REPORT

Course:
Machine Learning 1
Homework 1:
Linear and Logistic Regression and Gradient Descent
Group:
233

Authors:
Shah Md Al Mohan - shahmdal@student.tugraz.at
Katarina Štor - katarina.stor@student.tugraz.at

Univariate Linear Regression

1.1.1. Partial derivation of the loss function w.r.t. w and b:

In univariate linear regression, we model the relationship between a single input feature x and an output y using a simple equation. The function of theta: $f_\theta(x) = b + wx$, where w is the weight and b is the bias. To find the optimal values for w and b, we need a loss function that measures how well the model fits the data. For this we use Mean Squared Error (MSE) as the loss function: $\mathcal{L}_U(\theta) = \frac{1}{N}\sum_{i=1}^{N}(f_\theta(x^{(i)}) - y^{(i)})^2$

The following model has been expanded by substituting the model equation so now we have a loss function of w and b: $\mathcal{L}_u(b,w) = \frac{1}{N}\sum_{i=1}^{N}(b + wx^{(i)} - y^{(i)})^2$

To find the optimal value of b, we need to compute the partial derivative of the loss function with respect to b: $\frac{\partial \mathcal{L}_u}{\partial b} = \frac{1}{N}\sum_{i=1}^{N} 2(b + wx^{(i)} - y^{(i)})$

Next, we also do the partial derivative of the loss function with respect to w:

$$\frac{\partial \mathcal{L}_u}{\partial b} = \frac{1}{N}\sum_{i=1}^{N} 2(b + wx^{(i)} - y^{(i)}) \cdot x^{(i)}$$

Derivative of w is similar to the one for b, but without the $x^{(i)}$ term. This is because b is a constant term that equally affects all predictions, while w is multiplied by features.
At the minimum of the loss function, the derivative equals zero. This is where the optimal value of b and w is found. So, the next step is to set both derivatives equal to 0.

For b: $\sum_{i=1}^{N}(b + wx^{(i)} - y^{(i)}) = 0$

For w: $\sum_{i=1}^{N}(b + wx^{(i)} - y^{(i)}) \cdot x^{(i)} = 0$

First we will find the value for b and we do this by dividing both sides of the equation by N:

$$\sum_{i=1}^{N}(b + wx^{(i)} - y^{(i)}) = 0 \Rightarrow Nb + w\sum_{i=1}^{N}x^{(i)} = \sum_{i=1}^{N}y^{(i)}$$

$$b + w\bar{x}^{(i)} = \bar{y}^{(i)}$$

Where x bar is the mean of all $x^{(i)}$ and y bar is the mean of all $y^{(i)}$:

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x^{(i)} \qquad \bar{y} = \frac{1}{N}\sum_{i=1}^{N} y^{(i)}$$

Solving for b, we get that the optimal bias equals the mean of y minus w times the mean of x.

$$b = \bar{y} - w \cdot \bar{x}$$

Next, we need to find the optimal value of w.

First substitute the b value in the derivative of w that is set equal to 0.

$$\sum_{i=1}^{N} (\bar{y} - w\bar{x} + wx^{(i)} - y^{(i)})\, x^{(i)} = 0$$

After that distribute the equation.

$$\sum_{i=1}^{N} \left((\bar{y} - y^{(i)}) + w(x^{(i)} - \bar{x})\right) x^{(i)} = 0$$

Separate the sums.

$$\sum_{i=1}^{N} (\bar{y} - y^{(i)})\, x^{(i)} + w \sum_{i=1}^{N} (x^{(i)} - \bar{x})\, x^{(i)} = 0$$

And lastly switch $\sum_{i=1}^{N} (\bar{y} - y^{(i)})\, x^{(i)}$ to the left side of equation and divide both sides by $\sum_{i=1}^{N} (x^{(i)} - \bar{x})\, x^{(i)}$:

$$w = \frac{\sum_{i=1}^{N} (y^{(i)} - \bar{y})\, x^{(i)}}{\sum_{i=1}^{N} (x^{(i)} - \bar{x})\, x^{(i)}}$$

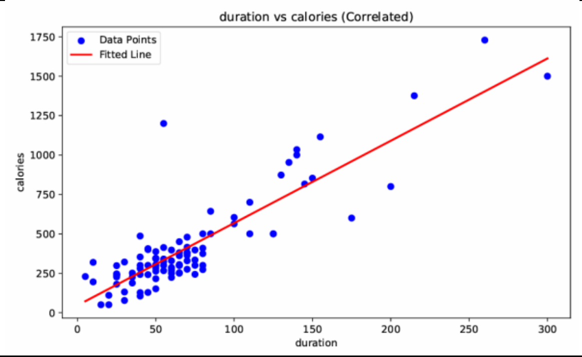
This equation can be also written as the standard covariance divided by variance form:

