Toward interpretable machine learning via Delaunay interpolation

Algorithms and challenges

Tyler Chang

Argonne National Laboratory

LANS Seminar Series
July 12, 2023

Outlines

Inference problems and high-dimensional modeling

High-dimensional interpolation via Delaunay triangulations

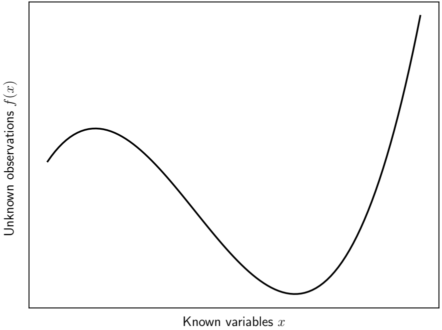
DelaunaySparse algorithm for high-dimensional interpolation

Preliminary Results and Future Work

Bonus Slides: Computing the Delaunay Graph

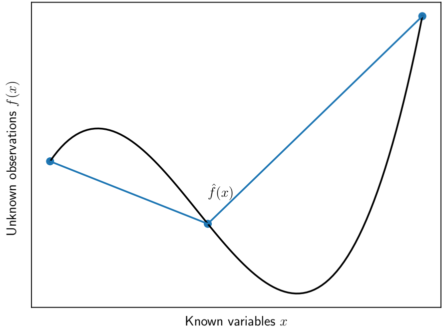
The fundamental machine learning problem

The fundamental machine learning problem



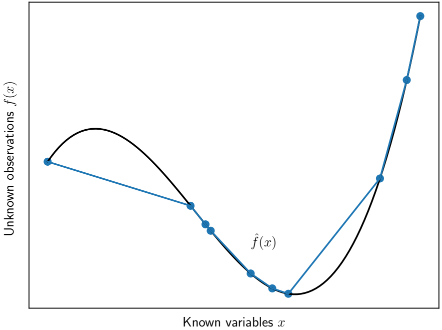
- Want to predict unknown $f(x)$ for observation x

The fundamental machine learning problem



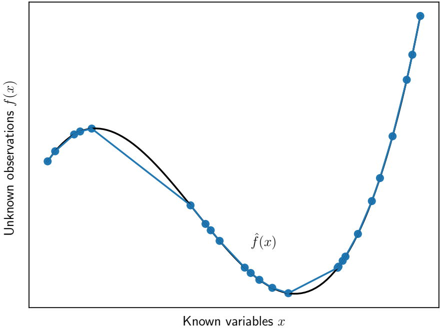
- Want to predict unknown $f(x)$ for observation x
- **ML**: *Learn* approximation $\hat{f} \sim f$ based on *training data* $\mathcal{X}$
- **NA**: fit an interpolant (piecewise-linear) to f on $\mathcal{X}$

The fundamental machine learning problem



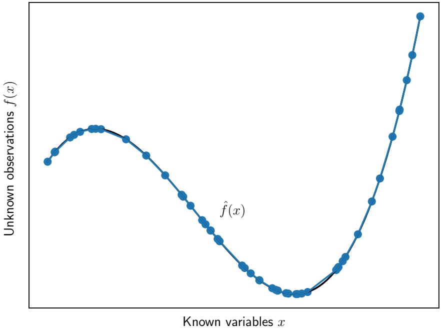
- Want to predict unknown $f(x)$ for observation x
- **ML**: *Learn* approximation $\hat{f} \sim f$ based on *training data* $\mathcal{X}$
- **NA**: fit an interpolant (piecewise-linear) to f on $\mathcal{X}$
- Both cases: more data $\Rightarrow$ better $\hat{f}$

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



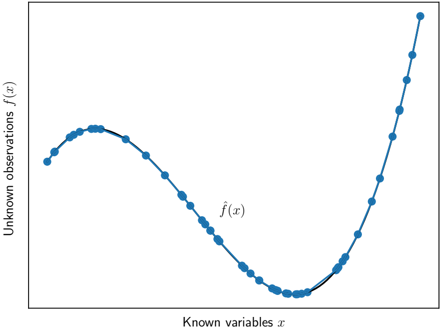
- Want to predict unknown $f(x)$ for observation x
- **ML**: *Learn* approximation $\hat{f} \sim f$ based on *training data* $\mathcal{X}$
- **NA**: fit an interpolant (piecewise-linear) to f on $\mathcal{X}$
- Both cases: more data $\Rightarrow$ better $\hat{f}$
- Real data not perfectly balanced $\Rightarrow$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

