Systems analysis and decision support methods in Computer Science

The final assignment: classification of clothes images

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

Assignment goal

The goal of this assignment is to propose a model for classification of small images, which present clothes (examples are presented in Figure 1). There are no restrictions regarding a classification model, a training algorithm or a feature extraction method. The final solution will be evaluated exclusively based on the test set classification accuracy.

While working on the assignment, the following steps should be considered:

- Feature extraction (from images).
- Implementation of a classification model.
- Implementation of a training algorithm.
- Prediction (for unknown image) using trained model.

Figure 1: Exemplary images selected randomly from the dataset.

Caution! Not all steps mentioned above are obligatory. For instance, κ-NN classifier does not require the training procedure and can classify images without prior feature extraction (using raw training images for prediction).

In the next part of the instruction, the **exemplary** solutions are presented. Please bear in mind that you are not limited to methods described in this document!

Feature extraction

Frequently, to achieve satisfactory quality of the classification, a feature extraction method is required. Feature extraction is about transforming the original object into a set of numerical features, which carry important information about this object. There are many feature extraction methods described in the literature. To select proper one, a problem domain must be considered (i.e. image analysis, audio processing, medical data analysis). In the next part of this section exemplary feature extraction methods are presented.

Gabor Filters. The idea behind Gabor Filters is to propose a function that is composed of Gaussian envelop and sinusoid carrier. This function, when convolved with image, gives response that is used as an input to the classifier. For detailed information about Gabor filters you can read description at `http://personal.lut.fi/users/joni.kamarainen/downloads/publications/ipta2012.pdf`

Haar Filters. The idea behind Haar filters is to define images (often rectangular), which contain bright and dark regions. In subsequent step, sum of pixels is calculated for respective fragments of an image. The Haar Filters are popular due to their simple form and possibility to calculate them efficiently with *integral images*. For detailed information about Haar filters you can read description at `http://en.wikipedia.org/wiki/Haar-like_features`, and in report: `http://www.merl.com/reports/docs/TR2004-043.pdf`

Kernel functions. In this approach an arbitrary subset of M objects from a dataset is selected (either randomly or in a deterministic way), and then function that describes similarity between new and known objects $k(\mathbf{x}^{new}, \mathbf{x}_n)$ is defined. Such function, if some conditions are met, is called *kernel function* or just *kernel*. In such case values of kernel function calculated for each individual observation give a rise to feature vector of size M. A similar approach is used in *radial-based neural networks*: `http://en.wikipedia.org/wiki/Radial_basis_function_network`

Principal Component Analysis. The Principal Component Analysis (PCA) is a dimensionality reduction technique. It allows to find linear transformation from high-dimensional space (original object) into low-dimensional one (feature vector). For detailed information about PCA you can read description at `http://cmp.felk.cvut.cz/~hlavac/TeachPresEn/11ImageProc/15PCA.pdf`

Manual feature extraction. Sometimes domain-specific knowledge allows for manual design of high-quality features. In example, for handwritten digits recognition (the final assignment 2018) the following features can be proposed:

- Proportion between height of the specific fragment of the symbol to the whole symbol height.
- Proportion between width of the specific fragment of the symbol to the whole symbol width.
- Density of pixels in particular region of an image.

It is troublesome to propose well-defined workflow for manual feature design, because this approach is specific to classification problem and, to vast extent, depends on creativity of feature designer.

It is worth mentioning that different feature extraction techniques can be combined. For example the vector calculated using Haar filters can be transformed using PCA.

Model

Generative models. The goal of generative modeling is to estimate joint probability distribution of random variables $\boldsymbol{\phi}$ (features) and y (class) using $p(\boldsymbol{\phi}, y|\boldsymbol{\theta})$, where $\boldsymbol{\theta} \in \mathbb{R}^M$ is a vector of model parameters. Notice that generative models allows for factorization of joint distribution[1]:

$$p(\boldsymbol{\phi}, y) = p(\boldsymbol{\phi}|y)p(y) \tag{1}$$

$$= p(y|\boldsymbol{\phi})p(\boldsymbol{\phi}). \tag{2}$$

Having estimate of joint distribution of features and classes, we can pick particular value of the class and generate corresponding feature vector (equation (1)), or (in the opposite way) pick feature vector at first and then generate value of the class (equation (2)). This is a reason why these models are called *generative*.

Examples of generative models:

- *Gaussian Discriminant Analysis* (GDA): if features in individual classes are modeled using normal distribution $p(\boldsymbol{\phi}|y)$, and distribution $p(y)$ is modeled by multinomial distribution;
- *Naive Bayes*: similarly to GDA, with assumption that features are independent (diagonal covariance matrices of normal distributions).

More examples of generative models you can find at `http://www.cs.ubc.ca/~murphyk/MLbook/`.

[1] for simplicity conditioning on $\boldsymbol{\theta}$ was omitted.

Discriminative models. The goal of discriminative modeling is to estimate conditional probability distribution $p(y|\boldsymbol{\phi}, \boldsymbol{\theta})$ directly, where $\boldsymbol{\theta} \in \mathbb{R}^M$ is a vector of model parameters. In some application domains we want to investigate the class dependence on a feature vector, whereas any relations between features are of no importance. In such cases, the distribution $p(y|\boldsymbol{\phi})$ is sufficient for decision making. A model that directly estimates the distribution y conditioned on $\boldsymbol{\phi}$ is called *discriminative model*.
Examples of discriminative models:

- *Logistic Regression*: it uses linear combination of features with sigmoid function for probability estimation (binary classification), or softmax function (multiclass classification);
- κ-*Nearest Neighbors*: given kernel function, a probability distribution is estimated in non-parametric way using class values of $\kappa \in \mathbb{N}$ nearest objects;
- *Neural network, Multilayer Perceptron* (MLP): if sigmoid function (or softmax) is used at the output of MLP, then such model can be considered as a probabilistic discriminative model.

