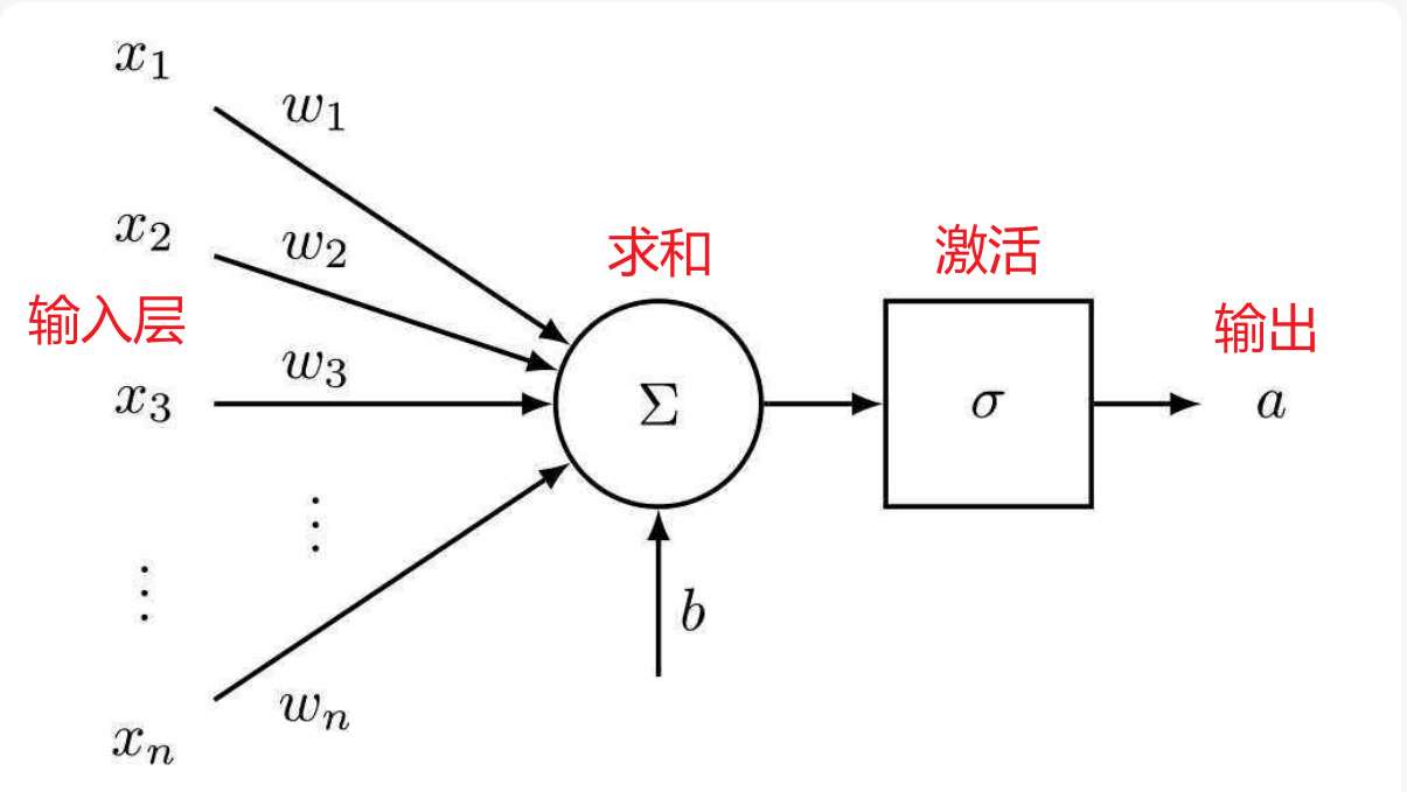
1、深度学习初窥之神经网络模型

1.1、单层神经网络

神经网络模型其实是在模仿人类大脑思考方式。神经元是神经系统最基本的结构，分为突起和细胞体两部分。人类大脑在思考时，神经元会接收外部的刺激，当传入的冲动使神经元的电位超过阈值时，神经元就从抑制状态转向兴奋，并将信号向下一个神经元传递。神经网络模型的思想就是通过构造人造神经元的方式模拟这一过程。