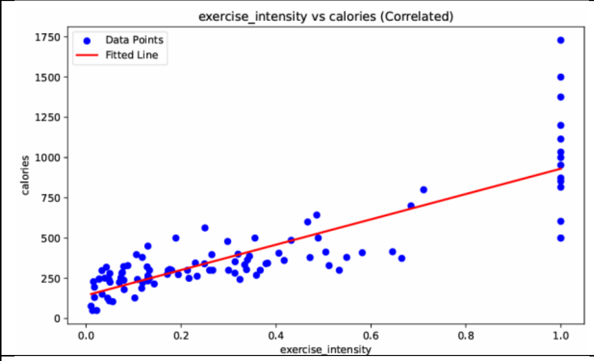
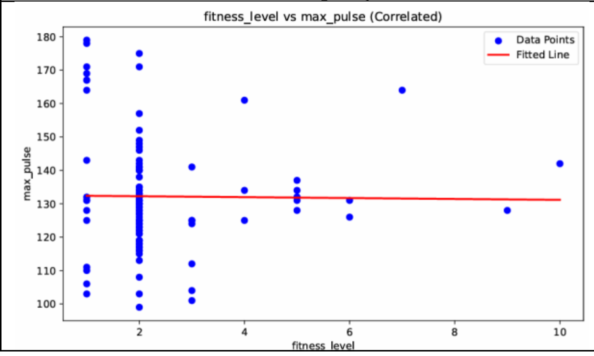
$$w^* = \frac{\sum_{i=1}^{N} (x^{(i)} - \bar{x})(y^{(i)} - y)}{\sum_{i=1}^{N} (x^{(i)} - x)^2}$$

Finally, we recompute the bias using the optimal w:

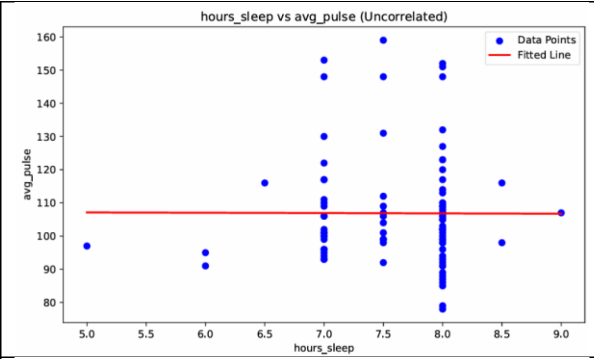
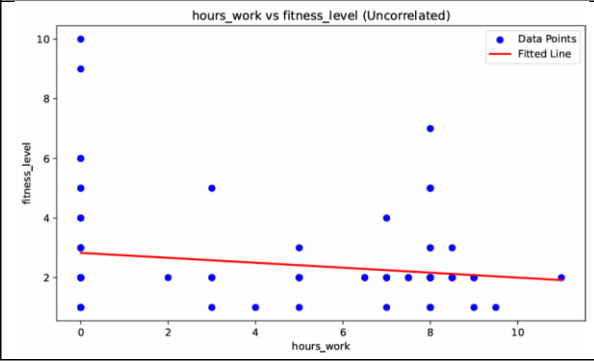
$$b^* = \bar{y} - w \cdot \bar{x}$$

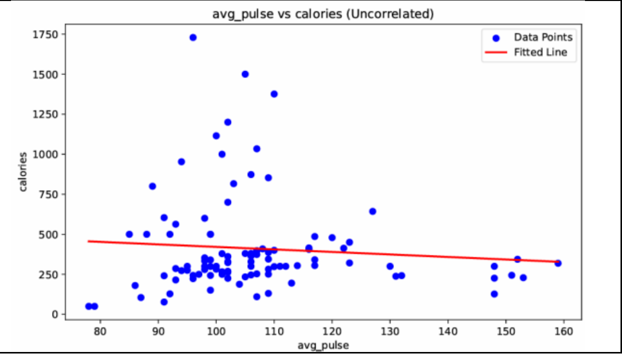
1.1.2. Selected variable pairs with meaningful relations between them:

exercise_duration & calories_burned: Pearson correlation coefficient: r = 0.8690 Parameter vector: θ* = [46.2387, 5.2211] MSE: LU(θ*) = 21843.03	

exercise_intensity & calories_burned: Pearson correlation coefficient: r = 0.8330 Parameter vector: θ* = [144.2855, 786.5502] MSE: LU(θ*) = 27311.77	
fitness_level & max_pulse: Pearson correlation coefficient: r = −0.0123 Parameter vector: θ* = [132.5193, −0.1361] MSE: LU(θ*) = 298.63	

1.1.3. Selected variable pairs that are not linearly dependent:

hours_sleep & average_pulse: Pearson correlation coefficient: r = −0.0035 Parameter vector: θ* = [107.6003, −0.0957] MSE: LU(θ*) = 253.51	
hours_work & fitness_level: Pearson correlation coefficient: r = −0.1965 Parameter vector: θ*=[2.8304, −0.0829] MSE: LU(θ*)=2.33	

average_pulse & calories_burned: Pearson correlation coefficient: $r = -0.0838$ Parameter vector: $\theta^* = [578.3271, -1.5712]$ MSE: $LU(\theta^*) = 88581.30$	

1.1.4. Interpretation of the scatter plots & Pearson correlation coefficient:

Scatter plots of correlated pairs:

For the **exercise_duration and calories_burned** pair, the points form a clear upward trend, and the fitted line aligns well with the data. This indicates a strong positive linear relationship.

