目录

一、网易面试

1. 优化技巧，调参经验:

1.先在**小数据集**上实验，可以提高速度

2.一般**在对数尺度进行超参数搜索**，比如学习率和正则化项，

3.网上搜索**经验参数**

4.**自动调参**：Grid search, random search 以及贝叶斯优化

5.**可视化**，画图，比较直观，以此来调整

6.**从粗到细分阶段调参**

7.换**不同的初始化方法**试试，比如 normal, Xavier uniform

8.**数据增强**，训练的时候 multi scale resize + crop 固定大小,测试的时候取平均

2. sigmoid 和 tanh 的区别

两个是线性相关的，只是值域不一样而已，需要 0 到 1 就用 sigmoid，需要-1~1 就用 tanh. 在有 bias 的情况下，表达能力一样。

3.分类问题为什么不能用 MSE loss 而一定要用交叉熵

分类问题的目标变量是离散的，不像回归是连续的数值。分类问题，最后必须是 one hot 形式算出各 label 的概率，然后通过 argmax 选出最终的分类，在计算各个 label 的概率时，用的是 softmax 函数：

$$P(y=j|\mathbf{x}) = \frac{e^{\mathbf{x}^\top \mathbf{w}_j}}{\sum_{k=1}^{K} e^{\mathbf{x}^\top \mathbf{w}_k}}.$$

如果用 MSE 计算 loss，输出的曲线是有波动的，有很多局部的极小值点，即非凸优化。

而用交叉熵的话，是一个凸优化问题，用梯度下降求解时，凸优化问题有很好的收敛特性。

交叉熵损失用来度量两个概率分布之间的差异性(p 是真实的，q 是网络输出的)：

$$H(p,q) = -\sum_x p(x)\log q(x)$$

熵实际上是香农信息量（log(1/p)）的期望，还有 KL 散度(相对熵),注意 KL 散度不是度量，因为恒为非负，不具有对称性。

分类问题，都用 one-hot + cross entropy。

training 过程中，分类问题用 cross entropy，回归问题用 mean squared error。

validation/testing 时，使用 classification error，更直观，而且是我们最关注的指标。

4.决策树要保存到硬盘要存哪些内容

决策树是字典形式存储，由于写入和读出都必须是字符串格式，所以需要序列化和反序列化。Python 序列化保存决策树，使用 pickle.dump()。存储为{'no surfacing': {0: 'no', 1: {'flippers': {0: 'no', 1: 'yes'}}}}，就是嵌套的字典，存储决策树的判断条件和叶子节点。

5.从 bias-variance 的角度回答下 bagging 和 boosting 的区别在哪里

bagging 是减少 variance, boosting 是减少 bias.

Bagging 的算法原理和 boosting 不同，它的**弱学习器之间没有依赖关系**，可以并行生成。bagging 的**个体弱学习器的训练集是通过随机采样**得到的。通过 3 次的随机采样，我们就可以得到 3 个采样集，对于这 3 个采样集，我们可以分别独立的训练出 3 个弱学习器，再对这 3 个弱学习器通过集合策略来得到最终的强学习器。比如随机森林，它使用决策树作为基学习器。

Bagging对样本重采样，对每一重采样得到的子样本集训练一个模型，最后取平均。由于子样本集的相似性以及使用的是同种模型，因此各模型有近似相等的bias和variance（事实上，各模型的分布也近似相同，但不独立）。由于 $E[\frac{\sum X_i}{n}] = E[X_i]$，所以bagging后的bias和单个子模型的接近，一般来说不能显著降低bias。另一方面，若各子模型独立，则有 $Var(\frac{\sum X_i}{n}) = \frac{Var(X_i)}{n}$，此时可以显著降低variance。若各子模型完全相同，则 $Var(\frac{\sum X_i}{n}) = Var(X_i)$

，此时不会降低variance。bagging方法得到的各子模型是有一定相关性的，属于上面两个极端状况的中间态，因此可以一定程度降低variance。为了进一步降低variance，Random forest通过随机选取变量子集做拟合的方式de-correlated了各子模型（树），使得variance进一步降低。

（用公式可以一目了然：设有i.d.的n个随机变量，方差记为 σ^2，两两变量之间的相关性为 ρ，则 $\frac{\sum X_i}{n}$ 的方差为 $\rho * \sigma^2 + (1-\rho) * \sigma^2/n$

，bagging降低的是第二项，random forest是同时降低两项。详见ESL p588公式15.1）

Boosting 算法的工作机制是首先从训练集用初始权重训练出一个弱学习器 1，根据弱学习的学习误差率表现来更新训练样本的权重，使得之前弱学习器 1 学习误差率高的训练样本点的权重变高，使得这些误差率高的点在后面的弱学习器 2 中得到更多的重视。然后**基于调整权重后的训练集来训练弱学习器 2**.，如此重复进行，直到弱学习器数达到事先指定的数目 T，最终将这 T 个弱学习器通过集合策略进行整合，得到最终的强学习器。有 Adaboost 和基于决策树的 boost

boosting从优化角度来看，是用forward-stagewise这种贪心法去最小化损失函数 $L(y, \sum_i a_i f_i(x))$。例如，常见的AdaBoost即等价于用这种方法最小化exponential loss：$L(y, f(x)) = exp(-yf(x))$。所谓forward-stagewise，就是在迭代的第n步，求解新的子模型f(x)及步长a（或者叫组合系数），来最小化 $L(y, f_{n-1}(x) + af(x))$，这里 $f_{n-1}(x)$

