

6.867 Machine Learning

Homework 1

1 IMPLEMENTING GRADIENT DESCENT

Gradient descent is an iterative procedure for find a vector that minimize an objective function. Typically, the vector is our parameters and the objective function is the cost function. In each iteration, we find the gradient of the objective function evaluated at the vector and update the vector in the direction of the negative gradient.

1.1 Basic Gradient Descent

To demonstrate the gradient descent algorithm, we began by finding the minimum of two well defined functions with closed form derivatives shown below.

Negative Gaussian:

$$f(x) = -\frac{1}{\sqrt{(2\pi)^n |\Sigma|}} \exp\left[-\frac{1}{2}(x-u)^T \Sigma^{-1}(x-u)\right] \quad (1)$$

$$\frac{\partial f(x)}{\partial x} = -f(x) \Sigma^{-1}(x-u) \quad (2)$$

Quadratic Bowl:

$$f(x) = \frac{1}{2} x^T A x - x^T b \quad (3)$$

$$\frac{\partial f(x)}{\partial x} = A x - b \quad (4)$$

How did we do?

The gradient descent algorithm takes three additional parameters, the starting guess, step size, and convergence criterion. Each affects the end result of the algorithm. The starting guess is important because gradient descent iteratively follows the gradient. Thus it can get stuck in local minimum. By running the algorithm repeated with random initialization, we increase our chance of finding a global minimum. The step size affects convergence behavior of the algorithm. As depicted in figure 1, if the step size is too small, the algorithm will converge slowly. If the step size is too big, the algorithm will overshoot the minimum and oscillate, converging more slowly. If the

step size is much too big, the algorithm will actually diverge. This is depicted in figure 1 where for different step sizes, the norm of the gradients are plotted over time. The convergence criterion determines for how little change in cost function is it okay to stop. Decreasing this gives greater accuracy but increase runtime.

Include Graphs at the end!!

1.2 Central Difference Approximation

For many objective functions, it is impossible to write a closed form gradient function. Thus, use central difference approximation to approximate the gradient. For each dimension i , estimate its partial derivative by

$$\nabla_i f(x) = \frac{f(x + d \hat{e}_i) - f(x - d \hat{e}_i)}{2d} \quad (5)$$

for some small d . The larger d is, the more inaccurate the gradient approximation is. We will use $d = 10^{-3}$.

1.3 Batch vs. Stochastic Gradient Descent

Now use gradient descent to find parameters θ that minimizes the least squared error objective function

$$J(\theta) = \|X\theta - y\|^2 \quad (6)$$

where each row of X and y is a data sample pair.

In batch gradient descent, θ is updated with the gradient of the cost function for the entire training dataset. How quickly did we converge and what did we converge to?

In contrast, stochastic gradient descent updates θ with the gradient of the cost function for each data point. θ is updated according to

$$\theta_{t+1} = \theta_t - \eta_t \nabla_{\theta} J(\theta_t; x^{(i)}, y^{(i)}). \quad (7)$$

We iterate through the entire dataset n times We converge when the cost between iterations

2 LINEAR BASIS FUNCTION REGRESSION

You can write stuff here to but don't have to.

2.1 Closed Form Solution

2.2 Gradient Descent Objective Functions

Write functions to compute SSE and derivative.

2.3 Batch vs Stochastic Gradient Descent

Compare!

2.4 Cosine Basis Functions

Shtuff.

3 RIDGE REGRESSION

3.1 Closed Form Solution

3.2 Results?

4 SPARSITY AND LASSO

4.1 Lasso