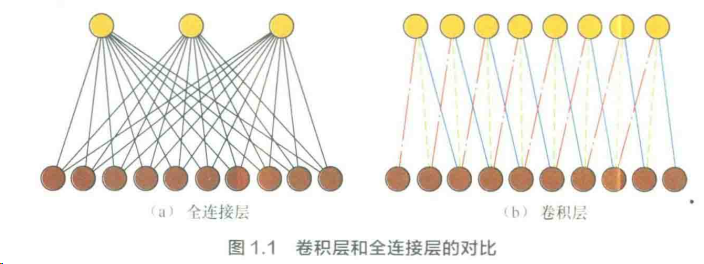
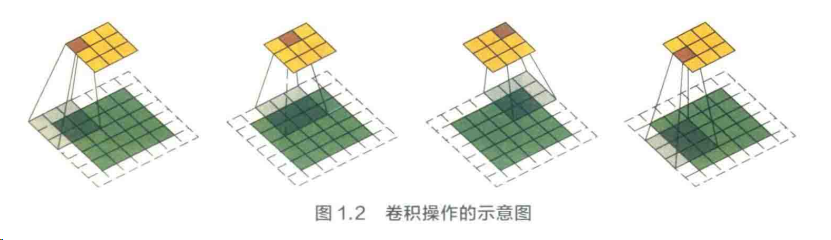
深度学习

* 简述卷积的基本操作，并分析其与全连接层的区别：

区别：卷积神经网络主要由卷积层构成，它具有局部连接和权值共享的特性。

基本操作：通过特定数目的卷积核对输入的多通道特征图进行扫描和运算，从而得到多个拥有更高层语义信息的输出特征图（通道数等于卷积核个数）。



局部连接：输出层上的每个节点都只与输入层的部分节点连接；

权值共享：卷积核的滑动窗机制使得输出层上不同位置的节点与输入层的连接权值都是一样的；

输入/输出数据的结构化：卷积操作能够再输出数据中保持输入数据的结构信息，例如二维图像中不同节点仍然保持着与原始图像基本一致的空间对应关系；而如果将二维图像输入进全连接层，其输出数据会被展平成一维数组，从而丧失输入数据和输出数据在结构上的对应关系；

* 如何计算各层的感受野：



F为卷积核大小，s为步长，l为感受野。

* 卷积层的输出尺寸、参数量和计算量：

Feature map大小计算公式：N=(W-F+2P)/S+1（如果S>1时，可能出现非整数情况，许多深度学习框架采取向下取整的方法，放弃输入特征图的一部分边界数据，一般向下取整），F为卷积核大小，P为padding大小，S为步长；

卷积层参数量：params=channels\_out\*(kernel\*kernel\*channels\_in)+bias，其中bias=channels\_out;

池化层参数量：池化主要用于特征降维，压缩数据和参数的数量，减小过拟合，同时提高模型的容错性。池化不需要参数控制；

全连接层参数计算：可以看作kernerl size为特征图大小的卷积层；

卷积层的计算量：channel\_in\*channel\_out\*kernel\*kernel\*N\_w\*N\_h

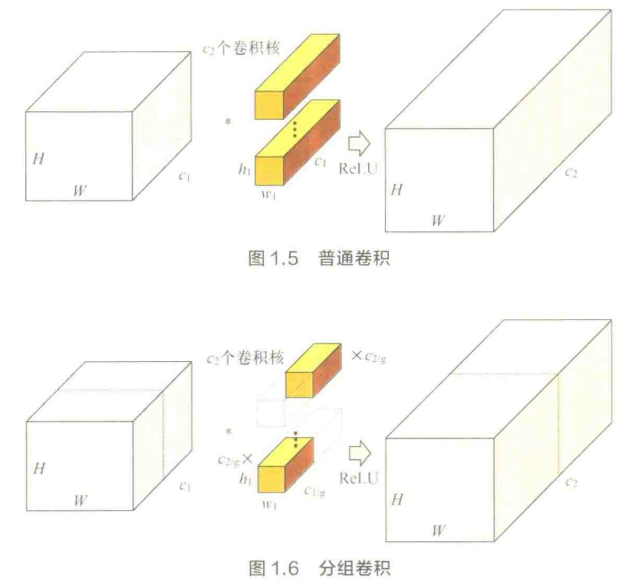
减小网络参数应主要针对全连接层，进行计算量优化时，重点应放在卷积层；

* 分组卷积

普通卷积的卷积核通道数等于输入特征图的通道数，因此最终输出特征图的每个通道都与输入特征图的所有通道相连接，因此最终输出特征图的每个通道都与输入特征图的所有通道相连接，在通道维度上是全连接的。

分组卷积就是将输入通道和输出通道都划分为相同的组数，然后仅让处于相同组号的输入通道和输出通道进行全连接。如果g为输入/输出通道所分的组数，则分组卷积能够将卷积操作的参数量和计算量都降低为普通卷积的1/g；

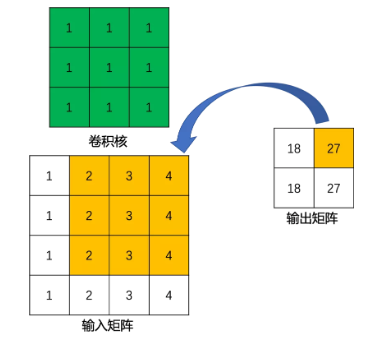
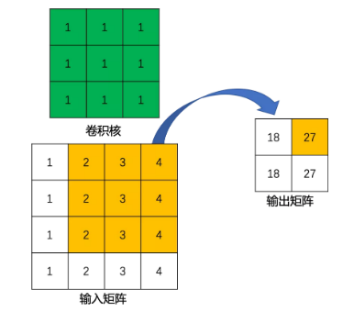
因此分组卷积主要被用来构建用于移动设备的小型网络模型；



* 转置卷积（反卷积）

标准卷积的运算操作 其实是对卷积核中的元素 与输入矩阵上对应位置的元素 进行逐像素的乘积并求和。然后，卷积核在输入矩阵上以步长为单位进行滑动，直到遍历完输入矩阵的所有位置。

对转置卷积而言，我们实际上是想建立一个逆向操作，即 **一对多的映射关系**。



标准卷积 转置卷积

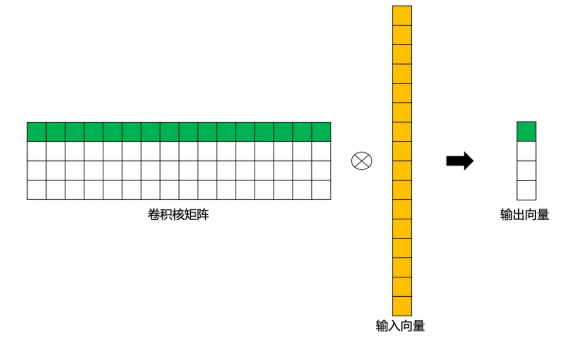
推导过程：

定义一个**4×4 输入矩阵 input**：

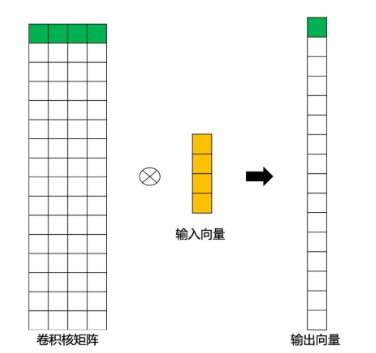
再定义一个 **3×3 标准卷积核 kernel**：

**将输入矩阵 input 和输出矩阵 output 展开成 16×1 列向量 X 和 4×1 列向量 Y，可分别表示为：**

经推到，可得到4×16的稀疏矩阵C：



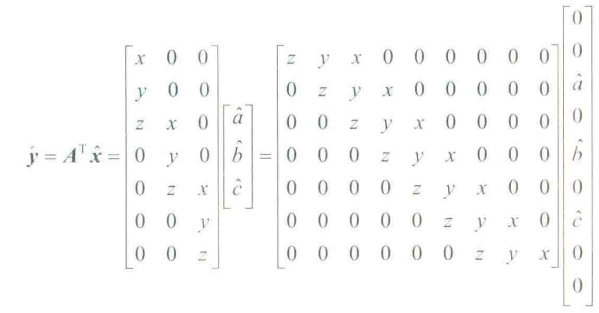
而转置卷积其实就是要对这个过程进行逆运算，即 **通过 C 和 Y 得到 X**：



转置卷积能将普通卷积中输入到输出的尺寸变换回来，但这只是回复了形状，并不能复原具体的取值。

转置卷积的信息正向传播与普通卷积的误差反向传播所用到的矩阵相同。

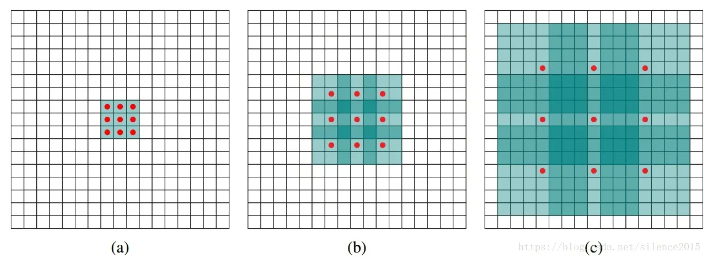
因此转置卷积本质上是对一个输入数据进行适当的变换（补零/上采样）的普通卷积操作。



普通卷积主要用来做特征提取，倾向于压缩特征图尺寸；转置卷积用于对特征图进行扩张或者上采样：语义分割/实例分割，物体检测（热力图），AE，VAE，GAN

* 空洞卷积

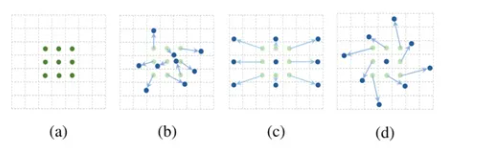
在语义分割任务中，一般采用池化操作来扩大特征图的感受野，但这同时会降低特征图的分辨率，丢失一些信息，导致后续的上采样无法还原一些细节，从而限制最终分割精度。



空洞卷积就是在卷积核中引入扩张率这个超参数，扩张率为r的空洞卷积，卷积核上相邻的数据点之间有r-1个空洞，如上图中扩张率分别为1，2，4；

* 可变性卷积要解决的问题

普通卷积操作是在固定的，规则的网格点上进行数据采样，这束缚了网络的感受野形状，限制了网络对几何形变的适应能力。为了克服这个限制，可变性卷积在卷积核的每个采样点上添加一个可学习的偏移量，让采样点不再局限于规则的网格点。



