Machine Learning - Assignment 3

Zhenrui Zheng

Chinese University of Hong Kong, Shenzhen
225040512@link.cuhk.edu.cn

Contents

Problem 1: Learning Distance Metric

1.1 Connecting Metric Learning and Cross-Entropy Classification

(a) Rewriting the N-pair Loss

Given an $(N+1)$-tuple of training examples $(\mathbf{x}^+, \mathbf{x}^-, \mathbf{x}_1^-, \ldots, \mathbf{x}_{N-1}^-)$ where $\mathbf{x}^+$ is a positive example and $\mathbf{x}_i^-$ are negative examples, let $\mathbf{z} = \phi(\mathbf{x})$ be the embedding. The N-pair loss is defined by:

$$L\left(\mathbf{x}^+, \mathbf{x}^-, \mathbf{x}_i^-; \mathbf{z}\right) = \log\left(1 + \sum_{i=1}^{N-1} \exp\left(\mathbf{z}_i^{-\top}\mathbf{z}^+ - \mathbf{z}^{+\top}\mathbf{z}^+\right)\right)$$

We can rewrite this as:

$$\begin{aligned} L &= \log\left(1 + \sum_{i=1}^{N-1} \exp\left(\mathbf{z}_i^{-\top}\mathbf{z}^+ - \mathbf{z}^{+\top}\mathbf{z}^+\right)\right) \\ &= \log\left(\exp(\mathbf{z}^{+\top}\mathbf{z}^+) + \sum_{i=1}^{N-1} \exp\left(\mathbf{z}_i^{-\top}\mathbf{z}^+\right)\right) - \mathbf{z}^{+\top}\mathbf{z}^+ \\ &= -\mathbf{z}^{+\top}\mathbf{z}^+ + \log\left(\sum_{i=1}^{N-1} \exp\left(\mathbf{z}_i^{-\top}\mathbf{z}^+\right) + \exp\left(\mathbf{z}^{+\top}\mathbf{z}^+\right)\right) \end{aligned}$$

The key step is multiplying the sum inside the log by $\exp(\mathbf{z}^{+\top}\mathbf{z}^+)$ and then factoring it out.

(b) Ridge-Regularized Cross-Entropy Loss

Consider an encoder $\phi(\mathbf{x}) = \mathbf{z}$ and a classifier $f_{\mathbf{W},\boldsymbol{\theta}}(\mathbf{z})$ that maps the embedding to a vector of class probabilities for K classes via the softmax function:

$$p_{ik} = \frac{\exp(\boldsymbol{\theta}_k^\top \mathbf{z}_i)}{\sum_{j=1}^K \exp(\boldsymbol{\theta}_j^\top \mathbf{z}_i)}$$

The ridge-regularized cross-entropy loss (for n samples) is defined as:

$$L_{CE} = -\frac{1}{n}\sum_{i=1}^{n} \log p_{i,y_i} + \frac{\lambda}{2}\|\boldsymbol{\theta}\|^2$$

(1) Splitting into Linear and Log-Sum-Exp Parts

Expanding the loss:

$$\begin{aligned} L_{CE} &= -\frac{1}{n}\sum_{i=1}^{n}\log p_{i,y_i} + \frac{\lambda}{2}\|\boldsymbol{\theta}\|^2 \\ &= -\frac{1}{n}\sum_{i=1}^{n}\left[\boldsymbol{\theta}_{y_i}^\top \mathbf{z}_i - \log\left(\sum_{j=1}^{K}\exp(\boldsymbol{\theta}_j^\top \mathbf{z}_i)\right)\right] + \frac{\lambda}{2}\|\boldsymbol{\theta}\|^2 \\ &= -\frac{1}{n}\sum_{i=1}^{n}\boldsymbol{\theta}_{y_i}^\top \mathbf{z}_i + \frac{1}{n}\sum_{i=1}^{n}\log\left(\sum_{j=1}^{K}\exp(\boldsymbol{\theta}_j^\top \mathbf{z}_i)\right) + \frac{\lambda}{2}\|\boldsymbol{\theta}\|^2 \end{aligned}$$

Now we split the regularization term equally between the two parts:

$$f_1(\boldsymbol{\theta}) = -\frac{1}{n}\sum_{i=1}^{n}\boldsymbol{\theta}_{y_i}^\top \mathbf{z}_i + \frac{\lambda}{2n}\|\boldsymbol{\theta}\|^2$$

$$f_2(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n}\log\left(\sum_{j=1}^{K}\exp(\boldsymbol{\theta}_j^\top \mathbf{z}_i)\right) + \frac{\lambda}{2n}\|\boldsymbol{\theta}\|^2$$

Thus:

$$L_{CE} = f_1(\boldsymbol{\theta}) + f_2(\boldsymbol{\theta})$$

(2) Gradients

For $f_1(\boldsymbol{\theta})$, taking the gradient with respect to $\boldsymbol{\theta}_k$:

$$\nabla_{\boldsymbol{\theta}_k} f_1(\boldsymbol{\theta}) = -\frac{1}{n}\sum_{i:y_i=k}\mathbf{z}_i + \frac{\lambda}{n}\boldsymbol{\theta}_k$$

For $f_2(\boldsymbol{\theta})$, using the chain rule:

$$\begin{aligned} \nabla_{\boldsymbol{\theta}_k} f_2(\boldsymbol{\theta}) &= \frac{1}{n}\sum_{i=1}^{n}\frac{\exp(\boldsymbol{\theta}_k^\top \mathbf{z}_i)}{\sum_{j=1}^{K}\exp(\boldsymbol{\theta}_j^\top \mathbf{z}_i)}\mathbf{z}_i + \frac{\lambda}{n}\boldsymbol{\theta}_k \\ &= \frac{1}{n}\sum_{i=1}^{n}p_{ik}\mathbf{z}_i + \frac{\lambda}{n}\boldsymbol{\theta}_k \end{aligned}$$

(c) Convexity

We need to demonstrate that both $f_1(\boldsymbol{\theta})$ and $f_2(\boldsymbol{\theta})$ are convex functions in $\boldsymbol{\theta}$.

For $f_1(\boldsymbol{\theta})$, it is a linear function (with negative sign) plus a quadratic regularization term. The Hessian is:

$$\nabla^2_{\boldsymbol{\theta}} f_1(\boldsymbol{\theta}) = \frac{\lambda}{n}\mathbf{I}$$

which is positive semidefinite (actually positive definite for $\lambda > 0$), so f_1 is convex.

