LaplaceInterpolation.jl: A Julia package for fast interpolation on a grid

28 Jun, 2021

Summary

We introduce a linear-time algorithm for interpolation on a regular multidimensional grid, implemented in the Julia language. The algorithm is an approximate Laplace interpolation [@press1992] when no parameters are given, and when parameters $m \in \mathbb{Z}$ and $\epsilon > 0$ are set, the interpolant approximates a Mat'ern kernel, of which radial basis functions and polyharmonic splines are a special case. We implement, in addition, Neumann, Dirichlet (trivial) and average boundary conditions with potentially different aspect ratios in the different dimensions. The interpolant functions in arbitrary dimensions.

Mathematical Background

Radial basis functions and splines can be unified conceptually through the notion of Green's functions and eigenfunction expansions [@fasshauer2012green]. The general multivariate Mat'ern kernels [@stein1995] are of the form

```
\begin{equation}
K(\mathbf{x}; \mathbf{z}) = K_{m-d/2}(\epsilon||\mathbf{x}-\mathbf{z}||)(\epsilon||\mathbf{x
\end{equation}
```

where K_ν is the modified Bessel function of the second kind, and can be obtained as Green's kernels of (see [@fasshauer2011reproducing])

```
\begin{equation}
L = (\epsilon^2I-\Delta)^m, \quad m>\frac{d}{2}
\end{equation}
```

where Δ denotes the Laplacian operator in d dimensions. Polyharmonic splines, that is, radial basis functions of the form

```
\begin{equation}
K(\mathbf{x}; \mathbf{z}) = \left\{ \begin{array}{ll} ||\mathbf{x} - \mathbf{z} ||^{2m-d} &
||\mathbf{x} - \mathbf{z} ||^{2m-d} \log ||\mathbf{x} - \mathbf{z} || & d \mbox{
```

```
even}
\end{equation}
```

are a special case of the above, and this class includes the thin plate splines.

The discrete gridded interpolation [@press1992,@mainberger2011optimising] seeks to find an interpolation $u(\mathbf{x})$ that satisfies the differential operator in d dimensions on the nodes `$\mathbf{x}_i$` where there is no data and equals `$y_i$` everywhere else. Discretely, one solves the matrix problem

```
\begin{equation}
\mathbf{C} (\mathbf{u} - \mathbf{y}) - (1 - \mathbf{C}) L \mathbf{u} = 0
\end{equation}
```

where $\mathbf{y}$ contains the `$y_i$`'s and placeholders where there is no data, L denotes the discrete matrix operator and

```
\begin{equation} C_{i,j} = \left \{ \begin{array}{ll} 1 & \mathbf{x}_i \mbox{ known, } i = j
\\ 0 & \mbox{otherwise.} \end{array} \right.
\end{equation}
```

indicates whether node `$\mathbf{x}_i$` is observed.

In $d-$ dimensions the matrix $A^{(d)}$ of size $M \times M$ expands the first order finite difference curvature and has entries

$$A^{(d)}_{i,j} = \begin{cases} -1 & j \in N(\mathbf{x}_i) \\ \sum_{j \in N(\mathbf{x}_i)} 1 & j = i \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

where `$\mathcal{N}(\mathbf{x}_i)$` is the set of neighbors of the node `$\mathbf{x}_i$`. Note that if node i is a boundary node, the row `$A^{(d)}_{i,:}$` has -1s in the neighboring node spots and the number of such nodes on the diagonal. In general, the rows of $A^{(d)}$ sum to zero.

Denote by $L = A^{(d)}$ the discrete analog of the Laplacian operator in (4). To use the Matern operator, one substitutes $L = B^{(d)}(m, \epsilon) = ((A^{(d)})^m - \epsilon^2 I)$ in (4). Importantly, A is sparse, containing at most 5 nonzero entries per row when $d = 2$ and 7 nonzero entries per row when $d = 3$ and so on. The Mat'ern matrix $B^{(d)}(m, \epsilon)$ is also sparse, having $2(m + d) - 1$ nonzero entries per row. The sparsity of the matrix allows for the solution to (4) in linear time.

Statement of Need

While there exist numerous implementations of interpolation routines that fill missing data points on arbitrary grids, these are (i) largely restricted to one and two dimensions (ii) slow to run. The implementation we propose is dimension-agnostic, based on a linear-time algorithm, and implements an approximate

Matern kernel interpolation (of which thin plate splines, polyharmonic splines, and radial basis functions are a special case.)

Why is it so fast?

This is because the problem largely boils down to the solution of $Ax = b$ [@mainberger2011optimising] where the square matrix A's size is the product of the number of points in each of the dimensions, and is dense. For the special case where the data points are on a regular grid, and the Matern kernel interpolant is used, a remarkable simplification occurs, in which a discrete approximation to the Green's function for the operator results in an interpolant having sparse matrix representation.

Other software for interpolation

Existing, related software includes, as of the time of this writing

Julia

- Interpolations.jl does B-splines and Lanczos interpolation, and has support for irregular grids.
- Dieerckx.jl a julia-wrapped Fortran package for 1-D and 2-D splins.
- GridInterpolations.jl
- Laplacians.jl, whose function `harmonic_interp` is similar to our vanilla implementation.

Python

Statement of Need

While there exist numerous implementations of interpolation routines that fill missing data points on arbitrary grids, these are (i) largely restricted to one and two dimensions (ii) slow to run. The implementation we propose is dimension-agnostic, based on a linear-time algorithm, and implements an approximate Matern kernel interpolation (of which thin plate splines, polyharmonic splines, and radial basis functions are a special case.)

Why is it so fast?

This is because the problem largely boils down to the solution of $Ax = b$ [@mainberger2011optimising] where the square matrix A's size is the product of the number of points in each of the dimensions, and is dense. For the special case where the data points are on a regular grid, and the Matern kernel interpolant is used, a remarkable simplification occurs, in which a discrete approximation

to the Green's function for the operator results in an interpolant having sparse matrix representation.

Other software for interpolation

Existing, related software includes, as of the time of this writing

Julia

- Interpolations.jl does B-splines and Lanczos interpolation, and has support for irregular grids.
- Dieerckx.jl a julia-wrapped Fortran package for 1-D and 2-D splins.
- GridInterpolations.jl
- Laplacians.jl, whose function `harmonic_interp` is similar to our vanilla implementation.

Python

- astropy.convolve will interpolate gridded data by rescaling a convoution kernel when it encounters missing values.
- scipy.interpolate.RBF

References