Introduction to Machine Learning (67577)

Exercise 3
Ensemble Methods, Regularization & Unsupervised

Second Semester, 2025

Contents

1 Theoretical Part

1.1 Regularization

Based on Lecture 5, Recitation 6 and Lab 6

1. In the following question we will show that although the Ridge estimator is biased it can achieve lower generalization error compared to the LS estimator, in the presence of noisy data.
   We discussed in class the concepts of bias and variance in an abstract sense. In some cases, for a given estimator $\hat{\theta}$, we can define and compute the bias and variance using mathematical formulas as follows:

$$\text{Bias}(\hat{\theta}) = \mathbb{E}[\hat{\theta}] - \theta$$

$$\text{Var}(\hat{\theta}) = \mathbb{E}\left[(\hat{\theta} - \mathbb{E}[\hat{\theta}])(\hat{\theta} - \mathbb{E}[\hat{\theta}])^\top\right]$$

   An estimator $\hat{\theta}$ is said to be unbiased if $\text{Bias}(\hat{\theta}) = 0$.
   Let $\mathbf{X} \in \mathbb{R}^{m \times d}$ be a **constant** design matrix, $\mathbf{y} \in \mathbb{R}^m$ a response vector, and assume that $\mathbf{X}^\top\mathbf{X}$ is invertible. Denote $\hat{\mathbf{w}}$ the LS solution and $\hat{\mathbf{w}}_\lambda$ the ridge solution for the parameter $\lambda \geq 0$.
   - Assume the linear model is correct, namely $\mathbf{y} = X\mathbf{w} + \varepsilon$ where $\varepsilon_i \overset{\text{iid}}{\sim} \mathcal{N}(0, \sigma^2)$.
   - Note that in this case LS estimator is unbiased:

$$\mathbb{E}[\hat{\mathbf{w}}] = \mathbb{E}[\mathbf{w} + (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\boldsymbol{\varepsilon}] = \mathbf{w} + (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbb{E}[\boldsymbol{\varepsilon}] = \mathbf{w}$$

   (a) Show that $\hat{\mathbf{w}}_\lambda = A_\lambda\hat{\mathbf{w}}$ where $A_\lambda := (\mathbf{X}^\top\mathbf{X} + \lambda I_d)^{-1}(\mathbf{X}^\top\mathbf{X})$
   (b) From the above, conclude that for any $\lambda > 0$, the ridge estimator is a biased estimator of $\mathbf{w}$. That is, show that for any $\lambda > 0$ we have $\mathbb{E}[\hat{\mathbf{w}}_\lambda] \neq \mathbf{w}$.
   (c) Show that: $\text{Var}(\hat{\mathbf{w}}_\lambda) = \sigma^2 A_\lambda(\mathbf{X}^\top\mathbf{X})^{-1}A_\lambda^\top$, for σ^2 the variance of the assumed noise.
   Hint:: Note that for a constant matrix B and a random vector $\mathbf{z}$ it holds that$\text{Var}(B\mathbf{z}) = B \cdot \text{Var}(\mathbf{z}) \cdot B^\top$ and that $\text{Var}(\hat{\mathbf{w}}) = \sigma^2(\mathbf{X}^\top\mathbf{X})^{-1}$.
   (d) Derive explicit expressions for the (squared) bias and variance of $\hat{\mathbf{w}}_\lambda$ as a function of λ, i.e., write a bias-variance decomposition for the expected squared error (generalization error of the MSE loss function) of $\hat{\mathbf{w}}_\lambda$.

