머신러닝
(Machine Learning)

LECTURE XII: 지도 학습 6
(Supervised Learning)

Dai-Gyoung Kim

Department of Applied Mathematics

Hanyang University ERICA

지도 학습 (Supervised Learning)

Contents

2) 로지스틱 회귀

- **로지스틱 회귀**(Logistic Regression) 또는 로짓 회귀(Logit Regression)는 특정 클래스에 속할 확률을 추정하는 데 사용되는 이진 분류기라 할 수 있음.

[확률 추정]

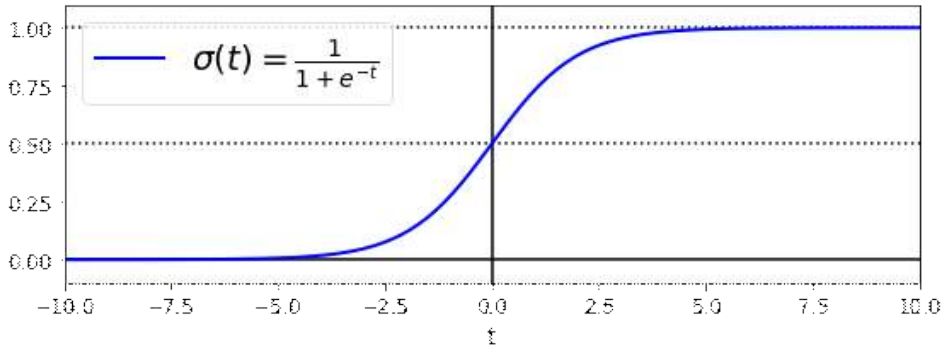
- 선형 회귀처럼 바로 결과를 출력하는 것이 아니라, 결괏값의 **로지스틱**을 출력함.

$$p = h_{\boldsymbol{w},b}(\boldsymbol{x}) = \sigma\left(\boldsymbol{w}^T\boldsymbol{x} + b\right)$$

✓ 로지스틱은 로짓(logit)이라고도 부르며 $\sigma(\cdot)$로 표시하고, 0과 1사이의 값을 출력하는 **시그모이드 함수**(sigmoid function)임.

- 시그모이드 또는 로지스틱 함수

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



- 로지스틱 회귀 모델은 샘플 x가 양성 클래스에 속할 확률 $p = h_{\boldsymbol{w}, b}(\boldsymbol{x})$를 추정하여 이에 대한 예측 $\hat{y}$을 쉽게 구할 수 있음.

$$\hat{y} = \begin{cases} 0 & \text{if } \hat{p} < 0.5 \\ 1 & \text{if } \hat{p} > 0.5 \end{cases}$$

✓ $t = \boldsymbol{w}^T\boldsymbol{x} + b > 0$이면, 1(양성 클래스: $+1$)로 예측하고,

✓ $t = \boldsymbol{w}^T\boldsymbol{x} + b < 0$이면, 0(음성 클래스: -1)으로 예측함.

[훈련과 비용함수]

- 훈련의 목적은 양성 샘플($y=1$)에 대해서는 높은 확률을 추정하고, 음성 샘플($y=0$)에 대해서는 낮은 확률을 추정하는 모델 파라미터 w, b를 찾는 것임.
- 위의 목적을 달성하기 위한 비용 함수는 다음과 같음.

$$c(\boldsymbol{w}, b) = \begin{cases} -\log(p) & \text{if } y=1 \\ -\log(1-p) & \text{if } y=0 \end{cases}$$

✓ **양성 훈련 샘플**을 훈련할 경우:

1) 0에 가까운 확률($p \approx 0$)로 추정하면, $c(\boldsymbol{w}, b) = -\log(p)$가 크게 증가함.

2) 1에 가까운 확률($p \approx 1$)로 추정하면, $c(\boldsymbol{w}, b) = -\log(p)$가 0에 가까워짐.

✓ **음성 훈련 샘플**을 훈련할 경우:

1) 1에 가까운 확률($p \approx 1$)로 추정하면, $c(\boldsymbol{w}, b) = -\log(1-p)$가 크게 증가함.

2) 0에 가까운 확률($p \approx 0$)로 추정하면, $c(\boldsymbol{w}, b) = -\log(1-p)$가 0에 가까워짐.

