从PM到GAN

——LSTM之父Schmidhuber横跨22年的怨念

分享人：郑华滨
知乎专栏：AI带路党

目录

- 八卦Schmidhuber与GAN之间的恩怨
- 讲解Schmidhuber在92年提出的PM模型
- 简单介绍GAN及其他模型
- 对比PM和GAN及其他模型之间的异同

目录

- 八卦Schmidhuber与GAN之间的恩怨
- 讲解Schmidhuber在92年提出的PM模型
- 简单介绍GAN及其他模型
- 对比PM和GAN及其他模型之间的异同

2016年12月，NIPS大会
Ian Goodfellow的GAN Tutorial上
发生了尴尬的一幕……

Generative Adversarial Networks (GANs)

Ian Goodfellow, OpenAI Research Scientist
NIPS 2016 tutorial
Barcelona, 2016-12-4

OpenAI

当进行到GAN与其他模型的比较时
一位神秘人物站起来打断了演讲

[视频片段]

Jürgen Schmidhuber

Comparison to NCE; …

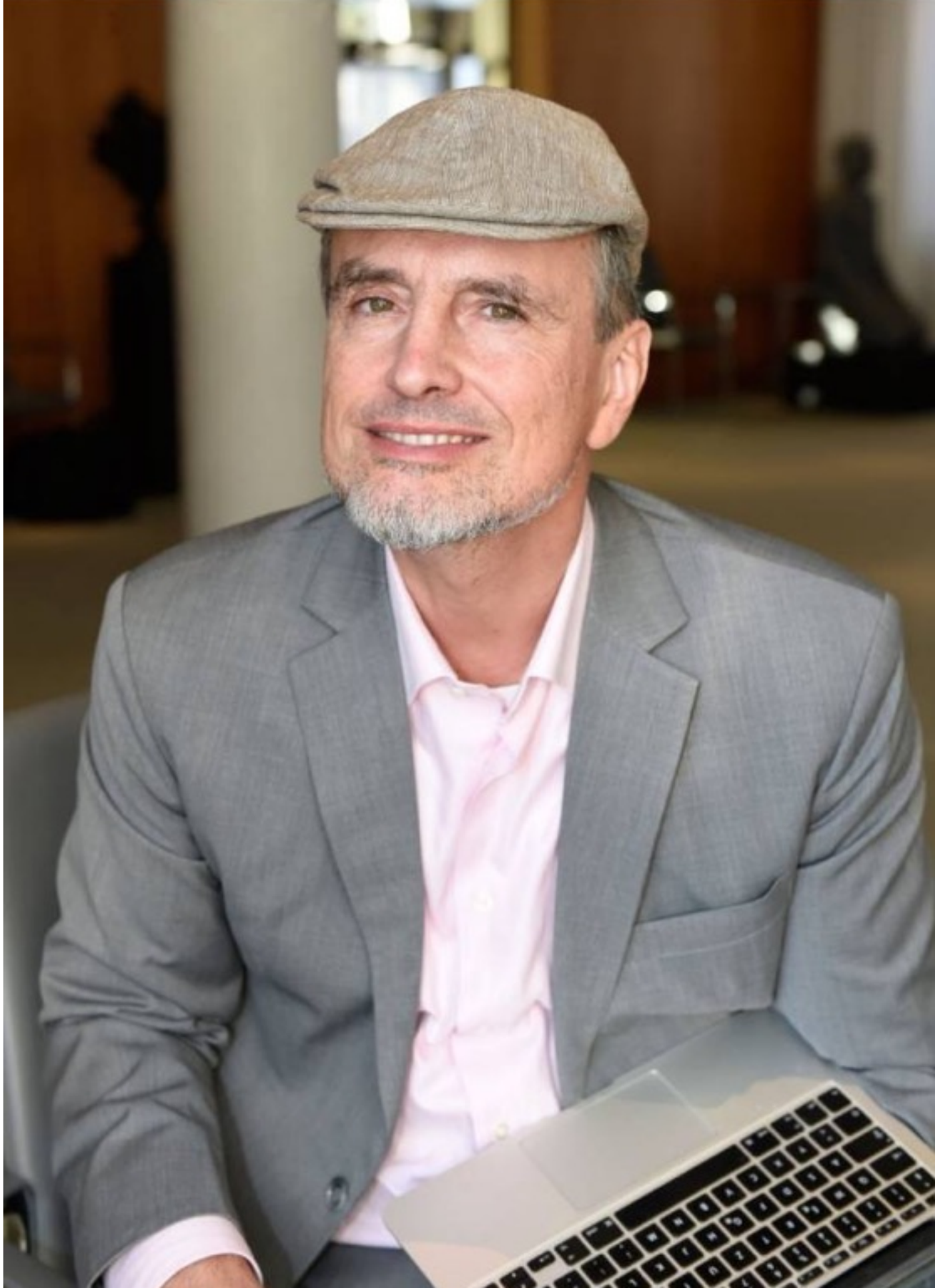
$$V(G,D) = \mathbb{E}_{p_{\text{data}}} \log D(\boldsymbol{x}) + \mathbb{E}_{p_{\text{generator}}} (\log (…$$

	NCE (Gutmann and Hyvärinen 2010)	MLE	GAN
D	$D(x) = \frac{p_{\text{model}}(\boldsymbol{x})}{p_{\text{model}}(\boldsymbol{x}) + p_{\text{generator}}(\boldsymbol{x})}$		Neur… netwo…
Goal	Learn p_{model}		Learn p_g
G update rule	None (G is fixed)	Copy p_{model} parameters	Gradi… descent
D update rule	Gradient ascent on V		

("On Distinguishability Criteria…", Goodfellow 20…

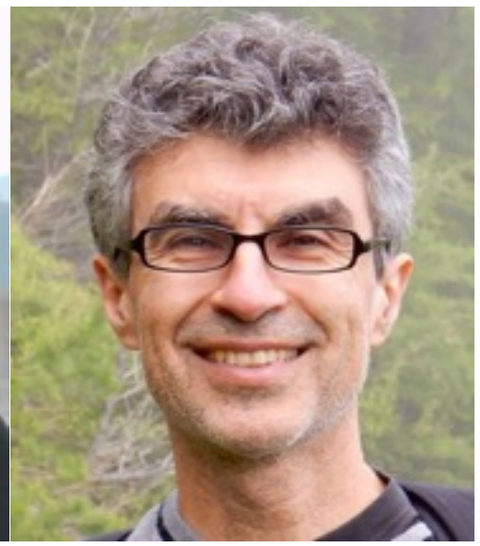
Jürgen Schmidhuber

- 德国AI科学家
- 深度学习先驱人物
- LSTM发明者之一



今天主要八卦Schmidhuber跟Goodfellow在GAN上的争论
其实之前还指责过三巨头的《Deep Learning》综述
认为自己的很多成果没有得到应有的重视和荣誉
感兴趣可自行挖掘：
Critique of Paper by "Deep Learning Conspiracy"
LeCun霸气回应





92年的时候我提出了
Predictability Minimization，它是
#￥%&@*##@￥&￥#……

请问你如何看待它和
GAN之间的相似之处?

Comparison to NCE;

$$V(G,D) = \mathbb{E}_{p_{\text{data}}} \log D(\boldsymbol{x}) + \mathbb{E}_{p_{\text{generator}}} (\log$$

	NCE (Gutmann and Hyvärinen 2010)	MLE	GA
D	$D(x) = \frac{p_{\text{model}}(\boldsymbol{x})}{p_{\text{model}}(\boldsymbol{x}) + p_{\text{generator}}(\boldsymbol{x})}$		Neur netwo
Goal	Learn p_{model}		Learn p_g
G update rule	None (G is fixed)	Copy p_{model} parameters	Gradi descent
D update rule	Gradient ascent on V		

("On Distinguishability Criteria ...", Goodfellow 20

论文和之前的邮件里
已经说得很清楚了

公开怼人很不优雅哦

大家还要听我讲课呢

其实早在2014年的GAN首作中
GAN已经和PM进行了比较

Some previous work has used the general concept of having two neural networks compete. The most relevant work is predictability minimization [26]. In predictability minimization, each hidden unit in a neural network is trained to be different from the output of a second network, which predicts the value of that hidden unit given the value of all of the other hidden units. This work differs from predictability minimization in three important ways: 1) in this work, the competition between the networks is the sole training criterion, and is sufficient on its own to train the network. Predictability minimization is only a regularizer that encourages the hidden units of a neural network to be statistically independent while they accomplish some other task; it is not a primary training criterion. 2) The nature of the competition is different. In predictability minimization, two networks' outputs are compared, with one network trying to make the outputs similar and the other trying to make the

