REGULARIZAÇÃO, BALANCEANDO BIAS(VIÉS) E VARIÂNCIA.

Vamos agora tratar da Regularização para contornarmos o sobreajuste de um modelo de machine learning referenciado na publicação anterior:
https://www.linkedin.com/posts/activity-7242607660968165376-fNIF?utm_source=share&utm_medium=member_desktop

Pelos gráficos da Figura 01 mostramos no primeiro gráfico da esquerda para a direita uma regressão linear onde é possível perceber que a função inclui poucos pontos da amostra na função (havendo alto bias-viés e baixa variância), ficando vários pontos fora da reta da função, ocorrendo aí um underfitting.
Vimos também que o terceiro gráfico da esquerda para a direita, da Figura 01, mostra que a curva da função polinomial passou por praticamente todos os pontos ocasionando um sobreajuste (havendo alta variância e baixo bias-viés) fazendo com que o modelo não generalize bem.
O ideal como visto na publicação anterior é haver um balanceamento entre bias e variância, quando aumentamos a bias diminuímos a variância e vice-versa.

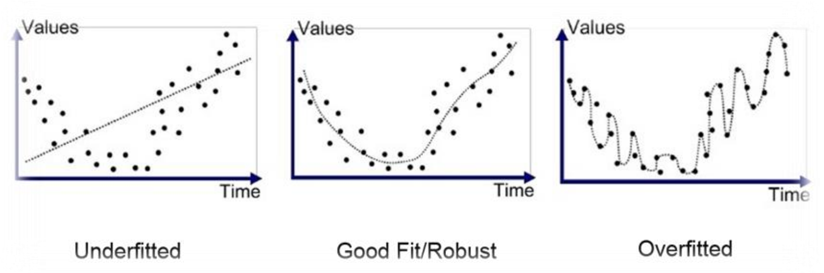


Figura 01

Partindo do underfitting para adequar a bias e a variância da função:

1. Podemos adicionar mais variáveis.
   a. Pode ocorrer overfitting.

Partindo do overfitting para adequar a bias e a variância da função:

2. Reduzir o nro de variáveis preditoras.
   a. Não é recomendado, pode haver perda de informações.

3. Regularização:
   a. Mantemos todas as variáveis preditoras e reduzimos a magnitude dos parâmetros *teta* da função

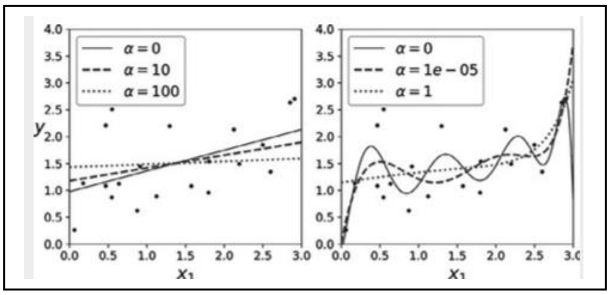
$h_\theta(x) = \theta_0 + \theta_1 x^2 + \theta_2 x^3 + \underline{\theta_3 x^3 + \theta_4 x^4} \rightarrow$ θ_3, θ_4 *tendendo para zero – significa que para valores muito pequenos para teta3 e teta4 a hipótese ficará mais simples passando a função de grau 4 para grau 2 reduzindo o overfitting.*

Nossa sorte é que não precisamos fazer isso tudo na mão, temos 3 algoritmos que foram elaborados para essa finalidade, fazer a regularização balanceando bias e variância:

- Regressão de Ridge
  - Versão regularizada da Regressão Linear.
  - Utiliza a Cost Function adicionando um termo de regularização nessa função forçando o algoritmo de aprendizado a ajustar os dados e manter o peso do modelo menor possível.

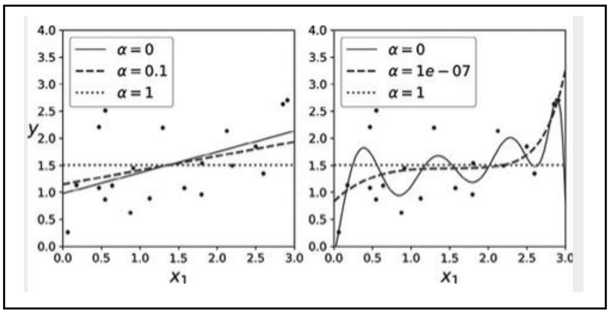
$$\text{RSS}_{\text{ridge}} = \sum_{i=1}^{n} \left[y_i - (\mathbf{w} \cdot \mathbf{x}_i + b)\right]^2 \underbrace{\boxed{+ \alpha \sum_{j=1}^{p} w_j^2}}_{\text{regularização } \ell_2}$$

  - É preciso escalonar os dados, pode usar StandarScaler, antes de executar a Regressão de Ridge.
  - À esquerda modelos de Ridge simples resultando em predições lineares.
  - À direita modelos de Ridge com dados escalonados resultando em predições polinomiais.
  - O alfa é o hiperparâmetro de regularização, podemos verificar a adequação da curva nos dois gráficos à medida que o alfa é modificado.



- Agora mostramos o algoritmo de Regressão de Ridge do sklearn que faz a regularização:
  from sklearn.linear.model import Ridge
  ridge_reg = Ridge(alpha=1, solver=”cholesky”)
  ridge_reg.fit(X, y)
  ridge_reg.predict([[1.5]])
  array([[1.55071465]])

- Regressão de Lasso
  - Assim como na Regressão de Ridge, também adiciona um termo de regularização na cost function.
  - O alfa também é o hiperparâmetro que modifica a regularização.
    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])



- Regressão Elastic Net
  - Combinação entre regularização de Ridge e de Lasso.
  - Controle dessa combinação em r=0(Ridge) ou r=1(Lasso), pode ser gradual mais próximo de zero ou mais próximo de 1.
  - O alfa é o hiperparâmetro que controla a regularização, l1_ratio é a combinação r.
    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])

Aqui você teve uma ideia bastante objetiva de como fazer uma regularização para balancear a bias e a variância evitando o sobreajuste, espero que tenha gostado, um abraço e até a próxima!