314337 類神經網路 Assignment #2 MLP分類器 - 鳶尾花

班級：創新AI碩一

學號： 111C71008

姓名：何哲平

Objective 作業目標

A. 改以 **MLP Classifier** 完成Assignment #1實作
（特徵選取、切分資料、評估模型）
B. 說明 MLP網路架構 & 超參數設定
C. 展示 Loss Curve 誤差曲線
D. 視覺化 MLP 所有資料的散佈圖

Outline 大綱

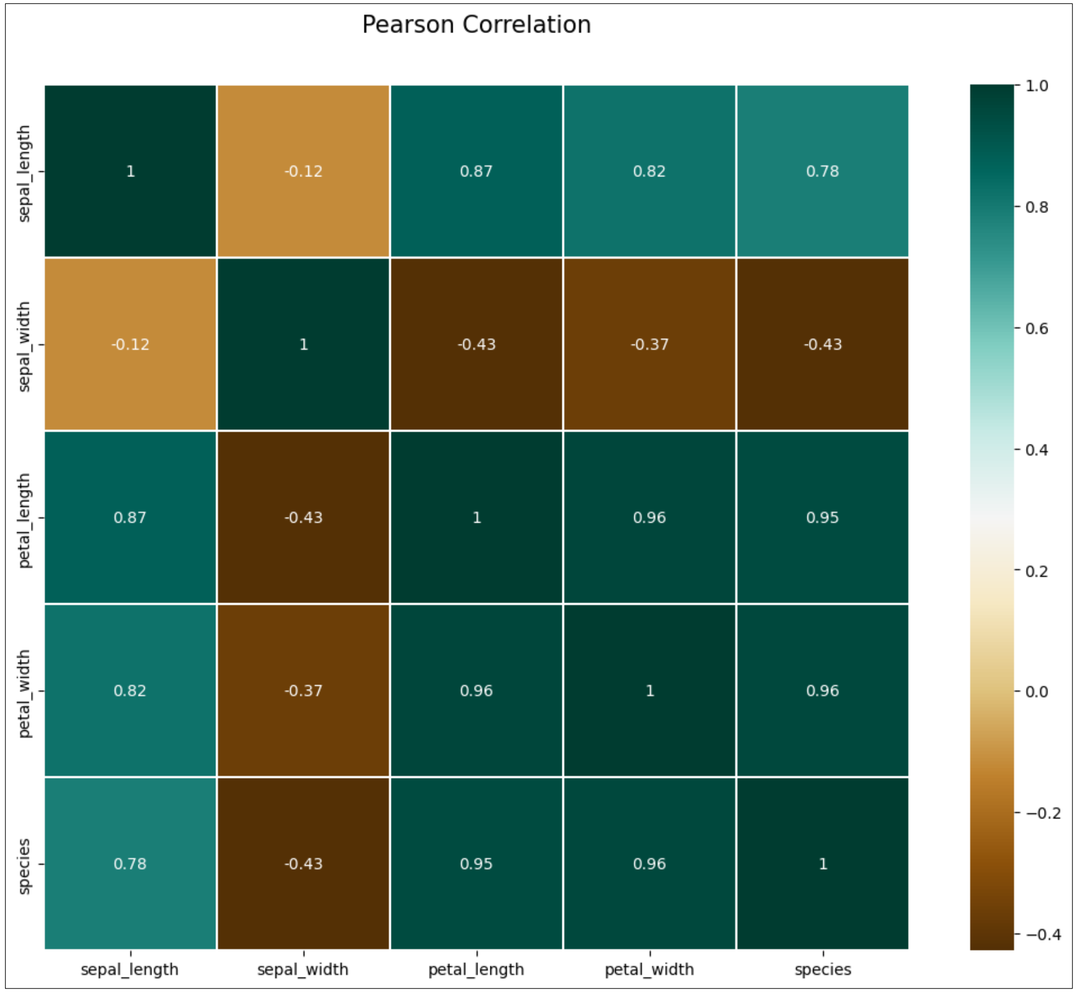
- 特徵選取
- 切分資料
- MLP分類器
- 評估模型

選取特徵

Notes：

使用兩個“越不具有代表性”的特徵變量。

HeatMap 觀察相關係數



Pearson Correlation	Species
Sepal Length 花萼長度	0.783
Sepal Width 花萼寬度	-0.427
Petal Length 花瓣長度	0.949
Petal Width 花瓣寬度	0.957
Species 花卉品種	1.000

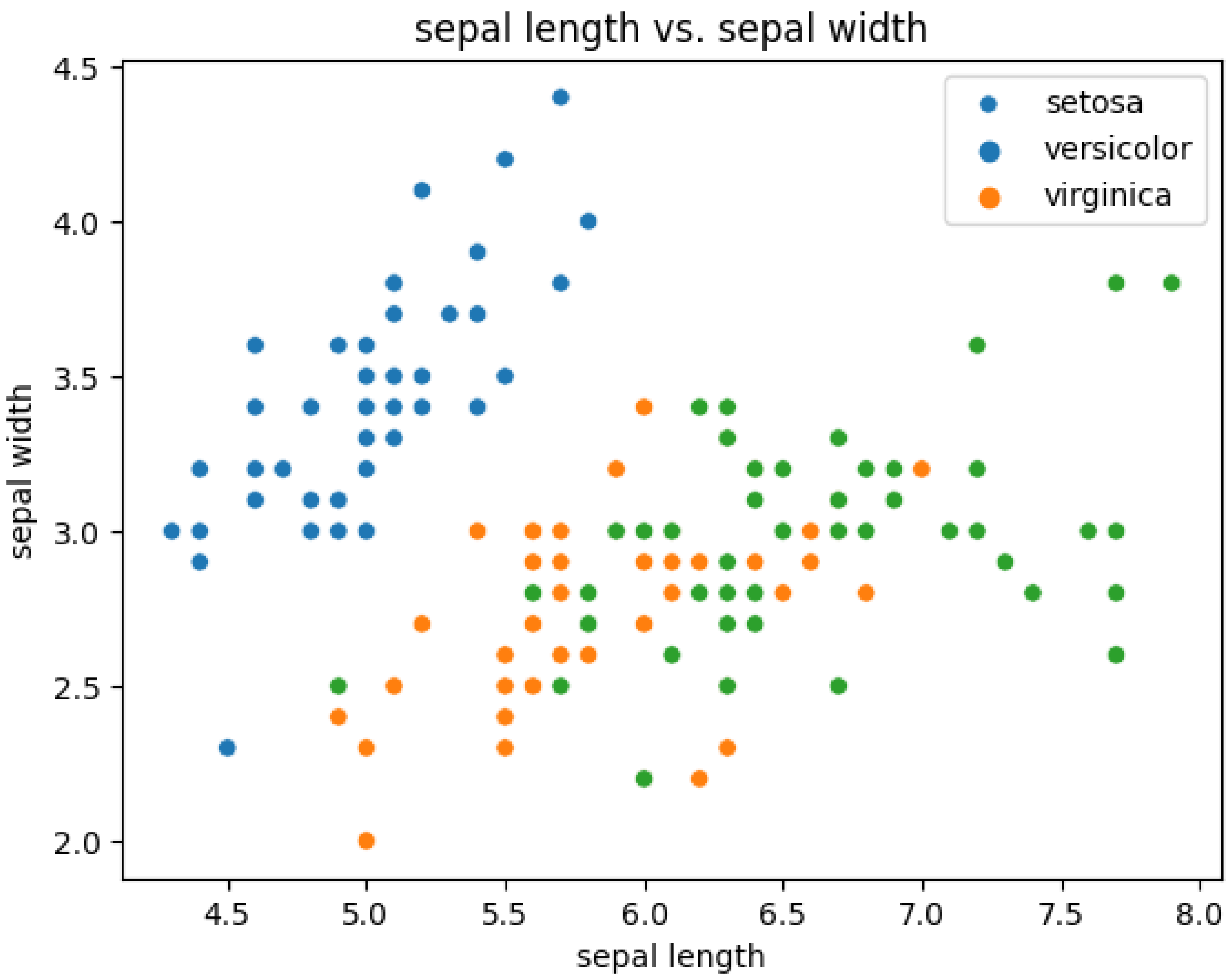
選擇Sepal Length & Sepal Width

Scatter Plot
散佈圖

- 選擇

➢ Sepal Length

➢ Sepal Width



切分資料

Train-Test-Split

Train-Test-Split

Train_Size：120筆

Test_Size：30筆

Train-Test-Split

Parameters

- Train_Size：80%
- Test_Size：20%
- shuffle：True
- stratify：抽樣比例依照原始'species'分布

```
# train_test_split
from sklearn.model_selection import train_test_split
train, test = train_test_split(data, train_size=0.8, test_size = 0.2, shuffle=True, stratify=data['target'])
```

Train-Test-Split 資料分佈

- Training 訓練資料

0	setosa	40
1	versicolor	40
2	virginica	40

- Testing 測試資料

0	setosa	10
1	versicolor	10
2	virginica	10

MLP Classifier

① 超參數設定

② Loss Curve 誤差曲線

③ 網路架構

MLP Classifier 超參數設定

HyperParameters

- hidden_layer_sizes：指定隱藏層層數+每層單元數
  設定 (10,) → 僅一層 10個神經元 的隱藏層。
- activation：隱藏層的激活函數
  設定 logistic sigmoid 函數 $f(x) = 1 / (1 + \exp(-x))$
- solver：優化器
  設定 lbfgs，適合小的數據集（少於幾千），收斂更快、效果更好。
  ※ 因為solver為lbfgs，不需要額外設定learning_rate及learning_rate_init。

```
▾ MLPClassifier
MLPClassifier(activation='logistic', hidden_layer_sizes=(10, 10),
              solver='lbfgs')
```

MLP Classifier 超參數設定

〔solver優化器〕根據Sklearn文件

- 訓練資料數量>1,000 → Solver設定為adam
- 訓練資料數量<1,000 → Solver設定為lbfgs

solver : *{'lbfgs', 'sgd', 'adam'}, default='adam'*

The solver for weight optimization.

