Reliable and Trustworthy Artificial Intelligence

Lecture 10 (Part II): Combining Logic and Deep Learning

Martin Vechev

ETH Zurich

Fall 2022

Now: Logic and Deep Learning

Can we query the network with questions beyond adversarial examples?

Can we enforce properties that the network should satisfy?

What are the guarantees of such methods?

Lecture is based on:

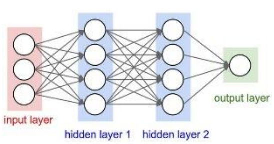
DL2: Training and Querying Neural Networks with Logic, ICML 2019

Combining Logic and Deep Learning

As Deep Learning makes more and more decisions (e.g: bank credit, job applications, university admissions, political elections), it becomes critical to understand how these decisions can be influenced and understood.

Query What should the applicant change to receive a bank credit?

Neural Network NN



Rejected for credit by NN

- income
- residence
- conditions
- job
- …

Deep Learning Query Engine

To be in the > 84% probability of receiving credit, increase income by at least 5K and be employed for at least 3 more months…

Combining Logic and Deep Learning

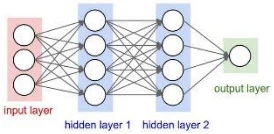
Adversarial examples are in fact just a special case of a query…

Find an image i which gets classified to 9 (truck) where the image i is within some distance of the image deer.

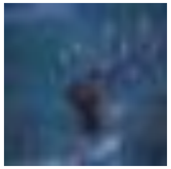
Query

```
find i[32, 32, 3]
where i in [0, 255],
    class(NN(i)) = 9,
    ‖i - deer‖∞ < 25,
    ‖i - deer‖∞ > 5
```

Neural Network NN

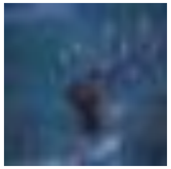


deer



Deep Learning Query Engine

image i



classified as truck by NN!

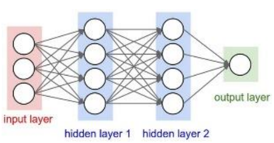
Combining Logic and Deep Learning

Adversarial examples are in fact just a special case of a query…

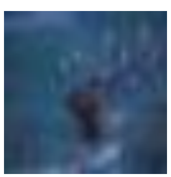
Hmm, but this involves logical constraints and a neural network! How can we unify these?

Query
```
find i[32, 32, 3]
where i in [0, 255],
    class(NN(i)) = 9,
    ‖i − deer‖∞ < 25,
    ‖i − deer‖∞ > 5
```

Neural Network NN

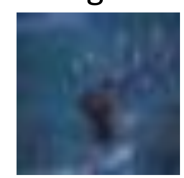


deer



Deep Learning Query Engine

image i

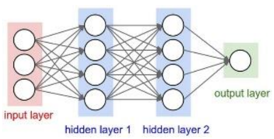


classified as truck by NN!

Combining Logic and Deep Learning

We can also train neural networks to satisfy a logical property

Network Topology



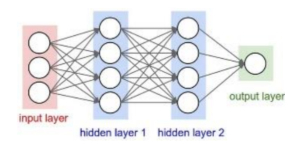
Dataset of images

Logical Property ϕ

Deep Learning + Logic Training

weights θ

Network $\models \phi$



In fact, this can help accuracy as we can label part of the data and specify properties on the remaining, unlabeled data.

Part I:
Querying the Network

Lets first define the logic

We introduce a standard logic with:

- no quantifiers: no $\forall, \exists$
- $\neg$, $\neq$, $\wedge$, $\vee$, $\leq$, $\geq$, <, >, $\Rightarrow$
- functions f: $\mathbb{R}^m \rightarrow \mathbb{R}^n$
- terms: variables, constants: represent vectors of reals
- terms: function application
- terms: arithmetic expressions over terms (e.g., +)

Comparison operations on vectors are done point-wise.

