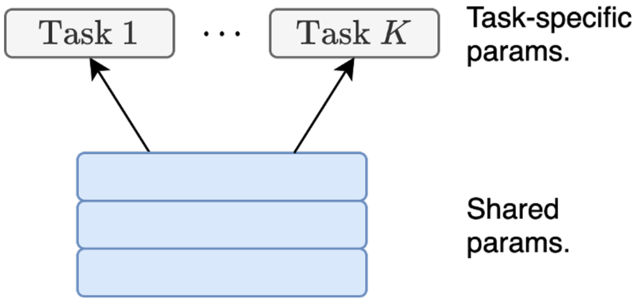
Multi-Task Learning as a Bargaining Game

A. Navon[1] A. Shamsian[1] I. Achituve[1] H. Maron[2] K. Kawaguchi[3]
G. Chechik[1,2] E. Fetaya[1]

[1]Bar-Ilan University [2]NVIDIA Research [3]National University of Singapore

Multi-task learning (MTL)

- Solving several learning problems simultaneously.
- For example, in autonomous vehicles: object detection, depth estimation, velocity estimation.
- The standard approach:
  - All tasks share an encoder (feature extractor).
  - Each task has a task-specific head.



Why MTL?

Compared to having several single-task (STL) models, MTL

- ***Reduces computation costs:*** By using a shared trunk we can reduce computation at inference time.
- ***Improves generalization and data efficiency:*** Tasks regularize each other.

A common approach to MTL optimization

Most MTL optimization algorithms follow:

- Calculate per-task gradients $g_i, i = 1, ..., K$.
- Combine gradients into a joint direction Δ using aggregation alg. $\mathcal{A}$.
- Update the parameters according to $\Delta = \mathcal{A}(g_1, ..., g_k)$.

A common approach to MTL optimization

Most MTL optimization algorithms follow:

- Calculate per-task gradients $g_i, i = 1, ..., K$.
- Combine gradients into a joint direction Δ using aggregation alg. $\mathcal{A}$.
- Update the parameters according to $\Delta = \mathcal{A}(g_1, ..., g_k)$.

The challenge: How to combine gradients and alleviate task interference?

- Gradients may conflict in directions or have large differences in magnitudes.
- Not clear how to combine the gradients.

A common approach to MTL optimization

Most MTL optimization algorithms follow:

- Calculate per-task gradients $g_i, i = 1, ..., K$.
- Combine gradients into a joint direction Δ using aggregation alg. $\mathcal{A}$.
- Update the parameters according to $\Delta = \mathcal{A}(g_1, ..., g_k)$.

The challenge: How to combine gradients and alleviate task interference?

- Gradients may conflict in directions or have large differences in magnitudes.
- Not clear how to combine the gradients

Our solution: A novel and principled MTL Algorithm, by viewing the gradient aggregation step as a Bargaining game.

Background: Bargaining games

- K players, each with their own utility function $u_i : A \cup \{D\} \to \mathbb{R}$.
- A is set of agreement points and D the disagreement point.
- The players must find a point they agree upon or default to D.

Under mild conditions the game has a unique solution that satisfies (Nash, 1953):

- Pareto optimality.
- Symmetry.
- Independence of irrelevant alternatives.
- Invariance to affine transformation.

