FUNDAMENTOS DE MACHINE LEARNING: REGRESSÃO

Fevereiro de 2021

Ref.: Cap. 4 do livro texto

Machine Learning: uma visão panorâmica de métodos, algoritmos e modelos

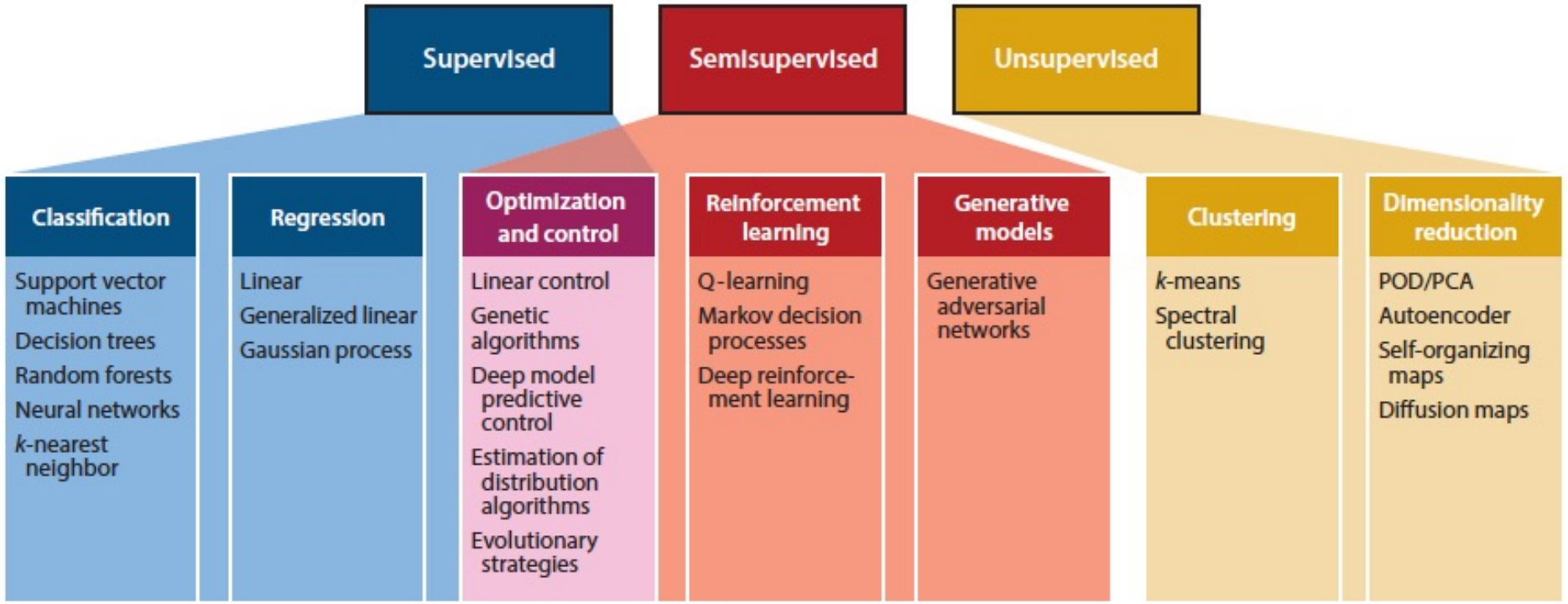


Figure 1

Machine learning algorithms may be categorized into supervised, unsupervised, and semisupervised, depending on the extent and type of information available for the learning process. Abbreviations: PCA, principal component analysis; POD, proper orthogonal decomposition.

Annual Review of Fluid Mechanics Machine Learning for Fluid Mechanics
Steven L. Brunton, Bernd R. Noack, and Petros Koumoutsakos

Regressão : aprendizado supervisionado e otimização

- Temos n valores (eventualmente x é um vetor d-dimensional) de entrada (variáveis de controle)

$$\mathbf{x}_{1:n} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$$

- e n valores de saída (eventualmente y é um vetor d'-dimensional)

