Probabilistic Machine Learning
Lecture 07
Parametric Regression

Philipp Hennig

11 May 2023

Eberhard Karls Universität Tübingen

Faculty of Science
Department of Computer Science
Chair for the Methods of Machine Learning

Recap: Gaussian Distributions

Gaussian distributions provide the linear algebra of inference

- products of Gaussians are Gaussians $\qquad C := (A^{-1} + B^{-1})^{-1} \quad c := C(A^{-1}a + B^{-1}b)$

$$\mathcal{N}(x; a, A)\mathcal{N}(x; b, B) = \mathcal{N}(x; c, C)\mathcal{N}(a; b, A + B)$$

- linear projections of Gaussians are Gaussians

$$p(z) = \mathcal{N}(z; \mu, \Sigma) \quad \Rightarrow \quad p(Az) = \mathcal{N}(Az, A\mu, A\Sigma A^\intercal)$$

- marginals of Gaussians are Gaussians

$$\int \mathcal{N}\left[\begin{bmatrix} x \\ y \end{bmatrix}; \begin{bmatrix} \mu_x \\ \mu_y \end{bmatrix}, \begin{bmatrix} \Sigma_{xx} & \Sigma_{xy} \\ \Sigma_{yx} & \Sigma_{yy} \end{bmatrix}\right] dy = \mathcal{N}(x; \mu_x, \Sigma_{xx})$$

- (linear) conditionals of Gaussians are Gaussians

$$p(x \mid y) = \frac{p(x, y)}{p(y)} = \mathcal{N}\left(x; \mu_x + \Sigma_{xy}\Sigma_{yy}^{-1}(y - \mu_y), \Sigma_{xx} - \Sigma_{xy}\Sigma_{yy}^{-1}\Sigma_{yx}\right)$$

Bayesian inference becomes linear algebra $\qquad p(x) = \mathcal{N}(x; \mu, \Sigma) \qquad p(y \mid x) = \mathcal{N}(y; A^\intercal x + b, \Lambda)$

$$p(B^\intercal x + c \mid y) = \mathcal{N}[B^\intercal x + c; B^\intercal \mu + c + B^\intercal \Sigma A(A^\intercal \Sigma A + \Lambda)^{-1}(y - A^\intercal \mu - b), B^\intercal \Sigma B - B^\intercal \Sigma A(A^\intercal \Sigma A + \Lambda)^{-1}A^\intercal \Sigma B]$$

