Regressão Linear

Fabrício Olivetti de França

Universidade Federal do ABC

Tópicos

1. Overfitting
2. Treino e Validação
3. Baseline dos modelos

UFABC

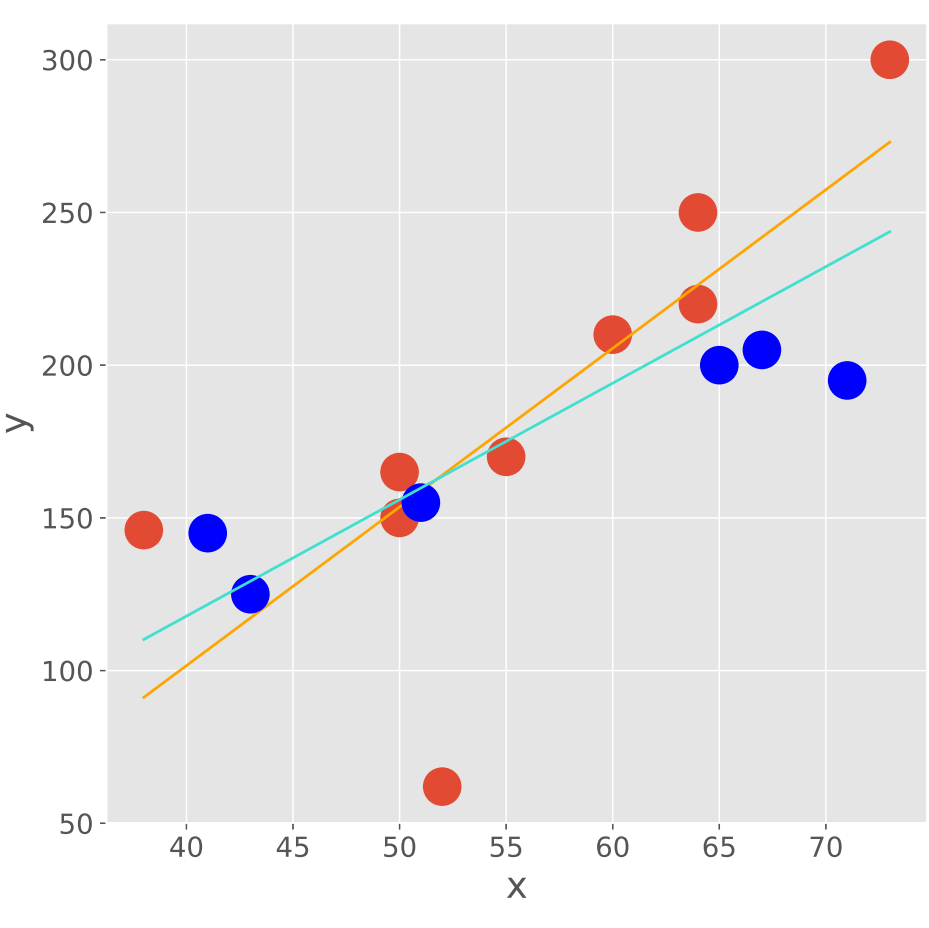
Overfitting

Overfit

Em muitos casos, a amostra de dados coletada não é representativa. No nosso exemplo, isso pode acontecer ao coletar dados de imóveis de apenas uma região específica.

Overfit

Isso faz com que a reta de regressão não necessariamente represente a realidade:



UFABC

Overfit

Isso pode ocorrer por causa de:

- Coleta de dados sem o devido cuidado estatístico.
- Apenas algumas da d variáveis são realmente importantes.
- Algumas das variáveis são correlacionadas (ex.: área útil e área total).

Regularização

Para resolver tal problema, utilizamos a regularização na função-objetivo. Para isso alteramos o cálculo do erro para:

$$e(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \mathbf{x_i}\cdot\mathbf{w})^2 + \lambda\sum_{i=0}^{d}|w_i|^p,$$

UFABC

Regularização

Com isso a equação de atualização de $\mathbf{w}$ passa a ser calculada como:

$$\mathbf{w^{t+1}} = \mathbf{w^t} + \alpha E((y_i - \mathbf{x_i} \cdot \mathbf{w}) \cdot x_i) - \lambda I_p,$$

com I_p sendo o gradiente da regularização escolhida.

Regularização l_2

Para $p = 2$ temos a norma quadrática cujo gradiente resultará em:

$$\mathbf{w^{t+1}} = \mathbf{w^t} + \alpha E((y_i - \mathbf{x_i} \cdot \mathbf{w}) \cdot x_i) - \lambda \mathbf{w}.$$

Essa é conhecida como regularização l_2 ou *ridge*. Esse tipo de regularização incentiva baixos valores para todos os elementos do vetor $\mathbf{w}$.