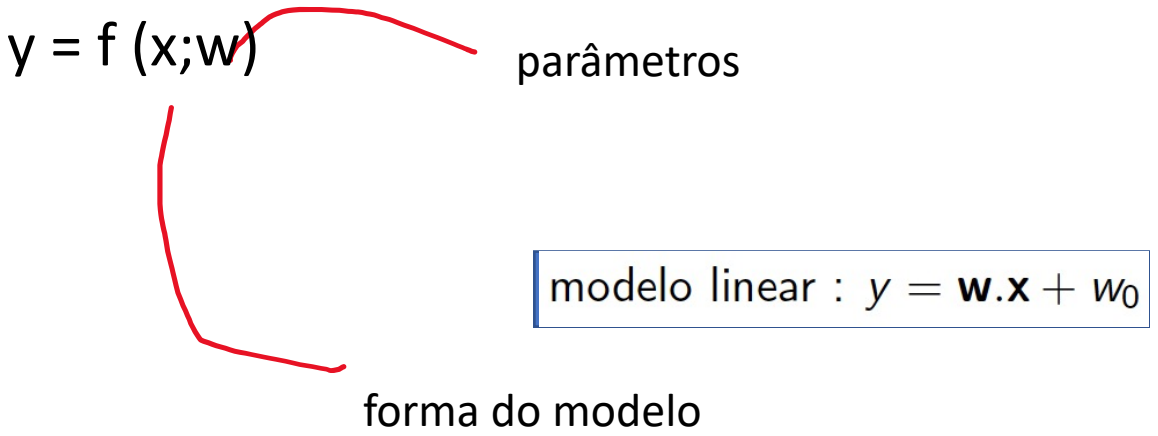
$$y_{1:n} = (y_1, \ldots, y_n)$$

- Obs: no contexto de regressão, y é uma variável assumindo valores contínuos (em caso contrário temos um problema de classificação)

- Como construir (sistematicamente) um modelo a partir desses dados (lembre-se de suas primeiras experiências em um laboratório... mínimos quadrados te diz algo?

Regressão como um problema de otimização

- O modelo : forma (polinômios, exponenciais, redes neurais, processos gaussianos, ...) e parâmetros: como escolher ou obter?



- Obs: no contexto de regressão, y é uma variável assumindo valores contínuos (em caso contrário temos um problema de classificação)

Regressão como um problema de otimização

Encontrar $\mathbf{w}_*$ tal que

$$\min_{\mathbf{w}} \underbrace{\mathcal{L}\left(f(\mathbf{x}_D, \mathbf{w}), y_{\mathcal{D}}\right)}_{\textit{loss function} - \textit{dados}} + \underbrace{R(\mathbf{x})}_{\textit{reg.}}$$

Função de perda : “distância”
entre dados e saídas dos modelos

$$E_\infty(f) = \max_{1<k<n} |f(x_k) - y_k| \qquad \text{Maximum Error } (\ell_\infty)$$

$$E_1(f) = \frac{1}{n}\sum_{k=1}^{n} |f(x_k) - y_k| \qquad \text{Mean Absolute Error } (\ell_1)$$

$$E_2(f) = \left(\frac{1}{n}\sum_{k=1}^{n} |f(x_k) - y_k|^2\right)^{1/2} \qquad \text{Least-squares Error } (\ell_2).$$

