Probabilistic Programming

im Proseminar Softwareentwicklung in der Wissenschaft

Katja T. J. Neumann

Juni 2020

Gliederung

Wofür benötigen wir Probabilistic Programming?

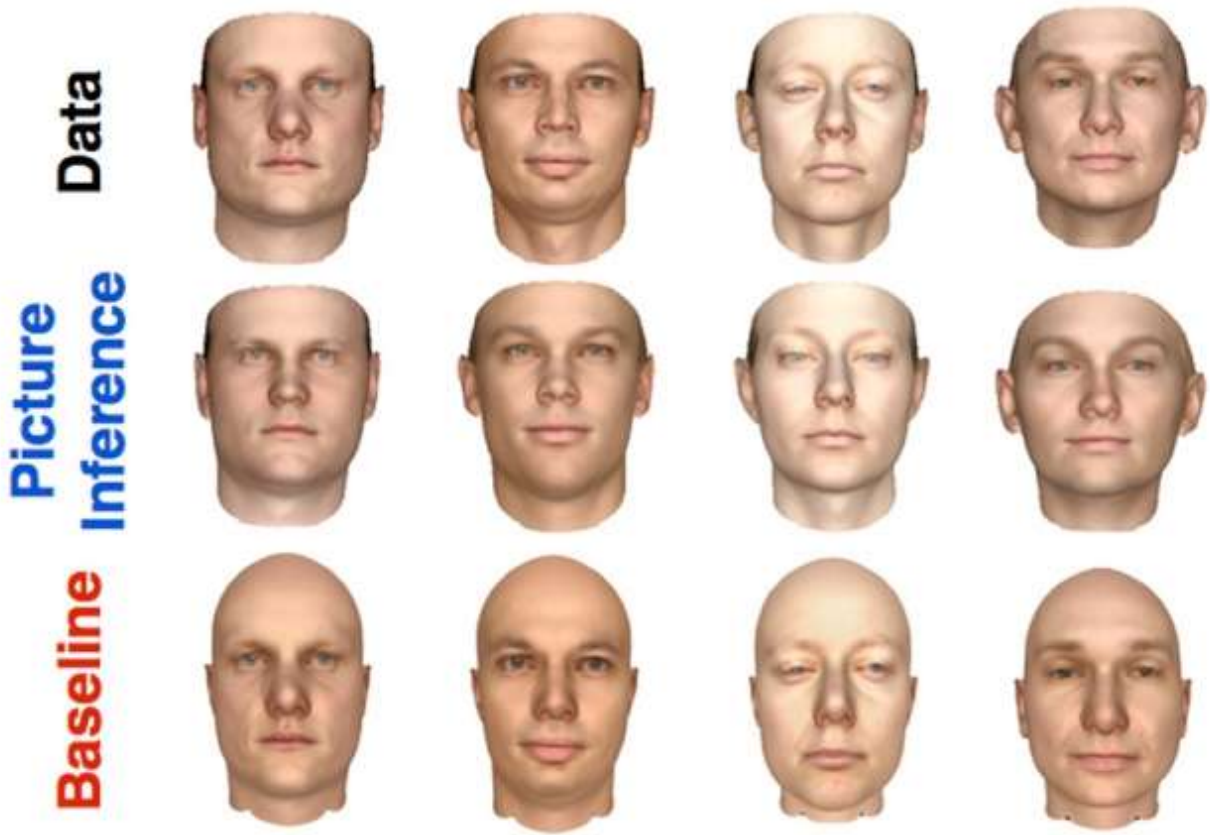
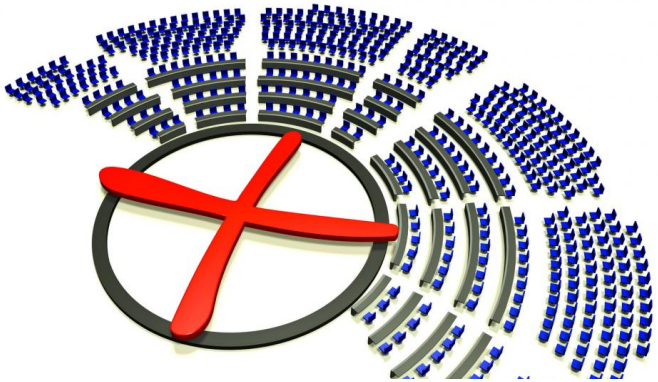


