

02477 – Bayesian Machine Learning: Lecture 1

Technical University of Denmark,
DTU Compute, Department of Applied Math and Computer Science

Outline for week 1

- ① Introduction and course formalities
- ② Bayesian machine learning
- ③ Brief recap on terminology from probability theory
- ④ The binomial model and maximum likelihood estimation
- ⑤ Bayesian inference and the Beta-binomial models
- ⑥ Introduction to the exercise sessions

Introduction and course formalities

Bayesian Machine Learning

■ Machine learning

- Understanding and finding pattern in data
- Making machines learn from data
- Making predictions



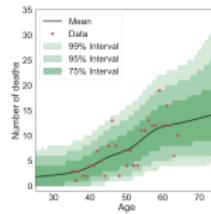
■ A multitude of applications

- Object detection
- Speech recognition and natural language processing
- Self-driving cars
- Spam & fraud detection
- Recommender systems
- Brain imaging
- ...



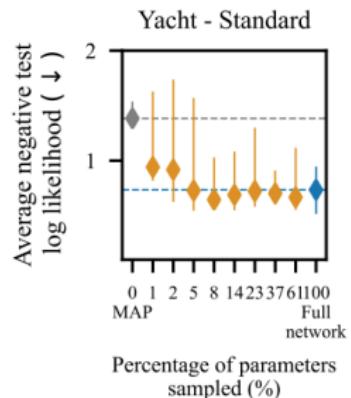
■ Bayesian statistics

- Named after Thomas Bayes (1702 - 1761)
- Mathematical framework for reasoning with uncertainty

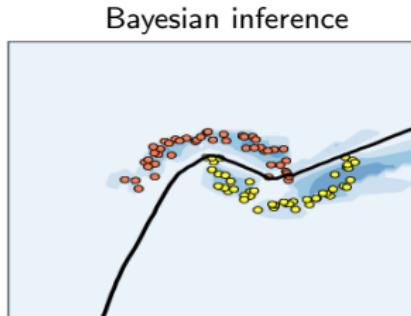


Motivation for studying Bayesian methods for machine learning

- Very flexible and intuitive modelling framework
- Can be less prone to overfitting
- Intuitive uncertainty quantification
- Can be computationally more intensive
- Deeper understanding of "traditional" methods
- Bayesian deep learning
 - Can improve generalization
 - May improve calibration and reduce overconfidence



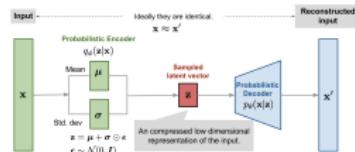
Kristiadi et al, 2019 and Sharma et al, 2022



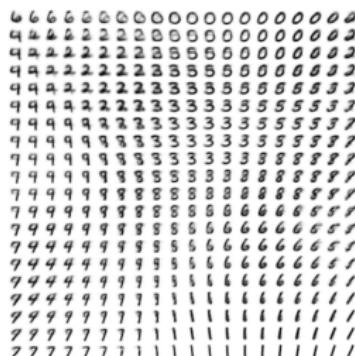
1.0
0.9
0.8
0.7
0.6
0.5

A probabilistic perspective on machine learning

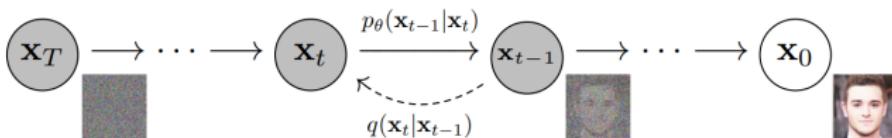
- We will study machine learning from a *probabilistic perspective*
- Interpreting classic methods in a probabilistic setting reveals a connection between loss functions (e.g. squared loss, cross entropy etc.) and probability distributions
- Gives *deeper insights into fundamentals* and enables us to *adapt methods to different types of data*
- The probabilistic framework is the *foundation for advanced machine learning* e.g. generative models such as variational autoencoders and diffusion models are typically explained using *variational methods*



<https://lilianweng.github.io/posts/2018-08-12-vae/>



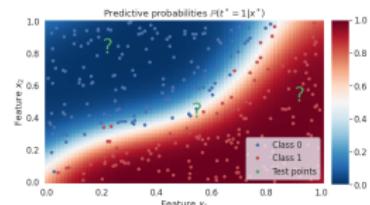
Kingma & Welling, 2014: Auto-encoding variational Bayes



What to expect from this course?

- We will work with several machine learning problems

- Regression
- Classification
- Clustering
- Optimal decision-making
- ...



- A probabilistic modelling approach

- Probability distributions as Lego-blocks
- Linear models, Gaussian processes, Neural networks
- Relation to common loss functions (cross-entropy, MSE etc.)

- Inference methods

- Maximum likelihood
- Bayesian inference
- Laplace approximations
- Variational inference
- Markov chain Monte Carlo methods

- Bayesian perspective: understanding how and why

- Generalizes "classical" training
- Incorporating prior knowledge into models
- Uncertainty quantification
- Deeper insights into fundamentals
- Emphasis on bridging gap between theory and intuition

- Prepare you for a M.Sc. thesis in advanced machine learning

Course formalities

- Flipped classroom
 - Exercise sessions every Monday 13-17
 - Pen & paper exercises and programming exercises
 - Engage with teachers/teaching assistant highly encouraged
- Location and other practical information will be announced on DTU Learn ASAP
- Assignments
 - 3 mandatory assignments
 - Groups of 3-5
 - Feedback for student and teachers
- Written exam
 - Curriculum: All course materials, e.g. slides, book chapter, exercises, assignments etc.
 - Details T.B.A.
- Teachers and teaching assistants
 - Course responsible: Michael Riis Andersen (Building 321, room 216, miri@dtu.dk)
 - TAs T.B.A.

Course plan

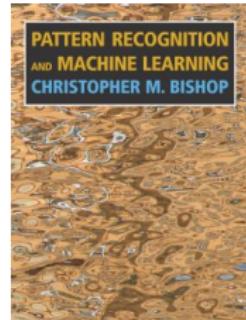
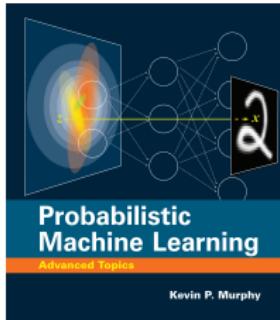
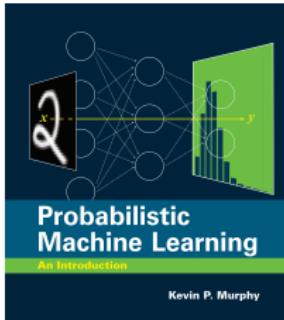
The plan is subject to change!

Week	Topic
1	Intro, basic concepts, Beta-Binomial model
2	Making predictions, grid approximations
3	Bayesian classification and Laplace approximations
4	Bayesian linear regression
5	Distributions on function spaces, Gaussian Processes
6	Gaussian process classification
7	Multi-class classification and decision theory
8	Monte Carlo & Markov Chain Monte Carlo methods
9	More on MCMC
10	Mixture models and variational inference
11	Black-box variational inference
12	More on variational inference
13	Bayesian neural networks

Course plan & list of reading material for each week can be on DTU Learn

Literature

- Main textbooks: Probabilistic Machine Learning by Kevin Murphy (vol. 1 & 2)
- Supplement: Pattern Recognition and machine learning by Christopher Bishop



- All freely available as PDFs
 - <https://probml.github.io/pml-book/book1.html>
 - <https://probml.github.io/pml-book/book2.html>
 - <https://www.microsoft.com/en-us/research/people/cmbishop/prml-book/>
- We will supplement with other resources in the second half of the course, e.g. book chapters, research papers etc

Course prerequisites

- 02450 Introduction to machine learning
- Math
 - 1. Linear algebra (Bishop appendix C)
 - 2. Calculus
 - 3. Probability theory (Bishop chap. 1, 2, appendix B)
 - 4. Statistics
- Programming
 - Python
 - Numpy, scipy, matplotlib etc
- It is indeed possible to complete the course without the all prerequisites, but you should *expect a significantly increased workload*



Continuous feedback

- I need your help to improve the course!
- Will announce feedback persons every week
- We meet on Monday at 16:45 (exact location TBA)
- Ideas for feedback
 - Were you able to follow the lecture?
 - How far did you make it in the exercise?
 - What was easy, what was difficult?
 - What did you like?
 - What can be improved?
 - etc etc.



Bayesian machine learning

Classical machine learning: supervised learning for regression

Common steps to fit a model to a given dataset $\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=1}^N$

1. Choose a model

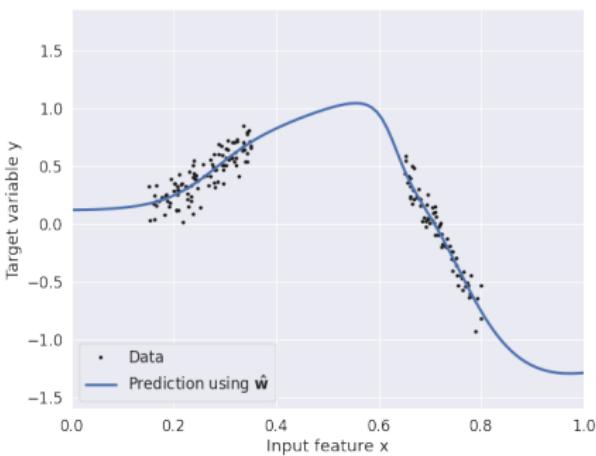
$$y_i = f(\mathbf{x}_i | \mathbf{w}) + e_i$$

2. Choose a loss function

$$\mathcal{L}(y, \hat{y}) = (y - \hat{y})^2$$

3. Find parameters \mathbf{w} that minimizes the average loss \mathcal{L} for the data set

$$\hat{\mathbf{w}} = \arg \min_{\mathbf{w}} \sum_{i=1}^N [y - f(\mathbf{x}_i | \mathbf{w})]^2$$

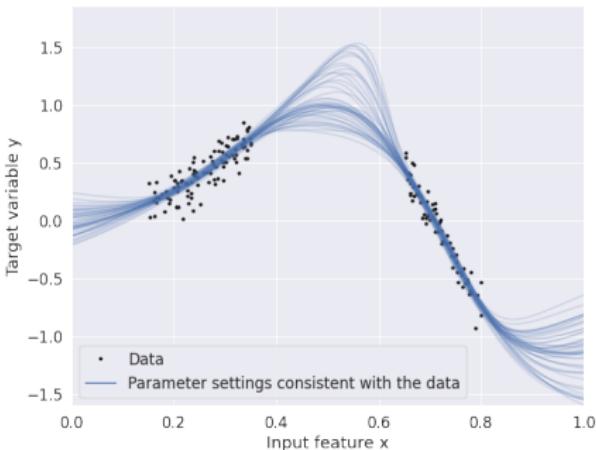


4. Make predictions for \mathbf{x}^* using estimated parameters \mathbf{w}

$$y^* = f(\mathbf{x}^* | \hat{\mathbf{w}})$$

Model ambiguity due to finite data

- For a given model, there may be several sets of model parameters consistent with the data
- Model parameters \hat{w}_1 , \hat{w}_2 , and \hat{w}_3 are all consistent with the data (as measured by training loss)
- Often many parameter settings consistent with data, but each can lead to very different predictions
- Classical machine learning: we choose *one* of these sets of parameters
- Bayesian machine learning: take *all* sets of model parameters consistent with the data into account



Bayesian inference and marginalization

- The *posterior distribution* $p(\mathbf{w}|\mathbf{y})$ measures how much weight to assign to each parameter set

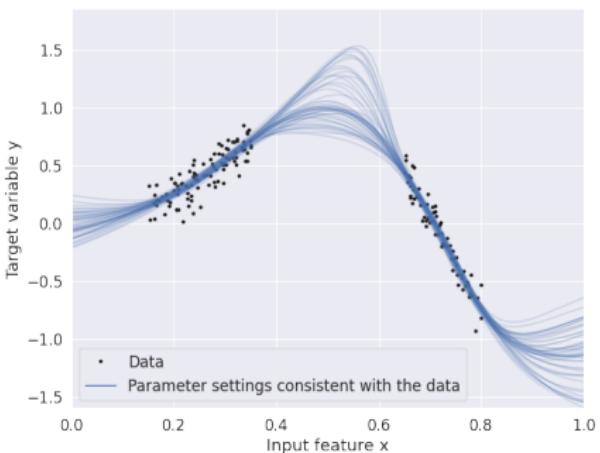
- Making predictions using a weighted average of all possible parameter sets

$$y^* = \sum_{i=1}^M f(x^* | \mathbf{w}_i) p(\mathbf{w}_i | \mathbf{y}),$$

- Often, we have infinitely many parameter settings

$$y^* = \int f(x^* | \mathbf{w}) p(\mathbf{w} | \mathbf{y}) d\mathbf{w}$$

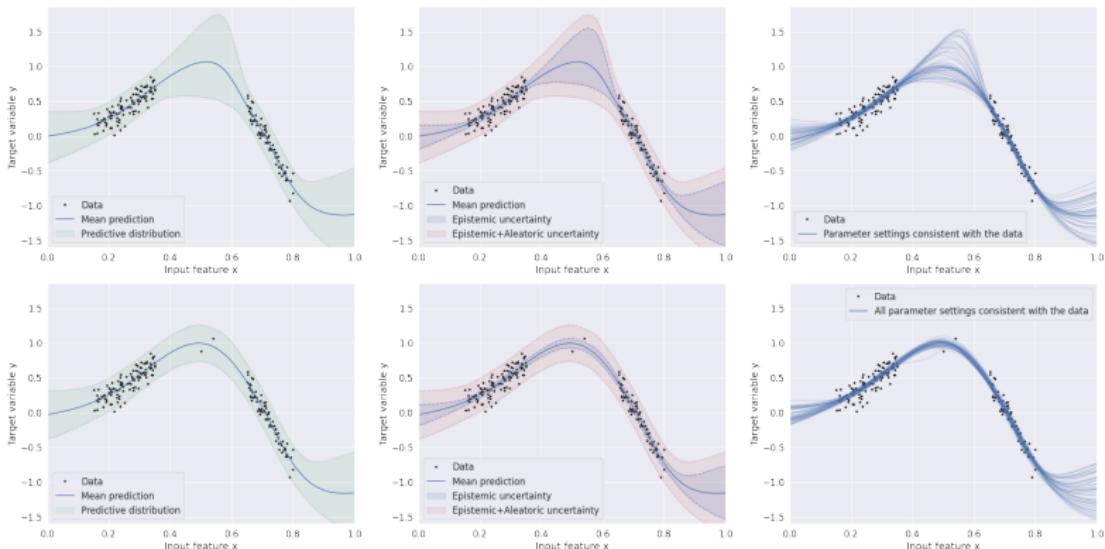
- The process takes the uncertainty about the parameters into account and is called *marginalization*



Uncertainty quantification

Two sources of uncertainty

1. *Epistemic uncertainty* is due to lack of knowledge (e.g. often due to a limited data set). Also sometimes called the *reducible* uncertainty.
2. *Aleatoric uncertainty* refers to the inherent randomness (e.g. measurement noise). Also sometimes called the *irreducible* uncertainty.



Bayesian machine learning

- In Bayesian methods, *all variables* (e.g. both parameters and data) are represented using *probability distributions*
- *Bayes' rule* provides a systematic way to combine data with prior knowledge

$$p(\mathbf{w}|\mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{w})p(\mathbf{w})}{p(\mathbf{y})}$$

- *Likelihood* $p(\mathbf{y}|\mathbf{w})$ - distribution of data \mathbf{y} given a specific set of parameters \mathbf{w}
- *Prior* $p(\mathbf{w})$ - prior belief about parameters \mathbf{w} before seeing any data
- *Posterior* $p(\mathbf{w}|\mathbf{y})$ - contains all knowledge about parameters after seeing data \mathbf{y}
- Why use priors?
 1. can encode domain knowledge
 2. can help prevent overfitting
 3. can generate artificial datasets from the model
- Why distributions rather than point estimates?
 1. Easy uncertainty quantification
 2. Avoid making predictions when uncertain is too large
 3. Better decision-making

Example: how can uncertainty improve decision making?

Example from Section 4.6.7.3 in Murphy1

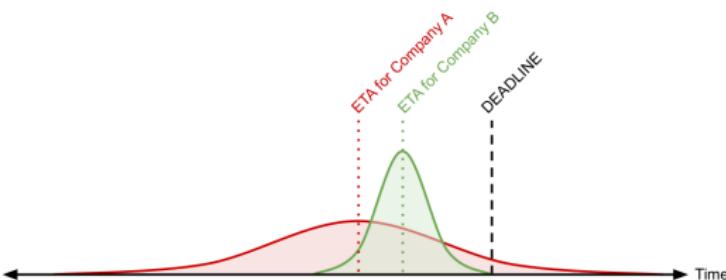


Figure 4.21: Distribution of arrival times for two different shipping companies. ETA is the expected time of arrival. A's distribution has greater uncertainty, and may be too risky. From <https://bit.ly/39bc4XL>. Used with kind permission of Brendan Hasz.

- Suppose you run a company that have promised to delivery a package to a customer before some deadline
- Goal: predict the *delivery time* for the package if shipped with company A rather than company B
- Based on previous shipments, we have estimated the mean and variance for the delivery times of the two company
- On *average company A is faster*, but due to the large variance we might not meet the deadline. Hence, *despite company B being slower on average, it might be the best option*.

Brief recap on terminology from probability theory

Probability notation and terminology

	Discrete distributions	Continuous distributions
Sample space	$\{0, 1\}, \{1, 2, 3, 4\}, \{\text{cat, dog}\}$	$\mathbb{R}, \mathbb{R}_+, [0, 1]$
Representation	Probability mass function (PMF) $0 \leq p(x) \leq 1$	Probability density functions (PDF) $p(x) \geq 0$
	$\sum_x p(x) = 1$	$\int p(x) dx = 1$
		$p(x \in [a, b]) = \int_a^b p(x) dx$
Mean	$\mathbb{E}[x] = \sum_x x p(x)$	$\mathbb{E}[x] = \int x p(x) dx$
Variance	$\mathbb{V}[x] = \sum_x (x - \mathbb{E}[x])^2 p(x)$	$\mathbb{V}[x] = \int (x - \mathbb{E}[x])^2 p(x) dx$
General expectations	$\mathbb{E}[f(x)] = \sum_x f(x) p(x)$	$\mathbb{E}[f(x)] = \int f(x) p(x) dx$
Joint distribution	$p(x, y)$	$p(x, y)$
Conditional distribution	$p(x y) = \frac{p(x,y)}{p(y)}$	$p(x y) = \frac{p(x,y)}{p(y)}$
Sum rule (marginalization)	$p(x) = \sum_y p(x, y)$	$p(x) = \int p(x, y) dy$
Product rule	$p(x, y) = p(x y)p(y)$	$p(x, y) = p(x y)p(y)$
Independence	$p(x, y) = p(x)p(y)$	$p(x, y) = p(x)p(y)$

Common operations in probabilistic machine learning

- Evaluation: evaluating the probability density of a random variable X taking value x

$$p(X = x) = p(x) \quad (\text{Lazy notation})$$

- Conditioning: Deriving the distribution of X conditioned on observed data \mathcal{D}

$$p(x|\mathcal{D}) = \frac{p(x, \mathcal{D})}{p(\mathcal{D})}$$

- Maximization: What is the most likely value for a given distribution?

$$\hat{x} = \arg \max_x p(x|\mathcal{D})$$

- Marginalizing: E.g. accounting for unobserved quantities

$$p(x) = \int p(x, y) dy$$

- Sampling: Generate samples from a given some probability distribution

$$x^{(i)} \sim p(x|\mathcal{D})$$

- Compute expectations and moments

$$\mu = \mathbb{E}[x] = \int xp(x)dx$$

- Constructing intervals: E.g. find values ℓ and u such that

$$p(\ell \leq x \leq u) = 0.95$$

The basic building blocks of probabilistic machine learning

Distribution	Outcome space	Examples
Bernoulli	0, 1	Binary data: Cancer/not cancer, click/no click
Binomial	0, 1, ..., N	k of out N successes
Poisson	0, 1, 2, ...	Count data: number of cancer cells in image
Categorical	1, 2, 3, ..., K	Multi-class: cat/dog/bird
Beta	[0, 1]	Reasoning about probabilities
Dirichlet	Probability simplex	Reasoning about probability vectors
Gaussian	\mathbb{R}	Real numbers
Student's t	\mathbb{R}	Real numbers
Gamma	\mathbb{R}_+	Non-negative real times, e.g. waiting times

The binomial model and maximum likelihood estimation

Motivating example: A/B testing

Your company's website has two ads. Ad A has been shown $N_A = 123$ times and generated $y_A = 12$ clicks, and Ad B has been shown $N_B = 145$ times and generated $y_B = 20$ clicks.

- What can we say about the click-rates for the two ads? Which one is best?
- We can calculate the sample click-rates

$$\hat{\theta}_A = \frac{y_A}{N_A} = \frac{12}{123} \approx 0.098, \quad \hat{\theta}_B = \frac{y_B}{N_B} = \frac{20}{145} \approx 0.138$$

- Should we trust these estimates or ask for more data?
- What is the probability that the population click-rate for ad B is below 10%?
- What is the probability that ad B generates more clicks than ad A?

We can answer such questions using the *beta-binomial model*

Goal for the rest of the lecture

- The *beta-binomial model* is a Bayesian model for estimating proportions, e.g.
 1. What proportion of users clicked the banner?
 2. What proportion of subject recovered after a certain medical treatment?
 3. What proportion of test images is correct classified?
 4. ...
- We will first look at the probabilistic building blocks
 1. Bernoulli distribution
 2. Binomial distribution
 3. Beta distribution
- Maximum likelihood inference
- Bayesian inference
- How does all this apply to A/B testing?

Recap: The Bernoulli distribution

Our first building block

- For a binary random variable $x \in \{0, 1\}$, where 1 represents "success"

$$P(x = 1|\theta) = \theta$$

$$P(x = 0|\theta) = 1 - \theta,$$

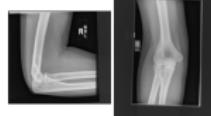
where $0 \leq \theta \leq 1$



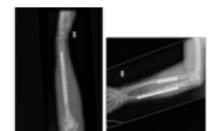
- *Bernoulli distribution* is a distribution over binary variables

$$\text{Ber}(x|\theta) = \theta^x(1 - \theta)^{1-x}$$

Elbow



Forearm



- The mean of Bernoulli variable

$$\mathbb{E}[x] = \theta$$

- The variance is

$$\mathbb{V}[x] = \theta(1 - \theta)$$

Click me!

Recap: The Binomial distribution

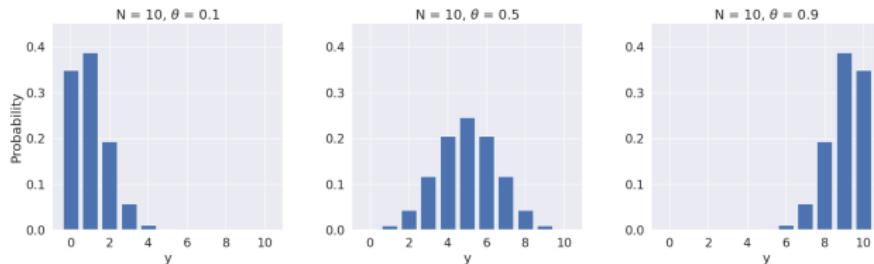
Modelling sequences of independent Bernoulli trials

- For a sequence of N independent Bernoulli trials $x_i \sim \text{Ber}(\theta)$ for $i = 1, \dots, N$, the number of successes $y = \sum x_i$ is said to follow a *Binomial distribution*

$$\text{Bin}(y|N, \theta) = \binom{N}{y} \theta^y (1 - \theta)^{N-y}$$

- Example: Suppose we flip a fair coin $N = 10$ times, the probability of getting $y = 4$ heads is given by

$$\text{Bin}(y = 4|N = 10, \theta = 0.5) = \binom{10}{4} 0.5^4 (1 - 0.5)^{10-4} \approx 0.21$$



- Calculating the mean of y using *linearity* of expectations

$$\mathbb{E}[y] = \mathbb{E}\left[\sum_{i=1}^N x_i\right] = \sum_{i=1}^N \mathbb{E}[x_i] = \sum_{i=1}^N \theta = N\theta$$

The Binomial distribution and maximum likelihood I

- Assume we collected a data set $\mathcal{D} = \{x_1, x_2, \dots, x_N\}$ of N independent Bernoulli trials with probability θ . How to estimate θ ?
- Let $y = \sum x_i$ denote number of successes, then

$$P(y|\theta) = \binom{N}{y} \theta^y (1-\theta)^{N-y}$$

- The *likelihood function* is defined as $\mathcal{L}(\theta) \equiv P(y|\theta)$
- The likelihood function measures: what is the probability of the observed data given the parameter value θ
- *Maximum likelihood:* We can estimate the parameters by maximizing the likelihood function \mathcal{L} wrt. θ

$$\hat{\theta}_{\text{MLE}} \equiv \arg \max_{\theta} \mathcal{L}(\theta) = \arg \max_{\theta} \log \mathcal{L}(\theta) = y/N$$

- We *maximize* the log likelihood function by *differentiating, equating to zero and solving* for θ .

$$\log \mathcal{L}(\theta) = \log \left[\binom{N}{y} \theta^y (1-\theta)^{N-y} \right] = \log \left[\binom{N}{y} \right] + y \log(\theta) + (N-y) \log(1-\theta)$$

The Binomial distribution and maximum likelihood II

$$\log \mathcal{L}_{\mathcal{D}}(\theta) = \log \left[\binom{N}{y} \right] + y \log(\theta) + (N - y) \log(1 - \theta)$$

- We *maximize* the log likelihood function by *differentiating, equating to zero and solving* for θ .
- Computing the derivative

$$\begin{aligned}\frac{d}{d\theta} \log \mathcal{L}_{\mathcal{D}}(\theta) &= \frac{d}{d\theta} \log \left[\binom{N}{y} \right] + \frac{d}{d\theta} \log(\theta)y + \frac{d}{d\theta} \log(1 - \theta)(N - y) \\ &= \frac{1}{\theta}y - \frac{1}{1 - \theta}(N - y) = 0\end{aligned}$$

- Solving for θ yields the *maximum likelihood estimator*

$$\hat{\theta}_{MLE} = \frac{y}{N} = \frac{1}{N} \sum_{n=1}^N x_i$$

- The solution is equal to the empirical mean of the dataset $\mathcal{D} = \{x_1, x_2, \dots, x_N\}$

The Binomial distribution and maximum likelihood: Quiz

- The *likelihood function* is defined as $\mathcal{L}(\theta) \equiv P(y|\theta)$ and the *maximum likelihood estimator* is

$$\hat{\theta}_{\text{MLE}} = \arg \max_{\theta} \mathcal{L}(\theta) = \frac{y}{N}$$

- Spend 4 minutes DTU Learn quiz: "Lecture 1: Likelihoods"

The Binomial distribution and maximum likelihood III

Small data and overfitting

- Suppose an ad is shown $N = 3$ times and observe $y = 0$ clicks, then

$$\hat{\theta}_{MLE} = 0/3 = 0$$

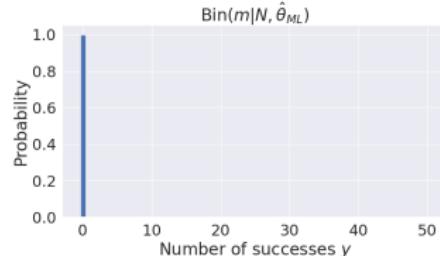
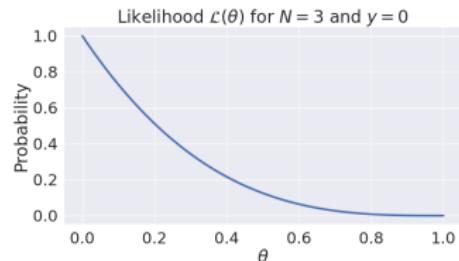
$$Bin(m|N, \theta) = \binom{N}{m} \theta^m (1 - \theta)^{N-m}$$

- What is the probability of exactly 0 clicks in the next 50 views? We *plug-in* the maximum likelihood estimator

$$\begin{aligned} P[y = 0|\theta_{MLE}] &= Bin(0|N = 50, \theta_{MLE}) \\ &= \binom{50}{0} \theta_{MLE}^0 (1 - \theta_{MLE})^{50-0} \\ &= 1 \end{aligned}$$

- According to this model, we are *absolutely sure* that there will be exactly 0 heads in the next 50 views based on information from *only 3 observations*.

- Does this seem like a reasonable conclusion?



Bayesian inference and the Beta-binomial models

Bayesian inference for θ

Example continued

- We observed $N = 3$ views and $y = 0$ clicks, then $\theta_{\text{MLE}} = 0/3 = 0$. This is *overfitting* and this is a common problem when using maximum likelihood for small data sets.
- We can reduce this effect using Bayesian inference

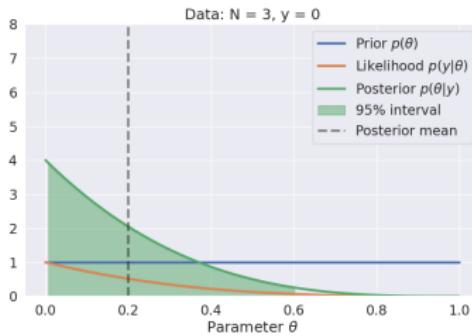
$$p(\theta|y) = \frac{p(y|\theta)p(\theta)}{p(y)}$$

- *The prior* $p(\theta)$ represents our prior belief about θ **before** seeing the data
- The *likelihood* $p(y|\theta)$ represents our information from data
- After observing y , *the posterior distribution* $p(\theta|y)$ summarizes all our available information about θ

Bayesian inference in images

- Bayes' rule gives for a probability distribution for θ conditioned on the observed value for y

$$p(\theta|y) = \frac{p(y|\theta)p(\theta)}{p(y)}$$



- We can estimate θ using the mean of the posterior distribution

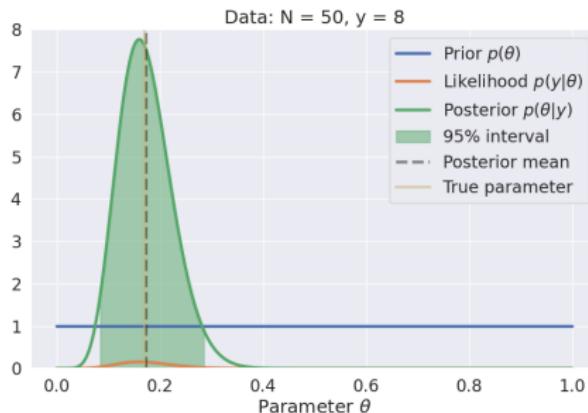
$$\theta_{\text{Bayes}} = \mathbb{E}[\theta|y] \equiv \int \theta p(\theta|y)d\theta = 0.2$$

- and use *credibility intervals* of the posterior to quantify the uncertainty

$$P(\theta \in [0.01, 0.60] | y) = 0.95$$

What happens as we collect more data?

- Simulated data: $x_i \sim \text{Ber}(\theta_0)$ for $i = 1, \dots, N$, where $\theta_0 = 0.17$



- The posterior concentrates as we collect more data

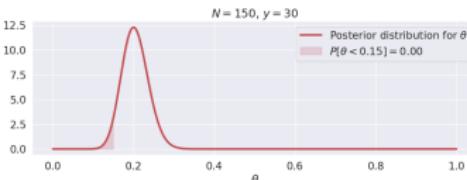
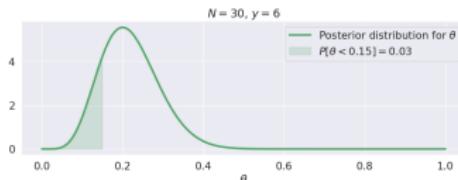
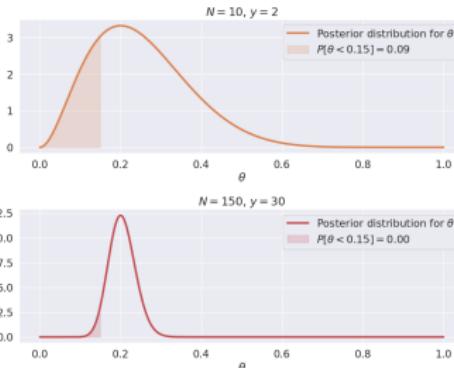
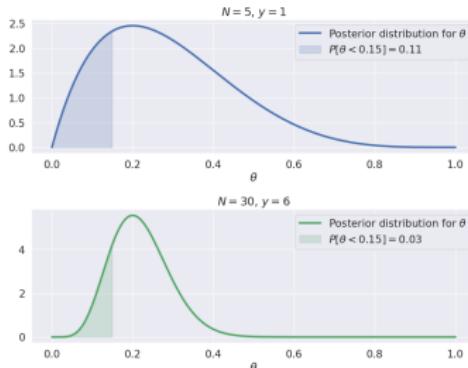
Bayesian analysis and probabilistic reasoning

From point estimates to probability distributions

- Bayesian analysis provides a *probability distribution* summarizing our knowledge of θ rather than a *point estimate* like $\hat{\theta}_{\text{MLE}}$
- Many common questions can be answered using *posterior summaries*, e.g. mode, mean, standard deviation, intervals, tail probabilities etc.

$$p(\theta < 0.15|y) = \int_0^{0.15} p(\theta|y) d\theta$$

Examples



The prior distribution: how to choose?

- The prior distribution $p(\theta)$ should reflect our prior assumptions before seeing the data

$$p(\theta|y) = \frac{p(y|\theta)p(\theta)}{p(y)} \propto p(y|\theta)p(\theta)$$

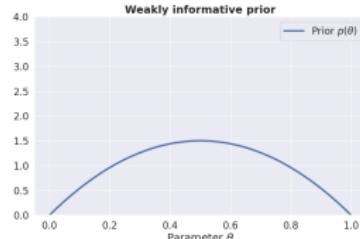
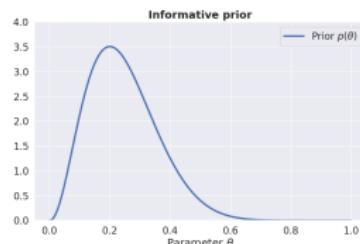
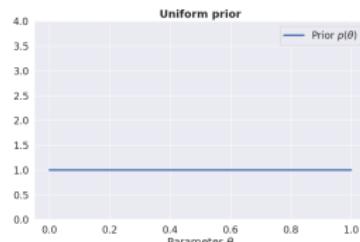
- Different types of priors

- Uniform priors
- Informative priors
- Weakly informative priors
- Priors for mathematical convenience

- Where does prior knowledge come from?

- Previous experiments
- Domain experts
- Regularization

- Specifying a prior forces us to be explicit about our assumptions



The Beta distribution

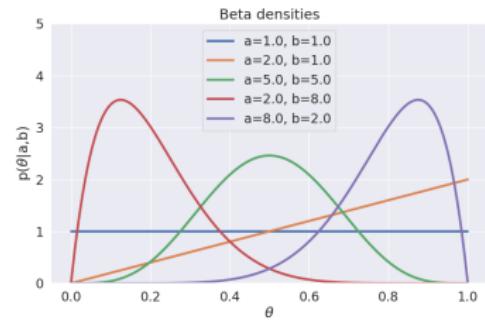
A mathematically convenient prior distribution for θ

- **Beta distribution** is a family of distributions for a random variable $\theta \in [0, 1]$ in the unit interval
- The density of the Beta distribution is given by

$$p(\theta|a, b) = \frac{1}{B(a, b)} \theta^{a-1} (1-\theta)^{b-1}$$

where $B(a, b)$ is a normalization constant given by

$$B(a, b) = \frac{\Gamma(a)\Gamma(b)}{\Gamma(a+b)}$$



$$\Gamma(z) = \int_0^\infty x^{z-1} e^{-x} dx$$

- Mean and variance

$$\mathbb{E}[\theta] = \frac{a}{a+b}$$

$$\mathbb{V}[\theta] = \frac{ab}{(a+b)^2(a+b+1)}$$

The functional form of a Beta distribution

- The density function of Beta distribution is given

$$p(\theta|a, b) = \frac{1}{B(a, b)} \theta^{a-1} (1-\theta)^{b-1}$$

- We know density functions integrate to one $\int p(\theta|a, b) = 1$, and hence

$$\int p(\theta|a, b) d\theta = \int \frac{1}{B(a, b)} \theta^{a-1} (1-\theta)^{b-1} d\theta = \frac{1}{B(a, b)} \int \theta^{a-1} (1-\theta)^{b-1} d\theta = 1$$

- This implies

$$B(a, b) = \int \theta^{a-1} (1-\theta)^{b-1} d\theta$$

- Example

$$p(\theta) \propto \theta^5 (1-\theta)^4 = \theta^{6-1} (1-\theta)^{5-1} \Rightarrow \int \theta^{6-1} (1-\theta)^{5-1} = B(6, 5)$$

- Therefore we know that

$$p(\theta) = \frac{1}{B(6, 5)} \theta^{6-1} (1-\theta)^{5-1} = \text{Beta}(\theta|6, 5)$$

- We say that the *functional form* of a Beta density is $f(\theta) = \theta^{a-1} (1-\theta)^{b-1}$.

Deriving the analytical posterior for the Beta prior

- The beta distribution is a particular convenient choice for Binomial likelihoods

$$\underbrace{p(\theta|a_0, b_0)}_{\text{prior distribution}} = \frac{1}{B(a_0, b_0)} \theta^{a_0-1} (1-\theta)^{b_0-1}$$
$$\underbrace{p(y|\theta)}_{\text{likelihood}} = \binom{N}{y} \theta^y (1-\theta)^{N-y}$$

- Bayes rule states

$$p(\theta|y) = \frac{p(y|\theta)p(\theta)}{p(y)}$$
$$\propto p(y|\theta)p(\theta)$$
$$= \underbrace{\binom{N}{y} \theta^y (1-\theta)^{N-y}}_{\text{Binomial PMF}} \underbrace{\frac{1}{B(a_0, b_0)} \theta^{a_0-1} (1-\theta)^{b_0-1}}_{\text{Beta density}}$$
$$\propto \theta^y (1-\theta)^{N-y} \theta^{a_0-1} (1-\theta)^{b_0-1}$$
$$= \theta^{y+a_0-1} (1-\theta)^{N-y+b_0-1}$$
$$\propto \text{Beta}(\theta|y + a_0, N - y + b_0)$$

- Key take-away: The posterior distribution is another Beta distribution with parameters

$$a = a_0 + y$$

$$b = b_0 + N - y$$

- The Beta distribution is said to be *conjugate* to the binomial distribution because the posterior is of the same *functional form* as the prior

The posterior mean

- The *key equations* for the beta-binomial model

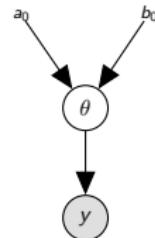
$$p(\theta) = \text{Beta}(\theta | a_0, b_0) \quad (\text{Prior})$$

$$p(y|\theta) = \binom{N}{y} \theta^y (1-\theta)^{N-y} \quad (\text{Likelihood})$$

$$p(\theta|y) = \text{Beta}(\theta | a_0 + y, b_0 + N - y) \quad (\text{Posterior})$$

- The posterior mean is a compromise between the prior mean and the maximum likelihood solution

$$\mathbb{E}[\theta|y] = \frac{a}{a+b} = \frac{a_0 + y}{a_0 + b_0 + N}$$



- We can interpret a_0 and b_0 as *pseudo observations* of prior successes and failures, respectively

Quick exercise

- The *key equations* for the beta-binomial model

$$p(\theta) = \text{Beta}(\theta|a_0, b_0) \quad (\text{Prior})$$

$$p(y|\theta) = \binom{N}{y} \theta^y (1-\theta)^{N-y} \quad (\text{Likelihood})$$

$$p(\theta|y) = \text{Beta}(\theta|a_0 + y, b_0 + N - y) \quad (\text{Posterior})$$

$$\mathbb{E}[\theta|y] = \frac{a_0 + y}{a_0 + b_0 + N} \quad (\text{Posterior mean})$$

Exercise

Assuming we have observed the following data $N = 20$ views and $y = 4$ click-rates and assume the prior is a Beta distribution with $a_0 = 2$ and $b_0 = 2$, compute ...

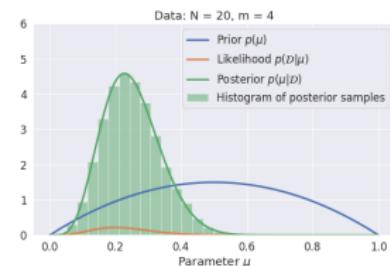
- the prior mean
- the parameters a and b for the posterior distribution
- the posterior mean

Exercise - follow up I

Calculating posterior summaries in practice

- Assume we have obtained our posterior of interest and that our goal is to estimate the *posterior mean* $\mathbb{E}_{p(\theta|y)} [\theta]$ and the *probability* $P(\theta < 0.15|y)$
- If we can generate *samples* from the posterior, then we can estimate the posterior mean using *Monte Carlo estimation*:

$$\mathbb{E}_{p(\theta|y)} [\theta] = \int \theta p(\theta|y) d\theta \approx \frac{1}{S} \sum_{i=1}^S \theta^{(i)}, \quad \theta^{(i)} \sim p(\theta|y)$$



- This works for any function of θ :

$$\mathbb{E}_{p(\theta|y)} [f(\theta)] = \int f(\theta) p(\theta|y) d\theta \approx \frac{1}{S} \sum_{i=1}^S f(\theta^{(i)}), \quad \theta^{(i)} \sim p(\theta|y)$$

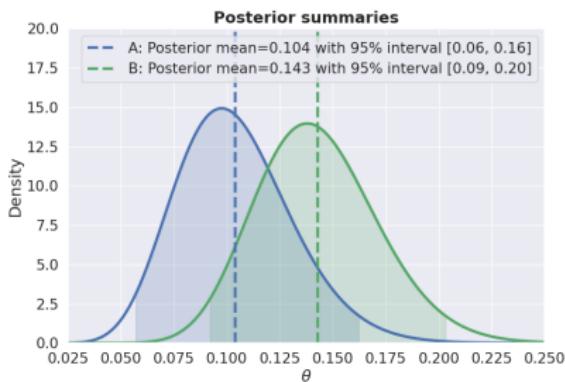
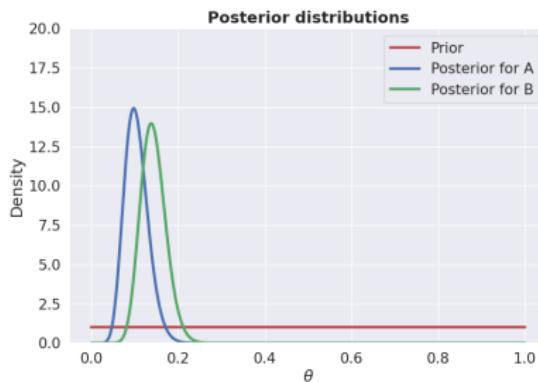
- We can estimate the probability by counting the fraction of samples below 0.15

$$P(\theta < 0.15|y) = \int_0^{0.15} p(\theta|y) d\theta = \int \mathbb{I}[\theta < 0.15] p(\theta|y) d\theta \approx \frac{1}{S} \sum_{i=1}^S \mathbb{I}[\theta^{(i)} < 0.15]$$

Posterior summaries for A/B testing example

- Ad A has been shown $N_A = 123$ times and generated $y_A = 12$ clicks, and Ad B has been shown $N_B = 145$ times and generated $y_B = 20$ clicks.
- We will use uniform Beta-priors with $a_0 = b_0 = 1$ for both
- We compute posterior distribution for each ad (using \mathcal{D} to denote observed data)

$$p(\theta_A | \mathcal{D}_A) = \text{Beta}(\theta_A | 13, 112) \quad p(\theta_B | \mathcal{D}_B) = \text{Beta}(\theta_B | 21, 126)$$



Yes, but which one is better? A or B?

- Let's introduce the difference of the click rates

$$\theta_D = \theta_B - \theta_A$$

- We compute θ_D for each pair of samples

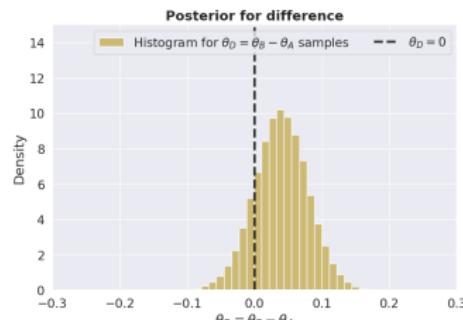
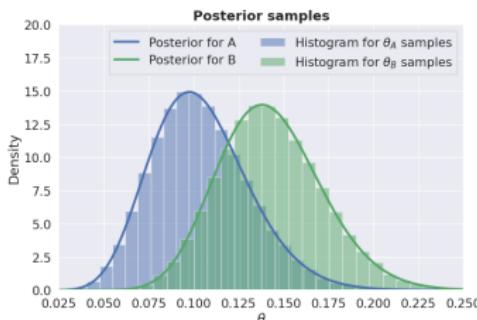
$$\theta_D^{(i)} = \theta_B^{(i)} - \theta_A^{(i)}$$

- Posterior probability that B is better than A

$$P(\theta_B > \theta_A | \mathcal{D}) = P(\theta_D > 0 | \mathcal{D}) \approx 0.85$$

- Calculating posterior mean and credibility interval from posterior samples $\theta_D^{(i)}$

$$\mathbb{E}[\theta_D | \mathcal{D}] = 0.039 \quad P(\theta_D \in [-0.038, 0.116] | \mathcal{D}) \approx 0.95$$



Main takeaways for today

1. Bayesian inference represents all variables using *probability distributions*

2. We update *from prior to posterior* belief using Bayes' rule

$$p(\theta|\mathcal{D}) = \frac{p(\mathcal{D}|\theta)p(\theta)}{p(\mathcal{D})}$$

3. The *posterior summarizes all information* about θ after we observed data

4. The Beta-binomial model is Bayesian approach for *estimating proportions* that uses the binomial distribution as likelihood and the beta distribution as prior

5. As we collect more and more data, the posterior distribution concentrates and becomes more and more independent of the prior

6. Distributions can be summarized using *modes, means, intervals, probabilities* etc. and these can be easily computed via *sampling*

Introduction to the exercise sessions

Intro to exercise

- On DTU Learn you will find an exercise for each week in notebook format
 - We will spend all 4 hours from 13-17 working with the exercises
 - Mix of pen&paper, programming and discussion questions
- The purpose of the exercise for this week is to get familiar with
 - Basic Bayesian terminology
 - The Beta-binomial model
 - Application to A/B testing
- The programming part is based on Python (which is a prerequisite)
 - Numpy, scipy, matplotlib, seaborn packages
 - We will use JAX for numerical computations
 - JAX is a state-of-the-art framework for machine learning with a numpy interface
 - See <https://jax.readthedocs.io/en/latest/quickstart.html>
- Feel free to collaborate with your peers
- Ask for help!
 - Ask for help when stuck
 - Use teachers/TAs to check your understanding
 - Engage in discussion to practice
- Feedback persons: Meet at 16:45

Binary variables and uncertainty quantification: example (bonus slide)

- We model the user click behavior as Bernoulli distributed with probability μ , where $x = 1$ means click and $x = 0$ means no-click such that

Click me!

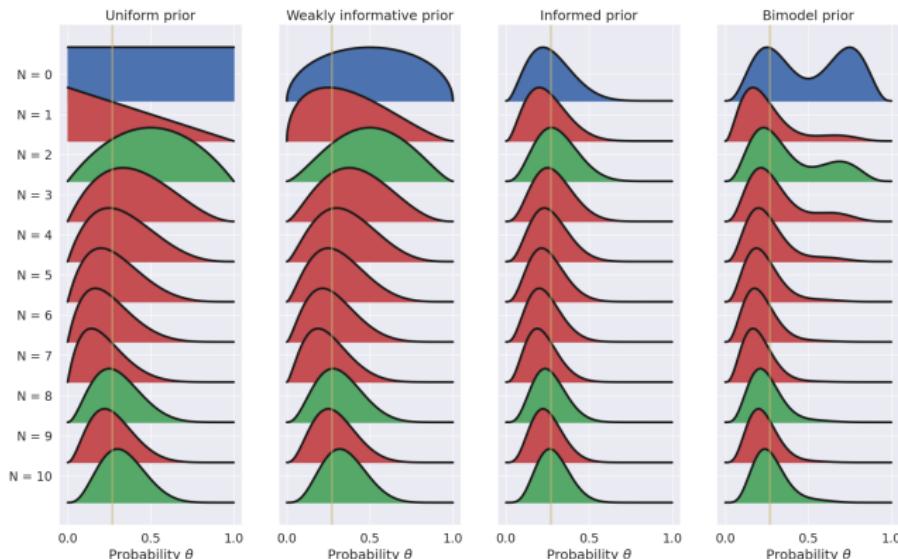
$$P(\text{click}|\mu) = P(x = 1|\mu) = \mu$$

- We know $0 \leq P(\text{click}|\mu) \leq 1$, but when are we most uncertain about the outcome?
 - $\mu = 1?$
 - $\mu = 0.5?$
 - $\mu = 0?$
- When $\mu = 1$, we are absolute sure that the user is going to click the ad, i.e. no uncertainty
- When $\mu = 0$, we are absolute sure that the user is *not* going to click the ad, i.e. no uncertainty
- When $\mu = 0.5$, we have maximum uncertainty because

$$P(\text{click}|\mu) = P(\text{not click}|\mu) = 0.5$$

The effect of the prior distribution (bonus slide)

- A Bayesian analysis starts with a *prior distribution* $p(\mu)$ representing our knowledge of μ **before** we observe any data



- The prior has a strong influence when the sample size is small, but its effect disappears when the sample size grows

02477 – Bayesian Machine Learning: Lecture 2

Michael Riis Andersen

Technical University of Denmark,
DTU Compute, Department of Applied Math and Computer Science

Outline

- 1 Quick re-cap of last week
- 2 Probabilistic machine learning
- 3 The plug-in approximation
- 4 Grid approximations for non-conjugate models
- 5 Introduction to exercise: towards logistic regression

Quick re-cap of last week

Quick re-cap of Beta-binomial model and what's next

- Bayes' rule provides a systematic way to combine data with prior knowledge

$$p(\theta|y) = \frac{p(y|\theta)p(\theta)}{p(y)}$$

- The beta-binomial model is a conjugate model

$$p(\theta) = \text{Beta}(\theta|a_0, b_0) \quad (\text{Prior})$$

$$p(y|\theta) = \binom{N}{y} \theta^y (1-\theta)^{N-y} \quad (\text{Likelihood})$$

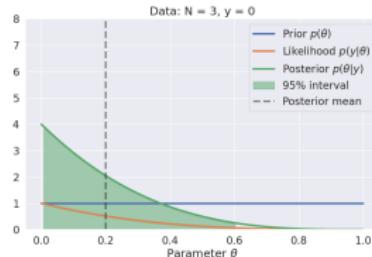
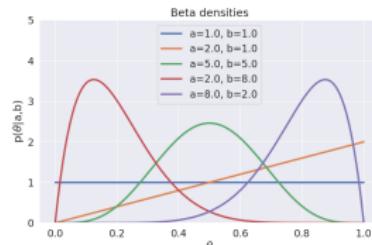
$$p(\theta|y) = \text{Beta}(\theta|y+a_0, N-y+b_0) \quad (\text{Posterior})$$

- Estimate θ using the mean of the posterior distribution

$$\theta_{\text{Bayes}} = \mathbb{E}[\theta|y] \equiv \int \theta p(\theta|y)d\theta$$

- ... and use credibility intervals of the posterior to quantify the uncertainty

$$P(\theta \in [0.01, 0.60] | y) = 0.95$$



What about making predictions?

Example continued: suppose we have this website ad with $N = 3$ views and $y = 0$ clicks.

- Using a *uniform prior*, i.e. $p(\theta) = \text{Beta}(\theta|1, 1)$

$$p(\theta) = \text{Beta}(\theta|1, 1) \quad (\text{Prior})$$

$$p(y|\theta) = \binom{3}{0} \theta^0 (1-\theta)^3 \quad (\text{Likelihood})$$

$$p(\theta|y) = \text{Beta}(\theta|1, 4) \quad (\text{Posterior})$$

- *Summarize* our knowledge using posterior

$$\mathbb{E}[\theta|y] = \frac{1}{5}, \quad P(\theta \in [0.01, 0.60] | y) \approx 0.95$$

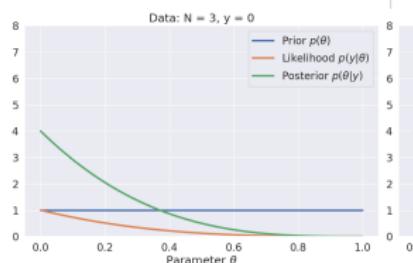
- **Goal:** *predict* number of clicks y^* in the next $N^* = 50$ views?

$$p(y^*|N^*, \theta) = \text{Bin}(y^*|N^*, \theta)$$

- Recall: mean of a Binomial random variable with prob. θ :

$$\mathbb{E}[y^*] = N^* \theta = 50\theta$$

Which value of θ to use? How do we use the *posterior knowledge* to make predictions?
How do we take the *uncertainty* into account?



Probabilistic machine learning

Probabilistic machine learning I

Product rule

$$p(a, b) = p(b|a)p(a)$$

Sum rule

$$p(b) = \int p(a, b)da$$

Conditional

$$p(a|b) = \frac{p(a,b)}{p(b)}$$

Conditional independence

$$p(a, b|c) = p(a|c)p(b|c)$$

- A probabilistic model is *completely specified* by its *joint distribution*
- Consider a model with two *random variables*: y (data) and θ (unknown parameter)
- The *joint distribution* of all random variables can be expressed via the *product rule*

$$p(\theta, y) = p(y|\theta)p(\theta)$$

- The *posterior distribution* can be obtained by *conditioning* on y

$$p(\theta|y) = \frac{p(y, \theta)}{p(y)} = \frac{p(y|\theta)p(\theta)}{p(y)}$$

- The *evidence* $p(y)$ can be obtained from the joint distribution via the *sum rule*

$$p(y) = \int p(y, \theta)d\theta = \int p(y|\theta)p(\theta)d\theta$$

- Hence, in theory, we can derive *all quantities of interest* from the *joint distribution*

Probabilistic machine learning II

Product rule

$$p(a, b) = p(b|a)p(a)$$

Sum rule

$$p(b) = \int p(a, b)da$$

Conditional

$$p(a|b) = \frac{p(a,b)}{p(b)}$$

Conditional independence

$$p(a, b|c) = p(a|c)p(b|c)$$

- A probabilistic model is *completely specified* by its *joint distribution*

$$p(\theta, y) = p(y|\theta)p(\theta)$$

- What if we have more than one observed variable, e.g. y_1 and y_2 ?
- For a *broad class of models* the likelihood can be further decomposed using *conditional independence*:

$$p(y_1, y_2|\theta) = p(y_1|\theta)p(y_2|\theta)$$

- ... or more generally for $\mathbf{y} = [y_1 \quad y_2 \quad \dots \quad y_N]$

$$p(\mathbf{y}|\theta) = p(y_1|\theta)p(y_2|\theta)\dots p(y_N|\theta) = \prod_{n=1}^N p(y_n|\theta)$$

- The *joint distribution* becomes

$$p(\theta, \mathbf{y}) = p(\mathbf{y}|\theta)p(\theta) = \prod_{n=1}^N p(y_n|\theta)p(\theta)$$

Website ad example continued: Making predictions

- **Example continued:** Your website ad has been shown $N = 123$ times and generated $y = 12$ clicks. Suppose you pay for another $N^* = 50$ views, how many clicks y^* should you expect *given the observed data*?
- Assuming each click can be modelled using *conditionally independent* Bernoulli trials with the *same probability* θ

$$p(y|\theta) = \text{Bin}(y|N, \theta)$$
$$p(y^*|\theta) = \text{Bin}(y^*|N^*, \theta)$$

- The assumption of *conditional independence* implies

$$p(y, y^*|\theta) = p(y|\theta)p(y^*|\theta) = \text{Bin}(y|N, \theta)\text{Bin}(y^*|N^*, \theta)$$

- Completing the model by *imposing a prior* for θ

$$p(\theta) = \text{Beta}(\theta|a_0, b_0)$$

- **Goal:** compute *predictive distribution* of y^* given we have observed $y = 12$, i.e. $p(y^*|y = 12)$.

A probabilistic perspective on making predictions

Product rule

$$p(a, b) = p(b|a)p(a)$$

Sum rule

$$p(b) = \int p(a, b)da$$

Conditional

$$p(a|b) = \frac{p(a,b)}{p(b)}$$

Conditional independence

$$p(a, b|c) = p(a|c)p(b|c)$$

Goal: Given some data y , what can we say about a new observation y^* ?

- Step 1: Formulate *joint distribution* for *all variables* of interests

$$p(y^*, y, \theta) = p(y^*, y|\theta)p(\theta) = p(y^*|\theta)p(y|\theta)p(\theta)$$

- Step 2: *Condition* on the *observed data* y

$$p(y^*, \theta|y) = \frac{p(y^*, y, \theta)}{p(y)} = \frac{p(y^*|\theta)p(y|\theta)p(\theta)}{p(y)}$$

- Step 3: *Marginalize* out parameter θ using the *sum rule* to get the *posterior predictive distribution*

$$p(y^*|y) = \int p(y^*, \theta|y)d\theta = \int \frac{p(y^*|\theta)p(y|\theta)p(\theta)}{p(y)}d\theta = \int p(y^*|\theta)p(\theta|y)d\theta = \mathbb{E}_{p(\theta|y)} [p(y^*|\theta)]$$

- **Key take-away:** To reason about y^* given y , we need to *average the likelihood* for y^* wrt. to the *posterior distribution* $p(\theta|y)$.

