Shapes of different methods
Example:
Task: task 1 and task 2 (specific)

GEM shapes:

Variable	Shape	Specific shape
memories (G)	(t-1, p)	(1, p)
gradient (g)	p vector	(p,)
$D = GG^T$	(t-1, t-1)	(1, 1)
$d = -Gg$	t-1 vector	(1,)
$A = E$	(t-1, t-1)	(1, 1)
$m = 0$	t-1 vector	(1,)

ModGEM shapes:

Variable	Shape	Specific shape
memories (G)	((t-1)*(2*20+1), p)= (41*(t-1), p)	(41, p)
gradient (g)	p vector	(p,)
$D = GG^T$	(41*(t-1), 41*(t-1))	(41, 41)
$d = -Gg$	41*(t-1) vector	(41,)
$A = E$	(41*(t-1), 41*(t-1))	(41, 41)
$m = relatedness$	41*(t-1) vector	(41,)

Note: 2 means the number of layers, 20 means the number of groups

AGEM shapes:

Variable	Shape	Specific shape
memories (G)	(1, p)	(1, p)
gradient (g)	p vector	(p,)
$D = GG^T$	(1, 1)	(1, 1)
$d = -Gg$	1 vector	(1,)
$A = E$	(1, 1)	(1, 1)
$m = 0$	1 vector	(1,)

ModAGEM shapes:

Variable	Shape	Specific shape
memories (G)	((t-1)*(2*20+1), p)= (41*(t-1), p)	(41, p)
gradient (g)	p vector	(p,)
relatedness (h)	41*(t-1) vector	(41,)
$\tilde{G} = hG$	(1, p)	(1, p)
$D = \tilde{G}\tilde{G}^T$	(1, 1)	(1, 1)
$d = -\tilde{G}g$	1 vector	(1,)
$A = E$	(1, 1)	(1, 1)

$m = 0$	1 vector	(1,)

The logic behind $\tilde{G} = hG$:
Let's say we have 10 tasks now. The previous tasks are task 1, task 2,…, task 9. The current task is task 10. Due to modularization, we have 20*2 groups for each task. Besides, we have already got the relatedness between tasks on these groups. Here, let's assume all the groups of task 1 have the highest relatedness with the current task. So we can treat 20 groups in task 1 as a whole group and let's say this whole group just named task 1. When we finish training the model on task 10, we have to put memories (previous tasks) into the model and get losses. GEM aims at preventing a gradient step because of an individual task constraint violation. AGEM aims at considering the average loss constraint violation while ignoring the individual cases. In AGEM, we can also understand the average of gradients (loss) as the summation of gradients (loss). If the summation of memory loss increases in AGEM, we treat every loss from previous tasks equally and apply QP solver to update it. Here, since the current task (task 10) is more related with task 1 than other tasks, it is expected that the loss from task 1 is lower. If a violation occurs, we hope the loss of more related groups (like task 1 here) will decrease more after updating gradient which means the loss of task 1 should play a more important role in the summation of loss.

Another idea: should the less related tasks play a more important role here? Because since task 10 is more related with task 1, the loss of task 1 is expected to be lower than others. So the main loss comes from those less related tasks. When we update gradient, they should be considered more.