Basis Expansion Monte Carlo

Eric Kernfeld [*]

University of Washington, Seattle, WA, USA

March 18, 2016

Abstract

We introduce Basis Expansion Monte Carlo, which studies a Gibbs or Metropolis-Hastings sampler to infer the underlying transition kernel. To make inference about the target distribution, we compute the target produced by the approximate kernel. We reduce this to a fairly low-dimensional eigenvector calculation. The approach allows us to extract information wasted by common MCMC methods. Results show ...

1 Introduction

In Bayesian modeling, it is impossible to find a closed form for the posteror distribution of interest π. One work-around, originating in computational physics, is the Metropolis-Hastings algorithm. It relies on the fact that for points x_1 and x_2 in the parameter space, $\pi(x_1)/\pi(x_2)$ may still be calculable, though $\pi(x_1)$ and $\pi(x_2)$ are not. This fact is exploited to produce a Markov chain whose steady-state distribution is guaranteed to be π.

More and references about history, background, and/or introductions to monte carlo methods

The Metropolis-Hastings scheme consists of the following procedure.

Algorithm 1: Metropolis-Hastings algorithm

```
Set $x_0 = 0, i = 0$
Repeat ad nauseum:
  Increment $i$
  Draw $x$ from a proposal distribution $q(x|x_{i-1})$
  Set $\alpha(x|x_{i-1}) = \min(1, \frac{\pi(x)q(x_{i-1}|x)}{\pi(x_{i-1})q(x|x_{i-1})})$
  Set $x_i = x$ with probability $\alpha$ and $x_i = x_{i-1}$ with probability $1 - \alpha$.
```

Suppose this MCMC algorithm produces a chain $x_1, x_2, x_3, ...$ of samples. Because the algorithm is stochastic, these samples can be viewed as realizations of random variables $X_1, X_2, X_3, ...$ with marginal density functions f_1, f_2, f_3, etc. Because X_i is independent of past draws given X_{i-1}, the conditional distribution $f_{i|i-1}(x_i|x_{i-1})$ contains all the information we need about this method. In particular, to move forward an iteration, we can write $f_i(x_i) = \int f_{i|i-1}(x_i, x_{i-1})f_{i-1}(x_{i-1})dx_{i-1}$. Noting that $f_{i|i-1}$ doesn't depend on i, we can replace it with a function K so that $f_i(x_i) = \int K(x_i, x_{i-1})f_{i-1}(x_{i-1})dx_{i-1}$. This function K, called the Markov kernel, is analogous to the transition probability matrices of discrete-space Markov chain theory. We refer to the linear operator of integrating against K as L, so that $f_i = Lf_{i-1}$. The object of interest is the steady state of this operator, an eigenfunction π that has eigenvalue 1 so that for any x, $\pi(x) = (L\pi)(x) = \int K(x, t)\pi(t)dt$.

In MCMC methods, chains are usually left to run until the Markov chain converges to its stationary distribution. By contrast, in BEMC, we approximate L by methods that allow the chain to run briefly from many different places, and we compute π from the approximation.

[*]Electronic address: ekernf01@u.washington.edu; Corresponding author

1.1 Stage one: approximating the kernel

1.1.1 Notation and setup

In crafting this method, we will have to cope with two sources of error. One is discretization error: we must find a finite representation for the Markov kernel. The other is natural stochasticity from the sampler. Objects suffering only from discretization error will be labeled with tildes, while objects also suffering from stochasticity will be labeled with hats.

Our estimator begins with a fixed set of basis functions $\{h_i\}_1^B$. These might be multivariate normal density functions. We then define some relevant operators and kernels, plus the matrices of coefficients for when we expand them in the basis of h's. For an unknown transition kernel K, define $\tilde{M}$ to be the matrix that minimizes the squared $\mathcal{L}_2$ distance $\|K - \sum_{i,j}^B \tilde{m}_{ij} h_i \otimes h_j\|^2$, or in other terms the matrix that minimizes the integral

$$\int_{\Omega\times\Omega} (K(x,y) - \sum_{i,j}^B \tilde{m}_{ij} h_i(x) h_j(y))^2 dxdy.$$