Quiz time

Take the quiz called
Lecture 2: Prior, likelihood, posterior, posterior predictive
to test your understanding.

Website example

- **Example continued:** Your website ad has been shown $N = 123$ times and generated $y = 12$ clicks. Suppose you pay for another $N^* = 50$ views, how many clicks y^* should you expect *given the observed data?*
- We already defined the model

$$\begin{aligned} p(y|\theta) &= \text{Bin}(y|N, \theta) && (\text{Likelihood}) \\ p(y^*|\theta) &= \text{Bin}(y^*|N^*, \theta) && (\text{Predictive likelihood}) \\ p(\theta) &= \text{Beta}(\theta|a_0, b_0) && (\text{Prior}) \end{aligned}$$

- We know how to compute the *posterior distribution*

$$p(\theta|y) = \text{Beta}(\theta|y + a_0, N - y + b_0)$$

- Next, we want to compute the *posterior predictive distribution*

$$p(y^*|y) = \int p(y^*|\theta)p(\theta|y)d\theta = \int \text{Bin}(y^*|N^*, \theta)\text{Beta}(\theta|y + a_0, N - y + b_0)d\theta$$

- *Intuition:* Instead of plugging in a single value for the parameter estimate, we plug in all possible values for θ and weight the result according to $p(\theta|y)$

Website example

- Compute *posterior predictive distribution*

$$\begin{aligned} p(y^* = k|y) &= \int \text{Bin}(y = k|N^*, \theta) \text{Beta}(\theta|y + a_0, N - y + b_0) d\theta \\ &= \int \binom{N^*}{k} \theta^k (1-\theta)^{N^*-k} \frac{1}{B(\alpha, \beta)} \theta^{\alpha-1} (1-\theta)^{\beta-1} d\theta \\ &= \binom{N^*}{k} \frac{1}{B(\alpha, \beta)} \int \theta^k (1-\theta)^{N^*-k} \theta^{\alpha-1} (1-\theta)^{\beta-1} d\theta \quad (\text{Linearity}) \\ &= \binom{N^*}{k} \frac{1}{B(\alpha, \beta)} \int \theta^{k+\alpha-1} (1-\theta)^{\beta+N^*-k-1} d\theta \quad (\text{Simplify}) \\ &= \binom{N^*}{k} \frac{1}{B(\alpha, \beta)} \int \theta^{k+\alpha-1} (1-\theta)^{\beta+N^*-k-1} d\theta \end{aligned}$$

- We recognize the terms in green as the *functional form* of a Beta density, and hence, we know how to compute the integral

$$p(y^* = k|y) = \binom{N^*}{k} \frac{B(\alpha + k, \beta + N^* - k)}{B(\alpha, \beta)}$$

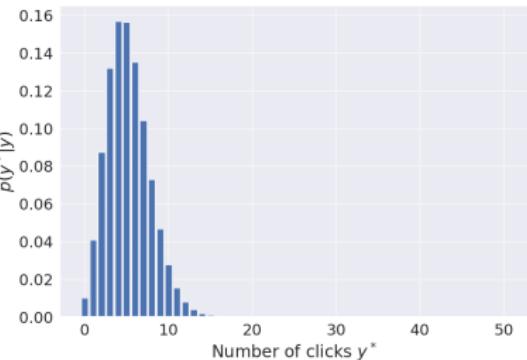
for $\alpha = y + a_0$ and $\beta = N - y + b_0$.

Website example: putting everything together

Example continued: Your website ad has been shown $N = 123$ times and generated $y = 12$ clicks. Suppose you pay for another $N^* = 50$ views, how many clicks y^* should you expect *given the observed data?*

$$\begin{aligned} p(\theta) &= \text{Beta}(\theta|1, 1) \\ p(y|\theta) &= \text{Bin}(y|123, \theta) \\ p(y^*|\theta) &= \text{Bin}(y^*|50, \theta) \\ p(\theta|y) &= \text{Beta}(\theta|13, 112) \end{aligned}$$

(*Prior*)
(*Likelihood*)
(*Predictive likelihood*)
(*Posterior*)



- Distribution of clicks y^* based on views $N^* = 50$ views

$$p(y^* = k|y) = \binom{N^*}{k} \frac{B(\alpha + k, \beta + N^* - k)}{B(\alpha, \beta)} = \binom{50}{k} \frac{B(13 + k, 162 - k)}{B(13, 112)}$$

- The expected number of clicks given the data is

$$\mathbb{E}_{p(y^*|y)} [y^*] = \sum_{k=0}^{50} kp(y^* = k|y) \approx 5.2$$

The plug-in approximation

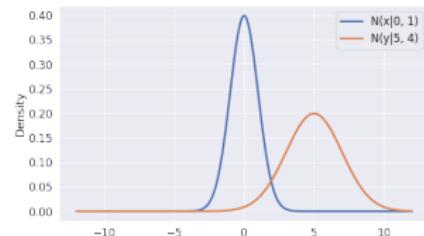
A few prerequisites first

Univariate Gaussians

- The *normal distribution* (also known as the Gaussian) is distribution over $x \in \mathbb{R}$ with density

$$\mathcal{N}(x|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

- Two parameters: $\mu \equiv \mathbb{E}[x]$ and $\sigma^2 \equiv \mathbb{V}[x]$
- Widely popular due to Central limit theorems, maximum entropy principle, relation to least squares minimization, nice mathematical properties
- We will talk more about Gaussians later in this course



A few prerequisites first

Dirac's delta function

- Consider a Gaussian random variable $x \sim \mathcal{N}(\mu, \sigma^2)$.
In the *limit* $\sigma^2 \rightarrow 0$, x is effectively a *constant* $x = \mu$:

- We say that x follows a *Dirac's delta distribution* centered at μ

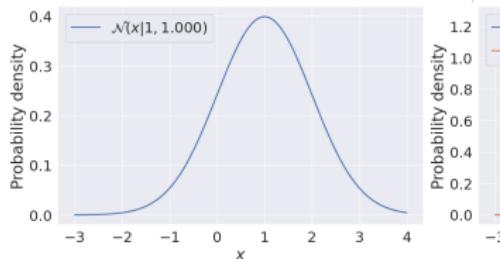
$$p(x) = \lim_{\sigma^2 \rightarrow 0} \mathcal{N}(x|\mu, \sigma^2) = \delta(x - \mu)$$

- Important properties

$$\delta(x - \mu) = \begin{cases} \infty & \text{if } x = \mu \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

$$\int \delta(x - \mu) dx = 1 \quad (2)$$

$$\int f(x) \delta(x - \mu) dx = f(\mu) \quad (3)$$



- Eq. (3) is called the *sifting property* and implies

$$\mathbb{E}_{\delta(x-\mu)} [f(x)] = f(\mu)$$

The plugin approximation

- We showed that the rules of probability theory dictates that

$$p(y^*|y) = \int p(y^*|\theta)p(\theta|y)d\theta$$

- While this is *optimal* given the model, it can be *non-trivial* in practice
- If we *assume* that there is a *single best parameter* $\hat{\theta}$, e.g. $\hat{\theta}_{\text{MLE}}$ or $\hat{\theta}_{\text{MAP}}$, then we can approximate $p(\theta|y)$ using a *Dirac's delta function* $\delta(\theta - \hat{\theta})$

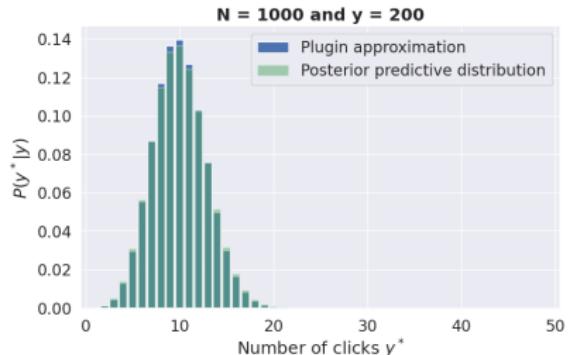
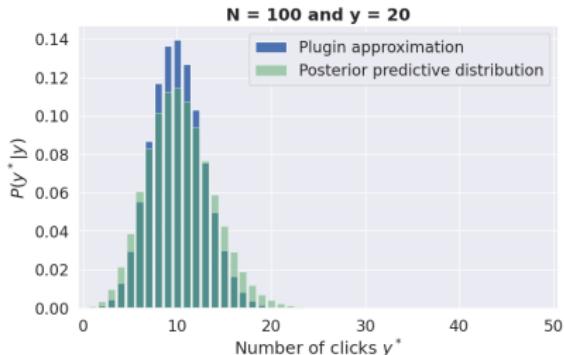
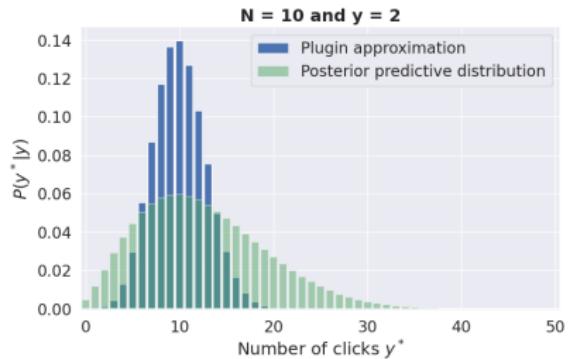
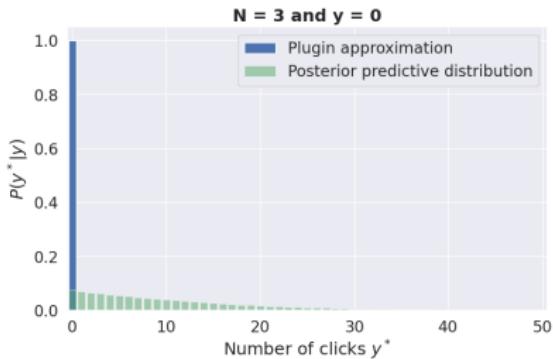
$$p(\theta|y) \approx \delta(\theta - \hat{\theta})$$

- Using the *sifting property* of Dirac's delta implies

$$p(y^*|y) = \int p(y^*|\theta)p(\theta|y)d\theta \approx \int p(y^*|\theta)\delta(\theta - \hat{\theta})d\theta = p(y^*|\hat{\theta})$$

- Therefore, this is called a *plug-in approximation*.
- Very *easy to compute*, but *ignores uncertainty* for our estimate of θ , and hence, often *producing overconfident predictions*.
- This is how we make predictions in deep learning...

The posterior predictive distribution and plugin approximations



Grid approximations for non-conjugate models

Introduction to non-conjugate models

Big picture so far...

- We studied the binomial model for estimating proportions and imposed a Beta prior for θ for Bayesian inference
- We derived the *posterior* and *posterior predictive* distributions *analytically*. This is possible due to *conjugacy* of the Beta prior and binomial likelihood
- Supposed our analysis required computing the posterior mean for a *different prior*, e.g. $p(\theta) = \frac{1}{Z} e^{\sin(\pi\theta^2)}$

$$\mathbb{E}[\theta|y] = \int \theta p(\theta|y) d\theta = \int \theta \frac{p(y|\theta)p(\theta)}{p(y)} d\theta$$

- To compute the posterior mean, variance etc. we need the evidence $p(y)$

$$\begin{aligned} p(y) &= \int p(\theta|y)p(\theta)d\theta = \int \text{Bin}(y|N, \theta) \frac{1}{Z} e^{\sin(\pi\theta^2)} d\theta \\ &= \int \binom{N}{y} \theta^y (1-\theta)^{N-y} \frac{1}{Z} e^{\sin(\pi\theta^2)} d\theta \\ &= \binom{N}{y} \frac{1}{Z} \int \theta^y (1-\theta)^{N-y} e^{\sin(\pi\theta^2)} d\theta = ? \end{aligned}$$

- Unfortunately, we *cannot* evaluate the evidence, i.e. $p(y)$ analytically *intractable* for most models of practical interest...

Approximate inference methods

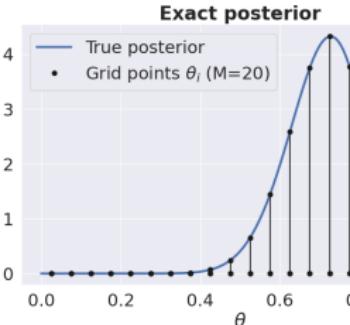
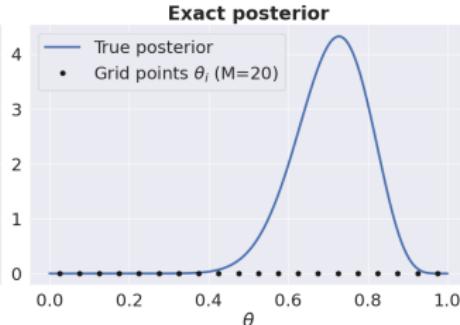
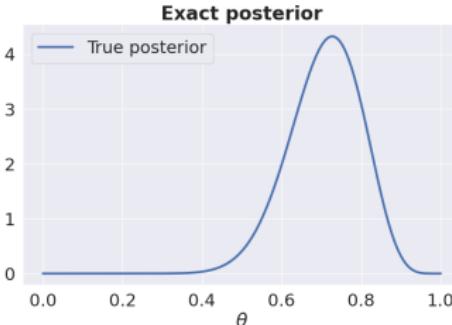
- **In this course:** We will study several computational tools and approximation inference methods for dealing with such *intractable distributions*
 1. Grid approximations
 2. Laplace approximations
 3. Variational inference
 4. Markov Chain Monte Carlo methods
- **Goal for these methods:** Compute *tractable approximation* $q(\theta)$ of true posterior $p(\theta|y)$ such that
 1. $q(\theta)$ resembles the true posterior, i.e. $p(\theta|y) \approx q(\theta)$
 2. $q(\theta)$ should be tractable s.t. we can compute posterior summaries, predictions etc.
- **This week:** We will focus on *grid approximations*, which are easy to understand and apply, and they will help build our intuition about marginalization.

The grid approximation

■ Constructing the grid approximation for $p(\theta|y)$

1. We define a set of grid points for θ : $0 \leq \theta_1 < \theta_2 < \dots < \theta_M \leq 1$
2. We evaluate the exact posterior (up to a constant) at all the grid points, i.e.
$$\tilde{\pi}_i \propto p(\theta_i|y) \propto p(y|\theta_i)p(\theta_i)$$
3. Sum all values to get normalization constant $Z = \sum_{i=1}^M \tilde{\pi}_i$
4. Compute normalized probabilities $\pi_i = \frac{1}{Z} \tilde{\pi}_i$ to get the grid approximation

$$q(\theta) = \sum_{i=1}^M \pi_i \delta(\theta - \theta_i)$$



The grid approximation

Posterior summaries

- The grid approximation is a discrete distribution, so computing summaries is easy, e.g the posterior mean

$$\mathbb{E}_{p(\theta|y)} [\theta] \approx \mathbb{E}_{q(\theta)} [\theta] = \sum_{i=1}^M \theta_i \pi_i$$

- ... and general expectations of $f(\theta)$

$$\mathbb{E}_{p(\theta|y)} [f(\theta)] \approx \mathbb{E}_{q(\theta)} [f(\theta)] = \sum_{i=1}^M f(\theta_i) \pi_i$$

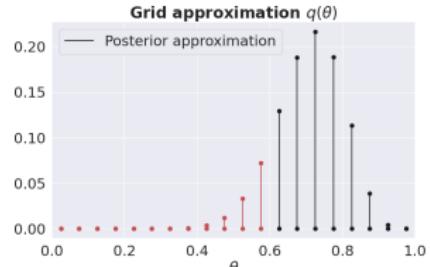
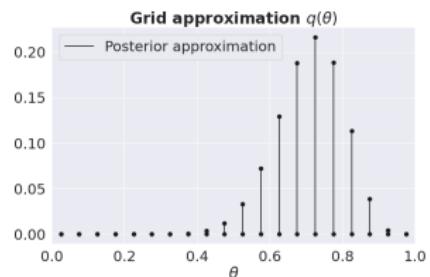
- Example: Computing post. probabilities for $\theta < 0.6$

$$p(\theta < 0.6|y) \approx q(\theta < 0.6)$$

$$= \sum_{i=1}^M \mathbb{I} [\theta_i < 0.6] \pi_i$$

$$= \sum_{i=1}^j \pi_i, \quad j = \max \{ i | \theta_i < 0.6 \}$$

$$q(\theta) = \sum_{i=1}^M \pi_i \delta(\theta - \theta_i)$$



The grid approximation

The posterior predictive distribution

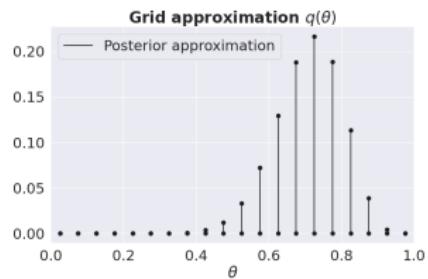
- General expectations of $f(\theta)$

$$\mathbb{E}_{p(\theta|y)} [f(\theta)] \approx \mathbb{E}_{q(\theta)} [f(\theta)] = \sum_{i=1}^M f(\theta_i) \pi_i$$

$$q(\theta) = \sum_{i=1}^M \pi_i \delta(\theta - \theta_i)$$

- The posterior predictive distribution $p(y^*|y)$

$$p(y^* = k|y) = \mathbb{E}_{p(\theta|y)} [\text{Bin}(y^* = k|N^*, \theta)]$$



- Hence, setting $f(\theta) = \text{Bin}(y^* = k|N^*, \theta_i)$ yields

$$p(y^* = k|y) \approx \sum_{i=1}^M \text{Bin}(y^* = k|N^*, \theta_i) \pi_i$$

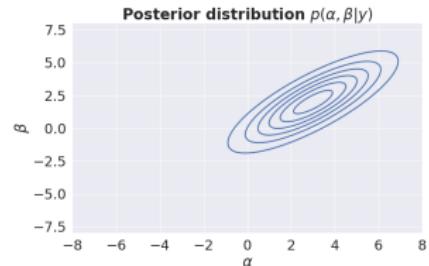
- To make predictions we literally compute a weighted sum of all possible parameter values

The grid approximation

A few practical considerations

■ Choosing the grid range

- Range for $\theta \in [0, 1]$ is easy
- For a different parameter $\alpha \in \mathbb{R}$, we need to choose an interval $[a, b]$ for the grid. Often identified visually.



■ Scaling with model dimensionality

- Suppose we use $M = 20$ points for each dimension

1D: 20 evaluations

2D: $20^2 = 400$ evaluations

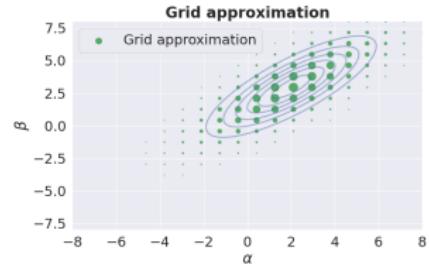
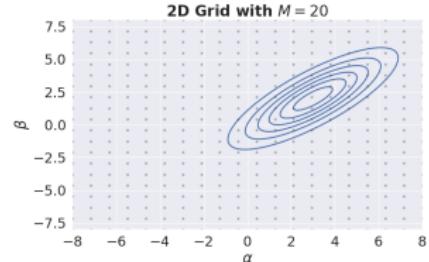
3D: $20^3 = 8000$ evaluations

4D: $16000^4 = 160000$ evaluations

- Grid approximations do not scale well beyond 3-4 dimensions

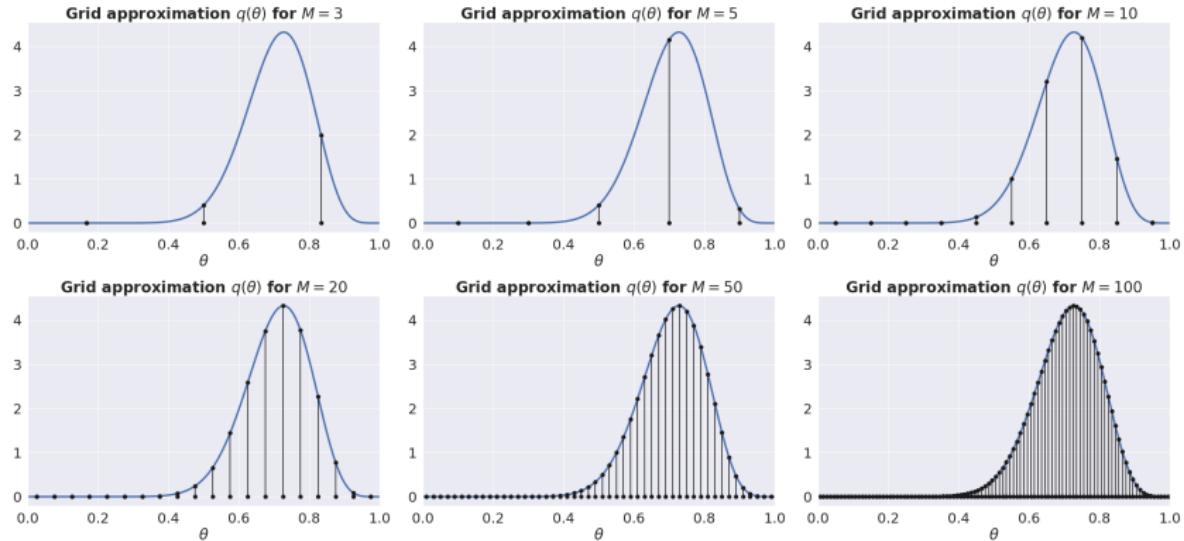
■ Number of grid points M

- M is balance between computational cost and accuracy
- Grid approximation is zero when evaluated outside the grid points
- Often diminishing returns as M increases (next slide)



The grid approximation

Summary



- **Pros:** Simple, easy and intuitive.
- **Cons:** Suffers from curse of dimensionality and does not scale beyond 3-4 dimensions

Introduction to exercise: towards logistic regression

Towards logistic regression

- So far we focussed on modelling *proportions*, i.e. $\theta \in [0, 1]$ given data about y successes in N *conditionally independent trials*
- The binomial likelihood is also often used in *dose-response* models, which is key for determining "safe" dosages for drugs, pollution, foods etc.
- **Example:** A company wants to study side effects of their new drug

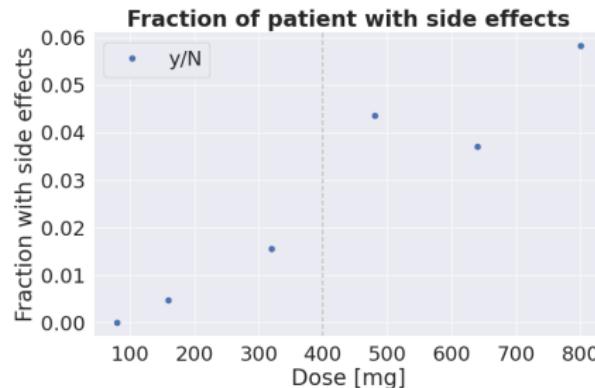
x (Dose in mg)	y (# side effects)	N (# patients)
80	0	69
160	4	832
320	13	835
480	20	459
640	12	324
800	6	103

- We could analyze the data for each dose independently using a beta-binomial model, but we would like to ...
 1. understand how dose affect the probability of side effects
 2. make predictions for new dosages x^*
 3. borrow "statistical strength" across dosages

Towards logistic regression

Example & motivation II

x (Dose in mg)	y (# side effects)	N (# patients)	y/N
80	0	69	0
160	4	832	0.005
320	13	835	0.016
480	20	459	0.044
640	12	324	0.037
800	6	103	0.058



- How accurate can we predict the probability of side effects for $x^* = 400\text{mg}$?

Towards logistic regression

Setting up the likelihood

- For each dose x_i , we assume

$$y_i|x_i \sim \text{Bin}(y_i|N_i, \theta_i), \quad \theta_i \in [0, 1]$$

- We model the probability θ_i as function of the dose x_i , i.e.

$$\theta_i \equiv \theta(x_i) = \sigma(\alpha + \beta x_i),$$

where $\sigma(x) : \mathbb{R} \rightarrow [0, 1]$ is a sigmoid function and $\alpha, \beta \in \mathbb{R}$ are model parameters.

- The likelihood of the i 'th observation (x_i, y_i)

$$p(y_i|x_i, \alpha, \beta) = \text{Bin}(y_i|N_i, \theta_i)$$

- Assuming conditional independence we can write the joint likelihood

$$p(\mathbf{y}|\mathbf{x}, \alpha, \beta) = \prod_{i=1}^M p(y_i|x_i, \alpha, \beta) = \prod_{i=1}^M \text{Bin}(y_i|N_i, \theta_i),$$

where $\mathbf{y} = [y_1, y_2, \dots, y_6]$ and similar for $\mathbf{x} = [x_1, x_2, \dots, x_6]$

- The predictive likelihood for y^* is

$$p(y^*|x^*, \alpha, \beta) = \text{Bin}(y^*|N_i, \theta^*)$$

where $\theta^* \equiv \theta(x^*)$

Towards logistic regression

Setting up the prior

- The model parameters are α (intercept) and β (slope) of the generalized linear model
- Prior information: we have no prior information about the sign of the parameters. Hence, we choose a zero-mean Gaussian distributions

$$p(\alpha, \beta) = \mathcal{N}(\alpha|0, \sigma_\alpha^2)\mathcal{N}(\beta|0, \sigma_\beta^2), \quad \sigma_\alpha^2, \sigma_\beta^2 > 0$$

- We can now write the *joint distribution* of $\alpha, \beta, \mathbf{y}, y^*$ using the *product rule*

$$p(\mathbf{y}, y^*, \alpha, \beta | \mathbf{x}, x^*) = p(y^* | x^*, \alpha, \beta)p(\mathbf{y} | \mathbf{x}, \alpha, \beta)p(\alpha, \beta)$$

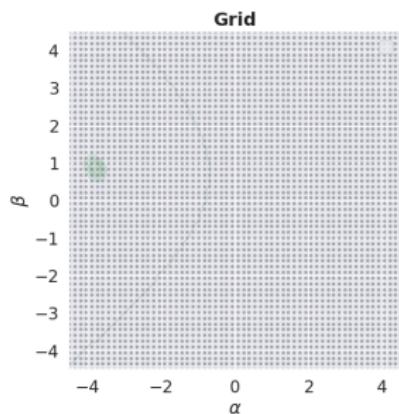
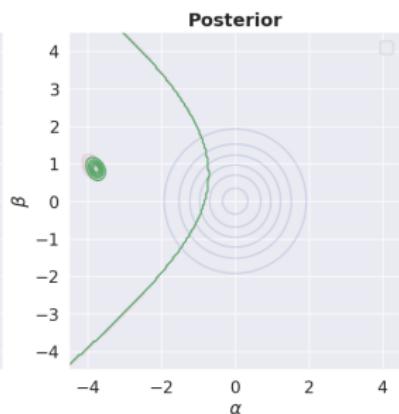
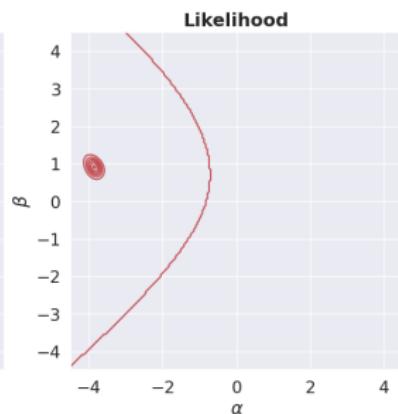
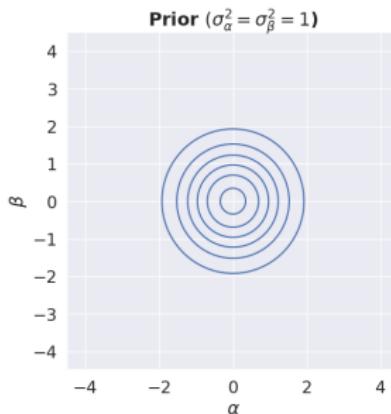
- The *posterior predictive distribution* is by *conditioning on \mathbf{y}* and *marginalizing out the parameters* via the sum rule

$$p(y^* | \mathbf{y}, \mathbf{x}, x^*) = \iint p(y^*, \alpha, \beta | \mathbf{y}, \mathbf{x}, x^*) d\alpha d\beta = \underbrace{\iint}_{\text{likelihood for } y^*} \underbrace{\frac{p(y^* | x^*, \alpha, \beta)}{p(\alpha, \beta | \mathbf{y}, \mathbf{x})}}_{\text{posterior distribution}} d\alpha d\beta$$

- After obtaining the posterior of α, β , we can *propagate* the posterior uncertainty of the parameters to any quantity that depends on α, β , i.e. $\theta(x) = \sigma(\alpha + \beta x)$, the fraction of people with side effects y^*/N etc.

Towards logistic regression

Visualizing the distributions



Towards logistic regression

Making predictions

Computing the posterior predictive distribution $p(y^* = k|\mathbf{y}, \mathbf{x}, x^*)$ for $x^* = 400\text{mg}$ and $N^* = 500$

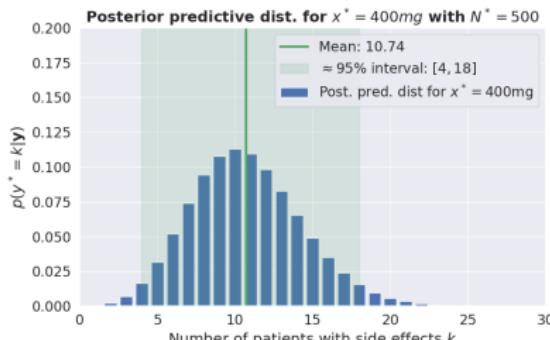
$$p(y^* = k|\mathbf{y}, \mathbf{x}, x^*) = \iint \underbrace{p(y^* = k|x^*, \alpha, \beta)}_{\text{likelihood for } y^*} \underbrace{p(\alpha, \beta|\mathbf{y}, \mathbf{x})}_{\text{posterior distribution}} d\alpha d\beta \quad (\text{Sum rule})$$

$$= \mathbb{E}_{p(\alpha, \beta|\mathbf{y}, \mathbf{x}, x^*)} [p(y^* = k|x^*, \alpha, \beta)] \quad (\text{Integrals as expectation})$$

$$= \mathbb{E}_{p(\alpha, \beta|\mathbf{y}, \mathbf{x}, x^*)} [\text{Bin}(y^*|N^*, \theta^*)] \quad (\text{Inserting dist.})$$

$$\approx \mathbb{E}_{q(\alpha, \beta)} [\text{Bin}(y^*|N^*, \theta^*)] \quad (\text{Grid approx.})$$

$$= \sum_{i,j} \text{Bin}(y^*|N^*, \sigma(\alpha_i + \beta_j x^*)) \pi_{ij}$$



Intro to exercise

- On DTU Learn you will find an exercise for each week in notebook format
- We will spend all 4 hours from 13-17 working with the exercises
- In this exercise you will
 - Dive deeper into the Bayesian framework
 - Study and implement the probabilistic model for logistic regression for the Challenger Distaster dataset
 - Study and implement the grid approximations
 - Practice probabilistic reasoning
- Mix of pen&paper, programming and discussion questions
- Feel free to collaborate with your peers
- Ask for help!
 - Ask for help when stuck
 - Use teachers/TAs to check your understanding
 - Engage in discussion to practice
- Feedback persons: Meet at 16:45

02477 – Bayesian Machine Learning: Lecture 3

Michael Riis Andersen

Technical University of Denmark,
DTU Compute, Department of Applied Math and Computer Science

Outline

- 1 Recap: Probabilistic machine learning
- 2 Recap: Multivariate Gaussian distributions
- 3 Recap: Linear regression and supervised learning
- 4 Bayesian linear regression
- 5 The posterior predictive distribution
- 6 Dealing with hyperparameters

Recap: Probabilistic machine learning

A probabilistic perspective on making predictions

Product rule

$$p(a, b) = p(b|a)p(a)$$

Sum rule

$$p(b) = \int p(a, b)da$$

Conditional

$$p(a|b) = \frac{p(a,b)}{p(b)}$$

Conditional independence

$$p(a, b|c) = p(a|c)p(b|c)$$

Goal: Given some data y , what can we say about a new observation y^* ?

- Step 1: Formulate *joint distribution* for *all variables* of interests

$$p(y^*, y, \theta) = p(y^*, y|\theta)p(\theta) = p(y^*|\theta)p(y|\theta)p(\theta)$$

- Step 2: *Condition* on the *observed data* y

$$p(y^*, \theta|y) = \frac{p(y^*, y, \theta)}{p(y)} = \frac{p(y^*|\theta)p(y|\theta)p(\theta)}{p(y)}$$

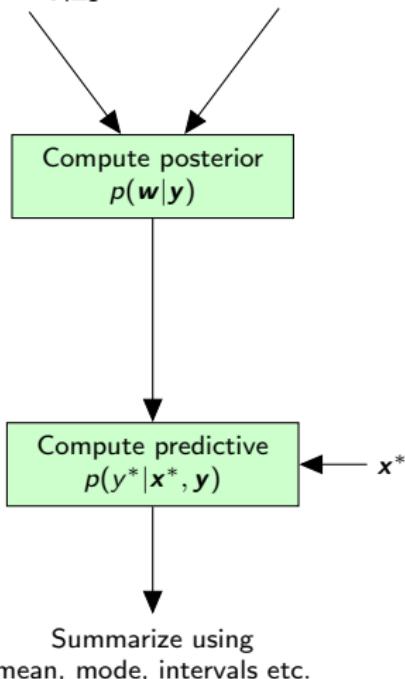
- Step 3: *Marginalize* out parameter θ using the *sum rule* to get the *posterior predictive distribution*

$$p(y^*|y) = \int p(y^*, \theta|y)d\theta = \int \frac{p(y^*|\theta)p(y|\theta)p(\theta)}{p(y)}d\theta = \int p(y^*|\theta)p(\theta|y)d\theta = \mathbb{E}_{p(\theta|y)} [p(y^*|\theta)]$$

- **Key take-away:** To reason about y^* given y , we need to *average the likelihood* for y^* wrt. to the *posterior distribution* $p(\theta|y)$.

Bayesian inference for supervised learning

$$\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=1}^N \quad p(\mathbf{w}, \mathbf{y}) = p(\mathbf{y}|\mathbf{w})p(\mathbf{w})$$



- Same principles for linear regression, logistic regression, neural networks etc. etc.

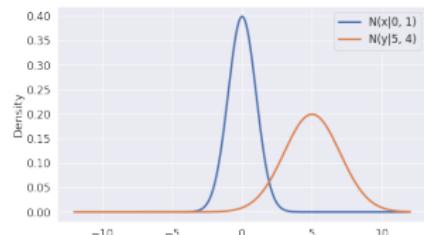
Recap: Multivariate Gaussian distributions

Univariate normal distribution

- The *normal distribution* (also known as the Gaussian)

$$\mathcal{N}(x|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

- Two parameters: $\mu = \mathbb{E}[x]$ and $\sigma^2 = \mathbb{V}[x]$
- Widely due to Central limit theorems, maximum entropy principle, relation to least squares minimization, nice mathematical properties



- Closed under affine transformations

$$x \sim \mathcal{N}(m, v) \quad \Rightarrow \quad a + bx \sim \mathcal{N}(a + bm, b^2v)$$

$$x \sim \mathcal{N}(0, 1)$$

$$y = 5 + 2x$$

- Let $x \sim \mathcal{N}(m_x, v_x)$ and $y \sim \mathcal{N}(m_y, v_y)$ for $x \perp y$, then

$$x + y \sim \mathcal{N}(m_x + m_y, v_x + v_y)$$

- Functional form: The *logarithm of a Gaussian density* is a *second order polynomial*

$$\ln \mathcal{N}(x|\mu, \sigma^2) = -\frac{1}{2\sigma^2}x^2 + \frac{\mu}{\sigma^2}x + K$$

The functional form of a Gaussian distribution I

- Recall we discussed the *functional form* of a Beta distribution
- Let's derive the functional form of the Gaussian

$$\ln \mathcal{N}(x|\mu, \sigma^2) = \ln \left[\frac{1}{\sqrt{2\pi\sigma^2}} \exp \left(-\frac{(x-\mu)^2}{2\sigma^2} \right) \right]$$

The functional form of a Gaussian distribution II: example

- Take-away: the *functional form* of a univariate Gaussian is

$$\ln \mathcal{N}(x|\mu, \sigma^2) = -\frac{1}{2\sigma^2}x^2 + \frac{\mu}{\sigma^2}x + K$$

- Example: suppose we are given the following log density for a random variable x

$$\ln p(x) = -\frac{1}{4}x^2 + 2x + K$$

- We recognize the 2nd order polynomial and conclude that $p(x)$ must be Gaussian
- We determine the *variance* by matching the coefficient for 2nd order term

$$-\frac{1}{2\sigma^2} = -\frac{1}{4} \quad \Rightarrow \quad \frac{1}{\sigma^2} = \frac{1}{2} \quad \Rightarrow \quad \sigma^2 = 2$$

- We determine the *mean* by matching coefficient for 1st order term

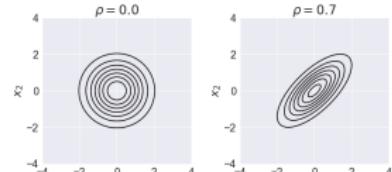
$$\frac{\mu}{\sigma^2} = 2 \quad \Rightarrow \quad \mu = 4$$

- Therefore, we conclude $p(x) = \mathcal{N}(x|4, 2)$.

The multivariate normal distribution

- The *multivariate normal distribution*

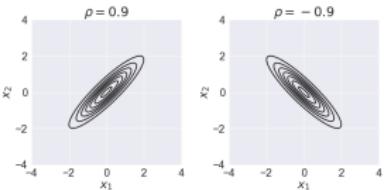
$$p(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = (2\pi)^{-\frac{D}{2}} |\boldsymbol{\Sigma}|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu})\right)$$



- Two parameters: $\boldsymbol{\mu} = \mathbb{E}[\mathbf{x}]$ and $\boldsymbol{\Sigma} = \text{cov}[\mathbf{x}]$

- Covariance matrix for $D = 2$

$$\boldsymbol{\Sigma} = \begin{bmatrix} a & c \\ c & b \end{bmatrix}$$

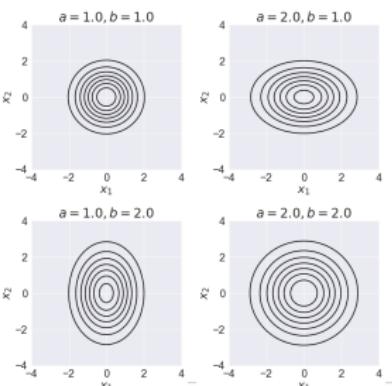


- Correlation coefficient $\rho = \frac{c}{\sqrt{ab}}$

- Let $\mathbf{x} \sim \mathcal{N}(\boldsymbol{m}_x, \boldsymbol{V}_x)$ and $\mathbf{y} \sim \mathcal{N}(\boldsymbol{m}_y, \boldsymbol{V}_y)$ for $\mathbf{x} \perp \mathbf{y}$, then

$$\mathbf{a} + \mathbf{B}\mathbf{x} \sim \mathcal{N}(\mathbf{a} + \mathbf{B}\boldsymbol{m}_x, \mathbf{B}\boldsymbol{V}_x \mathbf{B}^T)$$

$$\mathbf{x} + \mathbf{y} \sim \mathcal{N}(\boldsymbol{m}_x + \boldsymbol{m}_y, \boldsymbol{V}_x + \boldsymbol{V}_y)$$



The functional form of multivariate Gaussians

- Consider now the log density, focussing only on terms dependent on x .

$$\begin{aligned}\ln p(x|\mu, \Sigma) &= \ln \left[(2\pi)^{-\frac{D}{2}} |\Sigma|^{-\frac{1}{2}} \exp \left(-\frac{1}{2}(x - \mu)^T \Sigma^{-1} (x - \mu) \right) \right] \\ &= -\frac{D}{2} \ln(2\pi) - \frac{1}{2} \ln |\Sigma| - \frac{1}{2}(x - \mu)^T \Sigma^{-1} (x - \mu) \\ &= -\frac{1}{2}(x - \mu)^T \Sigma^{-1} (x - \mu) + \text{constant} \\ &= -\frac{1}{2}(x^T \Sigma^{-1} x - 2\mu^T \Sigma^{-1} x + \mu^T \Sigma^{-1} \mu) + \text{constant} \\ &= -\frac{1}{2}x^T \Sigma^{-1} x + \mu^T \Sigma^{-1} x + \text{constant}\end{aligned}$$

- Key take-aways

- Every time we encounter a distribution with a *quadratic log density*, it must be a Gaussian distribution (if Σ is a valid covariance matrix)
- We can *match coefficients* of first and second order term to *determine mean and covariance*

Recap: Linear regression and supervised learning

Supervised learning: linear regression

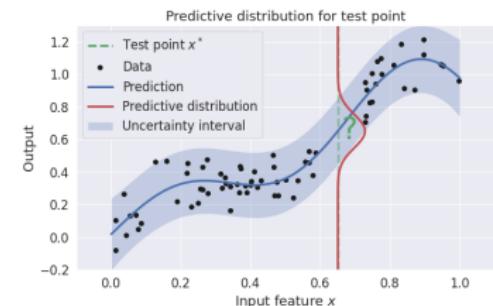
- Dataset $\mathcal{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_N, y_N)\}$
 - Input features: $\mathbf{x}_i \in \mathbb{R}^D$
 - Targets: $y_i \in \mathbb{R}$

- Additive noise models

$$y_i = f(\mathbf{x}_i | \mathbf{w}) + \epsilon_i$$

- Linear models are *linear wrt. parameters*, not data!

$$f(\mathbf{x} | \mathbf{w}) = w_0 + w_1 x_1 + \dots w_M x_M = \mathbf{w}^T \mathbf{x}$$



- *Non-linear* feature extractors $\phi(\cdot)$ (basis functions)

$$f(\mathbf{x} | \mathbf{w}) = \sum_{j=0}^M w_j \phi_j(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x})$$

Linear regression: the probabilistic model

- The *predictive distribution* of y^* given \mathbf{x}^* and the data \mathcal{D} is our goal

$$p(y^* | \mathcal{D}, \mathbf{x}^*)$$

- Model for the "signal"

$$f(\mathbf{x}_i | \mathbf{w}) = \mathbf{w}^T \phi(\mathbf{x}_i)$$

- The gaussian noise ϵ_i is assumed to be *independent and identically distributed (i.i.d.)*

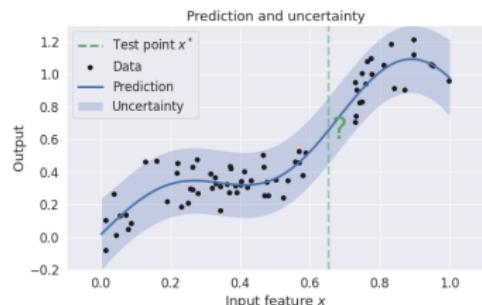
$$y_i = f(\mathbf{x}_i | \mathbf{w}) + \epsilon_i, \quad \epsilon_i \sim \mathcal{N}(0, \sigma^2)$$

- The *likelihood* for the i 'th data point

$$p(y_i | \mathbf{x}_i, \mathbf{w}, \sigma^2) = \mathcal{N}(y_i | \mathbf{w}^T \phi(\mathbf{x}_i), \sigma^2)$$

- Using the maximum likelihood solution as a *plug-in* estimator

$$p(y^* | \mathcal{D}, \mathbf{x}^*) = \mathcal{N}(y^* | \hat{\mathbf{w}}_{\text{MLE}}^T \phi(\mathbf{x}^*), \sigma^2)$$



Estimating the parameters using maximum likelihood

- Given a dataset $\mathcal{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_N, y_N)\}$, the *likelihood* for dataset is

$$p(\mathbf{y}|\mathbf{w}, \sigma^2) = \prod_{n=1}^N \mathcal{N}(y_n | f(\mathbf{x}_n | \mathbf{w}), \sigma^2)$$

- Taking the logarithm and using $f(\mathbf{x}_n, \mathbf{w}) = \mathbf{w}^T \phi(\mathbf{x}_n)$

$$\begin{aligned}\ln p(\mathbf{y}|\mathbf{w}, \sigma^2) &= \sum_{n=1}^N \ln \mathcal{N}(y_n | \mathbf{w}^T \phi(\mathbf{x}_n), \sigma^2) \\ &= \sum_{n=1}^N \left[-\frac{1}{2} \ln(2\pi\sigma^2) - \frac{1}{2\sigma^2} (y_n - \mathbf{w}^T \phi(\mathbf{x}_n))^2 \right] \\ &= -\frac{N}{2} \ln(2\pi\sigma^2) - \frac{1}{2\sigma^2} \sum_{n=1}^N (y_n - \mathbf{w}^T \phi(\mathbf{x}_n))^2\end{aligned}$$

- Maximum likelihood estimator $\hat{\mathbf{w}}_{MLE}$ is equivalent to minimizing sum-of-squares error

$$\hat{\mathbf{w}}_{MLE} = (\Phi^T \Phi)^{-1} \Phi^T \mathbf{y} \quad (\text{Normal equations})$$

$$\hat{\sigma}_{MLE}^2 = \frac{1}{N} \sum_{n=1}^N (y_n - \hat{\mathbf{w}}_{MLE}^T \phi(\mathbf{x}_n))^2$$

Example: Polynomial regression using maximum likelihood I

Example from Bishop

■ Polynomial basis functions

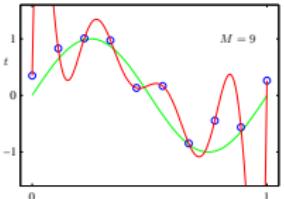
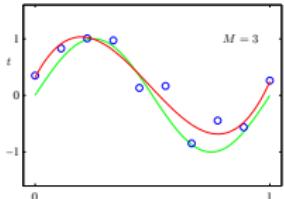
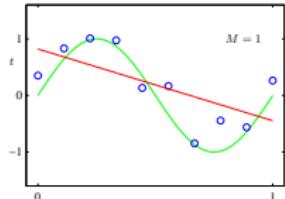
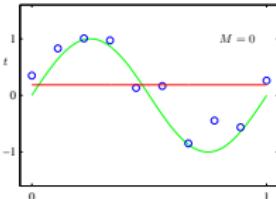
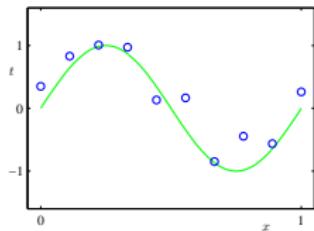
$$f(x|\mathbf{w}) = w_0 + w_1x + w_2x^2 + \cdots + w_Mx^M = \mathbf{w}^T \phi(x)$$

■ Feature transformations

$$\phi(x) = [1 \quad x \quad x^2 \quad \dots \quad x^M]^T$$

■ M controls the *model complexity*: Underfitting vs overfitting

■ *Model selection*: How to choose M ?



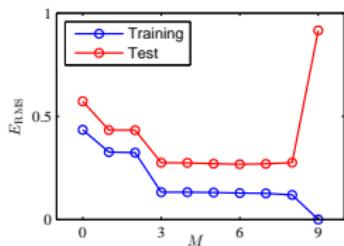
Example: Polynomial regression using maximum likelihood II

- *Cross-validation*

Split data into training and test sets

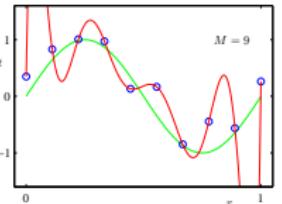
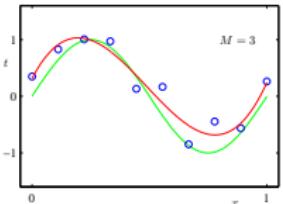
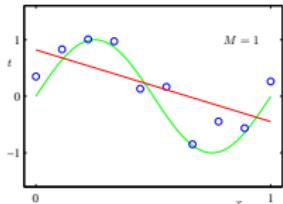
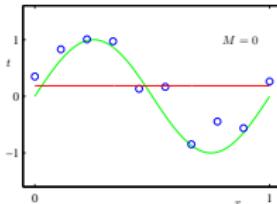
- *Overfitting (low training error, high test error)*

As the function become more flexible we start to fit the noise in the data



- *"Underfitting" (high training error, high test error)*

When the function is not sufficiently flexible to fit the data



Example: Polynomial regression using maximum likelihood III

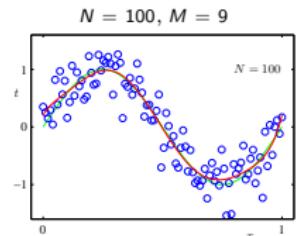
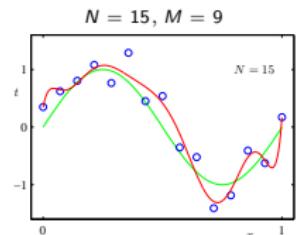
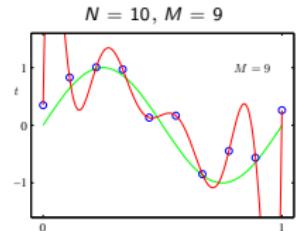
- The optimal model complexity depends on the amount of data

The more data the more flexible model we can "afford" to fit

- Regularization: Controlling the effective model complexity

We can use regularization to control the effect model complexity when we have limited data

	$M = 0$	$M = 1$	$M = 6$	$M = 9$
w_0^*	0.19	0.82	0.31	0.35
w_1^*		-1.27	7.99	232.37
w_2^*			-25.43	-5321.83
w_3^*				17.37
w_4^*				48568.31
w_5^*				-231639.30
w_6^*				640042.26
w_7^*				-1061800.52
w_8^*				1042400.18
w_9^*				-557682.99
				125201.43



Regularized least squares

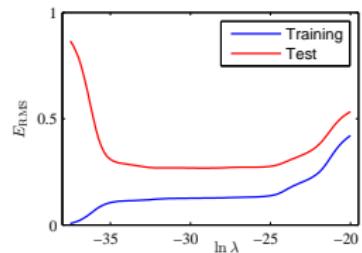
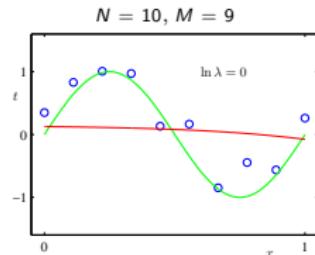
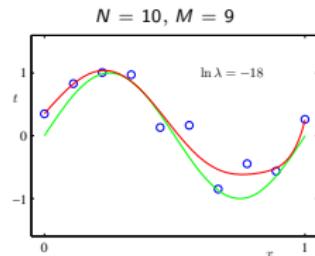
- Recall: *Maximum likelihood* is equivalent *minimizing sum-of-squares error*

$$E_D(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N (y_n - f(x_n | \mathbf{w}))^2$$

- Adding *penalty term* to prevent weights from becoming too large

$$\tilde{E}_D(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N (y_n - f(x_n | \mathbf{w}))^2 + \frac{\lambda}{2} \|\mathbf{w}\|^2$$

- What happens when $\lambda = 0$? $\lambda \rightarrow \infty$?
- Many names: Ridge regression, shrinkage, weight decay
- Regularization parameter λ controls the *effective complexity*



Bayesian linear regression

Bayesian Linear regression: motivation

- *Overfitting*

Maximum likelihood can be problematic for flexible models

- *Controlling model complexity*

Limitting number of basis functions and/or regularization?

- *Model selection*

How to choose the optimal value of λ ?

