6.883 Learning with Combinatorial Structure
Note for Lecture 10
Instructor: Prof. Stefanie Jegelka
Scribe: Gal Shulkind

1 A quick recap

In the previous lecture we considered problems of the form

$$S^\alpha \in \arg\min_S F(S) + \alpha|S| \tag{1}$$

where $F(S)$ is submodular and proved that the solutions satisfy monotonicity, i.e. for $\alpha < \beta$ we have $S^\beta \subseteq S^\alpha$.
Subsequently we studied the equivalence of the following set of problems:

$$\min_{y \in \mathcal{B}_F} \frac{1}{2}\|y\|^2 \qquad \longleftrightarrow \qquad \min_x f(x) + \frac{1}{2}\|x\|^2 \qquad \longleftrightarrow \qquad \min_S F(S) + \alpha|S| \tag{2}$$

the solution of each of which can be extracted from the solution(s) of any of the others.
We introduced a minimum norm active set algorithm for the solution of the first problem where at each iteration step the interim solution is expressed as a combination of an active set of extreme points of the base polytop $\mathcal{B}_F$. It is guaranteed to stop in finite time, and a recent analysis (Chakrabarty et al, NIPS 2014) shows a convergence rate of $O(1/t)$.

2 Applications

Next we survey some applications where solutions for the parametric optimization problem $\arg\min_S F(S) + \alpha|S|$ may be of intereset.

Example 2.1. Ratio problems.

$$\min_{S \subseteq \mathcal{V}} \frac{F(S)}{|S|} \tag{3}$$

An example application that adheres to this form is the following. Let $\mathcal{G} = (\mathcal{V}, \epsilon)$ be a graph. To find the most densely connected subgraph $S \subseteq \mathcal{V}$ define $\hat{F}(S)$ to count the number of edges in $\mathcal{G}$ that connect solely to vertices in S. The most densely connected

subgraph is then given according to $\max_{S\subseteq\mathcal{V}} \frac{\hat{F}(S)}{|S|}$. As we have previously seen (hw1, problem 5.1) $\hat{F}(S)$ is supermodular and so defining the submodular function $F(S) = -\tilde{F}(S)$ we write $\max_{S\subseteq\mathcal{V}} \frac{\hat{F}(S)}{|S|} = -\min_{S\subseteq\mathcal{V}} \frac{F(S)}{|S|}$ which adheres to (3).
To solve (3) we find the largest $\lambda^\star$ satisfying

$$\forall S: \qquad \frac{F(S)}{|S|} \geq \lambda^\star \iff F(S) - \lambda^\star |S| \geq 0 \tag{4}$$

We thus have:

$$\begin{aligned} \lambda > \lambda^\star: &\qquad \min_S \left(F(S) - \lambda|S|\right) < 0 \\ \lambda \leq \lambda^\star: &\qquad \min_S \left(F(S) - \lambda|S|\right) = 0 \end{aligned} \tag{5}$$

so we can apply the minimum norm algorithm while sweeping the parameter λ to search for the threshold $\lambda^\star$ satisfying the criterion in (5).

Example 2.2. Image denoising.

$$\min_x \|x - y\|^2 + \lambda \sum_{(i,j)\in\epsilon} |x_i - x_j| \tag{6}$$

y is a noisy image with e.g. salt and pepper noise and we want to reconstruct a clean version x. To obtain a smooth x the second term in (6) penalizes variation between adjacent pixels $|x_i - x_j|$.
The penalizing term can be shown to be the Lovász extension of a graph cut function, which is submodular. Thus (6) takes a form similar to the middle problem in (2), and with some adjustments is amenable to solution via the minimum norm algorithm (hw 2, problem 4).

