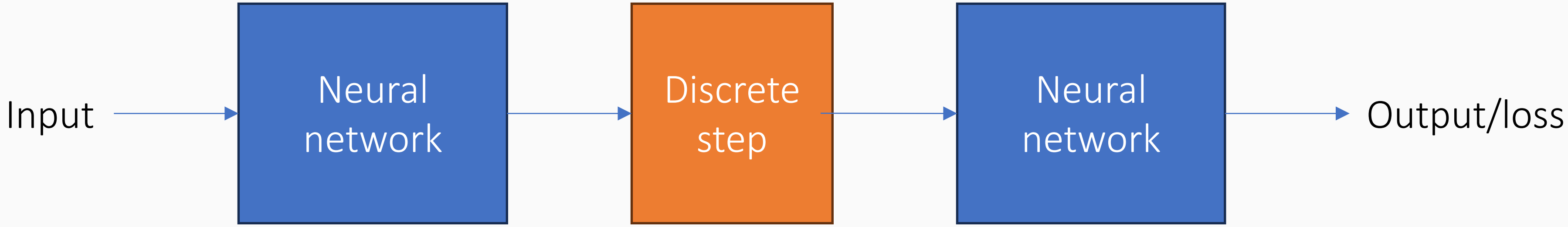
Learning with symbols within neural networks:
Gumbel-Softmax

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
- Gumbel-Softmax
- REINFORCE

(others if time permits)

Not all these approaches are always applicable

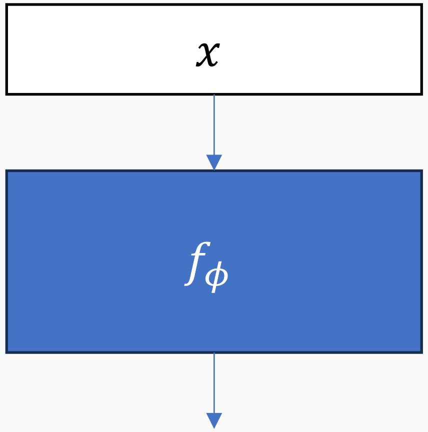
The problem: Stochastic nodes in neural networks

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

The problem: Stochastic nodes 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

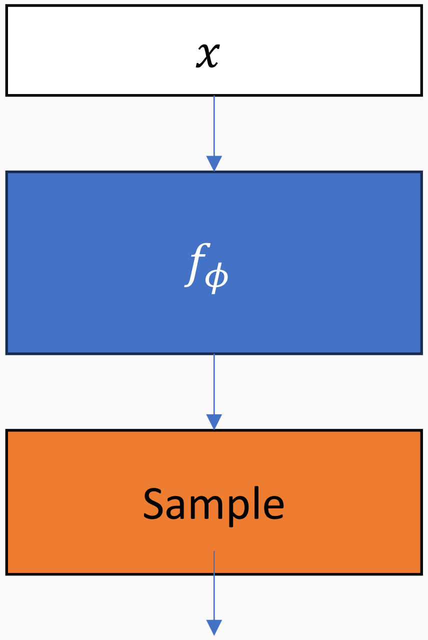


The problem: Stochastic nodes in neural networks

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

A discrete value z is *sampled* from the normalized distribution associated with these scores



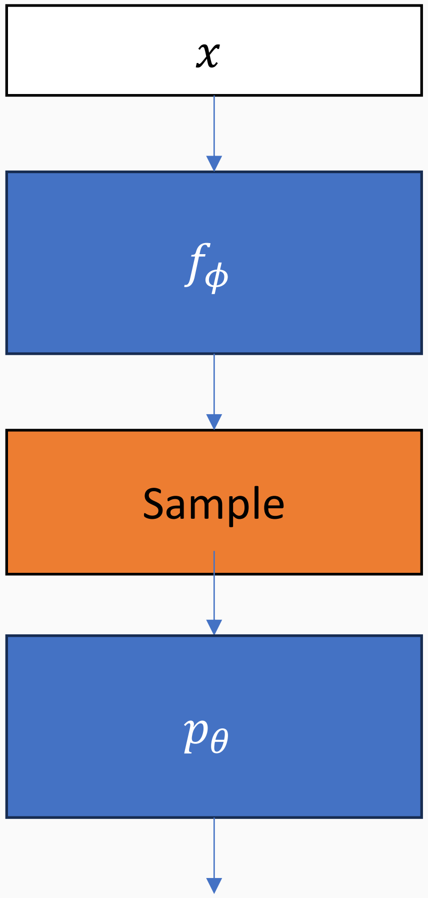
The problem: Stochastic nodes in neural networks

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

A discrete value z is sampled from the normalized distribution associated with these scores

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



The problem: Stochastic nodes in neural networks

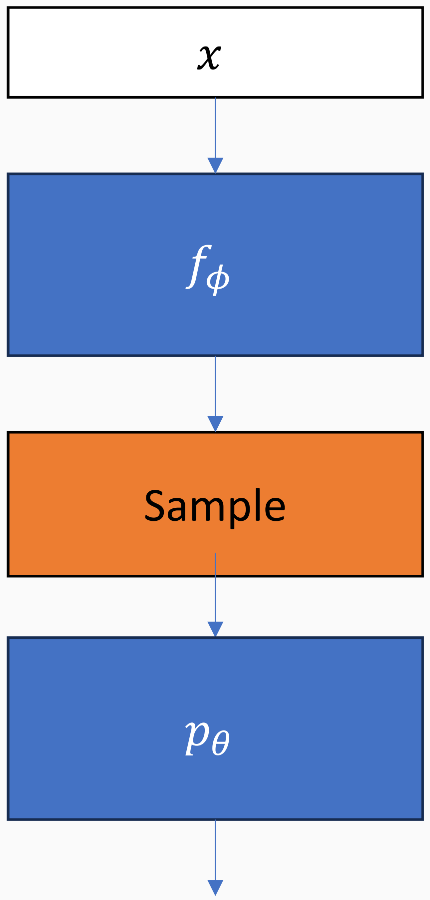
Let us consider a simple neural network consisting of two sets of parameters ϕ and θ

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

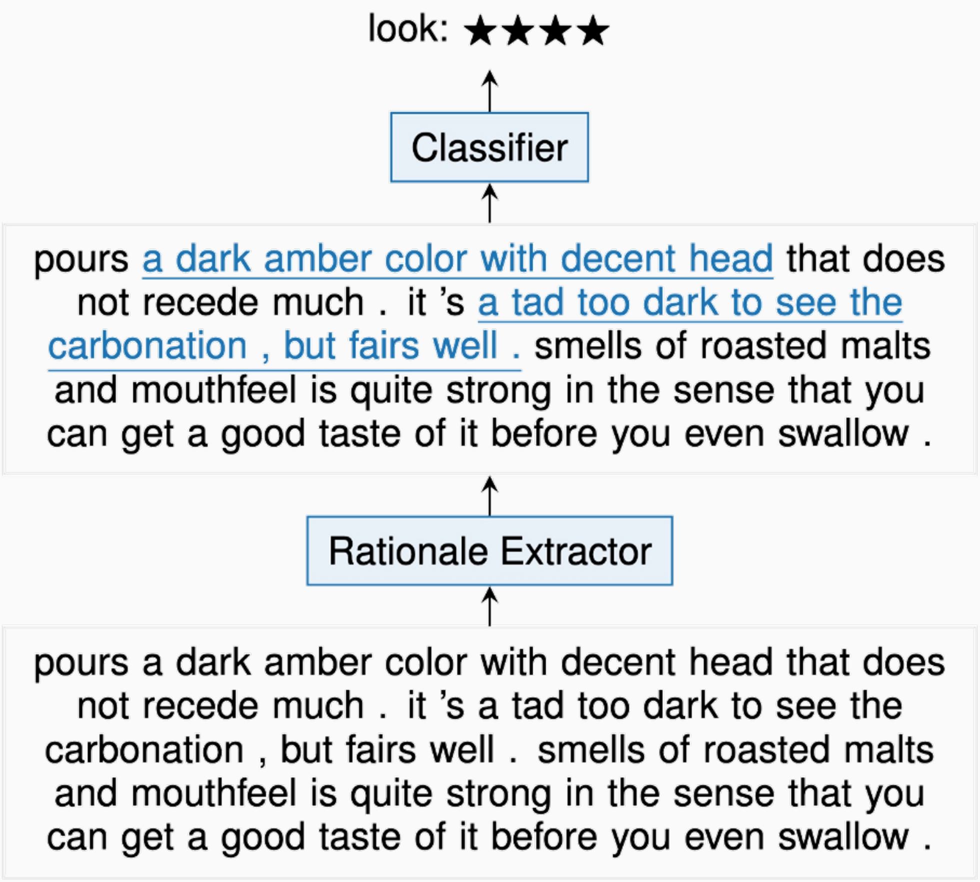
A discrete value z is sampled from the normalized distribution associated with these scores

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

By sampling z, we no longer have a differentiable link between the final output and the parameters θ



Example: Text classification and rationales



An alternative approach:

1. Identify the rationale (the highlighted words)

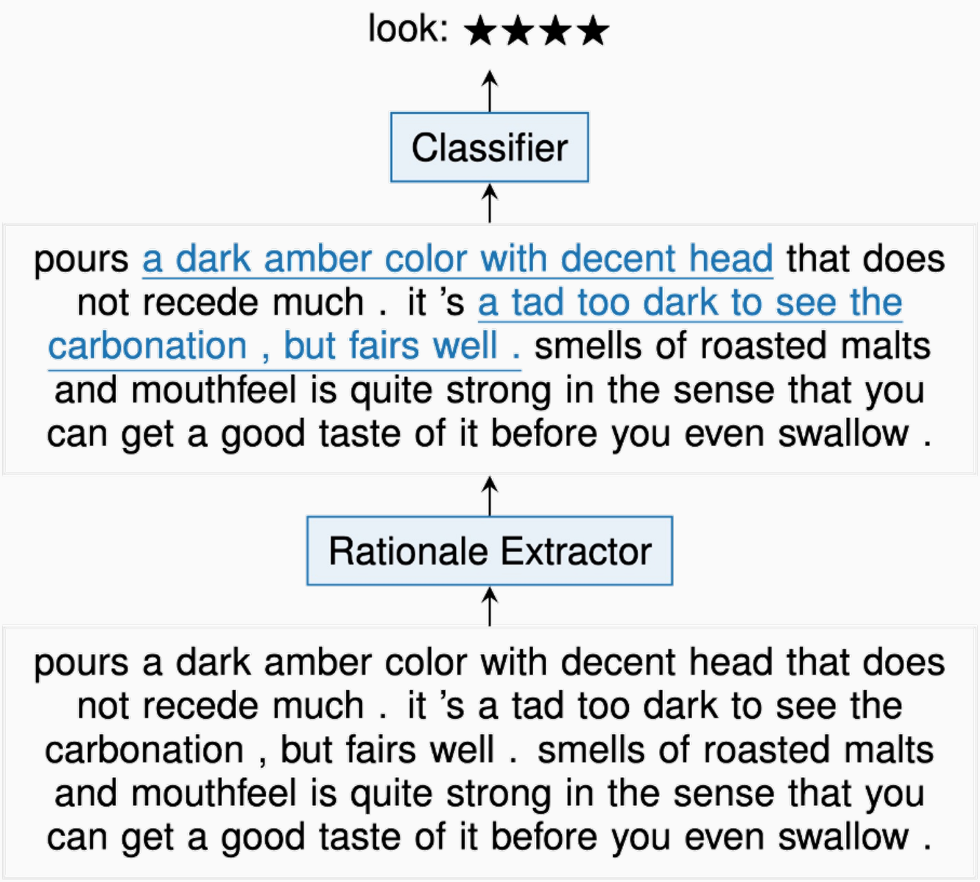
$$Z_i \mid x \sim Bernoulli\left(g_i(x, \phi)\right)$$

For each token, this represents whether the token is relevant or not

Each Z_i can be seen as a Boolean proposition

Example from Bastings, Jasmijn, Wilker Aziz, and Ivan Titov. "Interpretable Neural Predictions with Differentiable Binary Variables." In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 2963-2977. 2019.

Example: Text classification and rationales



An alternative approach:

1. Identify the rationale (the highlighted words)

$$Z_i \mid x \sim Bernoulli\big(g_i(x, \phi)\big)$$

2. Use only the highlighted words as input to the classifier

$$Y \mid x \sim Categorical\big(f(x \odot Z, \theta)\big)$$

Example from Bastings, Jasmijn, Wilker Aziz, and Ivan Titov. "Interpretable Neural Predictions with Differentiable Binary Variables." In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 2963-2977. 2019.

The Gumbel-Softmax trick

A strategy for training with discrete variables

Introduced concurrently by two papers in ICLR 2017:

- Jang, Eric, Shixiang Gu, and Ben Poole. "Categorical Reparameterization with Gumbel-Softmax"
- Maddison, Chris J., Andriy Mnih, and Yee Whye Teh. "The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables"

Combines two ideas:

1. The reparameterization trick
2. The Gumbel distribution

The Gumbel-Softmax trick

A strategy for training with discrete variables

Introduced concurrently by two papers in ICLR 2017:

- Jang, Eric, Shixiang Gu, and Ben Poole. "Categorical Reparameterization with Gumbel-Softmax"
- Maddison, Chris J., Andriy Mnih, and Yee Whye Teh. "The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables"

Combines two ideas:

1. The reparameterization trick
2. The Gumbel distribution

The reparameterization trick: Example

Suppose we have a sample from a Gaussian with mean μ and standard deviation σ

$$z \sim \mathrm{Normal}(\mu, \sigma)$$

And suppose we have some loss $L(z)$ defined over the sample

Can we compute the gradient of L with respect to the parameters of the Gaussian?

The reparameterization trick: Example

Suppose we have a sample from a Gaussian with mean μ and standard deviation σ

$$z \sim \text{Normal}(\mu, \sigma)$$

And suppose we have some loss $L(z)$ defined over the sample

Can we compute the gradient of L with respect to the parameters of the Gaussian?

