

Introduction to Statistical Machine Learning



Xiang Zhou

School of Data Science
Department of Mathematics
City University of Hong Kong

Learning is a problem of function estimation on the basis of empirical data.

– Vladimir N. Vapnik

Supervised Learning:

How to estimate f^* in Bayes rule, but based on (a finite number of) data?

$$f^*(x) = \mathbb{E}(Y|X = x)$$

Two approaches

- ① approximate the Bayes rule's formula : solve the inverse problem of finding its distribution $p_{X,Y}$
 - ▶ Density estimation: Given $\{x_i\}$, infer the distribution p_X (e.g. in unsupervised learning); Given $\{x_i, y_i\}$, infer the distribution $p_{X,Y}$.
 - ▶ estimate conditional expectation¹ $\mathbb{E}[Y|X = x]$ by using neighboring data around x .
- ② direct method: Minimize ~~expected error~~ empirical error;

Remark: The first one is usually suffering from curse of dimensionality.

¹or conditional probability for the Bayes classifier
Xiang Zhou CityU

Nearest neighbors method

to approximate condition expectation

A natural way to approximate the function $\mathbb{E}(Y|X = x)$ is to replace the expectation by the average from the data $D = \{(x_i, y_i)\}$:

$$f^*(x) = \mathbb{E}(Y|X = x) \approx \text{AVE} \{y_i : \text{where } x_i = x\}, \quad \forall x \in \mathcal{X}$$

or

$$f^*(x) = \int_{\mathcal{Y}} y p_{Y|X}(y; x) dy$$

where $p_{Y|X}(y; x) = \frac{p_{X,Y}(x,y)}{p_X(x)}$ by estimating the density $p_{X,Y}(x, y)$ first?

Question: Would these work well in practice ?

(1) this formula is defined point-wisely; (2) one does not have an accurate estimation of the expectation or the joint distribution $p(x, y)$, particularly in high dimension. (3) This is a **local** method for approximation $f^*(x)$: no data outside of the local neighbor of x are used to predict y .

- Use windows with size Δ

$$\mathbb{E}(Y|X = x) \approx \text{AVE} \{y_i : \text{where } |x_i - x| \leq \Delta\}$$

- Use k -nearest neighbors (KNN): have a look at its neighbors, and take a vote:

$$\mathbb{E}(Y|X = x) \approx \text{AVE} \{y_i : x_i \in \mathcal{N}_k(x)\}$$

where $\mathcal{N}_k(x)$ is a neighborhood of x that contains exactly k neighbors (k -nearest neighbors). The number samples in AVE is fixed (k).

Two approximations are happening here:

- ① expectation is approximated by averaging over sample data;
- ② conditioning at a point is relaxed to conditioning on some region “close” to the target point.

Note: A metric on \mathcal{X} is necessary to define “neighbor” here. The choice of metric, $|x - x_i|$, is a practical issue.

k -NN for classification

For classification:

$$\mathbb{P}(Y = \textit{class} | X = x) \approx \frac{1}{N_k(x)} \sum_{x_i \in \mathcal{N}_k(x)} \mathbf{1}(y_i = \textit{class}), \quad \textit{class} = 1, 2, \dots, K$$

where $\mathcal{N}_k(x)$ is the collection of k points in $\{x_i\}$ and $N_k(x)$ is the size of the set $\mathcal{N}_k(x)$. I is the indicator function.

Kernel method to generalize the k -NN

- The k -NN gives a discontinuous piecewise-constant function to fit data; By playing with “average”, a smooth function can be obtained.
- Define a Gaussian kernel as the weight

$$K_{\lambda}(x_0, x) = \frac{1}{\lambda} \exp \left[-\frac{\|x - x_0\|^2}{2\lambda} \right]$$

- The simplest form of kernel estimate is the (Nadaraya–Watson) weighted average

$$\hat{f}(x) = \sum_{i=1}^N y_i w_i(x), \quad \text{where } w_i(x) = \frac{K_{\lambda}(x, x_i)}{\sum_{j=1}^N K_{\lambda}(x, x_j)}$$

- This weighted average by kernel has many other applications.

Main Principle

If you possess a restricted amount of information for solving some problem, try to solve the problem directly and never solve a more general problem as an intermediate step. It is possible that the available information is sufficient for a direct solution but is insufficient for solving a more general intermediate problem.

V. N. Vapnik

density estimation is a universal problem of statistics — too general.

