Homework 3

A1.

(a)

False. Neural networks are non-convex, and for large networks, gradient decent can blow up or go to zero.

(b)

False. Initializing all weights to zero can cause all the neurons on the same layer to have the same weights and receive the same gradients updates, which is not ideal since we want the neurons to learn different features from the input.

(c)

True. If we don't have non-linearity, then the network would behave like a linear model regardless of the number of layers, and we wouldn't be able to learn complex (non-linear) decision boundaries from the input.

(d)

False. If we use a recursive algorithm where we store the intermediate results and errors of the forward pass to use in the backward pass, the time complexities of forward and backward passes are roughly the same.

(e)

False. The model we choose is dependant on the specific problem, the amount of data, computational resources etc. If we have a small dataset with a simple linear relationship, using a simple linear model would potentially produce better results than using neural networks where we might easily overfit the data.

A2.

(a)

$$
\begin{aligned}
\phi(x) \cdot \phi(x') &= \sum_{i=0}^{\infty} \left(\frac{1}{\sqrt{i!}} e^{-\frac{x^2}{2}} x^i \right) \left(\frac{1}{\sqrt{i!}} e^{-\frac{x'^2}{2}} x'^i \right) \\
&= \sum_{i=0}^{\infty} \frac{1}{i!} e^{-\frac{x^2}{2}} e^{-\frac{x'^2}{2}} x^i x'^i \\
&= e^{-\frac{x^2}{2}} e^{-\frac{x'^2}{2}} \sum_{i=0}^{\infty} \frac{x^i x'^i}{i!} \\
&= e^{-\frac{x^2}{2}} e^{-\frac{x'^2}{2}} e^{xx'} && \text{(Taylor series expansion of } e^{xx'}\text{)} \\
&= e^{-\frac{x^2}{2} - \frac{x'^2}{2} + xx'} \\
&= e^{-\frac{x^2 + x'^2 - 2xx'}{2}} \\
&= e^{-\frac{(x-x')^2}{2}}
\end{aligned}
$$

Therefore, $K(x, x') = e^{-\frac{(x-x')^2}{2}}$ is a kernel function for the feature map ϕ, since we have shown that $\phi(x) \cdot \phi(x') = e^{-\frac{(x-x')^2}{2}} = K(x, x')$.

A3.

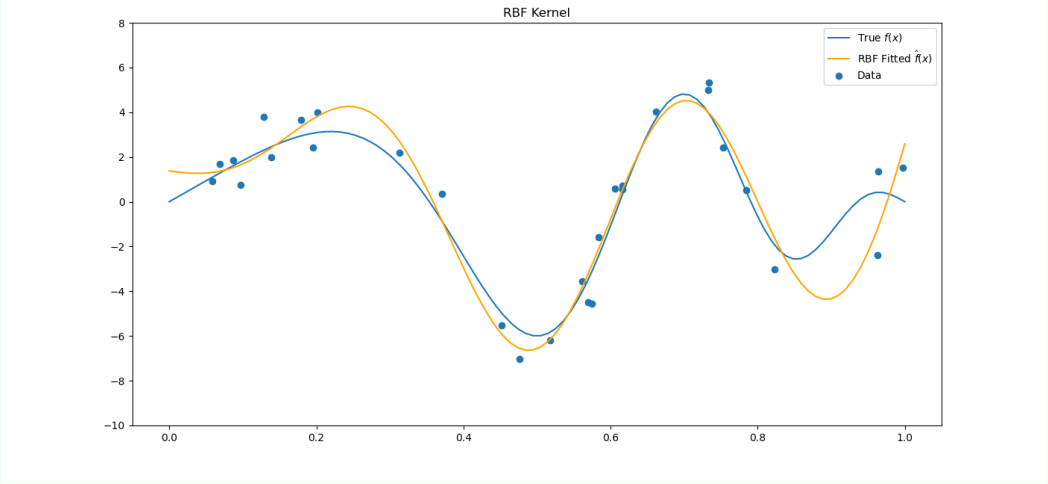
(a)

Grid search, Leave-One-Out cross-validation:

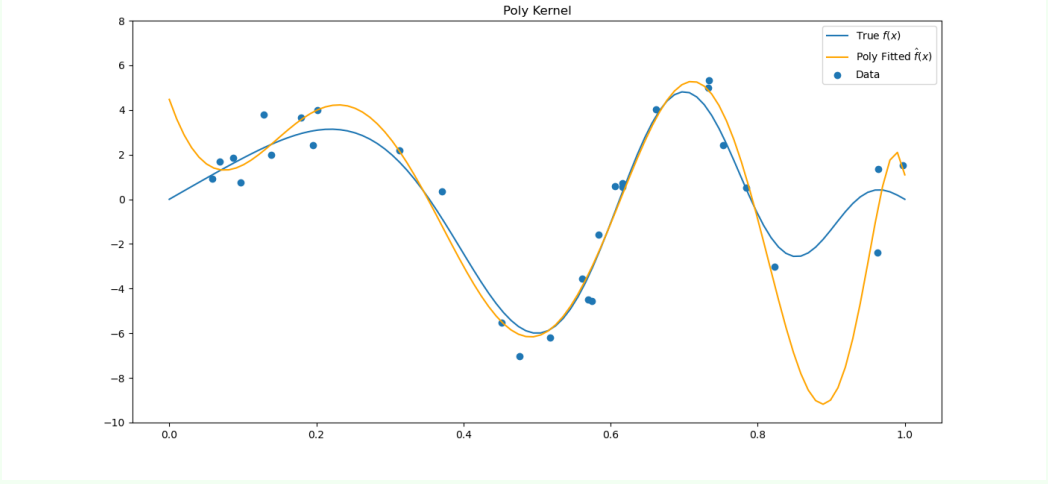
- **For RBF Kernel:** $\lambda = 0.002329951810515372$, $\gamma = 11.201924992299844$ (by the heuristic provided)
- **For Polynomial Kernel:** $\lambda =$ 3.727593720314938e-5, $d = 16$

(b)

$\hat{f}_{\text{rbf}}(x)$:



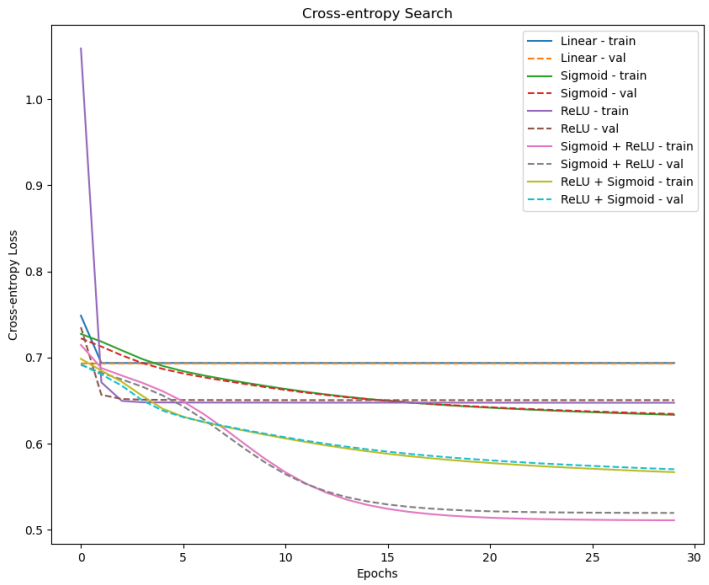
$\hat{f}_{\text{poly}}(x)$:



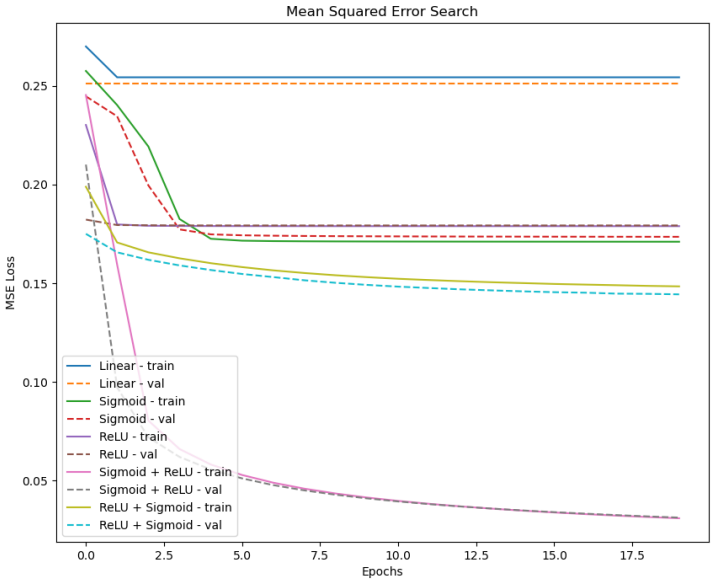
A4.