   Hint: in the multivariate case, the expected squared error (the generalization error of the MSE loss function) of $\hat{\mathbf{w}}_\lambda$ is defined as:

$$\mathbb{E}\left[\|\hat{\mathbf{w}}_\lambda - \mathbf{w}\|^2\right] = \mathbb{E}\left[(\hat{\mathbf{w}}_\lambda - \mathbf{w})^\top(\hat{\mathbf{w}}_\lambda - \mathbf{w})\right]$$

   where $\mathbf{w}$ is the true parameter vector, and $\hat{\mathbf{w}}_\lambda$ is our estimator.
   (e) The derivative of the generalization error from the previous subquestion with respect to λ is negative at $\lambda = 0$ (in this case, where there is noise in the data). We omit here the proof. Conclude that, if the linear model is correct, a little Ridge regularization helps to reduce this generalization error.

2. Recall that in the recitation, we examined Tikhonov regularization for linear regression:

$$\hat{w}_L^{Tik} = \arg\min_{w\in\mathbb{R}^d} \left(\|Xw-y\|^2 + \|Lw\|^2\right)$$

with some matrix L. In this exercise, we will further explore this regularization and its properties, particularly in comparison with ridge regularization.

(a) Let $X \in \mathbb{R}^{m\times d}$, $y \in \mathbb{R}^m$, and $L \in \mathbb{R}^{k\times d}$. Assume that X^TX is invertible. Find a closed-form solution for the LS problem with Tikhonov regularization for X, y, L as given above. That is, find $\hat{w}_L^{Tik}$ as defined above.

(b) Is the condition that X^TX is invertible a necessary requirement to find the solution derived in question (a)? If not, is there a weaker condition that can be assumed to obtain the same solution?

(c) **Solving Least Squares with and without regularization** - Consider the following data matrix X and the vector of labels y:

$$X = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}, \quad y = \begin{bmatrix} 5 \\ 6 \end{bmatrix}$$

solve the least squares (LS) problem on these data in three different ways:

i. Without regularization

ii. With ridge regularization, using $\lambda = 1$

iii. With Tikhonov regularization, using the given matrix: $L = \begin{bmatrix} 0 & 0 \\ 0 & 2 \end{bmatrix}$

Hint - use the fact that: $\begin{bmatrix} a & b \\ c & d \end{bmatrix}^{-1} = \frac{1}{ad-bc}\begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$

Then, analyze the results:

i. Compare the norms of the three weight vectors: which one is the largest, which one is the smallest, and why?

ii. Compare the absolute values of the second coordinate of the three weight vectors: which one is the largest, which one is the smallest, and why?

1.2 PCA

Based on Lecture 6 and Recitation 7

1. This question is designed to help you understand how samples are projected onto the closest subspace of dimension k in PCA, with respect to minimizing the mean squared reconstruction error. That is, to understand the geometry of the vector $W^\top W(\mathbf{x}_i - \bar{\mathbf{x}}) + \bar{\mathbf{x}}$ alongside the original vector $\mathbf{x}_i$, and see why this distance is the one we want to minimize.

In each part of the question, you are required to: (1) Write the final expression (the relevant vectors or matrices), and (2) Draw the corresponding vectors.

You do not need to show the intermediate steps or how the expressions were calculated. You may use external tools such as Symbolab to assist you. In this exercise, you will work with the following vectors:

$$\mathbf{x}_1 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}, \quad \mathbf{x}_2 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}, \quad \mathbf{x}_3 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}.$$

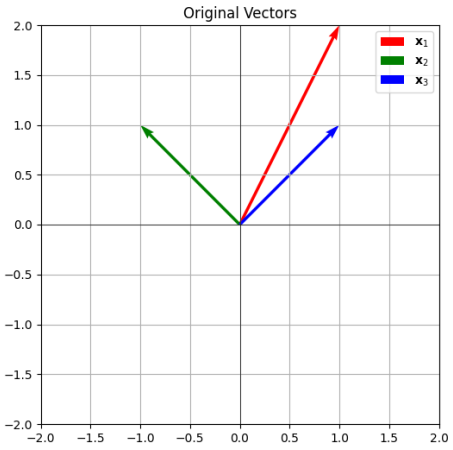

Figure 1: *An illustration of the original vectors. This figure is provided to give you an idea of how your drawings in parts (a)–(e) should look. You may either draw the vectors by hand or write code (e.g., in Python) to plot them. Please make sure to use the same grid, with both the x-axis and y-axis ranging from* -2 *to* 2*.*

(a) Calculate the sample mean: $\bar{\mathbf{x}} = \frac{1}{3}\sum_{i=1}^{3} \mathbf{x}_i$ and draw the centered vectors: $\mathbf{z}_i = \mathbf{x}_i - \bar{\mathbf{x}}$ for each $i \in [3]$.

