Lecture 5

First Lesson on "Learning from Data", Parameter Estimate

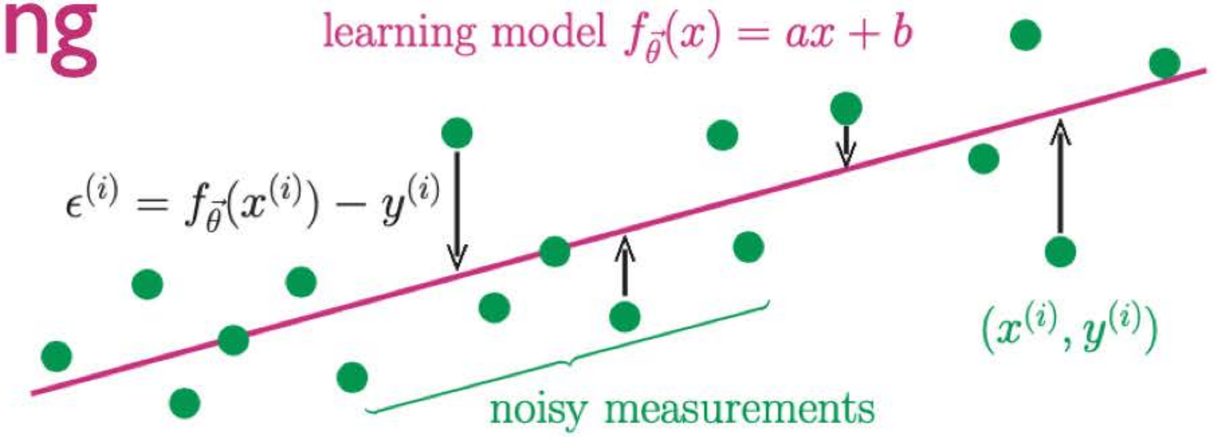
Bao-Jun Cai, 4/1/2026

Introduction to Algorithms for Data Science and Physics IMP@Fudan, 2026

Topics of this lecture:

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

Problem of linear curve fitting



aim: minimizing the loss function

$$J(\vec{\theta}) = J(a,b) = \frac{1}{2}\sum_{i=1}^{m}\left(f_{\vec{\theta}}(x^{(i)}) - y^{(i)}\right)^2$$

$$f_{\vec{\theta}}(x) = ax + b$$

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

data samples: $(x^{(i)}, y^{(i)})$

fitting model: $f_{\vec{\theta}}(x) = ax + b$

model parameters: $\vec{\theta} = (a,b)$

sample averages

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

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

$m = 2 : (1,2), (2,5)$

$$\langle x\rangle = \frac{1}{2}\left(x^{(1)} + x^{(2)}\right) = \frac{3}{2}, \quad \langle y\rangle = \frac{1}{2}\left(y^{(1)} + y^{(2)}\right) = \frac{7}{2}$$

$$\langle x^2\rangle = \frac{1}{2}\left(x^{(1),2} + x^{(2),2}\right) = \frac{5}{2}, \quad \langle y^2\rangle = \frac{1}{2}\left(y^{(1),2} + y^{(2),2}\right) = \frac{29}{2}$$

$$\langle xy\rangle = \frac{1}{2}\left(x^{(1)}y^{(1)} + x^{(2)}y^{(2)}\right) = 6$$

Closed solutions for a and b

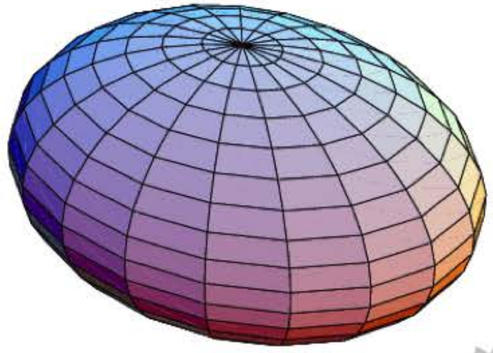
$\sim Xa^2 + Yb^2 + Zab + Ua + Vb + W$

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

$$H = \frac{\partial^2 J(\vec{\theta})}{\partial\theta_i \partial\theta_j} = m\begin{pmatrix}\langle x^2\rangle & \langle x\rangle \\ \langle x\rangle & 1\end{pmatrix}$$

easy to check $\det H \geq 0$?

$(\mathrm{var}[x] = \langle x^2\rangle - \langle x\rangle^2 \geq 0)$



$$\frac{\partial J}{\partial a} = 0, \frac{\partial J}{\partial b} = 0$$

optimal values for a and b

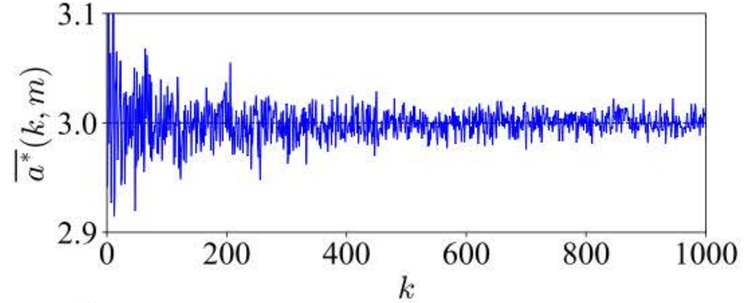
$$a^* = \frac{\langle xy\rangle - \langle x\rangle\langle y\rangle}{\langle x^2\rangle - \langle x\rangle^2}, \quad b^* = \frac{\langle x^2\rangle\langle y\rangle - \langle x\rangle\langle xy\rangle}{\langle x^2\rangle - \langle x\rangle^2}$$

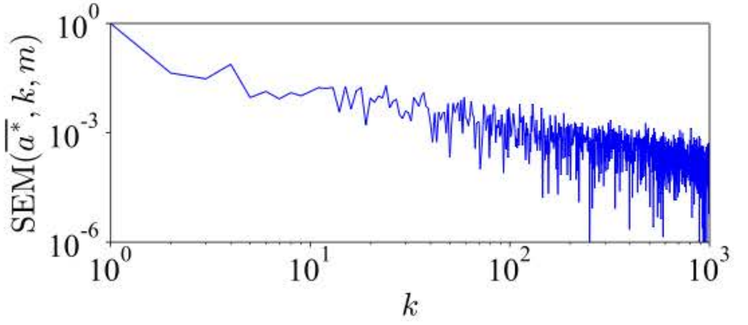
data preparation:

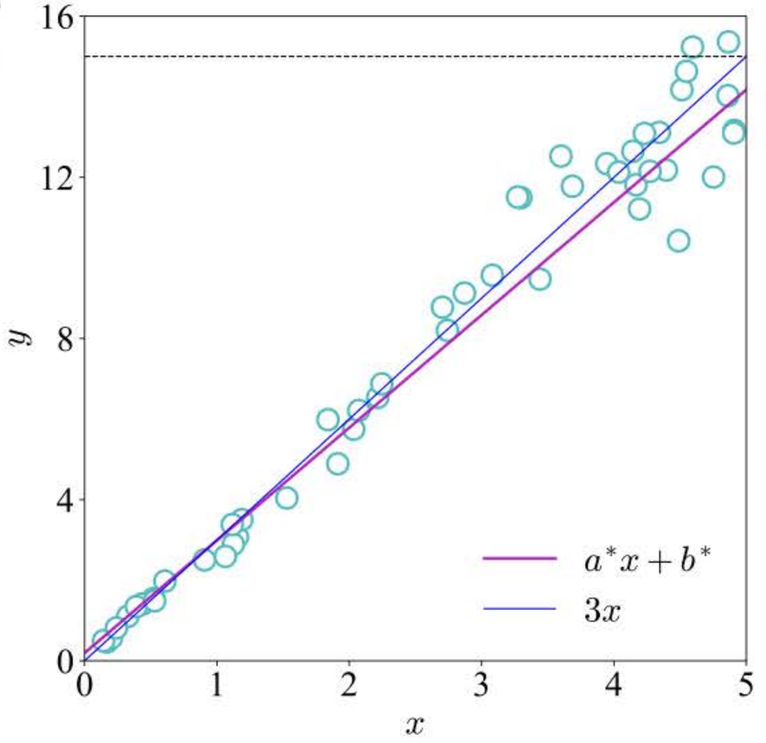
$y^{(i)} = (3 + \Delta)x^{(i)}$

$x^{(i)} \sim \mathrm{Unif}[0, 5]$

$\Delta \sim \mathcal{N}(0, 0.3)$

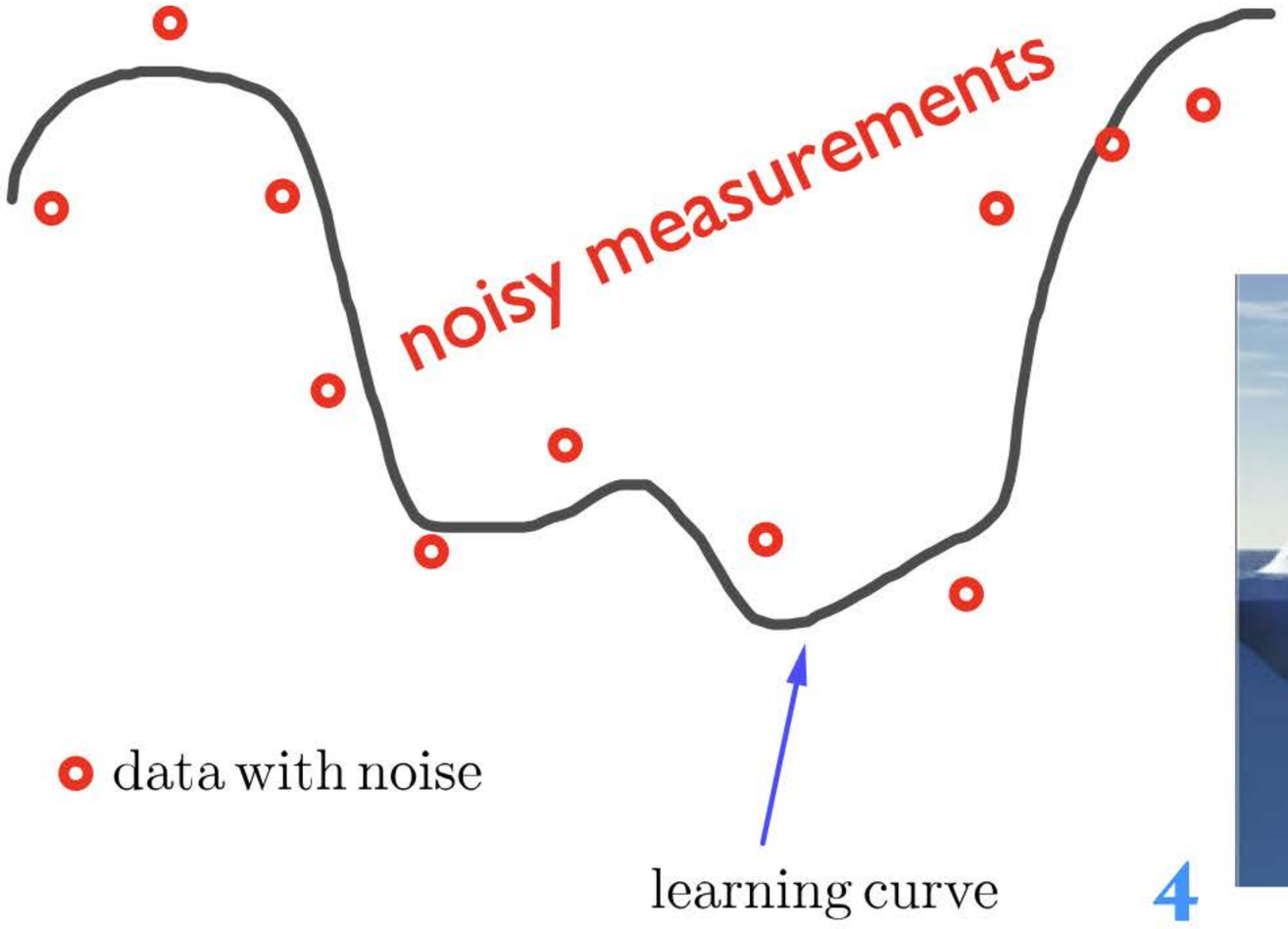






Nonlinear curve fitting (concepts)

approximation: $f_{\mathbf{w}}(\mathbf{x}) \rightarrow f_{\text{phys}}(\mathbf{x})$