Example 2.3. Approximate inference in graphical models.
Start with a distribution function defined according to $P(S) = \frac{1}{Z}\exp(-F(S))$.
If $F(S)$ is submodular we can find the mode of the distribution efficiently through $\hat{S} = \arg\min_S F(S)$. However, extracting other features of the distribution, e.g. the partition function, may be computationally much harder. Next we consider an efficient approximation scheme for the partition function.
First notice:

$$\log Z \equiv \log \sum_{A\subseteq\mathcal{V}} e^{-F(A)} \leq \log \sum_{A\subseteq\mathcal{V}} e^{-G(A)} \tag{7}$$

holds whenever $G(A)$ satisfies $\forall A: \quad G(A) \leq F(A), G(\mathcal{V}) = F(\mathcal{V})$. This is akin to upper bounding the partition function associated with $F(\cdot)$ with the partition function

associated with a suitable $G(\cdot)$.
Specifically, if we restrict $G(\cdot)$ to take modular form $G(S) = \sum_{i \in S} s_i$ we have $\log \sum_{A \subseteq \mathcal{V}} e^{-G(A)} = \sum_{i \in \mathcal{V}} \log(1 + e^{-s_i})$ such that:

$$\log Z \leq \sum_{i \in \mathcal{V}} \log(1 + e^{-s_i}) \tag{8}$$

To obtain a good approximation we want the bound as tight as possible so we minimize the RHS of (8) over all all suitable choices for $G(S)$, which is easy in lieu of the additive form and it being a convex function of the variables. It can be shown that achieving this is possible through application of the minimum norm algorithm as described in [Djolonga & Krause, 2014].

3 Composite functions

In this section we study problems taking the form

$$\min_x \sum_{i=1}^{k} f_i(x) \tag{9}$$

where optimization over the sum is computationally hard, however optimizing over the individual components $f_i(x)$ ia a more managable task.

3.1 Applications

We begin by introducing several examples where we encounter formulations like (9).

Example 3.1. Graphical models.
Let $P(x) = \prod_C \Psi_C(x)$ where C are cliques of few variables (i.e. the fuctions $\Psi_C(x)$ are simple), and we want to find the mode, i.e.

$$\arg\max_x P(x) = \arg\min_x -\log P(x) = \arg\min_x \sum_C f_c(x) \tag{10}$$

where $f_c(x) \equiv -\log \Psi_C(x)$.
Alternatively, we could have a large graph that can be decomposed into multiple simpler graphs that are easy to work with. We will see an example for this in the sequel with a decomposition of a complex grid graph into simple tree graph components (section 3.3).

Example 3.2. Sums of submodular functions.
Sometimes a function of interest can be decomposed as a sum of simple submodular functions. For example, as we have seen before, a graph cut function can be written as a sum of simple cut functions on single edges. We can further generalize this by introducing hypergraph cut functions, which are defined as follows. Let C be a hyperedge (list of vertices), then define the corresponding cut function:

$$F_C(S) \equiv \begin{cases} 0 & S \cap C = \emptyset, C \\ 1 & \text{otherwise} \end{cases} \tag{11}$$

$F_C(S)$ penalizes subsets S that split C, and is easily recognized to be submodular.

Example 3.3. Total variation functions.

3.2 Decomposition

Next we develop algorithms to solve problem (9). Here we assume that the functions $f_i(x)$ are convex and finite.
The basic idea is to introduce new variables for each of the functions and a constraint that forces them to agree:

$$\min_x \sum_{i=1}^k f_i(x) = \begin{array}{ll} \min\limits_{x_0, x_1, \ldots, x_k} & \sum_{i=1}^k f_i(x_i) \\ \text{s.t.} & x_i = x_0,\ i = 1, \ldots, k \end{array} \tag{12}$$

