

Exemplo: prever preços de casas

UCI: housing.data, 506 exemplos (subúrbio de Boston)

(explanatories)

(response)

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
0	0.00632	18	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4.98	24.0
1	0.02731	0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.14	21.6
2	0.02729	0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03	34.7
3	0.03237	0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	33.4
4	0.06905	0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	5.33	36.2

Regressão linear: uso da regularização

- Abordagem **Ridge Regression**

$$J(\mathbf{w})_{ridge} = \sum_{i=1}^n (y^{(i)} - \hat{y}^{(i)})^2 + \lambda \|\mathbf{w}\|_2^2$$

L2

$$\lambda \|\mathbf{w}\|_2^2 = \lambda \sum_{j=1}^m w_j^2$$

Assim como nos modelos para classificação, ao aumentar o valor de λ , comprimimos os pesos aprendidos.

Regressão linear: uso da regularização

- Abordagem Least Absolute Shrinkage and Selection Operator (LASSO)

$$J(\mathbf{w})_{LASSO} = \sum_{i=1}^n (y^{(i)} - \hat{y}^{(i)})^2 + \lambda \|\mathbf{w}\|_1$$

L1

$$\lambda \|\mathbf{w}\|_1 = \lambda \sum_{j=1}^m |w_j|$$

```
1 from sklearn.linear_model import Lasso
2 lasso = Lasso(alpha=0.5)
3 lasso.fit(X_train, y_train)
4 y_train_pred = lasso.predict(X_train)
5 y_test_pred = lasso.predict(X_test)
6 print(lasso.coef_)
```

```
[-0.0926693  0.04820685 -0.01406582  0.          -0.          2.44619324
 -0.00279111 -0.99274477  0.20992275 -0.01388192 -0.85539024  0.00750462
 -0.62386874]
```

Regularização mais forte. É possível que ao final do treinamento, muitos pesos possuam valor zero, fazendo com que LASSO seja também útil como selecionador de features.

Regressão linear: uso da regularização

- Abordagem **Elastic Net**

$$J(\mathbf{w})_{ElasticNet} = \sum_{i=1}^n (y^{(i)} - \hat{y}^{(i)})^2 + \lambda_1 \sum_{j=1}^m w_j^2 + \lambda_2 \sum_{j=1}^m |w_j|$$

L1 + L2

Combina Ridge com LASSO. Objetivo é usar L1 para permitir geração de esparsidade e L2 para superar limitação do LASSO no tocante ao número de variáveis selecionadas.

Regressão polinomial

- A regressão linear **assume relação de linearidade** entre variáveis explanatórias e variáveis de resposta
- Uma forma de lidar com esse pressuposto tentar modelar uma **relação não linear** é **adicionar termos polinomiais** (d = grau do polinômio)

$$y = w_0 + w_1x + w_2x^2 + \cdots + w_dx^d$$

Exemplo simples de Regressão polinomial

- Adicionando um termo polinomial (ex: grau 2) e aplicando a transformação nas características

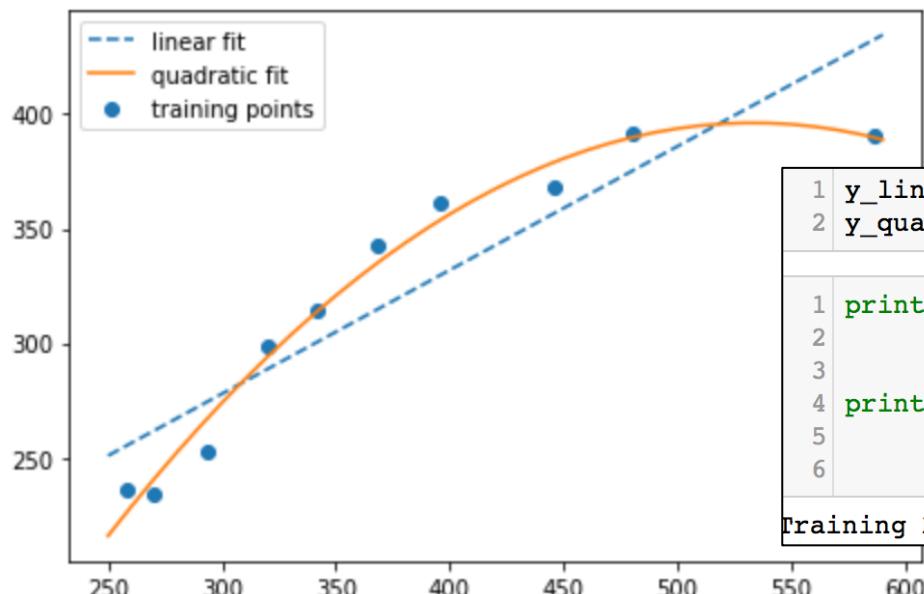
```
1 from sklearn.preprocessing import PolynomialFeatures
2
3 X = np.array([258.0, 270.0, 294.0,
4                 320.0, 342.0, 368.0,
5                 396.0, 446.0, 480.0, 586.0])[:, np.newaxis]
6
7 y = np.array([236.4, 234.4, 252.8,
8                 298.6, 314.2, 342.2,
9                 360.8, 368.0, 391.2,
10                390.8])
11
12 lr = LinearRegression()
13 pr = LinearRegression()
14 quadratic = PolynomialFeatures(degree=2)
15 X_quad = quadratic.fit_transform(X)
```

