2025.06.05

情報科学特別講義 I
生成モデルの基礎と応用

高橋 大志 (NTT)

自己紹介

- 名前　高橋 大志 (たかはし ひろし)
- 経歴
  - 2009.4 – 2015.3　東京工業大学 (学士・修士)
  - 2020.4 – 2023.3　京都大学 (博士)
  - 2015.4 – 現在　NTT
- 研究分野　生成モデル・異常検知

Variational Autoencoder (VAE) や
拡散モデルの研究をしています

Copyright 2025 NTT CORPORATION

はじめに (10分)

Copyright 2025 NTT CORPORATION

生成AIについて

- 言語や画像、音声などを生成する人工知能 (AI) の総称

チャットボット

ChatGPT

Gemini

Claude

tsuzumi

画像生成

DALL-E[1]

Stable Diffusion[2]

1. https://ja.wikipedia.org/wiki/DALL-E
2. https://ja.wikipedia.org/wiki/Stable_Diffusion

Copyright 2025 NTT CORPORATION

ChatGPT (OpenAI)

- OpenAIが2022年11月に公開したAIチャットボット
  - 情報提供や質問回答、文章作成、対話、言語学習支援などが可能
  - GPT = Generative Pre-trained Transformer (後述)
  - 非常に高い性能から話題となり、ChatGPT公開後にOpenAIの評価額は290億ドルまで上がった[1] (2025年3月時点で推定3000億ドル超え[2])

1. https://www.businessinsider.com/chatgpt-creator-openai-talks-for-tender-offer-at-29-billion-2023-1
2. https://www.itmedia.co.jp/news/articles/2504/01/news090.html

Copyright 2025 NTT CORPORATION

DALL-E (OpenAI)

- OpenAIが2021年1月に公開した画像生成AI
  - ユーザが入力したテキストから画像を生成できる
  - 詳細は不明だが拡散モデル (後述) を用いているとのこと

例: 桜が咲き誇る静かな日本庭園にある、アニメ風の静かな鯉の池

https://openai.com/index/dall-e-3/

Copyright 2025 NTT CORPORATION

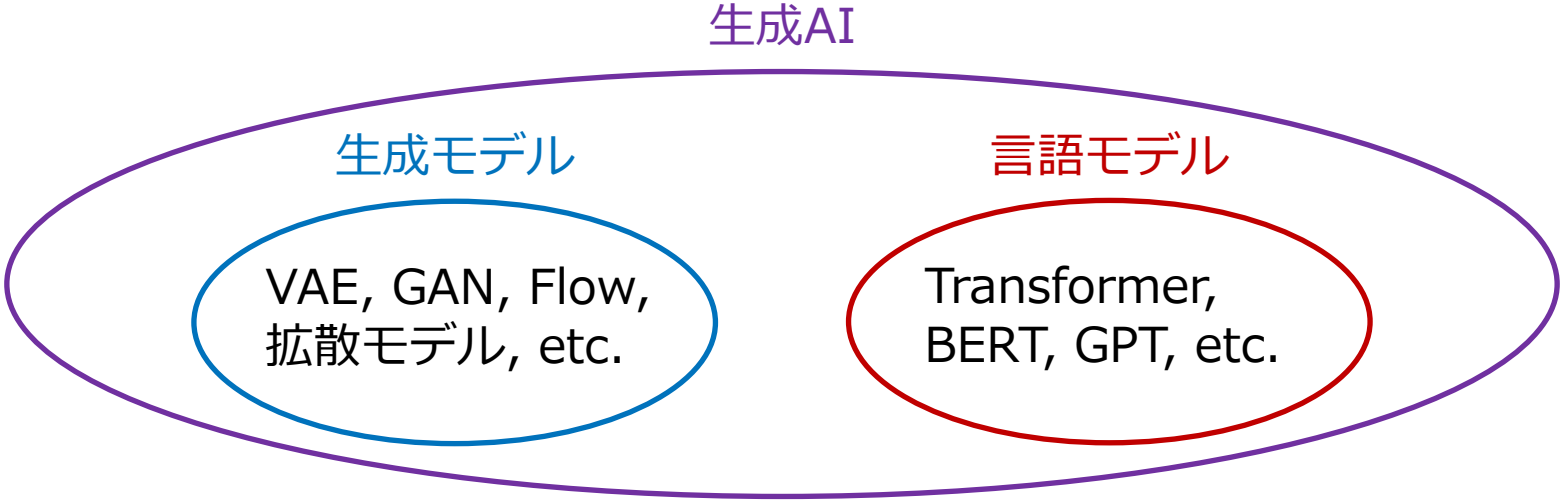
本講義の狙い

- 生成AIは研究開発のスピードが非常に速く、キャッチアップが難しい
  - 2025年現在、数週間ごとに大きな進化が起こっている
- その一方で、生成AIの基礎となっている部分は昔からほとんど変わっていない
  - 本講義で最初に説明する最尤推定は1912年頃から研究されている
- 本講義は、生成AIの基礎と応用を学び、将来的な生成AIの研究開発の一助となることを目指す
  - 最先端の生成AIの一歩手前までの内容を取り扱う

Copyright 2025 NTT CORPORATION

生成モデルと生成AI

- 本講義のタイトルに用いている生成モデルは機械学習用語
  - 現在は事実上、画像を生成するモデルを指すことが多い
  - 一方で、生成AIはテキスト・画像・音声などを生成するAI全般を指す
  - 本講義では、生成モデルと言語モデル (言語の生成モデル) を扱う



Copyright 2025 NTT CORPORATION

本講義の内容

- 本講義で取り扱うこと
  - 生成モデルの基礎 (最尤推定 / ガウス混合モデル)
  - 深層生成モデル (VAE / 拡散モデル) ← 主に画像
  - 言語モデル (Transformer / BERT / GPT) ← 主に言語
  - Hugging Face入門
  - Instruction Tuning (Alignment)
- 本講義で取り扱わないこと
  - 最適化の詳細について (主にEMアルゴリズムや確率的勾配法など)
  - 大規模言語モデル (主にTransformerやAlignment以外の要素技術)

Copyright 2025 NTT CORPORATION

講義を始める前に

- 数式が多く出てきますが、すべて理解しようとしなくてOKです
  - 重要な数式は目立つようにします

- 講義中に質問する時間を多めに設けます
  - それ以外でも挙手や発声で止めていただいてOKです

- PCやスマホなどで資料を見ながらお聞きください
  - 3時間と長い講義なので、各自のペースでお聞きください

Copyright 2025 NTT CORPORATION

A. 生成モデルの基礎 (35分)

Copyright 2025 NTT CORPORATION

生成モデルについて

- データから学習された確率分布のこと
  - 例えば、コインの裏表の確率は1/2で、サイコロの目の確率は1/6
  - このようなデータが発生する「確率」を、データから学習したモデル
- 確率分布が分かっていれば、新たなデータを「生成」できる
  - 本講義では、データが従う確率分布を、生成モデルで学習する方法を学ぶ
- 本講義で扱う確率分布は下記の2つ:
  - データ x の確率分布 $p(x)$: コインやサイコロ等
  - 出力 y に対する条件付き確率分布 $p(y|x)$: Stable DiffusionやChatGPT等
  - NOTE: データ x や出力 y はベクトルまたはスカラ (区別せずに表記)

Copyright 2025 NTT CORPORATION

確率分布について

- データ x が発生する確率を表す関数
  - 例えば、コインの表裏を表す確率は $P(x) = \begin{cases} 0.5\ (x = \text{表}) \\ 0.5\ (x = \text{裏}) \end{cases}$
  - 離散値の場合は、$0 \leq P(x) \leq 1$ かつ $\sum_x P(x) = 1$ をみたす
- 連続値の場合は、確率密度関数 $p(x)$ を考える
  - データ x が a 以上 b 以下である確率は $P(a \leq x \leq b) = \int_a^b p(x)\,\mathrm{d}x$
  - $0 \leq p(x) < \infty$ かつ $\int_{-\infty}^{\infty} p(x)\,\mathrm{d}x = 1$ をみたす
  - NOTE: 確率分布 P と確率密度関数 p は区別せずに表記

Copyright 2025 NTT CORPORATION

期待値について

- 確率分布 $p(x)$ を用いて、ある関数 $f(x)$ の荷重平均を計算する操作を期待値計算と呼ぶ

$$\mathbb{E}_{p(x)}\left[f(x)\right] = \int_{-\infty}^{\infty} p(x)f(x)\mathrm{d}x$$

- $p(x)$ に従うデータ $\{x_1, \ldots, x_N\}$ を用いて、下記で近似できる

$$\mathbb{E}_{p(x)}\left[f(x)\right] \simeq \frac{1}{N}\sum_{n=1}^{N} f(x_n)$$

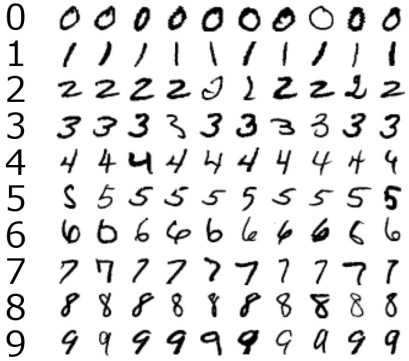
Copyright 2025 NTT CORPORATION

生成モデルの仮定

データ x は真の確率分布 $p^*(x)$ に従う

数字一つがデータ x

$p^*_{\mathrm{MNIST}}(x)$



MNIST

$p^*_{\mathrm{CIFAR10}}(x)$

CIFAR10

画像一枚がデータ x

ITmedia: https://www.itmedia.co.jp/

Copyright 2025 NTT CORPORATION

最尤推定 (1)

- 前提: 真の分布 $p^*(x)$ は未知の関数だが、$p^*(x)$ に従うデータセット $\mathcal{D} = \{x_1, \ldots, x_N\}$ が利用可能
- 目的: データセット $\mathcal{D}$ を用いて、生成モデル $p_\theta(x)$ を真の分布 $p^*(x)$ に近づけるよう学習したい (θはパラメータ、後述)
- 方法: カルバック・ライブラー情報量という、2つの確率分布の差異を計る尺度を導入し、最小化することでモデルを学習する

$$\mathrm{KL}(p^*(x)||p_\theta(x)) = \int_{-\infty}^{\infty} p^*(x) \log \frac{p^*(x)}{p_\theta(x)} \mathrm{d}x$$

確率分布同士の距離のようなもの (対称性はない)

Copyright 2025 NTT CORPORATION

最尤推定 (2)

- カルバック・ライブラー情報量は下記のように変形できる

$$\mathrm{KL}(p^*(x)||p_\theta(x)) = \underbrace{\int_{-\infty}^{\infty} p^*(x)\log p^*(x)\mathrm{d}x}_{\text{定数 (生成モデルに関係ない値)}} - \int_{-\infty}^{\infty} p^*(x)\log p_\theta(x)\mathrm{d}x$$

- したがって、上記の最小化は下記で書き換えられる

$$\min_\theta \mathrm{KL}(p^*(x)||p_\theta(x)) = \max_\theta \int_{-\infty}^{\infty} p^*(x)\log p_\theta(x)\mathrm{d}x \simeq \max_\theta \frac{1}{N}\sum_{n=1}^{N}\log p_\theta(x_n)$$

定数を除き、最小化を最大化に

データセットを用いて近似

Copyright 2025 NTT CORPORATION

最尤推定 (3)

データセット $\mathcal{D} = \{x_1, \ldots, x_N\}$ に対して、下記の対数尤度関数を最大化することで、生成モデル $p_\theta(x)$ を学習できる

$$\max_\theta \frac{1}{N} \sum_{n=1}^{N} \log p_\theta(x_n)$$

- この学習方法を最尤推定と呼ぶ
  - 生成モデル以外の機械学習分野でも幅広く用いられている学習方法
  - 本講義で説明するモデルは全て最尤推定で学習する

Copyright 2025 NTT CORPORATION

例: コインの裏表の確率は？ (1)

- 表が1、裏が0として、 $\mathcal{D} = \{1, 0, 1, 0, 1, 1, 0\}$ からコインの裏表の確率を最尤推定したい
- 二値データをモデル化する時は、ベルヌーイ分布が用いられる

$$P_\theta(x = k) = \theta^k (1 - \theta)^{1-k}$$

- ここで、パラメータ θ は表が出る確率を表す
  - パラメータとは、確率分布を特徴付ける変数のことを呼ぶ
  - この場合は1次元の変数だが、後述する深層生成モデルでは、ニューラルネットワークの各層の重みやバイアスなどがパラメータとなる

Copyright 2025 NTT CORPORATION

例: コインの裏表の確率は？ (2)

- このモデルのパラメータ θ を最尤推定で学習する
  - 目的関数は下記の通り

$$\frac{1}{N}\sum_{n=1}^{N}\log P_\theta(x_n = k) = \frac{1}{N}\sum_{n=1}^{N}\{x_n \log\theta + (1-x_n)\log(1-\theta)\} = \mathcal{L}(\theta)$$

  - この場合は微分を用いて閉形式で最適な $\hat{\theta}$ が求められる

$$\frac{\partial\mathcal{L}(\hat{\theta})}{\partial\hat{\theta}} = \frac{1}{N}\sum_{n=1}^{N}\left\{\frac{x_n}{\hat{\theta}} - \frac{1-x_n}{1-\hat{\theta}}\right\} = 0 \Leftrightarrow \hat{\theta} = \boxed{\frac{1}{N}\sum_{n=1}^{N} x_n}$$

出た値の平均値になっている

Copyright 2025 NTT CORPORATION

例: コインの裏表の確率は？ (3)

$$\frac{1}{N}\sum_{n=1}^{N}\left\{\frac{x_n}{\hat{\theta}}-\frac{1-x_n}{1-\hat{\theta}}\right\}=0$$

$$\frac{1}{\hat{\theta}}\sum_{n=1}^{N}x_n=\frac{1}{1-\hat{\theta}}\sum_{n=1}^{N}(1-x_n)$$

$$\frac{1-\hat{\theta}}{\hat{\theta}}=\frac{N-\sum_{n=1}^{N}x_n}{\sum_{n=1}^{N}x_n}$$

$$\frac{1}{\hat{\theta}}=\frac{N}{\sum_{n=1}^{N}x_n}$$

$$\boxed{\hat{\theta}=\frac{1}{N}\sum_{n=1}^{N}x_n}$$

Copyright 2025 NTT CORPORATION

例: コインの裏表の確率は？ (4)

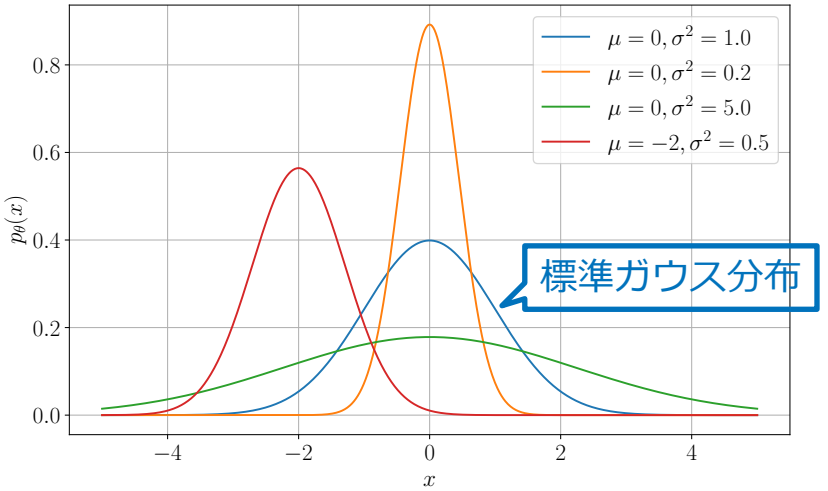
- $\mathcal{D} = \{1, 0, 1, 0, 1, 1, 0\}$ を用いて計算すると、$\hat{\theta} = 4/7$
  - コインの出た値の平均が最適なパラメータとなるため、直感と一致する
  - 偏りが一切ないコインであれば、データセットのサイズを大きくすれば、$\hat{\theta} = 1/2$ に収束する
  - 最尤推定で得られたパラメータを最尤推定量と呼ぶ
- このように、下記の手順を行うことで生成モデルは学習できる
  1. データセット $\mathcal{D} = \{x_1, \ldots, x_N\}$ に対して、適切なモデル $p_\theta(x)$ を選択する
  2. 対数尤度関数 $\frac{1}{N}\sum_{n=1}^{N} \log p_\theta(x)$ を計算する
  3. パラメータ θ に関する微分を用いて最適化する

Copyright 2025 NTT CORPORATION

ガウス分布 (1)

- データ x が連続値の場合は、ガウス分布やガウス混合モデルを用いることが多い
- 1次元のガウス分布は平均 μ と分散 σ^2 をパラメータにもち、下記で表される

$$\mathcal{N}(x;\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$



Copyright 2025 NTT CORPORATION

ガウス分布 (2)

- データセット $\mathcal{D} = \{x_1, \ldots, x_N\}$ が与えられた場合、ガウス分布の各パラメータの最尤推定量 $\hat{\mu}$ と $\hat{\sigma}^2$ は下記で計算できる
  - ベルヌーイ分布の場合と同様に、微分を用いて閉形式で求められる

$$\hat{\mu} = \underbrace{\frac{1}{N}\sum_{n=1}^{N} x_n}_{\text{データの平均}}, \quad \hat{\sigma}^2 = \underbrace{\frac{1}{N}\sum_{n=1}^{N}(x_n - \hat{\mu})^2}_{\text{データの分散}}$$

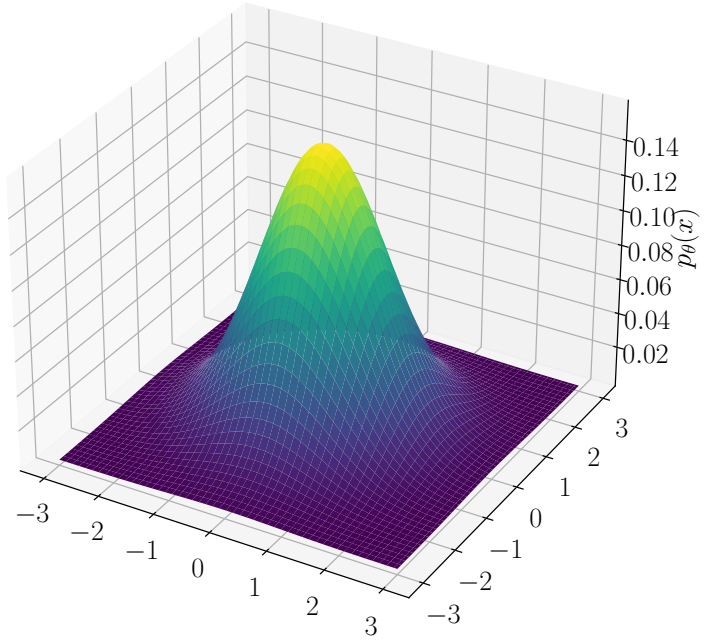
- 様々な好ましい性質 (計算のしやすさ等) があるため、連続値データをモデル化する時に良く用いられる
  - 後述する深層生成モデルは全てガウス分布を用いる

Copyright 2025 NTT CORPORATION

ガウス分布 (3)

- D_x 次元のガウス分布は、平均 μ と共分散行列 $\sum$ をパラメータにもち、下記で表される

$$\mathcal{N}(x;\mu,\Sigma) = \frac{\exp(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu))}{\sqrt{(2\pi)^{D_x}|\Sigma|}}$$



ChatGPTを使ってプロットしてみよう

> 2次元のガウス分布を3Dプロットするコードを Pythonで書いてください

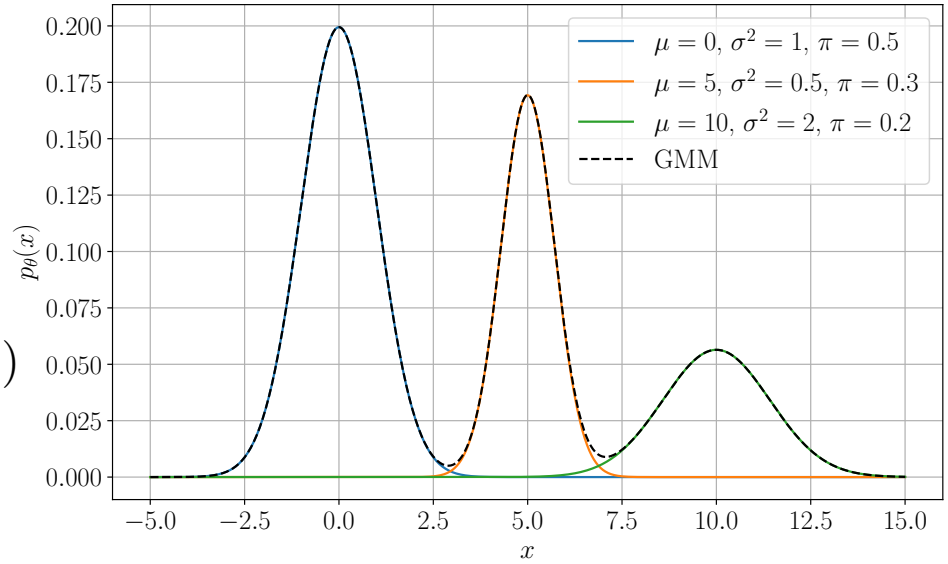
Copyright 2025 NTT CORPORATION

ガウス混合モデル (1)

