TP#3: Perceptron as a Linear Regressor

Mathematical Formulation, Numerical Example, and Python Implementation

Dr. Samir Kenouche

November 1, 2025

1 Introduction

The perceptron is commonly used for classification, but it can also perform linear regression when the activation function is **linear** (identity function) and the objective is to minimize the Mean Squared Error (MSE). This report provides the mathematical formulation, a detailed numerical example, and Python code to demonstrate training a perceptron as a linear regressor.

2 Mathematical Formulation

A perceptron with n inputs can be represented as:

$$y = f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$$

where:

$$\mathbf{x} = [x_1, x_2, \ldots, x_n]^T, \quad \mathbf{w} = [w_1, w_2, \ldots, w_n]^T, \quad b \text{ is the bias.}$$

For regression, we use the identity activation function:

$$\hat{y} = \mathbf{w}^T\mathbf{x} + b$$

The Mean Squared Error (MSE) is defined as:

$$E(\mathbf{w}, b) = \frac{1}{2m}\sum_{i=1}^{m}(\hat{y}^{(i)} - y^{(i)})^2$$

where m is the number of samples.

The gradients of the loss with respect to the parameters are:

$$\frac{\partial E}{\partial w_j} = \frac{1}{m}\sum_{i=1}^{m}(\hat{y}^{(i)} - y^{(i)})x_j^{(i)}, \quad \frac{\partial E}{\partial b} = \frac{1}{m}\sum_{i=1}^{m}(\hat{y}^{(i)} - y^{(i)})$$

Then, parameters are updated iteratively using gradient descent:

$$w_j \leftarrow w_j - \eta\frac{\partial E}{\partial w_j}, \quad b \leftarrow b - \eta\frac{\partial E}{\partial b}$$

where η is the learning rate.

3 Diagram of the Linear Perceptron

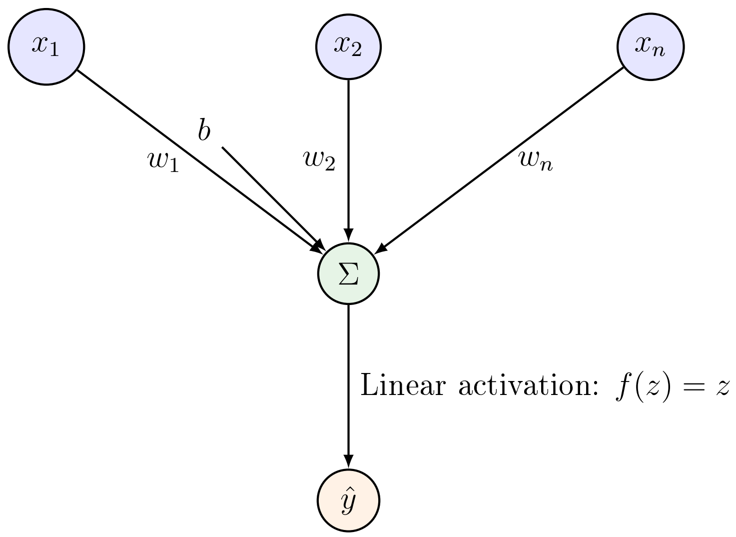

Figure 1: Linear Perceptron Architecture for Regression

4 Algorithm Summary

1. Initialize $\mathbf{w}$ and b (zeros or small random values).
2. For each iteration:
   - Compute predicted output $\hat{y}^{(i)} = \mathbf{w}^T\mathbf{x}^{(i)} + b$.
   - Compute the gradient of the loss function.
   - Update the weights and bias using gradient descent.
3. Stop when the loss converges or the maximum number of epochs is reached.

5 Detailed Numerical Example

We consider the following training data, which follows a perfect linear relation $y = 2x$:

i	x_i	y_i
1	1	2
2	2	4
3	3	6

We use:

$$\eta = 0.01, \quad w^{(0)} = 0, \quad b^{(0)} = 0$$

The model is:

$$\hat{y}_i = wx_i + b$$

Iteration 0

$$\hat{y} = [0, 0, 0], \quad e = [-2, -4, -6]$$
$$\frac{\partial E}{\partial w} = \frac{1}{3}(-2 \times 1 - 4 \times 2 - 6 \times 3) = -9.333, \quad \frac{\partial E}{\partial b} = \frac{-2 - 4 - 6}{3} = -4$$
$$w^{(1)} = 0 - 0.01(-9.333) = 0.0933, \quad b^{(1)} = 0 - 0.01(-4) = 0.04$$

