CPSC 663 Assignment 1

Author: Wenxin Xu

The `xuw_assignment.zip` file includes following files:

- `xuw_assignment1.pdf` : A detailed report.
- `/code`
    - `ps1_functions.py`: contains 3 functions.
    - `xuw_assignment1.ipynb`: A Jupyter Notebook contains all the code.

Problem 1

1. Characteristics of a ML algorithm:

Experience, performance measure and task. A ML algorithm is a computer program that can learn from original dataset/experience (E) to improve performance (P) on a task (T).

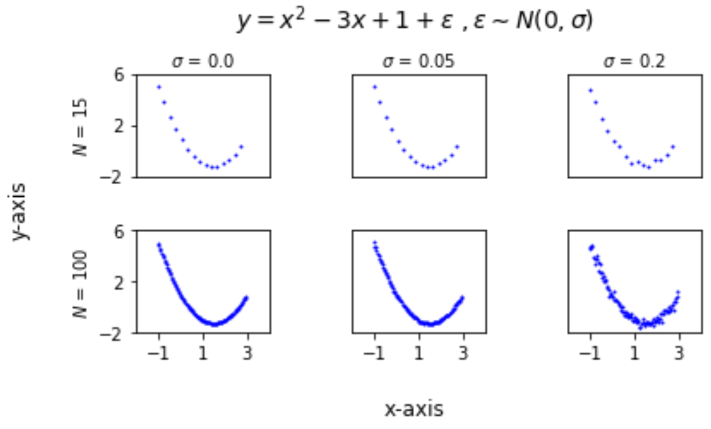
2. Meaning of "learning" from data.

Finding latent patterns in data without explicit instruction.

Problem 2 Polynomial

Q 2.1 Generate random points from quadratic function with noise.

Randomly generates N points sampled uniformly in the interval $x \in [-1, 3]$. Then output the function $y = x^2 - 3x + 1$ for each of the points generated. Then adds zero-mean Gaussian noise with standard deviation u to y. Make plots of x and y with $N \in \{15, 100\}$ and $\sigma \in \{0, 0.05, 0.2\}$



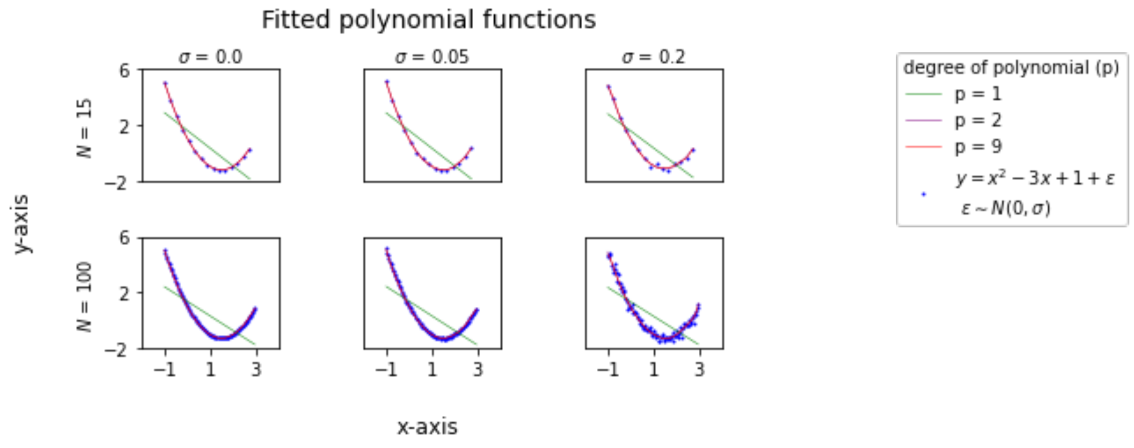
Q 2.2 Fit polynomial functions of different degrees to data

- Find the optimal weights in terms of MSE for fitting a polynomial function to the data in all 6 cases generated above using a polynomial of degree 1, 2, and 9. Use the least squares analytical solution given below.

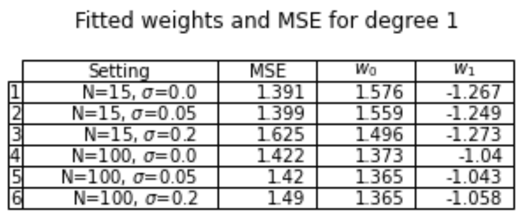
$$\text{least square solution } \mathbf{w}^* = (X^T X)^{-1} X^T \mathbf{y}$$

$$\text{L2 norm } \|x\|_2 = \left(\sum_{i=1}^{n} |x_i|^2\right)^{\frac{1}{2}}$$

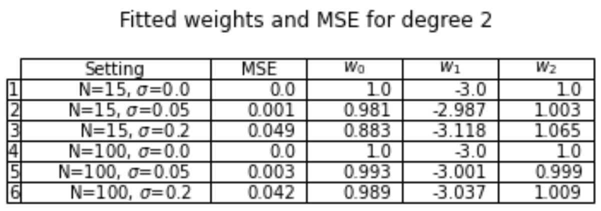
- **Plot the fitted curves on the same plot as the data points.**