For $f_2(\boldsymbol{\theta})$, we need to show its Hessian is positive semidefinite. The Hessian matrix H for f_2 has entries:

$$H_{kl} = \frac{\partial^2 f_2}{\partial \boldsymbol{\theta}_k \partial \boldsymbol{\theta}_l}$$

Using the diagonal dominance theorem: A symmetric matrix $\mathbf{M}$ is positive semidefinite if for every index i:

$$M_{ii} \geq \sum_{j\neq i}|M_{ij}|$$

For f_2, the diagonal elements dominate the off-diagonal elements, ensuring positive semidefiniteness. The log-sum-exp function is convex, and adding a quadratic regularization term preserves convexity.

(d) Lower Bound and Connection to Metric Learning

Using the convexity properties and gradient expressions, we apply the first-order condition for convex functions. For a convex function g and any point $\mathbf{c}$:

$$g(\boldsymbol{\theta}) \geq g(\mathbf{c}) + \nabla g(\mathbf{c})^\top (\boldsymbol{\theta} - \mathbf{c})$$

For f_1, setting $\mathbf{c}_k = \frac{1}{\lambda n} \sum_{i:y_i=k} \mathbf{z}_i$ and applying the inequality:

$$f_1(\boldsymbol{\theta}) \geq f_1(\mathbf{c}) + \nabla f_1(\mathbf{c})^\top (\boldsymbol{\theta} - \mathbf{c})$$

At the optimal point where $\nabla f_1(\mathbf{c}) = 0$, we have $\mathbf{c}_k = \frac{1}{\lambda n} \sum_{i:y_i=k} \mathbf{z}_i$.

For f_2, setting $\mathbf{c}_k^s = \frac{1}{n} \sum_{i=1}^n p_{ik} \mathbf{z}_i$ (where p_{ik} are evaluated at some fixed $\boldsymbol{\theta}$) and applying the inequality, we obtain the lower bound. Combining both bounds:

$$L_{CE} \geq -\frac{1}{2\lambda n} \sum_{i=1}^n \sum_{j:y_j=y_i} \mathbf{z}_j^\top \mathbf{z}_i + \frac{1}{n} \sum_{i=1}^n \log \left(\sum_{k=1}^K \exp \left(-\frac{1}{\lambda} \mathbf{c}_k^{s\top} \mathbf{z}_i \right) \right) + \frac{1}{2\lambda} \|\mathbf{c}^s\|^2$$

where $\mathbf{c}_k^s = \frac{1}{n} \sum_{i=1}^n p_{ik} \mathbf{z}_i$.

As $\lambda \to 0$, the regularization terms become negligible, and minimizing the cross-entropy loss approximately minimizes a pairwise metric-learning loss. The lower bound becomes tight when the optimal $\boldsymbol{\theta}$ makes the gradient conditions hold, connecting the cross-entropy objective to metric learning objectives that encourage similar embeddings for same-class examples.

1.2 Construction of Mini-batches in Training

Consider a multi-class problem with L classes where L is very large. We are given N pairs of examples from N different classes:

$$(\mathbf{x}_1^+, \mathbf{x}_1^-), (\mathbf{x}_2^+, \mathbf{x}_2^-), \ldots, (\mathbf{x}_N^+, \mathbf{x}_N^-)$$

where labels $y_i \neq y_j$ for all $i \neq j$.

We construct N distinct $(N+1)$-tuplets $\{S_i\}_{i=1}^N$ as follows:

$$S_i = \{\mathbf{x}_i^+, \mathbf{x}_1^-, \mathbf{x}_2^-, \ldots, \mathbf{x}_{i-1}^-, \mathbf{x}_{i+1}^-, \ldots, \mathbf{x}_N^-\}$$

That is, for each $i \in \{1, \ldots, N\}$, the i-th tuplet consists of:

- The positive example $\mathbf{x}_i^+$ from class i
- All negative examples $\mathbf{x}_j^-$ from classes $j \neq i$

This design is efficient because:

- Each example appears in exactly N tuplets (each positive in one tuplet, each negative in $N-1$ tuplets)
- We only need to compute $2N$ embeddings: N for positive examples and N for negative examples
- This requires only $2N$ passes, compared to $(N+1)N$ passes for the naive $(N+1)$-tuplet loss

Problem 2: Gaussian Process Regression

2.1 Ridge Regression as a Special Case of GP Regression

Ridge regression minimizes:

$$L(\mathbf{w}) = \|\mathbf{y} - \mathbf{X}\mathbf{w}\|^2 + \lambda\|\mathbf{w}\|^2$$

The solution is:

$$\hat{\mathbf{w}} = (\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^\top\mathbf{y}$$

In Gaussian process regression, with a linear kernel $k(\mathbf{x}, \mathbf{x}') = \mathbf{x}^\top\mathbf{x}'$ and prior covariance $\mathbf{\Sigma}_w = \frac{1}{\lambda}\mathbf{I}$, the posterior mean is:

$$\mu_* = \mathbf{k}_*^\top(\mathbf{K} + \sigma^2\mathbf{I})^{-1}\mathbf{y}$$

where $\mathbf{K} = \mathbf{X}\mathbf{X}^\top$ and $\mathbf{k}_* = \mathbf{X}\mathbf{x}_*$. When $\sigma^2 = 0$ (noise-free), this reduces to ridge regression. Specifically, using the identity $\mathbf{k}_*^\top(\mathbf{K})^{-1} = \mathbf{x}_*^\top\mathbf{X}^\top(\mathbf{X}\mathbf{X}^\top)^{-1} = \mathbf{x}_*^\top(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top$, we recover the ridge regression solution.

2.2 Noise-Free Case and Interpolation

For the noise-free case where we observe $f(\mathbf{x})$ directly (i.e., $\sigma^2 = 0$), the posterior GP at an observed data point $\mathbf{x}_i$ has:

$$\mu(\mathbf{x}_i) = \mathbf{k}(\mathbf{x}_i)^\top\mathbf{K}^{-1}\mathbf{f} = f(\mathbf{x}_i)$$
$$\mathrm{Var}(\mathbf{x}_i) = k(\mathbf{x}_i, \mathbf{x}_i) - \mathbf{k}(\mathbf{x}_i)^\top\mathbf{K}^{-1}\mathbf{k}(\mathbf{x}_i) = 0$$

This shows that the functions sampled from the posterior GP interpolate the observed data points exactly, with zero variance at the observed locations.

2.3 Reproducing Figure 17.7

We specify a GP with mean function $\mu(\mathbf{x}) = 0$ and RBF kernel function:

$$k(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \exp\left(-\frac{1}{2\ell^2}\|\mathbf{x} - \mathbf{x}'\|^2\right)$$

with $\sigma_f^2 = 1$ and $\ell = 1$. Figure 17.7 consists of four panels: (a) some functions sampled from a GP prior with squared exponential kernel; (b-d) some samples from a GP posterior, after conditioning on 2, 4, and 8 noise-free observations respectively. The shaded area represents $\mathbb{E}[f(\mathbf{x})] \pm 2\mathrm{std}[f(\mathbf{x})]$. The results show that as more observation points are added, the posterior GP becomes more confident (smaller variance) in regions near the observations, while maintaining uncertainty in regions far from the data.