- 전체 훈련 세트에 대한 비용 함수는 모든 훈련 샘플의 비용을 평균한 것임. 이를 **로지스틱 손실**(logistic loss)이라 부름.

$$L(\boldsymbol{w},b) = -\frac{1}{m}\sum_{i=1}^{m}\left[y_i \log\left(p_i\right) + \left(1-y_i\right)\log\left(1-p_i\right)\right]$$

$$= \frac{1}{m}\sum_{i=1}^{m}\log\left(1+e^{(-1)^{(1-y_i)}\left(\boldsymbol{w}^T\boldsymbol{x}_i+b\right)}\right)$$

▷ $y_i \in \{0,1\}$

▷ $p_i = \dfrac{1}{1+\exp\left(-\left(\boldsymbol{w}^T\boldsymbol{x}_i+b\right)\right)}$

✓ 이 손실 함수의 최솟값을 계산하는 알려진 공식(해)은 없음.

✓ 이 손실 함수가 w, b에 관하여 볼록 함수이기 때문에 경사 하강법을 사용하여 최솟값을 찾을 수 있음. (Hessian Matrix of L is symmetric positive semidefinite!)

- 로지스틱 회귀 손실 함수의 그래디언트:

$$\nabla_{\boldsymbol{w},b} L = \left(\frac{\partial L}{\partial w_0}, \ \cdots \ , \frac{\partial L}{\partial w_p}, \frac{\partial L}{\partial b} \right)^T$$

▷ $\dfrac{\partial L}{\partial w_j}(\boldsymbol{w}, b) = \dfrac{1}{m} \sum_{i=1}^{m} [p_i - y_i] x_{ij}$

▷ $\dfrac{\partial L}{\partial b}(\boldsymbol{w}, b) = \dfrac{1}{m} \sum_{i=1}^{m} [p_i - y_i]$

✓ 각 i번째 샘플에 대해 예측 오차를 계산하고 j번째 특성 값을 곱해서 모든 훈련 샘플에 대해 평균을 냄.

- 경사 하강법:

$$\begin{pmatrix} \boldsymbol{w}^{(new)} \\ b^{(new)} \end{pmatrix} = \begin{pmatrix} \boldsymbol{w} \\ b \end{pmatrix} - \eta \nabla_{\boldsymbol{w},b} L(\boldsymbol{w}, b)$$

규제 모델

[로지스틱 회귀의 ℓ_2 규제 모델]

$$J(\theta) = \frac{1}{2}\theta^T\theta + CL(\theta)$$

▷ $\theta = \begin{pmatrix} w \\ b \end{pmatrix}$

▷ $C > 0$: 규제에 관한 하이퍼파라미터 ($C = 1/\alpha$)

[SVM 분류의 ℓ_2 규제 모델]

$$J(w, b) = \frac{1}{2}w^T w + C\sum_{i=1}^{m}\max\left(0, 1 - \left(w^T x_i + b\right)t_i\right)$$

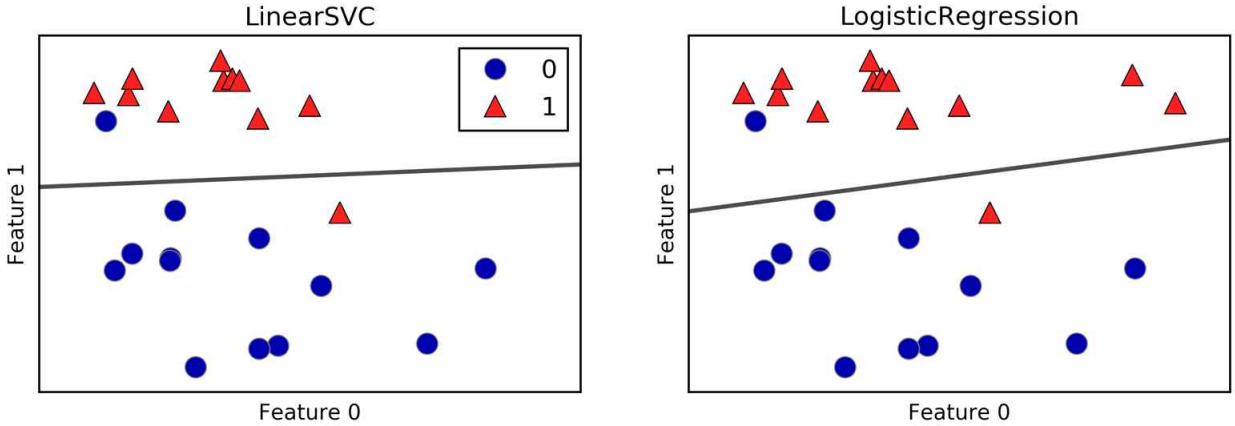
✓ $C \gg 1$: 작은 제약, 과대 적합,

✓ $C \approx 0$: 큰 제약, 과소 적합($\| w \| \approx 0$)

