CS492: Probabilistic Programming

Variational Inference

Hongseok Yang
KAIST

Posterior inference

Given:

- Prior $p(x)$ and likelihood $p(y|x)$.
- Observations y.

Goal:

- Find a good approximation of $p(x|y)$.

Monte-Carlo approach:

- Approximates posterior p(x|y) by samples.
- Exact in the limit, but scalability issue.
- MCMC, importance sampling, etc.

Monte-Carlo approach:

- Approximates posterior $p(x|y)$ by samples.
- Exact in the limit, but scalability issue.
- MCMC, importance sampling, etc.

Variational inference:

- Approximates $p(x|y)$ by a distribution $q_\theta(x)$.
- Scales, but not exact (biased).
- Stochastic variational inference, expectation propagation, etc.

Monte-Carlo approach:

- Approximates posterior p(x|y) by samples.
- Exact in the limit, but scalability issue.
- MCMC, importance sampling, etc.

Variational inference:

- Approximates p(x|y) by a distribution $q_\theta(x)$.
- Scales, but not exact (biased).
- Stochastic variational inference, expectation propagation, etc.

Monte-Carlo approach:

- Approximates posterior $p(x|y)$ by samples.
- Exact in the limit, but scalability issue.
- MCMC, importance sampling, etc.

Variational inference:

- Approximates $p(x|y)$ by a distribution $q_\theta(x)$.
- Scales, but not exact (biased).
- Stochastic variational inference, expectation propagation, etc.

Microsoft's infer.NET

Monte-Carlo approach:

- Approximates posterior p(x|y) by samples.
- Exact in the limit, but scalability issue.

Uber's pyro
Google's Edward
Anglican, Stan

- nportance sampling, etc.

Variational inference:

- Approximates p(x|y) by a distribution $q_\theta(x)$.
- Scales, but not exact (biased).
- Stochastic variational inference, expectation propagation, etc.

Microsoft's infer.NET

Learning outcome

- Should be able to use stochastic variational inference (SVI) algorithms for data analysis.
- Should be able to derive key equations for a basic SVI algorithm with score estimator.
- Should be able to implement this basic SVI algorithm for probabilistic PLs.

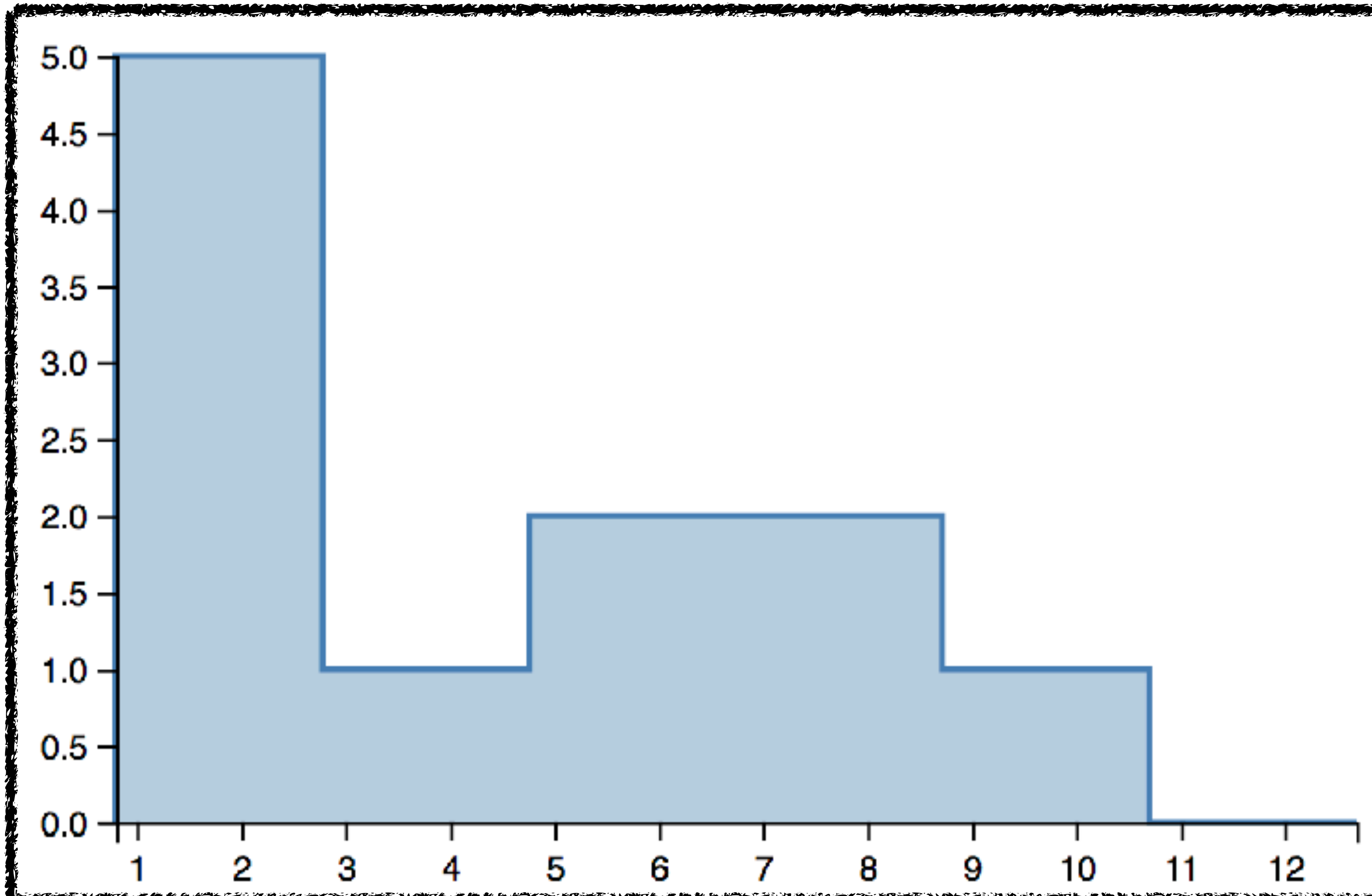
Using stochastic variational inference in Anglican

Mixture of two Gaussians

```
(defquery mixture []
  (let [p   (sample :p (beta 1 1))
        m1  (sample :m1 (normal 0 5))
        m2  (sample :m2 (normal 0 5))
        f   (fn [y]
              (if (sample (flip p))
                (observe (normal m1 1.) y)
                (observe (normal m2 1.) y)))]
    (map f (list 0.8 1.2 7.8 2.4 8.2
                 10.7 5.3 2.6 1.2 3.4 5.6))
    [p m1 m2]))
```

