Probabilistic Machine Learning

Exercise Sheet #4

due on Tuesday, 18 May 2021, 10am sharp

1. **EXAMple: Gaussian inference**

   Consider the Gaussian random variable $\boldsymbol{w} \in \mathbb{R}^F$ with probability density function

   $$p(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{w}; \boldsymbol{\mu}, \Sigma) \qquad \text{with} \qquad \boldsymbol{\mu} \in \mathbb{R}^F \text{ and symmetric positive definite } \Sigma \in \mathbb{R}^{F\times F}.$$

   You have access to data $\boldsymbol{y} \in \mathbb{R}^N$ assumed to be generated from $\boldsymbol{w}$ through a linear map $\Phi \in \mathbb{R}^{F\times N}$ according to the likelihood

   $$p(\boldsymbol{y} \mid \boldsymbol{w}) = \mathcal{N}(\boldsymbol{y}; \Phi^\intercal \boldsymbol{w}, \Lambda),$$

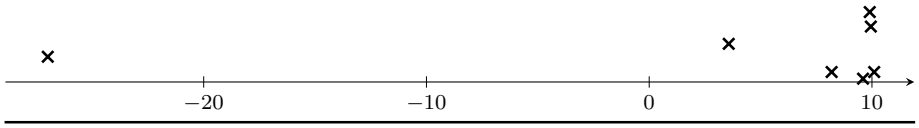
   where $\Lambda \in \mathbb{R}^{N\times N}$ symmetric positive definite. What is

   (a) the pdf of the *marginal* $p(\boldsymbol{y}) = \int p(\boldsymbol{y} \mid \boldsymbol{w}) p(\boldsymbol{w})\, d\boldsymbol{w}$

   (b) the pdf of the *posterior* $p(\boldsymbol{w} \mid \boldsymbol{y})$?

2. **Theory Question: The Seven Scientists**

   The terribly important quantity μ has been experimentally measured by seven scientists (A, B, C, D, E, F, G) with wildly differing experimental skills. They have reported the following measurements:



Scientist	A	B	C	D	E	F	G
x_i	-27.020	3.570	8.191	9.898	9.603	9.945	10.056

   We assume that they have all, independently of each other, made an unbiased Gaussian measurement of μ: $p(\boldsymbol{x} \mid \mu, \boldsymbol{\sigma}) = \prod_{i=1}^7 \mathcal{N}(x_i; \mu, \sigma_i^2)$. But we have to assume that their measurement errors σ_i vary a lot (some are skilled experimentalists, others are klutzes).

   (a) Write down the log-likelihood $\log(p(\boldsymbol{x} \mid \mu, \boldsymbol{\sigma}))$. Can you find points $(\mu, \boldsymbol{\sigma})$ that maximize this function? **Hint**: You don't have to compute $\nabla \log(p(\boldsymbol{x} \mid \mu, \boldsymbol{\sigma}))$ for this.

   You probably agree that, intuitively, it looks pretty certain that A and B are both inept measurers, that D–G are better, and that the true value of μ is somewhere close to 10. Are your findings consistent with this intuition?

   (b) Provide a Bayesian answer: Let the prior on each σ_i^{-2} be a broad Gamma[1] distribution $p(\boldsymbol{\sigma}) = \prod_{i=1}^7 \mathcal{G}(\sigma_i^{-2}; \alpha, \beta)$, e.g. with $\alpha = 1$, $\beta = 0.1$ and let the prior for μ be a broad Gaussian with mean 0 and standard deviation 10^3. Find the posterior for μ.

   Hint: First find the posterior for $\boldsymbol{\sigma}$ given μ and $\boldsymbol{x}$, $p(\boldsymbol{\sigma} \mid \boldsymbol{x}, \mu)$. Note that the normalization constant of this posterior is $p(\boldsymbol{x} \mid \mu)$. Marginalize over $\boldsymbol{\sigma}$ to find this normalizing constant. Note that this is a known integral. Then use Bayes' theorem a second time to find $p(\mu \mid \boldsymbol{x})$ up to normalization. Do not expect to find a *particularly* pretty algebraic form.

   (c) Plot the above posterior for μ, both for the data given above and for $\{\boldsymbol{x}\} = \{13.01, 7.39\}$.

3. **Practical Question** This week's practical task is to solve the above *Seven Scientists* question numerically, using Gibbs sampling. More on Ilias.

[1] The Gamma distribution is given by $\mathcal{G}(x; \alpha, \beta) = \frac{\beta^\alpha}{\Gamma(\alpha)} \boldsymbol{x}^{\alpha-1} e^{-\beta x}$.