«Generative Adversarial Nets» final version

然而原稿并无这段
前后变化的原因竟是……

Some previous work has used the general concept of having two neural networks compete. The most relevant work is predictability minimization [26]. In predictability minimization, each hidden unit in a neural network is trained to be different from the output of a second network, which predicts the value of that hidden unit given the value of all of the other hidden units. This work differs from predictability minimization in three important ways: 1) in this work, the competition between the networks is the sole training criterion, and is sufficient on its own to train the network. Predictability minimization is only a regularizer that encourages the hidden units of a neural network to be statistically independent while they accomplish some other task; it is not a primary training criterion. 2) The nature of the competition is different. In predictability minimization, two networks' outputs are compared, with one network trying to make the outputs similar and the other trying to make the

当年Schmidhuber（唯一reject）对GAN原稿的评审意见

Finally, how is the submission related to the first work on "adversarial" MLPs for modeling data distributions through estimating conditional probabilities, which was called "predictability minimisation" or PM (Schmidhuber, NECO 1992)? The new approach seems similar in many ways. Both approaches use "adversarial" MLPs to estimate certain probabilities and to learn to encode distributions. A difference is that the new system learns to generate a non-trivial distribution in response to statistically independent, random inputs, while good old PM learns to generate statistically independent, random outputs in response to a non-trivial distribution (by extracting mutually independent, factorial features encoding the distribution). Hence the new system essentially inverts the direction of PM - is this the main difference? Should it perhaps be called "inverse PM"?

第一个对抗网络是PM?

Finally, how is the submission related to the first work on "adversarial" MLPs for modeling data distributions through estimating conditional probabilities, which was called "predictability minimisation" or PM (Schmidhuber, NECO 1992)? The new approach seems similar in many ways. Both approaches use "adversarial" MLPs to estimate certain probabilities and to learn to encode distributions. A difference is that the new system learns to generate a non-trivial distribution in response to statistically independent, random inputs, while good old PM learns to generate statistically independent, random outputs in response to a non-trivial distribution (by extracting mutually independent, factorial features encoding the distribution). Hence the new system essentially inverts the direction of PM - is this the main difference? Should it perhaps be called "inverse PM"?

GAN = “inverse PM” ?

Finally, how is the submission related to the first work on "adversarial" MLPs for modeling data distributions through estimating conditional probabilities, which was called "predictability minimisation" or PM (Schmidhuber, NECO 1992)? The new approach seems similar in many ways. Both approaches use "adversarial" MLPs to estimate certain probabilities and to learn to encode distributions. A difference is that the new system learns to generate a non-trivial distribution in response to statistically independent, random inputs, while good old PM learns to generate statistically independent, random outputs in response to a non-trivial distribution (by extracting mutually independent, factorial features encoding the distribution). Hence the new system essentially inverts the direction of PM - is this the main difference? Should it perhaps be called "inverse PM"?

LEARNING FACTORIAL CODES BY PREDICTABILITY MINIMIZATION

(*Neural Computation*, 4(6):863–879, 1992)

Jürgen Schmidhuber
Department of Computer Science
University of Colorado
Campus Box 430, Boulder, CO 80309, USA
yirgan@cs.colorado.edu

Learning Factorial Codes by Predictability Minimization

- Schmidhuber的态度：
  - 在2014年的NIPS评审中提出GAN只是PM基础上的变种
  - 私下里用邮件跟Goodfellow进行了一番往来争论
  - 在2016年的NIPS大会上公开打断Goodfellow进行对质

- Goodfellow的态度：
  - 在2014年的最终版本中加入了对PM的比较和引用
  - 但是在2016年的GAN Tutorial中完全移除
    - “我从不有否认GAN跟另外一些模型有联系，比如NCE，但是GAN跟PM之间我真的认为没太大联系。”（来源）
    - “Jürgen和我准备合写一篇paper来比较PM和GAN——如果我们能够取得一致意见的话。”（来源）

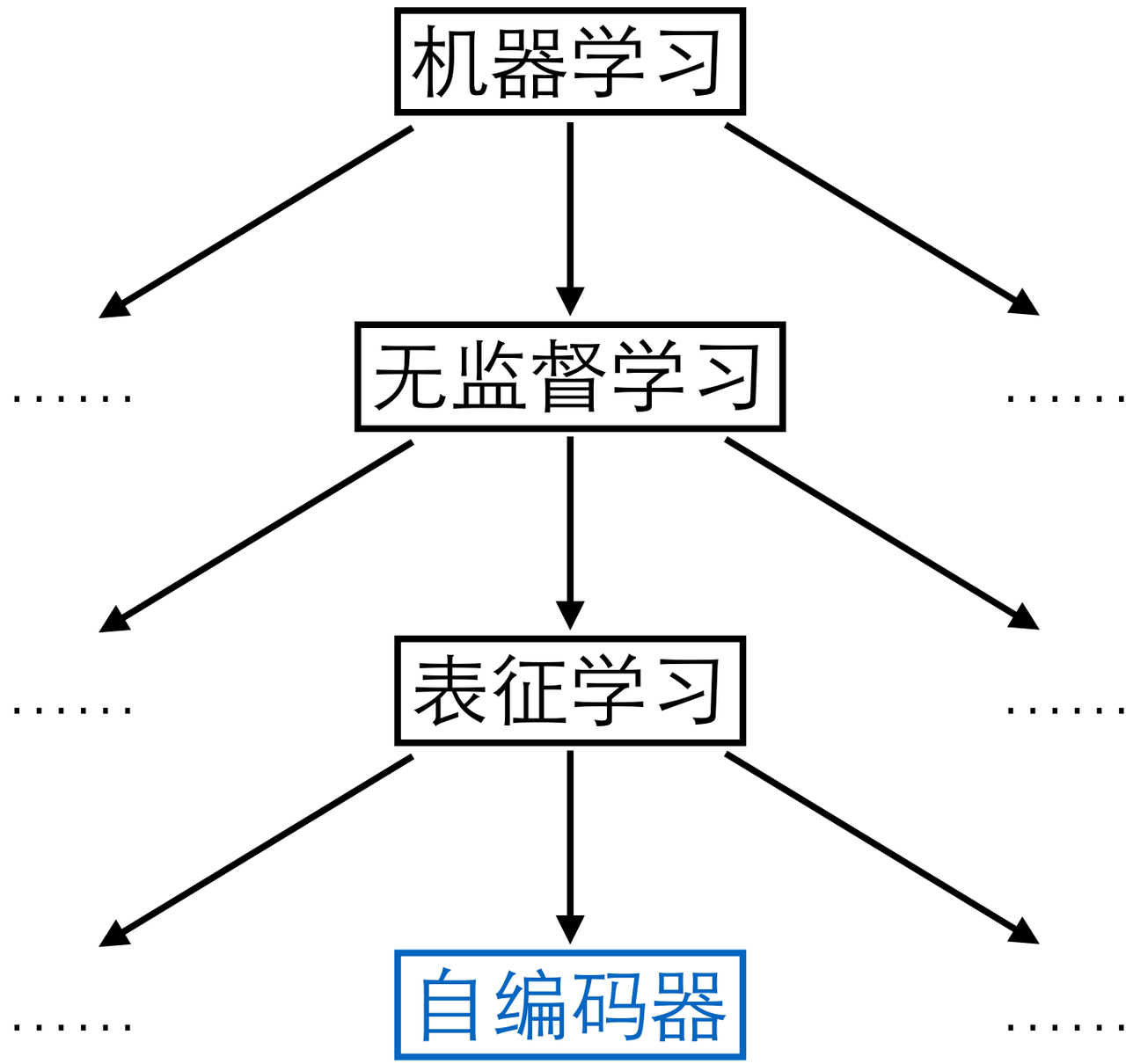
- Predictability Minimization（可预测性最小化）究竟是什么?

