Supervised Scoring with Monotone Multidimensional Splines

Abraham Othman
U.S. Green Building Council
aothman@cs.cmu.edu

Abstract

Scoring involves the compression of a number of quantitative attributes into a single meaningful value. We consider the problem of how to generate scores in a setting where they should be *weakly monotone* (either non-increasing or non-decreasing) in their dimensions. Our approach allows an expert to score an arbitrary set of points to produce meaningful, continuous, monotone scores over the entire domain, while exactly interpolating through those inputs. In contrast, existing monotone interpolating methods only work in two dimensions and typically require exhaustive grid input. Our technique significantly lowers the bar to score creation, allowing domain experts to develop mathematically coherent scores. The method is used in practice to create the LEED Performance scores that gauge building sustainability.

Introduction

In the *supervised scoring problem*, a domain expert assigns scalar scores to objects with a number of quantitative attributes. Formally, for objects with d attributes the supervised scoring problem is the design of a *scoring function* $f : \mathbb{R}^d \mapsto \mathbb{R}$ that serves as a scalar field over the input domain. Informally, scoring is useful for settings in which domains are complex and there is significant value in compressing an object's set of attributes into a single comparable value. In the supervised scoring setting there are no objective scores. Scores are a product of the subjective input of domain experts and the only metric to gauge the validity of a scoring function is its acceptance by acclamation.

The technique developed in this paper was created with a specific use in mind—to assess the sustainability of buildings. The US Green Building Council (USGBC) uses the technique described here for the new LEED Performance scores that rate buildings directly based on their energy and water use. The USGBC provides a reference set of buildings, their resource consumptions, and their desired scores, then our algorithm produces a complex scoring function over the entire domain of potential inputs. On the whole, the USGBC is an organization with deep technical expertise about buildings, but mathematical considerations are not core to their mission. Our approach allows the USGBC to leverage their strengths to create an appropriately complex but still internally adaptable scoring function.

This concrete application motivates several properties of the algorithm we develop here. First, as opposed to a supervised learning or ranking method (Burges et al. 2005; Altman and Tennenholtz 2005), we adopt an interpolative approach. This is because the level of credit (e.g., LEED Gold) hinges on a site receiving a specific numerical value. Furthermore, the scoring function should be continuous, so that small changes in input data result in small changes in score, and it should be monotone, so that scores only increase if buildings become more efficient. Intuitive responses in the scoring function are key to furthering the score's perceived validity.

Based on discussions with practitioners, scores are traditionally developed as follows:

- A domain expert comes up with a set of descriptive scoring functions. (For instance, a radial function scoring a location's distance to the nearest grocery store, or an exponential dropoff function scoring time since a credit card applicant's last credit default.)
- The domain expert determines how to combine those functions.
- The domain expert checks the resulting score function against a reference set of objects to see if the score of those objects "looks right".
- The above steps are repeated until the domain expert is satisfied.

Abusing terminology for the sake of descriptive clarity, we refer to this as the "dual" approach, because the domain expert operates directly on constituent functions of a score and their coefficients. In contrast, the approach we develop in this paper is "primal" in the sense that the domain expert interacts only with objects and their scores.

The primal approach we develop in this paper solves two major issues of the dual approach. First, how to deal with objects the domain expert identifies as mis-scored. In the dual approach, the expert must tweak existing coefficients and potentially develop new scoring functions to adjust the score of an erroneous point. But this could disrupt the score of all of the existing, putatively properly scored points. Because our primal approach is interpolative, adding the mis-scored object (with its correct score) to the input of our scoring algorithm will automatically give that object the desired score without changing the scores of the other reference points.

The second issue with the dual approach is one of big-picture relationships. If the expert wants to ensure that, for instance, the sustainability score of an office building never decreases if that building cuts its greenhouse gas emissions, she must take care to ensure that the scoring functions all obey this property and that they are combined in a way that obeys this property. In the primal approach we develop here, these big-picture relationships are set by labeling object dimensions as either non-increasing or non-decreasing.