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



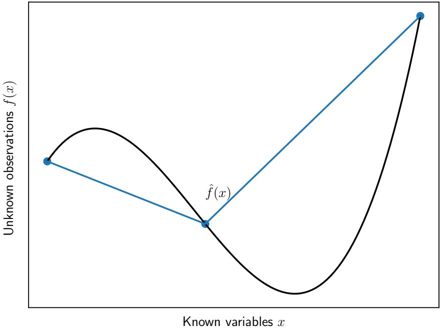
- h is a "mesh fineness" parameter $\sim$ distance between points in $\mathcal{X}$
- For irregular $\mathcal{X}$, h could be the distance from q to the nearest neighbor in $\mathcal{X}$
- Constants proportional to the Lip constant of ∇f

Some basic numerical analysis results

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

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

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



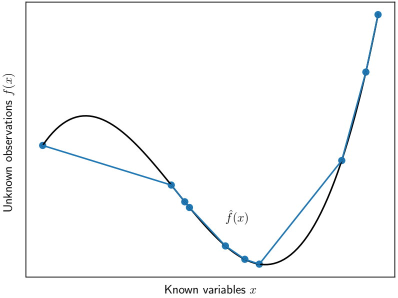
- h is a "mesh fineness" parameter $\sim$ distance between points in $\mathcal{X}$
- For irregular $\mathcal{X}$, h could be the distance from q to the nearest neighbor in $\mathcal{X}$
- Constants proportional to the Lip constant of ∇f

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

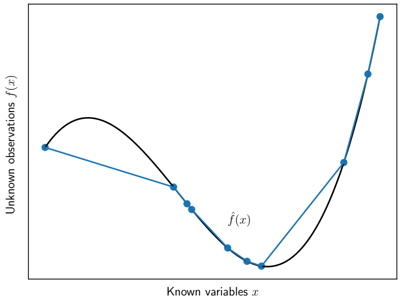
- Train a fully-connected multi-layer perceptron (MLP) using $\mathcal{X}$
- The most popular activation function is ReLU (piecewise linear)
- In modern ML, train as close to zero error as possible (interpolate)



- Piecewise linear interpolant ✓

Some basic deep learning

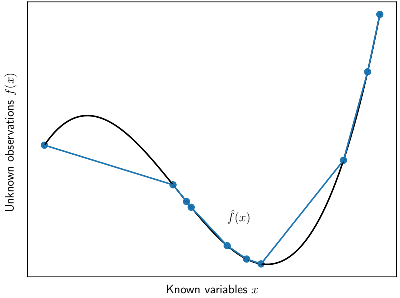
- Train a fully-connected multi-layer perceptron (MLP) using $\mathcal{X}$
- The most popular activation function is ReLU (piecewise linear)
- In modern ML, train as close to zero error as possible (interpolate)



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

Some basic deep learning

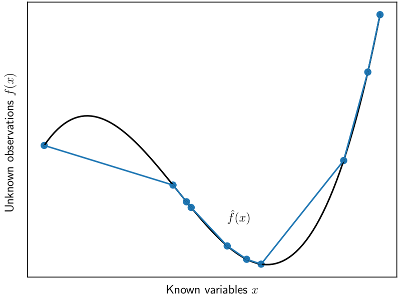
- Train a fully-connected multi-layer perceptron (MLP) using $\mathcal{X}$
- The most popular activation function is ReLU (piecewise linear)
- In modern ML, train as close to zero error as possible (interpolate)



- 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

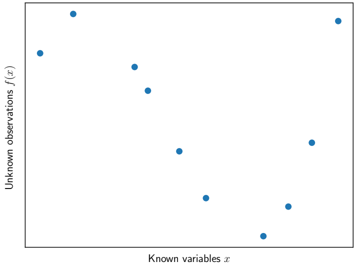
“There’s more to machine learning than function approximation”

