INTRODUCTION TO ALGORITHMS FOR DATA SCIENCE AND PHYSICS 2025/2026

(数据科学及物理问题中的算法导论)

@IMP/FUDAN

SYLLABUS

Bao-Jun Cai, bjcai@fudan.edu.cn

—Course Description—

This course offers an introduction to the fundamental concepts of algorithms and several widely used methods in machine learning, state estimation/inference, and modern data science. A key objective is to help students understand the underlying principles and theoretical foundations of machine learning algorithms. Topics covered include basic probability and statistical techniques, simulation methods, optimization strategies, and essential ideas from high-dimensional problems and reconstruction algorithms. To keep pace with the rapid evolution of big data and modern analytics, the course also explores a selection of contemporary algorithms developed for large-scale data processing and high-dimensional statistical analysis.

—Target Audience—

This course is intended for advanced undergraduate and graduate students in physics, computer science, data science, or related fields who are interested in the foundations and practical applications of machine learning and modern data analysis. It is also suitable for researchers and professionals seeking to strengthen their understanding of algorithmic methods in state estimation, optimization, computer vision and physics. A basic background in calculus, linear algebra, and probability is recommended, though key concepts will be reviewed as needed throughout the course. You are all very welcome to attend the course!

—Exercises and Problems—

Each class will last approximately 90 to 135 minutes. There will be two types of exercises: analytical problems (such as estimations or derivations) and programming assignments. The analytical problems are relatively straightforward, while the programming tasks require more thoughtful design and implementation. Each week, 1-2 programming exercises will be assigned to reinforce the key concepts covered in class. These exercises are designed to closely reflect practical techniques in areas such as physics, state estimation, geometric reasoning under uncertainty, and optimization.

—Course Grading Policy—

(a) *Homework*: 10 points/per module × 5 modules=50 points, except Module F.

(b) *Quiz*: 10 points/per module × 5 modules=50 points, except Module F.

—Detailed Schedules (Preliminary)—

Module–A *Basis of Calculus, Probability and Statistics* (微积分、概率与统计基础)

LECTURE 1 — *Course Introduction and Some Examples*

(a) method of guess and estimation, period of a simple pendulum system
(b) master theorem, divide-and-conquer, Fibonacci series
(c) fast algorithm for matrix multiplication
(d) solution of the algebraic equation $x^n(t) = \Omega + tx(t)/\Lambda, x(t) \in \mathrm{R}^+, n \geq 5, n \in \mathrm{N}^+$

LECTURE 2 — *Calculus and Concept of Sampling*

(a) Taylor's expansion of a function
(b) five-point algorithm
(c) gradient and Hessian of a matrix
(d) Monte Carlo method for integration
(e) estimating π by hit-or-miss and by Markov chain sampling

LECTURE 3 — *Elementary Introduction to Statistics and Probability*

(a) mean and variance of a distribution
(b) Bayes' theorem
(c) moment-generating function
(d) random variable generation, Box–Muller method

Module–B *Optimization, Learning from Data and Bias–variance Trade–off* (基本优化算法、方差-偏差分解)

LECTURE 4 — *First-order Optimization Algorithms*

(a) gradient descent search, exact-line search
(b) Jensen's inequality for convex function
(c) regularization, singular behavior of Hessian
(d) concept of momentum
(e) AdaGrad, RMSProp, Adam, hypergradient

LECTURE 5 — *Trade–off between Bias and Variance*

(a) linear fitting, loss function
(b) nonlinear curve fitting, learning model
(c) no–free–lunch theorem, trade–off between bias and variance
(d) stochastic gradient descent
(e) geometrical error learning model

LECTURE 6 — *Annealing and Metropolis Algorithm*

(a) thermal motion, Boltzmann's distribution
(b) simulated (thermal) annealing
(c) rejection sampling, importance sampling
(d) fast annealing on finding the minimum of a multi-dimensional function
(e) 2D Ising model: simulations

LECTURE 7 — *Convergent Analysis and Large-scale Sparse Matrix*

(a) some facts related to the Hessian
(b) search algorithm directly using Hessian
(c) Newton–Raphson method, order of convergence
(d) Gauss–Newton and Levenberg–Marquardt schemes, applicable conditions
(e) conjugate gradient search

Module–C *State Estimation/Inference and Data-driven Algorithms* (状态估计/推断与数据驱动的算法)

LECTURE 8 — *Kalman Filter as a Data-driven Optimization Method*

(a) Kalman filter (without analytical derivations)
(b) nonlinear and non-Gaussian extensions
(c) bias estimation for the learning model $f_w(x) = e^{-wx}$

LECTURE 9 — *Bayesian Curve Fitting as a State Estimation Problem*

(a) high-dimensional Gaussian, decomposition of Gaussian
(b) Bayesian consideration on coin drawing experiment
(c) linear curve fitting using Bayesian calculation

LECTURE 10 — *3D Reconstruction from 2D Images for Computer Vision Problems*

(a) depth determination from stereo pairs
(b) picture of 3d reconstruction, motion determination, SLAM
(c) least squares, weighted least squares
(d) bundle adjustment (conceptual introduction)

Module–D *High-dimensional Problems, Randomized Algorithms and Fast Computing* (高维问题导论、随机算法与快速计算)

LECTURE 11 — *Clustering, Robustness and Sparsity*

(a) concept of classification, effects of dimensions
(b) k-means, k-center, k-median
(c) robustness and sparsity, LASSO
(d) Huber loss with ℓ-1 constraint, the optimization algorithm

LECTURE 12 — *High Dimensions, Singular-value Decomposition and Best-fit Subspace*

(a) law of large numbers, random data in high dimensions
(b) properties of a high-dimensional ball
(c) singular value decomposition, a greedy algorithm
(d) basic principle component analysis
(e) analysis on the MNIST dataset and whitening of noise data

LECTURE 13 — *Randomized Matrix Multiplication, Integer Partition and Phase Transition*

(a) massive data problem
(b) distinct elements of a set
(c) CUR decomposition
(d) saddle-point approximation
(e) Laplace's method for integration

Module–E *Fast Fourier Transform and Solvers for Partial Differential Equations* (快速 Fourier 变换与偏微分方程的离散化计算)

LECTURE 14 — *Fast Fourier Transform for Data Processing*

(a) product of polynomials
(b) fast Fourier transform for signal processing
(c) convolution of signals

LECTURE 15 — *Differencing Schemes for Partial Differential Equations*

(a) Euler search, Runge–Kutta algorithm
(b) convection equation, differencing schemes and stability conditions
(c) up-wind method
(d) random walk for Laplaces' equation

Module–F *A Primer for Quantum Algorithms* (量子算法基础)

LECTURE 16 — *Review of Basis of Quantum Mechanics*

(a) wave nature of particles and complex amplitude
(b) uncertainty relation between momentum and position
(c) operator and wave function
(d) amplitude as sum over histories, example: harmonic oscillator
(e) path integral simulations

LECTURE 17 — *Algorithmic Interference*

(a) quantum state, qubit, coherence and decoherence
(b) collapse of wave function and entanglement
(c) Shor algorithm and Deutsch's problem
(d) Gover's searching algorithm, square-root acceleration

SPECIAL LECTURE **Renormalization Group, Effective Field Theory and Deep Neural Network*

If time permits, I will introduce some modern idea on understanding deep neural network, via the methods of renormalization group and effective field theory.

— Lecture Notes and Selected References —

There will be no designated textbook for the course, but lecture notes will be provided progressively as the course develops. However, several reference books that offer broad coverage of relevant topics may be helpful for those seeking a deeper understanding (books with "♠" are recommended). You do not need to read everything.

(1) I. Jolliffe, *Principal Component Analysis*, Springer, 1986.
(2) R. Neal, *Bayesian Learning for Neural Networks*, Springer, 1996.
(3) G. Cowan, *Statistical Data Analysis*, Oxford, 1998.
(4) E. Lehmann and G. Casella, *Theory of Point Estimation*, Springer, 1998.
(5) M. Newman and G. Barkema, *Monte Carlo Methods in Statistical Physics*, Clarendon, 1999.
(6) B. Schölkopf and A. Smola, *Learning with Kernels*, Cambridge, 2002.
(7) ♠D. MacKay, *Information Theory, Inference, and Learning Algorithms*, Cambridge, 2003.
(8) S. Boyd and L. Vandenberghe, *Convex Optimization*, Cambridge, 2004.
(9) R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, Cambridge, 2004.
(10) ♠J. Kleinberg and E. Tardos, *Algorithm Design*, Pearson, 2005.
(11) C. Rasmussen and C. Williams, *Gaussian Processes for Machine Learning*, MIT, 2005.
(12) S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*, MIT, 2005.
(13) ♠C. Bishop, *Pattern Recognition and Machine Learning*, Springer, 2006.
(14) G. Casella and R. Berger, *Statistical Inference*, Thompson, 2006.
(15) T. Cover and J. Thomas, *Elements of Information Theory*, Wiley, 2006.
(16) ♠W. Krauth, *Statistical Physics: Algorithms and Computations*, Oxford, 2006.
(17) ♠J. Nocedal and S. Wright, *Numerical Optimization*, Springer, 2006.
(18) D. Sivia and J. Skilling, *Data Analysis: A Bayesian Tutorial*, Oxford, 2006.
(19) ♠♠W. Press, S. Teukolsky, W. Vetterling, and B. Flannery, *Numerical Recipes*, Cambridge, 2007.
(20) T. Hastie, R. Tibshirani, and J. Friedmann, *The Elements of Statistical Learning*, Springer, 2008.
(21) D. Koller and N. Friedman, *Probabilistic Graphical Models*, MIT, 2009.
(22) L. Wasserman, *All of Statistics*, Springer, 2010.
(23) C. Moore and S. Mertens, *The Nature of Computations*, Springer, 2011.
(24) D. Barber, *Bayesian Reasoning and Machine Learning*, Cambridge, 2012.
(25) R. Horn and C. Johnson, *Matrix Analysis*, Cambridge, 2012.
(26) Y. Mostafa, M. Ismail, and H.T. Lin, *Learning from Data*, AMLBook, 2012.
(27) ♠S. Prince, *Computer Vision: Models, Learning, and Inference*, Cambridge, 2012.
(28) S. Aaronson, *Quantum Computing Since Democritus*, Cambridge, 2013.
(29) A. Gelman, J. Carlin, H. Stern, D. Dunson, A. Vehtari, and D. Rubin, *Bayesian Data Analysis*, CRC, 2013.
(30) ♠♠G. Golub and Van C. Loan, *Matrix Computations*, John Hopkins, 2013.
(31) I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT, 2016.
(32) R. Vidal, Y. Ma, and S. Sastry, *Generalized Principal Component Analysis*, Springer, 2016.
(33) M. Mitzenmacher and E. Upfal, *Probability and Computing*, Cambridge, 2017.
(34) M. Wilde, *The Quantum Information Theory*, Cambridge, 2017.
(35) R. Vershynin, *High-dimensional Probability*, Cambridge, 2018.
(36) M. Wainwright, *High-dimensional Statistics*, Cambridge, 2018.
(37) G. Strange, *Linear Algebra and Learning from Data*, Cambridge, 2019.
(38) ♠A. Blum, J. Hopcroft, and R. Kannan, *Foundations of Data Science*, Cambridge, 2020.
(39) M. Deisenroth,*Mathematics for Machine Learning*, Cambridge, 2020.
(40) T. Hastie, R. Tibshirani, and M. Wainwright, *Statistical Learning with Sparsity*, Chapman & Hall, 2020.
(41) S. Skiena, *The Algorithm Design Manual*, Springer, 2020.
(42) L. Böttcher and H. Herrmann, *Computational Statistical Physics*, Cambridge, 2021.
(43) S. Brunton and J. Kutz, *Data-Driven Science and Engineering*, Cambridge, 2022.

