

# 音频信号处理 及基于音频的深度学习教程

---

Audio Signal Processing  
Audio-based Deep Learning Tutorials

b站: 今天声学了吗

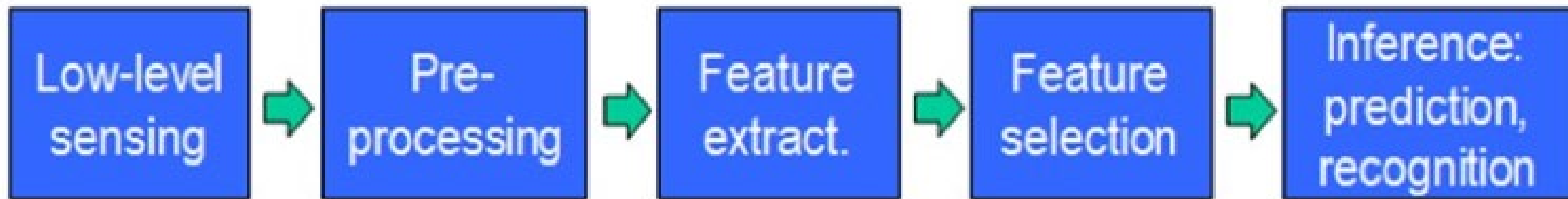
公众号: 今天声学了吗

邮箱: 1319560779@qq.com

## 机器学习介绍

### ● 什么是机器学习？

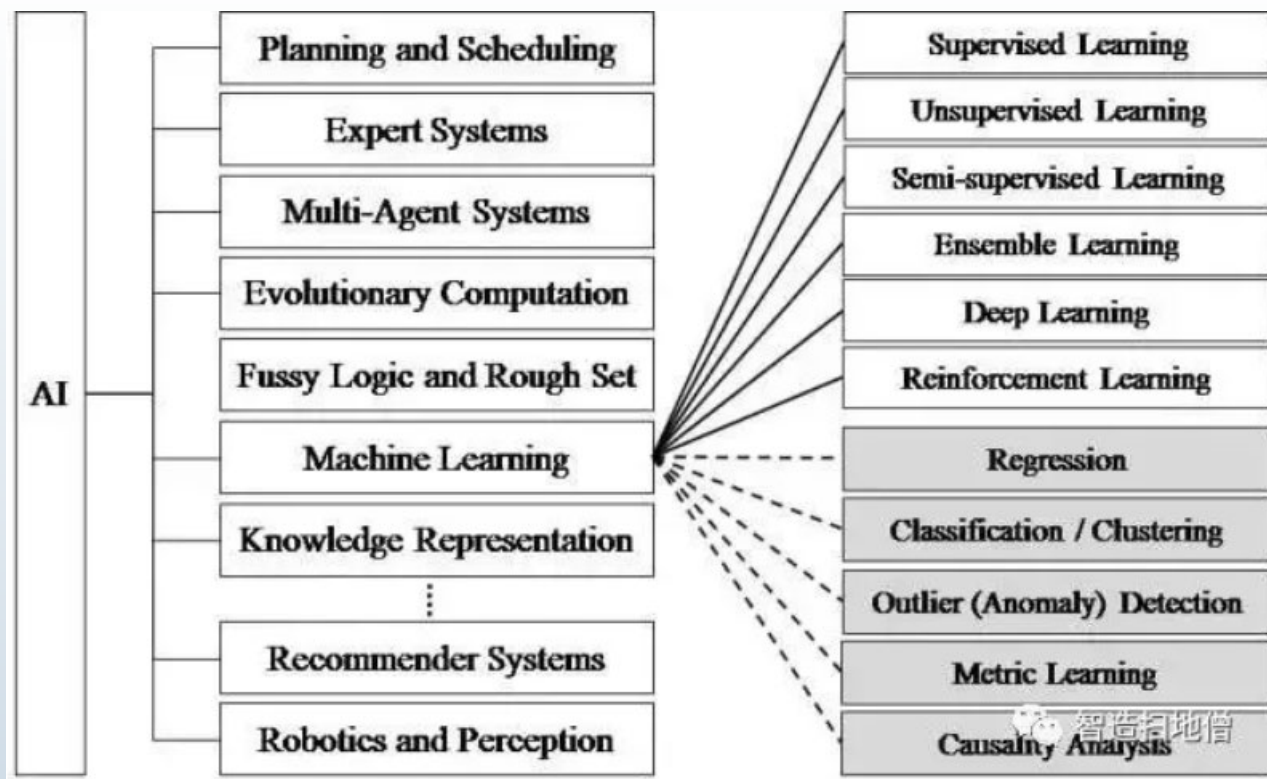
1. 机器学习 (Machine Learning) 是计算机模拟人类的学习行为，以获取新的知识或技能。实际上机器学习是构建函数的过程，给定一个输入后，输出我们想要的结果。
2. 流程：通过传感器来获得数据、预处理、特征提取、特征选择，最后进行推理、预测或者识别，这就是机器学习的过程。



## 机器学习介绍

### ● 机器学习与AI、Deep Learning的关系是什么？

机器学习是人工智能的一个分支，而深度学习又是机器学习的下属分支，因此它们是环环紧扣的。



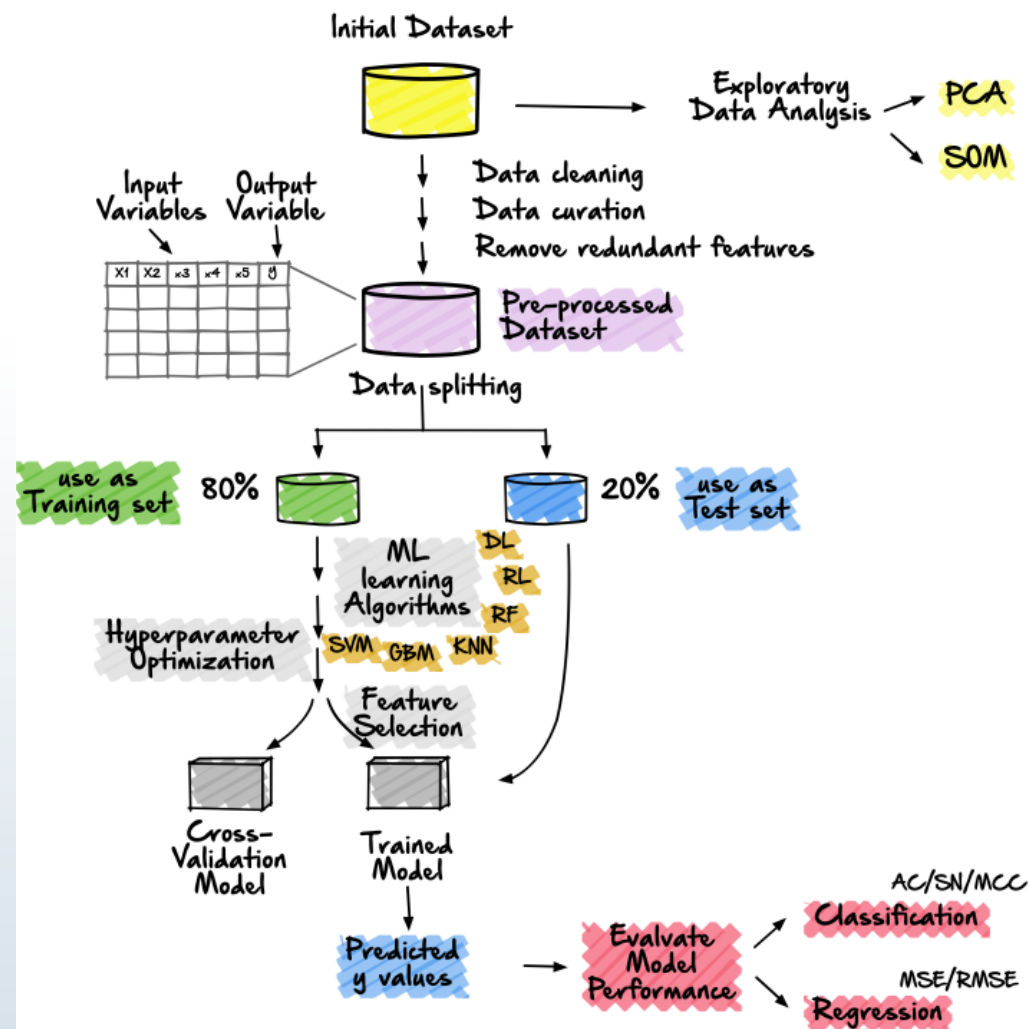
# CHAPTER 3--构建神经网络

## 机器学习介绍

### ● 机器如何学习？

1. 准备充足的数据集
2. 对数据进行分析，人为了解数据集的分布形式与参考值（平均值、方差、中位数等）
3. 对数据集做预处理。
4. 分割数据集，将已经处理好的数据集分割成train set和test set。
5. 依据数据集的target变量建立有映射关系的模型。其中最核心的部分是参数调优，也就是让机器输出理想结果。
6. 机器学习得到输出，在监督学习中，常见的任务是分类和回归。
7. 完成学习任务，评价结果

Machine Learning Model (by zomi) 🌈





## 传统机器学习介绍

### ● 为什么需要充足的数据集？

因为学习的样本越多，机器对于这个领域获取的信息越多，输出的结果越理想，偏差越少。

### ● 为什么要人为了解数据集的分布，并作预处理？

方便机器明白这组数据集中哪些信息是重要的，需要重点采集。

### ● 机器怎么建立这种映射关系得到输出？

通过正向和反向传播两个过程，根据**大量训练样本**，机器学习统计他们之中的规律，从而对未知事件做预测，并且生成模型的输出值与期望值越来越拟合。

- 正向传播：通过模型的每一层运行输入数据以进行预测，prediction。

- 反向传播：使用模型的预测和已知的标签来计算误差（有监督学习），

然后通过模型**反向传播**，利用误差导数在**梯度下降处**得到最小误差来优化模型参数完成学习。

# CHAPTER 3--构建神经网络

今天声学了吗🔊

## 传统机器学习介绍

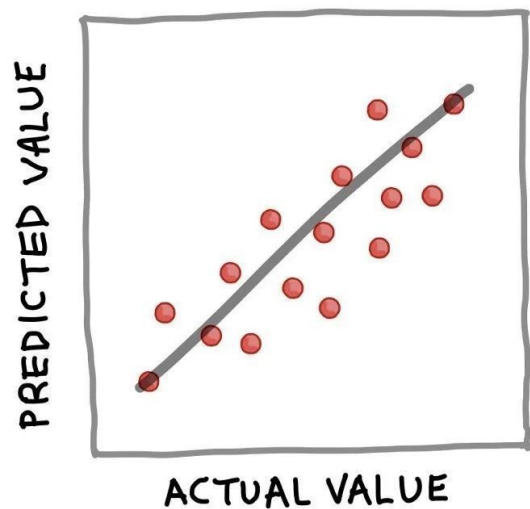
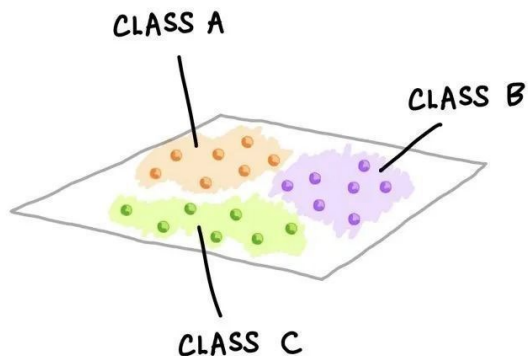
### ● 分类与回归是什么，有什么区别？

