Revisiting Line Searches

Machine Learning Reading Group Fall 2025

Betty Shea

2025-11-18

University of British Columbia

Learning rate α_t


```
for a in range(1, num_epoch+1):
    for (inputs, targets) in train_dataloader:
        optimizer.zero_grad()
        outputs = model(inputs)
        loss = criterion(outputs, targets)
        loss.backward()
        optimizer.step()
```

Learning rate α_t


```
for a in range(1, num_epoch+1):
    for (inputs, targets) in train_dataloader:
        optimizer.zero_grad()
        outputs = model(inputs)
        loss = criterion(outputs, targets)
        loss.backward()
        optimizer.step()
```

$$w_{t+1} = w_t + \alpha_t p_t$$

Learning rate α_t

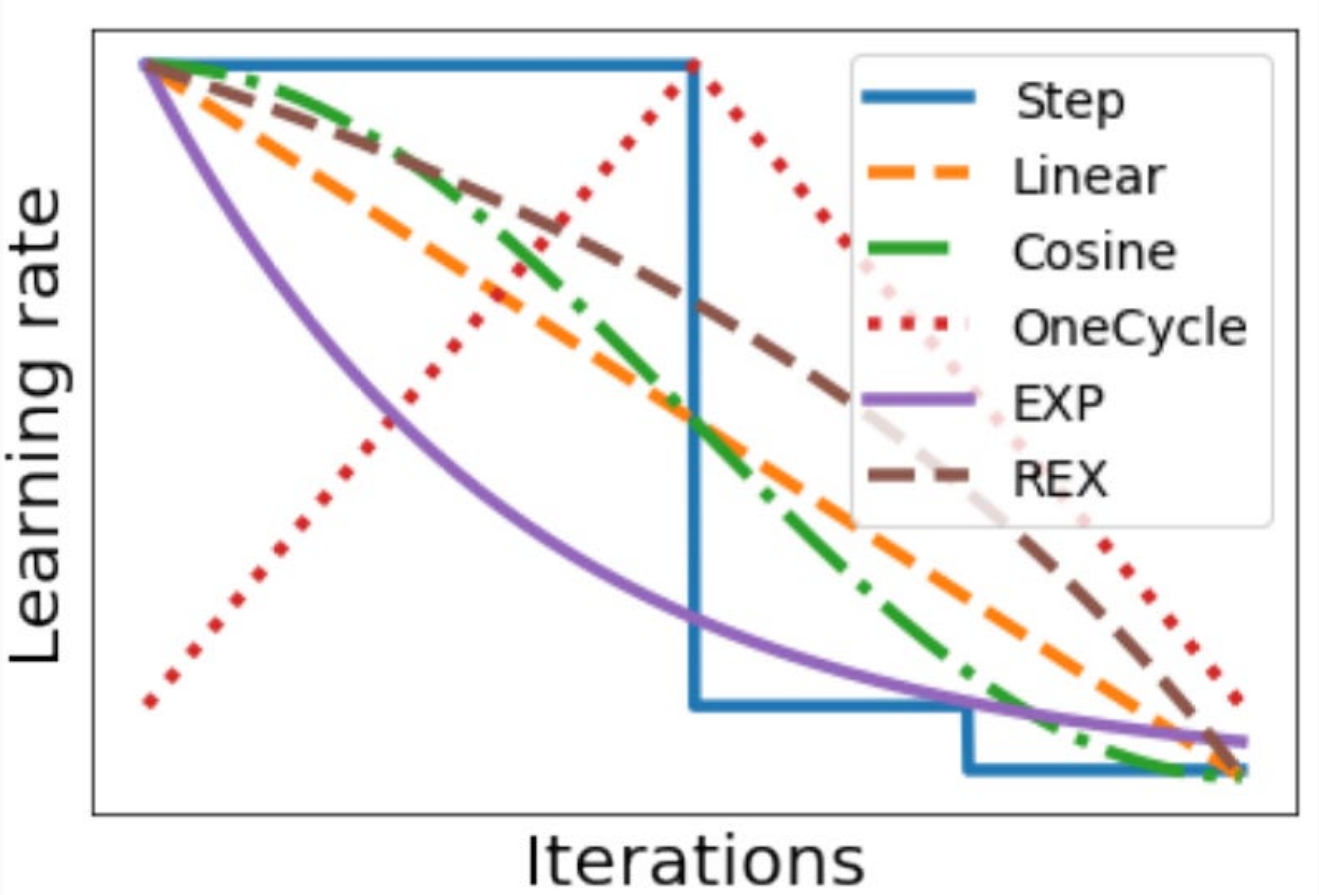
THE UNIVERSITY OF BRITISH COLUMBIA

```
for a in range(1, num_epoch+1):
    for (inputs, targets) in train_dataloader:
        optimizer.zero_grad()
        outputs = model(inputs)
        loss = criterion(outputs, targets)
        loss.backward()
        optimizer.step()
```

