

10주차(2/3)

로지스틱 회귀

파이썬으로 배우는 기계학습

한동대학교
김영섭 교수

로지스틱 회귀

- 학습 목표
 - 회귀 분석을 익힌다.
 - 오차 함수의 차이를 인식한다.
 - 다층 신경망에 두 개의 활성화 함수를 사용한다.
 - 로지스틱 함수와 로지스틱 회귀 알고리즘을 배운다.
- 학습 내용
 - 회귀와 회귀 분석 정의
 - 오차 함수의 차이
 - 로지스틱 회귀
 - 로짓 함수

1. 회귀 분석: 예

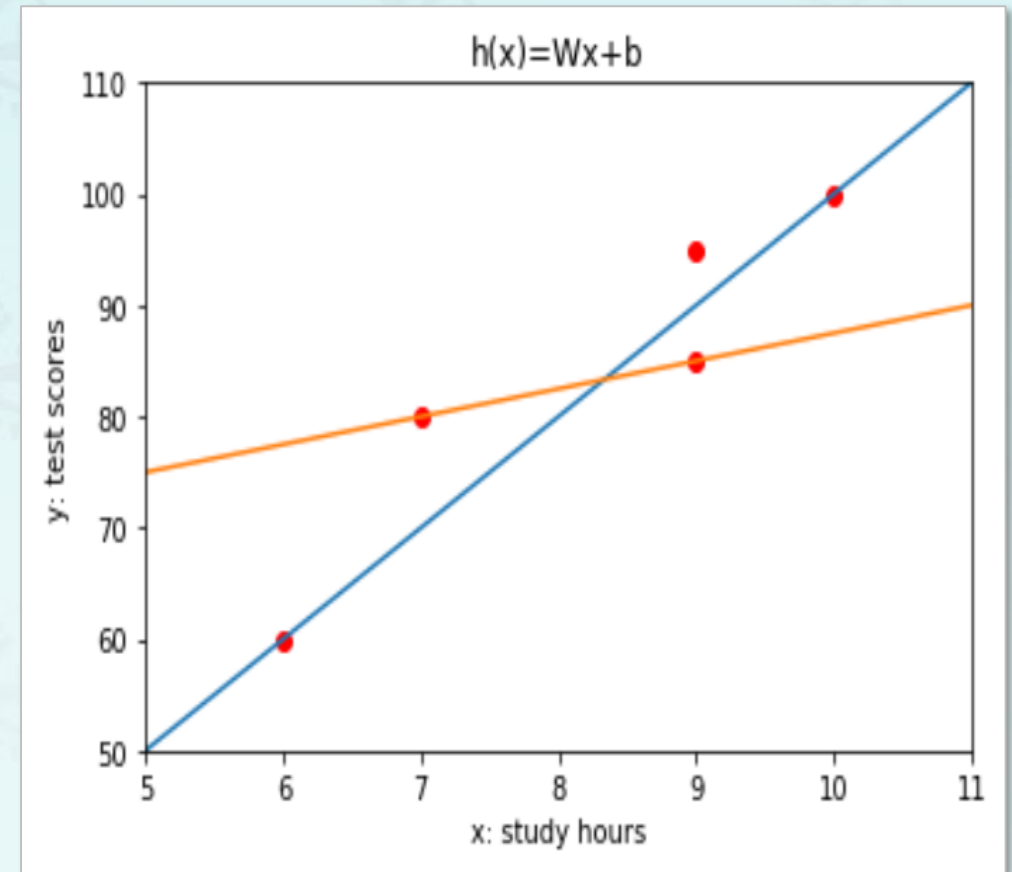
- 회귀(Regression)
 - 변수 간 관계 분석
- 회귀 분석(Regression Analysis)
 - 회귀에서 한 분석을 바탕으로 함수를 정의해서 값을 예측하는 기법
- 회귀나 회귀분석 혼용해서 사용

공부 시간(x)	시험 성적(y)
10	100
9	85
7	80
6	60
9	95
...	...

1. 회귀 분석: 단순 회귀 분석

- x, y 두 변수만 취급

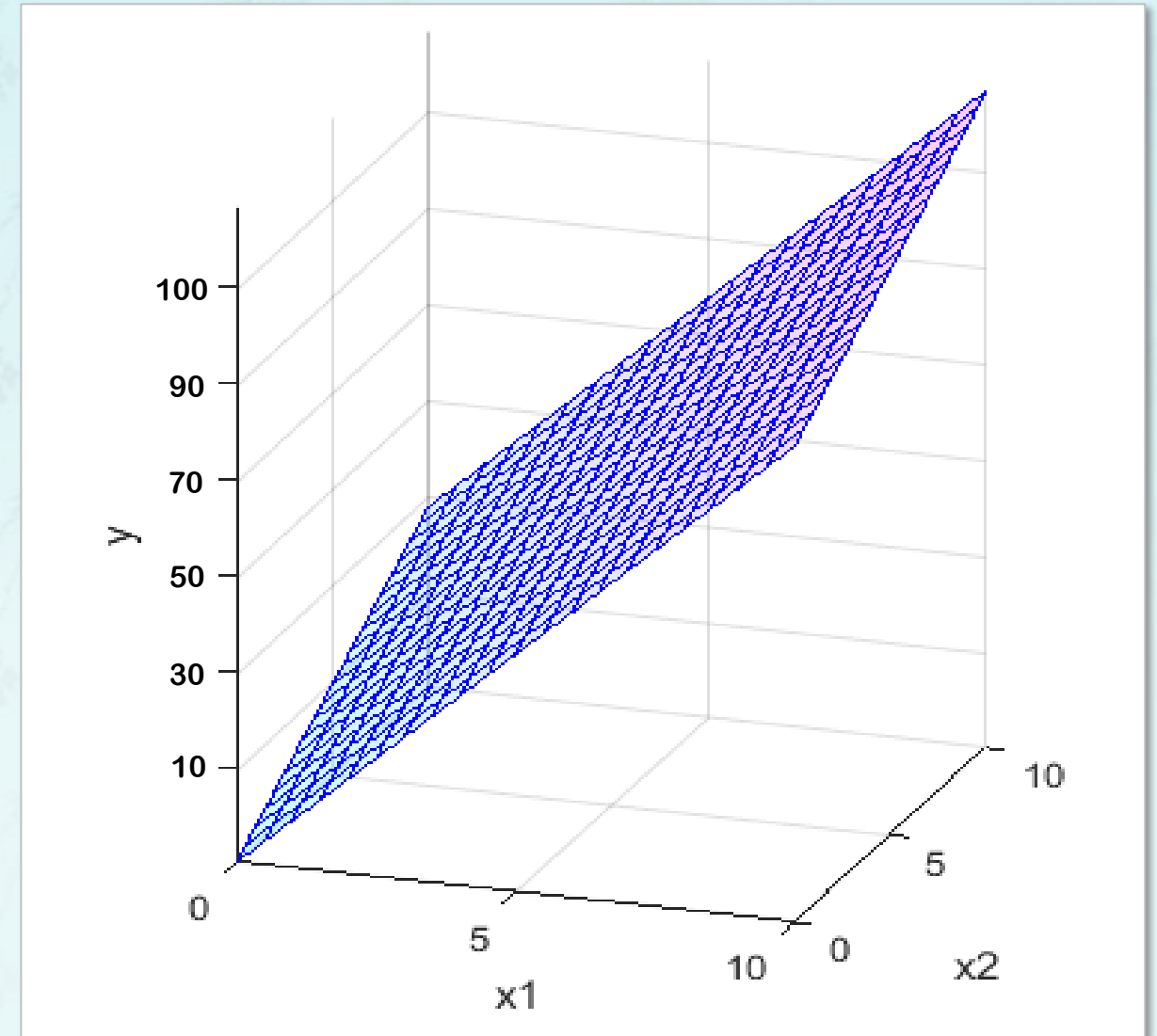
공부 시간(x)	시험 성적(y)
10	100
9	85
7	80
6	60
9	95
...	...