分类模型和回归模型本质一样，都是要建立映射关系，利用一个训练好的模型将一组变量作为输入，输出其对应的标签。

- 分类：输出对象是离散变量，并且输出空间不是同一个度量空间。上图是由不同颜色和标签表示的三个类，每一个小的彩色球体代表一个数据样本。
- 回归：输出对象是连续变量，输出空间是同一个度量空间。下图的函数为这个映射关系，离散的点为实际值(actual value)，对应到函数上的值为输出值(predicted value)

### ● 为什么要分割数据集？

这是为了在开发模型流程中，希望生成的模型能在新的、未见过的数据上表现良好。



# CHAPTER 3--构建神经网络

## 现代机器学习介绍

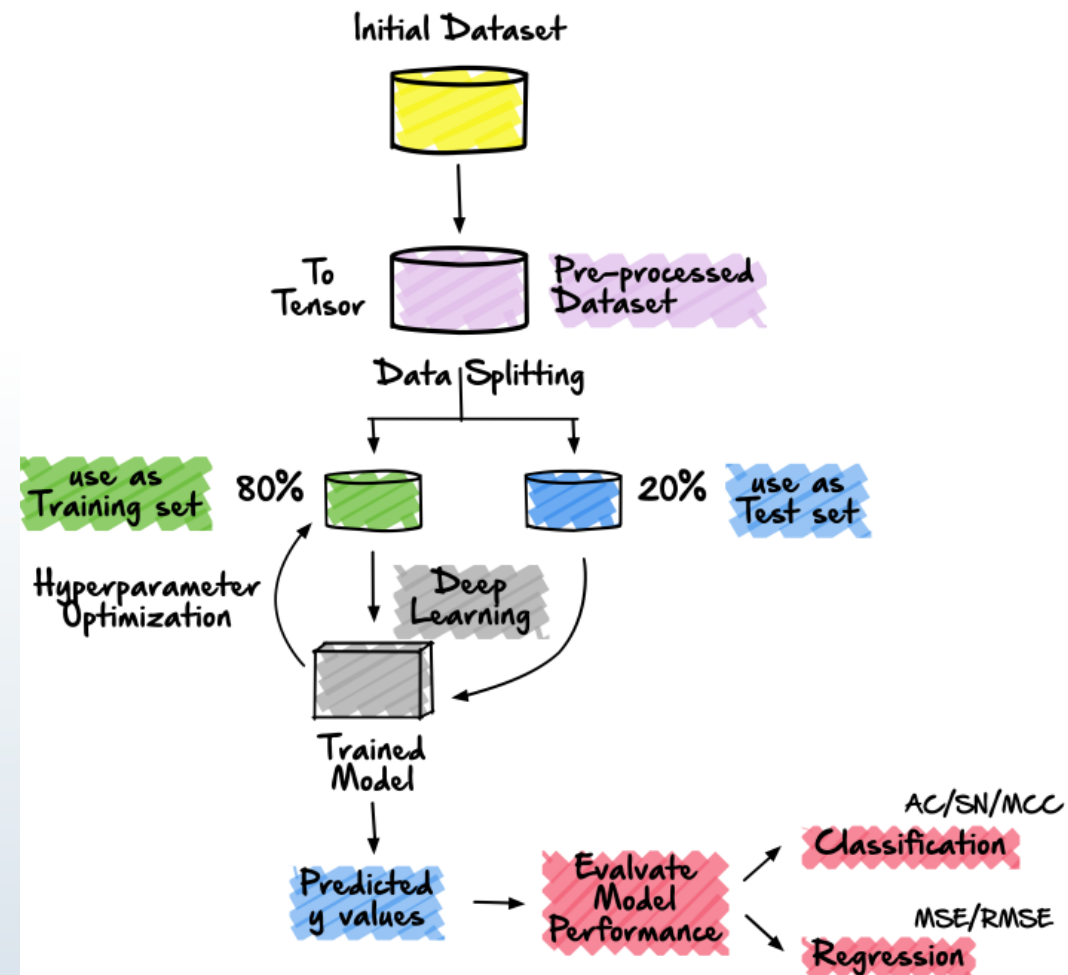
### ● 现代机器学习也是这样吗？

基本思想是一致的，区别在于不需要人为分析数据，提取特征。

### ● 现代机器学习只有优点吗？

1. 深度学习也就是现代机器学习，不需要人为去提取数据特征，而是通过神经网络自动对数据进行高维抽象学习，其输出的结果准确率大大提升，并且可以完成很多复杂的任务。
2. 但是调参工作变得更加繁重。因为引入了更加深、更复杂的网络模型结构，所以需要定义神经网络模型结构、确认损失函数、确定优化器，最后就是反复调整模型参数的过程。并且很多模型是黑盒模型。

## Deep Learning Model (by zomi)





# CHAPTER 3--构建神经网络

## 机器学习的分类

### ● 传统机器学习

人为进行提取数据

流程：1) 数据集准备、2) 探索性的对数据进行分析、3) 数据预处理、4) 数据分割、5) 机器学习算法建模、6) 选择机器学习任务，评价结果

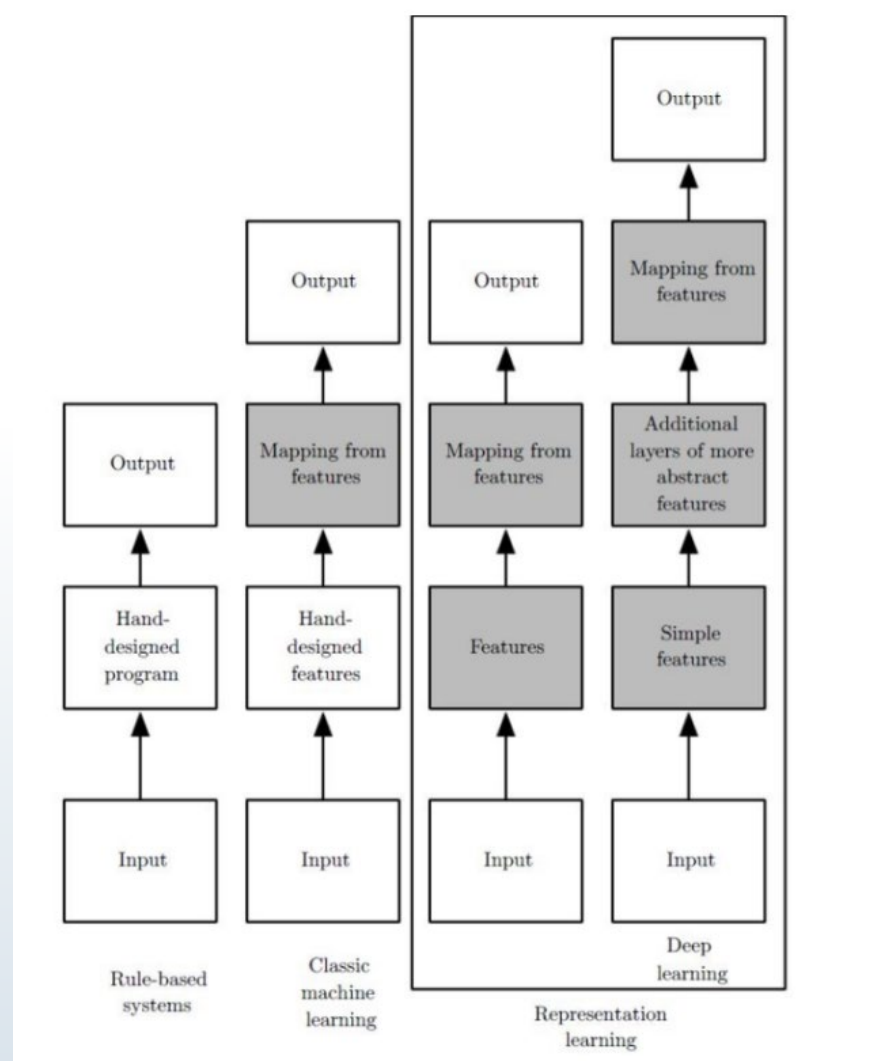
常见的算法：支持向量机、线性判别分析LDA、主成分分析PCA、决策树算法、聚类算法、波尔兹曼机（图模型）

### ● 现代机器学习

机器通过学习生成模型，机器对数据进行提取。

流程：1) 数据集准备、2) 数据预处理、3) 数据分割、4) 定义神经网络模型，5) 训练网络。

类别：浅层学习与深度学习





## 机器学习的其他分类

### ● 有监督学习

- 概念：通过已有的训练样本去训练得到一个最优模型，再利用这个模型将所有的输入映射为相应的输出，对输出进行简单的判断从而实现预测和分类的目的，也就具有了对未知数据进行预测和分类的能力。
- 监督学习中的数据中是提前做好了分类信息的，它的训练样本中是同时包含有特征和标签信息的，因此根据这些来得到相应的输出。简单来说，就像有标准答案的练习题，然后再去考试，相比没有答案的练习题然后去考试准确率更高。