1/

a) (Theorem Lec. 6, Slide 14 (missing transpose there))

$$p(y) = \mathcal{N}(y \mid \phi^T \mu, \Lambda + \phi^T \Sigma \phi)$$

b)

$$p(w \mid y) = \mathcal{N}\left(w \mid \mu + \Sigma\phi(\Lambda + \phi^T\Sigma\phi)^{-1}(y - \phi^T\mu),\ \Sigma - \Sigma\phi(\phi^T\Sigma\phi + \Lambda)^{-1}\phi^T\Sigma\right)$$

2/

a) $\log p(\{x_i\}_i \mid \mu, \{\sigma_i\}_i)$

$$= \sum_i \log p(x_i \mid \mu, \sigma_i) \qquad \text{(iid data)}$$

$$= \sum_i \left[-\frac{1}{2}\log(2\pi\sigma_i) - \frac{1}{2}\frac{(x_i-\mu)^2}{\sigma_i^2}\right]$$

$$= -\frac{1}{2}\sum_{i=1}^{7}\left[\log(2\pi\sigma_i) + \frac{(x_i-\mu)^2}{\sigma_i^2}\right]$$

One easy way to maximize this expression is by setting μ and $\{\sigma_i\}$ accordingly, is to assume one scientist is perfectly accurate (and the others aren't). This scientist, represented by measurement x_i, measures perfectly, thus $\mu = x_i$ and $\sigma_i = \varepsilon \to 0$. We cannot set the variance to zero, but we can still achieve any arbitrarily large likelihood this way. Assume all other scientists have some finite variance, e.g. $\sigma_{-i} = 1$. Then the log-likelihood becomes:

$$\underbrace{-\frac{1}{2}\sum_{\substack{j=1\\ j\neq i}}^{7}\log(2\pi) + (x_j-\mu)^2}_{\text{finite and known } (\mu = x_i)} \underbrace{-\frac{1}{2}\log(2\pi\varepsilon)}_{\to\infty \text{ as } \varepsilon\to 0.} \underbrace{-\frac{1}{2}\frac{(x_i-\mu)^2}{\varepsilon}}_{=0,\ \text{because } x_i = \mu} \xrightarrow{\varepsilon\to 0} \infty$$

These findings are not consistent with initial thoughts. Here, any scientist (even A) might be the perfect one.

b)

$$p(\sigma \mid x, \mu) = \frac{p(x \mid \sigma, \mu)\, \overbrace{p(\sigma \mid \mu)}^{=p(\sigma)}}{p(x \mid \mu)} \qquad (N := 7)$$

$$p(x \mid \mu) = \prod_{i=1}^{N} p(x_i \mid \mu)$$

$$= \prod_{i=1}^{N} \int_0^\infty p(x_i \mid \mu, \sigma_i^{-2})\, p(\sigma_i^{-2})\, d\sigma_i^{-2}$$

$$= \prod_{i=1}^{N} \int_0^\infty \frac{1}{\sqrt{2\pi}} (\sigma_i^{-2})^{\frac{1}{2}} e^{-\frac{1}{2}\sigma_i^{-2}(x_i-\mu)^2} \frac{\beta^\alpha}{\Gamma(\alpha)} \underbrace{(\sigma_i^{-2})^{\alpha-1}}_{=1\ (\alpha=1)} e^{-\beta\sigma_i^{-2}}\, d\sigma_i^{-2}$$