Iteration 1

$$w = 0.0933, \quad b = 0.04$$
$$\hat{y} = [0.133, 0.227, 0.320], \quad e = [-1.867, -3.773, -5.680]$$
$$\frac{\partial E}{\partial w} = -8.818, \quad \frac{\partial E}{\partial b} = -3.773$$
$$w^{(2)} = 0.1815, \quad b^{(2)} = 0.0777$$

Iteration 2

$$w = 0.1815, \quad b = 0.0777$$
$$\hat{y} = [0.259, 0.440, 0.621], \quad e = [-1.741, -3.560, -5.379]$$
$$\frac{\partial E}{\partial w} = -8.333, \quad \frac{\partial E}{\partial b} = -3.560$$
$$w^{(3)} = 0.2648, \quad b^{(3)} = 0.1133$$

After about 1000 epochs:

$$w^{(1000)} \approx 1.9995, \quad b^{(1000)} \approx 0.001$$

and the predictions become nearly exact.

x	y_{true}	$\hat{y}$
1	2	2.00
2	4	4.00
3	6	6.00

6 Python Implementation

Listing 1: Perceptron as a Linear Regressor

```
import numpy as np
import matplotlib.pyplot as plt

# Samir KENOUCHE

# Training data
X = np.array([[1], [2], [3]])
y = np.array([[2], [4], [6]])

# Initialization
w, b = 0.0, 0.0
eta = 0.01
epochs = 1000
m = len(X)
losses = []

# Training loop
for epoch in range(epochs):
    y_pred = X * w + b
    error = y_pred - y
    loss = (1/(2*m)) * np.sum(error**2)
    losses.append(loss)

    dw = (1/m) * np.sum(error * X)
    db = (1/m) * np.sum(error)

    w = w - eta * dw
    b = b - eta * db

print(f"Final w = {w:.4f}, b = {b:.4f}")

# Plot fitted line
plt.figure(figsize=(6,4))
```

```
plt.scatter(X, y, color='blue', label='Training data')
plt.plot(X, X*w + b, color='red', label='Fitted line')
plt.xlabel("x"); plt.ylabel("y")
plt.legend(); plt.grid(True)
plt.title("Perceptron as Linear Regressor")
plt.show()

# Plot loss curve
plt.figure(figsize=(6,4))
plt.plot(losses)
plt.title("Convergence of Loss (MSE)")
plt.xlabel("Epoch"); plt.ylabel("Loss")
plt.grid(True)
plt.show()
```

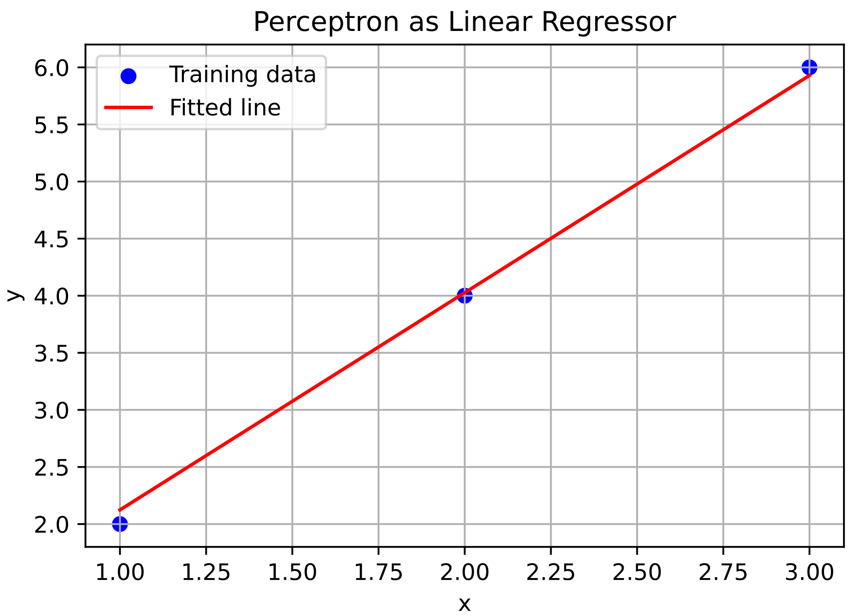

Figure 2: Perceptron as Linear Regressor

7 Conclusion

In this report, we demonstrated that the perceptron model can be used for linear regression by adopting a linear activation function and minimizing the mean squared error. Gradient descent successfully recovers the parameters of a linear model ($w \approx 2$, $b \approx 0$) when trained on simple data following $y = 2x$.

