Mathematics For AI

Madiba Hudson-Quansah

Contents

Chapter 1

Foundations

The main structure of the machine learning part of AI is as follows:

1. Identify the problem
2. Acquire the appropriate data
3. Create a hypothesis/ learning/ prediction / training function
4. Find the numerical values of weights
5. Create an error function
6. Decide on mathematical formulas
7. Find a way to search for minimizers
8. Use the backpropagation algorithm
9. Regularize a function

1.1 Vector Notation

Definition 1.1.1: Vector

A vector is a mathematical object that has both magnitude and direction. It is represented by an arrow with a starting point and an end point. The length of the arrow represents the magnitude of the vector, and the direction of the arrow represents the direction of the vector. Denoted by $\vec{v}$ or $\mathbf{v}$ which in vector space $\mathbb{R}^n$ represents:

$$\mathbf{v} = \begin{pmatrix} v_1 \\ \vdots \\ v_n \end{pmatrix}$$

The transpose of a column vector $\mathbf{v}$ is denoted by $\mathbf{v}^t$ and is a row vector:

$$\mathbf{v}^t = \begin{pmatrix} v_1 & \dots & v_n \end{pmatrix}$$

Definition 1.1.2: Dot Product

The dot product of two vectors $\mathbf{v}$ and $\mathbf{w}$ is a scalar value denoted by $\mathbf{v} \cdot \mathbf{w}$ and is defined as:

$$\mathbf{v} \cdot \mathbf{w} = \sum_{i=1}^{n} v_i w_i$$

The dot product is the same as the product of a row vector / transposed column vector and a column vector, for example for vectors $\mathbf{a} = \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \end{pmatrix}$ and $\mathbf{b} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \end{pmatrix}$, the dot product is:

$$\begin{aligned} \mathbf{a} \cdot \mathbf{b} &= \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \end{pmatrix} \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \end{pmatrix} \\ &= a_1 b_1 + a_2 b_2 + a_3 b_3 + a_4 b_4 + a_5 b_5 \end{aligned}$$

Or

$$\begin{aligned} \mathbf{a}^t \mathbf{b} &= \begin{pmatrix} a_1 & a_2 & a_3 & a_4 & a_5 \end{pmatrix} \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \end{pmatrix} \\ &= a_1 b_1 + a_2 b_2 + a_3 b_3 + a_4 b_4 + a_5 b_5 \end{aligned}$$

Moreover:

$$\| \mathbf{a} \|_{l^2}^2 = \mathbf{a}^t \mathbf{a} = a_1^2 + a_2^2 + a_3^2 + a_4^2 + a_5^2$$

1.2 Numerical vs Analytical Solutions

Definition 1.2.1: Numerical

Has to do with numbers. Are approximations of analytical solutions obtained by discretizing a continuous problem.

Definition 1.2.2: Analytical

Has to do with analysis. Are exact solutions to problems. Not all problems have analytical solutions.

1.3 Multivariable Calculus

In multivariable calculus, the gradient of a function is a vector that points in the direction of the greatest rate of increase of the function at a given point. The gradient is denoted by ∇ and is defined as:

$$\nabla F = \begin{pmatrix} \frac{\partial F}{\partial x_1} \\ \frac{\partial F}{\partial x_2} \\ \vdots \\ \frac{\partial F}{\partial x_n} \end{pmatrix}$$

1.3.1 Common derivatives of linear algebra

- When a and ω and a is a constant, the derivate of $f(\omega) = a\omega$ is

$$f'(\omega) = a$$

Therefore when $\mathbf{a}$ and $\boldsymbol{\omega}$ and the entries of $\mathbf{a}$ are constant, then the derivate / gradient of $f(\boldsymbol{\omega}) = \mathbf{a}^t\boldsymbol{\omega}$ is:

$$\nabla f(\boldsymbol{\omega}) = \mathbf{a}$$

And similarly the derivate of $f(\boldsymbol{\omega}) = \boldsymbol{\omega}^t\mathbf{a}$ is

$$\nabla f(\boldsymbol{\omega}) = \mathbf{a}$$

- When s is constant and ω is scalars, then the derivate of the quadratic function $f(\omega) = s\omega^2$ is :

