

# Introduction to Machine Learning

## Homework 5: Gradient Calculations and Nonlinear Optimization

Prof. Sundeep Rangan

Submit answers only to problems 1, 3 and 4(b) and (c). You do not need to answer 2 or 4(a). But, make sure you know how to do all the problems.

1. Suppose we want to fit a model,

$$\hat{y} = \frac{1}{w_0 + \sum_{j=1}^d w_j x_j},$$

for parameters  $\mathbf{w}$ . Given training data  $(\mathbf{x}_i, y_i)$ ,  $i = 1, \dots, n$ , a nonlinear least squares fit could use the loss function,

$$J(\mathbf{w}) = \sum_{i=1}^n \left[ y_i - \frac{1}{w_0 + \sum_{j=1}^d w_j x_{ij}} \right]^2$$

- (a) Find a function  $g(\mathbf{z})$  and matrix  $\mathbf{A}$  such that the loss function is given by,

$$J(\mathbf{w}) = g(\mathbf{z}), \quad \mathbf{z} = \mathbf{A}\mathbf{w},$$

and  $g(\mathbf{z})$  is factorizable, meaning  $g(\mathbf{z}) = \sum_i g_i(z_i)$  for some functions  $g_i(z_i)$ .

- (b) What is the gradient  $\nabla J(\mathbf{w})$ ?
  - (c) What is the gradient descent update for  $\mathbf{w}$ ?
  - (d) Write a few lines of python code to compute the loss function  $J(\mathbf{w})$  and  $\nabla J(\mathbf{w})$ .
2. In this problem, we will see why gradient descent can often exhibit very slow convergence, even on apparently simple functions. Consider the objective function,

$$J(\mathbf{w}) = \frac{1}{2}b_1 w_1^2 + \frac{1}{2}b_2 w_2^2,$$

defined on a vector  $\mathbf{w} = (w_1, w_2)$  with constants  $b_2 > b_1 > 0$ .

- (a) What is the gradient  $\nabla J(\mathbf{w})$ ?
- (b) What is the minimum  $\mathbf{w}^* = \arg \min_{\mathbf{w}} J(\mathbf{w})$ ?
- (c) Part (b) shows that we can minimize  $J(\mathbf{w})$  easily by hand. But, suppose we tried to minimize it via gradient descent. Show that the gradient descent update of  $\mathbf{w}$  with a step-size  $\alpha$  has the form,

$$w_1^{k+1} = \rho_1 w_1^k, \quad w_2^{k+1} = \rho_2 w_2^k,$$

for some constants  $\rho_i$ ,  $i = 1, 2$ . Write  $\rho_i$  in terms of  $b_i$  and the step-size  $\alpha$ .

- (d) For what values  $\alpha$  will gradient descent converge to the minimum? That is, what step sizes guarantee that  $\mathbf{w}^k \rightarrow \mathbf{w}^*$ .
- (e) Take  $\alpha = 2/(b_1 + b_2)$ . It can be shown that this choice of  $\alpha$  results in the fastest convergence. You do not need to show this. But, show that with this selection of  $\alpha$ ,

$$\|\mathbf{w}^k\| = C^k \|\mathbf{w}^0\|, \quad C = \frac{\kappa - 1}{\kappa + 1}, \quad \kappa = \frac{b_2}{b_1}.$$

The term  $\kappa$  is called the *condition number*. The above calculation shows that when  $\kappa$  is very large,  $C \approx 1$  and the convergence of gradient descent is slow. In general, gradient descent performs poorly when the problems are ill-conditioned like this.

3. *Matrix minimization.* Consider the problem of finding a matrix  $\mathbf{P} \in \mathbb{R}^{m \times m}$  to minimize the loss function,

$$J(\mathbf{P}) = \sum_{i=1}^n \left[ \frac{z_i}{y_i} - \ln(z_i) \right], \quad z_i = \mathbf{x}_i^T \mathbf{P} \mathbf{x}_i.$$

The problem arises in wireless communications where an  $m$ -antenna receiver wishes to estimate a spatial covariance matrix  $\mathbf{P}$  from  $n$  power measurements. In this setting,  $y_i > 0$  is the  $i$ -th receive power measurement and  $\mathbf{x}_i$  is the beamforming direction for that measurement. In reality, the quantities would be complex, but for simplicity we will just look at the real-valued case. See the following article for more details:

Eliasi, Parisa A., Sundeep Rangan, and Theodore S. Rappaport. “Low-rank spatial channel estimation for millimeter wave cellular systems,” *IEEE Transactions on Wireless Communications* 16.5 (2017): 2748-2759.

