Toward interpretable machine learning via Delaunay interpolation

Algorithms and challenges

Tyler Chang

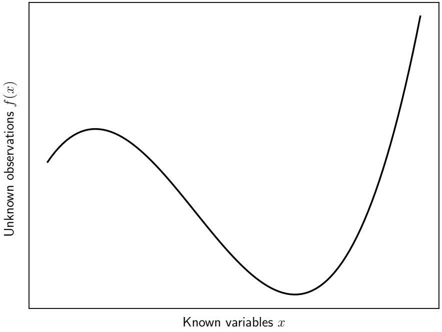
Argonne National Laboratory

LANS Seminar Series
July 12, 2023

Outlines

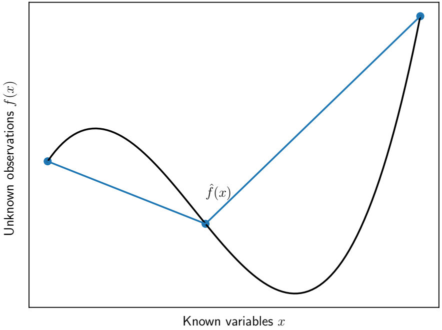
The fundamental machine learning problem

The fundamental machine learning problem



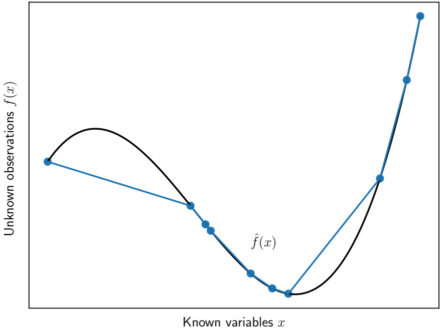
- Want to predict unknown $f(x)$ for observation x

The fundamental machine learning problem



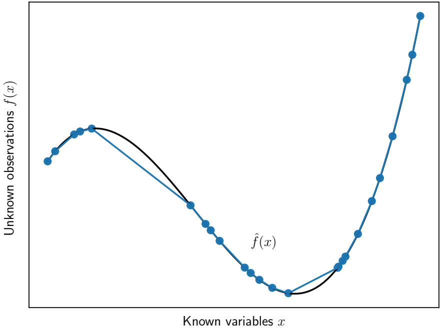
- Want to predict unknown $f(x)$ for observation x
- **ML**: *Learn* approximation $\hat{f} \sim f$ based on *training data* $\mathcal{X}$
- **NA**: fit an interpolant (piecewise-linear) to f on $\mathcal{X}$

The fundamental machine learning problem



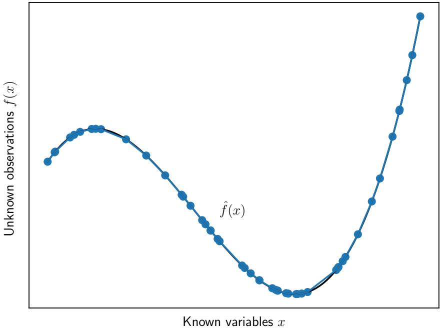
- Want to predict unknown $f(x)$ for observation x
- **ML**: *Learn* approximation $\hat{f} \sim f$ based on *training data* $\mathcal{X}$
- **NA**: fit an interpolant (piecewise-linear) to f on $\mathcal{X}$
- Both cases: more data $\Rightarrow$ better $\hat{f}$

The fundamental machine learning problem



- Want to predict unknown $f(x)$ for observation x
- **ML**: *Learn* approximation $\hat{f} \sim f$ based on *training data* $\mathcal{X}$
- **NA**: fit an interpolant (piecewise-linear) to f on $\mathcal{X}$
- Both cases: more data $\Rightarrow$ better $\hat{f}$
- Real data not perfectly balanced $\Rightarrow$ $\hat{f} \to f$ non-uniformly

