ning-algorithms-linear-regression

May 26, 2025

1 Machine Learning Algorithms – (Banglish Version)

Machine Learning (ML) —
:

Supervised Unsupervised Semi-Supervised Reinforcement Learning

---

1.1 1. Supervised Learning

(→) Classification Regression

1.1.1 Classification

- **Logistic Regression**: Binary outcome (e.g., yes/no)
- **Decision Trees**: (: C4.5, CART)
- **Random Forest**: Decision tree- ensemble (overfitting)
- **Support Vector Machines (SVM)**: optimal hyperplane
- **k-Nearest Neighbors (k-NN)**: nearest values classify
- **Naïve Bayes**: Bayes' theorem feature independence

1.1.2 Regression

- **Linear Regression**: (e.g., Ordinary Least Squares)
- **Ridge/Lasso Regression**: Overfitting L2 L1 regularization
- **Gradient Boosting Machines (GBM)**: tree (: XGBoost, LightGBM)

1.2 2. Unsupervised Learning

(→)

1.2.1 Clustering

- **k-Means**: k
- **Hierarchical Clustering**: Tree structure (agglomerative divisive)
- **DBSCAN**: Density- outlier

1.2.2 Dimensionality Reduction

- **PCA (Principal Component Analysis)**: orthogonal axes-
- **t-SNE**: High-dimensional non-linear visualization

1.2.3 Association Rules

- **Apriori Algorithm**: itemsets (e.g., Market Basket Analysis)

1.3 3. Semi-Supervised Learning

( )

- **Self-Training**: Labelled unlabeled pseudo-label
- **Generative Adversarial Networks (GANs)**: Synthetic labelled

---

1.4 4. Reinforcement Learning (RL)

(Agent)

- **Q-Learning**: Q-value (model-free)
- **Deep Q-Networks (DQN)**: Q-learning + deep neural networks
- **Policy Gradient Methods**: optimize (e.g., REINFORCE)

1.5 5. Neural Networks & Deep Learning

(-)

- **Feedforward Neural Networks (FNN)**: Input → Hidden → Output layer NN
- **Convolutional Neural Networks (CNN)**: Image/video data- (e.g., ResNet)
- **Recurrent Neural Networks (RNN)**: Time series sequential (e.g., LSTM, GRU)
- **Transformers**: Self-attention model (e.g., BERT, GPT)

1.6 6. Ensemble Methods

( )

- **Bagging**: Parallel (e.g., Random Forest)
- **Boosting**: Sequential (e.g., AdaBoost, XGBoost)
- **Stacking**: Multiple model- meta-model

1.7 Algorithm

Problem Type	Classification, Regression, Clustering
Data Size & Quality	SVM
Interpretability	Linear models ; Deep Learning “Black Box”
Training Time	Deep Learning

1.8 Popular Libraries/Frameworks

Scikit-learn	Traditional machine learning (Python)
TensorFlow / PyTorch	Deep learning & neural networks
XGBoost / LightGBM	High-performance gradient boosting

1.9 . Supervised Learning

1.9.1 Regression ( )

- **Linear Regression**
- **Ridge / Lasso Regression**
- **Polynomial Regression**
- **Support Vector Regression (SVR)**
- **Decision Tree Regressor**
- **Random Forest Regressor**
- **Gradient Boosting Regressor (XGBoost, LightGBM, etc.)**

1.9.2 Classification ( )

- **Logistic Regression**
- **K-Nearest Neighbors (KNN)**
- **Decision Tree Classifier**
- **Random Forest Classifier**
- **Naive Bayes**
- **Support Vector Machine (SVM)**
- **Gradient Boosting Classifier (XGBoost, CatBoost, etc.)**

1.10 . Unsupervised Learning

1.10.1 Clustering

- **K-Means Clustering**
- **Hierarchical Clustering**
- **DBSCAN**

1.10.2 Dimensionality Reduction

- **Principal Component Analysis (PCA)**
- **t-SNE**
- **UMAP**

1.10.3 Association Rule Learning

- **Apriori**
- **Eclat**

1.11 . Semi-Supervised Learning

-
- Example: Label Spreading, Self-training Classifier

1.12 . Reinforcement Learning (RL)

- **Q-Learning**
- **Deep Q Network (DQN)**
- **Policy Gradient Methods**
- **Actor-Critic Methods**

1.13 . Neural Networks (Deep Learning)

1.13.1 Basic

- **Perceptron**
- **Multi-layer Perceptron (MLP)**

1.13.2 Computer Vision

- **Convolutional Neural Network (CNN)**

1.13.3 NLP / Time Series

- **Recurrent Neural Network (RNN)**
- **LSTM / GRU**
- **Transformer / BERT**

1.14 Frequently Used Libraries (Python)

Basic ML	`scikit-learn`

Deep Learning	TensorFlow, Keras, PyTorch
Gradient Boosting	XGBoost, LightGBM, CatBoost
NLP	spaCy, nltk, transformers
Clustering & PCA	scikit-learn, UMAP, hdbscan

1.15 ?

