6.790 Notes

Lecturers: Pulkit Agrawal, Stephen Bates, Tommi Jaakkola, Shen Shen

Andrew Liu

Fall 2023

My notes for 6.790, "Grad ML", at the Massachusetts Institute of Technology during the Fall 2024 semester. The instructors for this course were Shen Shen (https://shenshen.mit.edu/), Pulkit Agrawal (https://people.csail.mit.edu/pulkitag/), Stephen Bates (https://stephenbates19.github.io/), and Tommi Jaakkola (https://people.csail.mit.edu/tommi/).

Last updated on Monday 27th November, 2023.

Contents

1 September 8, 2023

2 October 26, 2023

2.1 Gaussian Mixture Models

Each component or "cluster" is a Gaussian. A k component GMM in $\mathbb{R}^d$ has parameters

$$\theta = \{p_1, \dots, p_k, \mu_1, \dots, \mu_k, \sigma_1^2, \dots, \sigma_k^2\},$$

where $x \in \mathbb{R}^d$ (data), $\mu_j \in \mathbb{R}^d$, and $\sigma_j^2 \in \mathbb{R}^+$. We want to maximize

$$\log P(x|\theta) = \log\left(\sum_{z=1}^{k} P(z|\theta) \cdot P(x|z,\theta)\right),$$

where $P(z|\theta) = p_z$ is our prior mixing proportion, and

$$P(x|z,\theta) = \frac{1}{(2\pi\sigma_z^2)^{d/2}} \exp\left(-\frac{1}{2\sigma_z^2}\|x - \mu_z\|^2\right)$$

is Gaussian. We can maximize the log likelihood with gradient ascent, where the gradient is given by:

$$\begin{aligned}
\nabla_\theta \log P(x|\theta) &= \nabla_\theta \log\left(\sum_{z=1}^{k} P(z|\theta)P(x|z,\theta)\right) \\
&= \frac{1}{\sum_{z=1}^{k} P(z|\theta)P(x|z,\theta)} \nabla_\theta \sum_{z=1}^{k} P(z|\theta)P(x|z,\theta) \\
&= \sum_{z=1}^{k} \frac{P(z|\theta)P(x|z,\theta)}{P(x|\theta)} \nabla_\theta \log\left(P(z|\theta)P(x|z,\theta)\right) \\
&= \sum_{z=1}^{k} P(z|x,\theta) \nabla_\theta \log\left(P(z|\theta)P(x|z,\theta)\right) \\
&= \sum_{z=1}^{k} P(z|x,\theta) \nabla_\theta \left(\log p_z - \frac{1}{2\sigma_z^2}\|x - \mu_z\|^2 - \frac{d}{2}\log(2\pi\sigma_z^2)\right)
\end{aligned}$$

Now we can perform our gradient ascent as an EM algorithm on the posterior probabilities $Q(z|x) = P(z|x,\theta)$:

- E-step: assign data points x according to $Q(z|x)$.
- M-step: update the model based on the generated data,

$$\theta \leftarrow \theta + \eta \sum_{z=1}^{k} Q(z|x) \nabla_\theta \log\left(P(z|\theta)P(x|z,\theta)\right).$$

2.2 ELBO lower bound

The above works, but we can converge faster by using the ELBO lower bound:

$$\log P(x|\theta) \geq \sum_{z=1}^{k} Q(z|x) \log\left(P(z|\theta)P(x|z,\theta)\right),$$

which holds for all Q, θ. Proof that this works:

$$\begin{aligned} \sum_{z=1}^{k} Q(z|x) \log\left(P(z|\theta)P(x|z,\theta)\right) &\leq \max_z \log\left(P(z|\theta)P(x|z,\theta)\right) \\ &= \log\left(\sum_{z=1}^{k} P(z|\theta)P(x|z,\theta)\right) \\ &= \log P(x|\theta). \end{aligned}$$

The idea here is that instead of trying to optimize $P(x|\theta)$, we can directly optimize the ELBO lower bound to make larger steps during each iteration of our algorithm. We can make this bound tight by adding an entropy term so that ELBO is the same as the true log likelihood if and only if Q is equal to the true posterior:

$$\log P(x|\theta) \geq \sum_{z=1}^{k} Q(z|x) \log\left(P(z|\theta)P(x|z,\theta)\right) + H(Q_{z|x}),$$

where

$$H(Q_{z|x}) = -\sum_{z=1}^{k} Q(z|x) \log Q(z|x).$$

We can show that this works as follows:

$$\begin{aligned}
&\sum_z Q(z|x)\log\left(P(x|\theta)P(x|z,\theta)\right) + \sum_z Q(z|x)\log\frac{1}{Q(z|x)} \\
&= \sum_z Q(z|x)\log\left(\frac{P(z|\theta)P(x|z,\theta)}{Q(z|x)}\right) \\
&= \sum_z Q(z|x)\log\left(P(x|\theta)\frac{P(z|\theta)P(x|z,\theta)}{P(x|\theta)Q(z|x)}\right) \\
&= \log P(x|\theta) + \sum_z Q(z|x)\log\left(\frac{P(z|\theta)P(x|z,\theta)}{P(x|\theta)Q(z|x)}\right) \\
&= \log P(x|\theta) - \sum_z Q(z|x)\log\left(\frac{Q(z|x)}{P(z|x,\theta)}\right) \\
&= \log P(x|\theta) - KL(Q_{z|x}|P_{z|x,\theta}).
\end{aligned}$$

The KL-divergence of $Q_{z|x}$,$P_{z|x,\theta}$ is guaranteed to be non-zero and equal to zero if and only if $Q_{z|x} = P_{z|x,\theta}$, so this works.