是前n-1步得到的子模型的和。因此boosting是在sequential地最小化损失函数，其bias自然逐步下降。但由于是采取这种sequential、adaptive的策略，各子模型之间是强相关的，于是子模型之和并不能显著降低variance。所以说boosting主要还是靠降低bias来提升预测精度。

此外，集成学习除了 bagging 和 boosting，还有 Stacking。**Stacking** 将**训练好的所有基模型对训练基进行预测**，第 j 个基模型对第 i 个训练样本的**预测值将作为新的训练集中**

第 i 个样本的第 j 个特征值，最后基于新的训练集进行训练。同理，预测的过程也要先经过所有基模型的预测形成新的测试集，最后再对测试集进行预测。

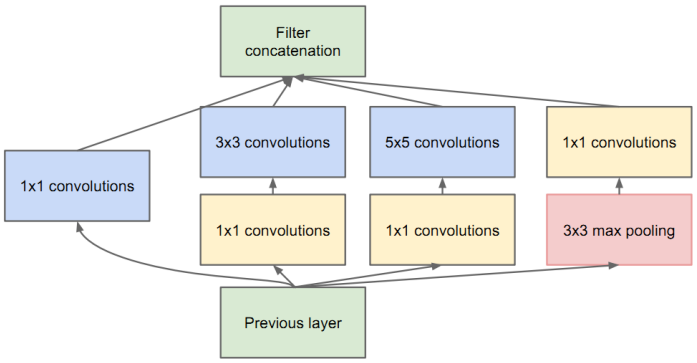
Stacking 算法分为 2 层，第一层是用不同的算法形成 T 个弱分类器，同时产生一个与原数据集大小相同的新数据集，利用这个新数据集和一个新算法构成第二层的分类器。Stacking 就像是 Bagging 的升级版，Bagging 中的融合各个基础分类器是相同权重，而 Stacking 中则不同，**Stacking 中第二层学习的过程就是为了寻找合适的权重或者合适的组合方式。**

6.历代的 CNN 模型的创新点

1.AlexNet：和 LeNet 相似，使用了 ReLU、Dropout，还有一些很好的训练技巧，如数据增广、学习率策略、weight decay 等

2.VGG：不再使用大卷积核，用**堆叠的 3*3 卷积来代替大卷积**，两个 3*3 就相当于一个 5*5，同时参数量更少，从而网络从 AlexNet 的 8 层增加到 19 层。

3.Inception：**multi-branch，每单元有很多层并行计算，网络更宽了**，inception module:



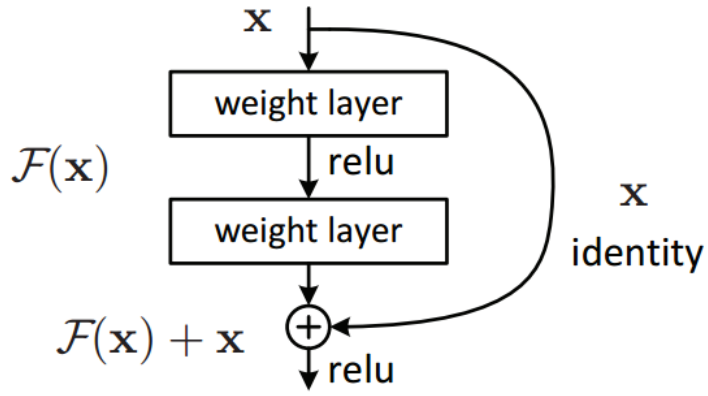
主要就是**增加了网络的宽度和深度，还有就是 1*1 卷积。**

后续的 V2 加入了 Batch Normalization, V3 增加了 1*1 的使用，V4 加入了残差。

4.ResNet：通过引入 **shortcut** 改进了 GoogLeNet 网络太深难以训练的问题。

原来需要学习完全的重构映射，现在只要学习输出和原来输入的差值，绝对量变相对量，容易训练。此外，通过**引入残差和 identity 恒等映射**，相当于构造了一个梯度高速通道，可以**避免梯度消失的问题**。从而网络从之前的 22 层到了 152 层。