1. 회귀 분석: 다중 회귀 분석

- x_1, x_2, \dots, y 등 여러 변수 취급

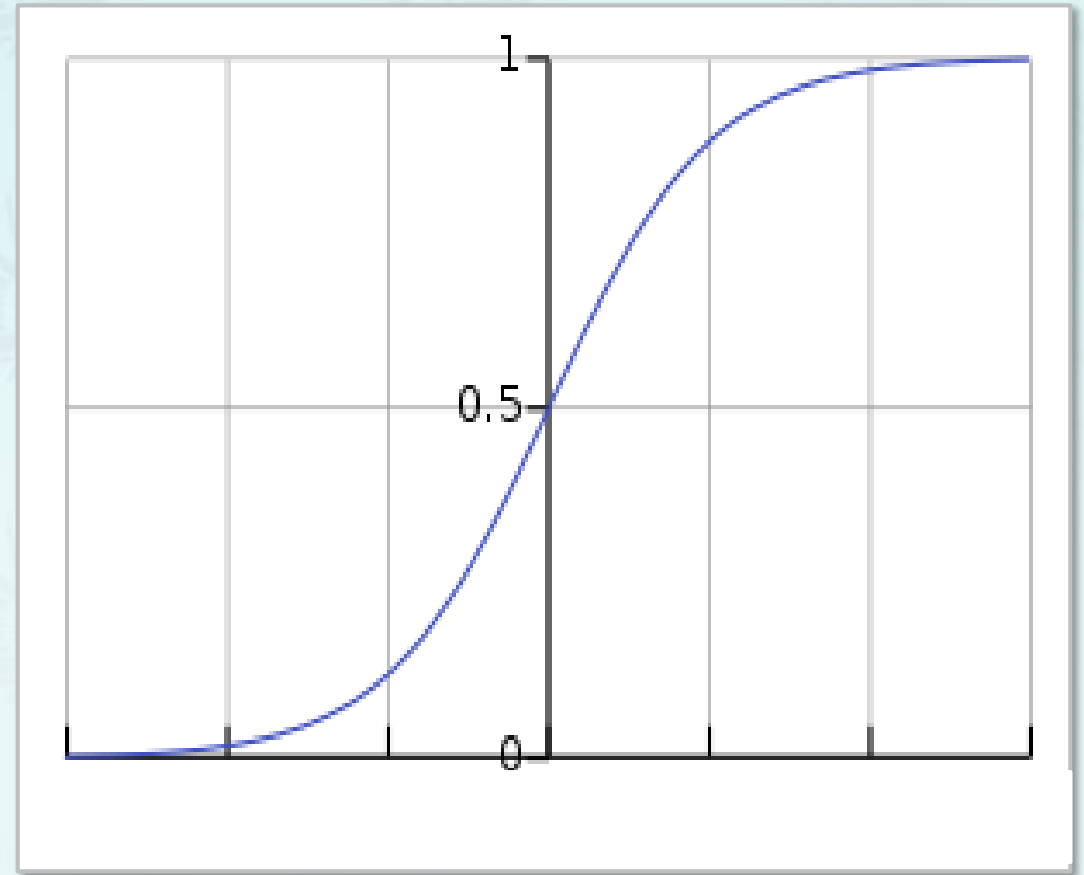
공부 시간 (x_1)	집중 시간 (x_2)	시험 성적 (y)
10	10	100
9	8	85
7	6	80
6	4	60
9	9	95
...



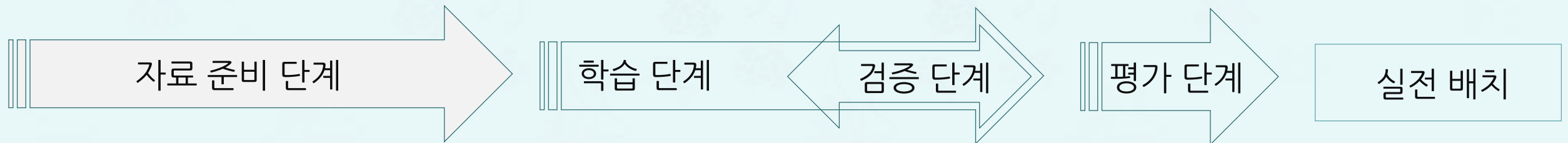
1. 회귀 분석: 로지스틱 회귀 (Logistic Regression)

- 0, 1 / Pass, Fail
- 두 가지로 결론을 내는 회귀 분석

시험 성적(x)	합격(y)
100	P
45	F
50	P
30	F
95	P
...	...



2. 로지스틱 회귀: 학습자료 준비



2. 로지스틱 회귀: 학습자료 준비


- 학습자료 읽기
- 시각화
- 이항 분류
 - 두 개의 클래스로 분류



```
1 import joy
2 X, Y = joy.planar_data()
3 m = X.shape[1]
4 print ('X.shape={} Y.shape={}, m={}'.
5         format(X.shape, Y.shape, m))
6 joy.plot_xyw(X.T, Y.squeeze(), title="PlanaData")
```

X.shape=(2, 400) Y.shape=(1, 400), m=400

2. 로지스틱 회귀: 학습자료 준비



```
1 import joy
2 X, Y = joy.planar_data()
3 m = X.shape[1]
4 print ('X.shape={} Y.shape={}, m={}'.
5         format(X.shape, Y.shape, m))
6 joy.plot_xyw(X.T, Y.squeeze(),title="PlanaData")
```

X.shape=(2, 400) Y.shape=(1, 400), m=400

2. 로지스틱 회귀: 학습자료 준비

```
1 import joy
2 X, Y = joy.planar_data()
3 m = X.shape[1]
4 print ('X.shape={} Y.shape={}, m={}'.
5         format(X.shape, Y.shape, m))
6 joy.plot_xyw(X.T, Y.squeeze(),title="PlanaData")
```


X.shape=(2, 400) Y.shape=(1, 400), m=400



2. 로지스틱 회귀: 학습자료 준비

```
1 import joy
2 X, Y = joy.planar_data()
3 m = X.shape[1]
4 print ('X.shape={} Y.shape={}, m={}'.
5         format(X.shape, Y.shape, m))
6 joy.plot_xyw(X.T, Y.squeeze(), title="PlanaData")
```


X.shape=(2, 400) Y.shape=(1, 400), m=400


$$X = \begin{pmatrix} x_1^{(1)} & x_2^{(1)} & \cdots & x_{400}^{(1)} \\ x_1^{(2)} & x_2^{(2)} & \cdots & x_{400}^{(2)} \end{pmatrix}$$

2. 로지스틱 회귀: 학습자료 준비

```
1 import joy
2 X, Y = joy.planar_data()
3 m = X.shape[1]
4 print ('X.shape={} Y.shape={}, m={}'.
5         format(X.shape, Y.shape, m))
6 joy.plot_xyw(X.T, Y.squeeze(),title="PlanaData")
```

X.shape=(2, 400) Y.shape=(1, 400), m=400


$$Y = \begin{pmatrix} y_1^{(1)} & y_2^{(1)} & \cdots & y_{400}^{(1)} \end{pmatrix}$$

2. 로지스틱 회귀: 시각화

```
1 import joy
2 X, Y = joy.planar_data()
3 m = X.shape[1]
4 print ('X.shape={} Y.shape={}, m={}'.
5         format(X.shape, Y.shape, m))
6 joy.plot_xyw(X.T, Y.squeeze(),title="PlanaData")
```

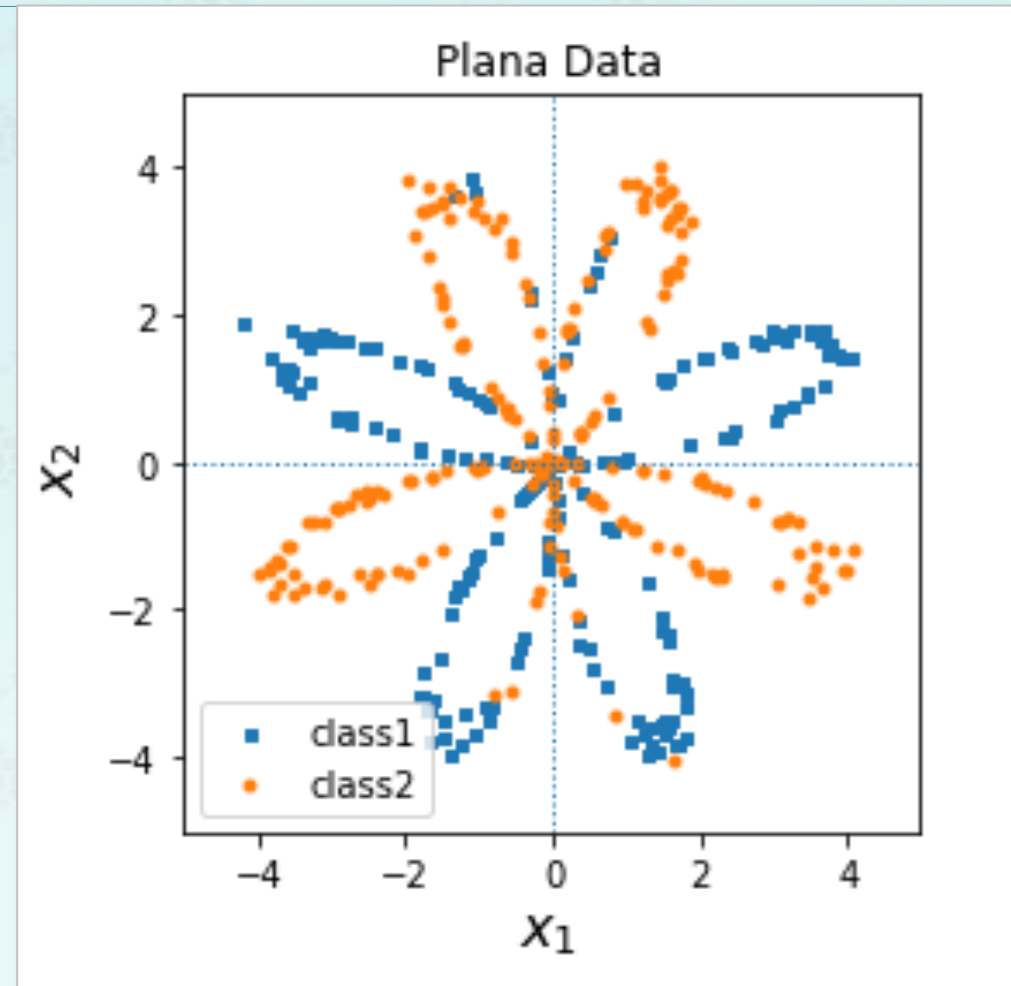
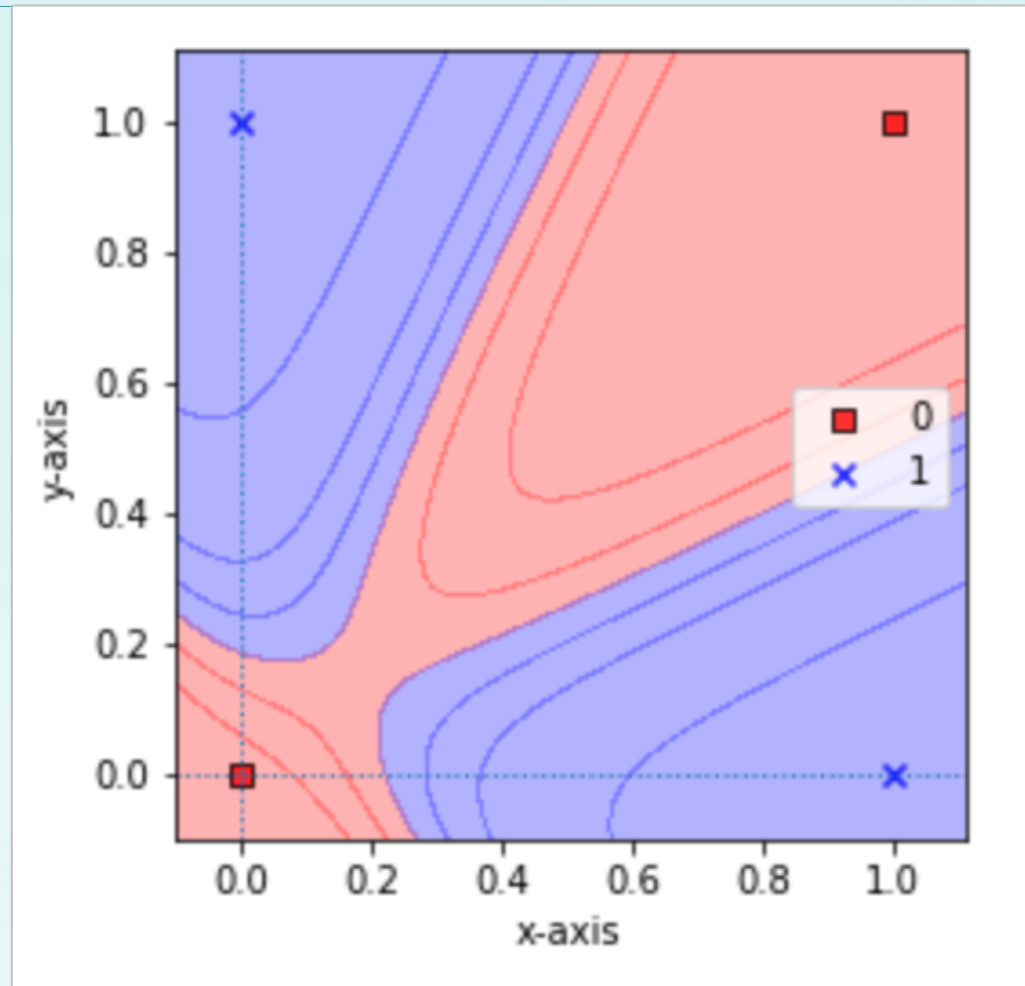
X.shape=(2, 400) Y.shape=(1, 400), m=400