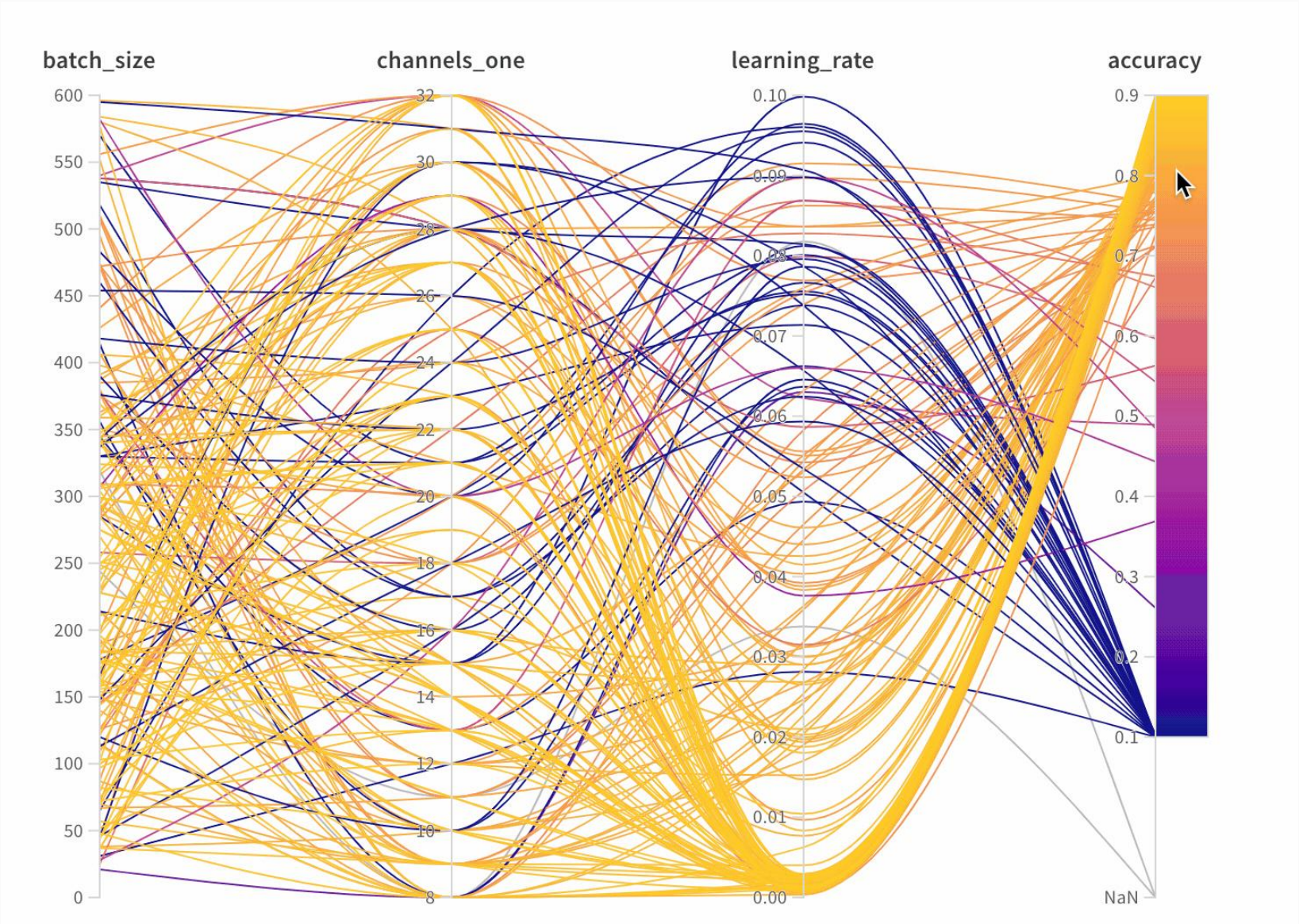
$$w_{t+1} = w_t + \alpha_t p_t$$

$\alpha_t = ??$ $\qquad$ $\alpha_t = \alpha$

Learning rate α_t



Learning rate α_t



Learning rate α_t

Can we set α_t on the fly? Some ideas outside of the scope of this talk:

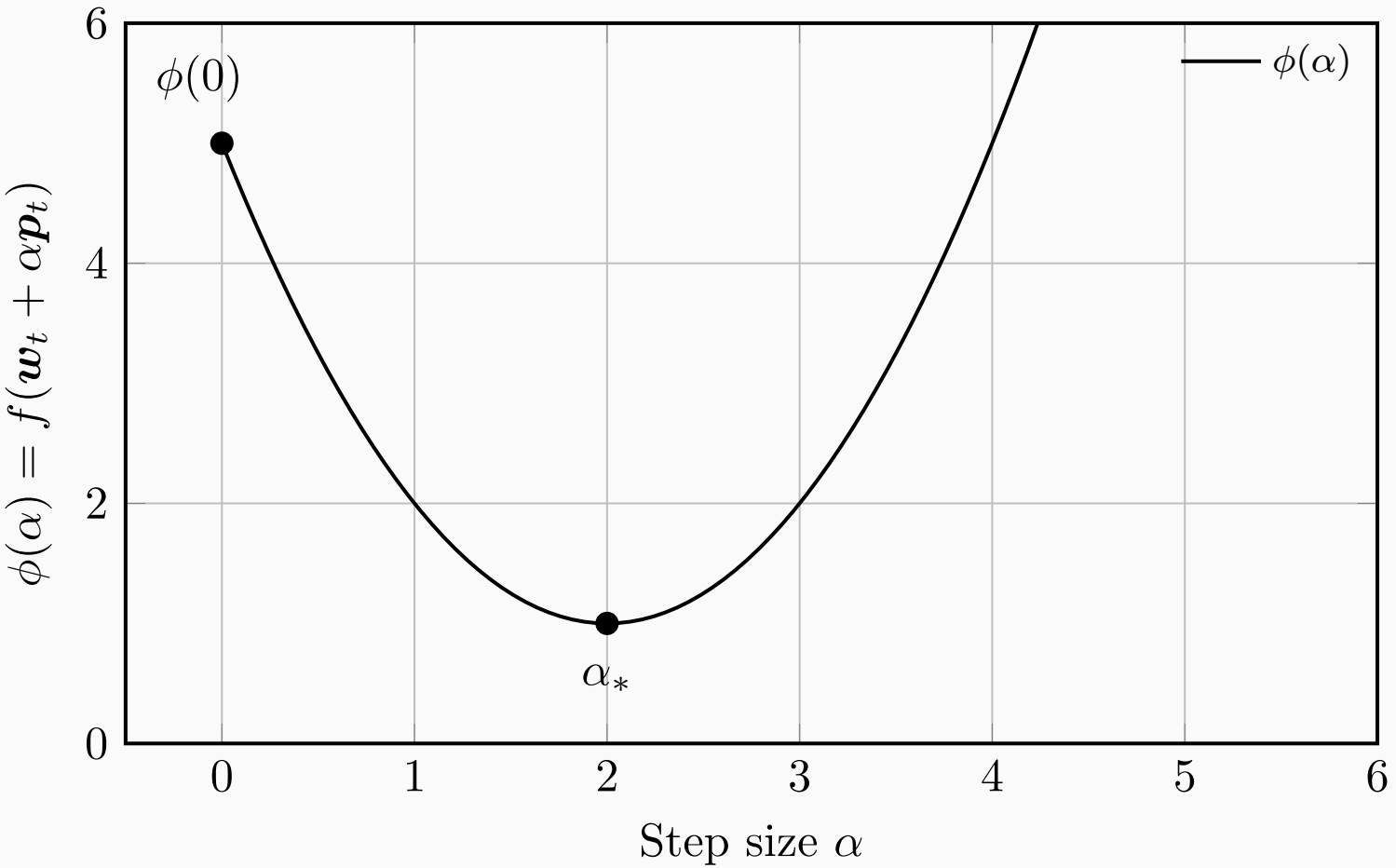
- Barzilai-Borwein
- Polyak step sizes (ask Curtis)
- Preconditioning
- Hyper-gradient descent