- 'lbfgs' is an optimizer in the family of quasi-Newton methods.
- 'sgd' refers to stochastic gradient descent.
- 'adam' refers to a stochastic gradient-based optimizer proposed by Kingma, Diederik, and Jimmy Ba

Note: The default solver 'adam' works pretty well on relatively large datasets (with thousands of training samples or more) in terms of both training time and validation score. For small datasets, however, 'lbfgs' can converge faster and perform better.

MLP Classifier 超參數設定

- 使用solver=lbfgs，準確率大約為9成

```
clf = MLPClassifier(hidden_layer_sizes=(10,), activation='logistic', solver='lbfgs')
clf.fit(train_select_features, train['target'])
print(accuracy_score(test['target'], clf.predict(test_select_features)))
0.9
```

← Accuracy Score (Testing Data)

- 使用solver=adam，準確率也是大約為8~9成

```
clf = MLPClassifier(hidden_layer_sizes=(10,), activation='logistic', solver='adam', learning_rate='constant', learning_rate_init=0.2)
clf.fit(train_select_features, train['target'])
print(accuracy_score(test['target'], clf.predict(test_select_features)))
0.8
```

← Accuracy Score （ Testing Data ）

MLP Classifier 超參數設定

solver=lbfgs

使用兩層隱藏層，效果並未較好！？準確率未能達九成！？

- 使用**單層**，準確率大約為9成

```
clf = MLPClassifier(hidden_layer_sizes=(10,), activation='logistic', solver='lbfgs')
clf.fit(train_select_features, train['target'])
print(accuracy_score(test['target'], clf.predict(test_select_features)))
```

0.9 ← Accuracy Score（Testing Data）

- 使用**雙層**，準確率**下降**為73%

```
clf = MLPClassifier(hidden_layer_sizes=(10,10), activation='logistic', solver='lbfgs')
clf.fit(train_select_features, train['target'])
print(round(accuracy_score(test['target'], clf.predict(test_select_features)),2))
```

0.73 ← Accuracy Score（Testing Data）

MLP Classifier 超參數設定

solver=adam

使用兩層隱藏層，效果並未較好！？準確率未能達九成！？

- 使用**單層**，準確率也是大約為8~9成

```
clf = MLPClassifier(hidden_layer_sizes=(10,), activation='logistic', solver='adam', learning_rate='constant', learning_rate_init=0.2)
clf.fit(train_select_features, train['target'])
print(accuracy_score(test['target'], clf.predict(test_select_features)))
```

0.8 ← Accuracy Score（Testing Data）

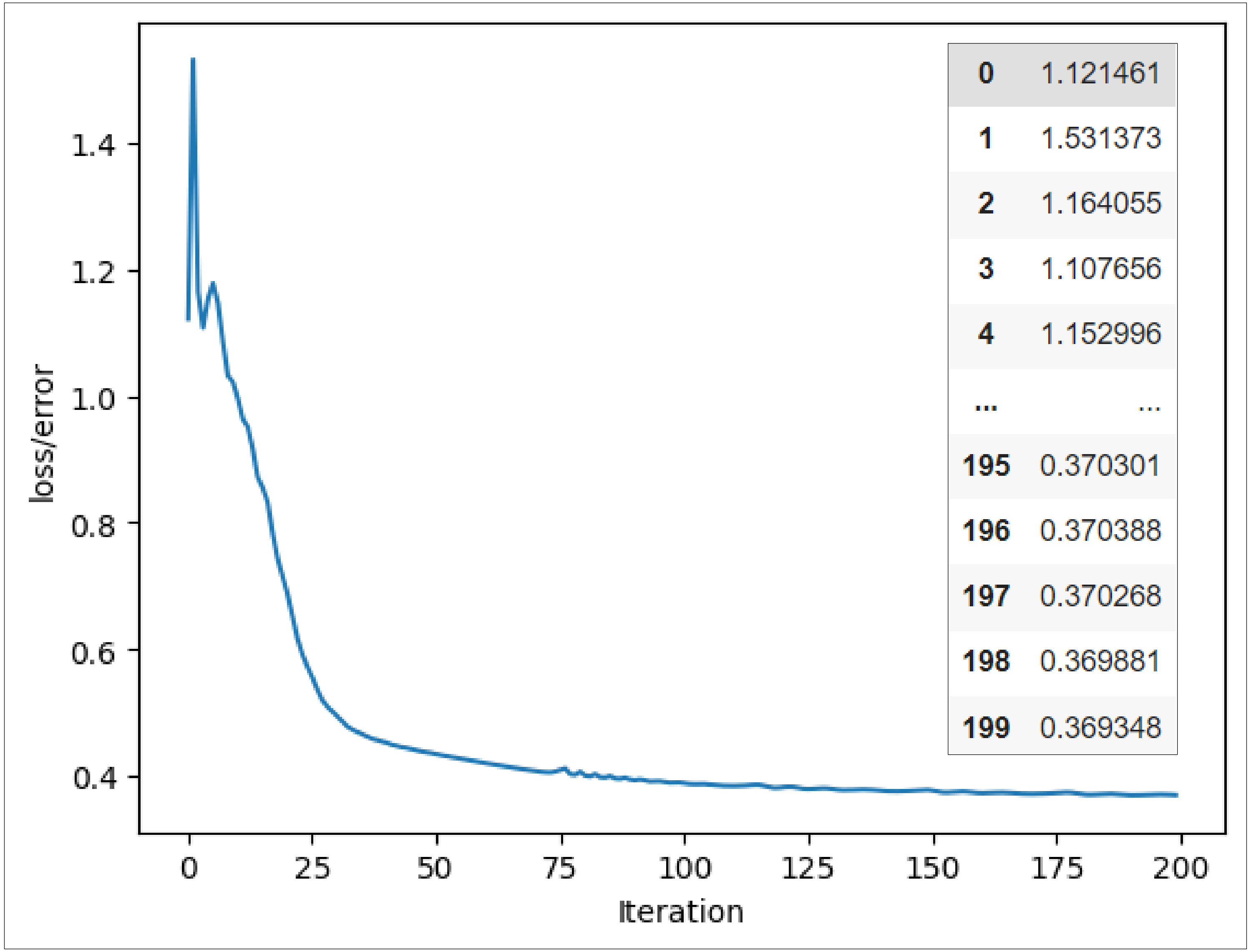
- 使用**雙層**，準確率**下降**為7成

```
clf = MLPClassifier(hidden_layer_sizes=(10,10), activation='logistic', solver='adam', learning_rate='constant', learning_rate_init=0.2)
clf.fit(train_select_features, train['target'])
print(round(accuracy_score(test['target'], clf.predict(test_select_features)),2))
```

0.7 ← Accuracy Score（Testing Data）

Loss Curve
誤差曲線

- solver = adam
- max_iter 最大迭代次數 = 200



0	1.121461
1	1.531373
2	1.164055
3	1.107656
4	1.152996
...	...
195	0.370301
196	0.370388
197	0.370268
198	0.369881
199	0.369348

```
clf = MLPClassifier(hidden_layer_sizes=(10,), activation='logistic', solver='adam', learning_rate='constant', learning_rate_init=0.2)
clf.fit(train_select_features, train['target'])
print(accuracy_score(test['target'], clf.predict(test_select_features)))

0.8
```

MLP Classifier 超參數設定

若使用solver=lbfgs

```
clf = MLPClassifier(hidden_layer_sizes=(10,), activation='logistic', solver='lbfgs')
clf.fit(train_select_features, train['target'])
print(accuracy_score(test['target'], clf.predict(test_select_features)))
```

```
0.9
```

