CS492: Probabilistic Programming

Amortised Inference

Hongseok Yang
KAIST

```
(defquery biased-coin []
  (let [r (sample
              (uniform-continuous 0 1))
        a (observe (flip r) true)
        b (observe (flip r) true)
        c (observe (flip r) false)]
    r))
```

```
(defquery biased-coin []
  (let [r (sample
             (uniform-continuous 0 1))
        a (observe (flip r) true)
        b (observe (flip r) true)
        c (observe (flip r) false)]
    r))
```

```
(defquery biased-coin []
  (let [r (sample
             (uniform-continuous 0 1))
        a (observe (flip r) true)
        b (observe (flip r) true)
        c (observe (flip r) false)]
   r))
```

```
(defquery biased-coin []
  (let [r (sample
               (uniform-continuous 0 1))
        a (observe (flip r) true)
        b (observe (flip r) true)
        c (observe (flip r) false)]
    r))
```

```
(defquery biased-coin []
  (let [r (sample
            (uniform-continuous 0 1))
        a (observe (flip r) true)
        b (observe (flip r) true)
        c (observe (flip r) false)]
    r))
```

Importance sampling with proposal $q(r)$.

1. Generate $(w_1,r_1), \ldots, (w_n,r_n)$ by running prog.

2. Estimate $\mathbb{E}_{p(r|a,b,c)}[f(r)] \approx \sum_i f(r_i)*(w_i/\sum_j w_j)$.

```
(defquery biased-coin []
  (let [r (sample
             (uniform-continuous 0 1))
        a (observe (flip r) true)
        b (observe (flip r) true)
        c (observe (flip r) false)]
   r))
```

Importance sampling with proposal $q(r)$.

1. Generate $(w_1,r_1), \ldots, (w_n,r_n)$ by running prog.

2. Estimate $\mathbb{E}_{p(r|a,b,c)}[f(r)] \approx \sum_i f(r_i)*(w_i/\sum_j w_j)$.

```
(defquery biased-coin []
  (let [r (sample
               (uniform-continuous 0 1))
        a (observe (flip r) true)
        b (observe (flip r) true)
        c (observe (flip r) false)]
   r))
```

$w_1 = 1$

Importance sampling with proposal q(r).

1. Generate $(w_1,r_1), \ldots, (w_n,r_n)$ by running prog.

2. Estimate $\mathbb{E}_{p(r|a,b,c)}[f(r)] \approx \sum_i f(r_i)*(w_i/\sum_j w_j)$.

```
(defquery biased-coin []
  (let [r (sample
            (uniform-continuous 0 1))
        a (observe (flip r) true)
        b (observe (flip r) true)
        c (observe (flip r) false)]
   r))
```

$w_1 = 1$

Importance sampling with proposal $q(r)$.

1. Generate $(w_1,r_1), \ldots, (w_n,r_n)$ by running prog.

2. Estimate $\mathbb{E}_{p(r|a,b,c)}[f(r)] \approx \sum_i f(r_i) * (w_i / \sum_j w_j)$.

```
(defquery biased-coin []
  (let [r (sample
             (uniform-continuous 0 1))
        a (observe (flip r) true)
        b (observe (flip r) true)
        c (observe (flip r) false)]
    r))
```

$w_1 = 1 * p(.4)/q(.4)$

$r_1 = .4$

Importance sampling with proposal $q(r)$.

1. Generate $(w_1, r_1), \ldots, (w_n, r_n)$ by running prog.

2. Estimate $\mathbb{E}_{p(r|a,b,c)}[f(r)] \approx \sum_i f(r_i) * (w_i / \sum_j w_j)$.

```
(defquery biased-coin []
  (let [r (sample
             (uniform-continuous 0 1))
        a (observe (flip r) true)
        b (observe (flip r) true)
        c (observe (flip r) false)]
   r))
```

$w_1 = .096 * p(.4)/q(.4)$

$r_1 = .4$

Importance sampling with proposal $q(r)$.

1. Generate $(w_1,r_1), \ldots, (w_n,r_n)$ by running prog.

2. Estimate $\mathbb{E}_{p(r|a,b,c)}[f(r)] \approx \sum_i f(r_i)*(w_i/\sum_j w_j)$.

```
(defquery biased-coin []
  (let [r (sample
              (uniform-continuous 0 1))
        a (observe (flip r) true)
        b (observe (flip r) true)
        c (observe (flip r) false)]
    r))
```

$w_1 = .096 * p(.4)/q(.4)$
$r_1 = .4$

Importance sampling with proposal $q(r)$.

1. Generate $(w_1,r_1), \ldots, (w_n,r_n)$ by running prog.

2. Estimate $\mathbb{E}_{p(r|a,b,c)}[f(r)] \approx \sum_i f(r_i)*(w_i/\sum_j w_j)$.

```
(defquery biased-coin []
  (let [r (sample
            (uniform-continuous 0 1))
        a (observe (flip r) true)
        b (observe (flip r) true)
        c (observe (flip r) false)]
   r))
```

$w_1 = .096 * p(.4)/q(.4)$
$r_1 = .4$

$w_2 = .144 * p(.6)/q(.6)$
$r_2 = .6$

Importance sampling with proposal $q(r)$.