Next, introduce the Lagrangian:

$$\mathcal{L}(x_0, \{x_i\}, \{\lambda_i\}) = \sum_{i=1}^k f_i(x_i) + \lambda_i(x_i - x_0) = \sum_{i=1}^k (f_i(x_i) + \lambda_i x_i) - \left(\sum_{i=1}^k \lambda_i \right) x_0 \tag{13}$$

where $\lambda = [\lambda_1, \ldots, \lambda_k]^T$ are the dual variables. The Lagrange dual function is given according to:

$$g(\lambda) = \min_{x_0, \{x_i\}} \mathcal{L}(x_0, \{x_i\}, \{\lambda_i\}) = \begin{cases} -\infty & \sum\limits_{i=1}^k \lambda_i \neq 0 \\ \min\limits_{x_1, \ldots, x_k} \sum\limits_i f_i(x_i) + \lambda_i x_i & \sum\limits_{i=1}^k \lambda_i = 0 \end{cases} \tag{14}$$

and the dual problem is:

$$\max_\lambda g(\lambda) = \begin{array}{ll} \max\limits_{\lambda} & \tilde{g}(\lambda) \\ \text{s.t.} & \lambda \in \Lambda \end{array} \tag{15}$$

where we have defined $\tilde{g}(\lambda) \equiv \min_{x_1,\ldots,x_k} \sum_i f_i(x_i) + \lambda_i x_i$ and $\Lambda \equiv \left\{\lambda | \sum_i \lambda_i = 0\right\}$.
Notice that (15) is a constrained optimization problem over a convex set where $\tilde{g}(\lambda)$ is convex as a minimum over linear functions of λ.
We iteratively solve by means of the constrained subgradient method:

$$\lambda^{t+1} = \Pi_\Lambda(\lambda^t + \alpha_t \nabla \tilde{g}(\lambda^t)) \tag{16}$$

where t is the iteration index, α_t the step size coefficient, $\Pi_\Lambda(\cdot)$ the projection operator and $\nabla\tilde{g}(\lambda^t)$ a subgradient. To complete the picture we need to find the exact form of $\Pi_\Lambda(\cdot)$ and a suitable subgradient.

The projection operator is just centering: $\Pi_\Lambda(\mu) = \mu - \frac{1}{k}\sum_{i=1}^{k} \mu_i$.
To find a subgradient notice that we can write $\tilde{g}(\lambda) = \sum_i \min_{x_i}(f_i(x_i) + \lambda_i x_i)$ such that $\tilde{g}(\lambda)$ is a sum of functions of decoupled variables $\min_{x_i}(f_i(x_i) + \lambda_i x_i)$, each of these is a minimum over linear functions of λ and so the subgradient is easily derived here. The ith component of the subgradient is the coefficient of λ_i in a linear function achieving the minimum, i.e.

$$\begin{aligned} &\nabla\tilde{g}(\lambda^t) = [\hat{x}_1, \ldots, \hat{x}_k] \\ &\hat{x}_i \in \arg\min_{x_i} f_i(x_i) + \lambda_i x_i \end{aligned} \tag{17}$$

Notice that (17) entails that to obtain a subgradient we interact with the simple functions $f_i(x)$ individually rather than their sum, which is desirable due to their simpler form, from which we started. Additionally, this computation can be performed efficiently in parallel.

3.3 Example: Decomposition in graphical models

In many applications the distribution function of a graphical model takes the form

$$P(x,\theta) = \exp(-E(x,\theta)) \qquad \Longleftrightarrow \qquad E(x,\theta) = -\log P(x,\theta) \tag{18}$$

where $E(x,\theta)$ is linear in the parameters.
For example, we might be interested in studying a model with pairwise interactions. With $\mathcal{G} = (\mathcal{V}, \epsilon)$ and L the set of possible labels each node can take, define

$$\begin{aligned} &\forall\ v \in \mathcal{V},\ l \in L: && x_v(l) \in \{0,1\} \\ &\forall\ (u,v) \in \epsilon,\ l,l' \in L: && x_{uv}(l,l') \in \{0,1\} \end{aligned} \tag{19}$$

to be indicator vectors specifying the labels the nodes take. $x_v(l) = 1 \Leftrightarrow$ label l is assigned to v and $x_{uv}(l, l') = 1 \Leftrightarrow$ labels (l, l') are assigned to u, v.
We have the following constraints expressing the above descriptions mathematically:

$$\begin{aligned} \sum_{l \in L} x_v(l) &= 1 && \forall v \in \mathcal{V} \\ \sum_{l' \in L} x_{uv}(l, l') &= x_u(l) && \forall (u, v) \in \epsilon \end{aligned} \tag{20}$$