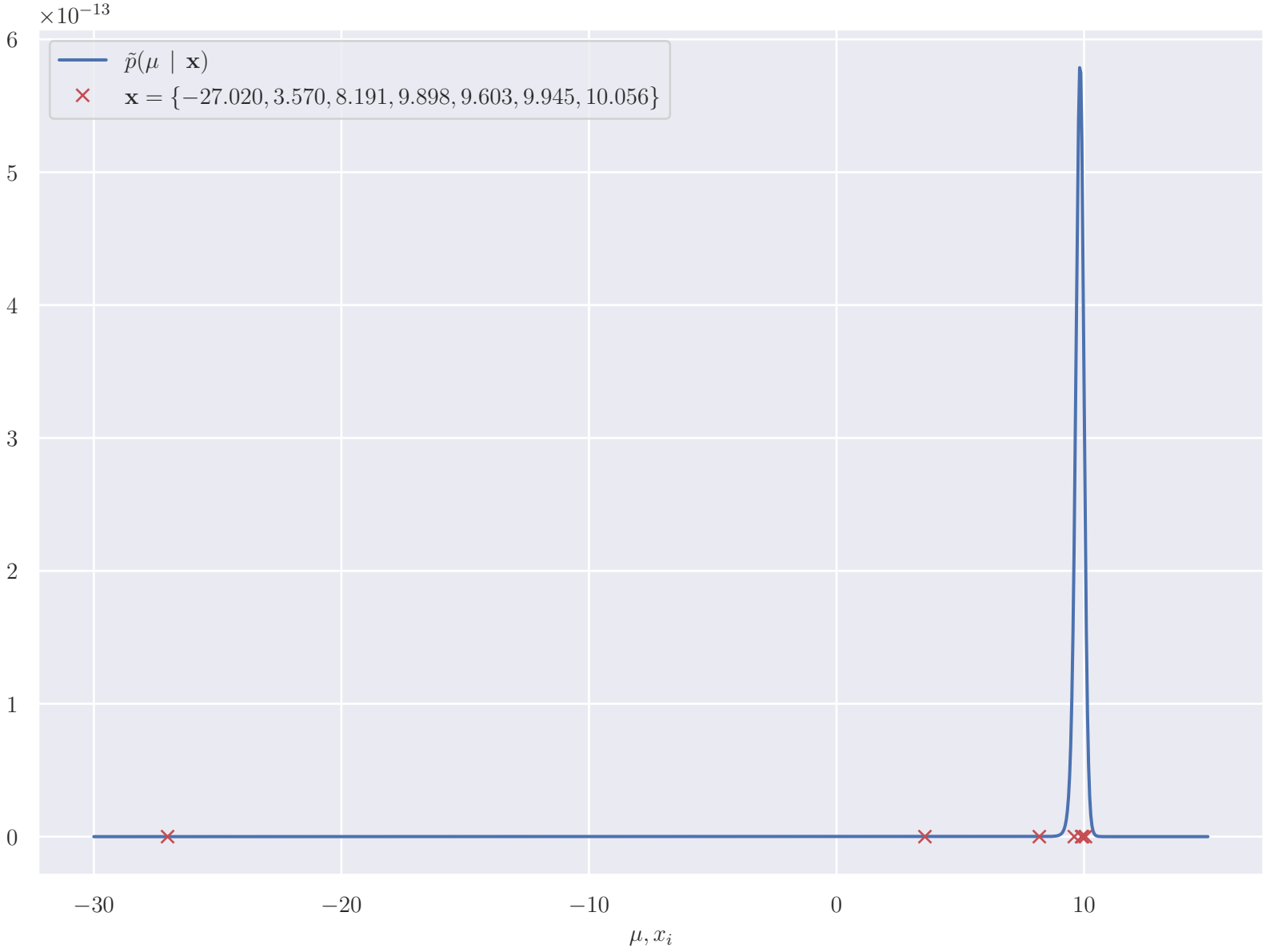
$$= \left(\frac{\beta}{\underbrace{\Gamma(1)}_{1}} \cdot \frac{1}{\sqrt{2\pi}}\right)^N \prod_{i=1}^{N} \underbrace{\int_0^\infty (\sigma_i^{-2})^{\frac{1}{2}} \exp\left(-\sigma_i^{-2}\left(\tfrac{1}{2}(x_i-\mu)^2+\beta\right)\right) d\sigma_i^{-2}}_{\text{has the form } \int_0^\infty \sqrt{y}\, e^{-ay}\, dy = \frac{\sqrt{\pi}}{2a^{3/2}} \text{ (looked up.)}}$$