(b) Calculate the empirical covariance matrix: $\Sigma = \frac{1}{3}\sum_{i=1}^{3} \mathbf{z}_i \mathbf{z}_i^\top$. Then, compute the eigenvector $\mathbf{v}$ corresponding to the largest eigenvalue of Σ, such that $\|\mathbf{v}\| = 1$. Draw the vector $\mathbf{v}$.

(c) Draw the **scalar projections** of each $\mathbf{z}_i$ onto the direction of $\mathbf{v}$.
That is, for each $i \in [3]$ draw $\mathbf{v}^\top \mathbf{z}_i$ on a one-dimensional axis.

(d) For each $i \in [3]$, draw the **projection of** $\mathbf{z}_i$ onto the 1D subspace spanned by $\mathbf{v}$, which is: $\mathbf{v}(\mathbf{v}^\top \mathbf{z}_i)$. This is the closest point to $\mathbf{z}_i$ that lies in the subspace Span$(\mathbf{v})$.

(e) For each $i \in [3]$, we define the **reconstructed version** of $\mathbf{x}_i$ as:

$$\mathbf{v}(\mathbf{v}^\top \mathbf{z}_i) + \bar{\mathbf{x}} = \mathbf{v}(\mathbf{v}^\top (\mathbf{x}_i - \bar{\mathbf{x}})) + \bar{\mathbf{x}}$$

On the same graph, draw the following:

i. The original vectors $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$
ii. The reconstructed vectors from above
iii. The shifted span of $\mathbf{v}$, i.e., the line representing the subspace spanned by $\mathbf{v}$, shifted by the mean - $span(\mathbf{v}) + \bar{\mathbf{x}}$
iv. For each $i \in [3]$, draw the **reconstruction error vectors** by connecting the original vectors $\mathbf{x}_i$ to their corresponding reconstructed vectors $\mathbf{v}(\mathbf{v}^\top \mathbf{z}_i) + \bar{\mathbf{x}}$, using dashed lines to indicate the distance between them.

These vectors represent the difference between each original vector and its projection onto the 1D subspace in the sense of minimizing the mean squared error.

2. In this question, you will show that the problem of minimizing the reconstruction error (the sum of squared errors between the samples and their reconstructions) is equivalent to the problem of maximizing variance. That is, these two problems are equivalent:
Minimizing the sum of squared errors:

$$W^* = \arg \min_{W \in \mathbb{R}^{k \times d}, WW^\top = I_k} \sum_{i=1}^{m} \left\| \mathbf{x}_i - \bar{\mathbf{x}} - W^\top W(\mathbf{x}_i - \bar{\mathbf{x}}) \right\|^2$$

Maximizing projected variance:

$$W^* = \begin{bmatrix} \mathbf{v}_1^\top \\ \vdots \\ \mathbf{v}_k^\top \end{bmatrix} = \arg \max_{\substack{\mathbf{v}_1,\ldots,\mathbf{v}_k \in \mathbb{R}^d \\ \mathbf{v}_i^\top \mathbf{v}_j = \delta_{ij}}} \sum_{j=1}^{k} \mathrm{Var}(\mathbf{v}_j \mathbf{x}) = \arg \max_{\substack{\mathbf{v}_1,\ldots,\mathbf{v}_k \in \mathbb{R}^d \\ \mathbf{v}_i^\top \mathbf{v}_j = \delta_{ij}}} \sum_{j=1}^{k} \mathbf{v}_j^\top \Sigma \mathbf{v}_j$$