$$f'(\omega) = 2s\omega$$

Therefore when S is a symmetric matrix with constant entries, then the derivative of the function $f(\boldsymbol{\omega}) = \boldsymbol{\omega}^t S\boldsymbol{\omega}$ is:

$$\nabla f(\boldsymbol{\omega}) = 2S\boldsymbol{\omega}$$

Because $\boldsymbol{\omega}^t\boldsymbol{\omega}$, if $\boldsymbol{\omega} = \begin{bmatrix}\omega_0\\ \omega_1\\ \omega_2\end{bmatrix}$, then $\boldsymbol{\omega}^t = \begin{bmatrix}\omega_0 & \omega_1 & \omega_2\end{bmatrix}$ is defined as:

$$\boldsymbol{\omega}^t\boldsymbol{\omega} = \begin{bmatrix}\omega_0 & \omega_1 & \omega_2\end{bmatrix}\begin{bmatrix}\omega_0\\ \omega_1\\ \omega_2\end{bmatrix}$$

1.4 Gradient Descent

Definition 1.4.1: Gradient

The gradient of a function is a vector that points in the direction of the greatest rate of increase of the function at a given point.

Definition 1.4.2: Gradient Descent

An optimization algorithm used to minimize some function by iteratively moving in the direction of steepest descent as defined by the negative of the gradient.

This algorithm is used in the optimization of training functions using a loss function. Gradient descent is carried out as follows:

1. Choose a random starting point within the search space
2. Calculate the gradient of the function at that point
3. Move in the direction of the negative of the gradient by subtracting the gradient from the current point
4. Repeat steps 2 and 3 until the gradient is close to zero

As denoted by the following formula:

$$\boldsymbol{\omega}_{n+1} = \boldsymbol{\omega}_n - \alpha\nabla F(\boldsymbol{\omega}_\mathbf{n})$$

Where $\boldsymbol{\omega}_\mathbf{n+1}$ is the new point, $\boldsymbol{\omega}_\mathbf{n}$ is the current point, α is the learning rate, and $\nabla F(\boldsymbol{\omega}_\mathbf{n})$ is the gradient of the function at the current point.

Definition 1.4.3: Learning Rate

A hyperparameter that controls how much the weights are updated during training. A large learning rate means the weights are updated by a large amount, while a small learning rate means the weights are updated by a small amount.

1.5 Regression

Definition 1.5.1: Regression

A statistical method used to determine the strength and character of the relationship between one dependent variable and one or more independent variables.

1.6 Classification

Definition 1.6.1: Classification

A supervised learning approach that categorizes input data into one of a number of classes.

Chapter 2

Linear Models

2.1 Linear Regression

A linear regression model is a linear approach to modelling the relationship between a dependent variable and one or more independent variables. In using a linear model we make the assumption that the dependent variable / predicted value depends linearly on the independent variables / features.

2.1.1 Simple Linear Regression

In the case of simple linear regression, there is only one independent variable. This relationship can be expressed as:

$$y = \alpha + \beta x$$

Where y is the predicted value / dependent variable, α is the intercept, β is the slope, and x is the feature / independent variable.

2.1.1.1 Training Function

When using linear models the assumption of linear dependence is made. This means that the predicted value found using a line of best fit derived from the data, giving the training function as:

$$y = \beta_0 + \beta_1 x$$

Where β_0 is the y-intercept, β_1 is the slope of the line, y is the predicted value and x is the feature under consideration.

2.1.1.2 Loss Function

For linear regression models the most common loss function is the mean squared error function. Defined as:

$$\text{Mean Squared Error} = \frac{1}{m} \sum_{i=1}^{m} | \ y_{\text{predict}} - y_{\text{true}} \ |^2$$

Where m is the number of rows in the sample data, y_{predict} is the predicted value, and y_{true} is the actual value. The mean squared error function can also be expressed as:

$$\text{Mean Squared Error} = \frac{1}{m} \left(\mathbf{y}_{\text{predicted}} - \mathbf{y}_{\text{true}} \right)^t \left(\mathbf{y}_{\text{predict}} - \mathbf{y}_{\text{true}} \right)$$

