LOGISTIC REGRESSION, LINEAR REGRESSION AND ONE VS REST APPROACH

WEEK 7

OUTLINE FOR THE SESSION

REPETITION ON LINEAR REGRESSION

LOGISTIC REGRESSION

LINEAR VS LOGISTIC

CODE EXAMPLES

GRADIENT DESCENT AND MINIMISATION OF ERRORS

ONE VS REST CLASSIFICATION

WHAT IS LINEAR REGRESSION/CLASSIFICATION?

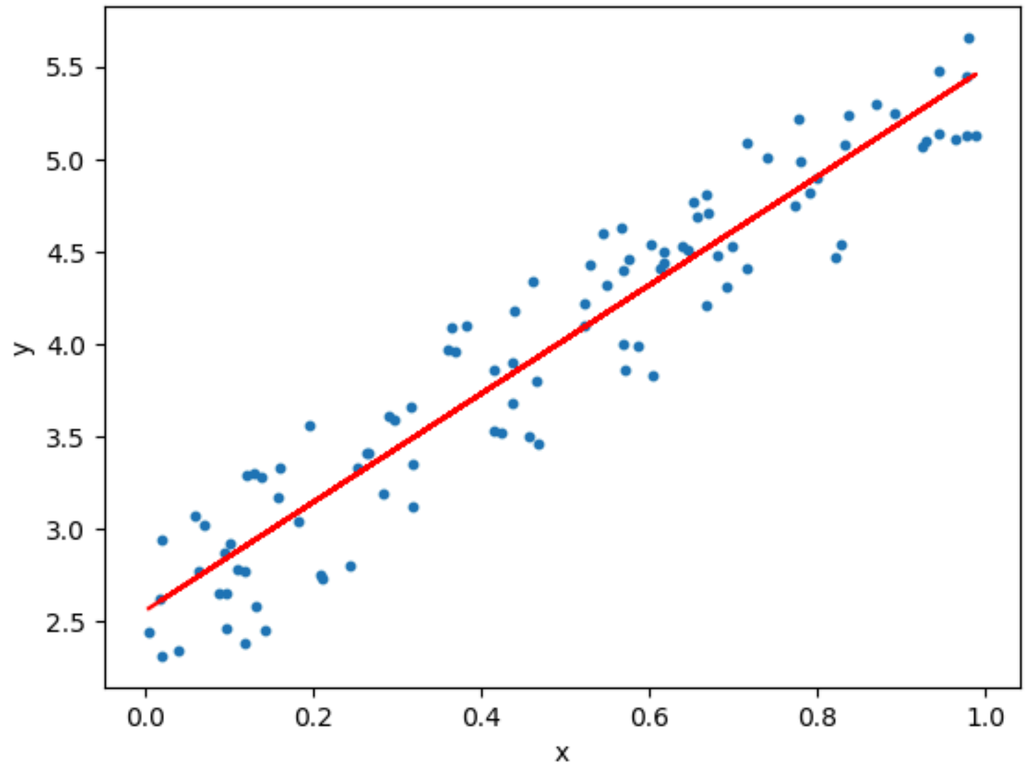
REP ON LINEAR REGRESSION

Linear Regression Formula:

$$\hat{y} = mx + b$$

Where:

- $\hat{y}$ is the predicted value.
- x is the independent variable.
- m is the slope of the line.
- b is the y-intercept.



The fundamental idea is to find the best-fitting line that minimizes the difference between the predicted and actual values.

WHAT TYPE OF LOSS FUNCTION IS APPROPRIATE FOR LINEAR REGRESSION?

LINEAR REGRESSION USES MSE

Error Function Formula:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Where:

n is the number of data points.

y_i is the actual value.

$\hat{y}_i$ is the predicted value.

WHAT IS LOGISTIC REGRESSION?

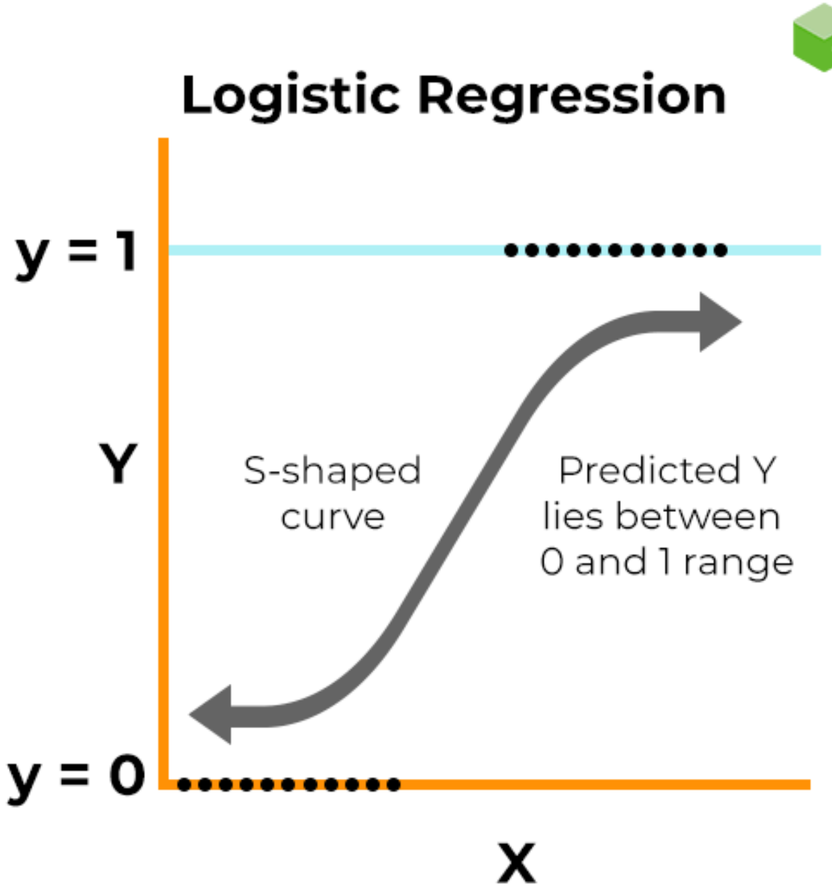
LOGISTIC REGRESSION

Binary classification, predicting the probability that an instance belongs to a particular class.

