ML Concepts

Behrooz Azarkhalili[1]

[1]Life Language Processing Lab, University of California, Berkeley
[1]azarkhalili@behrooz.tech

Contents

Preprint. Under review.

1 Fundamental Concepts in Machine Learning

In machine learning, understanding the concepts of loss, metrics, bias, and variance is essential for developing, evaluating, and improving models. These concepts are central to diagnosing model performance and ensuring that the model predictions are accurate and reliable.

1.1 Loss Functions

Loss Functions measure the discrepancy between the actual values and the predictions made by a model. They quantify how well the model performs; the lower the loss, the better the model's predictions.

1.1.1 Classification Losses

- **Cross-Entropy Loss (Log Loss)**: Measures the performance of a classification model whose output is a probability value between 0 and 1. Cross-entropy loss increases as the predicted probability diverges from the actual label.

$$\text{Cross-Entropy Loss} = -\frac{1}{N}\sum_{i=1}^{N} y_i \log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i)$$

  Example

  Let's consider a binary classification problem with 10 samples and their predicted probabilities and true labels:

Sample	Predicted Probability	True Label	Log Loss
1	0.9	1	$-\log(0.9)$
2	0.1	0	$-\log(0.9)$
3	0.8	1	$-\log(0.8)$
4	0.4	0	$-\log(0.6)$
5	0.7	1	$-\log(0.7)$
6	0.2	0	$-\log(0.8)$
7	0.6	1	$-\log(0.6)$
8	0.3	0	$-\log(0.7)$
9	0.5	1	$-\log(0.5)$
10	0.1	0	$-\log(0.9)$

  The average log loss can be computed by averaging these values.

- **Hinge Loss**: Used primarily for binary classification tasks. It is intended for use with Support Vector Machine (SVM) models.

$$\text{Hinge Loss} = \frac{1}{N}\sum_{i=1}^{N} \max(0, 1 - y_i\hat{y}_i)$$

  Example

  Consider 10 samples with true labels $y_i \in \{-1, 1\}$ and predicted scores $\hat{y}_i$:

Sample	Predicted Score	True Label	Hinge Loss
1	0.9	1	$\max(0, 1-0.9)$
2	-0.1	-1	$\max(0, 1-0.1)$
3	0.8	1	$\max(0, 1-0.8)$
4	-0.4	-1	$\max(0, 1-0.6)$
5	0.7	1	$\max(0, 1-0.7)$
6	-0.2	-1	$\max(0, 1-0.8)$
7	0.6	1	$\max(0, 1-0.6)$
8	-0.3	-1	$\max(0, 1-0.7)$
9	0.5	1	$\max(0, 1-0.5)$
10	-0.1	-1	$\max(0, 1-0.9)$

  The average hinge loss can be computed by averaging these values.

In classification:

$(x_1,y_1),(x_2,y_2),\dots,(x_n,y_n)$

↳ Input feature vector

↳ output feature scalar

Trained

$$\begin{tikzcd} \arrow[r, "x"] & \boxed{\text{Model}} \arrow[r, "\tilde{y}"] & {} \end{tikzcd}$$

In Regression:-

$(x_1,\tilde{y}_1),(x_2,\tilde{y}_2)\dots(x_n,\tilde{y}_n)$ comes from the output

→ How to evaluate the performance

↳ metrics:- checks the performance of already trained model

↳ Loss:- when we are going to train the model. [Means findings the best possible parameters of our model]

Training the model is to find the best nearest = to minimizing loss i.e. the expectation & actuality are as close as possible (↑ accuracy)

min. loss can't be negative

min loss → training → find the best possible Parameters for the model

$\begin{cases} y_1, y_2, \dots, y_n \\ \downarrow \\ \tilde{y}_1, \tilde{y}_2, \dots \tilde{y}_n \end{cases}$

Binary - classification

$y_i \in \{0,1\}$

{Red, Blue}

$0 \leq \tilde{y}_i \leq 1 \quad \tilde{y}_1 = 0.9$

class-Entropy (g) $= -\frac{1}{N}\sum_{i=1}^{n} y_i \log(\tilde{y}_i) + (1-y_i)\log(1-\tilde{y}_i)$

Perfect case:- for every sample $y_i = \tilde{y}_i$

$y_i = 1 \Rightarrow \tilde{y}_i = 1 \Rightarrow g(y_i,\tilde{y}_i) = 1 \times \log 1 + 0(\log 0)) = 0$

$y_i = 0 \Rightarrow \tilde{y}_i = 0 \Rightarrow g(y_i,\tilde{y}_i) = 0 \times \log 0 + 1(\log 1) = 0$

$g(y_i,\tilde{y}_i) \leq 0$ [always]

$0 \leq \tilde{y}_i \leq 1 \Rightarrow \log(\tilde{y}_i) \leq \log(1) = 0$

1) $y_i \geq 0$ & $\log(\tilde{y}_i) \leq 0 \Rightarrow y_i \log(\tilde{y}_i) \leq 0$

$0 \leq 1-y_i = \{0,1\}$

$0 \leq 1-y_i \leq 1 \sim \log(1-y_i) \leq 0 \quad g(y_i,\tilde{y}_i) = y_i \log(1-y_i) + (1-y_i)\log(1-\tilde{y}_i) \leq 0$

---

we train the model to minimise loss function

what is a good candidate to be used as loss?

