Learning Combinatorial Functions

Maria-Florina Balcan
Carnegie Mellon University
2-Minute Version

This talk: learning combinatorial functions from data.

Relevant to many AI, ML, multi-agent settings. E.g.:

- Learning valuation functions of users for combinatorial auctions
- Learning the influence function in information diffusion networks, e.g., social networks
- Learning value function of various coalitions of agents in cooperative game theory
2-Minute Version

In large multi-agent systems (many users, huge networks) cannot assume these core objects (e.g., valuation functions, influence functions, values of coalitions) are known in advance.

Techniques for learning them, better decision making.
This talk: learning combinatorial functions from data.

- General learnability results in a statistical setting.
- Surprising lower bounds for learning submodular functions showing unexpected structure.
- Much better upper bounds for combinatorial functions with more structure, coming from social networks & alg. game theory.
Structure of the talk

• Standard formulation for learning from data.

• Combinatorial set functions (submodular functions, coverage functions, etc).

• Learning combinatorial functions.

With connections and applications to Algorithmic Game Theory, Economics, Social Networks.
Supervised Machine Learning

• E.g., which emails are spam and which are important.

Not spam

spam

• E.g., will a news article be of interest to a user; will a car deal or sale would be of interest to a user.
Labeled Examples

Statistical / PAC learning model

Data Source

Distribution $D$ on $X$

Learning Algorithm

$(x_1, \ldots, x_k)$

Labeled Examples

$(x_1, f(x_1)),\ldots, (x_k, f(x_k))$

Alg. outputs

$g : X \rightarrow \{0,1\}$

Expert / Oracle

$f : X \rightarrow \{0,1\}$
• Algo sees \((x_1,f(x_1)),\ldots,(x_k,f(x_k))\), \(x_i\) i.i.d. from \(D\), produces \(g\).

• Want: with probability \(\geq 1-\delta\) we have \(\Pr_x[g(x)=f(x)] \geq 1-\varepsilon\)

Probably Approximately Correct (PAC)
Supervised Machine Learning

- Can learn important classes of Boolean functions:
  - linear separators
  - kernelized linear separators (i.e., linear separators in implicit higher dim features spaces)
- Successful algorithms (e.g., Boosting, SVM).

This talk: how well we can learn important classes of combinatorial real-valued fns (e.g., submodular functions).
Structure of the talk

• Standard formulation for learning from data.

• Combinatorial set functions (submodular functions, coverage functions, etc).

• Learning combinatorial functions.

  With connections and applications to Algorithmic Game Theory, Economics, Social Networks.
Combinatorial Set Functions

- Function over sets.
  - e.g., value on some set of items in a store.

- Formally, ground set \( V = \{1, 2, \ldots, n\} \).

  set-functions \( f : 2^V \rightarrow \mathbb{R} \)

This talk: focus on

**Monotone:** \( f(S) \leq f(T), \ \forall \ S \subseteq T \)

**Non-negative:** \( f(S) \geq 0, \ \forall \ S \subseteq V \)
Submodular functions

- $V=\{1,2, \ldots, n\}$; set-function $f : 2^V \rightarrow \mathbb{R}$ submodular if

  For all $S, T \subseteq V$: $f(S)+f(T) \geq f(S \cap T)+f(S \cup T)$

- Equivalent decreasing marginal return:

  For $T \subseteq S$, $x \notin S$, $f(T \cup \{x\}) - f(T) \geq f(S \cup \{x\}) - f(S)$
Submodular functions

- \( V = \{1, 2, \ldots, n\} \); set-function \( f : 2^V \rightarrow \mathbb{R} \) submodular if

\[
\text{For } T \subseteq S, \ x \notin S, \ f(T \cup \{x\}) - f(T) \geq f(S \cup \{x\}) - f(S)
\]

E.g.,

Large improvement

Small improvement
Coverage and Reachability Functions

- **Coverage function:** Let $A_1, ..., A_n$ be sets. For each $S \subseteq V$, let $f(S) = |\bigcup_{j \in S} A_j|$

- **Reachability function:** $f(S) = \#$ nodes reachable from $S$.

