"Math for ML and DS" Specialization

Andrii X

1 Linear Algebra for Machine Learning and Data Science

1.1 System of linear equations:

- Systems of equations: **Non-singular** (complete), **Singular** (Redundant, Contradictory).
- **Determinant** of a matrix is the signed **factor by which areas are scaled by this matrix**.
  For A = $\begin{bmatrix} a & b \\ c & d \end{bmatrix}$ determinant is $\det(A) = ad - cb$.
  For **non-singular** system determinant has **non-zero** value.
  Determinant of an **inverse matrix** is an **inverse of determinant** for original matrix: $det(A^{-1}) = \frac{1}{det(A)}$.

1.2 Solving system of linear equations:

- **Rank** of a matrix tells how much information matrix has. For example, a matrix with 3 rows max rank is 3, since 3 eq and 3 pieces of information, but if one of eq is just a combination of 2 others then rank will be 2.
  Rank of a matrix can be calculated via **row echelon form**:
  - Zero rows at the bottom.
  - Each row has pivot (leftmost non-zero entry).
  - Every pivot is to the right of the pivots on the rows above.
  - Rank of the matrix is the number of pivots

  Difference of Reduced REF from REF is that any number above a pivot is **0** in RREF.

1.3 Vectors and Linear Transformations:

- **Norm** is a function from vector space to the non-nega tive real numbers that behaves **like the distance** from the origin.
- **Operations** on vectors: sum and difference of vectors, multiplication by scalar, dot product.

- **Dot product**: $a \cdot b = |a| \cdot |b| \cdot \cos\Theta$ or $a \cdot b = a_x \cdot b_x + a_y \cdot b_y$.
  Dot product provides an easy way to **test the orthogonality** between vectors. If $\mathbf{x}$ and $\mathbf{y}$ are orthogonal (the angle between vectors is $90°$), then since $\cos(90°) = 0$, it implies that the d**ot product of any two orthogonal vectors must be** 0. The geometric definition of the Dot Product can be used to evaluate **vector similarity**.

- **Matrices** can be seen as **linear transformations**.
  Matrix multiplication is defined only if the number of columns of matrix A is equal to the number of rows of matrix B.

- A **transformation** is a function from one vector space to another that respects the underlying (linear) structure of each vector space. Referring to a specific transformation, you can use a symbol, such as T. Specifying the spaces containing the input and output vectors, e.g., $\mathbb{R}^2$ and $\mathbb{R}^3$, you can write $T : \mathbb{R}^2 \rightarrow \mathbb{R}^3$. Transforming vector $v \in \mathbb{R}^2$ into the vector $w \in \mathbb{R}^3$ by the transformation T, you can use the notation $T(v) = w$ and read it as "T of v equals w"or "vector w is an image of vector v with the transformation T".

- A **transformation** T is said to **be linear** if the following two properties are true for any scalar k and any input vectors u and v:
  - $T(k \cdot v) = k \cdot T(v)$,
  - $T(u + v) = T(u) + T(v)$.

- **Transformations Defined as a Matrix Multiplication**: Let $L : \mathbb{R}^m \rightarrow \mathbb{R}^n$ be defined by a matrix A, where $L(v) = A \cdot v$, multiplication of the matrix $A(n \times m)$ and vector $v(m \times 1)$ results in the vector $w(n \times 1)$.

- **Simple Linear Regression:**
  - **Linear regression** is a **linear approach** for modelling the relationship between a **scalar response** (dependent variable) and one or more **explanatory variables** (independent variables).
  - **Simple linear regression model** can be written as $\hat{y} = wx + b$, where $\hat{y}$ is a prediction of dependent variable y based on independent variable x using a line equation with the slope w and intercept b.
  - The **simplest neural network** model has only **one perceptron**. It takes some inputs and calculates the output value. Weight (w) and bias (b) are the parameters which will get updated when you will train the model.
  - If you have m training examples, vector operations will give you a chance to perform the calculations **simultaneously** for all of them! Organise all training examples as a **vector** X of size $(1 \times m)$. Then perform **scalar multiplication** of X $(1 \times m)$ by a scalar w, adding b, which will be **broadcasted** to the vector of size $(1 \times m)$: $\hat{Y} = wX + b$. This set of calculations is called **forward propagation**.

- Now, you can compare the resulting vector of the predictions $\hat{Y}(1\times m)$ with the original vector of data Y. This can be done with the so-called **cost function** that measures how close vector of predictions to the training data. It evaluates how well the parameters w and b work to solve the problem. There are many different cost functions available depending on the nature of a problem. For a simple neural network it can be calculated it as:

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

  The **aim is to minimize the cost function** during the training, which will minimize the differences between original values $y^{(i)}$ and predicted values $\hat{y}^{(i)}$ (division by $2m$ is taken just for scaling purposes).