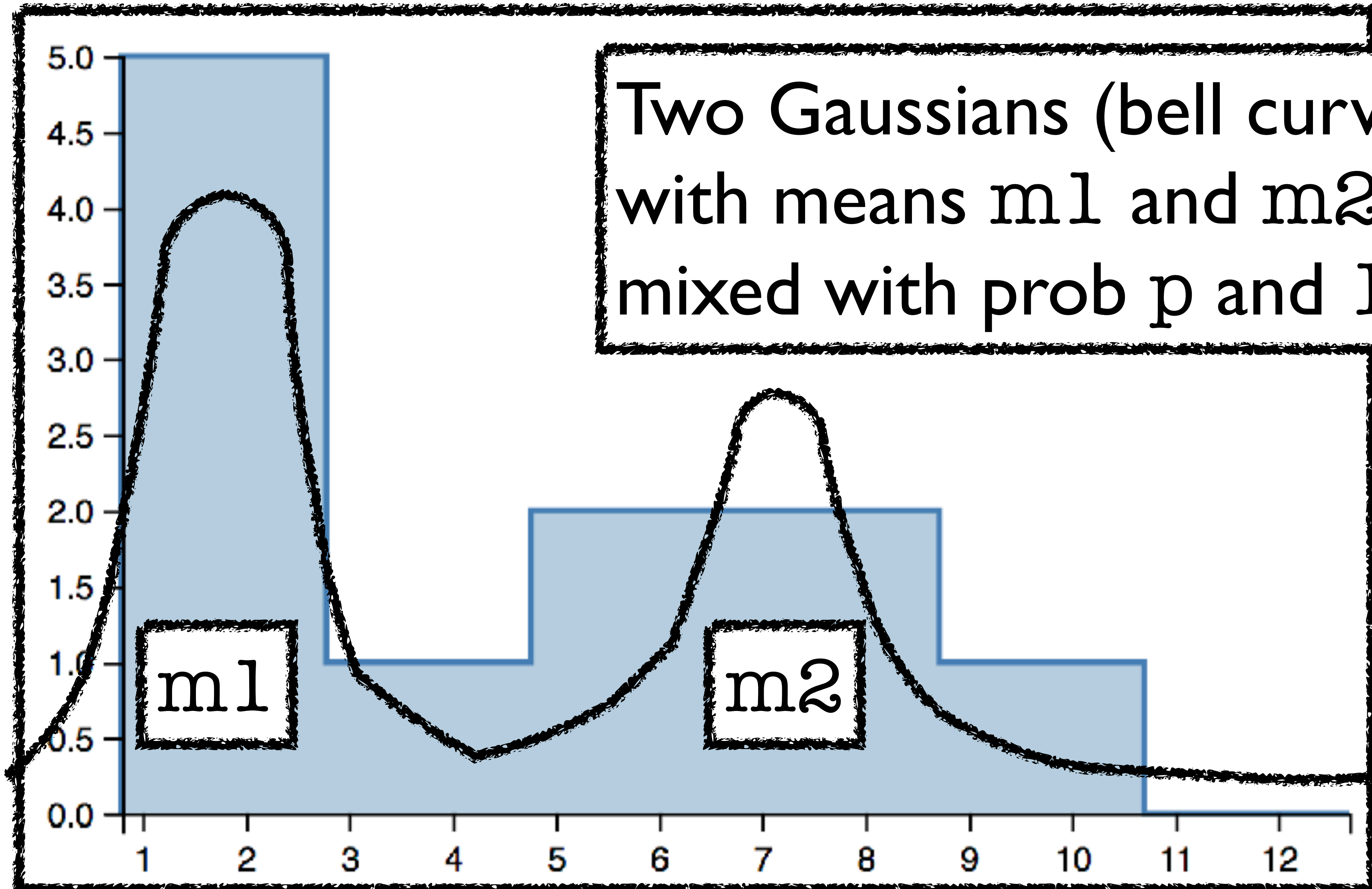
Mixture of two Gaussians

```
(defquery mixture []
  (let [p   (sample :p (beta 1 1))
        m1 (sample :m1 (normal 0 5))
        m2 (sample :m2 (normal 0 5))
        f    (fn [y]
               (if (sample (flip p))
                 (observe (normal m1 1.) y)
                 (observe (normal m2 1.) y)))]
    (map f (list 0.8 1.2 7.8 2.4 8.2
                 10.7 5.3 2.6 1.2 3.4 5.6))
    [p m1 m2]))
```

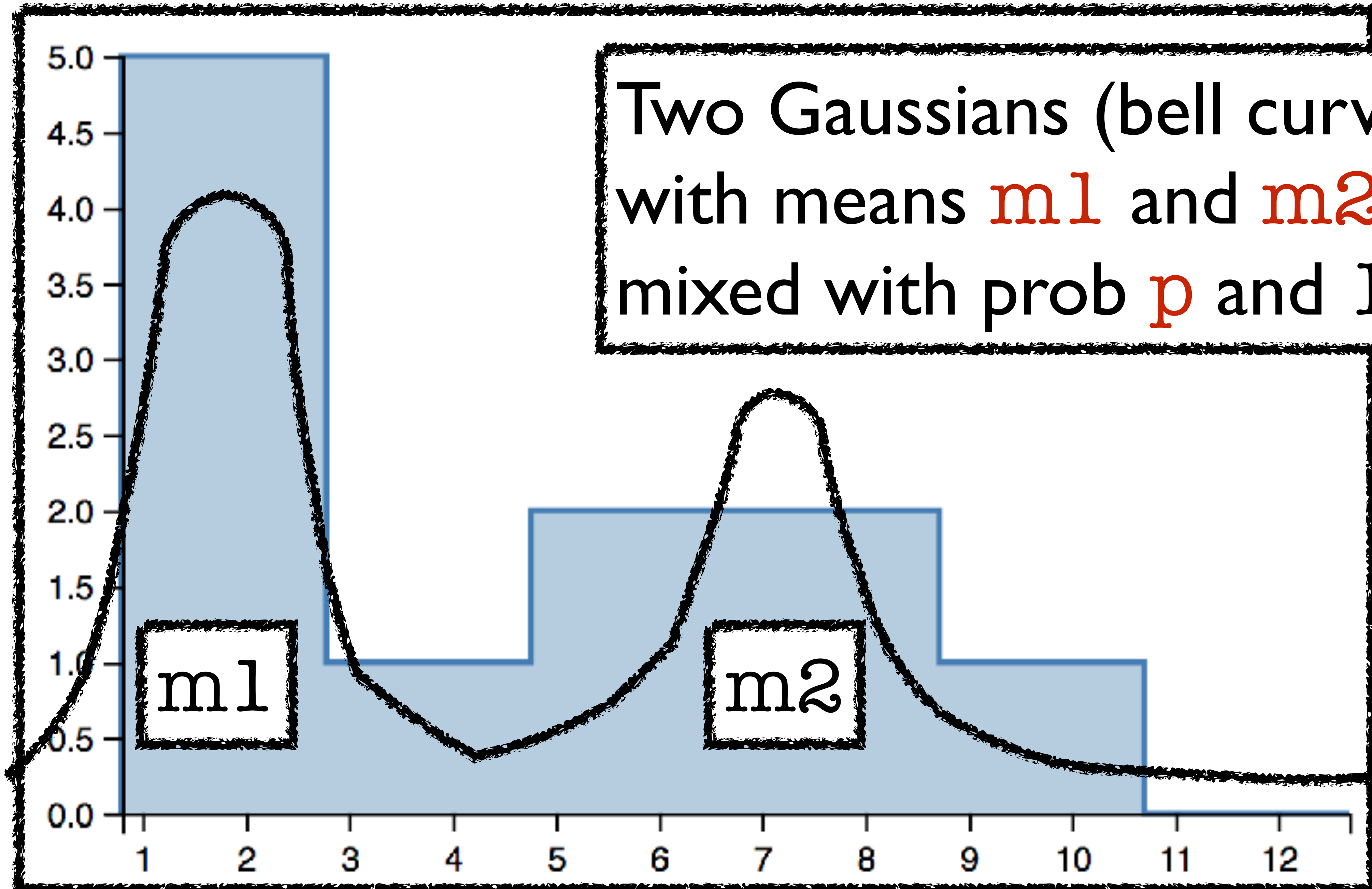



```
(map f (list 0.8  1.2  7.8  2.4  8.2
             10.7  5.3  2.6  1.2  3.4  5.6))
[p m1 m2]))
```

