KKT Nets: A KKT conditions-Informed Neural Network approach to solving convex optimization problems

Rishabh Sharad Pomaje
Department of Electrical Engineering
Indian Institute Of Technology Dharwad,
India.
210020036@iitdh.ac.in

Abstract

Nulla in ipsum. Praesent eros nulla, congue vitae, euismod ut, commodo a, wisi. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Aenean nonummy magna non leo. Sed felis erat, ullamcorper in, dictum non, ultricies ut, lectus. Proin vel arcu a odio lobortis euismod. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Proin ut est. Aliquam odio. Pellentesque massa turpis, cursus eu, euismod nec, tempor congue, nulla. Duis viverra gravida mauris. Cras tincidunt. Curabitur eros ligula, varius ut, pulvinar in, cursus faucibus, augue.

1 Introduction

1.1 Optimization: General form and some definitions

Optimization problems are ubiquitous and appear in almost all disciplines ranging from science, finance, logistics, and economics. Thus, finding a more efficient algorithm to this fundamental task is beneficial. An optimization problem is of the form, Boyd & Vandenberghe (2004),

$$\min_{\boldsymbol{x}\in\mathcal{D}} \quad f_0(\boldsymbol{x}) \tag{1a}$$

$$\text{subject to} \quad f_i(\boldsymbol{x}) \leq 0 \quad , i = 1, 2, \ldots, m. \tag{1b}$$

$$h_i(\boldsymbol{x}) = 0 \quad , i = 1, 2, \ldots, p. \tag{1c}$$

where, $f_0 : \mathbb{R}^n \to \mathbb{R}$ is called the objective function, with $\boldsymbol{x} \in \mathbb{R}^n$ being the objective variable. The functions $f_i : \mathbb{R}^n \to \mathbb{R}$ are called the inequality constraint functions and $h_i : \mathbb{R}^n \to \mathbb{R}$ are the equality constraint functions. $\mathcal{D}$ denotes the domain of the problem and is defined as

$$\mathcal{D} = \bigcap_{i=0}^{m} \mathbf{dom} f_i \ \cap \ \bigcap_{i=1}^{p} \mathbf{dom} g_i. \tag{2}$$

An optimization problem is called a convex program if the following hold:

1. $f_i, i = 0, 2, \ldots, m$ are convex functions of $\boldsymbol{x}$.
2. $h_i, i = 1, 2, \ldots, p$ are affine functions of $\boldsymbol{x}$, i.e., $h_i(\boldsymbol{x}) = \boldsymbol{a}_i^T\boldsymbol{x} + b$.

There are several standard forms of convex program depending on the nature of the objective and the constraint functions.

1.2 A few standard convex programs

1.2.1 Linear Program (LP)

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & \boldsymbol{c}^T\boldsymbol{x} + d && \text{(3a)} \\ \text{subject to} \quad & \boldsymbol{G}\boldsymbol{x} \preceq \boldsymbol{h} && \text{(3b)} \\ & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} && \text{(3c)} \end{aligned}$$

where, $\boldsymbol{G} \in \mathbb{R}^{m\times n}$ and $\boldsymbol{A} \in \mathbb{R}^{p\times n}$. Several practical examples of LPs can be found in,Boyd & Vandenberghe (2004).

1.2.2 Quadratic Program

$$\begin{aligned} \min_{\boldsymbol{x}\in\mathbb{R}^n} \quad & \frac{1}{2}\boldsymbol{x}^T\boldsymbol{P}\boldsymbol{x} + \boldsymbol{q}^T\boldsymbol{x} + r && \text{(4a)} \\ \text{subject to} \quad & \boldsymbol{G}\boldsymbol{x} \preceq \boldsymbol{h} && \text{(4b)} \\ & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} && \text{(4c)} \end{aligned}$$

1.2.3 Other standard problems

There are other standard problems that categorize themselves under convex optimization such as Quadratic Programming Quadratic Constraints (QCQP), Second Order Cone Programming (SOCP), etc. We consider them as future work.

