Theme: Curve Fitting

Mark van der Wilk

Department of Computing
Imperial College London

@markvanderwilk
m.vdwilk@imperial.ac.uk

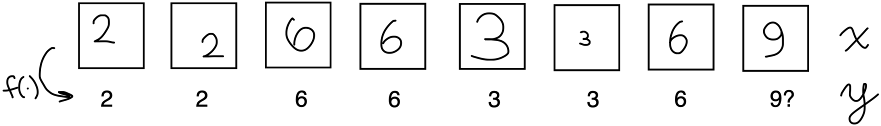
October 3, 2022

Curve Fitting (Regression) Examples

We will be considering *curve fitting* or *supervised learning*.

- Given a dataset of N examples of inputs and outputs...
- predict what the output will be for a new input.

Image classification. Inputs $\in \mathbb{R}^D$, outputs $\in \mathbb{N}$:



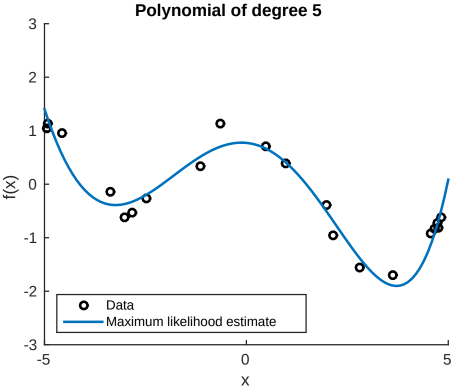
Translation. Inputs $\in \bigcup_{\ell=1}^{\infty} \mathbb{N}^{\ell}$, outputs $\in \bigcup_{k=1}^{\infty} \mathbb{N}^{k}$:

`Wiskunde is belangrijk.` $\rightarrow$ `Mathematics is important.`

Dutch English

Regression Example

Curve fitting in 1D. Inputs $\in \mathbb{R}$, outputs $\in \mathbb{R}$:



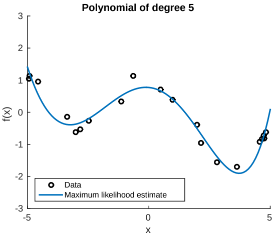
Curve fitting

"All the impressive achievements of deep learning amount to just curve fitting."

— Judea Pearl

Curve Fitting: Representing functions

Q: How do we represent functions?



- We need a *collection* of functions from which to pick a good one.
- **Parameterise** a set of functions, i.e. take some numbers $\boldsymbol{\theta}$ that map to a function.

For example, linear or polynomial functions:

$$f_{\boldsymbol{\theta}}(x) = a \cdot x + b\,, \qquad \boldsymbol{\theta} = \begin{bmatrix} a \\ b \end{bmatrix}, \tag{1}$$

$$f_{\boldsymbol{\theta}}(x) = a \cdot x^3 + b \cdot x^2 + c \cdot x + d\,, \qquad \boldsymbol{\theta} = \begin{bmatrix} a & b & c & d \end{bmatrix}^{\mathrm{T}}. \tag{2}$$

Linear-in-the-Parameters representation

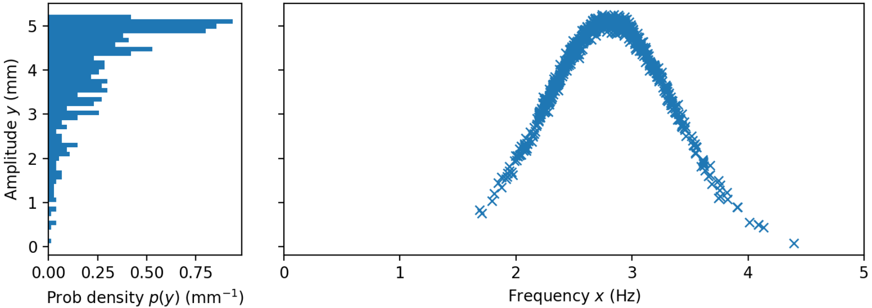
$$f_{\boldsymbol{\theta}}(x) = \boldsymbol{\phi}(x)^{\mathsf{T}}\boldsymbol{\theta} \tag{3}$$

$$\boldsymbol{\phi}(x) \qquad \text{Feature vector} \tag{4}$$

$$\boldsymbol{\theta} \qquad \text{Parameters} \tag{5}$$

- Can use this to represent complicated functions.
- We call this “linear in the parameters” because the relationship betwen the function values and the parameters is linear.
- This enables regressio solutions to be found in closed form.

