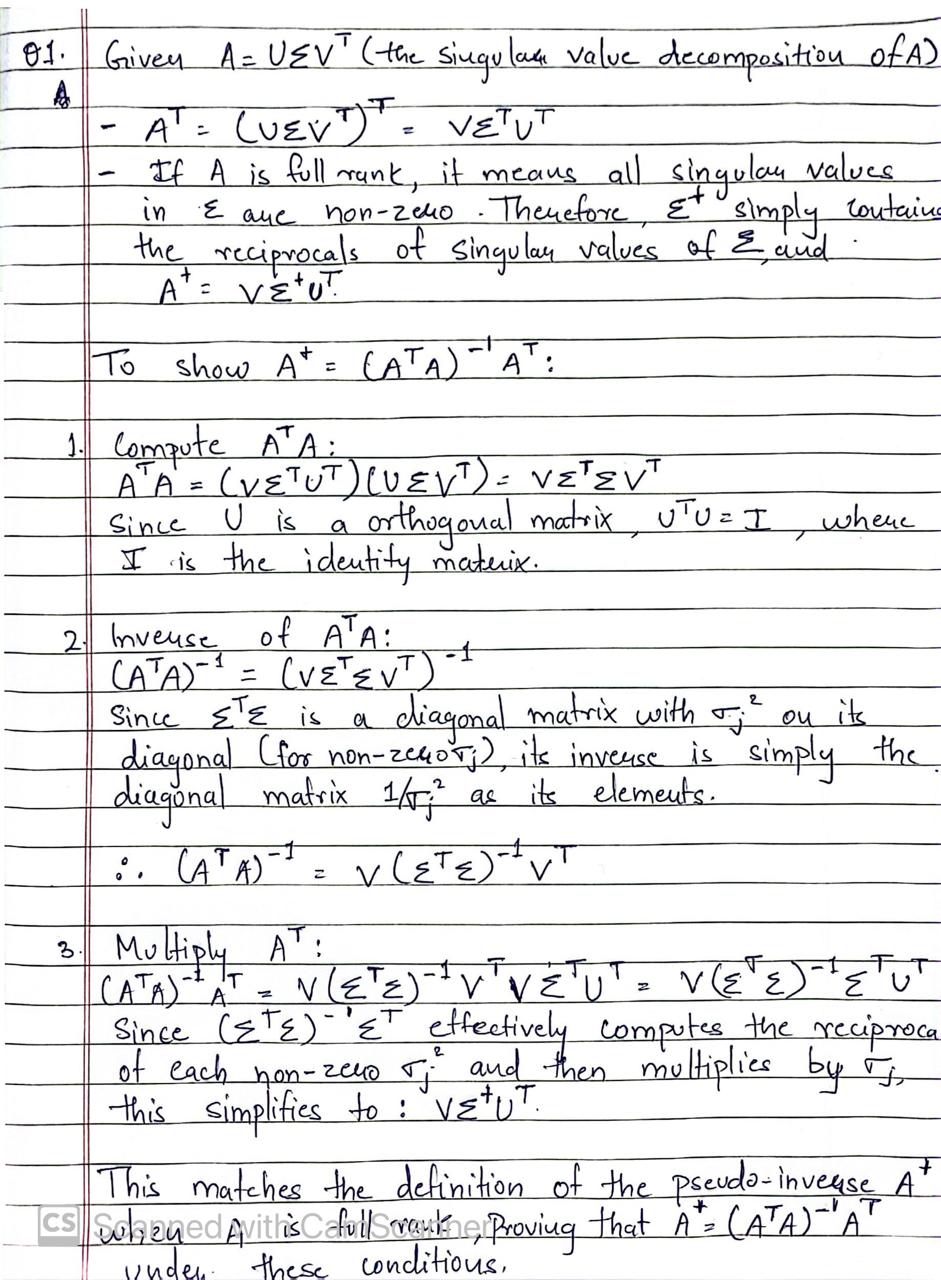
**Solution 1:**



**Solution 2A):**

* **Ridge Regularization (L2 regularization or Tikhonov regularization):**

The loss function is defined as L(c) = ∥y−Ac∥ 2  + λ∥c∥2 , where the goal is to find coefficient vectors c that not only provide a good fit to the data (minimizing ∥y−Ac∥ 2 ) but are also small in magnitude (minimizing λ∥c∥2 ) to avoid overfitting. The regularization parameter λ balances fitting accuracy against the simplicity of the model.

