# Diffusion Tutorial

compiled by D.Gueorguiev, 6/17/2024

## Introductory Notes

The goal of generative modeling is: given iid samples from unknown distribution , construct a sampler for approximately the same distribution.

*Example*: given a training set of dog images from some underlying distribution , we want a method of producing new images of dogs from this distribution.

One way to solve this problem is to learn a transformation from some easy-to-sample distribution (such as Gaussian noise) to our target distribution . Diffusion models offer a general framework for learning such transformations. We want to reduce the problem of sampling from distribution to a sequence of easier sampling problems.

## Gaussian Diffusion

Let be a random variable in distributed according to the target distribution . Then construct a sequence of r.v.’s by successively adding independent Gaussian noise with some small scale :

, (1)

(1) represents forward process, which transforms the data distribution into a noise distribution. Thus (1) defines a joint distribution over , and we let denote the marginal distributions of each . Notice that at large step count , the distribution is nearly Gaussian, so we can approximate sample from by just sampling a Gaussian.

Now, suppose we can solve the following subproblem –

Given a sample marginally distributed as , produce a sample marginally distributed as .

We will call a method that does this a *reverse sampler*, since it tells us how to sample from assuming we can already sample from . If we had a reverse sampler, we could sample from our target by simply starting with a Gaussian sample from , and iteratively applying the reverse sampling procedure to get samples from , and finally .

The key insight of diffusion is, learning to reverse each step can be easier than learning to sample from target distribution in one step. There are many ways to construct reverse samplers, but for concreteness let us first see the standard diffusion sampler which we will call the *Denoising Diffusion Probabilistic Model* (DDPM) sampler.

The *ideal DDPM sampler* uses an obvious strategy: at time , given input (a sample from ), we output a sample from the conditional distribution

(2)

(2) represents a reverse sample. The problem is, it requires learning a generative model for the conditional distribution for every , which could be complicated. But if the per-step noise is sufficiently small, then it turns out this conditional distribution becomes simple:

Property of Diffusion Reverse Process

For small , and the Gaussian diffusion process defined in (1), the conditional distribution is itself close to Gaussian. That is, for all times and conditionings , there exists some mean parameter such that

(3)

For a given time and conditioning value , learning the mean of is sufficient to learn the full conditional distribution . This not obvious fact enables a drastic simplification – instead of having to learn an arbitrary distribution from scratch, we now know everything about this distribution except its mean, which we denote .

Learning the mean of is a much simpler problem than learning the conditional distribution itself; we can solve it by regression. We have a joint distribution from which we can sample and want to estimate . This is done by minimization of the standard regression loss

(4)

(5)

(6)

where the expectation is taken over samples from our target distribution . We simulate samples of by adding noise to the samples of as defined by (1).

When the target is a distribution on images, then the corresponding regression problem (6) is an *image denoising objective*, which can be approached with CNNs.

### Constructing Diffusion-like Generative Models

Let us now abstract away the Gaussian setting, to define diffusion-like models in a way that will capture their many instantiations (including deterministic samples, discrete domains, and flow-matching).

We start with the target distribution , and we pick some base distribution which is easy to sample from, e.g. a standard Gaussian or i.i.d. bits. We then try to construct a sequence of distributions which interpolate between the target and the base distribution . That is, we construct distributions

(7)

such that is our target, is the base distribution, and adjacent distributions are close enough in some well-defined sense. Then we learn a reverse sampler which transforms to . Formally,

**Definition** (*Reverse Sampler*)

Given a sequence of marginal distributions , a reverse sampler for step t is a potentially stochastic function such that if , then the marginal distribution of is exactly .

(8)

There are many possible reverse samplers; some samplers can be deterministic.

We will consider three possible reverse samplers : the *DDPM sampler*, the *DDIM sampler,* which is deterministic and the family of *flow-matching models* which can be thought as generalization of *DDIM*.

### Discretization

Let us elaborate what it means adjacent distributions to be close. The sequence can be thought of as the discretization of some (well-behaved) time-evolving function , that starts from the target distribution at time and ends at the noisy distribution at time :

, where (9)

The number of steps controls the fineness of the discretization and hence the closeness of adjacent distributions.

In order to ensure that the variance of the final distribution, , is independent of the number of discretization steps, we also need to be more specific about the variance of each increment.

Note that if , then . Therefore, we need to scale the variance of each increment by , that is, choose

(10)

where is the desired terminal variance. The choice (10) ensures that the variance of is always , regardless of .

