1. (10 points) **This problem addresses Variational Inference (VI). Answer the following questions:**

   (a) (3 points) **What is ELBO and how is it used in Variational Inference?**

   The ELBO, which stands for Evidence Lower Bound, is the objective function used in Variational Inference (VI) to parametrically approximate an intractable posterior probability over latent variables, $\mathbf{z}$. To do this, we assume a family of tractable posterior distributions as surrogate posteriors. Maximizing the ELBO is equivalent to finding the best approximation of the true posterior distribution within the family of approximating distributions. This is because the ELBO is a lower bound on the log marginal likelihood of the data, and maximizing it means finding the approximation that comes closest to the true posterior in terms of this quantity.

   (b) (4 points) **How can the ELBO be decomposed and what is the intuition behind each term?**

   The ELBO expression is as follows:

$$\begin{aligned}
\mathcal{L}(q,\Theta) &= \mathbb{E}_{z\sim q(z)}\left[\ln p(\mathbf{x},\mathbf{z}|\Theta)\right] - \mathbb{E}_{z\sim q(z|\mathbf{x})}\left[\ln q(\mathbf{z})\right] \\
&= \mathbb{E}_{z\sim q(z)}\left[\ln p(\mathbf{z}|\mathbf{x},\Theta)p(\mathbf{x}|\Theta)\right] - \mathbb{E}_{z\sim q(z|\mathbf{x})}\left[\ln q(\mathbf{z})\right] \\
&= \mathbb{E}_{z\sim q(z)}\left[\ln p(\mathbf{z}|\mathbf{x},\Theta)\right] + \mathbb{E}_{z\sim q(z)}\left[\ln p(\mathbf{x}|\Theta)\right] - \mathbb{E}_{z\sim q(z|\mathbf{x})}\left[\ln q(\mathbf{z})\right] \\
&= \mathbb{E}_{z\sim q(z)}\left[\ln p(\mathbf{x}|\Theta)\right] + \mathbb{E}_{z\sim q(z)}\left[\ln \frac{p(\mathbf{z}|\mathbf{x},\Theta)}{q(\mathbf{z})}\right] \\
&= \mathbb{E}_{z\sim q(z)}\left[\ln p(\mathbf{x}|\Theta)\right] - \mathbb{E}_{z\sim q(z)}\left[\ln \frac{q(\mathbf{z})}{p(\mathbf{z}|\mathbf{x},\Theta)}\right] \\
&= \mathbb{E}_{z\sim q(z)}\left[\ln p(\mathbf{x}|\Theta)\right] - \mathrm{KL}(q \parallel p)
\end{aligned}$$

   The first term is the expected log likelihood of the data under the approximating distribution, and the negative Kullback-Leibler (KL) divergence between the approximating distribution and the prior distribution over the latent variables. The expected log likelihood measures how well the approximating distribution fits the observed data, while the KL divergence measures how much the approximating distribution deviates from the prior. The intuition behind maximizing the ELBO is to find an approximating distribution that fits the data well while staying close to the prior.

   (c) (3 points) **How does the choice of prior distribution affect the value of the ELBO and the resulting approximation of the posterior distribution?**

The choice of prior distribution can affect the value of the ELBO and the resulting approximation of the posterior distribution. If the prior distribution is very different

from the true posterior, then the approximating distribution will need to deviate further from the prior, which may result in a lower ELBO. Conversely, if the prior distribution is similar to the true posterior, then the approximating distribution may not need to deviate as much, which may result in a higher ELBO.