With these definitions in place, we may consider the following linear form for $E(x, \theta)$:

$$\begin{aligned} E(x, \theta) &= \sum_{v \in \mathcal{V}} \sum_{l \in L} \theta_v(l) x_v(l) + \sum_{(u,v) \in \epsilon} \sum_{l, l' \in L} \theta_{uv}(l, l') x_{uv}(l, l') \\ &= \sum_{v \in \mathcal{V}} \theta_v \cdot x_v + \sum_{(u,v) \in \epsilon} \theta_{uv} \cdot x_{uv} \\ &= \theta \cdot x \end{aligned} \tag{21}$$

Graphical models corresponding to descriptions such as the one suggested in (21) may be complicated to solve. An example would be the grid graph depicted in Fig. 1 (top) where each node is connected to four sorrounding neighbors. To compute the mode we need to solve

$$\begin{aligned} \min_{x} \quad & \theta \cdot x \\ \text{s.t.} \quad & x \in \chi \end{aligned} \tag{22}$$

where χ is the marginal polytope representing the set of constraints from Eq. (19), (20). Using the description in Eq. (21) we can decompose the complicated grid graph into two simpler, tree components as schematically depicted in Fig. 1 (bottom). There are multiple ways to decomopse a complicated graph into constituent components. The idea is to decompose to simple graphs amenable to efficient solution, with each edge and node of the original graph appearing in at least one of the components.

We associate a parameter vector $\theta^{\mathcal{T}}$ with each component graph $\mathcal{T}$ and formally require:

$$\sum_{\mathcal{T}} \theta^{\mathcal{T}} = \theta \tag{23}$$

Similarly, define restricted variables on the component graphs $x^{\mathcal{T}} = x_{|\mathcal{T}}$, and correponsing sets of constraints on their restriceted domains $x^{\mathcal{T}} \in \chi^{\mathcal{T}}$. With these in place we have:

$$E(x, \theta) = \sum_{\mathcal{T}} E(x^{\mathcal{T}}, \theta^{\mathcal{T}}) = \sum_{\mathcal{T}} \theta^{\mathcal{T}} \cdot x^{\mathcal{T}} \tag{24}$$

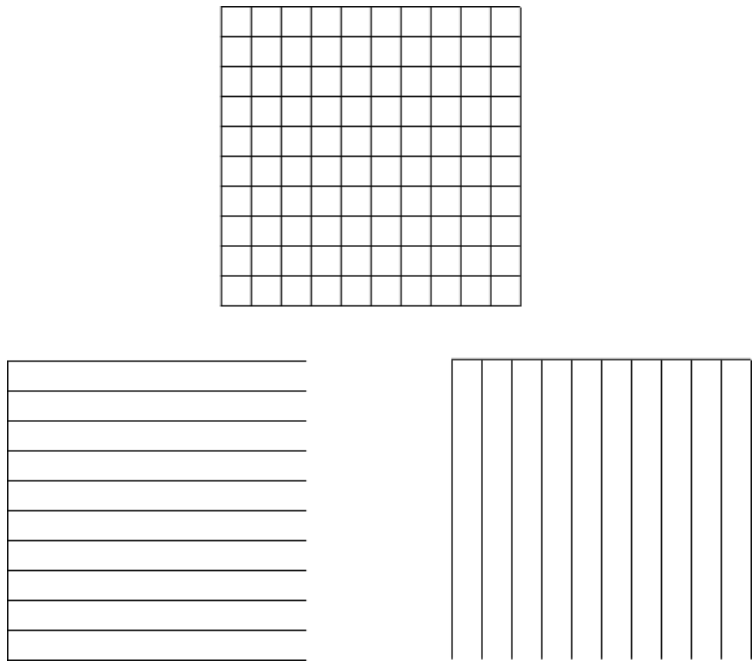

Figure 1: Graphical model decomposition. (top) Original graph $\mathcal{G}$ (bottom) Two constituent tree graphs $\mathcal{T}_1, \mathcal{T}_2$