For binary classification, Cross entropy can be a good candidate

1) Mean-Squared Error (MSE): $\frac{1}{N}\left(\sum_{i=1}^{n}(y_i - \tilde{y}_i)^2\right)$

$\tilde{y}_i$ → Predicted Value

y_i → Expected Value

in perfect-model: $y_i = \tilde{y}_i \Rightarrow y_i - \tilde{y}_i = 0$

M.S.E = 0

[M.S.E is always non-negative]

Mean-abs-Error: $0 \leq MAE = \left(\frac{1}{N}\right)\left(\sum_{i=1}^{n}|y_i - \tilde{y}_i|\right)$

in perfect Model: MAE = 0

MAE takes its minimal value when the model is perfect.

$$\begin{tikzcd} \{x_i,y_i\}_{i=1}^{N} \arrow[r] & \boxed{\text{Model}} \arrow[r] & \{y_i\}_{i=1}^{N} \end{tikzcd}$$

: $Loss = f(x_i, y_i, \tilde{y}_i)$

$\frac{d}{dy}(Loss_1) = 0 \qquad arg(Loss_1 = 0)$

$\frac{d}{dy}(Loss_2) = 0 \qquad arg(Loss_2 = 0)$

$L = f(x_i, y_i, \tilde{y}_i)$

↓

L is minimum

$L' = 0$

$x_{t+1} = x_t - \alpha f'(x_t)$

$t \to \infty : |x_{opt} - x_t| \to 0$

$|x_{opt} - x_t|$

1.1.2 Regression Losses

- **Mean Squared Error (MSE)**: The average of the squares of the errors—that is, the average squared difference between the estimated values and the actual value.

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

Example

Given 10 sample points:

True Value	Predicted Value	Squared Error
3	2.8	$(3-2.8)^2 = 0.04$
1	1.2	$(1-1.2)^2 = 0.04$
4	3.9	$(4-3.9)^2 = 0.01$
2	2.1	$(2-2.1)^2 = 0.01$
5	4.8	$(5-4.8)^2 = 0.04$
6	6.2	$(6-6.2)^2 = 0.04$
3.5	3.7	$(3.5-3.7)^2 = 0.04$
2.5	2.3	$(2.5-2.3)^2 = 0.04$
4.5	4.6	$(4.5-4.6)^2 = 0.01$
5.5	5.4	$(5.5-5.4)^2 = 0.01$

The average MSE is:

$$\text{MSE} = \frac{0.04+0.04+0.01+0.01+0.04+0.04+0.04+0.04+0.01+0.01}{10} = 0.028$$

- **Mean Absolute Error (MAE)**: The average of the absolute differences between predictions and actual observations. It provides a measure of how wrong the predictions are.

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|$$

Example

Using the same 10 sample points as above, the MAE is calculated as follows:

True Value	Predicted Value	Absolute Error
3	2.8	$\|3-2.8\| = 0.2$
1	1.2	$\|1-1.2\| = 0.2$
4	3.9	$\|4-3.9\| = 0.1$
2	2.1	$\|2-2.1\| = 0.1$
5	4.8	$\|5-4.8\| = 0.2$
6	6.2	$\|6-6.2\| = 0.2$
3.5	3.7	$\|3.5-3.7\| = 0.2$
2.5	2.3	$\|2.5-2.3\| = 0.2$
4.5	4.6	$\|4.5-4.6\| = 0.1$
5.5	5.4	$\|5.5-5.4\| = 0.1$

The average MAE is:

$$\text{MAE} = \frac{0.2+0.2+0.1+0.1+0.2+0.2+0.2+0.2+0.1+0.1}{10} = 0.16$$

1.2 Performance Metrics

Performance Metrics evaluate the effectiveness of a model using the test data. Unlike loss functions, metrics are used to interpret model performance and are not always differentiable.

1.2.1 Classification Metrics

- **Accuracy**: The ratio of correctly predicted observations to the total observations. Best used when there are equal numbers of samples in each class.

  $$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}}$$

  Example

  Given 10 samples with 8 correct predictions:

  $$\text{Accuracy} = \frac{8}{10} = 0.8$$

- **Precision**: Precision is the ratio of correctly predicted positive observations to the total predicted positives.

  $$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives + False Positives}}$$

- **Recall**: Recall (Sensitivity) is the ratio of correctly predicted positive events to all actual positives.

  $$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives + False Negatives}}$$

  Example

  For 10 samples with:

True Positives	False Positives	True Negatives	False Negatives
4	1	3	2

  Precision and Recall are:

  $$\text{Precision} = \frac{4}{4+1} = 0.8$$

  $$\text{Recall} = \frac{4}{4+2} = 0.6667$$

- **F1 Score**: The weighted harmonic average of Precision and Recall. This score takes both false positives and false negatives into account.

  $$\text{F1 Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

  Example

  Given Precision = 0.8 and Recall = 0.6667:

  $$\text{F1 Score} = 2 \cdot \frac{0.8 \cdot 0.6667}{0.8 + 0.6667} = 0.727$$

1.2.2 Regression Metrics

- **Mean Squared Error (MSE)**: The average of the squares of the errors—that is, the average squared difference between the estimated values and the actual value.
- **Mean Absolute Error (MAE)**: The average of the absolute differences between predictions and actual observations.