Two Gaussians (bell curves) with means `m1` and `m2`, mixed with prob `p` and `1-p`




```
(map f (list 0.8  1.2  7.8  2.4  8.2
             10.7  5.3  2.6  1.2  3.4  5.6))
[p m1 m2]))
```



Two Gaussians (bell curves) with means `m1` and `m2`, mixed with prob `p` and `1-p`

```
(map f (list 0.8  1.2  7.8  2.4  8.2
             10.7  5.3  2.6  1.2  3.4  5.6))
[p m1 m2]))
```

Mixture of two Gaussians

```
(defquery mixture []
  (let [p   (sample :p (beta 1 1))
        m1 (sample :m1 (normal 0 5))
        m2 (sample :m2 (normal 0 5))
        f    (fn [y]
               (if (sample (flip p))
                 (observe (normal m1 1.) y)
                 (observe (normal m2 1.) y)))]
    (map f (list 0.8  1.2  7.8  2.4  8.2
                 10.7  5.3  2.6  1.2  3.4  5.6))
    [p m1 m2]))
```

Mixture of two Gaussians

Labels for sample expressions

```
(defquery mixture []
  (let [p   (sample :p (beta 1 1))
        m1 (sample :m1 (normal 0 5))
        m2 (sample :m2 (normal 0 5))
        f    (fn [y]
               (if (sample (flip p))
                 (observe (normal m1 1.) y)
                 (observe (normal m2 1.) y)))]
    (map f (list 0.8  1.2  7.8  2.4  8.2
                 10.7  5.3  2.6  1.2  3.4  5.6))
    [p m1 m2]))
```

Mixture of two Gaussians

Distribution on [0,1]
Uniform in this case

```
(defquery mixture []
  (let [p   (sample :p (beta 1 1))
        m1 (sample :m1 (normal 0 5))
        m2 (sample :m2 (normal 0 5))
        f   (fn [y]
              (if (sample (flip p))
                (observe (normal m1 1.) y)
                (observe (normal m2 1.) y)))]
    (map f (list 0.8  1.2  7.8  2.4  8.2
                 10.7  5.3  2.6  1.2  3.4  5.6))
    [p m1 m2]))
```

Mixture of two Gaussians

```
(defquery mixture []
  (let [p    (sample :p (beta 1 1))
        m1  (sample :m1 (normal 0 5))
        m2  (sample :m2 (normal 0 5))
        f    (fn [y]
               (if (sample (flip p))
                 (observe (normal m1 1.) y)
                 (observe (normal m2 1.) y)))]
    (map f (list 0.8 1.2 7.8 2.4 8.2
                 10.7 5.3 2.6 1.2 3.4 5.6))
    [p m1 m2]))
```

Mixture of two Gaussians

```
(defquery mixture []
  (let [p    (sample :p (beta 1 1))
        m1 (sample :m1 (normal 0 5))
        m2 (sample :m2 (normal 0 5))
        f     (fn [y]
                (if (sample (flip p))
                  (observe (normal m1 1.) y)
                  (observe (normal m2 1.) y)))]
    (map f (list 0.8 1.2 7.8 2.4 8.2
                 10.7 5.3 2.6 1.2 3.4 5.6))
    [p m1 m2]))
```

```
(def r1 (nth (doquery :bbvb mixture []) 50000))
(def r2 (anglican.bbvb/get-variational r1))

(print (:p r2) "\n\n" (:m1 r2) "\n\n" (:m2 r2))
```

```
(defquery mixture []
  (let [p   (sample :p (beta 1 1))
        m1 (sample :m1 (normal 0 5))
        m2 (sample :m2 (normal 0 5))
        f    (fn [y]
               (if (sample (flip p))
                 (observe (normal m1 1.) y)
                 (observe (normal m2 1.) y)))]
    (map f (list 0.8  1.2  7.8  2.4  8.2
                 10.7  5.3  2.6  1.2  3.4  5.6))
    [p m1 m2]))
```

```
(def r1 (nth (doquery :bbvb mixture []) 50000))
(def r2 (anglican.bbvb/get-variational r1))

(print (:p r2) "\n\n" (:m1 r2) "\n\n" (:m2 r2))
```

```
(defquery mixture-posterior-approx []
  (let [p   (sample :p (beta 1 1))
        m1  (sample :m1 (normal 0 5))
        m2  (sample :m2 (normal 0 5))
        f   (fn [y]
              (if (sample (flip p))
                (observe (normal m1 1.) y)
                (observe (normal m2 1.) y)))]
    (map f (list 0.8 1.2 7.8 2.4 8.2
                 10.7 5.3 2.6 1.2 3.4 5.6))
    [p m1 m2]))
```