In the case of **exercise_intensity and calories_burned**, the data points also show a general upward trend. While the fitted line captures the overall direction of the relationship, there is more variation in the data, particularly at higher intensity levels. This pair also suggests a strong positive linear relationship.

For the **fitness_level and max_pulse** pair, the data points are widely scattered without any clear trend, and the fitted line is almost flat. This indicates that there is no meaningful linear relationship.

Scatter plots of uncorrelated pairs:

For the **hours_sleep and average_pulse** pair, the data points are widely scattered without any clear trend, and the fitted line is almost flat. This indicates that there is no meaningful linear relationship.

For the **hours_work and fitness_level**, the data points show a slight downward trend, but they are still quite scattered, and the fitted line is only slightly sloped. This suggests a weak negative linear relationship.

For the **average_pulse and calories_burned**, the data points are spread out with no obvious pattern, and the fitted line is almost flat. This indicates that there is no meaningful linear relationship.

Pearson correlation coefficient (r)

Range of Values: $-1 \leq r \leq 1$

Value of **r**	Interpretation

r = 1	Perfect **positive** linear correlation
0 < r < 1	**Positive** linear relationship
r = 0	**No** linear correlation
-1 < r <0	**Negative** linear relationship
r = -1	Perfect **negative** linear correlation

The closer **|r|** is to **|1|**, the stronger is the linear relationship between two variables. If the value is near **0**, then this means weak or no linear dependence, even if some nonlinear pattern exists.

1.1.5. Can a small positive slope result in a high Pearson correlation coefficient?

No, if the slope w^* is very small (close to zero), then r cannot be close to 1. There are many reasons for this such as:

1. The slope w^* in the linear model represents the strength of the influence of x on y.
2. A small slope means that as x changes, y barely changes, suggesting little to no linear relationship.
3. Pearson correlation coefficient r measures how well the data follows a linear trend.

If $r \approx 1$, then it indicates a strong, consistent, positive linear relationship, which would show as a steep upward trend in the data. However, a near-zero slope means the regression line is almost flat, showing almost no relationship between x and y. Therefore, it is not possible for the Pearson correlation coefficient to be close to 1.

Multiple Linear Regression

1.2.1. Definition and derivation of the design matrix X:

<u>Definition of the design matrix X:</u>

The matrix $X \in \mathbb{R}^{N \times (D+1)}$ is defined as:

$$X = \begin{bmatrix} 1 & x_1^{(1)} & \cdots & \cdots x_1 \\ 2_1 & x_2^{(2)} & \cdots & \cdots x_D \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_n^{(N)} & \cdots & \cdots x_D \end{bmatrix}$$

That is: $X_{i1} = 1$, $X_{i(j+1)} = x_j^{(i)}$ for $i = 1, \dots, N$; $j = 1, \dots, D$

Dimension of X is: $X \in \mathbb{R}^{N \times (D+1)}$

<u>Derivative of the design matrix X:</u>

First define the multiple regression loss. $L_M(\theta) = \frac{1}{N} \|X\theta - y\|_2^2$

You start by expanding the loss function. $L_M(\Theta) = \frac{1}{N} (X\theta - y)^T (X\theta - y)$

After that you compute the gradient $\nabla_\theta L_M(\theta)$. $\nabla_\theta [(X\theta - y)^T (X\theta - y)] = 2X^T (X\theta - y)$

$\nabla_\theta L_M(\theta) = \frac{2}{N} X^T (X\theta - y)$

Next step is to set the gradient to 0. $\nabla_\theta L_M(\theta^*) = 0 \Rightarrow X^T (X\theta^* - y) = 0$

After that solve for θ* so you can find the equation of the θ*. $X^T X\theta = X^T y$

$\theta^* = (X^T X)^{-1} X^T y$ (if the equation is invertible)

In the case where the equation of θ* is not invertible or poorly conditioned, then you should use the Moore-Penrose pseudoinverse: $\theta^* = X^+ y = (X^T X)^+ X^T y$. This is the closed-form solution of the multiple linear regression, and it always works with the be singular matrices.

1.2.3: Multiple linear regression on smartwatch data:

The selected variable pair from task 1.1. is **exercise_duration** and **calories_burned**. Calories burned is kept as a fixed y. New additional feature added is **exercise_intensity**.

Using the function **fit_multiple_lin_model**, the model returns the optimal parameter vector: θ^*_M = [49.8088, 3.3669, 371.6853]

- Intercept (bias): 49.8088
- Weight for exercise_duration: 3.3669
- Weight for exercise_intensity: 371.6853

The final model equation is: $f\,\theta^*_M(x) = 49.8088 + 3.3669 \cdot x_1 + 371.6853 \cdot x_2$

Multiple Linear Regression MSE: $L_M(\theta^*_M) = 16517.21$

Univariate Regression MSE using exercise_duration alone: $L_U(\theta^*_M) = 21843.03$

Comparison between $L_M(\theta^*_M)$ & $L_U(\theta^*_M)$: The multiple linear regression model has a lower MSE compared to the univariate model. This shows that using more relevant features improves predictive accuracy by capturing more variance in the target variable.

