DGM assignment 1 solution

Question 1

1.1 Log-Likelihood Function

Maximum Likelihood Estimation (MLE) of an i.i.d. sample:

Given an i.i.d. sample $(X_t; t = 1, \dots, T)$ where $X_t \sim f_\theta(x)$, the likelihood function is:

$$L(\theta) = f_\theta(X_1, X_2, \dots, X_T) = \prod_{t=1}^{T} f_\theta(X_t),$$

due to independence. Therefore, the log-likelihood function is:

$$\log L(\theta) = \sum_{t=1}^{T} \log f_\theta(X_t),$$

and the maximum likelihood estimate is:

$$\hat{\theta}_{ML} = \arg\max_\theta \log L(\theta).$$

For the AR(1) model:

$$Y_t = \phi Y_{t-1} + \varepsilon_t, \quad t = 1, \dots, T,$$

where $\varepsilon_t \sim$ i.i.d. $N(0, \sigma^2)$ and y_0 is fixed/known. The parameter vector is $\Theta = (\phi, \sigma^2)$.

The joint probability can be expressed using the chain rule:

$$f_\theta(y_0, y_1, \dots, y_T) = f_\theta(y_T | y_{T-1}, \dots, y_0) f_\theta(y_{T-1}, \dots, y_0).$$

By recursively applying this, we get:

$$f_\theta(y_0, y_1, \dots, y_T) = \prod_{t=1}^{T} f_\theta(y_t | y_{t-1}, \dots, y_0) f_\theta(y_0).$$

Since Y_t given past values depends only on Y_{t-1}:

$$Y_t | Y_{t-1}, \dots, y_0 \equiv Y_t | Y_{t-1},$$

so:

$$f_\theta(Y_t | Y_{t-1}, \dots, y_0) = f_\theta(Y_t | Y_{t-1}).$$

The conditional distribution is:

$$Y_t | Y_{t-1} \sim N(E(Y_t | Y_{t-1}), V(Y_t | Y_{t-1})),$$

where:

$$E(Y_t | Y_{t-1}) = \phi Y_{t-1}, \quad V(Y_t | Y_{t-1}) = \sigma^2.$$

Thus, the conditional density is:

$$f_\theta(Y_t | Y_{t-1}) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(Y_t - \phi Y_{t-1})^2}{2\sigma^2} \right).$$

The conditional likelihood function is:

$$L(\theta) = f_\theta(Y_1, Y_2, \dots, Y_T | y_0) = \prod_{t=1}^{T} f_\theta(Y_t | Y_{t-1}).$$

Therefore, the log-likelihood function is:

$$\log L(\theta) = \sum_{t=1}^{T} \log f_\theta(Y_t | Y_{t-1}).$$

1.2 Maximum Likelihood Estimation

To find the maximum likelihood estimates, we solve:

$$\hat{\theta} = \arg\max_{\theta} \log L(\theta).$$

The estimate for ϕ is:

$$\hat{\phi} = \frac{\frac{1}{T}\sum_{t=1}^{T} Y_{t-1}Y_t}{\frac{1}{T}\sum_{t=1}^{T} Y_{t-1}^2},$$

which is the Ordinary Least Squares (OLS) estimator.

The estimate for σ^2 is:

$$\hat{\sigma}^2 = \frac{1}{T}\sum_{t=1}^{T} \hat{\varepsilon}_t^2, \quad \hat{\varepsilon}_t = Y_t - \hat{\phi}Y_{t-1}.$$

Question 2

2.1 Maximum Likelihood Estimation

Deriving the Log-Likelihood Function:

The joint probability distribution of the sequence $x = (x_1, x_2, \ldots, x_T)$ is given by:

$$p(x) = p(x_1)\prod_{t=2}^{T} p(x_t|x_1, x_2, \ldots, x_{t-1})$$

Given that each conditional distribution $p(x_t|x_1, x_2, \ldots, x_{t-1})$ is modeled as a Gaussian with mean $\mu_t = f_\theta(x_1, x_2, \ldots, x_{t-1})$ and fixed variance σ^2:

$$p(x_t|x_1, \ldots, x_{t-1}) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x_t - \mu_t)^2}{2\sigma^2}\right)$$

