머신러닝 (Machine Learning)

LECTURE X: 지도 학습 4 (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
  - 나이브베이즈 모델
  - 결정트리
  - 결정트리의 앙상블
  - 커널 서포트 벡터 머신
  - 신경망, 딥러닝
- 분류 예측의 불확실성 추정

ⓒ dgkim@hanyang.ac.kr

규제가 있는 선형 모델

- 과대적합을 줄이는 좋은 방법은 모델을 규제하는 것임 (모델 파라미터의 자유도를 제한함). 자유도를 줄이면 데이터에 과대적합되는 것을 줄일 수 있음.
- 선형 회귀 모델에서는 모델의 가중치를 제한함으로써 규제를 가함.
  1) 릿지 회귀(Ridge Regression) 또는 Tikhonov regularization.
  2) 라쏘 회귀(Lasso Least Absolute Shrinkage and Selection Operator Regression)
  3) 엘라스틱넷(Elastic Net)

- 규제가 부과된 모델식

$$J(\theta) = \alpha\,\text{규제함수}(\theta) + \text{비용함수}(\theta)$$

✓ 규제함수는 가중치 벡터의 크기를 제약하는 함수임.

✓ 이러한 규제는 과대적합이 되지 않도록 모델을 강제로 제한하는 것임.

✓ 하이퍼파라미터 $\alpha \geq 0$는 모델을 얼마나 많이 규제할지 조절하는 모수임.

ⓒ dgkim@hanyang.ac.kr

❖ 릿지 회귀

- 다음과 같이 ℓ_2 규제로 θ에 제약을 가한 선형 모델을 **릿지**(Ridge) **회귀**라고 함.

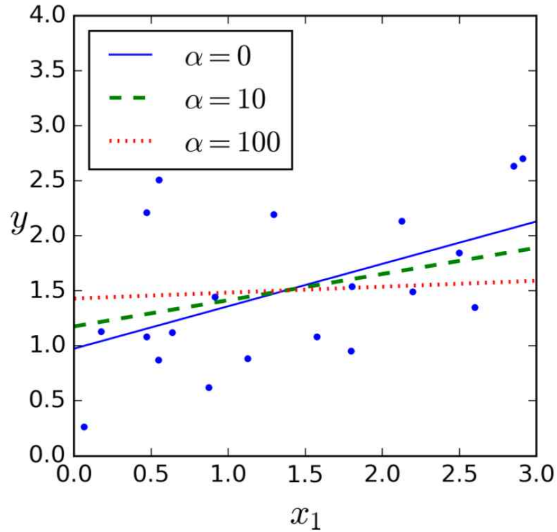
$$J(\boldsymbol{\theta}) = \frac{\alpha}{2}\sum_{j=1}^{n}\theta_j^2 + \mathrm{MSE}\left(\boldsymbol{\theta}\right)$$

✓ 편향 $\theta_{n+1} = b$는 규제하지 않음.

✓ $\alpha \approx 0$: 선형 회귀에 가까움.

✓ $\alpha \gg 1$: 모든 가중치는 거의 0에 가까워짐.

- 다음 그림은 선형 데이터에 몇 가지 다른 하이퍼파라미터 α를 적용해 릿지 모델을 훈련시킨 결과임.



- 릿지 회귀를 구현하는 계산으로 정규 방정식 또는 경사 하강법을 사용할 수 있음.

[경사 하강법]

$$\boldsymbol{\theta}^{(new)} = \boldsymbol{\theta} - \eta\left(\nabla_{\boldsymbol{\theta}} \mathrm{MSE}(\boldsymbol{\theta}) + \alpha \tilde{\boldsymbol{\theta}}\right)$$

▷ $\tilde{\theta} = (\theta_1, \ldots, \theta_n, 0)^T$

[정규 방정식]

$$\hat{\boldsymbol{\theta}} = \left(\frac{2}{m} A^T A + \alpha \tilde{I}\right)^{-1} A^T \boldsymbol{y}$$

▷ $\tilde{I} = diag[1, \ldots, 1, 0] \in \mathbb{R}^{(n+1)\times(n+1)}$

- 다음은 사이킷런에서 릿지 회귀의 정규 방정식을 확장된 보스턴 주택가격 데이터셋에 적용한 예임.

```
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)))
```

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

✓ 이 모델은 선형 회귀(LinearRegression) 보다 일반화 성능이 더 좋음.