The sigmoid function

$$f(x) = \frac{1}{1+e^{-(x)}}$$

$$h\theta(X) = \frac{1}{1+e^{-(\beta_0+\beta_1 X)}}$$

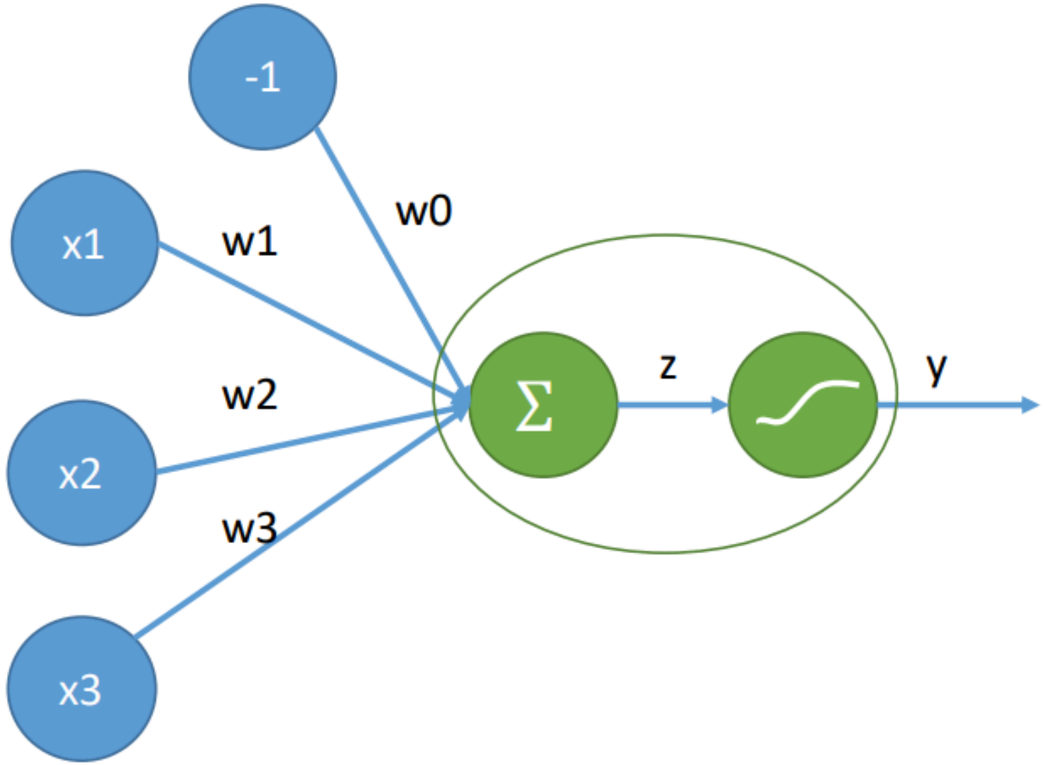


LOGISTIC REGRESSION

- First sum of weighted inputs :
- $z = \sum_{i=0}^{m} w_i x_i = \boldsymbol{w} \cdot \boldsymbol{x}$
- Apply the logistic function σ to this sum

