Autoencoders and Variational Autoencoders

CS 145: Introduction to Data Mining (Spring 2024)

Yanqiao Zhu

UCLA

May 10, 2024

Announcements

- HW3 scores have been released
- HW4 solution has been released
- HW5 due next Friday (May 17)
- Feel free to skip HW5 if you have submitted the first four
- Project proposal due next Monday (May 13)

AdaBoost Algorithm

- Given: $(x_1, y_1), \ldots, (x_N, y_N)$ where $x_i \in X, y_i \in \{-1, +1\}$
- Initialize $D_1(i) = 1/N$
- For $t = 1, \ldots, T$:
  1. Train weak learner $h_t : X \to \{-1, +1\}$ using training data weighted according to the distribution D_t
  2. Measure "goodness" of h_t by its weighted error with respect to D_t:
  $$\epsilon_t = \Pr_{i \sim D_t}[h_t(x_i) \neq y_i] = \sum_{i : h_t(x_i) \neq y_i} D_t(i)$$
  3. Let $\alpha_t = \frac{1}{2} \ln \left(\frac{1-\epsilon_t}{\epsilon_t} \right)$
  4. Update $D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t} & \text{if } y_i = h_t(x_i) \\ e^{\alpha_t} & \text{if } y_i \neq h_t(x_i) \end{cases}$, where Z_t is a normalization factor
- Output the final classifier:
$$H(x) = \operatorname{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x) \right).$$

Insights from the D_{t+1} Update Rule

- The update rule for the weight distribution $D_{t+1}(i)$ in AdaBoost is:

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t} & \text{if } y_i = h_t(x_i) \\ e^{\alpha_t} & \text{if } y_i \neq h_t(x_i) \end{cases}$$

  - It increases the weights of misclassified examples by a factor of e^{α_t} and decreases the weights of correctly classified examples by a factor of $e^{-\alpha_t}$
  - The amount of increase or decrease depends on the performance of the current weak classifier h_t, as measured by its weighted error rate ϵ_t and the resulting value of α_t
  - The normalization factor $\frac{1}{Z_t}$ ensures that D_{t+1} is a valid probability distribution
- This forces the weak learner in the next round to focus more on the examples that were misclassified in the current round, which can lead to improved performance of the final strong classifier

Training Error Theorem

- The training error theorem states that the training error of the final classifier H is at most

$$\exp\left(-2\sum_{t=1}^{T}\gamma_t^2\right)$$

  where $\epsilon_t = \frac{1}{2} - \gamma_t$
- We prove this in three steps:
  1. Show that $D_{T+1}(i) = \frac{1}{N} \cdot \frac{\exp(-y_i f(x_i))}{\prod_t Z_t}$, where $f(x) = \sum_t \alpha_t h_t(x)$
  2. Show that the training error of H is at most $\prod_{t=1}^{T} Z_t$
  3. Compute Z_t and show that $Z_t \leq e^{-2\gamma_t^2}$

Step 1: Recurrence for D_{t+1}

- We can rewrite the update rule for D_{t+1} as:

$$D_{t+1}(i) = \frac{D_t(i)\exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

- Unwrapping this recurrence, we get:

$$\begin{aligned} D_{T+1}(i) &= D_1(i) \cdot \frac{\exp(-\alpha_1 y_i h_1(x_i))}{Z_1} \cdots \frac{\exp(-\alpha_T y_i h_T(x_i))}{Z_T} \\ &= \frac{1}{N} \cdot \frac{\exp(-y_i \sum_t \alpha_t h_t(x_i))}{\prod_t Z_t} \\ &= \frac{1}{N} \cdot \frac{\exp(-y_i f(x_i))}{\prod_t Z_t} \end{aligned}$$

  - $D_1(i) = \frac{1}{N}$ is the initial uniform distribution
  - $\frac{\exp(-\alpha_1 y_i h_1(x_i))}{Z_1}$, $\frac{\exp(-\alpha_T y_i h_T(x_i))}{Z_T}$ are the multiplicative factors from each round of boosting
  - The cumulative effect of these factors gives the exponential term $\exp(-y_i \sum_t \alpha_t h_t(x_i)) = \exp(-y_i f(x_i))$

