25-1

Machine Learning Programming

5주차

소프트웨어학부 최종환 교수님

한림대학교

학습 내용

- Ridge Regression
- Lasso
- ElasticNet
- Stochastic Gradient Descent

[REMIND] Linear Regression

- In linear models, the target value is expected to be a linear combination of the features

$$y = \boldsymbol{w}^T \boldsymbol{x} = w_1 x_1 + w_2 x_2 + \cdots + w_p x_p + b$$

  - y: target value
  - $\boldsymbol{x} = (x_1, x_2, \ldots, x_p, 1)$: input vector (or feature vector)
  - $\boldsymbol{w} = (w_1, w_2, \ldots, w_p, b)$: coefficient vector

- The objective of linear models is to minimize the residual sum of squares between the actual and predicted targets

$$\min_{w} \|\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}\|_2^2$$

https://scikit-learn.org/stable/modules/linear_model.html#ordinary-least-squares

[Scikit-learn] LinearRegression

```
class   sklearn.linear_model.LinearRegression(*, fit_intercept=True,
        copy_X=True, n_jobs=None, positive=False)
```

Parameters	
fit_intercept	**bool, default=True** Whether to calculate the intercept for this model. If set to False, no intercept will be used in calculations (i.e. data is expected to be centered).
n_jobs	**int, default=None** The number of jobs to use for the computation. This will only provide speedup in case of sufficiently large problems, that is if firstly n_targets > 1 and secondly X is sparse or if positive is set to True. None means 1 unless in a joblib.parallel_backend context. -1 means using all processors. See Glossary for more details.

쓰기 전

https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html#sklearn.linear_model.LinearRegression

[Scikit-learn] LinearRegression

```
class   sklearn.linear_model.LinearRegression(*, fit_intercept=True,
        copy_X=True, n_jobs=None, positive=False)
```

Attributes	
coef_	**ndarray of shape (n_features,) or (n_targets, n_features)** Weight vector(s).
intercept_	**float or ndarray of shape (n_targets,)** Independent term in decision function. Set to 0.0 if fit_intercept = False.

사용하고 난 다음에 쓰는 표현(?)

https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html#sklearn.linear_model.LinearRegression

[Scikit-learn] LinearRegression

Example

```
import numpy as np
from sklearn.linear_model import LinearRegression
X = np.array([[1, 1], [1, 2], [2, 2], [2, 3]])
y = np.dot(X, np.array([1, 2])) + 3
reg = LinearRegression().fit(X, y)
reg.score(X, y) # 1.0 <-- r2_score
reg.coef_ # array([1., 2.])
reg.intercept_ # np.float64(3.0, ...)
reg.predict(np.array([[3, 5]])) # array([16.0])
```

Linear 모델을 만들어 냄.

9줄 중에서 일부러 에러가 나도록 한줄 바꿈.
어디 때문에 이 코드가 동작을 못하는지.
(9줄일지 확실하지 않음. 20줄이 될 수도 있음. 중간고사에서)

[Scikit-learn] LinearRegression

- The followings are equivalent

Example

```python
reg = LinearRegression().fit(X, y)
```

이 두 칸은 같은걸 실행함.

Example

```python
reg = LinearRegression()
reg = reg.fit(X, y)
```

Problem of OLS regression

- The least squares solution is computed using the singular value decomposition of X

$$\underset{\boldsymbol{w}}{\operatorname{argmin}} \|\boldsymbol{X}\boldsymbol{w}-\boldsymbol{y}\|_2^2 = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}$$

- If X is a matrix of shape (n_samples, n_features) and n_samples 〈 n_features, then the solution $\boldsymbol{w}$ is not unique!

$$\left|\begin{pmatrix}1 & 1\end{pmatrix}\begin{pmatrix}w_1\\ w_2\end{pmatrix}-1\right|^2$$

정답이 하나만 있는게 아님.

$$\left|\begin{pmatrix}1 & 0 & 0\\ 1 & 1 & 1\end{pmatrix}\begin{pmatrix}w_1\\ w_2\\ w_3\end{pmatrix}-\begin{pmatrix}0\\ 1\end{pmatrix}\right|^2$$

Ridge Regression

- Ridge regression addresses some of the problems of OLS by imposing a penalty on the size of the coefficients

- The ridge coefficients minimize a penalized residual sum of squares with $L2$ regularization:

$$\min_{w} \|\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}\|_2^2 + \alpha\|\boldsymbol{w}\|_2^2$$

- The parameter $\alpha \geq \mathbf{0}$ controls the amount of shrinkage (a.k.a regularization strength)
  - ➢ the larger the value of α, the greater the amount of shrinkage

https://scikit-learn.org/stable/modules/linear_model.html#ordinary-least-squares

[Scikit-learn] Ridge

```
class    sklearn.linear_model.Ridge(alpha=1.0, *, fit_intercept=True,
         copy_X=True, max_iter=None, tol=0.0001, solver='auto',
         positive=False, random_state=None)
```

Parameters	
alpha	**{float, ndarray of shape (n_targets,)}, default=1.0** Constant that multiplies the L2 term, controlling regularization strength. alpha must be a non-negative float i.e. in [0, inf).
max_iter	**int, default=None** Maximum number of iterations for conjugate gradient solver.
tol	**float, default=1e-4** The precision of the solution (coef_) is determined by tol which specifies a different convergence criterion for each solver: • ‘sag’ and ‘saga’: relative change of coef smaller than tol. • ‘lbfgs’: maximum of the absolute (projected) gradient=max¦residuals¦ smaller than tol.
random_state	**int, RandomState instance, default=None** Used when solver == ‘sag’ or ‘saga’ to shuffle the data.

https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Ridge.html#sklearn.linear_model.Ridge

[Scikit-learn] Ridge

```
class   sklearn.linear_model.Ridge(alpha=1.0, *, fit_intercept=True,
        copy_X=True, max_iter=None, tol=0.0001, solver='auto',
        positive=False, random_state=None)
```

Attributes	
coef_	**ndarray of shape (n_features,) or (n_targets, n_features)** Weight vector(s).
intercept_	**float or ndarray of shape (n_targets,)** Independent term in decision function. Set to 0.0 if fit_intercept = False.