No. When we sample z, we have broken the dependency between L and μ, σ

Let's rewrite the Gaussian sample

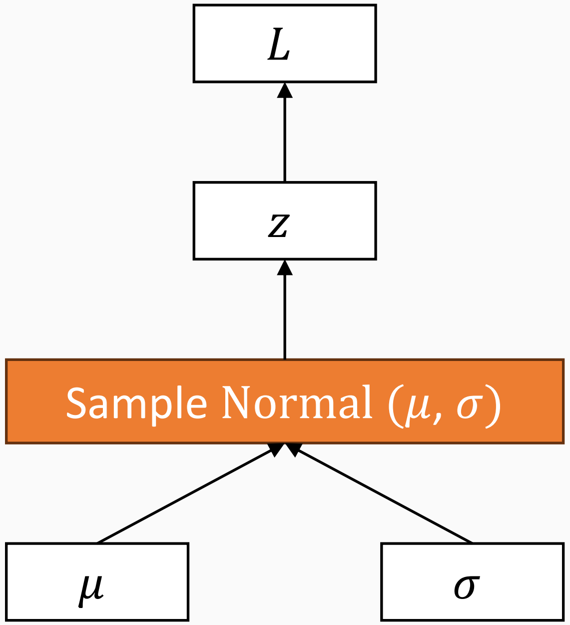
Instead of writing $z \sim \mathrm{Normal}(\mu, \sigma)$, we can write

$$\epsilon \sim \mathrm{Normal}(0, 1)$$
$$z = \mu + \sigma\epsilon$$

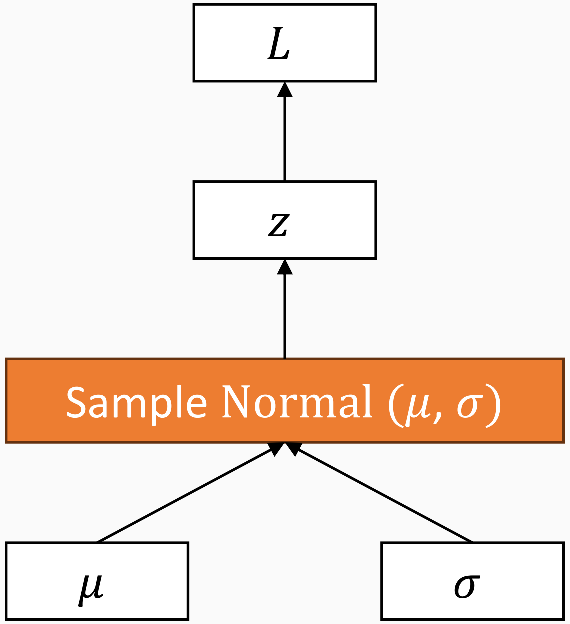
Now the desired function $L(z)$ has a direct dependency on μ and σ

We can take derivatives of L with respect to them

As computation graphs…



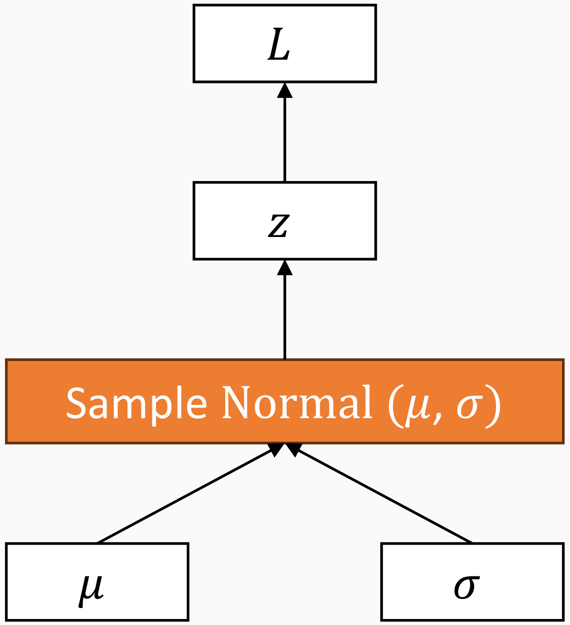
As computation graphs…



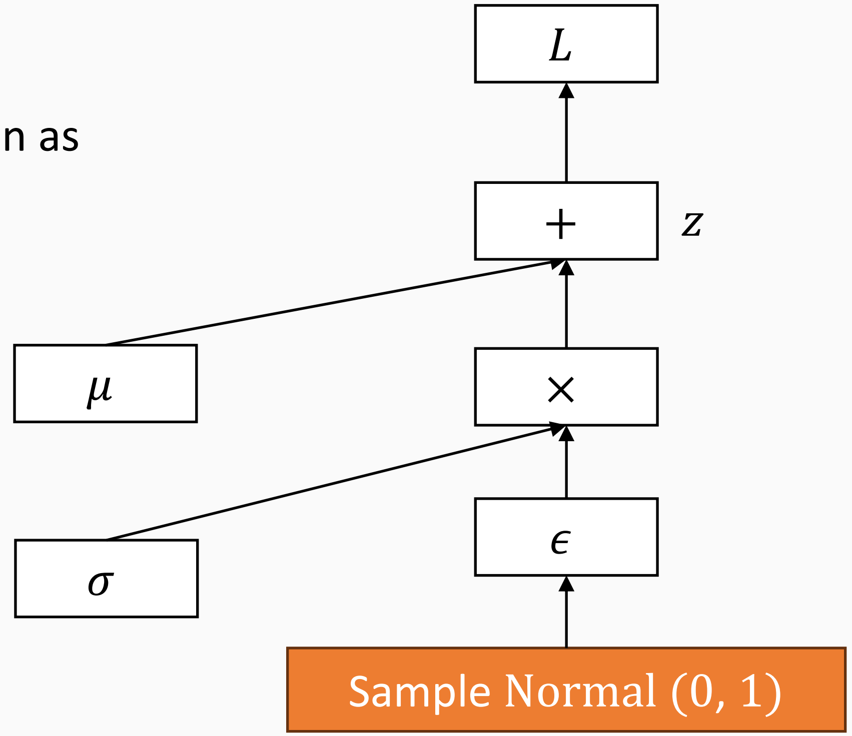
No backward path from the loss to the parameters because of the stochastic node

Cannot compute gradient ☹

As computation graphs…



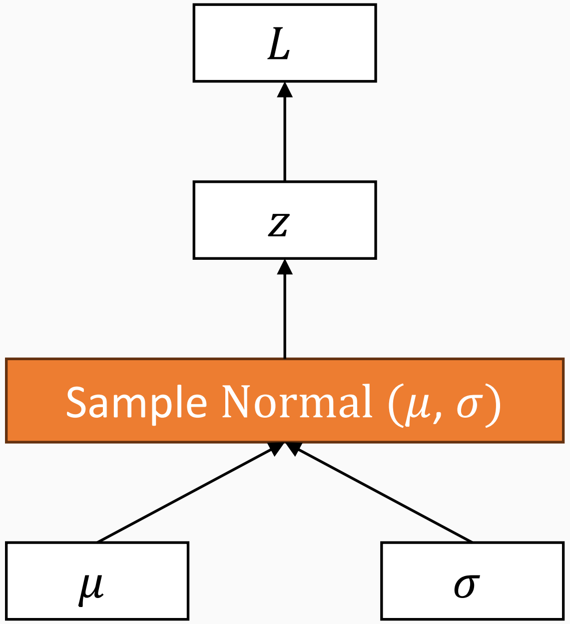
Rewrite the original function as



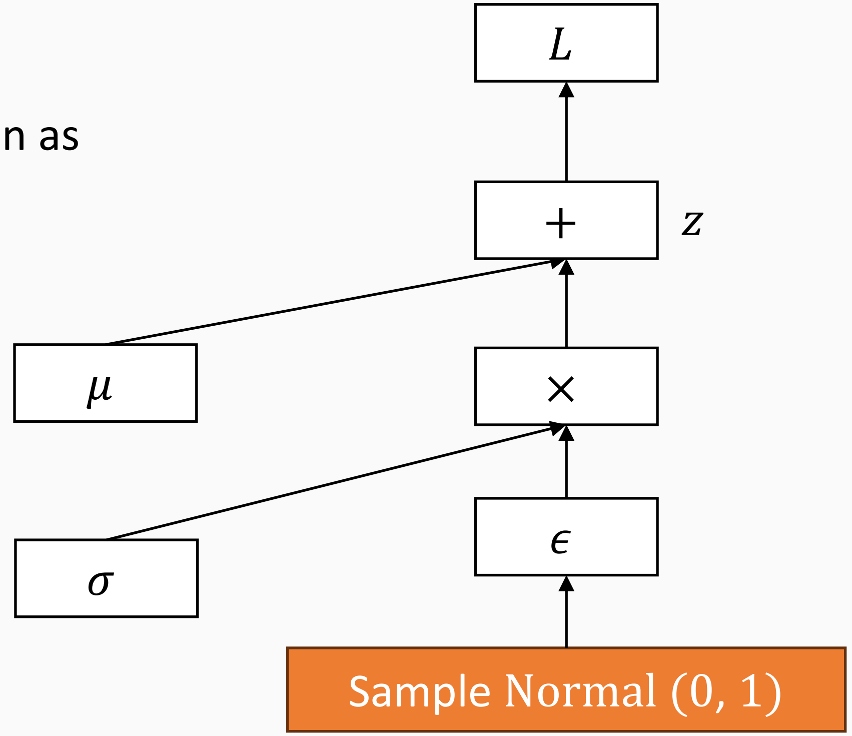
No backward path from the loss to the parameters because of the stochastic node

Cannot compute gradient ☹

As computation graphs…



Rewrite the original function as



No backward path from the loss to the parameters because of the stochastic node

Cannot compute gradient ☹

Stochastic node does not block any gradient computation with respect to μ and σ

The reparameterization trick: More generally

Suppose we have $\mathrm{z} \sim f(x, \phi)$ and $L(x, \phi, \theta) = g(z, \theta)$

The reparameterization trick: More generally

Suppose we have $\mathrm{z} \sim f(x, \phi)$ and $L(x, \phi, \theta) = g(z, \theta)$

Our goal is to compute the gradient of L with respect to both sets of parameters

The reparameterization trick: More generally

Suppose we have $\mathrm{z} \sim f(x, \phi)$ and $L(x, \phi, \theta) = g(z, \theta)$

Our goal is to compute the gradient of L with respect to both sets of parameters

$\nabla_\theta L$ is easy, so let us focus on $\nabla_\phi L$

The reparameterization trick: More generally

Suppose we have $\mathrm{z} \sim f(x, \phi)$ and $L(x, \phi, \theta) = g(z, \theta)$

Our goal is to compute the gradient of L with respect to both sets of parameters

1. Find a fixed *noise* distribution p and sample noise $\epsilon \sim p$

The reparameterization trick: More generally

Suppose we have $z \sim f(x, \phi)$ and $L(x, \phi, \theta) = g(z, \theta)$

Our goal is to compute the gradient of L with respect to both sets of parameters

1. Find a fixed *noise* distribution p and sample noise $\epsilon \sim p$
2. Define $z = h(x, \epsilon, \phi)$ such that z ends up being a sample from $f(x, \phi)$

The reparameterization trick: More generally

Suppose we have $z \sim f(x, \phi)$ and $L(x, \phi, \theta) = g(z, \theta)$

Our goal is to compute the gradient of L with respect to both sets of parameters

1. Find a fixed *noise* distribution p and sample noise $\epsilon \sim p$
2. Define $z = h(x, \epsilon, \phi)$ such that z ends up being a sample from $f(x, \phi)$
3. We have $L(x, \phi, \theta) = g(z, \theta) = g(h(x, \epsilon, \phi), \theta)$

The reparameterization trick: More generally

Suppose we have $\mathrm{z} \sim f(x, \phi)$ and $L(x, \phi, \theta) = g(z, \theta)$

Our goal is to compute the gradient of L with respect to both sets of parameters

1. Find a fixed *noise* distribution p and sample noise $\epsilon \sim p$
2. Define $z = h(x, \epsilon, \phi)$ such that z ends up being a sample from $f(x, \phi)$
3. We have $L(x, \phi, \theta) = g(z, \theta) = g(h(x, \epsilon, \phi), \theta)$

L is now a differentiable function of the parameters

The Gumbel-Softmax trick

A strategy for training with discrete variables

Introduced concurrently by two papers in ICLR 2017:

- Jang, Eric, Shixiang Gu, and Ben Poole. "Categorical Reparameterization with Gumbel-Softmax"
- Maddison, Chris J., Andriy Mnih, and Yee Whye Teh. "The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables"

Combines two ideas:

1. The reparameterization trick
2. The Gumbel distribution

Gumbel distribution

Models the distribution of the max of a set of samples from an exponential distribution

The Gumbel distribution is defined by the cdf

$$F(x) = \exp\left(-\exp\left(-\frac{x-\mu}{\beta}\right)\right)$$

Gumbel distribution

Models the distribution of the max of a set of samples from another distribution

The Gumbel distribution is defined by the cdf

Location

$$F(x) = \exp\left(-\exp\left(-\frac{x-\mu}{\beta}\right)\right)$$

Scale

Gumbel distribution

Models the distribution of the max of a set of samples from another distribution

The Gumbel distribution is defined by the cdf

Location

$$F(x) = \exp\left(-\exp\left(-\frac{x - \mu}{\beta}\right)\right)$$

Scale

For the "standard" Gumbel distribution, we have $\mu = 0, \sigma = 1$

Gumbel distribution

Models the distribution of the max of a set of samples from another distribution

The Gumbel distribution is defined by the cdf

$$F(x) = \exp\left(-\exp\left(-\frac{x-\mu}{\beta}\right)\right)$$

Sampling from this distribution is easy

$$g \sim -\log\left(-\log\left(\text{Uniform}(0,1)\right)\right)$$

Gumbel distribution

Models the distribution of the max of a set of samples from another distribution

The Gumbel distribution is defined by the cdf

$$F(x) = \exp\left(-\exp\left(-\frac{x-\mu}{\beta}\right)\right)$$

Sampling from this distribution is easy

$g \sim$ numpy.random.Generator.gumbel

method

```
random.Generator.gumbel(loc=0.0, scale=1.0, size=None)
```

Draw samples from a Gumbel distribution.

Draw samples from a Gumbel distribution with specified location and scale. For more information on the Gumbel distribution, see Notes and References below.

Why is the Gumbel distribution interesting?

Suppose we have a set of real valued scores $x_1, x_2, \cdots, x_k$ assigned to k discrete categories

Why is the Gumbel distribution interesting?

Suppose we have a set of real valued scores $x_1, x_2, \cdots, x_k$ assigned to k discrete categories

We wish to sample from the softmax of these scores, where:

$$P(i) = \frac{\exp(x_i)}{\sum_j \exp(x_j)}$$

Why is the Gumbel distribution interesting?

Suppose we have a set of real valued scores $x_1, x_2, \cdots, x_k$ assigned to k discrete categories

We wish to sample from the softmax of these scores, where:

$$P(i) = \frac{\exp(x_i)}{\sum_j \exp(x_j)}$$

To do so, generate k independent samples from the standard Gumbel distribution $g_1, g_2, \cdots, g_k$

Why is the Gumbel distribution interesting?

Suppose we have a set of real valued scores $x_1, x_2, \cdots, x_k$ assigned to k discrete categories

We wish to sample from the softmax of these scores, where:

$$P(i) = \frac{\exp(x_i)}{\sum_j \exp(x_j)}$$

To do so, generate k independent samples from the *standard* Gumbel distribution $g_1, g_2, \cdots, g_k$

Then $\arg\max_i (x_i + g_i) \sim \mathrm{softmax}(x_1, x_2, \cdots, x_k)$

The argmax is a sample from the desired distribution!

Why is the Gumbel distribution interesting?

Suppose we have a set of real valued scores $x_1, x_2, \cdots, x_k$ assigned to k discrete categories

We wish to sample from the softmax of these scores, where:

$$P(i) = \frac{\exp(x_i)}{\sum_j \exp(x_j)}$$

To do so, generate k independent samples from the *standard* Gumbel distribution $g_1, g_2, \cdots, g_k$

Then $\arg\max_i (x_i + g_i) \sim \text{softmax}(x_1, x_2, \cdots, x_k)$

The argmax is a sample from the desired distribution!

