머신러닝
(Machine Learning)

LECTURE XVI: 지도 학습 10
(Supervised Learning)

Dai-Gyoung Kim

Department of Applied Mathematics

Hanyang University ERICA

지도 학습
(Supervised Learning)

Contents

- 분류와 회귀
- 일반화, 과대적합, 과소적합
- 지도 학습 알고리즘
  - k-최근접 이웃
  - 선형모델
  - 나이브 베이즈 모델
  - 결정트리
  - 결정트리의 앙상블
  - 커널 서포트 벡터 머신
  - 신경망, 딥러닝
- 분류 예측의 불확실성 추정

▶ 신경망(딥러닝)

인공 신경망의 역사

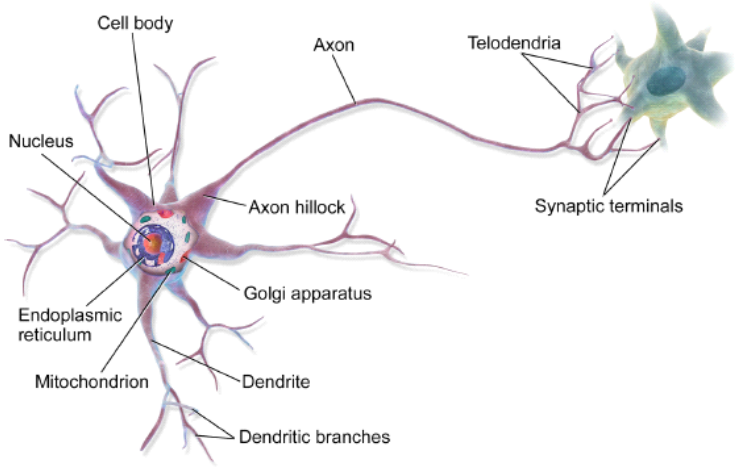
- **제 1 기 (1940 ~ 1980) 이론적인 환경**
  - [1943] W. McCulloch(신경생리학자), W. Pitts(수학자): 명제 논리를 기반으로 한 임계 논리(threshold logic) 알고리즘을 바탕으로 신경망을 위한 계산 모델을 소개.
  - [1958] F. Rosenblatt: 퍼셉트론으로 간단한 덧셈과 뺄셈을 하는 이층구조의 학습 컴퓨터 망에 근거한 패턴 인식을 위한 알고리즘을 소개.
  - [1969] Marvin Minsky, Seymour Papert: 단순 퍼셉트론은 선형 분리가 불가능한 패턴을 식별할 수 없다는 것을 밝힘.
    - ✓ 단층 신경망은 배타적 논리합 회로를 처리하지 못함.
    - ✓ 거대한 신경망에 의해 처리되는 긴 시간을 컴퓨터가 충분히 효과적으로 처리하지 못함.
  - [1969 ~ 1980] 인공 신경망의 침체기

- **제 2 기 (1980 ~ 2000) 컴퓨터 환경**
  - [1980] Fukushima의 연구진: 네오코크니트론(neocognitron)을 제안함.
    - ✓ 생물의 시각 피질을 모방한 합성곱층 기법을 제안.
  - [1982] Hopfield: 홉필드 모형 제안.
  - [1986] Rumelhart 연구진: 역전파법(back propagation)을 제안.
  - [1989] LeCun 연구진: 합성곱 신경망 제안.
  - [1989 ~ 2000] 서포트 벡터 머신의 등장으로 인공 신경망이 주목 받지 못함.

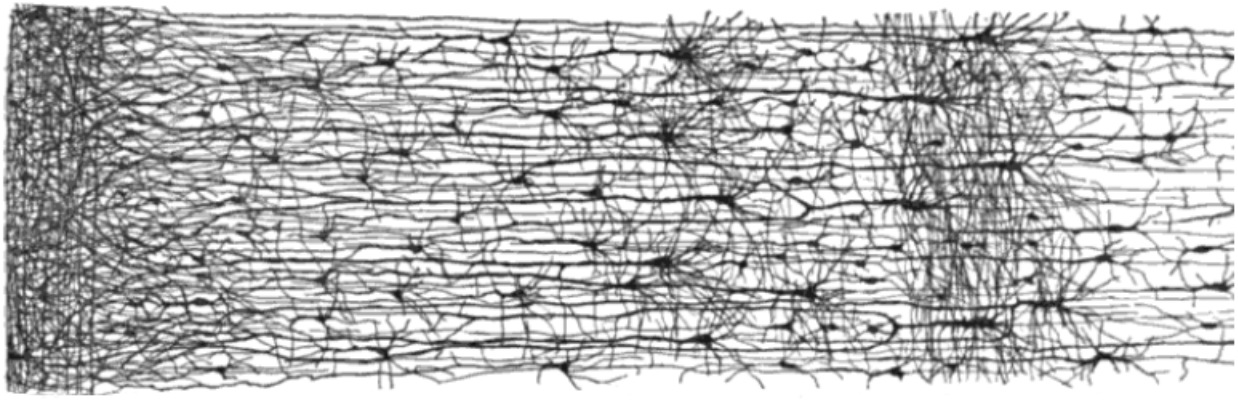
- **제 3 기 (2000 ~) 컴퓨터 환경 + 네트워크 환경 + 빅데이터**
  - [2006] Hinton, Bengio: 딥뉴럴 넷에 사전훈련과 자기부호화 기법을 제안.
  - [2011] Seide 연구진: 음성 인식 벤치마크 테스트에서 압승을 거둠.
  - [2012] Krizhevsky 연구진: ReLU를 제안하고 이미지 인식 테스트에서 압승을 거둠.
  - [2016] 구글 딥마인드: 알파고의 압승

생물학적 뉴런

- 뉴런 세포(Neuron Cells)는 대부분의 동물의 대뇌 피질(cerebral cortexes)에서 발견되며 핵을 포함하는 세포체(cell body)와 세포의 복잡한 구성 요소로 이루어 있음.
  - ✓ **수상돌기**(dendrite)라는 나뭇가지 모양의 돌기
  - ✓ **축삭돌기**(Axon)
  - ✓ **축삭끝가지**(telodendria)
  - ✓ **시냅스**(Synapse): 한 뉴런에서 다른 세포로 신호를 전달하는 연결 지점.

