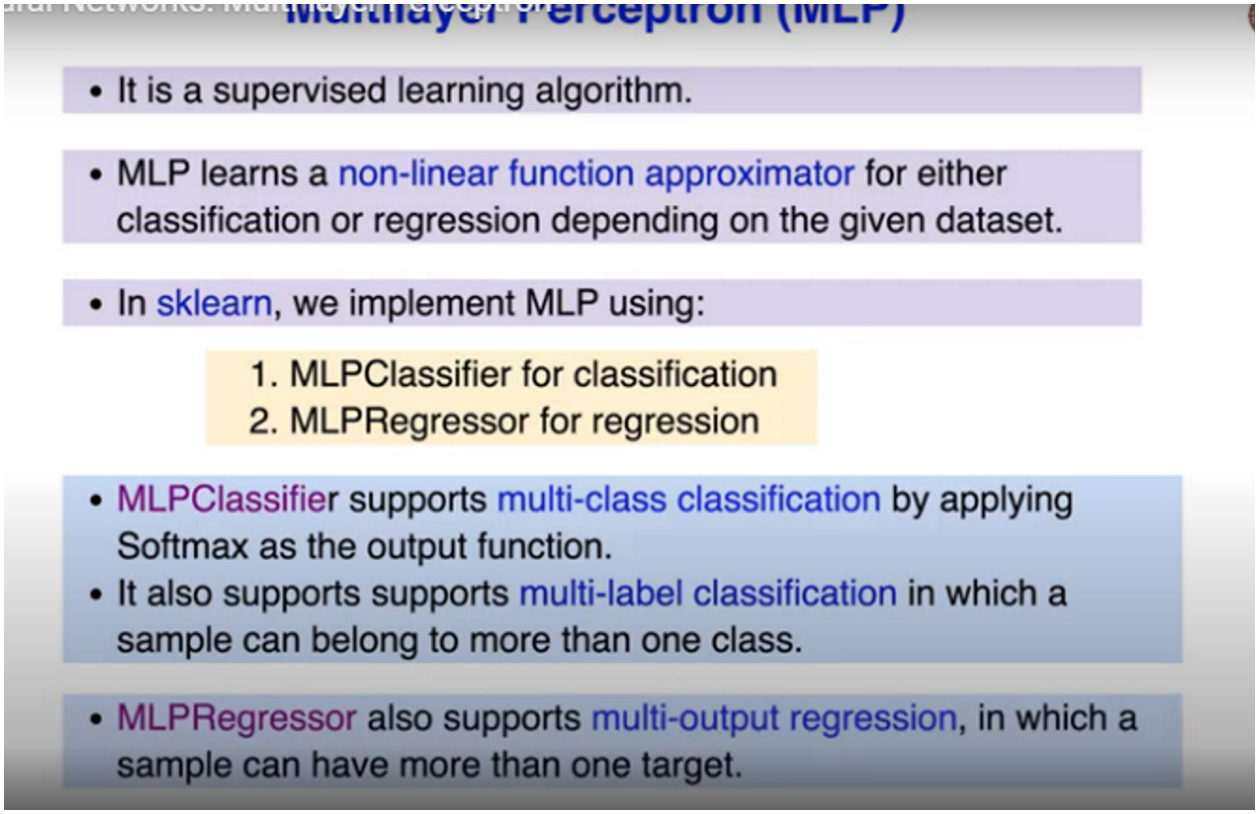
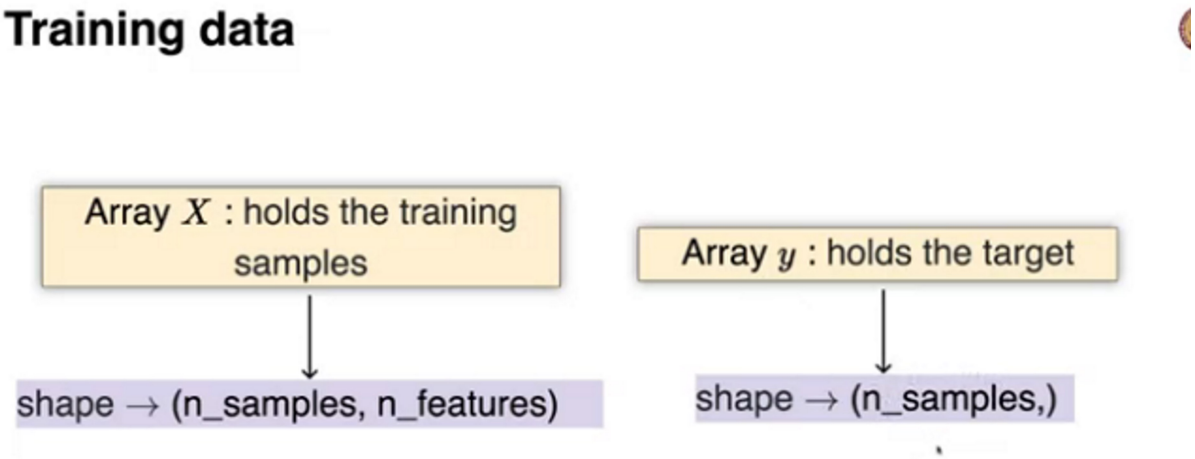
Neural Network

