Lecture 16

Learning Lyapunov Functions

In this lecture, we will explore a method for synthesizing a Lyapunov controller using a neural network.

To synthesize neural network controllers and Lyapunov functions, one must be able to verify that the neural network functions satisfy the Lyapunov condition.

There are two main approaches for doing this:

- Satisfiability Modulo Theories (SMT) solvers, which generalize the boolean satisfiability problem.
- Mixed-Integer Programs.

An interesting observation is the a neural network with a ReLu activation function is amenable to being verified with a mixed-integer program.

16.1. Mixed-Integer Linear Programs

For a moment, let's review some common convex optimization problems.

16.1.1 Linear Program

A linear program optimizes a linear objective function subject to linear equality and inequality constraints.

$$\min c^T x \tag{16.1}$$

$$\text{s.t.} \quad Ax \leq b \tag{16.2}$$

16.1.2 Quadratic Program

A quadratic program optimizes a quadratic objective function subject to linear equality and inequality constraints.

$$\min \frac{1}{2} x^T Q x + c^T x + d \tag{16.3}$$

$$\text{s.t.} \quad Ax \leq b \quad Q > 0 \tag{16.4}$$

16.1.3 Semidefinite Program

A semidefinite program optimizes a linear objective function subject to a matrix inequality.

$$\min c^T x \tag{16.5}$$
$$\text{s.t.} \quad F(x) \geq 0 \tag{16.6}$$
$$F(x) = F_0 + \sum_{i=1}^{N} x_i F_i \tag{16.7}$$

16.2. Mixed-Integer Program

Now let's introduce the notion of an integer program.

$$\min c^T x \tag{16.8}$$
$$\text{s.t.} \quad Ax \leq b \tag{16.9}$$
$$x \geq 0 \tag{16.10}$$
$$x \in \mathbb{Z}^n \tag{16.11}$$

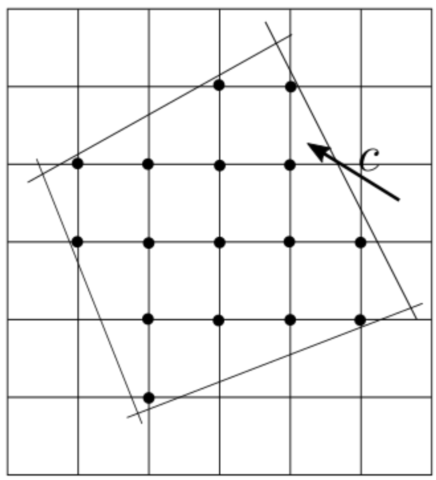

Figure 16.1: Depiction of an integer program

A mixed integer program then has the form:

$$\min c^T x \tag{16.12}$$
$$\text{s.t.} \quad Ax \leq b \tag{16.13}$$
$$x \geq 0 \tag{16.14}$$
$$x_i \in \mathbb{Z}^{n_i} \quad \forall i \in \mathbb{I} \tag{16.15}$$

16.3. Solving Mixed-Integer Programs

How can we solve mixed-integer program?

First we relax the mixed-integer or integer program in to a linear program (i.e., remove $x_i \in \mathbb{Z}^{n_i}, \forall i \in \mathbb{I}$) and solve.

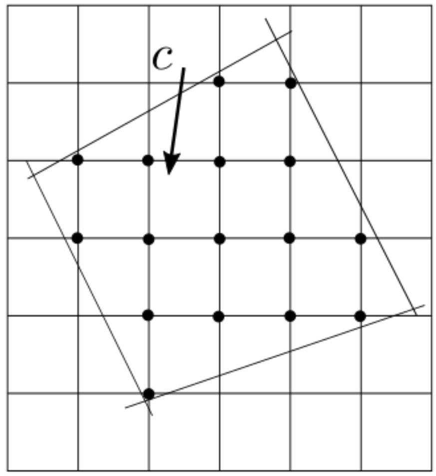

Figure 16.2: Depiction of an integer program where the solution to the relaxed problem is an integer value.