✓ 여기서 하이퍼파라미터의 기본값은 $\alpha = 1$.

✓ 최적의 α 값은 사용하는 훈련 데이터셋에 달려있음.

 ⓒ dgkim@hanyang.ac.kr

- 다음 코드는 각각 $\alpha = 10,\ 0.1$ 경우임.

```
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)))
```

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

```
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)))
```

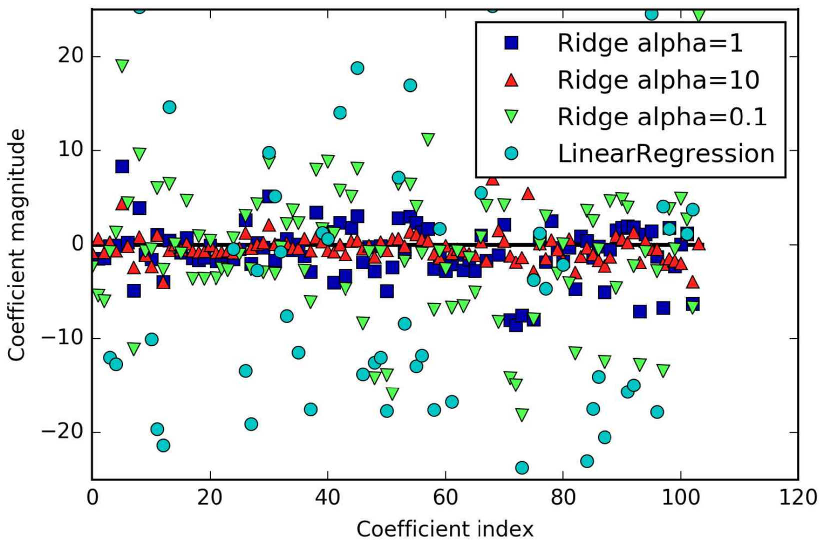
```
Training set score: 0.93
Test set score: 0.77
```

✓ $\alpha = 0.1$일 때 일반화 성능이 우수함을 볼 수 있음.

✓ 최적의 α 값은 모델 평가와 성능 향상의 방법으로 구할 수 있음.

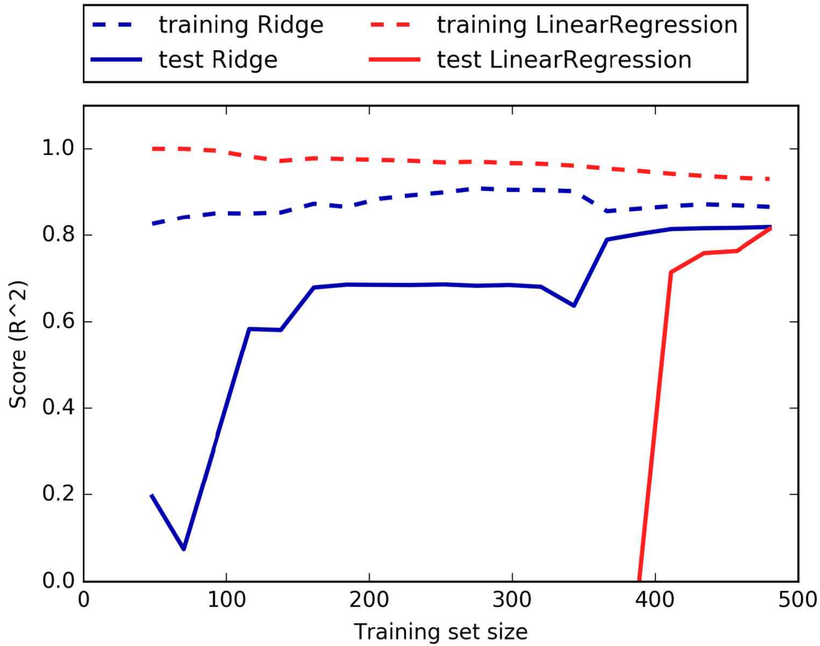
- 다음 코드는 α 값에 따라 모델의 가중치 또는 계수 $\theta_1, \ldots, \theta_n$ 크기의 속성을 보여주며, 이를 통하여 규제 효과를 살펴 볼 수 있음.

```
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()
```