Because the predicted and true values for all the rows in the sample data can be represented as vectors. And using $\mathbf{y}_{\text{predict}} = \mathbf{x}\boldsymbol{\beta}$, (where $\boldsymbol{\beta} = \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix}$ and $\mathbf{x} = \begin{bmatrix} 1 & x_1 \\ \vdots & \vdots \\ 1 & x_m \end{bmatrix}$) can then be expressed as:

$$\text{Mean Squared Error} = \frac{1}{m} \left(\mathbf{x}\boldsymbol{\beta} - \mathbf{y}_{\text{true}} \right)^t \left(\mathbf{x}\boldsymbol{\beta} - \mathbf{y}_{\text{true}} \right)$$

2.1.1.3 Optimization

In minimizing the mean squared error function the weights of the model, α and β, can be found using the minimizing the derivative of loss function with respect to the weights and equating it to zero. Therefore for the loss function:

$$\text{Mean Squared Error} = \frac{1}{m}\left(\mathbf{x}\boldsymbol{\beta} - \mathbf{y}_{\text{true}}\right)^t\left(\mathbf{x}\boldsymbol{\beta} - \mathbf{y}_{\text{true}}\right)$$

Let $L\left(\boldsymbol{\beta}\right)$ be:

$$L\left(\boldsymbol{\beta}\right) = \frac{1}{m}\left(\mathbf{x}\boldsymbol{\beta} - \mathbf{y}_{\text{true}}\right)^t\left(\mathbf{x}\boldsymbol{\beta} - \mathbf{y}_{\text{true}}\right)$$

Simplified:

$$\begin{aligned}
L\left(\boldsymbol{\beta}\right) &= \frac{1}{m}\left(\mathbf{x}\boldsymbol{\beta} - \mathbf{y}_{\text{true}}\right)^t\left(\mathbf{x}\boldsymbol{\beta} - \mathbf{y}_{\text{true}}\right) \\
&= \frac{1}{m}\left(\left(\mathbf{x}\boldsymbol{\beta}\right)^t - \mathbf{y}_{\text{true}}^t\right)\left(\mathbf{x}\boldsymbol{\beta} - \mathbf{y}_{\text{true}}\right) \\
&= \frac{1}{m}\left(\mathbf{x}^t\boldsymbol{\beta}^t - \mathbf{y}_{\text{true}}^t\right)\left(\mathbf{x}\boldsymbol{\beta} - \mathbf{y}_{\text{true}}\right) \\
&= \frac{1}{m}\left(\mathbf{x}^t\mathbf{x}\boldsymbol{\beta}^t\boldsymbol{\beta} - \mathbf{y}_{\text{true}}^t\mathbf{x}\boldsymbol{\beta} - \mathbf{y}_{\text{true}}\mathbf{x}^t\boldsymbol{\beta}^t + \mathbf{y}_{\text{true}}^t\mathbf{y}_{\text{true}}\right)
\end{aligned}$$

Notice that $\left(\mathbf{y}_{\text{true}}^t\mathbf{x}\boldsymbol{\beta}\right)^t = \mathbf{y}_{\text{true}}\mathbf{x}^t\boldsymbol{\beta}^t$, and since this is a 1×1 matrix, a scalar, $\mathbf{y}_{\text{true}}^t\mathbf{x}\boldsymbol{\beta} = \mathbf{y}_{\text{true}}\mathbf{x}^t\boldsymbol{\beta}^t$ making the last expression:

$$= \frac{1}{m}\left(\mathbf{x}^t\mathbf{x}\boldsymbol{\beta}^t\boldsymbol{\beta} - 2\mathbf{y}_{\text{true}}\mathbf{x}^t\boldsymbol{\beta}^t + \mathbf{y}_{\text{true}}^t\mathbf{y}_{\text{true}}\right)$$

Then we take the derivate of the last expression with respect to $\boldsymbol{\beta}$ using the common derivatives 1.3.1.