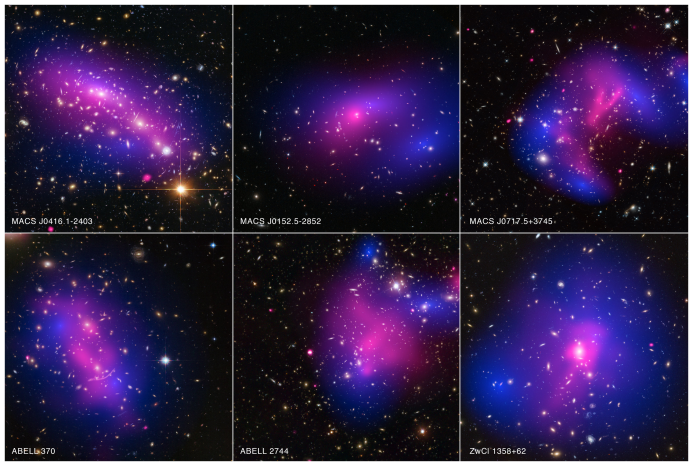
Abbildung: MIT

Wofür benötigen wir Probabilistic Programming?



Wahlvorhersagen

(Bild: Fotolia)



Suche nach Dunkler Materie

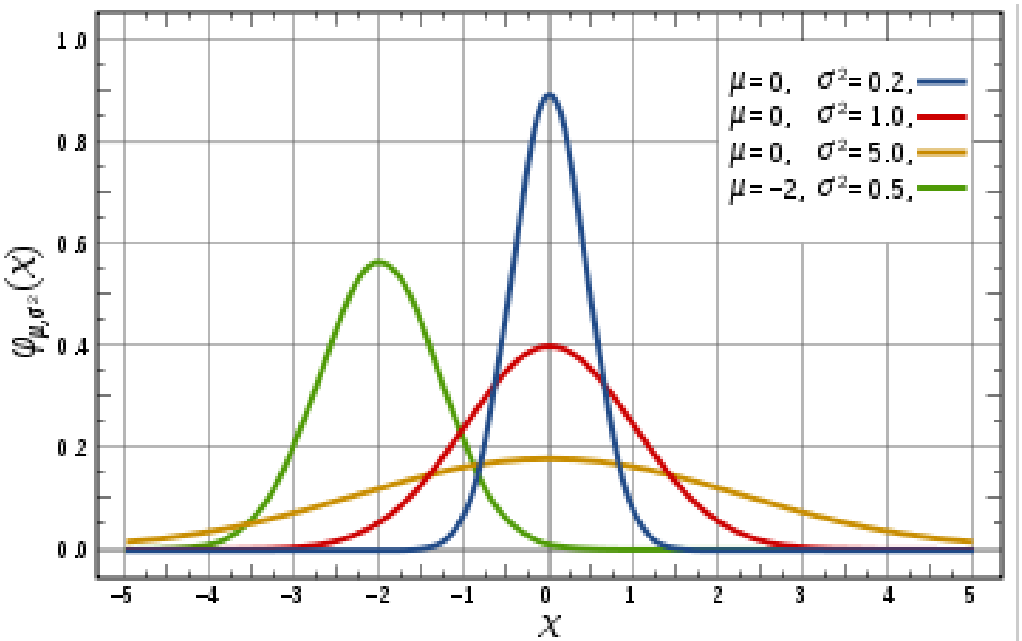
(Bild: Scitech Daily)

Anwendungsfelder

- Computer Vision
- Klima- und Wettermodelle
- Börsenkurse vorhersagen
- Genetische Analysen
- Pharmazeutik
- Matching: Stellenausschreibungen und Jobsuchende, Werbevorschläge beim Einkaufen
- ...

