# Regression Diagnostics & Optimizations

Jerry Peng

## 1 Regression Diagnostics

### 1.1 Mean Squared Error (MSE)

It is fully called "Mean Squared Error" (MSE), which is a commonly used performance metric for regression models. It calculates the average of the squared errors between each predicted value and the actual value. Suppose we have a linear regression model $\hat{y} = X\theta$, where $\hat{y}$ represents the predicted value of the model (prediction), and $y$ represents the actual value (label). The formula for MSE is as follows:"

$$\begin{aligned} MSE &= \frac{1}{N}\|y - \hat{y}\|_2^2 \\ &= \frac{1}{N}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \end{aligned} \tag{1}$$

### 1.2 Why MSE?

**Sensitivity to Large Errors** Due to the squared error term, MSE is more sensitive to larger errors than the Mean Absolute Error (MAE). This means it will penalize larger errors more heavily.

**Differentiability** MSE is differentiable, so it is suitable for optimization algorithms that require derivatives, such as Gradient Descent. On the other hand, MAE is not differentiable at zero, which may pose problems for these optimization algorithms.

**Statistical Properties** In a linear regression environment with Gaussian noise, MSE will produce an unbiased estimate of the coefficients, and it is also the maximum likelihood estimate.

However, MSE is not always the best metric for every situation. For example, if the error distribution is not symmetrical, or if there are extreme outliers in the data, then MAE might be a better choice because it is less sensitive to outliers.

### 1.3 Confusion Matrix

A Confusion Matrix is a table used to evaluate the performance of a classification model. It provides a detailed breakdown of true positive, true negative, false positive, and false negative predictions made by the model. The matrix is often used in machine learning and statistics to understand how well a classification model is performing and where it is making mistakes.

| | Actual Positive | Actual Negative |
|---|---|---|
| Predicate Positive | True Positive | False Positive |
| Predicate Negative | False Negative | True Negative |

**Definitions**

- **True Positive (TP):** The model correctly predicted the positive class.
- **True Negative (TN):** The model correctly predicted the negative class.
- **False Positive (FP):** The model incorrectly predicted the positive class (Type I error).
- **False Negative (FN):** The model incorrectly predicted the negative class (Type II error).

**Derived Metrics** Various performance metrics can be derived from the Confusion Matrix:

- **Accuracy:** (TP+TN)/(TP+TN+FP+FN)
- **Precision:** TP/(TP+FP)
- **Recall (Sensitivity):** TP/(TP+FN)
- **Specificity:** TN/(TN+FP)
- **F1 Score:** $2 \times (Precision \times Recall)/(Precision + Recall)$

The Confusion Matrix is particularly useful when the classes are imbalanced or when the costs of different types of errors vary. It provides a more detailed view of the model's performance compared to using a single metric like accuracy.

## 1.4 Balanced Error Rate

It is a metric used to evaluate the performance of classification models, particularly in cases where the classes are imbalanced or when different types of classification errors have different costs. BER is designed to give a balanced view of the model's performance across all classes. In a binary classification problem, the Balanced Error Rate is defined as the average of the error rates for each class. Specifically, it is the average of the false positive rate and the false negative rate, or equivalently, one minus the average of the true positive rate (sensitivity) and the true negative rate (specificity).

**Formula for Binary Classification**

$$\text{BER} = \frac{1}{2}(\frac{FP}{FP+TN} + \frac{FN}{TP+FN}) \tag{2}$$

**Formula for Multi-Class Classification** For multi-class classification problems, BER is often calculated as the average of the error rates for each class, weighted by the class distribution in the test set.

$$\text{BER} = \frac{1}{N}\sum_{i=1}^{N}(\text{Error Rate of Class } i) \tag{3}$$

**Importance of BER**

- **Class Imbalance:** In datasets where one class significantly outnumbers the other(s), accuracy can be misleading. BER provides a more balanced measure of performance.
- **Different Error Costs:** In some applications, false positives and false negatives may have different costs. BER allows for a more balanced evaluation in such cases.
- **Fairness:** BER can be a useful metric when it's important for the model to perform equally well across different classes, such as in fairness-sensitive applications.

