Obedient machine learning

Machine learning models that provably obey constraints

Sebastijan Dumancic

joint work with Kshitij Goyal, Hendrik Blockeel

Is AI trustworthy?

New AI can guess whether you're gay or straight from a photograph

An algorithm deduced the sexuality of people on a dating site with up to 91% accuracy, raising tricky ethical questions

FROM POLITICOPRO

Dutch scandal serves as a warning for Europe over risks of using algorithms

The Dutch tax authority ruined thousands of lives after using an algorithm to spot suspected benefits fraud – and critics say there is little stopping it from happening again.

Bernard Parker, left, was rated high risk; Dyl

Machine Bias

There's software used across the country to predict future criminals. And it's biased against blacks.

by Julia Angwin, Jeff Larson, Surya Mattu and Lauren Kirchner, ProPublica
May 23, 2016

Uber's self-driving operator charged over fatal crash

ARTIFICIAL INTELLIGENCE | OPINION

AI-Influenced Weapons Need Better Regulation

The weapons are error-prone and could hit the wrong targets

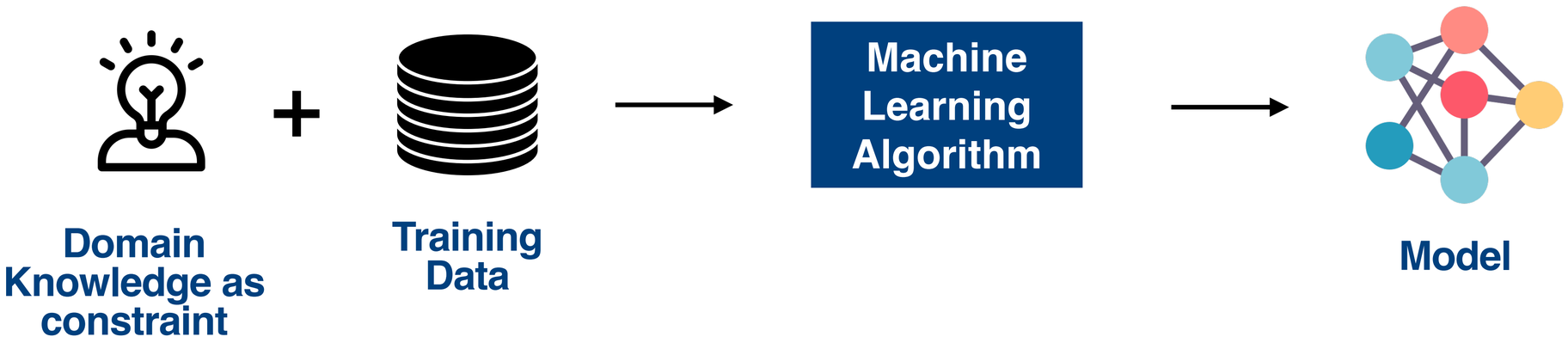
Towards trustworthy systems

The problem lies in our approach: not everything is in training data...

Towards trustworthy systems

The problem lies in our approach: not everything is in training data...

... many desires for trustworthiness have to be incorporated in some other ways

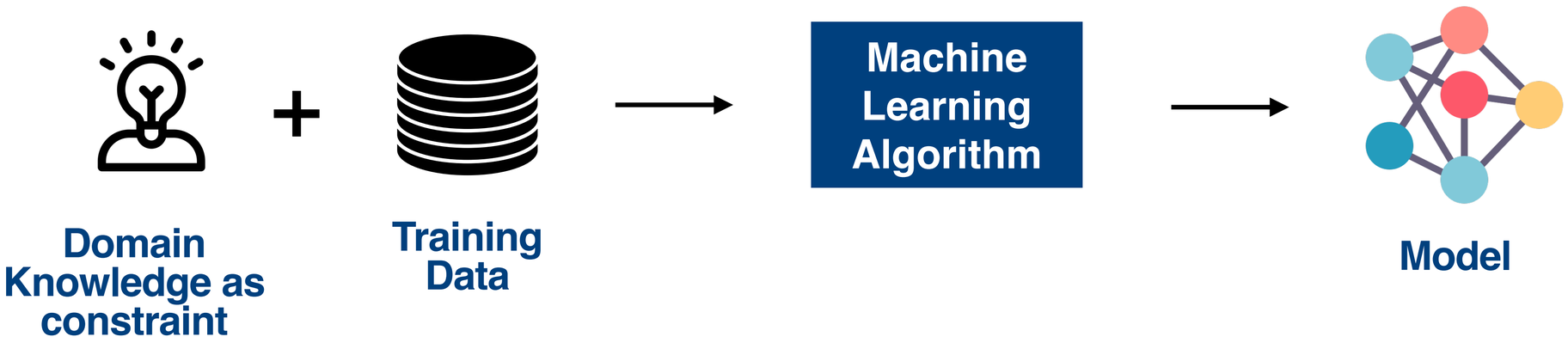


Towards trustworthy systems

The problem lies in our approach: not everything is in training data...

... many desires for trustworthiness have to be incorporated in some other ways

The law requires a certification that the system satisfies constraints



Desiderata

Learning machine learning models that
- guarantee constraint satisfaction
- of a wide range of constraints
- on all possible instances

ML and constraints: A note on constraints

Syntactic Constraints

Constraints on the model structure

Semantic Constraints

Constraints on the behaviour of the model predictions

A note on constraints

Syntactic Constraints

Constraints on the model structure

- **Box Constraints:** upper and lower bound on one of more parameters of the model
- **Depth in decision trees:** to learn smaller trees which are more interpretable

Semantic Constraints

Constraints on the behaviour of the model predictions

- **Safety Constraint:** If there's a stationary object right in front of a moving autonomous car, the car must make a hard stop in all such scenarios.
- **Multi Target Predictions with Constrained Outputs:** The speech to text translation must follow certain linguistic constraints for all predictions.

What have we tried so far

Loss + $\lambda \times$ penalty

Semantic loss, DL2,
Semantic-Based Regularisation

Constraints are soft and
not guaranteed

Very difficult to encode anything
complex-ish

Specific architecture

MultiplexNet

Constraints are guaranteed

Very difficult to support wide constraints

Semantic layers

If only we had….

Techniques that solve constraints

Constraint satisfaction problems

Constraint satisfaction problems

Decision variables =
which colour is a state?

```
% Colouring Australia using nc colours
int: nc = 3;

var 1..nc: wa;   var 1..nc: nt;  var 1..nc: sa;   var 1..nc: q;
var 1..nc: nsw;  var 1..nc: v;   var 1..nc: t;
```

Constraint satisfaction problems

Decision variables =
which colour is a state?

```
% Colouring Australia using nc colours
int: nc = 3;

var 1..nc: wa;   var 1..nc: nt;  var 1..nc: sa;   var 1..nc: q;
var 1..nc: nsw;  var 1..nc: v;   var 1..nc: t;

constraint wa != nt;
constraint wa != sa;
constraint nt != sa;
constraint nt != q;
constraint sa != q;
constraint sa != nsw;
constraint sa != v;
constraint q != nsw;
constraint nsw != v;
```

Constraints =
neighbours should have different colours

Constraint satisfaction problems

Decision variables =
which colour is a state?

```
% Colouring Australia using nc colours
int: nc = 3;

var 1..nc: wa;   var 1..nc: nt;  var 1..nc: sa;   var 1..nc: q;
var 1..nc: nsw;  var 1..nc: v;   var 1..nc: t;

constraint wa != nt;
constraint wa != sa;
constraint nt != sa;
constraint nt != q;
constraint sa != q;
constraint sa != nsw;
constraint sa != v;
constraint q != nsw;
constraint nsw != v;
solve satisfy;
```

Constraints =
neighbours should have different colours

Constraint satisfaction

Boolean Satisfiability (SAT)

$\neg a \vee (b \wedge c)$

$a = False$

Satisfiability Modulo Theories

$(a > 3) \wedge (a < 3)$

$unsat$

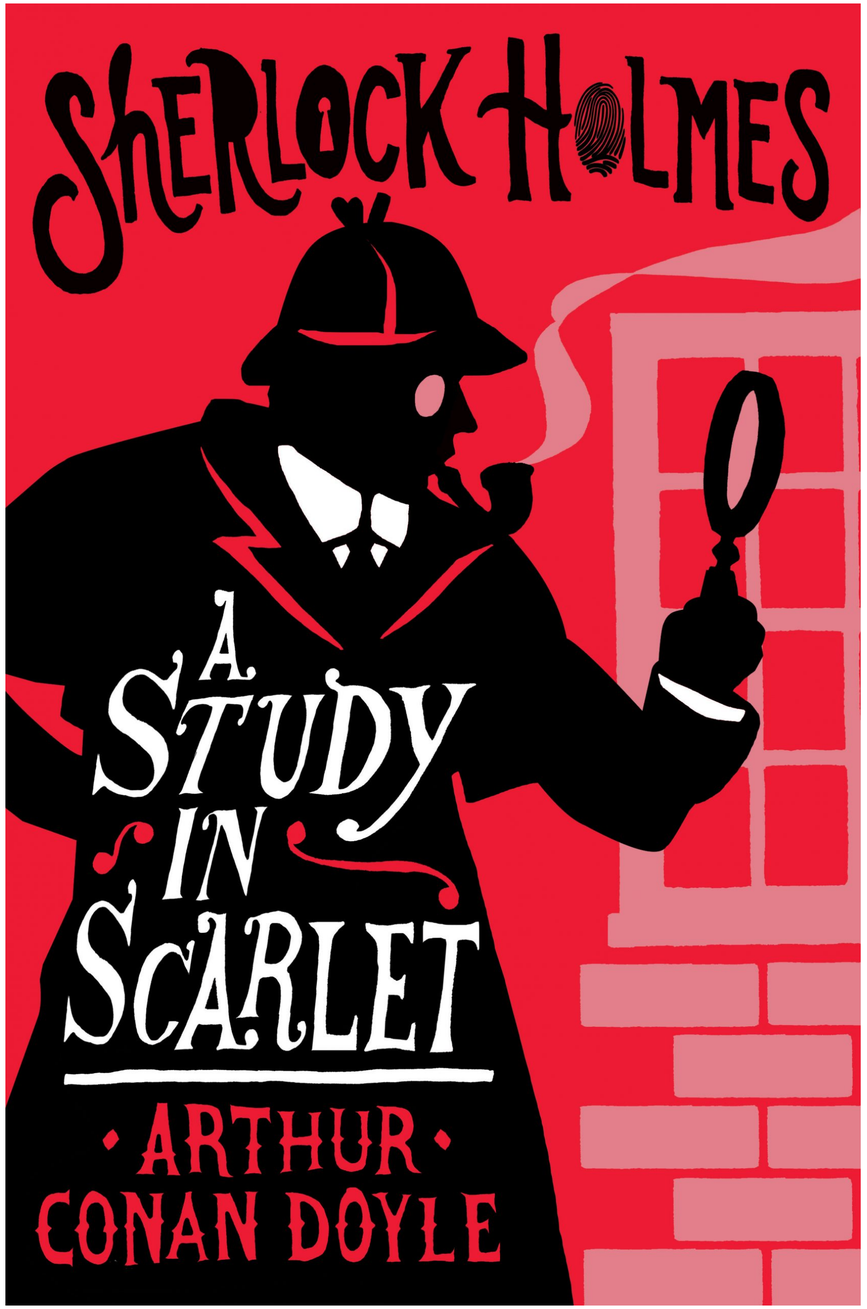
Maximum Satisfiability (MaxSAT/MaxSMT)

$\phi_h : \neg a \wedge (b \vee c)$ $\phi_s : a \wedge b$

$a = False, b = True$

A study in...

making logic do something it was not supposed to do



SaDe:
ML+Constraints with satisfiability descent

Desiderata revisited

Given
- a dataset D
- (domain) constraints C
- domain of instances I

Desiderata revisited

Given
- a dataset D
- (domain) constraints C
- domain of instances I

x_1	x_2	y
1	4	0
2	3	0
-3	2	1

Desiderata revisited

Given
- a dataset D
- (domain) constraints C
- domain of instances I

x_1	x_2	y
1	4	0
2	3	0
-3	2	1

$$\forall x_1, x_2 \quad f(x_1, x_2) > 0$$

Desiderata revisited

Given
- a dataset D
- (domain) constraints C
- domain of instances I

x_1	x_2	y
1	4	0
2	3	0
-3	2	1

$$\forall x_1, x_2 \quad f(x_1, x_2) > 0$$

$$x_1 \in [-4,2] \qquad x_2 \in [0,6]$$

Desiderata revisited

Given
- a dataset D
- (domain) constraints C
- domain of instances I

Learn a model that
- satisfies constraints C
  on entire instance space I
- maximises predictive performance

A closer look into the learning problem

Learn a model that

- satisfies constraints C
  on entire instance space I
- maximises predictive performance

A closer look into the learning problem

Learn a model that
- satisfies constraints C
  on entire instance space I
- maximises predictive performance

Point 1

Impossible to ensure with gradient descent and only from data

A closer look into the learning problem

Learn a model that
- satisfies constraints C
  on entire instance space I
- maximises predictive performance

Point 1

Impossible to ensure with gradient descent and only from data

Point 2

Domain constraints need to be satisfied

Machine learning as MaxSMT

$\mathbf{x_1}$	$\mathbf{x_2}$	$\mathbf{y}$
1	4	0
2	3	0
-3	2	1

$$\forall x_1, x_2 \quad f(x_1, x_2) > 0$$

$$x_1 \in [-4,2] \qquad x_2 \in [0,6]$$

Machine learning as MaxSMT

x_1	x_2	y
1	4	0
2	3	0
-3	2	1

Translate model decisions into soft constraints

Classification

$$f_{\mathbf{w}}([1,4]) < 0$$
$$f_{\mathbf{w}}([2,3]) < 0$$
$$f_{\mathbf{w}}([-3,2]) > 0$$

Regression

$$|f_{\mathbf{w}}([1,4]) - 0| \leq e$$
$$|f_{\mathbf{w}}([2,3]) - 0| \leq e$$
$$|f_{\mathbf{w}}([-3,2]) - 1| \leq e$$

$$\forall x_1, x_2 \quad f(x_1, x_2) > 0$$

$$x_1 \in [-4,2] \qquad x_2 \in [0,6]$$

Machine learning as MaxSMT

x_1	x_2	y
1	4	0
2	3	0
-3	2	1

Translate model decisions into soft constraints

Classification

$$f_w([1,4]) < 0$$
$$f_w([2,3]) < 0$$
$$f_w([-3,2]) > 0$$

Regression

$$|f_w([1,4]) - 0| \leq e$$
$$|f_w([2,3]) - 0| \leq e$$
$$|f_w([-3,2]) - 1| \leq e$$

$$\forall x_1, x_2 \quad f(x_1, x_2) > 0$$

Translate domain constrains into hard constraints

$$\forall x_1 \in [-4,2],\ x_2 \in [0,6]\ f_w(x_1, x_2) > 0$$

$$x_1 \in [-4,2] \qquad x_2 \in [0,6]$$

Machine learning as MaxSMT

Assuming a linear model $f(x_1, x_2) = w_1 x_1 + w_2 x_2 + w3$

Machine learning as MaxSMT

Assuming a linear model $f(x_1, x_2) = w_1 x_1 + w_2 x_2 + w3$

Decision variables = model parameters

var $w_1 : float$

var $w_2 : float$

var $w_3 : float$

Machine learning as MaxSMT

Assuming a linear model $f(x_1, x_2) = w_1 x_1 + w_2 x_2 + w3$

Decision variables = model parameters

$$\text{var } w_1 : float$$
$$\text{var } w_2 : float$$
$$\text{var } w_3 : float$$

Hard constraint = domain constraint

$$\forall x_1 \in [-4,2],\ x_2 \in [0,6]\ \ w_1 x_2 + w_2 x_2 + w_3 > 0$$

Machine learning as MaxSMT

Assuming a linear model $f(x_1, x_2) = w_1 x_1 + w_2 x_2 + w3$

Decision variables = model parameters

$$\text{var } w_1 : float$$
$$\text{var } w_2 : float$$
$$\text{var } w_3 : float$$

