Nonlinear PDE continuum limits in data science and machine learning

Jeff Calder

School of Mathematics
University of Minnesota

University of Wisconsin PDE & GA seminar
Monday, April 9, 2018

This research was supported by NSF-DMS grants 0914567, 1500829 and 1713691.
Outline

1. Nondominated sorting
2. Convex hull peeling
3. Semi-supervised learning
4. References
Outline

1. Nondominated sorting
2. Convex hull peeling
3. Semi-supervised learning
4. References
Motivating example: Google Goggles
Motivating example: Google Goggles

Figure: Query image
Motivating example: Google Goggles

**Figure:** Query image

**Figure:** Retrieved images
Multi-query image retrieval

**Problem:** Find images in a dataset $S$ that are similar to multiple query images.

**Pareto method:** “Solve” the multi-objective optimization problem

$$\arg \min_{I \in S} (\text{dist}(I, Q_1), \ldots, \text{dist}(I, Q_d)).$$
Multi-query image retrieval

**Problem:** Find images in a dataset $S$ that are similar to multiple query images.

**Pareto method:** “Solve” the multi-objective optimization problem

$$\underset{I \in S}{\arg \min} (\text{dist}(I, Q_1), \ldots, \text{dist}(I, Q_d)).$$

**Pareto points:**
Multi-objective optimization

How do we solve the multi-objective optimization problem

\[
\arg \min_{I \in S} (f_1(I), \ldots, f_d(I))?
\]

Basic approach:

1. Choose some weights \(\alpha_i \in [0, 1]\) with \(\sum \alpha_i = 1\) and define \(f_\alpha(I) = \alpha_1 f_1(I) + \alpha_2 f_2(I) + \cdots + \alpha_d f_d(I)\).

2. Solve the scalarized optimization problem \(\arg \min_{I \in S} f_\alpha(I)\).

Problems:

1. Difficult to choose weights
2. Ignores relevant solutions
Multi-objective optimization

How do we solve the multi-objective optimization problem

$$\arg \min_{I \in S} (f_1(I), \ldots, f_d(I))$$

Basic approach:

1. Choose some weights $\alpha_i \in [0, 1]$ with $\sum_{i=1}^d \alpha_i = 1$ and define

$$f_\alpha(I) = \alpha_1 f_1(I) + \alpha_2 f_2(I) + \cdots + \alpha_d f_d(I).$$
Multi-objective optimization

How do we solve the multi-objective optimization problem

$$\arg\min_{I \in S} (f_1(I), \ldots, f_d(I))?$$

Basic approach:

1. Choose some weights $\alpha_i \in [0, 1]$ with $\sum_{i=1}^{d} \alpha_i = 1$ and define

$$f_\alpha(I) = \alpha_1 f_1(I) + \alpha_2 f_2(I) + \cdots + \alpha_d f_d(I).$$

2. Solve the scalarized optimization problem

$$\arg\min_{I \in S} f_\alpha(I).$$
Multi-objective optimization

How do we solve the multi-objective optimization problem

$$\arg\min_{I \in S} (f_1(I), \ldots, f_d(I))?$$

Basic approach:

1. Choose some weights $\alpha_i \in [0, 1]$ with $\sum_{i=1}^d \alpha_i = 1$ and define

   $$f_\alpha(I) = \alpha_1 f_1(I) + \alpha_2 f_2(I) + \cdots + \alpha_d f_d(I).$$

2. Solve the scalarized optimization problem

   $$\arg\min_{I \in S} f_\alpha(I).$$

Problems:

1. Difficult to choose weights
2. Ignores relevant solutions
Basic approach
Basic approach
Basic approach

\[ \alpha = 0.1 \]

\[ \alpha = 0.3 \]
Basic approach
Nondominated solutions
Nondominated solutions
Nondominated solutions
Nondominated solutions
Nondominated solutions
Nondominated solutions
Nondominated solutions
Multi-query image retrieval

First Pareto front:

Query 1

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
Query 2

Nondominated sorting

Let $X_1, \ldots, X_n$ be points in $\mathbb{R}^d$ and set $S = \{X_1, \ldots, X_n\}$.

Define the partial order

$$x \leq y \iff x_i \leq y_i \text{ for all } i \in \{1, \ldots, d\}.$$
Nondominated sorting

Let $X_1, \ldots, X_n$ be points in $\mathbb{R}^d$ and set $S = \{X_1, \ldots, X_n\}$.

Define the partial order

$$x \leq y \iff x_i \leq y_i \text{ for all } i \in \{1, \ldots, d\}.$$  

**Definition**

Nondominated sorting is the process of arranging $S$ into layers $\mathcal{F}_1, \mathcal{F}_2, \mathcal{F}_3, \ldots$, defined by

$$\mathcal{F}_1 = \text{Minimal elements of } S,$$

$$\mathcal{F}_k = \text{Minimal elements of } S \setminus (\mathcal{F}_1 \cup \cdots \cup \mathcal{F}_{k-1}).$$
Applications

Multi-objective optimization

- Genetic algorithms [Deb et al., 2002]
- Gene selection and ranking [Hero, 2003]
- Database systems [Papadias et al., 2005]
- Anomaly detection [Hsiao et al., 2012]
- Image retrieval [Hsiao et al., 2015]
Applications

Multi-objective optimization
- Genetic algorithms [Deb et al., 2002]
- Gene selection and ranking [Hero, 2003]
- Database systems [Papadias et al., 2005]
- Anomaly detection [Hsiao et al., 2012]
- Image retrieval [Hsiao et al., 2015]

Combinatorics and probability
- Longest monotone subsequences [Ulam, 1961]
- Longest chain in Euclidean space [Hammersley, 1972]
- Patience sorting [Aldous and Diaconis, 1999]
- Young Tableaux [Viennot, 1984]
- Graph theory [Lou and Sarrafzadeh, 1993]
- Polynuclear growth (crystals) [Prähofer and Spohn, 2000]
Applications

**Multi-objective optimization**
- Genetic algorithms [Deb et al., 2002]
- Gene selection and ranking [Hero, 2003]
- Database systems [Papadias et al., 2005]
- Anomaly detection [Hsiao et al., 2012]
- Image retrieval [Hsiao et al., 2015]

**Combinatorics and probability**
- Longest monotone subsequences [Ulam, 1961]
- Longest chain in Euclidean space [Hammersley, 1972]
- Patience sorting [Aldous and Diaconis, 1999]
- Young Tableaux [Viennot, 1984]
- Graph theory [Lou and Sarrafzadeh, 1993]
- Polynuclear growth (crystals) [Prähofer and Spohn, 2000]

**Other applications**
- Molecular biology [Pevzner, 2000]
- Integrated circuit design [Adhar, 2007]
Demo: 50 Random samples
Demo: Uniform distribution

\[ n = 10^2 \text{ points} \]
Demo: Uniform distribution

\[ n = 10^3 \text{ points} \]
Demo: Uniform distribution

\[ n = 10^4 \text{ points} \]
Demo: Uniform distribution

\[ n = 10^5 \text{ points} \]
Demo: Uniform distribution

\[ n = 10^6 \text{ points} \]
Demo: Gaussian distribution

\[ n = 10^2 \text{ points} \]
Demo: Gaussian distribution

\[ n = 10^3 \text{ points} \]
Demo: Gaussian distribution

\[ n = 10^4 \text{ points} \]
Demo: Gaussian distribution

\[ n = 10^5 \text{ points} \]
Demo: Gaussian distribution

\[ n = 10^6 \text{ points} \]
Demo: Uniform distribution on \([0, 1]^2 \setminus [0, 0.5]^2\)

\[ n = 10^2 \text{ points} \]
Demo: Uniform distribution on $[0, 1]^2 \setminus [0, 0.5]^2$

$n = 10^3$ points
Demo: Uniform distribution on $[0, 1]^2 \setminus [0, 0.5]^2$

$n = 10^4$ points
Demo: Uniform distribution on \([0, 1]^2 \setminus [0, 0.5]^2\)

\[ n = 10^5 \text{ points} \]
Demo: Uniform distribution on $[0, 1]^2 \setminus [0, 0.5]^2$

$n = 10^6$ points
A PDE continuum limit for nondominated sorting

Let $X_1, \ldots, X_n$ be i.i.d. random variables in $[0, \infty)^d$ with continuous density $f$. 
A PDE continuum limit for nondominated sorting

Let $X_1, \ldots, X_n$ be i.i.d. random variables in $[0, \infty)^d$ with continuous density $f$.

Let $U_n : \mathbb{R}^d \to \mathbb{N}_0$ be the function that ‘counts’ the layers $\mathcal{F}_1, \mathcal{F}_2, \ldots$
Theorem (Calder, Esedoḡlu, Hero, 2014)

There exists a universal constant \( c_d > 0 \) such that with probability one

\[
n^{-\frac{1}{d}} U_n \longrightarrow c_d u \quad \text{locally uniformly as } n \to \infty
\]

where \( u \in C^{0, \frac{1}{d}}([0, \infty)^d) \) is the unique nondecreasing (\( u_{x_i} \geq 0 \)) viscosity solution of

\[
(P) \quad \begin{cases} 
  u_{x_1} \cdots u_{x_d} = f & \text{in } \mathbb{R}_+^d := (0, \infty)^d \\
  u = 0 & \text{on } \partial \mathbb{R}_+^d.
\end{cases}
\]
Theorem (Calder, Esedoḡlu, Hero, 2014)

There exists a universal constant \( c_d > 0 \) such that with probability one

\[
n^{-\frac{1}{d}} U_n \longrightarrow c_d u \quad \text{locally uniformly as } n \to \infty
\]

where \( u \in C^{0, \frac{1}{d}}([0, \infty)^d) \) is the unique nondecreasing \((u_{x_i} \geq 0)\) viscosity solution of

\[
(P) \begin{cases} 
 u_{x_1} \cdots u_{x_d} &= f \quad \text{in } \mathbb{R}_+^d := (0, \infty)^d \\
 u &= 0 \quad \text{on } \partial \mathbb{R}_+^d.
\end{cases}
\]


**Current work:** Rate of convergence (Brendan Cook)
Demo: \( f = 1 - \chi_{[0,0.5]^2} \)
Demo: Multimodal $f$
Quick “proof”

Let $X_1, \ldots, X_n$ be \textit{i.i.d.} random variables in $[0, \infty)^d$ with continuous density $f$. 
Quick “proof”

Let $X_1, \ldots, X_n$ be i.i.d. random variables in $[0, \infty)^d$ with continuous density $f$.

