Constrained Optimization for Neural Networks: a Mini-Lesson

Kevin Pedro (FNAL)

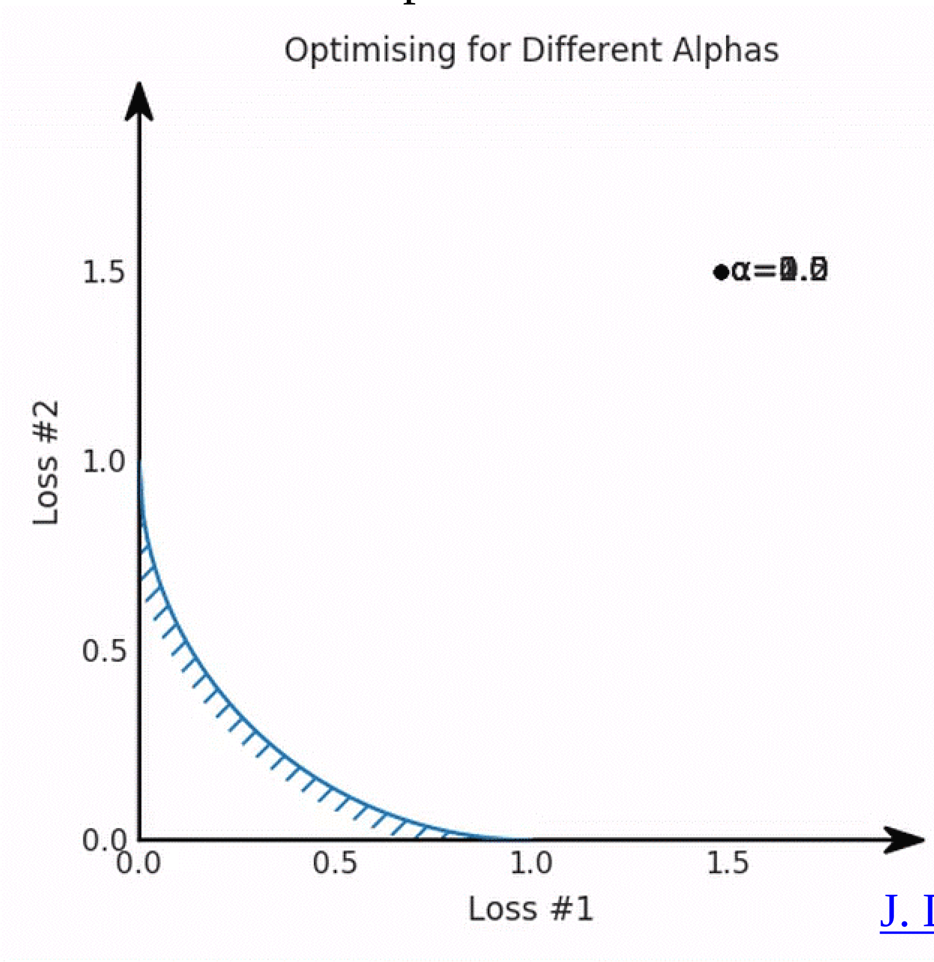
December 17, 2023

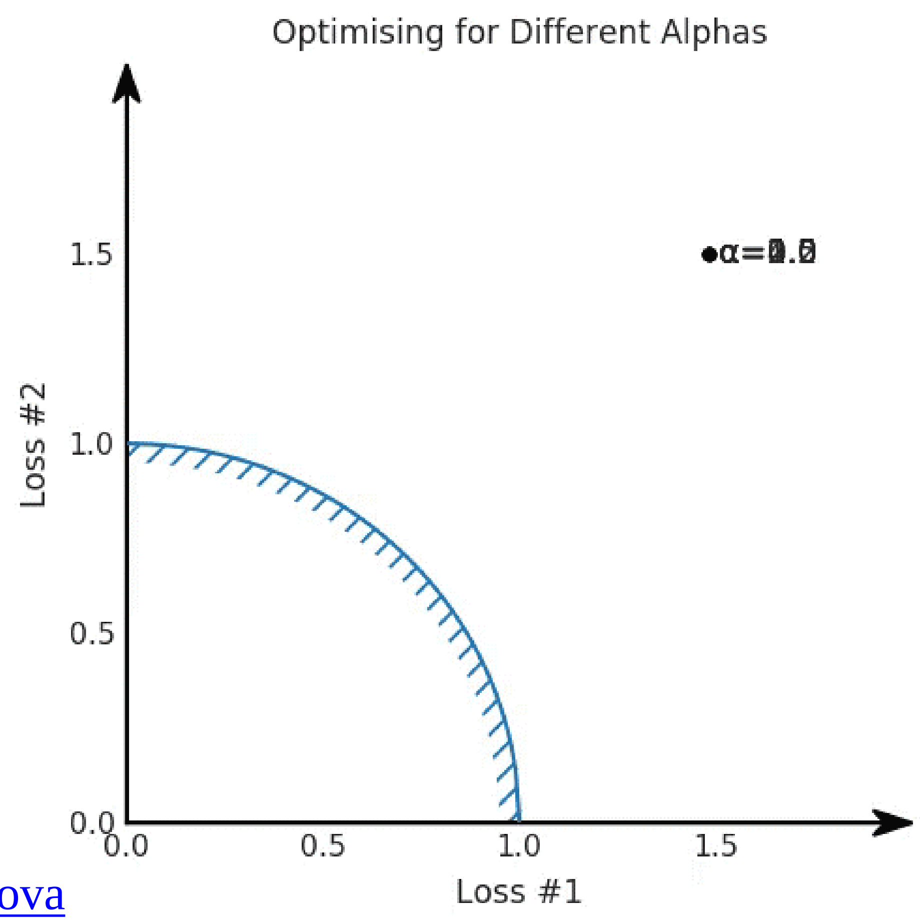
Multiple Loss Terms

- Choice of loss function is essential for training any neural network
- Many options discussed this week:
  - (Binary) crossentropy, L1 (absolute error), L2 (squared error), Wasserstein metric (earth mover's distance), maximum mean discrepancy, divergences, etc.
- We often include additional loss terms for several reasons:
  - Incorporate domain knowledge, i.e. physics
  - Account or correct for unwanted effects
- Simplest approach: $\mathcal{L} = f(\theta) + \lambda g(\theta)$
  - λ (relative weight) treated as a hyperparameter:
    guess its value based on magnitudes of f and g, how much you want to control an effect, etc.
  - In generalize, N–1 λ parameters for N loss terms
- What can go wrong with this approach?

Pareto Fronts

- *Pareto optimal solution*: any change to improve one criterion will degrade another criterion
- *Pareto front*: set of all Pareto optimal solutions
- Which of these pareto fronts will work well when choosing an arbitrary value for α (λ)?





[J. Degrave & I. Korshunova]

Challenges in Multi-Objective Optimization

- Pareto front shape is *unknown*
  - Depends not just on loss functions, but also on training data, network architecture & weights, etc.
  - May be convex in some areas and concave in others
- Unclear relationship between λ values and loss values at Pareto front
  - Hard to control and understand the behavior
- Underlying problem: *no mathematical guarantee* to be able to optimize for two things at once!
- Instead: optimize for one thing with *constraints* on others
  - Lagrange multiplier method, introduced in 1804

Basic Differential Method of Multipliers

- Lagrange multiplier approach: combined loss is $\mathcal{L} = f(\theta) + \lambda(\varepsilon - g(\theta))$
  - o ε is the constraint on loss term g
  - o λ is now a *learnable* parameter

- Apply gradient descent:

