

선형 모델 Linear Model

파이썬 라이브러리를 이용한 머신러닝







휴먼프밍 발표

황서현 Hwang seo hyeon

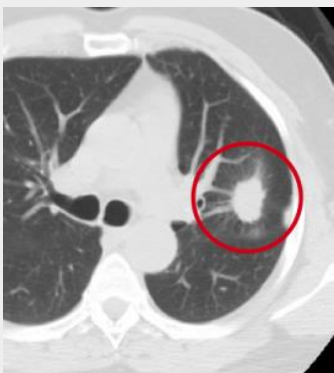
휴먼기계바이오공학



목차 INDEX

	회귀를 위한 선형 모델
	선형 회귀(OLS)
	리지 회귀
	라쏘 회귀
	분류를 위한 선형 모델
	다중 클래스 선형 모델

Wave



환자	암 크기	암일 확률
INDEX	Feature(X)	Label(Y)
1	0.25	0.21
2	0.70	0.99
3	0.67	0.78
4	0.35	0.53

Make

환자	암 크기	표면모양	암 여부
INDEX	Feature1 (X1)	Feature2 (X2)	Label(Y)
1	0.25	0.53	0
2	0.70	0.77	1
3	0.67	0.11	1
4	0.35	0.52	1

환자 암크기 표면모양 장기와 거리 암일 확률

INDEX	Feature1 (X1)	Feature2 (X2)	Feature3 (X3)	Label(Y)
1	0.25	0.53	0.65	0.21
2	0.70	0.77	0.02	0.99
3	0.67	0.11	0.31	0.78
4	0.35	0.52	0.23	0.53

환자 암크기 표면모양 장기와 거리 암 여부

INDEX	Feature1 (X1)	Feature2 (X2)	Feature3 (X3)	Label(Y)
1	0.25	0.53	0.65	0
2	0.70	0.77	0.02	1
3	0.67	0.11	0.31	1
4	0.35	0.52	0.23	1

$$\hat{y} = w[0] * x[0] + w[1] * x[1] + \dots + w[p] * x[p] + b$$

$$h_{\theta}(x) = \theta_0 x_0 + \theta_1 x_1 + \dots + \theta_n x_n$$

OLS = Ordinary Least Square 잔차 제곱 평균 합이 최소가 되게 하라

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^m \left(\hat{y}^{(i)} - y^{(i)} \right)^2 = \frac{1}{2m} \sum_{i=1}^m \left(h_{\theta}(x^{(i)}) - y^{(i)} \right)^2$$

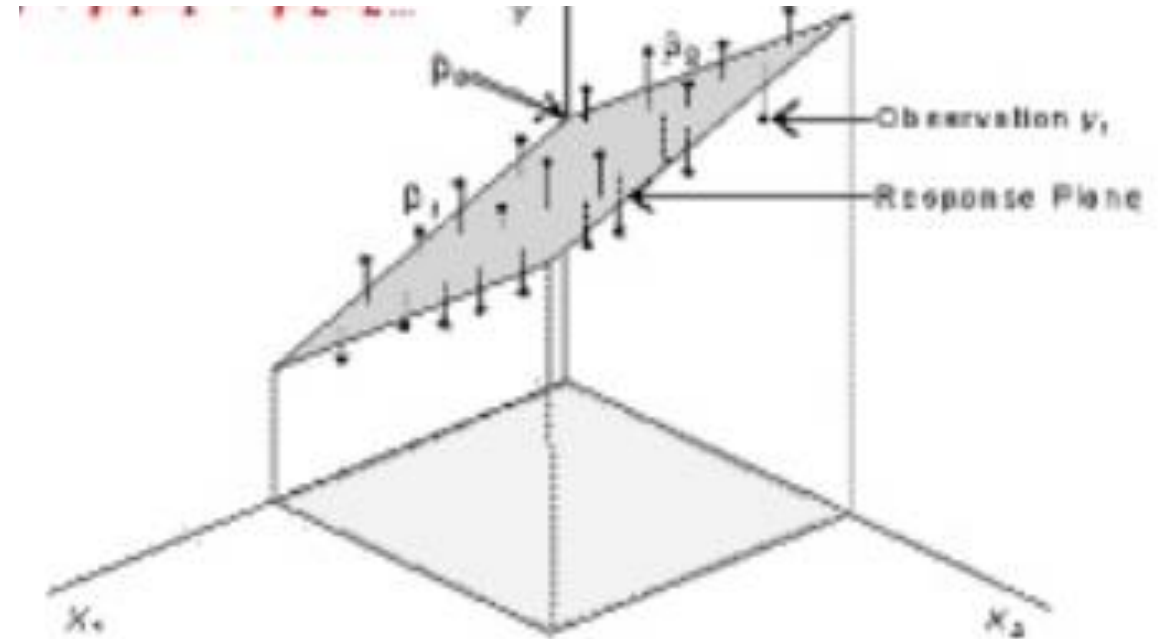
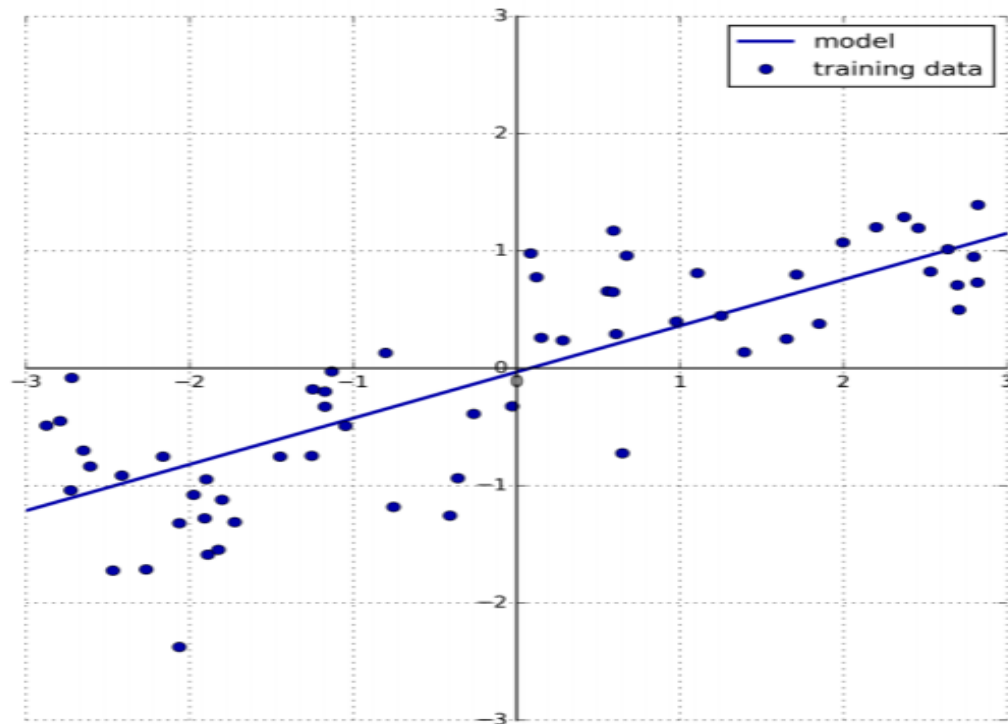


