HW 6 - SVM, Logistic regression, & SGD

Michael Young - u0742759

Q1) Logistic regression

1.a What is the derivative of $g(w) = \log(1 + e^{-y_i w^T x_i})$ with respect to the weight vector w?

We can evaluate this derivative using the chain rule:

$$\frac{d}{dx} log(x) = \frac{1}{x}$$
$$\frac{d}{dx} e^x = e^x \frac{d}{dx} x$$
$$\frac{d}{dw} g(w) = \frac{1}{1 + e^{-y_i w^T x_i}} e^{-y_i w^T x_i} (-y_i x_i)$$
$$= \frac{e^{-y_i w^T x_i}}{1 + e^{-y_i w^T x_i}} (-y_i x_i)$$

Also worth noting that another way to represent this is:

$$sigmoid(x) = \delta(x) = \frac{1}{1 + e^{-y_i w^T x_i}}$$
$$\frac{d}{dw} g(w) = (1 - \delta(x))(-y_i x_i)$$

1.b Write down the objective where the entire dataset is composed of single example, say (x_i, y_i).

The original objective had a sum term which was meant to sum the losses over the entire dataset. Because we are now only considering a single example to compute our gradient, we no longer use the sum term, thus, the new objective is:

$$J(w) = \min_w [log(1 + e^{-y_i w^T x_i}) + \frac{1}{\sigma^2} w^T w]$$

1.c Derive the gradient of the SGD object for a single example (from the previous question) with respect to the weight vector.

The gradient of this objective is the gradient of both terms with respect to w. The gradient of the first term was found in 1.a, and the gradient of the second "regularizer" term is simply $2w$:

$$\begin{aligned}\frac{dJ(w)}{dw} &= \frac{e^{-y_i w^T x_i}}{1+e^{-y_i w^T x_i}}(-y_i x_i) + \frac{2}{\delta^2}w \\ &= (1-\delta(x))(-y_i x_i) + \frac{2}{\delta^2}w\end{aligned}$$

1.d Write down the pseudo code for the stochastic gradient algorithm using the gradient from the previous part.

```
1. Initialize $w^0 = 0 \in R^n$.
2. For epoch = $1 \ldots T$.
    1. Pick a random example $(x_i, y_i)$ from the training set.
    2. Treat $(x_i, y_i)$ as a full dataset and compute:
       $\frac{dJ(w^{t-1})}{dw} = \frac{e^{-y_i w^T x_i}}{1+e^{-y_i w^T x_i}}(-y_i x_i) + \frac{2}{\delta^2}w$
    3. Update: $w^t = w^{t-1} - \gamma_t(\frac{dJ(w^{t-1})}{dw})$, where $\gamma_t$ is the learning rate.
3. Return final $w$.
```