（c）和（d）是可变性卷积的两个特例，一个体现了可变性卷积的尺度变换特性，后者则是对卷积核进行旋转。

实际上，空洞卷积可以看作一种特殊的可变性卷积。

* ResNet

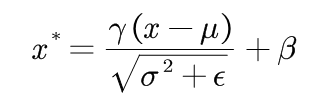
ResNet的提出源于这一种现象：随着网络层数的加深，网络的训练误差和测试误差都会上升，这种现象称为网络的退化。它采用跳跃连接，缩短误差反向传播到各层的路径，有效抑制梯度消失的现象；由于有近道的存在，若在网络层数加深时性能退化，则它可以通过控制近道退化到浅层时的连接状态。

* 批归一化

在机器学习中一般会假设模型的输入数据的分布时稳定的。如果模型输入数据分布发生变换，则成为协变量偏移。由于深度模型前一层的输出与网络参数的取值有很大关系，在训练过程中，如果之前层的参数被更新后，该层的输入数据的分布必然也跟着变化。网络越深这种现象越明显。

这会影响学习效率，并使得学习过程变得不稳定；网络前几层的参数更新很可能使后几层的输入数据变得过大或过小，从而掉进激活函数的饱和区，导致学习过程过早停止；为了减小这种影响会采用较小的学习率，降低收敛速度；

批归一化的作用就是确保网络中的各层，即使参数发生了变化，输入/输出的数据分布也不能产生较大的变化；采用批归一化后，模型的训练过程更加稳定，对初始值不再那么敏感，可以采用更大的学习率来加速收敛。



在批归一化过程中放置了两个可学习参数，他们有如下作用：

保留网络各层在训练过程中的学习成果，如果没有这两个参数，批归一化会退化为普通的标准化，使得输出的分布几乎不变，不能有效地进行学习；

保证激活单元的非线性表达能力，如果采用Sigmoid和Tanh等函数，则经过标准化的数据会落在激活函数的近似线性区域，没利用上他们的非线性区域；

批归一化具有自我关闭能力，可以回复到原始的输入值，当批归一化导致特征分布被破环，网络泛化性能减弱时，可以通过这两个参数将其关闭；

批归一化放置的位置：

在激活层之前，可以有效避免批归一化破坏非线性特征的分布，可以使数据点尽量不落入激活函数的饱和区域，缓解梯度消失问题；

由于ReLU没有那些问题，因此也可以把BN放在激活层之后，避免在激活层之前被转换成相似的模式从而使得非线性特征分布趋于同化；

* 用于分类的CNN最后几层一般是什么层？

网络末端一般是几层全连接层，这是因为卷积层具有局部连接，权值共享的特性，其在不同位置是采用相同的卷积核进行特征提取的，也就是说卷积层的特征是局部的且位置不敏感的。因此，全卷积网络也许能知道图片中不同位置有哪些元素，但无法提取这些元素之间的关联关系。而分类任务要考虑它们之间的关联关系，需要一个全局的、位置敏感的特征提取器。

近几年，分类网络在卷积层之后，最后一层之前通常采用全局平局池化，它与全连接层有着相似的效果（可以提取全局信息），并且减小参数量和计算量，具有较好的可解释新，可以知道特征图上哪些点对最后的分类贡献大。**对整个网路在结构上做正则化防止过拟合。**其直接剔除了全连接层中黑箱的特征，直接赋予了每个channel实际的内别意义。

* 瓶颈结构

瓶颈结构的初衷是为了降低大卷积层的计算量，即在计算比较大的卷积层之前，先用1×1的卷积核来压缩大卷积层输出特征的通道数，再采用一个1×1的卷积核提升通道数（可选）。瓶颈结构可以用更小的计算代价达到与之前相似甚至更好的效果。

* 沙漏结构：

一般包含两个分支：自底向上（bottom-up）分支，利用卷积池化压缩特征图尺寸，类似encoder；自顶向下（top-down）分支，利用反卷积或插值将特征图的尺寸增大，类似decoder。沙漏结构一般是将多尺度信息进行融合，同时，沙漏结构单元中的多个卷积层可以提升感受野，增强模型对小尺寸但又依赖上下文的物体的感知能力。

* Dropout为什么可以缓解过拟合

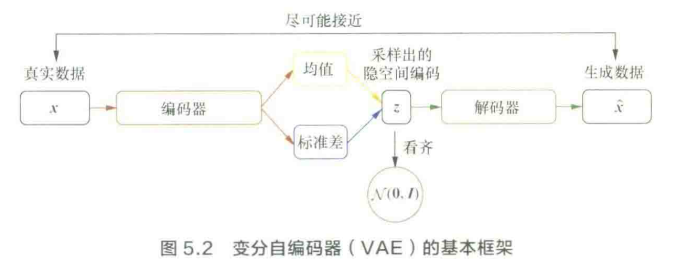
Dropout是指在网络训练阶段，每次迭代时会从基础网络中随机丢弃一定比例的神经元。一方面，Dropout可以看作是集成了大量神经网络的Bagging方法（相同数据训练若干个不同的模型，最终预测结果是这些模型投票得到的）。另一方面，Dropout可以减少神经元之间复杂的共适应关系，网络中的每个神经元不会对另一个特定的神经元的激活非常敏感。

* AE,VAE,GAN的联系与区别：

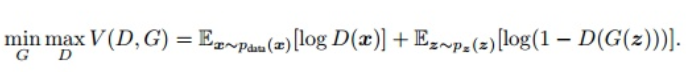
AE，VAE，GAN同属于可微生成网络，他们常常用神经网络来表示一个可谓函数G，用这个函数来刻画隐变量z到样本分布的映射关系。

AE：由encoder和decoder组成，首先将真实数据x压缩为低维隐空间中的一个向量z，然后decoder将z解压，得到生成样本。最终目的是期望由真是样本x得到的隐向量能够尽可能地抓住输入的精髓，使重建的数据尽可能接近真实样本。可应用于数据去噪，可视化降维以及数据生成等方向。

VAE：的主要优势在于能够产生新的隐向量z，进而生成有效的新样本。VAE能够生成新样本的原因是，VAE在编码过程中加入了一些限制，迫使编码器产生的隐向量的后验分布q(z|x)尽量接近某个特定的分布。VAE的编码器输出的不再是隐向量，而是所属正态分布的均值和标准差，然后再根据均值和标准差来采样出隐向量z。由于采样操作存在随机性，因此生成的图像不再是唯一的。



GAN：解决的是生成模型的一大难点，如果度量生成分布与真实分布的相似度。GAN把度量任务交给一个神经网络来做，这个网络称为判别器。



AE与VAE的联系与区别：都属于有向图模型，模型的目的都是对隐变量空间进行建模，但AE只会模仿不会创造。AE的优化目标是最小化真实样本与生成样本的重构误差，VAE除了考虑重构误差，还加入了对隐变量空间的约束目标。

GAN与AE/VAE的区别：VAE的重构损失目的和迫使隐变量后验分布接近正态分布的这两种损失相互对立，这与GAN一样，他们内部都存在对抗思想，只不过VAE是将两部分同步优化，而GAN是交替优化的。与AE相同，GAN的优化目标只涉及到生成样本与真实样本之间的比较，没有VAE中对后验分布的约束，但是GAN利用对抗训练绕过了对分布间距离的度量，且判断样本真假时不需要真实样本与生成样本一一对应。GAN没有像AE那样从学习后验分布q(z|x)中获得生成样本的能力，所以存在模式坍塌，训练不稳定的问题。此外，AE/VAE可以通过对数据的操控实现数据在语义层次上的操控。

* 物体检测中单步模型和两步模型的性能差异：

单步模型是指没有独立的，显示地提取候选区域（region proposal），直接由输入图像得到其中存在的物体的类别和位置信息的模型，典型的单步模型有SSD，YOLO。

两步模型先在输入图像上筛选出一些可能存在物体的候选区域，然后针对每个候选区域判断是否存在物体，如果存在，就给出物体的类别和位置修正信息。典型的两步模型有R-CNN,Fast R-CNN,Faster R-CNN, Mask R-CNN。

单步模型在计算效率上有优势，两步模型在检测精度上有优势。多数单步模型是利用预设的Anchor Box来捕捉存在于图像中各个位置的物体，因此单步模型会对数量庞大的Anchor Box进行分类，由于一幅图像中实际含有的物体数目远小于Anchor Box的数量，因此在训练这个分类器的时候正负样本是极不均衡的，导致分类器训练效果不佳。

两步模型在候选区域提取的过程中会对候选框位置和大小进行修正，因此在进入第二部之前，候选区域的特征已被对齐，这样有利于第二步的分类提供质量更高的特征。另外两步模型在第二步中会再次修正候选框，这带来了更高的定位精度，但增加了模型复杂度。但不模型没有候选框提取过程，Anchor Box的预测基于该层上每个特征点的感受野，输入特征未被对齐，质量较差，因而精度较低。

两步模型在第二步对候选区域进行分类和位置回归时，是针对每个候选区域独立进行的，因此该部分的算法复杂度线性正比于预设的候选区域数目。

* 有哪些措施可以增强模型对小物体的检测效果：

在模型设计方面，可以采用特征金字塔，沙漏结构等特征子结构，增强网络对小尺度特征的感知和处理能力，尽可能提高网络的感受野，使得网络能够利用更多的上下文信息，同时减少网络总的下采样比，使得最后用于检测的特征分辨率更高；

在训练方面，可以提高小物体样本在总体样本中的比例，或者利用数据增强手段，将图像缩小以生成小物体样本；

在计算允许的范围内，可以尝试使用更大的输入图像尺寸；

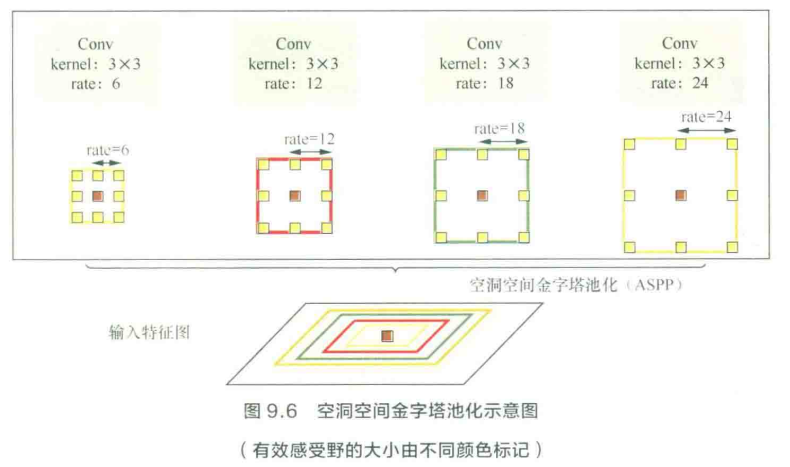
* 编码器解码器结构的设计

图像分割中的编码器可视为特征提取网络，通常使用池化层来逐渐缩减输入数据的空间维度；而解码器则通过上采样/反卷积等网络层来逐步恢复目标的细节和相应的空间维度。由于引入池化层可以增加后续卷积层的感受野，并能使特征提取聚焦在重要信息中，降低背景干扰。然而池化操作使位置信息大量流失，不足以对像素进行精确的分割，因此提出建立快捷连接，使高分辨率的特征信息参与到后续的解码环节。