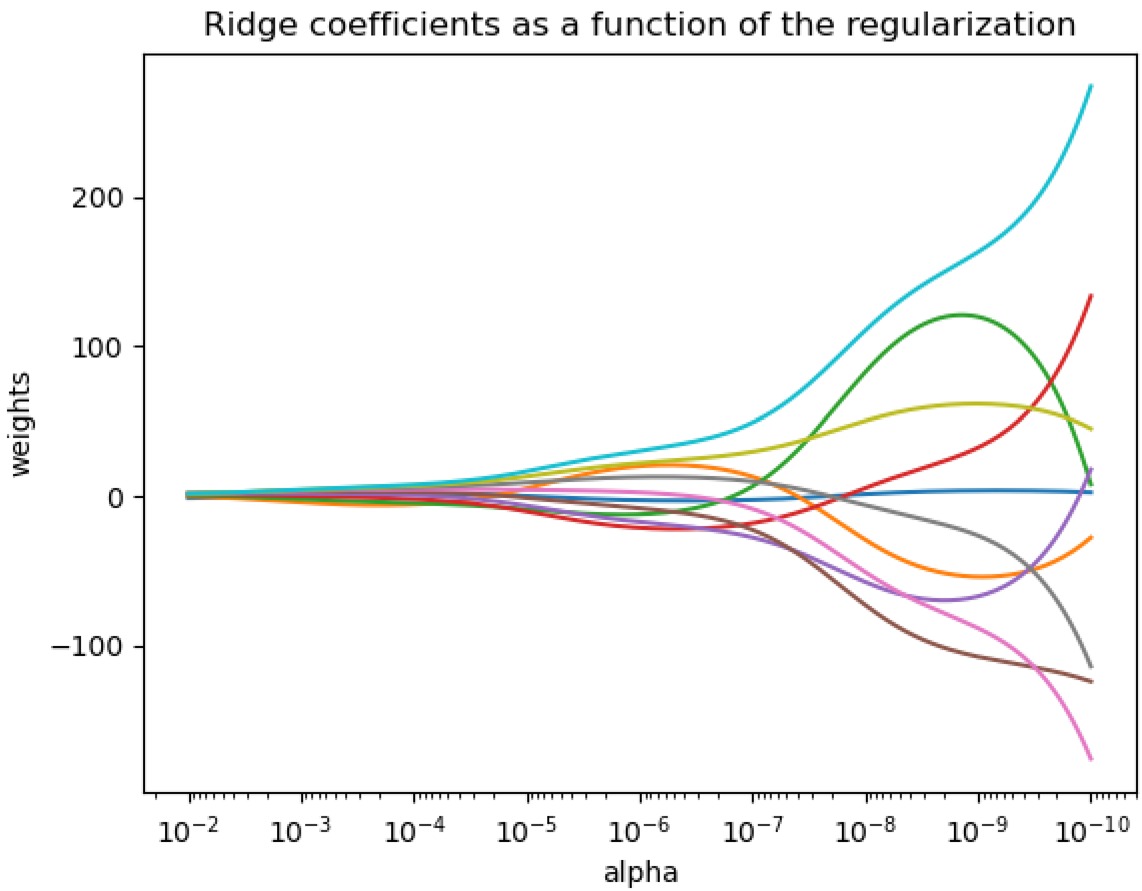
https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Ridge.html#sklearn.linear_model.Ridge

[Scikit-learn] Ridge

Example

```
from sklearn.linear_model import Ridge
import numpy as np
n_samples = 10
n_features = 5
rng = np.random.RandomState(0)
y = rng.randn(n_samples)
X = rng.randn(n_samples, n_features)
reg = Ridge(alpha=1.0)
reg.fit(X, y)
X_test = rng.randn(1, n_features)
reg.predict(np.array(X_test)
```