Was ist Probabilistic Programming?

- Wahrscheinlichkeitsmodelle...
- ...an denen wir Inferenz betreiben wollen

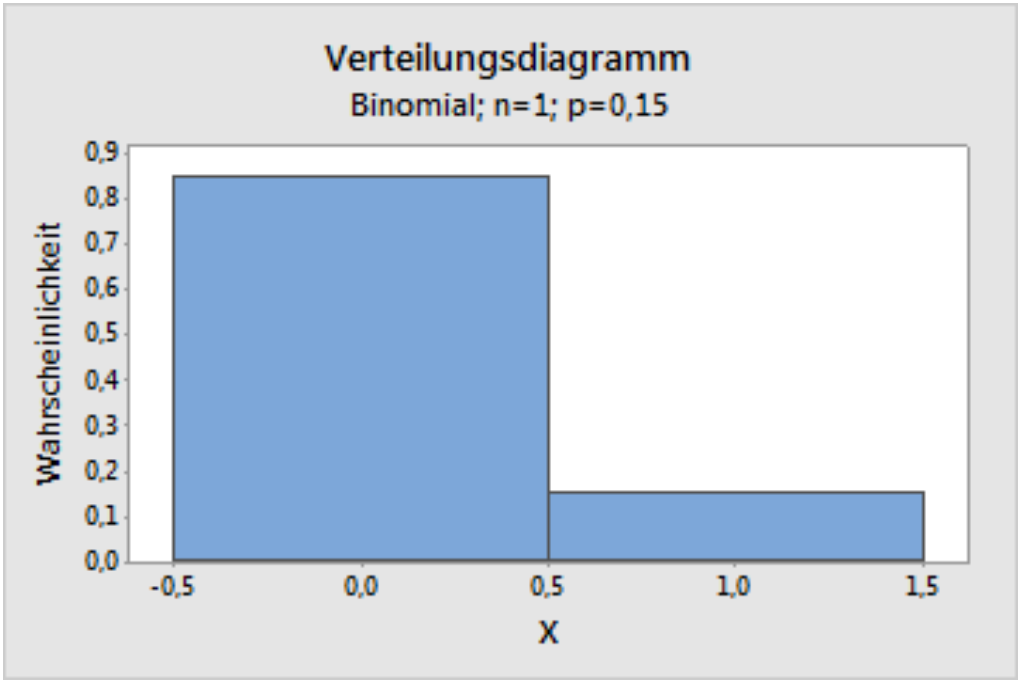


(a) Gauß'sche Normalverteilung

(Bild: phpgangsta.de)

Was ist Probabilistic Programming?

