

深度学习笔记

zhliangqi

2020 年 11 月 30 日

目录

1	Linear Algebra	7
1.1	特征值和特征空间	7
1.2	奇异值分解	7
1.2.1	降维和图像压缩	7
2	Statistics	9
2.1	Frequentist statistics VS Bayesian statistics	9
2.2	期望	9
2.3	方差	9
2.4	最小方差无偏估计	10
3	Basic	11
3.1	Mechine Learning	11
3.2	iid	11
3.3	Universal approximation theorem	11
3.4	Maximum likelihood estimation	11
3.4.1	KL divergence	12
3.4.2	熵、KL散度和交叉熵	12
3.5	Maximum A Posteriori	12
3.6	Initialization scheme	13
3.6.1	constant initialization	13
3.6.2	random initialization	13
3.6.3	xavier initalization	13
3.6.4	orthogonal initalization	15
3.6.5	kaiming initalization	15
3.7	正则化	15
3.8	Data proprocessing	15
3.8.1	whitening	15
3.9	Activation functions	15
3.9.1	sigmoid	15
3.9.2	tanh(x)	16
3.9.3	Rectified linear units	16
3.9.4	haha	17
3.9.5	Loss functions	17
3.10	信息熵	17

3.10.1	信息量	17
3.10.2	信息熵	17
3.10.3	条件熵	17
3.10.4	Mutual Information	18
3.11	Batch Normalization	18
3.12	How Does Batch Normalization Help Optimization?	19
4	Optimization Algorithms	21
4.1	Challenges	21
4.1.1	Local Minima	21
4.1.2	Saddle Points	21
4.1.3	Vanishing gradients	22
4.1.4	noise / noise-free	22
4.2	Gradient Descent	22
4.2.1	Adaptive Methods	22
4.2.2	Stochastic Gradient Descent	23
4.2.3	Minibatch Stochastic Gradient Descent	23
4.3	SGDM - SGD with Momentum	24
4.4	NAG - Nesterov Accelerated Gradient	24
4.5	Adagrad	24
4.6	RMSProp	25
4.7	Adadelta	25
4.8	Adam	25
4.9	AdaMax	26
4.10	Nadam	26
4.11	AMSGrad	26
5	Models	27
6	Skip Connections	29
6.1	Residual Units	29
6.2	The Shattered Gradients Problem	30
6.3	DenseNet	31
6.4	FCN	31
6.5	UNet family	32
6.6	FCN	32
7	Semantic Segmentation	35
7.1	FCN - Fully Convolutional Network	35
8	Autoencoder	37
8.1	欠完备自编码器	37
8.2	正则自编码器	37
8.2.1	稀疏自编码器	37
8.2.2	DAE - Denoising Autoencoder	38

8.2.3 CAE - Contractive Autoencoder	38
9 Variational Autoencoder	39
9.1 Variational Inference	39
9.2 VAE	39
10 Generative Adversarial Nets	41
10.1 Mode collapse	41
10.2 Wasserstein GAN and the Kantorovich-Rubinstein Duality	41
10.3 GAN	41
10.4 CGAN	42
10.5 KLD - JSD - Wasserstein Distance	42

Chapter 1

Linear Algebra

1.1 特征值和特征空间

$$Au = \lambda u \quad (1.1)$$

特征值 λ 代表线性变化的伸缩倍数，特征向量 u 代表变换的方向

1.2 奇异值分解

定义 对于 $A \in C^{m \times n}$, $rank(A) = r$, 矩阵 $A^H A$ 的特征值为 $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_r > 0$, $\lambda_{r+1} = \lambda_{r+2} = \dots = \lambda_n = 0$, 称正数 $\sigma_i = \sqrt{\lambda_i} (i = 1, 2, \dots, r)$ 为矩阵 A 的**奇异值**

$$\begin{aligned} A &= U \Sigma V^H \\ &= (u_1 \ u_2 \ \dots \ u_m) \begin{pmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_r & & \\ & & & 0 & \end{pmatrix} \begin{pmatrix} v_1^H \\ v_2^H \\ \vdots \\ v_n^H \end{pmatrix} \\ &= \sigma_1 u_1 v_1^H + \sigma_2 u_2 v_2^H + \dots + \sigma_r u_r v_r^H \end{aligned} \quad (1.2)$$

1.2.1 降维和图像压缩

将图片作为矩阵进行奇异值分解，提取前 n 个奇异值，则可以达到图像压缩的目的.

Chapter 2

Statistics

2.1 Frequentist statistics VS Bayesian statistics

Throughout our subsequent discussions, we viewed θ as an unknown parameter of the world. This view of the θ as being constant-valued but unknown is taken in frequentist statistics. In the frequentist this view of the world, θ is not random—it just happens to be unknown—and it's our job to come up with statistical procedures (such as maximum likelihood) to try to estimate this parameter.

An alternative way to approach our parameter estimation problems is to take the Bayesian view of the world, and think of θ as being a random variable whose value is unknown. In this approach, we would specify a prior distribution $p(\theta)$ on θ that expresses our "prior beliefs" about the parameters. Given a training set $S = (x_i, y_i)$, make a prediction on a new value of x , we can then compute the posterior distribution on the parameters.(CS229)

2.2 期望

- 若干个随机变量之和的期望等于各变量的期望之和, $E(X_1 + X_2 + \dots + X_n) = E(X_1) + E(X_2) + \dots + E(X_n)$
- 若干个独立随机变量之和的期望等于各变量的期望之和, $E(X_1 X_2 \dots X_n) = E(X_1) E(X_2) \dots E(X_n)$

2.3 方差

均匀分布 $X \sim R(a, b)$ 其期望为

$$\begin{aligned} E(X) &= \int_a^b x f(x) dx \\ &= \frac{1}{b-a} \int_a^b x dx \\ &= \frac{1}{b-a} \frac{1}{2} (b^2 - a^2) \\ &= \frac{1}{2} (b + a) \end{aligned} \tag{2.1}$$

其方差为

$$\begin{aligned} \text{Var}(X) &= E(X - EX)^2 \\ &= E(X)^2 - (EX)^2 \\ &= \int_a^b x^2 f(x) dx - (EX)^2 \\ &= \frac{1}{3(b-a)}(b^3 - a^3) - \frac{1}{4}(b+a)^2 \\ &= \frac{1}{12}(b-a)^2 \end{aligned} \tag{2.2}$$

对于正态分布 $N(\mu, \sigma^2)$ ，其期望 $E(X) = \mu$ ，方差为 $\text{Var}(X) = \sigma^2$ 。

2.4 最小方差无偏估计

Chapter 3

Basic

3.1 Mechine Learning

- supervised learning
- semi-supervised learning
- unsupervised learning
- reinforcement learning
- active learning

3.2 iid

3.3 Universal approximation theorem

Any continuous function can be uniformly approximated by a continuous neural network having only one internal, hidden layer and with an arbitrary continuous sigmoidal nonlinearity.[2]

万能近似定理最初以sigmoidal激活函数来描述，后被证明对于更广泛的激活函数也适用[7]，包括ReLU。

3.4 Maximum likelihood estimation

Frequentist statistics

深度学习来就是用模型 $p_{model}(\mathbf{x}; \boldsymbol{\theta})$ 来估计数据的真实分布 $p_{data}(\mathbf{x}; \boldsymbol{\theta})$ ，对于一组确定的数据集 X ，在样本已被观察到的情况下，需要找到使得 $p_{model}(X; \boldsymbol{\theta})$ 出现可能性最大的一组参数 $\boldsymbol{\theta}$ ，也就是最大似然估计：

$$\begin{aligned}\hat{\boldsymbol{\theta}}_{MLE} &= \arg \max_{\boldsymbol{\theta}} p_{model}(X, \boldsymbol{\theta}) \\ &= \arg \max_{\boldsymbol{\theta}} \prod_{i=1}^m p_{model}(\mathbf{x}^i; \boldsymbol{\theta})\end{aligned}\tag{3.1}$$

等价于

$$\begin{aligned}\hat{\theta}_{MLE} &= \arg \max_{\theta} \sum_{i=0}^m \log p_{model}(\mathbf{x}^i; \theta) \\ &= \arg \min_{\theta} - \sum_{i=0}^m \log p_{model}(\mathbf{x}^i; \theta)\end{aligned}\quad (3.2)$$

