1/14/19 Class Notes

Amy Rechkemmer

arechke@purdue.edu

1 Course Announcements

- TA Office Hours Announced - Located in HAAS G50, find times on course website
- Sign up for class Piazza
- Midterm is set for February 28th from 8-9:30 PM - Further details on course website
- Course website: `https://www.cs.purdue.edu/homes/yexiang/courses/19spring-cs578/index.html`

2 Course Project Information

There will be four components to the class project: project proposal, midterm progress report, final report, and final presentation.

The project proposal is due January 27th at 11:59 PM on CMT.

2.1 Project Objectives

1. Trying Machine Learning - everything works perfectly on paper, but not in practice!
2. Basic Scientific Training
   - Can you identify an important research question?
   - Can you justify your research question?
   - Can you write code to validate your results?

- Can you explain yourself through writing and speech?

2.2 Professor Project Tips & Suggestions

1. Apply machine learning to some application in your domain of expertise
2. Come up with a specific extension of a machine learning algorithm
3. Apply a specific module in machine learning to a new problem
   - Better sampling techniques?
   - Reinforcement learning for a specific problem?
4. The work does not need to be new!
5. The simpler the project is, the better.
6. Libraries such as sci-kit are allowed to be used.

2.3 Writing the Proposal

Evaluation Metrics:

1. Intellectual Merit - How does your proposed research advance our/your understanding of machine learning?
2. Broad Impact - How does your project make our society a better place?
3. Tractability - Is it practical for a three month course project?

Each metric needs to be explained explicitly in its own section of the project proposal. There is no page limit, but ALL text must be at least 11 point font.

3 The Machine Learning Pipeline

Important questions to answer:

- What are you learning?
- What are you trying to predict?
- Why is it important?
- What is the input/output?
- What is the functional mapping from input to output?

3.1 THE BASICS OF SUPERVISED CLASSIFICATION

We start with a set of features as **input**. Features are the data/characteristics we would like to use to make our predictions. If, for example, we wanted to predict the type of a flower, some of our features may include the color of the flower and the length of its stem.

The **output** from our model is a label. This label may fall within a set of predefined categories. In our flower example above, for the sake of simplicity, let's say that our possible labels are tulip and rose.

We would like to provide our model with the input and have it give us the output as a predicted label. Let's call x our features and $h(x)$ our output label. Our model is trying to learn a function h that will map from our features to the correct label:

$$h : x \rightarrow h(x)$$

This function h exists within a hypothesis space of possible functions that could be the best fit for our model. This is represented as $h \in \mathcal{H}$ where $\mathcal{H}$ is the hypothesis space. It is the job of our model to find the function within this space that does the best job at predicting output labels based on the features it is given. In order to train our model, specific data is given as training data that consists of features and a correct label for several instances of what we are trying to predict. Our data is given in the following form:

$$D = (x_1, y_1), (x_2, y_2), ..., (x_n, y_n)$$

where x again represents features, y represents the label associated with that set of features, and here we have n instances.

We want the function chosen by our model to have the best performance on our training set where performance is measured by how many output labels the function is able to correctly map to based on the features. We will define the **loss** function as the following:

$$L(h(x), y) = 1(h(x) \neq y)$$

meaning that the loss for a specific training example is 1 if the function's predicted output label is not the same as the example's actual output label, and 0 otherwise. The loss for the function h used for mapping is represented as the sum of the loss over all examples in the training set. This basic loss tells us how many mistakes were made in the training set we were given.

$$\min_{h} \sum_{i \in D} L(h(x_i), y_i)$$

There is an issue with this approach though. By basing function performance so heavily on training set accuracy, we are training a model that will perform very well on examples that it has already seen before in training, but will not perform so well on unseen examples. This is called **overfitting** the model because it is formed too closely to our training data and cannot generalize

for future examples.

One way we help prevent overfitting is by changing our objective from minimizing the loss function to minimizing the expectation of the loss function. In order to do so, we assume that our x and y (features and output labels) are coming from a specific underlying data distribution.

$$\min_{h} \mathbb{E}_{x,y \in Pr(x,y)} L(h(x), y)$$

3.2 The Correct Way to Train, Validate, and Test

We have spoken of the importance and use of having training data in order to train your model. We additionally want to have a testing set that can be used to evaluate how well our model works on data that it has not yet seen. Is this all that we need?

The answer is NO. In future lectures, we will see that our functions will contain hyperparameters, which are function parameters that are set by us. These parameters include deciding which features will be selected as being the best for making accurate predictions, and we will talk about others later. It is tempting to train a model with the training data set, test it with the testing data set, see that the model's performance was poor, and tweak the parameters to see if that helps improve it. However, the testing data set should only be utilized once at the very end, as we do not want that data influencing the final state of our model.

Here is what we want to do instead. We want to have a separate data set called our validation data set. This means that over our entire data set, we have partitioned it three ways: training, validation, and testing. We will train with the training set, test our model with different hyperparameters to find the best combination that gives us the smallest loss, retrain the final model on both the training and validation sets, and then test using the testing data set. The following is the procedure for correctly training, validating, and testing your model:

For each function h:
    For each parameter:
        Train using training data set
        Evaluate on validation data set
Select function and parameters that perform best on validation set
Use the model to fit both training and validation data sets
Test final model on the testing data set