Mini-Project: Categorical Naive Bayes

In this mini-project, we will work on the Extended MNIST dataset. You will use scikit-learn https://scikit-learn.org/stable/index.html.

1 Visualizing the EMNIST Dataset

First you will visualize the EMNIST dataset. See more details on the dataset here: https://www.nist.gov/itl/products-and-services/emnist-dataset. To download the data, use the code snippet 'emnist_project.py' in the folder. You will be using a balanced version of the dataset for this exercise, so set the split parameter to balanced. This brings the number of classes to C=47. However, this does not mean that the training data will always be balanced, and therefore you should not omit modeling the class distribution $p(C_k)$ in your naive bayes model. For hints on how to visualize the data, you can use the example code here: https://github.com/probml/pyprobml/blob/master/deprecated/scripts/emnist_viz_pytorch.py as a starting point (or the file 'emnist_viz_pytorch.py' in the folder).

Task 1: Display samples from this dataset in a 5 x C table: columns correspond to character IDs, rows to five random samples from the same category.

2 Building a Model

Implement a conditional Categorical model (Categorical Naive Bayes) for EMNIST, where you assume that all pixels are independent binary RVs. For this, you will use scikit-learn's Estimator object, https://scikit-learn.org/stable/developers/develop.html.

- For priors of the class distribution, $p(\Theta|\alpha)$, you will assume Dirichlet priors, https://en.wikipedia.org/wiki/Dirichlet_distribution, with symmetric hyperparameters, i.e., $\alpha_1 = \alpha_2 = \cdots = \alpha_C = \alpha$, where C denotes the number of the classes.
- For priors of each pixel, $p(\Theta|\beta)$, you will assume Beta priors, https://en.wikipedia.org/wiki/Beta_distribution, with the same hyperparameters, i.e., $\beta_i = \beta_j = \beta$, where i, j represents any two positions of the pixels, and $p(x_i|\beta) \propto x_i^{\beta_i - 1}(1 - x_i)^{\beta_i - 1}$.

Task 2.1: Implement two learning paradigms, MLE and MAP for model parameters, i.e., $p(\Theta|\mathcal{D}, \alpha, \beta)$, as Estimator Fit method.

Task 2.2: Then implement predictive models based on each type of learning, i.e., $p(y|x, \mathcal{D}, \alpha, \beta) = \int p(y|x, \Theta)p(\Theta|\mathcal{D}, \alpha, \beta)d\Theta$, as Estimator Predict method. (Hint: You may want to use Bayes' Rule for $p(y|x, \Theta)$ here.)

Task 2.3: Also implement the scoring methods for the two models Θ_{MLE} and Θ_{MAP},

$$\text{Score} = \frac{1}{|\mathcal{D}|}\log p(\mathcal{D}|\Theta_{MLE}) \quad \text{and} \quad \text{Score} = \frac{1}{|\mathcal{D}|}\log p(\mathcal{D}|\Theta_{MAP})$$

as Estimator Score/Model method, where $|\mathcal{D}|$ denotes the dataset size.

3 Plotting the Learning Curves for (almost) Balanced Training Data

Follow the example in this scikit-learn documentation page to plot Learning Curves for the two models. For that purpose, your training dataset will consist of 10% (11280) of EMNIST data points, as defined in 'emnist_project.py' (see the folder). For learning curves, use five points, equally spread between 10% and 100% of the training dataset (thus, from 1128 to 11280 data points).

Task 3.1: Fix the pixel prior $\beta = 1$ and plot the training&validation learning curves for $\alpha = (1, 10, 50, 100, 200)$.

Task 3.2: Fix the class prior $\alpha = 1$ and plot the training&validation learning curves for $\beta = (1, 2, 10, 100)$.

4 Plotting the Learning Curves for Imbalanced Training Data

Although we tend to assume each class has the same degree of importance (that is, in the test set, each class will have same amount of examples), in reality, we sometimes cannot construct such a balanced training set. For example, if we want to do image classification for different species of animals, the examples of cats and dogs can be easily sampled, while the examples of sharks and whales can only be sampled in a lower frequency, which will lead to an imbalanced training dataset. To simulate this imbalanced class distribution, we can use Dirichlet distribution for our training dataset. When constructing the imbalanced training dataset, we assume the ratio of each class is sampled from a Dirichlet distribution with $\alpha_{\text{class}_1} = \cdots = \alpha_{\text{class}_C} = \alpha_{\text{class}}$. We first sample the class ratio from this distribution, and then sample examples of each class according to this ratio. By adjusting the value of α_{class}, we can construct the training data of different degrees of imbalance. Typically, larger α_{class} will produce more balanced class distribution. Now your task is to compare the performance of the MLE and MAP models in training settings with different degrees of imbalance.

Task 4.1: For MAP model, fix the class prior $\alpha = 1$ and try $\beta = (1, 1.2, 2, 10, 100)$ for $\alpha_{\text{class}} = (0.1, 0.2, 0.5, 1, 10, 100)$. Plot the training&validation learning curves and compare its performance with the MLE model.

Task 4.2: For MAP model, fix the pixel prior $\beta = 1$ and try $\alpha = (1, 10, 100, 1000)$ for $\alpha_{\text{class}} = (0.1, 0.2, 0.5, 1, 10, 100)$. Plot the training&validation learning curves and compare its performance with the MLE model.

5 Discussion

Discuss performance of different models. Specifically:

Task 5.1: How does the generalization performance of each model change as a function of the amount of training data and the choice of hyper-parameters?

Task 5.2: Which model would you select for deployment and why?

6 Hints

See http://d2l.ai/chapter_appendix-mathematics-for-deep-learning/naive-bayes.html for ideas about how to implement the models.