第二行的优化函数也被称为Negative Log Likelihood(NLL)

除 m ，等价于

$$\hat{\theta}_{MLE} = \arg \max_{\theta} E_{\mathbf{x} \sim \hat{p}_{data}} \log p_{model}(\mathbf{x}^i; \theta) \quad (3.3)$$

3.4.1 KL divergence

KL散度用来衡量两种分布之间的差异，

$$\begin{aligned}D_{KL}(\hat{p}_{data} || p_{model}) &= \sum_i \hat{p}_{data}(x_i) \log \frac{\hat{p}_{data}(x_i)}{p_{model}(x_i)} \\ &= \sum_i \hat{p}_{data}(x_i) \log \hat{p}_{data}(x_i) - \sum_i \hat{p}_{data}(x_i) \log p_{model}(x_i)\end{aligned}\quad (3.4)$$

$\hat{p}_{data}(\mathbf{x})$ 与模型无关，其中

$$\sum_i \hat{p}_{data}(x_i) \ln \hat{p}_{data}(x_i) \quad (3.5)$$

为常量，则问题转换为

$$\begin{aligned}\arg \min D_{KL}(\hat{p}_{data} || p_{model}) &= \arg \min - \sum_i \hat{p}_{data}(x_i) \log p_{model}(x_i) \\ &= \arg \min - E_{\mathbf{x}} \log p_{model}(\mathbf{x})\end{aligned}\quad (3.6)$$

在本问题中，极大似然估计、最小化KL散度和最小化交叉熵等价。

3.4.2 熵、KL散度和交叉熵

- 熵: $H(X) = - \sum_i^n p(x_i) \log p(x_i)$ ，表示不确定程度，越不确定值越大
- KL散度(相对熵): $D_{KL}(p_{\theta} || p_{\hat{\theta}}) = \sum_i^n p_{\theta}(x_i) \log p_{\theta}(x_i) - \sum_i^n p_{\theta}(x_i) \log p_{\hat{\theta}}(x_i)$
- 交叉熵: $CE(X) = - \sum_i^n p_{\theta}(x_i) \log p_{\hat{\theta}}(x_i)$

从定义里可以看出，当熵为常量时，KL散度和交叉熵等价。

3.5 Maximum A Posteriori

$p(\theta)$ 先验概率分布， $p(\mathbf{X}|\theta)$ 是似然函数，根据贝叶斯定理，可用以下公式计算后验概率

$$p(\theta|\mathbf{X}) = \frac{p(\theta)p(\mathbf{X}|\theta)}{p(\mathbf{X})} \quad (3.7)$$

模型估计时，估计整个后验概率分布 $p(\theta|\mathbf{X})$ ，如需给出一个模型，通常取后验概率最大的模型。

$$\begin{aligned}\hat{\theta}_{MAP} &= \arg \max_{\theta} p(\theta|\mathbf{X}) \\ &= \arg \min - \log p(\theta|\mathbf{X}) \\ &= \arg \min - \log p(\mathbf{X}|\theta) - \log p(\theta) + \log p(\mathbf{X}) \\ &= \arg \min - \log p(\mathbf{X}|\theta) - \log p(\theta)\end{aligned}\quad (3.8)$$

$\log p(\mathbf{X})$ 与 θ 无关，可以丢掉。 $-\log p(\mathbf{X}|\theta)$ 其实就是NLL，所以MLE和MAP不同在 $-\log p(\theta)$ 。假定先验是一个高斯分布

$$p(\theta) = C \times e^{-\frac{\theta^2}{2\sigma^2}} \quad (3.9)$$

那么

$$-\log p(\theta) = C + \frac{\theta^2}{2\sigma^2} \quad (3.10)$$

在MAP中，使用一个高斯分布的先验等价于在MLE中采用 L^2 的正则化。MAP贝叶斯推断提供了一个直观的方法来设计复杂但可解释的正则化，更复杂的惩罚项可以通过混合高斯分布作为先验得到，而不是一个单独的高斯分布。

预测新的观察数据 x 时，计算数据对后验概率分布的期望值：

$$p(x|\mathbf{X}) = \int p(x|\theta, \mathbf{X})p(\theta|\mathbf{X})d\theta \quad (3.11)$$

3.6 Initialization scheme

3.6.1 constant initialization

3.6.2 random initialization

按照某一分布随机初始化

normal initialization

$$W \sim N(\mu, \sigma^2) \quad (3.12)$$

uniform initialization

$$W \sim U[-\frac{1}{\sqrt{n}}, \frac{1}{\sqrt{n}}] \quad (3.13)$$

3.6.3 xavier initialization

针对使用对称激活函数 $\tanh(x)$ 的网络进行参数初始化，对于ReLU激活函数并不适用[3].

Forward

对于一个卷积层来说

$$\begin{aligned} \mathbf{y}_l &= \mathbf{W}_l \mathbf{x}_l + \mathbf{b}_l \\ \mathbf{x}_l &= f(\mathbf{y}_{l-1}) \end{aligned} \quad (3.14)$$

在如下前提和假设下

- 初始化 \mathbf{W}_l 元素为独立同分布
- 假设 \mathbf{x}_l 元素也为独立同分布
- $\mathbf{w}_l, \mathbf{x}_l$ 互相独立

则有

$$\begin{aligned} \text{Var}(\mathbf{y}_l) &= \text{Var}\left(\sum \mathbf{w}_l \mathbf{x}_l + \mathbf{b}_l\right) \\ &= \text{Var}\left(\sum \mathbf{w}_l \mathbf{x}_l\right) \\ &= n_l \text{Var}(\mathbf{w}_l \mathbf{x}_l) \end{aligned} \quad (3.15)$$

令 \mathbf{w}_l 期望为0, $E(\mathbf{w}_l) = 0$, $\text{Var}(\mathbf{w}_l) = E(\mathbf{w}_l - E(\mathbf{w}_l))^2 = E\mathbf{w}_l^2$, 则

$$\begin{aligned} \text{Var}(\mathbf{y}_l) &= n_l E(\mathbf{w}_l^2 \mathbf{x}_l^2) - n_l E^2 \mathbf{w}_l E^2 \mathbf{x}_l \\ &= n_l E(\mathbf{w}_l^2 \mathbf{x}_l^2) \\ &= n_l \text{Var}(\mathbf{w}_l) E(\mathbf{x}_l^2) \end{aligned} \quad (3.16)$$

若 $E\mathbf{x}_l = 0$, 则

$$\text{Var}(\mathbf{y}_l) = n_l \text{Var}(\mathbf{w}_l) \text{Var}(\mathbf{x}_l) \quad (3.17)$$

若要实现 $\text{Var}(\mathbf{y}_l) = \text{Var}(\mathbf{x}_l)$, 则需要满足 $n_l \text{Var}(\mathbf{w}_l) = 1$, 即

$$\text{Var}(\mathbf{w}_l) = \frac{1}{n_l} \quad (3.18)$$

- 若 \mathbf{w}_l 服从正态分布, 则 $\mathbf{w}_l \sim N(0, \frac{1}{n_l})$
- 若 \mathbf{w}_l 服从均匀分布, 则 $\mathbf{w}_l \sim U(-\sqrt{\frac{3}{n_l}}, \sqrt{\frac{3}{n_l}})$

Backword

反向传播过程中, 需要保证梯度的方差不变, 每一层的梯度为:

$$\Delta \mathbf{x}_l = \hat{\mathbf{W}}_l \Delta \mathbf{y}_l \quad (3.19)$$

假设

- \mathbf{w}_l 和 $\Delta \mathbf{y}_l$ 互相独立
- $E\mathbf{w}_l = 0, E\Delta \mathbf{x}_l = 0$

同前向传播, 可得

$$\text{Var}(\Delta \mathbf{x}_l) = \hat{n}_l \text{Var}(\mathbf{w}_l) \text{Var}(\Delta \mathbf{y}_l) \quad (3.20)$$

