Applied Machine Learning
Exercise 4

Prof. Dr. Amr Alanwar

November 2024

Classification Datasets

- **Bank Marketing:** `https://archive.ics.uci.edu/ml/datasets/Bank+Marketing`
- **Occupancy Detection:** `https://archive.ics.uci.edu/dataset/357/occupancy+detection`

You are required to pre-process the given datasets as follows:

1. Convert any non-numeric values to numeric values. For example, you can replace a country name with an integer value or use one-hot encoding. *(Hint: use hashmap (dict) or pandas.get_dummies).* Please explain your solution.
2. If required, drop rows with missing values or NA. In the next lectures, we will handle sparse data, allowing us to use records with missing values.
3. Split the data into a train (80%) and test (20%) set.

1 Linear Classification with Gradient Descent

Exercise 1: Linear Classification with Stochastic Gradient Descent/Ascend (5 Points)

In this part, you are required to implement a linear classification algorithm using the stochastic gradient descent/ascend algorithm.

For each dataset mentioned above:

1. Define a set of training data $D_{train} = \{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(N)}, y^{(N)})\}$, where $x \in \mathbb{R}^M$ and $y \in \{0, 1\}$. N is the number of training examples, and M is the number of features.
2. The linear regression model is given as $\hat{y}_n = \sigma(\beta^T x_n)$, where σ is a logistic function:
$$\sigma(z) = \frac{1}{1 + e^{-\beta^T x_n}}$$
3. Optimize the log-likelihood function $l(x, y)$ using the gradient descent algorithm. Implement both log-regSGA/SGD and SGA/SGD algorithms. Choose $i_{\max}$ between 100 and 1000.
4. Use the *steplengthbolddriver* for choosing the step length:

- In each iteration of the SGA/SGD algorithm, calculate $|f(x_{i-1}) - f(x_i)|$ and, at the end of learning, plot it against the iteration number i. Explain the resulting graph.
- In each iteration step, also calculate the log-loss on the test set, and plot it against iteration number i. Explain this graph as well.

2 Exercise 2: Implement AdaGrad for Adaptive Step Length (Learning Rate) (5 Points)

In this task, you are required to implement the AdaGrad algorithm as presented in the lecture slides.

1. In each iteration of the SGA/SGD algorithm, calculate $|f(x_{i-1}) - f(x_i)|$ and, at the end of learning, plot it against the iteration number i. Provide an explanation of the graph.
2. In each iteration step, also calculate the log-loss on the test set and plot it against iteration number i. Explain this graph.
3. Compare AdaGrad with the *steplengthbolddriver* algorithm:
    - Compare the log-loss graphs of AdaGrad and the *steplengthbolddriver* algorithms, and explain the resulting graphs.

ANNEX

- You can use `numpy` or `scipy` for linear algebra operations.
- `pandas` can be used for data reading and processing.
- `matplotlib` may be used for plotting.
- Do not use any machine learning libraries (e.g., `scikit-learn`) for solving the problem. Using them will result in no points for the task.

Algorithm 1 maximize-GA

Input: $f : \mathbb{R}^N \to \mathbb{R}$, starting point $x^{(0)} \in \mathbb{R}^N$, step length μ, maximum iterations $t_{\max}$, tolerance ϵ
for $t = 1$ **to** $t_{\max}$ **do**
    $x^{(t)} := x^{(t-1)} + \mu \cdot \frac{\partial f}{\partial x}(x^{(t-1)})$ # Gradient ascent update
    if $|f(x^{(t)}) - f(x^{(t-1)|}) < \epsilon$ **then**
        return $x^{(t)}$
    end if
end for
raise exception "not converged in $t_{\max}$ iterations"

Definitions:

- $x^{(0)}$: initial starting point
- μ: (fixed) step length / learning rate
- $t_{\max}$: maximal number of iterations
- ϵ: minimum improvement threshold
- $\frac{\partial f}{\partial x}(x^{(t-1)})$: gradient of f with respect to x at $x^{(t-1)}$

Algorithm 2 minimize-Newton

Input: $f : \mathbb{R}^N \to \mathbb{R}$, starting point $x^{(0)} \in \mathbb{R}^N$, step length μ, maximum iterations $t_{\max}$, tolerance ϵ
for $t = 1$ **to** $t_{\max}$ **do**
    $g := \nabla f(x^{(t-1)})$ # Gradient at $x^{(t-1)}$
    $H := \nabla^2 f(x^{(t-1)})$ # Hessian at $x^{(t-1)}$
    $x^{(t)} := x^{(t-1)} - \mu H^{-1} g$ # Newton's update step
    if $f(x^{(t-1)}) - f(x^{(t)}) < \epsilon$ **then**
        return $x^{(t)}$
    end if
end for
raise exception "not converged in $t_{\max}$ iterations"

Definitions:

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

Algorithm 3 learn-logreg-GA

1: **Input:** training data $D^{\text{train}} := \{(x_1, y_1), \ldots, (x_N, y_N)\}$, step length μ, maximum iterations $t_{\max}$, tolerance ϵ

2: $\ell := \log L_D^{\text{cond}}(\beta) := \sum_{n=1}^N y_n \langle x_n, \beta \rangle - \log\left(1 + e^{\langle x_n, \beta \rangle}\right)$ # Log-likelihood

3: $\hat{\beta} :=$ maximize-GA$(\ell, 0_M, \mu, t_{\max}, \epsilon)$

4: **return** $\hat{\beta}$

Algorithm 4 learn-logreg-Newton

1: **Input:** training data $D^{\text{train}} := \{(x_1, y_1), \ldots, (x_N, y_N)\}$, step length μ, maximum iterations $t_{\max}$, tolerance ϵ

2: $\ell := \log L_D^{\text{cond}}(\beta) := \sum_{n=1}^N y_n \langle x_n, \beta \rangle - \log\left(1 + e^{\langle x_n, \beta \rangle}\right)$ # Log-likelihood

3: $\hat{\beta} :=$ minimize-Newton$(\ell, 0_M, \mu, t_{\max}, \epsilon)$

4: **return** $\hat{\beta}$

Algorithm 5 Stochastic Gradient Descent (SGD)

Require: Objective function $f(\theta; x, y)$, initial parameters $\theta^{(0)}$, learning rate η, maximum number of epochs $t_{\max}$, tolerance ϵ, training data $\{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$

```
Initialize θ := θ^(0)
for epoch = 1, ..., t_max do
    Shuffle the training data
    for each training example (x_i, y_i) do
        Compute the gradient of the loss with respect to θ for (x_i, y_i):
            g := ∇f(θ; x_i, y_i)
        Update parameters:
            θ := θ − η · g
    end for
    if convergence criterion is met (e.g., ‖η · g‖ < ε) then
        return θ
    end if
end for
raise exception "not converged in t_max epochs"
```

Algorithm 6 Logistic Regression using SGD

Require: Training data $\{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$ where $y_i \in \{0, 1\}$, learning rate η, initial parameters $\theta^{(0)}$, maximum number of epochs $t_{\max}$, tolerance ϵ

Define the logistic regression objective function for a single example (x_i, y_i):

$$f(\theta; x_i, y_i) := -y_i \log(\sigma(\theta^T x_i)) - (1 - y_i) \log(1 - \sigma(\theta^T x_i))$$

where $\sigma(\theta^T x_i) = \frac{1}{1+e^{-\theta^T x_i}}$

Call **SGD**$(f, \theta^{(0)}, \eta, t_{\max}, \epsilon, \{(x_i, y_i)\})$

return Optimized parameters θ