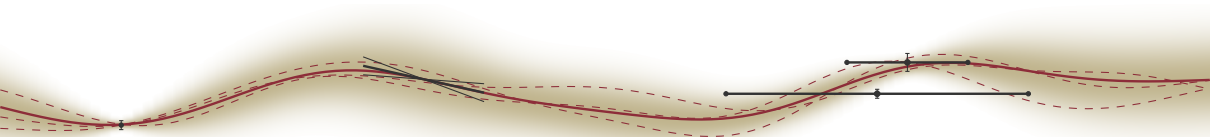
Probabilistic Inference and Learning
Lecture 05
More on Gaussian Regression

Philipp Hennig

31 October 2018

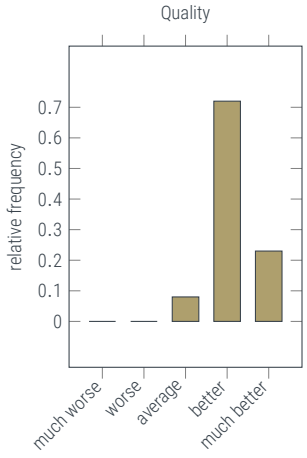
EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

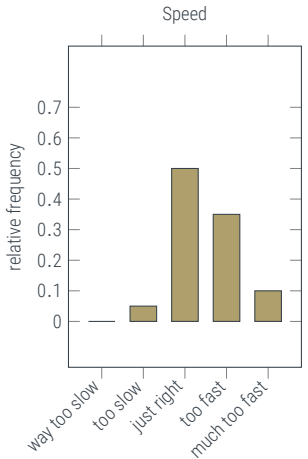
Faculty of Science
Department of Computer Science
Chair for the Methods of Machine Learning

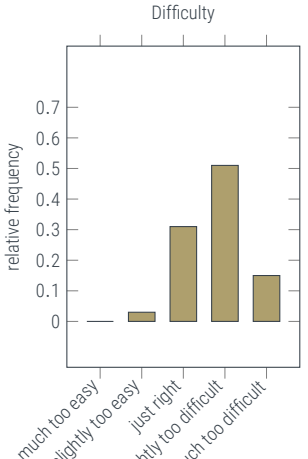


Last Lecture: Debrief

Feedback dashboard







Last Lecture: Debrief

Detailed Feedback

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Things you did not like:

- long terms on the blackboard
- I have no python experience
- "I feel a lot more stupid than every physicist now"
- black, red and gold are hard to distinguish on the slides

Things you did not understand:

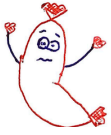
- weight vs. function space
- when do we need matrix inverses instead of Cholesky
- sampling
- could you please give a preview of the coming lecture at the end?
- why adding 10^{-6} to the diagonal?

Things you enjoyed:

- plots / animations (!!!)
- code(!!)
- pointing out the uncertainty comes for free
- sampling from Gaussians

Which part of today's lecture did you enjoy **most**?

the sausage of uncertainty



Overview of Lectures so far:

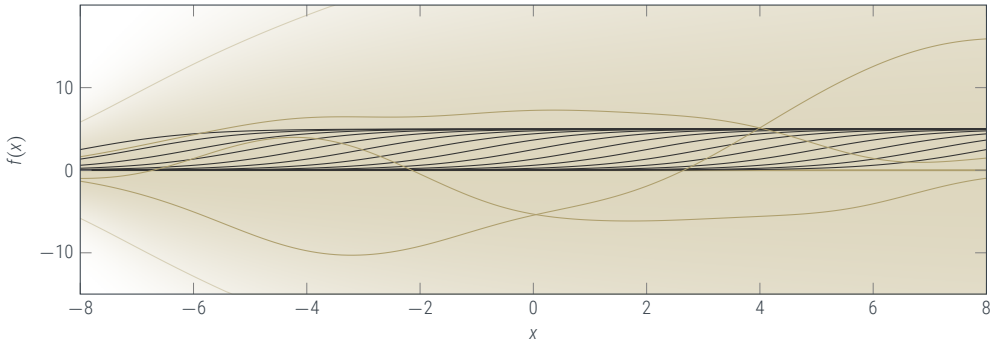
0. Introduction to Reasoning under Uncertainty
   - Probabilities are the mathematical formalization of uncertainty
1. Probabilistic Reasoning
   - Probabilities extend deductive to plausible reasoning. Conditional independence affects complexity
2. Probabilities over Continuous Variables
   - Probability **densities** distribute probability over continuous domains
3. Gaussian Probability Distributions
   - Gaussians map probabilistic inference to **linear algebra**
4. Gaussian Parametric Regression
   - Gaussian's can be used to infer **functions**, using linear models

Today:

- Can we *learn* the features?
- How do we do this in practice?
- Connections to deep learning

