

# Creating a layer

## Function Signature

```
def add_dense(  
    neurons: int,  
    activation: str = 'sigmoid',  
    dropout: float = 0.,  
    l1_penalty: float = 0.,  
    l2_penalty: float = 0.  
) -> None:
```

## Parameters

- `neurons` (int): The number of neurons in the dense layer. Required parameter.
- `activation` (str, default='sigmoid'): The activation function to use in the dense layer. Valid options are 'sigmoid', 'relu', 'leaky relu', 'tanh', 'softmax', and 'linear'
- `dropout` (float, default=0.): The dropout rate to use in the dense layer. This parameter should be a float between 0 and 1, where 0 means no dropout and 1 means all neurons are dropped.
- `l1_penalty` (float, default=0.): The L1 regularization penalty to use in the dense layer. This parameter should be a float greater than or equal to 0.

`l2_penalty` (float, default=0.): The L2 regularization penalty to use in the dense layer. This parameter should be a float greater than or equal to 0.

## Return Value

This function does not return anything. It modifies the `deeprai.models.FeedForward` instance by adding a dense layer with the specified parameters.

# Description

The `add_dense` function adds a dense layer to the `deeprai.models.FeedForward` instance. A dense layer is a layer of fully connected neurons where each neuron is connected to every neuron in the previous layer.

If this is the first layer added to the model, then `add_dense` will automatically treat it as an input layer and ignore any arguments other than `neurons`. This is because an input layer does not have an activation function, dropout, or regularization penalties.

If this is not the first layer added to the model, then `add_dense` will add a dense layer with the specified parameters to the model.

# Examples

Here is an example of how to use the `add_dense` function:

```
from deeprai.models import FeedForward

model = FeedForward()
model.add_dense(784)
model.add_dense(128, activation='relu', dropout=0.2)
model.add_dense(64, activation='sigmoid', l2_penalty=0.001)
```

This code creates a `FeedForward` model with an input with 784 neurons, adds a dense layer with 128 neurons, a ReLU activation function, and a 20% dropout rate, and then adds another dense layer with 64 neurons, a sigmoid activation function, and an L2 regularization penalty of 0.001.

---

Revision #1

Created 7 April 2023 06:50:37 by Kieran Carter

Updated 7 April 2023 07:18:17 by Kieran Carter