1. Generate $(w_1,r_1), \ldots, (w_n,r_n)$ by running prog.

2. Estimate $\mathbb{E}_{p(r|a,b,c)}[f(r)] \approx \sum_i f(r_i)*(w_i/\sum_j w_j)$.

```
(defquery biased-coin []
  (let [r (sample
              (uniform-continuous 0 1))
        a (observe (flip r) true)
        b (observe (flip r) true)
        c (observe (flip r) false)]
   r))
```

$w_1 = .096 * p(.4)/q(.4)$
$r_1 = .4$

$w_2 = .144 * p(.6)/q(.6)$
$r_2 = .6$

Importance sampling with proposal q(r).

1. Generate $(w_1,r_1), \ldots, (w_n,r_n)$ by running prog.

2. Estimate $\mathbb{E}_{p(r|a,b,c)}[f(r)] \approx \sum_i f(r_i)*(w_i/\sum_j w_j)$.

```
(defquery biased-coin []
  (let [r (sample
            (uniform-continuous 0 1))
        a (observe (flip r) true)
        b (observe (flip r) true)
        c (observe (flip r) false)]
   r))
```

$w_1 = .096 * p(.4)/q(.4)$
$r_1 = .4$

$w_2 = .144 * p(.6)/q(.6)$
$r_2 = .6$

Importance sampling with proposal q(r).

1. Generate $(w_1,r_1), \ldots, (w_n,r_n)$ by running prog.

2. Estimate $\mathbb{E}_{p(r|a,b,c)}[f(r)] \approx \sum_i f(r_i)*(w_i/\sum_j w_j)$.

How to find good q?

```
(defquery biased-coin []
  (let [r (sample
            (uniform-continuous 0 1))
        a (observe (flip r) true)
        b (observe (flip r) true)
        c (observe (flip r) false)]
   r))
```

$w_1 = .096 * p(.4)/q(.4)$
$r_1 = .4$

$w_2 = .144 * p(.6)/q(.6)$
$r_2 = .6$

Importance sampling with proposal q(r).

1. Generate $(w_1,r_1), \ldots, (w_n,r_n)$ by running prog.

2. Estimate $\mathbb{E}_{p(r|a,b,c)}[f(r)] \approx \sum_i f(r_i)*(w_i/\sum_j w_j)$.

How to find good q? Use amortised inference!

We often need to infer posterior of one model multiple times with different observations.

```
(defquery biased-coin []
  (let [r (sample (uniform-continuous 0 1))
        a (observe (flip r) true)
        b (observe (flip r) true)
        c (observe (flip r) false)]
   r))
```

We often need to infer posterior of one model multiple times with different observations.

```
(defquery biased-coin []
  (let [r (sample (uniform-continuous 0 1))
        a (observe (flip r) false)
        b (observe (flip r) false)
        c (observe (flip r) false)]
   r))
```

We often need to infer posterior of one model multiple times with different observations.

```
(defquery biased-coin []
  (let [r (sample (uniform-continuous 0 1))
          a (observe (flip r) true)
          b (observe (flip r) true)
          c (observe (flip r) true)]
    r))
```

We often need to infer posterior of one model multiple times with different observations.

```
(defquery biased-coin []
  (let [r (sample (uniform-continuous 0 1))
        a (observe (flip r) false)
        b (observe (flip r) false)
        c (observe (flip r) true)]
    r))
```

We often need to infer posterior of one model multiple times with different observations.

```
(defquery biased-coin []
  (let [r (sample (uniform-continuous 0 1))
        a (observe (flip r) false)
        b (observe (flip r) false)
        c (observe (flip r) true)]
    r))
```

Other examples:
Financial model,
captcha, brain, etc.

We often need to infer posterior of one model multiple times with different observations.

```
(defquery biased-coin []
  (let [r (sample (uniform-continuous 0 1))
        a (observe (flip r) false)
        b (observe (flip r) false)
        c (observe (flip r) true)]
    r))
```

Other examples: Financial model, captcha, brain, etc.

Amortised inference.

We often need to infer posterior of one model multiple times with different observations.

```
(defquery biased-coin []
  (let [r (sample (uniform-continuous 0 1))
        a (observe (flip r) false)
        b (observe (flip r) false)
        c (observe (flip r) true)]
    r))
```

Other examples: Financial model, captcha, brain, etc.

Amortised inference. 1) Learn a proposal q(x; y) parameterized by obs. y via preprocessing.

We often need to infer posterior of one model multiple times with different observations.

```
(defquery biased-coin []
  (let [r (sample (uniform-continuous 0 1))
        a (observe (flip r) false)
        b (observe (flip r) false)
        c (observe (flip r) true)]
   r))
```

Other examples: Financial model, captcha, brain, etc.

Amortised inference. 1) Learn a proposal $q(x; y)$ parameterized by obs. y via preprocessing. 2) Use $q(x;y_0)$ for any actual observation y_0 later.

We often need to infer posterior of one model multiple times with different observations.

```
(defquery biased-coin []
  (let [r (sample (uniform-continuous 0 1))
        a (observe (flip r) false)
        b (observe (flip r) false)
        c (observe (flip r) true)]
    r))
```

Other examples: Financial model, captcha, brain, etc.