Exercise: How would you prove this?

Why is the Gumbel distribution interesting?

Suppose we have a set of real valued scores $x_1, x_2, \cdots, x_k$ assigned to k discrete categories

We wish to sample from the softmax of these scores, where:

$$P(i) = \frac{\exp(x_i)}{\sum_j \exp(x_j)}$$

These scores could be assigned to more complicated structures as well. The same procedure works

To do so, generate k independent samples from the standard Gumbel distribution $g_1, g_2, \cdots, g_k$

Then $\arg\max_i (x_i + g_i) \sim \mathrm{softmax}(x_1, x_2, \cdots, x_k)$

The argmax is a sample from the desired distribution!

Why is the Gumbel distribution interesting?

Suppose we have a set of real valued scores $x_1, x_2, \cdots, x_k$ assigned to k discrete categories

We wish to sample from the softmax of these scores, where:

$$P(i) = \frac{\exp(x_i)}{\sum_j \exp(x_j)}$$

These scores could be assigned to more complicated structures as well. The same procedure works

To do so, generate k independent samples from the standard Gumbel distribution $g_1, g_2, \cdots, g_k$

Then $\arg\max_i (x_i + g_i) \sim \mathrm{softmax}(x_1, x_2, \cdots, x_k)$

Reduces the problem of estimating a sample from a distribution to a maximization problem

The argmax is a sample from the desired distribution!

The Gumbel-Softmax trick

A strategy for training with discrete variables

Introduced concurrently by two papers in ICLR 2017:

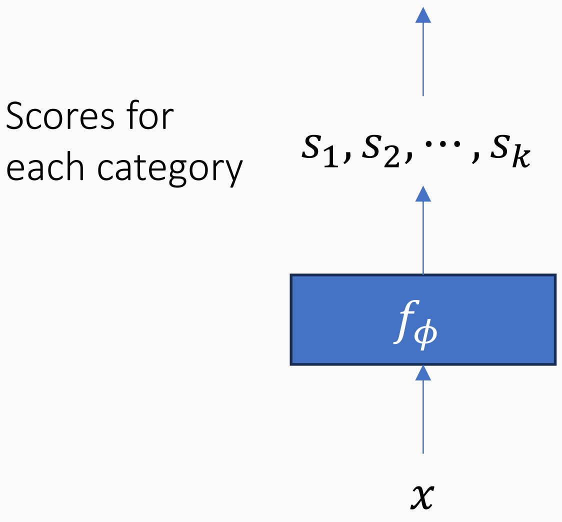
- Jang, Eric, Shixiang Gu, and Ben Poole. “Categorical Reparameterization with Gumbel-Softmax”
- Maddison, Chris J., Andriy Mnih, and Yee Whye Teh. “The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables”

Combines two ideas:

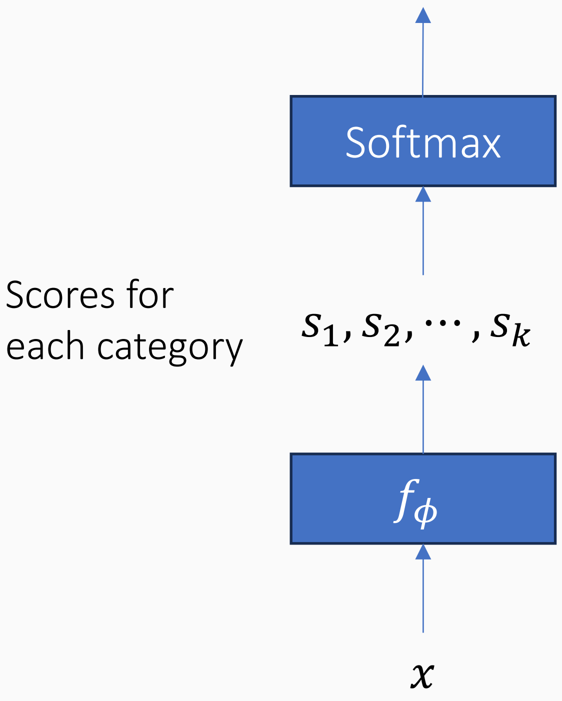
1. The reparameterization trick
2. The Gumbel distribution

Sampling from a categorical distribution

Suppose we have a component of a neural network that produces scores for k categories



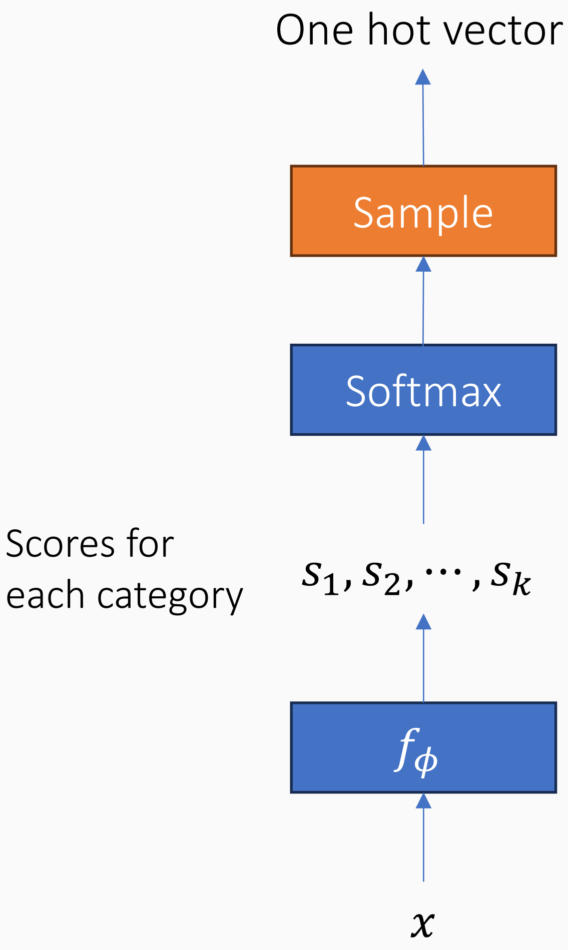
Sampling from a categorical distribution



Suppose we have a component of a neural network that produces scores for k categories

This corresponds to a categorical distribution via softmax

Sampling from a categorical distribution

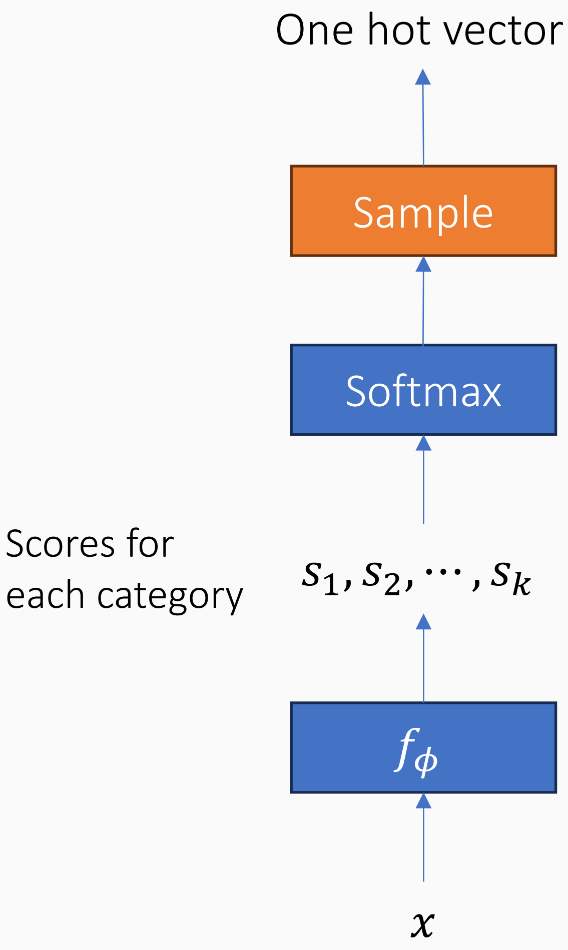


Suppose we have a component of a neural network that produces scores for k categories

This corresponds to a categorical distribution via softmax

And we can sample from the distribution to produce a k-dimensional one-hot vector

Sampling from a categorical distribution



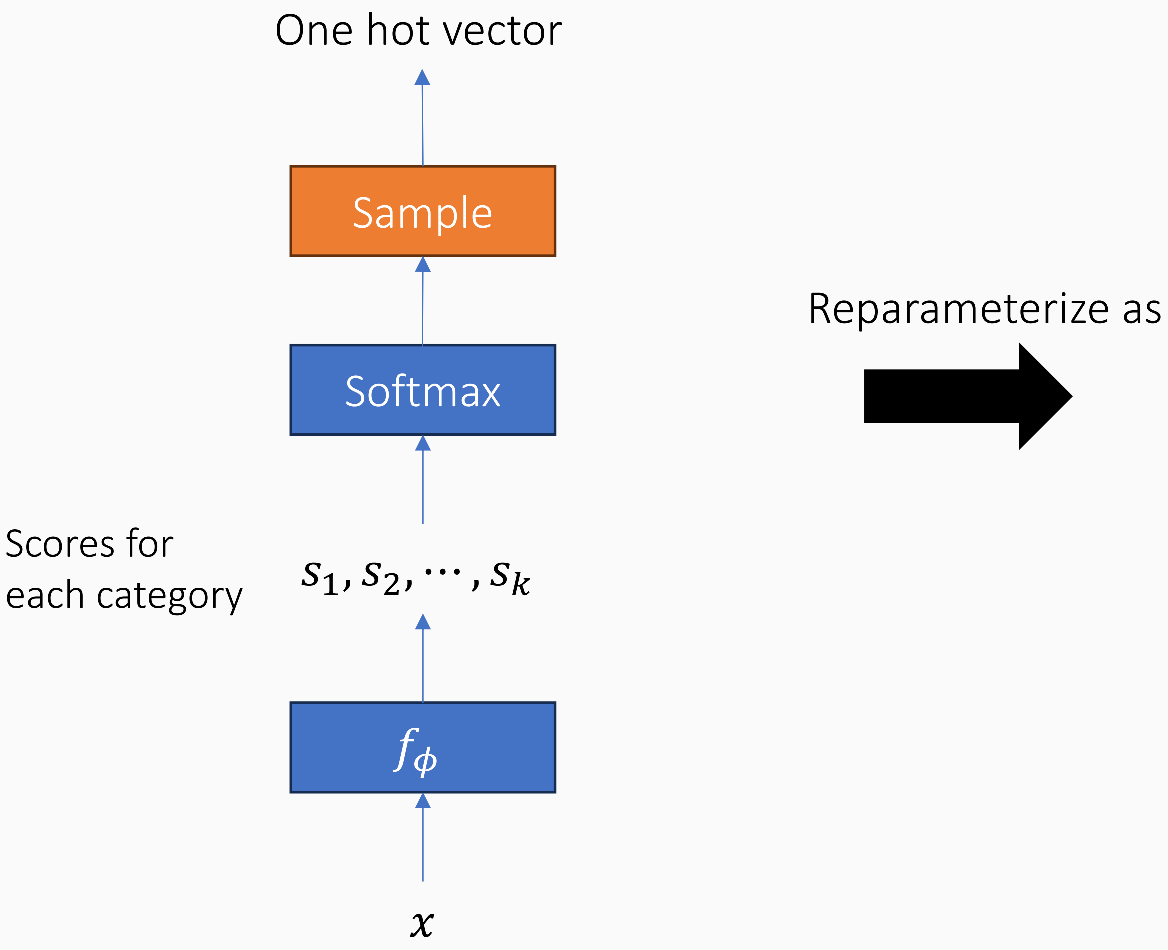
Suppose we have a component of a neural network that produces scores for k categories