E.g., in a network, $A_s$ nodes reachable from $s$
Coverage and Reachability Functions

- Reachability function: \( f(S) = \# \) nodes reachable from \( S \).
  
  E.g., in a network, \( A_s \) nodes reachable from \( s \)

Diminishing Returns

- Marginal value of \( x \) given \( S \) is \( \# \) number of new nodes that \( x \) can reach, but cannot be reached from any of the nodes in \( S \).

- \( T \subset S, x \notin S \), more chance reach new nodes when adding \( x \) to \( T \), than when adding \( x \) to \( S \).
Reachability function is submodular

Marginal value of $x = \#$ new nodes reachable from $x$.

\[
T = \{s_2\}, \quad f(T) = 5 \\
\geq \frac{3}{f(T \cup \{x\}) - f(T)} \geq \frac{2}{f(S \cup \{x\}) - f(S)}
\]

\[
S = \{s_1, s_2\}, \quad f(S) = 8
\]
Probabilistic Reachability Functions

- Given a distribution over graphs

\[ f(S) = \mathbb{E}_G[\text{\# reachable from } S|G] \text{ also submodular.} \]
Submodular functions

More examples:

- **Concave Functions** Let $h : \mathbb{R} \to \mathbb{R}$ be concave. For each $S \subseteq V$, let $f(S) = h(|S|)$

- **Vector Spaces** Let $V = \{v_1, \ldots, v_n\}$, each $v_i \in \mathbb{R}^n$. For each $S \subseteq V$, let $f(S) = \text{rank}(V[S])$

- **Cut Function in a Graph** Let $f(S) = \# \text{ of edges between } S \text{ and } V \setminus S$.

This talk: focus on

**Monotone:** $f(S) \leq f(T), \forall S \subseteq T$

**Non-negative:** $f(S) \geq 0, \forall S \subseteq V$
Combinatorial Functions

- A lot of work on Optimization Problems, including optimization of Submodular Functions.

  Traditionally: Optimization, operations research

Most recently

- Algorithmic Game Theory [Lehman-Lehman-Nisan’01], ....
- Machine Learning [Bilmes’03] [Guestrin-Krause’07], ...
- Social Networks [Kleinberg-Kempe-Tardos’03]

- This talk: learning them from data.
Learning Combinatorial Functions

Combinatorial fns such as submodular fns commonly used in Economics and AGT to model valuation functions of customers.

Supermarket chain
• $V$ = all the items you sell.
• $f(S)$ = valuation on set of items $S$.

$f(\text{Chex}) \rightarrow \mathbb{R}$
Learning Combinatorial Functions

In online social networks and other networks:

• How much influence various sets of individuals/nodes have?

• If a given set of individuals are the initial adaptors of an idea or product, what is the expected number of followers?
Learning Combinatorial Functions

Influence Function in Social Networks

- $V =$ set of nodes.
- $f(S) =$ expected number of nodes $S$ will influence.

$f$ is a probabilistic reachability function in classic diffusion models (e.g., independent cascade model, random threshold model) [Kleinberg-Kempe-Tardos'03]

Past Work

Assume an explicit model on how info spreads; use it to estimate the influence func.

Could be mis-specified.

Our Work

Learn the influence function directly from data
Learning Combinatorial Functions

Influence Function in Networks

epidemiology: influenza spread

biology:
gene expression cascade

cybersecurity: computer virus spread
Learning Combinatorial Functions

General Learnability Results for Submodular Fns

• Upper & lower bounds on their intrinsic complexity.
  • Implications to Alg. Game Theory, Economics, Discrete Optimization, Matroid Theory.

• Highlights importance of beyond worst case analysis.

