perf: optimize abliteration matrix op #46

Code ▾ Jump to bottom

Merged

p-e-w merged 4 commits into `p-e-w:master` from `red40maxxer:abliteration-matmul-optimization`
20 hours ago

Conversation 21 Commits 4 Checks 4 Files changed 1

red40maxxer commented last week • edited ▾ Contributor

Found a possible small VRAM optimization when hacking around. We want to compute `M - weight * v (v^T M)` where `v` is the refusal direction for a layer. Instead of building the projector matrix in `O(d^2)` memory and `projector @ matrix` which is `O(d^2*k)`, we can use the identity `(v v^T) M = v(v^T M)` to apply the transformation directly to the weights. I made the varnames and comments correspond to the original Arditi et. al paper just for clarity

Was able to save about 2MB VRAM on my very low-power 4060 with Qwen-0.6B and it should increase for larger models. If my math is correct, the projector matrix size would be around 256MB for a 70B model which is a decent chunk saved, especially with the cost of VRAM these days! There was a very slight increase in computation time, possibly from CPU overhead from Python (I have no idea tbh) but it should be negligible.

Before optimization

```
GPU type: NVIDIA GeForce RTX 4060 Laptop GPU

Loading model Qwen/Qwen3-0.6B...
* Trying dtype auto... Ok
* Transformer model with 28 layers
* Abliterable components:
  * attn.o_proj: 1 matrices per layer
  * mlp.down_proj: 1 matrices per layer

* Transformer model with 28 layers
* Abliterable components:
  * attn.o_proj: 1 matrices per layer
  * mlp.down_proj: 1 matrices per layer

Loading good prompts from mlabonne/harmless_alpaca...
* 400 prompts loaded
```

```
Loading bad prompts from mlabonne/harmful_behaviors...
* 400 prompts loaded

Loading good evaluation prompts from mlabonne/harmless_alpaca...
* 100 prompts loaded
* Obtaining first-token probability distributions...

Loading bad evaluation prompts from mlabonne/harmful_behaviors...
* 100 prompts loaded
* Counting model refusals...
* Initial refusals: 38/100

Calculating per-layer refusal directions...
* Obtaining residuals for good prompts...
* Obtaining residuals for bad prompts...

Running trial 1 of 200...
* Parameters:
  * direction_index = 14.25
  * attn.o_proj.max_weight = 1.40
  * attn.o_proj.max_weight_position = 21.88
  * attn.o_proj.min_weight = 1.15
  * attn.o_proj.min_weight_distance = 5.46
  * mlp.down_proj.max_weight = 1.21
  * mlp.down_proj.max_weight_position = 20.43
  * mlp.down_proj.min_weight = 0.72Was able to save about 2MB VRAM on my very low-
power 4060 with Qwen-0.6B and it should increase for larger models. If my math is
correct, the projector matrix size would be around 256MB for a 70B model which is a
decent chunk saved, especially with the cost of VRAM these days! There was a very
slight increase in computation time, possibly from CPU overhead from Python (I have no
idea tbh) but it should be negligible.

  * mlp.down_proj.min_weight_distance = 15.00
* Reloading model...
* Abliterating...
  Abliteration logic took 0.0188s
  Peak VRAM overhead: 14.00 MiB
* Evaluating...
  * Obtaining first-token probability distributions...
  * KL divergence: 0.02
  * Counting model refusals...
  * Refusals: 17/100

Elapsed time: 8s
Estimated remaining time: 25m 13s

Running trial 2 of 200...
* Parameters:
  * direction_index = per layer
  * attn.o_proj.max_weight = 1.24
  * attn.o_proj.max_weight_position = 22.26
  * attn.o_proj.min_weight = 0.94
  * attn.o_proj.min_weight_distance = 1.12
  * mlp.down_proj.max_weight = 1.37
  * mlp.down_proj.max_weight_position = 26.44
  * mlp.down_proj.min_weight = 1.04
  * mlp.down_proj.min_weight_distance = 3.13
* Reloading model...
* Abliterating...
  Abliteration logic took 0.0017s
  Peak VRAM overhead: 14.00 MiB
* Evaluating...
  * Obtaining first-token probability distributions...
  * KL divergence: 0.06
  * Counting model refusals...
  * Refusals: 33/100

Elapsed time: 15s
Estimated remaining time: 25m 14s

Running trial 3 of 200...
* Parameters:
  * direction_index = per layer
  * attn.o_proj.max_weight = 0.90
  * attn.o_proj.max_weight_position = 21.58
  * attn.o_proj.min_weight = 0.53
  * attn.o_proj.min_weight_distance = 4.89
  * mlp.down_proj.max_weight = 1.47
  * mlp.down_proj.max_weight_position = 18.15
  * mlp.down_proj.min_weight = 0.19
  * mlp.down_proj.min_weight_distance = 13.89
* Reloading model...
* Abliterating...
  Abliteration logic took 0.0043s
  Peak VRAM overhead: 14.00 MiB
* Evaluating...
  * Obtaining first-token probability distributions...
  * KL divergence: 0.14
  * Counting model refusals...
```