Empirical Risk Minimization

Start from two key ideas

- ① Search f in certain hypothesis space (hypothesis class, model class) \mathcal{H} , instead of any function.
- ② approximate the expectation \mathbb{E} in \mathcal{E} with the average from a dataset $D = \{x_i, y_i\}$.

- Loss function $\ell(y, \hat{y}) : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}^+ \cup \{0\}$
 - ▶ L_2, L_1 norm
 - ▶ 0-1 loss: $\ell(y, \hat{y}) = I(y \neq \hat{y})$.
 - ▶ ...
- Population risk/loss (Generalization error):

$$\mathcal{E}(f) := \mathbb{E} \ell(Y, f(X))$$

The direct approach of minimizing the risk

$$\min_f \mathcal{E}(f) = \mathbb{E} \ell(Y, f(X))$$

is impossible since f can be any function herel.

It is important to specify a hypothesis space \mathcal{H} to restrict the search space.

Empirical risk minimization(ERM)

- ① The minimizer in the model space of

$$\min_{f \in \mathcal{H}} \mathcal{E}(f)$$

is denoted by $f_{\mathcal{H}}$.

- ② Population loss/risk is approximated by **empirical error/risk/loss**, $\hat{\mathcal{E}}$, associated with a dataset $D = \{(x_i, y_i) : i = 1, \dots, N\}$:

$$\mathcal{E}(f) := \mathbb{E} \ell(Y, f(x)) \approx \hat{\mathcal{E}}(f) := \frac{1}{N} \sum_{(x_i, y_i) \in D} \ell(y_i, f(x_i))$$

The learning algorithm produces the function \hat{f} (**regression function/predictor**) as the solution to

$$\hat{\mathcal{E}}(\hat{f}) = \min_{f \in \mathcal{H}} \hat{\mathcal{E}}(f) = \min_{f \in \mathcal{H}} \frac{1}{N} \sum_{(x_i, y_i) \in D} \ell(y_i, f(x_i))$$

- ③ The dataset D used in practical minimization of $\hat{\mathcal{E}}$ is called the **training dataset (training samples)**.

- By restricting the learner to choosing a predictor from \mathcal{H} , we may *bias* it¹ toward a particular set of predictors.
- \hat{f} depends on *three* elements: the hypothesis space \mathcal{H} , the (training) data D and the loss function ℓ . We may denote \hat{f}_D to emphasize the dependence on D .
- The **training error** is $\hat{\mathcal{E}}(\hat{f})$.
- The **test error/prediction error** is $\mathbb{E} \ell(y_{new}, \hat{f}(x_{new}))$
- The **expected** test/prediction error is

$$\mathbb{E}_{(x_{new}, y_{new}) \sim p_{X,Y}} [\ell(y_{new}, \hat{f}(x_{new}))] = \mathcal{E}(\hat{f})$$

- The expected predictor error (generalization error) is the true performance indicator of the function \hat{f} , which requires running on several (statistically identical) datasets.
- For the fixed \mathcal{H} and ℓ , \hat{f}_D is essentially a mapping from D to the function space \mathcal{H} . As $N \rightarrow \infty$, $\hat{f}_D \rightarrow f_{\mathcal{H}}$ by the law of large number.

¹called “inductive bias” in [UML]

Hypothesis space: representation of the function f

- parametric approach: e.g. $f(x) = \theta \cdot x$ with a set of finite and (usually fixed) number of parameters θ
- non-parametric approach: functional analysis viewpoint, f is only in some function space. (such as in the traditional computational math for representing the solution to some PDE)
- there is no rigorous boundary between parametric/non-parametric: eventually the computer represents a function f by a finite number of freedoms.
- need to be restricted to certain model class in practice. Examples: linear model, polynomial, spline, kernel machine, neural network, etc.

choice of \mathcal{H} : the hidden structure of the function

- Linear function $f(x) = \beta \cdot x \rightarrow$ linear regression
- Piece-wise constant functions $\rightarrow k$ -NN method
- Given a set of pre-specified¹ basis functions (dictionary) h_m ,

$$f_{\theta}(x) = \sum_{m=1}^M \theta_m h_m(x),$$

\rightarrow generalized linear model, dictionary methods

- nonlinearly transform: dimensionality reduction, neural network etc.
- Radial basis function (kernel method)

$$f(x) = \sum_{m=1}^M \theta_m K_{\lambda_m}(\mu_m, x)$$

where K is a kernel measuring the “closeness” between points.

- Roughness penalty

$$\mathcal{E}(f; \lambda) = \mathcal{E}(f) + \lambda \mathcal{R}(f),$$

where $\mathcal{R}(f)$ is a regularization term on smoothness of f .