無法產生Loss Curve，會發生錯誤。

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-25-40d20335df1f> in <cell line: 1>()
----> 1 pd.DataFrame(clf.loss_curve_)

AttributeError: 'MLPClassifier' object has no attribute 'loss_curve_'
```

※我想要最高準確率

所以仍使用solver=lbfgs

MLP Classifier 超參數設定

- Sklearn Document

Maximum number of iterations. The solver iterates until convergence (determined by 'tol') or this number of iterations. For stochastic solvers ('sgd', 'adam'), note that this determines the number of epochs (how many times each data point will be used), not the number of gradient steps.

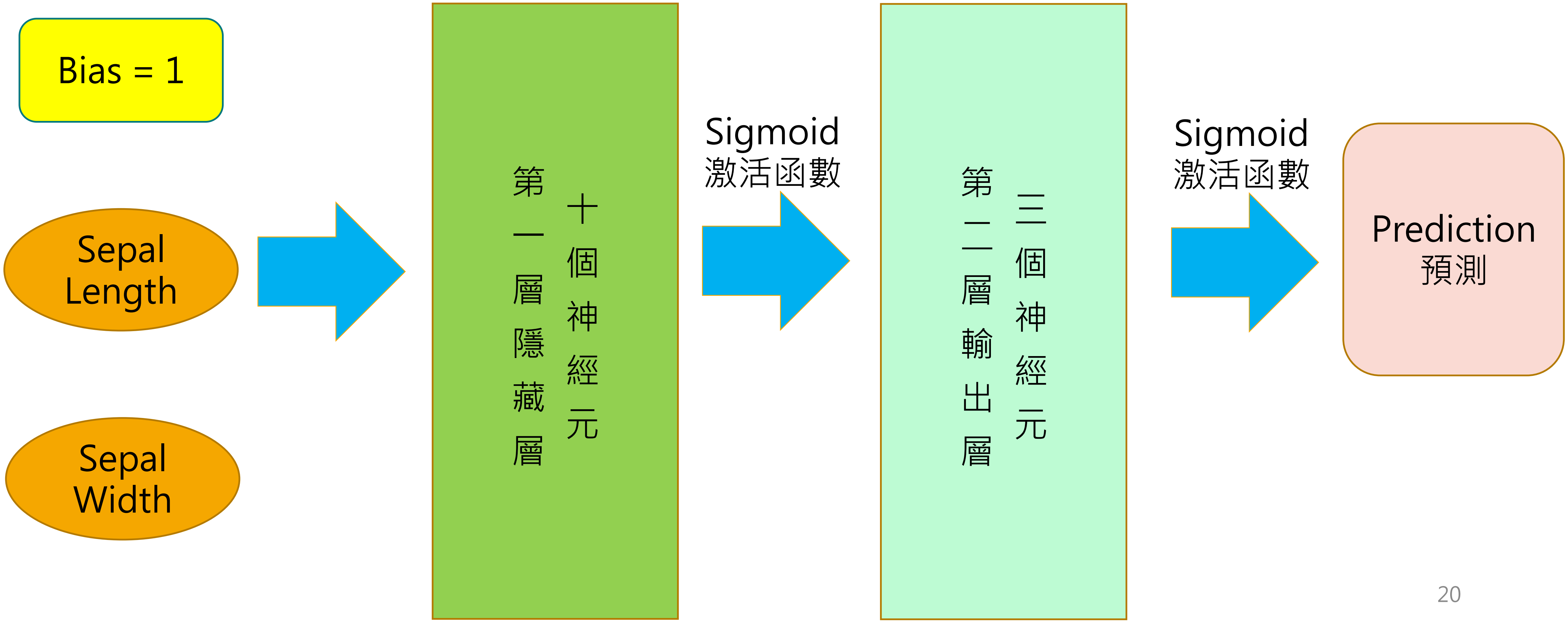
- Stack Overflow

error using loss_curve_ attribute of MLPClassifier in python

Yes sorry I forgot to do so. Eventually I found out that in the MLPClassifier function I had solver="lbfgs", but only the solver="sgd" can give me the losses (loss functionl mean) in every step, and then I can use the .loss_curve_ – George Andreadis Jan 15, 2019 at 13:21

※我想要最高準確率，所以仍使用solver=lbfgs

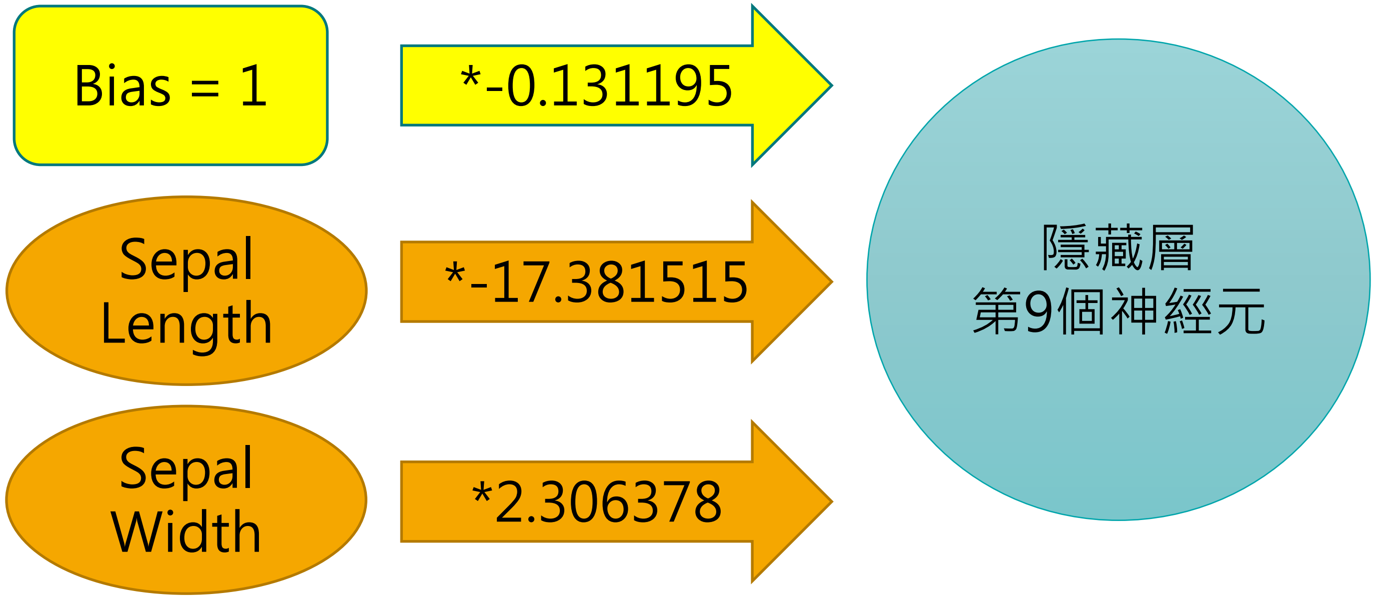
MLP Classifier 網路架構



MLP Classifier 1st Hidden-Layer

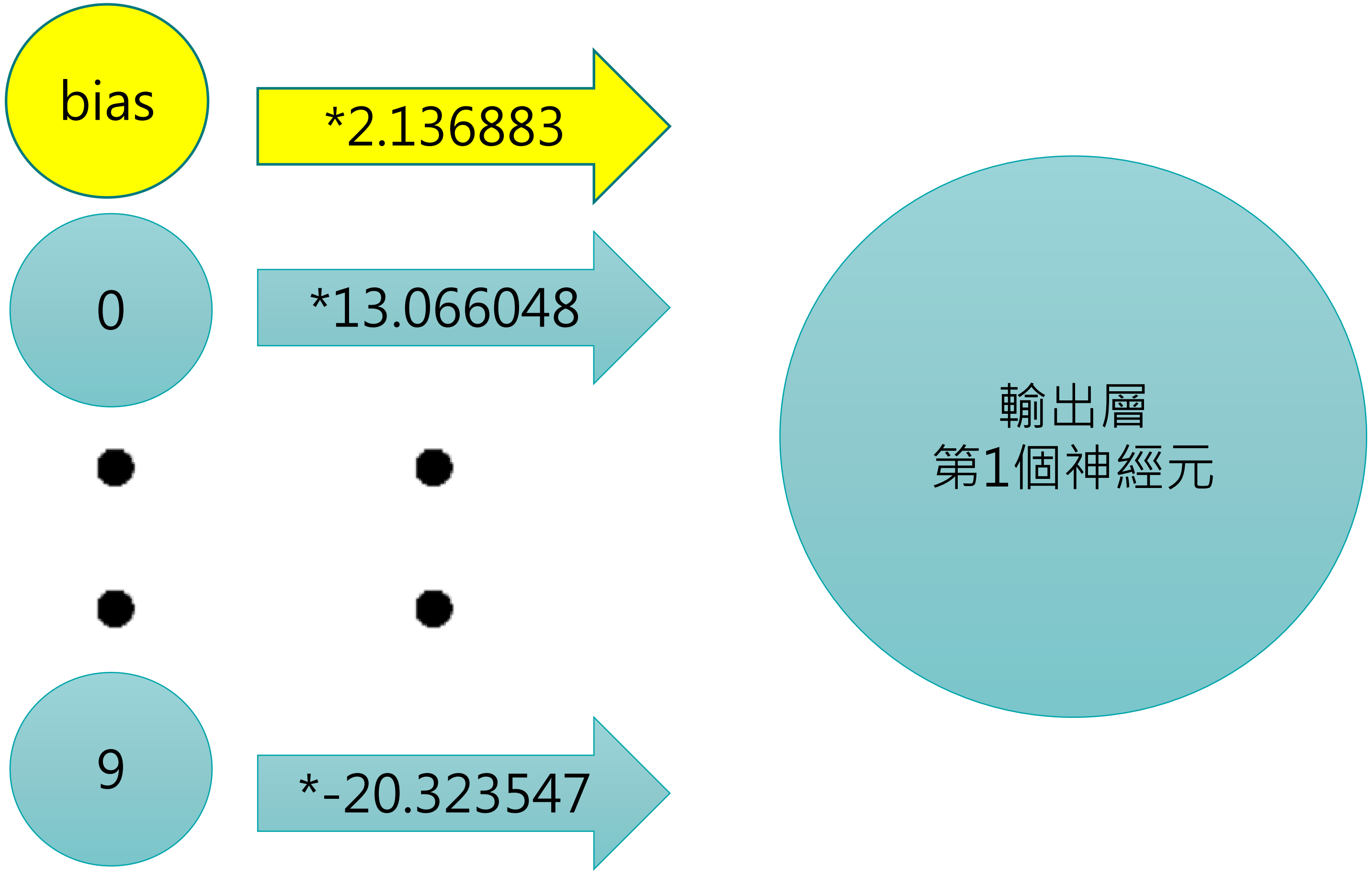
	0	1	2	3	4	5	6	7	8	9
W0	1.706706	-8.216318	4.084725	-0.860690	-6.234481	13.368878	22.474681	12.754654	6.094646	-0.131195
W1	7.275263	-2.924477	-14.569311	18.601892	0.780317	-5.650845	-8.111180	0.067817	14.655926	-17.381515
W2	-1.468660	-16.190923	16.815650	-12.209019	-0.066613	-9.588151	6.154574	-6.723332	23.105844	2.306378

MLP Classifier 1st Hidden-Layer



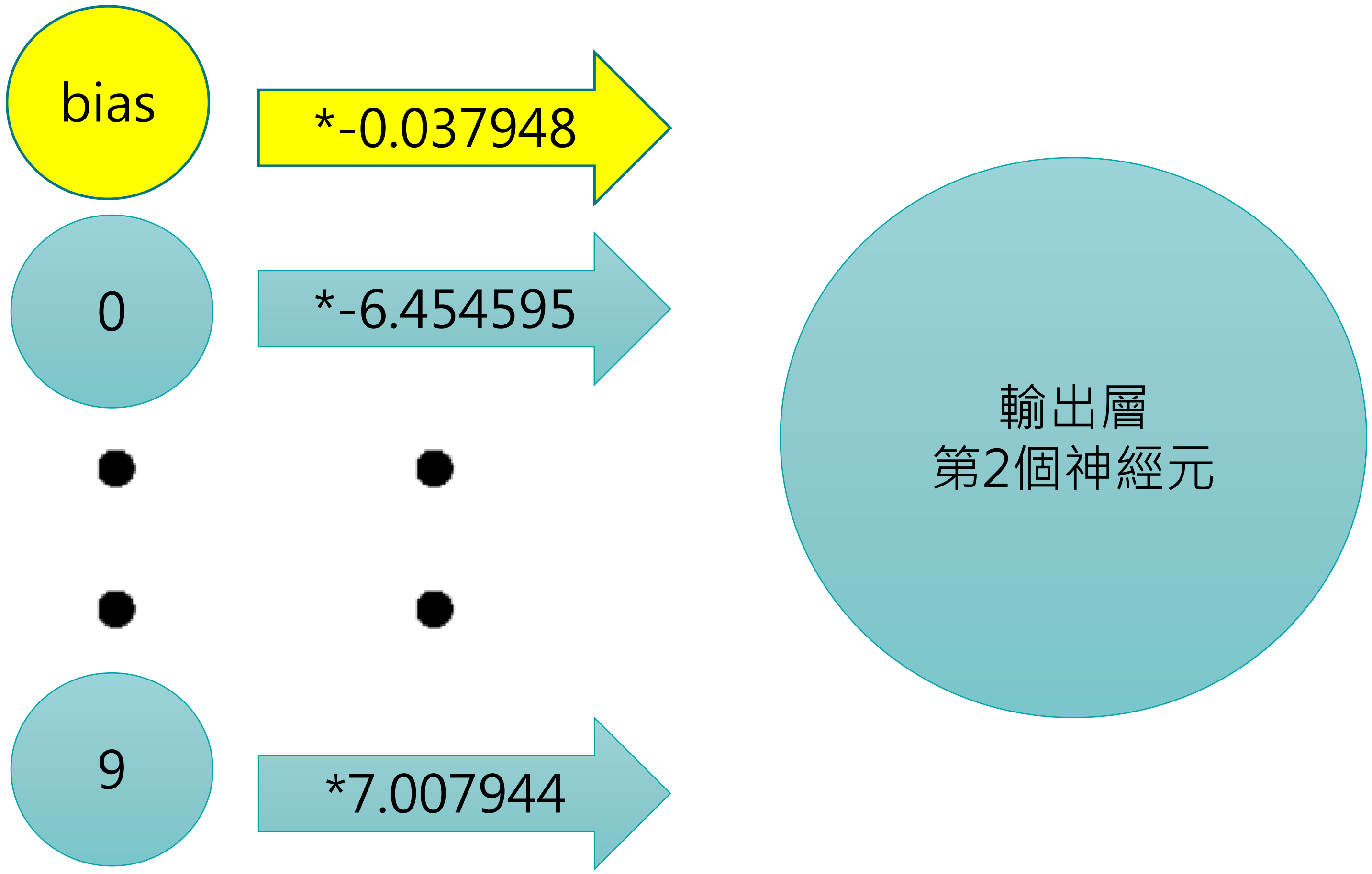
	0	1	2	3	4	5	6	7	8	9
W0	1.706706	-8.216318	4.084725	-0.860690	-6.234481	13.368878	22.474681	12.754654	6.094646	-0.131195
W1	7.275263	-2.924477	-14.569311	18.601892	0.780317	-5.650845	-8.111180	0.067817	14.655926	-17.381515
