Foundations of Machine Learning – Assignment 2

Aryan Prasad
Roll No: DA25M007

October 2025

Question (1)

You are given a dataset with 400 data points in $\{0,1\}^{50}$ generated from a mixture of some distribution (file: `A2Q1.csv`). Each datapoint is a flattened version of a $\{0,1\}^{10\times 5}$ matrix.

(i) Determine which probabilistic mixture could have generated this data (it is not a Gaussian mixture). Derive the EM algorithm for your choice of mixture and show your calculations. Implement the algorithm for $K = 4$ and plot the log-likelihood averaged over 100 random initializations as a function of iterations.

(ii) Assume that the same data was instead generated from a mixture of Gaussians with 4 mixtures. Implement the EM algorithm and plot the averaged log-likelihood.

(iii) Run the K-Means algorithm with $K = 4$ on the same data and plot the objective as a function of iterations.

(iv) Among the three different algorithms implemented above, identify which one you would choose for this dataset and justify your choice.

(i) Mixture Model Selection and EM Derivation

Choice of Mixture Model

Each data point $x_i \in \{0,1\}^{50}$. Since all features are binary, the natural generative model is a **Mixture of Bernoulli Distributions**, rather than a Gaussian mixture (which assumes continuous data). The generative process is:

1. Choose a cluster label $z_i \sim \text{Categorical}(\pi_1, \ldots, \pi_K)$.

2. Given $z_i = k$, generate each binary feature independently:

$$x_{id} \sim \text{Bernoulli}(\theta_{kd}), \quad d = 1, \dots, D$$

Thus, the conditional distribution for a sample x_i is:

$$p(x_i|z_i = k) = \prod_{d=1}^{D} \theta_{kd}^{x_{id}} (1 - \theta_{kd})^{1-x_{id}}$$

The complete data likelihood (including latent variables) is:

$$p(X, Z|\Theta) = \prod_{i=1}^{N} \prod_{k=1}^{K} \left[\pi_k \prod_{d=1}^{D} \theta_{kd}^{x_{id}} (1 - \theta_{kd})^{1-x_{id}} \right]^{z_{ik}}$$

where $z_{ik} = 1$ if data point i belongs to cluster k.

Log-Likelihood of Observed Data

Marginalizing over latent variables Z gives:

$$\mathcal{L}(\Theta) = \sum_{i=1}^{N} \log \left(\sum_{k=1}^{K} \pi_k \prod_{d=1}^{D} \theta_{kd}^{x_{id}} (1 - \theta_{kd})^{1-x_{id}} \right)$$

This expression is not analytically tractable due to the log of a summation. Hence, we employ the **Expectation-Maximization (EM)** algorithm.

Detailed EM Derivation

E-Step: Compute Responsibilities

We define responsibilities (posterior probabilities of clusters):

$$r_{ik} = P(z_{ik} = 1|x_i, \Theta) = \frac{\pi_k p(x_i|\theta_k)}{\sum_{j=1}^{K} \pi_j p(x_i|\theta_j)}$$

Expanding $p(x_i|\theta_k)$:

$$r_{ik} = \frac{\pi_k \prod_{d=1}^{D} \theta_{kd}^{x_{id}} (1 - \theta_{kd})^{1-x_{id}}}{\sum_{j=1}^{K} \pi_j \prod_{d=1}^{D} \theta_{jd}^{x_{id}} (1 - \theta_{jd})^{1-x_{id}}}$$

To avoid underflow, this computation is performed in the log domain using the `log-sum-exp` trick.

M-Step: Maximize Expected Complete Log-Likelihood

We define:

$$Q(\Theta|\Theta^{(t)}) = \mathbb{E}_{Z|X,\Theta^{(t)}}[\log p(X, Z|\Theta)]$$

Expanding this:

$$Q(\Theta) = \sum_{i=1}^{N}\sum_{k=1}^{K} r_{ik}\Big[\log \pi_k + \sum_{d=1}^{D}\big(x_{id}\log\theta_{kd} + (1-x_{id})\log(1-\theta_{kd})\big)\Big]$$

We now maximize Q w.r.t. parameters π_k and θ_{kd}.