The fundamental machine learning problem



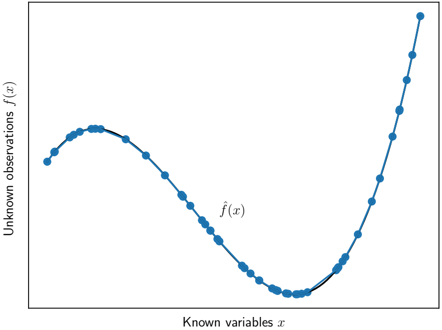
- Want to predict unknown $f(x)$ for observation x
- **ML**: *Learn* approximation $\hat{f} \sim f$ based on *training data* $\mathcal{X}$
- **NA**: fit an interpolant (piecewise-linear) to f on $\mathcal{X}$
- Both cases: more data $\Rightarrow$ better $\hat{f}$
- Real data not perfectly balanced $\Rightarrow$ $\hat{f} \to f$ non-uniformly
- If we have enough data, it doesn't matter

Some basic numerical analysis results

When $\hat{f}$ is a piecewise linear spline:

For h "small enough" – let q be the querry point

$$|f(q) - \hat{f}(q)| \sim \mathcal{O}(h^2)$$



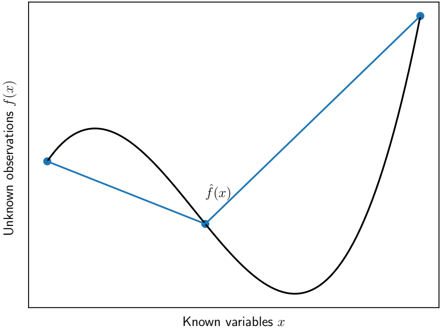
- h is a "mesh fineness" parameter $\sim$ distance between points in $\mathcal{X}$
- For irregular $\mathcal{X}$, h could be the distance from q to the nearest neighbor in $\mathcal{X}$
- Constants proportional to the Lip constant of ∇f

Some basic numerical analysis results

When $\hat{f}$ is a piecewise linear spline:

For h “small enough” – let q be the querry point

$$|f(q) - \hat{f}(q)| \sim \mathcal{O}(h^2)$$



- h is a “mesh fineness” parameter $\sim$ distance between points in $\mathcal{X}$
- For irregular $\mathcal{X}$, h could be the distance from q to the nearest neighbor in $\mathcal{X}$
- Constants proportional to the Lip constant of ∇f

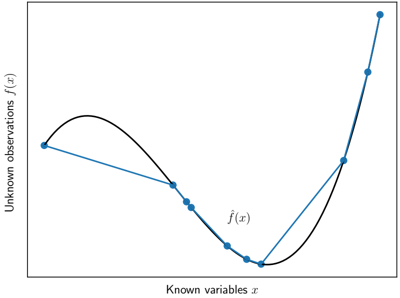
Some basic deep learning

- Train a fully-connected multi-layer perceptron (MLP) using $\mathcal{X}$
- The most popular activation function is ReLU (piecewise linear)
- In modern ML, train as close to zero error as possible (interpolate)

Some basic deep learning

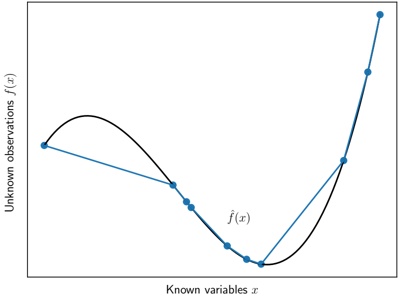
- Train a fully-connected multi-layer perceptron (MLP) using $\mathcal{X}$
- The most popular activation function is ReLU (piecewise linear)
- In modern ML, train as close to zero error as possible (interpolate)