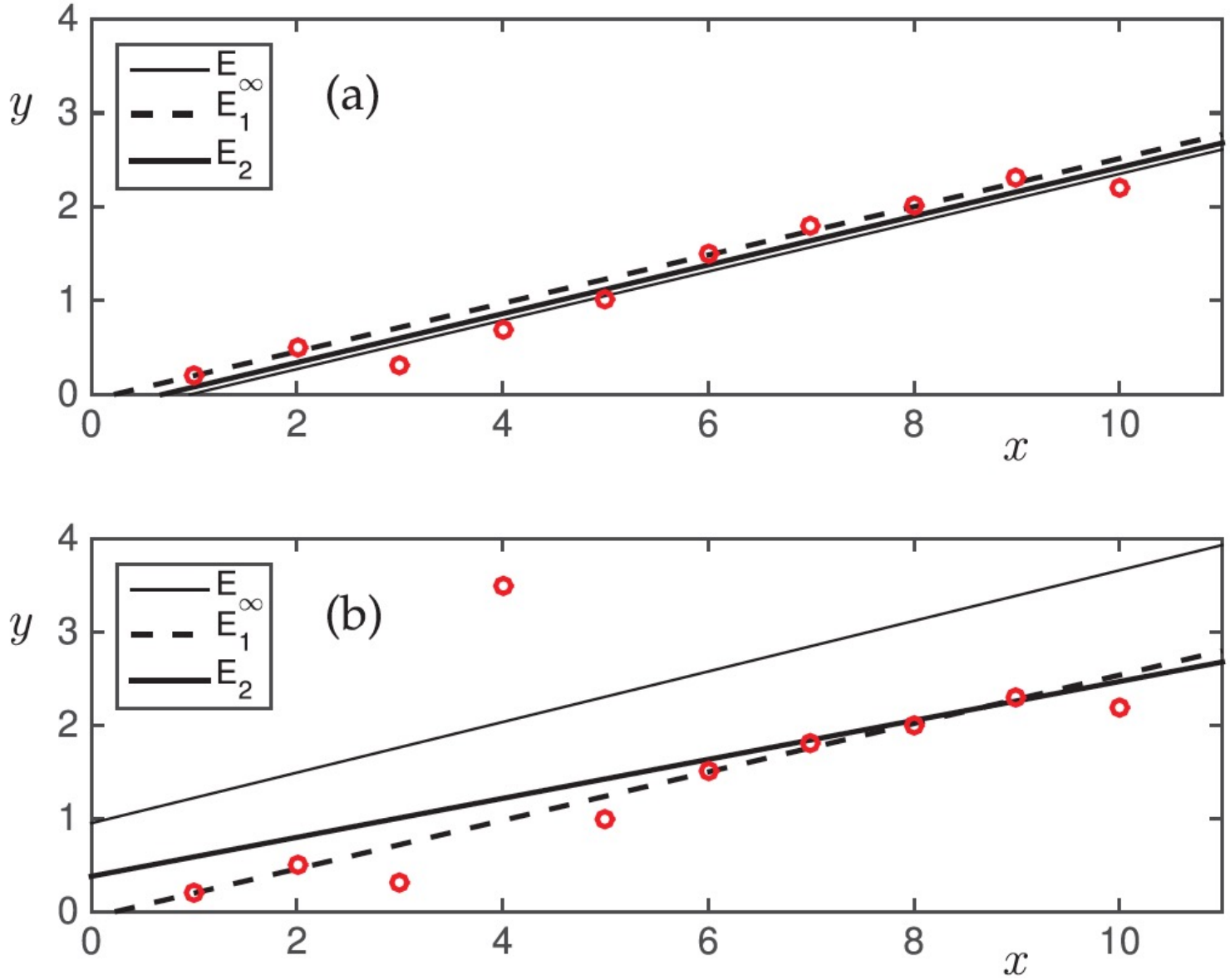


Figure 4.2 Line fits for the three different error metrics E_∞, E_1 and E_2. In (a), the data has not outliers and the three linear models, although different, produce approximately the same model. With outliers, (b) shows that the predictions are significantly different.

Code 4.1 Regression for linear fit.

```
% The data
x=[1 2 3 4 5 6 7 8 9 10]
y=[0.2 0.5 0.3 3.5 1.0 1.5 1.8 2.0 2.3 2.2]

p1=fminsearch('fit1',[1 1],[],x,y);
```

```
p2=fminsearch('fit2',[1 1],[],x,y);
p3=fminsearch('fit3',[1 1],[],x,y);

xf=0:0.1:11
y1=polyval(p1,xf); y2=polyval(p2,xf); y3=polyval(p3,xf);

subplot(2,1,2)
plot(xf,y1,'k'), hold on
plot(xf,y2,'k--','Linewidth',[2])
plot(xf,y3,'k','Linewidth',[2])
plot(x,y,'ro','Linewidth',[2]), hold on
```

Code 4.2 Maximum error ℓ_∞.

```
function E=fit1(x0,x,y)
E=max(abs( x0(1)*x+x0(2)-y ));
```

Code 4.3 Sum of absolute error ℓ_1.

```
function E=fit2(x0,x,y)
E=sum(abs( x0(1)*x+x0(2)-y ));
```

Code 4.4 Least-squares error ℓ_2.

```
function E=fit3(x0,x,y)
E=sum(abs( x0(1)*x+x0(2)-y ).^2 );
```

Complexidade do modelo...

