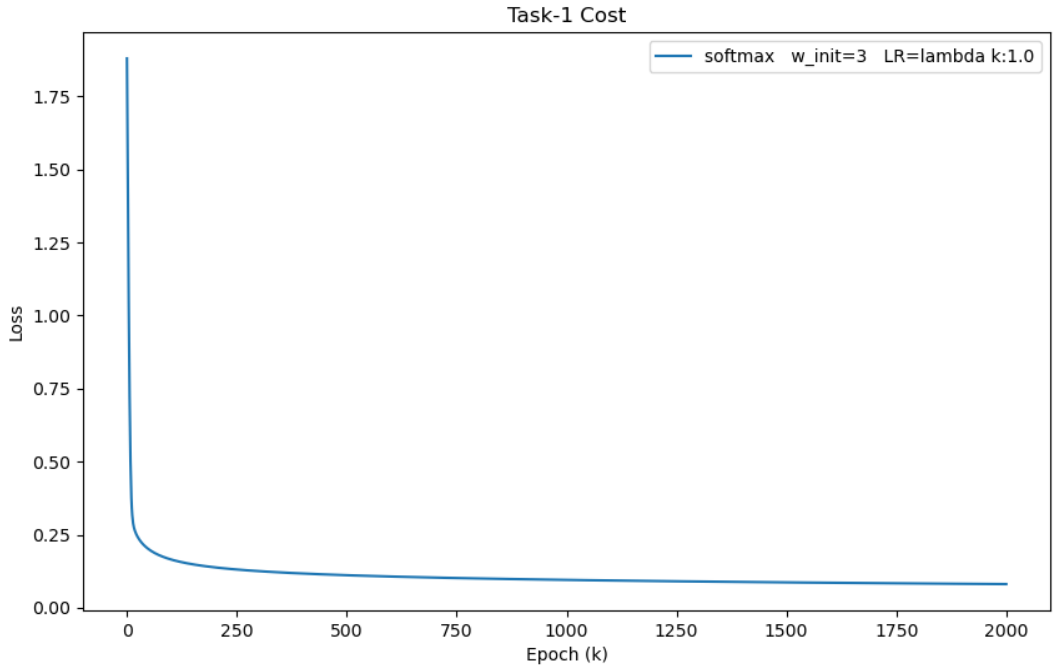
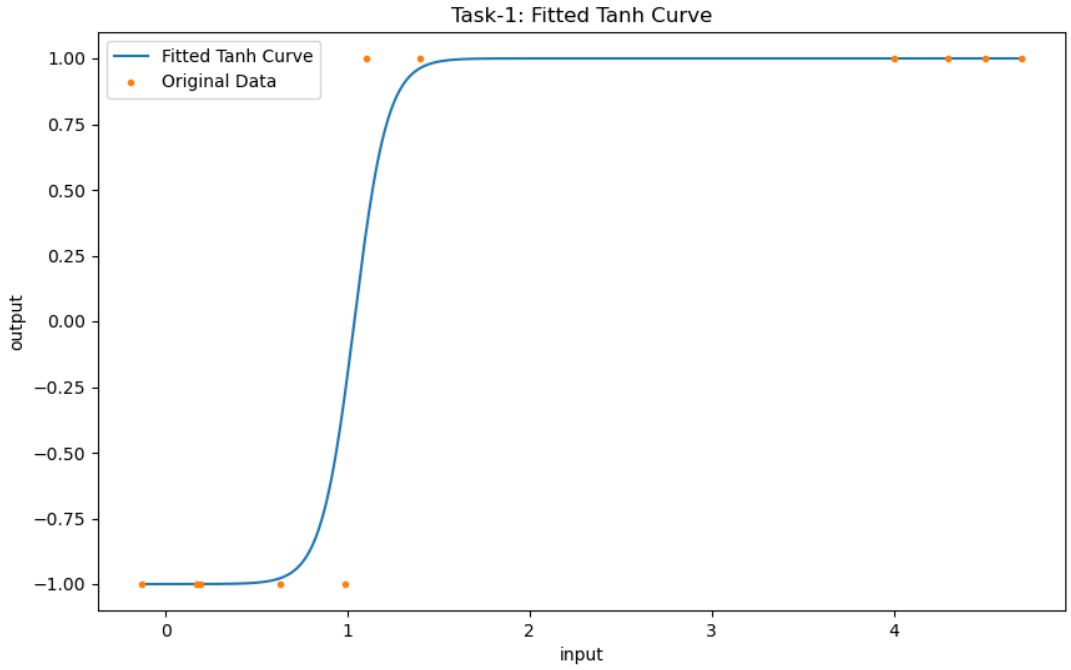
Andrew Hall
CS589 HW4 Report

Task 1

100% model accuracy with 0/11 misclassifications





Preliminary Task 2 experiments

Total Iterations = 2000 $\mathbf{w_0 = 1_9}$	LR = 1		LR = 1 / k		LR = 1 / (k % 50 + 1)	
	Softmax	Perceptron	Softmax	Perceptron	Softmax	Perceptron
(Best) Iteration / Loss	326 / .102	923 / .062	2000 / .282	2000 / .029	2000 / .117	349 / .012
Correct	673	663	644	659	673	672
Accuracy	96.28%	94.85%	92.13%	94.28%	96.28%	96.14%
Precision	97.16%	94.51%	92.96%	95.83%	97.16%	96.54%
Recall	97.16%	97.82%	95.20%	95.41%	97.16%	97.60%
F1	97.16%	96.14%	94.07%	95.62%	97.16%	97.07%

I gave my models 2000 iterations to ensure ample time for convergence while testing various learning rates. Initial observations revealed that, given equal hyperparameters, perceptron loss seemed to be more effective for shorter training periods (~10-100 iterations) compared to softmax. However, softmax outperformed when given more iterations. This can be attributed to the perceptron's steeper gradient which becomes null at zero, in contrast to the softmax's more gradual decline. Softmax also penalizes correct classifications with low confidence which may be why it is able to consistently achieve higher accuracy after more training iterations.

For weight initialization, I opted for a uniform value of 1. Initial weight choice seems less important than the other hyperparameters given the steep gradient of both methods for higher losses that may result from poor initialization. Rapid descent can be observed in the initial iterations of both loss methods, but the descent slows substantially in the later stages. Random weight initialization using normal and Xavier distributions yielded comparable results, with no major difference in the model's final accuracy.

For the learning rate, alpha, a constant rate of 1 allowed both softmax and perceptron to converge swiftly, but with perceptron having slightly compromised accuracy and both displaying instabilities in accuracy and loss. Implementing a learning rate of 1/k enhanced stability but slowed convergence, impacting softmax model’s final accuracy. A cyclic learning rate, oscillating from 1/1 to 1/50 and then resetting, was most consistent, merging swift convergence with greater softmax stability, and notably increasing perceptron accuracy. Lowering the learning rate to 0.01 got rid of the unstable gradient descent on perceptron loss.

Given what I’ve observed, I’d probably prefer to go with the softmax loss and some form of scheduled learning rate for this problem, and initial weights are fine at 1.

I chose 2000 max iterations to give plenty of time for convergence.

I went with initial weights of 1 because it seemed to work well.

I went with constant 0.01 learning rate on perceptron because it has good stability.
I chose the resetting/diminishing learning rate for softmax because it gave a good accuracy. A constant learning rate of 1 worked well too but this one avoids instabilities in loss which makes the graph more smooth.

For the plots shown below:
softmax: 96.28% accuracy with 26/699 misclassifications
perceptron: 95.57% accuracy with 31/699 misclassifications