If a and b are vectors of dimension 2, then a = b is written as $a[0] = b[0] \wedge a[1] = b[1]$

The logic used in our example query

$$\mathbf{class}\big(\mathrm{NN(i)}\big) = 9 \wedge \|\mathrm{i} - \mathrm{deer}\|_\infty < 25 \;\wedge\; \|\mathrm{i} - \mathrm{deer}\|_\infty > 5$$

Lets expand this a bit

The logic used in our example query

$$\mathbf{class}\big(\mathrm{NN}(i)\big) = 9 \wedge \|i - \mathrm{deer}\|_\infty < 25 \wedge \|i - \mathrm{deer}\|_\infty > 5$$

$$\overbrace{\bigwedge_{j=1,\, j\neq 9}^{k} \mathrm{NN}(i)[j] < \mathrm{NN}(i)[9]}^{\text{Syntactic sugar}} \wedge \|i - \mathrm{deer}\|_\infty < 25 \wedge \|i - \mathrm{deer}\|_\infty > 5$$

This is the actual formula being used after expanding the syntactic sugar.

Here, k is the number of labels.

The logic used in our example query

$$\phi \doteq \bigwedge_{j=1,\, j\neq 9}^{k} \mathrm{NN(i)[j]} < \mathrm{NN(i)[9]} \wedge \|\mathrm{i} - \mathrm{deer}\|_\infty < 25 \;\wedge\; \|\mathrm{i} - \mathrm{deer}\|_\infty > 5$$

Here we have 2 functions: NN and the norm $\| \quad \|_\infty$.

Function NN returns a probability distribution over labels.

We have 4 constants: 9, 5, 25 and deer (real-valued vector)

We have 1 free variable i

Goal: find a value for i that satisfies the constraint above

How do we solve this problem?

$$\phi \doteq \bigwedge_{j=1,\, j\neq 9}^{k} \mathrm{NN(i)[j]} < \mathrm{NN(i)[9]} \wedge \|\mathrm{i} - \mathrm{deer}\|_\infty < 25 \wedge \|\mathrm{i} - \mathrm{deer}\|_\infty > 5$$

One approach to finding the value of i is to invoke standard a constraint solver (e.g., SMT solver which generalize SAT to richer theories). Unfortunately, unless the network NN is really small, these solvers simply time out (one of the problem is the non-linear constraints that the network exhibits). Thus, we need another approach.

Solve as optimization

$$\phi \doteq \bigwedge_{\mathrm{j}=1,\,\mathrm{j}\neq 9}^{\mathrm{k}} \mathrm{NN(i)[j]} < \mathrm{NN(i)[9]} \ \wedge \|\mathrm{i} - \mathrm{deer}\|_{\infty} < 25 \ \wedge \ \|\mathrm{i} - \mathrm{deer}\|_{\infty} > 5$$

Instead, the idea is to introduce a particular translation T of logical formulas into a differentiable loss function T(ϕ) to be solved with (mostly standard) optimization, where the translation T has a certain property.

Wanted Property of Translation

Theorem: $\forall x$, $T(\phi)(x) = 0$ **if and only if** x satisfies ϕ

What this theorem says is that: if we can find a solution x where the loss function of ϕ is 0, then that solution x is a satisfiable assignment to ϕ, that is, it is a solution to our original problem. It also states that if x satisfies ϕ then the loss function at x is 0

Optimize to find a solution

Theorem: $\forall$x, T(ϕ)(x) = 0 **if and only if** x satisfies ϕ

Given this theorem, our goal is to find an assignment $\mathrm{x} = \mathrm{i}$ such that T(ϕ)(i) is 0.

We can use standard gradient-based optimization to minimize the function T(ϕ). There can potentially be many solutions which set the function to 0.

Translation: Formula to Loss

Logical Term	Translation
$t_1 \leq t_2$	$\max(0, t_1 - t_2)$
$t_1 \neq t_2$	$[t_1 = t_2]$
$t_1 = t_2$	$T(t_1 \leq t_2 \wedge t_2 \leq t_1)$
$t_1 < t_2$	$T(t_1 \leq t_2 \wedge t_1 \neq t_2)$
$\varphi \vee \psi$	$T(\varphi) \cdot T(\psi)$
$\varphi \wedge \psi$	$T(\varphi) + T(\psi)$

Two comments on the translation:

- Translation is recursive: translating a term defined in a way which refers to how the constituents of the term are translated.
- The resulting loss function is non-negative

Intuition: example

Logical Term	Translation
$\varphi \vee \psi$	$\mathrm{T}(\varphi) \cdot \mathrm{T}(\psi)$

what this says is:

if one of the terms is 0, then the entire translated expression will be 0 (that is, the formula is satisfied).