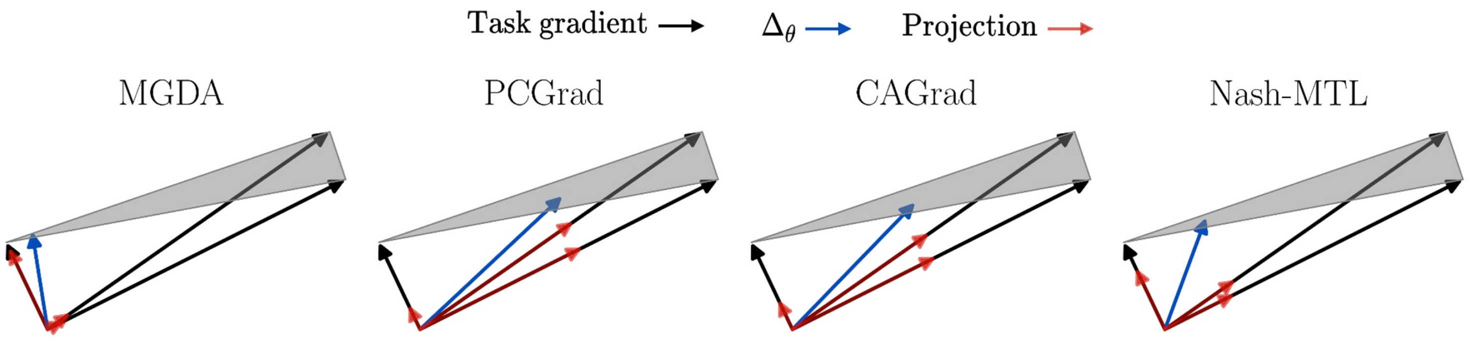
This unique solution is called the ***Nash bargaining solution***.

Our approach: Nash-MTL

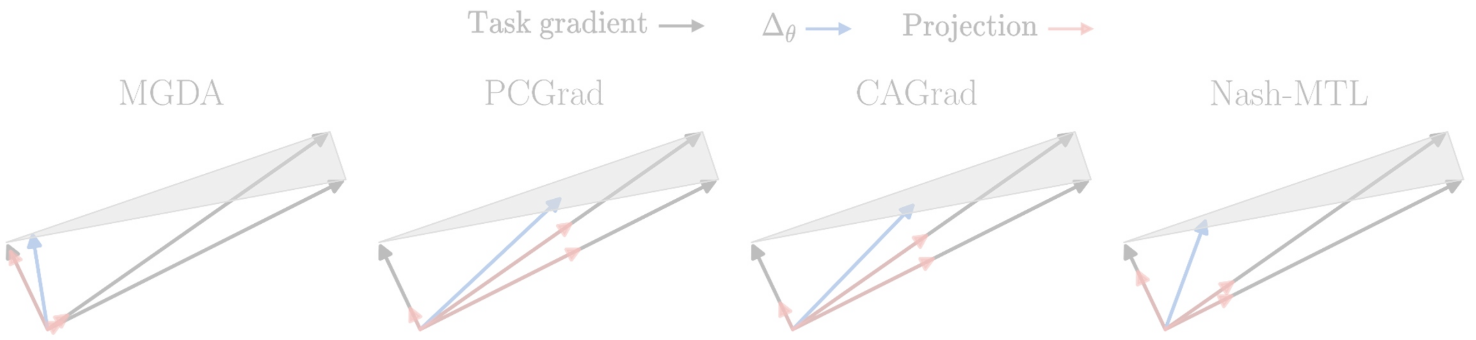
- Given an MTL problem with parameters θ.
- Search for update $\Delta\theta$ in an ϵ-ball around zero.
- Define the utility for task i as a directional derivative $u_i(\Delta\theta) = \Delta\theta^T g_i$.
- Denote G the matrix whose columns are the gradients g_i .

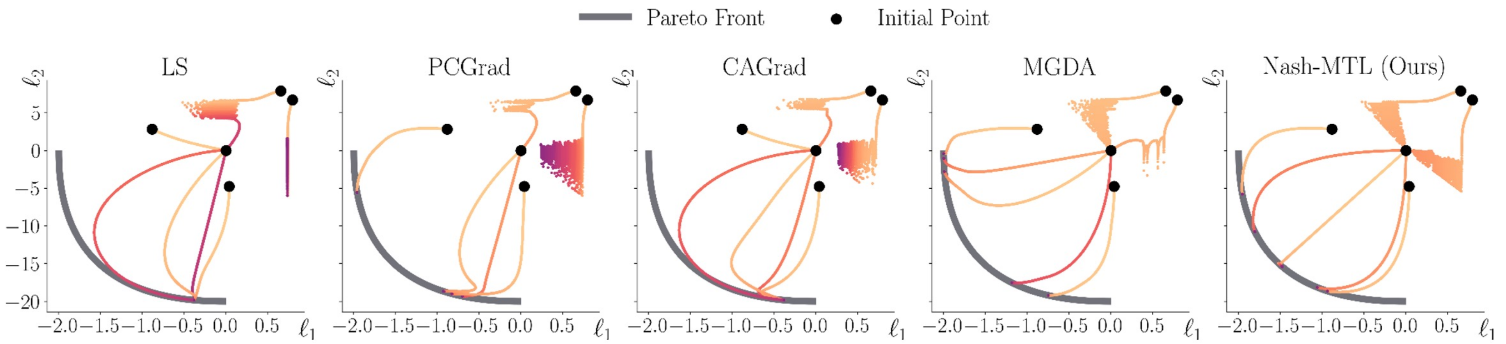
Claim: The *Nash bargaining solution* for our problem is given by $\Delta\theta = \sum_i \alpha_i g_i$ s.t. $G^T G\alpha = 1/\alpha$ where $1/\alpha$ is taken element-wise.