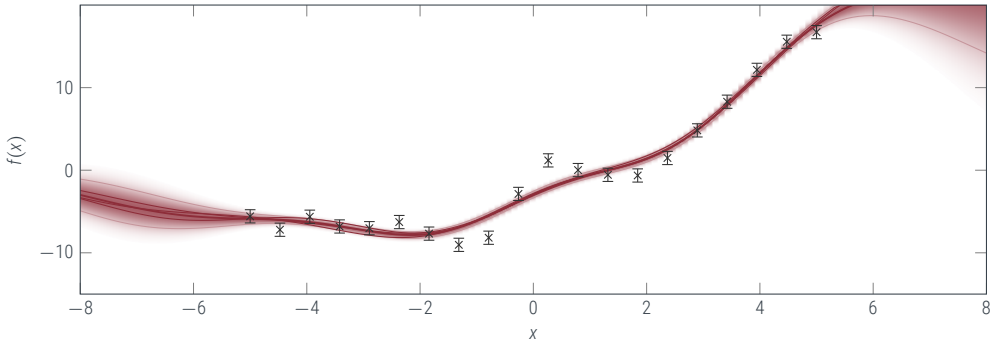
Reminder: General Linear Regression

An unbounded abundance of choices for features



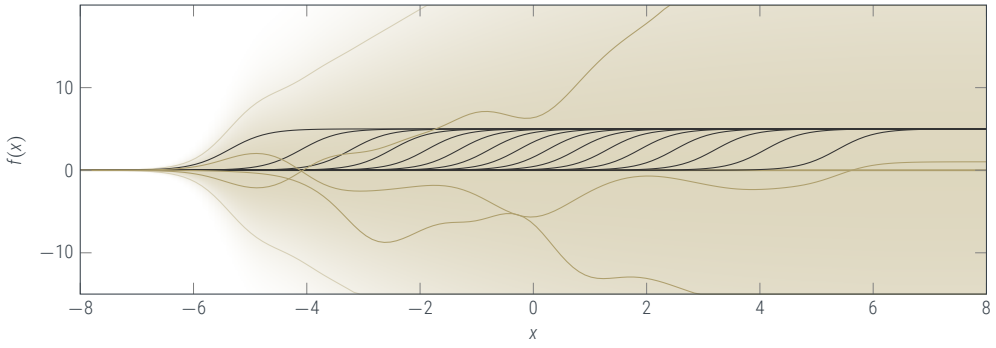
Reminder: General Linear Regression

An unbounded abundance of choices for features



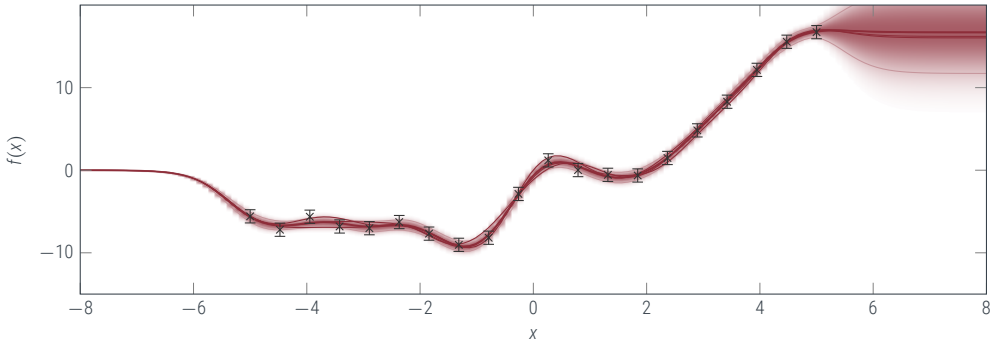
Reminder: General Linear Regression

An unbounded abundance of choices for features



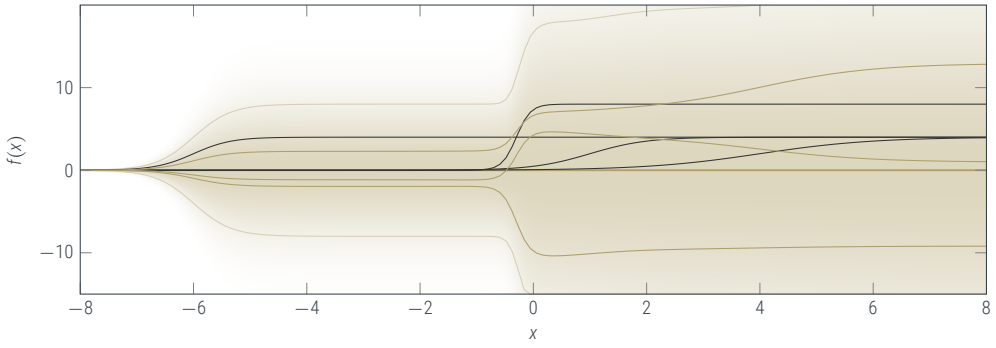
Reminder: General Linear Regression

An unbounded abundance of choices for features



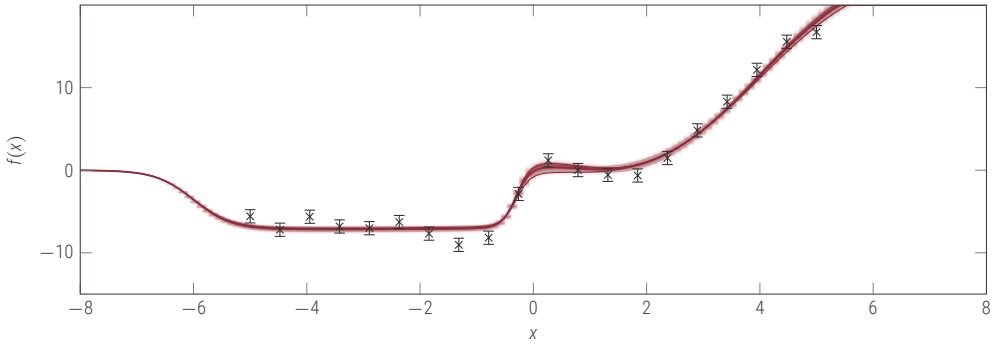
Reminder: General Linear Regression

An unbounded abundance of choices for features



Reminder: General Linear Regression

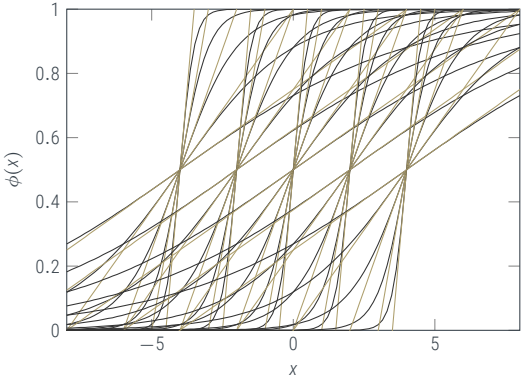
An unbounded abundance of choices for features



Can we **Learn** the Features?

Hierarchical Bayesian Inference

$$p(w \mid y, \phi) = \frac{p(y \mid w, \phi) p(w \mid \phi)}{p(y \mid \phi)}$$



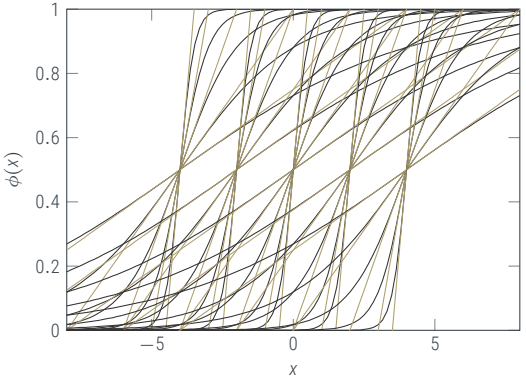
- There is an infinite-dimensional space of feature functions to choose from
- Maybe we can restrict to a finite-dimensional sub-space and **search** in there? Say

$$\phi_i(x; \boldsymbol{\theta}) = \frac{1}{1 + \exp(-\frac{x - \theta_1}{\theta_2})}$$

Can we **Learn** the Features?