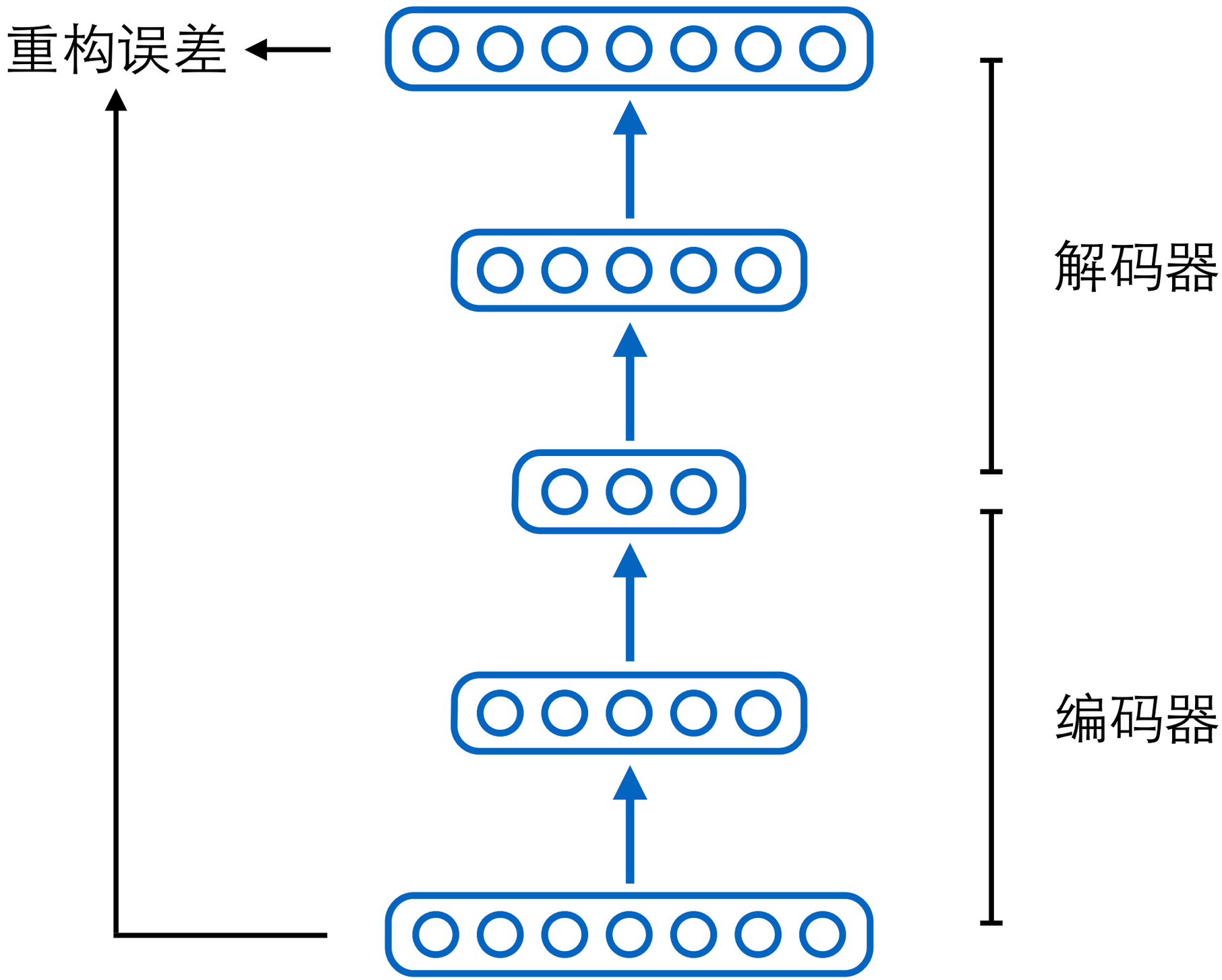
- PM究竟跟GAN有多少相同点?

- PM这个“古老”的模型能给我们今天的GAN研究带来新的启发吗?

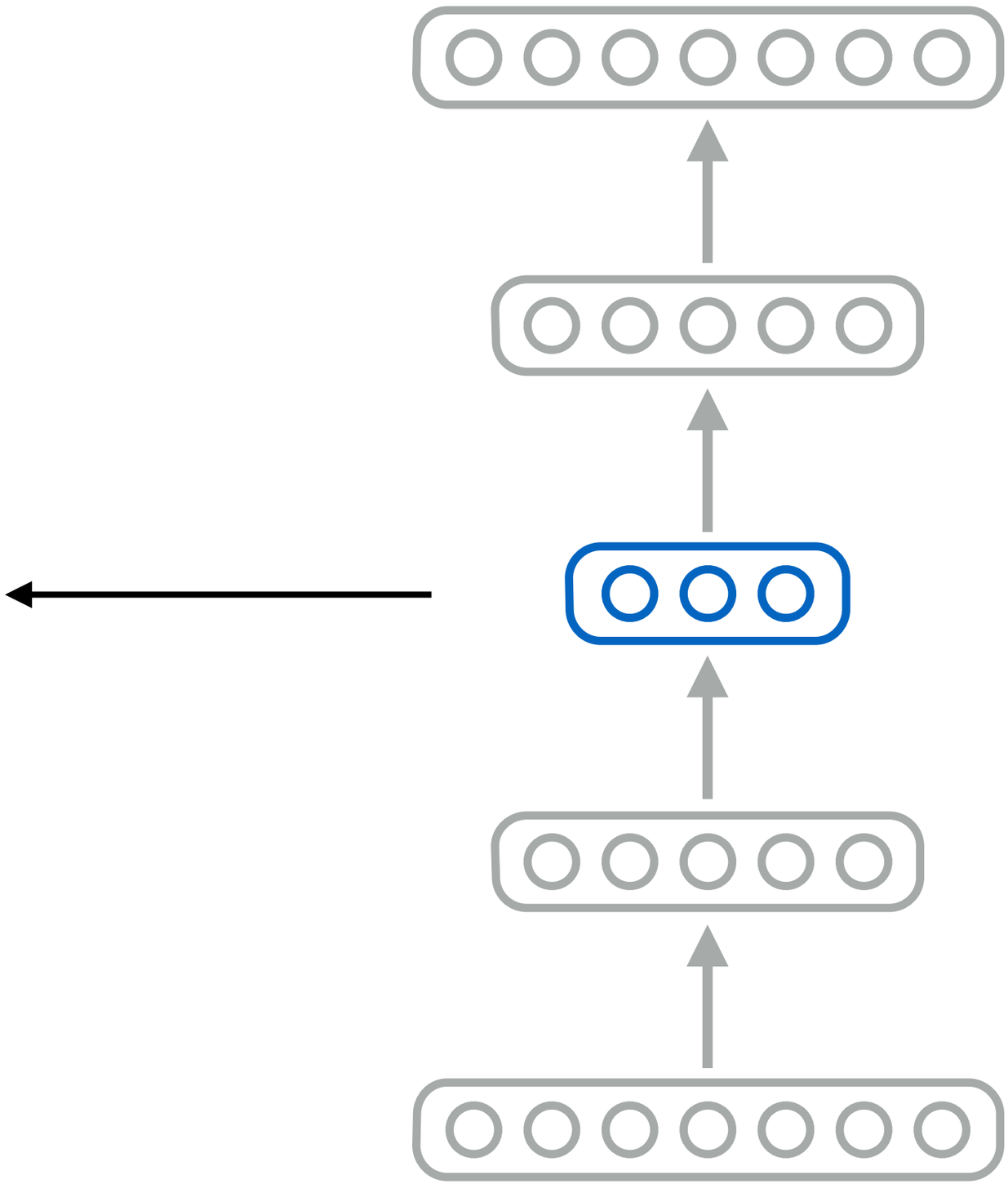
目录

- 八卦Schmidhuber与GAN之间的恩怨
- 讲解Schmidhuber在92年提出的PM模型
- 简单介绍GAN及其他模型
- 对比PM和GAN及其他模型之间的异同


机器学习
无监督学习
表征学习
自编码器
……
……
……
……
……
……


重构误差
解码器
编码器

表征（representation）
特征（feature）
编码（code）



- 表征学习的难点：我们不仅想要一个表征，还想要一个“好”的表征

- 什么是一个“好”的表征?
  - 稀疏（稀疏自编码器）
  - 理解数据而非死记硬背（降噪自编码器）
  - 解耦

- 解耦（factored / disentangled）即相互独立：

$$P(c_1, c_2, c_3|X) = P(c_1|X)P(c_2|X)P(c_3|X)$$

- 其中 X 是全部训练样本而非单个样本
- 直观意义：拆解样本中的各个独立要素
- 为简单起见，后面省略 X

c_1 c_2 c_3

- 换另一个说法:

$$P(c_1|c_2, c_3) = P(c_1)$$

$$P(c_2|c_1, c_3) = P(c_2)$$

$$P(c_3|c_1, c_2) = P(c_3)$$

- 含义：**一个维度的“兄弟”对于预测该维没有额外帮助**

- Schmidhuber是怎么实现这一点的呢?

