基础

Sisyphes

2020 年 9 月 4 日

目录

第一节　问题

- 什么是不变的？
- 一个良好的 DL 实验平台需要什么？
- 怎么落地？

1.1　计划

此 pdf 为基础章节，涉及基础理论，基本元素，基本代码，实验平台的设计，其他会附加三个 pdf，包含分类，检测，和生成。分类集中几个经典网络，优化技巧，和实用代码；检测核心围绕 mmdetection 讲解；生成精选内容细讲，理论，应用，优化技巧。

第二节　理论

这节内容主要由书本，维基，网络资源整理而得。学习某些概念，方法我一般从维基开始。每一种概念，方法背后都有丰富的历史，对一些概念的理解，不是一蹴而就，往往随着论文阅读，实验次数增加，会不断加深，这里只是抛砖迎玉。

2.1　熵，交叉熵，KL 散度，极大似然，最大后验

信息熵 来源于 热力熵，是对不确定性的度量，由随机变量的信息量的期望值来表示。怎么刻画信息量？概率的倒数的对数值。

定义 2.1 *Named after Boltzmann's H-theorem, Shannon defined the entropy H (Greek capital letter eta) of a discrete random variable X with possible values* $\{x_1, \ldots, x_n\}$ *and probability mass function* $\mathrm{P}(X)$ *as:*

$$\mathrm{H}(X) = \mathrm{E}[\mathrm{I}(X)] = \mathrm{E}[-\log(\mathrm{P}(X))]$$

Here E *is the expected value operator, and I is the information content of* X.I(X) *is itself a random variable.*

The entropy can explicitly be written as:

$$\mathrm{H}(X) = -\sum_{i=1}^{n} \mathrm{P}\left(x_i\right) \log_b \mathrm{P}\left(x_i\right)$$

where b is the base of the logarithm used. Common values of b are 2(bits), Euler's number e(nats), and 10(bans).

交叉熵 在给定集合下，概率分布 q 相对概率分布 p 的交叉熵定义为：

$$H(p, q) = -\mathrm{E}_p[\log q]$$

即离散概率分布 p, q 有相同支撑集 $\mathcal{X}$，则

$$H(p, q) = -\sum_{x \in \mathcal{X}} p(x) \log q(x)$$

KL 散度 (相对熵) Q 相对 P 的散度定义为：

$$D_{\mathrm{KL}}(P \| Q) = \sum_{x \in \mathcal{X}} P(x) \log \left(\frac{P(x)}{Q(x)} \right)$$

容易得出 $\mathrm{H}(P,Q)-\mathrm{H}(P)$。

最小二乘发展于天文学和大地测量学领域，科学家和数学家尝试为大航海探索时期的海洋航行挑战提供解决方案。十八世纪对一些观测数据的分析情况：

- 1722 年 Roger Cotes 阐明：不同观测值的组合是真实值的最佳估计；多次观测会减少误差而不是增加
- 评定对误差达到最小的解决方案标准，1799 前后拉普拉斯指明了误差的概率密度的数学形式，并定义了误差最小化的估计方法 (极大似然影子)。

1809 年高斯发表《天体运动论》著作，首次出现了最小二乘方法，所以通常将最小二乘法归功于高斯，但阿德里安-马里・勒让德早在 1806 年也发表过同样的方法。

最小二乘的核心思想是：使得观测值与模型拟合值之差的平方和最小。

对于任何表达式具体的带参数函数，理论上都是可以解出参数的。存在将一些非线性变量变换到线性形式，不过一般非线性问题由迭代来解决，在每次迭代中，系统由线性近似。

损失函数为 mse 的神经网络回归可以看作非线性最小二乘，其参数由梯度迭代优化。

极大似然估计

定义 2.2 给定一个概率分布 D，已知其概率密度函数（连续分布）或概率质量函数（离散分布）为 f_D，以及一个分布参数 θ，我们可以从这个分布中抽出一个具有 n 个值的采样 $X_1, X_2, \ldots, X_n$，利用 f_D 计算出其似然函数：$L(\theta \mid x_1, \ldots, x_n) = f_\theta(x_1, \ldots, x_n)$. 其估计