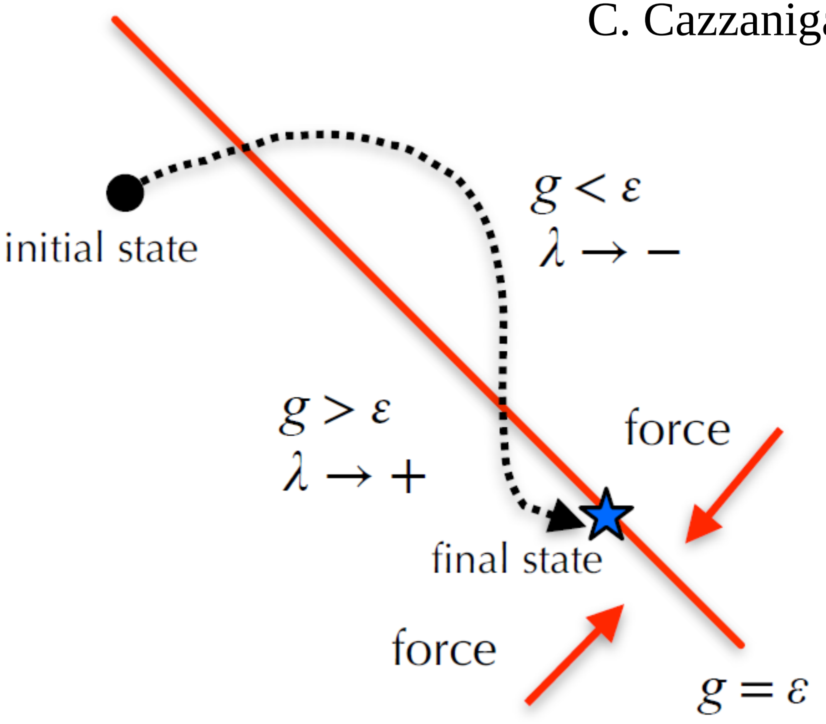
  $\theta' = -\,{}^{\partial\mathcal{L}}\!/_{\partial\theta} = -\,{}^{\partial f}\!/_{\partial\theta} + \lambda\,{}^{\partial g}\!/_{\partial\theta}$

  $\lambda' = -\,{}^{\partial\mathcal{L}}\!/_{\partial\lambda} = -\,g(\theta) + \varepsilon$

  - o Critical points of this system are *saddle points* rather than minima → no convergence

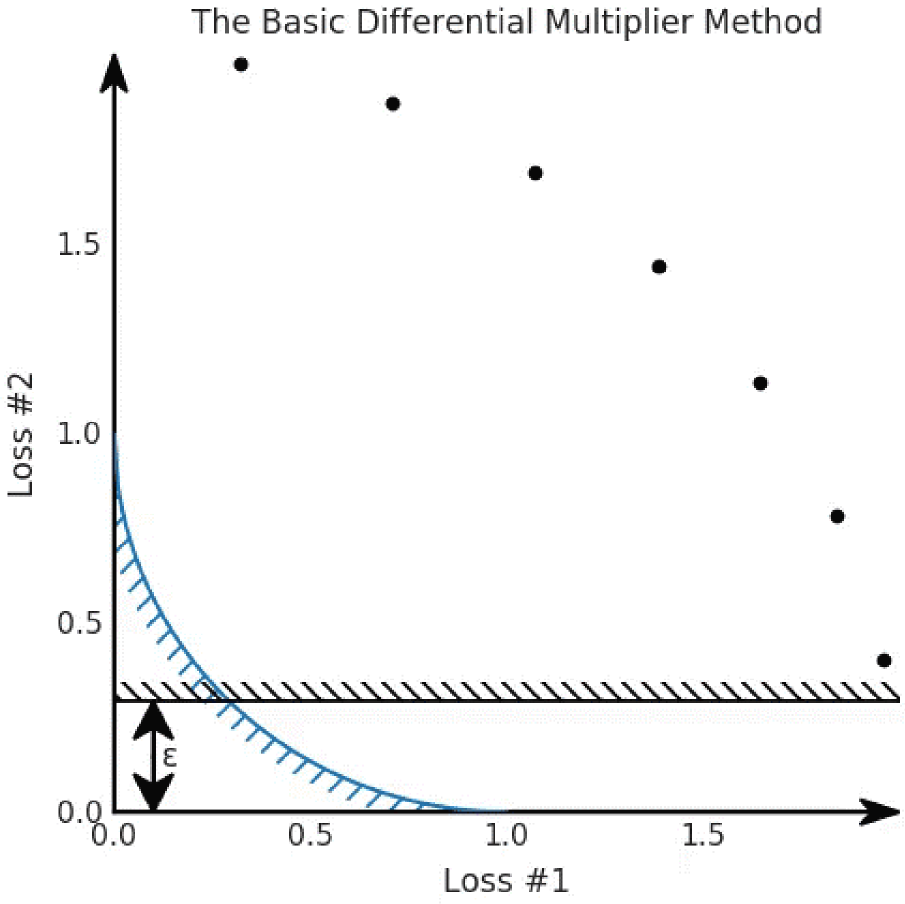
- *Differential* method: $\mathcal{L} = f(\theta) - \lambda(\varepsilon - g(\theta))$
  - o Gradient *ascent* in λ → ensure critical points are attractors
    $\theta' = -\,{}^{\partial f}\!/_{\partial\theta} - \lambda\,{}^{\partial g}\!/_{\partial\theta}$
    $\lambda' = g(\theta) - \varepsilon$
  - o Does it work?