2. 로지스틱 회귀: 시각화

```
1 import joy
2 X, Y = joy.planar_data()
3 m = X.shape[1]
4 print ('X.shape={} Y.shape={}, m={}'.
5         format(X.shape, Y.shape, m))
6 joy.plot_xyw(X.T, Y.squeeze(), title="PlanaData")
```


X.shape=(2, 400) Y.shape=(1, 400), m=400

2. 로지스틱 회귀: 시각화




3. 신경망 모델: 3층 신경망

- 입력 노드: 2개
- 은닉 노드: 3개
- 출력 노드: 1개




```
1  n_h = 3
2  net_arch=[2, n_h, 1]
3  nn = joy.NeuralNetwork(net_arch, eta=0.1, epochs=100)
4  nn.fit(X, Y)
5
6  joy.plot_decision_regions(X.T, Y, nn)
7  yhat = nn.predict(X.T)
8  accuracy = float(np.dot(Y, yhat.T) +
9                  np.dot(1 - Y, 1 - yhat.T))/Y.size * 100
10 plt.title('Hidden neurons={}, accuracy={}'.
11           format(n_h, np.round(accuracy,2)))
```


3. 신경망 모델: 3층 신경망



```
1  n_h = 3
2  net_arch=[2, n_h, 1]
3  nn = joy.NeuralNetwork(net_arch, eta=0.1, epochs=100)
4  nn.fit(X, Y)
5
6  joy.plot_decision_regions(X.T, Y, nn)
7  yhat = nn.predict(X.T)
8  accuracy = float(np.dot(Y, yhat.T) +
9                  np.dot(1 - Y, 1 - yhat.T))/Y.size * 100
10 plt.title('Hidden neurons={}, accuracy={}'.
11           format(n_h, np.round(accuracy, 2)))
```

3. 신경망 모델: 3층 신경망



```
1  n_h = 3
2  net_arch=[2, n_h, 1]
3  nn = joy.NeuralNetwork(net_arch, eta=0.1, epochs=100)
4  nn.fit(X, Y)
5
6  joy.plot_decision_regions(X.T, Y, nn)
7  yhat = nn.predict(X.T)
8  accuracy = float(np.dot(Y, yhat.T) +
9                  np.dot(1 - Y, 1 - yhat.T))/Y.size * 100
10 plt.title('Hidden neurons={}, accuracy={}'.
11           format(n_h, np.round(accuracy, 2)))
```

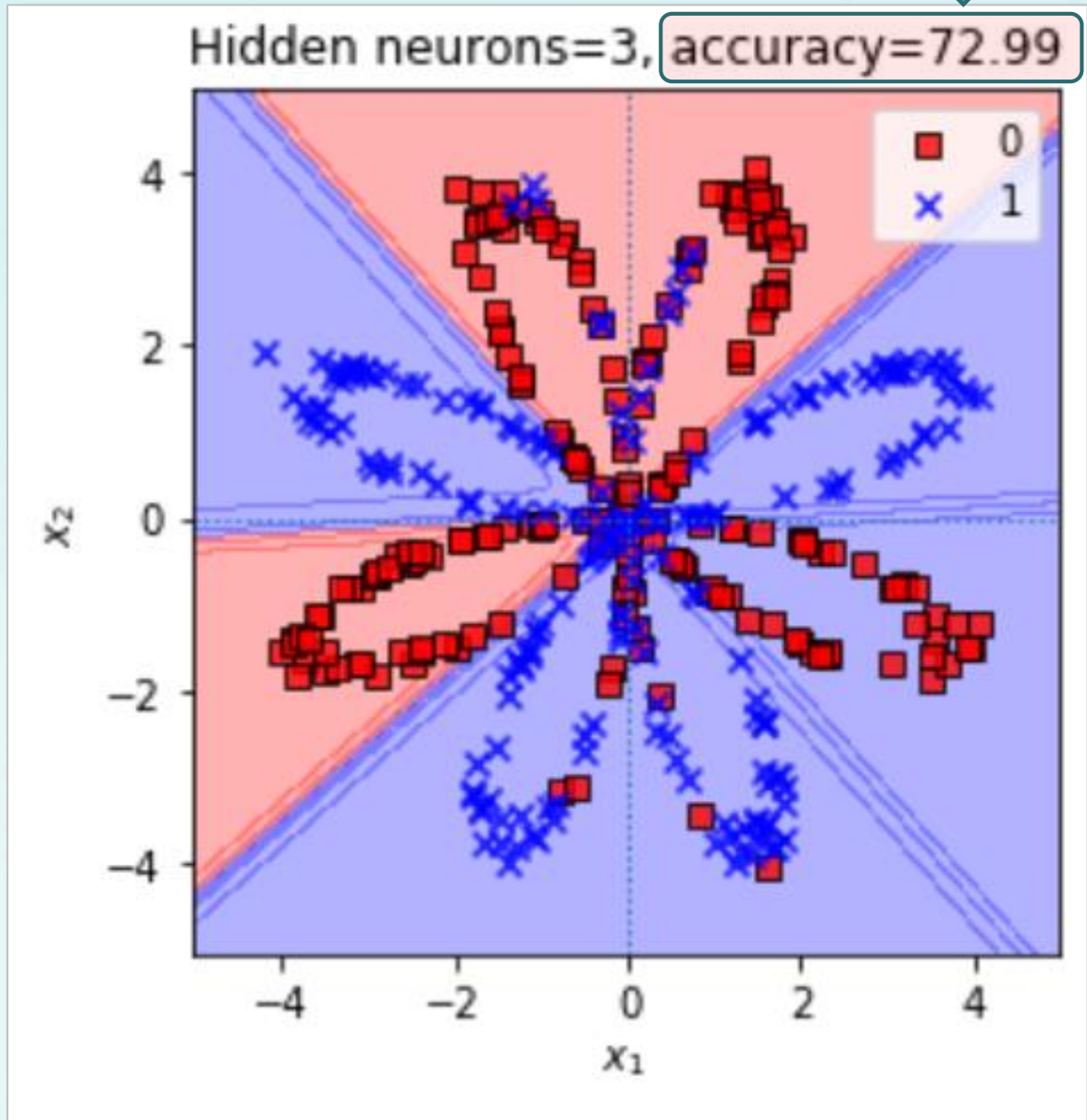
3. 신경망 모델: 3층 신경망

```
1  n_h = 3
2  net_arch=[2, n_h, 1]
3  nn = joy.NeuralNetwork(net_arch, eta=0.1, epochs=100)
4  nn.fit(X, Y)
5
6  joy.plot_decision_regions(X.T, Y, nn)
7  yhat = nn.predict(X.T)
8  accuracy = float(np.dot(Y, yhat.T) +
9                  np.dot(1 - Y, 1 - yhat.T))/Y.size * 100
10 plt.title('Hidden neurons={}, accuracy={}'.
11           format(n_h, np.round(accuracy,2)))
```