Illustrative example



Illustrative example





Nash-MTL

Analysis

We prove the sequence generated by our method converges to a Pareto optimal (stationary) point in the (non-convex) convex case.

Nash-MTL

Analysis

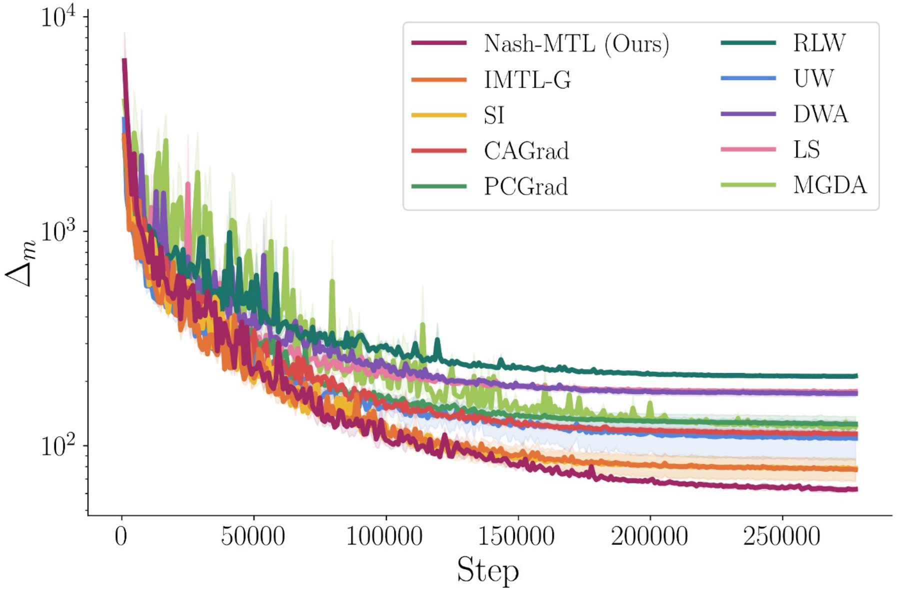
We prove the sequence generated by our method converges to a Pareto optimal (stationary) point in the (non-convex) convex case.

Approximation and practical speedup

- The problem is solved at each iteration: optimization must be efficient.
- We cast non-convex problem as a sequence of convex optimization problems.
- For additional speedup, we apply Nash-MTL once every N optimization steps.

Results – Multi-Task Regression on Graphs

QM9 dataset: Predict properties of molecules (11 tasks).



	MR ↓	$\mathbf{\Delta_m}$**%** ↓
LS	6.8	177.6 ± 3.4
SI	4.0	77.8 ± 9.2
RLW	8.2	203.8 ± 3.4
DWA	6.4	175.3 ± 6.3
UW	5.3	108.0 ± 22.5
MGDA	5.9	120.5 ± 2.0
PCGrad	5.0	125.7 ± 10.3
CAGrad	5.7	112.8 ± 4.0
IMTL-G	4.7	77.2 ± 9.3
Nash-MTL	**2.5**	**62.0 ± 1.4**