Ridge - Scikit-Learn

```
from sklearn.linear_model import Ridge

lr = Ridge(alpha = 0.1) # alpha é nosso lambda
lr.fit(X, y)
yi = lr.predict(xi)
```

UFABC

Regularização l_1

Para $p = 1$ temos o valor absoluto, o que resulta em descontinuidade. Portanto a fórmula de atualização fica:

$$\Delta \mathbf{w^t} = \mathbf{w^t} + \alpha E((y_i - \mathbf{x_i} \cdot \mathbf{w}) \cdot x_i)$$

$$\mathbf{w_i^{t+1}} = \begin{cases} \Delta \mathbf{w_i^t} - \lambda & \text{se } \Delta \mathbf{w_i^t} > \lambda \\ \Delta \mathbf{w_i^t} + \lambda & \text{se } \Delta \mathbf{w_i^t} < -\lambda \\ \Delta \mathbf{w_i^t} & \text{c.c.} \end{cases}$$

Essa é conhecida como regularização l_1 ou *lasso*. Esse tipo de regularização incentiva que alguns valores de $\mathbf{w}$ sejam zeros.

Lasso - Scikit-Learn

```
from sklearn.linear_model import Lasso

lr = Lasso(alpha = 0.1) # alpha é nosso lambda
lr.fit(X, y)
yi = lr.predict(xi)
```

UFABC

Regularização ElasticNet

Podemos combinar ambas as regularizações:

$$\Delta \mathbf{w^t} = \mathbf{w^t} + \alpha E((y_i - \mathbf{x_i} \cdot \mathbf{w}) \cdot x_i)$$

$$\mathbf{w_i^{t+1}} = \begin{cases} \Delta \mathbf{w_i^t} - \lambda_1 - \lambda_2 \mathbf{w} & \text{se } \Delta \mathbf{w_i^t} > \lambda \\ \Delta \mathbf{w_i^t} + \lambda_1 - \lambda_2 \mathbf{w} & \text{se } \Delta \mathbf{w_i^t} < -\lambda \\ \Delta \mathbf{w_i^t} - \lambda_2 \mathbf{w} & \text{c.c.} \end{cases}$$

Essa é conhecida como regularização ElasticNet.

ElasticNet - Scikit-Learn

```
from sklearn.linear_model import ElasticNet

# o lambda do l1 será l1_ratio * alpha
# o lambda do l2 será (1 - l1_ratio) * alpha
lr = ElasticNet(alpha = 0.1, l1_ratio = 0.5)
lr.fit(X, y)
yi = lr.predict(xi)
```

Treino e Validação

Avaliando o Modelo

Como verificar se meu modelo está causando *overfitting*?

Para isso podemos separar um pedaço de nossa base para avaliar seu desempenho em exemplos não vistos durante o treino.

Ajustando Parâmetros do Modelo

A ideia é que a maior parte da base de dados seja utilizada para o treinamento e uma pequena parte, não utilizada na fase anterior, seja utilizada para validação do desempenho do modelo, dados os parâmetros escolhidos.

Treino e Validação

Costuma-se definir uma divisão de 80% e 20%:

Símbolos	Links	Viagra	Contatos	Classe
3	1	4	0	Spam
5	3	0	0	Spam
1	1	0	2	Ham
5	3	0	0	Spam
1	1	0	2	Ham
5	3	0	0	Spam
1	1	0	2	Ham
5	3	0	0	Spam
1	1	0	2	Ham
5	3	0	0	Spam
5	3	0	0	Spam
1	1	0	2	Ham
5	3	0	0	Spam
1	1	0	2	Ham

UFABC

Treino e Validação

O procedimento então se torna:

- Para cada combinação de valores de parâmetros a serem considerados:
- Treina o modelo na base de treino
- Calcula o erro na base de validação
- Retorna a melhor combinação