Example: a satisfying formula

Logical formula:

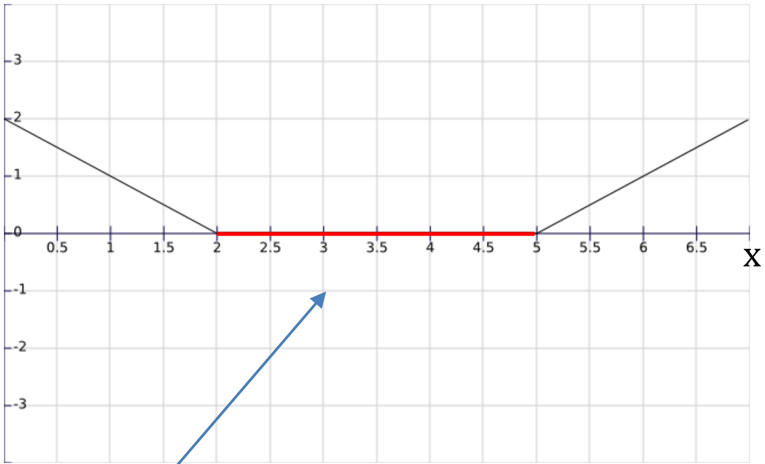
$$x \geq 2 \wedge x \leq 5$$

Translated Loss:

$$\max(0, 2 - x) + \max(0, x - 5)$$

Satisfying assignments:

Any value between 2 and 5, inclusive



The function is 0 when x is between 2 and 5.
We need to find one such assignment.

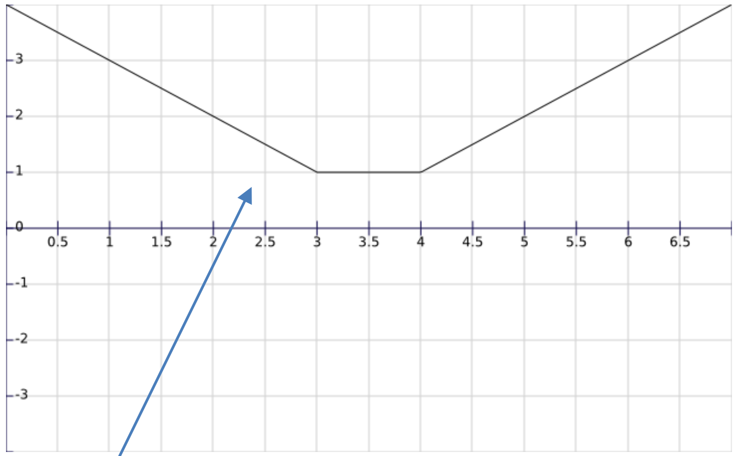
Example: an unsatisfiable formula

Logical formula:

$x \geq \color{red}{4} \wedge x \leq \color{red}{3}$

Translated Loss:

$\max(0, \color{red}{4} - x) + \max(0, x - 3)$



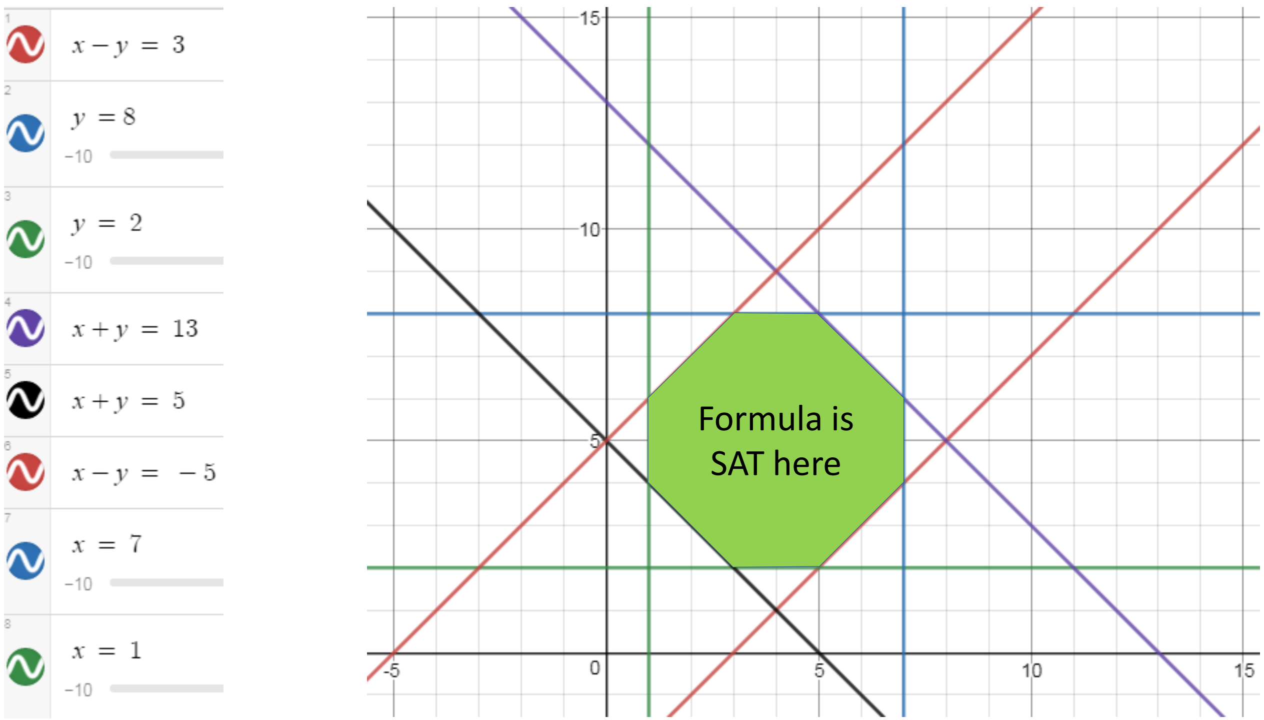
Satisfying assignments:

There are no satisfying assignments

The function is never 0

Something more fun: Octagon

See plot here:
https://www.desmos.com/calculator/kw38cpoirk



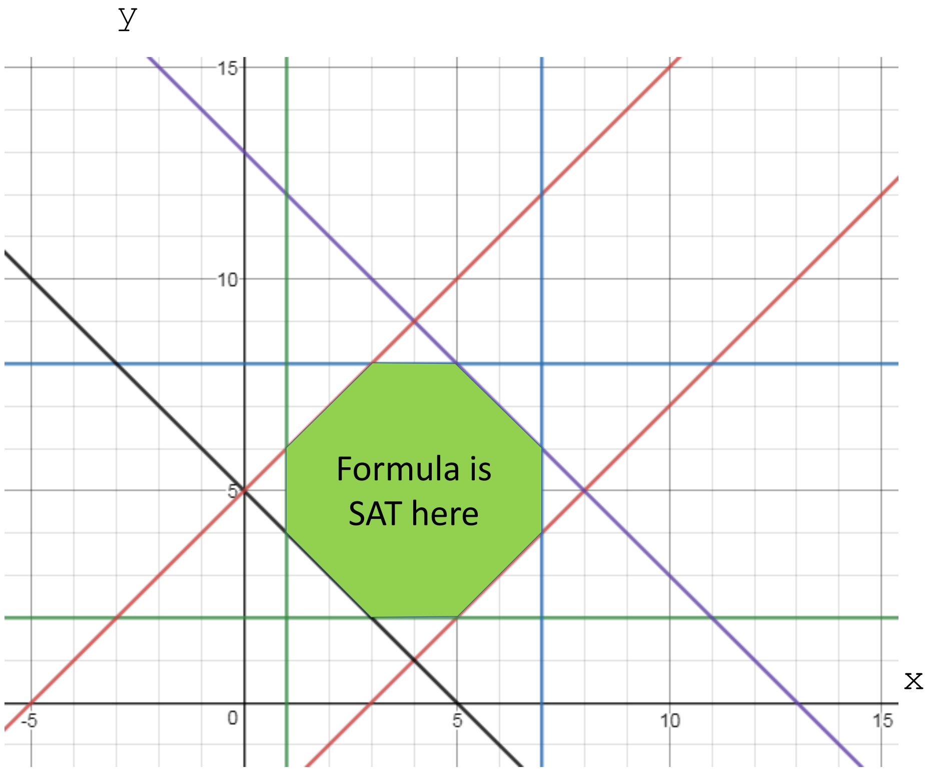

```
x - y ≤ 3        ∧
y ≤ 8            ∧
y ≥ 2            ∧
x + y ≤ 13       ∧
x + y ≥ 5        ∧
x ≥ 1            ∧
x − y ≥ -5       ∧
x ≤ 7
```