Let $U_n : \mathbb{R}^d \to \mathbb{N}_0$ be the function that ‘counts’ the layers $\mathcal{F}_1, \mathcal{F}_2, \ldots$
Quick “proof”

Let’s suppose that \( n^{-\alpha} U_n \to u \in C^1 \) as \( n \to \infty \) for some \( \alpha \in [0, 1] \).
Quick “proof”

Let’s suppose that $n^{-\alpha} U_n \longrightarrow u \in C^1$ as $n \to \infty$ for some $\alpha \in [0, 1]$. 

![Diagram](image)

\begin{align*}
\ell_1 &= \frac{\langle Du, v \rangle}{u_{x_1}} \\
\ell_2 &= \frac{\langle Du, v \rangle}{u_{x_2}}
\end{align*}

If $\alpha = 1$, or $\alpha = 1/d$, then $u_{x_1} \cdots u_{x_d} = f(x)$. 

Calder (UofM)
PDE continuum limitsWisconsin PDE&GA 21 / 83
Quick “proof”

Let’s suppose that $n^{-\alpha}U_n \rightarrow u \in C^1$ as $n \rightarrow \infty$ for some $\alpha \in [0, 1]$.

\[ \langle Du, v \rangle \approx u(x + v) - u(x) \]

\[ \ell_1 = \frac{\langle Du, v \rangle}{u_{x_1}} \]
\[ \ell_2 = \frac{\langle Du, v \rangle}{u_{x_2}} \]
Quick “proof”

Let’s suppose that $n^{-\alpha}U_n \rightarrow u \in C^1$ as $n \rightarrow \infty$ for some $\alpha \in [0, 1]$.

\[
\langle Du, v \rangle \approx u(x + v) - u(x) \\
\approx (\# \text{ fronts in } A)n^{-\alpha}
\]
Quick “proof”

Let’s suppose that $n^{-\alpha} U_n \rightarrow u \in C^1$ as $n \rightarrow \infty$ for some $\alpha \in [0, 1]$.

\[
\langle Du, v \rangle \approx u(x + v) - u(x)
\]
\[
\approx (# \text{ fronts in } A)n^{-\alpha}
\]
\[
\approx (# \text{ samples in } A)^\alpha n^{-\alpha}
\]
Quick “proof”

Let’s suppose that $n^{-\alpha} U_n \to u \in C^1$ as $n \to \infty$ for some $\alpha \in [0, 1]$.

\[
\langle Du, v \rangle \approx u(x + v) - u(x)
\]
\[
\approx (# \text{ fronts in } A)n^{-\alpha}
\]
\[
\approx (# \text{ samples in } A)^{\alpha} n^{-\alpha}
\]

\[
\ell_1 = \frac{\langle Du, v \rangle}{u_{x_1}}
\]
\[
\ell_2 = \frac{\langle Du, v \rangle}{u_{x_2}}
\]

$u = u(x)$
Quick “proof”

Let's suppose that $n^{-\alpha} U_n \rightarrow u \in C^1$ as $n \rightarrow \infty$ for some $\alpha \in [0, 1]$.

\[
\langle Du, v \rangle \approx u(x + v) - u(x) \\
\approx (\# \text{ fronts in } A)n^{-\alpha} \\
\approx (\# \text{ samples in } A)^\alpha n^{-\alpha} \\
\approx (n|A|f(x))^\alpha n^{-\alpha} \\
\approx |A|^\alpha f(x)^\alpha.
\]
Quick “proof”

Let’s suppose that \( n^{-\alpha} U_n \longrightarrow u \in C^1 \) as \( n \to \infty \) for some \( \alpha \in [0, 1] \).

\[
\langle Du, v \rangle \approx u(x + v) - u(x)
\]

\[
\approx (\# \text{ fronts in } A)n^{-\alpha}
\]

\[
\approx (\# \text{ samples in } A)^\alpha n^{-\alpha}
\]

\[
\approx (n|A|f(x))^\alpha n^{-\alpha}
\]

\[
\approx |A|^{\alpha} f(x)^\alpha.
\]

Use \( |A| \approx \frac{\langle Du, v \rangle d}{u_{x_1} \cdots u_{x_d}} \)
Quick “proof”

Let’s suppose that $n^{-\alpha} U_n \longrightarrow u \in C^1$ as $n \to \infty$ for some $\alpha \in [0, 1]$.

\[ \langle Du, v \rangle \approx u(x + v) - u(x) \]
\[ \approx (\# \text{ fronts in } A) n^{-\alpha} \]
\[ \approx (\# \text{ samples in } A)^\alpha n^{-\alpha} \]
\[ \approx (n |A| f(x))^{\alpha} n^{-\alpha} \]
\[ \approx |A|^\alpha f(x)^\alpha. \]

Use $|A| \approx \frac{\langle Du, v \rangle^d}{u_{x_1} \cdots u_{x_d}}$ to find

\[ \langle Du, v \rangle \approx \left( \frac{f(x)}{u_{x_1} \cdots u_{x_d}} \right)^\alpha \langle Du, v \rangle^{\alpha d} \]
Quick “proof”

Let’s suppose that \( n^{-\alpha} U_n \to u \in C^1 \) as \( n \to \infty \) for some \( \alpha \in [0, 1] \).

\[
\langle Du, v \rangle \approx u(x + v) - u(x) \\
\approx (\# \text{ fronts in } A) n^{-\alpha} \\
\approx (\# \text{ samples in } A)^\alpha n^{-\alpha} \\
\approx (n|A| f(x))^{\alpha} n^{-\alpha} \\
\approx |A|^\alpha f(x)^\alpha.
\]

Use \( |A| \approx \frac{(Du,v)^d}{u_{x_1} \cdots u_{x_d}} \) to find

\[
\langle Du, v \rangle \approx \left( \frac{f(x)}{u_{x_1} \cdots u_{x_d}} \right)^\alpha \langle Du, v \rangle^{\alpha d}
\]

If \( \alpha d = 1 \), or \( \alpha = 1/d \), then

\[
u_{x_1} \cdots u_{x_d} = f
\]
Ordering within each front

Let $X_1, \ldots, X_n$ be i.i.d. random variables with density $f$ on $[0, 1]^2$. Define

$$V_n(X_i) = \text{Index of } X_i \text{ within its Pareto front}.$$
Demo: Uniform distribution on $[0, 1]^2$

\begin{align*}
\text{\textbf{(T)}} \quad \langle Dv, D^\perp u \rangle &= f \quad \text{in } (0, 1)^2, \\
v &= 0 \quad \text{on } (0, 1) \times \{ x_2 = 1 \}.
\end{align*}

\begin{align*}
\text{\textbf{(T')}} \quad \langle Dw, vD^\perp u \rangle &= wf \quad \text{in } (0, 1)^2, \\
w &= 1 \quad \text{on } \{ x_1 = 1 \} \times (0, 1).
\end{align*}
Fast approximate sorting

Algorithm (PDE-based Ranking)

1. Select \( k \) points from \( X_1, \ldots, X_n \) at random. Call them \( Y_1, \ldots, Y_k \). (\( k \ll n \))

Notes:
- Total complexity is \( O(k + h^{-d} + n) \).
- If we fix \( k \) and \( h \), independent of \( n \), then Steps 1-3 have \( O(1) \) complexity.

Fast approximate sorting

Algorithm (PDE-based Ranking)

1. Select $k$ points from $X_1, \ldots, X_n$ at random. Call them $Y_1, \ldots, Y_k$. ($k \ll n$)
2. Estimate $f$ with a histogram

$$\hat{f}(x) = \frac{1}{kh^d} \cdot \#\left\{ Y_i : Y_i \in [x, x + h1] \right\}.$$
Fast approximate sorting

Algorithm (PDE-based Ranking)

1. Select $k$ points from $X_1, \ldots, X_n$ at random. Call them $Y_1, \ldots, Y_k$. ($k \ll n$)

2. Estimate $f$ with a histogram

$$
\hat{f}(x) = \frac{1}{kh^d} \cdot \#\{ Y_i : Y_i \in [x, x + h1] \}.
$$

3. Compute the numerical solution $\hat{U}_h$ of the PDE.
Fast approximate sorting

Algorithm (PDE-based Ranking)

1. Select $k$ points from $X_1, \ldots, X_n$ at random. Call them $Y_1, \ldots, Y_k$. ($k \ll n$)
2. Estimate $f$ with a histogram

\[
\hat{f}(x) = \frac{1}{kh^d} \cdot \# \left\{ Y_i : Y_i \in [x, x + h1] \right\}.
\]

3. Compute the numerical solution $\hat{U}_h$ of the PDE.
4. Evaluate $\hat{U}_h(X_i)$ for $i = 1, \ldots, n$ via interpolation.

Notes:
- Total complexity is $O(k + h^{-d} + n)$.
- If we fix $k$ and $h$, independent of $n$, then Steps 1-3 have $O(1)$ complexity.