Approx. by Anglican query

```
(def r1 (nth (doquery :bbvb mixture []) 50000))
(def r2 (anglican.bbvb/get-variational r1))

(print (:p r2) "\n\n" (:m1 r2) "\n\n" (:m2 r2))
```

```
(defquery mixture-posterior-approx []
  (let [p    (sample :p (beta 1 1))
        m1 (sample :m1 (normal 0 5))
        m2 (sample :m2 (normal 0 5))
        f     (fn [y]
                (if (sample (flip p))
                  (observe (normal m1 1.) y)
                  (observe (normal m2 1.) y)))]
    (map f (list 0.8 1.2 7.8 2.4 8.2
                  10.7 5.3 2.6 1.2 3.4 5.6))
    [p m1 m2]))
```

Approx. by Anglican query
(1) No obs.

```
(def r1 (nth (doquery :bbvb mixture []) 50000))
(def r2 (anglican.bbvb/get-variational r1))

(print (:p r2) "\n\n" (:m1 r2) "\n\n" (:m2 r2))
```

```
(defquery mixture-posterior-approx []
  (let [p   (sample :p (beta 1 1))
        m1 (sample :m1 (normal 0 5))
        m2 (sample :m2 (normal 0 5))
        f    (fn [y]
               (if (sample (flip p))
                 (observe (normal m1 1.) y)
                 (observe (normal m2 1.) y)))]
    (map f (list 0.8 1.2 7.8 2.4 8.2
                 10.7 5.3 2.6 1.2 3.4 5.6))
    [p m1 m2]))
```

Approx. by Anglican query

(1) No obs. (2) New params.

```
(def r1 (nth (doquery :bbvb mixture []) 50000))
(def r2 (anglican.bbvb/get-variational r1))

(print (:p r2) "\n\n" (:m1 r2) "\n\n" (:m2 r2))
```

```
(defquery mixture-posterior-approx []
  (let [p    (sample :p (beta 1 1))
        m1 (sample :m1 (normal 0 5))
        m2 (sample :m2 (normal 0 5))
        f    (fn [y]
                                                l m1 1.) y)
                                                l m2 1.) y)))]
                                                ~~8.2~~
                                                ~~2 3 4 5 6))~~
```

```
{(0 anglican.runtime.beta-distribution)
(anglican.runtime/beta 6.765287422267994
5.346760324478156)}

 {(0 anglican.runtime.normal-distribution)
(anglican.runtime/normal 2.103819374566423
0.3831737586707096)}

 {(0 anglican.runtime.normal-distribution)
(anglican.runtime/normal 7.242181702264095
0.417015340354034l)}
```

y Anglican query
s. (2) New params.

```
(def r1 (nth (doquery :bbvb mixture []) 50000))
(def r2 (anglican.bbvb/get-variational r1))

(print (:p r2) "\n\n" (:m1 r2) "\n\n" (:m2 r2))
```

```
(defquery mixture-posterior-approx []
  (let [p    (sample :p (beta 1 1))
        m1 (sample :m1 (normal 0 5))
        m2 (sample :m2 (normal 0 5))
        f     (fn [y]
...l m1 1.) y)
...l m2 1.) y)))]
```

```
{(0 anglican.runtime.beta-distribution)
(anglican.runtime/beta 6.765287422267994
5.346760324478156)}

 {(0 anglican.runtime.normal-distribution)
(anglican.runtime/normal 2.103819374566423
0.3831737586707096)}

 {(0 anglican.runtime.normal-distribution)
(anglican.runtime/normal 7.242181702264095
0.4170153403540341)}
```

y Anglican query
s. (2) New params.

```
(def r1 (nth (doquery :bbvb mixture []) 50000))
(def r2 (anglican.bbvb/get-variational r1))

(print (:p r2) "\n\n" (:m1 r2) "\n\n" (:m2 r2))
```

```
(defquery mixture-posterior-approx []
  (let [p    (sample :p (beta 1 1))
        m1 (sample :m1 (normal 0 5))
        m2 (sample :m2 (normal 0 5))
        f     (fn [y]
```

```
{(0 anglican.runtime.beta-distribution)
(anglican.runtime/beta 6.765287422267994
5.346760324478156)}

 {(0 anglican.runtime.normal-distribution)
(anglican.runtime/normal 2.103819374566423
0.3831737586707096)}

 {(0 anglican.runtime.normal-distribution)
(anglican.runtime/normal 7.242181702264095
0.4170153403540341)}
```

```
l m1 1.) y)
l m2 1.) y)))]
~~8.2~~
~~2 3 4 5 6))~~
```

y Anglican query
s. (2) New params.

```
(def r1 (nth (doquery :bbvb mixture []) 50000))
(def r2 (anglican.bbvb/get-variational r1))

(print (:p r2) "\n\n" (:m1 r2) "\n\n" (:m2 r2))
```

```
(defquery mixture-posterior-approx []
  (let [p    (sample :p (beta 1 1))
        m1 (sample :m1 (normal 0 5))
        m2 (sample :m2 (normal 0 5))
        f     (fn [y]
                                   l m1 1.) y)
                                   l m2 1.) y)))]
```

```
{(0 anglican.runtime.beta-distribution)
(anglican.runtime/beta 6.765287422267994
5.346760324478156)}

 {(0 anglican.runtime.normal-distribution)
(anglican.runtime/normal 2.103819374566423
0.3831737586707096)}

 {(0 anglican.runtime.normal-distribution)
(anglican.runtime/normal 7.242181702264095
0.4170153403540341)}
```

y Anglican query
s. (2) New params.

Variational inference from the users' perspective

- Approximates a posterior by a distribution.
- In Anglican, the approximation has the form of the original query except that

1. all observes are gone;
2. dist. parameters at sample get changed.

What does stochastic variational inference do?

Variational inference

1. Fix a family of approximating distr. $\{q_\theta(x)\}_\theta$.
2. Find θ s.t. $q_\theta(x)$ is closest to post. $p(x|y)$.

Variational inference

1. Fix a family of approximating distr. $\{q_\theta(x)\}_\theta$.
2. Find θ s.t. $q_\theta(x)$ is closest to post. $p(x|y)$.

Don't sum. Optimise.

Variatio

Three choices:

1. Fix a family of approximating distr. $\{q_\theta(x)\}_\theta$.
2. Find θ s.t. $q_\theta(x)$ is closest to post. $p(x|y)$.

Don't sum. Optimise.

Variatio

Three choices:
(1) Which family?

1. Fix a family of approximating distr. $\{q_\theta(x)\}_\theta$.
2. Find θ s.t. $q_\theta(x)$ is closest to post. $p(x|y)$.

Don't sum. Optimise.

Variatio

Three choices:
(1) Which family?
(2) What does "closest" mean?

1. Fix a family of approximating distr. $\{q_\theta(x)\}_\theta$.
2. Find θ s.t. $q_\theta(x)$ is closest to post. $p(x|y)$.

Don't sum. Optimise.