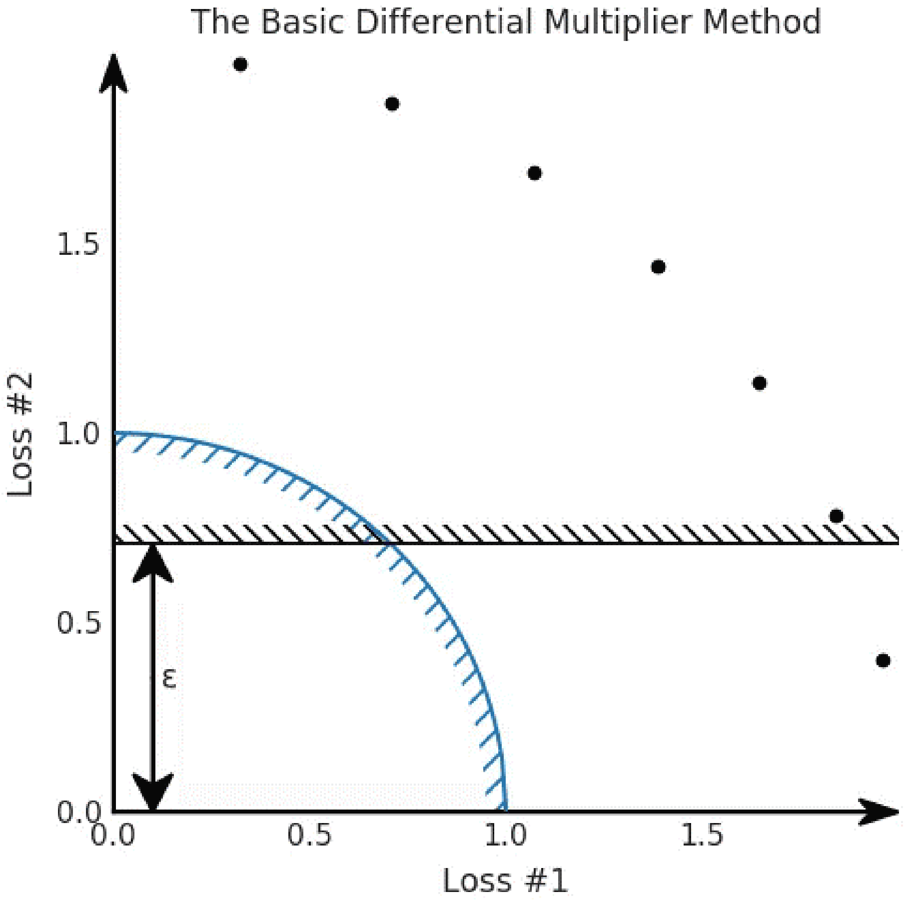
C. Cazzaniga



Pareto Fronts with BDMM

- Converges to ε in convex case
- Oscillates around ε in concave case → no convergence