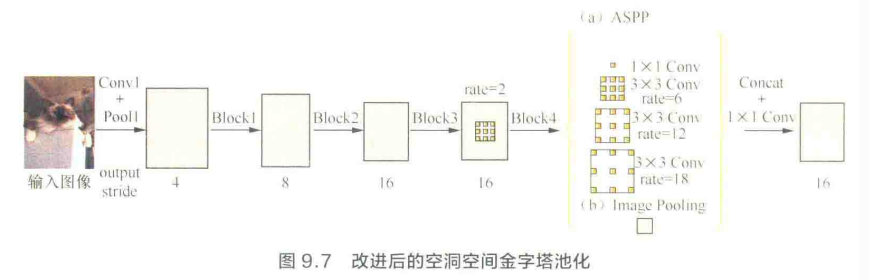
* DeepLab

DeepLab v1的创新分别是空洞卷积和全连接条件随机场。空洞卷积是为了解决编码过程中信号不断被下采样，细节丢失的问题。由于卷积层提取的特征具有平移不变性，这就限制了定位精度，所以引入了全连接条件随机场来提高模型捕获局部结构信息的能力。

DeepLab v2提出了空洞空间金字塔池化（ASPP），并将VGG网络替换成了更深的ResNet网络。ASPP可用于解决不同检测目标大小差异的问题：通过在给定的特征层上使用不同扩张率的空洞卷积，ASPP可以有效地进行重采样，最后将ASPP各个空洞卷积分支采样后地结果融合到一起。

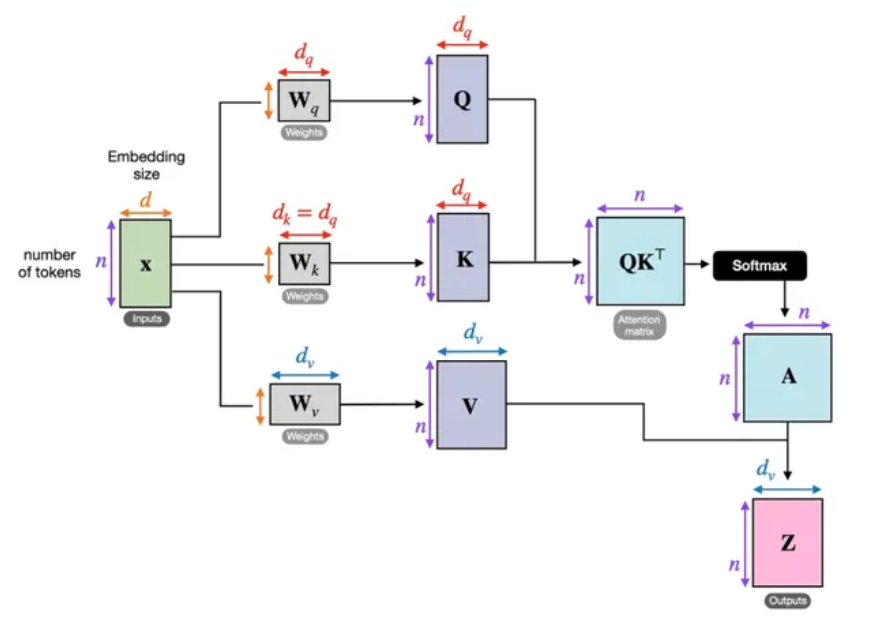


DeepLab v3：加入了批归一化（BN）层；其次将ASPP中尺寸为3×3，空洞大小为24地卷积替换成一个普通的1×1卷积，以保留滤波器中间部分的有效权重。因为，随着空洞卷积扩张率的增大，滤波器中有效权重的个数在减小，为了克服长距离下有效权重减小的问题，在ASPP最后增加了全局平均池化以便更好地捕捉全图信息。



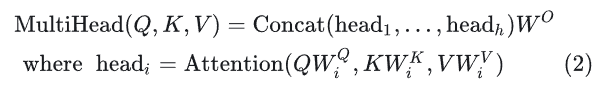
Transformer

* Self-Attention机制



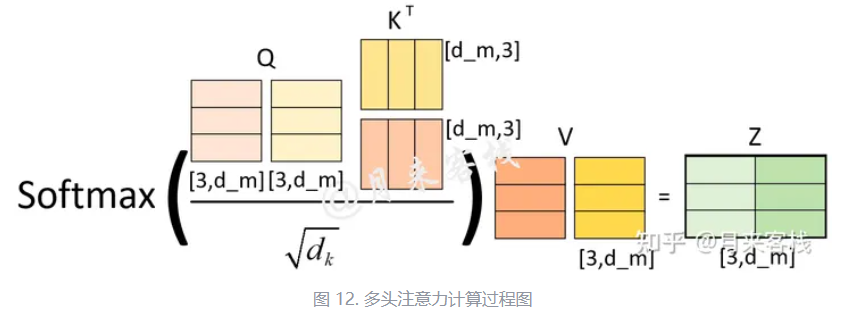
* Transformer为何使用多头注意力机制：**多头保证了transformer可以注意到不同子空间的信息，捕捉到更加丰富的特征信息。**

所谓的多头注意力机制其实就是将原始的输入序列进行多组的自注意力处理过程；然后再将每一组自注意力的结果拼接起来进行一次线性变换得到最终的输出结果。具体的，其计算公式为：

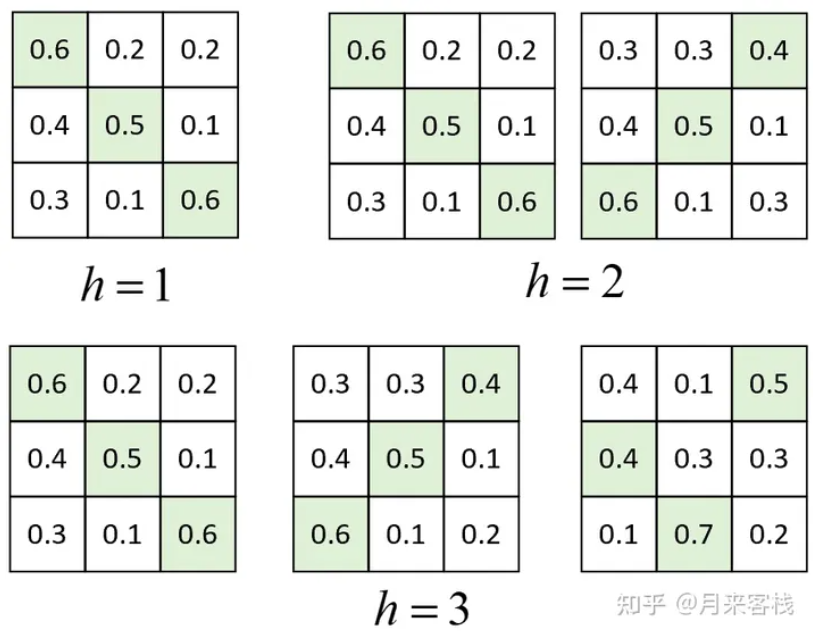




自注意力机制的缺陷就是：模型在对当前位置的信息进行编码时，会过度的将注意力集中于自身的位置，因此作者提出了通过多头注意力机制来解决这一问题。同时，使用多头注意力机制还能够给予注意力层的输出包含有不同子空间中的编码表示信息，从而增强模型的表达能力。



在dm固定的情况下，不管是使用单头还是多头的方式，在实际的处理过程中直到进行注意力权重矩阵计算前，两者之前没有任何区别。当进行进行注意力权重矩阵计算时，h越大那么Q,K,V就会被切分得越小，进而得到的注意力权重分配方式越多，



* Transformer为什么Q和K使用不同的权重矩阵生成，为何不能使用同一个值进行自身的点乘？

点乘的物理意义，两个向量的点乘表示两个向量的相似度。K和Q的点乘是为了计算一个句子中每个token相对于句子中其他token的相似度，这个相似度可以理解为attetnion score，关注度得分。经过与attention score相乘后，V中每个token的向量，在每一列上，都会对其他token做出调整（关注度不同）。与V相乘这一步，相当于提纯，让每个单词关注该关注的部分。

但是直接拿K和K点乘的话，你会发现attention score 矩阵是一个对称矩阵。因为是同样一个矩阵，都投影到了同样一个空间，所以泛化能力很差。

**使用Q/K/V不相同可以保证在不同空间进行投影，增强了表达能力，提高了泛化能力。**

* Transformer计算attention的时候为何选择点乘而不是加法？两者计算复杂度和效果上有什么区别

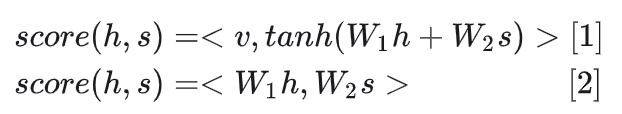
**计算效率：** 点乘（也称为内积）计算的复杂度较低。在点乘的注意力机制中，将查询向量和键向量进行点乘操作，然后对结果进行缩放（通过除以sqrt(d\_k)）以控制梯度的稳定性，最后通过softmax函数进行归一化。而加法注意力机制需要将查询向量和键向量相加，然后再进行softmax归一化。相比之下，点乘的计算量更小，尤其是在硬件加速器如GPU上更加高效。

**梯度传播：** 点乘的归一化（除以sqrt(d\_k)）可以控制梯度的大小，使得梯度更加稳定地传播。这有助于避免在深度网络中出现梯度爆炸或梯度消失的问题。

**模型性能：** 实践中发现，点乘的注意力机制在大多数情况下能够取得良好的效果，特别是在自然语言处理任务中，如机器翻译、文本生成等。

* 为什么在进行softmax之前需要对attention进行scaled（为什么除以dk的平方根）

Self-attention的两种基本形式：



<>代表矩阵点积，使用点乘的计算速度更快，整体计算复杂度二者相当，但是点乘有更多的硬件加速。随着attention dim的增大，加法形式的self-attention的性能强于点乘。点乘的性能更弱可能是由于极大的点积值将整个softmax推向梯度平缓区，使得收敛困难，导致梯度消失。如果s和h分布在[0,1]上，那么经过点乘之后整体的分布会在[0,dk]上。而除以根号dk是为了让QK的分布方差纠正回1。

* 在计算attention score的时候如何对padding做mask操作？

将padding位置取值为负无穷（如-1000），softmax的函数会导致为padding的值占全局一定概率，mask就是让这部分值取无穷小，让他在softmax之后基本也为0，不去影响非attention socore的分布。

* 为什么在进行多头注意力的时候需要对每个head进行降维？

计算效率： 注意力机制涉及对所有序列位置进行加权计算，这可能需要大量的计算资源，尤其是在深度学习模型中，如Transformer。通过降维，可以减少每个头的计算复杂度，从而提高整体模型的计算效率。

