6.790 Notes

Lecturer:

Andrew Liu

Fall 2023

Last updated on Saturday 25th 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

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

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

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

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

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

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 ex-

ists 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:

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

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,

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

Define $S = \sum_{i=1}^{n} x_ix_i^T \in \mathbb{R}^{d\times d}$; we want to maximize $\operatorname{Tr}(U^TSU)$. 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,

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

Denoting $B = P^T U$, then

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

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

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

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

Generative models specify a mechanism for creating objects. Latent variables represent some of the underlying choices that generate these objects. To optimize mixtures of latent variables, we can use the ELBO lower bound discussed previously:

$$\log P(x|\theta) \geq \int Q(z|x) \log\left(P(x|z)P(z)\right) \mathrm{d}z + H(Q_{z|x}).$$

At a high level, VAEs map input data into a lower dimensional latent space, which can then be remapped to data that should closely resemble the original data. Mapping data into the latent space is done by the **inferential model**, or encoder, and mapping from the latent space back into the data space is done by the **generative model**, or decoder.

The role of the encoder is to take in data and produce a posterior distribution $Q(z|x)$ approximating the true $P(z|x,\theta)$. In particular, given x, it might produce $\mu(x;\phi)$ and $\sigma(x;\phi)$ specifying the posterior distribution, from which we can produce samples $z \sim Q(z|x,\phi) = \mathcal{N}(z;\mu(x;\phi),\text{diag}(\sigma(x;\phi)^2))$. These sampled values can then be used to update both the encoder and decoder:

- decoder:
$$\theta = \theta + \eta\nabla_\theta \log P(x|z_\phi,\theta).$$
- encoder:
$$\phi = \phi + \eta\nabla_\phi \left(\log P(x|z_\phi,\theta) - KL(Q_{z|x,\phi}|P_z)\right).$$

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

fix this i think something is wrong

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,

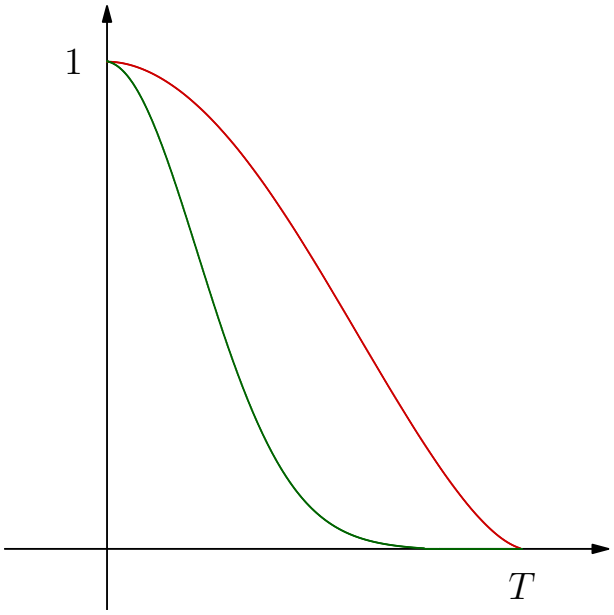
$$x_t = \sqrt{\overline{\alpha}_t} x_0 + \sqrt{1 - \overline{\alpha}_t}\varepsilon.$$

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 $t \to T$. Here is a graph visualizing linear vs cosine scheduling:



where the y-axis is $\overline{\alpha}_t$, 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 Backwards Diffusion

5.3 Turning diffusion models into classifiers

6 November 7, 2023

Hi