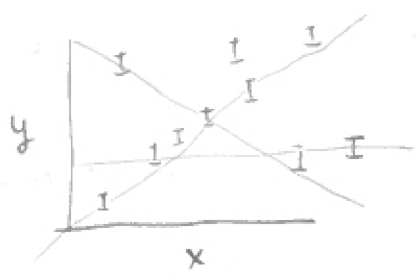
Last time

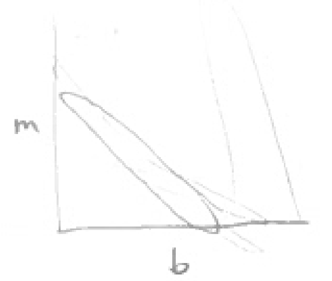
- we set up a foreground-background model:
  - data points can either be drawn from the real phenomenom we're interested in (fg) OR a "bad" distribution (bg).
  - we fit for parameters describing both fg & bg and the "mixture weights" (P_{bad}, $1-P_{bad}$)

$$p(y_i | m, b, P_{bad}, Y, V) = P_{bad} \cdot \frac{1}{\sqrt{2\pi(V+\sigma_i^2)}} \exp\left(-\frac{(y_i - Y)^2}{2(V+\sigma_i^2)}\right)$$

$$+ (1-P_{bad}) \frac{1}{\sqrt{2\pi}\,\sigma_i} \exp\left(-\frac{(y_i - m x_i - b)^2}{2\sigma_i^2}\right)$$

- this is a 5-parameter model
  - → hard to explore by gridding up & plotting
  - → we used Julia's <u>optimize</u> function to find maximum-likelihood (but we want the covariance
  - → but we found that it was multimodal

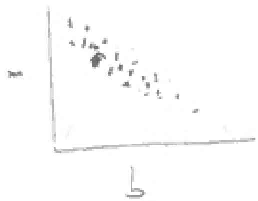




Today - Markov Chain Monte Carlo

The problem it solves:

- You have a probability distribution function with some parameters
- You want a sampling of parameter values that are fair draws from the distribution



$\theta = (b, m)$

The Idea:

- move a "particle" around in the parameter space.
- move the particle according to a "proposal distrib"

$$\theta_{new} = \theta + N(0, \sigma_\theta^2)$$

- compute the probability at the new parameter

$$P_{new} = P(\theta_{new})$$

- always keep improvements; sometimes keep moves that reduce the $P(\theta_{new})$
- in the long run, particle positions are your samples.

Markov Chain Monte Carlo

- list of samples
- jump from θ_i to θ_{i+1} depends only on θ_i
- the jump involves randomness.

Algorithm - Metropolis-Hastings

proposal

\# params

```
chain = []
params = [150., 2.]
jumpsizes = [10, 0.1]

prob = prob_function(params)

for i in 1:Nsteps
    params_new = params .+ randn(2) .* jumpsizes

    prob_new = prob_function(params_new)

    if (prob_new / prob ≥ rand(Float64))
        params = params_new
        prob = prob_new
    end
    append!(chain, params)

return chain
```

accept?

uniform [0,1]