6.867: Machine Learning, Homework 1

1. Implement Gradient Descent

1.1 Implementation the gradient Descent

We implemented a function `gradientDescent` that takes the parameters:

- A scalar function $f : \mathbb{R}^n \mapsto \mathbb{R}$.
- The gradient of f $\nabla f : \mathbb{R}^n \mapsto \mathbb{R}^n$.
- An initial guess $x \in \mathbb{R}^n$.
- A step size $s > 0$.
- A convergence threshold $\epsilon > 0$.

`gradientDescent` returns an array of $n + 1$ numbers, corresponding to the n coordinates of the current x_t, and the value of $f(x_t)$, for each of the steps t before convergence of the algorithm.

This choice of output lets us monitor the rate of convergence of the algorithm (which zones lead to slower or faster convergence), which will be useful when we compare different methods (see section 1.4). It also allows us to plot the path followed by the algorithm (see figures **??**to **??**).

1.2 Benchmark and Choice of the parameters

We chose to benchmark our gradient descent on a variety of functions, designed to test all the potential cases that we might encounter. Because of plotting constraints, we focus the figures to function of two parameters: $f : \mathbb{R}^2 \mapsto \mathbb{R}$.

1.3 Numerical Gradient

1.4 Comparison with existing optimization methods

2. Linear Basis Function Regression

3. Ridge Regression

4. Generalizations

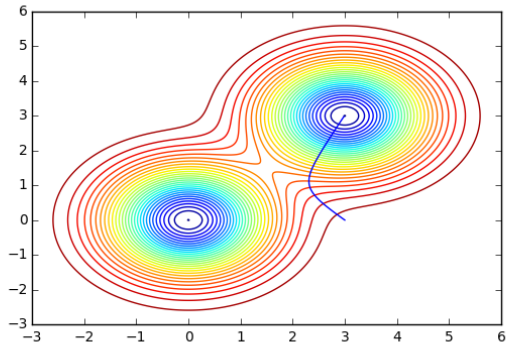

Figure 1. $f : (x, y) \to e^{-\frac{x^2+y^2}{2}} + e^{-\frac{(x-3)^2+(y-3)^2}{2}}$

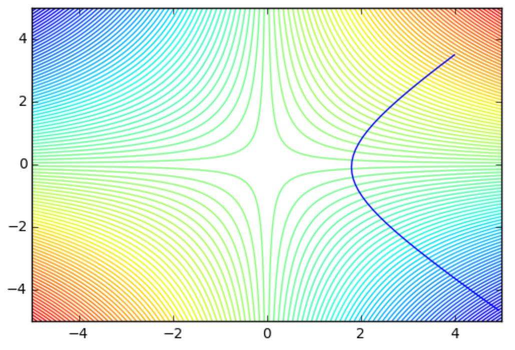

Figure 2. $f : (x, y) \to xy$

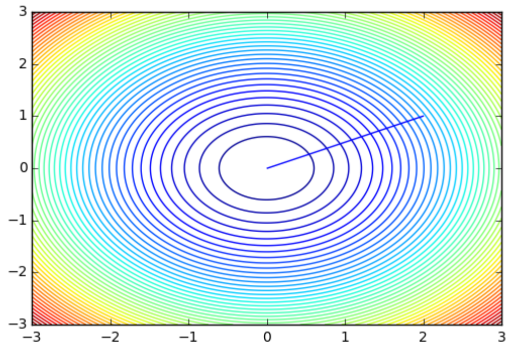

Figure 3. $f : (x, y) \to x^2 + y^2$