선형 분류 모델 적용하기

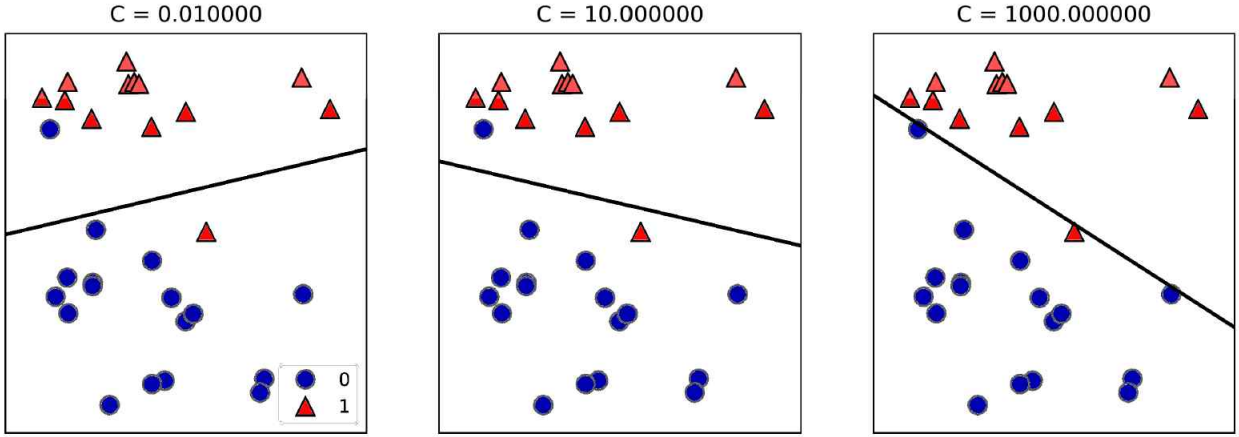
- 다음은 forge 데이터셋에 “LogisticRegression”과 “LinearSVC” 모델 적용하여 선형 모델들을 만들고 그 결정 경계를 보여주는 코드임.

```
from sklearn.linear_model import LogisticRegression
from sklearn.svm import LinearSVC

X, y = mglearn.datasets.make_forge()

fig, axes = plt.subplots(1, 2, figsize=(10, 3))

for model, ax in zip([LinearSVC(), LogisticRegression()], axes):
    clf = model.fit(X, y)
    mglearn.plots.plot_2d_separator(clf, X, fill=False, eps=0.5, ax=ax, alpha=.7)
    mglearn.discrete_scatter(X[:, 0], X[:, 1], y, ax=ax)
    ax.set_title("{}".format(clf.__class__.__name__))
    ax.set_xlabel("Feature 0")
    ax.set_ylabel("Feature 1")
axes[0].legend()
```



✓ 위 코드는 ℓ_2규제 ($C = 1$)를 기본값으로 함.

- LinearSVC 하이퍼파라미터 C의 효과:
  - ✓ C 값을 낮추면 모델이 데이터 포인트 중 다수에 맞추려는 경향이 높음.
  - ✓ C 값을 높이면 모델이 개개의 데이터 포인트를 정확히 분류하려는 경향이 높음.



- 다음은 “LogisticRegression” 모델을 유방암 데이터셋에 적용하는 코드임.

```
from sklearn.datasets import load_breast_cancer

cancer = load_breast_cancer()
X_train, X_test, y_train, y_test = train_test_split(
    cancer.data, cancer.target, stratify=cancer.target, random_state=42)
logreg = LogisticRegression().fit(X_train, y_train)

print("Training set score: {:.3f}".format(logreg.score(X_train, y_train)))
print("Test set score: {:.3f} ".format(logreg.score(X_test, y_test)))
```

```
Training set score: 0.953
Test set score: 0.958
```

✓ 기본값 $C=1$인 경우로서, 훈련 세트와 테스트 세트 모두 95% 정확도를 구현하여 아주 좋은 성능을 보여주고 있음.

✓ 훈련 세트와 테스트 세트의 성능이 비슷하므로 과소적합일 가능성이 높음(규제가 큼).

```
logreg100 = LogisticRegression(C=100).fit(X_train, y_train)

print("Training set score: {:.3f}".format(logreg100.score(X_train, y_train)))
print("Test set score: {:.3f}".format(logreg100.score(X_test, y_test)))
```

```
Training set score: 0.972
Test set score: 0.965
```

✓ $C = 100$인 경우로서, 훈련 세트와 테스트 세트 모두 정확도가 증가했음.

