06.05 정규화 선형회귀

정규화 선형회귀 방법

선형회귀 계수(weight)에 대한 제약조건 추가한 최적화 -> 과최적화 막음 (모델을 조금 더 부드럽게)

1) Ridge 회귀모형

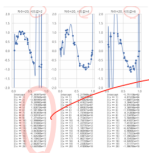
2) Lasso 회귀모형

3) Elastic Net 회귀모형

각각 하이퍼모수 '람다'(페널티)로 두고, w에 제한을 둠

*페널티가 클 수록, 정규화정도가 커짐 (1page)

- Ridge 회귀모형
- Lasso 회귀모형
- Elastic Net 회귀모형



과최적화시, 1. 임계값이 조금만 변해도 가중치값(모형)이 크게 변함
2. 변화는 없고 매우 큰 숫자 끼리 공존한다.
⇒ 가중치에 제한조건 주어서 (정규화)

Ridge 회귀모형

Ridge 회귀모형에서는 가중치들의 제곱합(squared sum of weights)을 최소화하는 것을 추가적인 제약 조건으로 한다.

$$w = \arg\min_w \left(\sum_{i=1}^{N} e_i^2 + \lambda \sum_{j=1}^{M} w_j^2 \right)$$

λ는 기존의 잔차 제곱합과 추가적 제약 조건의 비중을 조절하기 위한 하이퍼 모수(hyper parameter)이다. λ가 크면 정규화 정도가 커지고 가중치의 값들이 작아진다. λ가 작아지면 정규화 정도가 작아지며 λ 가 0이 되면 일반적인 선형 회귀모형이 된다.

하이퍼모수 λ ↑ ⟺ 정규화 정도 ↑ (= 페널티 높)

크기를 제한

Lasso 회귀모형

Lasso(Least Absolute Shrinkage and Selection Operator) 회귀모형은 가중치의 절대값의 합을 최소화하는 것을 추가적인 제약 조건으로 한다.

$$w = \arg\min_w \left(\sum_{i=1}^{N} e_i^2 + \lambda \sum_{j=1}^{M} |w_j| \right)$$

Elastic Net 회귀모형

Elastic Net 회귀모형은 가중치의 절대값의 합과 제곱합을 동시에 제약 조건으로 가지는 모형이다.

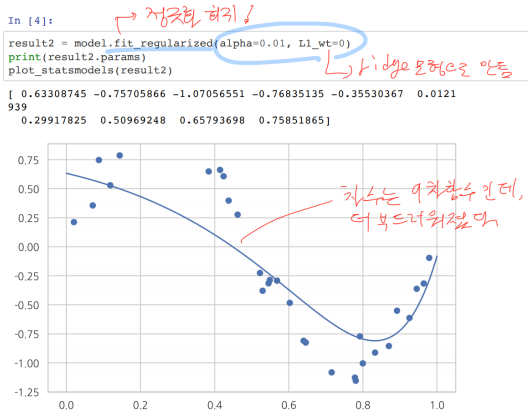
$$w = \arg\min_w \left(\sum_{i=1}^{N} e_i^2 + \lambda_1 \sum_{j=1}^{M} |w_j| + \lambda_2 \sum_{j=1}^{M} w_j^2 \right)$$

λ_1, λ_2 두 개의 하이퍼 모수를 가진다.

둘 다 사용!

1. statsmodels의 정규화 회귀모형

```
1) Lasso : L1_wt = 1
2) Ridge : L1_wt = 0
3) Elastic net : L1_wt = 0과 1사이
```



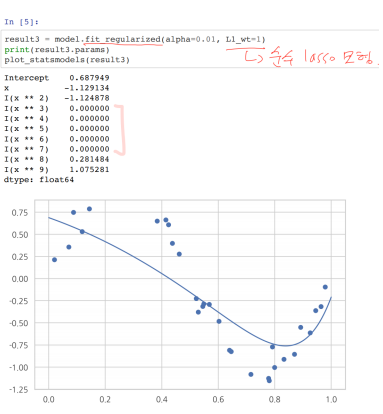
```
In [4]:
result2 = model.fit_regularized(alpha=0.01, L1_wt=0)
print(result2.params)
plot_statsmodels(result2)

[ 0.63308745 -0.75705866 -1.07056551 -0.76835135 -0.35530367  0.0121
939
  0.29917825  0.50969248  0.65793698  0.75851865]
```

정규화 해지!

릿지 모형으로 만듦

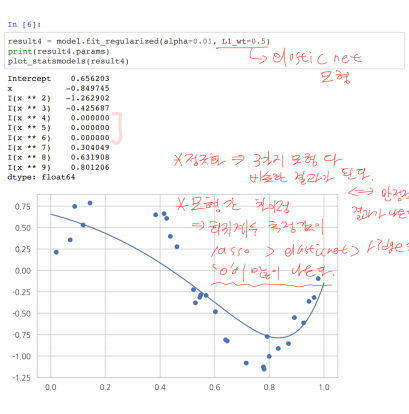
차수는 9차함수 인데, 더 부드러워졌다.



```
In [5]:
result3 = model.fit_regularized(alpha=0.01, L1_wt=1)
print(result3.params)
plot_statsmodels(result3)

Intercept    0.687949
x           -1.129134
I(x ** 2)   -1.124878
I(x ** 3)    0.000000
I(x ** 4)    0.000000
I(x ** 5)    0.000000
I(x ** 6)    0.000000
I(x ** 7)    0.000000
I(x ** 8)    0.281484
I(x ** 9)    1.075281
dtype: float64
```

순수 lasso 모형



```
In [6]:
result4 = model.fit_regularized(alpha=0.01, L1_wt=0.5)
print(result4.params)
plot_statsmodels(result4)

Intercept    0.656203
x           -0.849745
I(x ** 2)   -1.262902
I(x ** 3)   -0.425687
I(x ** 4)    0.000000
I(x ** 5)    0.000000
I(x ** 6)    0.000000
I(x ** 7)    0.304049
I(x ** 8)    0.631908
I(x ** 9)    0.801206
dtype: float64
```

elastic net 모형

※정규화 ⇒ 3가지 모형 다 비슷한 결과가 된다.