- (a) What is the gradient  $\nabla_{\mathbf{P}} z_i$ ?
- (b) What is the gradient  $\nabla_{\mathbf{P}} J(\mathbf{P})$ ?
- (c) Write a few lines of python code to evaluate  $J(\mathbf{P})$  and  $\nabla_{\mathbf{P}} J(\mathbf{P})$  given data  $\mathbf{x}_i$  and  $y_i$ . You can use a for loop.
- (d) See if you can rewrite (c) without a for loop. You will need Python broadcasting.
4. *Nested optimization.* Suppose we are given a loss function  $J(\mathbf{w}_1, \mathbf{w}_2)$  with two parameter vectors  $\mathbf{w}_1$  and  $\mathbf{w}_2$ . In some cases, it is easy to minimize over one of the sets of parameters, say  $\mathbf{w}_2$ , while holding the other parameter vector (say,  $\mathbf{w}_1$ ) constant. In this case, one could perform the following *nested* minimization: Define

$$J_1(\mathbf{w}_1) := \min_{\mathbf{w}_2} J(\mathbf{w}_1, \mathbf{w}_2), \quad \hat{\mathbf{w}}_2(\mathbf{w}_1) := \arg \min_{\mathbf{w}_2} J(\mathbf{w}_1, \mathbf{w}_2),$$

which represent the minimum and argument of the loss function over  $\mathbf{w}_2$  holding  $\mathbf{w}_1$  constant. Then,

$$\hat{\mathbf{w}}_1 = \arg \min_{\mathbf{w}_1} J_1(\mathbf{w}_1) = \arg \min_{\mathbf{w}_1} \min_{\mathbf{w}_2} J(\mathbf{w}_1, \mathbf{w}_2).$$

Hence, we can find the optimal  $\mathbf{w}_1$  by minimizing  $J_1(\mathbf{w}_1)$  instead of minimizing  $J(\mathbf{w}_1, \mathbf{w}_2)$  over  $\mathbf{w}_1$  and  $\mathbf{w}_2$ .

- (a) Show that the gradient of  $J_1(\mathbf{w}_1)$  is given by

$$\nabla_{\mathbf{w}_1} J_1(\mathbf{w}_1) = \nabla_{\mathbf{w}_1} J(\mathbf{w}_1, \mathbf{w}_2)|_{\mathbf{w}_2 = \hat{\mathbf{w}}_2}.$$

Thus, given  $\mathbf{w}_1$ , we can evaluate the gradient from (i) solve the minimization  $\hat{\mathbf{w}}_2 := \arg \min_{\mathbf{w}_2} J(\mathbf{w}_1, \mathbf{w}_2)$ ; and (ii) take the gradient  $\nabla_{\mathbf{w}_1} J(\mathbf{w}_1, \mathbf{w}_2)$  and evaluate at  $\mathbf{w}_2 = \hat{\mathbf{w}}_2$ .

- (b) Suppose we want to minimize a nonlinear least squares,

$$J(\mathbf{a}, \mathbf{b}) := \sum_{i=1}^n \left( y_i - \sum_{j=1}^d b_j e^{-a_j x_i} \right)^2,$$

over two parameters  $\mathbf{a}$  and  $\mathbf{b}$ . Given parameters  $\mathbf{a}$ , describe how we can minimize over  $\mathbf{b}$ . That is, how can we compute,

$$\hat{\mathbf{b}} := \arg \min_{\mathbf{b}} J(\mathbf{a}, \mathbf{b}).$$

- (c) In the above example, how would we compute the gradients,

$$\nabla_{\mathbf{a}} J(\mathbf{a}, \mathbf{b}).$$