- δ_{ij} is the Kronecker delta: it equals 1 if $i = j$ and 0 otherwise. This enforces orthonormality between the rows $\mathbf{v}_1, \ldots, \mathbf{v}_k$.
- $\Sigma \in \mathbb{R}^{d \times d}$ is the empirical covariance matrix of the data, and $\bar{\mathbf{x}}$ is the empirical mean of the data, defined as:

$$\Sigma = \frac{1}{m} \sum_{i=1}^{m} (\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^\top \quad \bar{\mathbf{x}} = \frac{1}{m} \sum_{i=1}^{m} \mathbf{x}_i$$

(a) mark $\mathbf{z}_i = \mathbf{x}_i - \bar{\mathbf{x}}$, and show that if $W^\top W$ is orthogonal projection matrix then:

$$\sum_{i=1}^{m} \left\| \mathbf{x}_i - \bar{\mathbf{x}} - W^\top W(\mathbf{x}_i - \bar{\mathbf{x}}) \right\|^2 = \sum_{i=1}^{m} ||\mathbf{z}_i||^2 - \mathbf{z}_i^\top W^\top W \mathbf{z}_i$$

(b) Show that $\sum_{i=1}^{m} \mathbf{z}_i^\top W^\top W \mathbf{z}_i = m \cdot Tr(W \Sigma W^\top)$

(c) Show that if $W = \begin{bmatrix} \mathbf{v}_1^\top \\ \vdots \\ \mathbf{v}_k^\top \end{bmatrix}$, then $Tr(W \Sigma W^\top) = \sum_{j=1}^{k} \mathbf{v}_j^\top \Sigma \mathbf{v}_j$

(d) Conclude that the two problems stated above are equivalent.

3. Let $X : \Omega \to \mathbb{R}^d$ be a random variable with zero mean and covariance $\Sigma \in \mathbb{R}^{d \times d}$. Show that for any $\mathbf{v} \in \mathbb{R}^d$, where $\|\mathbf{v}\|_2 = 1$, the variance of $\langle \mathbf{v}, X \rangle$ is not larger than the variance obtained by the PCA embedding of X into a one-dimension subspace (assume that the PCA uses the actual Σ).

1.3 Clustering

Based on Lecture 6 and Recitation 7

1. Question 1, 2023 B - Consider the following unsupervised clustering algorithm:

Algorithm 1 Top-Down Clustering

```
1:  procedure TopDownClustering(\{x_i\}_{i=1}^m, m_min)
2:      C_temp ← {{x_1, ..., x_m}}                ▷ Initialize all samples as one cluster
3:      C_final ← ∅
4:      while C_temp ≠ ∅ do
5:          G ← pop(C_temp)                       ▷ Pop the next group of samples
6:          {G_j}_{j=1}^2 ← K-Means(G, 2)         ▷ Cluster group using K-Means into 2 clusters
7:          for i ← 1 to |{G_j}| do
8:              if |G_i| > m_min then
9:                  push(C_temp, G_i)             ▷ Further split large clusters
10:             else
11:                 push(C_final, G_i)            ▷ Otherwise, finalize cluster
12:             end if
13:         end for
14:     end while
15:     return C_final
16: end procedure
```

Where $\{x_i\}_{i=1}^m$ is a set of size m, with each sample $x_i \in \mathbb{R}^d$, and $m_{min} \in \mathbb{N}$. The K-Means procedure receives as input a set of points and the number of clusters to return. It uses the Euclidean distance to measure distances between points, and its output is a collection of sets, each with the points clustered together.

In each of the following, mark whether the statement is True or False, and briefly explain your answer. Note: In the original exam, this specific question did not require an explanation, but for the purpose of this exercise, we ask you to provide a short justification for each answer.

(a) $\forall m_{min} \in \mathbb{N}$, the algorithm might produce a singleton, i.e. a cluster of a single point.