Amortised inference. 1) Learn a proposal $q(x; y)$ parameterized by obs. y via preprocessing. 2) Use $q(x;y_0)$ for any actual observation y_0 later.

neural nets

We often need to infer posterior of one model multiple times with different observations.

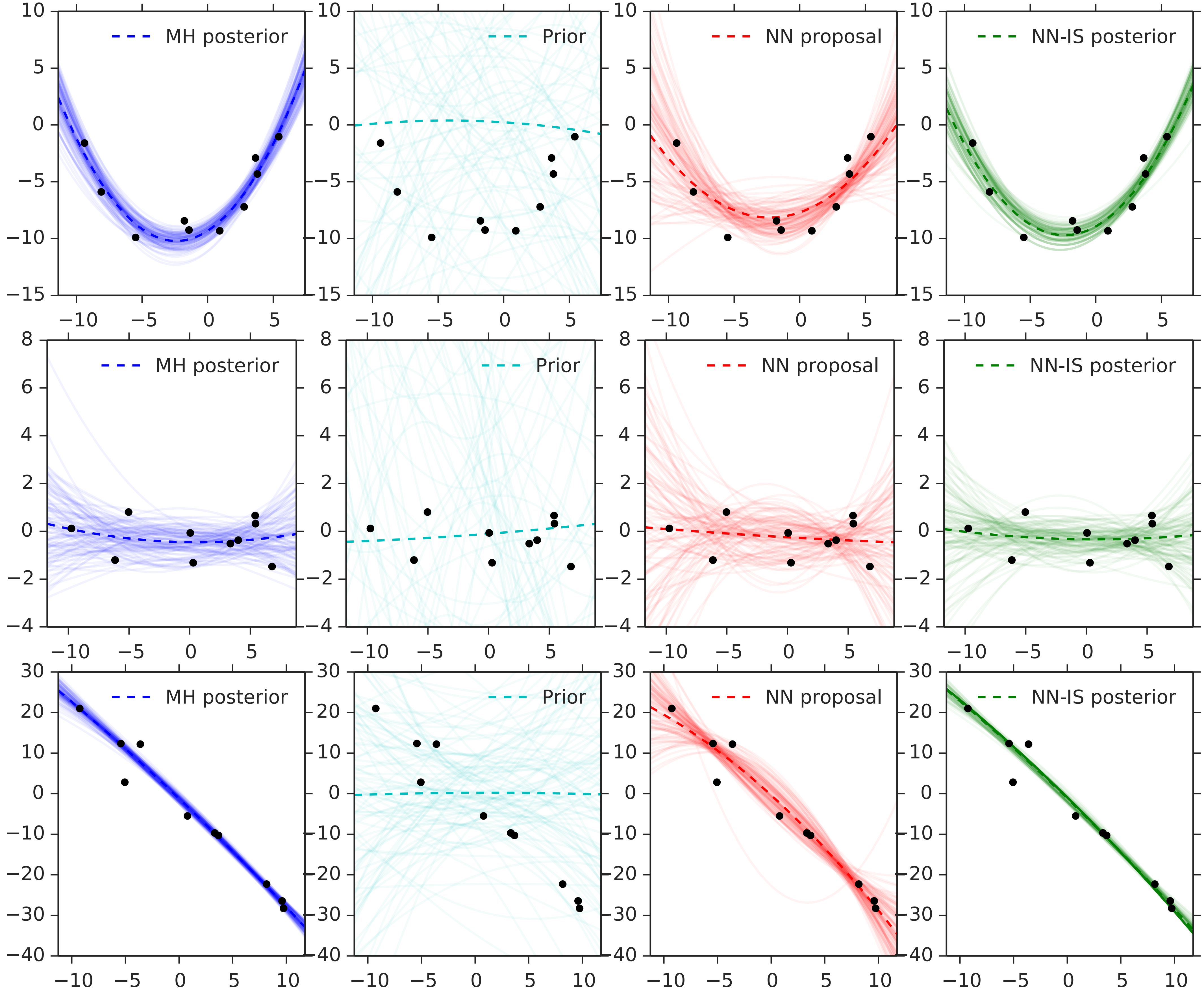
```
(defquery biased-coin []
  (let [r (sample (uniform-continuous 0 1))
        a (observe (flip r) false)
        b (observe (flip r) false)
        c (observe (flip r) true)]
```

Other examples: Financial model, captcha, brain, etc.