1.2.4. Mathematical proof of the statement:

The least squares solution minimizes the squared distance between y and the column space of the design matrix. Let: $L_U = \mathrm{Col}(X_U) \subseteq RN$ be the space of all predictions by the univariate model. Because X_U is a submatrix of X_M (using fewer features), and both matrices contain the same bias column. We have $L_U \subseteq L_M$, where all predictions of the univariate model can make also lie within the space that the multivariate model can access (but not vice versa).

From linear algebra: If $L_U \subseteq L_M$, then the orthogonal projection of y onto L_M is closer to y than the projection onto L_U.

Formally: $\|y - \hat{y}_M\|^2 \leq \|y - \hat{y}_U\|^2$

Therefore: $L_M(\theta^*_M) = \frac{1}{N}\|y - X_M\theta^*_M\|^2 \leq \frac{1}{N}\|y - X_U\theta^*_U\|^2 = L_U(\theta^*_U)$

Polynomial Regression

1.3.1. New design matrix X with a polynomial model:

To express this model in the matrix form: $L_M(\theta) = \frac{1}{N} \| X\theta - y \|_2^2$, we define the design matrix $X \in \mathbb{R}^{N \times (K+1)}$ such that: $X_{ij} = (x^{(i)})^{j-1}, \quad \text{for } i = 1, \dots, N, \ j = 1, \dots, K+1$

The matrix X looks like:

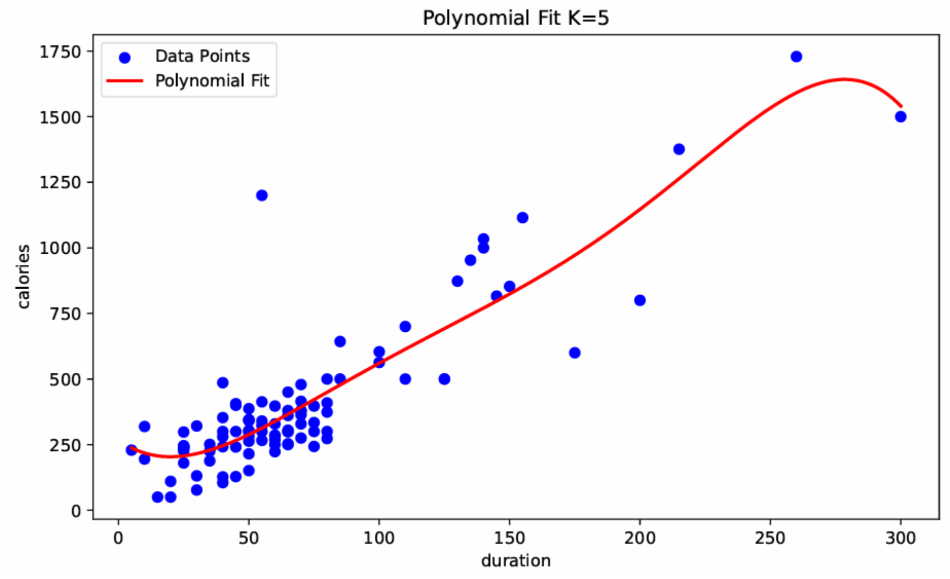
$$X = \begin{bmatrix} 1 & x^{(1)} & (x^{(1)})^2 & \cdots & (x^{(1)})^K \\ 1 & x^{(2)} & (x^{(2)})^2 & \cdots & (x^{(2)})^K \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x^{(N)} & (x^{(N)})^2 & \cdots & (x^{(N)})^K \end{bmatrix} \in \mathbb{R}^{N \times (K+1)}$$

This allows to write the model predictions for all N data points as: $\hat{y} = X\theta$ and to find the optimal parameters through the least squares: $\theta^* = \underset{\theta}{\arg\min} \frac{1}{N} \| X\theta - y \|_2^2$

1.3.2. Polynomial Regression for Improved Fit

Polynomial fit:

Here we picked the same variable pair: exercise_duration & calories_burned, because it showed the strongest linear correlation (r = 0.8690). To improve the model fit, we applied polynomial regression of degree K = 5. The MSE we achieved is $L_M(\theta) = 20357.56$.