- Piecewise linear interpolant ✓



Some basic deep learning

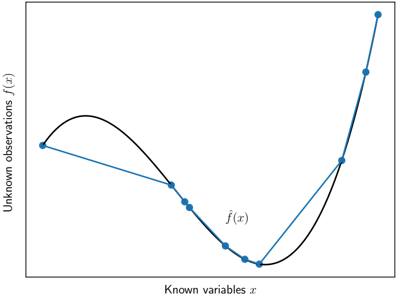
- Train a fully-connected multi-layer perceptron (MLP) using $\mathcal{X}$
- The most popular activation function is ReLU (piecewise linear)
- In modern ML, train as close to zero error as possible (interpolate)



- Piecewise linear interpolant ✓
- Scalable to large training sets $\mathcal{X}$ and dimension d ✓

Some basic deep learning

- Train a fully-connected multi-layer perceptron (MLP) using $\mathcal{X}$
- The most popular activation function is ReLU (piecewise linear)
- In modern ML, train as close to zero error as possible (interpolate)

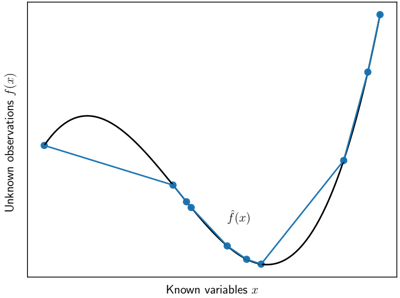


- Piecewise linear interpolant ✓
- Scalable to large training sets $\mathcal{X}$ and dimension d ✓

- Error bounds ✗

Some basic deep learning

- Train a fully-connected multi-layer perceptron (MLP) using $\mathcal{X}$
- The most popular activation function is ReLU (piecewise linear)
- In modern ML, train as close to zero error as possible (interpolate)



- Piecewise linear interpolant ✓
- Scalable to large training sets $\mathcal{X}$ and dimension d ✓

- Error bounds ✗
- Verifiability and interpretability ✗

Real machine learning

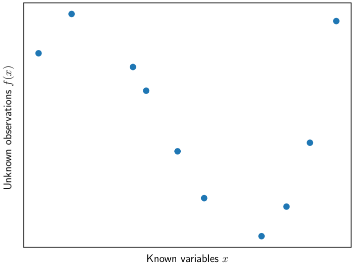
“There’s more to machine learning than function approximation”

Real machine learning

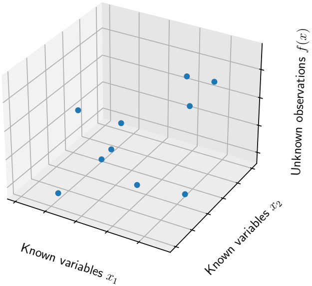
“There’s more to machine learning than function approximation”

- Training samples $\mathcal{X}$ are *high-dimensional* and *mixed-variables*
- Training samples $\mathcal{X}$ could be *noisy* or f could be stochastic
- f is often highly *structured* – MLPs with nothing else are from the 60s

The curse of dimensionality

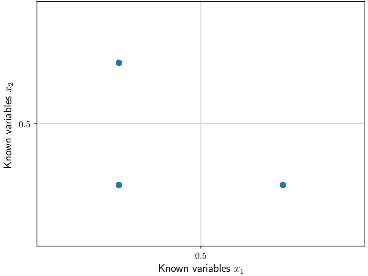


10 training points in 1D



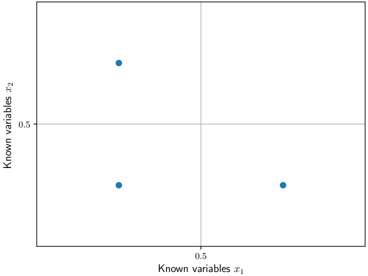
10 training points in 2D

The curse of ~~dimensionality~~ no data



Need data in all quadrants?