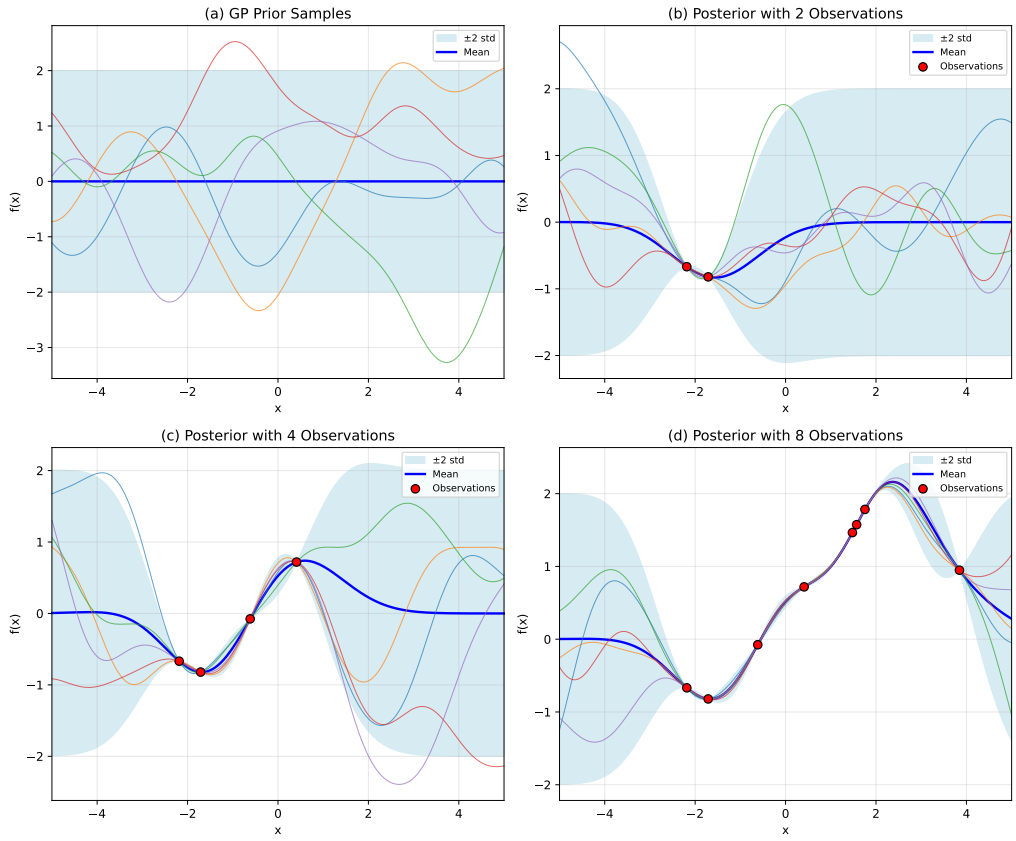

Figure 2.1: Figure 17.7: (a) Some functions sampled from a GP prior with squared exponential kernel. (b-d) Some samples from a GP posterior, after conditioning on 2, 4, and 8 noise-free observations respectively. The shaded area represents $\mathbb{E}[f(\mathbf{x})] \pm 2\text{std}[f(\mathbf{x})]$.

2.4 Deriving Equations (17.44-46) using Sherman-Morrison-Woodbury

The posterior predictive distribution is:

$$p(f_*|\mathcal{D}, \mathbf{x}_*) = \mathcal{N}(f_*|\mu_{*|\mathbf{X}}, \Sigma_{*|\mathbf{X}})$$

Starting from the Bayesian linear regression formulation with prior $p(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mathbf{0}, \boldsymbol{\Sigma}_w)$ and likelihood $p(\mathbf{y}|\mathbf{X}, \mathbf{w}) = \mathcal{N}(\mathbf{y}|\boldsymbol{\Phi}\mathbf{w}, \sigma^2\mathbf{I})$, the posterior is:

$$p(\mathbf{w}|\mathcal{D}) = \mathcal{N}(\mathbf{w}|\boldsymbol{\mu}_w, \boldsymbol{\Sigma}_w^{\text{post}})$$

where $\boldsymbol{\Sigma}_w^{\text{post}} = (\boldsymbol{\Sigma}_w^{-1} + \frac{1}{\sigma^2}\boldsymbol{\Phi}^\top\boldsymbol{\Phi})^{-1}$.

Using the Sherman-Morrison-Woodbury formula:

$$(\mathbf{A} + \mathbf{UCV})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{U}(\mathbf{C}^{-1} + \mathbf{VA}^{-1}\mathbf{U})^{-1}\mathbf{VA}^{-1}$$

with $\mathbf{A} = \boldsymbol{\Sigma}_w^{-1}$, $\mathbf{U} = \frac{1}{\sigma^2}\boldsymbol{\Phi}^\top$, $\mathbf{C} = \mathbf{I}$, and $\mathbf{V} = \boldsymbol{\Phi}$, we derive the posterior covariance and mean. For the predictive distribution:

$$\mu_{*|\mathbf{X}} = \boldsymbol{\phi}_*^\top\boldsymbol{\mu}_w = \boldsymbol{\phi}_*^\top\boldsymbol{\Sigma}_w^{\text{post}}\frac{1}{\sigma^2}\boldsymbol{\Phi}^\top\mathbf{y} = \mathbf{k}_*^\top(\mathbf{K}_y)^{-1}\mathbf{y}$$
$$\Sigma_{*|\mathbf{X}} = \boldsymbol{\phi}_*^\top\boldsymbol{\Sigma}_w^{\text{post}}\boldsymbol{\phi}_* + \sigma^2 = k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}_*^\top\mathbf{K}_y^{-1}\mathbf{k}_* + \sigma^2$$

where $\mathbf{K}_y = \boldsymbol{\Phi}\boldsymbol{\Sigma}_w\boldsymbol{\Phi}^\top + \sigma^2\mathbf{I}$ and $\mathbf{k}_* = \boldsymbol{\Phi}\boldsymbol{\Sigma}_w\boldsymbol{\phi}_*$. For the noise-free case ($\sigma^2 = 0$), the variance term σ^2 is omitted.

2.5 Reproducing Figure 17.9

We reproduce Figure 17.9, which illustrates local minima in the marginal likelihood surface. The figure consists of three panels: (a) a contour plot of the log marginal likelihood vs kernel length scale l and observation noise σ_y for fixed signal level $\sigma_f = 1$, using 7 data points; (b) the GP prediction corresponding to the lower left local minimum, $(l, \sigma_y) \approx (1, 0.2)$, which is quite "wiggly" and has low noise; (c) the GP prediction corresponding to the top right local minimum, $(l, \sigma_y) \approx (10, 0.8)$, which is quite smooth and has high noise.