This provides a lower bound on the integer LP, and if the solution is an integer, then it solves the integer linear program.

16.3.1 Branch and Bound Algorithm

Using successive relaxations, we can solve the mixed-integer program via the "Branch and Bound Algorithm".

$$\min c^T x \tag{16.16}$$
$$\text{s.t.} \quad x \in \mathcal{P} \tag{16.17}$$

We recursively partition $\mathcal{P}$ in smaller sets $\mathcal{P}_i$ and solve the sub-problems.

$$\min c^T x \tag{16.18}$$
$$\text{s.t.} \quad x \in \mathcal{P}_i \tag{16.19}$$

Use LP relaxations to discard subproblems that don't lead to the solution.

Example

$$\min -2x_1 - 3x_2 \tag{16.20}$$
$$\text{s.t.} \quad (x_1, x_2) \in \mathcal{P} \tag{16.21}$$

where

$$\mathcal{P} = \left\{ x \in Z_+^2 \middle| \frac{2}{9}x_1 + \frac{1}{4}x_2 \leq 1, \frac{1}{7}x_1 + \frac{1}{3}x_2 \leq 1 \right\}. \tag{16.22}$$

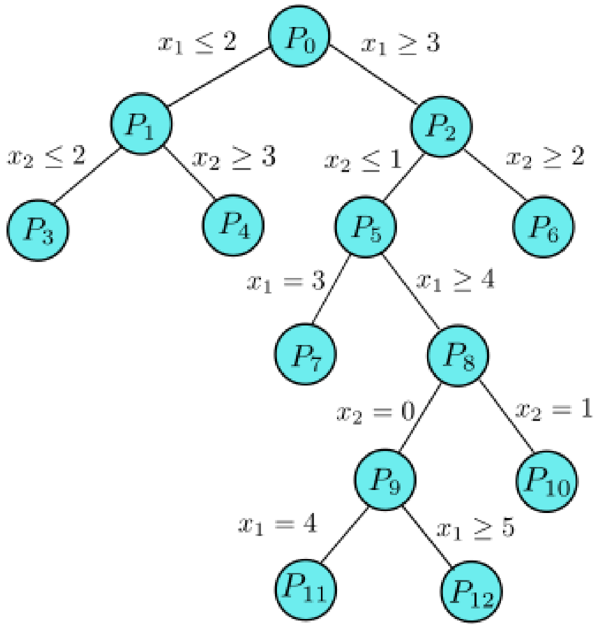

Figure 16.3: Depiction of an integer program where the solution to the relaxed problem is an integer value.

	x^*	P^*
P_0	2.17,2.07	-10.56
P_1	2.00,2.14	-10.43
P_2	3.00,1.33	-10.00
P_3	2.00, 2.00	-10.00
P_4	0.00, 3.00	-9
P_5	3.38,1.00	-9.75
P_6		$+\infty$
P_7	3,1	-9
P_8	4,0.44	-9.3
P_9	4.5,0	-9
P_{10}		∞
P_{11}	4.0,0.00	-8
P_{12}		

16.4. Piecewise Linear Costs

Consider the piecewsie linear cost:

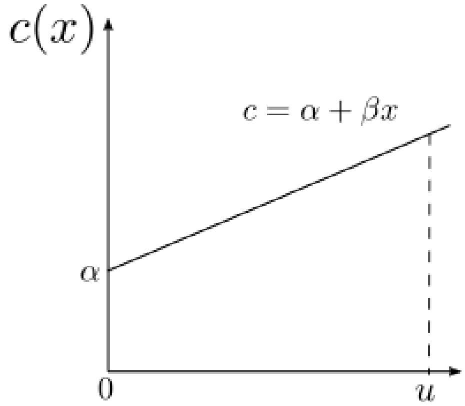

Figure 16.4: Piecewise-linear objective function.

