AIML - CS 337

Lecture 4: ML Terminology

11 August 2022

Lecturer: Abir De *Scribe: Akshay, Alan, Lakshya, Virendra*

Over the last two lectures, we reviewed probability and linear algebra. Now, we introduce some basic ML terminology.

1 An Example: Average Consensus

This example is in continuation from Lecture 3. We have a graph $G = (V, E)$, where $V = \{1, \dots, n\}$ is the set of vertices, and E is the set of edges. The degree of node i is given by d_i. Define the value $x_i(t)$ to be the *opinion* of node i at time t. $x_1(0), \dots, x_n(0)$ are given. For $t \geq 0$,

$$x_i(t+1) = \frac{\sum_{j \in \mathcal{N}(i)} x_j(t)}{|\mathcal{N}(i)|} \tag{1}$$

where $\mathcal{N}(i)$ denotes the set of neighbors of node i. That is, the opinion of i at time $t+1$ is average of the opinions of its neighbors at time t.

Consider $\boldsymbol{x}(t) = [x_1(t) \cdots x_n(t)]^T$. Define $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ such that $\boldsymbol{A}\boldsymbol{x}(t) = \boldsymbol{x}(t+1)$. From (1), $\boldsymbol{A}$ is a doubly-stochastic matrix, i.e., its entries are non-negative, and its rows and columns add up to 1. For instance, the first row of $\boldsymbol{A}$ is of the form $[a_{1j}]^T$, where $a_{1j} = 1/d_1$ if $j \in \mathcal{N}(i)$, else 0. Moreover, $\mathbf{1}$ and $\mathbf{1}^T$ are right and left eigenvectors of $\boldsymbol{A}$, with eigenvalue one.

We state the following result from [4]. $\boldsymbol{W}$ is the weight matrix of a connected graph.

Theorem 1.1. $\lim_{t \to \infty} \boldsymbol{W}^t = (1/n)\mathbf{1}\mathbf{1}^T$ *if and only if*

$$\mathbf{1}^T \boldsymbol{W} = \mathbf{1}^T, \tag{2}$$

$$\boldsymbol{W}\mathbf{1} = \mathbf{1}, \tag{3}$$

$$\rho(\boldsymbol{W} - (1/n)\mathbf{1}\mathbf{1}^T) < 1, \tag{4}$$

where $\rho(\cdot)$ denotes the spectral radius of a matrix, i.e., maximum of the absolute values of its eigenvalues.

For $\boldsymbol{A}$ as defined earlier, (2) and (3) are already satisfied. If (4) also holds, then for all nodes i, $\lim_{t \to \infty} x_i(t) = \frac{\sum_{j=1}^n x_j(0)}{n}$.

2 Image Classification Problem

We are given a set of images $\{I_1, \ldots, I_n\}$, each corresponding to exactly one of three animal classes: cat, dog, or tiger. The pixel data of I_i is encoded into feature-matrix $X_i \in \mathbb{R}^{d \times d}$, and IDs of the three classes are $\{0, 1, 2\}$ respectively. *Binary* classification problems involve only two labels, usually $\{0, 1\}$.

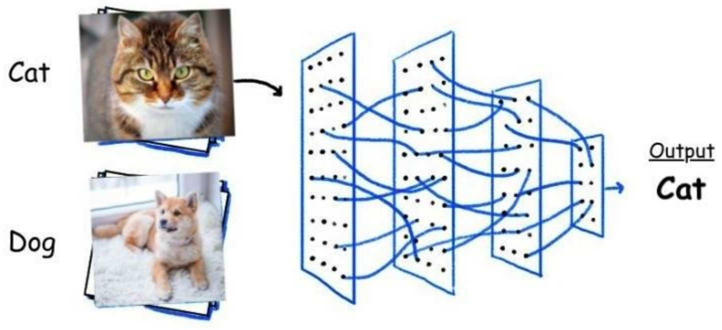

Figure 1: Dog or Cat? A binary classification example[1]

The goal is to correctly predict the label of any (unseen) image. Unseen images are inputs that were not revealed during the algorithm's development. We can't possibly write an algorithm for this prediction, since the model itself changes with the input space. Moreover, an algorithm with one-to-one mappings for the given set $\{X_i\}$ will perform poorly on unseen images. This is because the model has completely overfit to these images.