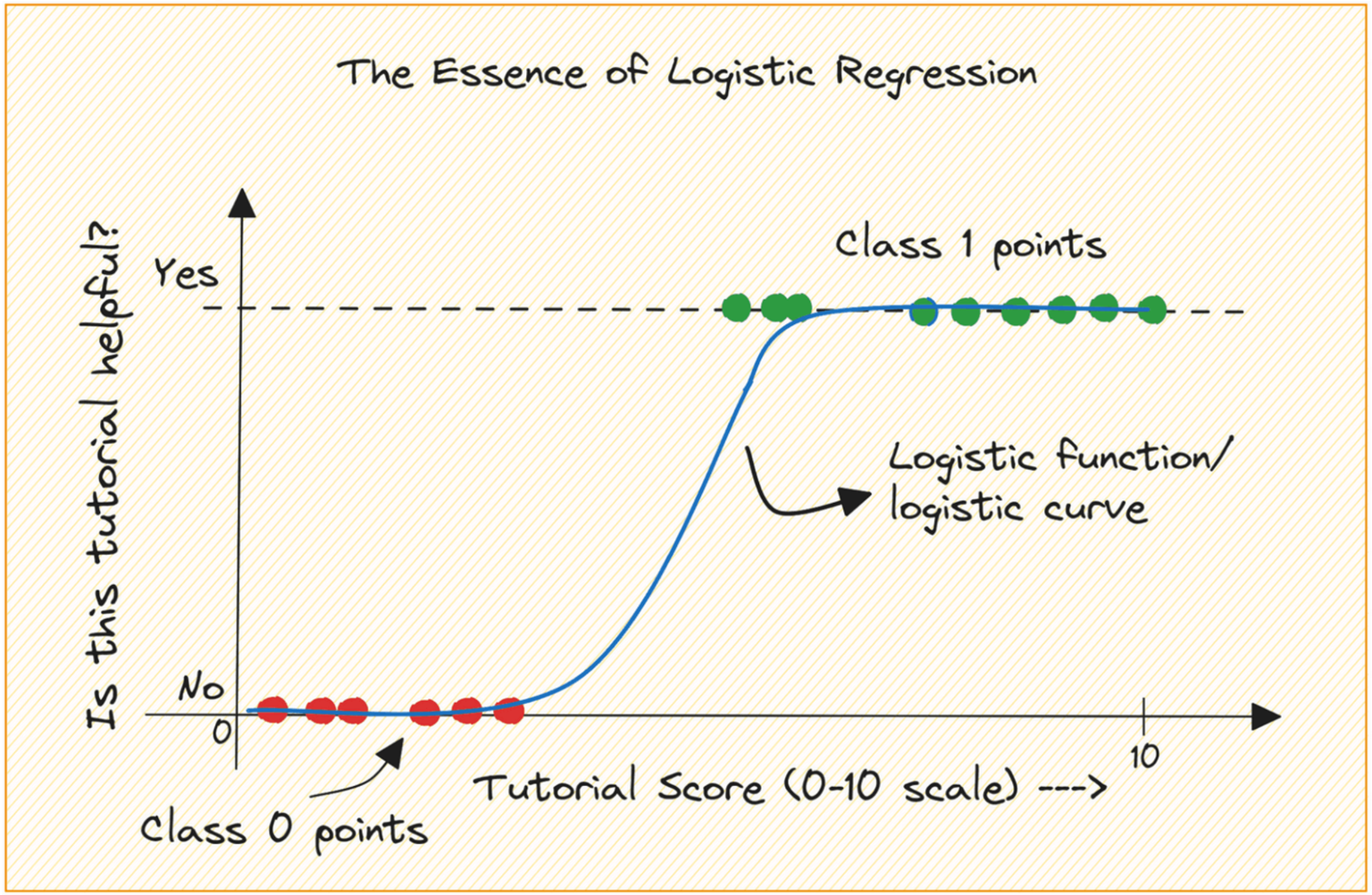
$$y = \sigma(z) = \frac{1}{1 + e^{-\vec{w} \cdot \vec{x}}}$$

- For $\boldsymbol{x} = \vec{x}$ predict
  - as the positive class if $y > 0.5$,
  - otherwise, the negative class



LOGISTIC REGRESSION

The logistic function squeezes the output between 0 and 1, representing the probability. If the probability is greater than a threshold (often 0.5), the instance is predicted to belong to the positive class; otherwise, it is predicted to belong to the negative class.



WHAT IS APPROPRIATE LOSS FUNCTION FOR LOGISTIC REGRESSION?

CROSS ENTROPY LOSS

Error Function Formula:

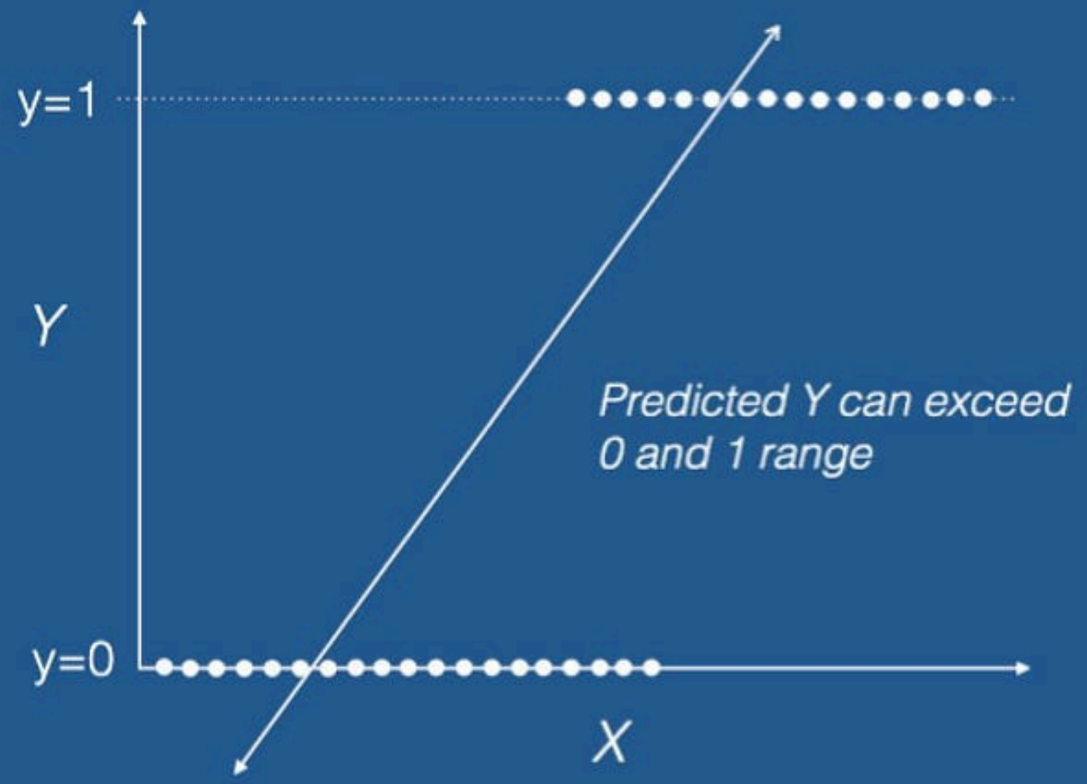
$$\text{Cross-Entropy Loss} = -\frac{1}{n}\sum_{i=1}^{n}[y_i \log(\hat{p}_i) + (1 - y_i)\log(1 - \hat{p}_i)]$$