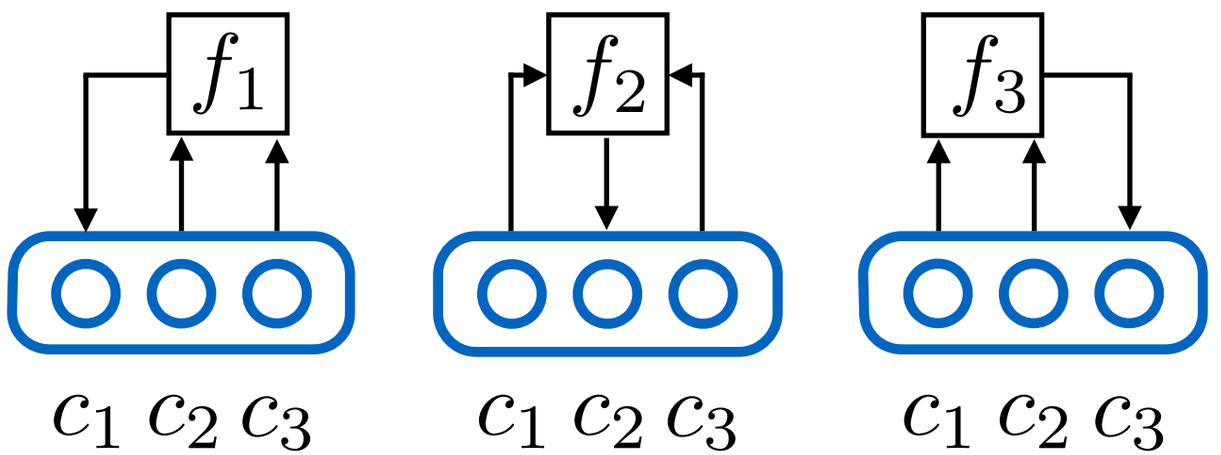
c_1 c_2 c_3

- 我们可以用3个预测器网络 f_1, f_2, f_3
- 预测器1预测维度1，输入维度2和3，以此类推

$$\hat{c}_1 = f_1(c_1, c_2)$$

$$\hat{c}_2 = f_2(c_1, c_3)$$

$$\hat{c}_3 = f_3(c_1, c_2)$$



- 以预测器1为例，我们用L2 loss让它试图预测对：

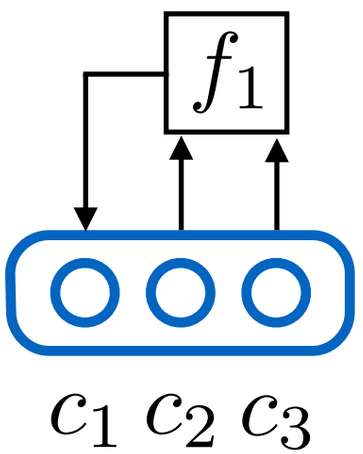
$$L(f_1) = \mathbb{E}[c_1 - \hat{c}_1]^2 = \mathbb{E}[c_1 - f_1(c_2, c_3)]^2$$

- 除了L2 loss，也可以定其他预测误差loss

- 如果 c_1 与 c_2, c_3 很有关系，f_1 就能猜得很准

- 说明 c_1 很有可预测性（**predictability**）

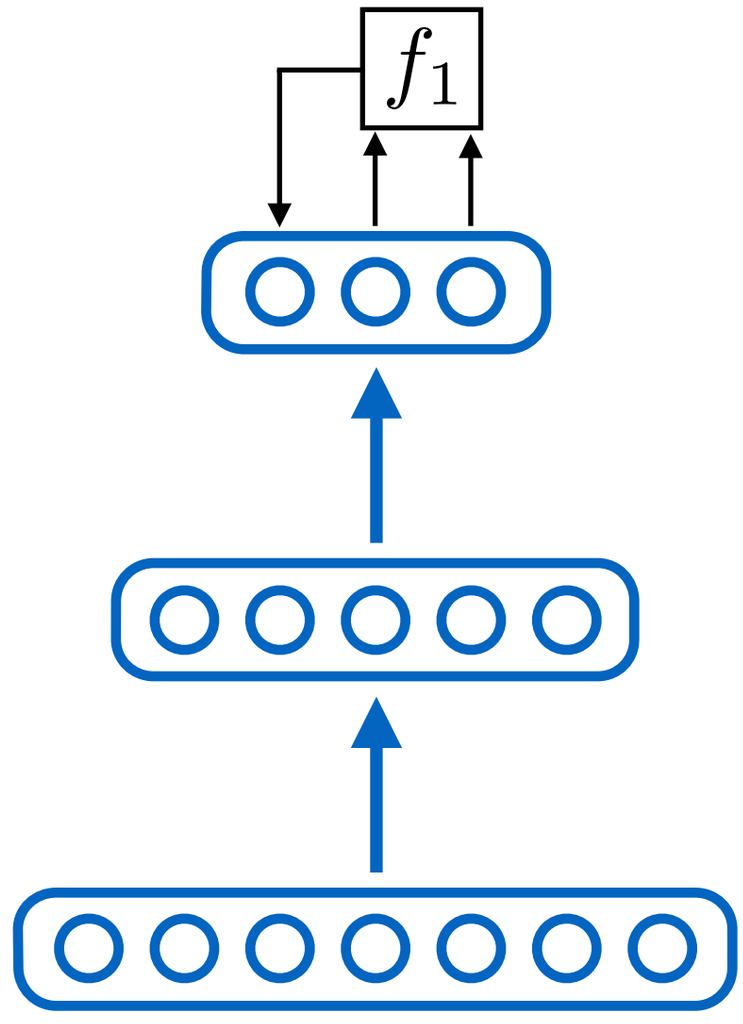
- 也说明 c_1 没能从 c_2, c_3 中解耦



- 为了将 c_1 与 c_2, c_3 解耦，编码器要让预测器猜不中

  $$L(Enc) = -\mathbb{E}[c_1 - f_1(c_2, c_3)]^2$$

- 如果做到，说明 c_1 与 c_2, c_3 关系不大



- 考虑所有维度，预测器试图**猜中**，代表了可预测性:

$$L(Pred) = \sum_i \mathbb{E}\big[c_i - f_i(c_1, \ldots, c_{i-1}, c_{i+1}, \ldots, c_N)\big]^2$$

- 编码器试图让预测器**猜不中**，最小化可预测性:

$$L(Enc) = -\sum_i \mathbb{E}\big[c_i - f_i(c_1, \ldots, c_{i-1}, c_{i+1}, \ldots, c_N)\big]^2$$

- 考虑所有维度，预测器试图**猜中**，代表了可预测性:

$$L(Pred) = \sum_i \mathbb{E}\big[c_i - f_i(c_1, \ldots, c_{i-1}, c_{i+1}, \ldots, c_N)\big]^2$$

- 编码器试图让预测器**猜不中**，最小化可预测性:

$$L(Enc) = -\sum_i \mathbb{E}\big[c_i - f_i(c_1, \ldots, c_{i-1}, c_{i+1}, \ldots, c_N)\big]^2$$

