Journal Report 8
10/28/19-11/1/19
Sohom Paul
Computer Systems Research Lab
Period 1, White

Daily Log

Monday October 28

Calculated correlations between tasks.
Investigated physical meanings behind correlations.

Tuesday October 29

Used correlations to make task adjacency graph (where a connected component indicates that those tasks may be learned together)
Tested clustering by learning network with 10 tasks.

Thursday October 31

Compared losses of clustered learning to nonclustered learning
Started reading about neural clustering algorithms.

Timeline

Date	Goal	Met
Oct 21	Build network with soft parameter sharing; modify loss accordingly.	Yes.
Oct 28	Finish building soft parameter sharing model. Compare to results with single-task learning and hard parameter sharing.	Yes.
Nov 4	Find algorithm to perform task clustering. Perform task clustering.	Yes
Nov 11	Implement neural clustering (CMTL) and compare with non-clustered results	
Nov 18	Implement deep relationship networks and compare with previous methods	

Reflection

Earlier this week, I worked on doing some preliminary clustering based on looking at simple linear correlations between tasks. The highest correlations were between the basic thermochemical quantities (U_0, U_{298}, H_{298} or G_{298}), which makes sense, considering that the definitions of these quantities include simple linear relationships between them. Thus, any model that learns any of the four can modified to learn all of them simultaneously. Alternatively, if I'm worried that I'm rewarding the network too much for learning only the thermochemical quantities, I could instead train a secondary network to compute the rest of the quantities given one. The other pair of quantities that had a very high correlation was the rotational constants B and C, which correspond to the moments of inertia about the second and third principal axes. I cannot give a clean, physical justification for why the correlation is so high ($R^2 > 0.98$), but I am grateful, for this will make training slightly easier.

I then constructed a task adjacency graph by thresholding correlations and found the connected components. When I used a threshold of 0.8, I found the following tasks were clustered together:

```
('B', 'C')
('A',)
('homo',)
('g298', 'h298', 'u0', 'u298')
('alpha', 'cv', 'g298_atom', 'gap', 'h298_atom', 'lumo', 'r2',
    'u0_atom', 'u298_atom', 'zpve')
('mu',)
```

To try to verify if this clustering was beneficial, I compared the results for when I trained all 19 tasks together to when I train only the 10 tasks in the largest cluster. Below are the losses after 15 epochs, training on 4000 samples.

Property	Cluster Learning	All-Task Learning
alpha	0.0661	0.0627
lumo	0.1848	0.163
gap	0.2568	0.1835
r2	0.3661	0.2264
zpve	0.0629	0.0456
cv	0.1014	0.0913
u0_atom	0.0386	0.0343
u298_atom	0.0381	0.0352
h298_atom	0.0376	0.0346
g298_atom	0.0411	0.0352

Note that the cluster-based learning is actually slightly worse than just training on all tasks together, implying that the benefits of having all the tasks learned together might outweigh the costs incurred when disimilar tasks are learned together. However, I am still committed to looking at more advanced clustering algorithms just to make sure that I didn't just cluster badly in this case.

I took longer than I had anticipated working on naively clustering methods because naively methods were giving surprisingly good results. However, for the next week, I want to try implementing neural clustering algorithms so I can learn nonlinear relationships between tasks. I am looking at the paper "Robust Task Clustering for Deep and Diverse Multi-Task and Few-Shot Learning" by Yu et al., 2018 and their clustering based on calculations of cross-task transfer performances. I will compare this to the naive cluster I got from comparing correlations. The following week, I want to use the knowledge I'll've gained about cluster structure to start building deep relationship networks and compare those results to my previously implemented hard and soft parameter sharing.