Where:

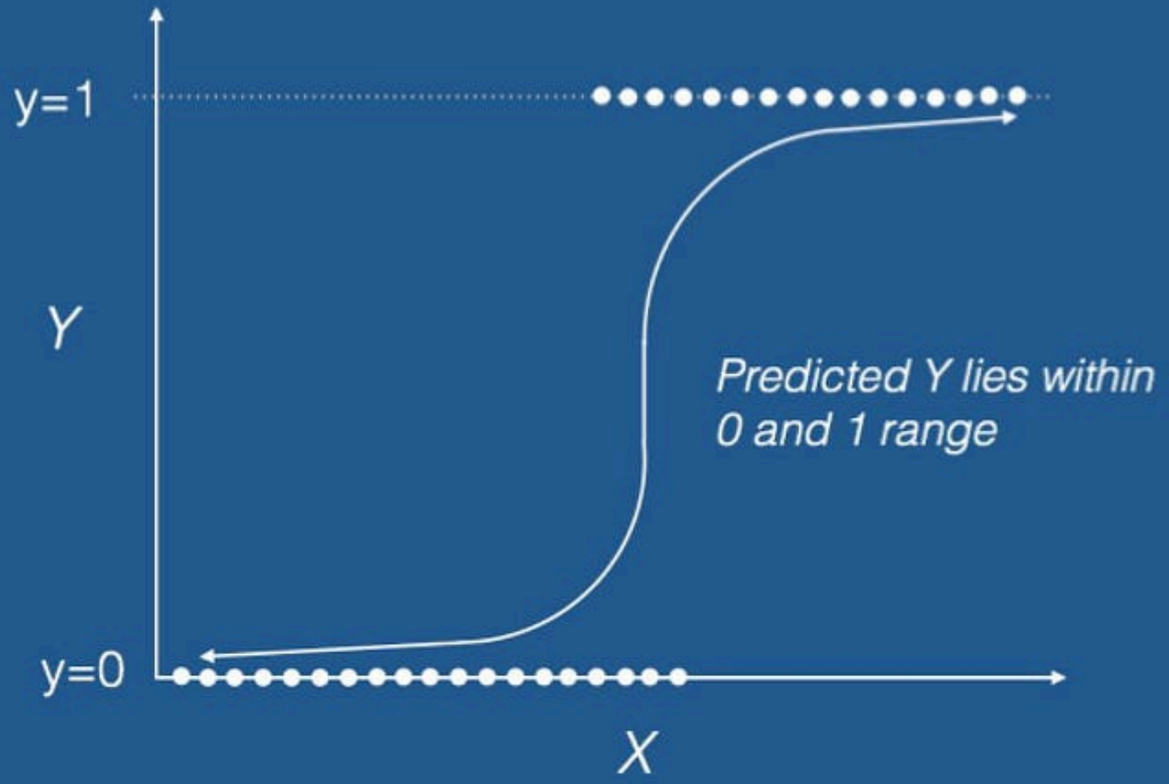
- n is the number of data points.
- y_i is the actual binary label (0 or 1).
- $\hat{p}_i$ is the predicted probability of belonging to the positive class.

WHAT IS THE DIFFERENCE BETWEEN THEM

Linear Regression



Logistic Regression



WHAT IS THE DIFFERENCE BETWEEN THEM

Linear classification	Logistic regression
Used for predicting a continuous outcome. It models the relationship between the dependent variable and independent variables as a linear equation.	Used for predicting the probability of an instance belonging to a particular class in binary classification problems. It models the relationship using the logistic function to produce values between 0 and 1.
Outputs any real number, positive or negative. Predictions can be outside the range [0, 1].	Outputs values between 0 and 1 due to the logistic (sigmoid) function, representing probabilities.
Assumes a linear function: $f(\boldsymbol{x}) = f((x_1, x_2, \dots, x_m)) = w_0 + w_1x_1 + w_2x_2 + \dots w_mx_m$	Applies the logistic function to the sum of weighted inputs: $\frac{1}{1 + e^{-\vec{w}\cdot\vec{x}}}$
Type of problem: regression analysis. E.g.: predicting housing prices.	Type of problem: Binary classification. E.g.: predict an email is spam or not.