Fast approximate sorting

Algorithm (PDE-based Ranking)

1. **Select $k$ points from $X_1, \ldots, X_n$ at random. Call them $Y_1, \ldots, Y_k$. ($k \ll n$)**

2. **Estimate $f$ with a histogram**

   $$
   \hat{f}(x) = \frac{1}{kh^d} \cdot \# \{ Y_i : Y_i \in [x, x+h1] \}.
   $$

3. **Compute the numerical solution $\hat{U}_h$ of the PDE.**

4. **Evaluate $\hat{U}_h(X_i)$ for $i = 1, \ldots, n$ via interpolation.**

**Notes:**

- Total complexity is $O(k + h^{-d} + n)$.
- If we fix $k$ and $h$, independent of $n$, then Steps 1-3 have $O(1)$ complexity.

CPU Time (C/C++)

- # Subsamples = $k = 10^7$, Grid for solving PDE = $250 \times 250$.
- $O(n \log n)$ non-dominated sorting of [Felsner and Wernisch, 1999].
Application in anomaly detection

(a) Example trajectories

(b) \(5 \times 10^5\) Pareto points

Results

Anomaly detection with PDE-based ranking: Reduces complexity from $O(n^2)$ to $O(n)$.

Results

Anomaly detection for streaming data:

Examples of detected anomalies... with classifications using the new transport equations.

Outline

1. Nondominated sorting
2. Convex hull peeling
3. Semi-supervised learning
4. References
Convex hull peeling

**Question:** How to define ‘median’ in dimensions $d \geq 2$?
Convex hull peeling

Question: How to define ‘median’ in dimensions $d \geq 2$?

Barnett [Barnett, 1976]: Convex hull peeling
Convex hull peeling

**Question:** How to define ‘median’ in dimensions $d \geq 2$?

Barnett [Barnett, 1976]: Convex hull peeling
Convex hull peeling

**Question:** How to define ‘median’ in dimensions $d \geq 2$?

Barnett [Barnett, 1976]: Convex hull peeling
Convex hull peeling

**Question:** How to define ‘median’ in dimensions $d \geq 2$?

Barnett [Barnett, 1976]: Convex hull peeling
Convex hull peeling

**Question:** How to define ‘median’ in dimensions $d \geq 2$?

Barnett [Barnett, 1976]: Convex hull peeling
Convex hull peeling

**Question:** How to define ‘median’ in dimensions $d \geq 2$?

Barnett [Barnett, 1976]: Convex hull peeling
Convex hull peeling

**Question:** How to define ‘median’ in dimensions $d \geq 2$?

Barnett [Barnett, 1976]: Convex hull peeling
Question: How to define ‘median’ in dimensions $d \geq 2$?

Barnett [Barnett, 1976]: Convex hull peeling

Convex hull peeling median := Centroid of final layer
MNIST handwritten digit dataset
Convex hull peeling

Let $X_1, \ldots, X_n$ be points in $\mathbb{R}^d$ and set $S = \{X_1, \ldots, X_n\}$. 

Convex hull peeling

Let $X_1, \ldots, X_n$ be points in $\mathbb{R}^d$ and set $S = \{X_1, \ldots, X_n\}$.

**Definition**

Convex hull peeling is the process of arranging $S$ into convex layers $C_1, C_2, C_3, \ldots$, defined by

$$C_1 = \text{Vertices of convex hull of } S,$$

$$C_k = \text{Vertices of convex hull of } S \setminus (C_1 \cup \cdots \cup C_{k-1}).$$
Convex hull peeling

Applications:

- Robust statistics, machine learning, etc.
  - [Rousseeuw and Struyf, 2004], [Donoho and Gasko, 1992], [Hodge and Austin, 2004].
Convex hull peeling

Applications:

- Robust statistics, machine learning, etc.
  - [Rousseeuw and Struyf, 2004], [Donoho and Gasko, 1992], [Hodge and Austin, 2004].

- Matching of deformed pointclouds [Suk and Flusser, 1999].
Convex hull peeling

Applications:

- Robust statistics, machine learning, etc.
  - [Rousseeuw and Struyf, 2004], [Donoho and Gasko, 1992], [Hodge and Austin, 2004].
- Matching of deformed pointclouds [Suk and Flusser, 1999].
- Fingerprint matching [Poulos et al., 2005].
Convex hull peeling: Demo - Uniform distribution

\[ n = 10^2 \text{ points} \]
Convex hull peeling: Demo - Uniform distribution

$n = 10^3$ points
Convex hull peeling: Demo - Uniform distribution

\[ n = 10^4 \text{ points} \]
Convex hull peeling: Demo - Uniform distribution

\[ n = 10^5 \text{ points} \]
Convex hull peeling: Demo - Triangle distribution

$n = 10^2$ points
Convex hull peeling: Demo - Triangle distribution

\[ n = 10^3 \text{ points} \]
Convex hull peeling: Demo - Triangle distribution

$n = 10^4$ points
Convex hull peeling: Demo - Triangle distribution

\[ n = 10^5 \text{ points} \]
Convex hull peeling: Demo - Gaussian distribution

$n = 10^2$ points
Convex hull peeling: Demo - Gaussian distribution

\[ n = 10^3 \text{ points} \]
Convex hull peeling: Demo - Gaussian distribution

\[ n = 10^4 \text{ points} \]
Convex hull peeling: Demo - Gaussian distribution

\[ n = 10^5 \text{ points} \]
A two player game for convex hull peeling

**Players:** Paul and Carol  
**State space:** $\mathcal{X} := \{X_1, \ldots, X_n\}$
A two player game for convex hull peeling

**Players:** Paul and Carol

**State space:** \( \mathcal{X} := \{X_1, \ldots, X_n\} \)

**Paul’s goal:** Reach vertex of convex hull

**Carol’s goal:** Obstruct Paul
A two player game for convex hull peeling

**Players:** Paul and Carol

**State space:** $\mathcal{X} := \{X_1, \ldots, X_n\}$

**Paul’s goal:** Reach vertex of convex hull

**Carol’s goal:** Obstruct Paul

**Rules of the game:** Token starts at $x^0 \in \mathcal{X}$ and is moved according to:

1. Paul picks $v \in S^{d-1}$
2. Carol moves token to any $x^{k+1} \in \mathcal{X}$ satisfying

$$ (x^{k+1} - x^k) \cdot v > 0. $$
A two player game for convex hull peeling

Players: Paul and Carol
State space: $\mathcal{X} := \{X_1, \ldots, X_n\}$

Paul’s goal: Reach vertex of convex hull
Carol’s goal: Obstruct Paul

Rules of the game: Token starts at $x^0 \in \mathcal{X}$ and is moved according to:

1. Paul picks $v \in S^{d-1}$
2. Carol moves token to any $x^{k+1} \in \mathcal{X}$ satisfying

$$ (x^{k+1} - x^k) \cdot v > 0. $$
A two player game for convex hull peeling

**Players:** Paul and Carol

**State space:** \( \mathcal{X} := \{X_1, \ldots, X_n\} \)

**Paul’s goal:** Reach vertex of convex hull

**Carol’s goal:** Obstruct Paul

**Rules of the game:** Token starts at \( x^0 \in \mathcal{X} \) and is moved according to:

1. Paul picks \( v \in S^{d-1} \)
2. Carol moves token to any \( x^{k+1} \in \mathcal{X} \) satisfying
   \[
   (x^{k+1} - x^k) \cdot v > 0.
   \]
A two player game for convex hull peeling

**Players:** Paul and Carol  
**State space:** $\mathcal{X} := \{X_1, \ldots, X_n\}$

**Paul’s goal:** Reach vertex of convex hull  
**Carol’s goal:** Obstruct Paul

**Rules of the game:** Token starts at $x^0 \in \mathcal{X}$ and is moved according to:

1. Paul picks $v \in S^{d-1}$
2. Carol moves token to any $x^{k+1} \in \mathcal{X}$ satisfying

$$ (x^{k+1} - x^k) \cdot v > 0. $$
A two player game for convex hull peeling

**Players:** Paul and Carol  
**State space:** $\mathcal{X} := \{X_1, \ldots, X_n\}$

**Paul’s goal:** Reach vertex of convex hull  
**Carol’s goal:** Obstruct Paul

**Rules of the game:** Token starts at $x^0 \in \mathcal{X}$ and is moved according to:

1. Paul picks $v \in S^{d-1}$
2. Carol moves token to any $x^{k+1} \in \mathcal{X}$ satisfying

$$ (x^{k+1} - x^k) \cdot v > 0. $$
A two player game for convex hull peeling

**Players:** Paul and Carol

**State space:** $\mathcal{X} := \{X_1, \ldots, X_n\}$

**Paul’s goal:** Reach vertex of convex hull

**Carol’s goal:** Obstruct Paul

**Rules of the game:** Token starts at $x^0 \in \mathcal{X}$ and is moved according to:

1. Paul picks $v \in \mathbb{S}^{d-1}$
2. Carol moves token to any $x^{k+1} \in \mathcal{X}$ satisfying

\[(x^{k+1} - x^k) \cdot v > 0.\]
A two player game for convex hull peeling

**Players:** Paul and Carol  
**State space:** $X := \{X_1, \ldots, X_n\}$

**Paul’s goal:** Reach vertex of convex hull  
**Carol’s goal:** Obstruct Paul

**Rules of the game:** Token starts at $x^0 \in X$ and is moved according to:

1. Paul picks $v \in S^{d-1}$
2. Carol moves token to any $x^{k+1} \in X$ satisfying

   $$(x^{k+1} - x^k) \cdot v > 0.$$
A two player game for convex hull peeling

**Players:** Paul and Carol

**State space:** $\mathcal{X} := \{X_1, \ldots, X_n\}$

**Paul’s goal:** Reach vertex of convex hull

**Carol’s goal:** Obstruct Paul

**Rules of the game:** Token starts at $x^0 \in \mathcal{X}$ and is moved according to:

1. Paul picks $v \in S^{d-1}$
2. Carol moves token to any $x^{k+1} \in \mathcal{X}$ satisfying
   
   $$(x^{k+1} - x^k) \cdot v > 0.$$
A two player game for convex hull peeling

Paul’s optimal choice: Any halfspace supporting current convex layer
Carol’s optimal choice: Any point on the previous convex layer
A two player game for convex hull peeling

Paul’s optimal choice: Any halfspace supporting current convex layer
Carol’s optimal choice: Any point on the previous convex layer
A two player game for convex hull peeling

Paul’s optimal choice: Any halfspace supporting current convex layer
Carol’s optimal choice: Any point on the previous convex layer
A two player game for convex hull peeling

Paul’s optimal choice: Any halfspace supporting current convex layer

Carol’s optimal choice: Any point on the previous convex layer

Value function $= U_n(x^0) = \text{Convex depth function}$.
A two player game for convex hull peeling

\[ n = 50 \text{ points} \]
A two player game for convex hull peeling

$n = 10^5$ points
A two player game for convex hull peeling

\[ n = 10^5 \text{ points} \]
A PDE continuum limit for convex hull peeling

Let $X_1, \ldots, X_n$ be i.i.d. with a continuous density $f$ on a convex set $\Omega \subset \mathbb{R}^d$.