Hierarchical Bayesian Inference

$$p(w \mid y, \phi) = \frac{p(y \mid w, \phi) p(w \mid \phi)}{p(y \mid \phi)}$$

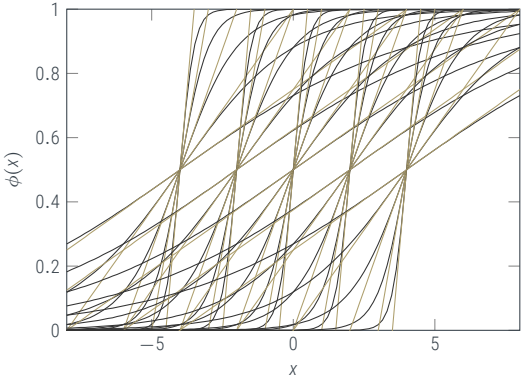


- There is an infinite-dimensional space of feature functions to choose from
- Maybe we can restrict to a finite-dimensional sub-space and **search** in there? Say

$$\phi_i(x; \boldsymbol{\theta}) = \frac{1}{1 + \exp(-\frac{x - \theta_1}{\theta_2})}$$

- θ_1, θ_2 are just unknown parameters!
- So can we infer them just like $\boldsymbol{w}$?

$$p(w \mid y, \phi) = \frac{p(y \mid w, \phi) p(w \mid \phi)}{p(y \mid \phi)}$$



- There is an infinite-dimensional space of feature functions to choose from
- Maybe we can restrict to a finite-dimensional sub-space and **search** in there? Say

$$\phi_i(x; \boldsymbol{\theta}) = \frac{1}{1 + \exp(-\frac{x - \theta_1}{\theta_2})}$$

- θ_1, θ_2 are just unknown parameters!
- So can we infer them just like $\boldsymbol{w}$?
- Yes, but not as easily: the likelihood

$$p(\boldsymbol{y} \mid w, \theta) = \mathcal{N}(\boldsymbol{y}; \phi(\boldsymbol{x}; \boldsymbol{\theta})^\intercal w, \sigma^2)$$

contains a **non-linear** map of $\boldsymbol{\theta}$.

$$p(f \mid \boldsymbol{y}, \boldsymbol{x}, \boldsymbol{\theta}) = \frac{p(\boldsymbol{y} \mid f, \boldsymbol{x}, \boldsymbol{\theta}) p(f \mid, \boldsymbol{\theta})}{\int p(\boldsymbol{y} \mid f, \boldsymbol{x}, \boldsymbol{\theta}) p(f \mid, \boldsymbol{\theta}) \, df} = \frac{p(\boldsymbol{y} \mid f, \boldsymbol{x}, \boldsymbol{\theta}) p(f \mid, \boldsymbol{\theta})}{p(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{\theta})}$$

- **Model parameters** like θ are also known as **hyper-parameters**.
- This is largely a computational, practical distinction:

data	are observed	→ **condition**
variables	are the things we care about	→ **full probabilistic treatment**
parameters	are the things we have to deal with to get the model right	→ **integrate out**
hyper-parameters	are the top-level, too expensive to properly infer	→ **fit**

The **model evidence** in Bayes' Theorem is the (marginal) **likelihood** for the model.

Hierarchical Bayesian Inference

Bayesian model adaptation

$$p(f \mid \boldsymbol{y}, \boldsymbol{x}, \boldsymbol{\theta}) = \frac{p(\boldsymbol{y} \mid f, \boldsymbol{x}, \boldsymbol{\theta}) p(f \mid, \boldsymbol{\theta})}{\int p(\boldsymbol{y} \mid f, \boldsymbol{x}, \boldsymbol{\theta}) p(f \mid, \boldsymbol{\theta}) \, df} = \frac{p(\boldsymbol{y} \mid f, \boldsymbol{x}, \boldsymbol{\theta}) p(f \mid, \boldsymbol{\theta})}{p(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{\theta})}$$

- For Gaussians, die evidence has **analytic form**:

$$\underbrace{\mathcal{N}(\boldsymbol{y}; \phi_X^{\boldsymbol{\theta}\mathsf{T}} w + \boldsymbol{b}, \Lambda)}_{p(\boldsymbol{y}|f,\boldsymbol{x},\boldsymbol{\theta})} \cdot \underbrace{\mathcal{N}(w, \mu, \Sigma)}_{p(f)} = \underbrace{\mathcal{N}(w; m_{\text{post}}^{\boldsymbol{\theta}}, V_{\text{post}}^{\boldsymbol{\theta}})}_{p(f|\boldsymbol{y},\boldsymbol{x},\boldsymbol{\theta})} \cdot \underbrace{\mathcal{N}(\boldsymbol{y}; \phi_X^{\boldsymbol{\theta}\mathsf{T}} \mu + b, \phi_X^{\boldsymbol{\theta}\mathsf{T}} \Sigma \phi_X^{\boldsymbol{\theta}} + \Lambda)}_{p(\boldsymbol{y}|\boldsymbol{\theta},\boldsymbol{x})}$$

- **BUT:** It's not a linear function of $\boldsymbol{\theta}$, so analytic Gaussian inference is not available!

Computational complexity is *the* principal challenge of probabilistic reasoning.

The Toolbox

Five principal methods for dealing with computational complexity in probabilistic inference

1. **Maximum Likelihood (ML) / Maximum A-Posteriori (MAP)** estimation:
   To estimate θ in $p(D \mid \theta)$ or $p(\theta \mid D)$, set $\hat{\theta} = \arg\max_\theta p$.
2. ????
3. ????
4. ????
5. ????

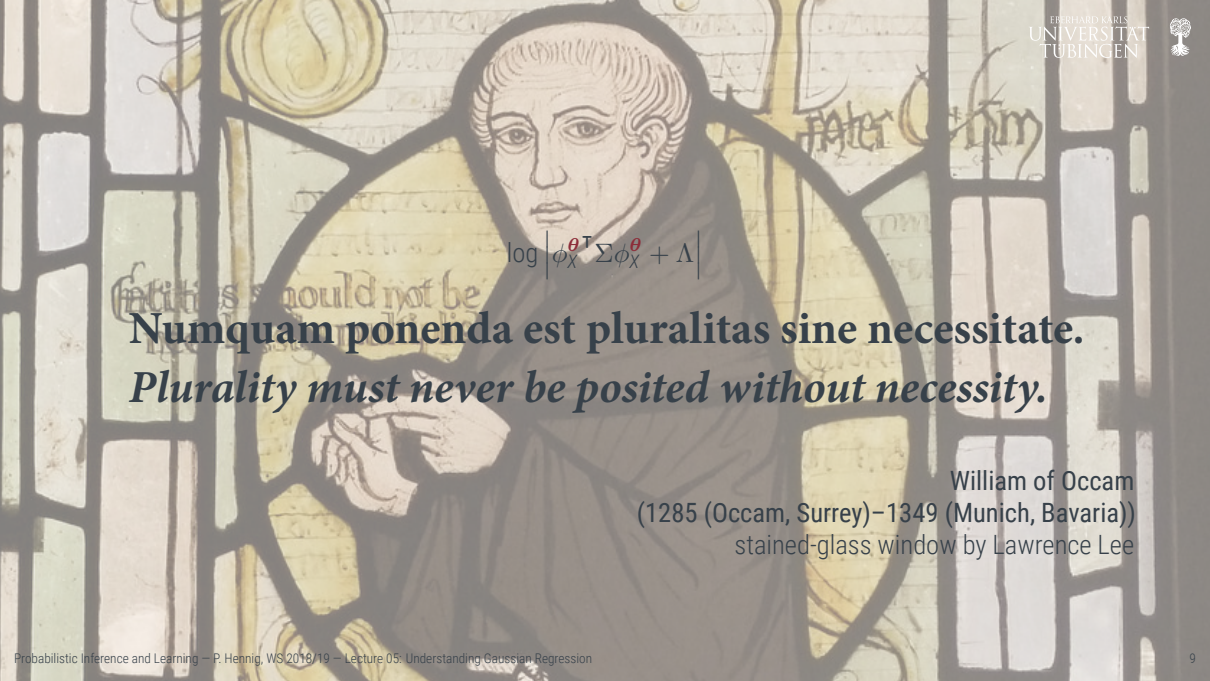
Disclaimer: The listed items are neither mutually exclusive nor collectively exhaustive. Some of the methods are intricately interrelated.