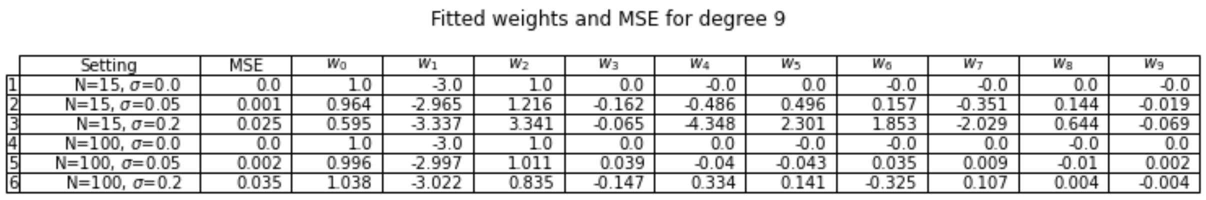
- **Report the fitted weights and the MSE in tables.**


Fitted weights and MSE for degree 1

	Setting	MSE	w_0	w_1
1	N=15, σ=0.0	1.391	1.576	-1.267
2	N=15, σ=0.05	1.399	1.559	-1.249
3	N=15, σ=0.2	1.625	1.496	-1.273
4	N=100, σ=0.0	1.422	1.373	-1.04
5	N=100, σ=0.05	1.42	1.365	-1.043
6	N=100, σ=0.2	1.49	1.365	-1.058


Fitted weights and MSE for degree 2

	Setting	MSE	w_0	w_1	w_2
1	N=15, σ=0.0	0.0	1.0	-3.0	1.0
2	N=15, σ=0.05	0.001	0.981	-2.987	1.003
3	N=15, σ=0.2	0.049	0.883	-3.118	1.065
4	N=100, σ=0.0	0.0	1.0	-3.0	1.0
5	N=100, σ=0.05	0.003	0.993	-3.001	0.999
6	N=100, σ=0.2	0.042	0.989	-3.037	1.009


Fitted weights and MSE for degree 9

	Setting	MSE	w_0	w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8	w_9
1	N=15, σ=0.0	0.0	1.0	-3.0	1.0	0.0	-0.0	0.0	-0.0	-0.0	0.0	-0.0
2	N=15, σ=0.05	0.001	0.964	-2.965	1.216	-0.162	-0.486	0.496	0.157	-0.351	0.144	-0.019
3	N=15, σ=0.2	0.025	0.595	-3.337	3.341	-0.065	-4.348	2.301	1.853	-2.029	0.644	-0.069
4	N=100, σ=0.0	0.0	1.0	-3.0	1.0	0.0	0.0	-0.0	-0.0	0.0	-0.0	0.0
5	N=100, σ=0.05	0.002	0.996	-2.997	1.011	0.039	-0.04	-0.043	0.035	0.009	-0.01	0.002
6	N=100, σ=0.2	0.035	1.038	-3.022	0.835	-0.147	0.334	0.141	-0.325	0.107	0.004	-0.004

- **Qualitatively assess the fit of the curves.**

When degree is 1, the model is underfitting because it doesn't go through most of the data points, also it makes sense that a linear function can't fit data generated from a quadratic function.

When degree is 2, the model appropriately fit the data, because it goes through lots of data points and the original function is exactly quadratic.

When degree is 9, the model is overfitting, it wiggly and perfectly goes through most data points, but we don't need so many parameters for fitting a quadratic function.

Q 2.3 Fit polynomial function of 9-degree with L2-norm regularization to data

- Apply L-2 norm regularization with a 9-degree polynomial model to the cases with $\sigma = 0.2$ and $N \in \{15,100\}$.

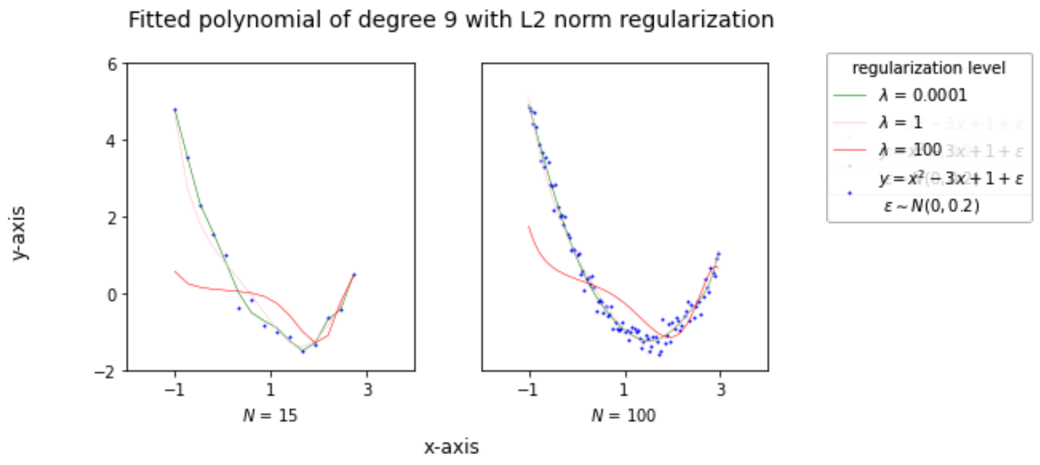
least square solution of Ridge regresssion:

$$\mathbf{w}^* = (\lambda I + X^T X)^{-1} X^T \mathbf{y}$$

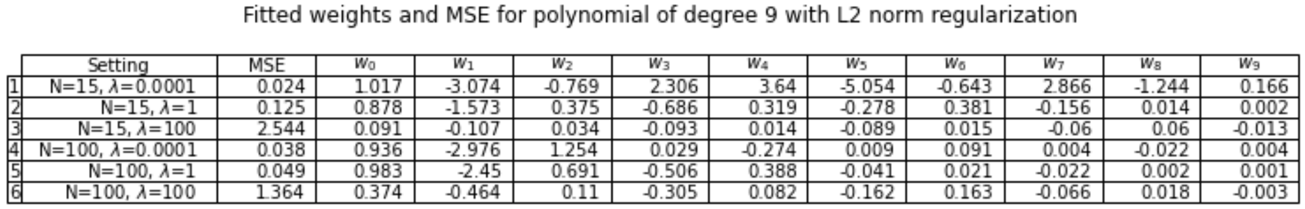
where $\lambda > 0$ is a regularization factor