After optimization

```
GPU type: NVIDIA GeForce RTX 4060 Laptop GPU

Loading model Qwen/Qwen3-0.6B...
* Trying dtype auto... Ok
* Transformer model with 28 layers
* Abliterable components:
  * attn.o_proj: 1 matrices per layer
  * mlp.down_proj: 1 matrices per layer

Loading good prompts from mlabonne/harmless_alpaca...
* 400 prompts loaded

Loading bad prompts from mlabonne/harmful_behaviors...
* 400 prompts loaded

Loading good evaluation prompts from mlabonne/harmless_alpaca...
* 100 prompts loaded
* Obtaining first-token probability distributions...

Loading bad evaluation prompts from mlabonne/harmful_behaviors...
* 100 prompts loaded
* Counting model refusals...
* Initial refusals: 38/100
```

```
Calculating per-layer refusal directions...
* Obtaining residuals for good prompts...
* Obtaining residuals for bad prompts...

Running trial 1 of 200...
* Parameters:
  * direction_index = per layer
  * attn.o_proj.max_weight = 1.17
  * attn.o_proj.max_weight_position = 17.45
  * attn.o_proj.min_weight = 0.17
  * attn.o_proj.min_weight_distance = 1.93
  * mlp.down_proj.max_weight = 1.29
  * mlp.down_proj.max_weight_position = 24.54
  * mlp.down_proj.min_weight = 0.58
  * mlp.down_proj.min_weight_distance = 5.00
* Reloading model...
* Abliterating...
  Abliteration logic took 0.0087s
  Peak VRAM overhead: 12.01 MiB
* Evaluating...
  * Obtaining first-token probability distributions...
  * KL divergence: 0.08
  * Counting model refusals...
  * Refusals: 30/100

Elapsed time: 8s
Estimated remaining time: 26m 32s

Running trial 2 of 200...
* Parameters:
  * direction_index = 19.83
  * attn.o_proj.max_weight = 1.49
  * attn.o_proj.max_weight_position = 23.52
  * attn.o_proj.min_weight = 0.85
  * attn.o_proj.min_weight_distance = 12.74
  * mlp.down_proj.max_weight = 0.81
  * mlp.down_proj.max_weight_position = 25.95
  * mlp.down_proj.min_weight = 0.52
  * mlp.down_proj.min_weight_distance = 8.02
* Reloading model...
* Abliterating...
  Abliteration logic took 0.0041s
  Peak VRAM overhead: 12.01 MiB
* Evaluating...
  * Obtaining first-token probability distributions...
  * KL divergence: 0.05
  * Counting model refusals...
  * Refusals: 29/100

Elapsed time: 16s
Estimated remaining time: 25m 56s

Running trial 3 of 200...
* Parameters:
  * direction_index = per layer
  * attn.o_proj.max_weight = 1.05
  * attn.o_proj.max_weight_position = 20.51
  * attn.o_proj.min_weight = 0.85
  * attn.o_proj.min_weight_distance = 15.69
  * mlp.down_proj.max_weight = 1.05
```

```
    * mlp.down_proj.max_weight_position = 23.62
    * mlp.down_proj.min_weight = 0.49
    * mlp.down_proj.min_weight_distance = 12.59
  * Reloading model...
  * Abliterating...
    Abliteration logic took 0.0087s
    Peak VRAM overhead: 12.01 MiB
  * Evaluating...
    * Obtaining first-token probability distributions...
    * KL divergence: 0.11
    * Counting model refusals...
```

red40maxxer added 2 commits last week

`perf: optimize abliteration matrix op` 3d2a12b

`refactor: comments and var names correspond with arditi` ✓ a547bad

p-e-w commented last week Owner

This was actually itself a performance optimization.