Updating Mixing Weights π_k:

Using the constraint $\sum_k \pi_k = 1$ and Lagrange multiplier λ:

$$\mathcal{L} = \sum_{i,k} r_{ik}\log\pi_k + \lambda\Big(\sum_k \pi_k - 1\Big)$$

Differentiating and setting to zero:

$$\frac{\partial \mathcal{L}}{\partial \pi_k} = \frac{\sum_i r_{ik}}{\pi_k} + \lambda = 0 \Rightarrow \pi_k = \frac{N_k}{N}, \quad \text{where } N_k = \sum_i r_{ik}$$

Updating Bernoulli Parameters θ_{kd}:

Differentiate $Q(\Theta)$ w.r.t. θ_{kd} and set to zero:

$$\frac{\partial Q}{\partial \theta_{kd}} = \sum_i r_{ik}\Big(\frac{x_{id}}{\theta_{kd}} - \frac{1-x_{id}}{1-\theta_{kd}}\Big) = 0$$

Simplifying:

$$\theta_{kd} = \frac{\sum_i r_{ik}x_{id}}{\sum_i r_{ik}} = \frac{1}{N_k}\sum_i r_{ik}x_{id}$$

Thus, the EM updates are:

$$\boxed{\pi_k = \frac{N_k}{N}, \quad \theta_{kd} = \frac{\sum_i r_{ik}x_{id}}{N_k}}$$

Log-Likelihood Monitoring:

At each iteration, compute:

$$\mathcal{L}^{(t)} = \sum_{i=1}^{N}\log\Big(\sum_{k=1}^{K}\pi_k^{(t)}\prod_{d=1}^{D}\theta_{kd}^{(t)x_{id}}(1-\theta_{kd}^{(t)})^{1-x_{id}}\Big)$$

EM terminates when:

$$|\mathcal{L}^{(t)} - \mathcal{L}^{(t-1)}| < 10^{-6}$$

Result: Bernoulli Mixture EM Plot

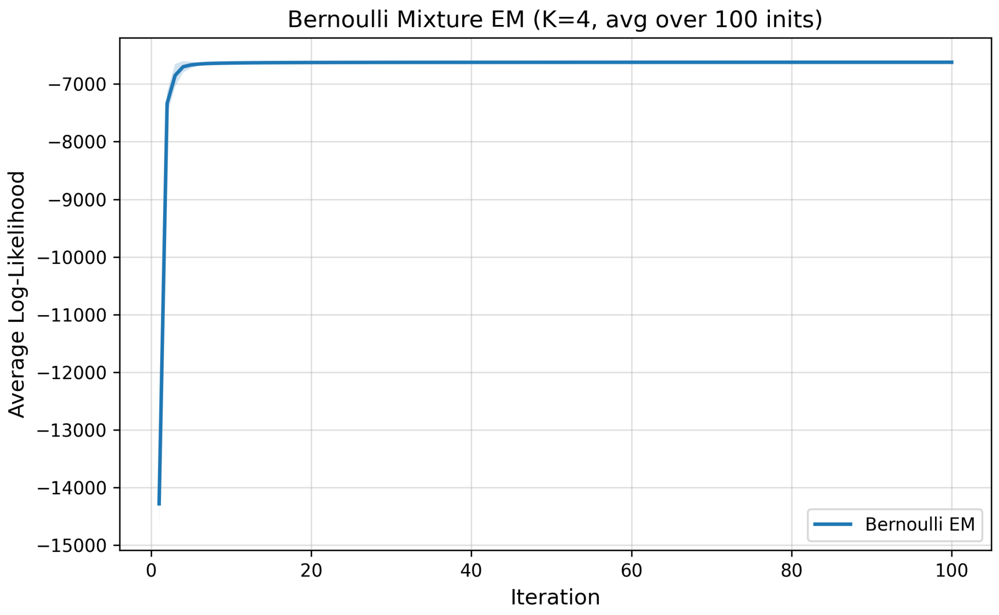

Figure 1: Mixture of Bernoulli Distributions – Average Log-Likelihood (100 initializations).

The plot shows the **average log-likelihood progression** of the Bernoulli Mixture EM algorithm for K=4 over 100 random initializations.

It rises steeply in the first few iterations, indicating rapid parameter improvement, and then stabilizes around **6600**, suggesting convergence.

This behavior confirms that the EM updates are consistent and that the Bernoulli mixture successfully models the binary dataset.