:

- **Classification: Logistic Regression, KNN, Decision Tree**
- **Regression: Linear Regression**
- **Unsupervised: K-Means, PCA**

2 Simple Linear Regression Implemented my handsOn

[ ]:
```
class myLR:
  def __init__(self):
    self.m=None
    self.b=None

  def fit(self,X_train,y_train):
    num=0
    den=0

    for i in range(X_train.shape[0]):
      num+= ((y_train[i]-y_train.mean()) * (X_train[i]-X_train.mean()))
      den+= (X_train[i]-X_train.mean())**2

    self.m=num/den
    self.b= y_train.mean()-(self.m*X_train.mean())

    print(f'Slop (m):{self.m} and y-intercept (b):{self.b}')

  def predict(self,X_test):
    y=self.m*X_test+self.b
    print(f'Predicted value :{y}')
```

[ ]:
```
import numpy as np
import pandas as pd
```

[ ]:
```
df=pd.read_csv('/content/placement.csv')
df.head()
```

```
[ ]:    cgpa  package
     0  6.89     3.26
     1  5.12     1.98
     2  7.82     3.25
     3  7.42     3.67
     4  6.94     3.57
```

```
[ ]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 2 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   cgpa     200 non-null    float64
 1   package  200 non-null    float64
dtypes: float64(2)
memory usage: 3.3 KB
```

```
[ ]: from sklearn.model_selection import train_test_split
     X_train,X_test,y_train,y_test=train_test_split(df['cgpa'].values,df['package'].
      ↪values,test_size=0.2,random_state=2)#.values cause our model need values not␣
      ↪Series/Dataframe
     X_train
```

```
[ ]: array([7.14, 8.93, 5.42, 5.1 , 7.77, 6.76, 6.89, 6.68, 7.91, 7.89, 8.71,
            7.95, 6.61, 6.26, 6.53, 6.42, 5.11, 6.09, 6.93, 7.04, 5.94, 6.05,
            5.83, 5.95, 9.31, 5.58, 7.88, 6.13, 7.76, 4.85, 6.19, 8.6 , 6.07,
            7.18, 5.12, 7.39, 8.25, 8.28, 7.13, 7.35, 5.66, 5.99, 8.01, 7.14,
            6.34, 6.89, 5.42, 6.47, 7.69, 7.4 , 7.28, 5.95, 7.38, 6.93, 8.99,
            7.36, 7.08, 5.38, 7.56, 8.22, 5.84, 6.78, 7.19, 7.28, 6.79, 6.12,
            6.85, 8.2 , 6.84, 7.37, 6.22, 6.61, 5.23, 7.21, 6.85, 6.19, 7.3 ,
            6.17, 5.89, 8.09, 7.11, 4.26, 6.94, 5.98, 6.71, 7.33, 9.06, 6.1 ,
            5.48, 6.1 , 7.56, 7.29, 5.84, 7.48, 7.61, 5.79, 5.61, 7.34, 9.38,
            7.91, 6.94, 7.94, 8.31, 6.96, 6.93, 7.11, 8.44, 8.18, 6.66, 8.44,
            7.12, 6.3 , 5.84, 6.98, 7.63, 5.64, 7.43, 8.87, 7.84, 5.84, 9.58,
            8.37, 7.63, 6.31, 6.5 , 8.11, 6.07, 4.73, 7.3 , 6.51, 7.28, 6.92,
            6.35, 8.62, 7.05, 9.26, 6.33, 6.22, 6.94, 5.13, 8.13, 5.9 , 9.04,
            6.06, 7.57, 8.1 , 9.16, 5.84, 7.89, 6.63, 7.09, 5.53, 6.75, 7.62,
            6.97, 7.66, 6.14, 7.78, 7.25, 8.65])
```

```
[ ]: mylr=myLR()
```

```
[ ]: mylr.fit(X_train,y_train)
```

```
Slop (m):0.5579519734250721 and y-intercept (b):-0.8961119222429152
```

[ ]:
```
mylr.predict(X_test[0])
```

```
Predicted value :3.891116009744203
```

[ ]:
```

```