Systems analysis and decision support methods in Computer Science

Lab – Python – Assignment 2

κ-NN and Naive Bayes

authors: M. Zięba, J.M. Tomczak, A. Gonczarek, S. Zaręba translation: P. Klukowski

Assignment goal

The goal of this assignment is to implement κ-NN and Naive Bayes classifiers for text document analysis.

Text document classification

Consider a problem of assigning a text document $\mathcal{T}$ to one of predefined topics y (for instance $y = 1$ corresponds to *computer*, 2 – *recreation*, 3 – *science*, 4 – *talk*). Each document is represented by feature vector $\mathbf{x} = (\phi^1(\mathcal{T}), \ldots, \phi^D(\mathcal{T}))^{\mathrm{T}}$, where each binary feature $\phi^d(\mathcal{T}) \in \{0, 1\}$ indicates whether d-th word is present in the document ($\phi^d(\mathcal{T}) = 1$) or absent ($\phi^d(\mathcal{T}) = 0$) in the document.

When classifying a new text document $\mathbf{x}^{new}$ into one of the topic categories y, we would like to estimate probability $p(y|\mathbf{x}^{new})$, and then select the most probable class:

$$y^* = \arg\max_y p(y|\mathbf{x}^{new}). \tag{1}$$

Therefore a conditional probability $p(y|\mathbf{x})$ is a key factor in the classification problem, which undergoes modeling. There are two distinct approaches that allows to model $p(y|\mathbf{x})$:

- **Generative approach**: conditional distribution $p(y|\mathbf{x})$ is derived from Bayes theorem:

$$\begin{aligned} p(y|\mathbf{x}) &= \frac{p(\mathbf{x}|y)p(y)}{p(\mathbf{x})} \\ &= \frac{p(\mathbf{x}|y)p(y)}{\sum_{y'} p(\mathbf{x}|y')p(y')} \end{aligned}$$

  To calculate conditional distribution $p(y|\mathbf{x})$ we model $p(\mathbf{x}|y, \theta)$ and $p(y|\pi)$, where θ and π represent model parameters.

- **Discriminative approach**: conditional distribution $p(y|\mathbf{x})$ is modeled directly from $p(y|\mathbf{x}, \theta)$, where θ represents model parameters.

Generative approach: Naive Bayes

Model In generative approach the goal is to model distributions $p(\mathbf{x}|y,\theta)$ and $p(y|\pi)$. The latter one (distribution of text topic categories) can be easily modeled using categorical pdf:

$$p(y|\pi) = \mathrm{Cat}(y|\pi), \tag{2}$$

where $\pi = (\pi_1, \ldots, \pi_K)$ and π_k represents *a priori* probability for k-th topic group.

In document classification problem, features that describe a document content are binary. In result an appropriate distribution is the one, which assign probability to each possible sequence of words occurrences. Note that in general case there are 2^D such combinations, so the corresponding model would have $2^D - 1$ parameters, what leads to severe limitations. For example, if $D \geqslant 100$ training such a model is impossible in practice.

To cope with this problem, we assume that words appear in the text independently to each other. In such case the model has only D parameters. By making this assumption, we lose chance to model dependencies between words, but at the same time we gain ability to follow a training procedure.

The model that assumes feature independence is called **Naive Bayes** and has the following form:

$$p(\mathbf{x}|y,\theta) = \prod_{d=1}^{D} p(x_d|y,\theta) \tag{3}$$

for text classification purposes, conditional distribution can be modeled using Bernoulli distribution:

$$p(x_d|y=k,\theta) = \mathrm{Ber}(x_d|\theta_{d,k}) \tag{4}$$

$$= \theta_{d,k}^{x_d}(1-\theta_{d,k})^{1-x_d}. \tag{5}$$

Training The goal of Naive Bayes training procedure is to estimate $\{\pi_k\}_{k=1\ldots,4}$ and $\{\theta_{d,k}\}_{\substack{d=1,\ldots,D \\ k=1,\ldots,4}}$ using training data $\mathcal{D}$.

Using Maximum Likelihood Estimation (MLE) we can calculate these values using following formulas:

$$\pi_k = \frac{1}{N}\sum_{n=1}^{N} \mathbb{I}(y_n = k), \tag{6}$$

$$\theta_{d,k} = \frac{\sum_{n=1}^{N} \mathbb{I}(y_n = k, x_{n,d} = 1)}{\sum_{n=1}^{N} \mathbb{I}(y_n = k)}, \tag{7}$$

where $\mathbb{I}(\cdot)$ is an indicator function, which returns 1, if all logical rules, taken as a function argument, are satisfied (otherwise it returns 0).

It happens that either some words are not present in the training dataset or our dataset is too small to model probability properly. In such cases an additional *a priori* distribution is introduced. It expresses our beliefs about word presence a or absence b in the scope of the classification problem. For binary features (like in text classification problem) an convenient prior is a beta distribution:

$$p(\theta_{d,k}) = \text{Beta}(\theta_{d,k}|a, b), \tag{8}$$

where $a, b > 0$ are called *hyperparameters*. In such case maximum a posteriori estimator (MAP) of $\theta_{d,k}$ has form :

$$\theta_{d,k} = \frac{\sum_{n=1}^{N} \mathbb{I}(y_n = k, x_{n,d} = 1) + a - 1}{\sum_{n=1}^{N} \mathbb{I}(y_n = k) + a + b - 2}. \tag{9}$$

Discriminative approach: κ-NN

Model κ-Nearest Neighbours (κ-NN) is an example of discriminative non-parametric model (non-parametric means that training data plays the role of model parameters). Probability distribution of topic groups conditioned on text document has the following form:

$$p(y|\mathbf{x}, \kappa) = \frac{1}{\kappa} \sum_{i \in N_\kappa(\mathbf{x}, \mathcal{D})} \mathbb{I}(y_i = y) \tag{10}$$

where κ is the number of neighbours, $N_\kappa(\mathbf{x}, \mathcal{D})$ indicates set of κ nearest neighbour indexes in training set $\mathcal{D}$ calculated for document $\mathbf{x}$.