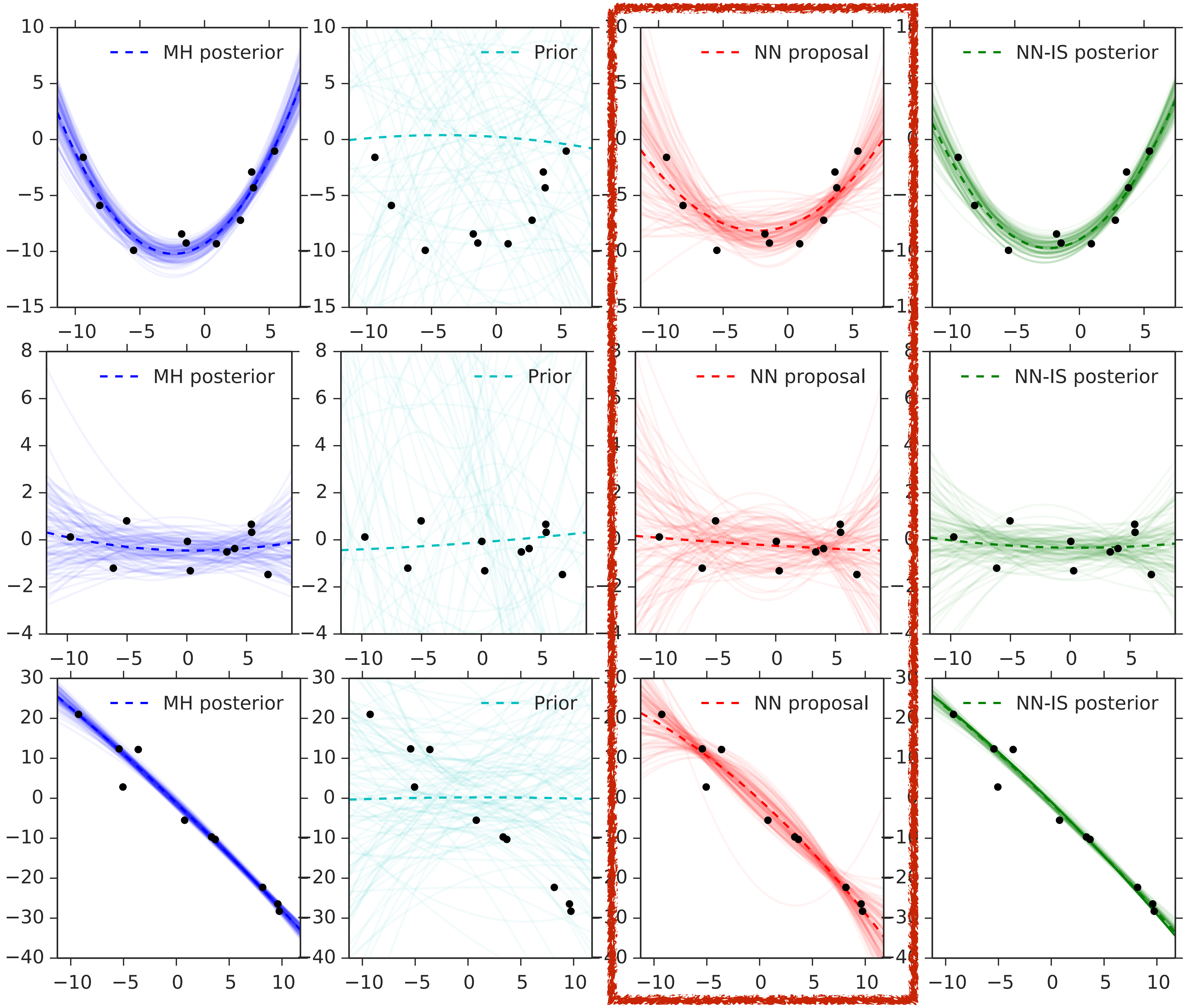
sample/object duality and reverse KL

Amortised inference. 1) Learn a proposal $q(x; y)$ parameterized by obs. y via preprocessing. 2) Use $q(x;y_0)$ for any actual observation y_0 later.

neural nets



Model for non-linear regression [Paige et al., ICML16]



Model for non-linear regression [Paige et al., ICML16]

Observed images

(W4kgvQ) (uV7FeWB) (MqhnpT)

more preprocessing

Samples

W4kgvQ	uV7EeWB	MqhnpT
WA4rjvQ	uV7FeWB	MypppT
Woxewd9	mTTEMMm	RIrpES
BKvu2Q	C9QDsoN	rS5FP2B

less preprocessing

Captcha solving [Le et al., AISTATS16]

Learning outcome

Can describe how amortised inference works for models written in math.

Can explain key ideas behind implementing amortised inference for probabilistic programs.

Proposal learning problem

Given:

1. joint dist. $p(x,y)$ for latent x and observed y,
2. IS proposal $q_\theta(x;y)$ parameterized by θ & y.

Find θ such that $q_\theta(x;y)$ is good for most y.

Proposal learning problem

Specified by $p(x)$ and $p(y|x)$.
Interested in $p(x|y)$.
But specific y not given yet.

Given:

1. joint dist. $p(x,y)$ for latent x and observed y,
2. IS proposal $q_\theta(x;y)$ parameterized by θ & y.

Find θ such that $q_\theta(x;y)$ is good for most y.

Proposal learning problem

Given:

1. joint dist. $p(x,y)$ for latent x and observed y,
2. IS proposal $q_\theta(x;y)$ parameterized by θ & y.

Find θ such that $q_\theta(x;y)$ is good for most y.

Differentiable wrt. θ for fixed x,y.
E.g. $q_\theta(x;y) = \mathrm{normal}(x; f_\theta(y), g_\theta(y))$ for neural nets f,g.

Proposal learning problem

Given:

1. joint dist. $p(x,y)$ for latent x and observed y,
2. IS proposal $q_\theta(x;y)$ parameterized by θ & y.