CODE EXAMPLES: LINEAR REGRESSION

```python
def add_bias(X):
    # Put bias in position 0
    sh = X.shape
    if len(sh) == 1:
        #X is a vector
        return np.concatenate([np.array([1]), X])
    else:
        # X is a matrix
        m = sh[0]
        bias = np.ones((m,1)) # Makes a m*1 matrix of 1-s
        return np.concatenate([bias, X], axis  = 1)
```

```python
class NumpyLinRegClass(NumpyClassifier):

    def fit(self, X_train, t_train, eta = 0.1, epochs=10):
        """X_train is a Nxm matrix, N data points, m features
        t_train are the targets values for training data"""

        (k, m) = X_train.shape
        X_train = add_bias(X_train)

        self.weights = weights = np.zeros(m+1)

        for e in range(epochs):
            weights -= eta / k *  X_train.T @ (X_train @ weights - t_train)

    def predict(self, x, threshold=0.5):
        z = add_bias(x)
        score = z @ self.weights
        return score>threshold
```

CODE EXAMPLES: LOGISTIC REGRESSION

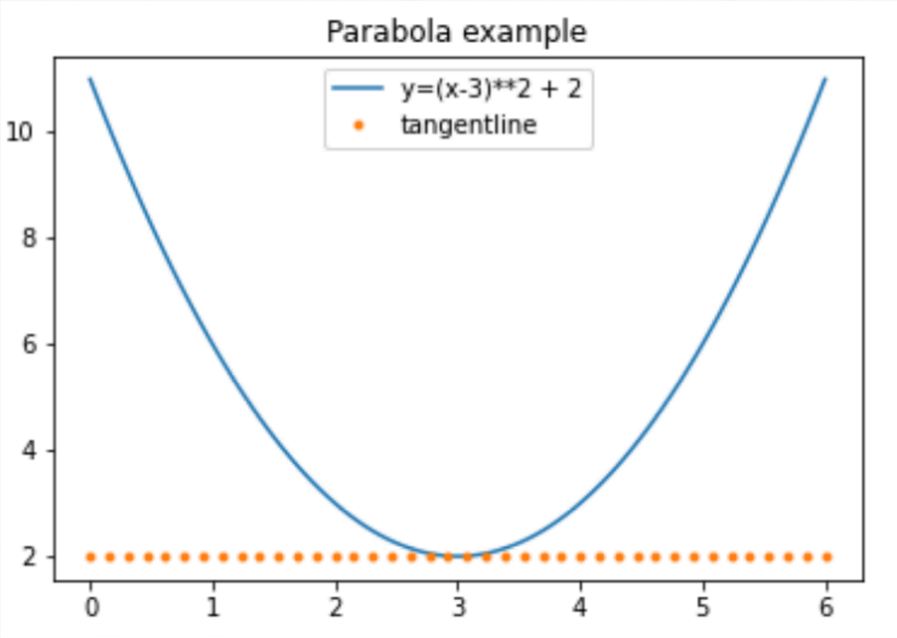
```python
def logistic(x):
    return 1/(1+np.exp(-x))
class NumpyLogReg(NumpyClassifier):

    def fit(self, X_train, t_train, eta = 0.1, epochs=10):
        """X_train is a Nxm matrix, N data points, m features
        t_train are the targets values for training data"""

        (k, m) = X_train.shape
        X_train = add_bias(X_train)

        self.weights = weights = np.zeros(m+1)

        for e in range(epochs):
            weights -= eta / k *  X_train.T @ (self.forward(X_train) - t_train)

    def forward(self, X):
        return logistic(X @ self.weights)

    def score(self, x):
        z = add_bias(x)
        score = self.forward(z)
        return score

    def predict(self, x, threshold=0.5):
        z = add_bias(x)
        score = self.forward(z)
        return (score>threshold).astype('int')
```

HOW TO MINIMISE ERRORS

USING DERIVATIVE FOR MINIMIZING VARIABLES

Through Derivatives we can find the value of variables that gives the minimal value for that equation, a simple example is given below

- $f(x) = (x-3)^2+2$
- $f'(x) = 2(x-3)$
- $f'(x) = 0 \text{ iff } x = 3$

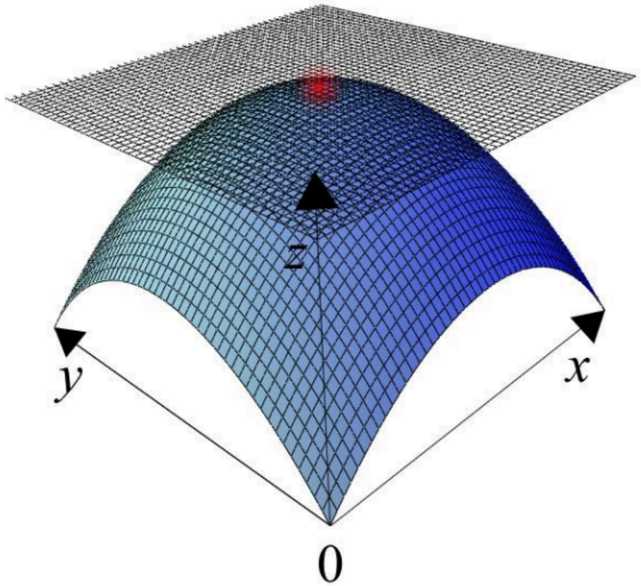


GRADIENT DESCENT AND HOW TO MINIMISE ERROR

In case of Linear Regression the MSE lose function has to be minimized. The MSE forms a Convex surface , with only a global minimum and no local minimas

$w_0, w_1, \ldots\ w_m$ that minimizes the MSE

$$\frac{1}{N}\sum_{j=1}^{N}\left(t_j - \sum_{i=0}^{m} w_i x_{ji}\right)^2$$



IMPLEMENTING THE GRADIENT DESCENT

- Input:
  - X, input, a $N \times m$ NumPy matrix:
    - N items, m features
  - T, corresponding target values,
    - $N \times 1$ column vector
    - (maybe it is given as a vector of length N and must be transformed)
  - Make a weight matrix, W,
    - $m \times 1$ column vector

- Forward step:
  - $Y = X@W$
- Update step:
  - $W = W - \eta \nabla f$
  - $W -= \eta X.T(Y - T)$
  - (η, *eta*, is a learning rate)

ONE VS REST CLASSIFICATION

ONE VS REST CLASSIFICATION APPROACH

Problem

Perceptron, Linear regression , Logistic regression all are binary classifiers, which means that they do not natively support classification tasks with more than two classes.
But these algorithms can be modified for multi-class classification problems , i.e. tasks that have more than two classes.