27 March 2023 10:41 AM





MLPClassifier

- How to implement MLPClassifier?

Step 1: Instantiate a MLP classifier estimator.

```
1 from sklearn.neural_network import MLPClassifier
2 MLP_clf = MLPClassifier()
```

Step 2: Call fit method on MLP classifier object with training feature matrix and label vector as arguments.

```
1 # Model training with feature matrix X_train and
2 # label vector or matrix y_train
3 MLP_clf.fit(X_train, y_train)
```

MLPClassifier

Step 3: After fitting (training), the model can make predictions for new samples (X_test) using two methods:

```
1 MLP_clf.predict(X_test)
2 MLP_clf.predict_proba(X_test)
```

predict →

- gives labels for new samples
- for example:

```
array([1, 0])
```

predict_proba →

- gives vector of probability estimates per sample
- for example:

```
array([1.967...e-04, 9.998...-01])
```

- MLPClassifier supports only the Cross-Entropy loss function

hidden_layer_sizes

- This parameter sets the number of layers and the number of neurons in each layer.

- It is a **tuple** where each element in the tuple represents the number of neurons at the i th position where i is the index of the tuple.
- The length of tuple denotes the total number of hidden layers in the network.

To create a 3 hidden layer neural network with 15 neurons in first layer, 10 neurons in second layer and 5 neurons in third layer:

```
1 MLPClassifier(hidden_layer_sizes=(15,10,5))
```

COPY

How to perform regularization in MLPClassifier?

