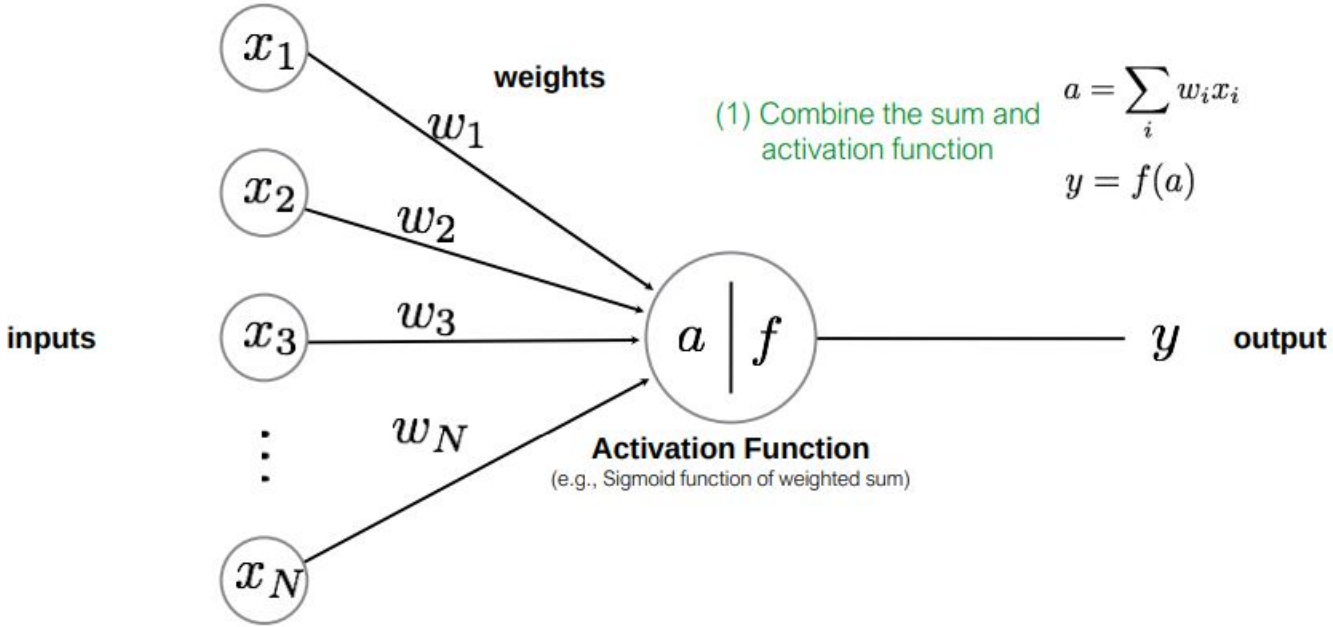
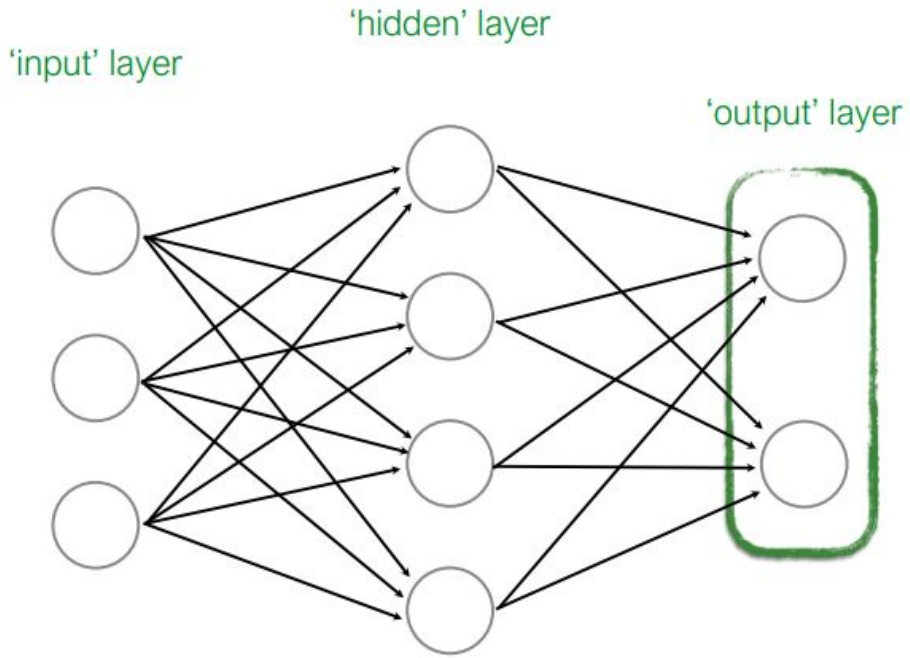
# Sieci neuronowe i regularyzacja w Kerasie

Marcin Wierzbiński

# Model Regresji Liniowej, ale inaczej



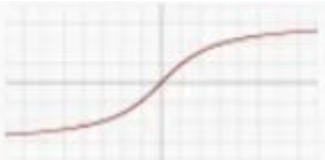
# Nazewnictwo



# Sieci neuronowe

Sieci neuronowe to rodzaj algorytmu uczenia maszynowego, który naśladuje sposób działania ludzkiego mózgu. Składa się on z neuronów, które przetwarzają dane wejściowe, a następnie przesyłają wynik do kolejnych neuronów w sieci. Sieci neuronowe mogą uczyć się poprzez obserwowanie przykładów i dopasowywanie wag między neuronami, aby osiągnąć pożądane wyniki.