$$\hat{\theta}_{\mathrm{ML}}(x) = \arg\max_{\theta} f(x \mid \theta)$$

就是 θ 的极大似然估计。

极大似然估计核心有两点，似然函数和极大。似然函数是一种关于统计模型中的参数的函数，表示模型参数中的似然性，用某些观测数据，对有关

事物的性质的参数进行估值。比如神经网络的分类问题，用网络参数去估计图片的类别概率，网络关于数据集的输出概率的乘积就是该数据集的似然函数 (不过神经网络可能不是一个真的概率密度函数)。极大就是似然函数值最大，这样解出参数能最好的拟合该数据集。具体例子见维基百科。不过极大似然估计不一定存在，也不一定唯一。

最大后验概率估计

定义 2.3 假设 θ 存在一个先验分布 g, 这就允许我们将 θ 作为贝叶斯统计中的随机变量, 这样 θ 的后验分布就是:

$$\theta \mapsto \frac{f(x \mid \theta)g(\theta)}{\int_\Theta f\left(x \mid \theta'\right) g\left(\theta'\right) d\theta'}$$

其中 Θ 是 g 的 *domain*, 那么最大后验估计为:

$$\hat{\theta}_{\mathrm{MAP}}(x) = \arg\max_\theta \frac{f(x \mid \theta)g(\theta)}{\int_\Theta f\left(x \mid \theta'\right) g\left(\theta'\right) d\theta'} = \arg\max_\theta f(x \mid \theta)g(\theta)$$

简单讲就是给极大似然的参数附加一个先验概率分布。不过极大似然可用频率学派解释，最大后验则有贝叶斯学派思想，两者的区别，深化需自己完善。

回到机器学习，深度学习：

机器学习有回归算法 Least Square，Ridge，LASSO，可参考sklearn linear model，从概率论角度讲，可以使用上面的基础理论给出统一解释：

1 Least Square 的解析解可以用 Gaussian 分布以及极大似然估计求得

2 Ridge 回归可以用 Gaussian 分布和最大后验估计解释

3 LASSO 回归可以用 Laplace 分布和最大后验估计解释

数学表达我采用知乎 bsdelf 的解析来说明。

假设线性回归模型具有如下形式:

$$f(\mathbf{x}) = \sum_{j=1}^{a} x_j w_j + \epsilon = \mathbf{x}\mathbf{w}^\top + \epsilon$$

其中$\mathbf{x} \in \mathbb{R}^{1\times d}, \mathbf{w} \in \mathbb{R}^{1\times d}$, 误差$\epsilon \in \mathbb{R}$

当前已知$\mathbf{X} = (\mathbf{x}_1 \cdots \mathbf{x}_n)^\top \in \mathbb{R}^{n\times d}$,　$\mathbf{y} \in \mathbb{R}^{n\times 1}$, 怎样求$\mathbf{w}$ 呢?

对应最小二乘：假设 $\epsilon_i \sim \mathcal{N}(0,\sigma^2)$，也就是说 $\mathbf{y}_i \sim \mathcal{N}\left(\mathbf{x}_i\mathbf{w}^\top,\sigma^2\right)$，那么用极大似然估计推导:

$$\begin{aligned}\arg\max_{\mathbf{w}} L(\mathbf{w}) &= \ln\prod_{i=1}^{n}\frac{1}{\sigma\sqrt{2\pi}}\exp\left(-\frac{1}{2}\left(\frac{\mathbf{y}_i-\mathbf{x}_i\mathbf{w}^\top}{\sigma}\right)^2\right)\\ &= -\frac{1}{2\sigma^2}\sum_{i=1}^{n}\left(\mathbf{y}_i-\mathbf{x}_i\mathbf{w}^\top\right)^2 - n\ln\sigma\sqrt{2\pi}\\ \arg\min_{\mathbf{w}} f(\mathbf{w}) &= \sum_{i=1}^{n}\left(\mathbf{y}_i-\mathbf{x}_i\mathbf{w}^\top\right)^2 = \left\|\mathbf{y}-\mathbf{X}\mathbf{w}^\top\right\|_2^2\end{aligned}$$