3. 신경망 모델: 3층 신경망

```
1  n_h = 3
2  net_arch=[2, n_h, 1]
3  nn = joy.NeuralNetwork(net_arch, eta=0.1, epochs=100)
4  nn.fit(X, Y)
5
6  joy.plot_decision_regions(X.T, Y, nn)
7  yhat = nn.predict(X.T)
8  accuracy = float(np.dot(Y, yhat.T) +
9                  np.dot(1 - Y, 1 - yhat.T))/Y.size * 100
10 plt.title('Hidden neurons={}, accuracy={}'.
11           format(n_h, np.round(accuracy,2)))
```

3. 신경망 모델: 결과



3. 신경망 모델: 은닉 뉴런 개수 증가

```
1 X, Y = joy.planar_data()
2 plt.figure(figsize=(12, 24))
3 accuracy = []
4 number_of_neurons = [2, 3, 4, 8, 16, 32, 64, 128]
5 for i, n_h in enumerate(number_of_neurons):
6     print('[{}] Processing {} neurons case....'.format(i, n_h))
7     net_arch = [2, 1]
8     net_arch.insert(1, n_h)
9     nn = joy.NeuralNetwork(net_arch, eta=0.1, epochs=200)
10    nn.fit(X, Y)                                # train the net
11
12    plt.subplot(5, 2, i+1)
13    joy.plot_decision_regions(X.T, Y, nn)
14    yhat = nn.predict(X.T)
15    accuracy.append(float(np.dot(Y, yhat.T) +
16                          np.dot(1 - Y, 1 - yhat.T))/Y.size * 100)
17    plt.title('Hidden neurons={}, accuracy={}'.
18              format(n_h, np.round(accuracy[i], 2)))
```


3. 신경망 모델: 은닉 뉴런 개수 증가

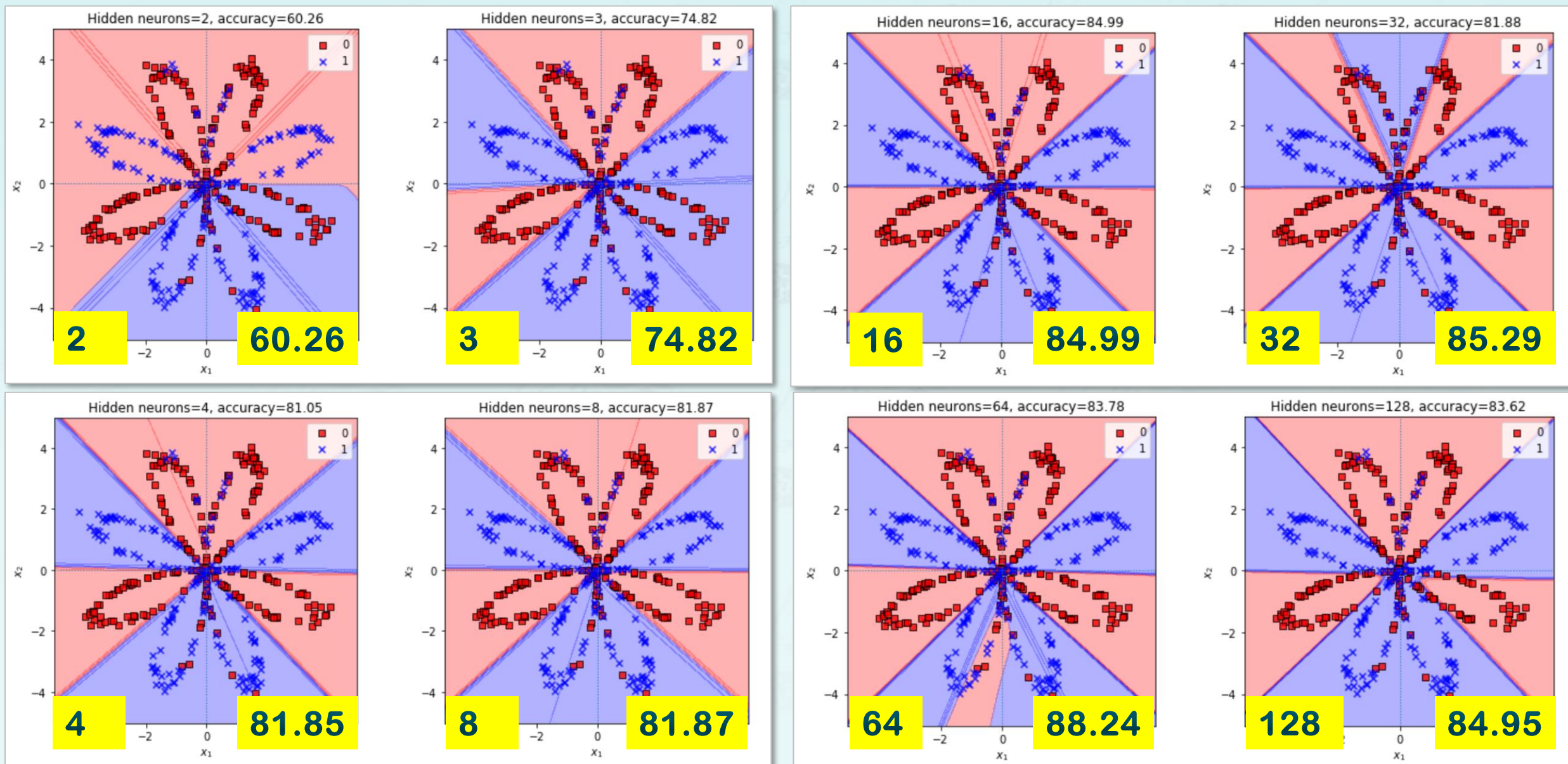


```
1 X, Y = joy.planar_data()
2 plt.figure(figsize=(12, 24))
3 accuracy = []
4 number_of_neurons = [2, 3, 4, 8, 16, 32, 64, 128]
5 for i, n_h in enumerate(number_of_neurons):
6     print('[{}] Processing {} neurons case....'.format(i, n_h))
7     net_arch = [2, 1]
8     net_arch.insert(1, n_h)
9     nn = joy.NeuralNetwork(net_arch, eta=0.1, epochs=200)
10    nn.fit(X, Y) # train the net
11
12    plt.subplot(5, 2, i+1)
13    joy.plot_decision_regions(X.T, Y, nn)
14    yhat = nn.predict(X.T)
15    accuracy.append(float(np.dot(Y, yhat.T) +
16                          np.dot(1 - Y, 1 - yhat.T))/Y.size * 100)
17    plt.title('Hidden neurons={}, accuracy={}'.
18              format(n_h, np.round(accuracy[i], 2)))
```

