Modelos Lineares Regularizados

Slides extraídos do livro "*Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*" de Aurélien Géron

Sumário

Sumário

Modelos Lineares Regularizados

- Como vimos nos capítulos anteriores, uma boa maneira de reduzir o **overfitting** é regularizar o modelo (ou seja, **restringi-lo**).
- Para um modelo **polinomial**, uma maneira simples de regularizar é reduzir o número de **graus** polinomiais.
- Para um modelo **linear** (incluindo a Regressão Polinomial), a regularização é tipicamente alcançada restringindo os **pesos** do modelo.

Veremos agora a Regressão **Ridge**, **Lasso** e **Elastic Net**, que implementam três maneiras diferentes de restringir os pesos.

Sumário

Regressão Ridge (também chamada de regularização de Tikhonov) é versão regularizada da Regressão Linear.

Equação 4-8. Função de custo da Regressão Ridge

$$J(\theta) = \text{MSE}(\theta) + \frac{\alpha}{m} \sum_{i=1}^{n} \theta_i^2$$

- Um termo de regularização é adicionado ao MSE. Isso força o algoritmo de aprendizado a não apenas ajustar os dados, mas também a manter os pesos do modelo o menor possível.
- O hiperparâmetro α controla o quanto você quer regularizar o modelo.
  Se $\alpha = 0$, então a Regressão Ridge é apenas Regressão Linear.
- O termo de viés θ_0 não é regularizado (a soma começa em $i = 1$).
- **Importante**: É crucial escalar os dados antes de realizar a Regressão Ridge, pois ela é sensível à escala das features de entrada.

Efeito da Regularização Ridge

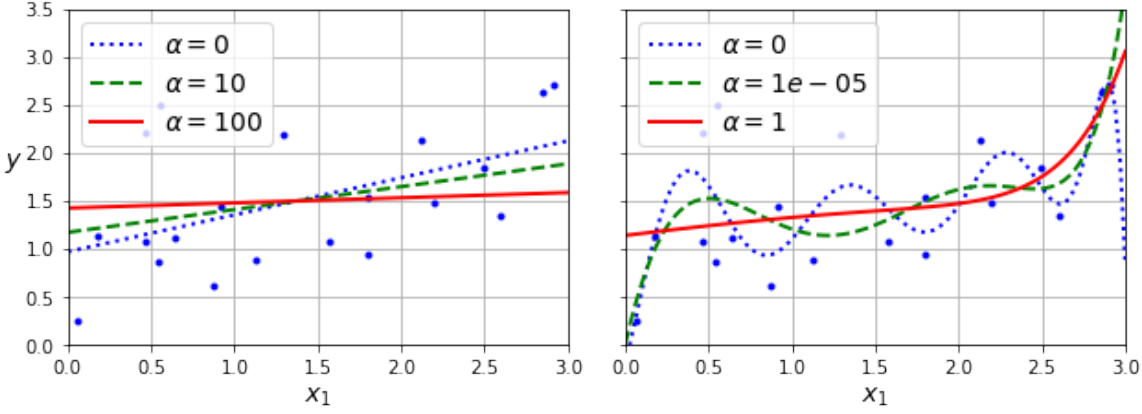

Figure: Um modelo linear (esquerda) e um modelo polinomial (direita), ambos com vários níveis de regularização Ridge.

- Aumentar α leva a previsões mais "planas" (ou seja, menos extremas, mais razoáveis).
- Isso reduz a variância do modelo, mas aumenta seu viés.

Implementando a Regressão Ridge

A Regressão Ridge pode ser feita computando uma equação de forma fechada ou por Gradiente Descendente.