ML / MAP in Practice

Finding the "best fit" θ in Gaussian models

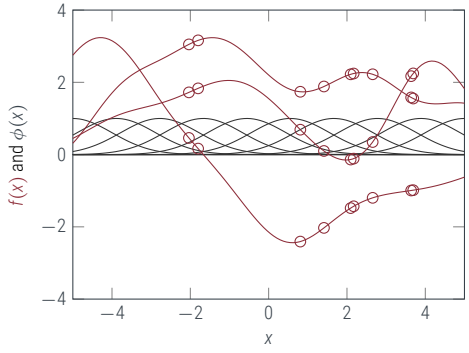
[e.g. DJC MacKay, *The evidence framework applied to classification networks*, 1992]

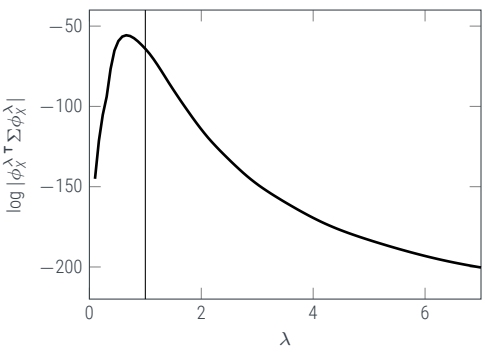
$$
\begin{aligned}
\hat{\boldsymbol{\theta}} &= \arg\max_{\boldsymbol{\theta}} \mathcal{N}(\boldsymbol{y}; \phi_X^{\boldsymbol{\theta}\intercal}\mu + b, \phi_X^{\boldsymbol{\theta}\intercal}\Sigma\phi_X^{\boldsymbol{\theta}} + \Lambda) \\
&= \arg\max_{\boldsymbol{\theta}} \log \mathcal{N}(\boldsymbol{y}; \phi_X^{\boldsymbol{\theta}\intercal}\mu + b, \phi_X^{\boldsymbol{\theta}\intercal}\Sigma\phi_X^{\boldsymbol{\theta}} + \Lambda) \\
&= \arg\min_{\boldsymbol{\theta}} -\log \mathcal{N}(\boldsymbol{y}; \phi_X^{\boldsymbol{\theta}\intercal}\mu + b, \phi_X^{\boldsymbol{\theta}\intercal}\Sigma\phi_X^{\boldsymbol{\theta}} + \Lambda) \\
&= \arg\min_{\boldsymbol{\theta}} \frac{1}{2}\left(\underbrace{(\boldsymbol{y} - \phi_X^{\boldsymbol{\theta}\intercal}\mu)^\intercal \left(\phi_X^{\boldsymbol{\theta}\intercal}\Sigma\phi_X^{\boldsymbol{\theta}} + \Lambda\right)^{-1} (\boldsymbol{y} - \phi_X^{\boldsymbol{\theta}\intercal}\mu)}_{\text{square error}} + \underbrace{\log\left|\phi_X^{\boldsymbol{\theta}\intercal}\Sigma\phi_X^{\boldsymbol{\theta}} + \Lambda\right|}_{\text{model complexity / Occam factor}} \right) + \frac{N}{2}\log 2\pi
\end{aligned}
$$


$\log \left| \phi_X^{\boldsymbol{\theta}^T} \Sigma \phi_X^{\boldsymbol{\theta}} + \Lambda \right|$
Numquam ponenda est pluralitas sine necessitate.
Plurality must never be posited without necessity.
William of Occam
(1285 (Occam, Surrey)–1349 (Munich, Bavaria))
stained-glass window by Lawrence Lee

What is Model Complexity?

The Occam factor is not always straightforward



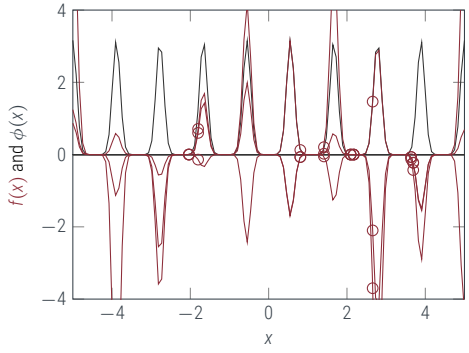


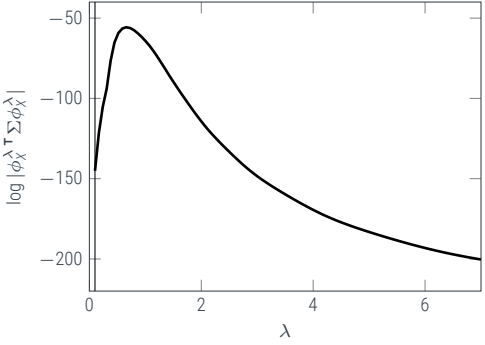
$$\log \left| \phi_X^{\boldsymbol{\theta}\,\mathsf{T}} \Sigma \phi_X^{\boldsymbol{\theta}} + \Lambda \right|$$

measures model complexity as the "volume" of hypotheses covered by the joint Gaussian distribution.

What is Model Complexity?