对应二范数正则化：假设 $\epsilon_i \sim \mathcal{N}(0,\sigma^2)$，　$\mathbf{w}_i \sim \mathcal{N}(0,\tau^2)$，那么用最大后验估计推导

$$\begin{aligned}\arg\max_{\mathbf{w}} L(\mathbf{w}) &= \ln\prod_{i=1}^{n}\frac{1}{\sigma\sqrt{2\pi}}\exp\left(-\frac{1}{2}\left(\frac{\mathbf{y}_i-\mathbf{x}_i\mathbf{w}^\top}{\sigma}\right)^2\right)\cdot\prod_{j=1}^{d}\frac{1}{\tau\sqrt{2\pi}}\exp\left(-\frac{1}{2}\left(\frac{\mathbf{w}_j}{\tau}\right)^2\right)\\ &= -\frac{1}{2\sigma^2}\sum_{i=1}^{n}\left(\mathbf{y}_i-\mathbf{x}_i\mathbf{w}^\top\right)^2 - \frac{1}{2\tau^2}\sum_{j=1}^{d}\mathbf{w}_j^2 - n\ln\sigma\sqrt{2\pi} - d\ln\tau\sqrt{2\pi}\\ \arg\min_{\mathbf{w}} f(\mathbf{w}) &= \sum_{i=1}^{n}\left(\mathbf{y}_i-\mathbf{x}_i\mathbf{w}^\top\right)^2 + \lambda\sum_{j=1}^{d}\mathbf{w}_j^2\\ &= \left\|\mathbf{y}-\mathbf{X}\mathbf{w}^\top\right\|_2^2 + \lambda\|\mathbf{w}\|_2^2\end{aligned}$$

对应一范数正则化：$\epsilon_i \sim \mathcal{N}(0,\sigma^2)$, $\mathbf{w}_i \sim \text{Laplace}(0,b)$, 同样用最大后验估计推导:

$$\begin{aligned}\arg\min_{\mathbf{w}} f(\mathbf{w}) &= \sum_{i=1}^{n}\left(\mathbf{y}_i-\mathbf{x}_i\mathbf{w}^\top\right)^2 + \lambda\sum_{j=1}^{d}|\mathbf{w}_j|\\ &= \left\|\mathbf{y}-\mathbf{X}\mathbf{w}^\top\right\|_2^2 + \lambda\|\mathbf{w}\|_1\end{aligned}$$

上面只是针对线性模型，明显对任意的 $f_\theta(X)$,$Y = f_\theta(X)+\varepsilon$ 都可做如上解释。

不过也能想到拉格朗日条件视角：将正则化因子解释为参数距离原点的范数限制。于是对于：$\min_\theta \sum_i \left(y_i - f_\theta\left(x_i\right)\right)^2 + \lambda\theta^2$, 可写出

$$\min_\theta \sum_i \left(y_i - f_\theta\left(x_i\right)\right)^2 \ \text{且}\theta^2 \le r$$

这样可以利用几何的观点去解释分析。

对于深度学习

Mean Square Loss 可理解为预测误差服从标准正太分布的极大似然估计,

Binary Cross Entropy Loss 可理解为采用伯努利分布的极大似然估计模型,

Cross Entropy Loss 也可采用极大似然角度理解。

对分布的详细了解，可参考distribution-is-all-you-need。

总结：

给定数据集，其熵值是的常数，因此优化 KL 散度与交叉熵等价，从概率角度看，优化交叉熵与求解极大似然估计等价，所以，极小化 KL 散度、交叉熵、极大化似然得到的结果等价。而正则化，是将观点从频率转到贝叶斯，可以理解为给似然函数加了先验。具体的，L1 正则化相当于拉普拉斯先验，L2 正则化相当于高斯先验。