Real machine learning

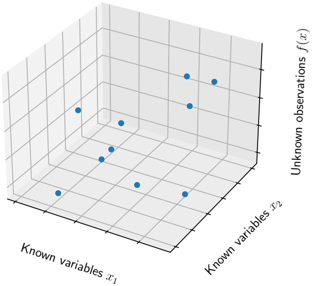
“There’s more to machine learning than function approximation”

- 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

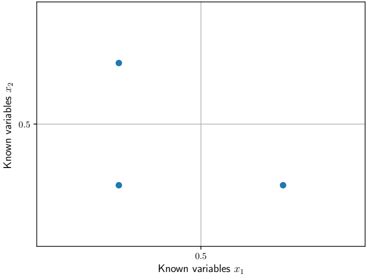


10 training points in 1D



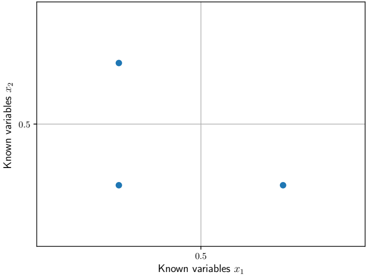
10 training points in 2D

The curse of ~~dimensionality~~ no data



Need data in all quadrants?

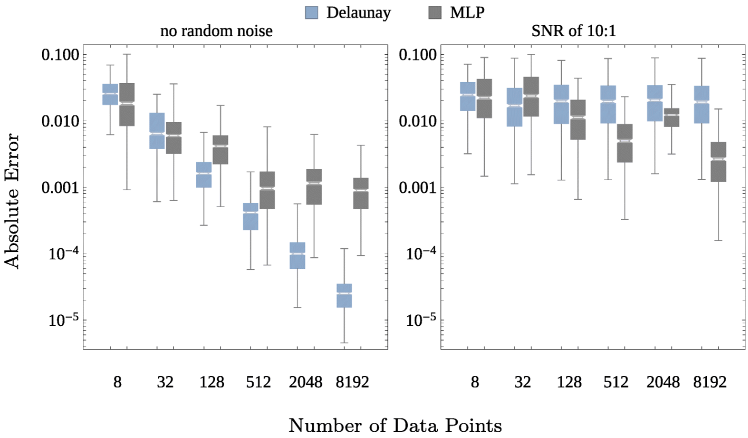
The curse of ~~dimensionality~~ no data



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

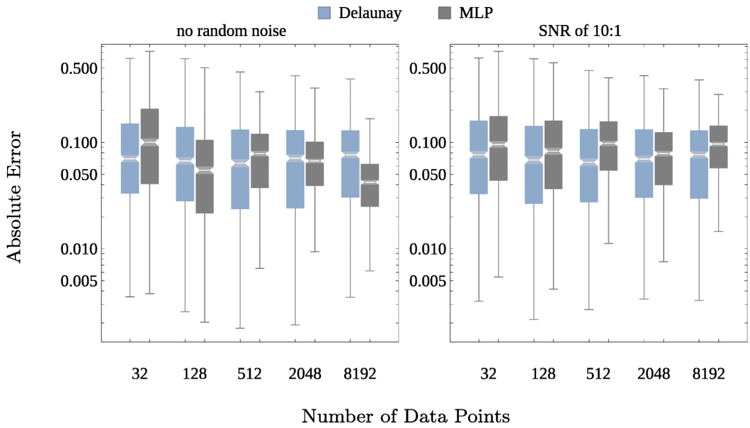
The blessing of dimensionality (no noise)



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 $[0,1]^d$

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

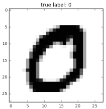
$$\mathbb{E}\left[\left(x_i - \frac{1}{2}\right)^2\right] = \int_0^1 \left(u - \frac{1}{2}\right)^2 du = \frac{1}{12}$$

