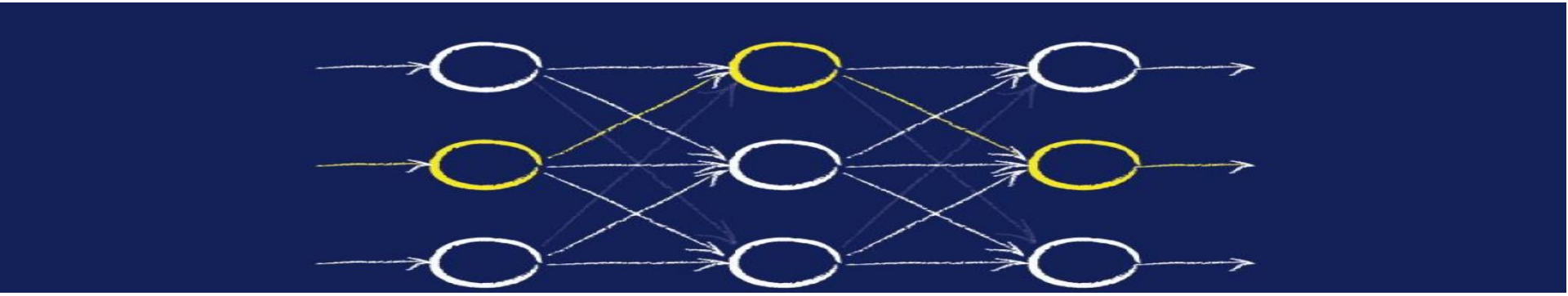
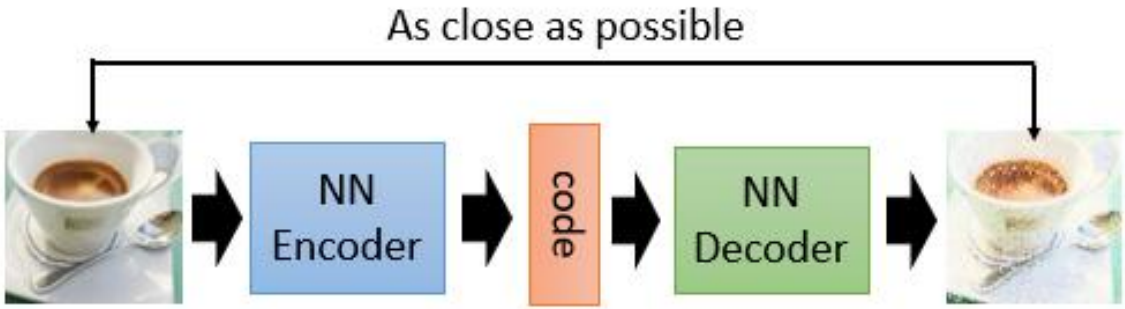
表示学习和生成式模型

（representation learning and generative model）



0、引入

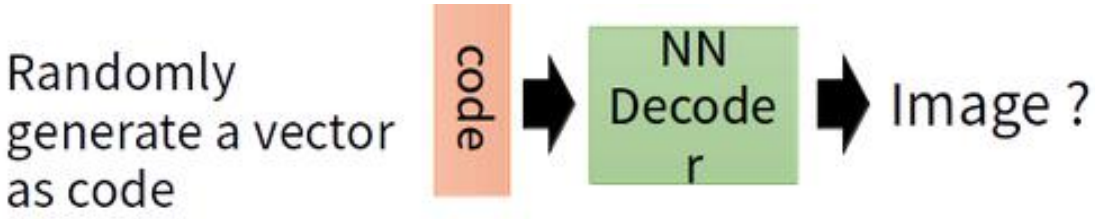
自编码器（autoencoder）是神经网络的一种，经过训练后能尝试将输入复制到输出。**自编码器**（autoencoder）内部有一个隐藏层 $\boldsymbol{h}$，可以产生**编码**（code）表示输入。该网络可以看作由两部分组成：一个由函数 $\boldsymbol{h} = f(\boldsymbol{x})$ 表示的编码器和一个生成重构的解码器 $\boldsymbol{r} = g(\boldsymbol{h})$。



$$L(\boldsymbol{x}, g(f(\boldsymbol{x})))$$

从图6.1中能够看到两个部分：第一个部分是编码器（Encoder），第二个部分是解码器（Decoder），编码器和解码器都可以是任意的模型，通常使用神经网络模型作为编码器和解码器。输入的数据经过神经网络降维到一个编码（code），接着又通过另外一个神经网络去解码得到一个与输入原数据一模一样的生成数据，然后通过比较这两个数据，最小化它们之间的差异来训练这个网络中编码器和解码器的参数。当这个过程

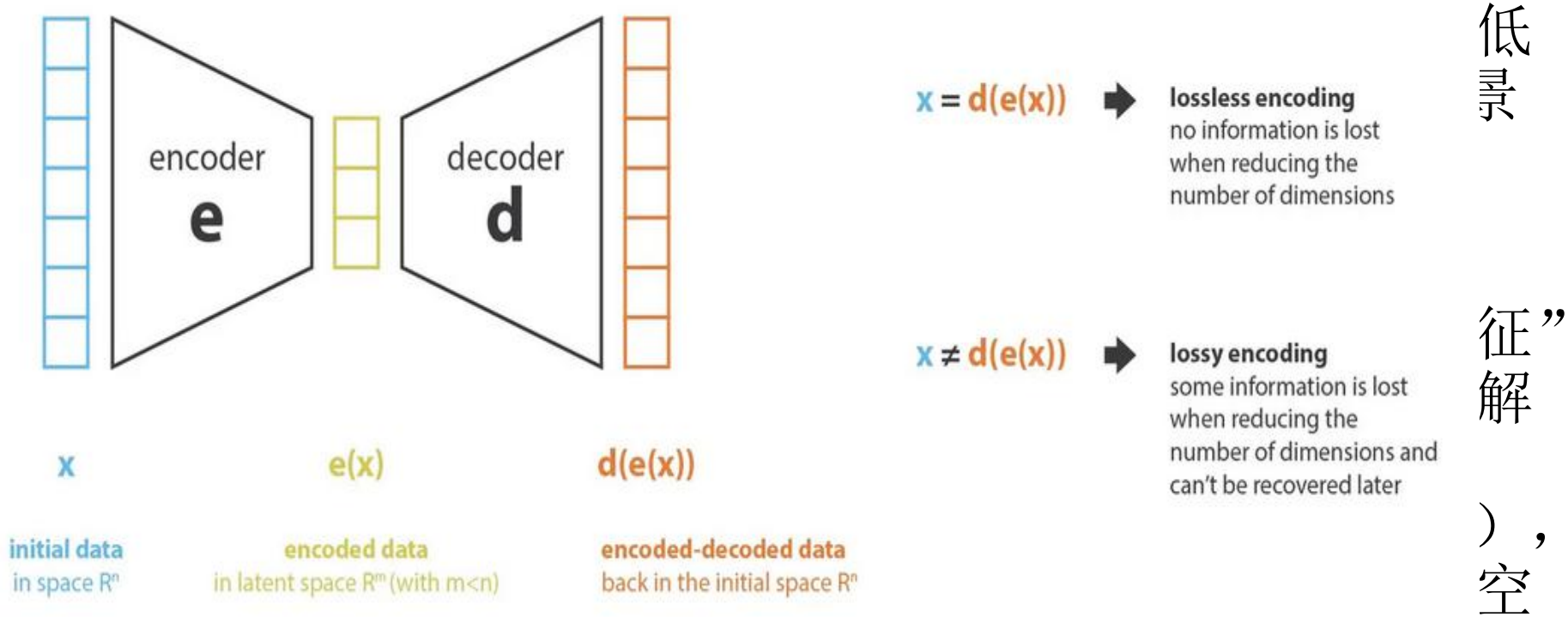
训练完之后，拿出这个解码器，随机传入一个编码（code），通过解码器能够生成一个和原数据差不多的数据，图6.2就是希望能够生成一张差不多的图片。



传统自编码器被用于降维或特征学习。近年来，自编码器与潜变量模型理论的联系将自编码器带到了生成式建模的前沿。

什么是降维？

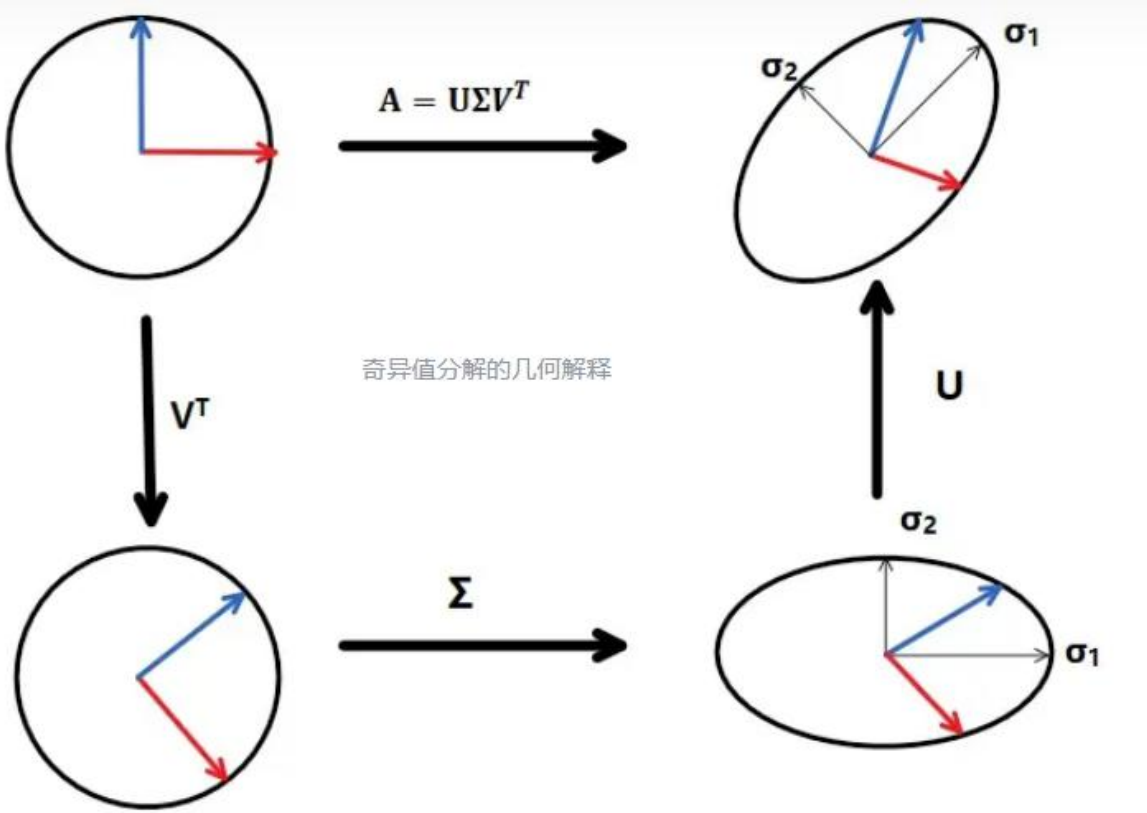
- 降维是减少描述数据的特征数量的过程。可以通过选择（仅保留一些现有特征）或通过提取（基于旧特征组合来低景



征”解），空间大小和编码器的选择，压缩可能是有损的，即一部分信息会在编码过程中丢失，并且在解码时无法恢复。

奇异值分解（Singular Value Decomposition,SVD）

SVD是一种
都可以将其
分别是m阶
的m×n矩
解求得特征
原矩阵（复
括：通过分
果矩阵中保
量，即可解
据降维。



奇异值分解的几何解释

原始向量空间的**标准正交基**（红色与蓝色），经过坐标系的**旋转变换** V^T 、坐标轴的**缩放变换** Σ（黑色 σ_1, σ_2 ）、坐标系的**旋转变换** U ，这个过程所得到和经过一个**线性变换A**结果是等价的。

矩阵，
形式，
组成
SVD分
示出
来概
解结
征向
现数

1、表示学习（representation learning）

- 尽管当今许多公司都拥有大量数据，但其中绝大多数数据通常是非结构化和未标记的。事实上，针对特定业务需求适当标记的数据量通常非常小（甚至可能为零），获取新标签通常是一项缓慢而昂贵的工作。因此，能够从未标记的数据中提取特征以提高数据有限任务性能的算法非常有价值。

1、表示学习（representation learning）

- 大多数机器学习从业者首先通过无监督学习接触特征提取技术。在无监督学习中，算法试图在某些（显式或隐含）假设下发现描述数据集“结构”的潜在特征。例如，低秩奇异值分解（主成分分析是一个具体示例）将数据矩阵分解为三个简化秩矩阵，以最小化重建数据矩阵的平方误差。

1、表示学习（representation learning）

- 尽管传统的无监督学习技术将永远是机器学习管道的主要内容，但随着深度学习的持续成功，表示学习已成为特征提取的替代方法。在表示学习中，通过在辅助监督学习任务上训练神经网络，从未标记的数据中提取特征。
- 由于其受欢迎程度，word2vec已经成为表示学习事实上的“Hello，world！”应用。将深度学习应用于自然语言处理（NLP）任务时，模型必须同时学习多个语言概念。

Deep Learning for NLP:

Word Vectors and Lexical Semantics

词向量模型——

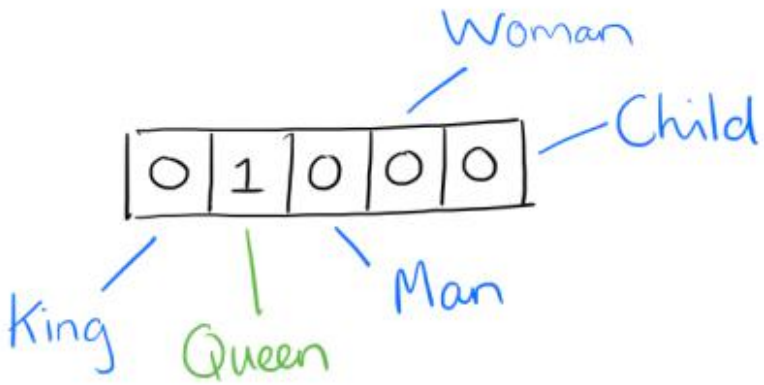
Word2Vec

How to represent words?

