Bayesian free-knot splines

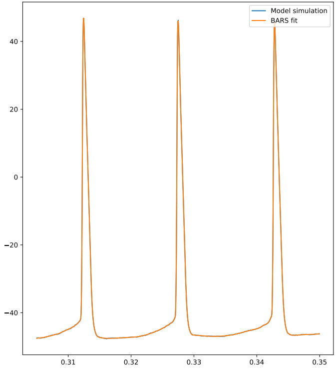
Mark Blyth

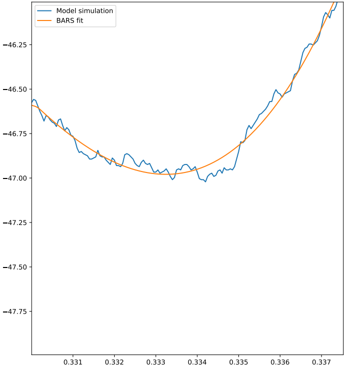
Week's goals

- Make changes to continuations paper
  - Looked at feedback, haven't started making changes yet
- Fix MSPE downsampling errors
  - Haven't fixed this yet
- Implement and test free-knot splines
  - Learn how it works
  - Code it up
  - Use it to validate splines method

Bayesian free-knot splines

Identified as being a good method for modelling neuron data





How it works

1. Assume a spline model fits the data
2. Find a distribution over spline models, given the data
3. Condition on this distribution with new data, to get posterior estimates

Step 0: problem setup

- Take some data $Y_i = g(x_i) + \varepsilon$
  - $\varepsilon \sim \mathcal{N}\{0, \sigma^2\}$
  - σ unknown
  - g unknown
  - Y_i random variables
  - Then, $Y_i | x_i, \sigma \sim \mathcal{N}\{g(x_i), \sigma^2\}$

- Goal: estimate g from noisy samples (x_i, Y_i)

This is a very standard problem formulation so far. . .

Step 1: spline model

- Model latent function g as being some piecewise-polynomial function f
- Tie polynomials together at knot-points ξ_i

$$f(x) = \begin{cases} f_1(x)\ , & x \in [a, \xi_0) \\ f_2(x)\ , & x \in [\xi_0, \xi_1) \\ \dots & \\ f_{k+2}\ , & x \in [\xi_k, b] \end{cases} \tag{1}$$

- $f_i(x)$ is an $\mathcal{O}(3)$ polynomial passing through $(\xi_{i-1}, g(\xi_{i-1}))$, $(\xi_i, g(\xi_i))$
  - . . . or allow the polynomials to pass *near* the knot-points, for smoothing splines

Spline bases

- Calculating the coefficients for each f_i is inconvenient

Spline bases

- Calculating the coefficients for each f_i is inconvenient
- Nicer approach:

Spline bases

- Calculating the coefficients for each f_i is inconvenient
- Nicer approach:
  - Let $f(x) = \sum_i \beta_i b_i(x)$

Spline bases

- Calculating the coefficients for each f_i is inconvenient
- Nicer approach:
  - Let $f(x) = \sum_i \beta_i b_i(x)$
  - Functions b_i form a basis for some spline $f(x)$ with k knots at $\{\xi_0, \ldots, \xi_k\}$

Spline bases

- Calculating the coefficients for each f_i is inconvenient
- Nicer approach:
  - Let $f(x) = \sum_i \beta_i b_i(x)$
  - Functions b_i form a basis for some spline $f(x)$ with k knots at $\{\xi_0, \ldots, \xi_k\}$
- b_i are found by

Spline bases

- Calculating the coefficients for each f_i is inconvenient
- Nicer approach:
  - Let $f(x) = \sum_i \beta_i b_i(x)$
  - Functions b_i form a basis for some spline $f(x)$ with k knots at $\{\xi_0, \ldots, \xi_k\}$
- b_i are found by
  - Specifying knot locations

Spline bases

- Calculating the coefficients for each f_i is inconvenient
- Nicer approach:
  - Let $f(x) = \sum_i \beta_i b_i(x)$
  - Functions b_i form a basis for some spline $f(x)$ with k knots at $\{\xi_0, \ldots, \xi_k\}$
- b_i are found by
  - Specifying knot locations
  - Requiring $\mathcal{C}^1$ smoothness