Figure 2-11. Predictions of a linear model on the wave dataset

피쳐가 1개인 데이터!

$$\hat{y} = w[0] * x[0] + b$$

In[25]:

```
mglearn.plots.plot_linear_regression_wave()
```

Out[25]:

```
w[0]: 0.393906  b: -0.031804
```

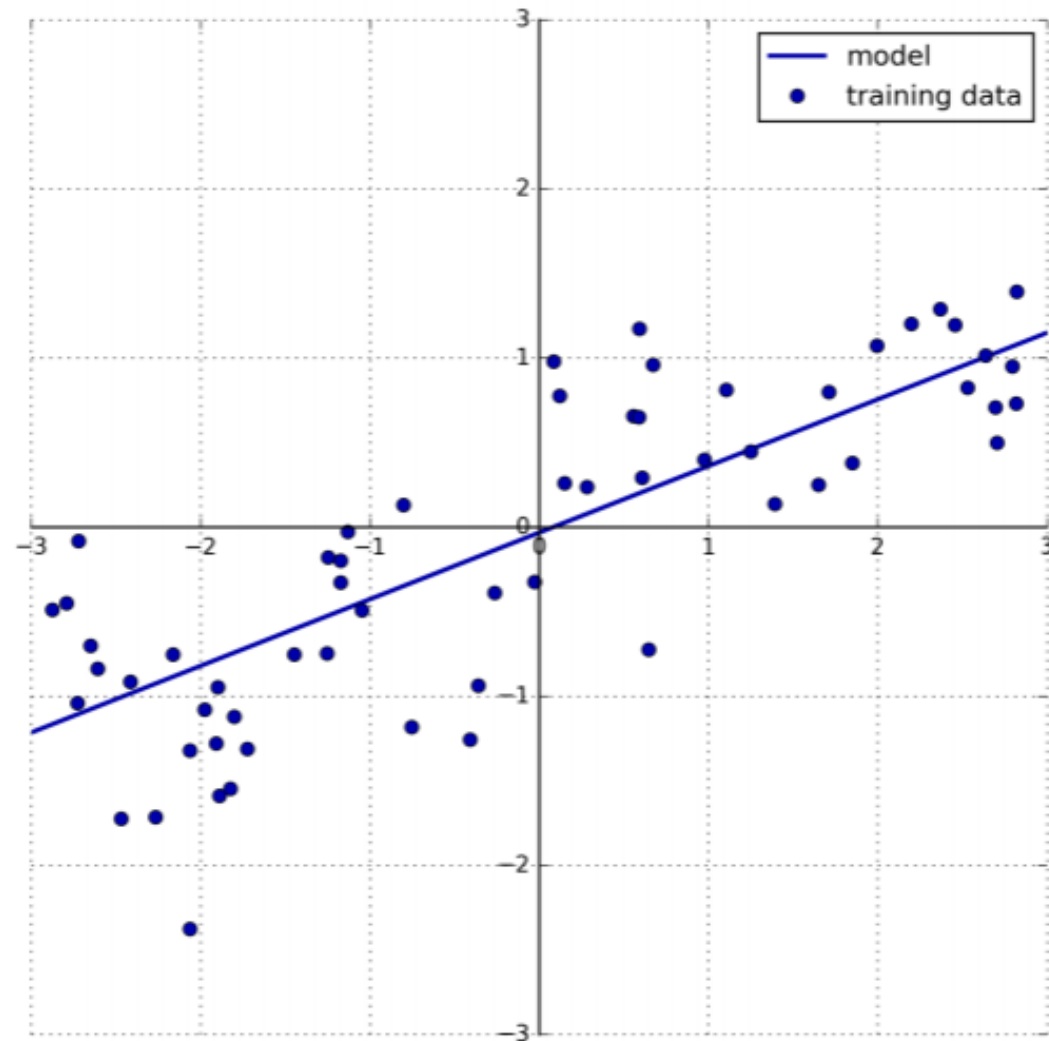


Figure 2-11. Predictions of a linear model on the wave dataset

In[26]:

```
from sklearn.linear_model import LinearRegression
X, y = mglearn.datasets.make_wave(n_samples=60)
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=42)

lr = LinearRegression().fit(X_train, y_train)
```

The “slope” parameters (w), also called weights or *coefficients*, are stored in the `coef_` attribute, while the offset or *intercept* (b) is stored in the `intercept_` attribute:

In[27]:

```
print("lr.coef_: {}".format(lr.coef_))
print("lr.intercept_: {}".format(lr.intercept_))
```

Out[27]:

```
lr.coef_: [ 0.394]          array
lr.intercept_: -0.031804343026759746    float
```

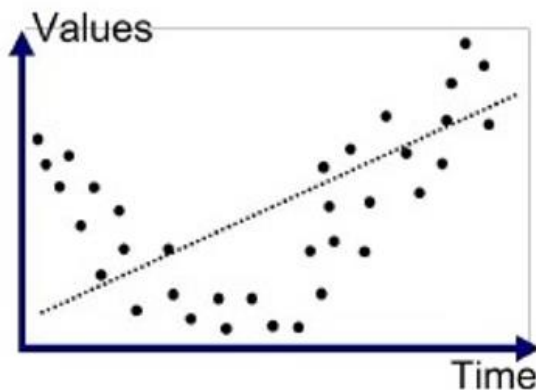
In[28]:

```
print("Training set score: {:.2f}".format(lr.score(X_train, y_train)))  
print("Test set score: {:.2f}".format(lr.score(X_test, y_test)))
```

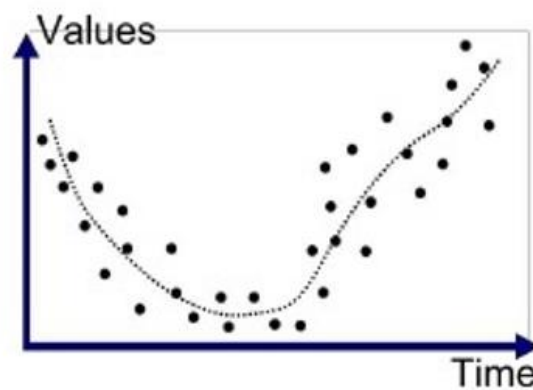
Out[28]:

```
Training set score: 0.67  
Test set score: 0.66
```

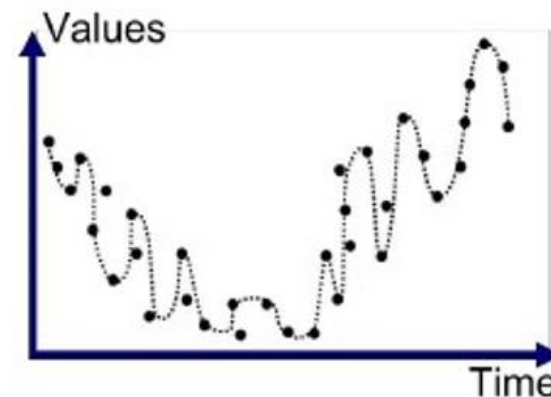
Q. 위와 같은 정확도는
다음 중 어떤 fit이 된 걸까요?