Equação 4-9. Solução de forma fechada da Regressão Ridge

$$\hat{\theta} = (X^T X + \alpha I)^{-1} X^T y, \text{ onde } I \text{ é matriz identidade } (n+1) \times (n+1)$$

Com solução de forma fechada:

```
from sklearn.linear_model import Ridge
ridge_reg = Ridge(alpha=0.1, solver="cholesky")
ridge_reg.fit(X, y)
>>> ridge_reg.predict([[1.5]])
array([[1.55325833]])
```

Com Gradiente Descendente Estocástico:

```
sgd_reg = SGDRegressor(penalty="l2", alpha=0.1/m,
                                    ...)
sgd_reg.fit(X, y)
>>> sgd_reg.predict([[1.5]])
array([1.55302613])
```

O hiperparâmetro `penalty` define o tipo de termo de regularização a ser usado. `"l2"` indica que você deseja adicionar um termo de regularização igual ao alpha vezes o quadrado da norma ℓ_2 do vetor de pesos.

Sumário

Regressão Lasso

A Regressão **Lasso** (Least Absolute Shrinkage and Selection Operator) é outra versão regularizada da Reg. Linear.

Equação 4-10. Função de custo da Regressão Lasso

$$J(\theta) = \text{MSE}(\theta) + 2\alpha \sum_{i=1}^{n} |\theta_i|$$

- Usa a norma ℓ_1 do vetor de pesos em vez do quadrado da norma ℓ_2.

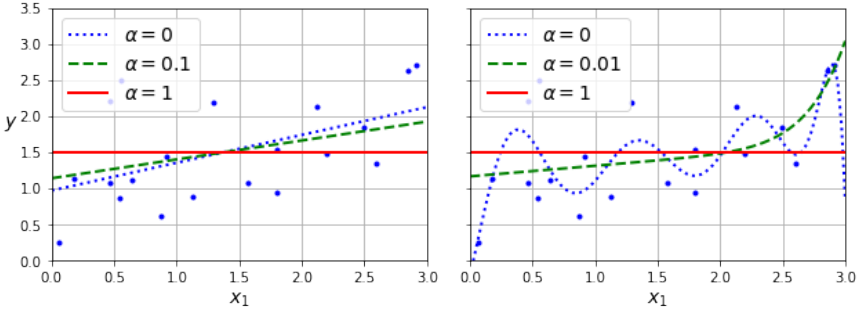

Figure: Um modelo linear (esquerda) e um modelo polinomial (direita), ambos com vários níveis de regularização Lasso.

Característica da Regressão Lasso

A Regressão Lasso tende a zerar os pesos dos atributos menos importantes (**seleção de features**).

- Na ℓ_2, o movimento de redução da penalidade é na direção do mínimo (derivadas proporcionais às features).
- Na ℓ_1, a redução da penalidade altera todas as features na mesma quantidade (em módulo) - move 45^o.
  - Feature é considerada só até atingir valor zero (subgradiente assume derivada nula no valor zero).

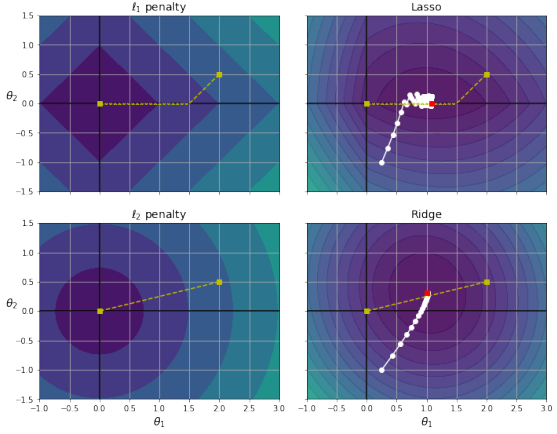

Figure: Extremos no segmento amarelo: (0,0) e solução ótima do MSE

Implementando a Regressão Lasso

Um pequeno exemplo no Scikit-Learn usando a classe `Lasso`.

```
from sklearn.linear_model import Lasso
lasso_reg = Lasso(alpha=0.1)
lasso_reg.fit(X, y)
>>> lasso_reg.predict([[1.5]])
array([1.53788174])
```

Nota

Você também poderia usar `SGDRegressor(penalty="l1", alpha=0.1)`.
Não há solução de forma fechada (como no Ridge), pois a penalidade tem discontinuidade na derivada, impossibilitando solução analítica (é um problema de otimização combinatória).