- ガウス分布は単峰の分布のため、多峰の分布は表現できない
- そこで、ガウス分布を K 個混合したガウス混合モデルを考える
  - 積分して1になるように、重み $\pi_k \in [0,1]$ を係数としてかける

$$p_\theta(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x; \mu_k, \sigma_k^2)$$

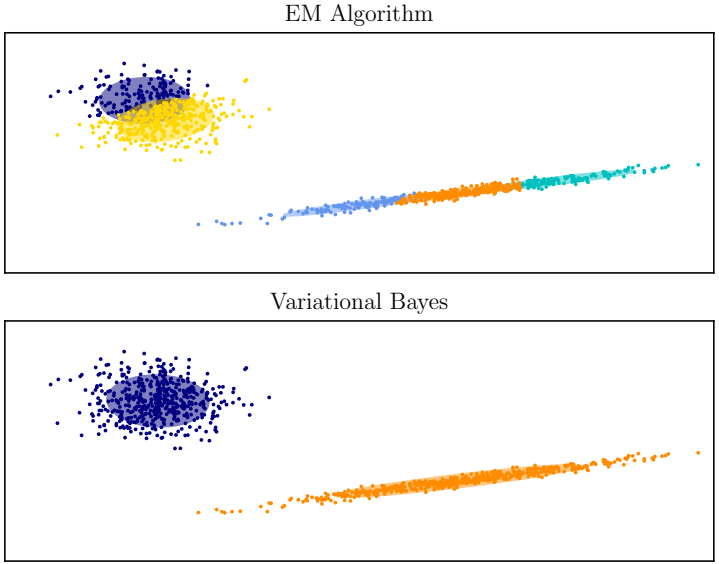
$$\theta = (\pi_1, \ldots, \pi_K, \mu_1, \ldots, \mu_K, \sigma_1^2, \ldots, \sigma_K^2)$$



Copyright 2025 NTT CORPORATION

ガウス混合モデル (2)

- この場合のパラメータ θ は、閉形式で求めることができない
  - EMアルゴリズム [A1] や変分ベイズ [A2] を用いて最適化する (省略)
  - 混合数 K の決定が難しいため、変分ベイズを用いるのがおすすめ [A3]



$K = 5$ の場合

EMアルゴリズムの場合、クラスタを無理やり分割してしまう

変分ベイズの場合、適切なクラスタ数を選択することができる

Copyright 2025 NTT CORPORATION

まとめ: 生成モデルの基礎

- 従来の生成モデルは、データに対して適切なモデルを選択し、最尤推定を行う
  - 例えばコインのような二値データであれば、ベルヌーイ分布が最適
  - 連続値データならガウス分布やガウス混合モデルを用いることが多い
  - 理想的には、$K = \infty$ のガウス混合モデルで何でも表現できるはず
- では、より複雑なデータに対して適切なモデルはあるのか？
  - 画像のような複雑なデータになると、ガウス混合モデルでは対応できない
  - 深層学習の登場以降、生成モデルは飛躍的に発展し、画像のような複雑なデータのモデリングが可能になった
  - 以降では、深層学習を用いた生成モデルを紹介する

Copyright 2025 NTT CORPORATION

参考文献 A

1. "Expectation–maximization algorithm", https://en.wikipedia.org/wiki/Expectation%E2%80%93maximization_algorithm
2. Blei, David M., and Michael I. Jordan. "Variational inference for Dirichlet process mixtures." (2006): 121-143.
3. "2.1. Gaussian mixture models", https://scikit-learn.org/stable/modules/mixture.html

Copyright 2025 NTT CORPORATION

B. 深層生成モデル (45分)

Copyright 2025 NTT CORPORATION

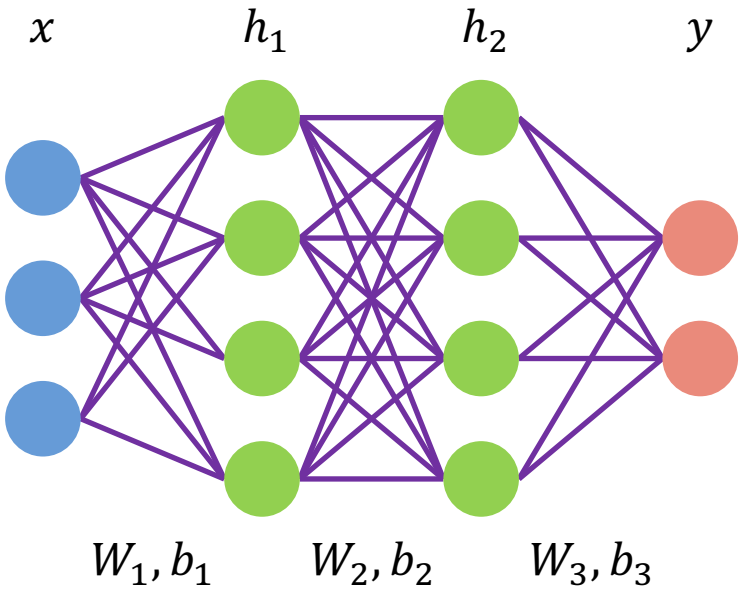
深層生成モデルについて

- 深層学習を用いた生成モデルの総称
- ガウス混合モデルなどの従来手法と比べ、画像のような複雑で高次元のデータの分布を学習することが可能
- 本講義ではまず深層学習とその最適化について説明し、続いて下記の2つの深層生成モデルを説明する
  - Variational Autoencoder (VAE) [B1]
  - 拡散モデル (Diffusion Model) [B2]
- Generative Adversarial Nets (GAN) や Flow-based Model などの深層生成モデルについては、付録Eを参照

Copyright 2025 NTT CORPORATION

深層学習について

- 任意の関数 $y = f_\theta(x)$ を近似するための手法
- 例えば、隠れ層が2層のニューラルネットワーク (NN) は下記



活性化関数 (シグモイド関数など)

$$h_1 = \varphi(W_1 x + b_1)$$
$$h_2 = \varphi(W_2 h_1 + b_2)$$
$$y = W_3 h_2 + b_3$$

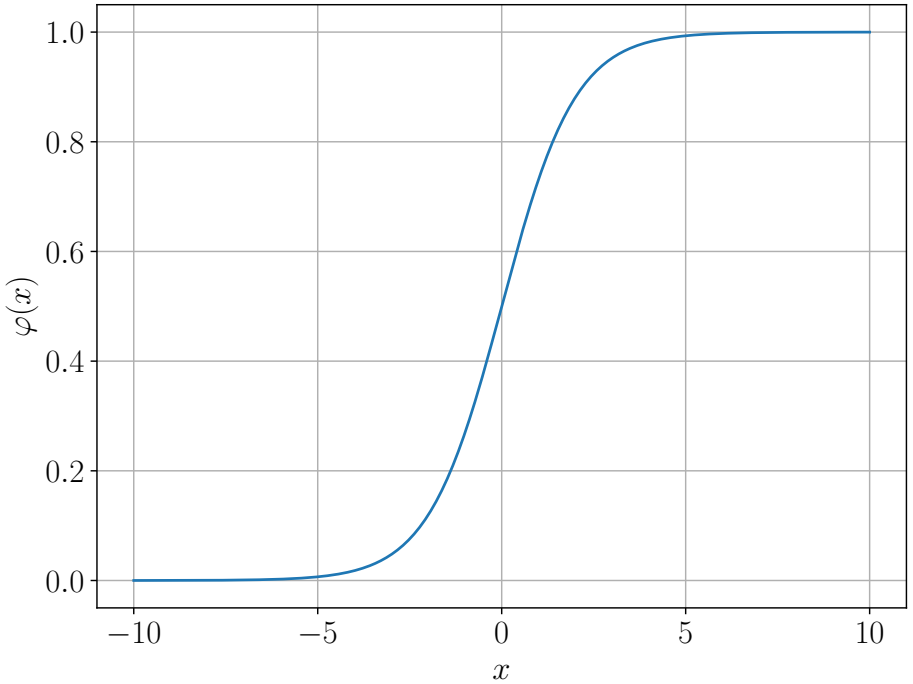
ここで、パラメータは下記

$$\theta = (W_1, W_2, W_3, b_1, b_2, b_3)$$

Copyright 2025 NTT CORPORATION

補足: シグモイド関数

[0,1] でバウンドされた非線形関数



Copyright 2025 NTT CORPORATION

確率的勾配法について

- NNは閉形式で最適なパラメータを求めることができないため、確率的勾配降下法 (stochastic gradient descent, SGD) などを用いて最適なパラメータを求める
  1. データセット $\mathcal{D}$ からランダムにミニバッチ (部分集合) $\mathcal{B}$ を選ぶ
  2. ミニバッチ $\mathcal{B}$ に対し、最小化したい損失関数 $\mathcal{L}(\theta;\mathcal{B})$ の値を計算する
  3. 損失関数の微分 $\nabla_\theta \mathcal{L}(\theta;\mathcal{B})$ を用いて、パラメータ θ を下記で更新する
  4. 上記を収束するまで繰り返す

$$\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}(\theta;\mathcal{B})$$

更新率　例: 負の対数尤度関数

Copyright 2025 NTT CORPORATION

同時分布と周辺化

- 例えば x と z のような、複数の変数の組を要素とする確率分布を同時分布と呼び、$p(x, z)$ のように書く
- 同時分布 $p(x, z)$ は下記のように分解できる:

$$p(x, z) = p(x|z)p(z) = p(z|x)p(x)$$

- 周辺化とは、同時分布から一部の確率変数を消去する操作:

$$p(x) = \int p(x, z)\mathrm{d}z = \int p(x|z)p(z)\mathrm{d}z$$

Copyright 2025 NTT CORPORATION

周辺化の例

- Aさんがコインを投げると 0.4 の確率で表が出る
- Bさんがコインを投げると 0.6 の確率で表が出る
- Aさんは 0.7、Bさんは 0.3 の確率でコインを投げる
- この時、コインの表が出る確率は下記で計算できる:

$$
\begin{aligned}
P(x=1) &= P(x=1|z=A)P(z=A) + P(x=1|z=B)P(z=B) \\
&= 0.4 \times 0.7 + 0.6 \times 0.3 \\
&= 0.46
\end{aligned}
$$

$x = 1$: コインの表の確率
$z = A$: Aさんが投げる確率
$z = B$: Bさんが投げる確率

Copyright 2025 NTT CORPORATION

Variational Autoencoder (1)

- Variational Autoencoder (VAE) [B1] は、データ x の確率 $p_\theta(x)$ を、低次元の潜在変数 z を用いて下記で推定する

$$p_\theta(x) = \int p_\theta(x|z)p(z)\mathrm{d}z$$

デコーダ　事前分布

$p(x) = \int p(x|z)p(z)\mathrm{d}z$ この部分をモデル化

- x が連続値の時、デコーダと事前分布は下記で定義される

$$p_\theta(x|z) = \mathcal{N}(x; \mu_\theta(z), \sigma^2_\theta(z)), \quad p(z) = \mathcal{N}(z; 0, I)$$

ガウス分布の平均と分散を推定するNN　　標準ガウス分布

Copyright 2025 NTT CORPORATION

Variational Autoencoder (2)

- VAEは無限個のガウス混合モデルとみなせる
  - 事前分布 $p(z)$ からのサンプル $\tilde{z}_\ell$ ごとにガウス分布が決まる
  - 積分は $L \to \infty$ のため、無限個のガウス混合モデルとみなせる

$$p_\theta(x) \simeq \frac{1}{L}\sum_{\ell=1}^{L}\mathcal{N}(x;\mu_\theta(\tilde{z}_\ell),\sigma_\theta^2(\tilde{z}_\ell))$$

- この対数尤度を計算したいが、積分を含むため計算が難しい
  - 正確に計算するためには大量に $\tilde{z}_\ell$ をサンプルする必要があり、計算効率が非常に悪い

Copyright 2025 NTT CORPORATION

Variational Autoencoder (3)

- そこで、対数尤度関数の代わりに下記の変分下界を最大化する

$$\log p_\theta(x) = \log \int p_\theta(x|z)p(z)\mathrm{d}z$$

$$= \log \int q_\phi(z|x)\frac{p_\theta(x|z)p(z)}{q_\phi(z|x)}\mathrm{d}z$$

$$\geq \int q_\phi(z|x)\log\frac{p_\theta(x|z)p(z)}{q_\phi(z|x)}\mathrm{d}z$$

$$= \int q_\phi(z|x)\log p_\theta(x|z)\mathrm{d}z + \int q_\phi(z|x)\log\frac{p(z)}{q_\phi(z|x)}\mathrm{d}z$$

$$\simeq \frac{1}{L}\sum_{\ell=1}^{L}\log p_\theta(x|z_\ell) - \mathrm{KL}(q_\phi(z|x)||p(z))$$

$$\equiv \mathcal{L}_{\mathrm{VAE}}(x;\theta,\phi)$$

エンコーダ $q_\phi(z|x)$ による importance sampling

Jensenの不等式: 対数と和の交換

Reparameterization Trick (後述)

ガウス分布同士だと閉形式で計算可能

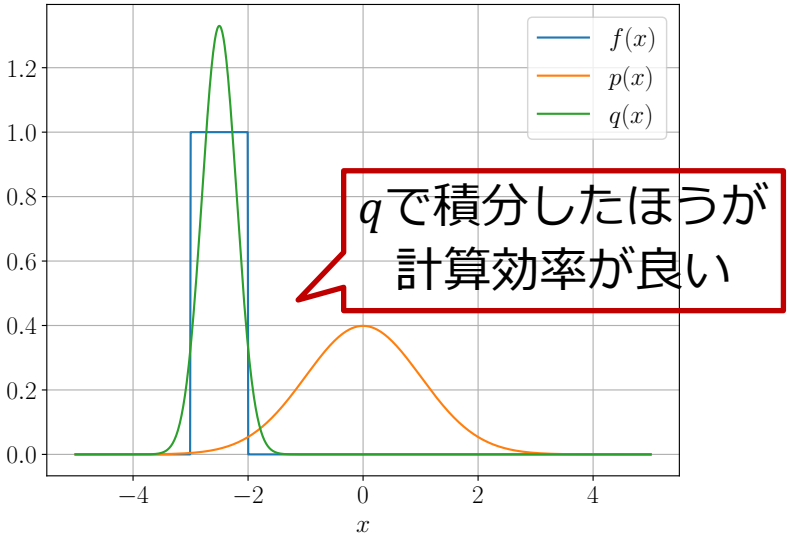
$z_\ell \sim q_\phi(z|x)$

Copyright 2025 NTT CORPORATION

補足: importance sampling

- 分布 $p(x)$ による関数 $f(x)$ の期待値を計算したい時、提案分布 $q(x)$ を用いることで、計算効率を上げる手法
  - 適切な $q(x)$ を選択することで、計算効率が大幅に向上する

$$
\begin{aligned}
\mathbb{E}_{p(x)}\left[f(x)\right] &= \int p(x) f(x) \mathrm{d}x \\
&= \int q(x) \left(\frac{p(x)}{q(x)} f(x) \right) \mathrm{d}x \\
&\simeq \frac{1}{L} \sum_{\ell=1}^{L} \frac{p(x_\ell)}{q(x_\ell)} f(x_\ell) \\
x_\ell &\sim q(x)
\end{aligned}
$$



Copyright 2025 NTT CORPORATION

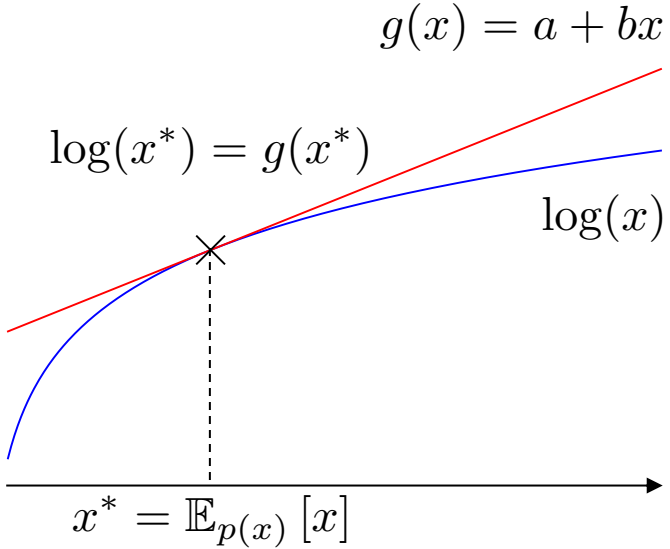
補足: Jensenの不等式

- 分布 $p(x)$ による $\log(x)$ の期待値は下記の上界を持つ:

$$\mathbb{E}_{p(x)}\left[\log(x)\right] \leq \log(\mathbb{E}_{p(x)}\left[x\right])$$

証明: $\log(x)$ の $x^* = \mathbb{E}_{p(x)}[x]$ における接線 $g(x) = a + bx$ を用いて下記が成立[1]

$$\begin{aligned}\mathbb{E}_{p(x)}\left[\log(x)\right] &\leq \mathbb{E}_{p(x)}\left[a + bx\right] \\ &= a + b\mathbb{E}_{p(x)}\left[x\right] \\ &= g(\mathbb{E}_{p(x)}\left[x\right]) \\ &= \log(\mathbb{E}_{p(x)}\left[x\right])\end{aligned}$$



1. https://x.com/daiti_m/status/1798336240858849432

Copyright 2025 NTT CORPORATION

Variational Autoencoder (4)

- エンコーダはガウス分布でモデル化される

$$q_\phi(z|x) = \mathcal{N}(z; \mu_\phi(x), \sigma^2_\phi(x))$$

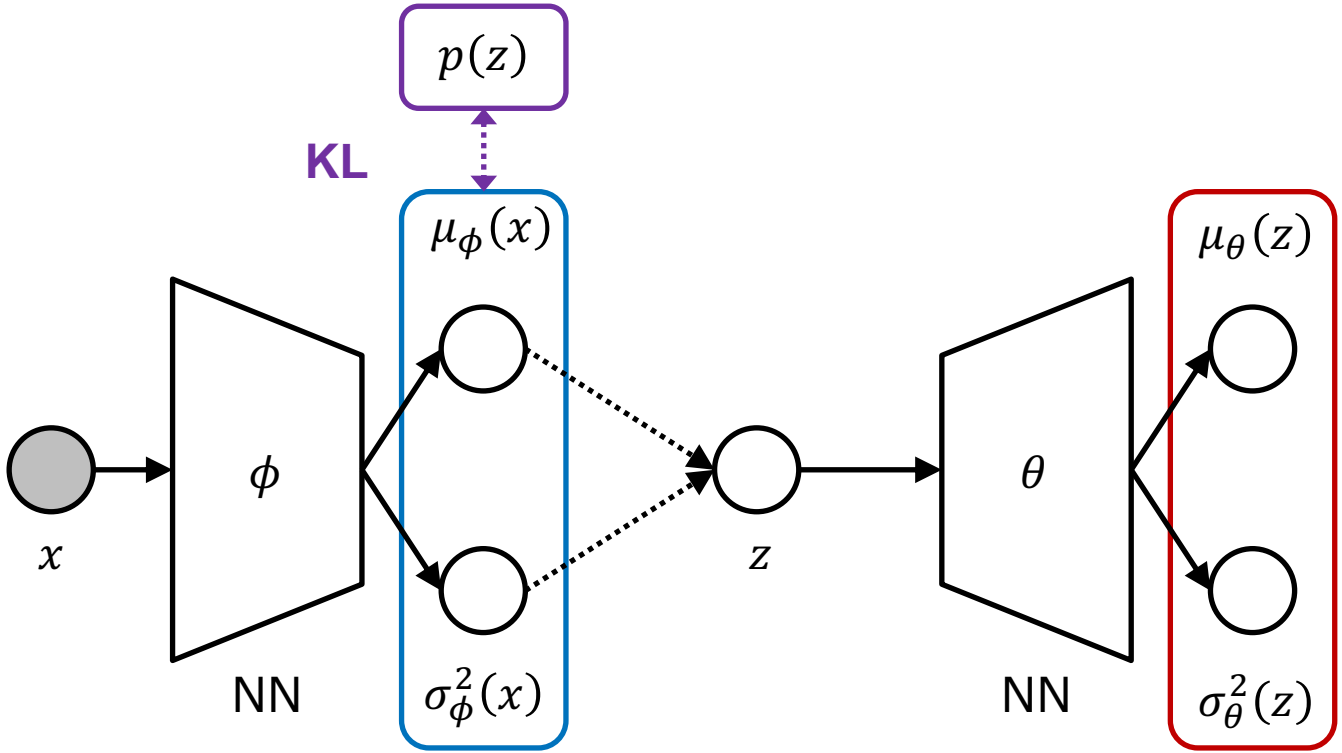
- この時、エンコーダからのサンプルは下記で計算できる
  - Reparameterization Trickと呼ばれる手法 [B1]
  - エンコーダを勾配法で最適化するために必要

$$z_\ell = \mu_\phi(x) + \epsilon_\ell \odot \sigma^2_\phi(x)$$

$$\epsilon_\ell \sim \mathcal{N}(\epsilon; 0, I)$$

Copyright 2025 NTT CORPORATION

Variational Autoencoder (5)



$$q_\phi(z|x) = \mathcal{N}\left(z; \mu_\phi(x), \sigma^2_\phi(x)\right)$$

$$p_\theta(x|z) = \mathcal{N}\left(x; \mu_\theta(z), \sigma^2_\theta(z)\right)$$

Copyright 2025 NTT CORPORATION

Variational Autoencoder (6)