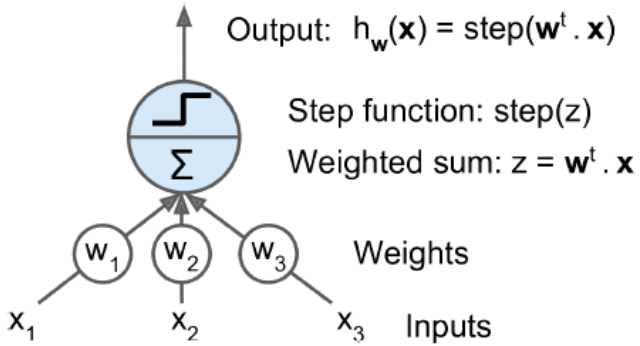


- 뉴런은 시냅스를 통해 다른 뉴런으로부터 짧은 전기 신호를 받음. 이때, 한 뉴런이 천분의 몇 초 동안 다른 뉴런으로부터 충분한 횟수의 신호를 받으면 자체적인 신호를 발생시킴.
- 개개의 뉴런은 아주 단순하게 동작하지만, 수십억 개의 뉴런으로 구성된 거대한 네트워크로 조직되어 있고 각 뉴런은 보통 수천 개의 다른 뉴런과 연결되어 있음.
- 다음 그림은 인간 대뇌 피질의 생물학적 신경망이 여러 층으로 조직화되어 있는 것을 보여줌.



퍼셉트론

- 퍼셉트론(Perceptron)은 1957년 Frank Rosenblatt가 제안한 가장 간단한 인공 신경망 구조 중 하나임.
- 퍼셉트론은 TLU(threshold logic unit)라는 형태의 인공 뉴론을 기반으로 함.



✓ 입력과 출력이 이진 값이 아니라 어떤 숫자이며 각각의 입력 연결은 가중치와 연관되어 있음.

✓ 퍼셉트론은 입력의 가중치 합 ($z = w^T x$)을 계산하고, 그 계산된 값에 계단함수(step function)를 적용하여 그 결과를 출력함: $h_w(x) = \text{step}(z)$.

- 퍼셉트론에서 일반적으로 사용되는 계단 함수는 다음과 같음.

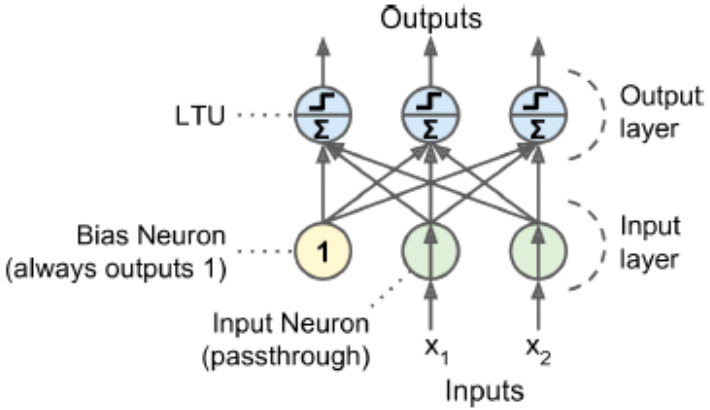
$$H(z) = \begin{cases} 0 & \text{if } z < 0 \\ 1 & \text{if } z \geq 0 \end{cases}, \qquad \text{sign}(z) = \begin{cases} -1 & \text{if } z < 0 \\ 0 & \text{if } z = 0 \\ 1 & \text{if } z > 0 \end{cases}$$

- 하나의 퍼셉트론은 간단한 선형 이진 분류기 문제에 사용할 수 있음.
- 퍼셉트론을 훈련시킨다는 것은 최적의 가중치 w를 결정하는 것임.

[퍼셉트론 구성]

- 퍼셉트론은 층이 하나뿐인 TLU로 구성됨. 각 뉴런은 모든 입력에 연결되어 있고, 이 연결은 입력 뉴런(input neuron)이라 부르는 특별한 **통과 뉴런**을 사용해 표현됨.
- 통과 뉴런은 무엇이 주입되든 입력을 그대로 출력으로 통과시키며, 보통 이 뉴런에 편향 특성이 더해짐. 전형적으로 편향 특성은 항상 1을 출력하는 특별한 종류의 뉴런인 **편향 뉴런**(bias neuron)으로 표현됨.

- 다음 그림은 입력 두 개와 출력 세 개로 구성된 퍼셉트론의 예임.



✓ 위의 퍼셉트론은 샘플을 세 개의 다른 이진 클래스로 동시에 분류할 수 있는 일종의 다중 출력 분류기(multioutput classifier)라 할 수 있음.

퍼셉트론의 훈련

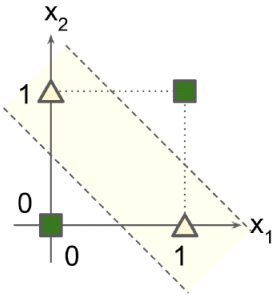
- **헤브의 규칙**(Hebb's rule): "서로 활성화되는 세포는 서로 연결이 강화됨" 즉, 두 뉴런이 동일한 출력을 낼 때마다 **그들 사이의 연결 가중치**가 증가함.
- 퍼셉트론에 한 번에 한 개의 샘플이 주입되면 각 샘플에 대한 예측이 만들어 짐.
  - ✓ 이때, 잘못된 예측을 하는 모든 출력 뉴런에 대해 올바른 예측을 만들 수 있도록 입력에 연결된 가중치를 갱신함.
- 퍼셉트론에 학습 규칙(가중치 업데이트)

