

Reparameterization of the categorical distribution

We will work with Torch throughout this notebook.

```
In [1]: import torch
from torch.distributions import Beta, Categorical #, ... import the distributions
from torch.nn import functional as F
import numpy as np
```

```
In [2]: torch.manual_seed(0)
```

```
Out[2]: <torch._C.Generator at 0x19a05c5e4f0>
```

A helper function to visualize the generated samples:

```
In [3]: import matplotlib.pyplot as plt
def compare_samples (samples_1, samples_2, bins=10, range=None, save=False):
    fig = plt.figure()
    if range is not None:
        plt.hist(samples_1, bins=bins, range=range)
        plt.hist(samples_2, bins=bins, range=range)
    else:
        plt.hist(samples_1, bins=bins)
        plt.hist(samples_2, bins=bins, alpha=0.7)
    plt.xlabel('value', size=20)
    plt.ylabel('number of samples', size=20)
    plt.legend(['direct','via reparameterization'])
    plt.tight_layout(pad=1.5)
    if save:
        plt.savefig("reparam_gumble.png")
    plt.show()
```

Categorical Distribution

Below write a function that generates N samples from Categorical (**a**), where **a** = $[a_0, a_1, a_2, a_3]$.

```
In [4]: def categorical_sampler(a, N):
    # insert your code
    dist = torch.distributions.Categorical(probs=a)
    samples = dist.sample((N,))
    return samples # should be size N
```

Now write a function that generates samples from Categorical (**a**) via reparameterization:

```
In [5]: # Hint: approximate the Categorical distribution with the Gumbel-Softmax distribution
def categorical_reparametrize(a, N, temp=0.1, eps=1e-20): # temp and eps are hyperparameters
    # insert your code
    K = a.shape[0]

    logits = torch.log(a + eps)
```

```

u = torch.rand(N, K, device=logits.device)
g = -torch.log(-torch.log(u + eps) + eps)
y = (logits + g) / temp
samples = F.softmax(y, dim=1)

return samples # make sure that your implementation allows the gradient to bac

```

Generate samples when $a = [0.1, 0.2, 0.5, 0.2]$ and visualize them:

```
In [6]: a = torch.tensor([0.1,0.2,0.5,0.2])
N = 1000
direct_samples = categorical_sampler(a, N)
reparametrized_samples = categorical_reparametrize(a, N, temp=0.1, eps=1e-20) # hard_samples = torch.argmax(reparametrized_samples, dim=1)      # convert repara
compare_samples(direct_samples, hard_samples, bins=4, save=True)
```