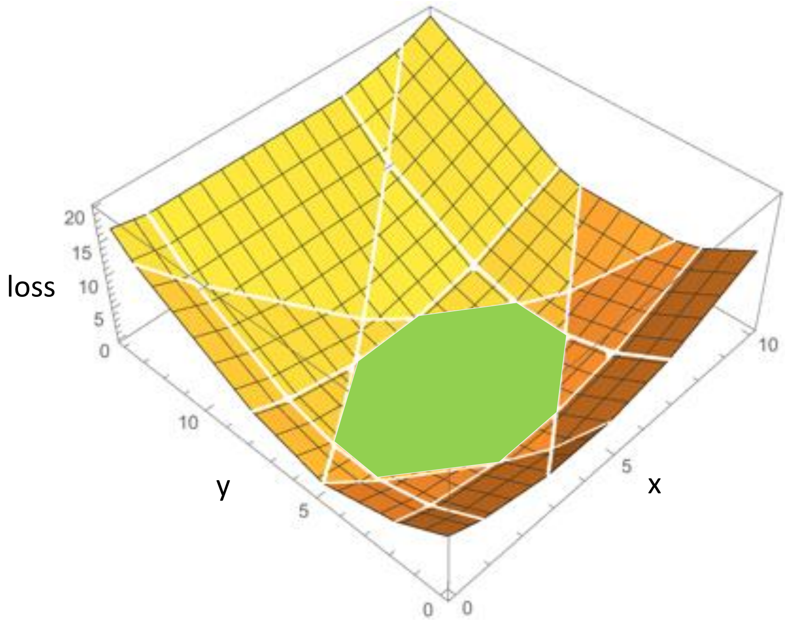
Translate

```
max(0, x-y-3)     +
max(0, y-8)       +
max(0, 2-y)       +
max(0, x+y-13)    +
max(0, 5-x-y)     +
max(0, 1-x)       +
max(0, -5-x+y)    +
max(0, x-7)
```

Plot with Mathematica

Plot3D
[Max[0, 2 - y] + Max[0, -3 + x - y] + Max[0, -8 + y] + Max[0, -13 + x + y] + Max[0, 5 - x - y] + Max[0, 1 - x] + Max[0, -5 - x + y] + Max[0, x - 7], {x, 0, 10}, {y, 0, 14}]





We can visually see that for values of x between 1 and 7 and of y between 2 and 8, the loss is 0

Back to Neural Nets

$$\bigwedge_{j=1,\, j\neq 9}^{k} \mathrm{NN(i)[j]} < \mathrm{NN(i)[9]} \;\wedge \|\mathrm{i} - \mathrm{deer}\|_{\infty} < 25 \;\wedge\; \|\mathrm{i} - \mathrm{deer}\|_{\infty} > 5$$

Original Formula ϕ

$$\mathrm{NN(i)[1]} < \mathrm{NN(i)[9]} \;\wedge$$
$$\mathrm{NN(i)[2]} < \mathrm{NN(i)[9]} \;\wedge$$
$$\mathrm{NN(i)[3]} < \mathrm{NN(i)[9]} \;\wedge$$
$$\mathrm{NN(i)[4]} < \mathrm{NN(i)[9]} \;\wedge$$
$$\mathrm{NN(i)[5]} < \mathrm{NN(i)[9]} \;\wedge$$
$$\mathrm{NN(i)[6]} < \mathrm{NN(i)[9]} \;\wedge$$
$$\mathrm{NN(i)[7]} < \mathrm{NN(i)[9]} \;\wedge$$
$$\mathrm{NN(i)[8]} < \mathrm{NN(i)[9]} \;\wedge$$
$$\boldsymbol{\|\mathrm{i} - deer\|_{\infty} < 25 \;\wedge}$$
$$\boldsymbol{\|i - deer\|_{\infty} > 5}$$

