Journal Report 7
10/14/19-10/25/19
Sohom Paul
Computer Systems Research Lab
Period 1, White

Daily Log

Tuesday October 15

Refactored code to allow for easy changes in number of tasks learned simultaneously.

Thursday October 17

Finished refactor.
Started building toy soft-parameter sharing model.
Started reading about parameter- versus feature- regularization.

Monday October 21

Finished building toy soft-parameter sharing model.
Copied code from soft-parameter sharing toy model into actual GCN code.

Tuesday October 22

Tested soft-parameter sharing with different weights.
Started reading paper on feature learning.

Thursday October 24

Continued reading paper on feature learning.
Realized that paper was unhelpful.

Timeline

Date	Goal	Met
Oct 14	Train network and compare results to tasks done without multitask learning.	Yes.
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.	
Nov 11	Continue performing task clustering.	

Reflection

During the Anchor Day on Tuesday, October 15, I refactored my code in order to allow for me to easily specify and change the tasks being learned. After the refactor, I had discovered that my network was no longer producing good results. This was resolved when I came in, next class, and realized that I had forgotten that I had changed the learning rate, which was forcing my losses to blow up. I spent the next few classes building and testing my soft-paramter sharing model. Because I have no experience with multitask learning, and because there are relatively few online resources, I could not find a good value to give my regularizing hyperparamter. I spent a substantial amount of time on the following Tuesday testing out different values for the hyperparameter to see what worked while reading more surveys on multitask learning. It turns out that there are many classes of multitask learning that I had stumbled upon earlier, such as

I implemented a very simple version of soft-parameter sharing, where I penalize the network for differences between the matrix elements for different layers. Letting $X^{(1)}$ and $X^{(2)}$ represent two such constrained layers, my loss function takes the form:

$$\mathcal{L}(y, \bar{y}, X^{(1)}, X^{(2)}) = \sum_i (y_i - \bar{y}_i)^2 + \lambda \sum_i \sum_j (X_{ij}^{(1)} - X_{ij}^{(2)})^2$$

where λ is some weighting parameter that I manually inputted. I'd have any additional penalty term for each of the pairs of tasks learned. I couldn't actually find the paper where this ℓ_2 norm-based loss was used, so I wasn't sure what a good value for λ was, so I trained the network with different powers of 10. In the low λ limit, we expect the network to behave as if it were single-task learning, and in the large λ limit, we expect the network to behave is in hard-parameter sharing. Included below are my losses for various values of λ as I learned heat capacity (C_V) and internal energy at 298K (U_{298}). It looks like an optimal value for λ is close to 0.1.

λ	Total Loss	C_V Loss	U_{298} Loss
0.001	0.0915	0.0568	0.0346
0.01	0.1059	0.0793	0.0265
0.1	0.0654	0.0446	0.0207
1	0.0895	0.0507	0.0388

On Thursday, I read a paper on Convex Multi-Task Feature Learning[1]. It was a pretty cool paper and I don't regret spending time reading it; however, I eventually realized that I have no idea how to reconcile convex optimization with backpropagation. Some ideas that I had was introducing another penalty to my loss for when certain orthogonality contrainsts were violated, or projecting my gradient onto the space of changes consistent with maintaining matrix orthogonality. However, I could not find a good way to do this on paper, especially with the knowledge that I would eventually need to generalize this to include deep learning and graph convolutions, so I gave up trying to implement it. Also, feature selection would not be helpful for this particular project, because our inputs are already fairly low dimensional and any further reduction would be stripping out information about our molecule.

For the future, it is very important that we find out which tasks are highly correlated and which aren't. Multitask learning can be harmful if we try to learn simultaneously uncorrelated tasks. I have allotted two weeks for this goal because I am not yet sure how I'm going to accomplish it. The simplest, but least effective solution, is to just calculate Pearson's correlation coefficients, but that hides the deeper nonlinear relationships between variables. On the other hand, I don't think it's wise to try out the $2^{19} - 20 = 524268$ different sets of tasks we could learn simultaneously and see which of these provide results better than single-task learning. I hope that next week I'll be able to find some existing algorithm for task clustering and I'll be able to implement it over the following few days.

[1] https://ttic.uchicago.edu/~argyriou/papers/mtl$_{f}eat.pdf$