- 규제 효과를 이해하는 또 다른 방법은 α 값을 고정하고 훈련 데이터의 크기를 변화시켜 모델의 성능을 살펴보는 것임.
- 다음은 보스턴 주택가격 데이터셋에서 여러 가지 크기로 샘플링하여 선형 회귀와 릿지 회귀($\alpha = 1$)를 적용한 결과를 학습곡선으로 보여주는 것임.

- 훈련 과정을 여러 번 반복하면서 학습하는 알고리즘에서 반복횟수에 따른 성능 변화 또는 데이터셋의 크기에 따른 모델의 성능 변화를 나타내는 그래프를 **학습곡선** (**learning curve**)라고 함.
  - ✓ 릿지 회귀의 훈련 데이터에 대한 성능은 선형 회귀에 비해 전체적으로 낮음.
  - ✓ 릿지 회귀의 테스트 데이터에 대한 성능은 선형 회귀에 비해 전체적으로 높음.
  - ✓ 데이터의 양이 충분히 많으면 릿지 회귀와 선형 회귀의 성능이 같아지는 경향이 있음.
  - ✓ 선형 회귀는 훈련 데이터의 양이 많아질수록 훈련 데이터 성능이 감소하는 경향이 있음.

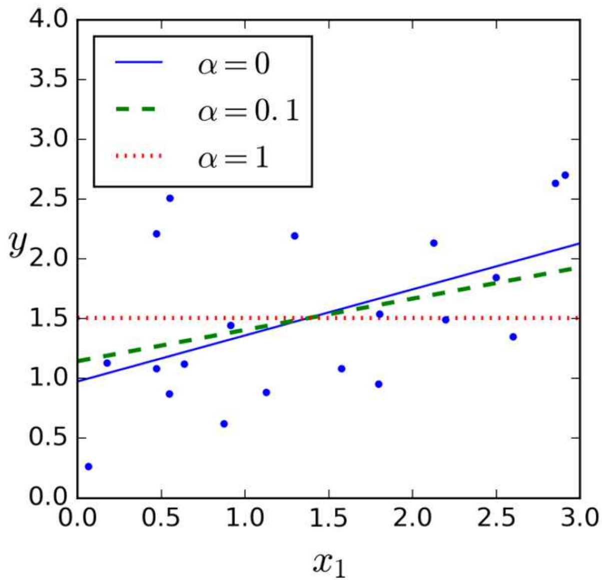


❖ 라쏘 회귀

- **라쏘 회귀(Lasso**Least Absolute Shrinkage and Selection Operator **Regression)**는 선형 회귀의 또 다른 규제된 모델임.
- 릿지 회귀는 ℓ_2노름의 규제항을 사용하지만, 라쏘 회귀는 ℓ_1노름의 규제항을 사용함.
- 라쏘 회귀의 비용함수