(1) linear fitting is ideal/approximated

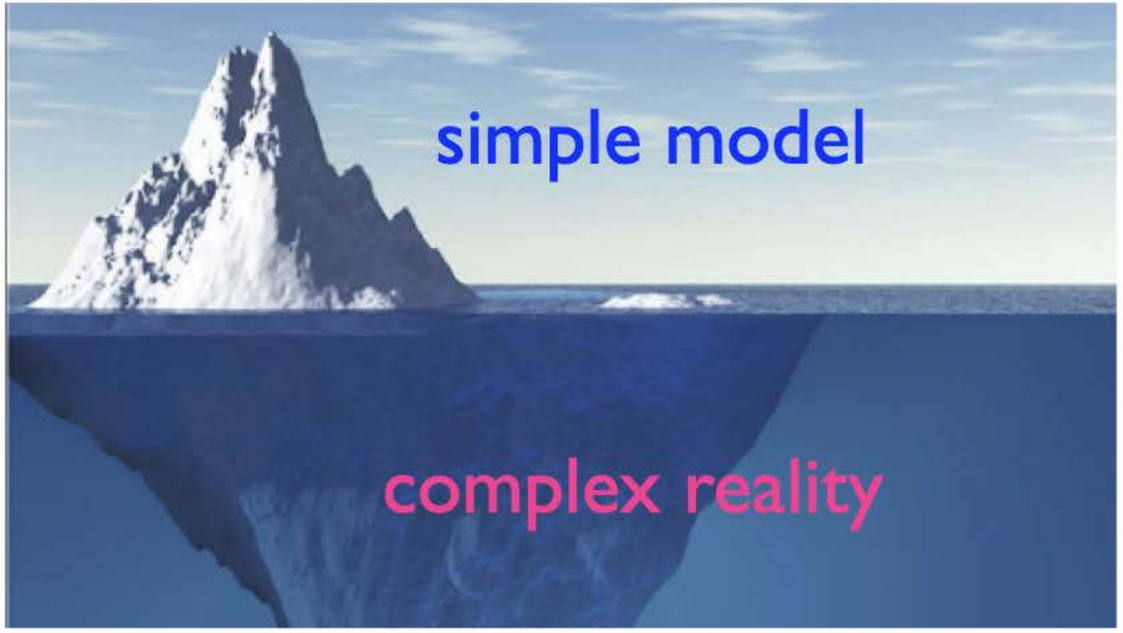
(2) a physical (realistic) model $f_{\text{phys}}(\mathbf{x})$

(3) generally $y^{(i)} \neq f_{\text{phys}}(\mathbf{x}^{(i)}) \leftrightarrow$ noise $\epsilon^{(i)}$

(4) a learning model $f_{\mathbf{w}}(\mathbf{x})$

(5) learning parameter vector $\mathbf{w} \leftrightarrow \vec{\theta}$

(6) some types of loss function $J(\mathbf{w})$



Least-squares with polynomials

what will happen if $J(\mathbf{w}) = \sum_{i=1}^{m} |f_{\mathbf{w}}(\mathbf{x}^{(i)}) - y^{(i)}|$?

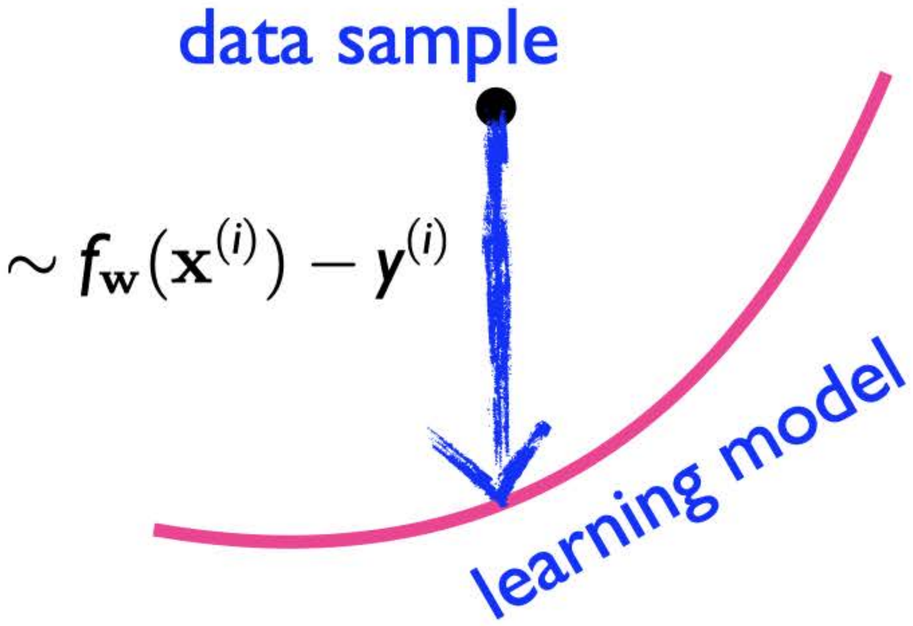
ℓ_2 loss

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

least-squares

polynomial learning model

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



optimal solution:

$$\frac{\partial J}{\partial \mathbf{w}} = \vec{\mathbf{0}} \leftrightarrow \frac{\partial J}{\partial w_j} = 0$$

Details on derivation

what will happen if $m < n$?

