Introduction to Statistical Machine Learning

Xiang Zhou

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

Supervised Learning:

How to estimate f^* in Bayes rule, but only use data?

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

Bayes rule $f^*(x)$ serves as the analytical solution to our problem, is the ground truth we can best have in idealized situation.

Two approaches

- less popular method: direct method by definition of conditional expectation [1];
- very popular method: minimize ~~generalization error~~ empirical error;
- They are not strictly distinct.

[1] or conditional probability for the Bayes classifier

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 $\mathtt{D} = \{(x_i, y_i)\}$:

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

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.

k-NN

- 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:

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

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

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

- This weighted average by kernel has many other applications.

Empirical Risk Minimization

Start from two key ideas

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

- Loss function $\ell(y, \hat{y}) : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^+ \cup \{0\}$
  - 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 loss/risk** $\widehat{\mathcal{E}}$ associated with a dataset $\mathtt{D} = \{(x_i, y_i) : i = 1, \ldots, N\}$:

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

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

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

3. The dataset $\mathtt{D}$ used in practical minimization of $\widehat{\mathcal{E}}$ is called the training dataset.

- $\hat{f}$ depends on *three* elements: the hypothesis space $\mathcal{H}$, the (training) data $\mathtt{D}$ and the loss function ℓ. We may denote $\hat{f}_{\mathtt{D}}$ to emphasize the dependence on $\mathtt{D}$.
- The **training error** is $\widehat{\mathcal{E}}(\hat{f})$.
- The **test error/prediction error** is $\mathbb{E}\,\ell(y_{new}, \hat{f}(x_{new}))$
- The **expected** test/prediction error is

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

- 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, $\hat{f}_{\mathtt{D}}$ is essentially a mapping from $\mathtt{D}$ to the function space $\mathcal{H}$. As $N \to \infty$, $\hat{f}_{\mathtt{D}} \to f_{\mathcal{H}}$ by the law of large number.

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 $f(x) = \beta \cdot x \longrightarrow$ linear regression
- Piece-wise constant functions $\longrightarrow k$-NN method
- Given a set of pre-specified[1] basis functions (dictionary) h_m,

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

  $\longrightarrow$ 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.

[1] problem dependent

Occam's razor

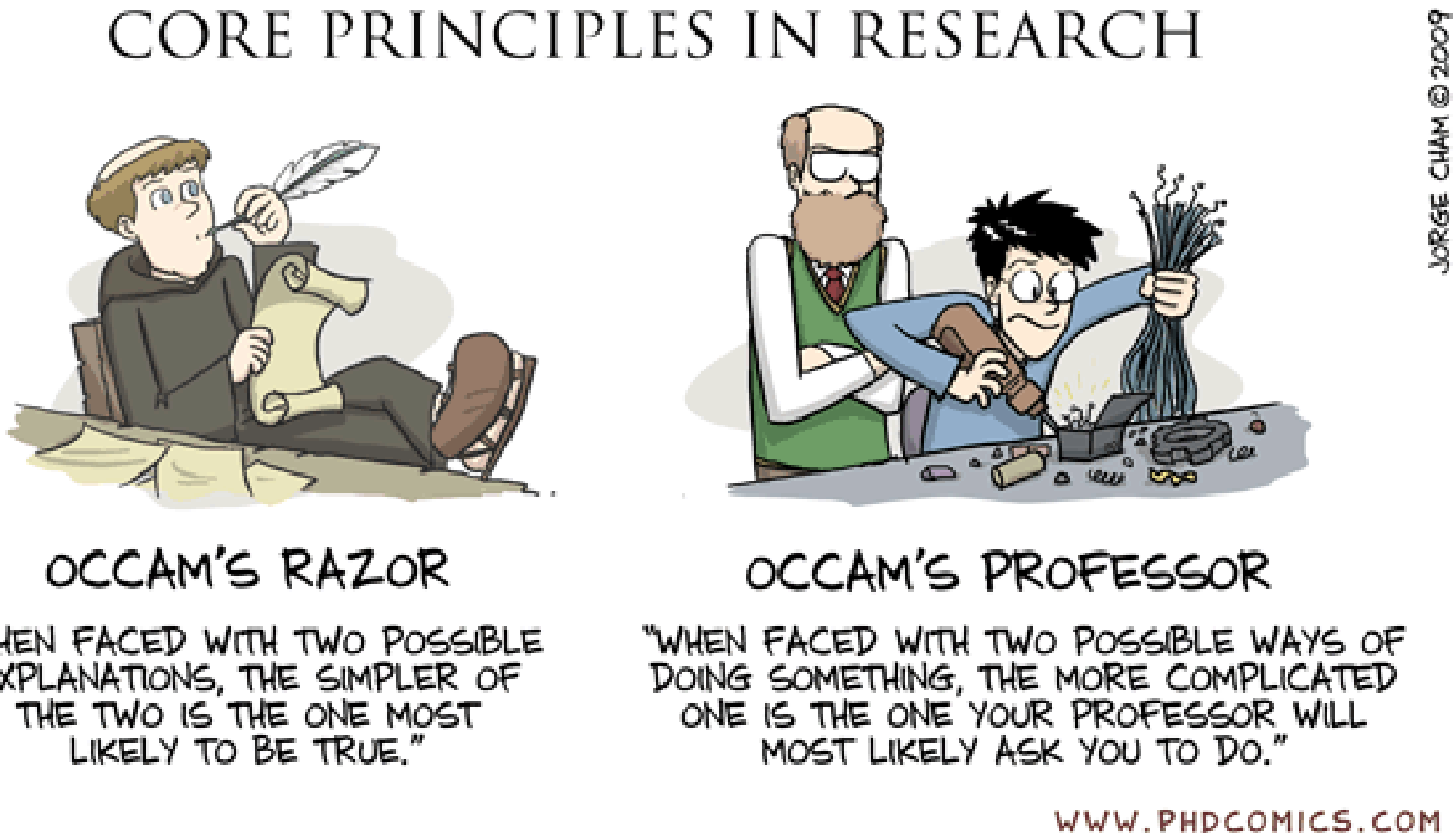
Is a larger[a] model space is better than a smaller model space ?