以上统计学角度，信息论角度理解同样的问题，给出了一些东西可以从不同角度去理解的例子，这样的例子很多，似乎都在证明，很多东西最终都会交汇在一起。

信息熵	极大似然	最大后验	正则化
KL, Binary/Cross Entropy		L1 拉普拉斯，L2 高斯	
	MSELoss, 最小二乘	Ridge, LASSO	

2.2　卷积

2.2.1　函数卷积

参考 stein fourier analysis。

2.2.2　深度学习

im2col，卷积加速 Winograd 原理：Winograd-梁德澎

从 0 实现，自己看看就行。

2.3　优化

这小节可看看mmdet.pdf

2.3.1　常用算法

2.3.2　初始化，学习率

2.4　向量矩阵微分

向量或矩阵求导的含义直观说就是分别求出因变量各分量各自关于自变量各分量的导数 (定义)，但存在如何对结果布局的问题。实际有分子，分母两种布局 (分子布局及保持分子的形状)，见Matrix calculus。

这里约定向量为列向量。向量对向量以分子布局，其余以分母布局。标量，向量，矩阵求导有如下九种情况：

自变量 \ 因变量	标量 y	向量 $\mathbf{y}$	矩阵 $\mathbf{Y}$
标量 x	$\frac{\partial y}{\partial x}$	$\frac{\partial \mathbf{y}}{\partial x}$	$\frac{\partial \mathbf{Y}}{\partial x}$
向量 $\mathbf{x}$	$\frac{\partial y}{\partial \mathbf{x}}$	$\frac{\partial \mathbf{y}}{\partial \mathbf{x}}$	$\frac{\partial \mathbf{Y}}{\partial \mathbf{x}}$
矩阵 $\mathbf{X}$	$\frac{\partial y}{\partial \mathbf{X}}$	$\frac{\partial \mathbf{y}}{\partial \mathbf{X}}$	$\frac{\partial \mathbf{Y}}{\partial \mathbf{X}}$

其中标量对向量、矩阵，以及向量对向量和深度学习相关。因此以下只总结这三块内容。完整的逻辑可参考刘建平数学统计。

定义法求解标量对向量，矩阵，向量对向量只能针对简单的，比如 $y=\mathbf{x}^T\mathbf{A}\mathbf{x}$，有 $\frac{\partial \mathbf{x}^T\mathbf{A}\mathbf{x}}{\partial \mathbf{x}}=\mathbf{A}^T\mathbf{x}+\mathbf{A}\mathbf{x}$，$y=\mathbf{a}^T\mathbf{X}\mathbf{b}$ 有 $\frac{\partial \mathbf{a}^T\mathbf{X}\mathbf{b}}{\partial \mathbf{X}}=ab^T$，$\mathbf{y}=\mathbf{A}\mathbf{x}$ 有 $\frac{\partial \mathbf{A}\mathbf{x}}{\partial \mathbf{x}}=\mathbf{A}^T$。复杂的排布是个问题，这时可用通用求解方法：矩阵微分。但在多层链式关系中 (深度学习中)，直接使用微分法也不会容易。不过针对一些简单情形，可总结出一些结论，简化 DL 反向传播的推导流程。

回想数分，多变量微分有：

$$df=\sum_{i=1}^{n}\frac{\partial f}{\partial x_i}dx_i=(\frac{\partial f}{\partial \mathbf{x}})^T d\mathbf{x}$$

推广到矩阵微分：

$$df=\sum_{i=1}^{m}\sum_{j=1}^{n}\frac{\partial f}{\partial X_{ij}}dX_{ij}=tr((\frac{\partial f}{\partial \mathbf{X}})^T d\mathbf{X})$$