Beyond ameliorating these two concerns, a benefit of our approach is its accessibility to domain experts that are not mathematically sophisticated. In the dual methodology, domain experts must be adept at creating and manipulating various mathematical functions and their coefficients, while in the primal approach we develop here they only need to be able to identify and label relevant dimensions and score reference objects. Although the technique was developed for a specific application in sustainability there are many other applications of the technique to domains both in and outside of sustainability. The set of domain experts is logically larger, and intuitively far larger, than the set of domain experts with mathematical expertise. Consequently, we believe that the straightforward approach to scoring we develop here could find widespread adoption.

Technical Preliminaries

In this section we lay the technical groundwork for our exposition and then use that machinery to more formally describe the failures of existing approaches. We begin by presenting the supervised scoring problem formally, then discuss *ad hoc* generalizations of univariate interpolation, and finally discuss existing techniques from the literature.

Formal Problem Statement

We are given a *reference set* of input tuples of points and values

$$\{(\mathbf{x_i}, v(\mathbf{x_i})) \mid \mathbf{x_i} \in \mathbb{R}^d, v(\mathbf{x_i}) \in \mathbb{R}\}_{i=1}^n$$

and a *relationship vector* $\mathbf{r} \in \{-1, +1\}^d$, where $r_k = 1$ indicates that the score function should be non-decreasing in the k-th dimension and $r_k = -1$ indicates that the score function should be non-increasing in the k-th dimension. We will often use the ordering imposed by the relationship vector $\geq_{\mathbf{r}}$, where

$$\mathbf{x} \geq_{\mathbf{r}} \mathbf{y} = \begin{cases} x_i \geq y_i & \text{if } r_i = 1; \\ x_i \leq y_i & \text{if } r_i = -1. \end{cases}$$

For the input points to be compatible with a given relationship vector, we must be unable to find a *contradicting pair* that violates the relation. A pair of input points $\mathbf{x}$ and $\mathbf{y}$ form a contradicting pair if $\mathbf{x} \geq_{\mathbf{r}} \mathbf{y}$ but also $v(\mathbf{x}) < v(\mathbf{y})$. Observe that there are some inputs that are incompatible with any relationship vector (e.g., the `xor` function) and some inputs that are compatible with every relationship vector.

Our goal is to output a scoring function f that satisfies the following three desiderata:

- f is continuous.
- f interpolates through the input points. Formally, $f(\mathbf{x_i}) = v(\mathbf{x_i})$.
- f respects the monotonicity conditions imposed by the relationship vector $\mathbf{r}$. Formally, if $\mathbf{x} \geq_{\mathbf{r}} \mathbf{y}$ then $f(\mathbf{x}) \geq f(\mathbf{y})$.

Looking at these desiderata, it may be thought that a simple simplical interpolation scheme would suffice, but it does not.

How Linear Interpolation Fails

The simple presentation and motivation of the problem suggests that it should admit a simple solution; furthermore, the problem is indeed very straightforward in the univariate context. Consider a simple univariate monotone problem, with a set of inputs $(x_i, v(x_i))$. We can create a function satisfying our desiderata by simply drawing a line segment between each pair of consecutive points.

Now consider how to extend this idea into more than one dimension. For simplicity, consider two dimensions. As discussed in (Judd 1998), in unstructured multidimensional contexts the linear interpolation scheme is analogous to a triangulation scheme. First, the input points are triangulated, then linear interpolants are used for points interior to that triangle. (In d dimensions, this process involves sharding the space into convex simplices defined by $d + 1$ points.) This scheme is continuous over all the triangles.

However, even if the input points defining the triangle obey the monotonicity property, there is no guarantee that the interpolating triangle is monotone. Consider Figure 1, a simple example with four input points, with desired scores indicated by the numbers. The black arrows indicate the relationship that the score should be non-increasing in each axis. Observe that the set of input points and values obeys the desired monotone relationship. Furthermore, it is easy to verify that the right interpolating triangle satisfies the monotone relationship. However, the left interpolating triangle does not; its values are increasing in the x axis, because moving along the x axis increases the weight of the high-valued points of 95 and 100, while decreasing the weight of the low-valued point of 5. Consequently, even though the input points satisfy the monotone property, straightforward simplical interpolation does not produce this property in the output scoring function.