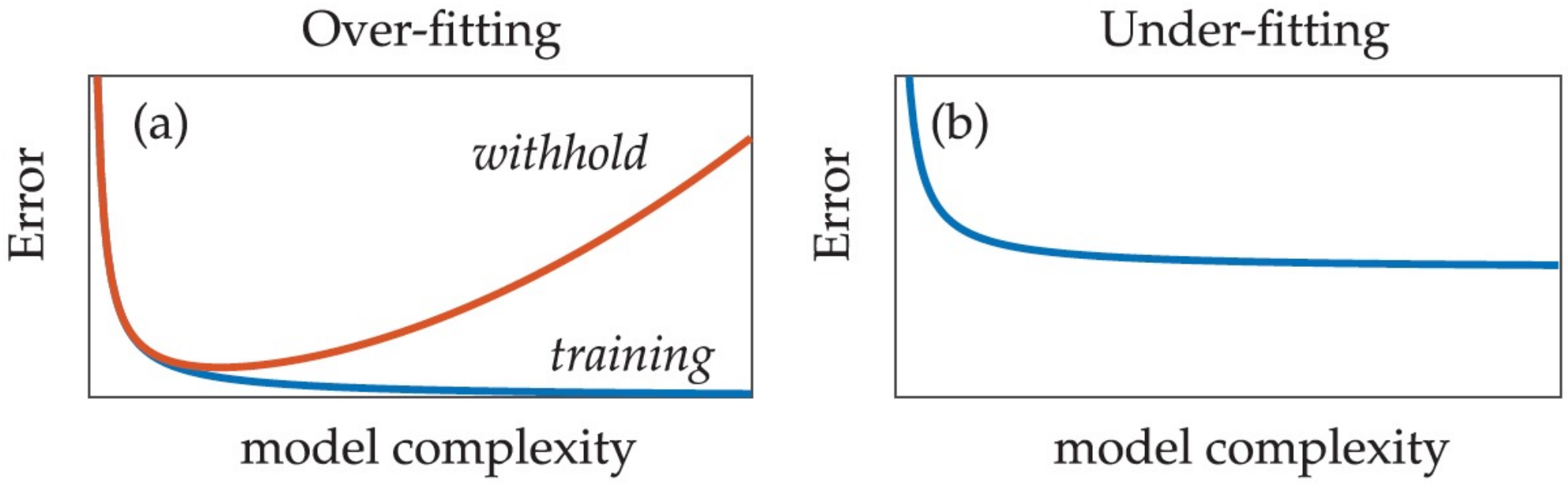


Figure 4.1 Prototypical behavior of over- and under-fitting of data. (a) For over-fitting, increasing the model complexity or training epochs (iterations) leads to improved reduction of error on training data while increasing the error on the withheld data. (b) For under-fitting, the error performance is limited due to restrictions on model complexity. These canonical graphs are ubiquitous in data science and of paramount importance when evaluating a model.

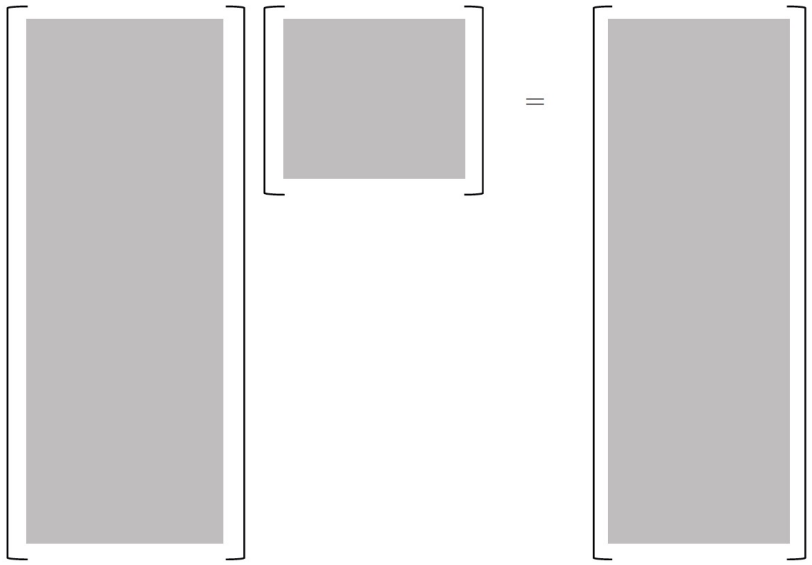
Mínimos Quadrados (modelos polinomiais)