¹problem dependent

Is a larger^a model space is better than a smaller model space ?

^amore complex, more flexible; heuristic language: the precise meaning needs a careful study

Classic Answer: (like physics)

- a simple model is easy to interpret;
 - a simple model is easy to compute and deploy;
 - but a simple *and* right model that explains data well is not easy to find.
-

Occam's razor

a classic philosophy of generalization being questioned by modern research

CORE PRINCIPLES IN RESEARCH



OCCAM'S RAZOR

"WHEN FACED WITH TWO POSSIBLE EXPLANATIONS, THE SIMPLER OF THE TWO IS THE ONE MOST LIKELY TO BE TRUE."



OCCAM'S PROFESSOR

"WHEN FACED WITH TWO POSSIBLE WAYS OF DOING SOMETHING, THE MORE COMPLICATED ONE IS THE ONE YOUR PROFESSOR WILL MOST LIKELY ASK YOU TO DO."

WWW.PHDCOMICS.COM

JORGE CHAM © 2009

Restricted model space in practical use

In practice, \mathcal{H} is further restricted to control the model complexity:

- add explicit restriction to the parameter, such as $\|\theta\| \leq r$
- hyper-parameters give rise to different model complexity

Structural Risk Minimization (SRM) method:

$$\cdots \subset \mathcal{H}_k \subset \mathcal{H}_{k+1} \cdots \subset \mathcal{H}$$

and pick up the right k .

Practical Strategies: Ask domain expert, or

- ① Start with a highly complex model space
- ② variable selection: Shrink to a proper restricted space by tuning the optimal hyper-parameter

or

- ① Use dimensionality reduction (unsupervised learning) to transform high dimension raw data to just a few features
- ② Apply the interpretable machine learning methods to the transformed features as the new input

or others...this magic box (deep learning) just rocks!

Three contributions to the generalization error

The generalization error is decomposed into¹

$$\mathcal{E}(f^*) = \underbrace{\mathcal{E}(f^*) - \mathcal{E}(f_{\mathcal{H}})}_{\text{approximation error}} + \underbrace{\mathcal{E}(f_{\mathcal{H}}) - \hat{\mathcal{E}}(\hat{f}_{\mathcal{D}})}_{\text{sampling error}} + \underbrace{\hat{\mathcal{E}}(\hat{f}_{\mathcal{D}})}_{\text{training error}}$$

- Approximation error:² only depends on the hypothesis space \mathcal{H}
- Sampling error: vanishes if the number of data points N goes to infinity. This error may be analyzed by probability (concentration) inequality (e.g. Chebychev);
- Training error: how accurate the optimization algorithm is in practice to find the minimizer(s) of $\hat{\mathcal{E}}$.

¹This divisions of triple error parts might not be useful for understanding deep learning, which nobody really understands today.

²sometimes, it refers to $\mathcal{E}(f_{\mathcal{H}})$ only

- Approximation error: characterize how large (complex) the hypothesis space \mathcal{H} is. A larger space of \mathcal{H} indicates
 - ▶ A larger effective dimension of this space;
 - ▶ A smaller approximation error; lower bias
 - ▶ More difficult for numerical optimization;
- Sampling error: vanishes if the number of data points N goes to infinity. This error may be analyzed by probability inequality (e.g. Chebychev); analysis method used for Monte Carlo simulation. A larger sample size for training data indicates:
 - ▶ Smaller sampling error; lower variance;
 - ▶ Severe challenges for big data issue and optimization;
 - ▶ The affordable or available sample size in practice may not be dominatively large in comparison to the number of parameters in the model (e.g. over-parametrized neural network) and the dimension of \mathcal{X} .
- Training error:
 - ▶ the stochastic gradient method is essential in training.
 - ▶ the time and memory cost ...
 - ▶ the output is a further approximation to \hat{f}_D : the optimization algorithm stops with a time cost T : $\hat{f}_D^{(T)}$, which introduce the *numerical errors* in solving the minimization problem.

- A small value of training error $\hat{\mathcal{E}}(\hat{f}_{\mathcal{D}})$ does not imply a small value of test error $\mathcal{E}(\hat{f}_{\mathcal{D}})$.
- These three topics are not independent. For example, the necessary number of sample size N may depend on the complexity of hypothesis space, which further depends on the dimension of \mathcal{X}, \mathcal{Y} and the specific representation form of functions in \mathcal{H} .
- A holistic viewpoint is very important to have a complete picture of the gap.