Finally, the ELBO lower bound is a function of both Q and θ, so we can modify the original EM algorithm to take turns maximizing this lower bound by fixing one or the other:

- E-step: fix θ and compute $\hat{Q}(z|x)$ that maximizes ELBO
- M-step: fix $\hat{Q}$ and update θ to maximize ELBO (e.g. with gradient ascent)

3 October 27, 2023

3.1 K-Means Clustering

Place k centroids at random $c_1, \ldots, c_k$. Then, repeat until convergence:

1. For each x_i, locate the nearest centroid $c_j = \arg\min_{j=1}^k \mathcal{D}(x_i, c_i)$ for some distance metric $\mathcal{D}$.
2. For each cluster c_j, update the position $c_j = \sum_{x_i \in c_j} x_i/n$.

Using $\mathcal{D} = \|\cdot\|_2^2$, the objective function being minimized here is

$$C = \sum_{j=1}^{K} \sum_{x_i \in c_j} \|x_i - c_j\|_2^2$$

There are other ways to choose the centroids. For example, the *k*-mediods algorithm places the center of each cluster at

$$x_m = \underset{x \in \{x_1,\ldots,x_n\}}{\arg\min} \sum_{i=1}^{n} \|x_i - x\|^2.$$

This is better for outliers.

3.2 PCA Intuition

Say we have some data about football players and their different attributes: speed, scoring, height, weight. If we were to plot this data on a graph with each axis representing an attribute, we know that these axes would not all be orthogonal; for example, height and weight are strongly correlated. The idea of PCA is to try to decouple the data space into orthogonal vectors; then, we can more easily select the vectors that "matter", which allows us to reduce the dimensionality of the data while keeping its most important aspects.

3.3 PCA with Encoder/Decoder framework

We have $x_1,\ldots,x_n$ as d-dimensional data in $\mathbb{R}^d$. In PCA, we want to reduce the dimensionality to $m < d$, which we can do with an encoder matrix $W \in \mathbb{R}^{m\times d}$. We can also use a decoder matrix $U \in \mathbb{R}^{d\times m}$ to try to retrieve the original data from its low-dimensional representation. Our goal is to minimize the distance between the approximated output $x' = UWx$, and its original vector, i.e., we want to find the "best" encoder/decoder pair

$$W^*, U^* = \underset{W\in\mathbb{R}^{m\times d}, U\in\mathbb{R}^{d\times m}}{\arg\min} \sum_{i=1}^{n} \|x_i - UWx_i\|_2^2.$$

First, fix (U,W) and consider the set of all condensed representations $R = \{UWx : x \in \mathbb{R}^d\}$. This is an m-dimensional linear subspace of $\mathbb{R}^d$, and there exists some $V \in \mathbb{R}^{d\times m}$ whose columns form an orthonormal basis of R. So, we may use $V^TV = I_m$ (but not the other way around).

For all x_i, there exists $y_i \in \mathbb{R}^m$ s.t. $Vy_i = UWx_i$, so our objective function becomes

$$\begin{aligned}\sum_{i=1}^n \|x_i - Vy_i\|_2^2 &= \sum_{i=1}^n (x_i - Vy_i)^T(x_i - Vy_i)\\ &= \sum_{i=1}^n x_i^Tx_i + y_i^TV^TVy - 2y_i^T(V^Tx_i)\\ &= \sum_{i=1}^n \|x_i\|^2 + \|y_i\|^2 - 2y_i^T(V^Tx_i)\end{aligned}$$

since V is orthonormal. Differentiating each summed term, we have

$$0 = \frac{\mathrm{d}}{\mathrm{d}y_i}(\|x_i\|^2 + \|y_i\|^2 - 2y_i^T(V^Tx_i)) = -2V^Tx_i + 2y_i \implies y_i = V^Tx_i.$$

This bound is achieved when $U = V, W = V^T$, so we can now reformulate our objective:

$$\begin{aligned}&\underset{U\in\mathbb{R}^{d\times m}, U^TU=I_m}{\arg\min} \quad \|x_i - UU^Tx_i\|^2\\ =& \underset{U\in\mathbb{R}^{d\times m}, U^TU=I_m}{\arg\min} \quad x_i^Tx_i + x_i^TUU^TUU^Tx_i - 2x_i^TUU^Tx_i\\ =& \underset{U\in\mathbb{R}^{d\times m}, U^TU=I_m}{\arg\max} \quad x_i^TUU^Tx_i\end{aligned}$$

Recall that this expression is not the same as $x_i^Tx_i$, since we can't say that VV^T is the identity. Finally, summing over all elements,

$$U^* = \underset{U\in\mathbb{R}^{d\times m}, U^TU=I_m}{\arg\max} \quad \mathrm{Tr}\left(U^T\sum_{i=1}^n x_ix_i^TU\right)$$

Define $S = \sum_{i=1}^n x_ix_i^T \in \mathbb{R}^{d\times d}$; we want to maximize $\mathrm{Tr}(U^TSU)$. S is positive semi-definite and symmetric, so by the spectral decomposition theorem we have $S = P\Lambda P^T$, where Λ is diagonal with elements the eigenvalues of S and P is the corresponding orthonormal eigenbasis. Choose P, Λ so that $\Lambda_{i,i} \geq \Lambda_{j,j}$ for all $i \leq j$, i.e., order eigenvalues decreasing from left to right.