This corresponds to a categorical distribution via softmax

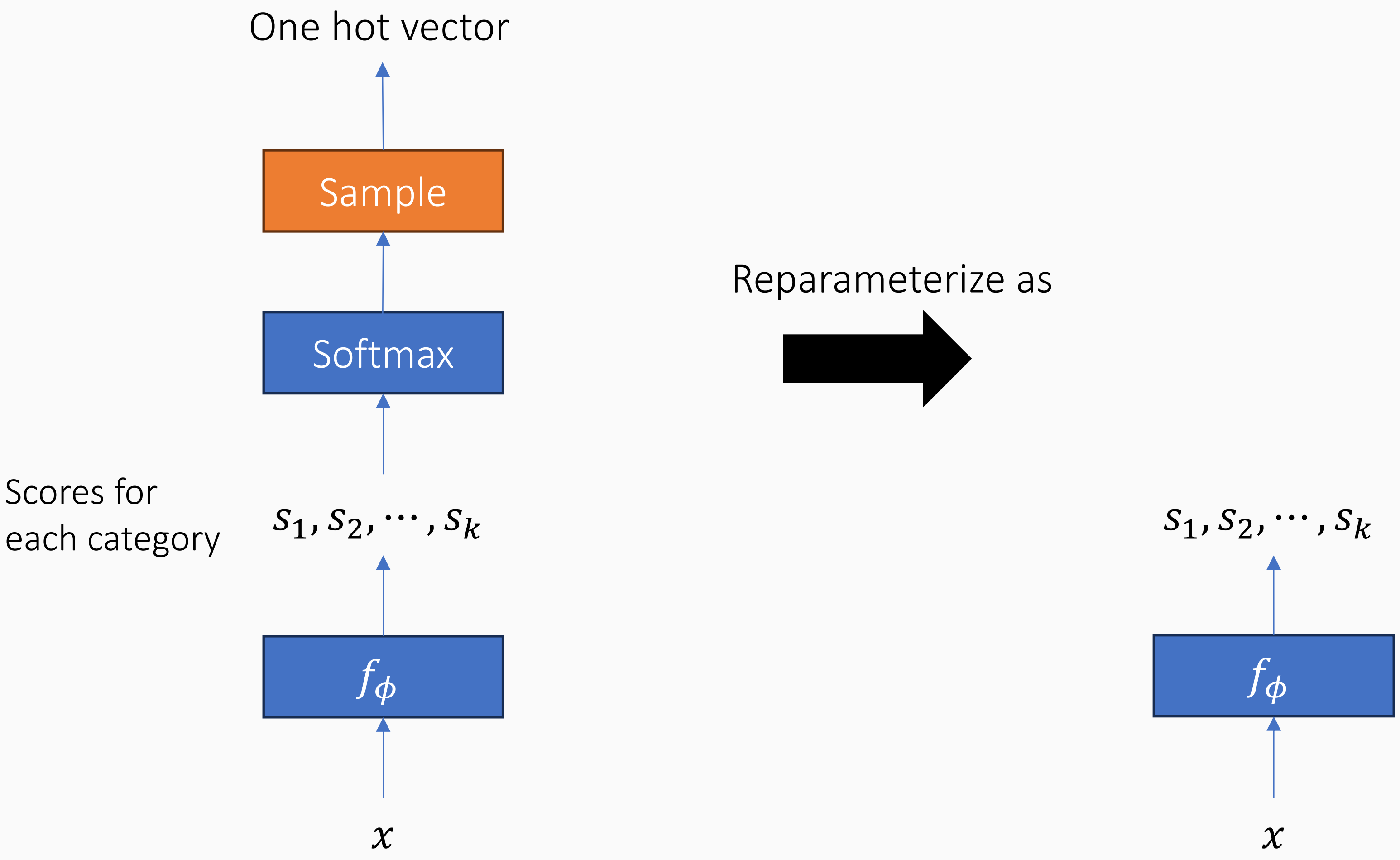
And we can sample from the distribution

But backprop is no longer viable

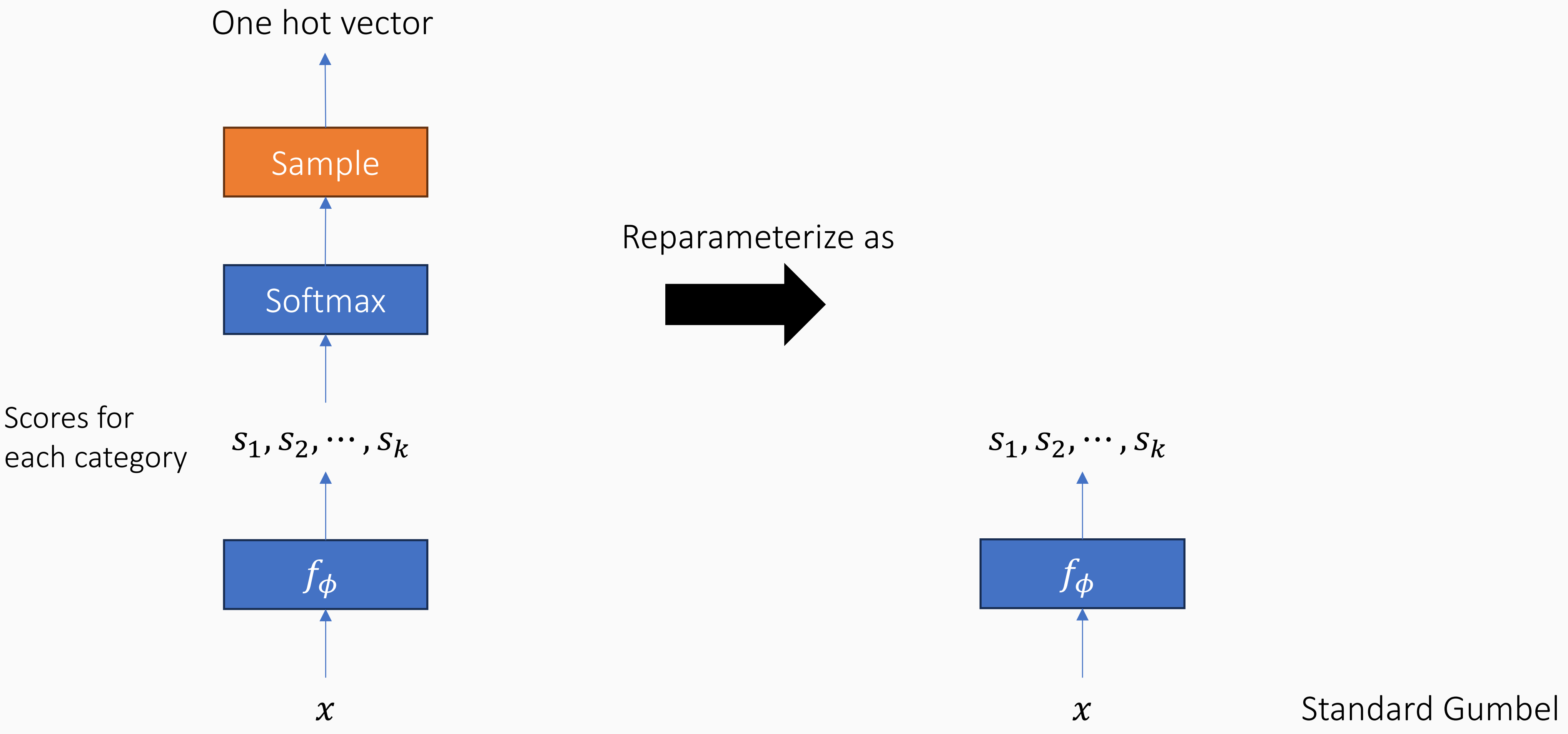
Sampling from a categorical distribution



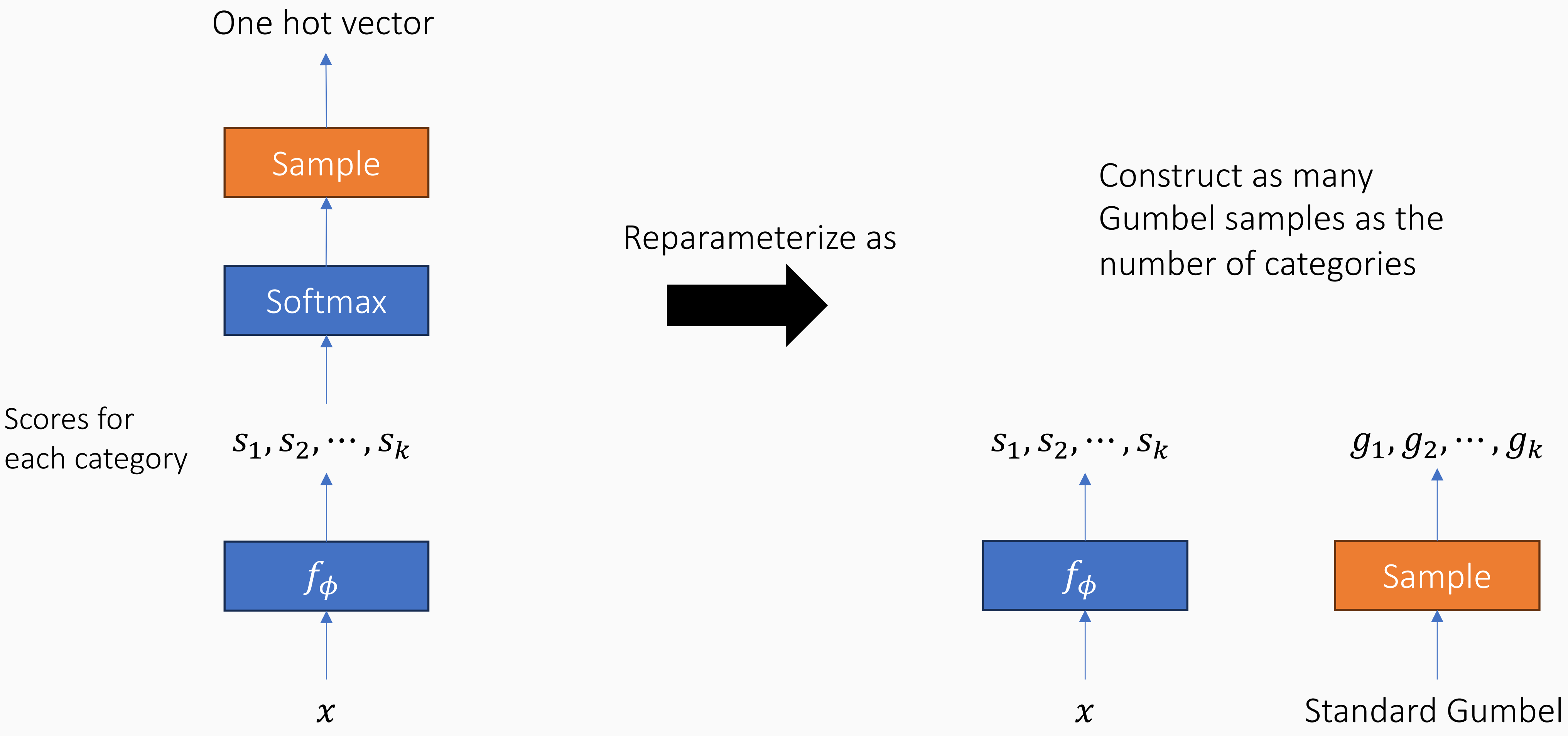
Sampling from a categorical distribution



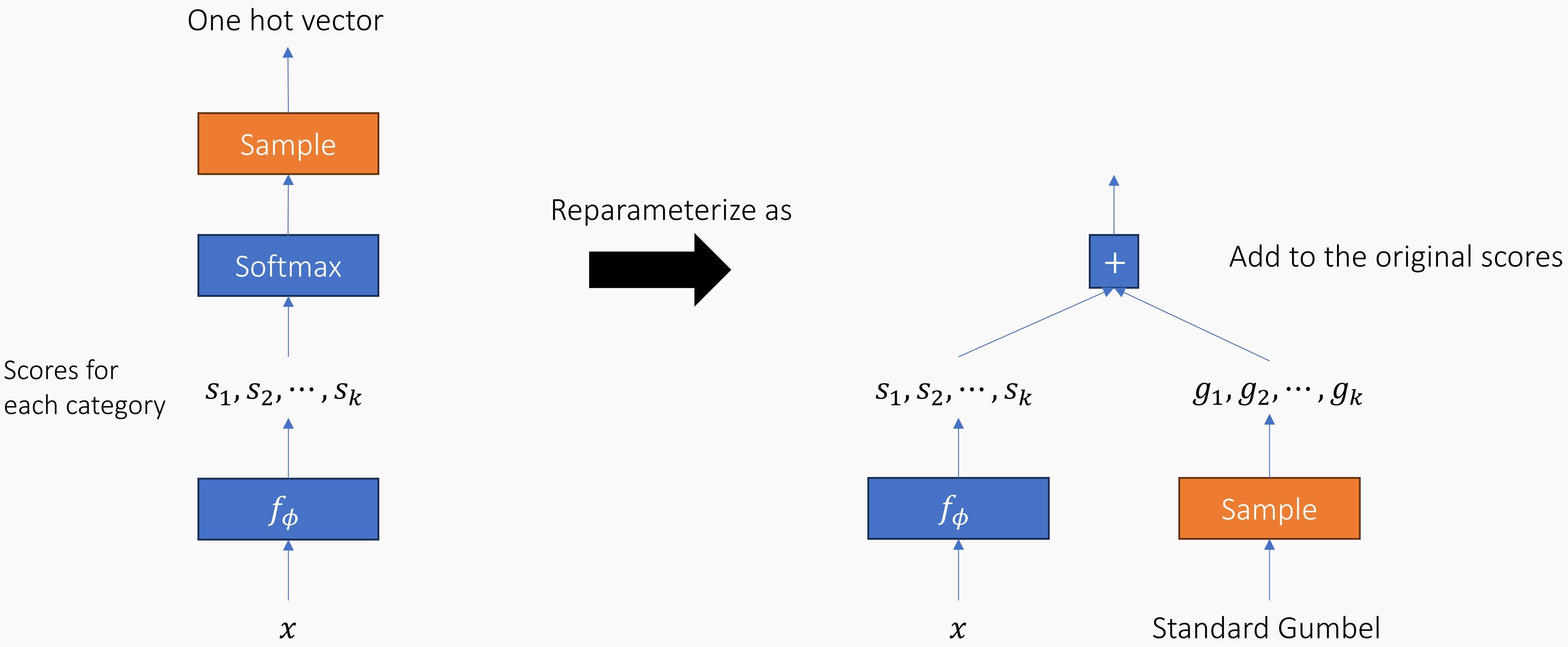
Sampling from a categorical distribution



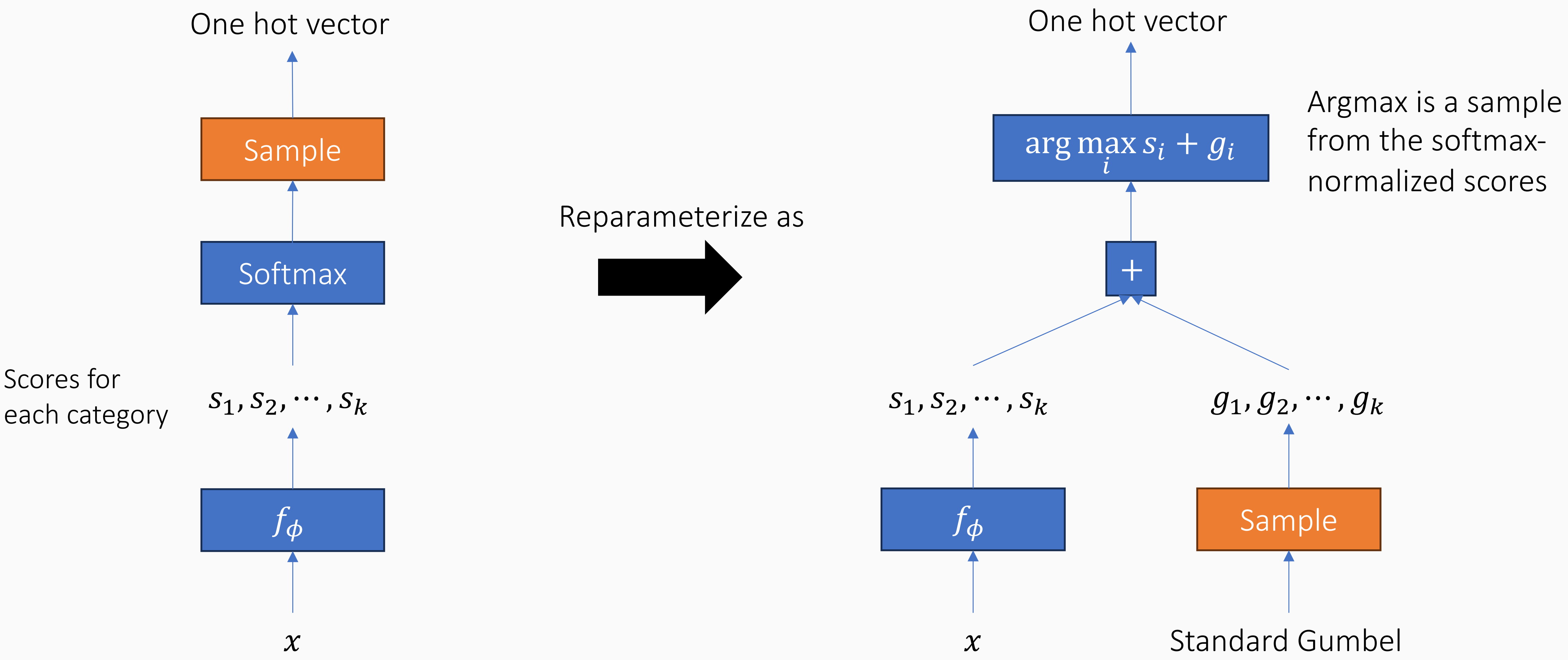
Sampling from a categorical distribution



