2 Optimisation

2.1 Gradient of vector-valued functions

1. First method:

$$J(\mathbf{w}) = \mathbf{a}^\top \mathbf{w} = \sum_{k=1}^{n} a_k w_k \qquad \Longrightarrow \qquad \frac{\partial J(\mathbf{w})}{\partial w_i} = a_i$$

Hence

$$\nabla J(\mathbf{w}) = \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{pmatrix} = \mathbf{a}.$$

Second method:

$$J(\mathbf{w} + \epsilon \mathbf{h}) = \mathbf{a}^\top (\mathbf{w} + \epsilon \mathbf{h}) = \underbrace{\mathbf{a}^\top \mathbf{w}}_{J(\mathbf{w})} + \epsilon \underbrace{\mathbf{a}^\top \mathbf{h}}_{\nabla J^\top \mathbf{h}}$$

Hence we find again $\nabla J(\mathbf{w}) = \mathbf{a}$.

2. First method: We start with

$$J(\mathbf{w}) = \mathbf{w}^\top \mathbf{A} \mathbf{w} = \sum_{i=1}^{n} \sum_{j=1}^{n} w_i A_{ij} w_j$$

Hence,

$$\begin{aligned} \frac{\partial J(\mathbf{w})}{\partial w_k} &= \sum_{j=1}^{n} A_{kj} w_j + \sum_{i=1}^{n} w_i A_{ik} \\ &= \sum_{j=1}^{n} A_{kj} w_j + \sum_{i=1}^{n} w_i (\mathbf{A}^\top)_{ki} \\ &= \sum_{j=1}^{m} \left(A_{kj} + (\mathbf{A}^\top)_{kj} \right) w_j \end{aligned}$$

where we have used that the entry in row i and column k of the matrix $\mathbf{A}$ equals the entry in row k and column i of its transpose $\mathbf{A}^\top$. It follows that

$$\begin{aligned} \nabla J(\mathbf{w}) &= \begin{pmatrix} \sum_{j=1}^{n} \left(A_{1j} + (\mathbf{A}^\top)_{1j} \right) w_j \\ \vdots \\ \sum_{j=1}^{n} \left(A_{nj} + (\mathbf{A}^\top)_{nj} \right) w_j \end{pmatrix} \\ &= (\mathbf{A} + \mathbf{A}^\top) \mathbf{w}, \end{aligned}$$

where we have used that sums like $\sum_j B_{ij} w_j$ are equal to the i-th element of the matrix-vector product $\mathbf{B}\mathbf{w}$.

Second method:

$$\begin{aligned} J(\mathbf{w}+\epsilon\mathbf{h}) &= (\mathbf{w}+\epsilon\mathbf{h})^\top\mathbf{A}(\mathbf{w}+\epsilon\mathbf{h}) \\ &= \mathbf{w}^\top\mathbf{A}\mathbf{w} + \mathbf{w}^\top\mathbf{A}(\epsilon\mathbf{h}) + \epsilon\mathbf{h}^\top\mathbf{A}\mathbf{w} + \underbrace{\epsilon\mathbf{h}^\top\mathbf{A}\epsilon\mathbf{h}}_{O(\epsilon^2)} \\ &= \mathbf{w}^\top\mathbf{A}\mathbf{w} + \epsilon(\mathbf{w}^\top\mathbf{A}\mathbf{h} + \mathbf{w}^\top\mathbf{A}^\top\mathbf{h}) + O(\epsilon^2) \\ &= \underbrace{\mathbf{w}^\top\mathbf{A}\mathbf{w}}_{J(\mathbf{w})} + \epsilon(\underbrace{\mathbf{w}^\top\mathbf{A} + \mathbf{w}^\top\mathbf{A}^\top}_{\nabla J(\mathbf{w})^\top})\mathbf{h} + O(\epsilon^2) \end{aligned}$$

where we have used that $\mathbf{h}^\top\mathbf{A}\mathbf{w}$ is a scalar so that $\mathbf{h}^\top\mathbf{A}\mathbf{w} = (\mathbf{h}^\top\mathbf{A}\mathbf{w})^\top = \mathbf{w}^\top\mathbf{A}^\top\mathbf{h}$. Hence

