

HAW HAMBURG
INFORMATIK MASTER

GRUNDPROJEKT

TensorFlow Probability

Evaluation der Bibliothek für
probabilistische und statistische Analysen

Bearbeiter:

Tom Schöner (2182801)

Betreuung:

Prof. Dr. Olaf Zukunft

11. April 2019

Inhaltsverzeichnis

1	Abstract	3
2	Tensorflow Probability Komponenten	3
2.1	Layers	3
2.1.1	Layer 0: Tensorflow	4
2.1.2	Layer 1: Statistical Building Blocks	4
2.1.3	Layer 2: Model Building	5
2.1.4	Layer 3: Probabilistic Inference	6
3	Pragmatik und Semantik	7
4	Integration in Tensorflow	7
5	Beispiel: Korrelation von Luftverschmutzung und Temperatur	7
6	Fazit	7

Abbildungsverzeichnis

1	$X \sim \mathcal{N}(0, 1)$ mit 2500 Samples	6
---	---	---

1 Abstract

Die auf Tensorflow basierende Bibliothek Tensorflow Probability¹ - fortan mit *TFP* abgekürzt - erweitert das Framework um eine probabilistische Komponente. Mittels einer breiten Masse an bereitgestellten Tools, wie statistischen Verteilungen, Sampling oder verschiedenster probabilistischer Erweiterungen für neuronale Netze, können einfache bis hin zu komplexen Modellen erstellt werden. Berechnungen werden, wie man es aus Tensorflow gewohnt ist, durch *Dataflow Graphs*² abgebildet. Auf die verschiedenen Funktionsweisen und Schichten von TFP wird in Abschnitt 2 detaillierter eingegangen.

In dieser Evaluation soll die Bibliothek auf ihre Semantik und Pragmatik, Effektivität beim Erstellen von statistischen Modellen und Integration in das Framework Tensorflow untersucht werden. Das maschinelle Lernen mit Hilfe von neuronalen Netzen und deren Abstraktion durch Keras ist hierbei als Schwerpunkt anzusehen.

2 Tensorflow Probability Komponenten

2.1 Layers

Die Struktur von TFP lässt sich, wie aus der Dokumentation zu entnehmen ist³, in die folgenden vier Schichten einteilen. Die Schichten bauen hierarchisch aufeinander auf, abstrahieren die unterliegenden Schichten aber nicht zwangsläufig. Möchte man beispielsweise durch *MCMC* in Schicht 3 Parameter seines probabilistischen Modells mittels Sampling ermitteln, sollten *Bijectors* aus Schicht 1 kein Fremdwort sein.

¹<https://www.tensorflow.org/probability>

²<https://www.tensorflow.org/guide/graphs>

³<https://www.tensorflow.org/probability/overview>

2.1.1 Layer 0: Tensorflow

TFP ist nicht als eigenständige Komponente neben Tensorflow anzusehen, sondern als Bestandteil dessen. Die probabilistischen Berechnungen werden mit demselben Berechnungsmodell mittels Tensorflow Sessions oder im *Eager*-Modus für sofortige Berechnungen ausgeführt. Tensorflow wird von mehreren Programmiersprachen wie Python, JavaScript oder C++ unterstützt. Die Bibliothek TFP ist aktuell nur für die primär unterstützte Programmiersprache Python implementiert.

2.1.2 Layer 1: Statistical Building Blocks

Als Fundament statistischer Modelle sind mehrere, in Python Module aufgeteilte, Klassen und Funktionen gegeben. Diese können in der API Dokumentation der TFP Website eingesehen werden. Ein Beispiel hierfür ist das Modul **fp.stats**. Unter **fp.stats** finden sich unter Anderem Funktionen für die Berechnung für Korrelationen, Quantilen oder Standardabweichungen. Diese Funktionen sind auf die Verwendung von Tensoren ausgelegt, lassen sich im Normalfall aber auch mit normalen n-dimensionalen Python Arrays oder Numpy Arrays aufrufen.

Verschiedenste, für probabilistische Modelle essentielle Verteilungen reihen sich unter dem Modul **tfp.distributions** in dieser Schicht ein: *Normal*-, *Bernoulli*-, *Exponential*- oder *Gammaverteilung*, um einige zu nennen. Generell erben Klassen für Verteilungen (wie etwa die Normalverteilung) von der Klasse **tfp.distributions.Distribution**⁴. Diese gemeinsame Schnittstelle vereinfacht die Benutzung und fordert eine Implementation für hilfreiche Methoden wie für logarithmische Wahrscheinlichkeit oder Sampling. Am Beispiel der Normalverteilung mit der Definition $X \sim \mathcal{N}(\mu, \sigma^2)$ lässt sich die Schnittstelle für Verteilungen verdeutlichen (siehe Listing 1). Visualisiert man die Verteilung auf Basis der Samples ergibt sich Abbildung 1.

```
1 normal_dist = tfd.Normal(name="N", loc=0., scale=1.)
```