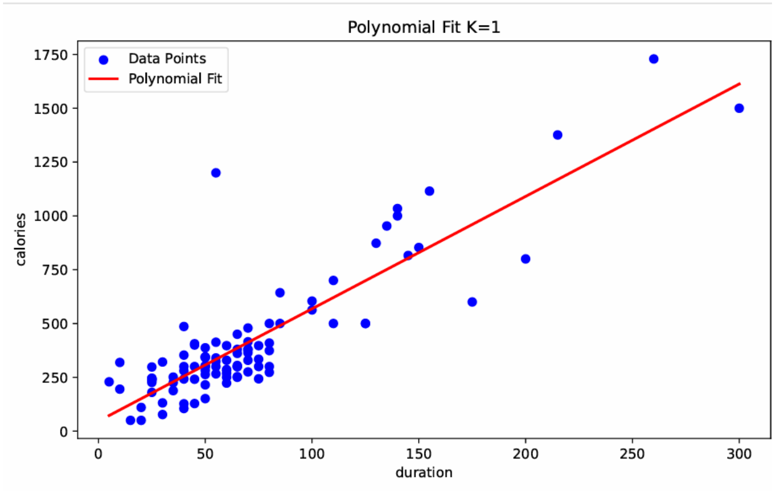
Comparison between multilinear regression loss and univariate loss:

Linear regression with K=1:

Model: $f_\theta(x) = 46.2387 + 5.2211\, x$

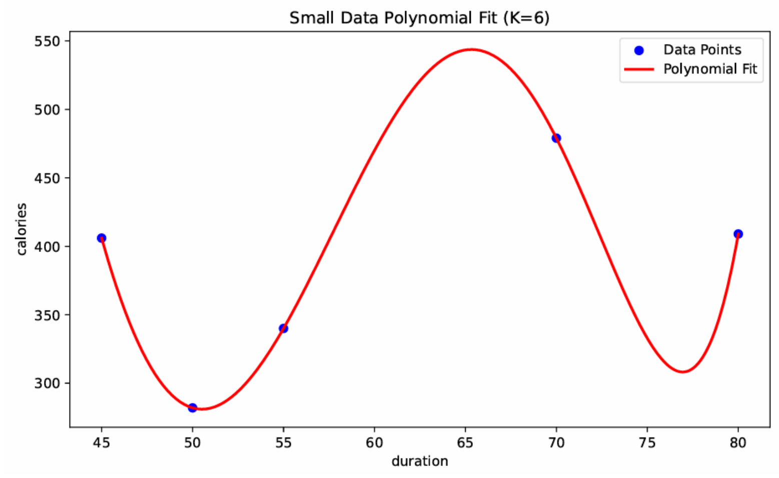
MSE: $L_M(\theta^*) = 21843.03$

The polynomial model of degree significantly improves the fit over the univariate linear model, reducing the MSE from 21843.03 to 20357.56. The polynomial curve also better captures the nonlinear trend visible in the data.



1.3.3. Polynomial regression with a degree K polynomial:

The smallest value of K that gives zero loss is: **K=4.** For K=4, the model interpolates all 5 data points exactly.



To achieve zero loss, the system must satisfy: $\mathbf{X\theta = y}$, where $X \in \mathbb{R}^{5\times(K+1)}$ is the polynomial design matrix, $\theta^* \in \mathbb{R}^{K+1}$ is the parameter vector and $y \in \mathbb{R}^5$ is the target vector.

For $\mathbf{X\theta = y}$ to have an exact solution, the system of equations must be determined, that means that the number of unknowns must match the number of equations:

$$\text{Exact fit} \Leftrightarrow (K+1) = N \Rightarrow K = N - 1 = 4$$

Logistic Regression

2.1. Design Matrix Setup:

	Dataset 1	Dataset 2	Dataset 3
Shape of features	(800,2)	(800,2)	(711,2)
Design matrix	$X = \begin{bmatrix} x_1^{(1)} & x_2^{(1)} \\ \vdots & \vdots \\ x_1^{(1)} & x_2^{(N)} \\ \vdots & \vdots \\ x^{(N)} & x_2^{(N)} \end{bmatrix} \in \mathbb{R}^{800\times 2}$	$X = \begin{bmatrix} x_1^{(1)} & x_2^{(2)} & x_{1x}^{(1)} & x_2^{12} \\ x_1^{(2)} & x_2^{(2)} & x_{1x}^{(x)} & x_2^{12} \\ \vdots & \vdots & \vdots & \vdots \\ x_N^{(N)} & x_N^{(N)} & x_{1N}^{(N)} & x_2^{2'} \end{bmatrix} \in \mathbb{R}^{800\times 5}$	$X \in \mathbb{R}^{11\times 9}$
Features used	/	$\varnothing(x_1, x_2) =$ $[x_1, x_2, x_1x_2, x_1^2, x_2^2]$	Polynomial feature expansion up to degree 4: $(x_1, x_2) = [x_1, x_2, x_1^2, x_2^2, x_1x_2, x_1^3, x_2^3, x_1^2 x_2, x_1x_2^2]$

2.3. Training and Evaluation of Logistic Regression Models:

Dataset 1:

Penalty used: l2
Features used: [x1, x2]
Train accuracy: 83.21%
Test accuracy: 85.83%
Train loss: 0.3261
Test loss: 0.3086

Dataset 2:

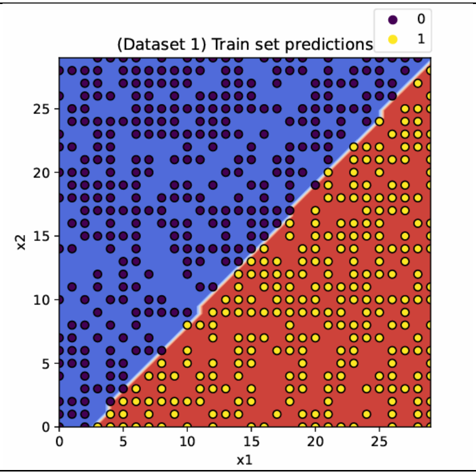
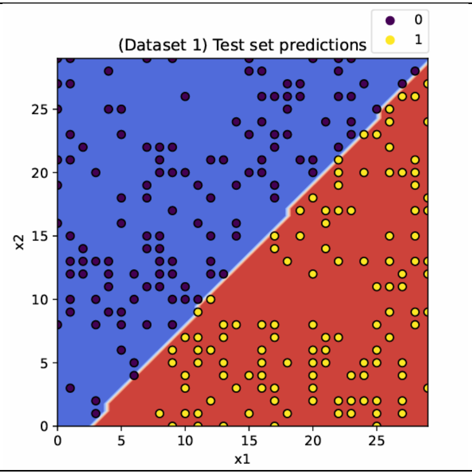
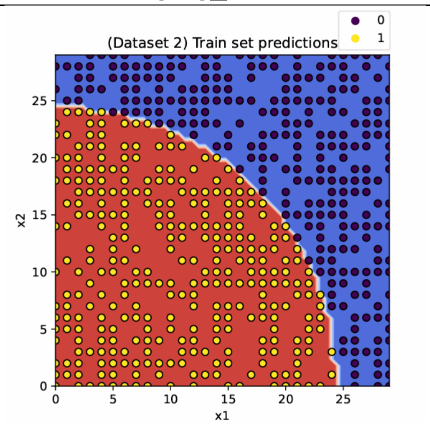
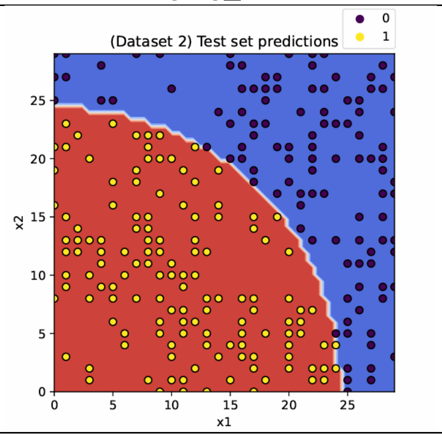
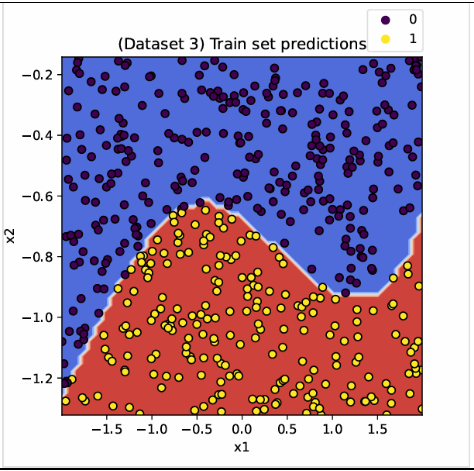
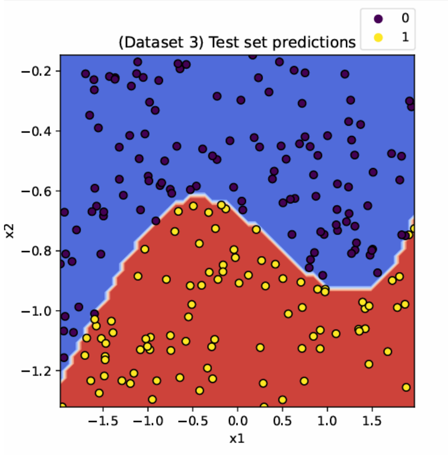
Penalty used: l2
Features used: Polynomial: $[x_1, x_2, x_1x_2, x_1^2, x_2^2]$
Train accuracy: 100.00%
Test accuracy: 100.00%
Train loss: 0.0005
Test loss: 0.0006

Dataset 3:

Penalty used: l2
Features used: Polynomial: $[x_1, x_2, x_1^2, x_2^2, x_1x_2, x_1^3, x_2^3, x_1^2 x_2, x_1x_2^2]$
Train accuracy: 89.54%
Test accuracy: 92.06%
Train loss: 0.2253
Test loss: 0.2178

2.4. Plots of the train and test set predictions:

Dataset 1	
logreg_train1	logreg_test1

	
Dataset 2	
logreg_train2	logreg_test2
	
Dataset 3	
logreg_train3	logreg_test3
	

2.5. Model Parameters:

Dataset 1:	**Dataset 2:**	**Dataset 3:**
Weights: w = [0.2777, −0.2565]	Weights (with polynomial features):	Weights (with 9 polynomial features):

	w = [0.0023, −0.0274, 0.0221, −0.6561, −0.6541]	w = [−0.9517, −2.5718, 0.9212, −1.6395, 2.6100, 1.1819, −1.8647, −0.3714, −0.7053]
Bias: b = −0.6903	Bias: b = 388.6538	Bias: b = −3.4731

2.6. Does 100% accuracy imply zero cross-entropy loss?

The statement is not correct. The model may classify all samples correctly but still incur positive loss if its predicted probabilities are not perfectly confident. Accuracy measures whether the predicted class labels match the true labels. While Cross-entropy loss also considers the confidence of the predictions. It penalizes predictions that are correct but made with low confidence.