$$\nabla J(\mathbf{w})^\top = \mathbf{w}^\top\mathbf{A} + \mathbf{w}^\top\mathbf{A}^\top = \mathbf{w}^\top(\mathbf{A}+\mathbf{A}^\top)$$

and

$$\nabla J(\mathbf{w}) = (\mathbf{A}+\mathbf{A}^\top)\mathbf{w}.$$

3. The easiest way to calculate the gradient of $J(\mathbf{w}) = \mathbf{w}^\top\mathbf{w}$ is to use the previous question with $\mathbf{A} = \mathbf{I}$ (the identity matrix). Therefore

$$\nabla J(\mathbf{w}) = \mathbf{I}\mathbf{w} + \mathbf{I}^\top\mathbf{w} = \mathbf{w} + \mathbf{w} = 2\mathbf{w}.$$

4. Note that $||\mathbf{w}||_2 = \sqrt{\mathbf{w}^\top\mathbf{w}}$.

   First method: We use the chain rule

$$\frac{\partial J(\mathbf{w})}{\partial w_k} = \frac{\partial\sqrt{\mathbf{w}^\top\mathbf{w}}}{\partial\mathbf{w}^\top\mathbf{w}}\frac{\partial\mathbf{w}^\top\mathbf{w}}{\partial w_k}$$

   and that

$$\frac{\partial\sqrt{\mathbf{w}^\top\mathbf{w}}}{\partial\mathbf{w}^\top\mathbf{w}} = \frac{1}{2\sqrt{\mathbf{w}^\top\mathbf{w}}}$$

   The derivatives $\partial\mathbf{w}^\top\mathbf{w}/\partial w_k$ were calculated in the question above so that

$$\nabla J(\mathbf{w}) = \frac{1}{2\sqrt{\mathbf{w}^\top\mathbf{w}}}2\mathbf{w} = \frac{\mathbf{w}}{||\mathbf{w}||_2}$$

   Second method: Let $f(\mathbf{w}) = \mathbf{w}^\top\mathbf{w}$. From the previous question, we know that

$$f(\mathbf{w}+\epsilon\mathbf{h}) = f(\mathbf{w}) + \epsilon 2\mathbf{w}^\top\mathbf{h} + O(\epsilon^2).$$

   Moreover,

$$\begin{aligned} \sqrt{z+\epsilon u+O(\epsilon^2)} &= \sqrt{z} + \frac{1}{2\sqrt{z}}(\epsilon u + O(\epsilon^2)) + O(\epsilon^2) \\ &= \sqrt{z} + \epsilon\frac{1}{2\sqrt{z}}u + O(\epsilon^2) \end{aligned}$$

   With $z = f(\mathbf{w})$ and $u = 2\mathbf{w}^\top\mathbf{h}$, we thus obtain

$$\begin{aligned} J(\mathbf{w}+\epsilon\mathbf{h}) &= \sqrt{f(\mathbf{w}+\epsilon\mathbf{h})} \\ &= \sqrt{f(\mathbf{w})} + \epsilon\frac{1}{2\sqrt{f(\mathbf{w})}}2\mathbf{w}^\top\mathbf{h} + O(\epsilon^2) \\ &= \sqrt{f(\mathbf{w})} + \epsilon\frac{\mathbf{w}^\top}{\sqrt{f(\mathbf{w})}}\mathbf{h} + O(\epsilon^2) \\ &= J(\mathbf{w}) + \epsilon\frac{\mathbf{w}^\top}{\sqrt{||\mathbf{w}||_2}}\mathbf{h} + O(\epsilon^2) \end{aligned}$$

so that

$$\nabla J(\mathbf{w}) = \frac{\mathbf{w}}{||\mathbf{w}||_2}.$$

5. Either the chain rule or the approach with the Taylor expansion can be used to deal with the outer function f. In any case:

$$\nabla J(\mathbf{w}) = f'(||\mathbf{w}||_2)\nabla||\mathbf{w}||_2 = f'(||\mathbf{w}||_2)\frac{\mathbf{w}}{||\mathbf{w}||_2},$$

where f' is the derivative of the function f.