Step 2: Bounding the Training Error

$$\begin{aligned}
\text{training error}(H) &= \frac{1}{N}\sum_i \begin{cases} 1 & \text{if } y_i \neq H(x_i) \\ 0 & \text{otherwise} \end{cases} && \text{By the definition of training error} \\
&= \frac{1}{N}\sum_i \begin{cases} 1 & \text{if } y_i f(x_i) \leq 0 \\ 0 & \text{otherwise} \end{cases} && \text{Since } H(x) = \operatorname{sign}(f(x)) \text{ and } y_i \in \{-1, +1\} \\
&\leq \frac{1}{N}\sum_i \exp(-y_i f(x_i)) && \text{Since } e^{-z} \geq 1 \text{ if } z \leq 0 \\
&= \sum_i D_{T+1}(i) \prod_t Z_t && \text{Substituted using the result from Step 1} \\
&= \prod_t Z_t && \text{Since } \sum_i D_{T+1}(i) = 1
\end{aligned}$$

Step 3: Computing Z_t

$$
\begin{aligned}
Z_t &= \sum_i D_t(i) \times \begin{cases} e^{-\alpha_t} & \text{if } h_t(x_i) = y_i \\ e^{\alpha_t} & \text{if } h_t(x_i) \neq y_i \end{cases} && \text{By the definition of } Z_t \\
&= \sum_{i:h_t(x_i)=y_i} D_t(i) e^{-\alpha_t} + \sum_{i:h_t(x_i)\neq y_i} D_t(i) e^{\alpha_t} && \text{Splitting the sum} \\
&= e^{-\alpha_t} \sum_{i:h_t(x_i)=y_i} D_t(i) + e^{\alpha_t} \sum_{i:h_t(x_i)\neq y_i} D_t(i) && \text{Factoring out } e^{-\alpha_t} \text{ and } e^{\alpha_t} \\
&= e^{-\alpha_t}(1-\epsilon_t) + e^{\alpha_t}\epsilon_t && \text{By the definition of } \epsilon_t \\
&= 2\sqrt{\epsilon_t(1-\epsilon_t)} && \text{By the choice of } \alpha_t \text{ that minimizes } Z_t \\
&= \sqrt{1-4\gamma_t^2} && \text{Substituting } \epsilon_t = \frac{1}{2} - \gamma_t \\
&\leq e^{-2\gamma_t^2} && \text{Using the inequality } 1 + x \leq e^x \text{ for all real } x
\end{aligned}
$$

Training Error Bound

- Putting it all together, we have:

$$\text{training error}(H) \le \prod_{t=1}^{T} Z_t \le \prod_{t=1}^{T} e^{-2\gamma_t^2} = \exp\left(-2\sum_{t=1}^{T} \gamma_t^2\right)$$

This concludes the proof of the training error theorem for AdaBoost

The Role of the Weak Learning Assumption

- The training error theorem relies on the "weak learning assumption":
  - Each weak classifier h_t should have a weighted error ϵ_t slightly less than $\frac{1}{2}$
  - Equivalently, each h_t should have a small positive edge γ_t over random guessing
- If this assumption holds, the training error theorem guarantees that AdaBoost will produce a highly accurate final classifier
- However, if the weak classifiers are too weak (i.e., $\gamma_t \approx 0$), the training error may decrease very slowly with T. In this case, a large number of boosting rounds may be needed to achieve good performance
- On the other hand, if the weak classifiers are too strong (i.e., $\gamma_t \approx \frac{1}{2}$), AdaBoost may overfit to the training data. Techniques like early stopping or regularization can help prevent overfitting in this case

Example Midterm Problems: True/False Questions

1. Logistic regression can learn non-linear decision boundaries by using polynomial features.
   - ✓ True
   - ☐ False
2. In K-Means clustering, increasing the number of clusters always leads to better performance.
   - ☐ True
   - ✓ False

   While increasing the number of clusters can reduce the within-cluster variance, it may also lead to overfitting and poor generalization.

