Lecture Notes on

Machine Learning

Will Lancer

will.m.lancer@gmail.com

Notes on machine learning.

Contents

1 Kaggle and Google’s Introduction to Machine Learning

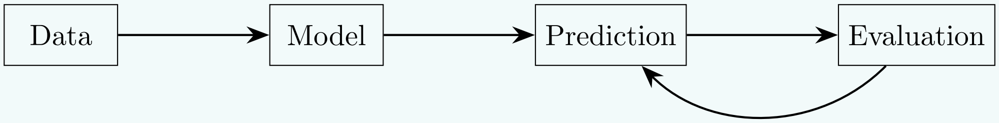
See the notes on Python to refresh on the necessary NumPy and Pandas to follow the example below.

Idea 1.1 (Basic terminology)

Your data points are called **examples**, your parameters are called **features**, and your desired prediction parameter is called the **label**. There are a few distinctions in machine learning:

- **Supervised learning** vs. **unsupervised learning**. Supervised just means you give the model the right answer in the end, and unsupervised means you don’t (doing that may not even be well-defined).
- Supervised learning: **regression** vs. **categorization**. Regression predicts a numerical value and categorization predicts a likelihood that the label is in a given category. You can have binary or multi-category categorification.
- **Reinforcement learning** (broader than supervised/unsupervised). Gives the model rewards/punishments based on actions peformed within its training environment.

The basic ML workflow is



where your prediction and evaluation cycles continue ad infinitum.

Example 1.1 (The decision tree)

This extended example is Kaggle’s intro course. It will help us get familiar with the general process before going into deeper theory. We first need to make our data into a DataTable to do analysis on it. We can import it from a csv by using `df = pd.read_csv(dataFilePath)`. We look through our data using `df.describe()` and `df.head()`. We now need to specify featurese and a label. We do this by

```
features = [’column1’, ’column2’, ..., ’columnN’]
# ’X’ is the standard name for the vector of feature data
X = df[features]
# ’y’ is the standard name for the label vector
y = df[’labelColumn’]
```

Then we use some machine learning magic to train this data on this data set. For the decision tree, we import the decision tree trainer and then run it on the data,

```
from sklearn.tree import DecisionTreeRegressor
```

We then declare a decision tree regression model and train it on our data,

```
decisionTree = DecisionTreeRegressor(random_state = 1)
decisionTree.fit(X, y)
```

Now we can use this to predict things, so we can say things like `decisionTree.predict()` or `decisionTree.predict(X.head())` to get predictions. You can optimize the number of leaves in your decision tree by minimizing the **mean absolute error**, which is defined as

$$\text{MAE} = \frac{1}{N}\sum_{i}^{N} |\mathbf{y}_{\text{train}} - \mathbf{y}_{\text{val}}|.$$

You can import the mean absolute error module from `sklearn.metrics` like before, `import mean_absolute_error`. You can also import a train-test-split tool from `sklearn.model_selection` to get some validation data from your training set.

That's it! That's our first basic ML model.