Now that we have demonstrated the simple *ad hoc* approach of simplical interpolation is insufficient, we proceed to discuss existing schemes of increased complexity from the literature which have the potential to satisfy our desiderata.

Existing Multidimensional Schemes

To our knowledge, there are at least two prominent existing monotone multidimensional interpolation schemes in the literature that may meet our set of desiderata. However, both of the schemes have been developed only for two-dimensional problems and require the input data to have specific forms.

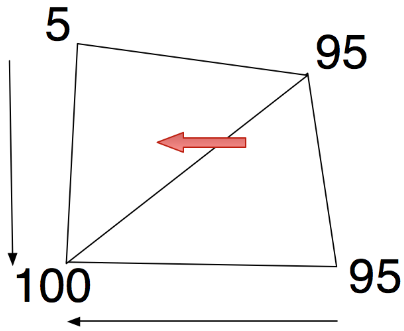

Figure 1: Illustration of the failure of simplical interpolation to maintain monotonicity of inputs. The left interpolating triangle is increasing, rather than non-increasing, along the x axis.

- (Carlson and Fritsch 1989) introduce the `BIMOND` method, which preserves monotonicity in two dimensions. It is an extension of a well-known technique by the same authors for fitting monotone cubic Hermite polynomials to univariate splines (Fritsch and Carlson 1980; Kocic and Milovanovic 1997). `BIMOND` requires function evaluations at all the points along a grid. (That is, it requires $v(x_i, y_j)$ for all x_i and y_j.)

- Constantini shape-preserving interpolation (Constantini and Fontanella 1990). This function preserves both monotonicity and convexity/concavity of the underlying function. It requires both function evaluations and partial derivatives at each point of the grid. To our knowledge, this technique has only been used in two-dimensional problems in the literature, although it appears from first principles that it could be extended into more than two dimensions (Wang and Judd 2000; Othman and Sandholm 2012).

In contrast to these approaches, the method we develop satisfies our desiderata with arbitrary input points in an arbitrary number of dimensions.

A third technique, bi-variate spline triangulation, exists in the literature and (at least mathematically) it could meet our desiderata in two dimensions over arbitrary inputs. (Lai and Schumaker 2007, Theorem 3.10) demonstrate how, by adding side constraints to the control problem for two-dimensional triangular spline interpolation, a monotone spline relative to any direction can be produced. By adding such constraints along each dimension, we could produce a monotone triangulation. However, the considerations here are complex and whether spline triangulations that preserve monotonicity can be used for three dimensions, or more, remains an open question. In this paper we develop a related approach that uses a more complex atomic interpolating region—hypercubes as opposed to simplices—that scales to multiple dimensions in a simpler way.

Splines and the Spline Program

In this section we describe the mathematical approach we use to satisfy our desiderata. We begin by introducing *B*-spline bases, and then we demonstrate how the properties of these bases can be used to ensure continuity, monotonicity, and interpolation.

Multidimensional *B*-spline Basis

B-spline bases of two dimensions are commonly used in computer vision and graphics to ensure smooth curves based on a set of knots or control points (Bartels, Beatty, and Barsky 1987; Blake and Isard 2000, Chapter 3). (The practical use of *B*-spline bases in graphics partially explains why their qualities over one, two, and to a lesser extent, three, dimensions have been explored in detail, while their qualities over larger number of dimensions are less explored.) In one dimension, the theory and practice of monotone interpolation using *B*-spline bases is well developed; (Wolberg and Alfy 2002) provide a comprehensive discussion.

While *B*-spline functions have desirable smoothness properties and rapid evaluation, we use them for scoring because monotonicity of their coefficients carries over to the function itself, and because they always sum to unity. These are qualities that are not shared by general or *ad hoc* bases.