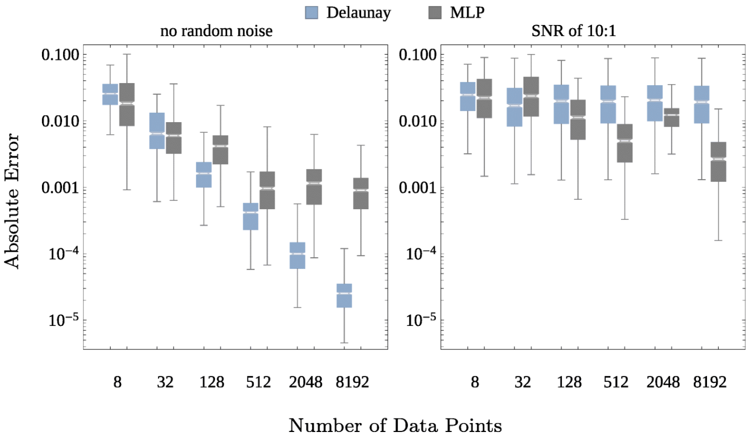
The curse of ~~dimensionality~~ no data



Need data in all quadrants?

- Inference in 2D : $2^2 = 4$
- Inference in 10D : $2^{10} \approx 1000$
- Inference in 100D : $2^{100} \approx 10^{30}$ (orders of magnitude bigger than exascale)
- Many ML problems : inference in 1000+ dimensions

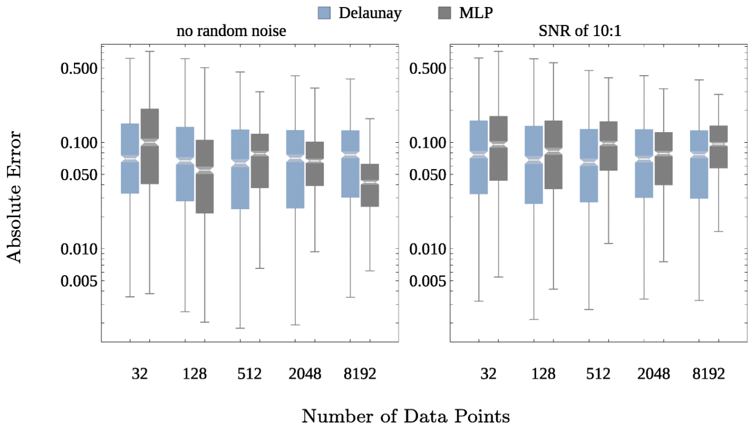
The blessing of dimensionality (no noise)



Delaunay interpolation vs MLP error in **2D** with and w/o noise

Lux, Watson, Chang, et al. Interpolation of sparse high-dimensional data. *Numerical Algorithms 88*, pp. 281–313 (2021).

The blessing of dimensionality (no noise)



Delaunay interpolation vs MLP error in **20D** with and w/o noise

Lux, Watson, Chang, et al. Interpolation of sparse high-dimensional data. *Numerical Algorithms 88*, pp. 281–313 (2021).

The hopelessness of dimensionality

Can we still make good predictions where we **do** have data?

The hopelessness of dimensionality

Can we still make good predictions where we **do** have data?

No, because we have no data anywhere

We measure where we *might* have enough data to make a prediction using the "convex hull" of the training data $CH(\mathcal{X})$

The hopelessness of dimensionality

Can we still make good predictions where we **do** have data?

No, because we have no data anywhere

We measure where we *might* have enough data to make a prediction using the "convex hull" of the training data $CH(\mathcal{X})$

If $\mathcal{X}$ are sampled from *any* distribution, $\mu(CH(\mathcal{X})) \to 0$ *exponentially* as d grows

This is called a *concentration of measure*

Gorban and Tyukin. Stochastic separation theorems. *Neural Networks 94*, pp. 255-259 (2017).