- *Cross-validation*

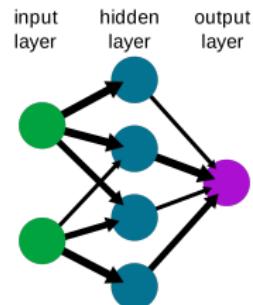
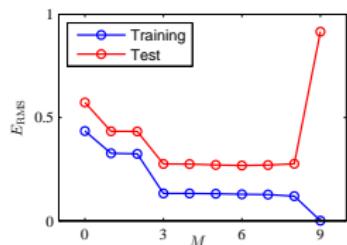
Training + validation/development + test

- *Bayesian methods*

- less prone to overfitting
- can (often) adapt model complexity automatically

- *Applications in modern machine learning*

1. Small datasets
2. Transfer learning
3. Component in more complex models
4. Simple uncertainty quantification for neural networks



Bayesian Linear regression: prior and likelihood

- Simplified set-up: assuming σ^2 is fixed and known, then Bayes' rule states

$$p(\mathbf{w}|\mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{w})p(\mathbf{w})}{p(\mathbf{y})}$$

- We already know the *likelihood*

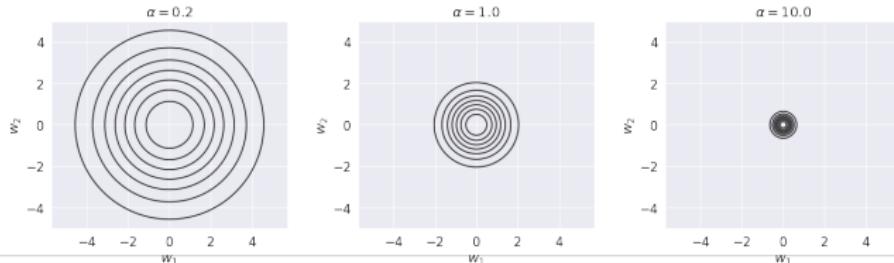
$$p(\mathbf{y}|\mathbf{x}, \mathbf{w}, \sigma^2) = \prod_{n=1}^N \mathcal{N}(y_n | \mathbf{w}^T \phi(\mathbf{x}_n), \sigma^2) = \mathcal{N}(\mathbf{y} | \Phi\mathbf{w}, \sigma^2 \mathbf{I})$$

- *The marginal likelihood* is the denominator in Bayes's theorem and is independent of \mathbf{w}

$$p(\mathbf{y}) = \int p(\mathbf{y}|\mathbf{w})p(\mathbf{w})d\mathbf{w} = \mathbb{E}_{p(\mathbf{w})} [p(\mathbf{y}|\mathbf{w})]$$

- The multivariate normal distribution is a *conjugate prior* for the \mathbf{w}

$$p(\mathbf{w}) = \mathcal{N}(\mathbf{w} | \mathbf{0}, \alpha^{-1} \mathbf{I})$$



Bayesian Linear regression: The MAP estimator

- The *posterior distribution* of the weights \mathbf{w} given the data \mathbf{y} is given by

$$p(\mathbf{w}|\mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{w})p(\mathbf{w})}{p(\mathbf{y})} = \frac{\mathcal{N}(\mathbf{y}|\Phi\mathbf{w}, \sigma^2 \mathbf{I})\mathcal{N}(\mathbf{w}|\mathbf{0}, \alpha^{-1} \mathbf{I})}{p(\mathbf{y})}$$

- Let's look at the *maximum a posteriori* (MAP) estimate: $\hat{\mathbf{w}}_{\text{MAP}} = \arg \max_{\mathbf{w}} p(\mathbf{w}|\mathbf{y})$

$$p(\mathbf{w}|\mathbf{y}) \propto \mathcal{N}(\mathbf{y}|\Phi\mathbf{w}, \sigma^2 \mathbf{I})\mathcal{N}(\mathbf{w}|\mathbf{0}, \alpha^{-1} \mathbf{I})$$

- Taking the logarithm

$$\begin{aligned}\ln p(\mathbf{w}|\mathbf{y}) &\propto \ln \mathcal{N}(\mathbf{y}|\Phi\mathbf{w}, \sigma^2 \mathbf{I}) + \ln \mathcal{N}(\mathbf{w}|\mathbf{0}, \alpha^{-1} \mathbf{I}) \\&= -\frac{N}{2} \ln(2\pi\sigma^2) - \frac{\beta}{2} \sum_{n=1}^N (y_n - \mathbf{w}^T \phi(\mathbf{x}_n))^2 - \frac{D}{2} \ln(2\pi\alpha^{-1}) - \frac{\alpha}{2} \mathbf{w}^T \mathbf{w} \\&= -\frac{\beta}{2} \sum_{n=1}^N (y_n - \mathbf{w}^T \phi(\mathbf{x}_n))^2 - \frac{\alpha}{2} \sum_{i=1}^D w_i^2 + \text{constant}\end{aligned}$$

- The *mode of the posterior (MAP)* is equivalent to ridge regression with $\lambda = \frac{\alpha}{\beta}$ and to maximum likelihood when $\alpha \rightarrow 0$

Deriving the posterior distribution of the weights

Recall the functional form of a generic multivariate Gaussian $\mathcal{N}(\mathbf{w}|\mathbf{m}, \mathbf{S})$

$$\ln \mathcal{N}(\mathbf{w}|\mathbf{m}, \mathbf{S}) = -\frac{1}{2} \mathbf{w}^T \mathbf{S}^{-1} \mathbf{w} + \mathbf{m}^T \mathbf{S}^{-1} \mathbf{w} + \text{constant}$$

- We focus on term that depends on \mathbf{w} . From the previous slide, we have

$$\begin{aligned}\ln p(\mathbf{w}|\mathbf{y}) &= -\frac{\beta}{2} \sum_{n=1}^N (y_n - \mathbf{w}^T \phi(\mathbf{x}_n))^2 - \frac{\alpha}{2} \mathbf{w}^T \mathbf{w} + \text{constant} \\ &= -\frac{\beta}{2} (\mathbf{y} - \Phi \mathbf{w})^T (\mathbf{y} - \Phi \mathbf{w}) - \frac{\alpha}{2} \mathbf{w}^T \mathbf{w} + \text{constant} \\ &= -\frac{\beta}{2} (\mathbf{y}^T \mathbf{y} - 2\mathbf{y}^T \Phi \mathbf{w} + \mathbf{w}^T \Phi^T \Phi \mathbf{w}) - \frac{\alpha}{2} \mathbf{w}^T \mathbf{w} + \text{constant} \\ &= -\frac{\beta}{2} (-2\mathbf{y}^T \Phi \mathbf{w} + \mathbf{w}^T \Phi^T \Phi \mathbf{w}) - \frac{\alpha}{2} \mathbf{w}^T \mathbf{w} + \text{constant} \\ &= -\frac{1}{2} \mathbf{w}^T (\beta \Phi^T \Phi + \alpha I) \mathbf{w} + \beta \mathbf{y}^T \Phi \mathbf{w} + \text{constant}\end{aligned}$$

- Equating coefficients for the second order term

$$\mathbf{S}^{-1} = \beta \Phi^T \Phi + \alpha I \iff \mathbf{S} = (\beta \Phi^T \Phi + \alpha I)^{-1}$$

- Equating coefficients for the first order term

$$\mathbf{m}^T \mathbf{S}^{-1} = \beta \mathbf{y}^T \Phi \iff \mathbf{m} = \beta \mathbf{S} \Phi^T \mathbf{y}$$

Bayesian linear regression model: the key equations

- Given design matrix $\Phi \in \mathbb{R}^{N \times D}$ and observations $\mathbf{y} \in \mathbb{R}^N$:

$$p(\mathbf{w}) = \mathcal{N}(\mathbf{w} | 0, \alpha^{-1} \mathbf{I}) \quad (\text{prior})$$

$$p(\mathbf{y} | \mathbf{w}) = \mathcal{N}(\mathbf{y} | \Phi \mathbf{w}, \sigma^2 \mathbf{I}) \quad (\text{likelihood})$$

$$p(\mathbf{w} | \mathbf{y}) = \mathcal{N}(\mathbf{w} | \mathbf{m}, \mathbf{S}) \quad (\text{posterior})$$

$$p(\mathbf{y}) = \mathcal{N}(\mathbf{y} | 0, \sigma^2 \mathbf{I} + \alpha^{-1} \Phi \Phi^T) \quad (\text{marginal likelihood})$$

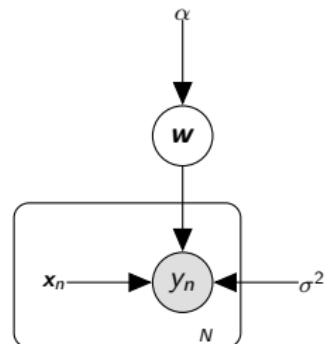
- The *posterior parameters* are given by (using $\beta \equiv \frac{1}{\sigma^2}$)

$$\mathbf{m} = \beta \mathbf{S} \Phi^T \mathbf{y}$$

$$\mathbf{S} = \left(\alpha \mathbf{I} + \beta \Phi^T \Phi \right)^{-1}$$

- Two hyperparameters*

α : prior precision of the regression weights
 β : precision of the measurements



Linear Gaussian-systems in general (see Section 3.3 in Murphy1)

- For *linear* systems: the Gaussian distribution is *conjugate* to itself
- The *posterior* for a *linear* Gaussian model with Gaussian prior is also *Gaussian*

$$p(\mathbf{y}|\mathbf{z}) = \mathcal{N}(\mathbf{y}|\mathbf{W}\mathbf{z} + \mathbf{b}, \Sigma_y) \quad p(\mathbf{z}) = \mathcal{N}(\mathbf{z}|\mu_z, \Sigma_z)$$

- The *joint* distribution $p(\mathbf{z}, \mathbf{y}) = \mathcal{N}\left(\begin{bmatrix} \mathbf{z} \\ \mathbf{y} \end{bmatrix} | \boldsymbol{\mu}, \boldsymbol{\Sigma}\right)$
- $$\boldsymbol{\mu} = \begin{bmatrix} \mu_z \\ \mathbf{W}\mu_z + \mathbf{b} \end{bmatrix} \quad \boldsymbol{\Sigma} = \begin{bmatrix} \Sigma_z & \Sigma_z \mathbf{W}^T \\ \mathbf{W}\Sigma_z & \Sigma_y + \mathbf{W}\Sigma_z \mathbf{W}^T \end{bmatrix}$$

- The *posterior* distribution of \mathbf{z} given \mathbf{y}

$$p(\mathbf{y}|\mathbf{z}) = \mathcal{N}(\mathbf{z}|\mu_{z|y}, \Sigma_{z|y})$$

$$\Sigma_{z|y}^{-1} = \Sigma_z^{-1} + \mathbf{W}^T \Sigma_y \mathbf{W}$$

$$\mu_{z|y} = \Sigma_{z|y} \left[\mathbf{W}^T \Sigma_y^{-1} (\mathbf{y} - \mathbf{b}) + \Sigma_z^{-1} \mu_z \right]$$

- The *marginal* distribution \mathbf{y}

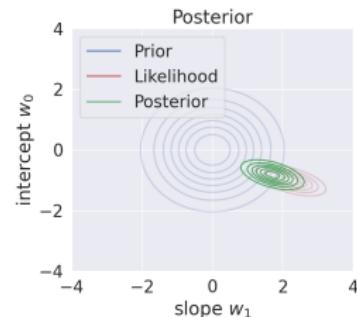
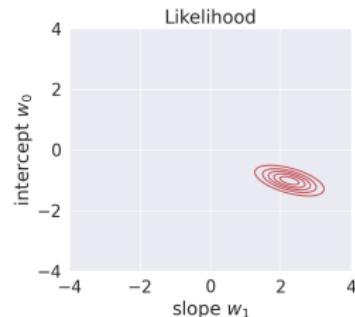
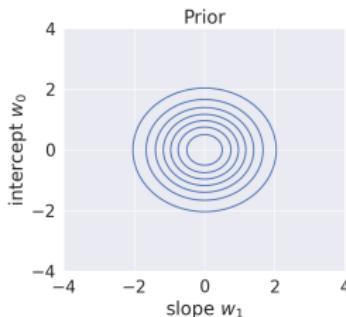
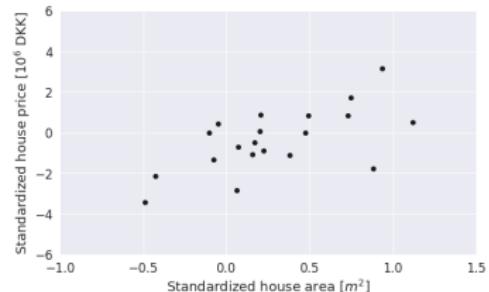
$$p(\mathbf{y}) = \mathcal{N}(\mathbf{y}|\mathbf{W}\mu_z + \mathbf{b}, \Sigma_y + \mathbf{W}\Sigma_z \mathbf{W}^T)$$

Example

- Simple linear model for fictive house prices

$$f(x|\boldsymbol{w}) = w_0 + w_1 x$$

- The posterior summarizes our beliefs about the parameters after seeing the data



$$p(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{w}|\boldsymbol{0}, \alpha^{-1} \boldsymbol{I})$$

$$p(\mathbf{y}|\boldsymbol{w}) = \mathcal{N}(\mathbf{y}|\Phi\boldsymbol{w}, \sigma^2 \boldsymbol{I})$$

$$p(\boldsymbol{w}|\mathbf{y}) = \mathcal{N}(\boldsymbol{w}|\boldsymbol{m}, \boldsymbol{S})$$

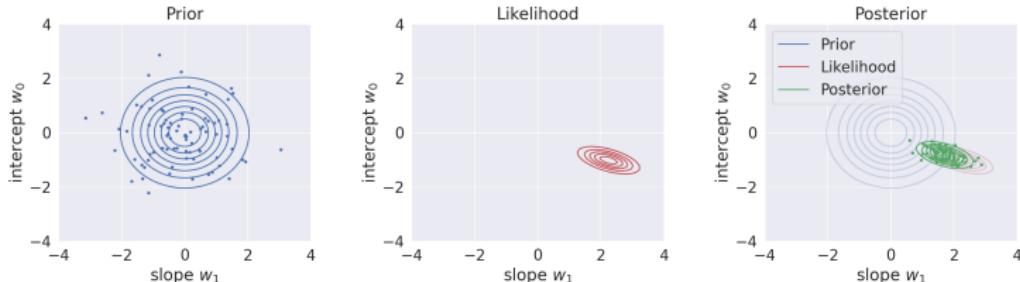
Posterior inference

$$\text{house price} = w_0 + w_1 \cdot \text{area}$$

- Question: Are larger areas associated with bigger house prices?
- We can calculate various probabilities of interest directly from the posterior (analytically or via sampling), e.g.

$$p(w_1 > 0 | y) \approx 0.99$$

- No need to remember whether to should use t-tests, F-tests, χ^2 -tests etc



The posterior predictive distribution

But how about making predictions?

- We defined a *linear* regression model

$$y_i = \mathbf{w}^T \phi(\mathbf{x}_i) + \epsilon_i$$

- We derived the *posterior distribution*

$$p(\mathbf{w}|\mathbf{y}) = \mathcal{N}(\mathbf{w}|\mathbf{m}, \mathbf{S})$$

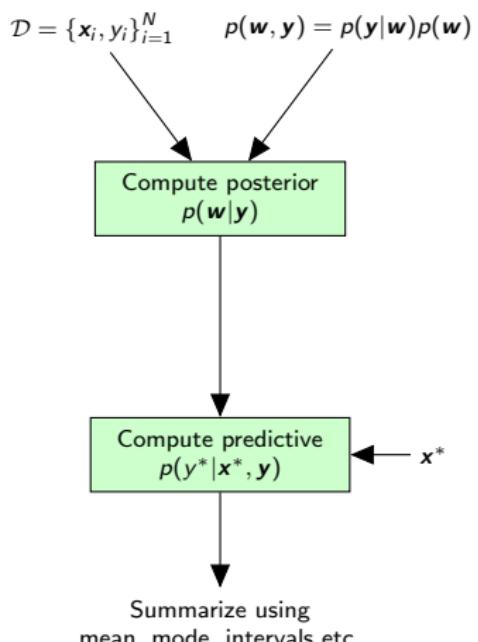
- Next goal: *making predictions* for new \mathbf{x}^*

$$p(y^*|\mathbf{x}^*, \mathbf{w}) = \mathcal{N}(y^* | \underbrace{\mathbf{w}^T \phi(\mathbf{x}^*)}_{\phi_*}, \sigma^2)$$

- The *posterior predictive distribution* is given by

$$\begin{aligned} p(y^*|\mathbf{y}, \mathbf{x}^*) &= \int \underbrace{p(y^*|\mathbf{x}^*, \mathbf{w})}_{\text{Pred. likelihood}} \underbrace{p(\mathbf{w}|\mathbf{y})}_{\text{Posterior}} d\mathbf{w} \\ &= \int \mathcal{N}(y^* | \mathbf{w}^T \phi_*, \sigma^2) \mathcal{N}(\mathbf{w}|\mathbf{m}, \mathbf{S}) d\mathbf{w} \\ &= \mathcal{N}(y^* | \mathbf{m}^T \phi_*, \phi_*^T \mathbf{S} \phi_* + \sigma^2) \end{aligned}$$

- Can be derived using eq. (3.38) in Murphy1



Quiz time!

$$p(\mathbf{w}|\mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{w})p(\mathbf{w})}{p(\mathbf{y})} \quad (\text{Bayes' rule})$$

$$p(\mathbf{y}) = \int p(\mathbf{y}|\mathbf{w})p(\mathbf{w})d\mathbf{w} \quad (\text{marginal likelihood})$$

$$p(y^*|\mathbf{y}, \mathbf{x}^*) = \int p(y^*|\mathbf{x}^*, \mathbf{w})p(\mathbf{w}|\mathbf{y})d\mathbf{w} \quad (\text{Posterior predictive dist.})$$

- Spend 5 minutes DTU Learn quiz: "Lecture 3: Bayesian inference"

Posterior Predictive distributions

Building intuition

- The *posterior distribution* is $p(\mathbf{w}|\mathbf{y}) = \mathcal{N}(\mathbf{w}|\mathbf{m}, \mathbf{S})$

- Two *model quantities* of interest

$$f^* = f(\mathbf{x}_*) = \mathbf{w}^T \boldsymbol{\phi}_*$$

$$y^* = y(\mathbf{x}_*) = \mathbf{w}^T \boldsymbol{\phi}_* + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma^2)$$

- Computing the *posterior* mean and variance

$$\mathbb{E}[f^*] = \mathbb{E}\left[\mathbf{w}^T \boldsymbol{\phi}_*\right] = \mathbb{E}\left[\mathbf{w}^T\right] \boldsymbol{\phi}_* = \mathbf{m}^T \boldsymbol{\phi}_*$$

$$\mathbb{V}[f^*] = \mathbb{V}\left[\mathbf{w}^T \boldsymbol{\phi}_*\right] = \boldsymbol{\phi}_*^T \mathbb{V}\left[\mathbf{w}^T\right] \boldsymbol{\phi}_* = \boldsymbol{\phi}_*^T \mathbf{S} \boldsymbol{\phi}_*$$

- Hence, the *posterior distribution* of f^* given the data \mathbf{y} is

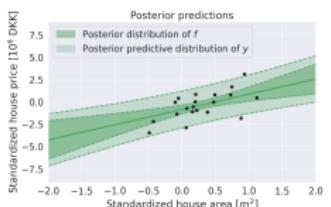
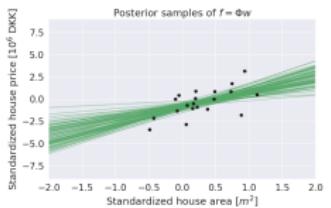
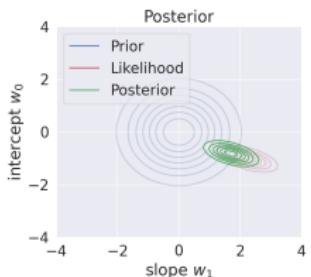
$$p(f^*|\mathbf{y}, \mathbf{x}_*) = \mathcal{N}(f^*|\mathbf{m}^T \boldsymbol{\phi}_*, \boldsymbol{\phi}_*^T \mathbf{S} \boldsymbol{\phi}_*)$$

- and the *posterior predictive distribution* for y^* becomes

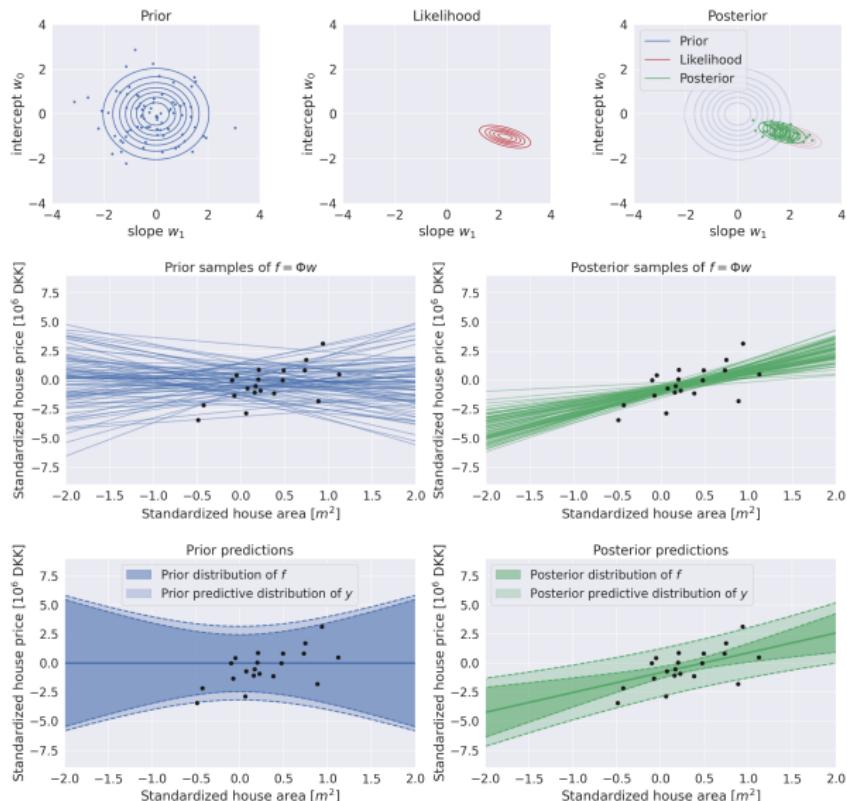
$$p(y^*|\mathbf{y}, \mathbf{x}_*) = \mathcal{N}(y^*|\mathbf{m}^T \boldsymbol{\phi}_*, \boldsymbol{\phi}_*^T \mathbf{S} \boldsymbol{\phi}_* + \sigma^2)$$

since $\mathbb{V}[\epsilon] = \sigma^2$

- We can also compute the *prior* predictive distribution.



Prior and posterior predictive distributions



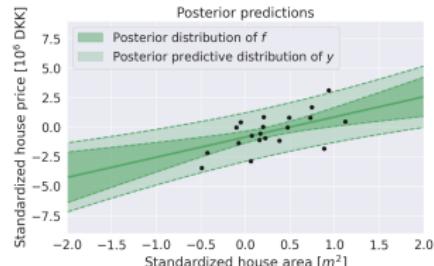
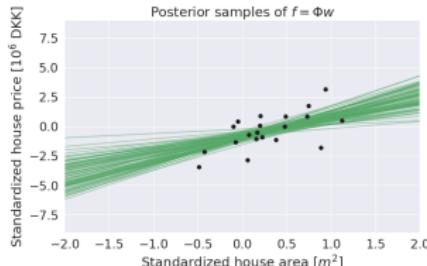
Posterior Predictive distributions

Epistemic and aleatoric uncertainty

- The *posterior predictive distribution* of $y^* = \mathbf{w}^T \phi(\mathbf{x}^*) + \epsilon$ is

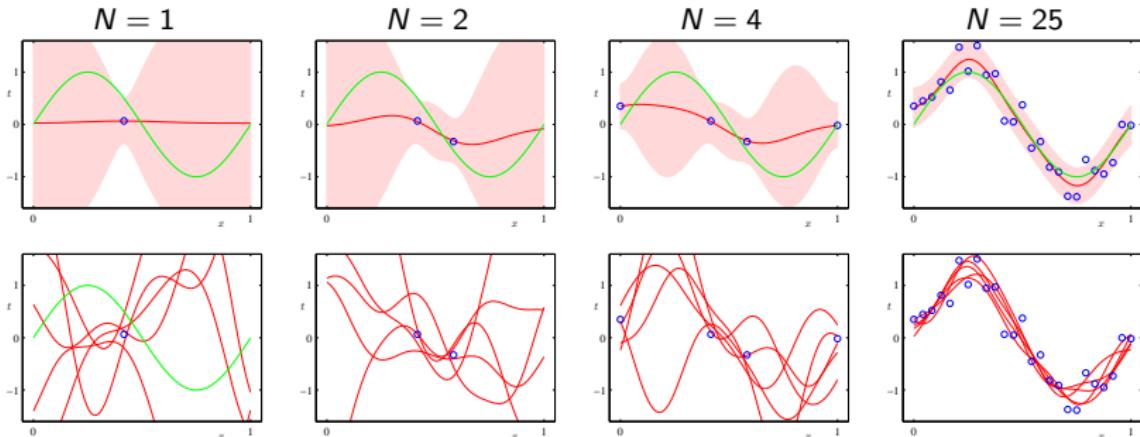
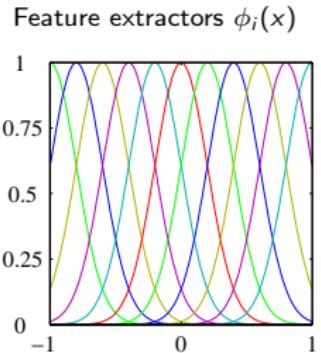
$$p(y^* | \mathbf{y}, \mathbf{x}_*) = \mathcal{N}(y^* | \mathbf{m}^T \boldsymbol{\phi}_*, \boldsymbol{\phi}_*^T \mathbf{S} \boldsymbol{\phi}_* + \sigma^2)$$

- Variance: The first term is due to parameter uncertainty (*epistemic/reducible*) and the second term is due to measurement noise (*aleatoric/irreducible*)
- The term $\boldsymbol{\phi}(\mathbf{x}_*)^T \mathbf{S} \boldsymbol{\phi}(\mathbf{x}_*)$ is the *posterior uncertainty projected to data space*
- A couple of questions for you:
 - What happens to the predictive variance when $N \rightarrow \infty$?
 - Why is the uncertainty "U-shaped"?
 - Why does the aleatoric uncertainty appear to dominate the total uncertainty near the data?



Example: posterior predictive distributions

- *Predictive distributions* for simple sinusoidal toy dataset using Gaussian Basis functions
- Samples from the posterior



Dealing with hyperparameters

But what about the hyperparameters?

- The Bayesian linear regression model

$$p(\mathbf{w}) = \mathcal{N}(\mathbf{w} | 0, \alpha^{-1} \mathbf{I}) \quad (\textit{prior})$$

$$p(\mathbf{y} | \mathbf{w}) = \mathcal{N}(\mathbf{y} | \Phi \mathbf{w}, \sigma^2 \mathbf{I}) \quad (\textit{likelihood})$$

$$p(\mathbf{w} | \mathbf{y}) = \mathcal{N}(\mathbf{w} | \mathbf{m}, \mathbf{S}) \quad (\textit{posterior})$$

$$p(\mathbf{y}) = \mathcal{N}(\mathbf{y} | \mathbf{0}, \sigma^2 \mathbf{I} + \alpha^{-1} \Phi \Phi^T) \quad (\textit{marginal likelihood})$$

- A fully Bayesian solution would require us to impose priors on α, β (recall $\beta = \frac{1}{\sigma^2}$)

$$p(\alpha, \beta | \mathbf{y}) \propto p(\mathbf{y} | \alpha, \beta) p(\alpha, \beta)$$

- ... and integrate them out

$$\begin{aligned} p(y^* | \mathbf{y}) &= \iiint p(y^* | \mathbf{w}, \beta) p(\mathbf{w} | \mathbf{y}, \alpha, \beta) p(\alpha, \beta | \mathbf{y}) d\mathbf{w} d\alpha d\beta \\ &= \mathbb{E}_{p(\alpha, \beta | \mathbf{y})} \left[\mathbb{E}_{p(\mathbf{w} | \mathbf{y}, \alpha, \beta)} [p(y^* | \mathbf{w}, \beta)] \right] \end{aligned}$$

- ... but this is not analytically tractable. Later in the course we will learn tools to deal with this in general

The evidence approximation

- How to deal with this bastard?

$$\begin{aligned} p(y^* | \mathbf{y}) &= \int \int \int p(y^* | \mathbf{w}, \beta) p(\mathbf{w} | \mathbf{y}, \alpha, \beta) p(\alpha, \beta | \mathbf{y}) d\mathbf{w} d\alpha d\beta \\ &= \mathbb{E}_{p(\alpha, \beta | \mathbf{y})} \left[\mathbb{E}_{p(\mathbf{w} | \mathbf{y}, \alpha, \beta)} [p(y^* | \mathbf{w}, \beta)] \right] \end{aligned}$$

- *The evidence approximation, maximum likelihood type II, Empirical Bayes*

- If the posterior is sharply peaked around $\hat{\alpha}$ and $\hat{\beta}$, then $p(\alpha, \beta | \mathbf{y})$ can be approximated by a Dirac's delta distribution

$$p(y^* | \mathbf{y}) \approx \mathbb{E}_{p(\mathbf{w} | \mathbf{y}, \hat{\alpha}, \hat{\beta})} [p(y^* | \mathbf{w}, \beta)]$$

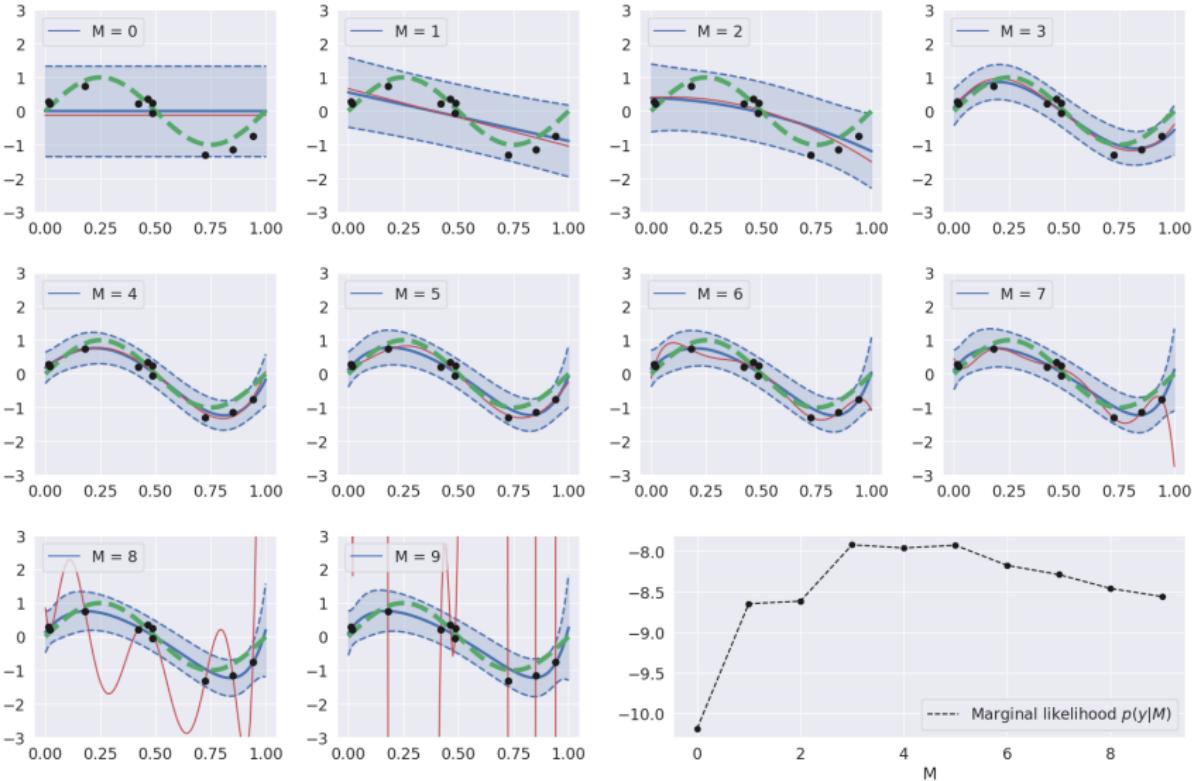
- Assume we impose a *flat prior* on α and β , then

$$p(\alpha, \beta | \mathbf{y}) \propto p(\mathbf{y} | \alpha, \beta) p(\alpha, \beta) \propto p(\mathbf{y} | \alpha, \beta)$$

- We can estimate $\hat{\alpha}, \hat{\beta}$ by *optimizing the marginal likelihood* $p(\mathbf{y} | \alpha, \beta)$

$$\hat{\alpha}, \hat{\beta} = \arg \max_{\alpha, \beta} \log p(\mathbf{y} | \alpha, \beta)$$

Sinoidal example revisited using the evidence approximation



02477 – Bayesian Machine Learning: Lecture 4

Michael Riis Andersen

Technical University of Denmark,
DTU Compute, Department of Applied Math and Computer Science

Outline

- 1 Re-cap from last week and a few words about graphical models
- 2 Bayesian vs. classical statistics
- 3 Bayesian methods for classification
 - Generative modeling
 - Discriminative modelling
- 4 Bayesian logistic regression
- 5 Laplace approximations
- 6 The posterior predictive distribution

Re-cap from last week and a few words about graphical models

Bayesian linear regression model: the key equations

- Linear regression model with Gaussian noise and Gaussian priors

$$y_n = f(\phi(x_n), \mathbf{w}) + e_n$$

- Given design matrix $\Phi \in \mathbb{R}^{N \times D}$ and observations $\mathbf{y} \in \mathbb{R}^N$

$$p(\mathbf{w}) = \mathcal{N}(\mathbf{w} | 0, \alpha^{-1} \mathbf{I}) \quad (\text{prior})$$

$$p(\mathbf{y} | \mathbf{w}) = \mathcal{N}(\mathbf{y} | \Phi \mathbf{w}, \beta^{-1} \mathbf{I}) \quad (\text{likelihood})$$

$$p(\mathbf{y}) = \mathcal{N}(\mathbf{y} | \mathbf{0}, \beta^{-1} \mathbf{I} + \alpha^{-1} \Phi \Phi^T) \quad (\text{marginal likelihood})$$

$$p(\mathbf{w} | \mathbf{y}) = \mathcal{N}(\mathbf{w} | \mathbf{m}_N, \mathbf{S}_N) \quad (\text{posterior})$$

with *posterior parameters*

$$\mathbf{m}_N = \beta \mathbf{S}_N \Phi^T \mathbf{y} \quad (\text{posterior mean})$$

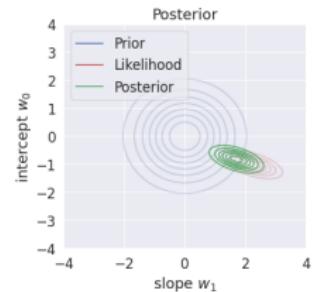
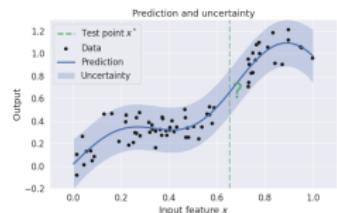
$$\mathbf{S}_N = (\alpha \mathbf{I} + \beta \Phi^T \Phi)^{-1} \quad (\text{posterior covariance})$$

- Two hyperparameters

α : prior precision of the regression weights

β : precision of the measurements

- *Lazy notation*: We should actually write $p(\mathbf{w} | \mathbf{y}, \alpha, \beta)$ etc., but we often suppress dependency of hyperparameter to ease notation

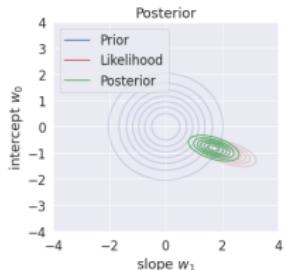


Posterior Predictive distributions

- The *posterior distribution* is $p(\mathbf{w}|\mathbf{y}) = \mathcal{N}(\mathbf{w}|\mathbf{m}, \mathbf{S})$ with

$$\mathbf{m} = \beta \mathbf{S} \Phi^T \mathbf{y}$$

$$\mathbf{S} = (\alpha \mathbf{I} + \beta \Phi^T \Phi)^{-1}$$



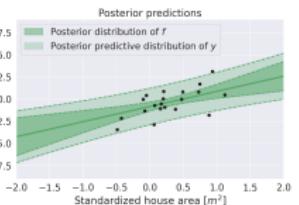
- When making predictions \mathbf{x}^* using Bayesian methods, we *average over all possible parameters values weighted by the posterior*

$$f(\mathbf{x}^* | \mathbf{w}) = \mathbf{w}^T \phi(\mathbf{x}^*)$$

$$y(\mathbf{x}^*) = f(\mathbf{x}^* | \mathbf{w}) + \epsilon$$

- First, we write the likelihood corresponding to the new input \mathbf{x}_*

$$p(y^* | \mathbf{x}^*, \mathbf{w}) = \mathcal{N}(y^* | \mathbf{w}^T \phi(\mathbf{x}^*), \beta^{-1})$$



- .. and marginalize with respect to the posterior distribution (sum rule)

$$p(y^* | \mathbf{y}) = \int p(y^* | \mathbf{x}^*, \mathbf{w}) p(\mathbf{w} | \mathbf{y}) d\mathbf{w}$$

- This is called *the posterior predictive distribution*

Posterior predictive distributions: how to make predictions?

- First, we write up the *likelihood* corresponding for the new input $\phi_* = \phi(x_*)$:

$$p(y_*|x_*, w) = \mathcal{N}(y_* | w^T \phi_*, \beta^{-1})$$

- For MAP (and similar for MLE), we simply *plug in* the estimate of w

$$p(y_*|y, x_*) \approx \mathcal{N}(y_* | \hat{w}_{MAP}^T \phi_*, \beta^{-1})$$

- Bayesian: use the *sum rule* to *marginalize* wrt. the *posterior* distribution

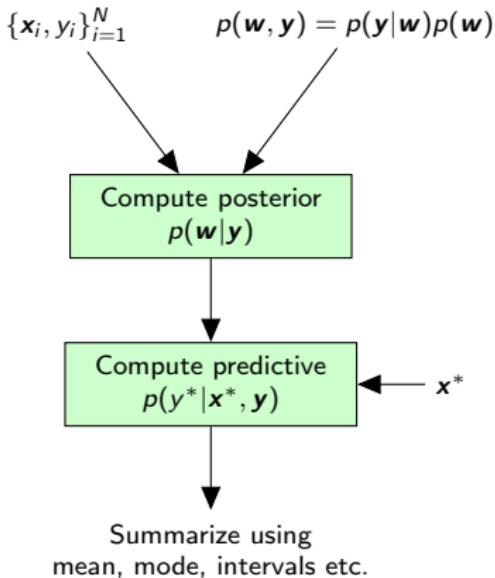
$$p(y_*|y, x_*) = \int p(y^*|x_*, w)p(w|y)dw = \mathcal{N}(y_* | \hat{w}_{MAP}^T \phi_*, \phi_*^T S \phi_* + \beta^{-1})$$

- If posterior covariance S is small, we get *approximately* the same result
- We can think of the *MAP* solution as an *approximate* posterior distribution

$$p(w|y) \approx \delta(w - w_{MAP}) = \begin{cases} \infty & \text{if } w = w_{MAP} \\ 0 & \text{otherwise} \end{cases}$$

- Such a distribution is called *Dirac's delta* distribution (mental picture: Gaussian with mean w_{MAP} and variance going to zero)
- MAP is sometimes called *poor man's Bayes*, but can still be a useful tool!

Bayesian inference for supervised learning



- Same principles for linear regression, logistic regression, neural networks etc. etc.

Hyperparameters and the evidence approximation

- Posterior depends on the hyperparameters α and β (but often suppressed in notation)

$$p(\mathbf{w}|\mathbf{y}, \alpha, \beta) = \frac{p(\mathbf{y}|\mathbf{w}, \beta)p(\mathbf{w}|\alpha)}{p(\mathbf{y}|\alpha, \beta)}$$

- We could assign priors to α and β to get the posterior on α and β given the data

$$p(\alpha, \beta|\mathbf{y}) \propto p(\mathbf{y}|\alpha, \beta)p(\alpha)p(\beta)$$

- fully Bayesian solution: use the sum rule to marginalize over all unknowns $(\mathbf{w}, \alpha, \beta)$

$$p(y_*|\mathbf{y}, \mathbf{x}_*) = \int p(y^*|\mathbf{x}_*, \mathbf{w}_*, \beta)p(\mathbf{w}|\mathbf{y}, \alpha, \beta)p(\alpha, \beta|\mathbf{y}) d\mathbf{w}d\alpha d\beta$$

The evidence approximation

- We estimate $\hat{\alpha}, \hat{\beta}$ by *optimizing the marginal likelihood* $p(\mathbf{y}|\alpha, \beta)$

$$\hat{\alpha}, \hat{\beta} = \arg \max_{\alpha, \beta} \log p(\mathbf{y}|\alpha, \beta)$$

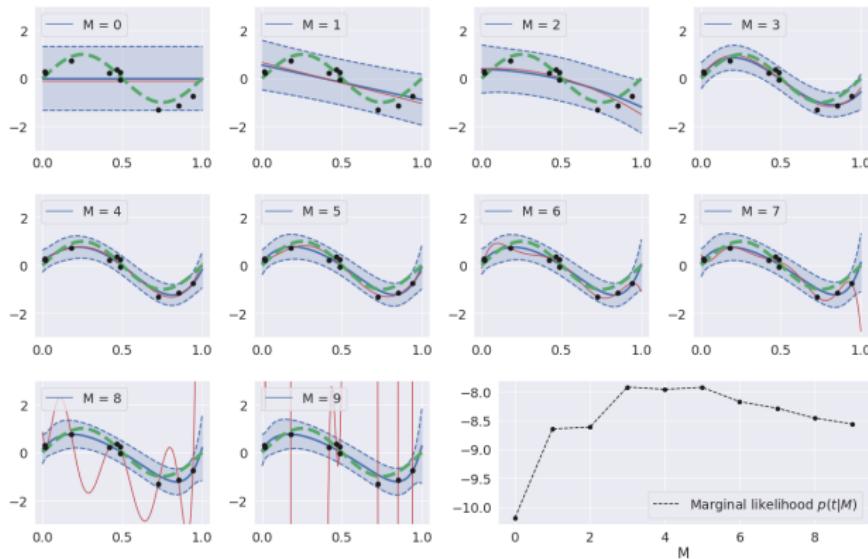
- Equivalent to maximizing the posterior $p(\alpha, \beta|\mathbf{y})$ assuming *flat priors* for α and β

$$p(\alpha, \beta|\mathbf{y}) \propto p(\mathbf{y}|\alpha, \beta)$$

- Equivalent to *poor man's Bayes* on hyperparameter level

Sinusoidal example revisited using the evidence approximation

- Also useful for model selection: $\alpha, \beta, M^* = \arg \max_{\alpha, \beta, M} p(\mathbf{y} | \alpha, \beta, M)$



- Implements "Occam's razor": choose the "simplest" model that explain the data
- Often works well for many models, but we should always assess the generalization error

A more general probabilistic perspective on supervised learning

Product rule

$$p(\mathbf{a}, \mathbf{b}) = p(\mathbf{b}|\mathbf{a})p(\mathbf{a})$$

Sum rule

$$p(\mathbf{b}) = \int p(\mathbf{a}, \mathbf{b})d\mathbf{a}$$

Conditional

$$p(\mathbf{a}|\mathbf{b}) = \frac{p(\mathbf{a}, \mathbf{b})}{p(\mathbf{b})}$$

Conditional independence

$$p(\mathbf{a}, \mathbf{b}|\mathbf{c}) = p(\mathbf{a}|\mathbf{c})p(\mathbf{b}|\mathbf{c})$$

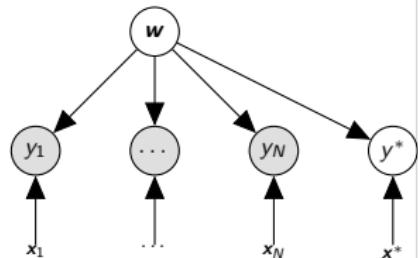
Supervised learning: Given some data $\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=1}^N$, what can we say about a new test point $y^* = y(\mathbf{x}^*)$?

- Step 1: Formulate joint distribution for all variables of interests

$$p(y^*, \mathbf{y}, \mathbf{w}) = p(y^*, \mathbf{y}|\mathbf{w})p(\mathbf{w}) = p(y^*|\mathbf{w})p(\mathbf{y}|\mathbf{w})p(\mathbf{w})$$

- Step 2: Conditioned on the observed data \mathbf{y}

$$p(y^*, \mathbf{w}|\mathbf{y}) = \frac{p(y^*|\mathbf{w})p(\mathbf{y}|\mathbf{w})p(\mathbf{w})}{p(\mathbf{y})}$$



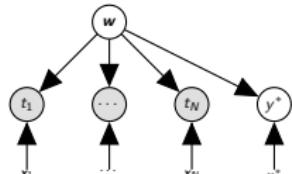
- Step 3: Marginalize over all parameters \mathbf{w} using sum rule

$$p(y^*|\mathbf{y}) = \int \frac{p(y^*|\mathbf{w})p(\mathbf{y}|\mathbf{w})p(\mathbf{w})}{p(\mathbf{y})} d\mathbf{w} = \int p(y^*|\mathbf{w})p(\mathbf{w}|\mathbf{y})d\mathbf{w}$$

A more complete graphical model

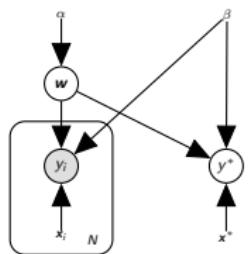
- Lazy, but common notation for Bayesian linear regression

$$p(y^*, \mathbf{y}, \mathbf{w}) = p(y^* | \mathbf{w}) p(\mathbf{y} | \mathbf{w}) p(\mathbf{w})$$



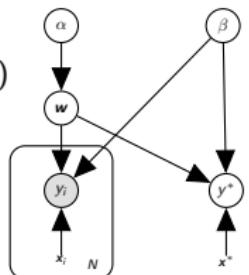
- More complete notation and corresponding graphical model

$$p(y^*, \mathbf{y}, \mathbf{w} | \alpha, \beta, \mathbf{X}, \mathbf{x}^*) = p(y^* | \mathbf{w}, \beta, \mathbf{x}^*) p(\mathbf{y} | \mathbf{w}, \beta, \mathbf{X}) p(\mathbf{w} | \alpha)$$



- Fully Bayesian inference on hyperparameter level

$$p(y^*, \mathbf{y}, \mathbf{w}, \alpha, \beta | \mathbf{X}, \mathbf{x}^*) = p(y^* | \mathbf{w}, \beta, \mathbf{x}^*) p(\mathbf{y} | \mathbf{w}, \beta, \mathbf{X}) p(\mathbf{w} | \alpha) p(\alpha) p(\beta)$$



Bayesian vs. classical statistics

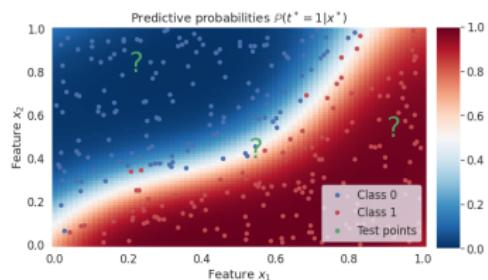
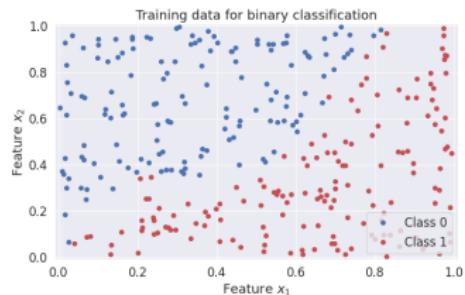
Bayesian vs. classical statistics

	Frequentist/classical	Bayesian
Probability interpretation	Long run frequencies	Degrees of belief
Parameters	Deterministic, but unknown Cannot make probabilistic statement about parameters	Random variables Probabilistic reasoning at levels: models, parameters and observations
	Point estimates	Probability distributions
Interpretation of intervals	<i>Confidence intervals</i> If the experiment is repeated infinitely many times, 95% of the intervals will contain the true population value	<i>Credibility intervals</i> The interval will contain the population value with 95% probability given the data
Sources of information	Data only	Data & prior knowledge
Computation	Often less computationally expensive	Often more computationally expensive

Bayesian methods for classification

Probabilistic approaches for classification

- Dataset $\mathcal{D} = \{(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\}$
 - Input features: $x_i \in \mathbb{R}^D$
 - Targets: $y_i \in \{0, 1\}$
- How to predict label for test point $x^* \in \mathbb{R}^D$?
- *Predictive distributions*: what is the probability that x^* belong to class 1, i.e. $p(y^* = 1 | \mathcal{D}, x^*)$?
- Two probabilistic approaches
 1. *Discriminative methods*
 2. *Generative methods*



Discriminative vs generative methods

- The *generative approach* models the *joint distribution* $p(\mathbf{x}_n, y_n)$, e.g. via Bayes rule

$$p(y_n = k | \mathbf{x}_n) = \frac{p(\mathbf{x}_n | y_n = k)p(y_n = k)}{p(\mathbf{x}_n)}$$

- Joint distribution over inputs \mathbf{x}_n and labels y_n allow sampling (*generating*) from the model

$$\mathbf{x}^{(i)}, y^{(i)} \sim p(\mathbf{x}, y)$$

- Pros and cons for generative models

- + Optimal if the assumptions are correct
- + Can easily handle *missing data*
- + Can reason about input data
- Assumptions are often hard to get correct

- The *discriminative approach* models the *conditional distribution* $p(y_n | \mathbf{x}_n)$ directly by assuming some parametric form for the posterior

$$p(y_n | \mathbf{x}_n) = f(\mathbf{x}_n | \mathbf{w})$$

- The function $f(\mathbf{x}_n | \mathbf{w})$ can be based on a linear model, a neural network etc.

- Pros and cons for discriminative models

- + Often superior when the assumptions for generative models are wrong
- + Often better calibrated (compared to e.g. generative methods like Naive Bayes etc)
- + Easy to make flexible
- Difficult to handle missing data
- Cannot reason about input data

Bayesian methods for classification: Generative modeling

The generative approach I

- Binary classification $y_n \in \{0, 1\}$

$$p(y_n = 1 | \mathbf{x}_n) = \frac{p(\mathbf{x}_n | y_n = 1)p(y_n = 1)}{p(\mathbf{x}_n)}$$

- Terminology

- *Class-conditional distribution* $p(\mathbf{x}_n | y_n)$
- *Prior probabilities* $p(y_n = k) = \pi_k$
- *Marginal data density* $p(\mathbf{x}_n)$

- The marginal density of \mathbf{x}_n is a *mixture distribution* and is obtained using the *sum rule*

$$p(\mathbf{x}_n) = \sum_{k \in \{0,1\}} p(\mathbf{x}_n | y_n = k)p(y_n = k) = \pi_0 p(\mathbf{x}_n | y_n = 0) + \pi_1 p(\mathbf{x}_n | y_n = 1)$$

- Let's plug the result into Bayes' rule

The generative approach II

- The posterior of y_n given the input \mathbf{x}_n

$$p(y_n = 1|\mathbf{x}_n) = \frac{\pi_1 p(\mathbf{x}_n|y_n = 1)}{\pi_0 p(\mathbf{x}_n|y_n = 0) + \pi_1 p(\mathbf{x}_n|y_n = 1)}$$

- Divide by numerator

$$p(y_n = 1|\mathbf{x}_n) = \frac{1}{1 + \frac{\pi_0 p(\mathbf{x}_n|y_n=0)}{\pi_1 p(\mathbf{x}_n|y_n=1)}}$$

- Define

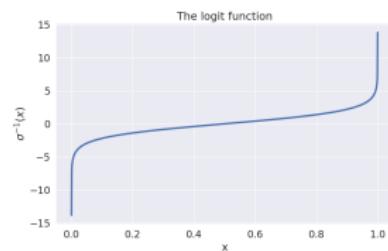
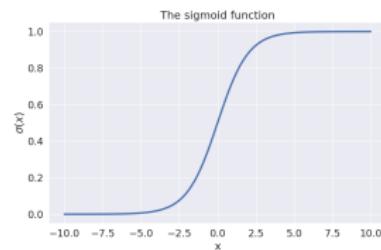
$$a = \ln \frac{\pi_1 p(\mathbf{x}_n|y_n = 1)}{\pi_0 p(\mathbf{x}_n|y_n = 0)}$$

- then

$$p(y_n = 1|\mathbf{x}_n) = \frac{1}{1 + \exp(-a)} = \sigma(a)$$

- Recall $\sigma(a)$ is the *logistic sigmoid* function and its inverse is called the *logit* function

$$a = \ln \left(\frac{\sigma}{1-\sigma} \right)$$



The generative approach III: multi-class problems and softmax

- Assume we have K different classes, where $k = 1, \dots, K$

- Define a_k

$$a_k = \ln p(\mathbf{x}_n | y_n = k) p(y_n = k)$$

- Using similar line of reasoning for K classes

$$p(y_n = k | \mathbf{x}_n) = \frac{p(\mathbf{x}_n | y_n = k) P(y_n = k)}{\sum_{i=1}^K p(\mathbf{x}_n | y_n = i) p(y_n = i)} = \frac{\exp(a_k)}{\sum_{j=1}^K \exp(a_j)}$$

- The *normalized exponentials* is the known as the *softmax* function

Example: Gaussian class conditionals

- Binary classification, normal class conditionals with common variance in 1D

$$p(x_n|y_n = 0) = \mathcal{N}(x_n|\mu_0, \sigma^2)$$

$$p(x_n|y_n = 1) = \mathcal{N}(x_n|\mu_1, \sigma^2)$$

- We know

$$p(y_n = 1|x_n) = \frac{1}{1 + \exp(-a)} = \sigma(a) = \sigma(w_0 + w_1 x_n)$$

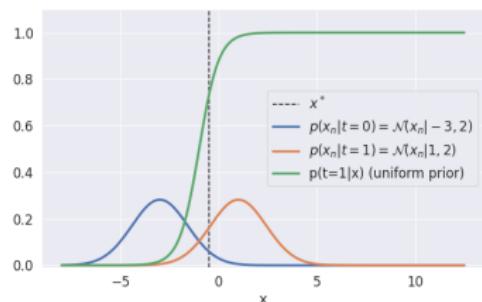
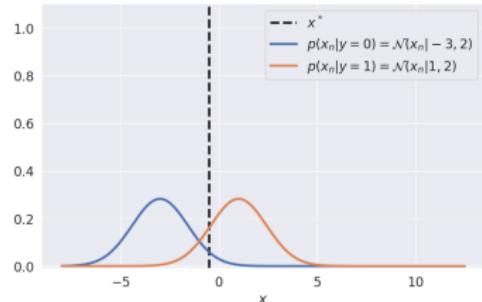
- The quantity a has a simple expression

$$a = \ln \frac{\pi_1 p(x_n|y_n = 1)}{\pi_0 p(x_n|y_n = 0)} = \ln \frac{\pi_1 \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x_n - \mu_1)^2}{2\sigma^2}\right)}{\pi_0 \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x_n - \mu_0)^2}{2\sigma^2}\right)}$$

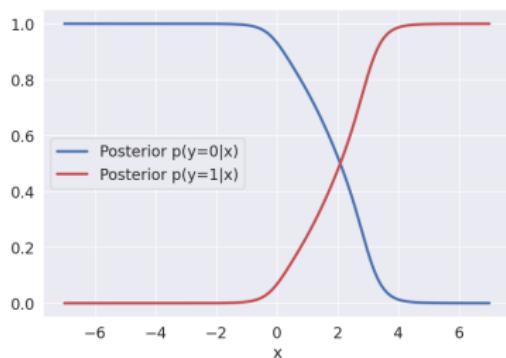
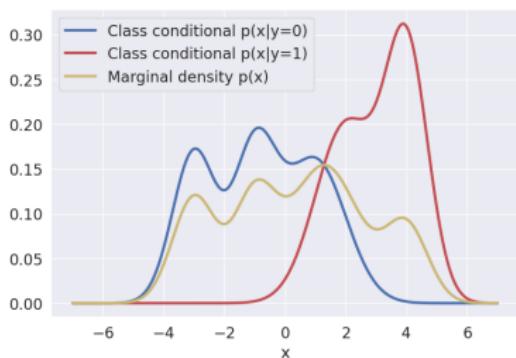
$$= \ln \frac{\pi_1}{\pi_0} - \frac{\mu_1^2 - \mu_0^2}{2\sigma^2} + \frac{\mu_1 - \mu_0}{\sigma^2} x_n$$

$$= w_0 + w_1 x_n$$

where $w_0 = \ln \frac{\pi_1}{\pi_0} - \frac{\mu_1^2 - \mu_0^2}{2\sigma^2}$ and $w_1 = \frac{\mu_1 - \mu_0}{\sigma^2}$



Example 2: More complex distributions



Bayesian methods for classification: Discriminative modelling

Discriminative modelling for binary classification

- In the generative model, we defined $p(y_n = 1) = \pi_1$ and a set of *class-conditionals* $p(\mathbf{x}_n | y_n = 1)$, applied Bayes rule and ended up with

$$a(\mathbf{x}_n) = \ln \frac{\pi_1 p(\mathbf{x}_n | y_n = 1)}{\pi_0 p(\mathbf{x}_n | y_n = 0)} = \ln \frac{p(y_n = 1 | \mathbf{x}_n)}{p(y_n = 0 | \mathbf{x}_n)}$$

and

$$p(y_n = 1 | \mathbf{x}_n) = \frac{1}{1 + \exp(-a)} = \sigma(a(\mathbf{x}_n))$$

- The distributional assumptions gave the specific functional form for $a(\mathbf{x})$, but in *discriminative modelling*, we directly assume a functional form for $a(\mathbf{x})$

- Example: logistic regression

$$p(y_n = 1 | \mathbf{x}_n, \mathbf{w}) = \frac{1}{1 + \exp(-a)} = \sigma(\phi(\mathbf{x}_n)^T \mathbf{w})$$

- We model each observation with a *Bernoulli* distribution with probability $\sigma(\phi(\mathbf{x}_n)^T \mathbf{w})$)
- We *estimate* \mathbf{w} using maximum likelihood, MAP or Bayesian inference with the likelihood function

$$p(\mathbf{y} | \mathbf{w}, \mathbf{X}) = \prod_{n=1}^N \sigma(\phi(\mathbf{x}_n)^T \mathbf{w})^{y_n} \left(1 - \sigma(\phi(\mathbf{x}_n)^T \mathbf{w})\right)^{1-y_n}$$

Maximum likelihood estimator for logistic regression: Quiz

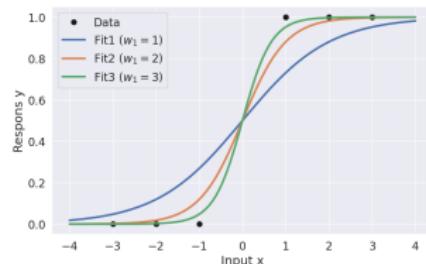
Set-up

- Consider a simple dataset with $N = 6$, $\phi(x) = x$

- Logistic regression likelihood

$$p(\mathbf{y}|\mathbf{w}_1) = \prod_{n=1}^N \sigma(w_1 x_n)^{y_n} (1 - \sigma(w_1 x_n))^{1-y_n}$$

- One parameter: w_1



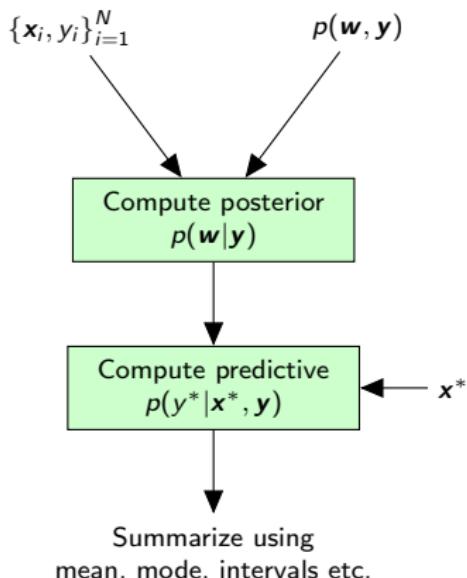
Questions (5mins)

- Spend 5 minutes DTU Learn quiz: "Lecture 4: Logistic regression"

Bayesian logistic regression

Bayesian supervised learning

- For conjugate models, both the posterior and predictive distributions can be computed analytically
- The real strength of the Bayesian framework lies in the modelling flexibility
- ... but for even rather simple models like *Bayesian logistic regression*, we cannot compute
 1. the posterior distribution
 2. the predictive distribution
- Today we will see how to use the *Laplace approximation* to approximate the posterior
- ... and we will discuss different strategies to evaluate the predictive distribution



Bayesian logistic regression

- Likelihood for logistic regression

$$p(\mathbf{y}|\mathbf{w}) = \prod_{n=1}^N \sigma(\phi(\mathbf{x}_n)^T \mathbf{w})^{y_n} \left(1 - \sigma(\phi(\mathbf{x}_n)^T \mathbf{w})\right)^{1-y_n}$$

- Let's impose a prior distribution on the weights \mathbf{w} assuming the individual weights w_i are *independent and identically distributed (i.i.d)* a priori

$$p(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mathbf{0}, \alpha^{-1} \mathbf{I}) = \prod_{i=1}^D \mathcal{N}(w_i|0, \alpha^{-1})$$

- The posterior follows from Bayes' theorem

$$p(\mathbf{w}|\mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{w})p(\mathbf{w})}{p(\mathbf{y})}$$

- Let's calculate the posterior mean

$$\mathbb{E}_{p(\mathbf{w}|\mathbf{y})} [\mathbf{w}] = \int \mathbf{w} p(\mathbf{w}|\mathbf{y}) d\mathbf{w} = \frac{1}{p(\mathbf{y})} \int \mathbf{w} p(\mathbf{y}|\mathbf{w})p(\mathbf{w}) d\mathbf{w}$$

- Clearly, we need $p(\mathbf{y})$ as well

$$p(\mathbf{y}) = \int p(\mathbf{y}|\mathbf{w})p(\mathbf{w}) d\mathbf{w}$$

Bayesian logistic regression II

- General problem: we *cannot* compute the posterior mean analytically

$$\mathbb{E}_{p(\mathbf{w}|\mathbf{y})} [\mathbf{w}] = \frac{1}{p(\mathbf{y})} \int \mathbf{w} p(\mathbf{y}|\mathbf{w}) p(\mathbf{w}) d\mathbf{w}$$

- This *roadblock* occurs for almost any other interesting posterior summary

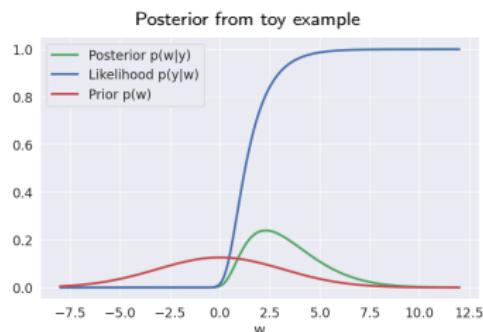
- The posterior distribution and the marginal likelihood for most Bayesian models is *analytically intractable*

- Posterior distribution from our toy example is asymmetric, but almost resembles a Gaussian

- Bernstein von Mises theorem*

Assuming certain regularity conditions, the posterior distribution of a parametric model becomes more and more Gaussian as N increases

- Let's approximate $p(\mathbf{w}|\mathbf{y})$ with a Gaussian!