## 1.5 Coefficient of Determination

The coefficient of determination, also known as R-squared ($R^2$), is a statistical metric used to measure how well a regression model fits the data. In simple linear regression, the coefficient of determination R-squared represents the proportion of the variance in the dependent variable that is explained by the model. In other words, it measures the explanatory power of the regression model.

First, let's review three commonly used parameters: Mean, Variance, and MSE:

$$Mean : \bar{y} = \frac{1}{N}\sum_{i=1}^{n} y_i \tag{4}$$

$$Variance : Var(y) = \frac{1}{N}\sum_{i=1}^{n}(y_i - \bar{y})^2 \tag{5}$$

$$MSE = \frac{1}{N}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{6}$$

**Fraction of Variance Unexplained(FVU)** It refers to that part of the total variability in the dataset that is not explained by the model. The total variance is the sum of squares of the differences between the observed dependent variable and its mean. This is commonly referred to as the total sum of squares (SST). The part of the variance that the model explains is called the explained sum of squares (SSE), and the part of the variance that the model does not explain is called the residual sum of squares (SSR). The calculation formula is as follows:

$$SSR = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \tag{7}$$

$$SST = \sum_{i=1}^{n} (y_i - \bar{y}_i)^2 \tag{8}$$

$$\begin{aligned} FVU(f) &= \frac{MSE(f)}{Var(y)} \\ &= \frac{SSR}{SST} \end{aligned} \tag{9}$$

A lower FVU indicates that the model explains a larger portion of the variance, and therefore fits the data better. Conversely, a higher FVU indicates that the model does not explain a large portion of the variance.

After obtaining the definition of FVU, we can calculate the coefficient of determination (R-squared). The calculation formula is as follows:

$$R - squared = 1 - FVU \tag{10}$$

The range of R-squared is from 0 to 1. If R-squared is close to 1, it indicates that the regression model can explain the variability of the dependent variable well, and the model fits well. If R-squared is close to 0, then the explanatory power of the model is very low, and the fit is poor.

It's important to note that while a higher R-squared usually means a better fit of the model, it doesn't necessarily mean the model is good. If the model is too complex, even though it can achieve a high R-squared, it might lead to overfitting, meaning the model may not perform well on new, unknown data. Therefore, when using R-squared to evaluate a model, one should also consider the complexity of the model and the issue of overfitting.

### 1.6 Overfitting

"Overfitting is a common problem in machine learning and statistical modeling. It occurs when the model is overly complex or when there is insufficient training data, causing the model to fit the training data too precisely. An overfitted model may have a very small error on the training set, but a much larger error on the test set, indicating poor generalization capability of the model. Typically, we can avoid overfitting through the following methods:

- Use more training data.
- Reduce the complexity of the model. In linear regression, choose fewer or more strongly correlated features.
- Use regularization, such as L1 or L2 regularization.
- Use cross-validation to select and validate the model.
- Preprocess the data.

## 2 Optimization

### 2.1 Regularization

Regularization is a technique to prevent overfitting in machine learning models. It controls the complexity of the model by adding a penalty term to the model's loss function, preventing the model from fitting the training data too closely. The basic idea is to add a regularization term to the loss function, which is related to the size of the model parameters (such as coefficients in linear regression or weights in neural networks). The role of the regularization term is to increase the model's loss when the complexity of the model increases (i.e., when the values of the parameters become very large).

There are various forms of regularization, with L1 regularization and L2 regularization being the most common. L1 regularization can make some parameters zero, achieving the purpose of feature selection, making the model sparse. L2 regularization, on the other hand, will make the parameters smaller but not exactly zero, preventing the model from being overly sensitive to a particular feature.

In summary, regularization improves the model's generalization capability on unseen data by sacrificing some of its performance on the training set, preventing overfitting.