结合微分的基本性质和迹函数的基本性，就可以求解更复杂的导数 (需多加练习)。比如 $y=\mathbf{a}^T exp(\mathbf{X}\mathbf{b}),$**求解** $\frac{\partial y}{\partial \mathbf{X}}$

$dy=tr(dy)=tr(\mathbf{a}^T dexp(\mathbf{X}\mathbf{b}))=tr(\mathbf{a}^T(exp(\mathbf{X}\mathbf{b})\odot d(\mathbf{X}\mathbf{b})))=tr((\mathbf{a}\odot exp(\mathbf{X}\mathbf{b}))^T d\mathbf{X}\mathbf{b})=tr(\mathbf{b}(\mathbf{a}\odot exp(\mathbf{X}\mathbf{b}))^T d\mathbf{X})$。

因此 $\frac{\partial y}{\partial \mathbf{X}}=(\mathbf{a}\odot exp(\mathbf{X}\mathbf{b}))b^T$。

其中第四个等式用了矩阵乘法和迹交换性质**:**$tr((A\odot B)^T C)=tr(A^T(B\odot C))$。

利用以上知识，即可总结一些对 DL 中链式求导有用的结论：

1. 向量对向量：若 $\mathbf{x}\to\mathbf{y}\to\mathbf{z}$ 则 $\frac{\partial \mathbf{z}}{\partial \mathbf{x}}=\frac{\partial \mathbf{z}}{\partial \mathbf{y}}\frac{\partial \mathbf{y}}{\partial \mathbf{x}}$
2. 标量对多个向量**:**若 $\mathbf{y_1}\to\mathbf{y_2}\to...\to\mathbf{y_n}\to z$ 则 $\frac{\partial z}{\partial \mathbf{y_1}}=(\frac{\partial \mathbf{y_n}}{\partial \mathbf{y_{n-1}}}\frac{\partial \mathbf{y_{n-1}}}{\partial \mathbf{y_{n-2}}}...\frac{\partial \mathbf{y_2}}{\partial \mathbf{y_1}})^T\frac{\partial z}{\partial \mathbf{y_n}}$
3. 矩阵 A,X,B,Y，标量 z

   3.1 $z=f(Y),Y=AX+B\to\frac{\partial z}{\partial X}=A^T\frac{\partial z}{\partial Y}$，$X$ 是向量同理

   3.2 $z=f(Y),Y=XA+B\to\frac{\partial z}{\partial X}=\frac{\partial z}{\partial Y}A^T$，$A$ 是向量同理

对于第 2 点，$\mathbf{x}\to\mathbf{y}\to z$。若 $\mathbf{x},\mathbf{y}$ 分别为 m,n 维向量，则 $\frac{\partial z}{\partial \mathbf{x}}$ $\frac{\partial z}{\partial \mathbf{y}}$, $\frac{\partial \mathbf{y}}{\partial \mathbf{x}}$ 分别是 $m\times 1, n\times 1$ 向量和 $n\times m$ 矩阵，没法直接乘，但转置一下有：

$$(\frac{\partial z}{\partial \mathbf{x}})^T=(\frac{\partial z}{\partial \mathbf{y}})^T\frac{\partial \mathbf{y}}{\partial \mathbf{x}}$$

因此，可以有链式关系：

$$\frac{\partial z}{\partial \mathbf{x}} = (\frac{\partial \mathbf{y}}{\partial \mathbf{x}})^T \frac{\partial z}{\partial \mathbf{y}}$$

推广到多个即为 2 所给结论。

说明：此小节内容简化自刘建平博客。

2.5　反向传播

Hinton，加点历史。$\odot$ 表示Hadamard product。

2.5.1　DNN

对最后一层 $a^L = \sigma(z^L) = \sigma(W^L a^{L-1} + b^L)$，若损失函数为 MSE(平方差)，则

$$J(W, b, x, y) = \frac{1}{2}||a^L - y||_2^2 = \frac{1}{2}||\sigma(W^L a^{L-1} + b^L) - y||_2^2$$