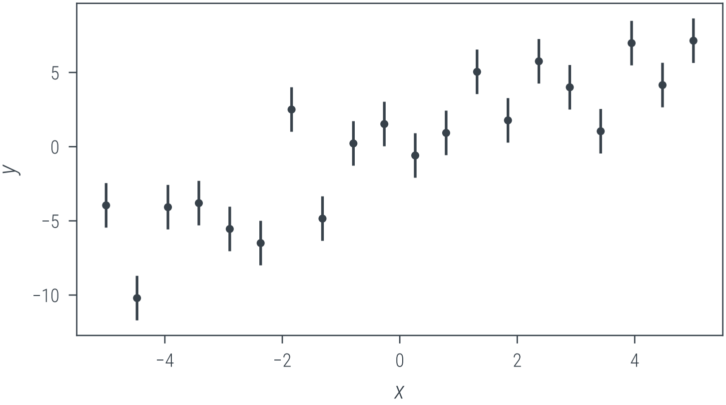
Code

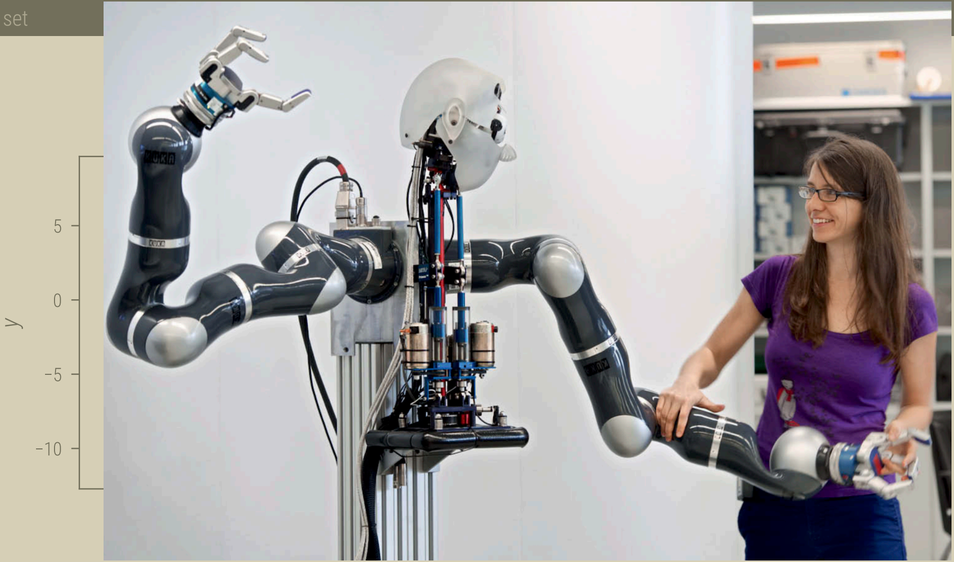
`gaussians.py`

Supervised Regression

A data set

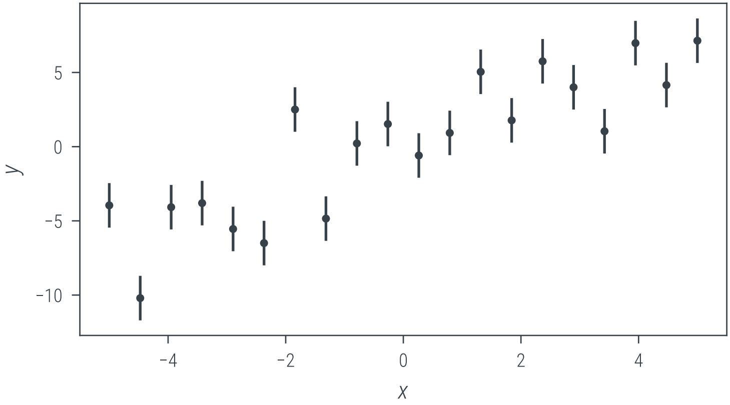
given: $\boldsymbol{y} \in \mathbb{R}^N, \quad p(\boldsymbol{y} \mid f) = \mathcal{N}(\boldsymbol{y}; \boldsymbol{f}(\boldsymbol{x}), \sigma^2 I_N).$ What is f?



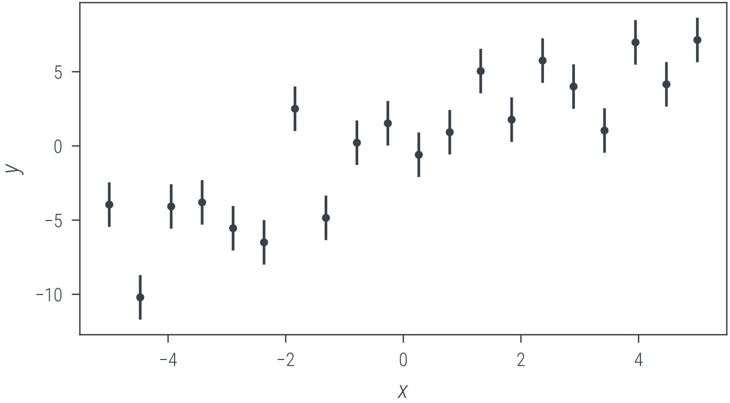

5
0
−5
−10
y

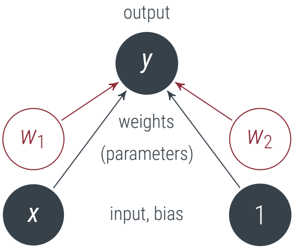
A data set

given: $\boldsymbol{y} \in \mathbb{R}^N, \quad p(\boldsymbol{y} \mid f) = \mathcal{N}(\boldsymbol{y}; \boldsymbol{f}(\boldsymbol{x}), \sigma^2 I_N).$ What is f?



Assume **linear** function $f(x) = w_1 + w_2 x = \phi_x^\intercal \mathbf{w}$ with **features** $\phi(x) = \begin{bmatrix} 1 \\ x \end{bmatrix} =: \phi_x$

