- Recent advancements have provided a remedy to catastrophic forgetting when training a network for new tasks
- Task arrival sequence matters with these methods
- I chose this paper because it immediately made intuitive sense to me and I thought I could even go as far as implementing the algorithm myself

$$-\frac{1}{N_t}\sum_{i=1}^{N^t}[\log p(y|x;\{\sigma + PQ^T\})] + \lambda_2||\sigma - \sigma^{(t-1)}||_2^2 + \lambda_1(||P|| + ||Q||_1) \qquad \text{ORACLE-Fixed}$$

- With the singular value decomposition of τ into $P,\ Q$, if m maintains the current network size, the product $PQ^T \approx \sigma$
- Intuitively, it would seem that the parameters of the loss function are expanding before optimization, and then contracting under two different penalty terms in order to arrive at a solution. The σ penalty constrains σ to be remain close to that of previous tasks. The P and Q penalties act as general penalties enforcing low parameter values.
- Since only the shared parameters from the direct previous task are taken into account when adding regularization, I would suspect that ORACLE-Fixed would slowly degrade performance of older tasks as the total task count grows.

The ORACLE algorithm adds more regularizations that keep it in line with earlier tasks, which would lead to a relatively more rigid value as time goes on, which was shown in figure 6.

$$\sum_{i=1}^{t-1}[\lambda_2||\theta_i^* - (\sigma \otimes M_i + \tau_i)||_2^2 + \lambda_3||M_i||_1] \qquad \text{ORACLE regularization 1}$$

- ORACLE keeps track of all θ values which it recovers on every new task. It is penalized for drifting away from ALL previous values and not just the directly preceding value.

$$\sum_{i=1}^{t-1}(||P||_1 + ||Q||_1) \qquad \text{ORACLE regularization 2}$$

- This penalty grows as the number of learned tasks grows and should enforce tighter value constraints as the number of tasks grows larger.
- In each iteration, all previous θ values are held constant, the σ values are allowed to change under constraints, and therefore the τ values are updated to keep θ constant learning new tasks
- τ contributes to future generations of learning by passing its modified values into the regularizations, which should have somewhat of an inverse effect on the current generations penalty.