### ● 无监督学习

- 概念：训练样本的标记信息未知，目标是通过通过对无标记训练样本的学习来揭示数据的内在性质及规律，为进一步的数据分析提供基础。

### ● 强化学习

- 概念：根据每一次输出的结果（1/0），改进其模型。

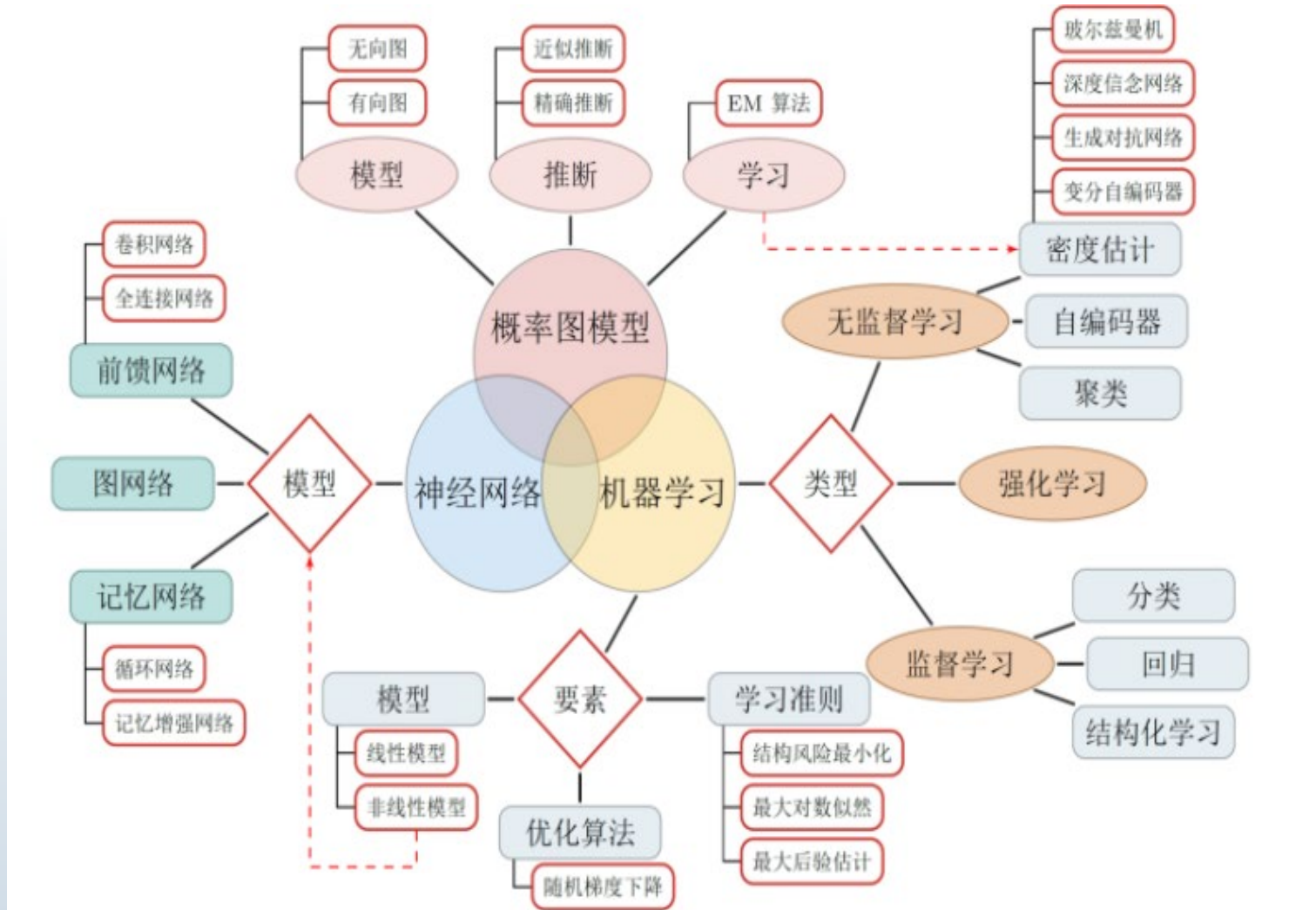
## 机器学习的几种分类

● 有、无监督学习区别

	是否需要训练样本	是否对样本预先处理	目标比较	黑盒体系比较	训练方法
有监督学习	√	√	分类	不透明性 不可解释性	输出与期望误差最小化
无监督学习	×	×	聚合	透明性 可解释性	最大似然估计
常见算法	有监督学习	支持向量机(Support Vector Machine, SVM), 朴素贝叶斯(Naive Bayes), 逻辑回归(Logistic Regression), K近邻 (K-Nearest Neighborhood, KNN), 决策树(Decision Tree), 随机森林(Random Forest), AdaBoost以及线性判别分析(Linear Discriminant Analysis, LDA)等			
	无监督学习	Apriori算法以及k-Means算法。			

## 机器学习小结

机器学习是一个大的领域，可分为**传统机器学习**和**现代机器学习**。其中现代机器学习有一个分支叫深度学习，深度学习形成的网络框架是神经网络，这个模型框架在传统机器学习的基础上得以发展（线性与非线性模型），他的核心理念是特征提取，按照模型结构可以分为前馈网络、图网络和记忆网络。其他都是沿用了传统机器学习的算法思想。



# 谢谢大家

---

Audio Signal Processing  
Audio-based Deep Learning Tutorials

b站：今天声学了吗

公众号：今天声学了吗

邮箱：1319560779@qq.com



# 音频信号处理 及基于音频的深度学习教程

---

Audio Signal Processing  
Audio-based Deep Learning Tutorials

b站: 今天声学了吗

公众号: 今天声学了吗

邮箱: 1319560779@qq.com

## 神经网络概述

### ● 什么是神经网络？

是现代机器学习的框架，对输入数据执行多个嵌套函数的集合。其基本单元为感知器：包含参数 $w$ 和 $b$ 的激活函数，神经网络的最基本单元。

### ● 流程：

将输入信号都按一定比例放大或缩小（乘以分配给该特定输入的权重值 $w$ ）。

累加所有带权重的输入并且加偏移量。

非线性激活，通过将结果馈送到激活函数（也称为传递函数）得到输出。

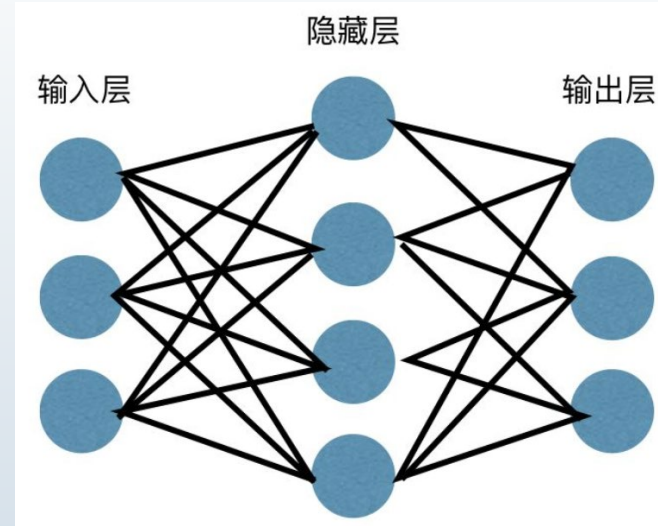
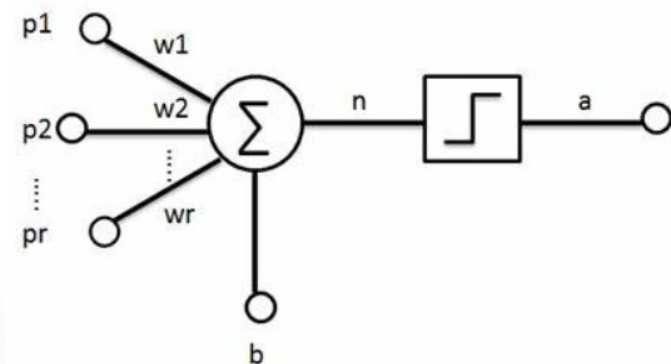
神经网络:利用多个感知器构造的，用于提取数据特征信息的模型。

### ● 结构：

输入层

隐藏层:激活函数(输入\*权重+偏置) = 下一层神经元。激活函数有多种，神经函数的复杂性来自于激活函数。

输出层：输出个数是分类的种类个数，每一个输出其实就是一个二分类：是/不是。



## 神经网络概述

### ● 给网络喂什么？

输入——含有特征的对象。具体类型取决于当前对象的类型以及需要完成的任务。如图像识别时输入的是图像信息，语音识别时绝大多数输入的是语谱图，也就是之前的信号处理中得到的结果。

### ● 网络如何得知自己吃了什么？

通过分解识别。当网络食入数据后，对这些“食物”进行分解识别，从“味觉、触感....”等多个维度来判断这个“食物”是什么。分解识别的过程也就是特征提取。

#### • 步骤：

根据原始数据，将其分成N个碎片中，选取一组碎片， $S[k]$ ，通过叠加的办法，合成出一个新的碎片，而这个新的碎片，应当与目标碎片  $T$ ，尽可能相似 同时， $S[k]$  的数量尽可能少。因此可以通过迭代的思路找到最适合的碎片集与目标集近似。

比如在识别猫时，将猫的图片分成众多的碎片，一部分是表示猫眼，一部分表示胡须，一部分表示身体...这些信息累加起来最终得到识别结果是猫

• 所以复杂图形，往往由一些基本结构组成。可以用一组正交的基本结构，将目标图像按照权重调和而成。

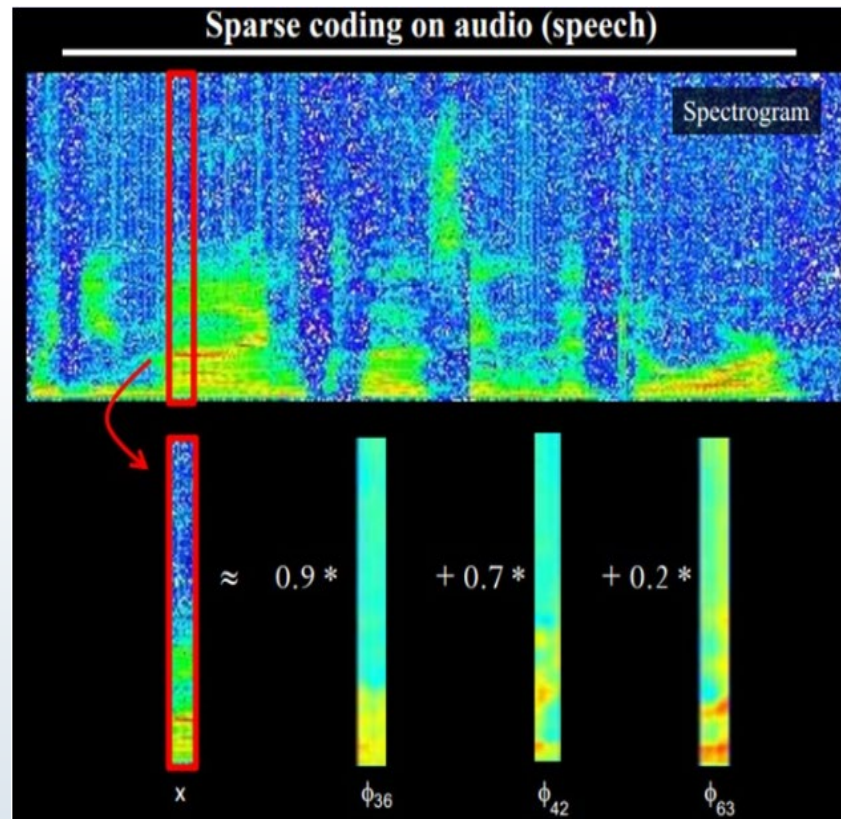
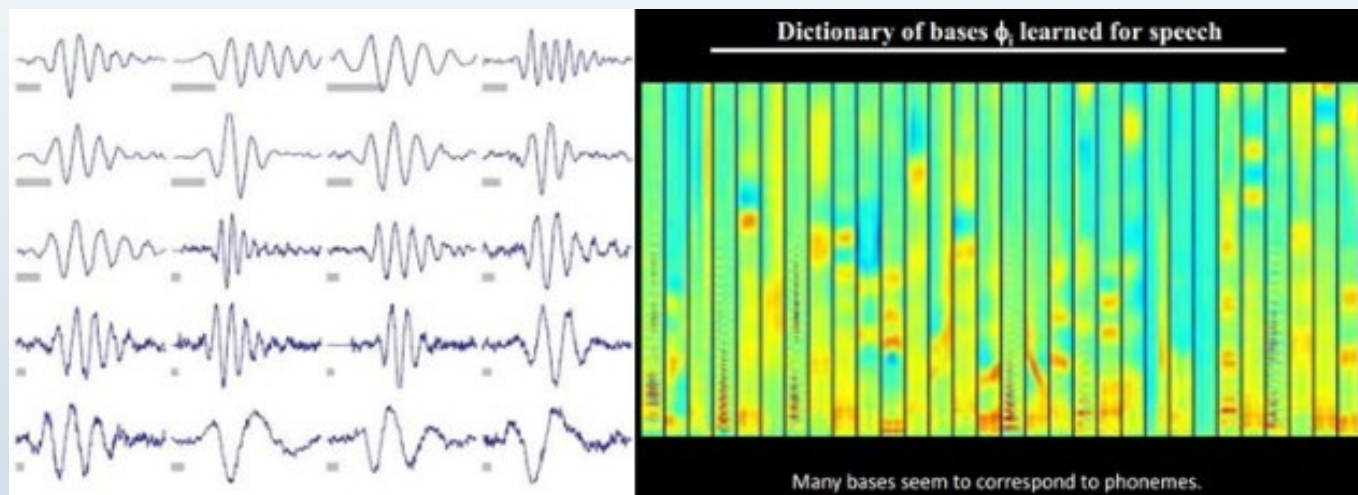