The Occam factor is not always straightforward



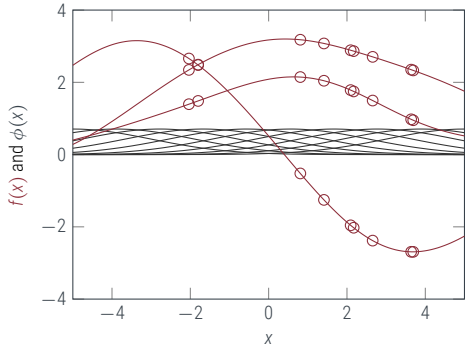


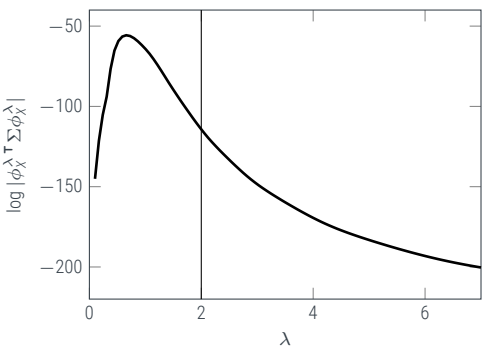
$$\log \left| \phi_X^{\boldsymbol{\theta}\mathsf{T}} \Sigma \phi_X^{\boldsymbol{\theta}} + \Lambda \right|$$

measures model complexity as the "volume" of hypotheses covered by the joint Gaussian distribution.

What is Model Complexity?

The Occam factor is not always straightforward



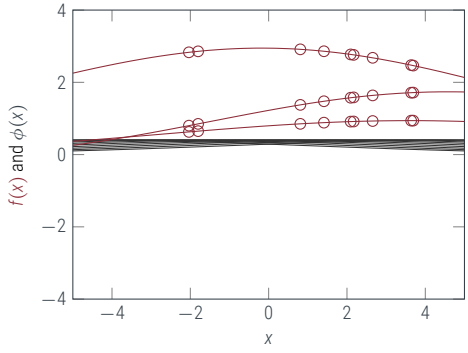


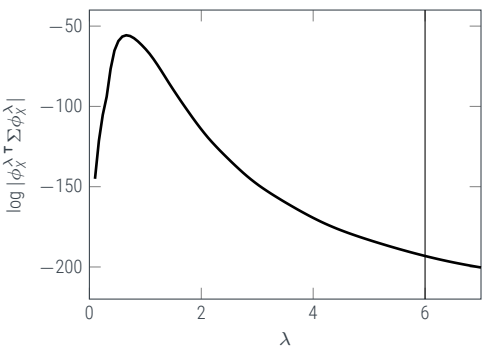
$$\log \left| \phi_X^{\boldsymbol{\theta}\,\mathsf{T}} \Sigma \phi_X^{\boldsymbol{\theta}} + \Lambda \right|$$

measures model complexity as the "volume" of hypotheses covered by the joint Gaussian distribution.

What is Model Complexity?

The Occam factor is not always straightforward



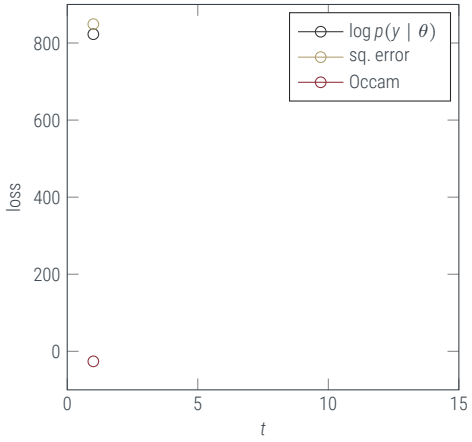


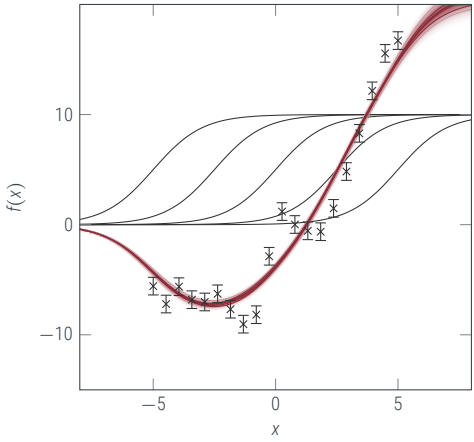
$$\log \left| \phi_X^{\boldsymbol{\theta}\,\mathsf{T}} \Sigma \phi_X^{\boldsymbol{\theta}} + \Lambda \right|$$

measures model complexity as the "volume" of hypotheses covered by the joint Gaussian distribution.

Type II Inference

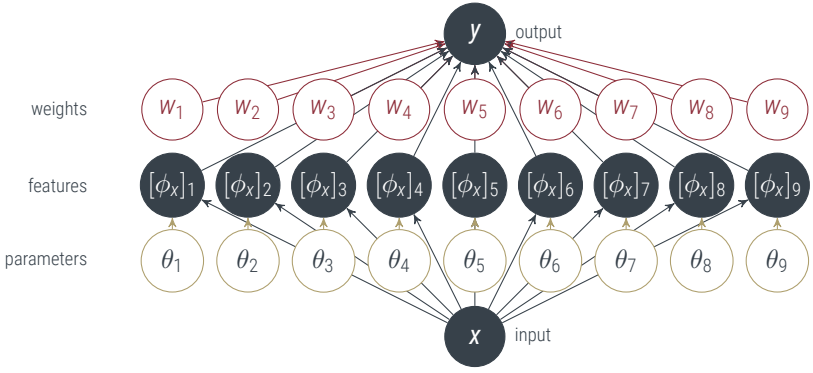
Fitting a probabilistic model by maximum marginal likelihood





A Structural Observation

Graphical Model



A linear Gaussian regressor is a **single hidden layer** neural network, with quadratic output loss, and fixed input layer. Hyperparameter-fitting corresponds to training the input layer. The usual way to train such network, however, does not include the Occam factor.

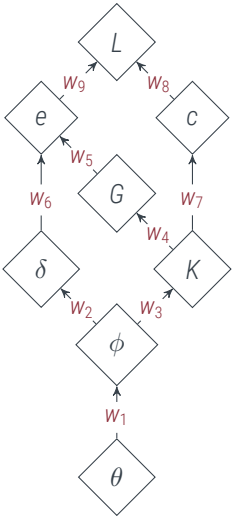
What does the Optimizer need from us?

A bit of algorithmic wizardry

$$L(\theta) = \frac{1}{2}\Bigg(\underbrace{(y - \phi_X^{\theta\intercal}\mu)^\intercal \underbrace{\Bigg(\underbrace{\phi_X^{\theta\intercal}\Sigma\phi_X^{\theta}}_{=:K} + \Lambda\Bigg)^{-1}}_{=:G} \overbrace{(y - \phi_X^{\theta\intercal}\mu)}^{=:\Delta}}_{=:e} + \underbrace{\log\left|\phi_X^{\theta\intercal}\Sigma\phi_X^{\theta} + \Lambda\right|}_{=:c} \Bigg)$$

What does the Optimizer need from us?