Line search

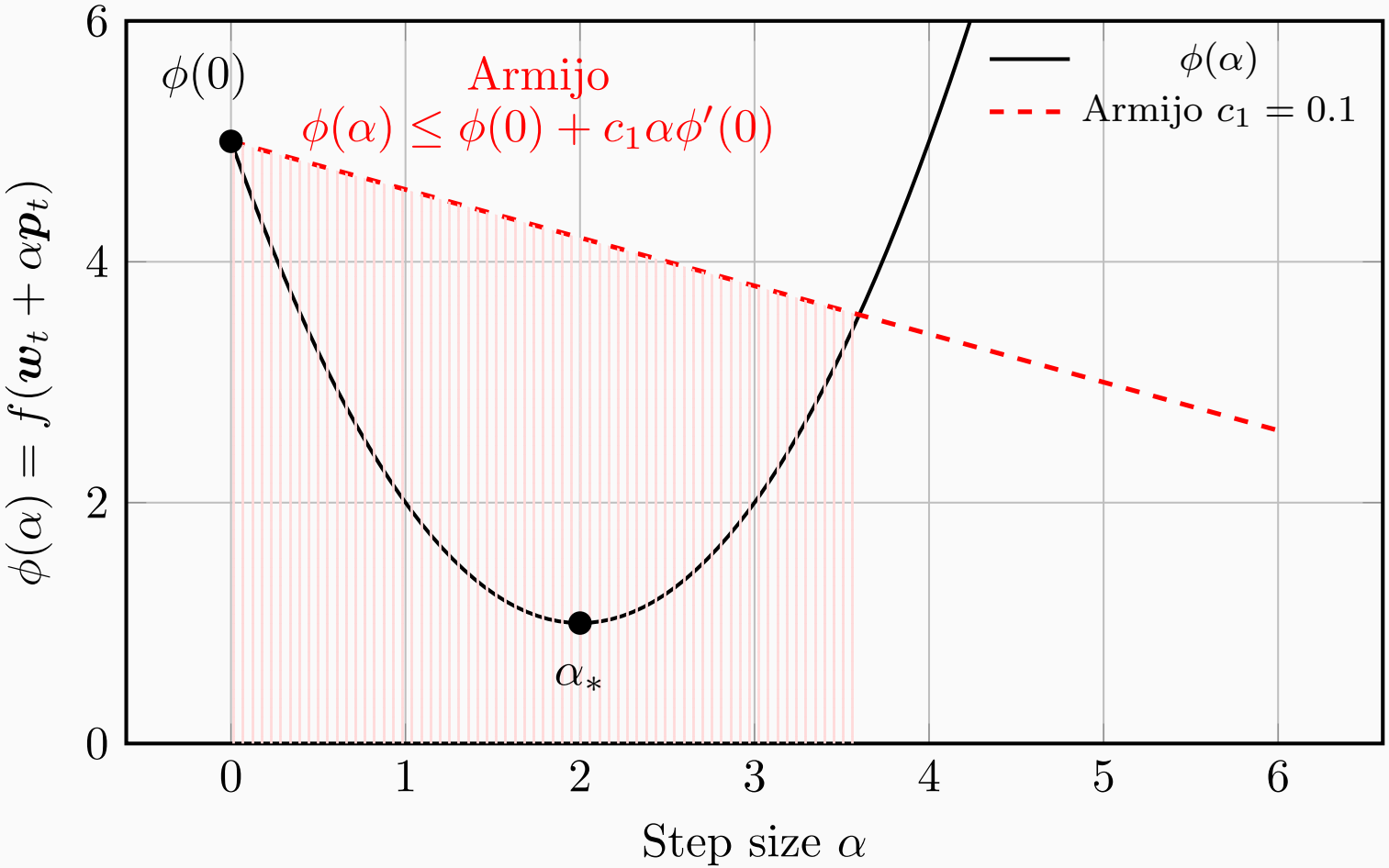
Sub-problem within every iteration of the main loop.

- Outer loop: training loop.
- Inner loop: find "good" learning rate α given fixed
  - weights w_t
  - search direction p_t
- $\phi(\alpha) = f(w_t + \alpha p_t)$
- $\phi'(0) = \nabla f(w_t)^\top p_t$, often assumed to be negative

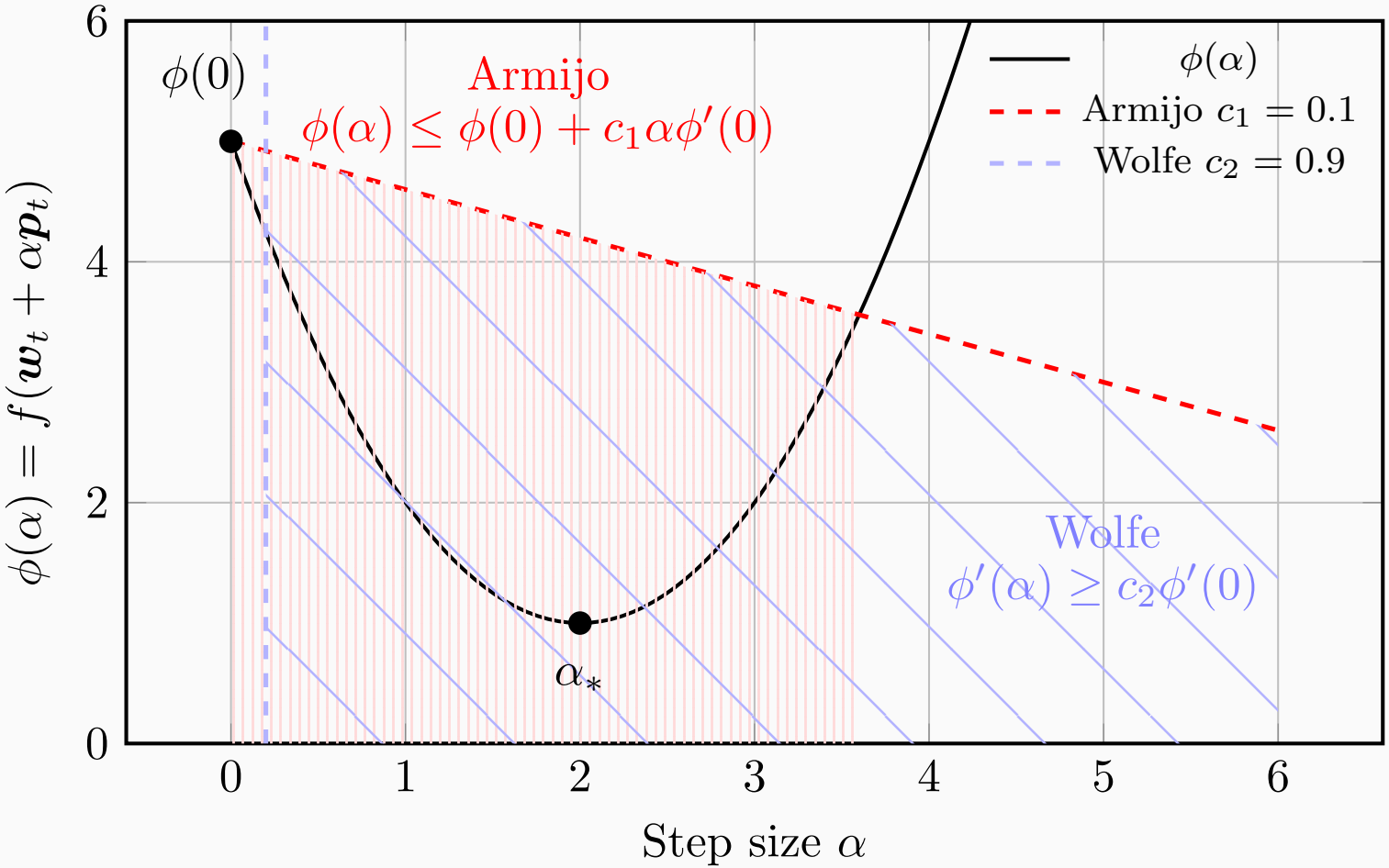
Line search


$\phi(0)$
$\phi(\alpha)$
α_*
$\phi(\alpha) = f(\boldsymbol{w}_t + \alpha \boldsymbol{p}_t)$
Step size α
0
2
4
6
0
1
2
3
4
5
6