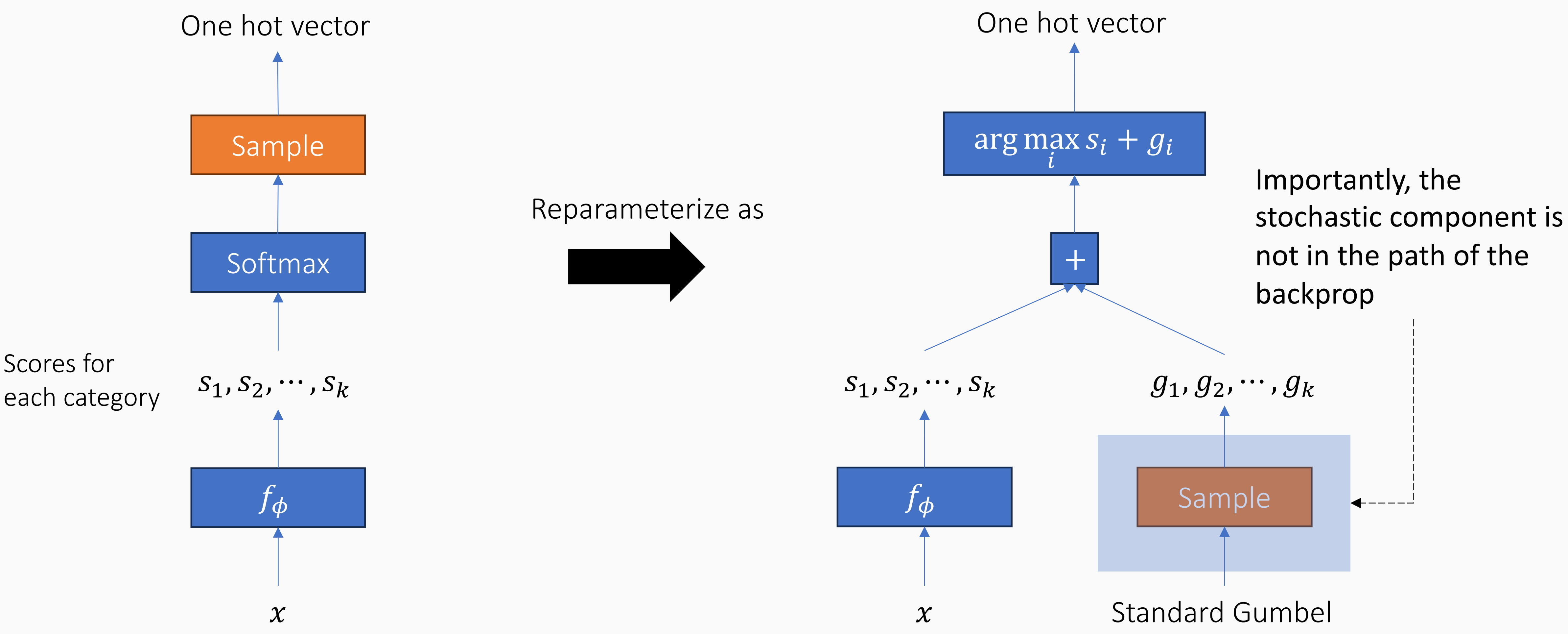
Sampling from a categorical distribution



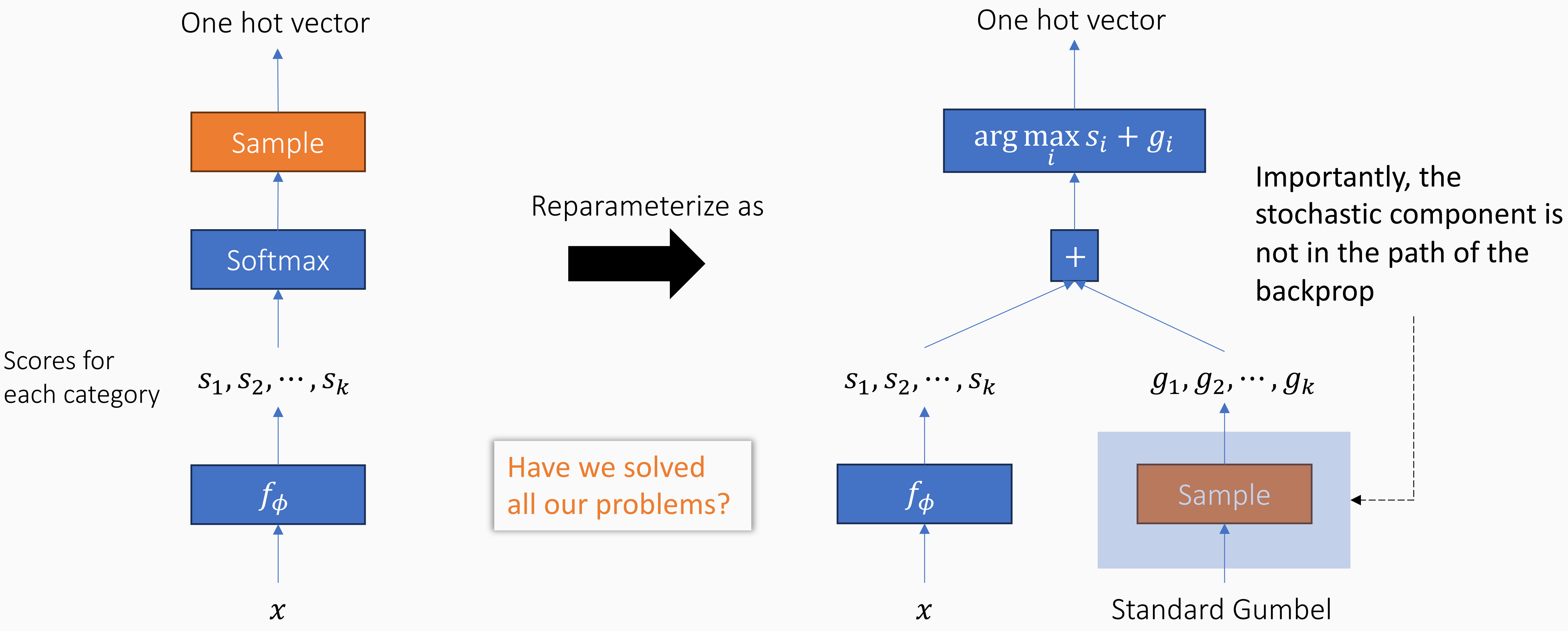
Sampling from a categorical distribution



Sampling from a categorical distribution

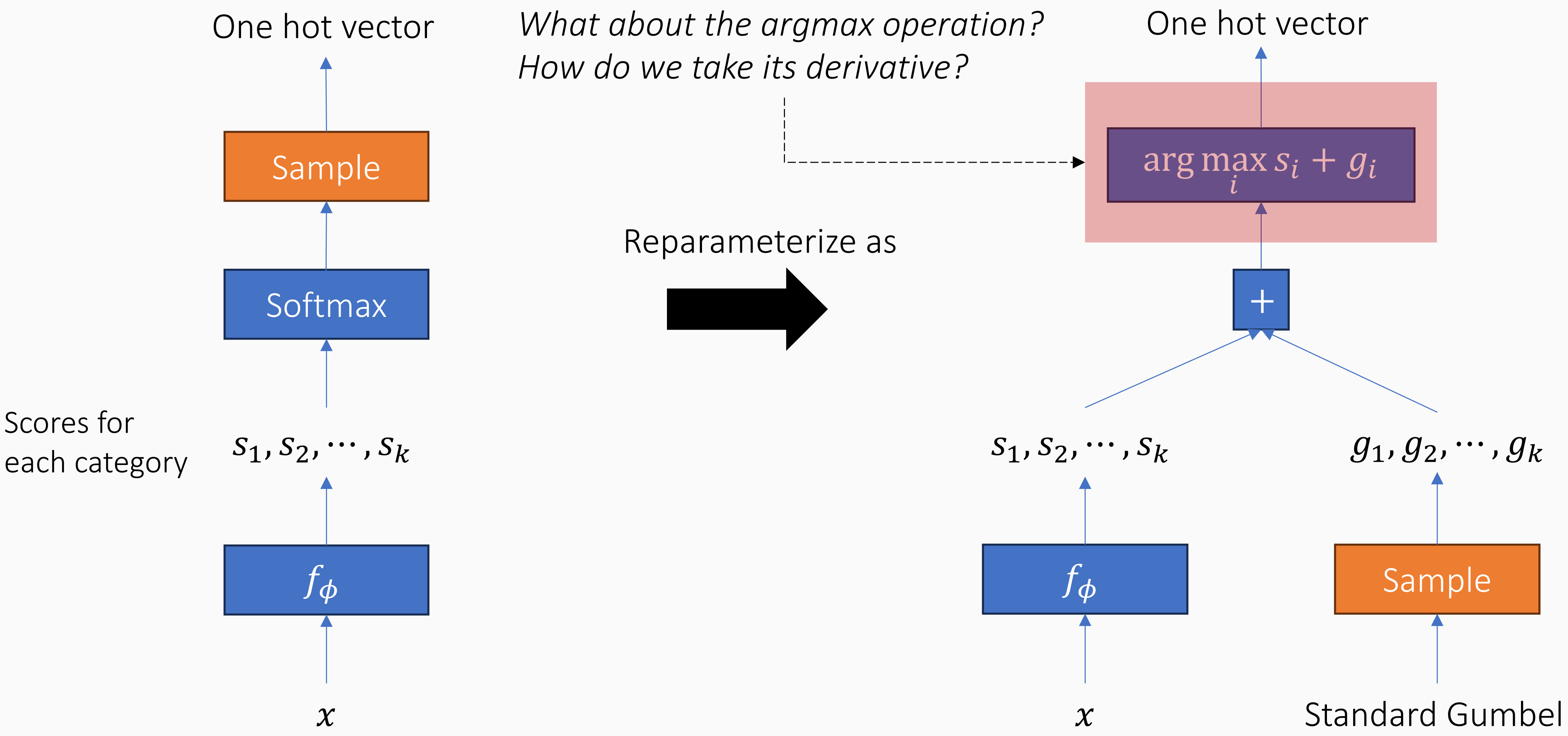


Sampling from a categorical distribution

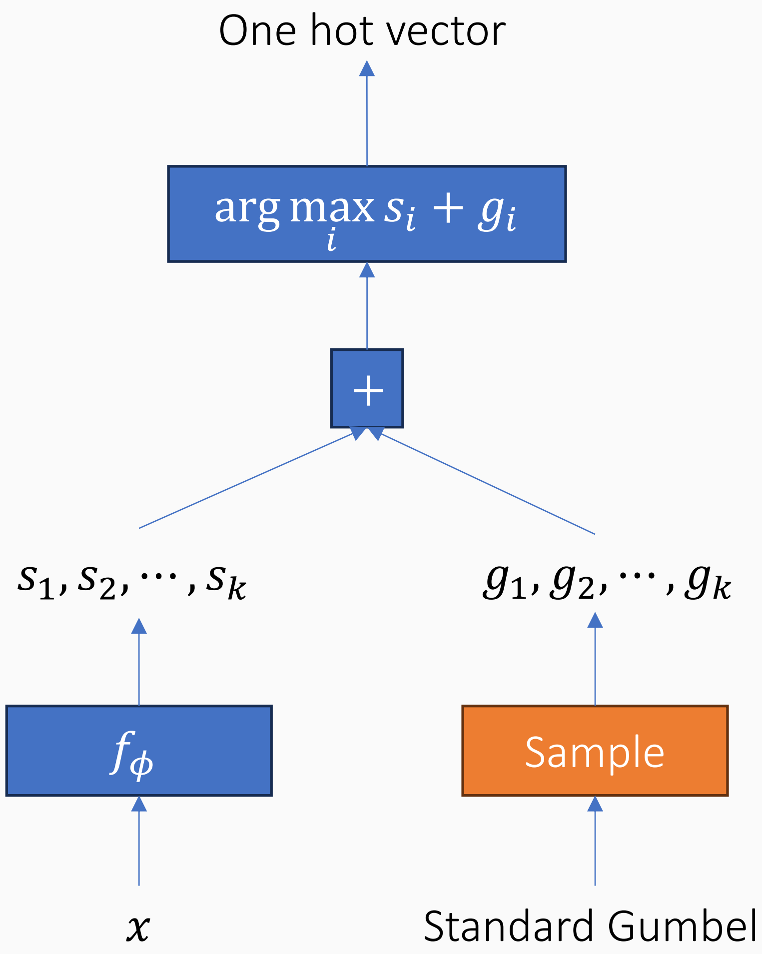


Have we solved all our problems?

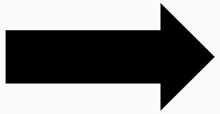
Sampling from a categorical distribution