⁴https://www.tensorflow.org/probability/api_docs/python/tfp/distributions/Distribution

```

2 # tfp.distributions.Normal("N/", batch_shape=(), event_shape=(),
   dtype=float32)
3
4 sample = normal_dist.sample(sample_shape=normal_sample_size)
5 # <tf.Tensor: id=188, shape=(2500,), dtype=float32, numpy=array
   ([ -0.45733708, -0.19126031, -0.33290815, ..., -1.1285563 ,
     -0.6958163 ,  0.552399 ], dtype=float32)>
6
7 normal_dist.mean()
8 # tf.Tensor(0.0, shape=(), dtype=float32)
9
10 normal_dist.prob(1.0)
11 # tf.Tensor(0.24197072, shape=(), dtype=float32)

```

Listing 1: Verwendung der Klasse `tfp.distributions.Normal`

Broadcasting, Batching und Shapes sorgen dafür, dass unabhängige Verteilungen als Batch in einer Entität gekapselt werden können. Um beispielsweise ein zweidimensionales Batch für die Normalverteilung aus Listing 1 zu erzeugen, kann als Erwartungswert μ durch `loc=[0., 10.]` übergeben werden. Da TFP hier ebenfalls broadcasting unterstützt müssen die folgenden Aufrufe nicht angepasst werden, nur das Ergebnis wird ebenfalls Zweidimensional sein. Broadcasting verhält sich analog zu dem in Numpy etablierten Konzept⁵.

Ein weiterer Bestandteil von Layer 1 sind *Bijectors*. Bijectors bilden eine Zahl aus \mathbb{R}^n auf \mathbb{R}^m ab oder jeweils einer Submenge dieser. Aufgrund der im Bijector hinterlegten bijektiven Funktion ist dieser Schritt umkehrbar, daher: $x = f^{-1}(f(x))$. Dies ist besonders bei der Transformierung von Samples aus Verteilungen nützlich und wird folglich für das Erstellen von Modellen (Abschnitt 2.1.3) und Berechnungen probabilistischer Interferenzen (Abschnitt 2.1.4) eingesetzt. TFP bietet bereits einige vorgefertigte Bijectors unter `tfp.bijectors`⁶ an.

2.1.3 Layer 2: Model Building

Edward2 / Probabilistic Layers with Keras / Trainable Distributions

⁵<https://docs.scipy.org/doc/numpy/user/basics.broadcasting.html>

⁶https://www.tensorflow.org/probability/api_docs/python/tfp/bijectors

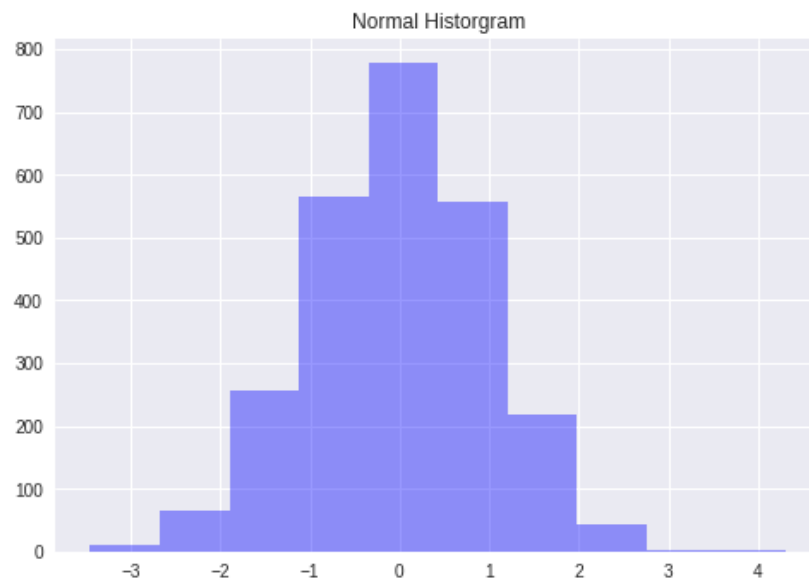


Abbildung 1: $X \sim \mathcal{N}(0, 1)$ mit 2500 Samples

2.1.4 Layer 3: Probabilistic Inference

MCMC / VI / Optimizers

3 Pragmatik und Semantik

4 Integration in Tensorflow

5 Beispiel: Korrelation von Luftverschmutzung und Temperatur

Das Jupyter Notebook ist unter https://github.com/tom-schoener/ml-probability/blob/master/tfp-evaluation/notebooks/air_quality.ipynb einsehbar.

6 Fazit