

优化 Optimisation. by Yann.

▷ 优化流程 Optimization Pipeline.

① 初始化参数 (因为每个神经元 Neuron 的参数都一样)
e.g. random initialisation / He initialisation.

② 选择更新算法 e.g. GD, Adam.

③ 前向输入 + 向后传播直引拟合.

Forward input \rightarrow loss \rightarrow gradient \rightarrow backpropagate.

▷ Gradient Descent.

① 定义: $J(\theta, x)$ 为 objective function 目标函数.

opt. ↗

优化为最小化 $J(\theta, x)$: $\min J(\theta, x)$.

by $\theta = \theta - \eta \nabla_{\theta} J(\theta, x)$ \rightarrow 梯度, gradient.
 \hookrightarrow learning rate.

② Mini-batch Gradient: $\theta = \theta - \eta \cdot \nabla_{\theta} J(\theta, x^{(t+1)})$

special {
1. 当 $x = x^{(1:\text{end})}$ 时, 使用所有的 batch 进行训练被称作 Batch GD.
2. 当 $x = x^{(i)}$ 时, 仅使用 1 个训练资料, 被称作 Stochastic GD.

③ GD 的限制性:

1. learning rate η 的选择困难 {
太大: Hard to converge.
太小: slow to converge.}

2. fixed learning rate, 学习率固定导致所有的参数更新一样.
 \hookrightarrow 希望对不同的参数进行不同程度的更新 (自适应).

e.g. High frequency, smaller learning rate.
Low ---, higher ---

3. 容易掉入 saddle point. \curvearrowleft \rightarrow 梯度为 0 无法 Backpropagate.
 \hookrightarrow How to escape.

解决方法: Momentum or NAG.

3) Momentum 动量更新.

① Motivation: Dampens oscillations through acceleration.
通过加上以前的梯度加速来减少 noise 带来的震荡.
和上一时刻梯度一样方向梯度的动量增加. \hookrightarrow 更快收敛.

$$\theta_t = \eta V_t + \gamma \nabla_{\theta} J(\theta) \\ \text{衰减动量系数 } \downarrow \quad \text{上一时刻动量.} \quad \rightarrow \text{梯度.}$$

4) Nesterov Accelerated Gradient (NAG).

① Motivation: 预测未来的梯度方向进行修正.

② 更新规则: 1. 先用之前的梯度向前更新.

2. 测量 end-up 的梯度.

3. 修正. 用之前的梯度更新. \hookrightarrow

$$\text{公式: } V_t = \eta V_{t-1} + \gamma \nabla_{\theta} J(\theta - \eta V_{t-1})$$

$$\theta = \theta - V_t.$$

④ Momentum.

NAG:



5) Adagrad.

① Motivation: 根据 past gradient 自适应计算 learning rate.

② 更新公式:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{G_{t+1}}} \nabla_{\theta} J(\theta_t)$$

$$\text{where } g_{t+1} = (\nabla_{\theta} J(\theta_t))_t, G_{t+1} = \sum_i^t g_{t,i}^2$$

③ 优点: 虽然梯度大, 更新步数小. 反之虽然梯度小, 更新大.

解决了稀疏数据的问题, 因为稀疏数据更新不频繁.

④ 缺点: 仍然需要手动设置 global learning rate η . (不尊问题)

⑤ $\sum_i g_{t,i}^2$ 是单调增函数.

所以 $\frac{\eta}{\sqrt{g_{t+1}^2 + \epsilon}}$ 最终将变的非常小.

\hookrightarrow 解决方法: 加入 window 机制.

6) Adam.

① Motivation: 解决了 adagrad 中 $\sum_i g_{t,i}^2$ 无限大的问题.

② Accumulate over window: 时间窗口. $t=2 \Rightarrow t=(t_1+t_2)$

③ Update rules: 处理的方法实际上也是用了动态衰减.

$$E[g_t^2] = \tau E[g_{t-1}^2] + (1-\tau) g_t^2$$

$$\text{RMS}[g_t] = \sqrt{E[g_t^2] + \epsilon}$$

$$\Delta \theta_t = \frac{\eta}{\text{RMS}[g_t]} g_t \Rightarrow \text{Also known as RMSprop.}$$

④ Hessian correction: 一阶导不能提供上升/下降的信息.