2. (20 points) **Consider a model in which the set of all hidden variables are denoted by z. Let x be a sample observation (single sample, e.g., $x = 1.5$). The data likelihood is defined as**

$$p(x|z) \sim \mathcal{N}\left(x|\mu = z, \sigma^2 = 1\right) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(x-z)^2}{2}\right)$$

and the prior probability on the latent variables is defined as a Gaussian with mean parameter $\mu_0 = 0$ and variance parameter $\sigma_0^2 = 1$,

$$p(z|\mu_0, \sigma_0^2) \sim G(\mu_0 = 0, \sigma_0^2 = 1) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{z^2}{2}\right), \quad z \geq 0$$

Suppose you want to use variational inference with the following proposed family of surrogate posteriors

$$q(z) \sim \text{Exp}(\alpha) = \alpha \exp(-\alpha z), \quad z \geq 0.$$

(a) (15 points) **Write down the expression for the Evidence Lower Bound (ELBO) for this problem. Simplify it as much as possible. Show your work.**

The *Evidence Lowe Bound (ELBO)* is given by:

$$\begin{aligned}
\mathcal{L} &= \mathbb{E}_{z\sim q(z)}[\ln p(z|x) - \ln q(z)] \\
&= \mathbb{E}_{z\sim q(z)}[lnp(z|x)] - \mathbb{E}_{z\sim q(z)}[\ln q(z)] \\
&= \mathbb{E}_{z\sim q(z)}\left[\ln\left(\frac{p(x|z)p(z)}{p(x)}\right)\right] - \mathbb{E}_{z\sim q(z)}[\ln q(z)] \\
&= \mathbb{E}_{z\sim q(z)}[\ln p(x|z)] + \mathbb{E}_{z\sim q(z)}[\ln p(z)] - \mathbb{E}_{z\sim q(z)}[\ln p(x)] - \mathbb{E}_{z\sim q(z)}[\ln q(z)] \\
&= \mathbb{E}_{z\sim q(z)}[\ln p(x|z)] + \mathbb{E}_{z\sim q(z)}[\ln p(z)] - \ln p(x) - \mathbb{E}_{z\sim q(z)}[\ln q(z)] \\
&= \mathbb{E}_{z\sim q(z)}[\ln p(x|z)] + \mathbb{E}_{z\sim q(z)}[\ln p(z)] - \mathbb{E}_{z\sim q(z)}[\ln q(z)] + \text{const.}
\end{aligned}$$

since $p(x)$ is fixed given a dataset and it does not depend on α, we treat it as a constant.

Substituting,

$$
\begin{aligned}
\mathcal{L} &= \mathbb{E}_{z\sim q(z)}[\ln p(x|z)] + \mathbb{E}_{z\sim q(z)}[\ln p(z)] - \ln p(x) - \mathbb{E}_{z\sim q(z)}[\ln q(z)] \\
&= \mathbb{E}_{z\sim q(z)}\left[-\frac{1}{2}\ln(2\pi) - \frac{1}{2}(x-z)^2\right] + \mathbb{E}_{z\sim q(z)}\left[-\frac{1}{2}\ln(2\pi) - \frac{z^2}{2}\right] - \mathbb{E}_{z\sim q(z)}[\ln\alpha - \alpha z] \\
&= -\frac{1}{2}\ln(2\pi) - \frac{1}{2}\mathbb{E}_{z\sim q(z)}[x^2 - 2xz + z^2] - \frac{1}{2}\ln(2\pi) - \frac{1}{2}\mathbb{E}_{z\sim q(z)}[z^2] - \ln\alpha + \alpha\mathbb{E}_{z\sim q(z)}[z] \\
&= -\ln(2\pi) - \frac{x^2}{2} + x\mathbb{E}_{z\sim q(z)}[z] - \mathbb{E}_{z\sim q(z)}[z^2] - \ln\alpha + \alpha\mathbb{E}_{z\sim q(z)}[z] \\
&= -\ln(2\pi) - \ln\alpha - \frac{x^2}{2} + (x-1+\alpha)\mathbb{E}_{z\sim q(z)}[z] - \mathbb{E}_{z\sim q(z)}[z^2]
\end{aligned}
$$

Since $z \sim q(z)$ and $q(z) \sim \text{Exp}(\alpha)$, $z \geq 0$, we know that

$$
\begin{aligned}
\mathbb{E}_{z\sim q(z)}[z] &= \frac{1}{\alpha} \\
\mathbb{E}_{z\sim q(z)}[z^2] &= \frac{2}{\alpha^2}
\end{aligned}
$$

