lecture_1-jan_22-overview_of_machine_learning

January 21, 2025

1 Lecture 1 (Jan 22): Overview of Machine Learning

Outline

- Datasets
  - A word on datasets
  - Two interesting datasets
  - Types of data
  - Problems of prediction
- Framework of (supervised) machine learning
  - Stating the problem
  - Evaluation metrics
  - What is a ML model?
  - Model parameters
    * Emphasize the presence of the noise term, which distinguishes ML from function approximation in (say) physics.
    * Mention some commonly used models (linear, logistic, k-nearest neighbors, decision tree, neural networks)
    * Explain what model parameters are.
  - Metrics for measuring goodness of predictions
    * Accuracy and MSE as metrics for classification/regression.
    * Using sklearn, create simple logistic/linear models for the two datasets.
    * Fit the models to the train sets.
    * Import the test sets, make predictions using models for the two datasets, compute the metrics.
  - Model fitting, I.e. finding the parameters which give the best predictions on the given dataset.
    * Loss functions, I.e. expressing (directly or indirectly) the scoring metric as a function of the model parameters.
    * Fitting = finding parameters that minimize the loss function on the dataset. Emphasize that if rows are added/removed then loss function changes, so the model fit depends closely on the data available to you.
  - Summarize a typical ML pipeline (for making a model at home, not for business or large-scale deployment): Data Collection > Data processing > Exploratory Data analysis > Feature engineering and Model selection > Model fitting > Making predictions on unseen data.
- Neural networks
  - What exactly is a neural network? A: a ML model that generalizes a lot of commonly

used models (in particular, linear/logistic).
- Neurons, activation functions, layers
- Types of NN architectures:
  * Perceptron.
  * Multilayer perceptron/ feed-forward network.
  * Recurrent neural networks (such as LSTM).
  * Convolutional neural networks.
  * Transformer.
- Why do we use neural networks?
  * (Math) Universal approx. theorem guarantees that (almost) any continuous function can be approximated with arbitrary precision by a neural network... (caveat: might need to make the network super complicated/require a lot of memory/computations).
  * (Computing) GPU clusters allow enormous models to be trained rapidly on enormous datasets. Mention the usefulness of automatic differentiation in model training (backpropagation).

1.1 Datasets

1.1.1 A word on datasets

I want to begin by offering a framework for viewing data and datasets. This section is informal, partly opinion-based, and not very rigorous, so be warned!

Data arises naturally when you ask a question about an *object*, which (typically) refers to something physical, like *flower*, *human*, *country*, *house*, *car*, *concrete mix*, and so on. Note that each of these objects is in fact referring to a *collection* of things. For example, *flower* refers to the collection of all flowers. The individual members of the collection are called *instances* of the object. For example, I am an instance of the object *human*, and you are also an instance of *human* (or perhaps *AI*).

"Asking a question about an object" simply means that you ask a question about every instance of the object. For example, if the object is *human*, you can ask *What is the height*? If the object is *house*, you can ask *What is the square footage*? Typically (as is the case with these examples), the answer to your question will vary as you vary the instance– it is a variable! In fact, when we make things more formal using probability theory, we'll call them *Random Variables*.

Any question about an object is called a *feature* of the object, because it usually refers to some kind of natural attribute of the object (e.x. height of a human, square footage of a house). Now, the main point to keep in mind about datasets is:

- *Columns correspond to features*: each column contains answers to a single question about the object for all the instances (being considered).
- *Rows correspond to instances*– each row contains answers to all the questions (being considered) about a particular instance.

For example, the questions *What is the age* and *What is the date of birth* define features of the object *human*. If we have 100 people, numbered $0, \ldots, 99$, for whom we know the answers to these two questions, then we can assemble these answers into a single dataset:

- There will be two columns, which we can name (for example) `age` and `data_of_birth`.
- There will be 100 rows, indexed (labelled) by $0, \ldots, 99$.

- The row with index i will have the age and date of birth of the i-th person, respectively.

