Regularizações

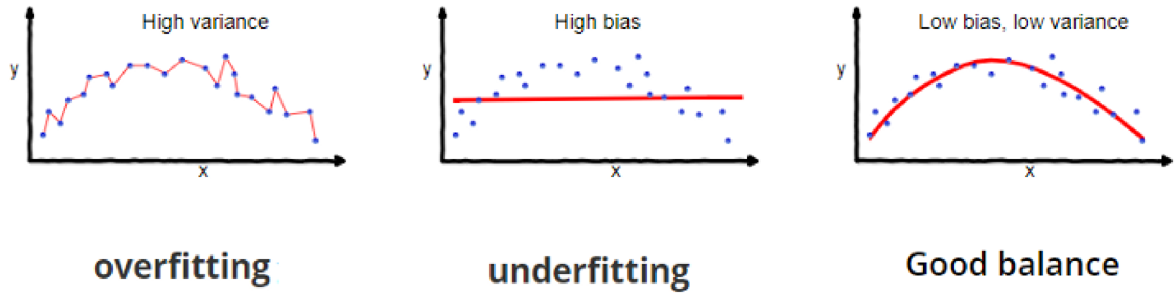
1. Introdução

A importância de utilizar regularizações está intrinsecamente relacionado com o modo que o modelo aprende de fato, pois existem casos que precisa-se auxiliar o modelo no processo de aprendizagem. Muitas vezes pode ocorrer de, ao utilizar uma base de **treinamento**, o modelo alcançar uma taxa de acerto muito alta (em torno de 90 a 100%) e, quando comparado ao resultado utilizando uma base de **teste**, este acerto caia para valores menores que 50%.

Este efeito é algo conhecido como **Overfitting**, significando que o modelo não está aprendendo as relações e generalizando, mas sim memorizando os resultados na base de treinamento, por isso esta discrepância nos resultados entre treino e teste. O *Overfitting* é um dos efeitos comuns de ocorrer em processos de modelagem e está relacionado com os conceitos de **viés** (*bias*) e **variância** (*variance*):

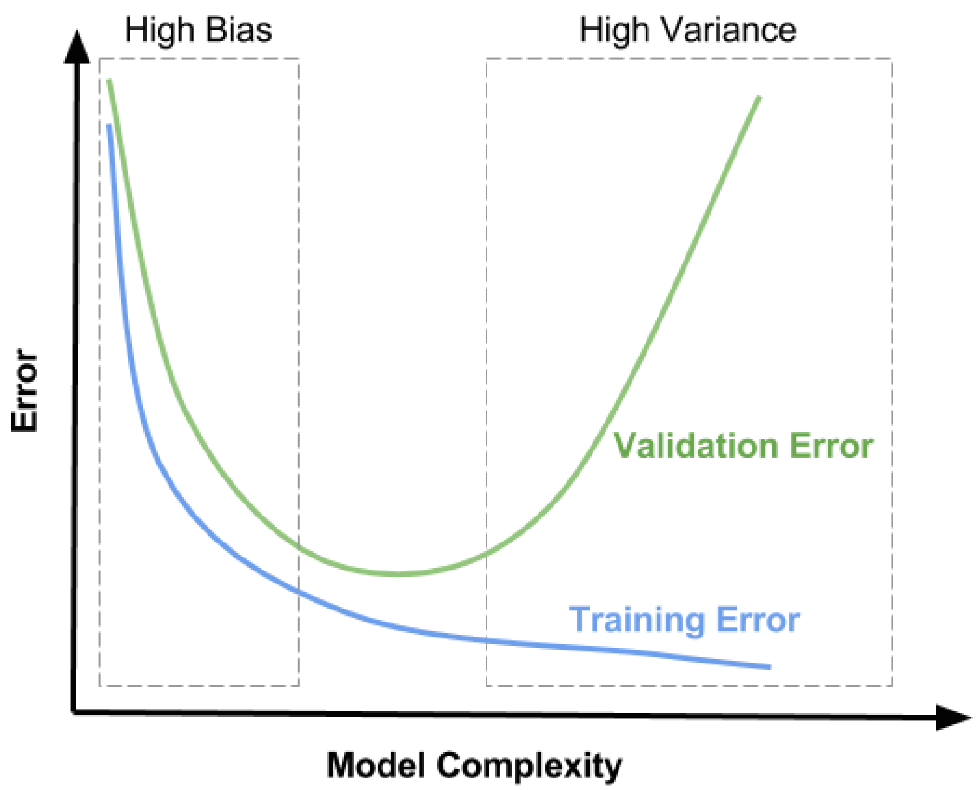
- **Viés**: É a diferença entre o que o modelo prediz, e o valor correto a ser predito. Modelos com alto viés são muito simples, de modo a **não conseguir capturar as relações que os dados de treino exibem**, ocasionando o chamado **Underfitting**. Também é bem comum acontecer o *Underfitting* quando modelos são aplicado em situações erradas, por exemplo em um problema de regressão utilizar um modelo de classificação e vice versa. Ambos os casos faz com que os erros de treino e de teste sejam altos, pois o modelo foi incapaz de compreender e generalizar as relações nos dados;

- **Variância**: Variância se refere à variabilidade das predições de um modelo. Modelos com alta variância são muito complexos, pois **aprendem muito as relações exibidas nos dados de treino**, no caso ocorrendo o *Overfitting*. Isso faz com que os erros de treinamento sejam baixos, mas os erros de teste sejam altos.