- **Vary the parameter λ, and choose three values of λ that result in the following scenarios: underfitting, overfitting, and an appropriate fit.**

  When $\lambda = 0.0001$, the model is overfitting.
  When $\lambda = 1$, the model is an appropriate fit.
  When $\lambda = 100$, the model is underfitting.



- **Report the fitted weights and the MSE in each of these scenarios.**


Fitted weights and MSE for polynomial of degree 9 with L2 norm regularization

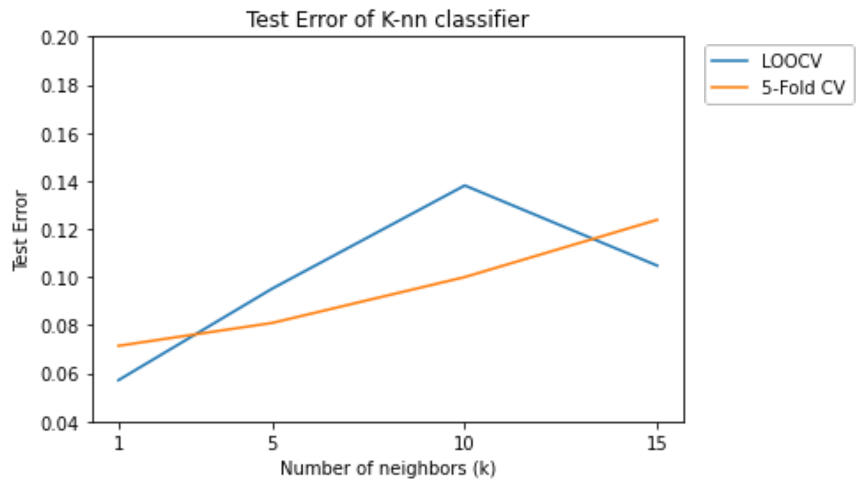
	Setting	MSE	w_0	w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8	w_9
1	N=15, λ=0.0001	0.024	1.017	-3.074	-0.769	2.306	3.64	-5.054	-0.643	2.866	-1.244	0.166
2	N=15, λ=1	0.125	0.878	-1.573	0.375	-0.686	0.319	-0.278	0.381	-0.156	0.014	0.002
3	N=15, λ=100	2.544	0.091	-0.107	0.034	-0.093	0.014	-0.089	0.015	-0.06	0.06	-0.013
4	N=100, λ=0.0001	0.038	0.936	-2.976	1.254	0.029	-0.274	0.009	0.091	0.004	-0.022	0.004
5	N=100, λ=1	0.049	0.983	-2.45	0.691	-0.506	0.388	-0.041	0.021	-0.022	0.002	0.001
6	N=100, λ=100	1.364	0.374	-0.464	0.11	-0.305	0.082	-0.162	0.163	-0.066	0.018	-0.003

Problem 3 KNN classifier

Q 3.1 knn classifier

- Write a program that applies a k-nn classifier to the data with $k \in \{1, 5, 10, 15\}$. Calculate the test error using both leave-one-out validation and 5-fold cross validation.

- **Plot the test error as a function of k.**



- **Do any values of k result in underfitting or overfitting?**

From the plot of test error vs. k, we can see as k increases, the test error also increases no matter which cross validation methods we choose (either Leave-one-out or 5-fold). Given the labels are 3 classes with equal numbers, this dataset is balanced, thus the baseline classification accuracy is 1/3. All the choices of k achieve test accuracy higher than baseline. When k = 1, model is overfitting. When k = 5 or 10, model fits well. When k = 15, model is underfitting.

Q 3.2 Other classifiers

- Apply **two other classifiers** of your choice to the same data, include but are not limited to logistic regression, QDA, naive Bayes, SVM, and decision trees.

  I choose SVM and decision tree.

Classifier 1: SVM

- Sklearn's SVM classifier `Sklearn.svm.svc` has 3 main hyperparameters:

  - `kernel`: {`'linear', 'poly', 'rbf', 'sigmoid', 'precomputed'`} or callable, default=`'rbf'`
  - `C` : `C` > 0, float, default=`1.0`. L2 norm regularization parameter. The strength of the regularization is inversely proportional to `C`.

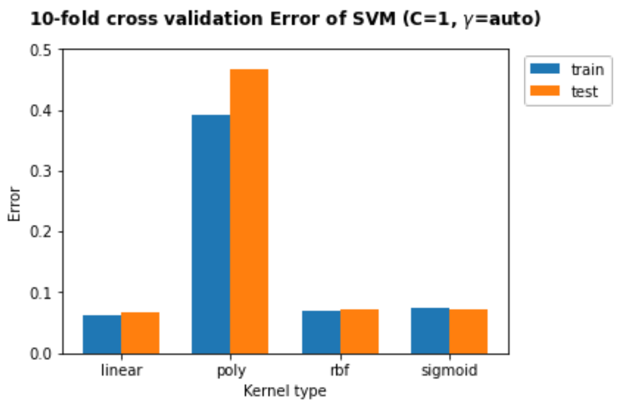
- `gamma`: {`'scale'`, `'auto'`} or float, default=`'scale'`. Kernel coefficient for `'rbf'`, `'poly'` and `'sigmoid'`. if gamma=`'scale'` (default) is passed then it uses `1 / (n_features * X.var())` as value of `gamma`; if `'auto'`, uses `1 / n_features`.

