**Regularization**

Table of Contents

[Mathematical Approach 3](#_Toc100760889)

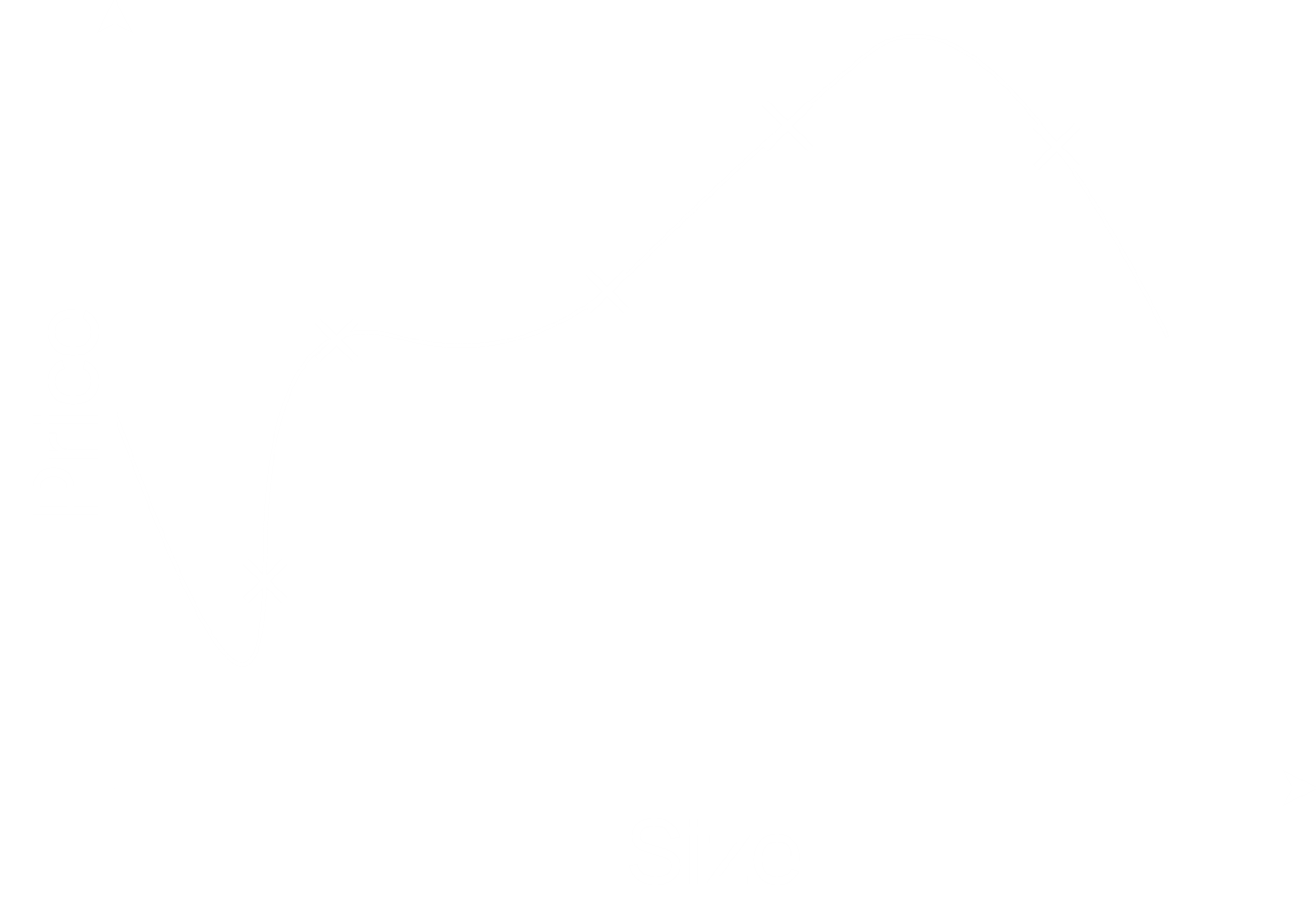
[Iterative Approach 5](#_Toc100760890)