Note that model κ-NN depends on training dataset and value of κ must be assigned before making any prediction.

Another important element of κ-NN model is **distance** that allows to find nearest neighbours. In discussed example each document is represented by D binary features representing appearance of words in a document. To calculate distance between two documents we use **Hamming distance**. For any two feature vectors, Hamming distance returns number positions, where the corresponding vector values do not match. In example, having vectors $\mathbf{x}_1 = (1, 0, 0, 1)$ and $\mathbf{x}_2 = (1, 1, 0, 0)$, Hamming distance between $\mathbf{x}_1$ and $\mathbf{x}_2$ is equal to 2:

$$\begin{array}{cccc} 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ \hline 0 & 1 & 0 & 1 \quad = 2 \end{array}$$

Model selection

In discussed problem, there are three parameters that are not trained from data: number of neighbours κ (for κ-NN) and two parameters of *a priori* distribution (Naive Bayes). We can select these

parameters using N_{val} element validation dataset $\mathcal{D}_{val}$ and following definition of **classification error**:

$$E(\mathcal{D}_{val}; \alpha) = \frac{1}{N_{val}} \sum_{n=1}^{N_{val}} \mathbb{I}(y_n \neq \hat{y}_n), \tag{11}$$

where α is a hyperparameter (κ in case of κ-NN, (a, b) for Naive Bayes), and $\hat{y}_n$ is a class predicted by model for n-th example from validation set.

Algorithm 1: Model selection procedure for κ-NN and Naive Bayes.

Input : Validation dataset $\mathcal{D}_{val}$, set of hyperparameters values Λ

Output: Value α

```
for α ∈ Λ do
    if Naive Bayes then
        Find estimators for π and θ with the use of a and b ;
        Calculate E(D_val; (a, b)) ;
    else if κ-NN then
        Calculate E(D_val; κ) ;
end
Return value α, whose E(D_val; α) is the lowest.
```

Testing the code

You can use function `main` in file `main.py` to validate correctness of your solution.

Content of the file `main.py` can not be modified.

`wourdcloud` package is required to execute the source code. On Windows platform it can be installed as follows:

1. Install Visual C++ 2015 Build Tools from website: http://landinghub.visualstudio.com/visual-cpp-build-tools
2. Open command line (Start → cmd) and execute command: `pip install wordcloud`

Tasks

1. Implement function that calculates Hamming distance (`hamming_distance`). Function takes as arguments two sparse matrices, each representing one set of objects, and returns matrix of Hamming distances between objects from first and second set.

2. Implement function that calculates matrix of sorted labels `sort_train_labels_KNN`. Given distance matrix and class labels, construct matrix that has in each row class labels sorted accordingly to distances stored in corresponding row of distance matrix. [1]

3. Implement function that calculates matrix of class probabilities (10) for κ-NN model (`p_y_x_knn`)

4. Implement function that calculates classification error (11) in file `error_fun.m`. If probability $p(y = k|\mathbf{x})$ for several classes k is maximal, a class with the highest index number k has to be returned as prediction.

5. Implement model selection procedure for κ-NN model, given set of κ values (`model_selection_knn`).

6. Implement function that calculates ML estimator of each class π_k (6) in Naive Bayes model (`estimate_a_priori_nb`).

7. Implement function that calculates MAP estimator for features, $\theta_{d,k}$ (9), and Naive Bayes model (`estimate_p_x_y_nb`).

8. Implement function that calculates matrix of class conditioned probabilities for Naive Bayes model (`estimate_p_y_x_nb`).

9. Implement model selection procedure for Naive Bayes model, given set of a and b parameter values (`model_selection_nb`).

REMARK! All functions names and variables names in the file `content.py` must stay unmodified.

Control questions

1. Derive maximum likelihood estimator for categorical distribution.

2. Derive maximum likelihood estimator for Bernoulli distribution.

3. Derive maximum *a posteriori* estimator for Bernoulli distribution.

4. Why assumption about feature independence is made? What are advantages and disadvantages of this approach?

5. How to interpret θ parameters? How many parameters a model has for D features and K classes?

6. How to interpret π parameters? How many parameters a model has for D features and K classes?

[1] EXAMPLE: distance matrix: `[2 5 3; 6 7 1]`, class labels: `[1 4 3]`, matrix of sorted labels: `[1 3 4; 3 1 4]`.

7. How to interpret κ hyperparameter? What is geometrical interpretation of this parameter? How its value affects classification result?

8. How neighbourhood is defined in κ-NN?

9. Is κ-NN generative or discriminant? Is that parametric or non-parametric model?

10. Is Naive Bayes generative or discriminant? Is that parametric or non-parametric model?