AIML - CS 419

Lecture 6: Wrapping up Regression and Intro to Classification

1 February 2023

Lecturer: Abir De *Scribe: Group 12*

1 Recap and Introduction

Q: How to decide which model will perform best on the Test data?
Ans: To decide which machine learning model will perform best on test data, there are several steps you can follow:

- **Data preparation:** Ensure that your data is properly preprocessed and cleaned.
- **Model selection:** Select those sets of models that are suitable for your problem and that have a good reputation for performance on similar problems.
- **Model evaluation:** Use techniques such as cross-validation to evaluate the performance of each model on the training data.
- **Model comparison:** Compare the performance of each model on the validation data and choose the one with the best performance.
- **Hyperparameter tuning:** Fine-tune the hyperparameters of the chosen model to further improve its performance.
- **Final evaluation:** Evaluate the final model on a test dataset, which should be separate from the training and validation data, to get an estimate of its performance on unseen data.

Lets take a quick look over Hyperparameter tuning and Cross Validation

1.1 Hyperparameter Tuning

- Hyperparameter tuning is the process of systematically searching for the best combination of hyperparameters for a machine learning model to optimize its performance on a given task. Hyperparameters are parameters that are set prior to training a model and have a significant impact on its performance.
- The goal of hyperparameter tuning is to find the hyperparameters that result in the best performance on the validation data. This is typically done through a search algorithm, such as grid search or random search, that explores different combinations of hyperparameters and selects the best one based on some performance metric.

1.2 Cross-Validation

- Cross-validation is a technique in machine learning used to evaluate the performance of a model. It involves dividing the available data into training and validation sets, and then training the model on the training data and evaluating its performance on the validation data. This process is repeated multiple times with different partitions of the data to get a more robust estimate of the model's performance.
- The main purpose of cross-validation is to prevent overfitting, which is when a model is too closely fit to the training data, leading to poor performance on unseen data. By evaluating the model on multiple validation sets, cross-validation provides a more realistic estimate of its generalization performance.

2 Wrapping up Regression

Let's say we have a dataset D. We now split this dataset into two parts. One is the training set which will call as $D1$ and the other is cross-validation set which we call as $D/D1$. Suppose we do this random splitting of training and cross-validation set N times such that each time we will have a different training and cross-validation set.

Dataset	Training Set	Cross-Validation Set
D	$D1$	$D/D1$
D	$D2$	$D/D2$
.	.	.
.	.	.
D	DN	D/DN

Table 1: Dataset Splitting

Let's say we have a regularised cost function that has a hyperparameter λ

$$J(w) = \frac{1}{2m} \sum_{\mathbf{x},y \in D} (w^T \mathbf{x} - y)^2 + \frac{\lambda}{2m} \sum_{j=0}^{n-1} (w_j)^2$$

Now lets calculate error on Cross Validation set

$$e(D/D_i, \lambda) = \frac{1}{2m} \sum_{\mathbf{x},y \in D/D_i} (w_{D_i\lambda}^T \mathbf{x} - y)^2$$

For hyper parameter tuning, we do a grid search over different values of λ and then take the model which gives minimum error on the Validation set. But now we have N different types of validation sets but we do hyperparameter tuning only on one validation set and choose the hyperparameter such that it gives the least error on the cross-validation set.

Q. How do we use those N-1 validation sets while doing hyperparameter tuning?
Ans:

- We first will create a list of values of hyperparameter which we are going to use for hyperparameter tuning. Assume that the list of hyperparameter λ is the following:

$$\lambda = [0.1, 0.2, ...10]$$

- Now lets say we have chosen λ to be 0.1 . For this value of lambda, we are going to train our model using different training datasets which we created earlier ($D_1, D_2...D_N$).
- After training our model with a different dataset, we will evaluate the error for each cross-validation set for λ = 0.1. We calculate the following:

$$e(D/D_i, \lambda) = \frac{1}{2m} \sum_{\mathbf{x}, y \in D/D_i} (w_{D_i \lambda})^T \mathbf{x} - y)^2$$

  When we have i = 1, means we are training the model with training dataset D_1 and then calculating cross-validation error on the cross-validation set D/D_1. Note that this we are doing for a particular value of λ, which in this case is 0.1.