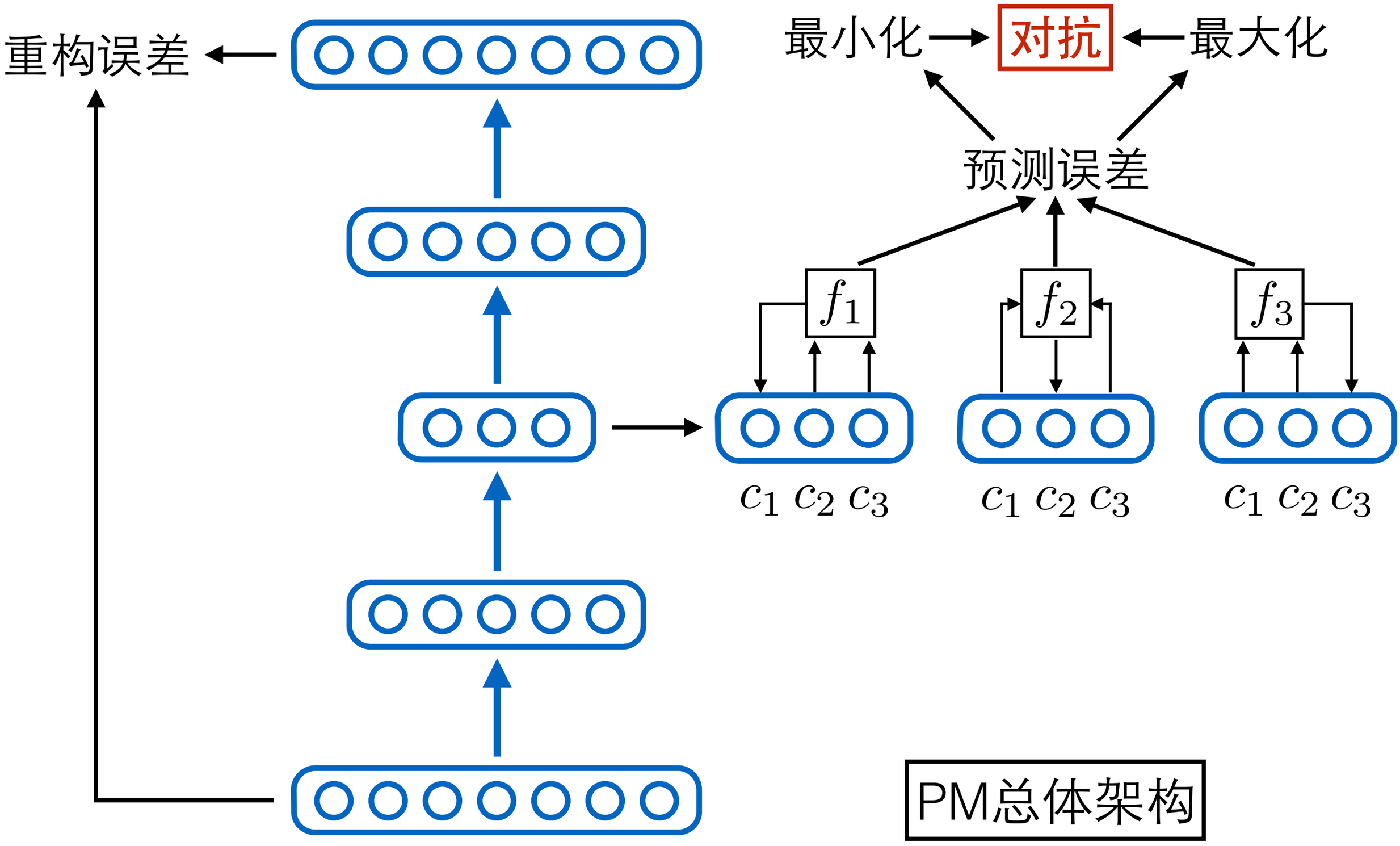
- 如果编码器“赢了”，就成功解耦了编码的各个维度

- 故名:

Learning Factorial Codes by Predictability Minimization

解耦编码　　可预测性最小化

- 当然别忘了还有个自编码的重构误差loss
- 从而保证编码中尽可能保留了原始输入的完整信息
- Schmidhuber的论文中还有其他loss，但不是重点
- 一图总结：



PM总体架构

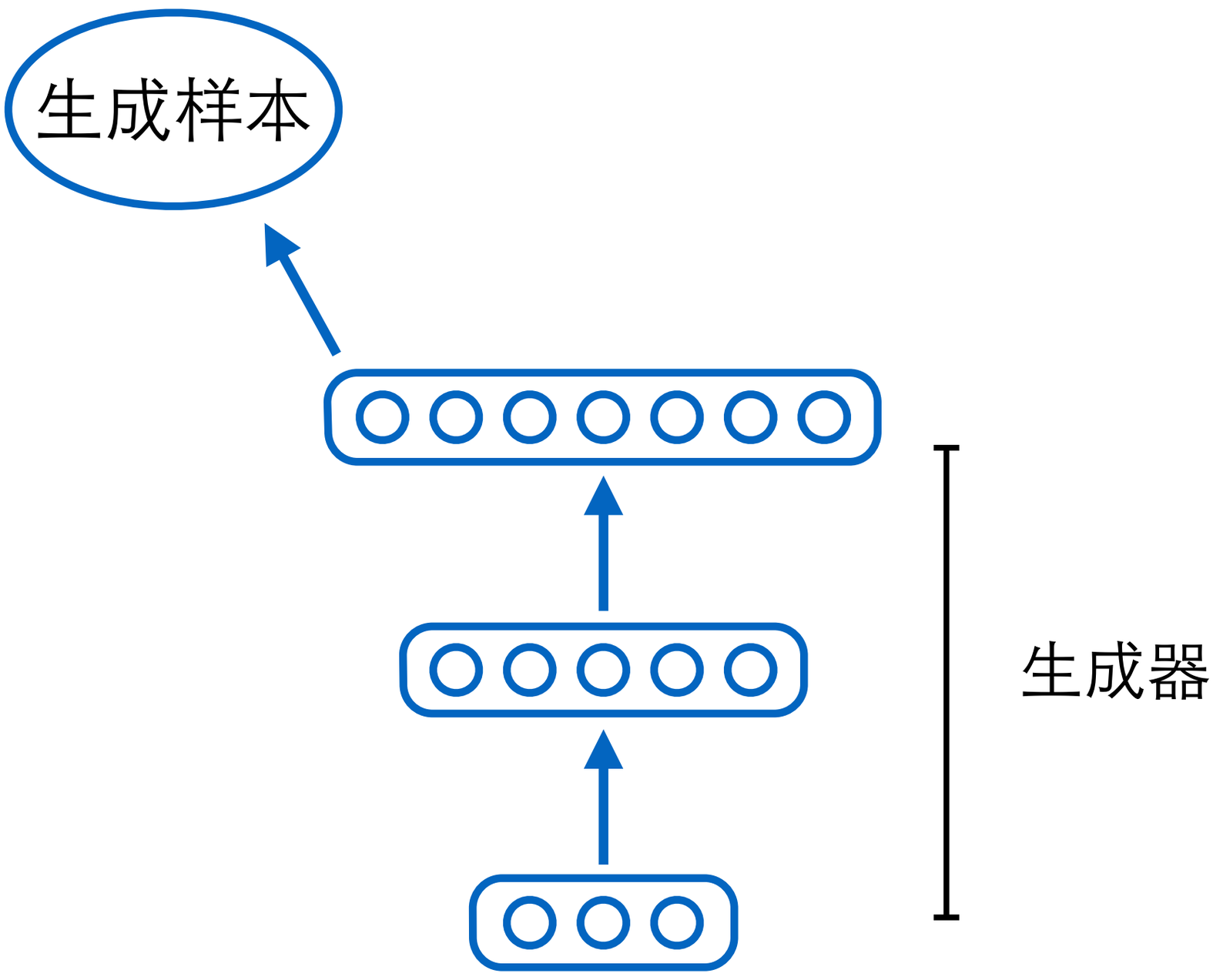
目录

- 八卦Schmidhuber与GAN之间的恩怨
- 讲解Schmidhuber在92年提出的PM模型
- 简单介绍GAN及其他模型
- 对比PM和GAN及其他模型之间的异同