2.2 Newton's method

1. We first write f as

$$\begin{aligned} f(\mathbf{w}) &= c + \mathbf{g}^\top(\mathbf{w} - \mathbf{w}_0) + \frac{1}{2}(\mathbf{w} - \mathbf{w}_0)^T\mathbf{H}(\mathbf{w} - \mathbf{w}_0) \\ &= c - \mathbf{g}^\top\mathbf{w}_0 + \frac{1}{2}\mathbf{w}_0^\top\mathbf{H}\mathbf{w}_0 + \\ &\quad \mathbf{g}^\top\mathbf{w} + \frac{1}{2}\mathbf{w}^\top\mathbf{H}\mathbf{w} - \frac{1}{2}\mathbf{w}_0^\top\mathbf{H}\mathbf{w} - \frac{1}{2}\mathbf{w}^\top\mathbf{H}\mathbf{w}_0 \end{aligned}$$

Using now that $\mathbf{w}^\top\mathbf{H}\mathbf{w}_0$ is a scalar and that $\mathbf{H}$ is symmetric, we have

$$\mathbf{w}^\top\mathbf{H}\mathbf{w}_0 = (\mathbf{w}^\top\mathbf{H}\mathbf{w}_0)^\top = \mathbf{w}_0^\top\mathbf{H}^\top\mathbf{w} = \mathbf{w}_0^\top\mathbf{H}\mathbf{w}$$

and hence

$$f(\mathbf{w}) = \text{const} + (\mathbf{g}^\top - \mathbf{w}_0^\top\mathbf{H})\mathbf{w} + \frac{1}{2}\mathbf{w}^\top\mathbf{H}\mathbf{w}$$

With the results from 2.1 and the fact that $\mathbf{H}$ is symmetric, we thus obtain

$$\begin{aligned} \nabla f(\mathbf{w}) &= \mathbf{g} - \mathbf{H}^\top\mathbf{w}_0 + \frac{1}{2}(\mathbf{H}^\top\mathbf{w} + \mathbf{H}\mathbf{w}) \\ &= \mathbf{g} - \mathbf{H}\mathbf{w}_0 + \mathbf{H}\mathbf{w} \end{aligned}$$

The expansion of $f(\mathbf{w})$ due to the $\mathbf{w} - \mathbf{w}_0$ terms is a bit tedious. It is simpler to note that gradients define a linear approximation of the function. We can more efficiently deal with $\mathbf{w} - \mathbf{w}_0$ by changing the coordinates and determine the linear approximation of f as a function of $\mathbf{v} = \mathbf{w} - \mathbf{w}_0$, i.e. locally around the point $\mathbf{w}_0$. We then have

$$\begin{aligned} \tilde{f}(\mathbf{v}) &= f(\mathbf{v} + \mathbf{w}_0) \\ &= c + \mathbf{g}^\top\mathbf{v} + \frac{1}{2}\mathbf{v}^\top\mathbf{H}\mathbf{v} \end{aligned}$$

With 2.1, the derivative is

$$\nabla_\mathbf{v}\tilde{f}(\mathbf{v}) = \mathbf{g} + \mathbf{H}\mathbf{v}$$

and the linear approximation becomes

$$\tilde{f}(\mathbf{v} + \epsilon\mathbf{h}) = c + \epsilon(\mathbf{g} + \mathbf{H}\mathbf{v})^\top\mathbf{h} + O(\epsilon^2)$$

The linear approximation for $\tilde{f}$ determines a linear approximation of f around $\mathbf{w}_0$, i.e.

$$f(\mathbf{w} + \epsilon\mathbf{h}) = \tilde{f}(\mathbf{w} - \mathbf{w}_0 + \epsilon\mathbf{h}) = c + \epsilon(\mathbf{g} + \mathbf{H}(\mathbf{w} - \mathbf{w}_0))^\top\mathbf{h} + O(\epsilon^2)$$

so that the derivative for f is

$$\nabla_\mathbf{w} f(\mathbf{w}) = \mathbf{g} + \mathbf{H}(\mathbf{w} - \mathbf{w}_0) = \mathbf{g} - \mathbf{H}\mathbf{w}_0 + \mathbf{H}\mathbf{w},$$

which is the same result as before.