- 若 \mathbf{w}_l 服从正态分布, 则 $\mathbf{w}_l \sim N(0, \frac{1}{\hat{n}_l})$
- 若 \mathbf{w}_l 服从均匀分布, 则 $\mathbf{w}_l \sim U(-\sqrt{\frac{3}{\hat{n}_l}}, \sqrt{\frac{3}{\hat{n}_l}})$

除非 $n = \hat{n}_l$, 否则同时保证信号在向前向后传播时的Var不变, 取调和平均数, $\text{Var}(\mathbf{w}_l) = \frac{2}{n_l + \hat{n}_l}$, 可得

- Normal distribution: $\mathbf{w}_l \sim N(0, \frac{2}{n_l + \hat{n}_l})$
- Uniform distribution: $\mathbf{w}_l \sim U(-\sqrt{\frac{6}{n_l + \hat{n}_l}}, \sqrt{\frac{6}{n_l + \hat{n}_l}})$

3.6.4 orthogonal initialization

3.6.5 kaiming initialization

Xavier 针对对称激活函数的层权重初始化进行设计，但对于使用ReLU激活函数的层并不适用[4]。
-> 期望为0，方差为1

对ReLU层来说， $E(\mathbf{x}_l^2) = \frac{1}{2}Var(\mathbf{y}_l)$ ，因此

$$\begin{aligned} Var(\mathbf{y}_l) &= \frac{1}{2}n_l Var(\mathbf{w}_l) Var(\mathbf{x}_l) \\ Var(\Delta \mathbf{x}_l) &= \frac{1}{2}\hat{n}_l Var(\mathbf{w}_l) Var(\Delta \mathbf{y}_l) \end{aligned} \quad (3.21)$$

且在权重初始化时，使用上述任意一个即可。

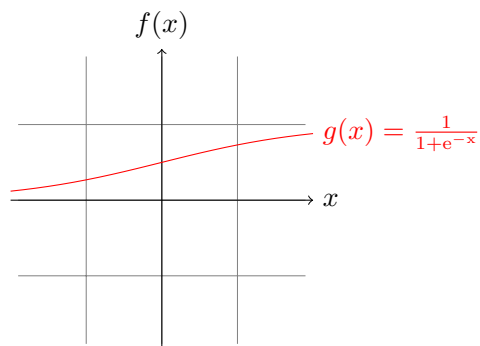
3.7 正则化

3.8 Data preprocessing

3.8.1 whitening

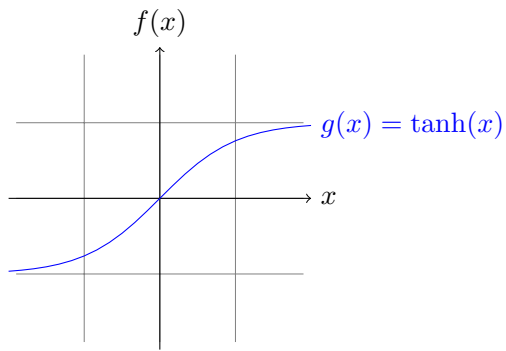
3.9 Activation functions

3.9.1 sigmoid



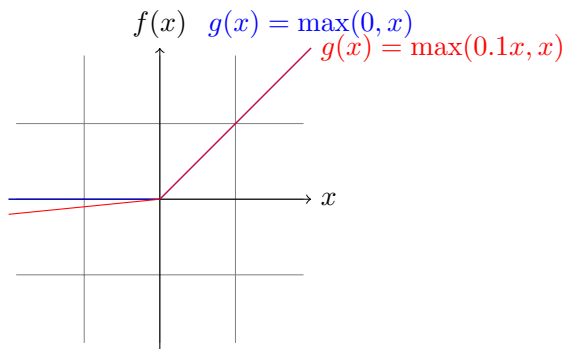
$$\begin{aligned} g(x) &= \frac{1}{1 + e^{-x}} \\ g(x)' &= g(x)(1 - g(x)) \end{aligned} \quad (3.22)$$

3.9.2 $\tanh(x)$



$$\begin{aligned}
 g(x) &= \tanh(x) \\
 g(x)' &= 1 - g(x)^2 \\
 &= 1 - \tanh(x)^2
 \end{aligned}
 \tag{3.23}$$

3.9.3 Rectified linear units



ReLU

$$g(x) = \max(0, x) \tag{3.24}$$

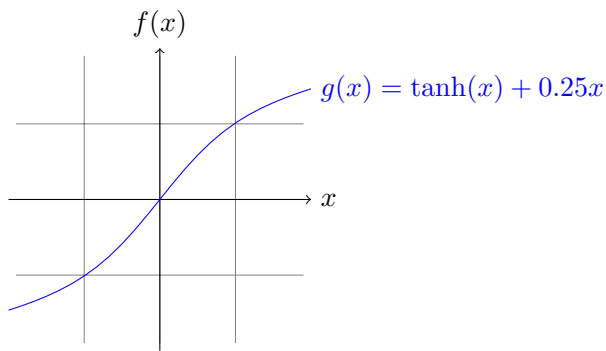
Leaky ReLU

$$g(x) = \max(0.01x, x) \tag{3.25}$$

PReLU

$$g(x) = \max(\alpha x, x) \tag{3.26}$$

3.9.4 haha



$$\begin{aligned}
 g(x) &= \tanh(x) + 0.25x \\
 g(x)' &= 0.75 - g(x)^2 \\
 &= 0.75 - \tanh(x)^2
 \end{aligned} \tag{3.27}$$

3.9.5 Loss functions

3.10 信息熵

3.10.1 信息量

信息量和具体的事件相关，且越小概率的事件发生产生的信息量越大。因此，一个具体事件的信息量应该是随着其发生概率而递减的，且不能为负。如果有俩个不相关的事件 x 和 y ，那么我们观察到的俩个独立事件同时发生时获得的信息应该等于观察到的事件各自发生时获得的信息之和，即为：

$$\begin{aligned}
 h(x, y) &= h(x) + h(y) \\
 p(x, y) &= p(x) + p(y)
 \end{aligned} \tag{3.28}$$

为满足以上性质，信息量 $h(x)$ 公式为下

$$h(x) = -\log_2 p(x) \tag{3.29}$$

3.10.2 信息熵

信息熵即为随机变量的信息量的期望

$$H(x) = -\sum_i p(x_i) \log_2 p(x_i) \tag{3.30}$$

3.10.3 条件熵

$$H(X|Y) = -\sum_{i,j} p(x_i, y_j) \log \frac{p(x_i, y_j)}{p(y_j)} \tag{3.31}$$

3.10.4 Mutual Information

互信息是衡量随机变量之间互相依赖程度的度量。

$$\begin{aligned} I(X; Y) &= H(X) - H(X|Y) \\ &= \sum_y \sum_x p(x, y) \log\left(\frac{p(x, y)}{p(x)p(y)}\right) \end{aligned} \quad (3.32)$$

3.11 Batch Normalization

机器学习中假设：源空间和目标空间的数据分布是一致的，如果不一致，就是新的机器学习问题，例如transfer learning/domain adaptation 等。而covariate shift就是分布不一致假设下的一个分支问题，它指的是源空间和目标空间的条件概率是一致的，但是其边缘概率不同。对于所有 $x \in \chi$

$$\begin{aligned} P_s(Y|X=x) &= P_t(Y|X=x) \\ P_s(X) &\neq P_t(X) \end{aligned} \quad (3.33)$$

Internal Covariate Shift Training Deep Neural Networks is complicated by the fact that the distribution of each layer's inputs changes during training, as the parameters of the previous layers change. And this slows down the training by requiring lower lr and careful parameter initialization, and makes it notoriously hard to train models with saturating nonlinearities.

解决方案：白化操作可以使得数据变为独立同分布

By whitening the inputs to each layer, we would take a step towards achieving the fixed distribution of inputs that would remove the ill effects of the ICS.

但是每层的白化会影响梯度下降优化过程。每层进行白化计算量过大。保证模型的表达能力不因为规范化而下降。

Simply normalizaing each input of a layer may change what the layer can represent. So, we make sure that the transformation inserted in the network can represent the identity transform.

