Diffusion Models

Himanshu Singh

Github

Introduction

$x \in \Re^n$

Probability Density Function $P(x): \Re^n \rightarrow \Re^+$

- $0 \leq P(x)$
- $P(x)dx \leq 1$: Probability of x landing in infitesimal region dx
- $\int P(x)dx = 1$

For two random vector $x_1 \in \Re^m$ and $x_2 \in \Re^m$, their joint PDF is $P(x_1, x_2)$

Marginal Distribution is given by $\int P(x_1, x_2)dx_2 = P(x_1)$

Conditional Probability $P(x_1|x_2) = P(x_1, x_2)/P(x_2)$

Bayes Rule $P(x_1|x_2) = \dfrac{P(x_2|x_1)P(x_1)}{P(x_2)}$

Statistical Independence $P(x_1, x_2) = P(x_1)P(x_2)$

Chain Rule $P(x_1, x_2, \ldots, x_n) = P(x_1)P(x2|x_1)P(x_3|x_1, x_2)\ldots P(x_n|x_1, x2, \ldots, x_{n-1})$

If the chain is Markovian, it implies that each vector is statistically dependent only on its predecessor. $P(x_1, x_2, \ldots, x_n) = P(x_1)P(x2|x_1)P(x_3|x_2)\ldots P(x_n|x_{n-1})$

Consider two independent random variables $x1 \in \Re^n$ and $x_2 \in \Re^m$, where $x_1 \sim P(x_1)$ and $x_2 \sim Q(x_2)$.

Let $z = x_1 + x_2$. Then the PDF of z, $G(z) = P(z) \otimes Q(z)$ i.e. convolution

The Expectation of arbitrary deterministic expression f(x) over random vector $x \sim P(x)$ is given by

$$E_x[f(x)] = \int f(x)P(x)dx$$

Properties:

- Linearity $E_x[f(x) + g(x)] = E_x[f(x)] + E_x[g(x)]$
- Mean and Covariance $\mu = E_x[x]$ and $\Sigma_x = E_x[(x-\mu)(x-\mu)^T]$
- If x_1 and x_2 are uncorrelated iff $E_x[x_1^T x_2] = E_x[x_1]^T E_x[x_2]$
- If x_1 and x_2 are independent then necessarily $E_x[x_1^T x_2] = E_x[x_1]^T E_x[x_2]$

Conditional Expectation $E_x[x|z] = \int xP(x|z)dx = \int x\dfrac{P(x,z)}{P(z)}dx$

Law of total expectation (smoothing property) $E_z[E_x[x|z]] = E_x[x]$

Multivariate Gaussian PDF $x \in \Re^n$

$$P(x) = \mathbb{N}(x; \mu, \Sigma) = \sqrt{\frac{1}{(2\pi)^n |\Sigma|}} \exp\left[-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\right]$$

Two uncorrelated Gaussian random vectors are necessarily independent.

If $x \sim \mathbb{N}(x; \mu, \Sigma), z = Ax + b$, then $z \sim \mathbb{N}(z; A\mu; A\Sigma A^T)$

If $x_1 \sim \mathbb{N}(x_1; \mu_1, \Sigma_1)$ and $x_2 \sim \mathbb{N}(x_2; \mu_2, \Sigma_2)$ are of same dimension and uncorrelated, then.

$$x_1 + x_2 = z \sim \mathbb{N}(z; \mu_1 + \mu_2, \Sigma_1 + \Sigma_2)$$

Axiom: The information carried by an instance $x = k$ is $\log \dfrac{1}{p(x=k)}$

Intuition: The higher the probability of x, the less is its "uncertainty" and this its information is smaller as well, log comes because information in two independent events should be their sum.

The Entoppy of the source is defined by expected information. $0 \leq H(x) \leq \log_2 C$

$$H(x) = E_x\left[\log_2 \frac{1}{p(x)}\right] = \sum_{k-1}^{C} p(x=k)\log_2 \frac{1}{p(x=k)}$$

The notion of Entropy can be extended to continuos random variables. The Differential Entropy of random variable $x \sim P(x)$ if given y expected information

$$H(x) = E_x\left[\log \frac{1}{P(x)}\right] = \int P(x) \log \frac{1}{P(x)} dx$$

Note: This may assume negative values.

