1 Overview

The goal of today’s lecture is to see how the multilinear extension of a submodular function that we introduced in the previous lecture can be used to solve a very general class of submodular optimization problems. In particular, we will introduce the Continuous Greedy Algorithm which is a general algorithm to optimize the multilinear extension of a submodular function over polytopes.

2 Multilinear Extension

In this section $N$ will denote a finite set with $n$ elements, $N = \{1, \ldots, n\}$ and $f$ will be a set function defined over the power set of $N$, $f : 2^N \to \mathbb{R}$.

**Definition 1.** The multilinear extension of $f$ is the function $F : [0, 1]^n \to \mathbb{R}$ defined by:

$$F(x) = \sum_{S \subseteq N} f(S) \prod_{i \in S} x_i \prod_{i \in N \setminus S} (1 - x_i)$$

**Remark 2.** There is a probabilistic interpretation of the multilinear extension. Given $x \in [0, 1]^n$ we can define $X$ to be the random subset of $N$ in which each element $i \in N$ is included independently with probability $x_i$ and not included with probability $1 - x_i$. We write $X \sim x$ to say that $X$ is the random subset sampled according to $x$. Then the multilinear extension $F$ is simply:

$$F(x) = \mathbb{E}_{X \sim x} [f(X)]$$

For this reason, using the multilinear extension is often called relaxing through expectation.

It is possible to relate properties of $f$ to properties of its multilinear extension $F$. In particular, we have:

**Proposition 3.** Let $F$ be the multilinear extension of $f$, then:

1. If $f$ is non-decreasing, then $F$ is non-decreasing along any direction $d \geq 0$.
2. If $f$ is submodular then $F$ is concave along any line $d \geq 0$.

**Proof.** Both properties can be established by first looking at how $F$ behaves along coordinates axes.
1. Let $i \in N$, since $F$ is linear in $x_i$, we have:

$$\frac{\partial F}{\partial x_i}(x) = F(x_1, \ldots, x_{i-1}, 1, x_{i+1}, \ldots, x_n) - F(x_1, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_n)$$

Let $R$ be the random subset of $N \setminus \{i\}$ where each element $j \in N \setminus \{i\}$ is included with probability $x_j$, then we can rewrite:

$$\frac{\partial F}{\partial x_i}(x) = \mathbb{E}\left[f(R \cup \{i\})\right] - \mathbb{E}\left[f(R)\right].$$

Since $f$ is non-decreasing we get that $\frac{\partial F}{\partial x_i}(x) \geq 0$.

2. Similarly, if we denote by $R$ the random subset of $N \setminus \{i,j\}$ where each element $k$ is included with probability $x_k$, we have:

$$\frac{\partial^2 F}{\partial x_i \partial x_j}(x) = \mathbb{E}\left[f(R \cup \{i, j\})\right] - \mathbb{E}\left[f(R \cup \{i\})\right] - \mathbb{E}\left[f(R \cup \{j\})\right] + \mathbb{E}\left[f(R)\right]$$

by reordering the terms we obtain:

$$\frac{\partial^2 F}{\partial x_i \partial x_j}(x) = \mathbb{E}\left[f(R \cup \{i, j\}) - f(R \cup \{i\})\right] - \left(\mathbb{E}\left[f(R \cup \{j\}) - f(R)\right]\right).$$

By submodularity of $f$ this last quantity is non-positive, i.e. $\frac{\partial^2 F}{\partial x_i \partial x_j}(x) \leq 0$.

We conclude the proof of the proposition as follows. Let $x \in [0,1]^n$ and $d \geq 0$. We define the function $F_{x,d}(\lambda) = F(x + \lambda d)$ of the real variable $\lambda$. We note that $F'_{x,d}(\lambda) = \langle d, \nabla F(x + \lambda d) \rangle$ and $F''_{x,d} = d^T H_f(x + \lambda d)d$.

1. If $f$ is non-decreasing, then $\nabla F(x + \lambda d) \geq 0$ and $\langle d, \nabla F(x + \lambda d) \rangle \geq 0$. Hence $F_{x,d}$ is non-decreasing.

2. If $f$ is submodular, then $H_f(x + \lambda d) \leq 0$ and $d^T H_f(x + \lambda d)d \leq 0$. Hence $F_{x,d}$ is concave. \qed

### 3 Submodular Welfare Problem

In the submodular welfare problem we have:

- a set $N = \{1, \ldots, n\}$ of $n$ items,
- a set $M = \{1, \ldots, m\}$ of $m$ agents,
- each agent $i \in M$ has a valuation function $v_i : 2^N \to \mathbb{R}^+$ over subsets of items. Valuation functions are assumed to be monotone and submodular.

A partition of the items is a $m$-tuple $(S_1, \ldots, S_m)$ such that $S_i \subseteq N$ for all $i \in M$ and $S_i \cap S_j = \emptyset$ for all pairs $(i, j)$ in $M^2$.