Variatio

Three choices:
(1) Which family?
(2) What does "closest" mean?
(3) How to optimise?

1. Fix a family of approximating distr. $\{q_\theta(x)\}_\theta$.
2. Find θ s.t. $q_\theta(x)$ is closest to post. $p(x|y)$.

Don't sum. Optimise.

Variatic

Three choices:
(1) Which family?
(2) What does "closest" mean?
(3) How to optimise?

1. Fix a family of approximating distr. $\{q_\theta(x)\}_\theta$.
2. Find θ s.t. $q_\theta(x)$ is closest to post. $p(x|y)$.

Don't sum. Optimise.

Basic (but not uncommon) choices.

1. Which family?

2. What does "closest" mean?

3. How to optimize?

Basic (but not uncommon) choices.

1. Which family?
   - Mean field — $q_\theta(x_1, \ldots, x_n) = \prod_k q_\theta(x_k)$.
2. What does "closest" mean?
3. How to optimize?

Basic (but not uncommon) choices.

1. Which family?
   - Mean field — $q_\theta(x_1, \ldots, x_n) = \prod_k q_\theta(x_k)$.
2. What does "closest" mean?
   - KL divergence — $KL[q_\theta(x) \,||\, p(x|y)]$.
3. How to optimize?

Basic (but not uncommon) choices.

1. Which family?
   - Mean field — $q_\theta(x_1, \ldots, x_n) = \prod_k q_\theta(x_k)$.
2. What does "closest" mean?
   - KL divergence — $KL[q_\theta(x) \,||\, p(x|y)]$.
3. How to optimize?
   - Stochastic gradient descent.

Basic (but not uncommon) choices.

1. Which family?
   - Mean field — $q_\theta(x_1, \ldots, x_n) = \prod_k q_\theta(x_k)$.
2. What does "closest" mean?
   - KL divergence — $KL[q_\theta(x) \,||\, p(x|y)]$.
3. How to optimize?
   - Stochastic gradient descent.

Mean-field approximation

Mean-field approximation

Approximating distribution q_θ formed by independent random variables:

$$q_\theta(x_{1,\ \ldots,}\ x_n) = \prod_k q_\theta(x_k).$$

E.g. $q_\theta(x_1,x_2) = \mathrm{normal}(x_1;\theta_1,2) \times \mathrm{normal}(x_2;\theta_{2,}2)$.

Very crude. But popular, because it simplifies optimisation in variational inference.

Naive mean-field approx. for probabilistic programs

1. Remove all observe expressions.
2. Replace all (sample (dist e_1 … e_n)) by (sample (dist θ_1 … θ_n)).

Model

```
(let [x  (sample (beta 3 2))
      y  (if (sample (flip x))
             (sample (normal (* x x) 1))
             (sample (normal (* 5 x) 1)))]
  (observe (normal y 1) 3)
  x)
```

Model

```
(let [x  (sample (beta 3 2))
      y  (if (sample (flip x))
             (sample (normal (* x x) 1))
             (sample (normal (* 5 x) 1)))]
 (observe (normal y 1) 3)
 x)
```

Approx. family


```
(let [x  (sample (beta 3 2))
      y  (if (sample (flip x))
             (sample (normal (* x x) 1))
             (sample (normal (* 5 x) 1)))]
 (observe (normal y 1) 3)
 x)
```

Model

```
(let [x  (sample (beta 3 2))
      y  (if (sample (flip x))
             (sample (normal (* x x) 1))
             (sample (normal (* 5 x) 1)))]
 (observe (normal y 1) 3)
 x)
```

Approx. family

```
(let [x  (sample (beta 3 2))
      y  (if (sample (flip x))
             (sample (normal (* x x) 1))
             (sample (normal (* 5 x) 1)))]

 x)
```

Model

```
(let [x  (sample (beta 3 2))
      y  (if (sample (flip x))
             (sample (normal (* x x) 1))
             (sample (normal (* 5 x) 1)))]
 (observe (normal y 1) 3)
 x)
```

Approx. family

```
(let [x  (sample (beta 3 2))
      y  (if (sample (flip x))
             (sample (normal (* x x) 1))
             (sample (normal (* 5 x) 1)))]

 x)
```

Model

```
(let [x  (sample (beta 3 2))
      y  (if (sample (flip x))
             (sample (normal (* x x) 1))
             (sample (normal (* 5 x) 1)))]
 (observe (normal y 1) 3)
 x)
```

Approx. family

```
(let [x  (sample (beta θ1 θ2))
      y  (if (sample (flip θ3))
             (sample (normal θ4 θ5))
             (sample (normal θ6 θ7)))]

 x)
```

Model

```
(let [x  (sample (beta 3 2))
      y  (if (sample (flip x))
             (sample (normal (* x x) 1))
             (sample (normal (* 5 x) 1)))]
  (observe (normal y 1) 3)
  x)
```

Approx. family

```
(let [x  (sample (beta θ1 θ2))
      y  (if (sample (flip θ3))
             (sample (normal θ4 θ5))
             (sample (normal θ6 θ7)))]

  x)
```

All data-dependencies disappear.
But finding good θ_i's gets easier.

Basic (but not uncommon) choices.

1. Which family?
   - Mean field — $q_\theta(x_1, \ldots, x_n) = \prod_k q_\theta(x_k)$.
2. What does "closest" mean?
   - KL divergence — $KL[q_\theta(x) \,||\, p(x|y)]$.
3. How to optimize?
   - Stochastic gradient descent.

Kullback-Leibler (KL) divergence

KL divergence from q(x) to p(x)

- Denoted by $KL[q(x) \| p(x)]$.
- $KL[q(x) \| p(x)] := \mathbb{E}_{q(x)}[\log(q(x)/p(x))]$.
- Average log ratio of two probabilities.
- Measures how q(x) is close to p(x).

KL divergence from q(x) to p(x)

- Denoted by KL[q(x) || p(x)].
- $KL[q(x) \| p(x)] := \mathbb{E}_{q(x)}[\log(q(x)/p(x))]$.
- Average log ratio of two probabilities.
- Measures how q(x) is close to p(x).

[Q1] What is KL[p(x) || p(x)]?

KL divergence from $q(x)$ to $p(x)$

- Denoted by $KL[q(x) \,||\, p(x)]$.
- $KL[q(x) \,||\, p(x)] := \mathbb{E}_{q(x)}[\log(q(x)/p(x))]$.
- Average log ratio of two probabilities.
- Measures how $q(x)$ is close to $p(x)$.

[Q1] What is $KL[p(x) \,||\, p(x)]$?

