# 深度学习笔记

zhliangqi

2020 年 12 月 22 日

# 目录

# Chapter 1

# Linear Algebra

## 1.1 特征值和特征空间

$$Au = \lambda u \tag{1.1}$$

特征值$\lambda$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 $\lambda_1 \geqslant \lambda_2 \geqslant ... \geqslant \lambda_r > 0$， $\lambda_{r+1} = \lambda_{r+2} = ... = \lambda_n = 0$，称正数$\sigma_i = \sqrt{\lambda_i}(i = 1, 2fi..., r)$ 为矩阵$A$的**奇异值**

$$\begin{aligned} A &= U\Sigma V^H \\ &= (u_1\ u_2\ \cdots\ u_m)\begin{pmatrix}\lambda_1 & & & \\ & & & \\ & & & \end{pmatrix}\begin{pmatrix} v_1^H \\ v_2^H \\ . \\ . \\ . \\ v_n^H \end{pmatrix} \\ &= \sigma_1 \boldsymbol{u}_1 \boldsymbol{v}_1^H + \sigma_2 \boldsymbol{u}_2 \boldsymbol{v}_1^H + \cdots + \sigma_r \boldsymbol{u}_r \boldsymbol{v}_r^H \end{aligned} \tag{1.2}$$

### 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 $\theta$ as an unknown parameter of the world. This view of the $\theta$ as being constant-valued but unknown is taken in frequentist statistics. In the frequentist this view of the world, $\theta$ 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 $\theta$ as being a random variable whose value is unknown. In this approach, we would specify a prior distribution $p(\theta)$ on $\theta$ 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，$E(X_1+X_2+...+X_n) = E(X_1)+E(X_2)+...+E(X_n)$
- 若干个独立随机变量之和的期望等于各变量的期望之和，$E(X_1X_2...X_n) = E(X_1)E(X_2)...E(X_n)$

## 2.3 方差

均匀分布$X \sim R(a,b)$ 其期望为

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

其方差为

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

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

## 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 Convolution

## 3.4 Transposed Convolution

$$
\begin{aligned}
H_{out} &= (H_{in} - 1) \times \text{stride}[0] - 2 \times \text{padding}[0] + \text{dilation}[0] \times (\text{kernel\_size}[0] - 1) + \text{output\_padding}[0] + 1 \\
W_{out} &= (W_{in} - 1) \times \text{stride}[1] - 2 \times \text{padding}[1] + \text{dilation}[1] \times (\text{kernel\_size}[1] - 1) + \text{output\_padding}[1] + 1
\end{aligned}
\tag{3.1}
$$

## 3.5 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。

## 3.6 $f$-divergence

| Divergence | f |
|---|---|
| KLD | $t \log t$ |
| reverse KLD | $-\log t$ |
| squared Hellinger distance | $(\sqrt{t}-1)^2, 2(1-\sqrt{t})$ |

## 3.7 Maximum likelihood estimation

**Frequentist statistics**

深度学习来就是用模型$p_{model}(\boldsymbol{x};\boldsymbol{\theta})$来估计数据的真实分布$p_{data}(\boldsymbol{x};\boldsymbol{\theta})$，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:

$$\begin{aligned}\hat{\boldsymbol{\theta}}_{MLE} &= \arg\max_{\theta} p_{model}(X,\boldsymbol{\theta}) \\ &= \arg\max_{\boldsymbol{\theta}} \prod_{i=1}^{m} p_{model}(\boldsymbol{x}^i;\boldsymbol{\theta})\end{aligned} \tag{3.2}$$

等价于

$$\begin{aligned}\hat{\boldsymbol{\theta}}_{MLE} &= \arg\max_{\boldsymbol{\theta}} \sum_{i=0}^{m} \log p_{model}(\boldsymbol{x}^i;\boldsymbol{\theta}) \\ &= \arg\min_{\boldsymbol{\theta}} -\sum_{i=0}^{m} \log p_{model}(\boldsymbol{x}^i;\boldsymbol{\theta})\end{aligned} \tag{3.3}$$

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

除$m$，等价于

$$\hat{\boldsymbol{\theta}}_{MLE} = \arg\max_{\boldsymbol{\theta}} E_{x\sim\hat{p}_{data}} \log p_{model}(\boldsymbol{x}^i;\boldsymbol{\theta}) \tag{3.4}$$

### 3.7.1 KL divergence

KL散度用来衡量两种分布之间的差异，

$$\begin{aligned}D_{KL}(\hat{p}_{data}||p_{model}) &= \sum_i \hat{p}_{data}(x_i) \log \frac{\hat{p}_{data}(x_i)}{p_{model}(x_i)} \\ &= \sum_i \hat{p}_{data}(x_i) \log \hat{p}_{data}(x_i) - \sum_i \hat{p}_{data}(x_i) \log p_{model}(x_i)\end{aligned} \tag{3.5}$$

$\hat{p}_{data}(\boldsymbol{x})$与模型无关，其中

$$\sum_i \hat{p}_{data}(x_i) \ln \hat{p}_{data}(x_i) \tag{3.6}$$

为常量，则问题转换为

$$\begin{aligned}\arg\min D_{KL}(\hat{p}_{data}||p_{model}) &= \arg\min -\sum_i \hat{p}_{data}(x_i) \log p_{model}(x_i) \\ &= \arg\min -E_x \log p_{model}(\mathbf{x})\end{aligned} \tag{3.7}$$

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。

### 3.7.2 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。

## 3.8 Maximum A Posteriori

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

$$p(\boldsymbol{\theta}|\mathbf{X}) = \frac{p(\boldsymbol{\theta})p(\mathbf{X}|\boldsymbol{\theta})}{p(\mathbf{X})} \tag{3.8}$$

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

$$\begin{aligned}\hat{\boldsymbol{\theta}}_{MAP} &= \arg\max p(\boldsymbol{\theta}|\mathbf{X}) \\ &= \arg\min -\log p(\boldsymbol{\theta}|\mathbf{X}) \\ &= \arg\min -\log p(\mathbf{X}|\boldsymbol{\theta}) - \log p(\boldsymbol{\theta}) + \log p(\mathbf{X}) \\ &= \arg\min -\log p(\mathbf{X}|\boldsymbol{\theta}) - \log p(\boldsymbol{\theta})\end{aligned} \tag{3.9}$$

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

$$p(\boldsymbol{\theta}) = C \times e^{-\frac{\theta^2}{2\sigma^2}} \tag{3.10}$$

那么

$$-\log p(\boldsymbol{\theta}) = C + \frac{\theta^2}{2\sigma^2} \tag{3.11}$$

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：

$$p(x|\mathbf{X}) = \int p(x|\boldsymbol{\theta},\mathbf{X})p(\boldsymbol{\theta}|\mathbf{X})\mathrm{d}\boldsymbol{\theta} \tag{3.12}$$

