Assignement -09

Dhanush V Nayak
ee23btech11015@iith.ac.in

1 Model Selection and Optimization

In machine learning, our main goal is to find a hypothesis that will make correct predictions on new or unseen data set. But there's an assumption we make, we assume that each unseen data example has the same prior distribution. This is called the stationarity assumption. To check how good our model is we check its error rate. Generally, we split our data into two parts: a training ser and a test set. However, when we tune various model parameters (called hyperparameters), we may accidentally use the test data too much. To avoid this, we introduce a validation set to choose the best model, saving the test set for final evaluation. If we don't have enough data to split into parts, then we use k-fold cross-validation. Here the data is divided into k parts, each part is used to verification while rest for training.

1.1 Model Selection

The MODEL-SELECTION algorithm helps us choose the best model by adjusting a hyperparameter called "size". The parameter "size" can imply different things in different scenarios, like tree nodes in decision tress and degree in polynomials. It starts with a simple model which unfits the data and gradually meets the requirement. In some cases, validation error forms a U-shaped curve, where it decreases first and then rises as the model overfits the training data. In other cases validation keeps decreasing like in neural networks. Overfitting is a serious problem sometimes as the model may fail to recover from overfitting.

1.2 From error rates to loss

The loss function should have correct quantification of what is expected. One perfect loss function would be :

$$L(\text{spam}, \text{nonspam}) = 1, L(\text{nonspam}, \text{spam}) = 10$$

This means it's 10 times worse to classify non-spam as spam.

Types of Loss Functions

Loss functions measure how far is the model's prediction $\hat{y}$ from the true value y. Here are three types of loss functions:

- **L1 Loss (Absolute Value Loss)**: This measures the difference between the actual and predicted values as:
$$L_1(y, \hat{y}) = |y - \hat{y}|$$
It just takes the absolute value of the difference, so it tells us how much the model is off in simple terms of distance.
- **L2 Loss (Squared Error Loss)**: Here, the difference is squared:
$$L_2(y, \hat{y}) = (y - \hat{y})^2$$
Squaring the difference means making this more sensitive to bigger mistakes.

- **0/1 Loss**: This loss function is very simple:

$$L_{0/1}(y, \hat{y}) = \begin{cases} 0 & \text{if } y = \hat{y} \\ 1 & \text{if } y \neq \hat{y} \end{cases}$$

  It only checks if the prediction is correct or not. Kind off Bernoulli Random Variable.

Expected Generalization Loss

This expected generalization loss is calculated as:

$$\text{GenLoss}_L(h) = \sum_{(x,y)\in E} L(y, h(x))P(x, y)$$

Here:

- (x, y) represents an input-output pair from a set E,
- $L(y, h(x))$ is the loss when the model h predicts $\hat{y} = h(x)$ for input x,
- $P(x, y)$ is the prior probability

The best hypothesis h^* is the one that minimizes this expected generalization loss:

$$h^* = \arg\min_{h\in H} \text{GenLoss}_L(h)$$

Where H is the set of all possible models (hypotheses).

Empirical Loss

Since the actual probability $P(x, y)$ is often unknown, the learning agent estimates the loss using the data it has. This is called *empirical loss*, which is the average loss over all the training examples:

$$\text{EmpLoss}_{L,E}(h) = \frac{1}{N} \sum_{(x,y)\in E} L(y, h(x))$$

Here, N is the number of training examples, and E is the set of those examples.

The model $\hat{h}^*$ with the minimum empirical loss is chosen as the best hypothesis based on the available data:

$$\hat{h}^* = \arg\min_{h\in H} \text{EmpLoss}_{L,E}(h)$$

The best hypothesis may differ from true function f for several reasons. If the hypothesis space H does not contain the true function, the learning problem is said to be unrealizable. Second, different training sets may lead to different hypotheses. Third, noise can cause the true function to return different values for same input. Finally, complexity may be too high which prevents the algorithm from fully exploring space H.
This balance between complexity and computation is key to successful machine learning.

1.3 Regularization

An alternative approach to model selection is where we minimize total cost which is defined as sum of empirical loss and hypothesis complexity.

$$Cost(h) = EmpLoss(h) + \lambda Complexity(h)$$

$$h^* = \arg\min_{h\in H} Cost(h)$$

Here λ is a hyperparameter that balances between empirical loss and complexity of hypothesis. This method, called regularization, penalizes complex models to prevent overfitting. Another way to simplify models is to reduce dimensions by discarding irrelevant attributes. This is called feature selection. A related concept is minimum description length(MDL) which minimizes the number of bits needed to describe both hypothesis and the data.

1.4 Hyperparameter tuning

Now the question arises what hyperparameter to choose and what value ?. When there is only one parameter then cross-validation can be used. When we have multiple hyperparameters, hand-tuning is a common approach. A more systematic method is grid search, where all combinations of possible values are tried, and then the best one is choosen.For very large search spaces, random search is an efficient alternative.Bayesian optimization treats hyperparameter tuning as a machine learning problem itself. Finally, Population-Based Training (PBT) is a method inspired by genetic algorithms. It starts with random search and trains multiple models in parallel.

Mastering machine learning isn't just about minimizing errors—it's about balancing complexity, loss, and computation to find the best fit. The key lies in thoughtful model tuning, from hand-crafted guesses to advanced optimization strategies