10907 Pattern Recognition

Lecturers
Prof. Dr. Ivan Dokmanić ⟨ivan.dokmanic@unibas.ch⟩

Tutors
Alexandra Flora Spitzer ⟨alexandra.spitzer@stud.unibas.ch⟩
Roman Fries ⟨r.fries@unibas.ch⟩
Cheng Shi ⟨cheng.shi@unibas.ch⟩
Vinith Kishore ⟨vinith.kishore@unibas.ch⟩
Valentin Debarnot ⟨valentin.debarnot@unibas.ch⟩

Assignment 2

Important:
Deadline: 27.10.2023
Total points: 6

Summary

The math part in this assignment contains exercises related to the materials in the lecture notes.

Instruction for submission We use Gradescope to submit and evaluate the assignments. You should have received an invitation to the Gradescope course page. Go to the course page `10907 Pattern Recognition`, you will see two assignment items, `Assignment 2: Coding` and `Assignment 2: Math`.

- For the Math part of this assignment, you upload the solution as a *single* `.pdf` file in the `Assignment 2: Math`. For each exercise, you must to specify which pages containing the answer.
- For the Coding part of this assignment, you will find a code template in our course repo[1]. The code template contains the functions to be completed. You upload the completed file to the `Assignment 2: Coding`.

Academic integrity You are encouraged to discuss the material with others, but it is essential to document these discussions by writing down the names of the individuals you worked with and any tools that helped you with the assignment. Any instance of cheating will be dealt extremely strictly, potentially resulting in receiving zero credit for the course at a minimum.

[1] `https://git.scicore.unibas.ch/dokmanic-courses/pr23/-/tree/main/assignments/assignment2`

Math

Exercise 1 (1 points)

In the context of multivariate linear regression, the residual sum of squares (RSS) is described by the following equation:

$$\text{RSS}(\mathbf{w}) = \frac{1}{2}\|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2 \tag{1}$$

Where:

- $\mathbf{X}$ represents the $n \times d$ data matrix, containing d-dimensional features for n observations.
- $\mathbf{w}$ stands for the weight vector.
- $\mathbf{y}$ is the vector of target values for each training sample.

Tasks:

1. **Derivation of Partial Derivative**: Prove the following equation:

$$\frac{\partial \text{RSS}(\mathbf{w})}{\partial w_k} = \|\mathbf{x}_{\cdot,k}\|_2^2 w_k - \mathbf{x}_{\cdot,k}^T(\mathbf{y} - \mathbf{X}_{\cdot,-k}\mathbf{w}_{-k})$$

   Note that:

   - $\mathbf{x}_{\cdot,k}$ denotes the k-th column of $\mathbf{X}$.
   - $\mathbf{X}_{\cdot,-k}$ represents the data matrix $\mathbf{X}$ excluding the k-th column.
   - $\mathbf{w}_{-k}$ is the weight vector without the k-th component.

   Hint: Break down the weights into the k-th component and the others.

2. **Derivation of Optimal Weight**: We define the residual as:

$$\mathbf{r}(k) = \mathbf{y} - \mathbf{X}_{\cdot,-k}\mathbf{w}_{-k}$$

   Prove that if $\frac{\partial \text{RSS}}{\partial w_k} = 0$, then the optimal weight for the k-th feature can be described as:

$$\hat{w}_k = \frac{\mathbf{x}_{\cdot,k}^T \mathbf{r}(k)}{\|\mathbf{x}_{\cdot,k}\|^2}. \tag{2}$$

   Interpretation: This indicates that when features are added sequentially, the optimal weight for the k-th feature is determined by projecting $\mathbf{x}_{\cdot,k}$ onto the existing residuals.

Exercise 2 (1 point)

In many applications, the classifier is allowed to "reject" a test example rather than classifying it into one of the classes. Consider, for example, a case in which the cost of a misclassification is \$10 and the reward (negative cost) of correct prediction is \$0.5. On the other hand, the cost of an additional human evaluation is only \$3. We can summarize this by the following loss matrix

Decision $\hat{Y}$	true label Y	
	Y=0	Y=1
predict $\hat{Y} = 0\|x$	-0.5	10
predict $\hat{Y} = 1\|x$	10	-0.5
reject	3	3

1. Let us denote $p_1 = p(Y = 1|x)$. Show that in general, for this loss matrix, but for any posterior distribution, there will be two thresholds θ_0 and θ_1 such that the optimal decision is to predict $\hat{Y} = 0$ if $p_1 < \theta_0$, reject if $\theta_0 < p_1 < \theta_1$ and predict 1 if $p_1 > \theta_1$.

2. Compute θ_0 and θ_1.

3. Sometime, you can choose to **double** the consequence of your decision (for example when you are super-confident), which means your costs become -1 for the correct and 20 for the wrong prediction. Compute the new threshold θ_2, θ_3 such that the optimal decision is to **double** the consequence of your prediction for $\hat{Y} = 1$ when $p_1 > \theta_2$, or for $\hat{Y} = 0$ when $p_1 < \theta_3$.