Notation Adjustment

From here on, will represent a continuous value in the interval ; it will be taking over the values . Subscripts will indicate *time* rather than *index*, so for example will now denote at a discretized time . That is, (1) becomes:

, (11)

which also implies that

, where (12)

since the total noise added up to time (i.e. ) is also Gaussian with mean zero and variance .

## Stochastic Sampling: DDPM

We will review DDPM-like reverse sampler and heuristically prove its correctness.

This sampler is similar as the sampler discussed in [3] and was originally introduced in [1]. The main difference with [3] is that we use the “Variance Exploding” diffusion forward process. We also use constant noise schedule and we do not discuss how to parametrize the predictor (predicting vs vs noise ).

Let us consider the same setup as before – a target distribution and a joint distribution of noisy samples defined in (11). The DDPM sampler will require estimates of the following conditional expectations:

(13)

This is a set of functions , one for every time step . In *the training phase*, we estimate these functions from i.i.d. samples of , by optimizing the denoising regression objective

(14)

Typically a neural network is used to parametrize . In practice, it is common to share parameters when learning the different regression functions ,

Then, in *the inference phase*, we use the estimated functions in the following reverse sampler.

Algorithm 1: Stochastic Reverse Sampler (DDPM-like)

For input sample , and timestep , output:

(15)

To actually generate a sample, say , we first sample as an isotropic Gaussian , and then run the iteration of Algorithm 1 down to , to produce a generated sample . Here , as our discretized notation (12) indicates, is the noise-only terminal distribution, and the iteration takes steps of size .

Question: why does iterating Algorithm 1 produce a sample from (approximately) the target distribution ? The key missing piece is, we need to prove the property of the diffusion processes given with (3) – that is , the true conditional can be approximated by a Gaussian and this approximation converges to the true conditional distribution if we discretize in smaller steps .

Here we continue with the discussion on the property of the diffusion processes-

Statement 1: Let be an arbitrary, sufficiently smooth density over . Consider the joint distribution of where and . Then, for sufficiently small , the following holds: for all conditionings , there exists such that

(16)

for some constant depending only on . Moreover, it suffices to take

(17)

(18)

where is the marginal distribution of .

One can recognize that the second term in (18) is the scaled by variance statistical score (aka informant) ([13], Appendix 1).

Tweedie’s formula ([14], [15], Appendix 2) implies that this mean is exactly correct even for large , with no approximation required. However, the distribution may deviate from Gaussian for larger ’s.

### Appendix

### Statistical Score / Informant

Score (aka Informant) is the gradient of the log-likelihood function with respect to the parameter vector . Evaluated at a particular point of the parameter vector, the score indicates the steepness of the log-likelihood function and thereby the sensitivity to infinitesimal changes to the parameter values.

Linear score

The likelihood of an observation is given by a density of the form

//TODO: finish the section on Statistical Score

### Exponential Family of Statistical Distributions

The Exponential Family is a parametric set of Probability Distributions of a certain form.

Most of the commonly used distributions form an exponential family or subset of an exponential family, are listed below:

Normal, exponential, gamma, chi-squared, beta, Dirichlet, Bernoulli, categorical, Poisson, Wishart, Geometric.

A number of common distributions are exponential families but only when certain parameters are fixed and known. For example:

Binomial (with fixed number of trials), multinomial (with fixed number of trials), negative binomial (with fixed number of failures).

Examples of common distributions that are not exponential families are Student t, most mixture distributions and even the family of the uniform distributions when the bounds are not fixed.

Requirements for Exponential Family of distributions

**Scalar parameter**

The value of is called the *parameter* of the family.

//TODO: finish the section on the Exponential Family

### Generalized Linear Model

GLM is a flexible generalization of ordinary linear regression. The GLM generalizes linear regression by allowing the linear model to be related to the response variable via a *link function* and by allowing the magnitude of the variance of each measurement to be a function of its predicted value.

#### Informal motivation for GLM

Ordinary linear regression predicts the expected value of a given unknown quantity (the *response variable* or *random variable*) as a linear combination of a set of observed values (*predictors*). This implies that a constant change in a predictor leads to a constant change in the response variable which constitutes *linear-response model*. This model is appropriate when the response variable can vary indefinitely in either direction. Note that the response variable under this model only varies by a relatively small amount compared to the variation of the predictive variables.

However, these assumptions are inappropriate for some types of response variables. For example in cases where the response variable is expected to be always positive and varying over a wide range, constant input changes lead to geometrically (i.e. exponentially) varying, rather than constantly varying output changes.

