Lecture Notes on

Machine Learning

Will Lancer

will.m.lancer@gmail.com

Notes on machine learning. Each section is another ML course or paper that I did. Links to the resources are included section by section. Then there is my organization of the content as a whole (coming after I familiarize myself with the landscape of AI/ML).

Contents

1 Kaggle's intro course 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

$$\begin{tikzcd}\boxed{\text{Data}} \arrow[r] & \boxed{\text{Model}} \arrow[r] & \boxed{\text{Prediction}} \arrow[r] & \boxed{\text{Evaluation}} \arrow[l, bend left]\end{tikzcd}$$

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 features 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. Let's summarize what we did in more abstract terms now:

- We had a given cleaned data set to work with.
- We set up our data for processing by turning it into a DataTable.
- We used a given algorithm to train our model on this data.
- We tested our model on some validation data.
- We further refined this model by giving it an optimization condition—in this case, we were minimizing the L^1 norm (MAE) as compared to the validation data.

From this I will remind ourselves of a meta-level heuristic from King Ilya:

> *The model just wants to learn.* All we did was set up data-processing, say which algorithm we were using, and get out of the model's way. Then it tried its best to learn given the training data we gave it. After checking its progress, we then gave it another learning tool (the MAE); this gives the model a more precise way of telling whether things are right or wrong (i.e. there is some measure of magnitude which isn't present in a binary classifier). The model did everything difficult here: we just helped it along its way.

A useful way of thinking about the difference between parameter correction and model correction.

Example 1.2 (Inner-loop vs. outer-loop learning, and a physical analogy)

The *inner loop* is the model learning by itself, and the *outer loop* is outside sources telling the model other ways to learn. This is the distinction between the model training itself on some set of training data vs. the validation and correction phase of learning. The model is doing inner loop learning, then you come in and do outer loop learning. Of course, human intervention is not necessary—one could imagine another AI telling the model to change something. But something has to come in and help this model do outer loop learning. There is some sense of discontinuity with model correction vs. parameter correction.

An analogy is electrostatic equilibrium: if you place two charged conductors in the vicinity of one another, they will continuously relax into equilibrium by minimizing $\mathcal{E} = 1/8\pi \int d^3\mathbf{r}'\, \mathbf{E} \cdot \mathbf{D}$. This is parameter correction. Model correction would be you changing the material of the conductor. There is no continuous sense in which this occurs: you just change it and let things re-run.

Another analogy is a kid learning how to ride a bike. The kid continuously adjusts their strategy in response to the optimization condition of balancing (i.e. keeping the COM over the supports): this is parameter adjustment. Then let's say the kid's dad comes in and gives him a smaller bike (he determined that the one he was riding was too big) and tells him to not try and pedal so fast (the kid was always trying to go to fast and thus was losing his balance too quickly), and tells him to go out there and try again. This is model correction.

2 CS 231n

This section of the notes follows Karpathy's CS 231n course. This is my first bonafide ML course.