Translated Loss $\mathrm{T}(\phi)$

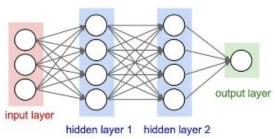
$$\max(0, \mathrm{NN(i)[1]} - \mathrm{NN(i)[9]}) + [\mathrm{NN(i)[1]} = \mathrm{NN(i)[9]}]$$
$$+ \max(0, \mathrm{NN(i)[2]} - \mathrm{NN(i)[9]}) + [\mathrm{NN(i)[2]} = \mathrm{NN(i)[9]}]$$
$$+ \max(0, \mathrm{NN(i)[3]} - \mathrm{NN(i)[9]}) + [\mathrm{NN(i)[3]} = \mathrm{NN(i)[9]}]$$
$$+ \max(0, \mathrm{NN(i)[4]} - \mathrm{NN(i)[9]}) + [\mathrm{NN(i)[4]} = \mathrm{NN(i)[9]}]$$
$$+ \max(0, \mathrm{NN(i)[5]} - \mathrm{NN(i)[9]}) + [\mathrm{NN(i)[5]} = \mathrm{NN(i)[9]}]$$
$$+ \max(0, \mathrm{NN(i)[6]} - \mathrm{NN(i)[9]}) + [\mathrm{NN(i)[6]} = \mathrm{NN(i)[9]}]$$
$$+ \max(0, \mathrm{NN(i)[7]} - \mathrm{NN(i)[9]}) + [\mathrm{NN(i)[7]} = \mathrm{NN(i)[9]}]$$
$$+ \max(0, \mathrm{NN(i)[8]} - \mathrm{NN(i)[9]}) + [\mathrm{NN(i)[8]} = \mathrm{NN(i)[9]}]$$
$$+ \max(0, \|\mathrm{i} - \mathrm{deer}\|_{\infty} - 25) + [\|\mathrm{i} - \mathrm{deer}\|_{\infty} = 25]$$
$$+ \max(0, 5 - \|\mathrm{i} - \mathrm{deer}\|_{\infty}) + [\|\mathrm{i} - \mathrm{deer}\|_{\infty} = 5]$$

*Note: **box constraints** can be tricky to optimize with gradient descend, so may need to take out convex constraints before translation and project or use LBFGS-B for box ones.

Training the Network with Logic

(aka: Generalized Adversarial Training beyond Robustness)

Neural Network Topology



Dataset of images

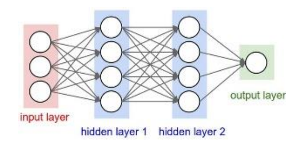
Logical Property ϕ

(e.g., fairness, next lecture)

Deep Learning
+ Logic Training

weights θ

Network $\models \phi$



Principles:

- We still use **the same logic** as before
- We still ``compile'' **property** ϕ **into a loss** as before
- We need a way to define the **optimization problem** now

Problem Statement

find	θ
maximize	$\rho(\theta)$
where	$\rho(\theta) = \mathbf{E}_{s \sim D} [\forall \boldsymbol{z} \ . \ \phi(\boldsymbol{z}, \ s, \ \theta)]$

What this says is: we want to find such parameters/weights θ for the network, so the expected value of the property increases.

Note that we even allow restricted quantified formulas here.

Rephrasing : Step I

became minimization

find such $\boldsymbol{z}$ where violation is maximized

find θ

minimize $\rho(\theta)$

where $\rho(\theta) = \mathbf{E}_{s \sim D}[\max_{\boldsymbol{z}} \neg\phi(\mathbf{z},\ s,\ \theta)]$

What this says is: we want to find such parameters/weights θ for the neural network, so that the maximum violation of the property ϕ is minimized.

This is essentially: **generalized adversarial training beyond robustness**

Rephrasing: Step II

find θ

minimize $\rho(\theta)$

where $\rho(\theta) = \mathbf{E}_{s \sim D}[\mathrm{T}(\phi)(bz,\ s, \theta)]$

and $bz = \underset{z}{\mathbf{argmin}}(\mathrm{T}(\neg\phi)(z,\ s, \theta))$

minimize the expected value of a 'bad' counter-example

find a 'bad' counter example

The translation leads to a differentiable function which we can optimize. Intuitively, we are trying to get a bad violation of the formula and then to find a network that minimizes its effect.