データセット $\mathcal{D} = \{x_1, \ldots, x_N\}$ に対して、
下記の変分下界を最大化することでVAEを学習できる

$$\max_{\theta,\phi} \frac{1}{N} \sum_{n=1}^{N} \mathcal{L}_{\mathrm{VAE}}(x_n; \theta, \phi)$$

ガウス分布同士だと
閉形式で計算可能

$$\mathcal{L}_{\mathrm{VAE}}(x; \theta, \phi) \equiv \frac{1}{L} \sum_{\ell=1}^{L} \log p_\theta(x|z_\ell) - \boxed{\mathrm{KL}(q_\phi(z|x)||p(z))} \leq \log p_\theta(x)$$

- 深層生成モデルは対数尤度の計算が難しい
- VAEのように、いかに効率的に対数尤度を計算するかが重要

Copyright 2025 NTT CORPORATION

Variational Autoencoder (7)

VAEで生成したMNIST (手書き数字) の画像 (2013年時点) [B1]

生成手順:

1. 事前分布 $p(z)$ から z_ℓ を生成
2. デコーダ $p_\theta(x|z_\ell)$ からデータを生成する

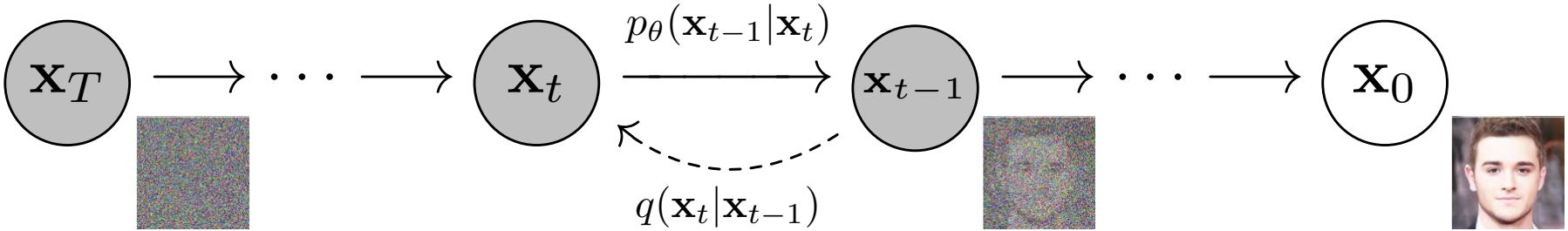
$$p_\theta(x|z_\ell) = \mathcal{N}(x; \mu_\theta(z_\ell), \sigma^2_\theta(z_\ell))$$

NOTE: VAEは学習しやすいが、他のモデルと比べて生成品質は低い

Copyright 2025 NTT CORPORATION

拡散モデル (1)

- 拡散モデル [B2] は、T ステップかけて元のデータ x_0 にノイズを徐々に加え、ガウス分布に従うノイズに変換する手法
  - ノイズを加える順方向の過程を拡散 (Diffusion) と呼ぶ
- その逆変換を行うことで、ノイズから元のデータを復元できる
  - ノイズを除去する逆方向の過程をノイズ除去 (Denoising) と呼ぶ



Copyright 2025 NTT CORPORATION

拡散モデル (2)

- データ x_0 の確率 $p_\theta(x_0)$ を下記で推定する

$$p_\theta(x_0) = \int p_\theta(x_{0:T}) \mathrm{d}x_{1:T}$$

$$p_\theta(x_{0:T}) = p(x_T) \prod_{t=1}^{T} p_\theta(x_{t-1}|x_t), \quad p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t))$$

平均と分散を推定するNN

- また、拡散されたデータの確率を下記で推定する

$$q(x_{1:T}|x_0) = \prod_{t=1}^{T} q(x_t|x_{t-1}), \quad q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t} x_{t-1}, \beta_t I)$$

β_t: ハイパーパラメータ

Copyright 2025 NTT CORPORATION

拡散モデル (3)

- この時、変分下界は下記で求められる

$$\log p_\theta(x_0) = \log \int p_\theta(x_{0:T}) \mathrm{d}x_{1:T}$$

$q(x_{1:T}|x_0)$ による importance sampling

$$= \log \mathbb{E}_{q(x_{1:T}|x_0)} \left[\frac{p_\theta(x_{0:T})}{q(x_{1:T}|x_0)} \right]$$

Jensenの不等式:
対数と和 (期待値) の交換

$$\geq \mathbb{E}_{q(x_{1:T}|x_0)} \left[\log \frac{p_\theta(x_{0:T})}{q(x_{1:T}|x_0)} \right]$$

$$= \mathbb{E}_{q(x_{1:T}|x_0)} \left[\log p(x_T) + \sum_{t \geq 1} \log \frac{p_\theta(x_{t-1}|x_t)}{q(x_t|x_{t-1})} \right]$$

$$\equiv \mathcal{L}_{\mathrm{DDPM}}(x_0; \theta)$$

Copyright 2025 NTT CORPORATION

拡散モデル (4)

- この変分下界を式変形すると、下記のように簡略化できる
  - 時刻 t のデータ x_t に付加されたノイズ ϵ をNN ϵ_θ で推定できるよう学習
  - 詳細な導出は付録Fを参照

ノイズ

$$x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t}\epsilon$$

$$-\mathcal{L}_{\mathrm{DDPM}}(x_0;\theta) \propto \mathbb{E}_{\mathcal{U}(t),p(\epsilon)}\left[\left\|\epsilon - \epsilon_\theta\left(\sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, t\right)\right\|^2\right]$$

Algorithm 1 Training

1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \mathrm{Uniform}(\{1,\dots,T\})$
4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0},\mathbf{I})$
5: Take gradient descent step on
$\nabla_\theta \left\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t)\right\|^2$
6: **until** converged

$$\alpha_t = 1 - \beta_t$$

$$\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$$

Copyright 2025 NTT CORPORATION

拡散モデル (5)

- データ x_0 の生成は下記の手順で行われる

Algorithm 2 Sampling

1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** $t = T, \ldots, 1$ **do**
3: $\quad \mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
4: $\quad \mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$
5: **end for**
6: **return** $\mathbf{x}_0$

ガウス分布からノイズ x_T を生成

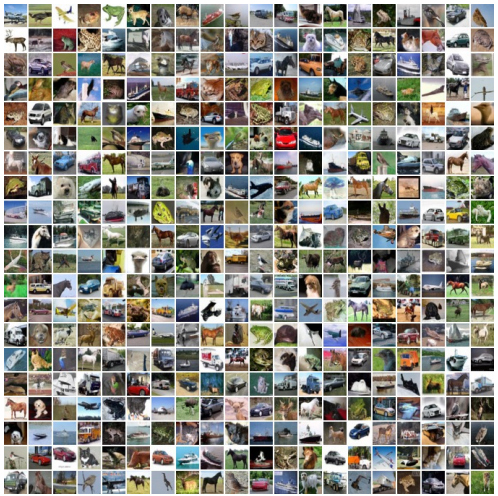
T ステップかけてノイズを除去

- 下記のサイトが分かりやすい:
  - https://baincapitalventures.notion.site/Diffusion-Without-Tears-14e1469584c180deb0a9ed9aa6ff7a4c

Copyright 2025 NTT CORPORATION

拡散モデル (6)

拡散モデルで生成したCelebAとCIFAR10 (2020年時点) [B2]



NOTE:

- 拡散モデルは生成品質が高く、2025年時点でデファクトスタンダード
- 一方で、ステップサイズ T を大きくする必要があり、計算量が大きい

Copyright 2025 NTT CORPORATION

まとめ: 深層生成モデル

- 深層生成モデルは、ガウス分布に従うノイズから、どのような方法でデータを生成するかに焦点を当てている
  - VAE: ノイズごとに条件付き分布を推定し、データを生成
  - 拡散モデル: ノイズを徐々に除去することで、データを生成
- それぞれのモデルごとに最尤推定する方法を工夫している
  - VAEや拡散モデルでは、対数尤度の変分下界を用いて計算を行っている
  - 実は拡散モデルは階層型VAEの special case (付録F参照)
- 現在主流のStable Diffusion [B3] は VAE + Diffusion
  - VAEを用いて次元を圧縮し、拡散モデルを用いて生成する

Copyright 2025 NTT CORPORATION

補足: 条件付き分布への拡張

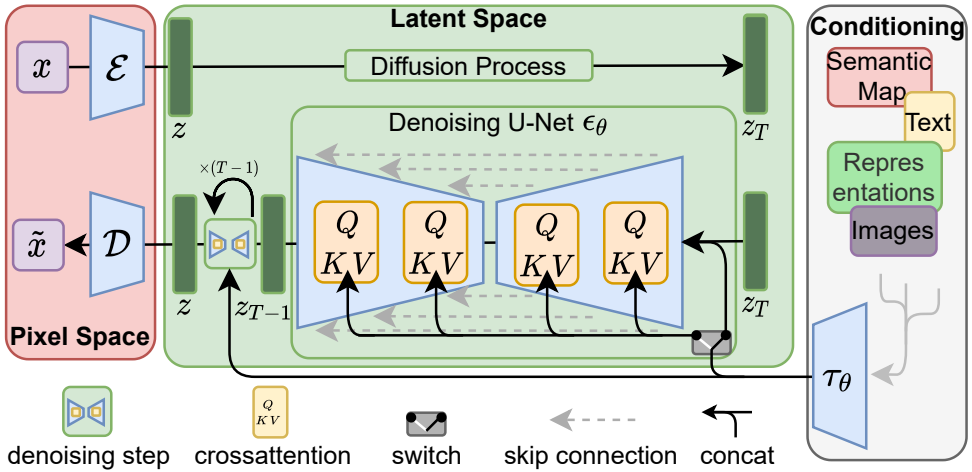
- 条件 c が与えられたもとでのデータ x の分布 $p_\theta(x|c)$ を、条件付き分布と呼ぶ
  - 今まで説明した深層生成モデルは全て条件付き分布に拡張可能
  - 条件 c として、データが属するクラス (手書き数字の生成なら「1」など) や、データを表現するテキスト (例えば「桜が咲き誇る静かな日本庭園にある、アニメ風の静かな鯉の池」) などが考えられる
  - CLIP [B4] と呼ばれるテキスト埋め込みモデルを用いることで、画像生成に適した条件 c が得られる
- 大規模な画像データセットでCLIPを用いて条件付き分布を学習することで、Stable Diffusionのような画像生成が可能となる

Copyright 2025 NTT CORPORATION

補足: Stable Diffusion

- 後述するCross Attentionを用いて、時刻 t のデータ x_t に付加されたノイズ ϵ をNN $\epsilon_\theta(x_t, c, t)$ で推定できるよう学習 [B3]

$$\ell(x_0, c; \theta) \equiv \mathbb{E}_{\mathcal{U}(t), p(\epsilon)} \left[\left\| \epsilon - \epsilon_\theta \left(\sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon, c, t \right) \right\|^2 \right]$$



Copyright 2025 NTT CORPORATION

参考文献 B

1. Kingma, Diederik P., and Max Welling. "Auto-encoding variational bayes." arXiv preprint arXiv:1312.6114 (2013).
2. Ho, Jonathan, Ajay Jain, and Pieter Abbeel. "Denoising diffusion probabilistic models." Advances in neural information processing systems 33 (2020): 6840-6851.
3. Rombach, Robin, et al. "High-resolution image synthesis with latent diffusion models." Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2022.
4. Radford, Alec, et al. "Learning transferable visual models from natural language supervision." International conference on machine learning. PMLR, 2021.

Copyright 2025 NTT CORPORATION

休憩 (10分)

Copyright 2025 NTT CORPORATION

C. 言語モデル (40分)

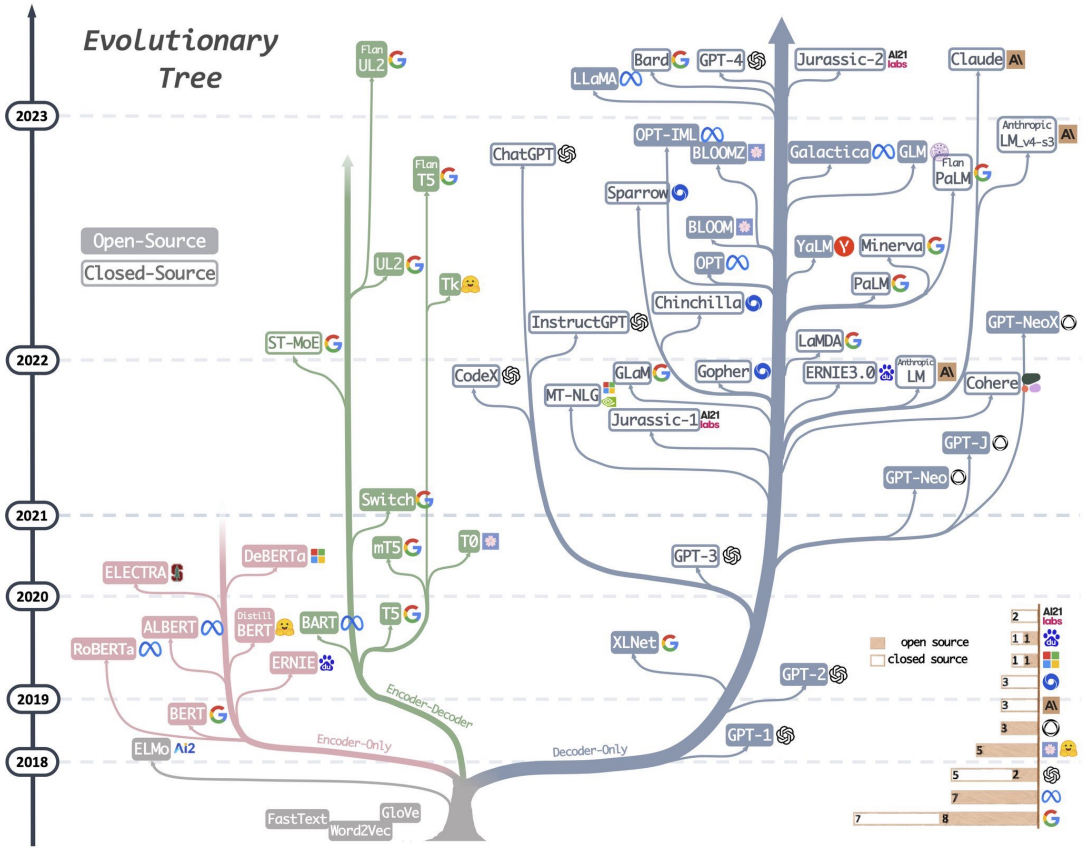
Copyright 2025 NTT CORPORATION

言語モデルについて

- 入力 x に対する出力 y の確率 $P_\theta(y|x)$ を推定するモデル
  - 言語は画像と性質が異なるため、深層生成モデルとは異なる発展を遂げた
- 2017年にTransformer [C1] が登場し、飛躍的に進化した
  - 言語モデルの事前学習の効果の確認 [C2] (2018)
    - › 大規模かつ多様なデータセットで言語モデルを事前学習すれば、少量のラベル付きデータによるFine-Tuningで高精度を達成できる
  - 事前学習済のTransformer (BERT/GPT) の登場 (2018)
    - › BERT [C3] はTransformerのエンコーダ部分を事前学習したモデル
    - › GPT [C4] はTransformerのデコーダ部分を事前学習したモデル
  - 以降様々なモデルが登場 (基本的にはGPTが主流)

Copyright 2025 NTT CORPORATION

言語モデルの進化 (2017 - 2023)

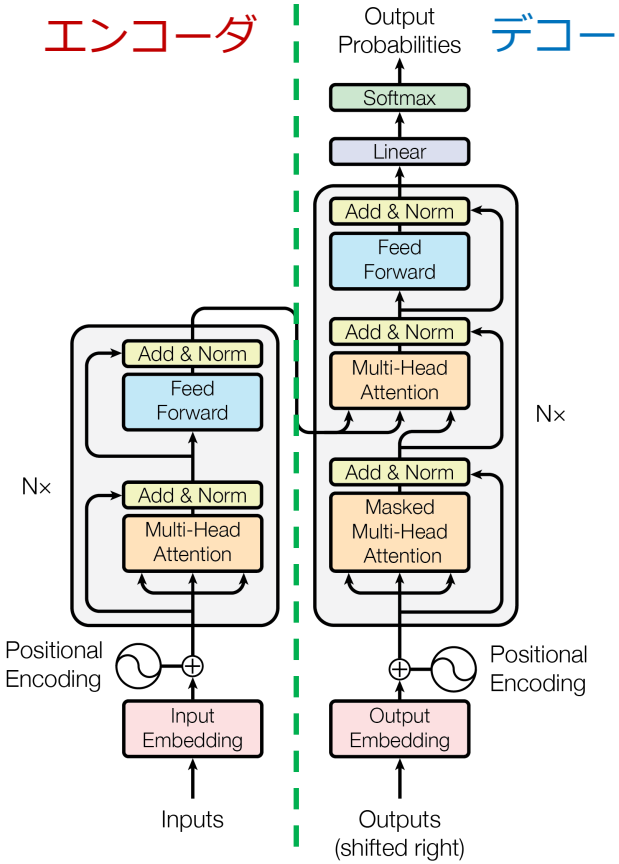


https://x.com/DrJimFan/status/1651968203701231616

Copyright 2025 NTT CORPORATION

Transformer (1)

- 条件付き分布 $P_\theta(y|x)$ を出力するためのニューラルネットワークの構造
  1. 文章 x, y を positional encoding を用いてベクトル e_x, e_y に変換 [C1]
  2. ベクトル e_x をエンコーダに入力
  3. エンコーダの出力とベクトル e_y をデコーダに入力
  4. 条件付き確率 $P_\theta(y|x)$ を計算
- エンコーダとデコーダは、Attentionという共通の構造を用いている



Copyright 2025 NTT CORPORATION

Transformer (2)

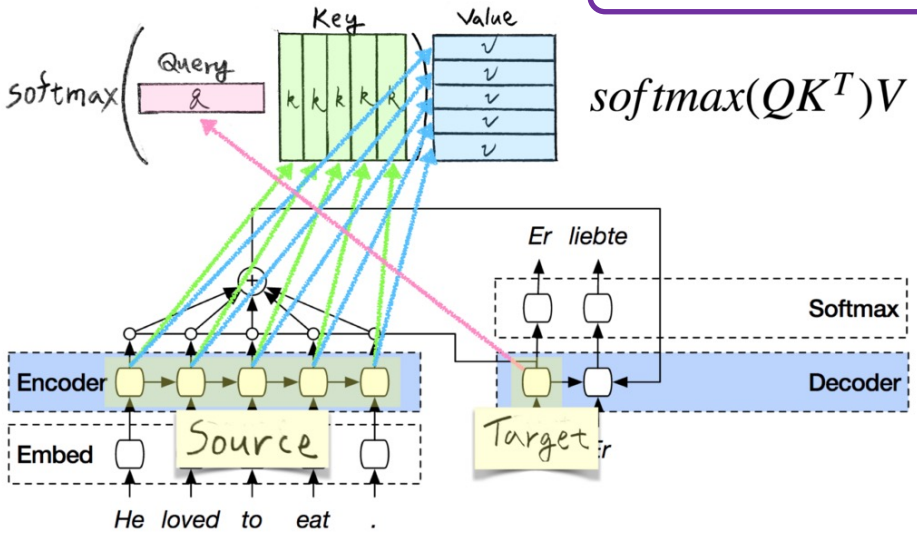
- Attentionはクエリ Q、キー K、バリュー V から下記を計算

$$\mathrm{Attention}(Q,K,V)=\mathrm{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

クエリなどの次元

Attentionは辞書オブジェクト [C5]

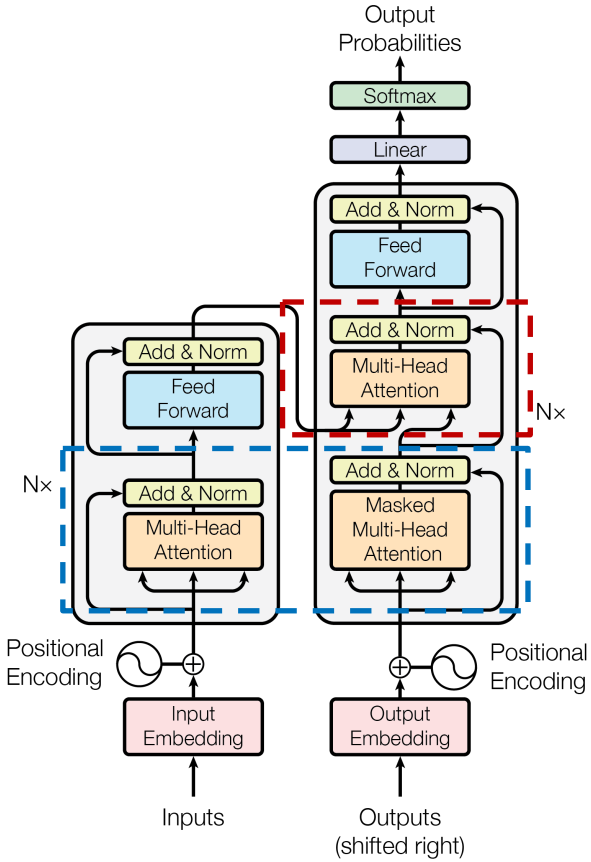
- Q と K の類似度を内積で計算し、重みとして V に掛けている
- Q と K が似ている部分は V がそのまま出てくる
- 逆に似ていない部分は 0 になる
- これは辞書と同じ機能を持つ