Note that for MoE models, there can be many MLP matrices per layer, as many as there are experts. So for a 128-expert MoE, pre-computing the projector saves 128 matrix-vector multiplications per layer (though of course their total rank is lower than when multiplying with the full projector matrix, the number of coefficient multiplications is still less).

I am also somewhat hesitant in general to complicate the ablation logic in any way unless the resulting gains are substantial, because reliably testing that logic is difficult so it needs to be obviously correct, especially since it's about to become more complex anyway with #43.

red40maxxer commented last week • edited Contributor Author

> This was actually itself a performance optimization.
>
> Note that for MoE models, there can be many MLP matrices per layer, as many as there are experts. So for a 128-expert MoE, pre-computing the projector saves 128 matrix-vector multiplications per layer (though of course their total rank is lower than when multiplying with the full projector matrix, the number of coefficient multiplications is still less).

> I am also somewhat hesitant in general to complicate the ablation logic in any way unless the resulting gains are substantial, because reliably testing that logic is difficult so it needs to be obviously correct, especially since it's about to become more complex anyway with #43.

The proof of correctness is really trivial, it's 1 line and follows from the associativity of matrix multiplication: instead of doing `projector <- r r^T` and then `projector @ W`, we do `r (r^T W)`. I think my comments do an OK job of demonstrating how it corresponds with the abliteration formula in Arditi et. al, but if it's not worth potential future confusion we can move on.

I took a look at #43 and I'm 99% sure my changes are compatible, I'm also OK with waiting for it to be merged and then rebasing. Lmk what you think!

p-e-w commented 5 days ago Owner

Have you benchmarked this with a large MoE model to test how much worse the performance is because of the issue I described above?

red40maxxer commented yesterday • edited Contributor Author

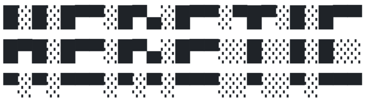
> Have you benchmarked this with a large MoE model to test how much worse the performance is because of the issue I described above?

Tried Qwen1.5-MoE-A2.7B on a Quadro RTX 6000. Without the optimization, double the VRAM is used for the abliteration op (32MiB -> 16MiB) and it takes 4x the time due to the issue you described.

I'm going to try with Phi-3.5-MoE-instruct on an A100 sometime later this week, but this looks OK to me