$$A\,\mathbf{w} = \mathbf{b}$$

n ... Dados

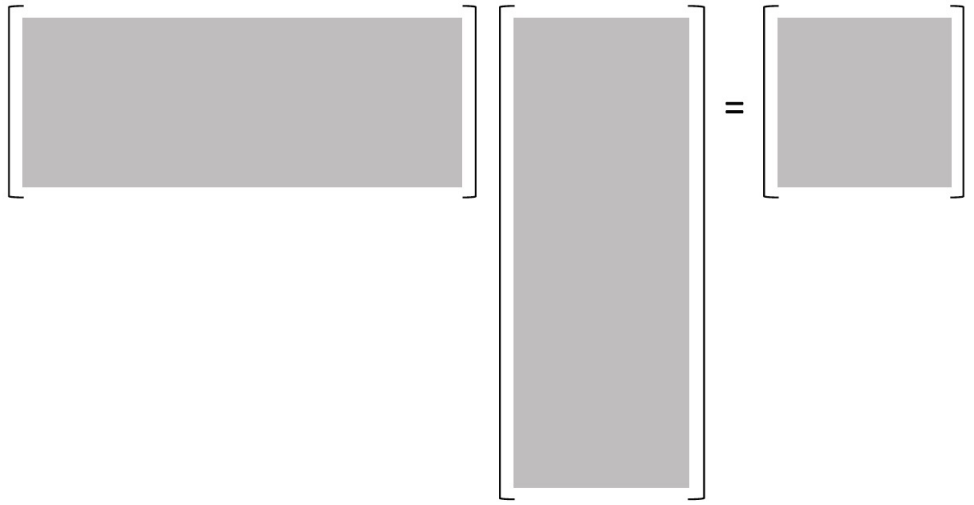
p .... Complexidade do modelo (ordem do polinômio é p-1)

$$[A]_{nxp}$$



$$\hat{\mathbf{x}} = \underset{\mathbf{x}}{\operatorname{argmin}} \|\mathbf{Ax} - \mathbf{b}\|_2 + \lambda_1 \|\mathbf{x}\|_1 + \lambda_2 \|\mathbf{x}\|_2$$

$$\min_{\mathbf{A}} (\lambda_1 \|\mathbf{x}\|_1 + \lambda_2 \|\mathbf{x}\|_2) \quad \text{subject to} \quad \mathbf{Ax} = \mathbf{b}$$