Spline bases

- Calculating the coefficients for each f_i is inconvenient
- Nicer approach:
  - Let $f(x) = \sum_i \beta_i b_i(x)$
  - Functions b_i form a basis for some spline $f(x)$ with k knots at $\{\xi_0, \ldots, \xi_k\}$
- b_i are found by
  - Specifying knot locations
  - Requiring $\mathcal{C}^1$ smoothness
  - Assuming linearity outside of data range

Spline bases

- Calculating the coefficients for each f_i is inconvenient
- Nicer approach:
  - Let $f(x) = \sum_i \beta_i b_i(x)$
  - Functions b_i form a basis for some spline $f(x)$ with k knots at $\{\xi_0, \ldots, \xi_k\}$
- b_i are found by
  - Specifying knot locations
  - Requiring $\mathcal{C}^1$ smoothness
  - Assuming linearity outside of data range
- b_i are called basis splines

Spline bases

- Calculating the coefficients for each f_i is inconvenient
- Nicer approach:
  - Let $f(x) = \sum_i \beta_i b_i(x)$
  - Functions b_i form a basis for some spline $f(x)$ with k knots at $\{\xi_0, \ldots, \xi_k\}$
- b_i are found by
  - Specifying knot locations
  - Requiring $\mathcal{C}^1$ smoothness
  - Assuming linearity outside of data range
- b_i are called basis splines
  - Our model now becomes $Y_i | x_i, \beta, \sigma, \xi \sim \mathcal{N}\{\sum_i \beta_i b_i(x_i), \sigma^2\}$

Easy approach

- Choose a nice number of knots k

Downside: bad choices for any of these parameters will give bad results:

Easy approach

- Choose a nice number of knots k
- Choose a uniformly spaced knot-set ξ

Downside: bad choices for any of these parameters will give bad results:

Easy approach

- Choose a nice number of knots k
- Choose a uniformly spaced knot-set ξ
- Guess σ

Downside: bad choices for any of these parameters will give bad results:

Easy approach

- Choose a nice number of knots k
- Choose a uniformly spaced knot-set ξ
- Guess σ
- Find a MLE for β

Downside: bad choices for any of these parameters will give bad results:

Easy approach

- Choose a nice number of knots k
- Choose a uniformly spaced knot-set ξ
- Guess σ
- Find a MLE for β

Downside: bad choices for any of these parameters will give bad results:

- Too few knots = underparameterised = can't capture shape of data

Easy approach

- Choose a nice number of knots k
- Choose a uniformly spaced knot-set ξ
- Guess σ
- Find a MLE for β

Downside: bad choices for any of these parameters will give bad results:

- Too few knots = underparameterised = can't capture shape of data
- Too many knots = overparameterised = overfit data and capture noise

Easy approach

- Choose a nice number of knots k
- Choose a uniformly spaced knot-set ξ
- Guess σ
- Find a MLE for β

Downside: bad choices for any of these parameters will give bad results:

- Too few knots = underparameterised = can't capture shape of data
- Too many knots = overparameterised = overfit data and capture noise
- Bad knot positioning = function can't adapt to changing rates

Step 2: find a distribution over models

- Specify a prior belief $\pi_k(k)$ for the numer of knots we have

Joint probability: $p(k, \xi, \beta, \sigma, y) = p(y|\beta, \sigma)\pi_\sigma(\sigma)\pi_\beta(\beta|\sigma, \xi, k)\pi_\xi(\xi|k)\pi_k(k)$
We can evaluate all of this!

Step 2: find a distribution over models

- Specify a prior belief $\pi_k(k)$ for the numer of knots we have
  - Eg. discrete uniform on $[k_{min}, k_{max}]$

Joint probability: $p(k, \xi, \beta, \sigma, y) = p(y|\beta, \sigma)\pi_\sigma(\sigma)\pi_\beta(\beta|\sigma, \xi, k)\pi_\xi(\xi|k)\pi_k(k)$
We can evaluate all of this!

