Lectures 22, 23: Submodular minimization via gradient descent

Last time we introduced submodular functions.

Recall that a set function $f: 2^V \to \mathbb{R}$ is submodular if

$$f(A) + f(B) \geq f(A \cap B) + f(A \cup B) \quad \text{for all subsets } A, B \subseteq V.$$

Equivalently,

$$f(A \cup \{v\}) - f(A) \geq f(B \cup \{v\}) - f(B) \quad \text{for all } A \subseteq B \text{ and } v \notin B.$$

Today we will focus on one of the most basic submodular optimization problems: submodular minimization (SFM).

Since we will be interested in designing algorithms, we must first discuss how is the function given to us. Note that this is important, since f is a set function defined on exponentially many sets, we cannot write down its values, since that will take exponential time. Instead, we will assume the most general representation model, namely the value oracle model.

<u>Representation of f.</u> We will assume that f is given as a <u>value oracle</u>, i.e, we are given access to a (black box) oracle that takes as input a set S and returns the value $f(S)$. Thus we can query the oracle to find the value on any set.

We will use EO to denote the running time of a single call to the evaluation oracle.

<u>Submodular Function Minimization (SFM)</u>: Given a submodular function f (represented as a value oracle), the goal is to find a set S that has minimum value: $\min_{S \subseteq V} f(S)$.

The SFM problem is inherently discrete, but as we will see shortly we can formulate it as a convex optimization problem.

This formulation is based on a fundamental connection between submodularity and convexity that we now establish.

Submodular functions and convexity

Let us think about how we can turn the SFM problem into a continuous optimization problem.

The main obstacle is that f is a discrete function defined on sets.

We can equivalently think of f as being defined on integral vectors whose entries are either 0 or 1, that is, $f : \{0,1\}^V \to \mathbb{R}$.

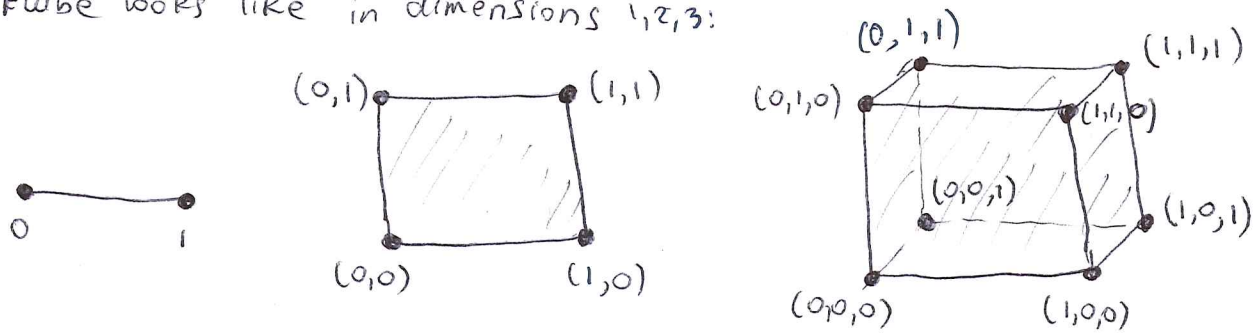
This is because we can view a set S as a vector $1_S \in \{0,1\}^V$; the vector 1_S is called the indicator (or characteristic) vector of S:

$$(1_S)_v \equiv \begin{cases} 1 & \text{if } v \in S \\ 0 & \text{otherwise} \end{cases}$$

In words, 1_S has an entry of 1 in each coordinate $v \in S$ and it has 0 in all other coordinates.

Thus we can think of f as $f : \{0,1\}^V \to \mathbb{R}$, $f(x) \equiv f(\{v : x_v = 1\})$.

The set $\{0,1\}^V$ is sometimes called the set of vertices of the boolean hypercube $[0,1]^V$ of dimension $n \equiv |V|$. Here is how the boolean hypercube looks like in dimensions 1,2,3:



$n = 1$ segment $[0,1]$ $\qquad$ $n = 2$ square $[0,1]^2$ $\qquad$ $n = 3$ cube $[0,1]^3$

The submodular function f is defined only on the vertices of the boolean hypercube $[0,1]^V$. In order to get a continuous formulation, it is very useful to <u>extend</u> f so that it takes on values everywhere on $[0,1]^V$ and not just at the vertices. More precisely, we have the following definition.

