

IIT Madras
ONLINE DEGREE

Machine Learning Practice
Indian Institute of Technology, Madras
Regularization

(Refer Slide Time: 0:12)



Regularization



Namaste, welcome to the next video of Machine Learning Practice course. In this video we will study how to implement regularization with sklearn.

(Refer Slide Time: 0:27)

How to perform ridge regularization with specific regularization rate?



[Option #1]

Step 1: Instantiate object of **Ridge** estimator

Step 2: Set parameter **alpha** to the required regularization rate.

```
1 from sklearn.linear_model import Ridge
2 ridge = Ridge(alpha=1e-3)
```

fit, **score**, **predict** work exactly like other linear regression estimators

[Option #2]

Step 1: Instantiate object of **SGDRegressor** estimator

Step 2: Set parameter **alpha** to the required regularization rate and **penalty = l2**.

```
1 from sklearn.linear_model import SGDRegressor
2 sgd = SGDRegressor(alpha=1e-3, penalty='l2')
```



There are two types of regularizations that we studied in machine learning techniques course ridge and lasso. Let us first see how to implement ridge regularization with sklearn, first we will

perform ridge regularization with specific regularization rate. There are two options to specify the ridge regularization in sklearn.

As a first option we can instantiate object of ridge estimator. Then we set parameter α to the required regularization rate. So, here I have a word of caution for you, so we have used α as a learning rate in machine learning techniques course. But in sklearn α is used as a regularization parameter.

In technique course we use λ as a regularization rate. So, that would be , one source of confusion so I would like you to be careful when you use sklearn and this α in sklearn corresponds to regularization rate when used as a parameter inside the ridge estimator. So, we can import the ridge class from sklearn.linear_model module. And we will instantiate the ridge object with value of α set to 0.0001.

And since ridge is an estimator fit, score and predict work exactly like any other linear regression estimator. As a second option we can instantiate object of SGDRegressor estimator. And in SGDRegressor estimator we set penalty to l2 and also set the parameter α to the required regularization rate.

(Refer Slide Time: 2:32)

How to search the best regularization parameter for ridge?

[Option #1]
Search for the best regularization rate with built-in cross validation in RidgeCV estimator.

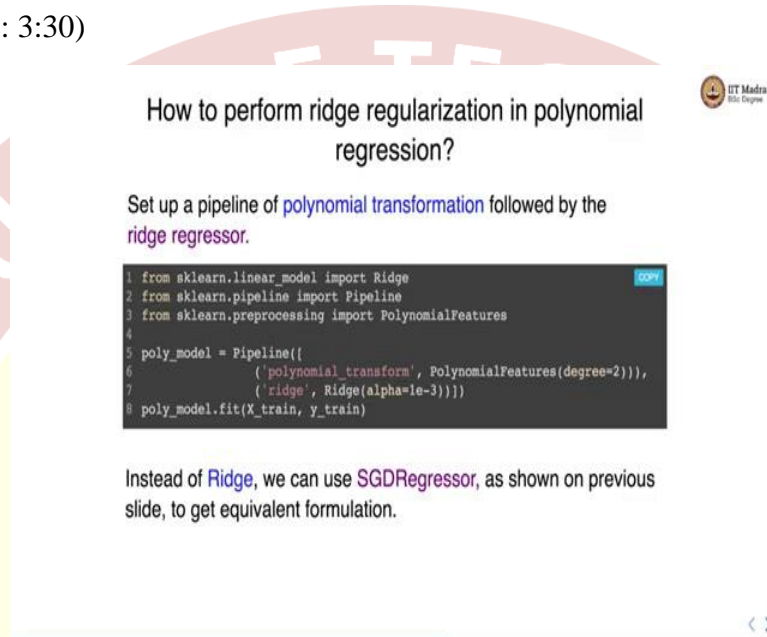
[Option #2]
Use cross validation with Ridge or SGDRegressor to search for best regularization.

- Grid search
- Randomized search

So, in the first illustration we set the α explicitly to a particular regularization rate but ideally, we would like to find α value automatically. So, let us try to understand how to search for the best regularization parameter for ridge regression. Again, there are two options, either we can use the

RidgeCV estimator that performs built-in cross validation and finds out the best regularization rate. And the second option is to use ridge or SVDR regressor estimators as defined in the previous slide and we can use hyper parameter tuning for searching the best regularization. We can either use grid search or randomized search for the search of α .