Now,

$$\mathrm{Tr}(U^T S U) = \mathrm{Tr}(U^T P \Lambda P^T U) = \mathrm{Tr}\Big((P^T U)^T \Lambda (P^T U)\Big).$$

Denoting $B = P^T U$, then

$$\mathrm{Tr}(U^T S U) = \sum_{i=1}^{m} \sum_{j=1}^{d} B_{ji}^2 \Lambda_{j,j} = \sum_{i=1}^{d} \Lambda_{i,i} \sum_{j=1}^{m} B_{ij}^2 \leq \sum_{i=1}^{m} \Lambda_{i,i}.$$

This bound is tight when U has columns equal to the first m eigenvectors in the eigenbasis of S, since these leads to B with identity values along the diagonal. More explicitly,

$$B_{ij} = \sum_{k=1}^{d} P_{ik}^T U_{kj} = P_i \cdot P_j,$$

where P_i is the ith eigenvector in S, i.e., the ith column of P. By orthonormality, this is 0 when $i \neq j$ and 1 when $i = j$, so $B = I_{d\times m}$. Finally, plugging this construction into the above trace formula gives the exact bound, so this works. This shows that the optimal U^*, W^* is given by the eigenbasis of S, and the corresponding error for this construction is

$$\mathrm{Tr}\left(\sum_{i=1}^{n} x_i x_i^T + U^T \sum_{i=1}^{n} x_i x_i^T U\right) = \sum_{i=1}^{d} \Lambda_{i,i} - \sum_{i=1}^{m} \Lambda_{i,i} = \sum_{i=m+1}^{d} \Lambda_{i,i}.$$

3.4 Probabilistic PCA

We have observation $x \in \mathbb{R}^d$ generated in latent space by some $z \in \mathbb{R}^m$, $m < d$, via

$$x = Wz + \mu + \varepsilon,$$

where $\varepsilon \sim \mathcal{N}(0, \sigma^2 I)$ is noise. μ represents the "mean" latent space representation of x. If we let $z \sim \mathcal{N}(0, I)$, then $Wz \sim \mathcal{N}(0, WW^T)$, and $x \sim \mathcal{N}(\mu, WW^T + \sigma^2 I)$. Thus, the log likelihood of a full dataset is

$$NLL(\mathcal{X}|\mu, \mathcal{W}, \sigma^2) = -\frac{nd}{2} \ln 2\pi - \frac{n}{2} \ln \det \Sigma - \frac{1}{2} \sum_{i=1}^{n} (x_n - \mu)^T \Sigma^{-1} (x_n - \mu).$$

The maximum likelihood PCA satisfies

$$0 = \frac{\partial NLL(\mathcal{X}|\mu, \mathcal{W}, \sigma^2)}{\partial \mu} = \sum_{i=1}^{n} \Sigma^{-1}(x_n - \mu) \implies \mu = \overline{x}.$$

3.5 Gaussian Mixture Model Intuition

Gaussian mixture models are an extension of K-means clustering. In these models, the model is parameters by some M components, each of which generates a distribution of points according to some model, e.g., multivariate normal. The likelihood of the model can then be evaluated according to some priors placed on how likely each point is to be in each of the M components, and then a max likelihood can be solved for using techniques like gradient ascent. We will show in the next problem that this unfortunately does not always work as we expect it to.

3.6 Arbitrarily Bad Local Maxima in GMMs

(This problem was sourced from Chi Jin, et. al. 2016).

Consider a GMM with $M = 3$ on the number line and true centers generated by $\mu_1^* = -R$, $\mu_2^* = R$, and $\mu_3^* = \gamma R$, where $R > 0$ and $\gamma \gg 1$, and all with variance 1. The idea here is for μ_1^*, μ_2^* to be close to the origin and μ_3^* to be very far away.

Assuming uniform prior on each of the three clusters, the likelihood of a model $\mu = (\mu_1, \mu_2, \mu_3)$ on n observations is given by

$$\mathcal{L}_n(\mu) = \frac{1}{n}\sum_{i=1}^{n} \log\left(\frac{1}{3}\sum_{j=1}^{3} \frac{1}{\sqrt{2\pi}} e^{-(x_i-\mu_j)^2/2}\right).$$

With infinite data, we can take the loss as an expected value over all possible data:

$$\begin{aligned}\mathcal{L}(\mu) &= \mathbb{E}_x\left[\log\left(\sum_{i=1}^{3} e^{-(x-\mu_i)^2/2}\right) - \log(3\sqrt{2\pi})\right].\\ &= \sum_{V\in\{A,B,C\}} \frac{1}{3}\mathbb{E}_V\left[\log\left(\sum_{i=1}^{3} e^{-(V-\mu_i)^2/2}\right)\right] - \log(3\sqrt{2\pi}),\end{aligned}$$

where the expected value is decomposed (by the law of total expectation) based on the assumption of uniform priors between the three clusters. A, B, and C are random variables that follow the true generating distribution μ^*, since this is what we should observe with infinite data.