3. 신경망 모델: 은닉 뉴런 개수 증가

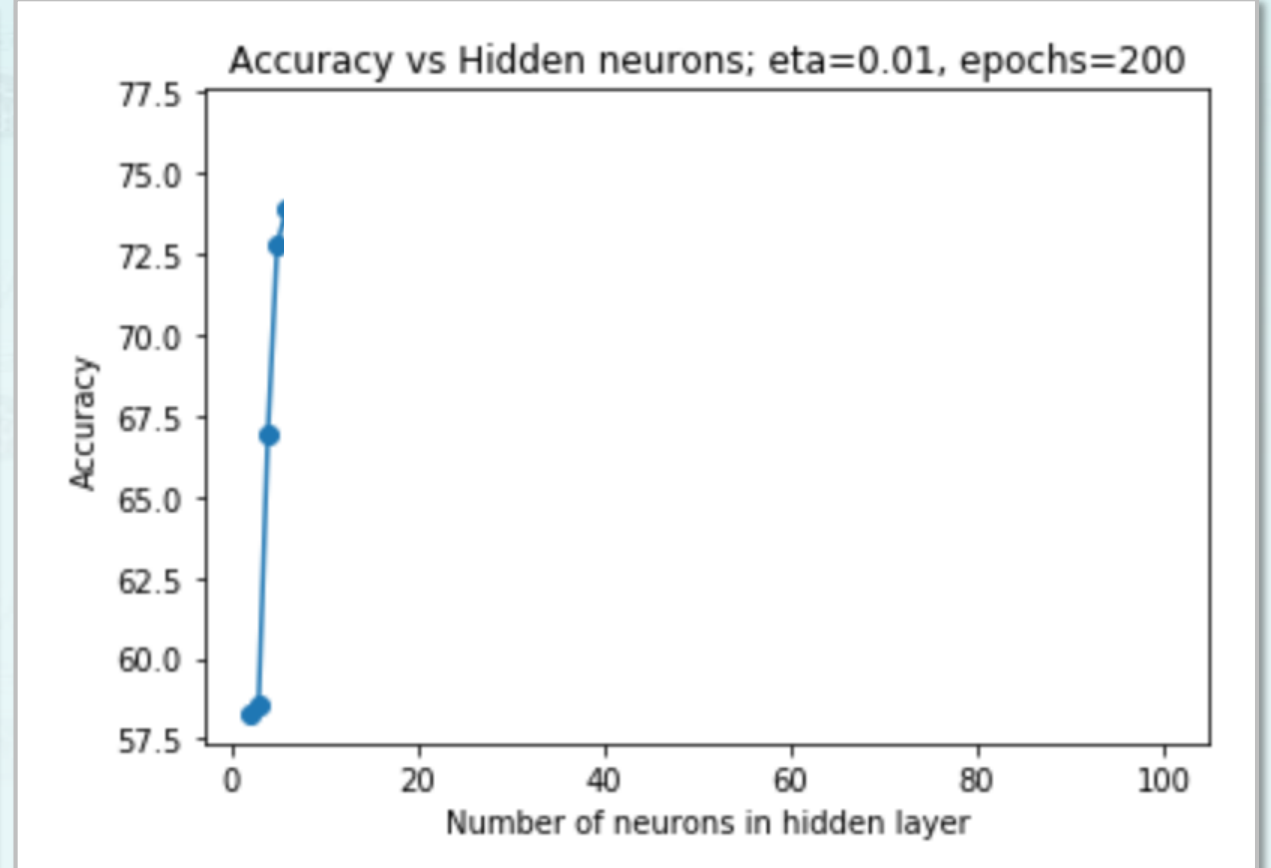
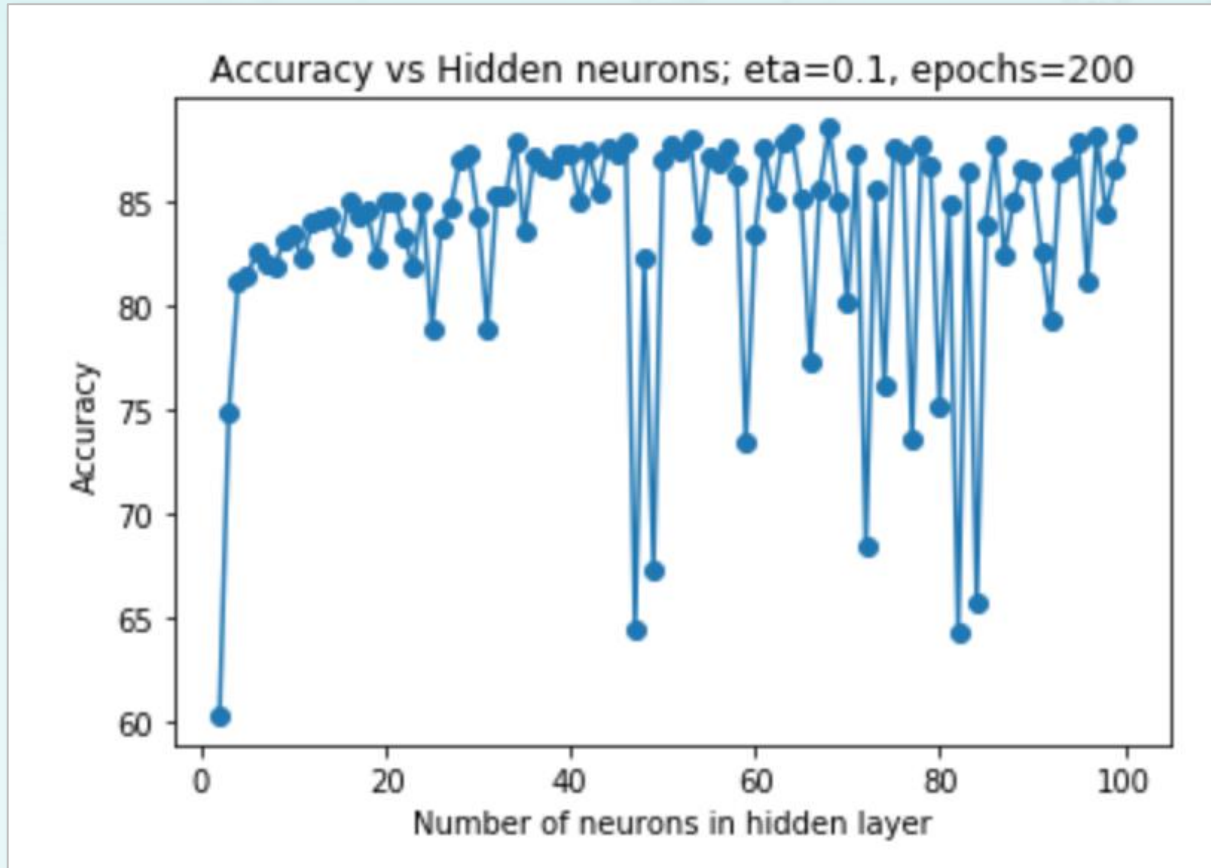
```
1 X, Y = joy.planar_data()
2 plt.figure(figsize=(12, 24))
3 accuracy = []
4 number_of_neurons = [2, 3, 4, 8, 16, 32, 64, 128]
5 for i, n_h in enumerate(number_of_neurons):
6     print('[{}] Processing {} neurons case....'.format(i, n_h))
7     net_arch = [2, 1]
8     net_arch.insert(1, n_h)
9     nn = joy.NeuralNetwork(net_arch, eta=0.1, epochs=200)
10    nn.fit(X, Y) # train the net
11
12    plt.subplot(5, 2, i+1)
13    joy.plot_decision_regions(X.T, Y, nn)
14    yhat = nn.predict(X.T)
15    accuracy.append(float(np.dot(Y, yhat.T) +
16                          np.dot(1 - Y, 1 - yhat.T))/Y.size * 100)
17    plt.title('Hidden neurons={}, accuracy={}'.
18              format(n_h, np.round(accuracy[i], 2)))
```


3. 신경망 모델: 은닉 뉴런 개수 증가

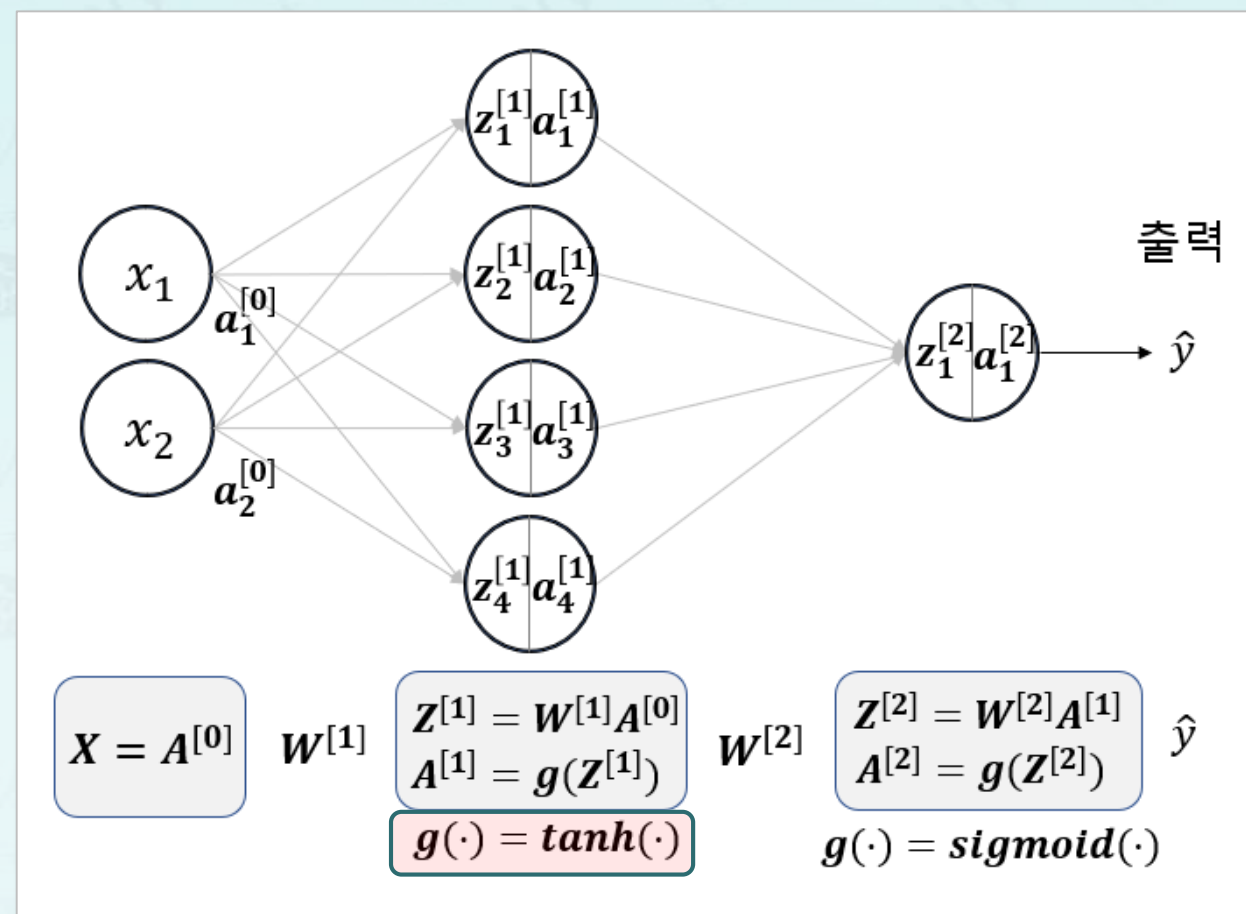
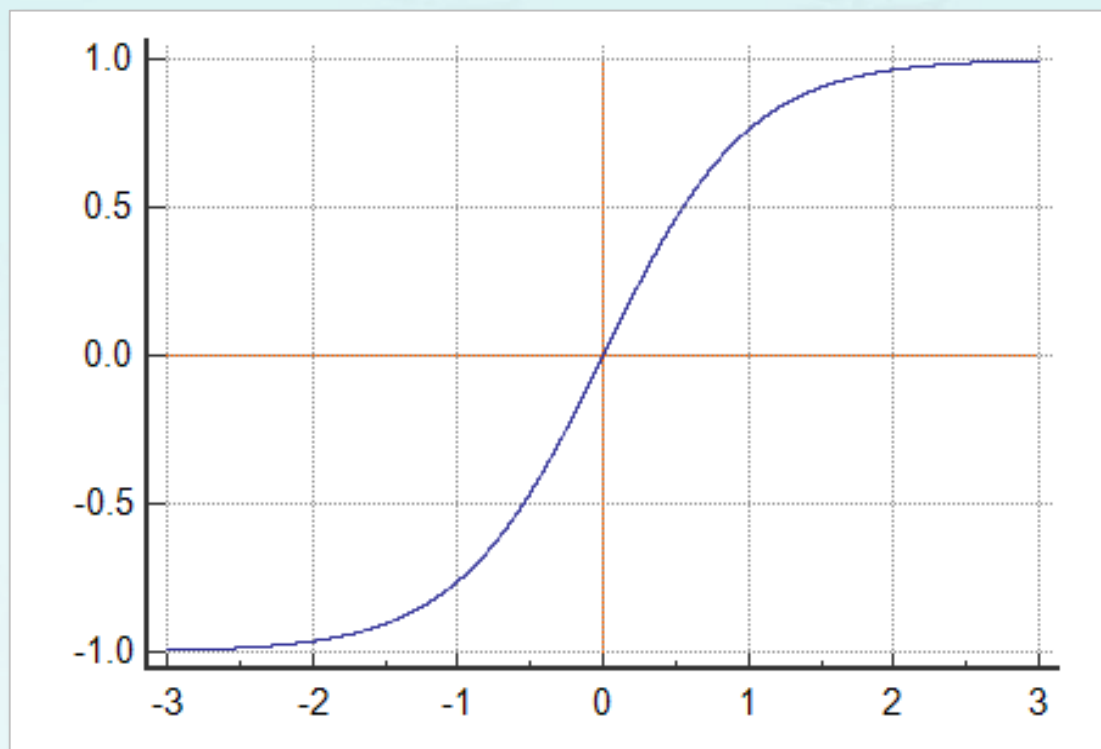