Laplace approximations

Laplace approximations I

- The *Laplace approximation* is a method for approximating intractable probability densities
- Assume we have a posterior distribution of interest $p(\mathbf{w}|\mathbf{y})$

$$p(\mathbf{w}|\mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{w})p(\mathbf{w})}{p(\mathbf{y})} = \frac{1}{Z}f(\mathbf{w}) \approx \mathcal{N}(\mathbf{w}|\mathbf{m}, \mathbf{S})$$

- The log density for Gaussians is quadratic wrt. \mathbf{w}

$$\ln \mathcal{N}(\mathbf{w}|\mathbf{m}, \mathbf{S}) = -\frac{D}{2} \ln(2\pi) - \frac{1}{2} \log |\mathbf{S}| - \frac{1}{2}(\mathbf{w} - \mathbf{m})^T \mathbf{S}^{-1}(\mathbf{w} - \mathbf{m})$$

- Let's make a second order Taylor expansion of $f(\mathbf{w})$ around the mode \mathbf{w}_{MAP}

$$\ln f(\mathbf{w}) \approx \ln f(\mathbf{w}_{\text{MAP}}) - \frac{1}{2}(\mathbf{w} - \mathbf{w}_{\text{MAP}})^T \mathbf{A}(\mathbf{w} - \mathbf{w}_{\text{MAP}}),$$

where \mathbf{A} is the Hessian at the mode, i.e. $\mathbf{A} = -\nabla \nabla \ln f(\mathbf{w})|_{\mathbf{w}=\mathbf{w}_{\text{MAP}}}$

- The Laplace approximation is defined

$$q(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mathbf{w}_{\text{MAP}}, \mathbf{A}^{-1})$$

- That is, we approximate the posterior mean using the MAP and the posterior covariance using the curvature at the MAP solution.

Laplace approximation II

- Suppose we want to approximate $p(\mathbf{w}|\mathbf{y})$ using the Laplace approximation

$$p(\mathbf{w}|\mathbf{y}) \approx q(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mathbf{w}_{\text{MAP}}, \mathbf{A}^{-1})$$

- Computational steps

- Locate the mode of $p(\mathbf{w}|\mathbf{y})$

$$\mathbf{w}_{\text{MAP}} = \arg \max_{\mathbf{w}} p(\mathbf{w}|\mathbf{y}) = \arg \max_{\mathbf{w}} p(\mathbf{y}|\mathbf{w})p(\mathbf{w})$$

- Evaluate the Hessian at \mathbf{w}_{MAP}

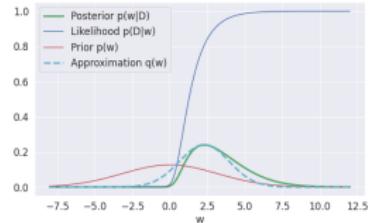
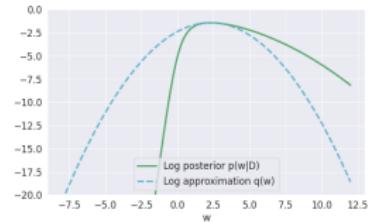
$$\mathbf{A} = -\nabla \nabla \ln p(\mathbf{y}|\mathbf{w})p(\mathbf{w}) \Big|_{\mathbf{w}=\mathbf{w}_{\text{MAP}}}$$

- Advantages

- Simple and well-understood
- Very fast to compute
- Gives good results for many problems

- Limitations

- Only applies to continuous parameters
- Gaussian (symmetric distribution, thin tails)
- Only capture local properties of $p(\mathbf{w}|\mathbf{y})$ near \mathbf{w}_{MAP}
- Does not work for hierarchical models in general



Laplace approximations III: Approximating the marginal likelihood

- Our second order Taylor approximation for $\ln f(\mathbf{w})$

$$\ln f(\mathbf{w}) \approx \ln f(\mathbf{w}_{\text{MAP}}) - \frac{1}{2}(\mathbf{w} - \mathbf{w}_{\text{MAP}})^T \mathbf{A}(\mathbf{w} - \mathbf{w}_{\text{MAP}}),$$

- By assumption

$$p(\mathbf{w}|\mathbf{y}) = \frac{1}{Z} f(\mathbf{w}) \quad \Rightarrow \quad Z = \int f(\mathbf{w}) d\mathbf{w}$$

- Plugging in the approximation for $\ln f(\mathbf{w})$

$$\begin{aligned} Z &= \int f(\mathbf{w}) d\mathbf{w} \\ &\approx f(\mathbf{w}_{\text{MAP}}) \int \exp \left(-\frac{1}{2}(\mathbf{w} - \mathbf{w}_{\text{MAP}})^T \mathbf{A}(\mathbf{w} - \mathbf{w}_{\text{MAP}}) \right) \\ &= f(\mathbf{w}_{\text{MAP}}) \frac{(2\pi)^{\frac{D}{2}}}{|\mathbf{A}|^{\frac{1}{2}}} \end{aligned}$$

- Using $f(\mathbf{w}) = p(\mathbf{y}|\mathbf{w})p(\mathbf{w})$, our approximation for $p(\mathbf{y})$ becomes

$$\ln p(\mathbf{y}) \approx \ln p(\mathbf{y}|\mathbf{w}_{\text{MAP}}) + \ln p(\mathbf{w}_{\text{MAP}}) + \frac{D}{2} \ln(2\pi) - \frac{1}{2} \ln |\mathbf{A}|$$

- Very useful for model selection, parameter tuning etc

The posterior predictive distribution

How to make predictions?

- For classification, we need the predictive distribution for a new input \mathbf{x}^* . The likelihood for a input data point \mathbf{x}^* is

$$p(y^* = 1 | \mathbf{w}, \mathbf{x}^*) = \sigma(\phi(\mathbf{x}^*)^T \mathbf{w})$$

- As always, we want to take the posterior uncertainty into account using the sum rule

$$\begin{aligned} p(y^* = 1 | \mathbf{y}, \mathbf{x}^*) &= \int p(y^* = 1 | \mathbf{x}^*, \mathbf{w}) p(\mathbf{w} | \mathbf{y}) d\mathbf{w} \\ &\approx \int p(y^* = 1 | \mathbf{x}^*, \mathbf{w}) q(\mathbf{w}) d\mathbf{w} \\ &= \int \sigma(\phi(\mathbf{x}^*)^T \mathbf{w}) \mathcal{N}(\mathbf{w} | \mathbf{m}, \mathbf{S}) d\mathbf{w} \\ &= \int \sigma(f) \mathcal{N}(f | \mu, \sigma^2) df \end{aligned}$$

where

$$\mu = \phi(\mathbf{x}^*)^T \mathbf{m} \quad \sigma^2 = \phi(\mathbf{x}^*)^T \mathbf{S} \phi(\mathbf{x}^*)$$

- The good news: we only have to calculate 1D integrals to make predictions
- The bad news: the integral does not have analytical solution

Evaluating predictive distributions for logistic regression

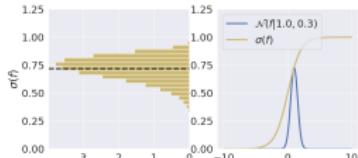
How does uncertainty in f affect the distribution of $\sigma(f)$?

$$p(y^* = 1 | \mathbf{y}, \mathbf{x}^*) = \int \sigma(f) \mathcal{N}(f | \mu, \sigma^2) df$$

■ General strategies for evaluating this integral

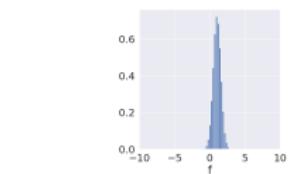
1. Monte Carlo methods (sampling)

$$p(y^* = 1 | \mathbf{y}, \mathbf{x}^*) \approx \frac{1}{S} \sum_{i=1}^S \sigma\left(f^{(i)}\right) \quad \text{for} \quad f^{(i)} \sim \mathcal{N}(f | \mu, \sigma^2)$$



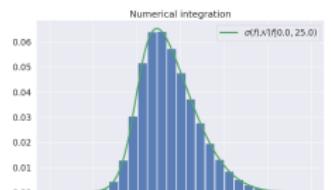
2. Numerical integration (Gauss-Hermite integration)

$$p(y^* = 1 | \mathbf{y}, \mathbf{x}^*) \approx \frac{1}{\sqrt{\pi}} \sum_{i=1}^S w_i h(\sqrt{2}\sigma x_i + \mu)$$

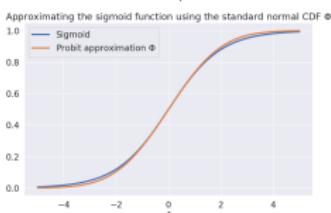


3. Probit approximation

$$\sigma(y) \approx \Phi\left(y \sqrt{\frac{\pi}{8}}\right)$$

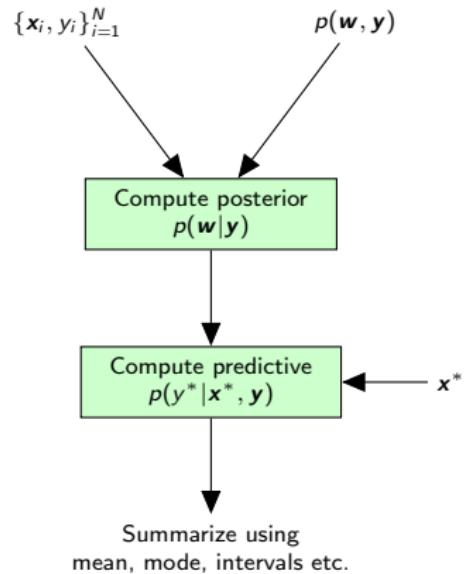


where Φ is the CDF of the standard normal



Let's zoom out and summarize

- We introduced *logistic regression* as a *discriminative* approach for binary classification
- We saw to use the *Laplace approximation* to approximate the *posterior* of the weights
- We briefly discussed three strategies to compute *the predictive distribution*
 1. Sampling
 2. Numerical integration
 3. Probit approximation



02477 – Bayesian Machine Learning: Lecture 5

Michael Riis Andersen

Technical University of Denmark,
DTU Compute, Department of Applied Math and Computer Science

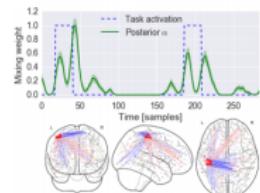
Outline

- 1 Towards prior distributions for function spaces
- 2 A visual approach towards Gaussian process regression
- 3 Gaussian process regression
- 4 Covariance functions
- 5 Hyperparameters and the marginal likelihood

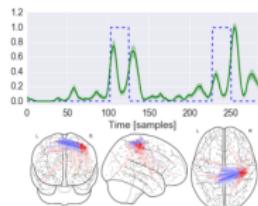
Multitude of Gaussian processes applications

■ Regression (supervised learning)

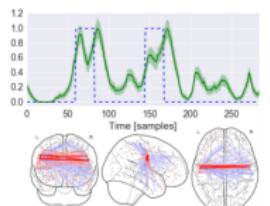
- Time series analysis
- EEG brain imaging
- Survival analysis for cancer data
- Predicting rainfall
- Robot dynamics
- ...



(a) Right hand tapping



(c) Left hand tapping



(b) Tongue wagging

■ Classification (supervised learning)

- Recognizing human movements
- Brain decoding
- ...

- Used as building block in more complex models
- Dimensionality reduction (unsupervised learning)
- Optimization of black functions (Bayesian optimization)
- Numerical integration (Bayesian quadrature)
- Solving differential equations (probabilistic numerics)

Towards prior distributions for function spaces

Parametric models

- In week 3, we studied linear models of the form

$$y_n = f_n + e_n = \phi(\mathbf{x}_n)^T \mathbf{w} + e_n$$

- In week 4, we studied Bayesian logistic regression

$$y_n | f_n \sim Ber(\sigma(f_n))$$

$$f_n = \phi(\mathbf{x}_n)^T \mathbf{w}$$

- Typical workflow

1. Specify prior $p(\mathbf{w})$ and likelihood $p(\mathbf{y}|\mathbf{w})$
2. Calculate posterior distribution $p(\mathbf{w}|\mathbf{y})$
3. Make predictions based on the predictive distribution $p(y^*|\mathbf{y}, \mathbf{x}^*)$

- All we care about is the parameters \mathbf{w} - Once we have calculated the posterior distribution $p(\mathbf{w}|\mathbf{y})$, we don't need the training data $\mathcal{D} = \{\mathbf{X}, \mathbf{y}\}$ anymore

- Linear and logistic regression are both *parametric models*: probability distributions indexed by finite dimensional parameters

From parameters to functions I

- Our linear model

$$y_n = f_n + e_n = \phi(\mathbf{x}_n)^T \mathbf{w} + e_n$$

- Our goal was to learn the latent function

$$f_n = \phi(\mathbf{x}_n)^T \mathbf{w}$$

- We focussed on \mathbf{w} and the joint distribution via the product rule

$$p(\mathbf{y}, \mathbf{w}) = p(\mathbf{y}|\mathbf{w})p(\mathbf{w})$$

- Let \mathbf{f} denote the function values, i.e. $\mathbf{f} = [f(\phi(\mathbf{x}_1)) \quad f(\phi(\mathbf{x}_2)) \quad \dots \quad f(\phi(\mathbf{x}_N))]$

$$p(\mathbf{y}, \mathbf{f}, \mathbf{w}) = p(\mathbf{y}|\mathbf{f})p(\mathbf{f}|\mathbf{w})p(\mathbf{w})$$

- The model is the same - we can recover the old model formulation via the sum rule

$$p(\mathbf{y}, \mathbf{w}) = \int p(\mathbf{y}|\mathbf{f})p(\mathbf{f}|\mathbf{w})p(\mathbf{w})d\mathbf{f}$$

From parameters to functions II

- The augmented model

$$p(\mathbf{y}, \mathbf{f}, \mathbf{w}) = p(\mathbf{y}|\mathbf{f})p(\mathbf{f}|\mathbf{w})p(\mathbf{w})$$

- What if we integrate out the parameters instead?

$$p(\mathbf{y}, \mathbf{f}) = p(\mathbf{y}|\mathbf{f}) \int p(\mathbf{f}|\mathbf{w})p(\mathbf{w})d\mathbf{w} = p(\mathbf{y}|\mathbf{f})p(\mathbf{f})$$

where $p(\mathbf{f}) = \int p(\mathbf{f}|\mathbf{w})p(\mathbf{w})d\mathbf{w}$

- Let's study the distribution of $\mathbf{f} = \Phi\mathbf{w}$ for our Gaussian prior on $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \alpha^{-1}\mathbf{I})$

$$p(\mathbf{f}) = \int p(\mathbf{f}|\mathbf{w})p(\mathbf{w})d\mathbf{w} = \int p(\mathbf{f}|\mathbf{w})\mathcal{N}(\mathbf{w}|\mathbf{0}, \alpha^{-1}\mathbf{I})d\mathbf{w}$$

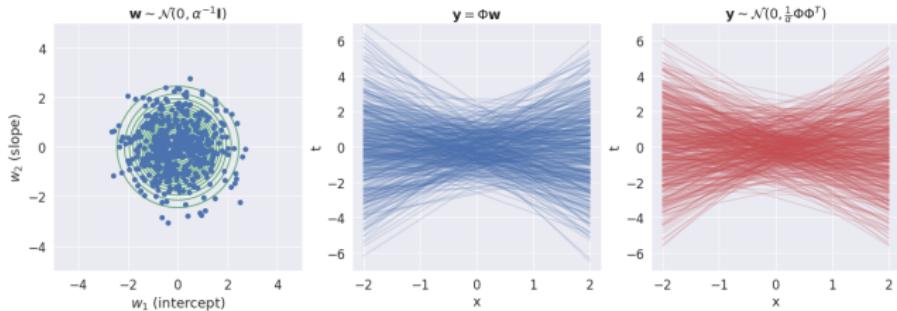
- We could do the integral directly, let's use this result instead

$$\mathbf{x} \sim \mathcal{N}(\mathbf{m}, \mathbf{V}) \quad \Rightarrow \quad \mathbf{a} + \mathbf{Bx} \sim \mathcal{N}(\mathbf{a} + \mathbf{Bm}, \mathbf{BV}\mathbf{B}^T)$$

- What is the distribution of \mathbf{f} ?

Changing perspective from weight space to function space

$$p(\mathbf{f}) = \int p(\mathbf{f}|\mathbf{w})p(\mathbf{w})d\mathbf{w} = \int p(\mathbf{f}|\mathbf{w})\mathcal{N}(\mathbf{w}|\mathbf{0}, \alpha^{-1}\mathbf{I})d\mathbf{w} = \mathcal{N}(\mathbf{f}|\mathbf{0}, \alpha^{-1}\Phi\Phi^T)$$



- Two ways to generate samples of $\mathbf{f} \sim p(\mathbf{f})$
- Weight space-perspective
 - Step 1: Generate a sample $\mathbf{w}^{(i)} \sim p(\mathbf{w})$
 - Step 2: Compute $\mathbf{f}^{(i)} = \Phi\mathbf{w}^{(i)}$
- Function space-perspective
 - Step 1: Generate a sample $\mathbf{f}^{(i)} \sim \mathcal{N}(\mathbf{0}, \alpha^{-1}\Phi\Phi^T)$

A closer look at the covariance

- A prior on linear functions: $p(\mathbf{f}) = \mathcal{N}(\mathbf{f}|\mathbf{0}, \mathbf{K})$, where $\mathbf{K} = \frac{1}{\alpha} \mathbf{\Phi} \mathbf{\Phi}^T$
- A closer look on the covariance between f_i and f_j

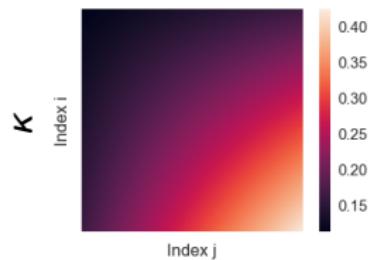
$$\begin{aligned}\mathbf{K}_{ij} &= \text{cov}(y_i, y_j) = \text{cov}(f(\mathbf{x}_i), f(\mathbf{x}_j)) \\ &= \text{cov}(\phi(\mathbf{x}_i)^T \mathbf{w}, \phi(\mathbf{x}_j)^T \mathbf{w}) \\ &= \phi(\mathbf{x}_i)^T \text{cov}(\mathbf{w}, \mathbf{w}) \phi(\mathbf{x}_j) \\ &= \phi(\mathbf{x}_i)^T \mathbb{V}(\mathbf{w}) \phi(\mathbf{x}_j) \\ &= \phi(\mathbf{x}_i)^T \frac{1}{\alpha} \mathbf{I} \phi(\mathbf{x}_j) \\ &= \frac{1}{\alpha} \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j) \\ &\equiv k(\mathbf{x}_i, \mathbf{x}_j)\end{aligned}$$

- What happens if we change the *covariance function* k ?

Covariance functions

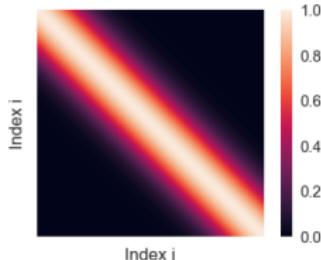
Linear

$$k(x_i, x_j) = \frac{1}{\alpha} \phi(x_i)^T \phi(x_j)$$



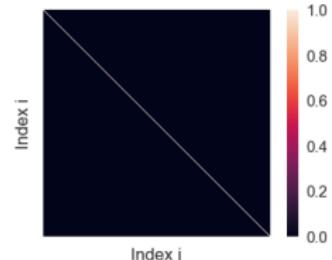
Squared exponential

$$k(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|^2}{2\ell^2}\right)$$

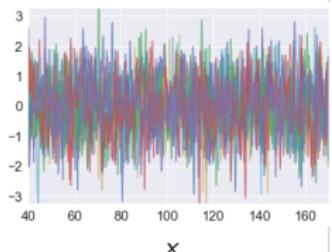
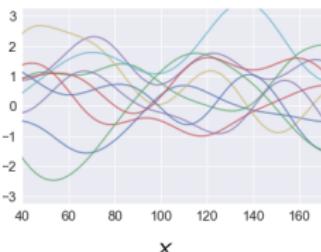
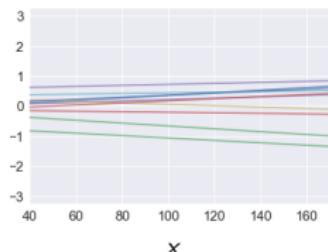


White noise

$$k(x_i, x_j) = \delta(x_i - x_j)$$



$f \sim \mathcal{N}(0, K)$



The form of the covariance function determines the characteristics of the functions

The big picture: Summary so far

1. We started with a Bayesian linear model

$$p(\mathbf{y}, \mathbf{w}) = p(\mathbf{y}|\mathbf{w})p(\mathbf{w})$$

2. We introduced \mathbf{f} into the model and marginalized over the weights \mathbf{w}

$$p(\mathbf{y}, \mathbf{f}) = \int p(\mathbf{y}|\mathbf{f})p(\mathbf{f}|\mathbf{w})p(\mathbf{w})d\mathbf{w} = p(\mathbf{y}|\mathbf{f})p(\mathbf{f})$$

3. This gave us a prior for linear functions in function space $p(\mathbf{f})$, where the covariance function for \mathbf{f} was given by

$$k(\mathbf{x}_i, \mathbf{x}_j) = \frac{1}{\alpha} \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$$

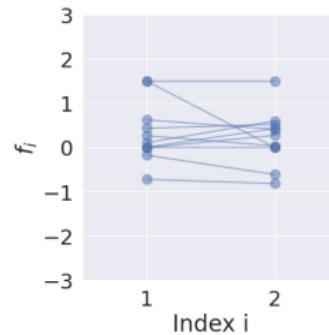
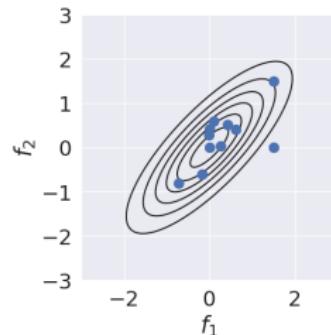
4. By changing the form of the covariance function $k(\mathbf{x}_i, \mathbf{x}_j)$, we can model much more interesting functions

A visual approach towards Gaussian process regression

Visualizing samples in higher dimensions

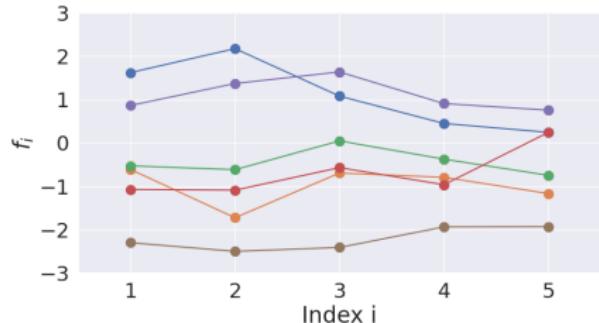
- How can a multivariate normal distribution represent functions?
- Visualizations in 2D

$$\boldsymbol{\kappa} = \begin{bmatrix} 1 & 0.8 \\ 0.8 & 1 \end{bmatrix}$$



Visualizing samples in higher dimensions

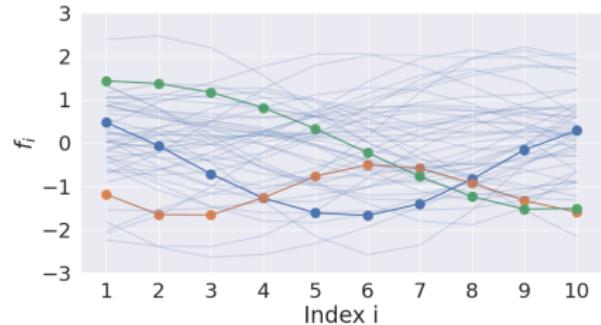
■ Visualizations in 5D



$$\boldsymbol{\kappa} = \begin{bmatrix} 1 & 0.8^1 & 0.8^2 & 0.8^3 & 0.8^4 \\ 0.8^1 & 1 & 0.8^1 & 0.8^2 & 0.8^3 \\ 0.8^2 & 0.8^1 & 1 & 0.8^1 & 0.8^2 \\ 0.8^3 & 0.8^2 & 0.8^1 & 1 & 0.8^1 \\ 0.8^4 & 0.8^3 & 0.8^2 & 0.8^1 & 1 \end{bmatrix}$$

Visualizing samples in higher dimensions

■ Visualizations in 10D



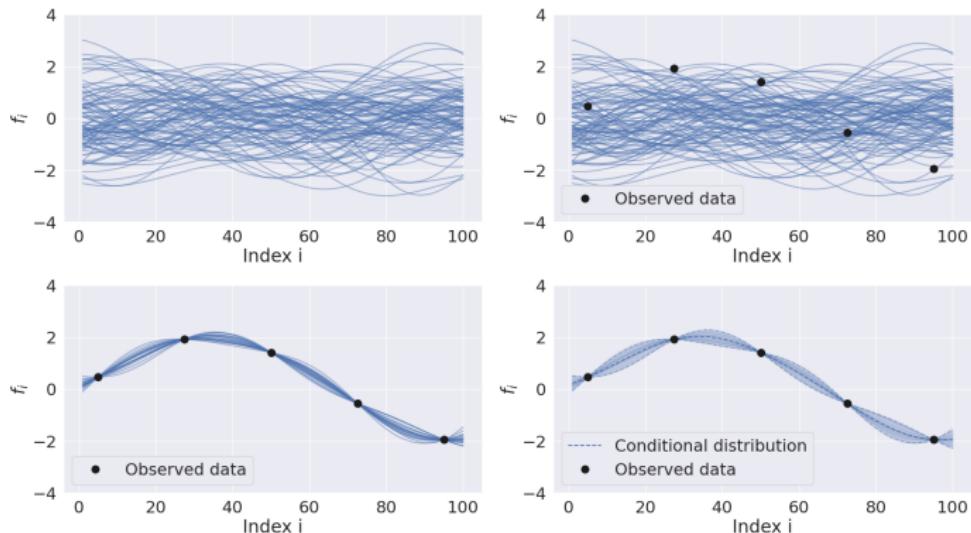
$$\boldsymbol{\kappa} = \begin{bmatrix} 1 & 0.8^1 & 0.8^2 & \dots & 0.8^9 \\ 0.8^1 & 1 & 0.8^1 & & \vdots \\ 0.8^2 & 0.8^1 & 1 & & \vdots \\ \vdots & & & \ddots & \vdots \\ 0.8^9 & \dots & \dots & \dots & 1 \end{bmatrix}$$

Conditioning

- So far, we have seen samples from the distribution $p(\mathbf{f}) = \mathcal{N}(\mathbf{f} | \mathbf{0}, \mathbf{K})$
- We can also write $p(\mathbf{f}) = p(f_1, \mathbf{f}_{2:10})$
- We now observe $f_1 = 0$
- Let's sample from the conditional distribution $p(\mathbf{f}_{2:10} | f_1 = 0)$

Conditioning II

- Let's now consider a case with $f \in \mathbb{R}^{100}$ dimensions with 5 observations



- Informally: We can think functions as vectors with infinite dimensions
- Using conditioning in multivariate Gaussian distributions, we can do non-linear regression!

Formal definitions

Definition of the multivariate Gaussian distribution

A random vector $\mathbf{x} = [x_1, x_2, \dots, x_D]$ is said to have the **multivariate Gaussian distribution** if all linear combinations of \mathbf{x} are (univariate) Gaussian distributed:

$$f = a_1x_1 + a_2x_2 + \dots + a_Dx_D \sim \mathcal{N}(m, v)$$

for all $a \in \mathbb{R}^D$

Definition of Gaussian process

A **Gaussian process** (GP) is a collection of random variables, any finite number of which have a joint Gaussian distribution.

Notation and characterization

- We'll use the notation

$$f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}'))$$

- A Gaussian process can be considered as a prior distribution over functions $f : \mathcal{X} \rightarrow \mathbb{R}$ (the domain \mathcal{X} is typically \mathbb{R}^D)
- A Gaussian process is completely characterized by its mean function $m(\mathbf{x})$ and its covariance function $k(\mathbf{x}, \mathbf{x}')$.

$$m(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})]$$

$$k(\mathbf{x}, \mathbf{x}') = \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))]$$

- This means that $f(\mathbf{x})$ and $f(\mathbf{x}')$ are jointly Gaussian distributed with covariance $k(\mathbf{x}, \mathbf{x}')$
- Not all functions are valid covariance functions - more on that later

Gaussian process regression

Recall: Linear Gaussian-systems in general (see Section 3.3 in Murphy1)

- For *linear* systems: the Gaussian distribution is *conjugate* to itself
- The *posterior* for a *linear* Gaussian model with Gaussian prior is also *Gaussian*

$$p(\mathbf{y}|\mathbf{z}) = \mathcal{N}(\mathbf{y}|\mathbf{W}\mathbf{z} + \mathbf{b}, \Sigma_y) \quad p(\mathbf{z}) = \mathcal{N}(\mathbf{z}|\boldsymbol{\mu}_z, \Sigma_z)$$

- The *joint* distribution $p(\mathbf{z}, \mathbf{y}) = \mathcal{N}\left(\begin{bmatrix} \mathbf{z} \\ \mathbf{y} \end{bmatrix} | \boldsymbol{\mu}, \boldsymbol{\Sigma}\right)$
- $$\boldsymbol{\mu} = \begin{bmatrix} \boldsymbol{\mu}_z \\ \mathbf{W}\boldsymbol{\mu}_z + \mathbf{b} \end{bmatrix} \quad \boldsymbol{\Sigma} = \begin{bmatrix} \Sigma_z & \Sigma_z \mathbf{W}^T \\ \mathbf{W}\Sigma_z & \Sigma_y + \mathbf{W}\Sigma_z\mathbf{W}^T \end{bmatrix}$$

- The *posterior* distribution of \mathbf{z} given \mathbf{y}

$$p(\mathbf{z}|\mathbf{y}) = \mathcal{N}(\mathbf{z}|\boldsymbol{\mu}_{z|y}, \Sigma_{z|y})$$

$$\Sigma_{z|y}^{-1} = \Sigma_z^{-1} + \mathbf{W}^T \Sigma_y \mathbf{W}$$

$$\boldsymbol{\mu}_{z|y} = \Sigma_{z|y} \left[\mathbf{W}^T \Sigma_y^{-1} (\mathbf{y} - \mathbf{b}) + \Sigma_z^{-1} \boldsymbol{\mu}_z \right]$$

- The *marginal* distribution \mathbf{y}

$$p(\mathbf{y}) = \mathcal{N}(\mathbf{y}|\mathbf{W}\boldsymbol{\mu}_z + \mathbf{b}, \Sigma_y + \mathbf{W}\Sigma_z\mathbf{W}^T)$$

Conditioning for multivariate Gaussians (Murphy1 Section 3.2.3)

- Suppose $\mathbf{y} = (\mathbf{y}_1, \mathbf{y}_2)$ is jointly Gaussian $\mathcal{N}(\mathbf{y}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$ with mean and covariance

$$\boldsymbol{\mu} = \begin{bmatrix} \boldsymbol{\mu}_1 \\ \boldsymbol{\mu}_2 \end{bmatrix}, \boldsymbol{\Sigma} = \begin{bmatrix} \boldsymbol{\Sigma}_{11} & \boldsymbol{\Sigma}_{12} \\ \boldsymbol{\Sigma}_{21} & \boldsymbol{\Sigma}_{22} \end{bmatrix}$$

- The precision matrix $\boldsymbol{\Lambda}$

$$\boldsymbol{\Lambda} = \boldsymbol{\Sigma}^{-1} = \begin{bmatrix} \boldsymbol{\Lambda}_{11} & \boldsymbol{\Lambda}_{12} \\ \boldsymbol{\Lambda}_{21} & \boldsymbol{\Lambda}_{22} \end{bmatrix}$$

- The marginals are given by

$$p(\mathbf{y}_1) = \mathcal{N}(\mathbf{y}_1 | \boldsymbol{\mu}_1, \boldsymbol{\Sigma}_{11})$$

$$p(\mathbf{y}_2) = \mathcal{N}(\mathbf{y}_2 | \boldsymbol{\mu}_2, \boldsymbol{\Sigma}_{22})$$

- The conditional

$$p(\mathbf{y}_1 | \mathbf{y}_2) = \mathcal{N}(\mathbf{y}_1 | \boldsymbol{\mu}_{1|2}, \boldsymbol{\Sigma}_{1|2})$$

$$\boldsymbol{\mu}_{1|2} = \boldsymbol{\mu}_1 + \boldsymbol{\Sigma}_{12} \boldsymbol{\Sigma}_{22}^{-1} (\mathbf{y}_2 - \boldsymbol{\mu}_2)$$

$$\boldsymbol{\Sigma}_{1|2} = \boldsymbol{\Sigma}_{11} - \boldsymbol{\Sigma}_{12} \boldsymbol{\Sigma}_{22}^{-1} \boldsymbol{\Sigma}_{21} = \boldsymbol{\Lambda}_{11}^{-1}$$

Gaussian process regression I

- Our model

$$y_n = f(\mathbf{x}_n) + e_n$$

- Likelihood for all datapoints (assuming homoscedastic noise)

$$p(\mathbf{y}|\mathbf{f}) = \prod_{n=1}^N \mathcal{N}(y_n|f_n, \beta^{-1}) = \mathcal{N}(\mathbf{y}|\mathbf{f}, \beta^{-1}\mathbf{I})$$

- We impose a prior directly on the *function values* $\mathbf{f} = [f(\mathbf{x}_1) \quad f(\mathbf{x}_2) \quad \dots \quad f(\mathbf{x}_N)]$

$$p(\mathbf{f}) = \mathcal{N}(\mathbf{f}|\mathbf{0}, \mathbf{K}) \quad \text{for} \quad (\mathbf{K})_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$$

- Goal: compute predictive distribution for $y^* = y(\mathbf{x}^*)$ given data \mathbf{y} , i.e. $p(y^*|\mathbf{y}, \mathbf{x}^*)$

- Two-step strategy

1. Calculate the joint Gaussian distribution $p(\mathbf{y}, y^*|\mathbf{x}^*)$
2. Use rule for conditioning in Gaussian distributions to compute $p(y^*|\mathbf{y}, \mathbf{x}^*)$

Gaussian process regression II

- Recall: General Linear Gaussian systems (Murphy1 page 86-77).

If

$$p(\mathbf{z}) = \mathcal{N}(\mathbf{z} | \boldsymbol{\mu}_z, \Sigma_z)$$
$$p(\mathbf{y} | \mathbf{z}) = \mathcal{N}(\mathbf{y} | \mathbf{W}\mathbf{z} + \mathbf{b}, \Sigma_y)$$

then

$$p(\mathbf{y}) = \int p(\mathbf{y} | \mathbf{z}) p(\mathbf{z}) d\mathbf{z} = \mathcal{N}(\mathbf{y} | \mathbf{W}\boldsymbol{\mu}_z + \mathbf{b}, \Sigma_y + \mathbf{W}\Sigma_z\mathbf{W}^T)$$

- We can compute the marginal distribution of \mathbf{y} using the *sum rule*

$$\begin{aligned} p(\mathbf{y}) &= \int p(\mathbf{y} | \mathbf{f}) p(\mathbf{f}) d\mathbf{f} \\ &= \int \mathcal{N}(\mathbf{y} | \mathbf{f}, \beta^{-1} \mathbf{I}) \mathcal{N}(\mathbf{f} | \mathbf{0}, \mathbf{K}) d\mathbf{f} \\ &= \mathcal{N}(\mathbf{y} | ?, ?) \end{aligned}$$

- Spend 5 minutes calculating the mean and variance of $p(\mathbf{y})$

$$p(\mathbf{y}) = \mathcal{N}(\mathbf{y} | ?, ?)$$

Gaussian process regression III

- The distribution of $\mathbf{y} \in \mathbb{R}^N$

$$p(\mathbf{y}) = \mathcal{N}(\mathbf{y} | \mathbf{0}, \mathbf{C}) \quad \text{for} \quad \mathbf{C} = \mathbf{K} + \beta^{-1} \mathbf{I}$$

- Let $\tilde{\mathbf{y}} = [y(x^*) \quad \mathbf{y}]^T \in \mathbb{R}^{N+1}$, then

$$p(\tilde{\mathbf{y}}) = \mathcal{N}(\tilde{\mathbf{y}} | \mathbf{0}, \tilde{\mathbf{C}}) \quad \text{for} \quad \tilde{\mathbf{C}} = \begin{bmatrix} c & \mathbf{k} \\ \mathbf{k}^T & \mathbf{C} \end{bmatrix}$$

where

$$c = k(x^*, x^*) + \beta^{-1}$$

$$\mathbf{k} = [k(x^*, x_1) \quad k(x^*, x_2) \quad \dots \quad k(x^*, x_N)]$$

Suppose $\mathbf{y} = (y_1, y_2)$ is jointly Gaussian
 $\mathcal{N}(\mathbf{y} | \boldsymbol{\mu}, \boldsymbol{\Sigma})$

- What is the mean and variance for the following distribution?

$$\boldsymbol{\mu} = \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}, \boldsymbol{\Sigma} = \begin{bmatrix} \boldsymbol{\Sigma}_{11} & \boldsymbol{\Sigma}_{12} \\ \boldsymbol{\Sigma}_{21} & \boldsymbol{\Sigma}_{22} \end{bmatrix}$$

$$p(y^* | \mathbf{y}) = \mathcal{N}(y^* | \mu_{y^*|y}, \sigma_{y^*|y}^2)$$

$$p(y_1 | y_2) = \mathcal{N}(y_1 | \mu_{1|2}, \boldsymbol{\Sigma}_{1|2})$$

$$\mu_{1|2} = \mu_1 + \boldsymbol{\Sigma}_{12} \boldsymbol{\Sigma}_{22}^{-1} (y_2 - \mu_2)$$

$$\boldsymbol{\Sigma}_{1|2} = \boldsymbol{\Sigma}_{11} - \boldsymbol{\Sigma}_{12} \boldsymbol{\Sigma}_{22}^{-1} \boldsymbol{\Sigma}_{21}$$

Example

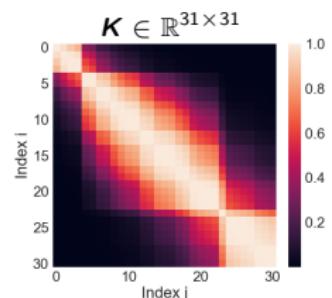
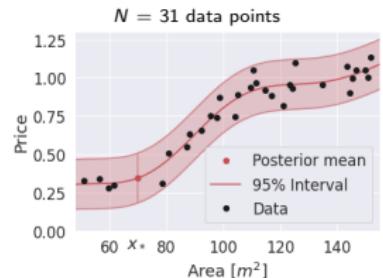
Key equations for Gaussian process regression

$$p(y^* | \mathbf{y}) = \mathcal{N} \left(y^* | \mu_{y^* | \mathbf{y}}, \sigma_{y^* | \mathbf{y}}^2 \right)$$

$$\mu_{y^* | \mathbf{y}} = \mathbf{k} (\mathbf{K} + \beta^{-1} \mathbf{I})^{-1} \mathbf{y}$$

$$\sigma_{y^* | \mathbf{y}}^2 = c - \mathbf{k} (\mathbf{K} + \beta^{-1} \mathbf{I})^{-1} \mathbf{k}^T$$

- Predict $y^* = f(x_*) + e^*$ for test input $x_* = 70$
- Observation vector $\mathbf{y} = [y_1, y_2, \dots, y_{31}]^T \in \mathbb{R}^{31 \times 1}$
- $k(x, x') = \text{cov}(f(x), f(x')) = \exp \left[-\frac{(x-x')^2}{2 \cdot 20^2} \right]$
- Covariance matrix for training data: $[\mathbf{K}]_{ij} = k(x_i, x_j)$
- Cov. between test and training $[\mathbf{k}]_j = k(x_*, x_j)$
- Covariance of test point $y^* = y(x_*)$: $c = k(x_*, x_*) + \beta^{-1}$
- Now we have all the ingredients for the key equations



Gaussian process intuition

- Gaussian process implements the assumption

$$\mathbf{x} \approx \mathbf{x}' \quad \Rightarrow \quad f(\mathbf{x}) \approx f(\mathbf{x}')$$

- In words: If the inputs are similar, the outputs should be similar as well.

- Using the squared exponential covariance function as example

$$k(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2}\right)$$

- Then covariance between $f(\mathbf{x})$ and $f(\mathbf{x})'$ is given by

$$\text{cov}[f(\mathbf{x}), f(\mathbf{x}')]=k(\mathbf{x}, \mathbf{x}')=\exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2}\right)$$

- Note: the covariance between outputs are given in terms of the inputs

True or false?

Key equations for Gaussian process regression

$$p(y^* | \mathbf{y}) = \mathcal{N} \left(y^* | \mu_{y^* | \mathbf{y}}, \sigma_{y^* | \mathbf{y}}^2 \right)$$

$$\mu_{y^* | \mathbf{y}} = \mathbf{k} (\mathbf{K} + \beta^{-1} \mathbf{I})^{-1} \mathbf{y}$$

$$\sigma_{y^* | \mathbf{y}}^2 = c - \mathbf{k} (\mathbf{K} + \beta^{-1} \mathbf{I})^{-1} \mathbf{k}^T$$

True or false?

- Spend 5 minutes on the DTU Learn quiz: “Lecture 5: Key equations for GP Regression.”

Covariance functions

Covariance functions

- A covariance function $k : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ maps a pair of inputs $\mathbf{x}_1, \mathbf{x}_2 \in \mathcal{X}$ from some input space \mathcal{X} to the real line \mathbb{R}

- Recall: the covariance / kernel matrix is given by

$$\mathbf{K}_{ij} = \text{cov}(f(\mathbf{x}_i), f(\mathbf{x})_j) = k(\mathbf{x}_i, \mathbf{x}_j)$$

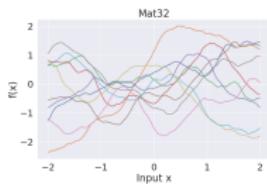
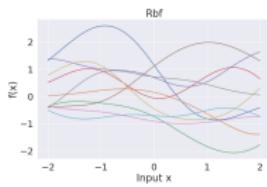
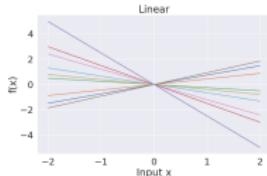
- Covariance functions must be symmetric & Positive Semi-Definite such that

$$(\text{Symmetric}) \quad \mathbf{K} = \mathbf{K}^T$$

$$(\text{PSD}) \quad \forall \mathbf{x} \neq 0 : \quad \mathbf{x}^T \mathbf{K} \mathbf{x} \geq 0$$

- Must hold for all possible data sets $\{\mathbf{x}_n\}_{n=1}^N \subset \mathcal{X}$ in the input space \mathcal{X}

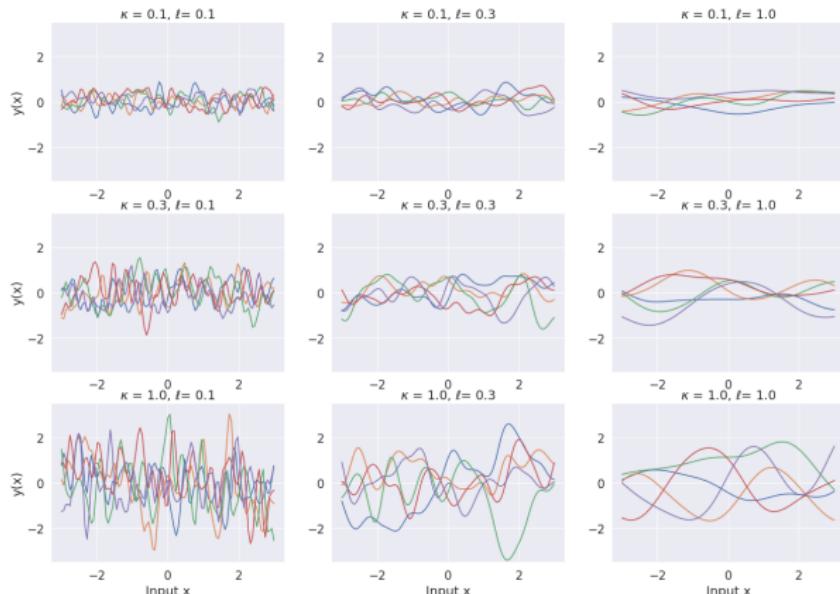
- Covariance functions as prior information



The squared exponential kernel - prior samples

$$k(\mathbf{x}, \mathbf{x}') = \kappa^2 \exp \left[-\frac{\|\mathbf{x} - \mathbf{x}'\|_2^2}{2\ell^2} \right]$$

Parameter ℓ is called the *lengthscale* and parameter κ is called the *magnitude*



Constructing new kernels from old ones

Techniques for Constructing New Kernels.

Given valid kernels $k_1(\mathbf{x}, \mathbf{x}')$ and $k_2(\mathbf{x}, \mathbf{x}')$, the following new kernels will also be valid:

$$k(\mathbf{x}, \mathbf{x}') = ck_1(\mathbf{x}, \mathbf{x}') \quad (6.13)$$

$$k(\mathbf{x}, \mathbf{x}') = f(\mathbf{x})k_1(\mathbf{x}, \mathbf{x}')f(\mathbf{x}') \quad (6.14)$$

$$k(\mathbf{x}, \mathbf{x}') = q(k_1(\mathbf{x}, \mathbf{x}')) \quad (6.15)$$

$$k(\mathbf{x}, \mathbf{x}') = \exp(k_1(\mathbf{x}, \mathbf{x}')) \quad (6.16)$$

$$k(\mathbf{x}, \mathbf{x}') = k_1(\mathbf{x}, \mathbf{x}') + k_2(\mathbf{x}, \mathbf{x}') \quad (6.17)$$

$$k(\mathbf{x}, \mathbf{x}') = k_1(\mathbf{x}, \mathbf{x}')k_2(\mathbf{x}, \mathbf{x}') \quad (6.18)$$

$$k(\mathbf{x}, \mathbf{x}') = k_3(\phi(\mathbf{x}), \phi(\mathbf{x}')) \quad (6.19)$$

$$k(\mathbf{x}, \mathbf{x}') = \mathbf{x}^T \mathbf{A} \mathbf{x}' \quad (6.20)$$

$$k(\mathbf{x}, \mathbf{x}') = k_a(\mathbf{x}_a, \mathbf{x}'_a) + k_b(\mathbf{x}_b, \mathbf{x}'_b) \quad (6.21)$$

$$k(\mathbf{x}, \mathbf{x}') = k_a(\mathbf{x}_a, \mathbf{x}'_a)k_b(\mathbf{x}_b, \mathbf{x}'_b) \quad (6.22)$$

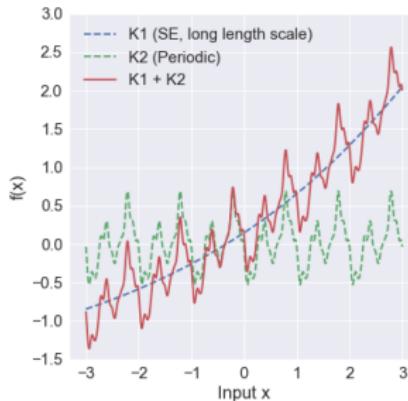
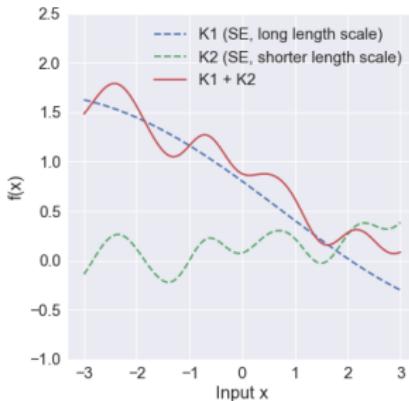
where $c > 0$ is a constant, $f(\cdot)$ is any function, $q(\cdot)$ is a polynomial with nonnegative coefficients, $\phi(\mathbf{x})$ is a function from \mathbf{x} to \mathbb{R}^M , $k_3(\cdot, \cdot)$ is a valid kernel in \mathbb{R}^M , \mathbf{A} is a symmetric positive semidefinite matrix, \mathbf{x}_a and \mathbf{x}_b are variables (not necessarily disjoint) with $\mathbf{x} = (\mathbf{x}_a, \mathbf{x}_b)$, and k_a and k_b are valid kernel functions over their respective spaces.

Additive kernels

- Adding two SEs kernels to model long term trends (long length scale) and short term fluctuations (short length scale)

$$k(\mathbf{x}, \mathbf{x}') = \kappa_1^2 \exp\left[-\frac{\|\mathbf{x} - \mathbf{x}'\|_2^2}{2\ell_1^2}\right] + \kappa_2^2 \exp\left[-\frac{\|\mathbf{x} - \mathbf{x}'\|_2^2}{2\ell_2^2}\right]$$

- Adding SE and period kernels to model long term trends (long length scale) and periodic fluctuations



Hyperparameters and the marginal likelihood

The marginal likelihood I

- Let θ denote all hyperparameters, then marginal likelihood for Gaussian likelihood

$$\begin{aligned} p(\mathbf{y}|\theta) &= \int p(\mathbf{y}|\mathbf{f})p(\mathbf{f}|\theta_K)d\mathbf{f} \\ &= \int \mathcal{N}(\mathbf{y}|\mathbf{f}, \beta^{-1}\mathbf{I}) \mathcal{N}(\mathbf{f}|\mathbf{0}, \mathbf{K}) d\mathbf{f} \\ &= \mathcal{N}(\mathbf{y}|\mathbf{0}, \beta^{-1}\mathbf{I} + \mathbf{K}) \end{aligned}$$

- We can tune the hyperparameters of the model by optimizing the marginal likelihood as we did for linear regression
 - Hyperparameters of the likelihood (e.g. β or σ)
 - Hyperparameters of the kernel θ_K (e.g. lengthscales and magnitudes)
- In practice, we compute the gradient of $p(\mathbf{y}|\theta)$ wrt. θ and use numerical optimization

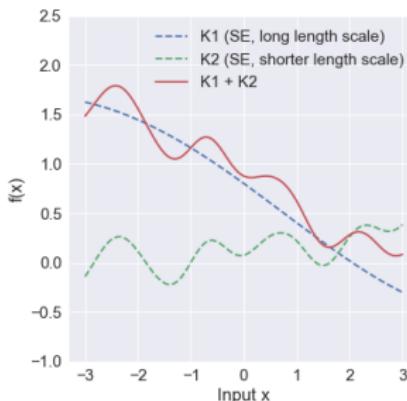
The marginal likelihood II

- Suppose we have 5 hyperparameters in total

$$\theta = \{\sigma, \kappa_1, \ell_1, \kappa_2, \ell_2\}$$

- Suppose we want to estimate those using 10-fold cross-validation and test out 10 values for each hyperparameter. How many times do we need to train the model?

$$10 \cdot 10^5 = 10^6$$



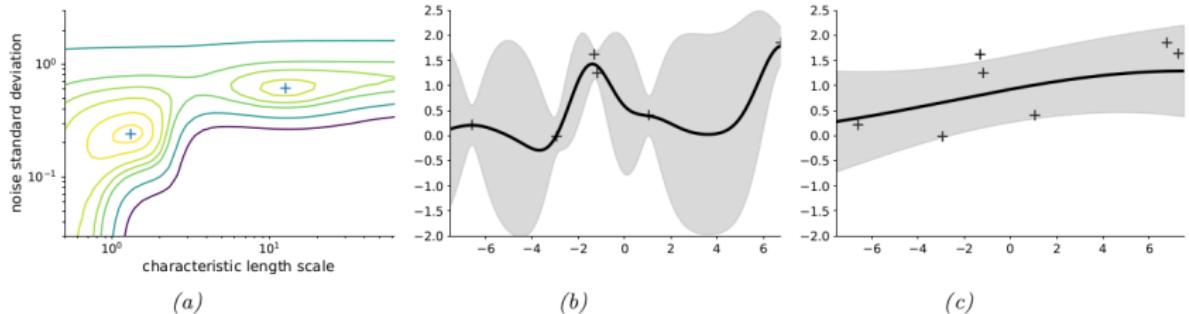
The marginal likelihood III

- The gradients of the marginal likelihood wrt. hyperparameters are given by

$$\frac{\partial}{\partial \theta_j} \log p(\mathbf{y}|\boldsymbol{\theta}) = \frac{1}{2} \text{tr} \left((\boldsymbol{\alpha} \boldsymbol{\alpha}^T - \boldsymbol{\mathcal{K}}^{-1}) \frac{\partial \boldsymbol{\mathcal{K}}}{\partial \theta_j} \right),$$

where $\boldsymbol{\alpha} = \boldsymbol{\mathcal{K}}^{-1} \mathbf{y}$ and $\frac{\partial \boldsymbol{\mathcal{K}}}{\partial \theta_j}$ depends on the specific choice of kernel.

- $\log p(\mathbf{y}|\boldsymbol{\theta})$ is also multimodal wrt. $\boldsymbol{\theta}$



From the Murphy1 book (p. 578)

The marginal likelihood: numerics

- In practice, we should avoid computing determinants and inverses!

$$\ln p(\mathbf{y} | \boldsymbol{\theta}) = -\frac{N}{2} \ln (2\pi) - \frac{1}{2} \ln |\boldsymbol{\beta}^{-1} \mathbf{I} + \boldsymbol{\kappa}| - \frac{1}{2} \mathbf{y}^T (\boldsymbol{\beta}^{-1} \mathbf{I} + \boldsymbol{\kappa})^{-1} \mathbf{y}$$

- In numpy: $\ln |0.1 \mathbf{I}_{400 \times 400}| = 0.0$, but $\ln |0.1 \mathbf{I}_{400 \times 400}| = -2302.58$ and $\exp(-2302.58) > 0$
- Step 1: Compute Cholesky factorization of $\boldsymbol{C} = \boldsymbol{\beta}^{-1} \mathbf{I} + \boldsymbol{\kappa}$ such that $\boldsymbol{C} = \mathbf{L} \mathbf{L}^T$
- Step 2: Compute the log determinant term as follows

$$\ln |\boldsymbol{C}| = \ln |\mathbf{L} \mathbf{L}^T| = \ln |\mathbf{L}| \cdot |\mathbf{L}^T| = \ln |\mathbf{L}|^2 = 2 \ln |\mathbf{L}| = 2 \ln \prod_{n=1}^N L_{nn} = 2 \sum_{n=1}^N \ln L_{nn}$$

- Step 3: Compute quadratic term as follows

$$\mathbf{y}^T \boldsymbol{C}^{-1} \mathbf{y} = \mathbf{y}^T (\mathbf{L} \mathbf{L}^T)^{-1} \mathbf{y} = \mathbf{y}^T \mathbf{L}^{-T} \mathbf{L}^{-1} \mathbf{y} = (\mathbf{L}^{-1} \mathbf{y})^T \underbrace{(\mathbf{L}^{-1} \mathbf{y})}_{=\mathbf{v}} = \mathbf{v}^T \mathbf{v}$$

- Step 4: Sum components

$$\ln p(\mathbf{y} | \boldsymbol{\theta}) = -\frac{N}{2} \ln (2\pi) - \frac{1}{2} 2 \sum_{n=1}^N \ln L_{nn} - \frac{1}{2} \mathbf{v}^T \mathbf{v}$$

- Note that we never compute the determinant or the inverse of \boldsymbol{C} directly!