- Next step is to **adjust the weights and bias**, in order to minimize the cost function. This process is called **backward propagation** and is done iteratively: you update the parameters with a small change and repeat the process.

- **Multiple Linear Regression:**
  - **Multiple linear regression** model with two independent variables x_1, x_2 can be written as

$$\hat{y} = w_1 x_1 + w_2 x_2 + b = \mathbf{W}\mathbf{x} + b,$$

    where $\mathbf{W}\mathbf{x}$ is the **dot product** of the input vector $\mathbf{x} = [x_1, x_2]$ and parameters vector $\mathbf{W} = [w_1, w_2]$, scalar parameter b is the intercept.

1.4 Eigenvectors:

- **Span** – set of all the linear combinations of a number vector. **1 vector span is a line**.
- A set B of vectors in vector space V is called **basis** if **every element of** V may be written in a unique way as a finite linear **combination of elements of** B. A **basis** is a **minimal spanning set**.
- An **eigenvector** or characteristic vector of a **linear transformation** is a **nonzero vector that changes at most by a scalar factor** when that linear transformation is applied to it.
  The corresponding **eigenvalue**, often denoted by λ, is the **factor by which the eigenvector is scaled**. Geometrically, an **eigenvector**, corresponding to a real nonzero eigenvalue, **points in a direction in which it is stretched by the transformation** and the eigenvalue is the factor by which it is stretched. If the eigenvalue is **negative**, the direction is **reversed**. Loosely speaking, in a multidimensional vector space, the eigenvector is not rotated.

- A square matrix is called a **Markov matrix** if all entries are **nonnegative** and the sum of each column elements is equal to **1**.

2 Calculus for Machine Learning and Data Science

2.1 Derivatives:

- The derivative is a **continuous description of how a function changes with small changes** in one or multiple **variables**.
- For a function denoted by $y = f(x)$, the **derivative of f is expressed as**:
  - In Lagrange's notation: $f'(x)$.
  - In Leibniz's notation: $\frac{dy}{dx} = f'(x)$.
- Some **common derivatives**:
  1) For a **line** represented by $y = f(x) = ax+b$, the derivative is $f'(x) = a$.
  2) For a function represented by $y = f(x) = x^n$, the derivative is $f'(x) = n \cdot x^{(n-1)}$.
  3) For the function $f(x) = e^x$, the derivative is $f'(x) = e^x$.
  4) The derivative of $\log(y)$ is $\frac{1}{y}$.
- Derivative of the **Inverse**: for functions $f(x) = x^2$ and $g(y) = \sqrt{y}$, the derivative of g is $g'(y) = \frac{1}{f'(x)}$.
- **Differentiable** Function:
  1) For a function to be differentiable at a point the derivative has to exist for that point.
  2) For a function to be differentiable at an interval the derivative has to exist for **every** point in the interval.
- **Non Differentiable** Function:
  1) Generally, when a function has a **corner or a cusp**, the function is not differentiable at that point.
  2) **Piece-wise** functions.
  3) Functions with **vertical tangents**.
- **Properties** of the derivative:
  1) Multiplication by scalars: $\frac{d}{dx}(c \cdot f(x)) = c \cdot \frac{d}{dx}f(x)$.
  2) The sum rule: $\frac{d}{dx}(f(x) + g(x)) = \frac{d}{dx}f(x) + \frac{d}{dx}g(x)$.
  3) The product rule: $\frac{d}{dx}(f(x) \cdot g(x)) = f'(x) \cdot g(x) + f(x) \cdot g'(x)$.
  4) The chain rule: $\frac{d}{dt}g(h(t)) = \frac{dg}{dh} \cdot \frac{dh}{dt} = g'(h(t)) \cdot h'(t)$.

2.2 Optimization:

- **Optimization of squared loss**: y_i as the actual value of the i-th sample, n as the total number of samples, w as the model parameters (weights). The squared loss for a single sample is given by: $L(y_i, \hat{y}_i) = (y_i - \hat{y}_i)^2$.

  In case of linear regression, the optimization problem can be written as:

$$\min_{w} \frac{1}{n} \sum_{i=1}^{n} (y_i - w^T x_i - b)^2$$

- **Log-loss**, also known as logistic loss or cross-entropy loss, is often used in binary classification problems. Given a set of true labels y_i and predicted probabilities p_i, the log-loss for a single observation is defined as:

$$L(y_i, p_i) = -y_i \log(p_i) - (1 - y_i) \log(1 - p_i)$$

  The goal of optimization is to find the model parameters that **minimize** the average log-loss across all observations in the training set. If we have N observations, the average log-loss is:

$$L = -\frac{1}{N} \sum_{i=1}^{N} [y_i \log(p_i) + (1 - y_i) \log(1 - p_i)]$$

- *continue