- The alpha parameter sets L2 penalty Regularization parameter

alpha

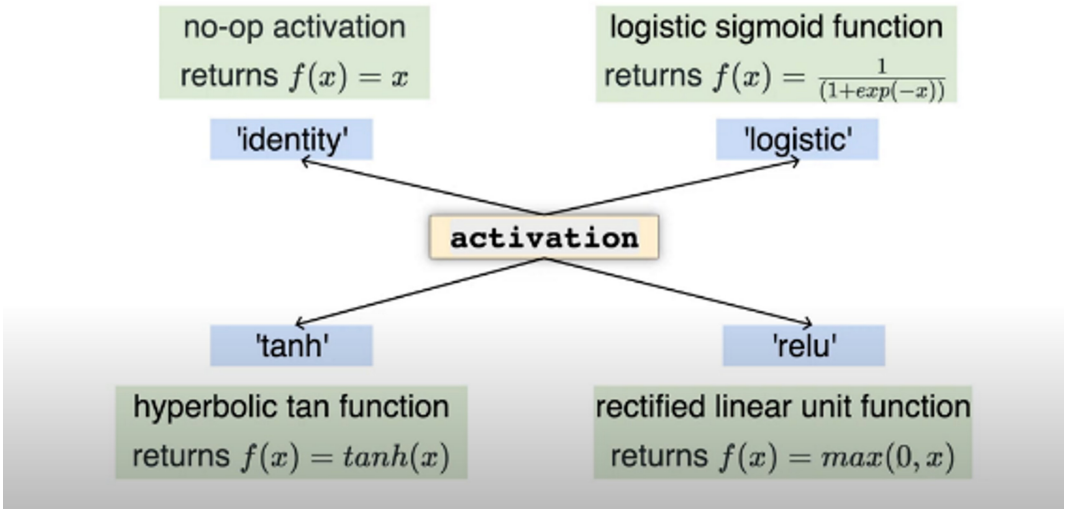
↓

float value

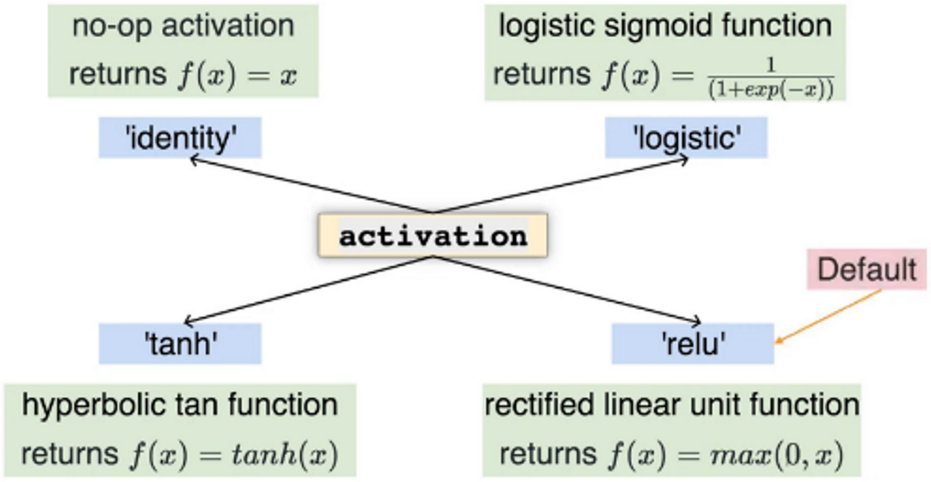
Default:

```
1 alpha = 0.0001
```

How to set the activation function for the hidden layers?

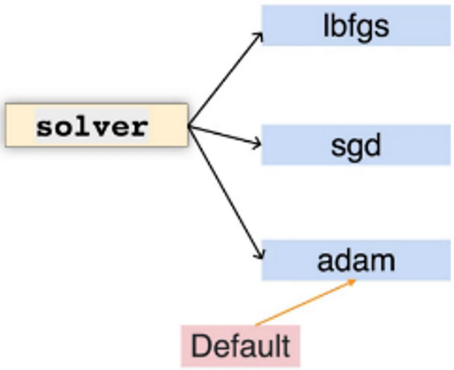


How to set the activation function for the hidden layers?



How to perform weight optimization in MLPClassifier?

- MLPClassifier optimizes the log-loss function using LBFGS or stochastic gradient descent



- If the **`solver`** is 'lbfgs', the classifier will not use minibatch.
- Size of minibatches can be set to other stochastic optimizers: **`batch_size`** (int)
  - default batch_size is 'auto'.

```
batch_size=min(200, n_sampl
```

How to view weight matrix coefficients of trained MLPClassifier?

`coefs_`

- It is a list of shape (n_layers - 1,)
- The i th element in the list represents the weight matrix corresponding to layer i.

Example:

- "weights between input and first hidden layer:"

```
print(MLP_clf.coefs_[0])
```

- "weights between first hidden and second hidden layer:"

```
print(MLP_clf.coefs_[1])
```

```
weights between input and first hidden layer:
[[-0.14203691 -1.18304359 -0.85567518 -4.53250719 -0.60466275]
 [ 0.69781111 -3.5850093  -0.26436018 -4.39161248  0.06644423]]

weights between first hidden and second hidden layer:
[[ 0.29179638 -0.14155284]
 [ 4.02666592 -0.61556475]
 [-0.51677234  0.51479708]
 [ 7.37215202 -0.31936965]
 [ 0.32920668  0.64428109]]
```

How to view bias vector of trained MLPClassifier?

`intercepts_`

- It is a list of shape (n_layers - 1,)
- The i th element in the list bias vector corresponding to layer $i+1$.

Example:

- "Bias values for first hidden layer:"

```
1 print(MLP_clf.intercepts_[0])
```

- "Bias values for second hidden layer:"

```
1 print(MLP_clf.intercepts_[1])
```

```
Bias values for first hidden layer:
```

Step 3: After fitting (training), the model can make predictions for new samples (X_test):

```
1 MLP_reg.predict(X_test)
```

- returns predicted values for new samples
- for example:

```
array([-0.9..., -7.1...])
```

```
1 MLP_reg.score(X_test,y_test)
```

- returns R^2 score
- for example:

```
0.45678889
```