INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Ceará

HUAWEI

Regression Models

Douglas Chielle
douglas.chielle@ifce.edu.br

Objectives

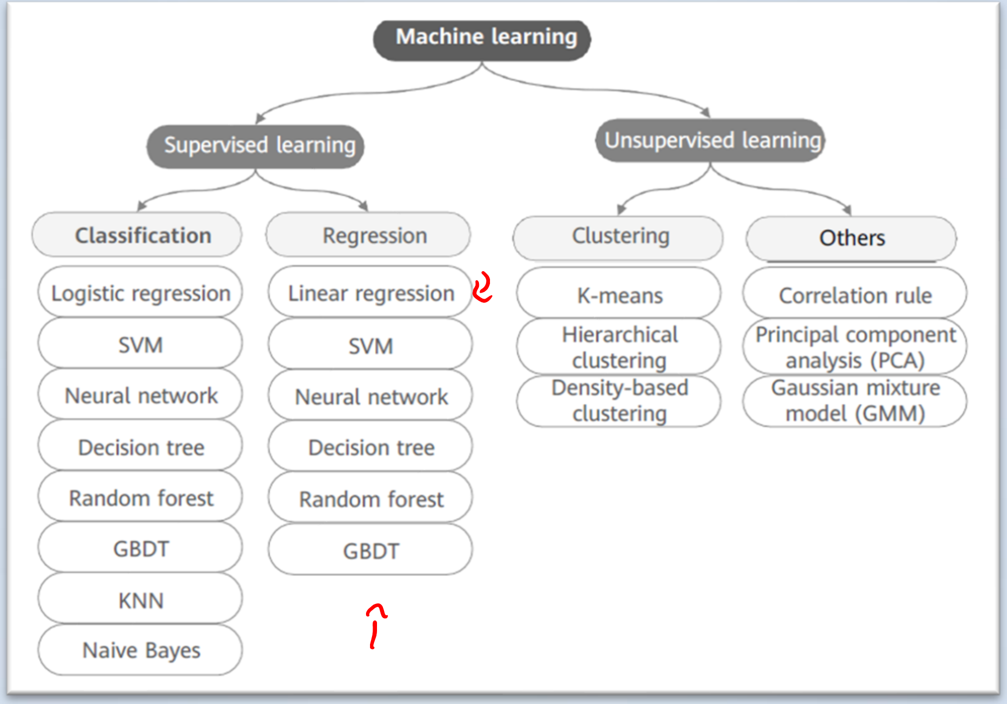
Upon completion of this lecture, you will be able to:

- Understand Linear Regression;
- Understand Regularized Linear Models;
- Understand Polynomial Regression; and
- Build Regression models with scikit-learn.

Content

1. ML Algorithm Overview
2. Linear Regression
   a) Closed-form
   b) Gradient Descent
3. Regularized Linear Models
   a) Ridge Regression
   b) Lasso Regression
4. Polynomial Regression
5. Regression with scikit-learn

ML Algorithm Overview



Regression

What is Regression?

- *Regression analysis* consists of a set of machine learning methods that allow us to estimate the relationship between a dependent variable and one or more independent variables.

  ↳ target

  ↳ predictors / features

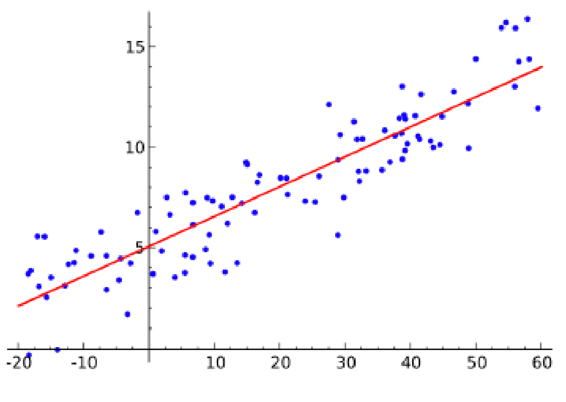
- It is widely used in supporting decision making, making predictions, time series modeling, etc.
- It is a supervised learning technique.