EMK: Surely this minimum must exist for a quadratic? Then define $\tilde{K}(x,y) \equiv \sum_{i,j=1}^B \tilde{m}_{ij} h_i(x) h_j(y)$ and define $\tilde{L}$ so that $(\tilde{L}f)(x)$ is

$$\int_\Omega \tilde{K}(x,y) f(y) dy.$$

Later, we will make a noisy estimate of $\tilde{M}$, calling it $\hat{M}$. The corresponding estimate for K will be

$$\hat{K}(x,y) = \sum_{i,j=1}^B \hat{M}_{ij} h_i(x) h_j(y). \tag{1}$$

Similar to before, $\hat{L}$ is defined so that $(\hat{L}f)(x)$ is

$$\int_\Omega \hat{K}(x,y) f(y) dy.$$

Finally, the next few definitions will help describe the interaction of the h's with one another. We define a matrix $\tilde{G}$ elementwise so that $\tilde{g}_{ij} \equiv \int_\Omega h_i(x)(\tilde{L}h_j)(x)dx$, with the corresponding statement for hatted variables so that $\hat{g}_{ij} \equiv \int_\Omega h_i(x)(\hat{L}h_j)(x)dx$. We also make use of the $\mathcal{L}_2$ inner products $c_{ij} \equiv \int_\Omega h_i(x)h_j(x)dx$.

1.2 The BEMC estimator

Since $\hat{g}_{ij}$ expands as

$$\int_\Omega h_i(x) \left[\sum_{\ell,k} M_{k\ell} h_k(x) \int_\Omega h_\ell(y) h_j(y) dy \right] dx = \sum_{\ell,k} M_{k\ell} C_{\ell j} C_{ik},$$

we know that $\hat{G} = C\hat{M}C$. One nice special case of this formula: for some choices of $\{h_i\}_1^B$, C is the identity matrix and $\hat{G} = \hat{M}$. We assume C is readily calculable and not too badly conditioned, so that if we can estimate G, we can estimate M as $C^{-1}GC^{-1}$. We now focus on G, which we approximate via importance sampling.

If h_j is a valid density function, then by definition, G can be written as an expectation $G_{ij} = E_{Lh_j}[h_i]$. Indeed, if h_j is a valid density function, then Lh_j is a probability distribution, too. What generative process does this distribution describe? It corresponds to initializing the sample from a draw $z \sim h_j$, then running a single step of Metropolis-Hastings. Consequently, even though L is unknown, sampling from Lh_j is straightforward. This motivates us to sample from a normalized version of Lh_j and approximate G_{ij} as an average. All we need to do is sample z from h_j, run an M-H iteration on z to get w, and retain w as our sample from Lh_j. EMK: Conjecture: the hats converge to the tildes as you run it for longer, and the tildes converge to the truth as you lengthen the basis.

1.2.1 Basic Limitations

Our approximation can only imitate continuous kernels, i.e. situations where $\int K(x,y)f(y)dy$ can be done with respect to the Lebesgue measure. This might be reasonable if K represented a complete cycle through a Gibbs sampler. For the Metropolis-Hastings algorithm, this presents an obstacle, because with positive probability, it will reject a proposed sample and stay in place. As a workaround, we can approximate the kernel not of a single M-H iteration but of r iterations for r around 10 or 20. The probability of r consecutive rejections is much smaller, pushing the true kernel closer to the subspace in which we approximate it. In section 5, we discuss a variant that explicitly models rejection events.

1.2.2 A more concrete procedure using a Gaussian basis

For something more tangible, we will consider $\Omega = \mathbb{R}^D$, using multivariate Gaussian densities as our basis so that $h_i(x) = (\frac{1}{(2\pi)^{\frac{D}{2}}|\Sigma_i|^{1/2}})\exp(\frac{-(x-\mu_i)^T\Sigma_i^{-1}(x-\mu_i)}{2})$. This makes it easy to draw starting points for the sampler. Also, C_{ij} can be computed easily using facts about the Normal distribution. Defining $\Lambda \equiv \Sigma_i^{-1}+\Sigma_j^{-1}$ and $\nu = \Lambda^{-1}(\Sigma_i^{-1}\mu_i + \Sigma_j^{-1}\mu_j)$, we have