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

$$= \sum_{i=1}^{m} \sum_{j'=0}^{n} w_{j'} x^{(i),j'+j} - \sum_{i=1}^{m} y^{(i)} x^{(i),j} = m \left(\sum_{j'=0}^{n} w_{j'} \langle x^{j'+j} \rangle - \langle x^j y \rangle \right)$$

linear equations for w_j:

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

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

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

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

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

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

$$\mathbf{Fw} = \mathbf{G}$$

$F_{ij} = F_{ji}$

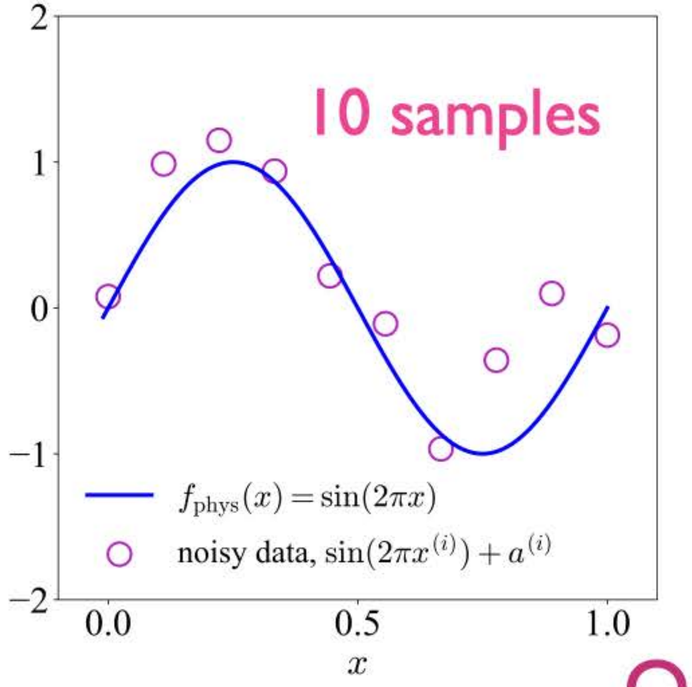
Matrix computing is fundamental in machine learning problems!

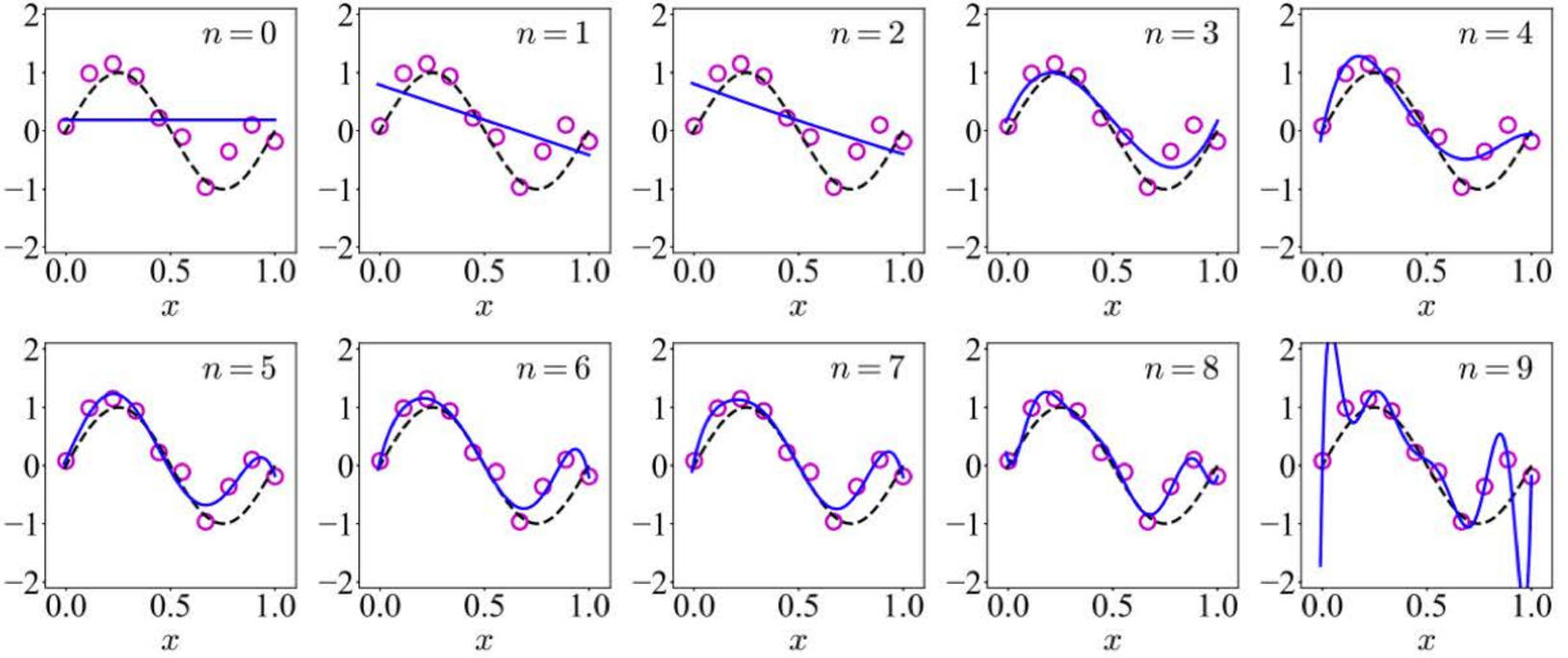
Simulated results for polynomial learning

$$f_w(x) = \sum_{i=1}^{n} w_j x^j$$

$$y^{(i)} = \underbrace{\sin\left(2\pi x^{(i)}\right)}_{\text{physical model}} + \overbrace{a^{(i)}}^{\text{noise}}$$

$$x^{(i)} = i/9, a^{(i)} \sim \text{Unif}[-0.8, 0.8]$$





(1) small *n*: smooth/inaccurate ↔ large bias/small variance

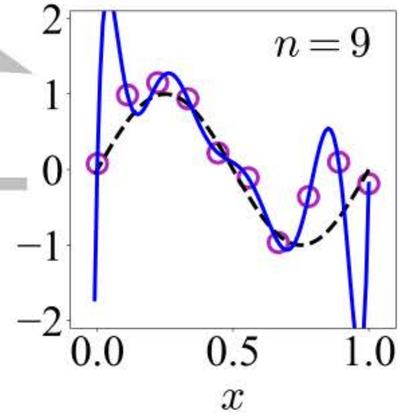
(2) large *n*: accurate/irregular ↔ small bias/large variance

One can not have simultaneous small bias and variance!