<u>Def</u>: (continuous extension on $[0,1]^V$). A function $\hat{f}: [0,1]^V \to \mathbb{R}$ is a continuous extension of f to the entire boolean hypercube if

- $\hat{f}$ is defined everywhere on $[0,1]^V$
- $\hat{f}$ and f agree on the vertices of $[0,1]^V$:

$$\hat{f}(x) = f(x) \quad \text{for all } x \in \{0,1\}^V.$$

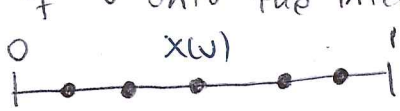
There are several continuous extensions that have been defined and studied. For our purposes, the most useful extension is the Lovász extension that is defined as follows.

<u>Def</u>: (Lovász extension of f). The Lovász extension of f is the function $\hat{f}: [0,1]^V \to \mathbb{R}$ such that, for all $x \in [0,1]^V$,

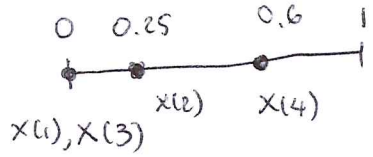
$$\hat{f}(x) = \mathop{\mathbb{E}}_{\theta \sim_R [0,1]} \left[f(\{v : x(v) \geq \theta\}) \right].$$

where $\theta \sim_R [0,1]$ is a number chosen uniformly at random from $[0,1]$.

This definition of the Lovász extension is based on the following very natural <u>rounding</u> strategy that rounds a (fractional) vector $x \in [0,1]^V$ to an integral vector $\hat{x} \in \{0,1\}^V$. Let us think of $x \in [0,1]^V$ as an embedding of V onto the interval $[0,1]$ that puts $v \in V$ at $x(v)$:



For example, consider $V = \{1,2,3,4\}$ and $x = (0, 0.25, 0, 0.6)$

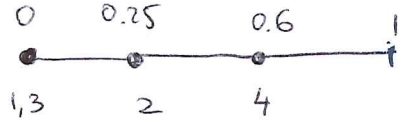


Given a value $\theta \in [0,1]$, we can round x to an integral vector x^θ as follows:

$$(x^\theta)(i) = \begin{cases} 1 & \text{if } x(i) \geq \theta \\ 0 & \text{otherwise} \end{cases}$$

Thus every coordinate of x that is at least θ gets rounded up to 1 and every coordinate that is less than θ gets rounded down to 0.

Then $\hat{f}(x) = \mathbb{E}_{\theta \sim_R [0,1]} [f(x^\theta)]$ is the expected value of the integral vector x^θ that we obtain by rounding based on a threshold chosen uniformly at random in $[0,1]$. Let us see what integral vectors we get on the example above:



$$\hat{f}((0, 0.25, 0, 0.6)) = 0.25 \cdot f(\{2,4\}) + (0.6 - 0.25) \cdot f(\{4\}) + (1 - 0.6)\, f(\emptyset)$$

$$= 0.25\, f(\{2,4\}) + 0.35\, f(\{4\}) + 0.4\, f(\emptyset)$$

Thus $\hat{f}(x)$ is a convex combination of values on sets defined by the permutation of V defined by x.

More precisely, let $\pi : [n] \to V$ be a permutation of V such that

$$x(\pi(1)) \geq x(\pi(2)) \geq \cdots \geq x(\pi(n)) \quad \text{where } n \equiv |V|$$

(in the example above, 4 2 3 1 and 4 2 1 3 are the permutations)

Let $P_0, P_1, \dots, P_n$ be the prefix sets of this permutation.

$$\begin{cases} P_0 = \emptyset \\ P_i = \{\pi(1), \pi(2), \dots, \pi(i)\} \quad \text{for all } 1 \le i \le n. \end{cases}$$

Then we can write $\hat{f}(x)$ as:

$$\hat{f}(x) = \sum_{i=1}^{n-1} (x(\pi(i)) - x(\pi(i+1)))\, f(P_i) + x(\pi(n)) \cdot f(P_n) + (1 - x(\pi(1)))\, f(P_0)$$

$$= \sum_{i=1}^{n} x(\pi(i))\,(f(P_i) - f(P_{i-1})) + f(P_0)$$

A common assumption is that $f(\emptyset) = 0$, which is without loss of generality; if $f(\emptyset) \neq 0$, we can "shift" the function by $f(\emptyset)$, i.e., consider the function

$$f'(S) = f(S) - f(\emptyset) \quad \text{for all } S \subseteq V.$$

The shifted function f' is submodular, and thus we can equivalently minimize f'.