Hard constraint = domain constraint

$$\forall x_1 \in [-4,2],\ x_2 \in [0,6]\ w_1 x_2 + w_2 x_2 + w_3 > 0$$

Soft constraints = model predictions

$$4w_1 + 1w_2 + w_3 < 0$$
$$2w_1 + 3w_2 + w_3 < 0$$
$$-3w_1 + 2w_2 + w_3 > 0$$

Representation of domain constraints

Goal: Enforce constraint(s) on the predictions
for all possible instances in the input domain

Representation of domain constraints

Goal: Enforce constraint(s) on the predictions
for all possible instances in the input domain

➞ **Universal Quantifiers (∀)**

Representation of domain constraints

Goal: Enforce constraint(s) on the predictions for all possible instances in the input domain

➞ **Universal Quantifiers (∀)**

$$\forall \mathbf{x} \in \times_{i=1}^{n} [l_i, u_i], \mathbf{x} \models P \Rightarrow f_{\mathbf{w}}(\mathbf{x}) \models C$$

Domain Bounds

Constraint enforced on all feasible points the input can take

Input Property

Constraint is enforced whenever a relevant property is satisfied by the input

Output Constraint

The output is constrained in a certain way

Representation of domain constraints

Goal: Enforce constraint(s) on the predictions for all possible instances in the input domain

→ **Universal Quantifiers (∀)**

Solving for such a constraint with constraint solvers potentially allows to find a model ($f_{\mathbf{w}}(\,.\,)$) that satisfies the constraint for all predictions

$$\forall \mathbf{x} \in \times_{i=1}^{n} [l_i, u_i], \mathbf{x} \models P \Rightarrow f_{\mathbf{w}}(\mathbf{x}) \models C$$

Domain Bounds

Constraint enforced on all feasible points the input can take

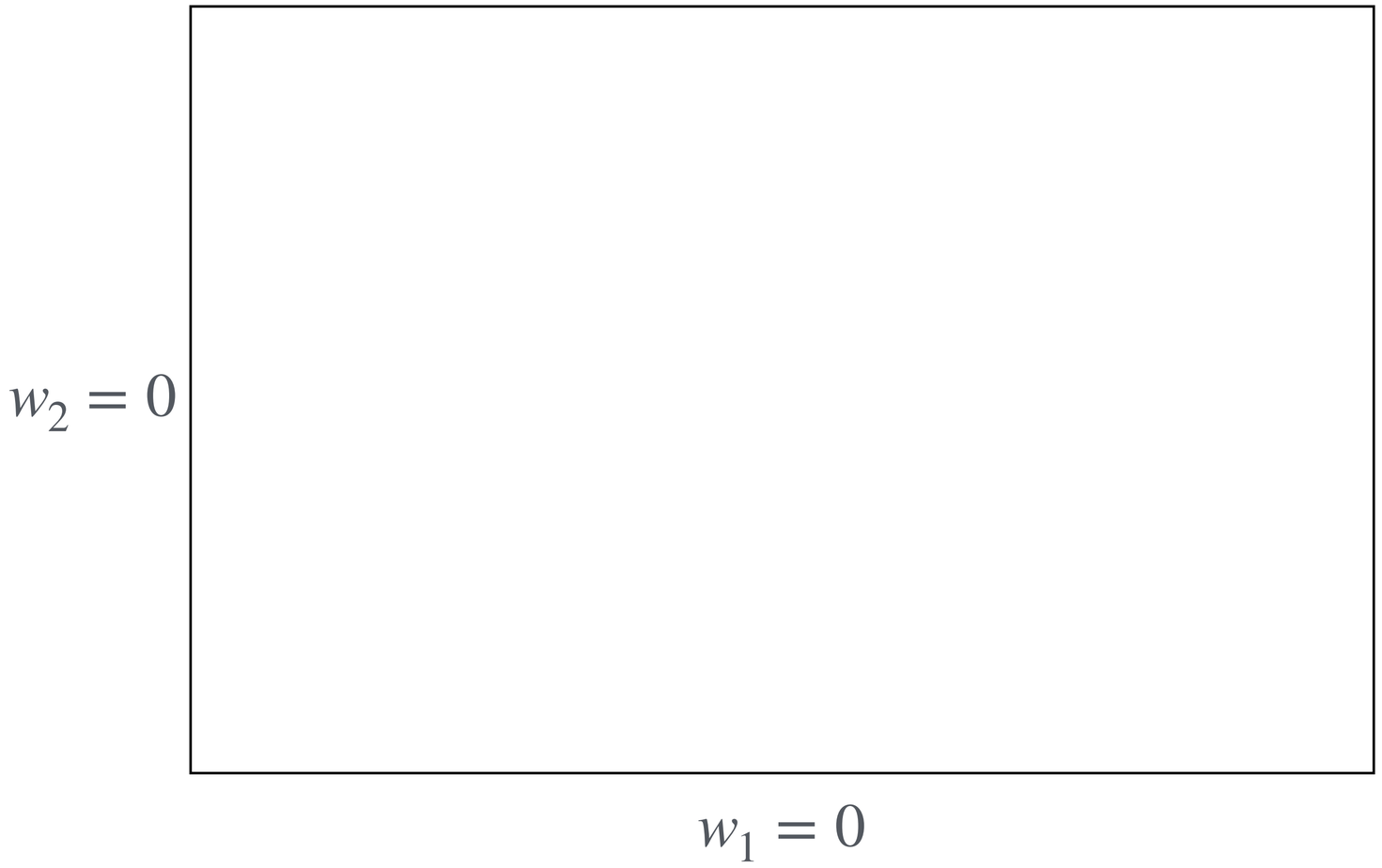
Input Property

Constraint is enforced whenever a relevant property is satisfied by the input

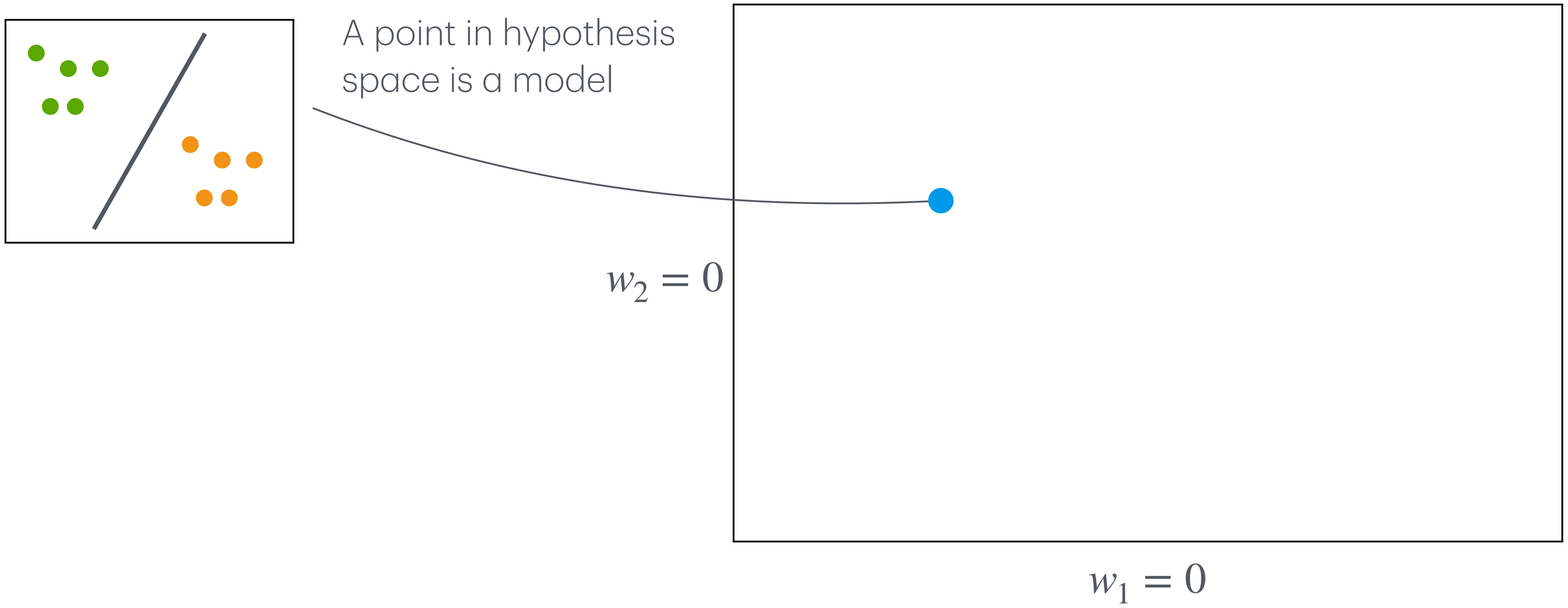
Output Constraint

The output is constrained in a certain way

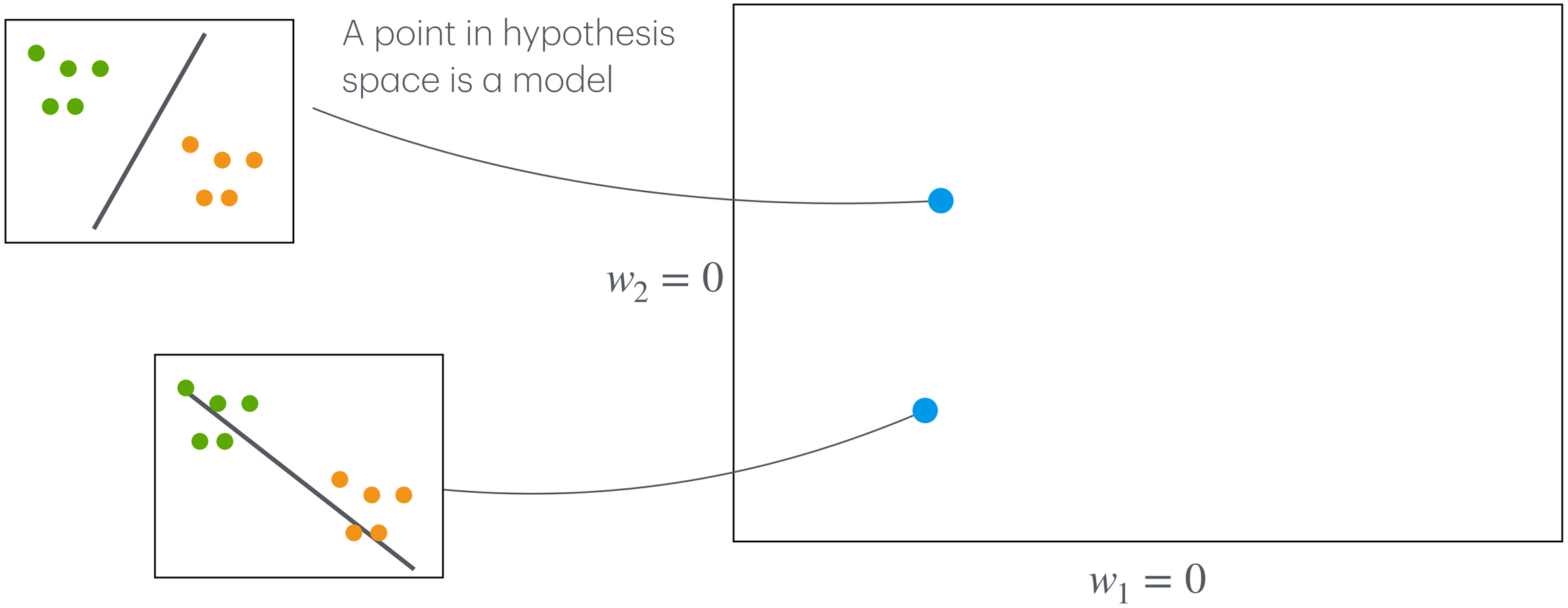
What are we doing here?



What are we doing here?

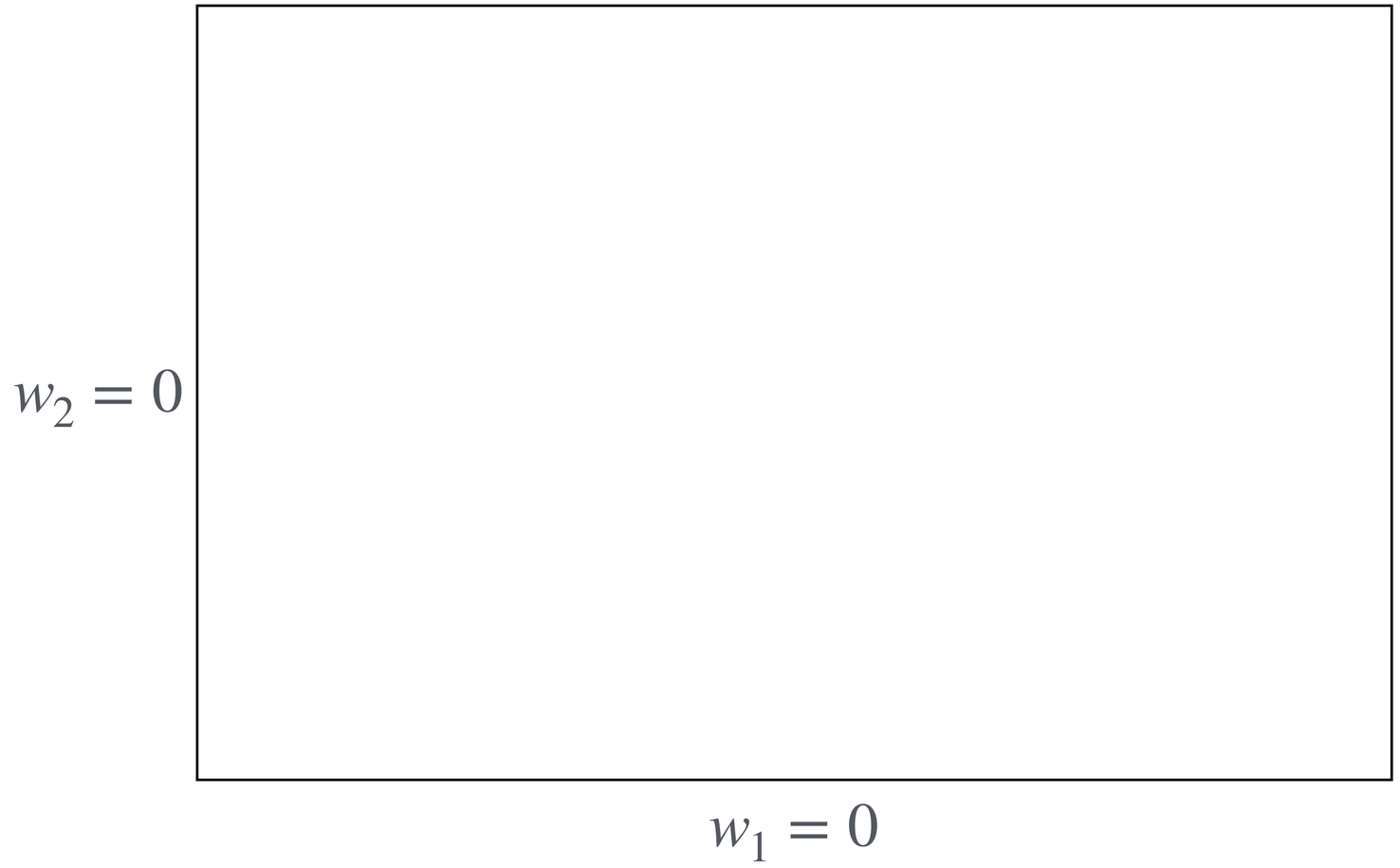


What are we doing here?



What are we doing here?

