Deep Learning

1. Find the gradient of $f(\mathbf{x}) = \|\mathbf{x}\|^2$, where $\mathbf{x} \in \mathbb{R}^2$.

2. Consider the function $f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^\top A\mathbf{x} + b^\top \mathbf{x}$, where $A \in \mathbb{R}^{d\times d}$, and $b \in \mathbb{R}^d$.

(a) Find the gradient of f.

(b) If $A = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$ and $b = \begin{pmatrix} 2 \\ 3 \end{pmatrix}$. Compute the gradient of f at $\mathbf{x} = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$.

(c) Write Python code to numerically compute the gradient in (b) using the definition of partial derivative. Compare your answer with the one in (b).

(d) Find all the stationary points of f.

3. Consider a function $f : \mathbb{R}^d \to \mathbb{R}$. Let $g(\mathbf{x}) = f(A\mathbf{x})$ for some matrix $A \in \mathbb{R}^{d\times d}$.

(a) What is the relationship between $\nabla g(\mathbf{x})$ and $\nabla f(\mathbf{x})$?

(b) If A is non-singular, show that if $\mathbf{x}$ is a stationary point of g, then $A\mathbf{x}$ is a stationary point of f.

4. Let X and Y be two real-valued random variables such that $Y = 3X + 1$. If $Y \sim N(4, 9)$, what is the distribution of X?

5. Let $X \sim \text{Bernoulli}(p)$ and $Y \sim N(\mu, \sigma^2)$ be two independent random variables. Compute the expected value of $Y \sin X$.

6. Suppose $P(A) = 0.5$, $P(B) = 0.4$, $P(A \cap B) = 0.2$. Compute $P(A \mid B)$.

7. A coin is tossed independently for 25 times, and 15 heads and 10 tails are observed. What is the maximum likelihood estimate for the probability of getting a head?

8. Consider the matrix $A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$. Write Python code to answer the following questions.

(a) What are the eigenvalues and their corresponding eigenvectors for A?

(b) What are A^2 and A^5?

(c) Assume that $X \sim N(\mu, \Sigma)$, where $\mu = (1, 2)^\top$, and $\Sigma = \begin{pmatrix} 1 & 0.3 \\ 0.3 & 1 \end{pmatrix}$, that is, X follows a multivariate normal distribution with mean μ and covariance Σ. Sample 100 observations of $Y = AX$ and draw a scatter plot for them. What is the mean of these observations? What is its difference with the true mean $\mathbb{E}\, Y$?

Solutions

1. Let $\mathbf{x} = (x_1, x_2)$, then we can write $f(\mathbf{x})$ as

$$f(\mathbf{x}) = x_1^2 + x_2^2.$$

The partial derivatives are

$$\frac{\partial f}{\partial x_1} = 2x_1, \qquad \frac{\partial f}{\partial x_2} = 2x_2.$$

The gradient is thus $\nabla f(\mathbf{x}) = \begin{pmatrix} 2x_1 \\ 2x_2 \end{pmatrix} = 2\mathbf{x}$.

2. **(a)** Assume that $A = (a_{ij})$, and $b = (b_1, \ldots, b_d)^\top$, then

$$f(\mathbf{x}) = \frac{1}{2}\sum_{i,j} a_{ij}x_i x_j + \sum_i b_i x_i, \tag{1}$$

where we have omitted the range for i and j for simplicity of notation ($1 \le i, j \le d$ in this case). The terms containing x_i are $\frac{1}{2}a_{ii}x_i^2$, $\frac{1}{2}a_{ij}x_i x_j$ for $j \neq i$, and $\frac{1}{2}a_{ji}x_j x_i$ for $j \neq i$. Thus we have

$$\frac{\partial f(\mathbf{x})}{\partial x_i} = a_{ii}x_i + \frac{1}{2}\sum_{j\neq i} a_{ij}x_j + \frac{1}{2}\sum_{j\neq i} a_{ji}x_j + b_i \tag{2}$$

$$= \frac{1}{2}\sum_{j} a_{ij}x_j + \frac{1}{2}\sum_{j} a_{ji}x_j + b_i \tag{3}$$

$$= \frac{1}{2}a_{i\cdot}^\top \mathbf{x} + \frac{1}{2}a_{\cdot i}^\top \mathbf{x} + b_i, \tag{4}$$

where we use $a_{i\cdot}$ and $a_{\cdot i}$ denote $(a_{i1}, \ldots, a_{id})^\top$ and $(a_{1i}, \ldots, a_{di})^\top$ respectively.
It is easy to verify that the expression in Eq. (4) is exactly the i-th entry in $\frac{1}{2}(A + A^\top)\mathbf{x} + b$. Thus