- **If any tuning parameters need to be selected, use cross-validation and report the training and test error for several values of the tuning parameters.**

  - Tuning parameter `kernel`

  Fixing `C` = 1, `gamma` = '`auto`', searching `kernel` in {linear, poly, rbf, sigmoid} using 10-fold cross validation

  Result: the best `kernel` is linear kernel with 0.06 test error.

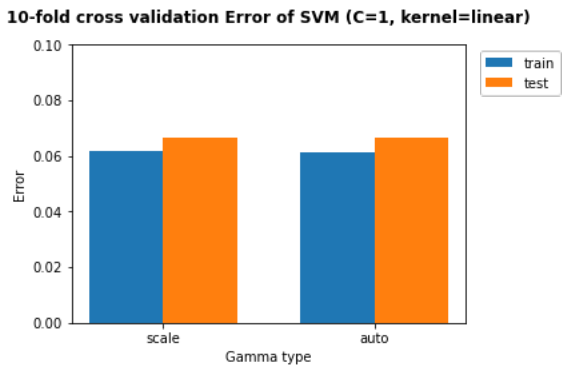


	Kernel	Training error	Test error
0	linear	0.061376	0.066667
1	poly	0.391005	0.466667
2	rbf	0.068783	0.071429
3	sigmoid	0.074074	0.071429

  - Tuning parameter `gamma`

  Fixing `C` = 1, linear kernel, using 10-fold cross validation to select `gamma` to be either '`scale`' or `'auto'`

  Result: no difference between `gamma` to be '`scale`' or '`auto`', both have 0.07 test error. Finally, I use `gamma`='`auto`' in the next step to tune C.

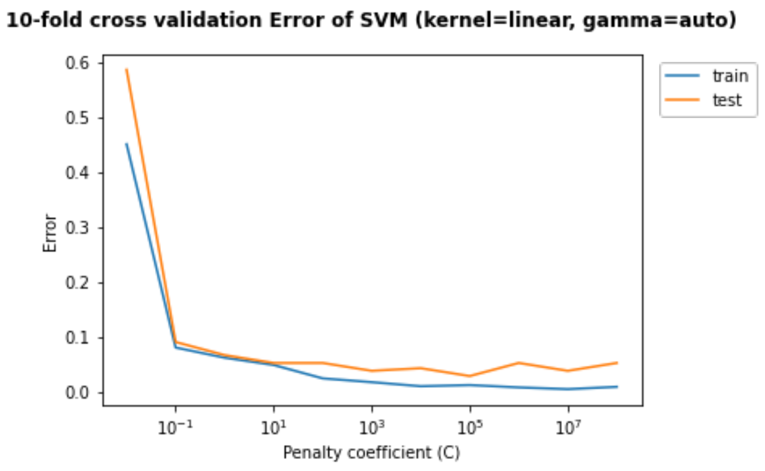


	Gamma	Training error	Test error
0	scale	0.061905	0.066667
1	auto	0.061376	0.066667

  - Tuning parameter `C`

Fixing kernel type to be linear, gamma='auto', searching C in $\{10^{-2}, 10^{-1}, 10^{0}, 10^{1}, 10^{2}, 10^{3}, 10^{4}, 10^{5}, 10^{6}, 10^{7}, 10^{8}\}$ using 10-fold cross validation.

Result: optimal C is 10^3 with test error 0.038.



	C	Training error	Test error
0	1e-02	0.514815	0.638095
1	1e-01	0.083069	0.085714
2	1e+00	0.061905	0.066667
3	1e+01	0.048148	0.061905
4	1e+02	0.021693	0.066667
5	1e+03	0.016402	0.047619
6	1e+04	0.010582	0.038095
7	1e+05	0.011640	0.042857
8	1e+06	0.008995	0.042857
9	1e+07	0.004762	0.038095
10	1e+08	0.008995	0.042857

- **Use 5-fold cross validation to calculate the test error. Report the training and test errors.**

  After tuning, run SVM classifier with `C=1000`, linear kernel and `gamma='auto'` on dataset using 5-fold cross validation. Training error is 0.015, test error is 0.038.

Classifier 2: Decision tree

Sklean's decision tree classifier `sklearn.tree.DecisionTreeClassifier` has 2 main hyperparameters:

- `min_sample_split`: The minimum number of samples required to split an internal node. default=2.

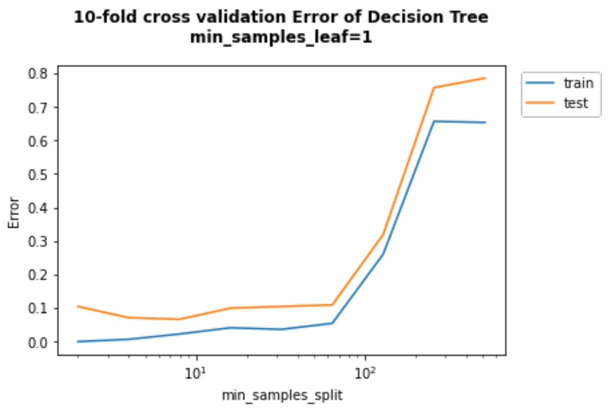
- `min_samples_leaf`: The minimum number of samples required to be at a leaf node. default=1.