(b)

Cross-entropy search (Learning rate = 1e-3, Epochs = 30):

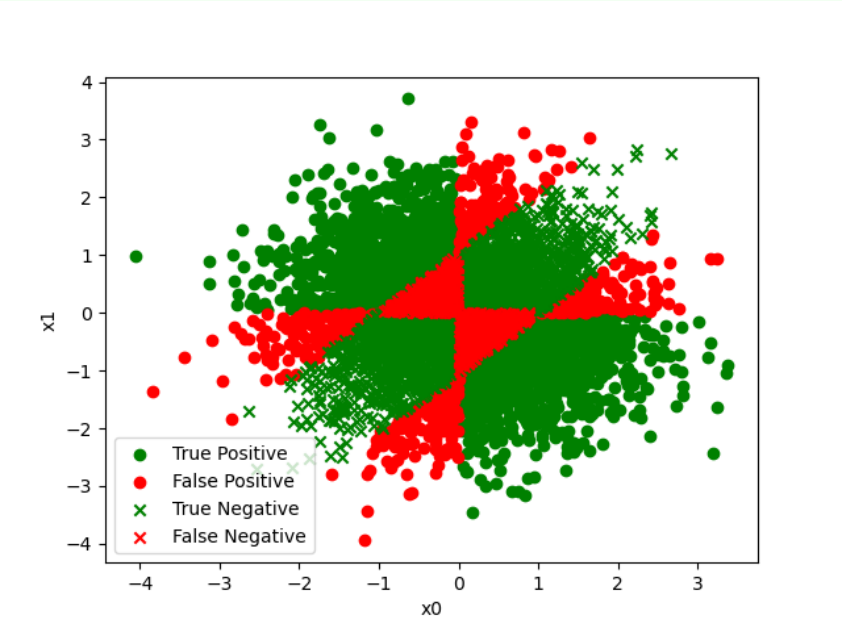


Mean Squared Error search (Learning rate = 1e-2, Epochs = 20):



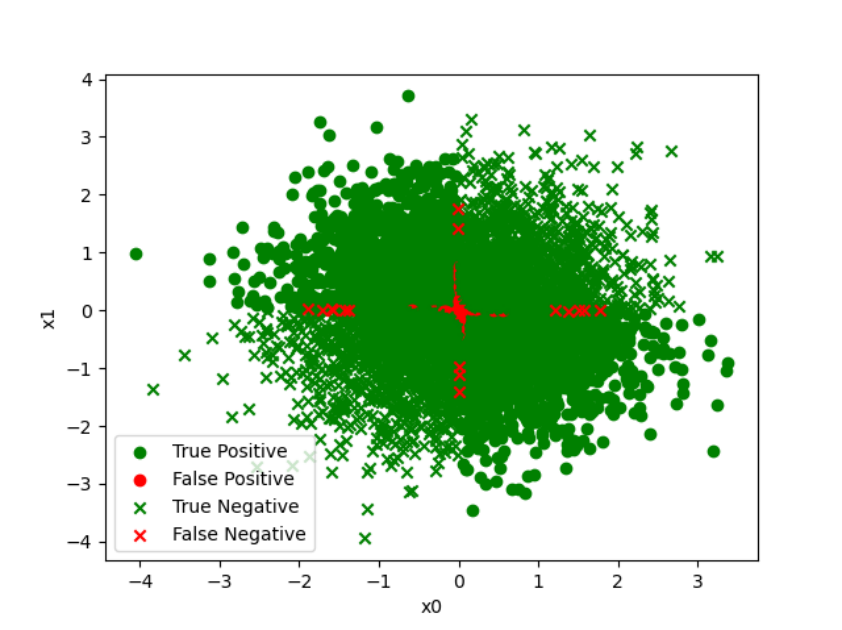
(c)

Best Performing Architecture for Cross-entropy loss: NN with two hidden layer (each with 2 units) and Sigmoid, ReLU activation functions after first and second hidden layers, respectively.



Accuracy on test set: 0.75

Best Performing Architecture for MSE loss: NN with two hidden layer (each with 2 units) and Sigmoid, ReLU activation functions after first and second hidden layers, respectively.