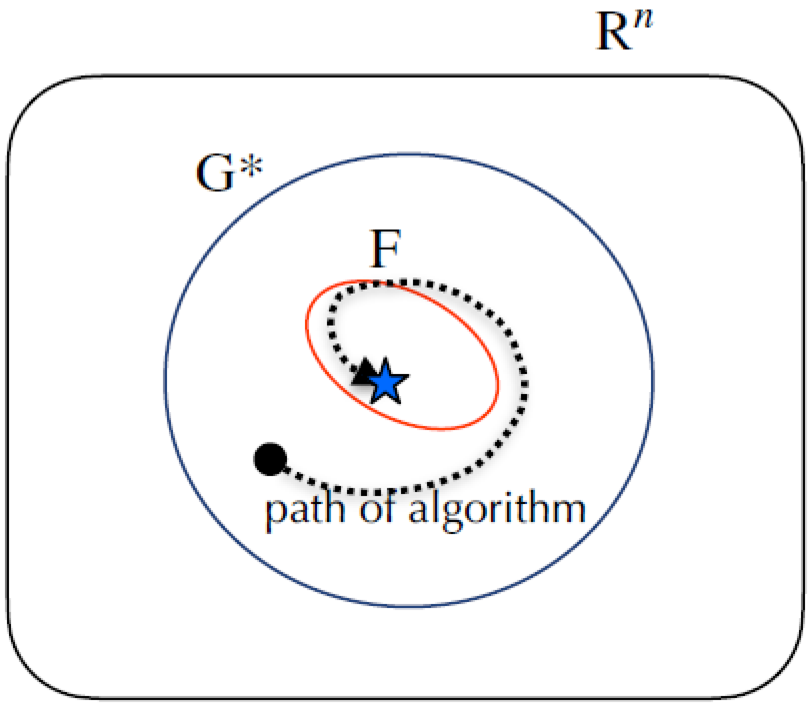




J. Degrave & I. Korshunova

Modified Differential Method of Multipliers

- We can obtain an equation of motion from the BDMM gradient descent system:
  $\theta'' + ({}^{\partial^2 f}\!/\!_{\partial\theta^2} + \lambda\,{}^{\partial^2 g}\!/\!_{\partial\theta^2})\theta' + \lambda'\,{}^{\partial g}\!/\!_{\partial\theta} = 0$
  $\rightarrow$
  $\theta'' + A(\theta, \lambda)\theta' + (g(\theta) - \varepsilon)\,{}^{\partial g}\!/\!_{\partial\theta} = 0$
  - o A can be identified as the *damping matrix*
- LaSalle's invariance principle:
  - o Consider region G as an open subset of R^n, and $F \subset G^*$ at equilibrium:
    $F = \{ \theta, \lambda \mid \theta' = 0; \lambda' = 0; \theta, \lambda \in G^* \}$
  - o If:
    a. A is positive definite in G
    b. θ, λ are bounded in G
    c. F is non-empty
  - o Then: θ, λ approach F as $t \rightarrow \infty$