$$\nabla f(\mathbf{x}) = \frac{1}{2}(A + A^\top)\mathbf{x} + b. \tag{5}$$

(b) Using the formula in (a), we have

$$\nabla f(x) = \frac{1}{2}\left(A + A^\top\right)x + b = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}\begin{pmatrix} 1 \\ 2 \end{pmatrix} + \begin{pmatrix} 2 \\ 3 \end{pmatrix} = \begin{pmatrix} 1 \\ -2 \end{pmatrix} + \begin{pmatrix} 2 \\ 3 \end{pmatrix} = \begin{pmatrix} 3 \\ 1 \end{pmatrix}.$$

(c)

```
import numpy as np

def f(x):
  A = np.array([[1.0, 0.0],
                [0.0, -1.0]])
  b = np.array([2.0, 3.0])
  return 0.5 * x.T @ A @ x + b.T @ x

def numerical_grad(f, x, h=1e-6):
    grad = np.zeros_like(x)
    for i in range(len(x)):
        e = np.zeros_like(x)
        e[i] = 1.0
        grad[i] = (f(x + h * e) - f(x)) / h
    return grad

x = np.array([1.0, 2.0])
grad_numerical = numerical_grad(f, x)
print(grad_numerical)
```

The output of the above code is $\begin{pmatrix} 3.0000005 \\ 0.9999995 \end{pmatrix}$, thus the numerical gradient is about the same as the exact gradient.

(d) To find a stationary point, we set the gradient to 0, that is,

$$\begin{aligned} \frac{1}{2}(A + A^\top)\mathbf{x} + b &= 0 \\ \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}\mathbf{x} + \begin{pmatrix} 2 \\ 3 \end{pmatrix} &= 0 \\ \mathbf{x} = -\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}^{-1}\begin{pmatrix} 2 \\ 3 \end{pmatrix} &= \begin{pmatrix} -2 \\ 3 \end{pmatrix} \end{aligned}$$

Thus there is only one stationary point $\mathbf{x} = \begin{pmatrix} -2 \\ 3 \end{pmatrix}$.

3. (a) Assume that $A = (a_{ij})$. Let $\mathbf{y} = (y_1, \ldots, y_d)^\top = A\mathbf{x}$, then we have $y_j = \sum_k a_{jk}x_k$, thus

$$\frac{\partial g(\mathbf{x})}{\partial x_i} = \frac{\partial f(\mathbf{y})}{\partial x_i} = \sum_j \frac{\partial f(\mathbf{y})}{\partial y_j}\frac{\partial y_j}{\partial x_i} = \sum_j \frac{\partial f(\mathbf{y})}{\partial y_j} a_{ji} = a_{\cdot i}^\top \nabla f(\mathbf{y}). \tag{6}$$

The last expression is exactly the i-th entry of $A^\top \nabla f(\mathbf{y})$. Thus

$$\nabla g(\mathbf{x}) = A^\top \nabla f(\mathbf{y}), \quad \text{where } \mathbf{y} = A\mathbf{x}. \tag{7}$$

As an illustration, consider $f(\mathbf{x}) = ||\mathbf{x} - \mathbf{u}||_2^2$. It is easy to show that $\nabla f(\mathbf{x}) = 2(\mathbf{x} - \mathbf{u})$. Computing the gradient of $g(\mathbf{x}) = ||A\mathbf{x} - \mathbf{u}||_2^2$ is less straightforward. Using the formula above, we have $\nabla g(\mathbf{x}) = 2A^\top(A\mathbf{x} - \mathbf{u})$.

(b) If $\mathbf{x}$ is a stationary point of g, then $\nabla g(\mathbf{x}) = 0$. Since $\nabla g(\mathbf{x}) = A^\top \nabla f(\mathbf{y})$ for $\mathbf{y} = A\mathbf{x}$, we have $A^\top \nabla f(\mathbf{y}) = 0$. Since A is non-singular, $A^\top$ is invertible, thus $\nabla f(\mathbf{y}) = 0$. That is, $\mathbf{y} = A\mathbf{x}$ is a stationary point of f.

4. We have $X = \frac{Y-1}{3}$. A linear function of a normal random variable is normal, thus X follows a normal distribution and we only need to determine the mean and variance of X.

$$\mathbb{E}\, X = \mathbb{E}\, \frac{Y-1}{3} = \frac{\mathbb{E}\, Y - 1}{3} = \frac{4-1}{3} = 1.$$
$$\operatorname{Var} X = \operatorname{Var} \frac{Y-1}{3} = \frac{1}{9} \operatorname{Var} Y = \frac{1}{9} \times 9 = 1.$$

Hence $X \sim N(1, 1)$.