(44) ♠♠T. Cormen, C. Leiserson, R. Rivest, and C. Stein, *Introduction to Algorithms*, MIT, 2022.
(45) J. Wright and Y. Ma, *High-dimensional Data Analysis with Low-dimensional Models*, Cambridge, 2022.
(46) ♠S. Prince, *Understanding Deep Learning*, MIT, 2023.
(47) A. Torralba, P. Isola, and W. Freeman, *Foundations of Computer Vision*, MIT, 2024.
(48) S. Dorogovtsev and J. Mendes, *The Nature of Complex Network*, Oxford, 2025.

LECTURE 5

Introduction to Algorithms for Data Science and Physics @IMP/Fudan

First Lesson from "Learning from Data", Curve Fitting, and "No-free-lunch" Theorem

Dr. Bao-Jun Cai

9/9/2025

DRAFT VERSION

Key Concepts of This Lecture

parameter estimation for the linear fitting function $f_{\vec{\theta}}(x) = ax + b$ from data
parabolic loss function $J(\vec{\theta}) = 2^{-1}\sum[f_{\vec{\theta}}(x^{(i)}) - y^{(i)}]^2$ and its optimization
goodness of learning model $\leftrightarrow$ decomposition of bias, variance and noise
penalty term $J \to J + \lambda g$, data and belief in data, avoidance of singularity
normal equation $\vec{\Phi}^\top\vec{\Phi}\mathbf{w} = \vec{\Phi}^\top\mathbf{y}$ and its stochastic gradient descent search

§1 Problem of Linear Curve Fitting

Assume that one has m data points $(x^{(i)}, y^{(i)})$ with $i = 1 \sim m$, here *the physical or the realistic relation* between $x^{(i)}$ and $y^{(i)}$ is assumed to be linear, e.g., the relation between the velocity v and the acceleration a as $v = at$. Since there exists measurement errors, the *measured or the experimental relation* between $x^{(i)}$ and $y^{(i)}$ is not exactly linear. In this case, one can still use the linear regression to obtain the model parameters from the noisy data.

Notations. In our notes, we use the superscript with a parentheses to number the data points. In addition, "m" denotes the number of data samples while "n" is used for denoting the number of parameters in the model. Conventionally, $m \gg n$ or equivalently $n/m \ll 1$; in some high-dimensional problems, however $n \gg m$ or $n/m \gg 1$.

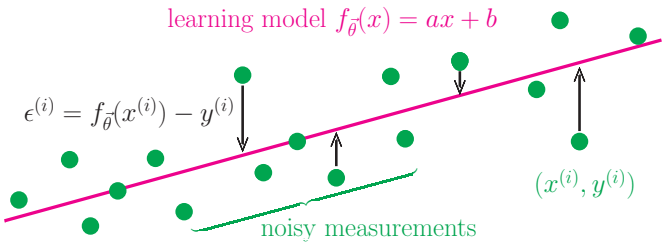

Fig. 1: Sketch for the linear regression, here the error for the data sample $x^{(i)}$ is defined as the difference between the model prediction $f_{\vec{\theta}}(x^{(i)})$ and measurement $y^{(i)}$.

To this end, we assume that the model is linear and denote it by $f_{\vec{\theta}}(x) = ax+b$ with a and b two parameters to be determined by the linear regression, the parameters are collectively denoted by $\vec{\theta} = (a, b)$. In order to obtain the model parameter $\vec{\theta}$, one needs to minimize the error between the model prediction for the data $x^{(i)}$, i.e., $f_{\vec{\theta}}(x^{(i)})$, and the measurement $y^{(i)}$. One of the frequently-used error is the squared loss, i.e., $(f_{\vec{\theta}}(x^{(i)}) - y^{(i)})^2$, here $f_{\vec{\theta}}(x^{(i)}) - y^{(i)}$ is called the *algebraic distance* between the model prediction and the measurement (which could take either positive or negative values), see Fig. 1 for the sketch of algebraic distance. The *total loss* is defined as the sum of the error of all samples

$$J(\vec{\theta}) = \frac{1}{2}\sum_{i=1}^{m}\left[f_{\vec{\theta}}(x^{(i)}) - y^{(i)}\right]^2. \tag{1}$$

Example. For two measurements, $(1, 2)$ and $(2, 5)$, i.e., $x^{(1)} = 1, y^{(1)} = 2$ and $x^{(2)} = 2, y^{(2)} = 5$, and $m = 2$, we have

$$\begin{aligned}
\langle x\rangle &= \frac{1}{2}\left(x^{(1)} + x^{(2)}\right) = 3/2,\\
\langle y\rangle &= \frac{1}{2}\left(y^{(1)} + y^{(2)}\right) = 7/2,\\
\langle x^2\rangle &= \frac{1}{2}\left(x^{(1),2} + x^{(2),2}\right) = 5/2,\\
\langle y^2\rangle &= \frac{1}{2}\left(y^{(1),2} + y^{(2),2}\right) = 29/2,\\
\langle xy\rangle &= \frac{1}{2}\left(x^{(1)}y^{(1)} + x^{(2)}y^{(2)}\right) = 6.
\end{aligned}$$

The factor $1/2$ is irrelevant here. It should be pointed out that J is a *function of $\vec{\theta}$ or equivalently of a and b, instead of $x^{(i)}$ or $y^{(i)}$*.

Other forms of the loss function exist, e.g., see the first problem of this lecture.

In order to obtain the parameters a and b, one needs to minimize the function J. Since J is a convex function (like the parabolic function x^2), the min-

imization of J is reduced to $\partial J/\partial a = 0$ and $\partial J/\partial b = 0$, these two equations determine the optimized parameters a^* and b^*. By expanding the loss function $J(\vec{\theta})$, we obtain

$$J(\vec{\theta}) = \frac{m}{2}\left[\langle x^2\rangle a^2 + b^2 + \langle y^2\rangle + 2\langle x\rangle ab - 2\langle xy\rangle a - 2\langle y\rangle b\right], \tag{2}$$

where the data sample averages are defined by,

$$\langle x\rangle = \frac{1}{m}\sum_{i=1}^{m} x^{(i)}, \quad \langle y\rangle = \frac{1}{m}\sum_{i=1}^{m} y^{(i)}, \tag{3}$$

$$\langle x^2\rangle = \frac{1}{m}\sum_{i=1}^{m} x^{(i),2}, \quad \langle y^2\rangle = \frac{1}{m}\sum_{i=1}^{m} y^{(i),2}, \quad \langle xy\rangle = \frac{1}{m}\sum_{i=1}^{m} x^{(i)}y^{(i)}, \tag{4}$$

which could be calculated once the measurement is available. After some long but straightforward calculations, one could prove that the a and b are given by,

$$a^* = \frac{\langle xy\rangle - \langle x\rangle\langle y\rangle}{\langle x^2\rangle - \langle x\rangle^2}, \quad b^* = \frac{\langle x^2\rangle\langle y\rangle - \langle x\rangle\langle xy\rangle}{\langle x^2\rangle - \langle x\rangle^2}. \tag{5}$$

In fact, if $y \sim x$, the numerator of $b^* \sim \langle x\rangle\langle x^2\rangle - \langle x^2\rangle\langle x\rangle = 0$.

We simulate the model and prepare the data as follows: Assume that the physical model is given by $y = 3x$, i.e., ideally a is 3 and b is zero. The data is generated through $y^{(i)} = a' x^{(i)}$ with $a' = 3 \pm \Delta$ where Δ denotes the noise, e.g., Δ is a zero-mean random variable with Gaussian distribution $\Delta \sim \mathcal{N}(0, \sigma^2)$, and consequently $a' = \mathcal{N}(3, \sigma^2)$. In our calculations, we fix $\sigma^2 = 0.3$. In addition, the data is generated by using this a' and the $x^{(i)}$ uniformly distributed within 0 and 5, i.e., $x^{(i)} \sim \mathrm{Unif}[0, 5]$. In Fig. 2 we show the simulated samples ($m = 50$), the fitting line $a^* x + b^*$ and the physical model used by randomly running the code. It is clearly shown that the *"learned model $a^* x + b^*$" has some deviation from the physical model (here $3x$).* Specifically, b^* is not exactly equal to zero.