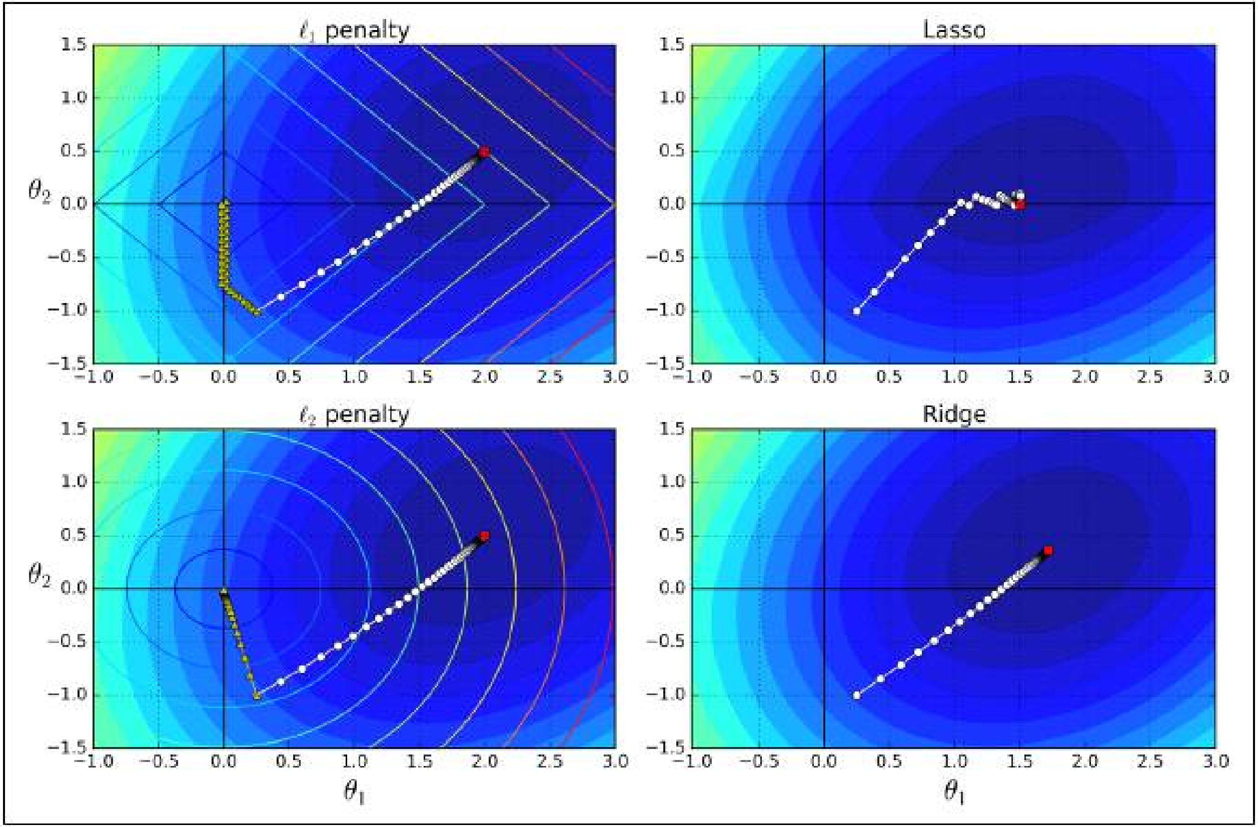
$$J(\boldsymbol{\theta}) = \alpha \sum_{j=1}^{n} \left|\theta_j\right| + \mathrm{MSE}\left(\boldsymbol{\theta}\right)$$

- 다음 그림은 선형 데이터에 몇 가지 다른 하이퍼파라미터 α를 사용해 라쏘 회귀 모델을 훈련시킨 결과임.



- ✓ 라쏘 회귀의 중요한 특징은 덜 중요한 가중치를 완전히 제거하는 경향이 있다는 것임.
- ✓ 따라서 어떤 특성은 라쏘 모델에서 완전히 제외될 수 있음.
- ✓ 일부 계수를 0으로 만들면 모델을 이해하기 쉽고 이 모델을 통해 데이터의 가장 중요한 특성을 파악할 수 있음.