Let $U_n$ be the function that ‘counts’ the associated convex layers $C_1, C_2, \ldots$.
Partial differential equation (PDE) continuum limit

**Theorem (Joint with C. Smart)**

There exists a universal constant $\alpha_d$ such that with probability one

$$n^{-\frac{2}{d+1}} U_n \longrightarrow \alpha_d u \quad \text{uniformly on } \Omega,$$

where $u \in C(\overline{\Omega})$ is the unique viscosity solution of

\[
\begin{cases}
\nabla u \cdot \text{cof}(-\nabla^2 u) \nabla u = f^2 & \text{in } \Omega \\
u = 0 & \text{on } \partial \Omega.
\end{cases}
\]

(1)
Partial differential equation (PDE) continuum limit

**Theorem (Joint with C. Smart)**

There exists a universal constant \( \alpha_d \) such that with probability one

\[
n^{-\frac{2}{d+1}} U_n \to \alpha_d u \quad \text{uniformly on } \Omega,
\]

where \( u \in C(\overline{\Omega}) \) is the unique viscosity solution of

\[
\begin{cases}
    \nabla u \cdot \text{cof}(-\nabla^2 u) \nabla u = f^2 & \text{in } \Omega \\
    u = 0 & \text{on } \partial \Omega.
\end{cases}
\]

(1)

This is just motion by a power of Gauss curvature

\[
\frac{dS}{dt} = f^{-2/(d+1)} \kappa_G^{1/(d+1)} n.
\]
A PDE continuum limit for convex hull peeling

Figure: Convex layers vs continuum limit for $n = 5 \times 10^3$. 
A nonconvex example

(a) Samples
(b) Convex layers

Figure: Convex layers corresponding to disjoint clusters.
A nonconvex example

Figure: Two different solutions continuum PDE.
The halfmoon

(a) Samples

(b) Convex layers

Figure: Convex layers corresponding to the halfmoon distribution.
The halfmoon

(a) Samples

(b) PDE

Figure: Solution of PDE for the halfmoon example.
Outline

1. Nondominated sorting
2. Convex hull peeling
3. Semi-supervised learning
4. References
Quick intro to learning

**Fully supervised:** In fully supervised learning, we are given training data \((x_i, y_i)\) for \(i = 1, \ldots, n\), where \(x_i \in \mathcal{X}\) are the data points and \(y_i \in \mathcal{Y}\) are the known labels.
Quick intro to learning

**Fully supervised:** In fully supervised learning, we are given training data \((x_i, y_i)\) for \(i = 1, \ldots, n\), where \(x_i \in \mathcal{X}\) are the data points and \(y_i \in \mathcal{Y}\) are the known labels. The goal is to learn a function

\[
u : \mathcal{X} \to \mathcal{Y} \quad \text{for which} \quad u(x_i) \approx y_i \quad \text{for} \quad i = 1, \ldots, n.
\] (2)
Quick intro to learning

**Fully supervised:** In fully supervised learning, we are given training data \((x_i, y_i)\) for \(i = 1, \ldots, n\), where \(x_i \in \mathcal{X}\) are the data points and \(y_i \in \mathcal{Y}\) are the known labels. The goal is to learn a function

\[
u : \mathcal{X} \rightarrow \mathcal{Y} \quad \text{for which } u(x_i) \approx y_i \text{ for } i = 1, \ldots, n. \tag{2}\]

**Semi-supervised learning:** In semi-supervised learning, we are additionally given a (usually large) amount of unlabeled data \(x_{n+1}, \ldots, x_{n+m}\) for \(m \geq 1\).
Quick intro to learning

**Fully supervised:** In fully supervised learning, we are given training data \((x_i, y_i)\) for \(i = 1, \ldots, n\), where \(x_i \in X\) are the data points and \(y_i \in Y\) are the known labels. The goal is to learn a function

\[
u : X \to Y \quad \text{for which} \quad u(x_i) \approx y_i \quad \text{for} \quad i = 1, \ldots, n.
\]

**Semi-supervised learning:** In semi-supervised learning, we are additionally given a (usually large) amount of unlabeled data \(x_{n+1}, \ldots, x_{n+m}\) for \(m \geq 1\). Goal is to use the unlabeled data to aid the learning.
Quick intro to learning

**Fully supervised:** In fully supervised learning, we are given training data \((x_i, y_i)\) for \(i = 1, \ldots, n\), where \(x_i \in \mathcal{X}\) are the data points and \(y_i \in \mathcal{Y}\) are the known labels. The goal is to learn a function

\[
u : \mathcal{X} \rightarrow \mathcal{Y} \quad \text{for which } u(x_i) \approx y_i \text{ for } i = 1, \ldots, n. \tag{2}\]

**Semi-supervised learning:** In semi-supervised learning, we are additionally given a (usually large) amount of unlabeled data \(x_{n+1}, \ldots, x_{n+m}\) for \(m \geq 1\). Goal is to use the unlabeled data to aid the learning.

**Inductive learning:** Learn a function

\[
u : \mathcal{X} \rightarrow \mathcal{Y} \quad \text{for which } u(x_i) \approx y_i \text{ for } i = 1, \ldots, n.\]
Quick intro to learning

**Fully supervised:** In fully supervised learning, we are given training data \((x_i, y_i)\) for \(i = 1, \ldots, n\), where \(x_i \in \mathcal{X}\) are the data points and \(y_i \in \mathcal{Y}\) are the known labels. The goal is to learn a function

\[
u: \mathcal{X} \to \mathcal{Y} \quad \text{for which } u(x_i) \approx y_i \text{ for } i = 1, \ldots, n.
\] (2)

**Semi-supervised learning:** In semi-supervised learning, we are additionally given a (usually large) amount of unlabeled data \(x_{n+1}, \ldots, x_{n+m}\) for \(m \geq 1\). Goal is to use the unlabeled data to aid the learning.

1. **Inductive learning:** Learn a function

\[
u: \mathcal{X} \to \mathcal{Y} \quad \text{for which } u(x_i) \approx y_i \text{ for } i = 1, \ldots, n.
\]

2. **Transductive learning:** Learn a function

\[
u: \{x_1, x_2, \ldots, x_{n+m}\} \to \mathcal{Y} \quad \text{for which } u(x_i) \approx y_i \text{ for } i = 1, \ldots, n
\]
Quick intro to learning

**Fully supervised:** In fully supervised learning, we are given training data \((x_i, y_i)\) for \(i = 1, \ldots, n\), where \(x_i \in \mathcal{X}\) are the data points and \(y_i \in \mathcal{Y}\) are the known labels. The goal is to learn a function

\[
u : \mathcal{X} \rightarrow \mathcal{Y} \quad \text{for which } u(x_i) \approx y_i \text{ for } i = 1, \ldots, n.\]

(2)

**Semi-supervised learning:** In semi-supervised learning, we are additionally given a (usually large) amount of unlabeled data \(x_{n+1}, \ldots, x_{n+m}\) for \(m \geq 1\). Goal is to use the unlabeled data to aid the learning.

1. **Inductive learning:** Learn a function

\[
u : \mathcal{X} \rightarrow \mathcal{Y} \quad \text{for which } u(x_i) \approx y_i \text{ for } i = 1, \ldots, n.\]

2. **Transductive learning:** Learn a function

\[
u : \{x_1, x_2, \ldots, x_{n+m}\} \rightarrow \mathcal{Y} \quad \text{for which } u(x_i) \approx y_i \text{ for } i = 1, \ldots, n\]

Classification when \(\mathcal{Y}\) finite – Regression when \(\mathcal{Y} = \mathbb{R}^d\).
Example: Automated image captioning
Example: Automated image captioning

- A woman is throwing a frisbee in a park.
- A dog is standing on a hardwood floor.
- A stop sign is on a road with a mountain in the background.
- A little girl sitting on a bed with a teddy bear.
- A group of people sitting on a boat in the water.
- A giraffe standing in a forest with trees in the background.

Example: Automated image captioning fail

[Andrej Karpathy’s NeuralTalk]

(-11.269838) a woman holding a baby giraffe in a zoo
Applications

Why is semi-supervised learning useful?
Applications

Why is semi-supervised learning useful?

It is expensive to label data, and we have an abundance of unlabeled data.
Applications

Why is semi-supervised learning useful?

It is expensive to label data, and we have an abundance of unlabeled data.

Brief list of example applications:

1. Speech recognition
Applications

Why is semi-supervised learning useful?

It is expensive to label data, and we have an abundance of unlabeled data.

Brief list of example applications:

1. Speech recognition
2. Webpage classification
Applications

Why is semi-supervised learning useful?

It is expensive to label data, and we have an abundance of unlabeled data.

Brief list of example applications:
1. Speech recognition
2. Webpage classification
3. Inferring protein structure from sequencing
Applications

Why is semi-supervised learning useful?

It is expensive to label data, and we have an abundance of unlabeled data.