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_1 \dots x_m\}$;	
Parameters to be learned: γ, β	
Output: $\{y_i = \text{BN}_{\gamma, \beta}(x_i)\}$	
$\mu_{\mathcal{B}} \leftarrow \frac{1}{m} \sum_{i=1}^m x_i$	// mini-batch mean
$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m} \sum_{i=1}^m (x_i - \mu_{\mathcal{B}})^2$	// mini-batch variance
$\hat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}}$	// normalize
$y_i \leftarrow \gamma \hat{x}_i + \beta \equiv \text{BN}_{\gamma, \beta}(x_i)$	// scale and shift

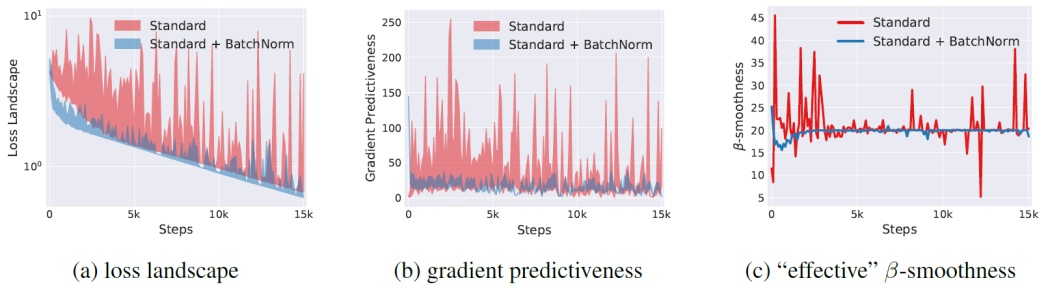
Algorithm 1: Batch Normalizing Transform, applied to activation x over a mini-batch.

图 3.1: Batch Normalization

3.12 How Does Batch Normalization Help Optimization?

Batch Normalization 对减少ICS起到了很少的作用, it makes the optimization landscape significantly smoother.

- **Is the effectiveness of BatchNorm indeed related to internal covariate shift?** The ‘noisy’ BatchNorm network has qualitatively less stable distributions than even the standard, non-BatchNorm network, yet it still better performs better in terms of training.
- **Is BatchNorm reducing internal covariate shift?** We Observe that networks with BatchNorm often exhibit an increase in ICS.



Why does BatchNorm work?

BatchNorm reparametrizes the underlying optimization problem to make its landscape significantly more smooth. The key implication of BatchNorm’s reparametrization is that it makes the gradients more reliable and predictive. After all, improved Lipschitzness of the gradients gives us confidence that when we take a larger step in a direction of a computed gradient, this gradient direction remains a fairly accurate estimate of the actual gradient direction after taking that step.

Is BatchNorm the best way to smoothen the landscape?

In fact, for deep linear networks, l_1 -normalization performs even better than BatchNorm. Note that, qualitatively, the l_p -normalization techniques lead to larger distributional shift than the vanilla, i.e., unnormalized, networks, yet they still yield improved optimization performance.

Chapter 4

Optimization Algorithms

4.1 Challenges

4.1.1 Local Minima

When the numerical solution of an optimization problem is near the local optimum, the numerical solution obtained by the final iteration may only minimize the objective function locally, rather than globally, as the gradient of the objective function's solutions approaches or becomes zero. *Only some degree of noise might knock the parameter out of the local minimum. In fact, this is the one of the beneficial properties of stochastic gradient descent where the natural variation of gradients over minibatches is able to dislodge the parameters from local minima.*[8]

4.1.2 Saddle Points

We assume that the input of a function is k -dimensional vector and its output is a scalar, so its *Hessian matrix* will have k eigenvalues. The solution of the function could be a local minimum, a local maximum or a saddle point at a position where the function gradient is zero:

- When the eigenvalues of the function's Hessian matrix at the zero-gradient position

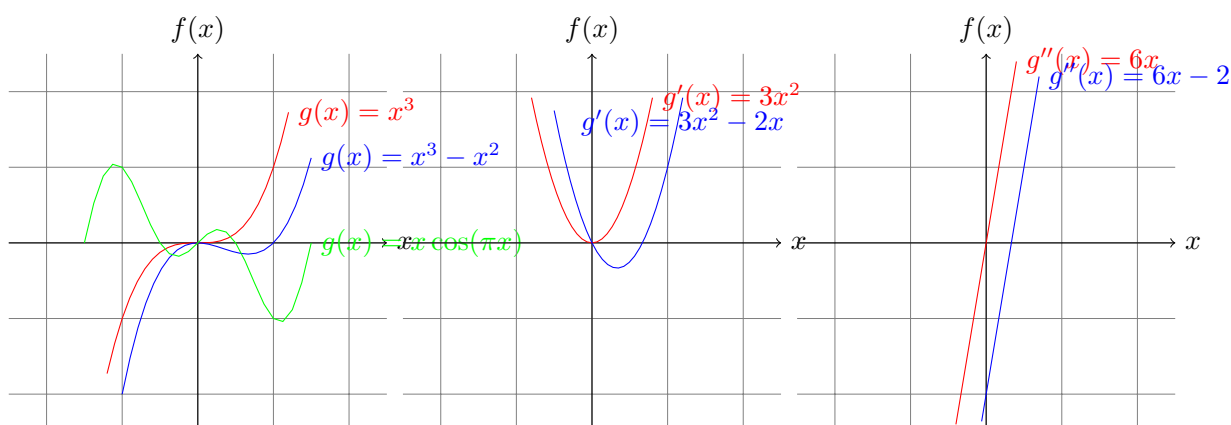


图 4.1: Challenges

are all positive, we have a local minimum for the function.

- When the eigenvalues of the function's Hessian matrix at the zero-gradient position are all negative, we have a local maximum for the function.
- When the eigenvalues of the function's Hessian matrix at the zero-gradient position are negative and positive, we have a saddle point for the function.
- 同号且至少有一个为0, 不确定

For high-dimensional problem the likelihood that at least some of the eigenvalues are negative is quite high. This makes saddle points are more likely than local minima.[8]

4.1.3 Vanishing gradients

Vanishing gradients can cause optimization to stall. Often a reparameterization of the problem helps. Good initialization of the parameters can be beneficial, too.[8]

4.1.4 noise / noise-free

noise对优化过程中saddle point的影响? what is noise? and how it effect the optimization. 无偏估计、一致性

4.2 Gradient Descent

Using a Taylor expansion we obtain that:

$$f(x + \epsilon) = f(x) + \epsilon f'(x) + \mathcal{O}(\epsilon^2) \quad (4.1)$$

For small ϵ moving in *the direction of negative gradient* will decrease f . Choose $\epsilon = -\eta f'(x)$, $\eta > 0$, then we get:

$$f(x - \eta f'(x)) = f(x) - \eta f'^2(x) + \mathcal{O}((\eta f'(x))^2) \quad (4.2)$$

Choose η small enough for the higher order terms to become irrelevant. then

$$f(x - \eta f'(x)) \lesssim f(x) \quad (4.3)$$

that means, if we use

$$x \leftarrow x - \eta f'(x) \quad (4.4)$$

to iterate x , the value of $f(x)$ decline.

4.2.1 Adaptive Methods

Getting the learning rate 'just right' is tricky. What if we could determine η automatically or get rid of having to select a step size at all? Second order methods that look not only at the value and gradient of the objective but also at its *curvature* can help in this case. These methods cannot be applied to deep learning directly due to the computational cost.

Newton's Method

当Taylor expansion展开到二阶导数时:

$$f(\mathbf{x} + \epsilon) = f(\mathbf{x}) + \epsilon^\top \nabla f(\mathbf{x}) + \frac{1}{2} \epsilon^\top H_f \epsilon + \mathcal{O}(\|\epsilon\|^3) \quad (4.5)$$

we define $H_f := \nabla \nabla^\top f(\mathbf{x})$ to be the *Hession* of f . H is a $d \times d$ matrix and may be prohibitively large, due to the cost of storing $\mathcal{O}(d^2)$ entries.