[Scikit-learn] Plot Ridge Coefficient



https://scikit-learn.org/stable/auto_examples/linear_model/plot_ridge_path.html

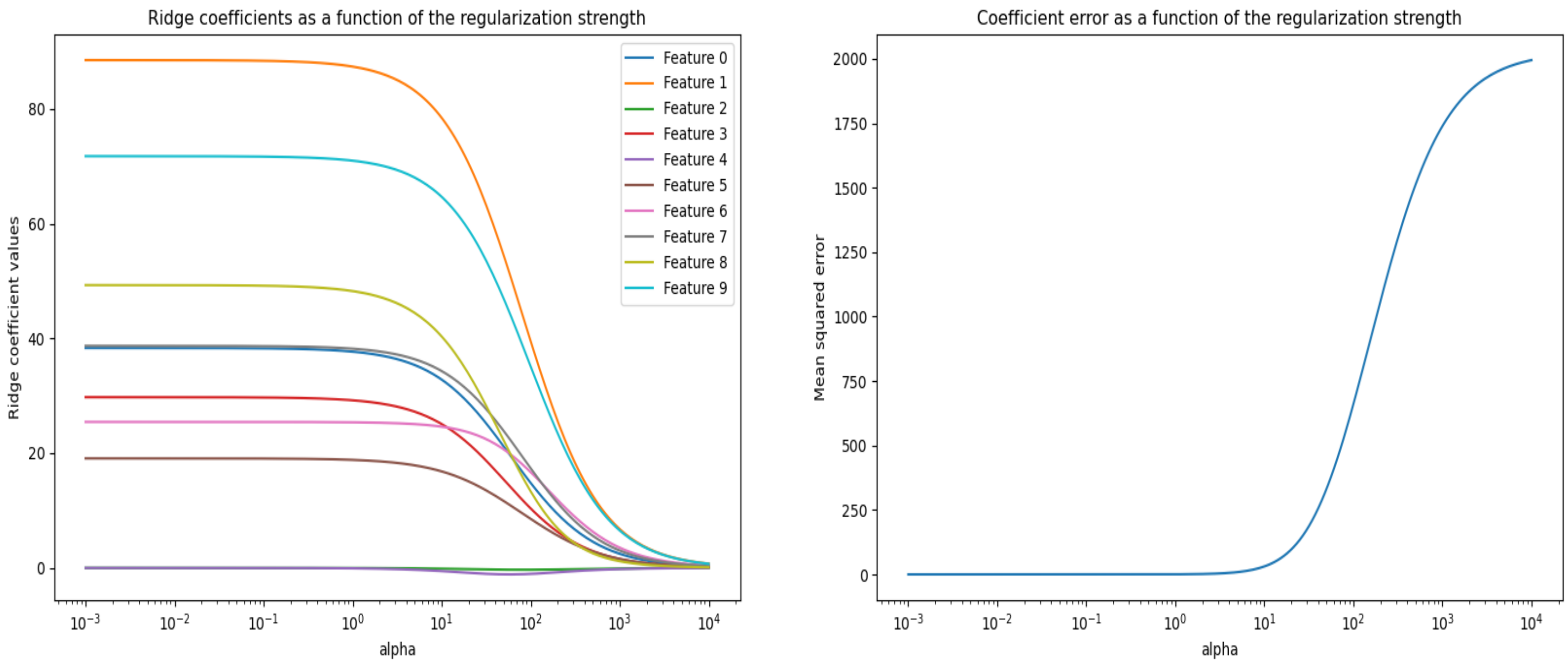
Effect of L2 Regularization

- A model that overfits learns the training data too well, capturing both the underlying patterns and the noise in the data
  - However, when applied to unseen data, the learned associations may not hold

- One way to overcome overfitting is through regularization, which can be done by penalizing large weights (coefficients) in linear models, forcing the model to shrink all coefficients.
  - The regularized loss function aims to balance the trade-off between accurately predicting the training set and to prevent overfitting

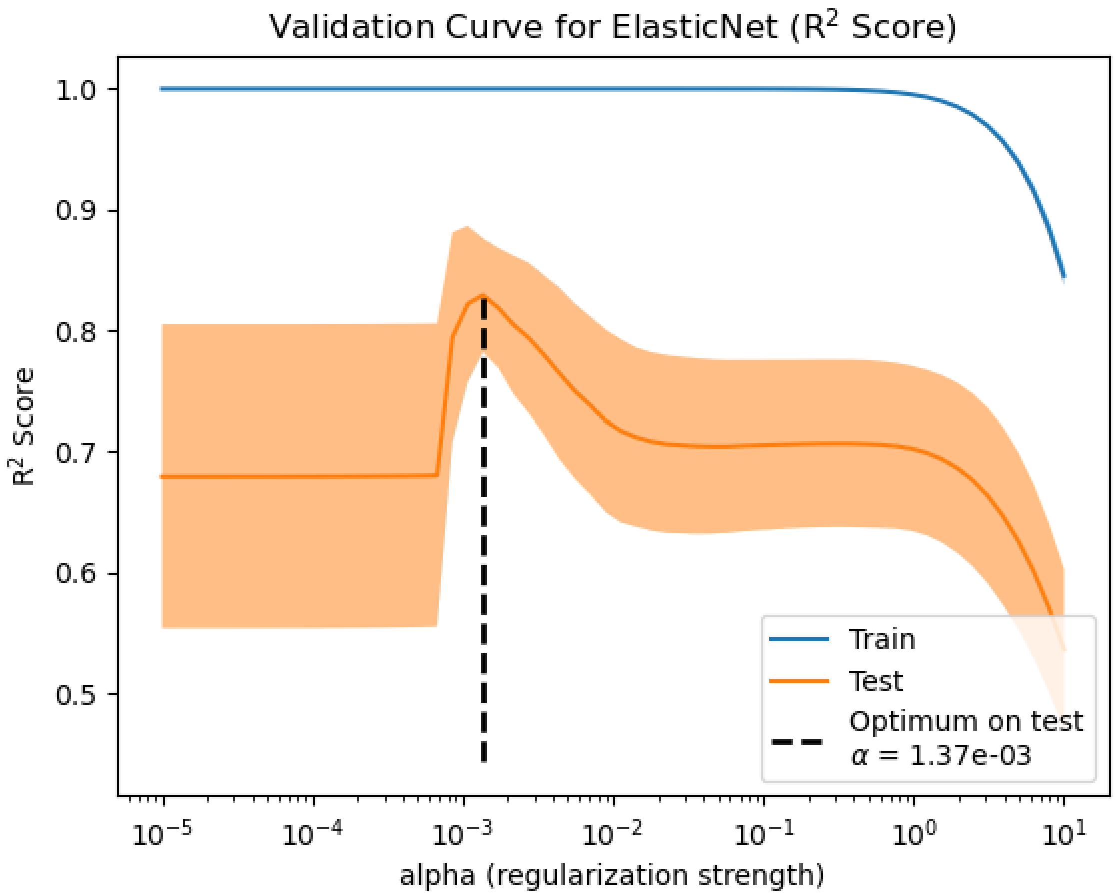
https://scikit-learn.org/stable/auto_examples/linear_model/plot_ridge_path.html

Effect of L2 Regularization



https://scikit-learn.org/stable/auto_examples/linear_model/plot_ridge_path.html

Regularization on training & test error



https://scikit-learn.org/stable/auto_examples/model_selection/plot_train_error_vs_test_error.html#sphx-glr-auto-examples-model-selection-plot-train-error-vs-test-error-py

Lasso

- The Lasso is a linear model that estimates sparse coefficients.