1.1.2 Two interesting datasets

Let's start by reading in two interesting datasets (sourced from the UCI ML repo). - The first is the famous `iris` dataset, a small and simple dataset containing measurements of three types of iris flower. - The second dataset `real_estate` contains various characteristics of houses.

[21]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Load the data
iris = pd.read_csv('../data/classification/iris/train.csv')
real_estate = pd.read_csv('../data/regression/real_estate_valuation/train.csv')

print(f'iris: {iris.shape[0]} rows and {iris.shape[1]} columns')
print(iris.head())
print()
print(f'real_estate: {real_estate.shape[0]} rows and {real_estate.shape[1]}␣
↪columns')
print(real_estate.head())
```

```
iris: 120 rows and 5 columns
   sepal length  sepal width  petal length  petal width            class
0           4.4          2.9           1.4          0.2      Iris-setosa
1           4.9          2.5           4.5          1.7   Iris-virginica
2           6.8          2.8           4.8          1.4  Iris-versicolor
3           4.9          3.1           1.5          0.1      Iris-setosa
4           5.5          2.5           4.0          1.3  Iris-versicolor

real_estate: 331 rows and 7 columns
   X1 transaction date  X2 house age  X3 distance to the nearest MRT station  \
0             2013.417          35.3                               614.13940
1             2013.083           6.2                                90.45606
2             2013.500          23.0                              3947.94500
3             2013.167           1.1                               193.58450
4             2013.417          17.1                               967.40000

   X4 number of convenience stores  X5 latitude  X6 longitude  \
0                                7     24.97913     121.53666
1                                9     24.97433     121.54310
2                                0     24.94783     121.50243
3                                6     24.96571     121.54089
4                                4     24.98872     121.53408

   Y house price of unit area
```

```
0                        33.1
1                        58.0
2                        25.3
3                        48.6
4                        40.0
```

1.1.3 Types of data

Observe that the features in these datasets fall into two categories.

1. *Continuous*:
    - These are features whose values are real numbers that (may) include decimal places, that is, the values are of type `float`.
    - E.g. all columns of the `iris` dataset except for the `class` column, and all columns of the `real_estate` dataset except for `X4 number of convenience stores`.
    - Mathematically, the variable corresponding to the column is called a *continuous random variable*.
2. *Categorical*:
    - These are features whose values are contained in a finite set.
    - The values may be numeric (e.g. `X4 number of convenience stores`is of type `int`) or non-numeric (e.g. `class` is of type `str`).
    - Mathematically, the variable corresponding to the column is called a *discrete random variable*.

Remark. There is a potential grey area where a variable could be viewed as either continuous or categorical; e.x. if the object is *US county* and the feature is *population*, then (in principle) any positive integer is a possible value, but these values are discrete… Let's ignore this subtlety for now!

1.1.4 Problems of prediction

Problems in ML are problems of prediction. Namely, one has an object and a certain distinguished feature Y, called the *target variable*, and given an instance, one wants to predict the value of the target. Such problems are of two kinds, depending on the nature of the target Y:

- *Classification.*
    - This is when Y is a categorical variable with finitely many possible values $c_1, \dots, c_r$.
    - The values of Y partition the instances into r classes.
    - Predicting Y amounts to classifying the instances, i.e. determining which class the instance lies in.
    - For example, a natural target for `iris` is `class`; given a particular iris flower, one wants to classify it as belonging to one of the three classes (`Iris-setosa`, `Iris-virginica`, or `Iris-versicolor`).
- *Regression.*
    - This is when Y is a continuous variable with real values.
    - One wants to predict the value of Y as closely as possible.
    - For example, a natural target for `real_estate` is `Y house price of unit area`; given a particular house, one wants to predict the price per unit area.

Notice that if we are looking at an instance that's already in the dataset, then there is nothing to predict! Thus, the goal is to look at instances for which we *don't* know the value of the target, and we want to predict it. For example, we might want to predict the targets that are missing from the following datasets.

[22]:
```
iris_test = pd.read_csv('../data/classification/iris/test.csv')
real_estate_test = pd.read_csv('../data/regression/real_estate_valuation/test.
 ↪csv')

print(f'iris_test: {iris_test.shape[0]} rows and {iris_test.shape[1]} columns')
print(iris_test.head())
print()
print(f'real_estate_test: {real_estate_test.shape[0]} rows and␣
 ↪{real_estate_test.shape[1]} columns')
print(real_estate_test.head())
```