Example

Suppose that we uniformly sample $x = (x_1, x_2, \ldots, x_d)$ from $[0,1]^d$

$$\|x - \frac{1}{2}\|_2^2 = \sum_{i=1}^{d} (x_i - \frac{1}{2})^2.$$

$$\mathbb{E}\left[\left(x_i - \frac{1}{2}\right)^2\right] = \int_0^1 \left(u - \frac{1}{2}\right)^2 du = \frac{1}{12}$$

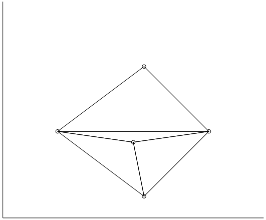
with finite variance ν

By CLT for all $x \in \mathcal{X}$: $\mathbb{E}[\|x - \frac{1}{2}\|_2^2] = \frac{d}{12}$ with variance $\frac{\nu}{d} \to 0$ as $d \to \infty$.

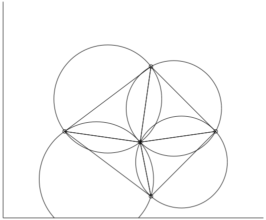
Garg, Chang, and Raghavan. Stochastic optimization of Fourier coefficiencts to generate space-filling designs. *To appear in Winter Sim 2023.*

About Delaunay Triangulations

- The *Delaunay triangulation* is an unstructured simplicial mesh defined by an arbitrary vertex set $\mathcal{X} = \{x^{(1)}, \ldots, x^{(n)}\} \subset \mathbb{R}^d$
- The defining property of the Delaunay triangulation $DT(P)$ is that for every simplex $S \in DT(\mathcal{X})$, the circumball $B^{(S)}$ must have empty intersection with $\mathcal{X}$: $B^{(S)} \cap \mathcal{X} = \emptyset$.



✗



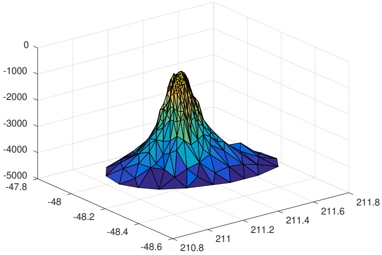
✓

- $DT(\mathcal{X})$ exists and is unique when P is in *general position*.

Delaunay Interpolation

Let $y \in S \in DT(P)$. S has vertex set $\{s^{(1)}, \ldots, s^{(d+1)}\}$ and there exist convex weights $\{w_1, \ldots, w_{d+1}\}$ such that $y = \sum_{i=1}^{d+1} w_i s^{(i)}$.

$$\hat{F}_{DT}(y) = \sum_{i=1}^{d+1} w_i F(s^{(i)}).$$



Scalability Issues

- Oweing to Klee, the size of the Delaunay triangulation is

$$\mathcal{O}\left(n^{\lceil d/2 \rceil}\right)$$

  - For $d > 4$, this is expensive!
  - For $d > 8$, this is not scalable!

Scalability Issues

- Oweing to Klee, the size of the Delaunay triangulation is

$$\mathcal{O}\left(n^{\lceil d/2 \rceil}\right)$$

  - For $d > 4$, this is expensive!
  - For $d > 8$, this is not scalable!

Observation: For interpolation at a single point y, we only need the vertices $(\{s^{(1)}, \ldots, s^{(d+1)}\})$ of $S \in DT(P)$ such that $y \in S$

$$\hat{F}_{DT}(y) = \sum_{i=1}^{d+1} w_i \; F(s^{(i)}).$$

Scalability Issues

- Oweing to Klee, the size of the Delaunay triangulation is

$$\mathcal{O}\left(n^{\lceil d/2 \rceil}\right)$$

  - For $d > 4$, this is expensive!
  - For $d > 8$, this is not scalable!

Observation: For interpolation at a single point y, we only need the vertices $(\{s^{(1)}, \ldots, s^{(d+1)}\})$ of $S \in DT(P)$ such that $y \in S$

$$\hat{F}_{DT}(y) = \sum_{i=1}^{d+1} w_i \; F(s^{(i)}).$$

Question: Can we find S containing y in polynomial time?