$$\begin{aligned}
L\left(\boldsymbol{\beta}\right) &= \frac{1}{m}\left(\mathbf{x}^t\mathbf{x}\boldsymbol{\beta}^t\boldsymbol{\beta} - 2\mathbf{y}_{\text{true}}\mathbf{x}^t\boldsymbol{\beta}^t + \mathbf{y}_{\text{true}}^t\mathbf{y}_{\text{true}}\right) \\
\text{Let } S = \mathbf{x}^t\mathbf{x} \text{ and let } \mathbf{a} &= \mathbf{x}^t\mathbf{y}_{\text{true}} \\
L\left(\boldsymbol{\beta}\right) &= \frac{1}{m}\left(S\boldsymbol{\beta}^t\boldsymbol{\beta} - 2\mathbf{a}\boldsymbol{\beta}^t + \mathbf{y}_{\text{true}}^t\mathbf{y}_{\text{true}}\right) \\
\nabla L\left(\boldsymbol{\beta}\right) &= \frac{1}{m}\left(2S\boldsymbol{\beta} - 2\mathbf{a} + 0\right) \\
\nabla L\left(\boldsymbol{\beta}\right) &= \frac{2}{m}\left(S\boldsymbol{\beta} - \mathbf{a}\right) \\
\nabla L\left(\boldsymbol{\beta}\right) &= \frac{2}{m}\left(\mathbf{x}^t\mathbf{x}\boldsymbol{\beta} - \mathbf{x}^t\mathbf{y}_{\text{true}}\right)
\end{aligned}$$

At this point we have the derivate of the loss function $L\left(\boldsymbol{\beta}\right)$ and can either equate it to zero or use gradient descent 1.4 to find the optimal weights $\boldsymbol{\beta}$. If we equate the derivative to zero:

$$\begin{aligned}
\nabla L\left(\boldsymbol{\beta}\right) &= \frac{2}{m}\left(\mathbf{x}^t\mathbf{x}\boldsymbol{\beta} - \mathbf{x}^t\mathbf{y}_{\text{true}}\right) \\
0 &= \frac{2}{m}\left(\mathbf{x}^t\mathbf{x}\boldsymbol{\beta} - \mathbf{x}^t\mathbf{y}_{\text{true}}\right) \\
&\text{As } \frac{2}{m} \text{ is a constant, we can remove it} \\
0 &= \mathbf{x}^t\mathbf{x}\boldsymbol{\beta} - \mathbf{x}^t\mathbf{y}_{\text{true}} \\
\mathbf{x}^t\mathbf{x}\boldsymbol{\beta} &= \mathbf{x}^t\mathbf{y}_{\text{true}} \\
\boldsymbol{\beta} &= \left(\mathbf{x}^t\mathbf{x}\right)^{-1}\mathbf{x}^t\mathbf{y}_{\text{true}}
\end{aligned}$$

2.1.2 Multi-Linear Regression

In the case of multi-linear regression, there are multiple independent variables. This relationship can be expressed as:

$$y = f(x_1, x_2, x_3, \ldots, x_n)$$

Where y is the predicted value, $x_1, x_2, x_3, \ldots, x_n$ are the features, and f is the function that maps the features to the predicted value.

2.1.2.1 Training Function

As with all linear models the assumption of linear dependence is made. This means that the predicted value is a linear combination of its features plus a bias term ω_0, giving the following training function:

$$y = \omega_0 + \omega_1 x_1 + \omega_2 x_2 + \ldots + \omega_n x_n$$

Where y is the predicted value, ω_0 is the bias term, $\omega_1, \omega_2, \ldots, \omega_n$ are the weights, and $x_1, x_2, \ldots, x_n$ are the features. In order to find the appropriate weights ω_n, they have to be learnt from the training data. This process is called training the model. Ergo a trained model is model that has decided on the appropriate weights to use.

Each weight ω is a scalar value multiple by a corresponding feature x, and represent the importance of a specific feature in determining the predicted value. This means the larger the weight the more important the feature is in determining the predicted value, and vice versa. *Dead features* are features that have a weight of zero and do not affect the predicted value. This means for each row in the total amount of rows m the prediction of this model is:

$$\begin{aligned}
y^1_{\text{predict}} &= \omega_0 + \omega_1 x^1_1 + \omega_2 x^1_2 + \ldots + \omega_n x^1_n \\
y^2_{\text{predict}} &= \omega_0 + \omega_1 x^2_1 + \omega_2 x^2_2 + \ldots + \omega_n x^2_n \\
&\vdots \\
y^m_{\text{predict}} &= \omega_0 + \omega_1 x^m_1 + \omega_2 x^m_2 + \ldots + \omega_n x^m_n
\end{aligned}$$