但是 Hessian 利用二阶导数描述曲率信息优化.

g. 低曲率 \rightarrow 平坦 \rightarrow 大的学习率.

高曲率 \rightarrow 陡峭 \rightarrow 小的学习率.

$$H^{-1} = \Delta \theta = (\partial J / \partial \theta) / (\partial^2 J / \partial \theta^2)$$

$$\Delta \theta = \frac{\text{RMS}[\Delta \theta_{t-1}]}{\text{RMS}[g_t]} g_t.$$

7) Adam. (Momentum + RMSprop)

① Motivation: 同时保留过去梯度, 和动量信息.

$$\left\{ \begin{array}{l} \text{过去的梯度. } V_t = \tau V_{t-1} + (1-\tau) g_t \\ \text{过去的动量. } M_t = \tau M_{t-1} + (1-\tau) g_t \end{array} \right.$$

② Update rules: $\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{M_t}} \frac{g_t}{V_t}$ \rightarrow 不是 g_t .

③ Bias correction: 由于初始的 V_t 和 M_t 都小, 接近 0.
或者后期的 decay rate 接近 1.

$$\left\{ \begin{array}{l} \hat{V}_t = V_t / (1 - \beta_1^t) \quad \text{前期: } mV+b, 1-rb \\ \hat{M}_t = M_t / (1 - \beta_2^t) \quad \text{后期: } mV+b, 1-rb \end{array} \right.$$

Reference: COMP5329 Deep Learning. Dr Chang Xu
University of Sydney. 悉尼大学.

参数初始化 Initialisation.

- 1) 初始化的宗旨：快速收敛
 非对称性：所有的神经元 Neurons 输出相同.
 \Rightarrow 相同的梯度 \Rightarrow 相同的参数.
- 不能太大：梯度爆炸.
- 不能太小：梯度消失.
- Mean 应该为 0. 每层的 Variance 应该一样.

2) Xavier Initialisation.

- ① Assumption: 激活函数为 tanh,
 并且在训练初期在 linear regime.
 所以 $\tanh(z^l) \approx z^l$ ①
- $$z^l = w^l a^{l-1} + b^l \quad \text{②}$$
- 通过①式和②式, $a^l = \tanh(z^l) \approx z^l$.
- 所以 $\text{Var}(a^l) = \text{Var}(z^l)$,
- $$= \text{Var}\left(\sum_{j=1}^n w_j^l a_j^{l-1}\right)$$
- $$= \sum_{j=1}^n \text{Var}(w_j^l a_j^{l-1}) \quad \text{--- ③}$$

Variance 公式: $\text{Var}(XY) = \mathbb{E}[X]\text{Var}(Y) + \text{Var}(X)\mathbb{E}[Y]^2 + \text{Var}(X)\text{Var}(Y)$
 将 Variance 公式带入③化简. $\rightarrow w_j^l \rightarrow g_j^{l-1}$ (final term).

因为 assume 权重的 mean 是 0, 输出是正态分布.

$$\Rightarrow \mathbb{E}(x) = 0, \text{Var}(x) = 1$$

$$\Rightarrow \text{Var}(w_j^l a_j^{l-1}) = \text{Var}(w_j^l) \text{Var}(a_j^{l-1}) \quad \text{--- ④}$$

$$\Rightarrow \text{Var}(a^l) = \sum_j \text{Var}(w_j^l a_j^{l-1}) = n \text{Var}(w^l) \text{Var}(a^{l-1}) \quad \text{--- ⑤}$$

$$\Rightarrow \text{Var}(w^l) = \frac{1}{n} \quad \text{backward: } \text{Var}(w^l) = \frac{1}{n}$$

