Deep Learning

Problem Set 1

Authors:
Aditi Joshi
A.Joshi@students.hertie-school.org

Laia Domenech Burin
L.Domenech-Burin@students.hertie-school.org

Lonny Chen
L.Chen@students.hertie-school.org

Franco Bastida
f.bastida@students.hertie-school.org

GitHub Repository:
assignment-1-ps-1-f

Sept/Oct 2025 – Deep Learning
Instructor: Dr. Lynn Kaack

1 Theoretical Part

1.1 Critical Points

Given Function

$$f(x, y, z) = x^2 + y^2 + z^2 + xyz$$

Finding Critical Points

To find critical points, compute the partial derivatives and set them equal to zero:

$$\frac{\partial f}{\partial x} = 2x - yz, \quad \frac{\partial f}{\partial y} = 2y - xz, \quad \frac{\partial f}{\partial z} = 2z - xy$$

Setting these equal to zero:

$$2x = yz \quad (1) \tag{1}$$
$$2y = xz \quad (2) \tag{2}$$
$$2z = xy \quad (3) \tag{3}$$

Multiplying equations (1), (2), and (3):

$$(2x)(2y)(2z) = (yz)(xz)(xy) \quad \Rightarrow \quad 8xyz = x^2y^2z^2$$

For non-zero solutions, this implies

$$xyz = 8.$$

From the system, one obtains:

$$z = \pm 2, \quad x = \pm 2, \quad y = \pm 2.$$

Checking which combinations satisfy $xyz = 8$ gives the following critical points:

$$(0, 0, 0), \quad (2, 2, 2), \quad (2, -2, -2), \quad (-2, 2, -2), \quad (-2, -2, 2).$$

Classification Using Hessian Matrix

The Hessian matrix is

$$H = \begin{bmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} & \frac{\partial^2 f}{\partial x \partial z} \\ \frac{\partial^2 f}{\partial y \partial x} & \frac{\partial^2 f}{\partial y^2} & \frac{\partial^2 f}{\partial y \partial z} \\ \frac{\partial^2 f}{\partial z \partial x} & \frac{\partial^2 f}{\partial z \partial y} & \frac{\partial^2 f}{\partial z^2} \end{bmatrix} = \begin{bmatrix} 2 & -z & -y \\ -z & 2 & -x \\ -y & -x & 2 \end{bmatrix}.$$

i) At $(0,0,0)$

$$H = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{bmatrix}$$

$$\det(H) = 8 > 0, \quad \text{Eigenvalues: } (2,2,2).$$

All eigenvalues positive $\Rightarrow$ **Local Minimum**.

ii) At $(2,2,2)$

$$H = \begin{bmatrix} 2 & -2 & -2 \\ -2 & 2 & -2 \\ -2 & -2 & 2 \end{bmatrix}$$

$$\begin{aligned} \det(H) &= -2\left(x^2 + xyz + y^2 + z^2 - 4\right) \\ &= -2\left(4 + 8 + 4 + 4 - 4\right) \\ &= -2(16) \\ &= -32 < 0 \end{aligned}$$

Mixed eigenvalues $\Rightarrow$ **Saddle Point**.

iii) At $(2,-2,-2), (-2,2,-2), (-2,-2,2)$

By symmetry, for each case,

$$\det(H) = -32 < 0 \quad \Rightarrow \quad \text{Saddle Points.}$$

Summary of Results

Critical Point	Type	$\det(H)$
$(0,0,0)$	Local Minimum	$+8$
$(2,2,2)$	Saddle Point	-32
$(2,-2,-2)$	Saddle Point	-32
$(-2,2,-2)$	Saddle Point	-32
$(-2,-2,2)$	Saddle Point	-32

1.2 Activation Function

Given Function

$$f(b, w) = \text{ReLU}(b + xw)$$

where the Rectified Linear Unit (ReLU) is defined as

$$\text{ReLU}(z) = \begin{cases} 0 & \text{if } z < 0, \\ z & \text{if } z \geq 0. \end{cases}$$

Therefore,

$$f(b, w) = \begin{cases} 0 & \text{if } b + xw < 0, \\ b + xw & \text{if } b + xw \geq 0. \end{cases}$$

Computing Partial Derivatives

(i) Derivative with respect to b

Let $z = b + xw$ and $\sigma = \text{ReLU}(z)$. Using the chain rule:

$$\frac{\partial f}{\partial b} = \frac{\partial \sigma}{\partial z} \cdot \frac{\partial z}{\partial b}.$$

$$\frac{\partial z}{\partial b} = 1, \qquad \frac{\partial \sigma}{\partial z} = \begin{cases} 0 & z < 0, \\ 1 & z \geq 0. \end{cases}$$