2. We set the gradient to zero and solve for $\mathbf{w}$:

$$\mathbf{g} + \mathbf{H}(\mathbf{w} - \mathbf{w}_0) = 0 \quad \leftrightarrow \quad \mathbf{w} - \mathbf{w}_0 = -\mathbf{H}^{-1}\mathbf{g}$$

so that

$$\mathbf{w}^* = \mathbf{w}_0 - \mathbf{H}^{-1}\mathbf{g}.$$

As we assumed that $\mathbf{H}$ is positive definite, the inverse $\mathbf{H}$ exists (and is positive definite too).

Let us consider f as a function of $\mathbf{v}$ around $\mathbf{w}^*$, i.e. $\mathbf{w} = \mathbf{w}^* + \mathbf{v}$. With $\mathbf{w}^* + \mathbf{v} - \mathbf{w}_0 = -\mathbf{H}^{-1}\mathbf{g} + \mathbf{v}$, we have

$$f(\mathbf{w}^* + \mathbf{v}) = c + \mathbf{g}^\top(-\mathbf{H}^{-1}\mathbf{g} + \mathbf{v}) + \frac{1}{2}(-\mathbf{H}^{-1}\mathbf{g} + \mathbf{v})^\top\mathbf{H}(-\mathbf{H}^{-1}\mathbf{g} + \mathbf{v})$$

Since $\mathbf{H}$ is positive definite, we have that $(-\mathbf{H}^{-1}\mathbf{g} + \mathbf{v})^\top\mathbf{H}(-\mathbf{H}^{-1}\mathbf{g} + \mathbf{v}) > 0$ for all $\mathbf{v}$. Hence, as we move away from $\mathbf{w}^*$, the function increases quadratically, so that $\mathbf{w}^*$ minimises $f(\mathbf{w})$.

3. The equation

$$\mathbf{w}^* = \mathbf{w}_0 - \mathbf{H}^{-1}\mathbf{g}.$$

corresponds to one update step in Newton's method where $\mathbf{w}_0$ is the current value of $\mathbf{w}$ in the optimisation of $J(\mathbf{w})$ and $\mathbf{w}^*$ is the updated value. In practice rather than determining the inverse $\mathbf{H}^{-1}$, we solve

$$\mathbf{H}\mathbf{p} = \mathbf{g}$$

for $\mathbf{p}$ and then set $\mathbf{w}^* = \mathbf{w}_0 - \mathbf{p}$. The vector $\mathbf{p}$ is the search direction, and it is possible include a step-length α so that the update becomes $\mathbf{w}^* = \mathbf{w}_0 - \alpha\mathbf{p}$. The value of α may be set by hand or can be determined via line-search methods.

4. To compute the first 4 iterations of Newton's method for the function $f(x) = \frac{x^4}{4} - \frac{x^3}{3} - x$ starting from $x_0 = 1$, we need to find the derivative of $f(x)$ and apply the iteration formula.

First, let's find the derivative of $f(x)$:

$$f'(x) = \frac{d}{dx}\left(\frac{x^4}{4} - \frac{x^3}{3} - x\right)$$

Using the power rule and the constant rule, we can simplify this to:

$$f'(x) = x^3 - x^2 - 1$$

the second derivative is given by:

$$f''(x) = 3x^2 - 2x$$

Now, let's apply the iteration formula for Newton's method:

$$x_{n+1} = x_n - \frac{f'(x_n)}{f''(x_n)}$$

Since we only need the first 4 iterations, we can calculate x_1, x_2, x_3, and x_4.