参数数量： 每个注意力头都具有自己的参数矩阵，降维可以减少每个头中参数的数量。这对于减少模型的参数量以及控制模型的过拟合有帮助。

更好的特征表达： 通过降维，可以将高维特征空间转换为低维空间，从而可能提取出更加抽象和有意义的特征表示。这有助于模型更好地学习输入序列之间的关系和语义信息。

减少过拟合： 随着模型深度的增加，过拟合成为一个常见的问题。通过降维，可以降低每个头的自由度，减少过拟合的风险。

更好的模型解释性： 降维可以使得每个头所关注的信息更加清晰，有助于理解模型在处理输入时的决策过程。

* MSA的代码

**import** torch

**import** torch.nn **as** nn

**import** torch.nn.functional **as** F

**import** math

**class** **MultiHeadAttention**(nn**.**Module):

**def** \_\_init\_\_(self, d\_model, num\_heads):

super(MultiHeadAttention, self)**.**\_\_init\_\_()

self**.**num\_heads **=** num\_heads

self**.**d\_model **=** d\_model

**assert** d\_model **%** self**.**num\_heads **==** 0

*# Define the dimension of each head or subspace*

self**.**d\_k **=** d\_model **//** self**.**num\_heads

*# These are still of dimension d\_model. They will be split into number of heads*

self**.**W\_q **=** nn**.**Linear(d\_model, d\_model)

self**.**W\_k **=** nn**.**Linear(d\_model, d\_model)

self**.**W\_v **=** nn**.**Linear(d\_model, d\_model)

*# Outputs of all sub-layers need to be of dimension d\_model*

self**.**W\_o **=** nn**.**Linear(d\_model, d\_model)

**def** **scaled\_dot\_product\_attention**(self, Q, K, V, mask**=**None):

batch\_size **=** Q**.**size(0)

K\_length **=** K**.**size(**-**2)

*# Scaling by d\_k so that the soft(arg)max doesn't explode*

QK **=** torch**.**matmul(Q, K**.**transpose(**-**2, **-**1)) **/** math**.**sqrt(self**.**d\_k)

*# Apply the mask*

**if** mask **is** **not** None:

QK **=** QK**.**masked\_fill(mask**.**to(QK**.**dtype) **==** 0, float('-inf'))

*# Calculate the attention weights (softmax over the last dimension)*

weights **=** F**.**softmax(QK, dim**=-**1)

*# Apply the self attention to the values*

attention **=** torch**.**matmul(weights, V)

**return** attention, weights

**def** **split\_heads**(self, x, batch\_size):

"""

The original tensor with dimension batch\_size \* seq\_length \* d\_model is split into num\_heads

so we now have batch\_size \* num\_heads \* seq\_length \* d\_k

"""

**return** x**.**view(batch\_size, **-**1, self**.**num\_heads, self**.**d\_k)**.**transpose(1, 2)

**def** **forward**(self, q, k, v, mask**=**None):

batch\_size **=** q**.**size(0)

*# linear layers*

q **=** self**.**W\_q(q)

k **=** self**.**W\_k(k)

v **=** self**.**W\_v(v)

*# split into multiple heads*

q **=** self**.**split\_heads(q, batch\_size)

k **=** self**.**split\_heads(k, batch\_size)

v **=** self**.**split\_heads(v, batch\_size)

*# self attention*

scores, weights **=** self**.**scaled\_dot\_product\_attention(q, k, v, mask)

*# concatenate heads*

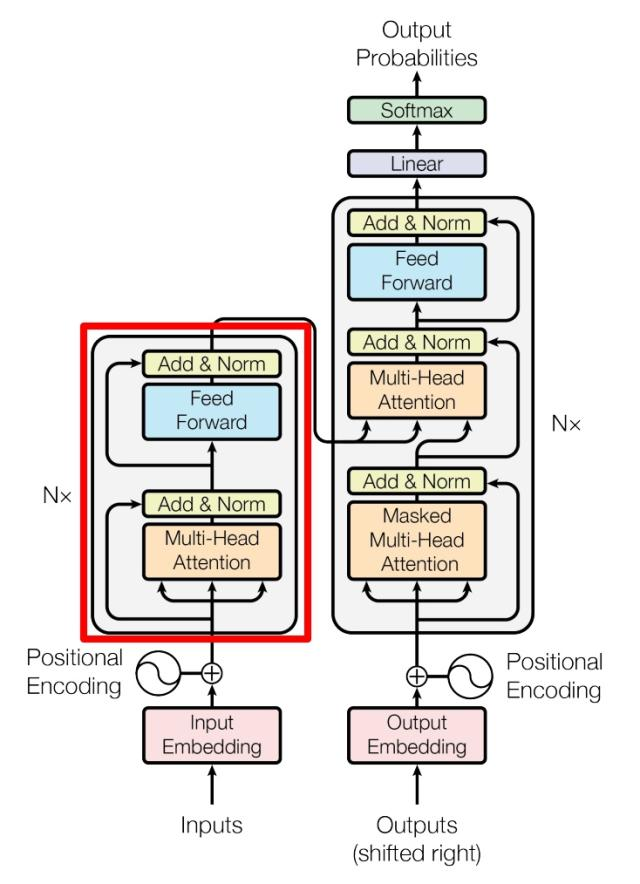
concat **=** scores**.**transpose(1,2)**.**contiguous()**.**view(batch\_size, **-**1, self**.**d\_model)

*# final linear layer*

output **=** self**.**W\_o(concat)

**return** output, weights

* Transformer的Encoder模块



Encoder由多个相同的层堆叠而成，每个层通常包括自注意力机制和前馈神经网络。

1. 自注意力层：使用自注意力机制对输入的上下文进行建模，捕获长距离的依赖关系；
2. 残差连接：将自注意力层的输入和输出相加，以便再梯度反向传播是防止梯度消失或爆炸；
3. 层归一化（Layer Normalization）：对残差连接的结果进行归一化，缓解训练过程中的梯度问题；
4. 前馈神经网络：通过两个FC和ReLU对上下文表示进行非线性变换和特征提取；
5. 再次残差连接和层归一化

* Transformer的位置编码，有什么意义和优缺点？

(1)对于任何一门语言，单词在句子中的位置以及排列顺序是非常重要的，它们不仅是一个句子的语法结构的组成部分，更是表达语义的重要概念。一个单词在句子的位置或排列顺序不同，可能整个句子的意思就发生了偏差。(2)当抛弃循环神经网络结构，完全采用Attention取而代之，这些词序信息就会丢失，模型就没有办法知道每个词在句子中的相对和绝对的位置信息。因此，有必要把词序信号加到词向量上帮助模型学习这些信息，位置编码（Positional Encoding）就是用来解决这种问题的方法。

优点：**简单有效**： 位置编码是一种简单而有效的机制，可以直接与输入嵌入相加，不需要额外的学习参数。**不受序列长度限制**： 位置编码与序列长度无关，可以应用于任何长度的输入序列。**允许模型学习序列顺序信息**： 通过位置编码，Transformer 能够学习到输入序列中词语的相对位置信息，有助于理解序列中的语义和语法结构。

缺点：**固定的位置信息**： 位置编码假设了输入序列中词的位置信息是固定的，这可能不适用于一些上下文动态变化的任务。**位置信息可能不够准确**： 对于较长的序列，位置编码可能无法提供足够精确的位置信息，导致模型难以理解长距离依赖关系。**不适用于变长序列**： 对于变长序列，如机器翻译中的句子长度可能不同，位置编码的设计可能需要额外的处理。

* 还有哪些关于位置编码的技术，各自的优缺点是什么？

**绝对位置编码（Absolute Positional Encoding）：**

常用的绝对位置编码方法包括使用三角函数（如正弦和余弦函数）或者使用固定的矩阵来编码序列位置信息。

优点：简单易实现，不需要额外的学习参数。

缺点：固定的位置编码可能无法捕捉输入序列中的动态模式，尤其是在处理变长序列时可能效果不佳。对于较长的序列，可能无法提供足够精确的位置信息。

**相对位置编码（Relative Positional Encoding）：**

一种实现方法是通过在自注意力计算中引入相对位置偏移量，并根据相对位置偏移量来调整注意力分数，从而捕捉输入序列中词与词之间的相对位置关系。

优点：能够捕捉输入序列中词与词之间的相对位置关系，有助于理解长距离依赖关系。在处理变长序列时可能比绝对位置编码更灵活。

缺点：相对位置编码的设计和实现可能较复杂，需要额外的计算开销。

**学习式位置编码（Learned Positional Encoding）：**

学习式位置编码通常是通过在模型中引入额外的可学习参数来实现的，这些参数可以被模型端到端地训练。

优点：允许模型通过学习来适应不同任务和数据集的位置信息。可以灵活地调整位置编码的参数，以适应不同长度和结构的输入序列。

缺点：需要额外的学习参数，并且可能增加模型的训练和推理成本。学习式位置编码的性能高度依赖于训练数据的质量和规模。

* Transformer中的残差结构以及意义

在 Transformer 的每个子层（如自注意力层和前馈全连接层）中，都会使用残差连接。

**梯度传播：** 残差连接有助于缓解梯度消失或梯度爆炸问题。通过将输入直接传递到输出中，可以确保即使在非常深的网络中，梯度仍然能够从输出反向传播到输入，有助于更有效地训练深层网络。

**减轻训练难度：** 在深度神经网络中，通过将每个层的输出与其输入相加，可以确保网络的训练过程更加稳定，更容易收敛到较好的解决方案。

**网络深度：** 残差连接允许构建非常深的网络，而不会出现梯度消失的问题。这在 Transformer 中尤为重要，因为 Transformer 模型通常包含大量的层，而残差连接确保了每个层都能够有效地学习到输入和输出之间的映射关系。

**网络表示能力：** 残差连接使得网络能够更好地学习数据的高级表示。通过残差连接，网络可以逐层地学习数据的不同抽象层次的表示，从而提高了网络的表示能力。

* 为什么transformer块使用LayerNorm而不是BatchNorm？LayerNorm 在Transformer的位置是哪里？

任何norm的意义都是为了让使用norm的网络的输入的数据分布变得更好，也就是转换为标准正态分布，数值进入敏感度区间，以减缓梯度消失，从而更容易训练。当然，这也意味着舍弃了除此维度之外其他维度的其他信息。首先要明确，如果在一个维度内进行normalization，那么在这个维度内，相对大小有意义的，是可以比较的；但是在normalization后的不同的维度之间，相对大小这是没有意义的。

**独立样本的处理：** BatchNorm 是在一个批次内对每个特征维度进行标准化，因此依赖于批次的统计信息。然而，在 Transformer 模型中，每个位置的输入都可以看作是一个单独的样本，而不是批次中的样本。因此，BatchNorm 不适用于 Transformer，因为它无法有效地对每个位置的输入进行归一化。相比之下，LayerNorm 对每个位置的输入独立进行归一化，更适合 Transformer 的结构。

