Mixture Models and the Expectation Maximization Algorithm

Presentation for the Mathematics of Artificial Intelligence and Machine Learning at the University of Denver

Evans Hedges

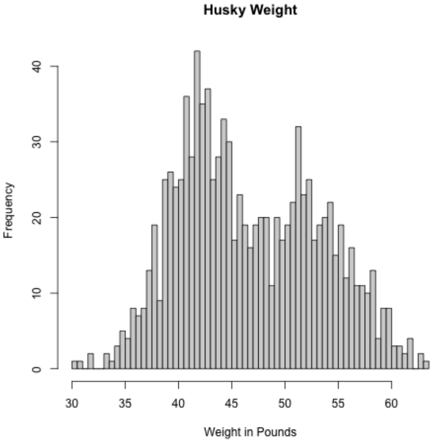
February 13th 2023

Presentation Outline

- Example Problem
- General Mixture Model Description
- Expectation Maximization Algorithm
- Implementation and Results

Husky Mixture Model

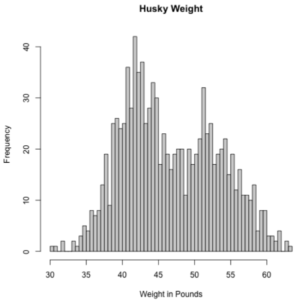
Scenario: Since we are in Colorado, we have access to veterinary records showing the bodyweight of a large number of adult huskies.



Husky Mixture Model

We want to use this data to infer the following:

- The average and variance of male husky weight
- The average and variance of female husky weight
- The percent of Colorado huskies that are Male/Female
- Given a husky's weight, what is the probability that it is a Male or Female.



Husky Mixture Model

We will do this by assuming weight is normally distributed and modeling the weight distribution of the total population as follows:

$$X = \pi_1 N(\mu_1, \sigma_1) + \pi_2 N(\mu_2, \sigma_2)$$

$$\text{Model Parameters: } \boldsymbol{\theta} = (\pi_1, \mu_1, \sigma_1, \pi_2, \mu_2, \sigma_2)$$

$$\pi_1 + \pi_2 = 1$$

Define $f_1(\cdot|\theta_1)$ to be the probability density function of a normal distribution given parameters $\theta_1 = (\mu_1, \sigma_1)$ (and similarly for f_2).

General Mixture Model

In general, a mixture model is given by the following parameters:

- $\{f_1(x,\theta_1),\ldots,f_n(x,\theta_n)\}$ a collection probability density functions f_i with parameters θ_i representing the density function of a given sub-population (or state).
- $\pi = (\pi_1,\ldots,\pi_n)$ where π_i is the prior probability that some observation comes from state i.

The mixture model density function given parameters $\boldsymbol{\theta}$ is then:

$$f(x,\boldsymbol{\theta}) = \sum_{i=1}^{n} \pi_i f_i(x,\theta_i)$$

Likelihood

A probability density function $f(x, \theta)$ depends on

- the realization of a random variable $x = X(s)$ and
- the model parameters θ.

When viewing f as a function of θ where x is fixed, we call this the likelihood.

$$L(\theta|x) = f(x, \theta)$$

Under the assumption that a collection of observed variables $y_{1:T}$ are independent, likelihood is multiplicative:

$$L(\theta|y_{1:T}) = \prod_{t=1}^{T} L(\theta|y_t)$$

Mixture Model Likelihood

In the general mixture model the likelihood of $\boldsymbol{\theta}$ given some dataset $y_{1:T}$ (assumed to be independent) is given by:

$$L(\boldsymbol{\theta}|y_{1:T}) = \prod_{t=1}^{T} \sum_{i=1}^{N} \pi_i f_i(y_t|\theta_i)$$

It is frequently more computationally useful to compute log likelihood, which in our case is:

$$l(\boldsymbol{\theta}|y_{1:T}) = \sum_{t=1}^{T} \log(\pi_1 f_1(y_t|\theta_1) + \pi_2 f_2(y_t|\theta_2))$$