## 神经网络概述

### ● 处理音频的网络如何得知自己吃了什么？

因为复杂图形，往往由一些基本结构组成。可以用一组正交的基本结构，将目标图像按照权重调和而成，音频的语谱图也是如此。

在音频信号中，在未标注的声音中发现了20种基本的声音结构，因此其余的声音可以由这20种基本结构合成。这也就是利用图像识别的思路完成语音识别的基础。





## 神经网络概述

- 每层网络为什么有那么多神经元？（为什么需要这么多级消化过程？）

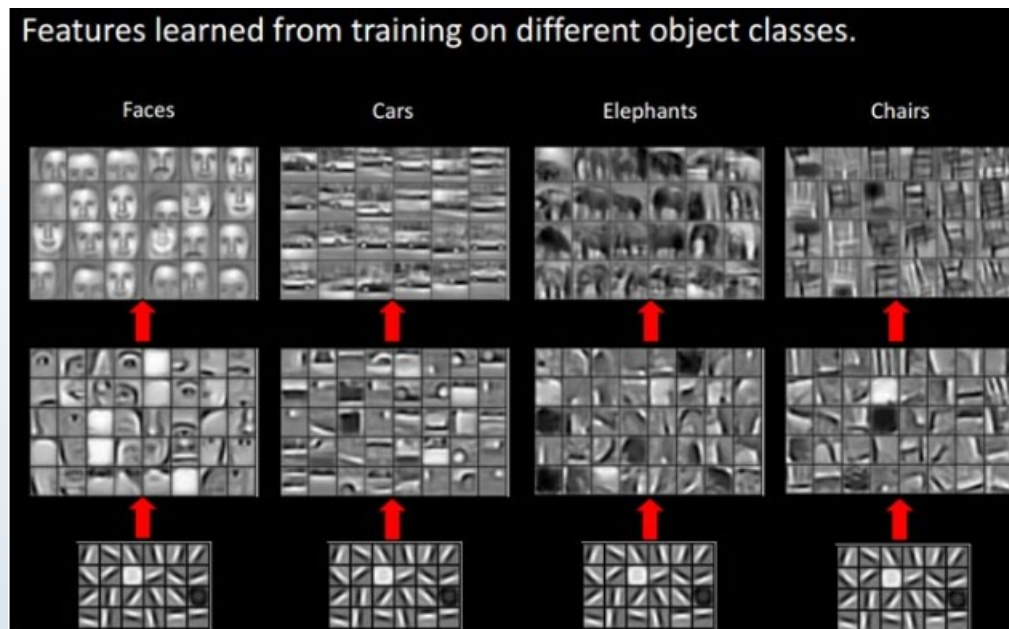
将原来的特征细分成多个小特征进行分析，比如第一个感知元只关注频率在( $f_0$ - $f_1$ )区间的语谱图，所以这部分的权重值会变大，其他频段的权重值会变小。

- 网络为什么有那么多层？（为什么需要多个感官才能判断食物）

因为一个具象的整体比一个抽象的小单元更容易理解。比如眼睛观察食物，鼻子嗅得气味，嘴巴品尝食物，最终整合于大脑做出判断。

因此，在神经网络中 第一层：升高维度，将原特征细分成多个基本单元。其他高层：从抽象对象中提取特征值。最初的一层输出的元素最基础，后面的层对这些基础单元通过组合重构，复用这些基础元素，层数越多整合的程度越高。

有个很生动的比喻，一个人在看一个doc的时候，眼睛看到的是word（basis），由这些word在大脑里自动切词形成term（object parts），在按照概念组织的方式，先验的学习，得到topic（models），然后再进行高层次的learning。



## 神经网络概述

### ● 网络如何学得好呢？

通过正向和反向传播两个过程，让神经网络模型从大量训练样本中学习统计规律，从而对未知事件做预测，并且模型输出值与期望值越来越拟合。

### • 正向传播：通过模型的每一层运行输入数据以进行

预测， `prediction = model(data) # forward pass``

### • 反向传播：使用模型的预测和已知的标签来计算误差（`loss`），然后通过网络反向传播，利用误差导数在梯度下降处得到最小误差的原理，来优化模型参数完成学习。

- ``loss = (prediction - labels).sum() loss.backward() # backward pass``

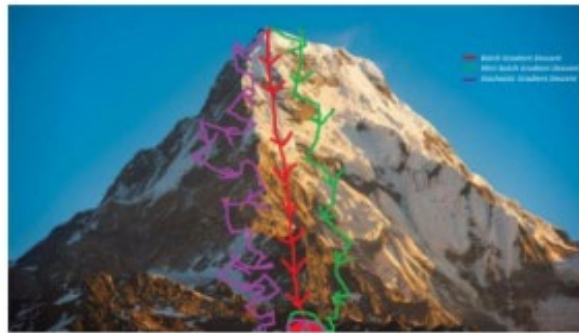
### • 具体操作：先定义Loss function,依据损失函数，衡量和实际结果之间差距。

反向找到一组参数使得损失函数最小。

别人眼中的深度学习



实际上的深度学习



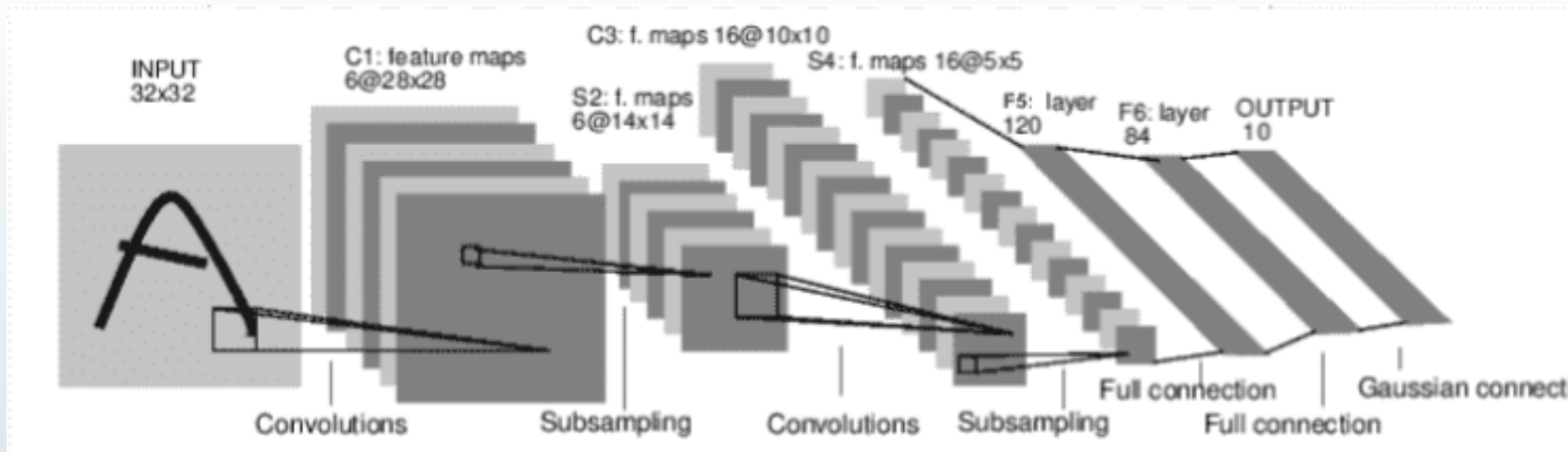
## 神经网络概述

### ● 什么是深度学习(Deep Learning)?

具有很多隐藏层的机器学习模型和海量的训练数据，因此可以学习更有用的特征，从而最终提升分类或预测的准确性。

### ● 基本流程

- 定义具有参数的神经网络模型
  - 遍历输入数据集
  - 通过网络处理输入
  - 计算损失（输出正确的距离有多远）
  - 将梯度传播回网络参数
  - 通常使用简单的更新规则来更新网络的权重： $\text{weight} = \text{weight} - \text{learning\_rate} * \text{gradient}$
- 常见的深度学习网络框架有：**CNN,RNN,TDNN,LSTM**





# 谢谢大家

---

Audio Signal Processing  
Audio-based Deep Learning Tutorials

b站：今天声学了吗

公众号：今天声学了吗

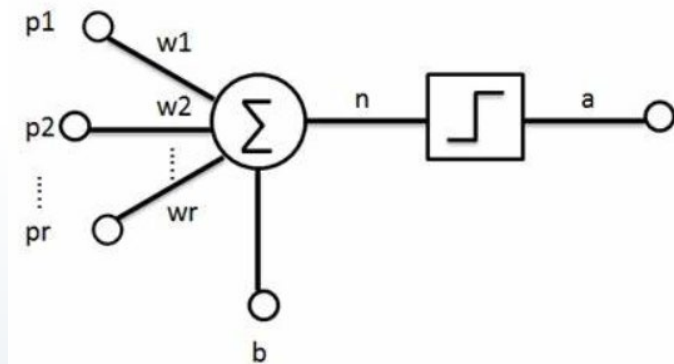
邮箱：1319560779@qq.com



## 感知器

- 感知器：包含参数 $w$ 和 $b$ 的激活函数，神经网络的最基本单元
- 操作流程：
  - 将输入信号都按一定比例放大或缩小。当信号进入时，它会乘以分配给该特定输入的权重值。
  - 累加所有信号，给总和添加偏移量。
  - 激活，通过将结果馈送到激活函数（也称为传递函数）得到输出。结构：
- 代码：

```
"""定义感知元函数(包含多个权值)
# 1.定义线性部分 $y = \text{math.fsum}(\text{input} * \text{weight}) + \text{bias}$ 
# 2.定义非线性激活部分 $y = 1 / (1 + \text{math.exp}(-y))$ 
# 3.给定输入与权重系数，得到输出
"""
```



## 浅层学习

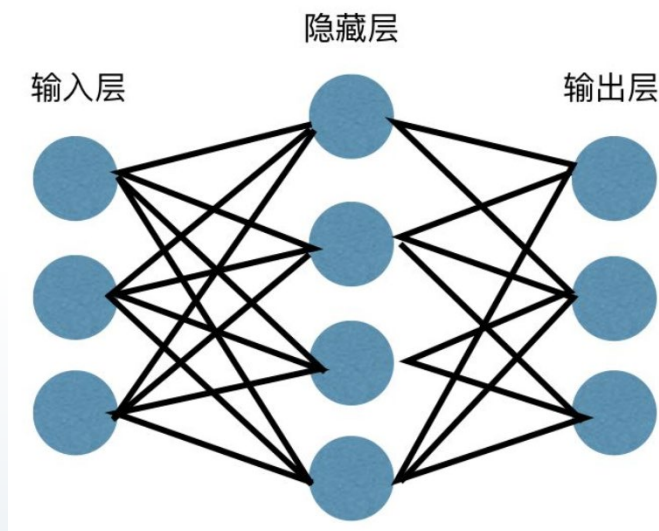
- 什么是浅层学习？

主要完成识别，分类等基本任务，其最主要的特点是模型简单，包含输入层，隐藏层和输出层。

- 代码：

```
"""定义浅层学习网络函数(含两个运算层)
# shallow_learning(input,weight1,weight2,bias1,bias2):
# 1.定义线性部分output1 = np.sum(input*weight)+bias
# 2.定义非线性激活部分output1 = 1 / (1 + np.exp(-output1))
# 3.给定输入与权重系数，得到输出
"""
```