Example 3.1

Consider the set of points $\mathcal{D} = \{(\mu_1, \mu_2, \mu_3) \in \mathbb{R}^3 | \mu_1 \le \gamma R/3, \mu_2 \ge 2\gamma R/3, \mu_3 \ge 2\gamma R/3\}$. In the limit as $\gamma \to \infty$, compute:

- the loss on the three borders of this region, i.e., when $\mu_1 = \gamma R/3$, when $\mu_2 = 2\gamma R/3$, or when $\mu_3 = 2\gamma R/3$.
- the loss on the model $\tilde{\mu} = (0, \gamma R, \gamma R)$.

Starting with $\tilde{\mu}$, first expand the entire definition:

$$\begin{aligned}\mathcal{L}(\tilde{\mu}) = &\frac{1}{3}\mathbb{E}_A\left[\log\left(e^{-(A-0)^2/2} + e^{-(A-\gamma R)^2/2} + e^{-(A-\gamma R)^2/2}\right)\right]\\ &+ \frac{1}{3}\mathbb{E}_B\left[\log\left(e^{-(B-0)^2/2} + e^{-(B-\gamma R)^2/2} + e^{-(B-\gamma R)^2/2}\right)\right]\\ &+ \frac{1}{3}\mathbb{E}_C\left[\log\left(e^{-(C-0)^2/2} + e^{-(C-\gamma R)^2/2} + e^{-(C-\gamma R)^2/2}\right)\right] - \log(3\sqrt{2\pi}).\end{aligned}$$

Taking the limit,

$$\lim_{\gamma\to\infty} \mathcal{L}(\tilde{\mu}) = \frac{1}{3}\mathbb{E}_A\left[\log(e^{-A^2/2})\right] + \frac{1}{3}\mathbb{E}_B\left[\log(e^{-B^2/2})\right] + \frac{1}{3}\mathbb{E}_C\left[\log(2e^{-(C-\gamma R)^2/2})\right] - \log(3\sqrt{2\pi}).$$

Doing a bit more algebra,

$$\begin{aligned}\mathbb{E}[A^2] &= \mathrm{Var}[A] + \mathbb{E}[A]^2 = 1 + R^2\\ \mathbb{E}[B^2] &= \mathrm{Var}[B] + \mathbb{E}[B]^2 = 1 + R^2\\ \mathbb{E}[(C-\gamma R)^2] &= \mathbb{E}[C^2 - 2C\gamma R + \gamma^2 R^2] = 1,\end{aligned}$$

so

$$\lim_{\gamma\to\infty} \mathcal{L}(\tilde{\mu}) = -\frac{2R^2 + 3 - 2\log 2}{6} - \log(3\sqrt{2\pi}).$$

Repeating the same algebra for the borders, it can be shown that the loss is strictly less on the borders than it is for $\tilde{\mu}$. This is significant because it shows that there is a local maxima in $\mathcal{D}$.

Example 3.2

Show that this local maximum can be arbitrarily bad compared to the global maximum, i.e., that the log likelihood can be arbitrarily worse in the limit $R \to \infty$.

Repeating the same calculation as above, we can compute the global loss by plugging in $\mu^* = (-R, R, \gamma R)$:

$$\lim_{R\to\infty} \mathcal{L}(\mu^*) = \frac{1}{3}\left(-(A+R)^2/2 - (B-R)^2/2 - (C-\gamma R)^2/2\right) - \log(3\sqrt{2\pi}),$$

where

$$\mathbb{E}[(A+R)^2] = \mathbb{E}[(B-R)^2] = \mathbb{E}[(C-\gamma R)^2] = 1,$$

so

$$\lim_{R\to\infty} \mathcal{L}(\mu^*) = -\frac{1}{2} - \log(3\sqrt{2\pi}).$$

We can show that $\lim_{R\to\infty} \mathcal{L}(\mu') = -\infty$ as follows. $\mathcal{L}(\mu')$ is bounded above by the μ_2 term, since A and C grow arbitrarily far from the center R; but, the contribution of loss from the μ_2 term is $-\infty$ since all three of μ_1', μ_2', μ_3' grow arbitrarily far from center. Therefore, in the limit, global loss is constant, while our local maximum can achieve arbitrarily bad loss. Since loss is continuous in R, this shows that we can grow the gap between our local and global maxima arbitrarily large by increasing R. The significance of this result is that gradient ascent does not always work as expected!

4 October 31, 2023

4.1 Variational Autoencoders

Another generative model is the variational autoencoder. The high level goal of VAEs is to compress a complex data distribution into a smaller latent space, and then be able to regenerate the distribution from latent space.

In particular, the **inference model**, or **encoder**, or **recognition model** attempts to learn $q_\phi(z|x) \approx P_\theta(z|x)$, where z is some data in the latent space, and $x \sim P_\theta$ is the true data distribution.