**稳定性：** LayerNorm 在不同批次和不同输入之间的行为更加稳定，而 BatchNorm 对批次大小和输入分布更敏感。这使得 LayerNorm 更适合应对不同长度和类型的输入序列，保证了模型的稳定性和泛化性能。

**模型并行性**： 在训练大型 Transformer 模型时，常常需要使用模型并行（Model Parallelism）来分布式处理模型的不同部分。LayerNorm 相对于 BatchNorm 更易于实现模型并行，因为每个位置的归一化操作可以独立进行，而不需要跨多个设备共享统计信息。

LN的位置：在 Transformer 中，LayerNorm 通常被应用在每个子层的输入和输出上，包括自注意力层（Self-Attention Layer）和前馈全连接层（Feedforward Layer）。具体来说，LayerNorm 会对每个位置的输入进行归一化，然后再应用残差连接。这样可以确保每个位置的输入都具有相似的分布特性，有助于模型的稳定训练和更好的收敛性能。

* BatchNorm技术，以及它的优缺点。

优点：（1）加速收敛，通过归一化输入，减少梯度消失问题，使得网络更快地收敛；（2）减少对初始化的敏感性；（3）抑制过拟合，BN在每个batch上都有一定的噪声，这可以看作一种正则化技术；

缺点：（1）计算代价：需要额外的计算量来进行均值和方差的计算；（2）需要较大的batch size，为了保证每个batch的均值和方差估计的准确性，通常需要较大的batch-size；

* Transformer中的前馈神经网络，使用了什么激活函数？

Transformer中的FFN通常由两个线性层和一个激活函数（ReLU）组成，激活函数有利于引入非线性，从而增加模型的表达能力。FFN将输入的维度映射到一个更高的维度空间，然后再映射回到原始输入的维度空间，有助于学习到更复杂的非线性特征，并且能够在更高维的空间中更加灵活的表示。

* Encoder端和Decoder端是如何进行交互的

他们之间通过Encoder-Decoder注意力子层进行交互，该层允许Encoder在生成目标序列时对输入序列进行关注和对齐。在训练过程中，Decoder的每个位置都会利用Encoder的所有位置进行上下文感知，以便生成下一个目标词；

* Decoder阶段的多头自注意力和encoder的多头自注意力有什么区别？

Encoder 的多头自注意力（Encoder Self-Attention）：（1）在 Encoder 部分，MSA用于捕捉输入序列中的词语之间的关系。每个词语都会与输入序列中的其他词语进行注意力计算，以便于模型理解每个词语在整个句子中的语义信息和重要性。（2）Encoder 的自注意力层只关注输入序列，它没有访问到 Decoder 部分的信息。因此，它在计算注意力权重时，只使用了 Encoder 输入序列的信息，没有考虑到 Decoder 部分的输出。

Decoder 的多头自注意力（Decoder Self-Attention）：（1）在 Decoder 部分，多头自注意力机制也用于捕捉输入序列中的词语之间的关系，但是它的输入包含两部分：Decoder 自身的输出和 Encoder 的输出。（2）Decoder 的自注意力层除了关注输入序列，还需要考虑到之前 Decoder 部分生成的输出，以便于模型在生成每个词语时考虑到之前生成的词语的信息，从而保证生成的序列连贯性。（3）Decoder 的自注意力层通过使用 Mask 来确保模型只能关注当前位置之前的词语，而不能关注到未来的词语，以避免信息泄露和模型在推理时出现错误。

* Transformer的并行化提现在哪个地方？Decoder端可以做并行化吗？

Transformer的Decoder推理时不可以并行，只能一个一个解码，当前时刻的输入依赖于上一时刻的输入以及Encoder的输出。但是在训练的时候可以通过Mask矩阵进行并行训练，但是预测的时候只能串行。

对于Encoder来说，Encoder block之间是串行的，每个Encoder中的MSA和FFN是串行的，但是MSA和FFN自身是可以并行的。

* Transformer训练的时候学习率是如何设定的？Dropout是如何设定的，位置在哪里？Dropout 在测试的需要有什么需要注意的吗？

Transformer训练过程中通常采用warmup and decay策略，目的是在训练初期保持较高的学习率以加快收敛速度，并在后期逐渐降低学习率以细化模型参数的调整，Warmup有助于模型在训练初期参数调整过于剧烈，从而提高模型的稳定性。

Dropout通常在enbedding的output，attention softmax的output，attention的output和FFN的输出，有助于减少自注意力机制过度依赖某些输入的情况，减少FFN对输入的过拟合，提高模型的泛化能力。

测试时，去掉Dropout层，将所有输出利用，但是需要对齐尺度，即缩小输出比例。R=R \*(1-P)。特别的， 为了使用方便，我们不在测试时再缩小输出，而在训练时直接将输出放大1/(1-p)倍。

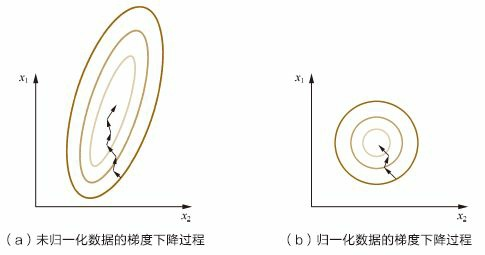
作用：Dropout达到了Vote的作用，减少神经元之间复杂的共适应性，可以比较有效地减轻过拟合，一定程度上达到了正则化的效果。

生成模型

特征工程

* 为什么要对数值类型的特征做归一化：

对数值类型的特征做归一化可以将所有的特征都统一到一个大致相同的数值区间内。在学习速率相同的情况下， *x1*的更新速度会大于*x2*， 需要较多的迭代才能找到最优解。在实际应用中， 通过梯度下降法求解的模型通常是需要归一化的。



* 对数据进行预处理时，应该怎样处理类别特征：

序号编码，例如，成绩可以分为低中高三档，分别用1，2，3表示；

One-hot编码：当类别间不具有大小关系特征时，例如，血型。在使用one-hot编码时需要注意（1）使用稀疏向量来节省空间，（2）配合特征选择来降低维度；

二进制编码：本质上时利用二进制对ID进行哈希映射，维数少于one-hot编码，节省存储空间。

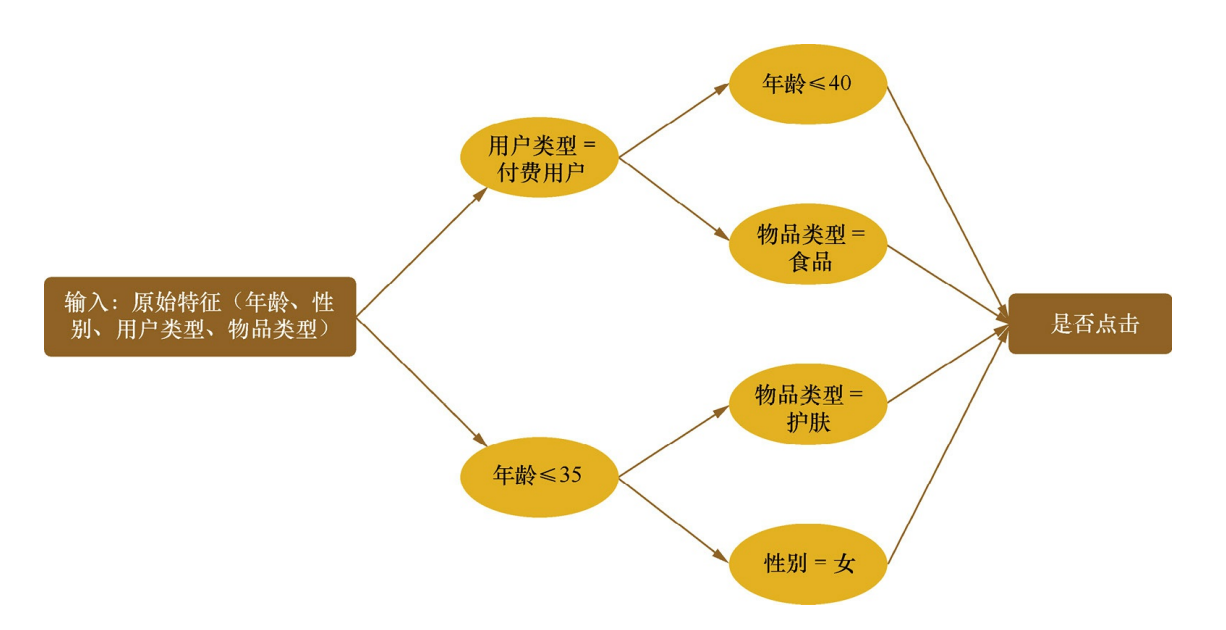
* 什么是组合特征？如何处理高维组合特征？

为了提高复杂关系的拟合能力，在特征工程中经常会把**一阶离散特征**两两组 合，构成高阶组合特征。当引入ID后，若用户的数量为m、物品的数量为n，那么需要学习的参数的规模为m×n。在 互联网环境下，用户数量和物品数量都可以达到千万量级，几乎无法学习m×n规模 的参数。在这种情况下，一种行之有效的方法是将用户和物品分别用k维的低维向 量表示



* 怎样有效地找到组合特征？

可以采用决策树寻找组合特征。可以采用梯度提升决策树，该方 法的思想是每次都在之前构建的决策树的残差上构建下一棵决策树。



* 有哪些文本表示模型？它们各有什么优缺点？

**词袋模型和N-gram模型**：最基础的文本表示模型是词袋模型。就是将整段文本以词为单位切分开， 然后每篇文章可以表示成一个长向量，向量中的每一维代表一个单词，而该维对 应的权重则反映了这个词在原文章中的重要程度。然而将文章进行单词级别的划分有时候并不是一种好的做法，因此可以将连续 出现的n个词（n≤N）组成的词组（N-gram）也作为一个单独的特征放到向量表示 中去，构成N-gram模型。

**主题模型**：用于从文本库中发现有代表性的主题（得到每个主题上面词的分布 特性），并且能够计算出每篇文章的主题分布；

**词嵌入与深度学习模型**：核心思想是将每个词都映射成低维 空间（通常K=50～300维）上的一个稠密向量（Dense Vector）。

* Word2Vec是如何工作的？它和LDA有什么区别与联系？

CBOW的目标是根据上下文出现的词语来预测当前词的生成概率；而Skip-gram是根据当前词来预测上下文中各词的生成概率。



LDA是利用文档中单词的共现关系来对单词按主题聚类，也可以理解为对“文档-单词”矩阵进行分解，得到“文档主题”和“主题-单词”两个概率分布。而Word2Vec其实是对“上下文-单词”矩阵进行学习，其中上下文由周围的几个单词组成，由此得到的词向量表示更多地融入了 上下文共现的特征。