- Now as we are getting N number of cross-validation errors, So we will take the expectation of all the errors for a particular value of λ, which in this case is 0.1. We will calculate the following for each value of λ which is there in our list.

$$e(\lambda) = E(e(D_i, \lambda))$$

- The best-fit lambda would be the one that will give the minimum expectation of error.

$$\lambda^* = \lambda$$

  such that we have

$$\min(e(\lambda))$$

  where λ^* is the best choice of λ

- This way we will get the best choice of λ, but now we are stuck with one problem. For a particular value of λ, we had N values of parameter **w**.

Training Set	**w**
$D1$	$w_{D_1}(\lambda)$
$D2$	$w_{D_2}(\lambda)$
.	.
.	.
DN	$w_{D_N}(\lambda)$

Table 2: Parameter

$w_{D_1}(\lambda)$ means we trained the model with the dataset D_1 using hyperparameter value λ.

- Now as we got our best fit λ to be equal to λ^*, we will have N number of values of parameter **w** to choose from. So which value of **w** should we use for our model ?
- Let's assume that we have our dataset D to be very large (roughly around 1 billion entries).

  We define a quantity w^t which is the true parameter **w** for our dataset, meaning the **w** from which the dataset was originally sampled. We define the error δ to be equal to the modulus of difference between true parameter w^t and $w_{D_i\lambda}$

  $$\delta = ||w_{D_i\lambda} - w^t||$$

  According to the maximum likelihood of estimates or the law of large numbers one can prove, that for a very large dataset

  $$||w_{D_i\lambda} - w^t|| < \epsilon$$

  Let's define another quantity Δ, which is the modulus of difference between $w_{D_m\lambda}$ and $w_{D_k\lambda}$

  $$\Delta = ||w_{D_m\lambda} - w_{D_k\lambda}||$$

  We can easily prove that if the original dataset is very large,

  $$||w_{D_m\lambda} - w_{D_k\lambda}|| < 2\epsilon$$

  This provides us with a conclusion that for any particular λ, we can choose any parameter $w_{D_i\lambda}$, as the difference between any two parameters is always less than 2ϵ. We can choose any w from the following list for the best fit λ.

  $$w = [w_{D_1\lambda}, w_{D_2\lambda}.....w_{D_N\lambda}]$$

Q. If we increase our regularizer parameter, what changes would be affected in δ and Δ ?
Ans: We can think of δ as the bias of our model as it takes the difference between true and predicted parameters. Also, we can think of Δ as the variance of the model as it takes the difference between two different parameters which were trained by two different datasets. Now if we increase the value of the regularizer, the weights of the parameter will reduce. As a result, δ will increase and Δ will decrease. This means that on increasing the value of the regularizer, bias increases and the variance decreases. This is known as the **bias-variance trade-off**.

Stability: Change in the model, when we change one point or one training sample in our dataset. On increasing the value of the regularizer, the stability of the model increases.

3 Classification

3.1 Introduction

Consider the following problem setting:

- Given data points: x_i, i = 1,2,3,....,m
- Given k possible class choices: $\{C_1, C_2, C_3, ..., C_k\}$
- We wish to estimate a mapping/classifier

$$f : X \rightarrow \{C_1, C_2, C_3, ..., C_k\}$$

that predicts class labels $\hat{y_1}, \hat{y_2}, \hat{y_3}, ...\hat{y_m}$.

This is a classification problem.
The algorithm which implements the classification on a dataset is known as a classifier. There are two types of classifiers:

- **Binary Classifier:** If the classification problem has only two possible outcomes, then it is called as Binary Classifier.
  Examples: YES or NO, CAT or DOG, SPAM or NOT SPAM
- **Multi-class Classifier:** If a classification problem has more than two outcomes, then it is called as Multi-class Classifier.
  Examples: Classifications of types of crops, Classification of types of music.