This encoding process is very similar to the encoding we use for GMMs. In particular, we can learn q_ϕ by minimizing log likelihood $\log P_\theta(x)$, which we can do with the ELBO lower bound (we quickly derive another way to write this expression):

$$\begin{aligned}\mathcal{L}_{\theta,\phi}(x) &= \sum_z q_\phi(z|x)\log(P_\theta(x)P_\theta(x|z)) + \sum_z q_\phi(z|x)\log\frac{1}{q_\phi(z|x)} \\ &= \sum_z q_\phi(z|x)\log\left(\frac{P_\theta(z)P(x|z)}{q_\phi(z|x)}\right) = \mathbb{E}_{q_\phi(z|x)}\left(\log\frac{P_\theta(x,z)}{q_\phi(z|x)}\right).\end{aligned}$$

We showed previously that $\log P_\theta(x) = \mathcal{L}_{\theta,\phi}(x) + KL(q_\phi(z|x), P_\theta(z|x))$, where $KL(q_\phi(z|x), P_\theta(z|x)) \geq 0$ with equality iff our model correctly predicts $q_\phi = P_\theta$. In other words,

$$\log P_\theta(x) \geq \int q_\phi(z|x)\log\left(\frac{P_\theta(x,z)}{q_\phi(z|x)}\right)\mathrm{d}z + H(q_\phi(z|x))$$

with equality iff we predict correctly, so we can try optimizing the ELBO lower bound instead of the log likelihood directly.

It is common to reparamaterize the latent space $z \sim q_\phi(z|x) = \mathcal{N}(z; \mu(x,\phi), (\sigma(x;\phi)^2))$, where $\mu(x;\phi)$ and $\sigma(x;\phi)$ completely specifies the posterior distribution. These can be learned with a neural net, e.g.,

$$\begin{aligned}(\mu, \log\sigma) &= \text{EncoderNeuralNet}_\phi(x) \\ q_\phi(z|x) &= \mathcal{N}(z; \mu, (\sigma)).\end{aligned}$$

During learning, it is common to represent q_ϕ as a function of some externally chosen noise; for example, we might sample $\varepsilon \sim \mathcal{N}(0,1)$, and then take $z_\phi = \mu(x;\phi) + \sigma(x;\phi) \odot \varepsilon$. This way, we can backprop from the final prediction through the latent space and back to ϕ. If we sample directly from the latent space, we can't backprop (we can compute the gradient when ε is an input to the function, but we can't when we have to sample ε as part of the function).

The **generative model**, or **decoder**, remaps data from the latent space into the complex data space. This part of the model attempts to learn $P_\theta(x|z)$, which is also commonly reparamaterized as a collection of gaussians $\mathcal{N}(z; g_\theta(z), \sigma^2 I)$. Parameters g, σ can again be learned with a neural net, $\text{DecoderNeuralNet}_\theta(z)$. Lastly, training can be done with EM, similar to the EM from Gaussian mixtures:

- decoder:
$$\theta = \theta + \eta \nabla_\theta \log P_\theta(x|z_\phi).$$

- encoder:
$$\phi = \phi + \eta \nabla_\phi \Big(\log P_\theta(x|z_\phi) - KL(q_\phi(z|x)|P_\theta(z))\Big).$$

where gradient ascent updates are made according to the ELBO criterion.

5 November 3, 2023

5.1 Forward Diffusion

In diffusion, we fix a forward process that adds Gaussian noise to an image. We then use a reverse de-noising process to reverse this process and generate images from noise.

More specifically, we start with some data x_0 sampled from distribution $q(x)$. Then, we define a forward diffusion process

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I),$$

where the probability of the entire process up to time T is

$$q(x_{1:T}|x_0) = \prod_{t=1}^{T} q(x_t|x_{t-1}).$$

At each time step, we're injecting a bit of noise into the image. By the end of the forwards process, x_T is isotropic (pure noise). Usually, $\beta_1 < \beta_2 < \ldots < \beta_T$ with some scheduling process (linear, cosine) to ensure that this is true. Using nice properties of Gaussians, we can sample any timestep directly instead of having to simulate the entire process each time.

Claim 5.1
Let $\alpha_t = 1 - \beta_t$ and $\overline{\alpha}_t = \prod_{i=1}^{t} \alpha_i$, and $\varepsilon \sim \mathcal{N}(0, I)$. Then,

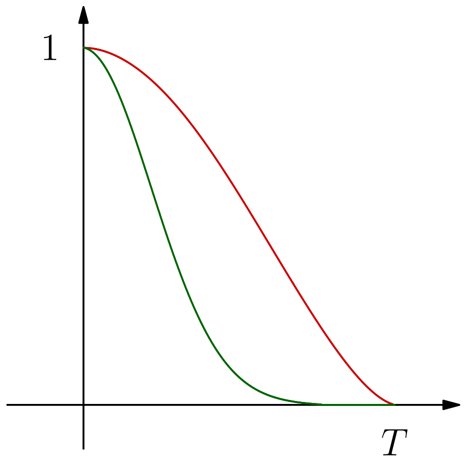
$$x_t = \sqrt{\overline{\alpha}_t}x_0 + \sqrt{1-\overline{\alpha}_t}\varepsilon.$$

