

Introduction to Machine Learning

Unit 7 Problems: Gradient Calculations and Nonlinear Optimization

Prof. Sundeep Rangan

1. *Simple gradient calculation.* Consider a function,

$$J = z_1 e^{z_1 z_2}, \quad z_1 = a_1 w_1 w_2, \quad z_2 = a_2 w_1 + a_3 w_2^2,$$

- (a) Compute the partial derivatives, $\partial J / \partial w_j$ for $j = 1, 2$.
 - (b) Write a python function that, given \mathbf{w} and \mathbf{a} computes $J(\mathbf{w})$ and $\nabla J(\mathbf{w})$.
2. *Gradient with a logarithmic loss.* Consider the loss function,

$$J(\mathbf{w}, b) := \sum_{i=1}^N (\log(y_i) - \log(\hat{y}_i))^2, \quad \hat{y}_i = \sum_{j=1}^p x_{ij} w_j + b,$$

This is an MSE loss function, but in log domain.

- (a) Find the gradient components, $\partial J / \partial w_j$ and $\partial J / \partial b$.
- (b) Complete the following python function

```
def Jeval(w,b,...):  
    ...  
    return J, Jgradw, Jgradb
```

that computes J and $\nabla_w J$ and $\nabla_b J$. You need to complete the arguments of the function. To receive full credit, avoid using for loops.

3. *Gradient with an inverse function.* Consider the nonlinear least squares fit 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) Compute the gradient components, $\partial J / \partial w_j$. You may want to define the intermediate variable,

$$z_i = w_0 + \sum_{j=1}^d w_j x_{ij}.$$

Also, you can write separate answers for $\partial J / \partial w_0$ and $\partial J / \partial w_j$ for $j = 1, \dots, d$.

- (b) Complete the following function to compute the loss and gradient,

```
def Jeval(w,...):
    ...
    return J, Jgrad
```

For the gradient, you may wish to use the function,

```
Jgrad = np.hstack((Jgrad0, Jgrad1))
```

to stack two vectors.

4. *Gradient with nonlinear parametrization.* Given data (x_i, y_i) with binary class labels $y_i \in \{0, 1\}$, consider the binary cross-entropy loss function,

$$J(\mathbf{a}, \mathbf{b}) := \sum_{i=1}^N \log(1 + e^{z_i}) - y_i z_i, \quad z_i = \sum_{j=1}^d a_j e^{-(x_i - b_j)^2/2}.$$

- (a) Compute the gradient components, $\partial J / \partial a_j$ and $\partial J / \partial b_j$.
 (b) Complete the following function to compute the loss and gradient,

```
def Jeval(a,b,...):
    ...
    return J, Jgrada, Jgradb
```

Avoid for loops to receive full credit.

5. *Finding local and global minima.* Consider the function

$$f(x) = \frac{1}{4}x^2 + 1 - \cos(2\pi x).$$

- (a) Approximately draw $f(x)$.
 (b) Write an equation for the gradient descent update to minimize $f(x)$.
 (c) Using the graph in part (a), where is the global minima of $f(x)$?
 (d) Using the graph in part (a), find one initial condition where gradient descent could end up converging to a local minima that is not the global minima. The local minima do not have a closed form expression, but you should be able to use the graph in part (a) to “eyeball” an initial condition close to a bad local minima.
6. 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 α 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 ρ_i , $i = 1, 2$. Write ρ_i in terms of b_i and the step-size α .

- (d) For what values α 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 α results in the fastest convergence. You do not need to show this. But, show that with this selection of α ,

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

The term κ is called the *condition number*. The above calculation shows that when κ 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.

7. *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.
8. *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,

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