After all, the minimum of f statisfies $\nabla f(\mathbf{x}) = 0$. Taking derivatives of above equataion with regard to η and ignoring higher order terms we arrive at

$$\begin{aligned} \nabla f(\mathbf{x}) + H_f \epsilon &= 0 \\ \epsilon &= -H_f^{-1} \nabla f(\mathbf{x}) \end{aligned} \quad (4.6)$$

then, $\epsilon = -\eta \nabla f(\mathbf{x})$, we get

$$\eta = -H_f^{-1} \quad (4.7)$$

For $f(x) = (x - 2)(x - 4) = x^2 - 6x + 8$, $f'(x) = 2x - 6$, $f''(x) = 2$, then

$$\epsilon = -f''(0)^{-1} f'(0) = -\frac{1}{2} \times -6 = 3 \quad (4.8)$$

Hessian

Hession 矩阵是是对称的，可以表示为一组特征值和一组特征向量的正交基底，在特定方向上 g 上的二阶导数为 $g^\top H g$ ，当 g 为特征向量时，这个二阶导数就是对应的特征值。最大特征值确定最大二阶导数，最小特征值确定最小导数。

在 g 方向上的learning rate为

$$\eta_g = \frac{1}{g^\top H_f g} \quad (4.9)$$

最差的情况下， g 与 H 最大的特征值 λ_{max} 对应的特征向量对齐，此时的最优步长为 $\frac{1}{\lambda_{max}}$ ，当要最小化的目标函数能用二次函数很好近似的情况下，Hession的特征值决定了学习率的量级。

4.2.2 Stochastic Gradient Descent

Dynamic Learning rate

$$\begin{aligned} \eta(t) &= \eta_i \text{ if } t_i \leq t \leq t_{i+1} && \text{piecewise constant} \\ \eta(t) &= \eta_0 e^{-\lambda t} && \text{exponential} \\ \eta(t) &= \eta_0 (\beta t + 1)^{-\alpha} && \text{polynomial} \end{aligned} \quad (4.10)$$

In the case of convex optimization there are a number of proofs which show that this rate is well behaved.

4.2.3 Minibatch Stochastic Gradient Descent

At each iteration of stochastic gradient descent, we uniformly sample an index $i \in 1, \dots, n$ for data examples at random, and compute the gradient $\nabla f_i(\mathbf{x})$ to update \mathbf{x} :

$$\mathbf{x} \leftarrow \mathbf{x} - \eta \nabla f_i(\mathbf{x}) \quad (4.11)$$

The stochastic gradient $\nabla f_i(\mathbf{x})$ is the *unbiased estimate* of gradient $\nabla f(\mathbf{x})$. That means the stochastic gradient is a good estimate of the gradient.

$$\mathbb{E}_i \nabla f_i(\mathbf{x}) = \frac{1}{n} \sum_{i=1}^n \nabla f_i(\mathbf{x}) = \nabla f(\mathbf{x}) \quad (4.12)$$

4.3 SGDM - SGD with Momentum

$$\begin{aligned} \mathbf{v}_t &\leftarrow \beta \mathbf{v}_{t-1} + \nabla f(\mathbf{x}) \\ \mathbf{x}_t &\leftarrow \mathbf{x}_{t-1} - \eta_t \mathbf{v}_t \end{aligned} \quad (4.13)$$

\mathbf{v} is called *momentum*. Momentum replaces gradients with a leaky average over past gradients. This accelerates convergence significantly. And it prevents stalling of the optimization process that is much more likely to occur for stochastic gradient descent.

An Ill-conditioned Problem

$$f(\mathbf{x}) = 0.1x_1^2 + 2x_2^2 \quad (4.14)$$

and we know that f has minimum at $(0, 0)$. Then we get

$$\nabla f(\mathbf{x}) = \begin{bmatrix} 0.2x_1 \\ 4x_2 \end{bmatrix} \quad (4.15)$$

let's start at $(2, 2)$, $\eta = 0.4$, $\beta = 0.6$.

4.4 NAG - Nesterov Accelerated Gradient

$$\begin{aligned} \mathbf{v}_t &= \alpha \mathbf{v}_{t-1} + \eta \nabla_{\theta} J(\theta - \alpha \mathbf{v}_{t-1}) \\ \theta &= \theta - \mathbf{v}_t \end{aligned} \quad (4.16)$$

4.5 Adagrad

Consider a model training on sparse features, i.e., features that occur only infrequently. Parameters associated with infrequent features only receive meaningful updates whenever these features occur. And to get good accuracy we want to decrease the lr as well keep on training, usually at a lr of $\mathcal{O}(t^{-\frac{1}{2}})$. we might end up in situation where the parameters from common features converge rather quickly to their optimal values.

$$\begin{aligned} \mathbf{g}_t &= \partial_{\mathbf{w}} l(y_t, f(\mathbf{x}_t, \mathbf{w})) \\ \mathbf{s}_t &= \mathbf{s}_{t-1} + \mathbf{g}_t^2 \\ \mathbf{w}_t &= \mathbf{w}_{t-1} - \frac{\eta}{\sqrt{\mathbf{s}_t + \epsilon}} \cdot \mathbf{g}_t \end{aligned} \quad (4.17)$$

learning rate不再简单的是时间的函数，而是和梯度相关，lr的减小速度和梯度正相关，每个维度都有自己的learning rate.

- It uses the magnitude of the gradient as a means of adjusting how quickly progress is achieved - coordinates with large gradients are compensated with a smaller learning rate.
- On deep learning problems Adagrad can sometimes be too aggressive in reducing learning rates

4.6 RMSProp

A variant of AdaGrad.

As a result s_t keeps on growing without bound due to the lack of normalization, essentially linearly as the algorithm converges.

$$\begin{aligned} s_t &\leftarrow \gamma s_{t-1} + (1 - \gamma) \mathbf{g}_t^2 \\ \mathbf{x}_t &\leftarrow \mathbf{x}_{t-1} - \frac{\eta}{\sqrt{s_t + \epsilon}} \odot \mathbf{g}_t \end{aligned} \quad (4.18)$$

$$\begin{aligned} s_t &= (1 - \gamma) \mathbf{g}_t^2 + \gamma s_{t-1} \\ &= (1 - \gamma) (\mathbf{g}_t^2 + \gamma \mathbf{g}_{t-1}^2 + \gamma^2 \mathbf{g}_{t-2}^2 + \dots) \end{aligned} \quad (4.19)$$

4.7 Adadelta

A variant of AdaGrad.

$$\begin{aligned} s_t &= \rho s_{t-1} + (1 - \rho) \mathbf{g}_t^2 \\ \mathbf{g}'_t &= \frac{\sqrt{\Delta \mathbf{x}_{t-1} + \epsilon}}{\sqrt{s_t + \epsilon}} \odot \mathbf{g}_t \\ \Delta \mathbf{x}_t &= \rho \Delta \mathbf{x}_{t-1} + (1 - \rho) \mathbf{g}'_t{}^2 \\ \mathbf{x}_t &= \mathbf{x}_{t-1} - \mathbf{g}'_t \end{aligned} \quad (4.20)$$

4.8 Adam

$$\begin{aligned} \mathbf{v}_t &\leftarrow \beta_1 \mathbf{v}_{t-1} + (1 - \beta_1) \mathbf{g}_t \\ s_t &\leftarrow \beta_2 s_{t-1} + (1 - \beta_2) \mathbf{g}_t^2 \end{aligned} \quad (4.21)$$

Common choices for them are $\beta_1 = 0.9$ and $\beta_2 = 0.9999$. That is, the variance estimate moves *much more slowly* than the momentum term. Note that if we initialize $\mathbf{v}_0 = \mathbf{s}_0 = 0$ we have a significant amount of bias initially towards smaller values. This can be addressed by using the fact that $\sum_{i=0}^t = \frac{1 - \beta^t}{1 - \beta}$ to re-normalize terms.

$$\begin{aligned} \hat{\mathbf{v}}_t &= \frac{\mathbf{v}_t}{1 - \beta_1^t} \\ \hat{s}_t &= \frac{s_t}{1 - \beta_2^t} \\ \mathbf{x}_t &\leftarrow \mathbf{x}_{t-1} - \frac{\eta \hat{\mathbf{v}}_t}{\sqrt{\hat{s}_t + \epsilon}} \end{aligned} \quad (4.22)$$

4.9 AdaMax

4.10 Nadam

4.11 AMSGrad

Chapter 5

Models

Chapter 6

Skip Connections

6.1 Residual Units

为了训练deeper model, 并解决梯度消失/爆炸以及网络degradation problem, 在ResNets[5]中提出, 如果在shallower model 中添加identity mapping, 那么deeper model理应不会有更高的错误率。

Residual unit表示如下:

$$\begin{aligned} \mathbf{y}_l &= h(\mathbf{x}_l) + \mathcal{F}(\mathbf{x}_l, \mathcal{W}_l) \\ \mathbf{x}_{l+1} &= f(\mathbf{y}_l) \end{aligned} \tag{6.1}$$

In ResNet, $h(\mathbf{x}) = \mathbf{x}_l$ is an identity mapping and f is a ReLU function.