3. 신경망 모델: 신경망의 불안정성 해결 방법

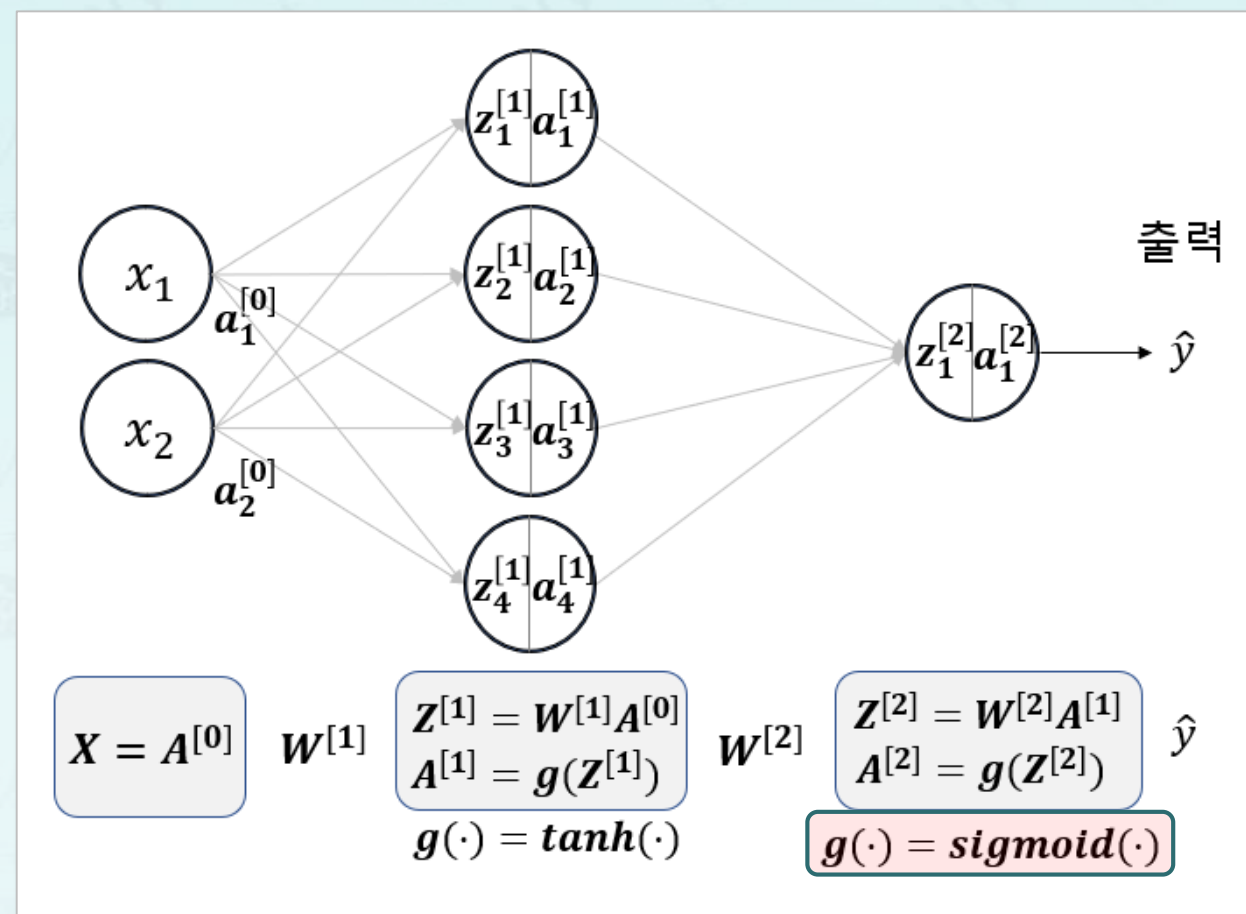
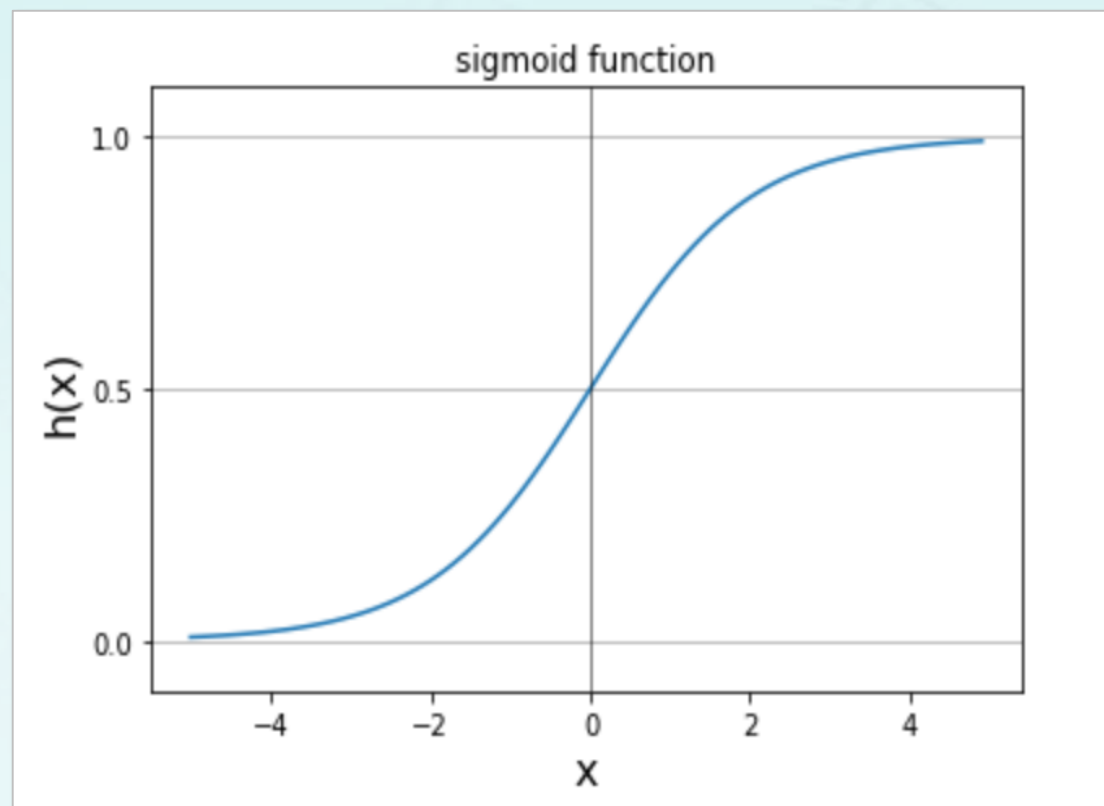
- 학습률 $\eta=0.1$, epochs=200
- 학습률 $\eta=0.01$, epochs=200