Which can be expressed as:

$$\begin{pmatrix} y^1_{\text{predict}} \\ y^2_{\text{predict}} \\ \vdots \\ y^m_{\text{predict}} \end{pmatrix} = \omega_0 \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix} + \omega_1 \begin{pmatrix} x^1_1 \\ x^2_2 \\ \vdots \\ x^m_n \end{pmatrix} + \ldots + \omega_n \begin{pmatrix} x^1_1 \\ x^2_2 \\ \vdots \\ x^m_n \end{pmatrix}$$

$$\begin{pmatrix} y^1_{\text{predict}} \\ y^2_{\text{predict}} \\ \vdots \\ y^m_{\text{predict}} \end{pmatrix} = \begin{pmatrix} 1 & x^1_1 & x^1_2 & \ldots & x^1_n \\ 1 & x^2_1 & x^2_2 & \ldots & x^2_n \\ \vdots & & & & \\ 1 & x^m_1 & x^m_2 & \ldots & x^m_n \end{pmatrix} \begin{pmatrix} \omega_0 \\ \omega_1 \\ \omega_2 \\ \vdots \\ \omega_n \end{pmatrix}$$

This can be then expressed as

$$\mathbf{y}_{\text{predict}} = X\boldsymbol{\omega}$$

Where $\mathbf{y}_{predict}$ is the vector of predictions X is the training subset, and $\boldsymbol{\omega}$ is the vector of unknown weights.

2.1.2.1.1 Parametric Models vs Non-Parametric Models

Definition 2.1.1: Parametric Model

A model that has parameters that are fixed in number, regardless of the size of the training data. For example:

$$y = \omega_0 + \omega_1 x_1 + \omega_2 x_2 + \ldots + \omega_n x_n$$

The number of parameters (ω) is fixed at $n + 1$. This means that the formula of training function is fixed ahead of training the model, and the model is trained to find the appropriate values of the parameters.

Definition 2.1.2: Non-Parametric Model

A model that does not specify the formula for the training function with it's parameters before training the model. This results in the number of parameters being dependent on the size of the training data, as such models adapt to the data and determines the required number of parameters during training.

Both parametric and non-parametric models have other parameters that are not weights called hyperparameters, that need to be tuned during training.

Definition 2.1.3: Hyperparameters

Parameters that are not weights, and are not learnt during training. They are set before training the model, and determine the behaviour of the model.

2.1.2.2 Loss Function

To determine the weights of the model using the training data we need to define and optimize a loss function.

Definition 2.1.4: Loss Function

Determines the error of the model, by measuring the difference between the predicted values generated by the model and the actual values in the training data.

For example assuming we trained a model with n features, giving us the weights $\omega_0, \omega_1, \ldots, \omega_n$, then the ith predicted values using the ith row of the training data would be:

$$y^i_{\text{predict}} = \omega_0 + \omega_1 x^i_1 + \omega_2 x^i_2 + \ldots + \omega_n x^i_n$$

However for the ith row of the training data the actual value is y^i_{true}. In determining the error between the predicted value and the actual value we can use the following approaches.

Absolute Value Distance - $\mid y_{\text{predict}} - y_{\text{true}} \mid$ derived from $\mid x \mid$

Squared Distance - $\mid y_{\text{predict}} - y_{\text{true}} \mid^2$ derived from $\mid x \mid^2$

In examining both functions $\mid x \mid$ and $\mid x \mid^2$ we notice that the squared distance function is smoother than the absolute value distance function. This means that the squared distance function is differentiable at all points, while the absolute value distance function is not differentiable at $x = 0$, this is called a *singularity*.

Other than the difference in *regularity* of both functions another consideration must be made, *if a number is large, then its square is even larger.* This means that the squared distance function is more sensitive to outliers in the data than the absolute value distance function.

Definition 2.1.5: Regularity

The property of a function that determines how smooth it is. A function is regular if it is differentiable at all points.

2.1.2.2.1 Functions with Singularities

Definition 2.1.6: Singularity

A point at which a function is not differentiable.

Generally graphs of differential functions do not have sharp corners, cusps, or vertical tangents. This is because at that point you can draw two different tangents lines to the graph of the function at that point giving two different slopes at that point. This discontinuity in the slope of the tangent creates a problem for methods that rely on evaluating the derivative of the function such as the gradient descent algorithm.

2.1.2.2.2 Mean Squared Error

For linear regression models the most common loss function is the mean squared error function.