Iteration 1:

$$x_1 = x_0 - \frac{f'(x_0)}{f''(x_0)}$$

Substituting the values:

$$x_1 = 1 - \frac{(1^3 - 1^2 - 1)}{(3 \cdot 1^2 - 2 \cdot 1)}$$

Simplifying:

$$x_1 = 1 + \frac{1 - 1 + 1}{3 - 2} = 1 + \frac{1}{1} = 2$$

Iteration 2:

$$x_2 = x_1 - \frac{f'(x_1)}{f''(x_1)}$$

Substituting the values:

$$x_2 = 2 - \frac{(2^3 - 2^2 - 1)}{(3 \cdot 2^2 - 2 \cdot 2)} = 1.625$$

Iteration 3:

$$x_3 = 1.625 - \frac{f'(x_2)}{f''(x_2)}$$

Substituting the values:

$$x_3 = 1.625 - \frac{(1.625^3 - 1.625^2 + 1)}{(3 \cdot 1.625^2 - 1.625 \cdot 2)} = 1.486$$

(result is rounded) Iteration 4:

$$x_4 = x_3 - \frac{f'(x_3)}{f''(x_3)}$$

Substituting the values:

$$x_2 = 1.485 - \frac{(1.486^3 - 1.486^2 + 1)}{(3 \cdot 1.486^2 - 2 \cdot 1.486)} = 1.466$$

5. Let's compute the first 4 iterations of gradient descent for the function $f(x) = \frac{x^4}{4} - \frac{x^3}{3} - x$ starting from $x_0 = 1$ with a learning rate of $\alpha = 0.1$.

Iteration 1:

$$x_1 = x_0 - \alpha f'(x_0)$$

$$x_1 = 1 - 0.1 \cdot (x_0^3 - x_0 - 1) = 1.1$$

Iteration 2:

$$x_2 = x_1 - \alpha f'(x_1)$$

$$x_2 = 1.1 - 0.1 \cdot (x_1^3 - x_1^2 - 1) = 1.188$$

Iteration 3:

$$x_3 = x_2 - \alpha f'(x_2)$$

$$x_3 = 1.188 - 0.1 \cdot (x_2^3 - x_2^2 - 1) = 1.261$$

Iteration 4:

$$x_4 = x_3 - \alpha f'(x_3)$$

$$x_4 = 1 - 0.1 \cdot (x_3^3 - x_3^2 - 1) = 1.319$$

(Always continue with the rounded results)

6. Difference between Newton's Method and Gradient Descent

   The main difference between Newton's method and gradient descent lies in the way they update the parameters during optimization.

   Gradient Descent: Gradient descent is an iterative optimization algorithm that updates the parameters in the direction of the negative gradient of the objective function. It uses the first-order derivative (gradient) of the function to determine the direction of steepest descent and updates the parameters by taking steps proportional to the negative gradient. The learning rate determines the step size in each iteration.

   Newton's Method: Newton's method is also an iterative optimization algorithm, but it uses the second-order derivative (Hessian) of the objective function in addition to the gradient. It approximates the objective function locally as a quadratic function and finds the minimum of this quadratic function. Newton's method updates the parameters by taking steps proportional to the inverse of the Hessian matrix multiplied by the gradient. This allows it to take into account curvature information and often leads to faster convergence compared to gradient descent.

   In machine learning, the choice between Newton's method and gradient descent depends on the specific problem and the characteristics of the objective function.

   Gradient Descent: Gradient descent is generally preferred when the objective function is convex, differentiable, and the number of parameters is large. It is computationally efficient and can handle large-scale optimization problems(complexity is $\mathcal{O}(n)$)). Gradient descent is also more robust to noise and outliers in the data. However, it may converge slowly in some cases and can get stuck in local minima.

   Newton's Method: Newton's method is preferred when the objective function is strongly convex, twice differentiable, and the number of parameters is relatively small. It can converge faster than gradient descent as it takes into account curvature information. Newton's method is particularly effective when the initial guess is close to the optimal solution. However, it requires computing and inverting the Hessian matrix, which can be computationally expensive and memory-intensive for large-scale problems(complexity is $\mathcal{O}(n)^2$). It is also more sensitive to noise and outliers in the data