In the remainder, we will assume that $f(\emptyset) = 0$ (such a function is called normalized). This will be convenient when we talk about certain polytopes associated with f.

Remarkably, the Lovász extension is convex if f is submodular, and this is in fact an if and only if statement.

<u>Thm (Lovász)</u> The Lovász extension $\hat{f}$ of f is $\boxed{\text{convex}}$ if and only if the function f is submodular.

This theorem will follow from things we will show later in the lecture.

To summarize, the Lovász extension has the following nice properties:

- $\hat{f}$ is convex
- $\hat{f}(x)$ can be evaluated in $O(n \cdot EO + n \log n)$ time:
  we first sort the coordinates of x in $O(n \log n)$ time and then we call the evaluation oracle to find $f(P_i)$ for each prefix set P_i of the sorted permutation.
- $\min_{x \in [0,1]^V} \hat{f}(x)$ is an exact formulation for $\min_{S \subseteq V} f(S)$:

  given an optimal solution $x^* \in \operatorname{argmin}_{x \in [0,1]^V} \hat{f}(x)$, we can recover an optimal solution $S^* \in \operatorname{argmin}_{S \subseteq V} f(S)$ in $O(n \cdot EO + n \log n)$ time

  As above, we sort x^* and evaluate all the prefix sets.
  We take S^* to be a prefix set of minimum value, $S^* \in \operatorname{argmin}_{P_i} f(P_i)$

  The correctness of this procedure follows from the definition of $\hat{f}(x^*)$:
  If we take a random threshold $\theta \in_R [0,1]$ and round x^* to an integral set S_θ, we have that $\mathbb{E}_{\theta \in_R [0,1]}[f(S_\theta)] = \hat{f}(x^*)$

  Since the best prefix set has value at most $\mathbb{E}_{\theta \in_R [0,1]}[f(S_\theta)]$, it follows that it is optimal.

  (Clearly, $\hat{f}(x^*) \leq f(S^*)$ where $S^* \in \operatorname{argmin}_{S \subseteq V} f(S)$, since 1_{S^*} is a candidate solution to $\min_{x \in [0,1]} \hat{f}(x)$).

Thus, in order to solve the SFM problem, it suffices to solve the convex minimization problem $\min_{x \in [0,1]^V} \hat{f}(x)$.

Now let us think back on the toolkit we have developed: which convex optimization techniques can we use here?

Since this is a constrained convex optimization problem, the answer depends on two things: what properties does the objective have, and what properties does the constraint set have?

The latter is easy to answer: the constraint set is $\mathcal{X} = [0,1]^V$, which is a very simple "box" constraint that decomposes along the individual coordinates. In particular, it is easy to project onto $\mathcal{X} = [0,1]^V$.

Projection onto $\mathcal{X}$ with respect to the ℓ_2-norm:

Recall that the ℓ_2-norm projection onto $\mathcal{X}$ is

$$\Pi_{\mathcal{X}}(x) = \underset{y \in \mathcal{X}}{\operatorname{argmin}} \|x-y\|_2$$

If $\mathcal{X} = [0,1]^V$, we can project a point $x \in \mathbb{R}^V$ onto $\mathcal{X} = [0,1]^V$ as follows.

Let us first think about the 1-dimensional setting: given $x \in \mathbb{R}$, find its closest point in $[0,1]$:

Clearly, the projection maps $x \leq 0$ to 0 and $x \geq 1$ to 1.

This immediately extends to any dimensions, since the constraint $y \in [0,1]^V$ decomposes along the dimensions. Thus the projection of x maps each coordinate $x(i)$ of x to 0 if $x(i) \leq 0$ and to 1 if $x(i) \geq 1$

(if $x(i)$ is already in $[0,1]$, the coordinate remains unchanged).

Formally,

$$\left(\Pi_{[0,1]^V}(x)\right)_i = \begin{cases} 0 & \text{if } x_i < 0 \\ 1 & \text{if } x_i > 1 \\ x_i & \text{if } x_i \in [0,1] \end{cases}$$

Given x, we can compute $\Pi_{[0,1]^V}(x)$ in $O(n)$ time where $n = |V|$.