Proof. We can do this with induction. This clearly holds for $t = 1$. Now, for arbitrary $t > 1$,

$$\begin{aligned} x_t &= \sqrt{\alpha_t} x_{t-1} + \sqrt{1-\alpha_t}\varepsilon_t \\ &= \sqrt{\alpha_t}\sqrt{\overline{\alpha}_{t-1}} x_0 + \sqrt{\alpha_t - \alpha_t \overline{\alpha}_{t-1}}\varepsilon + \sqrt{1-\alpha_t}\varepsilon_t \\ &= \sqrt{\overline{\alpha}_t} x_0 + \sqrt{1-\overline{\alpha}_t}\varepsilon, \end{aligned}$$

where the last equality comes from linearity of variance for independent gaussian noise. This completes the induction. □

This result shows that we can think of the image at timestep t as a linear combination of pure noise and the original image, where the proportion assigned to pure noise approaches 1 as $t \to T$. The graph below visualizes linear vs cosine scheduling:



where the y-axis is $\overline{\alpha}_t$ (i.e., the proportion of the original image that we are preserving), x-axis is time, and the green and red schedules are linear and cosine schedules, respectively. The cosine schedule is more gradual and has been shown to generally produce better results for smaller image sizes, e.g., $32x32$ pixels.

5.2 Going Backwards

The more difficult step in diffusion models is figuring out how to generate data in the complex data space (i.e., an image) from pure noise. When β_t is small, $q(x_{t-1}|x_t)$ is essentially a gaussian, so we can attempt to learn $p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t))$.

Unfortunately, $q(x_{t-1}|x_t)$ is intractable, but $q(x_{t-1}|x_t, x_0)$ can be calculated by flipping everything to the forwards direction with Bayes. Per the normal distribution for multivariate distributions, we want something that looks like this:

$$q(x_{t-1}|x_t, x_0) = \frac{1}{\sqrt{(2\pi)^k \det \Sigma_\theta(x_t, t)}} \exp\left(-\frac{\|x_{t-1} - \mu_\theta(x_t, t)\|^2}{2 \det \Sigma_\theta(x_t, t)}\right),$$

where we implicitly assume $\Sigma_\theta(x_t, t) = \tilde{\beta}_t I$ for some derived value of $\tilde{\beta}_t$ (our noise is independent). Recall that $q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{\alpha_t} x_{t-1}, \beta_t I)$ and $q(x_t|x_0) \sim \mathcal{N}(x_t; \sqrt{\overline{\alpha}_t} x_0, \sqrt{1 - \overline{\alpha}_t} I)$, so we have

$$\begin{aligned} q(x_{t-1}|x_t, x_0) &= q(x_t|x_{t-1}, x_0) \frac{q(x_{t-1}|x_0)}{q(x_t|x_0)} \\ &= \frac{1}{\sqrt{(2\pi)^k \beta_t^k}} \frac{\sqrt{(2\pi)^k (1 - \overline{\alpha}_t)^k}}{\sqrt{(2\pi)^k (1 - \sqrt{\overline{\alpha}_{t-1}})^k}} \exp\left(-\frac{\|x_{t-1} - \mu_\theta(x_t, t)\|^2)}{2\Sigma_\theta(x_t, t)}\right). \end{aligned}$$

Therefore,

$$\tilde{\beta} = \frac{1 - \sqrt{\overline{\alpha}_{t-1}}}{1 - \sqrt{\overline{\alpha}_t}} \beta_t.$$

To compute the mean, we continue expanding inside of the exp:

$$\begin{aligned} q(x_{t-1}|x_t, x_0) &= q(x_t|x_{t-1}, x_0) \frac{q(x_{t-1}|x_0)}{q(x_t|x_0)} \\ &= \frac{1}{\sqrt{(2\pi\tilde{\beta})^k}} \exp\left(-\frac{1}{2}\left(\frac{(x_t - \sqrt{\alpha}_t x_{t-1})^2}{\beta_t} + \frac{(x_{t-1} - \sqrt{\overline{\alpha}_{t-1}} x_0)^2}{1 - \overline{\alpha}_{t-1}} - \frac{(x_t - \sqrt{\overline{\alpha}_t} x_0)^2}{(1 - \overline{\alpha}_t)}\right)\right) \\ &= \frac{1}{\sqrt{(2\pi\tilde{\beta})^k}} \exp\left(-\frac{1}{2}\left(x_{t-1}^2 \left(\frac{\alpha_t}{\beta_t} + \frac{1}{1 - \overline{\alpha}_{t-1}}\right)\right.\right. \\ &\qquad \left.\left. - x_{t-1}\left(\frac{2\sqrt{\alpha_t}}{\beta_t} x_t + \frac{2\sqrt{\overline{\alpha}_{t-1}}}{1 - \overline{\alpha}_{t-1}} x_0\right) + \frac{x_t^2}{\beta_t} + \frac{\overline{\alpha}_{t-1} x_0^2}{1 - \overline{\alpha}_{t-1}} - \frac{(x_t - \sqrt{\overline{\alpha}_t} x_0)^2}{1 - \overline{\alpha}_t}\right)\right), \end{aligned}$$

where matrix multiplications are handwaved. We can verify

$$\frac{\alpha_t}{\beta_t} + \frac{1}{1 - \overline{\alpha}_{t-1}} = \frac{\alpha_t - \overline{\alpha}_t + 1 - \alpha_t}{\beta_t (1 - \overline{\alpha}_{t-1})} = \frac{1}{\tilde{\beta}_t},$$

so we have

$$q(x_{t-1}|x_t, x_0) = \frac{1}{\sqrt{(2\pi\tilde{\beta}^k)}} \exp\left(-\frac{\|x_{t-1} - \mu_\theta(x_t, t)\|^2}{2\tilde{\beta}_t^k}\right).$$