- **If any tuning parameters need to be selected, use cross-validation and report the training and test error for several values of the tuning parameters.**

  - Tuning `min_samples_splits`

  Fixing `min_samples_leaf` to be 1, searching `min_samples_split` in {2, 4, 8, 16, 32, 64, 128, 256, 512} by 10-fold CV.

  Result: the optimal `min_samples_splits` is 8 with smallest test error 0.067.

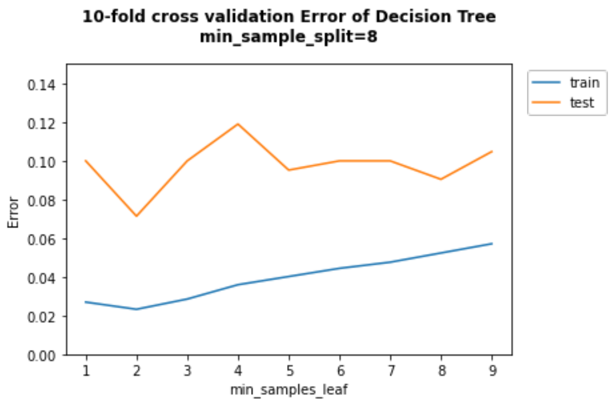


	min_samples_split	Training error	Test error
0	2	0.000000	0.104762
1	4	0.006878	0.071429
2	8	0.022751	0.066667
3	16	0.041270	0.100000
4	32	0.036508	0.104762
5	64	0.054497	0.109524
6	128	0.260317	0.319048
7	256	0.656614	0.757143
8	512	0.653439	0.785714

- Tuning `min_samples_leaf`

Fixing `min_samples_split` to be 8, searching `min_samples_leaf` in {1, 2, 3, 4, 5, 6, 7, 8, 9} by 10-fold CV.

Result: the optimal min_samples_leaf is 2 with smallest test error 0.071.



	min_samples_leaf	Training error	Test error
0	1	0.026984	0.100000
1	2	0.023280	0.071429
2	3	0.028571	0.100000
3	4	0.035979	0.119048
4	5	0.040212	0.095238
5	6	0.044444	0.100000
6	7	0.047619	0.100000
7	8	0.052381	0.090476
8	9	0.057143	0.104762

- **Use 5-fold cross validation to calculate the test error. Report the training and test errors.**

  After tuning, run decision tree on dataset with min_samples_split=8 and min_samples_leaf=2, training error is 0.032, test error is 0.105.

- **How do they compare to the k-nn classifiers in terms of performance?**

  SVM classifier performs best with smallest test error 0.038. KNN underfit the data and Decision Tree overfit the data.

Classifier	Training error	Test error	Hyperparameters
KNN	/	.057	K = 5
SVM	.015	.038	Kernel = ’linear’, C = 100, gamma = ’auto’
Decision tree	.032	.105	min_samples_split=8, min_samples_leaf=2

Problem 4 Perceptron

Q 4.1 Suppose we take all the weights and biases in a network of perceptrons, and multiply them by a positive constant, c > 0. Show that the behavior of the network doesn't change.

The output of a perceptron is

$$\text{output} = \begin{cases} 0 & \text{if } \mathbf{w}^T\mathbf{x} + b \le 0 \\ 1 & \text{if } \mathbf{w}^T\mathbf{x} + b > 0 \end{cases}$$

multiplying all the weights and biases by a constant c ($c > 0$), the output becomes:

$$\text{output} = \begin{cases} 0 & \text{if } c \cdot \mathbf{w}^T\mathbf{x} + c \cdot b = c \cdot (\mathbf{w}^T\mathbf{x} + b) \le 0 \\ 1 & \text{if } c \cdot \mathbf{w}^T\mathbf{x} + c \cdot b = c \cdot (\mathbf{w}^T\mathbf{x} + b) > 0 \end{cases}$$

because the sign of the quantity $c \cdot (\mathbf{w}^T\mathbf{x} + b)$ doesn't change, the output (behavior) of the network doesn't change

Q 4.2 Given the same set up of problem 4.1 - a network of perceptrons, suppose that the overall input to the network of perceptrons has been chosen and fixed. Suppose the weights and biases are such that $wx+b \neq 0$ for the input x to any particular perceptron in the network. Now replace all the perceptrons in the network by sigmoid neurons, and multiply the weights and biases by a positive constant c > 0.

- **Show that in the limit as $c \to \infty$ the behavior of this network of sigmoid neurons is exactly the same as the network of perceptrons.**

The output of a sigmoid neuron is

$$\sigma(\mathbf{w}^T\mathbf{x} + b) = \frac{1}{1 + \exp\left[-(\mathbf{w}^T\mathbf{x} + b)\right]}$$

- Replacing all perceptrons in the network by sigmoid neurons and multiply the weights and biases by a positive constant $c > 0$,

  for a neuron, we have:

$$\sigma(c(\mathbf{w}^T\mathbf{x} + b)) = \frac{1}{1 + \exp\left[-c(\mathbf{w}^T\mathbf{x} + b)\right]}$$

- Suppose $\mathbf{w}^T\mathbf{x} + b \neq 0$, as $c \to \infty$

  If $\mathbf{w}^T\mathbf{x} + b < 0$,

$$\frac{1}{1 + \exp\left[-c(\mathbf{w}^T\mathbf{x} + b)\right]} = \frac{1}{1 + \exp(+\infty)} \to \frac{1}{+\infty} = 0$$

  If $\mathbf{w}^T\mathbf{x} + b > 0$,

$$\frac{1}{1 + \exp\left[-c(\mathbf{w}^T\mathbf{x} + b)\right]} = \frac{1}{1 + \exp(-\infty)} \to \frac{1}{1 + 0} = 1$$

  which is same as the behavior of the network of perceptrons.