## 3.9 Initialization scheme

### 3.9.1 constant initialization

### 3.9.2 random initialization

按照某一分布随机初始化

**normal initialization**

$$W \sim N(\mu, \sigma^2) \tag{3.13}$$

**uniform initalization**

$$W \sim U[-\frac{1}{\sqrt{n}}, \frac{1}{\sqrt{n}}] \tag{3.14}$$

### 3.9.3 xavier initalization

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

$$\begin{aligned} \mathbf{y}_l &= \mathbf{W}_l \mathbf{x}_l + \mathbf{b}_l \\ \mathbf{x}_l &= f(\mathbf{y}_{l-1}) \end{aligned} \tag{3.15}$$

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

$$\begin{aligned} Var(\mathbf{y}_l) &= Var(\sum \mathbf{w}_l \mathbf{x}_l + \mathbf{b}_l) \\ &= Var(\sum \mathbf{w}_l \mathbf{x}_l) \\ &= n_l Var(\mathbf{w}_l \mathbf{x}_l) \end{aligned} \tag{3.16}$$

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

$$\begin{aligned} Var(\mathbf{y}_l) &= n_l E(\mathbf{w}_l^2 \mathbf{x}_l^2) - n_l E^2 \mathbf{w}_l E^2 \mathbf{x}_l \\ &= n_l E(\mathbf{w}_l^2 \mathbf{x}_l^2) \\ &= n_l Var(\mathbf{w}_l) E(\mathbf{x}_l^2) \end{aligned} \tag{3.17}$$

若$E\mathbf{x}_l = 0$，则

$$Var(\mathbf{y}_l) = n_l Var(\mathbf{w}_l) Var(\mathbf{x}_l) \tag{3.18}$$

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

$$Var(\mathbf{w}_l) = \frac{1}{n_l} \tag{3.19}$$

- 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:

$$\Delta \mathbf{x}_l = \hat{\mathbf{W}}_l \Delta \mathbf{y}_l \tag{3.20}$$

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

$$Var(\Delta\mathbf{x}_l) = \hat{n}_l Var(\mathbf{w}_l) Var(\Delta\mathbf{y}_l) \tag{3.21}$$

- 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，$Var(\mathbf{w}_l) = \frac{2}{n_l+\hat{n}_l}$，可得

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

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

### 3.9.4 orthogonal initalization

### 3.9.5 kaiming initalization

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

$$\begin{aligned} Var(\mathbf{y}_l) &= \frac{1}{2} n_l Var(\mathbf{w}_l) Var(\mathbf{x}_l) \\ Var(\Delta\mathbf{x}_l) &= \frac{1}{2} \hat{n}_l Var(\mathbf{w}_l) Var(\Delta\mathbf{y}_l) \end{aligned} \tag{3.22}$$

且在权重初始化时，使用上述任意一个即可。

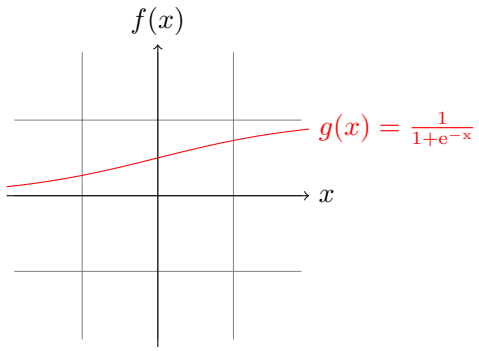
## 3.10 正则化

## 3.11 Data proprocessing

### 3.11.1 whitening

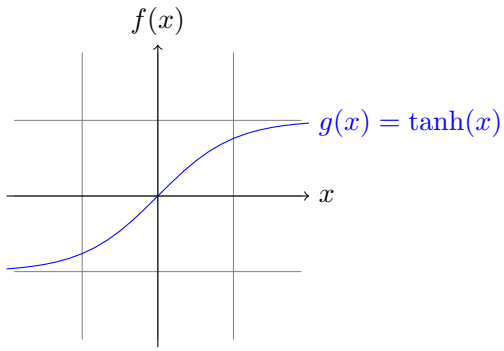
## 3.12 Activation functions

### 3.12.1 sigmoid



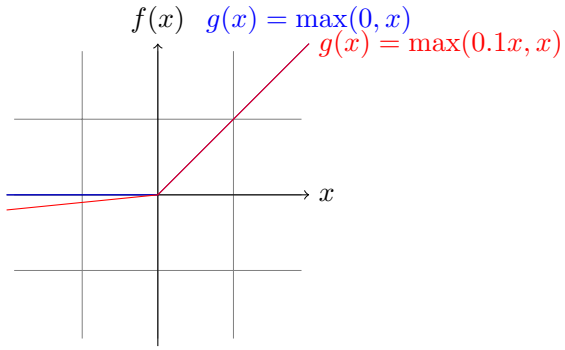

$$
\begin{aligned}
g(x) &= \frac{1}{1+\mathrm{e}^{-x}} \\
g(x)' &= g(x)(1-g(x))
\end{aligned}
\tag{3.23}
$$

### 3.12.2 tanh(x)




$$
\begin{aligned}
g(x) &= tanh(x) \\
g(x)' &= 1 - g(x)^2 \\
&= 1 - \tanh(x)^2
\end{aligned}
\tag{3.24}
$$

### 3.12.3 Rectified linear units




**ReLU**

$$g(x) = \max(0, x) \tag{3.25}$$

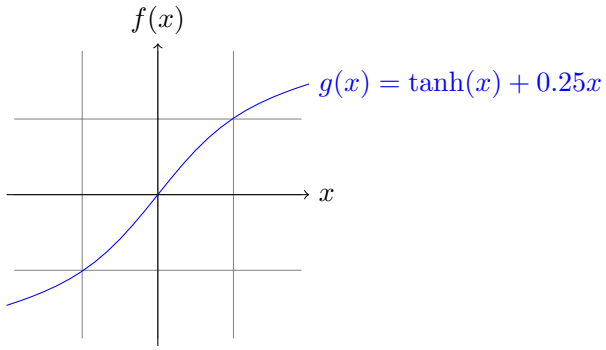
**Leaky ReLU**

$$g(x) = \max(0.01x, x) \tag{3.26}$$

**PReLU**

$$g(x) = \max(\alpha x, x) \tag{3.27}$$

### 3.12.4 haha




$$\begin{aligned} g(x) &= \tanh(x) + 0.25x \\ g(x)' &= 0.75 - g(x)^2 \\ &= 0.75 - \tanh(x)^2 \end{aligned} \tag{3.28}$$

### 3.12.5 Loss functions

## 3.13 信息熵

### 3.13.1 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**。如果我们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:

$$\begin{aligned} h(x,y) &= h(x) + h(y) \\ p(x,y) &= p(x) + p(y) \end{aligned} \tag{3.29}$$