What happens? more analyses

over-fitting

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



fine-tuning

$0.0001 = 10^{10} + 0.0001 - 10^{10}$

$n = 0$: $w_0^* = \dfrac{y^{(1)} + \cdots + y^{(10)}}{10} \leftrightarrow$ mean

$n = 9$: the model is too complicated

the model is too simple

n: model complexity

Select a reasonable n is very relevant/important!

Bias-variance decomposition

$$\Delta = \mathrm{E}\left[\left(f_{\mathrm{phys}}(\bar{x}) + a - \hat{f}(\bar{x})\right)^2\right]$$

$$= \mathrm{E}\left[f_{\mathrm{phys}}^2(\bar{x}) + a^2 + \hat{f}^2(\bar{x}) + 2a f_{\mathrm{phys}}(\bar{x}) - 2a\hat{f}(\bar{x}) - 2f_{\mathrm{phys}}(\bar{x})\hat{f}(\bar{x})\right]$$

$$= \mathrm{E}\left[f_{\mathrm{phys}}^2(\bar{x}) + a^2 + \hat{f}^2(\bar{x}) - 2f_{\mathrm{phys}}(\bar{x})\hat{f}(\bar{x})\right]$$

$$= \mathrm{E}\left[a^2\right] + \mathrm{E}\left[f_{\mathrm{phys}}^2(\bar{x})\right] + \mathrm{E}\left[\hat{f}^2(\bar{x})\right] - 2\mathrm{E}\left[f_{\mathrm{phys}}(\bar{x})\hat{f}(\bar{x})\right]$$

$$= \mathrm{E}\left[a^2\right] - \mathrm{E}^2[a] + \mathrm{E}\left[f_{\mathrm{phys}}^2(\bar{x})\right] + \mathrm{E}\left[\hat{f}^2(\bar{x})\right] - 2\mathrm{E}\left[f_{\mathrm{phys}}(\bar{x})\hat{f}(\bar{x})\right] + \mathrm{E}^2\left[\hat{f}(\bar{x})\right] - \mathrm{E}^2\left[\hat{f}(\bar{x})\right]$$

$$= \mathrm{var}[a] + f_{\mathrm{phys}}^2(\bar{x}) + \mathrm{E}\left[\hat{f}^2(\bar{x})\right] - 2f_{\mathrm{phys}}(\bar{x})\mathrm{E}\left[\hat{f}(\bar{x})\right] + \mathrm{E}^2\left[\hat{f}(\bar{x})\right] - \mathrm{E}^2[\hat{f}(\bar{x})]$$

$$= \mathrm{var}[a] + \mathrm{E}^2\left[\hat{f}(\bar{x})\right] - 2f_{\mathrm{phys}}(\bar{x})\mathrm{E}\left[\hat{f}(\bar{x})\right] + f_{\mathrm{phys}}^2(\bar{x}) + \mathrm{E}\left[\hat{f}^2(\bar{x})\right] - \mathrm{E}^2\left[\hat{f}(\bar{x})\right]$$

$$= \mathrm{var}[a] + \left[\mathrm{E}\left[\hat{f}(\bar{x})\right] - f_{\mathrm{phys}}(\bar{x})\right]^2 + \mathrm{E}\left[\left[\hat{f}(\bar{x}) - \mathrm{E}\left[\hat{f}(\bar{x})\right]\right]^2\right]$$

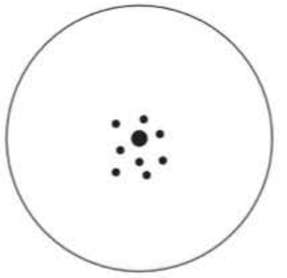
a: noise $\leftrightarrow \mathrm{E}[a] = 0$

$\hat{f}(x)$: learning model

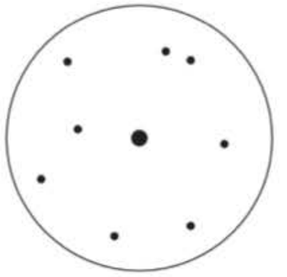
$\bar{x}$: testing data

$\hat{f}(\bar{x})$: model prediction

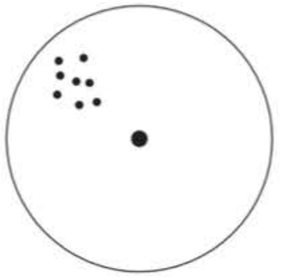
total error: $\mathrm{E}\left[\left(f_{\mathrm{phys}}(\bar{x}) + a - \hat{f}(\bar{x})\right)^2\right] = \mathrm{var}[a] + \left(\text{bias of } \hat{f}(\bar{x})\right)^2 + \text{variance of } \hat{f}(\bar{x})$