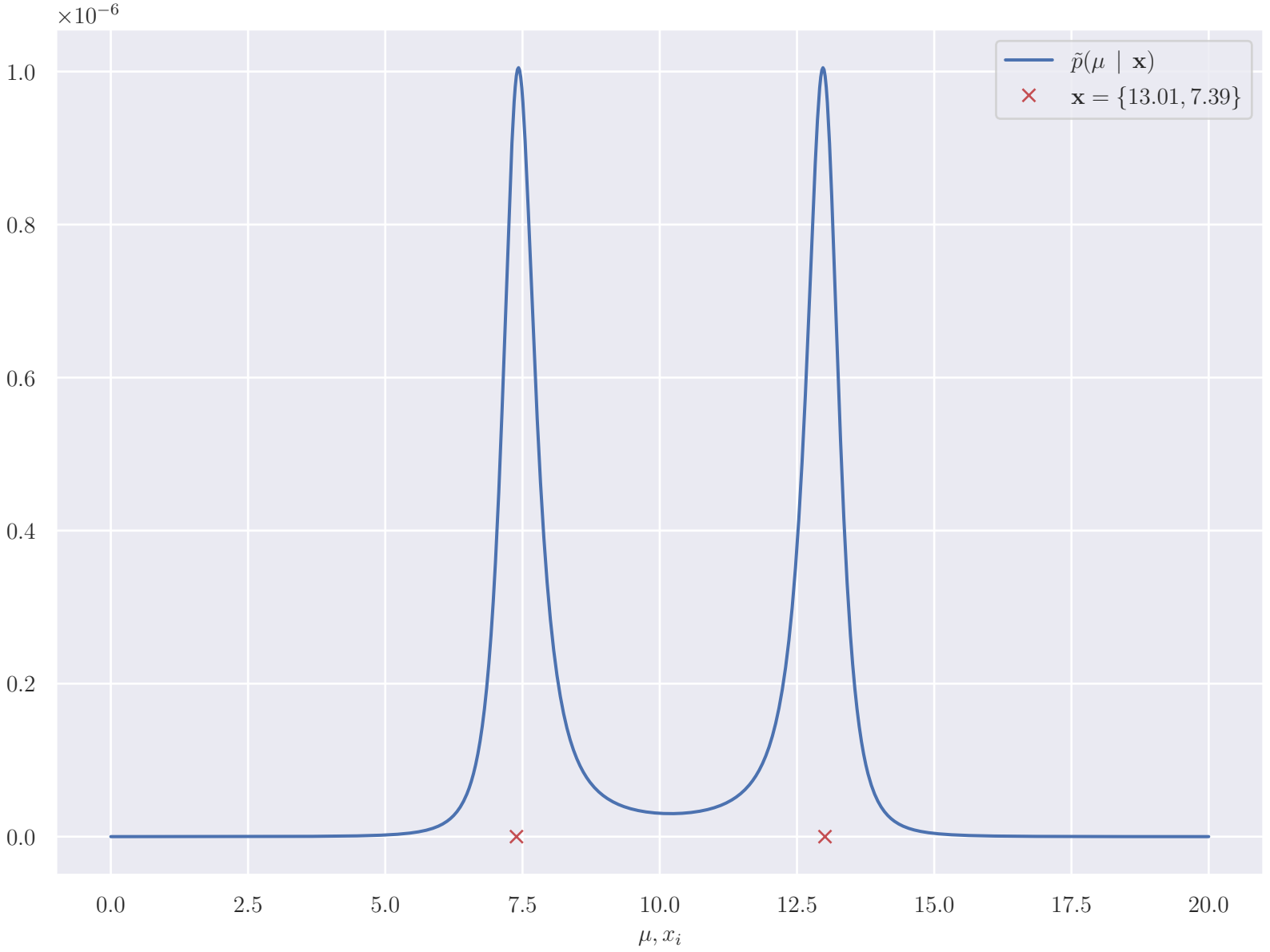
$$= \left(\frac{0.1}{\sqrt{2\pi}}\right)^N \prod_{i=1}^{N} \frac{\sqrt{\pi}}{2\left(\frac{1}{2}(x_i-\mu)^2+0.1\right)^{3/2}}$$

$$= \left(\frac{0.1}{2\sqrt{2}}\right)^N \prod_{i=1}^{N} \left(\tfrac{1}{2}(x_i-\mu)^2+0.1\right)^{-\frac{3}{2}}$$

$$p(\mu \mid x) \propto p(x \mid \mu)\, p(\mu)$$

$$= \left(\frac{0.1}{2\sqrt{2}}\right)^N \prod_{i=1}^{N} \left(\tfrac{1}{2}(x_i-\mu)^2+0.1\right)^{-\frac{3}{2}} \frac{1}{\sqrt{2\pi}\cdot 10^3} \exp\left(-\frac{1}{2}\frac{\mu^2}{10^6}\right) //$$


$\times 10^{-13}$
6
5
4
3
2
1
0
$\tilde{p}(\mu \mid \mathbf{x})$
$\mathbf{x} = \{-27.020, 3.570, 8.191, 9.898, 9.603, 9.945, 10.056\}$
−30
−20
−10
0
10
μ, x_i


$\times 10^{-6}$
1.0
0.8
0.6
0.4
0.2
0.0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
μ, x_i
$\tilde{p}(\mu \mid \mathbf{x})$
$\mathbf{x} = \{13.01, 7.39\}$

ExerciseSheet_04

May 15, 2021

1 Probabilistic Machine Learning

University of Tübingen, Summer Term 2021

1.1 Exercise Sheet 4

This sheet is **due on Tuesday 18 May 2021 at 10am sharp.**

1.2 Gibbs Sampling for the *Seven Scientists* Problem

This week we deal with the *Seven Scientists* problem. You can find the description of this problem in this week's theory exercise. For this particular problem it is actually possible to compute the posterior distribution over μ given x analytically (at least up to normalization). In practice, however, this is rarely the case. One way out is to use sampling methods that sample from the posterior. We can then use these samples for moment approximations, for example. In this tutorial, we are going to implement one member of these sampling methods: Gibbs sampling.

Ideally, we would like to sample directly from the posterior $p(\mu, \sigma|x)$, which is often not possible. However, we can easily sample from the conditional distributions $p(\mu|\sigma, x)$ and $p(\sigma|\mu, x)$. The idea for Gibbs sampling is the following: 1. Set the initial values $\mu^{(0)}$ and $\sigma^{(0)}$ by sampling from the priors $p(\mu)$ and $p(\sigma)$, respectively. 2. Sample alternately from the conditionals, i.e. sample 2.1. $\mu^{(1)}$ from $p(\mu|\sigma^{(0)}, x)$ 2.2. $\sigma^{(1)}$ from $p(\sigma|\mu^{(1)}, x)$ 2.3. $\mu^{(2)}$ from $p(\mu|\sigma^{(1)}, x)$ 2.4. ...