✓ 이 모델은 복잡도가 높을수록 성능이 좋아짐을 알 수 있음.

```
logreg001 = LogisticRegression(C=0.01).fit(X_train, y_train)

print("Training set score: {:.3f}".format(logreg001.score(X_train, y_train)))
print("Test set score: {:.3f}".format(logreg001.score(X_test, y_test)))
```

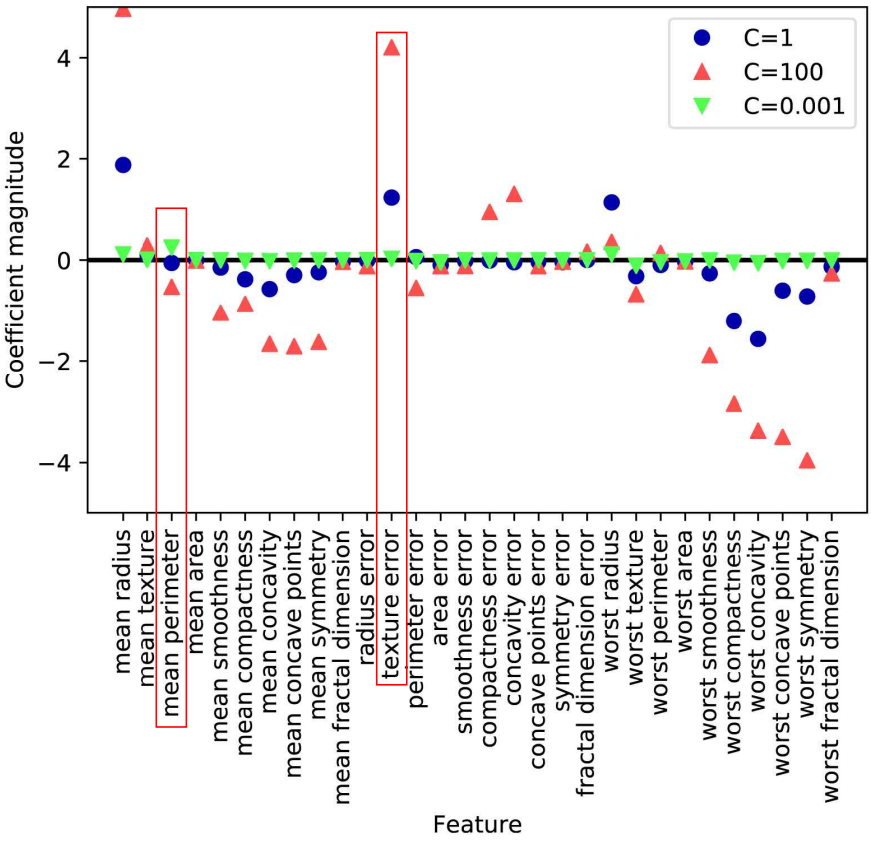
```
Training set score: 0.934
Test set score: 0.930
```

✓ 이 경우는 $C = 0.01$이며 규제가 강하게 이루어져 과소적합된 모델임.

- 다음은 유방암 데이터셋에 각기 다른 ℓ_2 규제 하이퍼파라미터 C 값을 사용하여 학습시킨 로지스틱 회귀 모델의 계수의 분포를 보여줌.

```
plt.plot(logreg.coef_.T, 'o', label="C=1")
plt.plot(logreg100.coef_.T, '^', label="C=100")
plt.plot(logreg001.coef_.T, 'v', label="C=0.001")
plt.xticks(range(cancer.data.shape[1]), cancer.feature_names, rotation=90)
plt.hlines(0, 0, cancer.data.shape[1])
plt.ylim(-5, 5)
plt.xlabel("Feature")
plt.ylabel("Coefficient magnitude")
plt.legend()
```

✓ C 값의 변화에 따른 계수의 특성을 분석함으로써, 유방암에 관련된 특성을 해석할 수 있음. 예를 들어, “texture error” 특성은 악성 샘플과 관련이 깊어 보이고, “mean perimeter” 계수의 부호가 바뀜에 따라 높은 “mean perimeter” 값은 양성이나 악성의 신호 모두가 될 수 있음.