Kullback-Leibler (KL) divergence offers an asymmetric "distance" measure between two distribution.

$$KL(P \| Q) = E_{x \sim P}\left[\log \frac{P(x)}{Q(x)}\right] = \int P(x) \log \frac{P(x)}{Q(x)} dx$$

Properties:

- $KL(P\|Q) \geq 0$ and $KL(P\|Q) \neq KL(Q\|P)$
- If $P(x) = Q(x)$, iff $KL(P\|Q) = 0$
- If $P(x) = \mathbb{N}(x_P; \mu_P, \Sigma_P)$ and $Q(x) = \mathbb{N}(x_Q; \mu_Q, \Sigma_Q)$ then $KL(P\|Q) \propto (\mu_p - \mu_q)^T \Sigma_q^{-1} (\mu_p - \mu_q) + \log \frac{|\Sigma_p|}{|\Sigma_q|} + tr(\Sigma_Q^{-1} \Sigma_p) - n$

Mutual information quantifies how dependent two random vectors are

$$I(x;z) = \int P(x,z) \log \frac{P(x,z)}{P(x)P(z)} dx = KL(P(x,z) \| P(x)Q(x))$$

Properties:

- $x, z \in \Re^n$ are independent iff $I(x;z) = 0$
- $I(x;z) = I(z;x)$ is symmetric function
- $0 \leq I(x;z) \leq min(H(x), H(z))$

Wasserstein's distance optimal transport/ Earth-Moving Distance

$$W[P(x), Q(x)] = \inf_{G(x,z)} \int_x \int_z \|x - z\|_2^2 G(x,z) dx dz$$

where $P(x) = \int_z G(x,z) dz$ and $Q(z) = \int_x G(x,z) dx$

For two Gaussian, $W[\mathbb{N}(x; \mu_p \Sigma_p), \mathbb{N}(x; \mu_q . \Sigma_q)] = \|\mu_p - \mu_q\|_2^2 + tr(\Sigma_Q + \Sigma_p - 2(\Sigma_p \Sigma_q)^{0.5})$

Inverse Problem:

$$\left.\begin{array}{l} \bullet \text{ De-Noising} \\ \bullet \text{ De-Blurring} \\ \bullet \text{ In-Painting} \\ \bullet \text{ De-Mosaicing} \\ \bullet \text{ Super Resolution} \end{array}\right\} y = Cx_0 + n$$

In general Inverse problem, the measurement is $y = Cx_0 + v, \|v\|_2 \leq \epsilon$. Goal is to recover x0 from y. C is general linear degradation operator (Blur, Projection, Downscaling, Subsampling, Holes).

$$\hat{x} = \arg\max_x P(x) \qquad s.t. \|y - Cx\|_2 \leq \epsilon$$

Let $v \sim \mathbb{N}(0, \sigma^2 I)$ Bayesian approach suggest to work with P(x|y) for solving the problem.

MAP estimator $\hat{x} = \arg\max_x P(x|y) = \arg\max_x \frac{P(y|x)P(x)}{P(y)}$

$$\arg\max_x P(x|y) = \arg\max_x \frac{P(y|x)P(x)}{P(y)}$$

$$\arg\max_x P(x|y) = \arg\min_x -\log(y|x) - \log P(x)$$

From the y and x relationship, we know P(y|x) is shifter Gaussian.

$$-\log P(y|x) = -\log\left[c\,.\exp\left\{-\frac{1}{2}\|y - Cx\|_2^2\right\}\right]$$

$$-\log P(y|x) \approx \frac{1}{2\sigma^2}\|y - Cx\|_2^2$$

$$\hat{x} = \underset{x}{arg\,min}\,\frac{1}{2\sigma^2}\|y - Cx\|_2^2 - \log P(x)$$

Minimum Mean Square Error Estimate $\hat{x} = E[x|y] = \int_x x\,P(x|y)\,dx$

The estimate leads to the best MSE by averaging over all possible solutions that explain y, thus it is expected to be somewhat blurry.

Various Image Synthesis Methods:

- Variational AutoEncoder (VAE)
- Generative Adversarial Networks (GAN)
- Normalizing Flow Methods
- Recursive Methods
- Energy based Methods (EBM)
- Diffusion Models

As Deep Learning era evolved, the desire of learning P(x) got shifted. The focus shifted on learning networks which can produce synthetic images. The ability to sample fairly from P(x) via learned machine $\hat{x} = G_\theta(z)$

Variational Autoencoder

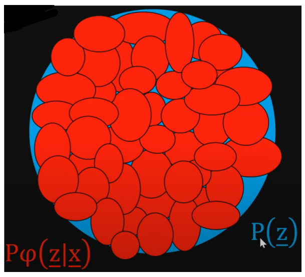
Training AutoEncoder amounts to training encoder Φ and decoder Θ, so as to best reproduce our training set images, while forcing a bottleneck of representation z

$$\text{Loss}(\Phi, \Theta) = \sum_{k=1}^{N} dist\left[D_\Theta(E_\Phi(x_k)), \tilde{x}_k\right]$$