Better Results for Cases with More Structure

Large Scale Application to Social Networks
Statistical learning model

Data Source

Distribution \( D \) on \( 2^{[n]} \)

Learning Algorithm

Labeled Examples

Alg. outputs

\( f : 2^{[n]} \to \mathbb{R}_+ \)

\( g : 2^{[n]} \to \mathbb{R}_+ \)
PMAC model for learning real valued functions

[Balcan&Harvey, STOC 2011 & Nectar Track, ECML-PKDD 2012, SICOMP 2018]

Data Source

Distribution D on $2^{[n]}$

Learning Algorithm

Labeled Examples

$(S_1, f(S_1)), ..., (S_k, f(S_k))$

Alg.outputs

$g : 2^{[n]} \rightarrow \mathbb{R}_+$

Expert / Oracle

$f : 2^{[n]} \rightarrow \mathbb{R}_+$

- Algo sees $(S_1, f(S_1)), ..., (S_k, f(S_k))$, $S_i$ i.i.d. from D, produces $g$.
- With probability $\geq 1 - \delta$ we have $\Pr_S[g(S) \leq f(S) \leq \alpha g(S)] \geq 1 - \epsilon$

Probably Mostly Approximately Correct
PMAC model for learning real valued functions

[Balcan&Harvey, STOC 2011 & Nectar Track, ECML-PKDD 2012, SICOMP 2018]

Data Source

Distribution
\[ D \text{ on } 2^n \]

Learning
Algorithm

Labeled Examples
\[(S_1,f(S_1)),\ldots,(S_k,f(S_k))\]

Expert / Oracle

Alg.outputs

\[ g : 2^n \rightarrow \mathbb{R}_+ \]

\[ f : 2^n \rightarrow \mathbb{R}_+ \]

• Algo sees \((S_1,f(S_1)),\ldots,(S_k,f(S_k))\), \(S_i\) i.i.d. from \(D\), produces \(g\).
• With probability \(\geq 1-\delta\) we have \(\Pr_s[g(S) \leq f(S) \leq \alpha g(S)] \geq 1-\epsilon\)

\[ \alpha = 1 \] , recover PAC model.
A General Upper Bound

**Theorem:** Can PMAC-learn submodular functions in poly time and poly samples with an approx. factor $\alpha = O(\sqrt{n})$.

**Key Idea:**

- Target $f$ submodular, then $f^2$ can be approx. within $n$ by a linear function.

$$f^2(S) \leq g(S) \leq n f^2(S), \forall S$$

- Reduction to learning a linear separator, over an $n+1$ dimensional feature space.

we have a feature for each item in the ground set and one additional feature related to the value of the function

Given $(S_i, f(S_i)), i = 1, ..., m$, $(\chi(S_i), f^2(S), +)$ and $(\chi(S_i), n f^2(S), -)$

are linearly separable in $\mathbb{R}^{n+1}$.
A General Lower Bound

**Theorem:** No algo can PMAC learn the class of submodular fns with approx. factor $\tilde{o}(n^{1/3})$.

- Even if value queries allowed; even for rank fns of matroids.

**Corollary:** Gross substitute fns do **not** have a concise, approximate representation.

- $f$ satisfies *gross substitutes* if raising prices on some items does not remove any items from optimal bundle whose price did not change.
A General Lower Bound

**Theorem:** No algo can PMAC learn the class of submodular fns with approx. factor $\tilde{O}(n^{1/3})$.

- Even if value queries allowed; even for rank fns of matroids.

**Corollary:** Gross substitute fns do **not** have a concise, approximate representation.

Surprising answer to open question in *AGT & Economics of* 

Paul Milgrom  
Noam Nisan
A General Lower Bound

Theorem
No algorithm can PMAC learn the class of non-neg., monotone, submodular fns with an approx. factor \( \tilde{o}(n^{1/3}) \).

Idea:
A new hard family of matroid rank functions with extremal properties.

\[
\begin{align*}
\text{High} &= n^{1/3} \\
\text{Low} &= \log^2 n \\
L &= n^\log\log n
\end{align*}
\]
**Partition Matroids**

\[ A_1, A_2, ..., A_k \subseteq V=\{1,2, ..., n\}, \text{ all disjoint; } u_i \leq |A_i|-1 \]

\[ \text{Ind}=\{I: |I \cap A_j| \leq u_j, \text{ for all } j \} \]

Then \((V, \text{Ind})\) is a matroid.

If sets \(A_i\) are not disjoint, then \((V,\text{Ind})\) might not be a matroid.