为满足以上性质，信息量$h(x)$公式为下

$$h(x) = -\log_2 p(x) \tag{3.30}$$

### 3.13.2 信息熵

信息熵即为随机变量的信息量的期望

$$H(x) = -\sum_i p(x_i) \log_2 p(x_i) \tag{3.31}$$

### 3.13.3 条件熵

$$H(X|Y) = -\sum_{i,j} p(x_i, y_j) \log \frac{p(x_i, y_i)}{p(y_i)} \tag{3.32}$$

### 3.13.4 Mutual Information

互信息是衡量随机变量之间互相依赖程度的度量。

$$\begin{aligned} I(X;Y) &= H(X) - H(X|Y) \\ &= \sum_y \sum_x p(x,y) \log(\frac{p(x,y)}{p(x)p(y)}) \end{aligned} \tag{3.33}$$

## 3.14 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$x \in \chi$

$$\begin{aligned} P_s(Y|X=x) &= P_t(Y|X=x) \\ P_s(X) &\neq P_t(X) \end{aligned} \tag{3.34}$$

**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 requring lower lr

ant 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: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: $\gamma$, $\beta$
**Output:** $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

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

图 3.1: Batch Normalization

## 3.15 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-BatchNoram 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.


(a) loss landscape (b) gradient predictiveness (c) "effective" $\beta$-smoothness

**Why does BatchNorm work?**

BatchNorm reparametrizes the underlying optimization problem to make its landscape significatly 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 minimium. 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 loacl minima.*[8]

### 4.1.2 Saddle Points

We assume that the input of a function is $k$-dimensional vector and its ouput is a scalar, so its *Hession 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 Hession matrix at the zero-gradient position

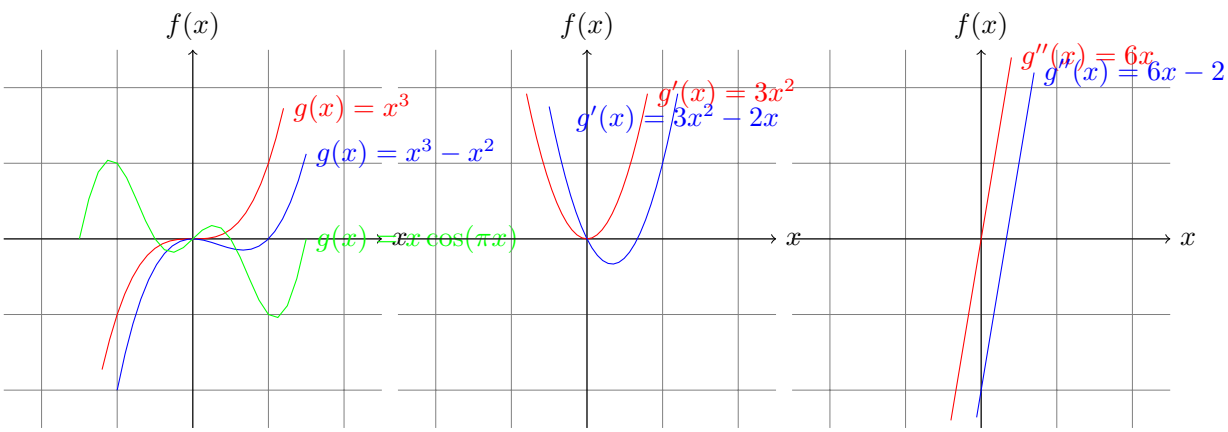



图 4.1: Challenges

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

- When the eigenvalues of the function's Hession matrix at the zero-gradient position are all negative, we have a local maximum for the function.
- When the eigenvalues of the function's Hession 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 likehood that at least some of the eigenvalues are negative is quite high. This makes sadlle points are more likely then 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 nosie? and how it effect the optimization. 无偏估计、一致性

## 4.2 Gradient Descent

Using a Taylor expansion we obtain that:

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

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

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

Choose $\eta$ samll enough for the higher order terms to become irrelevant. then

$$f(x - \eta f'(x)) \lessapprox f(x) \tag{4.3}$$

that means, if we use

$$x \leftarrow x - \eta f'(x) \tag{4.4}$$

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 $\eta$ automatically or get rid of having to select a setp size at all? Second order methods that look not only at the value and gradient of the objective but alse 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:

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

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 $f$ statifies $\nabla f(\mathbf{x}) = 0$. Taking derivatives of above equtaion with regard to $\eta$ and ignoring higher order terms we arrive at

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

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

$$\eta = -H_f^{-1} \tag{4.7}$$

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

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

**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$g^\top Hg$，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

$$\eta_g = \frac{1}{g^\top H_f g} \tag{4.9}$$

最差的情况下，$g$与$H$最大的特征值$\lambda_{max}$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**

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

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 $i \in i, ..., n$ for data examples at random, and compute the gradient $\nabla f_i(\mathbf{x})$ to update $\mathbf{x}$:

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

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.

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

## 4.3 SGDM - SGD with Momentum

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

$\mathbf{v}$ 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**

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

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

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

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

## 4.4 NAG - Nesterov Accelerated Gradient

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

## 4.5 Adagrad

Consider a model training on spare 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.

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

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 $\mathbf{s}_t$ keeps on growing without bound due to the lack of normalization, essentially linearly as the algorithm converges.

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

$$\begin{aligned} \mathbf{s}_t &= (1-\gamma)\mathbf{g}_t^2 + \gamma \mathbf{s}_{t-1} \\ &= (1-\gamma)(\mathbf{g}_t^2 + \gamma \mathbf{g}_{t-1}^2 + \gamma^2 \mathbf{g}_{t-2}^2 + ...) \end{aligned} \tag{4.19}$$

## 4.7 Adadelta

A variant of AdaGrad.

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

## 4.8 Adam

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

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 initally 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.

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

## 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 theentire 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):

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

反向传播过程:

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

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**。

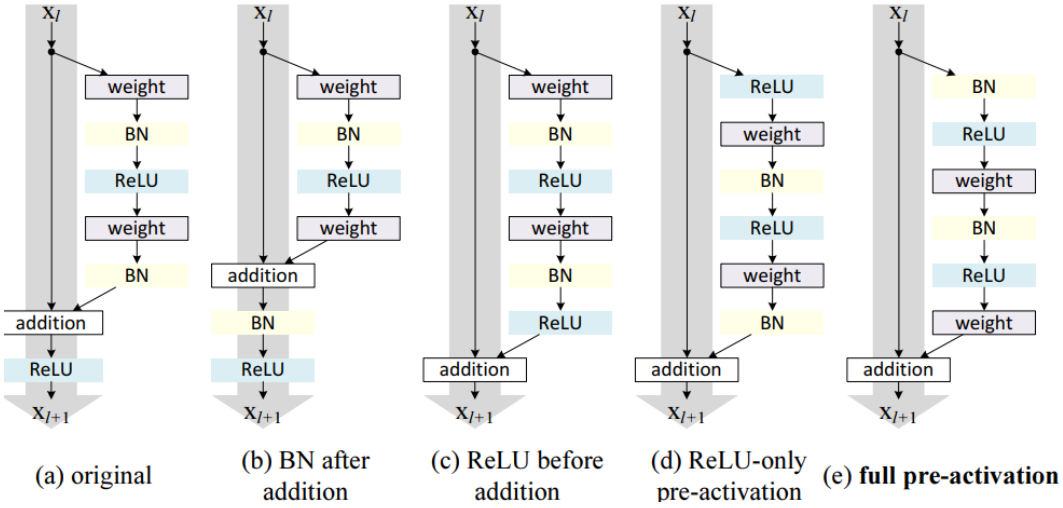



图 6.1: Residual Units

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:

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

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

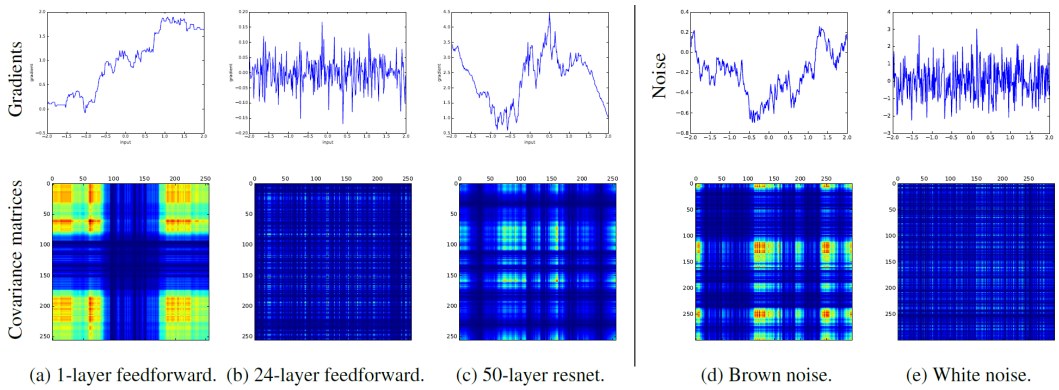
$$\mathbf{x}_{l+1} = \mathbf{x}_l + \mathcal{F}(\hat{f}(\mathbf{x}_l), \mathcal{W}_l) \tag{6.7}$$

## 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。




(a) 1-layer feedforward. (b) 24-layer feedforward. (c) 50-layer resnet. (d) Brown noise. (e) White noise.

