Week 3 Tutorial 1

(1.1) *Separating Hyperplane Theorem*
We recall the separating hyperplane Theorem 3.10 discussed in Lecture 1 Week 2.

Construct counter examples if the set C

- is not convex.
- is not closed.

For TAs: Please go through the proof of Theorem 3.10. The counter examples are straight forward. For example a moon shape object in 2D where the point x is inside the curved area. If the set is not closed than for any x on the boundary of the open set, there is a hyperplane but not necessarily a gap.

(1.2) *Convex cones*

- Let $\mathbf{S}^n$ denote the set of symmetric $n \times n$ matrices, that is

$$\mathbf{S}^n = \{\mathbf{A} \in \mathbf{R}^{n\times n} \colon \mathbf{A} = \mathbf{A}^T\}.$$

and by $\mathbf{S}^n_+$ the set of symmetric positive semi-definite matrices.
What is the dimension of $\mathbf{S}^n$. Show that $\mathbf{S}^n$is a convex cone.

Solution: The dimension is $n(n+1)/2$; for the cone property we have to show that if $\theta_1, \theta_2 \geq 0$ and $\mathbf{A}, \mathbf{B} \in \mathbf{S}^n_+$ then $\theta_1 + \mathbf{A} + \theta_2\mathbf{B} \in \mathbf{S}^n_+$. This follows from

$$\mathbf{x}^T(\theta_1\mathbf{A} + \theta_2\mathbf{B})\mathbf{x} = \theta_1\mathbf{x}^T\mathbf{A}\mathbf{x} + \theta_2\mathbf{x}^T\mathbf{B}\mathbf{x} \geq 0$$

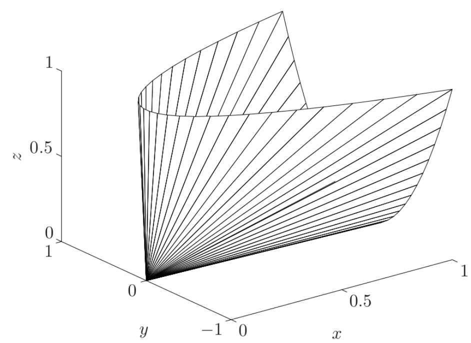
if $\mathbf{A}$ and $\mathbf{B}$ are positive semi-definite.

- The second order cone (also known as the ice cream cone) is the norm cone defined for the Euclidean norm, that is

$$\begin{aligned} C &= \{(\mathbf{x}, t) \in \mathbb{R}^{n+1} \colon \|\mathbf{x}\|_2 \leq t\} \\ &= \{\begin{pmatrix}\mathbf{x}\\ t\end{pmatrix} \colon \begin{pmatrix}\mathbf{x}\\ t\end{pmatrix}^T \begin{pmatrix} I & 0\\ 0 & -1\end{pmatrix} \begin{pmatrix}\mathbf{x}\\ t\end{pmatrix} \leq 0, t \geq 0\}. \end{aligned}$$

Show that C is indeed a cone. Plot the cone in $\mathbb{R}^3$, that is the set $\{(x_1, x_2, t) \colon (x_1^2 + x_2^2)^{\frac{1}{2}} \leq t\}$.

Solution: Follows immediately.



(1.3) *Operations that preserve convexity of functions*

- Affine mappings: Let $f\colon \mathbb{R}^n \to \mathbb{R}$, $\mathbf{A} \in \mathbb{R}^{n\times m}$ and $\mathbf{b} \in \mathbb{R}^n$. Define $g\colon \mathbb{R}^m \to \mathbb{R}$ by

$$g(\mathbf{x}) = f(\mathbf{A}\mathbf{x} + \mathbf{b})$$

with dom $g = \{\mathbf{x}\colon \mathbf{A}\mathbf{x} + \mathbf{b} \in \text{ dom } f\}$. Show that if f is convex, so is g.

Solution: trivial.

- Pointwise maximum: Let f_1 and f_2 be convex functions and define their pointwise maximum as

$$f(x) = \max\{f_1(x), f_2(x)\}$$

with dom $f =$ dom $f_1 \cap \text{dom} f_2$ also convex. Show that f is convex.

Solution:

$$\begin{aligned} f(\theta x + (1-\theta)y) &= \max\left(f_1(\theta x + (1-\theta)y), f_2(\theta x + (1-\theta)y)\right) \\ &\le \max\left(\theta f_1(x) + (1-\theta)f_1(y), \theta f_2(x) + (1-\theta)f_2(y)\right) \\ &\le \max\left(f_1(x), f_2(x)\right) + (1-\theta)\max\left(f_1(y), f_2(y)\right) \\ &= \theta f(s) + (1-\theta)f(y). \end{aligned}$$

- Scalar composition: Let $h\colon \mathbb{R} \to \mathbb{R}$ and $g\colon \mathbb{R} \to \mathbb{R}$ and $f = h \circ g : \mathbb{R} \to \mathbb{R}$ be defined as

$$f(x) = h(g(x)) \qquad \text{dom} f = \{x \in \text{dom } g\colon g(x) \in \text{dom } h\}.$$

Furthermore assume that h and g are twice differentiable.
Discuss under which conditions on h and g the function f is convex.

Solution: The second derivative of f is given by

$$f''(x) = h''(g(x))g'(x)^2 + h'(g(x))g''(x).$$

We see that

- f is convex if h is convex and non-decreasing, and g is convex.
- f is convex if h is convex and non-increasing, and g is concave.

(1.4) *Stochastic gradient descent (SDG)*
A common situation in machine learning is that the objective function is of the form

$$\min_x \frac{1}{N} \sum_{i=1}^{N} f_i(x)$$

where f_i is an individual loss function associated to the particular data point x_i and $N \in \mathbb{N}$. In gradient descent a full step iterates $\mathbf{x}_k = (x_{k,1}, \dots x_{k,n})$, $k = 1, 2, 3, \dots$ would be updated according to

$$\mathbf{x}_k = \mathbf{x}_{k-1} - \frac{\alpha}{N} \sum_{i=1}^{N} \nabla f_i(\mathbf{x}_{k-1})$$

where $f_i : \mathbb{R}^n \to \mathbb{R}$, $i = 1, \dots N$.

In SDG we update iterates $\mathbf{x}_k$ based on the descent in one component only, that is

$$\mathbf{x}_k = \mathbf{x}_{k-1} - \alpha_k \nabla f_{i_k}(\mathbf{x}_{k-1}) \qquad \text{for } k = 1, 2, \dots$$

where $i_k \in \{1, 2 \dots N\}$ is a randomly chosen index at iteration k.

A common technique in SDG is mini-batching, where one chooses a random subset $I_k \subseteq \{1, 2, \dots n\}$ with size $|I_k| = M \ll N$. Hence we have the following update rule

$$\mathbf{x}_k = \mathbf{x}_{k-1} - \frac{\alpha_k}{M} \sum_{i \in I_k} \nabla f_{i_k}(\mathbf{x}_{k-1})$$

Solution: Computational complexity for SDG is Nn in each step, for SDG it's only n and for mini-batch MN. Calculating the gradient becomes quite expensive if you have large N and a large n.

For TAs: Please go through the Jupyter notebook.