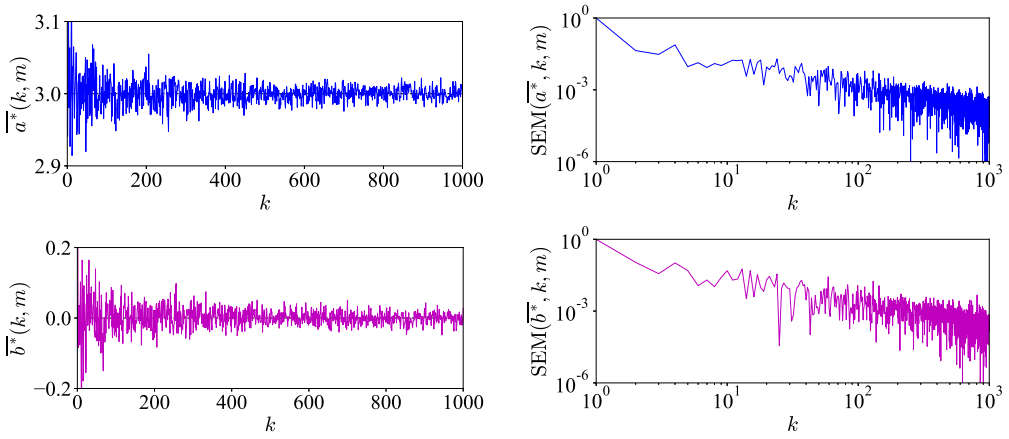

Fig. 3: The k-dependence of $\overline{a^*}(k,m)$, $\overline{b^*}(k,m)$, SEM$(\overline{a^*},k,m)$ and SEM$(\overline{b^*},k,m)$, $m = 10$ is fixed.

We can independently run the simulation for k times and obtain the a^* and b^* for k times. Consequently, the k-averages of a^* and b^* are calculated as,

$$\overline{a^*}(k,m) \equiv \frac{1}{k}\sum_{j=1}^{k} a^{*,(j)}, \quad \overline{b^*}(k,m) \equiv \frac{1}{k}\sum_{j=1}^{k} b^{*,(j)}. \tag{6}$$

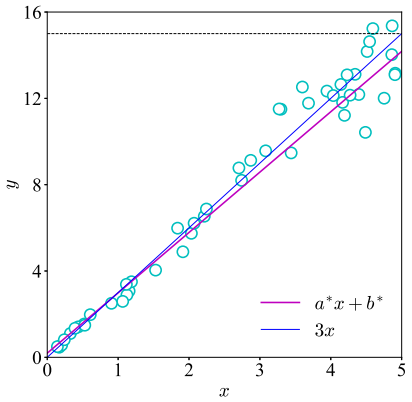

Fig. 2: The simulated samples, the fitting line $a^* x + b^*$ and physical model. The number of data samples is $m = 50$.

The coefficients a^* could be written as

$$a^* = \mathrm{cov}[x,y]/\mathrm{var}[x],$$

and consequently the optimal prediction for the output is given by

$$y^* = \mathrm{E}[y] + \frac{\mathrm{cov}[x,y]}{\mathrm{var}[x]}\left[x - \mathrm{E}[x]\right].$$

The mean square error (MSE) of observation is define as

$$\Delta^* = \mathrm{E}\left[y - y^*\right]^2 = \mathrm{var}[y]\left[1 - \rho^2[x,y]\right],$$

where $\rho[x,y]$ is the correlation between x and y. Consequently, the larger (in absolute value) the correlation coefficient the smaller the MSE of observation. In particular if $|\rho(x,y)| = 1$ then $\Delta^* = 0$, on the other hand if x's and y's are uncorrelated, $\Delta^* = \mathrm{var}[y]$ and $y^* = \mathrm{E}[y]$.

The corresponding k-dependence of the $\overline{a^*}$ and $\overline{b^*}$ is shown in the upper two panels of Fig. 3 where $m = 10$ is fixed for each simulation. As the k increases the k-average of the learning parameters eventually approach to the physical values. Similarly, by defining the *standard error of the mean (SEM)* of $\overline{a^*}$ and $\overline{b^*}$, we obtain

EXERCISE 1: Explore the quasi-linear dependence of Fig. 3.

$$\mathrm{SEM}(\overline{a^*}, k, m) = \sqrt{\frac{1}{k}\frac{1}{k-1}\sum_{j=1}^{k}\left(a^{*,(j)} - \overline{a^*}\right)^2}, \tag{7}$$

$$\mathrm{SEM}(\overline{b^*}, k, m) = \sqrt{\frac{1}{k}\frac{1}{k-1}\sum_{j=1}^{k}\left(b^{*,(j)} - \overline{b^*}\right)^2}, \tag{8}$$

one could study, e.g., the large-k behavior of the overall errors encapsulated in the learning parameters. We show in the lower two panels of Fig. 3 the k-dependence of the SEM for $\overline{a^*}$ and $\overline{b^*}$ while fixing $m = 10$. From the figure it is clearly shown that in the log-log plot *the overall tendency is quasi-linear.*

§2 Basic Concepts of Machine Learning

The linear learning model $f_{\vec{\theta}}(x) = ax + b$ is simple and convenient to implement, however it is sometimes too simple to capture other features encapsulated in the data, e.g., *the irregularity and/or the nonlinearity,* see Fig. 4 for an example. It is certain that the linear learning model could hardly work for these situations, where one needs to develop new learning techniques encapsulating the nonlinearity of the features.

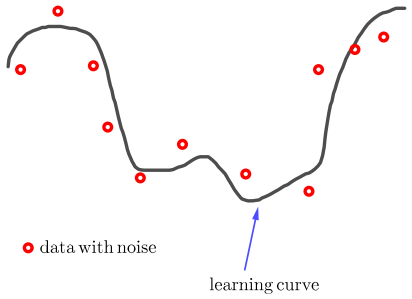

Fig. 4: Nonlinear curve fitting.

A few basic concepts are necessarily needed to be introduced. There is *a physical model, denoted by $f_{\mathrm{phys}}(\mathbf{x})$ (generally the input data $\mathbf{x}$ has the vector nature),* which is unknown in advance and maybe even very complicated. Although *the physical model $f_{\mathrm{phys}}(\mathbf{x})$ is unknown,* there exist relevant data generated by the model, and the data always has *measurement noise* (also unknown). We denote the data point as $(\mathbf{x}^{(i)}, y^{(i)})$, and since there exists noise in the output $y^{(i)}$ generally $y^{(i)} \neq f_{\mathrm{phys}}(\mathbf{x}^{(i)})$. For simplicity here we assume the output is a scalar. Besides the physical model, *a learning model denoted by $f_{\mathbf{w}}(\mathbf{x})$* is often introduced with $\mathbf{w}$ a set of parameters characterizing the learning model. Based on a fixed learning model, one could make *a prediction on each input data $\mathbf{x}^{(i)}$* to obtain $f_{\mathbf{w}}(\mathbf{x}^{(i)})$. *The learning model is an effective approximation of the physical model since the latter is generally very complicated.* Generally, the prediction $f_{\mathbf{w}}(\mathbf{x}^{(i)})$ will be different from the measured output data $y^{(i)}$, and the basic task in machine learning and/or data analysis/mining is to *minimize the difference between the measurement and the prediction.* Consequently, *an error function (or cost/loss function)* emerges, characterizing the above difference.

A very frequently-used error function is given by

$$J(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{m}\left[f_{\mathbf{w}}(\mathbf{x}^{(i)}) - y^{(i)}\right]^2, \tag{9}$$

Sometimes we also adopt the abbreviation for $f_{\mathbf{w}}(\mathbf{x}^{(i)})$ as $f_{\mathbf{w}}^{(i)}$, and under this notation,

$$J(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{m}\left(f_{\mathbf{w}}^{(i)} - y^{(i)}\right)^2.$$

i.e., the sum of the difference between the prediction $(f_{\mathbf{w}}(\mathbf{x}^{(i)}))$ and the measurement $(y^{(i)})$, which has already been used in the linear learning model, see (1). *This type of optimization is called the least-squares (LS).* As similar as in the linear fitting problem, $J(\mathbf{w})$ is function of the learning parameter $\mathbf{w}$, and not a function of the measurements $(\mathbf{x}^{(i)}, y^{(i)})$. The next task is to minimize the error

function $J(\mathbf{w})$, and since generally $J(\mathbf{w})$ is a convex function of $\mathbf{w}$, the minimization of it is reduced to the condition $\partial J(\mathbf{w})/\partial \mathbf{w} = 0$, or equivalently,

$$\partial J(\mathbf{w})/\partial w_j = 0, \quad j = 0, 1, 2, \cdots, n, \tag{10}$$

where one assumes that there are totally $n+1$ parameters namely w_0 to w_n. *It is necessary to point out that we use two letters "n" and "m" to represent the number of the learning parameters and the number of data points, respectively*. Specifically, if the input data has the scalar nature (i.e., x instead of $\mathbf{x}$), a very general learning model in data analysis and machine learning is the *polynomial of order n,* i.e.,

$$f_{\mathbf{w}}(x) = w_0 + w_1 x + w_2 x^2 + \cdots + w_n x^n = \sum_{j=0}^{n} w_j x^j, \tag{11}$$

here the learning parameter consists of $n+1$ scalars, i.e., w_0 to w_n. One uses the measured data points $(x^{(i)}, y^{(i)})$ to study these parameters, which is then called *"learning from data". Although $f_{\mathbf{w}}(x)$ is nonlinear in x, it is still linear in the parameters w_j. In this sense, we still call $f_{\mathbf{w}}(x)$ the linear model.*

§3 Trade-off between Bias and Variance

We solve the nonlinear curve fitting problem in details to demonstrate the important features of machine learning, i.e., how could one "learn from data", and what to learn?

