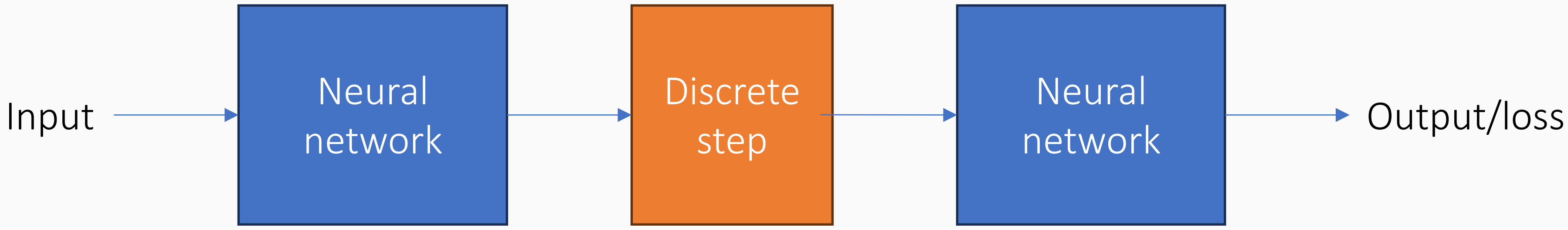
Learning with symbols within neural networks: Straight-through estimators

Neuro-symbolic modeling

THE UNIVERSITY OF UTAH

Neural networks containing discrete elements



Let's see some examples

This lecture

- Motivating examples
- The straight-through estimator
- The Gumbel trick
- REINFORCE

(others if time permits)

Not all these approaches are always applicable

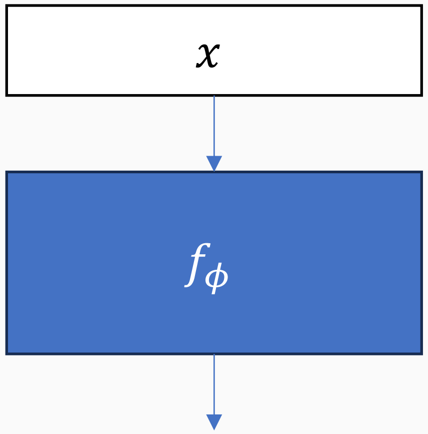
Binary values in neural networks?

Let us consider a simple neural network
consisting of two sets of parameters ϕ and θ

Binary values in neural networks?

Let us consider a simple neural network consisting of two sets of parameters ϕ and θ

Given an example x, it computes $f_\phi(x)$ to produce a set of d scores

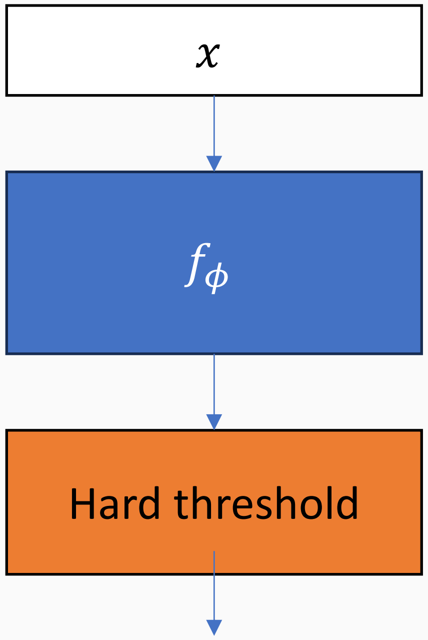


Binary values in neural networks?