Computational complexity of Gaussian Processes

Key equations for Gaussian process regression

$$p(y^* | \mathbf{y}) = \mathcal{N} \left(y^* | \mu_{y^* | \mathbf{y}}, \sigma_{y^* | \mathbf{y}}^2 \right)$$

$$\mu_{y^* | \mathbf{y}} = \mathbf{k} (\mathbf{K} + \beta^{-1} \mathbf{I})^{-1} \mathbf{y}$$

$$\sigma_{y^* | \mathbf{y}}^2 = c - \mathbf{k} (\mathbf{K} + \sigma^2 \mathbf{I})^{-1} \mathbf{k}^T$$

- Gaussian processes are *non-parametric models*
- Recall: If $\mathbf{A} \in \mathbb{R}^{N \times M}$ and $\mathbf{b} \in \mathbb{R}^M$, then the cost of computing \mathbf{Ab} is $\mathcal{O}(NM)$
- Recall: If $\mathbf{C} \in \mathbb{R}^{N \times N}$, then the cost of computing \mathbf{C}^{-1} is $\mathcal{O}(N^3)$
- What is computational complexity for computing the posterior distribution for 1 test point based on a data set with N observations? $\mathcal{O}(N^3)$
- What about the memory footprint? $\mathcal{O}(N^2)$

02477 – Bayesian Machine Learning: Lecture 6

Michael Riis Andersen

Technical University of Denmark,
DTU Compute, Department of Applied Math and Computer Science

Outline

- 1 A bit more theory on covariance functions
- 2 Gaussian processes in practice
- 3 Gaussian process classification
- 4 A bit about neural networks for probabilistic modelling

A bit more theory on covariance functions

The big picture

1. We started with a Bayesian linear model

$$p(\mathbf{y}, \mathbf{w}) = p(\mathbf{y} | \mathbf{w})p(\mathbf{w})$$

2. We introduced \mathbf{f} into the model and marginalized over the weights \mathbf{w}

$$p(\mathbf{y}, \mathbf{f}) = \int p(\mathbf{y} | \mathbf{f})p(\mathbf{f} | \mathbf{w})p(\mathbf{w})d\mathbf{w} = p(\mathbf{y} | \mathbf{f})p(\mathbf{f})$$

3. This gave us a prior for linear functions in function space $p(\mathbf{f})$,

$$p(\mathbf{f}) = \mathcal{N}(\mathbf{f} | \mathbf{0}, \alpha^{-1}\Phi\Phi^T),$$

where the covariance function for \mathbf{f} was given by

$$k(\mathbf{x}_i, \mathbf{x}_j) = \frac{1}{\alpha} \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$$

4. By changing the form of the covariance function $k(\mathbf{x}_i, \mathbf{x}_j)$, we can model much more interesting functions

Notation and characterization

- We'll use the notation

$$f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}'))$$

- A Gaussian process can be considered as a prior distribution over functions $f : \mathcal{X} \rightarrow \mathbb{R}$ (the domain \mathcal{X} is typically \mathbb{R}^D).
- A Gaussian process is completely characterized by its mean function $m(\mathbf{x})$ and its covariance function $k(\mathbf{x}, \mathbf{x}')$.

$$m(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})]$$

$$k(\mathbf{x}, \mathbf{x}') = \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))]$$

- This means that $f(\mathbf{x})$ and $f(\mathbf{x}')$ are jointly Gaussian distributed with covariance $k(\mathbf{x}, \mathbf{x}')$.
- Not all functions are valid covariance functions – more on that later.

Covariance functions

- A covariance function $k : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ maps a pair of inputs $\mathbf{x}_1, \mathbf{x}_2 \in \mathcal{X}$ from some input space \mathcal{X} to the real line \mathbb{R}
- Recall: the covariance / kernel matrix is given by

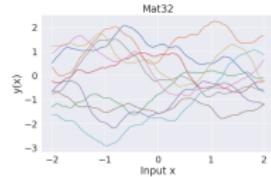
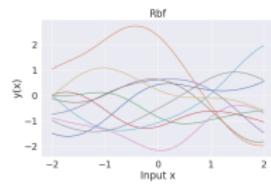
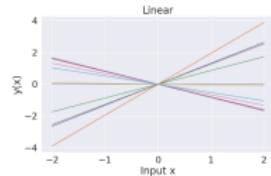
$$\mathbf{K}_{ij} = \text{cov}(y(\mathbf{x}_i), y(\mathbf{x})_j) = k(\mathbf{x}_i, \mathbf{x}_j)$$

- Covariance functions must be symmetric & Positive Semi-Definite such that

$$(\text{Symmetric}) \quad \mathbf{K} = \mathbf{K}^T$$

$$(\text{PSD}) \quad \forall \mathbf{x} \neq 0 : \quad \mathbf{x}^T \mathbf{K} \mathbf{x} \geq 0$$

- Must hold for all possible data sets $\{\mathbf{x}_n\}_{n=1}^N \subset \mathcal{X}$ in the input space \mathcal{X}
- Covariance functions as prior information
- Stationary and isotropic covariance functions

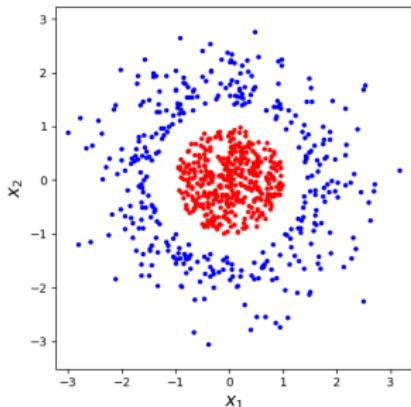


Digression: Feature expansions

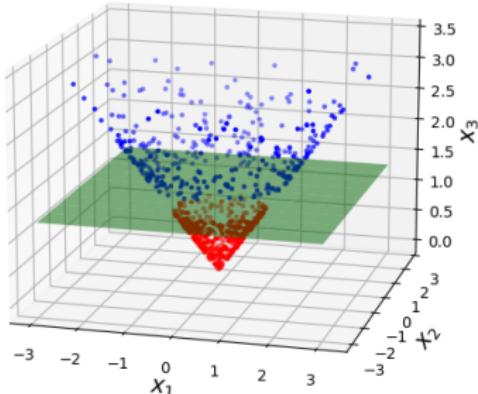
- We derived the covariance function from a linear model $f(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x})$. The choice of feature expansion $\phi(\mathbf{x})$ can be crucial.
- Example:** Binary classification problem in 2d, *not linear separable*
- Embedding \mathbf{x} in *higher dimensional space* can make the problem *linear separable*, e.g.

$$\phi(\mathbf{x}) = [x_1, x_2, \sqrt{x_1^2 + x_2^2}]$$

2d problem



3d feature space



Kernels and feature spaces I

- The linear model $f(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x})$ with Gaussian priors $p(\mathbf{w}|\mathbf{0}, \alpha^{-1} \mathbf{I})$ yields

$$\mathbf{K}_{nm} = \text{cov}(f(\mathbf{x}_n), f(\mathbf{x}_m)) = \alpha^{-1} \phi(\mathbf{x}_n)^T \phi(\mathbf{x}_m)$$

- Hence, changing the feature expansion $\phi(\mathbf{x})$ changes the covariance function.
- What about the other way around? If we adopt some covariance function, does it imply a specific feature expansion?
- Yes! (by *Mercer's theorem*).

Kernels and feature spaces II

- **Example:** Consider the following kernel for $\mathbf{x} = [x_1, x_2]^T \in \mathbb{R}^2$

$$\begin{aligned} k(\mathbf{x}, \mathbf{x}') &= (\mathbf{x}^T \mathbf{x}')^2 \\ &= (x_1 x'_1 + x_2 x'_2)^2 \\ &= x_1^2 (x'_1)^2 + x_2^2 (x'_2)^2 + 2x_1 x'_1 x_2 x'_2 \\ &= \begin{bmatrix} x_1^2 \\ x_2^2 \\ \sqrt{2}x_1 x_2 \end{bmatrix}^T \begin{bmatrix} x'_1^2 \\ x'_2^2 \\ \sqrt{2}x_1 x_2 \end{bmatrix} \end{aligned}$$

- Hence, this kernel is equivalent to embedding the 2d point \mathbf{x} in a 3D feature space, but we never explicitly construct the 3d feature representation
- How about the squared exponential kernel?

$$k(\mathbf{x}, \mathbf{x}') = \kappa^2 \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2\ell^2}\right)$$

- One can show that the implicit feature space for the squared exponential kernel is *infinite dimensional*

Gaussian processes in practice

Gaussian processes in practice

- GPs requires careful implementation to make robust and to make it scale
- Frameworks for easy Gaussian process modelling
 1. GPy
 2. GPflow
 3. GPytorch
 4. GPJax
 5. BRMS (MC Stan)
 6. ...
- Approximations and computational tools for scaling GPs to millions of data points
 1. Exact inference
 2. Variational approximation and inducing points
 3. KISS (Kernel interpolation for structured Gaussian)
 4. Basis functions approximations
 5. ...
- Regression, classification, latent variable models...

Gaussian process classification

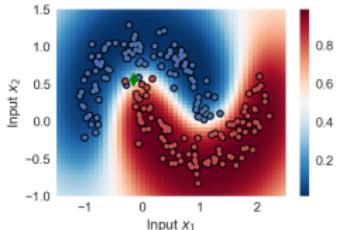
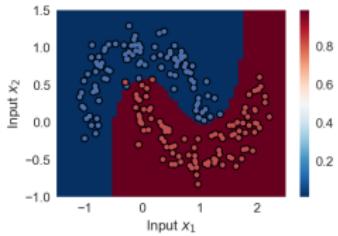
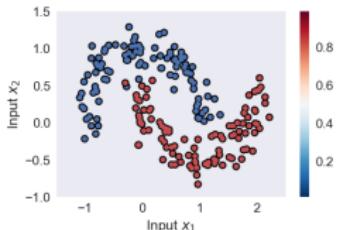
Why Gaussian processes for classification?

■ Complex decision boundaries

1. Non-linear boundary
2. Can learn complexity of decision boundary from data

■ Probabilistic classification

1. How would you classify the green point?
2. We want to model both epistemic and aleatoric uncertainty, while using highly flexible models



Gaussian processes for classification I

- Bayesian model for logistic regression

$$y_n \sim \text{Ber}(\sigma(f(\mathbf{x}_n)))$$

$$f(\mathbf{x}_n) = \phi(\mathbf{x}_n)^T \mathbf{w}$$

$$\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \alpha^{-1} \mathbf{I})$$

- Gaussian process classification

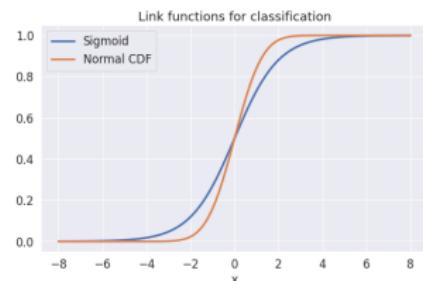
$$y_n \sim \text{Ber}(\sigma(f(\mathbf{x}_n)))$$

$$f \sim \mathcal{GP}(0, k(\mathbf{x}, \mathbf{x}'))$$

- Neural network classification

$$y_n \sim \text{Ber}(\sigma(f(\mathbf{x}_n)))$$

$$f = \text{NN}(\mathbf{x}_n | \mathbf{w})$$



- The function $\sigma : \mathbb{R} \rightarrow (0, 1)$ is called an *inverse link function*

1. Sigmoid: $\sigma(x) = \frac{1}{1 + \exp(-x)}$

2. CDF of standard normal: $\Phi(x) = \int_{-\infty}^x \mathcal{N}(x|0, 1) dx$

- Sigmoid can be more robust to outliers, but the CDF of the standard normal has appealing computational properties

Gaussian processes for classification II

- Likelihood for logistic regression (using $f_n = f(\mathbf{x}_n)$)

$$p(\mathbf{y}|\mathbf{f}) = \prod_{n=1}^N \sigma(f_n)^{y_n} (1 - \sigma(f_n))^{1-y_n}$$

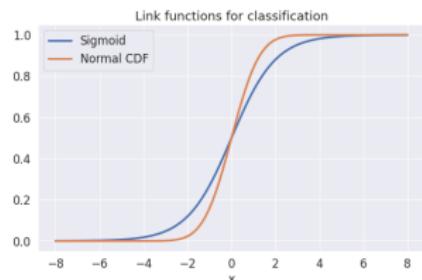
- Our Gaussian process prior

$$p(\mathbf{f}) = \mathcal{N}(\mathbf{f}|\mathbf{0}, \mathbf{K})$$

- For regression we derived the predictive distribution analytically because everything was jointly Gaussian
- The predictive distribution for classification

$$p(y^* = 1|\mathbf{y}, \mathbf{x}^*) = \int p(y^* = 1|f^*)p(f^*|\mathbf{y}, \mathbf{x}^*)df^*$$

- ... is again intractable, so we will use Laplace approximations in a 2 step-procedure
 - Approximate $p(f^*|\mathbf{y}, \mathbf{x}^*)$ using Laplace
 - Compute $p(y^*|\mathbf{y}, \mathbf{x}^*)$



Gaussian processes for classification III

- The predictive distribution for classification

$$p(y^* = 1 | \mathbf{y}, \mathbf{x}^*) = \int p(y^* = 1 | f^*) p(f^* | \mathbf{y}, \mathbf{x}^*) df^*$$

- Re-writing $p(f^* | \mathbf{y}, \mathbf{x}^*)$

$$p(f^* | \mathbf{y}, \mathbf{x}^*) = \int p(f^*, \mathbf{f} | \mathbf{y}, \mathbf{x}^*) d\mathbf{f} \quad (\text{Sum rule})$$

$$= \int p(f^* | \mathbf{f}, \mathbf{y}, \mathbf{x}^*) p(\mathbf{f} | \mathbf{y}, \mathbf{x}^*) d\mathbf{f} \quad (\text{Product rule})$$

$$= \int p(f^* | \mathbf{f}, \mathbf{x}^*) p(\mathbf{f} | \mathbf{y}) d\mathbf{f} \quad (\text{Conditional independence})$$

$$\approx \int p(f^* | \mathbf{f}, \mathbf{x}^*) q(\mathbf{f}) d\mathbf{f} \quad (\text{Laplace})$$

where $q(\mathbf{f}) = \mathcal{N}(\mathbf{f} | \mathbf{m}, \mathbf{S})$ is the Laplace approximation for $p(\mathbf{f} | \mathbf{y})$

- \mathbf{m}_i is the approximate posterior mean for $f(x_i)$ and similar for the variance S_{ii}

Gaussian processes for classification IV

- We have

$$p(f^* | \mathbf{y}, \mathbf{x}^*) = \int p(f^* | \mathbf{f}, \mathbf{x}^*) q(\mathbf{f}) d\mathbf{f}$$

- $p(f^* | \mathbf{f}, \mathbf{x}^*)$ is just a conditional Gaussian density from $p(f^*, \mathbf{f} | \mathbf{x}^*)$ (see Murphy1 p. 84 again)

$$p(f^* | \mathbf{f}, \mathbf{x}^*) = \mathcal{N}(f^* | \mathbf{k}^T \mathbf{K}^{-1} \mathbf{f}, k - \mathbf{k}^T \mathbf{K}^{-1} \mathbf{k})$$

- Therefore, we can use the equations for linear Gaussian models again (see Murphy1 Section 3.3) to derive

$$\begin{aligned} p(f^* | \mathbf{y}, \mathbf{x}^*) &= \int \mathcal{N}(f^* | \mathbf{k}^T \mathbf{K}^{-1} \mathbf{f}, k - \mathbf{k}^T \mathbf{K}^{-1} \mathbf{k}) \mathcal{N}(\mathbf{f} | \mathbf{m}, \mathbf{S}) d\mathbf{f} \\ &= \mathcal{N}(f^* | \mu_{f^*}, \sigma_{f^*}^2) \end{aligned}$$

where

$$\mu_{f^*} = \mathbf{k}^T \mathbf{K}^{-1} \mathbf{m}$$

$$\sigma_{f^*}^2 = k - \mathbf{k}^T \mathbf{K}^{-1} (\mathbf{K} - \mathbf{S}) \mathbf{K}^{-1} \mathbf{k}$$

Gaussian processes for classification V: Making predictions

- Step 1: Compute the Laplace approximation

$$p(\mathbf{f}|\mathbf{y}) \approx q(\mathbf{f}) = \mathcal{N}(\mathbf{f}|\mathbf{m}, \mathbf{S})$$

- Step 2: Posterior distribution for latent function f^* using the Laplace approximation

$$p(f^*|\mathbf{y}, \mathbf{x}^*) = \mathcal{N}(f^* | \mu_{f^*}, \sigma_{f^*}^2)$$

- Step 3: Several options for computing the predictive distribution for classification labels

$$p(y^* = 1|\mathbf{y}, \mathbf{x}^*) \approx \int p(y^* = 1|f^*) p(f^*|\mathbf{y}, \mathbf{x}^*) df^* = \int \sigma(f^*) p(f^*|\mathbf{y}, \mathbf{x}^*) df^*$$

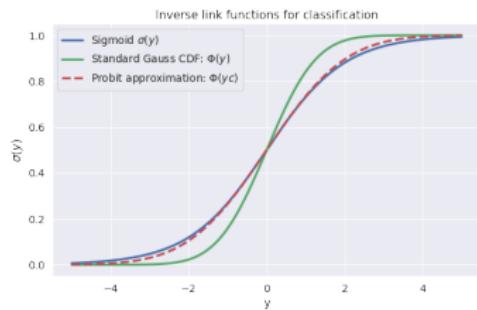
1. Monte Carlo methods (sampling)

$$p(y^* = 1|\mathbf{y}, \mathbf{x}^*) \approx \frac{1}{S} \sum_{i=1}^S \sigma\left(f^{(i)}\right) \quad \text{for} \quad f^{(i)} \sim \mathcal{N}(f|\mu_{f^*}, \sigma_{f^*}^2)$$

2. Probit approximation

$$\sigma(f) \approx \Phi\left(f \sqrt{\frac{\pi}{8}}\right) \Rightarrow p(y^* = 1|\mathbf{y}, \mathbf{x}^*) \approx \Phi\left(\frac{\mu_{f^*}}{\sqrt{\frac{8}{\pi} + \sigma_{f^*}^2}}\right)$$

where Φ is the CDF of the standard normal



Gaussian processes for classification VI: Putting everything together

- Step 1: Construct Laplace approximation

$$p(\mathbf{f}|\mathbf{y}) \approx q(\mathbf{f}) = \mathcal{N}(\mathbf{f}|\mathbf{m}, \mathbf{S})$$

- Step 2: Posterior distribution for latent function f^*

$$p(f^*|\mathbf{y}, \mathbf{x}^*) = \mathcal{N}(f^* | \mu_{f^*}, \sigma_{f^*}^2)$$

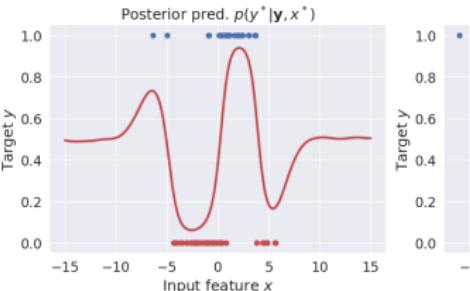
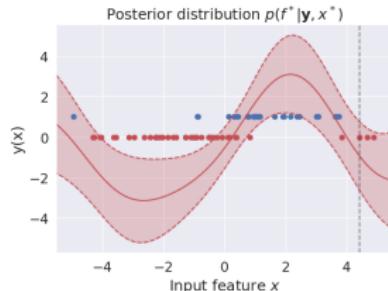
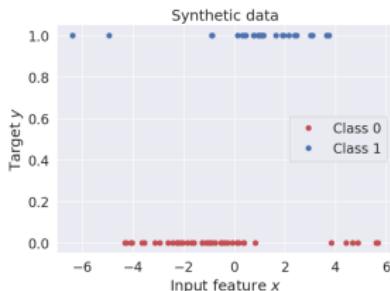
where

$$\mu_{f^*} = \mathbf{k}^T \mathbf{K}^{-1} \mathbf{m}$$

$$\sigma_{f^*}^2 = k - \mathbf{k}^T \mathbf{K}^{-1} (\mathbf{K} - \mathbf{S}) \mathbf{K}^{-1} \mathbf{k}$$

- Step 3: The predictive distribution for classification labels

$$p(y^* = 1|\mathbf{y}, \mathbf{x}^*) \approx \int p(y^* = 1|f^*) p(f^*|\mathbf{y}, \mathbf{x}^*) df^* = \int \sigma(f^*) p(f^*|\mathbf{y}, \mathbf{x}^*) df^*$$



A bit about neural networks for probabilistic modelling

Neural Networks

- Sequence of linear (affine) and non-linear mappings
- Two-layer neural network (NN) with single output

$$z_1 = h_1(\mathbf{W}_1 \mathbf{x} + \mathbf{b}_1)$$

$$z_2 = h_2(\mathbf{W}_2 z_1 + \mathbf{b}_2)$$

$$f = \mathbf{W}_3 z_2 + \mathbf{b}_3$$

- From input to output (bias terms left out)

$$f(\mathbf{x}) = \mathbf{W}_3 h_2(\mathbf{W}_2 h_1(\mathbf{W}_1 \mathbf{x}))$$

- Our linear model with basis functions from week 2

$$f(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x})$$

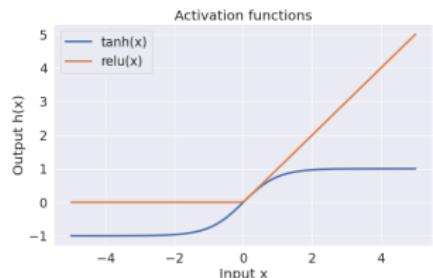
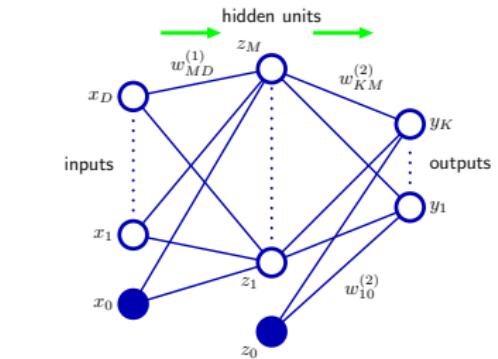
- Linear model with adaptive basis functions

$$z_2(\mathbf{x}) = h_2(\mathbf{W}_2 h_1(\mathbf{W}_1 \mathbf{x})) = \Phi_{\mathbf{W}}(\mathbf{x})$$

$$f(\mathbf{x}) = \mathbf{W}_3 z_2 = \mathbf{W}_3 \Phi_{\mathbf{W}}(\mathbf{x})$$

- NNs in probabilistic modelling

$$p(y_n | \mathbf{w}) = \mathcal{N}(y_n | f(\mathbf{x} | \mathbf{w}), \sigma^2)$$

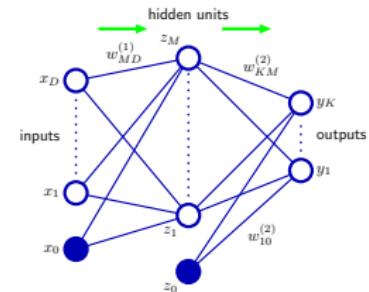
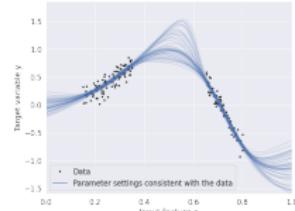


Bayesian Neural Networks

- Bayesian deep learning is a very active research area

- Why Bayesian neural networks?

1. Ensemble methods often work well
2. Uncertainty quantification
3. Incorporating prior knowledge
4. Less prone to overfitting
5. Data efficiency
6. Side step pathologies with maximum likelihood learning
7. Active learning



- Posterior geometry of NNs are complicated!

1. NNs are highly non-linear
2. NNs have weight-space symmetries
3. Often underdetermined by the data

- Bayesian inference in neural networks is generally a really *difficult problem*

$$\begin{aligned}z_1 &= h_1(\mathbf{w}_1 \mathbf{x}) + \mathbf{b}_1 \\z_2 &= h_2(\mathbf{w}_2 z_1) + \mathbf{b}_2 \\f &= \mathbf{w}_3 z_2 + \mathbf{b}_3\end{aligned}$$

MAP estimators for probabilistic neural networks

- NNs for binary classification

$$p(\mathbf{y}|\mathbf{f}) = \prod_{n=1}^N \sigma(f_n)^{y_n} (1 - \sigma(f_n))^{1-y_n}$$
$$f_n = \text{NN}(\mathbf{x}_n | \mathbf{w})$$

- We can impose Gaussian priors on all the weights (\mathbf{w} is vector containing all weights of the NN)

$$\mathbf{w} \sim \mathcal{N}(0, \alpha^{-1})$$

- The MAP (sometimes called to as *poor man's Bayes*) estimator is

$$\hat{\mathbf{w}}_{\text{MAP}} = \arg \max_{\mathbf{w}} \frac{p(\mathbf{y}|\mathbf{W})p(\mathbf{W})}{p(\mathbf{y})} = \arg \max_{\mathbf{W}} p(\mathbf{y}|\mathbf{W})p(\mathbf{W})$$

- We have

$$\ln p(\mathbf{y}|\mathbf{W})p(\mathbf{W}) = \sum_{n=1}^N \ln p(y_n|f(\mathbf{x}_n | \mathbf{w})) - \frac{\alpha}{2} \mathbf{w}^T \mathbf{w} + \text{const}$$

- For MAP estimation, Gaussian priors acts as ℓ_2 -regularization

- Pros: easy and fast to implement, easy build complex models, Cons: no uncertainty and prone to overfitting

Questions: True or False

Spend 5 minutes on the DTU Learn quiz: "Lecture 6: Probabilistic neural networks."

Given a model with likelihood $p(\mathbf{y}|\mathbf{W})$ and suppose we impose a flat prior on \mathbf{w} , i.e. $p(\mathbf{w}) \propto 1$, then ...

1. the maximum a posterior (MAP) solution is the same as the posterior mean. *True or false?*
2. ... the maximum likelihood solution and MAP (posterior mode) is the same. *True or false?*
3. ... the predictive distribution for MAP is the same as that for Bayesian inference? *True or False?*

For models with Gaussian priors

- 4 ... increasing α will cause MAP estimate of \mathbf{w} to numerically larger. *True or False?*
- 5 ... increasing α will increase strength of regularization. *True or False?*

02477 – Bayesian Machine Learning: Lecture 7

Michael Riis Andersen

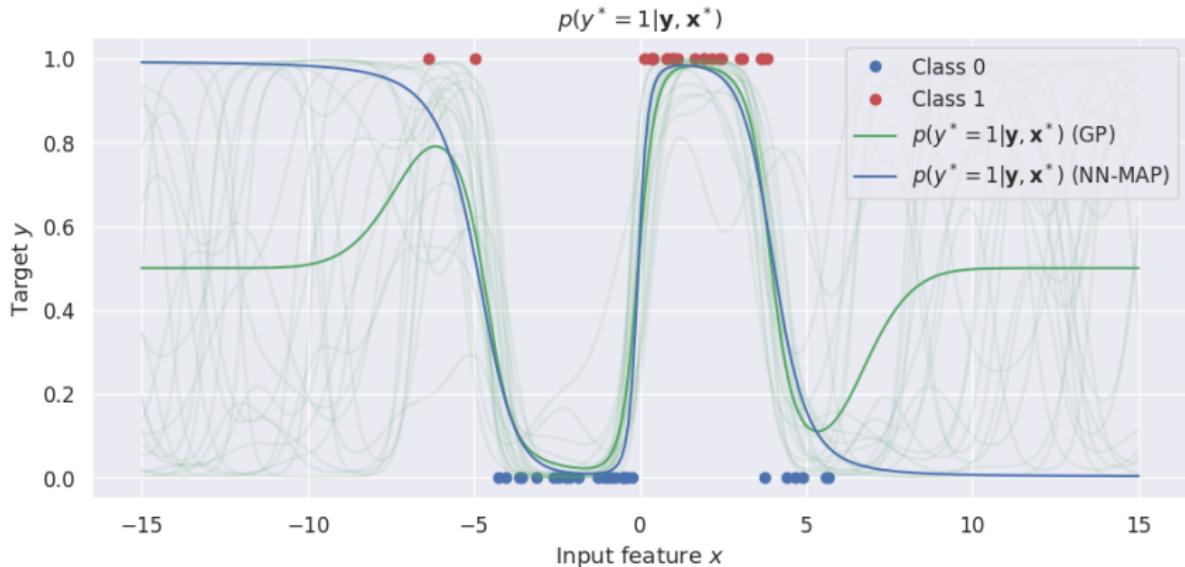
Technical University of Denmark,
DTU Compute, Department of Applied Math and Computer Science

Outline

- 1 Gaussian processes and neural networks
- 2 Generalized linear models and non-Gaussian likelihoods
- 3 Generalization and evaluation
- 4 Decision theory
- 5 Calibration

Gaussian processes and neural networks

From last week's exercise



- Some NN researchers are striving to make NNs behave more like Gaussian processes
- Some GP researchers are striving to make GPs flexible and as easy to scale as NNs
- Exploring relationships between GPs and NNs

Infinitely wide neural networks I

Exploring connections between GPs and NNs

- Consider a network with a single hidden layer with H neurons and one output

$$z(x) = h(\mathbf{W}_1 \mathbf{x} + \mathbf{b}_1)$$
$$f(x) = \mathbf{W}_2 z(x) + \mathbf{b}_2$$

- For $\mathbf{x} \in \mathbb{R}^D$, then $\mathbf{W}_1 \in \mathbb{R}^{H \times D}$ and $\mathbf{W}_2 \in \mathbb{R}^{H \times 1}$

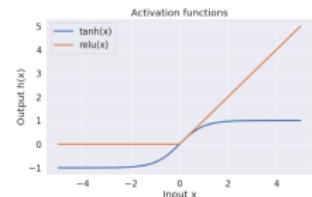
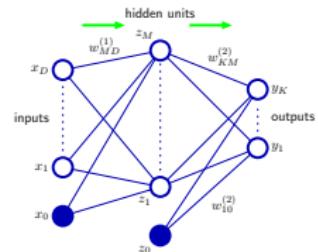
- Universal approximators* for many classes of functions

- Let's now make some assumptions

- Activation function h is bounded, e.g. $h(x) = \tanh(x)$
- \mathbf{W}_1 and \mathbf{b}_1 have i.i.d. zero-mean Gaussian priors
- \mathbf{W}_2 and \mathbf{b}_2 have zero-mean and prior variances σ_w^2 and σ_b^2
- Prior variance of $\sigma_w^2 = \frac{1}{H}$

- What is the mean and variance of $f(\mathbf{x})$?

$$\mathbb{E}[f(\mathbf{x})] = \mathbb{E}[\mathbf{W}_2 z(\mathbf{x}) + \mathbf{b}_2] = \sum_{j=1}^H \mathbb{E}[w_j] \mathbb{E}[h_j(\mathbf{x})] + \mathbb{E}[\mathbf{b}_2] = 0$$



Infinitely wide neural networks II

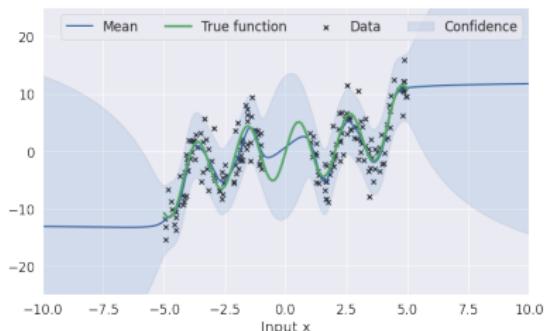
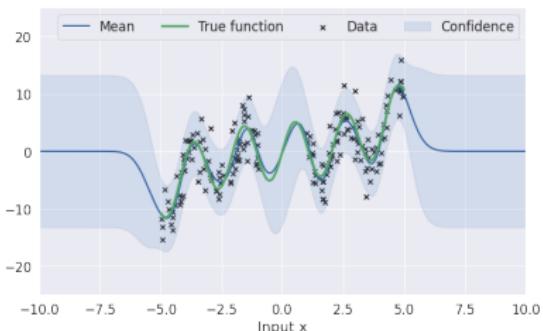
- The neural network is a zero-mean stochastic process, i.e.

$$\mathbb{E}[f(\mathbf{x})] = 0$$

- Let number of neurons H go to infinity, i.e. $H \rightarrow \infty$
- CLT implies the neural network $f(\mathbf{x})$ converges to a Gaussian process

$$k_{\text{NN}}(\mathbf{x}, \mathbf{x}') \equiv \frac{2}{\pi} \sin^{-1} \frac{2\tilde{\mathbf{x}}^T \tilde{\mathbf{x}}'}{\sqrt{(1 + 2\tilde{\mathbf{x}}^T \tilde{\mathbf{x}})(1 + 2\tilde{\mathbf{x}}'^T \tilde{\mathbf{x}}')}} \quad \text{for} \quad \tilde{\mathbf{x}} = [1, \mathbf{x}]^T$$

- Comparing *squared exponential* and *neural network kernels*



Generalized linear models and non-Gaussian likelihoods

Likelihoods as observation models: Why bother?

- Consider a linear model

$$f(\mathbf{x}) = \phi(\mathbf{x})^T \mathbf{w}$$

- Regression using Gaussian likelihood

$$y_n | f_n \sim \mathcal{N}(y_n | f_n, \beta^{-1})$$

- Binary classification with sigmoid function $\sigma(\cdot)$

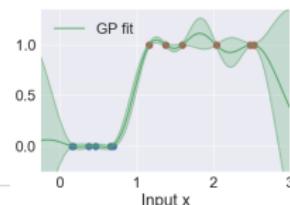
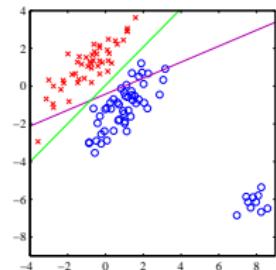
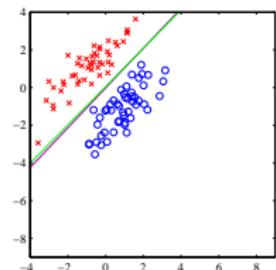
$$y_n | f_n \sim \text{Ber}(y_n | \sigma(f_n))$$

- ..., but why not just use regression for everything?

- More general setting: *generalized linear models* GLMs

$$y_n | f_n \sim p(y_n | f_n)$$

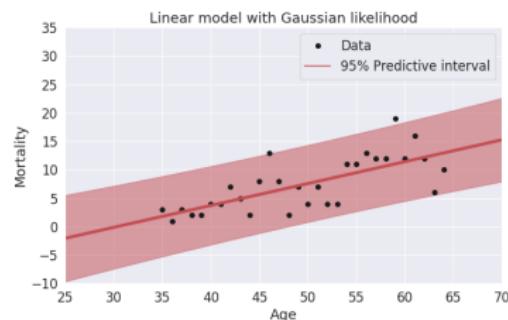
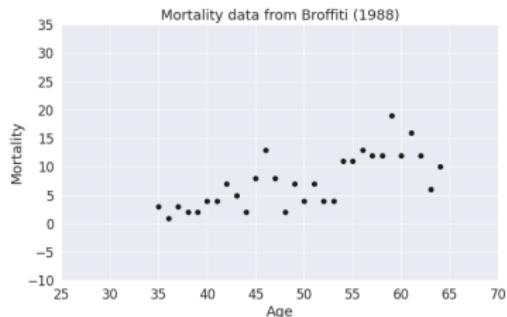
Least squared vs logistic regression



Common likelihoods in machine learning

Likelihood	Support	Example
Gaussian	$y_n \in \mathbb{R}$	Standard regression
Student's t	$y_n \in \mathbb{R}$	Standard regression with heavier tails
Exponential	$y_n \in (0, \infty)$	Strictly positive target values
Gamma	$y_n \in (0, \infty)$	Strictly positive target values
Bernoulli	$y_n \in \{0, 1\}$	Binary classification
Binomial	$y_n \in \{0, 1, \dots, N\}$	Sequence of Bernoulli trials
Poisson	$y_n \in \{0, 1, 2, 3, \dots, \}$	Count data
Categorial	$y_n \in \{0, 1, 2, \dots, K\}$	Multi-class classification (requires K latent functions)

Example: analyzing mortality as a function of age



Under this model, the posterior predictive distribution for age = 25 is $\mathcal{N}(-2.61, 14.76)$ meaning that $p(\text{mortality} < 0 | \text{age} = 25) \approx 0.75$

Common likelihoods in machine learning

Likelihood	Support	Example
Gaussian	$y_n \in \mathbb{R}$	Standard regression
Student's t	$y_n \in \mathbb{R}$	Standard regression with heavier tails
Exponential	$y_n \in (0, \infty)$	Strictly positive target values
Gamma	$y_n \in (0, \infty)$	Strictly positive target values
Bernoulli	$y_n \in \{0, 1\}$	Binary classification
Binomial	$y_n \in \{0, 1, \dots, N\}$	Sequence of Bernoulli trials
Poisson	$y_n \in \{0, 1, 2, 3, \dots, \}$	Count data
Categorial	$y_n \in \{0, 1, 2, \dots, K\}$	Multi-class classification (requires K latent functions)

Spend 5 minutes on the DTU Learn quiz: "Lecture 7a: Common likelihoods in machine learning."

Discuss what type of observation model might be appropriate for ...

1. Predicting whether a student will pass a course (based on the student's previous grades)
2. Predicting the time it takes to finish a written exam
3. Predicting number of errors in a multiple choice exam
4. Predicting whether the student will fill out the exam with a blue, red or black pen

Generalized linear models in three steps

The components of a generalized linear model (GLM)

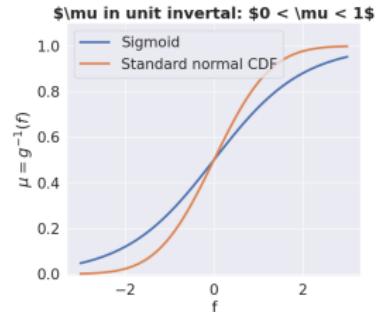
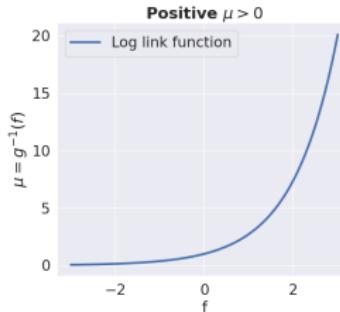
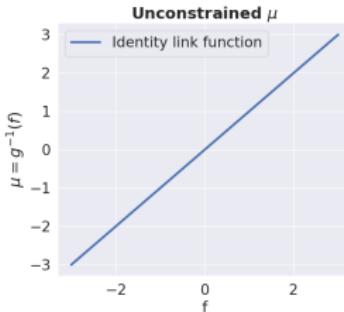
1. The linear model

$$f(\mathbf{x}) = \phi(\mathbf{x})^T \mathbf{w}$$

2. The *link function* g that relates the mean of the linear model to the mean of the response variable $\mathbb{E}[y(\mathbf{x})|\mathbf{x}] = \mu(\mathbf{x})$

$$g(\mu(\mathbf{x})) = f(\mathbf{x}) \iff \mathbb{E}[y|\mathbf{x}] = \mu(\mathbf{x}) = g^{-1}[f(\mathbf{x})]$$

3. The distribution $p(y_n|\mathbf{x}_n)$ for the response variable y_n , e.g. Poisson, binomial, gamma etc.



Example: Bayesian Poisson Regression I

- Step 1: We assume a linear model with $\mathbf{x} = [1 \quad \text{age}]^T$

$$f(\text{age}) = w_0 + w_1 \text{age} \iff f(\mathbf{x}) = \mathbf{x}^T \mathbf{w}$$

- Step 2: Since $\mu > 0$, we use the log link function $\log(\mu) = f$ or equivalently

$$\mu(\mathbf{x}) = \exp(f(\mathbf{x})) = \exp(\mathbf{x}^T \mathbf{w})$$

- Step 3: We use a Poisson likelihood for count data

$$\text{Poisson}(y_n = k | \mu) = \frac{\mu^k \exp(-\mu)}{k!}$$

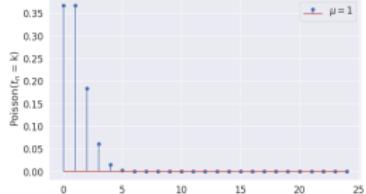
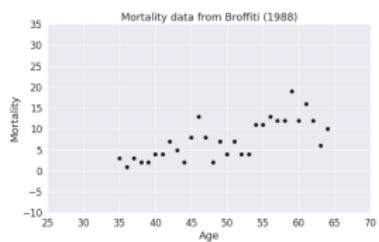
where $\mu > 0$ is the mean parameter.

- Thus, the likelihood becomes

$$y_n | \mathbf{x}_n, \mathbf{w} \sim \text{Poisson}(\mu_n)$$

- We use a Gaussian prior for \mathbf{w}

$$p(\mathbf{w}) = \mathcal{N}(\mathbf{w} | \mathbf{0}, \mathbf{I})$$



Example: Bayesian Poisson Regression II

- Using $\mu_n = \mu(\mathbf{x}) = \exp(\mathbf{x}_n^T \mathbf{w})$, the joint model becomes

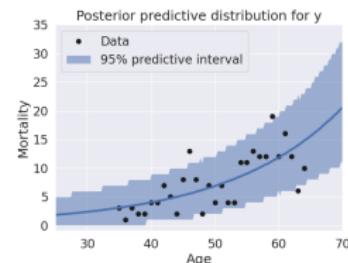
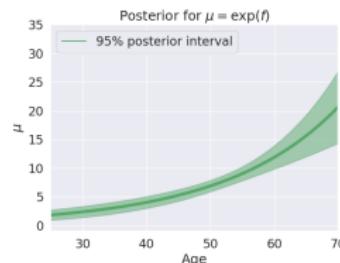
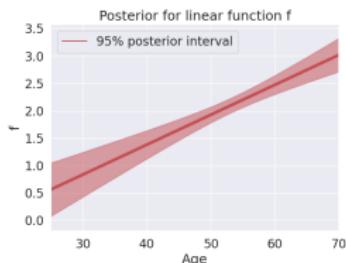
$$p(\mathbf{y}, \mathbf{w}) = p(\mathbf{y}|\mathbf{w})p(\mathbf{w}) = \prod_{n=1}^N \text{Poisson}(y_n|\mu_n)\mathcal{N}(\mathbf{w}|\mathbf{0}, \mathbf{I})$$

- The posterior is again intractable, so we use a Laplace approximation again

$$p(\mathbf{w}|\mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{w})p(\mathbf{w})}{p(\mathbf{y})} \approx q(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mathbf{m}, \mathbf{S})$$

- The (approximate) posterior predictive distribution for a new point \mathbf{x}_* with $\mu_* = g^{-1}(\mathbf{x}_*^T \mathbf{w})$

$$p(y_* = k|\mathbf{y}) \approx \int p(y_* = k|\mathbf{w})q(\mathbf{w})d\mathbf{w} = \int \text{Poisson}(y_* = k|\mu_*)\mathcal{N}(\mathbf{w}|\mathbf{m}, \mathbf{S})d\mathbf{w}$$



Example: Bayesian Poisson Regression III

- The (approximate) posterior predictive distribution for a new point \mathbf{x}_* with $\mu_* = g^{-1}(\mathbf{x}_*^T \mathbf{w})$

$$p(y_* = k | \mathbf{y}) \approx \int \text{Poisson}(y_* = k | \mu_*) \mathcal{N}(\mathbf{w} | \mathbf{m}, \mathbf{S}) d\mathbf{w}$$

- Calculating predictions for age = 30. Let $\mathbf{x}_* = [1 \ 30]$, then $f_* = \mathbf{x}_*^T \mathbf{w}$

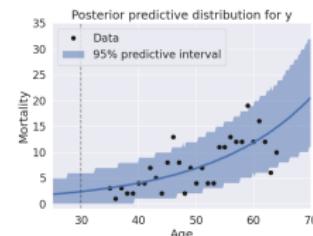
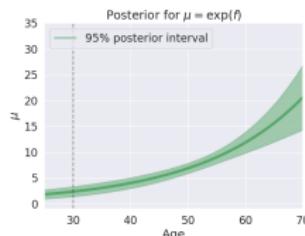
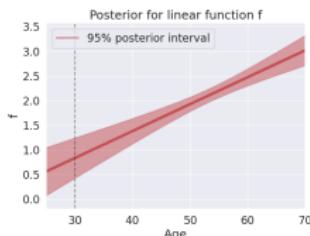
$$p(f_* | \mathbf{y}) = \mathcal{N}(f_* | \mathbf{x}_*^T \mathbf{m}, \mathbf{x}_*^T \mathbf{S} \mathbf{x}_*) \approx \mathcal{N}(f_* | 0.8, 0.2^2)$$

- We can calculate the distributions of $\mu_* | \mathbf{y}$ and $y_* | \mathbf{y}$ using samples $f_*^{(s)} \sim \mathcal{N}(f_* | 0.8, 0.2^2)$ for $s = 1, \dots, S$. For each sample $f_*^{(s)}, \dots$

1. Compute $\mu_*^{(s)} = \exp(f_*^{(s)})$
2. Sample $y_*^{(s)} | \mu_*^{(s)} \sim \text{Poisson}(\mu_*^{(s)})$

- Finally, we calculate the sample means (or variances, percentiles etc)

$$\mathbb{E} [\mu_*^{(s)} | \mathbf{y}] \approx \frac{1}{S} \sum_{s=1}^S \mu_*^{(s)} = 2.34 \quad \mathbb{E} [y_*^{(s)} | \mathbf{y}] \approx \frac{1}{S} \sum_{s=1}^S y_*^{(s)} = 2.34$$



Generalized GP/NN models in three steps

The components of a generalized model

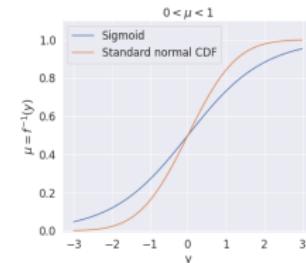
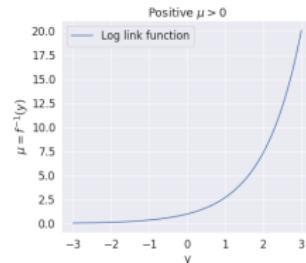
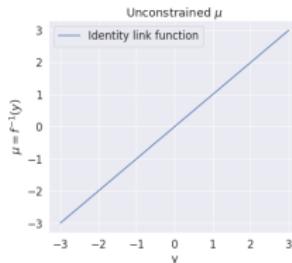
1. We replace the linear model with a Gaussian process (or a NN)

$$f(\mathbf{x}) \sim \mathcal{GP}(0, k(\mathbf{x}, \mathbf{x}'))$$

2. The *link function* f that related the mean of the linear model to the mean of the response variable $t(\mathbf{x})$

$$g[\mu(\mathbf{x})] = f(\mathbf{x}) \quad \iff \quad \mathbb{E}[y|\mathbf{x}] = \mu(\mathbf{x}) = g^{-1}[f(\mathbf{x})]$$

3. The distribution $p(y|\mathbf{x})$ for the response variable $y(\mathbf{x})$, e.g. Poisson, binomial, gamma etc.



Adapting GP models to different likelihoods

- From last weeks' exercise: Laplace approximation GP for classification

$$p(\mathbf{f}|\mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{f})p(\mathbf{f})}{p(\mathbf{y})} \approx q(\mathbf{f}) = \mathcal{N}(\mathbf{f}|\mathbf{m}, \mathbf{S}),$$

- The log joint of the target \mathbf{y} and latent function values \mathbf{f}

$$\log p(\mathbf{y}, \mathbf{f}) = \log p(\mathbf{y}|\mathbf{f}) + \log p(\mathbf{f}) = \sum_{n=1}^N \log p(y_n|f_n) - \frac{N}{2} \log(2\pi) - \frac{1}{2} |\mathbf{K}| - \frac{1}{2} \mathbf{f}^T \mathbf{K}^{-1} \mathbf{f}$$

- The gradient and Hessian of the log joint

$$\nabla_{\mathbf{f}} \log p(\mathbf{y}, \mathbf{f}) = \sum_{n=1}^N \nabla_{\mathbf{f}} \log p(y_n|f_n) - \mathbf{K}^{-1} \mathbf{f},$$

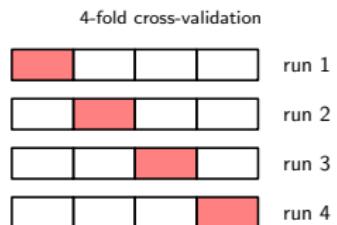
$$\nabla_{\mathbf{f}}^2 \log p(\mathbf{y}, \mathbf{f}) = \sum_{n=1}^N \nabla_{\mathbf{f}}^2 \log p(y_n|f_n) - \mathbf{K}^{-1}$$

- Hence, all we need to change is the first and second order derivative of log likelihood

Generalization and evaluation

Generalization error and model evaluation

- Consider a supervised problem with $\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=1}^N$
- We measure performance of models using *cross-validation*
 1. Held-out test set
 2. K-fold
 3. Leave-one-out
 4. Split-half
 5. ...
- Goal: assess the *generalization error* of the model, i.e. how well can we expect the model to perform on a new, unseen test example?



- We use cross-validation to *estimate* the generalization error
- Parameter tuning: training/validation/test or nested cross-validation

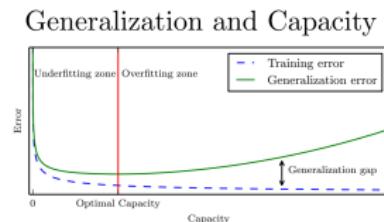


Figure from Goodfellow 2016

Generalization error and loss functions

- Let y be the target value for a input vector \mathbf{x} and let $\hat{y} \equiv \hat{y}(\mathbf{x})$ be a prediction
- The *loss function* $\mathcal{L}(y, \hat{y})$ define the cost of predicting \hat{y} when the true value is y

- The *quadratic loss* for regression is

$$\mathcal{L}(y, \hat{y}) = (y - \hat{y})^2$$

- The *0/1 loss* for classification is given by

$$\mathcal{L}(y, \hat{y}) = \mathbb{I}[y \neq \hat{y}]$$

- ...

- The *generalization error* (or *expected loss*, *risk*, *out-of-sample error*) for a model $\hat{y}(\mathbf{x})$ is defined as

$$\mathcal{R}_{\hat{y}} \equiv \mathbb{E} [\mathcal{L}(y, \hat{y}(\mathbf{x}))] = \iint \mathcal{L}(y, \hat{y}(\mathbf{x})) p(\mathbf{x}, y) d\mathbf{x} dy$$

- We rarely know the *true data generating mechanism* $p(\mathbf{x}, y)$ in practice

Estimating the generalization

- The *exact* generalization error

$$\mathcal{R}_{\hat{y}} \equiv \mathbb{E} [\mathcal{L}(y, \hat{y}(\mathbf{x}))] = \iint \mathcal{L}(y, \hat{y}(\mathbf{x})) p(\mathbf{x}, y) d\mathbf{x} dy$$

- For *independent and identically distributed (i.i.d)* samples $(\mathbf{x}_i^*, y_i^*) \sim p(\mathbf{x}, y)$ for $i = 1, \dots, N_{\text{test}}$

$$\mathcal{R}_{\hat{y}} \approx \widehat{\mathcal{R}}_{\hat{y}}^{\text{test}} = \frac{1}{N_{\text{test}}} \sum_{i=1}^{N_{\text{test}}} \mathcal{L}(y_i^*, \hat{y}(\mathbf{x}_i^*))$$

- The *estimator* is accurate when the test set is large: $\widehat{\mathcal{R}}_{\hat{y}}^{\text{test}} \rightarrow \mathcal{R}_{\hat{y}}$ for $N_{\text{test}} \rightarrow \infty$

- The *empirical risk* for the training set

$$\mathcal{R}_{\hat{y}} \approx \widehat{\mathcal{R}}_{\hat{y}}^{\text{train}} = \frac{1}{N_{\text{train}}} \sum_{i=1}^{N_{\text{train}}} \mathcal{L}(y_i, \hat{y}(\mathbf{x}_i))$$

- Many learning algorithms can be expressed as *empirical risk minimization* (ERM)

$$\hat{\mathbf{w}} = \arg \min_{\mathbf{w}} \widehat{\mathcal{R}}_{\hat{y}}^{\text{train}}$$

- For ERM, one can show that the training error is *optimistic*

$$\mathbb{E} [\widehat{\mathcal{R}}_{\hat{y}}^{\text{train}}] \leq \mathbb{E} [\widehat{\mathcal{R}}_{\hat{y}}^{\text{test}}]$$

Example: the expected generalization error wrt. to the squared loss

- Consider the following (very simple) linear regression model (no slope, only intercept)

$$y = w_{\text{true}} + e, \quad e \sim \mathcal{N}(0, \sigma^2)$$

- We will use some estimator \hat{w} for w_{true} (fixed) as a prediction for a new y , i.e. $y^* = \hat{w}$.
- What is the *generalization error* wrt. *squared loss*?

$$\begin{aligned}\mathcal{R}_{\hat{w}} &= \mathbb{E}[\mathcal{L}] = \iint (t - \hat{w})^2 p(y, x) dy dx \\ &= \int (y - \hat{w})^2 p(y) dy \\ &= \int (y - \hat{w})^2 \mathcal{N}(y | w_{\text{true}}, \sigma^2) dy \\ &= \int (y^2 + \hat{w}^2 - 2y\hat{w}) \mathcal{N}(y | w_{\text{true}}, \sigma^2) dy \\ &= \int y^2 \mathcal{N}(y | w_{\text{true}}, \sigma^2) dy + \hat{w}^2 \int \mathcal{N}(y | w_{\text{true}}, \sigma^2) dy - 2\hat{w} \int y \mathcal{N}(y | w_{\text{true}}, \sigma^2) dy \\ &= (w_{\text{true}} - \hat{w})^2 + \sigma^2\end{aligned}$$

- Unsurprisingly, the generalization error is minimized when $\hat{w} = w_{\text{true}}$, which leads to $\mathbb{E}[\mathcal{L}] = \sigma^2$.

What is the expected generalization error for a given dataset size?

- The generalization error

$$\mathcal{R}_{\hat{w}} = (w - \hat{w})^2 + \sigma^2$$

- Suppose we observe a dataset $\mathcal{D} = \{y_n\}_{n=1}^N$ with the goal of estimating the parameter w_{true}

$$y_n = w_{\text{true}} + e_n, \quad e_n \sim \mathcal{N}(0, \sigma^2)$$

- The maximum likelihood (or ERM) estimator for w_{true} and its sampling distribution is

$$w_{\text{MLE}} = \frac{1}{N} \sum_{n=1}^N y_n, \quad w_{\text{MLE}} \sim \mathcal{N}\left(w_{\text{true}}, \frac{1}{N} \sigma^2\right)$$

- The *expected generalization error* for the maximum likelihood estimator for a dataset of size N is

$$\langle \mathcal{R}_{w_{\text{MLE}}} \rangle_N = \int \left[(w_{\text{true}} - \hat{w}_{\text{ML}})^2 + \sigma^2 \right] \mathcal{N}\left(w_{\text{ML}} | w_{\text{true}}, \frac{\sigma^2}{N}\right) dw_{\text{ML}} = \frac{\sigma^2}{N} + \sigma^2$$

- Interpretation

1. Average excess error due to finite training data
2. Inherent noise in the data

Decision theory

Uncertainty in multi-class classification

- Categorical distributions for multi-class classification with K classes

$$y_n | \mathbf{f}_n \sim \text{Categorical} [\text{softmax}(\mathbf{f}_n)]$$

- The posterior predictive distribution will be another categorial distribution

$$p(y^* = k | \mathbf{y}, \mathbf{x}^*) = \pi_k \quad \text{for} \quad \sum_{k=1}^K \pi_k = 1 \quad \text{and} \quad 0 \leq \pi_k \leq 1$$

- Predict using most likely class

$$\hat{y}^* = \arg \max_k p(y^* = k | \mathbf{y}, \mathbf{x}^*)$$

- Example

$p(y^* = \text{cat} \text{img})$	$= 0.36$	$p(y^* = \text{healthy} \text{x-ray})$	$= 0.36$
$p(y^* = \text{dog} \text{img})$	$= 0.34$	$p(y^* = \text{pre-phase cancer} \text{x-ray})$	$= 0.34$
$p(y^* = \text{bird} \text{img})$	$= 0.30$	$p(y^* = \text{severe cancer} \text{x-ray})$	$= 0.30$

Quantifying uncertainty for multi-class classification

- The *confidence* of the posterior predictive distribution is defined as

$$\mathcal{C} = \max_k p(y^* = k | \mathbf{y}, \mathbf{x}^*) \quad (\text{range: } [\frac{1}{K}, 1])$$

- The *entropy* is defined as (higher \Rightarrow more uncertainty)

$$\mathcal{H} = - \sum_{k=1}^K \pi_k \log \pi_k \quad (\text{range: } [0, \log K])$$

- Convention for entropy calculations: $0 \log 0 = 0$

- Examples

$$p(y^* = \text{cat}|\text{img}) = 0.50 \quad p(y^* = \text{healthy}|\text{x-ray}) = 0.50$$

$$p(y^* = \text{dog}|\text{img}) = 0.50 \quad p(y^* = \text{pre-phase cancer}|\text{x-ray}) = 0.25$$

$$p(y^* = \text{bird}|\text{img}) = 0.0 \quad p(y^* = \text{severe cancer}|\text{x-ray}) = 0.25$$

$$\mathcal{C} = 0.5$$

$$\mathcal{C} = 0.5$$

$$\mathcal{H} \approx 0.69$$

$$\mathcal{H} \approx 1.04$$

Reject option: I don't know?