Brief list of example applications:

1. Speech recognition
2. Webpage classification
3. Inferring protein structure from sequencing

A great introductory book [Chapelle et al., 2006].
Graph-based semi-supervised learning

Model:

1. Data (labeled and unlabeled) is a graph \((\mathcal{X}, \mathcal{W})\).
Graph-based semi-supervised learning

Model:

Data (labeled and unlabeled) is a graph \((\mathcal{X}, \mathcal{W})\).

- \(\mathcal{X} \subset \mathbb{R}^d\) are the vertices and
- \(\mathcal{W} = (w_{xy})_{x,y \in \mathcal{X}}\) are the nonnegative edge weights.
- \(w_{xy} \approx 1\) if \(x, y\) similar, and \(w_{xy} \approx 0\) when dissimilar.

Labeled (or observed) vertices are a subset \(O \subset \mathcal{X}\).

We given a labelling function \(g: O \rightarrow \mathbb{R}\).

Task: Extend the labels from \(O\) to the entire graph \(\mathcal{X}\).

Semi-supervised smoothness assumption
Similar points \(x, y \in \mathcal{X}\) in high density regions of the graph should have similar labels.
Graph-based semi-supervised learning

Model:

1. Data (labeled and unlabeled) is a graph \((\mathcal{X}, \mathcal{W})\).
   - \(\mathcal{X} \subset \mathbb{R}^d\) are the vertices and
   - \(\mathcal{W} = (w_{xy})_{x, y \in \mathcal{X}}\) are the nonnegative edge weights.
   - \(w_{xy} \approx 1\) if \(x, y\) similar, and \(w_{xy} \approx 0\) when dissimilar.

2. Labeled (or observed) vertices are a subset \(\mathcal{O} \subset \mathcal{X}\).
Graph-based semi-supervised learning

Model:

1. Data (labeled and unlabeled) is a graph \((\mathcal{X}, \mathcal{W})\).
   - \(\mathcal{X} \subset \mathbb{R}^d\) are the vertices and
   - \(\mathcal{W} = (w_{xy})_{x,y \in \mathcal{X}}\) are the nonnegative edge weights.
   - \(w_{xy} \approx 1\) if \(x, y\) similar, and \(w_{xy} \approx 0\) when dissimilar.

2. Labeled (or observed) vertices are a subset \(\mathcal{O} \subset \mathcal{X}\).

3. We given a labelling function \(g : \mathcal{O} \rightarrow \mathbb{R}\).
Graph-based semi-supervised learning

Model:

1. Data (labeled and unlabeled) is a graph \((\mathcal{X}, \mathcal{W})\).
   - \(\mathcal{X} \subset \mathbb{R}^d\) are the vertices and
   - \(\mathcal{W} = (w_{xy})_{x,y \in \mathcal{X}}\) are the nonnegative edge weights.
   - \(w_{xy} \approx 1\) if \(x, y\) similar, and \(w_{xy} \approx 0\) when dissimilar.

2. Labeled (or observed) vertices are a subset \(\mathcal{O} \subset \mathcal{X}\).

3. We given a labelling function \(g : \mathcal{O} \rightarrow \mathbb{R}\).

Task: Extend the labels from \(\mathcal{O}\) to the entire graph \(\mathcal{X}\).
Graph-based semi-supervised learning

Model:

1. Data (labeled and unlabeled) is a graph \((\mathcal{X}, \mathcal{W})\).
   - \(\mathcal{X} \subset \mathbb{R}^d\) are the vertices and
   - \(\mathcal{W} = (w_{xy})_{x,y \in \mathcal{X}}\) are the nonnegative edge weights.
   - \(w_{xy} \approx 1\) if \(x, y\) similar, and \(w_{xy} \approx 0\) when dissimilar.

2. Labeled (or observed) vertices are a subset \(\mathcal{O} \subset \mathcal{X}\).

3. We given a labelling function \(g : \mathcal{O} \rightarrow \mathbb{R}\).

Task: Extend the labels from \(\mathcal{O}\) to the entire graph \(\mathcal{X}\).

Semi-supervised smoothness assumption

Similar points \(x, y \in \mathcal{X}\) in high density regions of the graph should have similar labels.
Laplacian regularization

\[
\min_{u: \mathcal{X} \to \mathbb{R}} \sum_{x, y \in \mathcal{X}} w_{xy}^2 (u(x) - u(y))^2 \quad \text{subject to } u(x) = g(x) \text{ for all } x \in \mathcal{O}.
\]
Laplacian regularization

\[
\min_{u: \mathcal{X} \to \mathbb{R}} \sum_{x, y \in \mathcal{X}} w_{xy}^2 (u(x) - u(y))^2 \quad \text{subject to } u(x) = g(x) \text{ for all } x \in \mathcal{O}.
\]

The minimizer \( u : \mathcal{X} \to \mathbb{R} \) satisfies the linear system

\[
\sum_{y \in \mathcal{X}} w_{xy}^2 (u(x) - u(y)) = 0 \quad \text{for all } x \in \mathcal{X} \setminus \mathcal{O}.
\]
Laplacian regularization

$$\min_{u: \mathcal{X} \rightarrow \mathbb{R}} \sum_{x, y \in \mathcal{X}} w_{xy}^2 (u(x) - u(y))^2 \quad \text{subject to } u(x) = g(x) \text{ for all } x \in \mathcal{O}.$$ 

The minimizer $u : \mathcal{X} \rightarrow \mathbb{R}$ satisfies the linear system

$$\sum_{y \in \mathcal{X}} w_{xy}^2 (u(x) - u(y)) = 0 \quad \text{for all } x \in \mathcal{X} \setminus \mathcal{O}.$$ 

References:

- Original work [Zhu et al., 2003]
- Learning [Zhou et al., 2005][Ando and Zhang, 2007]
- Manifold ranking [He et al., 2006] [Wang et al., 2013] [Yang et al., 2013] [Zhou et al., 2011] [Xu et al., 2011]
Ill-posed with small amount of labeled data

Graph is \( n = 10^n \) i.i.d. random variables uniformly drawn from \([0, 1]^2\).

\[
\begin{align*}
\text{if } |x - y| < 0.01, \quad w_{xy} &= 1 \\
\text{otherwise, } \quad w_{xy} &= 0
\end{align*}
\]

Over 95% of labels in \([0.4975, 0.5025]\). [Nadler et al., 2009] [El Alaoui et al., 2016]
Ill-posed with small amount of labeled data

Graph is $n = 10^5$ i.i.d. random variables uniformly drawn from $[0, 1]^2$.

- $w_{xy} = 1$ if $|x - y| < 0.01$ and $w_{xy} = 0$ otherwise.
- Over 95% of labels in $[0.4975, 0.5025]$.

[Nadler et al., 2009][El Alaoui et al., 2016]
$\ell_p$-based Laplacian regularization

For any $p < \infty$:

$$\min_{u: \mathcal{X} \rightarrow \mathbb{R}} \sum_{x, y \in \mathcal{X}} w_{xy}^{p} |u(x) - u(y)|^{p} \quad \text{subject to } u(x) = g(x) \text{ for all } x \in \mathcal{O}. \quad (3)$$

We can send $p \to \infty$:

$$\min_{u: \mathcal{X} \rightarrow \mathbb{R}} \max_{x, y \in \mathcal{X}} \{w_{xy}^{p} |u(x) - u(y)|^{p}\} \quad \text{subject to } u(x) = g(x) \text{ for all } x \in \mathcal{O}. \quad (4)$$

References:

Finite $p$: [Bridle and Zhu, 2013] [Alamgir and Luxburg, 2011]

$p = \infty$: [Kyng et al., 2015] [Luxburg and Bousquet, 2004]

Absolutely minimal Lipschitz extensions: [Aronsson et al., 2004]
\( \ell_p \)-based Laplacian regularization

For any \( p < \infty \):

\[
\min_{u: \mathcal{X} \rightarrow \mathbb{R}} \sum_{x, y \in \mathcal{X}} w_{xy}^p |u(x) - u(y)|^p \quad \text{subject to } u(x) = g(x) \text{ for all } x \in \mathcal{O}. \tag{3}
\]

We can send \( p \to \infty \):

\[
\min_{u: \mathcal{X} \rightarrow \mathbb{R}} \max_{x, y \in \mathcal{X}} \{w_{xy}|u(x) - u(y)|\} \quad \text{subject to } u(x) = g(x) \text{ for all } x \in \mathcal{O}. \tag{4}
\]
\( \ell_p \)-based Laplacian regularization

For any \( p < \infty \):

\[
\min_{u: \mathcal{X} \to \mathbb{R}} \sum_{x, y \in \mathcal{X}} w_{xy}^p |u(x) - u(y)|^p \quad \text{subject to } u(x) = g(x) \text{ for all } x \in \mathcal{O}. \tag{3}
\]

We can send \( p \to \infty \):

\[
\min_{u: \mathcal{X} \to \mathbb{R}} \max_{x, y \in \mathcal{X}} \{w_{xy}|u(x) - u(y)|\} \quad \text{subject to } u(x) = g(x) \text{ for all } x \in \mathcal{O}. \tag{4}
\]

References:

- Finite \( p \): [Bridle and Zhu, 2013][Alamgir and Luxburg, 2011]
- \( p = \infty \): [Kyng et al., 2015] [Luxburg and Bousquet, 2004]
- Absolutely minimal Lipschitz extensions: [Aronsson et al., 2004]
$p$-Laplacian learning: $n = 10^5$ points, $h = 10^{-2}$

Simulations are the work of Mauricio Flores (co-supervised by Gilad Lerman).
$p$-Laplacian learning: $n = 10^5$ points, $h = 10^{-2}$

Simulations are the work of Mauricio Flores (co-supervised by Gilad Lerman).
\( p \)-Laplacian learning:  \( n = 10^5 \) points,  \( h = 10^{-2} \)

\[
p = 2.5
\]

Simulations are the work of Mauricio Flores (co-supervised by Gilad Lerman).
$p$-Laplacian learning: $n = 10^5$ points, $h = 10^{-2}$

$p = 3$

Simulations are the work of Mauricio Flores (co-supervised by Gilad Lerman).
$p$-Laplacian learning: $n = 10^5$ points, $h = 10^{-2}$

Simulations are the work of Mauricio Flores (co-supervised by Gilad Lerman).
$p$-Laplacian learning: $n = 10^5$ points, $h = 10^{-2}$

$p = \infty$

Simulations are the work of Mauricio Flores (co-supervised by Gilad Lerman).
$p$-Laplacian learning: Varying density

Simulations are the work of Mauricio Flores (co-supervised by Gilad Lerman).

