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

Ex. digit recognition: \( \mathbf{x} \rightarrow \{0, \ldots, 9\}^N \) vector of pixel values

Training set: \( \{ \mathbf{x}_1, \ldots, \mathbf{x}_N \} \) (e.g. hand-labeled) \( \Downarrow \)

\( \mathbf{t} \) \( (N \times m \text{ target vector}) \)

\( \mathbf{t}_i = \{0, \ldots, 9\} \)

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, predicting.

\( \checkmark \) classification \( \checkmark \) regression

[discrete categories] \( \checkmark \) [real-valued predictions]
Unsupervised learning problem:
\[ \{ x_1, \ldots, x_N \} \] but no target values

CLUSTERING \quad DENSITY ESTIMATION \quad DIMENSIONALITY REDUCTION (VISUALIZATION)

Reinforcement learning

No training data at all \implies learn by trial and error

Objective: max score or reward

Ex: polynomial curve fitting

\[ y(x) = \sum_{j=0}^{N-1} c_j x^j \quad (N=10) \]

\[ y(x) \approx \sin(2\pi x) + \text{noise} \]

It also given, real-valued

Consider curve fitting:

\[ y(x, \overline{w}) = \sum_{j=0}^{M} w_j x^j \]

M - polynomial order

Minimize the error function:

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

\[ \frac{\partial E}{\partial \overline{w}_j} = 0 \] has a closed-form solution \implies \[ \overline{w}_j^* \]
\[ \frac{\partial E(\mathbf{\theta})}{\partial \theta_j} = \sum_{n=1}^{N} (y(x_n, \mathbf{\theta}) - t_n) \frac{\partial y(x_n, \mathbf{\theta})}{\partial \theta_j} = \]

\[ = \sum_{n=1}^{N} (y(x_n, \mathbf{\theta}) - t_n) x^j_n = 0 \]

\[ \sum_{n=1}^{N} t_n x^j_n = \sum_{j=0}^{M} \mathbf{\psi}_j^T \left[ \sum_{n=1}^{N} x^j_n x^j_n \right] \]

linear in \( \mathbf{\psi}_j \)

\[ \sum_{j=0}^{M} \mathbf{\psi}_j^T \mathbf{M} \mathbf{\psi}_j = \mathbf{C}_j \]

So, \( \sum_{j=0}^{M} \mathbf{\psi}_j^T \mathbf{M} \mathbf{\psi}_j = \mathbf{C}_j \)

can be used to find \( \mathbf{\psi}_j \).
How to choose $M$? (model selection)

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

True function

$\begin{array}{c}
\text{M=0 model} \\
\text{M=1 model} \\
\text{M=3 model}
\end{array}$

$E(w^*) = 0$ but the model is "bad"

In what way? Well, $\|w\|_1$ are often large & irregular when the model overfits.

How to fix it? One easy way is to 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 → switching to Bayesian modeling automatically controls model complexity.
Another, more empirical technique is regularization:

\[
\hat{E}(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^{N} (y(x_n, \mathbf{w}) - t_n)^2 + \frac{\lambda}{2} (w_o^2 + \cdots + w_M^2)
\]

\[
\frac{\partial \hat{E}(\mathbf{w})}{\partial w_j} = 0, \quad \forall j \text{ can still be evaluated}
\]

Drawback: things depend on \( \lambda \).

\( M = 9 \) fit

\[ \ln \lambda = -18 \]

\[ \ln \lambda = 0 \text{ [overfit at]} \]

[How to choose \( \lambda^2 \)]

Divide data into test & training sets

\( E(\mathbf{w}^*) \) behave reasonably near

[too constrained]

Can also use cross-validation

E.g. 4-fold:

\[ \text{training set} \]

\[ \text{test set} \]

\( \text{obtain } k=1.4 \)

\( \text{averaged over 4 runs} \)
Probabilistic interpretation of curve fitting

Input values: \( \tilde{X} = (x_1, \ldots, x_N) \)
Target values: \( \tilde{t} = (t_1, \ldots, t_N) \)

Assume: \( p(t_1 | x, \bar{\omega}, \beta) = N(t_1 | y(x, \bar{\omega}), \beta^{-1}) \)

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

Likelihood of the data:

\[
P(\tilde{t} | \tilde{x}, \bar{\omega}, \beta) = \prod_{n=1}^{N} N(t_n | y(x_n, \bar{\omega}), \beta^{-1})
\]
\( \uparrow \) independent samples!

\[
\log P = -\frac{\beta}{2} \sum_{n=1}^{N} (y(x_n, \bar{\omega}) - t_n)^2 + \frac{N}{2} \log \left( \frac{\beta}{2\pi \sigma} \right)
\]

\[
\frac{\partial}{\partial \bar{\omega}} \log P = -\beta \frac{2}{\sigma} E(\bar{\omega})
\]

Maximizing \( \log P \) is the same as minimizing \( E(\bar{\omega}) \), \( \Rightarrow \) find \( \bar{\omega}_{ML} \)
Further, \( \frac{1}{\beta_{\text{ML}}} = \frac{1}{N} \sum_{n=1}^{N} (y(x_n, \bar{\omega}) - t_n)^2 \)

Now can make predictions:

\[
p(t|x, \bar{\omega}_{\text{ML}}, \beta_{\text{ML}}) = N(t|y(x, \bar{\omega}_{\text{ML}}), \beta_{\text{ML}}^{-1})
\]

A bit more Bayesian:

\[
\begin{bmatrix}
P(t|x) = \frac{P(D|t)\ P(t)}{P(D)} \\
P(D) = \int d\omega\ P(D|\omega)\ P(\omega)
\end{bmatrix}
\]

posterior \(-\) likelihood \(\times\) prior \(\\sim\) normal restored later

Assume for the prior:

\[
P(\omega|x) = \left(\frac{1}{2\pi^d}\right)^{\frac{M+1}{2}} \exp \left(-\frac{1}{2} \omega^T \omega\right)
\]

product of \(d\) Gaussians centered on \(\Omega\)

\(d\) - precision param

\[
\log P(\omega|x, t, \bar{\omega}, \beta) \sim \log P(t|x, \bar{\omega}, \beta) + \log P(\omega|x)
\]

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

we minimize

\[
\frac{1}{2} \sum_{n=1}^{N} (y(x_n, \bar{\omega}) - t_n)^2 + \frac{1}{2} \bar{\omega}^T \bar{\omega}
\]

\(\sim\) same as regularization

with \(\lambda = \frac{1}{\beta}\)

\(-7-\)