Exercise 3

Deadline: 08.12.2021, 16:00.
Ask questions to #ask-your-tutor-viktor

Last exercise you implemented LDA yourself. In this exercise, you'll take a look under the hood of LDA: You will work on a step-by-step derivation of LDA from the Least Squares Error. In the second part of the exercise you will implement a Naive Bayes Classifier on the digits dataset.

Regulations

Regarding this week, hand-in your LDA derivation (task 1) as a PDF file `LDA.pdf`, created with LaTeX or another tool of your liking, or scanned-in from a (readable!) hand-written solution on paper. The solution to task 2 goes into `naive-bayes.ipynb` and `naive-bayes.html`.

Zip all files into a single archive `ex03.zip` and upload this file to your assigned tutor on MaMPF before the given deadline.
Note: Each team creates only a single upload, and all team members must *join* it as described in the MaMPF documentation at `https://mampf.blog/zettelabgaben-fur-studierende/`.
Important: Make sure that your MaMPF name is the same as your name on Muesli. We now identify submissions purely from the MaMPF name. If we are unable to identify your submission you will not receive points for the exercise!

1 LDA-Derivation from the Least Squares Error (24 points)

The goal of this exercise is to derive closed-form expressions for the optimal parameters $\hat{w}$ and $\hat{b}$ in Linear Discriminant Analysis, given some training set with two classes. Remember that the decision boundary in LDA is given by a $D-1$ dimensional hyperplane (where D is the dimension of the feature space) that we parametrize via

$$\hat{w}^T x + \hat{b} = 0. \tag{1}$$

$\hat{w}$ is the hyperplane's normal vector and $\hat{b}$ a scalar fixing its position in the D-dimensional space. **Note that w and x are column vectors in this exercise.** The decision rule for our two classes at query point x is then

$$\hat{y} = \text{sign}(\hat{w}^T x + \hat{b}) = \begin{cases} 1, & \text{if } \hat{w}^T x + \hat{b} > 0 \\ -1, & \text{if } \hat{w}^T x + \hat{b} < 0 \end{cases} \tag{2}$$

In the training phase we are given N datapoints $\{x_i\}_{i\in 1,\dots,N}$ with $x_i \in \mathbb{R}^D$ and their respective labels $\{y_i\}_{i\in 1,\dots,N}$ with $y_i \in \{-1, 1\}$. We assume that the training set is balanced, i.e.

$$N_1 = N_{-1} = \frac{N}{2} \tag{3}$$

with N_k denoting the number of instances in either class. The optimal parameters $\hat{w}$ and $\hat{b}$ are now the ones minimizing the least squares error criterion:

$$\hat{w}, \hat{b} = \text{argmin}_{w,b} \sum_{i=1}^{N} \left(w^T x_i + b - y_i\right)^2. \tag{4}$$

You shall solve this problem in three steps:

1.1 (4 points)

First, compute $\hat{b}$ from

$$\frac{\partial}{\partial b}\sum_{i=1}^{N}\left(w^T x_i + b - y_i\right)^2 = 0\ . \tag{5}$$

1.2 (16 points)

Second, use this result to reshuffle

$$\frac{\partial}{\partial w}\sum_{i=1}^{N}\left(w^T x_i + \hat{b} - y_i\right)^2 = 0\ . \tag{6}$$

into the intermediate equation

$$\left(S_W + \frac{1}{4}S_B\right)\hat{w} = \frac{\mu_1 - \mu_{-1}}{2}. \tag{7}$$

Here, μ_1 and μ_{-1} are the class means

$$\mu_{-1} = \frac{1}{N_{-1}}\sum_{i:\ y_i=-1} x_i \tag{8}$$

$$\mu_1 = \frac{1}{N_1}\sum_{i:\ y_i=1} x_i \tag{9}$$

and S_B and S_W are the between-class and within-class covariance matrices

$$S_B = (\mu_1 - \mu_{-1})(\mu_1 - \mu_{-1})^T \tag{10}$$

$$S_W = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu_{y_i})(x_i - \mu_{y_i})^T\ . \tag{11}$$