such that we can port Eq. (22) to the form:

$$\begin{array}{ll} \min\limits_{\{x^{\mathcal{T}}\},x} & \sum\limits_{\mathcal{T}} \theta^{\mathcal{T}} \cdot x^{\mathcal{T}} \\ \text{s.t.} & x^{\mathcal{T}} = x_{|\mathcal{T}} \qquad \forall \mathcal{T} \\ & x^{\mathcal{T}} \in \chi^{\mathcal{T}} \end{array} \tag{25}$$

Form the Lagrange dual function (where we relax the constraints $x^{\mathcal{T}} \in \chi^{\mathcal{T}}$):

$$g(\lambda) = \min_{\{x^{\mathcal{T}}\},x} \sum_{\mathcal{T}} \theta^{\mathcal{T}} \cdot x^{\mathcal{T}} + \sum_{\mathcal{T}} \lambda^{\mathcal{T}} (x^{\mathcal{T}} - x_{|\mathcal{T}}) \tag{26}$$

The form of the Lagrange dual function here is similar to the one we have seen before in Eq. (13), such that the solution technique will be identical. Define:

$$\Lambda \equiv \left\{ \lambda | \sum_{\mathcal{T}} \lambda^{\mathcal{T}} = 0 \right\} \qquad \tilde{g}(\lambda) \equiv \min_{\{x^{\mathcal{T}}\}} \sum_{\mathcal{T}} (\theta^{\mathcal{T}} + \lambda^{\mathcal{T}}) \cdot x^{\mathcal{T}} \tag{27}$$

Solving through the subgradient method we have:

$$\lambda^{t+1} = \Pi_{\Lambda}(\lambda^t + \alpha_t \nabla \tilde{g}(\lambda^t)) \tag{28}$$

As $\tilde{g}(\lambda)$ depends additively on the different constituent graphs $\{\mathcal{T}\}$ computing the subgradient is decoupled between the different graph components as we had following Eq. (13):

$$\begin{array}{l} \nabla \tilde{g}(\lambda^t) = [x^{\mathcal{T}_1}, \ldots, x^{\mathcal{T}_{|\mathcal{T}|}}] \\ x^{\mathcal{T}_i} \in \arg\min\limits_{x^{\mathcal{T}_i}} (\theta^{\mathcal{T}_i} + \lambda^{\mathcal{T}_i}) \cdot x^{\mathcal{T}_i} \end{array} \tag{29}$$

During each iteration the solutions of the constituent graphs $\left\{x^{\mathcal{T}_i}\right\}$ are decoupled as per Eq. (29). However, as the iterations progress, the dual parameters λ introduce coupling by driving the solutions of the different graphs into agreement.
Let's see an example of how this plays out for a case of a single graph with binary variables decomposed into two tree components $\mathcal{T}_1, \mathcal{T}_2$.
Assume for example that following the tth iteration we have the solution for both trees contradict each other on some given node v, e.g. $x_v^{t,\mathcal{T}_1} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, x_v^{t,\mathcal{T}_2} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$. The gradient descent will operate to change the λ's according to (where we only track the vth component of the solution):