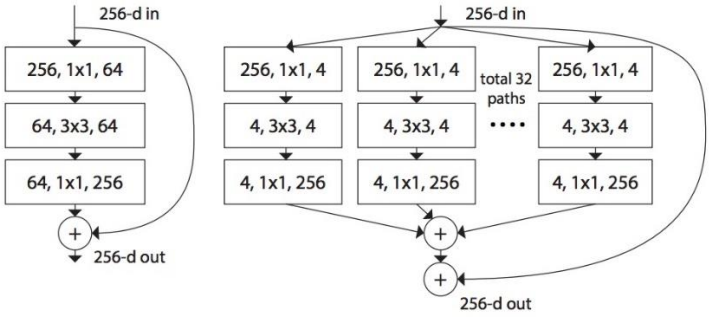
Single residual block：



之后主要是两个流派：Inception 和 ResNet 各自的改进

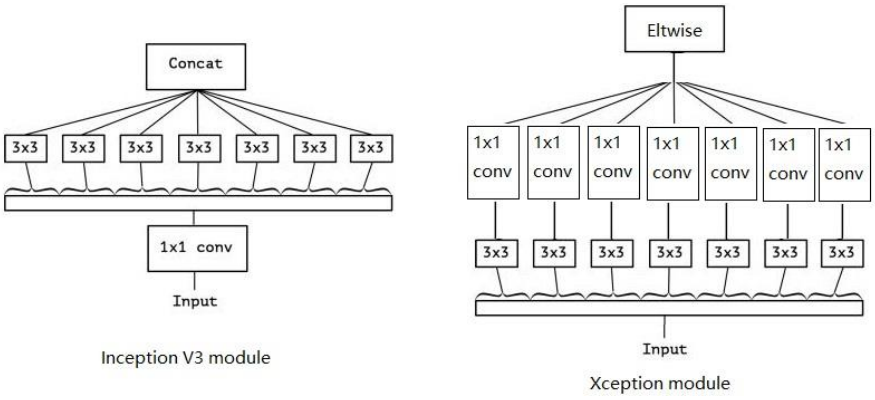
5.DenseNet：主要就是将 residual connection 发挥到极致，**每一层输出都直连到后面所有的层，可以更好的复用特征**，每一层都比较浅。缺点是显存占用大，而且反向传播复杂。

6.ResNeXt：ResNet 借鉴 Inception 得到的，主要就是将 ResNet 的基本单元，横向扩展，**在通道上对输入进行拆分**，进行分组卷积，每个卷积核不用扩展到所有通道，可以得到更多更轻量的卷积核，并且，卷积核之间减少了耦合，用相同的计算量，得到更高的精度。总体就是 **Split-transform-merge** 模式的一个实现。



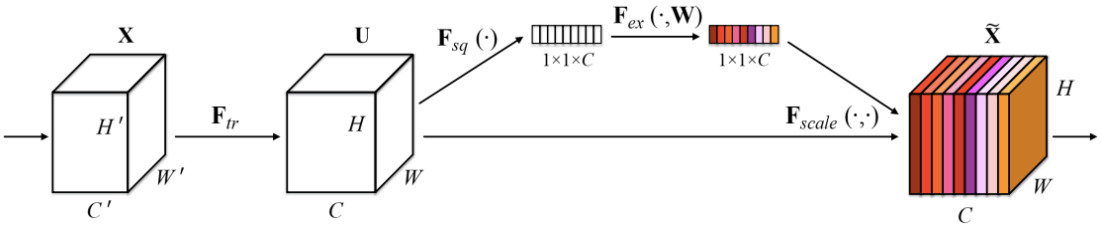
7.Xception:基于一个假设，就是水平和竖直方向的**空间卷积**（比如第一步的 3x3 卷积）和深度方向的**通道卷积**（比如第二步的 1x1 卷积）**可以完全独立进行**，这样减少了不同操作间的耦合，可以有效利用计算力。它把分组卷积的思想发挥到了极致，**每一个通道单独分为一组。**利用了 **depth-wise separable convolution。**J 个输入通道，每个通道用一个单独的空间卷积核卷积（比如 3x3），J 个卷积核得到 J 个输出通道，然后再用 K 个卷积核对上一步得到的 J 个输出通道进行 1x1 的普通卷积，得到 K 个最终的输出。

Inception V3 是先做 1*1 的卷积，再做 3*3 的卷积，这样就先将通道进行了合并，即**先通道卷积，然后再进行空间卷积**，而 **Xception 则正好相反，先进行空间的 3*3 卷积，再进行通道的 1*1 卷积。**



Inception V3 module　　Xception module

8.SENet: SENet 主要是提出了一个 Squeeze-Excitation 模块。在普通的卷积（单层卷积或复合卷积）由输入 X 得到输出 U 以后，对 U 的**每个通道进行全局平均池化得到通道描述子（Squeeze），再利用两层 FC 得到每个通道的权重值，对 U 按通道进行重新加权得到最终输出（Excitation）**，这个过程称之为 feature recalibration，**通过引入 attention 对通道重新加权(channel attention)**以抑制无效特征，提升有效特征的权重，并很容易地和现有网络结合，提升现有网络性能，而计算量不会增加太多。



7.轻量级的模型

1.SqueezeNet: 主要创新点是提出了 **fire module**，它包括 **squeeze 层和 expand 层。**

Squeeze 层采用 **1*1 卷积核**对上一层的 feature map 进行卷积，主要目的是**减少 feature map 的维数(通道数)。**Expand 层**分别用 1*1 和 3*3 卷积，然后 concat**，这个操作在 inception 系列里面也有。

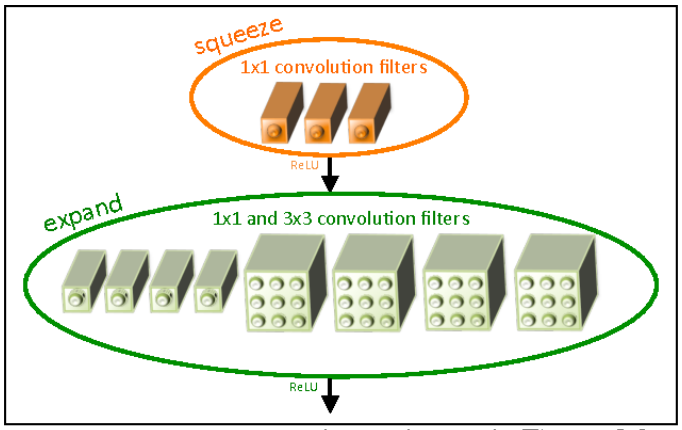


Figure 1. Organization of convolution filters in the **Fire module**. In

2.MobileNet: 创新点主要是名为 **depth-wise separable convolution** 的卷积方式代替传统卷积方式，以**达到减少网络权值参数的目的。**