在ResNet中, 因为 f 不是identity mapping, 所以残差只能ResNet units中学习且信息不能直接传达到后面的层中, In [6], 希望propagating information可以through the entire network.

Our derivations reveal that if both $h(\mathbf{x}_l)$ and $f(\mathbf{y}_l)$ are identity mappings, the signal could be directly propagated from one unit to any other units, in both forward and backward passes.[6]

if f is also an identity mapping:

$$\mathbf{x}_{l+1} = \mathbf{x}_l + \mathcal{F}(\mathbf{x}_l, \mathbf{W}_l) \tag{6.2}$$

then we will have:

$$\begin{aligned} \mathbf{x}_{l+n} &= \mathbf{x}_{l+n-1} + \mathcal{F}(\mathbf{x}_{l+n-1}, \mathcal{W}_{l+n-1}) \\ &= \mathbf{x}_{l+n-2} + \mathcal{F}(\mathbf{x}_{l+n-2}, \mathcal{W}_{l+n-2}) + \mathcal{F}(\mathbf{x}_{l+n-1}, \mathcal{W}_{l+n-1}) \\ &= \mathbf{x}_l + \sum_{i=l}^{l+n-1} \mathcal{F}(\mathbf{x}_i, \mathcal{W}_i) \end{aligned} \tag{6.3}$$

相较于plain network(ignoring BN and ReLU is an identity mapping):

$$\mathbf{x}_{l+n} = \prod_{i=l}^{l+n-1} \mathbf{W}_i \mathbf{x}_l \tag{6.4}$$

反向传播过程:

$$\begin{aligned}\frac{\partial \mathcal{E}}{\partial \mathbf{x}_l} &= \frac{\partial \mathcal{E}}{\partial \mathbf{x}_{l+n}} \frac{\partial \mathbf{x}_{l+n}}{\partial \mathbf{x}_l} \\ &= \frac{\partial \mathcal{E}}{\partial \mathbf{x}_{l+n}} \left(1 + \frac{\partial}{\partial \mathbf{x}_l} \sum_{i=l}^{l+n-1} \mathcal{F}(\mathbf{x}_i, \mathcal{W}_i) \right)\end{aligned}\quad (6.5)$$

可以看出, 上式可以分为两部分, $\frac{\partial \mathcal{E}}{\partial \mathbf{x}_{l+n}}$ 不经过任何权重信息, 可以直接传播到浅层, 只要括号中的第二部分不总是为-1, 那么即使权重任意小也不会发生梯度消失。

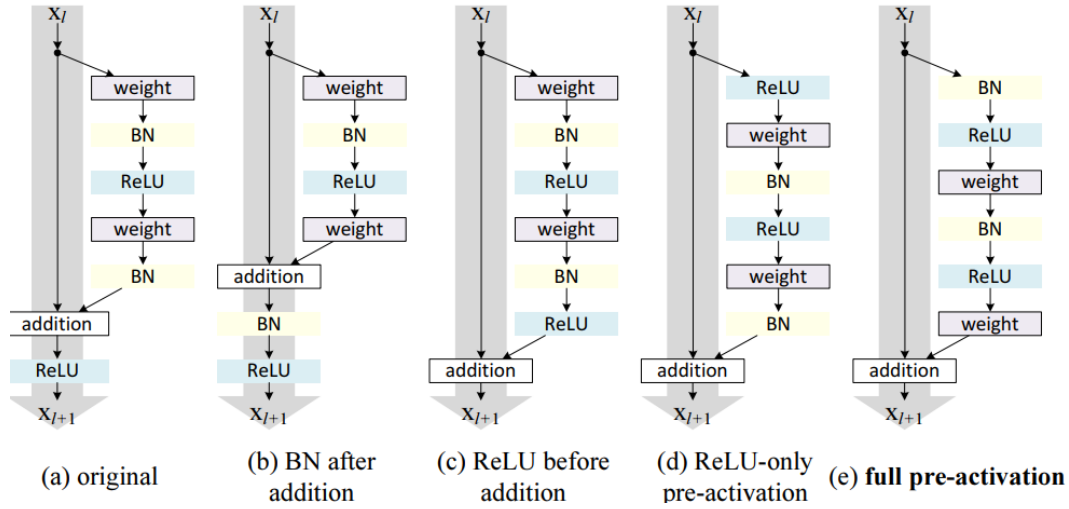


图 6.1: Residual Units

以上的分析基于 f 是一个恒等变换, 但实际上 f 会影响之前分析的两条信息传播的路径:

$$\mathbf{x}_{l+1} = f(\mathbf{x}_l) + \mathcal{F}(f(\mathbf{x}_l), \mathcal{W}_l) \quad (6.6)$$

因此, 在[6]提出了新的Residual unit结构pre-activation, 等式变为

$$\mathbf{x}_{l+1} = \mathbf{x}_l + \mathcal{F}(\hat{f}(\mathbf{x}_l), \mathcal{W}_l) \quad (6.7)$$

6.2 The Shattered Gradients Problem

A previously unnoticed difficulty with gradients in deep rectifier networks that orthogonal to vanishing and exploding gradients. The shattering gradients problem is that, as depth increases, **gradients in standard feedforward networks increasingly resemble white noise**[1].

- Gradients of shallow networks resemble brown noise(布朗噪声).
- Gradients of deep networks resemble white noise(白噪声).
- Training is difficult when gradients behave like white noise.
- Gradients of deep resnets lie in between brown and white noise.

在标准前馈神经网络中, 神经元相关性按指数级减少($\frac{1}{2^L}$), 同时, 梯度的空间结构也随着深度增加被逐渐消除。使用BatchNorm的ResNets中梯度相关系数减少的速度从指数级减少到亚线性级($\frac{1}{\sqrt{L}}$), 极大的保留梯度的空间结构。