(Refer Slide Time: 3:30)



The slide is titled "How to perform ridge regularization in polynomial regression?". It includes a small logo for "IIT Madras" in the top right corner. The text on the slide says: "Set up a pipeline of polynomial transformation followed by the ridge regressor." Below this is a code block with the following Python code:

```
1 from sklearn.linear_model import Ridge
2 from sklearn.pipeline import Pipeline
3 from sklearn.preprocessing import PolynomialFeatures
4
5 poly_model = Pipeline([
6     ('polynomial transform', PolynomialFeatures(degree=2)),
7     ('ridge', Ridge(alpha=1e-3))]
8 poly_model.fit(X_train, y_train)
```

Below the code block, the text says: "Instead of Ridge, we can use SGDRegressor, as shown on previous slide, to get equivalent formulation." At the bottom right of the slide, there are navigation arrows: "< >".

So, let us study how to perform ridge regularization in polynomial regression. Here we set up a pipeline of polynomial transformation followed by the ridge regressor. So, we have pipeline where we have polynomial transformation as instantiated with polynomial feature object with degree equal to 2 followed by ridge with the specified value of the parameter α .

And then we can simply call fit function on the pipeline object with the feature matrix and labels as arguments. So, here we will perform first the polynomial transformation and then we will use the ridge regression estimator for learning the parameters of the linear model. So, instead of ridge we can also use SGD Regressor as shown in a couple of slides before to get equivalent formulation.

(Refer Slide Time: 4:40)

How to perform lasso regularization with specific regularization rate?



[Option #1]

Step 1: Instantiate object of `Lasso` estimator

Step 2: Set parameter `alpha` to the required regularization rate.

```
1 from sklearn.linear_model import Lasso
2 lasso = Lasso(alpha=1e-3)
```

`fit`, `score`, `predict` work exactly like other linear regression estimators

[Option #2]

Step 1: Instantiate object of `SGDRegressor` estimator

Step 2: Set parameter `alpha` to the required regularization rate and `penalty = l1`.

```
1 from sklearn.linear_model import SGDRegressor
2 sgd = SGDRegressor(alpha=1e-3, penalty='l1')
```



So, that was about ridge, let us try to understand how to perform lasso regularization with specific regularization rate. Again, there are two options either we can use lasso estimator and set the parameter α to the required regularization rate lasso class is implemented as part of `sklearn.linear_model`. In option 2, we can instantiate an object of `SGDRegressor` estimator and set the parameter α to the required regularization rate and penalty to `l1`.

(Refer Slide Time: 5:20)

How to search the best regularization parameter for lasso regularization?



[Option #1]

Search for the best regularization rate with built-in cross validation in `LassoCV` estimator.

[Option #2]

Use cross validation with `Lasso` or `SVDRegressor` to search for best regularization.

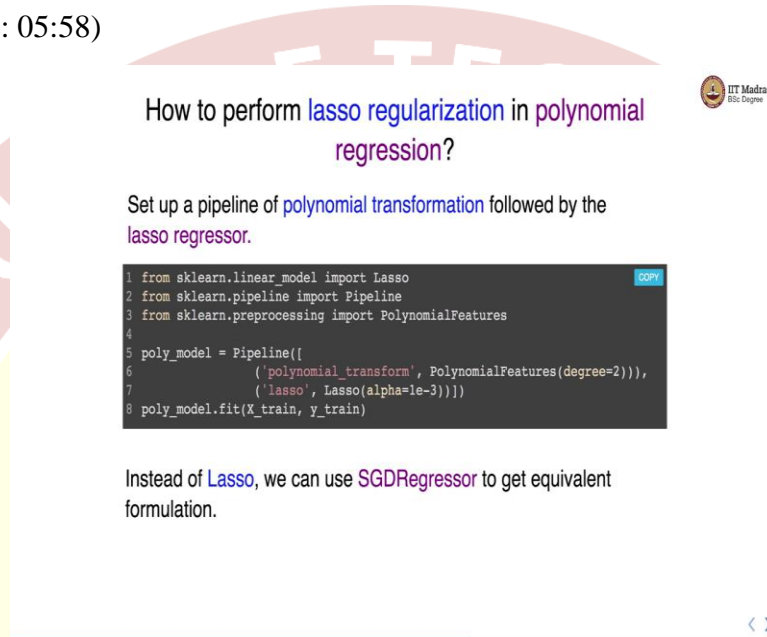
- Grid search
- Randomized search



Again, similar to ridge we also need to search for the best regularization parameter for lasso regularization. There are two options we can either use `LassoCV` estimator that has built in cross