The log-likelihood $L(\theta)$ of the joint distribution $p(x)$ is

$$L(\theta) = \log p(x) = \sum_{t=1}^{T} \log p(x_t|x_1, \ldots, x_{t-1})$$

Substituting the expression for the Gaussian conditional distributions:

$$L(\theta) = \sum_{t=1}^{T}\left(-\frac{1}{2}\log(2\pi\sigma^2) - \frac{(x_t - f_\theta(x_1, \ldots, x_{t-1}))^2}{2\sigma^2}\right)$$

Simplifying constants:

$$L(\theta) = -\frac{T}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}\sum_{t=1}^{T}(x_t - f_\theta(x_1, \ldots, x_{t-1}))^2$$

Computing Gradients Using Backpropagation:

To train the neural network f_θ, we aim to maximize the log-likelihood $L(\theta)$ or equivalently minimize the negative log-likelihood (NLL):

$$NLL(\theta) = -L(\theta) = \frac{1}{2\sigma^2}\sum_{t=1}^{T}(x_t - f_\theta(x_1, \ldots, x_{t-1}))^2 + \text{const}$$

This is equivalent to the mean squared error (MSE) loss scaled by $1/(2\sigma^2)$. The gradient of the NLL with respect to θ is:

$$\nabla_\theta NLL(\theta) = -\frac{1}{\sigma^2}\sum_{t=1}^{T}(x_t - f_\theta(x_1, \ldots, x_{t-1}))\nabla_\theta f_\theta(x_1, \ldots, x_{t-1})$$

Backpropagation Process:

1. **Forward Pass**:
   - For each time step t:
     - Input $x_1, \ldots, x_{t-1}$ into the neural network f_θ to compute $\mu_t = f_\theta(x_1, \ldots, x_{t-1})$.
     - Compute the prediction error $e_t = x_t - \mu_t$.
2. **Loss Computation**:
   - Accumulate the loss over all time steps:

$$\text{Loss} = \frac{1}{2\sigma^2} \sum_{t=1}^{T} e_t^2$$

3. **Backward Pass**:
   - Backpropagate the gradient of the loss with respect to the network parameters θ through f_θ:

$$\nabla_\theta \text{Loss} = -\frac{1}{\sigma^2} \sum_{t=1}^{T} e_t \nabla_\theta f_\theta(x_1, \ldots, x_{t-1})$$

   - Update θ using gradient descent or an appropriate optimization algorithm.

2.2 Predictive Sampling

Sampling Process:
To generate a new sequence $x' = (x'_1, x'_2, \ldots, x'_T)$:

1. **Initialize** x'_1:
   - If $p(x_1)$ is specified, sample $x'_1 \sim p(x_1)$.
   - If not specified, assume $x'_1 \sim N(\mu_0, \sigma^2)$, where $\mu_0 = f_\theta()$.
2. **Iterative Sampling for** $t = 2$ **to** T:
   - Compute Mean:

$$\mu_t = f_\theta(x'_1, x'_2, \ldots, x'_{t-1})$$

   - Sample x'_t:

$$x'_t \sim N(\mu_t, \sigma^2)$$

Influence of Autoregressive Structure:

- Each x'_t depends on all previous samples $x'_1, \ldots, x'_{t-1}$, making the sampling process sequential.
- The autoregressive model captures temporal dependencies by conditioning x'_t on the entire history up to $t - 1$.
- The sequential nature ensures that long-term dependencies can, in theory, influence future samples.

2.3 KL Divergence between AR Models

1. **KL Divergence Definition**:
Given two probability distributions, $p_\theta(x)$ and $q_\phi(x)$, parameterized by two different models (f_θ and g_ϕ), the KL divergence is defined as:

$$D_{KL}(p_\theta(x) \parallel q_\phi(x)) = E_{x \sim p_\theta}\left[\log \frac{p_\theta(x)}{q_\phi(x)}\right]$$

This measures the "distance" between the two distributions $p_\theta(x)$ and $q_\phi(x)$. It quantifies how much information is lost when using q_ϕ to approximate p_θ.
2. **Autoregressive Models Factorization**:
For autoregressive models, the joint probability of the sequence $x = (x_1, \ldots, x_T)$ is factorized as:

- Model 1:

$$p_\theta(x) = p_\theta(x_1) \prod_{t=2}^{T} p_\theta(x_t|x_1, \ldots, x_{t-1})$$

where $p_\theta(x_t|x_1, \ldots, x_{t-1})$ is modeled by the neural network f_θ.

- Model 2:

$$q_\phi(x) = q_\phi(x_1) \prod_{t=2}^{T} q_\phi(x_t|x_1, \ldots, x_{t-1})$$

where $q_\phi(x_t|x_1, \ldots, x_{t-1})$ is modeled by the neural network g_ϕ.