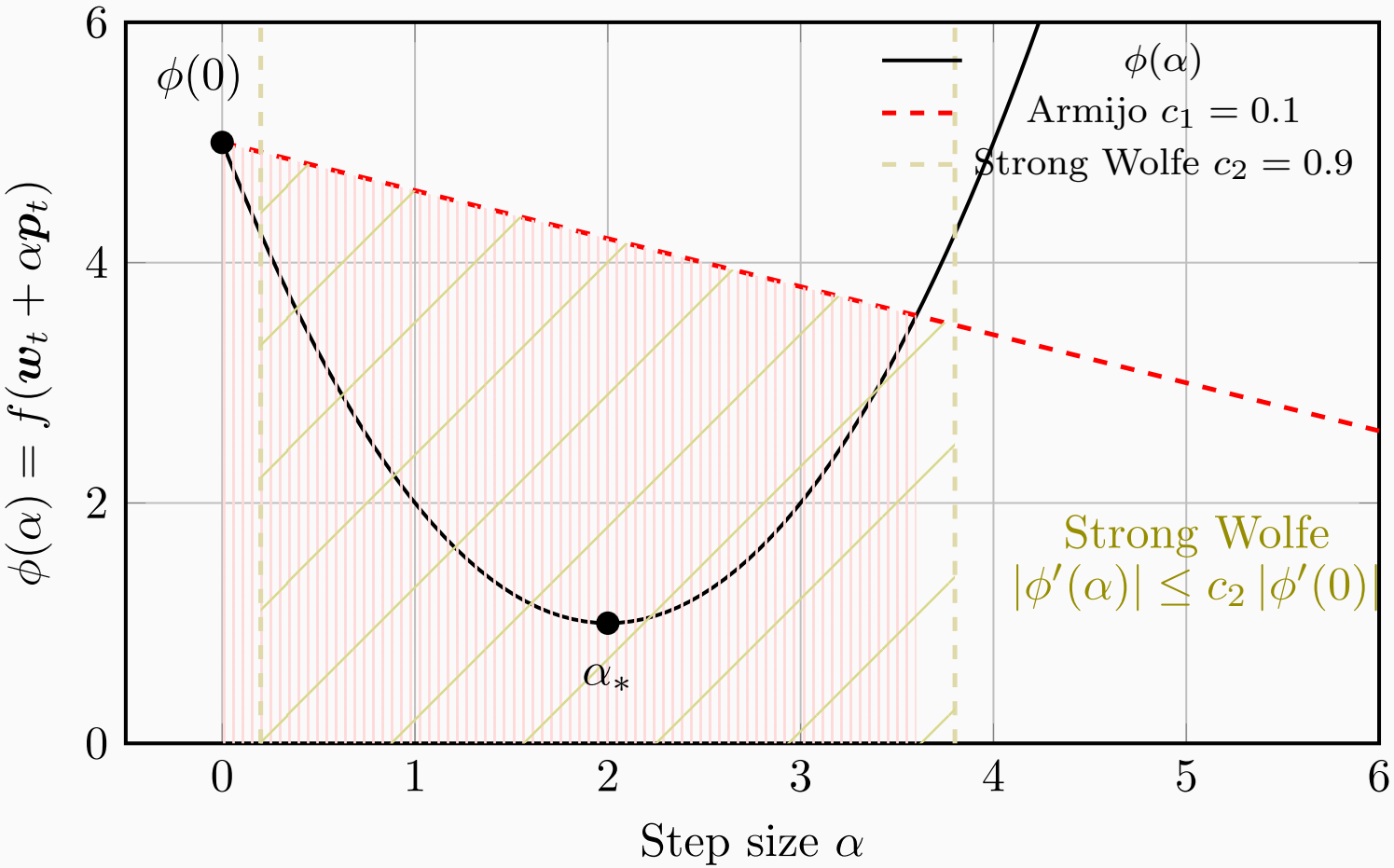
Armijo


$\phi(0)$
Armijo
$\phi(\alpha) \leq \phi(0) + c_1 \alpha \phi'(0)$
$\phi(\alpha)$
Armijo $c_1 = 0.1$
α_*
$\phi(\alpha) = f(\boldsymbol{w}_t + \alpha \boldsymbol{p}_t)$
Step size α
0
2
4
6
0
1
2
3
4
5
6

Wolfe


$\phi(0)$
Armijo
$\phi(\alpha) \leq \phi(0) + c_1\alpha\phi'(0)$
$\phi(\alpha)$
Armijo $c_1 = 0.1$
Wolfe $c_2 = 0.9$
Wolfe
$\phi'(\alpha) \geq c_2\phi'(0)$
α_*
$\phi(\alpha) = f(\boldsymbol{w}_t + \alpha\boldsymbol{p}_t)$
Step size α
0
2
4
6
0
1
2
3
4
5
6

Strong Wolfe



Why aren't line searches everywhere?

1. Stochasticity
2. Ineffectiveness
3. Cost per iteration

Stochasticity

- Line searches were originally designed for the deterministic setting.

Stochasticity

- Line searches were originally designed for the deterministic setting.
- There are theoretical results for stochastic versions.
    - Stochastic backtracking Armijo line searches [Vaswani et al. 2019, 2020, 2025]
    - Polyak Non-Monotone Stochastic backtracking Armijo line search [Galli et al. 2023] (ask Curtis)

Stochasticity

- Line searches were originally designed for the deterministic setting.
- There are theoretical results for stochastic versions.
    - Stochastic backtracking Armijo line searches [Vaswani et al. 2019, 2020, 2025]
    - Polyak Non-Monotone Stochastic backtracking Armijo line search [Galli et al. 2023] (ask Curtis)
- Some indication that noise from using mini-batches is beneficial for a line search. [Roulet et al. 2024]