Naive thought: Once, trained decoder can serve as generator, but this approach does not work because latent space is unorganised. It contains holes, and has no “probabilistic interpretation”

Intuitive formulation of VAE:

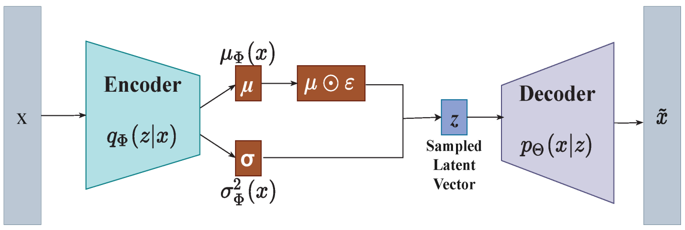
- Use original AE loss, add a regularization that forces latent space to be “well behaved”
- Add randomness to synthesis process, the encoder output serves as parameters for random sampler that feeds the decoder.



Assume $P(z) = \mathbb{N}(0, I)$. Assume $q_\Phi(z|x) = \mathbb{N}(m_\Phi(x), diag(s_\Phi^2(x)))$ i.e. for every image x, a smaller covariance $s_\Phi^2(x)$ defines portion of representation z. Our goal would be to create a good match between all these ellipsoid and latent Gaussian. This force well-structure of latent domain, with continuity and smoothness.

Given incoming image x, encoder output $m_\Phi(x)$ & $s_\Phi(x)$ is used to draw $z = m_\Phi + s_\Phi . v$ where $v \sim \mathbb{N}(0, I)$

Our goal is to get smallest recovery error as regular AE, and good match of P(z) to sum of all ellipsoid $P_\Phi(z|x)$ corresponding to



all training x. This is achieved by approximation of KL divergence between P(z) and $P_\Phi(z)$
VAE is not competitive tool because:

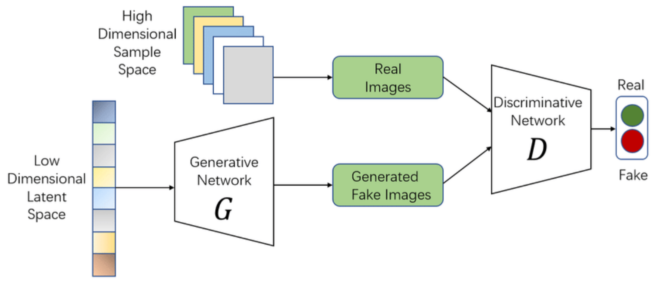
- Recovery loss is simply wrong! If the output of decoder should align with x for every latent sample z, we are getting blurry images.
- The parametric assumption over $P_\Phi(z|x)$ is quite limited
- The KL approximation used is not sufficient to bring delicate results desired.

Generative Adversarial Networks

Train both Generator and Discriminator

$$\max_\theta \min_\phi \text{Loss}(\theta,\phi)$$

$$\text{Loss}(\theta,\phi)=\sum_{k=1}^{N}[1-D_\phi(x_k)]+\sum_{k=1}^{N'}D_\phi(G_\theta(z_k))$$
$$=E_{x\sim P_{true}}[1-D_\phi(x)]+E_{x\sim P_{fake}}[D_\phi(x)]$$



Idea:

- Freezing generator, the discriminator aims to minimize this function, pushing $D_\phi(x_k)\rightarrow 1$ and $D_\phi(G_\phi(z_k))\rightarrow 0$, thus being able to separate true from fakes images.
- Freezing Discriminator, the generator tries to maximize the expression, thus pushing $D_\phi(G_\theta(z_k))\rightarrow 1$ aiming to confuse the discriminator.

Lets consider the non ideal case(non-parameterized functions) and find optimal discriminator

$$E_{x\sim P_{true}}[1-D_\phi(x)]+E_{x\sim P_{fake}}[D_\phi(x)]=\int\{[1-D(x)]P_{true}(x)+D(x)P_{fake}(x)\}dx$$

Minimizing this , will give, $D_{opt}(x)=\begin{cases}0, & if\ P_{true}(x)<P_{fake}(x)\\ 1, & if\ P_{true}(x)\geq P_{fake}(x)\end{cases}$

With choice of D(x), our loss becomes

$\text{Loss}=\int min\{P_{true}(x),P_{fake}(x)\}dx$

Maximizing the above wrt Generator leads to $\forall x, P_{fake}(x)\geq P_{true}(x)$, but it must sum to 1. Hence Max = $P_{true}(X)=P_{true}(x)$

An Alternative and more familiar loss uses log function (reversed min-max)