- 自然语言文本=离散符号序列（e.g. 词）
- Naive representation:

one-hot in $R^{|vocabulary|}$ (very large).

one 1, the rest 0s



1-of-N encoding

[1,0,0,0,0]
[0,1,0,0,0]
[0,0,1,0,0]
[0,0,0,1,0]
[0,0,0,0,1]

Problem with words as discrete symbols

- 问题:

  稀疏, 正交, 弱语义

- Example:

如果搜索“Seattle motel”，希望匹配包含“Seattle hotel”的文档。

But:

motel = [0 0 0 0 0 0 0 0 0 0 1 0 0 0 0]

hotel = [0 0 0 0 0 0 0 1 0 0 0 0 0 0 0]

这两个向量是orthogonal（正交的）.

one-hot向量不能表示similarity（相似）!

- Solution:

learn to encode similarity in the vectors themselves

Representing words by their context

- We want：

1.这种表示包含对应语言单元（词或文档）的语义信息。

2.这种表示直接度量文本之间的语义相似度。

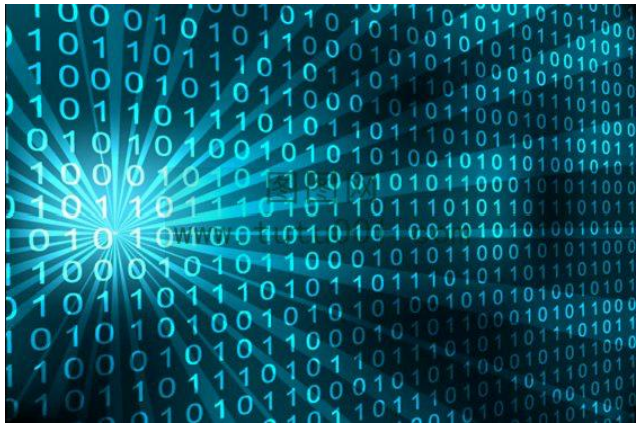
- Dristributed representation

"words that occur in the same contexts tend to have similar meanings. "—Z.S.Harris(1954)

"You shall know a word by the company it keeps." — J.R. Firth (1957)

Distributed Representation

易于数值计算的知识表示：分布式表示



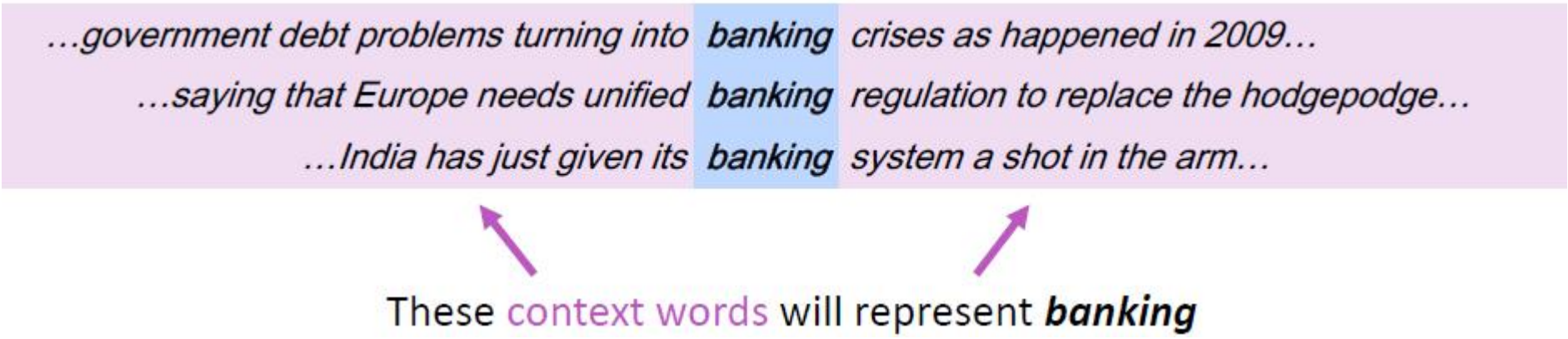
- 语言符号和知识符号
  - 符号逻辑系统
- 计算机擅长处理什么
  - 二进制：进行布尔运算
  - 数字：进行数值计算
- 自然语言处理
  - 用计算机能处理的方式表示和处理自然语言

使用数值表示自然语言中的概念和结构
通过数值计算进行语义计算

Distributed Representation

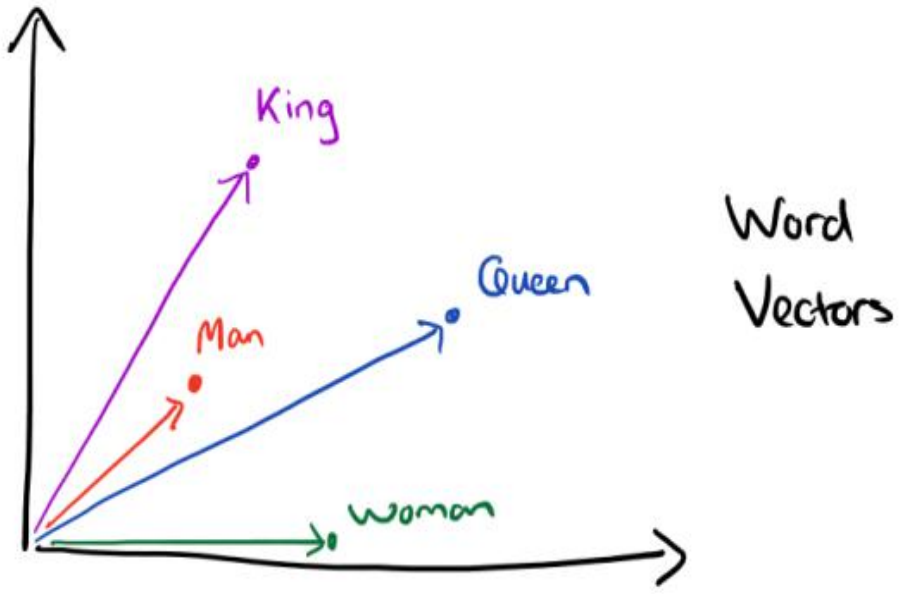
一个最成功的思想就是 modern statistical NLP!

- 当词w出现在文本中时，其上下文是出现在附近的单词集
- 使用w的许多上下文来构建w的表示



Distributed Representation

- 为每个词建立稠密的向量, 使得出现在相似上下文中的词向量也相似



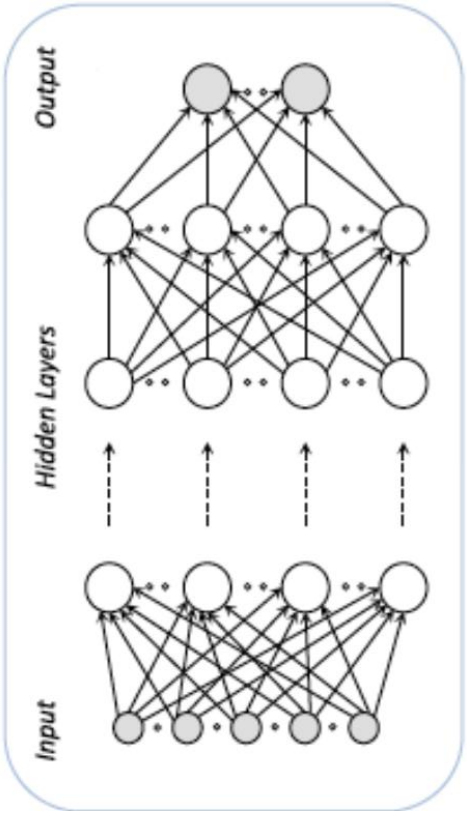
$$\overrightarrow{King} - \overrightarrow{Man} + \overrightarrow{Woman} = \overrightarrow{Queen}$$

Deep Neural Networks Are Our Friends

表示：特征工程→特征学习（表示学习）

Deep Learning

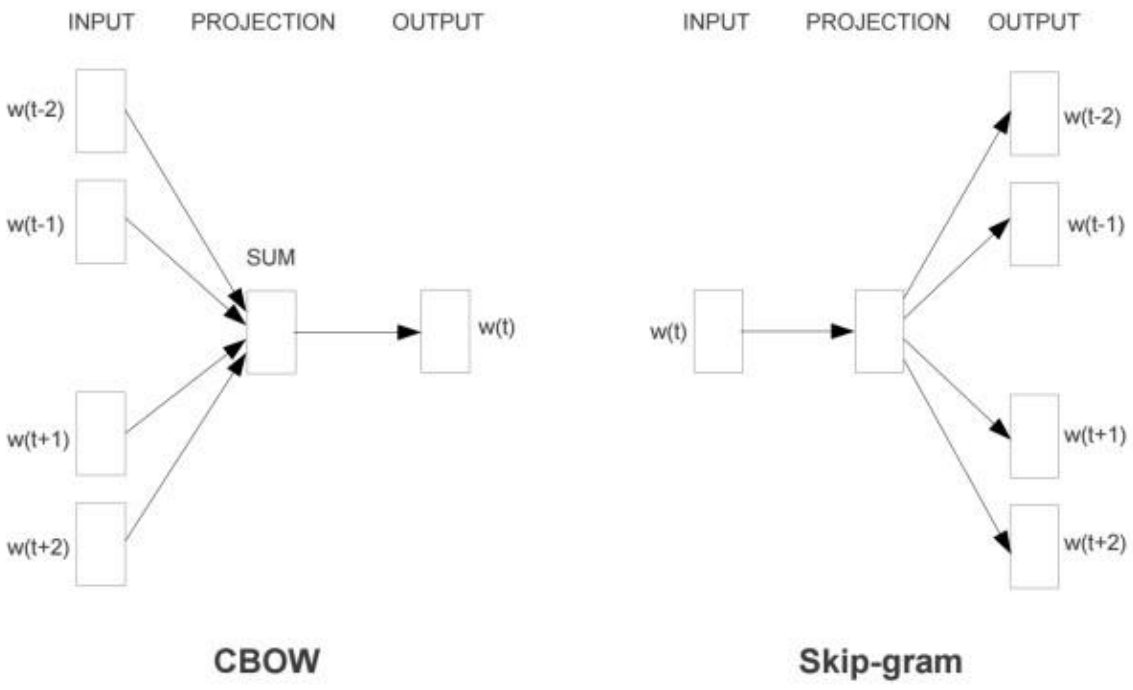
为什么需要研究深度学习



- 手工特征耗时耗力，且不易拓展
- 自动特征学习快，方便拓展
- 深度学习提供了一种通用的学习框架，可用来表示世界、视觉和语言学信息
- 深度学习既可以无监督学习，也可以监督学习

Neural Embedding Models

- 神经网络词向量表示技术通过神经网络技术对上下文，以及上下文与目标词之间的关系进行建模。
- Word2Vec从大量文本语料中以无监督的方式学习语义知识。其中，提出了Skip-Gram和CBOW两种模型。



Mikolov T, et al. Distributed representations of words and phrases and their compositionality. NIPS, 2013.