结合上节链式结论，可得：

$$\frac{\partial J(W, b, x, y)}{\partial W^L} = \frac{\partial J(W, b, x, y)}{\partial z^L} \frac{\partial z^L}{\partial W^L} = [(a^L - y) \odot \sigma^{'}(z^L)](a^{L-1})^T \tag{1}$$

$$\frac{\partial J(W, b, x, y)}{\partial b^L} = (a^L - y) \odot \sigma^{'}(z^L) \tag{2}$$

这里 σ 为 sigmoid 或者 Relu 等单个变量激活函数，非 softmax(否则不能简化为 $\odot$ 运算)。

令 $\sigma^l = \frac{\partial J(W,b,x,y)}{\partial z^l}, 1 \le l \le L$，则根据上小节链式，有

$$\sigma^l = (\frac{\partial z^L}{\partial z^{L-1}} \frac{\partial z^{L-1}}{\partial z^{L-2}} ... \frac{\partial z^{l+1}}{\partial z^l})^T \frac{\partial J(W, b, x, y)}{\partial z^L} \tag{3}$$

当 $l = L$，容易算出 σ^L(不同损失函数结果不同)。

假设第 σ^{l+1} 已算出,根据递推关系 $\delta^l = \frac{\partial J(W,b,x,y)}{\partial z^l} = (\frac{\partial z^{l+1}}{\partial z^l})^T \frac{\partial J(W,b,x,y)}{\partial z^{l+1}} = (\frac{\partial z^{l+1}}{\partial z^l})^T \delta^{l+1}$，因 $z^{l+1} = W^{l+1} a^l + b^{l+1} = W^{l+1} \sigma(z^l) + b^{l+1}$，可得 $\frac{\partial z^{l+1}}{\partial z^l} = W^{l+1} diag(\sigma^{'}(z^l))$，于是得到

$$\delta^l = diag(\sigma^{'}(z^l))(W^{l+1})^T \delta^{l+1} = (W^{l+1})^T \delta^{l+1} \odot \sigma^{'}(z^l) \tag{4}$$

利用归纳法可证明。

于是得到：

$$\frac{\partial J(W,b,x,y)}{\partial W^l} = \delta^l (a^{l-1})^T \tag{5}$$

$$\frac{\partial J(W,b,x,y)}{\partial b^l} = \delta^l \tag{6}$$

损失函数为 CrossEntropy，最后激活为 Softmax，即 $J(W,b,x,y) = -y^T \cdot log(softmax(z^L))$。此时的 $\sigma^L = \frac{-y}{softmax(z^L)} \cdot \frac{\partial softmax(z^L)}{z^L} = softmax(z^L) \odot (\sum y, \sum y, ..., \sum y)^T - y$。最后等式为化简后的结果。注意这里 $\frac{\partial softmax(z^L)}{z^L}$ 为一非对角矩阵，这和一般激活函数不同。

DNN BP 算法流程：

输入: 总层数 L，以及各隐藏层与输出层的神经元个数，激活函数，损失函数，迭代步长 α, 最大迭代次数 MAX 与停止迭代阈值 ϵ，输入的 m 个训练样本 $(x_1, y_1), (x_2, y_2), ..., (x_m, y_m)$。

1) 初始化各隐藏层与输出层的线性关系系数矩阵 W 和偏倚向量 b 的值为一个随机值。

2) for iter to 1 to MAX：

2-1) for i =1 to m：

a) 将 DNN 输入 a^1 设置为 x_i

b) for$l = 2$to L,进行前向传播算法计算 $a^{i,l} = \sigma(z^{i,l}) = \sigma(W^l a^{i,l-1} + b^l)$

c) 通过损失函数计算输出层的 $\delta^{i,L}$

d) for$l = L-1$to 2,进行反向传播算法计算 $\delta^{i,l} = (W^{l+1})^T \delta^{i,l+1} \odot \sigma^{'}(z^{i,l})$