***B*-spline Bases in One Dimension** In one dimension, given a set of knots $z_1, \ldots, z_t$ *B*-spline bases of degree b can be defined recursively by $\beta_i^0(x) = 1$ if $x \in [z_i, z_{i+1}],$ 0 otherwise, and

$$\beta_i^b(x) = \frac{x - z_i}{z_{i+k-1} - z_i}\beta_i^{b-1}(x) + \frac{z_{i+k} - x}{z_{i+k} - z_{i+1}}\beta_{i+1}^{b-1}(x)$$

(For clarity, when the degree of function b is fixed we will not write out the superscript.)

We pair each basis function β_k with a specific coefficient value v_k in order to arrive at a spline $v : \mathbb{R} \mapsto \mathbb{R}$ that can be evaluated at arbitrary points in the interval spanned by the knots:

$$v(x) = \sum_i \beta_i(x) v_i$$

We use the well-known technique of appending b copies of the first and last knot to the first and last elements of the set of knots to bound the basis. This results in the first and last basis functions having a distinct appearance than the other basis functions in the set. Figure 2 shows the *B*-spline bases for $b = 3$ and knots at $0, 5, 7, 9$ and 10. Observe how, even though the knots are not equally spaced, the basis functions always sum to unity and at most $b + 1$ basis functions are non-zero.

Because of the shape of our first and last basis function, we always include the extreme points as knots within the spline. However, selecting the interior knots for the bases is more art than science. Options can include selecting knots corresponding to equally-spaced percentiles within the data, or equally spaced knots in either the linear or logarithmic space of input values.

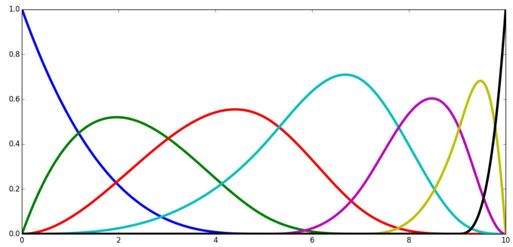

Figure 2: One-dimensional *B*-spline basis function evaluations.

B-spline Bases in More than One Dimension In order to produce a multidimensional basis, we take the tensor product of the set of univariate *B*-spline bases. We index each *B*-spline β by the vector index $\mathbf{k} = (k_1, k_2, \ldots, k_d)$. For instance, $\beta_{(1,7,6)}$ is the first *B*-spline in the first dimension, the seventh in the second, and the sixth in the third. Similarly, $v_{\mathbf{k}}$ becomes the coefficient associated with the function $\beta_{\mathbf{k}}$.

Because we take the tensor product over univariate interpolating splines, our resulting interpolating spline is defined over the hypercube $\mathcal{H}$ of the extreme points in the input in each dimension. If our set of d-dimensional input points is Z, then

$$\mathcal{H} = \bigtimes_{i=1}^{d} [\min_{\mathbf{z}\in Z} z_i, \max_{\mathbf{z}\in Z} z_i]$$

At any input $\mathbf{x} \in \mathcal{H}$, we evaluate the spline by taking

$$\sum_{(\beta_{\mathbf{k}}, v_{\mathbf{k}})} \beta_{\mathbf{k}}(\mathbf{x}) v_{\mathbf{k}}$$

Now that we have described the set of basis functions, we proceed to describe the two sets of constraints, interpolation and monotonicity, that must hold for the scoring spline to achieve our desiderata.

Interpolation Constraints

In order for the spline to actually interpolate through the inputs, evaluating the spline function at each input point must produce the specified input value. Formally, for each input point $\mathbf{z}$ with associated value $v(\mathbf{z})$ we need the following relation to hold

$$\sum_{(\beta_{\mathbf{k}}, v_{\mathbf{k}})} \beta_{\mathbf{k}}(\mathbf{z}) v_{\mathbf{k}} = v(\mathbf{z})$$

Observe that, although this appears to be a sum over t^d terms, only $(b+1)^d$ basis functions are non-zero.