Underfitted



Good Fit/Robust



Overfitted

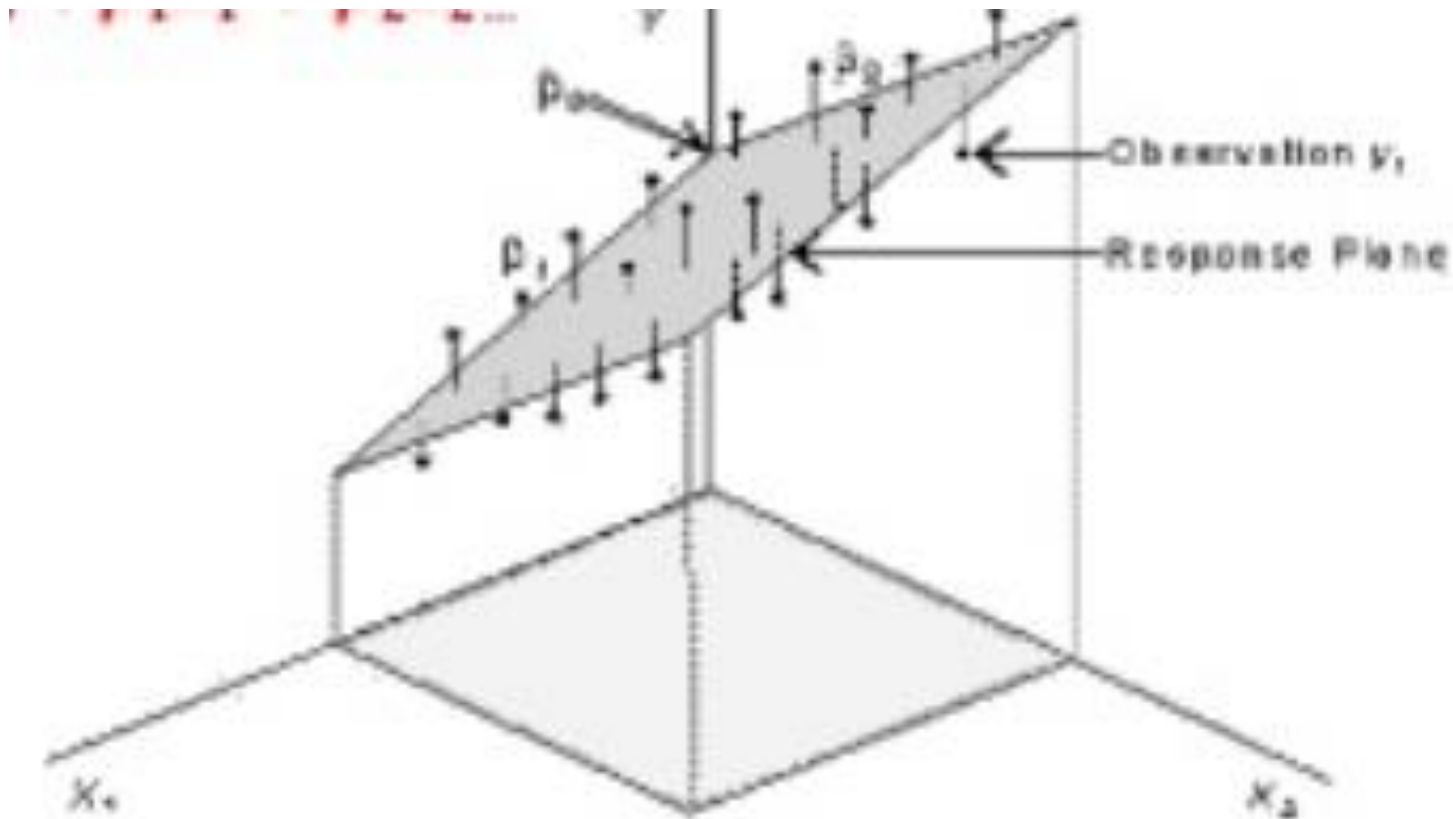
피쳐가 2개인 데이터!

$$Y = W[0]*X[0] + W[1]*X[1] + b$$

피쳐가 여러 개인 데이터??

그래프 상에 표현 불가!!!

$$h_{\theta}(x) = [\theta_0 \quad \theta_1 \quad \dots \quad \theta_n] \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix} = \theta^T x$$



In[29]:

```
X, y = mglearn.datasets.load_extended_boston()
```

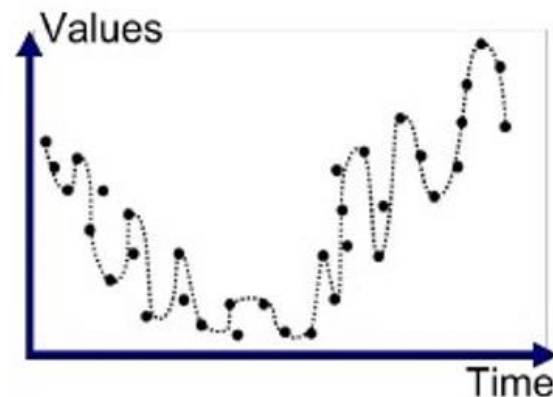
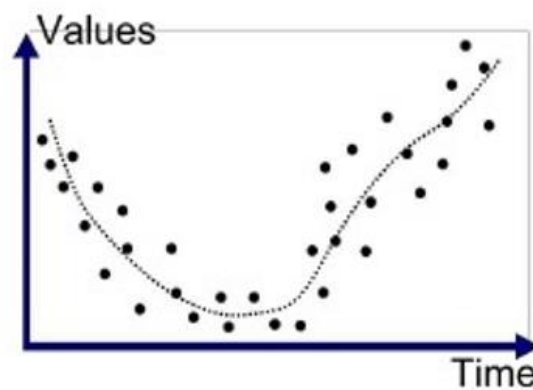
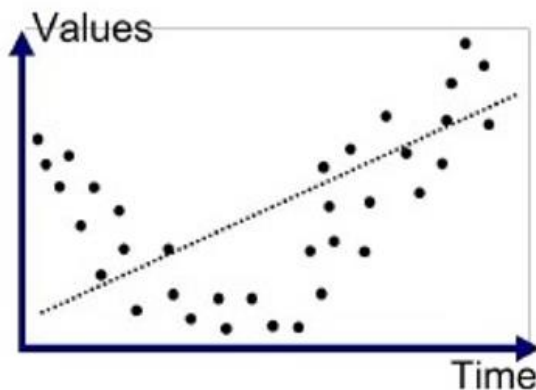
```
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)  
lr = LinearRegression().fit(X_train, y_train)
```

In[30]:

```
print("Training set score: {:.2f}".format(lr.score(X_train, y_train)))  
print("Test set score: {:.2f}".format(lr.score(X_test, y_test)))
```

Out[30]:

```
Training set score: 0.95  
Test set score: 0.61
```



Q. 위와 같은 정확도는
다음 중 어떤 fit이 된 걸까요?

Underfitted

Good Fit/Robust

Overfitted

W 값을 parameter로 이용하고 싶다.....!!!!!!!!!!!!!!