Substituting,

$$
\begin{aligned}
\mathcal{L} &= -\ln(2\pi) - \ln\alpha - \frac{x^2}{2} + (x-1+\alpha)\mathbb{E}_{z\sim q(z)}[z] - \mathbb{E}_{z\sim q(z)}[z^2] \\
&= -\ln(2\pi) - \ln\alpha - \frac{x^2}{2} + (x-1+\alpha)\frac{1}{\alpha} - \frac{2}{\alpha^2} \\
&= -\ln(2\pi) - \ln\alpha - \frac{x^2}{2} + (x-1)\frac{1}{\alpha} + 1 - \frac{2}{\alpha^2}
\end{aligned}
$$

(b) (5 points) **Suppose that you just started data collection, and obtained one data sample thus far, $x = 1.5$. Find the value for α that optimizes the ELBO for this surrogate posterior.**

Optimizing for α:

$$
\begin{aligned}
\frac{\partial\mathcal{L}}{\partial\alpha} &= 0 \\
\iff -\frac{1}{\alpha} - \frac{x-1}{\alpha^2} + \frac{4}{\alpha^3} &= 0 \\
\iff -\alpha^2 + 4 - \alpha(x-1) &= 0 \\
\iff \alpha^2 + \alpha(x-1) - 4 &= 0 \\
\iff \alpha &= -\frac{x-1}{2} \pm \sqrt{\frac{(x-1)^2+8}{2}}
\end{aligned}
$$

Since $\alpha \geq 0$ then we have: $\alpha^* = -\frac{x-1}{2} + \sqrt{\frac{(x-1)^2+8}{2}}$.
Given observed data $x = 1.5$, we find:

$$\alpha^* = -\frac{1.5-1}{2} + \sqrt{\frac{(1.5-1)^2+8}{2}} \approx 1.781$$

3. (15 points) **This problem addresses K-Nearest Neighbors (KNN). Answer the following questions:**

   (a) (3 points) **Suppose you are designing a KNN classifier for 10-dimensional data set where each dimension (or feature) is equally informative for classification but each dimension has a different scale and range. For example, suppose the first feature has a mean of 0.1 with a variance of 0.001 whereas the second feature has a mean of 100 with a variance of 35. Would normalization of the data prior to application of KNN be helpful? Clearly and completely describe why or why not. What sort of distance measure would you use for this data given your response?**

   Yes, if you are using a Euclidean distance then the second feature would outweigh the first feature in the distance computation. This is because Euclidean distance does not account for relative variance and scale during computation.

   (b) (4 points) **Suppose you are training a KNN for a binary classification task. Explain how the weighted voting scheme differs from the majority voting scheme. In which cases would you choose one voting scheme over the other?**

   The weighted distance voting scheme differs from the majority voting scheme in the sense that a neighbor's contribution is weighted by how far it is from the sample candidate.

   In the majority vote, a candidate sample in labeled according to the majority class in its neighborhood (as computed using distance metrics). In other words:

   $$x_i \in C_k \text{ if } \; k = \arg_l \max |N_l(x_i)|, \text{ for } l = 0, 1$$

   where $N_l(x_i)$ is the set of points in class l ($l = 0, 1$) of test point x_i, and $|N_l(x_i)|$ is the cardinality of the set (number of points within it).