Statistical Language Model

P(**high** price) > P(**large** price)

建立语言模型，即用来计算一个句子的概率模型

Example：

我/今天/下午/打/篮球

$P(S)=P(w_1,w_2,w_3,w_4,w_5,\ldots,w_n)$

$=P(w_1)\ P(w_2|w_1)\ P(w_3|w_1,w_2)\ldots P(w_n|w_1,w_2,\ldots,w_{n-1})$

Window based cooccurence matrix

- Example corpus:
  - I like deep learning.
  - I like NLP.
  - I enjoy flying.

counts	I	like	enjoy	deep	learning	NLP	flying	.
I	0	2	1	0	0	0	0	0
like	2	0	0	1	0	1	0	0
enjoy	1	0	0	0	0	0	1	0
deep	0	1	0	0	1	0	0	0
learning	0	0	0	1	0	0	0	1
NLP	0	1	0	0	0	0	0	1
flying	0	0	1	0	0	0	0	1
.	0	0	0	0	1	1	1	0

Statistical Language Model

- 统计语言模型的作用是为一个长度为m的字符串确定一个概率分布P(w_1,w_2,...,w_m)，表示其存在的可能性，其中w_1到w_m依次表示这段文本中的各个词。一般在实际求解过程中，通常采用下式计算其概率值：

$$P(w_1, w_2, ..., w_m) = P(w_1)\ P(w_2|w_1)\ P(w_3|w_1, w_2)$$
$$\ldots P(w_i \mid w_1, w_2, ..., w_{i-1}) \ldots P(w_m \mid w_1, w_2, ..., w_{m-1})$$

- 研究者们提出使用一个简化模型：n 元模型（n-gram model）。在n元模型中估算条件概率时，距离大于等于n的上文词会被忽略，也就是对上述条件概率做了以下近似：

$$P(w_i \mid w_1, w_2, ..., w_{i-1}) \approx P(w_i \mid w_{i-(n-1)}, \ldots, w_{i-1})$$

Statistical Language Model

$$P(w_1, \ldots, w_m) = \prod_{i=1}^{m} P(w_i \mid w_1, \ldots, w_{i-1}) \approx \prod_{i=1}^{m} P(w_i \mid w_{i-(n-1)}, \ldots, w_{i-1})$$

- To estimate probabilities, compute for unigrams and bigrams (conditioning on one/two previous word(s):

$$p(w_2|w_1) = \frac{\text{count}(w_1, w_2)}{\text{count}(w_1)} \qquad p(w_3|w_1, w_2) = \frac{\text{count}(w_1, w_2, w_3)}{\text{count}(w_1, w_2)}$$

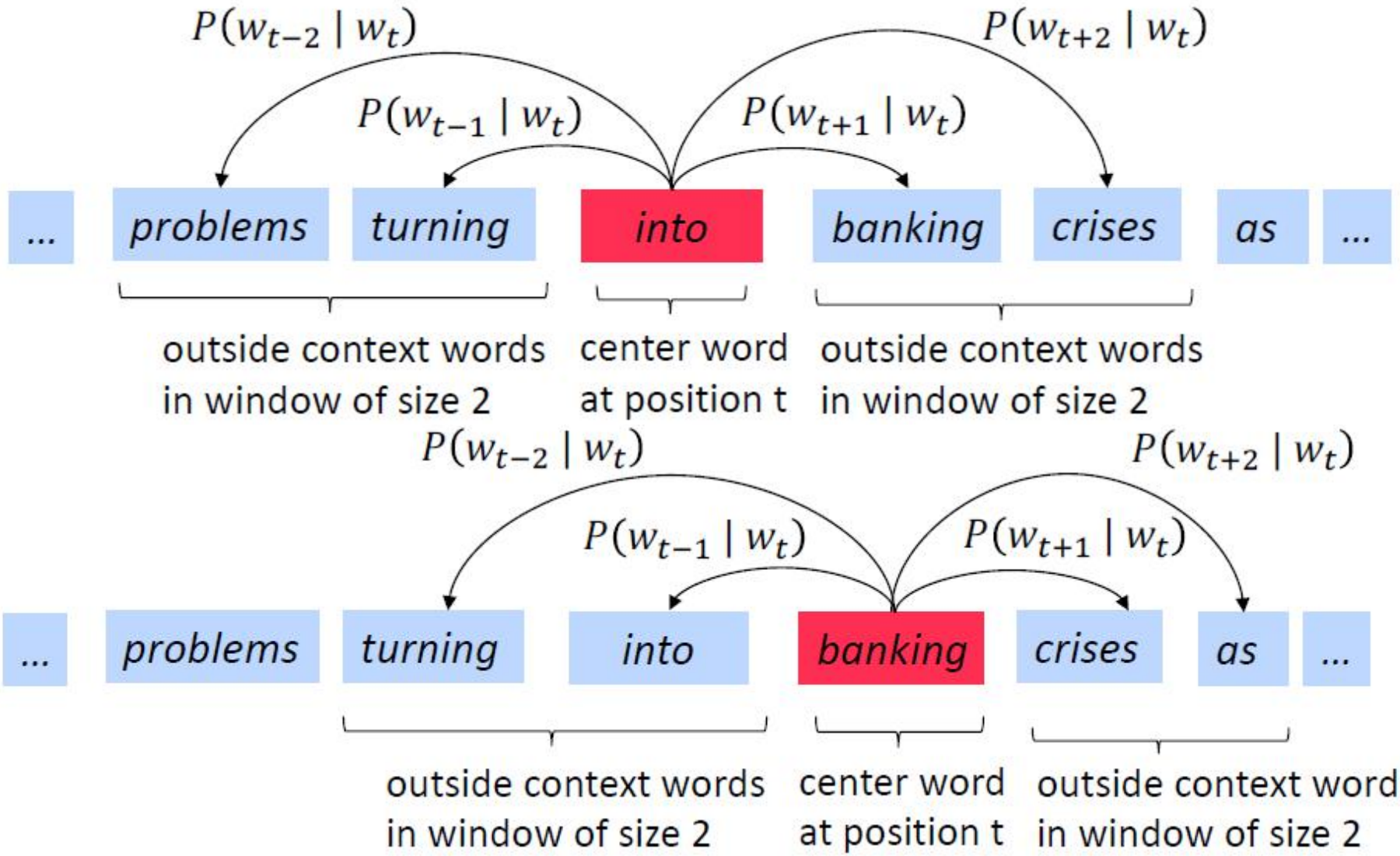
Word2Vec： Overview

Idea:

- 需要一个大的文本corpus（语料库）
- 在固定词汇表中的每个词被表示为一个vector（向量）
- 遍历文本中的每个位置*t*, 其中含有一个center word（中心词）*c*和它的context words（上下文词）*o*
- 使用*c*和*o*词向量的similarity（相似度）计算在给定*c*的条件下*o*的probability（概率）
- Adjust（调整）词向量以最大化这个概率

Word2Vec： Overview

Example windows and process for computing $P(w_{t+j} \mid w_t)$



Word2vec: objective function

- 给定训练词序列$w_1,w_2,w_3...,w_T$，训练目标是在固定词窗m中，从中心词w_t预测上下文词，即：

Likelihood = $L(\theta) = \prod_{t=1}^{T} \prod_{\substack{-m \le j \le m \\ j \neq 0}} P(w_{t+j} \mid w_t; \theta)$

θ is all variables to be optimized

sometimes called *cost* or *loss* function

The objective function $J(\theta)$ is the (average) negative log likelihood:

- $$J(\theta) = -\frac{1}{T}\log L(\theta) = -\frac{1}{T}\sum_{t=1}^{T} \sum_{\substack{-m \le j \le m \\ j \neq 0}} \log P(w_{t+j} \mid w_t; \theta)$$

Minimizing objective function ⟺ Maximizing predictive accuracy

Word2vec: objective function

$$J(\theta) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{\substack{-m\le j\le m \\ j\neq 0}} \log P(w_{t+j} \mid w_t; \theta)$$

- Question: How to calcu $P(w_{t+j} \mid w_t; \theta)$?

  - Answer: 每个词w使用两个向量表示：

    v_w when w is a center word

    u_w when w is a context word

  - 对于中心词 c 及上下文词 o:

$$P(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{w\in V}\exp(u_w^T v_c)}$$

Word2vec: objective function

Exponentiation makes anything positive

$$P(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{w \in V} \exp(u_w^T v_c)}$$

Dot product compares similarity of *o* and *c*.

$u^T v = u.v = \sum_{i=1}^{n} u_i v_i$

Larger dot product = larger probability

Normalize over entire vocabulary to give probability distribution

- This is an example of the **softmax function** $\mathbb{R}^n \to \mathbb{R}^n$

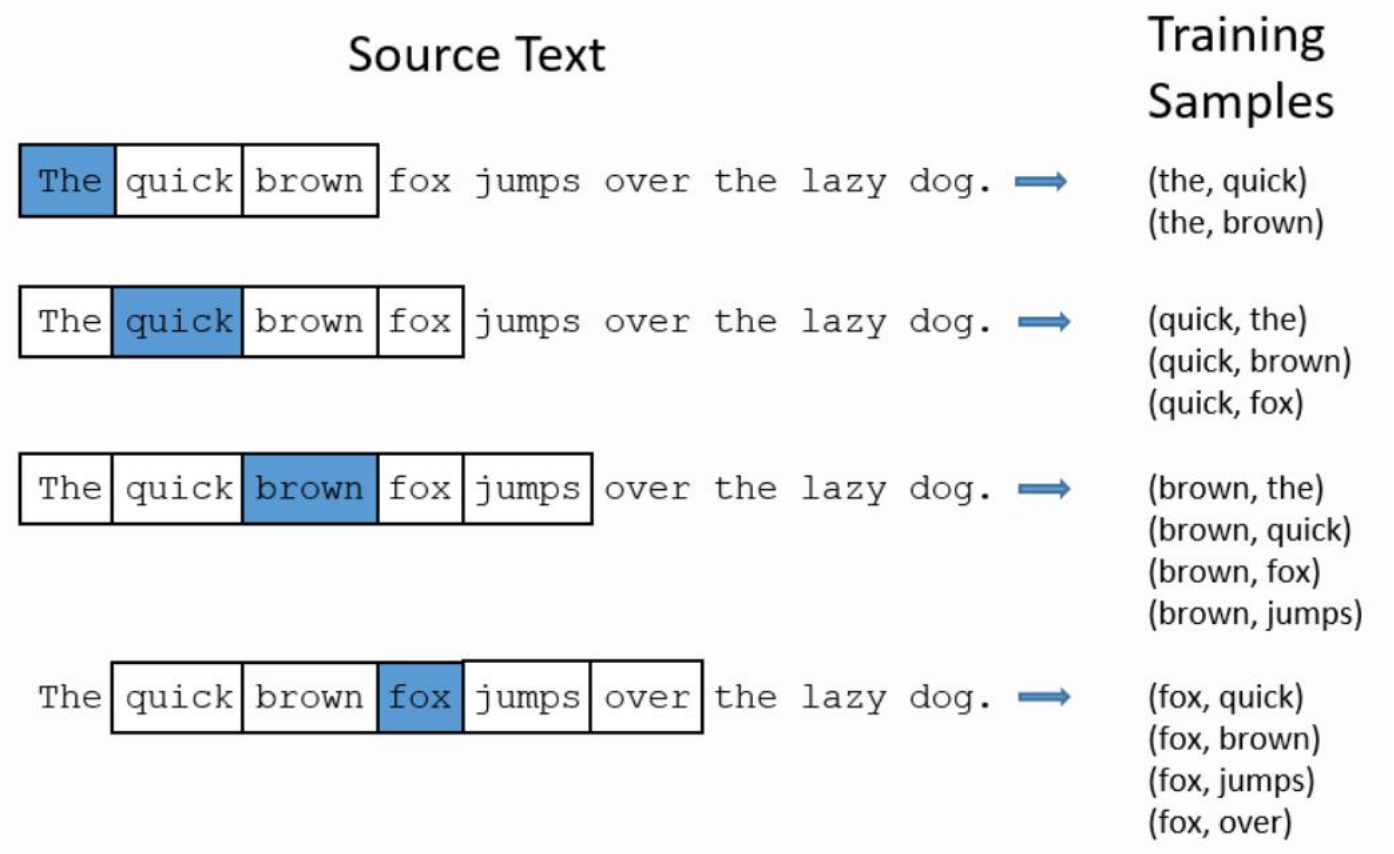
$$\text{softmax}(x_i) = \frac{\exp(x_i)}{\sum_{j=1}^{n} \exp(x_j)} = p_i$$