The notation μ_{y_i} means

$$\mu_{y_i} = \begin{cases}\mu_{-1} & \text{if } y_i = -1\\ \mu_1 & \text{if } y_i = 1\end{cases} \tag{12}$$

During your calculations you may find the following relation for general vectors a, b and c useful:

$$a \cdot (b^T \cdot c) = (a \cdot c^T) \cdot b \tag{13}$$

1.3 (4 points)

Finally, transform equation (7) into

$$\hat{w} = \tau\ S_W^{-1}(\mu_1 - \mu_{-1}) \tag{14}$$

where τ is an arbitrary positive constant, expressing the fact that $\text{sign}(\tau(\hat{w}^T x + \hat{b}))$ is the same decision rule as $\text{sign}(\hat{w}^T x + \hat{b})$.

2 Naive Bayes Classifier (16 points)

We will once again work with the digits dataset, this time using the naive Bayes classifier. If we assume that all classes are equally likely, i.e. the priors are $p(y = k) = 1/C$ with C the number of classes, the decision rule is defined by

$$\hat{y} = \text{argmax}_k \left(\prod_{j=1} p_j(x_j|y = k) \right) \tag{15}$$

where $p_j(x_j|y = k)$ are 1-dimensional histograms (or other suitable 1-dimensional probability estimators) for each feature j and class k. Since the product of many small probabilities is a tiny number prone to numerical inaccuracy, it is better to rewrite this as a sum of logarithms:

$$\hat{y} = \text{argmax}_k \left(\sum_{j=1} \log p_j(x_j|y = k) \right) \tag{16}$$

Implement training of the naive Bayes classifier as a function

```
histograms, binning = fit_naive_bayes(features, labels, bincount)
```

where `histograms` is the $C \times D \times L$ array of histograms (D is the number of feature dimensions, L the number of bins), and `binning` is a $C \times D \times 2$ array describing the bin layout. Specifically, `binning[k,j,0]` is the lower bound of the first bin for class k and feature j, and `binning[k,j,1]` is the corresponding bin width. The data in `binning` is needed during predicition to find out in which bin l_{kj} a new instance i belongs:

$$l_{kj} = \left\lfloor \frac{X_{ij} - \text{binning}[k, j, 0]}{\text{binning}[k, j, 1]} \right\rfloor \tag{17}$$

where $\lfloor . \rfloor$ is the floor function, i.e. rounding down. When `fit_naive_bayes()` is called with `bincount=0`, the number of bins shall be determined automatically according to the Freedman-Diaconis rule. However, this rule recommends a bin *width* Δx_{kj} for each feature j in class k, and the corresponding bin *count* must be computed by the equation

$$\#\text{bins}_{kj} = \left\lceil \frac{\max_{i:\ y_i=k}(X_{ij}) - \min_{i:\ y_i=k}(X_{ij})}{\Delta x_{kj}} \right\rceil \tag{18}$$

where $\lceil . \rceil$ is the ceiling function, i.e. rounding up. Thus, the rule may suggest a different number of bins for every feature, whereas we want a fixed bin count L for simplicity. You should therefore choose L as a suitable compromise between the different recommendations of the Freedman-Diaconis rule. For features that need fewer bins than your choice of L, you may just fill the first $L' < L$ bins and leave the rest empty. For features where the rule recommends more bins, you must increase Δx_{kj} so that the data fits in L bins.

Now implement prediction as a function

```
predicted_labels = predict_naive_bayes(test_features, histograms, binning)
```

Filter the training set to use only digits "3" and "9", train the classifier with

- the 2-dimensional feature space you constructed in exercise 2 (with `bincount=0`), and
- the full 64-dimensional feature space (one histogram per pixel with `bincount=8`)

and determine the two confusion matrices on the test set. Visualize the decision boundary for the 2-D case, overlayed with the scatterplot of the test data.