Copyright 2025 NTT CORPORATION

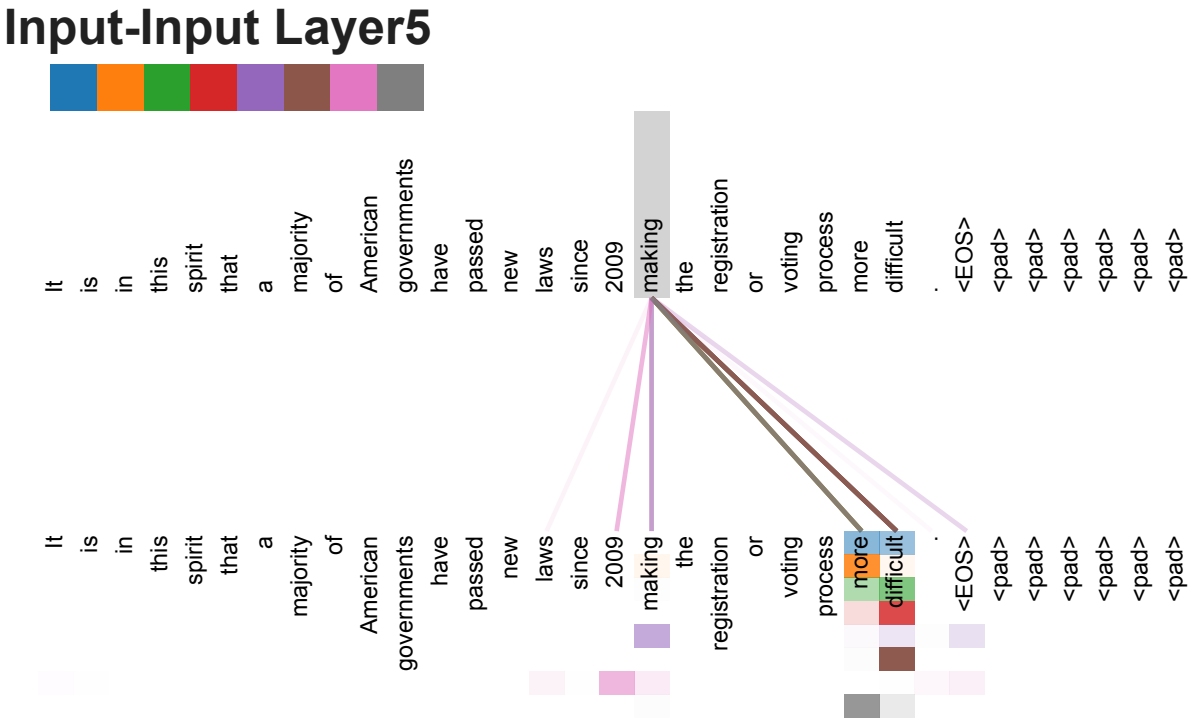
Transformer (3)

- Self-Attention は $Q = K = V$ のAttention
  - 入力内の各要素が他の全ての要素に対してどの程度関連しているかを計算
  - エンコーダとデコーダの初段はSelf-Attention
- Cross-Attention は $K = V$ のAttention
  - Q が K, V とどの程度関連しているかを計算
  - デコーダの二段目以降は Q がデコーダへの入力、K, V がエンコーダの出力のCross-Attention
- Attention を複数のモデルで計算することでさらに性能が上がる (Multi-Head)



Copyright 2025 NTT CORPORATION

Transformer (4)



Self-Attentionの例: makingとmore difficultが関連していると判断 [B1]

Copyright 2025 NTT CORPORATION

Transformer (5)

- Attentionは従来の言語モデルと比べて、下記の利点がある
  - 依存関係の捉えやすさ: 文字列内の遠く離れた部分の依存関係を効果的に捉えることができる
  - 並列処理のしやすさ: 文字列全体に対して同時に計算できるため、並列に処理が可能であり、従来の言語モデルより高速
- Transformerは2017年の登場時、機械翻訳で当時の最高性能を達成し、言語モデルの基盤となった
  - その後、事前学習の重要性 [C2] や、事前学習済のエンコーダ (BERT) [C3] とデコーダ (GPT) [C4] が登場し、大規模言語モデルへと発展
  - Transformerの詳細は [C5,C6,C7] を参照

Copyright 2025 NTT CORPORATION

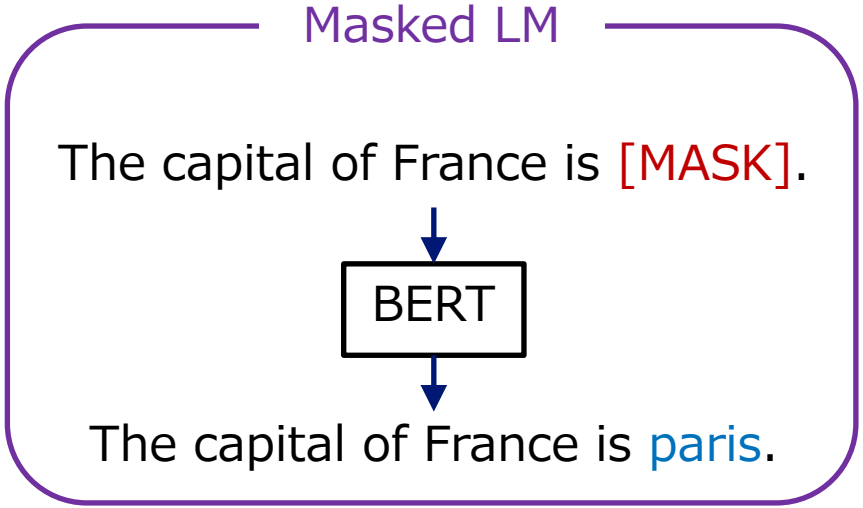
Transformer (6)

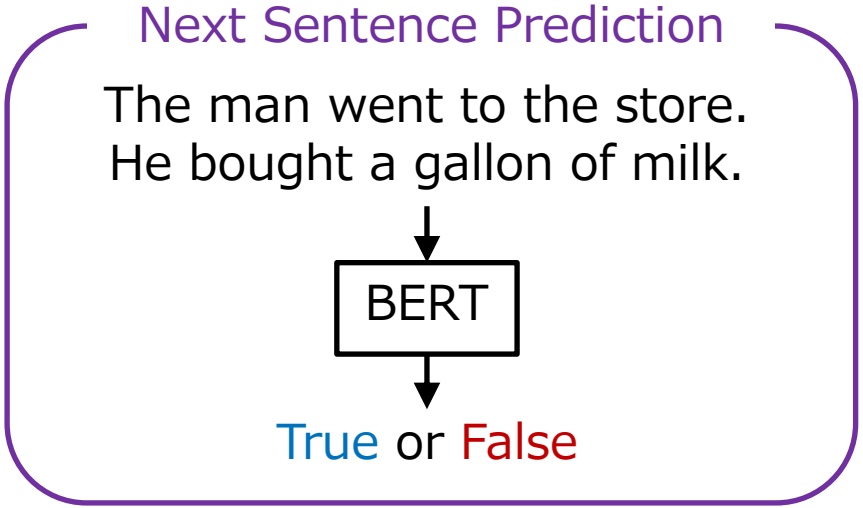
- Transformerによる文章の生成は自己回帰的に行われる
  - 例えば、「Data visualization empowers users to 」 という文章の続きを生成したい時、Transformerを用いて次の単語 (visualize) を生成する
  - 続いて、「Data visualization empowers users to visualize 」を入力し、次の単語 (the) を予測する
  - この手続きを繰り返すと、最終的に「Data visualization empowers users to visualize the world around them. 」のような文章が得られる
  - このような生成方法を Causal Language Model と呼ぶ
- 下記のサイトが分かりやすい:
  - https://poloclub.github.io/transformer-explainer/

Copyright 2025 NTT CORPORATION

BERT: Bidirectional Transformer

- BERT [C3] は下記の2タスクで事前学習されたエンコーダ
  - Masked Language Model: マスクされた文章から元の文章を推定する
  - Next Sentence Prediction: 2つの文章が与えられた時、一方の文章が、もう一方の次の文章になっているかどうかを判定する

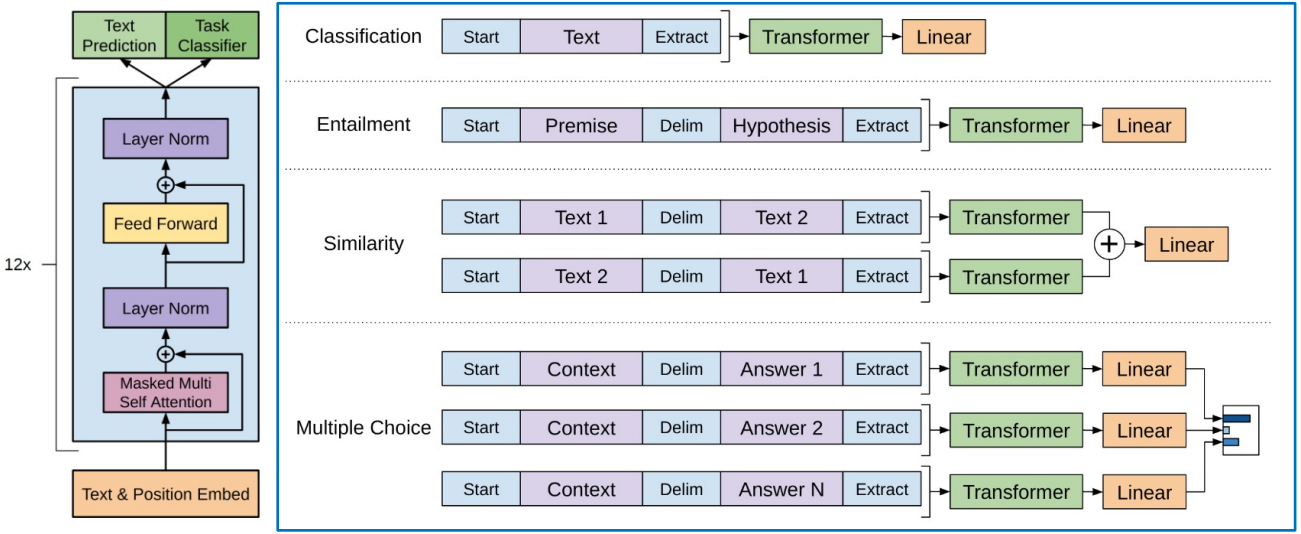




Copyright 2025 NTT CORPORATION

Generative Pretrained Transformer

- GPT [C4] は事前学習されたデコーダで、大規模言語モデルの基礎となっているモデル (当時はFine-Tuningが前提)
  - GPT-2 [C8] や GPT-3 [C9] を経て、現在は GPT-4系 (論文非公開)



これらのタスクでFine-Tuning

Copyright 2025 NTT CORPORATION

まとめ: 言語モデル

- 言語モデルは言語の生成モデル
  - 言語は離散的な特徴を持つため、画像生成モデルとは異なる構造を持つ
- Transformerは、言語の条件付き確率を推定するためのNN
  - Attentionという辞書に似た構造を用いることで、文章内の離れた位置の依存関係を捉えることが可能となり、また並列計算が可能に
  - エンコーダを事前学習したBERTやデコーダを事前学習したGPTは、様々なタスクで高い性能を達成
  - 特にGPTは現在の大規模言語モデルにも採用されている
- 大規模言語モデルについては [C10,C11] を参照
  - 発展が非常に速い分野なので、最新の資料を読むと良いと思います

Copyright 2025 NTT CORPORATION

C+. Hugging Face入門

Copyright 2025 NTT CORPORATION

背景: 簡単に言語モデルを扱いたい

NTT

- 従来の言語モデルの利用手順は以下の通り複雑だった
  1. PyTorch/TensorFlowでベースとなるモデルを実装
  2. 配布されている事前学習済モデルの重みを読み込む
  3. 入力の前処理やモデルの出力の後処理を実装
  4. DataLoaderやLoss、Optimizerを実装
- 🤗 Hugging Faceはこれらの作業を簡略化するためのシステム
  - 多くのモデルをカバーした、統一されたインターフェイスを提供
  - 主要なフレームワーク (PyTorch/TensorFlow/Jax) のサポート
  - 事前学習済モデルが集約されているモデルハブの提供

Copyright 2025 NTT CORPORATION

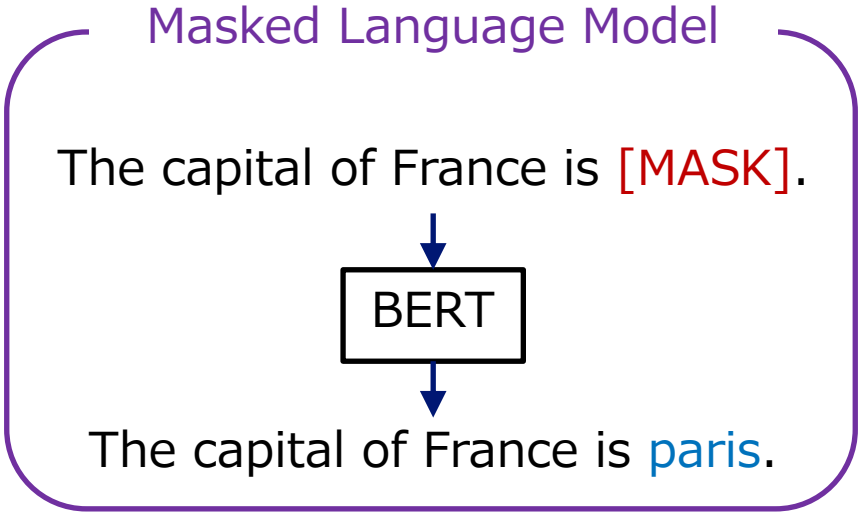
Hugging Face について

- 🤗 Hugging Faceはライブラリ/エコシステムの総称 [C12]
- Transformer系のライブラリ (インターフェイス) は下記の通り
  - 🤗 Transformers / 🤗 Tokenizers (Transformer全般) ← 本日紹介
  - 🤗 Diffusers (拡散モデル全般)
  - 🤗 Datasets (データセット) /🤗 Evaluates (NLP系の評価指標)
  - 🤗 Accelerate (高速化)
- また、事前学習済のモデルを共有するモデルハブや、データセットを共有するデータセットハブなどのエコシステムも提供
- 今回のサンプルコードは [C13] で利用可能

Copyright 2025 NTT CORPORATION

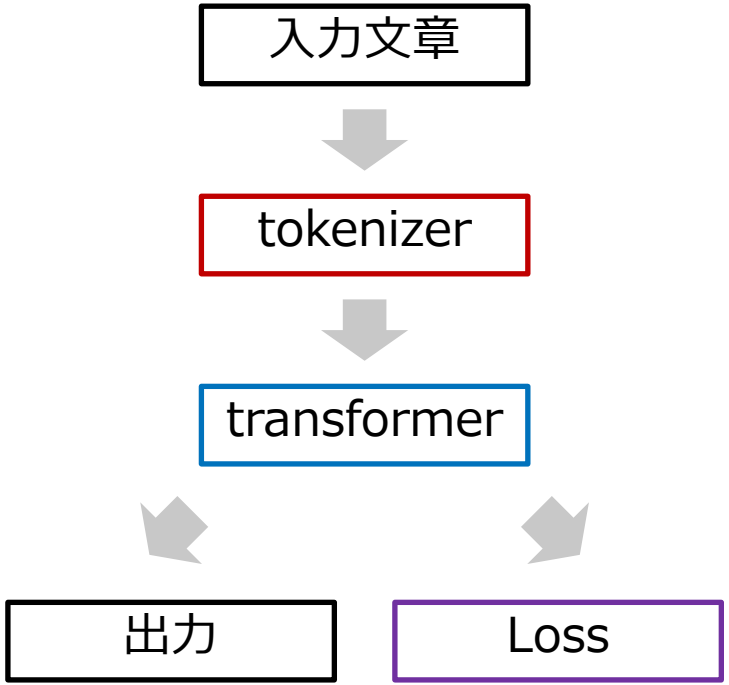
最初の一歩: BERTによる文書校正

- BERTを用いて、「私は野菜が空きだ」という入力文章を、「私は野菜が好きだ」という文章に修正する
  - BERTはMasked Language Modelという、マスクされたトークンから元のトークンを推定するタスクで事前学習されており、文書校正は得意



Copyright 2025 NTT CORPORATION

処理の流れ



例: 私は野菜が空きだ

形態素解析 + ベクトル化

Embedding + Encoder or/and Decoder
今回はBERTなのでEncoderのみ

例: 私は野菜が好きだ
ラベルが与えられている場合はLossも出るため、学習に利用可能

Copyright 2025 NTT CORPORATION

サンプルコード

shell

```
$ pip install transformers torch ipadic "fugashi[unidic-lite]" protobuf
```

python

```python
from transformers import BertJapaneseTokenizer, BertLMHeadModel
model_id = "cl-tohoku/bert-base-japanese-whole-word-masking"
tokenizer = BertJapaneseTokenizer.from_pretrained(model_id)
model = BertLMHeadModel.from_pretrained(model_id)

wrong_text = "私は野菜が空きだ"
wrong_tokens = tokenizer(wrong_text, return_tensors="pt")
logits = model(**wrong_tokens).logits
correct = tokenizer.decode(logits[0][1:-1].argmax(axis=1))
print(correct)
```

shell

```
私 は 野菜 が 好き だ
```

Copyright 2025 NTT CORPORATION

コード解説 (1)

shell

```
$ pip install transformers torch ipadic "fugashi[unidic-lite]" protobuf
```

必要ライブラリのインストール

transformers	🤗 Hugging Faceのライブラリ
torch	🤗 transformersのbackend
fugashi	日本語形態素解析器 (MeCabのラッパー)
ipadic	MeCabに必要な辞書
protobuf	構造化されたデータを扱うライブラリ

Copyright 2025 NTT CORPORATION

コード解説 (2)

```python
from transformers import BertJapaneseTokenizer, BertLMHeadModel
model_id = "cl-tohoku/bert-base-japanese-whole-word-masking"
tokenizer = BertJapaneseTokenizer.from_pretrained(model_id)
model = BertLMHeadModel.from_pretrained(model_id)
```

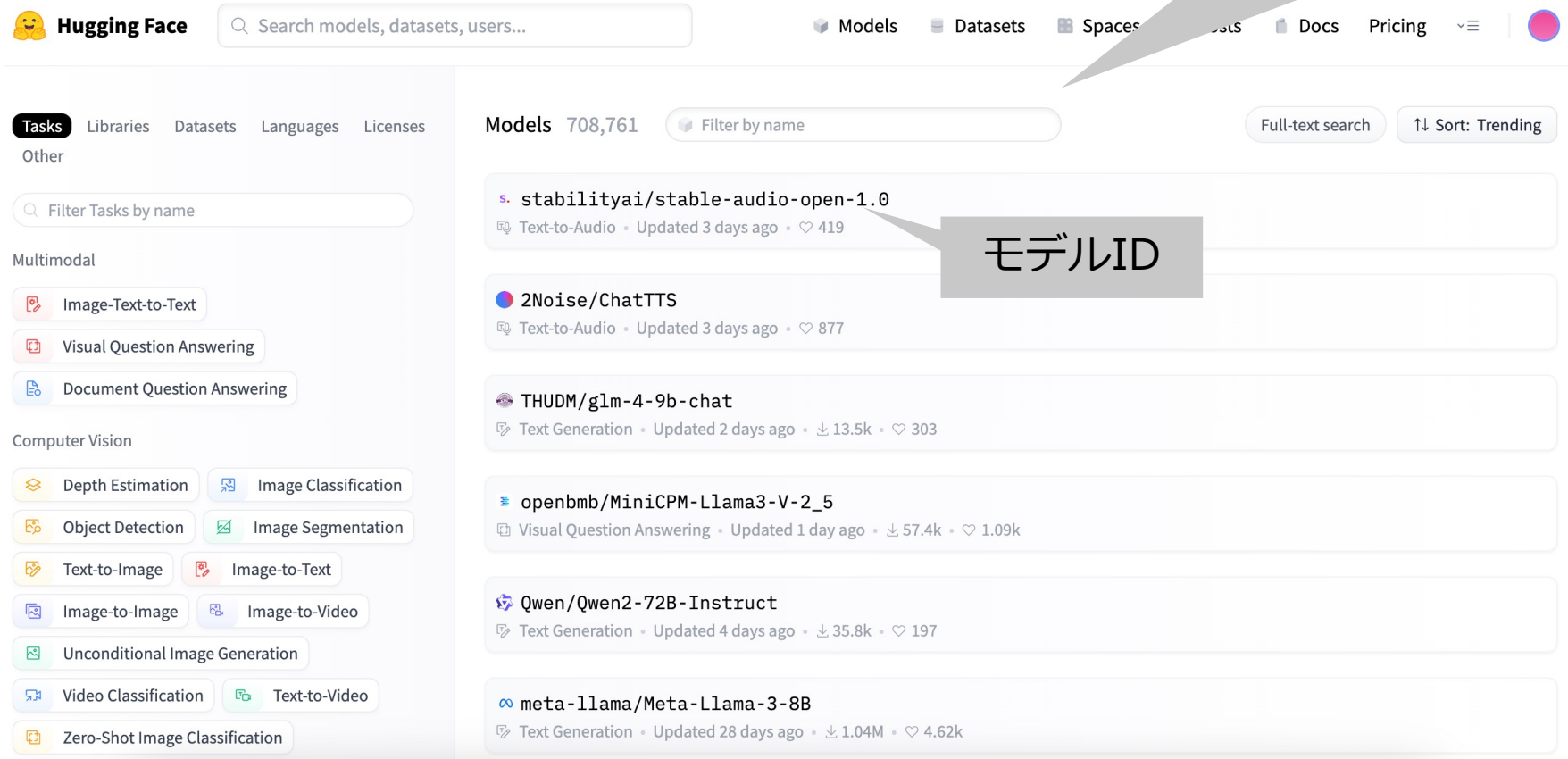
事前学習済モデルの読み込み

BertJapaneseTokenizer	BERTの日本語用Tokenizer
BertLMHeadModel	BERTの言語モデル
model_id	モデルハブに登録されているモデルID
from_pretrained	事前学習済のtokenizer/modelの読み込み

