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

Ex.: digit recognition: $\mathbf{x} \rightarrow \{0,\ldots,9\}$

\[ \mathbf{x} \] vector of pixels
\[ \rightarrow \mathbf{y} \] single-digit output

Training set: $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$
(e.g. hand-labeled) \( \Downarrow \)
\[ \mathbf{t} \] (N-dim target vector)
\[ t_i \in \{0, \ldots, 9\} \]

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

Often: preprocessing / feature extraction
\[ \nabla \] find useful features that are fast to compute

Supervised learning problem: training, predicting.

\[ \checkmark \] classification
\[ \{\text{discrete categories}\} \]

\[ \checkmark \] regression
\[ \{\text{real-valued predictions}\} \]
Unsupervised learning problem:
\[ \{ x_1, ..., x_N \} \] 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

Example: polynomial curve fitting
\[ y(x) \]
\( (N=10) \)
\[ \bar{x}_N \text{ comes from } \sin(2\pi x) + \text{noise} \]

\( \bar{x}_N \) also given, real-valued

Consider curve fitting:
\[ y(x, \bar{w}) = \sum_{j=1}^{M} w_j x^j \leq \text{linear model } (\text{in } \bar{w}) \]

- polynomial order

Minimize the error fn:
\[ E(\bar{w}) = \frac{1}{2} \sum_{n=1}^{N} (y(x_n, \bar{w}) - t_n)^2 \]

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

\[ = \sum_{n=1}^{N} (y(x_n, \theta) - t_n)x_n^j = 0 \quad \text{linear in } \theta_j \]

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

So, \( \sum_{j=0}^{M} \theta_j M_{jj} = \theta_j \) can be used to find \( \theta_j \).
Plan to choose $M$? (model selection)

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

In what way? Well, $|w_j|$ 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**:

\[
\hat{E}(\vec{w}) = \frac{1}{2} \sum_{n=1}^{N} (y(x_n, \vec{w}) - t_n)^2 + \frac{\lambda}{2} (\|w_0\|^2 + \ldots + \|w_M\|^2)
\]

This may be omitted not to make things origin-dependent.

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

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

- For \( M = 9 \) fit:
  - \( \ln \lambda = -18 \)
  - \( \ln \lambda = 0 \) [overfit it]
  - \( \lambda^{*} \) behave reasonably now

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

Divide data into test & training sets.

**E(\(\lambda^{*}\))** "sweet spot"

- **test**
- **training**

Can also use cross-validation:

- e.g. 4-fold:
  - \( \frac{\lambda}{4} \) training set
  - \( \frac{\lambda}{4} \) test set

\( \lambda \) obtain \( k \approx 4.4 \)

\( \frac{1}{4} \langle E(\lambda^{*}_k) \rangle \)

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: \( p(t | x, \tilde{w}, \beta) = N(t | y(x, \tilde{w}), \beta^{-1}) \)

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

Likelihood of the data:

\[ p(\tilde{t} | x, \tilde{w}, \beta) = \prod_{n=1}^{N} N(t_n | y(x_n, \tilde{w}), \beta^{-1}) \]

\( \uparrow \) independent samples!

\[ y(x, \tilde{w}) \]

\[ y(x, \tilde{w}) \]

\[ \log p = -\frac{\beta}{2} \sum_{n=1}^{N} (y(x_n, \tilde{w}) - t_n)^2 + \frac{N}{2} \log \left( \frac{\beta}{2\sigma^2} \right) \]

\[ \frac{\partial}{\partial \tilde{w}_j} \log p = -\beta \frac{2}{\sigma^2} E(\tilde{w}) \]

Maximizing \( \log p \) is the same as minimizing \( E(\tilde{w}) \).

\( \Rightarrow \) find \( \tilde{w}_{\text{ML}} \)
Further, \( \frac{1}{\beta_{ML}} = \frac{1}{N} \sum_{n=1}^{N} (y(x_n, \overline{w}_{ML}) - t_n)^2 \)

Now can make predictions:

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

A bit more Bayesian: likelihood \& prior

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

\[\text{posterior} = \text{likelihood} \times \text{prior}\]

Assume for the prior:

\[
P(\overline{w} | D) = \left( \frac{1}{2\pi} \right)^{M/2} e^{-\frac{1}{2} \overline{w}^T \overline{w}}
\]

\[\text{product of } M \text{ gaussian centered on } \emptyset\]

\[\text{product of } \beta \text{ gaussian centered on } \emptyset\]

\[
\log P(\overline{w} | x, t, \beta) \sim \log P(\overline{w} | x, t, \beta) + \log P(\overline{w} | D)
\]

\[
\log P(\overline{w} | x, t, \beta) = -\frac{1}{2} \overline{w}^T \overline{w} - \frac{1}{2} \sum_{n=1}^{N} (y(x_n, \overline{w}) - t_n)^2
\]

If we maximize the posterior prob.

(Instead of considering the whole distribution),

we minimize

\[
\frac{1}{2} \sum_{n=1}^{N} (\ldots)^2 + \frac{1}{2} \overline{w}^T \overline{w}
\]

\(\Leftrightarrow\) same as regularization

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