For a visualization on how the Gibbs sampling works, see e.g. https://chi-feng.github.io/mcmc-demo/app.html#GibbsSampling.

[2]:
```
import numpy as np
from matplotlib import pyplot as plt
from scipy.stats import norm, gamma
import seaborn as sns
```

[37]:
```
sns.set()
plt.rcParams.update({
    'figure.constrained_layout.use': True,
    'figure.figsize': ((W := 5), W / (4/3)),
    'figure.dpi': 150,
    'font.size' : 11,
```

```
    'axes.labelsize': 11,
    'legend.fontsize': 11,
    'font.family': 'lmodern',
    'text.usetex': True,
    'text.latex.preamble': (
        r'\usepackage{lmodern}'
        r'\usepackage{siunitx}'
        r'\usepackage{physics}'
    )
})
%config InlineBackend.figure_format = 'retina'
```

1.3 Priors for μ and σ_i

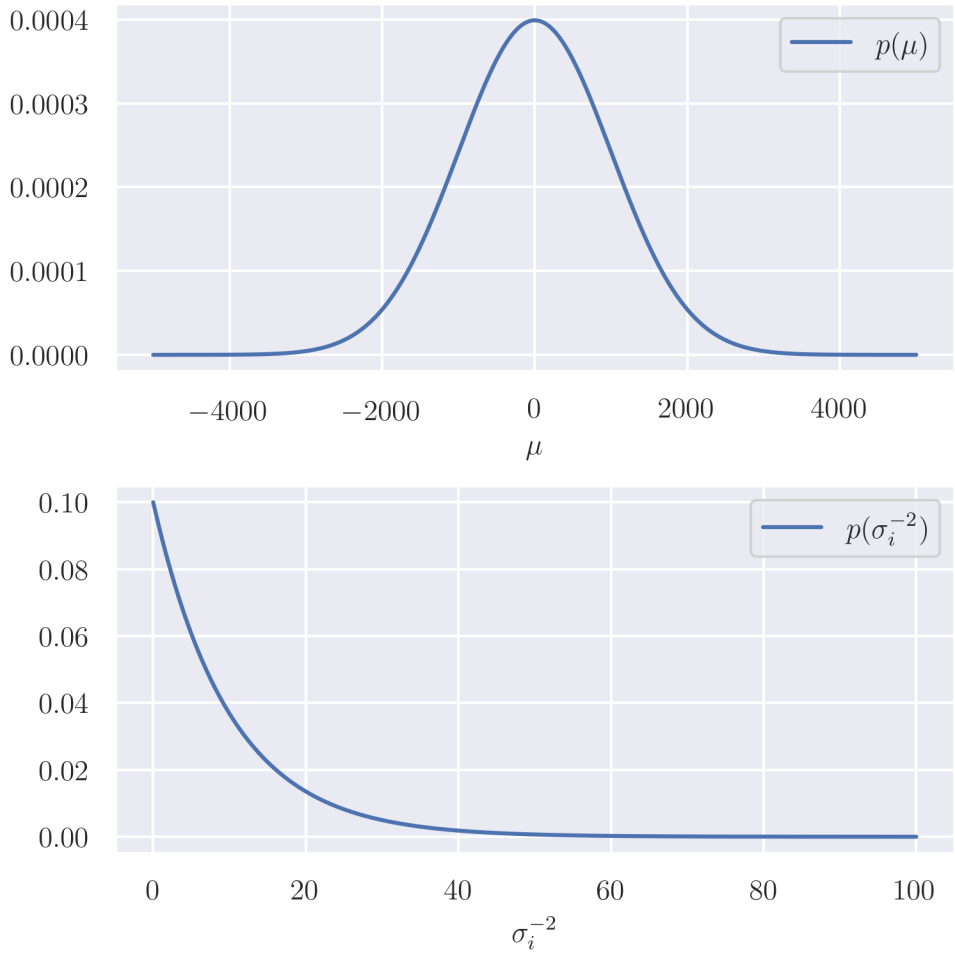
First of all, we implement the prior distributions for μ and σ_i, since we will need them to initialize the sampler. - $p(\mu)$: The prior over μ is a Gaussian $\mathcal{N}(\mu; m, v^2)$ with mean $m = 0$ and standard deviation $v = 10^3$. - $p(\sigma_i)$: Actually, we only know the distribution of σ_i^{-2}, which is a Gamma distribution $\mathcal{G}(\sigma_i^{-2}; \alpha, \beta)$ with $\alpha = 1$ and $\beta = 0.1$. Note: That means, if we sample from this Gamma distribution, we actually obtain samples for σ_i^{-2} (not σ_i).

