Systems analysis and decision support methods in Computer Science

Lab – Python – Assignment 3

Face detection

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

translation: P. Klukowski

Assignment goal

The goal of this assignment is to implement a logistic regression model and training algorithms in the context of face detection problem.

Face detection as classification problem

The problem of face detection in photography I is about automated selection of image area, where a face is present. Using the *shifting window* technique, the image is divided into small rectangles, which undergo binary classification independently from each other. Each rectangle is described using feature vector $\boldsymbol{x} = (1, \phi^1(I) \dots \phi^{D-1}(I))^{\mathrm{T}}$ obtained by applying a descriptor called Histogram of Oriented Gradients[1] (HOG). For each image fragment we need to solve a binary classification problem (with classes $y \in \{0, 1\}$), where we label a fragment with $y = 1$ if it contains a face or $y = 0$ if it does not contain a face.

We can define a probability of face presence in the image fragment (represented by the features $\boldsymbol{x}$) using logistic sigmoid function:

$$p(y = 1|\boldsymbol{x}, \mathbf{w}) = \sigma(\mathbf{w}^{\mathrm{T}}\boldsymbol{x}), \tag{1}$$

where $\mathbf{w}$ denotes D-dimensional vector of parameters. Logistic sigmoid function is defined as follows:

$$\sigma(a) = \frac{1}{1 + \exp(-a)}. \tag{2}$$

We classify the image fragment by checking if the probability of face presence is higher than some fixed threshold $\theta \in [0, 1]$:

$$y^* = \begin{cases} 1, & \text{if } p(y = 1|\boldsymbol{x}, \mathbf{w}) \geq \theta, \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

The classifier in the form given above is called **logistic regression**.

[1] The details how this descriptor is calculated are not important for this assignment and were omitted. However, if you are interested in the details see `http://en.wikipedia.org/wiki/Histogram_of_oriented_gradients`.

Learning as an optimization problem

The key aspect in building face detection model is calculation of its $\mathbf{w}$ parameters values. This procedure is called learning. Assume that we have a training set $\mathcal{D} = \{\boldsymbol{x}_n, y_n\}_{n=1}^N$, where $\boldsymbol{x}_n$ is an image fragment represented by HOG features, and y_n is a label denoting whether the fragment contains a face or not.

To train the model we can use the **maximum likelihood** method. Having the value of $\boldsymbol{x}$ defined, a random variable y has binomial distribution:

$$p(y|\mathbf{x}, \mathbf{w}) = \sigma(\mathbf{w}^{\mathrm{T}}\mathbf{x})^y(1 - \sigma(\mathbf{w}^{\mathrm{T}}\mathbf{x}))^{1-y}. \tag{4}$$

We introduce the following notation:

$$\sigma_n \equiv \sigma(\mathbf{w}^{\mathrm{T}}\boldsymbol{x}_n). \tag{5}$$

For equation (4) and notation (5), the likelihood function has form:

$$p(\mathcal{D}|\mathbf{w}) = \prod_{n=1}^{N} \sigma_n^{y_n}(1 - \sigma_n)^{1-y_n}. \tag{6}$$

Taking the negative logarithm we obtain:

$$-\ln p(\mathcal{D}|\mathbf{w}) = -\sum_{n=1}^{N} \big(y_n \ln \sigma_n + (1 - y_n) \ln(1 - \sigma_n)\big). \tag{7}$$

An objective function is defined using negative logarithm of likelihood function: [2]

$$L(\mathbf{w}) = -\frac{1}{N} \ln p(\mathcal{D}|\mathbf{w}). \tag{8}$$

To find the parameters that minimize the negative log-likelihood function, we have to solve the following unconstrained minimization problem:

$$\boxed{\text{minimize (wrt } \mathbf{w}) \quad -\ln p(\mathcal{D}|\mathbf{w})}$$

Since values of σ_n depend on the parameters $\mathbf{w}$, obtaining an analytical solution is impossible. Therefore we need to use numerical optimization method, such as **gradient descent algorithm** or **stochastic gradient descent**.

[2] Multiplying function by constant doesn't change solution. Scaling by factor $\frac{1}{N}$ allows to make optimization procedure independent on number of examples in training set, what simplifies comparison of different methods.

Gradient descent algorithm

The gradient descent method is presented below. Note that to apply gradient descent method, the gradient of the objective function $\nabla_{\mathbf{w}} F(\mathbf{w})$ must be calculated.

