Optnet for DDPG Constrained Projection

Abhinav Bhatia

October 19, 2018

For ease of implementation, we will consider only recursive greedy projection. i.e. do the allocations top down in reference to the constraints tree.

1 Problem Statement

Given a given node g (which has already been allocated C resources) from the constraints tree, we want to allocate resources to it's k children such that:

$$\sum_{i}^{k} z_i = C$$

$$\forall_i^k : 0 \leq \check{C}_i \leq z_i \leq \hat{C}_i \leq 1$$

We are given C and a vector $\vec{y} \in [0,1]^k$. We want to project this vector to the nearest feasible solution $\vec{z}$.

2 The Linear Program

$$\begin{aligned} \min_{\vec{z}} \sum_{i}^{k} |z_i - y_i| \quad \text{subject to} \\ \sum_{i}^{k} z_i = C \\ \forall_i^k : z_i \leq \hat{C}_i \\ \forall_i^k : \check{C}_i \leq z_i \end{aligned}$$

An equivalent linear program is:

$$\min_{\vec{d},\vec{z}} \sum_{i}^{k} d_i \quad \text{subject to}$$

$$\begin{aligned} \sum_{i}^{k} z_i - C = 0 \qquad & \lambda \\ \forall_i^k : y_i - z_i - d_i \leq 0 \qquad & \mu_i \\ \forall_i^k : z_i - y_i - d_i \leq 0 \qquad & \nu_i \\ \forall_i^k : z_i - \hat{C}_i \leq 0 \qquad & \alpha_i \\ \forall_i^k : \check{C}_i - z_i \leq 0 \qquad & \beta_i \end{aligned} \tag{1}$$

Here $\lambda, \mu_i, \nu_i, \alpha_i, \beta_i$ are the corresponding Lagrange multipliers.

Thus there are $4k + 1$ constraints.

3 The KKT Conditions

The Langragian is:

$$
\begin{aligned}
L(\vec{d}, \vec{z}, \vec{\mu}, \vec{\nu}, \vec{\alpha}, \vec{\beta}, \lambda) = &\sum_i^k d_i \\
&+ \sum_i^k \mu_i(y_i - z_i - d_i) \quad + \sum_i^k \nu_i(z_i - y_i - d_i) \\
&+ \sum_i^k \alpha_i(z_i - \hat{C}_i) \qquad\quad + \sum_i^k \beta_i(\check{C}_i - z_i) \\
&+ \lambda(\sum_i^k z_i - C)
\end{aligned}
\tag{2}
$$

The KKT conditions (conditions satisfied by the solution $\vec{d}^*, \vec{z}^*, \vec{\mu}^*, \vec{\nu}^*, \vec{\alpha}^*, \vec{\beta}^*, \lambda^*$ of the LP 1 is given by

$$
\begin{aligned}
& \nabla_{\vec{d}, \vec{z}, \lambda} L && = \vec{0} \\
\forall_i^k: \quad & \mu_i(y_i - z_i - d_i) && = 0 \\
\forall_i^k: \quad & \nu_i(z_i - y_i - d_i) && = 0 \\
\forall_i^k: \quad & \alpha_i(z_i - \hat{C}_i) && = 0 \\
\forall_i^k: \quad & \beta_i(\check{C}_i - z_i) && = 0
\end{aligned}
$$

which expand to:

$$
\begin{cases}
\qquad \sum_i^k z_i - C & = 0 \\
\forall_i^k: \quad 1 + \mu_i + \nu_i & = 0 \\
\forall_i^k: \quad -\mu_i + \nu_i + \alpha_i - \beta_i + \lambda & = 0 \\
\forall_i^k: \quad \mu_i(y_i - z_i - d_i) & = 0 \\
\forall_i^k: \quad \nu_i(z_i - y_i - d_i) & = 0 \\
\forall_i^k: \quad \alpha_i(z_i - \hat{C}_i) & = 0 \\
\forall_i^k: \quad \beta_i(\check{C}_i - z_i) & = 0
\end{cases}
\tag{3}
$$

4 Differentiating the KKT conditions

We can differentiate both sides of each equation in set of equations 3 w.r.t to inputs $\vec{y}$ and C.

The partial differential equations w.r.t. input y_j are:

$$\begin{cases} & \sum_i^k \frac{\partial z_i}{\partial y_j} & = 0 \quad (a) \\ \forall_i^k : & \frac{\partial \mu_i}{\partial y_j} + \frac{\partial \nu_i}{\partial y_j} & = 0 \quad (b) \\ \forall_i^k : & -\frac{\partial \mu_i}{\partial y_j} + \frac{\partial \nu_i}{\partial y_j} + \frac{\partial \alpha_i}{\partial y_j} - \frac{\partial \beta_i}{\partial y_j} + \frac{\partial \lambda}{\partial y_j} & = 0 \quad (c) \\ \forall_i^k : & \frac{\partial \mu_i}{\partial y_j}(y_i - z_i - d_i) + \mu_i(\delta_{ij} - \frac{\partial z_i}{\partial y_j} - \frac{\partial d_i}{\partial y_j}) & = 0 \quad (d) \\ \forall_i^k : & \frac{\partial \nu_i}{\partial y_j}(-y_i + z_i - d_i) + \nu_i(-\delta_{ij} + \frac{\partial z_i}{\partial y_j} - \frac{\partial d_i}{\partial y_j}) & = 0 \quad (e) \\ \forall_i^k : & \frac{\partial \alpha_i}{\partial y_j}(z_i - \hat{C}_i) + \alpha_i \frac{\partial z_i}{\partial y_j} & = 0 \quad (f) \\ \forall_i^k : & \frac{\partial \beta_i}{\partial y_j}(-z_i + \check{C}_i) - \beta_i \frac{\partial z_i}{\partial y_j} & = 0 \quad (g) \end{cases} \tag{4}$$