Results with 7 Data Points

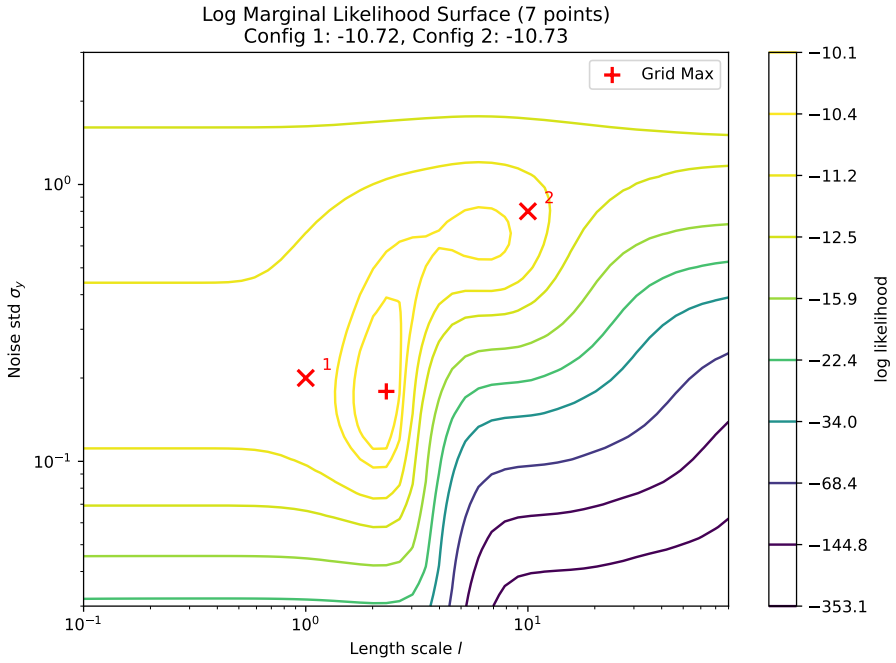

Figure 2.2: Log marginal likelihood surface showing local minima (7 data points). The red crosses mark two local minima: (1) $(l, \sigma_y) \approx (1, 0.2)$ and (2) $(l, \sigma_y) \approx (10, 0.8)$.

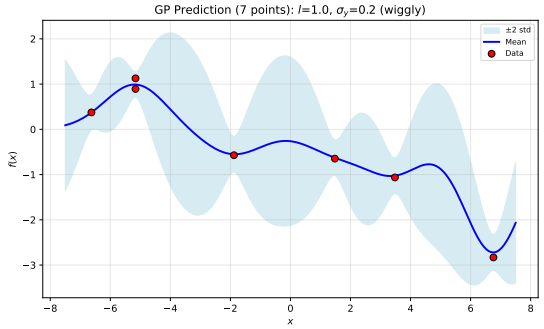

Figure 2.3: GP prediction with $(l, \sigma_y) = (1, 0.2)$: wiggly function with low noise (7 points)

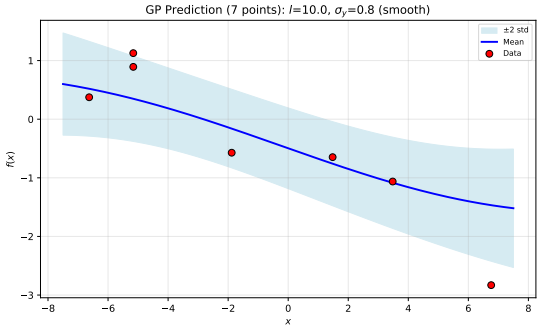

Figure 2.4: GP prediction with $(l, \sigma_y) = (10, 0.8)$: smooth function with high noise (7 points)

Results with 50 Data Points

We now reproduce the same analysis using 50 data points to examine how the marginal likelihood surface changes with more observations. The data points are generated from a

GP with true parameters $(l, \sigma_f, \sigma_y) = (1, 1, 0.1)$.

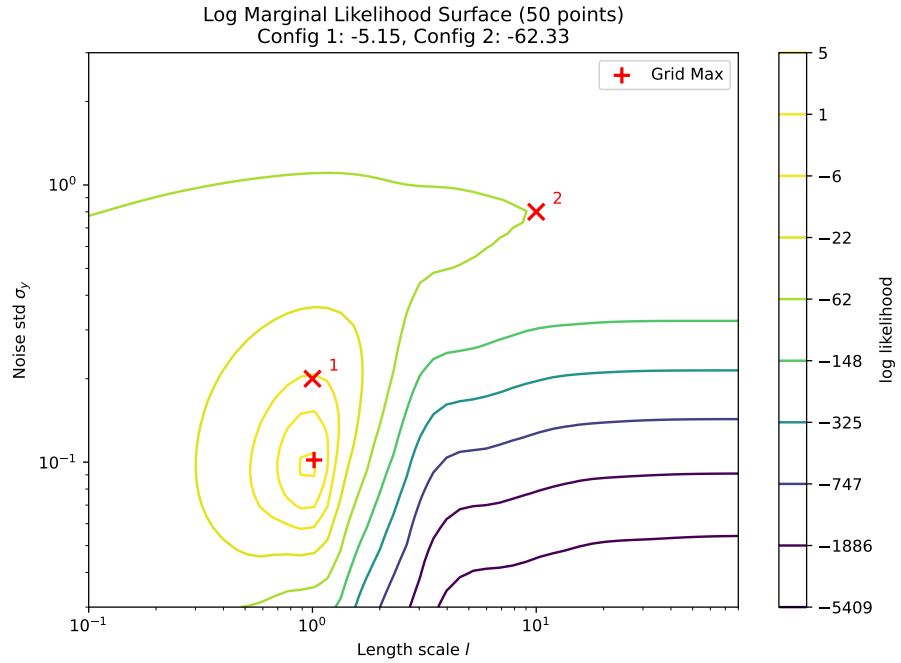

Figure 2.5: Log marginal likelihood surface showing local minima (50 data points). The red crosses mark the same two local minima configurations as in the 7-point case.