Maximum Likelihood Estimate

Our goal is to identify:

$$\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} l(\boldsymbol{\theta}|y_{1:T})$$

$$l(\boldsymbol{\theta}|y_{1:T}) = \sum_{t=1}^{T} \log(\pi_1 f_1(y_t|\theta_1) + (1-\pi_1) f_2(y_t|\theta_2))$$

$$= \sum_{t=1}^{T} \log\left(\pi_1 \frac{e^{-(y_t-\mu_1)^2/2\sigma_1^2}}{\sigma_1\sqrt{2\pi}} + (1-\pi_1)\frac{e^{-(y_t-\mu_2)^2/2\sigma_2^2}}{\sigma_2\sqrt{2\pi}}\right)$$

Maximum Likelihood Estimate

We have a closed form formula that is differentiable in every parameter we want to maximize.... What does that smell like?

$$\sum_{t=1}^{T} \log \left(\pi_1 \frac{e^{-(y_t-\mu_1)^2/2\sigma_1^2}}{\sigma_1\sqrt{2\pi}} + (1-\pi_1)\frac{e^{-(y_t-\mu_2)^2/2\sigma_2^2}}{\sigma_2\sqrt{2\pi}} \right)$$

Gradient Descent!

Great! Let's run some code and see what we get.

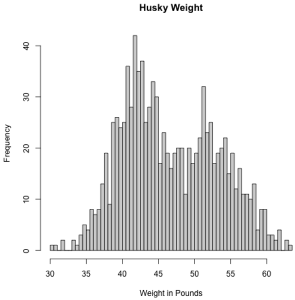
```r
# Learning Rate
delta = 0.01
log_likelihood_tracker = c()

# Stochastic Gradient Ascent
for (k in 1:10000){
  # Sample 5% of the dataset
  stochastic_sample = sample(x, floor(length(x) * 0.05), replace = FALSE)
  # Store each updated parameter
  newpi = pi + delta * partial_pi1(pi, muhat1, sigmahat1, muhat2, sigmahat2, stochastic_sample)
  newmu1 = muhat1 + delta * partial_mu1(pi, muhat1, sigmahat1, muhat2, sigmahat2, stochastic_sample)
  newsigma1 = sigmahat1 + delta * partial_sigma1(pi, muhat1, sigmahat1, muhat2, sigmahat2, stochastic_sample)
  newmu2 = muhat2 + delta * partial_mu2(pi, muhat1, sigmahat1, muhat2, sigmahat2, stochastic_sample)
  newsigma2 = sigmahat2 + delta * partial_sigma2(pi, muhat1, sigmahat1, muhat2, sigmahat2, stochastic_sample)


  # Update parameters
  pi = newpi
  muhat1 = newmu1
  sigmahat1 = newsigma1
  muhat2 = newmu2
  sigmahat2 = newsigma2

  # Track log likelihood of current iteration
  log_likelihood_tracker = append(log_likelihood_tracker, log_likelihood(pi, muhat1, sigmahat1, muhat2, sigmahat2, x))
}
```

Gradient Descent Results

We are left with the following model:

$$X = 99.3N(46.8, 36.5) - 98.3N(55.6, 78.6)$$

Hmm..



Expectation Maximization

Algorithm:

- Initialize $\boldsymbol{\theta}_1$.
- Iterate:
  - Expectation Step
  - Maximization Step
  - Update $\boldsymbol{\theta}_{n+1}$
- Halt when $||\boldsymbol{\theta}_{n+1} - \boldsymbol{\theta}_n|| < \epsilon$

Expectation Maximization

Expectation Step:

For each datapoint y_t, compute the probability that y_t comes from state i.

$$\gamma_i(t) = P(S = i|y_t) = \frac{\pi_i f_i(y_t|\theta_i)}{\sum_{j=1}^{N} \pi_j f_j(y_t|\theta_i)}$$