图 6.2: The shattered Gradients Problem

## 6.3 DenseNet

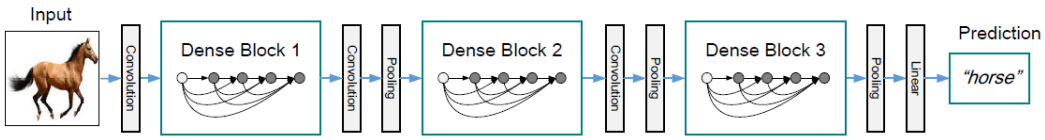



图 6.3: DenseNet

## 6.4 FCN

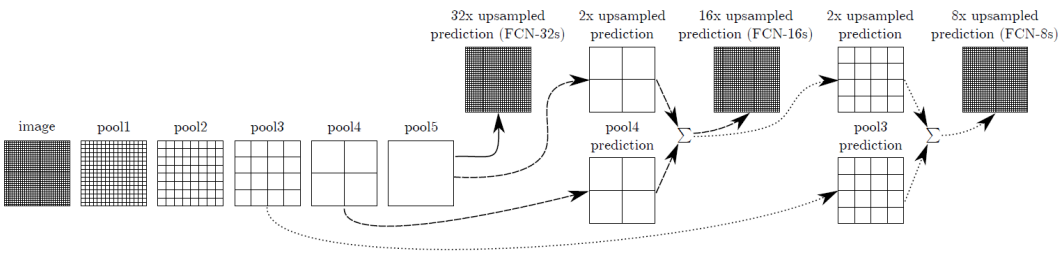



图 6.4: FCN

## 6.5 UNet family

## 6.6 FCN

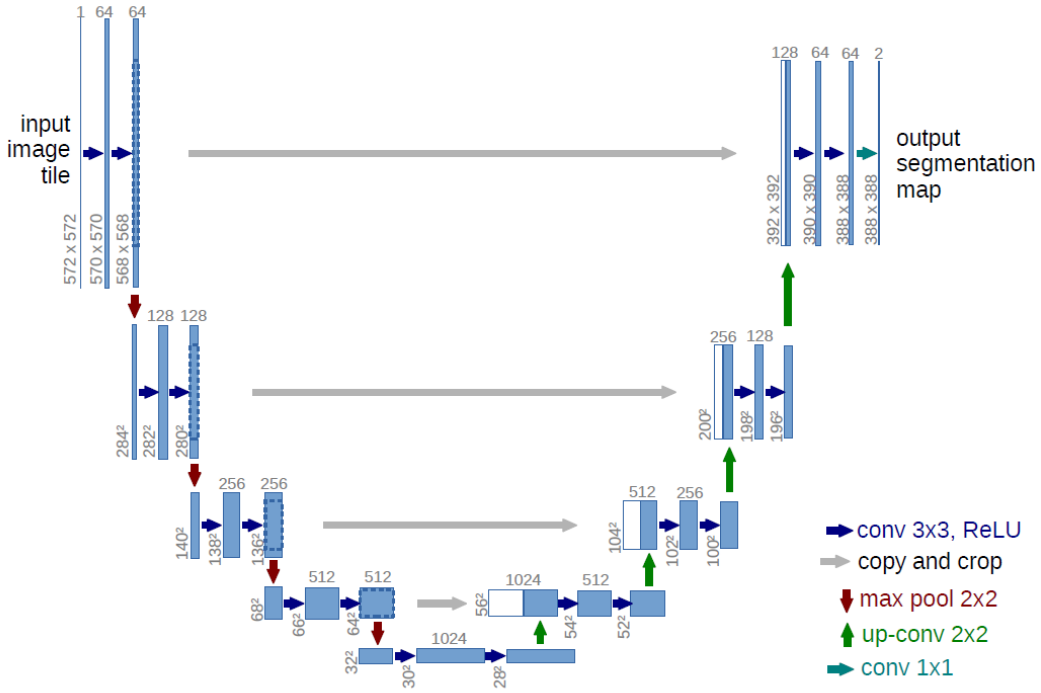



图 6.5: UNet

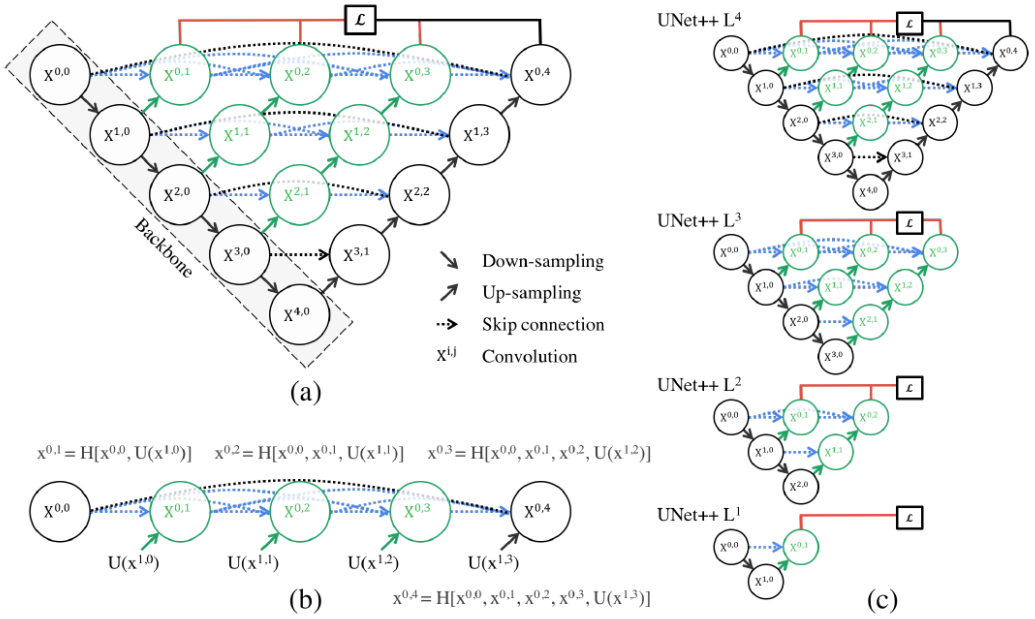



图 6.6: UNet 2+

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。

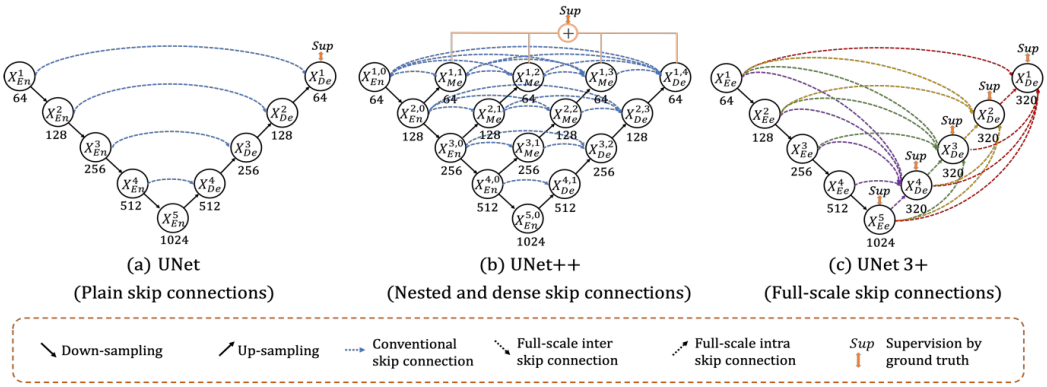



图 6.7: UNet 3+

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

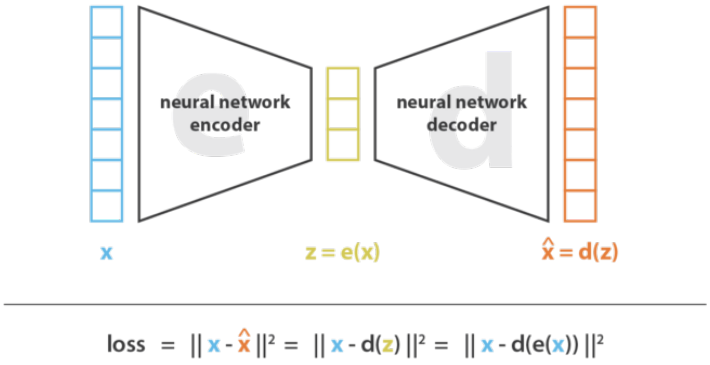



图 8.1: Autoencoder

## 8.1 linear autoencoder VS PCA

## 8.2 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(降维->特征)。

## 8.3 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。

### 8.3.1 稀疏自编码器

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

### 8.3.2 DAE - Denoising Autoencoder

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

### 8.3.3 CAE - Contractive Autoencoder

$$\begin{aligned} &L(\mathbf{x}, g(f(\mathbf{x}))) + \Omega(\mathbf{h}, \mathbf{x}) \\ &\Omega(\mathbf{h}, \mathbf{x}) = \lambda \sum_i \|\nabla_x h_i\|^2 \end{aligned} \tag{8.3}$$

# Chapter 9

# Deep Generation Model

- **Generative** models can capture the joint probability $p(X, Y)$, or just $p(X)$ if there are no labels
- **Discriminative** models capture the conditional probability $p(Y|X)$

How can we generate new data instances? If we had the data distribution $p(X)$, we could just sample from it and then we would get all the instances.

$$p(\mathbf{z}, \mathbf{x}) = p(\mathbf{z})p(\mathbf{x}|\mathbf{z}) = p(\mathbf{x})p(\mathbf{z}|\mathbf{x}) \tag{9.1}$$

$$p(\mathbf{z}|\mathbf{x}) = \frac{p(\mathbf{z}, \mathbf{x})}{p(\mathbf{x})} = \frac{p(\mathbf{z})p(\mathbf{x}|\mathbf{z})}{p(\mathbf{x})} \tag{9.2}$$

$$p(\mathbf{x}) = \int p(\mathbf{z})p(\mathbf{x}|\mathbf{z})d\mathbf{x} \tag{9.3}$$

函数的函数是泛函求泛函的极值方法是变分法

## 9.1 Variational Inference

Consider a joint density of latent variables $\mathbf{z}$ and observations $\mathbf{x}$

$$p(\mathbf{z}, \mathbf{x}) = p(\mathbf{z})p(\mathbf{x}|\mathbf{z}) \tag{9.4}$$

In Bayesian models, the latent variables help govern the distribution of the data. A Bayesian model draws the latent variables from a **prior density** $p(\mathbf{z})$ and then relates them to the observations through the **likelihood** $p(\mathbf{x}|\mathbf{z})$. Inference in a Bayesian model amounts to conditioning on data and computing the **posterior** $p(\mathbf{z}|\mathbf{x})$. In complex Bayesian models, this computation often requires approximate inference.