W2	-1.468660	-16.190923	16.815650	-12.209019	-0.066613	-9.588151	6.154574	-6.723332	23.105844	2.306378

MLP Classifier 2nd Output-Layer



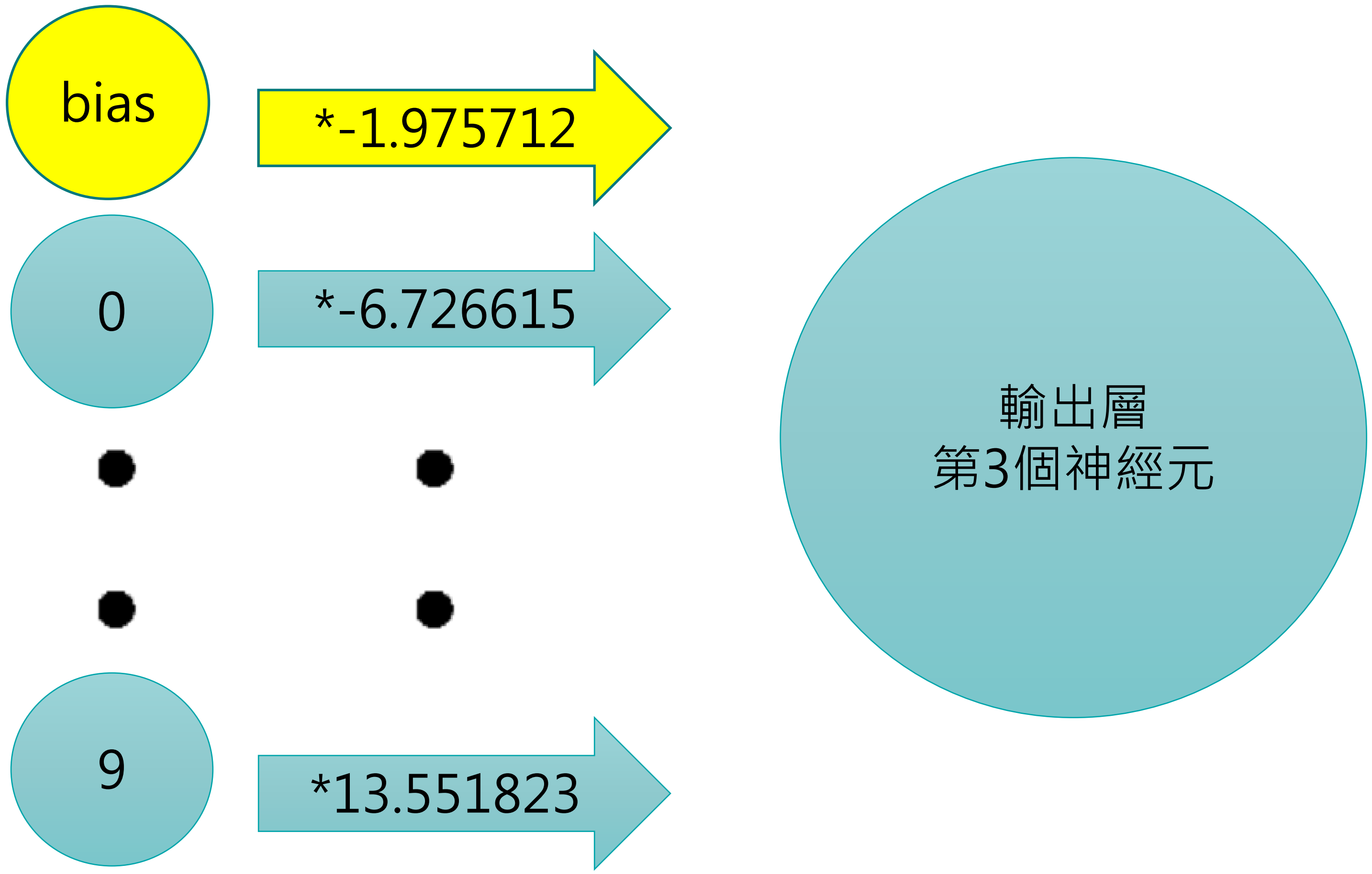
	0	1	2
bias	2.136883	-0.037948	-1.975712
W0	13.066048	-6.454595	-6.726615
W1	-9.112498	7.612147	1.135643
W2	13.686516	-3.683110	-9.991853
W3	-5.583614	4.103831	1.257583
W4	-3.767129	-6.074410	10.616023
W5	10.050021	0.328892	-11.162641
W6	29.569690	-19.925570	-9.697156
W7	12.668026	-3.333361	-9.459339
W8	-15.270382	6.627514	8.731228
W9	-20.323547	7.007944	13.551823

MLP Classifier 2nd Output-Layer



	0	1	2
bias	2.136883	-0.037948	-1.975712
W0	13.066048	-6.454595	-6.726615
W1	-9.112498	7.612147	1.135643
W2	13.686516	-3.683110	-9.991853
W3	-5.583614	4.103831	1.257583
W4	-3.767129	-6.074410	10.616023
W5	10.050021	0.328892	-11.162641
W6	29.569690	-19.925570	-9.697156
W7	12.668026	-3.333361	-9.459339
W8	-15.270382	6.627514	8.731228
W9	-20.323547	7.007944	13.551823

MLP Classifier 2nd Output-Layer



	0	1	2
bias	2.136883	-0.037948	-1.975712
W0	13.066048	-6.454595	-6.726615
W1	-9.112498	7.612147	1.135643
W2	13.686516	-3.683110	-9.991853
W3	-5.583614	4.103831	1.257583
W4	-3.767129	-6.074410	10.616023
W5	10.050021	0.328892	-11.162641
W6	29.569690	-19.925570	-9.697156
W7	12.668026	-3.333361	-9.459339
W8	-15.270382	6.627514	8.731228
W9	-20.323547	7.007944	13.551823

Visualization 視覺化

Notes：

- MLP輸出資料的散佈圖
- 共150筆資料

Visualization 視覺化

- 原始150筆資料 → *Weight權重 → 隱藏層 Hidden Neuron

	H0	H1	H2	H3	H4	H5	H6	H7	H8	H9
0	-69.358807	3.644399	11.817359	-128.762442	-27.007973	1.965684	-33.919081	117.479000	41.397778	-42.243343
1	-63.351166	2.149702	10.848811	-114.009513	-23.495252	2.238861	-33.147922	116.402926	39.502818	-41.829853
2	-63.888169	4.956931	12.085634	-118.321216	-24.820551	2.339823	-31.190896	108.439183	37.667592	-38.198726