1.3 Dual Problems: Lagrange Dual Problems and few more definitions

Let $\lambda \in \mathbb{R}^m$ and $\nu \in \mathbb{R}^p$. The *Lagrangian* $\mathcal{L} : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^p \to \mathbb{R}$ is defined as,

$$\mathcal{L}(\boldsymbol{x}, \lambda, \nu) = f_0(\boldsymbol{x}) + \sum_{i=1}^{m} \lambda_i f_i(\boldsymbol{x}) + \sum_{i=1}^{p} \nu_i g_i(\boldsymbol{x}). \tag{5}$$

The *Lagrange Dual function* (LDF) is given as,

$$g(\lambda, \nu) = \inf_{\boldsymbol{x}\in\mathcal{D}} \mathcal{L}(\boldsymbol{x}, \lambda, \nu) \tag{6}$$

Note that by definition the LDF is a pointwise infinimum of affine functions of λ and ν. Thus, the LDF is always a convex function regardless of the nature of the objective and constraint functions.

Suppose λ_i are the dual variables associated with the inequality constraints $f_i(x)$ for $i \in \{1, 2, \ldots, m\}$, and ν_i are the dual variables associated with the equality constraints $g_i(x)$ for $i \in \{1, 2, \ldots, p\}$. Under some regularity conditions, the following Karush-Kuhn-Tucker (KKT) conditions provide necessary and sufficient conditions for optimality in such problems:

$$\text{Primal feasibility: } f_i(x^*) \leq 0, \quad g_i(x^*) = 0, \tag{7}$$

$$\text{Dual feasibility: } \lambda_i^* \geq 0, \tag{8}$$

$$\text{Complementary slackness: } \lambda_i^* f_i(x^*) = 0, \tag{9}$$

$$\text{Stationarity: } \nabla f_0(x^*) + \sum_{i=1}^{M} \lambda_i^* \nabla f_i(x^*) + \sum_{i=1}^{K} \nu_i^* \nabla g_i(x^*) = 0, \tag{10}$$

In this article, we focus on training a deep learning model to take problem parameters for specific subclasses of convex optimization problems, such as linear or quadratic programs with up to a maximum n, and output the optimal primal variable x^* and dual variables $\{\lambda_i^*\}_{i=1}^{M}$ and $\{\nu_i^*\}_{i=1}^{K}$. We propose an architecture that embeds the KKT conditions for optimality into the neural network and defines loss functions accordingly. We refer to these networks as KKT Networks.

2 Neural Network Approach to Solving KKT Conditions

We propose using a neural network to solve the KKT conditions by learning the optimal primal and dual variables. The neural network takes problem parameters as inputs and outputs the optimal primal variables x^* and dual variables λ^*. The loss function is designed to capture the KKT conditions:

$$\text{Loss} = \text{Primal Feasibility Loss} + \text{Complementary Slackness Loss} + \text{Stationarity Loss} \tag{11}$$

where:

$$\text{Primal Feasibility Loss} = \frac{1}{N}\sum_{i=1}^{M} \max(0, f_i(x))^2 \tag{12}$$

$$\text{Complementary Slackness Loss} = \frac{1}{N}\sum_{i=1}^{M} (\lambda_i f_i(x))^2 \tag{13}$$

$$\text{Stationarity Loss} = \frac{1}{N}\left\| \nabla f_0(x) + \sum_{i=1}^{M} \lambda_i \nabla f_i(x) + \sum_{i=1}^{K} \nu_i \nabla g_i(x) \right\|^2 \tag{14}$$

3 System Methodology

3.1 Data Set Generation

There are two classes of optimization problems. One where there is no explicit formulation and instead we access the objective and constraint functions which are modelled as a black box. We can *query* this black box and get the value of the objective, or derivative of the objective and so on at any point. However, we do not have the closed form expression of any function involved. Such a formulation style is called *Oracle problem description*. We will not address these type of problems in this work and consider it as future work.

In other cases where optimization problems which can be expressed in explicit, closed-form expressions are called *parameterized problems*.

For training a neural network, it is imperative that we require labelled data. We can artificially generate the required data, consisting of parameters of a problem instance and the corresponding solution. To achieve this, we can use random number generators to populate the parameters of any optimization problem. Furthermore to find the optimal primal and dual solutions, one can use any one of the numerous solvers available. Specifically for our data, we used the CVXPY python module coupled with OSQP solver.