Exemplo simples de Regressão polinomial

- Treinando um modelo com adição das características quadráticas

```
1 # fit linear features
2 lr.fit(X, y)
3 X_fit = np.arange(250,600,10)[:, np.newaxis]
4 y_lin_fit = lr.predict(X_fit)
5
6 # fit quadratic features
7 pr.fit(X_quad, y)
8 y_quad_fit = pr.predict(quadratic.fit_transform(X_fit))
```

Exemplo simples de Regressão polinomial



- Modelo linear vs. quadrático

```
1 y_lin_pred = lr.predict(X)
2 y_quad_pred = pr.predict(X_quad)

1 print('Training MSE linear: %.3f, quadratic: %.3f' % (
2     mean_squared_error(y, y_lin_pred),
3     mean_squared_error(y, y_quad_pred)))
4 print('Training R^2 linear: %.3f, quadratic: %.3f' % (
5     r2_score(y, y_lin_pred),
6     r2_score(y, y_quad_pred)))
```

Training MSE linear: 569.780, quadratic: 61.330

MSE

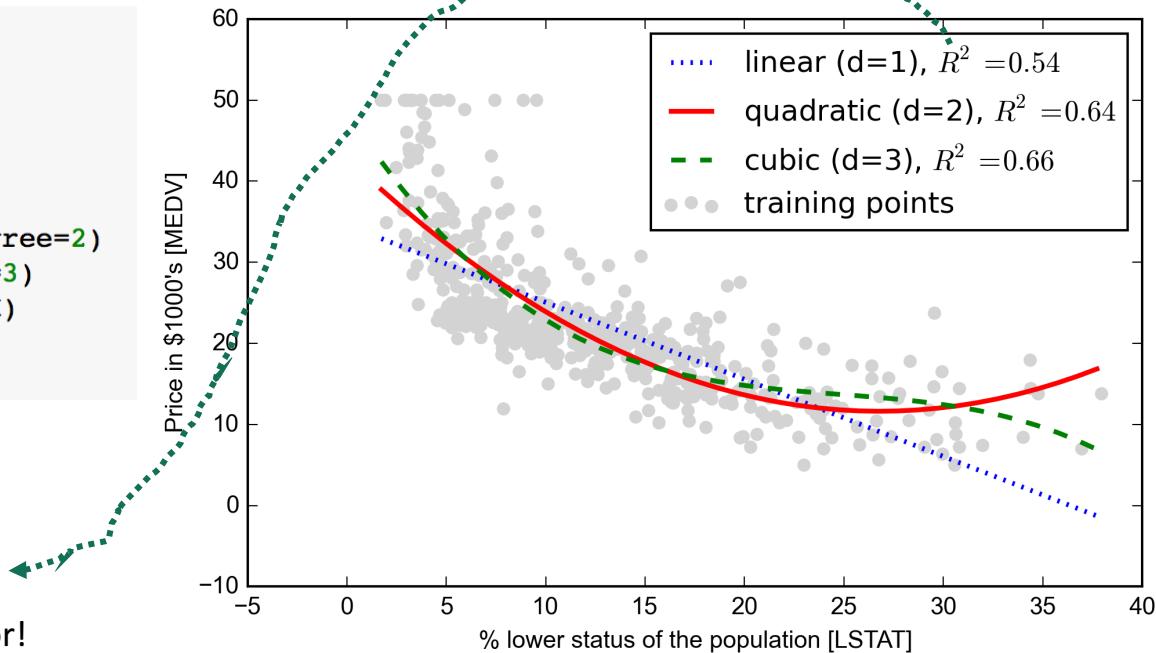
Aplicando para o dataset housing.data

- Características Lineares vs. Quadráticas vs. Cúbicas

```
1 x = df[['LSTAT']].values
2 y = df['MEDV'].values
3
4 regr = LinearRegression()
5
6 # create quadratic features
7 quadratic = PolynomialFeatures(degree=2)
8 cubic = PolynomialFeatures(degree=3)
9 X_quad = quadratic.fit_transform(X)
10 X_cubic = cubic.fit_transform(X)
11
```

$$R^2 = 1 - \frac{MSE}{Var(y)}$$

Versão normalizada do MSE:
quanto mais próximo de 1, melhor!





Hendrik Macedo

Escreve sobre Inteligência Artificial no Saense.

<http://www.saense.com.br/autores/artigos-publicados-por-hendrik-macedo/>