[A] 0

KL divergence from q(x) to p(x)

- Denoted by $KL[q(x) \,||\, p(x)]$.
- $KL[q(x) \,||\, p(x)] := \mathbb{E}_{q(x)}[\log(q(x)/p(x))]$.
- Average log ratio of two probabilities.
- Measures how q(x) is close to p(x).

[Q2] Prove $KL[p(x) \,||\, q(x)] \geq 0$. And .. $=0$ iff $p=q$.

KL divergence from q(x) to p(x)

- Denoted by $KL[q(x) \,||\, p(x)]$.
- $KL[q(x) \,||\, p(x)] := \mathbb{E}_{q(x)}[\log(q(x)/p(x))]$.
- Average log ratio of two probabilities.
- Measures how q(x) is close to p(x).

[Q2] Prove $KL[p(x) \,||\, q(x)] \geq 0$. And .. $=0$ iff $p=q$.

[Hint] $\mathbb{E}_{p(x)}[\log(f(x))] \leq \log(\mathbb{E}_{p(x)}[f(x)])$.

KL divergence

Function d : Pr(X) × Pr(X) $\to_{\text{partial}} \mathbb{R}$ s.t.

(1) d(p,q) = 0 iff p=q

(2) d(p,q) ≥ 0

Measures the similarity between p and q.

-
- $KL[q(x) \,||\, p(x)] := \mathbb{E}_{q(x)}[\log(q(x)/p(x))]$.
- Average log ratio of two probabilities.
- Measures how q(x) is close to p(x).

[Q2] Prove $KL[p(x) \,||\, q(x)] \geq 0$. And .. $=0$ iff p=q.

[Hint] $\mathbb{E}_{p(x)}[\log(f(x))] \leq \log(\mathbb{E}_{p(x)}[f(x)])$.

KL divergence from q(x) to p(x)

- Denoted by $KL[q(x) \,||\, p(x)]$.
- $KL[q(x) \,||\, p(x)] := \mathbb{E}_{q(x)}[\log(q(x)/p(x))]$.
- Average log ratio of two probabilities.
- Measures how q(x) is close to p(x).

[Q2] Prove $KL[p(x) \,||\, q(x)] \geq 0$. And .. $=0$ iff $p=q$.

[Hint] $\mathbb{E}_{p(x)}[\log(f(x))] \leq \log(\mathbb{E}_{p(x)}[f(x)])$.

KL divergence from q(x) to p(x)

- Denoted by KL[q(x) || p(x)].
- $KL[q(x) \,||\, p(x)] := \mathbb{E}_{q(x)}[\log(q(x)/p(x))]$.
- Average log ratio of two probabilities.
- Measures how q(x) is close to p(x).

[Q3] $KL[[0{\mapsto}.1;1{\mapsto}.8;2{\mapsto}.1] \,||\, [0{\mapsto}.5;1{\mapsto}.1;2{\mapsto}.4]]$?

KL divergence from q(x) to p(x)

- Denoted by KL[q(x) || p(x)].
- KL[q(x) || p(x)] := $\mathbb{E}_{q(x)}[\log(q(x)/p(x))]$.
- Average log ratio of two probabilities.
- Measures how q(x) is close to p(x).

[Q3] KL[[0$\mapsto$.1;1$\mapsto$.8;2$\mapsto$.1] || [0$\mapsto$.5;1$\mapsto$.1;2$\mapsto$.4]]?

[A] 1.36

KL divergence from q(x) to p(x)

- Denoted by KL[q(x) || p(x)].
- KL[q(x) || p(x)] := $\mathbb{E}_{q(x)}[\log(q(x)/p(x))]$.
- Average log ratio of two probabilities.
- Measures how q(x) is close to p(x).

[Q4] KL[[0↦.1;1↦.1;2↦.8] || [0↦.5;1↦.1;2↦.4]]?

[A] 1.36

KL divergence from q(x) to p(x)

- Denoted by $KL[q(x) \,||\, p(x)]$.
- $KL[q(x) \,||\, p(x)] := \mathbb{E}_{q(x)}[\log(q(x)/p(x))]$.
- Average log ratio of two probabilities.
- Measures how q(x) is close to p(x).

[Q4] $KL[[0 \mapsto .1; 1 \mapsto .1; 2 \mapsto .8] \,||\, [0 \mapsto .5; 1 \mapsto .1; 2 \mapsto .4]]$?

[A] 0.39

KL divergence from q(x) to p(x)

- Denoted by KL[q(x) || p(x)].
- $KL[q(x) \,||\, p(x)] := \mathbb{E}_{q(x)}[\log(q(x)/p(x))]$.
- Average log ratio of two probabilities.
- Measures how q(x) is close to p(x).

[Q5] $KL[[0\mapsto.8;1\mapsto.1;2\mapsto.1] \,||\, [0\mapsto.5;1\mapsto.1;2\mapsto.4]]$?

[A] 0.39

KL divergence from q(x) to p(x)

- Denoted by KL[q(x) || p(x)].
- KL[q(x) || p(x)] := $\mathbb{E}_{q(x)}[\log(q(x)/p(x))]$.
- Average log ratio of two probabilities.
- Measures how q(x) is close to p(x).

[Q5] KL[[0↦.8;1↦.1;2↦.1] || [0↦.5;1↦.1;2↦.4]]?

[A] 0.24

KL divergence from q(x) to p(x)

- Denoted by KL[q(x) || p(x)].
- $KL[q(x) \| p(x)] := \mathbb{E}_{q(x)}[\log(q(x)/p(x))]$.
- Average log ratio of two probabilities.
- Measures how q(x) is close to p(x).

[Q6] $KL[[0 \mapsto .3; 1 \mapsto .4; 2 \mapsto .3] \| [0 \mapsto .5; 1 \mapsto .1; 2 \mapsto .4]]$?

[A] 0.24

KL divergence from q(x) to p(x)

- Denoted by $KL[q(x) \| p(x)]$.
- $KL[q(x) \| p(x)] := \mathbb{E}_{q(x)}[\log(q(x)/p(x))]$.
- Average log ratio of two probabilities.
- Measures how q(x) is close to p(x).

[Q6] $KL[[0 \mapsto .3; 1 \mapsto .4; 2 \mapsto .3] \| [0 \mapsto .5; 1 \mapsto .1; 2 \mapsto .4]]$?

[A] 0.31

KL divergence from q(x) to p(x)

- $\mathrm{KL}[q(x) \,||\, p(x)] := \mathbb{E}_{q(x)}[\log(q(x)/p(x))]$.

[Q6] $\mathrm{KL}[[0 \mapsto .3; 1 \mapsto .4; 2 \mapsto .3] \,||\, [0 \mapsto .5; 1 \mapsto .1; 2 \mapsto .4]]$?