   In the weighted vote, each class first computes a score. The candidate is labeled according to the class label with largest score. The score for class l ($l = 0, 1$) is computed as: $s_l = \sum_{x_j \in N_l} \frac{1}{d(x_i, x_j)}$. Then,

   $$x_i \in C_k \text{ if } \; k = \arg_l \max s_l, \text{ for } l = 0, 1$$

   (c) (4 points) **Let C_i represent class i, where $i = 0, 1$, and x an input sample. Is KNN a probabilistic classifier? That is, for any given sample x, can you calculate the posterior probability $P(C_i|\mathbf{x})$? Explain why or why not. If yes, provide the expression for calculating the posterior probability.**

Yes, for any data sample x_i, we can compute the class posterior probability as a function of the number of neighbors around it. The posterior probability for class C_i, $i = 0, 1$, is defined as:

$$P(C_i|x) = \frac{|N_i(x)|}{k}$$

where $|N_i|$ is the cardinality of set $N_i(x)$, and $N_i(x)$ is the set of neighbors of x that belong to class C_i, $i = 0, 1$.

(d) (4 points) **Suppose that you are now working with a regression task. How would you use KNN for a regression task? Explain the equivalent of the weighted/majority vote schemes for regression tasks.**

For a regression task, the voting scheme works in a similar way as in classification, that is, each point will contribute equally to the prediction output. The majority voting scheme computes the absolute average,

$$x_i = \frac{\sum_{j \in N_{x_i}} x_j}{|N_{x_i}|}$$

where N_{x_i} are the k neighbors of x_i.

In the weighted distance voting scheme, we compute the weighted average,

$$x_i = \frac{\sum_{j \in N_{x_i}} w_j x_j}{\sum_{j \in N_{x_i}} w_j}$$

where $w_j = \frac{1}{d(x_i, x_j)}$ and $x_j \in N_{x_i}$.

4. (25 points) **In the paper:**

Pal, N. R., Pal, K., Keller, J. M., and Bezdek, J. C. (2005). *A possibilistic fuzzy c-means clustering algorithm.* IEEE transactions on fuzzy systems, 13(4), 517-530.

The authors define a clustering algorithm (PFCM) that combines Fuzzy C-Means (FCM) and Possibilistic C-Means (PCM). They did this in order to be able to simultaneously estimate both fuzzy membership (u_{ij}) and possibilistic typicality values (t_{ij}) for a data set. The objective function they defined to do this was:

$$J = \sum_{i=1}^{N} \sum_{j=1}^{K} (a u_{ij}^q + b t_{ij}^m) d^2(x_i, \theta_j) + \sum_{j=1}^{K} \gamma_j \sum_{i=1}^{N} (1 - t_{ij})^m$$

where

$$0 \leq u_{ij}, t_{ij} \leq 1, \quad \sum_{j=1}^{K} u_{ij} = 1, \quad m > 1, q > 1, \gamma > 1$$

Solve for the update equations of the memberships and typicality values.
The Lagrangian function for this constrained objective function is:

$$\mathcal{L}(u_{ij}, t_{ij}, \lambda) = \sum_{i=1}^{N}\sum_{j=1}^{K}(au_{ij}^{q} + bt_{ij}^{m})d^2(x_i, \theta_j) + \sum_{j=1}^{K}\gamma_j\sum_{i=1}^{N}(1-t_{ij})^m + \sum_{i=1}^{N}\lambda_i(1-\sum_{j=1}^{K}u_{ij})$$

Solving for the memberships, we find:

$$\frac{\partial \mathcal{L}}{\partial u_{ij}} = 0 \iff aqu_{ij}^{q-1}d^2(x_i, \theta_j) - \lambda_i = 0$$

$$u_{ik} = \left(\frac{\lambda_i}{qd^2(x_i, \theta_j)}\right)^{\frac{1}{q-1}}$$

Since $\sum_{j=1}^{K} u_{ij} = 1$, we find:

$$\begin{aligned}
1 &= \sum_{j=1}^{K}\left(\frac{\lambda_i^{\frac{1}{q-1}}}{(qd^2(x_i, \theta_j))^{\frac{1}{q-1}}}\right) \\
1 &= \lambda_i^{\frac{1}{q-1}}\sum_{j=1}^{K}\left(\frac{1}{(qd^2(x_i, \theta_j))^{\frac{1}{q-1}}}\right) \\
\lambda_i^{\frac{1}{q-1}} &= \left(\sum_{j=1}^{K}\left(\frac{1}{(qd^2(x_i, \theta_j))^{\frac{1}{q-1}}}\right)\right)^{-1} \\
u_{ik} &= \left(\sum_{j=1}^{K}\left(\frac{1}{(qd^2(x_i, \theta_j))^{\frac{1}{q-1}}}\right)\right)^{-1}\left(\frac{1}{qd^2(x_i, \theta_k)}\right)^{\frac{1}{m-1}} \\
u_{ik} &= \frac{1}{\sum_{j=1}^{K}\frac{1}{(qd^2(x_i, \theta_j))^{\frac{1}{q-1}}}}\left(\frac{1}{qd^2(x_i, \theta_k)}\right)^{\frac{1}{q-1}} \\
u_{ik} &= \frac{1}{\sum_{j=1}^{K}\left(\frac{d^2(x_i, \theta_k)}{d^2(x_i, \theta_j)}\right)^{\frac{1}{q-1}}}
\end{aligned}$$

This solution is identical to FCM. Solving for the typicality values, we find:

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial t_{ij}} = 0 &\iff bmt_{ij}^{m-1} d^2(x_i, \theta_j) - \gamma_j m(1 - t_{ij})^{m-1} = 0 \\
&\iff bmd^2(x_i, \theta_j) - \gamma_j m \left(\frac{1}{t_{ij}} - 1 \right) = 0 \\
&\iff \left(\frac{1}{t_{ij}} - 1 \right)^{m-1} = -\frac{bmd^2(x_i, \theta_j)}{m\gamma_j} \\
&\iff t_{ij} = \frac{1}{1 + \left(-\frac{bmd^2(x_i, \theta_j)}{m\gamma_j} \right)^{\frac{1}{m-1}}}
\end{aligned}$$

5. (15 points) **How do Principal Component Analysis (PCA) and Autoencoders (AEs) differ in their approaches to dimensionality reduction? Provide one advantage and one challenge of each method.**

   Principal Component Analysis (PCA) and autoencoders are both techniques used for dimensionality reduction, but they differ in their approach and implementation.

   PCA is a linear transformation technique that seeks to identify the underlying structure in a high-dimensional dataset by projecting the data onto a lower-dimensional subspace while preserving the maximum amount of variance in the original data. PCA works by identifying the principal components of the data, which are the orthogonal vectors that capture the most significant amount of variation in the data. PCA is a widely used technique for dimensionality reduction and has many practical applications in fields like data compression, signal processing, and image processing.

   Autoencoders, on the other hand, are a type of neural network that learns to compress data by mapping it from a high-dimensional space to a lower-dimensional space and then reconstructing the original data from this lower-dimensional representation. Unlike PCA, autoencoders can learn more complex and nonlinear representations of the data. Autoencoders consist of two main parts: an encoder that maps the input data to a compressed representation and a decoder that maps the compressed representation back to the original data. Autoencoders are widely used for tasks like data compression, anomaly detection, and feature extraction.

   Advantages of PCA include its simplicity, computational efficiency, and interpretability. PCA provides a clear and intuitive understanding of the underlying structure of the data and is easy to implement. However, PCA has limitations when dealing with complex, nonlinear data structures, and can struggle to capture subtle patterns and relationships in the data.

   In contrast, autoencoders are more flexible and can learn more complex and nonlinear representations of the data. Autoencoders are particularly useful when dealing with high-dimensional, unstructured data like images or natural language, and can learn to extract meaningful features and patterns from this data. However, autoencoders can be more computationally expensive than PCA and may require more data and training time to achieve optimal performance.

In summary, PCA and autoencoders are both powerful techniques for dimensionality reduction, but they differ in their approach and implementation. PCA is a linear transformation technique that is simple and interpretable but may struggle with complex, nonlinear data structures. Autoencoders are a type of neural network that can learn more complex and nonlinear representations of the data but may be more computationally expensive and require more data and training time.