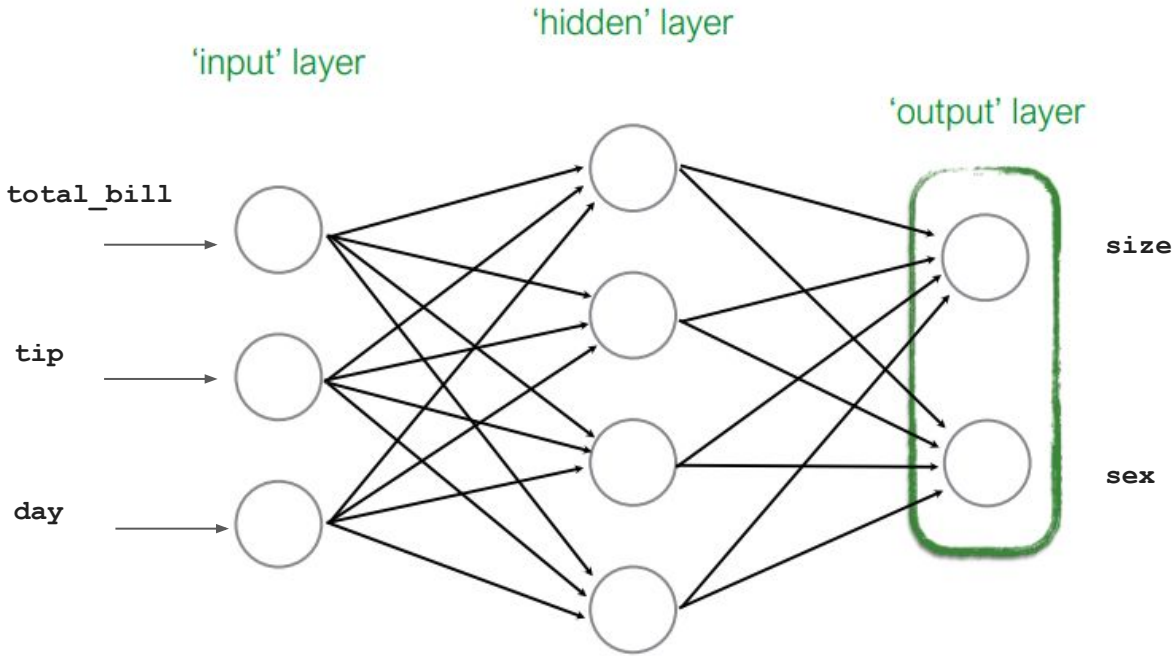
# Sprawdzone wybory funkcji aktywacji f

| | | |
|---|---|---|
| Rectified Linear Unit (ReLU) |  | $f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$ |
| |  | $f(x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$ |
| Logistic (a.k.a Soft step) |  | $f(x) = \frac{1}{1 + e^{-x}}$ |
| ArcTan |  | $f(x) = \tan^{-1}(x)$ |

# Tips dataset numeric

| | total_bill | tip | sex | smoker | day | time | size |
|---|---|---|---|---|---|---|---|
| **0** | 16.99 | 1.01 | 1 | 0 | 3 | 0 | 2 |
| **1** | 10.34 | 1.66 | 0 | 0 | 3 | 0 | 3 |
| **2** | 21.01 | 3.50 | 0 | 0 | 3 | 0 | 3 |
| **3** | 23.68 | 3.31 | 0 | 0 | 3 | 0 | 2 |
| **4** | 24.59 | 3.61 | 1 | 0 | 3 | 0 | 4 |

# Model dla tips



# Keras

Keras to biblioteka do tworzenia sieci neuronowych w języku Python. Jest to jedna z najpopularniejszych bibliotek do uczenia maszynowego, ponieważ jest łatwa w użyciu, ma intuicyjne API i obsługuje wiele różnych typów sieci neuronowych.

Tworzenie sieci neuronowych w Kerasie jest dość proste.

