Regression

Liang Liang

Regression

- Regression is a subcategory of supervised learning where the goal is to predict continuous target value given the features of a sample.

- The relationship between the features and the target value may be linear or nonlinear.

- The target value could be a scalar or a vector.

Linear Regression

$\hat{y} = w_{(1)}x_{(1)} + w_{(2)}x_{(2)} + b$, a linear model
$y = \hat{y} + \varepsilon$

a data point x is a feature vector $\left[x_{(1)}, x_{(2)}\right]$

y is the 'true' target value (ground-truth)

$\hat{y}$ is the predicted target value from the model

ε is something that can not be explained by the linear model

ε is treated as 'random noise'

$\{w_{(1)}, w_{(2)}, b\}$ are the parameters of the linear model

Linear Regression

simple linear regression: $\hat{y} = w_{(1)}x + b$

b is intercept

w is slope

x is a scalar

multiple linear regression: $\hat{y} = w_{(1)}x_{(1)} + w_{(2)}x_{(2)} \dots + w_{(M)}x_{(M)} + b$

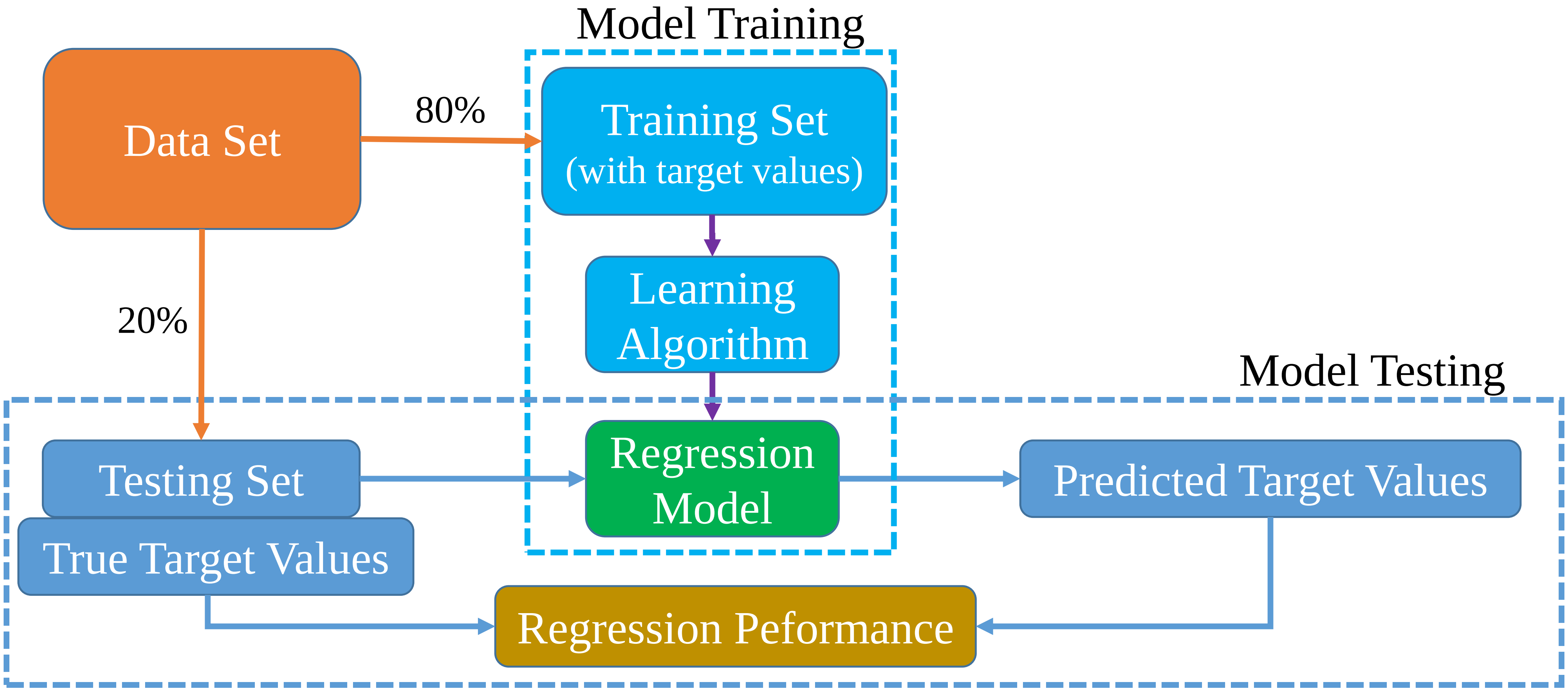
$\{w_{(1)}, w_{(2)}, \dots, w_{(M)}, b\}$ are the parameters of the linear model