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.*

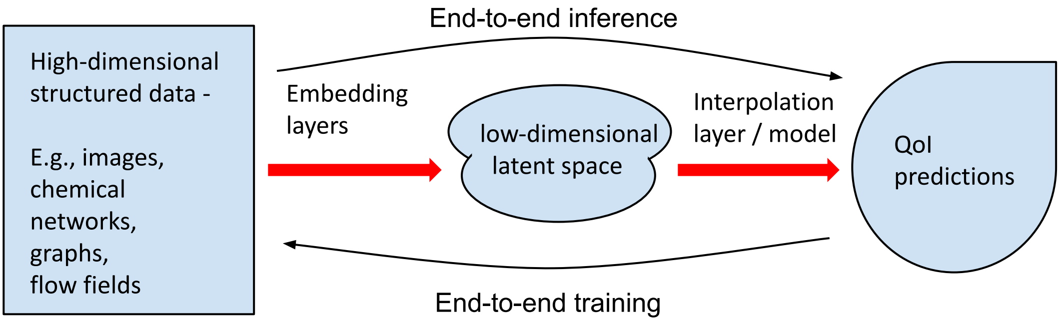
Hope in problem structure



$$\xrightarrow{f} \quad 0$$

28×28 pixels $\neq 784$ dimensions...

Modern deep learning pipeline



Tyler's hot takes on high-dimensional learning

Tyler's hot takes on high-dimensional learning

- "Big data" doesn't exist, all data is small (measure 0)

Tyler's hot takes on high-dimensional learning

- "Big data" doesn't exist, all data is small (measure 0)
- Overfitting is a myth, we can just interpolate

Tyler's hot takes on high-dimensional learning

- "Big data" doesn't exist, all data is small (measure 0)
- Overfitting is a myth, we can just interpolate
- SOA deep learning = *representation learning* + function approximation

Tyler's hot takes on high-dimensional learning

- "Big data" doesn't exist, all data is small (measure 0)
- Overfitting is a myth, we can just interpolate
- SOA deep learning = *representation learning* + function approximation
- The "function approximation" part is (piecewise linear) MLP regressors/classifiers

Tyler's hot takes on high-dimensional learning

- "Big data" doesn't exist, all data is small (measure 0)
- Overfitting is a myth, we can just interpolate
- SOA deep learning = *representation learning* + function approximation
- The "function approximation" part is (piecewise linear) MLP regressors/classifiers
- Interpolants + approximation theory can give you error bounds, model validation, interpretablility, and UQ

Outlines

Inference problems and high-dimensional modeling

High-dimensional interpolation via Delaunay triangulations

DelaunaySparse algorithm for high-dimensional interpolation

Preliminary Results and Future Work

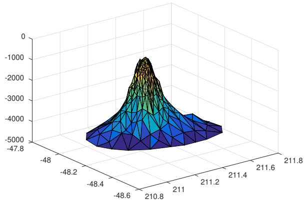
Bonus Slides: Computing the Delaunay Graph

Multidimensional piecewise linear interpolation

To define a piecewise linear interpolant in $\mathbb{R}^d$, **you need a simplicial mesh over $\mathcal{X}$**

$$\begin{bmatrix} x^{(i,0)} & \dots & x^{(i,d)} \\ 1 & \dots & 1 \end{bmatrix} w = \begin{bmatrix} q \\ 1 \end{bmatrix}$$

$$\hat{f}_{S^{(i)}}(q) = w^T \left(f(x^{(i,0)}), \dots, f(x^{(i,d)}) \right).$$

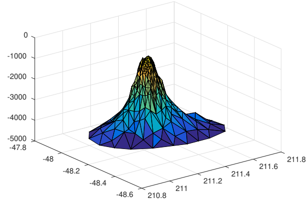


Multidimensional piecewise linear interpolation

To define a piecewise linear interpolant in $\mathbb{R}^d$, **you need a simplicial mesh over $\mathcal{X}$**

$$\begin{bmatrix} x^{(i,0)} & \dots & x^{(i,d)} \\ 1 & \dots & 1 \end{bmatrix} w = \begin{bmatrix} q \\ 1 \end{bmatrix}$$

$$\hat{f}_{S^{(i)}}(q) = w^T \left(f(x^{(i,0)}), \dots, f(x^{(i,d)}) \right).$$



Taylor bound at $x^{(i,0)}$ is:

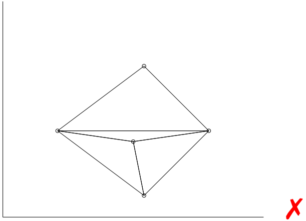
$$\left| f(q) - \hat{f}(q) \right| \leq \frac{\gamma \| q - x^{(i,0)} \|_2^2}{2} + \frac{\sqrt{d}\gamma k^2}{2\sigma_d} \| q - x^{(i,0)} \|_2.$$