Our approach is to write a program that can design an algorithm for classifying X_is. But this program does not know what the labels represent, or how to differentiate X_is. The problem with just the set of X_i's is ill-defined. Therefore we need to teach this program using some data, which consists of a set of examples $\{X_i\}$, already mapped to corresponding y_is. This brings up the idea of a training set.

3 Training and Validation Sets

3.1 Training Set

For the classification problem, the training set consists of pairs (X_i, y_i). For each image matrix X_i, a label y_i has already been provided, and this label is assumed to be correct. Assuming binary classification, we need to find a function $H : \mathbb{R}^{d \times d} \to \{0, 1\}$, such that $H(X_i) = y_i$. As mentioned above, this function should give correct results for unseen images too.

For finding the *best* function from the space of functions, we introduce an error function L. Intuitively, $L(f)$ must be large for a function that does the classification job poorly. Then, $H =$

$arg\, min_f L(f)$.

3.2 Validation Set

How do we know that the algorithm is “ready” for use? Relying only on training set performance isn’t a good idea, since the model would have overfit on this data. Thus, we reserve a section of the training data as the validation set. After training the model (on the train set alone), its performance is evaluated on the validation set. Since the latter is unseen during training, we get a more reliable estimate of the model’s performance.

4 (Bonus) Model Selection and the Three-Way Split

In practice, we often train a variety of different models on the same data. This is possible because we have several choices for the architecture of the model and every architecture in turn has several *parameters* we can tune to improve the accuracy. Ideally we’d like to measure the performance of each of these models and choose the one that performs the best on unseen data.

That is, we first train the model on one set, then choose the model which performs best on the second set and, once we are done finalizing the model, we use a third set see how well our model generalizes on new data. The second set is called the **validation set**. The third set, which, in principle, shouldn’t be touched until we are done with the complete training of the model, is known as the **test set**.

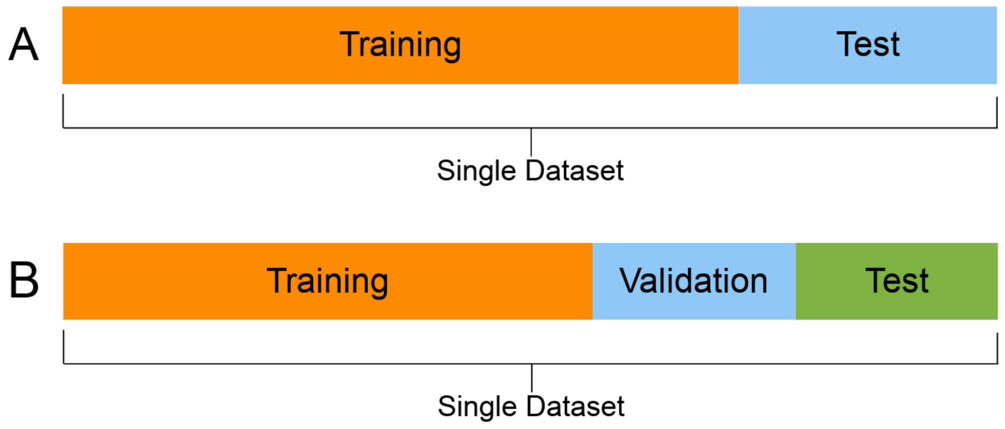

Figure 2: Three-Way Split into Training, Validation and Test Sets[2]

The fact that we were chose our model based on the second set implies that our model is *learning* the second set. This motivates the need for a third set, without which we’d never know how well our final model generalizes on unseen data because it has already ”seen” the validation set.