왜?? 투머치 피쳐 = 피쳐를 없애고 싶다 = 한 피쳐가
결과 Y에 미치는 영향이 작았으면 좋겠다 = overfit를
방지하고 싶다 등등 다 같은 말

In[31]:

```
from sklearn.linear_model import Ridge

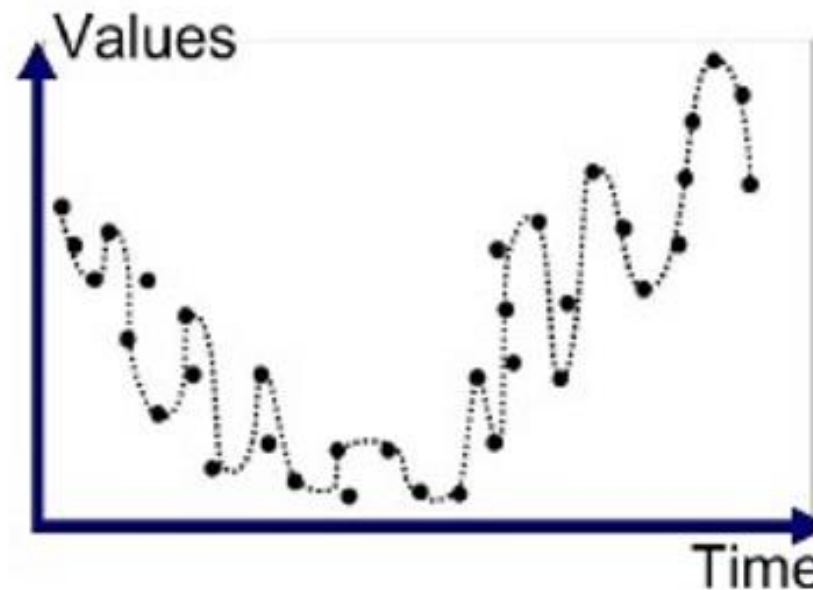
ridge = Ridge().fit(X_train, y_train)
print("Training set score: {:.2f}".format(ridge.score(X_train, y_train)))
print("Test set score: {:.2f}".format(ridge.score(X_test, y_test)))
```

Out[31]:

```
Training set score: 0.89
Test set score: 0.75
```

Out[30]:

```
Training set score: 0.95
Test set score: 0.61
```



Overfitted

리지 회귀(Ridge regression)

W 값을 parameter로 이용하고 싶다.....!!!
 → 알파를 조절해서!!!!!!!!!!

$$\theta_j := \theta_j - \alpha \left[\underbrace{\frac{1}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)}) x_j^{(i)}}_{(j = \textcolor{red}{x}, 1, 2, 3, \dots, n)} + \frac{\lambda}{n} \theta_j \right]$$

알파를 크게!

업데이트 되는 세타(w) 값이 0에 가까워짐
 세타(w)가 사라짐
 EX) n차 다항함수가 n-1차 다항함수가 됨
 → Overfit 방지

In[32]:

```
ridge10 = Ridge(alpha=10).fit(X_train, y_train)
print("Training set score: {:.2f}".format(ridge10.score(X_train, y_train)))
print("Test set score: {:.2f}".format(ridge10.score(X_test, y_test)))
```

Out[32]:

```
Training set score: 0.79
Test set score: 0.64
```

Out[30]:

```
Training set score: 0.95
Test set score: 0.61
```

In[33]:

```
ridge01 = Ridge(alpha=0.1).fit(X_train, y_train)
print("Training set score: {:.2f}".format(ridge01.score(X_train, y_train)))
print("Test set score: {:.2f}".format(ridge01.score(X_test, y_test)))
```

Out[33]:

```
Training set score: 0.93
Test set score: 0.77
```

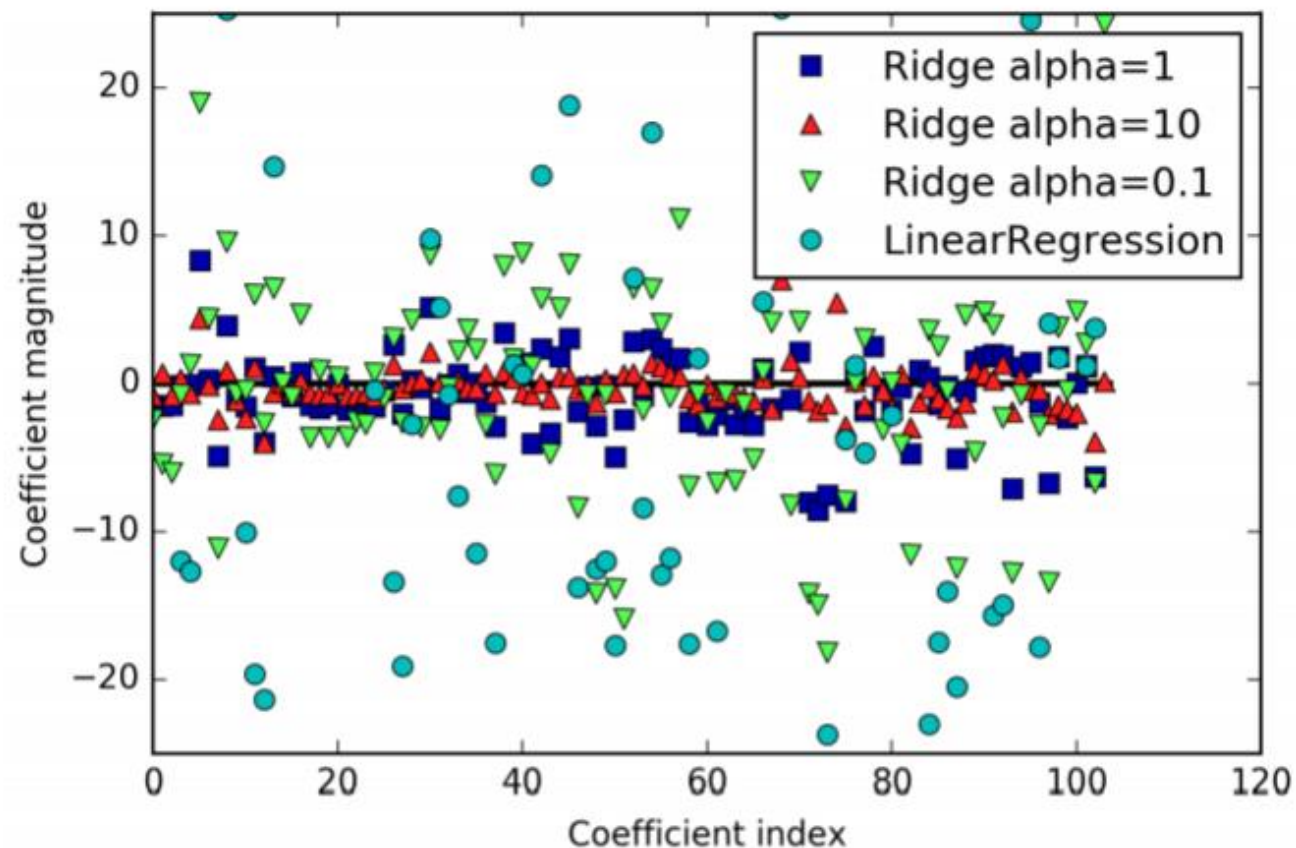
알파를 작게...