4. 새로운 3층 신경망: 쌍곡 탄젠트

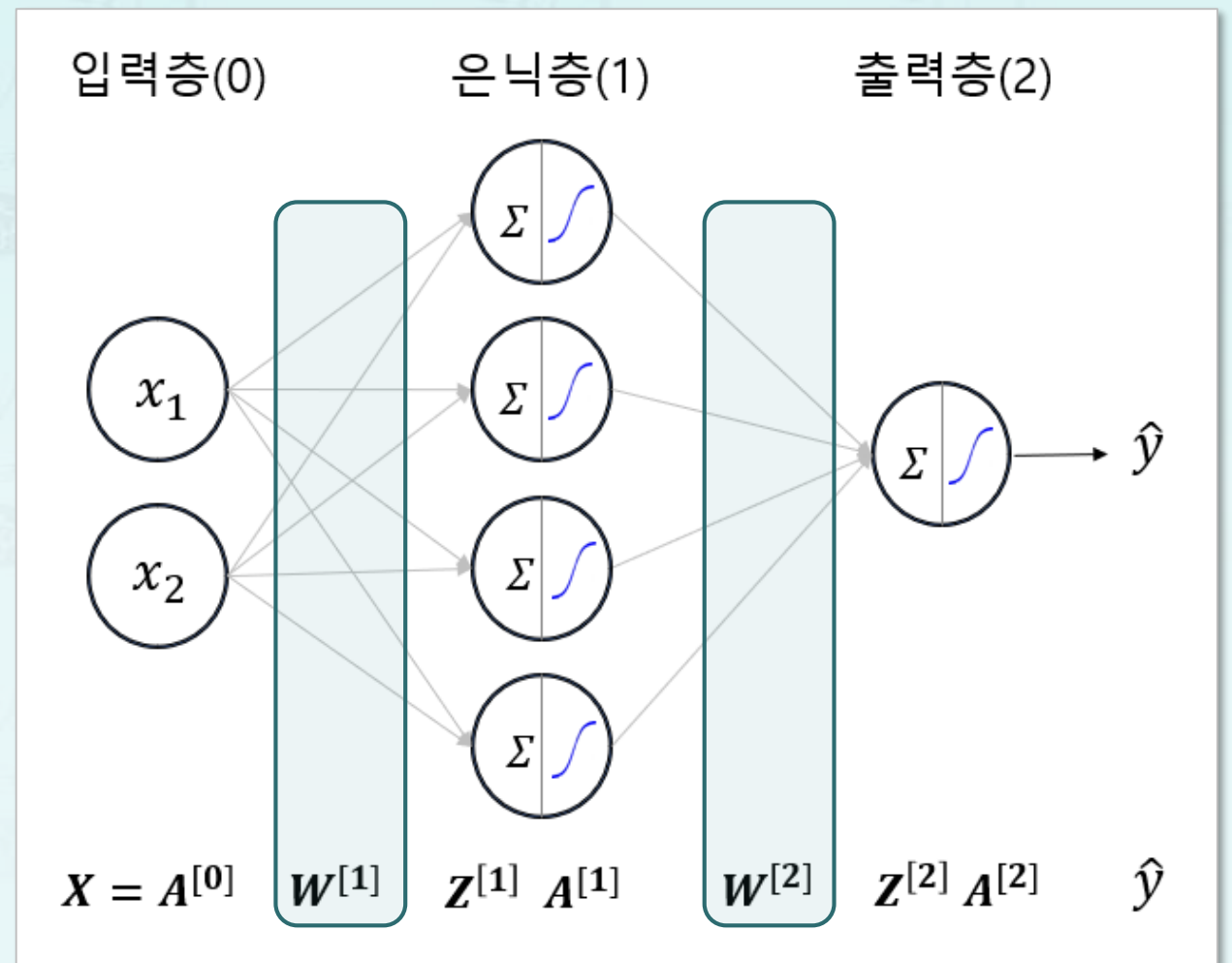


4. 새로운 3층 신경망: 시그모이드



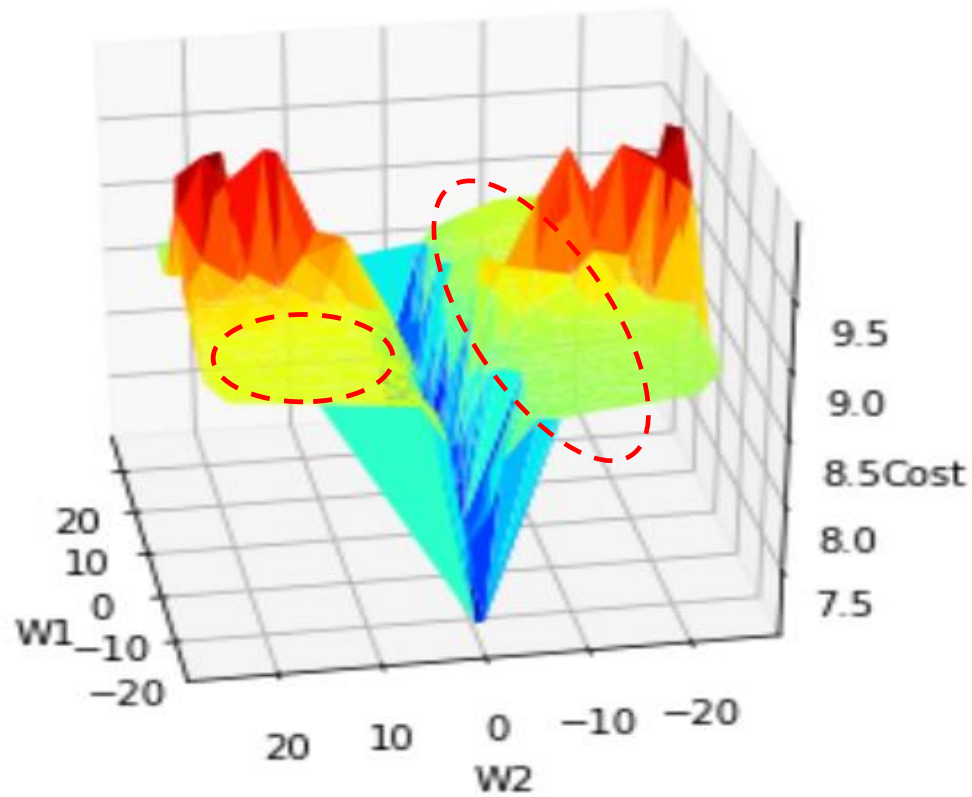
4. 새로운 3층 신경망: 제곱 합 오차(SSE)

$$J(h(z), y) = \frac{1}{m} \sum_{i=1}^m \frac{1}{2} (h(z) - y^{(i)})^2$$



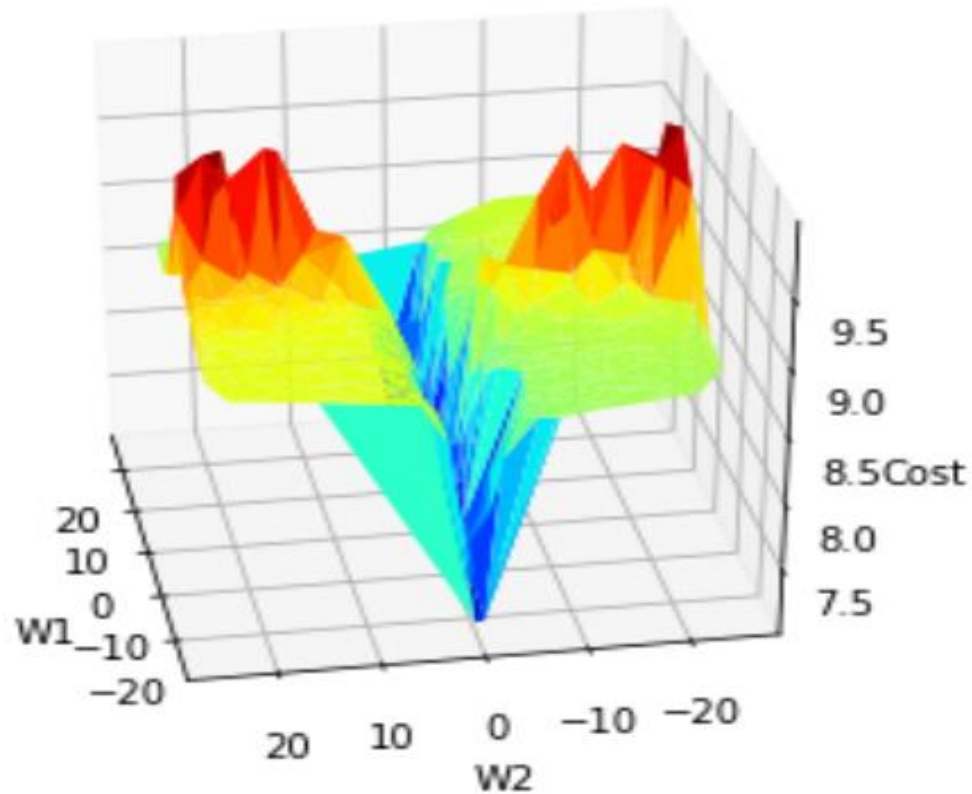
4. 새로운 3층 신경망: 제곱 합 오차(SSE)

$$J(h(z), y) = \frac{1}{m} \sum_{i=1}^m \frac{1}{2} (h(z) - y^{(i)})^2$$

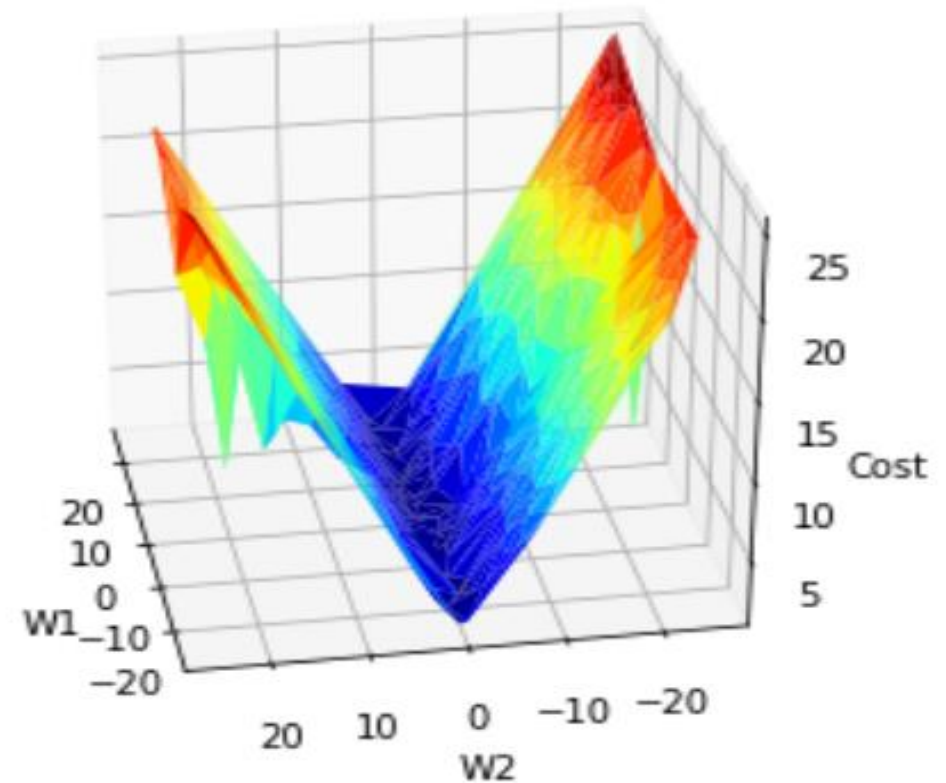


4. 새로운 3층 신경망: 교차 엔트로피(Cross Entropy)