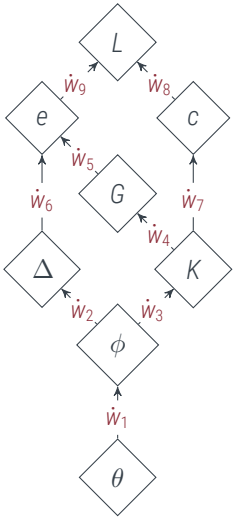
Automatic Differentiation



$$L(\theta) = \frac{1}{2}\Bigg(\underbrace{(y - \phi_X^{\boldsymbol{\theta}\intercal}\mu)^\intercal \underbrace{\Bigg(\underbrace{\phi_X^{\boldsymbol{\theta}\intercal}\Sigma\phi_X^{\boldsymbol{\theta}}}_{=:K} + \Lambda\Bigg)^{-1}}_{=:G} \overbrace{(y - \phi_X^{\boldsymbol{\theta}\intercal}\mu)}^{=:\Delta}}_{=:e} + \underbrace{\log\left|\phi_X^{\boldsymbol{\theta}\intercal}\Sigma\phi_X^{\boldsymbol{\theta}} + \Lambda\right|}_{=:c} \Bigg)$$

What does the Optimizer need from us?

Automatic Differentiation — Forward Mode

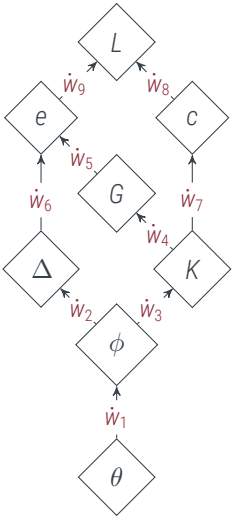


$$L(\theta) = \frac{1}{2}\Bigg(\underbrace{(\boldsymbol{y} - \phi_X^{\boldsymbol{\theta}\intercal}\mu)^\intercal \underbrace{\Bigg(\underbrace{\phi_X^{\boldsymbol{\theta}\intercal}\Sigma\phi_X^{\boldsymbol{\theta}}}_{=:K} + \Lambda \Bigg)^{-1}}_{=:G} \overbrace{(\boldsymbol{y} - \phi_X^{\boldsymbol{\theta}\intercal}\mu)}^{=:\Delta}}_{=:e} + \underbrace{\log\left|\phi_X^{\boldsymbol{\theta}\intercal}\Sigma\phi_X^{\boldsymbol{\theta}} + \Lambda\right|}_{=:c} \Bigg)$$

$$\begin{aligned}
\frac{\partial L}{\partial \theta} &= \frac{\partial L}{\partial e}\frac{\partial e}{\partial \theta} + \frac{\partial L}{\partial c}\frac{\partial c}{\partial \theta} = \dot{w}_9\frac{\partial e}{\partial \theta} + \dot{w}_8\frac{\partial c}{\partial \theta} = \dot{w}_9\left(\frac{\partial e}{\partial \Delta}\frac{\partial \Delta}{\partial \theta} + \frac{\partial e}{\partial G}\frac{\partial G}{\partial \theta}\right) + \dot{w}_8\frac{\partial c}{\partial K}\frac{\partial K}{\partial \theta} \\
&= \dot{w}_9\left(\dot{w}_6\frac{\partial \Delta}{\partial \theta} + \dot{w}_5\frac{\partial G}{\partial \theta}\right) + \dot{w}_8\dot{w}_7\frac{\partial K}{\partial \theta} = \dot{w}_9\left(\dot{w}_6\frac{\partial \Delta}{\partial \phi}\frac{\partial \phi}{\partial \theta} + \dot{w}_5\frac{\partial G}{\partial K}\frac{\partial K}{\partial \theta}\right) + \dot{w}_8\dot{w}_7\frac{\partial K}{\partial \theta} \\
&= \dot{w}_9\dot{w}_6\dot{w}_2\frac{\partial \phi}{\partial \theta} + (\dot{w}_9\dot{w}_5\dot{w}_4 + \dot{w}_8\dot{w}_7)\frac{\partial K}{\partial \theta} = \left(\dot{w}_9\dot{w}_6\dot{w}_2 + (\dot{w}_9\dot{w}_5\dot{w}_4 + \dot{w}_8\dot{w}_7)\frac{\partial K}{\partial \phi}\right)\frac{\partial \phi}{\partial \theta} \\
&= (\dot{w}_9\dot{w}_6\dot{w}_2 + (\dot{w}_9\dot{w}_5\dot{w}_4 + \dot{w}_8\dot{w}_7)\,\dot{w}_3)\frac{\partial \phi}{\partial \theta}\frac{\partial \theta}{\partial \theta} \\
&= (\dot{w}_9\dot{w}_6\dot{w}_2 + (\dot{w}_9\dot{w}_5\dot{w}_4 + \dot{w}_8\dot{w}_7)\,\dot{w}_3)\,\dot{w}_1 1
\end{aligned}$$

What does the Optimizer need from us?

Automatic Differentiation — Forward Mode



$$L(\theta) = \underbrace{\frac{1}{2}\Big(\underbrace{(y-\phi_X^{\boldsymbol{\theta}\intercal}\mu)^\intercal \underbrace{\Big(\underbrace{\phi_X^{\boldsymbol{\theta}\intercal}\Sigma\phi_X^{\boldsymbol{\theta}}}_{=:K}+\Lambda\Big)^{-1}}_{=:G} \overbrace{(y-\phi_X^{\boldsymbol{\theta}\intercal}\mu)}^{=:\Delta}}_{=:e} + \underbrace{\log\left|\phi_X^{\boldsymbol{\theta}\intercal}\Sigma\phi_X^{\boldsymbol{\theta}}+\Lambda\right|}_{=:c}\Big)}$$

$$\dot{w}_9 = \frac{\partial L}{\partial e} = 1/2 \qquad \dot{w}_8 = \frac{\partial L}{\partial c} = 1/2 \qquad [\dot{w}_7]_{ij} = \frac{\partial c}{\partial K_{ij}} = K^{-1}_{ij}$$

$$[\dot{w}_6]_i = \frac{\partial e}{\partial \Delta_i} = 2[G\Delta]_i \qquad [\dot{w}_5]_{ij} = \frac{\partial e}{\partial G_{ij}} = \Delta_i\Delta_j \qquad [\dot{w}_4]_{ij,k\ell} = \frac{\partial G_{ij}}{\partial K_{k\ell}} = -G_{ik}G_{j\ell}$$

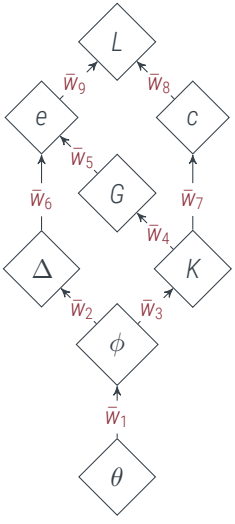
$$[\dot{w}_3]_{ij,ab} = \frac{\partial K_{ij}}{\partial \phi_{ab}} = \delta_{ia}[\Sigma\phi]_{bj} + \delta_{jb}[\Sigma\phi]_{kj}$$

$$[\dot{w}_2]_{i,ab} = \frac{\partial \Delta_i}{\partial \phi_{ab}} = -\delta_{ia}\mu_b \qquad [\dot{w}_1]_{ab,\ell} = \frac{\partial \phi_{ab}}{\partial \theta_\ell} = \text{your choice!}$$

What does the Optimizer need from us?