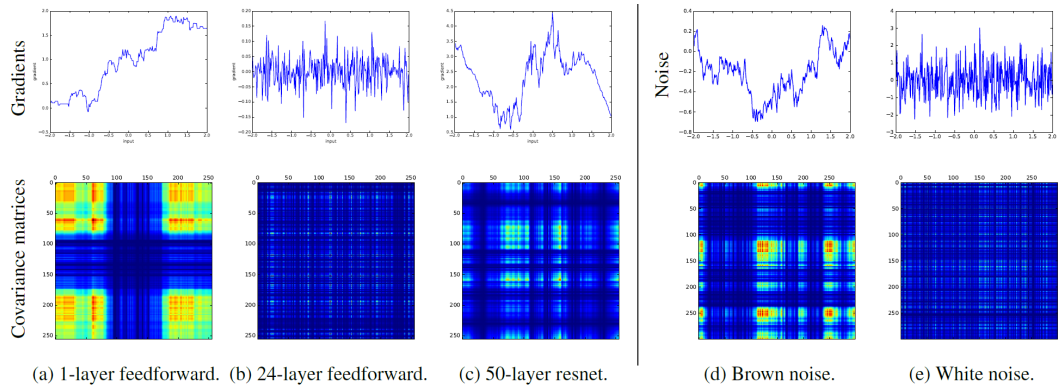


图 6.2: The shattered Gradients Problem

6.3 DenseNet

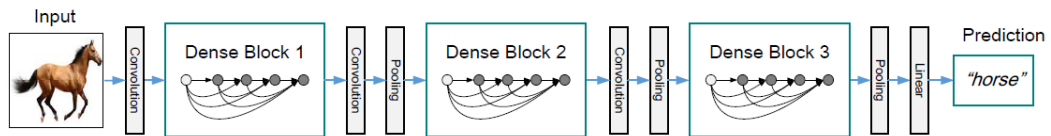


图 6.3: DenseNet

6.4 FCN

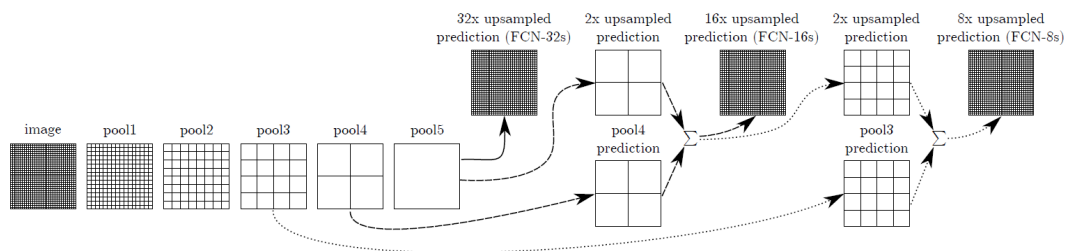


图 6.4: FCN

6.5 UNet family

6.6 FCN

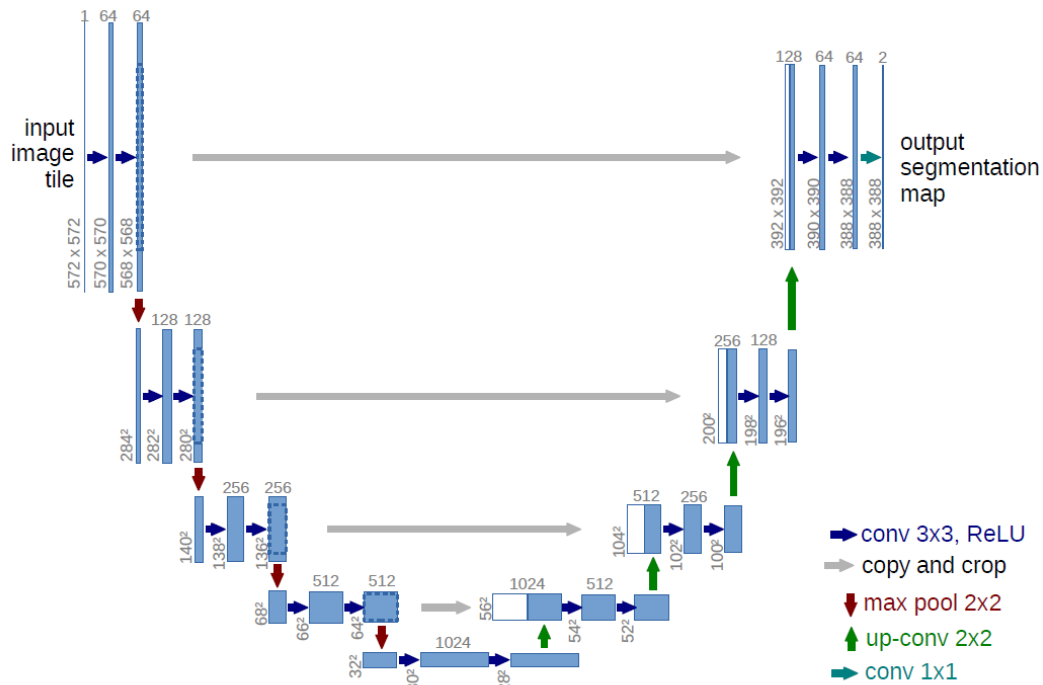


图 6.5: UNet

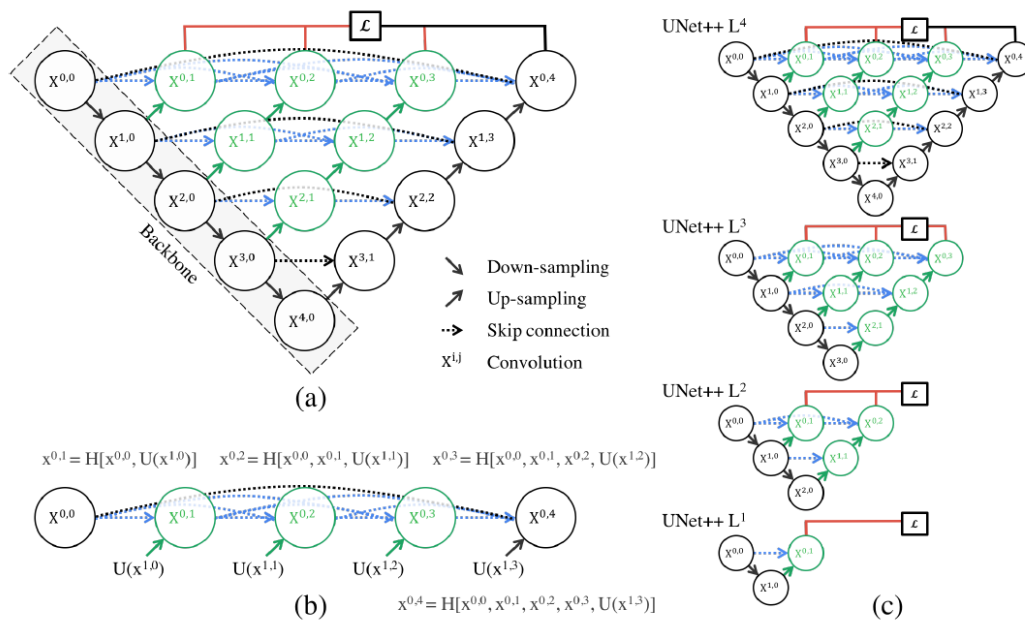


图 6.6: UNet 2+

多深合适？降采样对分割网络到底是不是必须的？不一定要降到第四次才上采样，使用浅层和深层的特征。

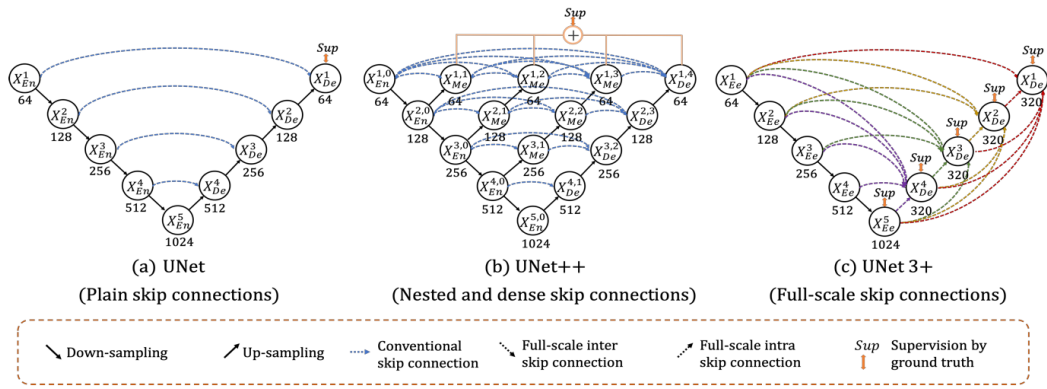


图 6.7: UNet 3+

都缺乏从全尺度探索足够信息的能力，未能明确了解器官的位置和边界。UNet 3+中的每一个解码器层都融合了来自编码器中的小尺度和同尺度的特征图，以及来自解码器的大尺度的特征图，这些特征图捕获了全尺度下的细粒度语义和粗粒度语义。

