Introduction to Machine Learning (67577)

Exercise 3
Classification

Second Semester, 2022

Contents

1 Submission Instructions

Please make sure to follow the general submission instructions available on the course website. In addition, for the following assignment, submit a single `ex3_ID.tar` file containing:

- An `Answers.pdf` file with the answers for all theoretical and practical questions (include plotted graphs *in* the PDF file).
- The following python files (without any directories): `loss_functions.py`, `perceptron.py`, `linear_discriminant_analysis.py`, `gaussian_naive_bayes.py`, `classifiers_evaluation.py`

The `ex3_ID.tar` file must be submitted in the designated Moodle activity prior to the date specified *in the activity*.

- Late submissions will not be accepted and result in a zero mark.
- Plots included as separate files will be considered as not provided.
- Do not forget to answer the Moodle quiz of this assignment.

2 Theoretical Part

2.1 Hard- & Soft-SVM

Based on Lecture 3 and Recitation 5

1. Prove that following Hard-SVM optimization problem is a Quadratic Programming problem:

$$\underset{(\mathbf{w},b)}{\operatorname{argmin}} ||\mathbf{w}||^2 \quad \text{s.t.} \quad \forall i\; y_i\left(\langle \mathbf{w},\mathbf{x}_i\rangle + b\right) \geq 1 \tag{1}$$

That is, find matrices Q and A and vectors $\mathbf{a}$ and $\mathbf{d}$ such that the above problem can be written in the following format

$$\underset{\mathbf{v}\in\mathbb{R}^n}{\operatorname{argmin}} \tfrac{1}{2}\mathbf{v}^\top Q\mathbf{v} + \mathbf{a}^\top\mathbf{v} \quad \text{s.t.} \quad A\mathbf{v} \leq \mathbf{d} \tag{2}$$

Hint: Observe that $||\mathbf{w}||^2 = \mathbf{w}^\top\mathbf{I}\mathbf{w}$

2. Consider the Soft-SVM optimization problem:

$$\underset{\mathbf{w},\{\xi_i\}}{\operatorname{argmin}} \frac{\lambda}{2}||\mathbf{w}||^2 + \frac{1}{m}\sum_i \xi_i \quad \text{s.t.} \quad \forall i\; y_i\langle \mathbf{w},\mathbf{x}_i\rangle \geq 1-\xi_i \wedge \xi_i \geq 0 \tag{3}$$

Denote the hinge-loss function as $\ell^{hinge}(a) := \max\{0, 1-a\}$. Show that the Soft-SVM optimization problem is equivalent to the following unconstraint optimization problem:

$$\underset{\mathbf{w},\{\xi_i\}}{\operatorname{argmin}} \frac{\lambda}{2}||\mathbf{w}||^2 + \frac{1}{m}\sum_i \ell^{hinge}\left(y_i\langle \mathbf{w},\mathbf{x}_i\rangle\right) \tag{4}$$

2.2 Naive Bayes Classifiers

Based on Lecture 3 and Recitation 6. Let $\mathcal{X}$ be a domain set and $\mathcal{Y}\in[K], K\in\mathbb{N}$ the response set and let us assume there exists a joint probability distribution $\mathcal{D}$ over $\mathcal{X}\times\mathcal{Y}$ with $f_{\mathcal{D}}$ the joint probability distribution function.

Recall the Bayes Optimal Classifier which predicts the response maximizing the posterior distribution:

$$\hat{y}^{MAP} := \underset{k\in[K]}{\operatorname{argmax}} f_{Y|X=\mathbf{x}}(k) = \underset{k\in[K]}{\operatorname{argmax}} \frac{f_{X|Y=k}(\mathbf{x}) f_Y(k)}{f_X(\mathbf{x})} \tag{5}$$

Naive Bayes classifiers are a family of classifiers realizing the Bayes Optimal classifier where we assume that all features are *independent*. That is, for $\mathbf{x} \sim \mathcal{P}$ then $f_{X_i,X_j}(x_i,x_j) = f_{X_i}(x_i) f_{X_j}(x_j) \quad \forall i,j$.