- The softmax function maps arbitrary values x_i to a probability distribution p_i

The Details

- 训练样本：

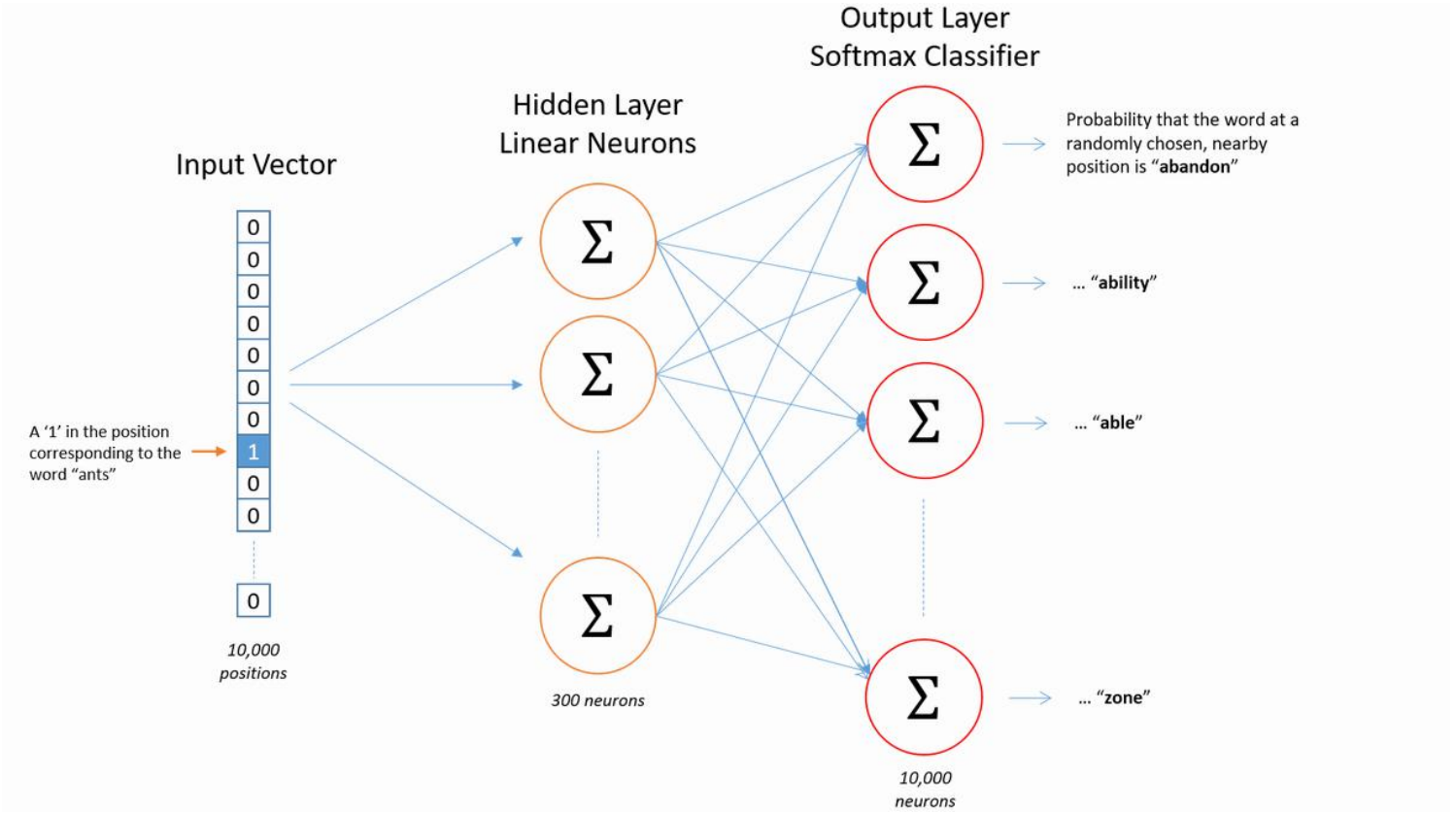
通过逐一选定句子中的单词作为输入词，将与之相邻的词提取出来，进行学习。



The Details

- 输入层：

通常会将文本中的词进行编码表示，如常见的有将文本库中的单词转换为词汇表，这样可以将每个单词通过one-hot编码表示。

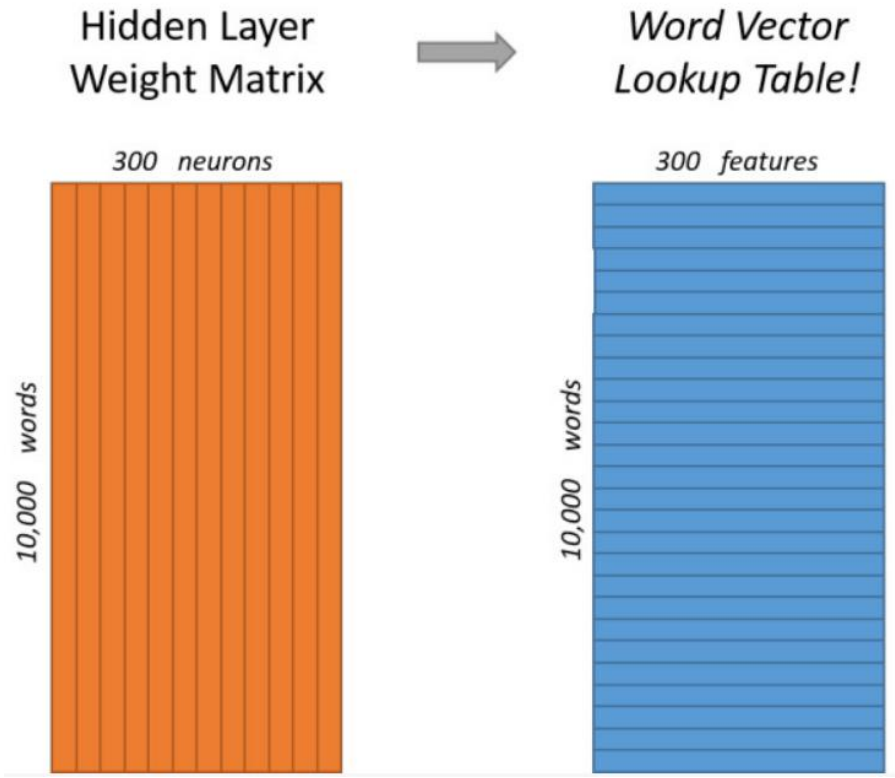


The Details

- 隐藏层：

如果现在想用300个特征来表示一个单词（即每个词可以被表示为300维的向量）。那么隐层的权重矩阵应该为10000行，300列（隐层有300个神经元）。

隐藏层中的权重矩阵大小为10000*300，一个10000维的one-hot输入词通过权重矩阵映射到了一个300维的向量。



The Details

- 隐层如何通过这个的权重矩阵映射？First，Input word被我们进行了one-hot编码。And then？

矩阵乘法

- One-hot与隐藏层权重矩阵相乘实际上是取权重矩阵特定的行，如下图所示：

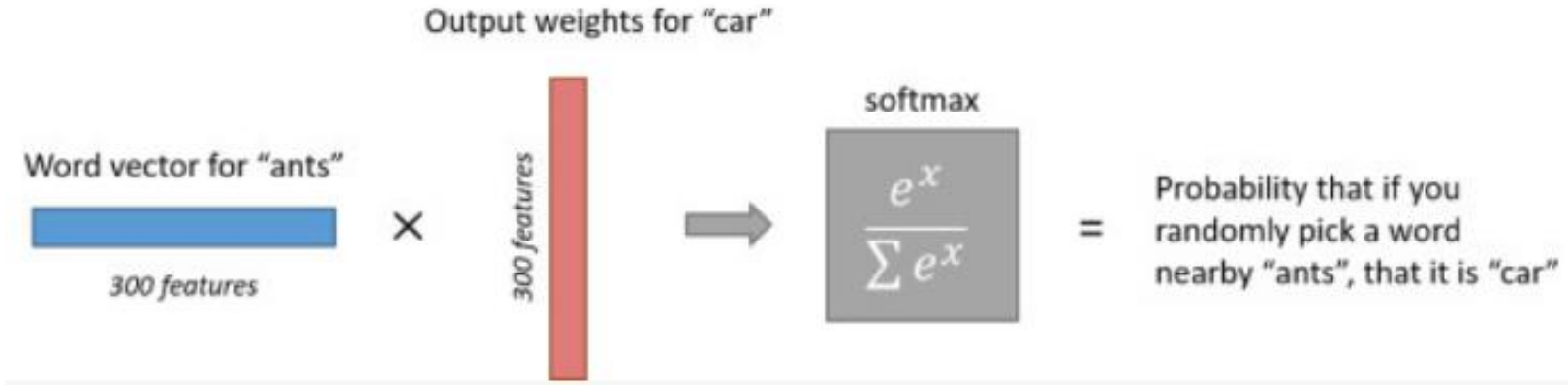
$$[0 \quad 0 \quad 0 \quad 1 \quad 0] \times \begin{bmatrix} 17 & 24 & 1 \\ 23 & 5 & 7 \\ 4 & 6 & 13 \\ 10 & 12 & 19 \\ 11 & 18 & 25 \end{bmatrix} = [10 \quad 12 \quad 19]$$

这样模型中的隐层权重矩阵便成了一个“查找表（lookup table）”，进行矩阵计算时，直接去查输入向量中取值为1的维度下对应的那些权重值。隐层的输出就是每个输入单词的“词向量”。

The Details

- 输出层：

经过神经网络隐层的计算，*ants*这个词会从一个1*10000的向量变成1*300的向量，再被输入到输出层。输出层是一个*softmax*回归分类器，它的每个结点将会输出一个0-1之间的值---概率。



To train the model: Compute all vector gradients!

- Recall：θ表示所有模型参数
- d-dimensional vectors and V-many words:

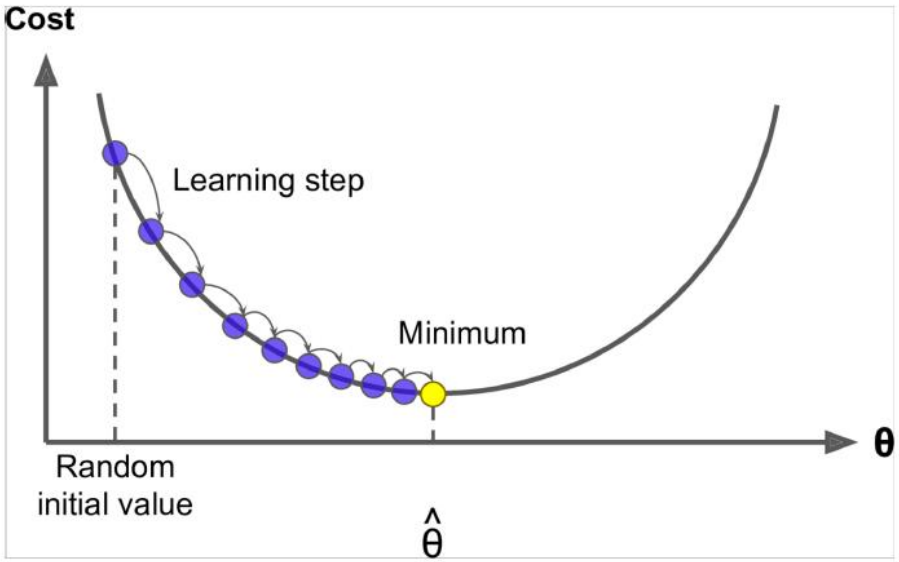
$$\theta = \begin{bmatrix} v_{aardvark} \\ v_a \\ \vdots \\ v_{zebra} \\ u_{aardvark} \\ u_a \\ \vdots \\ u_{zebra} \end{bmatrix} \in \mathbb{R}^{2dV}$$