Algorithm 1: Gradient descent method

Input : Start point $\mathbf{x}_0 \in \mathbf{dom} F$, precision ε, step η

Output: Optimal point $\mathbf{x}$ for the function F

$\mathbf{x} \longleftarrow \mathbf{x}_0$;
while $\|\mathbf{x} - \mathbf{x}_{old}\|_2 \geq \varepsilon$ **do**
  $\mathbf{x}_{old} \longleftarrow \mathbf{x}$;
  $\Delta\mathbf{x} \longleftarrow -\nabla F(\mathbf{x})$;
  $\mathbf{x} \longleftarrow \mathbf{x} + \eta\Delta\mathbf{x}$;
end

Stochastic gradient descent algorithm

Gradient descent algorithm involves calculation of the gradient for an entire dataset, what is inconvenient for large datasets. Moreover calculation of such gradient leads to many optimization steps, before the algorithm converges to a final solution. To cope with two problems mentioned above, the stochastic gradient descent algorithm was proposed. It operates on subsets of dataset.

Notice that objective function can be formulated as sum of observations:

$$-\ln p(\mathcal{D}|\mathbf{w}) = -\frac{1}{N}\sum_{n=1}^{N} \ln p(y_n|\mathbf{x}_n, \mathbf{w}).$$

Entire dataset can be divided into M mini-batches of data $\mathcal{D} = \{\mathcal{X}_m, \mathcal{Y}_m\}_{m=1}^{M}$, each having N_b observations. In such case:

$$\begin{aligned} L(\mathbf{w}) &= -\frac{1}{N}\ln p(\mathcal{D}|\mathbf{w}) \\ &= -\frac{1}{N}\sum_{n=1}^{N} \ln p(y_n|\mathbf{x}_n, \mathbf{w}) \\ &= -\frac{1}{MN_b}\sum_{m=1}^{M}\sum_{n_b=1}^{N_b} \ln p(y_{n_b}|\mathbf{x}_{n_b}, \mathbf{w}) \\ &= -\frac{1}{M}\sum_{m=1}^{M}\frac{1}{N_b} \ln p(\mathcal{Y}_m|\mathcal{X}_m, \mathbf{w}) \\ &= \frac{1}{M}\sum_{m=1}^{M} L(\mathbf{w}; \mathcal{X}_m, \mathcal{Y}_m), \end{aligned}$$

in the last step value of an objective function for single mini-batch was defined:

$$L(\mathbf{w}; \mathcal{X}_m, \mathcal{Y}_m) = -\frac{1}{N_b} \sum_{n_b=1}^{N_b} \ln p(y_{n_b} | \mathbf{x}_{n_b}, \mathbf{w}). \tag{9}$$

Notice that for substitution $N_b = N$ we obtain objective function from (8). Another corner case is $N_b = 1$, but in practice higher value is assigned to N_b (i.e. $N_b = 50$). For $N_b > 1$ stochastic gradient descent algorithm converges faster to final solution than gradient descent algorithm.

Stochastic gradient descent algorithm is presented below. To use it, a gradient $\nabla_{\mathbf{w}} L(\mathbf{w}; \mathcal{X}_m, \mathcal{Y}_m)$ must be calculated.

Algorithm 2: Stochastic gradient descent method

Input : Function L, starting point $\mathbf{w}_0 \in \mathbf{dom} L$, mini-batch size N_b, number of epochs K, learning rate η

Output: Optimal point $\mathbf{w}$ for function L

```
w ⟵ w_0;
Divide dataset D for mini-batches {X_m, Y_m}_{m=1}^M;
for k = 1, ..., K do
    for m = 1, ..., M do
        Δw ⟵ −∇_w L(w; X_m, Y_m);
        w ⟵ w + ηΔw;
    end
end
```

When implementing the method, consider an order of mini-batches. Having 400 examples dataset and mini-batch size 100, the first mini-batch should include first 100 examples, the second mini-batch examples 101 till 200 etc.

Regularization

Often, in training procedure, an objective function is regularized. For face recognition problem, an ℓ_2 regularization on parameters is going to be used (note that regularization is not applied to bias x_0, which is always equal to 1).

$$L_\lambda(\mathbf{w}) = L(\mathbf{w}) + \frac{\lambda}{2} \|\mathbf{w}_{-0}\|_2^2, \tag{10}$$

where $\mathbf{w}_{-0} = (w_1, \ldots, w_{D-1})^{\mathrm{T}}$ is weight vector without bias w_0, $\lambda > 0$ indicated regularization parameter.

Model selection

In analysed problem, there are two parameters that are not estimated directly from data: classification threshold θ and regularization parameter λ. In case validation set $\mathcal{D}_{val}$ is available, an model selection procedure can be used to estimate mentioned two parameters. To measure performance of the model that has weights $\mathbf{w}$ on validation set, the **F-measure** is used:

$$F(\mathcal{D}_{val}; \theta, \lambda, \mathbf{w}) = \frac{2TP}{2TP + FP + FN}, \tag{11}$$

where TP (*true positives*) indicates number of examples from $\mathcal{D}_{val}$ that have label 1 and trigger model response equal to 1, FP (*false positives*) is number of examples that have label 0 and trigger response 1, FN (*false negatives*) is number of examples labelled as 1 triggering response 0.

Selection of F-measure in analysed problems allows to evaluate model performance on unbalanced datasets (datasets, where number of examples belonging to each class varies significantly). Note that in face recognition is an unbalanced problem, since major part of an image doesn't contain any face.