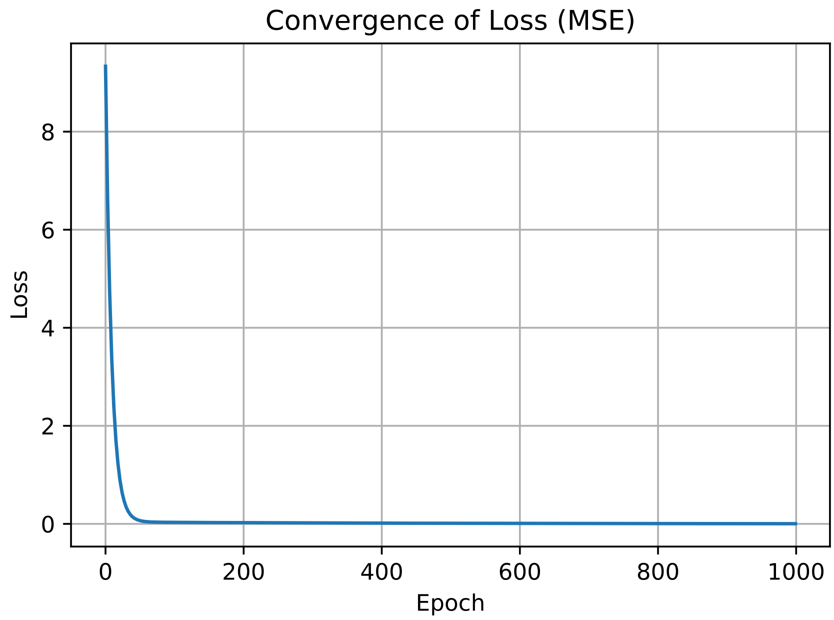

Figure 3: Convergence of Loss (MSE)

A Appendix. Mathematical Derivation of the Least Squares Parameters for a Linear Model

A.1 Model and Objective Function

Consider a set of n observed data points (x_i, y_i), $i = 1, 2, \ldots, n$. The simple linear regression model is written as

$$y_i = \beta_0 + \beta_1 x_i + \varepsilon_i, \tag{1}$$

where β_0 and β_1 are unknown parameters (intercept and slope, respectively), and ε_i represents the random error associated with each observation.

The least squares principle seeks to determine β_0 and β_1 that minimize the *sum of squared residuals*:

$$S(\beta_0, \beta_1) = \sum_{i=1}^{n} (y_i - \beta_0 - \beta_1 x_i)^2 . \tag{2}$$

A.2 First-Order Conditions

To find the minimum of S, we differentiate with respect to β_0 and β_1 and set the derivatives to zero.

Derivative with respect to β_0:

$$\begin{aligned}
\frac{\partial S}{\partial \beta_0} &= \sum_{i=1}^{n} 2\left(y_i - \beta_0 - \beta_1 x_i\right)(-1) \\
&= -2 \sum_{i=1}^{n} \left(y_i - \beta_0 - \beta_1 x_i\right). \qquad (3)
\end{aligned}$$

Setting this derivative to zero yields

$$\sum_{i=1}^{n} y_i - n\beta_0 - \beta_1 \sum_{i=1}^{n} x_i = 0. \qquad (4)$$

Derivative with respect to β_1:

$$\begin{aligned}
\frac{\partial S}{\partial \beta_1} &= \sum_{i=1}^{n} 2\left(y_i - \beta_0 - \beta_1 x_i\right)(-x_i) \\
&= -2 \sum_{i=1}^{n} x_i \left(y_i - \beta_0 - \beta_1 x_i\right). \qquad (5)
\end{aligned}$$

Setting this derivative to zero gives

$$\sum_{i=1}^{n} x_i y_i - \beta_0 \sum_{i=1}^{n} x_i - \beta_1 \sum_{i=1}^{n} x_i^2 = 0. \qquad (6)$$

Equations (4) and (6) are known as the **normal equations** of simple linear regression:

$$\boxed{\begin{aligned}
n\beta_0 + \beta_1 \sum x_i &= \sum y_i, \\
\beta_0 \sum x_i + \beta_1 \sum x_i^2 &= \sum x_i y_i.
\end{aligned}} \qquad (7)$$