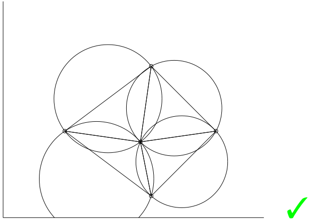
$k = \max_{x^{(i,j)} \neq x^{(i,1)}} \| x^{(i,1)} - x^{(i,j)} \|_2$,
γ is the Lip const of ∇f,
σ_d is the min singular val of Barycentric interpolation matrix

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

About Delaunay Triangulations

- The *Delaunay triangulation* ($DT(\mathcal{X})$) is the "optimal" unstructured simplicial mesh of $\mathcal{X}$
- **Defining property**: for all $S \in DT(\mathcal{X})$, the circumball $B^{(S)}$ satisfies $B^{(S)} \cap \mathcal{X} = \emptyset$.





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

Scalability Issues

- Meshes blow up exponentially in d
- Oweing to Klee (of Klee-Minty cube fame), the size of the $DT(\mathcal{X})$ is

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

  - For $d > 8$, this is impossible!

Scalability Issues

- Meshes blow up exponentially in d
- Oweing to Klee (of Klee-Minty cube fame), the size of the $DT(\mathcal{X})$ is

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

  - For $d > 8$, this is impossible!

Observation: For a given q, we only need vertices $(\{x^{(i,0)}, \ldots, x^{(i,d)}\})$ of $S \in DT(\mathcal{X})$ such that $q \in S$

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

Scalability Issues

- Meshes blow up exponentially in d
- Oweing to Klee (of Klee-Minty cube fame), the size of the $DT(\mathcal{X})$ is

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

  - For $d > 8$, this is impossible!

Observation: For a given q, we only need vertices $(\{x^{(i,0)}, \ldots, x^{(i,d)}\})$ of $S \in DT(\mathcal{X})$ such that $q \in S$

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

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

DelaunaySparse Algorithm outline

Algorithm to locate Delaunay simplex containing q:

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

Chang et al., A polynomial time algorithm for multivariate interpolation in arbitrary dimension via the Delaunay triangulation. *In 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

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

Linear programming interpretation

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

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

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

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 q

Linear programming interpretation

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

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 q

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

Linear programming interpretation

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

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 q

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

LP basic solution in polynomial time is an open problem!

Extrapolation

What about extrapolation?

- Project q on to the convex hull of $\mathcal{X}$
- Interpolate the projection (if the residual is small)
- Projection is a quadratic program

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

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

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

Chang et al., Remark on Algorithm 1012. *In preparation*.

DELAUNAYSPARSE Package

Standalone software package `DELAUNAYSPARSE`:

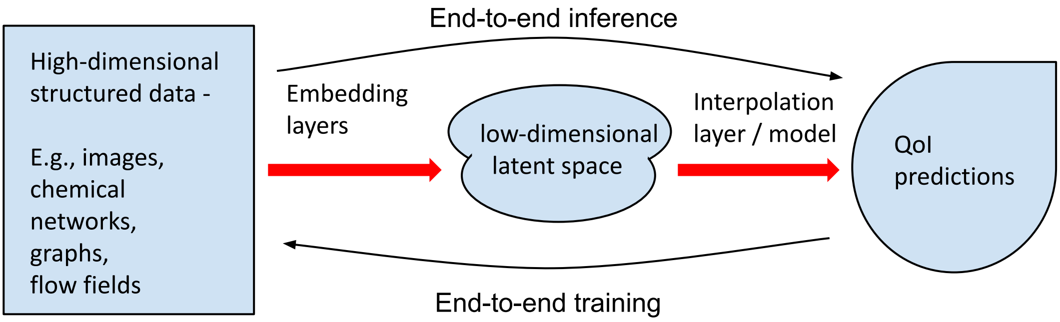
- Robust against degeneracy
- Runs in $\mathcal{O}(mnd^2\ell)$ time
- Parallel and serial implementations

Runtime (secs) for interpolating a single q

	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 et al., Algorithm 1012: DELAUNAYSPARSE. *ACM TOMS 46(4)*, Article No. 28 (2020).