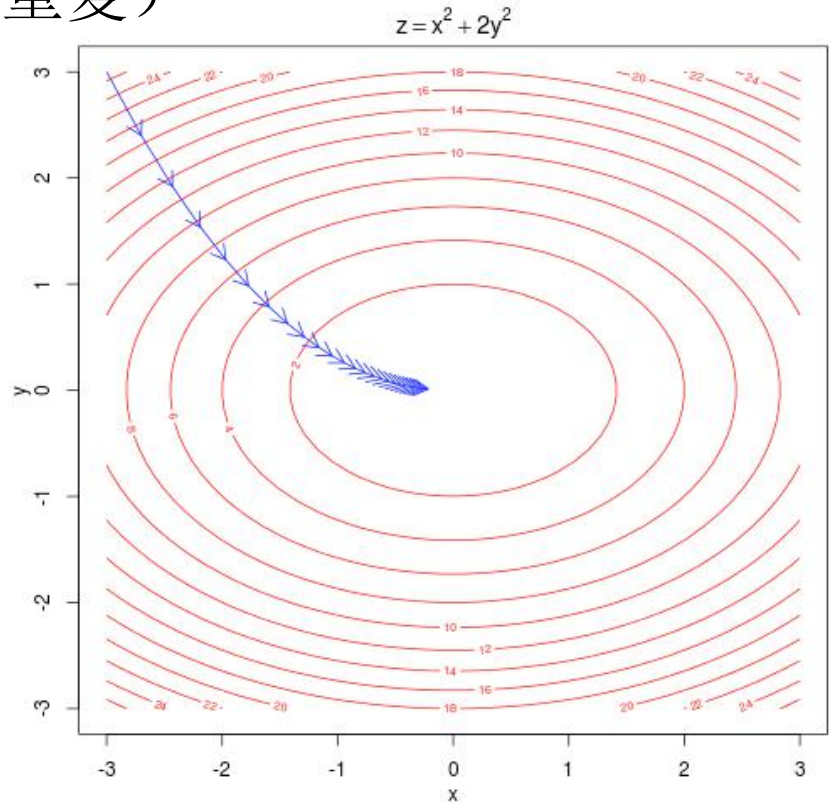
- Remember：每个词有两个向量
- 通过梯度下降优化模型参数

Optimization: Gradient Descent

- Gradient Descent（梯度下降）是最小化代价函数 $J(\theta)$ 的一种Algorithm（算法）

Idea：对 θ 的当前值，计算 $J(\theta)$的梯度，在负梯度方向上前进一小步。Repeat（重复）





Gradient Descent

- Update equation (in matrix notation):

$$\theta^{new} = \theta^{old} - \alpha \nabla_\theta J(\theta)$$

α = *step size* or *learning rate*

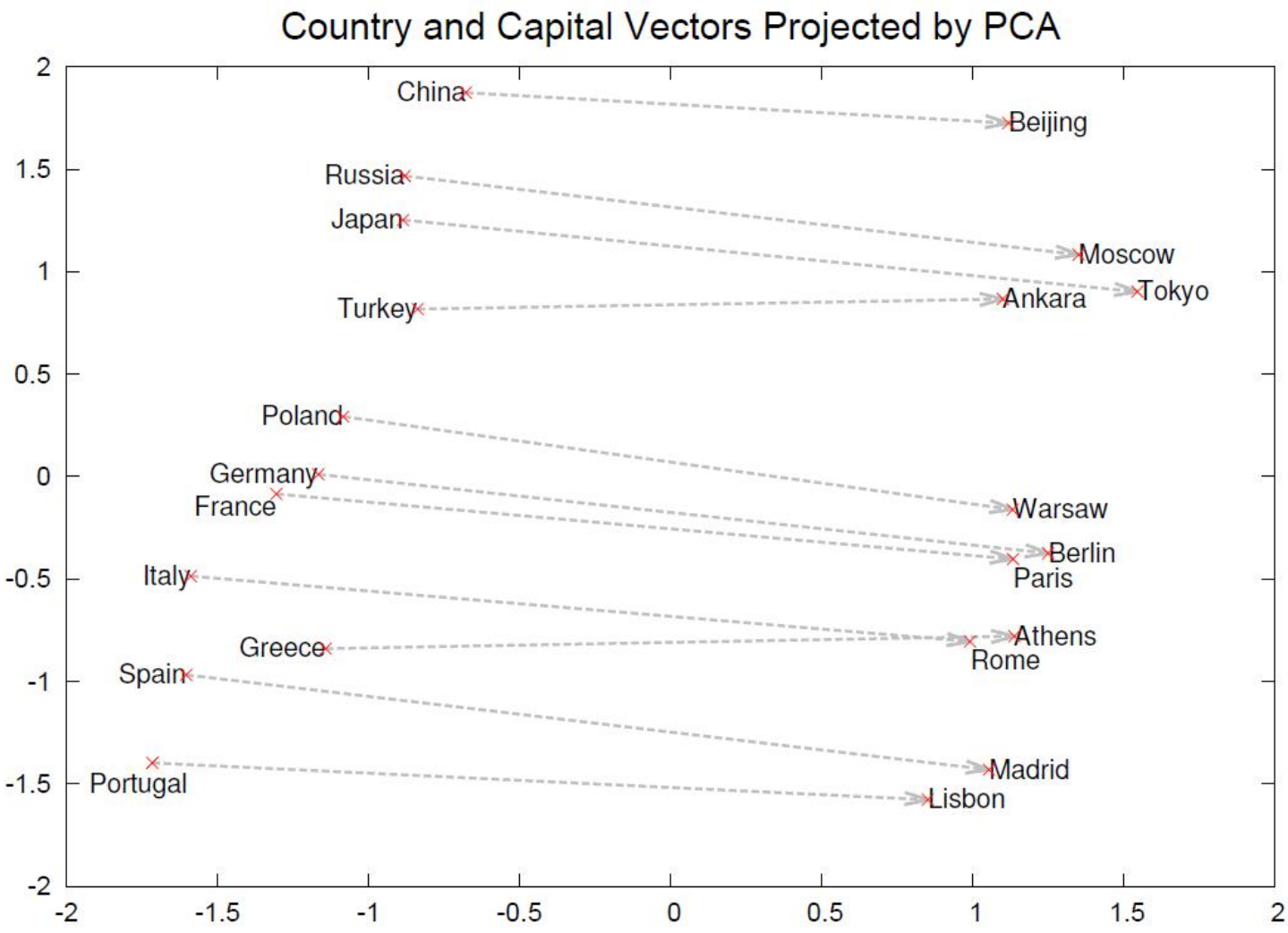
- Update equation (for single parameter):

$$\theta_j^{new} = \theta_j^{old} - \alpha \frac{\partial}{\partial \theta_j^{old}} J(\theta)$$

- Algorithm:

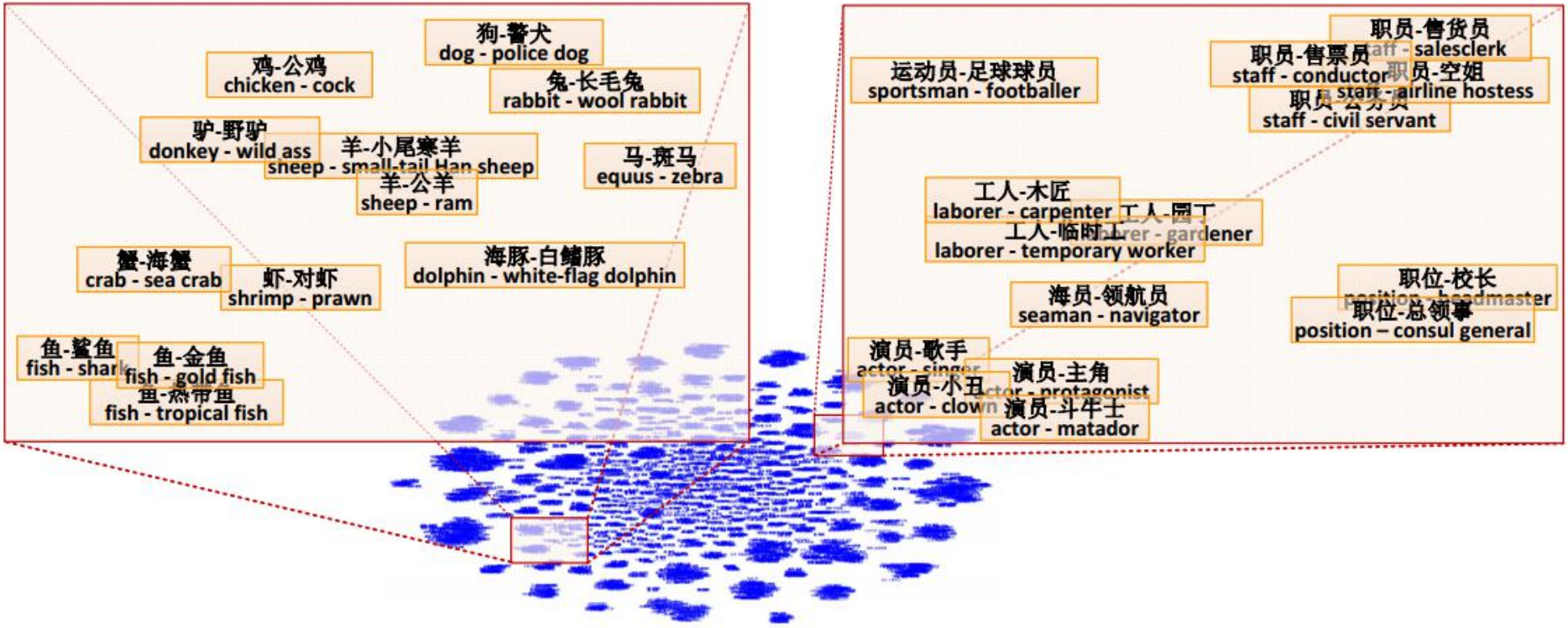
```
while True:
    theta_grad = evaluate_gradient(J,corpus,theta)
    theta = theta - alpha * theta_grad
```

Empirical Result



Mikolov T, et al. Distributed representations of words and phrases and their compositionality. NIPS, 2013.

Empirical Result

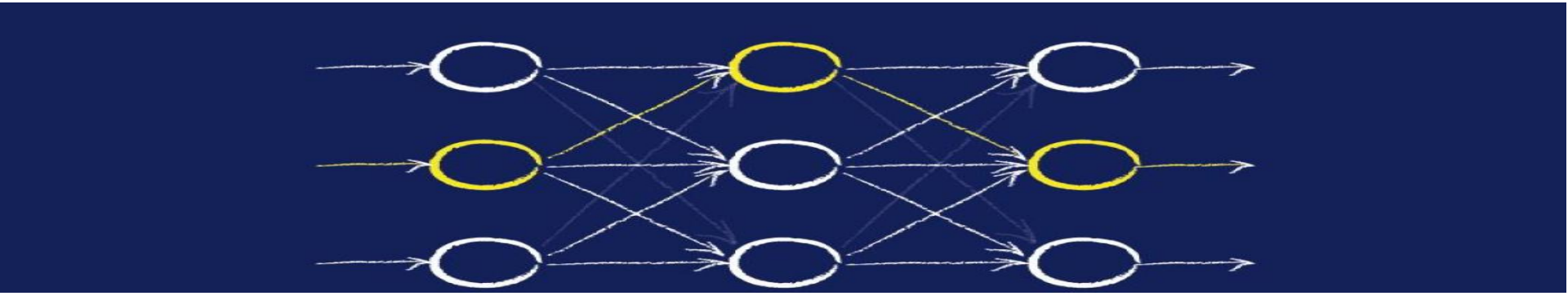


Fu, Ruiji, et al. Learning semantic hierarchies via word embeddings. ACL,2014.

Questions?

