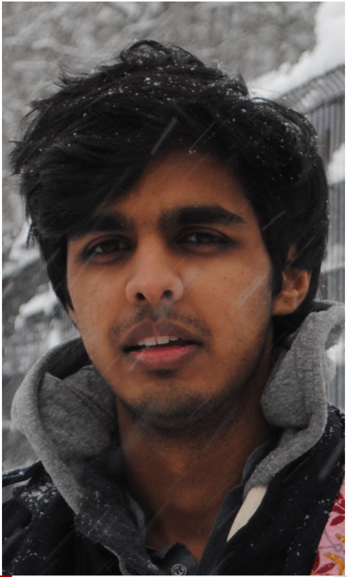
# An MCMC library for probabilistic programming

Rob Zinkov

June 13th, 2014

# Special Thanks to Praveen



# Why we need it?

- Prototyping probabilistic programming inference solutions
- Easier exploration of mcmc algorithms
- Easier to combine multiple mcmc strategies

# Acceptance ratios tricky to get right

$$\mathcal{A}\left(x^{(i)}, x^{\star}\right) = \min\left\{1, \frac{p(x^{\star})q\left(x^{(i)} \mid x^{\star}\right)}{p\left(x^{(i)}\right)q\left(x^{\star}|x^{(i)}\right)}\right\}$$

# Reversible-jump: trickier still

$$\mathcal{A}_{n\to m} = \min\left\{1, \frac{p(m, x_m^\star)}{p(n, x_n)} \times \frac{q(n \mid m)}{q(m \mid n)} \times \frac{q_{m\to n}(u_{m,n} \mid m, x_m^\star)}{q_{n\to m}(u_{n,m} \mid n, x_n)} \times \mathcal{J}_{f_{n\to m}}\right\}$$

# Split-merge proposals

$$\mathcal{A}_{split} = \min\left\{1, \frac{p(k+1, \mu_{k+1})}{p(k, \mu_k)} \times \frac{\frac{1}{k+1}}{\frac{1}{k}} \times \frac{1}{p(u_{n,m})} \times \mathcal{J}_{split}\right\}$$

$$\mathcal{A}_{merge} = \min\left\{1, \frac{p(k-1, \mu_{k-1})}{p(k, \mu_k)} \times \frac{\frac{1}{k-1}}{\frac{1}{k}} \times \mathcal{J}_{merge}\right\}$$

# Caveats

- We are only talking MCMC and no other inference methods
- We will not discuss how to use this library in a probabilistic programming system

# Core Primitives

# Providing a Density

```
type Density a = a -> Probability
data Target a = T (Density a)
```

# Providing a Proposal Distribution

```
type Sample a = Rand -> IO a
data Proposal a = P (Density a) (Sample a)
```

# Specifying a Step (Transition)

Steps are how we transition from one state to another

```
type Step x = Rand -> x -> IO x
```

# Specifying a Kernel

```
type Kernel x a = Target a ->
                  (a -> Proposal a) ->
                  Step x
```

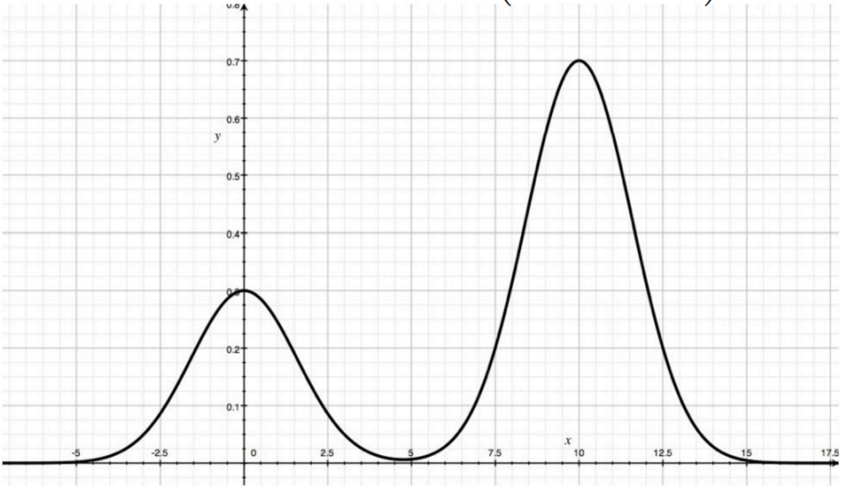
# Walking the MCMC chain

```
walk :: Step x ->
        x ->
        Int ->
        Rand ->
        Action x a ->
        IO a
```

# Demo

Demo!

$$0.3\exp\left(-0.2x^2\right) + 0.7\exp\left(-0.2\left(x-10\right)^2\right)$$



# Features we provide

- Blocking proposals
- Cyclic kernels
- Mixture kernels

# Further work

- Langevin and Hamiltonian MC
- Approximate MCMC (ABC, Noisy MALA)
- Adaptive MC
- Reversible Jump

# Conclusions

Let's write our inference solutions in more modular ways
Coming very soon to Hackage!

Questions?