```
iris_test: 30 rows and 4 columns
   sepal length  sepal width  petal length  petal width
0           4.4          3.0           1.3          0.2
1           6.1          3.0           4.9          1.8
2           4.9          2.4           3.3          1.0
3           5.0          2.3           3.3          1.0
4           4.4          3.2           1.3          0.2

real_estate_test: 83 rows and 6 columns
   X1 transaction date  X2 house age  X3 distance to the nearest MRT station  \
0             2013.083          33.0                                181.0766
1             2012.917          16.9                                964.7496
2             2012.917          31.9                               1146.3290
3             2013.083          17.5                                395.6747
4             2013.500          11.8                                533.4762

   X4 number of convenience stores  X5 latitude  X6 longitude
0                                9     24.97697     121.54262
1                                4     24.98872     121.53411
2                                0     24.94920     121.53076
3                                5     24.95674     121.53400
4                                4     24.97445     121.54765
```

The first dataset `iris` is a *labelled* dataset; it contains the value of the target `class` for all instances. The second one `iris_test` is *not labelled*; the value of `class` is not known for any instances. Machine Learning falls, broadly speaking, into one of two types: - Supervised Machine Learning seeks to train models on labelled datasets like `real_estate`, so that they can make predictions on unlabelled datasets like `real_estate_test`. - Unsupervised Machine Learning… is postponed until later.

In the next section, we move onto a more precise discussion of the framework of supervised machine learning.

1.2 Framework of supervised machine learning

1.2.1 Stating the problem

Suppose we have a labelled dataset `train` with features $X_1, \dots, X_n$ and target Y, and unlabelled dataset `test` with only the features $X_1, \dots, X_n$. Machine Learning begins with:

Assumption 1. *There exists some "ground truth function"* $\mathbf{F}$ *such that*

$$Y = \mathbf{F}(X_1, \dots, X_n) + \epsilon, \tag{1}$$

where ϵ *is a small "noise" term to account for randomness inherent in the problem.*

If we "know" the one true function $\mathbf{F}$, then predicting Y on the unlabelled dataset is easy: we simply plug in the values of the features into $\mathbf{F}$, and the output will be our predicted value of Y (note: by assumption, we will typically have a small error ϵ). The "learn" in machine learning comes from the following reformulation of our problem of prediction:

Goal 1. *Compute (a good approximation to) the function* $\mathbf{F}$. *That is, use the labelled dataset to learn the function* $\mathbf{F}$.

Remark. Note that, if we want to be able to glean $\mathbf{F}$ by looking at the labelled dataset, we require that there be enough rows to actually construct a good approximation to $\mathbf{F}$ which *generalizes* well to instances outside the dataset. More on this later.

1.2.2 Evaluation metrics

Suppose we have managed to determine a function F that (we believe) approximates $\mathbf{F}$ well, and we write a program to implement this function. Our program accepts a dataset (the test set) with columns $X_1, \dots, X_n$ and m rows (instances), and outputs a column vector of predictions $y_{\text{pred}} \in \mathbb{R}^m$.

In practice, the test set will also be labelled (more on this later). For example, running the code