DelaunaySparse Algorithm outline

Algorithm to locate Delaunay simplex containing y:

- Grow an initial Delaunay simplex (greedy algorithm) that is “nearby” to y
- “Flip” accross facets from which y is visible to a new Delaunay simplex (closer to y)
- This “visibility walk” converges to y in finite steps (Edelsbrunner’s acyclicity theorem)

Chang, Watson, Lux, Li, Xu, Butt, Cameron, and Hong. “A polynomial time algorithm for multivariate interpolation in arbitrary dimension via the Delaunay triangulation.” In Proc. 2018 ACMSE Conf.

Algorithm Complexity

- To grow the first simplex: $\mathcal{O}(nd^3)$ to apply n rank-1 updates to the QR factorization of $d \times j$ matrix for $j = 1, \ldots, d$
- To compute a flip: $\mathcal{O}(nd^2)$ to apply n rank-1 updates to the QR factorization of a $d \times d$ matrix
- ℓ total flips

	$n = 2K$	$n = 8K$	$n = 16K$	$n = 32K$
$d = 2$	3.05	2.90	3.25	3.10
$d = 8$	23.75	24.75	24.30	23.10
$d = 32$	95.25	125.60	131.85	150.10
$d = 64$	171.95	221.85	248.35	280.60

Algorithm Complexity

- To grow the first simplex: $\mathcal{O}(nd^3)$ to apply n rank-1 updates to the QR factorization of $d \times j$ matrix for $j = 1, \ldots, d$
- To compute a flip: $\mathcal{O}(nd^2)$ to apply n rank-1 updates to the QR factorization of a $d \times d$ matrix
- ℓ total flips

	$n = 2K$	$n = 8K$	$n = 16K$	$n = 32K$
$d = 2$	3.05	2.90	3.25	3.10
$d = 8$	23.75	24.75	24.30	23.10
$d = 32$	95.25	125.60	131.85	150.10
$d = 64$	171.95	221.85	248.35	280.60

Overall complexity: $\mathcal{O}(nd^2\ell)$

Algorithm Complexity

- To grow the first simplex: $\mathcal{O}(nd^3)$ to apply n rank-1 updates to the QR factorization of $d \times j$ matrix for $j = 1, \ldots, d$
- To compute a flip: $\mathcal{O}(nd^2)$ to apply n rank-1 updates to the QR factorization of a $d \times d$ matrix
- ℓ total flips

	$n = 2K$	$n = 8K$	$n = 16K$	$n = 32K$
$d = 2$	3.05	2.90	3.25	3.10
$d = 8$	23.75	24.75	24.30	23.10
$d = 32$	95.25	125.60	131.85	150.10
$d = 64$	171.95	221.85	248.35	280.60

Overall complexity: $\mathcal{O}(nd^2\ell)$

Unresolved question: $\ell \approx d$? ℓ independent of n?

Linear programming interpretation

$$\tilde{A} = \begin{bmatrix} (-x^{(1)})^T & 1 \\ (-x^{(2)})^T & 1 \\ \vdots & \vdots \\ (-x^{(n)})^T & 1 \end{bmatrix}, \; \tilde{b} = \begin{bmatrix} \|x^{(1)}\|_2^2 \\ \|x^{(2)}\|_2^2 \\ \vdots \\ \|x^{(n)}\|_2^2 \end{bmatrix}, \text{ and } \tilde{c} = \begin{bmatrix} -y \\ 1 \end{bmatrix}.$$

Linear programming interpretation

$$\tilde{A} = \begin{bmatrix} (-x^{(1)})^T & 1 \\ (-x^{(2)})^T & 1 \\ \vdots & \vdots \\ (-x^{(n)})^T & 1 \end{bmatrix}, \ \tilde{b} = \begin{bmatrix} \|x^{(1)}\|_2^2 \\ \|x^{(2)}\|_2^2 \\ \vdots \\ \|x^{(n)}\|_2^2 \end{bmatrix}, \text{ and } \tilde{c} = \begin{bmatrix} -y \\ 1 \end{bmatrix}.$$

Primal prob: $\max_{\tilde{u}} \tilde{c}^T \tilde{u}$ such that $\tilde{A}\tilde{u} \leq \tilde{b}, \tilde{u}$ free.