$p = 2$
$p$-Laplacian learning: Varying density

Simulations are the work of Mauricio Flores (co-supervised by Gilad Lerman).

$p = 2$
\( p \)-Laplacian learning: Varying density

\[
p = 2.5
\]

Simulations are the work of Mauricio Flores (co-supervised by Gilad Lerman).
\[ p = 3 \]

Simulations are the work of Mauricio Flores (co-supervised by Gilad Lerman).
$p$-Laplacian learning: Varying density

$p = 5$

Simulations are the work of Mauricio Flores (co-supervised by Gilad Lerman).
Random model

- **Labeled data:** The labeled data is a fixed finite collection of $N$ points

$$\mathcal{O} = \{y_1, \ldots, y_N\} \subset U \subset \mathbb{T}^d := \mathbb{R}^d / \mathbb{Z}^d.$$
Random model

- **Labeled data:** The labeled data is a fixed finite collection of $N$ points

$$\mathcal{O} = \{y_1, \ldots, y_N\} \subset U \subset \mathbb{T}^d := \mathbb{R}^d / \mathbb{Z}^d.$$

- **Unlabeled data:** The unlabeled data is a sequence $x_1, x_2, \ldots, x_n$ of i.i.d. random variables with probability density $f : \mathbb{T}^d \to \mathbb{R}$

$$X_{nf} := \{x_1, x_2, \ldots, x_n\}.$$
Random model

- **Labeled data:** The labeled data is a fixed finite collection of $N$ points

  $$\mathcal{O} = \{y_1, \ldots, y_N\} \subset U \subset \mathbb{T}^d := \mathbb{R}^d / \mathbb{Z}^d.$$

- **Unlabeled data:** The unlabeled data is a sequence $x_1, x_2, \ldots, x_n$ of i.i.d. random variables with probability density $f : \mathbb{T}^d \to \mathbb{R}$

  $$X_{nf} := \{x_1, x_2, \ldots, x_n\}.$$

- **Vertices of graph:** The vertices of the graph are

  $$\mathcal{X}_n = X_{nf} \cup \mathcal{O}.$$
Random model

- **Labeled data:** The labeled data is a fixed finite collection of $N$ points
  \[ O = \{ y_1, \ldots, y_N \} \subset U \subset \mathbb{T}^d := \mathbb{R}^d / \mathbb{Z}^d. \]

- **Unlabeled data:** The unlabeled data is a sequence $x_1, x_2, \ldots, x_n$ of i.i.d. random variables with probability density $f : \mathbb{T}^d \to \mathbb{R}$
  \[ X_{nf} := \{ x_1, x_2, \ldots, x_n \}. \]

- **Vertices of graph:** The vertices of the graph are
  \[ \mathcal{X}_n = X_{nf} \cup O. \]

- **Edge weights:** The edge weights are
  \[ w_{xy} = \Phi \left( \frac{|x - y|}{h} \right), \]
  where $h > 0$, and $\Phi : [0, \infty) \to [0, \infty)$. 

Random model

For $p < \infty$ we write

$$J_p(u) := \sum_{x, y \in X_n} w_{xy}^p |u(x) - u(y)|^p,$$

and for $p = \infty$ we write

$$J_\infty(u) := \max_{x, y \in X_n} \{ w_{xy} |u(x) - u(y)| \}.$$
Random model

For $p < \infty$ we write

$$J_p(u) := \sum_{x, y \in \mathcal{X}_n} w_{xy}^p |u(x) - u(y)|^p,$$

and for $p = \infty$ we write

$$J_\infty(u) := \max_{x, y \in \mathcal{X}_n} \{ w_{xy} |u(x) - u(y)| \}.$$

For $n \geq 1$, let $u_n : \mathcal{X}_n \to \mathbb{R}$ be the solution of

$$\min_{u : \mathcal{X}_n \to \mathbb{R}} J_p(u) \quad \text{subject to} \quad u(x) = g(x) \text{ for all } x \in \mathcal{O}.$$
Random model

For \( p < \infty \) we write
\[
J_p(u) := \sum_{x, y \in \mathcal{X}_n} w_{xy}^p |u(x) - u(y)|^p,
\]
and for \( p = \infty \) we write
\[
J_{\infty}(u) := \max_{x, y \in \mathcal{X}_n} \{ w_{xy} |u(x) - u(y)| \}.
\]

For \( n \geq 1 \), let \( u_n : \mathcal{X}_n \to \mathbb{R} \) be the solution of
\[
\min_{u: \mathcal{X}_n \to \mathbb{R}} J_p(u) \quad \text{subject to} \quad u(x) = g(x) \quad \text{for all} \ x \in \mathcal{O}.
\]

Question: What can we say about \( u_n \) as \( n \to \infty \)?
Let
\[ r_n = \sup \{ s > 0 \mid B(x, s) \cap \mathcal{X}_n = \emptyset \text{ for some } x \in U \}. \] (5)

**Theorem (p = \infty [Calder, 2017a])**

*Suppose that* \( h_n \to 0 \text{ such that} \)
\[ \lim_{n \to \infty} \frac{r_n^2}{h_n^3} = 0. \] (6)

*Then* \( u_n \to u \text{ uniformly as } n \to \infty, \) (7)

*where* \( u \in C(T^d) \text{ is the unique viscosity solution of the } \infty\text{-Laplace equation}*
\[
\begin{cases}
\Delta_\infty u = 0 & \text{in } T^d \setminus \mathcal{O} \\
u = g & \text{on } \mathcal{O}
\end{cases}
\] (8)

*Note that (6) holds almost surely when*
\[ \lim_{n \to \infty} \frac{nh_n^{3d/2}}{\log(n)} = \infty. \] (9)
Theorem (Finite $p$ [Calder, 2017b])

Let $d < p < \infty$, and suppose that $h_n \to 0$ such that

$$
\lim_{n \to \infty} nh_n^p = 0 \quad \text{and} \quad \lim_{n \to \infty} \frac{nh_n^{d+4}}{\log(n)} = \infty.
$$

(10)

Then with probability one

$$
\lim_{n \to \infty} u_n \to u \quad \text{uniformly as} \quad n \to \infty,
$$

(11)

where $u \in C(\mathbb{T}^d)$ is the unique viscosity solution of the weighted $p$-Laplace equation

$$
\begin{cases}
\text{div} (f^2 |\nabla u|^{p-2}\nabla u) = 0 & \text{in } \mathbb{T}^d \setminus \mathcal{O} \\
\quad u = g & \text{on } \mathcal{O}
\end{cases}
$$

(12)

A very similar result appeared recently in [Slepčev and Thorpe, 2017].
Regularity in semi-supervised learning

The PDE-limit can be used to prove Hölder regularity.

**Theorem**

Assume $p > d$. For every $\alpha < \frac{p-d}{p-1}$ there exists $C, \delta$ such that

$$\mathbb{P} \left[ \forall x, y \in \mathcal{X}_n, \ |u_n(x) - u_n(y)| \leq C(|x - y|^{\alpha} + n^{\frac{1}{p}} h) \right] \geq 1 - \exp \left( -\delta nh^{d+4} + C \log(n) \right).$$
Graph Laplacians

\[
\min_{u: \mathcal{X}_n \to \mathbb{R}} J_p(u) = \sum_{x, y \in \mathcal{X}_n} w_{xy}^p |u(x) - u(y)|^p \quad \text{subject to } u(x) = g(x) \text{ for } x \in \mathcal{O} \subset \mathcal{X}_n
\]
Graph Laplacians

\[
\min_{u : \mathcal{X}_n \to \mathbb{R}} J_p(u) = \sum_{x, y \in \mathcal{X}_n} w_{xy}^p |u(x) - u(y)|^p \quad \text{subject to } u(x) = g(x) \text{ for } x \in \mathcal{O} \subset \mathcal{X}_n
\]

The minimizer \( u : \mathcal{X}_n \to \mathbb{R} \) satisfies

\[
\begin{aligned}
\Delta_{\mathcal{X}_n}^p u &= 0 \quad \text{in } \mathcal{X}_n \setminus \mathcal{O}, \\
u &= g \quad \text{on } \mathcal{O},
\end{aligned}
\]

where \( \Delta_{\mathcal{X}_n}^p u : \mathcal{X} \to \mathbb{R} \) is the graph \( p \)-Laplacian defined by

\[
\Delta_{\mathcal{X}_n}^p u(x) = \sum_{y \in \mathcal{X}_n} w_{xy}^p |u(y) - u(x)|^{p-2}(u(y) - u(x)).
\]
Graph Laplacians

\[
\min_{u: \mathcal{X}_n \to \mathbb{R}} J_p(u) = \sum_{x, y \in \mathcal{X}_n} w_{xy}^p |u(x) - u(y)|^p \quad \text{subject to } u(x) = g(x) \text{ for } x \in \mathcal{O} \subset \mathcal{X}_n
\]

The minimizer \( u : \mathcal{X}_n \to \mathbb{R} \) satisfies

\[
\begin{align*}
\Delta_{\mathcal{X}_n}^p u &= 0 \quad \text{in } \mathcal{X}_n \setminus \mathcal{O}, \\
u &= g \quad \text{on } \mathcal{O},
\end{align*}
\]

where \( \Delta_{\mathcal{X}_n}^p u : \mathcal{X} \to \mathbb{R} \) is the graph \( p \)-Laplacian defined by

\[
\Delta_{\mathcal{X}_n}^p u(x) = \sum_{y \in \mathcal{X}_n} w_{xy}^p |u(y) - u(x)|^{p-2}(u(y) - u(x)).
\]

