Exercise 1 - Taylor Series

Compute the Taylor series expansion up to the second order term for the following multivariate functions around a given point:

(a) $f(x) = 5x^3$ around $x_0 = 1$.
(b) $f(x, y) = x^2 \cdot y^3 + x^2$ around $x_0 = 3$, $y_0 = 2$.
(c) $f(\mathbf{x}) = x_1^3 \cdot x_2 \cdot \log(x_2)$ around $\mathbf{x}_0 = (2, 1)^\top$.
(d) $f(\mathbf{x}) = \sin(x_1) + \cos(x_2)$ around $\mathbf{x}_0 = (-\pi, \pi)^\top$.

Solution

There are different ways to write the second order taylor series expansion a at a point $\mathbf{a}$ for multivariate functions f. We will use the following form

$$T(x) = f(\mathbf{a}) + \nabla f(\mathbf{a})^\top(\mathbf{x} - \mathbf{a}) + \frac{1}{2}(\mathbf{x} - \mathbf{a})^\top \mathbf{H}(\mathbf{x} - \mathbf{a})$$

where $\nabla f(\mathbf{a})$ is the gradient of f at $\mathbf{a}$ and $\mathbf{H}$ is the Hessian of f at $\mathbf{a}$. As a reminder, the Hessian is the matrix of second order partial derivatives. So, all we need to do for all of the exercises is evaluate $f(\mathbf{a})$ and compute its gradient and Hessian.

(a) $T(x) = 5 + 15(x - 1) + 15(x - 1)^2$

(b) First, we simplify $f(x, y) = x^2(y^3 + 1)$ and compute $f(3, 2) = 81$ and then we compute the gradient and Hessian of f resulting in

$$\nabla f(x, y) = \begin{pmatrix} 2x(y^3 - 1) \\ x^2(3y^2 - 1) \end{pmatrix}, \quad \mathbf{H} = \begin{pmatrix} 2(y^3 - 1) & 6xy^2 \\ 6xy^2 & 6x^2y \end{pmatrix}.$$

(c) This one is similar to (b) but we have to be careful with the logarithm. First, we have $f(\mathbf{x}_0) = 0$ because $\log(1) = 0$. Then, we compute the gradient and Hessian of f resulting in

$$\nabla f(\mathbf{x}) = \begin{pmatrix} 3x_1^2 x_2 \log(x_2) \\ x_1^3(1 + \log(x_2)) \end{pmatrix}, \quad \mathbf{H} = \begin{pmatrix} 6x_1x_2\log(x_2) & 3x_1^2(1 + \log(x_2)) \\ 3x_1^2(1 + \log(x_2)) & x_1^3 x_2^{-1} \end{pmatrix}$$

(d) Evaluating the trigonometric functions here is simpler than it seems because they are evaluated at π and $-\pi$. We get $f(\mathbf{x}_0) = \sin(-\pi) + \cos(\pi) = 0 - 1 = -1$ and

$$\nabla f(\mathbf{x}) = \begin{pmatrix} \cos(x_1) \\ -\sin(x_2) \end{pmatrix}, \quad \mathbf{H} = \begin{pmatrix} -\sin(x_1) & 0 \\ 0 & -\cos(x_2) \end{pmatrix}.$$

Exercise 2 - Eigenvalues, Eigenvectors

You are given the sets of eigevalues and eigenvectors. Compute the corresponding matrix.

(a) $\lambda_1 = 2$, $\lambda_2 = 3$, $\mathbf{v}_1 = (1, 0)^\top$, $\mathbf{v}_2 = (0, 1)^\top$.
(b) $\lambda_1 = 2$, $\lambda_2 = 3$, $\mathbf{v}_1 = (1, 1)^\top$, $\mathbf{v}_2 = (1, -1)^\top$.

Solution

First, remember that the normalized eigenvectors of a symmetric matrix are orthogonal. Thus, we have

$$\mathbf{e}_i^\top \mathbf{e}_j = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}.$$

Second, for symmetric $\mathbf{A}$, its spectral decomposition is given by $\mathbf{A} = \mathbf{Q}\ \mathbf{Q}^\top$, where $\mathbf{Q}$ is a matrix where each column is an (orthogonal) eigenvector of unit length.

(a) Here, the eigenvectors are already normalized and orthogonal, so we can simply write $\mathbf{Q} = (\mathbf{e}_1, \mathbf{e}_2)$ and $= \mathrm{diag}(\lambda_1, \lambda_2)$. Then, we have

$$\mathbf{A} = \begin{pmatrix} 2 & 0 \\ 0 & 3 \end{pmatrix}$$

(b) Here, we have to normalize the eigenvectors first. Each has length $\sqrt{2}$, so we have to divide each of them by $\sqrt{2}$, i.e. we set $\tilde{\mathbf{e}}_i := \mathbf{e}_i/\sqrt{2}$. With this, we can construct an orthogonal matrix of eigenvalues as $\mathbf{Q} = (\tilde{\mathbf{e}}_1, \tilde{\mathbf{e}}_2)$. The resulting matrix $\mathbf{A}$ is

$$\mathbf{A} = \begin{pmatrix} 2.5 & -0.5 \\ -0.5 & 2.5 \end{pmatrix}$$

Exercise 3 - SGD with Momentum

Implement stochastic gradient descent with momentum and apply it to optimize some elementary functions in 1d and 2d.

Solution

An example implementation could look like this. First we deinfe the objective function and its gradient:

```
def objective(x):
    return x**2

def obj_grad(x):
    return 2*x
```

Then, we implement the momentum optimizer itself:

```
def sgd_with_momentum(obj, grad, x_init, learning_rate, momentum, max_iter):
  x = x_init
  update = 0
  for i in range(max_iter):
    update = momentum * update - learning_rate * grad(x)
    x = x + update

    print('>%d f(%s) = %.5f' % (i, x, obj(x)))
  return x
```

It can now be invoked directly via:

```
sgd_with_momentum(
    objective, obj_grad, x_init=3.0, learning_rate=0.1, momentum=0.5,
    max_iter=20
    )
```