② 初始话公式: let $n = n^{l-1} + n^l$

Xavier normal: $N(0, 2/n)$

Xavier uniform: $U(-\sqrt{6}/n, \sqrt{6}/n)$

3) He initialisation. 此时的激活函数为 ReLU

- ① 推导. $\text{Var}(y) = \text{Var}(\sum_i w_i x_i)$
- $$= n \text{Var}(w) \mathbb{E}[x_i^2] \quad \text{by } \text{Var}(x) = \mathbb{E}[x^2] - \mathbb{E}[x]^2$$
- 因为 ReLU: $x_i^{e_1} = \max(0, y^{e_1})$
- $$\Rightarrow \mathbb{E}[x_i^2] = \frac{1}{2} \text{Var}(y^{e_1})$$
- $$\Rightarrow \text{Var}(y) = n \text{Var}(w) \cdot \frac{1}{2} \text{Var}(y^{e_1})$$
- $$\Rightarrow \text{Var}(w) = 2/n.$$

② 公式:

He normal: $N(0, 2/n)$

He uniform: $U(-\sqrt{6}/n, \sqrt{6}/n)$

正则化 Regularization.

1) 正则化: 为了避免过拟合.

- ① 影响因素. ① 参数过多 ② 权重过大.
- Dropout. L_p 正则化.

② 硬约束 vs 软约束.

g. Hard L_2 constraint:

$$\min_{\theta} \hat{L}(\theta) = \frac{1}{n} \sum_{i=1}^n l(\theta) \text{ s.t. } \|\theta\|_2^2 \leq r.$$

g. Soft L_2 constraint: \uparrow trade-off parameters.

$$\min_{\theta} \hat{L}(\theta) = \frac{1}{n} \sum_{i=1}^n l(\theta) + \lambda \|\theta\|_2^2$$

③ Bayesian Regularisation. 正则化对应先验信息.

Bayesian rule: $P(\theta|x) = \frac{P(x|\theta)P(\theta)}{P(x)}$

Maximum A Posteriori (MAP): 带着 prior

$$\max_{\theta} \log(P(\theta|x)) = \max_{\theta} \underbrace{\log(P(x|\theta))}_{\text{MLE loss}} + \underbrace{\log(P(\theta))}_{\text{Regularisation.}}$$

2) Weight Decay. (另一种 Lp 正则化解释)

① Motivation: 鼓励模型学习更小的参数.

$$\min_{\theta} \hat{L}(\theta) = \hat{L}(\theta) + \frac{\lambda}{2} \|\theta\|_2^2 \quad \text{Engineer's view:}$$

$$\nabla \hat{L}(\theta) = \nabla \hat{L}(\theta) + \lambda \|\theta\|_2^2 \quad \text{表示常数 } \lambda \text{ p.}$$

\uparrow trade-off parameter

② 数据集加入 Noise 相当于加入正则化.

$$\begin{aligned} g. f(x) &= w^T x, \quad \varepsilon \sim N(0, \lambda I). \\ \hat{L}(f) &= \mathbb{E}_{xy\varepsilon} [f(x+\varepsilon) - y]^2 = \mathbb{E}_{xy\varepsilon} [f(x) + w^T \varepsilon - y]^2 \\ &= \mathbb{E}_{xy\varepsilon} [f(x) - y]^2 + 2 \mathbb{E}_{xy\varepsilon} [w^T \varepsilon - y] + \mathbb{E}_{\varepsilon} [w^T \varepsilon]^2 \\ &= \mathbb{E}_{xy\varepsilon} [f(x) - y]^2 + \lambda \|w\|^2 \end{aligned}$$

3) Dropout

\rightarrow 参数的稀疏性.

① Motivation: 希望只用少量的参数就能进行预测.

② 实现: 训练阶段 P% 的神经元被遮住.

测试阶段 使用所有神经元. weight scale down P%.

g. Inverted Dropout. { At training: weight = 1/p weight.

At testing: use the whole network.

相当于训练了 2^n 个网络. 测试时平均了所有结果.

③ 优势: ① scale-free 对大规模 feature 不会有影响.

② Invariant to parameter scaling

④ DropConnect. 相当于随机丢弃神经元, 随机丢弃连接.

\hookrightarrow 更丰富的网络组合.