- **How can this fail when wx + b = 0 for one of the perceptrons?**

When $\mathbf{w}^T\mathbf{x} + b = 0$ for one of the perceptrons, then

$$\sigma(c(\mathbf{w}^T\mathbf{x} + b)) = \sigma(0) = \frac{1}{1 + \exp(0)} = \frac{1}{2}$$

which is different from the behavior of perceptron, so it fails.

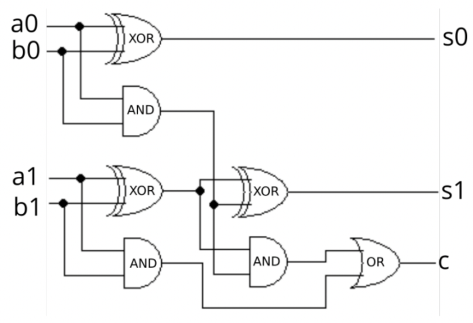
Q 4.3 For each possible input of the MLP in Figure 1, calculate the output. I.e., what is the output if X = [0,0, 0], X = [0, 0,1], etc.

Output is 0, 1, 1, 1, 1, 0, 1, 1 for 8 cases respectively.

Q 4.4 If we change the perceptrons in Figure 1 to sigmoid neurons, what are the outputs for the same inputs (e.g., inputs of [0,0,0], [0,0,1], ...)?

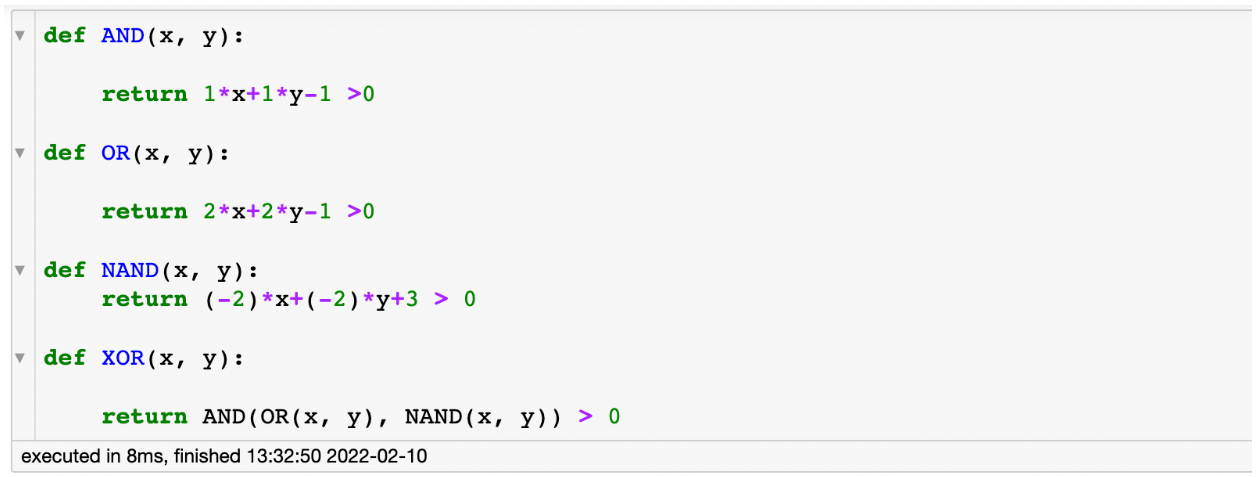
Output is 0.569265, 0.58501229, 0.62245933, 0.63314399, 0.56986717, 0.57508402, 0.61732588, 0.62831133 for 8 cases respectively.

Q 4.5 Using perceptrons with appropriate weights and biases, design an adder that does two-bit binary addition. That is, the adder takes as input 2 two-bit binary numbers (i.e. 4 binary inputs) and adds them together. Don't forget to include the carry bit. The resulting output should be the two-bit sum and the carry bit for a total of three binary outputs.