Ext pts: $\tilde{u} = (-2\text{circumcenter}, \text{circumradius}^2 - \|\text{circumcenter}\|_2^2)$

Linear programming interpretation

$$\tilde{A} = \begin{bmatrix} (-x^{(1)})^T & 1 \\ (-x^{(2)})^T & 1 \\ \vdots & \vdots \\ (-x^{(n)})^T & 1 \end{bmatrix}, \; \tilde{b} = \begin{bmatrix} \|x^{(1)}\|_2^2 \\ \|x^{(2)}\|_2^2 \\ \vdots \\ \|x^{(n)}\|_2^2 \end{bmatrix}, \text{ and } \tilde{c} = \begin{bmatrix} -y \\ 1 \end{bmatrix}.$$

Primal prob: $\max_{\tilde{u}} \tilde{c}^T \tilde{u}$ such that $\tilde{A}\tilde{u} \leq \tilde{b}, \tilde{u}$ free.

Ext pts: $\tilde{u} = (-2\text{circumcenter}, \text{circumradius}^2 - \|\text{circumcenter}\|_2^2)$

Dual prob: $\min_{\tilde{v}} \tilde{b}^T \tilde{v}$ such that $\tilde{A}^T \tilde{v} = \tilde{c}, \tilde{v} \geq 0$.

Ext pts: $\Rightarrow$ $\tilde{v}$ are convex weights for y

Linear programming interpretation

$$\tilde{A} = \begin{bmatrix} (-x^{(1)})^T & 1 \\ (-x^{(2)})^T & 1 \\ \vdots & \vdots \\ (-x^{(n)})^T & 1 \end{bmatrix}, \ \tilde{b} = \begin{bmatrix} \|x^{(1)}\|_2^2 \\ \|x^{(2)}\|_2^2 \\ \vdots \\ \|x^{(n)}\|_2^2 \end{bmatrix}, \text{ and } \tilde{c} = \begin{bmatrix} -y \\ 1 \end{bmatrix}.$$

Primal prob: $\max_{\tilde{u}} \tilde{c}^T \tilde{u}$ such that $\tilde{A}\tilde{u} \le \tilde{b}, \tilde{u}$ free.

Ext pts: $\tilde{u} = (-2\text{circumcenter}, \text{circumradius}^2 - \|\text{circumcenter}\|_2^2)$

Dual prob: $\min_{\tilde{v}} \tilde{b}^T \tilde{v}$ such that $\tilde{A}^T \tilde{v} = \tilde{c}, \tilde{v} \ge 0$.

Ext pts: $\Rightarrow$ $\tilde{v}$ are convex weights for y

Primal + dual feasible $\Rightarrow$ Delaunay simplex containing y

Linear programming interpretation

$$\tilde{A} = \begin{bmatrix} (-x^{(1)})^T & 1 \\ (-x^{(2)})^T & 1 \\ \vdots & \vdots \\ (-x^{(n)})^T & 1 \end{bmatrix}, \ \tilde{b} = \begin{bmatrix} \|x^{(1)}\|_2^2 \\ \|x^{(2)}\|_2^2 \\ \vdots \\ \|x^{(n)}\|_2^2 \end{bmatrix}, \text{ and } \tilde{c} = \begin{bmatrix} -y \\ 1 \end{bmatrix}.$$

Primal prob: $\max_{\tilde{u}} \tilde{c}^T \tilde{u}$ such that $\tilde{A}\tilde{u} \leq \tilde{b}, \tilde{u}$ free.

Ext pts: $\tilde{u} = (-2\text{circumcenter}, \text{circumradius}^2 - \|\text{circumcenter}\|_2^2)$

Dual prob: $\min_{\tilde{v}} \tilde{b}^T \tilde{v}$ such that $\tilde{A}^T \tilde{v} = \tilde{c}, \tilde{v} \geq 0$.