Thus it remains to see what we can say about the objective function.
Unfortunately, the Lovász extension is not very nice: it is not differentiable everywhere and it is not smooth or strongly convex.
Thus the main algorithm in our toolkit is only the projected subgradient descent algorithm.

Let us first recall the projected subgradient descent method. and then see how to apply it to solve $\min_{x \in [0,1]^V} \hat{f}(x)$.

<u>Recap: projected subgradient descent algorithm</u>

Consider the convex optimization problem $\min_{x \in \mathcal{X}} f(x)$.

The projected subgradient descent algorithm is the following:

<u>initialize</u>: $x_1 \in \mathcal{X}$

<u>iterate $(t \geq 1)$</u>: $y_{t+1} = x_t - \eta g_t$ where $g_t \in \partial f(x_t)$

$$x_{t+1} = \Pi_{\mathcal{X}}(y_{t+1}) = \arg\min_{x \in \mathcal{X}} \|x - (x_t - \eta g_t)\|_2^2$$

The analysis we saw in class gives us the following guarantee for the projected subgradient descent algorithm.

Thm (Projected subgradient descent guarantee)

Let B^2 be an upper bound on the ℓ_2^2-norm of the subgradients used by the algorithm: $\|g_t\|_2^2 \leq B^2$ for all $t \leq T$.

Let $R^2 = \sup_{x \in \mathcal{X}} \frac{1}{2}\|x\|_2^2$.

If we run the projected subgradient descent algorithm with:

$$x_1 = \arg\min_{x \in \mathcal{X}} \|x\|_2^2$$

$$\eta = \frac{R}{B}\sqrt{\frac{2}{T}}$$

After T iterations, we have:

$$f\left(\frac{1}{T}\sum_{t=1}^{T} x_t\right) - \min_{x \in \mathcal{X}} f(x) \leq RB\sqrt{\frac{2}{T}}$$

Thus we arrive at an ε-optimal solution in $T = O\left(\frac{R^2B^2}{\varepsilon^2}\right)$ iterations.

Thus in order to apply the projected subgradient algorithm, we need to give an algorithm that finds subgradients as efficiently as possible and upper bound their ℓ_2^2-norm. Note that $\sup_{x \in [0,1]^V} \frac{1}{2}\|x\|_2^2 = \frac{n}{2}$

so $R^2 = \frac{n}{2}$, where $n = |V|$ as usual.

Subgradients of the Lovász extension

Let $x \in [0,1]^V$. Recall that a point $g \in \mathbb{R}^V$ is a subgradient of $\hat{f}$ at x if we have

$$\hat{f}(y) \geq \hat{f}(x) + \langle g, y-x \rangle \quad \text{for all } y \in [0,1]^V$$

We can show that we can construct a subgradient $g \in \partial \hat{f}(x)$ using the permutations that we saw before.

Let us say that a permutation π of V is $\boxed{\text{consistent with } x}$ if $x(\pi(1)) \geq x(\pi(2)) \geq \cdots \geq x(\pi(n))$.

Thus, within a block of equal values, π can be an arbitrary permutation and the blocks need to be ordered according to x.

Given a permutation π that is consistent with x, we can construct a subgradient $g \in \partial \hat{f}(x)$ as follows.

<u>Def</u> (subgradient corresponding to permutation π).

Let $P_0 = \emptyset, P_1, \ldots, P_n$ be the prefix sets of π:

$$P_i = \{\pi(1), \pi(2), \ldots, \pi(i)\}$$

Let $g \in \mathbb{R}^V$ be the vector such that

$$g(\pi(i)) = f(P_i) - f(P_{i-1}) \quad \text{for all } 1 \leq i \leq n.$$

<u>Lemma</u>: If π is consistent with x, the vector g as defined above is a subgradient of $\hat{f}$ at x.

We will prove this lemma later.

This gives us a very simple algorithm for finding a subgradient.

Subgradient algorithm

Given x, pick a permutation π consistent with x (break ties arbitrarily) and return the subgradient corresponding to π.

The running time of the subgradient algorithm is $\boxed{O(n \cdot EO + n \log n)}$:
it takes $O(n \log n)$ time to sort x
and $n \cdot EO$ time to evaluate all prefix sets P_i and get $f(P_i)$.

Thus it suffices to bound the ℓ_2^2-norm of the subgradients. We can show the following upper bound:

Lemma: Let $g \in \partial \hat{f}(x)$ be a subgradient as above. Let M be the maximum value of f, i.e., $|f(S)| \leq M$ for all $S \subseteq V$.
Then $\|g\|_1 \leq 3M$. Since $\|g\|_2 \leq \|g\|_1$,
we have $\|g\|_2^2 \leq 9M^2$ as well.