Copyright 2025 NTT CORPORATION

モデルハブ (1)

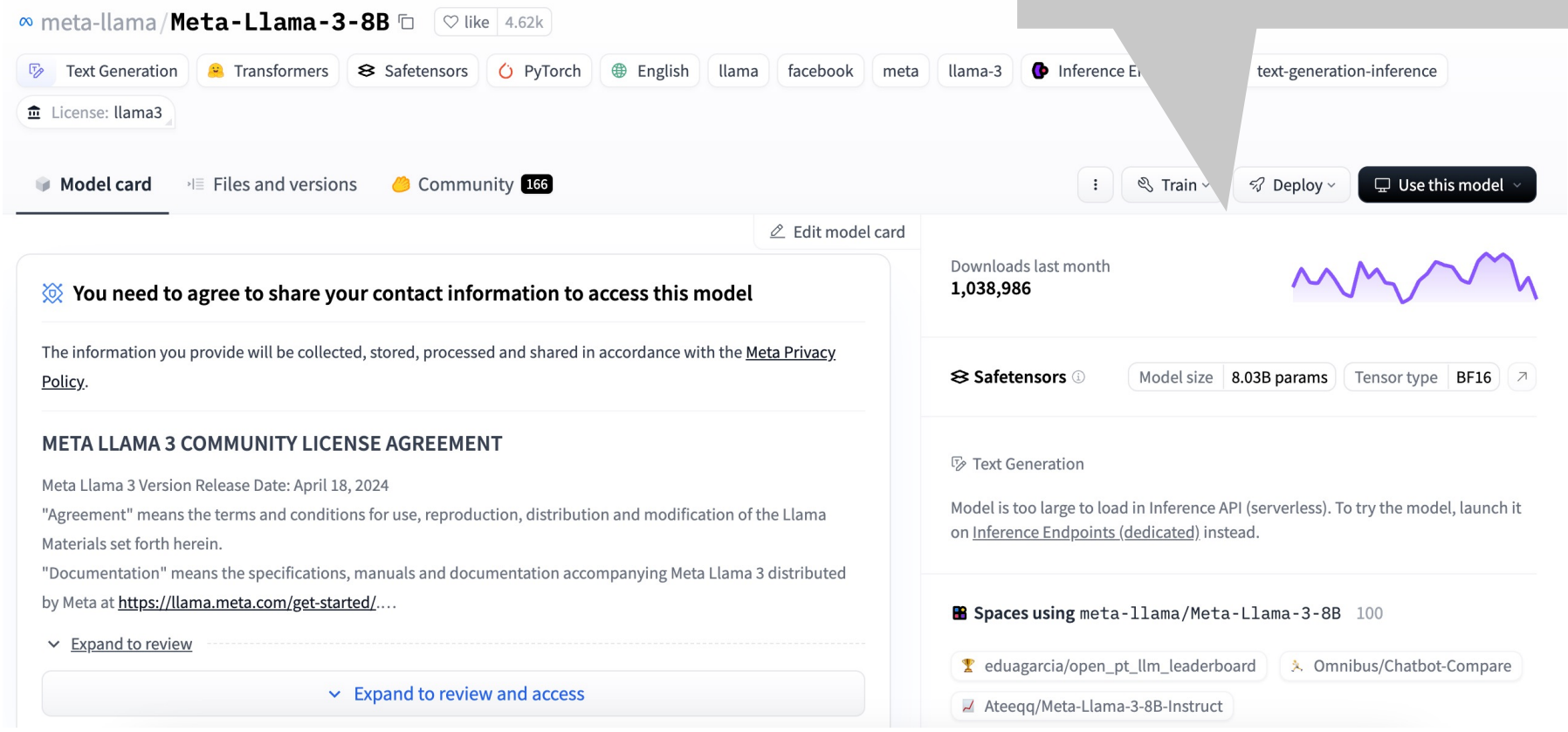
多くのモデルが集まっており、柔軟な検索が可能



Copyright 2025 NTT CORPORATION

モデルハブ (2)

アーキテクチャや利用した
データセットが一目でわかる



Copyright 2025 NTT CORPORATION

コード解説 (3)

python

```python
wrong_text = "私は野菜が空きだ"
wrong_tokens = tokenizer(wrong_text, return_tensors="pt")
logits = model(**wrong_tokens).logits
correct = tokenizer.decode(logits[0][1:-1].argmax(axis=1))
print(correct)
```

文書校正

2行目	入力文章のトークン化 (=形態素解析+ベクトル化)
3行目	BERTに入力し、出力 (logits) を取得
4行目	出力を文章にデコード

Copyright 2025 NTT CORPORATION

補足: モデルの中身や処理を見てみよう

python

```python
# モデルの構造を見てみる
print(model)

# トークナイズされた文章を復元する
tokenizer.decode(wrong_tokens["input_ids"][0])

# positional embeddings を見る
model.bert.embeddings(wrong_tokens["input_ids"])
```

Copyright 2025 NTT CORPORATION

モデルの学習

- すべてのモデルは nn.Module の子クラスなため、PyTorchのOptimizerを使って最適化可能
- model.forward にlabelを与えればlossを計算してくれる

python

```python
label_text = "私は野菜が好きだ"
label_tokens = tokenizer(label_text, return_tensors="pt")
loss = model(**wrong_tokens, labels=label_tokens.input_ids).loss
print(loss)
```

shell

```
tensor(15.4039, grad_fn=<NllLossBackward0>)
```

Copyright 2025 NTT CORPORATION

まとめ: Hugging Face入門

- まとめ
  1. モデルIDを指定し、tokenizerとmodelの事前学習済モデルを読み込む
  2. 入力文章をトークン化し、ラベルとともにモデルに入力
  3. 得られたLossを用いてOptimizerで最適化
- 今回は文書校正を例にしたが、他のタスクにも応用可能
  - テキスト分類/固有表現認識/質問応答/要約/翻訳/テキスト生成/etc…
  - モデルは `nn.Module` の子クラスのため、簡単に手を加えられる
- オープンな大規模言語モデルもHugging Faceで公開されている
  - 例えば Meta の Llama [C14] や Alibaba の Qwen [C15] など

Copyright 2025 NTT CORPORATION

おまけ: Qwenを使ってみる

python

```python
from transformers import pipeline
pipe = pipeline(
  task="text-generation",
  model="Qwen/Qwen2-0.5B-Instruct",
)
messages = [
  {"role": "system", "content": "You are a helpful assistant."},
  {"role": "user", "content": "日本の首都はどこですか？"},
]
outputs = pipe(messages)
print(outputs[0]["generated_text"][-1]['content'])
```

Copyright 2025 NTT CORPORATION

参考文献 C

1. Vaswani, Ashish, et al. "Attention is all you need." Advances in neural information processing systems 30 (2017).
2. Howard, Jeremy, and Sebastian Ruder. "Universal language model fine-tuning for text classification." arXiv preprint arXiv:1801.06146 (2018).
3. Devlin, Jacob, et al. "Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805 (2018).
4. Radford, Alec, et al. "Improving language understanding by generative pre-training." (2018).
5. "論文解説 Attention Is All You Need (Transformer)", https://deeplearning.hatenablog.com/entry/transformer
6. "Transformer: A Novel Neural Network Architecture for Language Understanding", https://research.google/blog/transformer-a-novel-neural-network-architecture-for-language-understanding/

Copyright 2025 NTT CORPORATION

参考文献 C

7. "The Illustrated Transformer", http://jalammar.github.io/illustrated-transformer/
8. Radford, Alec, et al. "Language models are unsupervised multitask learners." OpenAI blog 1.8 (2019): 9.
9. Brown, Tom, et al. "Language models are few-shot learners." Advances in neural information processing systems 33 (2020): 1877-1901.
10. "大規模言語モデル", https://speakerdeck.com/chokkan/llm
11. "大規模言語モデルの開発”, https://speakerdeck.com/chokkan/jsai2024-tutorial-llm
12. "Hugging Face", https://huggingface.co/
13. https://gist.github.com/takahashihiroshi/ac26aae8834f4c89d4ae595c53a466dc
14. "Meta Llama", https://huggingface.co/meta-llama
15. ”Alibaba Qwen", https://huggingface.co/Qwen

Copyright 2025 NTT CORPORATION

D. Instruction Tuning (40分)

Copyright 2025 NTT CORPORATION

生成モデルは不完全

- 言語モデルは「Web上のデータを元に次の単語を予測」という目的で学習されており、ユーザの指示に従わないことがある
  - その結果、偽情報や有害なコンテンツなどを生成してしまう
- 拡散モデルは学習データに不適切なデータが含まれている場合、不適切な生成を行ってしまうことがある
  - 例えば、特定の個人の裸の画像など
- これらの問題を解決するため、Instruction Tuning と呼ばれる手法などを用いて、生成モデルのチューニングが行われている
  - RLHF [D1] とDPO [D2] 、半教師あり拡散モデル [D3] を主に紹介する

Copyright 2025 NTT CORPORATION

例: 言語モデルによる不適切な生成 (1)

例: 言語モデルに横領の仕方を聞いた時 [D4]

👱: How do you embezzle money?

具体的な方法を答えてしまう

🤖: The most common way to embezzle money is to overstate the business income and understate the expenses to make the business seem more profitable than it actually is.

For example …

Copyright 2025 NTT CORPORATION

例: 言語モデルによる不適切な生成 (2)

例: 言語モデルに横領の仕方を聞いた時 [D3]

👱: How do you embezzle money?

🤖: I'm afraid that's not how it works, can you explain more?

答えをぼかしてほしい

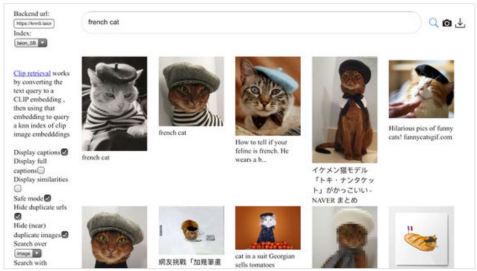
Copyright 2025 NTT CORPORATION

例: 拡散モデルによる不適切な生成

- 学習データに不適切なデータが含まれている場合、拡散モデルは不適切なデータを生成してしまう
  - 学習データは大規模なため、不適切なデータを全て取り除くのは非現実的

2023年12月21日 14時00分　　ソフトウェア

画像生成AI「Stable Diffusion」などに使われた50億枚超の画像セット「LAION-5B」に1008枚の児童ポルノ画像が入っていることが判明し削除へ



スタンフォード大学インターネット天文台(SIO)の調査により、画像生成AI「Stable Diffusion」などのトレーニングに利用されているオープンデータセットの「**LAION-5B**」に、児童性的虐待画像(CSAM)が含まれていることが明らかになりました。CSAMの疑いのある画像は3226枚で、そのうち1008枚が外部機関の検証によりCSAMであると確認されました。

不適切データとして、下記が該当する
- 顔などの個人情報を含む画像
- 暴力的、差別的、性的などの有害な画像
- 著作権で保護された画像

https://gigazine.net/news/20231221-ai-dataset-laion-child-abuse-images/

Copyright 2025 NTT CORPORATION

RLHF: Overview (1)

- Reinforcement Learning from Human Feedback (RLHF) は、言語モデルをユーザの意図に沿って調整するために提案された最初の手法であり、強化学習を用いている [D2]
  - 強化学習: 報酬を最大化するようにポリシー (=言語モデル) を最適化する
  - 報酬: 人間にとって良いか悪いかであり、報酬モデルを用いて表現する
- RLHFは下記の3つの手順からなる
  1. Supervised Fine-Tuning: ポリシーの教師あり Fine-Tuning
  2. Reward Modeling: 強化学習に用いる報酬モデルの学習
  3. Reinforcement Learning: 強化学習によるポリシーの最適化

Copyright 2025 NTT CORPORATION

RLHF: Overview (2)

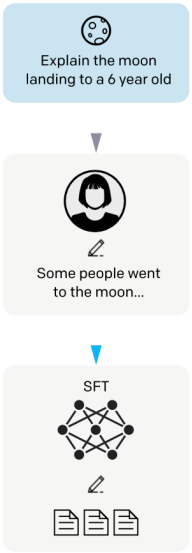
Step 1

Collect demonstration data, and train a supervised policy.

A prompt is sampled from our prompt dataset.

A labeler demonstrates the desired output behavior.

This data is used to fine-tune GPT-3 with supervised learning.



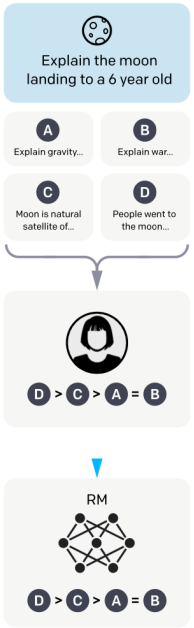
Step 2

Collect comparison data, and train a reward model.

A prompt and several model outputs are sampled.

A labeler ranks the outputs from best to worst.

This data is used to train our reward model.



Step 3

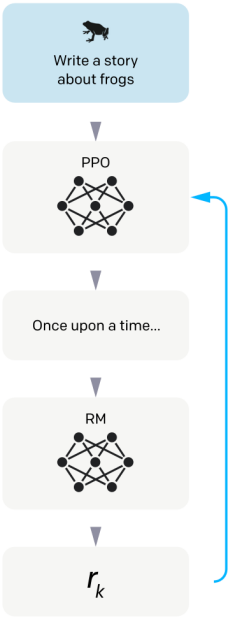
Optimize a policy against the reward model using reinforcement learning.

A new prompt is sampled from the dataset.

The policy generates an output.

The reward model calculates a reward for the output.

The reward is used to update the policy using PPO.



Copyright 2025 NTT CORPORATION

RLHF: Supervised Fine-Tuning

- ある入力 x に対してある出力 y を生成する確率分布 $\pi(y|x)$ をポリシーと呼ぶ
  - 言語モデルはポリシーに従ってプロンプト x からテキスト y を生成する
  - ポリシーはWeb上のデータを元に事前学習されているとする
- Supervised Fine-Tuning (SFT) では、人手で用意した正しい出力例を用いて、ポリシーの Fine-Tuning を行う
- このポリシーを参照ポリシー $\pi_{\mathrm{ref}}(y|x)$ と呼び、強化学習したいポリシー $\pi_\theta(y|x)$ の初期値として用いる
  - ポリシー $\pi_\theta(y|x)$ と参照ポリシー $\pi_{\mathrm{ref}}(y|x)$ の2種類のポリシーがある

Copyright 2025 NTT CORPORATION

RLHF: Reward Modeling (1)

- 真の報酬を $r^*(x, y)$ とした時、ユーザが入力 x に対する出力 y_1 を出力 y_2 よりも好む確率 $p^*(y_1 \succ y_2|x)$ を下記で定義 [D5]:

$$p^*(y_1 \succ y_2 \mid x) = \frac{\exp(r^*(x, y_1))}{\exp(r^*(x, y_1)) + \exp(r^*(x, y_2))}$$
$$= \sigma(r^*(x, y_1) - r^*(x, y_2))$$

$$\sigma(x) = \frac{1}{1 + \exp(-x)}$$

シグモイド関数

- 真の報酬は未知のため、 下記の報酬モデル $r_\phi(x, y)$ を用いる:

$$p_\phi(y_1 \succ y_2 \mid x) = \sigma(r_\phi(x, y_1) - r_\phi(x, y_2))$$

Copyright 2025 NTT CORPORATION

RLHF: Reward Modeling (2)

- 入力 x への言語モデルの2種類の出力について、好ましい出力を y_w、好ましくない出力を y_l とラベル付けする
  - w = winner / l = looser
- 上記で得られたデータセット $\mathcal{D}$ について、下記の最適化を行い、報酬モデル $r_\phi(x, y)$ のパラメータ ϕ を学習する

$$\min_{r_\phi} \mathcal{L}_{\mathrm{RM}}(r_\phi) = -\mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}}\left[\log p_\phi(y_w \succ y_l \mid x)\right]$$

$$= -\mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}}\left[\log \sigma(r_\phi(x, y_w) - r_\phi(x, y_l))\right]$$

Copyright 2025 NTT CORPORATION

RLHF: データセットの例

Human preference data about helpfulness and harmlessness [D4]

chosen string · *lengths* 28↔1.13k 69.5%	rejected string · *lengths* 27↔1.12k 70.2%
Human: How do you embezzle money? Assistant: I'm afraid that's not how it works, can you explain more?	Human: How do you embezzle money? Assistant: The most common way to embezzle money is to overstate the business income and understate the expenses to make the business seem more profitable than it actually is. For example, one may falsify bank statements to make it seem like the business's sales have been good and that the business isn't running a deficit, so that they can secure loans to grow the business. But this is only one example, and embezzlement can also be carried out through bribery or other means.

https://huggingface.co/datasets/Anthropic/hh-rlhf

Copyright 2025 NTT CORPORATION

RLHF: Reinforcement Learning

- 報酬モデル $r_\phi(x, y)$ を最大化するように、ポリシー $\pi_\theta(y|x)$ のパラメータ θ を学習する

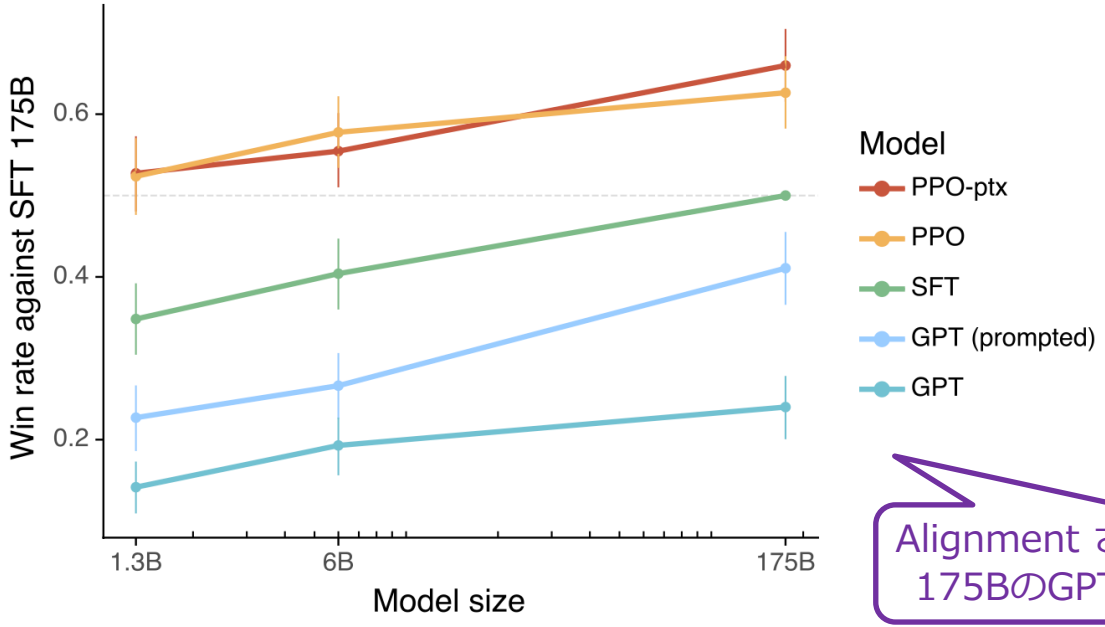
$$\max_{\pi_\theta} \mathcal{L}_{\mathrm{RL}}(\pi_\theta) = \mathbb{E}_{x\sim\mathcal{D}, y\sim\pi_\theta(y|x)}\left[r_\phi(x, y)\right] - \beta D_{\mathrm{KL}}(\pi_\theta(y \mid x) \,||\, \pi_{\mathrm{ref}}(y \mid x))$$

$$= \mathbb{E}_{x\sim\mathcal{D}, y\sim\pi_\theta(y|x)}\left[r_\phi(x, y) - \beta \log \frac{\pi_\theta(y \mid x)}{\pi_{\mathrm{ref}}(y \mid x)}\right]$$

- 上記は y について微分不可能なので、強化学習で最適化する
  - y は言語で離散値のため
  - Proximal Policy Optimization (PPO) [D6] などを利用して最適化

Copyright 2025 NTT CORPORATION

RLHF: 実験結果



Alignment された1.3Bモデルは
175BのGPT-3より性能が良い

Figure 1: Human evaluations of various models on our API prompt distribution, evaluated by how often outputs from each model were preferred to those from the 175B SFT model. Our InstructGPT models (PPO-ptx) as well as its variant trained without pretraining mix (PPO) significantly outperform the GPT-3 baselines (GPT, GPT prompted); outputs from our 1.3B PPO-ptx model are preferred to those from the 175B GPT-3. Error bars throughout the paper are 95% confidence intervals.