Hence,

$$\frac{\partial f}{\partial b} = \begin{cases} 0 & b + xw < 0, \\ 1 & b + xw \geq 0. \end{cases}$$

(ii) Derivative with respect to w

$$\frac{\partial f}{\partial w} = \frac{\partial \sigma}{\partial z} \cdot \frac{\partial z}{\partial w}.$$

$$\frac{\partial z}{\partial w} = x, \qquad \frac{\partial \sigma}{\partial z} = \begin{cases} 0 & z < 0, \\ 1 & z \geq 0. \end{cases}$$

Therefore,

$$\frac{\partial f}{\partial w} = \begin{cases} 0 & b + xw < 0, \\ x & b + xw \geq 0. \end{cases}$$

Thus,

$$\frac{\partial f}{\partial x} = \begin{cases} 0 & b + xw < 0, \\ w & b + xw \geq 0. \end{cases}$$

Gradient Vector

The gradient of f with respect to (b, w) is

$$\nabla f(b, w) = \begin{bmatrix} \frac{\partial f}{\partial b} \\ \frac{\partial f}{\partial w} \end{bmatrix}.$$

Hence,

$$\nabla f(b, w) = \begin{cases} \begin{bmatrix} 0 \\ 0 \end{bmatrix} & b + xw < 0, \\ \begin{bmatrix} 1 \\ x \end{bmatrix} & b + xw \geq 0. \end{cases}$$

Conclusion

- If $z < 0$ (inactive region), all gradients are zero $\Rightarrow$ no learning occurs.
- If $z \geq 0$ (active region), gradients are nonzero $\Rightarrow$ learning can happen through backpropagation.

1.3 Overfitting

1.3.1 Error rate versus dataset size

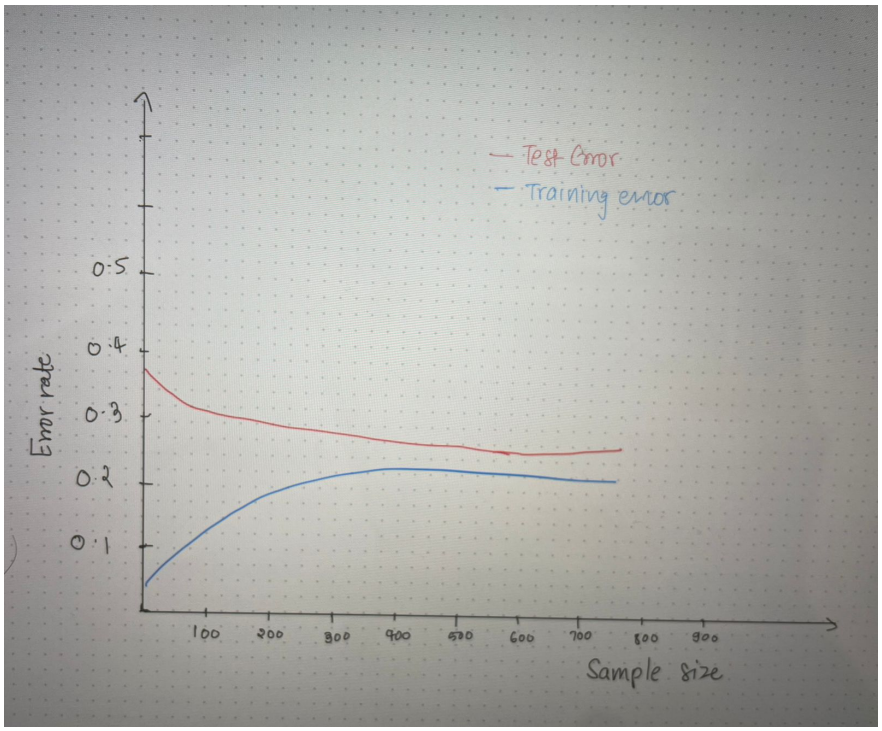

Figure 1: Training vs. Test Error Curve

The graph shows two curves plotted against:

- **x-axis:** Training set size (n)
- **y-axis:** Error rate (0 at bottom, increasing upwards)

Training Error (Blue Curve)

The training error follows a increasing pattern as the training set size grows. When the dataset is very small, the training error is close to zero because the model can easily memorize a few samples perfectly. As the number of training examples increases, the model—having fixed capacity—must account for more diverse data, which makes perfect fitting increasingly difficult. Consequently, the training error rises. Eventually, as the dataset becomes very large, the training error stabilizes and converges to a plateau. This plateau corresponds to the *approximation error* (or bias) of the model class, representing the inherent limitations of the chosen model architecture.