- For applications in medicine etc. it may be better to say "I don't know" rather than provide a prediction we don't really trust

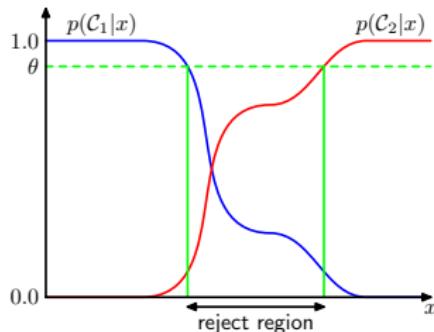
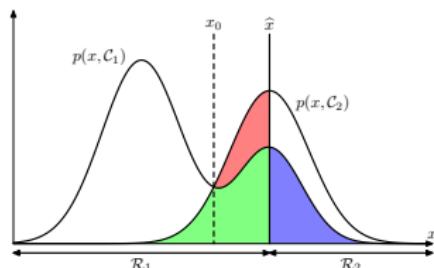
- Reject option:** Avoid making a decision if the uncertainty is too large

- Reject to make decision if

$$\mathcal{C} = \max_k p(y^* = i | \mathbf{y}, \mathbf{x}^*) < \theta_{\text{reject}}$$

- If $\theta_{\text{reject}} = 1$, then all samples will get rejected

- If $\theta_{\text{reject}} \leq \frac{1}{K}$, then no samples will get rejected



Decision theory: a more formal approach

- In Bayesian modelling we strive to represent all unknown quantities (parameters, predictions etc.) using *probability distributions*
- Yet, we are often forced to reduce these to single decisions/predictions
 1. Cancer or not cancer?
 2. Which online campaign is better? A, B or C etc?
 3. How many airline passengers in year 2025?
 4. Which model best describes the data?
 5. ...
- *Statistical decision theory* tells us to make optimal decisions under uncertainty

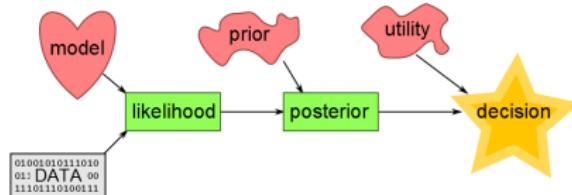


Figure from <https://www.azimuthproject.org>

Bayesian decision theory in a nutshell

- A decision maker has a set of actions/choices $a \in \mathcal{A}$ to choose from
- The optimal action depends on the true state $s \in S$ of the system of interest
- The *loss function* $\mathcal{L}(s, a)$ determines the cost for choice a when the true state is s . We can also use a *utility function* $\mathcal{U}(s, a) = -\mathcal{L}(s, a)$
- Example: assigning medical treatment to potential Covid19 patient

$\mathcal{L}(s, a)$	Do nothing	Isolate at home	Hospitalization
Healthy	0	10	50
Covid19, mild symptoms	10	0	20
Covid19, severe symptoms	100	10	0

- In practice, we do not know the true state s . What to do?
- *Bayesian decision theory*
 1. Compute posterior of the state s given data y , i.e. $p(s|y)$
 2. Choose the action that minimizes the posterior expected loss

$$\hat{a} = \arg \min_{a \in \mathcal{A}} \mathbb{E}_{p(s|y)} [\mathcal{L}(s, a)]$$

5 minutes exercise

$\mathcal{L}(s, a)$	Do nothing	Isolate at home	Hospitalization
Healthy	0	10	50
Covid19, mild symptoms	10	0	20
Covid19, severe symptoms	100	10	0

- Suppose a medical doctor collected data about a patient and arrived at the following posterior predictions for whether the patient will get Covid19 or not

State s	Healthy	Mild symptoms	Severe symptoms
$p(s y)$	0.65	0.3	0.05

Questions

- What is the posterior expected loss for each action, i.e. $\mathbb{E}_{p(s|y)} [\mathcal{L}(s, a)]$?
- What is the optimal action? ($\hat{a} = \arg \min_{a \in \mathcal{A}} \mathbb{E}_{p(s|y)} [\mathcal{L}(s, a)]$)

Decision theory for classification I

- Consider a binary classification problem with $\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=1}^N$ for $y_i \in \{0, 1\}$
- The *0/1 utility function* is given by $\mathcal{U}(y, \hat{y}(\mathbf{x})) = \mathbb{I}[y = \hat{y}(\mathbf{x})]$

$\mathcal{U}(y, \hat{y}(\mathbf{x}))$	$\hat{y} = 0$	$\hat{y} = 1$
$y = 0$	1	0
$y = 1$	0	1

- If $p \equiv p(y^* = 1 | \mathbf{y}, \mathbf{x}^*)$ denotes the posterior class probability for input point \mathbf{x}^* , then

$$\mathbb{E}[\mathcal{U}(y^*, \hat{y}(\mathbf{x}))] = \sum_{y^*} p(y^* | \mathbf{y}, \mathbf{x}^*) \mathcal{U}(y^*, \hat{y}(\mathbf{x})) = (1 - p) \mathbb{I}[0 = \hat{y}(\mathbf{x})] + p \mathbb{I}[1 = \hat{y}(\mathbf{x})]$$

- If we choose $\hat{y}(\mathbf{x}) = 0$, then

$$\mathbb{E}[\mathcal{U}(y^*, 0)] = (1 - p) \mathbb{I}[0 = 0] + p \mathbb{I}[1 = 0] = 1 - p$$

- ... and if we choose $\hat{y}(\mathbf{x}) = 1$, then

$$\mathbb{E}[\mathcal{U}(y^*, 1)] = (1 - p) \mathbb{I}[0 = 1] + p \mathbb{I}[1 = 1] = p$$

- Picking the class with largest posterior pred. probability is *Bayes optimal* under the 0/1-loss function

Decision theory for classification II

- Consider a binary classification problem with $\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=1}^N$ for $y_i \in \{0, 1\}$
- Example: classifying cancer from medical imagery \mathbf{x}

$\mathcal{U}(y, \hat{y}(\mathbf{x}))$	$\hat{y} = 0$ (not cancer)	$\hat{y} = 1$ (cancer)
$y = 0$ (not cancer)	1	-10
$y = 1$ (cancer)	-100	1

- How certain do we need to be before we "dare" to predict $\hat{y}^* = 0$?

$$\begin{aligned}\mathbb{E} [\mathcal{U}(y^*, 0)] &\geq \mathbb{E} [\mathcal{U}(y^*, 1)] \\ \Rightarrow (1-p)\mathcal{U}_{00} + p\mathcal{U}_{10} &\geq (1-p)\mathcal{U}_{01} + p\mathcal{U}_{11} \\ \Rightarrow p &\leq \frac{\mathcal{U}_{01} - \mathcal{U}_{00}}{\mathcal{U}_{10} - \mathcal{U}_{11} + \mathcal{U}_{01} - \mathcal{U}_{00}}\end{aligned}$$

- We have

$$p \leq \frac{\mathcal{U}_{01} - \mathcal{U}_{00}}{\mathcal{U}_{10} - \mathcal{U}_{11} + \mathcal{U}_{01} - \mathcal{U}_{00}} = \frac{-10 - 1}{-100 - 1 - 10 - 1} \approx 0.099$$

- That is, if $p(y^* = 0 | \mathbf{x}) = 1 - p > 0.901$, we predict "No cancer"

Decision theory for regression I

- Consider a regression problem with $\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=1}^N$ for $y_i \in \mathbb{R}$
- The most common loss function for regression is the *quadratic loss*

$$\mathcal{L}(y, y(\mathbf{x})) = (y - \hat{y}(\mathbf{x}))^2$$

- The expected loss for a complete probabilistic description $p(\mathbf{x}, y)$ is

$$\mathbb{E} [\mathcal{L}(y, y(\mathbf{x}))] = \iint (y - \hat{y}(\mathbf{x}))^2 p(\mathbf{x}, y) d\mathbf{x} dy$$

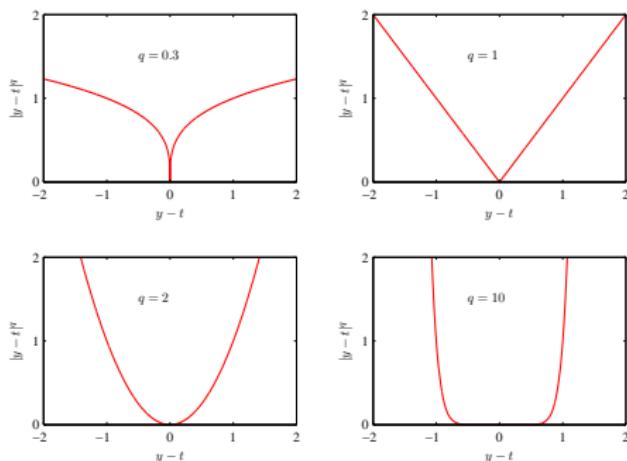
- What is the *optimal predictor function* $\hat{y}(\mathbf{x})$?

$$\hat{y}(\mathbf{x}) = \mathbb{E} [y | \mathbf{x}]$$

- The *posterior predictive mean* is *optimal* wrt. the quadratic loss

Decision theory for regression II

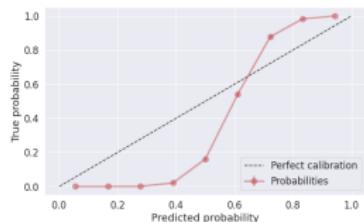
- The *Minkowski loss* is a generalization of the quadratic loss and is given by $\mathcal{L} = (\hat{y}(\mathbf{x}) - y)^q$
- One can show that ...
 - the posterior mean is optimal for $q = 2$
 - the posterior median is optimal for $q = 1$
 - the posterior mode is optimal for $q \rightarrow 0$
- $q = 2$ can be sensitive to outliers because of the quadratic form, while $q = 1$ is more robust



Calibration

Calibration for classification models

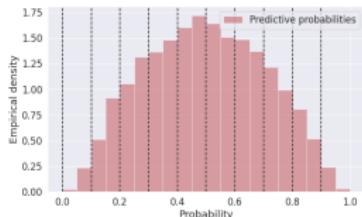
- We have seen that probabilities play an important role in decision-making, but how do we know if the probabilities are accurate?
- Training models via maximum likelihood or Bayesian methods should in theory yield calibrated models, however, in practice, models are rarely perfectly calibrated
- If we have an independent validation/test set, we can quantify the degree of calibration or miscalibration
- Among all the examples in the test set, where the predictive probability is approximately 80% we expect roughly 80% of the corresponding examples to belong to the positive class.
- Metrics for quantifying degree of miscalibration for classification
 1. Expected calibration error (ECE)
 2. Maximum calibration error
 3. Marginal calibration error
 4. Brier score



Expected calibration error (ECE)

- Compute predictions for validation set $\mathcal{D}_{\text{val}} = \{\mathbf{x}_m^*, y_m^*\}_{m=1}^M$
- Divide unit interval in B bins such that $I_b = \left(\frac{b-1}{B}, \frac{b}{B} \right]$
- Let \mathcal{B}_b be the set of indices of samples whose prediction confidence into interval and define

$$\begin{aligned}\hat{y}_m^* &= \arg \max_c p(y_m^* = c | \mathbf{y}, \mathbf{x}_m^*) \\ \hat{p}_m^* &= \max_c p(y_m^* = c | \mathbf{y}, \mathbf{x}_m^*)\end{aligned}$$



- Then *average accuracy* for bin b is defined as

$$\text{acc}(\mathcal{B}_b) = \frac{1}{|\mathcal{B}_b|} \sum_{m \in \mathcal{B}_b} \mathbb{I} [\hat{y}_m^* = y_m^*]$$

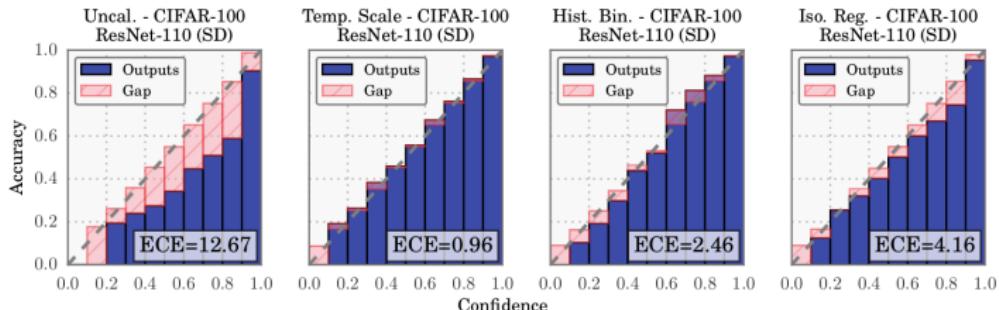
- .. and the *average confidence* for bin b is

$$\text{conf}(\mathcal{B}_b) = \frac{1}{|\mathcal{B}_b|} \sum_{m \in \mathcal{B}_b} \hat{p}_m^*$$

- The *expected calibration error (ECE)* is then given by

$$\text{ECE} = \sum_{b=1}^B \frac{|\mathcal{B}_b|}{M} |\text{acc}(\mathcal{B}_b) - \text{conf}(\mathcal{B}_b)|$$

Expected calibration error (ECE)



- Results on CIFAR-100 dataset from Guo et al, 2017 (and page 573 in Murphy2)
- Several methods for recalibration:
 1. Platt scaling
 2. Temperature scaling
 3. Histogram binning
 4. Isotonic regression
- In *temperature scaling*, we introduce a temperature parameter $T > 0$ in the softmax-function:
$$p(y|f(\mathbf{x}), T) = \text{Cat}(y|\text{softmax}(f(\mathbf{x})/T))$$
- T is estimated using a validation set.

Summary and take-aways

- Gaussian processes and neural networks are linked in many ways
- A suitable likelihood for your data generally gives better results
- We discussed generalized linear models (GLMs) and extensions to non-linear models like Gaussian process and neural networks
- Our goal is to minimize the *generalization error*, and we *estimate* the generalization error using cross-validation
- We talked about how we can use *confidence* and *entropy* to quantify the uncertainty for multi-class classification
- Bayesian decision theory: Uncertainties are crucial for optimal decision-making and therefore *calibration* becomes important
- The calibration error can be assessed via the *expected calibration error* (ECE) metric or *reliability curves*

$$\text{ECE} = \sum_{b=1}^B \frac{|\mathcal{B}_b|}{M} |\text{acc}(\mathcal{B}_b) - \text{conf}(\mathcal{B}_b)|$$

02477 – Bayesian Machine Learning: Lecture 8

Michael Riis Andersen

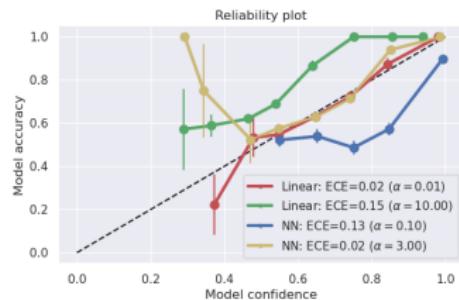
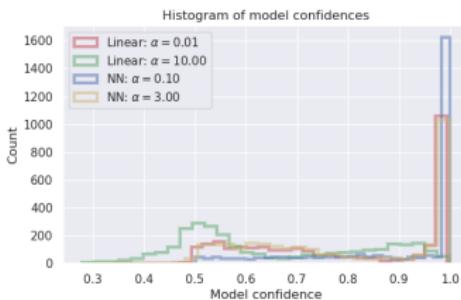
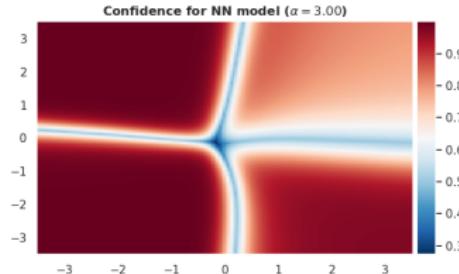
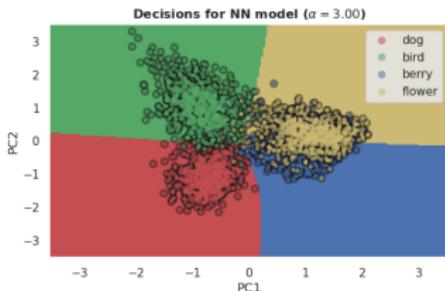
Technical University of Denmark,
DTU Compute, Department of Applied Math and Computer Science

Outline

- 1 More on calibration
- 2 Monte Carlo methods
- 3 Simple sampling methods
- 4 Markov Chain Monte Carlo methods

More on calibration

Calibration and reliability plots



Expected calibration error (ECE)

- Compute predictions for validation set $\mathcal{D}_{\text{val}} = \{\mathbf{x}_m^*, y_m^*\}_{m=1}^M$
- Divide unit interval in B bins such that $I_b = \left(\frac{b-1}{B}, \frac{b}{B}\right]$
- Let \mathcal{B}_b be the set of indices of samples whose prediction confidence into interval and define

$$\hat{y}_m^* = \arg \max_c p(y_m^* = c | \mathbf{y}, \mathbf{x}_m^*)$$

$$\hat{p}_m^* = \max_c p(y_m^* = c | \mathbf{y}, \mathbf{x}_m^*)$$

- Then the *average accuracy* and the *average confidence* for bin b is defined as

$$\text{acc}(\mathcal{B}_b) = \frac{1}{|\mathcal{B}_b|} \sum_{m \in \mathcal{B}_b} \mathbb{I} [\hat{y}_m^* = y_m^*]$$

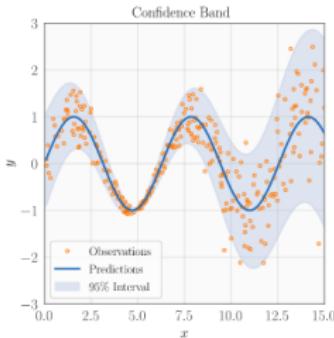
$$\text{conf}(\mathcal{B}_b) = \frac{1}{|\mathcal{B}_b|} \sum_{m \in \mathcal{B}_b} \hat{p}_m^*$$

- The *expected calibration error (ECE)* is then given by

$$\text{ECE} = \sum_{b=1}^B \frac{|\mathcal{B}_b|}{M} |\text{acc}(\mathcal{B}_b) - \text{conf}(\mathcal{B}_b)|$$

- Expected vs maximum calibration error

How about calibration for regression?



Images from the Python Uncertainty Toolbox ([link](#)).

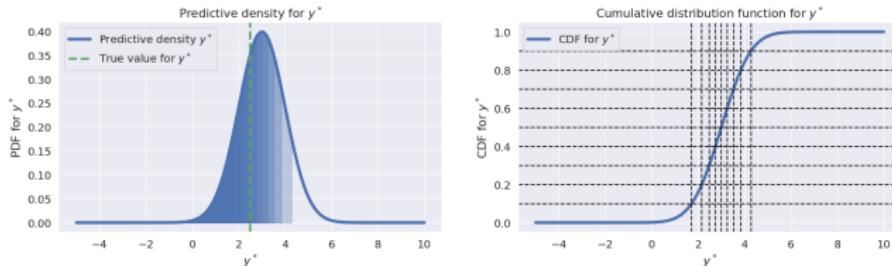
- How to measure calibration for regression? What's the natural quantity to bin?
- Recall: The CDF for predictive density $p(y^* | \mathbf{y}, \mathbf{x}^*) \equiv p(y^*)$

$$F(\tau) \equiv P(y^* \leq \tau) = \int_{-\infty}^{\tau} p(y^*) dy^* \in [0, 1]$$

Calibration for regression

- Recall: The CDF for predictive density $p(y^* | \mathbf{y}, \mathbf{x}^*) \equiv p(y^*)$

$$F(\tau) \equiv P(y^* \leq \tau) = \int_{-\infty}^{\tau} p(y^*) dy^* \in [0, 1]$$



- We discretize the unit interval again and for each value p we estimate

$$\hat{p} = \frac{\sum_{n=1}^N \mathbb{I}[y_n^* \leq F_n^{-1}(p)]}{N} \quad \text{if calibrated} \rightarrow p$$

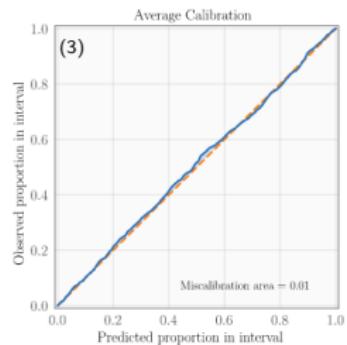
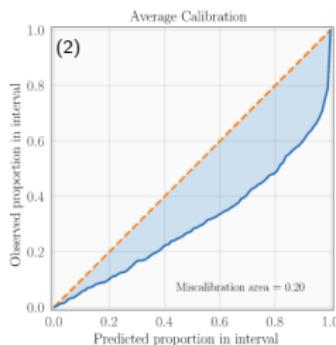
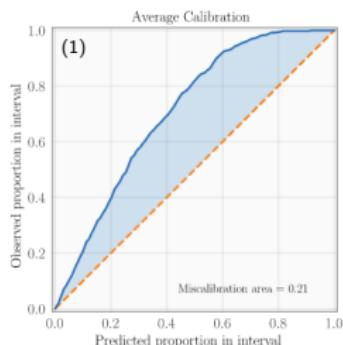
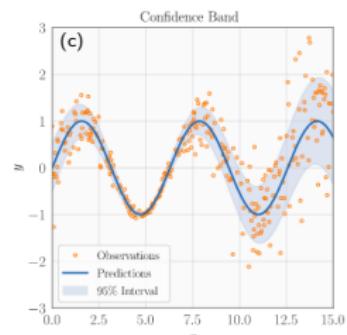
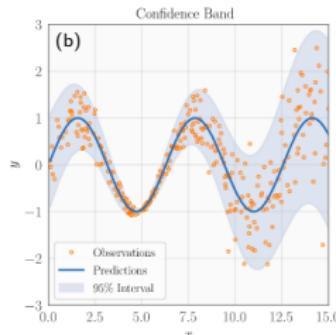
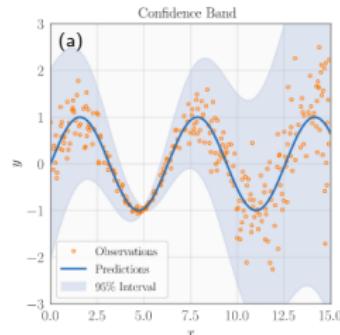
where F_n is the CDF of the predictive distribution for the n 'th datapoint.

- Extension to intervals for $p_1, p_2 \in [0, 1]$

$$\frac{\sum_{n=1}^N \mathbb{I}[F_n^{-1}(p_1) \leq y_n^* \leq F_n^{-1}(p_2)]}{N} \quad \text{if calibrated} \rightarrow p_2 - p_1$$

DTU Learn quiz: Calibration for regression

Three fits, one is well-calibrated, one is overconfident, and one is underconfident. Which is which? And which fit matches which calibration curve?



Images from the Python Uncertainty Toolbox ([link](#)).

Monte Carlo methods

Bayesian methods and those annoying integrals...

- We compute the posterior distribution of the weights \mathbf{w} given the data \mathbf{y} using Bayes rule

$$p(\mathbf{w}|\mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{w})p(\mathbf{w})}{p(\mathbf{y})}$$

- When making predictions, we evaluate the predictive distribution

$$p(y^*|\mathbf{y}, \mathbf{x}^*) = \int p(y^*|\mathbf{w}, \mathbf{x}^*)p(\mathbf{w}|\mathbf{y})d\mathbf{w}$$

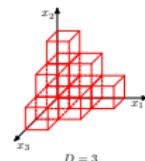
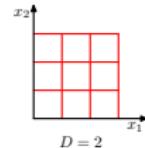
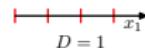
- For most models of practical interest, both requires intractable integrals.
- Conjugate priors: often a bit like “getting the right answer to the wrong question.”
- For general Bayesian inference, we need approximate inference methods.
 1. Laplace approximations.
 2. Variational approximations.
 3. Markov Chain Monte Carlo methods.

Monte Carlo methods

- Stanisław Ulam invented *Markov Chain Monte Carlo methods* in the 1940s while working on the first nuclear bomb with John von Neumann and Nicholas Metropolis.



- To keep their method a secret, they used a code name: *Monte Carlo*.
- Named after the famous casino in Monte Carlo, Monaco.



- *Monte Carlo methods* is a common term for algorithms that relies on *random sampling*.
- Monte Carlo integration breaks the *curse of dimensionality*.

$$\mathbb{E}_p[f(z)] = \int f(z)p(z)dz$$

Managing expectations...

Many posterior summaries of interest can be cast as expectations.

- The mean

$$\mathbb{E}[\mathbf{w}] = \int \mathbf{w} p(\mathbf{w}|\mathbf{y}) d\mathbf{w} = \mathbb{E}[f(\mathbf{w})] \quad \text{for } f(\mathbf{w}) = \mathbf{w}$$

- The variance

$$\mathbb{V}[\mathbf{w}] = \int (\mathbf{w} - \mathbb{E}[\mathbf{w}])^2 p(\mathbf{w}|\mathbf{y}) d\mathbf{w} = \mathbb{E}[f(\mathbf{w})] \quad \text{for } f(\mathbf{w}) = (\mathbf{w} - \mathbb{E}[\mathbf{w}])^2$$

- Probabilitites

$$\mathbb{P}(w > \tau) = \int_{\tau}^{\infty} p(w|\mathbf{y}) dw = \mathbb{E}[f(w)] \quad \text{for } f(w) = \mathbb{I}[w > \tau]$$

- The predictive density

$$p(y^*|\mathbf{y}, \mathbf{x}_*) = \int p(y^*|\mathbf{w}, \mathbf{x}^*) p(\mathbf{w}|\mathbf{y}) d\mathbf{w} = \mathbb{E}[f(\mathbf{w})] \quad \text{for } f(\mathbf{w}) = p(y^*|\mathbf{w}, \mathbf{x}^*)$$

Monte Carlo integration I

- Our goal is to estimate the expectation

$$\bar{f} = \mathbb{E}[f(\mathbf{z})] = \int f(\mathbf{z}) p(\mathbf{z}) d\mathbf{z}$$

- Suppose we can get a number of *i.i.d.* samples from the distribution $\mathbf{z}^i \sim p(\mathbf{z})$, then the *Monte Carlo (MC) estimator* is given by

$$\bar{f} \approx \hat{f} = \frac{1}{S} \sum_{i=1}^S f(\mathbf{z}^i)$$

- The Monte Carlo estimator \hat{f} is a *random variable*.

- Let's calculate the mean of the Monte Carlo estimator

$$\mathbb{E}[\hat{f}] = \mathbb{E}\left[\frac{1}{S} \sum_{i=1}^S f(\mathbf{z}^i)\right] = \frac{1}{S} \sum_{i=1}^S \mathbb{E}[f(\mathbf{z}^i)] = \frac{1}{S} \sum_{i=1}^S \bar{f} = \bar{f}$$

- Hence, the Monte Carlo estimator is *unbiased*.

Monte Carlo integration II

- The Monte Carlo estimator is

$$\bar{f} \approx \hat{f} = \frac{1}{S} \sum_{i=1}^S f(\mathbf{z}^i)$$

- Let's calculate the variance

$$\mathbb{V}[\hat{f}] = \mathbb{V}\left[\frac{1}{S} \sum_{i=1}^S f(\mathbf{z}^i)\right] = \frac{1}{S^2} \mathbb{V}\left[\sum_{i=1}^S f(\mathbf{z}^i)\right] = \frac{1}{S^2} \sum_{i=1}^S \mathbb{V}[f(\mathbf{z}^i)] = \frac{1}{S} \mathbb{V}[f(\mathbf{z})]$$

- Remarkably, the variance of \hat{f} is *independent of the dimension* of \mathbf{z} .
- The variance decreases with $1/S$ and the standard deviation decreases with $1/\sqrt{S}$.
- Law of large numbers:** If *i.i.d.* samples $\mathbf{z}^i \sim p(\mathbf{z})$, then \hat{f} converges (in prob.) to \bar{f} as $S \rightarrow \infty$,

$$\hat{f} = \frac{1}{S} \sum_{i=1}^S f(\mathbf{z}^i) \quad \rightarrow \quad \bar{f} = \int f(\mathbf{z}) p(\mathbf{z}) d\mathbf{z},$$

assuming \bar{f} exists.

Monte Carlo integration III

- The Monte Carlo estimator and its variance:

$$\bar{f} \approx \hat{f} = \frac{1}{S} \sum_{i=1}^S f(\mathbf{z}^i), \quad \mathbb{V} [\hat{f}] = \frac{1}{S} \mathbb{V} [f(\mathbf{z})]$$

- The *Monte Carlo Standard Error* (MCSE) is the expected difference between \bar{f} and \hat{f} :

$$\text{MCSE}_f = \frac{1}{\sqrt{S}} \sqrt{\mathbb{V} [f(\mathbf{z})]}$$

Exercise: DTU Learn quiz: The Monte Carlo Standard Error (Lecture 8)

- Suppose we use S samples to estimate \bar{f} and the resulting MCSE is 10.
- If our goal is to reduce the MCSE by a factor of 10, how many samples S' should we use instead?

Example

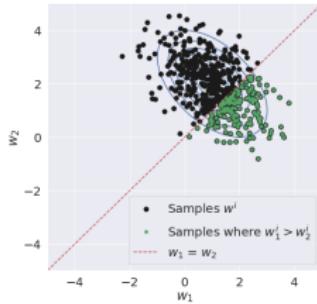
- Suppose our parameter of interest is $\mathbf{w} \in \mathbb{R}^2$ and that $\mathbf{w} \sim \mathcal{N}(\mu, \Sigma)$. What is the probability of $w_1 > w_2$?

- First we phrase this probability as an expectation

$$P(w_1 > w_2) = \int \mathbb{I}[w_1 > w_2] \mathcal{N}(\mu, \Sigma) d\mathbf{w} = \mathbb{E}[\mathbb{I}[w_1 > w_2]]$$

- We generate samples $\mathbf{w}^i \sim \mathcal{N}(\mu, \Sigma)$ for $S = 500$, then

$$P(w_1 > w_2) \approx \frac{1}{S} \sum_{i=1}^S \mathbb{I}\left[w_1^i > w_2^i\right] = 0.314$$



- The empirical variance is

$$\hat{\mathbb{V}}\left[\mathbb{I}\left[w_1^i > w_2^i\right]\right] = 0.215$$

- The Monte Carlo Standard Error (MCSE) is

$$\frac{1}{\sqrt{S}} \sqrt{\mathbb{V}[f(\mathbf{z})]} \approx \frac{1}{\sqrt{500}} \sqrt{\hat{\mathbb{V}}[f(\mathbf{z})]} \approx 0.021$$

- We conclude: $P(w_1 > w_2) \approx 0.31 \pm 0.02$

Simple sampling methods

Ancestral sampling

- Useful for *sampling* from *joint* distributions.
- Consider our linear regression model from week 3:

$$p(\mathbf{y}, \mathbf{w} | \kappa^2, \sigma^2) = \mathcal{N}(\mathbf{y} | \mathbf{X}\mathbf{w}, \sigma^2 \mathbf{I}) \mathcal{N}(\mathbf{w} | \mathbf{0}, \kappa^2 \mathbf{I})$$

- We optimized the hyperparameters $\kappa^2 = \alpha^{-1}$ and $\sigma^2 = \beta^{-1}$ using the marginal likelihood.
- What if we want to be fully Bayesian and impose *hyperpriors* to the hyperparameters?
- Let's impose a *half-normal distribution* on κ^2 and σ^2

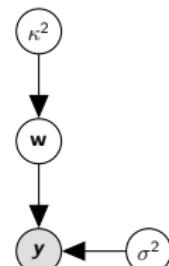
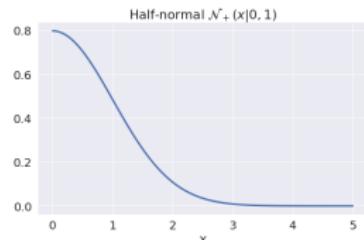
$$p(\kappa^2) = \mathcal{N}_+(\kappa^2 | 0, 1)$$

$$p(\sigma^2) = \mathcal{N}_+(\sigma^2 | 0, 1)$$

- The joint distribution of the model becomes

$$p(\mathbf{y}, \mathbf{w}, \kappa^2, \sigma^2) = p(\mathbf{y} | \mathbf{w}, \sigma^2) p(\mathbf{w} | \kappa^2) p(\sigma^2) p(\kappa^2)$$

- We can use *ancestral sampling* to generate samples from the joint distribution $p(\mathbf{y}, \mathbf{w}, \kappa^2, \sigma^2)$ and to understand the new model assumptions.

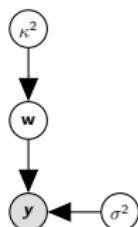


Ancestral sampling

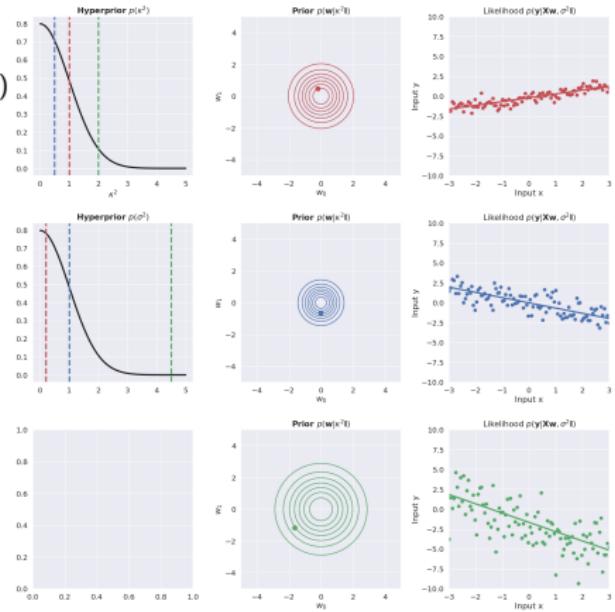
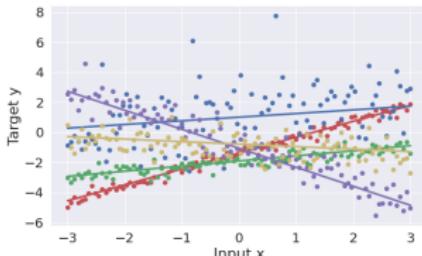
- The joint distribution of the model becomes

$$p(\mathbf{y}, \mathbf{w}, \kappa^2, \sigma^2) = p(\mathbf{y}|\mathbf{w}, \sigma^2)p(\mathbf{w}|\kappa^2)p(\kappa^2)p(\sigma^2)$$

- Let's sample some "fake" datasets using this model.



- A few more examples:



Rejection sampling

- *Rejection sampling* is a simple technique for generating samples from $p(z)$,

$$p(z) = \frac{1}{Z} \tilde{p}(z)$$

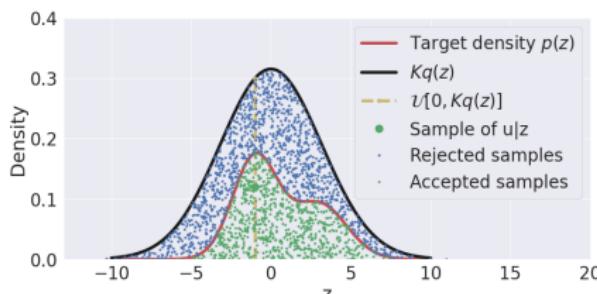
- Let $q(z)$ be an "easy" distribution and $K > 0$ a constant such that

$$\tilde{p}(z) \leq Kq(z) \quad \text{for all } z$$

- Steps in the sampling procedure:

1. Sample $z \sim q(z)$.
2. Sample $u|z$ from a uniform distribution: $u \sim \mathcal{U}[0, Kq(z)]$.
3. If $u > \tilde{p}(z)$ reject sample, otherwise keep.

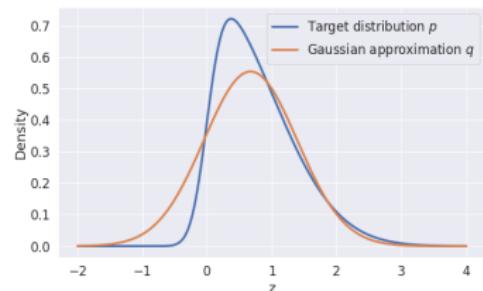
- Pros: easy to use and works well in low dimensions, Cons: In high dimensions, acceptance rate can be low.



Importance sampling

- *Importance sampling* is a technique for estimating the expectation of a distribution $p(z)$ when we cannot sample from p directly.
- Let q be an “easy” distribution.

$$\begin{aligned}\mathbb{E}_p[f(z)] &= \int f(z)p(z)dz \\ &= \int f(z)\frac{q(z)}{q(z)}p(z)dz \\ &= \int f(z)\frac{p(z)}{q(z)}q(z)dz \\ &= \mathbb{E}_q\left[f(z)\frac{p(z)}{q(z)}\right] \\ &\approx \frac{1}{S} \sum_{i=1}^S f(\mathbf{z}^i) \frac{p(\mathbf{z}^i)}{q(\mathbf{z}^i)} \equiv \hat{f} \quad \text{for } \mathbf{z}^i \sim q(\mathbf{z})\end{aligned}$$



- The ratios $w_i = \frac{p(\mathbf{z}^i)}{q(\mathbf{z}^i)}$ are called importance weights. The *variance* of the estimator

$$\hat{\sigma}_q^2 = \frac{1}{S} \sum_{i=1}^S (w_i f(\mathbf{z}^i) - \hat{f})^2$$

How to generate random numbers?

- Assuming we can generate random numbers in the unit interval,

$$u \sim \mathcal{U}[0, 1]$$

- Sampling the Bernoulli distribution $z \sim \text{Ber}(\pi)$:

$$z = \begin{cases} 1 & \text{if } u < \pi \\ 0 & \text{otherwise} \end{cases}$$

- The Box-Muller transform:

$$z = \sqrt{-2 \ln u_1} \cos(2\pi u_2) \sim \mathcal{N}(0, 1)$$

- Location-scale families:

$$x = a + bz \sim \mathcal{N}(x|a, b^2)$$

- Sampling the Exponential-distribution $z \sim \text{Exp}(\lambda)$:

$$z = \frac{-\ln u}{\lambda}$$

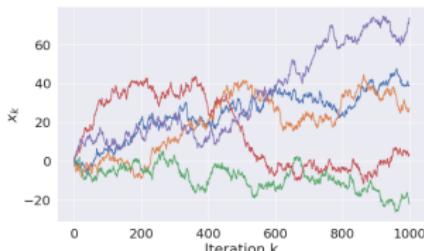
- Will take sampling from “easy” distribution for granted: uniform, Gaussian, gamma, Bernoulli, binomial, Poisson etc. and focus on more complex distributions.

Markov Chain Monte Carlo methods

Markov Chain Monte Carlo methods

- It turns out it is often really hard to obtain *i.i.d.* samples from an arbitrary distribution $p(\mathbf{w})$.
- *Markov Chain Monte Carlo* (MCMC) provides a way to sample from almost any distribution $p(\mathbf{w})$.
- A recent survey places the Metropolis algorithm among the ten algorithms that have had the greatest influence on the development and practice of science and engineering in the 20th century. (Andrieu, 2003).
- The idea is to give up the requirement of i.i.d. samples and construct a "random walk" with specific properties.
- The generated samples are thus correlated, so the variance of the estimator reduces more slowly than for standard Monte Carlo.
- A simple example of a Markov random walk:

$$x_{k+1} = x_k + e_k, \quad \mathcal{N}(0, \sigma^2)$$



Markov chains in a nutshell

- A first-order Markov chain is defined to be a series of random variables $\mathbf{z}^{(1)}, \mathbf{z}^{(2)}, \dots, \mathbf{z}^{(m)}$ with the following conditional independence property

$$p(\mathbf{z}^{m+1} | \mathbf{z}^{(1)}, \mathbf{z}^{(2)}, \dots, \mathbf{z}^{(m)}) = p(\mathbf{z}^{m+1} | \mathbf{z}^{(m)})$$

- "If you know the present, forget about the past."

$$x_{k+1} = x_k + e_k, \quad e_k \sim \mathcal{N}(0, \sigma^2)$$

- Implications of the Markov assumption:

$$\begin{aligned} p(z_1, z_2, z_3, z_4) &= p(z_4 | z_1, z_2, z_3)p(z_3 | z_1, z_2)p(z_2 | z_1)p(z_1) \\ &= p(z_4 | z_3)p(z_3 | z_2)p(z_2 | z_1)p(z_1) \end{aligned}$$

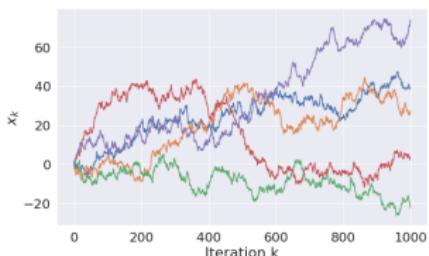
- The *transition kernel* is defined as

$$T_m(\mathbf{z}^{(m)}, \mathbf{z}^{(m+1)}) \equiv p(\mathbf{z}^{(m+1)} | \mathbf{z}^{(m)})$$

- T_m is *homogeneous* if it is the same for all m .

- Questions:

- What is the transition kernel for the random walk of x_k ?
- Is it homogeneous?



Markov chains for MCMC

- The distribution of $\mathbf{z}^{(m+1)}$ is calculated using the *sum rule*:

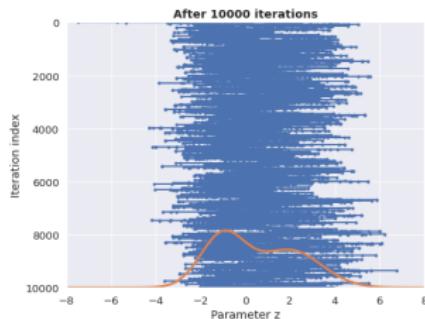
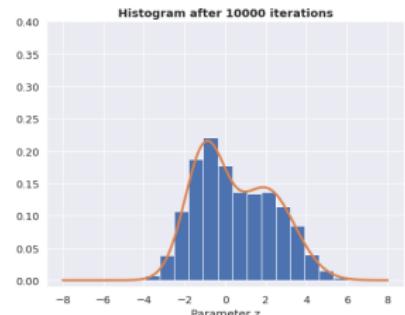
$$p(\mathbf{z}^{(m+1)}) = \int p(\mathbf{z}^{(m+1)} | \mathbf{z}^{(m)}) p(\mathbf{z}^{(m)}) d\mathbf{z}^{(m)}$$

- A distribution $p^*(\mathbf{z})$ is said to be *invariant* or *stationary* wrt. the Markov chain if each step does not change the distribution

$$p^*(\mathbf{z}) = \int T(\mathbf{z}', \mathbf{z}) p^*(\mathbf{z}') d\mathbf{z}'$$

- Central idea of MCMC: design a Markov chain such that the target distribution $p(\mathbf{z})$ is invariant wrt. the chain.

- Informally: we design a very specific random walk that will explore the space of $p(\mathbf{z})$ in a way, where the random walk spends most time in the regions of high density.



Markov Chain Monte Carlo and the Metropolis algorithm

- The *Metropolis* and the *Metropolis-Hastings* algorithms give a recipe for designing such Markov chains.
- Suppose our goal is to sample from the target distribution $p(\theta)$ and we assume $q(\theta^* | \theta^{k-1})$ to be a *symmetric proposal distribution*.

The Metropolis algorithm

- Start from some initial value θ^0 .
 - Repeat for $k = 1$ to K :
 1. Given the last value θ^{k-1} , generate a *candidate sample* using the proposal distribution
$$\theta^* \sim q(\theta^* | \theta^{k-1})$$
 2. Compute the *acceptance probability* A_k ,
- $$A_k = \min \left(1, \frac{p(\theta^*)}{p(\theta^{k-1})} \right)$$
3. Simulate $u_k \sim \mathcal{U}(0, 1)$ and define θ^k as

$$\theta^k = \begin{cases} \theta^* & \text{if } u_k < A_k \\ \theta^{k-1} & \text{otherwise} \end{cases}$$

The Metropolis algorithm: example

- Suppose we use a Gaussian proposal,

$$q(\theta^* | \theta^{k-1}) = \mathcal{N}(\theta^* | \theta^{k-1}, \tau I)$$

- Key steps in the Metropolis algorithm

- Generate a *candidate sample*:

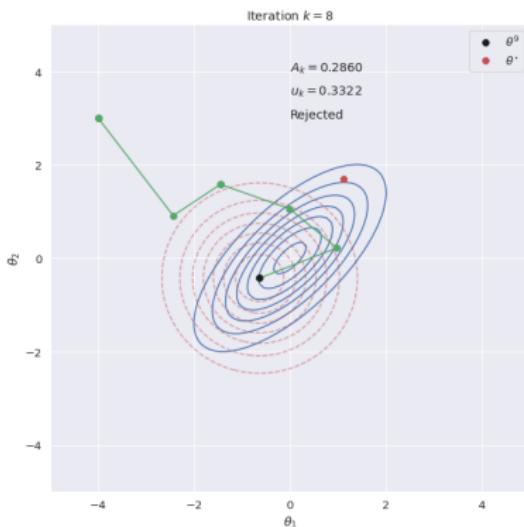
$$\theta^* \sim q(\theta^* | \theta^{k-1})$$

- Compute the *acceptance probability*:

$$A_k = \min \left(1, \frac{p(\theta^*)}{p(\theta^{k-1})} \right)$$

- Simulate $u_k \sim \mathcal{U}(0, 1)$ and define θ^k as

$$\theta^k = \begin{cases} \theta^* & \text{if } u_k < A_k \\ \theta^{k-1} & \text{otherwise} \end{cases}$$



A lot of questions...

- Wait? We don't know the density values for the posterior distribution?

$$p(\theta|y) = \frac{p(y|\theta)p(\theta)}{p(y)} = \frac{p(y, \theta)}{p(y)}$$

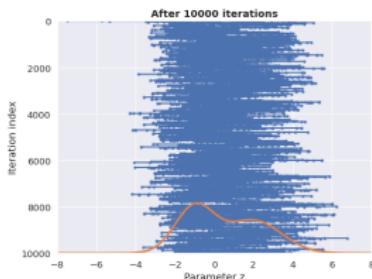
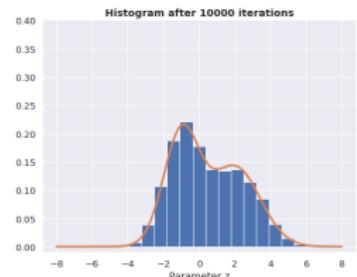
- *The normalization constant cancels out* in the acceptance probabilities

$$A_k = \min \left(1, \frac{p(\theta^*|y)}{p(\theta^{k-1}|y)} \right)$$

- Therefore, we only need to be able evaluate $p(\theta|y)$ up to a constant.

- When do we know whether the chain has converged?

- How many samples to collect?



02477 – Bayesian Machine Learning: Lecture 9

Michael Riis Andersen

Technical University of Denmark,
DTU Compute, Department of Applied Math and Computer Science

Outline

1 Markov chain Monte Carlo

2 Gibbs sampling

3 Convergence diagnostics

4 Hierarchical models

Markov chain Monte Carlo

Bayesian inference and probabilistic modelling

- Bayesian supervised learning in general:

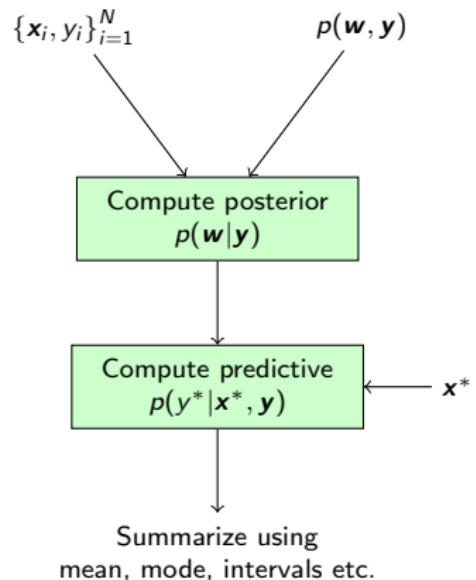
1. Joint model of data \mathbf{y} and parameters \mathbf{w} .
2. Summarize knowledge of \mathbf{w} given data \mathbf{y} .
3. Compute posterior predictive distribution.

- Goal: separate modelling from inference:

1. Build models reflecting domain knowledge.
2. Push “inference button” and get results.

- Inference methods:

1. Laplace approximations.
2. Markov chain Monte Carlo.
3. Variational approximations.



Monte Carlo: Posterior inference using samples

- Many posterior summaries can be phrased as expectations $\mathbb{E}_p [f(\mathbf{z})]$ for some function f
- We can compute expectations wrt. p using the *Monte Carlo estimator*

$$\bar{f} = \mathbb{E}_p [f(\mathbf{z})] = \int f(\mathbf{z}) p(\mathbf{z}) d\mathbf{z} \approx \frac{1}{S} \sum_{i=1}^S f(\mathbf{z}^i) \equiv \hat{f},$$

where $\mathbf{z}^i \sim p(\mathbf{z})$ for $i = 1, \dots, S$

- We showed last time that ...
 1. the Monte Carlo estimator \hat{f} is *unbiased*
 2. the variance of \hat{f} decreases with $1/S$ when the samples are *i.i.d.*

A zoo of sampling-based methods

■ Simple sampling methods

1. Rejection sampling
2. Ancestral sampling
3. Importance sampling
4. Transformation methods
5. Inverse transform sampling
6. ...

■ MCMC methods

1. Metropolis-Hastings
2. Gibbs Sampling
3. Slice sampling
4. Hamiltonian Monte Carlo
5. ...

MCMC using the Metropolis-Hastings algorithm

- We can use the MH to generate samples from a distribution of interest $p(\mathbf{z})$.

The Metropolis-Hastings algorithm

- Start from some initial value \mathbf{z}^1 (e.g., a sample from the prior).
- Repeat for $k = 1$ to K :
 1. Given last value \mathbf{z}^{k-1} , generate *candidate sample* using proposal distribution
$$\mathbf{z}^* \sim q(\mathbf{z}^* | \mathbf{z}^{k-1}).$$
 2. Compute *acceptance probability* A_k as follows

$$A_k = \min \left(1, \frac{p(\mathbf{z}^*) q(\mathbf{z}^{k-1} | \mathbf{z}^*)}{p(\mathbf{z}^{k-1}) q(\mathbf{z}^* | \mathbf{z}^{k-1})} \right).$$

3. Simulate $u_k \sim \mathcal{U}(0, 1)$ and define \mathbf{z}^k as

$$\mathbf{z}^{k+1} = \begin{cases} \mathbf{z}^* & \text{if } u_k < A_k \\ \mathbf{z}^{k-1} & \text{otherwise} \end{cases}$$

- What do we need in order to implement MH for a given model?

Markov chain Monte Carlo theory I

- Metropolis-Hastings defines a chain of samples $\mathbf{z}^0, \mathbf{z}^1, \mathbf{z}^2, \dots$ with a *Markov property*

$$p(\mathbf{z}^{k+1} | \mathbf{z}^1, \mathbf{z}^2, \dots, \mathbf{z}^k) = p(\mathbf{z}^{k+1} | \mathbf{z}^k)$$

- The *transition kernel* tells us how to iterate the chain

$$T(\mathbf{z}^{k+1} | \mathbf{z}^k) \equiv p(\mathbf{z}^{k+1} | \mathbf{z}^k)$$

- The distribution of \mathbf{z}^{k+1} is given by *sum rule*

$$p(\mathbf{z}^{k+1}) = \int T(\mathbf{z}^{k+1} | \mathbf{z}^k) p(\mathbf{z}^k) d\mathbf{z}^k$$

- A distribution $p^*(\mathbf{z})$ is said to be *invariant* or *stationary* wrt. the Markov chain if each step does not change the distribution

$$p^*(\mathbf{z}) = \int T(\mathbf{z} | \mathbf{z}') p^*(\mathbf{z}') d\mathbf{z}'$$

- We require $p^*(\mathbf{z})$ to be a limiting distribution of the chain (*independent of the initial distribution*).

$$p(\mathbf{z}^k) \rightarrow p^*(\mathbf{z}) \quad \text{for} \quad k \rightarrow \infty$$

Transistion kernel for Metropolis-Hastings

- Recall the acceptance probability for Metropolis-Hastings

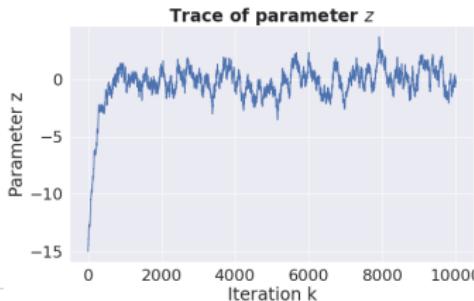
$$A(z^*|z^k) = \min \left[1, \frac{p(z^*)q(z^k|z^*)}{p(z^k)q(z^*|z^k)} \right]$$

- The transition kernel for Metropolis-Hastings

$$T(z'|z) = \begin{cases} q(z'|z)A(z'|z) & \text{if } z' \neq z \\ q(z|z)A(z|z) + \int q(z''|z) [1 - A(z''|z)] dz'' & \text{if } z' = z \end{cases}$$

- The big picture

- We initialize z^1 .
- We iterate $z^{k+1}|z^k \sim T(z^{k+1}|z^k)$ (*warm-up phase*).
- Eventually the distribution of z^k will converge to the target distribution p^* (*sampling phase*).



Markov chain Monte Carlo theory II

$$p^*(z) = \int T(z|z') p^*(z') dz'$$

- We require $p^*(z)$ to be a limiting distribution of the chain (*independent of the initial distribution*).

$$p(z^k) \rightarrow p^*(z) \quad \text{for } k \rightarrow \infty$$

- Conditions required for a Markov chain to have a limiting distribution equal to the unique stationary distribution:

(A1) *Irreducible*: all states z can be reached.

Counter example: $z^{k+1} = z^k + |e_k|$, where $e_k \sim \mathcal{N}(0, 1)$

(A2) *Aperiodic*: no deterministic cycles.

Counter example: $z^1 = 1, z^2 = 2, z^3 = 3, z^4 = 1, z^5 = 2, \dots$

(A3) *Positive recurrent*: the chain has positive probability of returning to any given state (+ finite expected return-time).

$P(z^k \in A | z^0) > 0$ for all sets A where $P(A) > 0$

- A chain that satisfies (A1)–(A3) is said to be *ergodic* and ensures $p(z^k) \rightarrow p^*(z)$.

Markov chain Monte Carlo theory III

- These conditions are generally hard to check in practice.
- Simpler condition: if a chain satisfies the *detailed balance condition*, then p^* is its stationary distribution

$$T(z'|z)p^*(z) = T(z|z')p^*(z').$$

- MH with reasonable proposal distributions satisfies detailed balance (see slide 33).
- It's often not a question of convergence or not, but rather how fast we converge for a given proposal distribution.
- For example, all (non-degenerate) Gaussian proposals lead to detailed balance, but convergence time may vary drastically depending on the proposal variance.

- More theory and details: Monte Carlo Statistical Methods by Robert and Casella.



Pros and cons of Metropolis-Hastings

Pros

- Strong mathematical guarantees: If we sample long enough, the iterates z^k will converge to the exact target distribution

$$p(z^k) \rightarrow p^*(z) \quad \text{for } k \rightarrow \infty$$

- Easy to implement.
- Easy to prototype and evaluate different models.

Cons

- May have to sample “infinitely” long for difficult distributions.
- Acceptance ratio can be low.
- Slow for large datasets.
- Proposal distribution may require tuning.

Questions: True or false?

Quiz via DTU Learn:

Lecture 9: Metropolis-Hastings (12 questions)

Check your knowledge

Gibbs sampling

Gibbs sampling

- When using Metropolis-Hastings,
 1. we have to choose a proposal distribution (and sometimes tune it), and
 2. it may suffer from low acceptance rates.
- *Gibbs sampling* works by iteratively updating each coordinate of \mathbf{z} by sampling from the posterior conditionals $p(z_i|\mathbf{z}_{-i})$ (\mathbf{z}_{-i} means the entire vector except index i).

The Gibbs Sampler

- Initialize all parameter values $\{\mathbf{z}_i^0\}_{i=1}^D$
- Repeat for $k = 1$ to K :
 - Sample $\mathbf{z}_1^k \sim p(\mathbf{z}_1|\mathbf{z}_2^{k-1}, \mathbf{z}_3^{k-1}, \dots, \mathbf{z}_D^{k-1})$.
 - Sample $\mathbf{z}_2^k \sim p(\mathbf{z}_2|\mathbf{z}_1^k, \mathbf{z}_3^{k-1}, \dots, \mathbf{z}_D^{k-1})$.
 - Sample $\mathbf{z}_3^k \sim p(\mathbf{z}_3|\mathbf{z}_1^k, \mathbf{z}_2^k, \mathbf{z}_4^{k-1}, \dots, \mathbf{z}_D^{k-1})$
 - Sample ...
 - Sample $\mathbf{z}_D^k \sim p(\mathbf{z}_D|\mathbf{z}_1^k, \mathbf{z}_2^k, \mathbf{z}_3^k, \dots, \mathbf{z}_{D-1}^k)$.

Example: Gaussian linear model I

- Suppose we want to derive a Gibbs sampler for the following target distribution

$$y|w \sim \mathcal{N}(y|w_1x_1 + w_2x_2, \sigma^2)$$

$$w_1 \sim \mathcal{N}(w_1|0, \kappa^2)$$

$$w_2 \sim \mathcal{N}(w_2|0, \kappa^2)$$

- The posterior distribution is proportional to the joint density $p(y, w_1, w_2)$

$$\begin{aligned} p(w_1, w_2|y) &= \frac{p(y|w_1, w_2)p(w_1)p(w_2)}{p(y)} \\ &\propto p(y|w_1, w_2)p(w_1)p(w_2) \\ &= \mathcal{N}(y|w_1x_1 + w_2x_2, \sigma^2)\mathcal{N}(w_1|0, \kappa^2)\mathcal{N}(w_2|0, \kappa^2) \end{aligned}$$

- Gibbs sampling* requires us to derive the *posterior conditionals* $p(w_1|y, w_2)$ and $p(w_2|y, w_1)$

- General technique* for identifying $p(w_i|y, \mathbf{w}_{-i})$:

1. Write up the log joint density.
2. Identify all quantities that depends on w_i and ignore the rest.
3. Identify the distribution $p(w_i|y, \mathbf{w}_{-i})$ from its *functional form*.