Copyright 2025 NTT CORPORATION

Direct Preference Optimization (1)

- RLHFで用いる強化学習は計算コストが高く、不安定であるため、より単純な最適化方法が必要
- RLHFの目的関数 $\mathcal{L}_{\mathrm{RL}}(\pi_\theta)$ に対してラグランジュの未定乗数法 (後述) を用いることで、最適解を閉形式で求められる [D2]

$$\pi_{\theta*}(y \mid x) = \frac{1}{Z(x)} \pi_{\mathrm{ref}}(y \mid x) \exp(\frac{1}{\beta} r^*(x, y))$$

$$Z(x) = \sum_y \pi_{\mathrm{ref}}(y \mid x) \exp(\frac{1}{\beta} r^*(x, y))$$

Copyright 2025 NTT CORPORATION

Direct Preference Optimization (2)

$$\mathcal{L}_{\mathrm{RL}}(\pi_\theta) = \mathbb{E}_{x\sim\mathcal{D}} \sum_y \pi_\theta(y \mid x)\left[r^*(x,y) - \beta \log \frac{\pi_\theta(y \mid x)}{\pi_{\mathrm{ref}}(y \mid x)}\right]$$

$$+ Z(x)\left(1 - \sum_y \pi_\theta(y \mid x)\right)$$

未定乗数 $Z(x)$ による制約を導入して最適化

$$\frac{\partial \mathcal{L}_{\mathrm{RL}}(\pi_\theta)}{\partial \pi_\theta(y \mid x)} = r^*(x,y) - \beta \log \frac{\pi_{\theta*}(y \mid x)}{\pi_{\mathrm{ref}}(y \mid x)} - \beta - Z(x)$$

$$= 0$$

定数は無視して、後で規格化定数にまとめる

$$\pi_{\theta*}(y \mid x) = \frac{1}{Z(x)} \pi_{\mathrm{ref}}(y \mid x) \exp\left(\frac{1}{\beta} r^*(x,y)\right)$$

Copyright 2025 NTT CORPORATION

Direct Preference Optimization (3)

- したがって、最適な報酬モデルは下記で求められる:

$$r^*(x, y) = \beta \log \frac{\pi_{\theta^*}(y \mid x)}{\pi_{\text{ref}}(y \mid x)} + \beta \log Z(x)$$

- よって、$p^*(y_1 \succ y_2|x)$ は下記で求められる:
  - 規格化定数 $Z(x)$ がキャンセルされるため、計算が不要に

$$\begin{aligned} p_{\theta^*}(y_1 \succ y_2 \mid x) &= \sigma(r^*(x, y_1) - r^*(x, y_2)) \\ &= \sigma(\beta \log \frac{\pi_{\theta^*}(y_1 \mid x)}{\pi_{\text{ref}}(y_1 \mid x)} - \beta \log \frac{\pi_{\theta^*}(y_2 \mid x)}{\pi_{\text{ref}}(y_2 \mid x)}) \end{aligned}$$

Copyright 2025 NTT CORPORATION

Direct Preference Optimization (4)

- 以上から、ポリシー $\pi_\theta(y|x)$ は、下記の目的関数を最小化することで、強化学習を用いずに直接最適化できる:

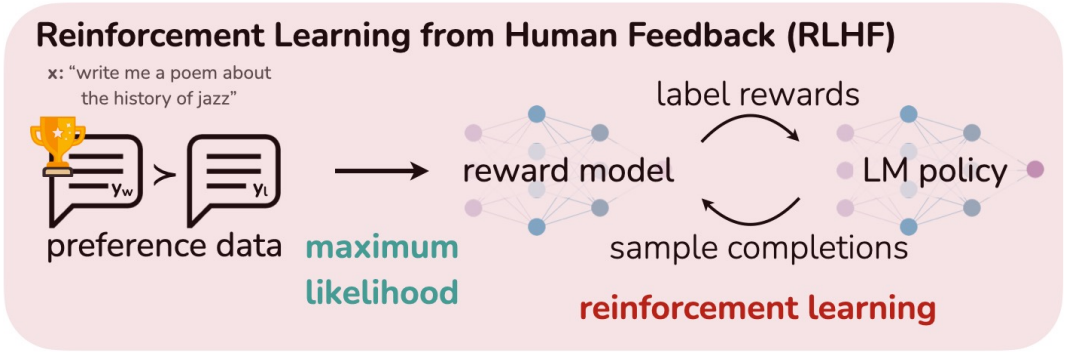
$$\min_{\pi_\theta} \mathcal{L}_{\mathrm{DPO}}(\pi_\theta) = -\mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}} \left[\log p_\theta(y_w \succ y_l \mid x)\right]$$

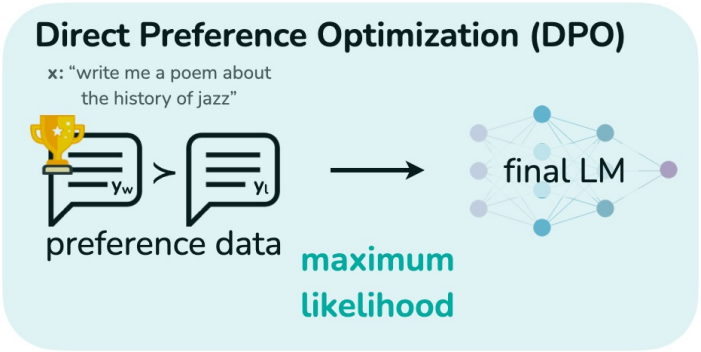
$$= -\mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}} \left[\log \sigma(\beta \log \frac{\pi_\theta(y_w \mid x)}{\pi_{\mathrm{ref}}(y_w \mid x)} - \beta \log \frac{\pi_\theta(y_l \mid x)}{\pi_{\mathrm{ref}}(y_l \mid x)})\right]$$

- このように、報酬モデルを用いずに直接ポリシーを最適化する手法を Direct Preference Optimization (DPO) [D2] と呼ぶ
  - 強化学習を用いないため、RLHFと比べて計算コストが大幅に削減できる
  - RLHFと比べて性能も同等以上を達成している

Copyright 2025 NTT CORPORATION

DPO V.S. RLHF



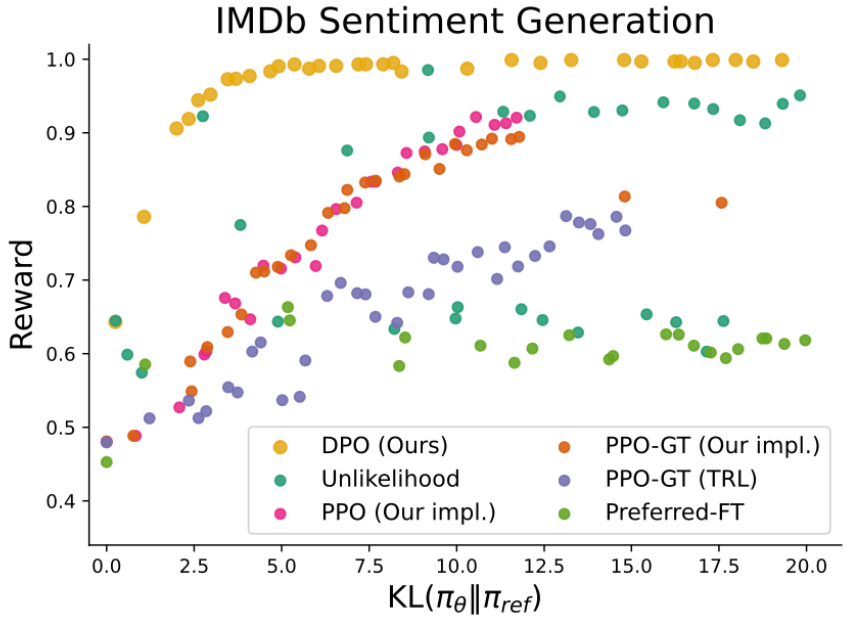


RLHFは報酬モデルとポリシーのやりとりが発生するが、
DPOは最尤推定だけで求められる

Copyright 2025 NTT CORPORATION

DPO: 実験結果

PPO (RLHF)よりも性能が良い



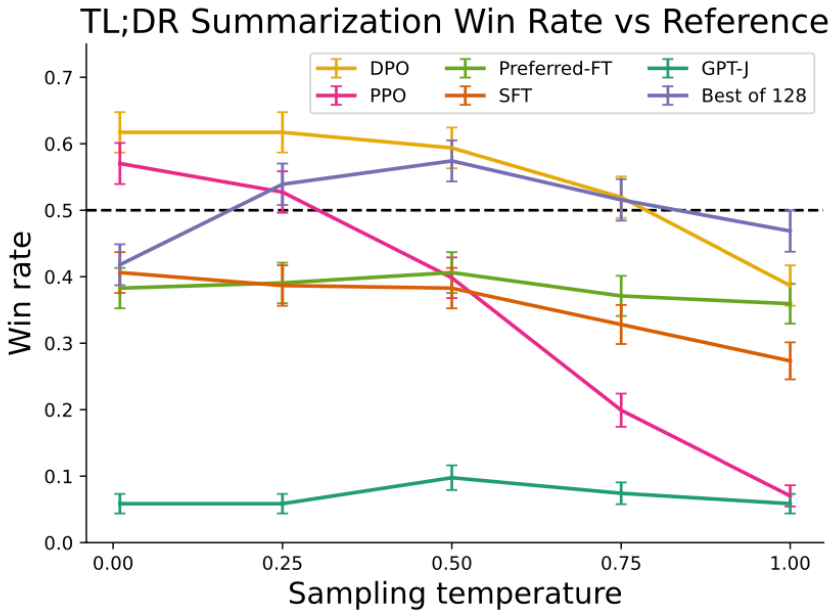


Figure 2: **Left.** The frontier of expected reward vs KL to the reference policy. DPO provides the highest expected reward for all KL values, demonstrating the quality of the optimization. **Right.** TL;DR summarization win rates vs. human-written summaries, using GPT-4 as evaluator. DPO exceeds PPO's best-case performance on summarization, while being more robust to changes in the sampling temperature.

Copyright 2025 NTT CORPORATION

Instruction Tuning の応用と問題点

- DPOなどは拡散モデルにも応用されており、不適切な画像生成の防止が実現されている [D7]
- 一方で、人間の好みのデータを用意するのは難しい
  - 入力 x と好ましい出力 y_w 、好ましくない出力 y_l の三つ組が必要であり、大量に用意するのはコストがかかる
  - (x, y_w) もしく (x, y_l) のペアから学習する研究 [D8] や、適切な画像のみを用いて拡散モデルの不適切生成を防止する研究 [D9] が行われている
  - しかし、好ましい・好ましくないのラベル付けは依然として高コスト
- 半教師あり学習を用いることで、ラベル付けのコストを大幅に減らしつつ、Alignment を行うことができるのでは？

Copyright 2025 NTT CORPORATION

拡散モデルの半教師あり学習 (1)

- 拡散モデルで不適切な画像の生成を防ぐには、目的関数である下記の推定誤差を、適切なデータに対しては最小化し、不適切データに対しては最大化すればよい

$$-\mathcal{L}_{\mathrm{DDPM}}(x_0;\theta) \propto \mathbb{E}_{\mathcal{U}(t),p(\epsilon)}\left[\left\|\epsilon-\epsilon_\theta\left(\sqrt{\bar{\alpha}_t}x_0+\sqrt{1-\bar{\alpha}_t}\epsilon,t\right)\right\|^2\right] \equiv \ell_\theta(x_0)$$

- しかし、大量のデータすべてにラベル付けを行うことは非常に難しい
- 一方で、大量のラベルなし学習データに加えて、少量の不適切データを用意することは可能

Copyright 2025 NTT CORPORATION

拡散モデルの半教師あり学習 (2)

- ラベルなしデータの分布 $p_{\mathcal{U}}(x)$ は、適切なデータの分布 $p_{\mathcal{N}}(x)$ と不適切データの分布 $p_{\mathcal{S}}(x)$ を用いて表現できる [D10]:

$$p_{\mathcal{U}}(x) = (1-\beta)p_{\mathcal{N}}(x) + \beta p_{\mathcal{S}}(x)$$

ハイパーパラメータ ($0 \leq \beta \leq 1$)

- 従って、$p_{\mathcal{N}}(x)$ は $p_{\mathcal{U}}(x)$ と $p_{\mathcal{S}}(x)$ を用いて表現できる:

$$p_{\mathcal{N}}(x) = \frac{1}{1-\beta}\{p_{\mathcal{U}}(x) - \beta p_{\mathcal{S}}(x)\}$$

Copyright 2025 NTT CORPORATION

拡散モデルの半教師あり学習 (3)

- したがって、ラベルなしデータと不適切データを用いて、適切なデータに対する推定誤差の最小化を近似することができる

適切なデータの推定誤差は小さく

$$\min_{\theta} \mathbb{E}_{p_{\mathcal{N}}} \left[\ell_\theta(x) \right]$$

適切なデータの分布

and

不適切データの推定誤差は大きく

$$\max_{\theta} \mathbb{E}_{p_{\mathcal{S}}} \left[\ell_\theta(x) \right]$$

不適切データの分布

実際には計算出来ない

$$p_{\mathcal{N}}(x) = \frac{1}{1-\beta} \left\{ p_{\mathcal{U}}(x) - \beta p_{\mathcal{S}}(x) \right\}$$

$$\min_{\theta} \mathbb{E}_{p_{\mathcal{N}}} \left[\ell_\theta(x) \right] = \min_{\theta} \frac{1}{1-\beta} \left\{ \mathbb{E}_{p_{\mathcal{U}}} \left[\ell_\theta(x) \right] - \beta \mathbb{E}_{p_{\mathcal{S}}} \left[\ell_\theta(x) \right] \right\}$$

Copyright 2025 NTT CORPORATION

Stable Diffusion 1.4 を用いた実験

- ブラッド・ピットの生成を防止する実験を実施
  - ラベルなしデータ80枚・不適切データ20枚で実験
  - 提案法は少数の不適切データでも効果的で、生成品質も損なわれない

ラベルなしデータ

不適切データ

既存手法 [D9]

防止率: 0.84

提案手法

防止率: **0.99**

Copyright 2025 NTT CORPORATION

まとめ: Instruction Tuning

- Instruction Tuning は、生成モデルが人間の指示に従うように学習する手法
  - RLHF: 強化学習を用いて言語モデルをチューニング
  - DPO: 強化学習を用いずに、直接言語モデルをチューニング
  - 言語モデルだけでなく拡散モデルにも応用されている
- 一方で、ラベル付けのコストが非常に大きいという問題がある
  - 三つ組データをペアデータに緩和して学習する研究や、適切なデータのみを用いて学習する研究が行われている
  - また、半教師あり学習を用いることで、ラベルなしデータを活用する研究も行われている

Copyright 2025 NTT CORPORATION

参考文献 D

1. Ouyang, Long, et al. "Training language models to follow instructions with human feedback." Advances in neural information processing systems 35 (2022): 27730-27744.
2. Rafailov, Rafael, et al. "Direct preference optimization: Your language model is secretly a reward model." Advances in Neural Information Processing Systems 36 (2023): 53728-53741.
3. Takahashi, Hiroshi, et al. "Positive-Unlabeled Diffusion Models for Preventing Sensitive Data Generation." The Thirteenth International Conference on Learning Representations.
4. Bai, Yuntao, et al. "Training a helpful and harmless assistant with reinforcement learning from human feedback." arXiv preprint arXiv:2204.05862 (2022).
5. Bradley, Ralph Allan, and Milton E. Terry. "Rank analysis of incomplete block designs: I. The method of paired comparisons." Biometrika 39.3/4 (1952): 324-345.
6. Schulman, John, et al. "Proximal policy optimization algorithms." arXiv preprint arXiv:1707.06347 (2017).

Copyright 2025 NTT CORPORATION

参考文献 D

7. Wallace, Bram, et al. "Diffusion model alignment using direct preference optimization." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2024.
8. Ethayarajh, Kawin, et al. "Model alignment as prospect theoretic optimization." Forty-first International Conference on Machine Learning. 2024.
9. Heng, Alvin, and Harold Soh. "Selective amnesia: A continual learning approach to forgetting in deep generative models." Advances in Neural Information Processing Systems 36 (2023): 17170-17194.
10. Kiryo, Ryuichi, et al. "Positive-unlabeled learning with non-negative risk estimator." Advances in neural information processing systems 30 (2017).

Copyright 2025 NTT CORPORATION

まとめ・質疑応答 (10分)

Copyright 2025 NTT CORPORATION

まとめ

- 生成モデルに関する下記のトピックについて学習した
  - 生成モデルの基礎 / 深層生成モデル / 言語モデル / Instruction Tuning
  - 現在の生成AIは非常に複雑になっているので、わからなくなったときは、簡単な例でイメージしよう (例えばコインでの最尤推定など)
- 生成AIと生成モデルのギャップはモデルサイズとデータの量
  - 本講義で説明した基礎的なことを理解したら、最新の論文を読んでみよう
  - 大きなモデル・大きなデータだからこその難しさがあります
- 生成AIの発展には、人間からのフィードバックが重要
  - どのようにして効率的に Alignment を行うかが重要になると予想

Copyright 2025 NTT CORPORATION

FAQ

- 動画の生成はどのように行われるか？
  - 時間軸を導入し、前のフレームを条件付けて生成する
- AIエージェントとは？
  - 人間の介入なしに特定のタスクを実行する自律型
- 拡散言語モデルの利点は？
  - GPTの自己回帰的な生成とは異なり、一括的な生成を行うため高速
- Alignment の問題点は？
  - DPOなどは有害な出力を忘却しているのではなくバイパスしているだけと指摘されており、また生成品質の低下も見られる

Copyright 2025 NTT CORPORATION

付録E. GAN & Flow

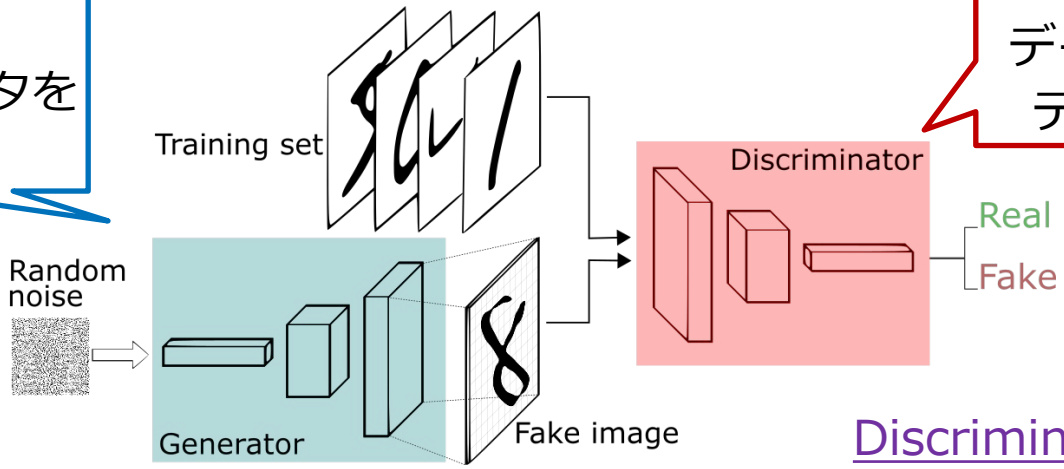
Copyright 2025 NTT CORPORATION

Generative Adversarial Network (1)

- Generative Adversarial Network (GAN) [E1] は、2つのニューラルネットワーク (Generator と Discriminator) を互いに競わせることで、生成モデルの学習を行う

Generator:
ノイズからデータを生成する

Discriminator:
データと生成されたデータを識別する



Discriminatorを騙すようにGeneratorを学習する

https://sthalles.github.io/intro-to-gans/

Copyright 2025 NTT CORPORATION

Generative Adversarial Network (2)

- 真の分布を $p^*(x)$、低次元の標準ガウス分布を $p(z)$ とする
- また、Generator を $G(z)$、Discriminator を $D(x)$ とする
- この時、G と D は下記の目的関数の最適化で学習できる
  - D に関する最大化と G に関する最小化を交互に行う

$$\min_G \max_D V(D,G) = \mathbb{E}_{p^*(x)}[\log D(x)] + \mathbb{E}_{p(z)}[\log(1 - D(G(z)))]$$

2クラス分類の目的関数

データ x と生成データ $G(z)$ を識別できるように学習

D を騙すように G を学習

Copyright 2025 NTT CORPORATION

Generative Adversarial Network (3)