4) Batch Normalisation. 减少 feature unit 带来的影响. \rightarrow 更快的学习率.

① Motivation: 减少 internal covariance shift, 加速收敛 \downarrow gradient vanishing.

② Group Normalisations. BN 在 batch size 上表现不好.

将 channel 分成 group 做归一化. 对 batch size 不敏感.

Reference: COMP5329 Deep Learning. Dr Chang Xu
 University of Sydney. 悉尼大学.

卷积神经网络 CNNs.

① Convolutional Neural Network. by Yann.

② Motivation: 抓取邻近特征，减少参数量。

③ 组成部分: 卷积层, 池化层, 全连接层.
提取特征 \downarrow 压缩特征 \downarrow 分类层.

④ 卷积层. Convolutional Layers.

$$g \cdot f = \sum_i^n f_i \cdot g_i$$

图片: $\begin{bmatrix} 1 & 2 & 0 \\ 2 & 1 & 1 \\ 1 & 0 & 0 \end{bmatrix}$ $\xrightarrow{\text{filter}} \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$ 轮廓: $\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$

卷积计算: $\begin{cases} (1 \cdot 1 + 2 \cdot 1 + 0 \cdot 0) = 1 \\ (2 \cdot 1 + 1 \cdot 0 + 1 \cdot 0) = -2 \\ (1 \cdot 0 + 0 \cdot 0 + 0 \cdot 1) = 0 \end{cases} \quad \left. \begin{array}{l} 1-2+0=-1 \\ 1-2+0=-1 \end{array} \right\} 1-2+0=-1$

rules: $f \ast g = \sum_i^n f_i \cdot g_i$

Stride 步长: 卷积核扫描的长度。

Padding 填充: 将边缘像素用0补齐。

Output size: 图片经过卷积后行和列。

$$\text{Input size} \xleftarrow[\text{stride}]{\text{padding}} \text{Kernel size}$$

$$\text{Size} = \frac{N+2P-K}{S} + 1$$

⑤ Receptive Field 感受野: 一个单位(神经元)对空间的感受野范围。

\hookrightarrow 随着网络的增加, 一个像素包含了前面层数的信息。

g. 3x3的kernel. 一个像素等于前面的9个像素。

⑥ Dilated Convolution 扩张卷积, 在卷积核上引入扩张率。

g. $D=1$ $D=2$ \rightarrow 在kernel中插入“空洞”。
因此能用更快增加感受野。
但是牺牲了精细度。

⑦ Pooling 池化层: 压缩特征 + 平移不变性。

种类: 平均池化, 最大池化, L池化。

网络结构 NN Architectures.

① 卷积神经网络总览.

向后传播.	正则化	参数增强.	更多 filters.	3x3 kernel size.
MNIST	AlexNet	5层卷积	ZFNet	VGG 16层
LeNet				Inception module. Global pooling. GoogleNet 22层
2008	2012	2013	2014	2015
			101层 残差连接.	ResNet

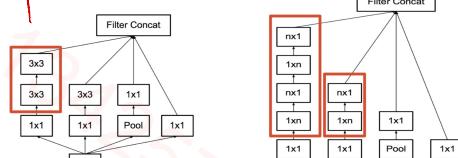
② Inception Module: 学习不同尺度的信息。

① Motivation: 网络的每层都用不同尺度的卷积核(kernel)和池化层(pooling).
然后将不同尺度的信息进行拼接。

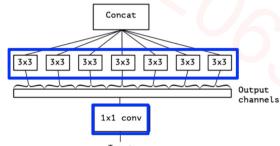
② Updates: Inception V2, V3, V4. 成差连接.
Inception V2: 增加 Batch Normalization.

更新的 Kernel. 去除 LRN 和 Dropout.

③ Factorisation: $n \times n$ filters \rightarrow 1×1 和 $1 \times n$ filters.