3. The ***Gaussian*** **Naive Bayes** classifier assumes a multinomial prior and independent feature-wise Gaussian likelihoods:

$$\begin{aligned} y &\sim \text{Multinomial}(\pi) \\ x_j|y=k &\overset{ind.}{\sim} \mathcal{N}\left(\mu_{kj}, \sigma^2_{kj}\right) \end{aligned} \tag{6}$$

   for π a probability vector: $\pi \in [0,1]^K, \sum \pi_j = 1$.
   (a) Suppose $x \in \mathbb{R}$ (i.e each sample has a single feature). Given a trainset $\{(x_i,y_i)\}_{i=1}^m$ fit a Gaussian Naive Bayes classifier solving (5) under assumptions (6).
   (b) Suppose $\mathbf{x} \in \mathbb{R}^d$ (i.e each sample has d feature). Given a trainset $S = \{(\mathbf{x}_i,y_i)\}_{i=1}^m$ fit a Gaussian Naive Bayes classifier solving (5) under assumptions (6). You are encouraged to use the results from (3.a).
4. The ***Poisson*** **Naive Bayes** classifier assumes a multinomial prior and independent feature-wise Poisson likelihoods:

$$\begin{aligned} y &\sim \text{Multinomial}(\pi) \\ x_j|y=k &\overset{ind.}{\sim} \text{Poi}\left(\lambda_{kj}\right) \end{aligned} \tag{7}$$

   for π a probability vector: $\pi \in [0,1]^K, \sum \pi_j = 1$.
   (a) Suppose $x \in \mathbb{R}$ (i.e each sample has a single feature). Given a trainset $\{(x_i,y_i)\}_{i=1}^m$ fit a Gaussian Naive Bayes classifier solving (5) under assumptions (7).
   (b) Suppose $\mathbf{x} \in \mathbb{R}^d$ (i.e each sample has d feature). Given a trainset $S = \{(\mathbf{x}_i,y_i)\}_{i=1}^m$ fit a Poisson Naive Bayes classifier solving (5) under assumptions (7). You are encouraged to use the results from (4.a).

3 Practical Part

In the following part you will implement and compare different classifiers for different data scenarios. Be sure to have pulled the latest version of the GreenGilad/IML.HUJI repository.

3.1 Perceptron Classifier

Based on Lecture 3 and Recitation 5. Complete the following implementations

- Implement the `misclassification_error` function in the `metrics/loss_functions.py` file as described in the function documentation.
- Implement the Perceptron algorithm in the `learners/classifiers/perceptron.py` file as described in the class documentation. In toy implementation use the misclassification error implemented above.

In the `exercises/classifiers_evaluation.py` file, implement the `run_perceptron` function as described in documentation.

- To retrieve the loss at each iteration **do not** change the previously implemented Perceptron class. Instead **specify a callback function** which receives the object and uses it's `loss` function to calculate the loss over the training set. Store these values in an array to be used for plotting.

1. Fitting and plotting over the `datasets/linearly_seprable.npy` dataset, what can we learn from the plot?
2. Next run the Perceptron algorithm over the `datasets/linearly_inseprable.npy` dataset and plot its loss as a function of the iterations. What is the difference between this plot and to the one in the previous question? How can we explain the difference in terms of the objective and parameter space?

3.2 Bayes Classifiers

Based on Lecture 3 and Recitation 6. Complete the following implementations

- Implement the `accuracy` function in the `metrics/loss_functions.py` file as described in the function documentation.
- Implement the `LDA` classifier in the `learners/classifiers/linear_discriminant_analysis.py` file as described in the class documentation. Use expressions derived in class.
- Implement the `GaussianNaiveBayes` classifier in the `learners/classifiers/gaussian_naive_bayes.py` file as described in the class documentation. Use expressions derived in question 3b of the theoretical part.

Then, implement and answer the following questions:

1. In the `compare_gaussian_classifiers` function, `classifiers_evaluation.py` file, load the `datasets/gaussians1.npy` dataset. Fit both the Gaussian Naive Bayes and LDA algorithms previously implemented. Plot the following:
   - A single figure with two subplots:
     (a) 2D scatter-plot of samples, with marker color indicating Gaussian Naive Bayes *predicted* class and marker shape indicating *true* class.
     (b) 2D scatter-plot of samples, with marker color indicating LDA *predicted* class and marker shape indicating *true* class.
     (c) Provide classifier name and accuracy (over train) in sub-plot title
   - For both subplots add:
     (a) Markers (colored black and shaped as 'X') indicating the center of fitted Gaussians.
     (b) An ellipsis (colored black) centered in Gaussian centers and shape dictated by fitted covariance matrix.
   - Specify dataset name in figure title.

   Explain what can be learned from the plots above regarding the distribution used to sample the data?

2. Repeat the procedure above (while avoiding code repetition) for `datasets/gaussians2.npy`. What is the difference between the two scenarios? What can be learned regarding the distribution used to sample the data? Which of the two classifiers better matches this dataset and why?