References on graph \( p \)-Laplacian:

- [Manfredi et al., 2015]
- [Zhou and Schölkopf, 2005]
- [Amghibech, 2003]
- [Bühler and Hein, 2009]
- [Luo et al., 2010]
Graph Laplacian as $p \to \infty$

Note that solutions of

$$\Delta_{p}^{X_n} u(x) = \sum_{y \in X_n} w_{xy}^{p} |u(y) - u(x)|^{p-2}(u(y) - u(x)) = 0$$

satisfy

$$\left( \sum_{\begin{subarray}{c} y \in X_n \\ u(y) \geq u(x) \end{subarray}} w_{xy}^{p} |u(y) - u(x)|^{p-1} \right)^{1/p} = \left( \sum_{\begin{subarray}{c} y \in X_n \\ u(y) < u(x) \end{subarray}} w_{xy}^{p} |u(y) - u(x)|^{p-1} \right)^{1/p}.$$
Graph Laplacian as $p \to \infty$

Note that solutions of

$$\Delta^{\mathcal{X}_n}_p u(x) = \sum_{y \in \mathcal{X}_n} w^{p}_{xy} |u(y) - u(x)|^{p-2}(u(y) - u(x)) = 0$$

satisfy

$$\left( \sum_{y \in \mathcal{X}_n \atop u(y) \geq u(x)} w^{p}_{xy} |u(y) - u(x)|^{p-1} \right)^{1/p} = \left( \sum_{y \in \mathcal{X}_n \atop u(y) < u(x)} w^{p}_{xy} |u(y) - u(x)|^{p-1} \right)^{1/p}.$$

Send $p \to \infty$ to get

$$\max_{y \in \mathcal{X}_n} w_{xy}(u(y) - u(x)) = \max_{y \in \mathcal{X}_n} w_{xy}(u(x) - u(y)).$$

or

$$\Delta^{\mathcal{X}_n}_\infty u(x) := \max_{y \in \mathcal{X}_n} w_{xy}(u(y) - u(x)) + \min_{y \in \mathcal{X}_n} w_{xy}(u(y) - u(x)) = 0.$$
Graph Laplacians

\[ \min_{u: \mathcal{X}_n \to \mathbb{R}} J_\infty(u) = \max_{x,y \in \mathcal{X}_n} w_{xy}|u(x) - u(y)| \quad \text{subject to } u(x) = g(x) \text{ for } x \in \mathcal{O} \subset \mathcal{X}_n \]
**Graph Laplacians**

\[
\min_{u : \mathcal{X}_n \to \mathbb{R}} J_\infty(u) = \max_{x, y \in \mathcal{X}_n} w_{xy} |u(x) - u(y)| \quad \text{subject to} \quad u(x) = g(x) \text{ for } x \in \mathcal{O} \subset \mathcal{X}_n
\]

The minimizer \( u : \mathcal{X}_n \to \mathbb{R} \) satisfies

\[
\begin{cases}
\Delta_{\mathcal{X}_n}^\infty u = 0 & \text{in } \mathcal{X}_n \setminus \mathcal{O} \\
 u = g & \text{in } \mathcal{O},
\end{cases}
\]

where \( \Delta_{\mathcal{X}_n}^\infty u : \mathcal{X}_n \to \mathbb{R} \) is the graph \( \infty \)-Laplacian defined by

\[
\Delta_{\mathcal{X}_n}^\infty u(x) = \max_{y \in \mathcal{X}_n} w_{xy}(u(y) - u(x)) + \min_{y \in \mathcal{X}_n} w_{xy}(u(y) - u(x))
\]
Graph Laplacians

\[
\min_{u: \mathcal{X}_n \to \mathbb{R}} J_\infty(u) = \max_{x, y \in \mathcal{X}_n} w_{xy} |u(x) - u(y)| \quad \text{subject to } u(x) = g(x) \text{ for } x \in \mathcal{O} \subset \mathcal{X}_n
\]

The minimizer \( u : \mathcal{X}_n \to \mathbb{R} \) satisfies

\[
\begin{cases}
\Delta_\infty \mathcal{X}_n u = 0 & \text{in } \mathcal{X}_n \setminus \mathcal{O} \\
u = g & \text{in } \mathcal{O},
\end{cases}
\]

where \( \Delta_\infty \mathcal{X}_n u : \mathcal{X}_n \to \mathbb{R} \) is the graph \( \infty \)-Laplacian defined by

\[
\Delta_\infty \mathcal{X}_n u(x) = \max_{y \in \mathcal{X}_n} w_{xy}(u(y) - u(x)) + \min_{y \in \mathcal{X}_n} w_{xy}(u(y) - u(x))
\]

Reference:

1. [Kyng et al., 2015]
Game theoretic $p$-Lapacian

We can also consider the game theoretic $p$-Laplacian for semi-supervised learning:

$$\begin{cases}
\frac{1}{d_n} \Delta_2^{\mathcal{X}_n} u_n + \lambda(p - 2) \Delta_\infty^{\mathcal{X}_n} u_n = 0 & \text{in } \mathcal{X}_n \setminus \mathcal{O} \\
\lambda = \lambda(\Phi) \\
u = g & \text{in } \mathcal{O},
\end{cases}$$

where $d_n(x) = \sum_{y \in \mathcal{X}_n} w_{xy}^2$ and $\lambda = \lambda(\Phi)$. 
Game theoretic $p$-Laplacian

We can also consider the game theoretic $p$-Laplacian for semi-supervised learning:

$$\begin{cases} 
\frac{1}{d_n} \Delta_2^X u_n + \lambda(p - 2) \Delta_\infty^X u_n = 0 & \text{in } X_n \setminus \mathcal{O} \\
\lambda \quad \text{in } \mathcal{O}, 
\end{cases}$$

where $d_n(x) = \sum_{y \in X_n} w_{xy}^2$ and $\lambda = \lambda(\Phi)$.

This is likely better conditioned numerically when $p$ is large.
Game theoretic $p$-Laplacian

Theorem (Finite $p$ [Calder, 2017b])

Let $d < p < \infty$, and suppose that $h \to 0$ such that

$$\lim_{n \to \infty} \frac{nh^q}{\log(n)} = \infty,$$

where $q = \max\{d + 4, 3d/2\}$. Then with probability one

$$u_n \to u \quad \text{uniformly as} \quad n \to \infty,$$

where $u \in C(\mathbb{T}^d)$ is the unique viscosity solution of the weighted $p$-Laplace equation

$$\begin{cases}
\text{div} \left(f^2 |\nabla u|^{p-2} \nabla u \right) = 0 \quad \text{in} \quad \mathbb{T}^d \setminus \mathcal{O} \\
u = g \quad \text{on} \quad \mathcal{O}
\end{cases}$$

Notice no upper bound on $h$ (i.e., we don’t require $nh^p \to 0$).
Ideas in proof

All graph Laplacians are monotone schemes. We just need consistency and stability.
Ideas in proof

All graph Laplacians are **monotone** schemes. We just need **consistency** and **stability**.

Consistency is straightforward, using concentration of measure and Taylor expansions. For example, for the Graph $p$-Laplacian

$$
\Delta X^p u(x) = \sum_{y \in X} w_{xy} |u(y) - u(x)|^{p-2} (u(y) - u(x))
$$

we have

$$
E[\Delta X^p \phi(x)] = nh d \int_\mathbb{R} d \Phi(|z|) |\phi(x+zh) - \phi(x)|^{p-2} (\phi(x+zh) - \phi(x)) f(x+zh) \, dz.
$$

Plug in Taylor expansions and plug away...
Ideas in proof

All graph Laplacians are monotone schemes. We just need consistency and stability.

Consistency is straightforward, using concentration of measure and Taylor expansions. For example, for the Graph $p$-Laplacian

$$\Delta^X_p u(x) = \sum_{y \in X_n} w_{xy}^p |u(y) - u(x)|^{p-2} (u(y) - u(x)).$$

we have

$$\mathbb{E}[\Delta^X_p \varphi(x)] = n h^d \int_{\mathbb{R}^d} \Phi(|z|)|\varphi(x + zh) - \varphi(x)|^{p-2} (\varphi(x + zh) - \varphi(x)) f(x + zh) \, dz.$$
Ideas in proof

All graph Laplacians are monotone schemes. We just need consistency and stability.