- E.g., \(n=5, A_1=\{1,2,3\}, A_2=\{3,4,5\}, u_1=u_2=2.\)
- \(\{1,2,4,5\}\) and \(\{2,3,4\}\) both maximal sets in \(\text{Ind}\); do not have the same cardinality.
Almost partition matroids

\(k=2, \ A_1, A_2 \subseteq V\) (not necessarily disjoint); \(u_i \leq |A_i| - 1\)

\(\text{Ind} = \{I: |I \cap A_j| \leq u_j , \ |I \cap (A_1 \cup A_2)| \leq u_1 + u_2 - |A_1 \cap A_2|\}\)

Then \((V, \text{Ind})\) is a matroid.
Almost partition matroids

More generally

\[ A_1, A_2, \ldots, A_k \subseteq V = \{1,2, \ldots, n\}, \quad u_i \leq |A_i| - 1; \quad f : 2^{[k]} \rightarrow \mathbb{Z} \]

\[ f(J) = \sum_{j \in J} u_j + |A(J)| - \sum_{j \in J} |A_j|, \quad \forall \ J \subseteq [k] \]

\[ \text{Ind} = \{ I : |I \cap A(J)| \leq f(J), \ \forall \ J \subseteq [k] \} \]

Then \((V, \text{Ind})\) is a matroid (if nonempty).

Rewrite \(f\), \(f(J) = |A(J)| - \sum_{j \in J} (|A_j| - u_j), \quad \forall \ J \subseteq [k] \)
Almost partition matroids

More generally  \( f : 2^[k] \to \mathbb{Z} \)

\[
f(J) = |A(J)| - \sum_{j \in J} (|A_j| - u_j), \quad \forall \ J \subseteq [k]
\]

\[
\text{Ind} = \{ I : |I \cap A(J)| \leq f(J), \quad \forall \ J \subseteq [k] \}
\]

Then \((V, \text{Ind})\) is a matroid (if nonempty).

\[
f : 2^[k] \to \mathbb{Z}, f(J) = |A(J)| - \sum_{j \in J} (|A_j| - u_j), \quad \forall \ J \subseteq [k]; \ u_i \leq |A_i| - 1
\]

Note: This requires \(k \leq n\) (for \(k > n\), \(f\) becomes negative)

More tricks to allow \(k = n^{\log \log n}\).
Learning submodular valuations

Theorem

No algorithm can PMAC learn the class of non-neg., monotone, submodular fns with an approx. factor $\tilde{\Theta}(n^{1/3})$.

Worst Case Analysis 😊
Moral: Exploit Additional Structure

• Product distribution.
  [Balcan-Harvey, STOC’11 & SICOMP’18] [Feldman-Vondrak, FOCS’13]

• Bounded Curvature (i.e., almost linear)
  [Iyer-Jegelka-Bilmes, NIPS’13]

• Learning valuation fns from AGT and Economics.
  [Balcan-Constantin-Iwata-Wang, COLT ’12]
  [Badanidiyuru-Dobzinski-Fu- Kleinberg-Nisan-Roughgarden, SODA’12]

• Learning influence fns in information diffusion networks
  [Du, Liang, Balcan, Song, ICML’14; NIPS’14]

• Learning values of coalitions in cooperative game theory
  [Balcan, Procacia, Zick, IJCAI’15]
Learning Valuation Functions

- Target dependent learnability for classes of valuation functions have a succinct description.
  
  [Balcan-Constantin-Iwata-Wang, COLT 2012]

Well-studied subclasses of subadditive valuations.

[Algorithmic game theory and Economics]

Additive $\subseteq$ OXS $\subseteq$ Submodular $\subseteq$ XOS $\subseteq$ Subadditive

[Sandholm'99] [Lehman-Lehman-Nisan'01]
XOS valuations

Functions that can be represented as a MAX of SUMs.

\[ g(\{1,2\}) = 16 \]
\[ g(\{2,3\}) = 10 \]
\[ g(\{1,2,3\}) = 16 \]
Target dependent Upper Bound for XOS

**Theorem:** (Polynomial number of Sum trees)

\( O(R^\epsilon) \) approximation in time \( O(n^{1/\epsilon}) \).

**Main Idea:**

- Target approx within \( O(R^\epsilon) \) by a linear function over \( O(n^{1/\epsilon}) \) feature space (one feature for each \( n^{1/\epsilon} \)-tuple of items).