업데이트 되는 세타(W) 값이 원래 값이
 랑 같음
 선형회귀랑 다를바가 없음
 → overfit 가능성 생김

In[34]:

```
plt.plot(ridge.coef_, 's', label="Ridge alpha=1")
plt.plot(ridge10.coef_, '^', label="Ridge alpha=10")
plt.plot(ridge01.coef_, 'v', label="Ridge alpha=0.1")
```

```
plt.plot(lr.coef_, 'o', label="LinearRegression")
plt.xlabel("Coefficient index")
plt.ylabel("Coefficient magnitude")
plt.hlines(0, 0, len(lr.coef_))
plt.ylim(-25, 25)
plt.legend()
```



리지 회귀(Ridge regression)

In[35]:

`mglearn.plots.plot_ridge_n_samples()`

Training score 값은 ridge나 linear나 비슷하지만
데이터가 적을 땐 ridge가 훨 이득.
500 넘어가면 둘 다 괜찮다.

linear의 predict이 점점 줄어든다?
overfit이 일어나기 어려워진다.

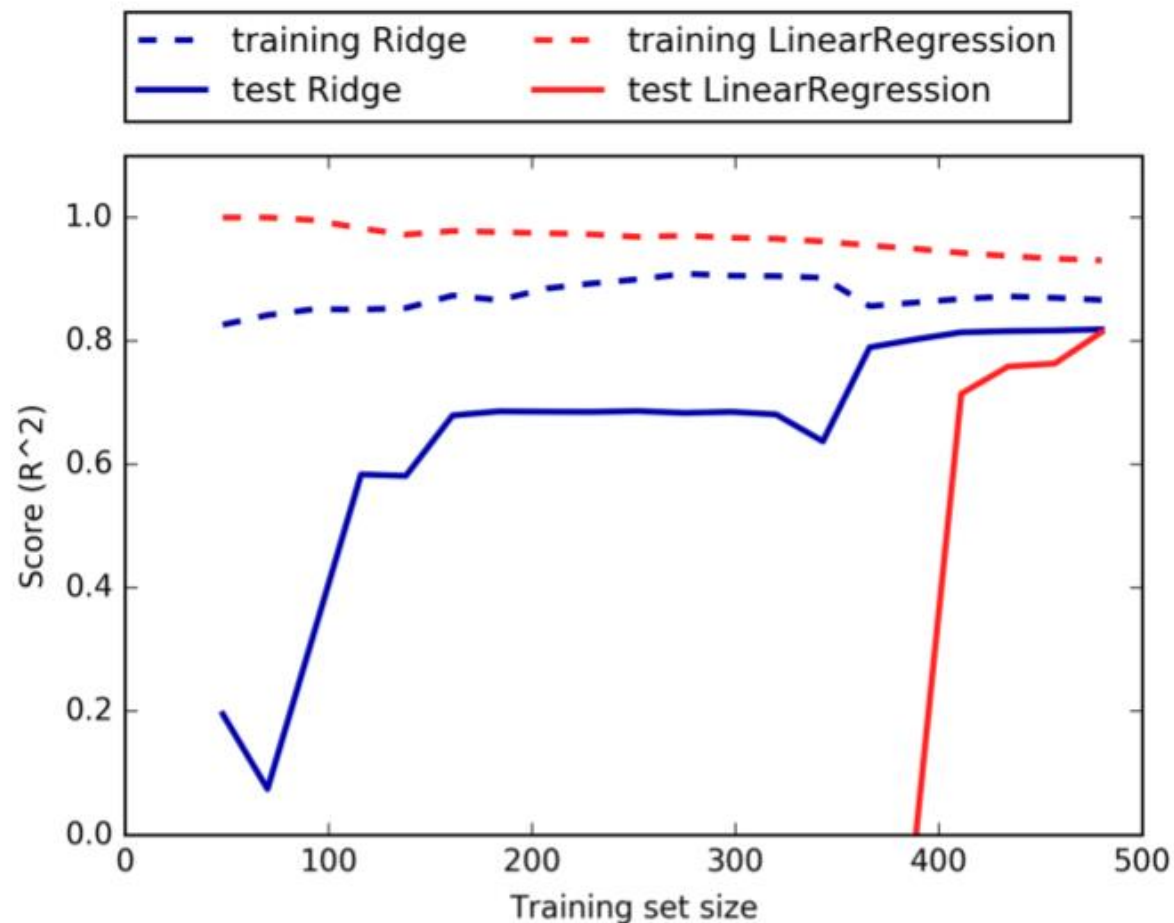


Figure 2-13. Learning curves for ridge regression and linear regression on the Boston Housing dataset

라쏘 회귀(Lasso regression)

리지 회귀랑 원리는 똑같다.

다만 선택적으로 W 를 버릴 수 있다.

예컨대, 알파를 엄청 크게해서 아예 W 가 0이 되어 없어지게 해버리기!

한 피쳐의 영향력을 없애버리기 = 자동으로 의미있는 피쳐만 남기는 선택 기계

In[36]:

```
from sklearn.linear_model import Lasso

lasso = Lasso().fit(X_train, y_train)
print("Training set score: {:.2f}".format(lasso.score(X_train, y_train)))
print("Test set score: {:.2f}".format(lasso.score(X_test, y_test)))
print("Number of features used: {}".format(np.sum(lasso.coef_ != 0)))
```

Out[36]:

```
Training set score: 0.29
Test set score: 0.21
Number of features used: 4
```

라쏘 회귀(Lasso regression)

In[37]:

```
# we increase the default setting of "max_iter"
# otherwise the model would warn us that we s
lasso001 = Lasso(alpha=0.01, max_iter=100000)
print("Training set score: {:.2f}".format(lasso001.score(X_train, y_train))
print("Test set score: {:.2f}".format(lasso001.score(X_test, y_test))
print("Number of features used: {}".format(np.sum(lasso001.coef_ != 0)))
```

Out[37]:

```
Training set score: 0.90
Test set score: 0.77
Number of features used: 33
```

In[38]:

```
l
p
p
p
```

Out[38]:

```
T
T
N
```

Out[36]:

```
Training set score: 0.29
Test set score: 0.21
Number of features used: 4
```