$$c(x) = \begin{cases} \alpha + \beta x & \text{if } 0 \leq x \leq U \\ 0 & \text{otherwsie} \end{cases} \tag{16.23}$$

By introducing the binary variable $y \in \{0, 1\}$, this can be written as:

$$c = \alpha y + \beta x \tag{16.24}$$

$$0 \leq x \leq Uy. \tag{16.25}$$

16.5. ReLU Neural Network

For a general, fully connected neural network, the input/ouput relationship for each layer is:

$$Z_i = \sigma(W_i z_{i-1} + b_i), \quad i = 1...n-1 \tag{16.26}$$

$$Z_n = W_n z_{n-1} + b_n, \quad z_0 = x \tag{16.27}$$

where W_i, b_i are the weights/biases of the i^{th} layer. The activation function $\sigma(\cdot)$ is the leaky ReLU.

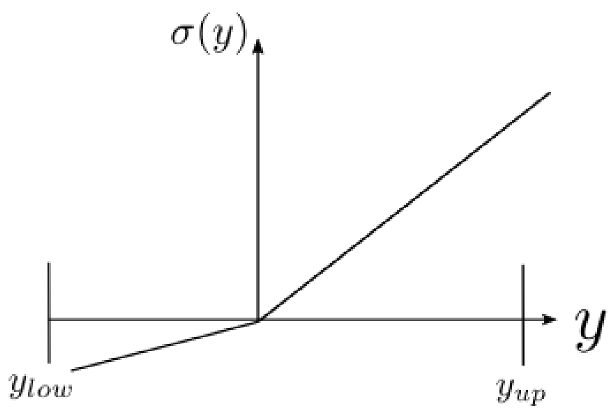

Figure 16.5: Leaky ReLU Network.

$$\sigma(y) = \max(y, cy), \quad 0 \le c < 1 \tag{16.28}$$

where the input y is bounded such that

$$y_{lo} \le y \le y_{up}, \quad y_{lo} < 0, \quad y_{up} > 0. \tag{16.29}$$

You can write the input/output relationship as a mixed-integer linear constraint as follows:

$$w \le cy - (c-1)y_{up}\beta, \quad w \le y - (c-1)y_{lo}(\beta - 1), \tag{16.30}$$

$$\beta \in \{0, 1\}, \tag{16.31}$$

$$w \ge y, w \ge cy. \tag{16.32}$$

Here, β is a binary variable active when $y \ge 0$. The nonlinearity of the NN is the leaky ReLU and can be replaced by constraints in the mixed integer program.

We expect bounded input to the neural network, since we care about the states in the neighborhood of the equilibrium.

16.6. Lyapunov Conditions

We want to find a control policy $u_t = \pi(x_t)$ and a Lyapunov function

$$V(x_t) : \mathbb{R}^{n_x} \to \mathbb{R} \tag{16.33}$$

$$\text{s.t. } V(x_t) > 0 \quad \forall x_t \in S, \quad x_t \ne x^* \tag{16.34}$$

$$V(x_{t+1}) - V(x_t) \le -\epsilon_2 V(x_t) \quad \forall x_t \in S, \quad x_t \ne x^* \tag{16.35}$$

$$V(x^*) = 0 \tag{16.36}$$

$$S = \{x_t | V(x_t) \le \rho\} \text{ and } \epsilon_2 > 0 \tag{16.37}$$

For any state starting in S, the state converges exponentially to the equilibrium state x^*.

16.7. Lyapunov Candidate

We will choose a Lyanpunov candidate of the form

$$V(x_t) = \phi_v(x_t) - \phi_v(x^*) + |R(x_t - x^*)|_1, \tag{16.38}$$

where R is a matrix with full column rank.

