

Machine Learning Notes

Contents

1	Machine Learning Applications in real life	1
1.1	Prerequisites and used libraries	1
1.2	Concept	1
2	Data prepressing	2
3	Regression models	4
3.1	Simple Linear regression	5
3.1.1	Assumptions of linear regression models	6
3.1.2	Multiple Linear regression models	6
3.1.3	Polynomial Linear Regression	8

1 Machine Learning Applications in real life

- Face Recognition in Facebook
-

1.1 Prerequisites and used libraries

1.2 Concept

Features (independent variables) and dependent variables, the features which you are going to predict the dependent variables.

Median and Medium, Mean average:

Most Frequent Value

Best fit line

Regression V.s. Classification: In regression we predict a value, but in classification we predict a category or class.

The ground truth is what you measured for your target variable for the training and testing examples (simply the observed independent variables in the test set and the training set).

2 Data prepressing

While working with the machine learning at any phase, this step is mandatory you have some data and you need to prepare it to work in a perfect phase.

Listing 1: example embed source

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

dataset = pd.read_csv('Data.csv')
# The features
x = dataset.iloc[:, :-1].values

# the dependent variable
y = dataset.iloc[:, -1].values

# Dealing with the missing data
from sklearn.impute import SimpleImputer
imputer = SimpleImputer(missing_values=np.nan, strategy='mean')

#Apply the the mean for all the missing values on both
#age and salary
imputer.fit(x[:, 1:3])
x[:, 1:3] = imputer.transform(x[:, 1:3])

#Create one hot encoder for the country
#but Labal encoder for the purchased
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder

#Transformers is a set of paramters (list of tuples)
# Type = Encoder, onehotencoder, the index of the values need to be included, whihc is the country column
# Reminder; By default, only the specified columns in transformers are transformed and combined in the output, and the no
# By specifying remainder='passthrough', all remaining columns that were not specified in transformers will be automatical
ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(), [0])], remainder='passthrough')
# we convert the output from the fit_transform to a numpy arry style for future
x = np.array(ct.fit_transform(x))

from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
# no need to numpy array from
y = le.fit_transform(y)

# Feature scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x = sc.fit_transform(x)
```

```

# Import the train_test_split function
# It returns 4 elements: x_training x_test y_training y_test
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=0)

print(x_train)
print(x_test)
print(y_train)
print(y_test)

```

1. Deal with missing data using sklearn
2. Deal with categorical data Using Label Encoder V.s. One-Hot Encoder:
LabelEncoder can turn [dog,cat,dog,mouse,cat] into [1,2,1,3,2], but then the imposed ordinality means that the average of dog and mouse is cat. Still there are algorithms like decision trees and random forests that can work with categorical variables just fine and LabelEncoder can be used to store values using less disk space.

One-Hot-Encoding has the advantage that the result is binary rather than ordinal and that everything sits in an orthogonal vector space. The disadvantage is that for high cardinality, the feature space can really blow up quickly and you start fighting with the curse of dimensionality. In these cases, I typically employ one-hot-encoding followed by PCA for dimensionality reduction. I find that the judicious combination of one-hot plus PCA can seldom be beat by other encoding schemes. PCA finds the linear overlap, so will naturally tend to group similar features into the same feature.

3. Feature Scaling Why feature scaling, if there are different ranges in features then the features biased in the correlation computation might happen which means that features with high values might dominate the features with low values as the features with higher ranges might be having big impact on the dependent variables.

In linear regression there is no need to apply a feature scaling as the features has coefficients and features with very high values will be compensated by low coefficients.

Two techniques for feature scaling:

Standardisation	Normalisation
$x_{\text{stand}} = \frac{x - \text{mean}(x)}{\text{standard deviation}(x)}$	$x_{\text{norm}} = \frac{x - \min(x)}{\max(x) - \min(x)}$

- (a) Standardization As we divide on the standard deviation, the values will be between -x, +x
 - (b) Normalization it will put all the values between 0-1
4. Split the dataset into training set and test set Splitting into two sets, 80% used to train the model and 20% to test it.
We need to have the test set to overcome the overfitting with the training set, as the model trained too much on the training set and hence produces bad observations (it's like a student trained on a lot of exams and certain models, but when a new exams with new models came out he fails)

3 Regression models

regression analysis is a set of statistical processes for estimating the relationships between a dependent variable (often called the 'outcome variable') and one or more independent variables (often called 'predictors', 'covariates', or 'features').

This is the branch is used to predict a continuous numerical values (e.g. temperature, salary, ...)

Assumptions of linear regression: so before building a linear regression models, we need to check that these assumptions are true.

1. Linearity
2. Homoscedasticity
3. Multivariate normality
4. Independence of errors
5. Lack of multicollinearity

- Linear regression

3.1 Simple Linear regression

is simply the straight line equation $y = b_0 + b_1 * x_1$ where y is the dependent variable, and x is the independent one and b is the coefficient.

In linear regression we have one independent variable x

Example: the relation between the salary and the years of experience.

The best fitting line: it's the line with the smallest $\sum(y - \hat{y})^2$

Listing 2: example embed source

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

dataset = pd.read_csv('Salary_Data.csv')

x = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].values

from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=0)

#Simple linear regression
from sklearn.linear_model import LinearRegression
lregressor = LinearRegression()
lregressor.fit(x_train, y_train) # Train it

y_predicted = lregressor.predict(x_test)

# visualize the results
#Visualize the training set results
plt.scatter(x_train, y_train, color='red')
#plot the regression line for the training set
plt.plot(x_train, lregressor.predict(x_train), color='blue')
plt.title('Salary vs Experience (Training set)')
plt.xlabel('Years of experience')
plt.ylabel('Salary')
plt.show()

#Visualize the test set results
plt.scatter(x_test, y_test, color='green')
#plot the regression line for the test set
# The regression line is the same for the xtrain, and xtest
plt.plot(x_train, lregressor.predict(x_train), color='blue')
```

```
plt.title('Salary vs Experience (test set)')
plt.xlabel('Years of experience')
plt.ylabel('Salary')
plt.show()
```

3.1.1 Assumptions of linear regression models

- Linearity
- Homoscedasticity
- Multivariate normality
- Independence of errors
- Lack of multicollinearity

3.1.2 Multiple Linear regression models

It's a linear relationships between the dependent variables with more than one feature $y = b_0 + b_1 * x_1 + b_2 * x_2 + ... + b_n * x_n$.