Seleção de modelos, overfitting e validação

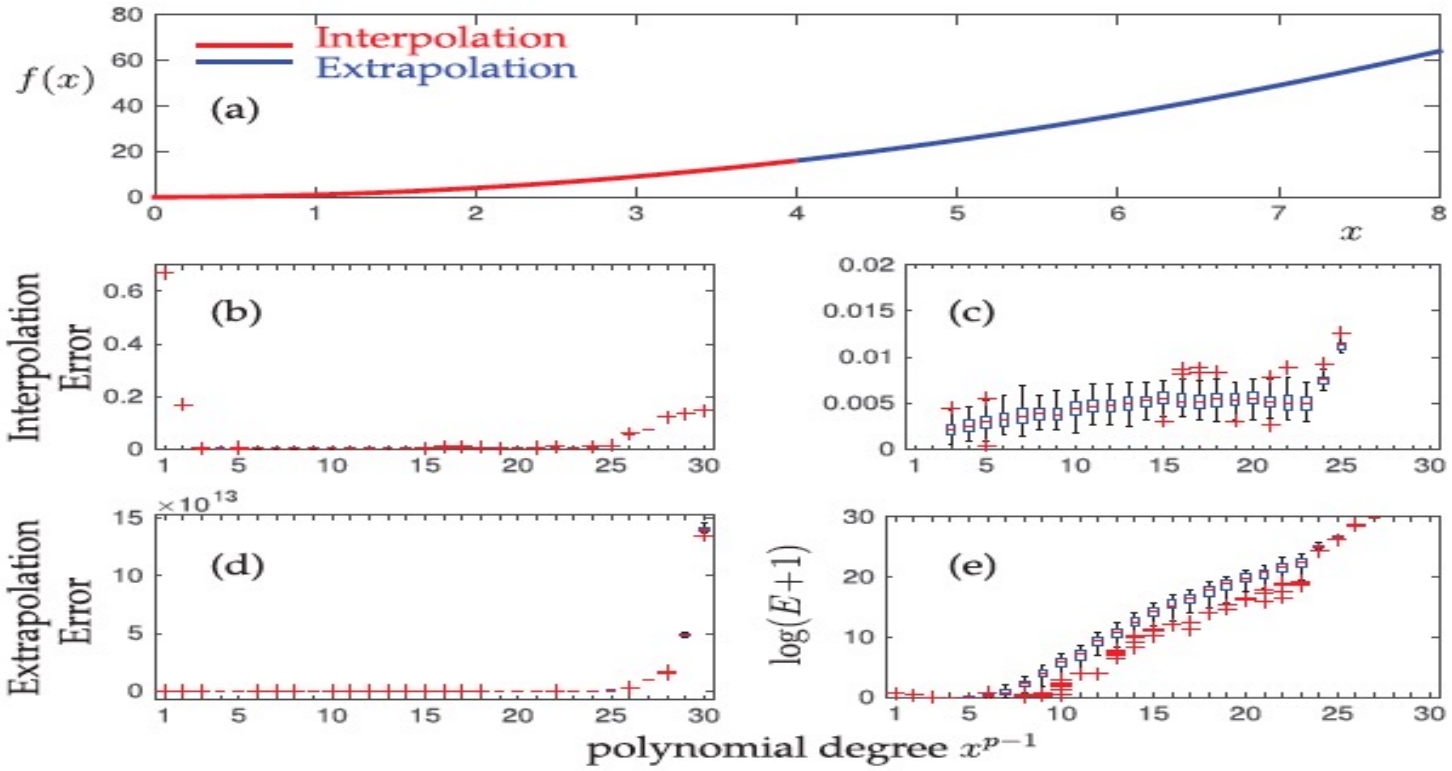


Figure 4.17 (a) The ideal model $f(x) = x^2$ over the domain $x \in [0, 8]$. Data is collected in the region $x \in [0, 4]$ in order to build a polynomial regression model (4.44) with increasing polynomial degree. In the interpolation regime $x \in [0, 4]$, the model error stays constrained, with increasing error due to overfitting for polynomials of degree greater than 2. The error is shown in panel (b) with a zoom in of the error in panel (c). For extrapolation, $x \in [4, 8]$, the error grows exponentially beyond a parabolic fit. In panel (d), the error is shown to grow to 10^{13}. A zoom in of the region on a logarithmic scale of the error ($\log(E+1)$ where unity is added so that zero error produces a zero score) shows the exponential growth of error. This clearly shows that the model trained on the interval $x \in [0, 4]$ does not generalize (extrapolate) to the region $x \in [4, 8]$. This example should serve as a serious warning and note of caution in model fitting.

Regressão Linear : uma visão probabilística (Bayesian...)

- Do que se trata uma abordagem Bayesiana?
  - Variáveis incertas modeladas como randômicas
  - Forte alicerce em conhecimento prévio (priors) e no conceito de prob. condicional
  - Teorema de Bayes para inferência (e derivações)
  - Resultantes de inferência caracterizadas por pdfs (quase sempre intratáveis)

$$P(AB) = P(A|B)P(B) = P(B|A)P(A)$$

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

A ... parâmetros ($\mathbf{w}$)
B ... dados
$P(A)$ ... conhecimento prévio (acumulado)
$P(B|A)$ ... Verossimilhança (likelihood – modelo)

Incertezas : ruídos e “falta” de dados

Realidade....

$$y = f(\mathbf{x}, \mathbf{w}) + \underbrace{\epsilon}_{ruído}$$

modelo

$$y = \mathbf{w}.\phi(x) = \sum_{i}^{p} w_i \phi_i(x)$$

regressão não linear

Verossimilhança (likelihood): introduz um modelo para o ruído

$$\mathbf{y}_{1:n}|\mathbf{x}_{1:n}, \mathbf{w} \sim p(\mathbf{y}_{1:n}|\mathbf{x}_{1:n}, \mathbf{w})$$

Densidade de distribuição de probabilidade (pdf)

O modelo acima expressa a plausibilidade (probabilidade) de se observar y tendo x como entrada e fixando-se (conhecendo-se) os parâmetros w.