* **Constrained Minimization Problem:**

Here, the objective is to minimize L(c)=∥y−Ac∥2  , with the condition that ∥c∥2  ≤ t. This formulation directly aims to minimize the discrepancy between the observed and predicted values while ensuring the coefficients' magnitude does not surpass a certain threshold t.

* **Equivalence and Relationship between λ and t:**

These two problem settings are essentially equivalent in that for a specific value of λ, one can find a corresponding t that leads to the same coefficient vector c, and similarly, a specific t correlates with a λ to yield the same outcome. This equivalence stems from both methodologies' shared goal of balancing a good data fit with preventing coefficient magnitudes from becoming excessively large, thereby reducing the risk of overfitting.

The interplay between λ and t is complex and generally non-linear, meaning the exact relationship can't be precisely defined without solving one of the optimization problems. However, it's generally observed that:

* A larger λ value in ridge regularization equates to a smaller t in the constrained approach, resulting in coefficients c of smaller magnitude. This happens because a higher λ imposes a greater penalty on the coefficients' size, effectively shrinking them, akin to setting a stricter (smaller) constraint on their magnitude through t.
* Conversely, a lower λ corresponds to a larger t, allowing for coefficients c of greater magnitude. This is because a smaller λ lessens the penalty on the coefficients' size, permitting them to grow larger, similar to a more lenient (larger) magnitude constraint t.

**Solution 2B):**

**LASSO Regularization:**

This technique optimizes the function L(c)=∥y−Ac∥2 +λ∥c∥1, aiming to minimize the sum of squared residuals while imposing a penalty on the L1-norm (the sum of the absolute values) of the coefficients, regulated by λ. The λ parameter plays a crucial role in encouraging a sparse solution, effectively setting some coefficients to zero for sufficiently high values.

**Constrained Minimization Problem:**

The goal here is to minimize L(c)=∥y−Ac∥2 with the constraint that ∥c∥1 ≤t. This setup directly targets minimizing the residual sum of squares while ensuring the L1-norm of the coefficients remains below a specified threshold t, promoting sparsity in the coefficient values through a direct constraint.

**Equivalence and Relationship between λ and t:**

The equivalence between LASSO regularization and the constrained minimization approach is grounded in the idea that for any chosen λ, one can identify a corresponding t that leads to an identical coefficient solution c, and similarly, for a chosen t, there is a λ that results in the same solution. This parallel is due to both λ and t's roles in fostering sparsity within the coefficients.

The relationship between λ and t is complex and inherently non-linear, meaning it cannot be directly quantified without delving into the specifics of the optimization problem at hand. Generally speaking:

* A higher λ value in the LASSO framework is analogous to a lower t in the constrained model, enforcing greater sparsity by imposing a more stringent penalty on the absolute sizes of the coefficients or by more strictly limiting their total sum.
* On the flip side, a lower λ aligns with a higher t, leading to less pronounced sparsity by applying a milder penalty or allowing a greater total sum of the absolute values of the coefficients.

**Solution 3A):**

**The Basic Idea Behind Naive Bayes:**

The term "naive" in Naive Bayes comes from its basic assumption that all the things you're looking at (features) don't depend on each other when you know the category (class label) they belong to. It's like saying that each piece of information affects the outcome on its own, without considering how it might be connected to other pieces of information. Even though this isn't always true in real life, since things can be related in complicated ways, Naive Bayes models still work really well for many tasks despite this simple assumption.

**Solution 3B):**

Using Bayes' Theorem, we can update what we think is likely when we get new information. Here, we're looking at whether a computer script is harmful (malicious) or safe (benign) based on whether it uses the ast library.

Given numbers:

There are 10,000 scripts in total, with 9,500 safe and 500 harmful.

So, P(M)= 500/10000 = 0.05 (chance of being harmful).

P(B) = 9500/10000 = 0.95 (chance of being safe).

P(A∣M) = 100/500 = 0.2 (using ast if harmful).

P(A∣B) = 200/9500 ≈ 0.0211 (using ast if safe).

First, let's find

P(A), the overall chance of using ast:

P(A) = P(A∣M) \* P(M) + P(A∣B) \* P(B) = (0.2) \* (0.05) + (0.0211) \* (0.95) ≈ 0.030095

Now, let's find

P(M∣A), the chance of being harmful given the script uses ast:

P(M∣A) = P(A) \* P(A∣M) \* P(M) = 0.030095 \* (0.2)(0.05) ≈ 0.3328

So, if a script uses the ast library, there's about a 33.28% chance it's harmful.