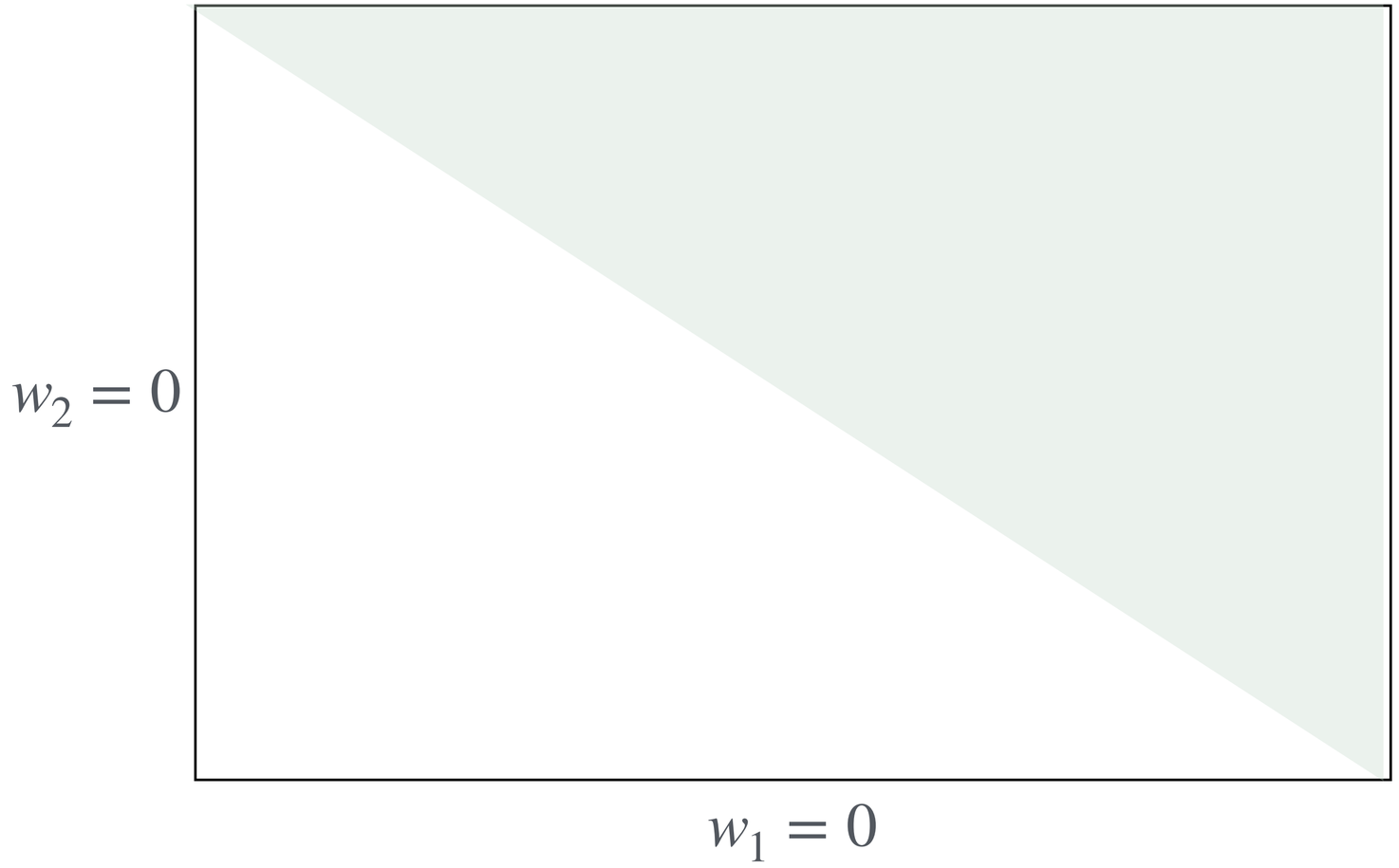
Hard constraints partition
your hypothesis space into
feasible and infeasible regions



What are we doing here?

Hard constraints partition your hypothesis space into feasible and infeasible regions

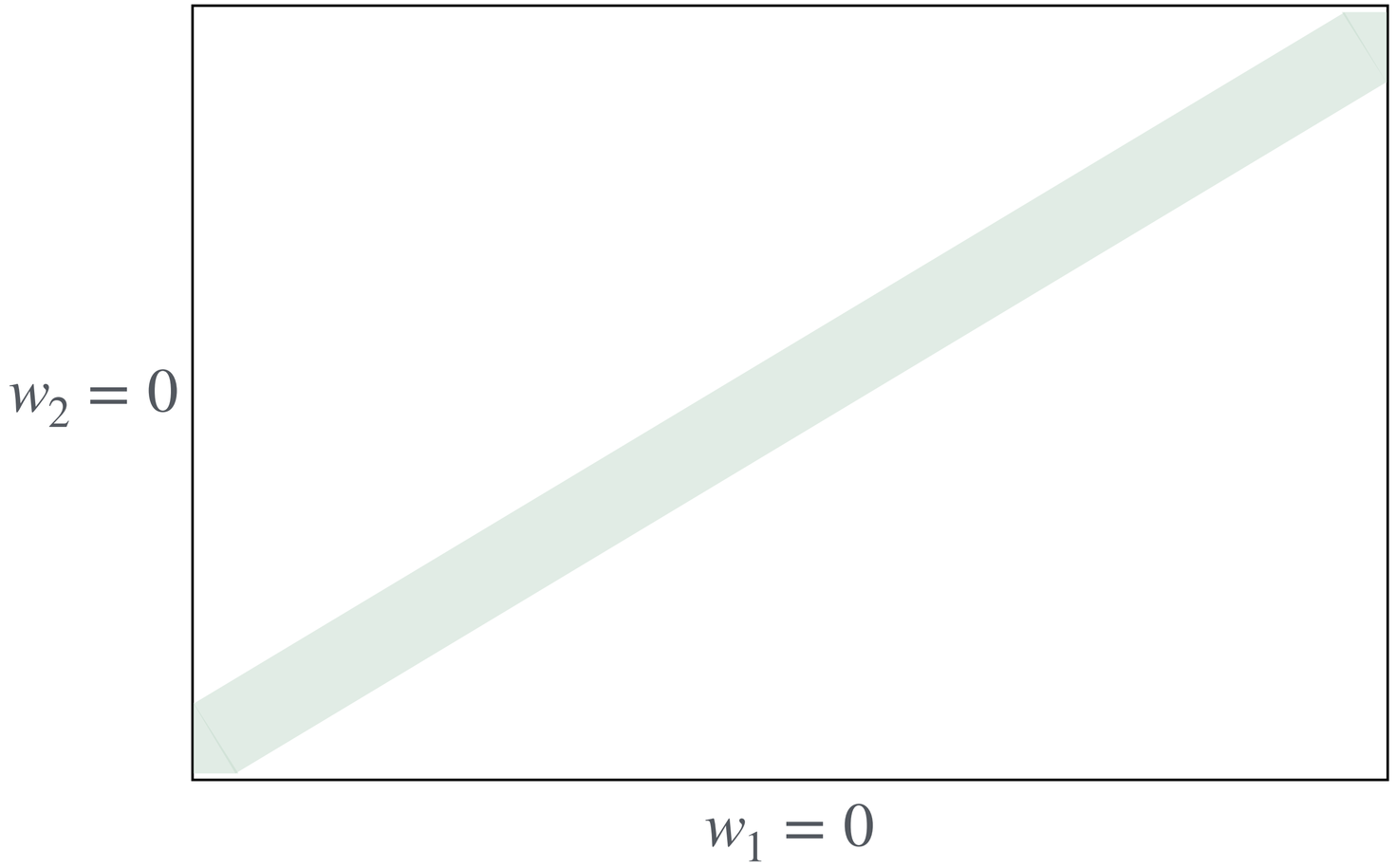
$$x + y > 0$$



What are we doing here?

Hard constraints partition your hypothesis space into feasible and infeasible regions

$x - y < 3$

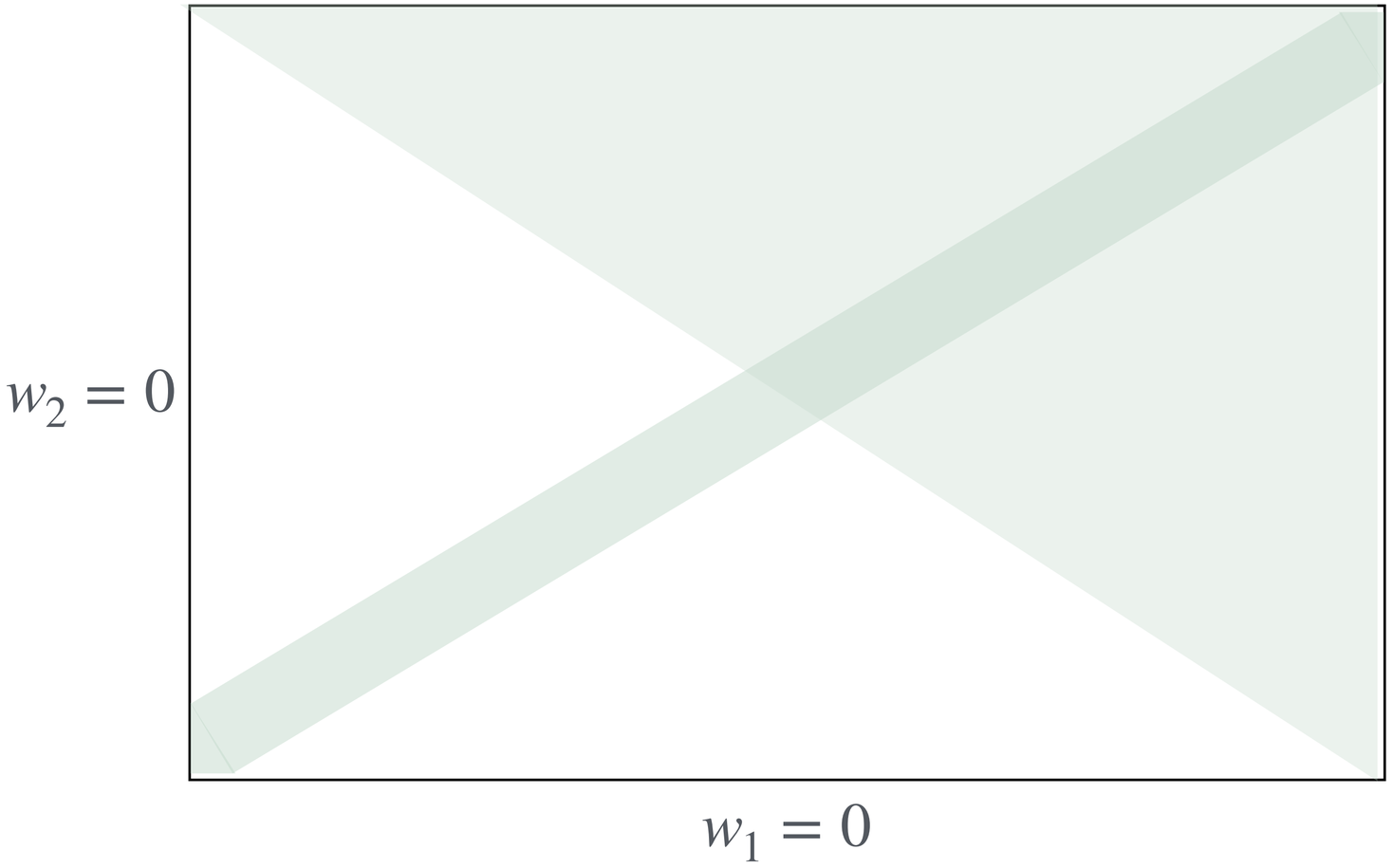


What are we doing here?

Hard constraints partition your hypothesis space into feasible and infeasible regions

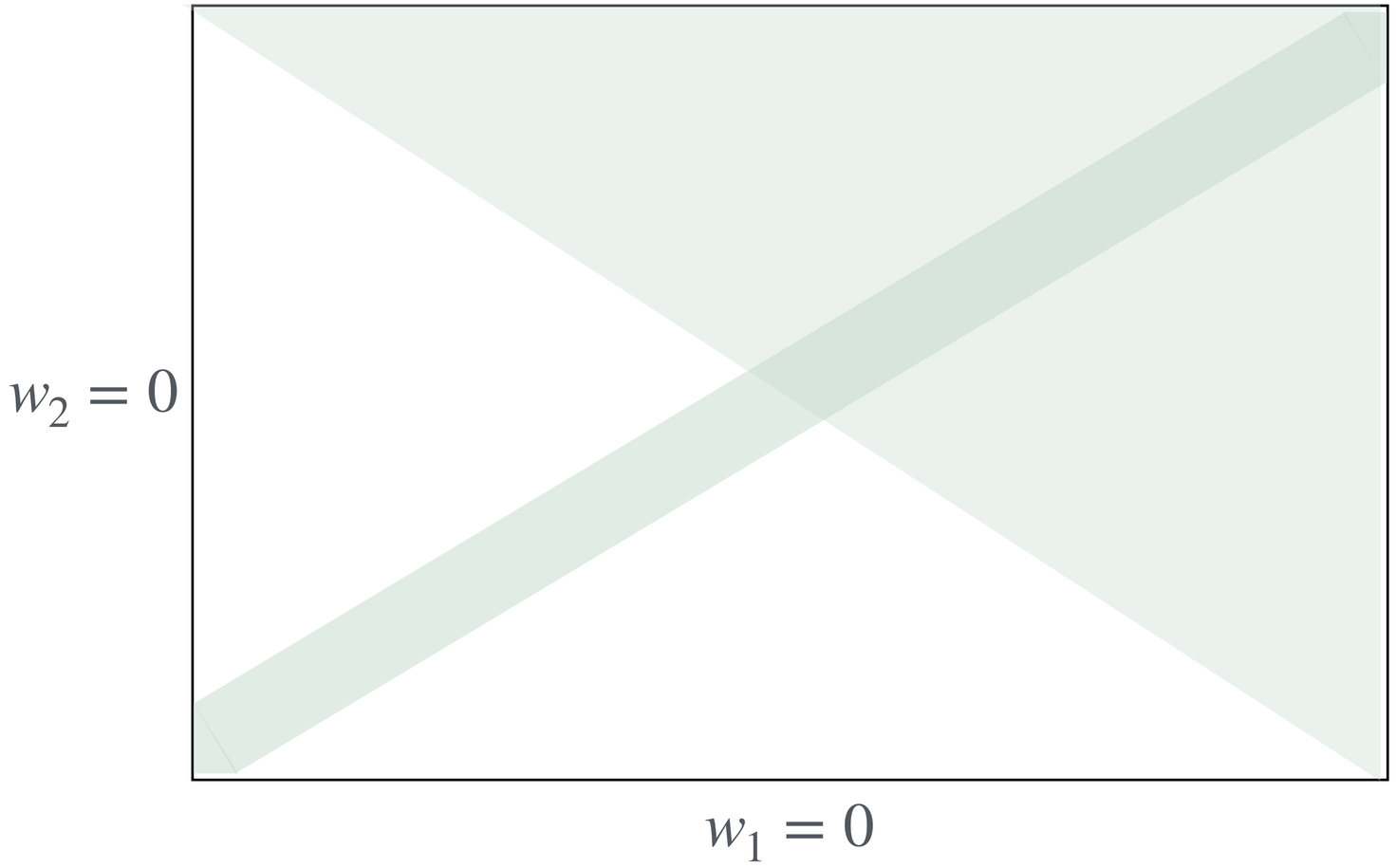
$$x + y > 0$$

$$x - y < 3$$



What are we doing here?