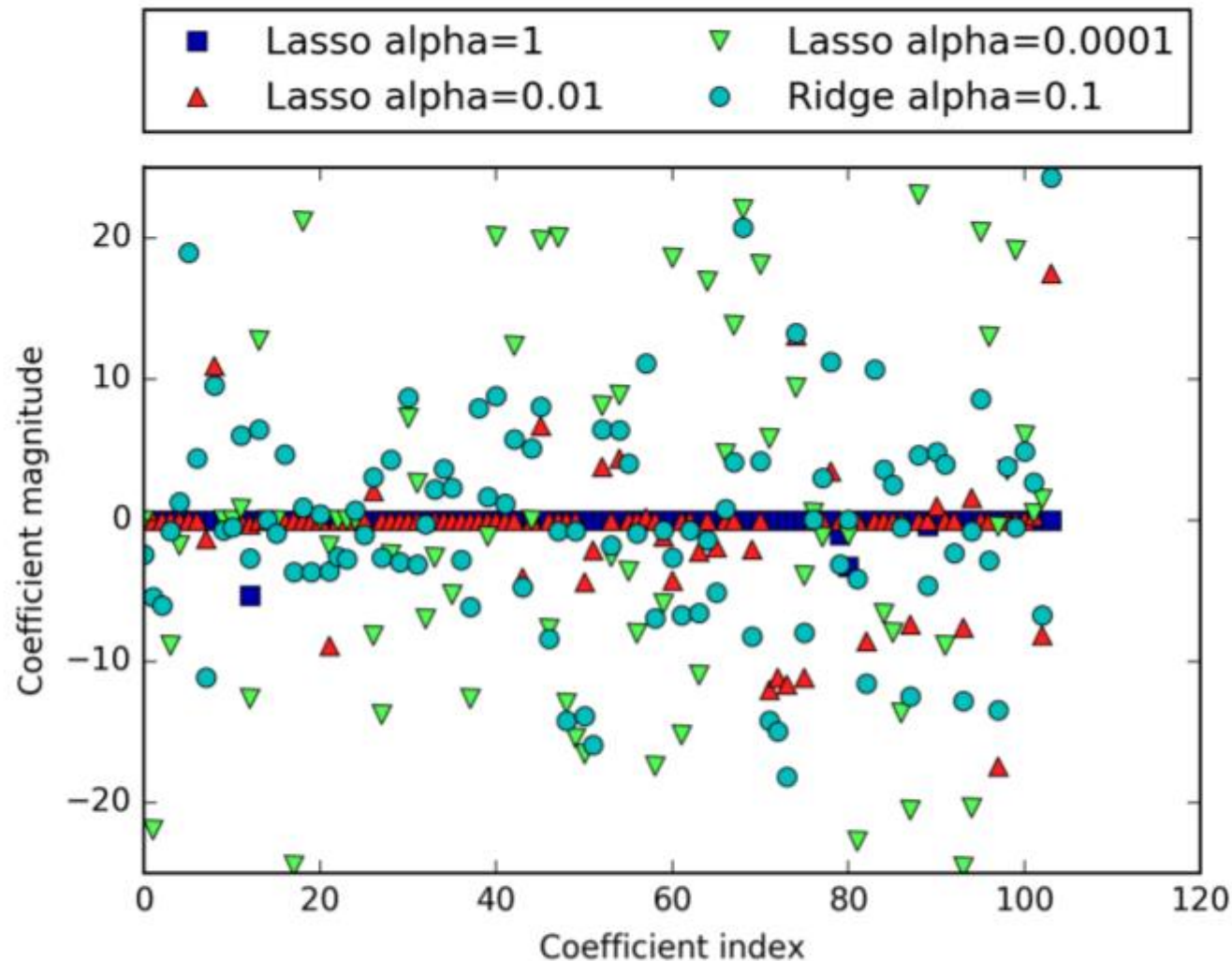


Figure 2-14. Comparing coefficient magnitudes for lasso regression with different values of alpha and ridge regression

$$\hat{y} = w[0] * x[0] + w[1] * x[1] + \dots + w[p] * x[p] + b > 0$$

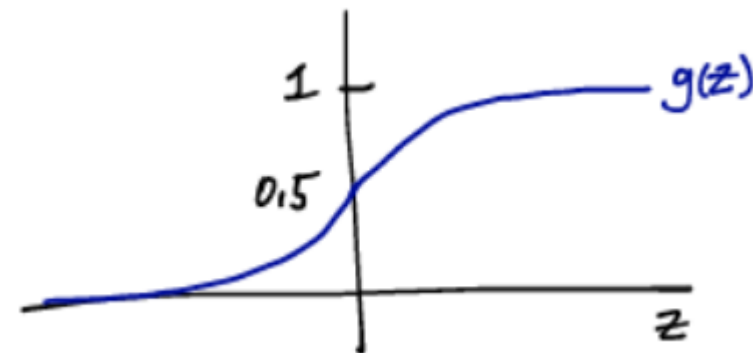
선형회귀의 선,면 대신에 decision boundary=영역 따먹기 로 바뀌었을 뿐...

$$\frac{\partial}{\partial \theta_j} J(\theta) = \frac{1}{m} \sum_{i=1}^m \left(h_{\theta}(x^{(i)}) - y^{(i)} \right) x_j^{(i)}$$

$$h_{\theta}(x) = \frac{1}{1 + \exp(-\theta^T x)}$$

환자 암 크기 표면모양 암 여부

INDEX	Feature1 (X1)	Feature2 (X2)	Label(Y)
1	0.25	0.53	0
2	0.70	0.77	1
3	0.67	0.11	1
4	0.35	0.52	1



피쳐가 2개인 make 함수

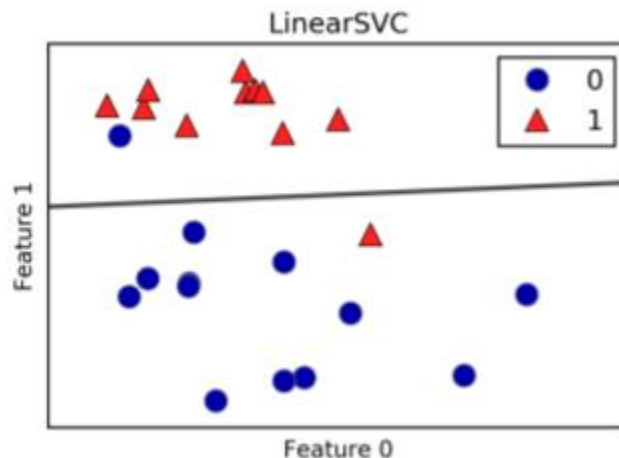
In[40]:

```
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()
```



2차원 데이터셋 분할 평면 그리기
 # model 객체, train 데이터, 평면 칠하기, 입실론, 축, 투명도

이 모델도 리지, 라쏘처럼 W 를 parameter로 이용하고 싶어!