Test Error (Red Curve)

The test error displays a U-shaped pattern, which is central to understanding the bias–variance trade-off. For very small training sets, the test error is extremely high. In this regime, the model has high variance, tends to memorize noise, and generalizes poorly to unseen data. As the training set size increases, the test error decreases steadily. With more representative data, the model begins to learn the true underlying patterns, reducing variance and improving generalization. At very large training set sizes, the test error converges to a plateau, which is the sum of the squared bias and the irreducible error, $\text{Bias}^2 + \sigma^2$, where σ^2 represents the Bayes error or unavoidable noise in the data.

Summary

In summary, the relationship between training set size and error highlights three distinct regimes. For small datasets, training error is nearly zero while test error is very high, leading to severe overfitting and a wide generalization gap. With medium-sized datasets, both errors improve simultaneously, the gap narrows, and the model starts to generalize effectively. For large datasets, training error levels off at the approximation error (bias), while test error stabilizes at Bias^2+ irreducible noise, leaving only a minimal gap between the two. This progression underscores the critical role of dataset size in determining model performance and generalization ability.

1.3.2 Error rate versus model complexity

The graph shows two curves plotted against:

- **x-axis:** Model Complexity
- **y-axis:** Error rate (0 at bottom, increasing upwards)

The graph illustrates the fundamental relationship between **model complexity** (x-axis) and **error rates** (y-axis), revealing a critical trade-off in machine learning.

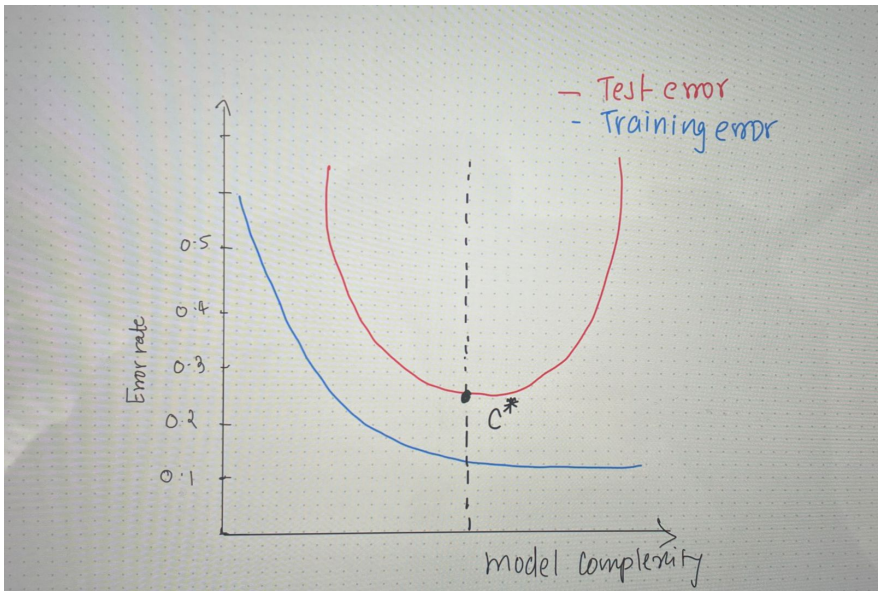

Figure 2: Training Error Rate vs.Model Complexity

Training Error (Blue Curve)

The training error decreases as model complexity increases. It starts relatively high when the model complexity is low, since a simple model cannot fit the data well. As complexity grows, the training error decreases continuously because the model gains more capacity to represent patterns in the data. With sufficiently high complexity, the model can even memorize every training example, driving the training error essentially to zero.

Test Error (Red Curve)

The test error follows a distinctive U-shaped pattern, which is the hallmark of the bias–variance trade-off. At low complexity, both training and test errors are high, reflecting underfitting where the model is too simple to capture the true structure of the data. As model complexity increases, the test error decreases in parallel with the training error, indicating improved learning and generalization. At an optimal complexity point, denoted C^* and marked by the black dashed vertical line, the test error reaches its minimum. Beyond this point, although the training error continues to decline, the test error begins to rise again. This is the regime of overfitting, where the model memorizes training-specific noise and loses its ability to generalize.

Summary

In summary, three distinct regions emerge from this relationship between model complexity and error. At low complexity, the model is too simple, leading to high bias and underfitting. At the optimal complexity point C^*, the model achieves the best balance between bias and variance, yielding maximum generalization. At high complexity, however, the model memorizes the training data, including noise, which results in high variance and overfitting.