```
iris_test_full = pd.read_csv(../data/classification/iris/iris_test_private.csv)
```

will create a dataframe `iris_test_full` with the same data as `iris_test` along with the target `class` column. Our claim that F is a good approximation to $\mathbf{F}$ is judged by how closely our prediction vector $y_{\text{pred}} \in \mathbb{R}^m$ aligns with the vector $y_{\text{true}} \in \mathbb{R}^m$ of true values in the test set. This measure of closeness is made quantitative and rigorous in the form of a **scoring function**, which associates to the pair $(y_{\text{pred}}, y_{\text{true}})$ some kind of numerical score $\text{Score}(y_{\text{pred}}, y_{\text{true}})$.

- For classification tasks, a simple (but often not very informative) scoring function is simply *accuracy score*: it is the fraction of instances that were correctly classified, divided by the total number of instances:

$$\text{Accuracy}(y_{\text{pred}}, y_{\text{true}}) = \frac{\text{Number of entries where} y_{\text{pred}} = y_{\text{true}}}{m}.$$

- For regression tasks, a simple (and perhaps the most widely used) scoring function is *Mean-squared error*: it is the square of the distance between the vectors y_{true} and y_{pred} in $\mathbb{R}^m$, and divided by m to (in some sense) account for the fact that there are m entries:

$$\text{MSE}(y_{\text{pred}}, y_{\text{true}}) = \frac{||y_{\text{pred}} - y_{\text{true}}||^2}{m}.$$

1.2.3 What is a ML model?

OK. We want to learn this hypothetical function $\mathbf{F}$? How? In fact, what does it even mean to "know" the function $\mathbf{F}$? The answer is tautological: knowing the function means that, given an input, we can compute the output. That is, knowing $\mathbf{F}$ means that we know the *formula* which defines $\mathbf{F}$, if such a formula at all exists, or more generally, we know an algorithm for computing outputs of $\mathbf{F}$.

Since there is a vast, vast jungle of possible functions, our starting point is always to first study the labelled dataset to explore any visible patterns in the data. A thorough and fruitful exploration of the data will allow us to narrow our search by making:

Assumption 2. *There is a certain class of functions $\mathcal{C}$ (called a **model**) such that some member $F \in \mathcal{C}$ is a good approximation to $\mathbf{F}$.*

The word model above is used in the mathematical sense; it refers to a family of functions that are all similar in some respect, for example, linear, logistic, exponential, polynomial, trigonometric, and so on. We also call them **machine learning models**.

Now, given assumption 2, we have a model $\mathcal{C}$ consisting of candidates for our approximation of $\mathbf{F}$. Then, our problem is re-phrased as:

Goal 2. *Produce the particular function $\widehat{F}$ in the chosen model $\mathcal{C}$ which best approximates $\mathbf{F}$ among all functions in $\mathcal{C}$.*

1.2.4 Model parameters

The examples of models given above are all **parametric models**: each model consists of functions having the same type of formula. These formulas involve constants, which are called **parameters**, and varying the parameters results in varying the function within the model.

Suppose for example that we have just one feature X and one target Y, both continuous. The simplest possible model is the *linear model*:

$$\mathcal{C}_{linear} = \{\text{Functions of the form } F(X) = mX + b\}.$$

The model parameters here are pairs of real numbers m and b, which can be grouped together and viewed as a single point $\theta = (m, b) \in \mathbb{R}^2$. This point corresponds to the linear function $F_\theta(X) = mX + b$. For example, the point $\theta = (1, 1)$ corresponds to $F_\theta(X) = X + 1$. In fact, we have a bijection

$$\begin{aligned} \mathbb{R}^2 &\leftrightarrow \mathcal{C}_{linear}, \\ \theta &\mapsto F_\theta(X). \end{aligned}$$

So, we want to think of the model $\mathcal{C}_{linear}$ itself as $\mathbb{R}^2$, which is typically referred to as the *parameter space*. Goal 2 of finding the best approximation to $\mathbf{F}$ within $\mathcal{C}_{linear}$ amounts to finding a special point $\widehat{\theta}$ in the parameter space. In general, we can reformulate our goal as:

Goal 3. *Produce the particular value $\widehat{\theta}$ of the parameters such that the corresponding function $F_{\widehat{\theta}}$ is the best approximation to $\mathbf{F}$ within the model $\mathcal{C}$.*