- 常见模型：**AutoEncoder自动编码**，是一种尽可能复现输入信号的神经网络。为了实现这种复现，自动编码器就必须捕捉可以代表输入数据的最重要的因素，就像PCA那样，找到可以代表原信息的主要成分。

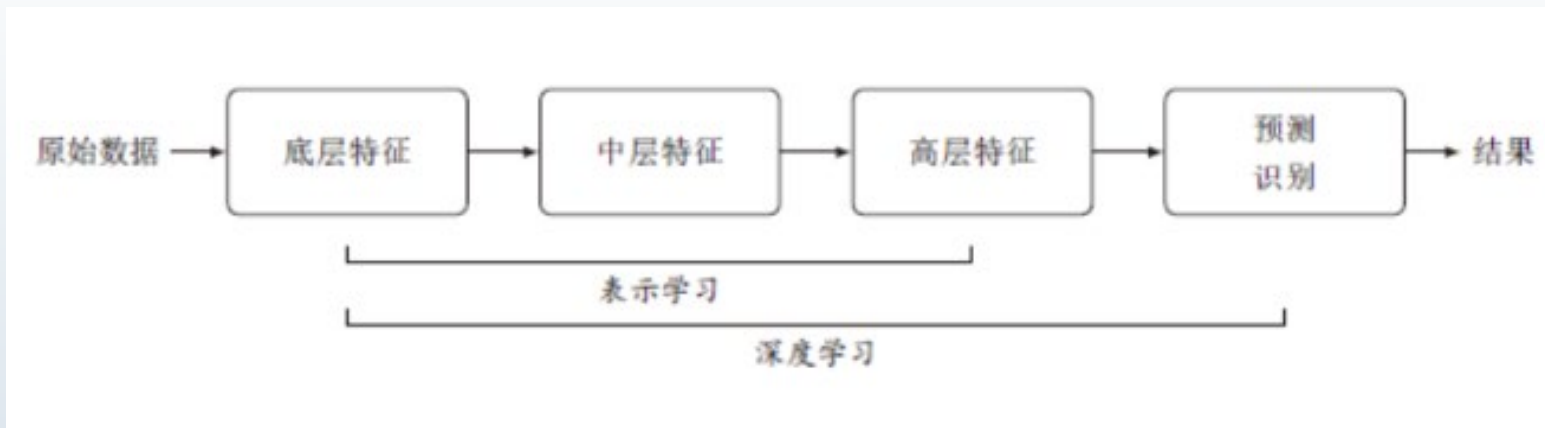


```
"""
1.第一部分的运算,有n个weight则有n个输出
2.第二部分运算要将weight取和
3.使用numpy数组表示
"""
```

## 深度学习

- 什么是深度学习？

用于特征提取的复杂任务中。其主要特点是通过构建具有很多隐层的机器学习模型和海量的训练数据，来学习更有用的特征，从而最终提升分类或预测的准确性。



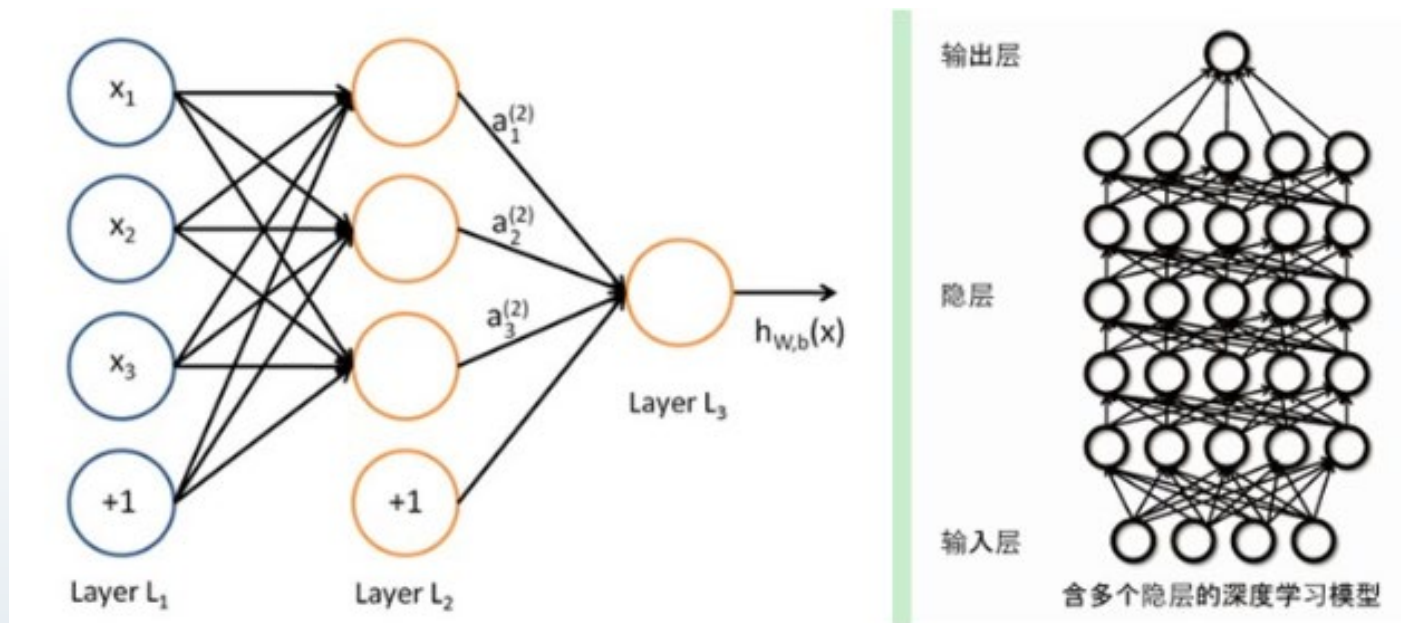
- 思想：堆叠多个层，也就是说这一层的输出作为下一层的输入。通过这种方式，就可以实现对输入信息进行分级表达了。



## 深度学习

### 与浅层学习比较

- 相同：都是分层结构，系统由包括输入层、隐层（多层）、输出层组成的多层网络，只有相邻层节点之间有连接，同一层以及跨层节点之间相互无连接，每一层可以看作是一个logistic regression模型；



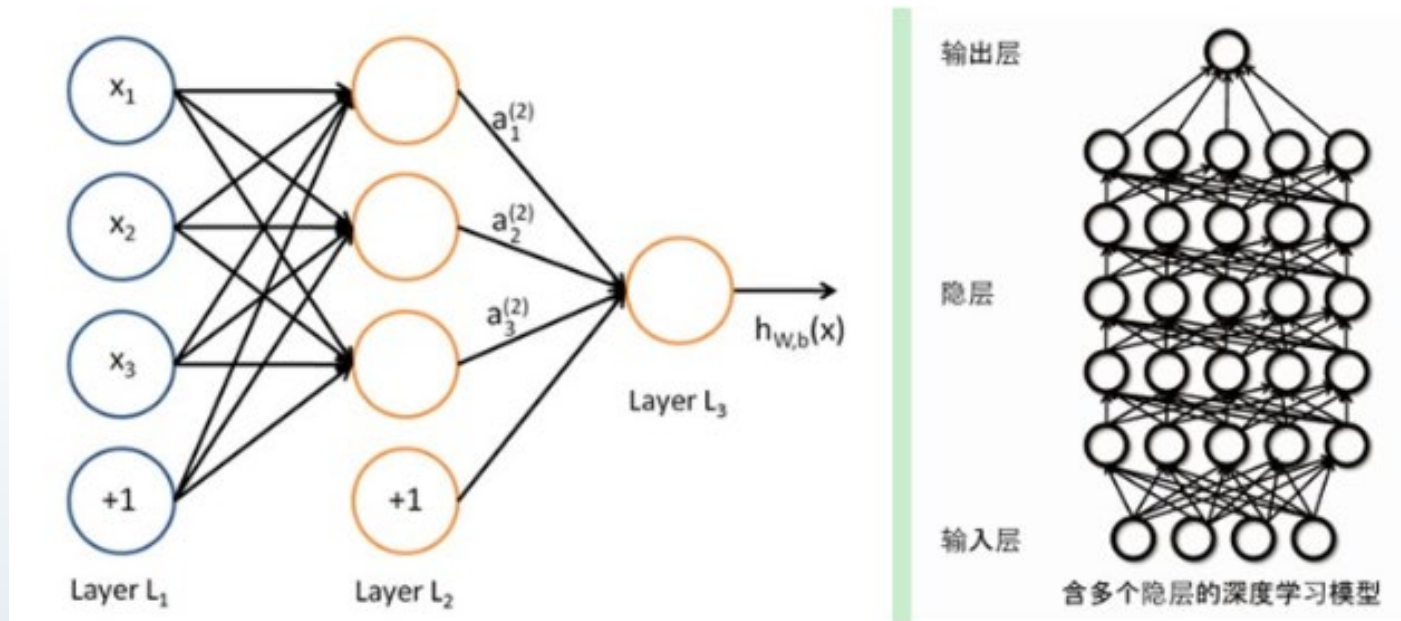
- 浅层结构算法多用于多数分类、回归等学习方法，其局限性在于有限样本和计算单元情况下对复杂函数的表示能力有限，针对复杂分类问题其泛化能力受到一定制约。
- 深度学习通过学习深层非线性网络结构，实现复杂函数逼近，表征输入数据分布式表示。同时展现了强大的从少数样本集中学习数据集本质特征的能力。多层的好处是可以用较少的参数表示复杂的函数。