- Reduction to learning a linear separator in a higher dim. feature space.

Highlights importance of considering the complexity of the target function.
Learning Influence Functions in Information Diffusion Networks

\[ \text{(Du, Liang, Balcan, Song, ICML 2014, NIPS'14)} \]

**Influence Function in Networks**

- \( V = \text{set of nodes.} \)
- \( f(S) = \text{expected number of nodes } S \text{ will influence.} \)

**Fact:** In classic diffusion models (discrete time independent cascade model/random threshold model, continuous time analogues, etc), the influence function is coverage function. [Kleinberg-Kempe-Tardos'03]

\[ f(S) = E_G[\# \text{ reachable from } S|G] \]

probabilistic reachability fnc
Discrete-time independent cascade model

- **Input**: directed graph, each edge has a weight $w \in [0,1]$
- **Cascade generative process for a source set $S$**
  - presence of edge is sampled independently according to $w$
  - influenced nodes are those reachable from at least one of the source nodes in the resulting “live edge graph”
- **Influence of $S$** is expected number of nodes influenced under this random process
Learning Influence Functions in Information Diffusion Networks

[Du, Liang, Balcan, Song, ICML 2014, NIPS’14]

Fact: in classic diffusion models, the influence function is a coverage function.

\[ f(S) = \mathbb{E}_G[\# \text{reachable from } S|G] \]
probabilistic reachability fnc

• Note 1: Do not know better guarantees for efficient algorithms if access only to value queries.

• Note 2: Do better theoretically and empirically, if have access to information diffusion traces or cascades.
Learning Influence Functions based on information propagation traces (cascades)
Another cascade
Learning the influence function

**Input:** past influence cascades \{ (S_1, I_1), (S_2, I_2), ..., (S_m, I_m) \}.

**Goal:** learn Influence function \( f(S) = \mathbb{E}[\#\text{influenced}(S)] \).

**Assumption:** \( f(S) \) is a probabilistic coverage function.

I.e., there is a distribution \( p_R \) over reachability matrices \( R \) s.t.:

\[
f(S) = \mathbb{E}_{R \sim p_R}[\#\text{influenced}(S|R)]
\]

\[
| \{ j : R_{sj} = 1 \text{ for some } s \in S \} |
\]

\( R_{sj} = 1 \text{ if } s \text{ can reach } j, \)
\( R_{sj} = 0 \text{ otherwise.} \)
Learning the influence function

**Input:** past influence cascades \( \{(S_1, I_1), (S_2, I_2), \ldots, (S_m, I_m)\} \).

**Goal:** learn Influence function \( f(S) = \mathbb{E}[\#\text{influenced}(S)] \).

**Idea:** \( f(S) = \sum_j f_j(S) \), where \( f_j(S) = \Pr_{R \sim p_R} (j \text{ is influenced by } S) \).

For each \( j \), will learn \( \hat{f}_j(S) \). Output \( \sum_j \hat{f}_j(S) \).

**Algorithm for learning \( f_j \)**

Use “random kitchen sink” approach:
- choose random binary vectors \( v_1, v_2, \ldots, v_K \) from \( q \).
- Parametrize \( \hat{f}_j(S) \) as \( \sum_i w_i \cdot I[\langle I_S, v_i \rangle \geq 1] (\sum_i w_i \leq 1, w_i \geq 0) \)

Learn weights via maximum conditional likelihood.
Influence estimation in real data

[Du, Liang, Balcan, Song, ICML 2014, NIPS’14]

- Memetracker Dataset, blog data cascades: “apple and jobs”, “tsunami earthquake”, “william kate marriage”
Conclusions

Learnability of submodular, other combinatorial fns

• Can model problems of interest to multi-agent systems.
• Very strong lower bounds in the worst case.
• Much better learnability results for classes with additional structure.
Conclusions

Learnability of submodular functions

• Very strong lower bounds in the worst case.
• Highlight the importance of considering the complexity of the target function.

Open Questions:

• Exploit complexity of target for better approx guarantees. [for learning and optimization]
  • What is a natural description language for submodular fns?
• Other interesting applications.