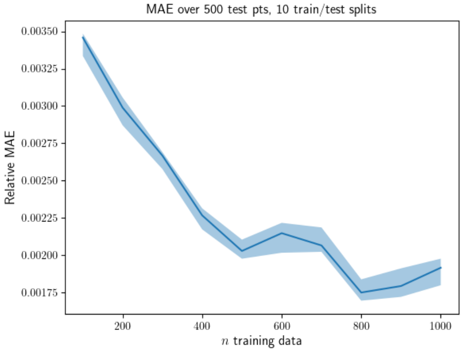
Recall...


End-to-end inference
High-dimensional structured data -
E.g., images, chemical networks, graphs, flow fields
Embedding layers
low-dimensional latent space
Interpolation layer / model
QoI predictions
End-to-end training

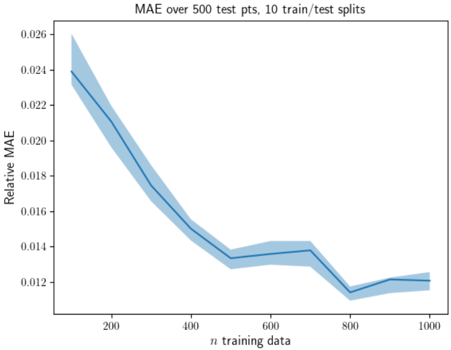
Early results on Airfoil Predictions

Thanks to Romit for providing airfoil prediction data, dimension reduced from 100,000+ down to **8D** via autoencoder

Delaunay interpolation in latent space



Lift predictions



Drag predictions

But what do you get?

But what do you get?

- **Interpretability:**
  - These $d+1$ training points were used in this prediction
  - Simplex is ill-conditioned, need more data in this direction

But what do you get?

- **Interpretability:**
  - These $d+1$ training points were used in this prediction
  - Simplex is ill-conditioned, need more data in this direction
- **Verifiability:**
  - Do the results agree with the error bounds?
  - See preprint on Delaunay Diagnostic from LLNL

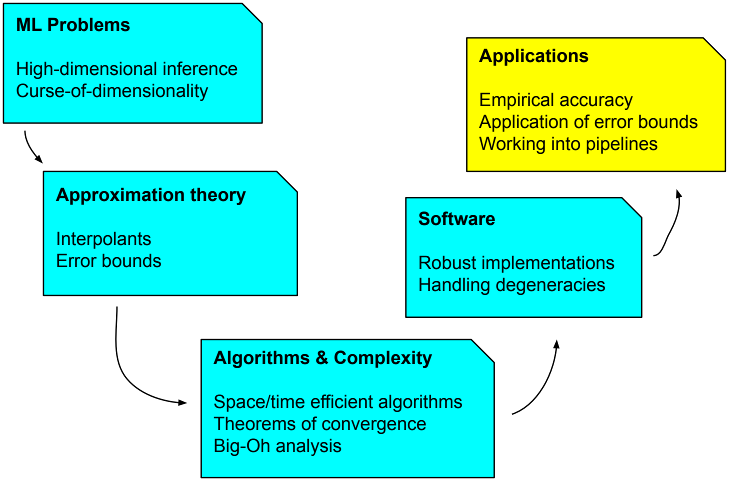
But what do you get?

- **Interpretability:**
  - These $d + 1$ training points were used in this prediction
  - Simplex is ill-conditioned, need more data in this direction
- **Verifiability:**
  - Do the results agree with the error bounds?
  - See preprint on Delaunay Diagnostic from LLNL
- **Error bounds and UQ:**
  - Coming soon...

Gillette and Kur, Data-driven geometric scale detection via Delaunay interpolation. *arXiv preprint 2203.05685.*

Chang, Gillette, and Maulik, *in preparation*.

Next steps



ML Problems
High-dimensional inference
Curse-of-dimensionality
Approximation theory
Interpolants
Error bounds
Algorithms & Complexity
Space/time efficient algorithms
Theorems of convergence
Big-Oh analysis
Software
Robust implementations
Handling degeneracies
Applications
Empirical accuracy
Application of error bounds
Working into pipelines