Step 2: find a distribution over models

- Specify a prior belief $\pi_k(k)$ for the numer of knots we have
  - Eg. discrete uniform on $[k_{min}, k_{max}]$
- Specify a prior belief $\pi_\xi(\xi|k)$ on the knot positions ξ, for any given number of knots

Joint probability: $p(k, \xi, \beta, \sigma, y) = p(y|\beta, \sigma)\pi_\sigma(\sigma)\pi_\beta(\beta|\sigma, \xi, k)\pi_\xi(\xi|k)\pi_k(k)$
We can evaluate all of this!

Step 2: find a distribution over models

- Specify a prior belief $\pi_k(k)$ for the numer of knots we have
  - Eg. discrete uniform on $[k_{min}, k_{max}]$
- Specify a prior belief $\pi_\xi(\xi|k)$ on the knot positions ξ, for any given number of knots
  - Eg. uniform on range of data

Joint probability: $p(k, \xi, \beta, \sigma, y) = p(y|\beta, \sigma)\pi_\sigma(\sigma)\pi_\beta(\beta|\sigma, \xi, k)\pi_\xi(\xi|k)\pi_k(k)$
We can evaluate all of this!

Step 2: find a distribution over models

- Specify a prior belief $\pi_k(k)$ for the numer of knots we have
  - Eg. discrete uniform on $[k_{min}, k_{max}]$
- Specify a prior belief $\pi_\xi(\xi|k)$ on the knot positions ξ, for any given number of knots
  - Eg. uniform on range of data
- Specify a prior belief $\pi_\sigma(\sigma)$ on the noise level

Joint probability: $p(k, \xi, \beta, \sigma, y) = p(y|\beta, \sigma)\pi_\sigma(\sigma)\pi_\beta(\beta|\sigma, \xi, k)\pi_\xi(\xi|k)\pi_k(k)$
We can evaluate all of this!

Step 2: find a distribution over models

- Specify a prior belief $\pi_k(k)$ for the numer of knots we have
  - Eg. discrete uniform on $[k_{min}, k_{max}]$
- Specify a prior belief $\pi_\xi(\xi|k)$ on the knot positions ξ, for any given number of knots
  - Eg. uniform on range of data
- Specify a prior belief $\pi_\sigma(\sigma)$ on the noise level
- Specify a prior on β

Joint probability: $p(k, \xi, \beta, \sigma, y) = p(y|\beta, \sigma)\pi_\sigma(\sigma)\pi_\beta(\beta|\sigma, \xi, k)\pi_\xi(\xi|k)\pi_k(k)$
We can evaluate all of this!

Bayesian approach

- We want to know where to put the knots
- Bayesian approach: find the posterior knot distribution $p(k, \xi | y)$

$$p(k, \xi | y) = \frac{p(k, \xi, y)}{p(y)} \,, \tag{2}$$

$$p(k, \xi, y) = \int \int p(k, \xi, \beta, \sigma, y) \mathrm{d}\beta \mathrm{d}\sigma \tag{3}$$

$$= \int \int p(y | \beta, \sigma) \pi_\sigma(\sigma) \pi_\beta(\beta | \sigma, \xi, k) \pi_\xi(\xi | k) \pi_k(k) \mathrm{d}\sigma \mathrm{d}\beta \tag{4}$$

Bayesian approach

Putting it together, we get

$$p(k,\xi|y) = \frac{\int\int p(y|\beta,\sigma)\pi_\sigma(\sigma)\pi_\beta(\beta|\sigma,\xi,k)\pi_\xi(\xi|k)\pi_k(k)\mathrm{d}\sigma\mathrm{d}\beta}{p(y)} \tag{5}$$

$$= \frac{\int\int p(y|\beta,\sigma)\pi_\sigma(\sigma)\pi_\beta(\beta|\sigma,\xi,k)\pi_\xi(\xi|k)\pi_k(k)\mathrm{d}\sigma\mathrm{d}\beta}{\sum_k\int\int\int p(k,\xi,\beta,\sigma,y)\mathrm{d}\xi\mathrm{d}\beta\mathrm{d}\sigma} \tag{6}$$

. . . which is analytically intractable

MCMC sampling

- Bayesian inference gives posteriors of form

$$\text{posterior} = \frac{\text{likelihood} \times \text{prior}}{\text{Normalising constant}}$$