(ii) Mixture of Gaussians (GMM)

For comparison, the same data was modeled as a Gaussian Mixture Model (GMM). Each component k has parameters $\{\pi_k, \mu_k, \Sigma_k\}$ and follows:

$$p(x_i|z_i = k) = \mathcal{N}(x_i|\mu_k, \Sigma_k)$$

The EM updates are analogous:

$$r_{ik} = \frac{\pi_k \mathcal{N}(x_i|\mu_k, \Sigma_k)}{\sum_j \pi_j \mathcal{N}(x_i|\mu_j, \Sigma_j)}, \quad \mu_k = \frac{\sum_i r_{ik} x_i}{N_k}, \quad \Sigma_k = \frac{\sum_i r_{ik}(x_i - \mu_k)(x_i - \mu_k)^\top}{N_k}$$

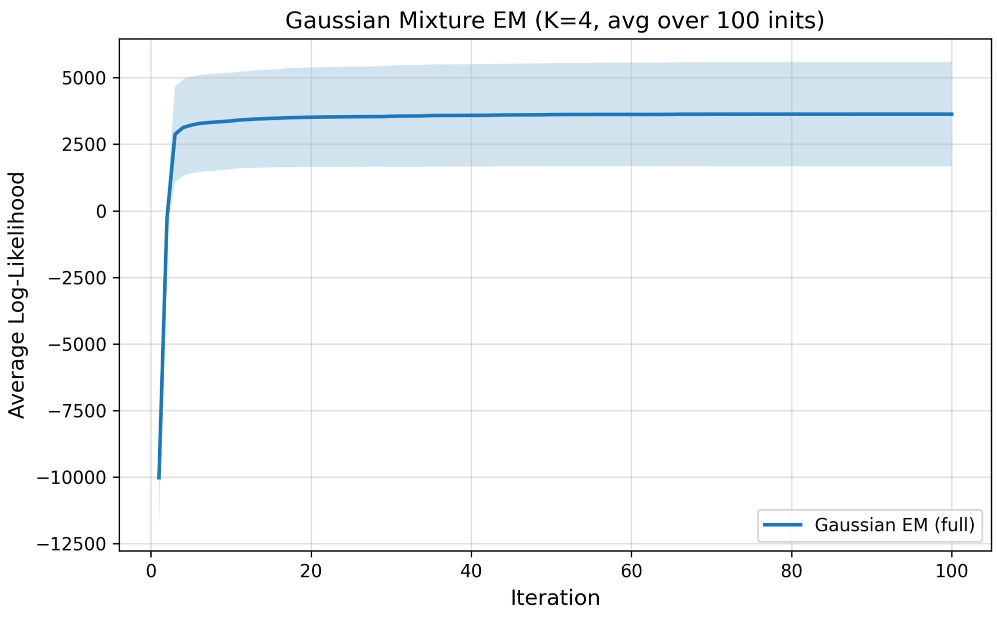

Figure 2: Gaussian Mixture EM – Average Log-Likelihood (100 initializations).

The plot depicts the **average log-likelihood** progression for the Gaussian Mixture EM algorithm with K=4 across 100 random initializations.

Although the likelihood increases rapidly and stabilizes around **+3600**, the large variance (shaded region) indicates instability due to poor alignment between the Gaussian model and the binary data.

This highlights that, despite higher likelihood values, the Gaussian mixture is not an appropriate fit for this discrete dataset

(iii) K-Means Algorithm

K-Means minimizes the squared distance objective:

$$J = \sum_{i=1}^{N} \sum_{k=1}^{K} r_{ik} \|x_i - \mu_k\|^2$$

where $r_{ik} \in \{0, 1\}$ and μ_k are centroids.