Scenario 1:

Suppose a linear prediction model learns from some data (perhaps primarily drawn from large beaches) that a 10-degree temperature decrease would lead to 1,000 fewer people visiting the beach. This model is unlikely to generalize well over different sized beaches. More specifically, the problem is that if one uses the model to predict the new attendance with a temperature drop of 10 for a beach that regularly receives 50 beachgoers, one would predict an impossible attendance value of -950. Logically, a more realistic model would instead predict a *constant rate* of increased beach attendance (e.g. an increase of 10 degrees leads to a doubling in beach attendance, and a drop of 10 degrees leads to a halving in attendance). Such a model is known as *exponential-response model* (or *log-linear model* since the logarithm in the response is predicted to vary linearly).

Scenario 2:

Suppose one needs a model that predicts a probability of making yes/no choice (a Bernoulli variable) in which probabilities are bounded on both ends (they must be between 0 and 1). Let this model predict the likelihood of a given person going to the beach as a function of the temperature. A reasonable model might predict that, for example, a change in 10 degrees makes a person two times more or less likely to go to the beach. But what does “twice as likely” means in terms of probability? It cannot literally mean to double the probability value (e.g. 50% becomes 100%, 75% becomes 150%, etc). Rather, it is the odds that are doubling: from 2:1 odds, to 4:1 odds, to 8:1 odds, etc. Such a model is referred to as a *log-odds* or *logistic model*.

GLMs cover these scenarios by allowing i) response variables that have arbitrary distributions (rather than simply normal distributions), and ii) an arbitrary function of the response variable (the *link function*) to vary linearly with the predictors (rather than assuming that the response itself must vary linearly). For example, the Scenario 1 above of predicted number of beach attendees would typically be modeled with a Poisson distribution and a *log link*, while the Scenario 2 of predicted probability of beach attendance would typically be modelled with Bernoulli distribution or binomial distribution and a *log-odds (logit) link* function.

#### More Formal Overview of GLM

In GLM each outcome of the dependent variables is assumed to be generated from a distribution in the exponential family [17] i.e. Normal, Binomial, Poisson, Gamma, etc.

//TODO: finish the section on Generalized Linear Model

### Poisson Regression

//TODO: finish the section on Poisson Regression

### Tweedie’s Formula

Let us assume that has been sampled from a prior density and then is observed, where is known:

and (A2.1)

Let denote the marginal distribution of ,

where (A2.2)

Tweedie’s formula calculates the posterior expectation of given as

where (A2.3)

All of the observations can be used to obtain a smooth estimate of , yielding

(A2.4)

(A2.4) represents an estimate which corrects for the selection bias.

Here is an example which we will analyze for evidence that indeed such correction occurs.

Let us suppose there is some large number of possibly correlated normal variates each with its own unobserved mean parameter

for (A2.5)

We are looking into the 100 largest ’s. *Selection bias* is the tendency of the corresponding 100 ’s to be less extreme, that is to lie closer to the center of the observed distribution, an example of regression to the mean. Figure 1 shows a data set with independent values obeying (A2.5) with . The largest ’s are indicated by blue dashes.

Question: How can we undo the effects of selection bias and estimate the corresponding values?

A green and blue graph

Description automatically generated

Figure 1: independent values obeying (A2.5) with . The largest ’s are indicated by blue dashes.

A graph of a curve

Description automatically generated

Figure 2: Empirical Bayes estimation curve . Blue dashes indicate the 100 largest ’s and their corresponding . Small green dots show the actual Bayes estimation curve.

Robbins ([15]) presents Tweedie’s formula as an [exponential family](https://en.wikipedia.org/wiki/Exponential_family) generalization of (A2.1)

and (A2.6)

Here is the natural or canonical parameter of the family, the [cumulant generating function](https://www.statlect.com/fundamentals-of-probability/cumulant-generating-function) or (which makes integrate to 1), and the density when .

The choice where is given with (A2.2) i.e. , yields the normal translation family , with . In this case .

Bayes rule provides the posterior density of given ,

(A2.7)

where is the marginal density

(A2.8)

is the sample space of the exponential family. Then (A2.6) gives

where (A2.9)

(A2.9) represents an exponential family with canonical parameter and .