* 在图像分类任务中，训练数据不足会带来什么问题？如何缓解数据量不足带 来的问题？

训练数据不足带来的问题主要表现在过拟合方面，即模型在训练样本上的效果可能不错，但在测试集上的泛化效果不佳。一是基于模型的方法，主要是采用降低过拟合风险的措施，包括简化模型（如将非线性模型简化为线性模型）、添加约束项以缩小假设空间（如L1/L2正则项）、集成学习、Dropout超参数等；二是基于数据的方法，主要通过数据扩充（Data Augmentation），即根据一些先验知识，在保持特定信息的前提下，对原始数据进行适当变换以达到扩充数据集的效果。使用生成模型也可以合成一些新样本。借助已有的其他模型或数据来进行迁移学习。

模型评估

* 准确率的局限性。

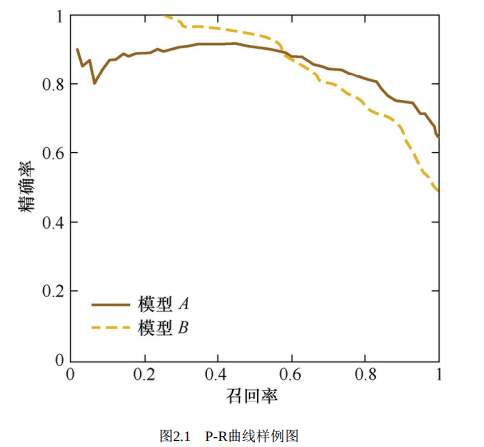
准确率是分类问题中最简单也是最直观的评价指标，但存在明显的缺陷。当不同类别的样本比例非常不均衡时，占比大的类别往往成为影响准确率的最主要因素。为了解决这个问题，可以使用更为有效的平均准确率（每个类别下的样本准确率的算术平均）作为模型评估的指标。

* 精确率和召回率的权衡

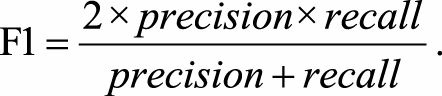
精确率是指分类正确的正样本个数占分类器判定为正样本的样本个数的比例。 召回率是指分类正确的正样本个数占真正的正样本个数的比例。

为了综合评估一个排序模型的好坏， 不仅要看模型在不同Top N下的Precision@N和Recall@N，而且最好绘制出模型的P-R（PrecisionRecall）曲线。

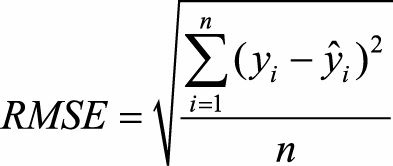
对于一个排序模型来说， 其P-R曲线上的一个点代表着， 在某一阈值下， 模型将大于该阈值的结果判定为正样本，小于该阈值的结果判定为负样本， 此时返回结果对应的召回率和精确率。



除此之外， F1 score和ROC曲线也能综合地反映一个排序模型的性能。 F1 score是精准率和召回率的调和平均值， 它定义为

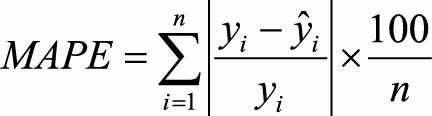


* 造成RMSE指标居高不下的原因

RMSE的计算公式：

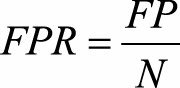
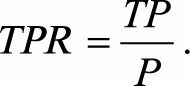
在实际问题中， 如果存在个别偏离程度非常大的离群点（ Outlier） 时， 即使离群点数量非常少， 也会让RMSE指标变得很差。

解决方法：（1）过滤掉离群点；（2）将离群点产生的机制建模进去；（3）找一个更加合适的指标来评估模型，比如平局绝对百分比误差MAPE，把每个点的误差进行了归一化，降低了个别离群点带来的绝对误差的影响；



* ROC曲线

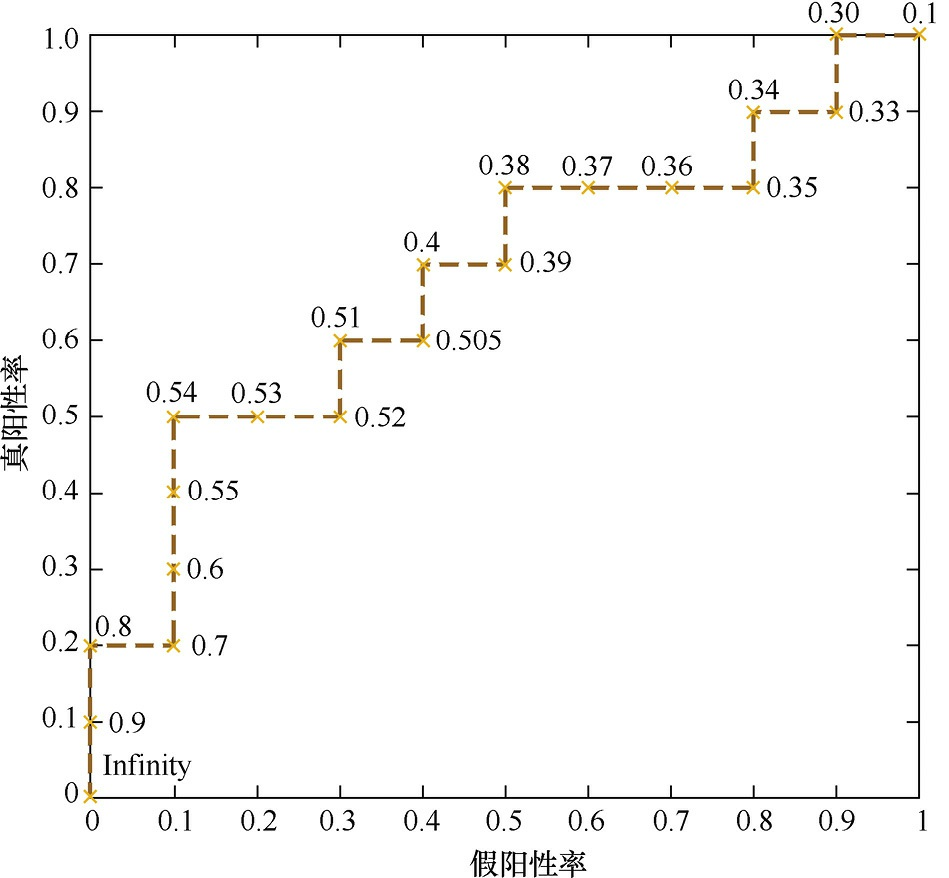
ROC曲线是Receiver Operating Characteristic Curve的简称， 中文名为“受试者工作特征曲线”。 ROC曲线的横坐标为假阳性率（False Positive Rate， FPR） ； 纵坐标为真阳性率（True Positive Rate， TPR）。

，

上式中， P是真实的正样本的数量， N是真实的负样本的数量， TP是P个正样本中被分类器预测为正样本的个数， FP是N个负样本中被分类器预测为正样本的个数。

* ROC曲线的绘制

ROC曲线是通过不断移动分类器的“截断点”来生成曲线上的一组关键点的。ROC曲线都是从（0， 0）到（1， 1）的一条曲线。



* AUC的计算

AUC指的是ROC曲线下的面积大小， 该值能够量化地反映基于ROC曲线衡量出的模型性能。 计算AUC值只需要沿着ROC横轴做积分就可以了。AUC的取值一般在0.5~1之间，AUC越大， 说明分类器越可能把真正的正样本排在前面， 分类性能越好。

* ROC曲线相比P-R曲线有什么特点

当正负样本的分布发生变化时，ROC曲线的形状能够基本保持不变，而P-R曲线的形状一般会发生较剧烈的变化。使得ROC曲线能够尽量降低不同测试集带来的干扰，更加客观的衡量模型本身的性能。但如果希望看到模型在特定数据集上的表现，P-R曲线则更能直观地反应其性能。

* 为什么在一些场景中要使用余弦相似度而不是欧式距离

余弦相似度地定义为，关注的是向量之间的角度关系， 并不关心它们的绝对大小， 其取值范围是[-1,1]。此外，在文本、 图像、视频等领域， 研究的对象的特征维度往往很高， 余弦相似度在高维情况下依然保持“相同时为1， 正交时为0， 相反时为-1”的性质， 而欧氏距离的数值则受维度的影响， 范围不固定， 并且含义也比较模糊。

总体来说，欧氏距离体现数值上的绝对差异， 而余弦距离体现方向上的相对差异。 例如， 统计两部剧的用户观看行为， 用户A的观看向量为(0,1)， 用户B为(1,0)； 此时二者的余弦距离很大， 而欧氏距离很小；

* 余弦距离是否为一个严格定义的距离

余弦距离满足正定性和对称性， 但是不满足三角不等式， 因此它并不是严格定义的距离。此外，KL距离也叫相对熵，它常用于计算两个分布之间的差异，但不满足对称性和三角不等式。

* 在对模型进行过充分的离线评估之后，为什么还要进行在线A/B测试？

（1）离线评估无法完全消除模型过拟合的影响；（2）离线评估无法完全还原线上的工程环境，例如延迟，数据丢失，标签数据确实等情况；（3）线上系统的某些商业指标评估中无法计算。

* 如何进行线上A/B测试

进行A/B测试的主要手段是进行用户分桶，即将用户分成实验组和对照组，对 实验组的用户施以新模型，对对照组的用户施以旧模型。在分桶的过程中，注意样本的独立性和采样方式的无偏性，确保同一个用户每次只能分到同一个桶中。

* 在模型评估过程中，有哪些主要的验证方法

**Holdout检验**：将原始的样本集合随即划分成训练集和验证集，在验证集上绘制ROC曲线，计算精确率和召回率来评估模型性能。因此Holdout的缺点是在验证集上计算出来的最后评估指标与原始分组有很大关系。

**K折交叉验证**：首先将全部样本划分成k个大小相等的样本子集； 依次遍历这k个子集， 每次把当前子集作为验证集， 其余所有子集作为训练集， 进行模型的训练和评估； 最后把k次评估指标的平均值作为最终的评估指标。

**留一验证**：每次留下1个样本作为验证集， 其余所有样本作为测试集。因此它的时间开销更是远远高于留一验证， 故而很少在实际工程中被应用。

**自助法**：当样本规模比较小时， 将样本集进行划分会让训练集进一步减小， 这可能会影响模型训练效果。自助法可以维持训练集样本规模。对于总数为n的样本集合， 进行n次有放回的随机抽样， 得到大小为n的训练集。 n次采样过程中， 有的样本会被重复采样， 有的样本没有被抽出过， 将这些没有被抽出的样本作为验证集， 进行模型验证

* 超参数有哪些调优方法

**网格搜索**：它通过查找搜索范围内的所有的点来确定最优值。然而， 这种搜索方案十分消耗计算资源和时间。在实际应用中， 网格搜索法一般会先使用较广的搜索范围和较大的步长； 然后会逐渐缩小搜索范围和步长，来寻找更精确的最优值

**随机搜索**：思想与网格搜索比较相似， 只是不再测试上界和下界之间的所有值， 而是在搜索范围中随机选取样本点。如果样本点集足够大， 那么通过随机采样也能大概率地找到全局最优值。随机搜索一般会比网格搜索要快一些， 但是和网格搜索的快速版一样， 它的结果也是没法保证的。