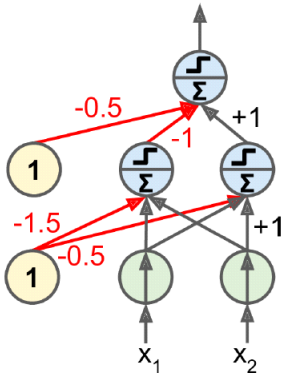
$$w_{ij}^{(\text{next update})} = w_{ij} + \eta\left(y_j - \hat{y}_j\right)x_i$$

  - ✓ w_{ij}는 i번째 입력 뉴런과 j번째 출력 뉴런 사이를 연결하는 가중치.
  - ✓ x_i는 현재 훈련 샘플의 i번째 뉴런의 입력값.
  - ✓ $\hat{y}_j$는 현재 훈련 샘플의 j번째 출력 뉴런의 출력값.
  - ✓ y_j는 현재 훈련 샘플의 j번째 출력 뉴런의 타깃값.
  - ✓ η는 학습률.

퍼셉트론의 약점

- 간단한 배타적 논리합(XOR) 분류 문제를 풀 수 없음. (1969: M. Minsky, S. Papert)



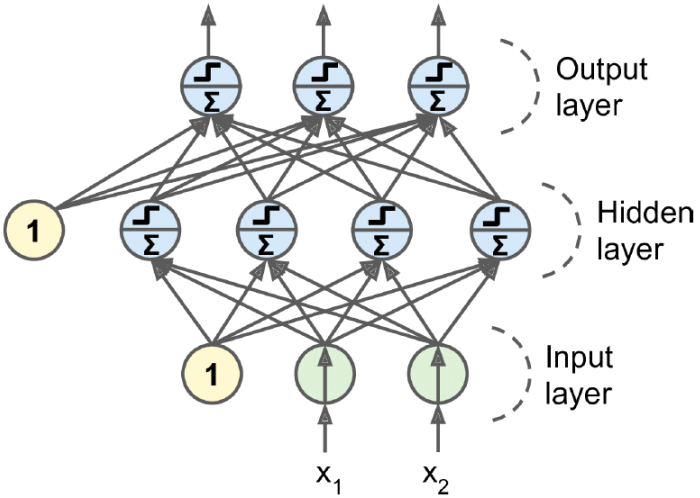


❖ 신경망 모델

[다층 퍼셉트론]

- 여러 퍼셉트론을 쌓아올려 일부 제약을 줄일 수 있음. 이런 인공 신경망을 **다층 퍼셉트론**(Multi-Layer Perceptron, MLP)이라고 함.

- **다층 퍼셉트론은 (통과) 입력층 하나와 은닉층(hidden layer)이라 불리는 하나 이상의 TLU 층과 마지막 층인 출력층(output layer)으로 구성되어 있음.**
- 출력층을 제외하고 모든 층은 편향 뉴런을 포함하며 다음 층과 완전히 연결되어 있음(fully connected neural network).
- 인공 신경망의 은닉층 2개 이상일 때, 이를 **심층신경망**(DNN, Deep Neural Network)이라고 함.



활성화 함수

- 입력층 이후 신경망의 층은 TLU 기반의 퍼셉트론으로서 입력의 가중치 합 $z = w^T x$ 을 계산하고, 그 계산된 값에 계단함수(step function)를 적용하여 그 결과 $h_w(x) = \text{step}(z)$ 를 출력함.
- 위의 계단함수를 **활성화 함수**(activation function)라 하며 비선형 함수임.
- 보다 효율적이고 널리 쓰이는 활성화 함수는 다음과 같음.
  - 로지스틱 함수(logistic function) 또는 시그모이드 함수(sigmoid function):

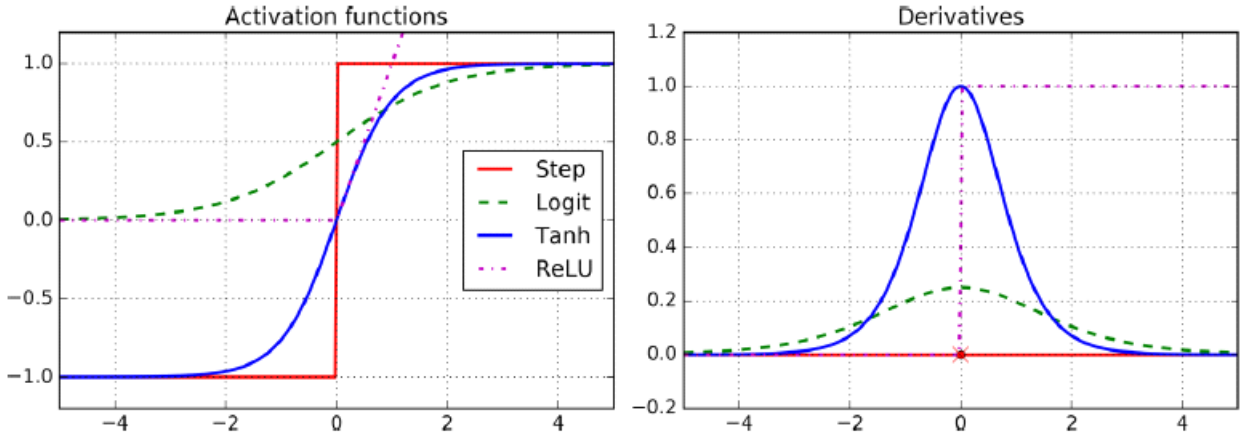
$$\sigma(z) = \frac{1}{1+e^{-z}}$$

  - 쌍곡 탄젠트 함수(hyperbolic tangent function):

$$\sigma_1(z) = \tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} = 2\sigma(2z) - 1$$