Example Midterm Problems: Multiple Choice Questions

1. Which of the following is **not** an advantage of using decision trees for classification?
   - ☐ They are easy to interpret and visualize.
   - ☐ They can handle both categorical and numerical features.
   - ✓ They are robust to outliers and irrelevant features.
   - ☐ They can learn non-linear decision boundaries.

   Decision trees are sensitive to outliers and irrelevant features, which can lead to overfitting. The other options are true advantages of decision trees.

2. Which of the following is true about the bias-variance tradeoff?
   - ☐ High bias and high variance models are preferred.
   - ✓ Increasing model complexity typically reduces bias but increases variance.
   - ☐ Regularization techniques can help reduce bias.
   - ☐ The tradeoff is not affected by the size of the training dataset.

   Increasing model complexity typically reduces bias but increases variance. Regularization helps reduce variance, and larger datasets can help reduce both bias and variance.

Example Midterm Problems: Fill-in-the-Blank

Unsupervised Learning

Fill in the table to compare K-Means and GMMs:

Property	K-Means	GMM
Can handle clusters with different sizes and densities	×	✓
Learns a generative model of the data	×	✓
Uses the EM algorithm for training	×	✓

Example Midterm Problems: Open-Answer Questions

Gaussian Mixture Models

Consider a Gaussian Mixture Model (GMM) with K components, where each component has a diagonal covariance matrix. Let π_k denote the mixing weight, $\boldsymbol{\mu}_k$ denote the mean vector, and σ_k^2 denote the diagonal elements of the covariance matrix for component k.

1. Write down the expression for the log-likelihood of this GMM.
2. Derive the update equations for the E-step and M-step of the EM algorithm for fitting this GMM.
3. How does the diagonal covariance assumption affect the complexity of the model and the EM algorithm, compared to a full covariance GMM?

Autoencoders

- Unsupervised learning models that learn efficient data representations
- Consist of an encoder and a decoder:
  - Encoder: Maps input data to a lower-dimensional latent space
  - Decoder: Reconstructs the input data from the latent representation
- Training objective: Minimize reconstruction loss between input and output

$$\mathcal{L}_{\mathsf{AE}}(\boldsymbol{\theta}; \boldsymbol{x}) = \|\boldsymbol{x} - \mathsf{Decoder}_{\boldsymbol{\theta}}(\mathsf{Encoder}_{\boldsymbol{\theta}}(\boldsymbol{x}))\|^2$$

Variational Autoencoders (VAEs)

- Deep latent variable models that learn a joint distribution over observed and latent variables
- Approximate the intractable posterior using a variational distribution
- Maximize the Evidence Lower Bound (ELBO):

$$\mathcal{L}(\boldsymbol{x};\boldsymbol{\theta},\boldsymbol{\phi}) = \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}[\log p(\boldsymbol{x} \mid \boldsymbol{z};\boldsymbol{\theta})] - D_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\boldsymbol{z} \mid \boldsymbol{x}) \parallel p(\boldsymbol{z}))$$

Derivation of the Evidence Lower Bound (ELBO)

- Start with the log-likelihood of a data point $\boldsymbol{x}$:

$$\log p(\boldsymbol{x}) = \log \int p(\boldsymbol{x}, \boldsymbol{z}) \mathrm{d}\boldsymbol{z} = \log \int p(\boldsymbol{x} \mid \boldsymbol{z}) p(\boldsymbol{z}) \mathrm{d}\boldsymbol{z}$$