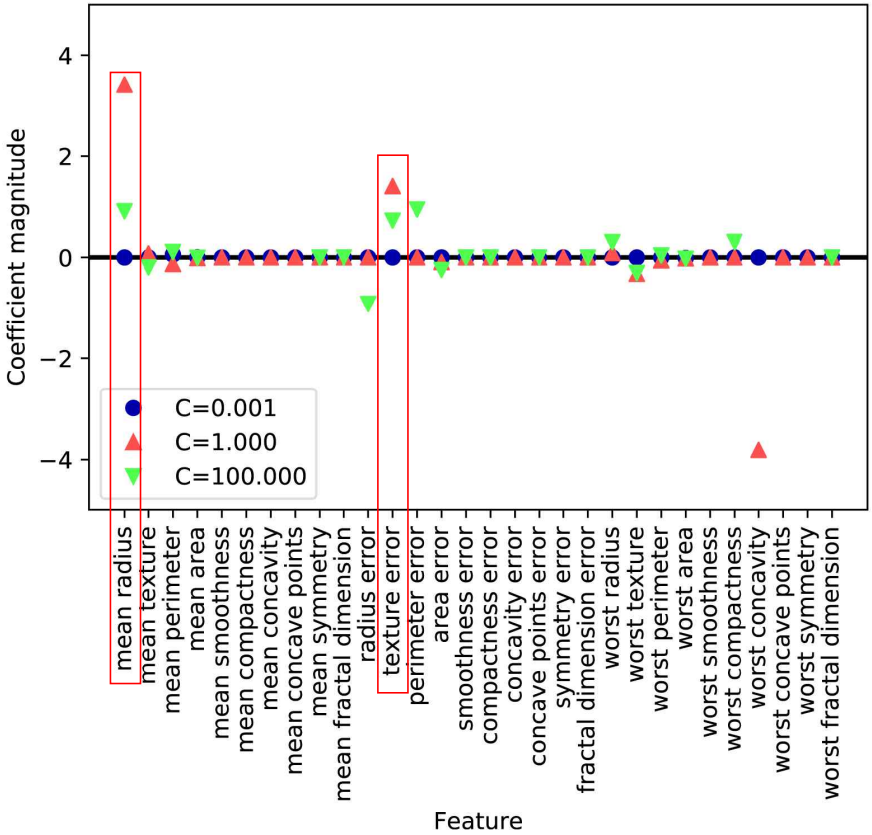


- ℓ_1 규제를 사용하면 보다 더 이해하기 쉬운 모델을 얻을 수 있음.
- 다음은 ℓ_1 규제를 사용할 때의 분류 정확도와 계수의 그래프를 보여줌.

```
for C, marker in zip([0.001, 1, 100], ['o', '^', 'v']):
    lr_l1 = LogisticRegression(C=C, penalty="l1").fit(X_train, y_train)
    print("Training accuracy of l1 logreg with C={:.3f} : {:.2f}".format(
          C, lr_l1.score(X_train, y_train)))
    print("Test accuracy of l1 logreg with C={:.3f} : {:.2f}".format(
          C, lr_l1.score(X_test, y_test)))
    plt.plot(lr_l1.coef_.T, marker, label="C= {:.3f}".format(C))

plt.xticks(range(cancer.data.shape[1]), cancer.feature_names, rotation=90)
plt.hlines(0, 0, cancer.data.shape[1])
plt.xlabel("Feature")
plt.ylabel("Coefficient magnitude")

plt.ylim(-5, 5)
plt.legend(loc=3)
```

```
Training accuracy of l1 logreg with C=0.001: 0.91
Test accuracy of l1 logreg with C=0.001: 0.92
Training accuracy of l1 logreg with C=1.000: 0.96
Test accuracy of l1 logreg with C=1.000: 0.96
Training accuracy of l1 logreg with C=100.000: 0.99
Test accuracy of l1 logreg with C=100.000: 0.98
```

- ✓ 이 경우 약한 규제 $C = 100$일 때의 ℓ_1 규제 모델이 아주 좋은 성능을 보여줌.
- ✓ ℓ_1 규제 모델의 특징은 ℓ_2 규제 모델과 달리 모든 특성을 사용하지 않고 일부 특성만을 사용하여 좋은 성능을 발휘한다는 것임.
- ✓ 다음은 각 특성의 계수에 관한 그래프에서 전반적으로 모델들이 많은 특성을 사용하지 않고 몇 개의 특성만으로 유방암의 양성, 악성을 분류함을 볼 수 있음.



❖ 다중 클래스 분류용 선형 모델

둘 이상의 클래스를 구분하는 분류기를 **다중 클래스 분류기**(multiclass classifier) 또는 **다항 분류기**(multinomial classifier)라 함.