Ext pts: $\Rightarrow \tilde{v}$ are convex weights for y

Primal + dual feasible $\Rightarrow$ Delaunay simplex containing y

LP basic solution in polynomial time is an open problem!

Extrapolation

What about extrapolation?

- Project y on to the convex hull of P
- Interpolate the projection (if the residual is small)
- Note: projection is a quadratic program (more expensive than an LP)

Let E be a $d \times n$ matrix whose columns are points in P, and let z be an extrapolation point (outside convex hull of P).

$$\xi^* = \arg\min_{\xi \in \mathbb{R}^n} \|E\xi - z\| \quad \text{subject to} \quad \xi \geq 0 \quad \text{and} \quad \sum_{i=1}^{n} \xi_i = 1.$$

Projection: $\hat{z} = E\xi^*$

DELAUNAYSPARSE Package

Standalone software package `DELAUNAYSPARSE`:

- Robust against degeneracy
- Runs in $\mathcal{O}(mnd^2\ell)$ time
- Parallel and serial implementations

Runtime (secs) for interpolating a single point ($m = 1$) with n pts in $\mathbb{R}^d$

	d				
n	2	8	32	64	128
250	0.005	0.013	0.150	3.404	27.078
500	0.021	0.042	0.325	6.479	59.511
1000	0.083	0.152	0.791	14.020	124.320
2000	0.344	0.583	2.230	28.984	242.066
4000	1.314	2.284	7.165	62.494	502.620
8000	5.580	9.027	26.210	151.177	905.711
16,000	22.086	35.725	109.448	386.596	2190.362
32,000	82.915	145.115	421.934	1097.060	5024.675

Chang, Watson, Lux, Butt, Cameron, and Hong. 2020. Algorithm 1012: DELAUNAYSPARSE: Interpolation via a sparse subset of the Delaunay triangulation in medium to high dimensions. ACM Trans. Math. Softw. 46(4).

The Delaunay Graph

- Delaunay graph of $P = DG(P)$
- Connect 2 vertices iff they are shared by a single Delaunay simplex
- Used for:
  - Neighbor structure in spatial data
  - Topological shape analysis
- There are at most $n(n-1)/2$ edges
- Current state-of-the-art implementation in CGAL computes $DG(P)$ from $DT(P)$ – scales well for large n, infeasible for $d \geq 10$

Getting the Delaunay Graph

- The number of connections in $DG(P)$ is upper bounded by $n(n-1)/2$
- Can recover $DG(P)$ by interpolating the midpoint between each pair of points in P
  - If the simplex containing the midpoint between $x^{(1)}$ and $x^{(2)}$ also contains both $x^{(1)}$ and $x^{(2)}$, then they are clearly connected
  - If not, then it certifies that they are not connected in *a* Delaunay triangulation (in case degenerate)
- Using `DELAUNAYSPARSE`, requires $\mathcal{O}(n^3 d^2 \ell)$ time — better than current state-of-the-art for d large, worse for n large

Implementation currently under review for publication.

Full proof/description in

T.H. Chang. Mathematical Software for Multiobjective Optimization Problems. Ph.D. Thesis, Virginia Tech, 2020.

Questions

Inference problems and high-dimensional modeling

DelaunaySparse algorithm for high-dimensional interpolation

Application for Computing the Delaunay Graph

- About the Delaunay Graph
- Algorithm using DELAUNAYSPARSE

This material is based upon work supported by the U.S. Dept. of Energy, Office of Science, Office of Advanced Scientific Computing Research, SciDAC program under contract number DE-AC02-06CH11357, and the Office of Science Graduate Student Research (SCGSR) program. The SCGSR program is administered by the Oak Ridge Institute for Science and Education (ORISE), which is managed by ORAU under contract number DE-SC0014664. All opinions in this paper are the authors' and do not necessarily reflect the policies and views of the DOE, ORAU, or ORISE.

This work was also supported in part by NSF Grants CNS-1565314 and CNS-1838271.