- **ReLU 함수 (Rectified Linear Unit):**

$$\sigma_2(z) = \mathrm{ReLU}(z) = \max(0, z)$$



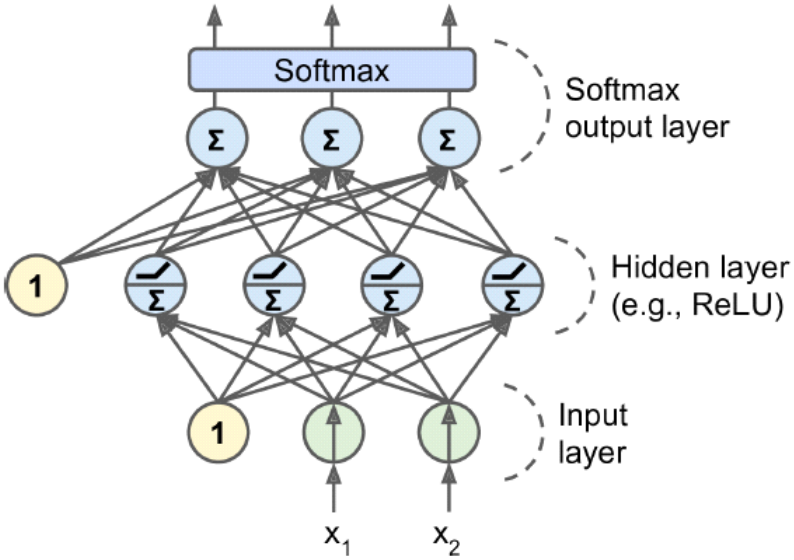
- 다층 퍼셉트론은 각 출력이 서로 다른 이진 클래스에 대응되는 분류 문제에 자주 사용됨.
- 클래스가 배타적일 때는 전형적으로 출력층의 활성화 함수를 소프트맥스(softmax) 함수로 바꿔줌. 이때 각 뉴런의 출력은 각각에 대응되는 클래스의 추정확률임.

- **소프트맥스 함수**:

$$\hat{p}_k = \sigma(\boldsymbol{s}(\boldsymbol{x}))_k := \frac{\exp\left(s_k(\boldsymbol{x})\right)}{\sum_{j=1}^{K}\exp\left(s_j(\boldsymbol{x})\right)}$$

  - ✓ $\boldsymbol{s}(\boldsymbol{x})$는 샘플 $\boldsymbol{x}$에 대한 각 클래스의 점수를 담고 있는 벡터임.
  - ✓ $\sigma(\boldsymbol{s}(\boldsymbol{x}))_k$는 샘플 $\boldsymbol{x}$에 대한 각 클래스의 점수가 주어졌을 때의 이 샘플이 클래스 k에 속할 추정 확률임.

- 이러한 알고리즘은 입력에서 출력으로 한 방향으로만 흐르기 때문에 피드포워드 신경망 (feed forward network, FNN)이라고 함.
- 다음 그림은 분류에 사용되는 (ReLU와 소프트맥스를 포함한) 현대적 다층 퍼셉트론임.



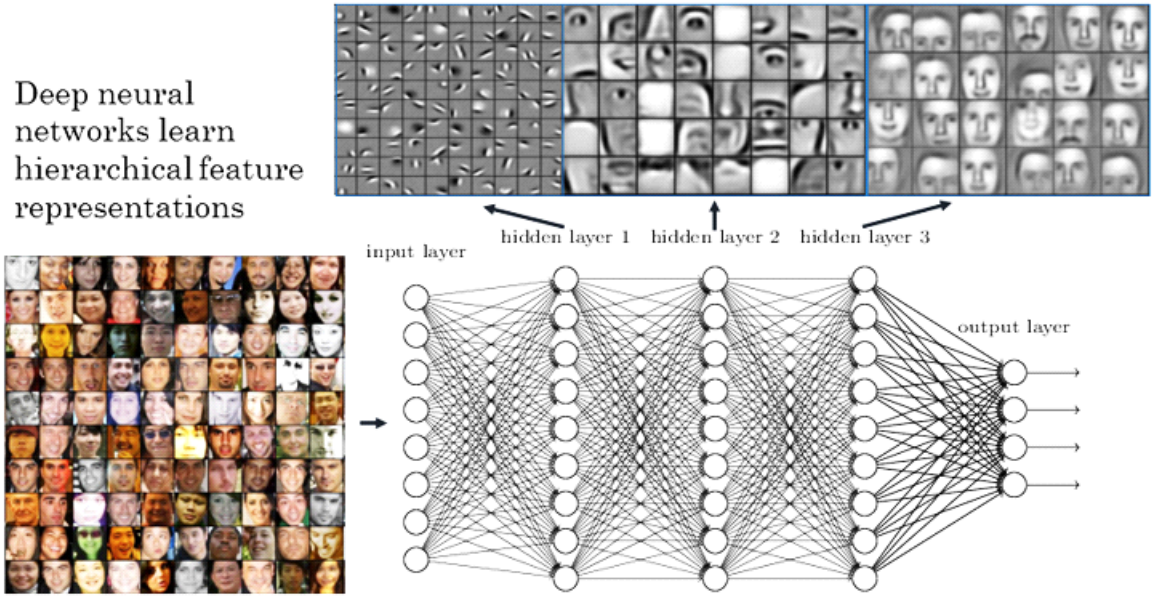
[신경망의 매개변수]

- 신경망의 유연성은 매개변수(하이퍼파라미터)의 자유도에 의존됨.
  - ✓ 은닉층 수
  - ✓ 각 층의 뉴런의 수
  - ✓ 활성화 함수

은닉층 수

- 거의 모든 문제가 은닉층 수를 적게 잡아도 좋은 결과를 얻을 수 있음.
  - ✓ 은닉층 수가 하나인 다층 퍼셉트론이더라도 뉴런수가 충분하다면 아주 복잡한 함수도 모델링 할 수 있음.
- 복잡한 문제의 경우 훈련 세트에 과대 적합이 될 때까지 점진적으로 은닉층을 늘릴 수 있음.
- 많은 은닉층으로 구성된 대규모의 신경망 모델을 딥러닝 모델이라고 함.

