1 Exercise 1

The implemented gradient descent procedure was tested on three functions:

(a) A convex function: $f(x) = -Gaussian(6, 4)$ (an upside-down Gaussian with mean 6 and variance 4), who's minimum value is at $(6, -1)$.

(b) A scalar convex function of two variables: $f(x, y) = (x + 2)^2 + y^2$, who's minimum value is at $(-2, 0)$.

(c) A non-convex function: $f(x) = \sin(x + \frac{\pi}{2})$, who has multiple minima at $(n\pi, -1)$ for all values of n.

Figure 1 shows the result of calculating the gradient descent on functions (a) and (b) written above. The dots are the calculated minima from gradient descent calculated at multiple step sizes, initial guesses, and thresholds. For convex functions with one minimum, the threshold is the biggest indicator of how many steps the procedure takes to converge. The threshold also determines how close the result is to the real minimum - the smaller the threshold, the closer the result is to the real minimum. Because there is one global minimum, the initial guess does not affect the end result of the function or significantly change the number of iterations required to converge.

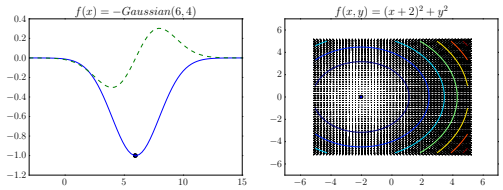

Figure 1: (a) Convex function with known minimum value and it's analytical derivative. The dots are the calculated minima according to gradient descent. (b) Contour plot of the gradient of a scalar function with vector-valued arguments. The dots are the calculated minima according to gradient descent.

Figure 2 shows the result of using gradient descent on function (c) with different parameters. The sequence of dots in each case represents the guesses that gradient descent takes to be the minimum. In general, the choice of starting point affects which local minimum gradient descent will find. If the starting point chosen is a local maximum, the procedure will not step towards a minimum because the gradient is zero at the maximum. If the step size is large, the procedure will overshoot the minimum and zig-zag back to the desired minimum, taking a large number of iterations to converge.

Approximating a function's gradient using central differences is very accurate if using an h of 1. Figure 3 shows graphs of either the gradient function or gradient vector field for the three functions tested above. It is hard to tell the difference between the analytical and numerical functions, so using central differences to numerically estimate the gradient is effective.

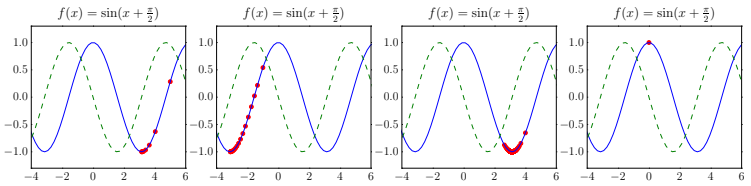

Figure 2: The solid line is the function, the dashed line is its analytical derivative, and the series of dots is the series of minimum guesses from gradient descent. (a) Initial guess: 5, step: 0.5, threshold: 0.1 converges in 9 iterations. (b) Initial guess: -2, step: 0.2, threshold: 0.1 converges in 20 iterations. (c) Initial guess: 4, step: 2, threshold: 0.1 converges in 59992 iterations. (d) Initial guess: 0, step: 0.5, threshold: 0.1 converges in 1 iteration, but does not converge to a minimum.

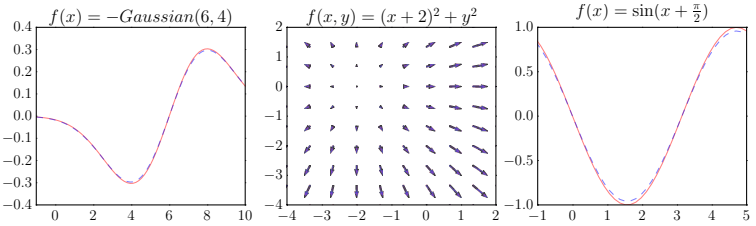

Figure 3: The analytical and numerical gradients for the functions (a) upside-down Gaussian, (b) $f(x, y) = (x + 2)^2 + y^2$, and (c) $f(x) = \sin(x + \frac{\pi}{2})$. The solid red line is the analytical gradient and the dashed blue line is the numerical gradient.