Solution:

Heuristic methods can be used to split a multi-class classification problem into multiple binary classification datasets and train a binary classification model each.There are two examples of heuristic method:

1. **One vs Rest (also known as One vs All)**
2. **One vs One**

ONE VS REST CLASSIFICATION APPROACH

Underlying Idea of OvR

It involves splitting the multi-class dataset into multiple binary classification problems. A binary classifier is then trained on each binary classification problem and predictions are made using the model that is the most confident.

For example, a multi-class classification problem with examples for each class ‘red,’ ‘blue,’ and ‘green‘, can be divided into the following three binary classification datasets.

- Binary Classification Problem 1: red vs [blue, green]
- Binary Classification Problem 2: blue vs [red, green]
- Binary Classification Problem 3: green vs [red, blue]

The model then predicts a class membership probability or a probability-like score. The argmax of these scores (class index with the largest score) is then used to predict a class.

ONE VS REST CLASSIFICATION APPROACH

Draw backs of One vs Rest

It requires one model to be created for each class. Hence, could be an issue for large datasets (e.g. millions of rows), slow models (e.g. neural networks), or very large numbers of classes (e.g. hundreds of classes).

ONE VS REST CLASSIFICATION APPROACH

Hot encoding

Hot Encoding

In one-hot encoding, each category is represented by a binary vector with a single 1 and the rest 0s, eliminating the issues caused by 1-of-N encoding:

1. It ensures that there's no implied order or hierarchy among the categories.
2. It represents each category as a distinct and independent entity.
3. It's compatible with a wide range of machine learning algorithms.

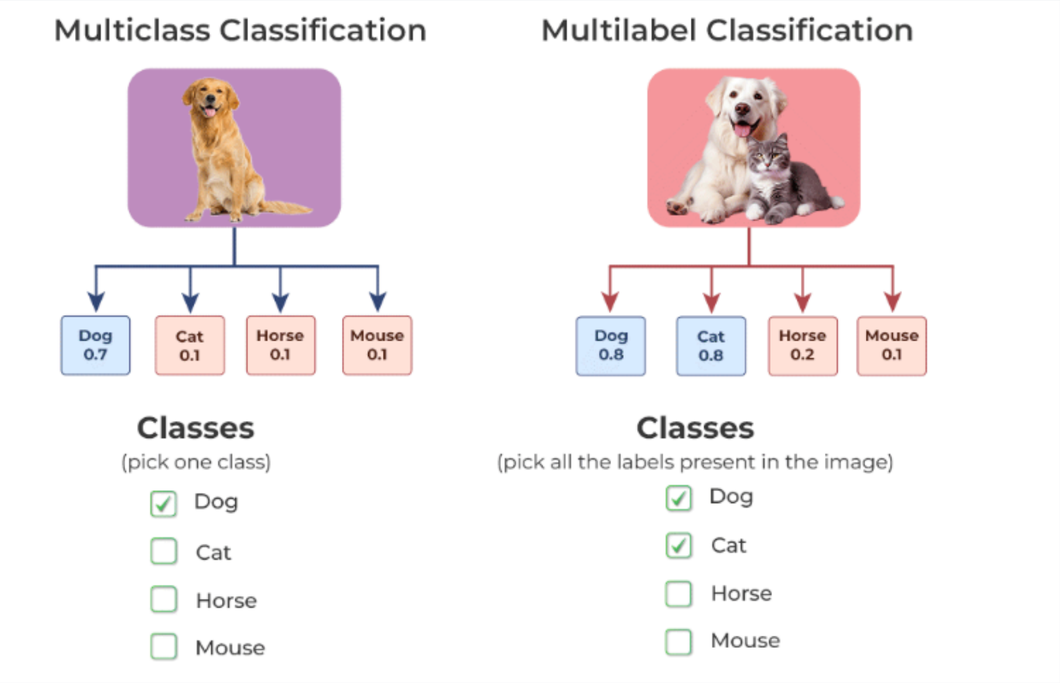
- 'apple' = (1, 0, 0, 0, 0, 0)
- 'tomato' = (0, 1, 0, 0, 0, 0)
- 'dog' = (0, 0, 1, 0, 0, 0)
- etc.

ONE VS REST CLASSIFICATION APPROACH

Multi-Label and Multi- Class Classification

Multiclass classification is a machine learning task where the goal is to assign instances to one of multiple predefined classes or categories, where each instance belongs to exactly one class.
Softmax Activation Function at the last layer

