upper-bounding errors incurred by single points, or 2) removing the points entirely. We experimented with removing the outlier point and ran our ridge regression again.

Train on A, Test on Modified B			
λ	Optimal M	Validation MSE	Test MSE
0	2	0.107	0.06
0.5	2	0.126	0.08
1	2	0.147	0.106
1.5	2	0.170	0.133
2	1	0.188	0.158
2.5	1	0.206	0.183
3	1	0.225	0.209
3.5	1	0.245	0.235
4	1	0.265	0.262

Train on Modified B, Test on A			
λ	Optimal M	Validation MSE	Test MSE
0	3	0.089	0.193
0.5	4	0.091	0.084
1	1	0.125	0.079
1.5	1	0.132	0.077
2	1	0.144	0.078
2.5	1	0.159	0.084
3	1	0.177	0.092
3.5	1	0.197	0.102
4	1	0.218	0.115

We see some differences between the two test runs. When training on A and testing on B, the best model is quadratic, while it is linear when swapping the training and test sets. In both cases, though, the models that performed well had relatively low dimension, which makes sense given what the data looks like. Furthermore, setting λ above 3 seemed too harsh of a regularization penalty. Ultimately, given the provided data sets, picking a linear or quadratic model with a small regularization penalty seems like the best bet.

4 Sparsity and Lasso

5 Appendix