④ Xception: 学习跨域 channel 的信息。



⑤ 残差网络 ResNet. 随着深度的增加, 感受野消失。

① Wide ResNet. 增加宽度减少深度, 提升计算效率。

残差连接 模块堆叠 分类头。

② ResNeXt. ResNet + VGG + Inception

Stacking building block: 类似 VGG 的纵向堆叠。

Split-merge: 类似 Inception 的横向堆叠 Cardinality.
和 ResNet 同数量级复杂度. $O(Cb) = O(b)$

⑥ More Architectures.

① DenseNet. 深度监督 Deep supervision.

每层加上之前所有层直接连接, 特征重用, 增加效率。
优点: 缓解梯度消失, 更好传播特征, 特征重用, 减少参数。

② SqueezeNet. 比 AlexNet \downarrow 50倍参数但是效果一样。

模块: Squeeze + Expand. 组合 1x1 和 3x3 kernel.

③ MobileNet. 深度分离卷积 Depthwise Separable Convolution

标准复杂度: $D_k \cdot D_k \cdot M \cdot N \cdot D_f \cdot D_f$

DSC 复杂度: $D_k \cdot D_k \cdot M \cdot D_f \cdot D_f + M \cdot N \cdot D_f \cdot D_f$

④ ShuffleNet: 分组卷积 Pointwise Group convolution.

在输入的通道进行分组, 独立进行卷积。

g. 4 In 4 Out standard: $4 \times 4 = 16$, PGC: $2 \times (2+2) = 2 \times 4 = 8$.

\hookrightarrow 造成问题: 分组信息不交互, 利用 shuffle, 随机排列, 交互信息。

循环神经网络 RNNs.

① Recurrent Neural Network. (文本情感分析).

① Input: 独热编码 One-hot Encoding. 二进制编码。

每个编码的距离一样, 但是维度太高. 替代: Word hash/embedding

Output: 概率分布 $\xrightarrow{\text{通过BERT生成}}$

被认为是一种输入

② 实现: 通过 Memory 未端存放序列。

③ 问题: 由于需要储存过去的时间序列的信息。

Input 的长度将会很大, 导致梯度消失/爆炸。

g. 如果序列过长, $0.9^{\infty} \approx 0$, $1.0^{1000} \approx 2000$.

gradient = tanh(x) n \hookrightarrow $x \in [0,1] \hookrightarrow x \in (1,\infty)$

Reference: COMP5329 Deep Learning. Dr Chang Xu
University of Sydney. 悉尼大学.

长短记忆网络 LSTM.

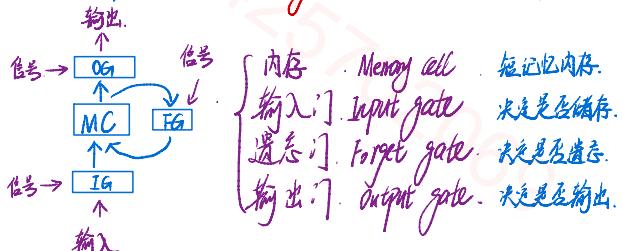
① Long Short-term Memory by Yann.

Motivation: 解决 RNN 处理

长距离时梯度消失/爆炸的问题

传统 RNN 难以记忆长期信息，梯度难以回传。

② 网络结构：通过信号 sigmoid 来控制信息的传递程度。



选择性地记住或者遗忘信息。gradient = $f_t \times C_{t-1} + i_t \cdot \tilde{C}_t$ 加性的梯度。

③ 短语。

Stacked LSTM: 堆叠多个 LSTM 增加表达能力，但复杂度高。

Bidirectional LSTM: 学习双向信息，更好理解上下文信息。

Grated Recurrent Unit: 简化了 LSTM, input gate 和 forget gate 结合。

自注意力 Self-Attention.

① Attention Mechanism.

Motivation: 模拟人类的注意力机制，选择性地聚焦重要部分，而忽略不重要的部分，在 NLP 中可以给予重要单词更高的权重。

优点：平行运算，更高的可解释性。Self-attention 是一种短语。

$$Q \in \mathbb{R}^{n \times d}$$

② 概念：给定 Query 查询，通过相似匹配 key，对 value 加权和。

Query 查询，想要查询的信息。

主体 Key 键，被查询的信息。

Value 值，被查询的值。

Q, K, V 的内积表示为两者的相似度，称作 A, Attention Matrix

$A \cdot V$ 表示通过查询之后的信息。

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) \cdot V \quad \text{结果也大。}$$

③ 多头自注意力 Multi-head Self-Attention.