5. Since X and Y are independent, we have

$$\mathbb{E}[Y \sin X] = \mathbb{E}\, Y\, \mathbb{E}[\sin X] = \mu[p \sin 1 + (1-p) \sin 0] = \mu p \sin 1.$$

6. Let p be the probability of getting a head. Then the log-likelihood is given by

$$\ell(p) = \ln[p^{15}(1-p)^{10}] = 15 \ln p + 10 \ln(1-p).$$

To find the maximizer, set the derivative to 0:

$$\frac{15}{p} - \frac{10}{1-p} = 0.$$

Solving the question gives $p = \frac{15}{25} = 0.6$.

7. $P(A \mid B) = \frac{P(A \cap B)}{P(B)} = \frac{0.2}{0.4} = 0.5$.

8. We will use two libraries from the SciPy ecosystem (https://www.scipy.org/ (https://www.scipy.org/)): the `numpy` library for manipulating matrices, and the `matplotlib` library for generating plots.

(a) We can use numpy to compute the eigenvalues and the eigenvectors of A.

```
import numpy as np

A = np.array([[1, 2], [3, 4]])
w, v = np.linalg.eig(A)
```

The eigenvalues are

```
w
```

```
array([-0.37228132,  5.37228132])
```

The corresponding eigenvectors are the column vectors of

```
v
```

```
array([[-0.82456484, -0.41597356],
       [ 0.56576746, -0.90937671]])
```

As an verification, note that Av and $v\text{diag}(w)$ are the same as shown below.

```
A @ v, v * w
```

```
(array([[ 0.30697009, -2.23472698],
        [-0.21062466, -4.88542751]]),
 array([[ 0.30697009, -2.23472698],
        [-0.21062466, -4.88542751]]))
```

(b) A^2 is given below

```
A @ A
```

```
array([[ 7, 10],
       [15, 22]])
```

A^5 is given below

```
A @ A @ A @ A @ A
```

```
array([[1069, 1558],
       [2337, 3406]])
```

Alternatively, we can calculate a power of a matrix as follows.

```
np.linalg.matrix_power(A, 5)
```

```
array([[1069, 1558],
       [2337, 3406]])
```

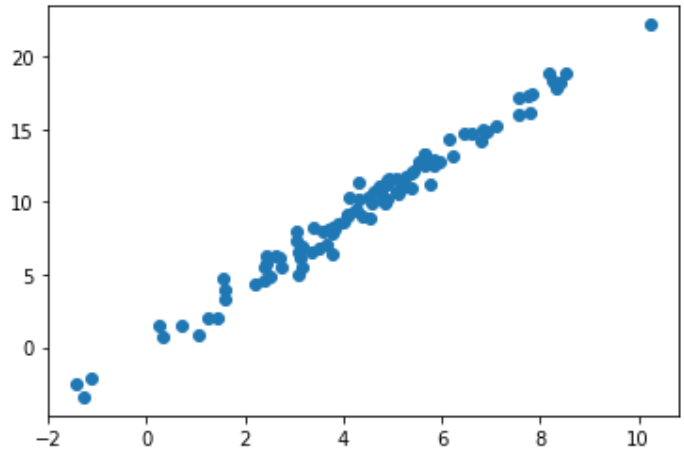
(c) We can draw the required sample as follows

```
np.random.seed(1)
mu_x = np.array([1, 2])
sigma_x = np.array([[1, 0.3], [0.3, 1]])
sample_x = np.random.multivariate_normal(mu_x, sigma_x, 100)
sample_y = (A @ sample_x.T).T
```

`sample_y` has dimension 100x2, and each row is an observation of Y. The scatter plot of these 100 observations are shown below.

```
import matplotlib.pyplot as plt
plt.scatter(sample_y[:,0], sample_y[:, 1])
```

```
<matplotlib.collections.PathCollection at 0x7faa33665358
>
```



The mean of these 100 observations is

```
sample_y.mean(axis=0)
```

```
array([4.44393835, 9.71299651])
```

The true mean is

```
A @ mu_x
```

```
array([ 5, 11])
```

The difference between them is

```
sample_y.mean(axis=0) - A @ mu_x
```

```
array([-0.55606165, -1.28700349])
```

This is large. The error is much smaller if we draw 10,000 observations of Y.

```
sample_x = np.random.multivariate_normal(mu_x, sigma_x, 10000)
sample_y = (A @ sample_x.T).T
sample_y.mean(axis=0) - A @ mu_x
```

```
array([-0.03775093, -0.08992001])
```