Chapter 7

Semantic Segmentation

7.1 FCN - Fully Convolutional Network

Chapter 8

Autoencoder

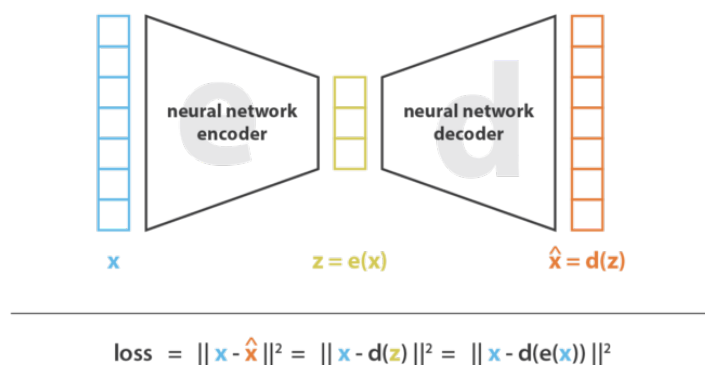


图 8.1: Autoencoder

8.1 欠完备自编码器

编码维度小于输入维度。学习欠完备的表示将强制自编码器捕捉训练数据中最显著的特征。

8.2 正则自编码器

如果隐藏层的编码的维度允许与输入相等，或隐藏编码维数大于输入的过完备的情况下，会学习将输入复制到输出，而学不到任何有关数据分布的有用信息。正则自编码器使用的损失函数可以鼓励模型学习其他特性（除了将输入复制到输出），而不必限制使用浅层的编码器和解码器以及小的编码维度来限制模型的容量。这些特性包括稀疏表示、表示的小导数以及对噪声或输入缺失的鲁棒性，即使模型容量大到足以学习一个无意义的恒等函数，非线性且过完备的正则自编码器仍然能够从数据中学到一些关于数据分布的有用信息。

8.2.1 稀疏自编码器

$$L(\mathbf{x}, g(f(\mathbf{x}))) + \Omega(h) \tag{8.1}$$

8.2.2 DAE - Denoising Autoencoder

$$L(\mathbf{x}, g(f(\tilde{\mathbf{x}}))) \quad (8.2)$$

8.2.3 CAE - Contractive Autoencoder

$$L(\mathbf{x}, g(f(\mathbf{x}))) + \Omega(\mathbf{h}, \mathbf{x})$$

$$\Omega(\mathbf{h}, \mathbf{x}) = \lambda \sum_i \|\nabla_x h_i\|^2 \quad (8.3)$$

Chapter 9

Variational Autoencoder

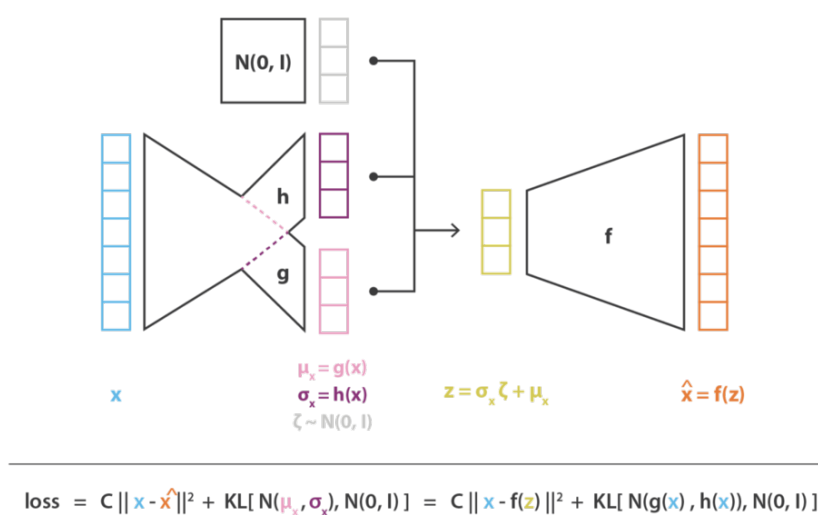


图 9.1: vae

9.1 Variational Inference

9.2 VAE

Chapter 10

Generative Adversarial Nets

10.1 Mode collapse

<https://aiden.nibali.org/blog/2017-01-18-mode-collapse-gans/>

10.2 Wasserstein GAN and the Kantorovich-Rubinstein Duality

<https://vincentherrmann.github.io/blog/wasserstein/>

10.3 GAN

$$\min_G \max_D V(D, G) = E_{\mathbf{x} \sim p_{data}(\mathbf{x})} [\log D(\mathbf{x})] + E_{\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})} [\log(1 - D(G(\mathbf{z})))] \quad (10.1)$$

The training criterion for the discriminator D , given any generator G , is to maximize the quantity $V(G, D)$

$$\begin{aligned} V(G, D) &= \int_{\mathbf{x}} p_{data}(\mathbf{x}) \log(D(\mathbf{x})) d\mathbf{x} \int_{\mathbf{z}} p_{\mathbf{z}}(\mathbf{z}) \log(1 - D(G(\mathbf{z}))) d\mathbf{z} \\ &= \int_{\mathbf{x}} p_{data}(\mathbf{x}) \log(D(\mathbf{x})) + p_g(\mathbf{x}) \log(1 - D(\mathbf{x})) d\mathbf{x} \end{aligned} \quad (10.2)$$

For any $(a, b) \in \mathbb{R}^2 \setminus \{0, 0\}$, the function $y \rightarrow a \log(y) + b \log(1 - y)$ achieves its maximum in $[0, 1]$ at $\frac{a}{a+b}$. So, for G fixed, the optimal discriminator D is

$$D_G^*(\mathbf{x}) = \frac{p_{data}(\mathbf{x})}{p_{data}(\mathbf{x}) + p_g(\mathbf{x})} \quad (10.3)$$

Note that the training objective for D can be interpreted as maximizing the log-likelihood for estimating the conditional probability $P(Y = y|\mathbf{x})$, $\mathbf{x} \in p_{data}$ or $\mathbf{x} \in p_g$, then

$$\begin{aligned} C(G) &= \max_D V(G, D) \\ &= E_{\mathbf{x} \sim p_{data}} [\log D_G^*(\mathbf{x})] + E_{\mathbf{z} \sim p_{\mathbf{z}}} [\log(1 - D_G^*(G(\mathbf{z})))] \\ &= E_{\mathbf{x} \sim p_{data}} [\log D_G^*(\mathbf{x})] + E_{\mathbf{x} \sim p_g} [\log D_G^*(\mathbf{x})] \\ &= E_{\mathbf{x} \sim p_{data}} [\log \frac{p_{data}(\mathbf{x})}{p_{data}(\mathbf{x}) + p_g(\mathbf{x})}] + E_{\mathbf{x} \sim p_g} [\log \frac{p_g(\mathbf{x})}{p_{data}(\mathbf{x}) + p_g(\mathbf{x})}] \end{aligned} \quad (10.4)$$

The global minimum of $C(G)$ is achieved if and only if $p_g = p_{data}$. and

$$C(G) = -\log 4 \quad (10.5)$$

and that by subtracting this expression from $C(G) = V(D_G^*, G)$, we obtain:

$$\begin{aligned} C(G) &= -\log(4) + KL(p_{data} \parallel \frac{p_{data} + p_g}{2}) + KL(p_g \parallel \frac{p_{data} + p_g}{2}) \\ &= -2\log(2) + 2JSD(p_{data} \parallel p_g) \end{aligned} \quad (10.6)$$

在D最优的情况下，等价于优化JSD

10.4 CGAN

$$\min_G \max_D V(D, G) = E_{\mathbf{x} \sim p_{data}(\mathbf{x})} [\log D(\mathbf{x}|\mathbf{y})] + E_{\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})} [\log(1 - D(G(\mathbf{z})|\mathbf{y}))] \quad (10.7)$$

10.5 KLD - JSD - Wasserstein Distance

参考文献

- [1] David Balduzzi, Marcus Frean, Lennox Leary, J. P. Lewis, Kurt Wan Duo Ma, and Brian McWilliams. The shattered gradients problem: If resnets are the answer, then what is the question? In *34th International Conference on Machine Learning, ICML 2017*, volume 1, 2017.
- [2] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems*, 2(4):303–314, dec 1989.
- [3] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. volume 9, pages 249–256, 2010.
- [4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *Proceedings of the IEEE International Conference on Computer Vision*, 2015 International Conference on Computer Vision, ICCV 2015:1026–1034, 2015.
- [5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2016-December, 2016.
- [6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 9908 LNCS, 2016.
- [7] Moshe Leshno, Vladimir Ya Lin, Allan Pinkus, and Shimon Schocken. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural Networks*, 6(6), 1993.
- [8] Aston Zhang, Zachary C. Lipton, Mu Li, and Alexander J. Smola. *Dive into Deep Learning*. 2020. <https://d2l.ai>.