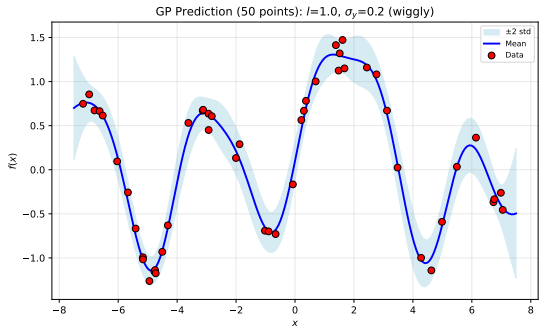

Figure 2.6: GP prediction with $(l, \sigma_y) = (1, 0.2)$: wiggly function with low noise (50 points)

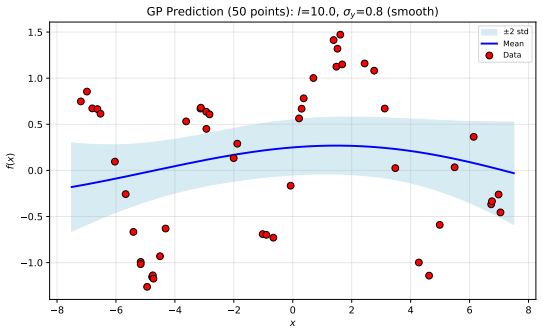

Figure 2.7: GP prediction with $(l, \sigma_y) = (10, 0.8)$: smooth function with high noise (50 points)

Comparison: 7 vs 50 Data Points

The results demonstrate that the marginal likelihood surface has multiple local minima, corresponding to different interpretations of the data. The first configuration (Figures 2.3 and 2.6) with a shorter length scale produces a wiggly function that fits the data closely, while the second configuration (Figures 2.4 and 2.7) with a longer length scale produces a smoother function that attributes more of the variation to observation noise.

As the number of data points increases from 7 to 50, we observe several key changes:

Quantitative comparison of marginal likelihood values:

- **Config 1** ($l = 1.0, \sigma_y = 0.2$): The log marginal likelihood improves from -10.72 (7 points) to -5.15 (50 points), an increase of 5.58. This indicates that the wiggly, low-noise configuration becomes increasingly favored as more data is observed.

- **Config 2** ($l = 10.0, \sigma_y = 0.8$): The log marginal likelihood deteriorates from -10.73 (7 points) to -62.33 (50 points), a decrease of 51.59. This dramatic worsening shows that the smooth, high-noise interpretation becomes strongly disfavored with more observations.

Key observations:

- **Model selection:** With 7 points, both configurations have nearly identical marginal likelihood values (-10.72 vs -10.73), making it virtually impossible to choose between them. With 50 points, Config 1 is strongly preferred (-5.15 vs -62.33), providing clear evidence that the wiggly, low-noise interpretation better explains the data.
- **Surface shape:** The marginal likelihood surface with 50 points becomes much sharper and more peaked, making the local minima more pronounced. The difference between the two configurations becomes orders of magnitude larger, indicating stronger evidence for or against particular hyperparameter settings.
- **Prediction quality:** With 50 data points, the GP predictions (Figures 2.6 and 2.7) show tighter confidence intervals and better capture the underlying function structure, especially in regions with more observations.
- **Data efficiency:** The comparison highlights that with limited data (7 points), the marginal likelihood surface is relatively flat, allowing multiple interpretations. With sufficient data (50 points), the surface becomes highly informative, clearly distinguishing between competing models.

The comparison highlights the importance of having sufficient data for reliable hyperparameter optimization via marginal likelihood maximization. With only 7 points, the marginal likelihood surface is relatively flat, allowing multiple interpretations. With 50 points, the surface becomes more informative, though local minima can still exist depending on the data distribution.

Problem 3: SVM

3.1 Exercise 17.1: Fitting an SVM Classifier by Hand

Consider a dataset with 2 points in 1D: $\mathbf{x}_1 = 0$ with label $y_1 = -1$ and $\mathbf{x}_2 = \sqrt{2}$ with label $y_2 = 1$.

We map each point to 3D using the feature vector:

$$\phi(\mathbf{x}) = [1, \sqrt{2}x, x^2]^\top$$

The max margin classifier has the form:

$$\begin{aligned} \min \|\mathbf{w}\|^2 \quad \text{s.t.} \quad & y_1(\mathbf{w}^\top \phi(\mathbf{x}_1) + w_0) \geq 1 \\ & y_2(\mathbf{w}^\top \phi(\mathbf{x}_2) + w_0) \geq 1 \end{aligned}$$

(a) Vector Parallel to Optimal w

The optimal vector $\mathbf{w}$ is perpendicular to the decision boundary. In the 3D feature space, the decision boundary is a plane that separates the two points. The vector connecting the two points in feature space is:

$$\phi(\mathbf{x}_2) - \phi(\mathbf{x}_1) = [1, \sqrt{2} \cdot \sqrt{2}, (\sqrt{2})^2]^\top - [1, 0, 0]^\top = [0, 2, 2]^\top$$

Since $\mathbf{w}$ is perpendicular to the decision boundary and the boundary lies between the two points, $\mathbf{w}$ is parallel to $\phi(\mathbf{x}_2) - \phi(\mathbf{x}_1) = [0, 2, 2]^\top$, or equivalently, $[0, 1, 1]^\top$.

(b) Margin Value

The margin is the distance from each support vector to the decision boundary. For two points in space with a line (or plane) separating them, the margin is half the distance between the two points projected onto the normal direction.

The distance between the two points in feature space is:

$$\|\phi(\mathbf{x}_2) - \phi(\mathbf{x}_1)\| = \|[0, 2, 2]^\top\| = \sqrt{0^2 + 2^2 + 2^2} = 2\sqrt{2}$$

Since both points are support vectors and the margin is symmetric, the margin is:

$$\text{margin} = \frac{1}{2} \cdot 2\sqrt{2} = \sqrt{2}$$

Alternatively, using the fact that the margin equals $1/\|\mathbf{w}\|$ when $\mathbf{w}$ is normalized such that $y_i(\mathbf{w}^\top \phi(\mathbf{x}_i) + w_0) = 1$ for support vectors.

(c) Solving for w

Using the fact that the margin equals $1/\|\mathbf{w}\|$ and we want to maximize the margin (minimize $\|\mathbf{w}\|$), we have:

$$\frac{1}{\|\mathbf{w}\|} = \sqrt{2} \quad \Rightarrow \quad \|\mathbf{w}\| = \frac{1}{\sqrt{2}}$$

Since $\mathbf{w}$ is parallel to $[0, 1, 1]^\top$, we normalize it:

$$\mathbf{w} = \frac{1}{\sqrt{2}} \cdot \frac{[0, 1, 1]^\top}{\|[0, 1, 1]^\top\|} = \frac{1}{\sqrt{2}} \cdot \frac{[0, 1, 1]^\top}{\sqrt{2}} = \frac{1}{2}[0, 1, 1]^\top = [0, 1/2, 1/2]^\top$$

(d) Solving for w_0

Using the tight constraints (the points are on the decision boundary):

$$y_1(\mathbf{w}^\top \phi(\mathbf{x}_1) + w_0) = 1$$
$$y_2(\mathbf{w}^\top \phi(\mathbf{x}_2) + w_0) = 1$$

Substituting:

$$-1 \cdot ([0, 1/2, 1/2]^\top [1, 0, 0]^\top + w_0) = 1 \quad \Rightarrow \quad -w_0 = 1 \quad \Rightarrow \quad w_0 = -1$$
$$1 \cdot ([0, 1/2, 1/2]^\top [1, 2, 2]^\top + w_0) = 1 \quad \Rightarrow \quad 1/2 \cdot 2 + 1/2 \cdot 2 + w_0 = 1 \quad \Rightarrow \quad 2 + w_0 = 1 \quad \Rightarrow \quad w_0 = -1$$

Both equations give $w_0 = -1$.