$x = [x_{(1)}, x_{(2)}, \dots, x_{(M)}]$ is a sample (feature vector)

Fit the linear model to training dataset, to obtain the optimal parameters

Evaluate the trained/fitted linear model on testing dataset

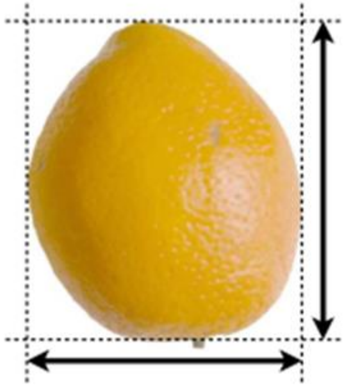
the workflow of a regression study



Example: linear regression on the fruit dataset

A bucket of fruits

The fruit dataset was created by Dr. Iain Murray at the University of Edinburgh. He bought a few dozen oranges, lemons and apples, and recorded their features in a table.



The fruit dataset {1:apple, 2:mandarin, 3:orange, 4:lemon},
Each row contains the information of a fruit sample/instance

fruit label	fruit_name	subtype	mass (g)	width (cm)	height (cm)	color_score
1	apple	granny_smith	192	8.4	7.3	0.55
4	lemon	spanish_belsan	194	7.2	10.3	0.70

http://usapple.org/the-industry/apple-varieties/

Example: linear regression on the fruit dataset

first step: load the dataset

The table has 59 rows (samples)

```
1  fruits = pd.read_table('fruit_data_with_colors.txt')
```

```
1  fruits
```

	fruit_label	fruit_name	fruit_subtype	mass	width	height	color_score
0	1	apple	granny_smith	192	8.4	7.3	0.55
1	1	apple	granny_smith	180	8.0	6.8	0.59
2	1	apple	granny_smith	176	7.4	7.2	0.60
3	2	mandarin	mandarin	86	6.2	4.7	0.80
4	2	mandarin	mandarin	84	6.0	4.6	0.79
5	2	mandarin	mandarin	80	5.8	4.3	0.77
6	2	mandarin	mandarin	80	5.9	4.3	0.81
7	2	mandarin	mandarin	76	5.8	4.0	0.81
8	1	apple	braeburn	178	7.1	7.8	0.92
9	1	apple	braeburn	172	7.4	7.0	0.89

linear regression on the fruit dataset
goal: predict mass given width, height and color

fruit label	fruit_name	subtype	mass (g)	width (cm)	height (cm)	color_score
1	apple	granny_smith	192	8.4	7.3	0.55

Feature Vector $x_n = [x_{(n,1)}, x_{(n,2)}, x_{(n,3)}]$

$x_{(n,1)}$: width
$x_{(n,2)}$: height
$x_{(n,3)}$: color

Linear Model

Target

mass $\hat{y}_n$

$$\hat{y}_n = w_{(1)}x_{(n,1)} + w_{(2)}x_{(n,2)} + w_{(3)}x_{(n,3)} + b$$

n is the index of the sample
(row index in the table)

linear regression on the fruit dataset - data splitting

split the data (59 samples) into a training dataset (80%) and a testing dataset (20%)

```
feature_names = ['width', 'height', 'color_score']
feature_names
```

```
['width', 'height', 'color_score']
```

```
target_name = ['mass']
target_name
```

```
['mass']
```

Split the data into a Training dataset and a Testing dataset

```
X = fruits[feature_names]
Y = fruits[target_name]
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=0)
```

Model Training: fit the model to the training dataset

The training set contains N input-output pairs: $\{(x_n, y_n), n = 1, \ldots, N\}$

x_n is a feature vector ; y_n is the true target value

MSE (mean squared error) loss function:

$$L(w_{(1)}, w_{(2)}, w_{(3)}, b) = \frac{1}{N}\sum_{n=1}^{N}(y_n - \hat{y}_n)^2$$

$$\hat{y}_n = w_{(1)}x_{(n,1)} + w_{(2)}x_{(n,2)} + w_{(3)}x_{(n,3)} + b$$

The goal of training is to find the best parameters $\{w_{(1)}, w_{(2)}, w_{(3)}, b\}$ such that the loss function is minimized. After training, the prediction $\hat{y}_n$ from the model should be very close to the true target y_n