1.3.3 Training Epochs

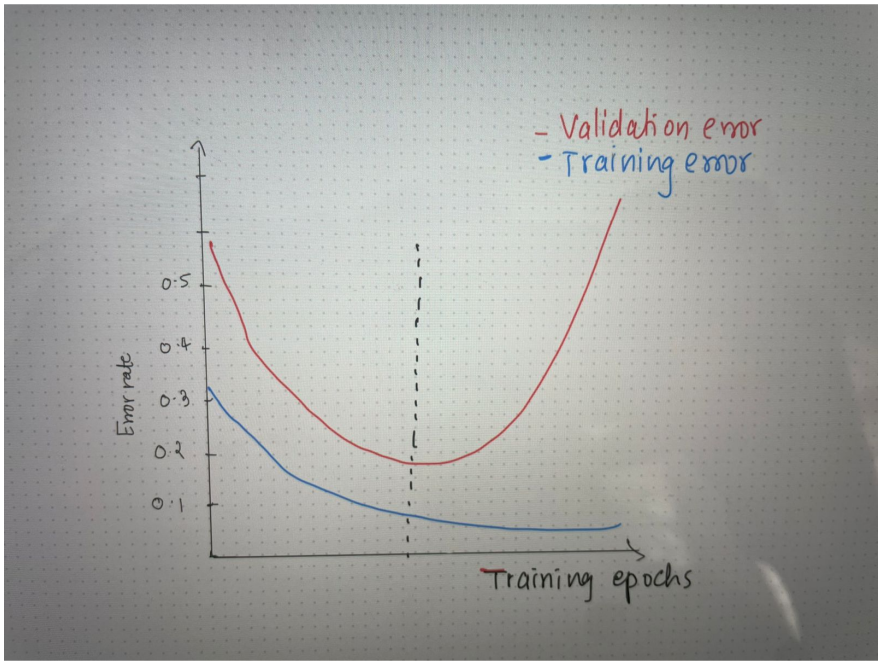

Figure 3: Training Error Rate vs. Training Epochs

The graph shows two curves plotted against:

- **x-axis:** Training Epochs
- **y-axis:** Error rate (0 at bottom, increasing upwards)

Training Error (Blue Curve)

The training error decreases throughout training. It starts at a relatively high value in the first epoch, drops quickly during the early epochs, and then continues to decrease steadily. With sufficient epochs and model capacity, the network eventually memorizes the training set completely, including noise, which drives the training error close to zero.

Validation Error (Red Curve)

The validation error follows a U-shaped trajectory, which reflects the bias–variance trade-off. In the early phase (assuming around epochs 1–35), the validation error decreases rapidly as the model learns meaningful and generalizable patterns; during this stage, both training and validation errors fall together, showing effective learning. Around epoch 35, the validation error reaches its minimum, representing the optimal point where bias is low and variance is still controlled. Beyond this point (assuming around epochs 35–100), the validation error begins to rise again even though the training error continues to fall. This indicates overfitting, where the model starts to memorize training-specific noise and loses its ability to generalize to unseen data.

Black Vertical Dashed Line (Epoch 35)

The black dashed vertical line at epoch 35 marks the optimal stopping point, where training should ideally end. At this point, the validation error is at its minimum, indicating the best balance between bias and variance. If training continues beyond this point, the training error decreases further, but the validation error worsens, leading to a widening generalization gap. This line illustrates the principle of early stopping, an implicit form of regularization that controls effective model complexity without requiring additional hyperparameter tuning.

Summary

In summary, the training error always decreases, but the validation error provides the true measure of generalization. More training is not always better, as performance in unseen data begins to decline after the optimal stopping point. Early stopping automatically identifies this "sweet spot" where the model has learned enough to generalize effectively without overfitting. This strategy is not only efficient but also interpretable and broadly applicable across neural network architectures.

1.4 Capacity of a neural network

The task defines a set of constraints to build a model that will fulfill them. Given that we have two features x_1, x_2, but the final model has to create a logistic model that has only the coefficient of the intercept term, we have to set weights that will produce a logistic form with an interaction term.