$$\begin{pmatrix} \begin{pmatrix} \lambda_v^{t+1,\mathcal{T}_1}(0) \\ \lambda_v^{t+1,\mathcal{T}_1}(1) \end{pmatrix} \\ \begin{pmatrix} \lambda_v^{t+1,\mathcal{T}_2}(0) \\ \lambda_v^{t+1,\mathcal{T}_2}(1) \end{pmatrix} \end{pmatrix} = \Pi_{\Lambda} \begin{pmatrix} \begin{pmatrix} \lambda_v^{t,\mathcal{T}_1}(0) \\ \lambda_v^{t,\mathcal{T}_1}(1) \end{pmatrix} + \alpha \begin{pmatrix} 1 \\ 0 \end{pmatrix} \\ \begin{pmatrix} \lambda_v^{t,\mathcal{T}_2}(0) \\ \lambda_v^{t,\mathcal{T}_2}(1) \end{pmatrix} + \alpha \begin{pmatrix} 0 \\ 1 \end{pmatrix} \end{pmatrix} = \begin{pmatrix} \begin{pmatrix} \lambda_v^{t,\mathcal{T}_1}(0) \\ \lambda_v^{t,\mathcal{T}_1}(1) \end{pmatrix} \\ \begin{pmatrix} \lambda_v^{t,\mathcal{T}_2}(0) \\ \lambda_v^{t,\mathcal{T}_2}(1) \end{pmatrix} \end{pmatrix} + \frac{\alpha}{2} \begin{pmatrix} \begin{pmatrix} +1 \\ -1 \end{pmatrix} \\ \begin{pmatrix} -1 \\ +1 \end{pmatrix} \end{pmatrix} \tag{30}$$

As per (29), in iteration $t+1$ the solution in $\mathcal{T}_1$ will be pushed towards $'1'$ and the solution in $\mathcal{T}_2$ towards $'0'$ such that the coupling introduced by the dual variables drives the solution of the same node in different graphs to agreement.

3.4 Example: Submodular decomposition

Consider the special case of a function decomposable as a sum of submodular functions. Written in the proximal form using the Lovász extension, this reads

$$\min_x \sum_i f_i(x) + \frac{1}{2}\|x\|^2 \tag{31}$$

We manipulate this using the techinques we have developed:

$$\begin{aligned}
\min_x \sum_i f_i(x) + \frac{1}{2}\|x\|^2 = & \min_{x_0,\{x_i\}} \quad \sum_i f_i(x_i) + \tfrac{1}{2k}\|x_i\|^2 \\
& \text{s.t.} \quad\quad x_i = x_0,\ i = 1,\dots,k \\
= & \max_\lambda \min_{x_0,\{x_i\}} \sum_i f_i(x_i) + \frac{1}{2k}\|x_i\|^2 + \lambda_i^T(x_i - x_0) \\
= & \max_\lambda \min_{x_0,\{x_i\}} \sum_i \max_{y_i \in \mathcal{B}_{F_i}} (y_i^T x_i + \frac{1}{2k}\|x_i\|^2 + \lambda_i^T(x_0 - x_i)) \\
= & \max_{\lambda\in\Lambda} \sum_i \min_{\{x_i\}} \max_{y_i \in \mathcal{B}_{F_i}} (y_i^T x_i + \frac{1}{2k}\|x_i\|^2 - \lambda_i^T x_i) \\
= & \max_{\lambda\in\Lambda} \sum_i \max_{y_i \in \mathcal{B}_{F_i}} \min_{\{x_i\}} (y_i^T x_i + \frac{1}{2k}\|x_i\|^2 - \lambda_i^T x_i) \\
= & \max_{\lambda\in\Lambda} \sum_i \max_{y_i \in \mathcal{B}_{F_i}} -\frac{k}{2}\|y_i - \lambda_i\|^2
\end{aligned} \tag{32}$$

where we have used strong duality to swap $\max_{y_i \in \mathcal{B}_{F_i}}$ and $\min_{\{x_i\}}$ and in the last equality found the minimum over $\{x_i\}$ by equating the derivative to zero.
This smooth dual problem is a sum over problems that decompose over y_i, with the λ coupling the expressions as we have seen before.
One way of solving this is to alternate between keeping the λ fixed and optimizing over y_i and then keeping y_i fixed and optimizing over λ, both of which are projection problems. This gives a sequence of alternating projections, which is guranteed to converge to the optimum (at a linear rate).