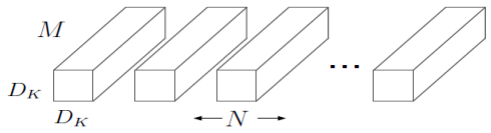
depth-wise convolution 和 **group convolution** 是类似的，depth-wise convolution 是**一个卷积核负责一部分 feature map，每个 feature map 只被一个卷积核卷积**；group convolution 是**一组卷积核负责一组 feature map，每组 feature map 只被一组卷积核卷积。Depth-wise convolution 可以看成是特殊的 group convolution**，即每一个通道是一组。采用 depth-wise separable convolution，会涉及两个超参：**Width**

Multiplier 和 Resolution Multiplier 这两个超参只是方便于设置要网络要设计为多小，方便于量化模型大小。

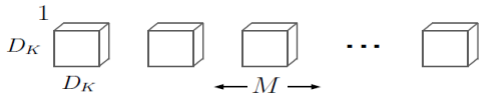
MobileNet 将标准卷积分成两步：

1. **Depth-wise convolution**，即**逐通道的卷积**，一个卷积核负责一个通道，一个通道只被一个卷积核「滤波」

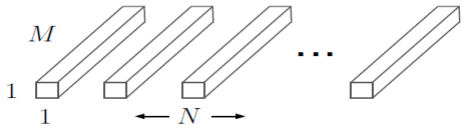
2. **Point-wise convolution，**将 depth-wise convolution 得到的 feature map 再「串」起来，使得**输出的每一个 feature map 要包含输入层所有 feature map 的信息。**



(a) Standard Convolution Filters



(b) Depthwise Convolutional Filters



(c) 1×1 Convolutional Filters called Pointwise Convolution in the context of Depthwise Separable Convolution

Figure 2. The standard convolutional filters in (a) are replaced by two layers: depthwise convolution in (b) and pointwise convolution in (c) to build a depthwise separable filter.

3.ShuffleNet: channel shuffle，是将各部分的 feature map 的 channel 进行有序的打乱，构成新的 feature map，以**解决 group convolution 带来的「信息流通不畅」问题，**而 MobileNet 是通过 point-wise convolution 来解决这个问题的。所以，该网络使用的前提是使用了 **group convolution。**

8. 各种卷积

1.卷积和互相关：

2.3D 卷积，转置卷积，扩张卷积，平展卷积

3.1*1 卷积：**出发点是想加宽加深网络**，作用有**降维/升维和加入非线性。**

4.空间可分卷积，通道可分卷积，分组卷积，混洗分组卷积，逐点分组卷积：**主要是整合 feature map 不同通道上的信息。**

9. pooling 的参数个数和操作后的 feature map 大小

1.max pooling 没有参数。普通卷积层参数个数：**卷积核的宽*卷积核的长*输入的通道数*卷积核的个数+卷积核个数。因为：y=weight * x + bias。**

2.Feature map 高度：output_h=（original_h+padding*2-kernel_h）/stride +1

10. 手推树，写出树的信息熵

构造树的基本想法是随着树深度的增加，**节点的熵迅速地降低。**熵降低的速度越快越好，这样我们有望得到一棵高度最矮的决策树。

信息：$I = -\log_2 p(x_i)$

信息熵 $H = -\sum_{i=1}^{n} p(x_i)\log_2 p(x_i)$

信息增益：分类前的信息熵减去分类后的信息熵

11.BatchNormalization

批规范化，即在每次 SGD 时，通过 mini-batch 来**对相应的 activation 做规范化操作**，使得结果**（输出信号各个维度）的均值为 0，方差为 1。**保证整个 network 的 capacity。

BN 本质上是为了解决反向传播的梯度问题，防止梯度弥散。此外，从高层来看统计机器学习中的一个经典假设是“源空间（source domain）和目标空间（target domain）的数据分布（distribution）是一致的”。如果不一致，那么就出现了新的机器学习问题，如，transfer learning/domain adaptation 等。虽然均值方差一致的分布不一定是同样的分布，但是 BN 起码保证了一定程度的一致性。

13.PCA 的推导, SVD

PCA 是一个降维的方法，其基本出发点是**方差越多的方向保留了越多的信息**，从而只保留方差比较大的方向，实现降维。

现在先来分析一下输入 x 的协方差矩阵。令 $C=E((x-\mu)(x-\mu)^T)$ ，则可以得到 C 是一个 $D\times D$ 的对称矩阵， (i,j) 项和 (j,i) 项相等，表示的都是 x 的第 i 个特征和 x 的第 j 个特征之间的协方差，而对角线上的 (i,i) 项表示的是 x 的第 i 个特征的方差。设 h 的协方差矩阵为 H ,则可以得到: $H=E((h-Eh)(h-Eh)^T)=E(hh^T)$

$=E(W^T(x-\mu)(x-\mu)^TW)=W^TE((x-\mu)(x-\mu)^T)W=W^TCW$ 。

刚才提到PCA的目标是使得组成 h 的 d 个随机分量两两之间的协方差为0，而且各自方差尽可能大，那么就要求 h 的协方差阵 H 是一个对角矩阵，而且对角线上的值要尽量大。所以，目标就转化为寻找 W ，使得 W^TCW 为对角阵， 并且对角线上元素按从大到小的顺序排列，这样用 W^T 左乘 x ,得到的就是在给定 d 维上保留 x 信息最多的 h 。