Each hidden unit computes

$$z^{[1]} = b^{[1]} + w_1 x_1 + w_2 x_2$$

We choose $g(z) = z^2$ as the hidden activation. Defining the first-layer pre-activations as

$$z_1 = x_1 + x_2, \qquad z_2 = x_1 - x_2,$$

which correspond to the weight matrix and bias vector

$$W^{[1]} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \qquad b^{[1]} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

Applying the quadratic activation yields

$$h_1 = g(z_1) = z_1^2 = (x_1 + x_2)^2, \qquad h_2 = g(z_2) = z_2^2 = (x_1 - x_2)^2.$$

Next, given the output pre-activation weights be

$$a = \beta_0 + \frac{\beta_1}{4} h_1 - \frac{\beta_1}{4} h_2.$$

Then

$$\frac{1}{4}(h_1 - h_2) = \frac{1}{4}\left[(x_1 + x_2)^2 - (x_1 - x_2)^2\right] = x_1 x_2.$$

Hence,

$$\boxed{a = \beta_0 + \beta_1 x_1 x_2}.$$

Finally, we apply a softmax activation at the output and it will return the desired model:

$$\text{logit}(p) = \beta_0 + \beta_1 x_1 x_2.$$

1.5 Neural network theory

To compute the the weight update for the weights of the first hidden layer $x_{ij}^{[1]}$, we need to compute the gradient descent of the weights with respect to the loss function.

$$x_{ij}^{[1](r+1)} = x_{ij}^{[1](r)} - n\frac{\mathrm{d}L}{\mathrm{d}w_{ij}^{[1](r)}}$$

The given loss function is:

$$L(f_k(\theta, X) = \frac{1}{2}\sum (o(a_k^{[3]} - \hat{y}_k)^2$$

With this, we apply chain rule to do backpropagation.

$$\frac{\mathrm{d}L}{\mathrm{d}w_{ij}^{[1]}} = \sum_{k=1}^{K} \frac{\mathrm{d}L}{\mathrm{d}o_k} \cdot \frac{\mathrm{d}a^{[3]}}{\mathrm{d}h_i^{[2]}} \cdot \frac{\mathrm{d}h_i^{[2]}}{\mathrm{d}a^{[2]}} \cdot \frac{\mathrm{d}a^{[2]}}{\mathrm{d}h_i^{[1]}} \cdot \frac{\mathrm{d}h_i^{[1]}}{\mathrm{d}a^{[1]}} \cdot \frac{\mathrm{d}a^{[1]}}{\mathrm{d}w_{ij}^{[1]}}$$

The next step is to take the derivative of each term.

$$\sum_{k=1}^{K} \frac{\mathrm{d}L}{\mathrm{d}o_k} = \sum_{k=1}^{K} \frac{\mathrm{d}L}{\mathrm{d}o_k} \cdot \frac{\mathrm{d}o_k}{\mathrm{d}a_m^{[3]}} = (o_k - y_k) \cdot o_k(1 - o_k)$$

$$\frac{\mathrm{d}a^{[3]}}{\mathrm{d}h_i^{[2]}} = \frac{d}{\mathrm{d}h_i^{[2]}}[b_i^{[3]} + \sum w_{kj}^{[3]} \cdot h_i^{[2]}] = w_{km}^{[2]}$$

$$\frac{\mathrm{d}h_i^{[2]}}{\mathrm{d}a^{[2]}} = \frac{d}{\mathrm{d}a^{[2]}}\mathrm{ReLU}'(a^{[2]}) = \mathbf{1}_{\{a^{[2]}>0\}}$$

$$\frac{\mathrm{d}a^{[2]}}{\mathrm{d}h_i^{[1]}} = \frac{d}{\mathrm{d}h_i^{[1]}}[b_i^{[1]} + \sum w_{ij}^{[1]} \cdot h_i^{[1]}] = w_j^{[1]}$$

$$\frac{\mathrm{d}h_i^{[1]}}{\mathrm{d}a^{[1]}} = \frac{d}{\mathrm{d}a^{[1]}}\mathrm{ReLU}'(a^{[1]}) = \mathbf{1}_{\{a^{[1]}>0\}}$$

$$\frac{\mathrm{d}a^{[1]}}{\mathrm{d}w_{ij}} = \frac{d}{\mathrm{d}w_{ij}}[b_i^{[1]} + \sum w_{ij}^{[1]} \cdot x_j] = x_j$$

So the final expression to identify the weight update is the following:

$$\frac{\mathrm{d}L}{\mathrm{d}w_{ij}^{[1]}} = (o_k - y_k) \cdot o_k(1 - o_k) \cdot w_{km}^{[2]} \cdot \mathbf{1}_{\{a^{[2]}>0\}} \cdot w_j^{[1]} \cdot \mathbf{1}_{\{a^{[1]}>0\}} \cdot x_j$$