[a] more complex, more flexbile; heuristic language: the precise meaning needs a careful study

Answer: (like physics)

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

Occam's razor



Restricted model space in practical use

The model space $\mathcal{H}$ is a family of functions described by parameters (such as θ) or hyper-parameters (M, λ); see previous slides.

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

Practical Strategies

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 input

or I do not care explanation...this magic box (deep learning) just rocks!

Three contributions to the generalization error

The generalization error is decomposed into [1]

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

- Approximation error: analogy to the (optimal) interpolation inequality in finite element method.
- 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 $\widehat{\mathcal{E}}$.

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

- 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 optimzation;
  - 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 $\hat{f}_{\mathtt{D}}$: the optimization algorithm stops with a time cost T: $\hat{f}_{\mathtt{D}}^{(T)}$, which introduce the *numerical errors* in solving the minimization problem.

Remarks

- A small value of training error $\widehat{\mathcal{E}}(\hat{f}_{\mathrm{D}})$ does not imply a small value of test error $\mathcal{E}(\hat{f}_{\mathrm{D}})$.
- 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 regularization[1] 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

[1] many 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 $\hat{f}_{\mathtt{D}}$. The difference in risks of these two:

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

or (someone uses)

$$\left|\widehat{\mathcal{E}}(\hat{f}_{\mathtt{D}}) - \mathcal{E}(f^*)\right|$$

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

Remark

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

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

First inequality for Generalization Gap

Theorem

The upper bound of the generalization gap:

$$\left|\mathcal{E}(\hat{f}_{\mathtt{D}}) - \mathcal{E}(f^*)\right| = \mathcal{E}(\hat{f}_{\mathtt{D}}) - \widehat{\mathcal{E}}(\hat{f}_{\mathtt{D}}) + \widehat{\mathcal{E}}(\hat{f}_{\mathtt{D}}) - \widehat{\mathcal{E}}(f^*) + \widehat{\mathcal{E}}(f^*) - \mathcal{E}(f^*)$$

$$\leq 2 \sup_{f \in \mathcal{H}} \left|\widehat{\mathcal{E}}(f) - \mathcal{E}(f)\right| \approx 2 \sup_{f \in \mathcal{H}} \frac{1}{\sqrt{N}} \sqrt{\mathit{Var}(\ell(Y, f(X)))}$$

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)

Evaluation of the trained model : test error

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

$$\mathcal{E}(\hat{f}_{\mathtt{D}}) = \mathbb{E}_{X,Y}\, \ell(Y, \hat{f}_{\mathtt{D}}(X)) \approx \frac{1}{N_{\mathtt{T}}} \sum_{({X^{\mathtt{T}}}_i, {Y^{\mathtt{T}}}_i) \in \mathtt{T}} \ell({X^{\mathtt{T}}}_i, \hat{f}_{\mathtt{D}}({X^{\mathtt{T}}}_i))$$

  This is called test error .

- The test dataset $\mathtt{T} = \left\{({X^{\mathtt{T}}}_i, {Y^{\mathtt{T}}}_i), 1 \le i \le N_{\mathtt{T}}\right\}$ are also iid drawn from the distribution $p_{X,Y}$. But $\mathtt{T}$ and $\mathtt{D}$ must be **independent**. In practice, this means training dataset and test dataset should never overlop:

$$\mathtt{T} \cap \mathtt{D} = \emptyset$$