- 여러 개의 클래스를 직접처리 할 수 있는 분류기
  - ✓ 로지스틱 회귀
  - ✓ 랜덤 포레스트 분류기(Random Forest),
  - ✓ 나이브베이즈 분류기(Naive Bayes)
- 이진 분류기를 여러 개 사용해서 다중 클래스를 처리할 수 있는 분류기
  - ✓ 서포트 벡터 머신 분류기(Support Vector Machine), 선형 분류기

- 로지스틱 회귀를 제외하고 많은 선형 분류 모델은 태생적으로 이진 분류만을 지원함.
- 이진 분류 알고리즘을 다중 클래스 분류 알고리즘으로 확장하는 보편적인 방법은 일대다(one-vs-rest) 기법임.

이진 분류기 기반의 다중 클래스 분류 기법

- **일대다**(또는 one-vs-rest, OvR 또는 one-vs-all, OvA)
  - ✓ 각 클래스를 다른 모든 클래스와 구분하도록 이진 분류 모델을 학습시켜, 클래스 수 만큼 이진 분류 모델을 만듦. 예측을 할 때 각 이진 분류기의 결정 점수 중에서 가장 높은 것을 클래스 예측값으로 선택함.
- **일대일**(one-vs-one, OvO)
  - ✓ 예를 들어 0과 1 구별, 0과 2구별, ... , 0과 9구별, 1과 2구별, ... , 8과 9 구별 같이 각 숫자의 조합마다 이진 분류기를 훈련시킴.
  - ✓ 클래스가 N개라면, 분류기는 N조합 N(N-1)/2개가 필요함. MNIST 문제의 경우 45개의 분류기를 훈련 시켜야 함. 따라서 이미지 하나를 분류하려면, 45개 분류기 모두를 통과시켜 가장 많이 양성으로 분류된 클래스를 선택함.
- 다중 클래스 분류에서는 클래스 마다 가중치와 절편을 만들어짐.

[다중 클래스 로지스틱 회귀]

- 로지스틱 회귀 모델은 여러 개의 이진 분류기를 훈련시켜 연결하지 않고 직접 다중 클래스를 다룰 수 있도록 일반화할 수 있는데, 이를 **소프트맥스 회귀**(Softmax Regression) 또는 **다중 클래스 로지스틱 회귀**(Multiclass Logistic Regression)라고 함.
- 소프트맥스 회귀의 개념은 다음과 같음.
  - ✓ 샘플 x가 주어지면 먼저 소프트맥스 회귀 모델이 각 클래스 k에 대한 점수 $s_k(x)$를 계산하고, 그 점수에 **소프트맥스 함수**(softmax function) 또는 **정규화된 지수 함수**(normalized exponential)를 적용하여 각 클래스의 확률을 추정함.

- 각 클래스에 대한 **소프트맥스 점수**:

$$s_k(\boldsymbol{x}) = \boldsymbol{w}_k^T \boldsymbol{x} + b_k$$

✓ 샘플 x에 대해 각 클래스의 소프트맥스 점수가 계산되면 다음의 소프트맥스 함수를 통과시켜 클래스 k에 속할 확률 $\hat{p}_k$를 추정함.

- **소프트맥스 함수**:

$$\hat{p}_k = \Pr(Y = k) := \frac{e^{s_k(\boldsymbol{x})}}{\sum_{j=1}^{K} e^{s_j(\boldsymbol{x})}}$$

- **소프트맥스 회귀 분류기의 예측**:

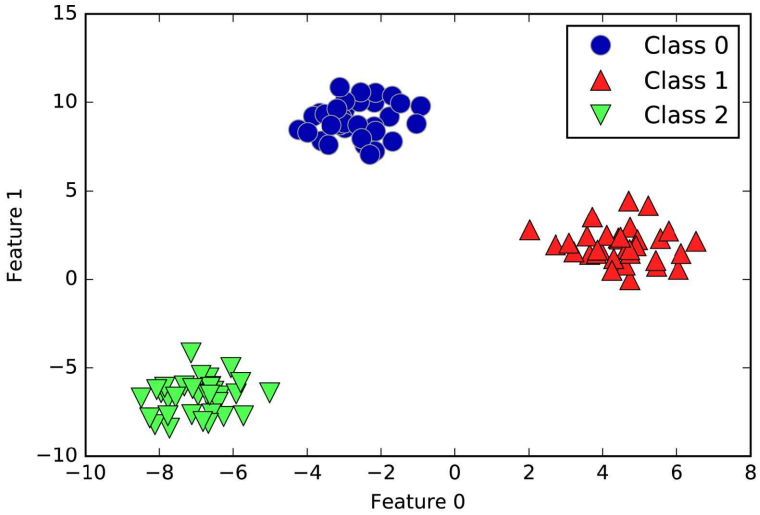
✓ 로지스틱 회귀 분류기와 마찬가지로 소프트맥스 회귀 분류기는 추정 확률이 가장 높은 클래스를 선택함.