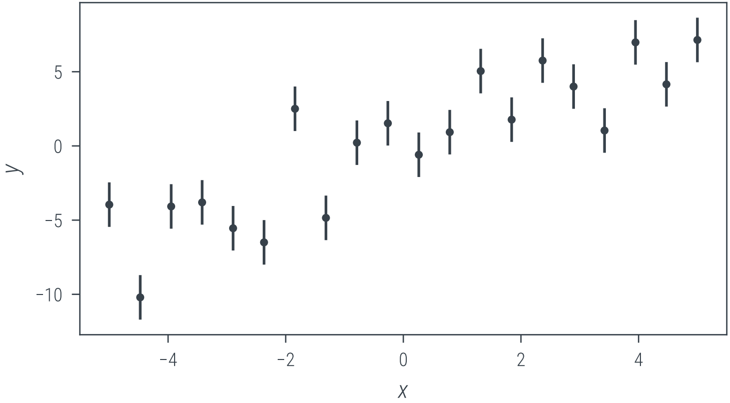


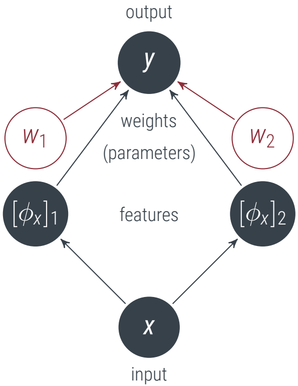


A Linear Model

linear regression

Assume **linear** function $f(x) = w_1 + w_2 x = \phi_x^\intercal w$ with **features** $\phi(x) = \begin{bmatrix} 1 \\ x \end{bmatrix} =: \phi_x$



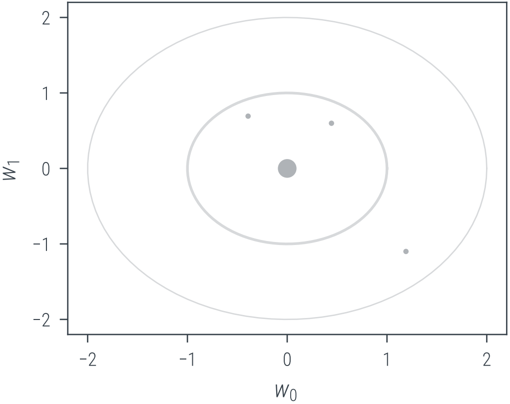


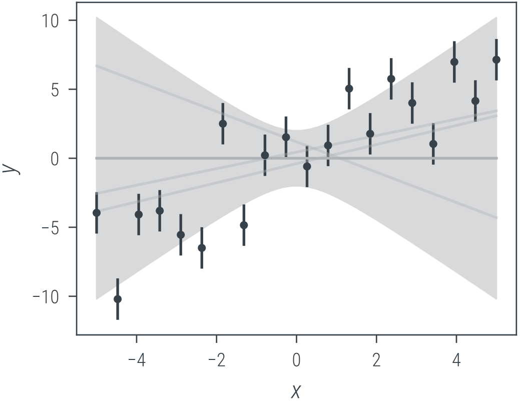
A linear **generative** model

if every variable is Gaussian and every relationship is linear, all marginals and conditionals are also Gaussian

$$f(x) = w_1 + w_2 x = \phi_x^\intercal w$$
$$p(w) = \mathcal{N}(w; \mu, \Sigma)$$

$$p(f) = \mathcal{N}(f; \phi_x^\intercal \mu, \phi_x^\intercal \Sigma \phi_x)$$





Notation

this will become exceedingly helpful later on

Dataset: $X := \begin{bmatrix} x_1 \\ \vdots \\ x_N \end{bmatrix} \in \mathbb{X}^N, \boldsymbol{y} := \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix} \in \mathbb{R}^N$. We will use the following very sloppy notation, sloppily

$$\phi_x := \phi(x) = \begin{bmatrix} 1 \\ x \end{bmatrix} \in \mathbb{R}^F \qquad \phi_X := \begin{bmatrix} \phi(x_1) & \phi(x_2) & \cdots & \phi(x_N) \end{bmatrix} = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ x_1 & x_2 & \cdots & x_N \end{bmatrix} \in \mathbb{R}^{F \times N}$$

$$f_x := f(x) \in \mathbb{R} \qquad f_X := \phi_X^\intercal \boldsymbol{w} = \begin{bmatrix} \phi_1(x_1) & \phi_2(x_1) \\ \phi_1(x_2) & \phi_2(x_2) \\ \vdots & \vdots \\ \phi_1(x_N) & \phi_2(x_N) \end{bmatrix} \cdot \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} \phi_{x_1}^\intercal \boldsymbol{w} \\ \phi_{x_2}^\intercal \boldsymbol{w} \\ \vdots \\ \phi_{x_N}^\intercal \boldsymbol{w} \end{bmatrix} = \begin{bmatrix} f(x_1) \\ f(x_2) \\ \vdots \\ f(x_N) \end{bmatrix} \in \mathbb{R}^N$$