- 딥러닝(심층 신경망) 모델은 계층적 구조를 가진 데이터에 효율적임.
  - ✓ 아래쪽 은닉층은 저수준 구조를 모델링하고, 중간 은닉층은 저수준의 구조를 연결하여 중간 수준의 구조를 모델링함. 그리고 가장 위쪽 은닉층은 이런 중간 수준의 구조를 연결하여 고수준의 구조를 모델링함.



은닉층의 뉴런수

- 입력층과 출력층의 뉴런수는 해당 작업에 필요한 입력과 출력의 형태에 따라 결정됨.
  - ✓ 예를 들어 MNIST는 28x28=768개의 입력 뉴런과 10개의 출력 뉴런이 필요함.
- 은닉층의 구성 방식은 일반적으로 각 층의 뉴런을 점차적으로 줄여서 깔때기처럼 구성함.

활성화 함수

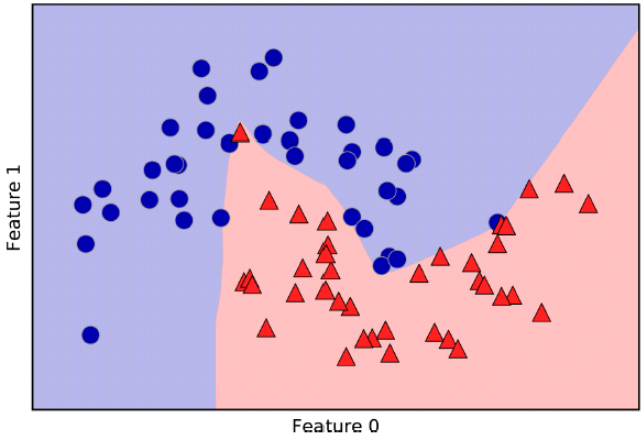
- 대부분의 경우 은닉층에 ReLU 활성화 함수를 사용함.
  - ✓ 이 함수는 다른 활성화 함수 보다 계산이 조금 더 빠르고, 입력값이 클 때 특정 값에 수렴하지 않으므로 경사 하강법이 평편한 지역에서 심하게 지체되지 않는 장점이 있음.
- 분류 작업의 경우 출력층에서는 소프트맥스 활성화 함수가 일반적으로 효과적임.
- 회귀 작업일 경우 활성화 함수를 사용하지 않음.

❖ 신경망 튜닝

- 예시를 위해 'two_moons' 데이터셋에 분류를 위한 다층 퍼셉트론(MLP)을 구현하는 'MLPClassifier'를 적용하고자함.

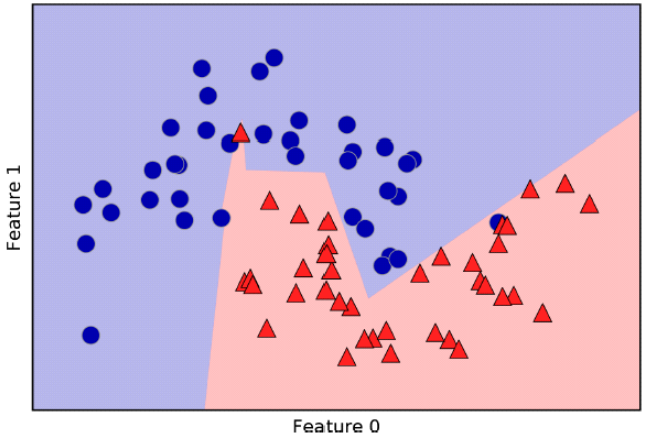
```
from sklearn.neural_network import MLPClassifier
from sklearn.datasets import make_moons

X, y = make_moons(n_samples=100, noise=0.25, random_state=3)

X_train, X_test, y_train, y_test = train_test_split(X, y, stratify=y, random_state=42)

mlp = MLPClassifier(solver='lbfgs', random_state=0).fit(X_train, y_train)
mglearn.plots.plot_2d_separator(mlp, X_train, fill=True, alpha=.3)
mglearn.discrete_scatter(X_train[:, 0], X_train[:, 1], y_train)
plt.xlabel("Feature 0")
plt.ylabel("Feature 1")
```

- ✓ MLP는 기본값으로 은닉층은 1개이며, 은닉 유닛 100개를 사용함.
- ✓ 기본 활성화 함수는 ReLU함수임.



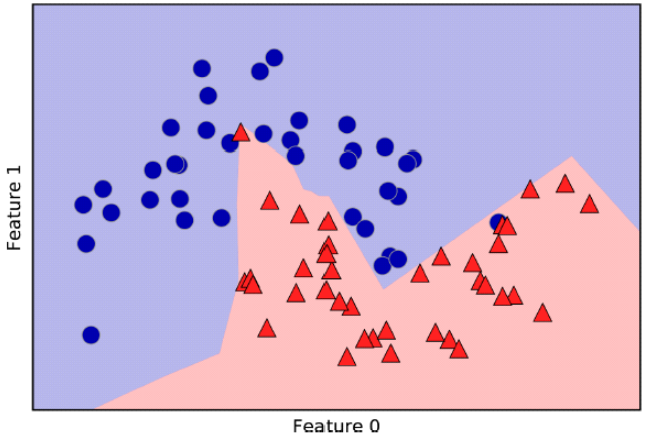
- 다음은 은닉 층 1개와 은닉 유닛의 개수를 10개로 훈련시킨 모델의 결과임.

```
mlp = MLPClassifier(solver='lbfgs', random_state=0, hidden_layer_sizes=[10])
mlp.fit(X_train, y_train)
mglearn.plots.plot_2d_separator(mlp, X_train, fill=True, alpha=.3)
mglearn.discrete_scatter(X_train[:, 0], X_train[:, 1], y_train)
plt.xlabel("Feature 0")
plt.ylabel("Feature 1")
```