- It is useful in some contexts due to its tendency to prefer solutions with fewer non-zero coefficients, effectively reducing the number of features upon which the given solution is dependent.

- Mathematically, it consists of a linear model with an added $L1$ regularization term. The objective function to minimize is:

$$\min_{w} ||\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}||_2^2 + \alpha||\boldsymbol{w}||_1$$

[Scikit-learn] Lasso

```
class   sklearn.linear_model.Lasso(alpha=1.0, *, fit_intercept=True,
        precompute=False, copy_X=True, max_iter=1000, tol=0.0001,
        warm_start=False, positive=False, random_state=None,
        selection='cyclic')
```

Parameters	
alpha	**float, default=1.0** Constant that multiplies the L1 term, controlling regularization strength. alpha must be a non-negative float i.e. in [0, inf).
max_iter	**int, default=1000** The maximum number of iterations.
tol	**float, default=1e-4** The tolerance for the optimization: if the updates are smaller than tol, the optimization code checks the dual gap for optimality and continues until it is smaller than tol
random_state	**int, RandomState instance, default=None** The seed of the pseudo random number generator that selects a ra ndom feature to update. Pass an int for reproducible output acro ss multiple function calls

https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Lasso.html#sklearn.linear_model.Lasso

[Scikit-learn] Lasso

```
class   sklearn.linear_model.Lasso(alpha=1.0, *, fit_intercept=True,
        precompute=False, copy_X=True, max_iter=1000, tol=0.0001,
        warm_start=False, positive=False, random_state=None,
        selection='cyclic')
```

Attributes	
coef_	**ndarray of shape (n_features,) or (n_targets, n_features)** Parameter vector (w in the cost function formula).
intercept_	**float or ndarray of shape (n_targets,)** Independent term in decision function.

https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Lasso.html#sklearn.linear_model.Lasso

[Scikit-learn] Lasso

Example

```
from sklearn import linear_model
reg = linear_model.Lasso(alpha=0.1)
reg.fit([[0,0], [1, 1], [2, 2]], [0, 1, 2])
print(reg.coef_) # [0.85 0.  ]
print(reg.intercept_) # 0.15...
```

굳이 Lasso 까지 안하고 linear_model만 해도 됨.