Next steps

ML Problems

High-dimensional inference
Curse-of-dimensionality

Approximation theory

Interpolants
Error bounds

Algorithms & Complexity

Space/time efficient algorithms
Theorems of convergence
Big-Oh analysis

Software

Robust implementations
Handling degeneracies

Applications

Empirical accuracy
Application of error bounds
Working into pipelines

Scalability

Large-scale linear programming

Next steps

ML Problems

High-dimensional inference
Curse-of-dimensionality

Approximation theory

Interpolants
Error bounds

Algorithms & Complexity

Space/time efficient algorithms
Theorems of convergence
Big-Oh analysis

Software

Robust implementations
Handling degeneracies

Applications

Empirical accuracy
Application of error bounds
Working into pipelines

Scalability

Large-scale linear programming

Usability

Differentiable optimization

Next steps

ML Problems

High-dimensional inference
Curse-of-dimensionality

Approximation theory

Interpolants
Error bounds

Algorithms & Complexity

Space/time efficient algorithms
Theorems of convergence
Big-Oh analysis

Software

Robust implementations
Handling degeneracies

Applications

Empirical accuracy
Application of error bounds
Working into pipelines

Scalability

Large-scale linear programming

Usability

Differentiable optimization

Flexibility

Arbitrary distance metric

Acknowledgements

This work was supported in part by the VarSys project: NSF Grants CNS-1565314 and CNS-1838271.

This work was also supported in part by FASTMath Institute: U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, SciDAC program under contract number DE-AC02-06CH11357.

Questions

Inference problems and high-dimensional modeling

High-dimensional interpolation via Delaunay triangulations

DelaunaySparse algorithm for high-dimensional interpolation

Preliminary Results and Future Work

Bonus Slides: Computing the Delaunay Graph

The Delaunay Graph

- Delaunay graph of $\mathcal{X} = DG(\mathcal{X})$
- 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 $DG(\mathcal{X})$ from $DT(\mathcal{X})$ – scales well for large n, infeasible for $d \geq 10$

Getting the Delaunay Graph

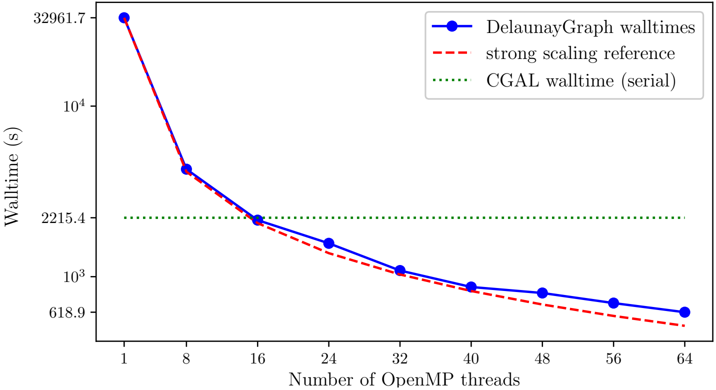
- The number of connections in $DG(\mathcal{X})$ is upper bounded by $n(n-1)/2$
- Can recover $DG(\mathcal{X})$ by interpolating the midpoint between each pair of points in $\mathcal{X}$
  - If the simplex containing the midpoint between $x^{(1)}$ and $x^{(2)}$ also contains both $x^{(1)}$ and $x^{(2)}$, 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

Full proof in my PhD dissertation:

Chang, *Mathematical Software for Multiobjective Optimization Problems*. PhD dissertation, Virginia Tech (2020).

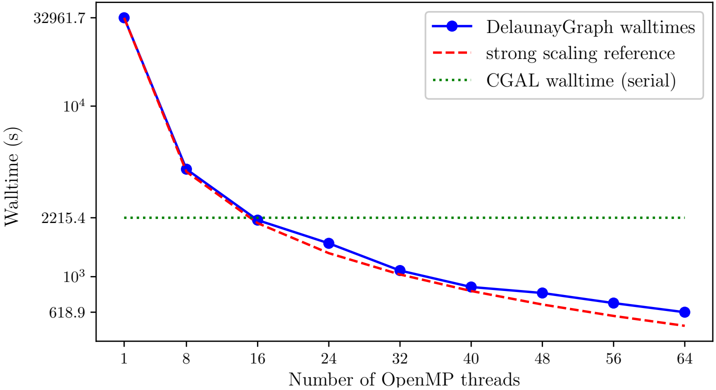
Parallel scaling

For $d > 8$, CGAL crashes with out-of-memory error



Parallel scaling

For $d > 8$, CGAL crashes with out-of-memory error



Paper has been rejected due to lack of real-world data