3. **KL Divergence for Autoregressive Models**:

Using the factorized forms of $p_\theta(x)$ and $q_\phi(x)$, the KL divergence between them can be written as:

$$D_{KL}(p_\theta(x) \parallel q_\phi(x)) = E_{x\sim p_\theta}\left[\log \frac{p_\theta(x_1)}{q_\phi(x_1)} + \sum_{t=2}^{T} \log \frac{p_\theta(x_t|x_1, \ldots, x_{t-1})}{q_\phi(x_t|x_1, \ldots, x_{t-1})}\right]$$

4. **KL Divergence for Gaussian Distributions**:

Since both models $p_\theta(x_t|x_1, \ldots, x_{t-1})$ and $q_\phi(x_t|x_1, \ldots, x_{t-1})$ are Gaussian:

$$p_\theta(x_t|x_1, \ldots, x_{t-1}) = \mathcal{N}(f_\theta(x_1, \ldots, x_{t-1}), \sigma_\theta^2)$$

$$q_\phi(x_t|x_1, \ldots, x_{t-1}) = \mathcal{N}(g_\phi(x_1, \ldots, x_{t-1}), \sigma_\phi^2)$$

The KL divergence between two univariate Gaussians $\mathcal{N}(\mu_1, \sigma_1^2)$ and $\mathcal{N}(\mu_2, \sigma_2^2)$ is given by:

$$D_{KL}(\mathcal{N}(\mu_1, \sigma_1^2) \parallel \mathcal{N}(\mu_2, \sigma_2^2)) = \log \frac{\sigma_2}{\sigma_1} + \frac{\sigma_1^2 + (\mu_1 - \mu_2)^2}{2\sigma_2^2} - \frac{1}{2}$$

Thus, for each time step t, the KL divergence between the conditional distributions is:

$$KL(p_\theta(x_t|x_1, \ldots, x_{t-1}) \parallel q_\phi(x_t|x_1, \ldots, x_{t-1})) = \log \frac{\sigma_\phi}{\sigma_\theta} + \frac{\sigma_\theta^2 + (f_\theta(x_1, \ldots, x_{t-1}) - g_\phi(x_1, \ldots, x_{t-1}))^2}{2\sigma_\phi^2} - \frac{1}{2}$$

5. **Full KL Divergence Expression**:

Now, summing over all time steps, the total KL divergence between the two autoregressive models is:

$$D_{KL}(p_\theta(x) \parallel q_\phi(x)) = \sum_{t=1}^{T} E_{x\sim p_\theta}\left[\log \frac{\sigma_\phi}{\sigma_\theta} + \frac{\sigma_\theta^2 + (f_\theta(x_1, \ldots, x_{t-1}) - g_\phi(x_1, \ldots, x_{t-1}))^2}{2\sigma_\phi^2} - \frac{1}{2}\right]$$

6. **Computational Challenges**:

Directly evaluating the KL divergence for high-dimensional sequences presents the following challenges:

- **Expectation over the distribution** p_θ: We need to compute the expectation $E_{x\sim p_\theta}$, which requires generating samples from the autoregressive model $p_\theta(x)$ and averaging the KL divergence over those samples.

- **Sequential Dependence**: In autoregressive models, each sample x_t depends on all previous values, which means generating each sample requires sequential computation. This can become computationally expensive for long sequences.

7. **Approximation Method**:

A practical approach to approximate the KL divergence is to use Monte Carlo sampling:

- Generate K samples $\{x^{(i)}\}_{i=1}^{K}$ from the distribution $p_\theta(x)$.
- Estimate the KL divergence as the empirical average over these samples:

$$D_{KL}(p_\theta(x) \parallel q_\phi(x)) \approx \frac{1}{K} \sum_{i=1}^{K} \left[\log \frac{p_\theta(x^{(i)})}{q_\phi(x^{(i)})}\right]$$

- For each sampled sequence $x^{(i)}$, compute the KL divergence between the two models' outputs at each time step and average over the samples.

2.4 Stationarity and Long-Term Dependencies

Analysis of Long-Term Dependencies:

- **Model Capacity**: The autoregressive model conditions x_t on all previous observations $x_{<t}$, which theoretically allows capturing long-term dependencies.