Exercise 3 (2 point)

A ROC illustrates the diagnostic ability of a binary classifier as its discrimination threshold is varied. The ordinate in a ROC is the true positive rate (TPR) while the abscissa is the false positive rate (FPR). In real-world tasks, curves are usually drawn using a finite number of test samples $\{x_i\}_{i=1}^n$, which yields a non-smooth appearance. In this case, we define the ROC as the piecewise linear curve formed by connecting points $\{(a_1, b_1), (a_2, b_2), \ldots, (a_m, b_m)\}$ in sequence, with $a_1 \leq a_2 \leq \cdots \leq a_m$, and $b_1 \leq b_2 \leq \cdots \leq b_m$.

1. Let's consider a **rank binary classifier**. This classifier first computes a score for each sample by some function $f(\cdot)$. Then, for some threshold η, the classifier assigns $Y_i = 1$ for x_i if $f(x_i) > \eta$ while $Y_i = -1$ if $f(x_i) < \eta$. Find a way to compute $\{(a_1, b_1), (a_2, b_2), \ldots, (a_m, b_m)\}$ with classifier function $f(\cdot)$, dataset $\{x_i\}_{i=1}^n$ and true labels $\{y_i\}_{i=1}^n \in \{\pm 1\}^n$.

2. Let's assume there are n^+ positive samples and n^- negative samples ($n^+ + n^- = n$), and we use D^+ and D^- to represent the *sets* of positive samples and negative samples. The loss l of the rank binary classifier is defined as

$$l = \frac{1}{n^+ n^-} \sum_{x^+ \in D^+} \sum_{x^- \in D^-} \mathbb{1}\left(f(x^+) < f(x^-)\right).$$

Assuming that every sample has a distinct score $f(x_i)$, that is $f(x_i) \neq f(x_j)$ for $i \neq j$, show that

$$\text{AUC} = 1 - l,$$

where AUC is the area under the ROC curve.

Coding

Exercise 4 (Evaluating a classifier - 2 points)

Once a classifier is trained, it is crucial to have specific metrics for evaluating its performance and for comparing different classifiers. A standard procedure for evaluating a classifier involves initially splitting the data into training and test sets. The training set is used to optimize the classifier, while the test set is used to assess the accuracy of the classifier. This process allows us to potentially compare different classifiers and select the most suitable one for a specific application. However, accuracy alone may not always be the most appropriate metric for comparing classifiers.

Accuracy can be misleading, especially when dealing with heavily imbalanced datasets or in cases where the cost of false positives is substantial, such as in medical diagnosis, leading to wasteful expenses for the patients. Similarly, in email spam detection, missing a few spam emails is often less critical than classifying non-spam emails as spam. To address these issues, we compute various other metrics.

Your task is to complete several functions in `metrics.py` which are used to evaluate classifiers. We will focus on binary classification problems, where the positive label is represented by $+1$ and the negative label is encoded as -1.

1. `accuracy()`: This function calculates accuracy based on the test and true labels. Accuracy is defined as:
$$\text{accuracy} = \frac{1}{N}\sum_{i=1}^{N} \mathbb{1}_{y_{\text{True}}[i]=y_{\text{Predict}}[i]}$$

2. `precision()`: This function computes precision using the true and predicted labels. Precision is defined as:
$$\text{precision} = \frac{\text{number of true positives}}{\text{number of true positives} + \text{number of false positives}}.$$

3. `recal()`: The recall function calculates the sensitivity of the classifier and is defined as:
$$\text{recall} = \frac{\text{number of true positives}}{\text{number of true positives} + \text{number of false negatives}}.$$

4. `precision_recall_curve()`: This function uses the score value of the classifier, which represents the predicted probability or the value before thresholding to +1 or -1, and the true labels. It outputs the precision, recall, and the thresholds used to obtain the precision-recall values. This information can be used to plot the precision-recall curve. The steps for this function include:

   (a) Arrange the predicted scores in ascending order.

   (b) Choose unique threshold values from the predicted scores.

   (c) For each threshold value, obtain predictions
   $$y_{\text{Predict}}[i] = 2\mathbb{1}_{y_{\text{Score}}[i]\geq\text{threshold}} - 1$$

   (d) Compute precision and recall for the predictions and store them.

   (e) Return the lists of precision and recall.

   Hint: The output should be same as the sklearns precision recall curve. Carefully check the end and start points with the sklearn output. You might need to add the start and end points manually. We will not evaluate the threshold output as these can be a bit arbitrary.

5. `true_positive_rate()`: Computes the true positive rate and is defined as :

$$\text{tpr} = \frac{\text{number of true positives}}{\text{number of true positives} + \text{number of false negatives}}.$$

   Note that this is the same as recall.

6. `false_positive_rate()`: Computes the false positive rate and is defined as :

$$\text{fpr} = \frac{\text{number of false positives}}{\text{number of false positives} + \text{number of true negatives}}.$$

7. `roc_curve()`:is a function which outputs the tpr (y-axis) and fpr (x-axis) and the thresholds using the predicted score and the true lables. This can be use to plot the Reciever Operating Characteristics (ROC) curve. Similar to precision recal curve function:

   (a) Sort the score value and obtain unique thresholds

   (b) Compute the predicted labels for each of the threshold

   (c) Compute true positive rate and false poitive rate for each predicitions.

   (d) return the list of true positive rate and false poitive rate

   Hint: The output should be same as the sklearn's ROC curve.