Linear Regression

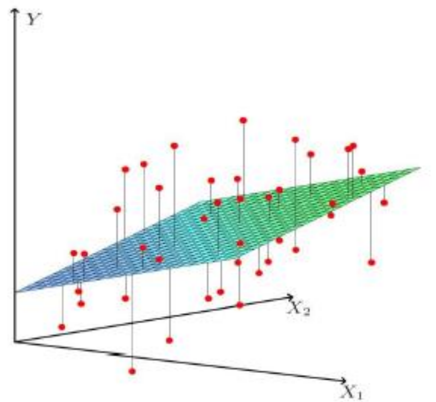
Linear Regression

- Linear regression is a linear approach to modelling the relationship between a scalar response and one or more explanatory variables.

$\hat{y} = \theta_0 + \theta_1 x$



Unary linear regression



Multi-dimensional linear regression

$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2$

Linear Regression

- A linear model makes a prediction by simply computing a weighted sum of the input features, plus a constant called the *bias term*:

$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n = \theta^T x$$

$\to [1 \;\; x_1 \;\; x_2 \;\; \cdots \;\; x_n]$

$\to [\theta_0 \;\; \theta_1 \;\; \theta_2 \;\; \cdots \;\; \theta_n]$

- In this equation:
  - $\hat{y}$ is the predicted value
  - n is the number of features
  - x_i is the i^{th} feature value
  - θ_j is the j^{th} model parameter

How to train it

- Training a model means setting its parameters so that the model best fits the training set.
- To measure how well this fit is, a common performance measure for is the Mean Squared Error (MSE), which for Linear Regression is:

$y = f(x)$

$f'(x) = 0$

$$MSE(\boldsymbol{X}, h_\theta) = \frac{1}{2m}\sum_{i=1}^{m}\left(\overbrace{\boldsymbol{\theta}^T\boldsymbol{x}^{(i)}}^{\hat{y}^{(i)}} - y_i\right)^2$$

The Normal Equation

- The ***Normal Equation*** is a closed-form solution that finds the value of $\boldsymbol{\theta}$ that minimizes the cost function:

$$\widehat{\boldsymbol{\theta}} = \left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\boldsymbol{X}^T\boldsymbol{y}$$

$$X = \begin{bmatrix} 1 & x_1^{(1)} & x_2^{(1)} & \cdots & x_n^{(1)} \\ 1 & \vdots & \vdots & \ddots & \\ \vdots & & & & \\ 1 & x_1^{(m)} & x_2^{(m)} & \cdots & x_n^{(m)} \end{bmatrix}$$

- In this equation
  - $\widehat{\boldsymbol{\theta}}$ is the value of $\boldsymbol{\theta}$ that minimizes the cost function.
  - $\boldsymbol{y}$ is the vector of target values containing y_1 to y_m.

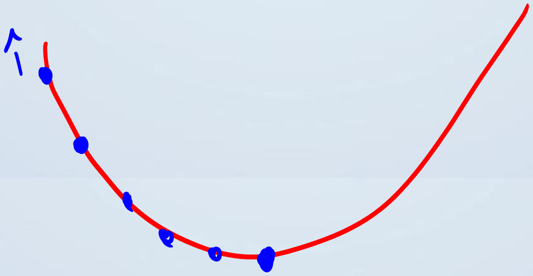
$$\hat{\theta} = \begin{bmatrix} \hat{\theta}_0 \\ \hat{\theta}_1 \\ \vdots \\ \hat{\theta}_n \end{bmatrix}$$

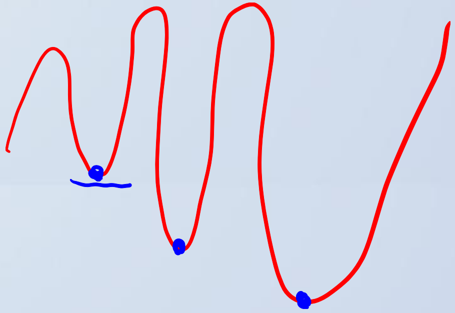
Gradient Descent (GD)

- The MSE cost function for a Linear Regression model is *convex* and continuous, with a slope that never changes abruptly.