Automatic Differentiation — Backward Mode

[Seppo Linnainmaa, 1970]



$$L(\theta) = \frac{1}{2}\Bigg(\underbrace{(y-\phi_X^{\boldsymbol{\theta}\intercal}\mu)^\intercal \underbrace{\Bigg(\underbrace{\phi_X^{\boldsymbol{\theta}\intercal}\Sigma\phi_X^{\boldsymbol{\theta}}}_{=:K}+\Lambda\Bigg)^{-1}}_{=:G} \overbrace{(y-\phi_X^{\boldsymbol{\theta}\intercal}\mu)}^{=:\Delta}}_{=:e} + \underbrace{\log\left|\phi_X^{\boldsymbol{\theta}\intercal}\Sigma\phi_X^{\boldsymbol{\theta}}+\Lambda\right|}_{=:c} \Bigg)$$

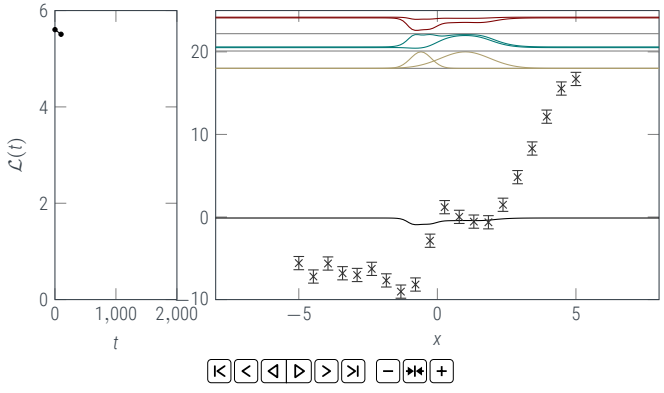
$$\frac{\partial L}{\partial \theta} = \frac{\partial L}{\partial \phi}\frac{\partial \phi}{\partial \theta} =: \bar{w}_1 = \left(\frac{\partial L}{\partial \Delta}\frac{\partial \Delta}{\partial \phi} + \frac{\partial L}{\partial K}\frac{\partial K}{\partial \phi}\right)\frac{\partial \phi}{\partial \theta} =: (\bar{w}_2+\bar{w}_3)\frac{\partial \phi}{\partial \theta}$$

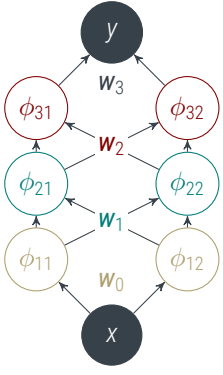
$$\bar{w}_2 = \frac{\partial L}{\partial e}\frac{\partial e}{\partial \Delta}\frac{\partial \Delta}{\partial \phi} =: \bar{w}_6\frac{\partial \Delta}{\partial \phi} \qquad \bar{w}_3 = \left(\frac{\partial L}{\partial G}\frac{\partial G}{\partial K} + \frac{\partial L}{\partial c}\frac{\partial c}{\partial K}\right)\frac{\partial K}{\partial \phi} =: (\bar{w}_4+\bar{w}_7)\frac{\partial K}{\partial \phi}$$

$$\bar{w}_4 = \frac{\partial L}{\partial e}\frac{\partial e}{\partial G}\frac{\partial G}{\partial K} =: \bar{w}_5\frac{\partial G}{\partial K} \qquad \bar{w}_5 = \frac{\partial L}{\partial e}\frac{\partial e}{\partial G} =: \bar{w}_9\frac{\partial e}{\partial G} \qquad \bar{w}_6 = \frac{\partial L}{\partial e}\frac{\partial e}{\partial \Delta} =: \bar{w}_9\frac{\partial e}{\partial \Delta}$$

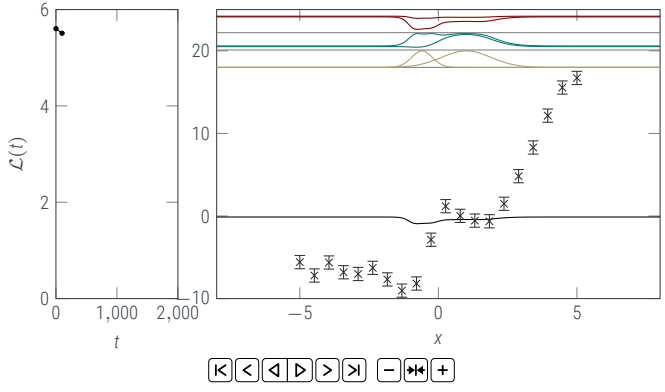
$$\bar{w}_7 = \frac{\partial L}{\partial c}\frac{\partial c}{\partial K} =: \bar{w}_8\frac{\partial c}{\partial K} \qquad \bar{w}_8 = \bar{w}_9 = 1/2$$

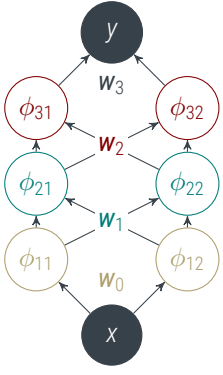
$\bar{w}_i = \frac{\partial L}{\partial \text{subgraph}_i}$ are known as *adjoints*. Traverse graph backward to collect the derivative. This is faster than forward-mode for single-output-many-input functions, but requires storing the above structure (known as a *Wengert list*). (cf. "Backpropagation")





$$\hat{f}(x, W) = \sum_{i=1}^{F} \phi_{3i}(x, w_{\text{lower}}) w_{3i} = \sum_{i} \phi_{3i} \left(\sum_{j} \phi_{2j} \left(\sum_{\ell} \phi_{1\ell}(w_{0\ell} x) w_{1\ell j} \right) w_{2ji} \right) w_{3i}$$





$$\hat{f}(x, \boldsymbol{W}) = \arg\min_{\boldsymbol{W} \in \mathbb{R}^D} \|\boldsymbol{y} - \hat{\boldsymbol{f}}(\boldsymbol{x}, \boldsymbol{W})\|^2 + \alpha^2 \|\boldsymbol{W}\|^2 = \mathcal{L}(\boldsymbol{W})$$

$$\boldsymbol{W}_{t+1} = \boldsymbol{W}_t + \tau \nabla \mathcal{L}(\boldsymbol{W})$$

What about Mini-batches?