Consistency is straightforward, using concentration of measure and Taylor expansions. For example, for the Graph $p$-Laplacian

$$\Delta^X_{X_n} u(x) = \sum_{y \in X_n} w_{xy}^p |u(y) - u(x)|^{p-2}(u(y) - u(x)).$$

we have

$$\mathbb{E}[\Delta^X_{X_n} \varphi(x)] = n h^d \int_{\mathbb{R}^d} \Phi(|z|)|\varphi(x + zh) - \varphi(x)|^{p-2}(\varphi(x + zh) - \varphi(x))f(x + zh) \, dz.$$ 

Plug in Taylor expansions and plug away...

$$\mathbb{E}[\Delta^X_{X_n} \varphi(x)] = \frac{1}{2} C_p f^{-1} \text{div}(f^2 |\nabla \varphi|^{p-2} \nabla \varphi)n h^{d+p} + R(x) nh^{d+p+1},$$

where

$$|R(x)| \leq C \|\varphi\|^{p-1}_{C^3(\mathbb{R}^d)}.$$
Hölder continuity for $p$-Laplace equation

The maximum principle can be used to prove Hölder continuity when $p > d$:

$$\begin{cases} 
\Delta_p u := \text{div}(|\nabla u|^{p-2}\nabla u) = 0 \quad \text{in} \ U \\
u = g \quad \text{on} \ \partial U,
\end{cases}$$
Hölder continuity for $p$-Laplace equation

The maximum principle can be used to prove Hölder continuity when $p > d$:

$$\begin{cases}
\Delta_p u := \text{div}(|\nabla u|^{p-2}\nabla u) = 0 & \text{in } U \\
u = g & \text{on } \partial U,
\end{cases}$$

Let us define

$$v(x) = u(x_0) + C|x - x_0|^\alpha \quad \text{for} \quad \alpha = \frac{p - d}{p - 1}.$$
Hölder continuity for $p$-Laplace equation

The maximum principle can be used to prove Hölder continuity when $p > d$:

\[
\begin{cases}
\Delta_p u := \text{div}(|\nabla u|^{p-2} \nabla u) = 0 & \text{in } U \\
u = g & \text{on } \partial U,
\end{cases}
\]

Let us define

\[v(x) = u(x_0) + C|x - x_0|^\alpha \quad \text{for } \alpha = \frac{p - d}{p - 1}.
\]

If $B(x_0, r) \subset U$ then for $C = (\max g - \min g)r^{-\alpha}$ we have

\[v(x) \geq u(x) \quad \text{for } |x - x_0| = r.
\]
Hölder continuity for $p$-Laplace equation

The maximum principle can be used to prove Hölder continuity when $p > d$:

$$\begin{cases} \Delta_p u := \text{div}(\nabla u)^{p-2}\nabla u) = 0 & \text{in } U \\ u = g & \text{on } \partial U, \end{cases}$$

Let us define

$$v(x) = u(x_0) + C|x - x_0|^\alpha \quad \text{for } \alpha = \frac{p-d}{p-1}.$$ 

If $B(x_0, r) \subset U$ then for $C = \left(\max g - \min g\right)r^{-\alpha}$ we have

$$v(x) \geq u(x) \quad \text{for } |x - x_0| = r.$$ 

Since $\Delta_p v(x) = 0$ for $x \neq x_0$, we can use the maximum principle to show that

$$u(x) \leq v(x) \quad \text{for all } x \in B(x_0, r).$$
Hölder continuity for $p$-Laplace equation

The maximum principle can be used to prove Hölder continuity when $p > d$:

$$
\begin{cases}
\Delta_p u := \text{div}(\nabla u^{p-2}\nabla u) = 0 & \text{in } U \\
u = g & \text{on } \partial U,
\end{cases}
$$

Let us define

$$v(x) = u(x_0) + C|x - x_0|^\alpha \quad \text{for} \quad \alpha = \frac{p-d}{p-1}.$$

If $B(x_0, r) \subset U$ then for $C = (\max g - \min g)r^{-\alpha}$ we have

$$v(x) \geq u(x) \quad \text{for} \quad |x - x_0| = r.$$

Since $\Delta_p v(x) = 0$ for $x \neq x_0$, we can use the maximum principle to show that

$$u(x) \leq v(x) \quad \text{for all } x \in B(x_0, r).$$

It follows that

$$u(x) - u(x_0) \leq C|x - x_0|^\alpha.$$
It is generally not the case that
\[ \Delta_p^h x_n |x|^{\frac{p-d}{p-1}} = 0. \]
It is generally not the case that

\[ \Delta_p^n |x|^{\frac{p-d}{p-1}} = 0. \]

Outline of regularity proof:

1. Choose \( 0 < \alpha < \frac{(p - d)}{(p - 1)} \) and set \( v(x) = |x - y|^{\alpha} \)
It is generally not the case that
\[ \Delta^X_n |x|^{\frac{p-d}{p-1}} = 0. \]

Outline of regularity proof:

1. Choose \( 0 < \alpha < \frac{(p - d)}{(p - 1)} \) and set \( v(x) = |x - y|^{\alpha} \)

2. Show that \( \Delta^X_n v(x) \leq 0 \) for \( |x - y| \geq ch \) with high probability.
It is generally not the case that
\[ \Delta_{p}^{Xn} |x|^{\frac{p-d}{p-1}} = 0. \]

Outline of regularity proof:
1. Choose \(0 < \alpha < \frac{(p - d)}{(p - 1)}\) and set \(v(x) = |x - y|^\alpha\)
2. Show that \(\Delta_{p}^{Xn} v(x) \leq 0\) for \(|x - y| \geq ch\) with high probability.
3. Fill in the gap \(|x - y| \leq ch\).
It is generally not the case that

\[ \Delta_{p} x |x|^{\frac{p-d}{p-1}} = 0. \]

Outline of regularity proof:

1. Choose \( 0 < \alpha < \frac{(p - d)}{(p - 1)} \) and set \( v(x) = |x - y|^\alpha \)

2. Show that \( \Delta_{p} x v(x) \leq 0 \) for \( |x - y| \geq ch \) with high probability.

3. Fill in the gap \( |x - y| \leq ch \).

   1. For the variational graph \( p \)-Laplacian

      \[ |u_n(x) - u_n(y)| \leq Cn^{1/p} h \text{ for } |x - y| \leq h. \]
It is generally not the case that
\[ \Delta_p x_n |x|^{\frac{p-d}{p-1}} = 0. \]

Outline of regularity proof:
1. Choose \( 0 < \alpha < (p - d)/(p - 1) \) and set \( v(x) = |x - y|^{\alpha} \)

2. Show that \( \Delta_p x_n v(x) \leq 0 \) for \( |x - y| \geq ch \) with high probability.

3. Fill in the gap \( |x - y| \leq ch \).
   1. For the variational graph \( p \)-Laplacian
      \[ |u_n(x) - u_n(y)| \leq Cn^{1/p} h \text{ for } |x - y| \leq h. \]
   2. For the game theoretic \( p \)-Laplacian, we use a different local barrier
      \[ v(x) = |x - y|^{\alpha} + Mh_n^{\alpha} \sum_{k=1}^{\infty} \beta^k 1_{\{2|x - y| > (k - 1)h_n\}}, \text{ where } \beta < 1. \]
The local barrier

\[ v(x) = |x - y|^\alpha + M h_n^\alpha \sum_{k=1}^{\infty} \beta^k 1_{\{2|x-y| > (k-1)h_n\}} \]

exploits the form of the graph \( \infty \)-Laplacian

\[ \Delta^\infty_{x_n} u(x) = \max_{y \in x_n} w_{xy}(u(y) - u(x)) + \min_{y \in x_n} w_{xy}(u(y) - u(x)). \]
Current/Future work

1. **Fast algorithms:** Primal dual/Nesterov acceleration for pLaplacian learning (Mauricio Flores)
Current/Future work

1. **Fast algorithms:** Primal dual/Nesterov acceleration for pLaplacian learning (Mauricio Flores)

2. **Rates of convergence:**
   - PageRank algorithm (Amber Yuan)
   - Nondominated Sorting (Brendan Cook)
Current/Future work

1. **Fast algorithms:** Primal dual/Nesterov acceleration for pLaplacian learning (Mauricio Flores)

2. **Rates of convergence:**
   1. PageRank algorithm (Amber Yuan)
   2. Nondominated Sorting (Brendan Cook)

3. **Infinite labeled data:** Suppose the set of labeled data $\mathcal{O}$ grows with $n$.
   - How fast should $\mathcal{O}$ grow to ensure $p \leq d$ is well-posed?
Current/Future work

1. **Fast algorithms:** Primal dual/Nesterov acceleration for pLaplacian learning (Mauricio Flores)

2. **Rates of convergence:**
   - PageRank algorithm (Amber Yuan)
   - Nondominated Sorting (Brendan Cook)

3. **Infinite labeled data:** Suppose the set of labeled data $\mathcal{O}$ grows with $n$.
   - How fast should $\mathcal{O}$ grow to ensure $p \leq d$ is well-posed?
   - What types of models can we take for $\mathcal{O}$?
Current/Future work

1. **Fast algorithms:** Primal dual/Nesterov acceleration for pLaplacian learning (Mauricio Flores)

2. **Rates of convergence:**
   - PageRank algorithm (Amber Yuan)
   - Nondominated Sorting (Brendan Cook)

3. **Infinite labeled data:** Suppose the set of labeled data $\mathcal{O}$ grows with $n$.
   - How fast should $\mathcal{O}$ grow to ensure $p \leq d$ is well-posed?
   - What types of models can we take for $\mathcal{O}$?

4. **Soft constraint:** Extend the results to the soft constraint

$$\min_{u: X_n \to \mathbb{R}} J_p(u) + \lambda \sum_{y \in \mathcal{O}} |u(x) - g(x)|^q.$$
Outline

1. Nondominated sorting
2. Convex hull peeling
3. Semi-supervised learning
4. References


The ordering of multivariate data.

p-voltages: Laplacian regularization for semi-supervised learning on high-dimensional data.
In *Eleventh Workshop on Mining and Learning with Graphs (MLG2013)*.

Spectral clustering based on the graph p-Laplacian.

A direct verification argument for the Hamilton–Jacobi equation continuum limit of nondominated sorting.

Consistency of lipschitz learning with infinite unlabeled data and finite labeled data.
*arXiv:1710.10364*.

The game theoretic p-Laplacian and semi-supervised learning with few labels.
*arXiv:1710.10364*. 
*Semi-supervised learning.*
MIT.

A fast and elitist multiobjective genetic algorithm: NSGA-II.

Breakdown properties of location estimates based on halfspace depth and projected outlyingness.

Asymptotic behavior of lp-based Laplacian regularization in semi-supervised learning.

Maximum k-chains in planar point sets: Combinatorial structure and algorithms.


In *Neural Information Processing Systems (NIPS)*.


Computation of robust statistics: depth, median, and related measures.

Analysis of $p$-laplacian regularization in semi-supervised learning.

Convex layers: a new tool for recognition of projectively deformed point sets.

Monte carlo calculations in problems of mathematical physics.

Chain and antichain families, grids and Young tableaux.