Think of f as an infinitely long vector, indexed by x:

$$\boldsymbol{v} \in \mathbb{R}^N, I \in \mathbb{N}^d \Rightarrow v_I := [v_{I_1}, \dots, v_{I_d}] \in \mathbb{R} \quad \Longleftrightarrow \quad f \in \mathbb{R}^\infty, X \in \mathbb{R}^N \Rightarrow f_X := [f_{x_1}, \dots, f_{x_N}] \in \mathbb{R}^N.$$

Code

our work pays off

```python
from gaussians import Gaussian
from jax import numpy as jnp

# define prior in weight space
prior = Gaussian(mu=jnp.zeros(2), Sigma=jnp.eye(2))
# map into function space
phi = lambda x: jnp.hstack([jnp.ones_like(x), x])
x = jnp.linspace(-5, 5, 100)[:, None]
f_prior = phi(x) @ prior
```

Gaussian Inference on a linear function

weight space / function space

$$\text{prior} \quad p(w) = \mathcal{N}(w; \mu, \Sigma) \quad \Rightarrow \quad p(f) = \mathcal{N}(f_x; \phi_x^\intercal \mu, \phi_x \Sigma \phi_x)$$

$$\text{likelihood} \quad p(y \mid w, \phi_X) = \mathcal{N}(y; \phi_X^\intercal w, \sigma^2 I) = \mathcal{N}(y; f_X, \sigma^2 I)$$

Gaussian Inference on a linear function

weight space / function space

$$
\begin{aligned}
\text{prior} \quad p(w) &= \mathcal{N}(w; \mu, \Sigma) \quad \Rightarrow \quad p(f) = \mathcal{N}(f_x; \phi_X^\intercal \mu, \phi_x \Sigma \phi_x) \\
\text{likelihood} \quad p(y \mid w, \phi_X) &= \mathcal{N}(y; \phi_X^\intercal w, \sigma^2 I) = \mathcal{N}(y; f_X, \sigma^2 I) \\
\text{posterior on } \boldsymbol{w} \quad p(w \mid \boldsymbol{y}, \phi_X) &= \mathcal{N}(w; \mu + \Sigma\phi_X(\phi_X^\intercal \Sigma \phi_X + \sigma^2 I)^{-1}(\boldsymbol{y} - \phi_X^\intercal \mu), \\
&\qquad \Sigma - \Sigma\phi_X(\phi_X^\intercal \Sigma \phi_X + \sigma^2 I)^{-1} \phi_X^\intercal \Sigma) \\
&= \mathcal{N}\left(w; (\Sigma^{-1} + \sigma^{-2}\phi_X \phi_X^\intercal)^{-1} \left(\Sigma^{-1}\mu + \sigma^{-2}\phi_X \boldsymbol{y}\right),\right. \\
&\qquad \left.(\Sigma^{-1} + \sigma^{-2}\phi_X \phi_X^\intercal)^{-1}\right)
\end{aligned}
$$

Gaussian Inference on a linear function

weight space / function space

$$
\begin{aligned}
\text{prior} \quad p(w) &= \mathcal{N}(w; \mu, \Sigma) \quad \Rightarrow \quad p(f) = \mathcal{N}(f_x; \phi_x^\intercal \mu, \phi_x \Sigma \phi_x) \\
\text{likelihood} \quad p(y \mid w, \phi_X) &= \mathcal{N}(y; \phi_X^\intercal w, \sigma^2 I) = \mathcal{N}(y; f_X, \sigma^2 I) \\
\text{posterior on } \boldsymbol{w} \quad p(w \mid \boldsymbol{y}, \phi_X) &= \mathcal{N}(w; \mu + \Sigma\phi_X(\phi_X^\intercal \Sigma \phi_X + \sigma^2 I)^{-1}(\boldsymbol{y} - \phi_X^\intercal \mu), \\
&\qquad \Sigma - \Sigma\phi_X(\phi_X^\intercal \Sigma \phi_X + \sigma^2 I)^{-1}\phi_X^\intercal \Sigma) \\
&= \mathcal{N}\left(w; (\Sigma^{-1} + \sigma^{-2}\phi_X\phi_X^\intercal)^{-1}\left(\Sigma^{-1}\mu + \sigma^{-2}\phi_X \boldsymbol{y}\right),\right. \\
&\qquad \left.(\Sigma^{-1} + \sigma^{-2}\phi_X\phi_X^\intercal)^{-1}\right) \\
\text{posterior on } f \quad p(f_x \mid y, \phi_X) &= \mathcal{N}(f_x; \phi_x^\intercal \mu + \phi_x^\intercal \Sigma\phi_X(\phi_X^\intercal \Sigma \phi_X + \sigma^2 I)^{-1}(y - \phi_X^\intercal \mu), \\
&\qquad \phi_x^\intercal \Sigma \phi_x - \phi_x^\intercal \Sigma\phi_X(\phi_X^\intercal \Sigma \phi_X + \sigma^2 I)^{-1}\phi_X^\intercal \Sigma \phi_x) \\
&\qquad \mathcal{N}\left(f_x; \phi_x(\Sigma^{-1} + \sigma^{-2}\phi_X\phi_X^\intercal)^{-1}\left(\Sigma^{-1}\mu + \sigma^{-2}\phi_X \boldsymbol{y}\right),\right. \\
&\qquad \left.\phi_x(\Sigma^{-1} + \sigma^{-2}\phi_X\phi_X^\intercal)^{-1}\phi_x^\intercal\right)
\end{aligned}
$$

