

Lecture 1

SEC-204
10.20-11.40 am MTh
PHYS568

Introduction

Data analysis: automated discovery of regularities & patterns in data, subsequent modeling (e.g. physical modeling)

Ex.: digit recognition: pixels \rightarrow $\{0, \dots, 9\}$
 \vec{x} (vector of pixels) single-digit output

Training set: $\{\vec{x}_1, \dots, \vec{x}_N\}$
(e.g. hand-labeled) \Downarrow
 \vec{t} (N-dim target vector)
 $t_i = \{0, \dots, 9\}$

Trained model can then be used to predict digits in a test set.

often: preprocessing / feature extraction

\downarrow
find useful features that are fast to compute

Supervised learning problem: training, then predicting.

\downarrow
classification
[discrete categories]

\downarrow
regression
[real-valued predictions]

Unsupervised learning problem:

$\{\vec{x}_1, \dots, \vec{x}_N\}$ but no target values

↓
clustering

↓
density estimation

↓
dimensionality reduction (visualization)

Reinforcement learning

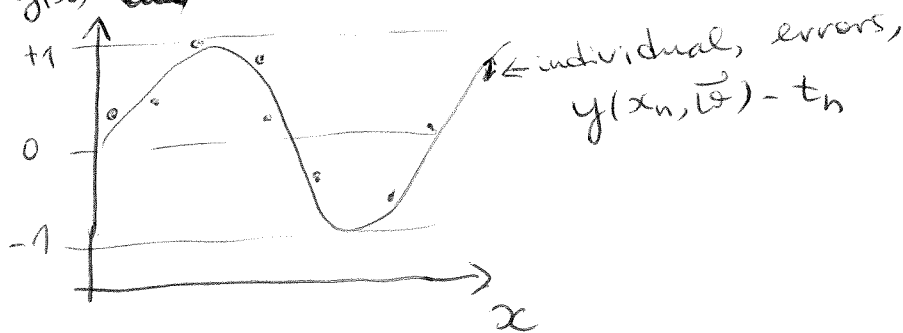
no training data at all \rightarrow learn by trial and error

Objective: max score or reward

Ex.: polynomial curve fitting

\vec{x}_N comes from $\sin(2\pi x) + \text{noise}$
($N=10$) $y(x)$ ~~curve~~

\vec{t} also given, real-valued



Consider curve fitting:

$$y(x, \vec{w}) = \sum_{j=0}^M w_j x^j \quad \leftarrow \text{linear model (in } \vec{w})$$

M -polynomial order

Minimize the error $f'n$:

$$E(\vec{w}) = \frac{1}{2} \sum_{n=1}^N (y(x_n, \vec{w}) - t_n)^2$$

$\frac{\partial E}{\partial w_j} = 0$ has a closed-form solution \rightarrow
 $\rightarrow \vec{w}^*$

$$\frac{\partial E(\vec{w})}{\partial w_j} = \sum_{n=1}^N (y(x_n, \vec{w}) - t_n) \frac{\partial y(x_n, \vec{w})}{\partial w_j} =$$

$$= \sum_{n=1}^N (y(x_n, \vec{w}) - t_n) x_n^j = 0 \quad j=0, \dots, M$$

$$\underbrace{y(x_n, \vec{w})}_{\sum_{j=0}^M w_j x_n^j}$$

$$\sum_{n=1}^N t_n x_n^j = \sum_{j'=0}^M w_{j'} \left[\sum_{n=1}^N x_n^{j'} x_n^j \right]$$