- **Practical Limitations**: - **Computational Complexity**: Feeding all past observations into f_θ becomes impractical as t increases. - **Vanishing/Exploding Gradients**: Standard feedforward networks struggle with learning long-term dependencies due to gradient issues during backpropagation. - **Fixed Input Size**: Neural networks with fixed-size inputs cannot handle varying-length histories efficiently.

Mathematical Explanation:

- **Inefficient Memory**: The model does not prioritize relevant past information, treating all past observations equally, which dilutes the influence of distant dependencies.

- **Curse of Dimensionality**: As t grows, the input space becomes vast, making learning and generalization difficult.

Suggested Modification: Using Recurrent Neural Networks (RNNs) or Transformers

Recurrent Neural Networks:

- **Architecture**: - Introduce a hidden state h_t that summarizes past information:

$$h_t = \text{RNNCell}(h_{t-1}, x_{t-1})$$

- The conditional mean becomes:

$$\mu_t = f_\theta(h_t)$$

Advantages:

- **Sequential Processing**: Efficiently processes sequences of arbitrary length.

- **Parameter Sharing**: Reuses weights across time steps, reducing the number of parameters.

- **Capturing Long-Term Dependencies**: Designed to maintain information over time, especially with variants like LSTMs or GRUs.

Transformers:

- **Architecture**: - Uses self-attention mechanisms to weigh the importance of past observations:

$$\mu_t = f_\theta(x_{<t}) = \text{Transformer}(x_{<t})$$

- **Advantages**: - **Parallelization**: Processes all positions simultaneously, improving computation speed. - **Attention Mechanism**: Explicitly models dependencies between all pairs of positions, effectively capturing long-term dependencies. - **Positional Encoding**: Incorporates the order of the sequence, essential for temporal data.

Probabilistic Interpretation:

- **Hidden States as Sufficient Statistics**: - The hidden state h_t or attention weights act as sufficient statistics summarizing past information needed to predict x_t.

- **Modified Conditional Distribution**:

$$p(x_t|h_t) = \mathcal{N}(f_\theta(h_t), \sigma^2)$$

- **Efficient Dependency Modeling**: - By summarizing past information, the model focuses on relevant dependencies, enhancing its ability to capture long-term patterns.

Question 3

autoregressive model

$$p(x) = p(x_1)p(x_2|x_{<2})\dots p(x_i|x_{<i})\dots p(x_n|x_{<n}),$$

and Real NADE models the conditional distribution as

$$p(x_1) = \mathcal{N}(x_1|\mu_1, \exp(s_1)),$$

$$\vdots$$

$$p(x_i|x_{<i}; W, c, v_i, b_i, u_i, d_i) = \mathcal{N}\left(x_i \middle| v_i^\top h_i + b_i, \exp\left(u_i^\top h_i + d_i\right)\right),$$

where $h_i, c, v_i, u_i \in R^d$.

Now, we would like to make $p(x_i|x_{<i})$ follow a mixture of Gaussian

$$p(x_i|x_{<i}) = \sum_{c=1}^{C} \pi_i^c \mathcal{N}\left(\mu_i^c, (\sigma_i^c)^2\right),$$

where $\sum_{c=1}^{C} \pi_i^c = 1$.

Now the question is: How do you propose to parameterize $\pi_i^c, \mu_i^c, \sigma_i^c, \forall c \in \{1, \dots, C\}$ as a function of h_i? Describe the parameters required and the total number of parameters required for a single $p(x_i|x_{<i})$.

Answer

We can let

$$\pi_i^c = \frac{\exp\left((g_i^c)^\top h_i\right)}{\sum_{c'=1}^{C} \exp\left((g_i^{c'})^\top h_i\right)}, \quad \forall c \in \{1, \dots, C\}$$

where $g_i^c \in R^d$.

In addition, we have

$$\mu_i^c = (v_i^c)^\top h_i, \quad \sigma_i^c = \exp\left((u_i^c)^\top h_i\right), \quad \forall c \in \{1, \dots, C\}$$

where $h_i \in R^d$, $v_i^c \in R^d$, and $u_i^c \in R^d$.

Therefore, the parameters required for each $p(x_i|x_{<i})$ are:

- For the mixture weights π_i^c: parameters $g_i^c \in R^d$ for each component c, totaling $C \times d$ parameters.
- For the means μ_i^c: parameters $v_i^c \in R^d$ for each component c, totaling $C \times d$ parameters.
- For the standard deviations σ_i^c: parameters $u_i^c \in R^d$ for each component c, totaling $C \times d$ parameters.