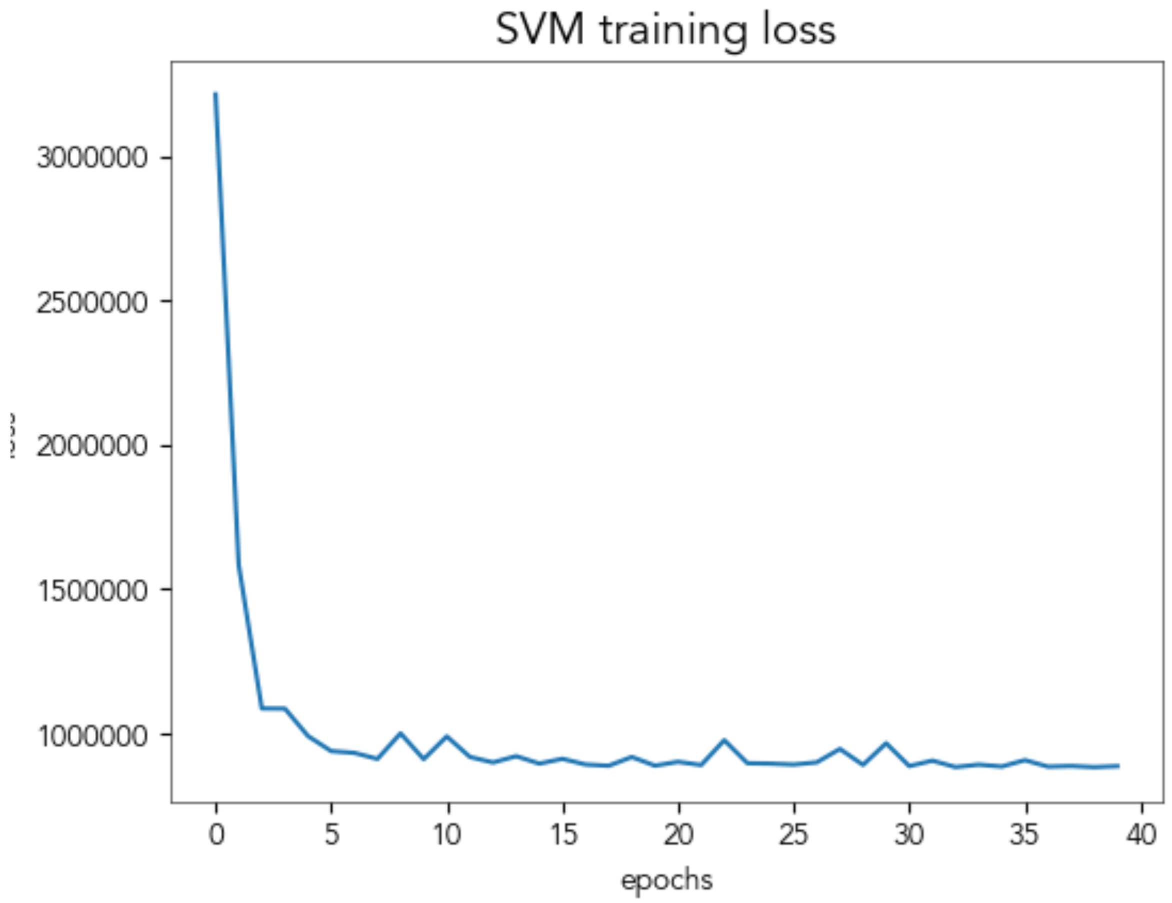
Q2) Experiments

	Best hyper parameters	Average CV accuracy	Training accuracy	Test accuracy
SVM	LR: 0.0001 C: 1000	0.7816	0.8561	0.8190
Logistic Regression	LR: 0.01 C: 1000	0.7525	0.8278	0.7885
SVM over trees	LR: 1e-05 C: 1000 D: 4	0.8219	0.8749	0.8459

All CV trails were run with 15 epochs, and all training was done with 40 epochs. These epochs were chosen based on visual inspection of the loss curves associated with each algorithm. Across all of them, 15 seemed to be at or near the "elbow" of these plots.