Monotonicity Constraints

The second set of constraints we construct ensure that the resulting score function is monotone in the way desired by the expert. In general, for each interpolating knot $\mathbf{k}$, we have a set of d constraints, one for each dimension, constructed so that the knot satisfies the monotonicity relationship locally. Recall that the relationship vector $\mathbf{r}$ establishes a relationship $\geq_{\mathbf{r}}$ for the d-dimensions of the input. Each input dimension corresponds to a pairwise constraint at each basis vector:

$$\begin{aligned} v_{\mathbf{k}} &\geq_{r_1} v_{(k_1-1, k_2, \ldots, k_d)} \\ v_{\mathbf{k}} &\geq_{r_2} v_{(k_1, k_2-1, \ldots, k_d)} \\ &\vdots \\ v_{\mathbf{k}} &\geq_{r_d} v_{(k_1, k_2, \ldots, k_d-1)} \end{aligned}$$

However, if an interpolating knot $\mathbf{k}$ has $k_i = 1$ ($\mathbf{k}$ is a minimal knot in the i-th dimension) then the i-th constraint for that knot is omitted. Thus, there are $(t-1)^d$ monotonicity constraints.

Any solution to these constraints yields spline knots with coefficients that satisfy the desired monotonicity relationship. We may wonder, however, if that is sufficient to ensure the desired monotonicity of the entire scoring function. The answer is not trivial. Recall from the linear interpolating triangle of Figure 1 that it is possible for the control points of an interpolating scheme to satisfy a monotonicity property without imparting that quality to the scheme itself. In order to show that our resulting multidimensional spline is monotone we use the following result from the literature:

Lemma 1. *(Schumaker 2007, Theorem 4.76) If the coefficients of a one-dimensional spline are monotone, then the resulting spline is monotone in the same way.*

Equipped with this result we proceed to prove the monotonicity of the overall spline.

Theorem 1. *If the pairwise monotonicity constraints on the knot coefficients are satisfied, then for any $\mathbf{x}, \mathbf{y} \in \mathcal{H}$ with $\mathbf{x} \geq_{\mathbf{r}} \mathbf{y}$, $v(\mathbf{x}) \geq v(\mathbf{y})$.*

Proof Sketch. Consider $\mathbf{x}$ and $\mathbf{y}$, identical except for their i-th dimension, and with $x_i \geq_{r_i} y_i$. Because we form multidimensional splines by taking the tensor product of single dimensional splines, the i-th dimension of the larger spline is itself a single-dimensional spline with a set of coefficients that are monotone. So by Lemma 1, $v(\mathbf{x}) \geq v(\mathbf{y})$.

The result holds for any two vectors $\mathbf{x} \geq_{\mathbf{r}} \mathbf{y}$ by repeating the argument for all the dimensions on which they differ. □

Maximum and Minimum Constraints

Specific applications may require that scores should be confined within some range (e.g., 0 to 100). These constraints are simple to add. A minimum score of $\underline{v}$ can be set with $v_{\mathbf{k}} \geq \underline{v}$ for every $\mathbf{k}$, while a maximum score of $\overline{v}$ can be set with $v_{\mathbf{k}} \leq \overline{v}$ for every $\mathbf{k}$. The following result shows that these constraints propagate from the knots to the scoring function as a whole.

Theorem 2. *If $\underline{v} \leq v_{\mathbf{k}} \leq \overline{v}$, then for any $\mathbf{x} \in \mathcal{H}$, $\underline{v} \leq v(\mathbf{x}) \leq \overline{v}$.*

Proof. *B*-spline basis functions are non-negative and sum to unity. Thus

$$\underline{v} = \sum_{(\beta_{\mathbf{k}}, v_{\mathbf{k}})} \beta_{\mathbf{k}}(\mathbf{x})\underline{v} \leq \sum_{(\beta_{\mathbf{k}}, v_{\mathbf{k}})} \beta_{\mathbf{k}}(\mathbf{x})v_{\mathbf{k}} \leq \sum_{(\beta_{\mathbf{k}}, v_{\mathbf{k}})} \beta_{\mathbf{k}}(\mathbf{x})\overline{v} = \overline{v}$$

□

Solving and Applying the Feasibility Program

The interpolation and monotonicity constraints define a linear feasibility program that takes as input a set of knots, a set of scored inputs, and a relationship vector and (if feasible) outputs a set of coefficients that define a scoring function that interpolates over the set of inputs and are monotone in the way described by the relationship vector.