Tipicamente

$$p(\mathbf{y}_{1:n}|\mathbf{x}_{1:n}, \mathbf{w}) = \prod_{i=1}^{n} p(y_i|\mathbf{x}_i, \mathbf{w}),$$

$$\begin{aligned} p(y_i|\mathbf{x}_i, \mathbf{w}, \sigma) &= \mathcal{N}\left(y_i|y(\mathbf{x}_i; \mathbf{w}), \sigma^2\right) \\ &= \mathcal{N}\left(y_i|\mathbf{w}^{\mathsf{T}}\phi(\mathbf{x_i}), \sigma^2\right) \end{aligned}$$

$$p(\mathbf{y}_{1:n}|\mathbf{x}_{1:n}, \mathbf{w}, \sigma) = (2\pi)^{-\frac{n}{2}}\sigma^{-n}e^{-\frac{1}{2\sigma^2}\|\mathbf{\Phi w}-\mathbf{y}_{1:n}\|^2}$$

Uma primeira solução: maximizando a verossimilhança

$$\mathbf{w}_{\text{MLE}} = \arg\max_{\mathbf{w}} p(\mathbf{y}_{1:n}, |\mathbf{x}_{1:n}, \mathbf{w}, \sigma)$$

$$\log p(\mathbf{y}_{1:n}, |\mathbf{x}_{1:n}, \mathbf{w}, \sigma) = -\frac{n}{2}\log(2\pi) - n\log\sigma - \frac{1}{2\sigma^2}\|\mathbf{\Phi w} - \mathbf{y}_{1:n}\|^2.$$

$$\mathbf{w}_{\text{MLE}} \equiv \mathbf{w}_{\text{LS}}$$

Nesta solução (não bayesiana) a versão probabilística coincide com mínimos quadrados, no entanto:

$$\sigma^2_{\text{MLE}} = \frac{\|\mathbf{\Phi w} - \mathbf{y}_{1:n}\|^2}{n}$$

$$p(y|\mathbf{x}, \mathbf{w}_{\text{MLE}}, \sigma^2_{\text{MLE}}) = \mathcal{N}\left(y\middle|\mathbf{w}^T_{\text{MLE}}\phi(\mathbf{x}), \sigma^2_{\text{MLE}}\right)$$

Ou...

$$log\ p(\mathbf{w}|\mathbf{x}_{1:n}, y_{1:n}, sigma) = \frac{1}{2}||\Phi\mathbf{w} - y_{1:n}||^2 - \frac{\alpha}{2}||\mathbf{w}||^2$$

$$\mathbf{w}_{MPE} = (\sigma^{-2}\Phi^T\Phi + \alpha I)^{-1}\Phi^T y_{1:n}$$

$$[\Phi]_{nxp}.\ \ com\ \Phi_{ij} = \phi_i(x_i))$$

ou ainda (fully Bayesian)

$$p(\mathbf{w}|y_{1:n}, \mathbf{x}_{1:n}, \sigma, \alpha) = \mathcal{N}(\mathbf{w}|\mathbf{m}, S)$$

$$S = (\sigma^{-2}\Phi^T\Phi + \alpha I)^{-1}$$

$$\mathbf{m} = \sigma^{-2}S\Phi^T y_{1:n}$$

A probabilistic predictive model: an abstract overview

The (often expensive to compute) high-fidelity model

$$\mathbf{y} = \mathcal{M}(\mathbf{x}; \theta)$$

with θ a vector of uncertain (random) parameters (at this point assumed to be known)

Making predictions (taking into consideration the uncertainties, marginalization:

$$p(y^*|x^*) = \int \underbrace{p(y^*|x^*, \theta)}_{\mathcal{M}(\mathbf{x};\theta)} \, p(\theta) \, d\theta$$

The above integral might be approximated through a Monte Carlo sampling algorithm

$$p(y^*|x^*) = \int \underbrace{p(y^*|x^*, \theta)}_{\mathcal{M}(\mathbf{x};\theta)} \, p(\theta) \, d\theta \approx \frac{1}{M} \sum_{m=1}^{M} p(y^*|x^*, \theta_m)$$

A probabilistic predictive model: the need for a (ML) surrogate

Assuming that $\mathcal{M}$ is deterministic

$$\mathbb{E}(y^*) \approx \frac{1}{M} \sum_{m=1}^{M} \mathcal{M}(x^*, \theta_m)$$

remembering that we want more.. we want to estimate uncertainties in the predictions (e.g. the variance that requires bigger M). So we need a cheap-to-compute surrogate for $\mathcal{M}$. Gaussian Processes and (Deep) Neural Networks are good candidates.

Supervised Learning: from N samples computed with the expensive high-fidelity model, built an ML surrogate