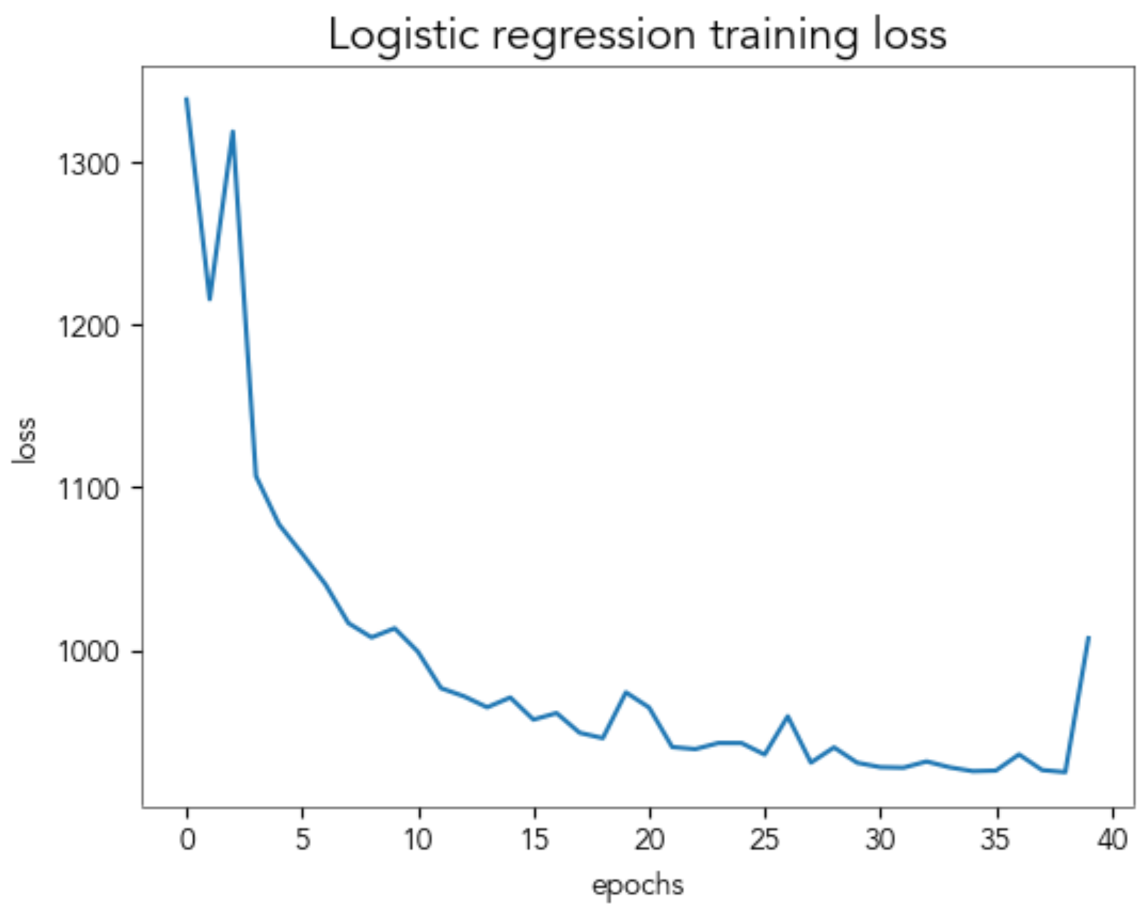
SVM

The results of the SVM CV and training/testing are shown in the table above, and the loss curve is shown below. My implementation was very similar to the Perceptron implementation, the only difference being using the SVM update rule instead of the perceptron. I initially had my weights initializing to 0, but after seeing a comment on the discussion board, I changed to my weights to initialize to a random number between -0.01 and 0.01. There didn't seem to be a huge difference between the two.



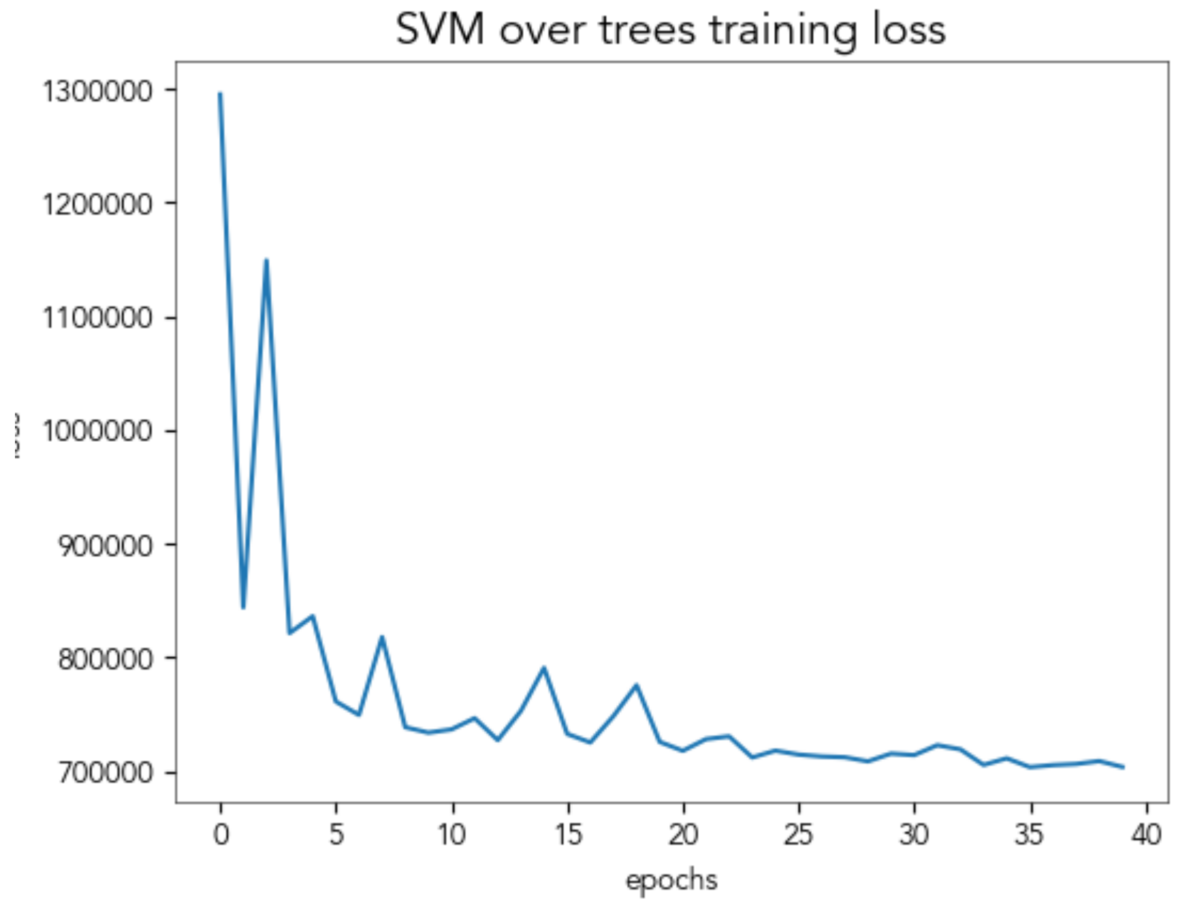
Logistic Regression

The results of my logistic regression CV and training/testing are shown in the table above, and the loss curve is shown below. I implemented the stochastic gradient descent update according to my answers from part 1. I noticed that my loss + accuracy curves looked much smoother if I initialized all my weights to 0 instead of random values, so I chose to initialize them to 0 in my final implementation.