The loss function

- Given N training data points with target values $\{(x_n, y_n), n = 1, \dots, N\}$,
  - $x_n = [x_{(n,1)}, x_{(n,2)}, \dots, x_{(n,M)}]$ is a data sample / point / feature vector
  - M is the number of components/attributes, and it is 3 for the fruit dataset
  - N is the number of samples
- find the best parameters that minimize the loss:

$$L(w_{(1)}, \dots, w_{(M)}, b) = \frac{1}{N} \sum_{n=1}^{N} (y_n - \hat{y}_n)^2$$

$$\hat{y}_n = w_{(1)} x_{(n,1)} + w_{(2)} x_{(n,2)} + \cdots + w_{(M)} x_{(n,M)} + b$$

$$\hat{y}_n = w_{(1)}x_{(n,1)} + w_{(2)}x_{(n,2)} + \cdots + w_{(M)}x_{(n,M)} + b$$

$$x_n = [x_{(n,1)}, x_{(n,2)}, \ldots, x_{(n,M)}]$$

Let $w = \left[w_{(1)}, \ldots, w_{(M)}\right]$

Let $w \cdot x_n = w_{(1)}x_{(n,1)} + w_{(2)}x_{(n,2)} + \cdots + w_{(M)}x_{(n,M)}$

then we have

$$\hat{y}_n = w \cdot x_n + b$$

$$L(w,b) = \frac{1}{N}\sum_{n=1}^{N}(y_n - \hat{y}_n)^2 \text{ and } \hat{y}_n = w \cdot x_n + b$$

$$x_n = [x_{(n,1)}, x_{(n,2)}, \ldots, x_{(n,M)}] \text{ and } w = \left[w_{(1)}, \ldots, w_{(M)}\right]$$

$$\text{Let } \frac{\partial L}{\partial w} = \left[\frac{\partial L}{\partial w_{(1)}}, \frac{\partial L}{\partial w_{(2)}}, \ldots, \frac{\partial L}{\partial w_{(M)}}\right]$$

Then we obtain:

$$\frac{\partial L}{\partial w} = -\frac{2}{N}\sum_{n=1}^{N}(y_n - w \cdot x_n - b) \times x_n$$

- Use gradient descent to find the optimal parameters w and b

  step-0: initialize w and b randomly

  step-1: compute $\frac{\partial L}{\partial w} = -\frac{2}{N}\sum_{n=1}^{N}(y_n - w \cdot x_n - b) \times x_n$

  $$\frac{\partial L}{\partial b} = -\frac{2}{N}\sum_{n=1}^{N}(y_n - w \cdot x_n - b)$$

  step-2: update $w \leftarrow w - \eta\frac{\partial L}{\partial w}$ and b $\leftarrow b - \eta\frac{\partial L}{\partial b}$

  where η is learning rate

  repeat step-1 and step-2 for a large number of iterations

Model Testing: apply the model to the testing dataset

The testing set contains K input-output pairs: $\{(x_k, y_k), k = 1, \dots, K\}$

mean squared error (MSE)

$$MSE = \frac{1}{K}\sum_{k=1}^{K}(y_k - \hat{y}_k)^2$$

mean absolute error (MAE)

$$MAE = \frac{1}{K}\sum_{k=1}^{K}|y_k - \hat{y}_k|$$

```
#prediction on the testing dataset
Y_test_pred = linear_model.predict(X_test)
MSE = np.mean((Y_test - Y_test_pred)**2)
MAE = np.mean(np.abs(Y_test - Y_test_pred))
MAPE =  np.mean(np.abs(Y_test - Y_test_pred)/Y_test)
print('MSE=', MSE)
print('MAE=', MAE)
print('MAPE=', MAPE)
```

we do not need for loops, use vectorized operation

Mean absolute percentage error (MAPE)

$$MAPE = \frac{1}{K}\sum_{k=1}^{K}\left|\frac{y_k - \hat{y}_k}{y_k}\right| \times 100\%$$

linear_regression_on_fruit_data.ipynb

LinearRegression_implementation.ipynb

Nonlinear regression using a polynomial model

- A polynomial model of degree K

$$\hat{y} = w_{(1)}x + w_{(2)}x^2 \ldots + w_{(K)}x^K + b$$

where x is a scalar

- Convert the polynomial model to a linear model:

$$\hat{y} = w_{(1)}\tilde{x}_{(1)} + w_{(2)}\tilde{x}_{(2)} + \cdots + w_{(K)}\tilde{x}_{(K)} + b$$

$$\tilde{x} = \left[\tilde{x}_{(1)}, \tilde{x}_{(2)}, \ldots, \tilde{x}_{(K)}\right] = \left[x, x^2, \ldots, x^K\right] \text{ is a feature vector}$$

- The optimal values of the parameters $\{w_{(1)}, \ldots, w_{(K)}, b\}$ can be obtained by using the gradient descent method with the MSE loss.