Task: Plot the pdf of the priors over μ and σ_i^{-2}.

[53]:
```
mu = norm(loc=0, scale=1e3)
sigma_n2 = gamma(a=1, scale=1/0.1)

mu_ = np.linspace(-5e3, 5e3, 1000)
sigma_n2_ = np.linspace(0, 100, 1000)
fig, ax = plt.subplots(2, 1, figsize=(W, W / (1/1)))
ax[0].plot(mu_, mu.pdf(mu_), label=r'$p(\mu)$')
ax[0].set_xlabel(r'$\mu$')
ax[0].legend()
ax[1].plot(sigma_n2_, sigma_n2.pdf(sigma_n2_), label=r'$p(\sigma_i^{-2})$')
ax[1].set_xlabel(r'$\sigma_i^{-2}$')
ax[1].legend()
```

[53]:
```
<matplotlib.legend.Legend at 0x7fb5f1461ee0>
```



1.4 Gibbs Sampling

For running the sampler, we still need the conditionals $p(\mu|\sigma, x)$ and $p(\sigma|\mu, x)$.

- $p(\mu|\sigma, x)$: We need to be able to sample μ given the measurements x and the measurement errors σ_i, $i \in \{1, ..., 7\}$. Using Bayes' Theorem, it can be shown that $p(\mu|\sigma, x) = \mathcal{N}(\mu; m_*, v_*^2)$ with $v_*^2 = (\frac{1}{v^2} + \sum_{i=1}^{7} \frac{1}{\sigma_i^2})^{-1}$ and $m_* = v_*^2(\frac{m}{v^2} + \sum_{i=1}^{7} \frac{x_i}{\sigma_i^2})$.
- $p(\sigma|\mu, x)$: Similarly, we can derive a distribution for σ_i^{-2} by applying Bayes' Theorem. It holds $p(\sigma_i^{-2}|\mu, x_i) = \mathcal{G}(\sigma_i^{-2}; \alpha_*, \beta_*^2)$ with $\alpha_* = \alpha + \frac{1}{2}$ and $\beta = \beta + \frac{1}{2}(x_i - \mu)^2$. Note that by sampling from this Gamma distribution, we obtain samples for σ_i^{-2} (not σ_i).

Task: Implement two functions `sample_mu` and `sample_sigma_i` to draw samples for μ and σ_i.

[94]:
```
def sample_mu(sigma, x):
    return norm(scale=np.sqrt(v2 := 1/(1/mu.kwds['scale']**2 + np.sum(1 /␣
↪sigma**2))),
                loc=(v2 * (mu.kwds['loc'] / mu.kwds['scale']**2 + np.sum(x /␣
↪sigma**2)))).rvs()
```

```
def sample_sigma_i(mu, x_i):
    return 1 / np.sqrt(gamma(a=(sigma_n2.kwds['a'] + 1/2),
                             scale=1/(1/sigma_n2.kwds['scale'] + 1/2*(x_i -␣
 ↪mu)**2)).rvs())
```

Now, we can actually implement the sampler.

Task: Implement the Gibbs sampler. After creating the samples, throw away the first `BURNIN` samples to mitigate the impact of the initialization.

[99]:
```
# Data
x = np.array([-27.020, 3.570, 8.191, 9.898, 9.603, 9.945, 10.056])
N = len(x)

# Number of Gibbs samples
NUM_SAMPLES = 10000

# Create matrices for storing the samples
mu_samples    = np.zeros(NUM_SAMPLES)
sigma_samples = np.zeros([NUM_SAMPLES, N])

# Initialize sampler by sampling from priors
mu_samples[0] = mu.rvs()
sigma_samples[0] = 1 / np.sqrt(sigma_n2.rvs(N))

# Run Gibbs sampler
for i in range(1, NUM_SAMPLES):
    mu_samples[i] = sample_mu(sigma_samples[i - 1], x)
    for j in range(N):
        sigma_samples[i, j] = sample_sigma_i(mu_samples[i], x[j])
```

[100]:
```
BURNIN = 20

# Drop the first BURNIN samples
mu_samples = mu_samples[BURNIN:]
sigma_samples = sigma_samples[BURNIN:]
```

1.5 Analysis of the Gibbs Samples

First, let's analyse the samples visually.