EMK: Definitely check this again eventually

$$\begin{aligned}
\int h_i(x)h_j(x)dx &= \int \frac{1}{(2\pi)^{\frac{D}{2}}|\Sigma_i|^{1/2}}\exp(\frac{-(x-\mu_i)^T\Sigma_i^{-1}(x-\mu_i)}{2})\frac{1}{(2\pi)^{\frac{D}{2}}|\Sigma_j|^{1/2}}\exp(\frac{-(x-\mu_j)^T\Sigma_j^{-1}(x-\mu_j)}{2}dx \\
&= \frac{1}{(2\pi)^D\Sigma_i|^{1/2}|\Sigma_j|^{1/2}}\int \exp(\frac{-(x-\mu_i)^T\Sigma_i^{-1}(x-\mu_i)}{2} - \frac{(x-\mu_j)^T\Sigma_j^{-1}(x-\mu_j)}{2}dx \\
&= \frac{1}{(2\pi)^D\Sigma_i|^{1/2}|\Sigma_j|^{1/2}}\int \exp\frac{-1}{2}\{(x-\mu_i)^T\Sigma_i^{-1}(x-\mu_i) + (x-\mu_j)^T\Sigma_j^{-1}(x-\mu_j)\}dx \\
&= \frac{1}{(2\pi)^D\Sigma_i|^{1/2}|\Sigma_j|^{1/2}}\int \exp\frac{-1}{2}\{x^T(\Sigma_i^{-1}+\Sigma_j^{-1})x - 2x^T(\Sigma_i^{-1}\mu_i+\Sigma_j^{-1}\mu_j) + \mu_i^T\Sigma_i^{-1}\mu_i + \mu_j^T\Sigma_j^{-1}\mu_j\}dx \\
&= \frac{\exp(\frac{-1}{2}\{\mu_i^T\Sigma_i^{-1}\mu_i + \mu_j^T\Sigma_j^{-1}\mu_j\})}{(2\pi)^D\Sigma_i|^{1/2}|\Sigma_j|^{1/2}}\int \exp\frac{-1}{2}\{x^T\Lambda x - 2x^T(\Sigma_i^{-1}\mu_i+\Sigma_j^{-1}\mu_j)\}dx \\
&= \frac{\exp(\frac{-1}{2}\{\mu_i^T\Sigma_i^{-1}\mu_i + \mu_j^T\Sigma_j^{-1}\mu_j\})}{(2\pi)^D\Sigma_i|^{1/2}|\Sigma_j|^{1/2}}\int \exp\frac{-1}{2}\{x^T\Lambda x - 2x^T\Lambda\nu\}dx \\
&= \frac{\exp(\frac{-1}{2}\{\mu_i^T\Sigma_i^{-1}\mu_i + \mu_j^T\Sigma_j^{-1}\mu_j\})}{(2\pi)^D\Sigma_i|^{1/2}|\Sigma_j|^{1/2}}\int \exp\frac{-1}{2}\{x^T\Lambda x - 2x^T\Lambda\nu + \nu\Lambda\nu - \nu\Lambda\nu\}dx \\
&= \frac{\exp(\frac{-1}{2}\{\mu_i^T\Sigma_i^{-1}\mu_i + \mu_j^T\Sigma_j^{-1}\mu_j - \nu\Lambda\nu\})}{(2\pi)^D\Sigma_i|^{1/2}|\Sigma_j|^{1/2}}\int \exp\frac{-1}{2}\{x^T\Lambda x - 2x^T\Lambda\nu + \nu\Lambda\nu\}dx \\
&= \frac{\exp(\frac{-1}{2}\{\mu_i^T\Sigma_i^{-1}\mu_i + \mu_j^T\Sigma_j^{-1}\mu_j - \nu\Lambda\nu\})}{(2\pi)^D|\Sigma_i|^{1/2}|\Sigma_j|^{1/2}}\frac{(2\pi)^{\frac{D}{2}}}{|\Lambda|^{\frac{1}{2}}} \\
&= \frac{\exp(\frac{-1}{2}\{\mu_i^T\Sigma_i^{-1}\mu_i + \mu_j^T\Sigma_j^{-1}\mu_j - \nu\Lambda\nu\})}{(2\pi)^{D/2}|\Sigma_i|^{1/2}|\Sigma_j|^{1/2}}\frac{1}{|\Lambda|^{\frac{1}{2}}}
\end{aligned}$$

2 Stage two: calculating the target

We now explain how to get, from our approximation for the kernel, an approximation for the target. We make use of the power method, a simple algorithm for eigenvector computation. Given a matrix M, the power method computes $v \leftarrow Mv$ for any initial vector v, iterating until convergence. For "nice" matrices, this converges rapidly to an eigenvector; for "nicer" ones, the result is always the unique dominant eigenvector. MCMC schemes essentially rely on the same idea: for any initial distribution f, and for a Markov kernel K,

Algorithm 2: BEMC algorithm–stage one

```
Set $M$ to a matrix of all zeroes.
For $i = 1 : B$
   For $j = 1 : B$
      For $n = 1 : N$
         Draw a sample $z_n$ from $h_j$, i.e. a normal draw with mean $\mu_j$ and variance $\sigma_j^2$.
         Run the M-H sampler for $\ell$ rounds on $z_n$. Call the result $w_n$.
         Increment $G_{ij}$ by $h_i(w_n)$.
Divide $G$ by $N$.
```

turn it into a sample from $Lf = \int K(\cdot, y)f(y)dy$, and iterate; samples from $L^P f$ for some large integer P are good enough because $L^P f$ converges to π just as Mv converges to an eigenvector.