Fonte: Medium

Dessa forma, no processo de modelagem deve-se ponderar e equilibrar entre o Viés e a Variância apresentada nos modelos conforme o gráfico abaixo, representando o chamado **Trade-Off Viés Variância**:



Fonte: Learn OpenCV

No caso dos modelos baseados em Regressão, as aplicações que podem auxiliar no processo de ajudar na generalização dos modelos são as chamadas **regularizações.**

2. Regularizações

As **regularizações** vão ser uma importante ferramenta para auxiliar no ajuste de modelos de Regressão Linear. Quando modela-se uma Regressão Linear Múltipla, o objetivo é calcular os coeficientes que determinam a equação abaixo:

$$Y_j = \beta_0 + \sum_{i=1}^{n} \beta_i X_{ij} = \beta_0 + \beta X$$

Para se determinar os valores de todos os parâmetros β, o processo de modelagem envolve achar os parâmetros que minimizam a chamada **função de custo**, função esta que avalia o custo (ou seja o erro empregado) ao estimar o valor de Y, que para o caso das regressões a função de custo é dada pela **soma residual dos quadrados**, conforme a seguir:

$$\Theta = \sum_{i=1}^{n} [y_i - (\beta_0 + \beta X)]^2$$

Mas durante o processo iterativo para o cálculo dos parâmetros, um problema que pode surgir é o caso do *overfitting*, como discutido em tópicos anteriores. Ao invés do modelo aprender a **generalizar os resultados**, ele apenas passa a **memorizar** as respostas dos dados fornecidos no treinamento, prejudicando assim o real poder de predição da Regressão Linear e qualquer outro modelo de *Machine Learning*.

A forma utilizada para diminuir esse efeito nas regressões, seria justamente a **regularização**, onde de acordo com o tipo de regularização será adicionado a função de custo um termo conhecido como **penalização** proporporcional aos coeficientes β. Dessa forma, ao minimizar a função de custo, também será minimizado os parâmetros β.

Nos tópicos a seguir, serão apresentados os principais métodos de regularização para as regressões, sendo eles o **Ridge**, **Lasso** e **Elastic-Net**.

2.1 Ridge (L2)

O método Ridge ou penalização L2, consiste em adicionar um termo quadrático dos parâmetros na função de custo:

$$\Theta_{Ridge} = \sum_{i=1}^{n}[y_i - (\beta_0 + \beta X)]^2 + \alpha \sum_{j=1}^{p} \beta_j^2$$

Esse tipo de regularização é mais interessante de se usar quando **todas as variáveis atributos dos dados são importantes**, mas esperasse que o modelo generalize mais. O parâmetro α é justamente o que define a complexidade do modelo, quanto maior o α, mais simples o modelo, ou seja, menor a variância e maiores chances de ocorrer um *underfitting*.

O processo de treinamento e geração de novas predições funciona de forma análoga ao que acontece para a função `LinearRegression`, no caso, para implementar o *Ridge* basta carregar a função específica para ele:

```
# Carregando a função para o Ridge
from sklearn.linear_model import Ridge

# Instanciar o modelo
model = Ridge(alpha = 1.0) # Parâmetro de Ajuste do Ridge
```

2.2 Lasso (L1)

O método Lasso (ou penalização L1) consiste em adicionar o módulo dos parâmetros na função de custo, ao invés do quadrado no Ridge:

$$\Theta_{Lasso} = \sum_{i=1}^{n}[y_i - (\beta_0 + \beta X)]^2 + \alpha \sum_{j=1}^{p} |\beta_j|$$

O Lasso tem uma aplicação adicional bem interessante pois, no processo interativo de minimizar a função de custo, alguns parâmetros β serão **zerados**. Ou seja, o método pode ser utilizado como **uma seleção de atributos**, onde serão zerados os atributos menos relevantes para a modelagem. No caso do Lasso, se tivermos $\alpha = 0$, cai-se no caso clássico de regressão linear e, para os casos $\alpha > 0$, quanto maior o valor de lambda, mais parâmetros serão zerados.

De forma análoga ao que acontece no *Ridge*, no caso para implementar o **Lasso** basta carregar a função específica para ele:

```
# Carregando a função para o Lasso
from sklearn.linear_model import Lasso

# Instanciar o modelo
model = Lasso(alpha = 1.0) # Parâmetro de Ajuste do Lasso
```

2.3 Elastic-Net (L1 + L2)

O **Elastic-Net** é um caso particular bem interessante, pois ele combina ambos os efeitos de penalização L1 e L2, conforme descrito pela fórmula a seguir:

$$\Theta_{EN} = \sum_{i=1}^{n}[y_i - (\beta_0 + \beta X)]^2 + \alpha_1 \sum_{j=1}^{p}|\beta_j| + \alpha_2 \sum_{j=1}^{p}\beta_j^2$$

Ou seja, o *Elastic-Net* é interessante pois combina o poder de penalização efetiva do *Ridge* com as características de seleção de atributos do *Lasso*. Para implementar o *Elastic-Net* basta carregar a sua função específica:

```
# Carregando a função para o ElasticNet
from sklearn.linear_model import ElasticNet

# Instanciar o modelo
model = ElasticNet(alpha = 1.0) # Parâmetro de Ajuste do ElasticNet
```

Materiais Complementares

Canal *StatsQuest*, vídeo sobre Ridge vs Lasso regression Visualized;

Documentação no Scikit-Learn sobre o Ridge;

Documentação no Scikit-Learn sobre o Lasso;

Documentação no Scikit-Learn sobre o Elastic-Net;

Artigo publicado por Prashant Gupta no *Towards Data Science* - *Regularization in Machine Learning*.

Referências

James, Gareth, et al. An Introduction to Statistical Learning: With Applications in R. Alemanha, Springer New York, 2013;

Bruce A., Bruce P. Estatística Prática para Cientistas de Dados. Segunda Edição, Alta books, 2019;

Tópico anterior

Próximo Tópico