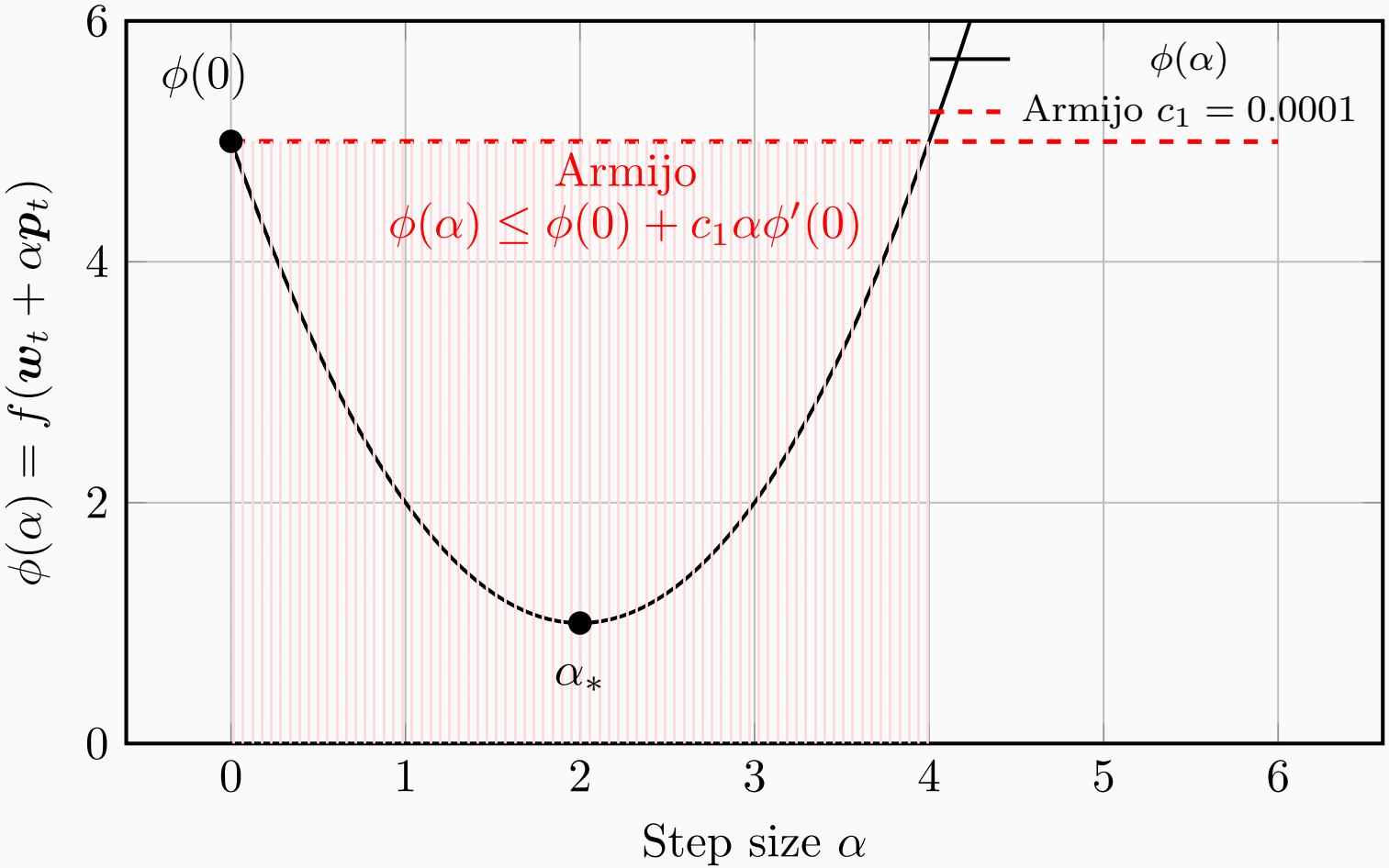
Ineffectiveness

- Are we really doing a line search?

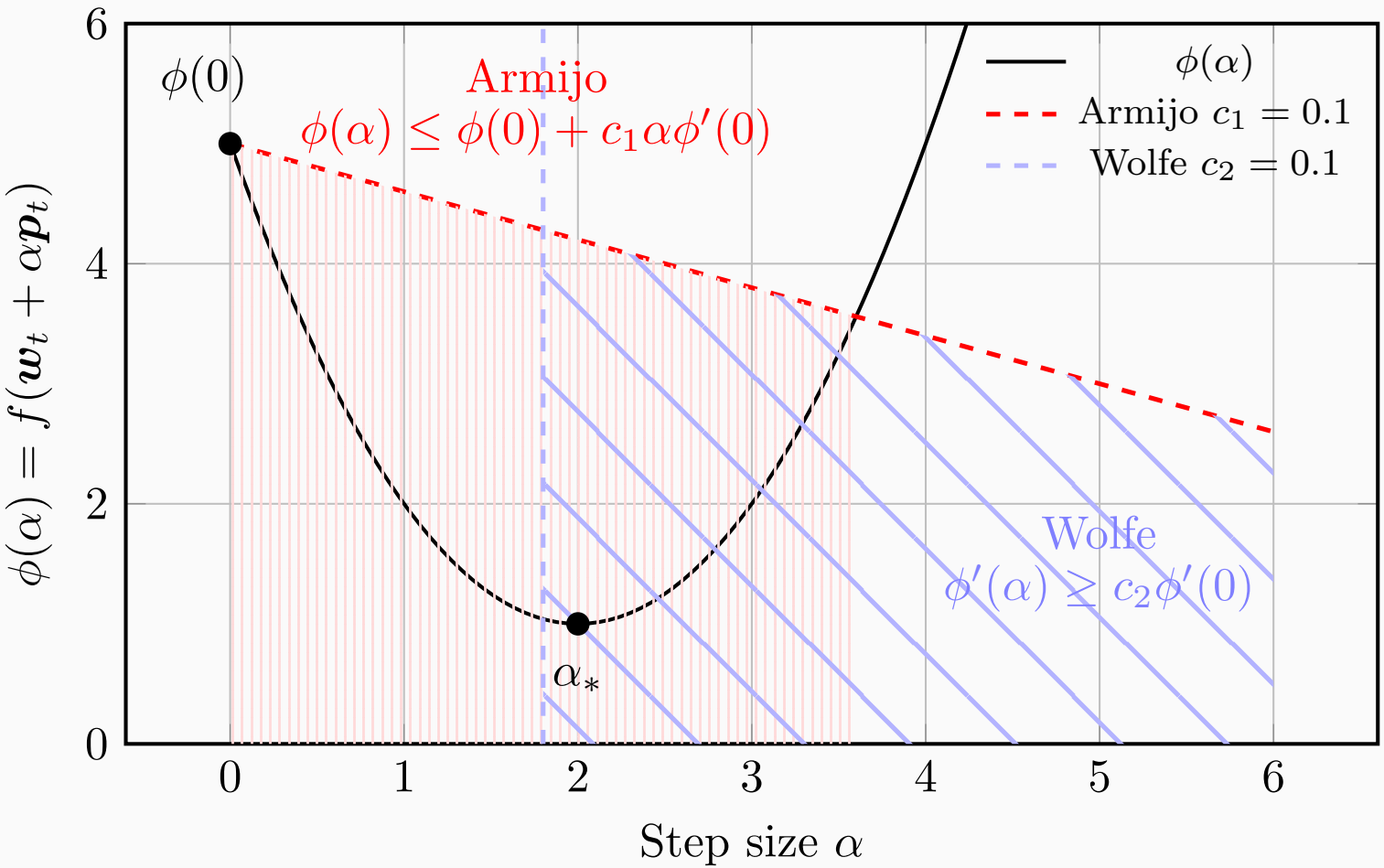
Ineffectiveness

- Are we really doing a line search?
- Usually backtracking Armijo with very loose conditions
- More of a sanity check than a search