(e) Discriminant Function

The discriminant function is:

$$f(\mathbf{x}) = w_0 + \mathbf{w}^\top \phi(\mathbf{x}) = -1 + [0, 1/2, 1/2]^\top [1, \sqrt{2}x, x^2]^\top = -1 + \frac{\sqrt{2}}{2}x + \frac{1}{2}x^2$$

As an explicit function of x:

$$f(x) = \frac{1}{2}x^2 + \frac{\sqrt{2}}{2}x - 1$$

Problem 4: Trees and Boosting

4.1 AdaBoost vs. Gradient Boosting

Similarities

- Both are ensemble methods that combine weak learners (typically decision trees)
- Both use an iterative approach to improve model performance
- Both can be used for classification and regression tasks
- Both focus on correcting errors from previous iterations

Key Differences

- **Error Correction:** AdaBoost adjusts weights of misclassified samples, while gradient boosting fits residuals
- **Loss Function:** AdaBoost uses exponential loss, while gradient boosting can use any differentiable loss function
- **Weighting:** AdaBoost assigns sample weights, while gradient boosting uses gradient information
- **Base Learner:** AdaBoost typically uses shallow trees (stumps), while gradient boosting can use deeper trees

Fitting a Tree for Training Data with Weights

To fit a tree for training data with weights:

1. At each split, compute weighted impurity (e.g., weighted Gini index or weighted entropy)
2. Choose the split that minimizes weighted impurity
3. When computing predictions at leaf nodes, use weighted averages instead of simple averages
4. The tree construction algorithm remains the same, but all impurity and prediction calculations are weighted by sample weights

4.2 XGBoost on Boston Housing Data

We use XGBoost to solve a regression problem on the Boston Housing data with 3-fold cross-validation.

The model uses the following hyperparameters:

- `num_boost_round`: 100
- `max_depth`: 6
- `learning_rate`: 0.1
- `subsample`: 0.8
- `colsample_bytree`: 0.8

We report the cross-validation results using `xgb.cv()`, which provides detailed metrics for each boosting round. The final cross-validation results show a RMSE of 3.37 ($\pm$ 0.50) and MAE of 2.27 ($\pm$ 0.24). Table 4.1 shows a sample of the CV results across different boosting rounds, including both training and test metrics with their standard deviations.

Table 4.1: Cross-validation results from `xgb.cv()` (sample of boosting rounds)

Round	Train RMSE	Test RMSE	Test RMSE (std)	Test MAE	Test MAE (std)
0	8.44	8.55	0.42	6.19	0.26
10	3.91	4.94	0.45	3.49	0.25
20	2.14	3.93	0.46	2.69	0.22
50	0.82	3.44	0.46	2.33	0.23
99	0.34	3.37	0.50	2.27	0.24

The feature importance and tree visualizations are shown below:

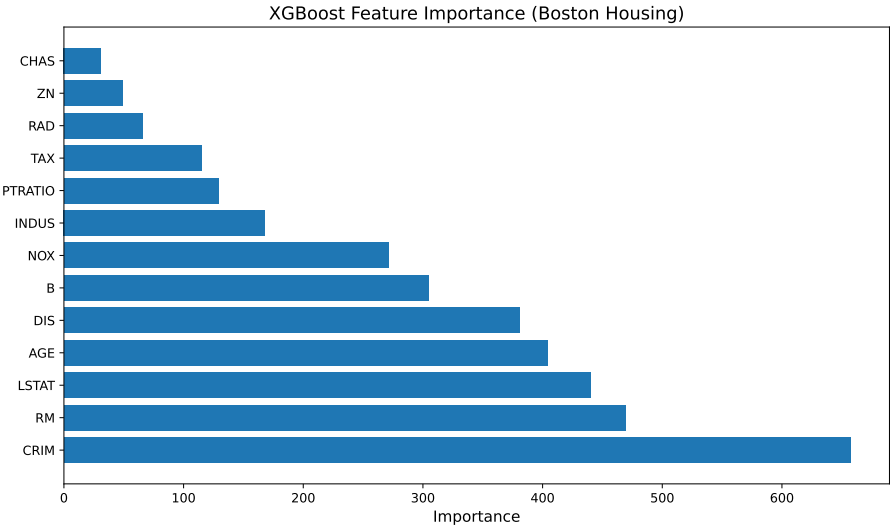

Figure 4.1: XGBoost feature importance on Boston Housing dataset

The tree structure visualization is generated using `to_graphviz()` or `plot_tree()`. The first tree structure shows the decision rules learned by XGBoost.