1.3 Bias and Variance

1.3.1 Bias

Bias refers to the error introduced by approximating a real-world problem, which may be complicated, by a much simpler model. High bias can cause an algorithm to miss the relevant relations between features and target outputs (underfitting).

1.3.2 Variance

Variance refers to the amount by which the model's prediction would change if different training data was used. High variance can cause a model to model the random noise in the training data, rather than the intended outputs (overfitting).

Examples

High Bias Scenario: A model that predicts housing prices using only the size of the house, ignoring other features like age, location, and number of rooms, may have high bias.

Examples

High Variance Scenario: A model that predicts stock prices by fitting a complex polynomial that passes through every single data point in the training data is likely to have high variance.

1.3.3 Trade-off

The Bias-Variance Tradeoff is a central problem in supervised learning. Ideally, one wants to choose a model that accurately captures the regularities in its training data but also generalizes well to unseen data. Unfortunately, a model with very low bias must pay for it with high variance and vice versa.

2 Training, Validation, and Testing of Machine Learning Models

In machine learning, the development and deployment of models involve three critical phases: training, validation, and testing. Each phase serves a distinct purpose in ensuring that the model performs well on unseen data, thereby generalizing effectively. Here's a detailed explanation of each phase:

2.1 Training the Model

Training involves feeding a machine learning algorithm with a dataset and allowing it to learn the relationships between the features (input variables) and the target (output variable). The goal is to optimize the model parameters to minimize the error on the training data.

Steps to Train the Model:

1. **Data Preparation:**
   - Split the dataset into features (X) and target (y).
   - Normalize or standardize the features if necessary.
   - Optionally, handle missing values, categorical variables, and outliers.
2. **Model Selection:**
   - Choose a suitable machine learning algorithm (e.g., linear regression, decision trees, neural networks) based on the problem (classification or regression).
3. **Initialization:**
   - Initialize the model parameters (weights and biases).
4. **Training:**
   - Use an optimization algorithm (e.g., gradient descent) to iteratively update the model parameters.
   - At each iteration (epoch), the algorithm computes the predictions, calculates the loss using a loss function (e.g., mean squared error for regression, cross-entropy for classification), and updates the parameters to minimize the loss.

Example

For a linear regression model, the training process involves finding the optimal weights $\mathbf{w}$ and bias b that minimize the mean squared error:

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - (\mathbf{w} \cdot \mathbf{x}_i + b))^2$$

where $\mathbf{x}_i$ are the features, and y_i are the true values.

2.2 Validating the Model

Validation involves assessing the model's performance on a separate dataset (validation set) that was not used during training. This helps in tuning the model's hyperparameters and preventing overfitting.

Steps to Validate the Model:

1. **Data Splitting:**
   - Split the training dataset into the new training alongside validation sets (e.g., 80% training, 20% validation).
2. **Hyperparameter Tuning:**
   - Use techniques like grid search or random search to find the best hyperparameters.
   - Train the model on the training set with different hyperparameter configurations and evaluate on the validation set.
3. **Cross-Validation:**
   - Perform k-fold cross-validation to ensure that the model's performance is consistent across different subsets of the data. In k-fold cross-validation, the dataset is divided into k subsets, and the model is trained k times, each time using a different subset as the validation set and the remaining subsets as the training set.

Example

For a decision tree, hyperparameters like max_depth and min_samples_split can be tuned using the validation set to find the best configuration that balances bias and variance.

3. Testing or Evaluating the Model

Testing involves assessing the final model's performance on a test dataset that was not used during training or validation. This gives an unbiased estimate of the model's performance on unseen data.

Steps to Test the Model:

1. **Data Splitting:**
   - Split the dataset into training, validation, and test sets (e.g., 70% training, 15% validation, 15% test).
2. **Final Model Training:**
   - Train the model on the combined training and validation sets using the best hyperparameters found during validation.
3. **Performance Evaluation:**
   - Evaluate the model on the test set using appropriate metrics (e.g., accuracy, precision, recall, F1 score for classification; mean squared error, R-squared for regression).

Example: For a classification model, the performance on the test set can be evaluated using the confusion matrix to calculate accuracy, precision, recall, and F1 score:

$$\text{Accuracy} = \frac{\text{True Positives + True Negatives}}{\text{Total Number of Samples}}$$

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives + False Positives}}$$

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives + False Negatives}}$$

$$\text{F1 Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

2.3 Summary

- **Training**: Learn model parameters using the training dataset.
- **Validation**: Tune hyperparameters and prevent overfitting using the validation dataset.
- **Testing**: Evaluate final model performance using the test dataset.