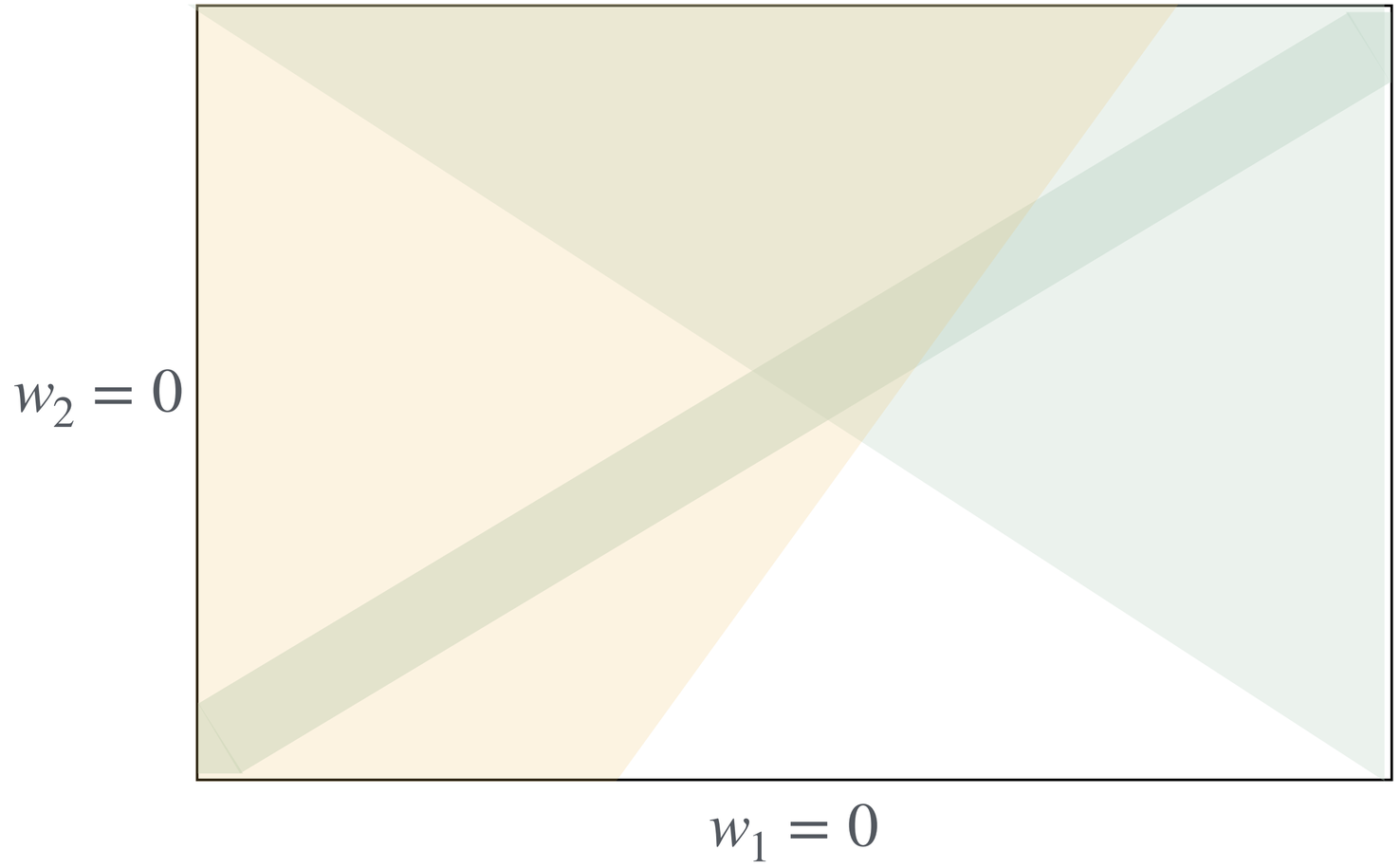
Soft constraints introduce
further optional partitioning



What are we doing here?

Soft constraints introduce
further optional partitioning

$$-3x + y > 0$$

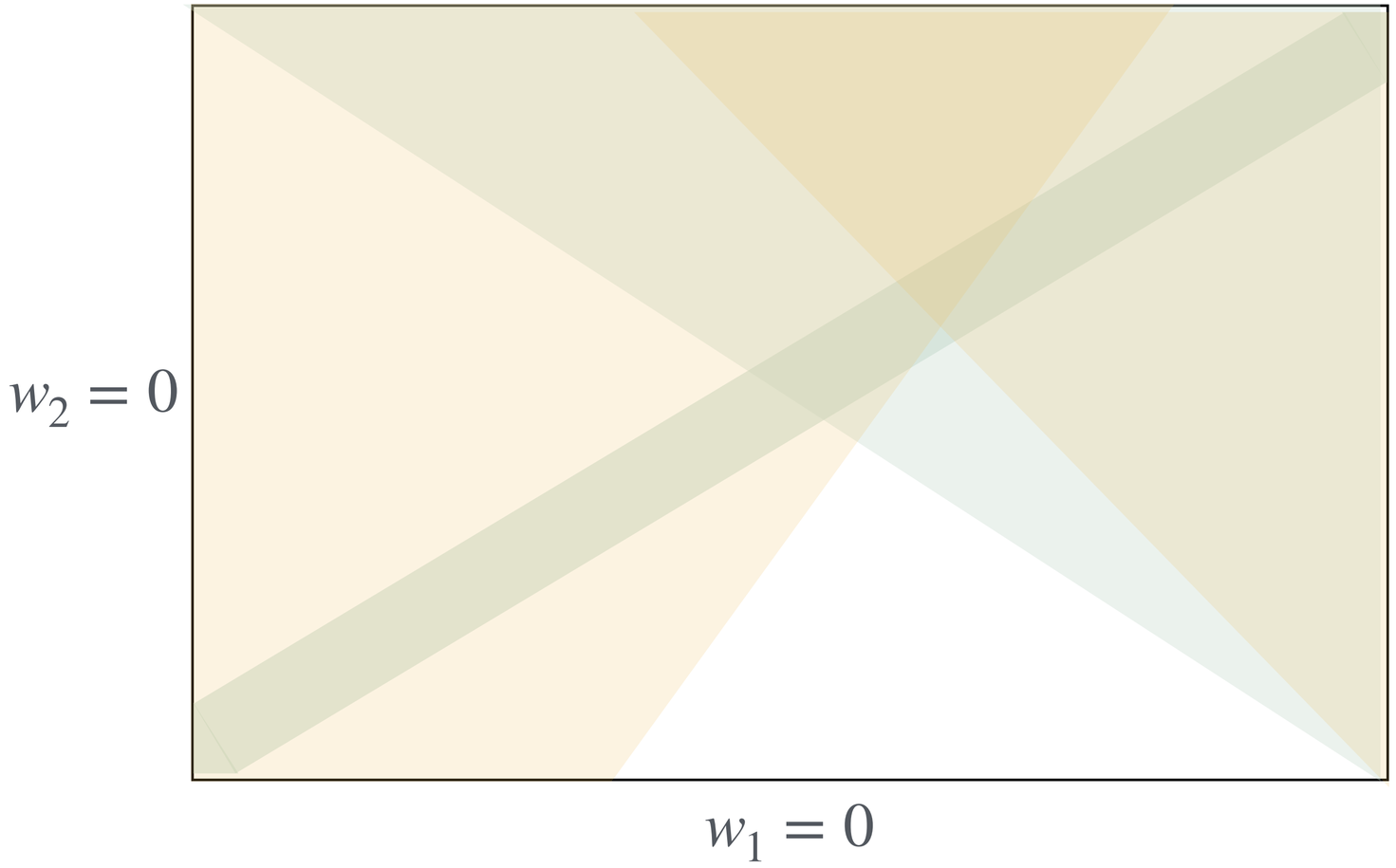


What are we doing here?

Soft constraints introduce
further optional partitioning

$$-3x + y > 0$$

$$4x + y < 0$$

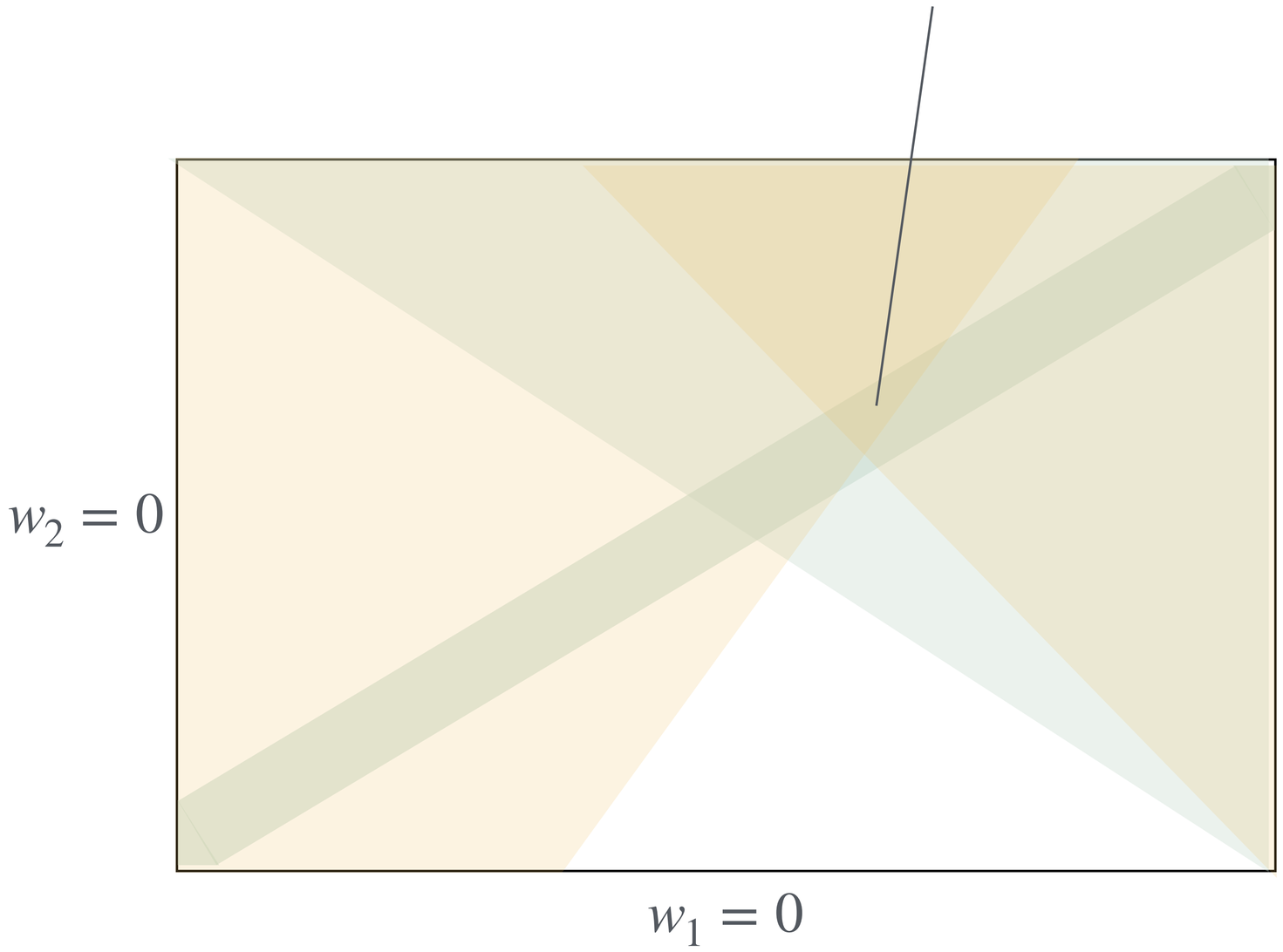


What are we doing here?

Soft constraints introduce
further optional partitioning

$$-3x + y > 0$$

$$4x + y < 0$$



This works only on small models!

Why do neural networks scale?

Why do neural networks scale?