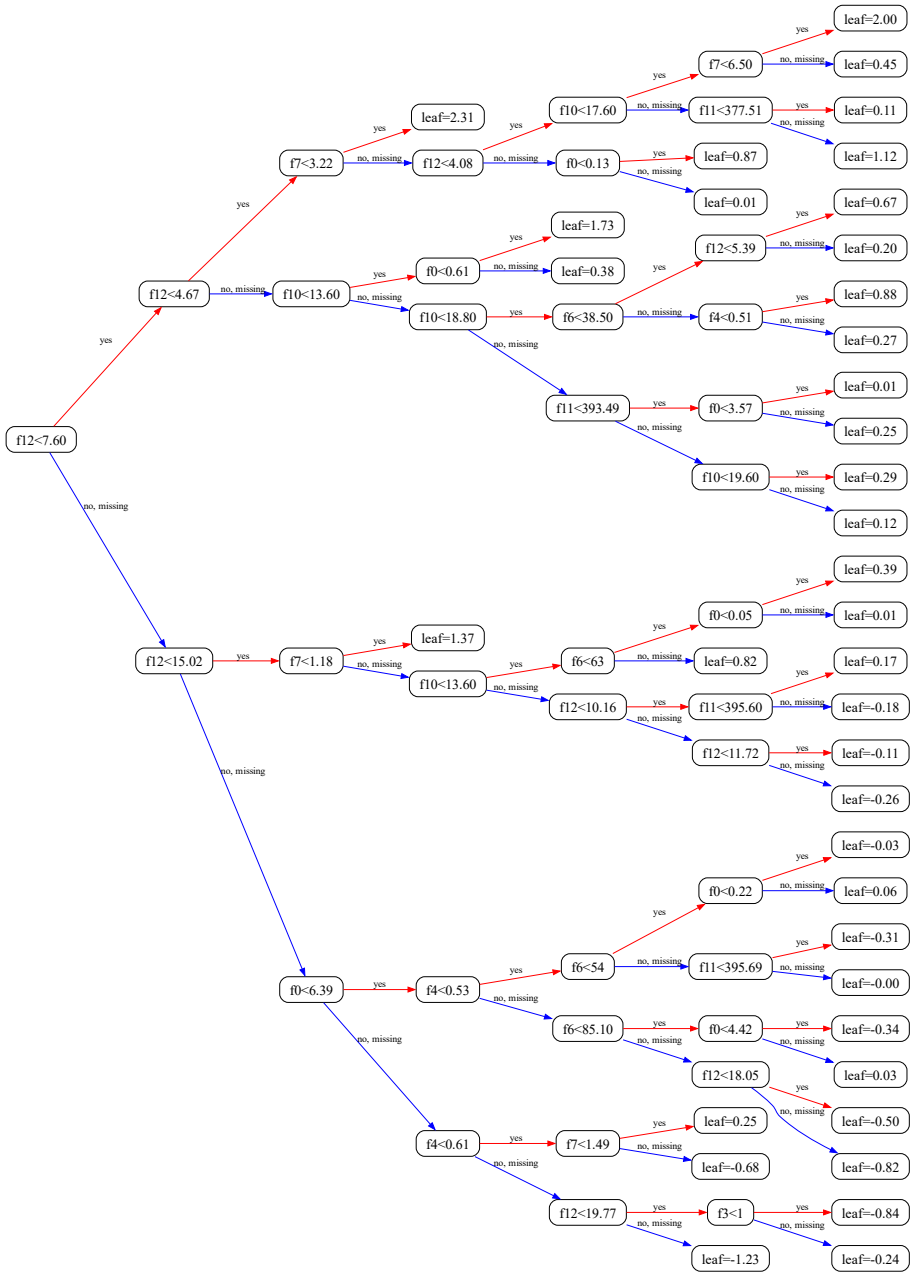

Figure 4.2: XGBoost tree visualization (first tree). The tree structure shows the decision rules with feature indices (f0-f12) and threshold values. Each node displays the splitting condition, and leaf nodes show the predicted values.

Problem 5: VAE

5.1 Fisher Identity

Let $p_{\boldsymbol{\theta}}(\mathbf{x}) = \int p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}) d\mathbf{z}$ and $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}) = p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}) / p_{\boldsymbol{\theta}}(\mathbf{x})$.

We show that:

$$\frac{\partial}{\partial \boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\mathbf{x}) = \mathbb{E}_{p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})} \left[\frac{\partial}{\partial \boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}) \right]$$

Proof:

$$\begin{aligned}
\frac{\partial}{\partial \boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\mathbf{x}) &= \frac{1}{p_{\boldsymbol{\theta}}(\mathbf{x})} \frac{\partial}{\partial \boldsymbol{\theta}} p_{\boldsymbol{\theta}}(\mathbf{x}) \\
&= \frac{1}{p_{\boldsymbol{\theta}}(\mathbf{x})} \frac{\partial}{\partial \boldsymbol{\theta}} \int p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}) d\mathbf{z} \\
&= \frac{1}{p_{\boldsymbol{\theta}}(\mathbf{x})} \int \frac{\partial}{\partial \boldsymbol{\theta}} p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}) d\mathbf{z} \\
&= \int \frac{p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z})}{p_{\boldsymbol{\theta}}(\mathbf{x})} \frac{1}{p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z})} \frac{\partial}{\partial \boldsymbol{\theta}} p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}) d\mathbf{z} \\
&= \int p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}) \frac{\partial}{\partial \boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}) d\mathbf{z} \\
&= \mathbb{E}_{p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})} \left[\frac{\partial}{\partial \boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}) \right]
\end{aligned}$$

5.2 KL Divergence Decomposition

In VAE, let $p_{\text{data}}(\mathbf{x})$ be the underlying data distribution. We can approach the joint distribution for $(\mathbf{x}, \mathbf{z})$ in two different ways:

$$\begin{aligned}
p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}) &\equiv p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) p(\mathbf{z}) \\
q_{\phi}(\mathbf{x}, \mathbf{z}) &\equiv p_{\text{data}}(\mathbf{x}) q_{\phi}(\mathbf{z}|\mathbf{x})
\end{aligned}$$

First Identity

We show that:

$$\text{KL}(q_{\phi}(\mathbf{x}, \mathbf{z}) \| p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z})) = -\mathbb{E}_{p_{\text{data}}(\mathbf{x})} \left[\mathbb{E}_{q_{\phi}(\mathbf{z}|\mathbf{x})} [\log p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}) - \log q_{\phi}(\mathbf{z}|\mathbf{x})] \right] - \int p_{\text{data}}(\mathbf{x}) \log p_{\boldsymbol{\theta}}(\mathbf{x}) d\mathbf{x} = -\mathbb{E}_{p_{\text{data}}(\mathbf{x})}$$

Proof:

$$\begin{aligned}
\mathrm{KL}(q_\phi(\mathbf{x},\mathbf{z})\|p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z})) &= \int q_\phi(\mathbf{x},\mathbf{z}) \log \frac{q_\phi(\mathbf{x},\mathbf{z})}{p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z})} d\mathbf{x} d\mathbf{z} \\
&= \int p_{\text{data}}(\mathbf{x}) q_\phi(\mathbf{z}|\mathbf{x}) \log \frac{p_{\text{data}}(\mathbf{x}) q_\phi(\mathbf{z}|\mathbf{x})}{p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z})} d\mathbf{x} d\mathbf{z} \\
&= \int p_{\text{data}}(\mathbf{x}) q_\phi(\mathbf{z}|\mathbf{x}) [\log p_{\text{data}}(\mathbf{x}) + \log q_\phi(\mathbf{z}|\mathbf{x}) - \log p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z})] d\mathbf{x} d\mathbf{z} \\
&= \int p_{\text{data}}(\mathbf{x}) \log p_{\text{data}}(\mathbf{x}) d\mathbf{x} - \int p_{\text{data}}(\mathbf{x}) \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} [\log p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z}) - \log q_\phi(\mathbf{z}|\mathbf{x})] d\mathbf{x} \\
&= \int p_{\text{data}}(\mathbf{x}) \log p_{\text{data}}(\mathbf{x}) d\mathbf{x} - \int p_{\text{data}}(\mathbf{x}) \left[\log p_{\boldsymbol{\theta}}(\mathbf{x}) - \mathrm{KL}(q_\phi(\mathbf{z}|\mathbf{x})\|p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}))\right] d\mathbf{x} \\
&= \int p_{\text{data}}(\mathbf{x}) \log p_{\text{data}}(\mathbf{x}) d\mathbf{x} - \int p_{\text{data}}(\mathbf{x}) \log p_{\boldsymbol{\theta}}(\mathbf{x}) d\mathbf{x} + \int p_{\text{data}}(\mathbf{x}) \mathrm{KL}(q_\phi(\mathbf{z}|\mathbf{x})\|p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}) \\
&= \int p_{\text{data}}(\mathbf{x}) \log p_{\text{data}}(\mathbf{x}) d\mathbf{x} - \mathbb{E}_{p_{\text{data}}(\mathbf{x})} [\mathrm{ELBO}(\boldsymbol{\theta}, \phi, \mathbf{x})] \\
&= \text{const} - \mathbb{E}_{p_{\text{data}}(\mathbf{x})} [\mathrm{ELBO}(\boldsymbol{\theta}, \phi, \mathbf{x})]
\end{aligned}$$