Definition 2.1.7: Mean Squared Error

The average of the squared differences between the predicted values and the actual values in the training data. Defined as:

$$\text{Mean Squared Error} = \frac{1}{m}\sum_{i=1}^{m} \mid y^i_{\text{predict}} - y^i_{\text{true}} \mid^2$$

Where m is the number of rows. And in linear algebra notation:

$$\text{Mean Squared Error} = \frac{1}{m}\left(\vec{y}_{\text{predict}} - \vec{y}_{\text{true}}\right)^t \left(\vec{y}_{\text{predict}} - \vec{y}_{\text{true}}\right) = \frac{1}{m} \parallel \vec{y}_{\text{predict}} - \vec{y}_{\text{true}} \parallel^2_{l^2}$$

Where l^2 denotes the l^2 norm of the vector which by definition is the $\sqrt{\text{sum of squares of its components}}$.
And using $\mathbf{y}_{\text{predict}} = X\boldsymbol{\omega}$

$$\text{Mean Squared Error} = \frac{1}{m}\left(X\boldsymbol{\omega} - \mathbf{y}_{\text{true}}\right)^t \left(X\boldsymbol{\omega} - \mathbf{y}_{\text{true}}\right) = \frac{1}{m} \parallel X\boldsymbol{\omega} - \mathbf{y}_{\text{true}} \parallel^2_{l^2}$$

2.1.2.3 Optimization

Definition 2.1.8: Optimization

Finding the best / optimal / maximal / minimal / extreme solution.

Given our linear training function:

$$y = \omega_0 + \omega_1 x_1 + \omega_2 x_2 + \ldots + \omega_n x_n$$

The goal of the optimization step is to find the values of $\omega_0, \omega_1, \omega_2, \ldots, \omega_n$, that make our training function best fit the training data, where an optimal fit is charactered by the loss function. Therefore we need the weights that minimize the value of our loss function, denoted by:

$$\min_{\boldsymbol{\omega}} \text{ loss function}$$

For our loss function, the mean squared error is:

$$\min_{\boldsymbol{\omega}} \frac{1}{m} \parallel X\boldsymbol{\omega} - \mathbf{y}_{\text{true}} \parallel^2_{l^2}$$

2.1.2.3.1 Convex Landscapes vs Non convex Landscapes

Definition 2.1.9: Convex Function

Let f be a function that is differentiable over an interval I. The function f is convex if the first derivative of the function f' is increasing over I, which means the second derivate of the function, f'' is greater than zero over the interval I, $f'' > 0$

Definition 2.1.10: Non Convex / Concave Function

Let f be a function that is differentiable over an interval I. The function f is concave if f' is decreasing over I, which means f'' is less than zero over the interval I, $f'' < 0$

Most functions and equations we will deal with are non-linear, but can be linearised at certain points of interest. This is done by drawing a tangent line to the point in 2D space or a tangent hyperplane in greater dimensions. To do this the derivative of the function must be found with respect to all of its variables to determine the slope of the approximating flat space.

One important type of function is a function that is the maximum of two or more convex functions, which are always convex. Linear functions are flat so they are both convex and concave. This is important because some functions are defined as the maxima of linear functions, which are not guaranteed to be linear, but are guaranteed to be convex. That is even if we lose linearity when taking the maximum of linear functions we still have convexity.

The Rectified Linear Unit function (ReLU), is an example of a non-linear function defined as the maximum of two linear functions:

$$ReLU\,(x) = max\,(0, x)$$

There is one more relationship between linearity and convexity. If we have a non-linear convex function, the maximum of all the linear functions that stay below the function is exactly equal to it.

Overall the landscape of a convex function is good for minimization problems as there is no chance at getting stuck at a local minima as any local minimum is also a global minimum for a convex function. However a non convex function could have peaks, valleys and saddle points, which runs the risk of getting stuck at a local minima, and never finding the global minima.

