Lecture 1: Introduction

Statistical Learning and Data Mining

Xingye Qiao

Department of Mathematical Sciences
Binghamton University

E-mail: qiao@math.binghamton.edu

Read: ISR Chs. 1–2 and ESLII Chs. 1–2

Outline

The next section would be . . . . . .

What is statistical learning?

- Intersects: Data science, statistics, applied math, computer science.
- How is statistical learning different from machine learning / artificial intelligence?
- Simple definition: ML = an algorithm that can learn from data without relying on rules-based programming;
  SL = formalization of relationships between variables in the form of mathematical equations.
- Different schools.
- Different eras.
- Assumptions involved.

Name conventions

Names which refer to almost the same things

ML	Statistics
network, graphs	(generative) model
weight	parameter, coefficient
learning	fitting
generalization	test set performance
supervised learning	regression + classification
unsupervised learning	density estimator, clustering, and so on
learner, algorithm	model

Statistical Learning Tasks

- Prediction.
- Exploratory Analysis (Data-Driven Discoveries).

Types of statistical learning

- Supervised Learning: Data with labels or a response.
- Unsupervised Learning: Data with no labels.
- Others: Semi-supervised learning, recommender systems, online learning, network models, sequential learning, text mining, etc.

Other categorizations

We have seen supervised vs unsupervised.

- Active vs. Passive learners. An active learner interacts with the environment at training time, while a passive learner only observes the information provided by the environment (or the teacher) without influencing or directing it.
- Online vs. Batch learning

Course overview and expectations

- Piazza: homework assignments, lecture notes, discussions.
- Piazza activity counts.
- Deduction of 25% of the grade for homeworks that are not typeset using LaTeX. Deduction of 15% of the grade for each day homeworks are late (the final grade for a late homework that is N days late will be 0.85^N times the real grade).
- Homeworks may be discussed with classmates but must be written and submitted individually.

The next section would be . . . . . .

The SL framework

- Domain set: $\mathcal{X}$. Can be Euclidean space for simplicity.
- Label set: $\mathcal{Y}$ can be a set of real numbers or a finite set
- Training data: $S = \{(x_i, y_i)_{i=1,\ldots,n} \in \mathcal{X} \times \mathcal{Y}\}$
- Unknown underlying distribution: $(X, Y) \sim F$.
- iid: Assume that each (X_i, Y_i) is iid according to F.
- Learner's output: a hypothesis or a prediction rule, $h_n : \mathcal{X} \mapsto \mathcal{Y}$. The subscript n emphasizes that the prediction rule depends on S (which has size n) and hence is random.
- Measure of success: In regression,

$$R(h) = \mathrm{E}_F(h(X) - Y)^2.$$

In classification,

$$R(h) = P_F(h(X) \neq Y).$$

Here h is general. May be deterministic, or may be data dependent h_n which is random. Synonymous names such as the generalization error, the test error, or the true error of h.

Classification: Bayes rule

We now focus on binary classification ($Y = \pm 1$) as an example.

- Given h, the risk of h is $R(h) = P_F(h(X) \neq Y) = \mathrm{E}_F(\mathbb{1}\{h(X) \neq Y\})$.
- Define the regression function $\eta(x) = \mathrm{E}_F(Y|X = x) = 2P_F(Y = 1|X = x) - 1$.
- Define a rule $h_B(x) = \mathrm{sign}(\eta(x))$
- h_B achieves the minimal risk over all possible measurable functions:

$$R(h_B) = \inf_{h \in \mathcal{M}} R(h)$$

- We denote $R(h_B)$ by R^*, called the Bayes risk. h_B is called the Bayes rule.
- Note h_B does not depend on the data, but on the unknown distribution F.

Empirical Risk Minimization

Our goal is to identify h_B. However, the risk $R(h)$ cannot be directly measured since F is unknown.

- It is common to find an approximation to the true mean, using the sample mean, such as approximating $\mathsf{E}_F(\mathbb{1}\{h(X) \neq Y\})$ by

$$R_n(h) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}\{h(X_i) \neq Y_i\}$$

- This is called the empirical risk, since this is equivalent to the risk with distribution F replaced by empirical distribution.
- The Empirical Risk Minimization (ERM) rule/algorithm finds a predictor h that minimizes $R_n(h)$. I.e. $\operatorname{argmin}_h R_n(h)$
- Note that there may be several h that minimize the training error and an ERM algorithm is free to choose any such h.

Overfitting

- Example: $Y = \pm 1$. Among 100 training data points uniformly distributed in the 2D sphere with radius 2, $Y = 1$ deterministically as long as $\|X\| < 1$, or -1 otherwise.
- We propose a possible ERM predictor (classifier): $h(x) = y_i$ if there exists $x_i = x$, or 1 otherwise.
- Perfect for training data: $R_n(h) = 0$. Zero training error.
- Calculate the generalization error: $R(h) = P_F(h(X) \neq Y) = 0.75$. Large test error.
- This is overfitting: performance on the training set is excellent but performance on the true world is very bad.
- Intuitively, overfitting occurs when we can explain almost every data point in the training data.

A possible solution

- Apply the ERM learning rule, but restrict the search space.
- Formally, the learner should choose in advance (before seeing the data) a set of predictors. This set is called a hypothesis class and is denoted by $\mathcal{H}$. That is, each $h \in \mathcal{H}$ is a function mapping from $\mathcal{X}$ to $\mathcal{Y}$. After deciding on $\mathcal{H}$, the learner samples a training set S and uses the ERM rule to choose a predictor out of the hypothesis class.
- It can be shown that this may reduce the generalization error.
- Why it might work: control the complexity of the model.