→ C 값을 변화시킴

→ C 는 알파와 반비례

In[41]:

```
mglearn.plots.plot_linear_svc_regularization()
```

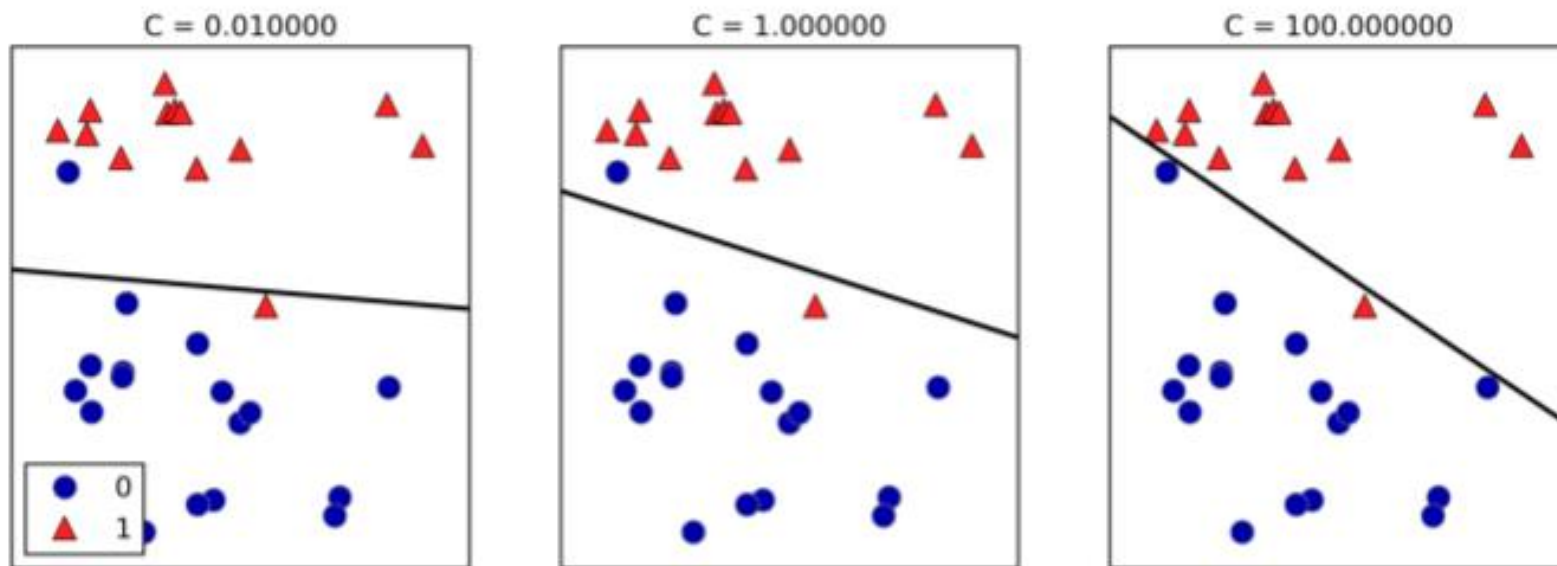


Figure 2-16. Decision boundaries of a linear SVM on the forge dataset for different values of C

분류를 위한 선형 모델

피쳐가 여러 개 일 때! → breast cancer data

In[42]:

```
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)))
```

Out[42]:

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

In[43]:

```
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)))
```

Out[43]:

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

C = 100

W가 커짐

Train 정확도가 높지만

Overfit 가능성 생김

In[44]:

```
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)))
```

Out[44]:

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

C = 0.01

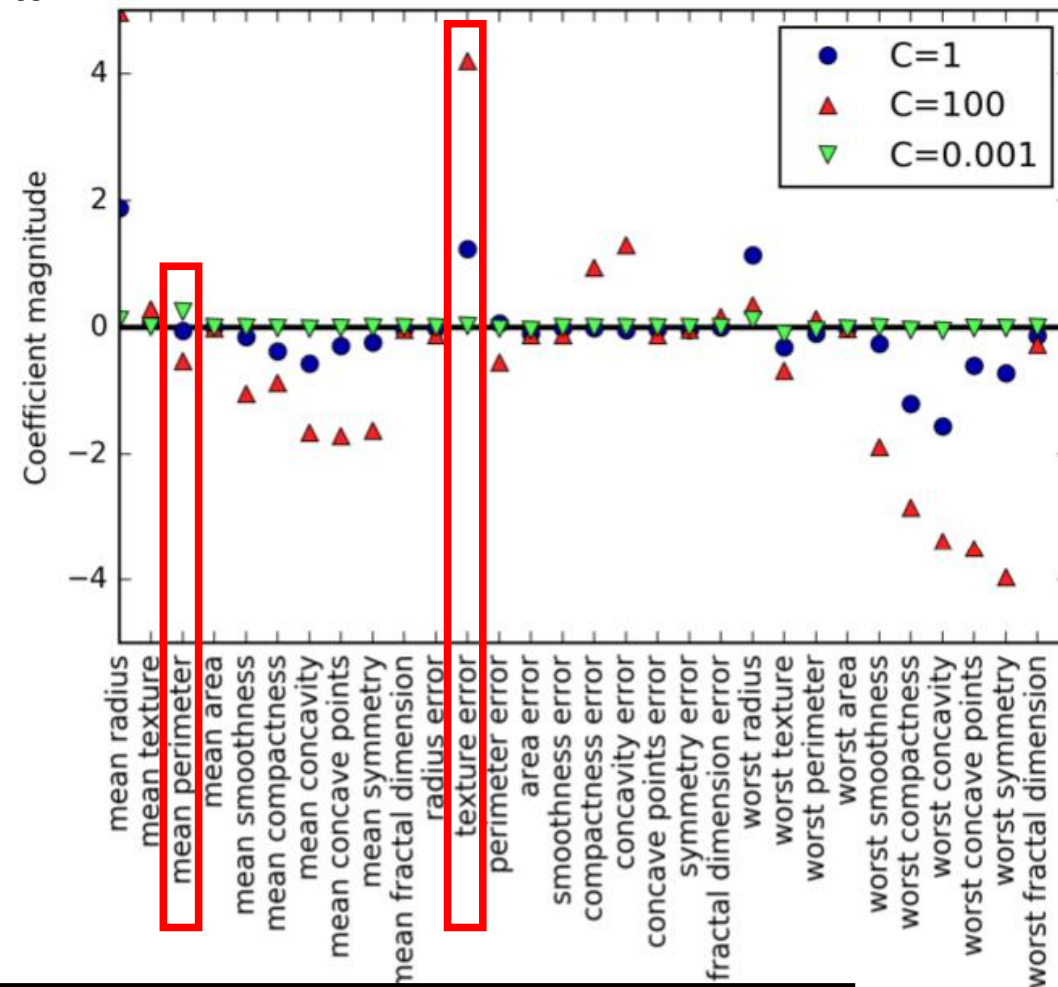
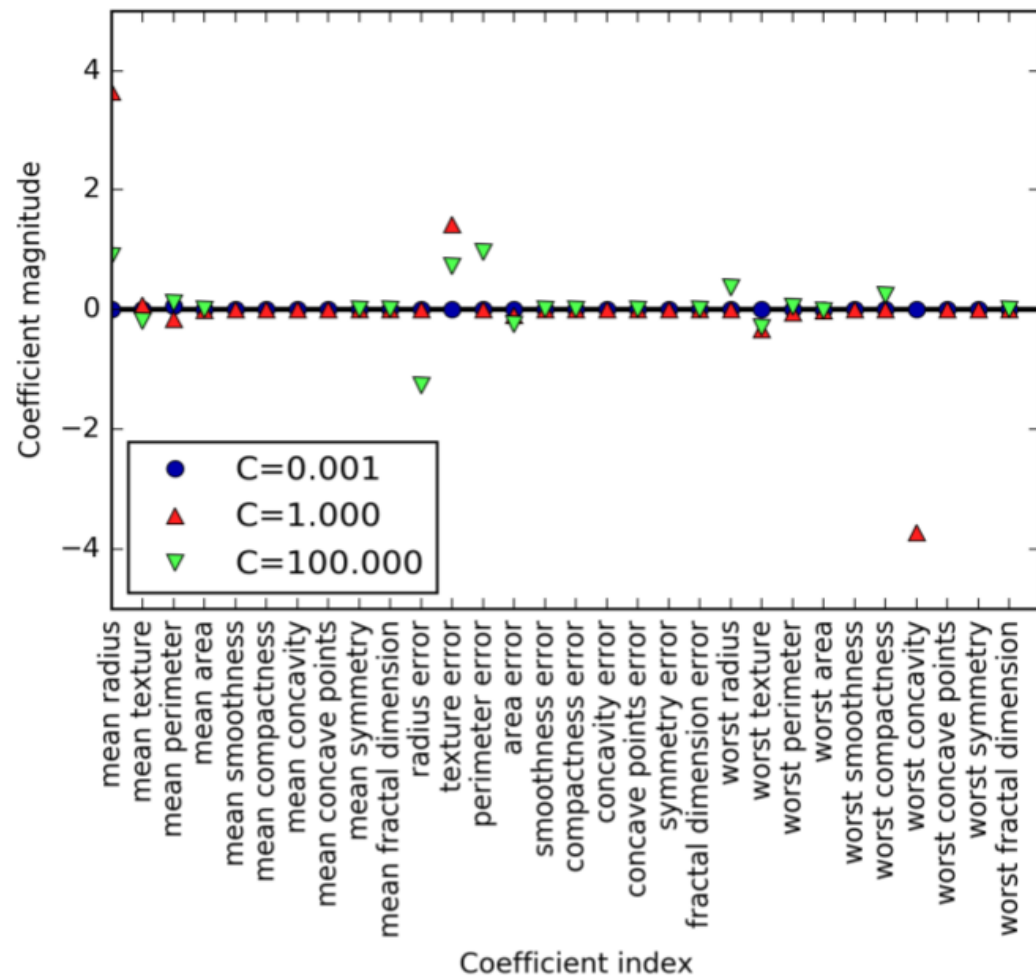
W가 0에 가까워짐