Proof of (A2.9):

Recall ([16]) that of is defined as:

where

Therefore

Substituting (A2.6) into the last equation leads to

. Substituting (A2.8) in the latter leads to . Thus

Differentiation of yields the posterior cumulants of given ,

, var (A2.10)

Let’s compute the first derivative of :

. (A2.11)

, (A2.12)

(A2.13)

(A2.14)

(A2.14) in (A2.13) leads to

(A2.15)

(A2.15) and (A2.12) in (A2.11) leads to

(A2.16)

Since the last two terms cancel out and using (A2.7) becomes

References

[1] [Deep Unsupervised Learning Using Nonequilibrium Thermodynamics, Jascha Sohl-Dickstein et al, Stanford U., 2015](https://github.com/dimitarpg13/deep_learning_for_image_processing/blob/main/literature/articles/generative_models/Deep_Unsupervised_Learning_using_Nonequilibrium_Thermodynamics_Sohl-Dickstein_2015.pdf)

[2] [Step-By-Step Diffusion: An Elementary Tutorial, P. Nakkiran et al, 2024](https://github.com/dimitarpg13/deep_learning_for_image_processing/blob/main/literature/articles/generative_models/StepByStepDiffusionAnElementaryTutorial_Nakkiran_2024.pdf)

[3] [Denoising Diffusion Probabilistic Models, J. Ho et al, UC Berkeley, 2020](https://github.com/dimitarpg13/deep_learning_for_image_processing/blob/main/literature/articles/generative_models/Denoising_Diffusion_Probabilistic_Models_Ho_UCBerkeley_2020.pdf)

[4] [Introduction to Flow Matching, Tor Fjelde, Emile Mathieu, Vincent Dutordoir, 2024 (online blog)](https://mlg.eng.cam.ac.uk/blog/2024/01/20/flow-matching.html)

[5] [Building Diffusion Model's theory from ground up, Ayan Das, ICRL blogposts, 2024](https://iclr-blogposts.github.io/2024/blog/diffusion-theory-from-scratch/)

[6] [Perspectives on Diffusion, Sander Dieleman, 2023](https://sander.ai/2023/07/20/perspectives.html)

[7] [Introduction to Diffusion Models for Deep Learning, Ryan O'Connor, 2022 (online blog)](https://www.assemblyai.com/blog/diffusion-models-for-machine-learning-introduction/)

[8] [Tutorial on Diffusion Models for Imaging and Vision, Stanley Chan, 2024](https://github.com/dimitarpg13/deep_learning_for_image_processing/blob/main/literature/articles/generative_models/Tutorial_on_Diffusion_Models_for_Imaging_and_Vision_Chan_2024.pdf)

[9] [Understanding Diffusion Models: Unified Perspective, Calvin Luo, Google Brain, 2022](https://github.com/dimitarpg13/deep_learning_for_image_processing/blob/main/literature/articles/generative_models/Understanding_Diffusion_Models-A_Unified_Perspective_Luo_GoogleBrain_2022.pdf)

[10] [Sampling, Diffusion, and Stochastic Localization, Andrea Montanari, 2023](https://github.com/dimitarpg13/deep_learning_for_image_processing/blob/main/literature/articles/generative_models/Sampling_diffusions_and_stochastic_localization_Montanari_2023.pdf)

[11] [Demystifying Variational Diffusion Models, Fabio De Sousa Ribeiro et al, Imperial College, 2024](https://github.com/dimitarpg13/deep_learning_for_image_processing/blob/main/literature/articles/generative_models/Demystifying_Variational_Diffusion_Models_Ribeiro_2024.pdf)

[12] [Lightweight Diffusion Models: A Survey, W. Song et al, 2024](https://github.com/dimitarpg13/deep_learning_for_image_processing/blob/main/literature/articles/generative_models/Lightweight_Diffusion_Models_A_Survey_Song_2024.pdf)

[13] [Score, Wikipedia](https://en.wikipedia.org/wiki/Informant_(statistics))

[14] [Tweedie's Formula and Selection Bias, Vradley Efron, Stanford U., 2011](https://github.com/dimitarpg13/probabilistic_machine_learning/blob/main/applied_statistics/articles/Tweedies_Formula_and_Selection_Bias_Effron_Stanford_2011.pdf)

[15] [Empirical Bayes Approach to Statistics, Herbert Robbins, Columbia U., 1956](https://github.com/dimitarpg13/probabilistic_machine_learning/blob/main/applied_statistics/articles/Empirical_Bayes_Approach_to_Statistics_Robbins_Columbia_1956.pdf)

[16] [Cumulant Generating Function, Marco Taboga, statlect.com](https://www.statlect.com/fundamentals-of-probability/cumulant-generating-function)

[17] [Exponential Family, Wikipedia](https://en.wikipedia.org/wiki/Exponential_family)