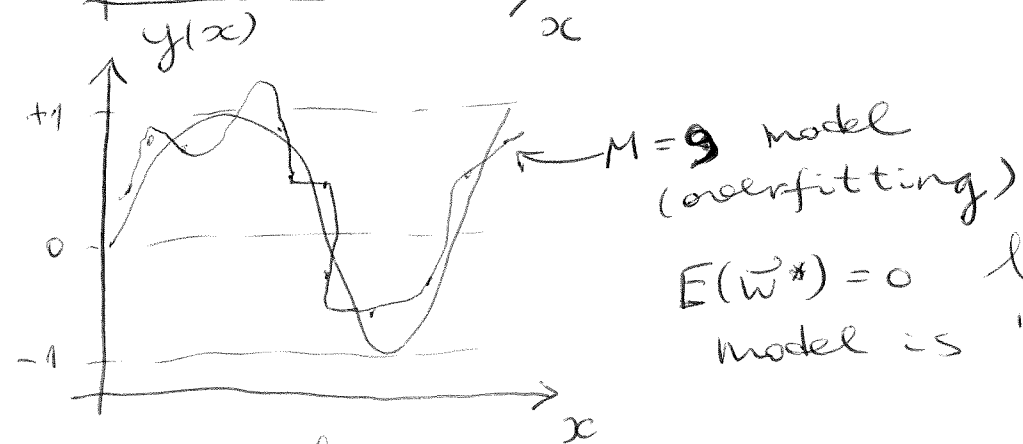
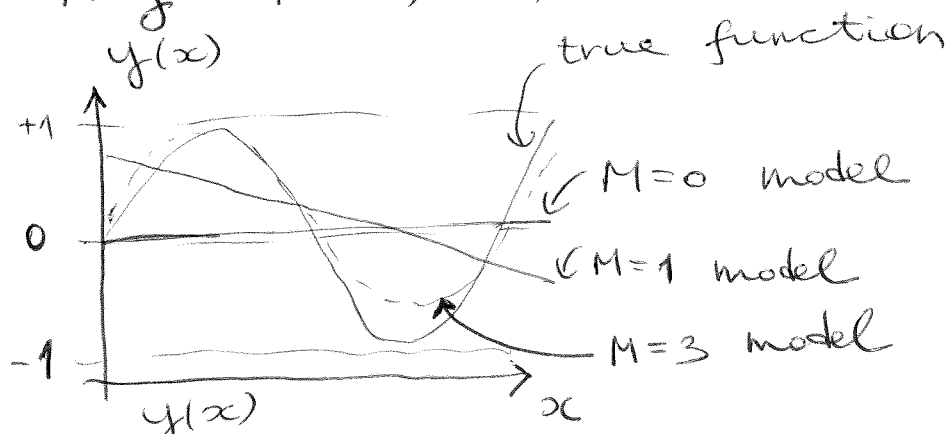
linear in w_j $\Rightarrow C_j$

M_{jj}

$\sum_{j'=0}^M w_{j'} M_{jj'} = C_j \leftarrow$ system of $M+1$ linear eq's for w_j , can be used to find w_j^{ML} .

How to choose M ? (model selection)

Try $M=0, 1, 3, 9$



$E(\bar{w}^*) = 0$ but the model is "bad"

In what way? Well, $\|w_j^*\|$ are often large & irregular when the model overfits.

How to fix it? One easy way \rightarrow add more data, but the model complexity then depends on N (whereas it should depend on the complexity of the underlying process).

Minimizing least squares is a specific case of the ML approach \rightarrow switching to Bayesian modeling automatically controls model complexity.

Another, more empirical technique is regularization:

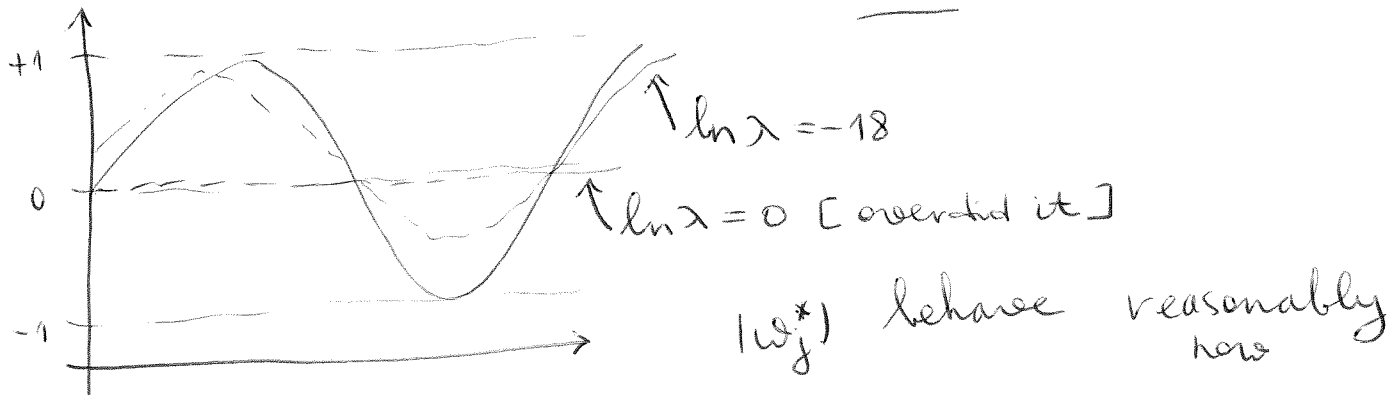
$$\tilde{E}(\vec{w}) = \frac{1}{2} \sum_{n=1}^N (y(x_n, \vec{w}) - t_n)^2 + \frac{\lambda}{2} (w_0^2 + \dots + w_M^2)$$

↑
may be omitted not to
make things origin-dependent

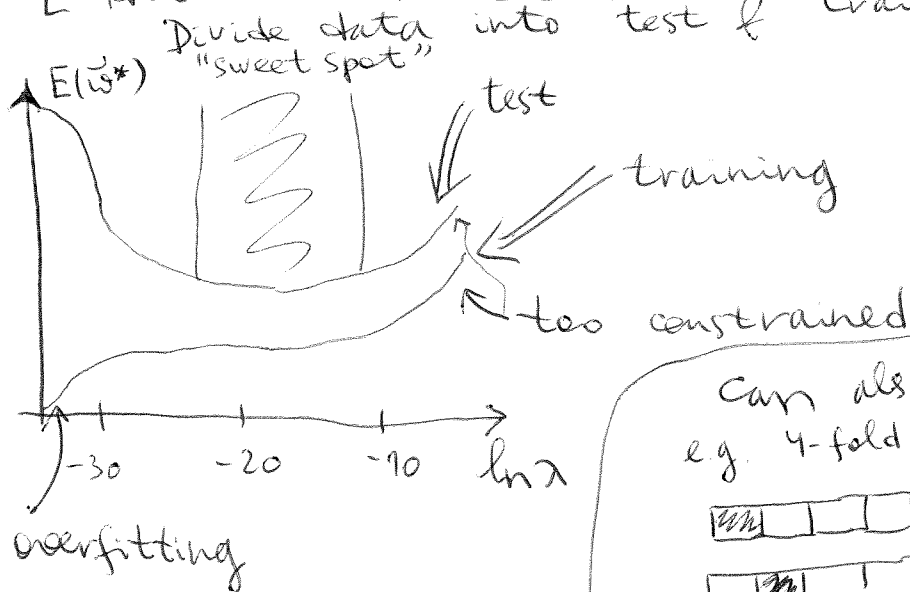
$$\frac{\partial \tilde{E}(\vec{w})}{\partial w_j} = 0, \quad \forall_j \text{ can still be evaluated}$$

Drawback: things depend on λ .

M=9 fit



[How to choose λ ?]



Can also use cross-validation
e.g. 4-fold:



☐ training set



☐ test set



> obtain $k=1..4$



$\langle E(\vec{w}_k^*) \rangle$
averaged over 4 runs

Probabilistic interpretation of curve fitting

Input values: $\vec{x} = (x_1 \dots x_N)$

Target values: $\vec{t} = (t_1 \dots t_N)$

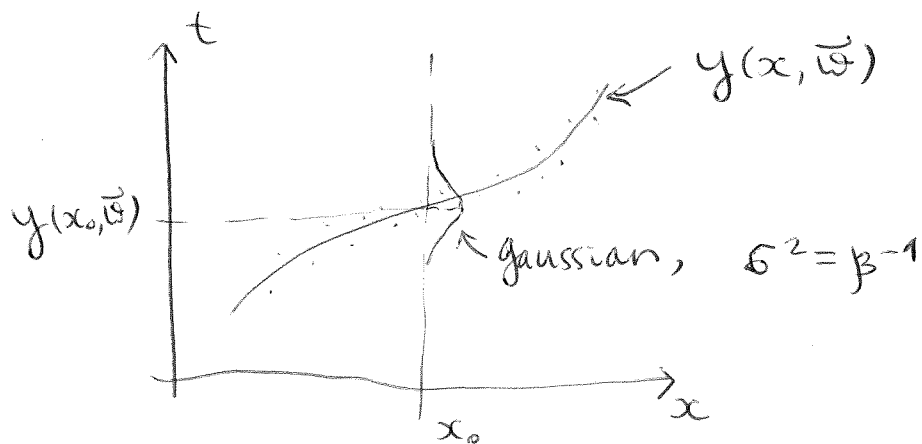
Assume: $p(t|x, \vec{w}, \beta) = \mathcal{N}(t|y(x, \vec{w}), \beta^{-1})$
 Gauss. noise distr'n

$$\mathcal{N}(x|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \quad \beta\text{-precision prm}$$

Likelihood of the data:

$$P(\vec{t}|\vec{x}, \vec{w}, \beta) = \prod_{n=1}^N \mathcal{N}(t_n|y(x_n, \vec{w}), \beta^{-1})$$

↑ independent samples!



$$\log P = - \underbrace{\frac{\beta}{2} \sum_{n=1}^N (y(x_n, \vec{w}) - t_n)^2}_{\beta E(\vec{w})} + \underbrace{\frac{N}{2} \log\left(\frac{\beta}{2\pi}\right)}_{\text{indep. of } \vec{w}}$$

$$\frac{\partial}{\partial w_j} \log P = -\beta \frac{\partial}{\partial w_j} E(\vec{w})$$

Maximizing $\log P$ is the same as minimizing $E(\vec{w})$.
 ML approach! \rightarrow find \vec{w}_{ML}

Further, $\frac{1}{\beta_{ML}} = \frac{1}{N} \sum_{n=1}^N (y(x_n, \vec{w}_{ML}) - t_n)^2$

Now can make predictions:

$$p(t|x, \vec{w}_{ML}, \beta_{ML}) = \mathcal{N}(t | y(x, \vec{w}_{ML}), \beta_{ML}^{-1})$$

A bit more Bayesian:

$$\left[\begin{array}{l} P(\vec{w} | D) = \frac{P(D | \vec{w}) P(\vec{w})}{P(D)} \\ P(D) = \int d\vec{w} P(D | \vec{w}) P(\vec{w}) \end{array} \right]$$

model prms \downarrow $P(\vec{w} | D)$ (posterior)
 data \downarrow $P(D | \vec{w})$ (likelihood)
 prior \downarrow $P(\vec{w})$
 $P(D)$ (evidence)

posterior \sim likelihood \times prior [norm'n restored later]

Assume for the prior:

$$P(\vec{w} | \alpha) = \left(\frac{\alpha}{2\pi} \right)^{\frac{M+1}{2}} \exp \left\{ -\frac{\alpha}{2} \vec{w}^T \vec{w} \right\}$$

α - precision prm

product of 1D gaussians centered on 0

$\sum_j w_j^2$

$$\log P(\vec{w} | \vec{x}, \vec{t}, \alpha, \beta) \sim \underbrace{\log P(t | \vec{x}, \vec{w}, \beta)}_{-\frac{\beta}{2} \sum_{n=1}^N (y(x_n, \vec{w}) - t_n)^2} + \underbrace{\log P(\vec{w} | \alpha)}_{-\frac{\alpha}{2} \vec{w}^T \vec{w}}$$

If we maximize the posterior prob. (instead of considering the whole distrib'n),

we minimize $\frac{\beta}{2} \sum_{n=1}^N (\dots)^2 + \frac{\alpha}{2} \vec{w}^T \vec{w}$

\hookrightarrow same as regularization with $\lambda = \alpha/\beta$