validation for finding the best regularization rate. And second is to use cross validation with lasso and SVDR regressor to search for the best regularization. We can use couple of hyper parameter search methods like grid search or randomized search for finding the best regularization parameter.

(Refer Slide Time: 05:58)



How to perform lasso regularization in polynomial regression?

Set up a pipeline of polynomial transformation followed by the lasso regressor.

```
1 from sklearn.linear_model import Lasso
2 from sklearn.pipeline import Pipeline
3 from sklearn.preprocessing import PolynomialFeatures
4
5 poly_model = Pipeline([
6     ('polynomial_transform', PolynomialFeatures(degree=2)),
7     ('lasso', Lasso(alpha=1e-3))]
8 poly_model.fit(X_train, y_train)
```

Instead of Lasso, we can use SGDRegressor to get equivalent formulation.

IT Madras
IIT Madras
800 201 5800

Let us see how to perform lasso regularization in polynomial regression. So, just like ridge here also we set up a pipeline of polynomial transformation followed by a lasso regressor. Instead of lasso we can also use SGDRegressor to get equivalent formulation. You can see that here we have instantiated an object of a pipeline class with polynomial transformation specified by instantiating object of polynomial features and, then we have lasso as a second step which is specified by constructing an object of lasso class we specified regularization rate.

(Refer Slide Time: 6:50)

How to perform both lasso and ridge regularization in polynomial regression?



Set up a pipeline of polynomial transformation followed by the SGDRegressor with `penalty = 'elasticnet'`

```
1 from sklearn.linear_model import Lasso
2 from sklearn.pipeline import Pipeline
3 from sklearn.preprocessing import PolynomialFeatures
4
5 poly_model = Pipeline([
6     ('polynomial_transform', PolynomialFeatures(degree=2)),
7     ('elasticnet', SGDRegressor(penalty='elasticnet',
8                               l1_ratio=0.3))]
9 poly_model.fit(X_train, y_train)
```

Remember `elasticnet` is a convex combination of L1 (Lasso) and L2 (Ridge) regularization.

- In this example, we have set `l1_ratio` to 0.3, which means `l2_ratio = 1 - l1_ratio = 0.7`. L2 takes higher weightage in this formulation.

< >

So, apart from using ridge and lasso separately we can also use what is called as elastic net, elastic net regularization where we can use both lasso and ridge regularization together, let us see how to do that. So, we can set up a pipeline of polynomial transformation followed by SGDRegressor with penalty equal to elastic net. So, here we have SGDRegressor object with penalty equal to elastic net and we set the `l1_ratio` to 0.3.

So, elastic net is basically a convex combination of l_1 and l_2 regularization. So, in this case we have set the value of `l1_ratio` to 0.3 which means `l2_ratio` will be $1 - l1_ratio$ and in this case it will be 0.7. So, l_2 or the ridge regression takes higher weightage in this formulation. So, we get the regularization that is combination of l_1 and l_2 .

In case we set `l1_ratio` to 1 we get the formulation that is equivalent to lasso and if you set `l1_ratio` equal to 0 we get formulation that is equivalent to the ridge regression when we set the penalty equal to elastic net. That is the end of this module.

(Refer Slide Time: 8:14)



Summary

How to implement

- Different regression models: standard linear regression and polynomial regression.
- Regularization.
- Model evaluation through different error metrics and scores derived from them.
- Cross validation - different iterators
- Hyperparameter tuning via grid search and randomized search.



In summary we learned how to implement different regression models like standard linear regression and polynomial regression with sklearn. We also learned how to specify the regularization. We learnt how to evaluate models through different error metrics and scores derived from the error matrix. Then we also studied cross validation with different iterators and hyper parameter tuning via grid and randomized search.