Find θ such that $q_\theta(x;y)$ is good for most y.

Proposal learning problem

Given:

1. joint dist. $p(x,y)$ for latent x and observed y,
2. IS proposal $q_\theta(x;y)$ parameterized by θ & y.

Find θ such that $q_\theta(x;y)$ is good for most y.

Proposal learning problem

Given:

1. joint dist. $p(x,y)$ for latent x and observed y,
2. IS proposal $q_\theta(x;y)$ parameterized by θ & y.

Find θ such that $q_\theta(x;y)$ is good for most y.

Proposal learning problem

Given:

1. joint dist. $p(x,y)$ for latent x and observed y,
2. IS proposal $q_\theta(x;y)$ parameterized by θ & y.

Find θ such that $q_\theta(x;y)$ is good for most y.

Proposal learning problem tackled by amortised inf.

Given:

1. joint dist. $p(x,y)$ for latent x and observed y,
2. IS proposal $q_\theta(x;y)$ parameterized by θ & y.

<u>Find θ</u> such that $q_\theta(x;y)$ is <u>good</u> for <u>most y</u>.

Proposal learning problem tackled by amortised inf.

y sampled from p(y)

Given:

1. joint dist. p(x,y) for latent x and observed y,
2. IS proposal $q_\theta(x;y)$ parameterized by θ & y.

Find θ such that $q_\theta(x;y)$ is good for most y.

Proposal learning problem tackled by amortised inf.

y sampled from p(y)

Given:

1. joint dist. $p(x,y)$ for latent x and observed y,
2. IS proposal $q_\theta(x;y)$ parameterized by θ & y.

<u>Find θ</u> such that $q_\theta(x;y)$ is <u>good</u> for <u>most y</u>.

Small KL divergence from $p(x|y)$ to $q_\theta(x;y)$.
$\mathrm{KL}[\, p(x|y) \,||\, q_\theta(x;y) \,] = \mathbb{E}_{p(x|y)}[\, \log(p(x|y)/q_\theta(x;y)) \,]$.

Proposal learning problem

argmin$_\theta$ $\mathbb{E}_{p(y)}$[KL[p(x|y) || q_θ(x;y)]].
Solve this by stochastic gradient descent.

inf.

y sampled from p(y)

Given:

1. joint dist. p(x,y) for latent x and observed y,
2. IS proposal q_θ(x;y) parameterized by θ & y.

Find θ such that q_θ(x;y) is good for most y.

Small KL divergence from p(x|y) to q_θ(x;y).
KL[p(x|y) || q_θ(x;y)]= $\mathbb{E}_{p(x|y)}$[log(p(x|y)/q_θ(x;y))].

Proposal learning problem inf.

$\mathrm{argmin}_\theta\ \mathbb{E}_{p(y)}[\mathrm{KL}[\ p(x|y)\ ||\ q_\theta(x;y)\ ]]$.
Solve this by stochastic gradient descent.

y sampled from p(y)

Given:

1. joint dist. $p(x,y)$ for latent x and observed y,
2. IS proposal $q_\theta(x;y)$ parameterized by θ & y.

Find θ such that $q_\theta(x;y)$ is good for most y.

Small KL divergence from $p(x|y)$ to $q_\theta(x;y)$.
$\mathrm{KL}[\ p(x|y)\ ||\ q_\theta(x;y)\ ] = \mathbb{E}_{p(x|y)}[\ \log(p(x|y)/q_\theta(x;y))\ ]$.

Stochastic gradient descent for $\mathbb{E}_{p(y)}[\mathrm{KL}[p(x|y)||q_\theta(x;y)]]$

Stochastic gradient descent for $\mathbb{E}_{p(y)}[\mathrm{KL}[p(x|y)||q_\theta(x;y)]]$

Initialise θ

Stochastic gradient descent for $\mathbb{E}_{p(y)}[KL[p(x|y)||q_\theta(x;y)]]$

Initialise θ

$\theta \leftarrow \theta - 0.01 * \nabla_\theta \mathbb{E}_{p(y)}[KL[p(x|y)||q_\theta(x;y)]]$

Stochastic gradient descent for $\mathbb{E}_{p(y)}[\mathrm{KL}[p(x|y)||q_\theta(x;y)]]$

Initialise θ

$\theta \leftarrow \theta - 0.01 * \nabla_\theta \mathbb{E}_{p(y)}[\mathrm{KL}[p(x|y)||q_\theta(x;y)]]$

$\theta \leftarrow \theta - 0.01 * \nabla_\theta \mathbb{E}_{p(y)}[\mathrm{KL}[p(x|y)||q_\theta(x;y)]]$

Stochastic gradient descent for $\mathbb{E}_{p(y)}[KL[p(x|y)||q_\theta(x;y)]]$

Initialise θ

$\theta \leftarrow \theta - 0.01 * \nabla_\theta \mathbb{E}_{p(y)}[KL[p(x|y)||q_\theta(x;y)]]$

$\theta \leftarrow \theta - 0.01 * \nabla_\theta \mathbb{E}_{p(y)}[KL[p(x|y)||q_\theta(x;y)]]$

$\theta \leftarrow \theta - 0.01 * \nabla_\theta \mathbb{E}_{p(y)}[KL[p(x|y)||q_\theta(x;y)]]$