- 라쏘 회귀는 자동으로 특성을 선택하고 **희소 모델(sparse model)**을 만듦.



ⓒ dgkim@hanyang.ac.kr

- ✓ 왼쪽 위 그래프 배경의 등고선(타원형)은 규제가 없는 MSE 비용함수를 나타내고, 하얀색 동그라미 점은 비용함수에 대한 경사 하강법의 경로임.
- ✓ 전방 등고선(다이아몬드 형태)은 ℓ_1 패널티를 나타내며 삼각형 점은 이 패널티에 대한 경사 하강법이 경로임 $(\alpha \to \infty)$.
- ✓ 오른쪽 위 그래프의 등고선이 나타내는 것은 $\alpha = 0.5$의 ℓ_1 규제가 추가된 라쏘의 비용 함수를 나타내고 있음.

[라쏘 회귀 비용함수의 그래디언트]

- 라쏘의 비용 함수는 $\theta_j = 0 \ (j = 1, \ldots, n)$에서 미분 가능하지 않음.
- $\theta_j = 0$일 때 ℓ_1 패널티 함수의 서브그래디언트(subgradient) 벡터를 사용하여 경사 하강법을 적용할 수 있음.

$$\nabla_\theta J = \nabla_\theta \mathrm{MSE}(\boldsymbol{\theta}) + \alpha \begin{pmatrix} \mathrm{sign}(\theta_1) \\ \vdots \\ \mathrm{sign}(\theta_n) \end{pmatrix}, \quad \mathrm{sign}(\theta_j) = \begin{cases} -1 & \text{if } \theta_j < 0 \\ 0 & \text{if } \theta_j = 0 \\ 1 & \text{if } \theta_j > 0 \end{cases}$$

- 다음은 확장된 보스턴 주택가격 데이터셋에 라쏘 회귀 모델을 적용한 결과임.

```
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)))
```

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

✓ 여기서 하이퍼파라미터의 기본값은 $\alpha = 1$.

✓ 위의 라쏘 모델의 결과는 훈련 세트와 테스트 세트 모두에서 결과가 좋지 않음. 이 경우에는 104개의 특성 중 4개만 사용한 것이고, 심하게 과소적합된 모델임.

- 위의 과소적합을 줄이기 위해서는 α값을 줄여야 함.
- 라쏘 모델은 경사 하강법을 쓰기 때문에 α값을 줄일 경우 많은 반복 횟수가 요구됨.

```
# we increase the default setting of "max_iter",
# otherwise the model would warn us that we should increase max_iter.
lasso001 = Lasso(alpha=0.01, max_iter=100000).fit(X_train, y_train)
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:", np.sum(lasso001.coef_ != 0))
```

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

- ✓ α을 낮추면 모델의 복잡도가 증가하여 훈련 세트와 테스트 세트에서 성능이 좋아짐.
- ✓ 위 경우 $\alpha = 0.01$이고 사용된 특성은 104개 중 33개뿐임.
- ✓ α을 너무 낮추면 규제 효과가 없어져 과대적합이 되어 선형 회귀 모델의 결과와 비슷해짐.