The value of a partition $S = (S_1, \ldots, S_m)$ is simply $v(S) = \sum_{i=1}^{m} v_i(S_i)$. The submodular welfare problem is to find a partition of maximum value.
3.1 Reformulation of the Submodular Welfare Problem

A more amenable way to write partition of items is to write them as subsets of $M \times N$: if $S \subseteq M \times N$ and if $(i, j) \in S$ then it means that we allocate item $j$ to agent $i$. The fact that $S$ has to be a partition of the items simply means that we cannot allocate the same item to more than one agent, in other terms:

$$\forall j \in N, \ |\{i \mid (i, j) \in S\}| \leq 1$$

(1)

We will denote by $I$ the set of all subsets $S$ of $M \times N$ satisfying the property (1). The value of a partition $S$ can then be written:

$$v(S) = \sum_{i \in M} v_i(\{j \mid (i, j) \in S\})$$

and the submodular welfare problem is simply:

$$\max_{S \in I} v(S)$$

(2)

3.2 Relaxation of the Submodular Welfare Problem

We now want to write a continuous relaxation of the problem (2). We can introduce a decision variable $x_{ij} \in [0, 1]$ for all $(i, j) \in M \times N$ expressing that we allocate the fraction $x_{ij}$ of item $j$ to agent $i$. The partition constraint now expresses that we cannot allocate more than 100% of the same object, i.e:

$$\sum_{i \in M} x_{ij} \leq 1, \ j \in N$$

we will denote by:

$$P = \{x \in [0, 1]^{m \times n} \mid \forall j \in N, \sum_{i \in M} x_{ij} \leq 1\}$$

the feasible domain of the relaxed problem.

To relax the value function $v$, one can simply use its multilinear extension $F$. Using the linearity of the expectation, we can write:

$$F(x) = \mathbb{E}_{X \sim x} [v(X)] = \sum_{i \in M} \mathbb{E}_{X \sim x} [v_i(\{j \mid (i, j) \in X\})] = \sum_{i \in M} \mathbb{E} [v_i(X_i)]$$

where $X_i$ is a random subset of $N$ such that item $j$ is included with probability $x_{ij}$ and excluded with probability $1 - x_{ij}$.

Finally our relaxation of problem (2) is:

$$\max_{x} F(x)$$

s.t. $x \in P$
Algorithm 1 Continuous Greedy Algorithm

Require: $F, P$
1: define: $v_{\max}(x) = \text{argmax}_{v \in P} \langle v, \nabla F(x) \rangle$
2: $x(0) \leftarrow 0 \in \mathbb{R}^n$
3: for $t \in [0, 1]$ do
4: $x'(t) = v_{\max}(x(t))$
5: end for
6: return $x(1)$

4 Continuous Greedy Algorithm

We see that problem (3) consists in maximizing the multilinear extension of $v$ over a polyhedron. More generally, many submodular maximization problems have a relaxation of this form and the Continuous Greedy Algorithm (Algorithm 1 was specifically designed for these relaxations.

We note that Algorithm 1 is not readily implementable. In particular, line 4 requires solving a differential equation. In practice we will only be able to solve it approximately. For now, we will study this abstract algorithm and come back to practical considerations in Section 4.2.

4.1 Analysis of Algorithm 1

In this section we assume that $F$ is the multilinear extension of a non-decreasing submodular function $f$. Our goal is to prove that $x(1)$ returned by Algorithm 1 is an approximate solution to problem 3. First we need the following lemma:

Lemma 4. For any $x \in \mathbb{R}^n$, there exists $v \in P$ such that $\langle v, \nabla F(x) \rangle \geq \text{OPT} - F(x)$, where OPT denotes the optimal solution to problem 3.

Proof. Let us take $v \in P$ such that $F(v) = \text{OPT}$. We want to show that $\langle v, \nabla F(x) \rangle \geq F(v) - F(x)$. We note that if $F$ was concave, this would follow from the characterization of concavity in terms of tangent lines. From Proposition 3, we know that $F$ is only concave along directions $d \geq 0$ and we need to cheat a bit.

Let us consider the direction $d = (v - x) \vee 0$, where $x \vee y$ denotes the coordinate-wise min of $x$ and $y$: $(x \vee y)_i := \min(x_i, y_i)$. Now $d \geq 0$ and $F$ is concave along direction $d$, hence:

$$\langle d, \nabla F(x) \rangle \geq F(x + d) - F(x) \tag{4}$$

But we note that:

1. $x + d = v \vee x \geq v$ and since $F$ is non-decreasing along positive directions, $F(x + d) \geq F(v)$.
2. $d \leq v$ and $\nabla F(x) \geq 0$ hence $\langle d, \nabla F(x) \rangle \leq \langle v, \nabla F(x) \rangle$.

Combining the two points above with (4), we obtain:

$$\langle v, \nabla F(x) \rangle \geq F(v) - F(x)$$

which concludes the proof of the lemma. □
We can now state the main result.

**Theorem 5.** When $F$ is the multilinear extension of a non-decreasing submodular function $f$, $x(1)$ computed by Algorithm 1 is such that:

1. $x(1) \in P$.
2. $F(x(1)) \geq (1 - \frac{1}{e}) \OPT$.

