Machine Learning for Engineers (ME644, hello.iitk.ac.in)

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

Office hours: W 1030-1130, SL-210

Previously:

Introduction, course policy, kNN

Today: review, kNN

Lecture notes are copyrighted; do NOT share without written permission from the instructor; check the course website everyday for notes/resources/assignments. Please read the 'course outcome' to see if this course is right for you.

If you have already taken a ML course from another department, please drop this course; otherwise, you will be deregistrared from this course; auditing this course is not permitted, you may credit in S/X mode.

Machine learning predicts using data; used when mechanistic models are not available

Input data are described by **features** (usually a feature vector); output is called **label** (usually a scalar, integer or real)

Machine learning

- Supervised learning: prediction without a mechanistic model
  - regression
  - classification
- Unsupervised learning: finding structure in training data
  - clustering
  - dimensionality reduction

Other paradigms of machine learning are beyond the scope of this course

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

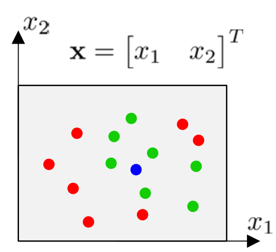
2D vector (a point in a 2Dplane, called feature space)

We use **kNN** (**k-nearest neighbors)**

$\{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$ training data: green $\equiv 1$, red $\equiv -1$

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

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



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

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

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

if $\sum_{i}^{k} y_i \ge 0$ then $\widehat{y}_\star(\mathbf{x}_\star) = 1$ else $\widehat{y}_\star(\mathbf{x}_\star) = -1$

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

'Distance' in kNN

In general, distance between two mD vectors, may be defined as $d_i = \|\mathbf{x}_i - \mathbf{x}_\star\|_p$

$$= \left(\sum_{j=1}^{m} |x_{ij} - x_{\star j}|^p \right)^{\frac{1}{p}}$$

$p \geq 1 \quad p \in \mathbb{R}$

Consider a 2D feature space $\mathbf{x}_i = \begin{bmatrix} x_{i1} & x_{i2} \end{bmatrix}^T$

$$d_i = \left(|x_{i1} - x_{\star 1}|^p + |x_{i2} - x_{\star 2}|^p \right)^{\frac{1}{p}}$$

for kNN, and in many other ML algorithms, we commonly use $p = 2$

Thus $d_i = \|\mathbf{x}_i - \mathbf{x}_\star\|_2 = \sqrt{(x_{i1} - x_{\star 1})^2 + (x_{i2} - x_{\star 2})^2}$ called 2-norm or l_2 norm

or Euclidian norm

or Euclidian distance

If $(x_{i1} - x_{\star 1})$ and $(x_{i2} - x_{\star 2})$

are of different **order,** one feature may unphysically dominate over other

for instance $(x_{i1} - x_{\star 1}) \sim \mathcal{O}(.01), (x_{i2} - x_{\star 2}) \sim \mathcal{O}(100) \quad \Rightarrow d_i \sim (O)(x_{i2} - x_{\star 2})$

5

Scaling (Feature normalization) $\mathbf{x}_i = \begin{bmatrix} x_{i1} & x_{i2} \end{bmatrix}^T$

We can scale the feature between $0,1$

$$x_{i1} \leftarrow \frac{x_{i1} - x_{1,\min}}{x_{1,\max} - x_{1,\min}}$$

$$x_{i2} \leftarrow \frac{x_{i2} - x_{2,\min}}{x_{2,\max} - x_{2,\min}}$$

$i = 1, 2, \cdots, n$

n: no. of training data

another popular option is to create zero mean, unit standard deviation data

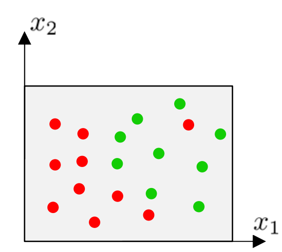
$$x_{i1} \leftarrow \frac{x_{i1} - \overline{x}_1}{\sigma_1} \qquad \overline{x}_1 = \frac{1}{n} \sum_{i=1}^{n} x_{i1} \qquad \sigma_1^2 = \frac{1}{n} \sum_{i=1}^{n} (x_{i1} - \overline{x}_1)^2$$

Balancing All classes must have comparable representations

nos. of red and green dots should be close to each other

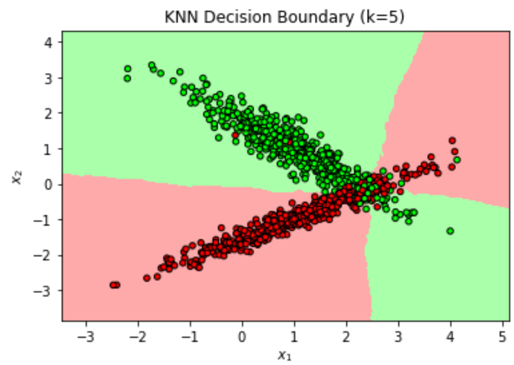
Outlier removal

red point surrounded by green (or vice versa) is an outlier



6

Decision boundary



Compute labels for various test data

We get a contour of labels, known as **decision boundary**

Prediction is quicker, once we have the decision boundary, until we change the training set

Computing decision boundaries with various k values provides more insight about the physical problem

Weighted kNN (an important variation of kNN)