(the exponents don't quite match up because we are handwaving away all the matrix logic). Comparing coefficients gives

$$\begin{aligned}
\mu_\theta(x_t, t) &= \left(\frac{\sqrt{\alpha}_t}{\beta_t} x_t + \frac{\sqrt{\overline{\alpha}_{t-1}}}{1 - \overline{\alpha}_{t-1}} x_0\right)\tilde{\beta} \\
&= \frac{\sqrt{\alpha_t}(1 - \overline{\alpha}_{t-1})}{1 - \overline{\alpha}_t} x_t + \frac{\sqrt{\overline{\alpha}_{t-1}}\beta_t}{1 - \overline{\alpha}_t} x_0 \\
&= \frac{\sqrt{\alpha_t}(1 - \overline{\alpha}_{t-1})}{1 - \overline{\alpha}_t} x_t + \frac{\sqrt{\overline{\alpha}_{t-1}}\beta_t}{1 - \overline{\alpha}_t} \frac{1}{\sqrt{\overline{\alpha}_t}}(x_t - \sqrt{1 - \overline{\alpha}_t}\varepsilon_t) \\
&= \frac{1}{\sqrt{\alpha_t}} \frac{\alpha_t - \overline{\alpha}_t}{1 - \overline{\alpha}_t} x_t + \frac{1}{\sqrt{\alpha_t}} \frac{1 - \alpha_t}{1 - \overline{\alpha}_t}(x_t - \sqrt{1 - \overline{\alpha}_t}\varepsilon_t) \\
&= \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{1 - \alpha_t}{\sqrt{1 - \overline{\alpha}_t}}\varepsilon_t\right).
\end{aligned}$$

5.3 Theoretical Loss

We can now explicitly compute $q(x_{t-1}|x_t, x_0) \sim \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t))$, so we can compute the backwards diffusion step at any timestep conditioned on knowing x_0. To find the best possible x_0, we can optimize log likelihood use the same ELBO lower bound used to optimize VAEs and gaussian mixtures:

$$\log p_\theta(x_0) \geq \mathbb{E}_q\left(\frac{p_\theta(x_0, x_{1:T})}{q(x_{1:T}|x_0)}\right) = \mathbb{E}_q\left(\frac{p_\theta(x_{0:T})}{q(x_{1:T}|x_0)}\right).$$

To turn this into a minimization problem, we optimize $L_{VLB} = -\log p_\theta(x_0)$, where VLB stands for variational lower bound. With a bit of algebra, we can simplify this expression:

$$
\begin{aligned}
L_{VLB} &= \mathbb{E}_q\left(\log\frac{q(x_{1:T}|x_0)}{p_\theta(x_{0:T})}\right)\\
&= \mathbb{E}_q\left(\log\frac{\prod_{t=1}^T q(x_t|x_{t-1})}{p_\theta(x_T)\prod_{t=1}^T p_\theta(x_{t-1}|x_t)}\right)\\
&= \mathbb{E}_q\left(-\log p_\theta(x_T)+\left(\sum_{t=1}^T\log\frac{q(x_t|x_{t-1})}{p_\theta(x_{t-1}|x_t)}\right)+\log\frac{q(x_1|x_0)}{p_\theta(x_0|x_1)}\right)\\
&= \mathbb{E}_q\left(-\log p_\theta(x_T)+\left(\sum_{t=2}^T\log\frac{q(x_{t-1}|x_t,x_0)q(x_t|x_0)}{p_\theta(x_{t-1}|x_t)q(x_{t-1}|x_0)}\right)+\log\frac{q(x_1|x_0)}{p_\theta(x_0|x_1)}\right)\\
&= \mathbb{E}_q\left(-\log p_\theta(x_T)+\left(\sum_{t=2}^T\log\frac{q(x_{t-1}|x_t,x_0)}{p_\theta(x_{t-1}|x_t)}\right)+\log\frac{q(x_T|x_0)}{q(x_1|x_0)}+\log\frac{q(x_1|x_0)}{p_\theta(x_0|x_1)}\right)\\
&= \mathbb{E}_q\left(\log\frac{q(x_T|x_0)}{p_\theta(x_T)}+\left(\sum_{t=2}^T\log\frac{q(x_{t-1}|x_t,x_0)}{p_\theta(x_{t-1}|x_t)}\right)-\log p_\theta(x_0|x_1)\right)\\
&= D_{KL}(q(x_T|x_0)\|p_\theta(x_T))+\sum_{t=2}^T D_{KL}(q(x_{t-1}|x_t,x_0)\|p_\theta(x_{t-1}|x_t))-\log p_\theta(x_0|x_1)\\
&= L_T+\sum_{t=2}^T L_{t-1}+L_0.
\end{aligned}
$$

5.4 Training Loss

In practice, L_T can be ignored, since x_T is always pure gaussian noise. There are some things we can do for L_0 that are not that important here. Therefore, we care about optimizing L_t for $t = 1$ to $T-1$.