After optimization

```
root@e7bfffcc2a37:~/heretic# heretic Qwen/Qwen1.5-MoE-A2.7B
```



```
v1.0.1

https://github.com/p-e-w/heretic

GPU type: Quadro RTX 6000

Loading model Qwen/Qwen1.5-MoE-A2.7B...
Loading checkpoint shards: 100%|
████████████████████████████████████████████████████████████████
8/8 [00:08<00:00,  1.12s/it]
Some parameters are on the meta device because they were offloaded to the cpu.
Ok
* Transformer model with 24 layers
* Abliterable components:
  * attn.o_proj: 1 matrices per layer
  * mlp.down_proj: 60 matrices per layer

Loading good prompts from mlabonne/harmless_alpaca...
* 400 prompts loaded

Loading bad prompts from mlabonne/harmful_behaviors...
* 400 prompts loaded

Loading good evaluation prompts from mlabonne/harmless_alpaca...
* 100 prompts loaded
* Obtaining first-token probability distributions...

Loading bad evaluation prompts from mlabonne/harmful_behaviors...
* 100 prompts loaded
* Counting model refusals...
* Initial refusals: 78/100

Calculating per-layer refusal directions...
* Obtaining residuals for good prompts...
* Obtaining residuals for bad prompts...

Running trial 1 of 200...
* Parameters:
  * direction_index = per layer
  * attn.o_proj.max_weight = 1.07
  * attn.o_proj.max_weight_position = 14.52
  * attn.o_proj.min_weight = 0.56
  * attn.o_proj.min_weight_distance = 10.60
  * mlp.down_proj.max_weight = 0.83
  * mlp.down_proj.max_weight_position = 13.94
  * mlp.down_proj.min_weight = 0.26
  * mlp.down_proj.min_weight_distance = 11.26
* Reloading model...
Loading checkpoint shards: 100%|
████████████████████████████████████████████████████████████████████████████████████████
8/8 [00:08<00:00,  1.03s/it]
Some parameters are on the meta device because they were offloaded to the cpu.
* Abliterating...
  Abliteration logic took 0.1855s
  Peak VRAM overhead: 16.01 MiB
* Evaluating...
  * Obtaining first-token probability distributions...
  * KL divergence: 0.02
  * Counting model refusals...
  * Refusals: 79/100

Elapsed time: 4m 33s
Estimated remaining time: 15h 4m

Running trial 2 of 200...
* Parameters:
  * direction_index = 12.71
  * attn.o_proj.max_weight = 1.38
  * attn.o_proj.max_weight_position = 13.96
  * attn.o_proj.min_weight = 0.37
  * attn.o_proj.min_weight_distance = 2.46
  * mlp.down_proj.max_weight = 0.94
  * mlp.down_proj.max_weight_position = 19.43
  * mlp.down_proj.min_weight = 0.81
  * mlp.down_proj.min_weight_distance = 5.78
* Reloading model...
Loading checkpoint shards: 100%|
█████████████████████████████████████████████████████████████████████████████████████████
8/8 [00:08<00:00,  1.02s/it]
Some parameters are on the meta device because they were offloaded to the cpu.
* Abliterating...
  Abliteration logic took 0.1515s
  Peak VRAM overhead: 16.02 MiB
* Evaluating...
  * Obtaining first-token probability distributions...
  * KL divergence: 0.00
  * Counting model refusals...
```

Before optimization

```
root@e7bfffcc2a37:~/heretic# heretic Qwen/Qwen1.5-MoE-A2.7B
█░█░█▀▀░█▀▄░█▀▀░▀█▀░█░█▀▀  v1.0.1
█▀█░█▀▀░█▀▄░█▀▀░░█░░█░█░░
▀░▀░▀▀▀░▀░▀░▀▀▀░░▀░░▀░▀▀▀  https://github.com/p-e-w/heretic

GPU type: Quadro RTX 6000

Loading model Qwen/Qwen1.5-MoE-A2.7B...
Loading checkpoint shards: 100%|
█████████████████████████████████████████████████████████████████████████████████████████
8/8 [00:08<00:00,  1.01s/it]
Some parameters are on the meta device because they were offloaded to the cpu.
Ok
* Transformer model with 24 layers
* Abliterable components:
  * attn.o_proj: 1 matrices per layer
  * mlp.down_proj: 60 matrices per layer

Loading good prompts from mlabonne/harmless_alpaca...
* 400 prompts loaded

Loading bad prompts from mlabonne/harmful_behaviors...
* 400 prompts loaded

Loading good evaluation prompts from mlabonne/harmless_alpaca...
* 100 prompts loaded
* Obtaining first-token probability distributions...

Loading bad evaluation prompts from mlabonne/harmful_behaviors...
* 100 prompts loaded
* Counting model refusals...
* Initial refusals: 78/100

Calculating per-layer refusal directions...
* Obtaining residuals for good prompts...
* Obtaining residuals for bad prompts...

Running trial 1 of 200...
* Parameters:
  * direction_index = 20.01
  * attn.o_proj.max_weight = 1.10
  * attn.o_proj.max_weight_position = 22.42
  * attn.o_proj.min_weight = 0.89
  * attn.o_proj.min_weight_distance = 12.21
  * mlp.down_proj.max_weight = 0.98
  * mlp.down_proj.max_weight_position = 17.17
  * mlp.down_proj.min_weight = 0.61
  * mlp.down_proj.min_weight_distance = 11.01
* Reloading model...
Loading checkpoint shards: 100%|
█████████████████████████████████████████████████████████████████████████████████████████
8/8 [00:08<00:00,  1.01s/it]
Some parameters are on the meta device because they were offloaded to the cpu.
* Abliterating...
  Abliteration logic took 0.6206s
  Peak VRAM overhead: 32.01 MiB
* Evaluating...
  * Obtaining first-token probability distributions...
  * KL divergence: 0.02
  * Counting model refusals...
  * Refusals: 75/100

Elapsed time: 4m 34s
Estimated remaining time: 15h 8m

Running trial 2 of 200...
* Parameters:
  * direction_index = 17.47
  * attn.o_proj.max_weight = 1.16
  * attn.o_proj.max_weight_position = 17.80
  * attn.o_proj.min_weight = 0.80
  * attn.o_proj.min_weight_distance = 9.82
  * mlp.down_proj.max_weight = 1.47
  * mlp.down_proj.max_weight_position = 15.15
  * mlp.down_proj.min_weight = 1.45
  * mlp.down_proj.min_weight_distance = 9.48
* Reloading model...
Loading checkpoint shards: 100%|
█████████████████████████████████████████████████████████████████████████████████████████
8/8 [00:08<00:00,  1.01s/it]
Some parameters are on the meta device because they were offloaded to the cpu.
* Abliterating...
  Abliteration logic took 0.7275s
  Peak VRAM overhead: 32.01 MiB
* Evaluating...
  * Obtaining first-token probability distributions...
  * KL divergence: 0.25
  * Counting model refusals...
```