- Wahrscheinlichkeitsmodelle...
- ...an denen wir Inferenz betreiben wollen

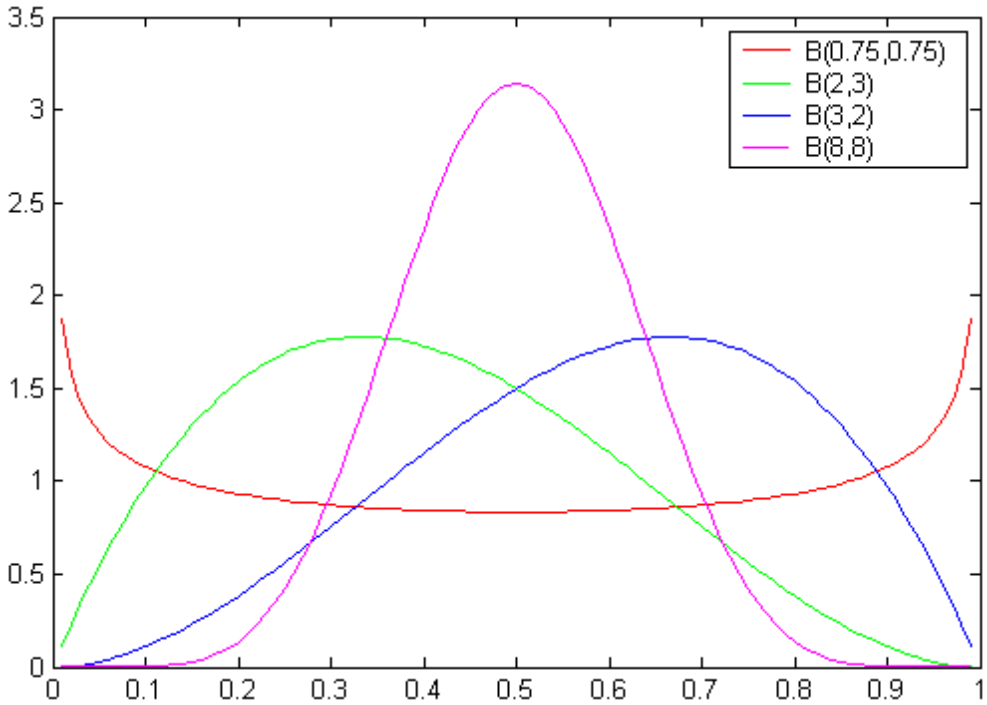


(b) Bernoulli-Verteilung

(Bild: minitab.com)

Was ist Probabilistic Programming?

- Wahrscheinlichkeitsmodelle...
- ...an denen wir Inferenz betreiben wollen



(c) Beta-Verteilung

(Bild: statsref.com)

Was ist Probabilistic Programming?

- Herausforderung und Stärke: Eine Vielzahl verknüpfter Wahrscheinlichkeitsmodelle
- Ein noch recht junges Feld
- Wichtigste Entwicklungen: Generalisierung der Inferenzalgorithmen und Optimierung für große Datenmengen
- Spezieller, aber hoch bedeutsamer Anwendungsbereich
- Grundlage und Erleichterung für Machine Learning
- Grundlage und Erleichterung für Datenanalysen

Ziel

- Wir beschreiben unsere Daten
- Das Programm übernimmt den mathematischen Hintergrund
- ...indem es die Inferenz Algorithmen überlässt

Grundlagen der Modellierung

- Bayes'sches Wahrscheinlichkeitsverständnis $\neq$ Frequentistisches Wahrscheinlichkeitsverständnis
- Satz von Bayes: $P(A|X) = \frac{P(X|A)P(A)}{P(X)}$
- Probabilistic Program arbeitet in zwei Richtungen:
  - Berechnet aus den Daten die Wahrscheinlichkeitsverteilung mit konkreten Werten
  - Berechnet aus den Daten mögliche Erklärungen
  - Beide beeinflussen einander
- Suche nach latenten Variablen: Eine nicht direkt beobachtbare Größe, der wir uns durch mathematische Berechnungen annähern

Was ist Pyro?

- Deep Probabilistic Programming Framework
- Baut auf Python und PyTorch auf
- Als Unterstützung und Grundlage für die KI-Forschung gedacht
- Vier grundlegende Prinzipien:
  - Expressivität
  - Skalierbarkeit
  - Flexibilität
  - Minimalität

Pyro: Primitive Typen

- `pyro.sample`: Konkreter Wert, stammt aus einer benannten Verteilung
- `pyro.param`: Parameter für Inferenzalgorithmen
- `model`: Annäherung an die a priori Verteilung
- `guide`: Annäherung an die posteriore Verteilung
- `model` und `guide` nehmen dieselben Parameter auf

Pyro: Codebeispiel

Modellierung der a priori-Wahrscheinlichkeitsverteilung

```
1  def model(data):
2          alpha0 = torch.tensor(10.0)
3          beta0 = torch.tensor(10.0)
4           f = pyro.sample("latent_fairness",
               ↪ dist.Beta(alpha0, beta0))
5           for i in range(len(data)):
6           pyro.sample("obs_{}".format(i),
               ↪ dist.Bernoulli(f), obs=data[i])
```