Proof: To simplify notation, for the purposes of this proof, let us relabel V such that π is the identity, i.e., $x(1) \geq x(2) \geq \cdots \geq x(n)$
Thus, for each $i \leq n$, we have $g(i) = f(\{1,\dots,i\}) - f(\{1,\dots,i-1\})$
Let $r_1 \leq r_2 \leq \cdots \leq r_R$ denote all coordinates such that $g(r_i) > 0$
and let $s_1 \leq s_2 \leq \cdots \leq s_S$ denote all coordinates such that $g(s_i) < 0$.
We bound the contribution of the positive and negative coordinates separately.

Let us consider the contribution of the positive coordinates to $\|g\|_1$.

Let $R_i = \{r_1, \dots, r_i\}$ for all $i \leq R$ and $R_0 = \emptyset$.

Recall that we assume that $f(\emptyset) = 0$ and thus $f(R_0) = 0$.

By submodularity, for each $i \leq R$, we have

$$f(R_i) - f(R_{i-1}) \geq f(\{1, \dots, r_i\}) - f(\{1, \dots, r_{i-1}\}) = g_{r_i} = |g_{r_i}|$$

Thus

$$f(R_R) - f(R_0) = \sum_{i \in [R]} (f(R_i) - f(R_{i-1})) \geq \sum_{i \in [R]} |g(r_i)|$$

Since $f(R_0) = 0$ and $f(R_R) \leq M$, we get $\sum_{i \in [R]} |g(r_i)| \leq M$, and thus the contribution of the positive coordinates to $\|g\|_1$ is at most M.

Let us now consider the negative coordinates of g. We use a similar argument as above. Let $S_i = \{s_1, \dots, s_i\}$ for all $i \leq S$ with $S_0 = \emptyset$

Let $\bar{S} = [n] \setminus S$. Note that, for all $i \in [S]$, the set $\bar{S} \cup S_{i-1}$ is a superset of $\{1, \dots, s_i - 1\}$. Therefore, by submodularity, for all $i \in [S]$

$$f(\bar{S} \cup S_i) - f(\bar{S} \cup S_{i-1}) \leq f(\{1, \dots, s_i\}) - f(\{1, \dots, s_{i-1}\})$$
$$= g(s_i) = -|g(s_i)|$$

Summing over all i, we get $f([n]) - f(\bar{S}) \leq \sum_{i \in [S]} -|g(s_i)|$

Since $f([n]) \geq -M$ and $f(\bar{S}) \leq M$, we have $\sum_{i \in [S]} |g(s_i)| \leq 2M$.

Thus $\|g\|_1 \leq 3M$, as desired.

$\blacksquare$

Thus, by putting everything together, we get a projected subgradient descent for $\min_{x \in [0,1]^V} \hat{f}(x)$ whose running time is as follows:

- the running time per iteration is $O(n \log n + n \cdot EO)$:
  - the time to find the subgradient is $O(n \log n + n\, EO)$
  - the time to implement the update and projection steps is $O(n)$

- the total number of iterations to reach an ε-optimal point is

$$T = O\left(\frac{R^2 B^2}{\varepsilon^2}\right) \quad \text{where} \quad R^2 = \Theta(n), \; B^2 = O(M^2)$$

and thus $T = O\left(\frac{n M^2}{\varepsilon^2}\right)$

- the total running time is therefore $O\left(\frac{n \cdot M^2}{\varepsilon^2}(n \log n + n \cdot EO)\right)$

$$= O\left(\frac{n^2 M^2 \log n}{\varepsilon^2} + \frac{n^2 M^2}{\varepsilon^2} \cdot EO\right)$$

<u>Remark</u>: Just like any gradient descent scheme, the algorithm does not return an exact solution, only an ε-optimal point.

For integer valued functions (that is, $f(S)$ is an integer for all $S \subseteq V$) an ε-optimal point will be the exact minimizer provided $\varepsilon < 1$.

It is very natural to assume that f is integer, or at least rational, since we can only represent rational values on a computer.

By scaling by the largest common denominators of $\{f(S) : S \subseteq V\}$, we obtain an integral function; or equivalently, to get an exact solution, we set $\varepsilon < 1/(\text{largest common denominator})$.