10907 Pattern Recognition

Lecturers
Prof. Dr. Thomas Vetter ⟨thomas.vetter@unibas.ch⟩

Assistants
Dr. Adam Kortylewski ⟨adam.kortylewski@unibas.ch⟩
Dennis Madsen ⟨dennis.madsen@unibas.ch⟩
Dana Rahbani ⟨dana.rahbani@unibas.ch⟩

Exercise 4 — Logistic Regression and Naïve Bayes

Introduction	05.11
Deadline	**13.11** Upload code to Courses.
	12.11+13.11 Group presentations, U1.001, see schedule online

This exercise is divided into two independent parts:

1. Implement and test a logistic regression classifier with and without regularization. The classifiers will first be applied to a toy problem and then to images from the MNIST dataset of handwritten digits.
2. Implement and test a Naive Bayes classifier. The classifier will be applied to an email dataset for spam detection.

You can download the data needed for this exercise and a code skeleton from the following repository: `https://bitbucket.org/gravis-unibas/pattern-recognition-2018`.
A number of helper functions are provided - your task is to implement the *TODO* parts in the python (.py) files.

Remember to upload your code to courses in a ZIP file. Do NOT include the data folder. Only the python files you edited should be included + a file containing the group members names. Only 1 person from the team should upload!

Data:

Each `mat`-file contains the training and the test set for a problem, named `NAME_(train|test)`. The sets are encoded as $(d+1) \times N$-matrix, where N is the total number of data points and d the dimension of the feature vector. The first column contains the label $y \in \{-1, 1\}$.

- `toy` ($d = 2$): A very small (x,y) toy data set that can be easily visualized. Use it for development and for studying the behaviour of your classifiers.
- `zip13` ($d = 256$): Handwritten digits automatically scanned by the U.S. Postal Service. The set is reduced to the digits 1 and 3. The digits are normalised to a grey scale 16×16 grid.
- `zip38` ($d = 256$): As above with digits 3 and 8.

Remarks:

- Do not use the test sets for training!
- Be aware of the computational demands. Some implementations may take a while to train with lots of data. During development use only a few data points until your implementation works, then train with more data.

1 Logistic Regression

1.1 Classification without regularization

Implement and test a Logistic Regression classifier without regularization. Start with the provided script `logreg` for the LOGREG class. Use a $regcoeff$ of 0 and set every `regularizationTerm` to zero for now. Refer to the slides for all the needed equations. Prepare your class by completing the functions listed here:

- Define the activation function in the `activationFunction` function. Use the equation of the logistic function.
- Define the cost function in the `costFunction` function. Remember, you are maximizing the loglikelihood of the posterior for class 1. Pay attention to what you give as input to the logarithm function! You will get a runtime error if the input value is 0.
- Calculate the gradient of the cost function in the `calculateDerivative` function. You may directly use the derived form of the derivative from the lecture slides.
- Calculate the Hessian matrix in the `calculateHessian` function. Again, you may directly use the equations derived in the lecture slides.
- Optimize using the Newton-Raphson algorithm in the `optimizeNewtonRaphson` function. To implement this, you should update the model parameters iteratively. Refer to the equation provided in the lecture slides for iterative optimization. Include a threshold on the proposed update, if the update is below the *self.__threshold* it is safe to assume convergence.
- Train using the `train` function.
- Implement the `classify` function which uses the *self.theta* model parameters to classify data points
- Implement the `printClassification` function to compute the classification error of the classifier as well as the number of incorrectly classified items

Once your `logreg` class is ready, you can start training and testing on the three provided datasets. Make sure you train your classifier on the training set and predict on the test set!

- Start with the toy dataset and the script `ex4-LOGREG-1-Toy.py`. Train the logistic regression classifier and calculate the accuracy of its predictions on the training set. Then calculate the accuracy of its predictions on the testing set. Visualize the results in 2D and 3D using the provided plotting functions `plot2D` and `plot3D`. Make sure you understand and can explain the resulting figures (legend, colors, marker shape, straight lines).
- Now move on to the MNIST dataset. Use the script `ex4-LOGREG-2-MNIST.py`. Train and test a classifier for each of the two MNIST datasets. Calculate the accuracy of each and show the number of misclassified input vectors.

1.2 Classification with regularization

Now you will add regularization to the logistic regression classifier class. Recall that the goal of regularization is to penalize large parameter values. There are 2 new terms to define in this part: *regularizationTerm* and the regularization coefficient r. The regularization coefficient r represents $(1/2\sigma^2)$ in the lecture slides. The *regularizationTerm* is what should be added to the cost function to perform L2 regularization on the weight vector $\underline{w}$. Start from your implementation of part 1.1 and modify the functions `costFunction`, `calculateDerivative`, `calculateHessian`:

- Implement the equation of the *regularizationTerm* in the cost function, its first derivative in the `calculateDerivative` function, and its second derivative in the `calculateHessian` function. Make sure you do not regularizing the bias term of the parameters (w_0) in all three functions! The equations of the term in the cost function and its first derivative can be found in the lecture slides. As for the Hessian derivation, you must perform the derivation yourself, then construct a regularization matrix to add to your previously defined Hessian matrix. Tip: The regularization matrix is a diagonal matrix with the same shape as the Hessian. The first entry of the matrix must be explicitly set to zero, as it represents the w_0 term which should not be regularized.

For each of the three datasets, train your classifier on the training data and predict on the test set. Use different values for the regularization coefficient: 0 (no regularization), 0.1 and 0.5. Note down the accuracy and number of misclassifications of each. How does regularization affect the accuracy on test data?
Plot the results on the toy dataset in 2D and 3D using the provided plotting functions `plot2D` and `plot3D`. Compare the figures of the three different regularization conditions. How does the value of the regularization coefficient affect the resulting hyperplane? Compare the posterior value prediction of the same datapoint from each of the three regularization conditions. What is the relationship between regularization and the confidence of predictions, and between regularization and the error rate on the test data? In which cases would you prefer to increase regularization?

2 Naive Bayes

Data:

For the email classification, the test and training emails can be found under `data/emails/train` and `data/emails/test`.

- `emails` - each training example can be found in separate `.txt` files. The filenames starting with the letter 's' are spam emails.

2.1 Linear Classification

Show that the Naïve Bayes model's classification using the spam score is linear if we use a fixed vocabulary and word counts x_i as our features. You have to show the correspondence of the classification function with the standard linear classifier $g(\boldsymbol{x}) = w_0 + \langle \boldsymbol{w}, \boldsymbol{x} \rangle$.

2.2 Implementation

(a) **Learning** Learn the likelihood terms and prior probabilities for the Bayes model based on the provided training data.

- What are the 10 most found words in spam and non-spam emails?
- What are the 10 most indicative words for spam / non-spam emails?

(b) **Classification Accuracy**

Implement the functions `classify` and `classifyAllInFolder` that classifies emails using the model estimated above. Test your Naïve Bayes Classifier on the provided test dataset.

Plot the classification accuracy using the top X features ($X = [1, 2, 5, 10, 20, 30, 40, 50]$).

What words are good indicators for spam?

2.3 Independence

The Naïve Bayes classifier assumes independence among the words for a given class. Find an example where this is not a proper assumption. Look for a combination of two words in the non-spam set whose frequencies do not satisfy the independence assumptions

$$P(x_1|h)P(x_2|h) = P(x_1, x_2|h).$$

Alternatively, you can also present a non-formal argument.