6. (15 points) **This problem addresses the training of artificial neural networks. Answer the following questions:**

   (a) (5 points) **How does the vanishing gradient problem arise during backpropagation, and what are some techniques to mitigate this problem? How does the exploding gradient problem differ from the vanishing gradient problem, and what are some techniques to address this problem?**

   The vanishing gradient problem arises during backpropagation as the gradients become smaller and smaller as it progresses down to lower layers. Solutions:

   - Weight initialization: proper weight initialization allows for the variance of the output of each layer equal to the variances of its inputs.
   - Activation function: utilizing non-saturating activation functions (ReLU, ELU, SELU, and others) can mitigate vanishing gradients.
   - Batch normalization: learns the optimal scale and mean of each of the layer's inputs.

   Exploding gradients occur when gradients become extremelly large when backpropagated to lower layers, typically arising when non-saturating functios (such as ReLU) are used. Mitigating strategies are similar to those use for vanishing gradients:

   - Weight initialization: proper weight initialization allows for the variance of the output of each layer equal to the variances of its inputs.
   - Batch normalization: learns the optimal scale and mean of each of the layer's inputs.
   - Gradient clipping.
   - Regularization: constraint weight and bias connections with L1 or L2 regularizers.

   (b) (5 points) **How does the Nesterov's momentum differ from the standard momentum? Explain it in words and provide the update equation with Nesterov's momentum.**

   Both the standard momentum and the Nesterov's momentum accelerate learning by adding a momentum vector to the gradient descent delta correction rule. The difference between the two is in the order of when this momentum vector is added.

   First, the gradient descent delta correction rule is defined as:

$$w^{(t+1)} = w^{(t)} + \Delta w^{(t)}$$
$$\Delta w^{(t)} = -\eta \nabla J(w^{(t)})$$

where η is the learning rate and $\nabla J(w^{(t)})$ is the gradient of objective function J at location $w^{(t)}$.

The **standard momentum**, adds a momentum vector directly to the delta correction rule:

$$w^{(t+1)} = w^{(t)} + \Delta w^{(t)}$$
$$\Delta w^{(t)} = -\eta \nabla J(w^{(t)}) + \alpha \Delta w^{(t-1)}$$

where $\alpha = 0.9$.

The **Nesterov's momentum** will first add the previous delta correction vector to $w^{(t)}$, and then compute the gradient vector, that is,

$$w^{(t+1)} = w^{(t)} + \Delta w^{(t)}$$
$$\Delta w^{(t)} = m^{(t)} - \eta \nabla J(m^{(t)})$$
$$m^{(t)} = w^{(t)} + \alpha \Delta w^{(t-1)}$$

where $\alpha = 0.9$.

(c) (5 points) **List at least three advantages of using convolutional layers over dense layers. Justify your choices.**

Convolutional layers are often preferred over dense layers in certain types of tasks because of the following advantages:

- **Parameter efficiency:** Convolutional layers have fewer parameters than dense layers, which means they require less memory and computation to train and run. This is because convolutional layers use shared weights, where a single set of filters is applied to all locations in the input volume, which allows the network to capture local features with fewer parameters.
- **Spatial invariance:** Convolutional layers are designed to be spatially invariant, which means they can recognize the same patterns regardless of their location in the input image. This is accomplished by using filters that slide over the input, producing a feature map that captures the presence of specific patterns at different locations. In contrast, dense layers require the input to be flattened into a vector, losing information about the spatial arrangement of the features.
- **Translation equivariance:** Convolutional layers are also translation equivariant, which means that if the input is translated, the output of the layer is also translated by the same amount. This property is useful in tasks such as object recognition, where the position of the object in the input image can vary. In contrast, dense layers are not translation equivariant, and therefore, they require more training data to learn this property.

Overall, convolutional layers are useful for tasks where the input data has a spatial structure, such as images, and where spatial invariance and translation equivariance

are important. Dense layers, on the other hand, are better suited for tasks where the input data is sequential or has a temporal structure, such as natural language processing or time series analysis.