ⓒ dgkim@hanyang.ac.kr

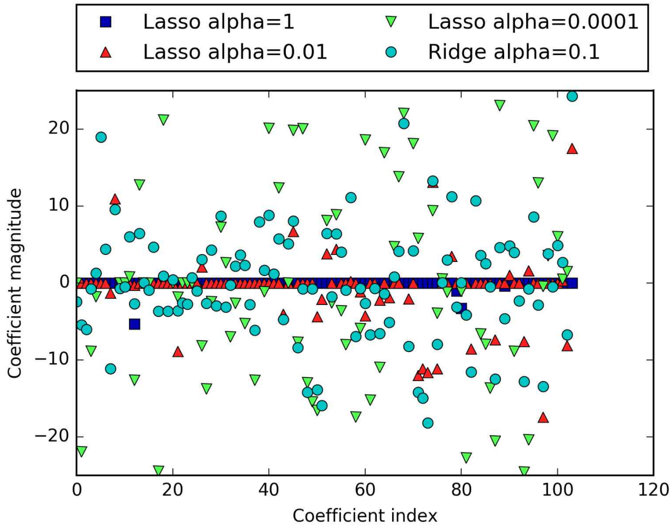
```
# we increase the default setting of "max_iter",
# otherwise the model would warn us that we should increase max_iter.
lasso00001 = Lasso(alpha=0.0001, max_iter=100000).fit(X_train, y_train)
print("Training set score: {:.2f}".format(lasso00001.score(X_train, y_train)))
print("Test set score: {:.2f}".format(lasso00001.score(X_test, y_test)))
print("Number of features used:", np.sum(lasso00001.coef_ != 0))
```

```
Training set score: 0.95
Test set score: 0.64
Number of features used: 94
```

- 다음은 α 값에 따른 라쏘 모델들의 계수를 그래프로 보여주는 코드임.

```
plt.plot(lasso.coef_, 's', label="Lasso alpha=1")
plt.plot(lasso001.coef_, '^', label="Lasso alpha=0.01")
plt.plot(lasso00001.coef_, 'v', label="Lasso alpha=0.0001")
#
plt.plot(ridge01.coef_, 'o', label="Ridge alpha=0.1")
plt.legend(ncol=2, loc=(0, 1.05))
```

```
plt.ylim(-25, 25)
plt.xlabel("Coefficient index")
plt.ylabel("Coefficient magnitude")
```



- ✓ $\alpha = 1$인 라쏘의 경우 거의 모든 계수가 0임.
- ✓ $\alpha = 0.1$인 릿지 모델은 $\alpha = 0.01$인 라쏘 모델과 성능이 비슷하지만 릿지의 경우 거의 모든 계수가 0이 아님.

- ✓ 보통의 경우 라쏘 모델보다 릿지 모델을 더 선호함.
- ✓ 그러나 특성이 아주 많고 그중 일부분만 중요하다는 사전 정보가 있으면 라쏘 모델이 아주 효율적임.
- ✓ 라쏘 모델은 보다 더 쉬운 분석 모델을 제공함.

엘라스틱넷

- 엘라스틱넷(Elastic Net)은 릿지 회귀와 라쏘 회귀를 절충한 모델임. 규제항은 릿지와 라쏘의 볼록조합(convex combination)으로 구성됨.

$$J(\boldsymbol{\theta}) = r\left(\alpha \sum_{j=1}^{n} |\theta_j|\right) + (1-r)\left(\frac{\alpha}{2} \sum_{j=1}^{n} \theta_j^2\right) + \mathrm{MSE}\,(\boldsymbol{\theta})$$

▷ $0 \le r \le 1$

- 엘라스틱넷이 실제로 최상의 성능을 내지만, ℓ_1규제와 ℓ_2규제를 위한 매개변수 두 개를 잘 조정해야함.
- 선형회귀, 릿지, 라쏘, 엘라스틱넷
  - ✓ 대부분의 경우 규제가 약간 있는 것이 좋은 결과를 얻을 수 있음.
  - ✓ 릿지가 기본적으로 쓰이지만, 실제로 특성이 몇 개뿐이라고 의심되면 라쏘나 엘라스틱넷이 효과적일 수 있음.
  - ✓ 특성 수가 훈련 샘플 수보다 많거나 특성 몇 개가 강하게 연관되어 있을 때는 보통 라쏘가 문제를 일으키므로 라쏘보다는 엘라스틱넷이 선호도가 높음.

- 다음은 사이킷런의 'ElasticNet'을 사용한 간단한 예제임.

```
from sklearn.linear_model import ElasticNet
elastic_net = ElasticNet(alpha=0.1, l1_ratio=0.5)
elastic_net.fit(X, y)
elastic_net.predict([[1.5]])
```

```
array([8.2459071])
```