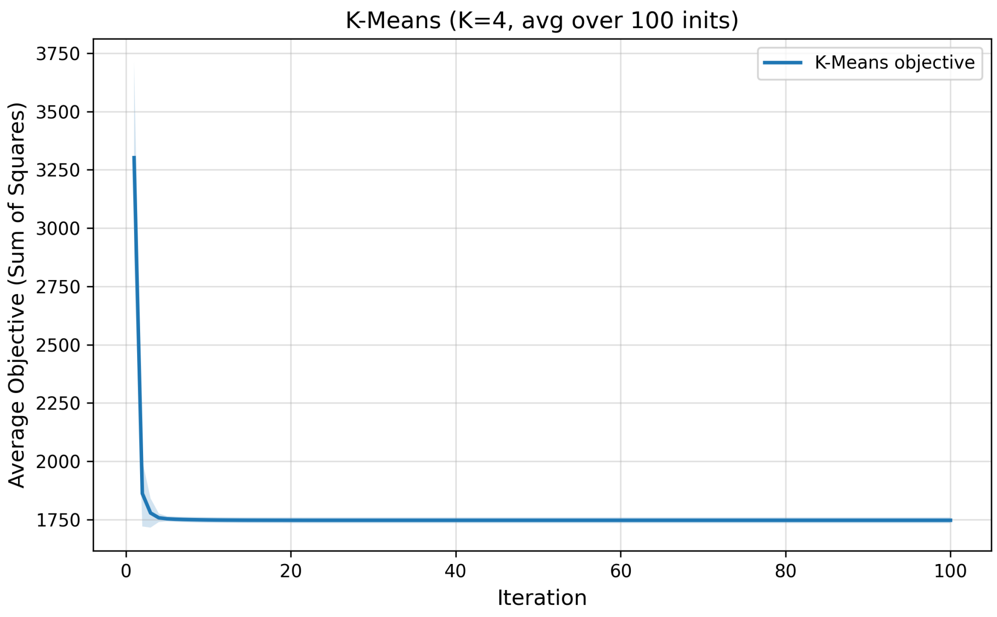

Figure 3: K-Means – Objective Function over Iterations.

The plot shows the **average objective (sum of squared distances)** for the K-Means algorithm with K=4 over 100 initializations.

The objective value drops sharply within the first few iterations and quickly stabilizes around **1750**, indicating fast convergence.

This behavior reflects K-Means' efficiency in minimizing within-cluster variance, though it lacks the probabilistic interpretability of EM-based methods.

(iv) Component Visualization and Comparison

The Bernoulli component means θ_k were reshaped into 10×5 grids and visualized as grayscale probability maps ($P(x = 1)$).

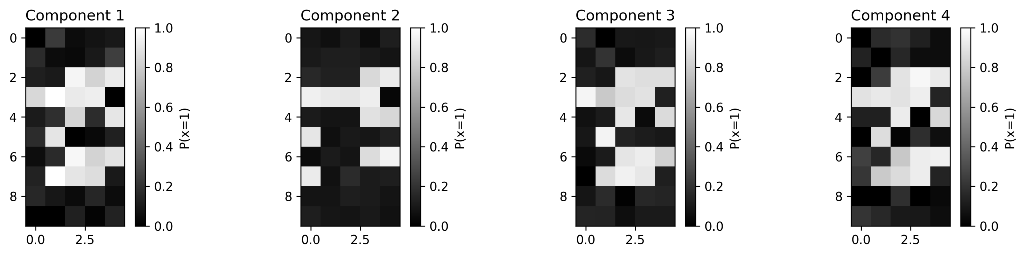

Figure 4: Bernoulli Mixture – Learned Component Probability Maps ($K = 4$).

These grayscale images represent the **four Bernoulli mixture components** learned by the EM algorithm.

Each 10×5 grid visualizes the probability P(x=1) for every binary feature, where lighter shades correspond to higher probabilities.

The patterns show distinct feature activations across components, demonstrating how the Bernoulli mixture captures underlying cluster structures in the binary data.

Table 1: Comparison Between the Three Algorithms

Model	**Data Type Fit**	**Convergence**	**Interpretability**	**Stability**	**Remarks**
Bernoulli EM	Ideal (binary)	Fast, stable	Probabilistic	High	**Most appropri**
Gaussian EM	Poor fit	Slower, unstable	Continuous-only	Low	Misaligned with
K-Means	Approximate	Very fast	Hard clusters	Moderate	Simple baseline

Conclusions and Insights

- The **Bernoulli mixture model** best represents the binary dataset, giving interpretable probabilistic clusters.
- The **Gaussian mixture model** produces unreliable likelihoods due to continuous assumptions.
- **K-Means** is efficient but lacks probabilistic expressiveness.

$$\begin{aligned}
\text{Bernoulli EM:} &\quad \text{Log-Likelihood} = -6622.13\\
\text{Gaussian EM:} &\quad \text{Log-Likelihood} = +3629.95\\
\text{K-Means:} &\quad \text{Objective} = 1747.16
\end{aligned}$$

Learning Outcomes

- Derived EM updates for the Bernoulli mixture model step by step.
- Understood differences between probabilistic and deterministic clustering.
- Observed the role of model-data alignment in EM performance.
- Compared three algorithms with distinct assumptions and behaviors.