## 深度学习

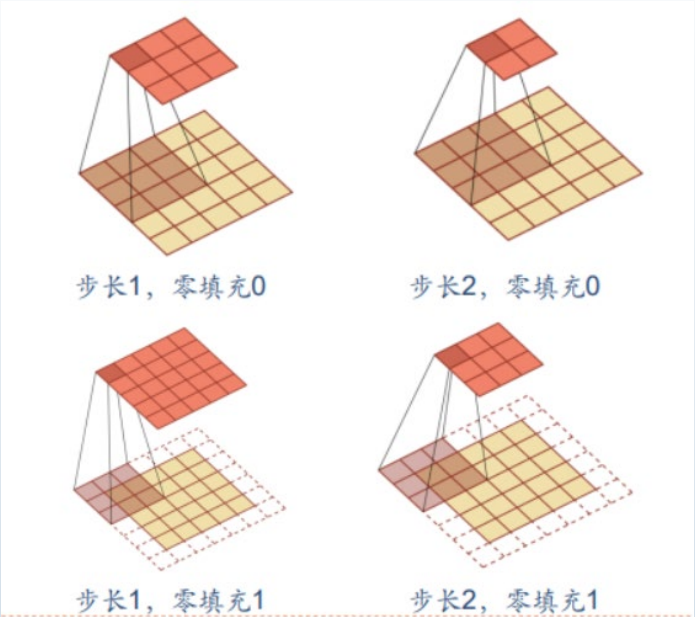
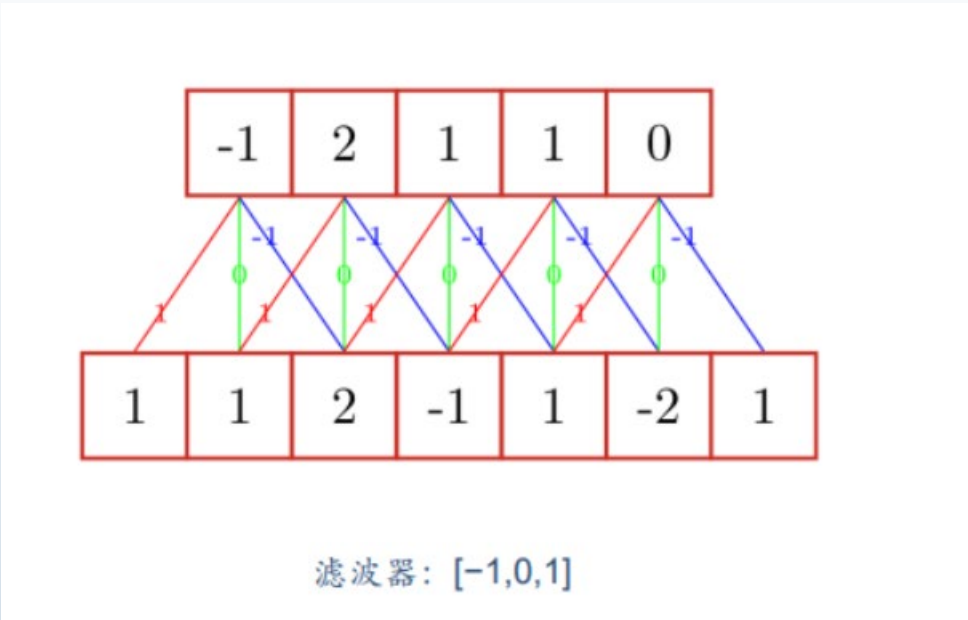
- 代码:

```
"""定义深度学习网络的类class(无继承型)
# 1.初始化网络模型
# 根据输入个数,输出个数以及中间层数,定义网络框架
layer
# layer_innum =
[input_num]+hidden_layer+[output_num]
# 随机生成weight值,weight个数与每层的输入个数相同
# current_weight =
np.random.rand(self.layer_innum[i],self.layer_innum[i
+1])
# 2.模型计算,每一部分实现线性运算np.dot()与非线性
激活output1=1/(1 + np.exp(-output1))
# 3.定义参数调用计算
"""
```



## 构造隐藏层之卷积层

- 概念：利用卷积核（kenel）实现数据的特征提取。
- 实际操作：按照一定权重处理输入对象并累加。与信号处理中的卷积有些不同，这里不存在时间反演，只是简单的相乘累加。如图为一维/二维卷积：

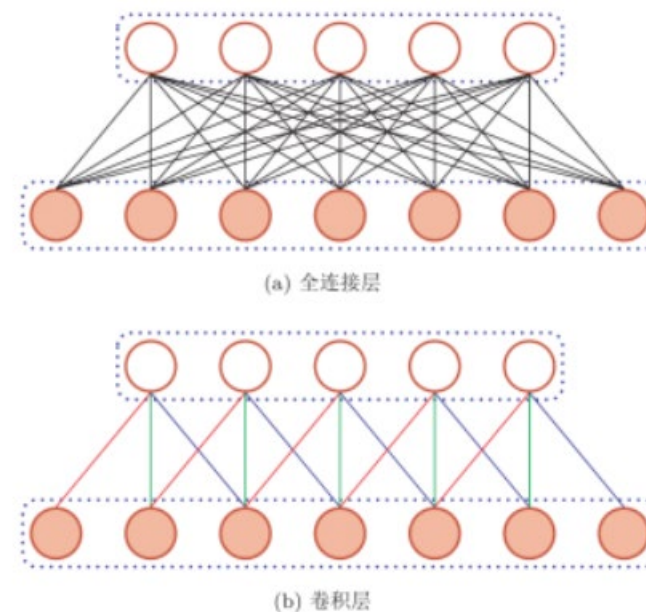
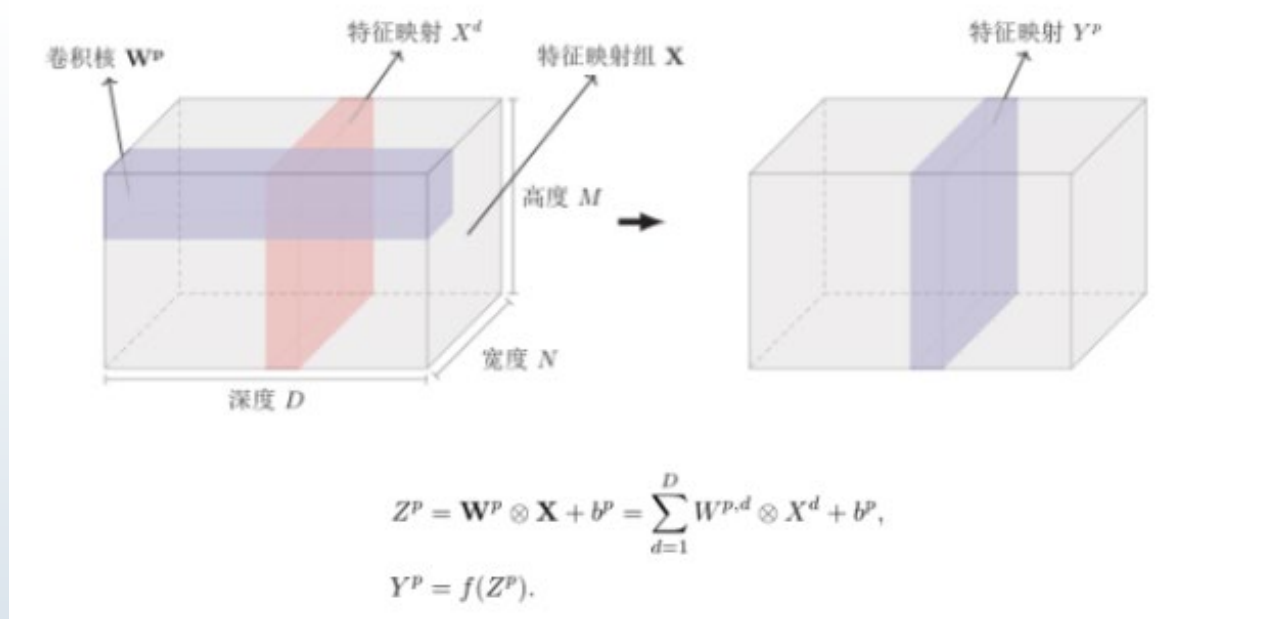


卷积核/kernel≈滤波器  
步长/stride ≈hop-size,  
填充值/padding ≈padding,  
扩张值/dilation

```
""" torch.nn.functional.conv2d的卷积计算
# 1. 设置环境import torch.nn.functional
# 2. 定义输入变量(5,5)，kernel值(2,2)，均为四维tensor,
(batch_size,channel,height,width)
# 3. 利用torch.nn.functional.conv2d得到输出
# 4. 得到输出
# 5. 更改padding值，打印输出大小
卷积后的长度 = (卷积前的长度 + 卷积核边长 + 2* padding)/步长 + 1
```

## 构造隐藏层之卷积层

- 三维卷积：输入对象多了一个维度（深度），卷积核也不是二维图像



- 卷积出现的意义：权值共享，用卷积层代替全连接层，参数数量减少，但神经元的个数没有变少，照样可以提取特征值！



## 构造隐藏层之卷积层

- 维度(padding)计算

- Input:  $(N, C_{in}, H_{in}, W_{in})$  or  $(C_{in}, H_{in}, W_{in})$
- Output:  $(N, C_{out}, H_{out}, W_{out})$  or  $(C_{out}, H_{out}, W_{out})$ , where

$$H_{out} = \left\lfloor \frac{H_{in} + 2 \times \text{padding}[0] - \text{dilation}[0] \times (\text{kernel\_size}[0] - 1) - 1}{\text{stride}[0]} + 1 \right\rfloor$$

$$W_{out} = \left\lfloor \frac{W_{in} + 2 \times \text{padding}[1] - \text{dilation}[1] \times (\text{kernel\_size}[1] - 1) - 1}{\text{stride}[1]} + 1 \right\rfloor$$

维度计算：卷积后的维度 = (卷积前的维度 - 卷积核边长 + 2\* padding)/步长 + 1

- 常见的卷积操作：
  - 转置卷积（先对原矩阵进行填充使其扩大到适配目标输出维度，进行普通的卷积操作的一个过程，其输入到输出的维度调返，但不是逆变换操作）
  - 微步卷积
  - 空洞卷积(很常用，增加数据集)
- 代码：

```
""" torch.nn.functional.conv2d的卷积计算
# 1.设置环境import torch.nn.functional
# 2.定义输入变量(5,5), kernel值(2,2), 均为四维
tensor,
(batch_size,channel,height,width)
# 3.利用torch.nn.functional.conv2d得到输出
# 4.得到输出
# 5.更改padding值, 打印输出大小
卷积后的长宽 = (卷积前的长宽 - 卷积核边长 + 2*
padding)/步长 + 1
"""
```

## 构造隐藏层之卷积层

- 卷积的结果：改变每个像素块的数值RGB值（视觉上改变图像颜色）

- 代码：

""" 卷积的网络模型类class对二维图片处理

# 0. 预设环境import torchvision, from torch import nn

# 1. 构建卷积模型，继承nn.Module，包含conv2d()

# 2. 加载图像数据，SummaryWriter写入到log文件中，  
图像PIL->tensor(transforms.ToTensor)

# from PIL import Image img = Image.open(img\_path)

# trans\_tensor = torchvision.transforms.ToTensor()

# 转换成四维trans\_tensor(img).unsqueeze(0)

# 3. 调用模型计算

# 4. 得到输出结果，SummaryWriter写入到log文件中  
tensor->PIL(transforms.ToPIL)

"""






1. 维度的变换unsqueeze&squeeze

网络模型的输入输出变量是四维

torch函数的计算变量是三维

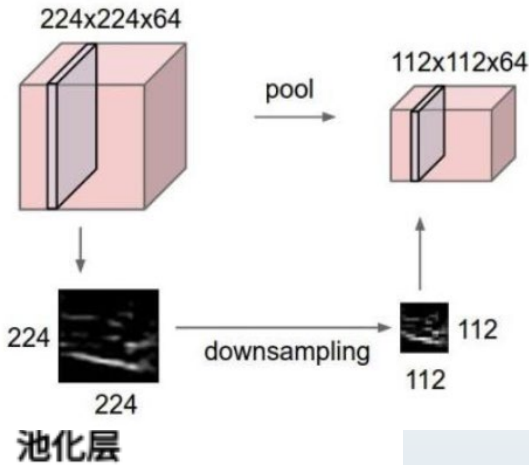
2. 数据类型的转换(transforms.ToTensor)(transforms.ToPILImage)

"""

卷积作用	卷积核	卷积后图像
输出原图	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
边缘检测（突出边缘差异）	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
边缘检测（突出中间值）	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
图像锐化	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
方块模糊	$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \times \frac{1}{9}$	

构造隐藏层之池化层

- 概念：也叫汇聚层，用于压缩数据和参数的量减小过拟合。也就是在不改变数据特征的情况下，将数据的尺寸变少。
- 具体作用：
  - 特征不变性，将图片缩小，但其特征的含量不变。
  - 特征降维，把这类冗余信息去除，把最重要的特征抽取出来。
  - 在一定程度上防止过拟合，更方便优化。
- 常见的池化层：最大池化，均值池化等。
- 卷积层和池化层的区别：