In this section, we discuss applications and extensions of the feasibility program. We start by discussing potential smoothness objectives to pair with the feasibility constraints. We then discuss the procedure for finding a set of knots that guarantee feasibility.

Finally, once the scoring function has been generated, we discuss how to extrapolate and score points outside of the defined hypercube $\mathcal{H}$ in a way that preserves the monotone properties of the spline. The end result is a smooth, monotone, and interpolating spline defined over every potential input.

Setting the Objective

Without an objective, the linear feasibility program we have constructed will output some feasible solution. However, certain solutions are preferable over others. In general, we should prefer the "smoothest" or "least bent" solution. This objective has a historical antecedent; *B*-spline bases were originally designed to mimic a draftsman's flexible spline (Malcolm 1977; de Boor 1978).

Currently, we use the squared difference in values at adjacent knots for smoothness in the LEED Performance scores but investigating different smoothing approaches remains an active topic of interest. Since B-spline derivatives are fast to compute, one area of interest is to minimize quantities related to the derivatives, such as the sum of instantaneous derivatives or second derivatives. We are guided in this approach by (Eilers and Marx 1996), who explore smoothing techniques in regression using splines.

Finding the Basis

At a small number of interior knots and large number of inputs, the linear feasibility problem is, most likely, infeasible. This is because each coefficient may be involved in a large number of mutually unsatisfiable constraints. To find the interior knots, we adopt some dense method of moving from a desired number of knots to producing those knots in each dimension. Then, starting from a small number of interior knots, we progressively increase the number of knots in each dimension until the feasibility problem is solved. The following result shows that this procedure eventually terminates.

Theorem 3. *If the method of selecting knots is dense along each dimension, iteratively increasing the number of knots eventually produces a feasible solution to the monotonicity and interpolation constraints.*

Proof Sketch. At a sufficiently large number of knots along each dimension, each input point becomes isolated from every other input point, so that at any two input points $\mathbf{z_1}$ and $\mathbf{z_2}$ and every $\beta_{\mathbf{k}}$ in the basis, $\beta_{\mathbf{k}}(\mathbf{z_1}) \cdot \beta_{\mathbf{k}}(\mathbf{z_2}) = 0$. An interpolating solution is then found by setting the coefficients of the basis functions that are non-zero at each input point to the value of that input point. A monotone solution can then be guaranteed by setting the coefficient of the other basis functions to their smallest possible interpolating value. □

Extrapolation

Recall that our interpolating spline is only defined within the hypercube $\mathcal{H}$ that is bounded by the extreme input values along each dimension. In this section we describe how to treat points outside of $\mathcal{H}$.

Assume that the spline is monotone non-decreasing in all of its input dimensions. Then if a point is outside of the hypercube, it falls into one of the following three categories:

- If it is larger in each of its dimensions than the maximal corner of the hypercube, it receives the score of the maximal corner of the hypercube.
- If it is smaller in *any* of its dimensions than the minimum boundary of the hypercube in that dimension, it receives the score of the minimal corner of the hypercube.
- If it is larger in some dimensions, but within the hypercube in all others, it receives the score obtained by projecting the point to its nearest boundary of the hypercube.

(This logic can be generalized to non-increasing dimensions by reversing the signs of the comparisons along those dimensions.)

Figure 3 below depicts the process for a simple two-dimensional example. Both axes are monotone non-increasing, and the defined hypercube (here, a rectangle) is denoted by the white center region. The green region is strictly better than the defined region according to the relationship vector so it gets the maximal score. The red region is worse than the defined region along at least one dimension and so it gets the minimal score. The yellow region is better along one dimension but not better along the other. To score points in the yellow region, they are projected up or to the right to the border of the defined region and then scored.