$$J(h(z), y) = \frac{1}{m} \sum_{i=1}^m \frac{1}{2} (h(z) - y^{(i)})^2$$



$$J(h(z), y) = - \sum_i y^{(i)} \log(\hat{y}^{(i)})$$



5. 로지스틱 함수: 로지스틱 회귀

- 결과 값 : 확률 ($0 \sim 1$)
- **Ex.)** 비 올 확률,
카드 사용 패턴 진단

5. 로지스틱 함수: 선형 회귀 분석

- 연속적인 값 예측(하나의 값)
- **Ex.)** 집 값 예측

$$Z = W \cdot X$$

5. 로지스틱 함수: 로지스틱 회귀

- 결과 값 : 확률 ($0 \sim 1$)
- **Ex.)** 비 올 확률,
카드 사용 패턴 진단
- 분류

5. 로지스틱 함수: 로지스틱 회귀

- 결과 값 : 확률 (0 ~ 1)
- **Ex.)** 비 올 확률,
카드 패턴 진단
- 분류
- 승산비(odds ratio)

$$\frac{Sucess}{Fail} = \frac{p}{1-p} \quad \begin{cases} 0 & \text{if } p \rightarrow 0. \\ +\infty & \text{if } p \rightarrow 1. \end{cases}$$

5. 로지스틱 함수: 로짓 함수 (logit function)

- 승산비 (odds ratio)

$$\frac{Success}{Fail} = \frac{p}{1-p} \quad \begin{cases} 0 & \text{if } p \rightarrow 0. \\ +\infty & \text{if } p \rightarrow 1. \end{cases}$$

- 로짓 함수 (logit function)

$$\text{logit(odds ratio)} = \log_e \frac{p}{1-p} \quad \begin{cases} -\infty & \text{if } p \rightarrow 0. \\ +\infty & \text{if } p \rightarrow 1. \end{cases}$$

$$e^{-\infty} = 0$$

5. 로지스틱 함수: 로짓 함수 (logit function)

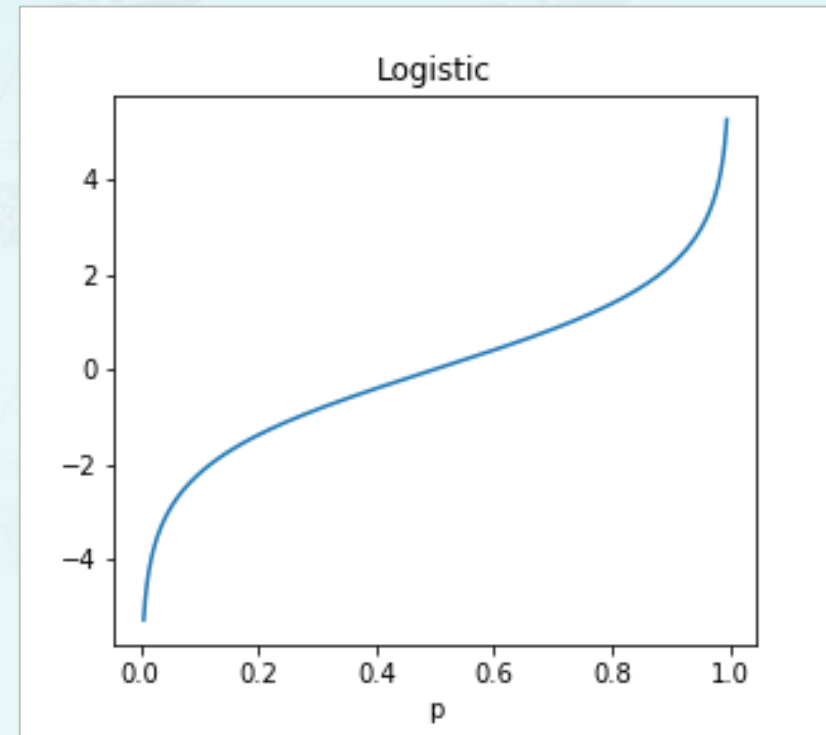
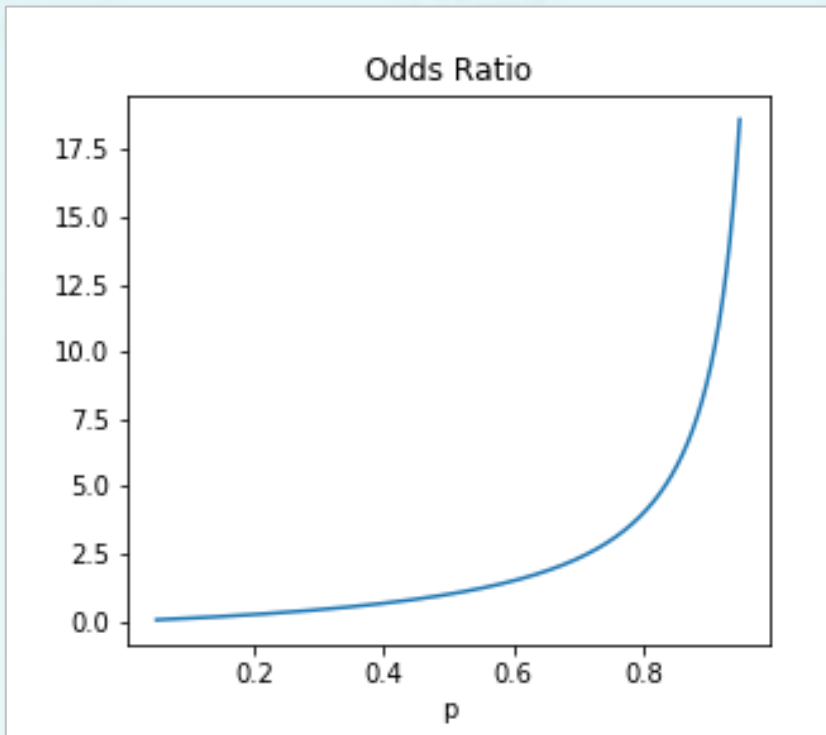
- 승산비 (odds ratio)

$$\frac{Success}{Fail} = \frac{p}{1-p} \quad \begin{cases} 0 & \text{if } p \rightarrow 0. \\ +\infty & \text{if } p \rightarrow 1. \end{cases}$$

- 로짓 함수 (logit function)

$$\text{logit}(\text{odds ratio}) = \log_e \frac{p}{1-p} \quad \begin{cases} -\infty & \text{if } p \rightarrow 0. \\ +\infty & \text{if } p \rightarrow 1. \end{cases}$$

$$e^{-\infty} = 0$$



5. 로지스틱 함수: 로짓 함수 (logit function) 정리

$$\log_e \frac{p}{1-p} = W \cdot X$$

$$\frac{p}{1-p} = e^{W \cdot X}$$

$$\begin{aligned} \text{odds ratio} &= \frac{1-p}{p} = \frac{1}{e^{W \cdot X}} \\ \frac{1}{p} - 1 &= \frac{1}{e^{W \cdot X}} \\ \frac{1}{p} &= \frac{1}{e^{W \cdot X}} + 1 \\ &= \frac{1}{e^{W \cdot X}} + \frac{e^{W \cdot X}}{e^{W \cdot X}} \\ &= \frac{1 + e^{W \cdot X}}{e^{W \cdot X}} \\ p &= \frac{e^{W \cdot X}}{1 + e^{W \cdot X}} \end{aligned}$$

5. 로지스틱 함수: 로짓 함수 (logit function) 정리

$$\begin{aligned} p &= \frac{e^{W \cdot X}}{1 + e^{W \cdot X}} \\ &= \frac{1}{\frac{1}{e^{W \cdot X}} + 1} \\ &= \frac{1}{1 + e^{-W \cdot X}} \end{aligned}$$

$$h(z) = \frac{1}{1 + e^{-W \cdot X}} = \frac{1}{1 + e^{-Z}}$$

로지스틱 함수 : 로짓 함수의 역함수

$$\begin{aligned} \text{odds ratio} &= \frac{1-p}{p} = \frac{1}{e^{W \cdot X}} \\ \frac{1}{p} - 1 &= \frac{1}{e^{W \cdot X}} \\ \frac{1}{p} &= \frac{1}{e^{W \cdot X}} + 1 \\ &= \frac{1}{e^{W \cdot X}} + \frac{e^{W \cdot X}}{e^{W \cdot X}} \\ &= \frac{1 + e^{W \cdot X}}{e^{W \cdot X}} \end{aligned}$$

$$p = \frac{e^{W \cdot X}}{1 + e^{W \cdot X}}$$

로지스틱 회귀

- 학습 정리
 - 오차 함수들의 분석과 차이
 - 3층 신경망에 두 개의 활성화 함수 사용
 - 로지스틱 함수와 로지스틱 회귀 알고리즘