According to Eq. (10), one can obtain the equations for determining the model parameters $w_0, w_1, w_2, \cdots, w_n$,

$$\begin{cases} \langle 1\rangle w_0 + \langle x\rangle w_1 + \langle x^2\rangle w_2 + \cdots + \langle x^n\rangle w_n = \langle y\rangle, \\ \langle x\rangle w_0 + \langle x^2\rangle w_1 + \langle x^3\rangle w_2 + \cdots + \langle x^{n+1}\rangle w_n = \langle xy\rangle, \\ \vdots, \\ \langle x^n\rangle w_0 + \langle x^{n+1}\rangle w_1 + \langle x^{n+2}\rangle w_2 + \cdots + \langle x^{2n}\rangle w_n = \langle x^n y\rangle, \end{cases} \tag{12}$$

where $\langle 1\rangle = m^{-1}\sum_{i=1}^{m} 1 = 1$, and

$$\langle x^k\rangle = \frac{1}{m}\sum_{i=1}^{m} x^{(i),k} = \frac{1}{m}\left(x^{(1),k} + x^{(2),k} + \cdots + x^{(m),k}\right), \tag{13}$$

$$\langle x^k y\rangle = \frac{1}{m}\sum_{i=1}^{m} x^{(i),k} y^{(i)} = \frac{1}{m}\left(x^{(1),k}y^{(1)} + x^{(2),k}y^{(2)} + \cdots + x^{(m),k}y^{(m)}\right). \tag{14}$$

As a special case, consider $n = 1$, i.e., for the linear fitting problem, Eq. (12) becomes $\langle 1\rangle w_0 + \langle x\rangle w_1 = \langle y\rangle$ and $\langle x\rangle w_0 + \langle x^2\rangle w_1 = \langle xy\rangle$. In addition, Eq. (12) could be rewritten in the form,

$$\mathbf{F}\mathbf{w} = \mathbf{G}, \tag{15}$$

by introducing

$$\mathbf{F} = \begin{pmatrix} \langle 1\rangle & \langle x\rangle & \cdots & \langle x^n\rangle \\ \langle x\rangle & \langle x^2\rangle & \cdots & \langle x^{n+1}\rangle \\ \vdots & \vdots & \ddots & \vdots \\ \langle x^n\rangle & \langle x^{n+1}\rangle & \cdots & \langle x^{2n}\rangle \end{pmatrix}, \quad \mathbf{w} = \begin{pmatrix} w_0 \\ w_1 \\ \vdots \\ w_n \end{pmatrix}, \quad \mathbf{G} = \begin{pmatrix} \langle y\rangle \\ \langle xy\rangle \\ \vdots \\ \langle x^n y\rangle \end{pmatrix}. \tag{16}$$

The second order derivative of the loss function with respect to w_j is given by

$$\begin{aligned} \frac{\partial^2 J}{\partial w_j^2} =& \frac{\partial}{\partial w_j}\frac{\partial}{\partial w_j}\left[\frac{1}{2}\sum_{i=1}^{m}\left[f_{\mathbf{w}}(\mathbf{x}^{(i)}) - y^{(i)}\right]^2\right] \\ =& \sum_{i=1}^{m}\left(\frac{\partial}{\partial w_j} f_{\mathbf{w}}(x^{(i)})\right)^2 \\ &+ \sum_{i=1}^{m}\left(f_{\mathbf{w}}(x^{(i)}) - y^{(i)}\right)\frac{\partial^2}{\partial w_j^2} f_{\mathbf{w}}(x^{(i)}). \end{aligned}$$

If the learning model is linear then the second term is zero at the optimal parameter, leading to

$$\frac{\partial^2 J}{\partial w_j^2} = \sum_{i=1}^{m}\left(\frac{\partial}{\partial w_j} f_{\mathbf{w}}(x^{(i)})\right)^2,$$

which is positive. Even in the situation where the learning model is nonlinear, we can still prove that $\partial^2 J/\partial w_j > 0$, using the normal equation given in the following part.

Specifically, we have

$$\begin{aligned} \frac{\partial J}{\partial w_j} =& \sum_{i=1}^{m}\left(f_{\mathbf{w}}(x^{(i)}) - y^{(i)}\right)\frac{\partial}{\partial w_j} f_{\mathbf{w}}(x^{(i)}) \\ =& \sum_{i=1}^{m} f_{\mathbf{w}}(x^{(i)})x^{(i),j} - \sum_{i=1}^{m} y^{(i)}x^{(i),j} \\ =& \sum_{i=1}^{m}\sum_{j'=0}^{n} w_{j'}x^{(i),j'+j} - \sum_{i=1}^{m} y^{(i)}x^{(i),j} \\ =& \sum_{j'=0}^{n} w_{j'}\sum_{i=1}^{m} x^{(i),j'+j} - \sum_{i=1}^{m} y^{(i)}x^{(i),j} \\ =& m\left(\sum_{j'=0}^{n} w_{j'}\langle x^{j'+j}\rangle - \langle x^j y\rangle\right), \end{aligned}$$

and the relevant equation is given by setting it to be zero, i.e.,

$$\sum_{j'=0}^{n} w_{j'}\langle x^{j'+j}\rangle = \langle x^j y\rangle, \quad j = 0 \sim n.$$

EXERCISE 2: Derive the analytical expressions for w_j's in the case of $n = 2$ and $n = 3$, write down the form of $\mathbf{F}^{-1}$.

EXERCISE 3: Prove $\mathbf{F}$ could be written as $\vec{\Phi}^{\top}\vec{\Phi}$ where the element of the matrix $\vec{\Phi}$ is defined as $\phi_{ji} = \phi_j(x^{(i)})$ with $j = 0 \sim n$ and $i = 1 \sim m$, see Eq. (32).

Note that the $(n+1)\times(n+1)$ matrix $\mathbf{F}$ is symmetric, namely $F_{ij} = F_{ji}$. The symmetry properties of $\mathbf{F}$ is useful for calculating its inverse.

We similarly prepare the data to be used in the simulation. Here the physical model is adopted as $f_{\rm phys}(x) = \sin(2\pi x)$ with $0 \le x \le 1$, and we generate total 10 data points uniformly distributed within this range, i.e., $x^{(i)} = i/9, i = 0,1,2,\cdots,9$, or, $x^{(i)} = 0, 1/9, 2/9, \cdots, 8/9, 1$. The output data $y^{(i)}$ is obtained from the physical model by including a noise, i.e., $y^{(i)} = \sin(2\pi x^{(i)}) + a^{(i)}$, here $a^{(i)} \sim {\rm Unif}[-\ell,\ell]$ is a uniformly distributed random number, see Fig. 5 for the physical model (shown as the dashed black line) and the generated data points (as the magenta circles), where $\ell = 0.8$ is used. After solving the equation (15), one obtains the parameter $\mathbf{w}^*$ and consequently the learning model $f_{\mathbf{w}^*}(x)$.

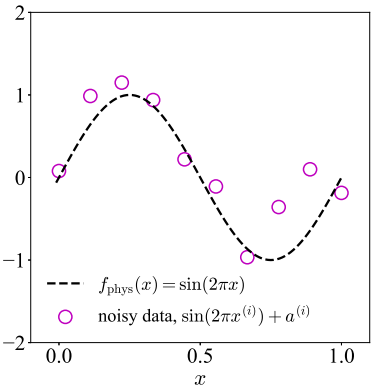

Fig. 5: Data preparation for nonlinear learning model, $\ell = 0.8$.

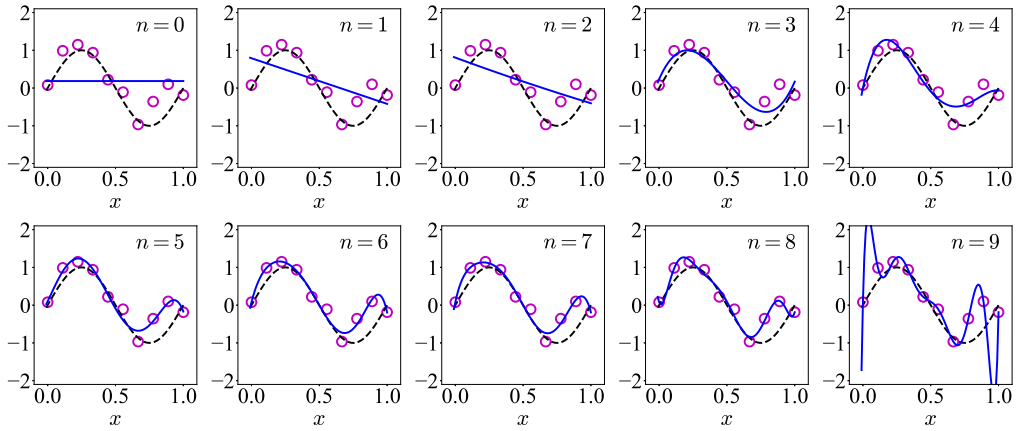

Fig. 6: Learning processes with different n.

	$n=0$	$n=1$	$n=2$	$n=3$	$n=4$	$n=5$	$n=6$	$n=7$	$n=8$	$n=9$
w_0^*	0.19	0.79	0.81	0.23	−0.00	0.10	0.08	0.08	0.08	0.08
w_1^*	∅	−1.21	−1.32	8.11	16.93	7.42	13.99	16.69	−9.48	145.11
w_2^*	∅	∅	0.11	−24.74	−69.30	10.00	−71.10	−115.61	412.07	−3158.52
w_3^*	∅	∅	∅	16.57	88.03	−136.10	212.92	476.91	−3503.21	28658.38
w_4^*	∅	∅	∅	∅	−35.73	221.18	−453.31	−1199.76	13815.03	−137275.68
w_5^*	∅	∅	∅	∅	∅	−102.76	497.20	1585.76	−29673.59	381953.29
w_6^*	∅	∅	∅	∅	∅	∅	−199.99	−989.95	35479.27	−638534.33
w_7^*	∅	∅	∅	∅	∅	∅	∅	225.70	−22102.39	631648.54
w_8^*	∅	∅	∅	∅	∅	∅	∅	∅	5582.02	−340299.59
w_9^*	∅	∅	∅	∅	∅	∅	∅	∅	∅	76862.54

Tab. 1: Optimal parameters w_j^*, $j = 0 \sim n$ in different learning models.