3	-62.286772	5.131424	12.073940	-115.030030	-24.100910	2.439508	-30.551202	106.425217	36.732912	-37.302488
4	-69.627308	5.048014	12.435770	-130.918294	-27.670622	2.016164	-32.940568	113.497128	40.480165	-40.427779
...	...	...	...	...	...	...	...	...	...	...

150 rows × 10 columns

- 經過激活函數Sigmoid後，加上bias=1

	Bias	H0	H1	H2	H3	H4	H5	H6	H7	H8	H9
0	1	7.548367e-31	0.974529	0.999993	1.200002e-56	1.864604e-12	0.877147	1.858362e-15	1.0	1.0	4.507641e-19
1	1	3.068584e-28	0.895641	0.999981	3.064064e-50	6.253767e-11	0.903685	4.018282e-15	1.0	1.0	6.815940e-19
2	1	1.793581e-28	0.993015	0.999994	4.109120e-52	1.661772e-11	0.912122	2.844239e-14	1.0	1.0	2.573381e-17
3	1	8.896080e-28	0.994127	0.999994	1.104315e-50	3.412777e-11	0.919791	5.392394e-14	1.0	1.0	6.305732e-17
4	1	5.770912e-31	0.993619	0.999996	1.389656e-57	9.611727e-13	0.882484	4.944167e-15	1.0	1.0	2.769737e-18
...	...	...	...	...	...	...	...	...	...	...	...

150 rows × 11 columns

Visualization 視覺化

- 隱藏層 Hidden Neuron
  *Weight權重
  = Output Neuron

	predict0	predict1	predict2
0	13.633426	-9.522105	-5.214052
1	11.538583	-8.038579	-4.570807
2	14.570705	-9.805370	-5.863842
3	14.687386	-9.813711	-5.970346
4	14.267793	-9.862726	-5.512183
...	...	...	...

150 rows × 3 columns

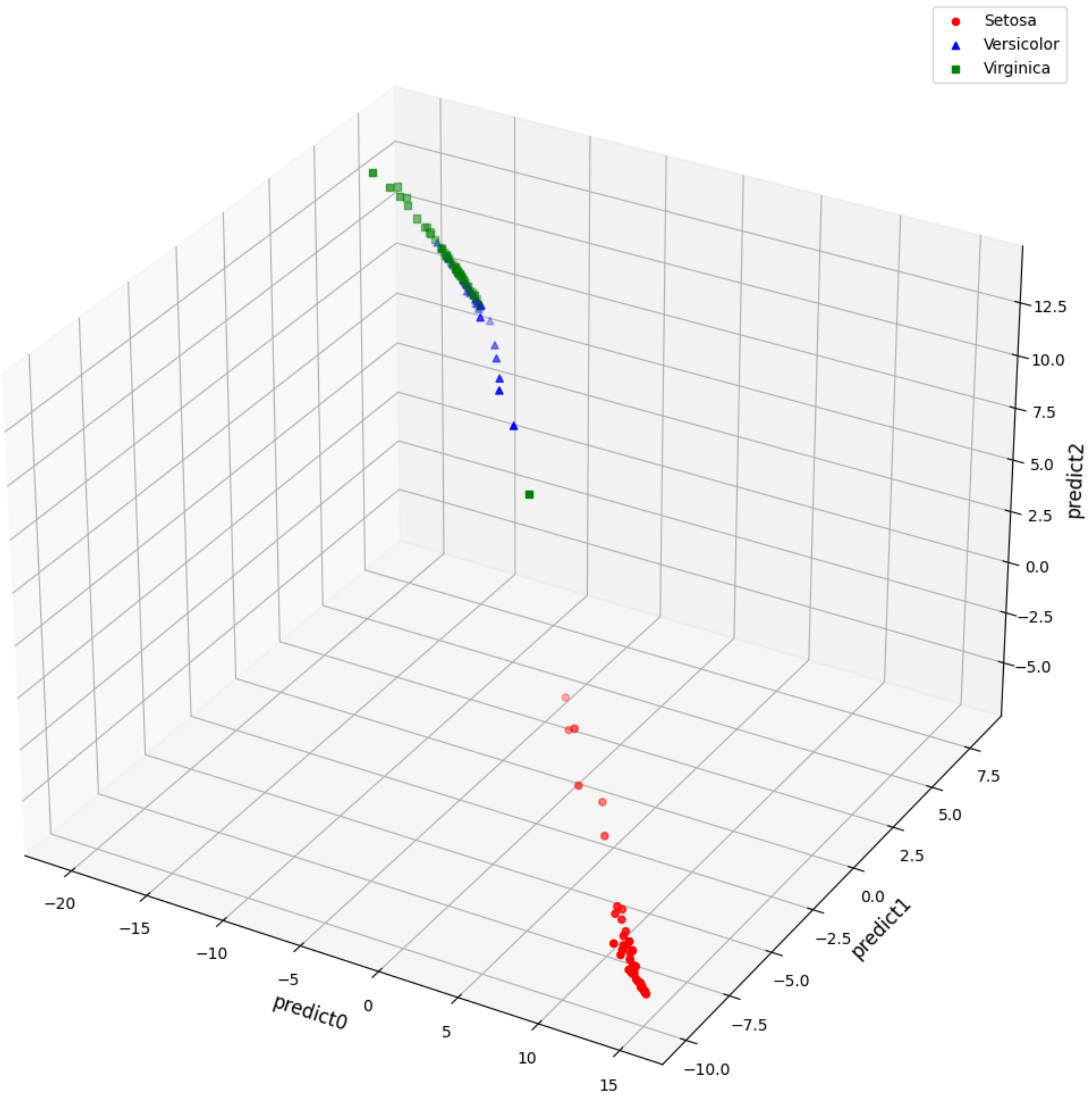
- 經過激活函數Sigmoid後

	predict0	predict1	predict2
0	9.999988e-01	0.000073	0.005410
1	9.999903e-01	0.000323	0.010244
2	9.999995e-01	0.000055	0.002832
3	9.999996e-01	0.000055	0.002547
4	9.999994e-01	0.000052	0.004021
...	...	...	...

150 rows × 3 columns

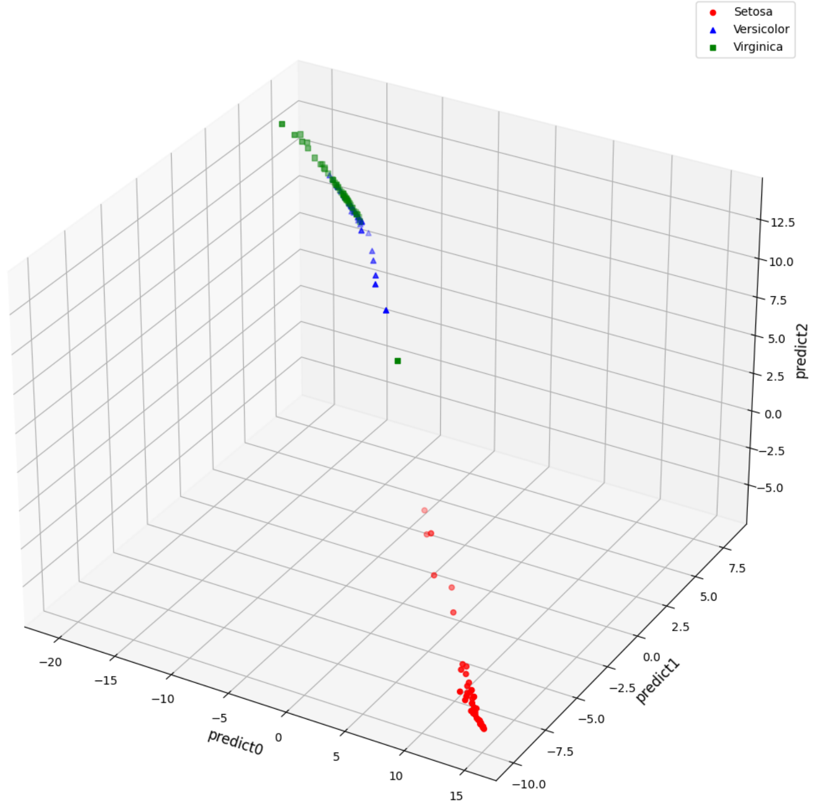
Visualization
視覺化

- 尚未激活函數Sigmoid前



Visualization 視覺化

尚未激活函數Sigmoid前



Setosa

	0	1	2	3	4	5	6	7	8	9
predict0	13.633426	11.538583	14.570705	14.687386	14.267793	13.272581	14.775708	13.866044	14.886054	12.876350