由于 C 为实对称矩阵，而根据实对称矩阵对角化定理，必然存在 D 个单位正交的特征向量，这 D 个向量按列组成正交矩阵 P ，使得 $P^{-1}CP=diag(\lambda_1,\lambda_2,\lambda_3\dots\lambda_D)$ ，其中， x 在某个特征向量的方向上的方差正好对应了决定该特征向量的特征值。注意要调整这 D 个单位正交的特征向量的顺序，使得对应的 D 个特征值 λ 按从大到小的顺序排列，其中可能包含多重的特征值，但是没有任何影响，因为重数为 k 的特征值对应了 k 个线性无关的特征向量。由于 P 为单位正交阵，有 $P^T=P^{-1}$ ，所以取 P 的前 d 列组成 W ，得到的 h 刚好为一个对角矩阵，且对角线上的元素刚好为 C 中最大的 d 个特征值。

SVD：奇异值分解（singular value decomposition）

假设M是一个$m\times n$阶矩阵，其中的元素全部属于域K，也就是实数域或复数域。如此则存在一个分解使得

$$M = U\Sigma V^*,$$

其中U是$m\times m$阶酉矩阵；Σ是$m\times n$阶非负实数对角矩阵；而V^*，即V的共轭转置，是$n\times n$阶酉矩阵。这样的分解就称作M的**奇异值分解**。Σ对角线上的元素Σ_{i}，即为M的**奇异值**。

常见的做法是将奇异值由大而小排列。如此Σ便能由M唯一确定了。（虽然U和V仍然不能确定。）

14.面试经历

基础复习：信息论、最优化、深度学习、机器学习

20190306 网易 技术中心 AI Lab 计算机视觉 暑期实习面试

1.项目/论文

2.Feature map 计算

3.参数量计算

4.各种激活函数

5.SVM，LR 等传统机器学习模型

6.Mobilenet 等轻量网络

7.各种 CNN 网络

8.各种卷积

二、CV 相关题目

1.逻辑回归和线性回归

逻辑回归假设因变量 y 服从伯努利分布，而线性回归假设因变量 y 服从高斯分布。因此与线性回归有很多相同之处，去除 Sigmoid 映射函数的话，逻辑回归算法就是一个线性回归。

可以说，逻辑回归是以线性回归为理论支持的，但是逻辑回归通过 Sigmoid 函数引入了非线性因素，因此可以轻松处理 0/1 分类问题。

2.K-means

已知初始的 k 个均值点，，按照下面两个步骤交替进行：

1.分配：将每个观测类分配到聚类种，使得组内平方和最小

2.更新：于上一步得到的每一个聚类，以聚类中观测值的图心，作为新的均值点。

K-means 是一个 NP-hard 问题，不过存在高效的启发式算法。K-means 重要的是初始化。

P Problem：对于任意的输入规模 n，问题都可以在 n 的多项式时间内得到解决；

NP(Non-deterministic Polynomial) Problem：在多项式的时间里**验证**一个解的问题；

NPC(Non-deterministic Polynomial Complete) Problem：满足两个条件：

(1)是一个 NP 问题 (2)所有的 NP 问题都可以约化到它

NP-Hard Problem：满足 NPC 问题的第二条，但不一定要满足第一条

3.如果训练集的正确率一直上不去，可能存在哪些问题

1.数据集的问题，标签、数据是否正常；2，参数，超参数的设置，学习率之类的；3.模型

4.svm，为什么要求对偶，核技巧

SVM 模型是将实例表示为空间中的点，这样映射就使得单独类别的实例被尽可能宽的明显的间隔分开。SVM 还可以使用所谓的**核技巧**有效地**进行非线性分类**，**将其输入隐式映射到高维特征空间中。**

核技巧 [编辑]

假设我们要学习与变换后数据点 $\varphi(\vec{x}_i)$ 的线性分类规则对应的非线性分类规则。此外，我们有一个满足 $k(\vec{x}_i, \vec{x}_j) = \varphi(\vec{x}_i) \cdot \varphi(\vec{x}_j)$ 的核函数 k。

5.反向传播算法(BP)，以及常见的目标函数，激活函数和优化算法

反向传播（英语：Backpropagation，缩写为 BP）是“**误差反向传播**”的简称，是一种与最优化方法（如梯度下降法）结合使用的，用来训练人工神经网络的常见方法。该方法**对网络中所有权重计算损失函数的梯度。**这个**梯度会反馈给最优化方法，用来更新权值**以最小化损失函数。

$$\frac{\partial E}{\partial o_j} = \sum_{l \in L} \left(\frac{\partial E}{\partial \mathrm{net}_l} \frac{\partial \mathrm{net}_l}{\partial o_j} \right) = \sum_{l \in L} \left(\frac{\partial E}{\partial o_l} \frac{\partial o_l}{\partial \mathrm{net}_l} \frac{\partial \mathrm{net}_l}{\partial o_j} \right) = \sum_{l \in L} \left(\frac{\partial E}{\partial o_l} \frac{\partial o_l}{\partial \mathrm{net}_l} w_{jl} \right)$$

Sigmoid：输入实数，输出 0 到 1。问题：1 函数饱和使梯度消失，2.输出非负，**非零中心。**

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

Tanh: 输出到[-1,1 之间]。问题：存在**饱和问题**

$$tanh(x) = 2\sigma(2x) - 1$$

ReLU: 运算量小，相对宽阔的兴奋边界，稀疏激活性（起到对网络正则化的效果）。

缺点：比较脆弱，容易死掉，且不可逆。会导致数据多样性的丢失。

$$f(x) = max(0, x)$$

Leaky ReLU：解决 ReLU 死掉的问题。

$$f(y) = max(\varepsilon y, y)$$

Maxout：是对 ReLU 和 leaky ReLU 的一般化归纳。优点：计算简单，不会饱和，不容易死掉，缺点：每个神经元的参数 double。

$$f(x) = max(w_1^T x + b_1, w_2^T x + b_2)$$

Softmax: 用于多分类网络输出，**目的是让大的更大。**Softmax 是 Sigmoid 的扩展，当类别数 k = 2 时，Softmax 回归退化为 Logistic 回归。

