Lab Course Machine Learning
Exercise Sheet 4

Prof. Dr. Dr. Lars Schmidt-Thieme, Shereen Elsayed
Information Systems and Machine Learning Lab
University of Hildesheim

November 26th, 2021
Submission on December 3rd, 2021 at 12 noon, (on learnweb, course code 3116)

Instructions

Please following these instructions for solving and submitting the exercise sheet.

1. You should submit a jupyter notebook detailing your solution.
2. Please set the seed(s) to 3116.
3. Please explain your approach i.e. how you solved a given problem and present your results in form of graphs and tables.
4. Please submit your jupyter notebook to learnweb before the deadline. Please refrain from emailing the solutions except in case of emergencies.
5. Unless explicitly noted, you are not allowed to use scikit, sklearn or any other library for solving any part.
6. Please refrain from plagiarism.

Exercise 0: Dataset preprocessing (2 Points)

In this part, you are required to preprocess the given dataset ('tic-tac-toe.data') according to the steps below:

1. Convert any non-numeric values to numeric values. For example you can replace a country name with an integer value or more appropriately use hot-one encoding. [Hint: use hashmap (dict) or pandas.get_dummies]. Please explain your solution.
2. This dataset is unbalanced, (**show how we can confirm this**). Explain what is stratified sampling and Implement a stratified sampler.
3. Split the data into a train(80%) and test(20%).

Exercise 1: Logistic Regression with Gradient Descent (9 Points)

In this part you are required to implement linear classification with stochastic gradient ascent algorithm. Reference lecture **ml-03-A2-linear-classification.pdf**

- 1. A set of training data $D_{train} = \{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), ..., (x^{(N)}, y^{(N)})\}$, where $x \in R^M, y \in \{0, 1\}$ N is number of training examples and M is number of features

learn-logreg-GA$(\mathcal{D}^{\text{train}} := \{(x_1, y_1), \ldots, (x_N, y_N)\}, \mu, t_{\max} \in \mathbb{N}, \epsilon \in \mathbb{R}^+)$:

  $X := (x_1, x_2, \ldots, x_N)^T$

  $y := (y_1, y_2, \ldots, y_N)^T$

  $\hat{\beta} := 0_M$

  $\ell := \sum_{n=1}^{N} y_n \langle x_n, \hat{\beta} \rangle - \log(1 + e^{\langle x_n, \hat{\beta} \rangle})$

  for $t = 1, \ldots, t_{\max}$:

    $\hat{y} := (1/(1 + e^{-\hat{\beta}^T x_n})_{n \in 1:N}$

    $\hat{\beta} := \hat{\beta} + \mu \cdot X^T (y - \hat{y})$

    $\ell^{\text{old}} := \ell$

    $\ell := \sum_{n=1}^{N} y_n \langle x_n, \hat{\beta} \rangle - \log(1 + e^{\langle x_n, \hat{\beta} \rangle})$

    if $\ell - \ell^{\text{old}} < \epsilon$:

      return $\hat{\beta}$

  raise exception "not converged in $t_{\max}$ iterations"

Figure 1: Algorithm: Learn-logreg-GA

- Logistic Regression model is given as $\hat{y}^n = \sigma(\beta^T \mathbf{x}^n)$ where σ is a logistic function $\frac{1}{1+e^{-\beta^T \mathbf{x}^n}}$
- Optimize the loglikelihood function $log(L_D^{cond})$ using Gradient Ascent algorithm. Implement **(learn-logreg-GA)**. Choose i_{max} between 100 to 1000.
- You will use $bolddriver$ as the step length controller.
  - In each iteration of the algorithm calculate $|f(x_{i-1}) - f(x_i)|$ and at the end of learning, plot it against iteration number i. Explain the graph.
  - In each iteration step also calculate logloss on test set (see ref:`https://www.kaggle.com/wiki/LogarithmicLoss`), plot it against iteration number i. Explain the graph.

Exercise 2: Implement Newton Algorithm for Logistic Regression (9 Points)

In this task you have to implement Newton Algorithm given in Fig. 3. Use the 'tic-tac-toe' dataset.

- In each iteration of the algorithm calculate $|f(x_{i-1}) - f(x_i)|$ and at the end of learning, plot it against iteration number i. Explain the graph.
- In each iteration step also calculate logloss on test set (see ref above.), plot it against iteration number i.

Comment on the rate of convergence in the light of plots from above.

```
minimize-Newton(f : ℝ^N → ℝ, x^(0) ∈ ℝ^N, μ, t_max ∈ ℕ, ε ∈ ℝ^+):
  for t := 1, ..., t_max:
    g := ∇f(x^(t−1))
    H := ∇²f(x^(t−1))
    x^(t) := x^(t−1) − μ H^(−1) g
    if f(x^(t−1)) − f(x^(t)) < ε:
      return x^(t)
  raise exception "not converged in t_max iterations"
```

$x^{(0)}$ start value
μ (fixed) step length / learning rate
$t_{\max}$ maximal number of iterations
ϵ minimum stepwise improvement
$\nabla f(x) \in \mathbb{R}^N$: gradient, $(\nabla f(x))_n = \frac{\partial}{\partial x_n} f(x)$
$\nabla^2 f(x) \in \mathbb{R}^{N \times N}$: Hessian matrix, $\nabla^2 f(x)_{n,m} = \frac{\partial^2 f}{\partial x_n \partial x_m}(x)$

Figure 2: Algorithm: minimize Newton

```
learn-logreg-Newton(D^train := {(x_1, y_1), ..., (x_N, y_N)}, μ, t_max ∈ ℕ, ε ∈ ℝ^+):
  ℓ := −log L_D^cond(β̂) := Σ_{n=1}^N y_n⟨x_n, β̂⟩ − log(1 + e^⟨x_n, β̂⟩)
  β̂ := minimize-Newton(ℓ, 0_M, μ, t_max, ε)
  return β̂
```

Figure 3: Algorithm: Newton Algorithm