表示学习和生成式模型

（representation learning and generative model）



2、生成模型（ generative model ）

频率学派与贝叶斯学派：统计学领域的两大学派

统计学作为一门重要的学科，涉及到对数据进行分析、推断和预测。而在统计学领域，存在着两大主流学派——频率学派和贝叶斯学派。这两个学派采用不同的理论基础和方法论，**对未知参数的推断和估计有截然不同的观点**。

2、生成模型（ generative model ）

一、频率学派

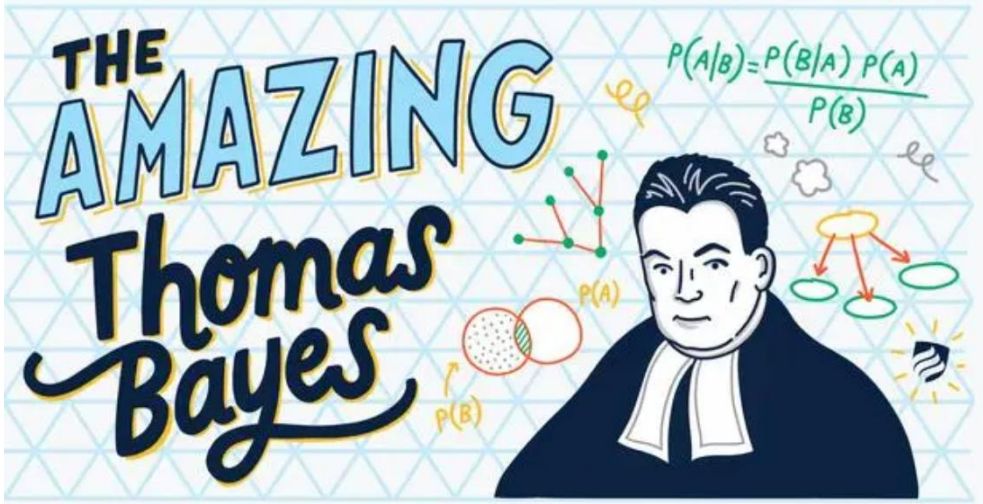
频率学派（传统学派）认为样本信息来自总体，通过对样本信息的研究可以合理地推断和估计总体信息，并且随着样本的增加，推断结果会更加准确。代表性人物是费希尔（R. A. Fisher, 1890-1962）。

频率学派的核心思想是**基于大样本**理论，**将概率看作频率的极限**，以样本观测值的频率为基础进行推断。频率学派注重数据的重复抽样和统计量的性质，比如点估计、置信区间和假设检验等。它强调的是**通过样本信息来推断总体参数**，并将此过程视为**客观的、可重复的**。

2、生成模型（ generative model ）

二、贝叶斯学派

贝叶斯学派源于英国学者贝叶斯（T. Bayes, 1702-1761）在1763年发表的著名论文《论有关机遇问题的求解》。贝叶斯学派认为**任何一个未知量都可以看作是随机的**，应该用一个**概率分布**去描述未知参数，而不是频率派认为的**固定值**。



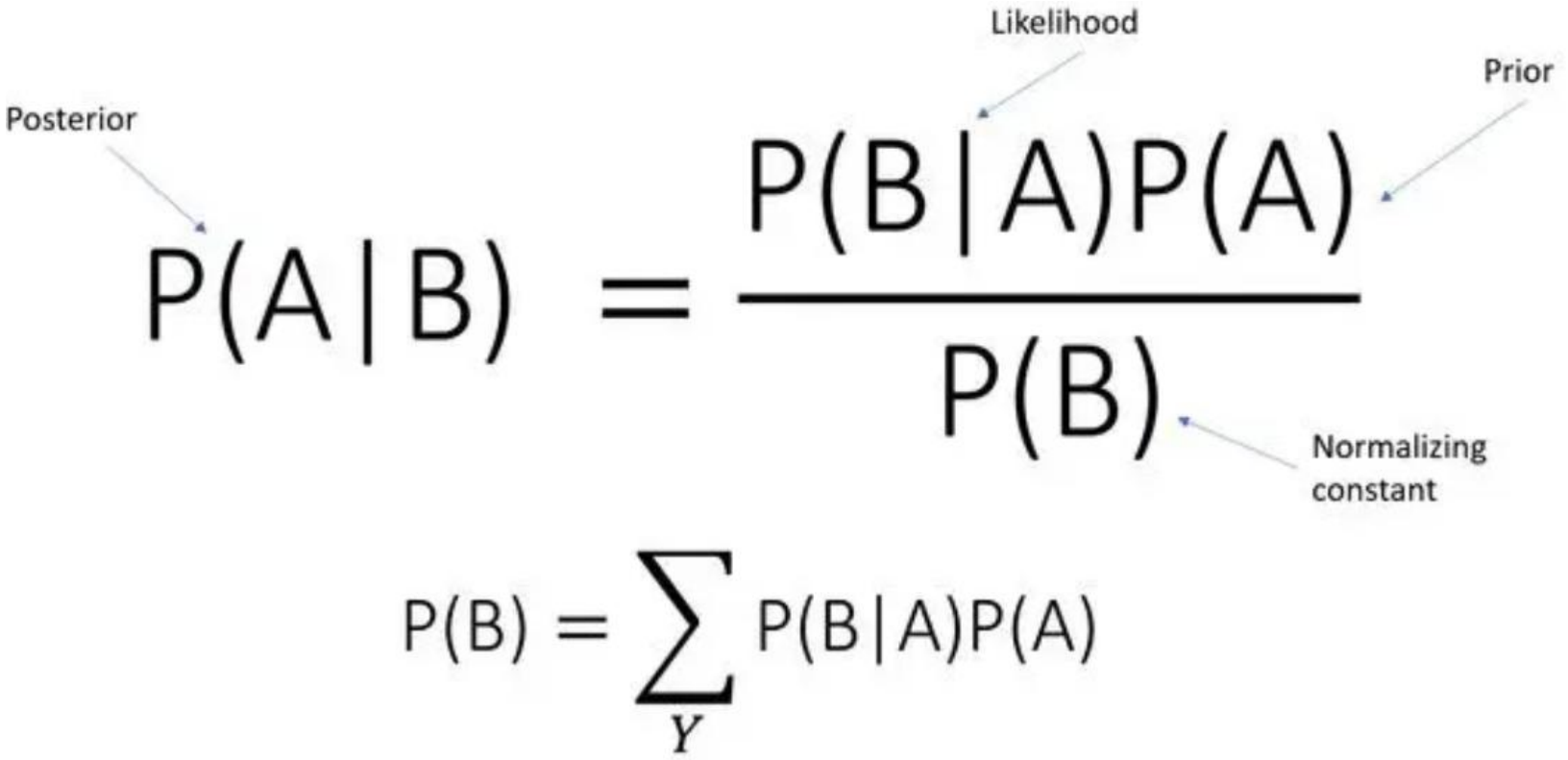
2、生成模型（ generative model ）

二、贝叶斯学派

贝叶斯学派的核心思想是先验信息与后验信息相结合，通过贝叶斯公式将先验信息与样本数据进行结合，得到后验分布，并以此作为对未知参数的推断。贝叶斯学派强调主观先验信息的引入，因此不同人可能会有不同的先验分布，从而导致不同的推断结果。贝叶斯学派注重个体的主观判断和背景信息，更加灵活和主观。

2、生成模型（ generative model ）

二、贝叶斯学派



2、生成模型（ generative model ）

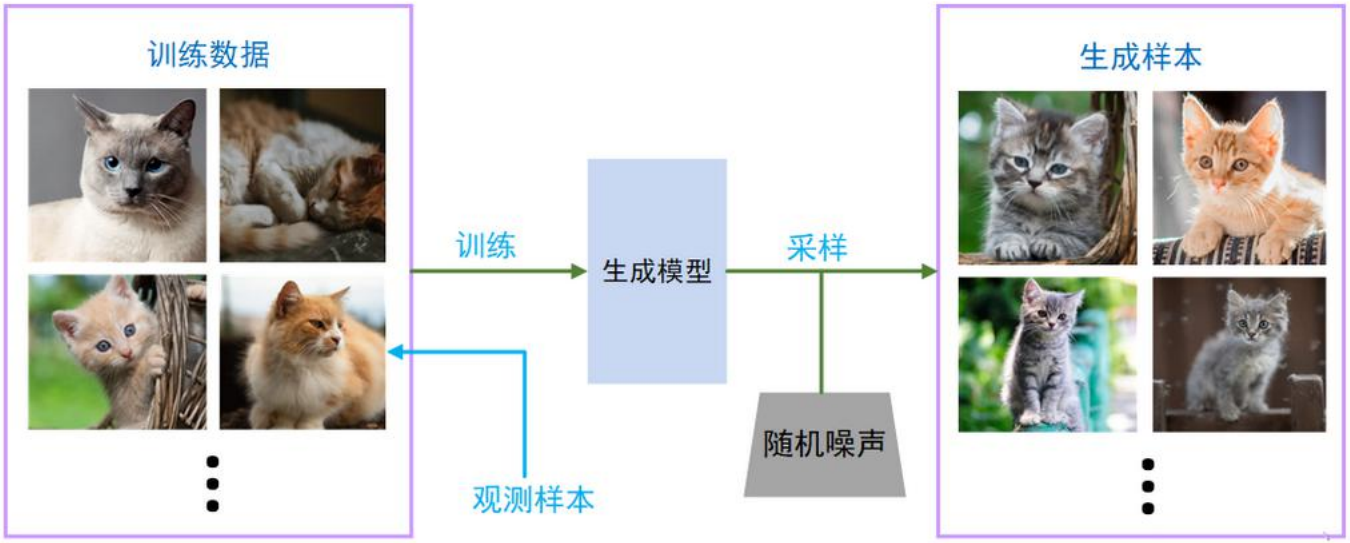
生成模型（Generative Model）和判别模（Discriminative Moel)

判别模型：由数据直接学习决策函数Y=f(X)或者条件概率分布P(Y|X)作为预测的模型，即判别模型。基本思想是有限样本条件下建立判别函数，不考虑样本的产生模型，直接研究预测模型。典型的判别模型包括k近邻，感知级，决策树，支持向量机等。

生成模型：由数据学习联合概率密度分布P(X,Y)，然后求出条件概率分布P(Y|X)作为预测的模型，即生成模型：P(Y|X)= P(X,Y)/ P(X)。基本思想是首先建立样本的联合概率概率密度模型P(X,Y)，然后再得到后验概率P(Y|X)，再利用它进行分类。常见的有NB HMM模型。

2、生成模型（ generative model ）

生成式人工智能 (Generative Artificial Intelligence, GAI) 旨在通过学习训练数据的分布模型来生成新的、原创的数据。人工智能生成内容（Artificial Intelligence Generated Content , AIGC)是生成式人工智能的一个具体应用和实现方式，是指利用人工智能技术生成各种形式的内容，如文字、图像、音频和视频等。



2、生成模型（generative model）

2.1. 人造分布与真实分布

一张人脸图片，其实本质上是一个由诸多像素值组成的矩阵，可视为一个高维的变量。在日常生活中，我们知道有所谓的“大众脸”，其实这侧面也就反应了人脸的生成其实是服从某一个概率分布的，只是我们不知道这个分布到底长什么样子？每当新生儿诞生甚至更早一点在胚胎形成的时候，其实就好像在这个庞大的人脸分布中进行了一次采样(条件采样)。所以其实你随便拿一张照片来，其实它都服从某一个分布，人脸有人脸的分布，狗脸有狗脸的分布。在此我们姑且将一个“造物主”创建的分布(比如某种猫脸的分布)称之为 $p(x)$ 。

2、生成模型（ generative model ）

2.1. 人造分布与真实分布

- 借助注
，这便
引出
网络的
强大
们通过
不断
，然后
再从
胃生成
的事
的猫
脸）。

由DDPM生成的猫脸

去噪扩散概率模型（Denoising Diffusion Probabilistic Model, DDPM）在2020年被提出，向世界展示了扩散模型的强大能力，带动了扩散模型的火热。

2、生成模型（ generative model ）

有了这个想法，那紧接着问题就来了:

1、该如何设计这个网络模型，来尽可能模拟这个世界上客观存在的分布?（关乎网络的设计）

2、如何度量两个分布之间的差距？如何让人造分布与真实分布尽可能接近？(关乎Loss的设计)

2、生成模型（ generative model ）

2.2.KL散度

第一个问题太泛了，先从第二个问题入手，因为有现成的度量工具(KL散度)。

(1)计算公式：$KL(q||p) = \int q(x) ln \frac{q(x)}{p(x)} dx$

(2)一般不满足对称性：$KL(q||p) \neq KL(p||q)$

(3)取值范围：$[0, \infty)$

2、生成模型（ generative model ）

我们是不可能有现成的真实分布给我们去比对计算Loss的，我们只能制造一些优质的数据集，数据集相当于对真实分布进行采样，在已知这些数据后我们可以初步得到一个真实数据的先验分布为 P_{data}，我们模型学到的分布是 $P_\theta(x)$，所以利用KL散度求得使其最小的网络参数 θ^*

$$\theta^* = arg\min_\theta KL(P_{data}(x)||P_\theta(x))$$

我们是用**有限的已知的数据**进行训练进而调整模型的参数 θ，这其实就是**最大似然估计的思想**，最大似然估计的思想就是: 在观测结果已知的情况下，求使得该观测最可能发生的参数 θ 。

$$\theta^* = arg\max_\theta \prod_{i=1}^{m} P_\theta(x^i) = arg\max_\theta log\prod_{i=1}^{m} P_\theta(x^i) = arg\max_\theta \sum_{i=1}^{m} logP_\theta(x^i)$$