Observe that this process still fulfills the monotonicity, continuity, and interpolation desiderata. However, it does this by taking a pessimistic approach to points outside the defined hypercube. Practitioners are therefore advised to make the hypercube relatively large by including extreme input points along each dimension.

Demonstration

In this section we demonstrate our method graphically on a toy example. Consider the following reference set:

$$(0,0) \mapsto 0 \qquad (1,0) \mapsto 20$$
$$(0,2) \mapsto 50 \qquad (3,1) \mapsto 60$$
$$(3,3) \mapsto 90 \qquad (4,4) \mapsto 100$$

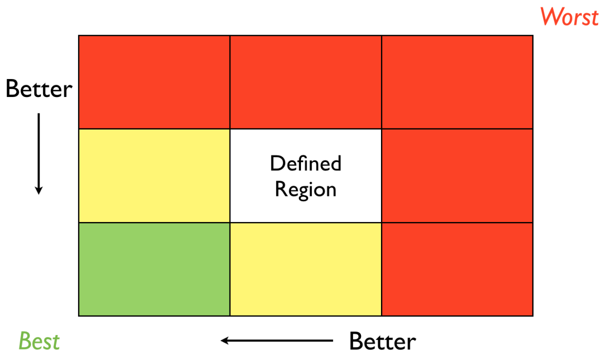

Figure 3: Different regions have different rules for extrapolating scores.

with the relationship vector $\{1, 1\}$ and values restricted to $[0, 100]$. This input produces the interpolating spline shown in Figure 4 on $[0, 4]^2$. Observe that the function is monotone non-decreasing in both dimensions.

When we add the point $(1.5, 1.5) \mapsto 75$, Figure 5 shows the resulting interpolating spline. Observe how adding this additional point changes the behavior of the function along the second dimension in order to preserve monotonicity.

Conclusion and Future Directions

Starting from a set of expert-scored reference points, we developed a methodology that produces a scoring function that interpolates through those points and obeys the expert's desired monotonicity relationships. This technique is applied in practice to produce the LEED Performance energy and water scores for the USGBC, but we believe it may hold significant interest to other applications both in and outside of sustainability. This is because, unlike existing dual approaches, our methodology does not require mathematical sophistication on the part of the domain expert creating the score. It is our belief that the limiting factor of scores is not a paucity of interest or domain expertise but rather a lack of domain experts that are also mathematically sophisticated, as these skills can be orthogonal in many domains.

Additional work should be done on the problems of smoothness and knot selection. The literature on one-dimensional smoothness for *B*-spline bases is quite sophisticated and it would be interesting to apply more of these techniques to the multidimensional problem. We are particularly interested in applications of quadrature on the hypercube to calculate the integrals in our smoothness objectives. Knot selection, and specifically automated knot selection, is an interesting problem as well. Perhaps an expert's reference set could be automatically checked in each dimension to determine whether to logarithmically transform the inputs. Additionally, given a large number of potential objects to score it would be valuable to develop automated ways to generate a reference set for an expert to focus on. For instance, the expert should be encouraged to include the convex hull of potential objects in her reference set.

Finally, the approach we described in this paper suffers from the curse of dimensionality, because the number of total knots scales exponentially in the dimension d of the input. As a practical matter this limits us to scoring applications with a small number of dimensions. In order to break out of this curse, multidimensional monotone simplical approaches to interpolation should be developed. (Lai and Schumaker 2007) pursue this idea, but only in two dimensions. In more dimensions the considerations are complex (Alfeld 1996; Foucart and Sorokina 2013) and remain theoretical.

Acknowledgements

We thank Stephen Dawson-Haggerty, Dhruv Gami, Scot Horst, Andrew Krioukov, Gautami Palanki, Gretchen Sweeney, and Lauren Riggs, as well as Jay Taneja and the staff at IBM Research – Africa, for constructive discussions.

