

Objectives

- Regression
 - Loss Functions
 - Lectures 1-8 Review
 - Practice Problems
-

1 Regression

The previous lectures and materials have mostly covered classification tasks. As we know, classification can be binary where there are two (2) labels:

$$y \in \{-1, 1\}$$

Though, classification tasks can contain $x \in \mathbb{Z}^+$ labels. In a small example, the Iris dataset has three (3) different labels representing the Iris flowers setosa, versicolor, and virginica.

$$y \in \{1, 2, 3\}$$

Regression has the same goal of “labeling” features, but not categorically and instead with real numbers.

$$y \in \mathbb{R}$$

For both regression and classification, the dataset takes the same form:

$$D = \{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$$

Where for each feature, x_i , there is an associated label, y_i .

2 Loss Functions

Loss is calculated by a function for supervised machine learning tasks. Its responsibility is to answer the question:

By how much is the model *wrong*?

2.1 Zero-to-One Loss

Loss functions can be simple and binary, similar to classification tasks. For example, a zero-to-one loss function would be denoted as

$$l_{01} \in \{0, 1\}$$

In this function, 1 represents a loss, and 0 represents no loss.

When speaking about loss functions, some may interchangeably call a function's **Aggregate** (summation) a loss function, but this is more accurately described as being a **Cost** function.

$$\sum_{i=1}^n l_{01}(i) = 1 + 0 + 1 + 1 + 0 \dots$$

In this cost function, the higher the value the more loss the model has. Simply put: the higher the score \implies the worse the performance.

2.2 Absolute Loss

Absolute Loss takes the actual label associated with the feature y_i and gets the absolute difference with \hat{y}_i label value.¹

$$\begin{aligned}\text{Loss} &= |y_i - \hat{y}_i| \\ \text{Cost} &= \sum_{i=1}^n |y_i - \hat{y}_i|\end{aligned}$$

The downside of the absolute loss function is that it is not **differentiable**. Recall that differentiable means there exists a derivative.

2.3 Square Loss

Square loss takes the actual label associated with the feature y_i and squares its difference with the computed \hat{y}_i label value.

$$\begin{aligned}\text{Loss} &= (y_i - \hat{y}_i)^2 \\ \text{Cost} &= \sum_{i=1}^n (y_i - \hat{y}_i)^2\end{aligned}$$

Because this loss function squares its result, it can be sensitive to outliers. However, because it is differentiable, it is often preferred over the absolute loss function.

¹

¹The hat symbol ^ implies that this variable is a computed value.

2.4 Objective Function

Recall that we covered how a **Cost** function is the aggregation of a **Loss** function.

$$\text{Cost} = \sum_{i=1}^n \text{Loss}$$

Similarly, there is another function known as the **Objective Function** which is the sum of the cost and a **Regularizer**:

$$\text{Objective Function} = \text{Cost} + \text{Regularizer}$$

The purpose of the regularizer is to add a penalty to shrink a coefficient, or in other words, to minimize the loss function. This can be helpful for loss functions like square loss as it is sensitive to outliers and can compute unexpectedly large coefficients.

2.5 Gradient Descent

Gradient descent is essentially a use case for calculating derivatives. Its purpose in Supervised Machine Learning (ML) tasks is to find local minimum and maximums (where the derivative is 0) in order to optimize the loss function of a model.

Here are the names of some notable gradient descent methods:

- Vinalla
- Stochastic
- Minibatch
- Hession
- Momentum

3 Lectures 1-7 Review

3.1 The Perceptron

- Developed by **Rosenblatt** in **1957** at Cornell University
- **Binary Classification** Task
- Assumes data is **Linearly Separable**
- Calculates weight vector \vec{w} and bias b

- “Bundle & Save”: This trick will save time by removing the need to calculate b . We do this by ”bundling” b into \vec{w} :

$$\vec{w} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_p \\ b \end{bmatrix}$$

- $\vec{w} \in \mathcal{H}$ where \mathcal{H} is the **Decision Boundary** or **Hyperplane**

$$\mathcal{H} = \{\vec{x} | \vec{w}^T \vec{x} + b = 0\}$$

3.2 KNN

- Stands for k **Nearest Neighbors**
- Loops over all neighbors and calculates distance
- Sorts distances and returns k nearest
- **Regression:** Averages values to nearest neighbor values
- **Classification:** Sets label to the mode of nearest neighbor values

```
w = 0
m = 1
while m > 0:
    m = 0
    for i = 1..n:
        if y[i] * dot(w, x[i]) <= 0:
            w = w + alpha * y[i] * x[i]
            m = m + 1
        end
    end
end
end
```

4 Practice Problems

1. What is the benefit of having less dimensions, features, or components in your vectors?
2. What are the different types of Supervised ML?

3. What is a loss function?
4. What is KNN? How could it be used differently for both Classification and Regression?
5. What does SVM stand for? What is its purpose?
6. What is the definition of a distance function? Give examples.