**L1** It is also known as Lasso (Least Absolute Shrinkage and Selection Operator). The penalty term for L1 regularization is the sum of the absolute values of the model weights. This characteristic of regularization will cause some of the model's weight parameters to shrink to zero, thereby achieving feature selection. Therefore, L1 regularization can be used to achieve sparsity in the model, meaning only a few features in the model are effective.

$$\begin{aligned} L_1 &= \frac{1}{N}\sum(y-\hat{y})^2 + \lambda\sum|b| \\ &= \frac{1}{N}\|y-\hat{y}\|_2^2 + \lambda|\theta| \end{aligned} \tag{11}$$

**L2** It is also known as Ridge (Ridge Regression). The penalty term for L2 regularization is half of the sum of the squares of the model weights. Unlike L1 regularization, L2 regularization does not reduce the model weights to zero, but instead makes all weights tend uniformly towards zero. L2 regularization can prevent the model from being overly sensitive to a single feature because all features will participate in the model training to some extent.

$$\begin{aligned} L_2 &= \frac{1}{N}\sum(y-\hat{y})^2 + \lambda\sum b^2 \\ &= \frac{1}{N}\|y-\hat{y}\|_2^2 + \lambda\|\theta\|_2^2 \end{aligned} \tag{12}$$

The regularization parameter $\lambda$ is a parameter that we need to set ourselves to control the strength of regularization. The larger the value of $\lambda$, the stronger the regularization, and the stricter the restriction on the model weights. When the value of $\lambda$ is 0, the model becomes a regular linear regression or logistic regression."

## 2.2 Gradient Descent

Gradient Descent is an optimization algorithm used to solve machine learning and deep learning models, especially in situations where direct solutions are not feasible. Its goal is to minimize the loss function by iteratively updating the model parameters. Mathematically, the gradient represents the directional derivative of a function at a certain point, which is the direction in which the function changes most rapidly. In the gradient descent algorithm, we update the parameters in the direction of the negative gradient because this direction results in the fastest decrease in the function value. The basic steps of the gradient descent algorithm are as follows:

1. Initialize the model parameters.
2. Calculate the gradient of the loss function. The gradient is the partial derivative of the loss function with respect to the model parameters.
3. Update the model parameters in the direction of the negative gradient. The magnitude of the update is determined by the learning rate (a preset positive number).
4. Repeat steps 2 and 3 until the termination criteria are met, such as when the gradient is close to 0 (i.e., a local minimum is found), or when a preset maximum number of iterations is reached.

### 2.2.1 Example

Let's look at an example of gradient descent. Suppose we are using a linear regression model to predict data. The model parameters (weights) are represented as $\theta$, the input data as $X$(with dimensions $m \times n$, where $m$ represents the number of samples and $n$ represents the number of features), the target values as $y$, and the Mean Squared Error (MSE) with L2 regularization as the loss function. The loss function can then be defined as:

$$J(\theta) = \frac{1}{2m}||X\theta - y||^2 + \frac{\lambda}{2m}||\theta||^2 \tag{13}$$

$\lambda$ is the regularization coefficient.

**Note:** Why do we use $\frac{1}{2m}$ and $\frac{\lambda}{2m}$ at here? Let's see the MSE expression without $\frac{1}{2m}$:

$$MSE = \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$$

Taking the derivative with respect to $\theta_j$, we get:

$$\frac{\partial MSE}{\partial \theta_j} = \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}$$

Now let's see the MSE expression with $\frac{1}{2m}$:

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$$

Take the derivative:

$$\frac{\partial J(\theta)}{\partial \theta_j} = \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}$$

At this point, the power of 2 in the power rule will be canceled out by the factor of $\frac{1}{2}$, so the two formulas after differentiation are the same. Therefore, in practice, we often use $\frac{1}{2m}$. The main reason for this is for mathematical convenience. Since the power of 2 is canceled out, including the $\frac{1}{2}$ factor can simplify the derivative. This makes the equation clearer and simplifies the calculation.

**Gradient** Taking the gradient (partial derivative) of the loss function 13, we get:

$$\nabla J(\theta) = \frac{1}{m}X^T(X\theta - y) + \frac{\lambda}{m}\theta \tag{14}$$

The formula of update $\theta$ is

$$\theta = \theta - \alpha\nabla J(\theta) \tag{15}$$

$\alpha$ is learning rate, and it usually has no fixed value, you need to tune it by yourself. The algorithm terminates when the gradient $\nabla J(\theta)$ approaches 0 or reaches the iteration limit.

## References