10907 Pattern Recognition

Lecturers
Prof. Dr. Thomas Vetter ⟨thomas.vetter@unibas.ch⟩
Prof. Dr. Ivan Dokmanić ⟨ivan.dokmanic@unibas.ch⟩
Dr. Dennis Madsen ⟨dennis.madsen@unibas.ch⟩
Dana Rahbani ⟨dana.rahbani@unibas.ch⟩

Tutors
Martin Ramm ⟨martin.ramm@unibas.ch⟩
Viktor Gsteiger ⟨v.gsteiger@unibas.ch⟩

Exercise 3 — Logistic Regression and Naïve Bayes

Deadline **07.11** Upload .ZIP on Adam.

The exercises are done in groups of 2 students. Only upload 1 version of the exercise on Adam and specify your group partner.

Upload in .zip format containing 1 .pdf file with your answers to the questions and the code folder files (do NOT include the data folder). Create the .zip file with the `createSubmission.py` script.

-0.5 Point for uploading in wrong format!

This exercise is divided into two independent parts:

1. Implement and test a **Naïve Bayes classifier**. The classifier will be applied to an email dataset for spam detection.

2. Implement and test a **Logistic Regression classifier** with and without regularization. The classifiers will first be applied to a toy problem and then to different image datasets.

1 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 *'spmsga'* are spam emails.

1.1 Implementation

All the functions should be implemented in the `naivebayes.py` file and can be tested by running the file `ex3-NaiveBayes.py`.

(a) **Learning** Learn the *likelihood terms* and *prior probabilities* for the Bayes model based on the provided training data. This is done in the `train` function.
Hint:

- Use the provided function `_extractWords`, to turn a files content into a list of words. `Note: There is no need to remove additional stop words, articles or similar. If implemented correctly, your system will automatically detect these as not important for spam/ham classification.`
- Construct the `final_dictionary` as a list of `Word` objects.
- The indicativeness in the `Word` class can be set as: $log(\frac{spam_{likelihood-of-word}}{ham_{likelihood-of-word}})$. By sorting the list based on the indicativeness, you would then get the most/least indicative word for spam in each end of the list.

(b) **Classification Accuracy**

Implement the functions `classify` and `classifyAndEvaluateAllInFolder` that classifies emails using the model estimated above (list of `Word` objects). Test your Naïve Bayes Classifier on the provided test dataset.

Hint: When using X number of features, use both the X most INDICATIVE spam words and the X most INDICATIVE ham words. If you store all the words in a sorted list, then it will just be taking the first and last X items from the list.

2 Logistic Regression

Data (automatically loaded in the provided skeleton-code):

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 row 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.
- `zip38` ($d = 256$): Handwritten digits automatically scanned by the U.S. Postal Service. The set is reduced to the digits 3 and 8. The digits are normalised to a grey scale 16×16 grid.
- `zip13` ($d = 16 \times 16 = 256$): As above with digits 1 and 3.
- `plane_no_plane` ($d = 32 \times 32 \times 3 = 3072$): An extract from the popular CIFAR dataset.

2.1 Classification without regularization

Implement and test a Logistic Regression classifier without regularization. Start with the provided script `logreg.py` for the LOGREG class. Use a `regularization_coefficients` of 0 (in the `ex3-LOGREG_1_Toy.py` file) 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 `activationFunction`. Use the equation of the logistic function.
- Define the cost function `costFunction`. 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.
  HINT: Set values smaller than `self._eps` *to self._eps .*
- Calculate the gradient of the cost function `_calculateDerivative`. You can directly use the derived form of the derivative from the lecture slides.
- Calculate the Hessian matrix `_calculateHessian`. Again, you can directly use the equations derived in the lecture slides.
- Optimize using the Newton-Raphson algorithm `_optimizeNewtonRaphson`. 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 `self._threshold` it is safe to assume convergence.
  HINT: the threshold check can be done by checking if any of the weights in the w_{update} vector are smaller than the set threshold value.
- Train using the `train` function.
- Implement the `classify` function which uses the `self.w` 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.

```
HINT: Remember to NOT regularize the bias term in the derivative and hessian
computations by setting the first regularization term to 0 in these functions.
```

Once your `logreg` class is ready, you can start training and testing on the three provided datasets.

2.2 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 2.1 and modify the `costFunction`, `_calculateDerivative` and `_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 regularize 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.

Use different values for the regularization coefficient: 0 (no regularization), 0.1 and 1.0 (already specified in the code).

3 Theory / coding questions General[1.5p]

- Describe how each of these algorithms (Naive Bayes and Logistic Regression) estimates the posterior distribution of belonging to a class. (1P)
- If you want to not only classify your dataset into two classes but also sample from one class afterwards, which of the two algorithms would you choose and why? (0.5P)

4 Theory / coding questions Naïve Bayes[4p]

- What are the two assumptions that make the Naive Bayes' classifier different from a Bayes' classifier? (1P)
- What are the 3 most found words in *spam* emails? (0.25P)
- What are the 3 most found words in *ham* emails? (0.25P)
- What are the 3 most indicative words for a *spam* email? (0.25P)
- What are the 3 most indicative words for a *ham* email? (0.25P)
- Plot the classification accuracy using the top X features ($X = [1, 2, 5, 10, 20, 30, 40, 50]$). (1P)
- Classification using the Naïve Bayes classifier. Show how this is the case by indicating what each of the terms in the linear classifier formula corresponds to in the Naïve Bayes spam classifier: $g(\boldsymbol{x}) = w_0 + \langle \boldsymbol{w}, \boldsymbol{x} \rangle$? (1P)

5 Theory / coding questions LogReg[4.5p]

- Toy dataset classfication (`ex3-LOGREG_1_Toy.py`)
  - Specify the number of misclassified (both training and test) for all 3 regularization values (0, 0.1, 1.0) (1P)
  - Compare the figures of the three different regularization conditions. How does the value of the regularization coefficient affect the resulting hyperplane? (0.5P)
  - Show the 2D/3D plots for all 3 regularization values. (1P)
- How is regularization introduced into the logistic regression cost function? (1P)
- From a Bayesian viewpoint, regularization determines the width of the normal distribution from which the weights are sampled. How does a large regularization term $r = 1/(2 * \sigma^2)$ influence the normal distribution? (0.5P)
- Specify the number of misclassified (both training and test) for all 3 regularization values (0, 0.1, 1.0) when running `ex3-LOGREG_2_ZIP13.py` (0.5P)