So, suppose we want a distribution $\hat{\pi}$ with the property $\hat{\pi} = \hat{L}\hat{\pi}$. From the form of $\hat{L}$, we know that $\hat{\pi} \in \text{span}\{h_i | i \in 1...B\}$, so we need only find the vector of coefficients, which we call v. As it turns out, v must be an eigenvector of MC: if

$$\hat{\pi}(x) = \sum_{k=1}^{B} v_k h_k(x),$$

then

$$\begin{aligned}
\sum_{k=1}^{B} v_k h_k(x) &= \hat{\pi}(x) \\
&= (\hat{L}\hat{\pi})(x) \\
&= \int \sum_{i,j=1}^{B} M_{ij} h_i(x) h_j(y) \sum_{k=1}^{B} v_k h_k(y) dy \\
&= \sum_{i,j,k=1}^{B} M_{ij} h_i(x) v_k \int h_j(y) h_k(y) dy \\
&= \sum_{i,j,k=1}^{B} M_{ij} h_i(x) v_k c_{jk} \\
&= \sum_{i,j=1}^{B} M_{ij} h_i(x) (Cv)_j \\
&= \sum_{i=1}^{B} h_i(x) (MCv)_i.
\end{aligned}$$

The fact $v_i = (MCv)_i$ follows from linear independence of the basis elements.

Algorithm 3: BEMC algorithm–stage two

```
Given an estimate $\hat{G}$ of $G$:
For each $i, j$ pair, compute $C_{ij}$ as $(\frac{1}{2\pi(\sigma_i+\sigma_j)^2})^{\frac{D}{2}} \exp(\frac{-(\mu_i-\mu_j)^2}{2\sigma_i^2+2\sigma_j^2})$.
Compute $MC = C^{-1}\hat{G}$.
Compute the leading eigenvector $v$ of $MC$.
Return $\sum_i v_i h_i$ as a posterior estimate
```

3 Adding basis functions

Especially in a high-dimensional parameter space, it is unreasonable to expect basis functions to be specified a priori: there must be some way to introduce new basis functions as the sampler runs. So we AAAARGHHH DON'T KNOW YET

4 Running BEMC in O(B) time

Estimating every single element of $\hat{G}$ is wasteful. Some basis functions are far from one another, and their values of $\langle h_i, Lh_j \rangle$ will be near zero. To take advantage of this, we maintain a list for each basis function of the five nearest neighbors, and we only estimate $\langle h_i, Lh_j \rangle$ for neighbors. (We also fill in additional entries so that the "neighbors" relation is symmetric.) This leaves us with an incomplete matrix $\hat{G}$.

Fortunately, we do not need the full matrix $C^{-1}\hat{G}$, only its leading eigenvector. To deal with the missing entries, we can cast this as an optimization problem and omit unknown terms of the penalty. In the quadratic-time scheme, our coefficients play the role of v in

$$\begin{aligned}
\text{argmin}_{v,w}||vw^T - C^{-1}\hat{G}||_2^2 &= \text{argmin}_{v,w}\text{Tr}((vw^T - C^{-1}\hat{G})(wv^T - \hat{G}^T C^{-1T})) && (2)\\
&= \text{argmin}_{v,w}\text{Tr}(vw^T wv^T) - 2\text{Tr}(C^{-1}\hat{G}wv^T) + \text{Tr}(C^{-1}\hat{G}\hat{G}^T C^{-1T}) && (3)\\
&= \text{argmin}_{v,w} v^T v w^T w - 2\text{Tr}(\hat{G}wv^T C^{-1}) && (4)\\
& && (5)
\end{aligned}$$

A simple extension for missing entries is to instead solve

$$\text{argmin}_{v,w} v^T v w^T w - 2\text{Tr}(Z\hat{G}Zwv^T C^{-1}),$$

where Z_{ij} is 0 unless h_i and h_j are neighbors. This is equivalent to finding the leading eigenvector of $C^{-1}Z\hat{G}Z$

5 BEMC-R, a variant modeling rejections

As we mention in section 1.1, our scheme is able to model continuous kernels. On the other hand, the Metropolis-Hastings algorithm sometimes rejects proposed samples, and its kernel will assign positive mass to intervals on the "diagonal" set $\{(x, y) \in \Omega^2 | x = y\}$. In this section, we introduce a variant of BEMC that explicitly models rejections by the sampler.

Let us look at the Metropolis-Hastings kernel in more detail. Going back to the algorithm, the quantity $\alpha(x|y)$ is the probability of accepting a move from y to x. This is a function of two variables, but for convenience, let us introduce a univariate function

$$\alpha(y) = P(\text{accept next move} \mid \text{currently at } y).$$

Splitting up the next draw as an alternative between moving and staying put, we can write $K(x, y) = (1 - \alpha(y))\delta_y(x) + \alpha(y)f_{acc}(x|y)$. In this expression, $f_{acc}(x|y)$ is the conditional density of x given that our move out of y was not rejected, while $\delta_y(\cdot)$ is a point mass at y. To set up the last line below, define the operator D_{rej} from α so that $(D_{rej}f)(x) \equiv (1 - \alpha(x))f(x)$, define $K_{acc}(x, y)$ as $f_{acc}(x|y)\alpha(y)$, and let $(L_{acc}f)(x) \equiv \int K_{acc}(x, y)f(y)dy$. The idea behind this notation is that D_{rej} stands for a diagonal operator that describes rejection, and K_{acc} is like the Markov kernel, but it describes movement conditioning on acceptance.