Thus, the total number of parameters required for a single $p(x_i|x_{<i})$ is:

$$\text{Total Parameters} = 3 \times C \times d$$

Question 4

4.1 A Quick Warm Up

What is the value of this expectation $E_{x \sim N(0,2)}[x^2 + x + 1]$? How do you propose to estimate this quantity with Monte Carlo estimation?

Answer

We know that for a Gaussian distribution $N(0, 2)$:

$$E[x] = 0, \quad E[x^2] = \text{Var}(x) = 2.$$

So we can compute:

$$E_{x \sim N(0,2)}[x^2 + x + 1] = E[x^2] + E[x] + 1 = 2 + 0 + 1 = 3.$$

To estimate this value using Monte Carlo estimation, we can:

1. Randomly sample K values $x_i \sim N(0, 2)$ for $i = 1, 2, \dots, K$.
2. Compute $y_i = x_i^2 + x_i + 1$ for each sample.
3. Estimate the expectation as:
$$\hat{y} = \frac{1}{K} \sum_{i=1}^{K} y_i.$$

4.2 Variance of K-sample Estimator

What is the variance of the Monte Carlo estimator using K samples?

Answer

Since each y_i is an independent and identically distributed (i.i.d.) random variable, the variance of the estimator $\hat{y}$ is:

$$\mathrm{Var}[\hat{y}] = \frac{1}{K}\,\mathrm{Var}[y_1].$$

First, compute $\mathrm{Var}[y_1]$:

$$\mathrm{Var}[y_1] = E[(y_1 - E[y_1])^2] = E[(x^2 + x + 1 - 3)^2] = E[(x^2 + x - 2)^2].$$

Expand the squared term:

$$(x^2 + x - 2)^2 = x^4 + 2x^3 - 3x^2 - 4x + 4.$$

Compute each expected value:

$$\begin{aligned} E[x^4] &= 3 \times (\mathrm{Var}(x))^2 = 3 \times 2^2 = 12, \\ E[x^3] &= 0, \quad \text{since } x \text{ is symmetric around zero,} \\ E[x^2] &= 2, \\ E[x] &= 0. \end{aligned}$$

Now, compute the variance:

$$\mathrm{Var}[y_1] = 12 + 0 - 3 \times 2 - 0 + 4 = 12 - 6 + 4 = 10.$$

Therefore, the variance of the estimator is:

$$\mathrm{Var}[\hat{y}] = \frac{1}{K} \times 10 = \frac{10}{K}.$$

4.3 Objective Minimization

Now assume we are interested in minimizing the objective:

$$F(\theta) = \sum_{n=1}^{N} w_n f(\theta; n) + \lambda R(\theta), \quad \text{where } w_n > 0, \forall n.$$

What is the asymptotic complexity of evaluating $F(\theta)$ and $\nabla F(\theta)$ given θ (assume the function call and gradient evaluation of $f(\theta; n)$ is constant)? How do you propose to use Monte Carlo estimation to acquire an unbiased estimation of the objective and its gradient?

Answer

Asymptotic Complexity:

- Evaluating $F(\theta)$ requires summing over N terms, so the complexity is $O(N)$. - Computing $\nabla F(\theta)$ also requires summing over N gradients $\nabla f(\theta; n)$, so the complexity is $O(N)$.

Monte Carlo Estimation:

To reduce computational complexity, we can use Monte Carlo estimation to approximate $F(\theta)$ and $\nabla F(\theta)$.

1. **Normalize Weights**:
   Define $W = \sum_{n=1}^{N} w_n$ and probability distribution $p(n) = \frac{w_n}{W}$.

2. **Sample Indices**:
   Sample K indices $\{n_i\}_{i=1}^{K}$ from the categorical distribution $p(n)$.

3. **Estimate Objective**:

   The unbiased estimator of $F(\theta)$ is:

$$\hat{F}(\theta) = W\left(\frac{1}{K}\sum_{i=1}^{K} f(\theta; n_i)\right) + \lambda R(\theta).$$

4. **Estimate Gradient**:

   The unbiased estimator of $\nabla F(\theta)$ is:

$$\nabla \hat{F}(\theta) = W\left(\frac{1}{K}\sum_{i=1}^{K} \nabla f(\theta; n_i)\right) + \lambda \nabla R(\theta).$$

By using a subset of K samples (where $K \ll N$), we reduce the computational complexity to $O(K)$, which is more efficient for large N.