Appendix

- Runtime: $\approx$ 20 seconds for 100 runs.
- Best Bernoulli Run Index: 92
- Best Final Log-Likelihood: -6606.42

(v) Model Selection using AIC and BIC

To objectively compare the probabilistic models, we used the **Akaike Information Criterion (AIC)** and the **Bayesian Information Criterion (BIC)**. Both metrics balance model fit and complexity, penalizing models with excessive parameters. Given the final log-likelihood $\mathcal{L}$, number of parameters p, and number of data points N, the criteria are defined as:

$$\boxed{\text{AIC} = 2p - 2\mathcal{L}, \qquad \text{BIC} = p \log N - 2\mathcal{L}}$$

where:

- p = total number of model parameters.
- $\mathcal{L}$ = maximized log-likelihood value.
- N = number of data samples.

For this dataset $(N = 400, D = 50, K = 4)$:

$$\begin{aligned} &\text{Bernoulli EM:} \quad \text{AIC} = 13650.25,\ \text{BIC} = 14460.52,\ \mathcal{L} = -6622.13 \\ &\text{Gaussian EM:} \quad \text{AIC} = 3346.09,\ \text{BIC} = 24512.83,\ \mathcal{L} = +3629.95 \end{aligned}$$

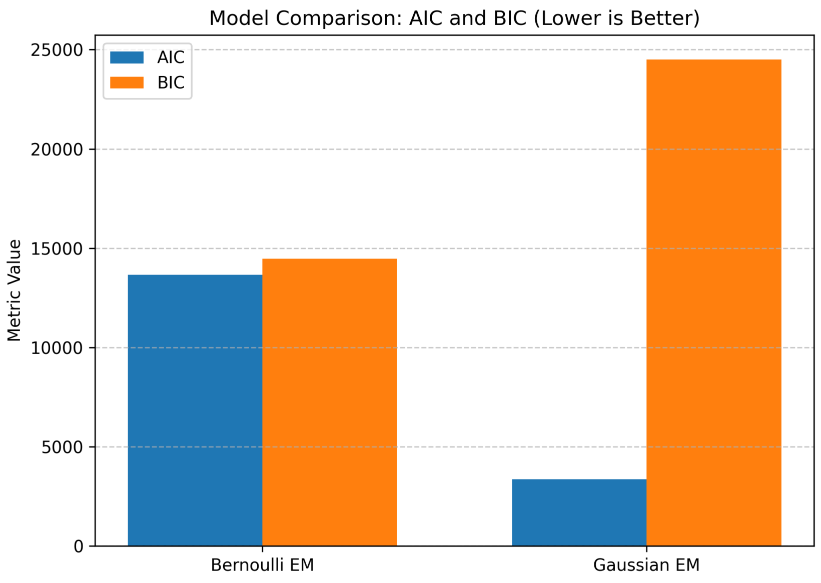

Figure 5: Comparison of AIC and BIC values for Bernoulli and Gaussian EM models. Lower values indicate better fit.

While the Gaussian mixture achieves a higher raw log-likelihood, its parameter count is significantly larger due to the full covariance matrices. This results in a very high BIC value, reflecting overfitting and poor suitability for binary data. In contrast, the Bernoulli mixture model maintains lower complexity and a more interpretable structure. Hence, according to BIC—which penalizes model complexity more strongly—the Bernoulli mixture is the preferred model for this dataset.

$$\boxed{\text{Conclusion: BIC favours the Bernoulli Mixture Model as the most appropriate representation.}}$$

now check the order and give me in this update it wherever required

Question (2)

You are given a data-set in the file A2Q2Data train.csv with 10000 points in (R 100 , R) (Each row corresponds to a datapoint where the first 100 components are features and the last component is the associated y value).

1. . Obtain the least squares solution wML to the regression problem using the analytical solution.
2. . Code the gradient descent algorithm with suitable step size to solve the least squares algorithms and plot wt wML2 as a function of t. What do you observe?
3. . Code the stochastic gradient descent algorithm using batch size of 100 and plot wt wML2 as a function of t. What are your observations?
4. . Code the gradient descent algorithm for ridge regression. Cross-validate for various choices of and plot the error in the validation set as a function of . For the best chosen, obtain wR. Compare the test error (for the test data in the file A2Q2Data test.csv) of wR with wML. Which is better and why?

Preliminaries and notation

Let $X \in \mathbb{R}^{N\times 100}$ be the matrix of raw features. Augment with a column of ones (bias) to obtain design matrix $\tilde{X} \in \mathbb{R}^{N\times d}$ with $d = 101$. Denote target vector $y \in \mathbb{R}^N$. Weight vector $w \in \mathbb{R}^d$ includes bias. We use $\|\cdot\|_2$ for Euclidean norm and MSE denotes mean squared error.