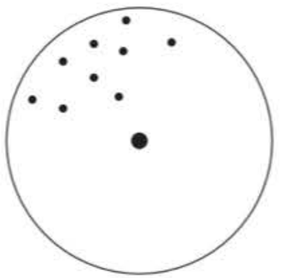
low bias, low variance



low bias, high variance

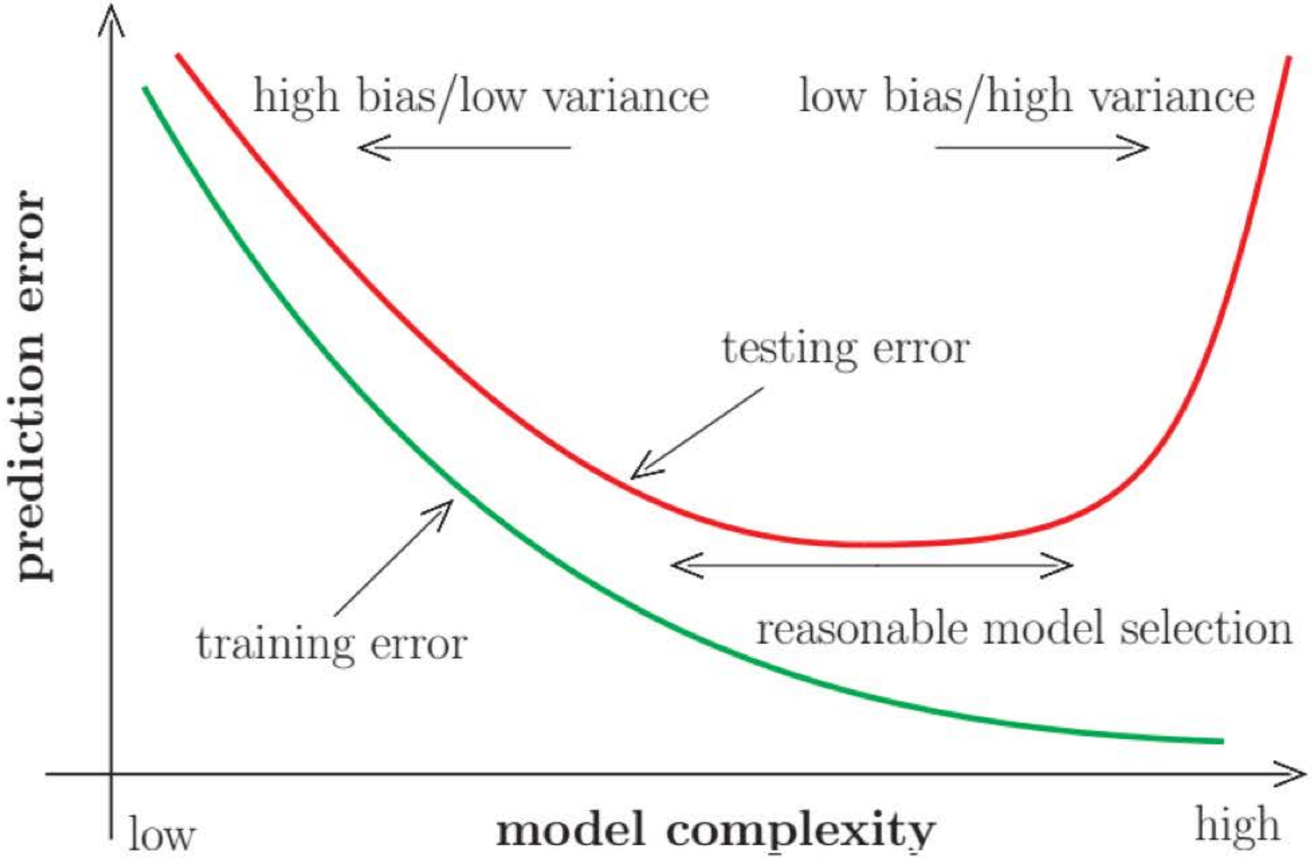


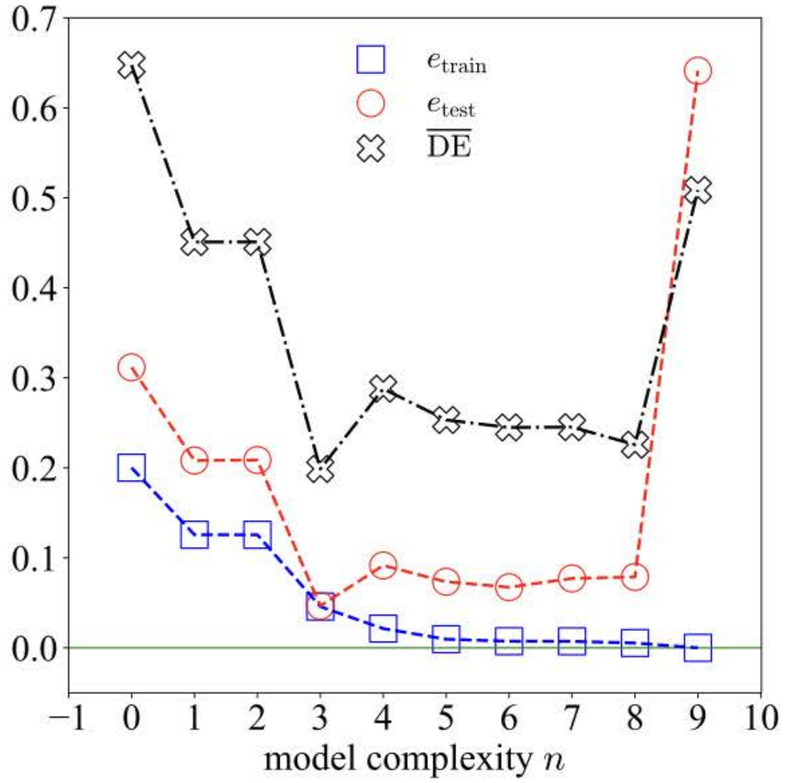
high bias, low variance



high bias, high variance

Related: training and testing losses





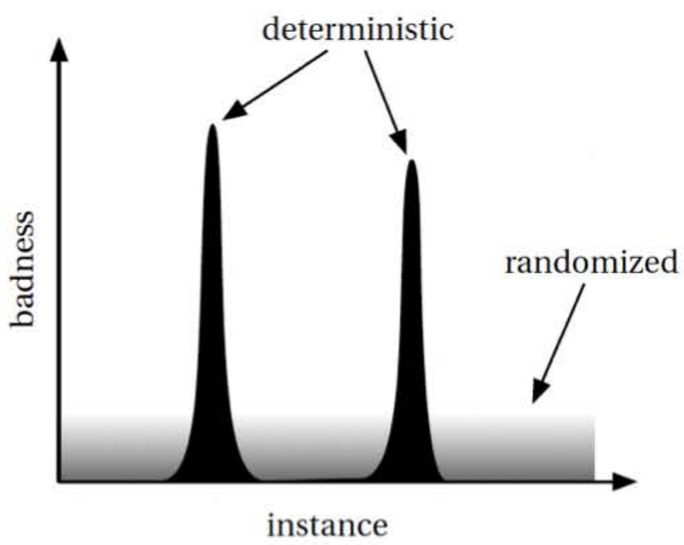
$$\mathrm{E}\left[\left(f_{\text{phys}}(\bar{x}) + a - \widehat{f}(\bar{x})\right)^2\right] = \mathrm{var}[a] + \left[\text{bias of } \widehat{f}(\bar{x})\right]^2 + \text{variance of } \widehat{f}(\bar{x})$$