If x comes right after y in the sampler, we can relate their PDF's with these operators.

$$\begin{aligned}
f_x(x) &= \int K(x,y) f_y(y) dy \\
&= \int ((1-\alpha(y))\delta_y(x) + \alpha(y) f_{acc}(x|y)) f_y(y) dy \\
&= \int (1-\alpha(y))\delta_x(y) f_y(y) dy + \int K_{acc}(x,y) f_y(y) dy \\
&= (1-\alpha(x)) f_y(x) + \int K_{acc}(x,y) f_y(y) dy \\
&= (D_{rej} f_y)(x) + (L_{acc} f_y)(x)
\end{aligned}$$

We can sample from a pdf proportional to $D_{rej} f$ by sampling y from $f_y()$, then running an M-H iteration on y to get x and retaining the sample y if $x = y$. We can sample from a pdf proportional to $L_{acc} f$ by doing nearly the same steps, but retaining x if $x \neq y$. These facts will be useful as we attempt to estimate L_{acc}.

To define another set of "tilde" objects, let $\tilde{M}_{acc}$ be the matrix that minimizes the squared $\mathcal{L}_2$ distance $\|K_{acc} - \sum_{i,j}^{B} \tilde{m}_{ij} h_i \otimes h_j\|^2$, or in other terms the matrix that minimizes the integral

$$\int_{\Omega \times \Omega} (K_{acc}(x,y) - \sum_{i,j}^{B} \tilde{m}_{ij} h_i(x) h_j(y))^2 dx dy.$$

We will use a separate set of functions $\{\phi_i\}_{i=1}^{B_\phi}$ to approximate α. Let $\tilde{r}$ be the vector that minimizes the integral $\int_\Omega (\alpha(x) - \sum_{i=1}^{B_\phi} \tilde{r}_i \phi_j(y))^2 dx$ and let $\tilde{r}$ be $\sum_{i=1}^{B_\phi} \tilde{r}_i \phi_j(x)$

This time around, we will try to estimate a function $\hat{\alpha}$ and a matrix M so that $\hat{\alpha} \approx \tilde{\alpha}$ and $\hat{L}_{acc} \approx \tilde{L}_{acc}$.So, we need still need to estimate M, but with the added complication of trying to infer $\hat{\alpha}$ at the same time. Fortunately, it is easy to tell when the sampler rejects and when it doesn't, and this provides a way to tease out information about α. Suppose for a moment that we start the sampler at a point y and it takes a single step to a value we call x. If $x \neq y$, then the sampler has shown less of a tendency to reject starting from y, and we label y with a 1. If $x = y$, we label y with a 0. Once Ω is covered in zeroes and ones, there are many probabilistic classifier methods that could give an estimate of $\hat{\alpha}$, which at any given point is just the probability of labeling with a one. Meanwhile, whenever the sampler moves, we gain information about L_{acc}, and we can update M as before.

This strategy still throws away useful information. To see why, recall that the Metropolis-Hastings algorithm makes a proposal, computes an rejection probability, and flips a metaphorical coin with that probability. Then, having rejected or accepted, it discards the rejection probability. When drawing a chain of samples, the rejection probability serves no further purpose, so discarding it is natural. In BEMC-R, though, we can keep it to provide a more efficient estimate of α. If the rejection probability when proposing a move to y from x is p, then the better procedure is to label y with p. Likewise, instead of updating the estimate of M_{ij} using a sample of weight 1 with probability p, we can update it using a sample of weight p. We summarize the procedure in Algorithm 4.

For one further refinement, we could include some prior information about M. Since L_{acc} mimics the action of the sampler as it moves, it might resemble the action of the proposal alone, with no rejections. That would mean $\hat{g}_{ij} \approx \int h_i(x) q(x|y) h_j(y) dy dx$. For simple proposal distributions like a uniform or normal centered on the current value, this integral may be easy to find as a convolution.