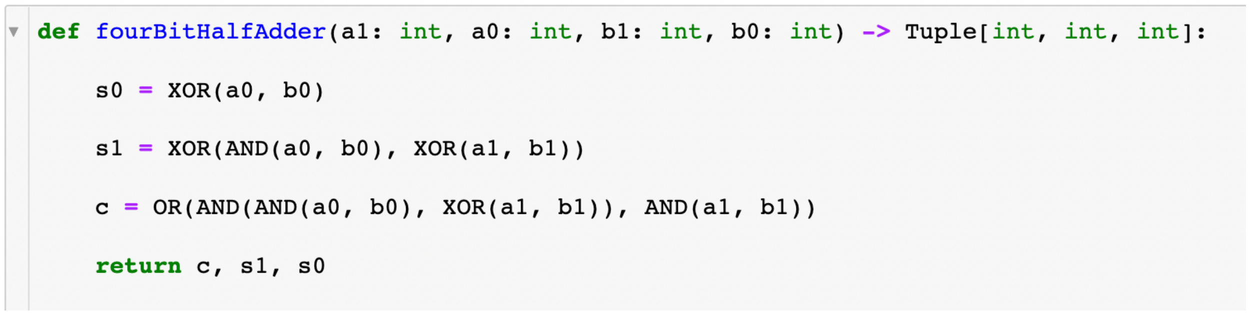
Truth table

Inputs				Outputs		
A1	A0	B1	B0	C	S1	S0
0	0	0	0	0	0	0
0	0	0	1	0	0	1
0	0	1	0	0	1	0
0	0	1	1	0	1	1
0	1	0	0	0	0	1
0	1	0	1	0	1	0
0	1	1	0	0	1	1
0	1	1	1	1	0	0
1	0	0	0	0	1	0
1	0	0	1	0	1	1
1	0	1	0	1	0	0
1	0	1	1	1	0	1
1	1	0	0	0	1	1
1	1	0	1	1	0	0
1	1	1	0	1	0	1
1	1	1	1	1	1	0



```
def AND(x, y):

    return 1*x+1*y-1 >0

def OR(x, y):

    return 2*x+2*y-1 >0

def NAND(x, y):
    return (-2)*x+(-2)*y+3 > 0

def XOR(x, y):

    return AND(OR(x, y), NAND(x, y)) > 0
```

```
executed in 8ms, finished 13:32:50 2022-02-10
```



```
def fourBitHalfAdder(a1: int, a0: int, b1: int, b0: int) -> Tuple[int, int, int]:

    s0 = XOR(a0, b0)

    s1 = XOR(AND(a0, b0), XOR(a1, b1))

    c = OR(AND(AND(a0, b0), XOR(a1, b1)), AND(a1, b1))

    return c, s1, s0
```

Problem 5 Fully-connected neural network

Note: In the following tables, the yellow highlighted hyperparameters are the one with highest test accuracy and decent loss plot.

Q 5.1 Learning rate

Try adjusting the learning rate (by making it smaller) if your model is not converging/improving in accuracy.

Learning rate 0.001 has highest test accuracy of 0.9510.

Table 1: Hyperparameter tuning of learning rate

Fixed: Activation function = ReLU, num_layers=2, hidden_size1 = 64, optimizer=SGD			
	Learning rate	Test accuracy	Occurred after which epoch
1	0.001	0.9510	48
2	0.01	0.6160	Doesn't converge
3	0.02	0.2090	Doesn't converge
4	0.05	0.1170	Doesn't converge

Q 5.2 Activation function

Try training your network without a non-linearity between the layers (i.e. a "linear activation").Then try adding a sigmoid non-linearity, first directly on the input to the first layer, then on the input to the second layer. Experiment with the non-linearity used before the middle layer (relu, softplus, elu, tanh)

Activation function ReLU on the input to the second layer. has highest test accuracy of 0.9545.

Table 2: Hyperparameter tuning of activation functions

Fixed: Learning rate = 0.001, num_layers=2, hidden_size = 64, optimizer=SGD			
	Activation function	Test accuracy	Occurred after which epoch
1	Linear	0.9045	37
2	ReLU	0.9545	46
3	SoftPlus	0.9465	27
4	ELU	0.948	40
5	Tanh	0.921	40
6	Sigmoid on the input to the first layer	0.8625	77
7	Sigmoid on the input to the second layer	0.907	74

Q 5.3 Width of hidden layer.

Try changing the width of the hidden layer, keeping the activation function that performs best.

Hidden size 256 has highest test accuracy of 0.9630.

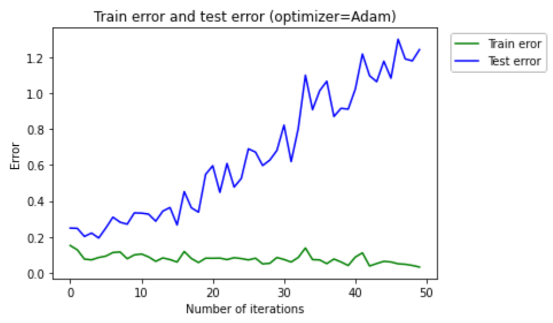
Table 3: Hyperparameter tuning of width of hidden layer

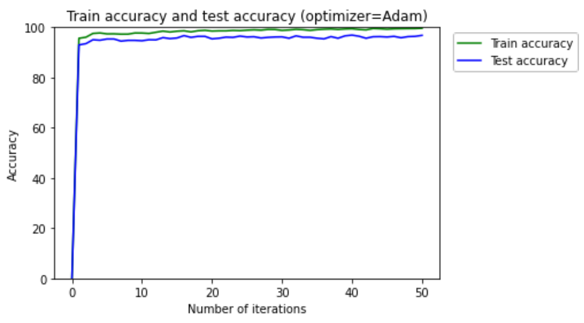
Fixed: Activation function = ReLU, num_layers=2, learning_rate=0.001, optimizer=SGD		
hidden_size1	Test accuracy	Occurred after which epoch
32	0.9445	49
64	0.9530	38
128	0.9570	37
256	0.9630	47

Q 5.4 Optimizer.

Experiment with the optimizer of your network.

Using Adam optimizer doesn't converge while using SGD optimizer converge well with fine-tuned hyperparameters.





Q 5.5 Depth of neural network.

Lastly, try adding additional layers to your network. How do 3, 4, and 5 layer networks perform? Is there a point where accuracy stops increasing?

3-layer network performs best with 0.9660 test accuracy, after 40 epochs, accuracy stops increasing.