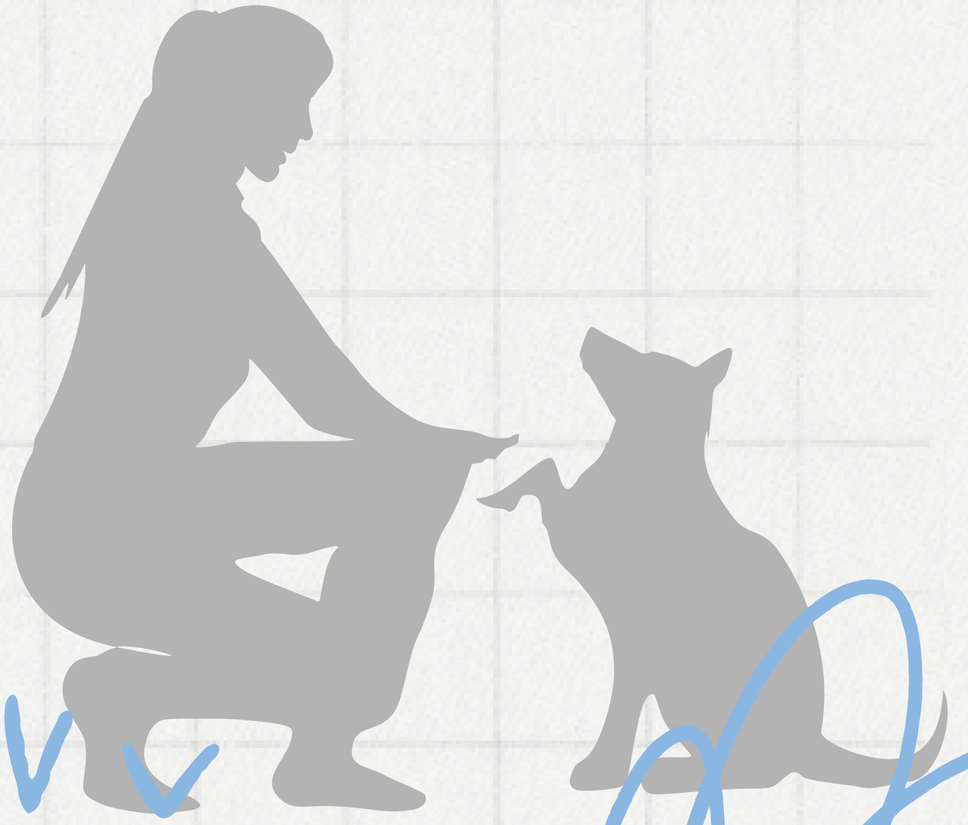
Multilabel classification is a machine learning task where each instance can be associated with multiple labels simultaneously, allowing for the assignment of multiple binary labels to the instance



Reinforcement learning

Reinforcement learning is an autonomous, self-teaching system that essentially learns by trial and error. It performs actions with the aim of maximising rewards, or in other words, it is learning by doing in order to achieve the best outcomes.

It is about taking suitable action to maximise reward and minimise penalty



Key elements of the Reinforcement learning

01. **State:** The input should be an initial state from which the model will start

02. **Action:** The model will take an action and move to a new state

03. **State:** The input should be an initial state from which the model will start

04. **Reward:** The best solution is decided based on the maximum reward

Agent

The entity that learns to interact with the environment. It makes decisions based on the information it receives.

Environment

The external system with which the agent interacts. It responds to the actions taken by the agent and provides feedback in the form of rewards or penalties.