A.3 Solution for the Parameters

From the first equation of (7),

$$\beta_0 = \frac{1}{n} \sum_{i=1}^{n} y_i - \frac{\beta_1}{n} \sum_{i=1}^{n} x_i. \qquad (8)$$

Introduce the sample means

$$\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i, \qquad \bar{y} = \frac{1}{n} \sum_{i=1}^{n} y_i, \qquad (9)$$

so that

$$\boxed{\beta_0 = \bar{y} - \beta_1 \bar{x}.} \qquad (10)$$

Substitute (10) into the second normal equation of (7):

$$\begin{aligned}(\bar{y} - \beta_1 \bar{x}) \sum x_i + \beta_1 \sum x_i^2 &= \sum x_i y_i, \\ n\bar{x}\bar{y} - n\beta_1 \bar{x}^2 + \beta_1 \sum x_i^2 &= \sum x_i y_i. \end{aligned} \tag{11}$$

Rearranging for β_1,

$$\beta_1 \left(\sum x_i^2 - n\bar{x}^2 \right) = \sum x_i y_i - n\bar{x}\bar{y}, \tag{12}$$

and thus

$$\boxed{\hat{\beta}_1 = \frac{\displaystyle\sum_{i=1}^{n} x_i y_i - n\bar{x}\bar{y}}{\displaystyle\sum_{i=1}^{n} x_i^2 - n\bar{x}^2}.} \tag{13}$$

A.4 Centered Variable Form

Note that

$$\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y}) = \sum x_i y_i - n\bar{x}\bar{y}, \quad \sum_{i=1}^{n} (x_i - \bar{x})^2 = \sum x_i^2 - n\bar{x}^2.$$

Hence Equation (13) can be rewritten compactly as

$$\boxed{\hat{\beta}_1 = \frac{\displaystyle\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{\displaystyle\sum_{i=1}^{n} (x_i - \bar{x})^2},} \tag{14}$$

and, substituting back into (10),

$$\boxed{\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x}.} \tag{15}$$

A.5 Verification of Minimization

The Hessian matrix of $S(\beta_0, \beta_1)$ is

$$H = 2 \begin{pmatrix} n & \sum x_i \\ \sum x_i & \sum x_i^2 \end{pmatrix}. \tag{16}$$

Since H is positive definite whenever $\sum (x_i - \bar{x})^2 > 0$, the stationary point found above corresponds to a *minimum* of the sum of squared errors.

A.6 Final Least Squares Estimates

The least squares estimates of the parameters in the linear model (1) are therefore:

$$\boxed{\begin{aligned}\hat{\beta}_1 &= \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^n (x_i - \bar{x})^2}, \\ \hat{\beta}_0 &= \bar{y} - \hat{\beta}_1 \bar{x}.\end{aligned}} \tag{17}$$

The fitted regression line is finally expressed as

$$\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_i, \tag{18}$$

and the residuals are

$$\hat{\varepsilon}_i = y_i - \hat{y}_i. \tag{19}$$

A.7 Interpretation

The parameter $\hat{\beta}_1$ represents the estimated change in y for a one-unit change in x, while $\hat{\beta}_0$ represents the fitted value of y when $x = 0$. Notably, the regression line always passes through the point $(\bar{x}, \bar{y})$.

Summary of the Least Squares Estimation Results

$$\textbf{Model:} \quad y_i = \beta_0 + \beta_1 x_i + \varepsilon_i$$

The least squares method minimizes

$$S(\beta_0, \beta_1) = \sum_{i=1}^n (y_i - \beta_0 - \beta_1 x_i)^2.$$

The resulting estimators are:

$$\hat{\beta}_1 = \frac{\displaystyle\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\displaystyle\sum_{i=1}^n (x_i - \bar{x})^2}, \tag{20}$$

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x}. \tag{21}$$

Hence, the fitted regression line is:

$$\boxed{\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_i}$$

and the residuals are:

$$\hat{\varepsilon}_i = y_i - \hat{y}_i.$$

Properties of the Least Squares Estimates

- The regression line passes through $(\bar{x}, \bar{y})$.
- Residuals are orthogonal to the fitted line:

$$\sum_i \hat{\varepsilon}_i = 0, \qquad \sum_i x_i \hat{\varepsilon}_i = 0.$$

- The slope $\hat{\beta}_1$ measures the average change in y for one unit change in x.