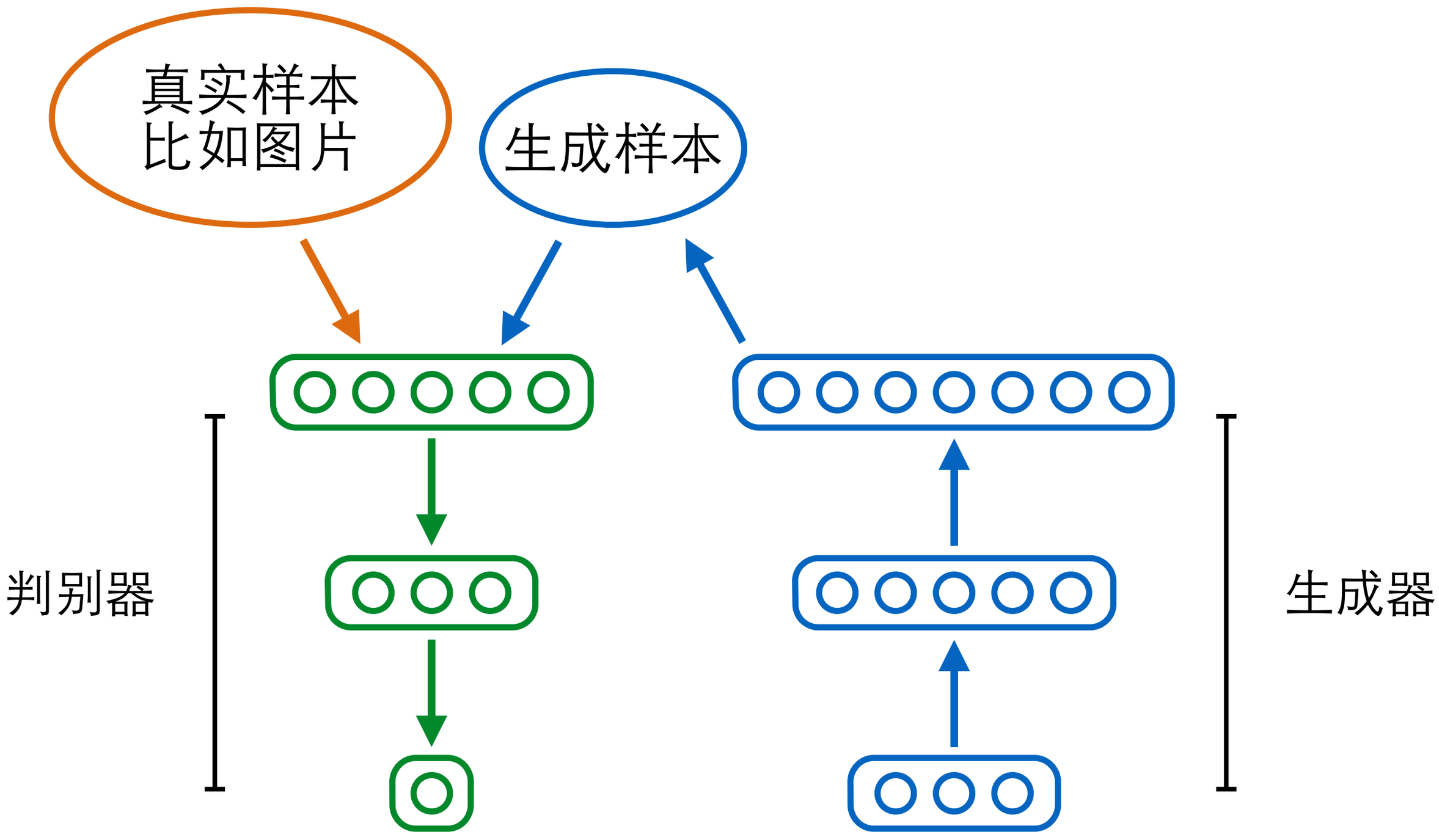
相关论文

- Generative Adversarial Nets
- InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets
- Adversarial Autoencoders

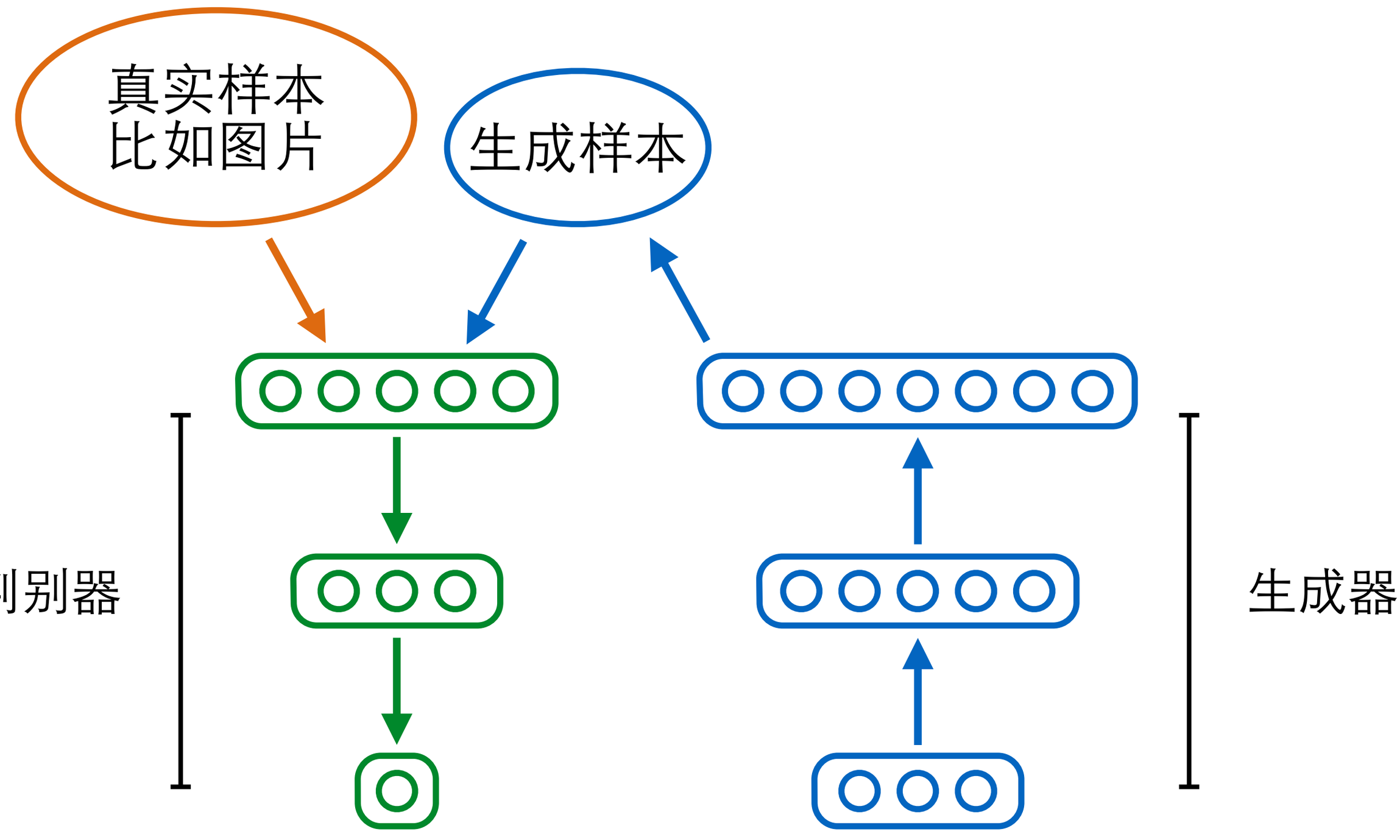
GAN



GAN



GAN



InfoGAN

重构部分code维度

真实样本
比如图片

生成样本

重构器

判别器

生成器

区分真假样本

让判别器区分不出真假样本

- 跟InfoGAN原论文的讲法不一样，但数学形式等价
- InfoGAN是从互信息开始推导，但最终结果变成：
  - 对于离散code维度得到对数似然的重构loss
  - 对于连续code维度得到L2的重构loss