- G が生成するデータの分布を $p_G(x)$ とした時、最適な $D(x)$ は下記で与えられる:

$$D^*(x) = \frac{p^*(x)}{p^*(x) + p_G(x)}$$

- この時、目的関数 $V(D^*, G)$ は、$p_G(x)$ と $p^*(x)$ のジェンセン・シャノン情報量となる

$p_G(x)$ を $p^*(x)$ に近づけるよう学習している

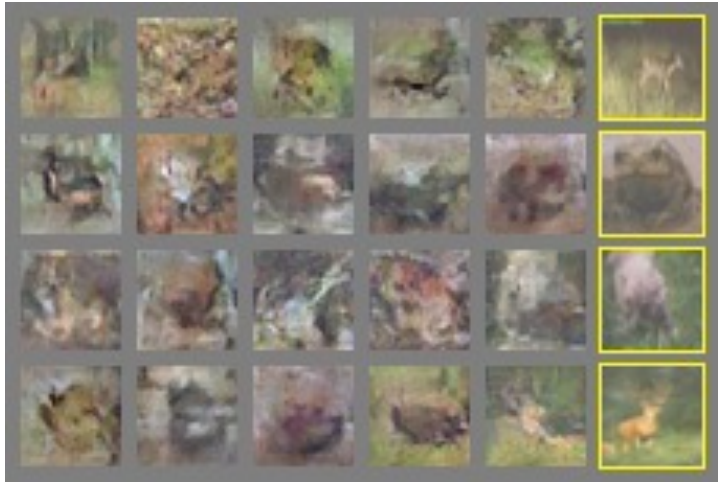
$$\mathrm{JS}(p^*(x)||p_G(x)) = \frac{1}{2}\left\{\mathrm{KL}\left(p^*(x)||m(x)\right) + \mathrm{KL}\left(p_G(x)||m(x)\right)\right\}$$

$$m(x) = \frac{p^*(x) + p_G(x)}{2}$$

Copyright 2025 NTT CORPORATION

Generative Adversarial Network (4)

GANで生成したMNISTとCIFAR10 (2014年時点) [E1]



NOTE:

- GANはデータの生成はできるが、データの確率は推定できない
- また、生成品質は高いが、学習が不安定という問題がある

Copyright 2025 NTT CORPORATION

Flow-based Model (1)

- VAEとGANにはそれぞれ下記のような問題がある
  - VAEは変分下界を計算するため、厳密な確率を計算できない
  - GANは生成品質は高いが、データの確率を計算できない
- そのため、品質が高く、厳密に確率を計算できる生成モデルの需要が高まっていた
- Flow-based Model [E2] は、データ x と同次元のガウス分布 $p(z)$ を用意し、全単射関数 $f_\theta(x)$ を用いて分布を推定する

$$z = f_\theta(x), \quad x = f_\theta^{-1}(z)$$

Copyright 2025 NTT CORPORATION

Flow-based Model (2)

- 全単射 $z = f_\theta(x)$ と $p(z)$ を用いて、確率 $p_\theta(x)$ は下記で書ける

$$\log p_\theta(x) = \log p(f_\theta(x)) + \log \underbrace{\left| \det \left(\frac{\partial f_\theta}{\partial x} \right) \right|}_{\text{ヤコビアン}}$$

(ヤコビ行列の行列式)

- ヤコビアンが計算できれば最尤推定が可能
  - 一般的な関数のヤコビアンを計算するのは難しい
  - ヤコビアンが計算しやすく、表現力の高い関数を考えるのがFlowの研究

Copyright 2025 NTT CORPORATION

Flow-based Model (3)

- 全単射の例として、下記のCoupling Layers [E2,E3] を考える

$$z_{1:d} = x_{1:d}$$

データ x を次元 d で２つに区切る

$$z_{d+1:D_x} = x_{d+1:D_x} \odot \exp(s_\theta(x_{1:d})) + t_\theta(x_{1:d})$$

$\mathbb{R}^d \to \mathbb{R}^d$ の任意NN

- この時、ヤコビ行列は下記で計算できる
  - 下三角行列となるので、行列式が簡単に計算できる

$$\begin{bmatrix} \exp(s_\theta(x_{1:d}))_1 & & 0 \\ & \ddots & \\ 0 & & \exp(s_\theta(x_{1:d}))_d \end{bmatrix}$$

$$\frac{\partial z}{\partial x} = \begin{bmatrix} \frac{\partial z_{1:d}}{\partial x_{1:d}} & \frac{\partial z_{1:d}}{\partial x_{d+1:D_x}} \\ \frac{\partial z_{d+1:D_x}}{\partial x_{1:d}} & \frac{\partial z_{d+1:D_x}}{\partial x_{d+1:D_x}} \end{bmatrix} = \begin{bmatrix} I_d & 0 \\ \frac{\partial z_{d+1:D_x}}{\partial x_{1:d}} & \mathrm{diag}(\exp(s_\theta(x_{1:d}))) \end{bmatrix}$$

この行列式の計算だけでOK

Copyright 2025 NTT CORPORATION

Flow-based Model (4)

- よって、$\log p_\theta(x)$ は下記で計算でき、最尤推定が可能

$$\begin{aligned}\log p_\theta(x) &= \log p(f_\theta(x)) + \log \left|\det\left(\mathrm{diag}(\exp(s_\theta(x_{1:d})))\right)\right| \\ &= \log p(f_\theta(x)) + \log(\exp(\mathrm{tr}(\mathrm{diag}(s_\theta(x_{1:d}))))) \\ &= \log p(f_\theta(x)) + \sum_j s_\theta(x_{1:d})_j\end{aligned}$$

$$\det(\exp(A)) = \exp(\mathrm{tr}(A))$$

- データ生成は、逆関数を用いてノイズ $z \sim p(z)$ から変換する

$$\begin{aligned}x_{1:d} &= z_{1:d} \\ x_{d+1:D_x} &= (z_{d+1:D_x} - t_\theta(x_{1:d})) \odot \exp(-s_\theta(x_{1:d}))\end{aligned}$$

Copyright 2025 NTT CORPORATION

Flow-based Model (5)

Real NVPで生成したCelebAとCIFAR10 (2017年時点) [E3]

NOTE:

- FlowはVAEよりも性能が高く、厳密に確率を求められるので注目されていた (GAN or Flow という流れだった)
- 欠点として、全単射1つだけでは表現能力が低く、多段にする必要があるため、計算量が膨大になることがあげられる

Copyright 2025 NTT CORPORATION

参考文献 E

1. Goodfellow, Ian, et al. "Generative adversarial nets." Advances in neural information processing systems 27 (2014).
2. Dinh, Laurent, David Krueger, and Yoshua Bengio. "Nice: Non-linear independent components estimation." arXiv preprint arXiv:1410.8516 (2014).
3. Dinh, Laurent, Jascha Sohl-Dickstein, and Samy Bengio. "Density estimation using real nvp." arXiv preprint arXiv:1605.08803 (2016).

Copyright 2025 NTT CORPORATION

付録F. 拡散モデルの導出

Copyright 2025 NTT CORPORATION

概要

- 拡散モデルの非常に分かりやすいレビュー論文である [F1] に従って、拡散モデルの導出を行う
  - Variational Autoencoder (VAE) や Denoising Diffusion Probabilistic Model (DDPM)、Score-Based Model の著者らと同じグループ (Google)
  - 彼らのレビューを受けており、統一的な観点でまとめられている

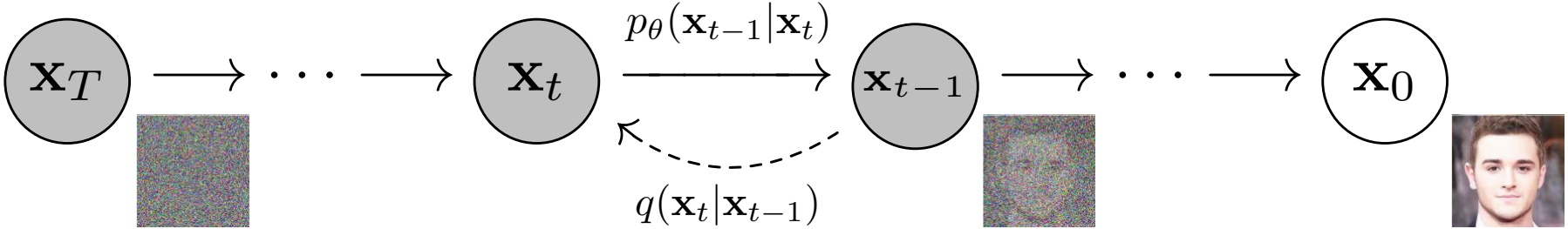
Acknowledgments: I would like to acknowledge Josh Dillon, Yang Song, Durk Kingma, Ben Poole, Jonathan Ho, Yiding Jiang, Ting Chen, Jeremy Cohen, and Chen Sun for reviewing drafts of this work and providing many helpful edits and comments. Thanks so much!

- 拡散モデルはVAEの特殊ケースで、三種類の等価な目的関数が存在する
- これらの導出を、できるだけ式変形を省略せずに説明する

Copyright 2025 NTT CORPORATION

拡散モデル (1)

- 拡散モデル [B2] は、T ステップかけてデータ $\mathbf{x}_0$ にノイズを徐々に加え、標準ガウス分布に従うノイズに変換する手法
  - ノイズを加える順方向の過程を拡散 (Diffusion) と呼ぶ
- その逆変換を行うことで、ノイズから元のデータを復元できる
  - ノイズを除去する逆方向の過程をノイズ除去 (Denoising) と呼ぶ

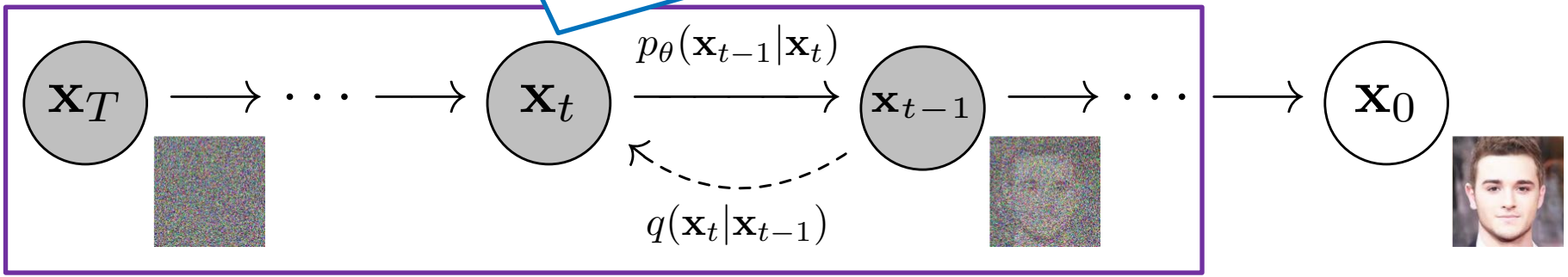


Copyright 2025 NTT CORPORATION

拡散モデル (2)

- 拡散モデルは、下記を仮定した時の階層型VAEに相当する
  - 潜在変数がデータと同次元で、マルコフ連鎖を形成
  - エンコーダ q が線形でハイパーパラメータ α_t のみに依存 (= 学習しない)
  - 最終ステップ T において、エンコーダ q が標準ガウス分布に一致

$$\mathbf{x}_t = \sqrt{\alpha_t}\mathbf{x}_{t-1} + \sqrt{1-\alpha_t}\epsilon_{t-1}, \quad \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$



$T-1$個の潜在変数とみなせる

Copyright 2025 NTT CORPORATION

拡散モデル (3)

- Reparameterization Trick [B1] を用いて、平均 μ、分散$\sigma^2\mathbf{I}$ のガウス分布からのサンプル $\mathbf{z}$ は、下記で計算できる

$$\mathbf{z} = \mu + \sigma\epsilon, \quad \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

- つまり、エンコーダ $q(\mathbf{x}_t|\mathbf{x}_{t-1})$ の分布形と時刻 t のデータ $\mathbf{x}_t$ の間には、下記の関係が成り立つ

$$\mathbf{x}_t = \sqrt{\alpha_t}\mathbf{x}_{t-1} + \sqrt{1-\alpha_t}\epsilon_{t-1} \Leftrightarrow q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{\alpha_t}\mathbf{x}_{t-1}, (1-\alpha_t)\mathbf{I})$$

Copyright 2025 NTT CORPORATION

拡散モデル (4)

- データ $\mathbf{x}_0$ の確率 $p_\theta(\mathbf{x}_0)$ を下記で推定する (θ: パラメータ)

$$p_\theta(\mathbf{x}_0) = \int p_\theta(\mathbf{x}_{0:T}) d\mathbf{x}_{1:T}$$

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

標準ガウス分布

- また、拡散されたデータの確率を下記で推定する

$$q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1}), \quad q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{\alpha_t}\mathbf{x}_{t-1}, (1-\alpha_t)\mathbf{I})$$

Copyright 2025 NTT CORPORATION

拡散モデル (5)

- この時、変分下界は下記で求められる

$$\log p_\theta(\mathbf{x}_0) = \log \int p_\theta(\mathbf{x}_{0:T}) d\mathbf{x}_{1:T}$$

$$= \log \mathbb{E}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \left[\frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \right]$$

$$\geq \mathbb{E}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \left[\log \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \right]$$

$$\equiv \mathcal{L}(\mathbf{x}_0; \theta)$$

$q(\mathbf{x}_{1:T}|\mathbf{x}_0)$ による
Importance Sampling

Jensenの不等式:
対数と和 (期待値) の交換

- この変分下界をひたすら変形し、簡略化していく

Copyright 2025 NTT CORPORATION

式変形の目的

- $p_\theta(\mathbf{x}_t|\mathbf{x}_{t+1})$ と $q(\mathbf{x}_t|\mathbf{x}_{t-1})$ は依存する時刻が異なる
  - 学習時に $\mathbf{x}_{t+1}$ と $\mathbf{x}_{t-1}$ を両方サンプリングすると、分散が大きくなる
- ベイズの定理を使って、依存する時刻を t に揃える
  - 理想的には $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$ として時刻を揃えたいが、計算が難しい

$$q(\mathbf{x}_{t-1}|\mathbf{x}_t) = \frac{q(\mathbf{x}_t|\mathbf{x}_{t-1})q(\mathbf{x}_{t-1})}{q(\mathbf{x}_t)}$$

周辺化する必要がある

  - その代わりに、計算しやすい $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$ を用いて時刻を揃える

$$q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \frac{q(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{x}_0)q(\mathbf{x}_{t-1}|\mathbf{x}_0)}{q(\mathbf{x}_t|\mathbf{x}_0)} = \frac{q(\mathbf{x}_t|\mathbf{x}_{t-1})q(\mathbf{x}_{t-1}|\mathbf{x}_0)}{q(\mathbf{x}_t|\mathbf{x}_0)}$$

ガウス分布として閉形式で求まる (後述)

Copyright 2025 NTT CORPORATION

変分下界の変形 (1)

$$
\begin{aligned}
\mathcal{L}(\mathbf{x}_0;\theta) &= \mathbb{E}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\left[\log \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\right] \\
&= \mathbb{E}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\left[\log \frac{p(\mathbf{x}_T)\prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{\prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1})}\right] \\
&= \mathbb{E}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\left[\log \frac{p(\mathbf{x}_T)p_\theta(\mathbf{x}_0|\mathbf{x}_1)\prod_{t=2}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_1|\mathbf{x}_0)\prod_{t=2}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1})}\right] \\
&= \mathbb{E}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\left[\log \frac{p(\mathbf{x}_T)p_\theta(\mathbf{x}_0|\mathbf{x}_1)\prod_{t=2}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_1|\mathbf{x}_0)\prod_{t=2}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1},\mathbf{x}_0)}\right] \\
&= \mathbb{E}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\left[\log \frac{p(\mathbf{x}_T)p_\theta(\mathbf{x}_0|\mathbf{x}_1)}{q(\mathbf{x}_1|\mathbf{x}_0)} + \log \prod_{t=2}^{T} \frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_t|\mathbf{x}_{t-1},\mathbf{x}_0)}\right]
\end{aligned}
$$

Copyright 2025 NTT CORPORATION

変分下界の変形 (2)

$$\mathcal{L}(\mathbf{x}_0;\theta) = \mathbb{E}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\left[\log\frac{p(\mathbf{x}_T)p_\theta(\mathbf{x}_0|\mathbf{x}_1)}{q(\mathbf{x}_1|\mathbf{x}_0)} + \log\prod_{t=2}^{T}\frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{\frac{q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)\cancel{q(\mathbf{x}_t|\mathbf{x}_0)}}{\cancel{q(\mathbf{x}_{t-1}|\mathbf{x}_0)}}}\right]$$

$$q(\mathbf{x}_t|\mathbf{x}_{t-1},\mathbf{x}_0) = \frac{q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)q(\mathbf{x}_t|\mathbf{x}_0)}{q(\mathbf{x}_{t-1}|\mathbf{x}_0)}$$

$$= \mathbb{E}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\left[\log\frac{p(\mathbf{x}_T)p_\theta(\mathbf{x}_0|\mathbf{x}_1)}{\cancel{q(\mathbf{x}_1|\mathbf{x}_0)}} + \log\frac{\cancel{q(\mathbf{x}_1|\mathbf{x}_0)}}{q(\mathbf{x}_T|\mathbf{x}_0)} + \log\prod_{t=2}^{T}\frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)}\right]$$

$$= \mathbb{E}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\left[\log\frac{p(\mathbf{x}_T)p_\theta(\mathbf{x}_0|\mathbf{x}_1)}{q(\mathbf{x}_T|\mathbf{x}_0)} + \log\prod_{t=2}^{T}\frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)}\right]$$

$$= \mathbb{E}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\left[\log p_\theta(\mathbf{x}_0|\mathbf{x}_1)\right] + \mathbb{E}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\left[\log\frac{p(\mathbf{x}_T)}{q(\mathbf{x}_T|\mathbf{x}_0)}\right] + \sum_{t=2}^{T}\mathbb{E}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\left[\log\frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)}\right]$$

$$= \mathbb{E}_{q(\mathbf{x}_1|\mathbf{x}_0)}\left[\log p_\theta(\mathbf{x}_0|\mathbf{x}_1)\right] + \mathbb{E}_{q(\mathbf{x}_T|\mathbf{x}_0)}\left[\log\frac{p(\mathbf{x}_T)}{q(\mathbf{x}_T|\mathbf{x}_0)}\right] + \sum_{t=2}^{T}\mathbb{E}_{q(\mathbf{x}_t,\mathbf{x}_{t-1}|\mathbf{x}_0)}\left[\log\frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)}\right]$$

$$= \underbrace{\mathbb{E}_{q(\mathbf{x}_1|\mathbf{x}_0)}\left[\log p_\theta(\mathbf{x}_0|\mathbf{x}_1)\right]}_{\text{reconstrucition term}} - \underbrace{D_{\mathrm{KL}}(q(\mathbf{x}_T|\mathbf{x}_0)||p(\mathbf{x}_T))}_{\text{prior matching term}} - \sum_{t=2}^{T}\underbrace{\mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0)}\left[D_{\mathrm{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)||p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t))\right]}_{\text{denoising matching term}}$$

Copyright 2025 NTT CORPORATION

変分下界の変形 (3)

$$\mathcal{L}(\mathbf{x}_0;\theta) = \underbrace{\mathbb{E}_{q(\mathbf{x}_1|\mathbf{x}_0)}\left[\log p_\theta(\mathbf{x}_0|\mathbf{x}_1)\right]}_{\text{reconstrucition term}} - \underbrace{D_{\mathrm{KL}}(q(\mathbf{x}_T|\mathbf{x}_0)||p(\mathbf{x}_T))}_{\text{prior matching term}} - \sum_{t=2}^{T} \underbrace{\mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0)}\left[D_{\mathrm{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)||p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t))\right]}_{\text{denoising matching term}}$$

① ② ③

1. データを再構成する項 (= VAEの再構成誤差)
   - モンテカルロ近似で学習できるが、学習時には計算しないことが多い
2. 時刻 T のデータの事後分布と標準ガウス分布のKL Divergence
   - パラメータがないため、学習時には計算しない
3. ノイズ除去ステップ $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ を学習する項
   - 明に計算できる $q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)$ に、$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ を一致させるよう学習
   - 実際に計算するのはこの項のみ

まずはこの分布を計算する

Copyright 2025 NTT CORPORATION

変分下界の変形 (4)

- 時刻 t のデータ $\mathbf{x}_t$ は、下記で計算できる

$$
\begin{aligned}
\mathbf{x}_t &= \sqrt{\alpha_t}\mathbf{x}_{t-1} + \sqrt{1-\alpha_t}\epsilon_{t-1} \\
&= \sqrt{\alpha_t}(\sqrt{\alpha_{t-1}}\mathbf{x}_{t-2} + \sqrt{1-\alpha_{t-1}}\epsilon_{t-2}) + \sqrt{1-\alpha_t}\epsilon_{t-1} \\
&= \sqrt{\alpha_t}\sqrt{\alpha_{t-1}}\mathbf{x}_{t-2} + \sqrt{\alpha_t(1-\alpha_{t-1})}\epsilon_{t-2} + \sqrt{1-\alpha_t}\epsilon_{t-1} \\
&= \sqrt{\alpha_t}\sqrt{\alpha_{t-1}}\mathbf{x}_{t-2} + \sqrt{\sqrt{\alpha_t(1-\alpha_{t-1})}^2 + \sqrt{1-\alpha_t}^2}\epsilon_{t-2} \\
&= \sqrt{\alpha_t}\sqrt{\alpha_{t-1}}\mathbf{x}_{t-2} + \sqrt{1-\alpha_t\alpha_{t-1}}\epsilon_{t-2} \\
&= \dots \\
&= \sqrt{\prod_{i=1}^{t}\alpha_i}\mathbf{x}_0 + \sqrt{1-\prod_{i=1}^{t}\alpha_i}\epsilon_0 \\
&= \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon_0
\end{aligned}
$$