2、生成模型（ generative model ）

$$
\begin{aligned}
\theta^* &\approx arg\max_{\theta} E_{x\sim P_{data}}[logP_\theta(x)] \\
&= arg\max_{\theta} \int P_{data}(x) logP_\theta(x) dx \\
&= arg\max_{\theta} \int P_{data}(x) logP_\theta(x) dx - \int P_{data}(x) logP_{data}(x) dx \\
&= arg\max_{\theta} \int P_{data}(x) log\frac{P_\theta(x)}{P_{data}(x)} dx \\
&= arg\max_{\theta} - \int P_{data}(x) log\frac{P_{data}(x)}{P_\theta(x)} dx \\
&= arg\min_{\theta} \int P_{data}(x) log\frac{P_{data}(x)}{P_\theta(x)} dx \\
&= arg\min_{\theta} KL(P_{data}(x)||P_\theta(x))
\end{aligned}
$$

所以最小KL散度和最大似然，二者殊路同归。所以就是说如果想要让生成的分布与真实的分布越接近，其实也可以近似的等价于求解数据集的极大似然，也就是求：

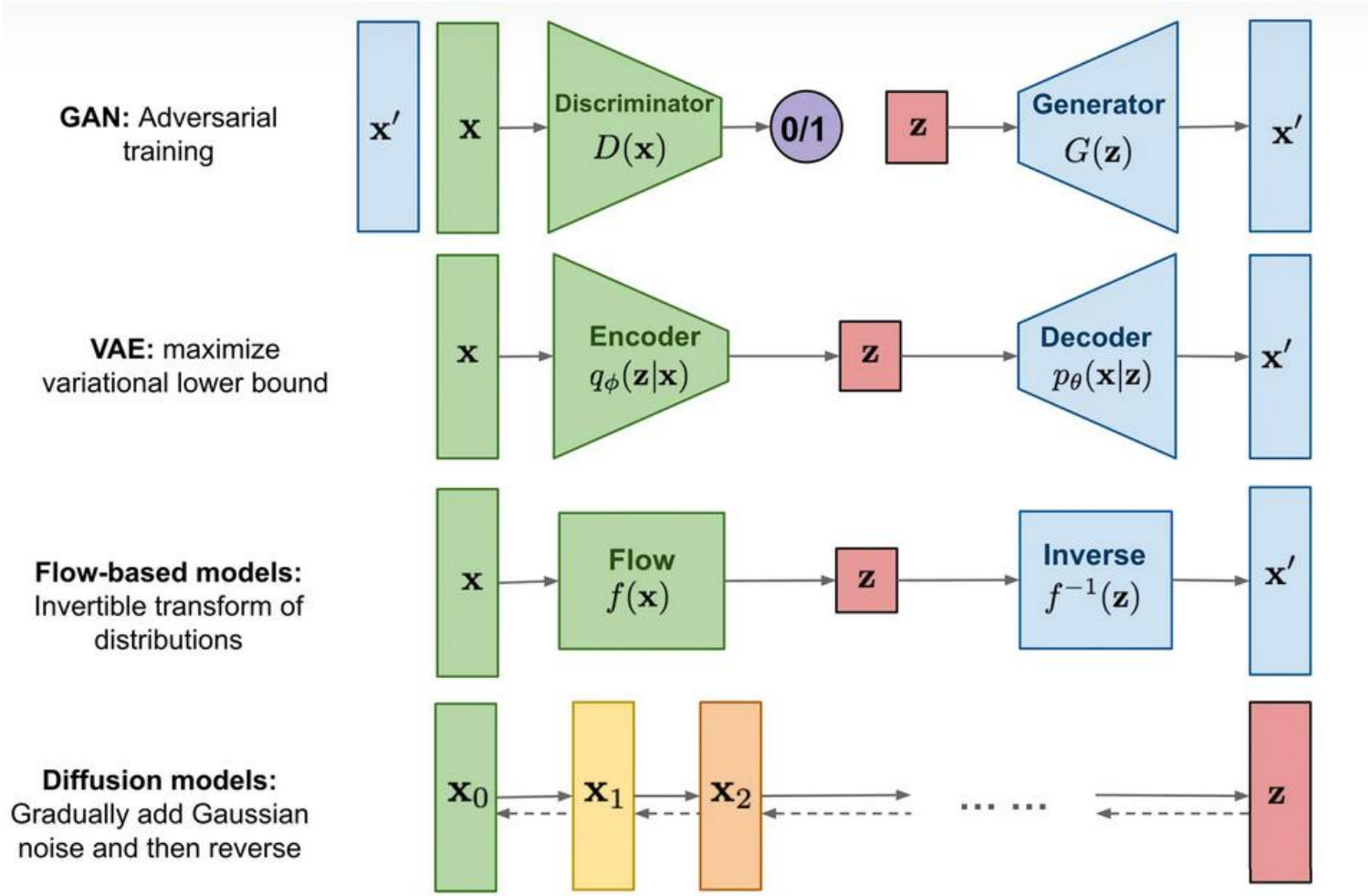
$$arg\max_{\theta} \sum_{i=1}^{m} logP_\theta(x^i)$$

2、生成模型（ generative model ）

第一个问题：怎么去模拟呢？怎么去设计模型呢？直接构造 $P_\theta(x)$十分麻烦，那只好间接去构造，共通的思路就是要把这个抽象的式子转化,与我们训练的损失函数对应起来,在减小loss的时候同时减小真实分布与模型分布之间的差异。 所以要有一个桥梁,比如这便很自然地引出了**隐变量模型**

2、生成模型（ generative model ）

2.3. 隐变量模型



2、生成模型（ generative model ）

从图中不难发现大致规律：**给定一个输入 x 对应一个分布 z，再利用这个 z 生成一个 x'**
用概率表示前面一部分就是 $p(z|x)$，后一部分就是 $p(x|z)$ 。这个 z 我们称之为隐变量。

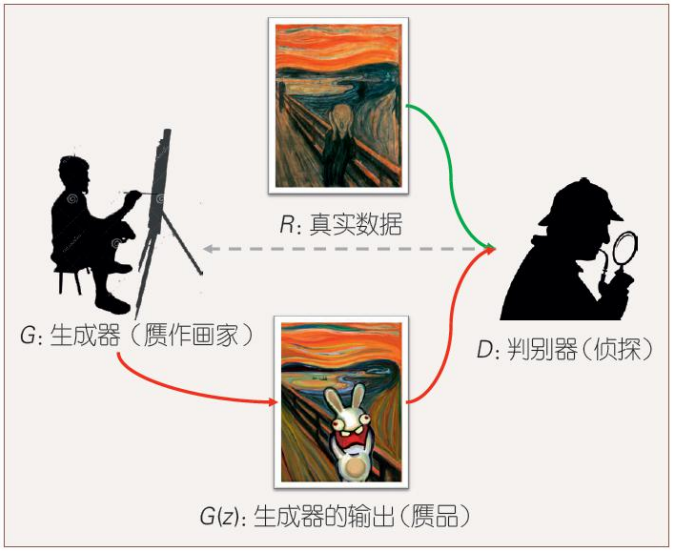
离散: $p(x) = \sum_z p(z)p(x|z)$ (全概率公式，GMM)

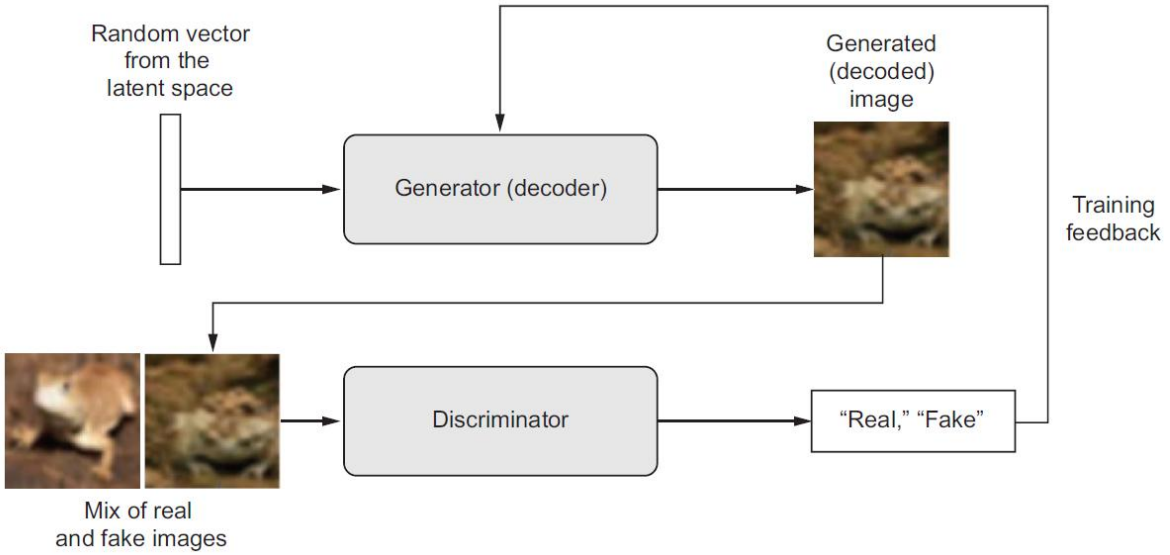
连续： $p(x) = \int p(z)p(x|z)dz$ (边缘分布)