**Proof.**

1. Using the fundamental theorem of calculus:

$$x(1) = \int_0^1 x'(t) dt = \int_0^1 v_{\text{max}}(x(t)) dt$$

where the second equality uses the fact that $x(t)$ is the solution of the differential equation given in line 4 of Algorithm 1. Now we can write the integral as the limit of its Riemann sum:

$$x(1) = \lim_{n \to \infty} \frac{1}{n} \sum_{i=1}^n v_{\text{max}} \left( \frac{i}{n} \right)$$

By definition $v_{\text{max}}(x) \in P$ for any $x$ and the term inside the limit is a convex combination of finitely many elements of $P$, since $P$ is convex, it belongs to $P$. $P$ being closed, the limit $x(1)$ belongs to $P$.

2. Using the chain rule, we can write:

$$\frac{d}{dt} F(x(t)) = \langle x'(t), \nabla F(x(t)) \rangle = \langle v_{\text{max}}(x(t)), \nabla F(x(t)) \rangle$$

Using lemma 4, we know that there exist $v \in P$ such that $\langle v, \nabla F(x(t)) \rangle \geq \OPT - F(x(t))$. In particular, this is true for $v_{\text{max}}(x(t))$ and we get:

$$\frac{d}{dt} F(x(t)) \geq \OPT - F(x(t))$$

Let us define $g : [0,1] \to \mathbb{R}$ by $g(t) = F(x(t))$. We have:

$$g'(t) + g(t) \geq \OPT \quad \text{and} \quad g(0) = 0$$

Defining $h(t) = g'(t) + g(t)$ and solving this differential equation about $g$:

$$g(t) = \int_0^t e^{x-t} h(x) dx$$

But by definition $h(x) \geq \OPT$, hence:

$$F(x(1)) = g(1) \geq \OPT \int_0^1 e^{x-1} dx = \OPT [e^{x-1}]_0^1 = \OPT \left( 1 - \frac{1}{e} \right)$$

which concludes the proof of the theorem. \qed
4.2 Practical Implementation

There are a few points to address before we can actually implement Algorithm 1.

1. **Computing** $F(x)$ and $\nabla F(x)$. Note that the definition of $F$ involves summing over all subsets $S$ of $N$. There are $2^n$ such subsets, hence even computing $F(x)$ for a single $x$ could take exponential time...

   Fortunately, using a Chernoff bound, we can obtain:

   \[
   \left| \frac{1}{t} \sum_{i=1}^{t} f(X_i) - F(x) \right| \leq \varepsilon f(N)
   \]

   with probability at least $1 - e^{-t\varepsilon^2/4}$, where $X_1, \ldots, X_t$ are random subsets of $N$ sampled according to $x$. What that means is that using $O\left(\frac{1}{\varepsilon^2}\right)$ random samples, we can compute a $\varepsilon$-approximation of $F(x)$ with constant probability.

   Similarly, we saw in Proposition 3 that $\frac{\partial F}{\partial x_i}(x) = \mathbb{E}[f(R \cup \{i\})] - \mathbb{E}[f(R)]$ where $R$ is a random subset of $N \setminus \{i\}$ sampled according to $x$. Again, using $O\left(\frac{1}{\varepsilon^2}\right)$ samples, we can obtain a $\varepsilon$-approximation of $\nabla F(x)$ with constant probability.

2. **Computing** $v_{\max}(x)$. By definition, $v_{\max}(x) = \arg\max_{v \in \mathcal{P}} \langle v, \nabla F(x) \rangle$. But observe that once we have computed $\nabla F(x)$, this is simply a linear program in $v$. We learned how to solve them in the first part of this course!

3. **Solving** $x'(t) = v_{\max}(x(t))$. The differential equation can be solved approximately by discretizing time. There are entire books written on this topic, but a simple approach is the following algorithm:

   **Algorithm 2** Solving the differential equation

   **Require:** Solver to compute $v_{\max}$
   
   1: $\delta \leftarrow \frac{1}{n}, x \leftarrow 0$
   2: for $k = 1$ to $n$ do
   3: \hspace{1em} $v \leftarrow v_{\max}(x)$
   4: \hspace{1em} $\hat{a}x \leftarrow x + \delta v$
   5: end for
   6: return $x$

   It is possible to show that the $x$ returned by Algorithm 2 is arbitrarily close to $x(1)$ returned by Algorithm 1 as $n$ goes to infinity.

   It remains to show that the different approximations that we introduced in these practical considerations can be combined to obtain an efficient (polynomial-time) algorithm which computes an arbitrarily good approximation of what is computed by the abstract Algorithm 1. For details about this see [2].
5 Discussion and Further Reading

Since 1978 it was known that there is a $\frac{1}{2}$ approximation algorithm for the submodular welfare problem (this is the algorithm that we discussed and analyzed in Lecture 19). Until 2008 it was unknown whether $\frac{1}{2}$ is the best approximation ratio achievable for this problem. The continuous greedy algorithm we discussed here was developed and analyzed by Vondrak [2], and settled this open question by giving a $1 - \frac{1}{e}$ which is optimal unless $P \neq NP$. Interestingly, in 2012 Filmus and Ward showed that the $1 - \frac{1}{e}$ approximation ratio is also achievable via a local-search algorithm [1].

References