p-e-w commented yesterday Owner

Wait, the new logic is also faster? How does that work given that there are fewer coefficients to multiply?

red40maxxer commented yesterday • edited Contributor Author

> Wait, the new logic is also faster? How does that work given that there are fewer coefficients to multiply?

Forming the dense projector `P = v v^T` turns a rank-1 operation into a full `d × d` matrix multiply. Applying it requires `(d × d) @ (d × k)` for every expert, which is `O(d²·k)` work per matrix and forces a large d^2 tensor through GPU memory.

The new approach `v(v^T M)` exploits the fact that the projector is rank-1. We use the associativity of matrix multiplication to factor the operation into 2 computationally cheap steps:

1. `v^T M`, a matrix-vector multiply in `O(d·k)` time
2. `v (v^T M)`, outer product in `O(d·k)` time

The real cost isn't in computing the projection, it's multiplying the dense projection matrix into every expert matrix. Matrix-matrix multiplies tend to be very expensive. This way, we can avoid constructing the projector matrix and doing matrix-matrix multiplies altogether, giving us some time and space back.

Let me know if this clears it up, I'm not an expert on linear algebra so I'm learning here as well :)

p-e-w commented yesterday Owner

Ah yes, sorry, I just crawled out of bed and had the dimensionalities confused there. I literally just tried to do the same analysis in my head and got the reverse results, but your math is certainly correct 😄

We're of course merging this then, let me just do a quick review.

❤️ 1

p-e-w requested changes yesterday

View reviewed changes

p-e-w left a comment Owner

Just some comments regarding the explanations. They also show how incredibly subtle this stuff is, which is why I always hesitate to modify such code.

src/heretic/model.py Outdated

```
-                    layer_refusal_direction,
-                ).to(self.model.dtype)
+                # We use the property (r r^T) W = r (r^T W) to avoid comp
+                # the O(d^2) projector matrix and the O(d^3) matrix mult
```

p-e-w yesterday Owner

The multiplication is actually $O(d^2k)$ as you noted.

Reply...

src/heretic/model.py Outdated

```
+
+                        # Calculate the projection scalars: (r^T W)
+                        # hat_r is (d, 1), matrix is (d, k) -> result is (k,
+                        r_transpose_W = torch.matmul(hat_r_device, matrix)
```

p-e-w yesterday Owner

It's inconsistent to use `hat_r` above but not `hat_r_transpose` here, even though the paper uses $\hat{r}$ for both cases.

p-e-w yesterday Owner

I suggest removing the `hat_` prefix everywhere, as it just complicates things and serves no explanatory purpose in the code.