predict1	-9.522105	-8.038579	-9.805370	-9.813711	-9.862726	-9.623161	-9.912763	-9.578893	-9.824064	-8.865893
predict2	-5.214052	-4.570807	-5.863842	-5.970346	-5.512183	-4.763998	-5.962742	-5.388176	-6.155349	-5.095808

3 rows × 50 columns

Versicolor

	50	51	52	53	54	55	56	57	58	59
predict0	-18.303318	-17.245179	-18.061249	-15.777338	-17.221798	-16.022383	-17.120174	-10.037105	-17.430294	-12.251148
predict1	7.894627	8.073389	7.929100	8.266134	8.058785	8.155423	8.079445	5.119206	8.029075	6.247364
predict2	9.545925	8.332748	9.275783	6.709843	8.327580	7.054798	8.203908	4.072732	8.560279	5.170955

3 rows × 50 columns

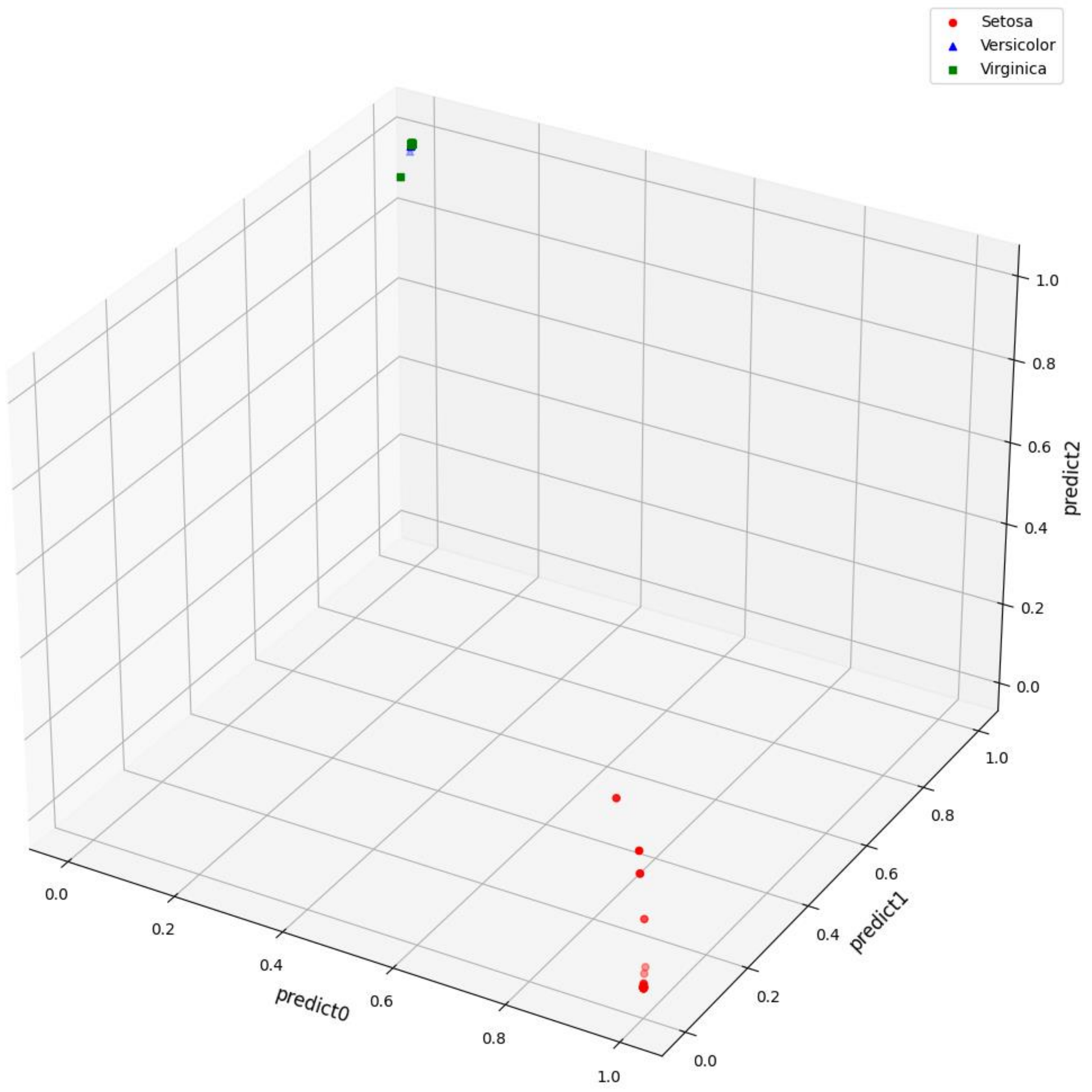
Virginica

	100	101	102	103	104	105	106	107	108	109
predict0	-17.120174	-16.200616	-18.413241	-16.957232	-17.312117	-19.640252	-6.812746	-18.841450	-17.465860	-18.922940
predict1	8.079445	8.204857	7.822683	8.112651	8.058647	7.336354	3.152159	7.625312	7.919630	7.822862
predict2	8.203908	7.182950	9.731794	8.013904	8.414010	11.450894	2.781647	10.362775	8.717175	10.218896

3 rows × 50 columns

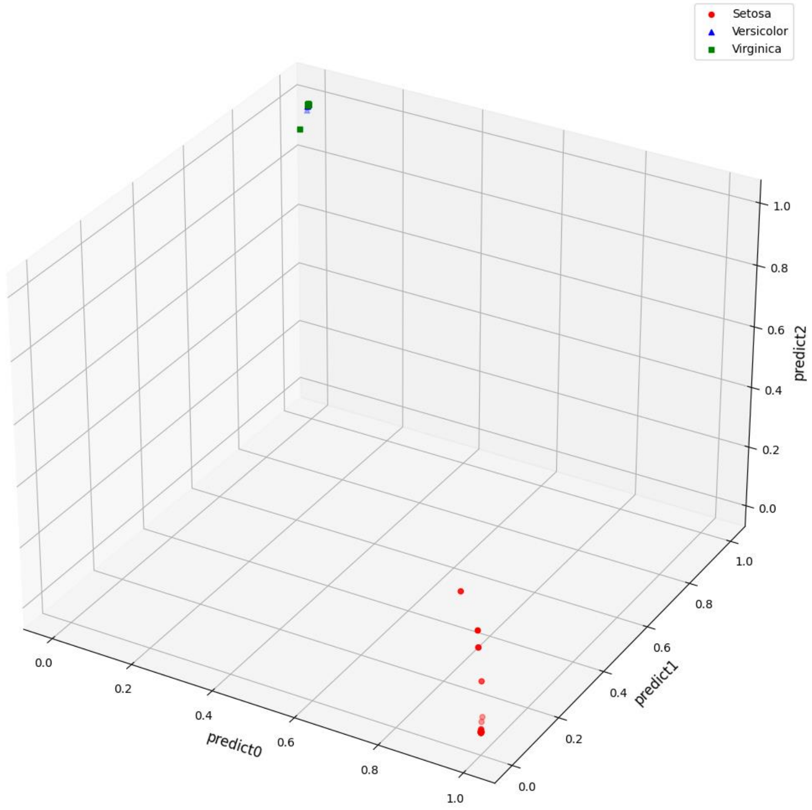
Visualization
視覺化

- 經過激活函數Sigmoid後



Visualization 視覺化

經過激活函數Sigmoid後



	0	1	2	3	4	5	6	7	8	9
predict0	0.999999	0.999990	1.000000	1.000000	0.999999	0.999998	1.000000	0.999999	1.000000	0.999997
predict1	0.000073	0.000323	0.000055	0.000055	0.000052	0.000066	0.000050	0.000069	0.000054	0.000141
predict2	0.005410	0.010244	0.002832	0.002547	0.004021	0.008459	0.002566	0.004550	0.002118	0.006085

3 rows × 50 columns

	50	51	52	53	54	55	56	57	58	59
predict0	1.124528e-08	3.239768e-08	1.432515e-08	1.406012e-07	3.316408e-08	1.100443e-07	3.671156e-08	0.000044	2.692276e-08	0.000005
predict1	9.996274e-01	9.996884e-01	9.996400e-01	9.997430e-01	9.996838e-01	9.997129e-01	9.996903e-01	0.994055	9.996743e-01	0.998068