Quelle: http://pyro.ai/examples/svi_part_i.html

Pyro: Codebeispiel

Erste Modellierung der posterioren Wahrscheinlichkeitsverteilung

```
1 def guide(data):
2     alpha_q = pyro.param("alpha_q",
          ↪ torch.tensor(15.0),
          ↪ constraint=constraints.positive)
3     beta_q = pyro.param("beta_q", torch.tensor(15.0),
          ↪ constraint=constraints.positive)
4     pyro.sample("latent_fairness", dist.Beta(alpha_q,
          ↪ beta_q))
```

Quelle: http://pyro.ai/examples/svi_part_i.html

Pyro: Codebeispiel

Vorbereiten und Durchführen des Inferenzprozesses

```
1 adam_params = {"lr": 0.0005, "betas": (0.90, 0.999)}
2 optimizer = Adam(adam_params)
3 svi = SVI(model, guide, optimizer, loss=Trace_ELBO())
4
5 n_steps = 2000
6 for step in range(n_steps):
7         svi.step(data)
```

Quelle: http://pyro.ai/examples/svi_part_i.html

Pyro: Codebeispiel

Aufbereiten der Ergebnisse

```
1 alpha_q = pyro.param("alpha_q").item()
2 beta_q = pyro.param("beta_q").item()
3 inferred_mean = alpha_q / (alpha_q + beta_q)
4 factor = beta_q / (alpha_q * (1.0 + alpha_q + beta_q))
5 inferred_std = inferred_mean * math.sqrt(factor)
6 print("\nbased on the data and our prior belief, the
    ↪ fairness " + "of the coin is %.3f +- %.3f" %
    ↪ (inferred_mean, inferred_std))
```

Quelle: http://pyro.ai/examples/svi_part_i.html

Pyro: Codebeispiel

Beispiele für Ergebnisse:

```
the fairness of the coin is 0.537 +- 0.089
the fairness of the coin is 0.533 +- 0.090
the fairness of the coin is 0.538 +- 0.090
the fairness of the coin is 0.532 +- 0.089
the fairness of the coin is 0.533 +- 0.090
```

Programmiersprachen und Frameworks

- Sprachen
  - Stan
  - Julia
  - Edward
  - ...
- Frameworks
  - Pyro (Python)
  - PyMC3 (Python)
  - webppl (JavaScript)
  - Infer.NET (.NET Sprachen wie zB C#, Java)
  - ...

Zusammenfassung

- Was spricht für/gegen Pyro?
  - Sehr expressiv
  - Aktuell stark in der Entwicklung, daher stetig Neuerungen und Verbesserungen (zB NumPyro)
  - Nur vereinzelt ausführliche Erklärungen und Codebeispiele zu finden
- Hält Probabilistic Programming sein Versprechen ein?
  - Statistisches Verständnis weiterhin nötig, um ein Modell zu erstellen
  - Benutzer müssen jedoch die Inferenzalgorithmen nicht mehr im Einzelnen beherrschen und implementieren

Literatur

- http://pyro.ai/examples/index.html (Pyro Documentation)
- http://pyro.ai/examples/svi_part_i.html (Codebeispiel)
- http://camdavidsonpilon.github.io/Probabilistic-Programming-and-Bayesian-Methods-for-Hackers/ (Empfehlung/auch als Buch in unserer Inf-Bib)

Literatur

- https://phys.org/news/2015-04-probabilistic-lines-code-thousands.html
- http://hci-kdd.org/wordpress/wp-content/uploads/2016/10/GORDON-HENZINGER-NORI-RAJAMANI-2014-Probabilistic-Programming.pdf (Grundlagen)
- http://www.columbia.edu/ jwp2128/Papers/HoffmanBleiWangPaisley2013.pdf (SVI)
- http://hirzels.com/martin/papers/arxiv18-deep-ppl.pdf (Vergleich Edward/Pyro)