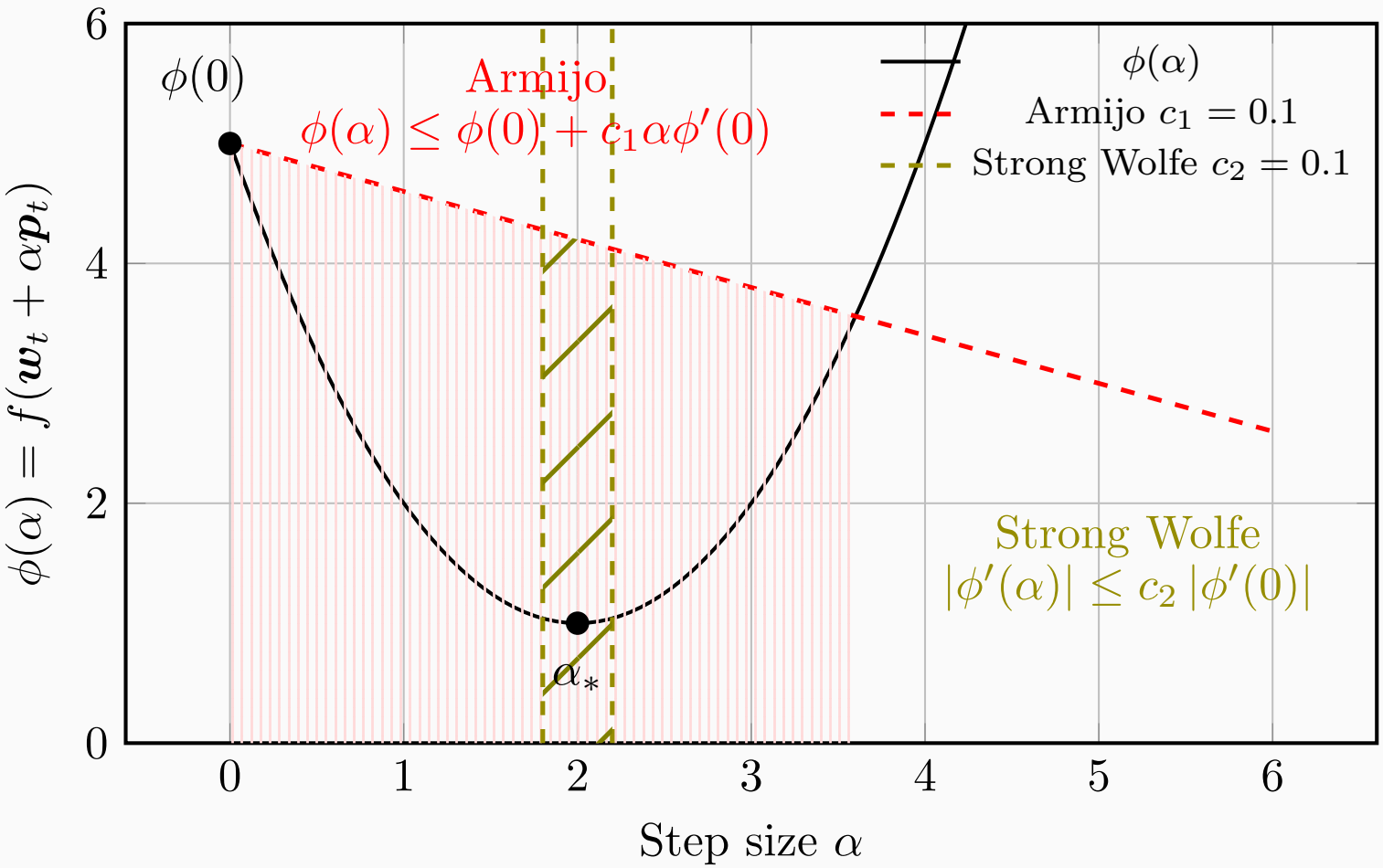
Ineffectiveness



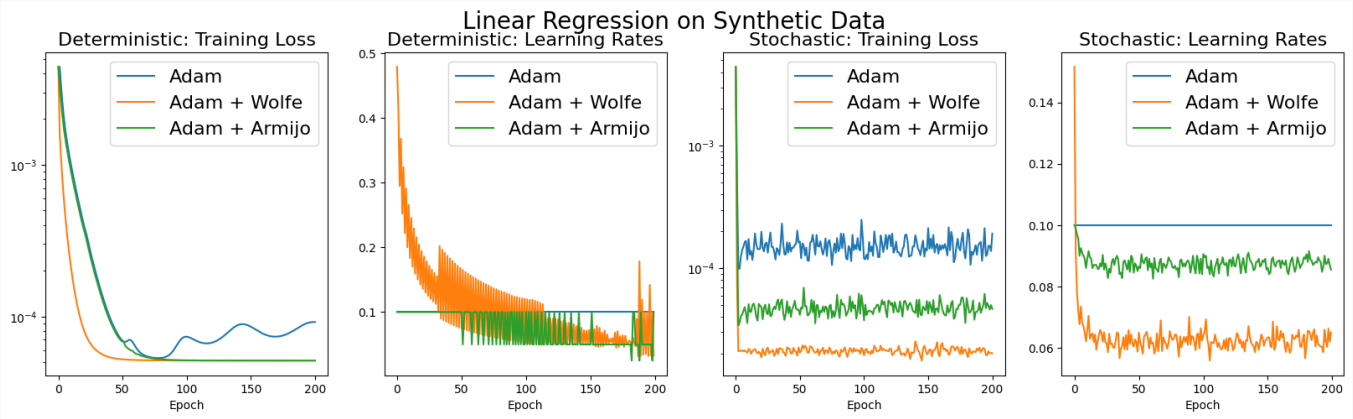
Stricter conditions

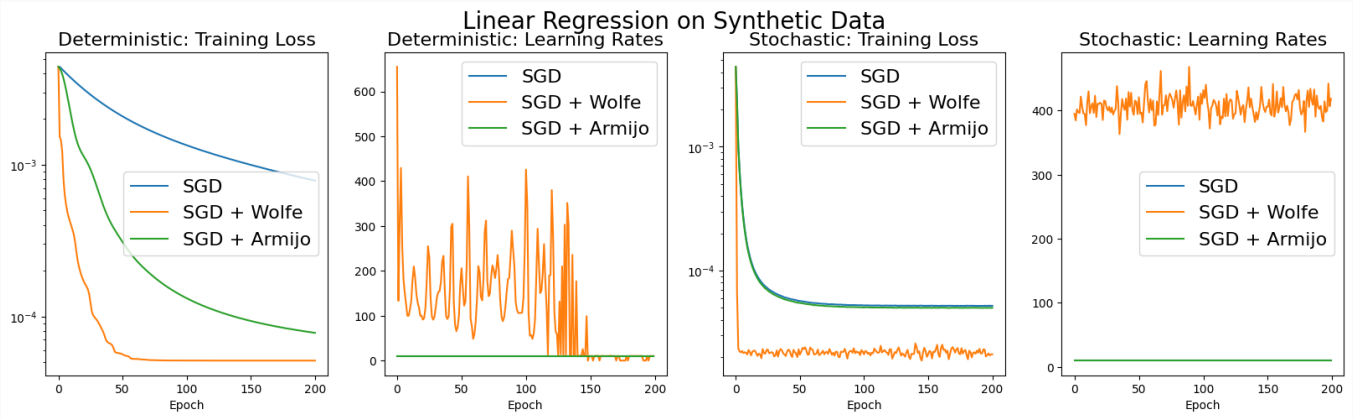


Stricter conditions

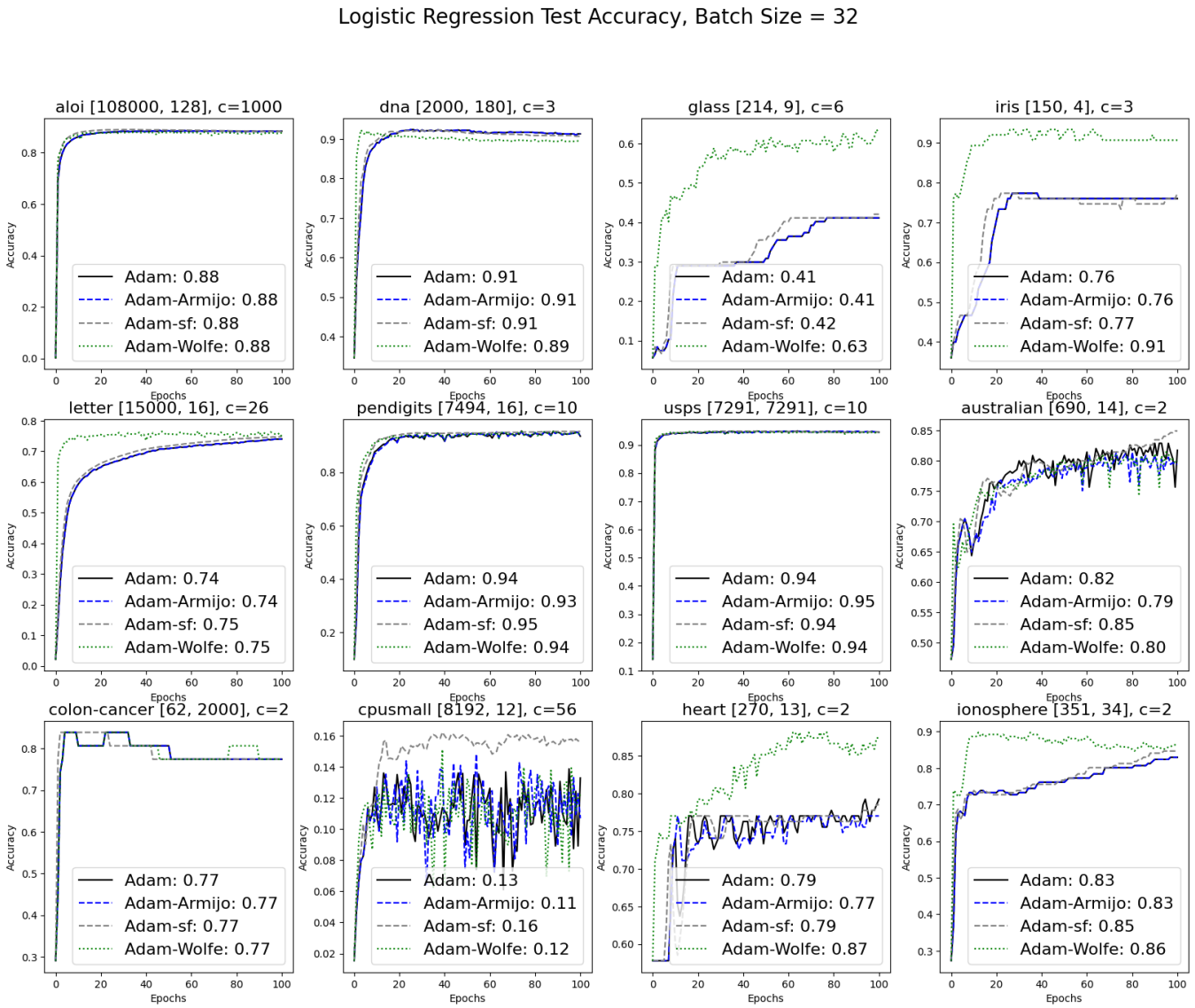


Stricter conditions

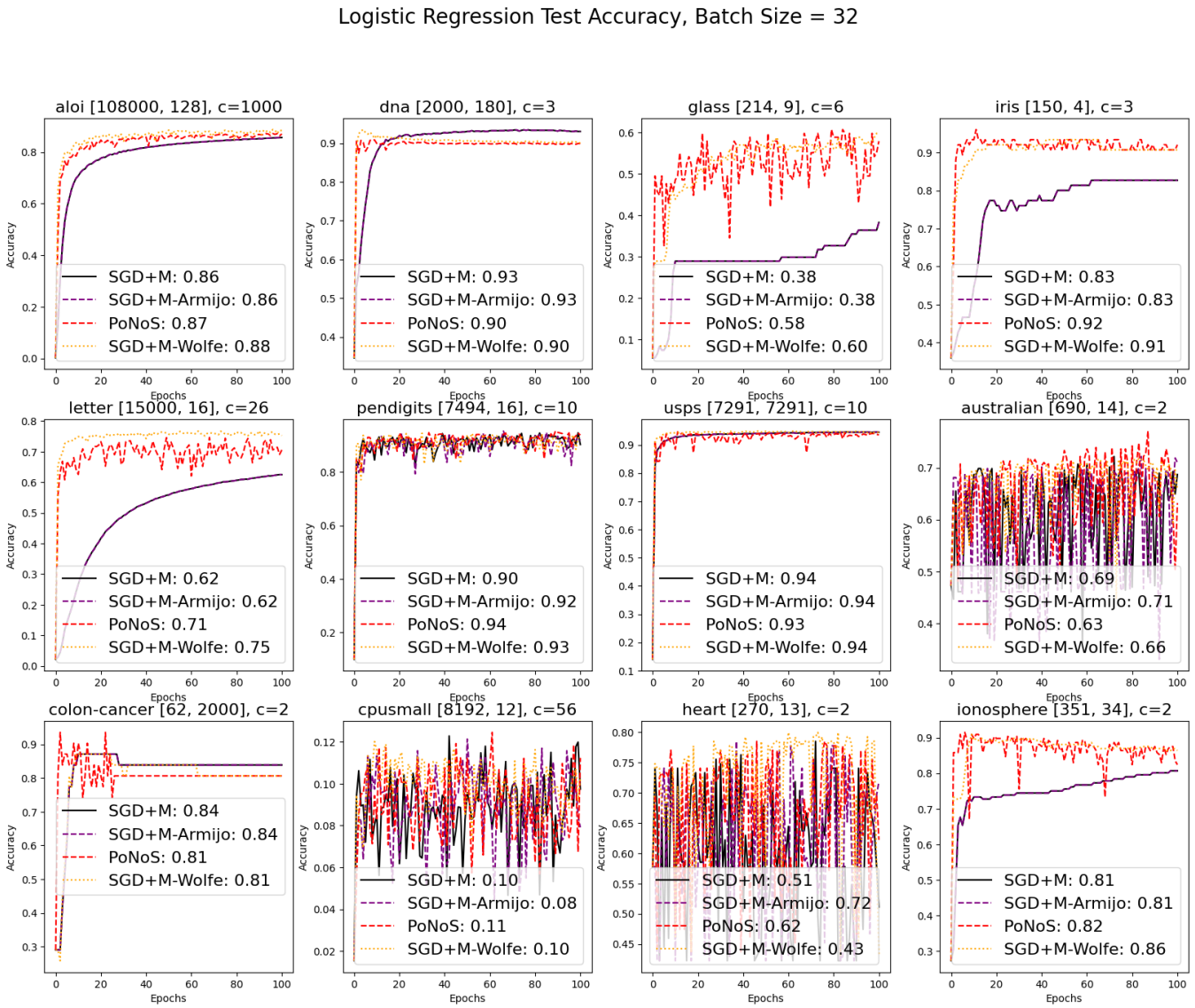




Stricter conditions



Stricter conditions



Cost per iteration

Searching is costly because of additional forward passes

- Generally, number of backtracks is closely monitored
- Wolfe is worse - requires additional backward passes too

Cost per iteration

Searching is costly because of additional forward passes

- Generally, number of backtracks is closely monitored
- Wolfe is worse - requires additional backward passes too

Is a line search needed on all model parameters?