The main idea behind variational inference is to use optimization. Variational inference thus turns the inference problem into an optimization problem First, we posit a family of approximate densities Q. This is a set of densities over the latent variables. Then, we try to find the member of that family that minimizes the Kullback-Leibler (KL) divergence to the exact posterior,

$$q^*(\mathbf{z}) = \arg\min_{q(\mathbf{z})\in\mathcal{Q}} KL(q(\mathbf{z})\|p(\mathbf{z}|\mathbf{x})) \tag{9.5}$$

Finally, we approximate the posterior with the optimized member of the family $q^*(\mathbf{z})$

$$\begin{aligned} KL(q(\mathbf{z})\|p(\mathbf{z}|\mathbf{x})) &= \mathbb{E}_{q(\mathbf{z})}[\log q(\mathbf{z})] - \mathbb{E}_{q(\mathbf{z})}[\log p(\mathbf{z}|\mathbf{x})] \\ &= \mathbb{E}_{q(\mathbf{z})}[\log q(\mathbf{z})] - \mathbb{E}_{q(\mathbf{z})}[\log p(\mathbf{z},\mathbf{x})] + \mathbb{E}_{q(\mathbf{z})}[\log p(\mathbf{x})] \\ &= \mathbb{E}_{q(\mathbf{z})}[\log q(\mathbf{z})] - \mathbb{E}_{q(\mathbf{z})}[\log p(\mathbf{z},\mathbf{x})] + \log p(\mathbf{x}) \end{aligned} \tag{9.6}$$

$$\log p(\mathbf{x}) = KL(q(\mathbf{z})\|p(\mathbf{z}|\mathbf{x})) + \mathbb{E}_{q(\mathbf{z})}[\log p(\mathbf{z},\mathbf{x})] - \mathbb{E}_{q(\mathbf{z})}[\log q(\mathbf{z})]$$

$\log p(\mathbf{x})$ is constant with respect to $q(\mathbf{z})$, $KL(\cdot) \geq 0$, then

$$ELBO(q) = \mathbb{E}_{q(\mathbf{z})}[\log p(\mathbf{z},\mathbf{x})] - \mathbb{E}_{q(\mathbf{z})}[\log q(\mathbf{z})] \leq \log p(\mathbf{x}) \tag{9.7}$$

The function is call **evdience low bound(ELBO)**. Maximizing the ELBO is equivalent to minimizing the KL divergence.

$$\begin{aligned} ELBO(q) &= \mathbb{E}_{q(\mathbf{z})}[\log p(\mathbf{z},\mathbf{x})] - \mathbb{E}_{q(\mathbf{z})}[\log q(\mathbf{z})] \\ &= \mathbb{E}_{q(\mathbf{z})}[\log p(\mathbf{z})] + \mathbb{E}_{q(\mathbf{z})}[\log p(\mathbf{x}|\mathbf{z})] - \mathbb{E}_{q(\mathbf{z})}[\log q(\mathbf{z})] \\ &= \mathbb{E}_{q(\mathbf{z})}[\log p(\mathbf{x}|\mathbf{z})] - KL(q(\mathbf{z})\|p(\mathbf{z})) \end{aligned} \tag{9.8}$$

The first termis an expected likelihood; it encourages densities that place their mass on configurations of the latent variables that explain the observed data. The second term is the negative divergence between the variational density and the prior; it encourages densities close to the prior.

### 9.1.1 Regularizer - Solution of $-KL(q(\mathbf{z})\|p(\mathbf{z}))$, Gaussion case

> VAEs take an unusual approach to dealing with this problem: they assume that there is no simple interpretation of the dimensions of z, and instead assert that samples of z can be drawn from a simple distribution, i.e., $\mathcal{N}(0, I)$, where $I$ is the identity matrix. The key is to notice that any distribution in $d$ dimensions can be generated by taking a set of $d$ variables that are normally distributed and mapping them through a sufficiently complicated function.

When both prior $p_\theta(z) = N(0; I)$ and the posterior approximation $q_\phi(z|x^{(i)})$ are Gaussian. Let $J$ be the dimensionality of $\mathbf{z}$. Let $\mu$ and $\sigma$ denote the variational mean and s.d. evaluated at datapoint $i$, and let $\mu_j$ and $\sigma_j$ simply denote the $j$-th element of these vectors. Then

$$\begin{aligned} \int q_\theta(\mathbf{z}) \log p(\mathbf{z}) d\mathbf{z} &= \int \mathcal{N}(\mathbf{z}; \mu, \sigma^2) \log \mathcal{N}(\mathbf{z}; \mathbf{0}, \mathbf{I}) d\mathbf{z} \\ &= -\frac{J}{2}\log(2\pi) - \frac{1}{2}\sum_{j=1}^{J}(\mu_j^2 + \sigma_j^2) \end{aligned} \tag{9.9}$$

And:

$$\begin{aligned} \int q_\theta(\mathbf{z}) \log q_\theta(\mathbf{z}) d\mathbf{z} &= \int \mathcal{N}(\mathbf{z}; \mu, \sigma^2) \log (N)(\mathbf{z}; \mu, \sigma^2) d\mathbf{z} \\ &= -\frac{J}{2}\log(2\pi) - \frac{1}{2}\sum_{j=1}^{J}(1 + \log \sigma_j^2) \end{aligned} \tag{9.10}$$

Therefore:

$$-KL(q(\mathbf{z})\|p(\mathbf{z})) = \frac{1}{2}\sum_{j=1}^{J}(1+\log\sigma_j^2-\mu_j^2-\sigma_j^2) \tag{9.11}$$

### 9.1.2 Reconstruction Error

The usual choice is to say that $q(z|x) = \mathcal{N}(z|\mu(x;\boldsymbol{\theta}),\Sigma(\mathbf{x};\boldsymbol{\theta}))$,where $\mu$ and $\Sigma$ are arbitrary deterministic functions with parameters $\theta$ that can be learned from data. In practice, $\mu$ and $\Sigma$ are again implemented via neural networks, and $\Sigma$ is constrained to be a diagonal matrix.

ELBO可以写为

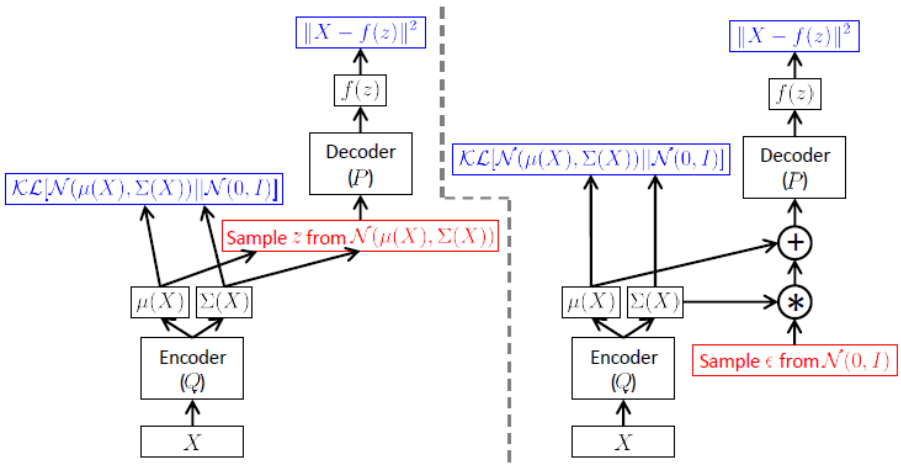
$$ELBO(\boldsymbol{\theta},\phi,\mathbf{x}) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})] - KL(q_\phi(\mathbf{z}|\mathbf{x})\|p_{\boldsymbol{\theta}}(\mathbf{z})) \tag{9.12}$$