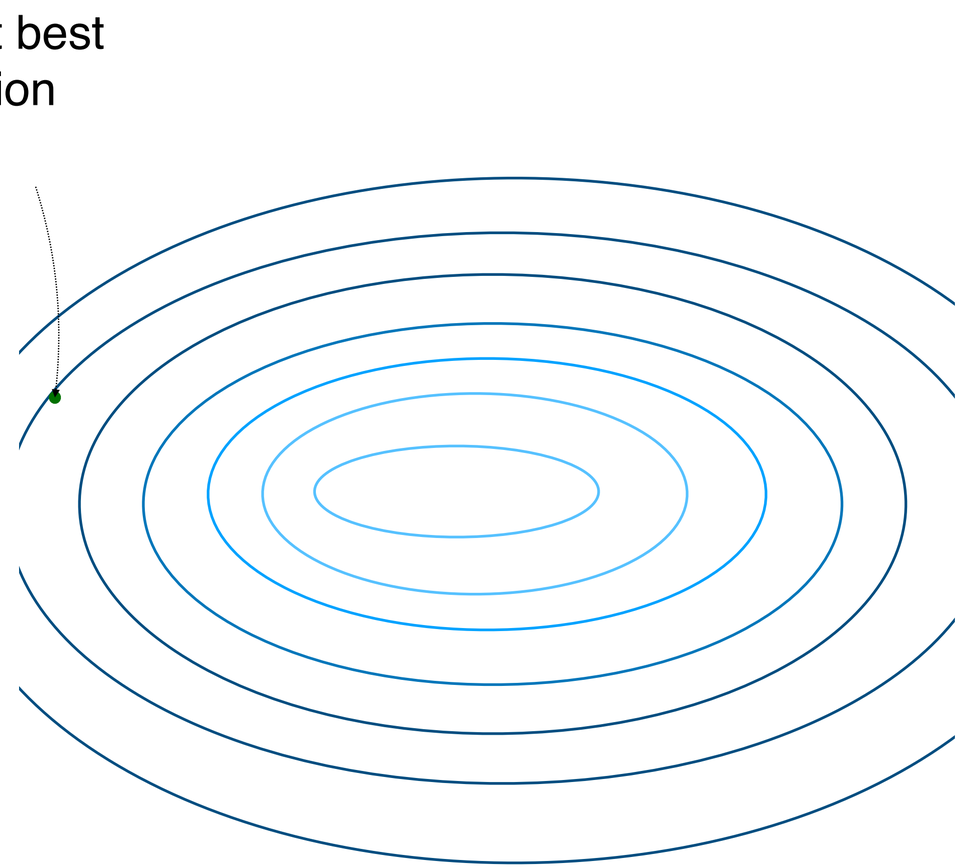
(mini) batching

stochastic gradient descent

small local changes

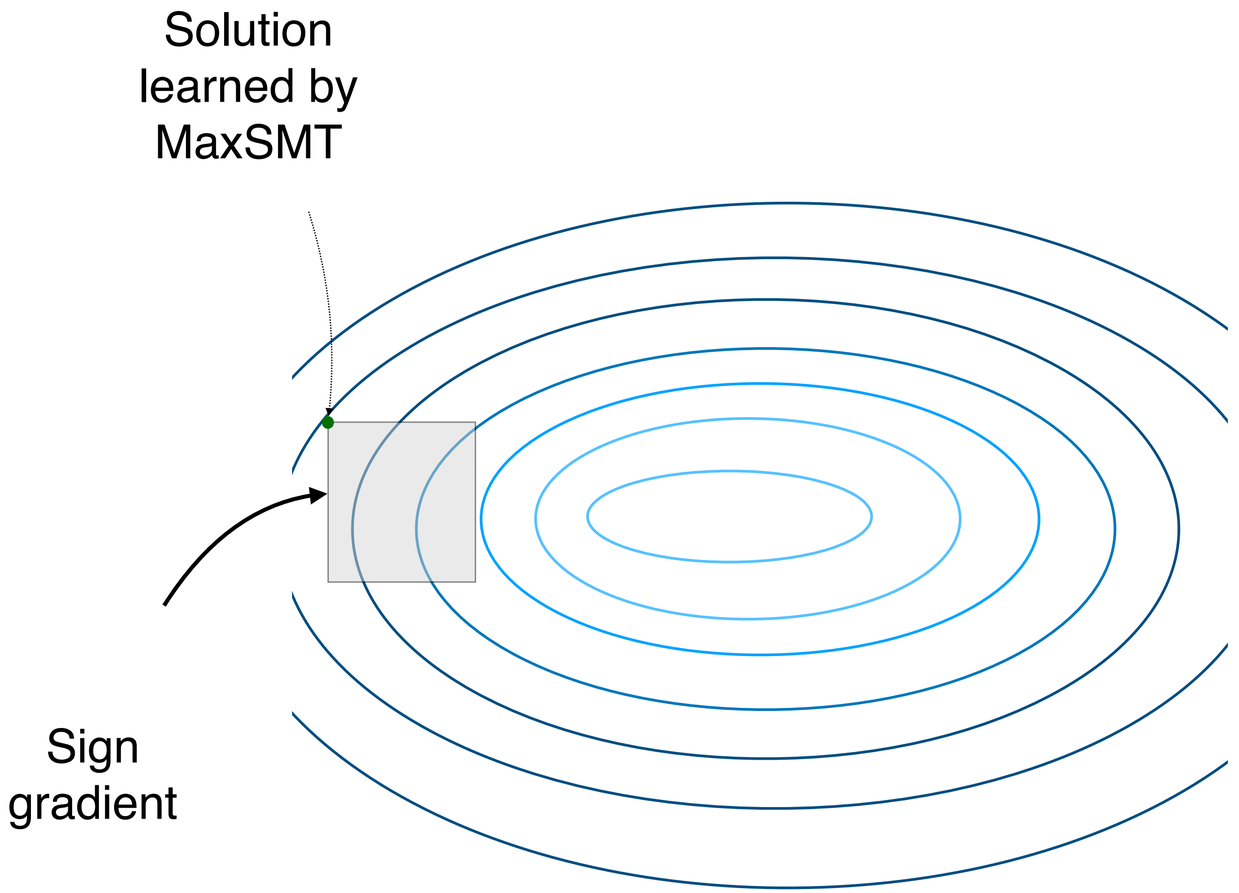
Satisfiability descent

Gradient Descent
+
Maximum Satisfiability



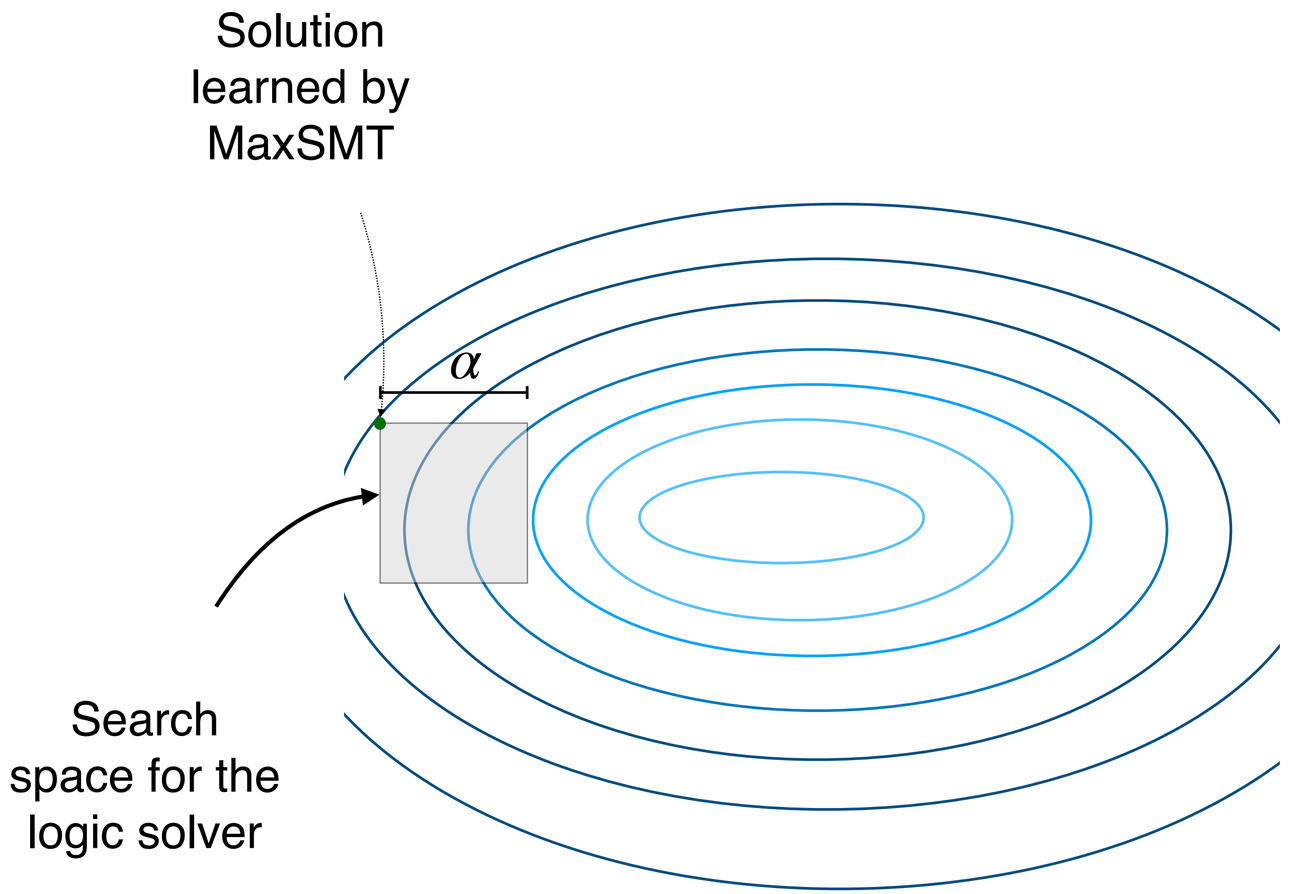
Satisfiability descent

Gradient Descent
+
Maximum Satisfiability



Satisfiability descent

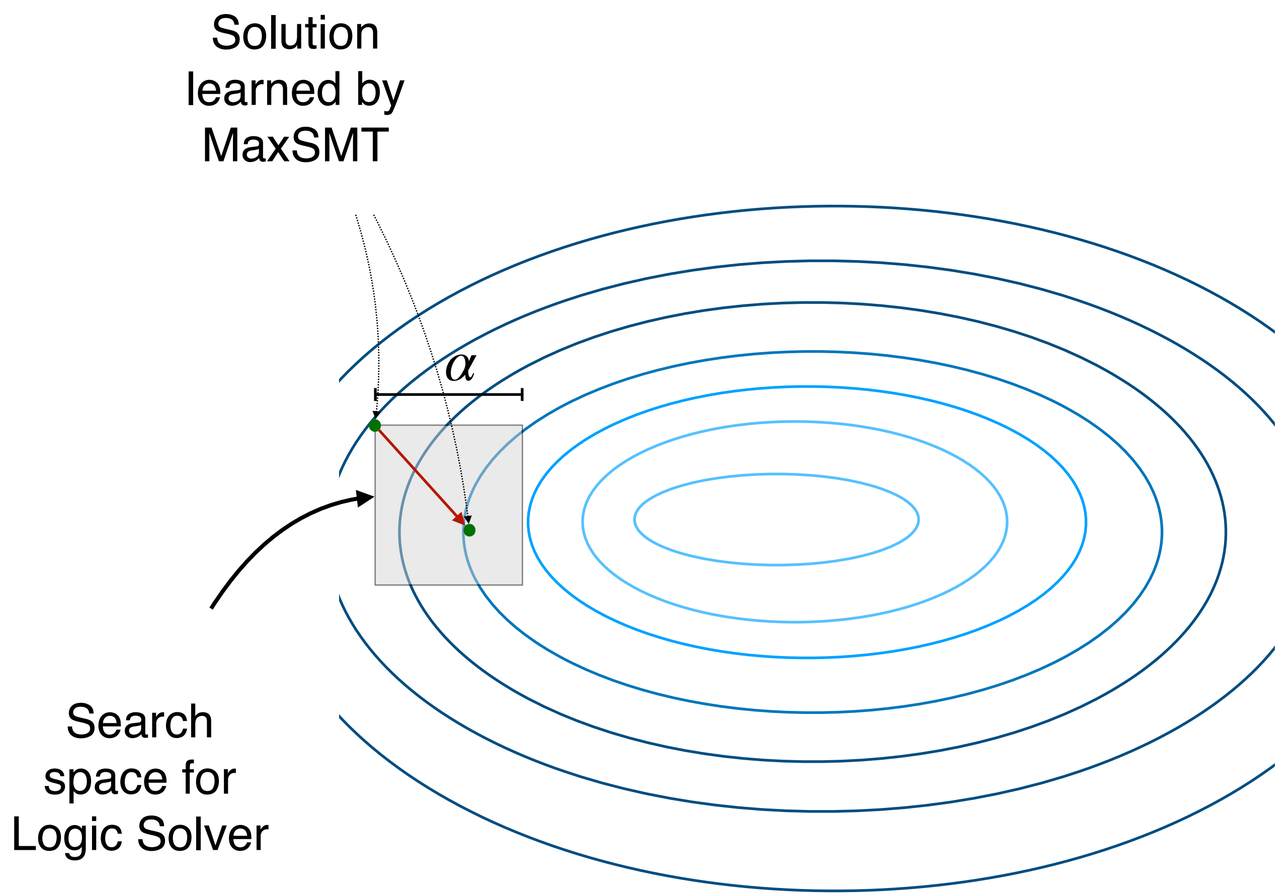
Gradient Descent
+
Maximum Satisfiability



Satisfiability descent

Gradient Descent
+
Maximum Satisfiability

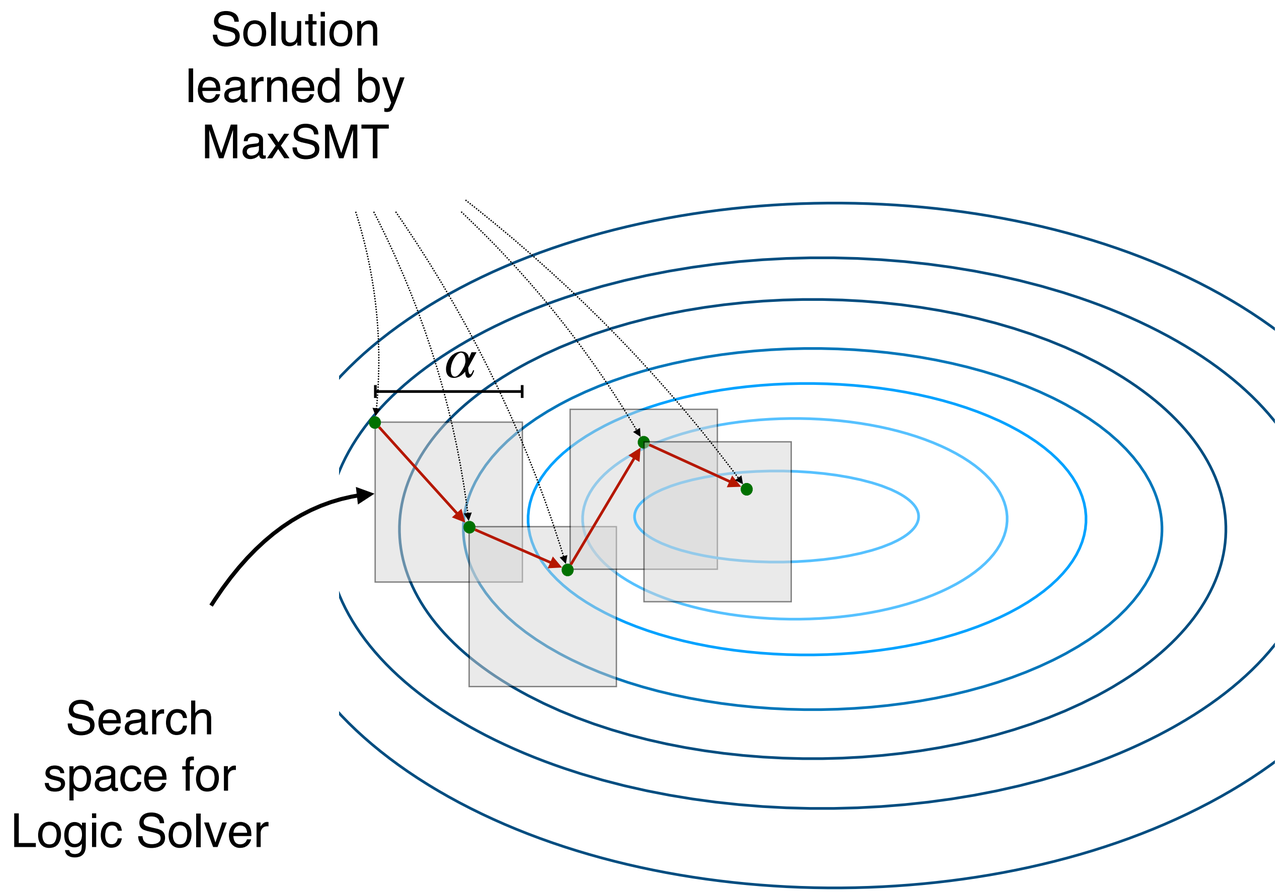
- Keeps the MaxSMT problem small
- Constraint is always satisfied by the learned solution.



Satisfiability descent

Gradient Descent
+
Maximum Satisfiability

- Keeps the MaxSMT problem small
- Constraint is always satisfied by the learned solution.



SaDe works as long as we don’t have too many features

SaDe works as long as we don’t have too many features

(Perhaps surprisingly, works well for $k \times 100$ without universal constraints)

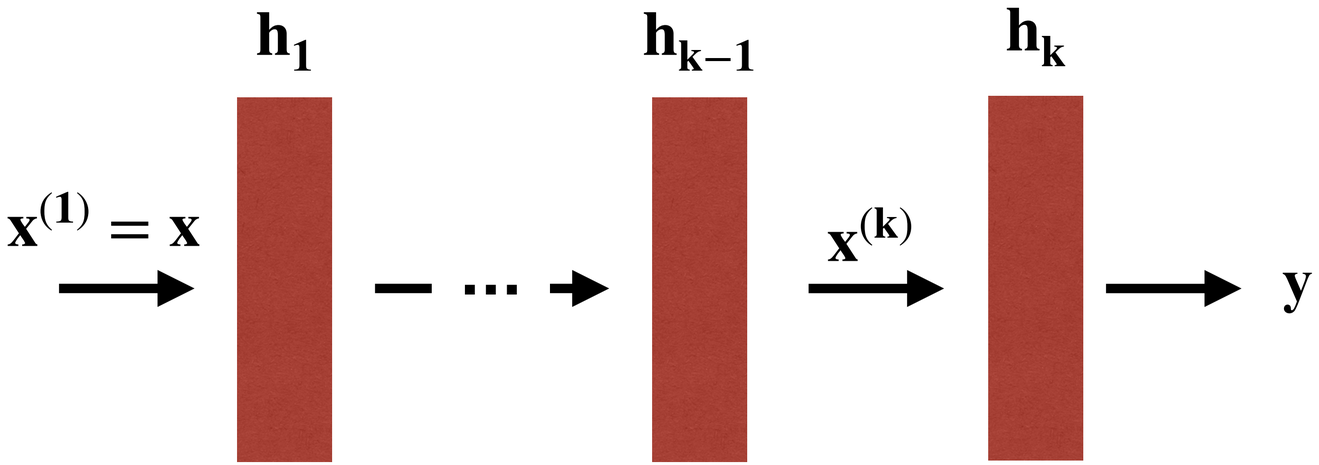
SaDe works as long as we don’t have too many features

(Perhaps surprisingly, works well for $k \times 100$ without universal constraints)

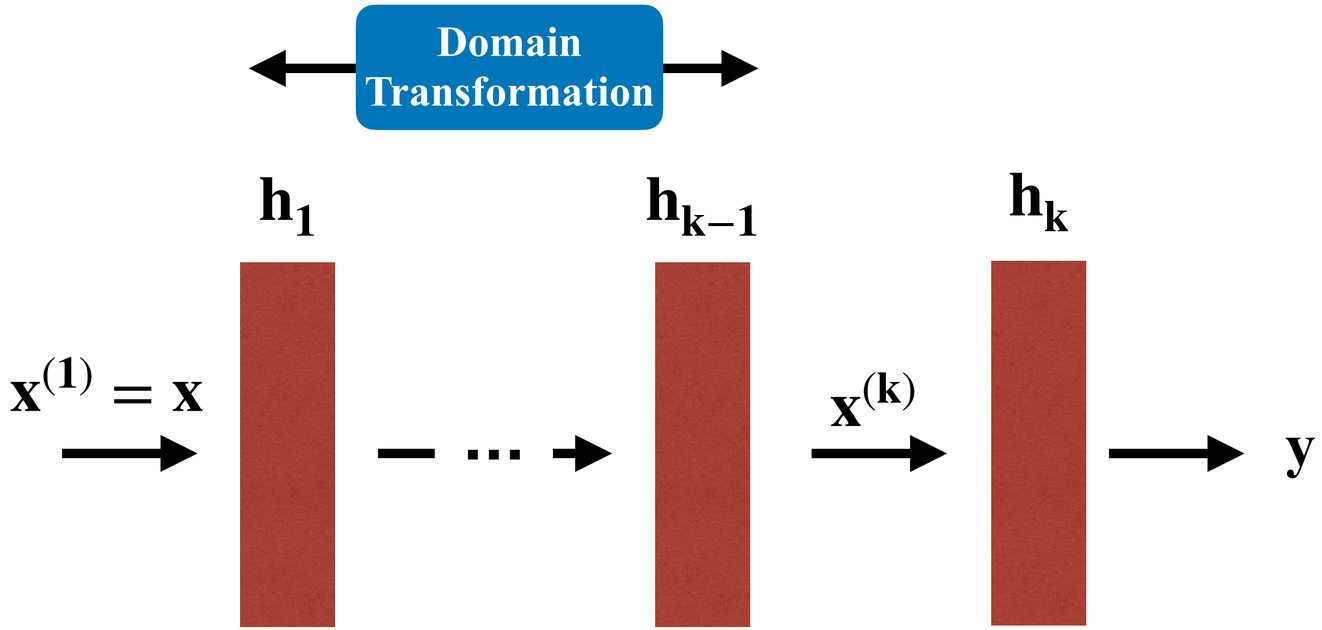
How do we get it to work on much bigger models?