Results for a series of learning model $f_{\mathbf{w}^*}(x)$ with different parameter n are shown in Fig. 6. Let's discuss starting with the case "$n = 0$", now the learning model is $f_{\mathbf{w}}(x) = w_0$ and in this case the optimized parameter w_0^* is just the mean of the output $y^{(i)}$, namely $w_0^* = (y^{(1)} + y^{(2)} + \cdots + y^{(10)})/10$. Next, we have re-obtained the linear fitting result if $n = 1$. There are several novel and important features shown in Fig. 6. Firstly, at small n the learning curve (blue line) predicts poorly for the data points, and the prediction becomes better and better when n increases. In the limit situation of $n = 9$, the prediction can perform perfectly on the data points *(there is no difference between the measurement $y^{(i)}$ and the prediction $f_{\mathbf{w}}(x^{(i)})$ at these points).* However, the learning curve becomes very strange at these larger n, while it is much smoother with smaller n.

EXERCISE 4: The parameter w_0^* given in Tab. 1 under different n shows similar values. However the value for w_j^* with $j \ge 1$ changes a lot as n increases. Explain the possible reason.

This indicates the model behavior under consideration.

The smoothness and the strangeness characterize two important aspects of the learning model: *When the learning curve is smoother we call it has a smaller variance, while when the curve is closer to the measurements we call it has a smaller bias. Thus, when n is small, the learning model has a small variance and a large bias and the learning model has a large variance and a small bias when n is large.* This phenomenon is sometimes called the *"no-free-lunch theorem"*, in the sense that one could not obtain a learning model with both small variance and small bias. It is a very general feature of all the learning problems in data analysis. *The "no-free-lunch theorem" is also called the bias-variance trade-off or the bias-variance decomposition.* The learning model with large n is very complicated and has a very strong power of fitting data but limited power of predicting new data. We thus often *call n the complexity of the model.* In our example, $n \approx 3 \sim 6$, is reasonable. In Tab. 1, the values of the optimal $\mathbf{w}_j^*$ are shown in models with different n. One of the important features is that as the model complexity n increases the magnitude of w_j^*'s eventually increase. It is very dangerous in the sense that in order to finally obtain a naturally prediction on the output the large terms $w_j^* x^j$ are added, and this phenomenon is called *fine-tuning.* One of the popular methods to avoid the fine-tuning is through the regularization term introduced into the model. We will introduce it in the following paragraphs.

The bias-variance decomposition is a general result in data fitting problems, independent of the learning model adopted, see Fig. 7 for four popular patterns of bias and variance. If the physical model for the quantity x is as before denoted as $f_{\text{phys}}(x)$, and the output generated by x is $y = f_{\text{phys}}(x) + a$, where a is a noise with mean $\mathrm{E}[a] = 0$ and variance $\mathrm{var}[a]$. In addition, the learning model is denoted by $\widehat{f}(x)$ without introducing the learning parameter $\mathbf{w}$. For each testing data $\overline{x}$, the output given by the learning model is consequently $\widehat{f}(\overline{x})$. After some straightforward derivations we could obtain the mean of *the square of the difference between the physical model and the learning prediction* $\Delta = \mathrm{E}[(f_{\text{phys}}(\overline{x}) + a - \widehat{f}(\overline{x}))^2]$, which characterizes the goodness of the model,

$$
\begin{aligned}
\Delta =& \mathrm{E}\left[\left(f_{\text{phys}}(\overline{x}) + a - \widehat{f}(\overline{x})\right)^2\right] \\
=& \mathrm{E}\left[f_{\text{phys}}^2(\overline{x}) + a^2 + \widehat{f}^2(\overline{x}) + 2a f_{\text{phys}}(\overline{x}) - 2a\widehat{f}(\overline{x}) - 2f_{\text{phys}}(\overline{x})\widehat{f}(\overline{x})\right] \\
=& \mathrm{E}\left[f_{\text{phys}}^2(\overline{x}) + a^2 + \widehat{f}^2(\overline{x}) - 2f_{\text{phys}}(\overline{x})\widehat{f}(\overline{x})\right] \\
=& \mathrm{E}\left[a^2\right] + \mathrm{E}\left[f_{\text{phys}}^2(\overline{x})\right] + \mathrm{E}\left[\widehat{f}^2(\overline{x})\right] - 2\mathrm{E}\left[f_{\text{phys}}(\overline{x})\widehat{f}(\overline{x})\right] \\
=& \mathrm{E}\left[a^2\right] - \mathrm{E}^2[a] + \mathrm{E}\left[f_{\text{phys}}^2(\overline{x})\right] + \mathrm{E}\left[\widehat{f}^2(\overline{x})\right] - 2\mathrm{E}\left[f_{\text{phys}}(\overline{x})\widehat{f}(\overline{x})\right] + \mathrm{E}^2\left[\widehat{f}(\overline{x})\right] - \mathrm{E}^2\left[\widehat{f}(\overline{x})\right] \\
=& \mathrm{var}[a] + f_{\text{phys}}^2(\overline{x}) + \mathrm{E}\left[\widehat{f}^2(\overline{x})\right] - 2f_{\text{phys}}(\overline{x})\mathrm{E}\left[\widehat{f}(\overline{x})\right] + \mathrm{E}^2\left[\widehat{f}(\overline{x})\right] - \mathrm{E}^2[\widehat{f}(\overline{x})] \\
=& \mathrm{var}[a] + \mathrm{E}^2\left[\widehat{f}(\overline{x})\right] - 2f_{\text{phys}}(\overline{x})\mathrm{E}\left[\widehat{f}(\overline{x})\right] + f_{\text{phys}}^2(\overline{x}) + \mathrm{E}\left[\widehat{f}^2(\overline{x})\right] - \mathrm{E}^2\left[\widehat{f}(\overline{x})\right] \\
=& \mathrm{var}[a] + \left[\mathrm{E}\left[\widehat{f}(\overline{x})\right] - f_{\text{phys}}(\overline{x})\right]^2 + \mathrm{E}\left[\left[\widehat{f}(\overline{x}) - \mathrm{E}\left[\widehat{f}(\overline{x})\right]\right]^2\right], \qquad (17)
\end{aligned}
$$

the meaning of each term is clear:

(a) The noise $\mathrm{var}[a]$ could not be reduced once the physical model is fixed.

(b) $\mathrm{E}[\widehat{f}(\overline{x})] - f_{\text{phys}}(\overline{x})$ is the bias between the learning model $\widehat{f}$ (characterized by its mean $\mathrm{E}[\widehat{f}(\overline{x})]$) and the physical model f_{phys}.

(c) $\mathrm{E}[[\widehat{f}(\overline{x}) - \mathrm{E}[\widehat{f}(\overline{x})]]^2]$ is the variance of the learning model $\widehat{f}$ on the testing data.

See, e.g., D. Wolpert and W. Macready, *No Free Lunch Theorems for Optimization,* IEEE Transactions on Evolutionary Computation **1**, 67 (1997).

An example on fine-tuning is the following quantity

$$u = \sqrt{x+1} - \sqrt{x} = \frac{1}{\sqrt{x+1} + \sqrt{x}},$$

where, e.g., x is a very large number. The first method to calculate d is dangerous if x is very large, i.e., rounding error may emerge. However the second method for calculating u is safe, namely it would not produce rounding error.

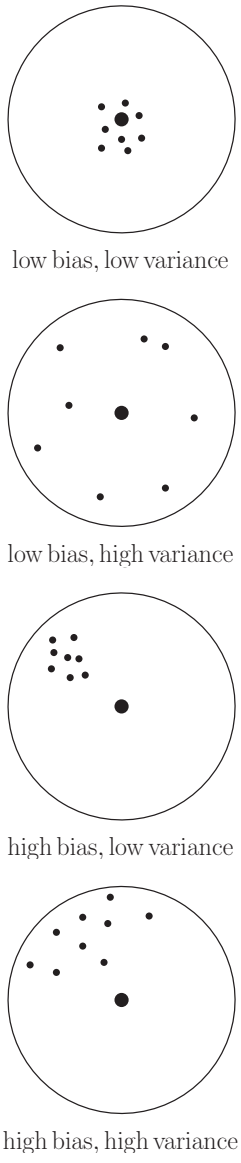

Fig. 7: Patterns of bias and variance.

thus

$$\mathrm{E}\left[\left(f_{\mathrm{phys}}(\overline{x})+a-\widehat{f}(\overline{x})\right)^2\right]=\mathrm{var}[a]+\left[\text{bias of }\widehat{f}(\overline{x})\right]^2+\text{variance of }\widehat{f}(\overline{x}), \tag{18}$$

or

$$(\text{total error})^2=\text{noise}+\text{bias}^2+\text{variance}. \tag{19}$$

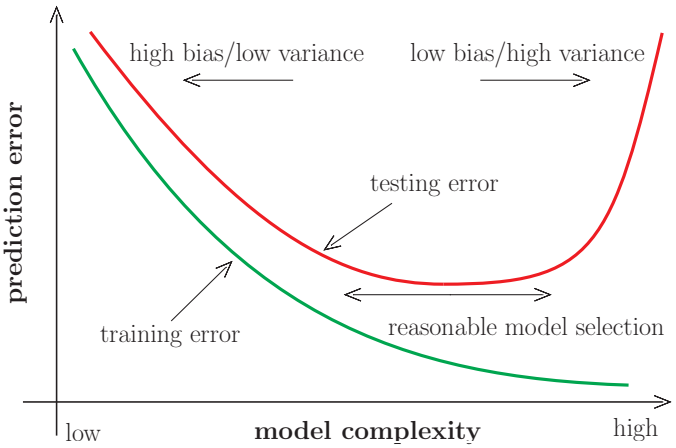

Fig. 8: Sketch of the bias-variance decomposition.

It is the bias-variance decomposition of the learning process, which partially explains that *if the bias of the learning is small then the variance is correspondingly large and vice versa, since the noise* var[a] *is generally a constant once the physical model is fixed.* See Fig. 8 for the sketch of the bias-variance decomposition of the learning model. In fact, there is *no prior* that the total error should be decomposed into the variance and the bias. A natural question is that could the bias and the variance be small simultaneously? Problems like these are at the center of modern deep learning theory. For example, in some model studies, the "double descent" for the prediction error is found, indicating that the testing error could be reduced even to be very small in the over-parametrized region. Deep understanding on the current neural networks is an important and exciting problem, we have no attempts to review the status on this issue, e.g., some people use ideas from physics like the renormalization group and the effective field theories to deal with the neural networks.