5.1 Computing the steady state in BEMC-R

In BEMC as presented in section 1.1, there was nothing to gain by representing our estimate $\hat{\pi}$ of the target outside of the span of $\{h_i\}_{i=1}^B$: any component orthogonal to span$\{h_i\}_{i=1}^B$ would get zeroed out upon a single application of $\hat{L}$. This is no longer the case; because of the diagonal term D_{rej}, we have no guarantee that our approximation stays within any particular finite-dimensional subspace. We still need to

Algorithm 4: BEMC-R algorithm–stage one

```
Set $\hat{G}$ to 0.
Set a scalar $W$ to zero. $W$ is the effective number of samples in an estimate of an entry of $\hat{G}$.
Set $T = \{\}$. $T$ will be the training set for $\hat{\alpha}$.
For $i = 1 : B$
  For $j = 1 : B$
    For $n = 1 : N$
      Draw a sample $y_n$ from $h_j$.
      From an MCMC sampler with the correct target, draw a proposal $x_n|y_n$ and compute its
      rejection probability $p$.
      Add $(y_n, p)$ to $T$.
      Increment $\hat{G}_{ij}$ by $ph_i(x_n)$.
      Increment $W_{ij}$ by $p$.
    Divide $\hat{G}_{ij}$ by $W_{ij}$.
Train on $T$ to get $\hat{\alpha}$.
```

represent $\hat{\pi}$ in computer memory, so for now, we will consider only approximations that we can write as $\hat{\pi}(x) = \sum_{i=1}^{B} v_i h_i(x)$. Our final approximation to the transition kernel will be

$$(P_{\text{span}\{h_i\}}\hat{D}_{rej} + \hat{L}_{acc}),$$

where $P_{\text{span}\{h_i\}}$ is the orthogonal projector onto the set of functions expressible as $\sum_{i=1}^{B} v_i h_i(x)$.

To simplify notation, let $\psi_{\tau(\ell,i)} \equiv \phi_\ell h_i$ for some bijective $\tau()$. In this paragraph, k or ℓ are outputs from τ while i and j are inputs or plain indices. We drop this convention later when three indices of one "type" are needed. We define matrices D so that $d_{jk} = \int \psi_k(x)h_j(x)dx$ and E so that $e_{k,\ell} = \int \psi_k(x)\psi_\ell(x)dx$. So, D is rectangular, and E is square and bigger than C. The product CDE makes sense. The projector $P_{\text{span}\{h_i\}}$ will take the form of a matrix Q, and Q will have the same size as D. In order to help Q carry out a single application of $P_{\text{span}\{h_i\}}$, we imagine coefficients v_k taken from a function of the form $\sum_k v_k\psi_k(x)$. Because of the properties of orthogonal projectors, performing Qv should produce coefficients w_i that minimize the $\mathcal{L}_2$ residual $\int(\sum_k v_k\psi_k(x) - \sum_i w_i h_i(x))^2$. This gives us a tool to deduce Q. Distributing the sum and moving the integral inside, the residual becomes

$$\begin{aligned}\int(\sum_k v_k\psi_k(x) - \sum_i w_i h_i(x))^2 &= \int \sum_k v_k\psi_k(x)\sum_\ell v_\ell\psi_\ell(x) - 2\sum_k v_k\psi_k(x)\sum_i w_i h_i(x) + \sum_i w_i h_i(x)\sum_j w_j h_j(x)\\ &= \sum_{k,\ell} v_k v_\ell e_{kl} - 2\sum_{k,i} w_i v_k d_{ik} + \sum_i\sum_j w_i w_j c_{ij}\\ &= v^T E v - 2v^T D w + w^T C w.\end{aligned}$$

Setting its gradient to zero yields $Q = C^{-1}D^T$ because

$$\begin{aligned}&0 = -2v^T D + 2w^T C\\ \implies &D^T v = C^T w\\ \implies &C^{-1}D^T v = w.\end{aligned}$$

EMK: also considered $\hat{\pi}(x) = \sum_{i,k=1}^{B,B_\phi} v_{ik}\phi_k(x)h_i(x)$.This would require "full-house" integrals with three ϕ terms and two h terms. Since we will choose ϕ_1 to be constant, this form is at least as expressive as its rejection-neglecting predecessor. Could also choose yet another basis for this, but it would have to play nice with the ϕ's and h's anyway.

Given M and $\hat{r}$, we follow the same tactic as in section 2.

$$
\begin{aligned}
\sum_{i}^{B} w_i h_i(x) &= \hat{\pi}(x) \\
&= ((P_{\text{span}\{h_i\}} \hat{D}_{rej} + \hat{L}_{acc})\hat{\pi})(x) \\
&= P_{\text{span}\{h_i\}} \left\{ \sum_{i=1}^{B_\phi} \phi_i(x) \sum_{\ell}^{B} w_\ell h_\ell(x) \right\} + \int \sum_{i,j=1}^{B} M_{ij} h_i(x) h_j(y) \sum_{\ell=1}^{B} w_\ell h_\ell(y) dy \\
&= P_{\text{span}\{h_i\}} \left\{ \sum_{i=1}^{B_\phi} \phi_i(x) \sum_{\ell}^{B} w_\ell h_\ell(x) \right\} + \sum_{i,j,\ell=1}^{B} M_{ij} h_i(x) c_{j\ell} w_\ell \\
&= P_{\text{span}\{h_i\}} \left\{ \sum_{i=1}^{B_\phi} \sum_{j}^{B} w_j \psi_{\tau(i,j)} \right\} + \sum_{i=1}^{B} h_i(x)(MCw)_i
\end{aligned}
$$