(b) $\forall m \in \mathbb{N}\ \forall \{x_i\}_{i=1}^m$ the algorithm achieves the optimal clustering solution.

(c) The algorithm's variance increases by changing line 5 to $\{G_j\}_j^4 \leftarrow$ K-Means$(G, 4)$.

(d) The algorithm is sensitive to the initialization step of K-Means.

2 Practical Part

You will implement a decision stump (decision tree of depth 1) and the AdaBoost algorithm in the first part, and k-fold-cross-validation for finding the hyperparameter λ of ridge and lasso regularization in the second part.

2.1 Boosting

Implement the following according to their documentation:

- `misclassification_error` function in `loss_functions.py`
- `decision_stump.py`
- `adaboost.py`
- Write code that runs the experiments detailed below in `adaboost_scenario.py`

Note:

- The `decision_stump.py` class selects a feature of the design matrix X and a corresponding threshold value that achieve the best classification result (minimize misclassification error) when thresholding only with regard to that feature.
- The `adaboost.py` class is parameterized by a weak learner (`wl`, like the decision stump above) and the number of weak learners to use (`iterations`). Its solution is described by a list of fitted weak learner models (`models_`), their weights (`weights_`), and the derived distribution over samples (`D_`, enhanced for samples we are wrong about).
- Note that there are two different ways to incorporate the distribution D over the training samples: either by resampling the data according to D, or by training and evaluating using weighted loss with the weights D_i assigned to each sample. These are distinct approaches, both of which are valid for use in this exercise. For more details, see the note at the end of the exercise.
- Also, notice that `predict`/`loss` are specific cases of `partial_predict`/`partial_loss` that return prediction/loss evaluation when using all learners.
- Note that if there exists a decision stump with zero error, then the entire AdaBoost algorithm becomes unnecessary, as the problem can be solved immediately. You may ignore such a case in this exercise.

Packages:

- You may use the packages: `numpy`, `plotly`, `matplotlib`, `itertools`
- You may use functions imported in the provided code
- Usage of any other package is **not allowed**!

Then, in the file `adaboost_scenario.py`, answer the following questions:

1. Use the provided `generate_data` function to generate 5000 train samples and 500 test samples, with no noise (i.e. `noise_ratio = 0`). Train an Adaboost ensemble of size 250 (i.e passing 250 as the number of iterations) using your implementation of the `DecisionStump` as weak learner. Plot, in a single figure, the training- and test errors as a function of the number of fitted learners (i.e. use the `partial_loss` function for $t = 1, \ldots, 250$). Explain your results.

2. Using the previously fitted ensemble, plot the decision boundary obtained by using the ensemble up to iteration 5, 50, 100 and 250. Use your implementation of the `partial_predict` to obtain the predictions of the ensemble up to the specified size. In each of these plots also add the test set (colored and/or shaped by the actual labels). Explain your results.

   You can use the `decision_surface` function in `utils.py`. Revisit lab 05 of comparing classifiers to see how to create such plots.

3. Using the test evaluation scores per number of learners, which ensemble size achieved the lowest test error? Plot the decision surface of that ensemble as well as the test set data points. In the plot's title provide the ensemble size and its accuracy.

4. Using the previously fitted ensemble, use the weights of the last iteration (i.e. $\mathcal{D}$ at T) to plot the *training* set with a point size proportional to its weight and color (and/or shape) indicating its label.
   - As previously, plot the decision surface (using the full ensemble).
   - As the weights are of very small numbers normalize and transform as follow: `D = D/np.max(D) * 5`

   Explain your results, and specifically explain which samples are "easier" and which are more "challenging" for the classifier.

5. Repeat the steps above (while avoiding code repetition) for train- and test sets generated with noise levels of 0.4. Show graphs as in question (1) and question (4). Explain the results. In your answer explain what is seen in the plot of the loss in terms of the bias-variance tradeoff.