The contribution of Machine Learning

(practical) machine learning (community) , in general, focuses on

- design specific loss functions ℓ for various specific learning tasks from different motivations and perspectives;
- the choice (and understanding) of the model space \mathcal{H} , the representation of functions in \mathcal{H}
- restrict \mathcal{H} in practice like regularization¹ to shrink this space for avoiding overfitting
- develop (stochastic/distributed) optimization methods for big data, large scale problem, etc.
- select, evaluate and validate the model
- theoretic understanding why these methods work (or fail).
- applications

¹many other practical tricks exists like dropout/pooling, even early stopping or stochastic gradient.

Generalization Gap

The optimal function in theory is f^* ; the optimal function in practical computation is \hat{f}_D . The difference in risks of these two:

$$\left| \mathcal{E}(\hat{f}_D) - \mathcal{E}(f^*) \right|$$

or (someone uses)

$$\left| \hat{\mathcal{E}}(\hat{f}_D) - \mathcal{E}(f^*) \right|$$

is called **generalization gap**. It measures to which extent the performance of the algorithm learnt from a given training data set D is valid for the whole distribution $p(x, y)$ ¹.

Remark

- $\mathcal{E}(\hat{f}_D) - \mathcal{E}(f^*)$ still depends on D , so it is essentially still a random number. One can use the expected $\mathbb{E}_D \left[\mathcal{E}(\hat{f}_D) - \mathcal{E}(f^*) \right]$ to represent the average.
- In fact, we can have a supremum upper bound.

¹In practice, this means for new (test) data with the same distribution

First inequality for Generalization Gap

Theorem

The upper bound of the generalization gap:

$$\begin{aligned} \left| \mathcal{E}(\hat{f}_D) - \mathcal{E}(f^*) \right| &= \mathcal{E}(\hat{f}_D) - \hat{\mathcal{E}}(\hat{f}_D) + \hat{\mathcal{E}}(\hat{f}_D) - \hat{\mathcal{E}}(f^*) + \hat{\mathcal{E}}(f^*) - \mathcal{E}(f^*) \\ &\leq 2 \sup_{f \in \mathcal{H}} \left| \hat{\mathcal{E}}(f) - \mathcal{E}(f) \right| \approx 2 \sup_{f \in \mathcal{H}} \frac{1}{\sqrt{N}} \sqrt{\text{Var}(\ell(Y, f(X)))} \end{aligned}$$

Remark: the second term in telescope is negative. The “ \approx ” is due to the central limit theorem.

The above inequality is a typical form in learning theory from the approximation theory viewpoint.

- The supremum in the above inequality can describe the **complexity** of hypothesis space \mathcal{H} , which increases when \mathcal{H} is “larger”.
- Although this upper bound is not too far way from the true gap; but it is not sharp in many cases and fails to serve as a good indicator of the generalization gap. A good lower bound is extremely difficult to obtain for challenging cases (like deep learning)

An example of PAC (Probably Approximate Correct) theorems
Assume $|\mathcal{H}| < \infty$. Write $\hat{\mathcal{E}}$ as $\hat{\mathcal{E}}_N$ for the dependency on N .

Theorem

$\delta > 0, \varepsilon > 0$. If the sample size $N = O\left(\frac{\log \frac{|\mathcal{H}|}{\delta}}{\varepsilon^2}\right)$, then

$$\mathbb{P}\left(\sup_{f \in \mathcal{H}} \left| \mathcal{E}(f) - \hat{\mathcal{E}}_N(f) \right| < \varepsilon\right) \geq 1 - \delta.$$

Evaluation of the trained model : test error

- Recall $\mathcal{E}(f) = \mathbb{E}_{X,Y} \ell(Y, f(X))$, then for \hat{f}_D from the training set D , its generalization performance is given by

$$\mathcal{E}(\hat{f}_D) = \mathbb{E}_{X,Y} \ell(Y, \hat{f}_D(X)) \approx \frac{1}{N_T} \sum_{(X_i^T, Y_i^T) \in T} \ell(X_i^T, \hat{f}_D(X_i^T))$$

This is called **test error** .

- The **test dataset** $T = \{(X_i^T, Y_i^T), 1 \leq i \leq N_T\}$ are also iid drawn from the distribution $p_{X,Y}$. But T and D must be **independent**. In practice, this means training dataset and test dataset should never overlap:

$$T \cap D = \emptyset$$

- [ELS] elements of statistical learning. Chapter 2, 7;
- [UML] Understanding Machine Learning: From Theory to Algorithms. 2014.