Metropolis

Random walk

Useful links:

http://www.mit.edu/~ilkery/papers/MetropolisHastingsSampling.pdf

Initialize $x^{(0)} \sim q(x)$

for iteration $i = 1, 2, \ldots$ **do**

Propose: $x^{cand} \sim q(x^{(i)}|x^{(i-1)})$

Acceptance Probability:

$$\alpha(x^{cand}|x^{(i-1)}) = \min\left\{1, \frac{q(x^{(i-1)}|x^{cand})\pi(x^{cand})}{q(x^{cand}|x^{(i-1)})\pi(x^{(i-1)})}\right\}$$

$u \sim \text{Uniform}\ (u; 0, 1)$

if $u < \alpha$ **then**

Accept the proposal: $x^{(i)} \leftarrow x^{cand}$

else

Reject the proposal: $x^{(i)} \leftarrow x^{(i-1)}$

end if

end for

Algorithm

First Step: $\text{Initialize } x^{(0)} \sim q(x)$
Start from a random position, maybe x = 0.

```
In [35]: hypothesis_list = []
         hypothesis_list.append(0)
```

In the for loop...

Second Step: Propose: $x^{cand} \sim \boxed{q(x^{(i)}|x^{(i-1)})}$

Add some noise to this x as the next candidate.

called a **proposal distribution**
In our assignment:
w_candidate = w + Normal(0,1)

```
w = hypothesis_list[-1]
print("W is : ", w)
w_candidate = hypothesis_list[-1] + np.random.normal(0, 0.1)
print("W' is : ", w_candidate)
```

In the for loop...

Third Step:
Calculate the acceptance rate.

Acceptance Probability:

$$\alpha(x^{cand}|x^{(i-1)}) = \min\left\{1, \frac{q(x^{(i-1)}|x^{cand})\pi(x^{cand})}{q(x^{cand}|x^{(i-1)})\pi(x^{(i-1)})}\right\}$$

In our assignment:
α =P(W'|D) / P(W|D)
this is a posterior

Problem 1: why log? What should be log?

```
def log_prior(W) :
    return log(exp(-W))
```

turns this to:

```
ratio = exp((log_posterior(w_next))-(log_posterior(w)))
```

$$\log P(W \mid a, n_1, n_2) = \log P(W) + \sum_{i=1}^{k} \log P(a_i \mid W, n_{1i}, n_{2i})$$

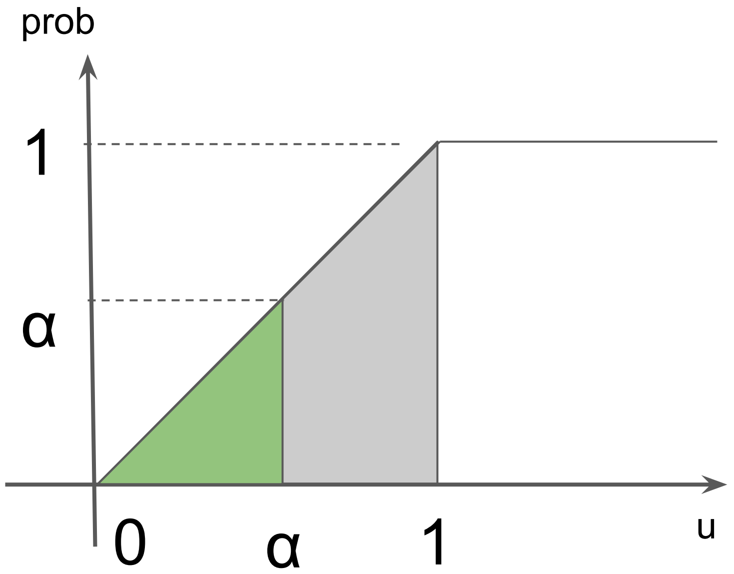
```
def log_posterior(W):
    return log_prior(W) + log_likelihood(data['n1'], data['n2'], data['correct'], W)
```

In the for loop...

Fourth Step: Accept with probability α

Why accept it with probability α...

$u \sim \text{Uniform}\ (u; 0, 1)$
if $u < \alpha$ **then**
  Accept the proposal: $x^{(i)} \leftarrow x^{cand}$
else
  Reject the proposal: $x^{(i)} \leftarrow x^{(i-1)}$
end if



Ratio is like a probability for whether we update.

- If ratio > 1, you will definitely accept it.
- If ratio < 1, you will accept it based on the value of u generated from a uniform distribution.

```
u= np.random.uniform(0,1)
if ratio > u :
    #accept w_candidate
else :
    #accept w
```

Problem 3

Correctly handles cases when W<0?

- What is W? Refer to Weber's Law.
- W < 0, what would prior probability be? P(W) = 0
- How to modify this code?

```
def prior(x) :
    if x < 0:
        return
    else:
        return log(exp(-x))
```

Problem 4: Organize everything together...

```
for i in range(0, 10) :
    w_current = hypothesis_list[-1]
    #print("W is : ", w_current)
    w_next = hypothesis_list[-1] + np.random.normal(0, 0.1)
    #print("W' is : ", w_next)

    if ratio of the two probabilities > 1:
        hypothesis_list.append(w_next)
        posterior_score.append(?)
    else :
        rand = np.random.uniform(0,1)
        ratio = ratio of the two probabilities ## Please pay extra atten
tion here on how to
        # calculate the two probabilities, where to take an exponential,
        # and how using the log probabilities changes things
        # Hint: log(a/b) = log(a) - log(b)
        if ratio > rand :
            hypothesis_list.append(w_next)
            posterior_score.append(?)
        else :
            hypothesis_list.append(w_current)
            posterior_score.append(?)
```

How many times to run?
For me, I run 11000 because of 'burn in'.

Problem 5

“ Look at the number of times W falls in the range [0.2, 0.3] and divide by the number of Ws calculated”.

Problem 6

- How to create two threads?

Think about:

for i in range(2):
    Run the Metropolis Code

- Not sure how to determine whether it is a good model. I'll check with Sam at his office hour
- Drop me an email and I'll get back to you.