[A] 0.31

KL divergence from q(x) to p(x)

- $KL[q(x) \| p(x)] := \mathbb{E}_{q(x)}[\log(q(x)/p(x))]$.
- Asymmetric.

[Q6] $KL[[0 \mapsto .3; 1 \mapsto .4; 2 \mapsto .3] \| [0 \mapsto .5; 1 \mapsto .1; 2 \mapsto .4]]$?

[A] 0.31

KL divergence from q(x) to p(x)

- $KL[q(x) \mid\mid p(x)] := \mathbb{E}_{q(x)}[\log(q(x)/p(x))]$.
- Asymmetric.

[Q7] $KL[[0 \mapsto .5; 1 \mapsto .1; 2 \mapsto .4] \mid\mid [0 \mapsto .3; 1 \mapsto .4; 2 \mapsto .3]]$?

~~[Q6] $KL[[0 \mapsto .3; 1 \mapsto .4; 2 \mapsto .3] \mid\mid [0 \mapsto .5; 1 \mapsto .1; 2 \mapsto .4]]$?~~

[A] 0.31

KL divergence from q(x) to p(x)

- $KL[q(x) \,||\, p(x)] := \mathbb{E}_{q(x)}[\log(q(x)/p(x))]$.
- Asymmetric.

[Q7] $KL[[0 \mapsto .5; 1 \mapsto .1; 2 \mapsto .4] \,||\, [0 \mapsto .3; 1 \mapsto .4; 2 \mapsto .3]]$?

~~[Q6] $KL[[0 \mapsto .3; 1 \mapsto .4; 2 \mapsto .3] \,||\, [0 \mapsto .5; 1 \mapsto .1; 2 \mapsto .4]]$?~~

[A] 0.23

KL divergence from q(x) to p(x)

- $KL[q(x) \,||\, p(x)] := \mathbb{E}_{q(x)}[\log(q(x)/p(x))]$.
- Asymmetric.

[Q8] $KL[[0 \mapsto .5; 1 \mapsto .1; 2 \mapsto .4] \,||\, [0 \mapsto .8; 1 \mapsto .1; 2 \mapsto .1]]$?

~~[Q6] KL[[0↦.3;1↦.4;2↦.3] || [0↦.5;1↦.1;2↦.4]]?~~

[A] 0.23

KL divergence from q(x) to p(x)

- $\mathrm{KL}[q(x) \parallel p(x)] := \mathbb{E}_{q(x)}[\log(q(x)/p(x))]$.
- Asymmetric.

[Q8] $\mathrm{KL}[[0 \mapsto .5; 1 \mapsto .1; 2 \mapsto .4] \parallel [0 \mapsto .8; 1 \mapsto .1; 2 \mapsto .1]]$?

~~[Q6] KL[[0↦.3;1↦.4;2↦.3] || [0↦.5;1↦.1;2↦.4]]?~~

[A] 0.32

KL divergence from q(x) to p(x)

- $KL[q(x) \,||\, p(x)] := \mathbb{E}_{q(x)}[\log(q(x)/p(x))]$.
- Asymmetric.

KL[approx || true] - mode seeking rewarded.
KL[true || approx] - mode covering rewarded.

KL divergence from q(x) to p(x)

p = [0: 0.5, 1: 0.1, 2: 0.4]
q1 = [0: 0.7, 1: 0.2, 2: 0.1]
q2 = [0: 0.3, 1: 0.4, 2: 0.3]
q1 and q2.
KL[p||q1] = 0.32 KL[p||q2] = 0.23
KL[q1||p] = 0.24 KL[q2||p] = 0.31

- KL[q(x) || p(x)] := $\mathbb{E}_{q(x)}[\log(q(x)/p(x))]$.
- Asymmetric.

KL[approx || true] - mode seeking rewarded.
KL[true || approx] - mode covering rewarded.
[Q9] Find best q_i s.t KL[q_i||p] is minimum.
Do the same for KL[p||q_i].

KL divergence from q(x) to p(x)

- $KL[q(x) \,||\, p(x)] := \mathbb{E}_{q(x)}[\log(q(x)/p(x))]$.
- Asymmetric.
- Support requirement: $p(x)=0 \implies q(x)=0$.
- The closer to violation, the larger $KL[q||p]$.

Popular objective for variational inference

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

Encourage mode seeking. Leads to q_θ that approximates one mode of $p(x|y)$ well.

Easier to optimise than $\mathrm{KL}[p(x|y)||q_\theta(x)]$, since easier to draw samples from $q_\theta(x)$ than $p(x|y)$.

Basic (but not uncommon) choices.

1. Which family?
   - Mean field — $q_\theta(x_1, \ldots, x_n) = \prod_k q_\theta(x_k)$.
2. What does "closest" mean?
   - KL divergence — $KL[q_\theta(x) \,||\, p(x|y)]$.
3. How to optimize?
   - Stochastic gradient descent.

Gradient descent with estimated gradient

Gradient descent

Goal: $\mathrm{argmin}_\theta\ KL[q_\theta(x)\ ||\ p(x|y)]$.

- Hill-descending. Iteratively move θ towards the downward direction of the hill.
- "downward" means $\nabla_\theta\, KL[q_\theta(x)||p(x|y)]$.

Gradient descent

Pick θ_1

Gradient descent

Pick θ_1

$\theta_2 \leftarrow \theta_1 - \eta \times (\nabla_\theta \, KL[q_\theta(x) \mid\mid p(x|y)])_{\theta=\theta_1}$

Gradient descent

Pick θ_1

$\theta_2 \leftarrow \theta_1 - \eta \times (\nabla_\theta \, KL[q_\theta(x) \,||\, p(x|y)])_{\theta=\theta_1}$

$\boxed{\text{Gradient. Direction towards minimising KL at } \theta_1.}$

Gradient descent

Learning rate.
Some small positive number.

Pick θ_1

$$\theta_2 \leftarrow \theta_1 - \eta \times (\nabla_\theta \, KL[q_\theta(x) \,||\, p(x|y)])_{\theta=\theta_1}$$

Gradient.
Direction towards minimising KL at θ_1.

Gradient descent

Pick θ_1

$\theta_2 \leftarrow \theta_1 - \eta \times (\nabla_\theta \mathrm{KL}[q_\theta(x) \,||\, p(x|y)])_{\theta=\theta_1}$

$\theta_3 \leftarrow \theta_2 - \eta \times (\nabla_\theta \mathrm{KL}[q_\theta(x) \,||\, p(x|y)])_{\theta=\theta_2}$

Gradient descent

Pick θ_1

$\theta_2 \leftarrow \theta_1 - \eta \times (\nabla_\theta \, KL[q_\theta(x) \,||\, p(x|y)])_{\theta=\theta_1}$

$\theta_3 \leftarrow \theta_2 - \eta \times (\nabla_\theta \, KL[q_\theta(x) \,||\, p(x|y)])_{\theta=\theta_2}$

$\theta_4 \leftarrow \theta_3 - \eta \times (\nabla_\theta \, KL[q_\theta(x) \,||\, p(x|y)])_{\theta=\theta_3}$

...