We talked about training, validation and test sets. But sometimes, we don't have enough data to justify a three-way split. It is in these cases that we employ a technique known as **cross-validation**. [3] is a great place to read more about it.

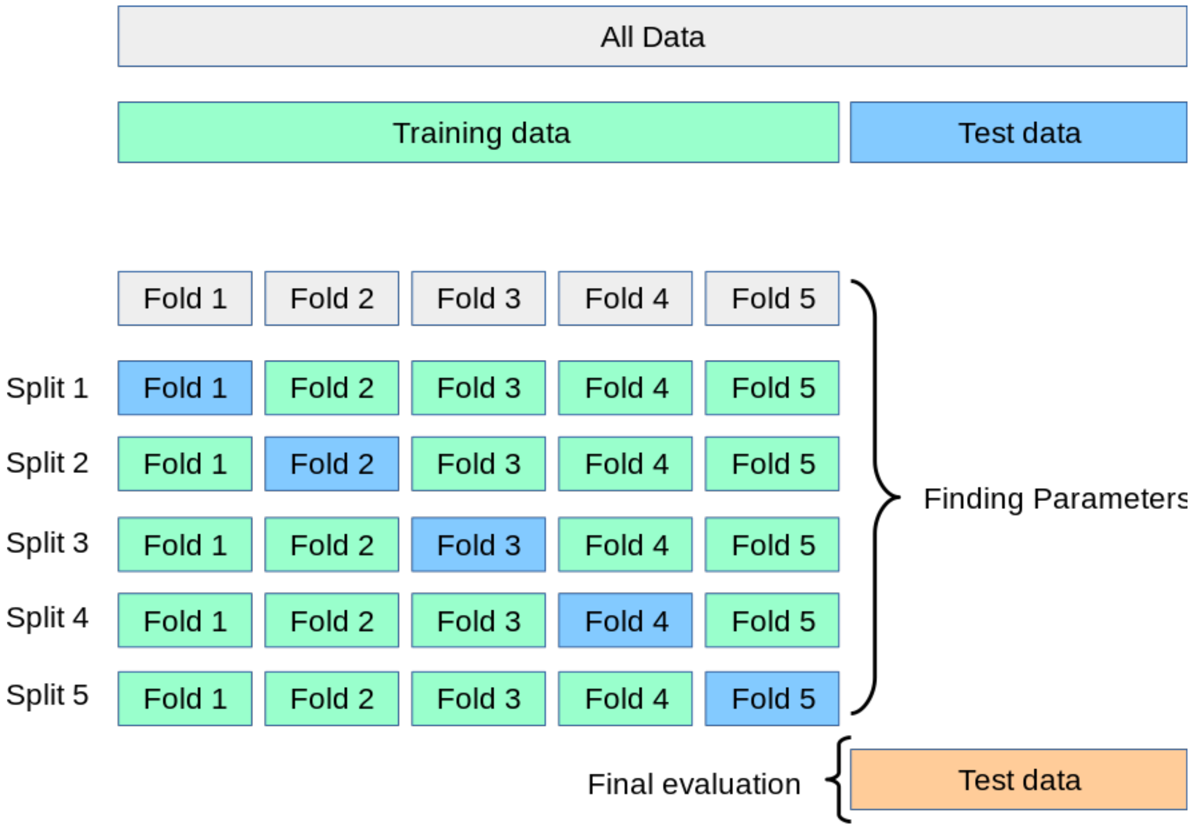

Figure 3: Cross-Validation[3]

References

[1] `https://www.researchgate.net/figure/Classification-of-two-classes-dog-an`
`fig2_353247632`.

[2] `https://commons.wikimedia.org/wiki/File:ML_dataset_training_validation_test_sets.png`.

[3] `https://scikit-learn.org/stable/modules/cross_validation.html`.

[4] L. Xiao and S. Boyd. Fast linear iterations for distributed averaging. In *42nd IEEE International Conference on Decision and Control (IEEE Cat. No.03CH37475)*, volume 5, pages 4997–5002 Vol.5, 2003.