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}$$

6.如何避免陷入鞍点

1.引入随机元素如噪音来避免过拟合，2.CNN 的参数压缩，3.遗传算法等，

7.pytorch 框架说明，和 tensorflow 等其他框架对比

PyTorch 中图结构是动态的，即图是运行时创建的，而 TensorFlow 中，图结构是静态的，即图是“编译”之后再运行的。此外 PyTorch 自动求导机制。PyTorch 更像一个框架，而 TensorFlow 更偏底层，像个库，自定义程度更高。

8.如何处理深度学习 overfitting

1.获取更多数据，数据增强

2.换一个模型，减少网络的参数量，限制网络的拟合能力，主要有下面几条：

减少网络层数和神经元个数等，训练时间 early stopping,

限制和减小权值（正则化）， 增加噪声

3.训练多种模型，取所有模型的平均输出为结果（集成学习：Bagging, Boosting, Stacking）

4. Dropout，每次随机（以一定概率）忽略隐层的某些节点

5.贝叶斯方法

9.如何在测试中，加速 1000 倍(不改变网络结构)

网络剪枝，参数共享，低秩估计(用若干小矩阵对参数矩阵进行估计)，矩阵合并分解之类的都会改变网络。测试的时候不改变网络结构加速有两种：（**并行**也算的话是三种）

1. **模型量化**：用有限的几个参数来估计连续的实数域，即量化成 8bit 整数计算之类的，即模型的定点化，比浮点数快。还有就是对输入数据做处理，减小数据量、计算量。
2. **光衍射神经网络**：训练得到的不是神经元的权重，而是神经元的透光/反射系数，然后用 3D 打印出来这些衍射层，残差可以用光分器，还可以考虑偏正片之类的。即**光做信息流，中间材料做存储器。**问题是一旦打印就定型了，测试的话倒是确实可以加速。

10.深度学习和传统机器学习对比

天下没有免费的午餐，数据对于深度学习来说更为重要，是驱动力之一，此外，训练也非常耗时，不过效果比传统的机器学习要好得多。

传统的机器学习方法有：**决策树、SVM、逻辑回归、贝叶斯、集成学习、聚类和降维。**

11.深度学习以后的发展，目前的缺陷

发展：GAN(生成对抗网络)，迁移学习，AutoML，压缩网络，移动化之类的

缺陷：没有理论性的框架，复杂性堆叠带来质变但是不清楚质变的根源，还有就是对抗样本说明深度网络还很脆弱。语义鸿沟问题没有办法解决，也没有相当靠谱的解释。

12. 深度学习正则化有哪些

1.数据增强，对原始图像数据做各种各样的处理，色彩、扰动、裁剪、旋转之类的。

2.L1 和 L2 正则化，限制权值。L1 正则化中很多参数向量是稀疏向量，因为很多模型导致参数趋于 0，因此常用于特征选择设置中。L2 一般用来限制权重。

3.Dropout　4.early stoping

13. 过拟合欠拟合以及其背后本质，从偏差方差角度如何理解

过拟合欠拟合本质是神经网络学习语义空间上的一个分布，因为数据集不是完完全全的，所以有偏差，有可能学的过分，参数数量过多，容量过大，完全映射训练集中所有数据，甚至包括噪声，形成了偏见，所以就过拟合。过拟合则相反，模型过于简单，没有学到复杂数据集上的一个合适的映射。

欠拟合会导致高 Bias ，过拟合会导致高 Variance。Bias 即为**模型的期望输出与其真实输出之间的差异**， 而 Variance 刻画了**不同训练集**得到的**模型的输出与这些模型期望输出的差异。**

(1)**解决欠拟合**的方法：

1、**增加新特征**，可以考虑加入进特征组合、高次特征，来增大假设空间;

2、**尝试非线性模型**，比如核 SVM 、决策树、DNN 等模型;

3、如果有正则项可以**减小正则项参数**;

4、**Boosting ,Boosting 往往会有较小的 Bias**，比如 Gradient Boosting 等.

(2)**解决过拟合**的方法：

1、**交叉检验**，通过交叉检验得到较优的模型参数;

2、**特征选择，减少特征数或使用较少的特征组合**，对于按区间离散化的特征，增大划分的区间;

3、**正则化**，常用的有 L1、L2 正则。而且 L1 正则还可以自动进行特征选择;

4、如果有正则项则可以考虑**增大正则项参数**;

5、**增加训练数据**可以有限的避免过拟合;

6、**Bagging ,将多个弱学习器 Bagging** 一下效果会好很多，bagging 有较小的 Variance，比如随机森林等.