Additional information about discriminative modeling you can find at `http://www.cs.ubc.ca/~murphyk/MLbook/`.

Functional models. Another approach for classification is to propose *discriminant function*, which has no probabilistic interpretation. Such function maps directly feature vectors to class values.
Examples of functional models are:

- *Naural Network, Multilayer Perceptron* (MLP): if sigmoid function (or softmax) are not used at the output of MLP, then such model can be considered as the discriminative function;
- *Support Vector Machines* (SVM): this classification model finds maximum margin that separates two classes. For multiclass classification K models have to be trained and prediction 1 vs. ALL can be used.

Detailed description of discriminative functions is provided in book `http://www.cs.ubc.ca/~murphyk/MLbook/`.

Training

Training is a procedure which allows to estimate model parameters $\boldsymbol{\theta}$ using training dataset $\mathcal{D} = \{(\boldsymbol{\phi}_n, y_n)\}_{n=1}^N$ by minimizing learning objective function. For probabilistic models we usually use *negative log-likelihood* or, given *a priori* distribution on parameters, *negative logarithm a*

posteriori.[2] For non-probabilistic models (i.e. discriminative functions) another type of learning objective have to be proposed, e.g., mean squared error or *cross-entropy loss* for binary classification.

Additionally, to ensure that trained model have desired properties (such as ability of generalization), a *regularization* can be introduced. Regularization improves training procedure by suppressing the effect of *overfitting*, or obtaining *sparse representation*. Exemplary regularization is norm ℓ_2 calculated for model parameters.

Sometimes parameter values that minimize objective function can be calculated analytically and presented in *closed form*. Nevertheless, for majority of cases it is impossible. Therefore, many approaches for numerical optimization have been developed. The most popular ones are *gradient descent* method or its modification – *stochastic gradient descent*. The latter one updates weights in each iteration based on several observations (*mini-batch*). To follow this numerical procedure a gradient (or at least its approximation) have to be calculated.

It is worth mentioning that for some models dedicated training procedures were developed (i.e. *Sequential Minimal Optimization* (SMO) for SVM).

Detailed description of various model training techniques can be found in book: `http://www.cs.ubc.ca/~murphyk/MLbook/`.

Prediction

Generative models. Having trained generative model, for each new object $\boldsymbol{\phi}^{new}$, an conditional probability $p(y|\boldsymbol{\phi}^{new})$ can be calculated using Bayesian theorem[3]:

$$\begin{aligned} p(y|\boldsymbol{\phi}^{new}) &= \frac{p(\boldsymbol{\phi}^{new}|y)p(y)}{p(\boldsymbol{\phi}^{new})} \\ &= \frac{p(\boldsymbol{\phi}^{new}|y)p(y)}{\sum_{y'} p(\boldsymbol{\phi}^{new}|y')p(y')} \end{aligned}$$

Particular class is assigned to the object based on value of above-mentioned probability. In other words, predicted class value maximizes the following probability:

$$y^* = \arg\max_y p(y|\boldsymbol{\phi}^{new}). \tag{3}$$

Discriminative models. Prediction with the use of trained discriminative models follow the same procedure as prediction in generative models (formula 3).

[2]Usually this method is called *maximum a posteriori* (MAP). Nevertheless, to preserve consistency with the rest of the tutorial (**minimization** of learning objective function), a minus sign must be introduced.

[3]For simplicity conditioning on $\boldsymbol{\theta}$ is omitted.

Functional models. Discriminant function assigns class value to particular object. Therefore the process of prediction is about calculating value of discriminant function for new object.

Tasks

1. Download `zip` file with training dataset `train.pkl` from course website. File `train.pkl` contains images, stored in the form of a single matrix ($N \times D$, $N = 30134$), that can be used for training purpose. Each row in the matrix is D-dimensional vector and represent one 56×56 image. To visualize an image, select a row from matrix `train.pkl` and use command `np.reshape` and `plt.imshow`.

   Also file `train.pkt` stores N labels that define classes of images. Labels have values $\{0, \ldots, 9\}$ (value of the label corresponds to type of clothes presented in the image).

2. Use data described above to train a model for image classification. If validation dataset is needed, it has to be established by splitting original dataset (`train.pkl`).

3. Implement model prediction in Python function `predict` (file `predict.py`) so that for input matrix $\mathbf{X}$ (each row is one image for classification) the returned value is vector $\mathbf{y}$ of predicted class labels (i-th row $\mathbf{X}$ corresponds i-th element of the vector $\mathbf{y}$).

4. Quality of prediction can be verified once a day, by uploading `zip` archive (containing `predict.py` and any other files required by solution to operate) on server:

   `http://mlg.ii.pwr.edu.pl/msid/`

   The quality of submission is going to be verified using 25% of testing dataset stored on the server.

5. Size of submitted `zip` archive must not exceed 5 MB. Execution wall-time (for `predict.py`) is set to 1 minute for 2500 images (25% of test set). Additionally, file `predict.py` must be stored in main directory of the `zip` archive (not in subfolder).

6. The server evaluates submission on a daily basis at 00:00 am with the use of Python 3.6 (with numpy package installed). Please verify, if your submission is compatible with mentioned environment.

7. The final submission must be uploaded before hard deadline, which is specified on the course website. **After the deadline it is impossible to submit any assignments. Once the server is closed by administrators, the course instructor can not restart the server or accept any solution.**

8. To pass the course an accuracy of prediction must exceed 50% (evaluated on whole dataset). Notice that evaluation on 25% test set (daily basis) might lead to slightly different results ($\pm 1\%$) than evaluation on 100% test test (after deadline).