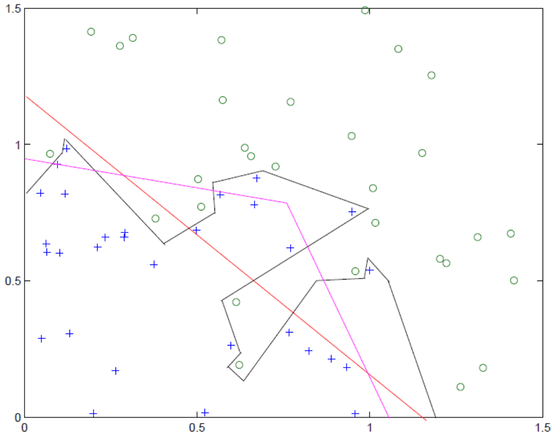


Figure: Trade-off between fit and complexity

Typically, one would look, in a collection of possible models, for one which fits well the data, but at the same time is as simple as possible

Complexity

- There is no universal way of measuring simplicity (or its counterpart complexity) and the choice of a specific measure inherently depends on the problem at hand.
- It is actually in this choice that the designer of the learning algorithm introduces knowledge about the specific phenomenon under study.
- This leads to the statement that data cannot replace knowledge, or in pseudo-mathematical terms:

$$\textbf{Generalization} = \textbf{Data} + \textbf{Knowledge}$$

Regularization

- Recall that the ERM seeks to find $\mathrm{argmin}_h R_n(h)$
- Another approach: choosing a large model $\mathcal{H}$ and then define on $\mathcal{H}$ a regularizer, typically a norm $\|h\|$. Then one has to minimize the regularized empirical risk:

$$h_n = \operatorname*{argmin}_{h \in \mathcal{H}} R_n(h) + \lambda \|h\|^2$$

- There is free parameter λ, called the regularization parameter which allows to choose the right trade-off between fit and complexity.
- Tuning λ is usually a hard problem and most often, one uses extra validation data for this task.
- Most existing (and successful) methods can be thought of as regularization methods.

A Simple Approach to Prediction: Nearest Neighbors

The k-nearest neighbor classifiers are memory based, and require no model to fit. Given a point $\boldsymbol{x}^*$, we find k points $\boldsymbol{x}_{(r)}, r = 1, \dots, k$, among training inputs, closest in *distance* to $\boldsymbol{x}^*$. $\boldsymbol{x}^*$ is classified using majority vote among the k neighbors.

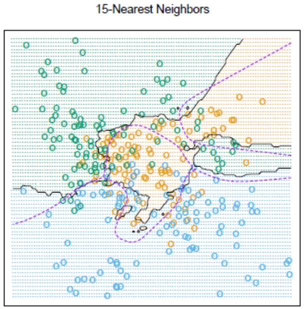
- simple to use.
- shown to be successful in real examples.
- requires large memory if the dataset is huge.
- for regression tasks, take the average of the y values of the k neighbors to predict y^*

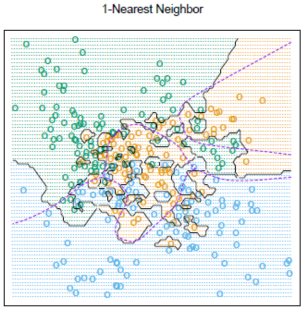
Example from ESLII (Hastie, Ribshirani, Friedman)

k-Nearest Neighbor classifiers applied to a simulated data set, with three groups.

The decision boundary of a 15-Nearest Neighbor classifier (top) is fairly smooth compared to a 1-Nearest Neighbor classifier (bottom).

Smaller *k*, more complex model. Larger *k*, more simple model.





lecture1 R code

1. Predict the label based on the labels of the K closest observations to the data point of interest.
2. What happens to the test prediction error when K is small? Large?

High-dimensional data & curse of dimensionality

- Intuition breaks down in high dimensions, and the phenomenon is commonly referred to as the curse of dimensionality
- Intuition 1: a small local neighborhood with s fraction of the entire space should approximate the true value well. However it has a radius of $s^{1/p}$ which goes to 1 quickly as p increases, hence **no longer local**.
- Intuition 2: it is easy to reach a neighbor. However, for large p, most data points are closer to the edge of the sample space than to any neighbor. (Not a good thing.)
- Local methods such as kNN can fail for high dimensional data.

Function Approximation

- Many SL problems can be seen as function approximation.
- (x_i, y_i) is seen as a point in a $(p+1)$-dimensional space. The function to be approximated $f(x)$ is related to the data via a model $y_i = f(x_i) + \varepsilon_i$
- While this looks familiar from regression, it is also related to classification problems via $y_i = \mathrm{sign}(f(x_i) + \varepsilon_i)$
- Common class of approximations:
  - Linear model: $f(x) = x^T \beta$
  - Linear basis expansion: $f(x) = \sum_{k=1}^{K} h_k(x)\theta_k$ where h_k are a suitable set of functions or transformations of the input vector x such as x_1^2, $x_1 x_2^2$, $\cos(x_1)$, etc.
- To this end, finding the optimal h boils down to f, then to β (or θ_k's).

Summary

- What is SL?
- Framework of SL. Training data. Goal. Risk function.
- Bayes rule. Training and test errors.
- Overfitting
- Model complexity and regularization.
- Curse of dimensionality
- kNN – an example.