	卷积层	池化层
结构	零填充时输出维度不变，而通道数改变	通常特征维度会降低，通道数不变
稳定性	输入特征发生细微改变时，输出结果会改变	感受域内的细微变化不影响输出结果
作用	感受域内提取局部关联特征	感受域内提取泛化特征，降低维度
参数量	与卷积核尺寸、卷积核个数相关	不引入额外参数

```
"""池化的网络模型
# 0. 预设环境from torch import nn,from PIL import Image
# 1.构建池化的网络模型，继承nn.Module，包含MaxPool2d()
maxpooling中的stride = stride if (stride is not None) else kernel_size
# 2.加载图像数据，SummaryWriter写入到log文件中，图像PIL-
>tensor(transforms.ToTensor)
# from PIL import Image img = Image.open(img_path)
# trans_tensor = torchvision.transforms.ToTensor()
```

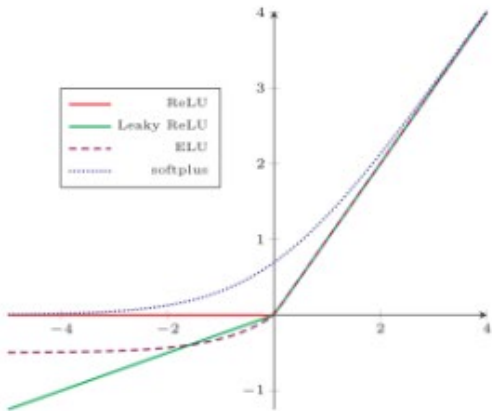


## 构造隐藏层之非线性层

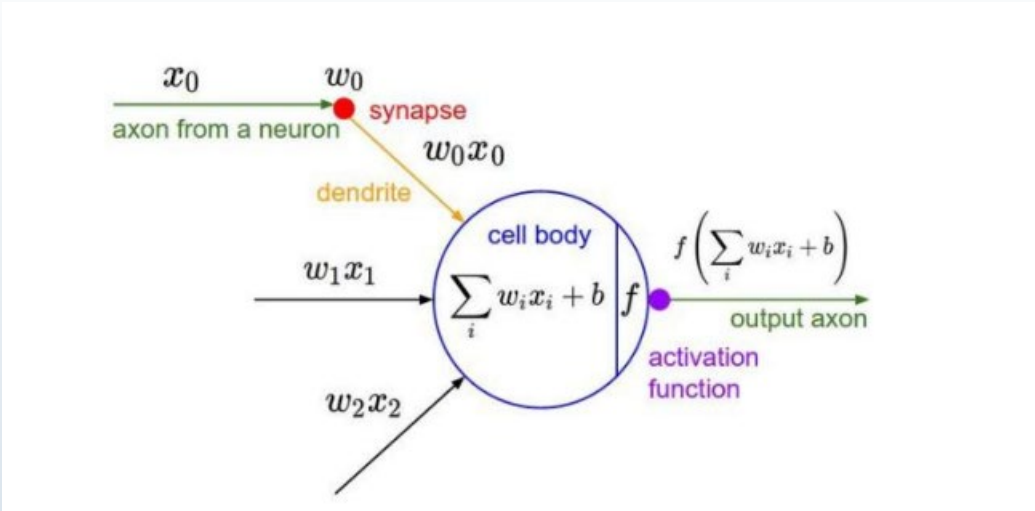
- 概念：因为只有若干线性层，其堆叠只能起到线性映射的作用，无法形成复杂的函数，因此引入非线性激活层。
- 常见的非线性激活函数

### 激活函数

有效减轻梯度消失问题！



激活函数	函数	导数
Logistic 函数	$f(x) = \frac{1}{1 + e^{-x}}$	$f'(x) = f(x)(1 - f(x))$
Tanh 函数	$f(x) = \frac{2}{1 + e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$
ReLU	$f(x) = \max(0, x)$	$f'(x) = I(x > 0)$
ELU	$f(x) = \max(0, x) + \min(0, \gamma(\exp(x) - 1))$	$f'(x) = I(x > 0) + I(x \leq 0) \cdot \gamma \exp(x)$
SoftPlus 函数	$f(x) = \ln(1 + e^x)$	$f'(x) = \frac{1}{1 + e^{-x}}$



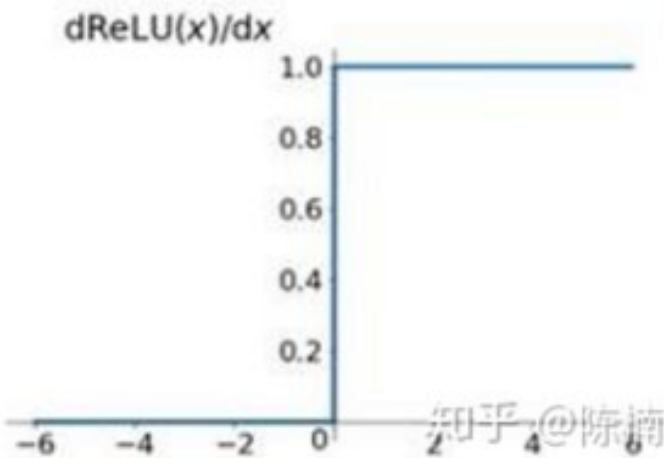
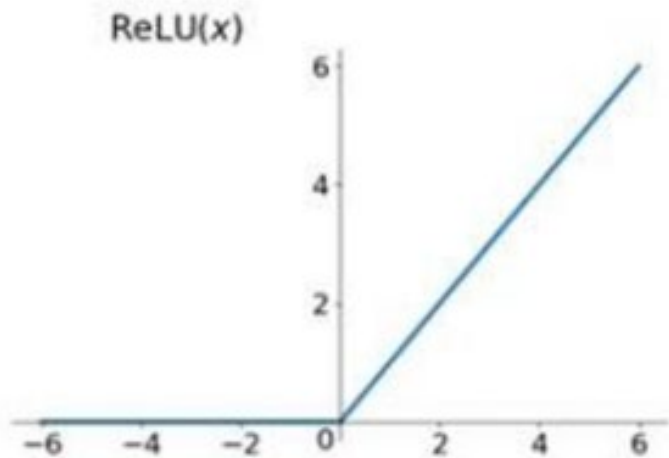
## 构造隐藏层之非线性层

### • ReLU

优点：解决梯度消失的问题（其导数是非0值），计算速度快（只有输出0，1）

缺点：没有输出负数，不是zero-centered，可能会出现Dead ReLU Problem，也就是一些神经元不会被激活到。因为计算速率太高导致训练过程中参数更新太快。

$$\text{ReLU} = \max(x, 0)$$



```
"""非线性激活ReLU的网络模型
# 0. 预设环境from torch import nn,from PIL import Image
# 1.构建池化的网络模型，继承nn.Module，包含ReLU()
# 2.加载图像数据，SummaryWriter写入到log文件中，图像PIL-
>tensor(transforms.ToTensor)
# from PIL import Image img = Image.open(img_path)
# trans_tensor = torchvision.transforms.ToTensor()
# 转换成四维trans_tensor(img).unsqueeze(0)
# 3.调用模型计算
# 4.得到输出结果，SummaryWriter写入到log文件中，tensor-
>PIL(transforms.ToPIL)
# tensorboard --
logdir=D:\WZY\Class\Pytorch_Audio\Chapter3\log_pooling --
port=1000
"""
```

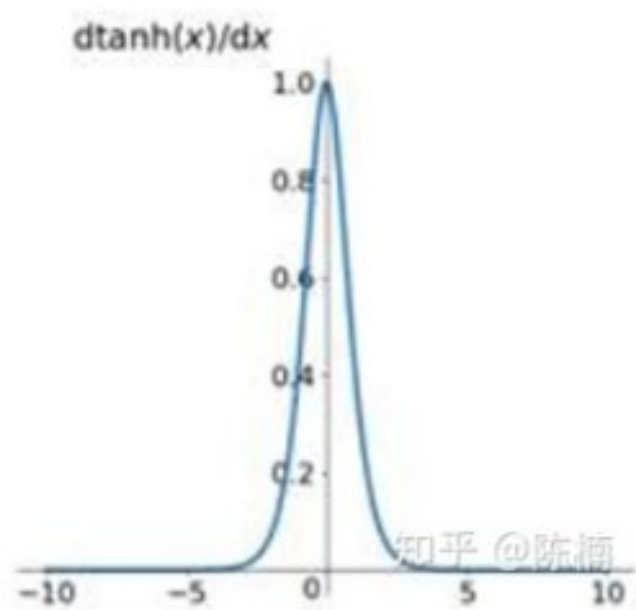
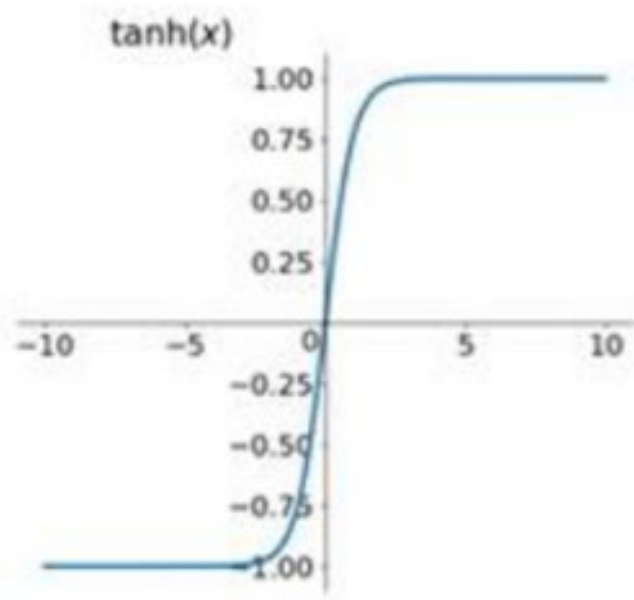
## 构造隐藏层之非线性层

- Tanh

优点：输出正负，解决了zero-centered的输出问题

缺点：梯度消失的问题和幂运算的问题仍然存在。

$$\tanh x = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$





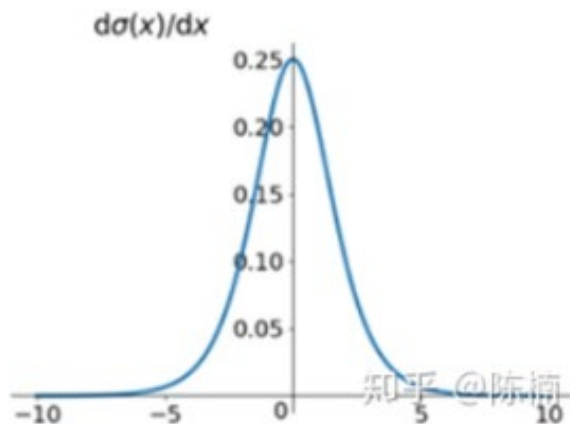
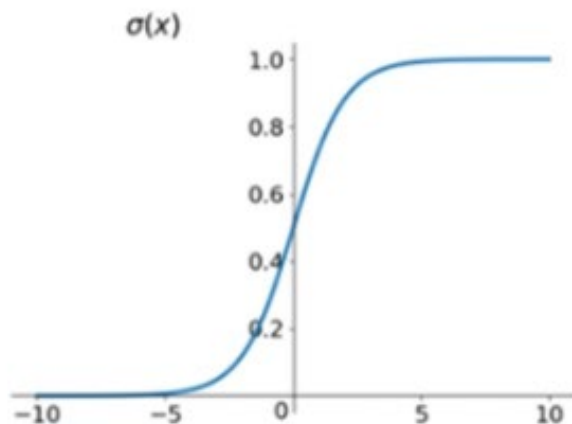
## 构造隐藏层之非线性层