2.2 Choosing Regularization Parameters Using Cross Validation

Here, we will compare the Lasso and Ridge regularization over a dataset of glucose levels of diabetes patients, using cross-validation.

- Implement the `RidgeRegression` class in `estimators.py` as specified in the class documentation. You are not allowed to use the scikit-learn implementation of Ridge Regression.
- Note that in `RidgeRegression`, you should provide an option to include an intercept (a boolean argument in the constructor), but the regularization should not be applied to the intercept term.
- Implement the `loss` function of `Lasso` and `LinearRegression` in `estimators.py`, as specified in the documentation (MSE loss function).
- In the rest of the exercise, use your implemented `RidgeRegression` class, alongside `Lasso` and `LinearRegression` estimators based on scikit-learn's implementation provided in `estimators.py`.

Then, implement the `cross_validate` function in `cross_validate.py` according to function documentation:

- Use the pseudo-code from the recitation; that is, in each iteration of the Cross Validation, split the data into a training set and a validation set (excluding the test set at this stage).
- You are *not allowed* to use sklearns Cross-Validation implementation.

In `perform_model_selection.py`, implement and answer the following:

1. Load the diabetes dataset and split it to a training and testing portion, using only 50 random samples for training (So, we can see the strength of k-fold cross-validation on such a small training set). The remaining samples should be used as a fixed test set for Question 3.

2. For both Ridge and Lasso regularizations (both with intercept), run 5-Fold Cross-Validation using different values of the regularization parameter λ. Explore possible ranges of values of λ (say, take `n_evaluations=500` equally spaced values in a range of your choice). Which range of values is meaningful for the given data for each of the algorithms?

   Then, over these selected ranges, for each of the two algorithms, plot the train and validation errors as a function of the tested regularization parameter value. Explain your results. Address both the differences between the train and validation errors as well as the differences in the plots for the two algorithms.

3. Report which regularization parameter values achieved the best validation errors for the Ridge and Lasso regularizations. Then fit the entire train set (of size 50) using these values for Ridge, Lasso and Least Squares (LinearRegression estimator in `estimators.py`) regressions. That is, train Ridge with its optimal λ, Lasso with its optimal λ, and standard Least Squares. Report the test errors of each of the fitted models.

Resampling vs. Reweighting in AdaBoost

In the AdaBoost algorithm, two common approaches for incorporating the sample weights $D^{(t)}$ into the learning process are **resampling** and **reweighting**. Both approaches affect *both the training and evaluation* stages of the weak learners.

- **Resampling**:
  - **Training:** A new dataset is sampled from the original dataset (in the same size of the original), using the current distribution $D^{(t)}$. The weak learner is trained only on this sampled dataset.
  - **Evaluation:** The model's error ε_t is computed on the same resampled dataset. Since the samples are drawn according to $D^{(t)}$, this implicitly reflects the weighted error. In our case, it means that $\varepsilon_t = \frac{1}{m}\sum_{i=1}^{m} \mathbb{1}_{[y_i \neq h_t(\mathbf{x}_i)]}$ with the resampled data $\{x_i, y_i\}_{i=1}^{m}$.
- **Reweighting**:
  - **Training:** The weak learner is trained on the *original* dataset, but each sample i is weighted by $D_i^{(t)}$. For example, a decision stump minimizes the weighted misclassification error: $\sum_{i=1}^{m} \mathcal{D}^{(t)} \mathbb{1}_{[y_i \neq h_t(\mathbf{x}_i)]}$
  - **Evaluation:** The error ε_t is computed on the full dataset using the same sample weights $D^{(t)}$ and the weighted misclassification metric. In our case, it means that $\varepsilon_t = \sum_{i=1}^{m} \mathcal{D}^{(t)} \mathbb{1}_{[y_i \neq h_t(\mathbf{x}_i)]}$ with the original data $\{x_i, y_i\}_{i=1}^{m}$.

Both approaches are valid and can be used in this exercise. The **resampling approach** was discussed in class, while the **reweighting technique** was provided in the recitation.