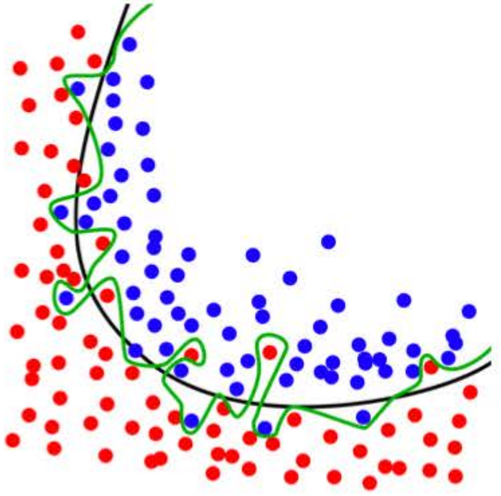
$$e_{\text{train}} \equiv \frac{1}{m}\left[\frac{1}{2}\sum_{i=1}^{m}\left[f_{\mathbf{w}^*}(\mathbf{x}^{(i)}) - y^{(i)}\right]^2\right]$$

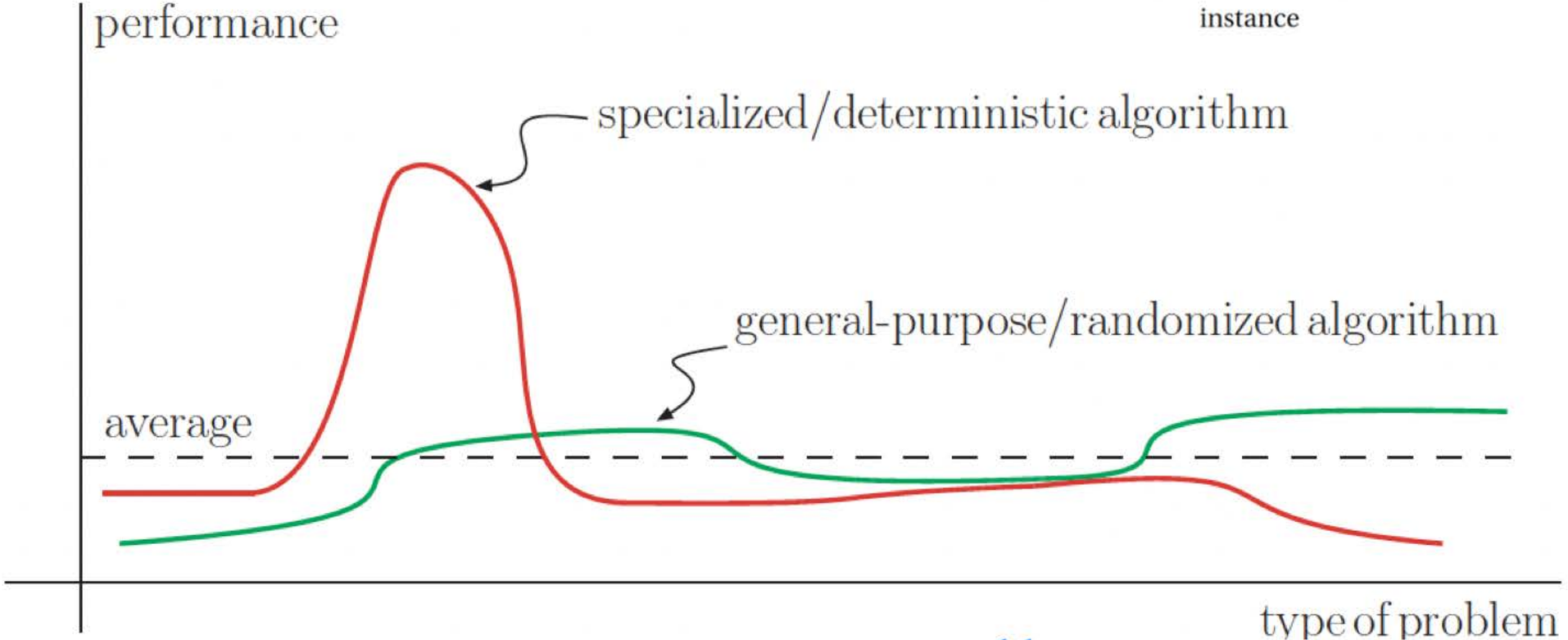
$$e_{\text{test}} \equiv \frac{1}{m'}\left[\frac{1}{2}\sum_{i=1}^{m'}\left[f_{\mathbf{w}^*}(\mathbf{x}^{(i')}) - y^{(i')}\right]^2\right]$$