$\theta_{n+1} \leftarrow \theta_n - \eta \times (\nabla_\theta \, KL[q_\theta(x) \,||\, p(x|y)])_{\theta=\theta_n}$

Gradient descent

Difficult to compute gradient analytically.

Pick θ_1

$\theta_2 \leftarrow \theta_1 - \eta \times (\nabla_\theta \mathrm{KL}[q_\theta(x) \,||\, p(x|y)])_{\theta=\theta_1}$

$\theta_3 \leftarrow \theta_2 - \eta \times (\nabla_\theta \mathrm{KL}[q_\theta(x) \,||\, p(x|y)])_{\theta=\theta_2}$

$\theta_4 \leftarrow \theta_3 - \eta \times (\nabla_\theta \mathrm{KL}[q_\theta(x) \,||\, p(x|y)])_{\theta=\theta_3}$

…

$\theta_{n+1} \leftarrow \theta_n - \eta \times (\nabla_\theta \mathrm{KL}[q_\theta(x) \,||\, p(x|y)])_{\theta=\theta_n}$

Stochastic gradient descent

Estimate the gradient using samples.

Stochastic gradient descent

Estimate the gradient using samples.

That is,

$$\nabla_\theta \, KL[q_\theta(x)||p(x|y)] \approx 1/N \times \textstyle\sum_{i=1..N} f(x_i,\theta)$$

where

1. samples $x_1, \ldots, x_N$ are drawn from q_θ;
2. function f is chosen by an estimator.

Stochastic gradient descent

Estimate the gradient using samples.

That is,

$$\nabla_\theta \mathrm{KL}[q_\theta(x)||p(x|y)] \approx 1/N \times \sum_{i=1..N} f(x_i,\theta)$$

where

1. samples $x_1, \ldots, x_N$ are drawn from q_θ;
2. function f is chosen by an estimator.

Stochastic gradient descent

Estimate the gradient using samples.

That is,

$$\nabla_\theta \, KL[q_\theta(x)||p(x|y)] \approx 1/N \times \sum_{i=1..N} f(x_i,\theta)$$

where

1. samples $x_1, \ldots, x_N$ are drawn from q_θ;
2. function f is chosen by an estimator.

$$\nabla_\theta \, KL[q_\theta(x) || p(x|y)] \approx 1/N \times \sum_{i=1..N} f(x_i, \theta)$$

for $x_1, \ldots, x_N$ drawn from q_θ.

$$\nabla_\theta \, KL[q_\theta(x)||p(x|y)] \approx 1/N \times \sum_{i=1..N} f(x_i,\theta)$$

for $x_1, \ldots, x_N$ drawn from q_θ.

Score estimator (aka REINFORCE):

$$f(x,\theta) = (\nabla_\theta \log(q_\theta(x))) \times \log(q_\theta(x)/p(x,y))$$

$$\nabla_\theta \, KL[q_\theta(x)||p(x|y)] \approx 1/N \times \sum_{i=1..N} f(x_i,\theta)$$

for $x_1, \ldots, x_N$ drawn from q_θ.

Score estimator (aka REINFORCE):

$$f(x,\theta) = (\nabla_\theta \log(q_\theta(x))) \times \log(q_\theta(x)/p(x,y))$$

1. Direction to increase probability of x

$$\nabla_\theta \, KL[q_\theta(x)||p(x|y)] \approx 1/N \times \sum_{i=1..N} f(x_i,\theta)$$

for $x_1, \ldots, x_N$ drawn from q_θ.

Score estimator (aka REINFORCE):

$$f(x,\theta) = (\nabla_\theta \log(q_\theta(x))) \times \log(q_\theta(x)/p(x,y))$$

1. Direction to increase probability of x weighted by its log ratio.

$$\nabla_\theta \, KL[q_\theta(x)||p(x|y)] \approx 1/N \times \sum_{i=1..N} f(x_i,\theta)$$

for $x_1, \ldots, x_N$ drawn from q_θ.

Score estimator (aka REINFORCE):

$$f(x,\theta) = (\nabla_\theta \log(q_\theta(x))) \times \log(q_\theta(x)/p(x,y))$$

1. Direction to increase probability of x weighted by its log ratio.
2. No need to know p(y).

$$\nabla_\theta \, KL[q_\theta(x)||p(x|y)] \approx 1/N \times \sum_{i=1..N} f(x_i,\theta)$$

for $x_1, \ldots, x_N$ drawn from q_θ.

Score estimator (aka REINFORCE):

$$f(x,\theta) = (\nabla_\theta \log(q_\theta(x))) \times \log(q_\theta(x)/p(x,y))$$

1. Direction to increase probability of x weighted by its log ratio.
2. No need to know p(y).
3. p(x,y) don't have to be differentiable wrt. x.

Score estimator

Goal: Estimate $\nabla_\theta \, KL[q_\theta(x) \,||\, p(x|y)]$

1. Sample $x_1, \ldots, x_N$ from $q_\theta(x)$.
2. Return:

$$1/N \times \sum_{i=1..N} (\nabla_\theta \log(q_\theta(x_i)) \times \log(q_\theta(x_i)/p(x,y))$$

Score estimator

Do the most of the derivation, but leave some part to the students.

Goal: Estimate $\nabla_\theta \, KL[q_\theta(x) \,||\, p(x|y)]$

1. Sample $x_1, \ldots, x_N$ from $q_\theta(x)$.
2. Return:

$$1/N \times \sum_{i=1..N} (\nabla_\theta \log(q_\theta(x_i)) \times \log(q_\theta(x_i)/p(x,y))$$

[Q] Prove that this estimator is unbiased. That is,

$$\mathbb{E}[1/N \times \sum_{i=1..N} (\nabla_\theta \log(q_\theta(x_i)) \times \log(q_\theta(x_i)/p(x,y))]$$

$$= \nabla_\theta \, KL[q_\theta(x) \,||\, p(x|y)]$$

Implementing this for probabilistic programs

- Blackboard lecture.
- Look at the last part of Note6.pdf (for operational semantics) in the webpage.

Reference

1. Ranganath et al.'s AISTATS14 paper "Black-Box Variational Inference" (https://arxiv.org/abs/1401.0118).
2. Section 7 of Tolpin et al.'s ITL'16 paper "Design and implementation of probabilistic programming language Anglican" (https://arxiv.org/abs/1608.05263)