- Therefore, GD is guaranteed to approach arbitrarily close the global minimum (given enough time and if the learning rate is not too high).

$$\nabla f = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_n} \right)$$





Gradient Descent (GD)

- To implement GD, we need to compute the gradient of the cost function w.r.t. each model parameter θ_j:

$$\frac{\partial}{\partial \boldsymbol{\theta}_j} MSE(\boldsymbol{\theta}) = \frac{1}{m}\sum_{i=1}^{m}\left(\boldsymbol{\theta}^T\boldsymbol{x}^{(i)} - y^{(i)}\right) x_j^{(i)}$$

- We can also compute all these partial derivatives in one go:

$$\nabla_{\boldsymbol{\theta}} MSE(\boldsymbol{\theta}) = \begin{pmatrix} \frac{\partial}{\partial \theta_0} MSE(\boldsymbol{\theta}) \\ \vdots \\ \frac{\partial}{\partial \theta_n} MSE(\boldsymbol{\theta}) \end{pmatrix} = \boxed{\frac{1}{m}\boldsymbol{X}^T(\boldsymbol{X}\boldsymbol{\theta} - \boldsymbol{y})}$$

Batch Gradient Descent

- Remarks:
  - The vector formula involves calculations over the full training set $\boldsymbol{X}$, at each GD step.
  - As a result, it is very slow on large training sets.
  - However, it scales well with the number of features.
- To update the model parameters, we multiply the gradient vector by the learning rate (η):

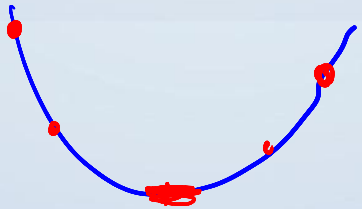
$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \eta \nabla_{\boldsymbol{\theta}} MSE\left(\boldsymbol{\theta}^{(t)}\right)$$

Stochastic Gradient Descent

- Stochastic-GD picks a random instance in the training set at every step and computes the gradients based only on that instance.
- This makes the algorithm much faster than Batch-GD.
- However, due to its stochastic nature, it is much less regular than Batch-GD.

- Updating the model parameters with SGD:

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \eta\left(\boldsymbol{x} \cdot \boldsymbol{\theta}^{(t)} - y\right)\boldsymbol{x}^{T}$$



Mini-Batch Gradient Descent

- Mini-Batch GD picks k random instances (mini-batch) in the training set at every step and computes the gradients based only on those instances.
- It is faster than Batch-GD and more stable than Stochastic-GD.

- Updating the model parameters with Mini-Batch-GD:

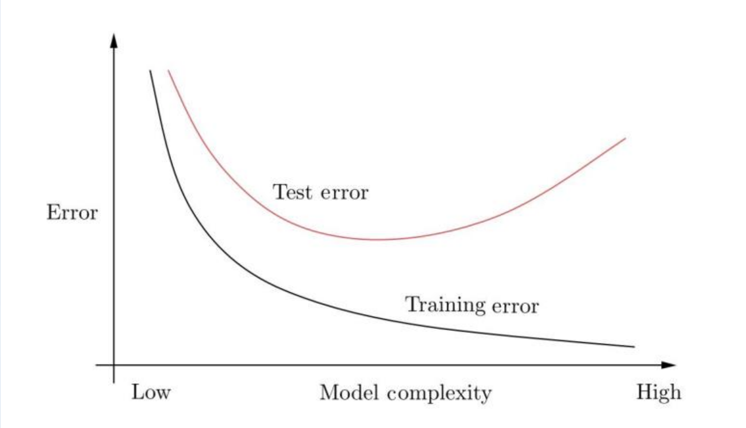
$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \eta \, \frac{1}{k} \boldsymbol{X}_k^T (\boldsymbol{X}_k \boldsymbol{\theta} - \boldsymbol{y})$$

  - $\boldsymbol{X}_k$ is a matrix with the k selected instances.