Sumário

Elastic Net

Elastic Net é um meio-termo entre a Regressão Ridge e a Regressão Lasso.
O termo de regularização é uma soma ponderada dos termos de regularização de Ridge e Lasso.

Equação 4-12. Função de custo da Elastic Net

$$J(\theta) = \text{MSE}(\theta) + r\left(2\alpha\sum_{i=1}^{n}|\theta_i|\right) + (1-r)\left(\frac{\alpha}{m}\sum_{i=1}^{n}\theta_i^2\right)$$

Você pode controlar a razão de mistura r.
Quando $r = 0$, Elastic Net é equivalente à Regressão Ridge, e quando $r = 1$, é equivalente à Regressão Lasso.

```
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([1.54333232])
```

Quando usar qual?

- É quase sempre preferível ter pelo menos um pouco de regularização.
- Ridge é um bom padrão.
- Se você suspeita que apenas algumas features são úteis, prefira Lasso ou Elastic Net.
- Em geral, Elastic Net é preferível ao Lasso, pois o Lasso pode se comportar erraticamente quando o número de features é maior que o número de instâncias ou quando várias features são fortemente correlacionadas.

Sumário

Parada Antecipada (Early Stopping)

- Uma maneira muito diferente de regularizar algoritmos de aprendizado iterativos, como o Gradiente Descendente, é parar o treinamento assim que o erro de validação atinge um mínimo (**parada antecipada**).
- É uma técnica de regularização simples e eficiente.

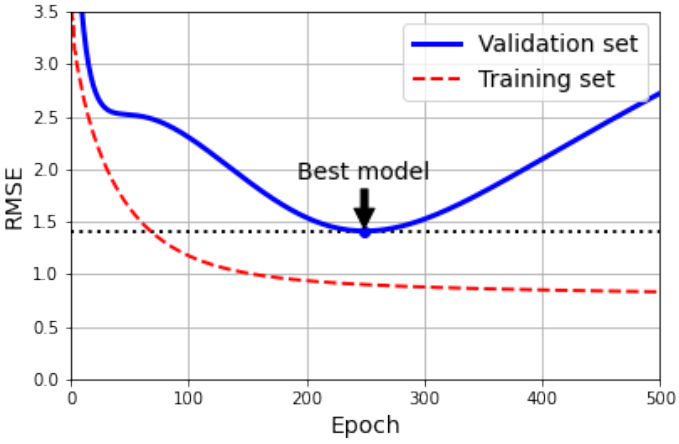

Figure: Regularização por parada antecipada.

Implementação Básica de Parada Antecipada

```
from copy import deepcopy

X_train, y_train, X_val, y_val = [...]

sgd_reg = SGDRegressor(penalty=None, eta0=0.002, random_state=42)
n_epochs = 500
best_val_rmse = float('inf')

for epoch in range(n_epochs):
    sgd_reg.partial_fit(X_train, y_train)
    y_val_predict = sgd_reg.predict(X_val)
    val_error = mean_squared_error(y_val, y_val_predict, squared=False)
    if val_error < best_val_rmse:
        best_val_rmse = val_error
        best_model = deepcopy(sgd_reg)
```

Dica para SGD e Mini-batch GD

Com Stochastic e Mini-batch GD, as curvas não são tão suaves. Uma solução é parar somente depois que o erro de validação estiver acima do mínimo por algum tempo e, em seguida, reverter os parâmetros do modelo para o ponto onde o erro de validação estava no mínimo.

Implementação Básica de Parada Antecipada

`partial_fit` é usado para treinar o modelo em uma única época (uma passagem pelos dados de treinamento).

`SGDRegressor` suporta parada antecipada nativamente:

```
sgd_reg = SGDRegressor(penalty=None, early_stopping=True, validation_fraction=0.1, n_iter_no_change=5)
```

- `early_stopping=True` ativa a parada antecipada.
- `validation_fraction=0.1` significa que 10% dos dados de treinamento serão usados para validação interna do early stopping.
- `n_iter_no_change=5` significa que o treinamento será interrompido se o erro de validação não melhorar por 5 épocas consecutivas.

Fim