DeepSaDe:
SaDe for deep neural networks

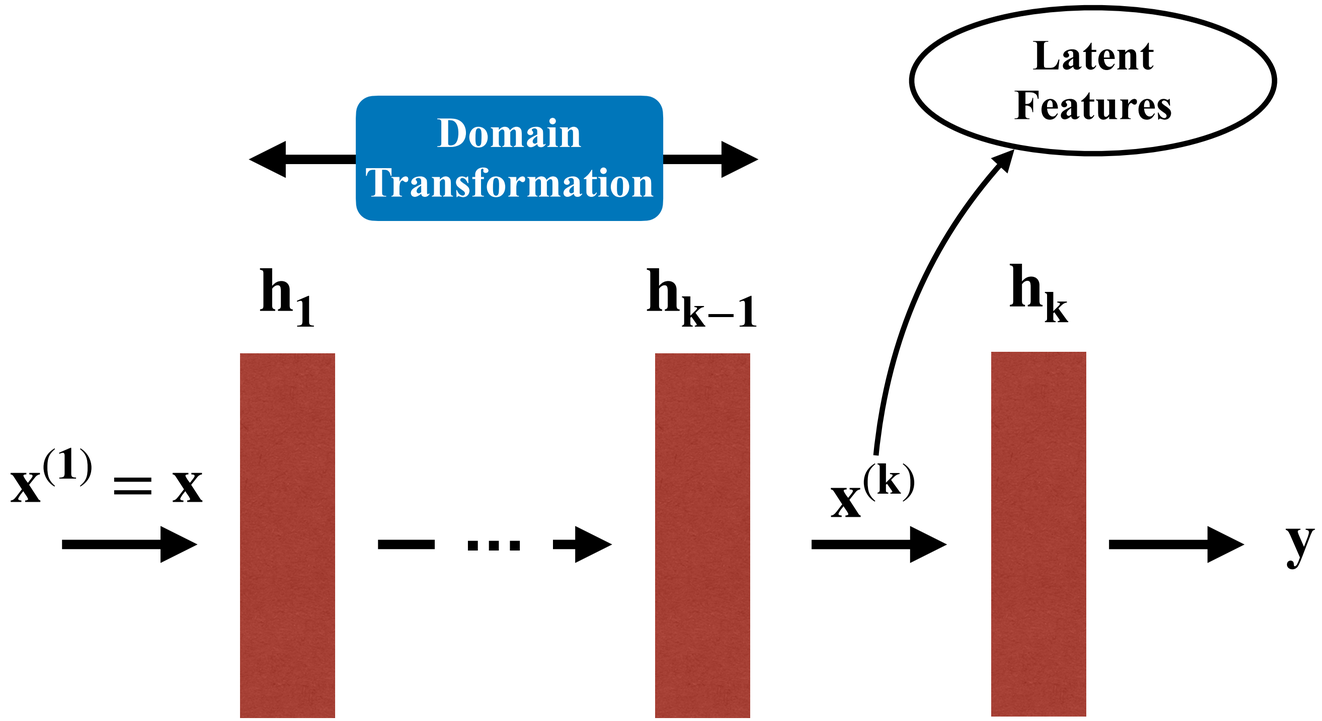
Deep satisfiability descent



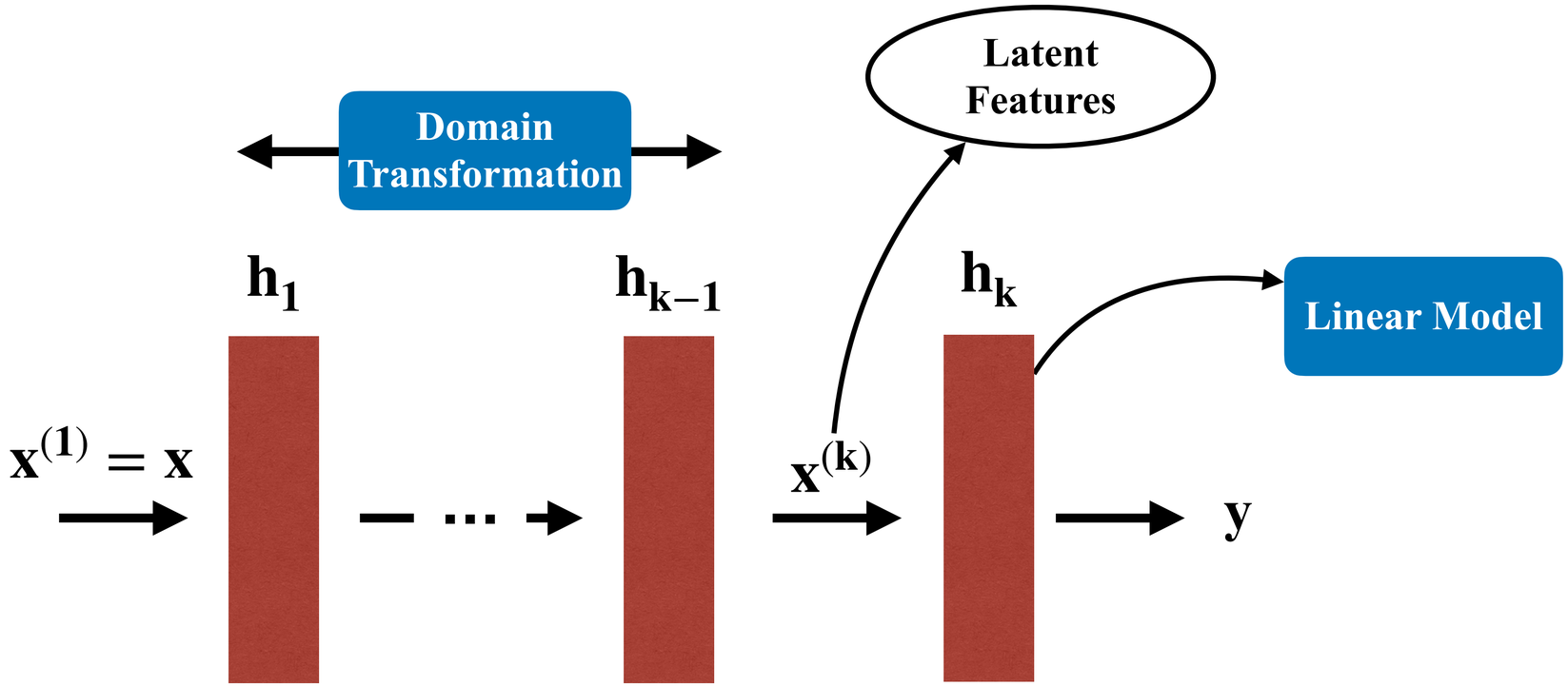
Deep satisfiability descent



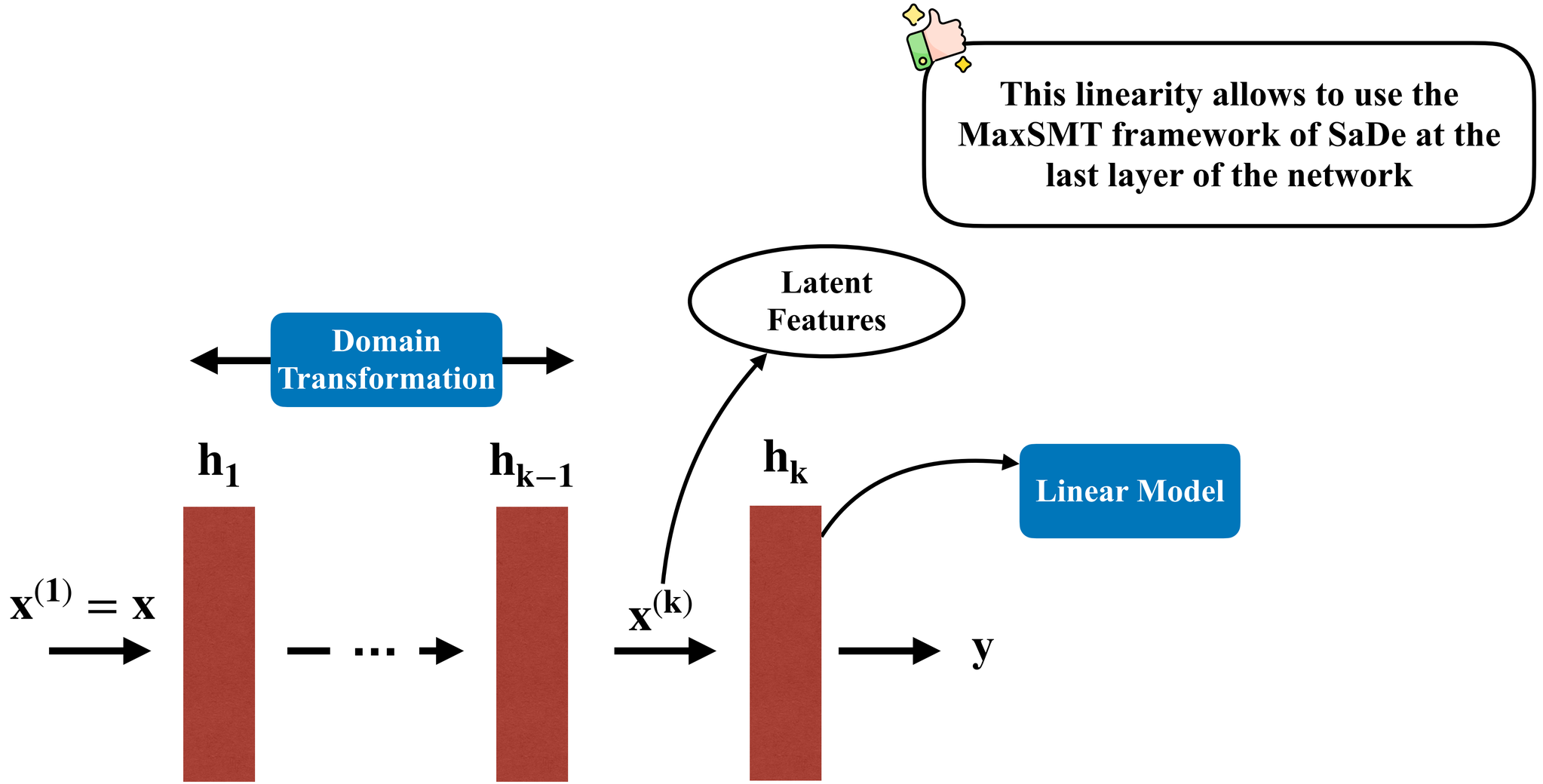
Deep satisfiability descent



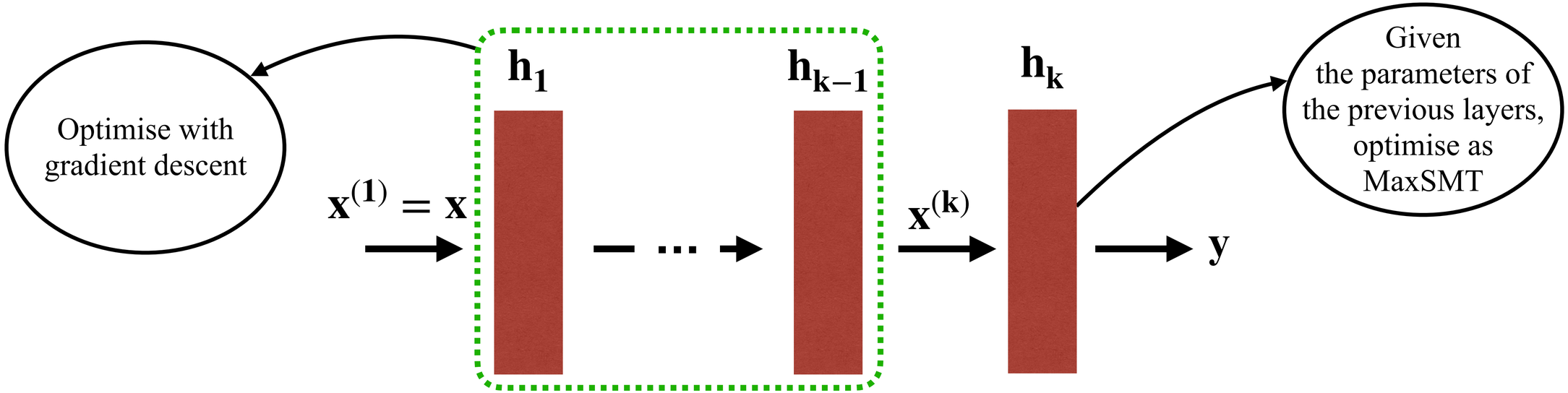
Deep satisfiability descent



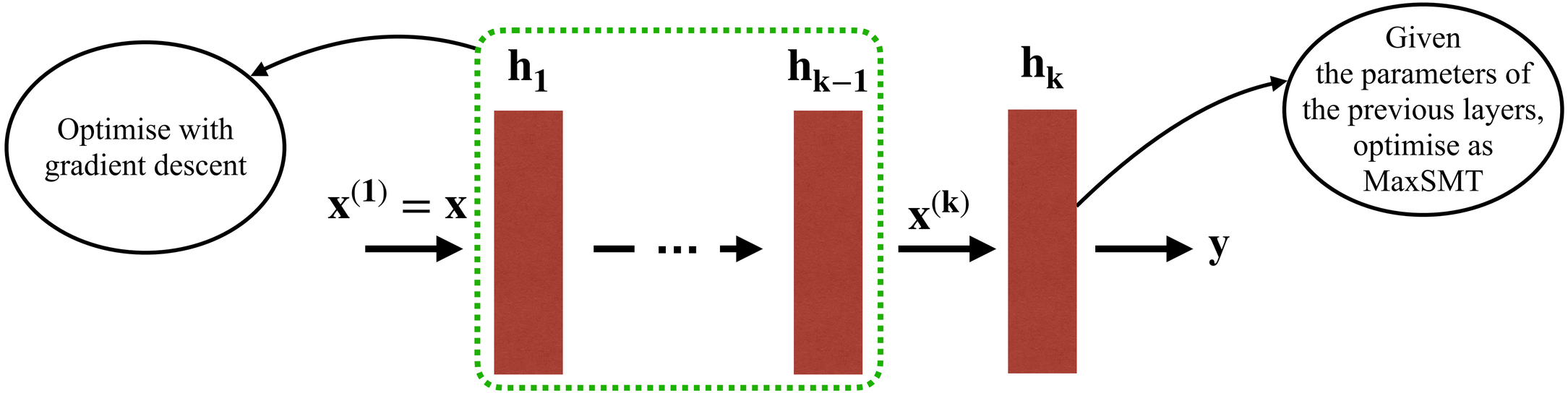
Deep satisfiability descent



Deep satisfiability descent



Deep satisfiability descent



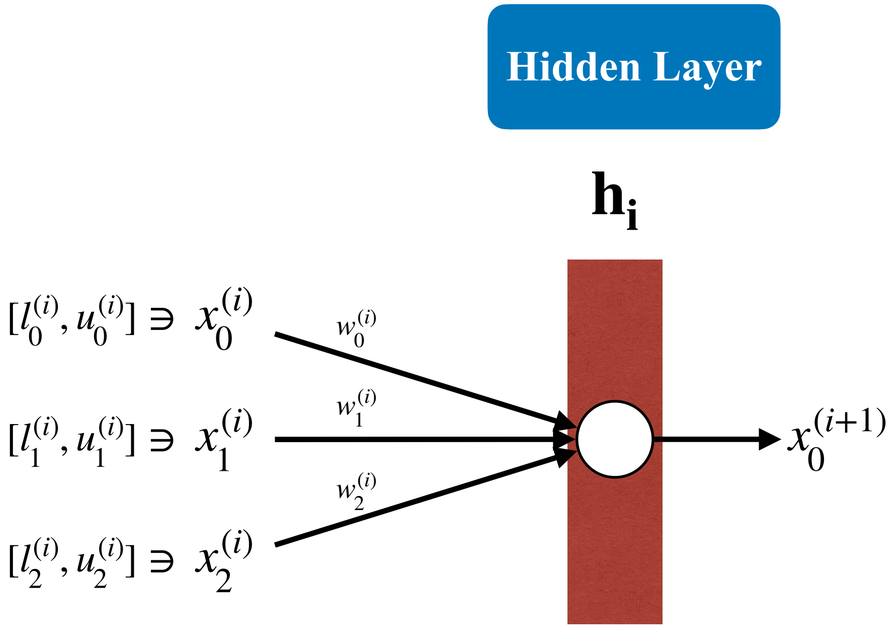
Simply rewrite soft constraints:

$$\begin{aligned} f_{\mathbf{w}}([1,4]) &< 0 \\ f_{\mathbf{w}}([2,3]) &< 0 \\ f_{\mathbf{w}}([-3,2]) &> 0 \end{aligned}$$

$$\Rightarrow$$

$$\begin{aligned} h_k(h_{k-1}\,..([1,4])\,..) &< 0 \\ h_k(h_{k-1}\,..([2,3])\,..) &< 0 \\ h_k(h_{k-1}\,..([-3,2])\,..) &> 0 \end{aligned}$$

Computing domain translation

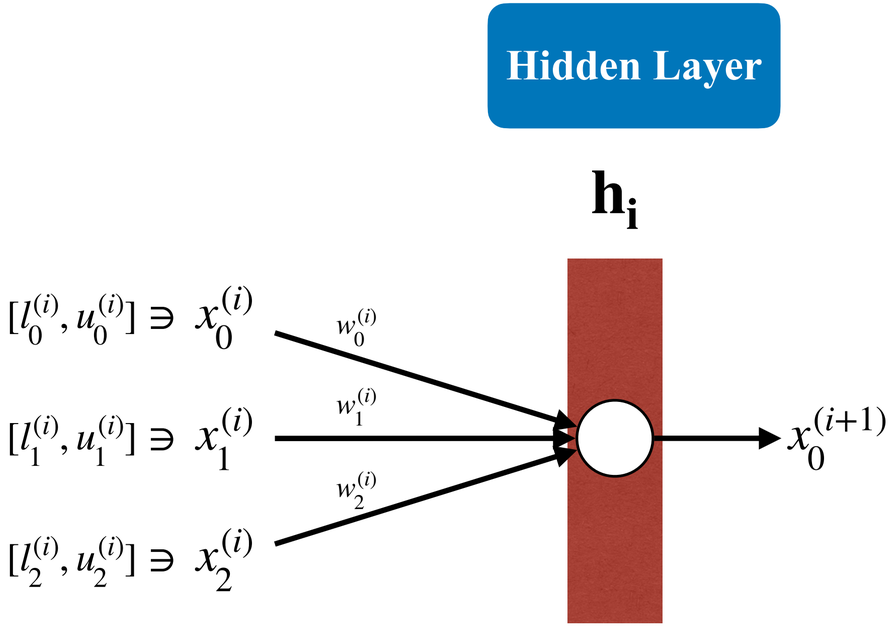


$$x_0^{(i+1)} = \mathbf{ReLU}(b_0^{(i)} + w_0^{(i)}x_0^{(i)} + w_1^{(i)}x_1^{(i)} + w_2^{(i)}x_2^{(i)})$$

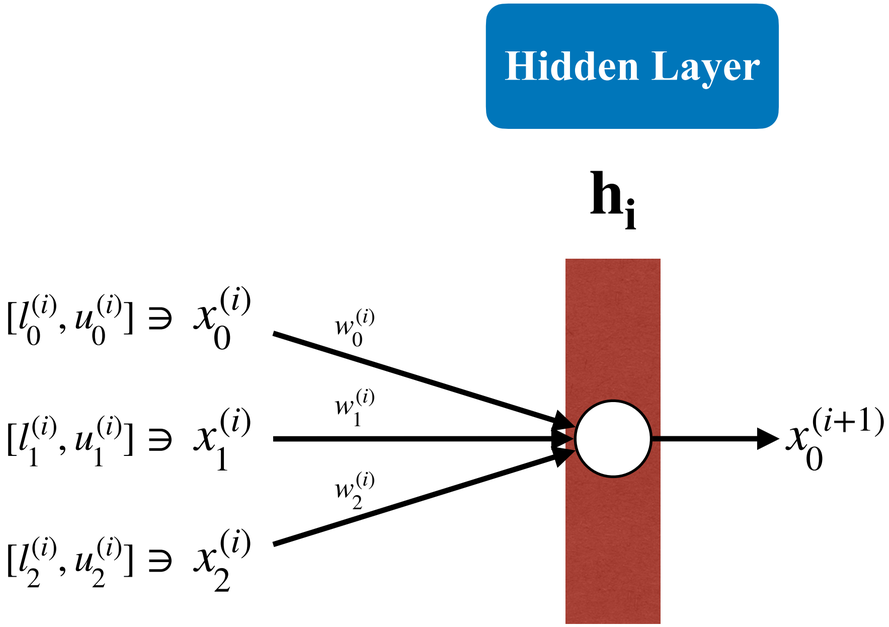
Computing domain translation

$$x_0^{(i+1)} = \mathbf{ReLU}(b_0^{(i)} + w_0^{(i)}x_0^{(i)} + w_1^{(i)}x_1^{(i)} + w_2^{(i)}x_2^{(i)})$$

Monotonically non-decreasing



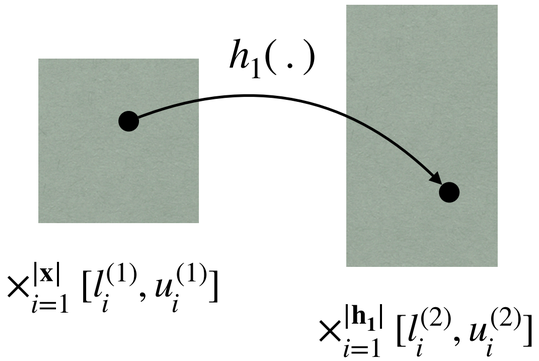
Computing domain translation



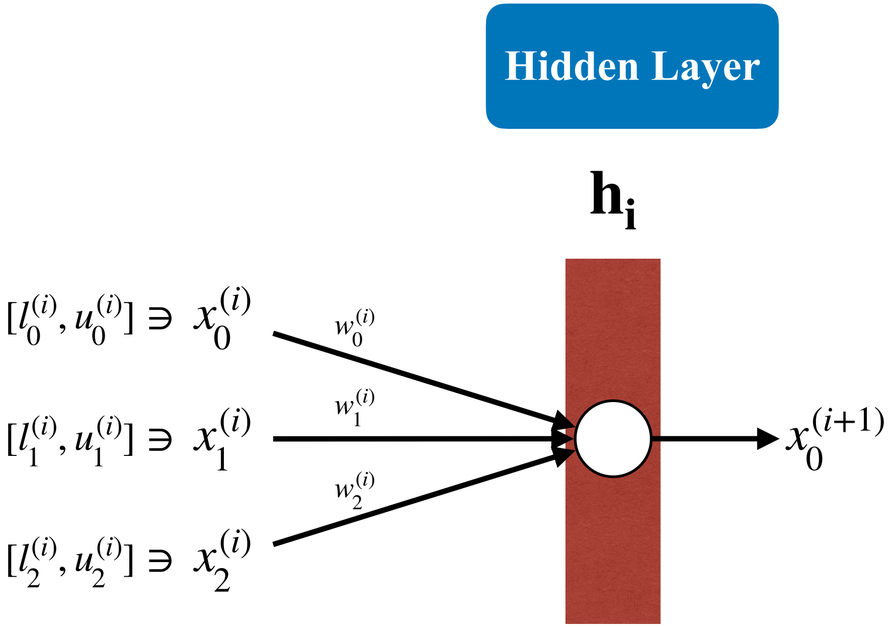
$$x_0^{(i+1)} = \mathbf{ReLU}(b_0^{(i)} + w_0^{(i)}x_0^{(i)} + w_1^{(i)}x_1^{(i)} + w_2^{(i)}x_2^{(i)})$$

$$l_0^{(i+1)} = \mathbf{ReLU}(\mathbf{min}(b_0^{(i)} + w_0^{(i)}x_0^{(i)} + w_1^{(i)}x_1^{(i)} + w_2^{(i)}x_2^{(i)}))$$

$$u_0^{(i+1)} = \mathbf{ReLU}(\mathbf{max}(b_0^{(i)} + w_0^{(i)}x_0^{(i)} + w_1^{(i)}x_1^{(i)} + w_2^{(i)}x_2^{(i)}))$$



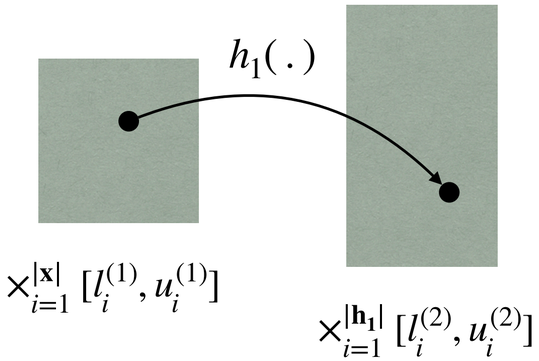
Computing domain translation



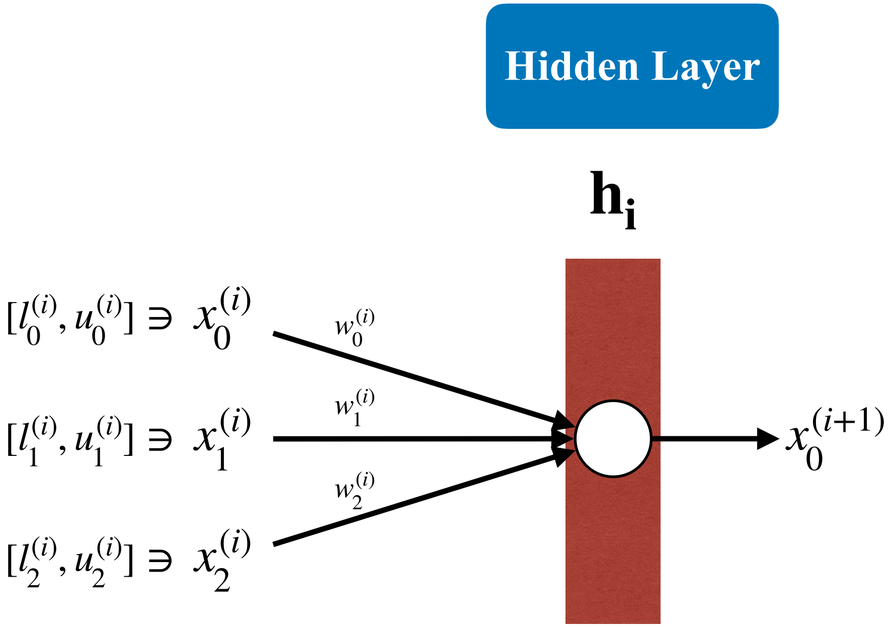
$$x_0^{(i+1)} = \mathbf{ReLU}(b_0^{(i)} + w_0^{(i)}x_0^{(i)} + w_1^{(i)}x_1^{(i)} + w_2^{(i)}x_2^{(i)})$$