Dummy Variables: when we have a categorical variables as features, then it's hard to represent them in the linear regression equation, instead we create dummy variables with a chosen coefficient and a switch to switch between the states.

Example:

Profit	R&D Spend	Admin	Marketing	State	Dummy
192,261.83	165,349.20	136,897.80	471,784.10	New York	1
191,792.06	162,597.70	151,377.59	443,898.53	California	0
191,050.39	153,441.51	101,145.55	407,934.54	California	0
182,901.99	144,372.41	118,671.85	383,199.62	New York	1
166,187.94	142,107.34	91,391.77	366,168.42	California	0

$$y = b_0 + b_1 * x_1 + b_2 * x_2 + b_3 * x_3 + b_4 * D$$

Dummy Variables trap:

Always omit one dummy variable out of the equation, so if we have 100 variables, just include 100-1.

Backward Elimination:

Before to go further we need to know how to calculate the p-value.

What is the null-hypothesis H_0 ? the p-value is used to reject the null-hypothesis, as if the p-value is too low so the null-hypothesis will be wrong which called a significant result (the alternative hypothesis), and if p value is large then the null-hypothesis is correct and this is non-significant result.

Building a Model:

In order to construct a reliable model, then we need first to decide which independent variables are important or which are not, which means that we might have some independent variables which are statistically significant, and others are not which might affect our model accuracy, we need need to keep only the ones with high statistically significant in our prediction, and the following methods are used to do so:

1. All-in You use this method if: 1. If you have a **prior knowledge** that you need to use them.
2. If you have to use them all.
3. Preparing to Backward Elimination.
2. Backward Elimination 1. Select a significance level to stay in the model (default $SL = 0.05$)
2. Fit the full model with all possible predictors
3. Consider the predictor with the highest P-value. if $P \leq SL$ then go to no.4, otherwise go to FIN.
4. Remove the predictor (remove the variable with the highest P-value)
5. Fit model, without the removed variable.
6. Repeat from 3 to 5 until the P-value $\leq SL$ is not correct go to FIN so this step your model is prepared.

By this we eliminate the number of variables till we

3. Forward Selection The opposite of backward one. 1. Select a significance level to enter the model (e.g. $SL = 0.05$)
2. Fit all **simple regression** models y x_n Select the one with the lowest P-value.
3. keep this variable and FIT all possible models with extra predictor added to the one(s) you already have.
4. Consider the predictor with the lowest P-value the lowest P-value, if $P \leq SL$, go to STEP3, otherwise go to FIN.
4. Bidirectional Elimination 1. Select a significance level to enter and stay in the model e.g. $SL_{ENTER} = 0.05$, $SL_{STAY} = 0.05$

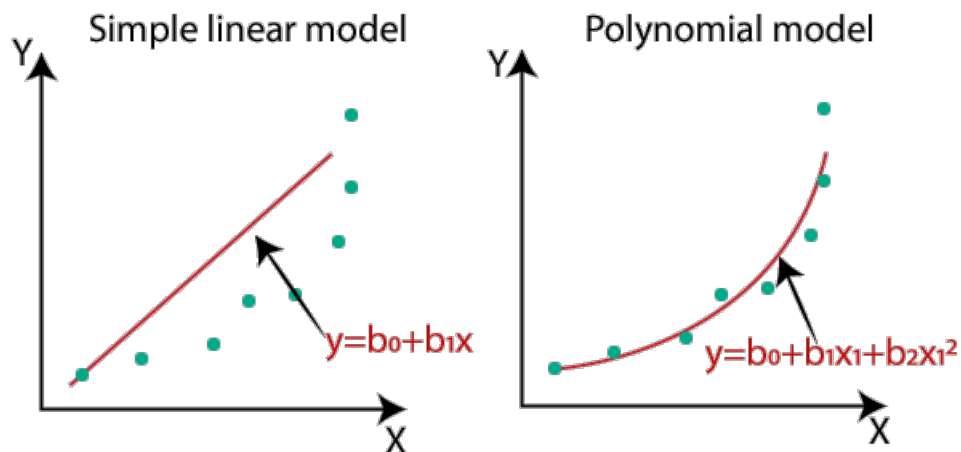
2. Perform the next step of the forward Selection (new variables must have: P ; SLENTER to enter).
3. Perform ALL steps of backward elimination (old variables must have P ; SLSTAY to stay).
4. Move back to step 2, until you can't add new variables, and no variables can't be eliminated.

5. Score Comparison

Note: step-wise regression refers to methods 2, 3, and 4.

3.1.3 Polynomial Linear Regression

It's a special case of multiple linear regression, in this case we will be having the equation on the shape: $y = b_0 + b_1 * x_1 + b_2 * x_1^2 + \dots + b_n * x_1^n$




Note: Why it's still called Linear regression, although we have some non-linearity ? The answer, the non-linearity comes from the coefficients not the X values.

- Non-linear models

—

•

 **Note:**

The Note here