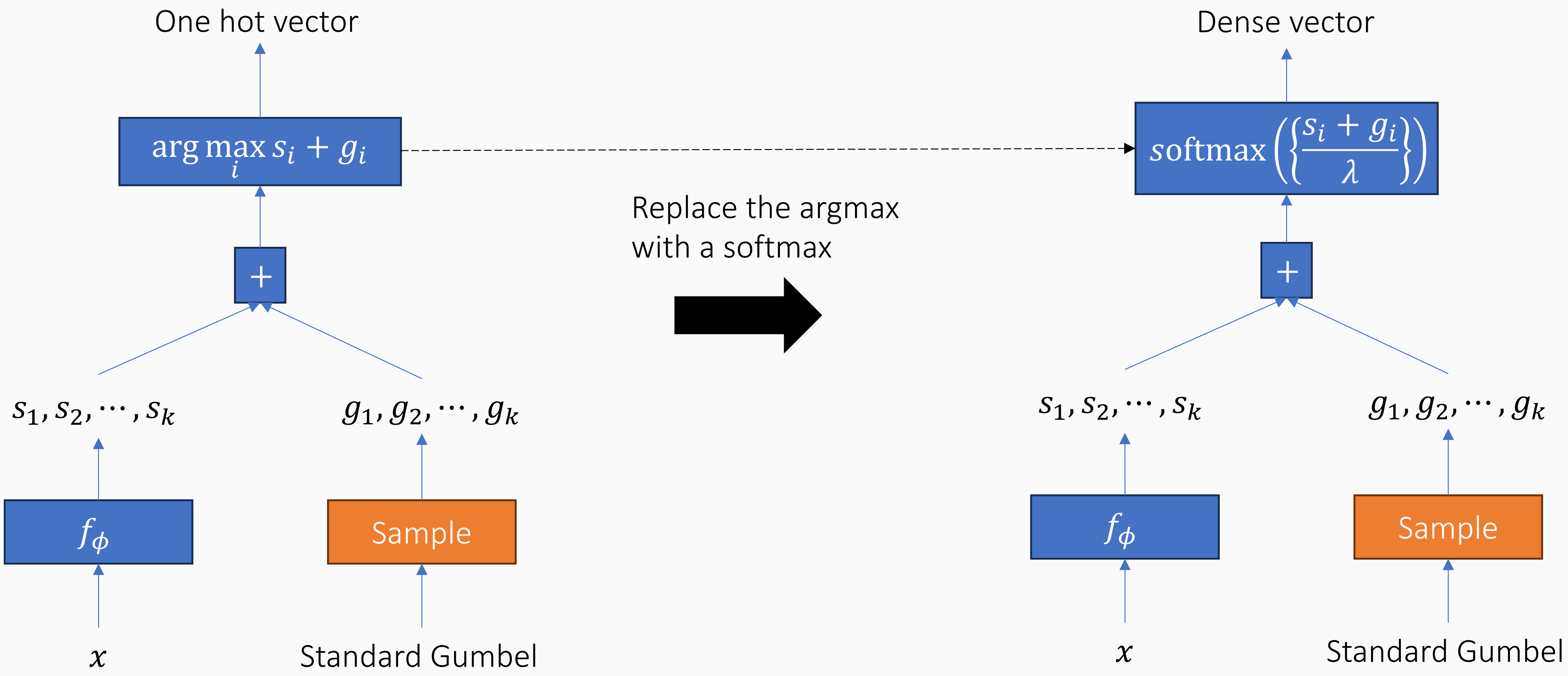
The Gumbel-softmax solution



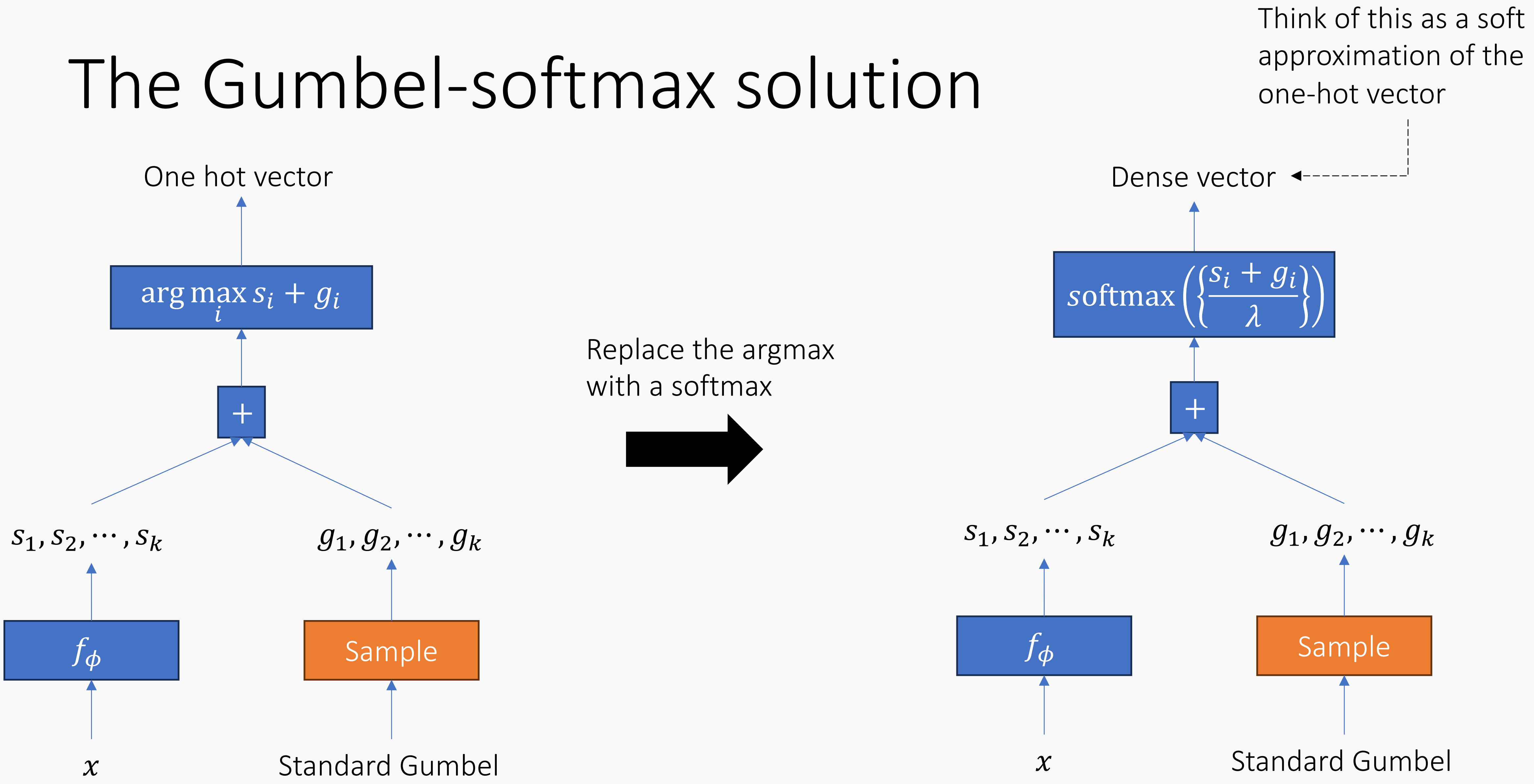
Replace the argmax
with a softmax



The Gumbel-softmax solution

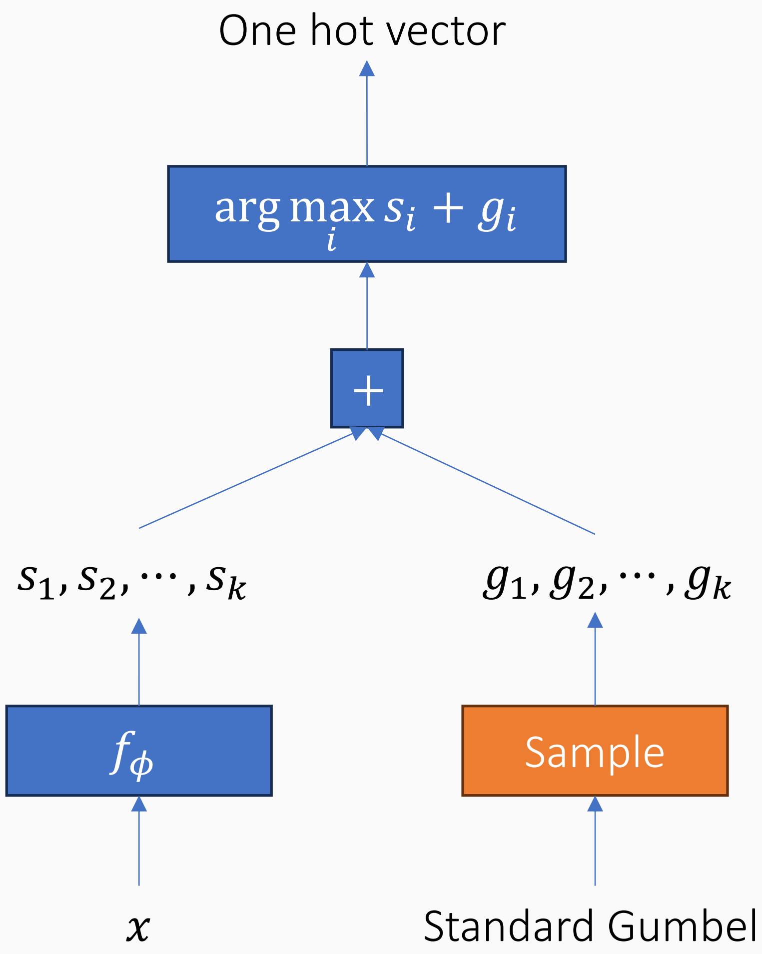


The Gumbel-softmax solution



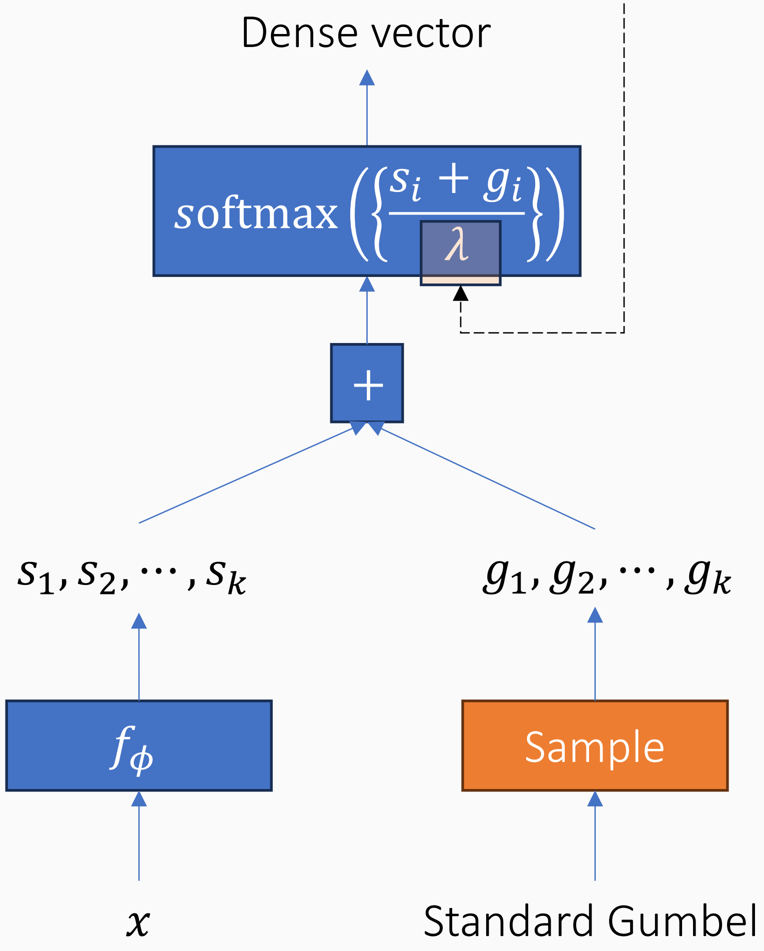
The Gumbel-softmax solution

The λ is the *softmax temperature*
High $\lambda \rightarrow$ uniform distribution
Low $\lambda \rightarrow$ Closer to argmax



Replace the argmax with a softmax

The resulting network is now fully differentiable w.r.t the parameters



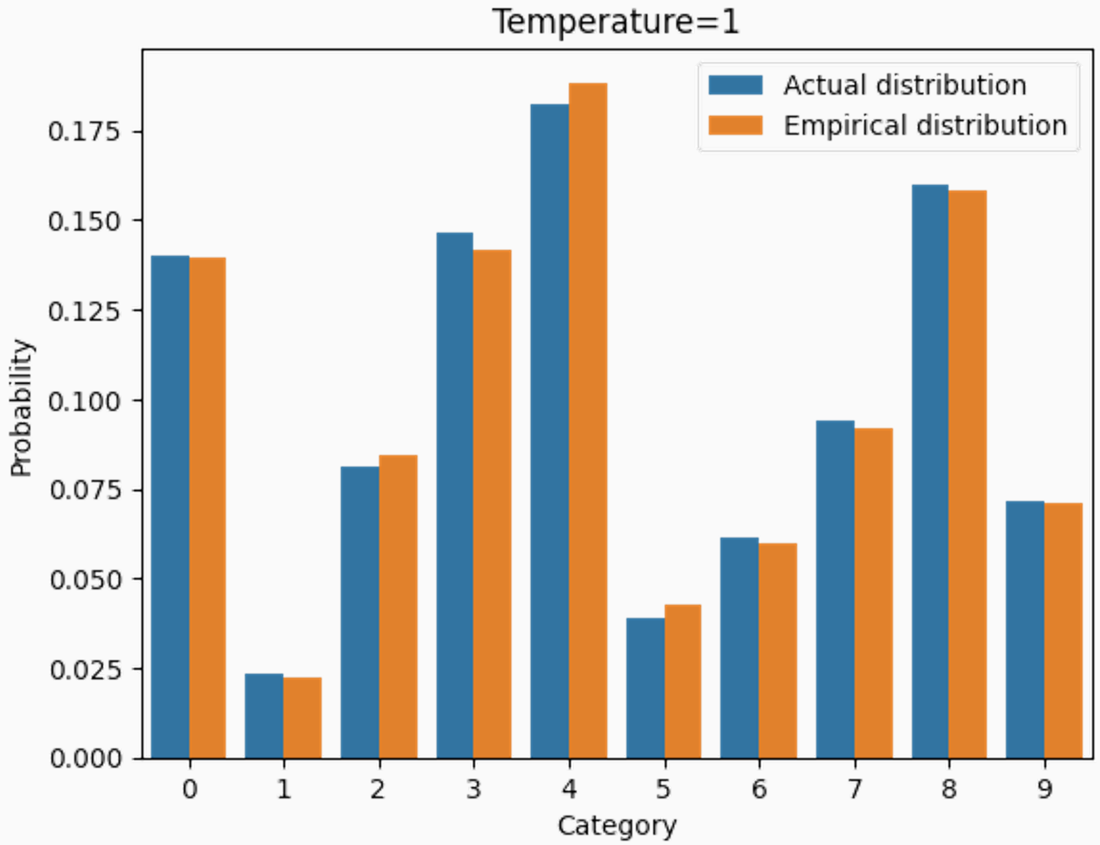
What does the temperature do? An example

An experiment:

1. Randomly create a set of scores over ten categories

2. Sample 10k times for different values of temperature

3. Compute empirical distribution of samples and compare to softmax of the scores



When temperature = 1, the samples represent the true distribution

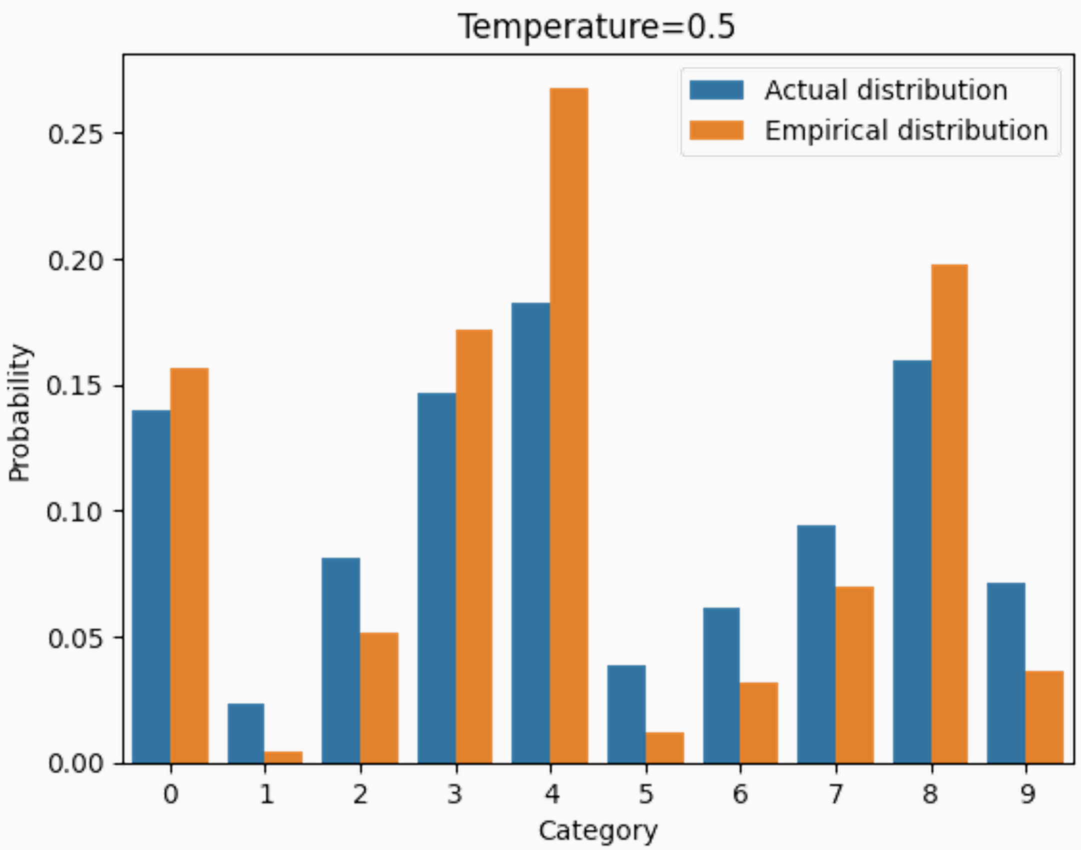
What does the temperature do? An example