For example, if the true label is 1 and the model predicts a probability of 0.51 for class 1 (i.e., it is just slightly above the decision threshold), the prediction is correct, but the loss is still non-zero. In contrast, the cross-entropy loss becomes 0 only if the model assigns a probability of exactly 1 to the correct class and 0 to the incorrect class for every sample.

Gradient Descent

3.3: Partial derivatives of f w.r.t. x and y of the Rastrigin function:

We are given the Rastrigin function: $f(x,y) = 20 + x^2 + y^2 + 10\cos(2\pi x) - 1 - \cos(2\pi y)$

Derive the function with x: $\frac{\partial f}{\partial x} = \frac{d}{dx}\left(20 + x^2 + y^2 - 10\cos(2\pi x) - 10\cos(2\pi y)\right)$

Differentiate each term and set all of them equal to 0:

$$\frac{d}{dx}(20) = 0$$

$$\frac{d}{dx}(x^2) = 2x$$

$$\frac{d}{dx}(y^2) = 0 \quad (y \text{ is constant w.r.t. } x)$$

$$\frac{d}{dx}(-10\cos(2\pi x)) = -10 \cdot \frac{d}{dx} \cdot (\cos(2\pi x))$$

$$\frac{d}{dx}(-10\cos(2\pi y)) = 0$$

After this you apply the chain rule:

$$\frac{d}{dx}(\cos(2\pi x)) = -10 \cdot (-2\pi \sin(2\pi x)) = 20\pi \sin(2\pi x)$$

$$\frac{d}{dx}(-10\cos(2\pi x)) = -10 \cdot (-2\pi \sin(2\pi x)) = 20\pi \sin(2\pi x)$$

$$\frac{\partial f}{\partial x} = 2x + 20\pi \sin(2\pi x)$$

Now you also need to do the derivative of y: $\frac{\partial f}{\partial y} = \frac{d}{dy}\left(20 + x^2 + y^2 - 10\cos(2\pi x) - 10(2\pi y)\right)$

Differentiate each term:

$$\frac{d}{dy}(20) = 0$$

$$\frac{d}{dy}(x^2) = 0$$

$$\frac{d}{dy}(y^2) = 2y$$

$$\frac{d}{dy}(-10\cos(2\pi x)) = 0$$

Apply the chain rule again:

$$\frac{d}{dy}(\cos(2\pi y)) = -\sin(2\pi y)\cdot 2\pi = -2\pi\cdot\sin(2\pi y)$$

$$\frac{d}{dy}(-10\cos(2\pi y)) = 20\pi\sin(2\pi y)$$

$$\frac{\partial f}{\partial y} = 2y + 20\pi\sin(2\pi y)$$

Gradient of the Rastrigin function is:

$$\nabla f(x,y) = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} = \begin{bmatrix} 2x + 20\pi\sin(2\pi x) \\ 2y + 20\pi\sin(2\pi y) \end{bmatrix}$$

3.5. Computed analytical derivatives and numerical approximations at (x_0, y_0):

Definition of partial derivatives:
Let f(x,y) be a differentiable function. The partial derivatives of f w.r.t. x and y are defined as:

$$\frac{\partial f}{\partial x}(x,y) = \lim_{h\to 0}\frac{f(x+h,y)-f(x,y)}{h} \qquad \frac{\partial f}{\partial x}(x,y) = \lim_{h\to 0}\frac{f(x+h,y)-f(x,y)}{h}$$

Point (x_0, y_0): $(x_0, y_0) = (0.5849, 1.2312)$

Analytical derivatives at (x_0, y_0): $$\nabla f(x_0, y_0) = \begin{bmatrix} -30,7721 \\ 64,8562 \end{bmatrix}$$

Numerical approximations at (x_0, y_0): $$\nabla f(x_0, y_0) \approx \begin{bmatrix} -30,7721 \\ 64,8562 \end{bmatrix}$$

The numerical gradient closely matches the analytical gradient at the evaluated point. The results show very high agreement up to 4–5 decimal places.

3.6. Why is the Rastrigin function challenging to optimize?

From the contour plot and the optimization trajectory can be seen that the surface of the function is highly non-convex. It contains many local minima and maxima due to the periodic cosine terms. Also the optimization trajectory shows the algorithm struggling to converge smoothly and taking oscillatory paths as it descends.