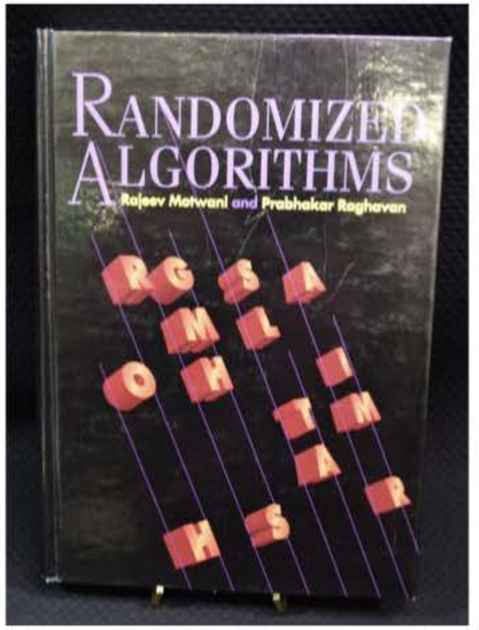
"No-free-lunch" theorem

design paradigm: randomized algorithms





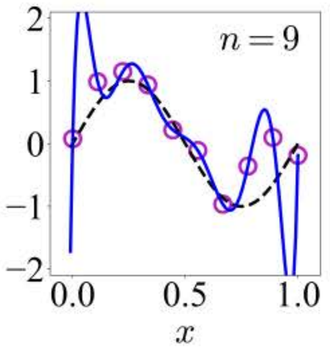


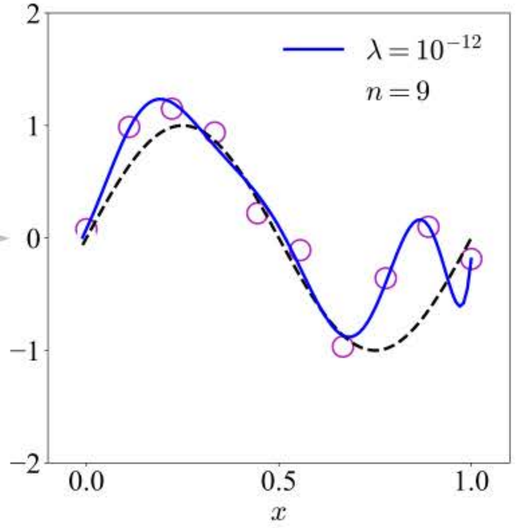


Regularization

$$J(\mathbf{w}) = \underbrace{\frac{1}{2}\sum_{i=1}^{m}\left[f_{\mathbf{w}}(x^{(i)}) - y^{(i)}\right]^2}_{\text{data+learning model}} + \overbrace{\lambda g(\mathbf{w})}^{\text{belief in data}}$$

over-fitting



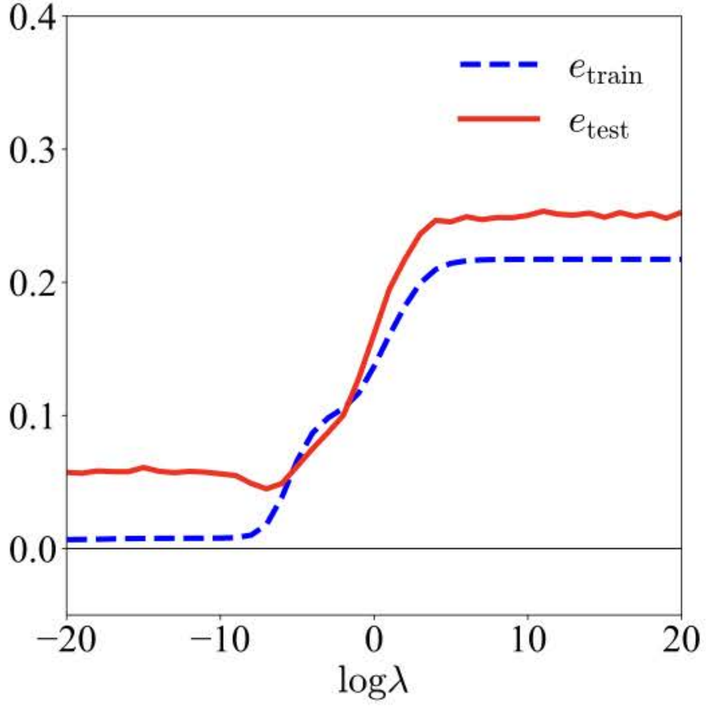


$$\ell_2 : g(\mathbf{w}) = w_0^2 + w_1^2 + \cdots + w_n^2$$

$$\begin{pmatrix} \langle 1\rangle & \langle x\rangle & \cdots & \langle x^n\rangle \\ \langle x\rangle & \langle x^2\rangle & \cdots & \langle x^{n+1}\rangle \\ \vdots & \vdots & \ddots & \vdots \\ \langle x^n\rangle & \langle x^{n+1}\rangle & \cdots & \langle x^{2n}\rangle \end{pmatrix} \begin{pmatrix} w_0 \\ w_1 \\ \vdots \\ w_n \end{pmatrix} = \begin{pmatrix} \langle y\rangle \\ \langle xy\rangle \\ \vdots \\ \langle x^n y\rangle \end{pmatrix} - \lambda \begin{pmatrix} w_0 \\ w_1 \\ \vdots \\ w_n \end{pmatrix} \leftrightarrow \overbrace{\boxed{(\mathbf{F} + \lambda \mathbf{1})\mathbf{w} = \mathbf{G}}}^{\text{Levenberg–Marquardt}}$$

$\lambda = 0 \leftrightarrow$ Newtonian; $\lambda \to \infty \leftrightarrow$ gradient descent

$$\lambda g(\mathbf{w}) \gg \frac{1}{2}\sum_{i=1}^{m}\left[f_{\mathbf{w}}(x^{(i)}) - y^{(i)}\right]^2 \to \mathbf{w}^* = \vec{0} \text{ (no data effects)}$$



Normal equation and basis functions

$$f_{\mathbf{w}}(x) = \mathbf{w}^\top \vec{\phi}(x) = \vec{\phi}^\top(x)\mathbf{w} \to J(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{m}\left[f_{\mathbf{w}}(x^{(i)}) - y^{(i)}\right]^2 = \boxed{\frac{1}{2}(\vec{\Phi}\mathbf{w} - \mathbf{y})^\top(\vec{\Phi}\mathbf{w} - \mathbf{y})}$$

basis function $\vec{\phi}(x) = \left(1, x, x^2, \cdots, x^n\right)^\top \to \phi_j(x) = x^j, \phi_j(x^{(i)}) = \phi_{ji}$

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

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

$$\vec{\Phi}^\top\vec{\Phi} \in \mathbf{R}^{(n+1)\times(n+1)}$$

$$\mathbf{w} \leftarrow \mathbf{w} - \epsilon\vec{\Phi}^\top \overbrace{(\vec{\Phi}\mathbf{w} - \mathbf{y})}^{\text{residual error}}$$

$$\vec{\mathbf{0}} = \partial J(\mathbf{w})/\partial\mathbf{w} = \vec{\Phi}^\top\vec{\Phi}\mathbf{w} - \vec{\Phi}^\top\mathbf{y} \leftrightarrow \boxed{\vec{\Phi}^\top\vec{\Phi}\mathbf{w} = \vec{\Phi}^\top\mathbf{y}} \to \mathbf{w}^* = (\vec{\Phi}^\top\vec{\Phi})^{-1}\vec{\Phi}^\top\mathbf{y}$$

$\dim(\vec{\Phi}^\top\vec{\Phi}) = n + 1 \ll m$

example: linear learning model

Levenberg–Marquardt: $\mathbf{w}^* = (\vec{\Phi}^\top\vec{\Phi} + \lambda\mathbf{1})^{-1}\vec{\Phi}^\top\mathbf{y}$