模拟了类似 CNN 中多通道的概念，不同的 head 可以学习不同的特征表示，允许子空间关注不同维及信息。

④ 位置编码 Positional Encoding.

记录位置信息在 NLP 中也很重要。

通过 cos/sin 函数实现，任意长度生成周期性编码。

① 相对的位置信息 ② 多维及独立性。

⑤ BERT (Bidirectional Encoder Representation Transformer)

Motivation: 将 CV 中的预训练模型应用于 NLP.

self-supervised Bi-directional Pre-train.

无掩码无监督学习 上下文信息 大模型。

对比 ELMo (Embedding from Language Model)

ELMo: Bi-LSTM 从左到右 复杂度高。

BERT: Transformer 掩码 更高性能。

② 机制。

掩码学习，无形填空，理解上下文信息。

CLS Token: 聚合所有 sequence 的信息作为分类标准。

SPE Token: 由于只有 Encoder，用于拼接输入/输出。

④ GPT (Generative Pre-training)

Motivation: 自回归模型，在大量的无监督样本上进行学习。

只采用 Transformer Decoder，用 不需要标注信息。

逐个单词预测下一个单词。 175B 的参数。

Two limitations of supervised model: ① 大量标注数据。

loss: $L(t) = -\sum_i \log P(t|t_1, t_2, \dots, t_{i-1}, \theta)$ ② 迭代性能差。

预测第 t 步， $t+1$ 时将 y 替换 t 。

GPT 1~3: 数据更多，参数更大。

⑤ InstructGPT: 将 GPT 的目标向人类喜好对齐。

原本的 GPT 生成 meaningless, untruthful, toxic outputs.

InstructGPT 使用了 RLHF 的技术

Reinforcement learning with Human Feedback.

③ RLHF 单独训练一个人类喜好模型，对 Pre-train
Pre-train Language Model (LM) 模型进行微调。
Train Reward Model (RM) → 通过 LM 的输出人类
Fine-tuning: LM with RM. 进行排序。
ChatGPT 在更好的数据集上训练的 InstructGPT.

⑤ Vision Transformers 将图片分割成多个 patches 平铺成向量。

Motivation: 创造简单通用的 transformer 架构。
和 BERT 相同的 CLS Token 用于分类 (有监督)。

通常在更大的数据集上表现得比 ResNet 好。

小数据集 ImageNet (small) 比 ResNet 差一些。

中数据集 ImageNet-21K (medium) 和 ResNet 差不多。

大数据集 JFT (large) 比 ResNet 好。

② 特点。

更好地学习 global feature。

更好的 scalability, 更大数据集，更多参数，更好效果。

较少 CNN 的归一化偏置，locality / translation invariance。

内存占用大，需要对所有的 patch 进行交互。

Reference: COMP5329 Deep Learning. Dr Chang Xu
University of Sydney. 悉尼大学。

图卷积神经网络 GCNs.

1) 图卷积神经网络. Graph Convolutional Networks. by yann.

① Motivations. 图是一种常见的表示关系的数据结构.

图神经网络的应用: 推荐系统, 分子结构预测...

关键思想是通过图的结构, 每个节点将聚合临近节点的信息, 进行上下文学习.

② 性质:

权重共享 weight sharing. 所有的节点使用相同的更新规则.

无论模型的大小/形状, 可以应用同模型 (更少参数).

对置不变性 Permutation invariance. 图顺序对结果无影响.

线性复杂度 Linear complexity. 复杂度和边数成正比.

转换到树的设置 Transductive / inductive settings. 对未见过的结构

→ 对未知边的节点预测. 预测.

③ 局限性:

① 信息消失. 由于一个节点聚合多节点信息, 多次传播.

② 边特征不直观. 主要标注 Nodes 而不是 Edges.