Since we now know we can apply $P_{\text{span}\{h_i\}}$ by calculating $C^{-1}D^T v$, we can express the entire process as a matrix multiplication with just one more definition. Let A have $A_{kj} = 1$ if for some ℓ, $k = \tau(\ell, j)$ and $A_{kj} = 0$ otherwise. Then by our definition, the coefficients for $\hat{\pi}$ in terms of $\{h_i\}$ come from the eigenvector $w = (C^{-1}D^T A + MC)w$.

5.2 Estimator properties

Prove $E[\hat{G}] = \tilde{G}$?

Our estimator w is an eigenvector; thus, it satisfies the estimating equation $(X - I)v = 0$ for some X. Its asymptotic variance is

$$(X - I)^{-1}\text{Var}[(X - I)v](X - I)^{-1T}$$

5.3 A concrete choice of basis for BEMC-R

We can implement this rejection-tolerant version on $\mathbb{R}^D$ using a list of Gaussians for $\{h_i\}_{i=1}^{B}$ like we do in section 1.2.2. In selecting $\{\phi_i\}_{i=1}^{B_\phi}$, we want to make it easy to compute inner products of the form $\int \phi_k(x)h_i(x)dx$. We also want something appropriate to express α as it occurs naturally. These requirement suggest using constants and Gaussians, but on top of that, we suggest one additional change. If Φ is the cumulative distribution function of the standard normal, and $f_D(\cdot|\mu, \Sigma)$ is a normal density on R^D, then for any pair of our basis functions, $\int \Phi(\theta^T x)h_i(x)h_j(x)dx$ will still be tractable. This is worth the added complexity because probits, regularly used in situations with binary response and continuous predictors, will capture α more easily.

The necessary calculation follows. First, simplify $h_i(x)h_j(x)$ into a single normal density. Below, the basis functions i and j have precision matrices Λ_i and Λ_j, with $\Lambda_{ij} \equiv \Lambda_i + \Lambda_j$ and $\mu_{ij} \equiv \Lambda_{ij}^{-1}(\Lambda_i\mu_i + \Lambda_j\mu_j)$.

$$\begin{aligned}
h_i(x)h_j(x) &= \frac{1}{(2\pi)^{n/2}|\Sigma_i|}\frac{1}{(2\pi)^{n/2}|\Sigma_j|}\exp(\frac{-(x-\mu_i)^T\Sigma_i^{-1}(x-\mu_i)}{2})\exp(\frac{-(x-\mu_j)^T\Sigma_j^{-1}(x-\mu_j)}{2}) \\
&= \frac{1}{(2\pi)^{n/2}|\Sigma_i|}\frac{1}{(2\pi)^{n/2}|\Sigma_j|}\exp(\frac{-(x-\mu_i)^T\Lambda_i(x-\mu_i)}{2} - \frac{(x-\mu_j)^T\Lambda_j(x-\mu_j)}{2}) \\
&= \frac{1}{(2\pi)^{n/2}|\Sigma_i|}\frac{1}{(2\pi)^{n/2}|\Sigma_j|}\exp(\frac{-x^T\Lambda_i x + 2x^T\Lambda_i\mu_i - \mu_i^T\Lambda_i\mu_i}{2} - \frac{-x^T\Lambda_j x + 2x^T\Lambda_j\mu_j - \mu_j^T\Lambda_j\mu_j}{2}) \\
&= \frac{1}{(2\pi)^{n/2}|\Sigma_i|}\frac{1}{(2\pi)^{n/2}|\Sigma_j|}\exp(\frac{-x^T(\Lambda_i+\Lambda_j)x + 2x^T(\Lambda_i\mu_i+\Lambda_j\mu_j) - \mu_i^T\Lambda_i\mu_i - \mu_j^T\Lambda_j\mu_j}{2}) \\
&= \frac{(2\pi)^{n/2}|\Sigma_{ij}|}{(2\pi)^{n/2}|\Sigma_i|(2\pi)^{n/2}|\Sigma_j|}\frac{1}{(2\pi)^{n/2}|\Sigma_{ij}|}\exp(\frac{-x^T(\Lambda_{ij})x + 2x^T\Lambda_{ij}\mu_{ij} - \mu_{ij}^T\Lambda_{ij}\mu_{ij} + \mu_{ij}^T\Lambda_{ij}\mu_{ij} - \mu_i^T\Lambda_i\mu_i - \mu_j^T\Lambda_j\mu_j}{2}) \\
&= \frac{(2\pi)^{n/2}|\Sigma_{ij}|}{(2\pi)^{n/2}|\Sigma_i|(2\pi)^{n/2}|\Sigma_j|}f_D(x|\mu_{ij},\Sigma_{ij})\exp(\frac{\mu_{ij}^T\Lambda_{ij}\mu_{ij} - \mu_i^T\Lambda_i\mu_i - \mu_j^T\Lambda_j\mu_j}{2}) \\
&\equiv \kappa_{ij} f_D(x|\mu_{ij},\Sigma_{ij})
\end{aligned}$$

