Assignment 1 — Statistical Learning Foundations

Advanced Machine and Deep Learning (WS 25/26)
Vedant Dave, M.Sc.

Deadline: 15th November 2025

Goal

This assignment connects statistical learning theory with practical implementation.

Q1. Estimators

Generate samples $x_i \sim \mathcal{N}(0, 1)$ for $n = [10, 100, 1000, 10000, 100000]$. Plot how the sample mean changes as n increases. Add the true mean as a horizontal reference line.

Q2. Regression with Gaussian Noise (MLE)

In the lecture, we discussed that when we assume a **Gaussian conditional model** for regression:

$$y_i = x_i^\top \theta + \varepsilon_i, \quad \varepsilon_i \sim \mathcal{N}(0, \sigma^2),$$

the likelihood of the dataset is given by

$$p(y|X, \theta) = (2\pi\sigma^2)^{-\frac{n}{2}} \exp\left(-\frac{1}{2\sigma^2} \sum_{i=1}^{n} (y_i - x_i^\top \theta)^2 \right).$$

Taking the logarithm and ignoring constants gives the **negative log-likelihood (NLL)**:

$$\mathcal{L}(\theta) = \frac{1}{2\sigma^2} \sum_{i=1}^{n} (y_i - x_i^\top \theta)^2.$$

Hence, maximizing the Gaussian likelihood is equivalent (up to a scaling factor) to minimizing the **Mean Squared Error (MSE)**. This means the **Maximum Likelihood Estimate (MLE)** of θ can be obtained analytically as:

$$\boxed{\hat{\theta} = (X^\top X)^{-1} X^\top y.}$$

(a) Fit the model using the closed-form solution

Use the `CaliforniaHousing` dataset from `sklearn.datasets`.

- Standardize the input features (do *not* normalize the target).
- Fit a linear regression model using the above closed-form equation.
- Report the **training** and **testing MSE**.

(b) Visualize learning behavior

To understand how the estimator behaves as we collect more data:

- Randomly shuffle the training data.
- Fit the model on increasing fractions of the training set (e.g., 10%, 20%, ..., 100%).
- For each fraction, compute:
  - **Training MSE**
  - **Testing MSE**
- Plot both as functions of the training fraction (a **learning curve**).

(c) Verify the MLE–MSE equivalence

Recall that, under the Gaussian model,

$$\mathrm{NLL}(\theta) = \frac{1}{2\sigma^2} \sum_i (y_i - x_i^\top \theta)^2.$$

This means that for a **fixed** variance σ^2, the NLL is simply a scaled version of the MSE.

1. Compute a fixed estimate of σ^2 from the residual variance of your full-data model:

$$\hat{\sigma}^2 = \mathrm{Var}(y - X\hat{\theta})$$

2. Using this same $\hat{\sigma}^2$ for all models, compute and plot the **per-sample Gaussian NLL** versus **training MSE** as the training fraction increases.
3. Observe that both curves have identical shapes, differing only by a constant scale factor, confirming the equivalence between minimizing MSE and maximizing Gaussian likelihood.

Q3. MAP Estimation and Regularization (Ridge vs Lasso)

In the previous question, we derived the **Maximum Likelihood Estimate (MLE)** for linear regression by maximizing the Gaussian likelihood, which was equivalent to minimizing the Mean Squared Error (MSE).

Now, we extend this idea to the **Maximum A Posteriori (MAP)** framework by introducing a prior belief on the model parameters. If we assume a **Gaussian prior** on the weights,

$$p(\theta) = \mathcal{N}(0, \tau^2 I),$$

the MAP estimate becomes

$$\hat{\theta}_{\mathrm{MAP}} = \arg\min_\theta \frac{1}{2\sigma^2} \sum_i (y_i - x_i^\top \theta)^2 + \frac{1}{2\tau^2} \|\theta\|_2^2,$$

which corresponds to **Ridge Regression**, where

$$\lambda = \frac{\sigma^2}{\tau^2}.$$

If we instead assume a **Laplacian prior**,

$$p(\theta) = \text{Laplace}(0, b),$$

then the MAP estimate becomes

$$\hat{\theta}_{\text{MAP}} = \arg\min_{\theta} \frac{1}{2\sigma^2} \sum_i (y_i - x_i^\top \theta)^2 + \frac{1}{b} \|\theta\|_1,$$

which corresponds to **Lasso Regression** (L1 regularization). Ridge (L2) penalizes large weights smoothly, while Lasso (L1) can drive some coefficients exactly to zero, producing a sparse model.

Fit Ridge and Lasso models

- Use the `CaliforniaHousing` dataset from `sklearn.datasets`.
- Standardize all input features using `StandardScaler` (keep the target unchanged).
- Import both models from `sklearn.linear model` and use `Ridge, Lasso`:
- For each method, fit models for a range of

$$\lambda \in \{10^{-6}, 10^{-5}, 10^{-4}, \ldots, 10^{3}\}.$$

- Compute the **training** and **testing RMSE** for each value of λ.
  Hint: use `mean_squared_error` from `sklearn.metrics` and take its square root.

Q4. Information and Cross-Entropy

For two discrete probability distributions p and q defined on the same support, the key information quantities are:

$$H(p) = -\sum_i p_i \log p_i, \qquad H(p, q) = -\sum_i p_i \log q_i, \qquad D_{\text{KL}}(p\|q) = H(p, q) - H(p) \geq 0.$$

The Kullback–Leibler (KL) divergence measures how different q is from p, and equals zero only when $p = q$.

Now, you will construct simple discrete distributions and visualize how $D_{\text{KL}}(p\|q)$ increases as q drifts away from p.

(a) Generate a base distribution p

- Create a discrete distribution p over indices $\{0, 1, \ldots, 20\}$.
- One simple option: take a Gaussian-shaped distribution centered at index 10 and normalize it so that $\sum_i p_i = 1$.
- Plot this probability distribution.

(b) Construct shifted distributions q

- For several drift values $\Delta \in \{-8, -6, \ldots, 8\}$, create new distributions q_Δ by shifting the center of the distribution by Δ.
- For one representative drift (e.g., $\Delta = 5$), plot p and q_Δ on the same graph for comparison.

(c) Compute and visualize KL divergence

- For each Δ, compute the KL divergence:

$$D_{\mathrm{KL}}(p \| q_\Delta) = \sum_i p_i \log \frac{p_i}{q_{\Delta,i}}.$$

- Plot $D_{\mathrm{KL}}(p \| q_\Delta)$ as a function of the drift Δ. Observe how KL divergence increases as q moves further from p.

Submission

- You can submit multiple scripts for every question in the assignment. The scripts must have comments. Make sure to submit a zip.
- Submit separate PDF for answering.
- Any format for PDF allowed.