$\text{Loss}=\int\{\log[1-D(x)]P_{fake}(x)+\log D(x)P_{true}(x)\}dx$

$$\frac{-P_{fake}(x)}{1-D_{opt}(x)}+\frac{P_{true}(x)}{D_{opt}(x)}=0\Rightarrow D_{opt}=\frac{P_{true}(x)}{P_{true}(x)+P_{fake}(x)}$$

With this choice of maximizing $D_{opt}(x)$, our loss become

$$\text{Loss}=\int\left\{\log\frac{P_{fake}(x)}{P_{true}(x)+P_{fake}(x)}P_{fake}(x)+\frac{P_{true}(x)}{P_{true}(x)+P_{fake}(x)}P_{true}(x)\right\}dx$$
$$\propto KL\left\{P_{true}(x),\frac{1}{2}[P_{true}(x)+P_{fake}(x)]\right\}+KL\left\{P_{fake}(x),\frac{1}{2}[P_{true}(x)+P_{fake}(x)]\right\}$$

The minima of expression is when $P_{true}(X)=P_{true}(x)$

Flow Methods

x and z are two random vectors with distribution $P_X(x)$ and $P_Z(z)$ and $x=f(z)$, where f is invertible $z=f^{-1}(x)$. x and z are of same dimension $J(z)=\frac{\partial f(z)}{\partial z}$ is $n\times n$ Jacobian Matrix.

$|det\, J(z)|=1$ implies Volume preserving relation, Then

$$P_X(x) = P_Z(f^{-1}(x)) \left| det\left(\frac{\partial f^{-1}(x)}{\partial x}\right) \right|$$
$$= P_Z(z) \left| det\left(\frac{\partial f(z)}{\partial z}\right) \right|$$

Normalizing Flow Method: Design an invertible function $G_\theta(z)$, so as to get a clear relation between $P_Z(z)$ and $P_X(x)$ We need x and z of same dimension 'n' and making sure $|det\, J(z)|$ is easily computable.
Goal: Start from $P_Z(z) = \mathbb{N}[0, I]$ and transform it via $G_\theta(z)$to $P_X(x)$
Idea: Maximize likelihood of training set.

$$\max_\theta P_X(x_1, x_2, \ldots, x_N) = \max_\theta P_X(x_1) P_X(x_2) \ldots P_X(x_N)$$
$$= \max_\theta \sum_{k=1}^{N} \log[P_X(x_k)]$$
$$= \max_\theta \sum_{k=1}^{N} \log\left[P_Z(f^{-1}(x_k)) . \left| det\left(\frac{\partial f^{-1}(x)}{\partial x}\right)_{x_k} \right| \right]$$

By maximizing this function wrt θ, where $G_\theta(z) = f(z)$, we make our training examples more probable, thus identifying flow function $G_\theta(z)$. We can enrich training by creating noisy version of training example and pushing them to have small likelihood.
$G_\theta(z)$ is built as chain of feed-forward steps, the determinant is a multiplication of individual step
Cons: Due to inconvertibility, limited architectures are present and tractable Jacobian poses severe limitation.
Pros: Easier to train than GANs, inverse operation, can evaluate probability of an image.

Synthesis Quality Evaluation

Observing images for the richness and visual quality is poor approach because:

- It is not quantifiable
- Cannot truly assess richness, some regions of image manifold may be compromised without us being able to identify it (Mode collapse)
- We cannot truly access visual quality, some intricate details might be missing.

Alternatives:

- Inception Score
- Fréchet Inception Distance
- Precision and Recall
- Negative Log Likelihood (NLL)

Inception Score

$$\left.\begin{array}{l}\text{Do synthesized images classifiable distinctly}\\ \text{Do synthesized images vary in their classification identity}\end{array}\right\}\text{Inception Classifier}$$

Pass each generated image $\hat{x}_k (1 \le k \le N = 50000)$ through classifier and get output label probability vector $y_k \in \Re^C (\mathbf{1}^T y_k = 1, y_k \ge 0)$
1. Distinct classification implies low entropy
2. Varying identities means mean probability vector $\bar{y} = (\sum_k y_k)/N$is nearly uniform thus having high entropy

$$\text{Entropy}(y)=\sum_{i=1}^{C} y_i \log_2 \frac{1}{y_{i0}} \leq Entropy(y) \leq \log_2 C$$

Summing both points into

$$\begin{aligned} N\log(IS) &= N\bar{y}^T \log\frac{1}{\bar{y}} - \sum_{k=1}^{N} y_k^T \log\frac{1}{y_k} \\ &= \sum_{k=1}^{N} y_k^T \log\frac{1}{\bar{y}} - \sum_{k=1}^{N} y_k^T \log\frac{1}{y_k} \\ &= \sum_{k=1}^{N} y_k^T \log\frac{y_k}{\bar{y}} \\ &= \sum_{k=1}^{N} KL(y_k, \bar{y}) \end{aligned}$$

$$IS=\exp\left\{\frac{1}{N}\sum_{k=1}^{N} KL(y_k,\bar{y})\right\}$$ The Bigger the better

Limitation:

- Limited to labelled data
- Tied to Inception and ImageNet
- Memorization is graded high
- No notion of comparison between true and fake images

Fréchet Inception Distance

As with IS, FID relies on the Inception Classifier, but takes an inner activation layer which has 2048 elements.