- When $V(x_t^*) = 0$, the term $|R(x_t - x^*)|_1$ assists in ensuring that $V(x_t) > 0$.
- $\phi_v(x_t) - \phi_v(x^*)$ is a piecewise affine function of x_t passing through x^*, which could be negative near x^*.
- $R = U(\Sigma + diag(r_1^2 ... r_{n_x}^2))V^T$ to make full column rank.

The control policy is given as

$$u_t = \pi(x_t) = \text{clamp}(\phi_\pi(x_t) - \phi_\pi(x^*) + u^*, u_{min}, u_{max}). \tag{16.39}$$

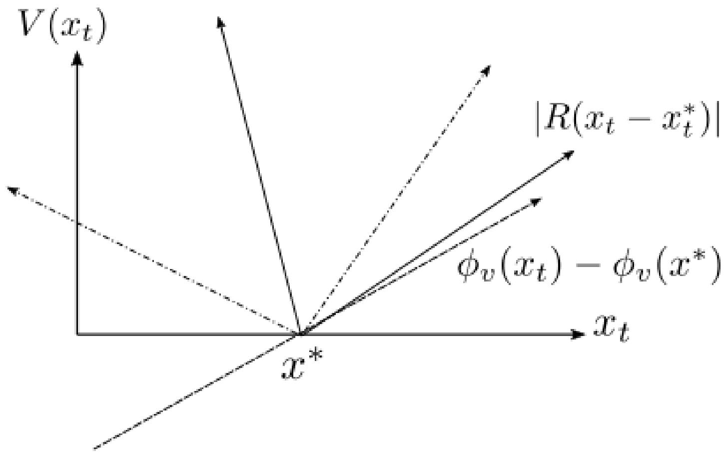

Figure 16.6: Condition imposed on the learned Lyapunov function to ensure positive definiteness.

The Lyapunov condition is a strict inequality.

$$V(x_t) \geq \epsilon_1 |R(x_t - x^*)|, \quad \forall x \in S \quad 0 < \epsilon_1 < 1 \tag{16.40}$$

We find V and use the following optimization program

$$\max_{x_t \in B} \epsilon_1 |R(x_t - x^*)|_1 - V(x_t) \tag{16.41}$$

$$\max_{x_t \in B} V(x_{t+1}) - V(x_t) + \epsilon_2 V(x_t) \tag{16.42}$$

$$\tag{16.43}$$

where B is a bounded polytope.

- If optimal values are zero, this satisfies the Lyapunov condition.
- Maximize a piecewise affine function within a bounded domain (B in this case) through solving a MIP.
- If we maximize the Lyapunov condition, we find the worst counter example (find a whole set of counter examples!).

16.8. Training

Minimize the violation via min − max program as follows:

$$\min_{\theta} \left(\max_{x_t \in B} \epsilon_1 |R(x_t - x^*)|_1 - V(x_t) + \max_{x_t \ \in B} V(x_{t+1}) - V(x_t) + \epsilon_2 V(x_t) \right) \tag{16.44}$$

where θ are parameters in the controller.

This is amenable to an iterative procedure given as

- Solve inner maximization using MIP solver.
- Compute gradient w.r.t. θ and apply gradient descent along the gradient direction.

To compute the gradient of the maximization problem object w.r.t. θ, fix the binary variables and keep the active linear constraints. We have

$$\gamma(\theta) = \max c_\theta^T s + d_\theta \tag{16.45}$$

$$\text{s.t. } A_\theta s = b_\theta \tag{16.46}$$

where $c_\theta, d_\theta, A_\theta, b_\theta$ are all explicit functions of θ.

$$\gamma(\theta) = c_\theta^T A_\theta^{-1} b_\theta + d_\theta \implies \frac{\partial \gamma(\theta)}{\partial \theta} \tag{16.47}$$

via backpropagation.

Bibliography

[1] Hongkai Dai, Benoit Landry, Lujie Yang, Marco Pavone, and Russ Tedrake. Lyapunov-stable neural-network control. *arXiv preprint arXiv:2109.14152*, 2021.