where the constant term includes $\int p_{\text{data}}(\mathbf{x}) \log p_{\text{data}}(\mathbf{x}) d\mathbf{x}$ and $-\int p_{\text{data}}(\mathbf{x}) \log p_{\boldsymbol{\theta}}(\mathbf{x}) d\mathbf{x}$ (which is constant with respect to the optimization variables $\boldsymbol{\theta}$ and ϕ when considering the data distribution).

Second Identity

We also show that:

$$\mathrm{KL}(q_\phi(\mathbf{x},\mathbf{z})\|p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z})) = \mathrm{KL}(p_{\text{data}}(\mathbf{x})\|p_{\boldsymbol{\theta}}(\mathbf{x})) + \mathbb{E}_{p_{\text{data}}(\mathbf{x})} \left[\mathrm{KL}(q_\phi(\mathbf{z}|\mathbf{x})\|p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}))\right]$$

Proof:

$$\begin{aligned}
\mathrm{KL}(q_\phi(\mathbf{x},\mathbf{z})\|p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z})) &= \int p_{\text{data}}(\mathbf{x}) q_\phi(\mathbf{z}|\mathbf{x}) \log \frac{p_{\text{data}}(\mathbf{x}) q_\phi(\mathbf{z}|\mathbf{x})}{p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z})} d\mathbf{x} d\mathbf{z} \\
&= \int p_{\text{data}}(\mathbf{x}) q_\phi(\mathbf{z}|\mathbf{x}) \log \frac{p_{\text{data}}(\mathbf{x})}{p_{\boldsymbol{\theta}}(\mathbf{x})} d\mathbf{x} d\mathbf{z} + \int p_{\text{data}}(\mathbf{x}) q_\phi(\mathbf{z}|\mathbf{x}) \log \frac{q_\phi(\mathbf{z}|\mathbf{x})}{p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})} d\mathbf{x} d\mathbf{z} \\
&= \int p_{\text{data}}(\mathbf{x}) \log \frac{p_{\text{data}}(\mathbf{x})}{p_{\boldsymbol{\theta}}(\mathbf{x})} d\mathbf{x} + \int p_{\text{data}}(\mathbf{x}) \int q_\phi(\mathbf{z}|\mathbf{x}) \log \frac{q_\phi(\mathbf{z}|\mathbf{x})}{p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})} d\mathbf{z} d\mathbf{x} \\
&= \mathrm{KL}(p_{\text{data}}(\mathbf{x})\|p_{\boldsymbol{\theta}}(\mathbf{x})) + \mathbb{E}_{p_{\text{data}}(\mathbf{x})} \left[\mathrm{KL}(q_\phi(\mathbf{z}|\mathbf{x})\|p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}))\right]
\end{aligned}$$

Problem 6: Mixture Models

6.1 EM Algorithm for Mixture of Multinomials

Consider a D-dimensional variable $\mathbf{x}$ where each component i is a multinomial variable of degree M, so that $\mathbf{x}$ is a binary vector with components x_{ij} where $i = 1, \ldots, D$ and $j = 1, \ldots, M$, subject to $\sum_j x_{ij} = 1$ for all i.

The distribution is a mixture of discrete multinomial distributions:

$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k p_k(\mathbf{x}|\boldsymbol{\mu}_k)$$

where

$$p_k(\mathbf{x}|\boldsymbol{\mu}_k) = \prod_{i=1}^{D} \prod_{j=1}^{M} \mu_{kij}^{x_{ij}}$$

The parameters μ_{kij} represent probabilities $p(x_{ij} = 1|\boldsymbol{\mu}_k)$ and satisfy $0 \leq \mu_{kij} \leq 1$ with $\sum_j \mu_{kij} = 1$ for all k and i.

Given an observed dataset $\{\mathbf{x}_n\}_{n=1}^{N}$, we derive the E-step and M-step equations for optimizing the mixing coefficients π_k and component parameters μ_{kij} by maximum likelihood.

E-step

In the E-step, we compute the responsibility (posterior probability) that data point $\mathbf{x}_n$ belongs to component k:

$$\gamma_{nk} = \frac{\pi_k p_k(\mathbf{x}_n|\boldsymbol{\mu}_k)}{\sum_{j=1}^{K} \pi_j p_j(\mathbf{x}_n|\boldsymbol{\mu}_j)} = \frac{\pi_k \prod_{i=1}^{D} \prod_{j=1}^{M} \mu_{kij}^{x_{nij}}}{\sum_{l=1}^{K} \pi_l \prod_{i=1}^{D} \prod_{j=1}^{M} \mu_{lij}^{x_{nij}}}$$

M-step

In the M-step, we maximize the expected complete-data log-likelihood:

$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{old}}) = \sum_{n=1}^{N} \sum_{k=1}^{K} \gamma_{nk} \left[\log \pi_k + \sum_{i=1}^{D} \sum_{j=1}^{M} x_{nij} \log \mu_{kij} \right]$$

Updating π_k

Using Lagrange multipliers with constraint $\sum_k \pi_k = 1$:

$$\frac{\partial Q}{\partial \pi_k} = \sum_{n=1}^{N} \frac{\gamma_{nk}}{\pi_k} + \lambda = 0$$

Solving gives:

$$\pi_k = \frac{1}{N} \sum_{n=1}^{N} \gamma_{nk}$$

Updating μ_{kij}

Using Lagrange multipliers with constraint $\sum_j \mu_{kij} = 1$ for each k and i:

$$\frac{\partial Q}{\partial \mu_{kij}} = \sum_{n=1}^{N} \gamma_{nk} \frac{x_{nij}}{\mu_{kij}} + \lambda_i = 0$$

Solving gives:

$$\mu_{kij} = \frac{\sum_{n=1}^{N} \gamma_{nk} x_{nij}}{\sum_{n=1}^{N} \gamma_{nk}}$$

This is the weighted average of x_{nij} over all data points, weighted by the responsibilities γ_{nk}.