Stochastic gradient descent for $\mathbb{E}_{p(y)}[KL[p(x|y)||q_\theta(x;y)]]$

Initialise θ

$\theta \leftarrow \theta - 0.01 * \nabla_\theta \mathbb{E}_{p(y)}[KL[p(x|y)||q_\theta(x;y)]]$

$\theta \leftarrow \theta - 0.01 * \nabla_\theta \mathbb{E}_{p(y)}[KL[p(x|y)||q_\theta(x;y)]]$

$\theta \leftarrow \theta - 0.01 * \nabla_\theta \mathbb{E}_{p(y)}[KL[p(x|y)||q_\theta(x;y)]]$

…

(until θ doesn't change much)

Stochastic gradient descent for $\mathbb{E}_{p(y)}[KL[p(x|y)||q_\theta(x;y)]]$

Learning rate

Initialise θ

$\theta \leftarrow \theta - 0.01 * \nabla_\theta \mathbb{E}_{p(y)}[KL[p(x|y)||q_\theta(x;y)]]$

$\theta \leftarrow \theta - 0.01 * \nabla_\theta \mathbb{E}_{p(y)}[KL[p(x|y)||q_\theta(x;y)]]$

$\theta \leftarrow \theta - 0.01 * \nabla_\theta \mathbb{E}_{p(y)}[KL[p(x|y)||q_\theta(x;y)]]$

…

(until θ doesn't change much)

Stochastic gradient descent for $\mathbb{E}_{p(y)}[KL[p(x|y)||q_\theta(x;y)]]$

Initialise θ

$\theta \leftarrow \theta - 0.01 * \nabla_\theta \mathbb{E}_{p(y)}[KL[p(x|y)||q_\theta(x;y)]]$

$\theta \leftarrow \theta - 0.01 * \nabla_\theta \mathbb{E}_{p(y)}[KL[p(x|y)||q_\theta(x;y)]]$

$\theta \leftarrow \theta - 0.01 * \nabla_\theta \mathbb{E}_{p(y)}[KL[p(x|y)||q_\theta(x;y)]]$

…

Can't compute, but can approximate.

Stochastic gradient descent for $\mathbb{E}_{p(y)}[KL[p(x|y)||q_\theta(x;y)]]$

Initialise θ

$\theta \leftarrow \theta - 0.01 * \nabla_\theta \mathbb{E}_{p(y)}[KL[p(x|y)||q_\theta(x;y)]]$

$\theta \leftarrow \theta - 0.01 * \nabla_\theta \mathbb{E}_{p(y)}[KL[p(x|y)||q_\theta(x;y)]]$

$\theta \leftarrow \theta - 0.01 * \nabla_\theta \mathbb{E}_{p(y)}[KL[p(x|y)||q_\theta(x;y)]]$

…

Can't compute, but can approximate.
Sample (x_1,y_1), …, (x_n,y_n) from $p(x,y)$.

Stochastic gradient descent for $\mathbb{E}_{p(y)}[KL[p(x|y)||q_\theta(x;y)]]$

Initialise θ

$\theta \leftarrow \theta - 0.01 * \nabla_\theta \mathbb{E}_{p(y)}[KL[p(x|y)||q_\theta(x;y)]]$

$\theta \leftarrow \theta - 0.01 * \nabla_\theta \mathbb{E}_{p(y)}[KL[p(x|y)||q_\theta(x;y)]]$

$\theta \leftarrow \theta - 0.01 * \nabla_\theta \mathbb{E}_{p(y)}[KL[p(x|y)||q_\theta(x;y)]]$

…

Can't compute, but can approximate.
Sample $(x_1,y_1), \ldots, (x_n,y_n)$ from $p(x,y)$.
$\nabla_\theta \mathbb{E}_{p(y)}[KL[p(x|y)||q_\theta(x;y)]] \approx -1/n * \sum_{i=1..n} \nabla_\theta \log q_\theta(x_i;y_i)$.

Stochastic gradient descent for $\mathbb{E}_{p(y)}[KL[p(x|y)||q_\theta(x;y)]]$

Initialise θ

$\theta \leftarrow \theta - 0.01 * \nabla_\theta \mathbb{E}_{p(y)}[KL[p(x|y)||q_\theta(x;y)]]$

$\theta \leftarrow \theta - 0.0$

$\theta \leftarrow \theta - 0.0$

…

Hard to sample x from p(x|y) for given y,
but easy to sample (x,y) from p(x,y).
Thus, no problem in sampling.

Can't compute, but can approximate.

Sample (x_1,y_1), …, (x_n,y_n) from p(x,y).

$\nabla_\theta \mathbb{E}_{p(y)}[KL[p(x|y)||q_\theta(x;y)]] \approx -1/n * \sum_{i=1..n} \nabla_\theta \log q_\theta(x_i;y_i)$.