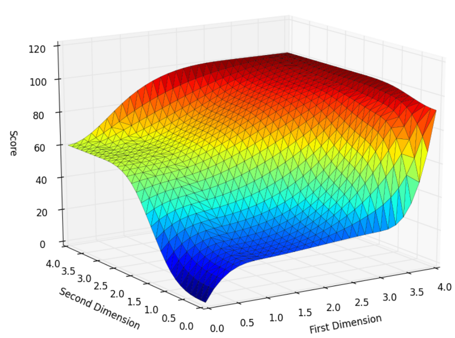

Figure 4: Demonstration spline interpolating through input data.

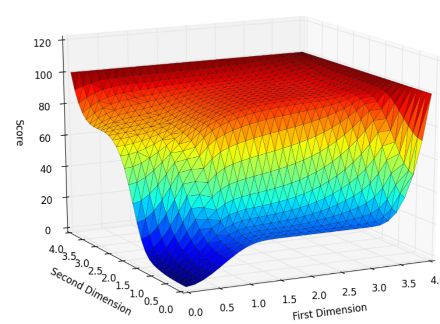

Figure 5: The shape of the spline changes after we add an additional input point.

References

Alfeld, P. 1996. Upper and lower bounds on the dimension of multivariate spline spaces. *SIAM Journal on Numerical Analysis* 33(2):571–588.

Altman, A., and Tennenholtz, M. 2005. On the axiomatic foundations of ranking systems. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI)*.

Bartels, R. H.; Beatty, J. C.; and Barsky, B. A. 1987. *An Introduction to Splines for Use in Computer Graphics and Geometric Modeling*. Morgan Kaufmann.

Blake, A., and Isard, M. 2000. *Active Contours: The Application of Techniques from Graphics, Vision, Control Theory and Statistics to Visual Tracking of Shapes in Motion*. Springer.

Burges, C.; Shaked, T.; Renshaw, E.; Lazier, A.; Deeds, M.; Hamilton, N.; and Hullender, G. 2005. Learning to rank using gradient descent. In *Proceedings of the 22nd International Conference on Machine Learning*, ICML '05, 89–96.

Carlson, R. E., and Fritsch, F. N. 1989. An algorithm for monotone piecewise bicubic interpolation. *SIAM J. Numer. Anal.* 26(1):230–238.

Constantini, P., and Fontanella, F. 1990. Shape-preserving bivariate interpolation. *SIAM J. Numer. Anal.* 27:488–506.

de Boor, C. 1978. *A Practical Guide to Splines*. New York: Springer-Verlag.

Eilers, P. H. C., and Marx, B. D. 1996. Flexible smoothing with b-splines and penalties. *Statistical Science* 11(2):89–121.

Foucart, S., and Sorokina, T. 2013. Generating dimension formulas for multivariate splines. *Albanian Journal of Mathematics* 7(1):24–35.

Fritsch, F., and Carlson, R. 1980. Monotone piecewise cubic interpolation. *SIAM Journal on Numerical Analysis* 17(2):238–246.

Judd, K. 1998. *Numerical methods in economics*. The MIT Press.

Kocic, L., and Milovanovic, G. 1997. Shape preserving approximations by polynomials and splines. *Computers & Mathematics with Applications* 33(11):59 – 97.

Lai, M.-J., and Schumaker, L. L. 2007. *Spline Functions on Triangulations*. Cambridge University Press.

Malcolm, M. A. 1977. On the computation of nonlinear spline functions. *SIAM Journal of Numerical Analysis* 14(2):254–282.

Othman, A., and Sandholm, T. 2012. Rational market-making with probabilistic knowledge. In *Autonomous Agents and Multi-Agent Systems*, 645–652.

Schumaker, L. L. 2007. *Spline Functions: Basic Theory*. Cambridge University Press, 3rd edition.

Wang, S.-P., and Judd, K. L. 2000. Solving a savings allocation problem by numerical dynamic programming with shape-preserving interpolation. *Comput. Oper. Res.* 27:399–408.

Wolberg, G., and Alfy, I. 2002. An energy-minimization framework for monotonic cubic spline interpolation. *Journal of Computational and Applied Mathematics* 143(2):145–188.