Take N images $\{x_k\}_{k=1}^N$ from validation set and produce their feature $\{z_k\}_{k=1}^N$. Construct 2048-dimensional corresponding Gaussian $\mathbb{N}(\mu_1, \Sigma_1)$. Repeat same for N synthesized images $\{\hat{x}_k\}_{k=1}^N$ and theirs corresponding Gaussian $\mathbb{N}(\mu_2, \Sigma_2)$. FID is Fréchet (2-Wasserstein's) distance between two Gaussian. The smaller the value, the better the match

$$\begin{aligned} FID(\hat{P}_{true}(x), \hat{P}_{fake}(x)) &= W_2(\mathbb{N}(\mu_1,\Sigma_1), \mathbb{N}(\mu_2,\Sigma_2)) \\ &= \|\mu_1-\mu_2\|^2 + tr(\Sigma_1+\Sigma_2-2(\Sigma_1\Sigma_2)^{0.5}) \end{aligned}$$

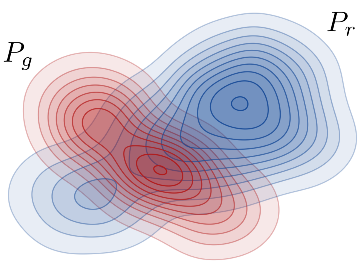
FID compares true and fakes images thus is more adequate for the task of evaluating the synthesis quality. It can be applied to unlabelled images.

FID is far from perfect, and recent work has shown it can be easily attacked.

Precision and Recall

Precision: A ratio between intersection area and whole generated support. P = 0.7 ⇒ 30% generated image are wrong.

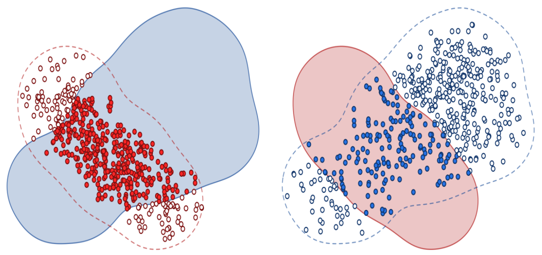
Recall: A ratio between intersection area and whole real Support. R = 0.6 ⇒ 40% of true images cannot be synthesized.



We do this by approximating via sphere in low dimension. Start by reducing dimensionality by taking same feature that were used in FID (Dim = 2048).

Take fakes images (N = 50000) and assign radius of sphere to each of them so as to contain the K(=5) nearest neighbour. Per each of N true images, see which falls within these fake sphere, the relative number of positive ones is the precision.

The same reversed applies to recall evaluation.


(b) Precision (c) Recall

Negative Log likelihood (NLL)
If generated model can provide the values P(x), we could calculate the term -logP(x) for all x in test set.

$$NLL = \frac{-1}{Nn} \sum_{k=1}^{N} \log P(x_k) \qquad [\text{bits}]$$

The better the generative model, the smaller the NLL.

Intoduction to Diffusion

Classical Image processing had desire to get P(x), then it moved to alternative of sampling from P(x) via a learned machine $\hat{x}=G_\theta(z)$.

Now researcher introduced **Score function** $\nabla_x \log P(x)$

In Inverse problem, we were trying to recover x_0 from y, where they are related via $y=Cx_0+\mu$, where $\mu \sim \mathbb{N}(0,\sigma^2 I)$. Bayesian approach that led to optimization task:

$$\hat{x}=\underset{x}{argmin}\frac{1}{2\sigma^2}\|y-Cx\|_2^2-\log P(x)$$

Steepest descent for minimizing this function, we get

$$\hat{x}_{k+1}=\hat{x}_k+\lambda\left[\frac{1}{\sigma^2}C^T(y-Cx)-\nabla_x \log P(x)\Big|_{\hat{x}_k}\right]$$

If we access the score function, we could solve (MAP-wise) various inverse problem without need for prior. Also in Gibbs form $P(x)=\frac{1}{Z}\exp[-\rho(x)] \quad \Rightarrow \quad \nabla_x \log P(x)=-\nabla_x \rho(x)$, here the score function skips the need to calculate the partition function, which is a source of difficulties.