An experiment:

1. Randomly create a set of scores over ten categories

2. Sample 10k times for different values of temperature

3. Compute empirical distribution of samples and compare to softmax of the scores



When temperature = 0.5, the empirical distribution becomes more peaky

More probable categories are over-represented

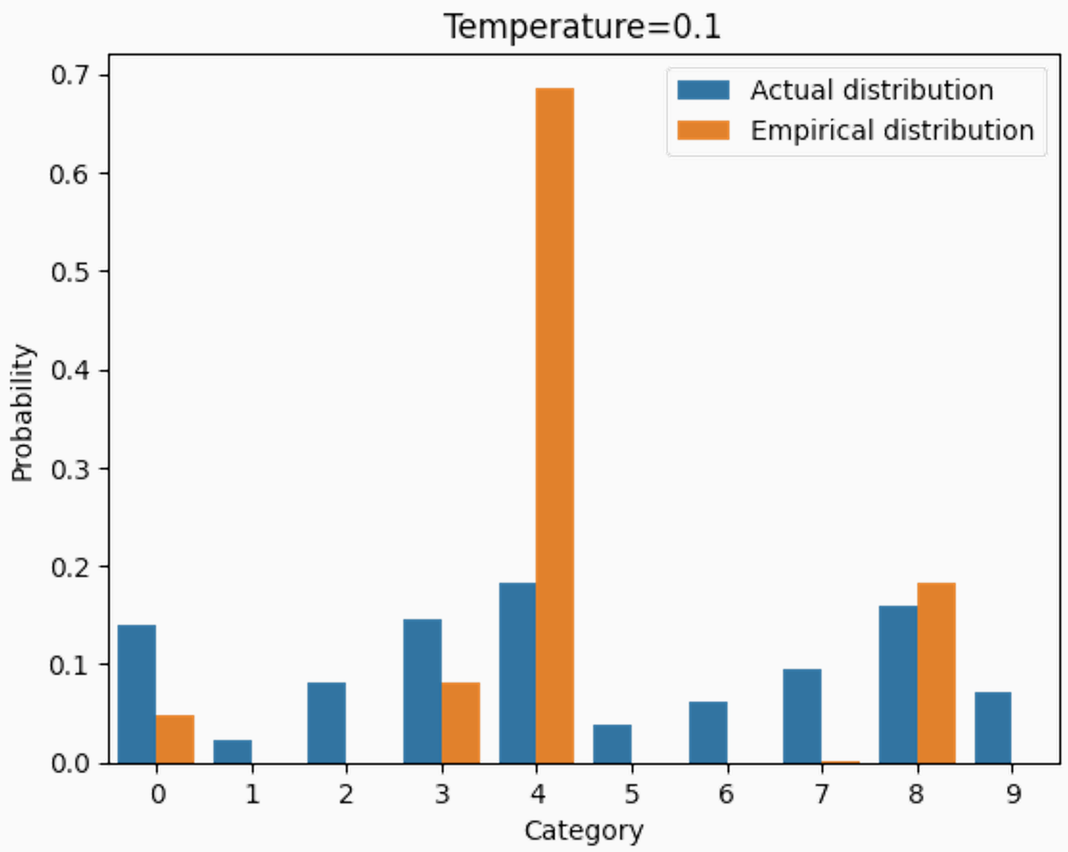
What does the temperature do? An example

An experiment:

1. Randomly create a set of scores over ten categories

2. Sample 10k times for different values of temperature

3. Compute empirical distribution of samples and compare to softmax of the scores



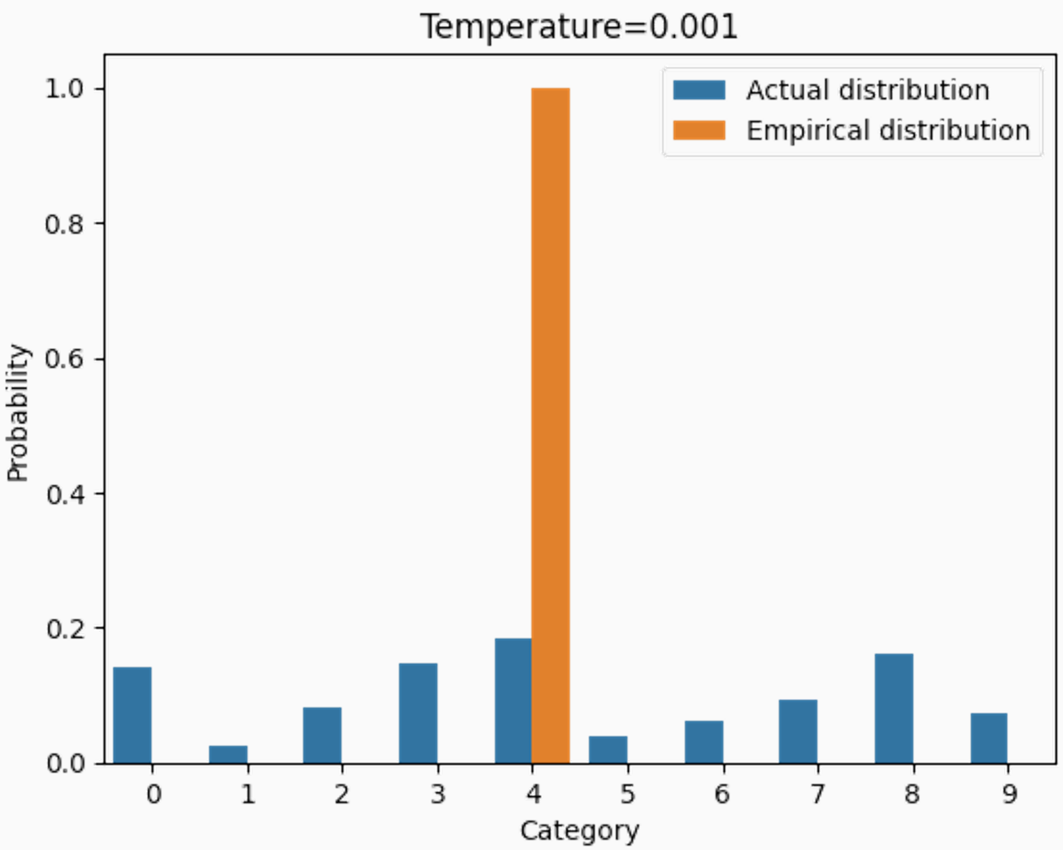
When temperature = 0.1, the peakiness is more visible

Only a few categories show up in the samples

What does the temperature do? An example

An experiment:

1. Randomly create a set of scores over ten categories
2. Sample 10k times for different values of temperature
3. Compute empirical distribution of samples and compare to softmax of the scores



When temperature = 0.001, only the most probable category (i.e. the argmax) is sampled

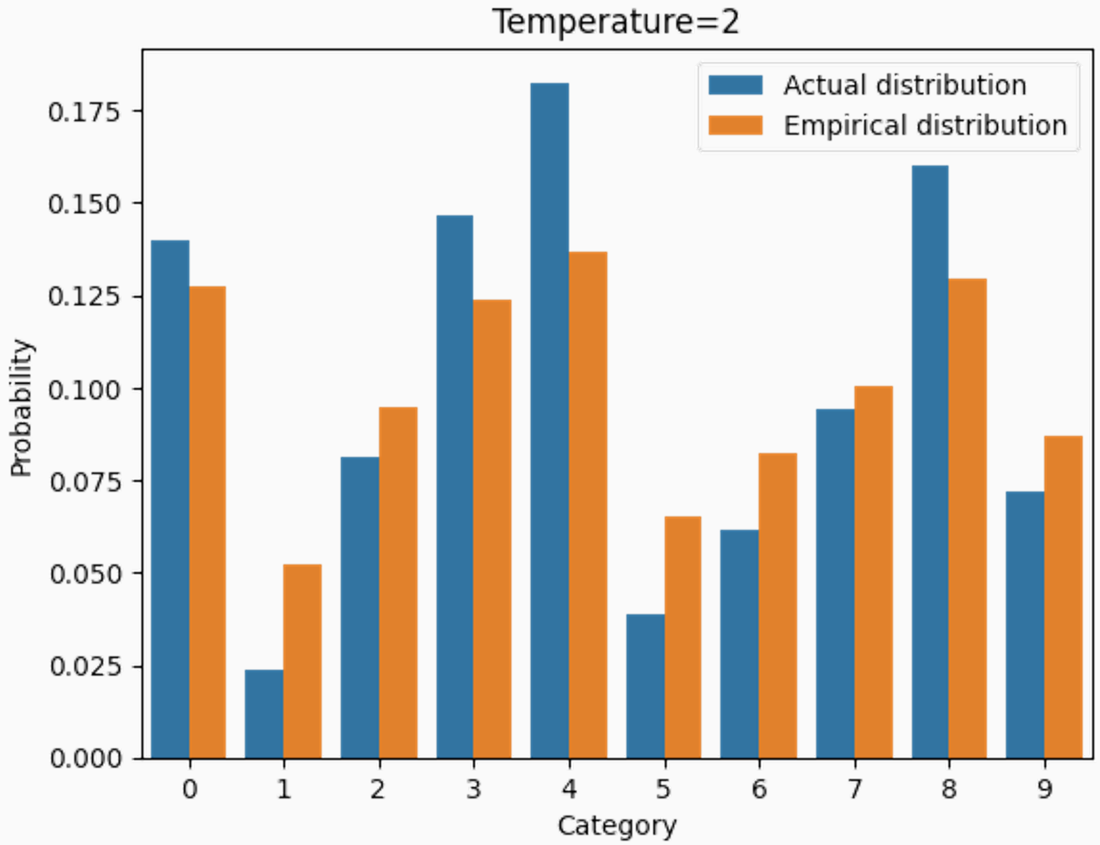
What does the temperature do? An example

An experiment:

1. Randomly create a set of scores over ten categories

2. Sample 10k times for different values of temperature

3. Compute empirical distribution of samples and compare to softmax of the scores



When the temperature is increased to 2, the empirical distribution becomes more "flat" than the actual softmax

More probable categories are undersampled and less probable ones are oversampled

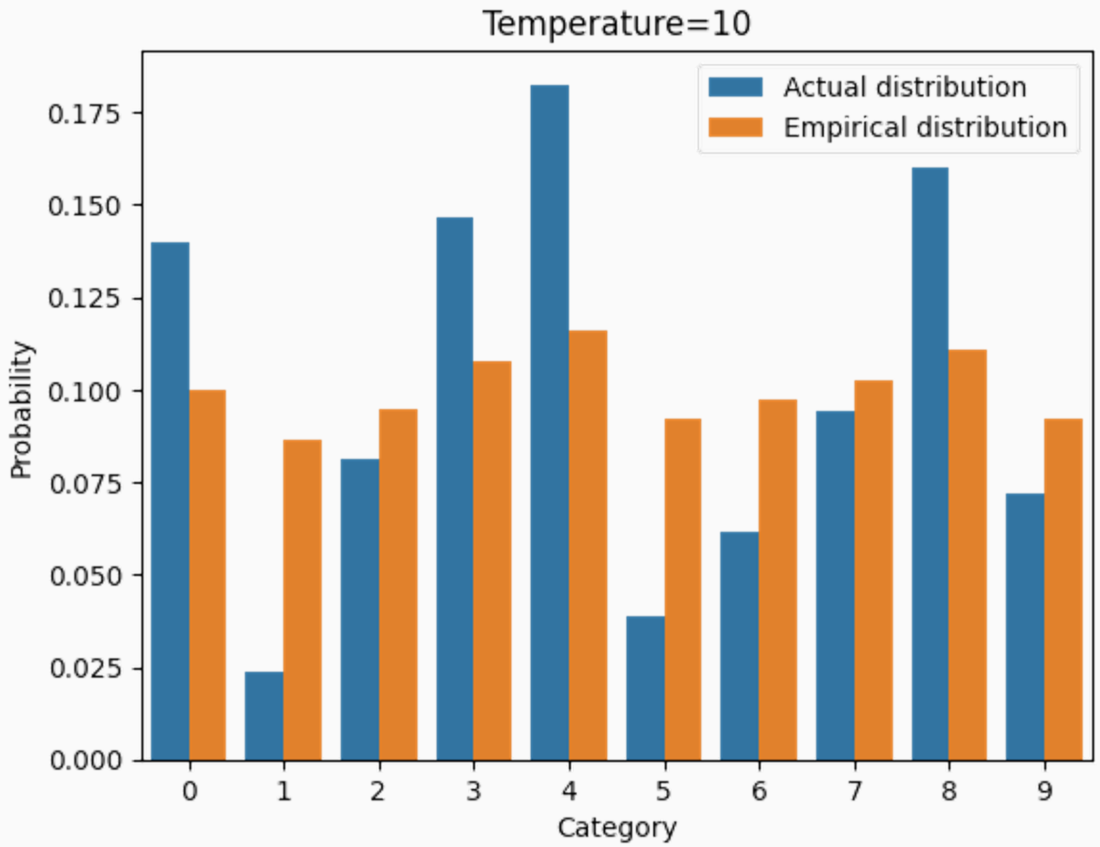
What does the temperature do? An example

An experiment:

1. Randomly create a set of scores over ten categories

2. Sample 10k times for different values of temperature

3. Compute empirical distribution of samples and compare to softmax of the scores



When the temperature is increased to 10, the “flatness” is more visible

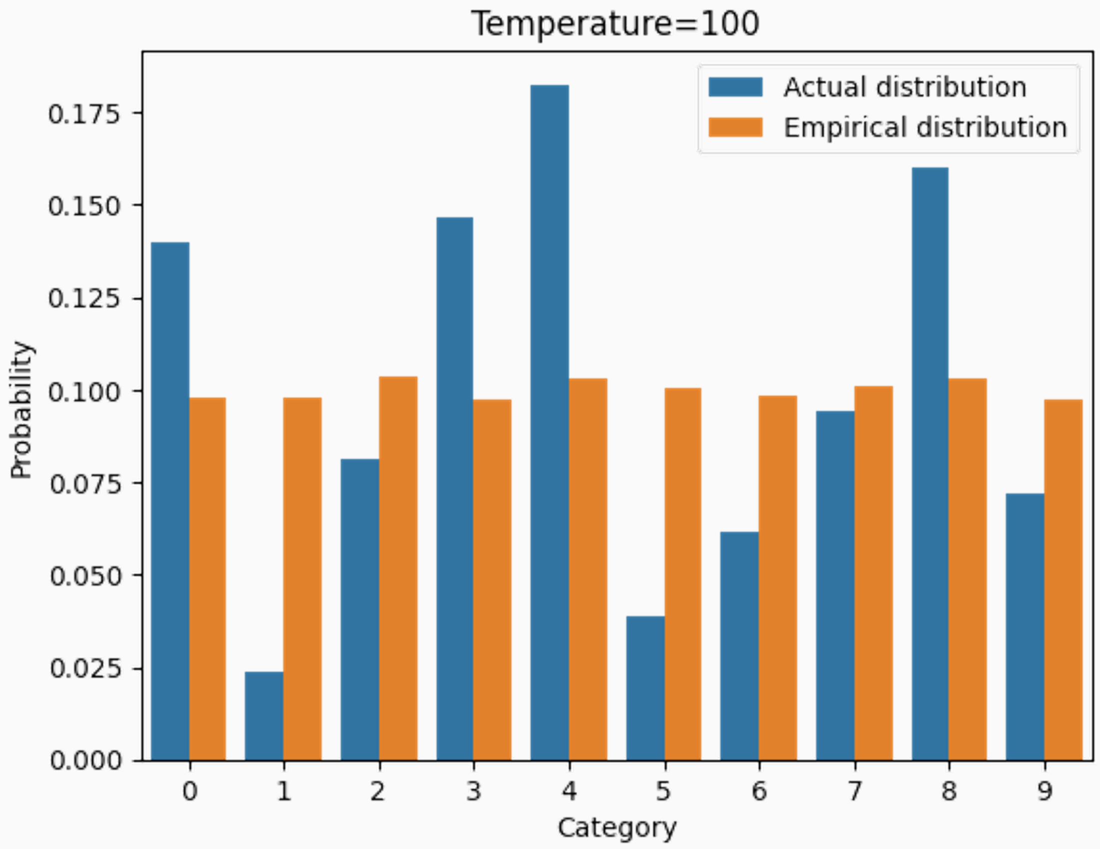
What does the temperature do? An example