Comparison of Linear Models

- OLS Linear Regression

$$\min_{w} \|\boldsymbol{Xw} - \boldsymbol{y}\|_2^2$$

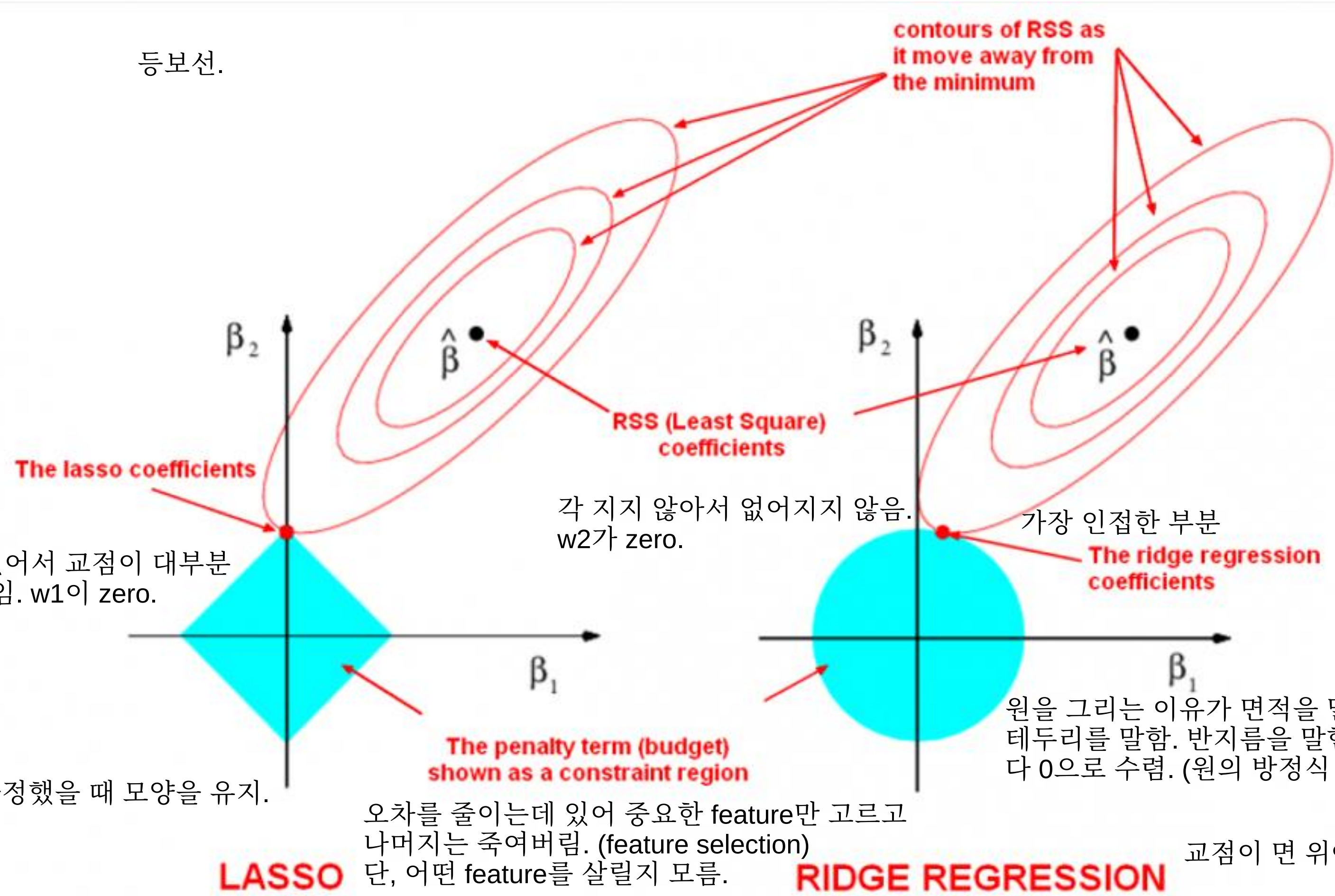
- Ridge Regression

$$\min_{w} \|\boldsymbol{Xw} - \boldsymbol{y}\|_2^2 + \alpha \|\boldsymbol{w}\|_2^2$$

- Lasso Regression

$$\min_{w} \|\boldsymbol{Xw} - \boldsymbol{y}\|_2^2 + \alpha \|\boldsymbol{w}\|_1$$

Ridge vs. Lasso Regualarization



등보선.

각 지지 않아서 없어지지 않음.
w2가 zero.

가장 인접한 부분

각 져 있어서 교점이 대부분
꼭짓점임. w1이 zero.

원을 그리는 이유가 면적을 말하는게 아닌
테두리를 말함. 반지름을 말함. 원의 직경.
다 0으로 수렴. (원의 방정식 이용)

노름의 크기를 가정했을 때 모양을 유지.

오차를 줄이는데 있어 중요한 feature만 고르고
나머지는 죽여버림. (feature selection)
단, 어떤 feature를 살릴지 모름.

교점이 면 위에서 만날거임.

https://www.linkedin.com/pulse/ridge-lasso-regularization-gurumaheswara-reddy

Elastic-Net

- ElasticNet is a linear regression model trained with both $L1$ and $L2$ regularization of the coefficients.

- This combination allows for learning a sparse model where few of the weights are non-zero like Lasso, while still maintaining the regularization properties of Ridge.

- The objective function to minimize is in this case:

$$\min_{w} \|\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}\|_2^2 + \alpha\rho\|\boldsymbol{w}\|_1 + \frac{\alpha(1-\rho)}{2}\|\boldsymbol{w}\|_2^2$$

Elastic-Net

- In Scikit-learn, an Elastic-net model control the convex combination of $L1$ and $L2$ using the l1_ratio parameter ρ.

- Elastic-net is useful when there are multiple features that are correlated with one another. Lasso is likely to pick one of these at random, while elastic-net is likely to pick both.

- A practical advantage of trading-off between Lasso and Ridge is that it allows Elastic-Net to inherit some of Ridge's stability under rotation.

한림대학교

[Scikit-learn] ElasticNet

```
class    sklearn.linear_model.ElasticNet(alpha=1.0, *, l1_ratio=0.5,
         fit_intercept=True, precompute=False, max_iter=1000, copy_X=True,
         tol=0.0001, warm_start=False, positive=False, random_state=None,
         selection='cyclic')
```

Parameters	
alpha	**float, default=1.0** Constant that multiplies the L1 term, controlling regularization strength. alpha must be a non-negative float i.e. in [0, inf).
l1_ratio	**float, default=0.5** The ElasticNet mixing parameter, with 0 <= l1_ratio <= 1. For l1_ratio = 0 the penalty is an L2 penalty. For l1_ratio = 1 it is an L1 penalty. For 0 < l1_ratio < 1, the penalty is a combination of L1 and L2.
max_iter	**int, default=1000** The maximum number of iterations.
tol	**float, default=1e-4** The tolerance for the optimization
random_state	**int, RandomState instance, default=None** The seed of the pseudo random number generator that selects a random feature to update. Pass an int for reproducible output across multiple function calls

https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.ElasticNet.html#sklearn.linear_model.ElasticNet

[Scikit-learn] ElasticNet

```
class   sklearn.linear_model.ElasticNet(alpha=1.0, *, l1_ratio=0.5,
        fit_intercept=True, precompute=False, max_iter=1000, copy_X=True,
        tol=0.0001, warm_start=False, positive=False, random_state=None,
        selection='cyclic')
```

Attributes	
coef_	**ndarray of shape (n_features,) or (n_targets, n_features)** Parameter vector (w in the cost function formula).
intercept_	**float or ndarray of shape (n_targets,)** Independent term in decision function.