Learning representations from big data

$$p(w, \theta \mid y) \propto p(y \mid w, \phi^{\theta}) p(w, \theta) = p(w, \theta) \cdot \prod_{i=1}^{n} p(y_i \mid w, \phi_i^{\theta}) = p(w, \theta) \cdot \prod_{i=1}^{n} \mathcal{N}(y_i; \phi_i^{\theta\intercal} w, \sigma^2)$$

What about Mini-batches?

Learning representations from big data

$$p(w, \theta \mid y) \propto p(y \mid w, \phi^{\theta}) p(w, \theta) = p(w, \theta) \cdot \prod_{i=1}^{n} p(y_i \mid w, \phi_i^{\theta}) = p(w, \theta) \cdot \prod_{i=1}^{n} \mathcal{N}(y_i; \phi_i^{\theta\intercal} w, \sigma^2)$$

$$\arg\max_{w,\theta} p(w, \theta \mid y) = \arg\min_{w,\theta} -\log p(w, \theta \mid y)$$

What about Mini-batches?

Learning representations from big data

$$p(w, \theta \mid y) \propto p(y \mid w, \phi^{\theta}) p(w, \theta) = p(w, \theta) \cdot \prod_{i=1}^{n} p(y_i \mid w, \phi_i^{\theta}) = p(w, \theta) \cdot \prod_{i=1}^{n} \mathcal{N}(y_i; \phi_i^{\theta\intercal} w, \sigma^2)$$

$$\begin{aligned}
\arg\max_{w,\theta} p(w, \theta \mid y) &= \arg\min_{w,\theta} -\log p(w, \theta \mid y) \\
&= \arg\min_{w,\theta} -\log p(w, \theta) - \frac{1}{2\sigma^2} \sum_{i=1}^{n} \|y_i - \phi_i^{\theta\intercal} w\|^2 = \arg\min_{w,\theta} r(w, \theta) + \underbrace{\sum_{i=1}^{n} \ell_2(y_i; \theta, w)}_{\mathcal{L}(\theta, w)}
\end{aligned}$$

$$p(w, \theta \mid y) \propto p(y \mid w, \phi^{\theta}) p(w, \theta) = p(w, \theta) \cdot \prod_{i=1}^{n} p(y_i \mid w, \phi_i^{\theta}) = p(w, \theta) \cdot \prod_{i=1}^{n} \mathcal{N}(y_i; \phi_i^{\theta\intercal} w, \sigma^2)$$

$$\begin{aligned}
\arg\max_{w,\theta} p(w, \theta \mid y) &= \arg\min_{w,\theta} -\log p(w, \theta \mid y) \\
&= \arg\min_{w,\theta} -\log p(w, \theta) - \frac{1}{2\sigma^2} \sum_{i=1}^{n} \|y_i - \phi_i^{\theta\intercal} w\|^2 = \arg\min_{w,\theta} r(w, \theta) + \underbrace{\sum_{i=1}^{n} \ell_2(y_i; \theta, w)}_{\mathcal{L}(\theta, w)} \\
&\approx \arg\min_{w,\theta} -\log p(w, \theta) - \frac{1}{2\sigma^2} \frac{n}{b} \sum_{\beta=1}^{b} \|y_\beta - \phi_\beta^{\theta\intercal} w\|^2
\end{aligned}$$

$$p(w, \theta \mid y) \propto p(y \mid w, \phi^{\theta}) p(w, \theta) = p(w, \theta) \cdot \prod_{i=1}^{n} p(y_i \mid w, \phi_i^{\theta}) = p(w, \theta) \cdot \prod_{i=1}^{n} \mathcal{N}(y_i; \phi_i^{\theta\intercal} w, \sigma^2)$$

$$\begin{aligned}
\arg\max_{w,\theta} p(w, \theta \mid y) &= \arg\min_{w,\theta} -\log p(w, \theta \mid y) \\
&= \arg\min_{w,\theta} -\log p(w, \theta) - \frac{1}{2\sigma^2} \sum_{i=1}^{n} \|y_i - \phi_i^{\theta\intercal} w\|^2 = \arg\min_{w,\theta} r(w, \theta) + \underbrace{\sum_{i=1}^{n} \ell_2(y_i; \theta, w)}_{\mathcal{L}(\theta, w)} \\
&\approx \arg\min_{w,\theta} -\log p(w, \theta) - \frac{1}{2\sigma^2} \frac{n}{b} \sum_{\beta=1}^{b} \|y_\beta - \phi_\beta^{\theta\intercal} w\|^2 \sim \mathcal{N}\left(r + \mathcal{L}(\theta, w), \mathcal{O}(b^{-1})\right)
\end{aligned}$$

Connections and Differences

Bayesian and Deep Learning

- MAP inference does not capture **uncertainty** on parameters:
  - no posterior uncertainty from not fully identified parameters
  - no model capacity control in the evidence term
- A linear Gaussian regressor is a **single hidden layer** neural network, with quadratic output loss, and fixed input layer (deep networks can of course be treated in the same way). Hyperparameter-fitting corresponds to training the input layer. The usual way to train such network, however, does not include the Occam factor. Data sub-sampling can be used just as in other areas to speed up computations at the cost of reduced computational precision.
- All worries one may have about fitting or hand-picking features for Bayesian regression also apply to deep learning. By highlighting assumptions and piors, the probabilistic view forces us to address many problems directly, rather than obscuring them with notation and intuitions.
- **Automatic Differentiation (AD)** is a algorithmic tool that is just as helpful for Bayesian inference as it is for deep learning

It is possible to construct a point estimate for a Bayesian model, and to construct full posteriors for deep networks. The two domains are not separate, they are just different mental scaffolds. If you're hoping for a theory of deep learning, probability theory is a primary contender.

Summary:

- The features used for Gaussian linear regression can be **learnt** by **hierarchical Bayesian Inference**
- This is usually **intractable**. Instead, **approximate inference** methods are used
- For example, **maximal a-posteriori probability (MAP)** inference fits a point-estimate for feature parameters
- MAP inference is an **optimization** problem, and can thus be performed in the same way as other optimization-based ML approaches, including deep learning. That is, using the same optimizers (e.g. stochastic gradient descent), the same automatic differentiation frameworks (e.g. TensorFlow / pyTorch, etc.) and the same data subsampling techniques.

The different viewpoints (probabilistic / statistical / empirical ("deep")) on Machine Learning often overlap and inform each other. Understanding of Bayesian linear (Gaussian) regression can help us build a better intuition for deep learning, too.

Next lecture: Instead of *learning* a few features, sometimes we can get away with using *infinitely many* features.