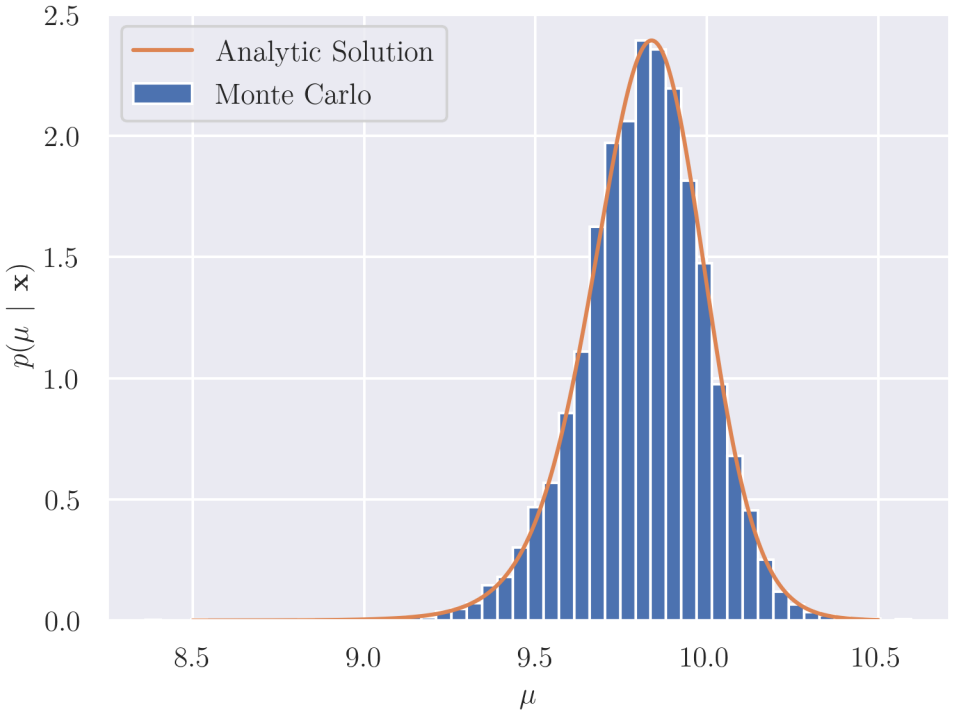
Task: Create a histogram for the samples of μ. If you solved the theory exercise, you can also plot the analytical posterior $p(\mu|x)$ into the figure, to check, whether the two "distributions" match.

[157]:
```
mu_ = np.linspace(8.5, 10.5, 1000)[:, np.newaxis]
```

```
p = ((0.1 / 2*np.sqrt(2))**N * np.prod(1/2 * (x - mu_)**2 + 0.1, 1)**(-3/2)
 ↪# Likelihood
     * 1/(np.sqrt(2*np.pi) * 1e3) * np.exp(-1/2 * mu_.flatten()**2 / 1e6))
 ↪# Prior

fig, ax = plt.subplots()
ax.hist(mu_samples, 50, density=True, label='Monte Carlo')
ax.plot(mu_, p / np.trapz(p, mu_.flatten()), label='Analytic Solution')
ax.set_xlabel(r'$\mu$')
ax.set_ylabel(r'$p(\mu \,\mid\, \vb x)$')
ax.legend()
```

[157]: <matplotlib.legend.Legend at 0x7fb5de153550>

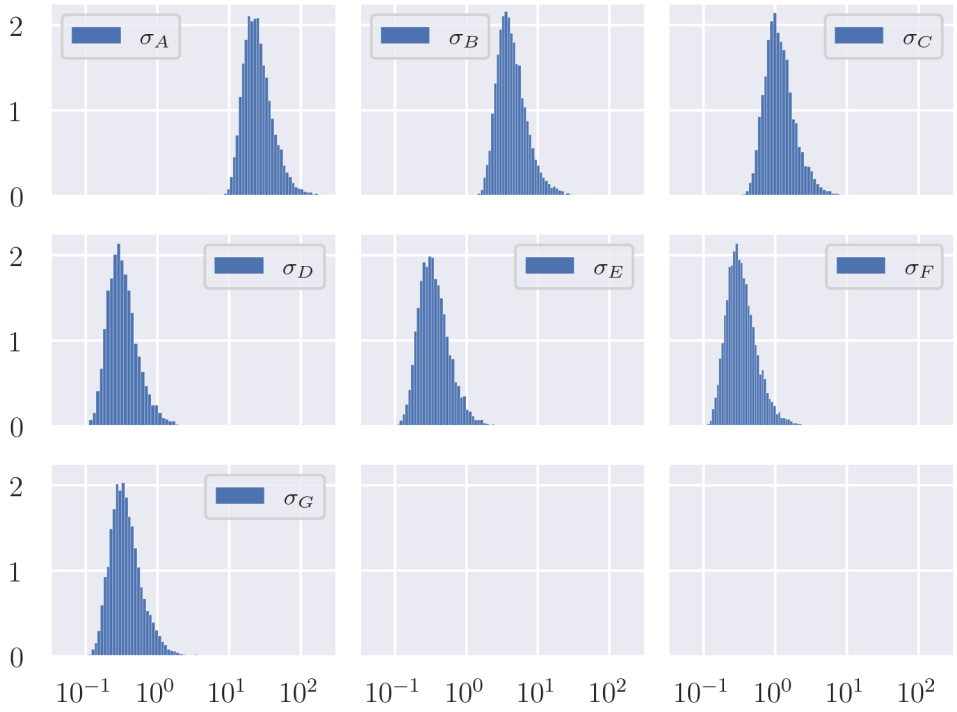


Task: Also create histograms for the samples of σ_i. Plot all these histograms into a single figure in a *meaningful* way (it could be a good idea to use a logarithmic x-scale).