Let us consider a simple neural network consisting of two sets of parameters ϕ and θ

Given an example x, it computes $f_\phi(x)$ to produce a set of d scores

Each score is thresholded at zero to produce a d-dimensional binary vector z



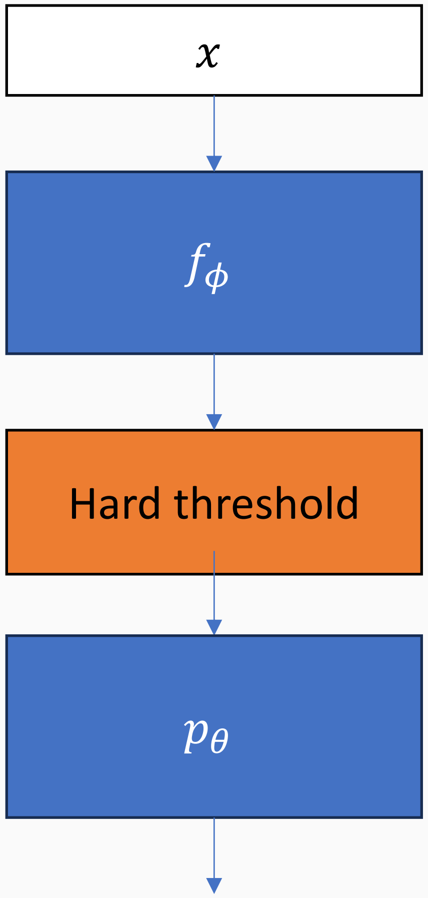
Binary values in neural networks?

Let us consider a simple neural network consisting of two sets of parameters ϕ and θ

Given an example x, it computes $f_\phi(x)$ to produce a set of d scores

Each score is thresholded at zero to produce a d-dimensional binary vector z

The final output is then $g_\theta(z)$



Let us write this formally

The prediction is

$$y = g_\theta\left(\text{Threshold}\left(f_\phi(x)\right)\right)$$

Let us write this formally

The prediction is

$$y = g_\theta\left(\text{Threshold}\left(f_\phi(x)\right)\right)$$

We can define a loss over this prediction, say $\mathrm{L}(y, \mathrm{y}^*)$

Let us write this formally

The prediction is

$$y = g_\theta\left(\text{Threshold}\left(f_\phi(x)\right)\right)$$

We can define a loss over this prediction, say $\mathrm{L}(y, \mathrm{y}^*)$

For convenience, let:

$f = f_\phi(x)$

$z = \text{Threshold}(f)$

$y = g_\theta(z)$

We can try to compute gradients of the loss

The prediction is

$$y = g_\theta\left(\text{Threshold}\left(f_\phi(x)\right)\right)$$

For convenience, let:

$f = f_\phi(x)$

$z = \text{Threshold}(f)$

$y = g_\theta(z)$

Let us write the derivative of this loss with respect to the parameters:

We can try to compute gradients of the loss

The prediction is

$$y = g_\theta\left(\text{Threshold}\left(f_\phi(x)\right)\right)$$

For convenience, let:
$f = f_\phi(x)$
$z = \text{Threshold}(f)$
$y = g_\theta(z)$

Let us write the derivative of this loss with respect to the parameters:

$$\nabla_\theta \mathrm{L} = \frac{\partial L}{\partial y} \cdot \frac{\partial y}{\partial \theta}$$

This part is standard: L and y are differentiable functions of the parameters θ

We can try to compute gradients of the loss

The prediction is

$$y = g_\theta\left(\text{Threshold}\left(f_\phi(x)\right)\right)$$

For convenience, let:

$f = f_\phi(x)$

$z = \text{Threshold}(f)$

$y = g_\theta(z)$

Let us write the derivative of this loss with respect to the parameters:

$$\nabla_\theta \mathrm{L} = \frac{\partial L}{\partial y} \cdot \frac{\partial y}{\partial \theta} \qquad \nabla_\phi \mathrm{L} = \frac{\partial L}{\partial y} \cdot \frac{\partial y}{\partial z} \cdot \frac{\partial z}{\partial f} \cdot \frac{\partial f}{\partial \phi}$$

We can try to compute gradients of the loss

The prediction is

$$y = g_\theta\left(\text{Threshold}\left(f_\phi(x)\right)\right)$$

For convenience, let:

$f = f_\phi(x)$

$z = \text{Threshold}(f)$

$y = g_\theta(z)$

Let us write the derivative of this loss with respect to the parameters:

$$\nabla_\theta \mathrm{L} = \frac{\partial L}{\partial y} \cdot \frac{\partial y}{\partial \theta}$$

$$\nabla_\phi \mathrm{L} = \frac{\partial L}{\partial y} \cdot \frac{\partial y}{\partial z} \cdot \frac{\partial z}{\partial f} \cdot \frac{\partial f}{\partial \phi}$$

This is not useful because $\frac{\partial z}{\partial f}$ is zero almost everywhere and infinite at $f = 0$

The straight through estimator

For convenience, let:

$f = f_\phi(x)$

$z = \text{Threshold}(f)$

$y = g_\theta(z)$

Let us write the derivative of this loss with respect to the parameters:

$$\nabla_\theta \mathrm{L} = \frac{\partial L}{\partial y} \cdot \frac{\partial y}{\partial \theta} \qquad \nabla_\phi \mathrm{L} = \frac{\partial L}{\partial y} \cdot \frac{\partial y}{\partial z} \cdot \cancel{\frac{\partial z}{\partial f}} \cdot \frac{\partial f}{\partial \phi}$$

For the backward pass alone, pretend that z is the result identity function

Hinton, Geoffrey. “Neural networks for machine learning”. *Coursera, lecture 15b* (2012).
Bengio, Yoshua, Nicholas Léonard, and Aaron Courville. "Estimating or propagating gradients through stochastic neurons for conditional computation." *arXiv preprint arXiv:1308.3432* (2013).

For convenience, let:
$f = f_\phi(x)$
$z = \text{Threshold}(f)$
$y = g_\theta(z)$

The straight through estimator

Let us write the derivative of this loss with respect to the parameters:

$$\nabla_\theta \mathrm{L} = \frac{\partial L}{\partial y} \cdot \frac{\partial y}{\partial \theta} \qquad \nabla_\phi \mathrm{L} = \frac{\partial L}{\partial y} \cdot \frac{\partial y}{\partial z} \cdot \frac{\partial f}{\partial \phi}$$

For the backward pass alone, pretend that z is the result identity function. So we can get rid of that partial derivative

Hinton, Geoffrey. “Neural networks for machine learning”. *Coursera, lecture 15b* (2012).
Bengio, Yoshua, Nicholas Léonard, and Aaron Courville. "Estimating or propagating gradients through stochastic neurons for conditional computation." *arXiv preprint arXiv:1308.3432* (2013).

The straight through estimator

For convenience, let:
$f = f_\phi(x)$
$z = \text{Threshold}(f)$
$y = g_\theta(z)$

Let us write the derivative of this loss with respect to the parameters:

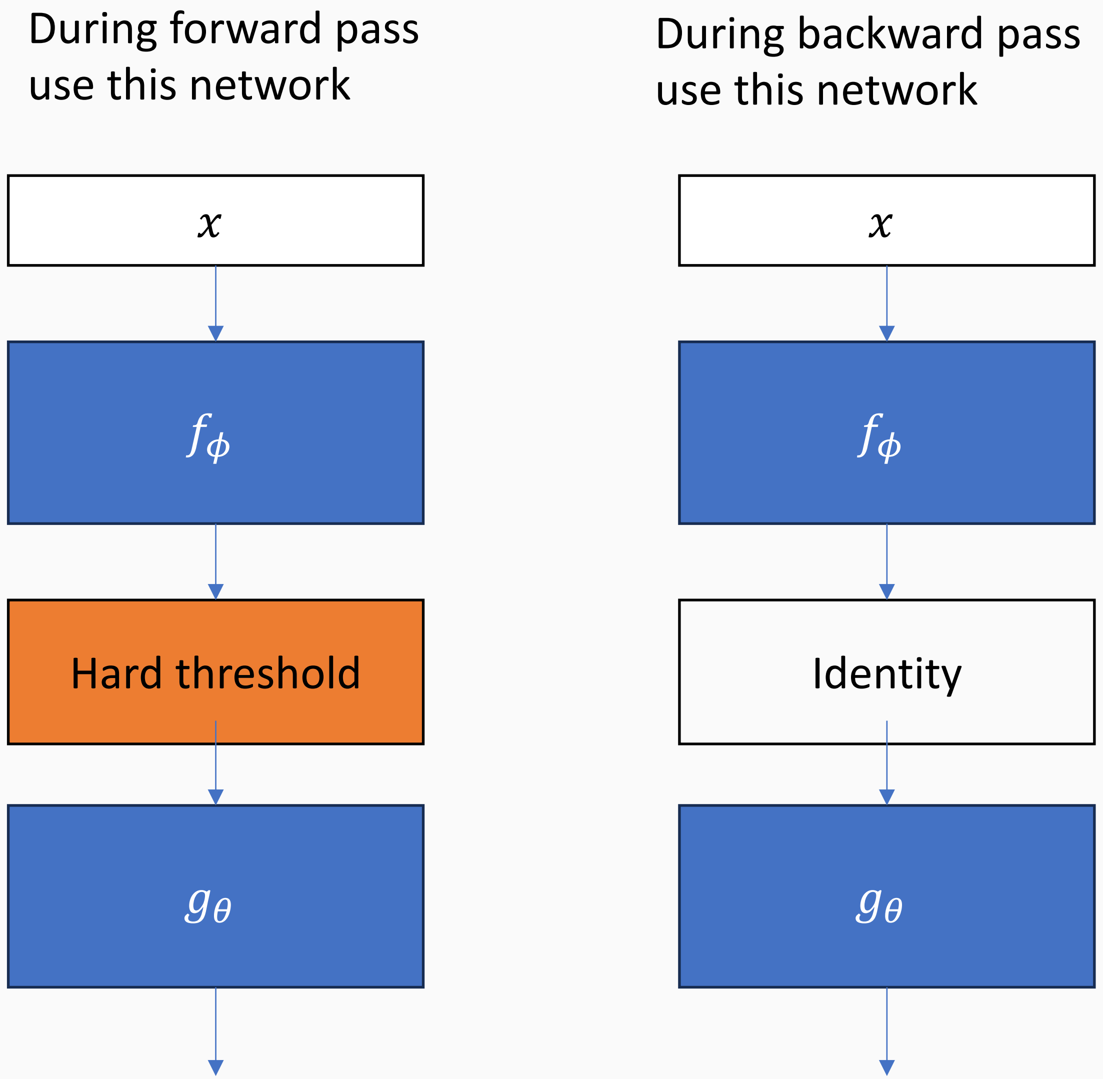
$$\nabla_\theta \mathrm{L} = \frac{\partial L}{\partial y} \cdot \frac{\partial y}{\partial \theta} \qquad \nabla_\phi \mathrm{L} = \frac{\partial L}{\partial y} \cdot \frac{\partial y}{\partial z} \cdot \frac{\partial f}{\partial \phi}$$

For the backward pass alone, pretend that z is the result identity function. So we can get rid of that partial derivative

In the forward pass, it still uses the threshold function

Hinton, Geoffrey. “Neural networks for machine learning”. *Coursera, lecture 15b* (2012).
Bengio, Yoshua, Nicholas Léonard, and Aaron Courville. "Estimating or propagating gradients through stochastic neurons for conditional computation." *arXiv preprint arXiv:1308.3432* (2013).