2.1.2.3.2 Locating Function Minimizers

There are generally two approaches to locating minimizers/maximizers of functions with the trade-off usually being between:

- Calculating only one derivative and converging to the minimum slowly. These are called *gradient* methods, where the gradient is one derivate of a function of several variables as is our case with $\omega_0, \ldots, \omega_n$.
- Calculating two derivatives and converging to the minimum faster. These are called *Newton's* methods, and the *Hessian* or an approximation of the Hessian is used in these methods

Definition 2.1.11: Hessian Matrix

A square matrix consisting of second derivates of a function

One key idea from calculus remains fundamental: minimizers / maximizers occur at critical points. So to locate these minimizers / maximizers we must search through both the boundary points (points where the function is undefined) and interior critical points (where the first derivative of the function is equal to 0). This can be done using one of two approaches

Approach 1

1. Find the derivate of the function
2. Equate the derivative to 0
3. Solve for the ω's that make the derivative 0.

This approach works when the derivate is linear and the ωs can easily be found but is not ideal when the derivate is non linear.

Approach 2 Follow the gradient direction to descend towards the minimum or ascend towards the maximum, using gradient descent 1.4.

2.1.2.3.3 Minimizing the mean squared error function

Let $L(\boldsymbol{\omega})$ be

$$\frac{1}{m}\left(X\boldsymbol{\omega}-\mathbf{y}_{\text{true}}\right)^{t}\left(X\boldsymbol{\omega}-\mathbf{y}_{\text{true}}\right)$$

Simplified:

$$\begin{aligned}
L(\boldsymbol{\omega}) &= \frac{1}{m}\left(X\boldsymbol{\omega}-\mathbf{y}_{\text{true}}\right)^{t}\left(X\boldsymbol{\omega}-\mathbf{y}_{\text{true}}\right)\\
&= \frac{1}{m}\left(\left(X\boldsymbol{\omega}\right)^{t}-\mathbf{y}_{\text{true}}\right)\left(X\boldsymbol{\omega}-\mathbf{y}_{\text{true}}\right)\\
&= \frac{1}{m}\left(\boldsymbol{\omega}^{t}\mathbf{X}^{t}-\mathbf{y}_{\text{true}}\right)\left(X\boldsymbol{\omega}-\mathbf{y}_{\text{true}}\right)\\
&= \frac{1}{m}\left(\boldsymbol{\omega}^{t}X^{t}X\boldsymbol{\omega}-\boldsymbol{\omega}^{t}X^{t}\mathbf{y}_{\text{true}}^{t}-\mathbf{y}^{t}X\boldsymbol{\omega}+\mathbf{y}_{\text{true}}^{t}\mathbf{y}_{\text{true}}\right)\\
&= \frac{1}{m}\left(\boldsymbol{\omega}^{t}S\boldsymbol{\omega}-\boldsymbol{\omega}^{t}\mathbf{a}-\mathbf{a}^{t}\boldsymbol{\omega}+\mathbf{y}_{\text{true}}^{t}\mathbf{y}_{\text{true}}\right)
\end{aligned}$$

Next we take the gradient of the last expression with respect to $\boldsymbol{\omega}$ and equate it to zero.

$$\begin{aligned}
\nabla L(\boldsymbol{\omega}) &= \frac{1}{m}\left(2S\boldsymbol{\omega}-\mathbf{a}-\mathbf{a}+0\right)\\
\nabla L(\boldsymbol{\omega}) &= 0
\end{aligned}$$

$$\begin{aligned}
\frac{1}{m}\left(2S\boldsymbol{\omega}-\mathbf{a}-\mathbf{a}+0\right) &= 0\\
2S\boldsymbol{\omega}-2\mathbf{a} &= 0\\
2S\boldsymbol{\omega} &= 2\mathbf{a}\\
\boldsymbol{\omega} &= \frac{\mathbf{a}}{S}\\
\boldsymbol{\omega} = S^{-1}\mathbf{a}\boldsymbol{\omega} &= \left(X^{t}X\right))^{-1}X^{t}\mathbf{y}_{\text{true}}
\end{aligned}$$

At this $\boldsymbol{\omega}$ the loss function is at it's lowest points and are the most optimal weights for predicting the original dataset. This is undesired though and regularization methods are used to help the data better work on new datasets.

Chapter 3

Polynomial Models

3.1 Polynomial Regression

3.1.1 Training Function

3.1.2 Loss Function

3.1.3 Optimization

Chapter 4

Decision Trees

4.1 Decision Tree Regressors

Chapter 5

Regularization

5.1 Regularization of Regression Models

Chapter 6

Evaluation

6.1 Evaluating Regression Models