Regularized Linear Models

Regularized Linear Models

- Regularization is a technique for reducing the complexity of a model.
- For a linear model, regularization is typically achieved by constraining the model’s parameters.



Ridge Regression

- In Ridge Regression a regularization term is added to the cost function of Linear Regression:

$$J(\boldsymbol{\theta}) = MSE(\boldsymbol{\theta}) + \boxed{\alpha \sum_{i=1}^{n} \theta_i^2} = MSE(\theta) + \alpha \|\theta\|_2^2$$

- Characteristics:
  - There is a closed-form equation for Ridge Regression.
  - For GD we need to add $2\alpha\theta$ to the MSE gradient vector.

Lasso Regression

- For Lasso Regression we also add a regularization term cost function of Linear Regression:

$$J(\boldsymbol{\theta}) = MSE(\boldsymbol{\theta}) + \alpha \sum_{i=1}^{n} |\theta_i| = MSE(\theta) + \alpha \|\theta\|_1$$

- Chatacteristics:
  - Tends to eliminate the weights of the least important features.
  - Cost function is non differentiable at $\theta_i = 0$, but we can still use GD.

Elastic NET → Mix entre Ridge e Lasso regression.

Polynomial Regression

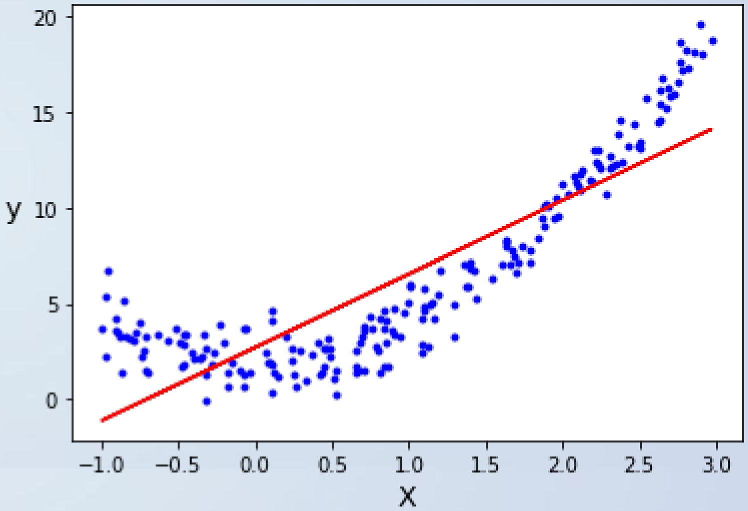
Polynomial Regression

$[x_1 \quad x_2]$

$\hookrightarrow [x_1 \quad x_2 \quad x_1^2 \quad x_2^2 \quad \sqrt{2}\, x_1 x_2]$

- What if a straight line does not properly fit the data?
- No problem, we can still use a linear model to fit the nonlinear data!
- To do this, we can add powers of each feature as new features.
- For example, if x is a scalar we might propose a quadratic model of the form:

$$\hat{y} = \theta_0 + \theta_1 x + \theta_2 x^2$$



$[x] \longrightarrow [x \quad x^2]$

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Ceará

HUAWEI

Regression with scikit-learn

Regression with scikit-learn

- Linear regression:
  - closed-form: sklearn.linear_model.LinearRegression
  - GD: sklearn.linear_model.SGDRegressor , penalty = None
- Polynomial regression
  - Use sklearn.preprocessing.PolynomialFeatures to transform the training data, then perform Linear Regression
- Ridge Regression
  - closed-form: sklearn.linear_model.Ridge, $solver = 'cholesky'$
  - GD: SGDRegressor with $penalty = 'l2'$
- Lasso Regression
  - sklearn.linear_model.Lasso
  - SGDRegressor with $penalty = 'l1'$

References

1. Géron, Aurélien. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems*. O'Reilly Media, 2019.

That's all Folks!