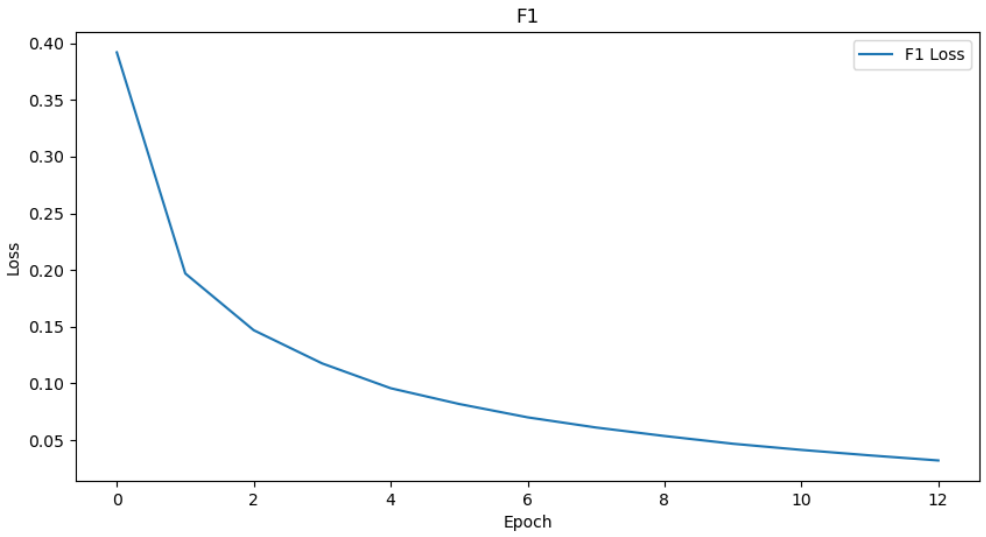


Accuracy on test set: 0.98

A5.

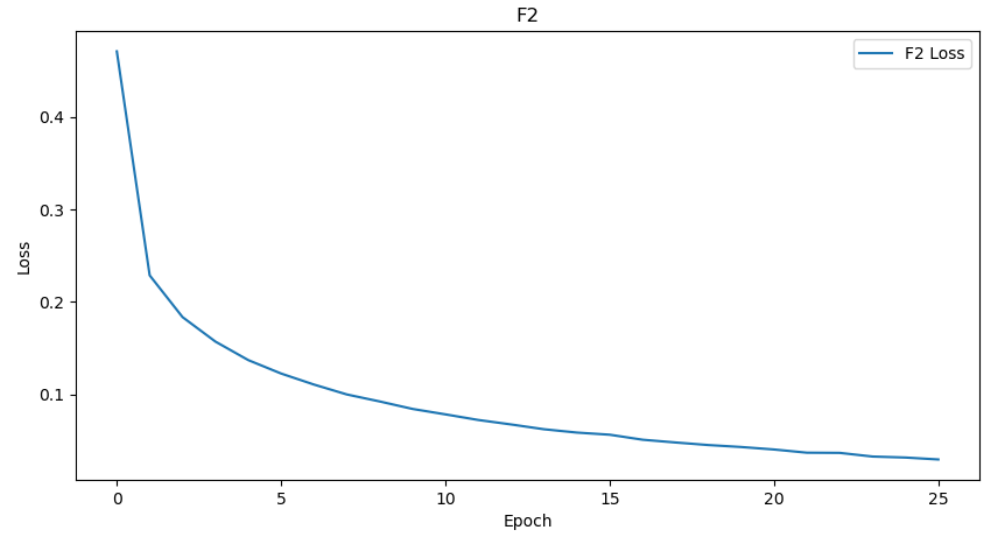
(a)

F1: Test Loss: 0.0829, Test Accuracy: 97.64%



(b)

F2: Test Loss: 0.1482, Test Accuracy: 96.47%



(c)

Number of Parameters in F1: 50890
Number of Parameters in F2: 26506

The F1 network, which is wide and shallow, was more accurate (performed better) than the narrow and deep F2. F1 has more parameters giving it higher degrees of flexibility to fit the data, which could be why it learned better for this task. Since the MNIST images are relatively simple, having a wider network like F1 seems to be enough to recognize the digits well, and making the network deeper (e.g. F2) didn't help as much.

A6.

(a)

20 hours