Example: Gaussian linear model II

- Write out the logarithm of the joint density

$$p(w_1, w_2 | y) \propto \mathcal{N}(t | w_1 x_1 + w_2 x_2, \sigma^2) \mathcal{N}(w_1 | 0, \kappa^2) \mathcal{N}(w_2 | 0, \kappa^2)$$

- Recall the expression for a Gaussian density

$$\mathcal{N}(x | m, v) = \frac{1}{\sqrt{2\pi v}} \exp\left(-\frac{(x - m)^2}{2v}\right)$$

- Let's write it out the log density and identify all terms that depend on w_1 or w_2

$$\begin{aligned} \log p(w_1, w_2 | y) &= -\frac{1}{2} \log(2\pi\sigma^2) - \frac{1}{2\sigma^2} (y - w_1 x_1 - w_2 x_2)^2 + \\ &\quad - \frac{1}{2} \log(2\pi\kappa^2) - \frac{1}{2\kappa^2} w_1^2 - \frac{1}{2} \log(2\pi\kappa^2) - \frac{1}{2\kappa^2} w_2^2 + K \\ &= -\frac{1}{2\sigma^2} (y - w_1 x_1 - w_2 x_2)^2 - \frac{1}{2\kappa^2} w_1^2 - \frac{1}{2\kappa^2} w_2^2 + K' \\ &= -\frac{1}{2\sigma^2} (y^2 + (w_1 x_1 + w_2 x_2)^2 - 2y(w_1 x_1 - w_2 x_2)) - \frac{1}{2\kappa^2} w_1^2 - \frac{1}{2\kappa^2} w_2^2 + K' \\ &= -\frac{1}{2\sigma^2} (w_1 x_1 + w_2 x_2)^2 + \frac{1}{\sigma^2} y(w_1 x_1 - w_2 x_2) - \frac{1}{2\kappa^2} w_1^2 - \frac{1}{2\kappa^2} w_2^2 + K'' \\ &= -\frac{1}{2\sigma^2} (w_1^2 x_1^2 + w_2^2 x_2^2 + 2w_1 x_1 w_2 x_2) + \frac{1}{\sigma^2} y(w_1 x_1 - w_2 x_2) - \frac{1}{2\kappa^2} w_1^2 - \frac{1}{2\kappa^2} w_2^2 + K'' \end{aligned}$$

Example: Gaussian linear model III

- We just arrived at

$$\log p(w_1, w_2 | y) = -\frac{1}{2\sigma^2}(w_1^2 x_1^2 + w_2^2 x_2^2 + 2w_1 x_1 w_2 x_2) + \frac{1}{\sigma^2} y(w_1 x_1 - w_2 x_2) - \frac{1}{2\kappa^2} w_1^2 - \frac{1}{2\kappa^2} w_2^2 + K''$$

- Let's compare that to a generic Gaussian distribution:

$$\begin{aligned}\log \mathcal{N}(w_1 | m, v) &= -\frac{1}{2} \log(2\pi v) - \frac{1}{2v} (w_1 - m)^2 \\ &= -\frac{1}{2} \log(2\pi v) - \frac{1}{2v} (w_1^2 + m^2 - 2w_1 m) = -\frac{1}{2v} w_1^2 + \frac{1}{v} mw_1 + C\end{aligned}$$

- Recall: *The functional form* of the logarithm of a Gaussian density is a quadratic function.

- Identify the distribution $p(w_1 | y, w_2)$ based on the functional dependence on w_1 :

$$\begin{aligned}\log p(w_1 | y, w_2) &= -\frac{1}{2\sigma^2}(w_1^2 x_1^2 + w_2^2 x_2^2 + 2w_1 x_1 w_2 x_2) + \frac{1}{\sigma^2} y(w_1 x_1 - w_2 x_2) - \frac{1}{2\kappa^2} w_1^2 - \frac{1}{2\kappa^2} w_2^2 + K'' \\ &= -\frac{1}{2\sigma^2}(w_1^2 x_1^2 + 2w_1 x_1 w_2 x_2) + \frac{1}{\sigma^2} y w_1 x_1 - \frac{1}{2\kappa^2} w_1^2 + K''' \\ &= -\frac{1}{2} \left(\frac{1}{\sigma^2} x_1^2 + \frac{1}{\kappa^2} \right) w_1^2 + \left(\frac{1}{\sigma^2} y x_1 - \frac{1}{\sigma^2} x_1 w_2 x_2 \right) w_1 + K'''\end{aligned}$$

- We conclude that the distribution $\log p(w_1 | y, w_2)$ must be a Gaussian, because *its functional form is quadratic wrt. w_1 .*

Example: Gaussian linear model IV

- A generic Gaussian distribution

$$\ln \mathcal{N}(w_1|m, v) = -\frac{1}{2v} w_1^2 + \frac{1}{v} mw_1 + C$$

- We know $p(w_1|y, w_2) = \mathcal{N}(w_1|m_1, v_1)$ is Gaussian, so all we need is a mean and variance

$$\log p(w_1|y, w_2) = -\frac{1}{2} \left(\frac{1}{\sigma^2} x_1^2 + \frac{1}{\kappa^2} \right) w_1^2 + \left(\frac{1}{\sigma^2} yx_1 - \frac{1}{\sigma^2} x_1 w_2 x_2 \right) w_1 + K'''$$

- *Comparing the coefficients* for the *second order term* w_1^2 , we get the variance

$$v_1 = \left(\frac{1}{\sigma^2} x_1^2 + \frac{1}{\kappa^2} \right)^{-1}$$

- and by comparing coefficients for the *first order* term w_1 , we get the mean

$$\frac{m_1}{v_1} = \left(\frac{1}{\sigma^2} yx_1 - \frac{1}{\sigma^2} x_1 w_2 x_2 \right) \iff m_1 = \frac{v_1}{\sigma^2} (yx_1 - x_1 w_2 x_2)$$

- *By symmetry*, we get $p(w_2|y, w_1) = \mathcal{N}(w_2|m_2, v_2)$

$$v_2 = \left(\frac{1}{\sigma^2} x_2^1 + \frac{1}{\kappa^2} \right)^{-1} \iff m_2 = \frac{v_2}{\sigma^2} (yx_2 - x_2 w_1 x_1)$$

Example: Gaussian linear model V

- Initialize w_1 and w_2 .

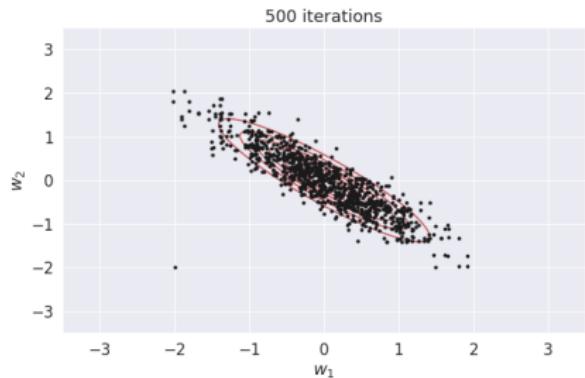
1. Sample w_1 conditioned w_2 ,

$$w_1 \sim p(w_1|y, w_2) = \mathcal{N}(w_1|m_1, v_1).$$

2. Sample w_2 conditioned w_1 ,

$$w_2 \sim p(w_2|y, w_1) = \mathcal{N}(w_2|m_2, v_2).$$

3. Repeat.



- Example shows typical *staircase behavior* of Gibbs samplers due sampling from the *posterior conditionals*

Why does Gibbs sampling work?

- The acceptance probability in Metropolis-Hastings algorithm:

$$A_k = \min \left[1, \frac{p(\mathbf{z}^*) q(\mathbf{z}^k | \mathbf{z}^*)}{p(\mathbf{z}^k) q(\mathbf{z}^* | \mathbf{z}^k)} \right]$$

- Writing $\mathbf{z}^k = \{z_i^k, \mathbf{z}_{-i}^k\}$, the proposal distribution for the Gibbs sampler is

$$q(z_i^* | \mathbf{z}^k) = p(z_i^* | \mathbf{z}_{-i}^k). \quad (1)$$

- We need the following fact

$$p(\mathbf{z}) = p(z_i, \mathbf{z}_{-i}) = p(z_i | \mathbf{z}_{-i}) p(\mathbf{z}_{-i}). \quad (2)$$

- Plugging in the proposal

$$A_k = \min \left[1, \frac{p(\mathbf{z}^*) p(z_i^k | \mathbf{z}_{-i}^*)}{p(\mathbf{z}^k) p(z_i^* | \mathbf{z}_{-i}^k)} \right] \quad (\text{Plugging in the proposal in eq. (1)})$$

$$= \min \left[1, \frac{p(z_i^* | \mathbf{z}_{-i}^*) p(\mathbf{z}_{-i}^k) p(z_i^k | \mathbf{z}_{-i}^*)}{p(\mathbf{z}_i^k | \mathbf{z}_{-i}^k) p(\mathbf{z}_{-i}^k) p(z_i^* | \mathbf{z}_{-i}^k)} \right] \quad (\text{Using eq. (2)})$$

$$= \min \left[1, \frac{p(z_i^* | \mathbf{z}_{-i}^k) p(\mathbf{z}_{-i}^k) p(z_i^k | \mathbf{z}_{-i}^k)}{p(\mathbf{z}_i^k | \mathbf{z}_{-i}^k) p(\mathbf{z}_{-i}^k) p(z_i^* | \mathbf{z}_{-i}^k)} \right] \quad (\text{Using } \mathbf{z}_{-i}^* = \mathbf{z}_{-i}^k)$$

$$= 1$$

- A Gibbs sampler is a special case of MH, where the proposed candidate is always accepted.

Questions: True or false?

Quiz via DTU Learn:

Lecture 9: Gibbs (5 questions)

Check your knowledge

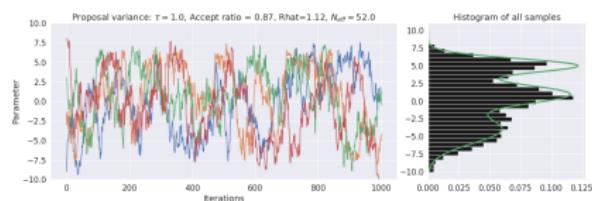
Convergence diagnostics

Are we there yet?

- MCMC theory states that samplers converge to the true target distribution as $S \rightarrow \infty$.
- Intuitive heuristic for assessing stationarity:** Run *multiple* chains from *different initial conditions*. After K iterations, we compare the distributions for each chain. If they are different, the chains have not yet reached the stationary distribution.
- Let B denote the *between-chain variance* and let W denote the *within-chain variance*, then \hat{R} -statistic (or the *potential scale reduction factor*) for chains of length N is defined by

$$\hat{R}^2 = \frac{S - 1}{S} + \frac{1}{S} \frac{B}{W}$$

- If $B = W$, then $\hat{R} = 1$. If $B > W$, then $\hat{R} > 1$. In practice, we say that the *chains have mixed* if $\hat{R} < 1.1$



For more details and motivation, see p. 284 in Bayesian Data Analysis (<http://www.stat.columbia.edu/~gelman/book/BDA3.pdf>)

How accurate is MCMC? Quantifying the error

- Recall for i.i.d. samples, the error of the MC estimator decreases with rate $1/\sqrt{S}$:

$$\mathbb{V}[\hat{f}] = \frac{1}{S} \mathbb{V}[f(\theta)]$$

- We can *estimate the variance* based on the samples and use this to *quantify the Monte Carlo error*. Let $\widehat{\text{sd}}(f(\theta))$ be the standard deviation of the MCMC samples,

$$\text{MCSE} = \frac{1}{\sqrt{S}} \widehat{\text{sd}}(f(\theta))$$

- MCMC methods produce *highly correlated* samples. The *autocorrelation function* ρ_t measures the correlation between two samples θ^i and θ^{i+t}

$$\rho_t = \frac{1}{\sigma^2} \int (\theta^i - \mu)(\theta^{i+t} - \mu) p(\theta) d\theta$$

- The *effective sample size (ESS)* S_{eff} takes this correlation into account

$$S_{\text{eff}} = \frac{S}{\sum_{t=-\infty}^{\infty} \rho_t} = \frac{S}{1 + 2 \sum_{t=1}^{\infty} \rho_t}$$

- and then

$$\text{MCSE} = \frac{1}{\sqrt{S_{\text{eff}}}} \widehat{\text{sd}}(f(\theta))$$

Example: MCMC diagnostics

- The \hat{R} – statistic

$$\hat{R}^2 = \frac{N - 1}{N} + \frac{1}{N} \frac{B}{W}$$

- The effective sample size

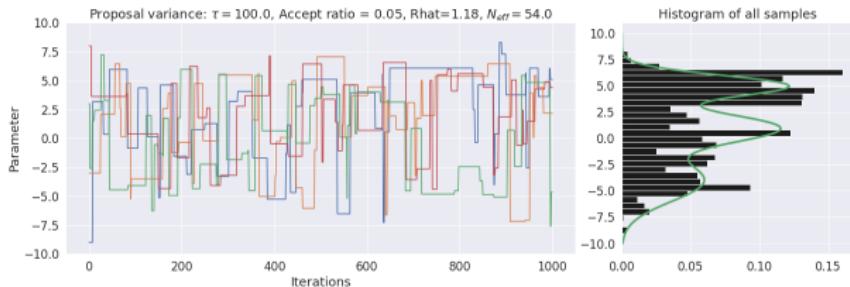
$$S_{\text{eff}} = \frac{S}{1 + 2 \sum_{t=1}^{\infty} \rho_t}$$

- The Monte Carlo Standard Error

$$\text{MCSE} = \frac{1}{\sqrt{S_{\text{eff}}}} \widehat{\text{sd}}(f(\theta))$$

- Next week

1. A few words on Hamiltonian Monte Carlo
2. We will do a short discussion on pros and cons of MCMC in practice
3. Start discussing variational inference.



Hierarchical models

Revisiting the Poisson regression

- "*Being Bayesian*" usually refers to treating quantities of interest as *random variable* and reason using the rules of *probability theory*
- When we talked about "*fully Bayesian*" inference, we refer the setting, where we have *prior distributions on all parameters, including hyperparameters*
- Before the holidays, we worked with a fully Bayesian Poisson regression model via MCMC

$$y_n | \mu_n \sim \text{Poisson}(\mu_n),$$

$$\mu_n = \exp(f_n)$$

$$f_n = \mathbf{w}^T \mathbf{x}_n$$

$$\mathbf{w} | \kappa \sim \mathcal{N}(0, \kappa^2 \mathbf{I})$$

$$\kappa \sim \mathcal{N}_+(0, 1)$$

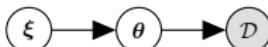
with the following joint distribution

$$p(\mathbf{y}, \mathbf{w}, \kappa) = \prod_{n=1}^N p(y_n | \mathbf{w}) p(\mathbf{w} | \kappa) p(\kappa),$$

- This is an example of a *hierarchical model*

Hierarchical modelling

- *Hierarchical* or *multi-level* models are one of the key strength of the Bayesian framework
- Suppose we have a model $p(\mathcal{D}|\theta)$ with data \mathcal{D} , parameters θ and hyperparameters ξ



with the following joint distribution

$$p(\mathcal{D}, \theta, \xi) \propto p(\mathcal{D}|\theta)p(\theta|\xi)p(\xi)$$

- Useful when you want to
 1. make inference more robust
 2. reason probabilistically about data, parameters and hyperparameters
 3. model hierarchical structure in data (random effects models)
 4. squeeze out every bit of predictive performance of a model/dataset
- Degrees of "Bayesianity" according to Murphy1

Method	Definition
Maximum likelihood	$\hat{\theta} = \arg \max_{\theta} p(\mathcal{D} \theta)$
MAP	$\hat{\theta} = \arg \max_{\theta} p(\mathcal{D} \theta)p(\theta \xi)$
ML-II	$\hat{\xi} = \arg \max_{\xi} \int p(\mathcal{D} \theta)p(\theta \xi)d\theta$
MAP-II	$\hat{\xi} = \arg \max_{\xi} \int p(\mathcal{D} \theta)p(\theta \xi)p(\xi)d\theta$
Full Bayes	$p(\theta, \xi \mathcal{D}) \propto p(\mathcal{D} \theta)p(\theta)p(\xi)$

Example

- Recall the model for Bayesian linear regression

$$p(\mathbf{w}|\alpha) = \mathcal{N}(\mathbf{w}|0, \alpha^{-1}\mathbf{I}) \quad (\textit{prior})$$

$$p(\mathbf{y}|\mathbf{w}, \beta) = \mathcal{N}(\mathbf{y}|\Phi\mathbf{w}, \beta^{-1}\mathbf{I}) \quad (\textit{likelihood})$$

$$p(\mathbf{w}|\mathbf{y}, \alpha, \beta) = \mathcal{N}(\mathbf{w}|\mathbf{m}, \mathbf{S}) \quad (\textit{posterior})$$

$$p(\mathbf{y}|\alpha, \beta) = \mathcal{N}(\mathbf{y}|\mathbf{0}, \beta^{-1}\mathbf{I} + \alpha^{-1}\Phi\Phi^T) \quad (\textit{marginal likelihood})$$

- Fully Bayesian inference on the full joint distribution

$$p(\alpha, \beta, \mathbf{w}|\mathbf{y}) \propto p(\mathbf{y}|\mathbf{w}, \beta)p(\mathbf{w}|\alpha)p(\alpha, \beta)$$

- Fully Bayesian inference on the marginalized joint distribution

$$p(\alpha, \beta|\mathbf{y}) \propto p(\mathbf{y}|\alpha, \beta)p(\alpha, \beta)$$

- Making predictions via MCMC

$$p(y^*|\mathbf{y}) = \mathbb{E}_{p(\alpha, \beta|\mathbf{y})} [p(y^*|\mathbf{y}, \alpha, \beta)] \approx \frac{1}{S} \sum_{i=1}^S p(y^*|\mathbf{y}, \alpha^{(i)}, \beta^{(i)})$$

for $\alpha^{(i)}, \beta^{(i)} \sim p(\alpha, \beta|\mathbf{y})$

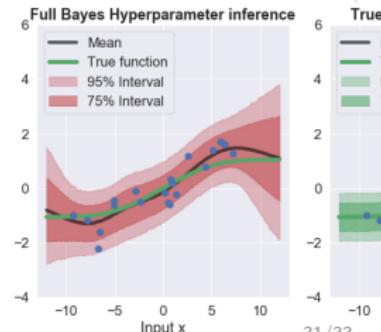
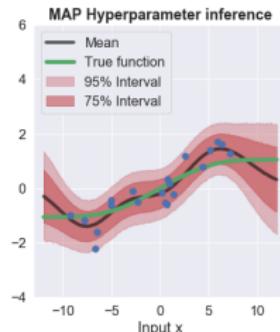
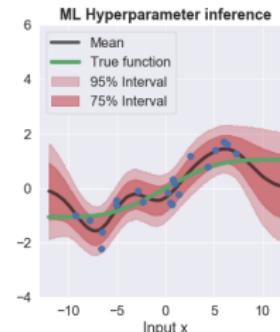
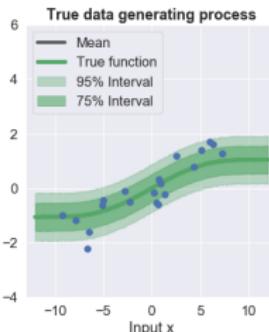
Fully Bayesian Gaussian process regression: Example

- Example with $N = 20$ data points and additive Gaussian noise
- Gaussian process regression with squared exponential kernel
- We impose a *weakly informative* prior on the lengthscale to
 1. rule out really short length scales (smaller than the grid size)
 2. rule out really long length scales (larger than span data of data)
- Joint distribution

$$p(\mathbf{y}, \mathbf{f}, \ell, \sigma, \kappa) = p(\mathbf{y}|\mathbf{f}, \sigma)p(\mathbf{f}|\ell, \kappa)p(\kappa)p(\ell)$$

- Marginalized joint distribution

$$p(\mathbf{y}, \ell, \sigma, \kappa) = p(\mathbf{y}|\sigma, \ell, \kappa)p(\kappa)p(\ell)$$



Medical example

- Suppose you work for a medical company and are asked to analyze data from a drug evaluation on rats prior to human trials
- Suppose the drug was administered to N rats, where y rats ended up developing tumors.
- We could use a Beta-Binomial model to estimate the probability of developing tumors θ

$$p(\theta|y) \propto \text{Bin}(y|N, \theta)\text{Beta}(\theta|\alpha_0, \beta_0)$$

- What if the company tested the drugs on J different types of rats, where (y_j, N_j) denote the data for the j 'th group. How to analyze the data?
 1. We could fit J *individual* models

$$p(\theta_j|y_j) \propto \text{Bin}(y_j|N_j, \theta_j)\text{Beta}(\theta_j|\alpha_0, \beta_0)$$

2. We could *pool* all the data such that $N_p = \sum_j N_j$ and $y_p = \sum_j y_j$

$$p(\theta_p|y_p) \propto \text{Bin}(y_p|N_p, \theta)\text{Beta}(\theta_p|\alpha_0, \beta_0)$$

- *Individual* models may perform poorly if N_j is small, but the *pooled* model might perform poorly if the different groups exhibit different behaviors
- *Bayesian hierarchical models* allows us to borrow statistical strength from groups with lots of data to help groups with less data

Medical example cont.

- Let $\mathbf{y} = \{y_j\}_{j=1}^J$ and $\boldsymbol{\theta} = \{\theta_j\}_{j=1}^J$, then

$$p(\mathbf{y}, \boldsymbol{\theta}, \alpha, \beta) = \prod_{j=1}^J \text{Bin}(y_j | N_j, \theta_j) \prod_{j=1}^J \text{Beta}(\theta_j | \alpha, \beta) p(\alpha, \beta)$$

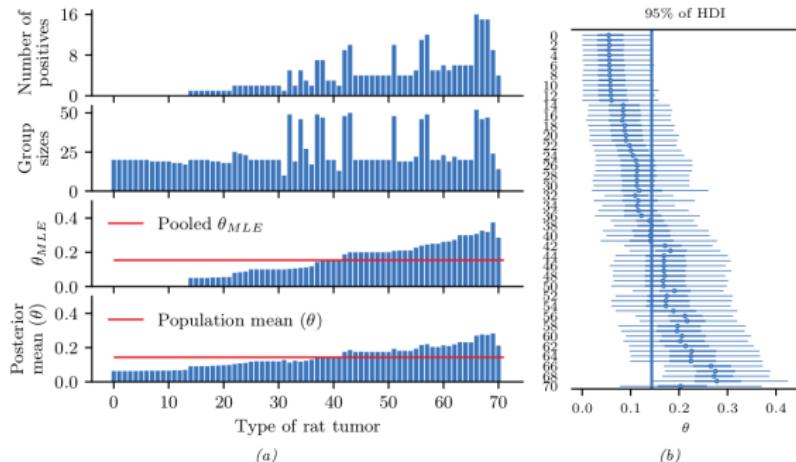


Figure 3.15: Data and inferences for the hierarchical binomial model fit using HMC. Generated by [hierarchical_binom_rats.ipynb](#).

The detailed balance condition I

- If a chain satisfies the *detailed balance condition*, then p^* is its stationary distribution

$$T(z'|z)p^*(z) = T(z|z')p^*(z')$$

- Integrating both sides wrt. z' yields the stationary distribution p^*

$$\int T(z'|z)p^*(z)dz' = p^*(z) = \int T(z|z')p^*(z')dz'$$

- Does the Metropolis-Hastings satisfy this condition? Let's check

- Recall the acceptance probability for Metropolis-Hastings

$$A(z^*|z^k) = \min \left[1, \frac{p(z^*)q(z^k|z^*)}{p(z^k)q(z^*|z^k)} \right]$$

- The transition kernel for Metropolis-Hastings

$$T(z'|z) = \begin{cases} q(z'|z)A(z'|z) & \text{if } z' \neq z \\ q(z|z)A(z|z) + \int q(z''|z) [1 - A(z''|z)] dz'' & \text{if } z' = z \end{cases}$$

The detailed balance condition II

- The acceptance probability

$$A(z^*, z^k) = \min \left[1, \frac{p(z^*)q(z^k|z^*)}{p(z^k)q(z^*|z^k)} \right]$$

- The transition kernel

$$T(z'|z) = \begin{cases} q(z'|z)A(z'|z) & \text{if } z' \neq z \\ q(z|z)A(z|z) + \int q(z''|z) [1 - A(z''|z)] dz'' & \text{if } z' = z \end{cases}$$

- If $p(z^k)q(z^*|z^k) > p(z^*)q(z^k|z^*)$, then $A(z^*|z^k) < 1$ and $A(z^k|z^*) = 1$

- To jump from z^k to z^* , we first need to propose it and then accept it

$$T(z^*|z^k) = q(z^*|z^k)A(z^*|z^k) = q(z^*|z^k) \frac{p(z^*)q(z^k|z^*)}{p(z^k)q(z^*|z^k)} = \frac{p(z^*)q(z^k|z^*)}{p(z^k)}$$

- It follows

$$p(z^k)T(z^*|z) = p(z^*)q(z^k|z^*)$$

The detailed balance condition III

- We just arrived at

$$p(z^k)T(z^*|z^k) = p(z^*)q(z^k|z^*)$$

- What about the opposite direction?

$$T(z^k|z^*) = q(z^k|z^*)A(z^k|z^*) = q(z^k|z^*)$$

- Combining the two and we are done

$$p(z^k)T(z^*|z^k) = p(z^*)T(z^k|z^*)$$

02477 – Bayesian Machine Learning: Lecture 10

Michael Riis Andersen

Technical University of Denmark,
DTU Compute, Department of Applied Math and Computer Science

Outline

1 Wrap up: Monte Carlo and sampling methods

2 Variational inference

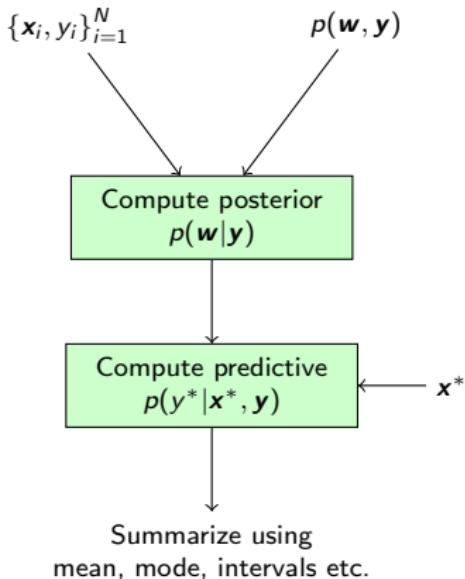
- Basic concepts
- Factorized approximations and CAVI

3 Mixture models

Wrap up: Monte Carlo and sampling methods

Probabilistic machine learning

Probabilistic machine learning = data + model + inference algorithm



Distributions as Lego blocks: Beta, Binomial, Gaussians, Gamma, Poisson, Categorical, Gaussian processes, Uniform, Dirichlet,

Overview of sampling methods

- Ancestral sampling
 - Sampling from joint distributions $p(x, z, w) = p(x|z)p(z|w)p(w)$ or marginals, e.g. $p(x)$, if we can sample from all the conditional distributions
 - Useful for sampling-based posterior inference, e.g. $p(x, z, w|\mathcal{D}) = p(x|z)p(z|w)p(w|\mathcal{D})$
- Gibbs Sampling (MCMC)
 - Usually the first choice for conditionally conjugate models
 - No tuning parameters, acceptance ratio is always 1, but can be very slow for highly correlated distributions
 - Requires model-specific derivations
- Metropolis-Hastings (MCMC)
 - Does not require conjugacy
 - Very flexible
 - Often requires tuning to work well
- Other techniques: Rejection sampling, importance sampling
- Hamiltonian Monte Carlo (MCMC)
 - MH-algorithm that uses Hamiltonian dynamics based on gradient information to generate new proposals
 - Much faster than 'simple' MH
 - Only applies to continuous parameters, where gradients are available
- Probabilistic programming tools supporting HMC/MCMC
 - Stan, PyMC3, Tensorflow Probability, Pyro, BlackJax

The HMC algorithm in a nutshell

- Goal: Generating samples from target distribution $p(\theta) = \frac{1}{Z} \tilde{p}(\theta)$
- We augment the target distribution: $p(\theta, \nu) \propto \tilde{p}(\theta) \mathcal{N}(\nu | \mathbf{0}, \Sigma)$
- Recipe for generating the $k + 1$ 'the sample using HMC
 1. Initialize $\theta'_0 = \theta_k$ and $\nu'_0 \sim \mathcal{N}(\nu | \mathbf{0}, \Sigma)$
 2. For $\ell = 1, \dots, L$

$$\begin{aligned}\nu'_\ell &= \nu'_{\ell-1} + \eta \nabla \log \tilde{p}(\theta_{\ell-1}) \\ \theta'_\ell &= \theta'_{\ell-1} + \eta \Sigma^{-1} \nu'_\ell\end{aligned}$$

3. Set $\theta^* = \theta'_L$ and $\nu^* = \nu'_L$
4. Compute acceptance probability

$$A_{k+1} = \min \left(1, \frac{p(\theta^*, \nu^*)}{p(\theta_k, \nu_k)} \right)$$

5. Accept proposal θ^* with probability A_{k+1} , keep θ_{k-1} otherwise
- Step-size η , covariance Σ , and the number of integration steps L are parameters of the algorithm.
 - Gradient information allows HMC to explore the target distribution much more efficiently
 - HMC NUTS (No U-turn Sampler) is state-of-the-art
 - Cool visualization of HMC: <https://chi-feng.github.io/mcmc-demo/app.html>

Inference methods

- Maximum likelihood
 - 1. Fast and often easy
 - 2. Prone to overfitting, no model uncertainty, not necessarily well-defined
- Exact Bayesian inference
 - 1. Extremely limited in choice of models (linear models, conjugate models etc)
 - 2. Very fast
- Laplace approximations
 - 1. Very simple to implement and very fast
 - 2. Limited to continuous distributions
 - 3. Works well when the exact posterior is close to Gaussian
 - 4. Can fail horribly for asymmetrical and skewed distributions
- Markov Chain Monte Carlo (MCMC)
 - 1. Very strong mathematical guarantees, asymptotically exact and Very flexible
 - 2. Might take forever to converge (literally)
 - 3. First choice when we have smaller or moderate sized datasets and/or when accuracy and/or uncertainty are prioritized
- Variational inference (VI)
 - 1. Applies to both continuous and discrete distributions
 - 2. Can be much faster than MCMC, but without any strict guarantees
 - 3. Control accuracy vs speed trade-off
 - 4. Useful for testing different models for very large data sets
 - 5. Foundation many Bayesian deep learning methods and Variational autoencoders (VAEs)

Variational inference

Variational inference: Basic concepts

Variational inference: big picture

- Our goal is to approximate a posterior distribution of interest

$$p \equiv p(\mathbf{z}|\mathcal{D}) = \frac{p(\mathcal{D}|\mathbf{z})p(\mathbf{z})}{p(\mathcal{D})}$$

- Variational inference in three steps

1. Define collection of "simple" approximate probability distributions \mathcal{Q} (*the variational family*)
 2. Define a measure of "distance" between probability distributions $\mathbb{D}[q||p]$ (*the divergence*)
 3. Search for the distribution $q \in \mathcal{Q}$ that resembles the exact posterior p as close as possible as measured by $\mathbb{D}[q||p]$ (*optimization*)
-
- The variational approximation q for the target distribution $p \approx q$ is defined as

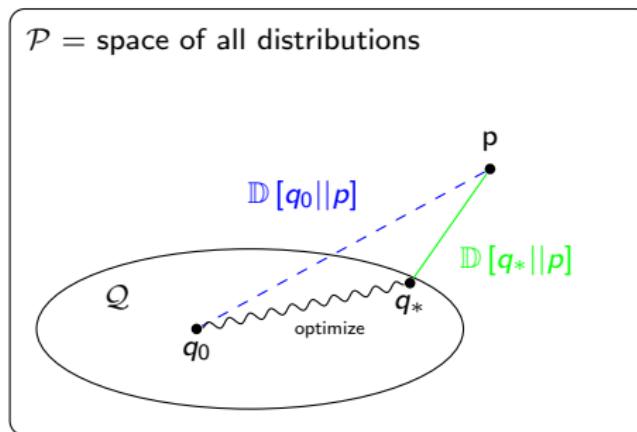
$$q_* = \arg \min_{q \in \mathcal{Q}} \mathbb{D}[q||p]$$

Variational inference: big picture

- The variational approximation q for target distribution $p \approx q$ is defined as

$$q_* = \arg \min_{q \in \mathcal{Q}} \mathbb{D}[q || p]$$

where \mathcal{Q} is a collection of "simple" approximate probability distributions



Example (Bishop p. 464)

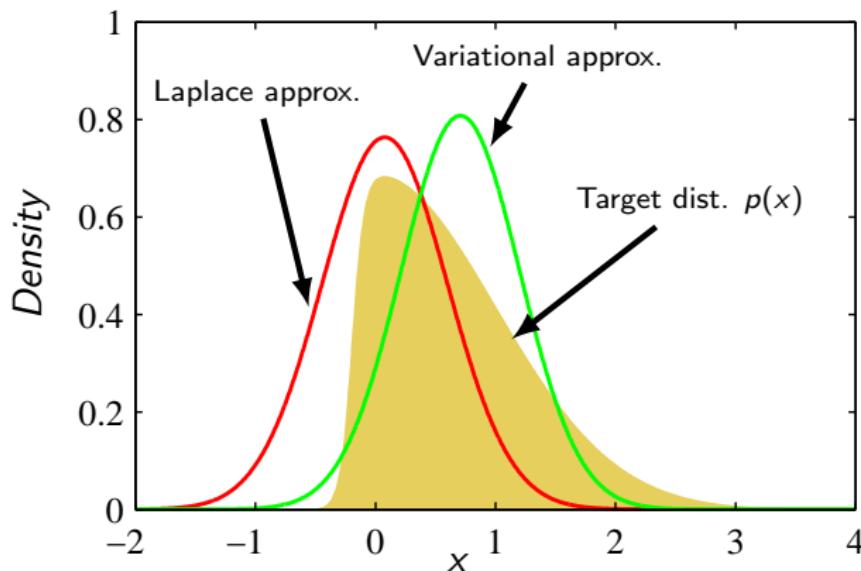
Goal: approximate some target distribution of interest $p(x)$ for $x \in \mathbb{R}$ (yellow).

1. We choose the variational family to be all the Gaussian densities

$$\mathcal{Q} = \{\mathcal{N}(\mu, \sigma^2) \mid \mu \in \mathbb{R}, \sigma^2 > 0\}$$

2. We choose some divergence measure $\mathbb{D}[q||p]$ and minimize it wrt. q

$$q_* = \arg \min_{q \in \mathcal{Q}} \mathbb{D}[q||p]$$



The variational family \mathcal{Q}

- The *variational family* \mathcal{Q} defines the collection of all possible approximations $q \in \mathcal{Q}$
- \mathcal{Q} is chosen as a compromise between speed/tractability and approximation quality. The larger \mathcal{Q} , the smaller approximation error and vice versa.
- Examples of common variational families for $\mathbf{z} = [z_1, z_2, \dots, z_D] \in \mathbb{R}^D$
 - *Full-rank Gaussians*

$$q(\mathbf{z}) = \mathcal{N}(\mathbf{z} | \mathbf{m}, \mathbf{V})$$

- *Mean-field Gaussians*

$$q(\mathbf{z}) = \prod_{i=1}^D \mathcal{N}(z_i | m_i, v_i)$$

- *Mean-field approximations*

$$q(\mathbf{z}) = \prod_{i=1}^D q(z_i)$$

Example: $\mathbf{z} \in \mathbb{R}^5$: $q(z_1, z_2, z_3, z_4, z_5) = q(z_1)q(z_2)q(z_3)q(z_4)q(z_5)$

- *Factorized approximations*

$$q(\mathbf{z}) = \prod_{j=1}^J q(z_j), \quad \text{where } \mathbf{z} = [z_1, z_2, \dots, z_J] \text{ for } J < D$$

Example: $\mathbf{z} \in \mathbb{R}^5$: $q(z_1, z_2, z_3, z_4, z_5) = q(z_1, z_2)q(z_3)q(z_4, z_5)$

Measuring distance between probability distributions

- How do we measure "distance" between probability distributions $\mathbb{D}[q, p]$?
- The choice of "distance" affects the properties of the approximation
- The *Kullback-Leibler* divergence (for continuous R.V.) is defined as

$$\text{KL}[q||p] = \int q(z) \ln \left[\frac{q(z)}{p(z)} \right] dz$$

- Properties

1. Identity of indiscernibles

$$\text{KL}[q||p] = 0 \iff p = q \text{ (a.e)}$$

2. Non-negativity

$$\text{KL}[q||p] \geq 0$$

3. Asymmetric

$$\text{KL}[q||p] \neq \text{KL}[p||q]$$

4. Does not satisfy the triangle inequality, i.e. $\text{KL}[q||p] \leq \text{KL}[q||r] + \text{KL}[r||p]$ does **not** hold in general.

Quiz

Quiz time!

Week 10: Variational inference

Minimizing the KL-divergence

The *variational approximation* q for target distribution $p(\mathbf{z}|\mathcal{D}) \approx q$ is defined as

$$q_* = \arg \min_{q \in \mathcal{Q}} \text{KL}[q||p], \quad \text{KL}[q||p] = \int q(\mathbf{z}) \ln \left[\frac{q(\mathbf{z})}{p(\mathbf{z})} \right] d\mathbf{z}$$

- Re-writing the KL-divergence for target posterior distribution $p \equiv p(\mathbf{z}|\mathcal{D})$

$$\begin{aligned} \text{KL}[q||p] &= \int q(\mathbf{z}) \ln \left(\frac{q(\mathbf{z})}{p(\mathbf{z}|\mathcal{D})} \right) d\mathbf{z} && \text{(Definition of KL)} \\ &= \mathbb{E}_q \left[\ln \frac{q(\mathbf{z})}{p(\mathbf{z}|\mathcal{D})} \right] && \text{(Def. of expectation)} \\ &= \mathbb{E}_q [\ln q(\mathbf{z})] - \mathbb{E}_q [\ln p(\mathbf{z}|\mathcal{D})] && \text{(Linearity of expectations)} \\ &= \mathbb{E}_q [\ln q(\mathbf{z})] - \mathbb{E}_q \left[\ln \frac{p(\mathcal{D}, \mathbf{z})}{p(\mathcal{D})} \right] && \text{(Using Bayes' theorem)} \\ &= \mathbb{E}_q [\ln q(\mathbf{z})] - \mathbb{E}_q [\ln p(\mathcal{D}, \mathbf{z})] + \mathbb{E}_q [\ln p(\mathcal{D})] && \text{(Linearity of expectations)} \\ &= \mathbb{E}_q [\ln q(\mathbf{z})] - \mathbb{E}_q [\ln p(\mathcal{D}, \mathbf{z})] + \ln p(\mathcal{D}) && (\mathcal{D} \text{ is independent of } q) \end{aligned}$$

Re-arranging

$$\ln p(\mathcal{D}) = \mathbb{E}_q [\ln p(\mathcal{D}, \mathbf{z})] - \mathbb{E}_q [\ln q(\mathbf{z})] + \text{KL}[q||p]$$

The evidence lower bound

- We just derived

$$\begin{aligned}\ln p(\mathcal{D}) &= \mathbb{E}_q [\ln p(\mathcal{D}, \mathbf{z})] - \mathbb{E}_q [\ln q(\mathbf{z})] + \text{KL}[q||p] \\ &= \mathcal{L}[q] + \text{KL}[q||p] \\ &\geq \mathcal{L}[q]\end{aligned}$$

- We define the *evidence lower bound* (ELBO) as

$$\mathcal{L}[q] \equiv \mathbb{E}_q [\ln p(\mathcal{D}, \mathbf{z})] - \mathbb{E}_q [\ln q(\mathbf{z})]$$

- Observations

1. Maximizing \mathcal{L} is equivalent to minimizing $\text{KL}[q||p]$ since $\text{KL}[q||p] \geq 0$ and $\ln p(\mathcal{D})$ is const.
2. We only need to be able to evaluate log joint distribution $p(\mathbf{z}, \mathcal{D})$, not the posterior $p(\mathbf{z}|\mathcal{D})$
3. The ELBO $\mathcal{L}[q]$ is a lower bound on the marginal likelihood, i.e.

$$\mathcal{L}[q] \leq \ln p(\mathcal{D})$$

- Key take away

$$q^* = \arg \min_{q \in \mathcal{Q}} \text{KL}[q||p] = \arg \max_{q \in \mathcal{Q}} \mathcal{L}[q]$$

Variational inference: Factorized approximations and CAVI

Minimizing the KL-divergence for factorized distributions I

- Factorized approximations: $q(\mathbf{z}) = \prod_{j=1}^J q(z_j)$ where $\mathbf{z} = [z_1, z_2, \dots, z_J]$

Example: If $\mathbf{z} = [z_1, z_2, z_3, z_4, z_5]$, then we may assume $q(\mathbf{z}) = q(z_1)q(z_2, z_3)q(z_4, z_5)$ ($J = 3$)

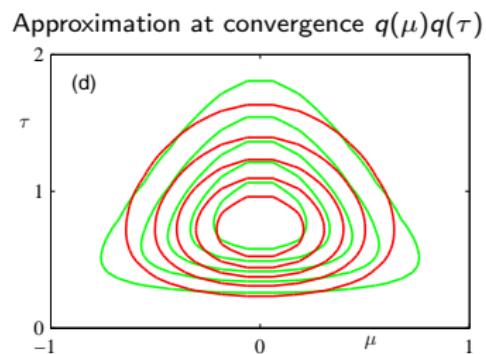
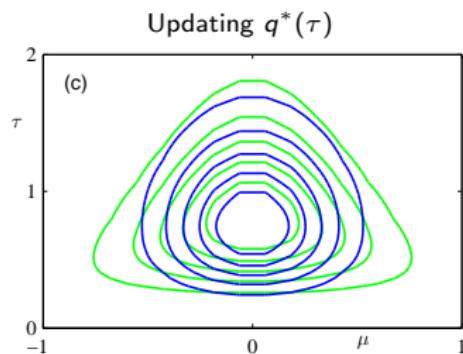
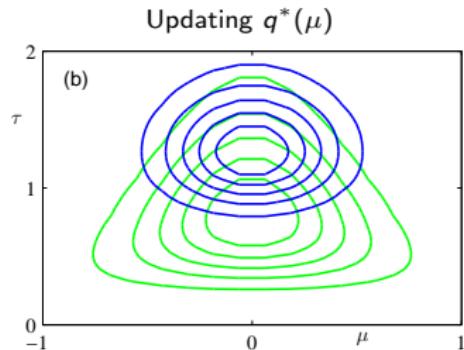
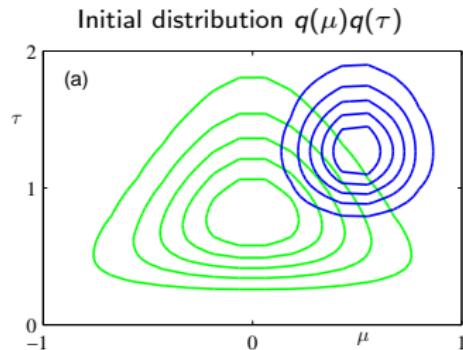
- The factorization can be in groups of variables or across all variables (mean-field)
- We make *no assumptions on the functional form for each factor*. Hence, often called *free-form* variational inference
- How to identify $q(z_j)$ for $j = 1, \dots, J$ for a given posterior?
- Minimize KL by substituting the approximation into the ELBO

$$\mathcal{L}[q] \equiv \mathbb{E}_q [\ln p(\mathcal{D}, \mathbf{z})] - \mathbb{E}_q [\ln q(\mathbf{z})] = \int \prod_{j=1}^J q(z_j) \ln p(\mathcal{D}, \mathbf{z}) d\mathbf{z} - \int \prod_{j=1}^J q(z_j) \ln \prod_{j=1}^J q(z_j) d\mathbf{z}$$

- Optimization strategy: *Coordinate ascent variational inference* (CAVI)
 - We iterate through all factors, updating one at a time. Starting with the k 'th factor
 - We'll identify all terms that depend on $q(z_k)$ and use that to optimize \mathcal{L} .
 - Repeat for all k and iterate until convergence

Minimizing the KL-divergence for factorized distributions I: Example

Target posterior distribution: $p(\mu, \tau | \mathcal{D})$ (Example from Section 10.1.3 in Bishop)



Minimizing the KL-divergence for factorized distributions IIa

We want to maximimize \mathcal{L} wrt. $q(\mathbf{z}_k)$

$$\mathcal{L}[q] = \int \prod_{i=1}^J q(\mathbf{z}_i) \ln p(\mathcal{D}, \mathbf{z}) d\mathbf{z} - \int \prod_{i=1}^J q(\mathbf{z}_i) \ln \prod_{j=1}^J q(\mathbf{z}_j) d\mathbf{z}$$

Identifying part of the second term that depends on $q(\mathbf{z}_k)$

$$\begin{aligned} \int \prod_{i=1}^J q(\mathbf{z}_i) \ln \prod_{j=1}^J q(\mathbf{z}_j) d\mathbf{z} &= \int \prod_{j=1}^J q(\mathbf{z}_j) \sum_{j=1}^J \ln q(\mathbf{z}_j) d\mathbf{z} && \text{(From product to sums)} \\ &= \sum_{i=1}^J \int \prod_{i=1}^J q(\mathbf{z}_i) \ln q(\mathbf{z}_j) d\mathbf{z} && \text{(Linearity of integrals)} \\ &= \sum_{j=1}^J \int q(\mathbf{z}_1) q(\mathbf{z}_2) \dots q(\mathbf{z}_J) \ln q(\mathbf{z}_j) d\mathbf{z} && \text{(Expand product)} \\ &= \sum_{j=1}^J \int q(\mathbf{z}_j) \ln q(\mathbf{z}_j) d\mathbf{z}_j && \text{(Marginalize)} \\ &= \int q(\mathbf{z}_k) \ln q(\mathbf{z}_k) d\mathbf{z}_k + \text{const} && \text{(Dependency on } q(\mathbf{z}_k)) \end{aligned}$$

Therefore, we can write

$$\mathcal{L}[q] = \int \prod_{j=1}^J q(\mathbf{z}_j) \ln p(\mathcal{D}, \mathbf{z}) d\mathbf{z} - \int q(\mathbf{z}_k) \ln q(\mathbf{z}_k) d\mathbf{z}_k + \text{const}$$

Minimizing the KL-divergence for factorized distributions IIb

Simplifying the first term

$$\begin{aligned}\mathcal{L}[q] &= \int \prod_{i=1}^J q(\mathbf{z}_i) \ln p(\mathcal{D}, \mathbf{z}) d\mathbf{z} - \int q(\mathbf{z}_k) \ln q(\mathbf{z}_k) d\mathbf{z}_k + \text{const} \\ &= \int q(\mathbf{z}_1)q(\mathbf{z}_2)\dots q(\mathbf{z}_J) \ln p(\mathcal{D}, \mathbf{z}) d\mathbf{z} - \int q(\mathbf{z}_k) \ln q(\mathbf{z}_k) d\mathbf{z}_k + \text{const} \quad (\text{expand product}) \\ &= \int q(\mathbf{z}_k) \left[\int \prod_{i \neq k} q(\mathbf{z}_i) \ln p(\mathcal{D}, \mathbf{z}) d\mathbf{z}_{-k} \right] d\mathbf{z}_k - \int q(\mathbf{z}_k) \ln q(\mathbf{z}_k) d\mathbf{z}_k + \text{const} \quad (\text{Factor out } q(\mathbf{z}_k)) \\ &= \int q(\mathbf{z}_k) \underbrace{\mathbb{E}_{i \neq k} [\ln p(\mathcal{D}, \mathbf{z})]}_{\ln \tilde{p}(\mathcal{D}, \mathbf{z}_k)} d\mathbf{z}_k - \int q(\mathbf{z}_k) \ln q(\mathbf{z}_k) d\mathbf{z}_k + \text{const} \quad (\text{Define } \tilde{p}(\mathcal{D}, \mathbf{z}_k)) \\ &= \int q(\mathbf{z}_k) \ln \tilde{p}(\mathcal{D}, \mathbf{z}_k) d\mathbf{z}_k - \int q(\mathbf{z}_k) \ln q(\mathbf{z}_k) d\mathbf{z}_k + \text{const} \quad (\text{Use def. of } \tilde{p}) \\ &= \int q(\mathbf{z}_k) \ln \frac{\tilde{p}(\mathcal{D}, \mathbf{z}_k)}{q(\mathbf{z}_k)} d\mathbf{z}_k + \text{const} \quad (\text{Linearity of integrals}) \\ &= -\text{KL}[q(\mathbf{z}_k) || \tilde{p}(\mathcal{D}, \mathbf{z}_k)] + \text{const} \quad (\text{Def. of KL})\end{aligned}$$

- Summary: When we consider the ELBO as a function of $q(\mathbf{z}_k)$ only, it is equal to the KL-divergence between $q(\mathbf{z}_k)$ and $\tilde{p}(\mathcal{D}, \mathbf{z}_k)$. When are KL-divergences minimized?

Minimizing the KL-divergence for factorized distributions III

- Goal: We want to minimize the KL-divergence $\text{KL}[q||p]$ between our approximation q and our target p by maximizing the ELBO \mathcal{L} iterative one factor $q(\mathbf{z}_k)$ at time
- We just showed that when we consider \mathcal{L} as a function of $q(\mathbf{z}_k)$, then

$$\mathcal{L}[q] = -\text{KL}[q(\mathbf{z}_k)||\tilde{p}(\mathcal{D}, \mathbf{z}_k)] + k$$

- The KL divergence is minimized wrt. \mathbf{z}_k when $q(\mathbf{z}_k) = \tilde{p}(\mathcal{D}, \mathbf{z}_k)$.
- The *optimal choice* for the factor $q(\mathbf{z}_k)$ is

$$\ln q^*(\mathbf{z}_k) = \ln \tilde{p}(\mathcal{D}, \mathbf{z}_k) = \mathbb{E}_{i \neq k} [\ln p(\mathcal{D}, \mathbf{z})] + K$$

- In words: The optimal distribution for $\ln q^*(\mathbf{z}_k)$ is obtained by taking the log joint distribution $\ln p(\mathcal{D}, \mathbf{z})$ and averaged it wrt. all the other factors, i.e. $q(\mathbf{z}_j)$ for $j \neq k$

Coordinate Ascent Variational Inference (CAVI)

Big picture

- We are given a joint distribution for a dataset \mathcal{D} and parameters $\mathbf{w} \in \mathbb{R}^D$

$$p(\mathcal{D}, \mathbf{w}) = p(\mathcal{D}|\mathbf{w})p(\mathbf{w})$$

- The *variational approximation* q for target distribution s.t. $p \equiv p(\mathbf{w}|\mathcal{D}) \approx q$ is defined as

$$q_* = \arg \min_{q \in \mathcal{Q}} \text{KL}[q||p]$$

- Factorized approximation

$$q(\mathbf{w}) = \prod_{j=1}^J q(\mathbf{w}_j), \quad \mathbf{w} = [\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_J]$$

- CAVI algorithm: Repeat until convergence (or fixed number of iterations)

- For $k = 1, \dots, K$

$$\ln q^*(\mathbf{w}_k) = \mathbb{E}_{\prod_{i \neq k} q(\mathbf{w}_i)} [\ln p(\mathcal{D}, \mathbf{w})] + K$$

- Compute ELBO $\mathcal{L}[q]$ (monitoring convergence, model selection)

- Close parallel to Gibbs sampling except we now compute the expectation wrt. $\prod_{i \neq k} q(\mathbf{w}_i)$ ("all other parameters")

CAVI Example

- Suppose we are working with a model with two parameters $\mathbf{w} \in \mathbb{R}^2$
- The joint distribution of the model is given by

$$\ln p(\mathbf{y}, \mathbf{w}) = \log p(\mathbf{y} | \mathbf{w}) + p(\mathbf{w}) = -w_1^2 - \frac{1}{2}w_2^2 + w_1 w_2 + 6w_1 - 3w_2$$

- We want to approximate the resulting posterior using a factorized distribution

$$q(\mathbf{w}) = q(w_1)q(w_2)$$

- The general CAVI update rule states that

$$\ln q^*(w_k) = \mathbb{E}_{\prod_{i \neq k} q(w_i)} [\ln p(\mathbf{y}, \mathbf{w})] + K$$

- For this example

$$\begin{aligned}\ln q^*(w_1) &= \mathbb{E}_{q(w_2)} [\ln p(\mathbf{w}, \mathbf{y})] + K \\ \ln q^*(w_2) &= \mathbb{E}_{q(w_1)} [\ln p(\mathbf{w}, \mathbf{y})] + K\end{aligned}$$

CAVI Example continued I

The optimal solution for $q(w_1)$ is given by

$$\begin{aligned}\ln q(w_1) &= \mathbb{E}_{q(w_2)} [\ln p(\mathbf{w}, \mathbf{y})] + K \\&= \mathbb{E}_{q(w_2)} \left[-w_1^2 - \frac{1}{2} w_2^2 + w_1 w_2 + 6w_1 - 3w_2 \right] + K \\&= -w_1^2 - \frac{1}{2} \mathbb{E}_{q(w_2)} [w_2^2] + w_1 \mathbb{E}_{q(w_2)} [w_2] + 6w_1 - 3\mathbb{E}_{q(w_2)} [w_2] + K \\&= -w_1^2 + w_1 \mathbb{E}_{q(w_2)} [w_2] + 6w_1 + K' \\&= -w_1^2 + w_1 (6 + \mathbb{E}_{q(w_2)} [w_2]) + K'\end{aligned}$$

We recognize the above as a second order polynomial in w_1 . Therefore, we conclude that $q(w_1)$ must be a Gaussian distribution and we can identify its mean and variance by matching the coefficients for the first and secord order term as follows

$$v_1^{-1} = 2 \iff v_1 = \frac{1}{2} \quad \frac{m_1}{v_2} = 6 + \mathbb{E}_{q(w_2)} [w_2] \iff m_1 = 3 + \frac{1}{2} \mathbb{E}_{q(w_2)} [w_2]$$

CAVI Example continued II

The optimal solution for $q(w_2)$ is given by

$$\begin{aligned}\ln q(w_2) &= \mathbb{E}_{q(w_1)} [\ln p(\mathbf{w}, \mathbf{y})] + K \\&= \mathbb{E}_{q(w_1)} \left[-w_1^2 - \frac{1}{2} w_2^2 + w_1 w_2 + 6w_1 - 3w_2 \right] + K \\&= -\mathbb{E}_{q(w_1)} [w_1^2] - \frac{1}{2} \mathbb{E}_{q(w_1)} [w_2^2] + \mathbb{E}_{q(w_1)} [w_1] w_2 + 6\mathbb{E}_{q(w_1)} [w_1] - 3w_2 + K \\&= -\frac{1}{2} w_2^2 + \mathbb{E}_{q(w_1)} [w_1] w_2 - 3w_2 + K' \\&= -\frac{1}{2} w_2^2 + w_2 (\mathbb{E}_{q(w_1)} [w_1] - 3) + K'\end{aligned}$$

Again, this is a second order polynomial in w_2 and therefore $q(w_2)$ must be Gaussian with parameters

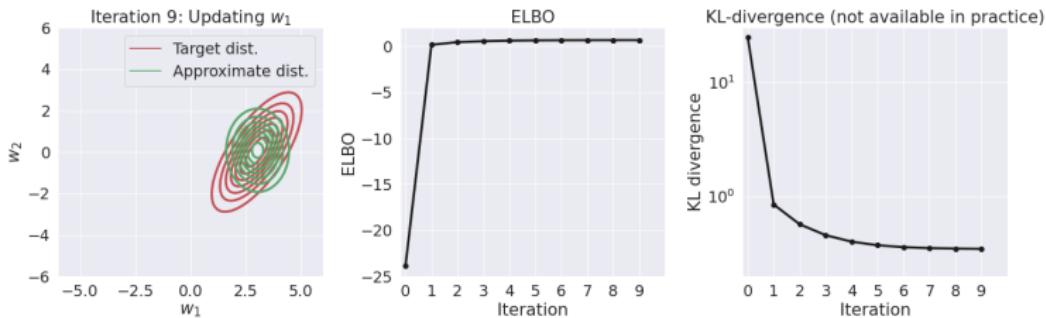
$$v_2^{-1} = 1 \iff v_2 = 1 \quad \frac{m_2}{v_2} = \mathbb{E}_{q(w_1)} [w_1] - 3 \iff m_2 = \mathbb{E}_{q(w_1)} [w_1] - 3$$

CAVI Example continued III

- Initialize variational parameters and iteratively use update equations

$$q(w_1) = \mathcal{N}(w_1 | 3 + \frac{1}{2}m_2, \frac{1}{2})$$

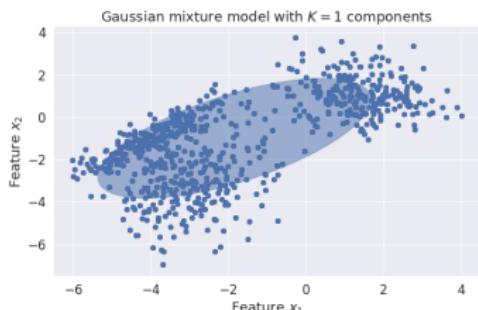
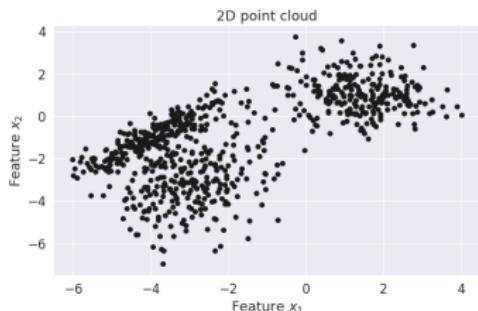
$$q(w_2) = \mathcal{N}(w_2 | m_1 - 3, 1)$$



Mixture models

Unsupervised learning

- Clustering and density estimation as examples of *unsupervised learning*
- Dataset $\mathcal{D} = \{x_1, x_2, \dots, x_N\}$
 - Input features: $x_i \in \mathbb{R}^D$
- Can we divide the dataset into K groups?
- *Model selection:* How to choose K ?
- Common steps
 1. Choose model for the data
 2. Infer parameters of model θ
 3. Use parameters to make predictions for new data, e.g. outlier detection
- Applications
 - Clustering (news articles, songs, ...)
 - Fraud detection
 - ...