**贝叶斯优化算法**：网格搜索和随机搜索在测试一个新点时， 会忽略前一个点的信息；而贝叶斯优化算法则充分利用了之前的信息。 贝叶斯优化算法通过对目标函数形状进行学习， 找到使目标函数向全局最优值提升的参数。

* 过拟合和欠拟合的具体现象

过拟合是指模型对于训练数据拟合呈过当的情况在训练集上的表现很好， 但在测试集和新数据上的表现较差。欠拟合指的是模型在训练和预测时表现都不好的情况。

* 降低过拟合的方法：

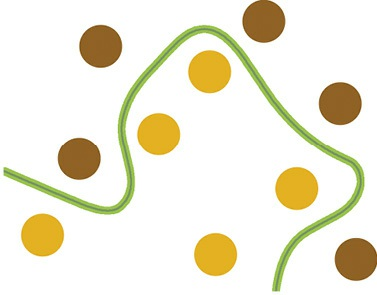
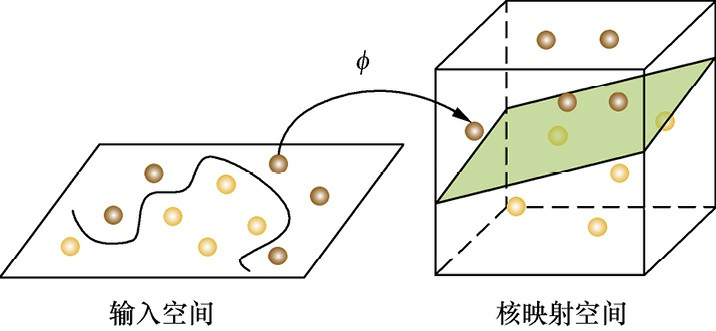
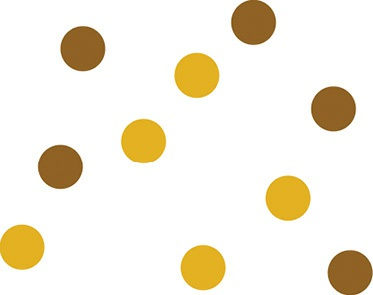
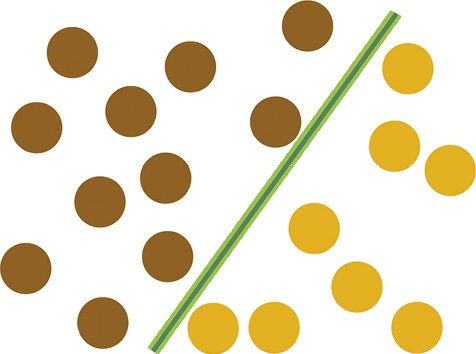
（1）从数据入手， 获得更多的训练数据，进行数据增广。（2）降低模型复杂度，减少网络层数，神经元个数，降低决策树模型的的深度。（3）正则化方法，L2正则化；（4）集成学习，把多个模型集成在一起。

* 降低欠拟合的方法：

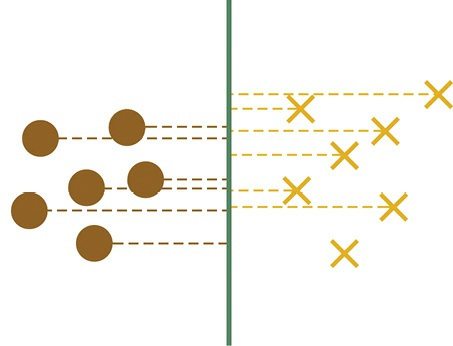
（1）增加新特征；（2）增加模型复杂度；（3）减小正则化系数

经典算法

* 支持向量机（Support Vector Machine， SVM）：是一种有监督学习方法。



* 在空间上线性可分的两类点，分别向SVM的分类超平面上做投影，这些点在超平面上的投影仍然是线性可分的吗



并且，对于任意线性可分的两组点， 它们在SVM分类的超平面上的投影都是线性不可分的。

* SVM的特点：

存在一组参数使得SVM训练误差为0；训练误差为0的SVM分类器一定存在，加入松弛变量的SVM的训练误差不一定能达到0.

* 逻辑回归与线性回归有何异同

逻辑回归处理的是分类问题， 线性回归处理的是回归问题， 这是两者的最本质的区别。逻辑回归中的因变量为离散的， 而线性回归中的因变量是连续的。逻辑回归中， 因变量取值是一个二元分布， 模型学习得出的是，即给定自变量和超参数后， 得到因变量的期望， 并基于此期望来处理预测分类问题。 而线性回归中实际上求解的是。

实际上，将逻辑回归的公式进行整理， 我们可以得到，其中， 也就是将给定输入x预测为正样本的概率。逻辑回归可以看作是对于，预测为正样本的对数几率的线性回归。

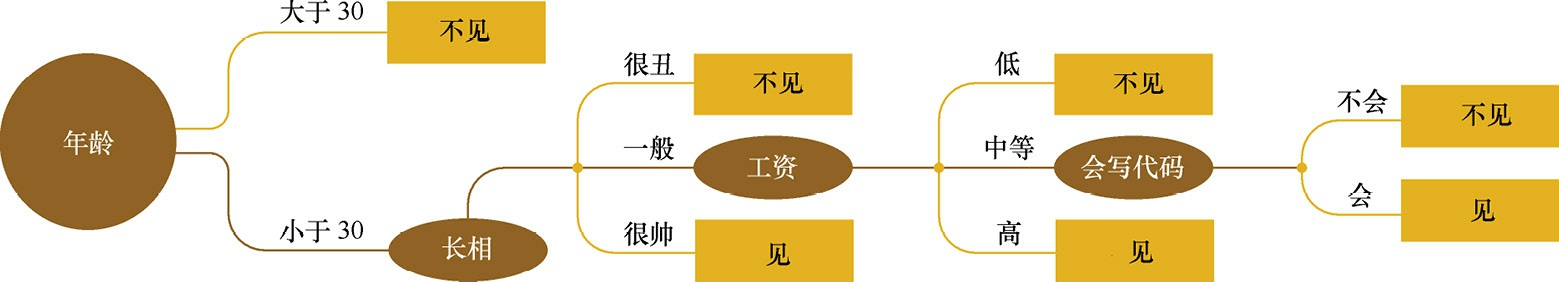
* 如何使用逻辑回归处理多标签的分类问题

如果一个样本只能对应于一个标签，我们可以假设每个样本属于不同标签的概率服从几何分布，使用多项逻辑回归（Softmax Regression）来进行分类。

当存在样本可能属于多个标签的情况时， 我们可以训练k个二分类的逻辑回归分类器。 第i个分类器用以区分每个样本是否可以归为第i类， 训练该分类器时， 需要把标签重新整理为“第i类标签”与“非第i类标签”两类。

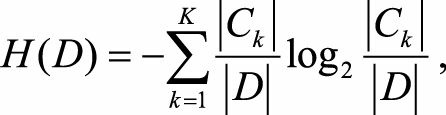
* 决策树

决策树是一种自上而下， 对样本数据进行树形分类的过程， 由结点和有向边组成。决策树作为最基础、 最常见的有监督学习模型， 常被用于分类问题和回归问题。一般而言， 决策树的生成包含了特征选择、 树的构造、 树的剪枝三个过程



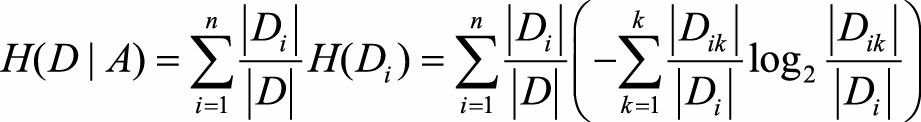
* 决策树的启发函数

**ID3—最大信息增益**：对于样本集合D， 类别数为K， 数据集D的经验熵表示为



其中Ck是样本集合D中属于第k类的样本子集， |Ck|表示该子集的元素个数， |D|表示样本集合的元素个数。

然后计算某个特征A对于数据集D的经验条件熵H(D|A)为



其中， Di表示D中特征A取第i个值的样本子集， Dik表示Di中属于第k类的样本子集。于是信息增益g(D,A)可以表示为二者之差， 可得



根据所有特征的信息增益，可以将增益最大的分到叶节点中。

ID3是采用信息增益作为评价标准，会倾向于取值较多的特征。因为， 信息增益反映的是给定条件以后不确定性减少的程度， 特征取值越多就意味着确定性更高， 也就是条件熵越小， 信息增益越大。

**C4.5—最大信息增益比**：

**CART—最大基尼指数**：

* 决策树的剪枝：

决策树的剪枝通常有两种方法， 预剪枝（Pre-Pruning） 和后剪枝（Post-Pruning） 。

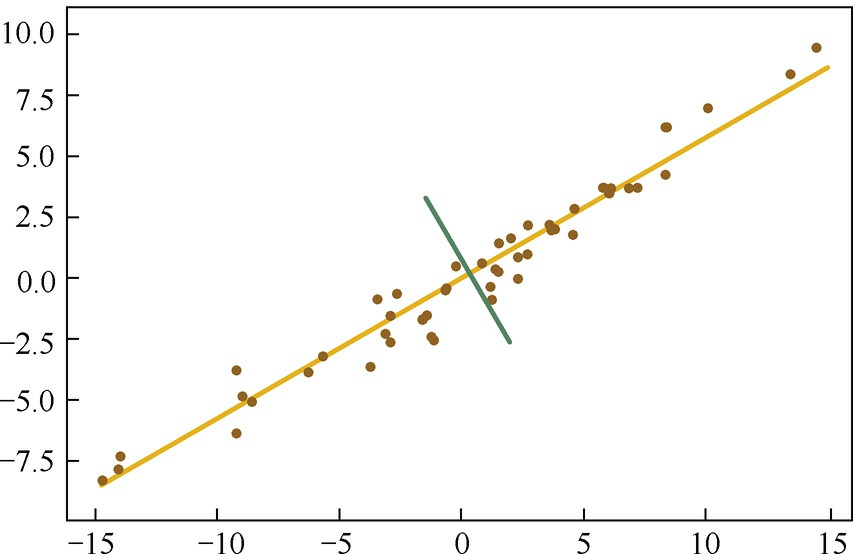
预剪枝的核心思想是在树中结点进行扩展之前， 先计算当前的划分是否能带来模型泛化能力的提升， 如果不能， 则不再继续生长子树。预剪枝的方法：（1）当树到达一定深度的时候， 停止树的生长。（2）当到达当前结点的样本数量小于某个阈值的时候， 停止树的生长。（3）计算每次分裂对测试集的准确度提升， 当小于某个阈值的时候， 不再继续扩展。预剪枝具有思想直接、 算法简单、 效率高等特点， 适合解决大规模问题。