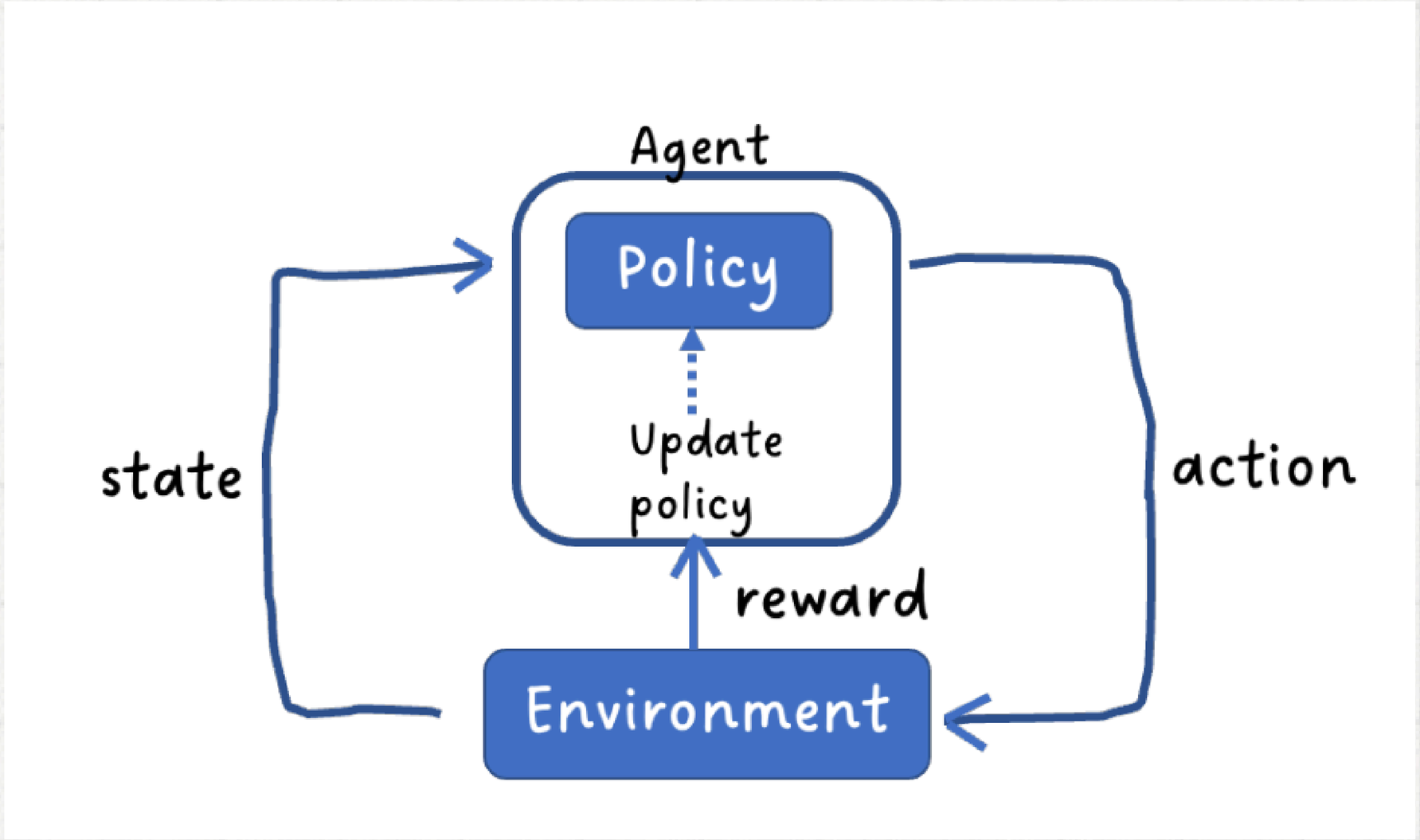
Policy

The strategy or rule that the agent uses to select actions in different states. It maps states to actions.

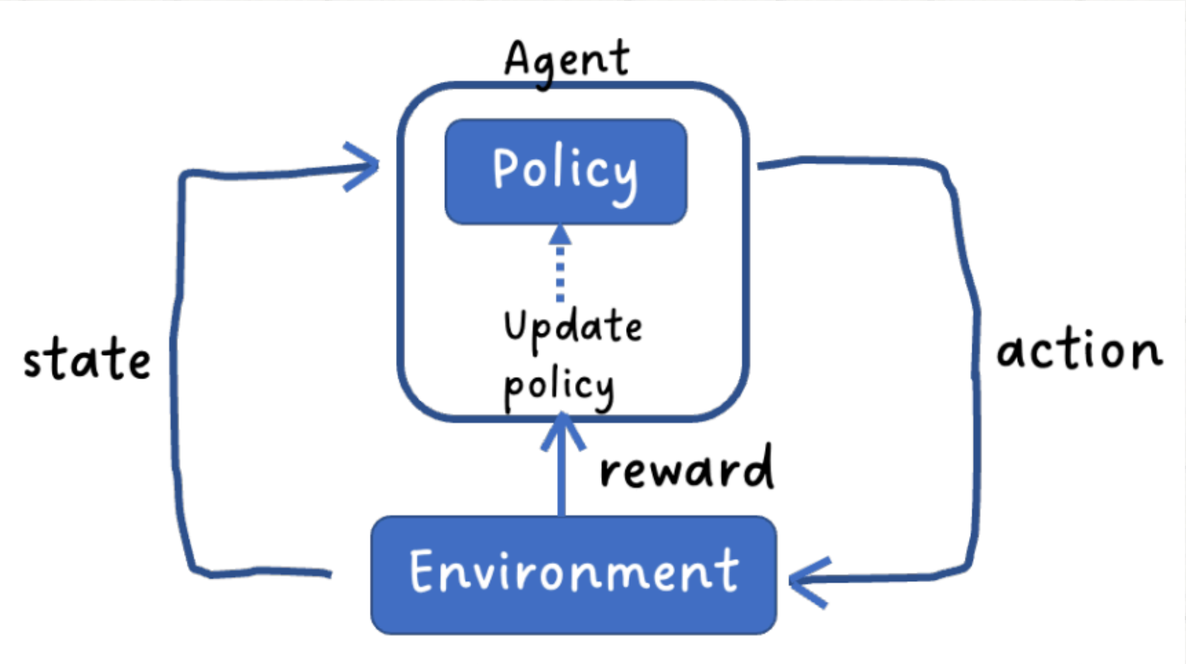
State

The current situation or configuration of the environment.

General algorithm



General algorithm – explanation



1. Initialization:
Initialise policy, value functions, or any other necessary parameters.

2. Interaction with Environment:
The agent observes the current state of the environment. Based on this observation and its current policy, the agent selects an action to take.

3. Action Execution and Reward Reception:
The selected action is executed in the environment. The environment transitions to a new state, and a reward is produced as feedback.The agent receives this reward from the environment.

4. Learning:
Using the received reward and the observed new state, the agent updates its policy and/or value functions to improve its decision-making strategy. This learning step incorporates the reward received into the agent's decision-making process, reinforcing actions that lead to higher rewards.

Iteration:
Steps 2-4 are repeated iteratively for multiple episodes or until convergence.

General algorithm


```
S[0] = <initial state>
for i in range(N):
    a = action(S[i])
    S[i+1] = newstate(S[i], a)
```

Result: S[0] S[1] S[2] ..... S[N-1]

Where:

a = action(S[i]) - implements the policy

S[i+1] = newstate(S[i], a) -implements the environment - internal logic not known

RL compared to supervised learning

Reinforcement learning	**Supervised learning**
Reinforcement learning is all about making decisions sequentially. In simple words, we can say that the output depends on the state of the current input and the next input depends on the output of the previous input	In Supervised learning, the decision is made on the initial input or the input given at the start
In Reinforcement learning decision is dependent, So we give labels to sequences of dependent decisions	In supervised learning the decisions are independent of each other so labels are given to each decision.
Example: Chess game,text summarization	**Example:** Object recognition,spam detetction