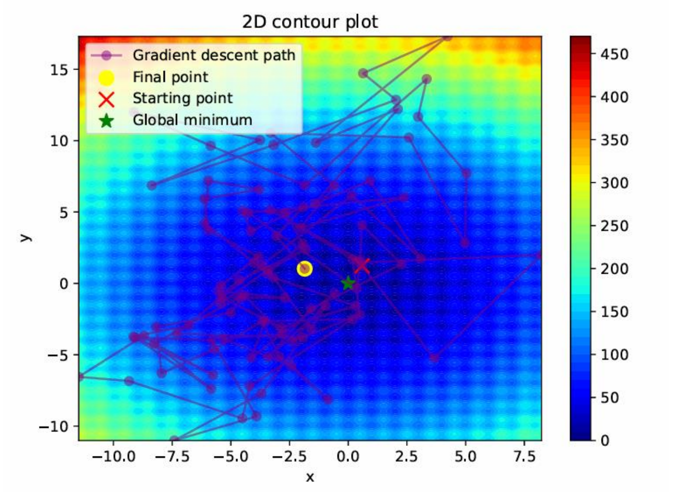
Rastrigin function is challenging to optimize because of the multiple local minima, because cosine terms introduce a large number of equally spaced "valleys" and "ridges". Gradient descent can easily get stuck in a local minimum that is far from the global minimum at (0,0). Another reason is the oscillatory gradient surface, where the gradients vary rapidly due to the sinusoidal components. This makes the step directions unstable and noisy, especially if the learning rate is not tuned carefully. Also the flat regions near minima make it challenging to optimize, because even when approaching the global minimum, the gradients become small. This can result in slow convergence and the risk of stopping prematurely.

3.7. Gradient descent minimization of the Rastrigin function:

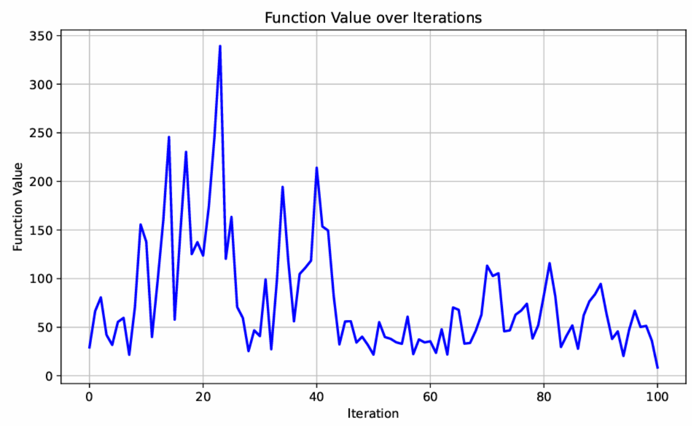
We apply the gradient descent algorithm using the true analytical gradient (not the finite difference approximation) to minimize the
Rastrigin function: $f(x,y)=20 + x^2 + y^2 - 10(\cos(2\pi x) + \cos(2\pi y))$
The following hyperparameters were used:
Initial point: $(x_0, y_0) = (0.5849, 1.2312)$
Learning rate (η): $\eta_0 = 0.01$
Learning rate decay: γ=0.99
Exponential decay: $\eta_t = \eta_0 \cdot \gamma^t = 0.01 \cdot 0{,}99^{1000}$
Number of iterations: $t = 1000$
<u>Results:</u>
Final point reached: $(x^*,y^*) \approx (5.8293, 4.8714)$
Function value at this point: $f(x^*,y^*) = 66.0243$
Global minimum: $f(0,0) = 0.0000$
Although the algorithm did not find the global minimum, it reached a good local minimum with relatively low function value.

3.8. Contour plot and gradient descent trajectory:

The trajectory reveals the non-convex nature of the Rastrigin function. The algorithm does not follow a direct path to the global minimum due to the multiple local minima and oscillations in the function landscape. The path includes several small, irregular steps (especially early on) where the algorithm adjusts to the steep gradients and periodic curvature of the function. Eventually, the updates settle into a local minimum, as seen by the convergence near a stable point on the plot.



3.9. Function Value Over Gradient Descent Iterations plot:



3.10: The Role of Learning Rate Decay in Optimizing the Rastrigin Function:

Setting the learning rate decay to 1.0 means: $\eta_t = \eta_0 \cdot (1{,}0)^t = \eta_0 \quad \text{for all } t$ that the learning rate stays constant throughout all iterations. There are many effects that happen such as:

- The optimization becomes unstable, especially in regions with steep or oscillatory gradients.
- The algorithm may overshoot minima, oscillate, or get trapped in poor local minima.
- The function value might fluctuate or stagnate, preventing smooth convergence.

Using a decaying learning rate helps because the large steps early on allow the optimizer to explore the space and avoid shallow local minima, while the smaller steps later help the optimizer to settle into precise local minima without overshooting or oscillating. It balances exploration and convergence, which enables escape from noise initially and fine-tuning near the end.