Assume Noisy Image $y=x+\nu$ where $\nu \sim \mathbb{N}(o,\sigma^2 I)$ and $x \sim P(x)$, then

$$P(y)=P(x)\otimes \mathbb{N}(0,\sigma^2 I)=P_\sigma(x)$$

$$P(y)=\int_x P(y|x)P(x)dx=\left(\frac{1}{2\pi\sigma^2}\right)^{n/2}\int_x \exp\left\{\frac{-1}{2\sigma^2}\|y-x\|_2^2\right\}P(x)dx$$

$$\nabla_y P(y)=\frac{1}{\sigma^2}\left(\frac{1}{2\pi\sigma^2}\right)^{n/2}\int_x (x-y)\exp\left\{\frac{-1}{2\sigma^2}\|y-x\|_2^2\right\}P(x)dx$$

$$\nabla_y P(y)=\frac{1}{\sigma^2}\int_x (x-y)P(y|x)P(x)dx$$

Dividing by $P(y)$ and multiplying by σ^2

$$\sigma^2\frac{\nabla_y P(y)}{P(y)}=\int_x (x-y)\frac{P(y|x)P(x)}{P(y)}dx$$

$$\sigma^2\nabla_y \log P(y)\equiv\sigma^2\frac{\nabla_y P(y)}{P(y)}=\int_x (x-y)P(x|y)$$

$$\sigma^2\nabla_y \log P(y)=\int_x xP(x|y)dx-y\int_x P(x|y)dx$$

$$\sigma^2\nabla_y \log P(y)=E[x|y]-y$$

$E[x|y]$ is MSE denoiser of image y

The score function calculated is for noisy images y, how can we get $\nabla_x \log P(x)$? By assuming a very weak noise($\sigma \sim 0.001-0.01$). Apply your (MMSE) denoiser on y and subtract this image, i.e. provide the estimated noise from the denoiser.

Image Denoiser

Removal of White Additive Gaussian Noise from Image.

Why Gaussian? Gaussian case is more common and much more important.

When considering Poisson noise,

- high count of photons – The distribution gets closer and closer to Gaussian case.
- Low count Poisson distributed images can be converted to a Gaussian-noisy one by Anscombe-Variance Stabilizing Transform.

Recently with evolution of Deep learning, we have architectures which predicts the denoised images by seeing multiple images and learning from them $\hat{x}=D_\theta(y,\sigma)$. Relevant architectures: RNN, UNet, Transformers & classic-oriented structures. This implies the approximating the score function is possible and easy $\nabla_y \log P(y)=\frac{1}{\sigma^2}\left[D_\theta(y,\sigma)-y\right]$
Being an MMSE denoiser implies $\hat{x}=D_\theta(y,\sigma)=E[x|y]$, should satisfy conditions like Symmetrical Jacobian, Passivity, ...

RED and PnP

Recall Inverse Problem $y=Cx_0+v$, where $v\sim\mathbb{N}(0,\sigma^2 I)$. Bayesian Approach led the optimization task $\hat{x}=\arg\min_x \frac{1}{2¿}\|y-Cx\|_2^2-\log P(x)$, with steepest descent we have

$$\hat{x}_{k+1}=\hat{x}_k+\lambda\left[\frac{1}{\sigma^2}C^T(y-C\hat{x}_k)-\nabla_x \log P(x)\Big|_{\hat{x}_k}\right]$$

Score function can be approximated by denoiser $\sigma^2\nabla_y \log P(y)=E[x|y]-y=D(y,\sigma)-y$. This recovering x_0 given y via MAP amounts for

$$\begin{aligned}\hat{x}_{k+1}&=\hat{x}_k+\lambda\left[\frac{1}{\sigma^2}C^T(y-C\hat{x}_k)-\nabla_x \log P_{\sigma_0}(x)\Big|_{\hat{x}_k}\right]\\ &=\hat{x}_k+\lambda\left[\frac{1}{\sigma^2}C^T(y-C\hat{x}_k)-\frac{1}{\sigma_0^2}(D(\hat{x},\sigma_0)-\hat{x}_k)\right]\end{aligned}$$

Plug and Play (PnP)

$\hat{x}=\min_x \frac{1}{2}\|Cx-y\|^2+\rho(x)$

Split the unknown

$\hat{x}=\min_{x,v} \frac{1}{2}\|Cx-y\|^2+\rho(v) \quad s..t.x=v$

Turn constraint into penalty

$\hat{x}=\min_{x,v} \frac{1}{2}\|Cx-y\|^2+\rho(v)+\beta\|x-v\|^2$

Solve Alternating between x and v

Least Square $\hat{x}=\min_x \frac{1}{2}\|Cx-y\|^2+\beta\|x-v\|^2$

Denoiser $\hat{v}=\min_v \rho(v)+\beta\|x-v\|^2$

Regularization by Denoiser (RED)