SVM over trees

The results of my SVM over trees implementation CV and training/testing are shown in the table above and the loss curve is shown below. To implement this ensemble, I made a function that took as input the original training data and test data (for CV, the validation data), then it built the 200 trees with a specified depth and returned the transformed feature representation of the training and test data. Then, I fed this new data into my SVM implementation. Examining the results, it appears this strategy was able to bump up the training accuracy by ~3%.



Q3) Neural Networks and the XOR function (extra cred)

1. Find a set of weights for 2-layer network w/ threshold activation that can correctly classify XOR function.

The following weights for the 2-layer network correctly classify the XOR function (determined them by just staring at the problem for way too long):

$$W^1_{00}, W^1_{10}, W^1_{20} = [1, 1, 1]$$
$$W^1_{01}, W^1_{11}, W^1_{21} = [-1, -1, 1]$$
$$W^o_{00}, W^o_{10}, W^o_{20} = [1, 1, -1]$$

Plugging these weights into our network we get:

$$h_1 = sign(x_1 W^1_{00} + x_2 W^1_{10} + (1) W^1_{20}) = sign(x_1 + x_2 + 1)$$
$$h_2 = sign(x_1 W^1_{01} + x_2 W^1_{11} + (1) W^1_{21}) = sign(-x_1 + -x_2 + 1)$$
$$y = sign(h_1 W^o_{00} + h_2 W^0_{10} + (1) W^o_{20}) = sign(h_1 + h_2 - 1)$$

We can determine that these weights correctly classify an XOR function by classifying each case:

$$\mathbf{x_1, x_2 = (-1, -1)}$$
$$h_1 = -1, h_2 = +1$$
$$y = -1$$
$$\mathbf{x_1, x_2 = (-1, +1)}$$
$$h_1 = +1, h_2 = +1$$
$$y = +1$$
$$\mathbf{x_1, x_2 = (+1, -1)}$$
$$h_1 = +1, h_2 = +1$$
$$y = +1$$
$$\mathbf{x_1, x_2 = (+1, +1)}$$
$$h_1 = +1, h_2 = -1$$
$$y = -1$$

2. Now find a set of weights using a sigmoid activation instead of a threshold activation.

We can use essentially the same weight as before, but with a different bias weight leading into our output:

$$W_{00}^1, W_{10}^1, W_{20}^1 = [1, 1, 1]$$
$$W_{01}^1, W_{11}^1, W_{21}^1 = [-1, -1, 1]$$
$$W_{00}^o, W_{10}^o, W_{20}^o = [1, 1, -1.3]$$

Plugging these weights into our network we get:

$$h_1 = \sigma(x_1 W_{00}^1 + x_2 W_{10}^1 + (1) W_{20}^1) = \sigma(x_1 + x_2 + 1)$$
$$h_2 = \sigma(x_1 W_{01}^1 + x_2 W_{11}^1 + (1) W_{21}^1) = \sigma(-x_1 + -x_2 + 1)$$
$$y = \sigma(h_1 W_{00}^o + h_2 W_{10}^0 + (1) W_{20}^o) = \sigma(h_1 + h_2 - 1.3)$$

Assuming that when $y \leq 0.5 = -1$ and $+1$ otherwise, plugging in the cases we get:

$$\sigma(-1) + \sigma(3) \approx 1.22$$
$$\sigma(1) + \sigma(1) \approx 1.46$$
$$\mathbf{x_1, x_2 = (-1, -1)}$$
$$h_1 = \sigma(-1), h_2 = \sigma(3)$$
$$y = \sigma(\sigma(-1) + \sigma(3) - 1.3) = \sigma(1.22 - 1.3) \approx 0.48 = -1$$
$$\mathbf{x_1, x_2 = (-1, +1)}$$
$$h_1 = \sigma(1), h_2 = \sigma(1)$$
$$y = \sigma(1.46 - 1.3) \approx 0.54 = +1$$
$$\mathbf{x_1, x_2 = (+1, -1)}$$
$$h_1 = \sigma(1), h_2 = \sigma(1)$$
$$y = \sigma(1.46 - 1.3) \approx 0.54 = +1$$
$$\mathbf{x_1, x_2 = (+1, +1)}$$
$$h_1 = \sigma(3), h_2 = \sigma(-1)$$
$$y = \sigma(1.22 - 1.3) \approx 0.48 = -1$$