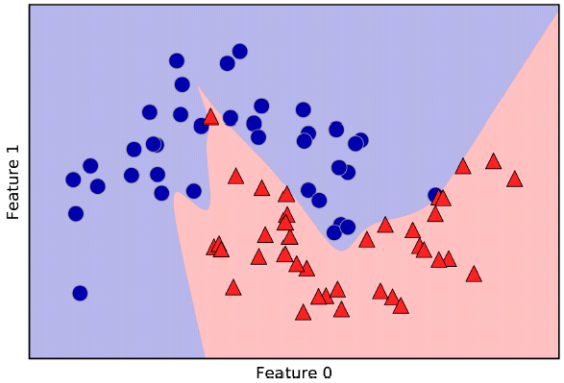
- 다음은 은닉 층 2개와 은닉 유닛의 개수를 10개로 훈련시킨 모델의 결과임.

```
# using two hidden layers, with 10 units each
mlp = MLPClassifier(solver='lbfgs', random_state=0, hidden_layer_sizes=[10, 10])
mlp.fit(X_train, y_train)
mglearn.plots.plot_2d_separator(mlp, X_train, fill=True, alpha=.3)
mglearn.discrete_scatter(X_train[:, 0], X_train[:, 1], y_train)
plt.xlabel("Feature 0")
plt.ylabel("Feature 1")
```



- 다음은 tanh 활성화 함수를 사용하여 은닉 층 2개와 은닉 유닛의 개수를 10개로 훈련시킨 모델의 결과임.

```
# using two hidden layers, with 10 units each, now with tanh nonlinearity
mlp = MLPClassifier(solver='lbfgs', activation='tanh', random_state=0,
hidden_layer_sizes=[10, 10])
mlp.fit(X_train, y_train)
mglearn.plots.plot_2d_separator(mlp, X_train, fill=True, alpha=.3)
mglearn.discrete_scatter(X_train[:, 0], X_train[:, 1], y_train)
plt.xlabel("Feature 0")
plt.ylabel("Feature 1")
```

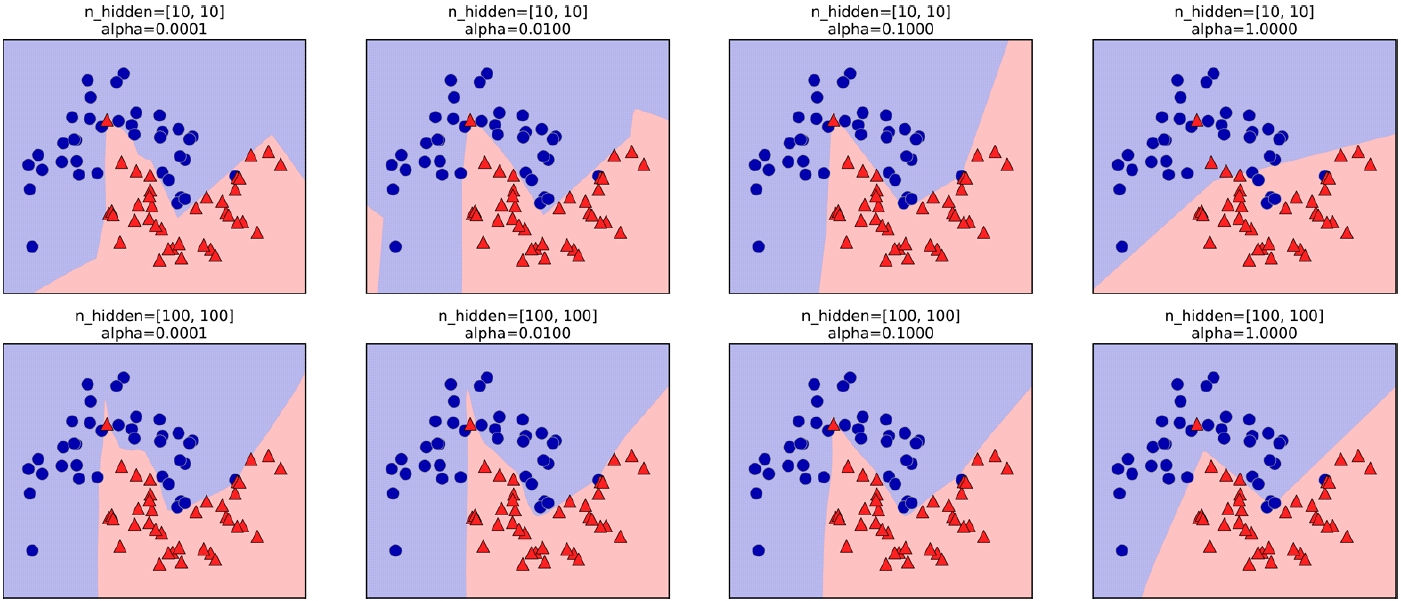


[신경망 모델의 복잡도 제어]

- 릿지 회귀와 선형 분류기에서와 같이 L_2 페널티를 사용하여 가중치 벡터의 크기를 규제함으로써 모델의 복잡도를 제어할 수 있음.
  - ✓ 'MLPClassifier'에서 L_2 페널티를 규제하는 매개변수는 'alpha'임. 기본값은 0.0001임.