Stochastic gradient descent for $\mathbb{E}_{p(y)}[KL[p(x|y)||q_\theta(x;y)]]$

Initialise θ

$\theta \leftarrow \theta - 0.01 * \nabla_\theta \mathbb{E}_{p(y)}[KL[p(x|y)||q_\theta(x;y)]]$

$\theta \leftarrow \theta - 0.01 * \nabla_\theta \mathbb{E}_{p(y)}[KL[p(x|y)||q_\theta(x;y)]]$

$\theta \leftarrow \theta - 0.01 * \nabla_\theta \mathbb{E}_{p(y)}[KL[p(x|y)||q_\theta(x;y)]]$

...

Can't compute, but can approximate.

Sample $(x_1,y_1), \ldots, (x_n,y_n)$ from $p(x,y)$.

$\nabla_\theta \mathbb{E}_{p(y)}[KL[p(x|y)||q_\theta(x;y)]] \approx -1/n * \sum_{i=1..n} \nabla_\theta \log q_\theta(x_i;y_i)$.

Exists since $q_\theta(x_i;y_i)$ is differentiable.

Stochastic gradient descent for $\mathbb{E}_{p(y)}[KL[p(x|y)||q_\theta(x;y)]]$

Initialise θ

$\theta \leftarrow \theta - 0.01 * \nabla_\theta \mathbb{E}_{p(y)}[KL[p(x|y)||q_\theta(x;y)]]$

$\theta \leftarrow \theta - 0.01 * \nabla_\theta \mathbb{E}_{p(y)}[KL[p(x|y)||q_\theta(x;y)]]$

$\theta \leftarrow \theta - 0.01 * \nabla_\theta \mathbb{E}_{p(y)}[KL[p(x|y)||q_\theta(x;y)]]$

…

[Q] Prove that this is an unbiased estimator.

Can't compute, but can approximate.

Sample $(x_1,y_1), \ldots, (x_n,y_n)$ from $p(x,y)$.

$\nabla_\theta \mathbb{E}_{p(y)}[KL[p(x|y)||q_\theta(x;y)]] \approx -1/n * \sum_{i=1..n} \nabla_\theta \log q_\theta(x_i;y_i)$.

Stochastic gradient descent for $\mathbb{E}_{p(y)}[KL[p(x|y)||q_\theta(x;y)]]$

Initialise θ

Repeat the following until θ doesn't change much:

1. Sample $(x_1,y_1), \ldots, (x_n,y_n)$ from $p(x,y)$
2. $G \leftarrow -1/n * \sum_{i=1..n} \nabla_\theta \log q_\theta(x_i;y_i)$
3. $\theta \leftarrow \theta - 0.01 * G$

Learning IS proposal $q_\theta(x;y)$ by amortised inference

Using stochastic gradient descent, solve:

$$\mathrm{argmin}_\theta\ \mathbb{E}_{p(y)}[\mathrm{KL}[p(x|y)\ ||\ q_\theta(x;y)]].$$

Learning IS proposal $q_\theta(x;y)$ by amortised inference

Using stochastic gradient descent, solve:

$$\mathrm{argmin}_\theta \; \mathbb{E}_{p(y)}[\mathrm{KL}[p(x|y) \,||\, q_\theta(x;y)]].$$

[Q] Differences from stochastic variational inf.?

SVI: $\mathrm{argmin}_\theta \; \mathrm{KL}[q_\theta(x) \,||\, p(x|y_0)]$ for a given y_0.

Learning IS proposal $q_\theta(x;y)$ by amortised inference

Using stochastic gradient descent, solve:

$$\text{argmin}_\theta \, \mathbb{E}_{p(y)}[\mathrm{KL}[p(x|y) \,||\, q_\theta(x;y)]].$$

[Q] Differences from stochastic variational inf.?

SVI: $\text{argmin}_\theta \, \mathrm{KL}[q_\theta(x) \,||\, p(x|y_0)]$ for a given y_0.

(a) KL[true||approx] vs KL[approx||true].

Learning IS proposal $q_\theta(x;y)$ by amortised inference

Using stochastic gradient descent, solve:

$$\mathrm{argmin}_\theta\ \mathbb{E}_{p(y)}[\mathrm{KL}[p(x|y)\ ||\ q_\theta(x;y)]].$$

[Q] Differences from stochastic variational inf.?

SVI: $\mathrm{argmin}_\theta\ \mathrm{KL}[q_\theta(x)\ ||\ p(x|y_0)]$ for a given y_0.

(a) KL[true||approx] vs KL[approx||true].

Choice consistent with IS's condition on q_θ's support.

Learning IS proposal $q_\theta(x;y)$ by amortised inference

Using stochastic gradient descent, solve:

$$\text{argmin}_\theta\ \mathbb{E}_{p(y)}[\mathrm{KL}[p(x|y)\ ||\ q_\theta(x;y)]].$$

[Q] Differences from stochastic variational inf.?

SVI: $\text{argmin}_\theta\ \mathrm{KL}[q_\theta(x)\ ||\ p(x|y_0)]$ for a given y_0.

(a) KL[true||approx] vs KL[approx||true].

Choice consistent with IS's condition on q_θ's support.

Lets us avoid sampling from posterior $p(x|y_0)$.

Learning IS proposal $q_\theta(x;y)$ by amortised inference

Using stochastic gradient descent, solve:

$$\operatorname{argmin}_\theta \mathbb{E}_{p(y)}[KL[p(x|y) \,||\, q_\theta(x;y)]].$$

[Q] Differences from stochastic variational inf.?