RED introduced a very specific regularization term in optimization task:

$\hat{x}=\min_x \frac{1}{2}\|Cx-y\|^2+\rho(x)=\min_x \frac{1}{2}\|Cx-y\|^2+\lambda x^T[x-D(x,\sigma)]$

Now taking Derivate of $D(x,\sigma)$ is pretty tough, if x being 1k$\times$1k image $\nabla_x D(x,\sigma)$ will be 1M$\times$1M. But, under conditions like Differentiability local homogeneity, passivity and symmetrical Jacobian gradient of regularization term becomes $[x-D(x,\sigma)]$

$$\hat{x}_{k+1}=\hat{x}_k-\mu\left[C^T(Cx-y)+\lambda[\hat{x}_k-D(\hat{x}_k,\sigma)]\right]$$

<u>Theorem</u>: If denoiser $D(x,\sigma)$ is differentiable, and satisfies the properties:

1. Local Homogeneity $D(\alpha x,\sigma)=\alpha D(x,\sigma)$ for $|\alpha-1|<\epsilon$
2. Strong Passivity $\rho(\nabla_x D(x,\sigma))<1$

3. Symmetrical Jacobian $\nabla_x D(x,\sigma)=[\nabla_x D(x,\sigma)]^T$

Then this energy function is strictly convex $E(x)=x^T[x-D(x,\sigma)] \Rightarrow \nabla_x E(x)=2[x-D(x,\sigma)]$

Proof:

$$\begin{aligned}\nabla_x x^T[x-D(x,\sigma)] &= \nabla_x x^T x-\nabla_x x^T D(x,\sigma)\\ &= 2x-D(x,\sigma)-[\nabla_x D(x,\sigma)]^T x\\ &\overset{3}{=} 2x-D(x,\sigma)-\nabla_x D(x,\sigma)x\end{aligned}$$

Directional Derivative $\nabla_x D(x)x=\lim_{\epsilon} \frac{D(x+\epsilon x)-D(x)}{\epsilon} \overset{1}{=} \lim_{\epsilon} \frac{(1+\epsilon)D(x)-D(x)}{\epsilon}=D(x)$

$\nabla_x x^T[x-D(x,\sigma)]=2x-2D(x,\sigma)=2[x-D(x,\sigma)]$

Hessian $\nabla_x^2 E(x) = 2[I-\nabla_x D(x,\sigma)] \overset{2}{>} 0$

Langevin Dynamics

Given a well behaved probability density function P(x), we aim to draw iid samples from it. Langevin Stochastic Differential approach $x_t=\underbrace{\nabla_x \log P(x)}_{\text{Good Score function}}+\sqrt{2}w_t$, x_t is temporal derivative of continuos time-varying vector x(t), w_t is temporal derivative of Brownian motion w(t), which is accumulation of canonical Gaussian white noise (Weiner Process, random walk). If initialized with $x(0)\sim\mathbb{N}(0,I)$ the process produces at infinity fair samples from P(x) i.e. their empirical distribution over many such runs is P(x)

Discretizing above equation: $x_k=x(k\tau)$ and $w_k=w(k\tau)$ and τ is very small

$$\frac{1}{\tau}[x_{k+1}-x_k]=\nabla_x \log P(x_k)+\frac{\sqrt{2}}{\tau}[w_{k+1}-w_k]$$
$$x_{k+1}=x_k+\tau\nabla_x \log P(x_k)+\sqrt{2\tau}z_k$$

$w_{k+1}-w_k\sim\mathbb{N}(0,\tau I)$ thus $z_k\sim\mathbb{N}(0,I)$

As we have access to score function via denoiser approximation, we can implement following algorithm and get samples from target probability distribution.

```
Initialization $x_0\sim\mathbb{N}(0,I), \sigma_*=0.01$
for k = 0: K-1
    $x_{k+1}=x_k+\tau\nabla_x \log P(x_k)+\sqrt{2}z_k$
           $=x_k+\frac{\tau}{\sigma^2}\left(D(x_k,\sigma_*)-x_k\right)+\sqrt{2\tau}z_k$
end
```

The above algorithm will work, as long as

- The value of %tau is small
- Score approximation is of high quality
- If we apply lot of iterations

Problem 1: With images, these vector are very high dimensional (n = 10^6), while their manifold is of low dimension. Thus most of embedding space – the cube $[0,1]^n$ is empty. The probability at this point is $P(x)\approx 0$, the log becomes $-\infty$, so gradient is either undefined or zero

Problem 2: In lower probability region of manifold, we are not likely to have sufficient examples to train on, approximation of score function in these region becomes poor. Iterating in these region behave inconsistently.