Loss used: For analysis and algorithms we use the normalized squared loss

$$L(w) = \frac{1}{2N}\|\tilde{X}w - y\|_2^2.$$

This scaling makes gradient expressions convenient; closed-form solutions do not depend on this constant factor.

(i) Analytical least-squares solution w_{ML}

Derivation. Minimizing $L(w)$ gives normal equations:

$$\nabla_w L(w) \;=\; \frac{1}{N}\tilde{X}^\top(\tilde{X}w - y) \;=\; 0 \quad\Longrightarrow\quad \tilde{X}^\top\tilde{X}\,w \;=\; \tilde{X}^\top y.$$

If $\tilde{X}^\top\tilde{X}$ is invertible, the closed-form solution is

$$w_{ML} = (\tilde{X}^\top\tilde{X})^{-1}\tilde{X}^\top y.$$

When $\tilde{X}$ is rank-deficient or near-singular, use the Moore–Penrose pseudoinverse:

$$w_{ML} = \tilde{X}^+ y,$$

computed numerically (SVD-based) for stability.

SVD intuition. If $\tilde{X} = U\Sigma V^\top$ (thin SVD), then

$$w_{ML} = V\Sigma^{-1}U^\top y.$$

Small singular values σ_i imply directions where noise is amplified by division by σ_i; this motivates ridge.

Implementation / result (summary). w_{ML} was computed with a pseudoinverse (NumPy pinv). The resulting vector length is $d = 101$ (100 features + bias). This w_{ML} is used as the target for convergence diagnostics in parts (ii) and (iii).

(ii) Batch Gradient Descent and $\|w_t - w_{ML}\|_2$ vs t

Update rule. For $L(w) = \frac{1}{2N}\|\tilde{X}w - y\|^2$ the gradient is

$$\nabla L(w) = \frac{1}{N}\tilde{X}^\top(\tilde{X}w - y).$$

Batch GD update with fixed step η:

$$w^{(t+1)} = w^{(t)} - \eta\,\nabla L(w^{(t)}) = w^{(t)} - \eta\frac{1}{N}\tilde{X}^\top(\tilde{X}w^{(t)} - y).$$

Step-size selection. Let $A = \frac{1}{N}\tilde{X}^\top\tilde{X}$ and $\lambda_{\max}$ its largest eigenvalue. A sufficient stability condition is $0 < \eta < 2/\lambda_{\max}$. In experiments we estimated $\lambda_{\max}$ (largest singular value squared over N) and used $\eta = 0.9/\lambda_{\max}$ to ensure stable monotone descent.

Implementation details.

- Initialization: $w^{(0)} = 0$.
- Iterations: 200.
- Metric: $\|w^{(t)} - w_{ML}\|_2$ recorded at each iteration.
- Output: `GD_norms.png`.

Theoretical remark (convergence rate). For quadratic objectives with A positive definite, GD converges linearly:

$$\|w^{(t)} - w^\star\|_2 \leq \rho^t \|w^{(0)} - w^\star\|_2, \qquad \rho = \max\{|1 - \eta\lambda_{\min}|, |1 - \eta\lambda_{\max}|\}.$$

Rate depends on condition number $\kappa = \lambda_{\max}/\lambda_{\min}$.

Observation (empirical). The plot shows steady monotonic decrease from about 1.55 to ≈ 0.88 across 200 iterations. This confirms proper step-size choice and that GD approaches w_{ML} smoothly with no oscillation.