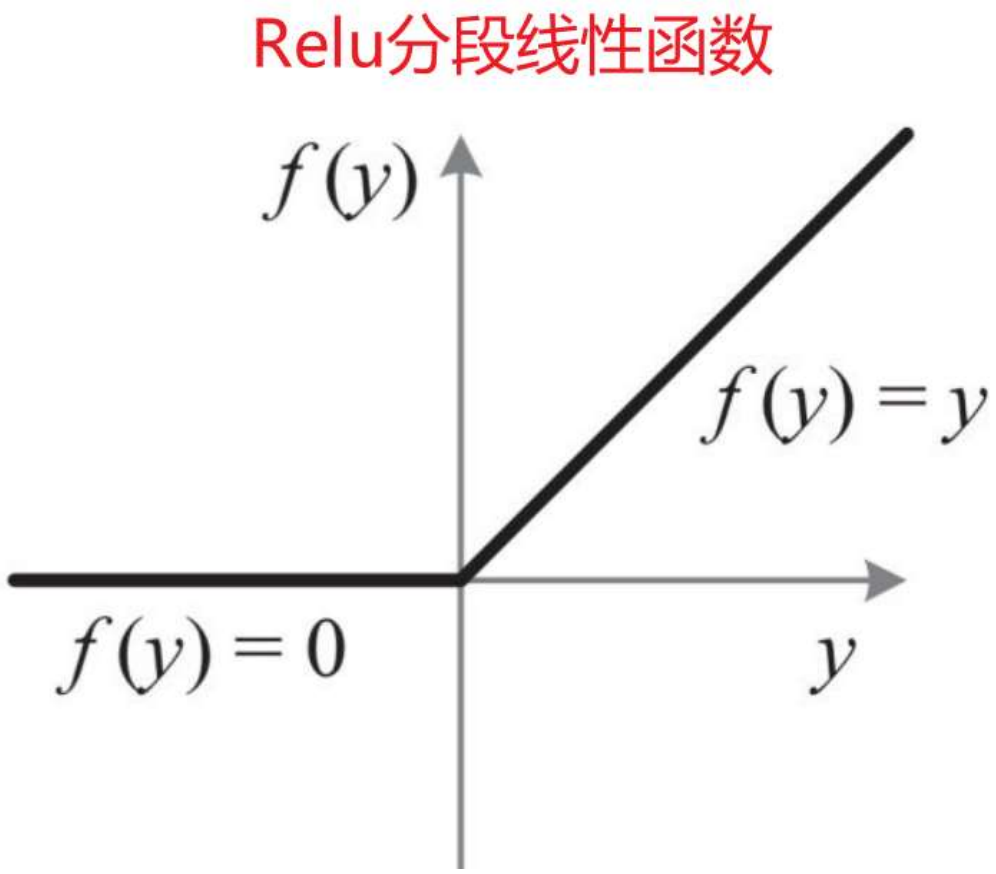


如上图所示，在一个简单的神经网络模型中有两组神经元，一组接收信号，一组输出信号。接收信号的一组神经元通过线性变换和非线性的激活函数转换来修改信号，并传递给下一组神经元。上图描述的就是大脑，基本工作模式。