- 다음은 'two_moons' 데이터셋에 유닛이 각각 10개와 100개인 은닉층 2개를 사용하였을 때 다양한 'alpha' 값이 미치는 영향을 보여줌.

```
fig, axes = plt.subplots(2, 4, figsize=(20, 8))
for axx, n_hidden_nodes in zip(axes, [10, 100]):
    for ax, alpha in zip(axx, [0.0001, 0.01, 0.1, 1]):
        mlp = MLPClassifier(solver='lbfgs', random_state=0,
                            hidden_layer_sizes=[n_hidden_nodes, n_hidden_nodes], alpha=alpha)
        mlp.fit(X_train, y_train)
        mglearn.plots.plot_2d_separator(mlp, X_train, fill=True, alpha=.3, ax=ax)
        mglearn.discrete_scatter(X_train[:, 0], X_train[:, 1], y_train, ax=ax)
        ax.set_title("n_hidden=[{}, {}]\nalpha={:.4f}".format(
                    n_hidden_nodes, n_hidden_nodes, alpha))
```

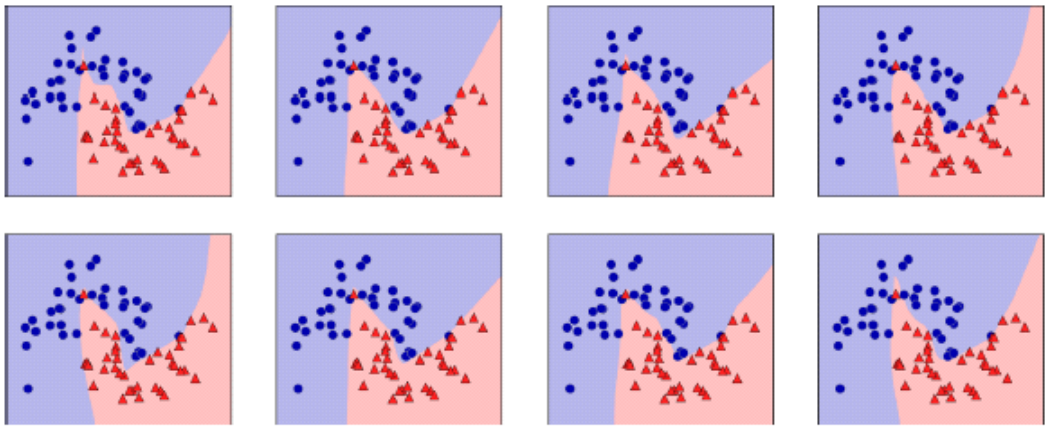


- 신경망의 복잡도를 제어하는 방법:
  - ✓ 은닉층의 수, 은닉층의 유닛 개수, 규제(alpha) 등등.

[가중치의 초기화]

- 신경망은 가중치 초기값을 랜덤하게 취하며, 초기값은 모델의 학습에 영향을 줌.
- 다음은 같은 매개변수를 사용하지만 초기화를 다르게 하여 만든 모델임.

```
fig, axes = plt.subplots(2, 4, figsize=(20, 8))
for i, ax in enumerate(axes.ravel()):
    mlp = MLPClassifier(solver='lbfgs', random_state=i, hidden_layer_sizes=[100, 100])
    mlp.fit(X_train, y_train)
    mglearn.plots.plot_2d_separator(mlp, X_train, fill=True, alpha=.3, ax=ax)
    mglearn.discrete_scatter(X_train[:, 0], X_train[:, 1], y_train, ax=ax)
```



[유방암 데이터셋 예시]

- 다음은 실제 데이터인 유방암 데이터셋에 'MLPClassifier'를 적용하고자 함.

```
print("Cancer data per-feature maxima:\n{}".format(cancer.data.max(axis=0)))
```

```
Cancer data per-feature maxima:

[ 28.11     39.28     188.5     2501.      0.163     0.345     0.427     0.201
   0.304     0.097     2.873     4.885    21.98     542.2     0.031     0.135
   0.396     0.053     0.079     0.03     36.04     49.54    251.2    4254.
   0.223     1.058     1.252     0.291     0.664     0.207]
```

```
X_train, X_test, y_train, y_test = train_test_split(cancer.data, cancer.target, random_state=0)

mlp = MLPClassifier(random_state=42)
mlp.fit(X_train, y_train)

print("Accuracy on training set: {:.2f}".format(mlp.score(X_train, y_train)))
print("Accuracy on test set: {:.2f}".format(mlp.score(X_test, y_test)))
```

```
Accuracy on training set: 0.92
Accuracy on test set: 0.90
```

✓ 신경망도 SVC와 같이 데이터의 스케일이 정확도에 영향을 미침.

- 다음은 모든 입력 특성을 평균 0, 분산 1로 변환한 데이터셋으로 신경망 모델을 훈련한 결과임.

✓ 'StandardScaler' 표준화 등 여러 기법이 있음.

```
# compute the mean value per feature on the training set
mean_on_train = X_train.mean(axis=0)
# compute the standard deviation of each feature on the training set
std_on_train = X_train.std(axis=0)

# subtract the mean, and scale by inverse standard deviation
# afterward, mean=0 and std=1
X_train_scaled = (X_train - mean_on_train) / std_on_train
# use THE SAME transformation (using training mean and std) on the test set
X_test_scaled = (X_test - mean_on_train) / std_on_train
```

```
mlp = MLPClassifier(random_state=0)
mlp.fit(X_train_scaled, y_train)

print("Accuracy on training set: {:.3f}".format(mlp.score(X_train_scaled, y_train)))
print("Accuracy on test set: {:.3f}".format(mlp.score(X_test_scaled, y_test)))
```

```
Accuracy on training set: 0.991
Accuracy on test set: 0.965

ConvergenceWarning:
    Stochastic Optimizer: Maximum iterations reached and the optimization
    hasn't converged yet.
```

✓ 'MLPClassifier' 모델은 Adam(Adaptive Moment Estimation) 경사하강 알고리즘을 사용하며, max_iter의 기본값은 200임.