For instance, the standard form of a QP is as follows:

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & \frac{1}{2}\boldsymbol{x}^T \boldsymbol{P}\boldsymbol{x} + \boldsymbol{q}^T\boldsymbol{x} + r && \text{(15a)} \\ \text{subject to} \quad & \boldsymbol{G}\boldsymbol{x} \preceq \boldsymbol{h} && \text{(15b)} \\ & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} && \text{(15c)} \end{aligned}$$

where,

$$\boldsymbol{P} \in \mathbb{R}^{n\times n}, \boldsymbol{q} \in \mathbb{R}^n, r \in \mathbb{R}, \boldsymbol{G} \in \mathbb{R}^{m\times n}, \boldsymbol{h} \in \mathbb{R}^m, \boldsymbol{A} \in \mathbb{R}^{p\times n}, \boldsymbol{b} \in \mathbb{R}^p.$$

We use random number generator to populate the entries of the matrices and the vectors (of parameters) in the above expression. However, there is one issue; any vector space V is defined over a field $\mathcal{F}$. If we consider a vector space over the field of real numbers, $\mathbb{R}$, we will never be able to generate a data set that covers any fraction of the entire vector space as, $\forall\, x \in \mathbb{R}, -\infty \leq x \leq \infty$.

To address this issue, we propose a simple Complete Problem Normalization (CPN) that transforms the any original problem and limits the “problem space” meaning it ensures that all entries of the parameter matrices and the parameter vectors are limited to the interval [-1, 1] and that by a simple scaling factor, we can “unnormalize” the problem to get back the original problem and optimal solution from the normalized ones.

3.1.1 Complete Problem Normalization (CPN)

Let Θ be defined as,

$$\Theta = \max\{P_{max}, q_{max}, r, G_{max}, h_{max}, A_{max}, b_{max}\}.$$

where each of the element of the set above is the maximum of absolute values of the entries of a matrix or a vector they belong to. For example,

$$G_{max} \triangleq \max\{|g_{ij}|| g_{ij} \text{is the element } i^{th}\text{row and } j^{th}\text{column}\}.$$

We then do the complete normalization as,

$$\tilde{\boldsymbol{P}} = \boldsymbol{P}/\Theta, \tilde{\boldsymbol{q}} = \boldsymbol{q}/\Theta, \tilde{r} = r/\Theta \tag{16a}$$

$$\tilde{\boldsymbol{G}} = \boldsymbol{G}/\Theta, \tilde{\boldsymbol{h}} = \boldsymbol{h}/\Theta \tag{16b}$$

$$\tilde{\boldsymbol{A}} = \boldsymbol{A}/\Theta, \tilde{\boldsymbol{b}} = \boldsymbol{b}/\Theta \tag{16c}$$

Thus, we see that a CPNed problem will be of the form: problem merely a positively scaled version and that the limits of the entries are as mentioned before.

3.1.2 Why is CPN required?

As mentioned above is not possible to generate a dataset that can cover any non-zero fraction of the problem space. Thus if we generate problem instances from the problem space mentioned above then we can solve any other optimization problem.

3.1.3 Solving problems outside the unit problem space using CPN

Let us consider the same problem. The CPNed problem will be of the form:

$$\min_{\boldsymbol{x}} \quad \frac{1}{2}\boldsymbol{x}^T\tilde{\boldsymbol{P}}\boldsymbol{x} + \tilde{\boldsymbol{q}}^T\boldsymbol{x} + \tilde{r} \tag{17a}$$

$$\text{subject to} \quad \tilde{\boldsymbol{G}}\boldsymbol{x} \preceq \tilde{\boldsymbol{h}} \tag{17b}$$

$$\tilde{\boldsymbol{A}}\boldsymbol{x} = \tilde{\boldsymbol{b}} \tag{17c}$$

If the solution to the normalized problem is $\tilde{\boldsymbol{x}}^*$, then due to the uniform scaling, we get the solution to the original problem as,

$$\boldsymbol{x}^* = \tilde{\boldsymbol{x}}^*.$$

3.1.4 A study of the dataset

4 Experiments, Results and Discussion

References

Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

A Appendix