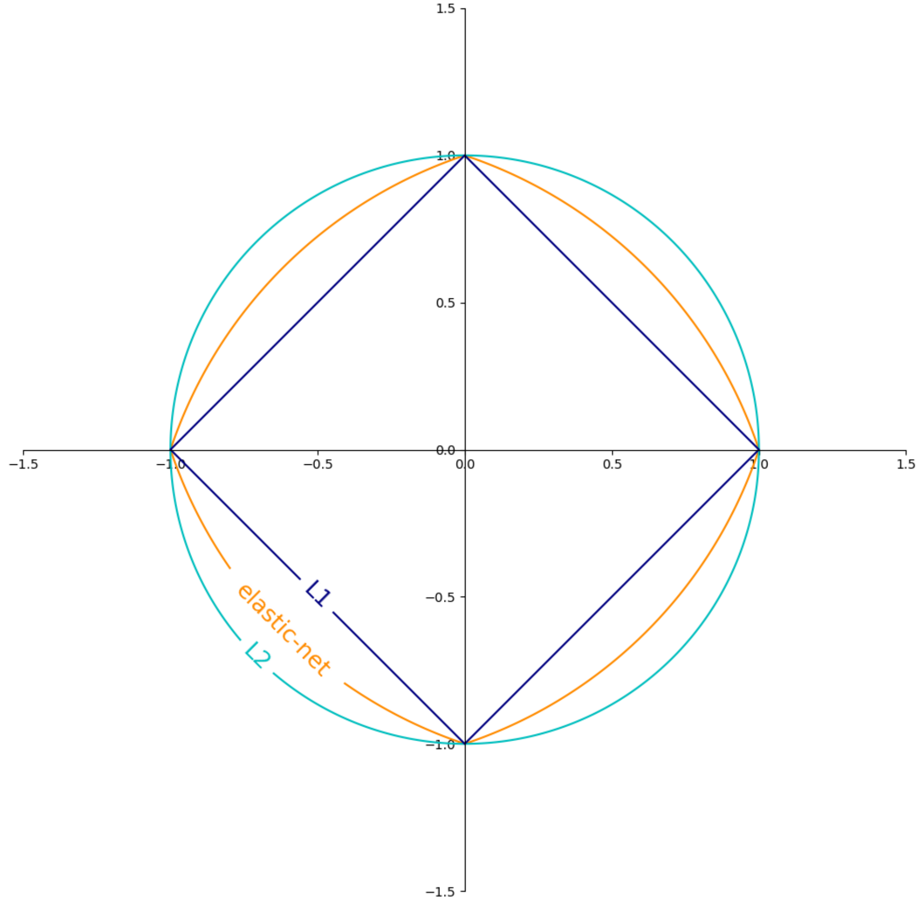
https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Lasso.html#sklearn.linear_model.Lasso

[Scikit-learn] ElasticNet

Example

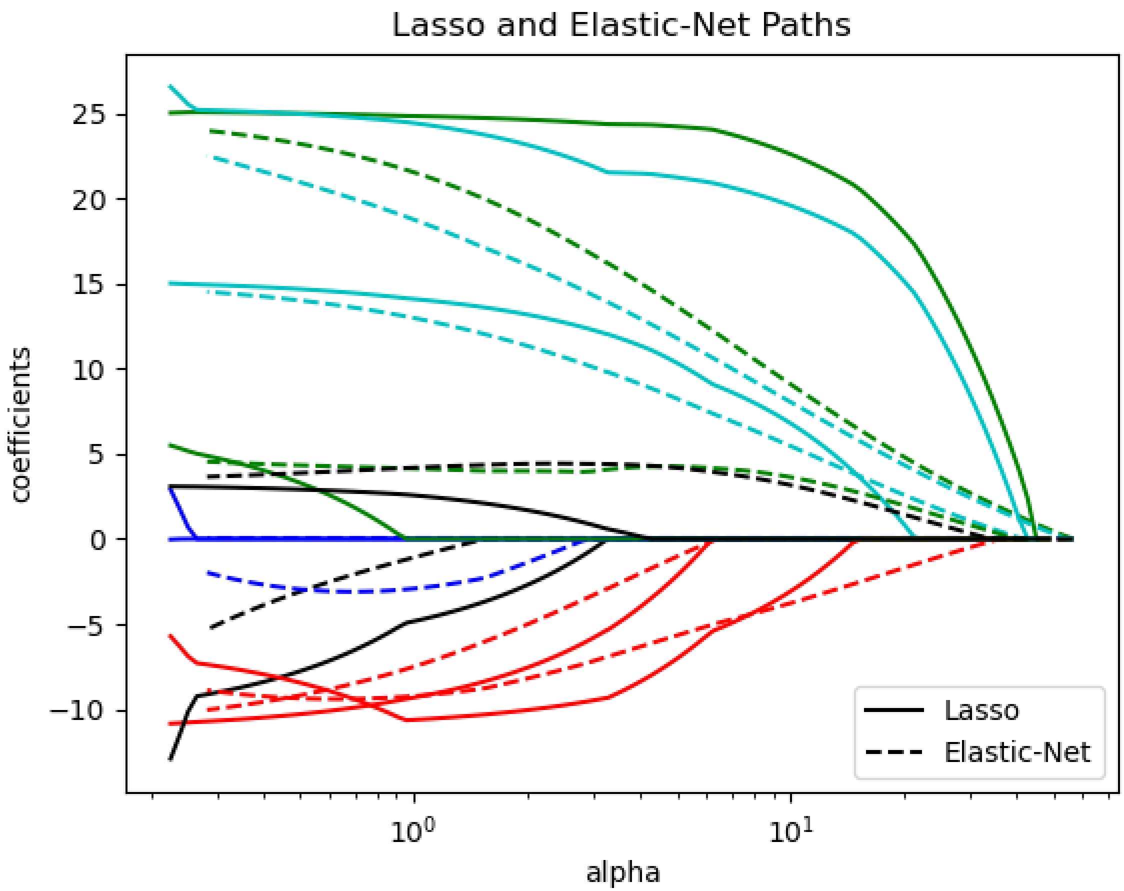
```
from sklearn.linear_model import ElasticNet
from sklearn.datasets import make_regression
X, y = make_regression(n_features=2, random_state=0)
regr = ElasticNet(random_state=0)
regr.fit(X, y)
print(regr.coef_) # [18.83816048 64.55968825]
print(regr.intercept_) # 1.451...
print(regr.predict([[0, 0]])) # [1.451...]
```

Comparison of Regularization terms



https://scikit-learn.org/stable/auto_examples/linear_model/plot_sgd_penalties.html#sphx-glr-auto-examples-linear-model-plot-sgd-penalties-py

Lasso vs. Elastic-Net



https://scikit-learn.org/stable/auto_examples/linear_model/plot_lasso_lasso_lars_elasticnet_path.html

Stochastic Gradient Descent

- Linear model fitted by minimizing a regularized empirical loss with stochastic gradient descent (SGD).
  - SGD is a simple yet very efficient approach to fit linear models.