- 다음은 반복 횟수를 늘려 훈련 세트와 테스트 세트의 성능을 올린 예시임.

```
mlp = MLPClassifier(max_iter=1000, random_state=0)
mlp.fit(X_train_scaled, y_train)

print("Accuracy on training set: {:.3f}".format( mlp.score(X_train_scaled, y_train)))
print("Accuracy on test set: {:.3f}".format(mlp.score(X_test_scaled, y_test)))
```

```
Accuracy on training set: 0.993
Accuracy on test set: 0.972
```

- 일반화의 성능을 올리기 위해 모델의 복잡도를 낮추기.
  - ✓ 매개변수 'alpha'를 0.0001에서 1로 증가시킴.

```
mlp = MLPClassifier(max_iter=1000, alpha=1, random_state=0)
mlp.fit(X_train_scaled, y_train)

print("Accuracy on training set: {:.3f}".format( mlp.score(X_train_scaled, y_train)))
print("Accuracy on test set: {:.3f}".format(mlp.score(X_test_scaled, y_test)))
```

```
Accuracy on training set: 0.993
Accuracy on test set: 0.972
```

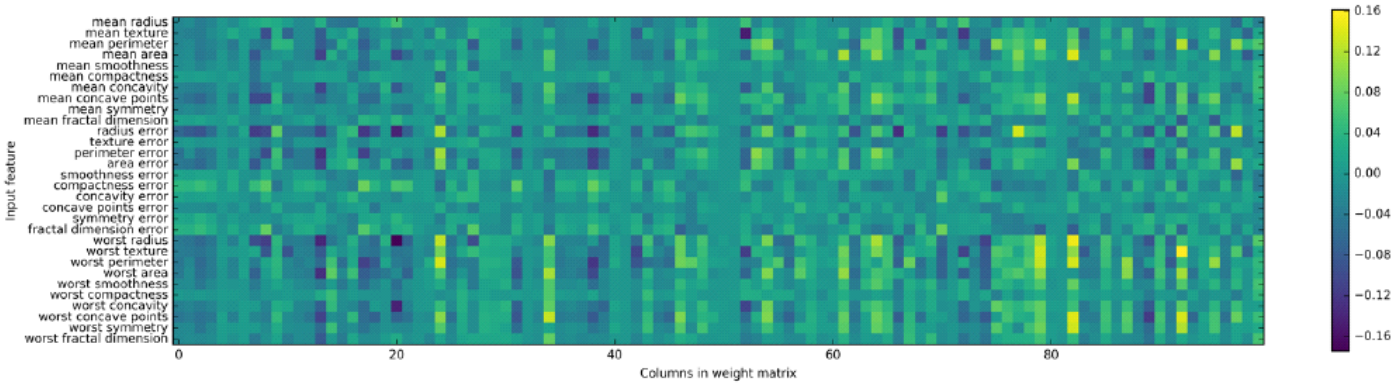
[모델의 가중치 관찰]

- 신경망의 학습 분석은 선형 모델이나 트리 모델보다 명확하지 않음.
- 모델의 가중치를 확인하면 신경망의 학습을 간접적으로 분석할 수 있음.
- 다음은 유방암 데이터셋으로 신경망을 학습시킨 모델의 가중치를 시각화한 것임.

```
plt.figure(figsize=(20, 5))
plt.imshow(mlp.coefs_[0], interpolation='none', cmap='viridis')
plt.yticks(range(30), cancer.feature_names)
plt.xlabel("Columns in weight matrix")
plt.ylabel("Input feature")
plt.colorbar()
```

✓ 아래 그림에서 30개의 입력 특성이 행으로 표시되고 100개의 은닉 유닛이 열로 표시됨.

✓ 여기서는 1개의 은닉층만이 사용됨.



- ✓ 모든 은닉 유닛에서 가장 작은 가중치를 가진 특성은 모델에 덜 중요하다고 추론할 수 있음.
- ✓ 위의 행렬 그림에서 “smoothness error”와 “fractal dimension error” 사이에 있는 특성과 “mean smoothness”와 “mean compactness”가 다른 특성들 보다 가중치가 낮음.

- ‘MLPClassifier’와 ‘MLPRegressor’는 일반적인 신경망을 제공하지만 보다 복잡하고 대규모의 모델 (CNN, RNN 등)을 만들려면 전문적인 딥러닝 라이브러리를 활용해야 함.
  - ✓ 파이썬 기반의 라이브러리는 keras, lasagna, tensorflow 등이 널리 사용됨.

❖ 장단점과 매개변수

- 대량의 데이터에 내재된 정보를 얻을 수 있으며, 매우 복잡한 모델을 만들 수 있음.
- 충분한 연산 시간과 데이터를 주고 매개변수를 세심하게 조정하면, 신경망 모델은 다른 머신러닝 알고리즘보다 아주 우수한 성능을 낼 수 있음.
- 신경망은 종종 학습이 오래 걸림
- 데이터 전처리를 한 후 학습시켜야함.
  - ✓ 특성들의 스케일이 비슷해야 좋은 성능을 냄.
- 신경망 매개변수 튜닝은 매우 세심하게 이루어져야함.

❖ 신경망의 복잡도 추정

- 신경망에서 가장 중요한 매개변수는 은닉층의 개수와 은닉층의 유닛수임.
  - ✓ 처음에는 한 개 또는 두 개의 은닉층으로 시작하여 점차 늘려감.
  - ✓ 각 은닉층의 유닛 수는 보통 입력 특성의 수와 비슷하게 설정함.

- 신경망 모델의 복잡도는 학습된 가중치(계수)의 개수로 측정할 수 있음.
- 특성 100개와 은닉 유닛 100개를 가진 이진 분류기의 경우:
  - ✓ 입력층과 첫 번째 은닉층 사이에는 편향을 포함해 100x100+100 = 10,100개의 가중치가 있음.
  - ✓ 은닉층과 출력층 사이에는 101개의 가중치가 있음.
  - ✓ 따라서 이 경우 총 10,201개의 가중치가 있음
  - ✓ 은닉층을 하나 추가하면, 20,301개의 가중치가 있음.
- 특성 100개와 은닉 유닛 1000개를 가진 이진 분류기의 경우:
  - ✓ 가중치가 총 1,103,001개로 늘어남.

- 신경망의 매개변수를 조정하는 일반적인 방법은 먼저 충분히 큰 과대적합 모델을 만들고, 그 다음 훈련 데이터가 충분히 학습되었을 때, 신경망 구조를 줄이거나 규제를 강화하여 일반화 성능을 향상시킴.