14.模型融合策略和方法

1.voting,包括加权投票　2.平均　3.stacking,集成学习之类的

15.优化算法：sgd　adam 牛顿法等区别。为什么牛顿法快，牛顿法的缺点

Sgd，adam 都属于梯度下降法，只不过在算法的步长选择，算法参数的初始值，归一化等方面有区别，对于非凸的神经网络来说，都是由可能达到局部最优解。

牛顿法与梯度下降法相比，都是迭代求解，不过**梯度下降法是梯度求解**，而**牛顿法是用二阶的海森矩阵的逆矩阵求解。**相对而言，使用**牛顿法收敛更快（迭代更少次数）。**但是**每次迭代的时间比梯度下降法长。**

牛顿法缺点：1. **局部收敛，初始点不恰当的话，往往不收敛。**

2. 当**二阶海塞矩阵非正定时**，**不能保证产生方向是下降方向。**

3. **运算量大，要求严苛**，二阶海塞矩阵必须可逆。

16. 如何解决训练类别数据不平衡

1.欠采样，删除一些数据

2.过采样，拷贝现有样本随机增加观测数量

3.合成采样，使用最近邻方法通过相似性合成

4.数据增强，对原始数据做变换

5.集成学习

17. Precision Recall ROC PR 等概念

- TP: 将正类预测为正类数 40
- FN: 将正类预测为负类数 20
- FP: 将负类预测为正类数 10
- TN: 将负类预测为负类数 30

准确率(accuracy) = 预测对的/所有 = (TP+TN)/(TP+FN+FP+TN) = 70%
精确率(precision) = TP/(TP+FP) = 80%
召回率(recall) = TP/(TP+FN) = 2/3

ROC 曲线：横坐标为 false positive rate，纵坐标为 true positive rate，越接近左上角越好

AUC：ROC 曲线下面的面积，越接近 1 越好

PR 曲线：横坐标为 precision，纵坐标为 recall，越接近右上角越好

18. 噪声项目

项目：有大量数据，数据处理后有一定比例的噪声数据，项目研究如何在含有噪声数据的情况下更好地进行深度学习网络的构建和训练，优化分类网络的准确率。

考虑：1.从数学上和信息论上谈处理，噪声一般符合高斯分布，均值为 0，无用信息。

2.自己的项目中有哪些处理噪声数据的部分

3.加性噪声还是乘性噪声？噪声的类型，如何分离

4.噪声是属于高频，从信号处理上可以考虑滤波器，傅里叶变换，小波变换

5.从物理上考虑，频率，振幅之类的

6.自编码器，编码器将输入压缩到潜在语义空间表示，解码器重构来自潜在语义空间表示的输入。需要加一些限制。考虑卷积自编码器，稀疏自编码器，降噪自编码器。结合信息瓶颈理论谈一谈。

19.RandomForest GBDT Xgboost lightgbm 区别

RandomForest: Bagging 可以简单的理解为：放回抽样，多数表决（分类）或简单平均（回归）,同时 Bagging 的基学习器之间属于并列生成，不存在强依赖关系。Random Forest（随机森林）是 Bagging 的扩展变体，它在以**决策树**为基学习器**构建 Bagging 集成**的基础上，**进一步在决策树的训练过程中引入了随机特征选择**，因此可以概括 RF 包括四个部分：**1、随机选择样本（放回抽样）**；**2、随机选择特征属性**；**3、构建决策树**；**4、随机森林投票（平均）。**因此防止过拟合能力更强，**降低方差。**

GBDT (Gradient Boosting Decision Tree)： Boosting 是一种与 Bagging 很类似的技术。不论是 Boosting 还是 Bagging，所使用的多个分类器类型都是一致的。但是在前者当中，**不同的分类器是通过串行训练而获得的**，每个新分类器都根据已训练的分类器的性能来进行训练。**Boosting 是通过关注被已有分类器错分的那些数据来获得新的分类器。**由于 Boosting 分类的结果是基于所有分类器的加权求和结果的，因此 Boosting 与 Bagging 不太一样，**Bagging 中的分类器权值是一样的，而 Boosting 中的分类器权重并不相等，每个权重代表对应的分类器在上一轮迭代中的成功度。**GBDT 的基本原理是 boost 里面的 boosting tree（提升树），并使用 gradient boost。GBDT 与传统的 Boosting 区别较大，它的每一次计算都是为了减少上一次的残差，而为了消除残差，我们可以在残差减小的梯度方向上建立模型,所以说，在 **GradientBoost 中，每个新的模型的建立是为了使得之前**

的模型的残差往梯度下降的方法，与传统的 Boosting 中关注正确错误的样本加权有着很大的区别。

XGBoost：是集成学习 Boosting 家族的成员，是**在 GBDT 的基础上对 boosting 算法进行的改进。**GBDT 是用模型在数据上的负梯度作为残差的近似值，从而拟合残差。**XGBoost 也是拟合的在数据上的残差，但是它是用泰勒展式对模型损失残差的近似**；同时 XGBoost 对模型的损失函数进行的改进，并加入了模型复杂度的正则项。

lightGBM：是微软出的新的 boosting 框架，基本原理与 XGBoost 一样，只是在框架上做了一优化，重点在**模型的训练速度的优化。**与 XGboost 对比：

1.xgboost 采用的是 level-wise 的分裂策略，而 lightGBM 采用了 **leaf-wise** 的策略。为了防止过拟合，lightGBM 对最大深度进行了限制。

2. lightgbm 使用了**基于 histogram 的决策树算法，**在内存和计算代价上都有不小优势。

3.lightgbm 有多线程优化和并行优化

20.backward 推导，conv、pooling 等的 backward 如何计算

DNN：$\frac{\partial J}{\partial z^{l-1}} = \frac{\partial J}{\partial z^{l}}\frac{\partial z^{l}}{\partial z^{l-1}} = (W^{l})^{T}\frac{\partial J}{\partial z^{l}}\sigma^{'}(z^{l-1})$