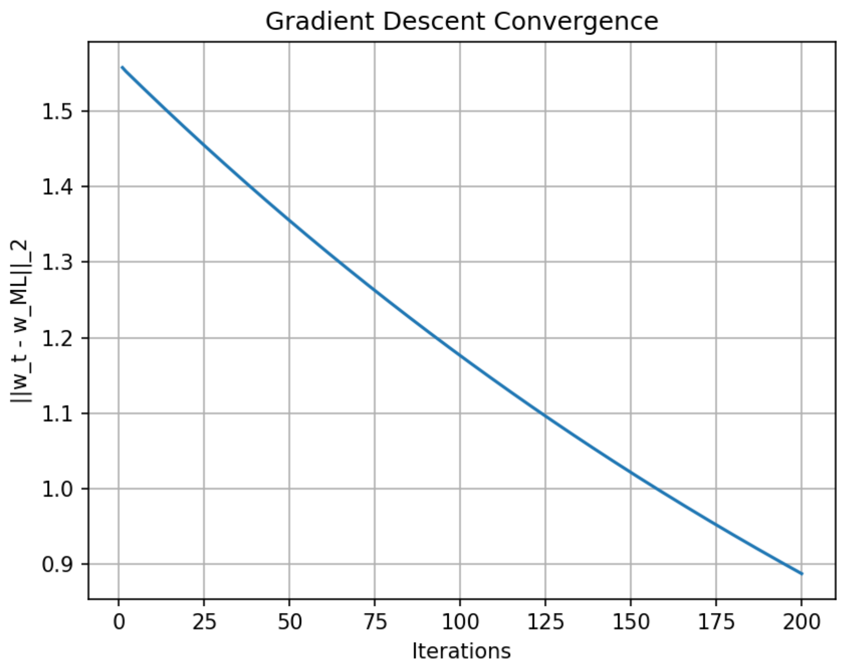

Figure: GD convergence, $\|w_t - w_{ML}\|_2$ vs iterations.

(iii) Stochastic Gradient Descent (mini-batch = 100) and observations

SGD update. For mini-batch B of size b the stochastic gradient is

$$g_B(w) = \frac{1}{b}\tilde{X}_B^\top(\tilde{X}_B w - y_B),$$

and update

$$w^{(t+1)} = w^{(t)} - \eta_{\text{SGD}}\, g_B(w^{(t)}).$$

Implementation details.

- Batch size $b = 100$.
- Step-size chosen relative to GD step: $\eta_{\text{SGD}} = 0.1 \times \eta_{\text{GD}}$ (empirically stable).
- Epochs: 50 (≈ 5000 updates).
- Metric: $\|w^{(t)} - w_{ML}\|_2$ after each update.
- Output: `SGD_norms.png`.

Theory (brief). SGD gives unbiased gradient estimates: $\mathbb{E}[g_B(w)] = \nabla L(w)$. With constant step size it converges to a neighborhood of optimum; with diminishing steps $\eta_t \downarrow 0$ it attains asymptotic convergence. Mini-batch size reduces variance by roughly factor b.

Observation (empirical). The SGD plot shows the distance falling from $\sim$1.6 to $\sim$0.39 after about 5000 updates. The trajectory is smoother in the aggregate but exhibits expected stochastic fluctuation. Compared to batch GD (200 full-data updates), SGD attains smaller distance to w_{ML} given the higher number of parameter updates; per-pass efficiency is often better for SGD in large-data regimes.

(iv) Ridge regression, CV, w_R and comparison with w_{ML}

Ridge objective and closed-form. Ridge (Tikhonov) minimization:

$$L_\lambda(w) = \frac{1}{2N}\|\tilde{X}w - y\|_2^2 + \frac{\lambda}{2}\|w\|_2^2,$$

gives closed-form solution

$$w_R(\lambda) = (\tilde{X}^\top\tilde{X} + N\lambda I)^{-1}\tilde{X}^\top y.$$

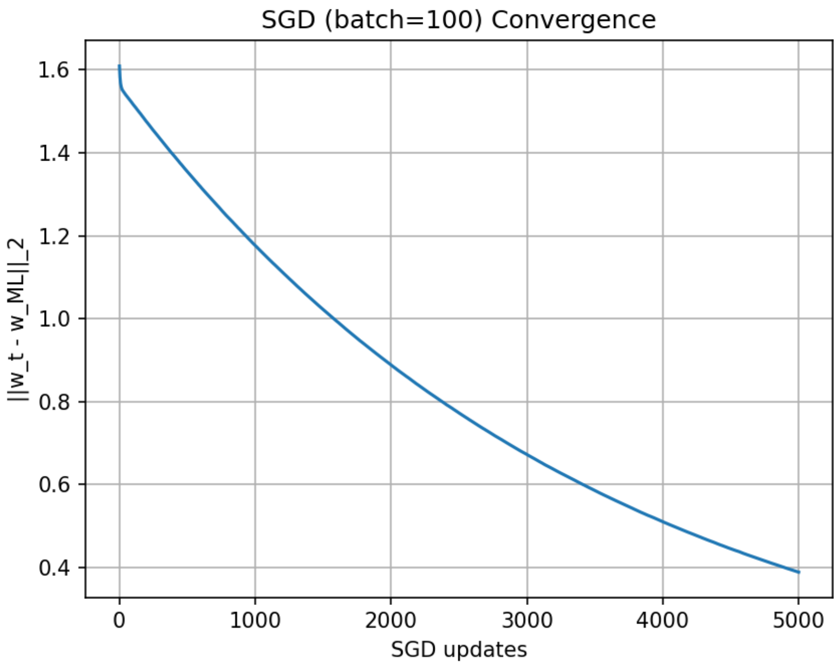

Figure: SGD convergence, $\|w_t - w_{ML}\|_2$ vs number of updates.