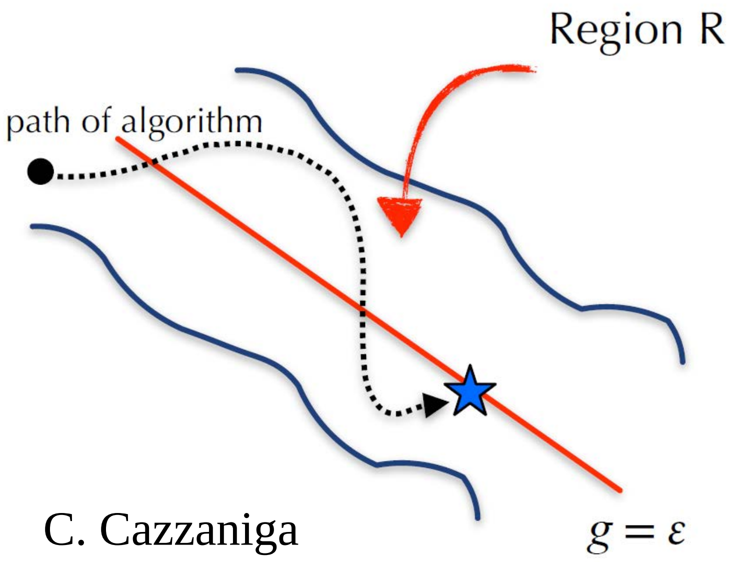
C. Cazzaniga

Modified Differential Method of Multipliers

- How to ensure *A* is positive definite?
  - A quadratic penalty term can be added: $\mathcal{L} = f(\theta) - \lambda(\varepsilon - g(\theta)) + \delta(\varepsilon - g(\theta))^2$
  - Now $A = {}^{\partial^2 f}\!/_{\partial\theta^2} + \lambda\,{}^{\partial^2 g}\!/_{\partial\theta^2} + 2\delta({}^{\partial g}\!/_{\partial\theta})^2 + 2\delta g(\theta)\,{}^{\partial^2 g}\!/_{\partial\theta^2}$
    - Theorem: $\exists\ \delta^* > 0$ such that for $\delta > \delta^*$, *A* is positive definite at the minimum
    - *A* is continuous, so *A* must also be positive definite in a region R around the minimum

➢ If system starts in R and is bounded in R, will always converge!

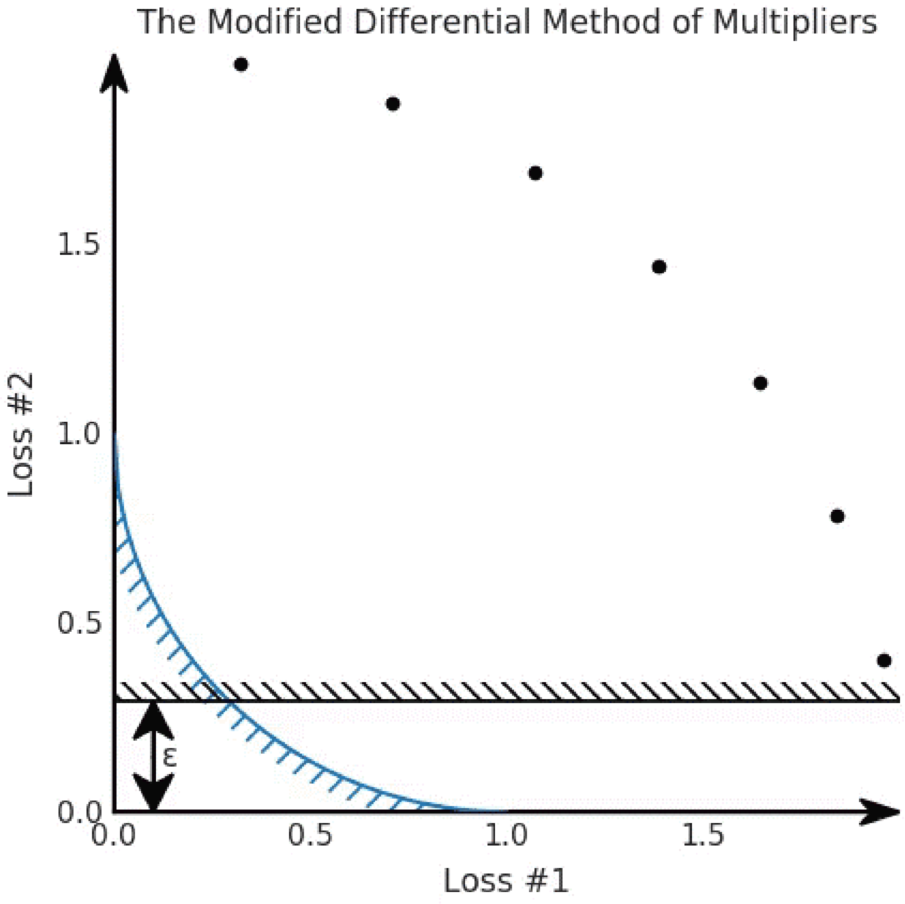
- This approach introduces a new hyperparameter δ
  - Only influences the *rate* of convergence
  - No change in minima location:
    quadratic term minimized for $g(\theta) = \varepsilon$
- Let's test it out!