Straight through estimator

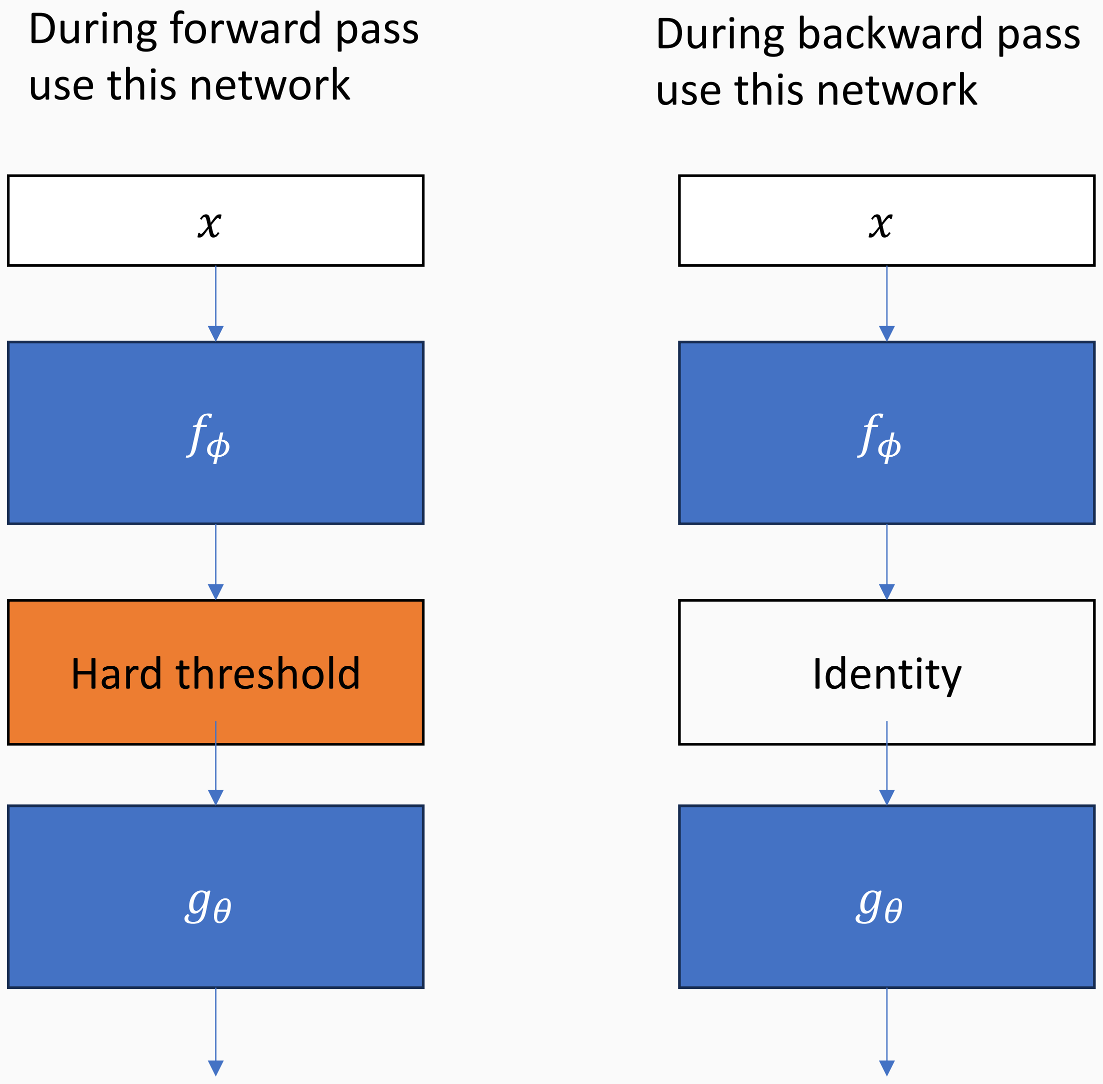


The gradient difficulties due to discreteness are ignored

This gradient estimator is poorly motivated

Yet, it sometimes works! (And easy to implement)

Straight through estimator



The gradient difficulties due to discreteness are ignored

This gradient estimator is poorly motivated

Yet, it sometimes works! (And easy to implement)

One failure case: When there are dependencies between the binary variables, these are not accounted for in the gradient