④ Update Formulas:

$$H^{(l+1)} = \text{Activation function}(\hat{D}^{-\frac{1}{2}} \hat{A} \hat{D}^{\frac{1}{2}} H^{(l)} W^{(l)})$$

Degree matrix \hat{A} → Normalised adjacency matrix.

归一化 $\hat{A} = A + I_n$, 增加 self-loop 保证节点能看到自己信息.

对称归一化: $\hat{D}^{\frac{1}{2}} \hat{A} \hat{D}^{\frac{1}{2}}$, 如果不做 degree 归一化, 大的 degree 将会

对节点产生更大的影响. 邻接矩阵的特征值范围会被限制在较小的范围内, 防止梯度消失.

2) 光谱法 Spectral Approach.

① Graph Laplacian. $L = D - A$.

Normalized: $L' = I_n - D^{\frac{1}{2}} A D^{\frac{1}{2}}$

② 卷积原理: f 是输入, h 是滤镜, 卷积表示为.

$$f * h(x) = \sum_{k=1}^K [F(k) \cdot F(k)]$$

→ 时域卷积, 频域乘积, 为函数 f 和 h 各自傅里叶变化乘积的逆傅里叶变化.

目标检测 Object Detection.

1) 视觉任务 Computer Vision Tasks.

单物体 { classification

classification + localisation

多物体 { object detection

Instance segmentation.

2) Region CNN.

① Selective search. 通过 search 等方法找到 Region of Interest. 这个过程也被称作 region proposal. 通过 greedy similarity measures 未找到 likely 的 RoI.

② Bounding Box Regression. 找到的 RoI 不一定好, 所以需要通过 BBR 对结果进行一些修正.

③ 局限性: 非 end-to-end 架构, 重复计算, 运行慢.

④ Fast R-CNN. 解决了 R-CNN 上述的局限性. 不再是 multi-stage training. 用一个 CNN 进行计算, 引用了 RoI pooling. 在 feature map 上进行 region propose, 成功率.

⑤ Faster R-CNN. 将 Fast R-CNN 中的 selective search 替换为 region proposal network 从而自动学习 RoI, 加速效率.

⑥ Masked R-CNN. 可以理解为一个多任务模型. 通过添加一个并行的分支网络来预测像素级别的掩码, 实现分割的效果. 并且加入了 RoI Align 解决了 Faster R-CNN 中空间不一致性的問題, 通过 bilinear interpolation.

生成网络 Generative Net.

1) 对抗生成网络 Generative Adversarial Network.

① Motivation: 同时训练 Generator 和 Discriminator.

$$L = \max_G \min_D V(D, G) = \mathbb{E}_{x \sim P_{\text{data}}} [\log D(x)] + \mathbb{E}_{z \sim P_{\text{G}}(z)} [\log (1 - D(G(z)))]$$

$G(z)$ 假逼真, $D(G(z))$ 忽变成 1, 从而使得整体差中.

2) Wasserstein GAN

① Motivation: 解决 GAN 训练不稳定的问题. 当 D 很强很小时 (基于 KL 故障), 训练容易出现梯度消失的问题. 也称作模型坍塌 model collapse.

$$\text{Wasserstein Distance: } W(P_r, P_g) = \inf_{\pi \in \Pi(P_r, P_g)} \mathbb{E}_{x \sim P_r, y \sim P_g} [\|x - y\|]$$

3) 变分编码器 Variational Auto-Encoder.

① Motivation: 在传统的 Auto-Encoder 中, 特征被压缩到潜在空间. 但是 VAE 目标是对特征编码为正态分布. 每一个输入都被编码为一组平均值和标准差, 并且引入了随机性. 且最后使用 VAE 的时候可以丢弃编码器.

4) VAE+GAN.

① Motivation: 结合 GAN 能够生成逼真的样本. VAE 能够生成稠密连贯的样本. 通常做法是将 VAE 的重构损失替换 GAN 的判别器损失.

Reference: COMP5329 Deep Learning. Dr Chang Xu
University of Sydney. 悉尼大学.