- The regularizer is a penalty added to the loss function that shrinks model parameters towards the zero-vector using either the squared Euclidean norm L2 or the absolute norm L1 or a combination of both (Elastic Net)

- It is particularly useful when the number of samples (and the number of features) is very large.
  - The partial_fit method allows online/out-of-core learning.

https://scikit-learn.org/stable/modules/linear_model.html#stochastic-gradient-descent-sgd

[Scikit-learn] SGDRegressor

```
class    sklearn.linear_model.SGDRegressor(loss='squared_error', *,
         penalty='l2', alpha=0.0001, l1_ratio=0.15, fit_intercept=True,
         max_iter=1000, tol=0.001, shuffle=True, verbose=0, epsilon=0.1,
         random_state=None, learning_rate='invscaling', eta0=0.01,
         power_t=0.25, early_stopping=False, validation_fraction=0.1,
         n_iter_no_change=5, warm_start=False, average=False)
```

Parameters	
penalty	**{'l2', 'l1', 'elasticnet', None}, default='l2'** The penalty (aka regularization term) to be used. Defaults to 'l2' which is the standard regularizer for linear SVM models. 'l1' and 'elasticnet' might bring sparsity to the model not achievable with 'l2'. No penalty is added when set to None.
alpha	**float, default=0.0001** Constant that multiplies the regularization term. The higher the value, the stronger the regularization. Also used to compute the learning rate when learning_rate is set to 'optimal'. Values must be in the range [0.0, inf)
l1_ratio	**float, default=0.15** The Elastic Net mixing parameter, with 0 <= l1_ratio <= 1. l1_ratio=0 corresponds to L2 penalty, l1_ratio=1 to L1. Only used if penalty is 'elasticnet'. Values must be in the range [0.0, 1.0].

https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDRegressor.html#sklearn.linear_model.SGDRegressor

[Scikit-learn] SGDRegressor

```
class  sklearn.linear_model.SGDRegressor(loss='squared_error', *,
       penalty='l2', alpha=0.0001, l1_ratio=0.15, fit_intercept=True,
       max_iter=1000, tol=0.001, shuffle=True, verbose=0, epsilon=0.1,
       random_state=None, learning_rate='invscaling', eta0=0.01,
       power_t=0.25, early_stopping=False, validation_fraction=0.1,
       n_iter_no_change=5, warm_start=False, average=False)
```

Parameters	
max_iter	**float or None, default=1e-3** The stopping criterion. If it is not None, training will stop when (loss > best_loss - tol) for n_iter_no_change consecutive epochs. Convergence is checked against the training loss or the validation loss depending on the early_stopping parameter. Values must be in the range [0.0, inf).
random_state	**int, RandomState instance, default=None** Used for shuffling the data, when shuffle is set to True. Pass an int for reproducible output across multiple function calls.
eta0	**float, default=0.01** The initial learning rate for the 'constant', 'invscaling' or 'adaptive' schedules. The default value is 0.01. Values must be in the range [0.0, inf)

https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDRegressor.html#sklearn.linear_model.SGDRegressor

[Scikit-learn] SGDRegressor

```
class sklearn.linear_model.SGDRegressor(loss='squared_error', *,
      penalty='l2', alpha=0.0001, l1_ratio=0.15, fit_intercept=True,
      max_iter=1000, tol=0.001, shuffle=True, verbose=0, epsilon=0.1,
      random_state=None, learning_rate='invscaling', eta0=0.01,
      power_t=0.25, early_stopping=False, validation_fraction=0.1,
      n_iter_no_change=5, warm_start=False, average=False)
```

Parameters	
early_stopping	**bool, default=False** Whether to use early stopping to terminate training when validation score is not improving. If set to True, it will automatically set aside a fraction of training data as validation and terminate training when validation score returned by the score method is not improving by at least tol for n_iter_no_change consecutive epochs.
validation_fraction	**float, default=0.1** The proportion of training data to set aside as validation set for early stopping. Must be between 0 and 1. Only used if early_stopping is True. Values must be in the range (0.0, 1.0).

https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDRegressor.html#sklearn.linear_model.SGDRegressor

[Scikit-learn] SGDRegressor

```
class    sklearn.linear_model.SGDRegressor(loss='squared_error', *,
         penalty='l2', alpha=0.0001, l1_ratio=0.15, fit_intercept=True,
         max_iter=1000, tol=0.001, shuffle=True, verbose=0, epsilon=0.1,
         random_state=None, learning_rate='invscaling', eta0=0.01,
         power_t=0.25, early_stopping=False, validation_fraction=0.1,
         n_iter_no_change=5, warm_start=False, average=False)
```

Attributes	
coef_	**ndarray of shape (n_features,)** Weights assigned to the features.
intercept_	**ndarray of shape (1,)** The intercept term.

https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDRegressor.html#sklearn.linear_model.SGDRegressor

[Scikit-learn] ElasticNet

Example

```
import numpy as np
from sklearn.linear_model import SGDRegressor
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
n_samples = 10
n_features = 5
rng = np.random.RandomState(0)
y = rng.randn(n_samples)
X = rng.randn(n_samples, n_features)
reg = make_pipeline(StandardScaler(),
                    SGDRegressor(max_iter=1000, tol=1e-3))
reg.fit(X, y)
X_test = rng.randn(1, n_features)
reg.predict(np.array(X_test)
```

Exercise – Linear Models

- Scikit-learn의 Linear Models을 실습하기 위한 예시를 만들어주세요. LinearRegressor, Ridge, Lasso, ElasticNet, SGDRegressor를 비교하는 스크립트를 작성바랍니다.