后剪枝的核心思想是让算法生成一棵完全生长的决策树， 然后从最底层向上计算是否剪枝。剪枝过程将子树删除， 用一个叶子结点替代， 该结点的类别同样按照多数投票的原则进行判断。

降维

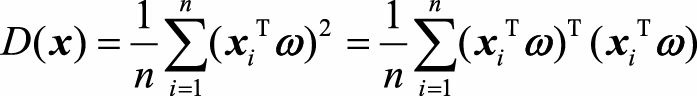
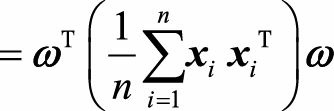
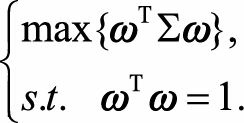
* 如何定义主成分

PCA旨在找到数据中的主成分， 并利用这些主成分表征原始数据， 从而达到降维的目的。通过x′,y′两个维度表达原始数据， 并且没有任何损失， 这样就完成了数据的降维。 而x′,y′两个轴所包含的信息就是我们要找到的主成分。

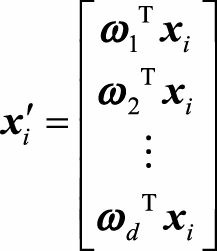


* 如何设计目标函数使得降维达到提取主成分的目的

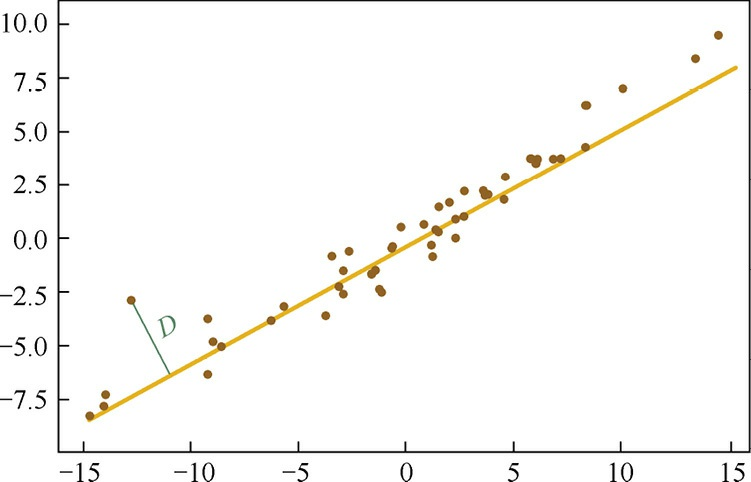
对于给定的一组数据点，中心化后的表示为=。向量内积在几何上表示为第一个向量投影到第二个向量上的长度， 因此向量xi在ω（ 单位方向向量） 上的投影坐标可以表示为。所以目标是找到一个投影方向ω，使得在ω上的投影方差尽可能大。因为，投影后的均值为0，这也是中心化的意义。

投影后的方差为：  ，因此求解一个最大化问题：

求解步骤：（1）对样本数据进行中心化处理；（2）求样本协方差矩阵；（3）最协方差矩阵进行特征值分解，将特征值从大到小排列；（4）去特征值前d大对应的特征向量*ω1*,*ω2*,...,*ωd，*通过映射将n维样本映射到d维：



PCA求解的其实时最佳投影方向，即一条直线，这与数学中线性回归问题的目标不谋而合，其目标也是求解一个线性函数使得对应直线能够更好地拟合样本点集合。



最小化样本点到直线的距离平方之和

数据集中每个点xk到d维超平面D的距离为：



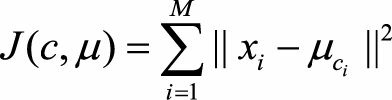
* 线性判别分析（Linear Discriminant Analysis，LDA）

LDA是一种有监督学习算法，同时经常被用来对数据进行降维。

非监督学习

相比于监督学习， 非监督学习的输入数据没有标签信息， 需要通过算法模型来挖掘数据内在的结构和模式。 非监督学习主要包含两大类学习方法： 数据聚类和特征变量关联。 其中， 聚类算法往往是通过多次迭代来找到数据的最优分割，而特征变量关联则是利用各种相关性分析方法来找到变量之间的关系。

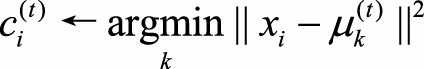
K均值聚类（KMeans Clustering） 是最基础和最常用的聚类算法。 它的基本思想是， 通过迭代方式寻找K个簇（Cluster） 的一种划分方案， 使得聚类结果对应的代价函数最小。代价函数可以定义为各个样本距离所属簇中心点的误差平方和：



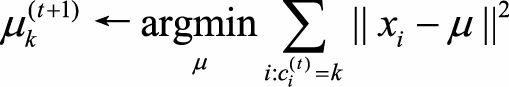
* 简述K均值算法的具体步骤

（1）数据预处理，归一化，离群点处理；（2）随机选取K个簇中心；（3）定义代价函数；（4）进行迭代，直到J收敛；

对于每一个样本，将其分配到距离最近的簇：



对于每一个簇，重新计算该类的簇中心：



* K均值的优缺点

缺点：例如受初值和离群点的影响每次的结果不稳定、 结果通常不是全局最优而是局部最优解、 无法很好地解决数据簇分布差别比较大的情况（比如一类是另一类样本数量的100倍） 、 不太适用于离散分类等。

优点：对于大数据集， K均值聚类算法相对是可伸缩和高效的， 它的计算复杂度是O(NKt)接近于线性， 其中N是数据对象的数目， K是聚类的簇数， t是迭代的轮数。

* K均值算法的调优

（1）数据归一化和离群点处理，K均值本质上是一种基于欧式距离度量的数据划分方法，均值和方差大的维度将对数据的聚类结果产生决定性的影响，所以未做归一化处理和统一单位的数据是无法直接参与运算和比较的；同时离群点会对数据的均值产生较大的影响，导致中兴偏移。

（2）合理选择K值，K值得选择一般基于经验和多次实验结果；

（3）采用核函数，对于非凸得数据分布形状时，需要引入和函数来优化，将输入空间中得数据点映射到高位得特征空间中。

优化算法

* 有监督学习涉及到的损失函数有哪些

对二分类问题， Y={1,-1}，最自然的损失函数是0-1损失， 即 

但是0-1损失非凸、 非光滑的特点，使得算法很难直接对该函数进行优化。

0-1损失的一个代理损失函数是Hinge损失函数：



Hinge损失在fy=1处不可导， 因此不能用梯度下降法进行优化， 而是用次梯度下降法（ Subgradient Descent Method） 。

0-1损失的另一个代理损失函数是Logistic损失函数：

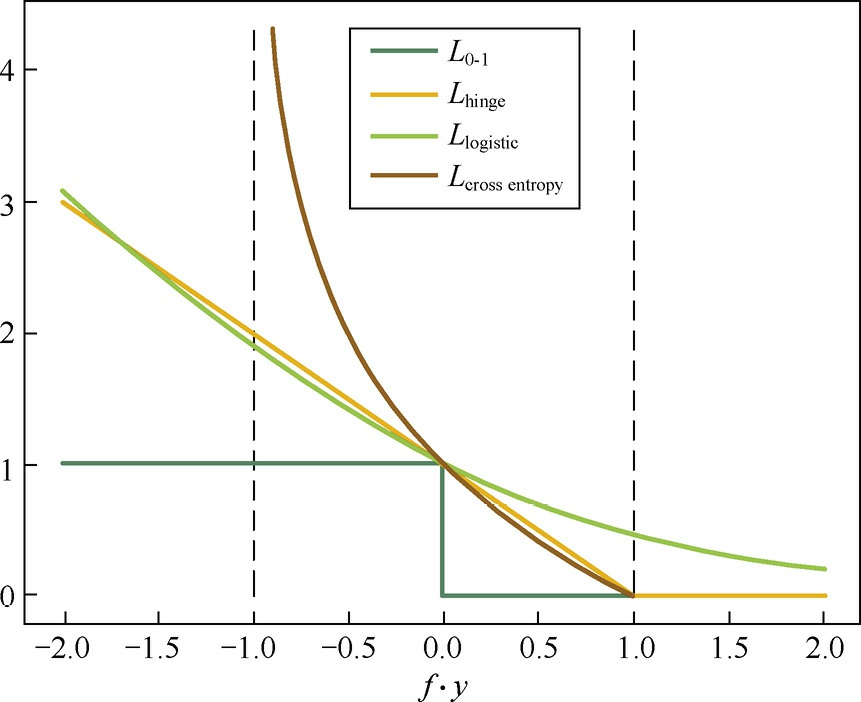


Logistic损失函数也是0-1损失函数的凸上界， 且该函数处处光滑， 因此可以用梯度下降法进行优化。 但是， 该损失函数对所有的样本点都有所惩罚， 因此对异常值相对更敏感一些。

另一个常用的代理损失函数是交叉熵（ Cross Entropy） 损失函数：



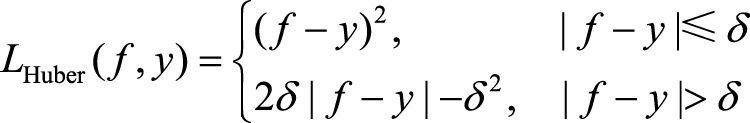
交叉熵损失函数也是0-1损失函数的光滑凸上界。



对于回归问题，最常用的损失函数是平方损失函数：平方损失函数是光滑函数， 能够用梯度下降法进行优化。 然而， 当预测值距离真实值越远时， 平方损失函数的惩罚力度越大， 因此它对异常点比较敏感。 为了解决该问题， 可以采用绝对损失函数：



但是， 绝对损失函数在f=y处无法求导数。 综合考虑可导性和对异常点的鲁棒性， 可以采用Huber损失函数



* 梯度下降算法：

在机器学习中， 优化问题的目标函数通常可以表示成因此， L(θ)刻画了当参数为θ时， 模型在所有数据上的平均损失。 我们希望能够找到平均损失最小的模型参数， 也就是求解优化问题



经典的梯度下降法采用所有训练数据的平均损失来近似目标函数， 即





其中M是训练样本的个数。 模型参数的更新公式为



因此， 经典的梯度下降法在每次对模型参数进行更新时， 需要遍历所有的训练数据。 当M很大时， 这需要很大的计算量， 耗费很长的计算时间， 在实际应用中基本不可行。

* 随机梯度下降法

用单个训练样本的损失来近似平均损失， 因此， 随机梯度下降法用单个训练数据即可对模型参数进行一次更新， 大大加快了收敛速率。





线程与进程：

线程间的通信方式：

进程间的通信方式：

面试题汇总

如何将RGB图像转成灰度图像

TCP与UDP

你了解设计模式吗

C++的特性

Python与C++的区别

多线程