Code

our work from last lecture pays off

$$p(w \mid y, \phi_X) = \mathcal{N}\left(w; \mu + \Sigma\phi_X(\phi_X^\intercal \Sigma \phi_X + \sigma^2 I)^{-1}(y - \phi_X^\intercal \mu), \Sigma - \Sigma\phi_X(\phi_X^\intercal \Sigma \phi_X + \sigma^2 I)^{-1}\phi_X^\intercal \Sigma\right)$$

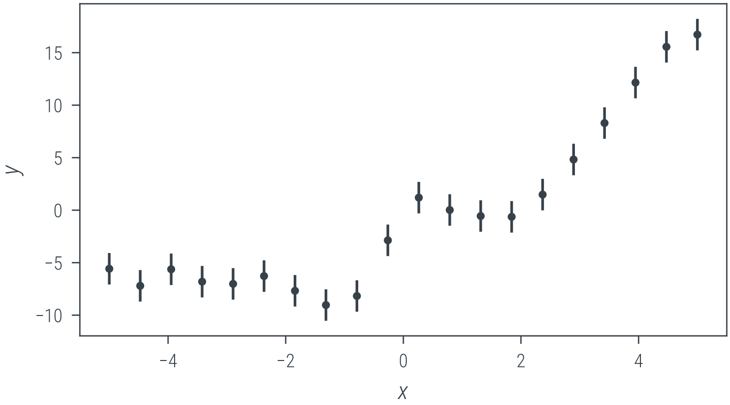
```python
from gaussians import Gaussian
from jax import numpy as jnp

# define prior in weight space
prior = Gaussian(mu=jnp.zeros(2), Sigma=jnp.eye(2))
# map into function space
phi = lambda x: jnp.hstack([jnp.ones_like(x), x])
x = jnp.linspace(-5, 5, 100)[:, None]
f_prior = phi(x) @ prior

# load data
import scipy.io
lin_data = scipy.io.loadmat("lindata.mat")
X = lin_data["X"]  # inputs
Y = lin_data["Y"][:, 0]  # outputs
sigma = lin_data["sigma"][0].flatten() # noise

# condition on data to get the posterior: p(w|X,Y) = N(Y|phi(X) @ w, sigma**2 I) * p(w) / p(Y|X)
posterior = prior.condition(phi(X), Y, sigma**2 * jnp.eye(len(X)))
```

DEMO

- `git clone https://github.com/philipphennig/ProbML_Apps.git`
- `cd ProbML_Apps/07`
- `pip install -r requirements.txt`
- `streamlit run Lecture_07.py`

A more Realistic Dataset

General linear regression



$$f(x) = w_1 + w_2 x = \phi_x^\intercal w$$

$$\phi_x := \begin{bmatrix} 1 \\ x \end{bmatrix}$$

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Summary:

- Gaussian distributions can be used to learn functions
- Analytical inference is possible using general linear models

$$f(x) = \phi(x)^\intercal w = \phi_x^\intercal w$$

- Then the posterior on both w and f is Gaussian
- The choice of features $\phi : \mathbb{X} \rightharpoonup \mathbb{R}$ is essentially unconstrained

Please cite this course, as

```
@techreport{Tuebingen_ProbML23 ,
    title =
    {Probabilistic Machine Learning},
    author = {Hennig, Philipp},
    series = {Lecture Notes
        in Machine Learning},
    year = {2023},
    institution = {Tübingen AI Center}}
```