SVI: $\operatorname{argmin}_\theta KL[q_\theta(x) \,||\, p(x|y_0)]$ for a given y_0.

(a) KL[true||approx] vs KL[approx||true].

Learning IS proposal $q_\theta(x;y)$ by amortised inference

Using stochastic gradient descent, solve:

$$\text{argmin}_\theta\ \mathbb{E}_{p(y)}[KL[p(x|y)\ ||\ q_\theta(x;y)]].$$

[Q] Differences from stochastic variational inf.?

SVI: $\text{argmin}_\theta\ KL[q_\theta(x)\ ||\ p(x|y_0)]$ for a given y_0.

(a) KL[true||approx] vs KL[approx||true].

(b) Generated y vs given y_0.

Learning IS proposal $q_\theta(x;y)$

Lets us avoid sampling x from posterior $p(x|y_0)$ for given y_0. Just need to sample (x,y) from joint $p(x,y)$.

Using stochastic gradient descent, solve:

$$\mathrm{argmin}_\theta\ \mathbb{E}_{p(y)}[KL[p(x|y)\ ||\ q_\theta(x;y)]].$$

[Q] Differences from stochastic variational inf.?

SVI: $\mathrm{argmin}_\theta\ KL[q_\theta(x)\ ||\ p(x|y_0)]$ for a given y_0.

(a) KL[true||approx] vs KL[approx||true].

(b) Generated y vs given y_0.

Learning IS proposal $q_\theta(x;y)$ by amortised inference

Using stochastic gradient descent, solve:

$$\operatorname{argmin}_\theta \mathbb{E}_{p(y)}[\mathrm{KL}[p(x|y) \,||\, q_\theta(x;y)]].$$

[Q] Differences from stochastic variational inf.?

SVI: $\operatorname{argmin}_\theta \mathrm{KL}[q_\theta(x) \,||\, p(x|y_0)]$ for a given y_0.

(a) KL[true||approx] vs KL[approx||true].

(b) Generated y vs given y_0.

What about probabilistic programs?

Stochastic gradient descent for $\mathbb{E}_{p(y)}[KL[p(x|y)||q_\theta(x;y)]]$

Initialise θ

Repeat the following until θ doesn't change:

1. Sample $(x_1,y_1), \ldots, (x_n,y_n)$ from $p(x,y)$
2. $G \leftarrow -1/n * \sum_{i=1..n} \nabla_\theta \log q_\theta(x_i;y_i)$
3. $\theta \leftarrow \theta - 0.01 * G$

Stochastic gradient descent for $\mathbb{E}_{p(y)}[KL[p(x|y)||q_\theta(x;y)]]$

Initialise θ

Repeat the following until θ doesn't change:

1. Sample (x_1,y_1), …, (x_n,y_n) from $p(x,y)$
2. $G \leftarrow -1/n * \sum_{i=1..n}\nabla_\theta \log q_\theta(x_i;y_i)$
3. $\theta \leftarrow \theta - 0.01 * G$

Stochastic gradient descent for $\mathbb{E}_{p(y)}[KL[p(x|y)||q_\theta(x;y)]]$

Initialise θ

Repeat the following until θ doesn't change:

1. Sample $(x_1,y_1), \ldots, (x_n,y_n)$ from $p(x,y)$
2. $G \leftarrow -1/n * \sum_{i=1..n} \nabla_\theta \log q_\theta(x_i;y_i)$
3. $\theta \leftarrow \theta - 0.01 * G$

How to sample y?

Sample/observe duality

To sample observations, just replace sample by observe.

```
(defquery biased-coin []
  (let [r (sample
            (uniform-continuous 0 1))
        a (observe (flip r) true)
        b (observe (flip r) true)
        c (observe (flip r) true)]
   r))
```

Sample/observe duality

To sample observations, just replace sample by observe.

```
(defquery biased-coin-joint []
  (let [r (sample
             (uniform-continuous 0 1))
        a (sample (flip r))
        b (sample (flip r))
        c (sample (flip r))]
    [r [a b c]]))
```

Stochastic gradient descent for $\mathbb{E}_{p(y)}[KL[p(x|y)||q_\theta(x;y)]]$

Initialise θ

Repeat the following until θ doesn't change:

1. Sample $(x_1,y_1), \ldots, (x_n,y_n)$ from $p(x,y)$
2. $G \leftarrow -1/n * \sum_{i=1..n} \nabla_\theta \log q_\theta(x_i;y_i)$
3. $\theta \leftarrow \theta - 0.01 * G$

Compute during execution.
Similar to the SVI case.
Just a new rule for sample.

Last remark

People also use "amortised inference" to mean parameter sharing via neural net in a distr.

Silly example:

1. $q(x_1,x_2;\theta_1,\theta_2) = q(x_1;\theta_1)q(x_2;\theta_2)$
2. $q(x_1,x_2;\theta) = q(x_1;f_\theta(1))q(x_2;f_\theta(2))$

where f_θ is a neural net.

References

1. Inference networks for sequential Monte Carlo in graphical models. Paige et al. ICML'16.
2. Inference compilation and universal probabilistic programming. Le et al. AISTATS'17.