### • Sigmoid

优点：适用于多分类问题，将输出值归一化

缺点：可能会产生梯度消失,并且权重更新过程不是一个收敛的过程，Sigmoid函数的输出值恒大于0，这会导致模型训练的收敛速度变慢。（为什么会梯度消失，后面详细介绍）。

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



"""非线性激活Sigmoid的网络模型

# 0.预设环境from torch import nn,from PIL import Image

# 1.构建池化的网络模型，继承nn.Module，包含Sigmoid()

# 2.加载图像数据，SummaryWriter写入到log文件中，图像PIL->tensor(transforms.ToTensor)

# from PIL import Image img = Image.open(img\_path)

# trans\_tensor = torchvision.transforms.ToTensor()

# 转换成四维trans\_tensor(img).unsqueeze(0)

# 3.调用模型计算

# 4.得到输出结果，SummaryWriter写入到log文件中，tensor->PIL(transforms.ToPIL)

# tensorboard --logdir=D:\WZY\Class\Pytorch\_Audio\Chapter3\log\_Sigmoid --port=1000

"""

## 构造隐藏层之线性层

- 概念：线性拟合  $y = xA^T + b$ （二维）将输出的最后维度变小，通过线性拟合，变成一维啦
- 代码：

"""线性层的网络模型

# 0.预设环境from torch import nn,from PIL import Image

# 1.构建池化的网络模型，继承nn.Module，包含Linear(in\_features=,)

# 2.加载图像数据，SummaryWriter写入到log文件中，图像PIL->tensor(transforms.ToTensor)

# from PIL import Image img = Image.open(img\_path)

# trans\_tensor = torchvision.transforms.ToTensor()

# 转换成四维trans\_tensor(img).unsqueeze(0)

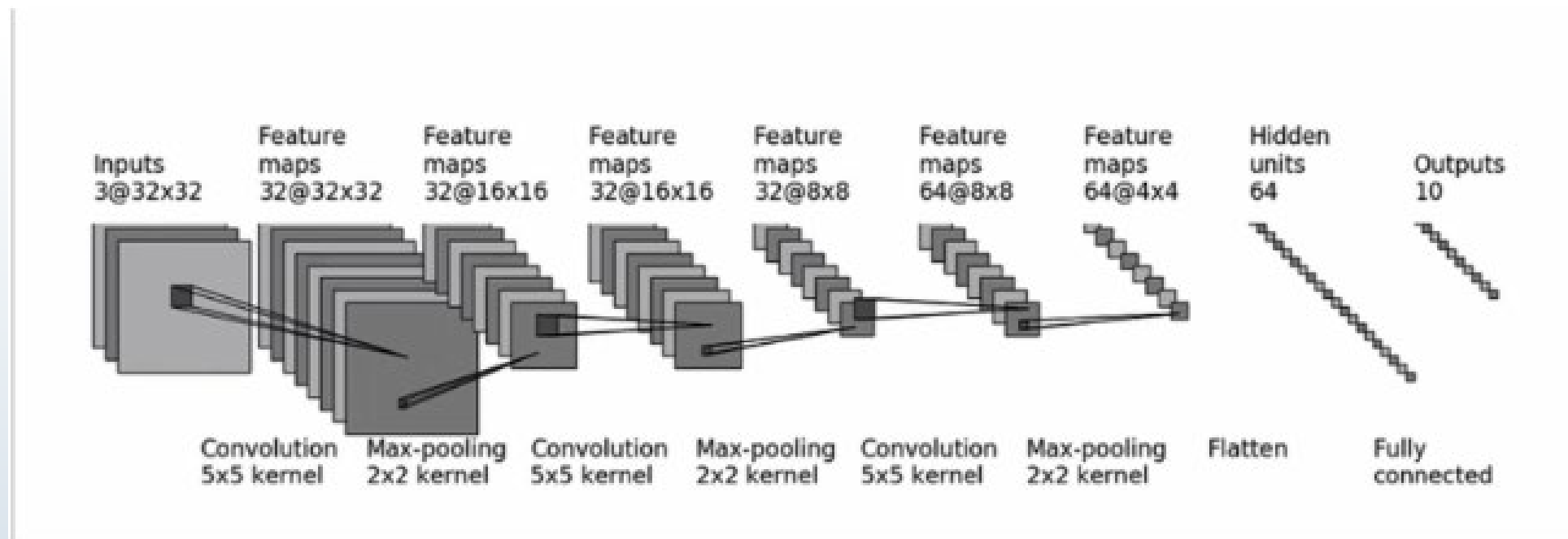
# 3.调用模型计算

# 4.得到输出结果，SummaryWriter写入到log文件中，tensor->PIL(transforms.ToPIL)

# tensorboard --logdir=D:\WZY\Class\Pytorch\_Audio\Chapter3\log\_Linear --port=1000

"""

## 构造隐藏层nn.Module



"""看图构造网络模型

# 1. 构造网络的类, 继承nn.Module

# 2. 该类中包含conv2d,maxpooling,flatten层

# 3. 参数的传递, padding值的计算  $d_{out} = d_{int} - kernel + 2 * padding / stride + 1$

# 4. 调用该类, 并用ones测试

# 5. 写入文件

"""

"""注意事项

1.  $padding1 = (32-1+5-32)/2 = 2$

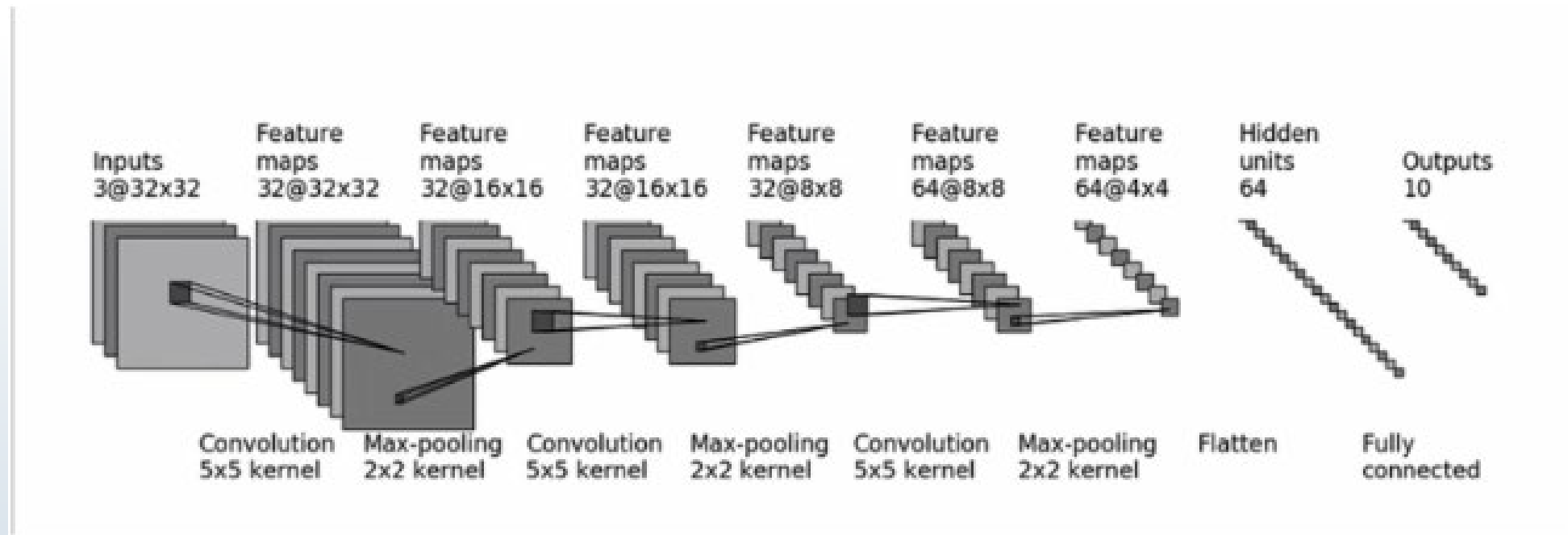
2.  $padding2 = (16-1+5-16)/2 = 2$

3.  $padding2 = (8-1+5-8)/2 = 2$

"""



## 构造隐藏层nn.Sequential



"""看图构造网络模型

# 1.构造网络的类,继承nn.Sequential

# 2.该类中包含conv2d,maxpooling,flatten层

# 3.参数的传递, padding值的计算  $d_{out} = d_{int} - kernel + 2 * padding / stride + 1$

# 4.调用该类,并用ones测试

# 5.写入文件

"""

"""注意事项

1.  $padding1 = (32-1+5-32)/2 = 2$

2.  $padding2 = (16-1+5-16)/2 = 2$

3.  $padding2 = (8-1+5-8)/2 = 2$

"""

## 神经网络之归一化输出

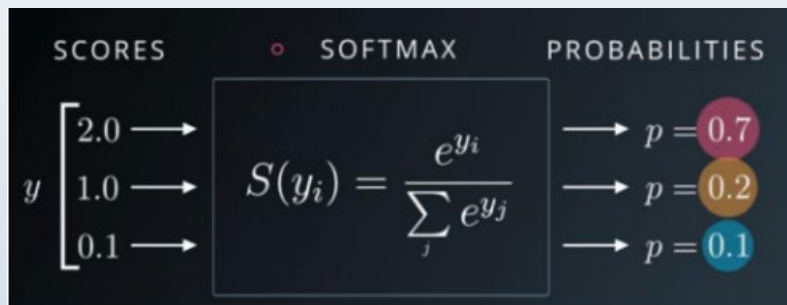
- 介绍：归一化输出使得最终的输出结果之和为1（转化成概率形式）
- 常见方法：
- 二分类——Sigmoid

输出结果只有一个，0或者1。概率表示为：

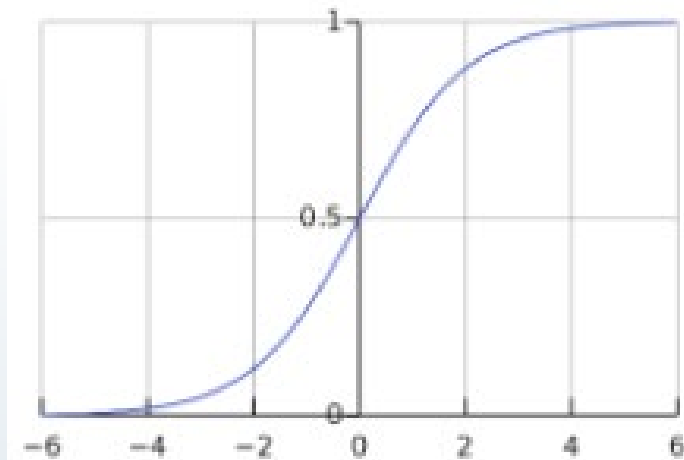
$$p(y = 1|\mathbf{x}) = \sigma(\mathbf{w}^T \mathbf{x}) \triangleq \frac{1}{1 + \exp(-\mathbf{w}^T \mathbf{x})}$$

- 多分类——Softmax

输出结果有多个，输出之和为1。解决多个分类的问题。



- 为什么选择指数函数（幂函数）？为了使输出的值是大于0，选择以e为底的激活函数进行输出，保证输入是全域，输出是>0



# 谢谢大家

---

## Audio Signal Processing Audio-based Deep Learning Tutorials

b站：今天声学了吗

公众号：今天声学了吗

邮箱：1319560779@qq.com