2-2) for$l = 2$to L，更新第 l 层的 W^l, b^l：

$$W^l = W^l - \alpha \sum_{i=1}^{m} \delta^{i,l} (a^{i,l-1})^T$$

$$b^l = b^l - \alpha \sum_{i=1}^{m} \delta^{i,l}, \leq i \leq m$$

2-3) 如果所有 W，b 的变化值都小于停止迭代阈值 ϵ，则跳出迭代循环到步骤 3。

3) 输出各隐藏层与输出层的线性关系系数矩阵 W 和偏倚向量 b。

说明：此小节内容简化自刘建平博客。

2.5.2 CNN

DNN 的反向传播核心公式如下四个：

$$\delta^L = \frac{\partial J(W,b)}{\partial z^L} = \frac{\partial J(W,b)}{\partial a^L} \odot \sigma^{'}(z^L) \tag{7}$$

$$\delta^l = (\frac{\partial z^{l+1}}{\partial z^l})^T \delta^{l+1} = (W^{l+1})^T \delta^{l+1} \odot \sigma^{'}(z^l) \tag{8}$$

$$\frac{\partial J(W,b)}{\partial W^l} = \delta^l (a^{l-1})^T \tag{9}$$

$$\frac{\partial J(W,b,x,y)}{\partial b^l} = \delta^l \tag{10}$$

而对于 CNN，因其中间操作 (卷积，池化等) 和仿射变换有差别，因此其具体反传方式需要对以上 4 式做一些改变，具体推导可阅读CNN BP-刘建平，Michael Nielsen。

2.6 泛化

模型的泛化可以理解成模型对输入数据的微小扰动不敏感。这时可用 Lipschitz 约束来刻画:

$$||f_w(x) - f_w(x+\epsilon)|| \leq C(w) \cdot ||\epsilon||$$

其中函数 f_w 的变动被其系数 w 控制。模型泛化能力越强，表示 ϵ 小幅度变化时 $C(w)$ 能保持很小的值。

由于深度学习的映射函数是线性映射加一个非线性变换的基本结构按层递推而得，所以分析某一层以及各层连接关系即可。对第 l 层，不难写出：

$$x^l = f^l(W^l x^{l-1} + b^l)$$

其中 $x^{l-1} \in \mathbb{R}^{n_{l-1}}$ 为 $l-1$ 层的输出，$f^l : \mathbb{R}^{n_l} \to \mathbb{R}^{n_l}$ 为 l 层的激活函数 (ReLU,maxout,maxpooling)，$W^l \in \mathbb{R}^{n_l \times_{l-1}}$ 表示映射 $\mathbb{R}^{n_l-1} \to \mathbb{R}^{n_l}$ 。于是我们可得到 l 层的 Lipschitz 关系：

$$\left\| f^l\left(W^l x^{l-1} + b^l\right) - f^l\left(W^l\left(x^{l-1} + \epsilon\right) + b^l\right)\right\| \leq C\left(W^l\right) \cdot \|\epsilon\|$$

当 ϵ 足够小时，左边即可用 Taylor 一阶项来估计，于是:

$$\left\| \frac{\partial f^l}{\partial x^{l-1}} W^l \epsilon \right\| \leq C\left(W^l\right) \cdot \|\epsilon\|$$