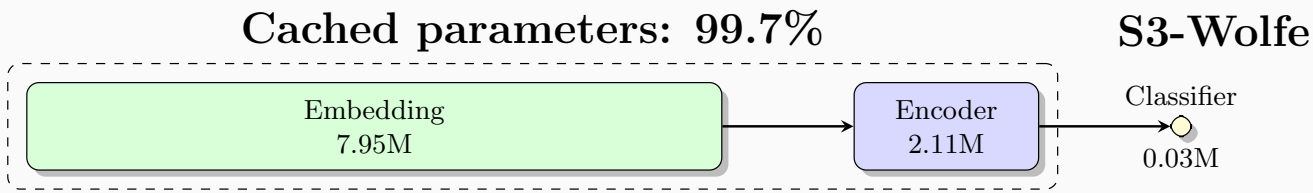
Cost per iteration

Maybe not in all cases....

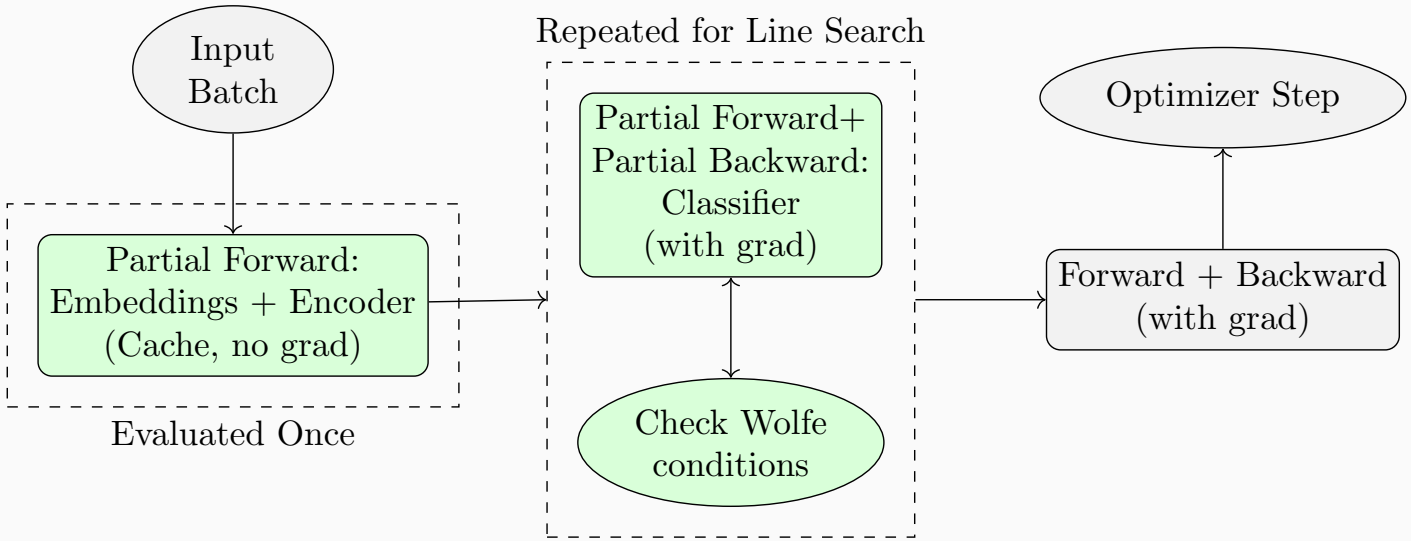
- Almost all parameters in BERT model in embedding and encoder layers
- Classifier parameters change most during fine-tuning
- Classifier parameters are a small fraction of all weights

Partial line search

- Line search accurately on small subset of parameters
- "Premium optimization" on selective parameters
- Similar to using parameter groups

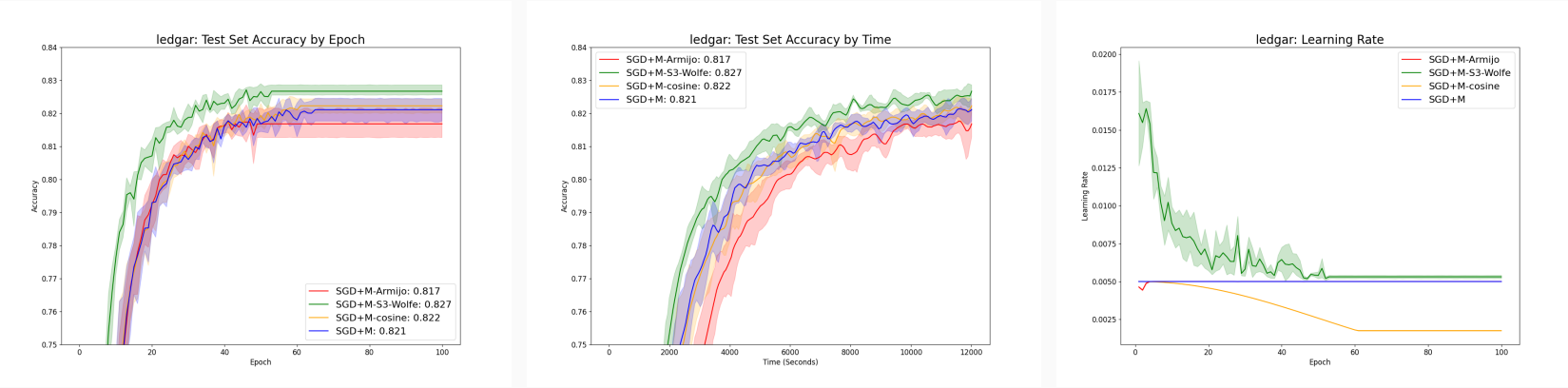


Partial line search



- Two forward passes and one backward pass regardless of accuracy
- With some work, can get rid of the final forward pass

Partial line search



Full fine-tuning `BertForSequenceClassification` model initialized with pretrained `legal-bert-base-uncased` checkpoint on the `ledgar` dataset.

Line search approach

Using a line search requires considering tradeoffs.

1. Accuracy vs cost per iteration

Line search approach

Using a line search requires considering tradeoffs.

1. Accuracy vs cost per iteration, or
2. Selectivity vs cost per iteration

Line search approach

Using a line search requires considering tradeoffs.

1. Accuracy vs cost per iteration, or
2. Selectivity vs cost per iteration

With large models, tradeoff 1 will always sacrifice accuracy.

Tradeoff 2 asks which parameters are worth spending optimization effort.

Work in progress

- Randomly initialized weights
    - Line search can give learning rates that are too far from the fixed learning rates used in the rest of the model
- Other criteria for search other than decreasing loss
    - Existing work on curvature aware learning rates suggest that being too greedy can hurt overall performance