Results – Scene Understanding

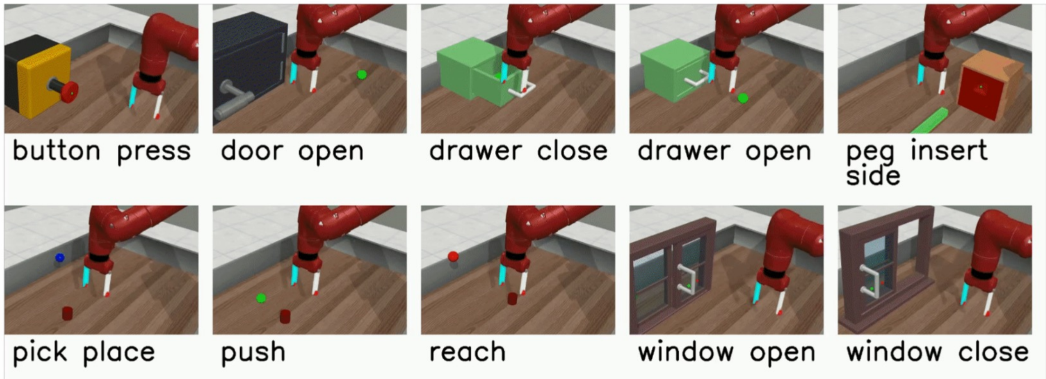
NYUv2 dataset with 3 tasks: Semantic segmentation, depth and surface normal.

	Segmentation		Depth		Surface Normal						
					Angle Distance ↓		Within t° ↑				
	mIoU ↑	Pix Acc ↑	Abs Err ↓	Rel Err ↓	Mean	Median	11.25	22.5	30	**MR** ↓	$\mathbf{\Delta}$**m%** ↓
STL	38.30	63.76	0.6754	0.2780	25.01	19.21	30.14	57.20	69.15		
LS	39.29	65.33	0.5493	0.2263	28.15	23.96	22.09	47.50	61.08	8.11	5.59
SI	38.45	64.27	0.5354	0.2201	27.60	23.37	22.53	48.57	62.32	7.11	4.39
RLW	37.17	63.77	0.5759	0.2410	28.27	24.18	22.26	47.05	60.62	10.11	7.78
DWA	39.11	65.31	0.5510	0.2285	27.61	23.18	24.17	50.18	62.39	6.88	3.57
UW	36.87	63.17	0.5446	0.2260	27.04	22.61	23.54	49.05	63.65	6.44	4.05
MGDA	30.47	59.90	0.6070	0.2555	**24.88**	**19.45**	**29.18**	**56.88**	**69.36**	5.44	1.38
PCGrad	38.06	64.64	0.5550	0.2325	27.41	22.80	23.86	49.83	63.14	6.88	3.97
GradDrop	39.39	65.12	0.5455	0.2279	27.48	22.96	23.38	49.44	62.87	6.44	3.58
CAGrad	39.79	65.49	0.5486	0.2250	26.31	21.58	25.61	52.36	65.58	3.77	0.20
IMTL-G	39.35	65.60	0.5426	0.2256	26.02	21.19	26.2	53.13	66.24	3.11	−0.76
Nash-MTL	**40.13**	**65.93**	**0.5261**	**0.2171**	25.26	20.08	28.4	55.47	68.15	**1.55**	**−4.04**

Results – Reinforcement Learning

MT10 from the Meta-world benchmark: 10 tasks.

	Success $\pm$ SEM
STL SAC	0.90 ± 0.032
MTL SAC	0.49 ± 0.073
MTL SAC + TE	0.54 ± 0.047
MH SAC	0.61 ± 0.036
SM	0.73 ± 0.043
CARE	0.84 ± 0.051
PCGrad	0.72 ± 0.022
CAGrad	0.83 ± 0.045
Nash-MTL	$\mathbf{0.91 \pm 0.031}$



Conclusion

- We presented Nash-MTL, a novel and principled approach for multitask learning.
- We framed the gradient combination step in MTL as a bargaining game and use the Nash bargaining solution to find the optimal update direction.
- We provided extensive theoretical and empirical analysis.
- Our code is publicly available at: https://github.com/AvivNavon/nash-mtl