Treino e Validação

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import Ridge

# 80-20
X_train, X_test,
   y_train, y_test = train_test_split(X, y, test_size=0.2)

lr = Ridge(alpha=alpha)
lr.fit(X_train, y_Train)
score = lr.score(X_test, y_test) # mais proximo de um teste real
```

UFABC

Treino e Validação

Mas e se:

Símbolos	Links	Viagra	Contatos	Classe
3	1	4	0	Spam
5	3	0	0	Spam
1	1	0	2	Ham
5	3	0	0	Spam
1	1	0	2	Ham
5	3	0	0	Spam
1	1	0	2	Ham
5	3	0	0	Spam
1	1	0	2	Ham
5	3	0	0	Spam
5	3	0	0	Spam
5	3	0	0	Ham
1	1	0	2	Ham
1	1	0	2	Ham

UFABC

Treino e Validação

Esse procedimento não garante que tenhamos uma amostra representativa, tanto para o ajuste do modelo quanto para o cálculo do erro.

Validação Cruzada

Uma ideia para contornar esse problema é a *Validação Cruzada*.

Na Validação Cruzada dividimos a base em k partes (denominadas pastas) e repetimos o experimento k vezes.

Em cada experimento, utilizamos uma das pastas como validação e o restante para treino, ao final retornando a média da medida de desempenho.

Validação Cruzada

Para $k = 4$, experimento 1:

Símbolos	Links	Viagra	Contatos	Classe
3	1	4	0	Spam
5	3	0	0	Spam
1	1	0	2	Ham
5	3	0	0	Spam
1	1	0	2	Ham
5	3	0	0	Spam
1	1	0	2	Ham
5	3	0	0	Spam
1	1	0	2	Ham
5	3	0	0	Spam
1	1	0	2	Ham
5	3	0	0	Spam
5	3	0	0	Spam
5	3	0	0	Ham

UFABC

Validação Cruzada

Para $k = 4$, experimento 2:

Símbolos	Links	Viagra	Contatos	Classe
3	1	4	0	Spam
5	3	0	0	Spam
1	1	0	2	Ham
5	3	0	0	Spam
1	1	0	2	Ham
5	3	0	0	Spam
1	1	0	2	Ham
5	3	0	0	Spam
1	1	0	2	Ham
5	3	0	0	Spam
5	3	0	0	Spam
5	3	0	0	Ham

UFABC

Validação Cruzada

Para $k = 4$, experimento 3:

Símbolos	Links	Viagra	Contatos	Classe
3	1	4	0	Spam
5	3	0	0	Spam
1	1	0	2	Ham
5	3	0	0	Spam
1	1	0	2	Ham
5	3	0	0	Spam
1	1	0	2	Ham
5	3	0	0	Spam
1	1	0	2	Ham
5	3	0	0	Spam
5	3	0	0	Spam
5	3	0	0	Ham

UFABC

Validação Cruzada

Para $k = 4$, experimento 4:

Símbolos	Links	Viagra	Contatos	Classe
3	1	4	0	Spam
5	3	0	0	Spam
1	1	0	2	Ham
5	3	0	0	Spam
1	1	0	2	Ham
5	3	0	0	Spam
1	1	0	2	Ham
5	3	0	0	Spam
1	1	0	2	Ham
5	3	0	0	Spam
5	3	0	0	Spam
5	3	0	0	Ham

UFABC

Validação Cruzada - ScikitLearn

```
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import Ridge

scoring = 'neg_mean_squared_error'
lr = Ridge(alpha=0.1)
meanScore = cross_val_score(lr, X_data, y_data,
                            cv=6, scoring=scoring
                            ).mean()
```

UFABC

Ajustando Parâmetros do Modelo

Uma outra questão importante para evitar o *Overfit* é o ajuste correto dos parâmetros dos algoritmos.

Do que vimos até agora, em um caso real deveríamos avaliar:

- O valor de k para o k-NN.
- A distância utilizada.
- O valor de α para a Regressão Linear.
- O valor de λ para o caso de utilizar regularização.

Ajustando Parâmetros do Modelo

Não podemos utilizar a separação anterior, pois ela foi feita para avaliar o modelo final.

Então aplicamos a validação cruzada dentro das pastas de treino!

Validação Cruzada - ScikitLearn