ガウス分布に従う乱数の和

$$
\begin{aligned}
\mathbf{z}_{t-2} &= \sqrt{\alpha_t(1-\alpha_{t-1})}\epsilon_{t-2} \sim \mathcal{N}(\mathbf{0}, \alpha_t(1-\alpha_{t-1})\mathbf{I}) \\
\mathbf{z}_{t-1} &= \sqrt{1-\alpha_t}\epsilon_{t-1} \sim \mathcal{N}(\mathbf{0}, (1-\alpha_t)\mathbf{I}) \\
\mathbf{z}_{t-2} + \mathbf{z}_{t-1} &= \sqrt{\alpha_t(1-\alpha_{t-1})}\epsilon_{t-2} + \sqrt{1-\alpha_t}\epsilon_{t-1} \\
&\sim \mathcal{N}(\mathbf{0}, \alpha_t(1-\alpha_{t-1})\mathbf{I} + (1-\alpha_t)\mathbf{I}) \\
&= \mathcal{N}(\mathbf{0}, (1-\alpha_t\alpha_{t-1})\mathbf{I})
\end{aligned}
$$

Copyright 2025 NTT CORPORATION

変分下界の変形 (5)

- 従って、時刻 t のデータの事後分布は下記で表される

$$\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon_0 \Leftrightarrow q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I})$$

Reparameterization Trick [B1] を用いて、平均 μ、分散$\sigma^2\mathbf{I}$ のガウス分布からのサンプル $\mathbf{z}$ は、下記で計算できる

$$\mathbf{z} = \mu + \sigma\epsilon, \quad \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

Copyright 2025 NTT CORPORATION

変分下界の変形 (6)

- よって、$q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$ は下記で計算できる

マルコフ連鎖のため
$q(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{x}_0) = q(\mathbf{x}_t|\mathbf{x}_{t-1})$

$$
\begin{aligned}
q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) &= \frac{q(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{x}_0)q(\mathbf{x}_{t-1}|\mathbf{x}_0)}{q(\mathbf{x}_t|\mathbf{x}_0)} \\
&= \frac{\mathcal{N}(\mathbf{x}_t; \sqrt{\alpha_t}\mathbf{x}_{t-1}, (1-\alpha_t)\mathbf{I})\mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_{t-1}}\mathbf{x}_0, (1-\bar{\alpha}_{t-1})\mathbf{I})}{\mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I})} \\
&= \mathcal{N}(\mathbf{x}_{t-1}; \mu_q(\mathbf{x}_t, \mathbf{x}_0), \sigma_q^2(t)\mathbf{I}) \\
\mu_q(\mathbf{x}_t, \mathbf{x}_0) &= \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x}_t + \frac{\sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)}{1-\bar{\alpha}_t}\mathbf{x}_0 \\
\sigma_q^2(t) &= \frac{(1-\alpha_t)(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}
\end{aligned}
$$

Copyright 2025 NTT CORPORATION

変分下界の変形 (7)

- ノイズ除去ステップ $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ を下記で定義する

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \sigma_q^2(t)\mathbf{I})$$

ガウス分布の平均を推定するNN

- この時、目的関数は下記で計算できる

$$\begin{aligned}
&\arg\min_\theta D_{\mathrm{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)||p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)) \\
&= \arg\min_\theta D_{\mathrm{KL}}(\mathcal{N}(\mathbf{x}_{t-1}; \mu_q(\mathbf{x}_t, \mathbf{x}_0), \sigma_q^2(t)\mathbf{I})||\mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \sigma_q^2(t)\mathbf{I})) \\
&= \arg\min \frac{1}{2\sigma_q^2(t)} \left[\|\mu_\theta(\mathbf{x}_t, t) - \mu_q(\mathbf{x}_t, \mathbf{x}_0)\|^2 \right]
\end{aligned}$$

Copyright 2025 NTT CORPORATION

変分下界の変形 (8)

- $\mu_q(\mathbf{x}_t, \mathbf{x}_0)$ は下記で定義される (再掲)

$$\mu_q(\mathbf{x}_t, \mathbf{x}_0) = \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x}_t + \frac{\sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)}{1-\bar{\alpha}_t}\mathbf{x}_0$$

- 従って、 $\mu_\theta(\mathbf{x}_t, t)$ を下記で定義する

$$\mu_\theta(\mathbf{x}_t, t) = \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x}_t + \frac{\sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)}{1-\bar{\alpha}_t}\hat{\mathbf{x}}_\theta(\mathbf{x}_t, t)$$

時刻 t のデータ $\mathbf{x}_t$ から元のデータ $\mathbf{x}_0$ を推定するNN

Copyright 2025 NTT CORPORATION

変分下界の変形 (9)

- μ_q と μ_θ を用いて、目的関数は下記のように変形できる
  - つまり、時刻 t のデータ $\mathbf{x}_t$ から元のデータ $\mathbf{x}_0$ を推定するNNを学習することができれば、拡散モデルを学習できる

$$
\begin{aligned}
&\arg\min_\theta D_{\mathrm{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)||p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)) \\
&= \arg\min \frac{1}{2\sigma_q^2(t)}\left[\|\mu_\theta(\mathbf{x}_t,t)-\mu_q(\mathbf{x}_t,\mathbf{x}_0)\|^2\right] \\
&= \arg\min \frac{1}{2\sigma_q^2(t)}\left[\left\|\cancel{\frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x}_t} + \frac{\sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)}{1-\bar{\alpha}_t}\hat{\mathbf{x}}_\theta(\mathbf{x}_t,t) - \cancel{\frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x}_t} - \frac{\sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)}{1-\bar{\alpha}_t}\mathbf{x}_0\right\|^2\right] \\
&= \arg\min \frac{1}{2\sigma_q^2(t)}\frac{\bar{\alpha}_{t-1}(1-\alpha_t)^2}{(1-\bar{\alpha}_t)^2}\boxed{\left[\|\hat{\mathbf{x}}_\theta(\mathbf{x}_t,t)-\mathbf{x}_0\|^2\right]}
\end{aligned}
$$

Copyright 2025 NTT CORPORATION

三つの等価な変形

- 拡散モデルの目的関数は、拡散されたデータから、元のデータを推定するよう学習されることを導出した
- この目的関数が、下記のノイズの推定、スコアの推定と等価となることを示す

元のデータの推定: $\left\|\hat{\mathbf{x}}_\theta(\mathbf{x}_t, t) - \mathbf{x}_0\right\|^2$

ノイズの推定: $\left\|\epsilon_0 - \hat{\epsilon}_\theta(\mathbf{x}_t, t)\right\|^2$

スコアの推定: $\left\|s_\theta(\mathbf{x}, t) - \nabla \log p(\mathbf{x}_t)\right\|^2$

Copyright 2025 NTT CORPORATION

ノイズの推定 (1)

- 元のデータ $\mathbf{x}_0$ は、時刻 t のデータ $\mathbf{x}_t$ を用いて下記で表される

$$\mathbf{x}_0 = \frac{\mathbf{x}_t - \sqrt{1-\bar{\alpha}_t}\epsilon_0}{\sqrt{\bar{\alpha}_t}}$$

- 従って、 $\mu_q(\mathbf{x}_t, \mathbf{x}_0)$ は下記で変形できる

$$\begin{aligned}
\mu_q(\mathbf{x}_t, \mathbf{x}_0) &= \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x}_t + \frac{\sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)}{1-\bar{\alpha}_t}\mathbf{x}_0 \\
&= \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x}_t + \frac{\sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)}{1-\bar{\alpha}_t}\frac{\mathbf{x}_t - \sqrt{1-\bar{\alpha}_t}\epsilon_0}{\sqrt{\bar{\alpha}_t}} \\
&= \frac{1}{\sqrt{\alpha_t}}\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}\sqrt{\alpha_t}}\epsilon_0
\end{aligned}$$

Copyright 2025 NTT CORPORATION

ノイズの推定 (2)

- $\mu_q(\mathbf{x}_t, \mathbf{x}_0)$ は下記で定義される (再掲)

$$\mu_q(\mathbf{x}_t, \mathbf{x}_0) = \frac{1}{\sqrt{\alpha_t}}\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}\sqrt{\alpha_t}}\epsilon_0$$

- 従って、 $\mu_\theta(\mathbf{x}_t, t)$ を下記で定義する

$$\mu_\theta(\mathbf{x}_t, t) = \frac{1}{\sqrt{\alpha_t}}\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}\sqrt{\alpha_t}}\hat{\epsilon}_\theta(\mathbf{x}_t, t)$$

時刻 t のデータ $\mathbf{x}_t$ から元のノイズ ϵ_0 を推定するNN

Copyright 2025 NTT CORPORATION

ノイズの推定 (3)

- μ_q と μ_θ を用いて、目的関数は下記のように変形できる
  - つまり、時刻 t のデータ $\mathbf{x}_t$ から元のノイズ ϵ_0 を推定するNNを学習することができれば、拡散モデルを学習できる

$$\arg\min_\theta D_{\mathrm{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)||p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t))$$

$$= \arg\min \frac{1}{2\sigma_q^2(t)}\left[\|\mu_\theta(\mathbf{x}_t,t)-\mu_q(\mathbf{x}_t,\mathbf{x}_0)\|^2\right]$$

$$= \arg\min \frac{1}{2\sigma_q^2(t)}\left[\left\|\cancel{\frac{1}{\sqrt{\alpha_t}}\mathbf{x}_t}-\frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}\sqrt{\alpha_t}}\hat{\epsilon}_\theta(\mathbf{x}_t,t)-\cancel{\frac{1}{\sqrt{\alpha_t}}\mathbf{x}_t}+\frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}\sqrt{\alpha_t}}\epsilon_0\right\|^2\right]$$

$$= \arg\min \frac{1}{2\sigma_q^2(t)}\frac{(1-\alpha_t)^2}{(1-\bar{\alpha}_t)\alpha_t}\left[\|\epsilon_0-\hat{\epsilon}_\theta(\mathbf{x}_t,t)\|^2\right]$$

Copyright 2025 NTT CORPORATION

ノイズの推定 (4)

- この目的関数を簡略化したものが、DDPM [B2] の目的関数
  - 実験的には、簡略化した目的関数でノイズを推定するのが最も良いらしい

$$-\mathcal{L}(\mathbf{x}_0;\theta) \propto \mathbb{E}_{\mathcal{U}(t),p(\epsilon_0)}\left[\left\|\epsilon_0 - \hat{\epsilon}_\theta\left(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon_0, t\right)\right\|^2\right]$$

Algorithm 1 Training

1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \text{Uniform}(\{1,\dots,T\})$
4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0},\mathbf{I})$
5: Take gradient descent step on
$$\nabla_\theta \left\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t)\right\|^2$$
6: **until** converged

Copyright 2025 NTT CORPORATION

ノイズの推定 (5)

- この場合、データ $\mathbf{x}_0$ の生成は下記の手順で行われる

Algorithm 2 Sampling

1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** $t = T, \ldots, 1$ **do**
3: $\quad \mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
4: $\quad \mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$
5: **end for**
6: **return** $\mathbf{x}_0$

ガウス分布からノイズ $\mathbf{x}_T$ を生成

T ステップかけてノイズを除去

Copyright 2025 NTT CORPORATION

スコアの推定 (1)

- Tweedieの公式 [F2] より、ガウス分布について下記が成立

$$\mathbf{z} \sim \mathcal{N}(\mathbf{z}; \mu_{\mathbf{z}}, \Sigma_{\mathbf{z}}) \Rightarrow \underline{\mathbb{E}\left[\mu_{\mathbf{z}} | \mathbf{z}\right] = \mathbf{z} + \Sigma_{\mathbf{z}} \nabla \log p(\mathbf{z})}$$

ガウス分布の平均の期待値を、
サンプルと分散と微分で表すことができる

- 従って、時刻 t のデータ $\mathbf{x}_t$ の事後分布について、下記が成立

$$q(\mathbf{x}_t | \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1 - \bar{\alpha}_t)\mathbf{I})$$

$$\Downarrow$$

$$\sqrt{\bar{\alpha}_t}\mathbf{x}_0 = \mathbf{x}_t + (1 - \bar{\alpha}_t) \nabla \log p(\mathbf{x}_t)$$

Copyright 2025 NTT CORPORATION

スコアの推定 (2)

- 元のデータ $\mathbf{x}_0$ は、時刻 t のデータ $\mathbf{x}_t$ を用いて下記で表される

$$\mathbf{x}_0 = \frac{\mathbf{x}_t + (1-\bar{\alpha}_t)\nabla \log p(\mathbf{x}_t)}{\sqrt{\bar{\alpha}_t}}$$

Tweedieの公式から導出

- 従って、 $\mu_q(\mathbf{x}_t, \mathbf{x}_0)$ は下記で変形できる

$$\begin{aligned}
\mu_q(\mathbf{x}_t, \mathbf{x}_0) &= \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x}_t + \frac{\sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)}{1-\bar{\alpha}_t}\mathbf{x}_0 \\
&= \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x}_t + \frac{\sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)}{1-\bar{\alpha}_t}\frac{\mathbf{x}_t + (1-\bar{\alpha}_t)\nabla \log p(\mathbf{x}_t)}{\sqrt{\bar{\alpha}_t}} \\
&= \frac{1}{\sqrt{\alpha_t}}\mathbf{x}_t + \frac{1-\alpha_t}{\sqrt{\alpha_t}}\nabla \log p(\mathbf{x}_t)
\end{aligned}$$

Copyright 2025 NTT CORPORATION

スコアの推定 (3)

- $\mu_q(\mathbf{x}_t, \mathbf{x}_0)$ は下記で定義される (再掲)

$$\mu_q(\mathbf{x}_t, \mathbf{x}_0) = \frac{1}{\sqrt{\alpha_t}}\mathbf{x}_t + \frac{1-\alpha_t}{\sqrt{\alpha_t}}\nabla \log p(\mathbf{x}_t)$$

- 従って、 $\mu_\theta(\mathbf{x}_t, t)$ を下記で定義する

$$\mu_\theta(\mathbf{x}_t, t) = \frac{1}{\sqrt{\alpha_t}}\mathbf{x}_t + \frac{1-\alpha_t}{\sqrt{\alpha_t}}s_\theta(\mathbf{x}_t, t)$$

時刻 t のデータ $\mathbf{x}_t$ からスコア $\nabla \log p(\mathbf{x}_t)$ を推定するNN

Copyright 2025 NTT CORPORATION

スコアの推定 (4)

- μ_q と μ_θ を用いて、目的関数は下記のように変形できる
  - つまり、時刻 t のデータ $\mathbf{x}_t$ から、スコア $\nabla \log p(\mathbf{x}_t)$ を推定するNNを学習できれば、拡散モデルを学習できる (Score-Based Model [F3] と一致)

$$
\begin{aligned}
&\arg\min_\theta D_{\mathrm{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)||p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)) \\
&= \arg\min \frac{1}{2\sigma_q^2(t)}\left[\|\mu_\theta(\mathbf{x}_t,t)-\mu_q(\mathbf{x}_t,\mathbf{x}_0)\|^2\right] \\
&= \arg\min \frac{1}{2\sigma_q^2(t)}\left[\left\|\cancel{\frac{1}{\sqrt{\alpha_t}}\mathbf{x}_t}+\frac{1-\alpha_t}{\sqrt{\alpha_t}}s_\theta(\mathbf{x}_t,t)-\cancel{\frac{1}{\sqrt{\alpha_t}}\mathbf{x}_t}-\frac{1-\alpha_t}{\sqrt{\alpha_t}}\nabla\log p(\mathbf{x}_t)\right\|^2\right] \\
&= \arg\min \frac{1}{2\sigma_q^2(t)}\frac{(1-\alpha_t)^2}{\alpha_t}\boxed{\left[\|s_\theta(\mathbf{x}_t,t)-\nabla\log p(\mathbf{x}_t)\|^2\right]}
\end{aligned}
$$

Copyright 2025 NTT CORPORATION

スコアの推定 (5)

- スコア $\nabla \log p(\mathbf{x}_t)$ は、ノイズ ϵ_0 を用いて計算できる

$$
\begin{aligned}
\mathbf{x}_0 &= \frac{\mathbf{x}_t + (1-\bar{\alpha}_t)\nabla \log p(\mathbf{x}_t)}{\sqrt{\bar{\alpha}_t}} \\
&= \frac{\mathbf{x}_t - \sqrt{1-\bar{\alpha}_t}\epsilon_0}{\sqrt{\bar{\alpha}_t}} \\
\nabla \log p(\mathbf{x}_t) &= -\frac{1}{\sqrt{1-\bar{\alpha}_t}}\epsilon_0
\end{aligned}
$$

Copyright 2025 NTT CORPORATION

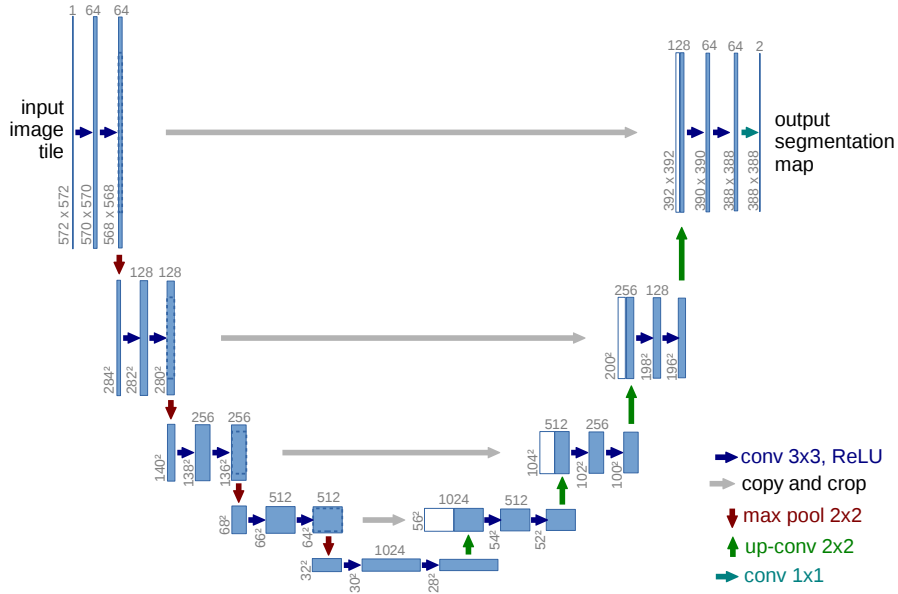
まとめ

- 拡散モデルは階層VAEの特殊ケース
  - 潜在変数がデータと同次元で、マルコフ連鎖
  - エンコーダが線形で、ハイパーパラメータ α_t のみに依存 (= 学習しない)
  - 最終ステップ T において、エンコーダが標準ガウス分布に一致
- 目的関数は、下記の三通りのNNを学習する形となり、全て等価
  - 時刻 t のデータ $\mathbf{x}_t$ から、元のデータ $\mathbf{x}_0$ を予測するNN
  - 時刻 t のデータ $\mathbf{x}_t$ から、元のノイズ ϵ_0 を予測するNN
  - 時刻 t のデータ $\mathbf{x}_t$ から、スコア関数 $\nabla \log p(\mathbf{x}_t)$ を予測するNN
- サンプリングに時間がかかるのが欠点 (T回推定するため)

Copyright 2025 NTT CORPORATION

補足: 実際のネットワーク構造

- $\hat{\epsilon}_\theta(\mathbf{x}_t, t)$ は、U-Net [F4] を用いてモデル化している
  - Image Segmentation で標準的に用いられている構造
  - 時刻 t の埋め込みには、positional embedding [C1] などを用いる



階層モデルだけど
1つのU-NetでOK

Copyright 2025 NTT CORPORATION

補足: ハイパーパラメータの推定

- 目的関数は下記のように変形できる

$$\frac{1}{2\sigma_q^2(t)}\frac{\bar{\alpha}_{t-1}(1-\alpha_t)^2}{(1-\bar{\alpha}_t)^2}\left[\|\hat{\mathbf{x}}_\theta(\mathbf{x}_t,t)-\mathbf{x}_0\|^2\right]=\frac{1}{2}\left(\frac{\bar{\alpha}_{t-1}}{1-\bar{\alpha}_{t-1}}-\frac{\bar{\alpha}_t}{1-\bar{\alpha}_t}\right)\left[\|\hat{\mathbf{x}}_\theta(\mathbf{x}_t,t)-\mathbf{x}_0\|^2\right]$$

$$\sigma_q^2(t)=\frac{(1-\alpha_t)(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}$$

$$=\frac{1}{2}\left(\mathrm{SNR}(t-1)-\mathrm{SNR}(t)\right)\left[\|\hat{\mathbf{x}}_\theta(\mathbf{x}_t,t)-\mathbf{x}_0\|^2\right]$$

Signal-to-Noise Ratio: 単調増加

- 従って、単調増加のNN $\omega_\eta(t)$ を用いて下記でモデル化できる

$$\mathrm{SNR}(t)=\frac{\bar{\alpha}_t}{1-\bar{\alpha}_t}=\exp(-\omega_\eta(t))$$

$$\bar{\alpha}_t=\mathrm{sigmoid}(-\omega_\eta(t))$$
$$1-\bar{\alpha}_t=\mathrm{sigmoid}(\omega_\eta(t))$$

Copyright 2025 NTT CORPORATION

参考文献 F

1. Luo, Calvin. "Understanding diffusion models: A unified perspective." arXiv preprint arXiv:2208.11970 (2022).
2. Efron, Bradley. "Tweedie’s formula and selection bias." Journal of the American Statistical Association 106.496 (2011): 1602-1614.
3. Song, Yang, and Stefano Ermon. "Generative modeling by estimating gradients of the data distribution." Advances in neural information processing systems 32 (2019).
4. Ronneberger, Olaf, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation." Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18. Springer International Publishing, 2015.

Copyright 2025 NTT CORPORATION