其中 $p(z)$ 是可以为任意的先验概率分布(常见的有高斯分布)，这样一来就可以进行等价转化了

2、生成模型——GAN

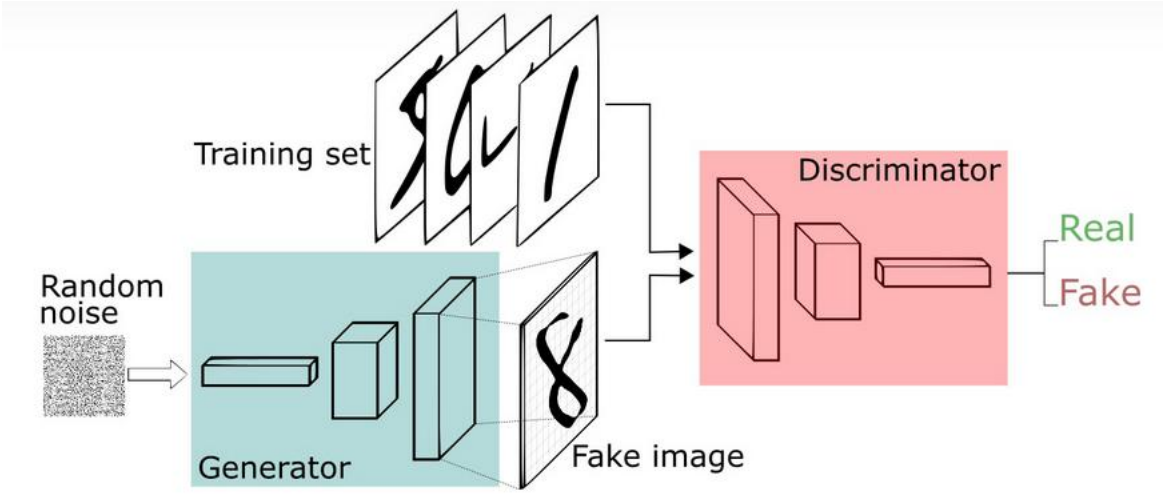
Generative Adversarial Nets（生成对抗网络）





2、生成模型——GAN

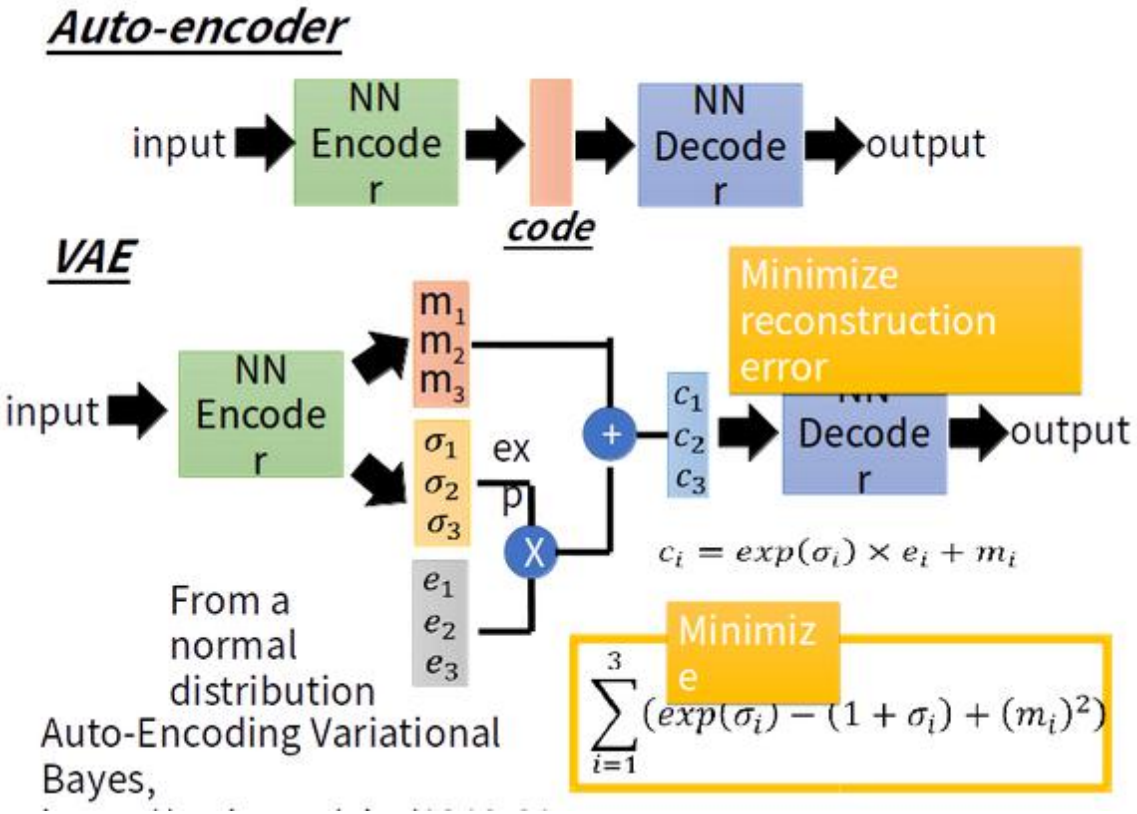
Generative Adversarial Nets（生成对抗网络）



$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))].$$

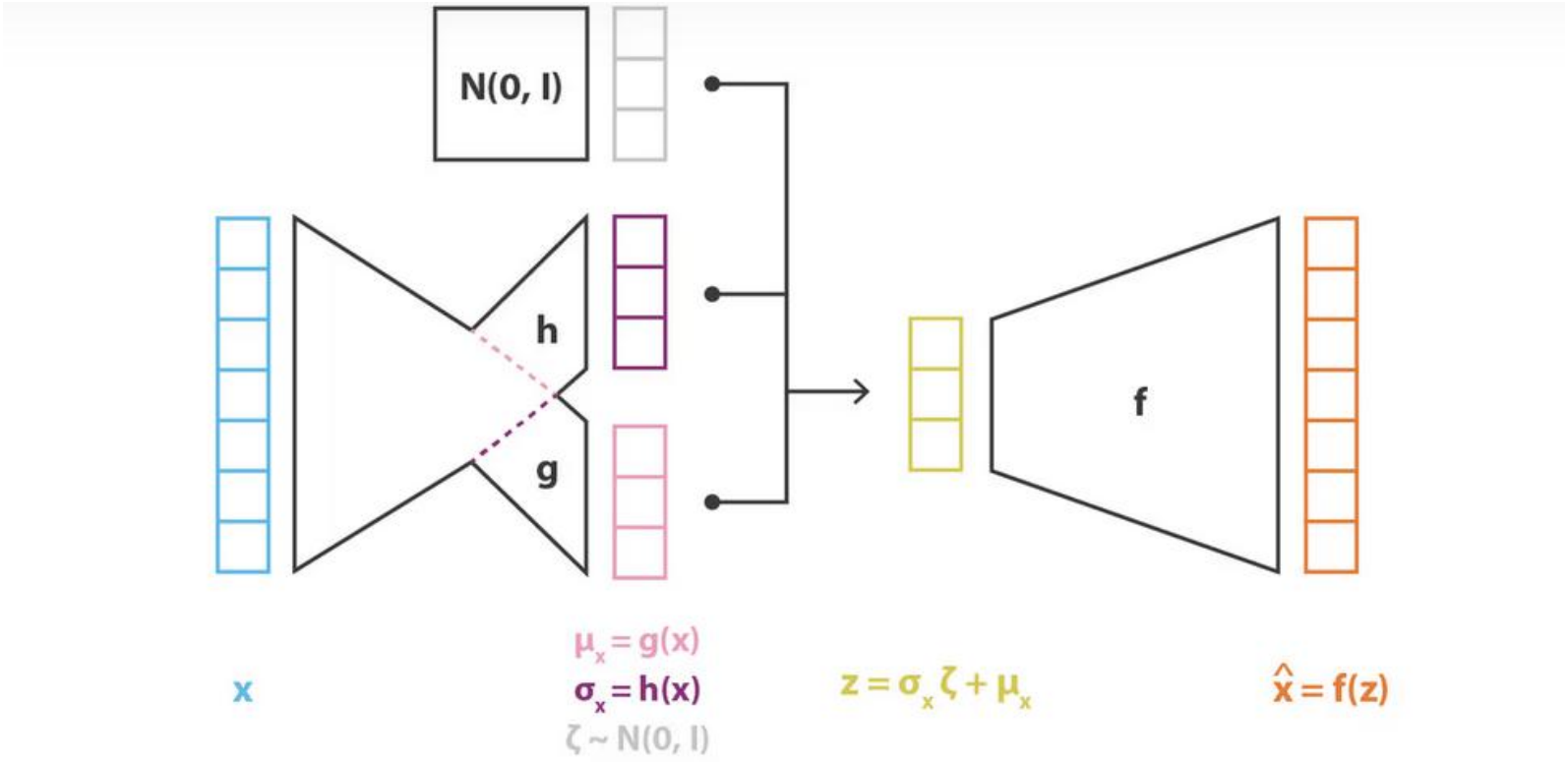
2、生成模型——VAE

变分自动编码器VAE(Variational Auto－encoder)



2、生成模型——VAE

变分自动编码器VAE(Variational Auto－encoder)



$$loss = C \| x - \hat{x} \|^2 + KL[\, N(\mu_x, \sigma_x), N(0, I)\,] = C \| x - f(z) \|^2 + KL[\, N(g(x), h(x)), N(0, I)\,]$$

2、生成模型——扩散模型（Diffusion Model）

去噪扩散概率模型（Denoising Diffusion Probabilistic Model, DDPM）

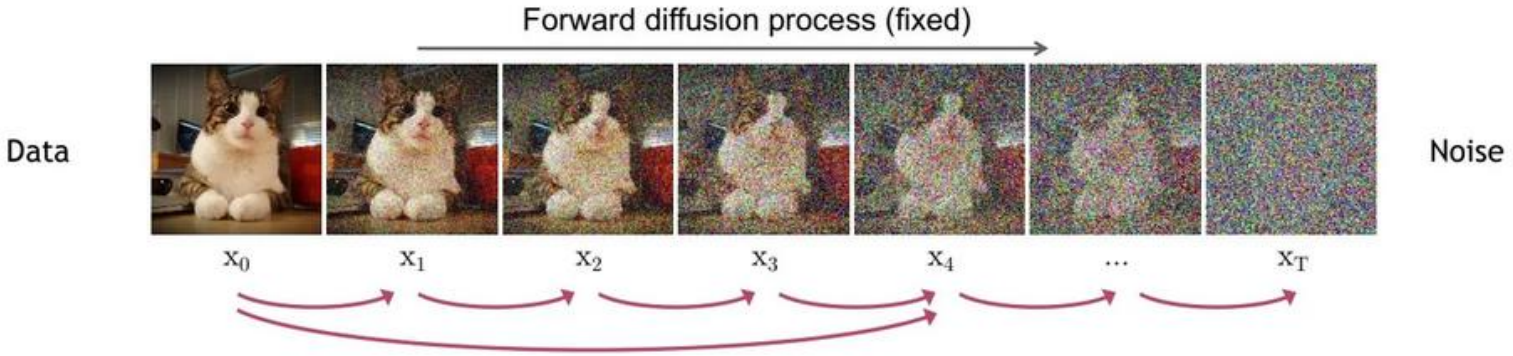
去噪扩散概率模型（Denoising Diffusion Probabilistic Model, DDPM）简单来说，扩散模型包含两个过程：前向扩散过程和反向生成过程，前向扩散过程是对一张图像逐渐添加高斯噪音直至变成随机噪音，而反向生成过程是去噪音过程，我们将从一个随机噪音开始逐渐去噪音直至生成一张图像，这也是我们要求解或者训练的部分。

2、生成模型——扩散模型（Diffusion Model）

扩散过程：

扩散过程是指的对数据逐渐增加高斯噪音直至数据变成随机噪音的过程。扩散过程的每一步都生成一个带噪音的数据x_t，整个扩散过程也就是一个马尔卡夫链：

$$q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1})$$



Define $\bar{\alpha}_t = \prod_{s=1}^{t}(1-\beta_s)$ ➡ $q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I}))$ (Diffusion Kernel)

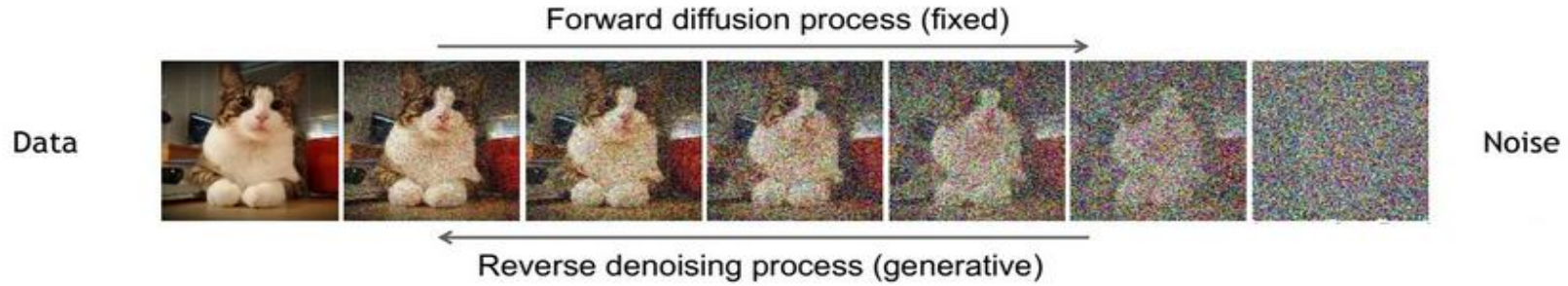
For sampling: $\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\,\mathbf{x}_0 + \sqrt{(1-\bar{\alpha}_t)}\,\epsilon$ where $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

β_t values schedule (i.e., the noise schedule) is designed such that $\bar{\alpha}_T \to 0$ and $q(\mathbf{x}_T|\mathbf{x}_0) \approx \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I}))$

2、生成模型——扩散模型（Diffusion Model）

反向过程：

扩散过程是将数据噪音化，那么反向过程就是**一个去噪的过程**，如果我们知道反向过程的每一步的真实分布$q(\mathbf{x}_{t-1}|\mathbf{x}_t)$，那么从一个随机噪音$\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$开始，逐渐去噪就能生成一个真实的样本，所以反向过程也就是**生成数据的过程**。



估计分布$q(\mathbf{x}_{t-1}|\mathbf{x}_t)$需要用到整个训练样本，我们可以用神经网络来估计这些分布。这里，我们将反向过程也定义为一个马尔卡夫链，只不过它是由一系列**用神经网络参数化的高斯分布**来组成：

$$p_\theta(\mathbf{x}_{0:T}) = p(\mathbf{x}_T)\prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) \quad p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t))$$

这里$p(\mathbf{x}_T) = \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I})$，而$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$为参数化的高斯分布，它们的均值和方差由训练的网络$\boldsymbol{\mu}_\theta(\mathbf{x}_t, t)$和$\boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t)$给出。实际上，**扩散模型就是要得到这些训练好的网络，因为它们构成了最终的生成模型。**

2、生成模型（ generative model ）


Thanks!
Obrigado!
Gracias!
谢谢！
Dziękuję!
Danke!
Grazie!
Merci!
شكرا لك !
감사합니다!
Kiitos!