See, e.g., M. Belkin *et al.*, *Reconciling Modern Machine Learning Practice and the Classical Bias-variance Trade-off*, PNAS, **116**, 15849 (2019).

H.W. Lin, M. Tegmark, and D. Rolnick, *Why Does Deep and Cheap Learning-Work So Well?*, J. Stat. Phys. **168**, 1223 (2017), in this paper the idea of effective field theories was applied; C. Beny, *Deep learning and the Renormalization Group*, arXiv:1301.3124 (2013), in this work the renormalization group technique was adopted.

§4 Training Error and Testing Error

In order to characterize the bias-variance decomposition in a more qualitatively manner, we define two errors both based on Eq. (9). We already have $m=10$ data points (denoted as the training data), we define the *training error per data sample* as

$$e_{\mathrm{train}}\equiv\frac{1}{m}\left[\frac{1}{2}\sum_{i=1}^{m}\left[f_{\mathbf{w}^*}(x^{(i)})-y^{(i)}\right]^2\right], \tag{20}$$

where the optimized $\mathbf{w}^*$ is used. For $n=9$, the e_{train} will be zero since the learning model can exactly pass through all the training data points. Besides these already existed training data, one could randomly generate another m' data points (different from the training data) according to $f_{\mathrm{phys}}(x^{(i')})+a^{(i')}$ where both

$x^{(i')}$ and $a^{(i')}$ are random, and define the *testing error per data sample* as

$$e_{\text{test}} \equiv \frac{1}{m'}\left[\frac{1}{2}\sum_{i=1}^{m'}\left[f_{\mathbf{w}^*}(x^{(i')}) - y^{(i')}\right]^2\right]. \tag{21}$$

It is reasonable to expect that the testing error for the situation with $n = 9$ would be larger than that in the case $n = 3$.

Selecting a model with a certain n is called model selection. A very simple scheme to select a reasonable n is to select the learning model with a smallest training error, a smallest testing error, or a total error defined as the weighted sum of the training and the testing error, i.e., $e_{\text{total}} = he_{\text{train}} + (1-h)e_{\text{test}}$, where $0 \le h \le 1$. Taking $h = 1/2$ means that the training error and the testing error have the same weights, otherwise they have different weights. Another scheme characterizing the learning model on the prediction power is to calculate the *average (overall) deviation between the physical model $f_{\text{phys}}(x)$ and the learning model $f_{\mathbf{w}}(x)$,* for our example, we have the following *integration error,*

$$\overline{\text{DE}} = \int_0^1 \left|f_{\text{phys}}(x) - f_{\mathbf{w}^*}(x)\right| \mathrm{d}x. \tag{22}$$

In Fig. 9 the above errors are shown as functions of n where $m' = 5$ is adopted. For the current problem we find that $n \approx 3 \sim 6$ is reasonable.

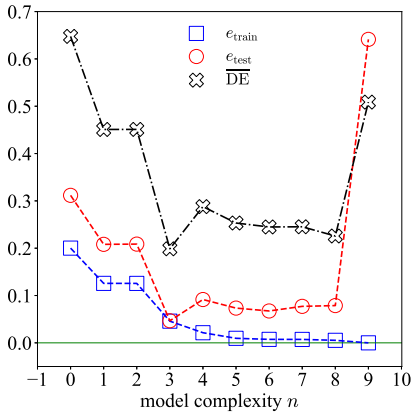

Fig. 9: Training error e_{train}, testing error e_{test} and the integration error $\overline{\text{DE}}$.

§5 Regularization (Penalty) Term(s)

As discussed above and shown in Fig. 9 if the *model complexity n* is selected reasonably, e.g., $n = 3$ or $n = 4$, the learning curve has both low training and testing errors. On the other hand, if n is very small, e.g., $n = 0$, the learning model $f_{\mathbf{w}^*}(x) = w_0^*$ has both large training and testing errors. If n is too large like $n = 9$ the learning model has *zero training error*. However the testing error is now extremely large, indicating that the model *has very poor prediction power* although it could pass through the training data perfectly. In addition, in the small n case (e.g., $n = 0$ or $n = 1$), the learning model is very poor in grasping the training data, and we call this situation is *under-fitting.* On the other hand, in the large n case (e.g., $n = 9$), the learning model is very strong in grasping the training data (but very poor in predicting new samples), and we say this situation is *over-fitting*. Either under-fitting or over-fitting is bad. *The overall performance of model is characterized by the prediction error.*

We introduce the *regularization method* to deal with the over-fitting problem. Besides the original loss $2^{-1}\sum_{i=1}^{m}[f_{\mathbf{w}}(x^{(i)}) - y^{(i)}]^2$, we introduce an extra term:

$$J(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{m}\left[f_{\mathbf{w}}(x^{(i)}) - y^{(i)}\right]^2 + \lambda g(\mathbf{w}), \tag{23}$$

where $\lambda > 0$ is a parameter *put by hand* and $g(\mathbf{w})$ is a function of $\mathbf{w}$, which is positive. We call $\lambda g(\mathbf{w})$ the *regularization term* to the loss function $J(\mathbf{w})$. There are many different forms of the regularization term and different form has different realistic meaning. Here we adopt the following regularization term,

$$\lambda g(\mathbf{w}) = \frac{1}{2}\lambda \mathbf{w}^\top \mathbf{w} = \frac{1}{2}\lambda\left(w_0^2 + w_1^2 + \cdots + w_n^2\right), \tag{24}$$

EXERCISE 5: Define the generalized integration error as $\overline{\text{DE}}(\nu) = \int |f_{\text{phys}}(\mathbf{x}) - f_{\mathbf{w}^*}(x)|^\nu \mathrm{d}x$, investigate its n-dependence for the polynomial model for $\nu = 2$ and $\nu = 3$.

EXERCISE 6: Change the noise $a^{(i)}$ to be sampled from the uniform distribution $\text{Unif}[-\ell, \ell]$ with ℓ a large number (e.g., 10) in the polynomial learning model, investigate the learning properties. In this case the noise $a^{(i)}$ itself exceeds the physical model $\sin(2\pi x^{(i)})$ and consequently the measurements $y^{(i)}$ are almost characterized by the noise. It tells us that noise has no regular behavior, and thus can not be studied via learning algorithms.

EXERCISE 7: The ℓ-1 norm is defined as $|\mathbf{w}|_1 = \sum_{j=0}^{n}|w_j|$. Explore the geometrical meaning of the ℓ-1 norm on the learning problem.

EXERCISE 8: Write down the equation for $\mathbf{w}^*$ if the regularization term is taken as $\lambda(\mathbf{w}^2)^2$, where $(\mathbf{w}^2)^2 = (w_0^2 + w_1^2 + \cdots + w_n^2)^2$. What's about $\mathbf{w}^4 = w_0^4 + w_1^4 + \cdots + w_n^4$? Moreover, what's about if the regularization term is $\lambda \mathbf{w}^\top \mathbf{T}\mathbf{w}$ with $\mathbf{T}$ a positive-definite matrix? Investigate the relevant learning properties.

which is called the *ℓ-2 regularization term.* Another important penalty term is the ℓ-1 form, which is used to guarantee the sparsity of the learning model.

The equation determining the parameter $w_0, w_1, \cdots, w_n$ in the presence of the regularization terms could be derived similarly as

$$\begin{pmatrix} \langle 1\rangle & \langle x\rangle & \cdots & \langle x^n\rangle \\ \langle x\rangle & \langle x^2\rangle & \cdots & \langle x^{n+1}\rangle \\ \vdots & \vdots & \ddots & \vdots \\ \langle x^n\rangle & \langle x^{n+1}\rangle & \cdots & \langle x^{2n}\rangle \end{pmatrix} \begin{pmatrix} w_0 \\ w_1 \\ \vdots \\ w_n \end{pmatrix} = \begin{pmatrix} \langle y\rangle \\ \langle xy\rangle \\ \vdots \\ \langle x^n y\rangle \end{pmatrix} - \lambda \begin{pmatrix} w_0 \\ w_1 \\ \vdots \\ w_n \end{pmatrix}, \tag{25}$$

or$(\mathbf{F} + \lambda \mathbf{1})\,\mathbf{w} = \mathbf{G}$, where $\mathbf{1}$ is the $(n+1)\times(n+1)$ unit matrix.

In Fig. 10 the effects of the regularization term with $\lambda = 10^{-12}$ are shown based on the 9th-order polynomial learning model. It is found that a tiny $\lambda = 10^{-12}$ essentially makes the overall behavior of the learning curve more regular and smoother, and one can expect as λ increases the curve becomes much smoother. It should be pointed out once again that *the regularization term $\lambda g(\mathbf{w})$ is not encapsulated in the data points themselves, instead it is put by hand.* In this sense the λ-term more or less characterizes people's belief in the data, i.e., *if λ is small the original data points are treated more importantly, and on the other hand, if λ is large it means that one does not believe the original data points.* A natural question is what will happen in this large λ limit. For example, by taking $\lambda = 1$ the w_j^*'s in the $n = 9$ model are found to be $0.44, -0.28, -0.31, -0.20, -0.10, -0.02, 0.03, 0.06, 0.08$ and 0.09, respectively. *The case $\lambda = \infty$ corresponds to the situation that one totally disbelieve the original data points,* since now

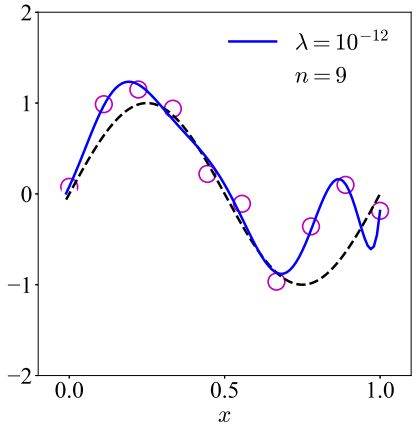

Fig. 10: Regularization term effect.

$$\lambda g(\mathbf{w}) \gg \frac{1}{2}\sum_{i=1}^{m}\left[f_{\mathbf{w}}(x^{(i)}) - y^{(i)}\right]^2, \tag{26}$$

and the learning model is approximately reduced to be $f_{\mathbf{w}}(x) = \lambda g(\mathbf{w})$ and if $g(\mathbf{w}) = \mathbf{w}^\top\mathbf{w}$ then the optimal $\mathbf{w}^*$ is essentially $\mathbf{0}$ without doubt. In Fig. 11 the training loss as well as the testing for the model with $n = 9$ are plotted as functions of $\ln\lambda$, where the testing sample number is $m' = 10^4$. As $\ln\lambda \to -\infty$, i.e., the limit without using the regularization term, the training loss tends to zero since when $n = 9$ the learning model could perfectly pass through all the data. In the meanwhile the testing loss is fixed at a nonzero constant. Under the opposite limit named $\ln\lambda \to \infty$, the learning model naturally becomes zero, and consequently either the training loss or the testing loss is fixed at constants which are determined by the simulated data points used.