```
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import RidgeCV

scoring = 'neg_mean_squared_error'
lr = RidgeCV(alphas=[0.1, 1.0, 10.0])
meanScore = cross_val_score(lr, X_data, y_data,
                            cv=6, scoring=scoring
                            ).mean()
```

UFABC

Regressão Linear

Prós

- Modelo simples.
- Fácil de interpretar.
- Predição Rápida.

Contras

- Treino envolve minimização de funções, o que pode ser lento e pode ser difícil.
- Nem todas as bases de dados apresentam uma relação linear.

Baseline dos modelos

Validando um novo modelo

Suponha que você tenha acabado de criar um novo modelo de regressão ou classificação.

Como saber se ele funciona?

Essa pergunta difere de saber se ele é o melhor modelo!

Validando um novo modelo

Para isso precisamos de *baselines*, ou seja, modelos extremamente simples que tem um desempenho similar a um especialista humano.

Um *chute* acertado.

Métricas de Avaliação

Para poder comparar nossos modelos com outros modelos ou com diferentes parâmetros, precisamos de uma métrica de avaliação.

Essa métrica deve indicar qual modelo possui o menor erro de predição (minimização) ou o que tem maior índice de acertos (maximização).

Métricas de Avaliação - Regressão

Para as técnicas de regressão, temos duas métricas bastante utilizadas:

- Média dos Erros Absolutos
- Média dos Erros Quadráticos

Média dos Erros Absolutos

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|.$$

Erro da magnitude das diferenças do valor real e valor predito. Mesmo peso para todos os erros.

Média dos Erros Quadráticos

$$MAE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2.$$

Erro da magnitude das diferenças do valor real e valor predito. Peso maior para erros maiores.

Métricas de Avaliação - Classificação

Para as técnicas de classificação, temos as seguintes métricas populares:

- Acurácia
- Precisão
- Revogação
- F1
- Matriz de confusão
- AUC

Acurácia

Proporção de acertos em relação ao total de predições

$$acc = \frac{|acertos|}{|predições|}.$$

UFABC

Matriz de Confusão

Matriz cujo elemento $c_{i,j}$ indica quantos objetos da classe i foram classificados como da classe j. Objetivo é maximizar os valores de $c_{i,i}$.

	c_1	c_2	c_3
c_1	1023		
c_2	051		
c_3	2311		

Precisão

Assumindo tp como as amostras classificadas como $+1$ e que realmente as são, fp aquelas classificadas como $+1$ e que são -1, e tn, fn com o mesmo significado para a classe -1.

Precisão é a quantidade de classificações $+1$ corretas em relação a tudo que foi classificado como $+1$:

$$prec = \frac{tp}{tp + fp}.$$

UFABC

Revogação

Quantidade de classificações $+1$ corretas em relação a tudo que deveria ser classificado como $+1$:

$$rec = \frac{tp}{tp + fn}.$$

UFABC

F1

Média harmônica entre Precisão e Revogação:

$$prec = 2 \cdot \frac{(prec \cdot rec)}{prec + rec}$$

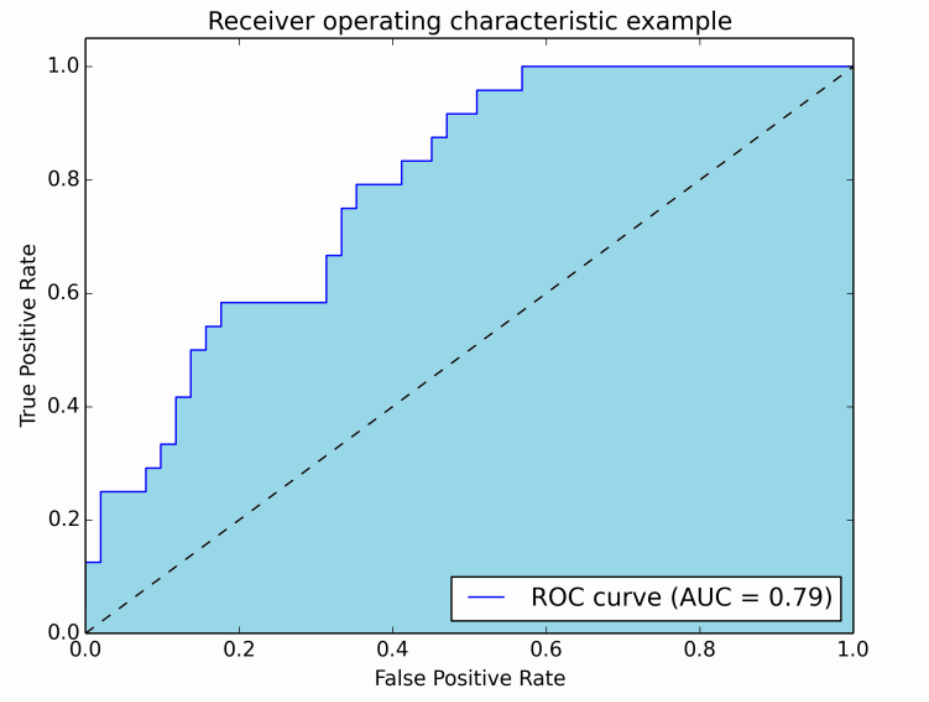
UFABC

AUC

Area Under the Curve: dado um classificador binário que responde um valor entre 0 e 1 representando a probabilidade de ser da classe $+1$. Podemos determinar que uma amostra é da classe $+1$ se a saída do classificador possui um valor maior que τ.

AUC

Se variarmos esse τ para calcular a precisão e revogação do nosso classificador, obtemos uma curva da seguinte forma:



UFABC

AUC

A área em azul é nossa área abaixo da curva, e quanto maior essa área melhor nosso classificador.

A linha pontilhada representa a performance de um classificador aleatório, se nosso classificador tem uma área abaixo dessa linha, então ele não está cumprindo seu papel.

Próxima Aula

Na próxima aula aprenderemos sobre:

- Transformação das variáveis - Linearizando as relações.
- Regressão Simbólica - Regressão não-linear interpretável (ou quase).

UFABC

Atividade 03

Complete o Laboratório:

Train_Test_Splits_Validation_Linear_Regression.ipynb

UFABC