Table 4: Hyperparameter tuning of width of hidden layer

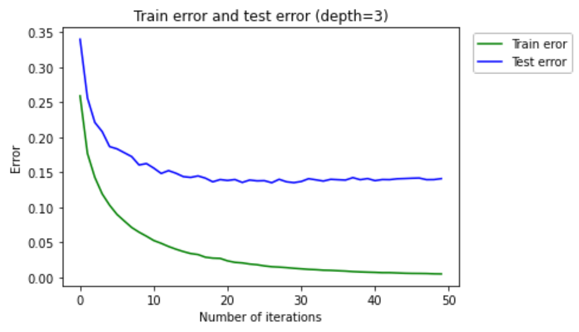
Fixed: Activation function = ReLU, hidden_size1=256, hidden_size2=128, hidden_size3=64, hidden_size4=32, learning_rate=0.001, optimizer=SGD		
num_layers	Test accuracy	Occurred after which epoch
2	0.9630	47
3	0.9660	40
4	0.9645	48
5	0.9645	38

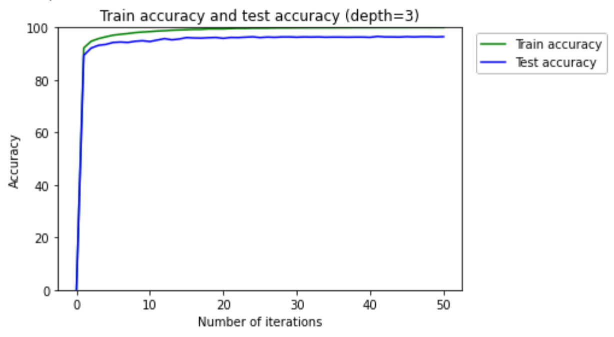
Q 5.6 Create a plot of the training and test error vs the number of iterations.

How many iterations are sufficient to reach good performance?

50 epochs.

Best performing 3-layer Neural network

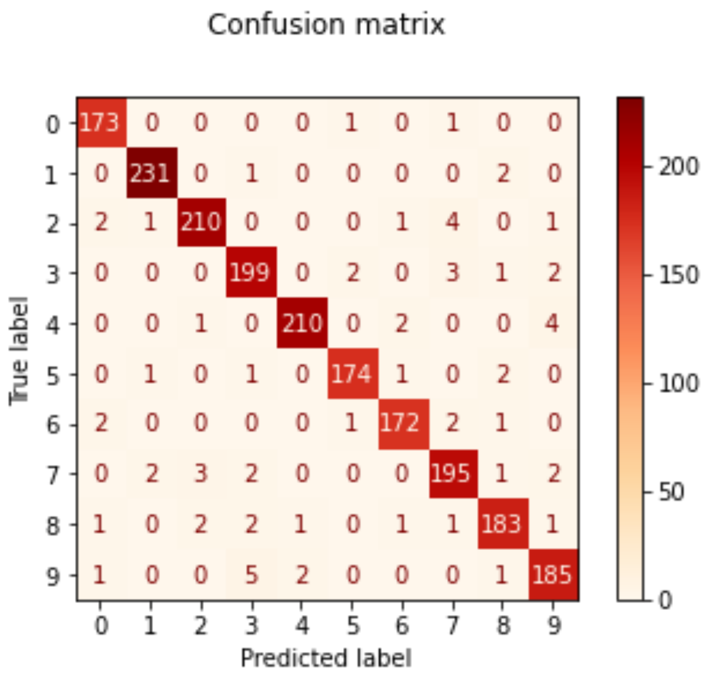




Q 5.6 Confusion matrix

Print a confusion matrix showing which digits were misclassified, and what they were misclassified as. What numbers are frequently confused with one another by your model?

Number 2 is frequently misclassified as number 7 (4/219)
Number 4 is frequently misclassified as number 9 (4/217)
Number 9 is frequently misclassified as number 3 (5/294)



	0	1	2	3	4	5	6	7	8	9
0	173	0	0	0	0	1	0	1	0	0
1	0	231	0	1	0	0	0	0	2	0
2	2	1	210	0	0	0	1	4	0	1
3	0	0	0	199	0	2	0	3	1	2
4	0	0	1	0	210	0	2	0	0	4
5	0	1	0	1	0	174	1	0	2	0
6	2	0	0	0	0	1	172	2	1	0
7	0	2	3	2	0	0	0	195	1	2
8	1	0	2	2	1	0	1	1	183	1
9	1	0	0	5	2	0	0	0	1	185

Q 5.7 Best performing model

What was the highest percentage of classification accuracy your FCN achieved? Briefly describe the architecture and training process that produced it.

The highest classification accuracy is 96.3%.

Architecture: 3 layers with ReLU activation function for the 1st layer and 2nd layer; hidden size of the 1st layer is 256, hidden size of the 2nd layer is 128.

Training process: learning rate .001, 50 epochs, cross entropy loss with L2-norm regularization (λ=0.01), optimizer is SGD.

References

[1] "1.4. Support Vector Machines." [Online]. Available: https://scikit-learn.org/stable/modules/svm.html

[2] "RBF SVM parameters." [Online]. Available: https://scikit-learn.org/stable/auto_examples/svm/plot_rbf_parameters.html

[3] "Difference between min_samples_split and min_samples_leaf in sklearn DecisionTreeClassifier." [Online]. Available: https://stackoverflow.com/questions/46480457/difference-between-min-samples-split-and-min-samples-leaf-in-sklearn-decisiontre

[4] "Two-bit adder." [Online]. Available: https://web2.qatar.cmu.edu/~srazak/courses/15348-f19/hw/hw3/hw3.pdf