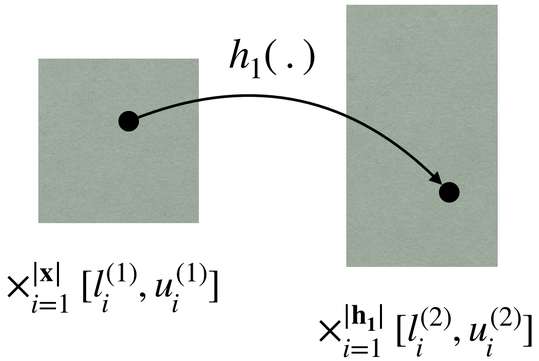
$$l_0^{(i+1)} = \mathbf{ReLU}(\mathbf{min}(b_0^{(i)} + w_0^{(i)}x_0^{(i)} + w_1^{(i)}x_1^{(i)} + w_2^{(i)}x_2^{(i)}))$$
$$u_0^{(i+1)} = \mathbf{ReLU}(\mathbf{max}(b_0^{(i)} + w_0^{(i)}x_0^{(i)} + w_1^{(i)}x_1^{(i)} + w_2^{(i)}x_2^{(i)}))$$

Each term can be optimised independently



Computing domain translation





$$x_0^{(i+1)} = \mathbf{ReLU}(b_0^{(i)} + w_0^{(i)} x_0^{(i)} + w_1^{(i)} x_1^{(i)} + w_2^{(i)} x_2^{(i)})$$

$$l_0^{(i+1)} = \mathbf{ReLU}(\mathbf{min}(b_0^{(i)} + w_0^{(i)} x_0^{(i)} + w_1^{(i)} x_1^{(i)} + w_2^{(i)} x_2^{(i)}))$$

$$u_0^{(i+1)} = \mathbf{ReLU}(\mathbf{max}(b_0^{(i)} + w_0^{(i)} x_0^{(i)} + w_1^{(i)} x_1^{(i)} + w_2^{(i)} x_2^{(i)}))$$

$$l_0^{(i+1)} = \mathbf{ReLU}(b_0^{(i)} + \sum_{j:w_j^{(i)} \geq 0} w_j^{(i)} l_j^{(i)} + \sum_{j:w_j^{(i)} < 0} w_j^{(i)} u_j^{(i)})$$

$$u_0^{(i+1)} = \mathbf{ReLU}(b_0^{(i)} + \sum_{j:w_j^{(i)} \geq 0} w_j^{(i)} u_j^{(i)} + \sum_{j:w_j^{(i)} < 0} w_j^{(i)} l_j^{(i)})$$

Experiments

1

Task: Multi-target regression to predict 5 different household expenses based on certain features

Constraint:

Sum of the predicted expenses must be smaller than “income” & going out expense must be smaller than 5% of the “income”

2

Task: Binary classification to predict if a person should be given the loan by a bank

Constraint:

People with absent “credit history” and low “income” must be denied the loan

3

Task: Multi-class classification to predict the genre of a song

Constraint:

A song by “the beatles” must only be predicted as either pop or rock

Experiments

4

Task: Multi label classification to identify labels from a sequence of 4 MNIST images

 ⟶ **[1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0]**

Constraint:

Sum of the predicted labels must be greater than 10

5

Task: Preference learning task to predict preference order of sushi.