This stabilizes inversion and shrinks coefficients; in SVD coordinates each singular direction is scaled by $\sigma_i^2/(\sigma_i^2 + N\lambda)$.

Cross-validation setup.

- 5-fold CV on training set (data shuffled before splitting).
- Candidate λ grid: 30 log-spaced values between 10^{-6} and 10^2.
- Metric: mean validation MSE averaged across folds.

Selected λ and test comparison. Cross-validation selected

$$\hat{\lambda} \approx 1.61026 \times 10^{-4}.$$

Evaluated on provided test set `A2Q2Data_test.csv`:

$$\mathrm{MSE}_{\mathrm{test}}(w_{ML}) \approx 0.370752, \qquad \mathrm{MSE}_{\mathrm{test}}(w_R) \approx 0.370075.$$

Thus ridge with $\hat{\lambda}$ slightly outperforms unregularized least squares. The improvement is small but consistent with weak regularization reducing variance/overfitting.

Interpretation. The very small $\hat{\lambda}$ indicates the original problem is not strongly ill-conditioned or highly overfit; ridge provides a small amount of shrinkage which marginally improves generalization. If singular values had shown a long tail of near-zero values, we would expect a larger $\hat{\lambda}$.

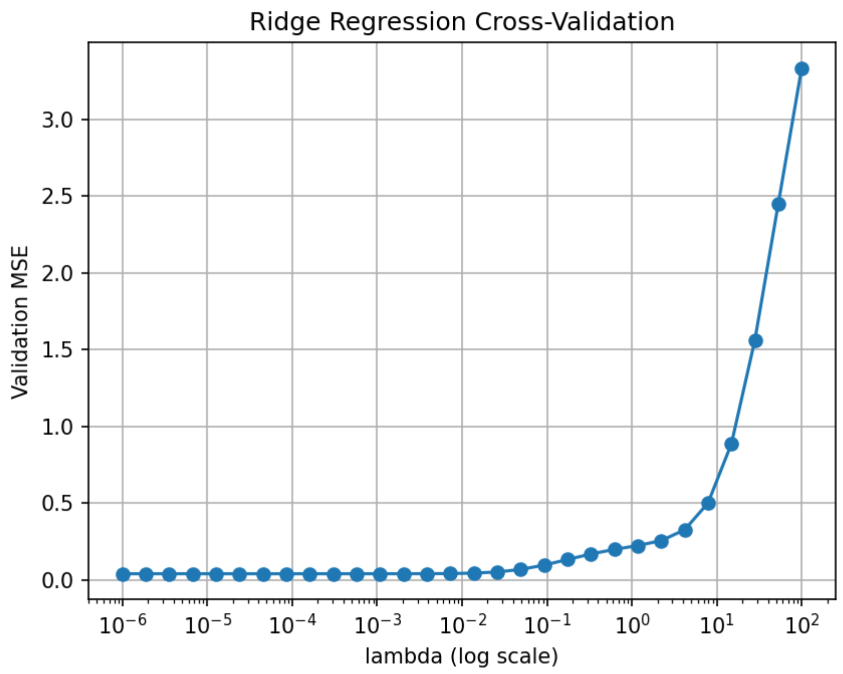

Figure: Cross-validated validation MSE vs λ (log scale).

Summary and concise answers

- **(i)** w_{ML} computed by pseudoinverse; dimension 101.
- **(ii)** Batch GD with $\eta \approx 0.9/\lambda_{\max}$ converges smoothly to w_{ML}; $\|w_t - w_{ML}\|_2$ decreases monotonically .
- **(iii)** SGD (batch = 100) converges faster in updates, but with stochastic noise; $\|w_t - w_{ML}\|_2$ decreases to a small value over ≈ 5000 updates .
- **(iv)** Ridge CV selects $\hat{\lambda} \approx 1.6 \times 10^{-4}$; test MSE slightly improves from ≈ 0.370752 (OLS) to ≈ 0.370075 (ridge), so w_R marginally outperforms w_{ML} due to slight variance reduction via regularization.