- The normalising constant is regularly analytically intractable
- Markov-chain Monte carlo methods allow us to sample from the posterior distribution anyway

MCMC

MCMC sets up a Markov chain whose stationary distribution is equal to the posterior distribution:

- Generate a random state from a proposal distribution

MCMC

MCMC sets up a Markov chain whose stationary distribution is equal to the posterior distribution:

- Generate a random state from a proposal distribution
- Accept it with some probability

MCMC

MCMC sets up a Markov chain whose stationary distribution is equal to the posterior distribution:

- Generate a random state from a proposal distribution
- Accept it with some probability
- Reject it with some probability

MCMC

MCMC sets up a Markov chain whose stationary distribution is equal to the posterior distribution:

- Generate a random state from a proposal distribution
- Accept it with some probability
- Reject it with some probability
- On acceptance, change the current state to the accepted state; else, remain at current state

MCMC

MCMC sets up a Markov chain whose stationary distribution is equal to the posterior distribution:

- Generate a random state from a proposal distribution
- Accept it with some probability
- Reject it with some probability
- On acceptance, change the current state to the accepted state; else, remain at current state
- Acceptance and rejection probabilities are chosen such that the distribution of accepted states matches that of the prior

MCMC

MCMC sets up a Markov chain whose stationary distribution is equal to the posterior distribution:

- Generate a random state from a proposal distribution
- Accept it with some probability
- Reject it with some probability
- On acceptance, change the current state to the accepted state; else, remain at current state
- Acceptance and rejection probabilities are chosen such that the distribution of accepted states matches that of the prior
- Doesn't require us to calculate the normalising constant!

Reversible-jump MCMC

- States are the model configuration (k, ξ)
- These are of many different dimensions
- To sample from a posterior with varying dimension, we use reversible-jump MCMC
  - Jump up and down in dimension, probabilistically
  - Do so in such a way that the posterior is accurate both within and across dimensions

Model inference

- Using RJMCMC, we can sample from the posterior $p(k, \xi | y)$, even though the dimensionality of ξ is not fixed
- We can use samples $k, \xi | y$ to condition on new data (x^*, y^*)
  - $p(y^* | x^*) = p(y^* | k, \xi, x) p(k, \xi | y)$

- We predict new points without ever actually setting up a splines model
  - Find a probability distribution over candidate splines models
  - Weight each spline model's output according to its probability

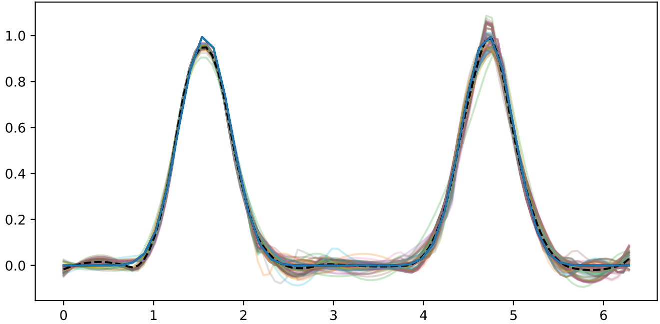
My results

- Three different MCMC actions can be taken
  - Add a new knot
  - Relocate a knot
  - Delete a knot
- Each action has a proposal probability (how likely are we to take this action?)
- Each step has an acceptance probability (how likely are we to accept this action?)
- The BARS paper does a rather bad job of explaining these!

In my implementation, probabilities are sometimes coming back negative, making it crash

Results

- Results can't be trusted!



BARS and GPR

- BARS maintains a distribution over splines
- GPR maintains a distribution over arbitrarily many functions
- Both methods refine the distribution with Bayesian methods

- BARS probabilistically finds the most informative knot point configuration
  - Finds set of spline-points that tell us the most about the data
  - Sparse GPR probabilistically finds the most informative inducing points distribution
- Tenuous link to optimal experiment design?

Next steps

1. Redraft paper
2. Get BARS to work
   - Useful as it's the most promising method for a conference abstract
   - Either get my implementation working, or adapt C code to my needs
3. Fix MSPEs
   - *Should* be quick and easy
4. (Re)validate all the models I'm playing with
5. Put results into a conference abstract