An experiment:

1. Randomly create a set of scores over ten categories

2. Sample 10k times for different values of temperature

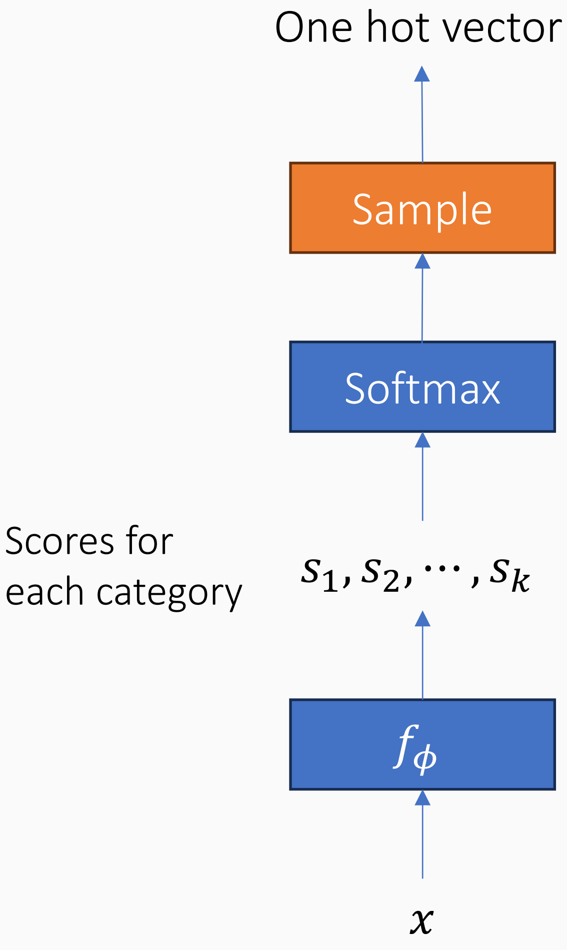
3. Compute empirical distribution of samples and compare to softmax of the scores



When the temperature is increased to 100, the empirical distribution is nearly uniform

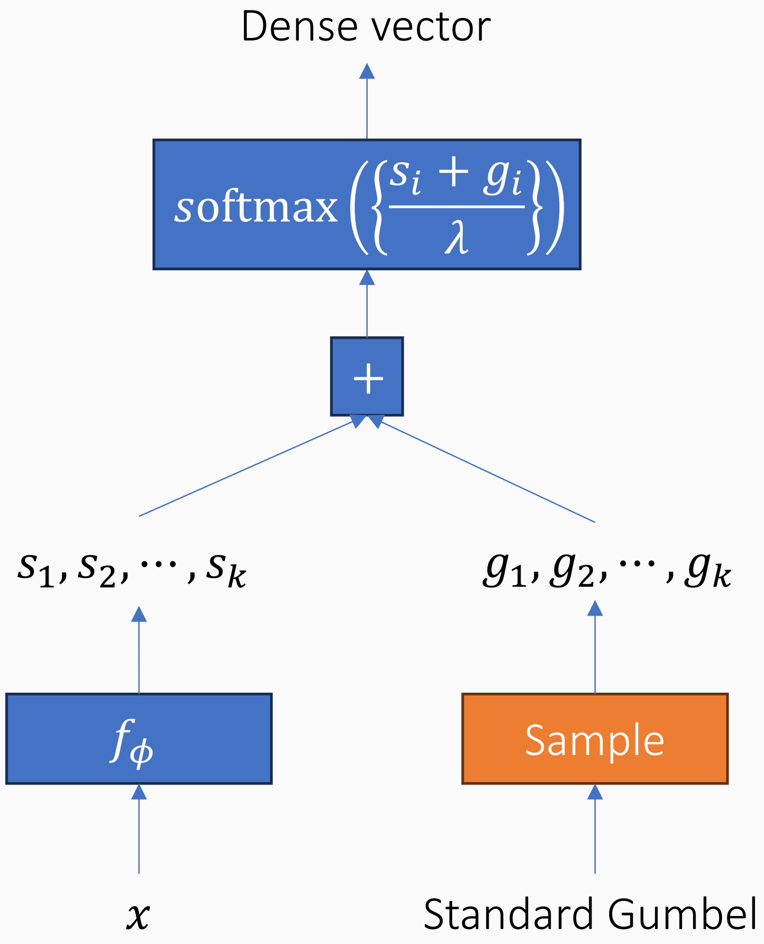
At high temperatures, the underlying scores are ignored

Using the Gumbel-softmax trick: Approach 1

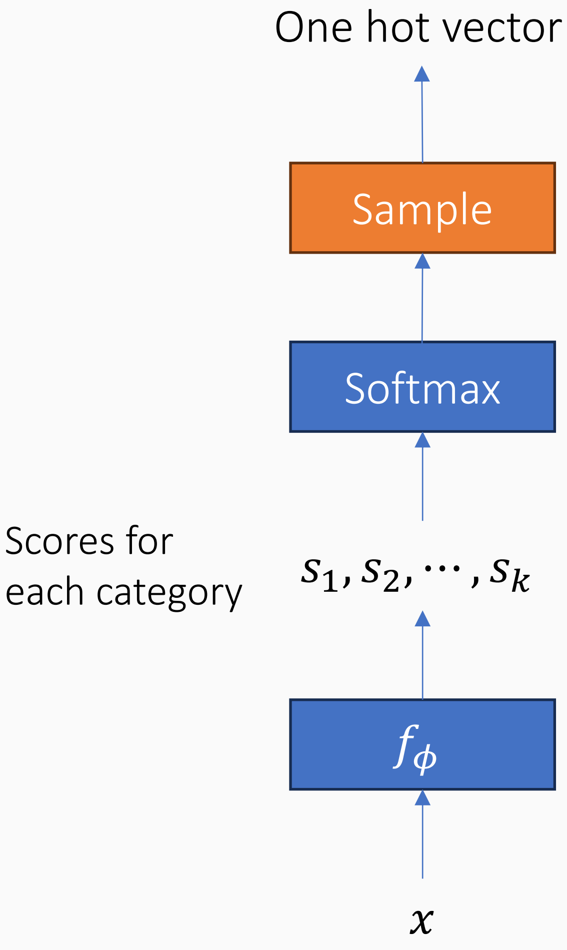


Training time: Replace with the relaxed reparameterized version

Test time: use the argmax (or the actual sampling)



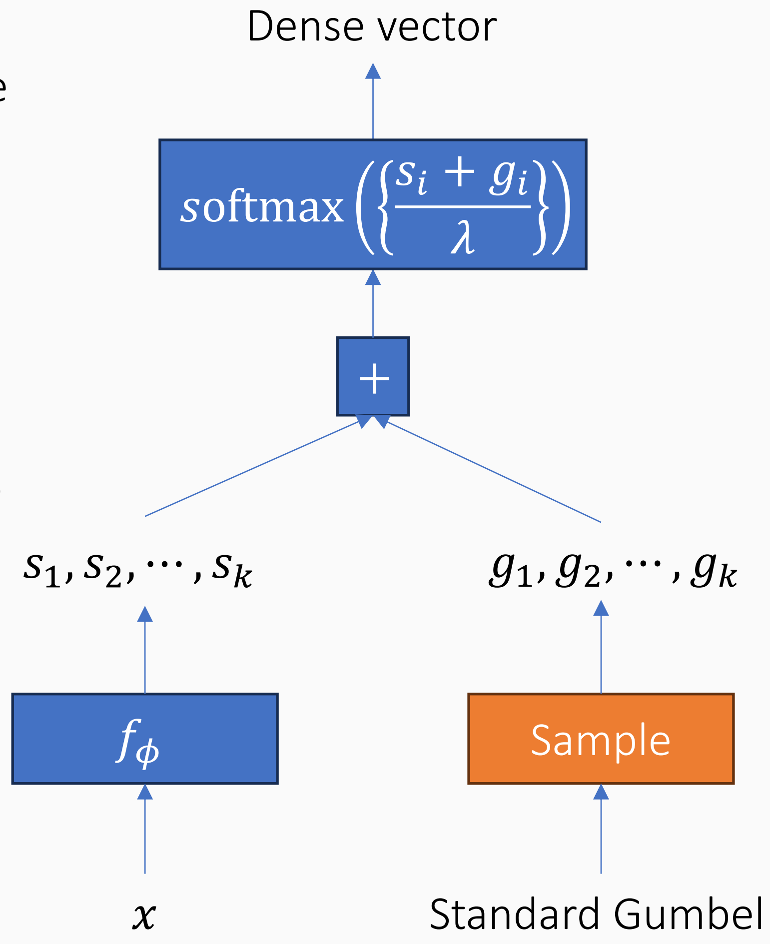
Using the Gumbel-softmax trick: Approach 2



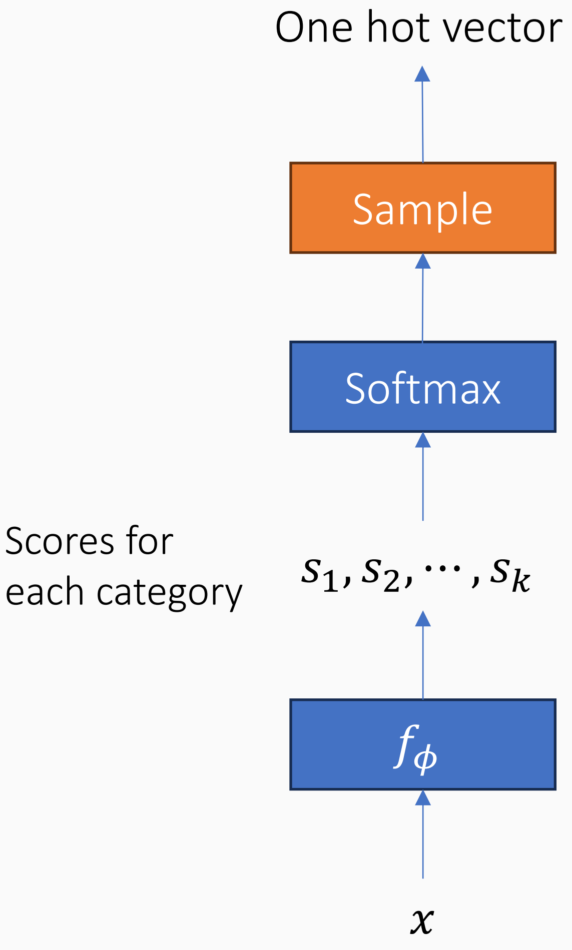
Training time (forward pass): Use the argmax version

Training time (backward pass): Use the relaxed reparameterized version

Test time: use the argmax (or the actual sampling)



Using the Gumbel-softmax trick: Approach 2

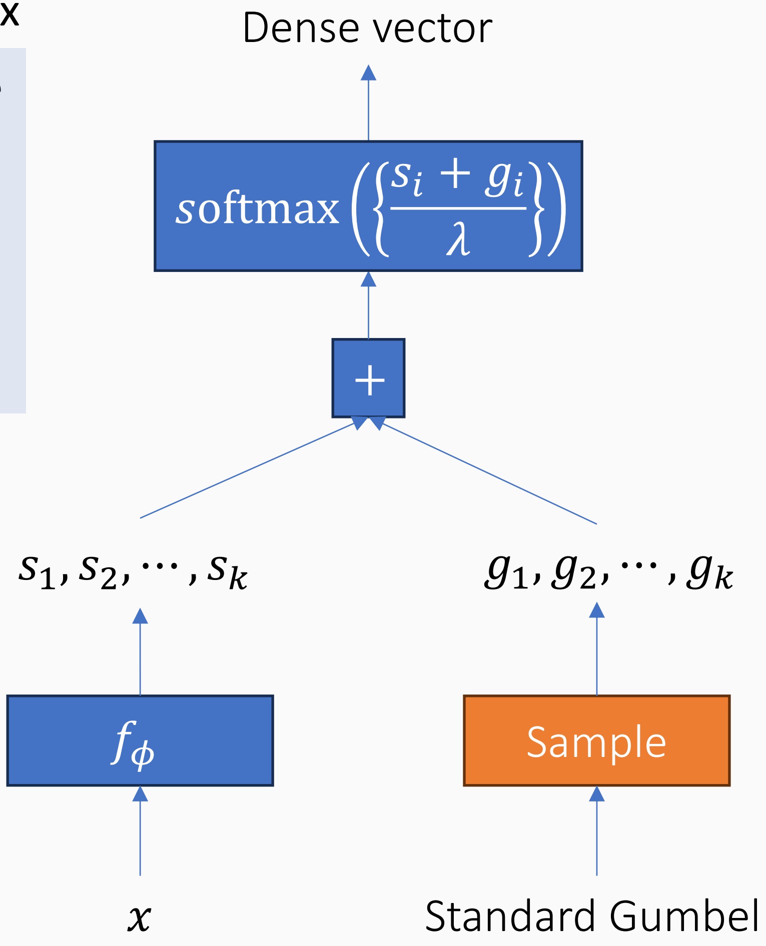


Straight-through Gumbel-softmax

Training time (forward pass): Use the argmax version

Training time (backward pass): Use the relaxed reparameterized version

Test time: use the argmax (or the actual sampling)



Gumbel-softmax: Takeaways

Helps train models that have a categorical sampling node in them

- Important: Does not have to be multiclass sampling, more complicated structures possible as well
- Any discrete distribution can be approximated with the Gumbel-max

An easy idea to incorporate in your code

- Can combine with the straight through estimator

The approach is sensitive to the choice of the softmax temperature

- The Concrete paper uses $\lambda = \frac{2}{3}$
- Another approach: Anneal the temperature from a high temperature to a low one while training proceeds