The KL divergence for multivariate normal distributions is given by

$$D_{KL}(\mathcal{N}_0\,\|\,\mathcal{N}_1)=\frac{1}{2}\left(\mathrm{Tr}(\Sigma_1^{-1}\Sigma_0)+\frac{1}{\det\Sigma_1}\|\mu_1-\mu_0\|^2-k+\ln\frac{\det\Sigma_1}{\det\Sigma_0}\right).$$

To minimize L_t, we only care about terms that depend on θ. Our expression for posterior variance

$$\tilde{\beta}_t=\frac{1-\overline{\alpha}_{t-1}}{1-\overline{\alpha}_t}\cdot\beta_t$$

does not depend on θ, since all of our α,β are predetermined, so the only terms we care about in the expression for KL divergence are the terms with μ_0, μ_1, i.e.,

$$\frac{1}{\det \Sigma_1} \|\mu_1 - \mu_0\|^2,$$

in the notation of the above expression. We know $q(x_{t-1}|x_t, x_0) = \mathcal{N}(x_{t-1}; \tilde{\mu}(x_t, x_0), \tilde{\beta}_t I)$, and we want our model $p_\theta(x_{t-1}|x_t)$ to learn $\mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t))$, where $\mu_\theta(x_t, t) \approx \tilde{\mu}(x_t, x_0)$ and $\Sigma_\theta(x_t, t) \approx \tilde{\beta}_t I$. Since we are ultimately trying to minimize log loss across all possible x_0, we minimize the EV across x_0. Also, like the reparamaterization trick for VAEs, we don't want to sample directly from each gaussian when computing values like $\tilde{\mu}_t(x_t, x_0)$; instead, we first sample $\varepsilon_t \sim \mathcal{N}(0, 1)$, and let $\tilde{\mu}$ take it as a parameter, so that we can backprop through the function properly. In sum, our goal is to now minimize EV across both x_0 and ε_t, which gives us

$$L_t - C = \mathbb{E}_{x_0, \varepsilon_t}\left(\frac{1}{2\|\Sigma_\theta(x_t, t)\|_2^2} \|\tilde{\mu}_t(x_t, x_0) - \mu_\theta(x_t, t)\|^2\right)$$

with C independent of θ. Plugging in known values, this simplifies:

$$\begin{aligned} L_t - C &= \mathbb{E}_{x_0, \varepsilon_t}\left(\frac{1}{2\|\Sigma_\theta(x_t, t)\|_2^2} \left\| \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{1-\alpha_t}{\sqrt{1-\overline{\alpha}_t}}\varepsilon_t\right) - \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{1-\alpha_t}{\sqrt{1-\overline{\alpha}_t}}\varepsilon_\theta(x_t, t)\right)\right\|^2\right) \\ &= \mathbb{E}_{x_0, \varepsilon_t}\left(\frac{(1-\alpha_t)^2}{2\alpha_t(1-\overline{\alpha}_t)\|\Sigma_\theta(x_t, t)\|^2} \|\varepsilon_t - \varepsilon_\theta(\sqrt{\overline{\alpha}_t}x_0 + \sqrt{1-\overline{\alpha}_t}\varepsilon_t, t)\|^2\right). \end{aligned}$$

In practice, it has been shown that total loss L given by

$$L - C = \mathbb{E}_{t, x_0, \varepsilon_t}(\|\varepsilon_t - \varepsilon_\theta(\sqrt{\overline{\alpha}_t}x_0 + \sqrt{1-\overline{\alpha}_t}\varepsilon_t, t)\|^2)$$

gives better results (note that we have added t to the expected value, so this represents a total loss objective over all timesteps). This is really convenient for training, because this is just the expected mean squared error between the actual noise added to an image at time t, ε_t, and the error predicted by the backwards diffusion process. We can therefully usually train by randomly selecting a batch of images, timesteps, and noise to add to each image, and then incurring loss equal to the MSE of the noise predicted by the unet and the actual noise that was added to each image.

5.5 Turning diffusion models into classifiers

In addition to normal diffusion models that learn $p_\theta(x_{0:T})$, there are also conditional diffusion models that learn $p_\theta(x_{0:T}|c_i)$. The only difference between class conditional models and normal models is that the neural network learned during the backwards diffusion process takes the class as an additional parameter. It turns out that we can also use these models as classifiers.

$$p_\theta(c_i|x) = \frac{p(c_i)p_\theta(x|c_i)}{\sum_j p(c_j)p_\theta(x|c_j)}.$$

The learned UNet produces $\mathcal{L}_\theta(x|c_i) \leq \log p_\theta(x|c_i)$. In principle, our learned ELBO should be the same as the actual log likelihood, so we may say $\mathcal{L}_\theta(x|c_i) \approx \log p_\theta(x|c_i)$. Assuming uniform priors on all of the classes, we thus have

$$p_\theta(c_i|x) = \frac{\exp(-\mathbb{E}_{t,\varepsilon}||\varepsilon - \varepsilon_\theta(x_t, c_i)||^2)}{\sum_j \exp(-\mathbb{E}_{t,\varepsilon}||\varepsilon - \varepsilon_\theta(x_t, c_j)||^2)}.$$

ELBO values can be approximated via Monte Carlo by sampling (t, ε) pairs, using the trained UNet to predict $\varepsilon_\theta(x_i, c_i)$, and computing the expected mean squared loss of the errors over all samples. This is an overkill but interesting way to classify things that has shown decent results.

6 November 7, 2023

Hi