We want to **differentiate** and optimize the lower bound $ELBO(\boldsymbol{\theta},\phi,\mathbf{x})$ w.r.t both the variational parameters $\phi$ and generativet parameters $\theta$.由于隐变量$z$从$q(z|x)$的采样过程不可微，所以需要改写成一个可微的形式. we can reparameterize the random variale $z \sim q_\phi(z|x)$ using a differentiable transformation $g_\phi(\epsilon,x)$ of an (auxiliary) noise variable $\epsilon$

$$\tilde{\mathbf{z}} = g_\phi(\boldsymbol{\epsilon},\mathbf{x}) \quad with \quad \boldsymbol{\epsilon}\sim p(\boldsymbol{\epsilon}) \tag{9.13}$$

### 9.1.3 Reparameterization trick

Given $\mu(\mathbf{x})$ and $\Sigma(\mathbf{x})$—the mean and covariance of $q(z|x)$ —we can sample from $\mathcal{N}(\mu(x),\Sigma(x))$ by first sampling $\epsilon\sim\mathcal{N}(0,I)$, then computing $z=\mu(x)+\Sigma^{\frac{1}{2}}(x)*\epsilon$.



## 9.2 VAE

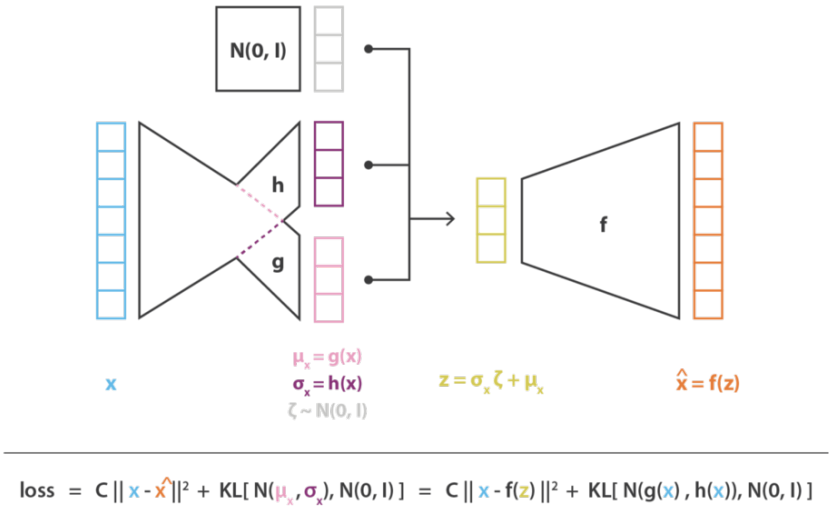



图 9.1: vae

### 9.2.1 为什么需要VAE？为什么不直接使用Autoencoder的decoder来生成图片？

Autoencoder的encoder生成的latent space不够regurlarity(不连续)，我们无法从latent space的中随机采样来生成或修改图片

the lack of interpretable and exploitable structures in the latent space (**lack of regularity**). the regularity of the latent space for autoencoders is a difficult point that depends on the distribution of the data in the initial space, the dimension of the latent space and the architecture of the encoder. So, it is pretty difficult (if not impossible) to ensure, a priori, that the encoder will organize the latent space in a smart way compatible with the generative process we just described.

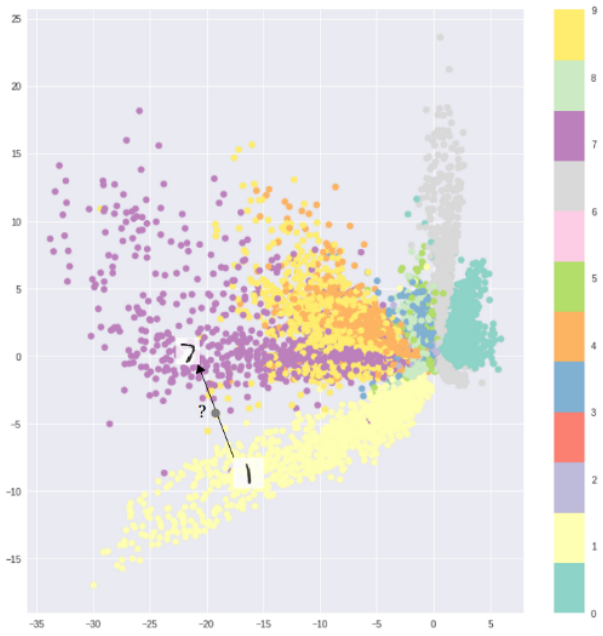


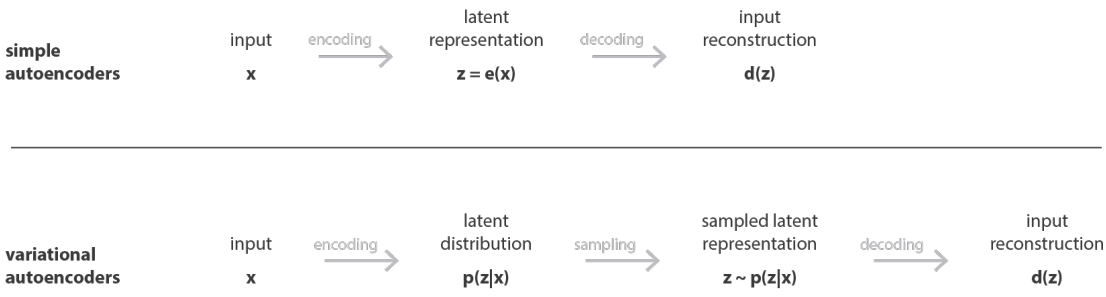

图 9.2: the encodings from a 2D latent space / MNIST

### 9.2.2 VAE

a variational autoencoder can be defined as being an autoencoder whose training is regularised to avoid overfitting and ensure that the latent space has good properties that enable generative process.