Model selection procedure is the following: [3]

Algorithm 3: Model selection procedure

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

Output: Optimal values θ i λ

```
for λ ∈ Λ do
    Find solution w for objective function L_λ(w) ;
    for θ ∈ Θ do
        Calculate value F(D_val; θ, λ, w) ;
    end
end
Return values λ i θ, where F-measure is the highest.
```

Use stochastic gradient descent to find parameters $\mathbf{w}$. Pass objective function with regularization ℓ_2 to algorithm throught function pointer.

Testing the code

To check the code correctness use the function `main` (file `main.py`).

In the file `main.m` you must not modify anything and add any additional code.

To execute program correctly a package `pillow` must be installed. On Windows platform it can be done as follows:

[3] Note that threshold selection do not require multiple model training.

- Start command line (Start → cmd)
- Execute command `pip install pillow`

Tasks

1. Implement sigmoid logistic function (2)
2. Calculate the gradient of the objective (8), and then implement objective value and objective gradient in `logistic_cost_function`.
3. Implement the function calculating the values of $\mathbf{w}$ using gradient descent method in `gradient_descent`. This function must also return a list of the objective function values at each iteration of the optimization algorithm.
4. Implement the function calculating the values of $\mathbf{w}$ using stochastic gradient descent method in `stochastic_gradient_descent`. This function must also return a list of the objective function values, calculated for entire dataset, at each iteration of the optimization algorithm.
5. Calculate objective function gradient (10) and implement function

   `regularized_logistic_cost_function`, which returns objective function and its gradient value.
6. Implement function that makes prediction (3) based on trained model (`prediction`).
7. Implement F-measure (11) calculation in `f_measure`.
8. Implement model selection procedure in `model_selection`. The function returns the best model parameters and matrix with F-measure values calculated for each pair of hyperparameters (λ, θ).

The final result of the running program should be the same as presented in Figure 1.

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

Control questions

1. Calculate the derivative of the sigmoid logistic function. Write down the derivative using the sigmoid logistic function $\sigma(a)$ only.
2. Calculate the gradient of the negative log-likelihood (8) or (10).

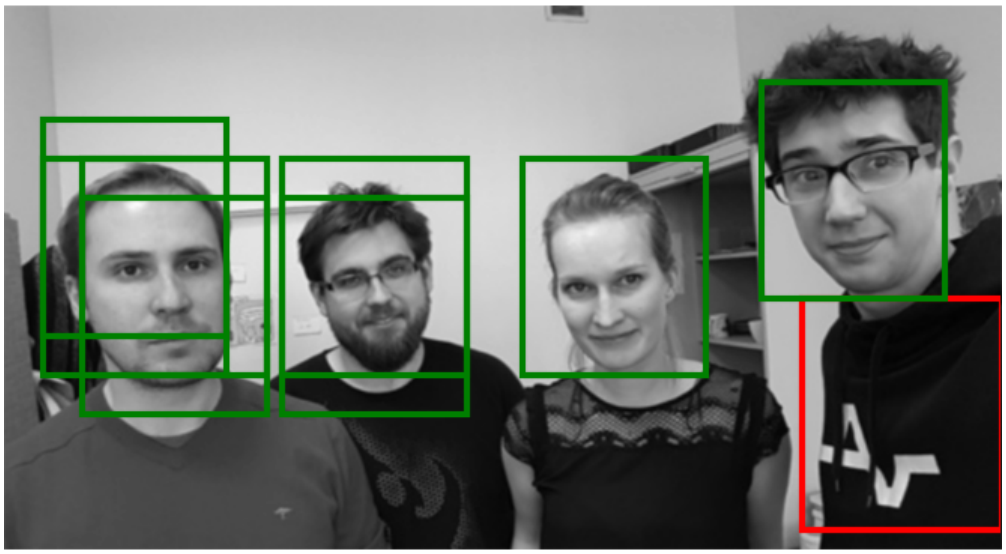

Figure 1: The final effect of the correctly implemented program.

3. What is logistic regression model? How it models conditional probability?

4. What is the meaning of the threshold θ? How to calibrate its value?

5. What is F-measure? For what purposes it is used in assignment? Why accuracy is not used for measuring model performance of validation set?

6. What is the meaning of the parameter η in the gradient descent algorithm? How algorithm behaves for different values for this parameter?

7. How object detection problem is solved in this assignment? Why this is a classification problem?

8. Why stochastic gradient descent converges faster that gradient descent? What is the impact of mini-batch on convergence? How algorithm behaves for small and large mini-batches?

9. How to introduce regularization ℓ_2 on parameters of logistic regression? What is the result of this procedure? In what cases we have to use regularization, in what cases it is not necessary?

10. How model selection is made? What hyperparameters are calibrated? What hyperparameters require multiple model training for calibration? What hyperparameters don't require that? Why?