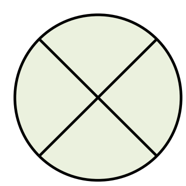
Standard **kNN** decides label of test data based on what majority of neighbors votes

weighted **kNN** puts more weightage on the close neighbors' votes

K-fold Cross Validation

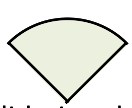
Split the training data in K equal (or nearly equal) blocks

Use K-1 block for training, 1 block for testing (validation)

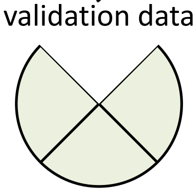


example: $K = 4$

Fold 1

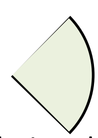


validation data

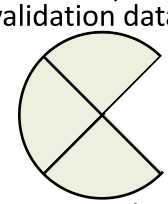


training data

Fold 2



validation data



training data

Errors in each fold

$$e_{ij} = \frac{q_{ij}}{p_{ij} + q_{ij}} \qquad j = 1, 2, \cdots, K$$

no. of correct predictions

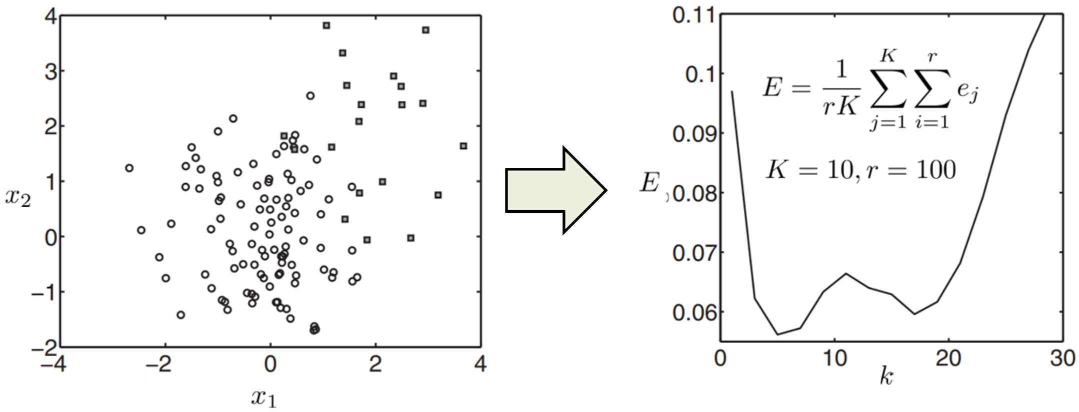
no. of incorrect predictions

$i = 1, 2, \cdots, r$

the calculation is repeated r times by changing the partition

Average error $E = \frac{1}{rK} \sum_{j=1}^{K} \sum_{i=1}^{r} e_{ij}$

K-fold Cross Validation: finding k (of kNN)



Calculating E for various k, we find the optimum value of k

Malay K. Das, ME, IIT Kanpur, mkdas@iitk.ac.in

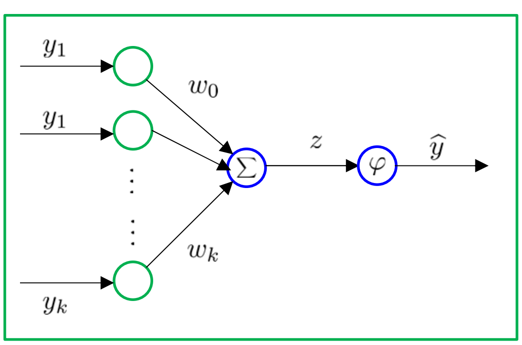
Visualizing **kNN**

we find k-nearest neighbors, and

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

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

$y_i = \{-1, 1\}$



$$z = \sum_{i}^{k} y_i \qquad \varphi(z) = \begin{cases} +1 & z \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

generally $w_0 = w_1 = \cdots = w_k = 1$

unless we are using weighted **kNN**

The above structure (called **Artificial Neuron**) is assumed to mimic a biological neuron; network of such **Artificial Neurons** (called artificial neural network, **ANN)** will be discussed later

Recall the definition of machine learning

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

kNN doesn't learn the fitting parameter **(k)** until we intervene

the learning process in kNN is called **lazy learning**

In **lazy learning,** learning starts when a test data is given

such parameters are known as hyperparameter

hyperparameter controls the learning process, learning doesn't determine hyperparameter

The opposite is **eager learning,** where learning is input-independent

Lazy learning, while efficiently handles new data, usually requires more memory/computation

Malay K. Das, ME, IIT Kanpur, mkdas@iitk.ac.in

Supervised Learning

Given a set of discrete data points $\mathcal{T} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$

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

'hat' sign indicates estimation (prediction)

NOT exact

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

feature

label

$\mathbf{x}_\star$ test data (unseen data)

If the label is **categorial** $y_i \in \mathbb{Z}$ (integer) called **classification problems**

Conversely, we may have **regression problems** where label is numerical $y_i \in \mathbb{R}$ (real)

Regression, Classification together constitute **Supervised Learning**

Summary: week 01

Course policy and outcome

Machine learning: definitions, comparison with mechanistic modeling framework

k-nearest neighbors

Remember the following terms/phrases/ideas: Feature, label, hyperparameter supervised/unsupervised learning, lazy/eager learning, regression, classification, artificial neural network, decision boundary, k-nearest neighbors, K-fold cross-validation

Think: kNN is not very effective for high-dimensional feature space, why?

Coming up in week 02

Review of linear algebra, regression