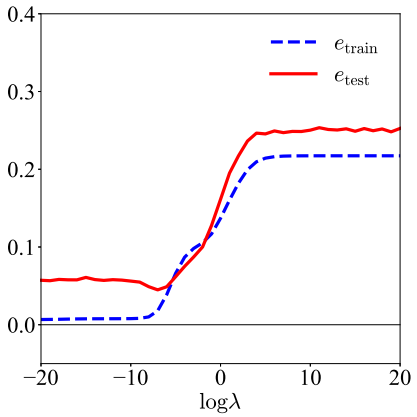

Fig. 11: Training and testing errors as functions of parameter λ, here $n = 9$.

§6 *Normal Equation and More Theoretical Discussions

Let's discuss the polynomial learning model in some more details. In our polynomial learning model, $f_{\mathbf{w}}(x) = w_0 + w_1 x + w_2 x^2 + \cdots + w_n x^n$, which could be rewritten in the form

$$f_{\mathbf{w}}(x) = \mathbf{w}^\top\vec{\phi}(x) = \vec{\phi}^\top(x)\mathbf{w} \tag{27}$$

with

$$\mathbf{w} = (w_0, w_1, w_2, \cdots, w_n)^\top, \quad \vec{\phi}(x) = \left(1, x, x^2, \cdots, x^n\right)^\top, \tag{28}$$

i.e., $\mathbf{w} \in \mathrm{R}^{(n+1)\times 1} \equiv \mathrm{R}^{n+1}$ is a column vector *(column vector is thin and tall)* and its transpose $\mathbf{w}^\top \in \mathrm{R}^{1\times(n+1)}$ is a row vector *(row vector is fat and short).* The jth component of the vector $\vec{\phi}$ is x^j with $j = 0 \sim n$. Denoting $\phi_j(x) = x^j$, then the

One has $\mathbf{a}^\top\mathbf{b} = \mathbf{b}^\top\mathbf{a}$ since this quantity is a scalar, i.e., $\sum_{i=1}^{d} a_i b_i$ with d the dimension of the vector $\mathbf{a}$ or $\mathbf{b}$.

function $\vec{\phi}(x)$ could be written

$$\vec{\phi}(x) = (\phi_0(x), \phi_1(x), \cdots, \phi_n(x))^\top, \quad \phi_j(x) = x^j, \quad j = 0 \sim n. \tag{29}$$

In the polynomial learning model, each component $\phi_j(x)$ takes the form x^j. However, in more general situations, the component $\phi_j(x)$ is free to take other forms, e.g., $\phi_j(x) = \sin(jtx)$ with t a constant. We call *the functions $\phi_j(x)$'s the basis functions.*

Adopting the basis function representation, the loss function without regularization term is given by

$$J(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{m}\left[f_{\mathbf{w}}(x^{(i)}) - y^{(i)}\right]^2, \quad f_{\mathbf{w}}(x^{(i)}) = \mathbf{w}^\top\vec{\phi}(x^{(i)}). \tag{30}$$

The loss function could be written as in another form,

$$J(\mathbf{w}) = \frac{1}{2}\left(\vec{\Phi}\mathbf{w} - \mathbf{y}\right)^\top\left(\vec{\Phi}\mathbf{w} - \mathbf{y}\right), \tag{31}$$

where

$$\vec{\Phi} = \begin{pmatrix} \phi_0(x^{(1)}) & \phi_1(x^{(1)}) & \cdots & \phi_n(x^{(1)}) \\ \phi_0(x^{(2)}) & \phi_1(x^{(2)}) & \cdots & \phi_n(x^{(2)}) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_0(x^{(m)}) & \phi_1(x^{(m)}) & \cdots & \phi_n(x^{(m)}) \end{pmatrix} = \begin{pmatrix} 1 & \phi_1(x^{(1)}) & \cdots & \phi_n(x^{(1)}) \\ 1 & \phi_1(x^{(2)}) & \cdots & \phi_n(x^{(2)}) \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \phi_1(x^{(m)}) & \cdots & \phi_n(x^{(m)}) \end{pmatrix} \in \mathrm{R}^{m\times(n+1)}, \tag{32}$$

and

$$\mathbf{y} = \left(y^{(1)}, y^{(2)}, \cdots, y^{(m)}\right)^\top \in \mathrm{R}^m. \tag{33}$$

We call *the matrix $\vec{\Phi}$ the design matrix with its component given by $\phi_j(x^{(i)})$.* The derivative of the loss $J(\mathbf{w})$ with respect to $\mathbf{w}$ is given by $\partial J(\mathbf{w})/\partial\mathbf{w} = \vec{\Phi}^\top\vec{\Phi}\mathbf{w} - \vec{\Phi}^\top\mathbf{y}$, then taking $\partial J(\mathbf{w})/\partial\mathbf{w}$ to be zero, namely $\vec{\Phi}^\top\vec{\Phi}\mathbf{w} = \vec{\Phi}^\top\mathbf{y}$, one obtains the solution of this *normal equation,*

$$\mathbf{w}^* = \left(\vec{\Phi}^\top\vec{\Phi}\right)^{-1}\vec{\Phi}^\top\mathbf{y}. \tag{34}$$

After the regularization term $2^{-1}\lambda\mathbf{w}^\top\mathbf{w}$ is included into the loss function $J(\mathbf{w})$, the optimized solution is changed to be

$$\mathbf{w}^* = \left(\vec{\Phi}^\top\vec{\Phi} + \lambda\mathbf{1}\right)^{-1}\vec{\Phi}^\top\mathbf{y}. \tag{35}$$

It should be point out that the regularization term introduced into the least-squares *plays an important role in situations where the matrix $\vec{\Phi}^\top\vec{\Phi}$ is singular,* i.e., $\det(\vec{\Phi}^\top\vec{\Phi}) = 0$. Choosing a large λ indicates that the original data information is put at the secondary position. Let's give more analysis on the normal equation. Assume that we want to find the minimum of the quadratic objective function $K(\mathbf{w}) = 2^{-1}\mathbf{w}^\top\vec{\Phi}\mathbf{w} - \mathbf{y}^\top\mathbf{w}$, the gradient of $K(\mathbf{w})$ is given by $\vec{\Phi}\mathbf{w} - \mathbf{y}$. In order to find the optimal $\mathbf{w}^*$ one naturally needs to solve this equation, i.e., $\vec{\Phi}\mathbf{w} = \mathbf{y}$. However due to some reasons, e.g., *there exist more equations than unknowns (the number of rows of $\vec{\Phi}$ is larger than that of columns), this equation may have no solutions, i.e., the system is over-determined.* It is often the origin of the situation $\det(\vec{\Phi}^\top\vec{\Phi}) = 0$ aforementioned. Hence we can not expect to a solution of $\vec{\Phi}\mathbf{w} = \mathbf{y}$ and may instead try to change the problem to solving the

Specifically, the derivation is given by,

$$\begin{aligned}
&\sum_{i=1}^{m}\left[f_{\mathbf{w}}(x^{(i)}) - y^{(i)}\right]^2 \\
&= \left[f_{\mathbf{w}}(x^{(1)}) - y^{(1)}\right]^2 + \cdots + \left[f_{\mathbf{w}}(x^{(m)}) - y^{(m)}\right]^2 \\
&= \begin{pmatrix} \underbrace{w_0\phi_0(x^{(1)}) + \cdots + w_n\phi_n(x^{(1)})}_{f_{\mathbf{w}}(x^{(1)})} - y^{(1)} \\ \vdots \\ \underbrace{w_0\phi_0(x^{(m)}) + \cdots + w_n\phi_n(x^{(m)})}_{f_{\mathbf{w}}(x^{(m)})} - y^{(m)} \end{pmatrix}^\top \\
&\times \begin{pmatrix} w_0\phi_0(x^{(1)}) + \cdots + w_n\phi_n(x^{(1)}) - y^{(1)} \\ \vdots \\ w_0\phi_0(x^{(m)}) + \cdots + w_n\phi_n(x^{(m)}) - y^{(m)} \end{pmatrix}.
\end{aligned}$$

EXERCISE 9: Prove the following identities which are useful when deriving the normal equation

$$\frac{\partial\mathbf{a}^\top\mathbf{b}}{\partial\mathbf{a}} = \mathbf{b}, \quad \frac{\partial\mathbf{a}^\top\mathbf{M}\mathbf{a}}{\partial\mathbf{a}} = \mathbf{a}^\top\left(\mathbf{M} + \mathbf{M}^\top\right),$$

where $\mathbf{a}, \mathbf{b} \in \mathrm{R}^{d\times1} \equiv \mathrm{R}^d, \mathbf{M} \in \mathrm{R}^{d\times d}$.

least-squares problem, $\min_{\mathbf{w}}(\vec{\Phi}\mathbf{w}-\mathbf{y})^\top(\vec{\Phi}\mathbf{w}-\mathbf{y})$. This objective function is just the $J(\mathbf{w})$, and the solution is given by that of the normal equation.