Sake, Unagi, Ika ⟶ **[1, 0, 0, 0, 0, 1, 0, 1, 0]**

Constraint:

The preference order must be coherent [Xu et al.]

How do we evaluate these models?

Accuracy?

How do we evaluate these models?

Accuracy?

What if your historical data violates your constraints?

How do we evaluate these models?

Accuracy?

What if your historical data violates your constraints?

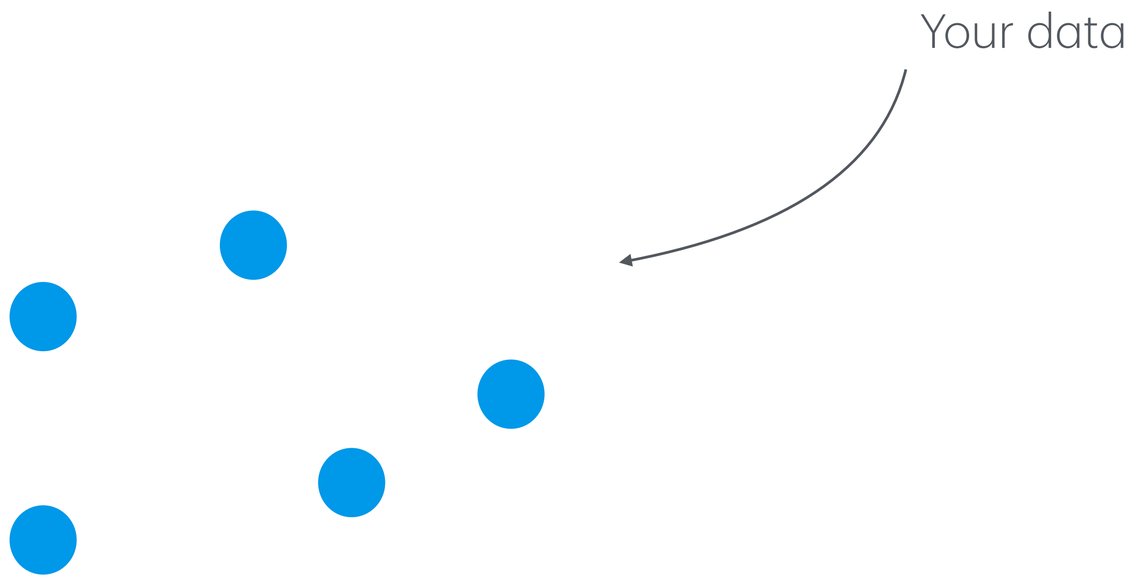
Adversity index

How do we evaluate these models?

Accuracy?

What if your historical data violates your constraints?

Adversity index

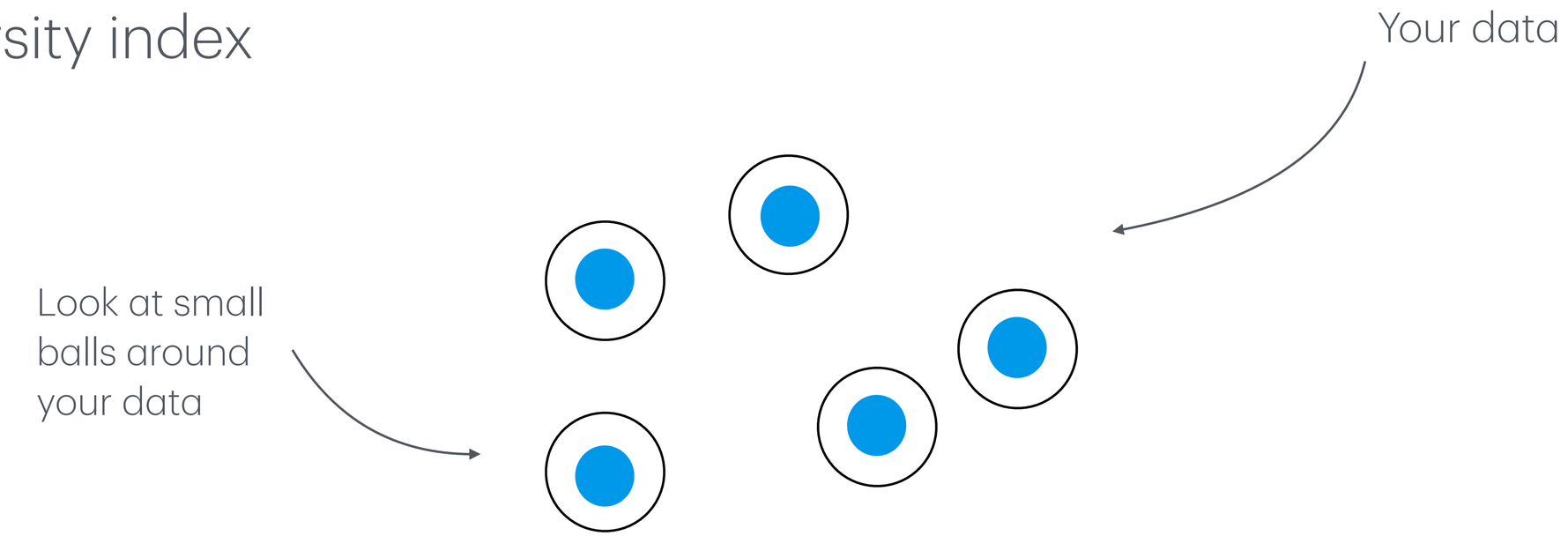


How do we evaluate these models?

Accuracy?

What if your historical data violates your constraints?

Adversity index



How do we evaluate these models?

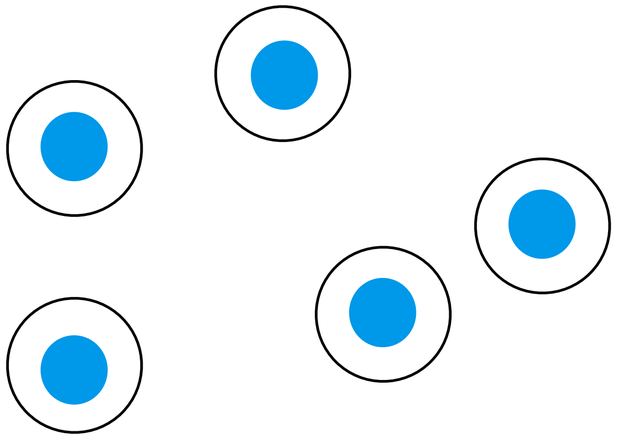
Accuracy?

What if your historical data violates your constraints?

Adversity index

Your data

Look at small balls around your data



In how many of these balls can I find a counterexample?

Experiments: Results

UC	Approach	Constraint	AdI($\delta = 0.1$)	Accuracy/MSE	Runtime (sec)
UC1	DeepSaDe	$\mathbf{100_{\pm 0}}$	$\mathbf{0_{\pm 0}}$	$^{*}38.36_{\pm 4.59}$	$102341_{\pm 40198}$
	REG	$93.50_{\pm 2.01}$	$0.97_{\pm 0.003}$	$^{*}\mathbf{30.50_{\pm 6.49}}$	$227_{\pm 70}$
UC2	DeepSaDe	$\mathbf{100_{\pm 0}}$	$\mathbf{0_{\pm 0}}$	$80.04_{\pm 4.29}$	$447_{\pm 105}$
	SL	$\mathbf{100_{\pm 0}}$	$0.002_{\pm 0.006}$	$\mathbf{80.17_{\pm 3.88}}$	$45_{\pm 30}$
	SBR	$\mathbf{100_{\pm 0}}$	$0.002_{\pm 0.004}$	$80.04_{\pm 3.95}$	$45_{\pm 27}$
UC3	DeepSaDe	$\mathbf{100_{\pm 0}}$	$\mathbf{0_{\pm 0}}$	$80.11_{\pm 4.99}$	$6580_{\pm 1915}$
	SL	$99.97_{\pm 0.03}$	$0.42_{\pm 0.28}$	$\mathbf{82.53_{\pm 3.58}}$	$101_{\pm 45}$
	SBR	$99.97_{\pm 0.05}$	$0.29_{\pm 0.10}$	$76.51_{\pm 7.50}$	$196_{\pm 68}$

Experiments: Results

UC	Approach	Constraint	AdI($\delta = 0.1$)	Accuracy/MSE	Runtime (sec)
UC1	DeepSaDe	$\mathbf{100_{\pm 0}}$	$\mathbf{0_{\pm 0}}$	$*38.36_{\pm 4.59}$	$102341_{\pm 40198}$
	REG	$93.50_{\pm 2.01}$	$0.97_{\pm 0.003}$	$\mathbf{*30.50_{\pm 6.49}}$	$227_{\pm 70}$
UC2	DeepSaDe	$\mathbf{100_{\pm 0}}$	$\mathbf{0_{\pm 0}}$	$80.04_{\pm 4.29}$	$447_{\pm 105}$
	SL	$\mathbf{100_{\pm 0}}$	$0.002_{\pm 0.006}$	$\mathbf{80.17_{\pm 3.88}}$	$45_{\pm 30}$
	SBR	$\mathbf{100_{\pm 0}}$	$0.002_{\pm 0.004}$	$80.04_{\pm 3.95}$	$45_{\pm 27}$
UC3	DeepSaDe	$\mathbf{100_{\pm 0}}$	$\mathbf{0_{\pm 0}}$	$80.11_{\pm 4.99}$	$6580_{\pm 1915}$
	SL	$99.97_{\pm 0.03}$	$0.42_{\pm 0.28}$	$\mathbf{82.53_{\pm 3.58}}$	$101_{\pm 45}$
	SBR	$99.97_{\pm 0.05}$	$0.29_{\pm 0.10}$	$76.51_{\pm 7.50}$	$196_{\pm 68}$

Existing methods do not find models that guarantee domain constraint satisfaction

Experiments: Results

DeepSaDe's hard guarantees lead to drop in predictive performance in some cases compared to baselines

UC	Approach	Constraint	AdI($\delta = 0.1$)	Accuracy/MSE	Runtime (sec)
UC1	DeepSaDe	$\mathbf{100}_{\pm 0}$	$\mathbf{0}_{\pm 0}$	$*38.36_{\pm 4.59}$	$102341_{\pm 40198}$
	REG	$93.50_{\pm 2.01}$	$0.97_{\pm 0.003}$	$\mathbf{*30.50}_{\pm \mathbf{6.49}}$	$227_{\pm 70}$
UC2	DeepSaDe	$\mathbf{100}_{\pm 0}$	$\mathbf{0}_{\pm 0}$	$80.04_{\pm 4.29}$	$447_{\pm 105}$
	SL	$\mathbf{100}_{\pm 0}$	$0.002_{\pm 0.006}$	$\mathbf{80.17}_{\pm \mathbf{3.88}}$	$45_{\pm 30}$
	SBR	$\mathbf{100}_{\pm 0}$	$0.002_{\pm 0.004}$	$80.04_{\pm 3.95}$	$45_{\pm 27}$
UC3	DeepSaDe	$\mathbf{100}_{\pm 0}$	$\mathbf{0}_{\pm 0}$	$80.11_{\pm 4.99}$	$6580_{\pm 1915}$
	SL	$99.97_{\pm 0.03}$	$0.42_{\pm 0.28}$	$\mathbf{82.53}_{\pm \mathbf{3.58}}$	$101_{\pm 45}$
	SBR	$99.97_{\pm 0.05}$	$0.29_{\pm 0.10}$	$76.51_{\pm 7.50}$	$196_{\pm 68}$

Existing methods do not find models that guarantee domain constraint satisfaction

Experiments: Results

DeepSaDe's hard guarantees lead to drop in predictive performance in some cases compared to baselines

UC	Approach	Constraint	AdI($\delta = 0.1$)	Accuracy/MSE	Runtime (sec)
UC1	DeepSaDe	$\mathbf{100}_{\pm 0}$	$\mathbf{0}_{\pm 0}$	$*38.36_{\pm 4.59}$	$102341_{\pm 40198}$
	REG	$93.50_{\pm 2.01}$	$0.97_{\pm 0.003}$	$\mathbf{*30.50}_{\pm \mathbf{6.49}}$	$227_{\pm 70}$
UC2	DeepSaDe	$\mathbf{100}_{\pm 0}$	$\mathbf{0}_{\pm 0}$	$80.04_{\pm 4.29}$	$447_{\pm 105}$
	SL	$\mathbf{100}_{\pm 0}$	$0.002_{\pm 0.006}$	$\mathbf{80.17}_{\pm \mathbf{3.88}}$	$45_{\pm 30}$
	SBR	$\mathbf{100}_{\pm 0}$	$0.002_{\pm 0.004}$	$80.04_{\pm 3.95}$	$45_{\pm 27}$
UC3	DeepSaDe	$\mathbf{100}_{\pm 0}$	$\mathbf{0}_{\pm 0}$	$80.11_{\pm 4.99}$	$6580_{\pm 1915}$
	SL	$99.97_{\pm 0.03}$	$0.42_{\pm 0.28}$	$\mathbf{82.53}_{\pm \mathbf{3.58}}$	$101_{\pm 45}$
	SBR	$99.97_{\pm 0.05}$	$0.29_{\pm 0.10}$	$76.51_{\pm 7.50}$	$196_{\pm 68}$

Existing methods do not find models that guarantee domain constraint satisfaction

Training time of DeepSaDe is significantly higher than the existing methods

Experiments: Results

DeepSaDe's hard guarantees lead to drop in predictive performance in some cases compared to baselines

UC	Approach	Constraint	Coherent	Flattened	Jaccard	Runtime (sec)
UC4	DeepSaDe	$\mathbf{100}_{\pm 0}$	$6.62_{\pm 1.72}$	$78.52_{\pm 1.73}$	$62.97_{\pm 2.28}$	$227928_{\pm 30559}$
	FFN	$88.00_{\pm 3.26}$	$\mathbf{23.94}_{\pm \mathbf{4.25}}$	$\mathbf{85.81}_{\pm \mathbf{1.18}}$	$\mathbf{71.55}_{\pm \mathbf{2.40}}$	$3215_{\pm 2663}$
UC5	DeepSaDe	$\mathbf{100}_{\pm 0}$	$\mathbf{11.08}_{\pm \mathbf{2.61}}$	$67.17_{\pm 1.48}$	$\mathbf{25.94}_{\pm \mathbf{2.89}}$	$17586_{\pm 5074}$
	FFN	$0.04_{\pm 0.15}$	$0.01_{\pm 0.04}$	$\mathbf{75.69}_{\pm \mathbf{0.15}}$	$13.04_{\pm 1.06}$	$48_{\pm 10}$
	SL	$\mathbf{100}_{\pm 0}$	$4.06_{\pm 3.33}$	$63.16_{\pm 2.62}$	$18.08_{\pm 3.33}$	$298_{\pm 110}$

Existing methods have even worse constraint satisfaction for complex constraints

Training time of DeepSaDe is significantly higher than the existing methods

Summary

We still don’t know how to learn models that provably obey constraints

Training models with guarantees requires new learning techniques

SaDe combines maximum satisfiability with gradient descent for performance and guarantees

To cover big models, you only need to consider “the end” of the model