- Introduce a variational distribution $q(\boldsymbol{z} \mid \boldsymbol{x})$ and use Jensen's inequality:

$$\begin{aligned}
\log p(\boldsymbol{x}) &= \log \int q(\boldsymbol{z} \mid \boldsymbol{x}) \frac{p(\boldsymbol{x} \mid \boldsymbol{z}) p(\boldsymbol{z})}{q(\boldsymbol{z} \mid \boldsymbol{x})} \mathrm{d}\boldsymbol{z} \\
&\geq \int q(\boldsymbol{z} \mid \boldsymbol{x}) \log \frac{p(\boldsymbol{x} \mid \boldsymbol{z}) p(\boldsymbol{z})}{q(\boldsymbol{z} \mid \boldsymbol{x})} \mathrm{d}\boldsymbol{z} \\
&= \int q(\boldsymbol{z} \mid \boldsymbol{x}) \log p(\boldsymbol{x} \mid \boldsymbol{z}) \mathrm{d}\boldsymbol{z} - \int q(\boldsymbol{z} \mid \boldsymbol{x}) \log \frac{q(\boldsymbol{z} \mid \boldsymbol{x})}{p(\boldsymbol{z})} \mathrm{d}\boldsymbol{z} \\
&= \mathbb{E}_{q(\boldsymbol{z} \mid \boldsymbol{x})} [\log p(\boldsymbol{x} \mid \boldsymbol{z})] - D_{\mathrm{KL}}(q(\boldsymbol{z} \mid \boldsymbol{x}) \parallel p(\boldsymbol{z}))
\end{aligned}$$

- The final expression is the Evidence Lower Bound (ELBO)
- We use $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ to denote the parameters of the generative model $p_{\boldsymbol{\theta}}(\boldsymbol{x} \mid \boldsymbol{z})$ and the variational distribution $q_{\boldsymbol{\phi}}(\boldsymbol{z} \mid \boldsymbol{x})$, respectively

Reparameterization Trick

- We want to compute the gradient of the expected value of a function $f(\boldsymbol{z})$ with respect to the parameters ϕ of the distribution $q_\phi(\boldsymbol{z})$:

$$\nabla_\phi \mathbb{E}_{q_\phi(\boldsymbol{z})}[f(\boldsymbol{z})] = \nabla_\phi \int q_\phi(\boldsymbol{z}) f(\boldsymbol{z}) \mathrm{d}\boldsymbol{z}$$

- The reparameterization trick expresses sampling from the variational distribution as a deterministic function of parameters and a random variable:

$$\boldsymbol{z} = g_\phi(\boldsymbol{\epsilon}), \quad \boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon})$$

  where $p(\boldsymbol{\epsilon})$ is a simple distribution, such as $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$
- The gradient can be computed as:

$$\begin{aligned}\nabla_\phi \mathbb{E}_{q_\phi(\boldsymbol{z})}[f(\boldsymbol{z})] &= \nabla_\phi \mathbb{E}_{p(\boldsymbol{\epsilon})}[f(g_\phi(\boldsymbol{\epsilon}))] \\ &= \mathbb{E}_{p(\boldsymbol{\epsilon})}[\nabla_\phi f(g_\phi(\boldsymbol{\epsilon}))]\end{aligned}$$

- Allows gradients to flow through the sampling process and enables efficient training of VAEs using stochastic gradient descent