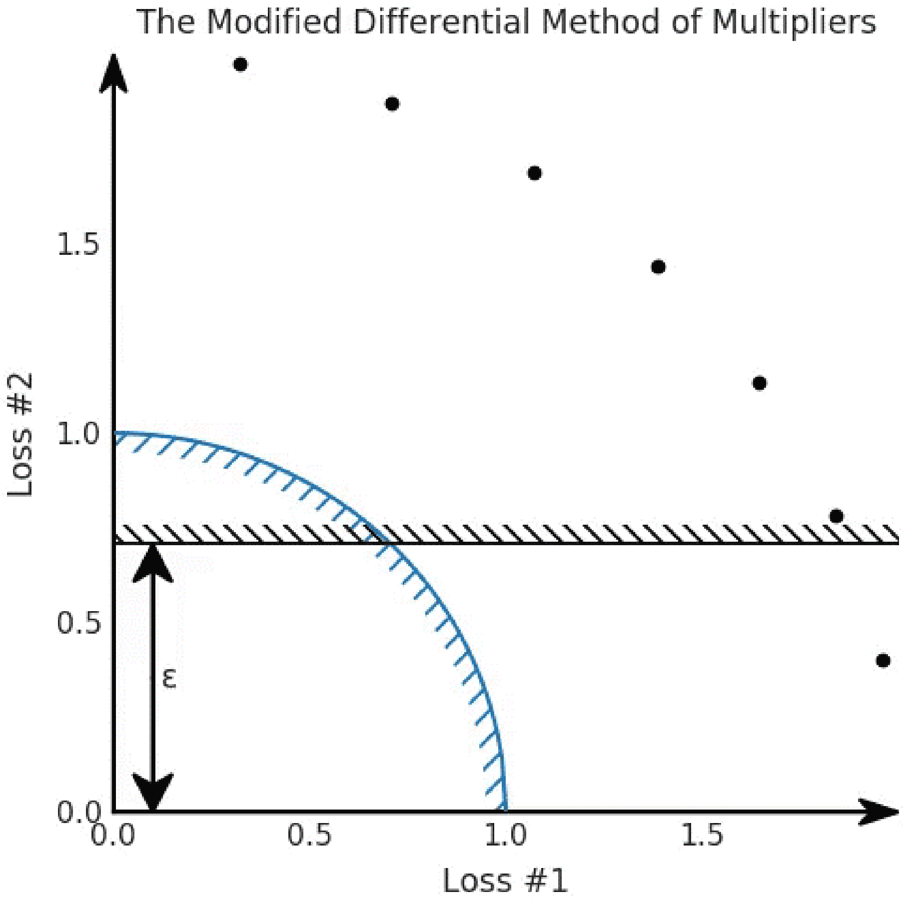


C. Cazzaniga

Pareto Fronts with MDMM

- Success!
- Reliable convergence for any Pareto front



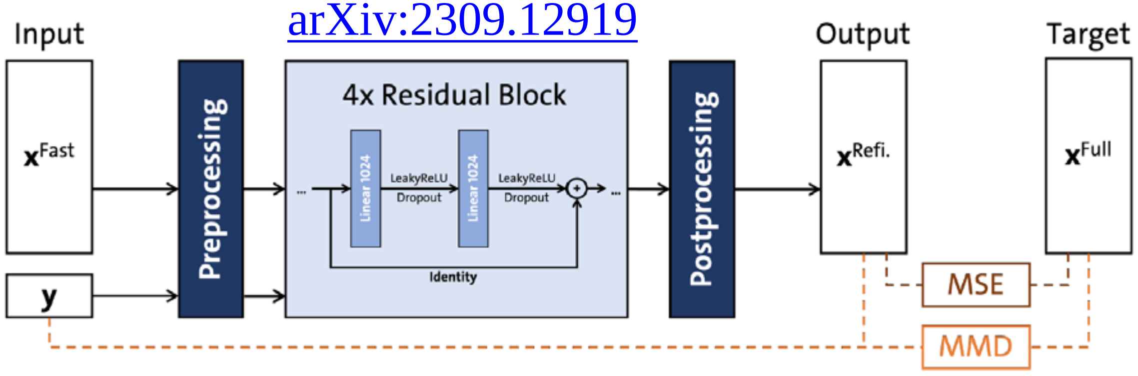


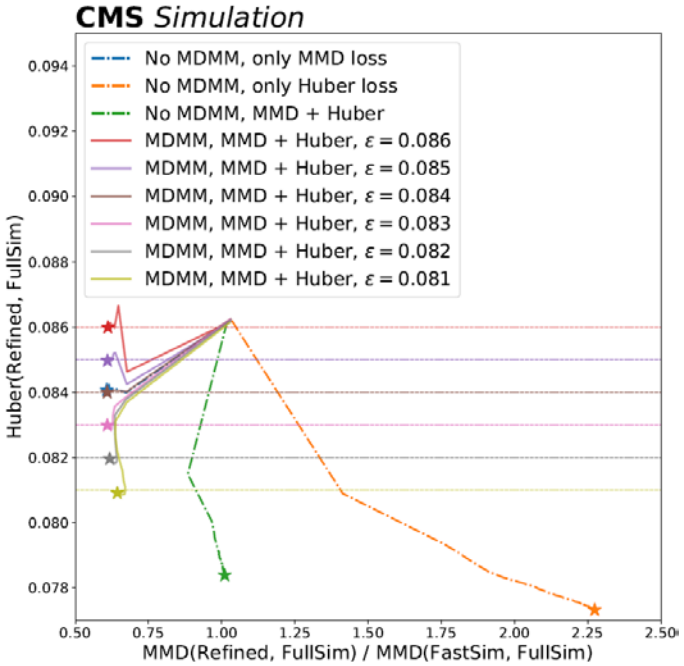
[J. Degrave & I. Korshunova]

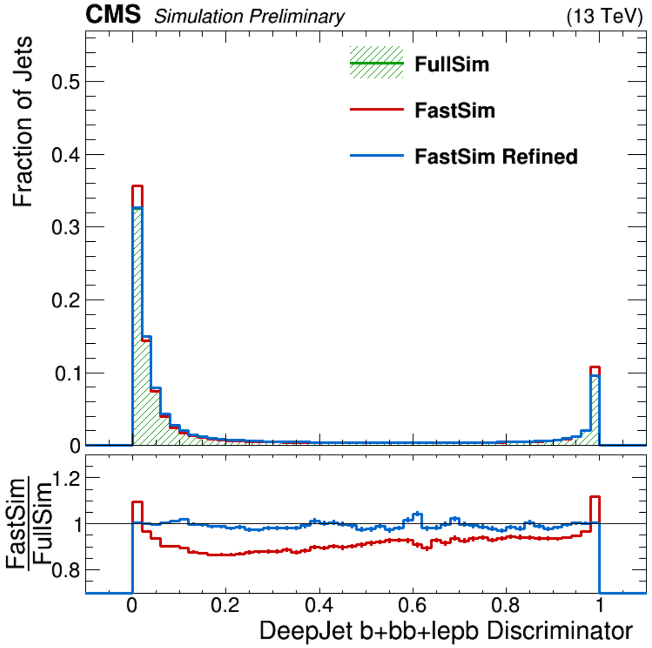
Application to Physics

- FastSim refinement: adjust high-level quantities from lower-quality fast simulation to better match high-quality (slow) full simulation
  - Target: b-jet tagging discriminators
- Two loss terms:
  - MSE (Huber): per-object comparison
  - MMD: ensemble comparison
- MDMM balances optimally:
  - Minimize MSE: bad MMD values
  - Minimize MMD: still good MSE!
  - Mechanically sketch out Pareto front by varying ε
- Substantial improvement in agreement w/ FullSim
- First known usage of MDMM in HEP!

[arXiv:2309.12919](https://arxiv.org/abs/2309.12919)







Summary

- Multi-objective training is a natural way to incorporate physics knowledge or other constraints
- MDMM ensures convergence
  - Specify constraints on loss terms: easily interpretable
  - Pick preferred tradeoff on Pareto front → no guessing!
  - Minimal hyperparameter tuning
- Not the only way to handle constrained optimization…
  - But (almost) always the best way
- Original paper: J. Platt, A. Barr, "Constrained Differential Optimization", NeurIPS, 1987
- PyTorch implementation available at https://github.com/crowsonkb/mdmm
  - Includes equality, min, and max constraints
  - Previously linked article by Degrave and Koshunova includes a basic JAX implementation
- A useful addition to your AI/ML toolkit!