Since $\phi_0 = 1$, without losing generality the design matrix is

$$\begin{pmatrix} \phi_1(x^{(1)}) & \phi_2(x^{(1)}) & \cdots & \phi_n(x^{(1)}) \\ \phi_1(x^{(2)}) & \phi_2(x^{(2)}) & \cdots & \phi_n(x^{(2)}) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_1(x^{(m)}) & \phi_2(x^{(m)}) & \cdots & \phi_n(x^{(m)}) \end{pmatrix}, \tag{36}$$

where each column forms a vector $\vec{\varphi}_j = (\phi_j(x^{(1)}), \cdots, \phi_j(x^{(m)}))^\top \in \mathrm{R}^m$ with $j = 1 \sim n$ (the difference between n and $n+1$ is essentially irrelevant for the following discussion). On the other hand, each row $\vec{\phi}(x^{(i)})$ has dimension n. Under the assumption that the model complexity n is smaller than the data number m, the n-vector $\vec{\phi}(x^{(i)})$ may span a sub-space S with dimension m. Denote $\mathbf{t}$ as an m-vector with its ith component given by $f_{\mathbf{w}}(x^{(i)})$, i.e., $\mathbf{t} = (f_{\mathbf{w}}(x^{(1)}), \cdots, f_{\mathbf{w}}(x^{(m)}))^\top$. Since the vector $\mathbf{t}$ is some linear combination of the basis $\vec{\varphi}_j$, it could be at any point in the n-dimensional space. Under these considerations, *the loss function $J(\mathbf{w}) \sim \sum_{i=1}^m [y^{(i)} - f_{\mathbf{w}}(x^{(i)})]^2 = (y^{(1)} - t_1)^2 + \cdots + (y^{(m)} - t_m)^2$ is the Euclidean distance between $\mathbf{y}$ and $\mathbf{t}$. The least-squares searching for $\mathbf{w}$ is consequently to find a vector $\mathbf{t}$ in the sub-space S in order to make the distance between $\mathbf{t}$ and $\mathbf{y}$ be smallest, i.e., to project the vector $\mathbf{y}$ into the sub-space S.*

We investigate the meaning of the bias parameter w_0. By computing the derivative of $J(\mathbf{w})$ with respect to w_0 with the former given by

$$J(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^m \left[y^{(i)} - w_0 - \sum_{j=1}^n w_j \phi_j(\mathbf{x}^{(i)}) \right]^2, \tag{37}$$

one obtains the optimal value for w_0^* as

$$w_0^* = \langle y \rangle - \sum_{j=1}^n w_j \langle \phi_j \rangle, \;\; \langle y \rangle = \frac{1}{m}\sum_{i=1}^m y^{(i)}, \;\; \langle \phi_j \rangle = \frac{1}{m}\sum_{i=1}^m \phi_j(\mathbf{x}^{(i)}). \tag{38}$$

It shows *the bias parameter w_0 compensates the difference between the output and the weighed-average of the average of basis function on the data samples.*

For the situations the data samples $\mathbf{x}^{(i)}$'s have different weights, e.g., some of which are considered to be much important that others, we could design the weighted loss function,

$$J_{\vec{\Theta}}(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^m \theta^{(i)} \left[f_{\mathbf{w}}(x^{(i)}) - y^{(i)} \right]^2 + \frac{\lambda}{2}\mathbf{w}^\top \mathbf{w},$$

a subscript $\vec{\Theta}$ is added here with $\vec{\Theta} = \mathrm{diag}(\theta^{(1)}, \cdots, \theta^{(m)})$, $\theta^{(i)} > 0$, and $\sum_{i=1}^m \theta^{(i)} = 1$. The corresponding normal equation modifies to be

$$\left(\vec{\Phi}^\top \vec{\Theta} \vec{\Phi} + \lambda \mathbf{1}\right)\mathbf{w} = \vec{\Phi}\vec{\Theta}\mathbf{y}.$$

§7 Stochastic Gradient Descent (Conceptual)

It is useful to notice that *the dimension of the matrix $\vec{\Phi}^\top\vec{\Phi}$ is $n+1$, which may possibly be far smaller than the data number m, making the solving of the normal equation possible. It of course should depend on the algorithms like the gradient descent to search the optimal parameter if the model complexity n is a very large number which hinders the direct inverse of the matrix $\vec{\Phi}^\top\vec{\Phi}$.* In this case, one just uses the information of the gradient of $J(\mathbf{w})$, namely, $\mathbf{w} \leftarrow \mathbf{w} - \epsilon(\vec{\Phi}^\top\vec{\Phi}\mathbf{w} - \vec{\Phi}^\top\mathbf{y})$, to update the learning parameter $\mathbf{w}$. Here ϵ is the learning rate or the step size of the gradient descent search. Since the learning problems in this lecture are based on the polynomial model,there exists the closed form for the optimal parameter, i.e., the one given by the normal equation.

In more general situations in which the basis function takes other forms, there exists no closed form for the learning parameter. In these situations one should necessary use the optimization algorithms to do the search. The search

The second-order derivative of the loss function with respective to $\mathbf{w}$ is given as

$$\frac{\partial^2 J}{\partial \mathbf{w}^2} = \frac{\partial}{\partial \mathbf{w}}\left(\vec{\Phi}^\top\vec{\Phi}\mathbf{w}\right) = \vec{\Phi}^\top\vec{\Phi},$$

which is often positive-definite. If not, by adding the term $\lambda\mathbf{1}$, one could make it to be positive-definite.

is composed of the following steps: 1. Initialize the learning parameter $\mathbf{w}$. 2. Randomly select a data sample $(x^{(i)}, y^{(i)})$. 3. Update the learning parameter according to $\mathbf{w} \leftarrow \mathbf{w} - \epsilon_i \nabla J^{(i)}(\mathbf{w})$, where the $\nabla J^{(i)}(\mathbf{w})$ is the gradient associated with the selected data sample, i.e., $\nabla J^{(i)}(\mathbf{w}) = -\vec{\phi}(x^{(i)})(y^{(i)} - f_{\mathbf{w}}(x^{(i)}))$ where the n-vector $\vec{\phi}$ is constructed from the row of the design matrix. 4. Recursively do the search to fulfill the termination condition. Here one selects the data sample in a stochastic manner in order to reduce the calculation task in the gradient of the loss function at each step. We call this gradient descent the *stochastic gradient descent (SGD),* which plays a central role in modern large-scale optimization problems.

EXERCISE 10: Use gradient descent to solve the nonlinear curve fitting problem.

Problems For This lecture

THEORETICAL

(1) Prove the relation $\partial \mathbf{a}^\top \mathbf{X}^\top \mathbf{b}/\partial \mathbf{X} = \mathbf{b}\mathbf{a}^\top$, and

$$\frac{\partial \det \mathbf{Y}}{\partial x} = \det \mathbf{Y} \mathrm{tr}\left[\mathbf{Y}^{-1}\frac{\partial \mathbf{Y}}{\partial x}\right], \quad \frac{\partial \det \mathbf{Y}}{\partial \mathbf{Y}} = \det \mathbf{Y}\mathbf{Y}^{-\top}, \quad \frac{\partial \ln \det \mathbf{Y}}{\partial \mathbf{Y}} = \mathbf{Y}^{-\top}, \tag{39}$$

where capital letters are for matrices.

(2) The sigmoid function is often denoted as $\sigma(x) = 1/(1+e^{-x})$. Prove that $\tanh(x) = 2\sigma(2x) - 1$. For two learning models,

$$f_{\mathbf{w}}(x) = w_0 + \sum_{j=1}^{n} w_j \sigma\left(\frac{x-\nu_j}{s}\right), \quad f_{\mathbf{u}}(x) = u_0 + \sum_{j=1}^{n} u_j \tanh\left(\frac{x-\nu_j}{2s}\right), \tag{40}$$

find the relationship between two sets of parameter $\{w_j\}$ and $\{u_j\}$.

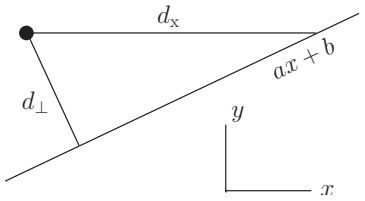

Fig. 12: Distances $d_\perp$ and d_x.

(3) In the linear learning model, define the distance of the point $(x^{(i)}, y^{(i)})$ to the fitting line $y = ax + b$ as,

$$d_\perp^{(i)}(a,b) = \left|ax^{(i)} + b - y^{(i)}\right| \Big/ \sqrt{a^2+1}\,, \quad d_{\mathrm{x}}^{(i)}(a,b) = \left|ax^{(i)} + b - y^{(i)}\right| \Big/ a. \tag{41}$$

Consequently, the loss function is obtained as $J_\perp(a,b) = 2^{-1}\sum_{i=1}^m d_\perp^{(i),2}(a,b)$ or $J_{\mathrm{x}}(a,b) = 2^{-1}\sum_{i=1}^m d_{\mathrm{x}}^{(i),2}(a,b)$. Derive the equations for determining the parameters a and b. See Fig. 12 for the geometrical meaning of $d_\perp$ or d_{x}, and if a is large $d_\perp \approx d_{\mathrm{x}}$.

PROGRAMMING

(4) Adopt the gradient descent algorithm to find the optimal values for the learning parameters of the loss function (41), and compare the results with Fig. 2. Are the predictions for a and b biased or not?

(5) Consider the following learning model,

$$J(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{m}\left[\sum_{j=1}^{m} w_j G(x^{(i)}, x^{(j)}) - y^{(i)}\right]^2 + \frac{\lambda}{2}\mathbf{w}^\top\mathbf{w}, \quad G(x, x^{(j)}) = e^{-(x-x^{(j)})^2/2\nu^2}, \tag{42}$$

where there is no w_0-term here and G is the Gauss kernel. Investigate via SGD the data produced by $y^{(i)} = \sin(2\pi x^{(i)}) + a^{(i)}$ based on (42) with large training number m, e.g., $m = 10^6$. The width parameter $\nu = 0.4$ and the learning rate $\epsilon = 0.01$ are fixed.