- Wystarczy zdefiniować model i warstwy, które mają być wykorzystane w sieci.
- Można to zrobić za pomocą następujących kroków:

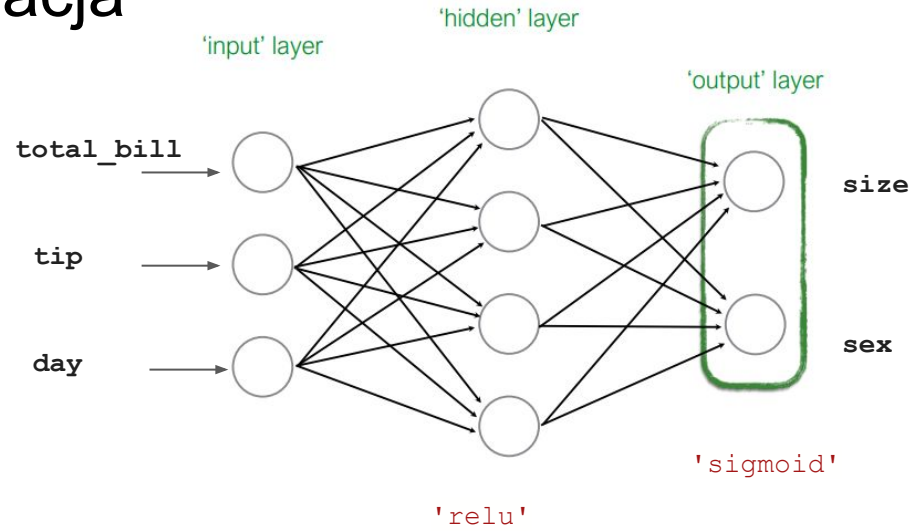
# Keras definicja modelu i kompilacja

Zdefiniuj model:

```
from keras.models import Sequential
from keras.layers import Dense

model = Sequential()
#dodanie warstw do modelu
model.add(Dense(4, activation='relu', input_dim=3))
model.add(Dense(2, activation='sigmoid'))

# kompilacja modelu
model.compile(loss='categorical_crossentropy',
optimizer=Adam(learning_rate=0.001), metrics=['accuracy'])
```



# Trening w Kerasie

```
from sklearn.preprocessing import StandardScaler

# skalowanie danych
scaler = StandardScaler()
X = scaler.fit_transform(X)

# podział danych na zbiór treningowy i testowy
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size= 0.2, random_state=42)


# trening modelu
model.fit(X_train, y_train, batch_size= 32, epochs=100, verbose=1)

# ocena modelu na zbiorze testowym
score = model.evaluate(X_test, y_test, verbose= 0)

# wypisanie wyniku
print('Test loss:', score[0])
print('Test MSE:', score[1])
```

# Overfitting

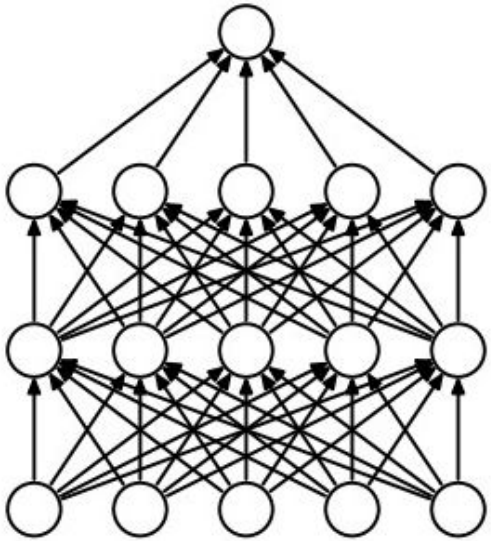
Overfitting to sytuacja, w której sieć neuronowa zbytnio dopasowuje się do danych treningowych i osiąga wysoką dokładność na tym zbiorze, ale niską dokładność na danych testowych.

Przyczynami overfitting mogą być:

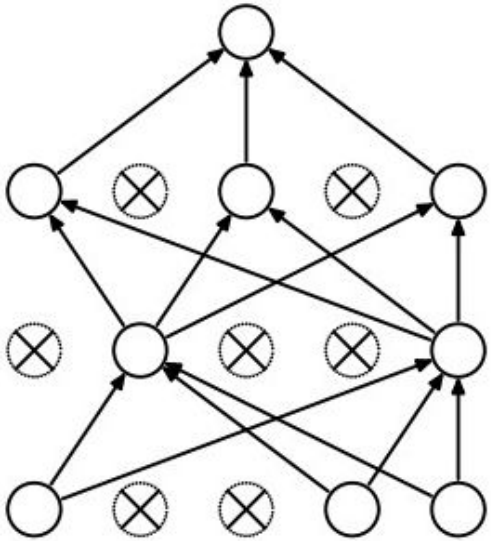
- zbyt duża liczba parametrów,
- zbyt mała liczba danych treningowych,
- niskiej jakości dane treningowe,
- nadmierna kompleksowość modelu,
- zbyt długi czas treningu, itp.

Do metod regularyzacji, które można zastosować w Keras, należą m.in.: regularyzacja L1, regularyzacja L2, dropout, regularyzacja kombinowana.

# Dropout w Kerasie



Standard Neural Net



After applying dropout

# Zadanie: Mnist model