The BFGS optimizer in SciPy uses a more sophisticated method for estimating minima, so in general, the number of function evaluations needed before convergence is smaller. Table 1 shows a comparison of the gradient descent and BFGS optimizer, where in all cases the step size was 0.2 and the threshold was 0.01.

Function	Initial Guess	Algorithm	Evaluations	Min at
$f(x) = -\text{Gaussian}(6, 4)$	$x = 5$	GD	34	$x = 5.81$
		BFGS	6	$x = 6.00$
$f(x, y) = (x + 2)^2 + y^2$	$(x, y) = (5, 5)$	GD	13	$(x, y) = (-2, 0)$
		BFGS	4	$(x, y) = (-2, 0)$
$f(x) = \sin(x + \frac{\pi}{2})$	$x = 5$	GD	20	$x = 3.15$
		BFGS	6	$x = 3.14$

Table 1: Steps to convergence given a similar set of parameters. GD is gradient descent, BFGS is from SciPy. For gradient descent, all step sizes were 0.5 and thresholds were 0.01.

2 Exercise 2

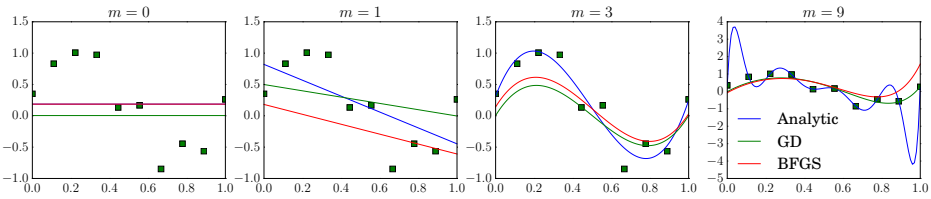

Figure 4: Plots replicating Figure 1.4 in Bishop for different values of M using different methods. The blue is generated with the analytical solution of the SSE, the green with the gradient descent method on the SSE, and the red using Python's BFGS optimizer.

	$M=0$	$M=1$	$M=6$	$M=9$
$\hat{w}_0$	0.186	0.820	0.353	0.349
$\hat{w}_1$		−1.268	2.619	232.411
$\hat{w}_2$			32.096	−5322.833
$\hat{w}_3$			−206.273	48577.404
$\hat{w}_4$			399.001	−231682.150
$\hat{w}_5$			−332.709	640159.305
$\hat{w}_6$			105.163	−1061992.54
$\hat{w}_7$				1042586.68
$\hat{w}_8$				−557781.754
$\hat{w}_9$				125223.393

	$M=0$	$M=1$	$M=6$	$M=9$
$\hat{w}_0$	0.003	0.499	0.623	−0.014
$\hat{w}_1$		−0.502	5.577	6.018
$\hat{w}_2$			−14.465	−12.820
$\hat{w}_3$			5.502	6.294
$\hat{w}_4$			0.475	−5.629
$\hat{w}_5$			0.453	4.425
$\hat{w}_6$			0.435	0.464
$\hat{w}_7$				0.493
$\hat{w}_8$				0.516
$\hat{w}_9$				0.533

	$M=0$	$M=1$	$M=6$	$M=9$
$\hat{w}_0$	0.186	0.181	0.133	−0.077
$\hat{w}_1$		−0.791	4.974	6.080
$\hat{w}_2$			−0.150	−12.746
$\hat{w}_3$			4.964	6.398
$\hat{w}_4$			−0.027	−5.494
$\hat{w}_5$			−0.020	4.586
$\hat{w}_6$			−0.013	0.646
$\hat{w}_7$				0.692
$\hat{w}_8$				0.729
$\hat{w}_9$				0.758

Figure 5: Table 1.1 from Bishop replicated using (a)analytical solutions, (b) gradient descent, and (c) the Scipy BFGS optimizer

3 Exercise 3

This is a test.

4 Exercise 4

This is a test.