Reply...

src/heretic/model.py Outdated

```
+                        hat_r_device = hat_r.to(matrix.device)
+
+                        # Calculate the projection scalars: (r^T W)
+                        # hat_r is (d, 1), matrix is (d, k) -> result is (k,
```

p-e-w yesterday Owner

No, that doesn't work. The shapes you describe aren't compatible. The inner dimensions must match in matrix multiplication.

What actually happens is that Torch *prepends* `(1,` to 1d vectors, so `hat_r` is `(1, d)`, not `(d, 1)`.

Reply...

`src/heretic/model.py` Outdated

```
+                       r_transpose_W = torch.matmul(hat_r_device, matrix)
+
+                       # Calculate the update matrix: r * (r^T W)
+                       # Outer product of (d, 1) and (1, k) -> result is (d,
```

p-e-w yesterday Owner

Not quite, actually. What you are describing is matrix multiplication. The outer product takes two column vectors a and b, and is the equivalent to the matrix multiplication ab^T, but the outer product does not transpose the second vector. Both arguments to the outer product are column vectors.

red40maxxer yesterday Contributor Author

Mmm, technically the outer product already includes a transposition of the second vector, though at this point it's just semantics. I get that the comment is a bit misleading though so I'll be more clear on why the outer product is used here

p-e-w 20 hours ago Owner

The outer product is formally not a type of matrix multiplication at all, though it happens to be equal to a matrix multiplication. The inputs are not transposed, and passing tensors of shapes `(d, 1)` and `(1, k)` is undefined, and would raise an error in PyTorch.

Reply...

refactor: fix comments and improve var notation ✓ 2839d6f

red40maxxer requested a review from **p-e-w** yesterday

p-e-w requested changes 20 hours ago

View reviewed changes

src/heretic/model.py Outdated

```
-                        layer_refusal_direction,
-                    ).to(self.model.dtype)
+                    # We use the property (r r^T) W = r (r^T W) to avoid comp
+                    # the O(d^2 k) projector matrix and the O(d^3) matrix mul
```

p-e-w 20 hours ago Owner

I think those complexities are still not correct. Computing the projector takes d^2 float multiplications, not d^2k, and multiplying it with the matrix takes d^2k multiplications, not d^3.

Reply...

src/heretic/model.py Outdated

```
-                    ).to(self.model.dtype)
+                    # We use the property (r r^T) W = r (r^T W) to avoid comp
+                    # the O(d^2 k) projector matrix and the O(d^3) matrix mul
+                    # W_new = W - r * (r^T W)
```

p-e-w 20 hours ago Owner

We have a weight, so this formula is incomplete.

Reply...

src/heretic/model.py Outdated

```
+                        r_device = r.to(matrix.device)
+
+                        # Calculate the projection scalars: (r^T W)
+                        # r is (1, d), matrix is (d, k) -> result is (k,)
```

p-e-w 20 hours ago Owner

r is actually `(d,)`. `torch.matmul` internally transforms it to `(1, d)`.

Reply...

src/heretic/model.py Outdated

```
            return torch.cat(logprobs, dim=0)

-       def stream_chat_response(self, chat: list[dict[str, str]]) -> str:
+       def stream_cresponse(self, chat: list[dict[str, str]]) -> str:
```

p-e-w 20 hours ago Owner

?

red40maxxer 20 hours ago Contributor Author

🤦

Reply...

fix: accidental line change and improve comments ✓ cc66f78

p-e-w merged commit **60bd531** into p-e-w:master 20 hours ago View details

4 checks passed

p-e-w commented 20 hours ago Owner

Okay, this is going in. Thanks!

accemlcc added a commit to accemlcc/heretic-lora that referenced this pull request 16 hours ago

perf: optimize abliteration matrix op (p-e-w#46) ··· 5e65ab1

Reviewers

p-e-w

Assignees

No one assigned

Labels

None yet

Projects

None yet

Milestone

No milestone

Development

Successfully merging this pull request may close these issues.

None yet

2 participants