predict2	9.999285e-01	9.997595e-01	9.999063e-01	9.987826e-01	9.997583e-01	9.991375e-01	9.997265e-01	0.983254	9.998085e-01	0.994353

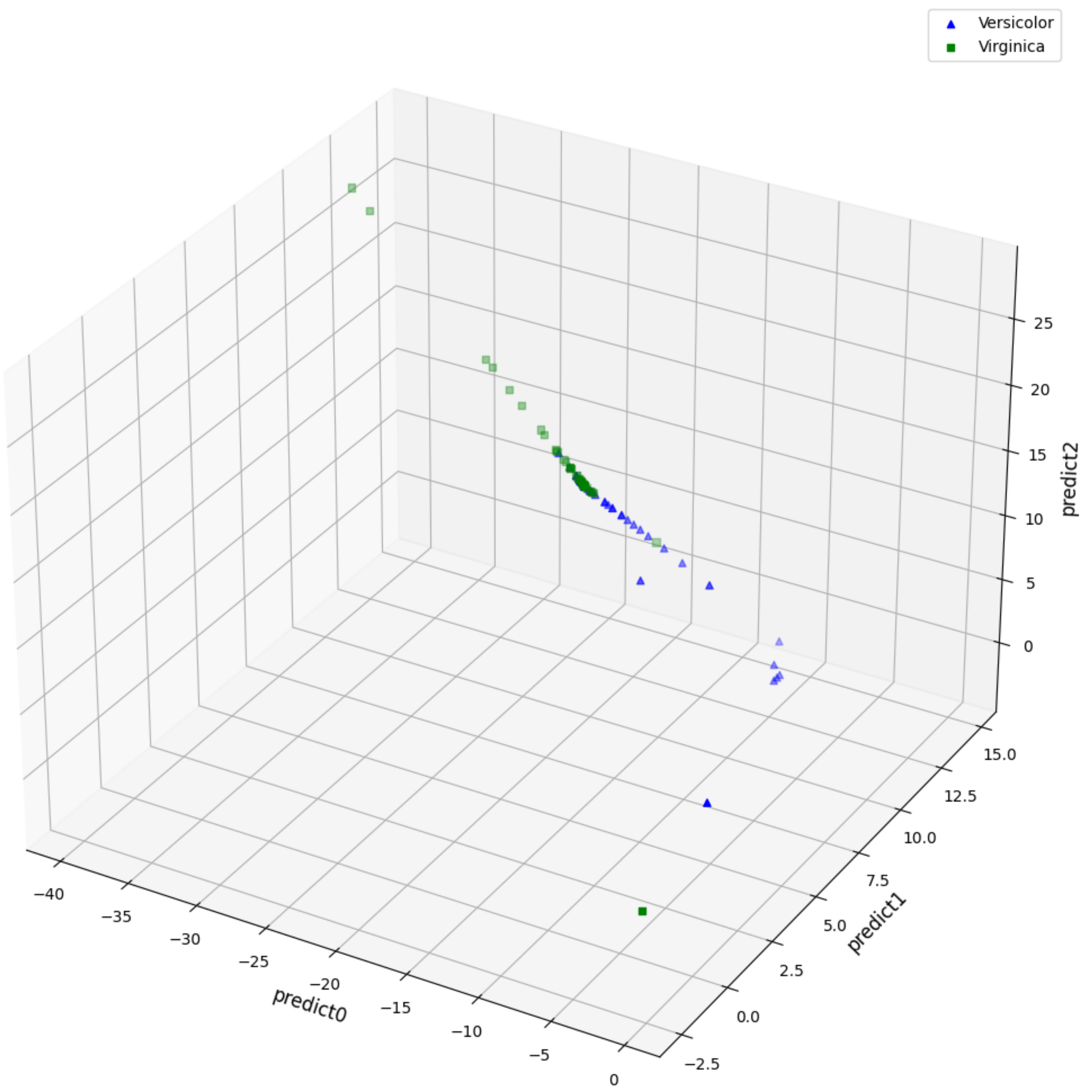
3 rows × 50 columns

	100	101	102	103	104	105	106	107	108	109	...
predict0	3.671156e-08	9.207923e-08	1.007467e-08	4.320835e-08	3.030005e-08	2.953569e-09	0.001098	6.565417e-09	2.598204e-08	6.051619e-09	...
predict1	9.996903e-01	9.997268e-01	9.995996e-01	9.997004e-01	9.996837e-01	9.993490e-01	0.958994	9.995123e-01	9.996366e-01	9.995997e-01	...
predict2	9.997265e-01	9.992412e-01	9.999406e-01	9.996693e-01	9.997783e-01	9.999894e-01	0.941676	9.999684e-01	9.998363e-01	9.999635e-01	...

3 rows × 50 columns

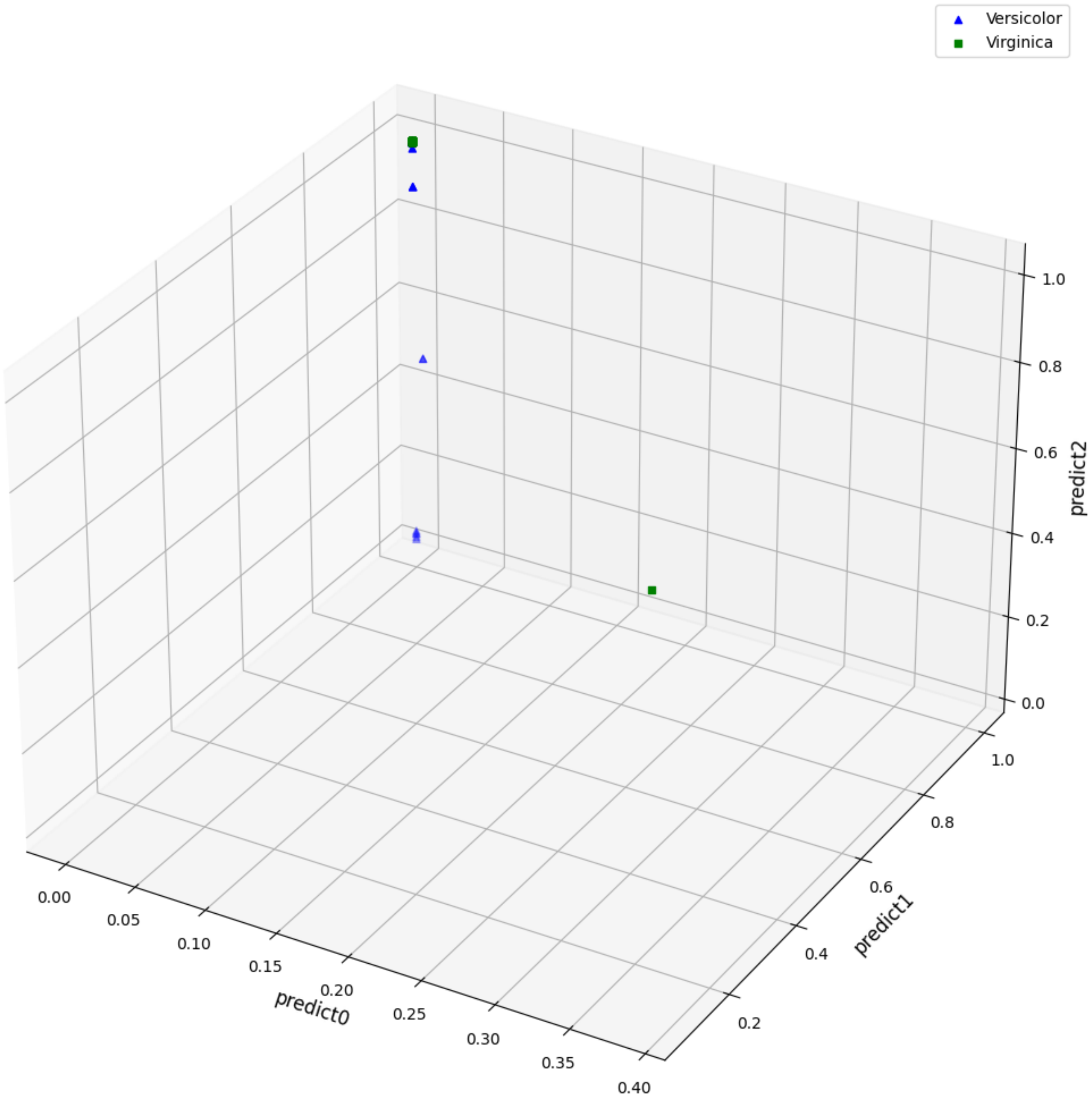
〔補充〕Visualization 視覺化

- 檢視Versicolor及Virginica
- 經過激活函數Sigmoid前



〔補充〕Visualization 視覺化

- 檢視Versicolor及Virginica
- 經過激活函數Sigmoid後



評估模型

① Confusion Matrix

② Accuracy / Precision / Recall / F1 Score

評估指標

- **正確率 Accuracy**：有多少比例的樣本預測對了
- **精確率 Precision**：預測為正的樣本中有多少預測對了
- **召回率 Recall**：真實正的樣本有多少被預測對了
- *Fβ_Score：綜合考量 Precision與Recall*
- **F1-Score**：Precision與Recall同等重要

$$\text{F1 Score} = \frac{2}{\frac{1}{\text{Precision}} + \frac{1}{\text{Recall}}} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Predicted

True labels	A	B	C	
A	2	2	0	4
B	1	2	0	3
C	0	0	3	3
	3	4	3	Total

Prediction 預測 Testing測試資料

- 答對27個；答錯3個 （Total：30筆）
- Accuracy：[實際值=預測值]數量 / 測試資料數量 = 27/30 = 90%
- 判斷錯誤的資料：

	sepal length (cm)	sepal width (cm)	預測值	實際值
0	6.0	2.2	1	2
1	6.7	3.1	2	1
2	6.7	3.1	2	1

預測值

實際值

	0 setosa	1 versicolor	2 virginica
0 setosa	10	0	0
1 versicolor	0	8	2
2 virginica	0	1	9

Prediction 預測 Testing測試資料

Setosa

- 類別：0

➢Precision = 10/10 = 100%

➢Recall = 10/10 = 100%

➢F1-Score = 100%

預測值

實際值

	0 setosa	1 versicolor	2 virginica	*Sum*
0 setosa	10	0	0	*10*
1 versicolor	0	8	2	*10*
2 virginica	0	1	9	*10*