Theoretical Foundations: Let ϵ be a target precision level, n the dimension of vector generated, τ the LD step, K the number of iterations, m and M the smallest and largest eigenvalues of Hessian of convex energy function -logP(x). Then, for $\alpha>1$, we get $|P_{true}(x)-P_{LD}(x)|\le\epsilon$, for

$$K \geq \frac{4\log\left(\frac{1}{\epsilon}\right)+n\log\left(\frac{M}{m}\right)}{2m} \qquad \tau < \frac{\epsilon}{M}\sqrt{\frac{2\alpha-1}{Kp\alpha}}$$

Flow of PDF as function of time: The Fokker-Planck PDE(1D)

$$\frac{\partial p(x,t)}{\partial t} = \frac{\partial}{\partial x}\left[\frac{\partial^2 \log P_T(x)}{\partial x^2} p(x,t)\right] + \frac{\partial^2 p(x,t)}{\partial x^2}$$

Image Synthesis via Denoiser

Annealing: Instead of working with P(x), work with a blurry version of it $P(x)\otimes\mathbb{N}(0,\sigma^2 I)$ for large value of $\sigma . y=x+v$, where $v\sim\mathbb{N}(0,\sigma^2 I)$ and $x\sim P(x)$

$$P(y)=\int_x P(y|x)P(x)dx=\left(\frac{1}{2\pi\sigma^2}\right)^{n/2}\int_x \exp\left[\frac{-1}{2\sigma^2}\|y-x\|_2^2\right]P(x)dx$$
$$=P(x)\otimes\mathbb{N}(0,\sigma^2 I)\triangleq P_\sigma(x)$$

Annealing Langevin Dynamics :

Use a sequence of L (= 500) decreasing noise levels $(\infty\approx)\sigma_0>\sigma_1>...>\sigma_L\approx 0.01$

We start with very blurred manifold $P_{\sigma_0}(x)$ from which sampling is easy and reliable, apply K(=5) iterations and move to $P_{\sigma_1}(x)$, and we stop at σ_L

```
Initialization: $x_0\sim\mathbb{N}(0,I)(\text{or } U[0,1]^n)$
for j = 0 : L
    for k = 0: K-1
        $x_{k+1}=x_k+\tau\nabla_x \log P_{\sigma_j}(x_k)+\sqrt{2\tau}z_k$
    end
    $x_0=x_K$
end
```

Note: $\nabla_x \log P_{\%signa_j}(x)=\frac{1}{\sigma_j^2}(D(x,\sigma_j)-x)$

No need to use the same τ throughout the algorithm, large τ can be use initially and then we decrease.

An alternative to above approach is to take $y=(1-\alpha)x+\alpha v$ where $v\sim\mathbb{N}(0,I)$. In many papers, the mixing of equation is $y=\sqrt{1-\alpha}\,x+\sqrt{\alpha}\,v$ in order to preserve energy

$y=(1-\alpha)x+\alpha v$ where $v\sim\mathbb{N}(0,I)$

$$P(y)=\int_x P(y|x)P(x)dx=\left(\frac{1}{2\pi\alpha^2}\right)^{n/2}\int_x \exp\left[\frac{-1}{2\alpha^2}\|y-(1-\%lapha)x\|_2^2\right]P(x)dx$$

Let $z=(1-\alpha)x$

$$P(y)=\left(\frac{1}{2\pi\alpha^2}\right)^{n/2}\int_z \exp\left[\frac{-1}{2\alpha^2}\|y-z\|_2^2\right]\frac{1}{1-\alpha}P\left(\frac{z}{1-\alpha}\right)dz$$
$$=\frac{1}{1-\alpha}P\left(\frac{x}{1-\alpha}\right)\otimes\mathbb{N}(0,\alpha^2 I)\triangleq P_\alpha(x)$$

The denoiser used the below algorithm removes the noise from the image of the form $\tilde{x}=(1-\alpha_j)x+\alpha_j v$, where $v\sim\mathbb{N}(0,I)$ producing an estimate of x.

```
Algorithm: $1=\alpha_0>\alpha_1>\alpha_2>\ldots>\alpha_L\approx 0.01$
Initialization: j = 0 : L
    for k = 0: K-1
        $x_{k+1}=x_k+\tau\nabla_x \log P_{\alpha_j}(x_k)+\sqrt{2\tau}\,z_k$
              $=x_k+\frac{\tau}{\alpha_j^2}\left((1-\alpha_j)D^*(x_k,\alpha_j)-x_k\right)+\sqrt{2\tau}\,z_k$
    end
    x_0 = x_K
end
```

Lecture 4 00:00