※모형간 차이 ⇒ 회귀계수 추정값이 lasso > elastic net > ridge 순으로 '0'이 많음

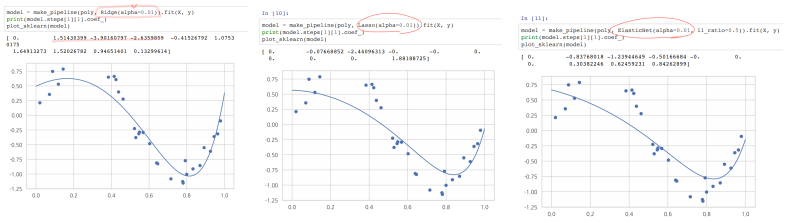
⟺ 안정적인 결과가 나온다.

3가지 모델의
공통점 : 회귀 선의 추세는 비슷한 모습
차이점 : 회귀계수 추정값이 Lasso > Elastic net > Ridge 순으로 '0'이 많음

*Lasso 는 정규화 정도가 클수록(람다 클수록) 0의 숫자가 많아짐 (Lasso path)

2. sklearn의 정규화 회귀모형

- Ridge, Lasso, ElasticNet 이라는 별도의 클래스 사용



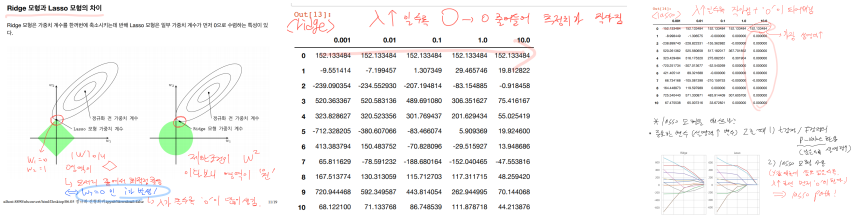
3. 정규화 모형의 장점, 의미, Lasso - Ridge 차이

1) 정규화 모형 장점 : 안정적 모형 <=> 다중공선성 -> 조건수 커짐 -> 데이터가 조금만 커져도 추정값 변화 큼(불안정성)

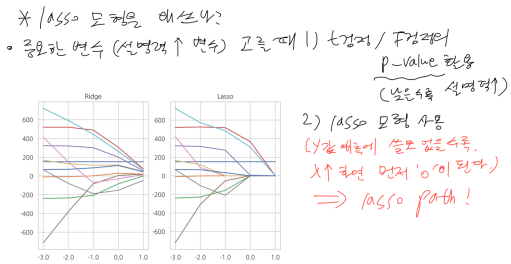
2) 정규화의 의미 : 정규화가 없는 최적화 문제에 부등식 제한 조건을 추가하는 것

3) Ridge - Lasso 차이 (11page - 13page)

: Lasso는 일부 가중치 계수를 먼저 0으로 수렴시키며 정규화 정도를 높여감 (ridge는 가중치 계수를 전부다 조금씩 축소하면서 정규화 진행)

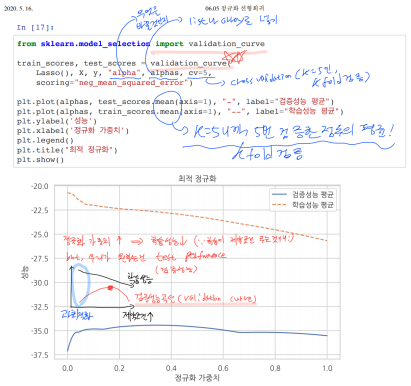


4. Lasso 모형의 쓸모 : 중요하지 않은 변수 먼저 0으로 수렴한다 (Lasso Path)



5. Validation-Curve : 그렇다면, 페널티는 얼마를 줘야 검증성능이 제일 높을까?

- 최적 정규화
- validation curve 를 그려 최적 람다 찾기
  *사실, 람다마다 test-performance를 일일이 찾아서 그래프를 그린 것 (18page)



6. Validation-Curve : 다항회귀의 최적 차수 결정

- 정규화 : 람다가 클수록 정규화 정도가 높음(제약이 큰 것)
- 다항회귀 : 차수가 클수록 정규화 정도가 낮음(제약이 낮은 것. 사용 가능한 계수가 많아짐) (18page)

다항회귀의 차수 결정

다항회귀에서 차수가 감소하면 모형의 제약조건이 더 강화되므로 정규화 가중치가 커지는 것과 같다. 반대로 차수가 증가하면 모형의 제약조건이 감소하므로 정규화 가중치가 작아지는 것과 같다. 따라서 다항회귀에서 최적의 차수를 결정하는 문제는 최적 정규화에 해당한다.

다음 예제 코드는 파이프라인이 있는 모형에 대해 `validation_curve` 명령을 적용하는 방법을 보이고 있다. 파이프라인으로 만들어진 모형에서는 적용할 모형의 이름 문자열(이 예제에서는 `poly`)과 인수의 이름 문자열(이 예제에서는 `degree`)을 두 개의 밑줄(underscore)로 연결한다.