1) 데이터집합 불러오기

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression, Ridge, Lasso, ElasticNet
from sklearn.linear_model import SGDRegressor
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.preprocessing import StandardScaler
## 예제 데이터집합
np.random.seed(0)
X = np.random.rand(100, 5)  # 100개의 샘플, 5개의 특징
y = 2*X[:, 0] + 3*X[:, 1] - X[:, 2] + np.random.randn(100) # 선형 관계 + 노이즈
```

Exercise – Linear Models

2) 데이터 분할

```
X_tr, X_te, y_tr, y_te = train_test_split(X, y, test_size=0.2, random_state=2025)
print("X_tr.shape:", X_tr.shape)
print("X_te.shape:", X_te.shape)
print("y_tr.shape:", y_tr.shape)
print("y_te.shape:", y_te.shape)
```

3) 데이터 정규화

```
scaler_X = StandardScaler().fit(X_tr)
X_norm_tr = scaler_X.transform(X_tr)
X_norm_te = scaler_X.transform(X_te)
```

Exercise – Linear Models

4) 모델 초기화

```python
models = {
    'Linear Regression': LinearRegression(),
    'Ridge': Ridge(alpha=1.0),
    'Lasso': Lasso(alpha=1.0),
    'Elastic Net': ElasticNet(alpha=1.0, l1_ratio=0.5),
    'SGD Regressor': SGDRegressor(max_iter=1000, tol=1e-3)
}
```

5) 모델 훈련

```python
for name, model in models.items():
    _ = model.fit(X_norm_tr, y_tr)
```

6) 모델 예측

```python
predictions = dict()
for name, model in models.items():
    predictions[name] = model.predict(X_norm_te)
```

한림대학교

Exercise – Linear Models

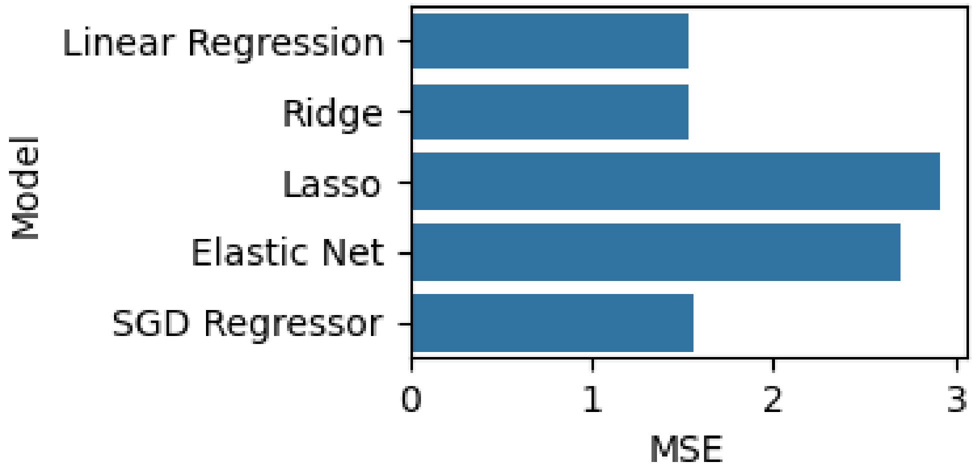
7) 모델 평가

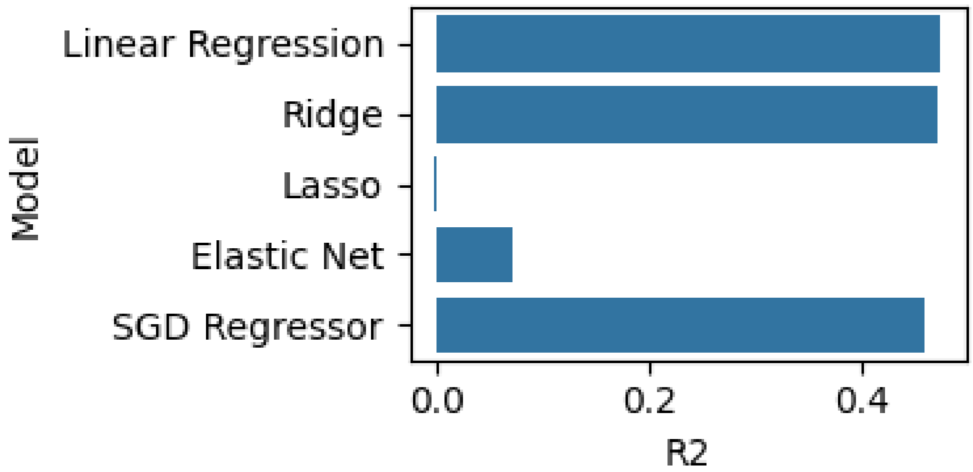
```
results = []
for name, p_te in predictions.items():
    mse = mean_squared_error(y_te, p_te)
    r2 = r2_score(y_te, p_te)
    results.append({'Model':name, 'MSE':mse, 'R2':r2})
df_res = pd.DataFrame(results)
print(df_res)
```

8) 결과 시각화

```
import matplotlib.pyplot as plt
import seaborn as sns
fig, ax = plt.subplots(2, 1, figsize=(4,4))
_ = sns.barplot(df_res, x='MSE', y='Model', ax=ax[0])
_ = sns.barplot(df_res, x='R2', y='Model', ax=ax[1])
plt.tight_layout()
plt.show()
```

Exercise – Linear Models





Q & A

Jonghwanc@hallym.ac.kr