3.2 Binary classification problem

Consider a binary classification problem:

$$f(x) \in \{-1, 1\}$$

We want that any new input pattern similar to a seen pattern is classified correctly.

Linear Classification
Perceptron is an algorithm that can be used to classify data that is linearly classifiable. Perceptron function f(x) can be achieved as output by taking the dot product of the input 'x' with the learned weight coefficient vector 'w'.
Mathematically, we can express it as follows:

$$f(x) = 1; \text{ if } \mathbf{w}^\intercal \mathbf{x} + \mathbf{b} > 0$$
$$f(x) = -1; \text{ otherwise}$$

- $\mathbf{w}$ represents real-valued weights vector
- $\mathbf{b}$ represents the bias
- $\mathbf{x}$ represents a vector of input x values

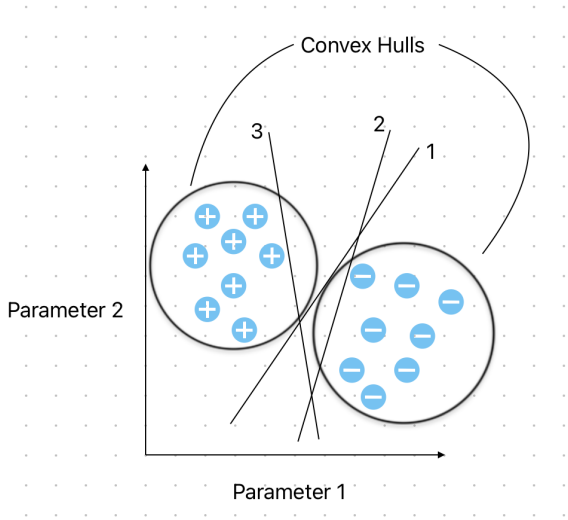

Figure 1: Example of a linear classifier

Note that perceptron does not necessarily find the best separating hyperplane, it finds any one. However, in the figure 1, we can clearly see that the hyperplane 1 is better than 2 and 3.

Support Vector Machine Algorithm

The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.

SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine. Consider the figure 2 in which there are two different categories that are classified using a decision boundary or hyperplane:

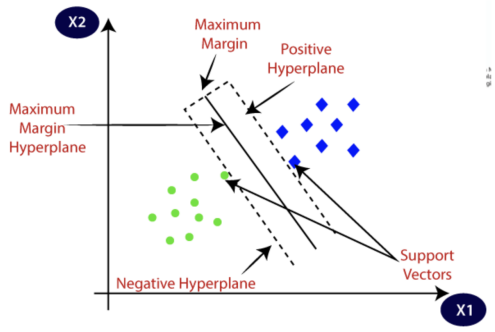

Figure 2: Support Vector Machine

Distance of the decision boundary from any point $\mathbf{x}$:

$$\Delta = \frac{|\mathbf{w}^\intercal \mathbf{x} + \mathbf{b}|}{\|\mathbf{w}\|}$$

So, we need to minimize $\|\mathbf{w}\|$ such that

$$y(\mathbf{w}^\intercal \mathbf{x} + \mathbf{b}) \geq \delta \; \forall (\mathbf{x}, y).$$

Here $y \in \{-1, 1\}$ is the label. If the point lies on the correct side of the hyperplane, $y(\mathbf{w}^\intercal \mathbf{x} + \mathbf{b})$ will be positive.
How should we select δ here? The optimization problem above becomes equivalent to

$$\begin{array}{lll} \min\limits_{\mathbf{w},\mathbf{b}} & \|\mathbf{w}\|^2 & \\ \text{such that} & y(\mathbf{w}^\intercal \mathbf{x} + \mathbf{b}) \geq 1 & \forall (\mathbf{x}, y) \end{array}$$

This is the support vector machine algorithm.