InfoGAN

重构部分code维度

真实样本
比如图片

生成样本

重构器

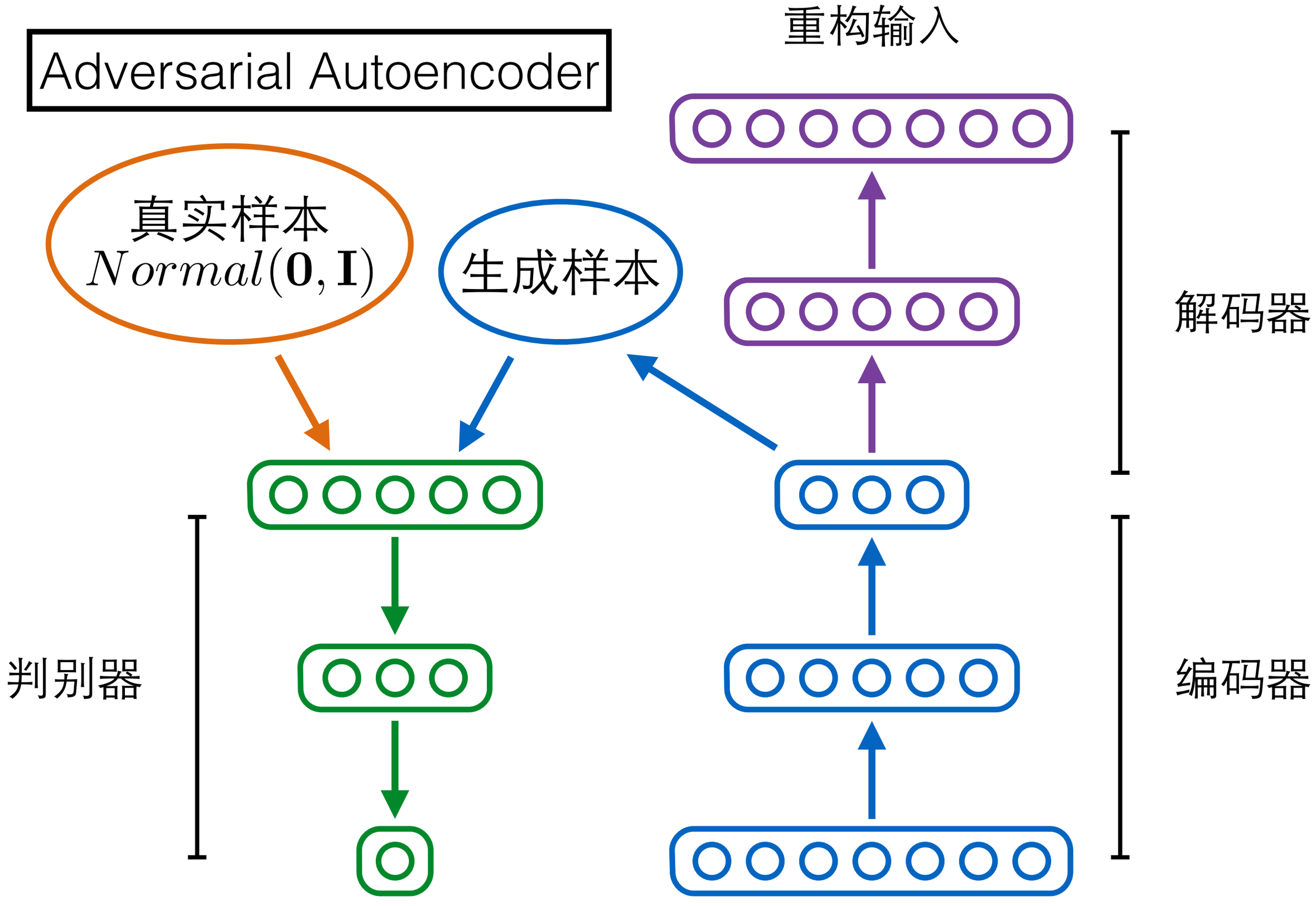
判别器

生成器

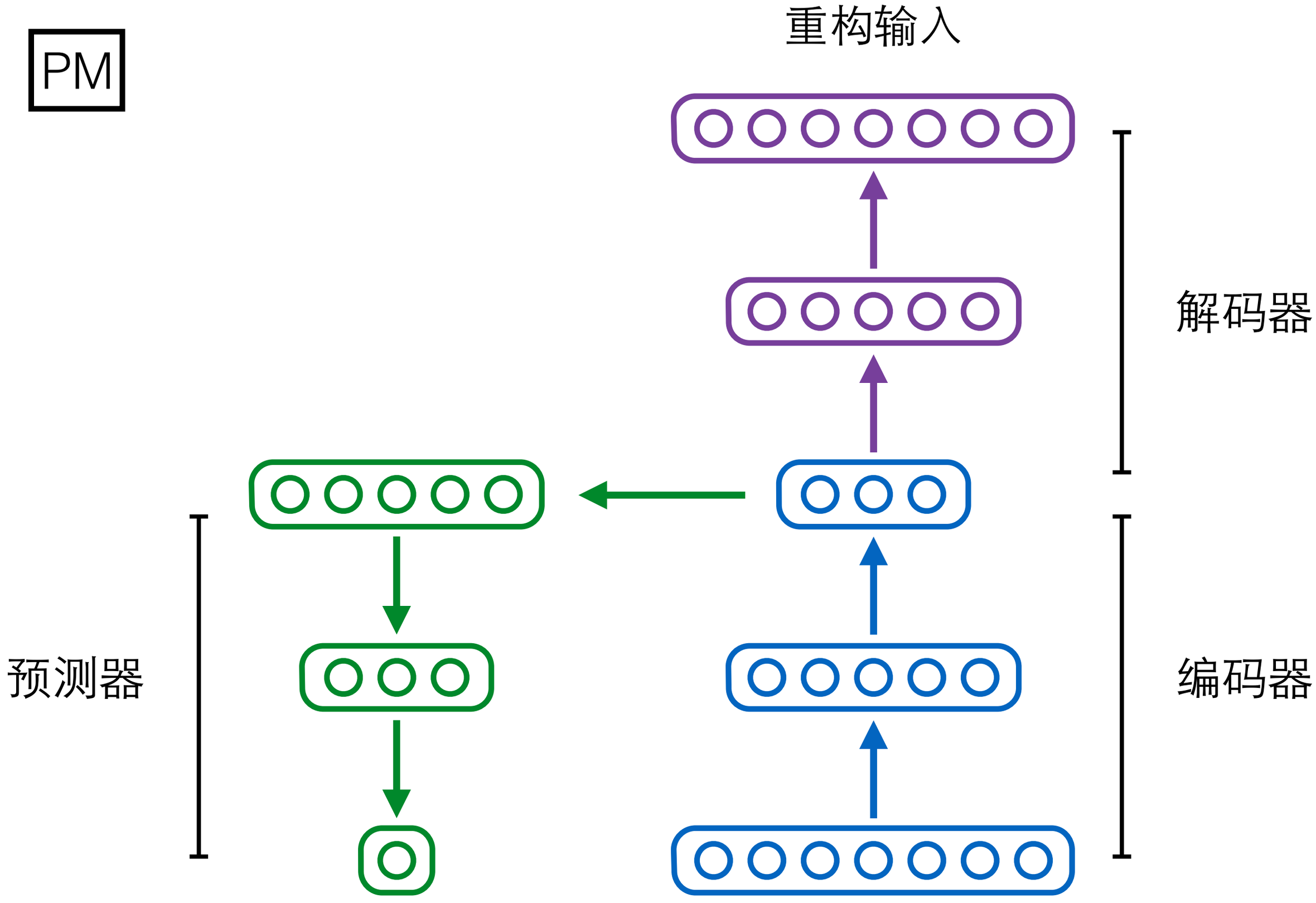
区分真假样本

让判别器区分不出真假样本

Adversarial Autoencoder



PM



给N-1个维度，预测维度i　　让预测器预测不准

PM

重构输入

解码器

预测器

编码器

给N-1个维度，预测维度i

让预测器预测不准

GAN

真实样本
比如图片

生成样本

判别器

生成器

区分真假样本

让判别器区分不出真假样本

Adversarial Autoencoder

重构输入

真实样本
$Normal(\mathbf{0}, \mathbf{I})$

生成样本

解码器

判别器

编码器

区分真假code

让判别器区分不出真假code

InfoGAN

重构部分code维度信息

真实样本
比如图片

生成样本

重构器

判别器

生成器

区分真假样本

让判别器区分不出真假样本

目录

- 八卦Schmidhuber与GAN之间的恩怨
- 讲解Schmidhuber在92年提出的PM模型
- 简单介绍GAN及其他模型
- 对比PM和GAN及其他模型之间的异同

PM

重构输入

解码器

预测器

编码器

给N-1个维度，预测维度i

让预测器预测不准

GAN

真实样本
比如图片

生成样本

判别器

生成器

区分真假样本

让判别器区分不出真假样本

Adversarial Autoencoder

重构输入

真实样本
$Normal(\mathbf{0}, \mathbf{I})$

生成样本

解码器

判别器

编码器

区分真假code

让判别器区分不出真假code

InfoGAN

重构部分code维度信息

真实样本
比如图片

生成样本

重构器

判别器

生成器

区分真假样本

让判别器区分不出真假样本

PM

PM的编码器：

复杂分布—>解耦分布

PM的解码器：

解耦分布—>复杂分布

GAN

GAN的生成器：

解耦高斯—>复杂分布

PM

PM的编码器：

复杂分布—>解耦分布

PM的解码器：

解耦分布—>复杂分布

GAN

GAN的生成器：

解耦高斯—>复杂分布

Schmidhuber把GAN称为“inverse PM”的原因

PM

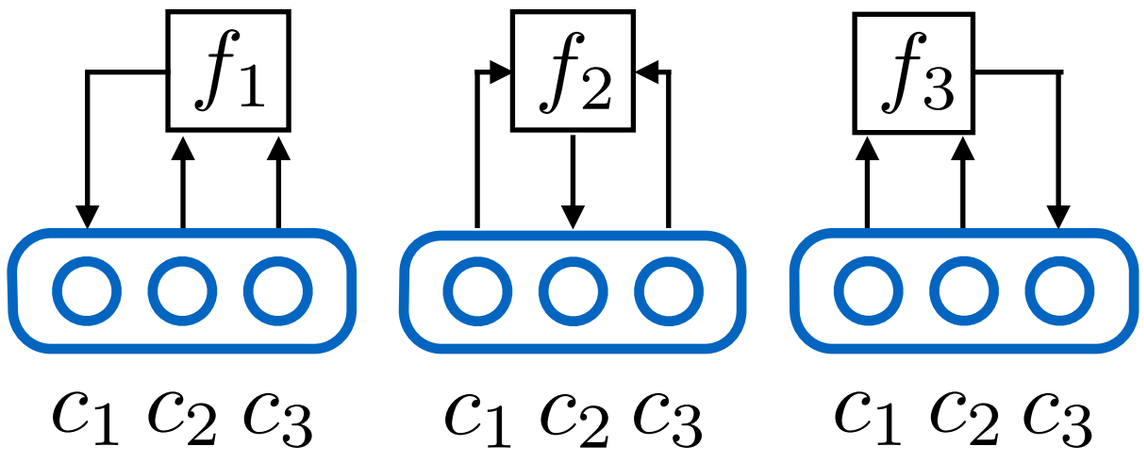
预测器和编码器**对抗优化相反的目标**：预测器要猜准某一维code，编码器要让它猜不准