Expectation Maximization

Maximization Step:

Update model parameters to maximize the log likelihood of our new expected assignments.

$$\pi_i = \frac{\sum_{t=1}^{T} \gamma_i(t)}{N}$$

$$\mu_i = \frac{\sum_{t=1}^{T} \gamma_i(t) y_t}{\sum_{t=1}^{T} \gamma_i(t)}$$

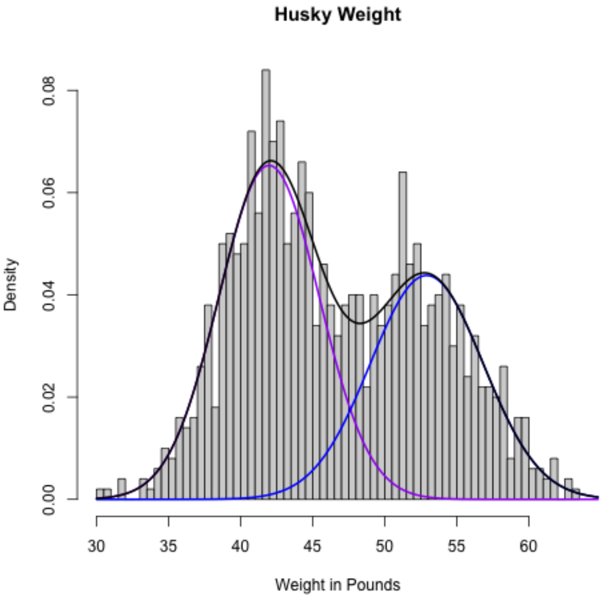
$$\sigma_i = \sqrt{\frac{\sum_{t=1}^{T} \gamma_i(t)(y_t - \mu_i)^2}{\sum_{t=1}^{T} \gamma_i(t)}}$$

Expectation Maximization

Let's run some code, take 2!

```r
# Initialize Parameters
pi = 0.5
muhat1 = 35
sigmahat1 = 1
muhat2 = 60
sigmahat2 = 1

# Track expectations
expectations = rep(0, length(x))

counter = 0
while(TRUE){
  counter = counter + 1
  # Expectation Step
  expectations = compute_expectations(x, pi, muhat1, sigmahat1, muhat2, sigmahat2)

  newpi = max_pi(expectations)
  newmu1 = max_mu1(x, expectations)
  newmu2 = max_mu2(x, expectations)
  newsigma1 = max_sigma1(x, expectations, newmu1)
  newsigma2 = max_sigma2(x, expectations, newmu2)

  norm_diff = 0
  norm_diff = sqrt((newpi-pi)^2 + (newmu1 - muhat1)^2 + (newmu2 - muhat2)^2 + (newsigma1 - sigmahat1)^2 + (newsigma2 - sigmahat2)^2)
  pi = newpi
  muhat1 = newmu1
  muhat2 = newmu2
  sigmahat1 = newsigma1
  sigmahat2 = newsigma2
  if(norm_diff < epsilon){
    break
  }

  if (counter > 100){
    break
  }
}
```

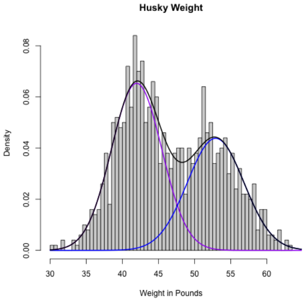
Expectation Maximization



Expectation Maximization

Final Results: $X = 0.57N(41.9, 3.48) + 0.43N(52.9, 3.9)$

Convergence within $\epsilon = 0.1$ in only 4 steps.
Actual values used to generate the data:
$0.57N(42, 3.5) + 0.43N(53, 4)$