考虑到神经网络的激活函数导数都是有界的，故可以将其去掉，于是我们得到第 l 层权重参数的 Lispschitz 约束关系 (其他层类似)。我们希望 $C(W)$ 足够小使得我们的模型有足够的泛化能力，其实就是计算 W 的谱范数。因为谱范数的定义:

$$||W||_2 = \max_{x \neq 0} \frac{||Wx||}{||x||} \tag{11}$$

我们熟知的 $l1, l2$ 正则化，分别是矩阵的 1 范数，Frobenius 范数 (2 范数)，当然也有 p 范数详见Matrix_norm。从而，可以看出矩阵范数在模型泛化中的位置。根据 (1) 式容易得到谱范数对应了 $\boldsymbol{W}^T\boldsymbol{W}$ 的最大特征根，而矩阵特征根我们可以用迭代法得到:

$$\boldsymbol{u}^l \leftarrow \boldsymbol{W}^l \boldsymbol{v}^l, \boldsymbol{v}^l \leftarrow \left(\boldsymbol{W}^l\right)^T \boldsymbol{u}^l, \sigma^l \leftarrow \left\|\boldsymbol{u}^l\right\| / \left\|\boldsymbol{v}^l\right\|$$

因我们需要最终函数关于输入的微小改动，变化不大，而不是各层的函数，这点在论文Spectral Normalization-gan 6,7,8 式证明了，只需限制每层的权重的谱范数为 1 即可 (这里是针对生成网络的，并不是对任何网络都成立)。更理论的论文可参考Identifying Generalization Properties in Neural Networks，比较新的可参考Gradient-Centralization。

实际情况，增加模型的泛化 (Generalization) 能力，不会局限于如上思路，比如直接加扰动训练，模拟出更多样化的数据，Dropout 一些模型组件等。

第三节　代码

3.1　元素

从整体上说一下，DL 所包含的一些东西。为何存在，什么含义，作用，改进等。随意写，后改。

- 可微
- 优化
- 卷积
- 归一化
- 泛化
- 增强
- anchor
- 数据，标注
- 感受野，尺度，插值，参数量，特征，表示，损失函数
- 分类，检测，生成
- opencv
  - 一些算法

3.2　函数

这里提供所有代码实现，选例 (待优化)：

- SVD
- MLP
- Conv
- BP
- Adam
- spectral_norm
- Cross Entropy
- Iou
- NMS，Soft-NMS，DIoU NMS
- Init
- Find Learning
- MAP
- Confuse Matrix
- Augmentation
- KL
- Plot(画图很重要)plot

3.3　网络

这里会从整体上把这些网络讲一遍。最后会在分类章节，检测章节，生成章节，进行更细致的讲解。

- ResNet, DarkNet, MoibleNet, EfficientNet
- Face-RFB, RFBNet
- CenterNet-DLA-Hourglass, hourglass
- Attention(Selayer(channel-wise attention), cbam, Skblock, gcblock)
- FPN, BiFPN, PAN, SFAM, ASFF
- DCN
- SSH, SPP, ASPP, RFB(增加感受野)
- SAM(point-wise attention),CSP(Skip-connections) 特征融合
- Generator(stylegan2)

性能图展示。

问题：**不同任务，不同网络结构之间的本质区别在哪？**

一些情况是实验组的祖传代码导致，但问题本身是值得考虑的。

3.4　设计

不关注底层框架，只关注在算法实验平台 (炼丹炉) 的设计。可参考代码如下：

- detectron2
- mmdetection
- AlexeyAB-YoLo
- torchgan
- PyAnomaly
- pytorch lightning
- pytorch-CycleGAN-and-pix2pix
- stylegan2

最终会详细讲解 mmdetection。

3.4.1　构成

待优化。

- 配置系统 (训练，数据，模型等)
- 可视化 (训练监视，特征可视化，结果可视化)
- 日志
- 评估 (分类，检测，生成)
- 训练流程 (动态调整学习率，模型保存，续训练，期间评估)
- 模型构建 (灵活，解耦，注册机制)
- 数据结构 (统一)
- 数据类 (格式互转，统一化)
- 模型管理 (灵活加载，格式转化，保存策略)
- 其他

提供一个模板代码，有时间可以将所讲内容的代码整合进去 (比较费时间，提供简易版本也行)。

3.5　工程化

提供具体的落地项目代码。

- 量化
- 业务逻辑

第四节　资源

寻找好的资源是进步的关键。