Regression Example



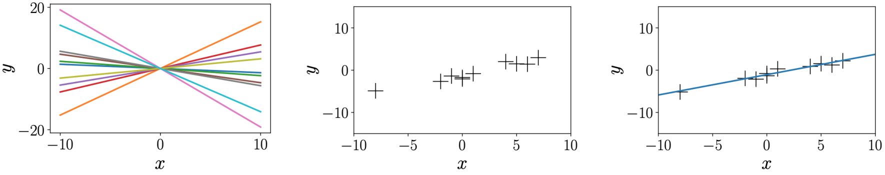
For some observed x, the world is generating data from $\pi(y|x)$.

We can choose two possible goals for regression:

- Loss view: Find a function $f(x)$ that goes "near" outputs y.
- Stats view: Match a statistical model $p(y|x, \boldsymbol{\theta})$ to $\pi(y|x)$.

Loss view: Good and bad functions

We now have many functions that we can choose from:



Left: example functions. Middle: Training set. Right: A good fit. Source: Mathematics for Machine Learning book.

Q: Which function do we pick?

- Need to define what good and bad functions are. Good functions have $f(\boldsymbol{x}_i, \boldsymbol{\theta}^*) \approx y_i$.
- Define a loss function, e.g., $L(\boldsymbol{\theta}) = \sum_{i=1}^{N} (y_i - f(\boldsymbol{x}_i, \boldsymbol{\theta}))^2$
- Choose a good function, i.e. $\boldsymbol{\theta}^* = \operatorname{argmin}_{\boldsymbol{\theta}} L(\boldsymbol{\theta})$

Maximum Likelihood Estimation

Revision from `50008: Probability & Statistics`

- Model is a probability distribution on data: $p(y|\boldsymbol{\theta})$
- For an observed dataset (fixed), we can evaluate the probability assigned to it for different $\boldsymbol{\theta}$
- This defines the likelihood $\ell(\boldsymbol{\theta}) = p(y|\boldsymbol{\theta})$

Maximum likelihood does:

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\, \ell(\boldsymbol{\theta}) = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \log \ell(\boldsymbol{\theta}) \tag{6}$$

Likelihood for Linear Regression

Assume:

- Gaussian deviations from the function:

$$p(y_n|x_n, \boldsymbol{\theta}) = \mathcal{N}\left(y_n; f_{\boldsymbol{\theta}}(x_n), \sigma^2\right) \tag{7}$$

- Independent deviations between datapoints. So denoting $y \in \mathbb{R}^N$, $x \in \mathbb{R}^N$ for N datapoints, we get the likelihood:

$$p(y|x, \boldsymbol{\theta}) = \prod_{n=1}^{N} \mathcal{N}\left(y_n; f_{\boldsymbol{\theta}}(x_n), \sigma^2\right) \tag{8}$$

- You will show that this is equivalent to the loss view (exercises).

Curve Fitting Summary

- Training data, e.g., N pairs $(\boldsymbol{x}_i, y_i)$ of inputs $\boldsymbol{x}_i$ and observations y_i
- Parameterise functions as $f(\mathbf{x}_i, \boldsymbol{\theta})$
- Training the model means finding parameters $\boldsymbol{\theta}^*$, such that
  - $f(\boldsymbol{x}_i, \boldsymbol{\theta}^*) \approx y_i$ (loss is minimised)
  - $p(y_n|x_n, \boldsymbol{\theta}) \approx \pi(y_n|x_n, \boldsymbol{\theta})$ (max likelihood)
- Not discussed: How to find $\boldsymbol{\theta}^*$