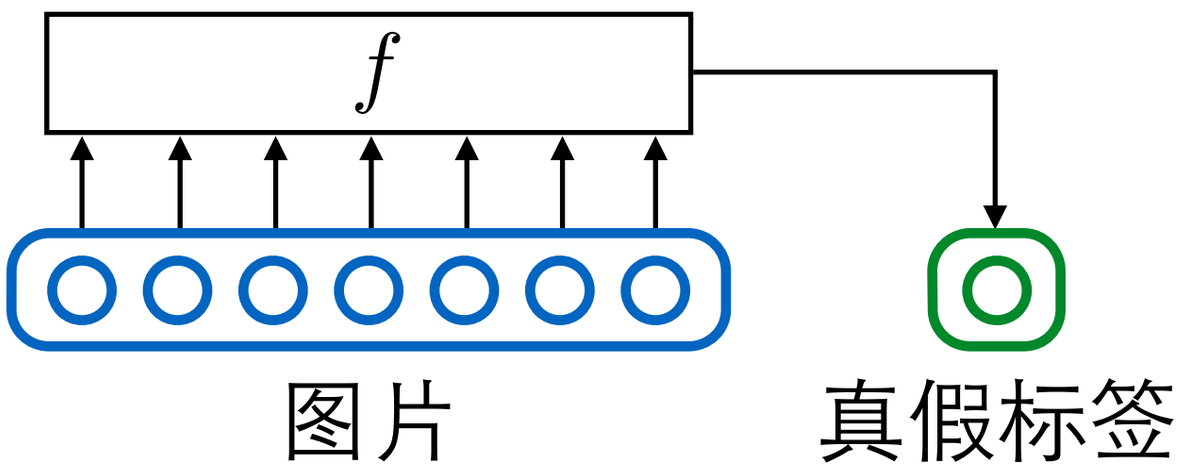
GAN

判别器和生成器**对抗优化相反的目标**，判别器要区分真假样本，生成器要让它区分不准

PM

GAN





PM

主体是自编码重构

对抗仅仅作为一个正则，
起辅助作用

GAN

主体就是对抗

没有其他目标

PM

只能把code建模为解耦分布，没办法建模为别的样子

GAN

虽然具体把code建模为解耦高斯分布，但是并没有特别偏好，直接就可以换成其他分布来建模code

甚至可以用另一个复杂分布作为code，比如图片

PM

重构输入

解码器

预测器

编码器

给N-1个维度，预测维度i

让预测器预测不准

GAN

真实样本
比如图片

生成样本

判别器

生成器

区分真假样本

让判别器区分不出真假样本

Adversarial Autoencoder

重构输入

真实样本
$Normal(\mathbf{0}, \mathbf{I})$

生成样本

解码器

判别器

编码器

区分真假code

让判别器区分不出真假code

InfoGAN

重构部分code维度信息

真实样本
比如图片

生成样本

重构器

判别器

生成器

区分真假样本

让判别器区分不出真假样本

PM

主体是自编码器

重构复杂分布的样本

Adversarial AE

主体是自编码器

重构复杂分布的样本

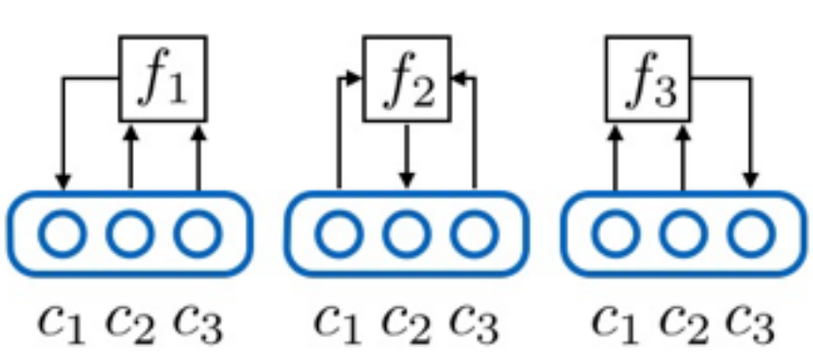
InfoGAN

主体可视为自编码器

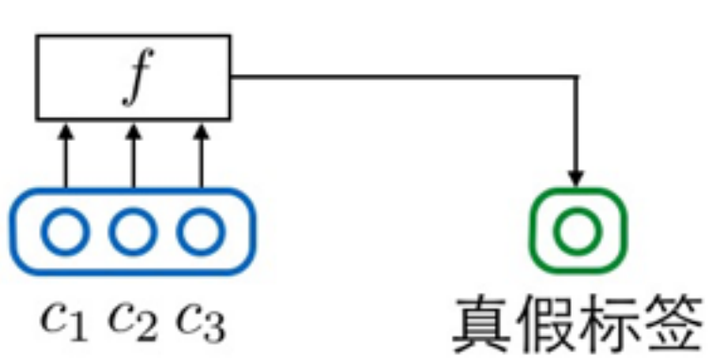
重构（部分）code

PM	Adversarial AE	InfoGAN
要求中间层解耦	要求中间层解耦高斯	要求中间层满足复杂样本分布
通过最小化可预测性解耦code分布	通过对抗拉近解耦高斯分布与code分布	通过对抗拉近生成分布与真实复杂分布

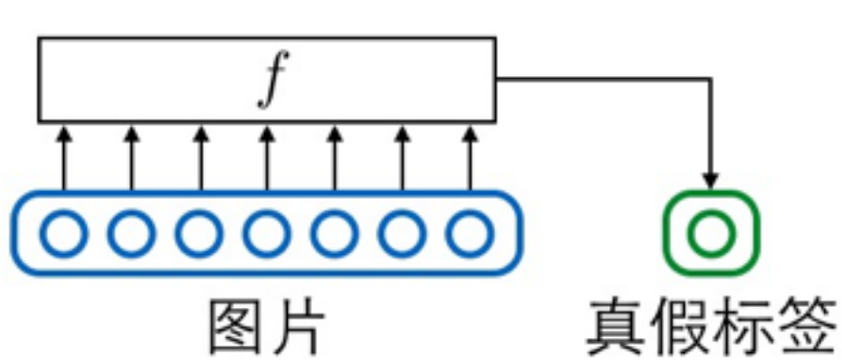
PM



Adversarial AE



InfoGAN



对抗自编码器是“杀鸡用牛刀”？ （原文链接）

inFERENCe

posts on machine learning, statistics, opinions on things I'm reading in the space

About

关于生成模型的著名博客，推荐

So there are multiple reasons why one might want to control the aggregate posterior $q(\mathbf{z})$ to match a predefined prior $p(\mathbf{z})$. The authors achieve this by introducing an additional term in the autoencoder loss function, one that measures the divergence between q and p. The authors chose to do this via adversarial training: they train a discriminator network that constantly learns to discriminate between real code vectors $\mathbf{z}$ produced by encoding real data, and random code vectors sampled from p. If q matches p perfectly, the optimal discriminator network should have a large classification error.

Is this an overkill?

My main question about this paper was whether the adversarial cost is really needed here, because I think it's an overkill. Let me explain:

Adversarial training is powerful when all else fails to quantify divergence between complicated, potentially degenerate distributions in high dimensions, such as images or video. Our toolkit for dealing with images is limited, CNNs are the best tool we have, so it makes sense to incorporate them in training generative models for images. GANs - when applied directly to images - are a great idea.

However, here adversarial training is applied to an easier problem: to quantify the divergence between a simple, fixed prior (e.g. Gaussian) and an empirical distribution of latents. The latent space is usually lower-dimensional, distributions better behaved. Therefore, matching to $p(\mathbf{z})$ in latent space should be considerably easier than matching distributions over images.

- GAN：“牛刀”
- 复杂分布（比如图片）：“牛”
- 简单分布（如解耦高斯分布）：“鸡”

- 要让一个分布像解耦高斯分布不需要动用GAN!
- 这么简单的任务可以用更高效的方式（博客作者提了MMD，此处略）

- 另一种方式：分别要求解耦和高斯
  - 解耦用PM
  - 解耦完每个维度的高斯用GAN，标量的GAN训练可能比向量的GAN训练要容易和稳定
- 进一步体现了PM和对抗自编码器的联系

总结

- Schmidhuber在92年提出的PM模型比GAN更早地使用了“对抗”的思想
- PM不仅跟GAN，还跟Adversarial Autoencoder、InfoGAN有很多相似之处，但还是存在很明显的差异
- 这些相似和差异也许能够启发出一些新模型
- 更多像PM这样“古老”但有趣的想法不应该被埋没

Q&A