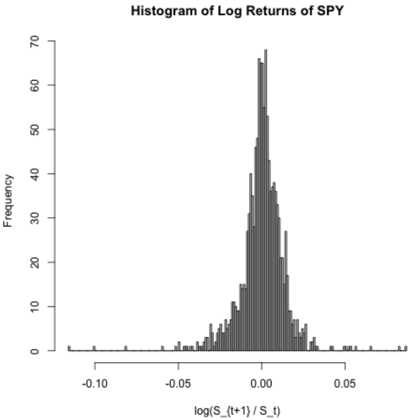


Mixture Model with Expectation Maximization Recap

- Start with a probability distribution that is made of a mixture of individual population distributions.
- Identify/estimate the number and type of underlying distributions
- Initialize model parameters
- Iterate the EM Algorithm:
  - Expectation Step (Update our state expectation for each datapoint)
  - Maximization Step (Given our new expectations, update model parameters)

S&P500 Daily Returns

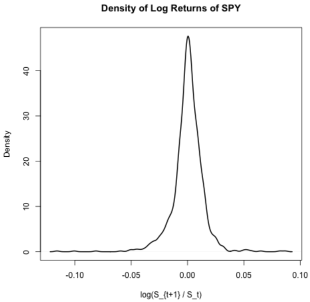
We now look at the log daily returns of the S&P 500 from January 2018 through January 2023.



S&P500 Daily Returns

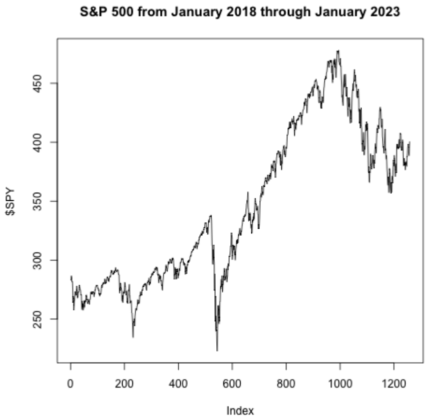
Is this normal?

Shapiro-Wilk test says NO! $p = 2.2 \times 10^{-16}$



S&P500 Daily Returns

If we take a look at the original data as a time series, it looks like there could be various periods of different kinds of returns (Bull vs Bear Markets)



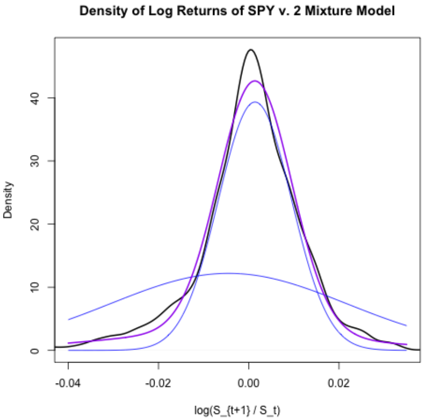
S&P500 Daily Returns

Using the **lca()** function of the **hmmr** package, we can quickly test mixture models following 1, 2, and 3 different Gaussian distributions.

	Log Likelihood
1 Parameter	3607
2 Parameters	3789
3 Parameters	3807

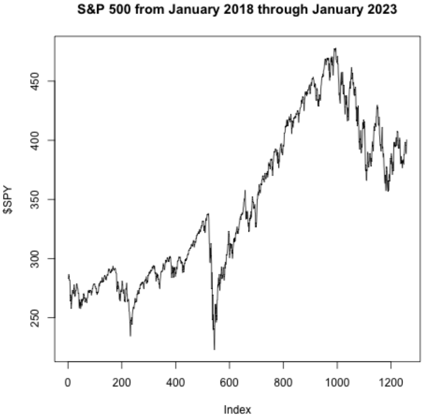
S&P500 Daily Returns

$$X = 0.811N(0.001387, 0.008114) + 0.189N(-0.004486, 0.02617)$$



S&P500 Daily Returns

We can do better! Datapoints are not independent.



Thank You

On February 27th we will discuss how to incorporate time dependence using Hidden Markov Models.

References:

- "Mixture and Hidden Markov Models with R" - Visser and Speekenbrink, 2022.
- https://github.com/depmix/hmmr