

Homework 1 - Learning By Example - Spring 2025

Essay Question 1: *There are numerous factors that led to the explosive interest in Machine Learning since 2012.* Write an essay that starts with this sentence and goes on to identify and describe the main forces giving rise to this renewed interest. Be sure to discuss how/why the forces you list led to this interest. Conclude your essay with a single sentence summarizing the main assertion in your essay.

Problem 1: Consider a logistic regression problem whose targets are in the set $Y = \{0, 1\}$. Show that minimizing the negative log likelihood of the dataset is equivalent to minimizing the dataset's empirical risk based on the binary cross-entropy loss function.

Problem 2: Consider a learning-by-example problem whose inputs $x \in [0, 2]$ and whose targets $y \in \{-1, +1\}$. Assume the "system" draws samples (x, y) with an equal probability of the sample being in class -1 or $+1$. Assume that the conditional probability of the inputs, x are

$$P(x | y = -1) = \begin{cases} 1 & \text{if } 0 \leq x \leq 1 \\ 0 & \text{otherwise} \end{cases}, \quad P(x, | y = +1) = \begin{cases} 1/2 & \text{if } 0 \leq x \leq 2 \\ 0 & \text{otherwise} \end{cases}$$

Assume the model has the form $h_w(x) = \text{sgn}(x - w)$ where $w \in \mathbb{R}$ is the weight. Assume the loss function is $L(y, h_w(x)) = \mathbb{1}(y \neq h_w(x))$.

1. Determine the true risk, $R[h_w]$, as a function of w .
2. Determine the optimal model weight, w , that minimizes the model's true risk, $R[h_w]$.

Notebook Assignment 1: We will consider a logistic regression problem that trains a model using a dataset consisting of N input vectors, $x_k \in \mathbb{R}^n$ and N target labels $y_k \in \{0, 1\}$ for $k = 1, \dots, N$. The dataset $\mathcal{D} = (\mathbf{X}, \mathbf{Y})$ consists of two data matrices

$$\mathbf{X} = [x_1 \quad x_2 \quad \cdots \quad x_N], \quad \mathbf{Y} = [y_1 \quad y_2 \quad \cdots \quad y_N]^T$$

The model is $h_w : \mathbb{R}^n \rightarrow [0, 1]$ where

$$h_w(x) = \sigma(w^T x) \tag{1}$$

where $w \in \mathbb{R}^n$ is a weight vector and $\sigma(s) = \frac{1}{1 + e^{-s}}$ is a softmax function. The loss function will be the binary cross-entropy function you examined in problem 1.

Such models are often trained using *gradient descent* algorithms. These algorithms update the weights of the model using the equation

$$w_{k+1} \leftarrow w_k - \eta \frac{\partial \widehat{R}_{\mathcal{D}}(w)}{\partial w} \tag{2}$$

where $\eta > 0$ is called the *learning rate* and $\widehat{R}_{\mathcal{D}}(w)$ is the empirical risk function of the model h_w on the given dataset. The algorithm starts with an initial weight, w_0 and then updates w_k using equation (2) until a termination condition is satisfied. The termination condition is satisfied if either

1) the number of recursions (`iter`) exceed a specific limit (`maxiter`) or 2) that the Euclidean norm of the difference between two successive weight updates

$$|w_{k+1} - w_k| \leq \text{tol}$$

where `tol` is a specified *tolerance* level.

The training data you will use for this notebook exercise is a numpy array `ring_data.npy`. This numpy array has a shape $(3, 2000)$. It may be seen as a matrix whose k th column is the k th sample in the dataset ($k = 1, 2, \dots, 2000$). The first two elements of the k th column is the input $x_k \in \mathbb{R}^2$ and the third element of that column is the target label y_k . If you want to use this matrix in Google Colab, you'll first need to upload it from your local directory into Google Colab session storage. The specific tasks to be completed in this assignment are enumerated below.

1. **TO DO:** Load the data set `ring_data.npy` and print the dataset's shape. The first two columns are the components of the input sample vector. The third column is the target data. Take the loaded data and crate an input data array, `X`, and target array, `Y`.

TO DO: The class targets in the data set are drawn from $\{-1, 1\}$. Since the loss function you will be using assumes non-negative integers for class labels, you will have to transform the negative targets (-1) in `Y` to 0.

TO DO: Do a scatter plot of the data, painting the +1 class samples blue and the 0 class samples green.

2. **TO DO:** Write two Python functions

```
def Rhat(w, X, Y):  
def gradRhat(w, X, Y):
```

whose inputs are the weight vector, w , input data matrix, `X` and target matrix `Y`. The function `Rhat` returns the empirical loss at w for the dataset $\mathcal{D} = (\mathbf{X}, \mathbf{Y})$. The function `gradRhat` returns the gradient $\frac{\partial \widehat{R}_{\mathcal{D}}(w)}{\partial w}$ evaluate at w for dataset $\mathcal{D} = (\mathbf{X}, \mathbf{Y})$. To test your function, compute the empirical risk and gradient of that risk on the given dataset when the components of the model parameter vector, $w = [-1, 1]$.

3. **TO DO:** Write a Python function

```
def fit(X, Y, lr, maxiter, tol):
```

whose input arguments are the numpy array of input samples, `X`, numpy array of binary targets, `Y`, the learning rate (`lr`) η , and the hyperparameters `maxiter` and `tol`. Have this function return the last weight vector, w , obtained after the gradient descent algorithm terminates and a numpy array, `history`, whose entries contain the value of the loss function and the weight vector computed after each recursion of the gradient descent algorithm.

TO DO: Test your function with a learning rate $\eta = 0.1$ for a maximum of 100 iterations (`maxiter`) and a tolerance (`tol`) of 10^{-4} . Then plot the empirical loss as a function of the iteration.

TO DO: Scatter plot for the dataset's input samples, x_k , coloring the class 1 samples blue and the class 0 samples green. On the scatter plot also plot the initial discriminant surface and the final discriminant surfaces obtained after the gradient descent algorithm terminates.