$$\hat{y} = \underset{k}{\operatorname{argmax}} \sigma\left(\hat{p}_k\right) = \underset{k}{\operatorname{argmax}} \left(\boldsymbol{w}_k^T \boldsymbol{x} + b_k\right)$$

✓ 소프트맥스 회귀 분류기는 한 번에 하나의 클래스만 예측함 (다중 클래스이지만 다중 출력은 아님).

- 다음은 세 개의 클래스를 가진 간단한 데이터셋에 일대다(OvA) 방식을 적용하는 예시임. 이 데이터셋은 2차원이며 각 클래스의 데이터는 정규분포를 따름.

```
from sklearn.datasets import make_blobs

X, y = make_blobs(random_state=42)
mglearn.discrete_scatter(X[:, 0], X[:, 1], y)
plt.xlabel("Feature 0")
plt.ylabel("Feature 1")
plt.legend(["Class 0", "Class 1", "Class 2"])
```



- 이 데이터셋으로 “LinearSVC” 분류기를 훈련하기.

```
linear_svm = LinearSVC().fit(X, y)
print("Coefficient shape: ", linear_svm.coef_.shape)
print("Intercept shape: ", linear_svm.intercept_.shape)
```

```
Coefficient shape:  (3, 2)
Intercept shape:  (3,)
```

✓ “coef_” 배열의 크기는 3x2 행렬임. 행은 세 개의 클래스에 각각 대응하는 계수벡터를 담고 있으며, 열은 각 특성(특성0, 특성1)에 따른 계수 값을 가지고 있음.

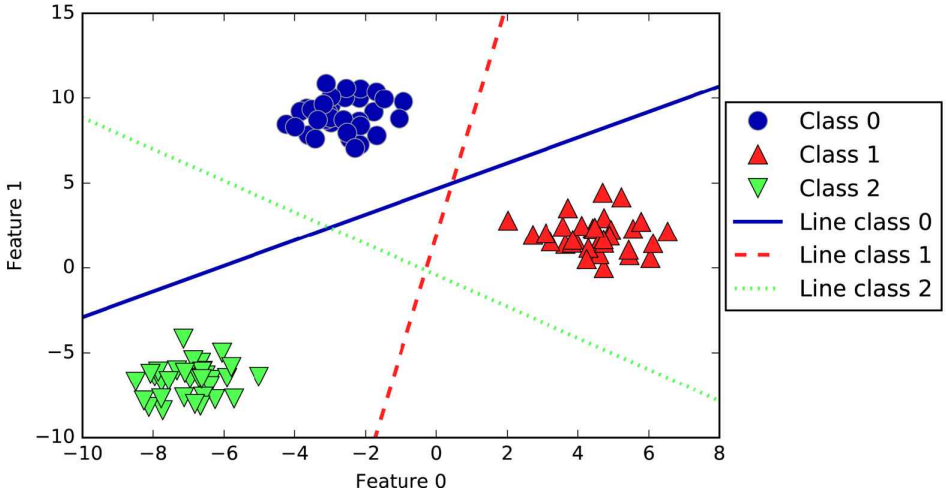
✓ “intercept_”는 각 클래스의 절편을 담은 1차원 벡터임.

- 다음은 세 개의 이진 분류기가 만드는 경계를 시각화를 보여주는 코드임.

```
mglearn.discrete_scatter(X[:, 0], X[:, 1], y)
line = np.linspace(-15, 15)

for coef, intercept, color in zip(linear_svm.coef_, linear_svm.intercept_,
                                  mglearn.cm3.colors):
```