1.2、激活函数

1.2.1、Relu激活函数

如下图所示，Relu函数是一种分段线性函数，当输入为正数时，输入多少输出多少；当输入为负数时，输出为0。



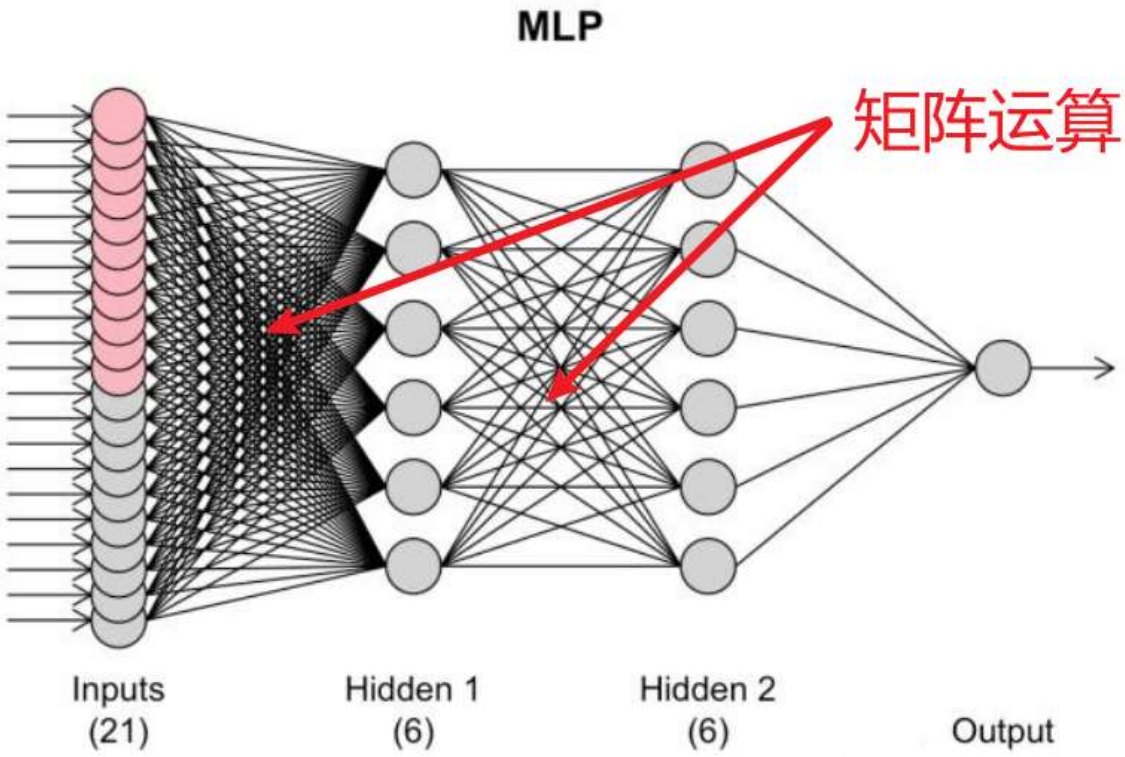
1.2.2、其他激活函数

常用的有：Sigmoid、Tanh、Softmax。详情见老师个人博客（之前课程中介绍过）。

代码点亮人生

1.3、多层神经网络

在实际应用中，神经网络模型往往不是单层的，而是如下图所示的多层神经网络模型（正如大脑的结构）。在多层神经网络模型中，输入层和输出层之间可以有多个隐含层，层与层之间连接，信号不断地从上一层传递到下一层，最后由输出层输出。这样，就可以处理复杂问题了！这正是深度学习中“深度”的由来！



2、神经网络模型简单代码实现

2.1、神经网络分类模型

```
from sklearn.neural_network import MLPClassifier
import pandas as pd
X = [[1, 0], [5, 1], [6, 4], [4, 2], [3, 2]]
y = [0, 1, 1, 0, 0]
mlp =MLPClassifier(hidden_layer_sizes=(100,50,30),max_iter=1000)
mlp.fit(X, y)
y_pred = mlp.predict(X)
result = pd.DataFrame()  # 创建一个空DataFrame
result['预测值'] = list(y_pred)
result['实际值'] = list(y)
result
```

2.2、神经网络回归模型

```
from sklearn.neural_network import MLPRegressor
X = [[1, 2], [3, 4], [5, 6], [7, 8], [9, 10]]
y = [1, 2, 3, 4, 5]
```

```
# 设置random_state随机状态参数，使得每次训练的模型都是一样的
model = MLPRegressor(random_state=1024)
model.fit(X, y)


print(model.predict([[5, 5]]))
```

3、用户评论情感分析

3.1、数据读取

```
import pandas as pd
data = pd.read_excel('产品评价.xlsx')
display(data.head(),data.shape)
```

3.2、中文分词

演示代码

```
# jieba库分词示例
import jieba
word = jieba.cut('我爱北京天安门')
for i in word:
    print(i)
```

用户评论分词

```
# 这里先演示第一条评论的分词效果
import jieba
word = jieba.cut(data.iloc[0]['评论'])
result = ' '.join(word)
print(result)
```

遍历所有评论分词

```
# 遍历整张表格，对所有评论进行分词
words = []
for i, row in data.iterrows():
    word = jieba.cut(row['评论'])
    result = ' '.join(word)
```

```
    words.append(result)
words[:3]
```

3.3、文本向量化

演示代码

```
# 文本向量化CountVectorizer()函数的使用技巧：使用示例
from sklearn.feature_extraction.text import CountVectorizer
test = ['手机 外观 亮瞎眼', '电脑 速度 YYDS ']
vect = CountVectorizer()
X = vect.fit_transform(test)
X = X.toarray()
display(X)
words_bag = vect.vocabulary_
print(words_bag)
```

文本向量化

```
# 实际应用
from sklearn.feature_extraction.text import CountVectorizer
vect = CountVectorizer()
X = vect.fit_transform(words)
X = X.toarray()
print(X)
words_bag = vect.vocabulary_
print(words_bag)
```

3.4、目标提取

```
y = data['评价']
y.head()
```

3.5、神经网络建模与应用

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, random_state=1)
from sklearn.neural_network import MLPClassifier
mlp = MLPClassifier()
```

```
mlp.fit(X_train, y_train)
y_pred = mlp.predict(X_test)
print(y_pred)
```

真实值和预测值对比

```
result = pd.DataFrame()  # 创建一个空DataFrame
result['预测值'] = list(y_pred)
result['实际值'] = list(y_test)
result
```

3.6、准确率与应用

```
# 获取预测准确度
from sklearn.metrics import accuracy_score
score = accuracy_score(y_pred, y_test)
print('------------',score)
# 通过模型自带的score()函数也可以获取预测准确度
mlp.score(X_test, y_test)
```

应用检验

```
# 检验：商品真棒，我非常喜欢，这是一次满意的购物，特别开心
# 商品垃圾，物流不给力，商品破损，体验不好
comment = input('请输入您对本商品的评价：')
comment = [' '.join(jieba.cut(comment))]
print(comment)
X_try = vect.transform(comment)
y_pred = mlp.predict(X_try.toarray())
print(y_pred)
```

3.7、朴素贝叶斯算法

```
# 朴素贝叶斯模型对比
from sklearn.naive_bayes import GaussianNB,MultinomialNB,BernoulliNB
nb_clf = BernoulliNB()
nb_clf.fit(X_train,y_train)


y_pred = nb_clf.predict(X_test)
```

```
print(y_pred)


from sklearn.metrics import accuracy_score
score = accuracy_score(y_pred, y_test)
print(score)
```