[158]:
```
# First visualization

fig, ax = plt.subplots(3, 3, sharex=True, sharey=True)
s = 'ABCDEFG'
for i, a in enumerate(ax.flatten()[:N]):
```

```
    a.hist(np.log10(sigma_samples[:, i]), 50, density=True, linewidth=.1,␣
↪label=fr'$\sigma_{s[i]}$')
    a.set_xbound(-1.5, 2.5)
    a.set_xticks([-1, 0, 1, 2])
    a.set_xticklabels([fr'$10^{{{x}}}$' for x in [-1, 0, 1, 2]])
    a.legend(fontsize='small')
```

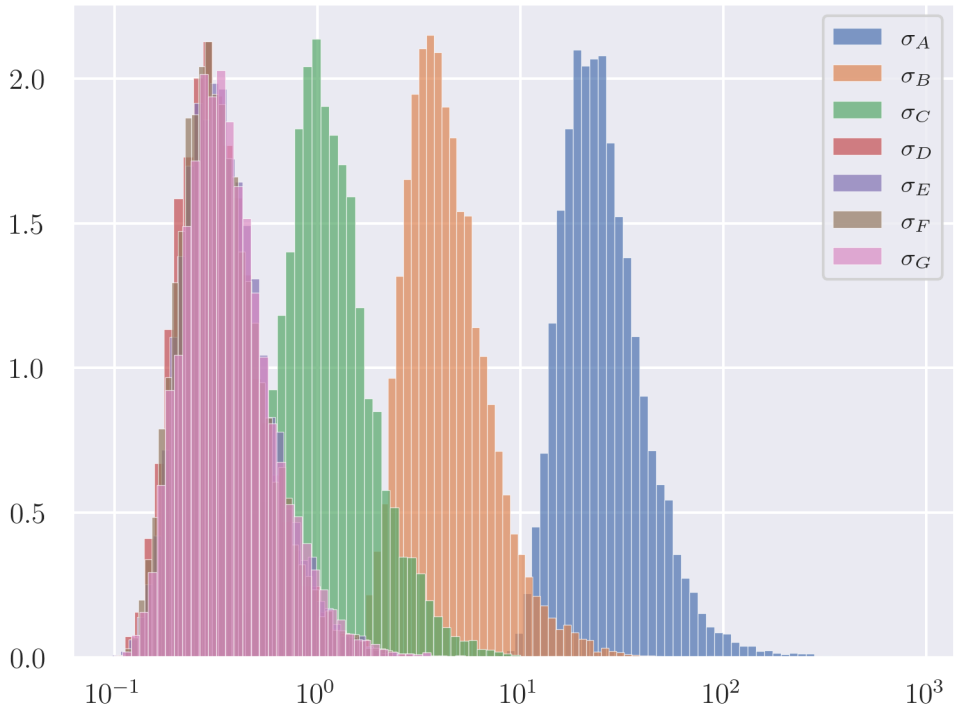


[180]:
```
# Second visualization

fig, ax = plt.subplots()
s = 'ABCDEFG'
for i in range(N):
    ax.hist(np.log10(sigma_samples[:, i]), 50, density=True, linewidth=.3,␣
 ↪alpha=.7, label=fr'$\sigma_{s[i]}$')

ax.set_xticks([-1, 0, 1, 2, 3])
ax.set_xticklabels([fr'$10^{{{x}}}$' for x in [-1, 0, 1, 2, 3]])
ax.legend(fontsize='small')
```

[180]:
```
<matplotlib.legend.Legend at 0x7fb5e3d68b80>
```



We can also use the Gibbs samples for estimating moments of the respective distributions.

Task: Give the Monte Carlo estimates for the expected value of μ (under $p(\mu|x)$) and the expected value of σ_i (under $p(\sigma_i|x)$).

[171]:
```
E_mu = mu_samples.mean()
E_sigma = sigma_samples.mean(0)
print("Expected mu: {:.3f}\nExpected sigmas: {}".format(E_mu, ", ".join([f'{S}:␣
 ↪{v:.3f}' for S, v in zip(s, E_sigma)])))
```

```
Expected mu: 9.814
Expected sigmas: A: 29.498, B: 4.993, C: 1.344, D: 0.385, E: 0.415, F: 0.394, G:
0.427
```