Machine Learning for Engineers (ME644)

© Malay K. Das, 210 Southern Lab, mkdas@iitk.ac.in

Office hours: W 1030-1130, SL-210

Today: Introduction to Machine Learning

k-nearest neighbors

If you have already taken a ML course from another department, please drop this course; otherwise, you will be deregistrared from this course

1

we always use **models** for prediction

input data → model → output (prediction)

Example

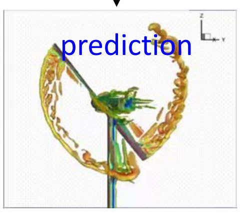
$\delta Q, dU$ (input) → $\delta W = \delta Q - dU$ (model) → δW (prediction)

Example: wind turbine

input → model:

$$\nabla \cdot \mathbf{u} = 0$$
$$\mathbf{u}_t + \mathbf{u}\nabla\mathbf{u} = -\nabla p/\rho + \nu\nabla^2\mathbf{u}$$

→ solution algorithm → prediction



Often, we do not have a well-accepted mechanistic model, we only have some (mostly experimental) data

Machine Learning (ML) could be useful in such cases

2

Machine Learning

prediction improves with experience

computer **program**

Definition (T. M. Mitchell, Machine Learning, 1997)

A **computer program** is said to **learn** from experience E with respect to some class of task T and performance measure P, if its performance at task T, as measured by P, improves with experience E

Machine learning may be useful when

- We don't have a model to describe a process
- We have an incomplete model
- We have a model but it is too complicated to solve
- We have a model but it is ill-posed

We need **data** to compensate for these lacuna; machine learning comes as a useful tool in such cases

Prediction: from traditional modeling	**Prediction: from machine learning**
Models are primarily mechanistic (developed from some mechanism)	Models are primarily empirical (data-driven); may have some physical basis (bias)
Problem (we are trying to solve) is the input; prediction is the output	Problem and known problem/solutions are the input; prediction is the output
Models are rigid; algorithm, boundary conditions, properties may vary	Models are flexible; change with experience (especially bad predictions)
Mostly deterministic	Mostly probabilistic

Machine learning is closely tied with **mathematics**

- linear algebra, vector calculus
- optimization
- probability, statistics (not much in ME644)

Programming is essential

In this course

- Mathematics for machine learning
- Supervised learning: Finding label of test data using feature + labels of training data
- Unsupervised learning: Finding structure in training data (unlabeled)
- Artificial Neural Network (ANN): A powerful technique in ML

Supervised and unsupervised learnings will be of major importance in this course, please go through the course policy, and the course syllabus for further details

Important information

- **Weekly Quiz (10mins, open book/notes) every Wednesday, in-class**

- **Weekly HW every Friday,** most HWs will require programming

- **Computing Quiz (~1 hr., in-class),** will require programming

- No specific textbook, we will use all the books listed in the handout (and few more); will try to upload lecture notes as much as possible, **rely on creating your own notes**

- The course is 'outcome' oriented; carefully read the **'course outcome'**

Example: Should I registrar for the **machine learning** course or not

model

input: you → ?? → solution algorithm → prediction: yes or no

Inputs are described by **features**

Outputs are known as **labels**

Output: student's perceived grade B or higher → yes (take the course) label $= 1$

No, otherwise label $= -1$

Inputs (features): Current CPI, hours of sleep the night before exam

2D vector (a point in a 2Dplane, called feature space)

7

We now use experimental data (history from previous students)

We wish to estimate $\widehat{y}_\star(\mathbf{x}_\star)$

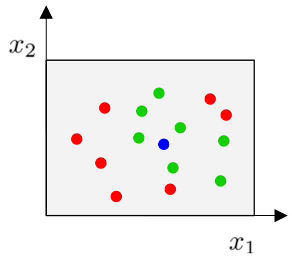
$\mathbf{x} = [x_1 \quad x_2]^T$

$\{(\mathbf{x}_i, y_i)\}_{i=1}^n$ training data

$\equiv 1$

$\equiv -1$

test data $\mathbf{x}_\star$



Fundamental assumption: if $\mathbf{x}_\star$ is close to $\mathbf{x}_i$ then $\widehat{y}_\star(\mathbf{x}_\star)$ is close to y_i

Let us define: $d_i = \|\mathbf{x}_i - \mathbf{x}_\star\| \quad i = 1, 2, \cdots, n$ 'distance' between $\mathbf{x}_i, \mathbf{x}_\star$

1-nearest neighbor algorithm: $\widehat{y}_\star(\mathbf{x}_\star) = y_j \qquad j = \arg\min_i d_i$

'label' of test data is same as the label of 'closest' training data

Problem: too much randomness, one new data can completely reverse the prediction

8

$d_i = \|\mathbf{x}_i - \mathbf{x}_\star\| \qquad i = 1, 2, \cdots, n$

1-nearest neighbor (1-NN) algorithm:

$\widehat{y}_\star(\mathbf{x}_\star) = y_j \qquad j = \arg\min_i d_i$

if 'nearest' $y_i = 1$ then $\widehat{y}_\star(\mathbf{x}_\star) = 1$

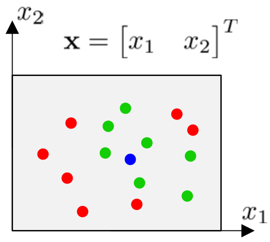
else $\widehat{y}_\star(\mathbf{x}_\star) = -1$

$\{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$ training data $\begin{cases} \bullet \equiv 1 \\ \bullet \equiv -1 \end{cases}$

• test data $\mathbf{x}_\star$

our estimate $\widehat{y}_\star(\mathbf{x}_\star)$

$\mathbf{x} = [x_1 \quad x_2]^T$



High randomness of 1-NN may be tamed by the following algorithm

if $\sum_i^n y_i \geq 0$ then $\widehat{y}_\star(\mathbf{x}_\star) = 1$

else $\widehat{y}_\star(\mathbf{x}_\star) = -1$

The case of $\sum_i^n y_i = 0$ can go wither way

we make a choice (bias)

bias isn't arbitrary; comes from domain knowledge

Too general, output does not depend on test input at all

We seek the middle ground

***k*-NN (*k*-nearest neighbors)**

$d_i = \|\mathbf{x}_i - \mathbf{x}_\star\| \qquad i = 1, 2, \cdots, n$

Define an integer *k* $\quad 1 \leq k \leq n$

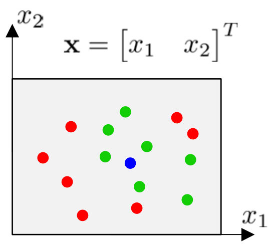
such that $d_j < d_i$ for any $j = 1, 2, \cdots, k \quad i = k+1, k+2, \cdots, n$

$\{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$ training data $\begin{cases} \bullet \equiv 1 \\ \bullet \equiv -1 \end{cases}$

• test data $\mathbf{x}_\star$

our estimate $\widehat{y}_\star(\mathbf{x}_\star)$

$\mathbf{x} = [x_1 \quad x_2]^T$



if $\sum_i^k y_i \geq 0$ then $\widehat{y}_\star(\mathbf{x}_\star) = 1$

else $\widehat{y}_\star(\mathbf{x}_\star) = -1$

Our first ML algorithm!!

The first ML algorithm; proposed by Alhazen (1030)

Problems: finding *k*, validation, uncertainty quantification

How to ensure that our model is correct

Our model cannot guarantee correct prediction always, what to do about that