Here δ_{ij} is the Kronecker delta function, which is 1 when $i = j$, and 0 otherwise.
Partial differential equations w.r.t C are:

$$\begin{cases} & \sum_i^k \frac{\partial z_i}{\partial C} - 1 & = 0 \\ \forall_i^k : & \frac{\partial \mu_i}{\partial C} + \frac{\partial \nu_i}{\partial C} & = 0 \\ \forall_i^k : & -\frac{\partial \mu_i}{\partial C} + \frac{\partial \nu_i}{\partial C} + \frac{\partial \alpha_i}{\partial C} - \frac{\partial \beta_i}{\partial C} + \frac{\partial \lambda}{\partial C} & = 0 \\ \forall_i^k : & \frac{\partial \mu_i}{\partial C}(y_i - z_i - d_i) + \mu_i(-\frac{\partial z_i}{\partial C} - \frac{\partial d_i}{\partial C}) & = 0 \\ \forall_i^k : & \frac{\partial \nu_i}{\partial C}(-y_i + z_i - d_i) + \nu_i(\frac{\partial z_i}{\partial C} - \frac{\partial d_i}{\partial C}) & = 0 \\ \forall_i^k : & \frac{\partial \alpha_i}{\partial C}(z_i - \hat{C}_i) + \alpha_i \frac{\partial z_i}{\partial C} & = 0 \\ \forall_i^k : & \frac{\partial \beta_i}{\partial C}(-z_i + \check{C}_i) - \beta_i \frac{\partial z_i}{\partial C} & = 0 \end{cases} \tag{5}$$

The equations can be solved independently per input y_j and C.

5 Solving system of equations 4 and 5

For equations 4, the variables are $\frac{\partial}{\partial y_j}$ of $\mu_i, \nu_i, \alpha_i, \beta_i, \lambda, d_i, z_i$. So there are $n = 6k + 1$ variables and that many equations. Trying to write equations 4 in matrix form:

$$A_{n \times n} J^{y_j}_{n \times 1} = B_{n \times 1}$$

where

$$J^{y_j} = \frac{\partial}{\partial y_j}[\lambda, \mu_1, \nu_1, \alpha_1, \beta_1, d_1, z_1, \mu_2, ..., z_2, ..., u_k, ..., z_k]^T$$

and

$$A_{rc} = \begin{cases} 1 & r = 1, c = 6m+1 & m = 1,2,...,k \\ 1 & r = 6m-4, c = r, r+1 & m = 1,2,...,k \\ 1 & r = 6m-3, c = 1, r, r+1 & m = 1,2,...,k \\ -1 & r = 6m-3, c = r-1, r+2 & m = 1,2,...,k \\ y_m - z_m - d_m & r = 6m-2, c = r-2 & m = 1,2...,k \\ -\mu_m & r = 6m-2, c = r+2, r+3 & m = 1,2,...,k \\ -y_m + z_m - d_m & r = 6m-1, c = r-2 & m = 1,2,...,k \\ -\nu_m & r = 6m-1, c = r+1 & m = 1,2,...,k \\ \nu_m & r = 6m-1, c = r+2 & m = 1,2,...,k \\ z_m - \hat{C_m} & r = 6m, c = r-2 & m = 1,2,...,k \\ \alpha_m & r = 6m, c = r+1 & m = 1,2,...,k \\ -z_m + \check{C_m} & r = 6m+1, c = r-2 & m = 1,2,...,k \\ -\beta_m & r = 6m+1, c = r & m = 1,2,...,k \\ 0 & \text{otherwise} \end{cases}$$

and

$$B_r = \begin{cases} -\mu_m \delta_{mj} & r = 6m-2 & m = 1,2,...,k \\ \nu_m \delta_{mj} & r = 6m-1 & m = 1,2,...,k \\ 0 & \text{otherwise} \end{cases}$$

Thus

$$J^{y_j} = A^{-1}B$$

Similary, we can find $J^C = \frac{\partial}{\partial C}[\lambda, \mu_1, \nu_1, \alpha_1, \beta_1, d_1, z_1, \mu_2, ..., z_2, ..., u_k, ..., z_k]^T$ by solving set of equations 5.

Then the overall Jacobian would be:

$$J_{(6k+1)\times(k+1)} = \begin{bmatrix} J^{y_1} & J^{y_2} & ... & J^{y_k} & J^C \end{bmatrix}$$

6 The overall picture

From J, we can extract the rows corresponding to z_i and traspose it and thus write $\nabla_{\vec{y},C}\vec{z}$, which is a $(k+1)\times k$ matrix.

Thus we can get gradient of output $\vec{z}$ w.r.t network parameters $\theta \in \mathbb{R}^p$ using the chain rule as:

$$(\nabla_\theta \vec{z})_{p\times k} = (\nabla_\theta(\vec{y}, c))_{p\times(k+1)}(\nabla_{(\vec{y},C)}\vec{z})_{(k+1)\times k}$$