VAE Architecture

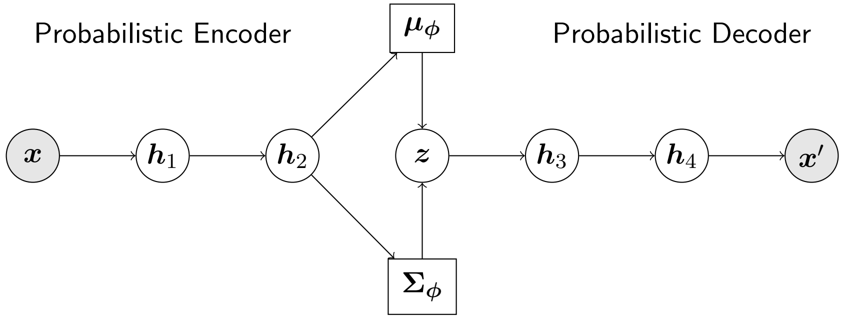

Figure: Variational Autoencoder (VAE) architecture

VAE as an Autoencoder

- VAEs can be seen as a probabilistic autoencoder:
  - Encoder: $q_{\boldsymbol{\phi}}(\boldsymbol{z} \mid \boldsymbol{x})$ maps input $\boldsymbol{x}$ to a distribution over latent codes $\boldsymbol{z}$
  - Decoder: $p_{\boldsymbol{\theta}}(\boldsymbol{x} \mid \boldsymbol{z})$ reconstructs $\boldsymbol{x}$ from a sample $\boldsymbol{z} \sim q_{\boldsymbol{\phi}}(\boldsymbol{z} \mid \boldsymbol{x})$
- The ELBO objective:

$$\begin{aligned}\mathcal{L}(\boldsymbol{x}; \boldsymbol{\theta}, \boldsymbol{\phi}) &= \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}[\log p(\boldsymbol{z}, \boldsymbol{x}; \boldsymbol{\theta}) - \log q_{\boldsymbol{\phi}}(\boldsymbol{z} \mid \boldsymbol{x}))] \\ &= \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}[\log p(\boldsymbol{x} \mid \boldsymbol{z}; \boldsymbol{\theta})] - D_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\boldsymbol{z} \mid \boldsymbol{x}) \parallel p(\boldsymbol{z}))\end{aligned}$$

  - Good reconstruction: $\mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}[\log p(\boldsymbol{x} \mid \boldsymbol{z}; \boldsymbol{\theta})]$ should be high
  - Regularization: $D_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\boldsymbol{z} \mid \boldsymbol{x}) \parallel p(\boldsymbol{z}))$ encourages the posterior to be close to the prior

Information-Theoretic Interpretation of VAE Objective

- The VAE objective can be interpreted as maximizing the mutual information between the latent variables $\boldsymbol{z}$ and the observed variables $\boldsymbol{x}$:

$$\begin{aligned}\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathcal{D}) &= \sum_{\boldsymbol{x} \in \mathcal{D}} \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}[\log p(\boldsymbol{x} \mid \boldsymbol{z}; \boldsymbol{\theta})] - D_{\mathrm{KL}}(q_{\phi}(\boldsymbol{z} \mid \boldsymbol{x}) \parallel p(\boldsymbol{z})) \\ &= \sum_{\boldsymbol{x} \in \mathcal{D}} \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}[\log p(\boldsymbol{x} \mid \boldsymbol{z}; \boldsymbol{\theta})] - D_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\boldsymbol{z} \mid \boldsymbol{x}) \parallel p(\boldsymbol{z} \mid \boldsymbol{x})) + I_q(\boldsymbol{x}; \boldsymbol{z})\end{aligned}$$

  where $I_q(\boldsymbol{x}; \boldsymbol{z}) = \mathbb{E}_{p(\boldsymbol{x})}[D_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\boldsymbol{z} \mid \boldsymbol{x}) \parallel p(\boldsymbol{z}))]$ is the mutual information between $\boldsymbol{x}$ and $\boldsymbol{z}$ under the variational distribution $q_{\boldsymbol{\phi}}(\boldsymbol{z} \mid \boldsymbol{x})$
- Maximizing the ELBO is equivalent to:
  - Maximizing the likelihood of the data under the generative model (reconstruction term)
  - Minimizing the KL divergence between the variational posterior and the true posterior (inference term)
  - Maximizing the mutual information between the latent variables and the observed variables (compression term)
- The VAE objective: Learn a compact and informative latent representation of the data

Summary

- Autoencoders learn efficient data representations by minimizing reconstruction loss
- VAEs are probabilistic autoencoders that learn a joint distribution over observed and latent variables
- VAEs maximize the ELBO, balancing reconstruction accuracy and regularization of the posterior
- The VAE objective can be interpreted as maximizing the mutual information between latent and observed variables
- The reparameterization trick enables efficient training of VAEs using stochastic gradient descent