Sum	*10*	*9*	*11*	*30*

Prediction 預測 Testing測試資料

Versicolor

- 類別：1

➢Precision = 8/9 = 88.9%

➢Recall = 8/10 = 80%

➢F1-Score = 84.2%

預測值

實際值

	0 setosa	**1 versicolor**	2 virginica	*Sum*
0 setosa	10	0	0	*10*
1 versicolor	0	8	2	*10*
2 virginica	0	1	9	*10*
Sum	*10*	*9*	*11*	*30*

Prediction 預測 Testing測試資料

Virginica

- 類別：2

➢Precision = 9/11 = 81.8%

➢Recall = 9/10 = 90%

➢F1-Score = 85.7%

預測值

實際值

	0 setosa	1 versicolor	2 virginica	*Sum*
0 setosa	10	0	0	*10*
1 versicolor	0	8	2	*10*
2 virginica	0	1	9	*10*
Sum	*10*	*9*	*11*	*30*

Prediction 預測 所有資料

- 答對122個；答錯28個（Total：150筆）
- Accuracy：[實際值=預測值]數量 / 測試資料數量 = 122/150 = 81.3%

預 測 值

實際值	0 setosa	1 versicolor	2 virginica
0 setosa	50	0	0
1 versicolor	0	38	12
2 virginica	0	16	34

Prediction 預測 <u>所有資料</u>

Setosa

- 類別：0

➢Precision = 50/50 = 100%

➢Recall = 50/50 = 100%

➢F1-Score = 100%

預測值

實際值

	0 setosa	1 versicolor	2 virginica	*Sum*
0 setosa	50	0	0	*50*
1 versicolor	0	38	12	*50*
2 virginica	0	16	34	*50*
Sum	*50*	*54*	*46*	*150*

Prediction 預測 所有資料

Versicolor

- 類別：1

➢Precision = 38/54 = 70.3%

➢Recall = 38/50 = 76%

➢F1-Score = 73.1%

預測值

實際值

	0 setosa	1 versicolor	2 virginica	*Sum*
0 setosa	50	0	0	*50*
1 versicolor	0	38	12	*50*
2 virginica	0	16	34	*50*
Sum	*50*	*54*	*46*	*150*

Prediction 預測 所有資料

Virginica

- 類別：2

➢Precision = 34/46 = 73.9%

➢Recall = 34/50 = 68%

➢F1-Score = 70.8%

預測值

實際值		0 setosa	1 versicolor	2 virginica	*Sum*
	0 setosa	50	0	0	*50*
	1 versicolor	0	38	12	*50*
	2 virginica	0	16	34	*50*
	Sum	*50*	*54*	*46*	*150*

Conclusion 結論

- 隨意刪除資料欄位

→影響模型的準確度（僅使用Sepal，準確度些許下降）。

- 簡單的問題

→ 使用複雜的模型，效果並沒有較好。

※〔思考〕參數應如何設定才會得到最高準確率？

（試圖利用 GridSearchCV貪婪演算法 或 迭代方式 尋找）

※〔思考〕應如何訓練模型才能最準確預測所有150筆資料？

（目前Testing的Accuracy為90%，但整體的Accuracy卻為81.34% ）

簡報完畢

人若賺得全世界，卻賠上自己的魂生命，有什麼益處？

馬太福音 第十六章 26節