池化层：$\frac{\partial J}{\partial z^{l-1}} = upsample(\frac{\partial J}{\partial z^{l}}).\sigma^{'}(z^{l-1})$ **，池化层需要做一个上采样的过程**

卷积层：$\frac{\partial J}{\partial z^{l-1}} = \frac{\partial J}{\partial z^{l}} * rot180(W^{l})\sigma^{'}(z^{l-1})$，卷积层输出的误差和旋转卷积核的卷积，然后和激活函数的导相乘，和DNN唯一的区别就是W。

3、最后我们求出了每一层的$\frac{\partial J}{\partial z^{l}}$，**然后就是求W和b的导数的过程。DNN可以直接根据z、w、b的相乘公式求解出来；池化层没有参数，所以不必要求解；CNN的卷积层包含参数需要求解。**

DNN：$\frac{\partial J}{\partial W^{l}} = \frac{\partial J}{\partial z^{l}}(a^{l-1})^{T}$　　$\frac{\partial J}{\partial b^{l}} = \frac{\partial J}{\partial z^{l}}$

卷积层：$\frac{\partial J}{\partial w^{l}} = a^{l-1} * \frac{\partial J}{\partial z^{l}}$　　$\frac{\partial J}{\partial b^{l}} = \sum_{u,v}(\frac{\partial J}{\partial z^{l}})_{u,v}$

21. R-CNN、Fast R-CNN、Faster R-CNN、Mask R-CNN

R-CNN:

1. **找候选框：**文中利用传统方法（ selective search）找出2000个可能是物体的候选框，然后将这些框缩放到同一尺寸，便于输入CNN网络。
2. **CNN提取特征：**选用已有CNN架构（比如AlexNet）及其参数作为初始参数，然后再训练过程中微调参数。（若选取网络所有参数都随机初始化，由于训练样本少，会导致跑不到最理想的参数）
3. **训练一个SVM分类器：**计算所有区域的得分，使用非极大抑制选择最合适的候选框，然后使用回归器精修候选框的位置。

Fast R-CNN:

Fast R-CNN和R-CNN是同一个作者，是在R-CNN基础上做的改进，在Fast R-CNN中，作者指出了R-CNN的几个不足：

1. **训练是一个多阶段的过程：**文中训练需要经过调参，建立SVM，边界回归几个过程，使训练时间显得很长。
2. **占用很大的时间和计算机空间:**在训练SVM和边界回归过程中，从图片的候选框中得到的特征都要写入磁盘，非常占用内存和GPU。
3. **对象检测速度很慢：**在测试时，需要从每个测试图像候选框中提取特征。使用VGG16进行检测平均每幅图像需要47s。

做的改进： 相比 R-CNN，Fast R-CNN 将整幅图像作为输入，消除了候选框之间的重叠和冗余，**特征提取的更快。**将 **RoI feature vector 作为两个全链接层的输入**，不需要磁盘存储特征，同时也更快。

Faster R-CNN:

与Fast R-CNN的结构相比，Faster R-CNN把区域提案（选择候选框）放在了CNN网络之后，相当于和后面的图像检测网络共享了一个CNN网络，那么为什么要这样做？

原文作了如下解释：在Fast R-CNN中，选取候选框的工作在CPU中进行，然后再GPU中进行CNN提取特征，这会拖慢整体的运行速度，所以能够想到把区域提案工作也在GPU中完成，而把这一步骤放在CNN提特征之后，即能共享CNN提取的特征，也不会影响后续的检测工作，使得产生区域提案的开销达到最小。

到这，整个架构就很清楚了：**RPN+Fast R-CNN**
作者保留了Fast R-CNN的主体架构（提特征，检测，分类，回归），在CNN网络和检测网络间并入了RPN网络用来产生区域提案。

Mask R-CNN:

可以看出在原有的Faster R-CNN的架构上，多了一个分支，用于对每个ROI区域预测分割mask，其结构实质上是一个FCN。
所以我们可以近似的认为：Mask R-CNN=Faster R-CNN + FCN，其中，Faster R-CNN用于提取候选框，对候选框进行分类和回归，FCN则用于实例分割，也就是求掩膜mask。

总体而言，本文的贡献主要在于以下两点：

1. 在Fast R-CNN架构上增加FCN分支，实现了目标分割
2. 提出了一种ROIAlign，提升了算法在像素级分割时的精确性

R-CNN：选取候选框 → 提特征 → SVM分类器

Fast R-CNN： 输入是整幅图片+ROI Projection，减少了冗余

Faster R-CNN：等于 **Fast R-CNN+RPN**

Mask R-CNN：等于**Faster R-CNN + FCN**

三、其他题目

1.Python 多进程实现

1. os.fork()
2. 使用 multiprocessing 模块：创建 Process 的实例，传入任务执行函数作为参数
3. 使用 multiprocessing 模块：派生 Process 的子类，重写 run 方法
4. 使用进程池 Pool

2.MergeSort，quickSort

归并排序的思想是将数组分成两部分，分别进行排序，然后归并起来。

快速排序通过一个切分元素将数组分为两个子数组，左子数组小于等于切分元素，右子数组大于等于切分元素，将这两个子数组排序也就将整个数组排序了。

3.TopK 算法

1. 若数据量不太大，可以用常规的**排序**方式；
2. 若数据量太大，且中间数据无需保存，则可用**最小堆**解决；
3. 若数据量太大，且中间数据需要保存，则可用**外部排序**解决；