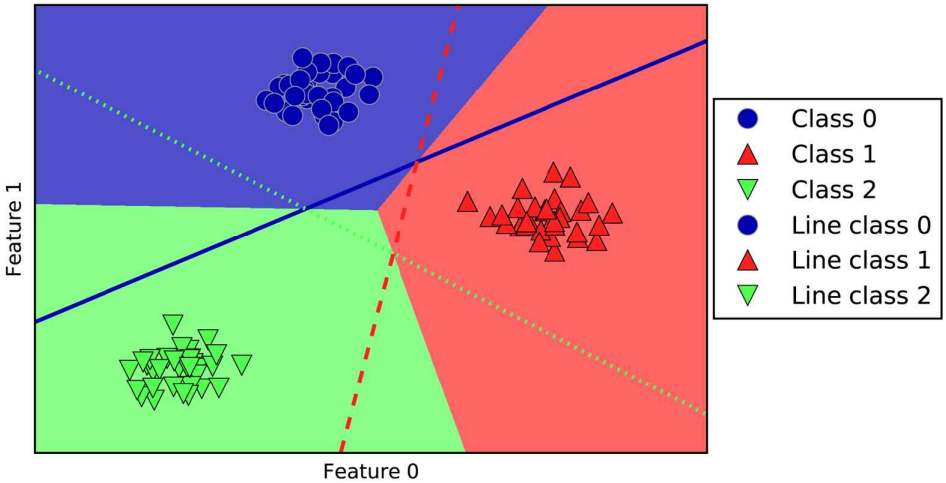
```
    plt.plot(line, -(line * coef[0] + intercept) / coef[1], c=color)

plt.ylim(-10, 15)
plt.xlim(-10, 8)
plt.xlabel("Feature 0")
plt.ylabel("Feature 1")
plt.legend(['Class 0', 'Class 1', 'Class 2', 'Line class 0', 'Line class 1',
            'Line class 2'], loc=(1.01, 0.3))
```

- ✓ 세 개의 일대다 분류기가 만든 결정 경계
- ✓ 공통부분에 분포되어 있는 데이터 포인트는 가장 가까운 직선의 클래스로 분류됨.



- 다음 예는 2차원 평면의 모든 포인트에 대한 예측 결과를 보여줌.
  - ✓ 세 개의 일대다 분류기가 만든 다중 클래스 결정경계를 보여줌.

```
mglearn.plots.plot_2d_classification(linear_svm, X, fill=True, alpha=.7)
mglearn.discrete_scatter(X[:, 0], X[:, 1], y)
line = np.linspace(-15, 15)

for coef, intercept, color in zip(linear_svm.coef_, linear_svm.intercept_,
                                  mglearn.cm3.colors):
    plt.plot(line, -(line * coef[0] + intercept) / coef[1], c=color)

plt.legend(['Class 0', 'Class 1', 'Class 2', 'Line class 0', 'Line class 1',
            'Line class 2'], loc=(1.01, 0.3))
plt.xlabel("Feature 0")
plt.ylabel("Feature 1")
```



❖ 선형 모델의 장단점과 매개변수

매개변수의 선택

- 선형 모델의 주요 매개 변수

  1) 회귀 모델: α

  2) 분류 모델: C (LinearSVC, LogisticRegression)

- α 값이 클수록 또는 C 값이 작을수록, 모델이 단순해짐 (과소적합 경향).
- α 값이 작을수록 또는 C 값이 클수록, 모델이 복잡해짐 (과대적합 경향).
  - ✓ 선형모델에서 위의 매개변수(하이퍼파라미터)를 미세 조정하는 일이 중요함.

규제를 가하는 방식

- ℓ_1 규제와 ℓ_2 규제의 선택의 기준은 다음과 같음.

  1) ℓ_2 규제: 범용적인 선택.

  2) ℓ_1 규제: 중요한 특성이 많지 않거나 모델의 해석이 중요할 때.

✓ ℓ_1 규제는 몇 가지 특성만을 사용하므로 해당 모델에 중요한 특성과 그 효과가 어느 정도 인지를 설명하기 쉬움.

3) 엘라스틱넷 규제: 릿지 회귀와 라쏘 회귀를 절충한 모델임.

선형 모델의 장단점

- 선형 모델의 장점

1) 학습 속도가 빠르고 예측도 빠름.

2) 매우 큰 데이터셋과 희소한 데이터셋에 잘 작동함.

✓ 선형 모델의 대용량 처리 버전으로 구현된 “SGDRegressor”와 “SGDClassifier”를 사용하면 보다 빠르게 처리할 수 있음.

3) 회귀 및 분류의 수학적 공식이 명확하여 예측이 어떻게 구현되는지 쉽게 이해할 수 있음.

4) 선형 모델은 특히 샘플에 비해 특성이 많을 때 잘 작동함.

✓ 다른 모델로 학습하기 어려운 매우 큰 데이터셋에서도 선형 모델을 많이 사용함.

5) 비선형 모델로 확장하는데 사용됨.

- 선형 모델의 단점

  1) 저차원의 데이터셋에서는 다른 비선형 모델들 보다 일반화 성능이 떨어짐.

  2) 비선형 데이터에서 잘 작동하지 않음.