몇몇 피쳐가 사라짐

Overfit 방지

분류를 위한 선형 모델

피쳐가 여러 개 일 때! → breast cancer data



라쏘이용하면 강력한 피쳐를 필터링 가능

Figure 2-18. Coefficients learned by logistic regression with L1 penalty on the Breast Cancer dataset for different values of C

Cancer dataset for

다중클래스 선형 모델

분류 결과가 여러 개일 때...
나 아니면 나머지 라고 생각해서 binary class랑 똑같이 분류

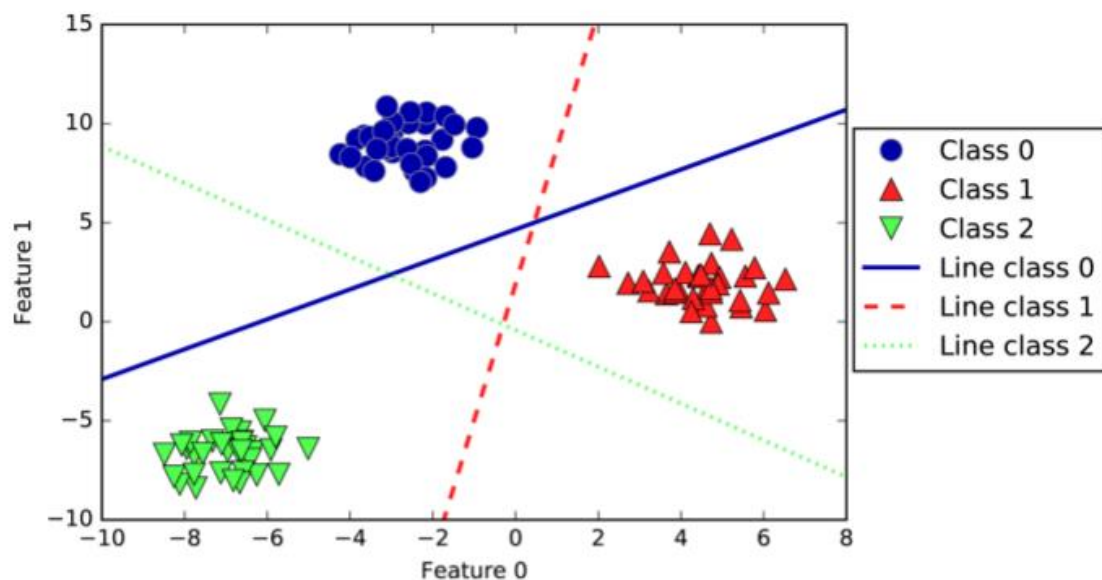


Figure 2-20. Decision boundaries learned by the three one-vs.-rest classifiers

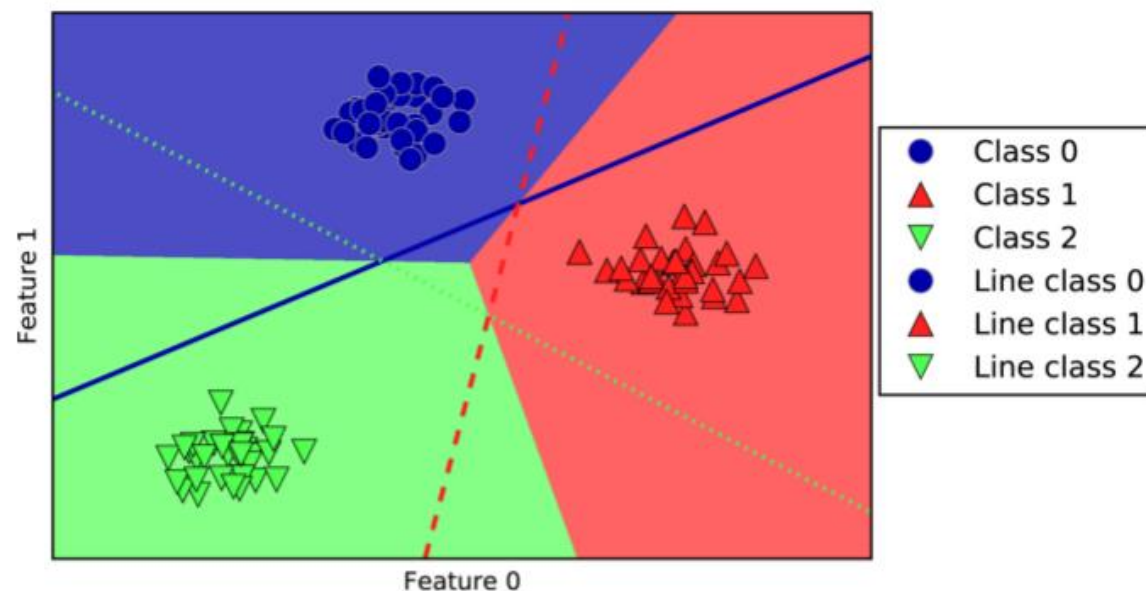


Figure 2-21. Multiclass decision boundaries derived from the three one-vs.-rest classifiers

회귀에서 a 와 분류에서 C 가 중요하다.

→ 알파가 크고 C 가 작으면 없어지는 피쳐가 생기는,

간단한 모델=일반화가 쉬운 모델=overfit 방지 모델

→ 알파가 작고 C 가 크면 피쳐가 대부분 살아있는,

복잡한 모델=굉장히 제한적인 모델=overfit 가능성 존재 모델

근데 a, C 는 보통 로그 함수를 따르기 때문에 어느정도 답이 정해져 있다.

→ 그래서 regularization을 알고리즘의 선택이 중요하다.

몇몇 특이한 피쳐가 중요하다면(해석가능성이 크면) $L1$ (라쏘)를 쓰고, 아니면 $L2$ (리지)를 쓰고(디폴트값).

→ 선형 모델의 장점은 빠른 학습과 빠른 예측. 대신 단점은 데이터가 많아야한다.

또 장점은 왜 저렇게 예측되었는지 이해하기 쉽다.

따라서, 주로 샘플이 많고 피쳐는 적을 때 사용한다.

감사합니다