JointDistributions in TensorFlow Probability

Dan Piponi, Dave Moore, Christopher Suter, Junpeng Lao, Joshua V. Dillon

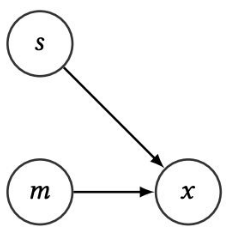
Google Research

`JointDistribution`s represent directed graphical models in TensorFlow Probability. They:

- Extend and generalize the interface of univariate **`Distribution`**s.
- Provide a shared representation for both sampling and log prob queries.
- Abstract multiple flavors of model specification behind a common interface.
- Support vectorized sampling and inference.

You can use them today to build models and run inference at scale, in TensorFlow or JAX.

The code matches the math.



$$s \sim \mathrm{InverseGamma}(3, 2)$$
$$m \sim \mathrm{Normal}(0, 1)$$
$$x \sim \mathrm{Normal}(m, s)$$

```
simple_model = tfd.JointDistributionSequential([
    tfd.InverseGamma(3., 2.),        # s
    tfd.Normal(0., 1.),              # m
    lambda m, s: tfd.Normal(m, s),   # x
])
# Samples are tuples of `Tensor`s.
s, m, x = simple_model.sample()
```

Different specifications, same statistical model.

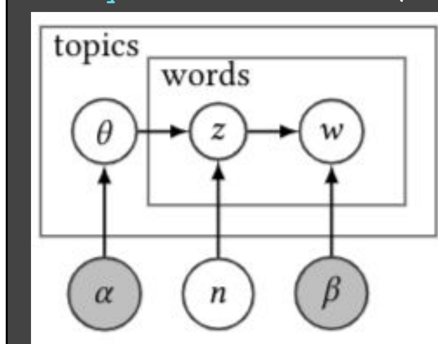
```
named_model = tfd.JointDistributionNamed(dict(
    s = tfd.InverseGamma(3., 2.),
    m = tfd.Normal(0., 1.),
    x = lambda m, s: tfd.Normal(m, s),
))
sample_dict = named_model.sample()  #  ==> {'s'=..., 'm'=..., 'x'=...}

# Coroutine (most 'PPL-like') flavor.
def model():
  s = yield tfd.InverseGamma(3., 2.)
  m = yield tfd.Normal(0., 1.)
  x = yield tfd.Normal(m, s)
coroutine_model = tfd.JointDistributionCoroutineAutoBatched(model)
s, m, x = coroutine_model.sample()  # a tuple
```

A unified interface.

```
# Draw a prior sample and evaluate its log density.
s, m, x = simple_model.sample()
simple_model.log_prob(s, m, x)

# Draw predictive samples given known `s`.
_, m, x = simple_model.sample(sample_shape=[100], s=2.0)

# Inspect conditional distributions.
(_, m_dist, x_dist), _ = (
     simple_model.sample_distributions(s=2.0))
```

Complicated things are simple.

```
alpha = tfp.util.TransformedVariable(
      init_alpha, tfb.Softplus())
beta = tf.Variable(init_beta)

@tfd.JointDistributionCoroutineAutoBatched
def latent_dirichlet_allocation():
  n = yield tfd.Poisson(rate=avg_doc_length)
  theta = yield tfd.Dirichlet(concentration=alpha)
  z = yield tfd.Multinomial(total_count=n, probs=theta)
  w = yield tfd.Multinomial(total_count=z, logits=beta)
```



Additional features

- **Nesting is supported**: can define distributions over arbitrary nested data structures.
- **AutoBatched variants transparently apply vectorized_map (TF) or vmap (JAX)** so that drawing multiple samples or evaluating log-densities in parallel 'just works'.
- **NEW**: bijectors can **Split** a vector-valued distribution, like a trainable flow, into a joint distribution over multiple RVs.

Discussion

- JDs deliberately focus on deterministic control flow, for easy vectorization.
- JD models may refer to trainable parameters as `tf.Variable`s, as in our LDA example. Variables are automatically tracked and may be accessed as `alpha, beta = lda.trainable_variables`
- Most TFP inference APIs take a callable specifying a `target_log_prob_fn`; joint distribution methods integrate seamlessly. TFP also provides utilities to generate fully-factorized or structured variational distributions from joint distribution models.
- Like most of TFP, joint distributions are supported in both Tensorflow and JAX backends:
  `import tensorflow_probability as tfp`
  or
  `from tensorflow_probability.substrates import jax as tfp`

Contact

https://www.tensorflow.org/probability/
Reach out to us on our Google group with questions or feedback: tfprobability@tensorflow.org