[Vectorized Approach 5](#_Toc100760891)

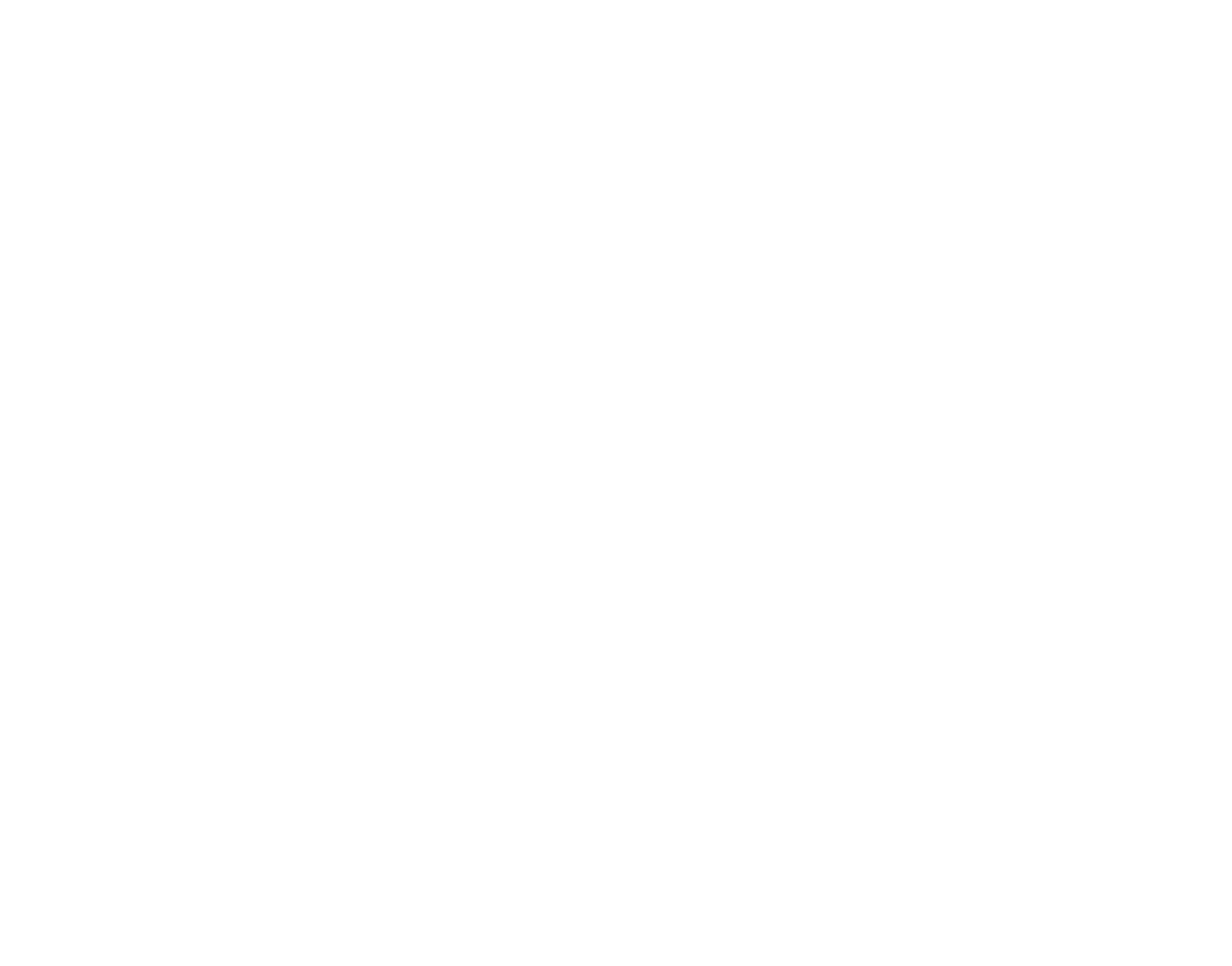
[Logistic Regression 6](#_Toc100760892)