Then, defining a constant

$$\kappa_{ij} \equiv \frac{(2\pi)^{n/2}|\Sigma_{ij}|}{(2\pi)^{n/2}|\Sigma_i|(2\pi)^{n/2}|\Sigma_j|}\exp(\frac{\mu_{ij}^T\Lambda_{ij}\mu_{ij} - \mu_i^T\Lambda_i\mu_i - \mu_j^T\Lambda_j\mu_j}{2}),$$

the integral follows. Define U_θ to be an orthogonal matrix so that $\theta^T U_\theta$ is $(\|\theta\|, 0, 0, ..., 0)$, i.e. so that the first column is a multiple of θ, and define $u \equiv U_\theta^T x$.

$$\begin{aligned}
\int \Phi(\theta^T x)h_i(x)h_j(x)dx &= \kappa_{ij}\int \Phi(\theta^T x)f_D(x|\mu_{ij},\Sigma_i)dx \\
&= \kappa_{ij}\int \Phi(\theta^T U_\theta u)f_D(U_\theta u|\mu_{ij},\Sigma_i)du \\
&= \kappa_{ij}\int \Phi(\|\theta\|u_1)f_D(u|U_\theta^T\mu_{ij}, U_\theta^T\Sigma_i U_\theta)du \\
&= \kappa_{ij}\int \Phi(\|\theta\|u_1)f_1(u_1|(U_\theta^T\mu_{ij})_1, (U_\theta^T\Sigma_i U_\theta)_{11})du_1 \\
&= \kappa_{ij}\int \Phi(\|\theta\|u_1)f_1(u_1|\frac{\theta^T\mu_{ij}}{\|\theta\|}, \frac{\theta^T\Sigma_i\theta}{\|\theta\|^2})du_1 \\
&= \kappa_{ij}\int \Phi(z)f_1(z|\theta^T\mu_{ij}, \theta^T\Sigma_i\theta)\frac{1}{\|\theta\|}dz \\
&= \frac{\kappa_{ij}}{\|\theta\|}\int \Phi(z)f_1(\frac{z-\theta^T\mu_i}{\sqrt{\theta^T\Sigma_i\theta}}|0,1)dz \\
&= \frac{\kappa_{ij}\sqrt{\theta^T\Sigma_i\theta}}{\|\theta\|}\int \Phi(w\sqrt{\theta^T\Sigma_i\theta} + \theta^T\mu_{ij})f_1(w|0,1)dw \\
&= \frac{\kappa_{ij}\sqrt{\theta^T\Sigma_i\theta}}{\|\theta\|}\int \Phi(w\sqrt{\theta^T\Sigma_i\theta} + \frac{\theta^T\mu_{ij}}{\sqrt{\theta^T\Sigma_i\theta}}\sqrt{\theta^T\Sigma_i\theta})f_1(w|0,1)dw \\
&= \frac{\kappa_{ij}s}{\|\theta\|}\int \Phi(\frac{w+m}{s}))f_1(w|0,1)dw \text{ if } s \equiv \sqrt{\theta^T\Sigma_i\theta}^{-1} \text{ and } m \equiv \frac{\theta^T\mu_{ij}}{\sqrt{\theta^T\Sigma_i\theta}} \\
&= \frac{\kappa_{ij}s}{\|\theta\|}\Phi(\frac{m}{1+s^2})
\end{aligned}$$

This provides a formula for the matrix D from section 5.1.

References

[1] Weisstein, E.W.: Hermite polynomial. From MathWorld–A Wolfram Web Resource. (December 2014)

[2] Golub, G.H., Welsch, J.H.: Calculation of Gauss quadrature rules. Math. Comp. **23**(106) (1969) 221–230 Loose microfiche suppl. A1–A10.

[3] Blocker, A.W.: fastGHQuad: Fast Rcpp implementation of Gauss-Hermite quadrature. (2014) R package version 0.2.

[4] Bunck, B.: A fast algorithm for evaluation of normalized hermite functions. Bit Numer Math **49**(2) (2009) 281–295