VAE模型需要符合以下条件

- the input is **encoded as distribution** over the latent space, instead of encoding an input as a single point
- a point from the latent space is sampled from that distribution
- the sampled point is decoded and the reconstruction error can be computed
- the reconstruction error is backpropagated through the network



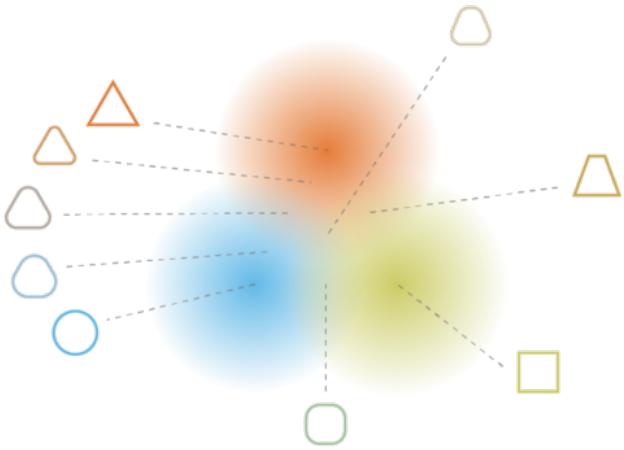

Latent space应该具有以下的性质

- **Continuity** two close points in the latent space should not give two completely different contents once decoded

- **Completeness** for a chosen distribution, a point sampled from the latent space should give "meaningful" content once decoded


The only fact that VAEs encode inputs as distributions instead of simple points is not sufficient to ensure continuity and completeness. Without a well defined regularisation term, the model can learn, in order to minimise its reconstruction error, to "ignore" the fact that distributions are returned and behave almost like classic autoencoders (leading to overfitting). To do so, the encoder can either return distributions with tiny variances (that would tend to be punctual distributions(点分布)) or return distributions with very different means (that would then be really far apart from each other in the latent space).


we have to regularise both the covariance matrix and the mean of the distributions returned by the encoder. In practice, this regularisation is done by **enforcing distributions to be close to a standard normal distribution**



### 9.2.3

### 9.2.4 贝叶斯混合高斯模型

## 9.3 Generative Adversarial Nets

### 9.3.1 Mode collapse

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

### 9.3.2 Wasserstein GAN and the Kantorovich-Rubinstein Duality

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

### 9.3.3 vanila GAN

$z \sim p(z) -> p_{data}(x)$

$$\min_G \max_D V(D,G) = E_{\boldsymbol{x}\sim p_{data}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + E_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))] \tag{9.14}$$

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

$$\begin{aligned} V(G,D) &= \int_{\boldsymbol{x}} p_{data}(\boldsymbol{x}) \log(D(\boldsymbol{x}))dx \int_{\boldsymbol{z}} p_{\boldsymbol{z}}(\boldsymbol{z}) \log(1 - D(g(\boldsymbol{z})))dz \\ &= \int_{\boldsymbol{x}} p_{data}(\boldsymbol{x}) \log(D(\boldsymbol{x})) + p_g(\boldsymbol{x}) \log(1 - D(\boldsymbol{x}))dx \end{aligned} \tag{9.15}$$

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

$$D_G^*(\boldsymbol{x}) = \frac{p_{data}(\boldsymbol{x})}{p_{data}(\boldsymbol{x}) + p_g(\boldsymbol{x})} \tag{9.16}$$

Note that the training objective for D can be interpreted as maximizing the log-likelihood for

estimating the conditional probability $P(Y = y|\boldsymbol{x}), \boldsymbol{x} \in p_{data} or \in p_g$, then

$$\begin{aligned} C(G) &= max_D V(G, D) \\ &= \mathbb{E}_{\boldsymbol{x}\sim p_{data}}[\log D_G^*(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}}[\log(1 - D_G^*(G(\boldsymbol{z})))] \\ &= \mathbb{E}_{\boldsymbol{x}\sim p_{data}}[\log D_G^*(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{x}\sim p_g}[\log D_G^*(\boldsymbol{x})] \\ &= \mathbb{E}_{\boldsymbol{x}\sim p_{data}}[\log \frac{p_{data}(\boldsymbol{x})}{p_{data}(\boldsymbol{x}) + p_g(\boldsymbol{x})}] + \mathbb{E}_{\boldsymbol{x}\sim p_g}[\log \frac{p_g(\boldsymbol{x})}{p_{data}(\boldsymbol{x}) + p_g(\boldsymbol{x})}] \end{aligned} \tag{9.17}$$

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

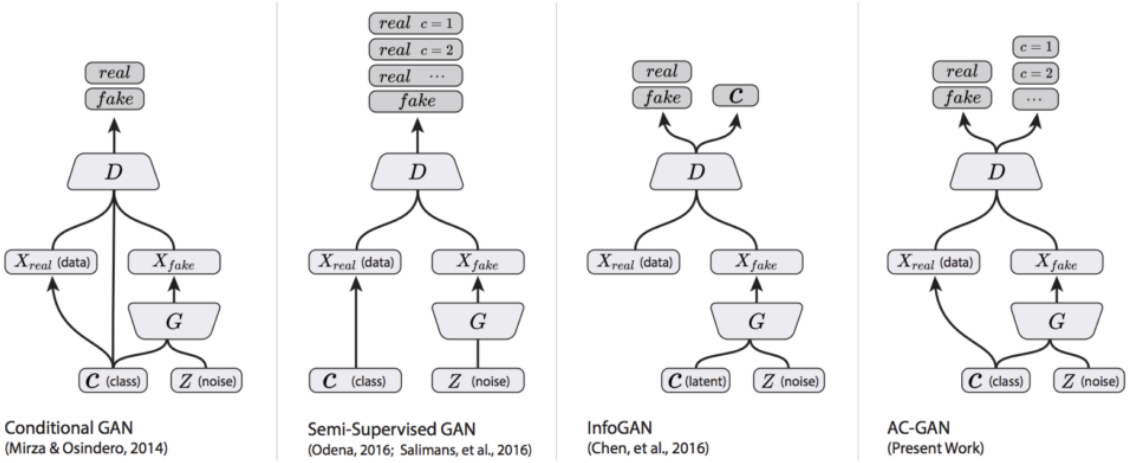
$$C(G) = -\log 4 \tag{9.18}$$

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

$$\begin{aligned} C(G) &= -log(4) + KL(p_{data}\|\frac{p_{data} + p_g}{2}) + KL(p_g\|\frac{p_data + p_g}{2}) \\ &= -2\log(2) + 2JSD(p_{data}\|p_g) \end{aligned} \tag{9.19}$$

在D最优的情况下，等价于优化JSD

### 9.3.4 Latent space




**Conditional GAN**

使用辅助信息——label，将label和image绑定，分别输入到D和G中，

$$\min_G \max_D V(D, G) = E_{\boldsymbol{x}\sim p_{data}(\boldsymbol{x})}[\log D(\boldsymbol{x}|\boldsymbol{y})] + E_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z}|\boldsymbol{y})))] \tag{9.20}$$

可以使用标签控制生成指定类型的图片

**Semi-Supervised GAN**

**InfoGAN**

**ACGAN**

### 9.3.5 Architecture

### 9.3.6 Object functions

# 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.