[L1 and L2 Regularization 6](#_Toc100760893)

We have already discussed the problem of **overfitting** a little. This occurs when we have **too many features**. For a polynomial regression problem, we may end up with a situation like the one below:



For logistic regression, we could end up with something like this:



These cases are said to be overfitted, or have **high variance**. The other extreme would be **underfitting**, also called **high bias**. We can deal with underfitting by adding features, as discussed when introducing polynomial linear regression. We can deal with overfitting by simply removing features, perhaps using some model selection algorithm, which we will cover later on.

However, just removing features is not a convenient solution. There are cases where some feature has some tiny effect on the outcome of our model, so removing it makes our model just a little less accurate. Instead, we can use regularization.

In **regularization**, we force the parameters of the higher order polynomials, such as , and so one, to be very close to . By doing this, we can keep the features while also reducing the effect they have on the model. This prevents the model from being overfitted.

## Mathematical Approach

Consider that we have the following hypothesis:

Here, suppose the parameters and are causing overfitting, so we need to make them approximately .

Using regularization, this is how we can do it:

Remember that our goal is to **minimize** . If we add to this cost, in order to minimize it, the values of and have to become very small.

However, we do not know for sure that we need to make and small. It could be that we need to make small to avoid overfitting. To deal with this, we simply add a large value for **all the parameters**.

Here, is the **regularization parameter**, which was in the previous example, and is the **regularization term**.

Consider why this works. Suppose all the parameters become approximately . This will cause the curve to become a straight line, which is **underfitted**. This does not give us the minimum cost. This means that some parameters, which have a lower effect on the overall shape of the curve, will become , while other parameters, which describe the overall shape of the curve, will stay high. This allows us to essentially remove the parameters which cause overfitting without even knowing which ones they are.

The regularization parameter is our choice. We need to gradually increase it, check the outcome and decide what value to use. If the value is too small, overfitting will remain. If the value is too large, we will have underfitting.

Notice that the regularization term starts at . , the bias, is not included. This is because the bias cannot be causing overfitting. It’s like the intercept in . It does not describe the shape itself.

### Iterative Approach

For the equation of above,

The second equation can be re-written:

Thus, an easy way to remember this is that we are multiplying with for . Whatever value of we are getting, we decrease it a little bit, say by making .

### Vectorized Approach

Here, we added times a matrix. This matrix is special. It is the identity matrix, with one extra row and column of zeroes due to the fact that regularization is not applied on . We are basically adding to every other than . The matrix is an matrix.

A fun thing about this is that due to the addition of the identity matrix, there is no possibility anymore of this equation every being **non-invertible**.

## Logistic Regression

Regardless of whether we are performing linear regression or logistic regression, the equations will remain the **same**. This is because we did not change at all. The only difference between linear and logistic regression is in there, since linear regression states that while logistic regression states that .

## L1 and L2 Regularization

Regularization is actually of 2 types, L1 and L2.

Everything we have seen so far falls under **L2 regularization**, also called **Ridge Regularization**. It is called L2 regularization because we added to the cost function. **L1 regularization**, also called **Lasso Regularization**, on the other hand, adds .

The difference, practically, is that in L2 regularization, we are trying to pull back the cost to prevent it from reaching the minimal possible value. We end up with values that are nearly but not exactly. This situation is called a **non-sparse** situation. This can be seen from the part where we set the value of to . By contrast, in L1 regularization, we are adding or subtracting some value of (the derivative of is if and if ). This means we can end up with some values that are **exactly** . This situation is called a **sparse** situation. L1 regularization is arguably a form of **feature selection**.