Solving the inner minimization problem

find	θ
minimize	$\rho(\theta)$
where	$\rho(\theta) = \mathbf{E}_{s\sim D}[\mathrm{T}(\phi)(bz,\ s,\theta)]$
and	$bz = \underset{z}{\mathbf{argmin}}(\mathrm{T}(\neg\phi)(z,\ s,\theta))$

In principle, we can use a standard optimizer to solve for the inner minimization problem. However, variable z can participate in all kinds of constraints in ϕ. Even if its just norm constraints that z participates in, SGD-style optimizers can have a hard time. Thus, we focus on a restricted fragment where z participates in constraints that restrict z to be a convex set where we have an efficient algorithm for projection (a closed form solution). Note that in general, projection onto arbitrary convex sets is hard.

Example: Generating the Loss

$$\phi(z,x,\theta) = \|x - z\|_\infty \leq \epsilon \Rightarrow NN_\theta(z)[3] > \delta$$

we want to enforce the constraint $\forall z.\, \phi(z,x,\theta)$

expansion of $\Rightarrow$

$$\phi(z,x,\theta) = \neg\|x - z\|_\infty \leq \epsilon \vee NN_\theta(z)[3] > \delta$$

negation $\neg$

$$\neg\phi(z,x,\theta) = \|x - z\|_\infty \leq \epsilon \wedge NN_\theta(z)[3] \leq \delta$$

translation to loss

$$loss(z,x,\theta) = \underbrace{\max(0, \|x - z\|_\infty - \epsilon) + \max(0, NN_\theta(z)[3] - \delta)}$$

difficult to solve: minimization over arbitrary constraints

A possible solution to minimizing the loss

this part aims to restrict z to be in the L_∞ ball around x of size ϵ

$$\underset{z}{\mathbf{argmin}}\ loss(z, x, \theta) = \underbrace{\boxed{\max(0, \|x - z\|_\infty - \epsilon)}} + \boxed{\max(0, NN_\theta(z)[3] - \delta)}$$

split the problem

L_∞ ball around x of size ϵ

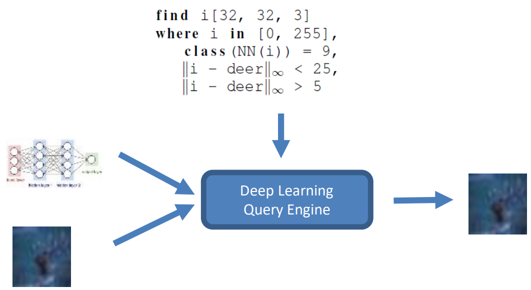
$$\underset{z}{\mathbf{argmin}}\ loss(z, x, \theta) = \boxed{\max(0, NN_\theta(z)[3] - \delta)} \qquad \boxed{L_\infty(x, \epsilon)}$$

Solve with Projected Gradient Descent (PGD) while projecting z onto the L_∞ ball

Note: in general, efficient projections (closed form solutions) on convex sets is a hard problem. Such algorithms exist for L_1, L_2, L_∞ and some others. Because of this, in practice, the logic is restricted to having z participate only in constraints where **efficient projections** are possible.

Lecture (Part II) Summary

Combine Deep Learning with Logic



Logic to loss

Logical Term	Translation
$t_1 \leq t_2$	$\max(0, t_1 - t_2)$
$t_1 \neq t_2$	$[t_1 = t_2]$
$t_1 = t_2$	$T(t_1 \leq t_2 \wedge t_2 \leq t_1)$
$t_1 < t_2$	$T(t_1 \leq t_2 \wedge t_1 \neq t_2)$
$\varphi \vee \psi$	$T(\varphi) \cdot T(\psi)$
$\varphi \wedge \psi$	$T(\varphi) + T(\psi)$

Training with logic as maximization

find θ
maximize $\rho(\theta)$

where $\rho(\theta) = \mathop{\mathbf{E}}_{s \sim D} [\forall \boldsymbol{z} \,.\, \phi(\boldsymbol{z}, s, \theta)]$