The Gaussian Mixture Model

- How to model non-Gaussian data?
- We can construct arbitrary complex distribution by

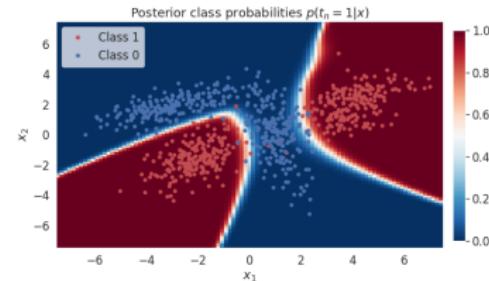
$$p(\mathbf{x}) = \sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

- The *mixing weights* $0 \leq \pi_k \leq 1$ and

$$\sum_{k=1}^K \pi_k = 1$$

- Example: Generative classification

$$p(y_n = k | \mathbf{x}_n) = \frac{p(\mathbf{x}_n | y_n = k)p(y_n = k)}{p(\mathbf{x}_n)}$$



Fitting Gaussian Mixtures using Maximum likelihood

Expectation-maximization algorithm: Maximum likelihood estimation for Gaussian mixture models

$$p(\mathbf{x}) = \sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

1. Initialize all parameters: $\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k$ for $k = 1, \dots, K$
2. Repeat until convergence

- Expectation-step:

$$\gamma_{nk} = \frac{\pi_k \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^K \pi_j \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}$$

- Maximization-step:

$$N_k = \sum_{n=1}^N \gamma_{nk}$$

$$\pi_k^* = \frac{N_k}{N}$$

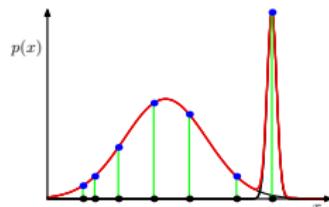
$$\boldsymbol{\mu}_k^* = \frac{1}{N_k} \sum_{n=1}^N \gamma_{nk} \mathbf{x}_n$$

$$\boldsymbol{\Sigma}_k^* = \frac{1}{N_k} \sum_{n=1}^N \gamma_{nk} (\mathbf{x}_n - \boldsymbol{\mu}_k^*) (\mathbf{x}_n - \boldsymbol{\mu}_k^*)^T$$

Problems with EM

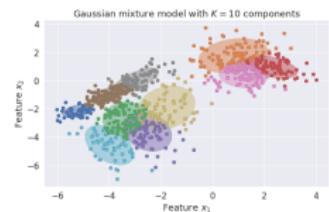
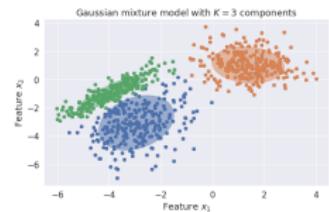
Several issues with the EM algorithm

1. Components can "collapse" onto single data points causing the maximum likelihood to diverge (overfitting due to maximum likelihood)
2. How to determine the number of clusters?
3. Sensitive to initialization



Bayesian approach

1. We can remove problem 1 entirely
2. Problem 2 is non-trivial, but Bayesian methods do have something to offer
3. The variational approximation we will study is also sensitive to initialization



Bayesian Gaussian Mixture Model I

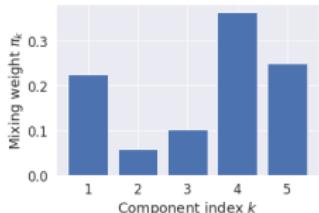
- We follow Bishop and switch to *precision matrix* parametrization $\Lambda_k = \Sigma^{-1}$. The Gaussian Mixture Model (GMM) becomes

$$p(\mathbf{x}_n) = \sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k^{-1})$$

- Introducing *binary one-hot encoded latent variables \mathbf{z}*

$$p(\mathbf{x}_n | \mathbf{z}_n) = \prod_{k=1}^K \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k^{-1})^{z_{nk}}$$

$$p(\mathbf{z}_n) = \text{Categorical}(\mathbf{z}_n | \boldsymbol{\pi}) = \prod_{k=1}^K \pi_k^{z_n}$$



- Example: suppose $K = 5$ and observation n belongs to the 4th cluster

$$\mathbf{z}_n = [0 \quad 0 \quad 0 \quad 1 \quad 0]$$

- We can always go back to the original model via the sum rule

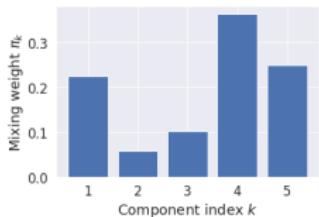
$$p(\mathbf{x}_n) = \sum_{k=1}^K p(\mathbf{x}_n | z_n = k) p(z_n = k)$$

Bayesian Gaussian Mixture Model II

$$p(\mathbf{x}_n | \mathbf{z}_n) = \prod_{k=1}^K \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k^{-1})^{z_{nk}}$$

$$p(\mathbf{z}_n) = \text{Categorical}(\mathbf{z} | \boldsymbol{\pi}) = \prod_{k=1}^K \pi_k^{z_n}$$

- *Latent variables*: \mathbf{z}_n are variable we cannot observe directly
- We need priors for $\boldsymbol{\pi}$, $\boldsymbol{\mu}$, $\boldsymbol{\Sigma}$ to complete the model



$$\begin{aligned}\boldsymbol{\pi} &\sim \text{Dirichlet}(\boldsymbol{\alpha}_0) \\ \boldsymbol{\Lambda}_k &\sim \text{Wishart}(\mathbf{W}_0, \nu_0) \\ \boldsymbol{\mu}_k | \boldsymbol{\Lambda}_k &\sim \text{Normal}(\mathbf{m}_0, (\beta_0 \boldsymbol{\Lambda}_k)^{-1}) \\ \mathbf{z}_n | \boldsymbol{\pi} &\sim \text{Categorical}(\boldsymbol{\pi}) \\ \mathbf{x}_n | \boldsymbol{\mu}, \boldsymbol{\Lambda}, \mathbf{z}_n &\sim \text{Normal}(\boldsymbol{\mu}_{z_n}, \boldsymbol{\Lambda}_{z_n}^{-1}),\end{aligned}$$

The joint distribution

$$p(\mathbf{X}, \mathbf{Z}, \boldsymbol{\mu}, \boldsymbol{\Lambda}, \boldsymbol{\pi}) = \prod_{n=1}^N p(\mathbf{x}_n | \mathbf{z}_n, \boldsymbol{\mu}, \boldsymbol{\Lambda}) p(\mathbf{z}_n | \boldsymbol{\pi}) p(\boldsymbol{\pi}) \prod_{k=1}^K p(\boldsymbol{\mu}_k | \boldsymbol{\Lambda}_k) p(\boldsymbol{\Lambda}_k)$$

Quiz

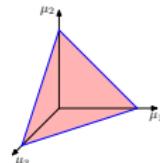
Quiz time!

Lecture 10: Mixture models on DTU Learn

The Dirichlet distribution

- A categorial distribution with values $= 1, \dots, K$ is parametrized by a K -dimensional probability vector π :

$$z \sim \text{Categorical}(\pi)$$

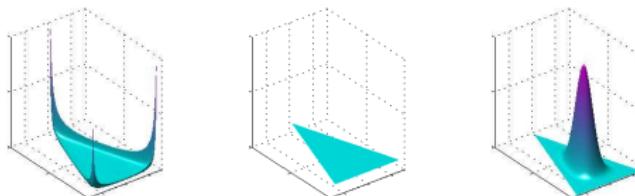


where $0 \leq \pi_k \leq 1$ and $\sum_{k=1}^K \pi_k = 1$

- The *Dirichlet distribution* $\pi \sim \text{Dir}(\alpha)$ is a conjugate prior for the categorical distribution

$$\text{Dir}(\pi | \alpha) = \frac{\Gamma(\alpha)}{\Gamma(\alpha_1) \cdots \Gamma(\alpha_K)} \prod_{k=1}^K \pi_k^{\alpha_k - 1} \quad \text{where} \quad \alpha = \sum_{k=1}^K \alpha_k$$

- Hyperparameter: $\alpha = [\alpha_1 \quad \alpha_2 \quad \dots \quad \alpha_K]$, where $\alpha_k > 0$. We often use $\alpha = a \cdot \mathbf{1}_K$ for some $a > 0$.
- Example with $K = 3$ and $a = 0.1$ (left), $a = 1$ (center), $a = 10$ (right)



The Wishart distribution

- For a univariate Gaussian likelihood with unknown mean and precision

$$p(\mathcal{D}|\mu, \tau) = \prod_{n=1}^N \mathcal{N}(x_n|\mu, \tau^{-1})$$

- Conjugate prior for the mean and the precision

$$p(\tau) = \text{Gamma}(\tau|a_0, b_0)$$

$$p(\mu|\tau) = \mathcal{N}(\mu|\mu_0, (\lambda_0\tau)^{-1})$$

- High-dimensional equivalent

$$p(\mathcal{D}|\boldsymbol{\mu}, \boldsymbol{\Lambda}) = \prod_{n=1}^N \mathcal{N}(x_n|\boldsymbol{\mu}, \boldsymbol{\Lambda}^{-1})$$

- The *Wishart* prior is a *distribution over precision matrices* and is the high-dimensional equivalent of the Gamma distribution

$$\mathcal{W}(\boldsymbol{\Lambda}|\boldsymbol{W}_0, \nu_0) = B|\boldsymbol{\Lambda}|^{(\nu-D-1)/2} \exp\left(-\frac{1}{2}\text{Tr}\left[\boldsymbol{W}_0^{-1}\boldsymbol{\Lambda}\right]\right)$$

- Hyperparameters: $\nu_0 > D - 1$ and $\boldsymbol{W}_0 \in \mathbb{R}^{D \times D}$

- Mean: $\mathbb{E}[\boldsymbol{\Lambda}] = \nu \boldsymbol{W}_0$ and $\mathbb{E}[\boldsymbol{\Lambda}^{-1}] = \frac{1}{\nu-D-1} \boldsymbol{W}_0^{-1}$

Variational inference for the mixture model

- Our goal is to compute the posterior distribution of all parameters of the mixture model

$$p(\mathbf{Z}, \boldsymbol{\mu}, \boldsymbol{\Lambda}, \boldsymbol{\pi} | \mathbf{X}) = \frac{p(\mathbf{X} | \mathbf{Z}, \boldsymbol{\mu}, \boldsymbol{\Lambda}) p(\mathbf{Z} | \boldsymbol{\pi}) p(\boldsymbol{\pi}) p(\boldsymbol{\mu}, \boldsymbol{\Lambda})}{p(\mathbf{X})}$$

- Calculating the evidence requires us to sum over all possible K^N possible assignments

- We use variational inference with a factorized approximation

$$q(\mathbf{Z}, \boldsymbol{\mu}, \boldsymbol{\Lambda}, \boldsymbol{\pi}) = q(\mathbf{Z})q(\boldsymbol{\mu}, \boldsymbol{\Lambda}, \boldsymbol{\pi})$$

- This is *the only assumption* we need to make inference feasible!

- Iterative algorithm to minimize the KL divergence

$$\ln q(\mathbf{Z}) \propto \mathbb{E}_{q(\boldsymbol{\mu}, \boldsymbol{\Lambda}, \boldsymbol{\pi})} [\ln p(\mathbf{X}, \mathbf{Z}, \boldsymbol{\mu}, \boldsymbol{\Lambda}, \boldsymbol{\pi})]$$

$$\ln q(\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Lambda}) \propto \mathbb{E}_{q(\mathbf{Z})} [\ln p(\mathbf{X}, \mathbf{Z}, \boldsymbol{\mu}, \boldsymbol{\Lambda}, \boldsymbol{\pi})]$$

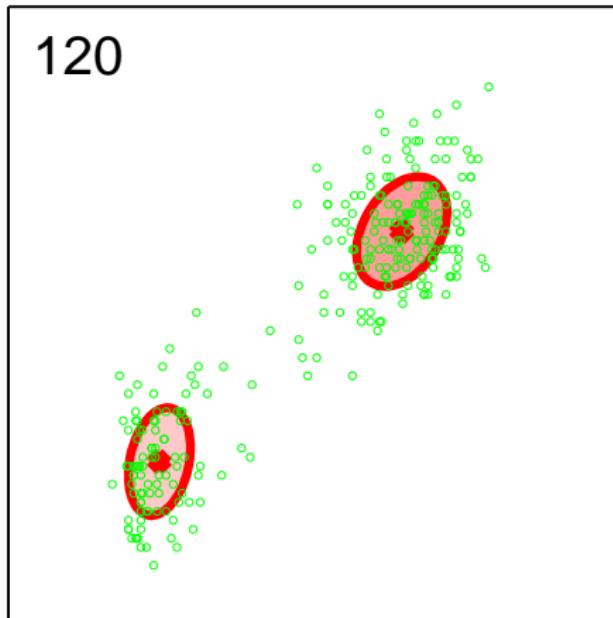
- Resulting approximation

$$q(\mathbf{Z}, \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Lambda}) = q(\mathbf{Z})q(\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Lambda})$$

$$= \underbrace{\prod_{n=1}^N \text{Categorical}(z_n | r_n)}_{q(\mathbf{Z})} \underbrace{\text{Dir}(\boldsymbol{\pi} | \boldsymbol{\alpha})}_{q(\boldsymbol{\pi})} \underbrace{\prod_{k=1}^K \mathcal{N}(\boldsymbol{\mu}_k | \mathbf{m}_k, [\beta_k \boldsymbol{\Lambda}_k^{-1}])}_{q(\boldsymbol{\mu}, \boldsymbol{\Lambda})} \mathcal{W}(\boldsymbol{\Lambda}_k | W_k, \nu_k)$$

Example: Old faithful

- $N = 272$ observations from hydrothermal geyser in Yellowstone National Park
- Feature: x_1 eruption time (minutes), x_2 time until next eruption (minutes)
- Initialize Variational GMM with $K = 6$ clusters



Clustering images

- Initialize Variational GMM using 30 clusters, 10k images for training and 10k test

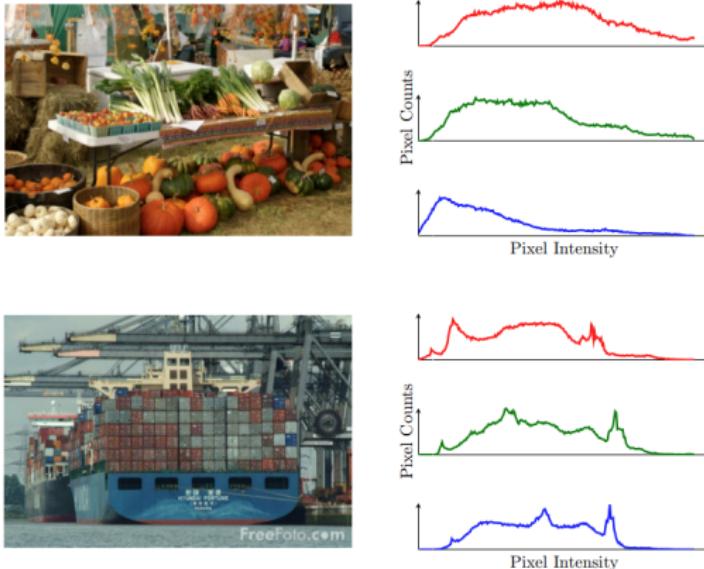


Figure 4: Red, green, and blue channel image histograms for two images from the imageCLEF dataset. The top image lacks blue hues, which is reflected in its blue channel histogram. The bottom image has a few dominant shades of blue and green, as seen in the peaks of its histogram.

Clustering images

Visualizing 4 of the clusters



(a) Purple



(b) Green & White



(c) Orange



(d) Grayish Blue

Clustering images

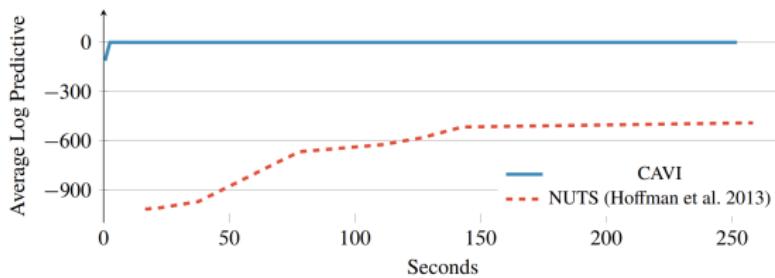


Figure 6: Comparison of CAVI to a Hamiltonian Monte Carlo-based sampling technique. CAVI fits a Gaussian mixture model to ten thousand images in less than a minute.

02477 – Bayesian Machine Learning: Lecture 11

Michael Riis Andersen

Technical University of Denmark,
DTU Compute, Department of Applied Math and Computer Science

Outline

- 1 The ELBO objective
- 2 Free-form vs fixed-form variational inference
- 3 Hyperparameter estimation in VI
- 4 Scaling Gaussian processes using variational inference

The ELBO objective

Variational inference: big picture

- Our goal is to approximate a posterior distribution of interest

$$p \equiv p(\mathbf{w}|\mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{w})p(\mathbf{w})}{p(\mathbf{y})}$$

- Variational inference in three steps

1. Define collection of "simple" approximate probability distributions \mathcal{Q} (*the variational family*)
 2. Define a measure of "distance" between probability distributions $\mathbb{D}[q||p]$
 3. Search for the distribution $q \in \mathcal{Q}$ that resembles the exact posterior p as close as possible as measured by $\mathbb{D}[q||p]$
- The variational approximation q for the target distribution $p \approx q$ is defined as

$$q_* = \arg \min_{q \in \mathcal{Q}} \mathbb{D}[q||p]$$

Understanding the ELBO objective

- Suppose our model of interest is

$$p(\mathbf{y}, \mathbf{w}) = \prod_{n=1}^N p(y_n | \mathbf{w}) p(\mathbf{w})$$

- We optimize the ELBO \mathcal{L} to minimize the KL divergence (*between approximation and target*)

$$\mathcal{L}[q] = \mathbb{E}_q [\ln p(\mathbf{y}, \mathbf{w})] - \mathbb{E}_q [\ln q(\mathbf{w})]$$

- Re-writing the lower bound

$$\begin{aligned}\mathcal{L}[q] &\equiv \mathbb{E}_q \left[\ln \prod_{n=1}^N p(y_n | \mathbf{w}) p(\mathbf{w}) \right] - \mathbb{E}_q [\ln q(\mathbf{w})] \\&= \sum_{n=1}^N \mathbb{E}_q [\ln p(y_n | \mathbf{w})] + \mathbb{E}_q [\ln p(\mathbf{w})] - \mathbb{E}_q [\ln q(\mathbf{w})] \\&= \sum_{n=1}^N \mathbb{E}_q [\ln p(y_n | \mathbf{w})] - \mathbb{E}_q \left[\ln \frac{q(\mathbf{w})}{p(\mathbf{w})} \right] \\&= \sum_{n=1}^N \mathbb{E}_q [\ln p(y_n | \mathbf{w})] - \text{KL}[q(\mathbf{w}) || p(\mathbf{w})]\end{aligned}$$

- The first term (expected log likelihood) is a *data-fit* term, while the KL term encourages q to be close to the prior $p(\mathbf{w})$ (*regularization term*)

Free-form vs fixed-form variational inference

Free-form variational inference

- *Factorized approximation* for approximating the target distribution $p \equiv p(\mathbf{w}|\mathbf{y})$

$$q(\mathbf{w}) = \prod_{j=1}^J q(\mathbf{w}_j), \quad \text{where } \mathbf{w} = [\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_J]$$

- Optimality condition for mean-field families

$$\ln q^*(\mathbf{w}_k) = \mathbb{E}_{i \neq k} [\ln p(\mathbf{y}, \mathbf{w})] + K$$

- Strategy for free-form variational inference

1. Deduce optimal form for each factor and derive parameters
2. Derive fixed-point algorithm based on the optimal parameters

+ Optimal functional form given assumptions

- Requires model-specific derivations

+ Fast optimization

- Required integrals may be intractable

- Optimal forms may not be "known" distributions

Fixed-form variational inference

- **Fixed-form** variational inference as an alternative to **free-form**: We give up an optimal functional form and simply *assume* some family of distributions q_ψ with parameters ψ

- We often refer to ψ as *variational parameters*

- **Example:** Consider a model $p(\mathbf{y}, \mathbf{w})$, where $\mathbf{w} \in \mathbb{R}^D$

- Full-rank Gaussians: $\mathbf{m} \in \mathbb{R}^D$, $\mathbf{V} \in \mathbb{R}^{D \times D}$

$$q(\mathbf{w}) = \mathcal{N}(\mathbf{w} | \mathbf{m}, \mathbf{V}),$$

- Low-rank Gaussians: $\mathbf{m} \in \mathbb{R}^D$, $\mathbf{B} \in \mathbb{R}^{D \times K}$, and $\mathbf{C} \in \mathbb{R}^{D \times D}$ is diagonal.

$$q(\mathbf{w}) = \mathcal{N}(\mathbf{w} | \mathbf{m}, \mathbf{B}\mathbf{B}^T + \mathbf{C}^2),$$

- Mean-field Gaussians: $\mathbf{m} = [m_1, \dots, m_D] \in \mathbb{R}^D$ and $\mathbf{v} = [v_1, \dots, v_D] = \mathbb{R}^D$

$$q(\mathbf{w}) = \prod_{i=1}^D \mathcal{N}(w_i | m_i, v_i),$$

- Non-restricted to Gaussians: Gamma, Gauss, Beta, Dirichlet, Bernouilli etc

Fixed-form variational inference II

- *Fixed-form* variational inference as an alternative to *free-form*

- **Example:** Consider a model $p(\mathbf{y}, \mathbf{w})$, where $\mathbf{w} \in \mathbb{R}^D$

- Suppose we choose a full-rank Gaussian family

$$q(\mathbf{w}) = \mathcal{N}(\mathbf{w} | \mathbf{m}, \mathbf{V}),$$

such that the variational family \mathcal{Q} consists of all multivariate Gaussian distributions

- q_ψ is now *parametrized* by the *variational parameters* $\psi = \{\mathbf{m}, \mathbf{V}\}$

$$\begin{aligned}\mathcal{L}[q_\psi] &= \mathbb{E}_{q_\psi} [\ln p(\mathbf{y}, \mathbf{w})] - \mathbb{E}_{q_\psi} [\ln q_\psi(\mathbf{w})] \\ &= \mathbb{E}_{\mathcal{N}(\mathbf{w} | \mathbf{m}, \mathbf{V})} [\ln p(\mathbf{y}, \mathbf{w})] - \mathbb{E}_{\mathcal{N}(\mathbf{w} | \mathbf{m}, \mathbf{V})} [\ln \mathcal{N}(\mathbf{w} | \mathbf{m}, \mathbf{V})]\end{aligned}$$

- Fitting the approximation using gradient-based methods

$$q^* = \arg \max_{q \in \mathcal{Q}} \mathcal{L}[q] \iff \psi^* = \arg \max_{\psi} \mathcal{L}[q_\psi] \iff \mathbf{m}^*, \mathbf{V}^* = \arg \max_{\mathbf{m}, \mathbf{V}} \mathcal{L}[q_\psi]$$

Hyperparameter estimation in VI

Dealing with hyperparameters for variational inference

- Consider a model with data \mathbf{y} , parameters θ , and hyperparameters ξ

$$p(\theta|\mathbf{y}, \xi) = \frac{p(\mathbf{y}|\theta, \xi)p(\theta|\xi)}{p(\mathbf{y}|\xi)}$$

Examples

- Linear regression

$\theta = \mathbf{w}$ would be the regression weights
 $\xi = \{\alpha, \beta\}$ would be prior and noise precision.

- Gaussian process regression

$\theta = f$ would be the latent function values
 $\xi = \{\sigma^2, \kappa, \ell\}$ would be noise variance and kernel hyperparameters

- Hyperparameter estimation via the marginal likelihood (MLII/MAPII)

$$\hat{\xi} = \arg \max_{\xi} \log p(\mathbf{y}|\xi)$$

- But what to do for non-conjugate model, where we cannot compute the marginal likelihood?

$$p(\theta|\mathbf{y}, \xi) \approx q(\theta)$$

Dealing with hyperparameters for variational inference

- Consider a model with data \mathbf{y} , parameters θ , and hyperparameters ξ

$$p(\theta|\mathbf{y}, \xi) = \frac{p(\mathbf{y}|\theta, \xi)p(\theta|\xi)}{p(\mathbf{y}|\xi)}$$

- Variational approximation $p(\theta|\mathbf{y}, \xi) \approx q(\theta)$ by optimizing the ELBO

$$\mathcal{L}_\xi[q] = \mathbb{E}_q[\ln p(\mathbf{y}, \theta|\xi)] - \mathbb{E}_q[\ln q(\theta)]$$

- The ELBO is a lowerbound on the log marginal likelihood

$$\log p(\mathbf{y}|\xi) \geq \mathcal{L}_\xi[q]$$

- and hence, we can do hyperparameter estimation via ELBO

$$\hat{\xi} = \arg \max_{\xi} p(\mathbf{y}|\xi) \approx \arg \max_{\xi} \mathcal{L}_\xi[q]$$

- Key take-away:** We can optimize the ELBO wrt. *variational parameters* ψ and *hyperparameters* ξ simultaneously

$$\hat{\psi}^*, \hat{\xi}^* = \arg \max_{\xi} \mathcal{L}_\xi[q_\psi]$$

Scaling Gaussian processes using variational inference

Speeding up Gaussian process inference

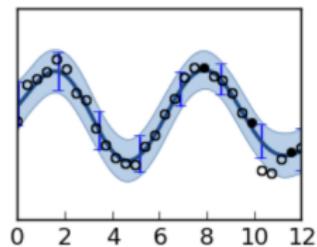
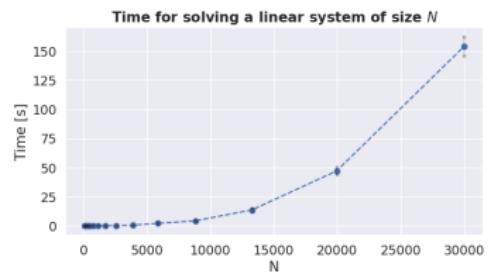
- Gaussian process priors, $f \sim \mathcal{GP}(0, k(\mathbf{x}, \mathbf{x}'))$ can be extremely powerful, but scales poorly: $\mathcal{O}(N^3)$ in computational complexity and $\mathcal{O}(N^2)$ in memory footprint

$$p(y^* | \mathbf{y}) = \mathcal{N}\left(y^* | \mu_{y^* | \mathbf{y}}, \sigma_{y^* | \mathbf{y}}^2\right)$$

$$\mu_{y^* | \mathbf{y}} = \mathbf{k} (\mathbf{K} + \beta^{-1} \mathbf{I})^{-1} \mathbf{y}$$

$$\sigma_{y^* | \mathbf{y}}^2 = c - \mathbf{k} (\mathbf{K} + \beta^{-1} \mathbf{I})^{-1} \mathbf{k}^T$$

- Obviously, using a subset of data would be faster, but can we do something more clever? Yes!
- We introduce *inducing points* \mathbf{z}_i for $i = 1, \dots, M$ for $M \ll N$
- Combining *variational inference* with *inducing points* allows us to approximate the posterior much faster, i.e. $\mathcal{O}(NM^2)$, and even faster, $\mathcal{O}(M^2)$, using *mini-batching* (next week)



Hensman et al, 2019: Gaussian Processes for Big Data

Introducing inducing points

- Suppose we have dataset $\mathcal{D} = \{\mathbf{x}_n, y_n\}$, where $\mathbf{x}_n \in \mathbb{R}^D$ and $y_n \in \mathbb{R}$

$$y_n = f(\mathbf{x}_n) + e_n$$

- Assuming Gaussian noise and a GP prior, i.e. $f(\mathbf{x}) \sim \mathcal{GP}(\mathbf{0}, k(\mathbf{x}, \mathbf{x}'))$ yields the joint distribution

$$p(\mathbf{y}, \mathbf{f}) = p(\mathbf{y}|\mathbf{f})p(\mathbf{f}) = \mathcal{N}(\mathbf{y}|\mathbf{f}, \sigma^2 \mathbf{I})\mathcal{N}(\mathbf{f}|\mathbf{0}, \mathbf{K}_{ff})$$

- Goal:** fast way compute $p(\mathbf{f}|\mathbf{y})$

$$f_n = f(\mathbf{x}_n)$$

- We introduce M *inducing points* $\mathbf{z}_i \in \mathbb{R}^D$ for $m = 1, \dots, M$ such that

$$u_i = f(\mathbf{z}_i)$$

- Extended joint distribution for data \mathbf{y} and latent function values for both \mathbf{f} and \mathbf{u}

$$p(\mathbf{y}, \mathbf{f}, \mathbf{u}) = p(\mathbf{y}|\mathbf{f})p(\mathbf{f}|\mathbf{u})p(\mathbf{u}) = \mathcal{N}(\mathbf{y}|\mathbf{f}, \sigma^2 \mathbf{I})\mathcal{N}(\mathbf{f}|\mathbf{K}_{fu}\mathbf{K}_{uu}^{-1}\mathbf{u}, \mathbf{K}_{ff} - \mathbf{K}_{fu}\mathbf{K}_{uu}^{-1}\mathbf{K}_{fu})\mathcal{N}(\mathbf{u}|\mathbf{0}, \mathbf{K}_{uu})$$

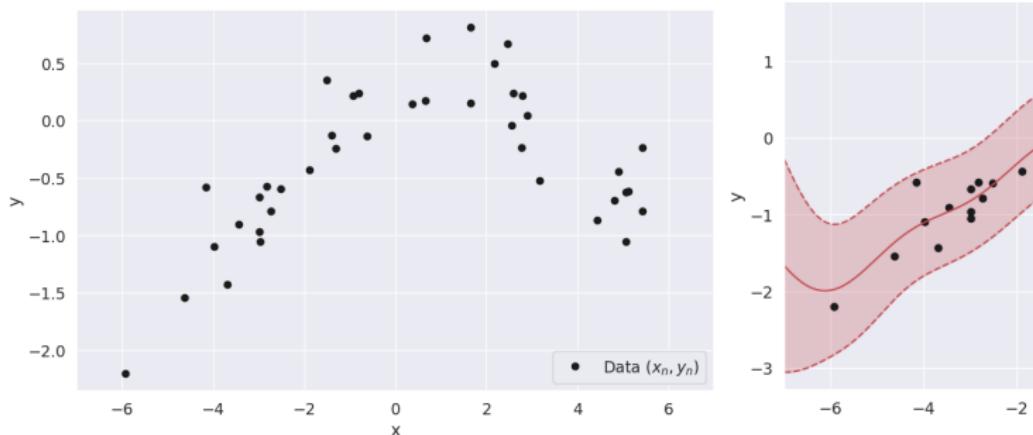
- We can always marginalize to get the original model back

$$p(\mathbf{y}, \mathbf{f}) = \int p(\mathbf{y}, \mathbf{f}, \mathbf{u})d\mathbf{u}$$

Intuition

- Our extended joint distribution

$$p(\mathbf{y}, \mathbf{f}, \mathbf{u}) = p(\mathbf{y}|\mathbf{f})p(\mathbf{f}|\mathbf{u})p(\mathbf{u}) = \mathcal{N}(\mathbf{y}|\mathbf{f}, \sigma^2 \mathbf{I})\mathcal{N}(\mathbf{f}|\mathbf{K}_{fu}\mathbf{K}_{uu}^{-1}\mathbf{u}, \mathbf{K}_{ff} - \mathbf{K}_{fu}\mathbf{K}_{uu}^{-1}\mathbf{K}_{fu})\mathcal{N}(\mathbf{u}|\mathbf{0}, \mathbf{K}_{uu})$$



- Can we find a set of *inducing points* $\{\mathbf{z}_i\}_{i=1}^M$ and associated function values $u_i = f(\mathbf{z}_i)$ such that we can absorb all the information from the full data set?

Setting up the approximation and choosing a variational family

- Our extended joint distribution

$$p(\mathbf{y}, \mathbf{f}, \mathbf{u}) = p(\mathbf{y}|\mathbf{f})p(\mathbf{f}|\mathbf{u})p(\mathbf{u}) = \mathcal{N}(\mathbf{y}|\mathbf{f}, \sigma^2 \mathbf{I})\mathcal{N}(\mathbf{f}|\mathbf{K}_{fu}\mathbf{K}_{uu}^{-1}\mathbf{u}, \mathbf{K}_{ff} - \mathbf{K}_{fu}\mathbf{K}_{uu}^{-1}\mathbf{K}_{fu})\mathcal{N}(\mathbf{u}|\mathbf{0}, \mathbf{K}_{uu})$$

- We will make a variational approximation: $p(\mathbf{f}, \mathbf{u}|\mathbf{y}) \approx q(\mathbf{f}, \mathbf{u})$

- A clever choice for the variational family

$$q(\mathbf{f}, \mathbf{u}) = p(\mathbf{f}|\mathbf{u})q(\mathbf{u}) = \mathcal{N}(\mathbf{f}|\mathbf{K}_{fu}\mathbf{K}_{uu}^{-1}\mathbf{u}, \mathbf{K}_{ff} - \mathbf{K}_{fu}\mathbf{K}_{uu}^{-1}\mathbf{K}_{fu})\mathcal{N}(\mathbf{u}|\mathbf{m}_u, \mathbf{S}_u)$$

where

$\mathbf{m}_u \in \mathbb{R}^M$ and $\mathbf{S}_u \in \mathbb{R}^{M \times M}$ are variational parameters to be estimated

\mathbf{K}_{ff} is the prior covariance matrix for \mathbf{f}

\mathbf{K}_{uu} is the prior covariance matrix for \mathbf{u}

\mathbf{K}_{fu} is the prior covariance between \mathbf{f} and \mathbf{u}

- Recall $p(\mathbf{f}|\mathbf{u})$ is the prior conditional distribution derived from the prior $p(\mathbf{f}, \mathbf{u})$.

The approximate posterior distribution

- Our choice of variational family:

$$q(\mathbf{f}, \mathbf{u}) = p(\mathbf{f}|\mathbf{u})q(\mathbf{u}) = \mathcal{N}(\mathbf{f}|\mathbf{K}_{fu}\mathbf{K}_{uu}^{-1}\mathbf{u}, \mathbf{K}_{ff} - \mathbf{K}_{fu}\mathbf{K}_{uu}^{-1}\mathbf{K}_{fu})\mathcal{N}(\mathbf{u}|\mathbf{m}_u, \mathbf{S}_u)$$

implies a marginal variational approximation for $q(\mathbf{f})$ (linear Gaussian system)

$$\begin{aligned} q(\mathbf{f}) &= \int p(\mathbf{f}|\mathbf{u})q(\mathbf{u})d\mathbf{u} \\ &= \int \mathcal{N}(\mathbf{f}|\mathbf{K}_{fu}\mathbf{K}_{uu}^{-1}\mathbf{u}, \mathbf{K}_{ff} - \mathbf{K}_{fu}\mathbf{K}_{uu}^{-1}\mathbf{K}_{fu})\mathcal{N}(\mathbf{u}|\mathbf{m}_u, \mathbf{S}_u)d\mathbf{u} \\ &= \mathcal{N}(\mathbf{f} | \underbrace{\mathbf{K}_{fu}\mathbf{K}_{uu}^{-1}\mathbf{m}_u}_{\mathbf{m}_f}, \underbrace{\mathbf{K}_{ff} + \mathbf{K}_{fu}\mathbf{K}_{uu}^{-1}(\mathbf{S}_u - \mathbf{K}_{uu})\mathbf{K}_{uu}^{-1}\mathbf{K}_{uf}}_{\mathbf{S}_f}) \\ &= \mathcal{N}(\mathbf{f}|\mathbf{m}_f, \mathbf{S}_f) \end{aligned}$$

- ... and similarly we can make predictions for new input points to get $f^* = f(\mathbf{x}^*)$

$$\begin{aligned} p(f^*|\mathbf{y}) &= \int p(f^*|\mathbf{u})p(\mathbf{u}|\mathbf{y})d\mathbf{u} \\ &\approx \int p(f^*|\mathbf{u})q(\mathbf{u})d\mathbf{u} \\ &= \mathcal{N}(f^* | \underbrace{\mathbf{K}_{f^*u}\mathbf{K}_{uu}^{-1}\mathbf{m}_u}_{\mathbf{m}_{f^*}}, \underbrace{\mathbf{K}_{f^*f^*} + \mathbf{K}_{f^*u}\mathbf{K}_{uu}^{-1}(\mathbf{S}_u - \mathbf{K}_{uu})\mathbf{K}_{uu}^{-1}\mathbf{K}_{uf^*}}_{\mathbf{S}_{f^*s}}) \end{aligned}$$

- So all we need to do is to estimate the mean and covariance for $q(\mathbf{u}) = \mathcal{N}(\mathbf{u}|\mathbf{m}_u, \mathbf{S}_u)$

Calculating the ELBO

We substitute our model and variational approximation into the ELBO

$$\begin{aligned}\mathcal{L}[q] &= \mathbb{E}_{p(\mathbf{f}|\mathbf{u})q(\mathbf{u})} [\log p(\mathbf{y}, \mathbf{f}, \mathbf{u})] - \mathbb{E}_{p(\mathbf{f}|\mathbf{u})q(\mathbf{u})} [\log q(\mathbf{f}, \mathbf{u})] \\&= \mathbb{E}_{p(\mathbf{f}|\mathbf{u})q(\mathbf{u})} [\log p(\mathbf{y}|\mathbf{f})p(\mathbf{f}|\mathbf{u})p(\mathbf{u})] - \mathbb{E}_{p(\mathbf{f}|\mathbf{u})q(\mathbf{u})} [\log p(\mathbf{f}|\mathbf{u})q(\mathbf{u})] \\&= \mathbb{E}_{p(\mathbf{f}|\mathbf{u})q(\mathbf{u})} [\log p(\mathbf{y}|\mathbf{f})] + \mathbb{E}_{p(\mathbf{f}|\mathbf{u})q(\mathbf{u})} [\log p(\mathbf{f}|\mathbf{u})] + \mathbb{E}_{p(\mathbf{f}|\mathbf{u})q(\mathbf{u})} [\log p(\mathbf{u})] \\&\quad - \mathbb{E}_{p(\mathbf{f}|\mathbf{u})q(\mathbf{u})} [\log p(\mathbf{f}|\mathbf{u})] - \mathbb{E}_{p(\mathbf{f}|\mathbf{u})q(\mathbf{u})} [\log q(\mathbf{u})] \\&= \mathbb{E}_{p(\mathbf{f}|\mathbf{u})q(\mathbf{u})} [\log p(\mathbf{y}|\mathbf{f})] + \mathbb{E}_{p(\mathbf{f}|\mathbf{u})q(\mathbf{u})} [\log p(\mathbf{u})] - \mathbb{E}_{p(\mathbf{f}|\mathbf{u})q(\mathbf{u})} [\log q(\mathbf{u})] \\&= \mathbb{E}_{p(\mathbf{f}|\mathbf{u})q(\mathbf{u})} [\log p(\mathbf{y}|\mathbf{f})] + \mathbb{E}_{q(\mathbf{u})} [\log p(\mathbf{u})] - \mathbb{E}_{q(\mathbf{u})} [\log q(\mathbf{u})] \\&= \mathbb{E}_{p(\mathbf{f}|\mathbf{u})q(\mathbf{u})} [\log p(\mathbf{y}|\mathbf{f})] - \text{KL}[q(\mathbf{u})||p(\mathbf{u})]\end{aligned}$$

Next, recall that $p(\mathbf{y}|\mathbf{f}) = \mathcal{N}(\mathbf{y}|\mathbf{f}, \sigma^2 \mathbf{I}) = \prod_{n=1}^N \mathcal{N}(y_n|f_n, \sigma^2)$

$$\begin{aligned}\mathcal{L}[q] &= \sum_{n=1}^N \mathbb{E}_{p(\mathbf{f}|\mathbf{u})q(\mathbf{u})} [\log \mathcal{N}(y_n|f_n, \sigma^2)] - \text{KL}[q(\mathbf{u})||p(\mathbf{u})] \\&= \sum_{n=1}^N \mathbb{E}_{p(f_n|\mathbf{u})q(\mathbf{u})} [\log \mathcal{N}(y_n|f_n, \sigma^2)] - \text{KL}[q(\mathbf{u})||p(\mathbf{u})]\end{aligned}$$

Calculating the first term

$$\begin{aligned} \mathbb{E}_{p(f_n|\mathbf{u})q(\mathbf{u})} [\log \mathcal{N}(y_n|f_n, \sigma^2)] &= \int \int p(f_n|\mathbf{u})q(\mathbf{u}) \log \mathcal{N}(y_n|f_n, \sigma^2) df_n d\mathbf{u} \\ &= \int \int p(f_n|\mathbf{u})q(\mathbf{u}) d\mathbf{u} \log \mathcal{N}(y_n|f_n, \sigma^2) df_n && (\text{Recall: } q(\mathbf{f}) = \mathcal{N}(\mathbf{f}|\mathbf{m}_f, \mathbf{S}_f)) \\ &= \int \mathcal{N}(f_n|m_{f_n}, \sigma_{f_n}^2) \log \mathcal{N}(y_n|f_n, \sigma^2) df_n \\ &= \mathbb{E}_{\mathcal{N}(f_n|m_{f_n}, \sigma_{f_n}^2)} [\log \mathcal{N}(y_n|f_n, \sigma^2)] \\ &= \mathbb{E}_{\mathcal{N}(f_n|m_{f_n}, \sigma_{f_n}^2)} \left[-\frac{1}{2} \log(2\pi\sigma^2) - \frac{1}{2\sigma} (y_n - f_n)^2 \right] \\ &= \mathbb{E}_{\mathcal{N}(f_n|m_{f_n}, \sigma_{f_n}^2)} \left[-\frac{1}{2} \log(2\pi\sigma^2) - \frac{1}{2\sigma} (y_n^2 + f_n^2 - 2y_n f_n) \right] \\ &= -\frac{1}{2} \log(2\pi\sigma^2) - \frac{1}{2\sigma} (y_n^2 + \mathbb{E}_{\mathcal{N}(f_n|m_{f_n}, \sigma_{f_n}^2)} [f_n^2] - 2y_n \mathbb{E}_{\mathcal{N}(f_n|m_{f_n}, \sigma_{f_n}^2)} [f_n]) \\ &= -\frac{1}{2} \log(2\pi\sigma^2) - \frac{1}{2\sigma} (y_n^2 + m_{f_n}^2 + \sigma_{f_n}^2 - 2y_n m_{f_n}) \\ &= -\frac{1}{2} \log(2\pi\sigma^2) - \frac{1}{2\sigma} (y_n^2 + m_{f_n}^2 - 2y_n m_{f_n}) - \frac{1}{2\sigma} \sigma_{f_n}^2 \\ &= -\frac{1}{2} \log(2\pi\sigma^2) - \frac{1}{2\sigma} (y_n - m_{f_n})^2 - \frac{1}{2\sigma} \sigma_{f_n}^2 \\ &= \log \mathcal{N}(y_n|m_{f_n}, \sigma^2) df_n - \frac{1}{2\sigma} \sigma_{f_n}^2 \end{aligned}$$

The KL term

- The two terms in the lowerbound

$$\mathcal{L}[q] = \sum_{n=1}^N \mathbb{E}_{p(f_n|\mathbf{u})q(\mathbf{u})} [\log \mathcal{N}(y_n|f_n, \sigma^2)] - \text{KL}[q(\mathbf{u})||p(\mathbf{u})]$$

- The KL divergence between two multivariate Gaussians can be computed in closed-form:

$$\begin{aligned} & \text{KL}[\mathcal{N}(\mathbf{u}|\mathbf{m}_0, \Sigma_0) || \mathcal{N}(\mathbf{u}|\mathbf{m}_1, \Sigma_1)] \\ &= \frac{1}{2} \left[\text{trace}(\Sigma_1^{-1} \Sigma_0) + (\mu_1 - \mu_0)^T \Sigma_1^{-1} (\mu_1 - \mu_0) - D + \log \frac{|\Sigma_1|}{|\Sigma_0|} \right] \end{aligned}$$

- So for $q(\mathbf{u}) = \mathcal{N}(\mathbf{u}|\mathbf{m}_u, \mathbf{S}_u)$ and $p(\mathbf{u}) = \mathcal{N}(\mathbf{u}|\mathbf{0}, \mathbf{K}_{uu})$, we get

$$\text{KL}[q(\mathbf{u}) || p(\mathbf{u})] = \frac{1}{2} \left[\text{trace}(\mathbf{K}_{uu}^{-1} \mathbf{S}_u) + \mathbf{m}_u^T \mathbf{K}_{uu}^{-1} \mathbf{m}_u - M + \log \frac{|\mathbf{K}_{uu}|}{|\mathbf{S}_u|} \right]$$

Combining everything

- We derived

$$\mathcal{L}[q] = \sum_{n=1}^N \mathbb{E}_{p(f_n|\mathbf{u})q(\mathbf{u})} \left[\log \mathcal{N}(y_n | f_n, \sigma^2) \right] - \text{KL}[q(\mathbf{u}) || p(\mathbf{u})]$$

- ... and then showed that the first term simplifies to

$$\mathbb{E}_{p(f_n|\mathbf{u})q(\mathbf{u})} \left[\log \mathcal{N}(y_n | f_n, \sigma^2) \right] = \log \mathcal{N}(y_n | m_{f_n}, \sigma^2) df_n - \frac{1}{2\sigma} \sigma_{f_n}^2$$

- Combining yields

$$\begin{aligned} \mathcal{L}[q] &= \sum_{n=1}^N \left[\log \mathcal{N}(y_n | m_{f_n}, \sigma^2) df_n - \frac{1}{2\sigma} \sigma_{f_n}^2 \right] - \text{KL}[q(\mathbf{u}) || p(\mathbf{u})] \\ &= \sum_{n=1}^N \log \mathcal{N}(y_n | m_{f_n}, \sigma^2) - \frac{1}{2\sigma} \sum_{n=1}^N \sigma_{f_n}^2 - \text{KL}[q(\mathbf{u}) || p(\mathbf{u})] \\ &= \log \mathcal{N}(\mathbf{y} | \mathbf{m}_f, \sigma^2 \mathbf{I}) - \frac{1}{2\sigma} \text{trace}(\mathbf{S}_f) - \text{KL}[q(\mathbf{u}) || p(\mathbf{u})] \end{aligned}$$

where

$$\mathbf{m}_f = \mathbf{K}_{fu} \mathbf{K}_{uu}^{-1} \mathbf{m}_u$$

$$\mathbf{S}_f = \mathbf{K}_{ff} + \mathbf{K}_{fu} \mathbf{K}_{uu}^{-1} (\mathbf{S}_u - \mathbf{K}_{uu}) \mathbf{K}_{uu}^{-1} \mathbf{K}_{uf}$$

- We can optimize this bound wrt. the *variational parameters* \mathbf{m}_u and \mathbf{S}_u and wrt. the *hyperparameters* simultaneously!

One step more step

- It turns out we can optimize the bound wrt. \mathbf{m}_u and \mathbf{S}_u analytically

$$\begin{aligned}\mathcal{L}[q] &= \sum_{n=1}^N \left[\log \mathcal{N}(y_n | m_{f_n}, \sigma^2) df_n - \frac{1}{2\sigma} \sigma_{f_n}^2 \right] - \text{KL}[q(\mathbf{u}) || p(\mathbf{u})] \\ &= \sum_{n=1}^N \log \mathcal{N}(y_n | m_{f_n}, \sigma^2) - \frac{1}{2\sigma} \sum_{n=1}^N \sigma_{f_n}^2 - \text{KL}[q(\mathbf{u}) || p(\mathbf{u})] \\ &= \log \mathcal{N}(\mathbf{y} | \mathbf{m}_f, \sigma^2 \mathbf{I}) - \frac{1}{2\sigma} \text{trace}(\mathbf{S}_f) - \text{KL}[q(\mathbf{u}) || p(\mathbf{u})]\end{aligned}$$

- ... to get

$$\mathbf{S}_u^{-1} = \frac{1}{\sigma^2} \mathbf{K}_{uu}^{-1} \mathbf{K}_{uf} \mathbf{K}_{fu} \mathbf{K}_{uu}^{-1} + \mathbf{K}_{uu}^{-1}$$

$$\mathbf{m}_u = \frac{1}{\sigma^2} \mathbf{S}_u \mathbf{K}_{uu}^{-1} \mathbf{K}_{uf} \mathbf{y}$$

- ... which leads to the *collapsed lowerbound*

$$\mathcal{L}[q] = \log \mathcal{N}(\mathbf{y} | \mathbf{0}, \mathbf{K}_{fu} \mathbf{K}_{uu} \mathbf{K}_{uf} + \sigma^2 \mathbf{I}) - \frac{1}{2\sigma} \text{trace}(\mathbf{K}_{ff} - \mathbf{K}_{fu} \mathbf{K}_{uu}^{-1} \mathbf{K}_{uf})$$

The big picture and how to use it in practice

Goal: fast way compute $p(\mathbf{f}|\mathbf{y})$

1. Choose a set of *inducing points* \mathbf{z}_i , where $u_i = f(z_i)$
2. Optimize the *collapsed bound* wrt. our hyperparameters θ

$$\hat{\theta}^* = \arg \max_{\theta} \mathcal{L}_{\theta} [q] = \log \mathcal{N}(\mathbf{y} | \mathbf{0}, \mathbf{K}_{fu} \mathbf{K}_{uu} \mathbf{K}_{uf} + \sigma^2 \mathbf{I}) - \frac{1}{2\sigma} \text{trace}(\mathbf{K}_{ff} - \mathbf{K}_{fu} \mathbf{K}_{uu}^{-1} \mathbf{K}_{uf})$$

3. Compute the posterior mean and covariance for latent function values
 $q(\mathbf{u}) = \mathcal{N}(\mathbf{u} | \mathbf{m}_u, \mathbf{S}_u)$

$$\begin{aligned}\mathbf{S}_u^{-1} &= \frac{1}{\sigma^2} \mathbf{K}_{uu}^{-1} \mathbf{K}_{uf} \mathbf{K}_{fu} \mathbf{K}_{uu}^{-1} + \mathbf{K}_{uu}^{-1} \\ \mathbf{m}_u &= \frac{1}{\sigma^2} \mathbf{S}_u \mathbf{K}_{uu}^{-1} \mathbf{K}_{uf} \mathbf{y}\end{aligned}$$

4. Compute the approximate posterior distribution for $q(\mathbf{f}) = \mathcal{N}(\mathbf{f} | \mathbf{m}_f, \mathbf{S}_f)$

$$\begin{aligned}\mathbf{m}_f &= \mathbf{K}_{fu} \mathbf{K}_{uu}^{-1} \mathbf{m}_u \\ \mathbf{S}_f &= \mathbf{K}_{ff} + \mathbf{K}_{fu} \mathbf{K}_{uu}^{-1} (\mathbf{S}_u - \mathbf{K}_{uu}) \mathbf{K}_{uu}^{-1} \mathbf{K}_{uf}\end{aligned}$$

5. Use $p(\mathbf{f}|\mathbf{y}) \approx q(\mathbf{f})$ to make predictions

Airline delays dataset

- Flight arrival and departure times for every commercial flight in the USA from Jan. 2008 to April 2008
- 2 million flights: 700000 flight for training, 100000 for testing
- Target variable: Delay in minutes:
- $D = 8$ features: age of aircraft, flight distance, airtime, departure time, arrival time, day of the week, day of the month, month
- Squared exponential kernel with separate lengthscale $\ell_i > 0$ for each dimension

$$k(\mathbf{x}, \mathbf{x}') = \kappa^2 \exp \left[-\frac{1}{2} \sum_{i=1}^D \ell_i^{-1} |\mathbf{x}_i - \mathbf{x}'_i|^2 \right] + \tau$$

- Using $M = 1000$ inducing points

Hensman et al: Gaussian Processes for Big Data (2013)

Airline delays dataset

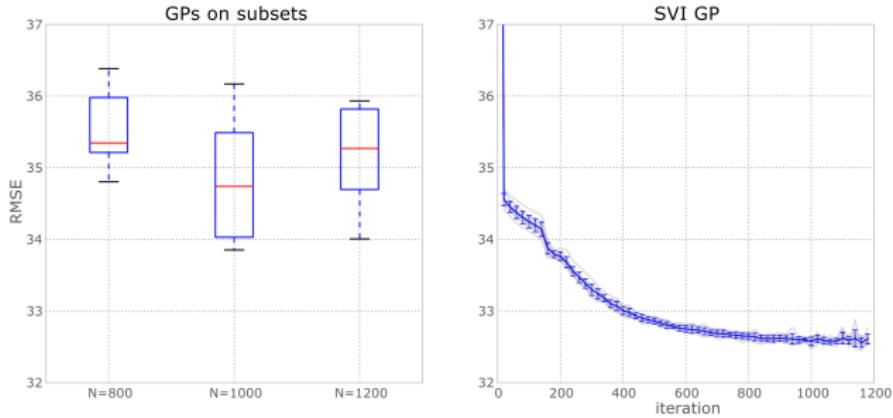


Figure 7: Root mean squared errors in predicting flight delays using information about the flight.

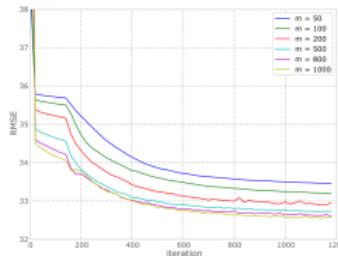


Figure 8: Root mean square errors for models with different numbers of inducing variables.

02477 – Bayesian Machine Learning: Lecture 12

Michael Riis Andersen

Technical University of Denmark,
DTU Compute, Department of Applied Math and Computer Science

Variational inference in modern machine learning

- *Variational inference* (VI) has been the topic for the last two weeks
- Variational inference has many applications in modern machine learning
 1. Variational autoencoders (VAEs)
 2. Diffusion models
 3. Bayesian neural networks
 4. ...
- Common challenges
 1. Highly non-linear models
 2. Large-scale models
 3. Huge datasets
- Last bits of VI theory today: *Black-box variational inference* (BBVI)

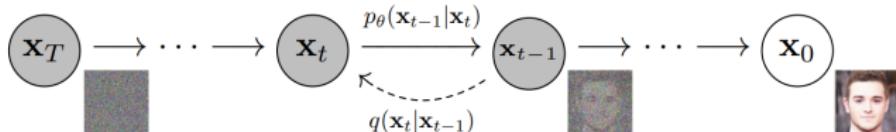


Figure 2: The directed graphical model considered in this work.

From Jonathan Ho et al (2020)

Outline

1 Variational inference and divergences

- Properties of the KL-divergence
- KL-divergences in other contexts (detour)

2 Black-box variational inference

- The evidence lower bound and entropy
- Dealing with expectations
- Stochastic gradients
- A few words on stochastic optimization
- Mini-batching for large scale inference

Variational inference and divergences

Variational inference and divergences: Properties of the KL-divergence

Measuring distance between probability distributions

- How do we measure "distance" between probability distributions $\mathbb{D}[q, p]$?
- The choice of "distance" affects the properties of the approximation
- The *Kullback-Leibler* divergence (for continuous R.V.) is defined as

$$\text{KL}[q||p] = \int q(z) \log \left[\frac{q(z)}{p(z)} \right] dz$$

- Properties

1. Identity of indiscernibles

$$\text{KL}[q||p] = 0 \iff p = q \text{ (a.e)}$$

2. Non-negativity

$$\text{KL}[q||p] \geq 0$$

3. Asymmetric

$$\text{KL}[q||p] \neq \text{KL}[p||q]$$

Properties of the KL divergence

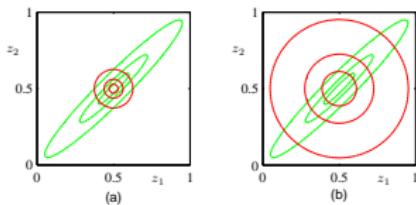
- The *variational approximation* q for target distribution $p \approx q$ is defined as

$$q_* = \arg \min_{q \in \mathcal{Q}} \text{KL}[q||p], \quad \text{KL}[q||p] = \int q(z) \log \left[\frac{q(z)}{p(z)} \right] dz$$

- Approximating a general Gaussian $p(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mu, \Sigma)$ with a mean-field Gaussian

$$q(\mathbf{w}) = \mathcal{N}(w_1|m_1, v_1)\mathcal{N}(w_2|m_2, v_2)$$

- Which approximation q (in red) has the smallest $\text{KL}[q||p]$ divergence? (a) or (b)?



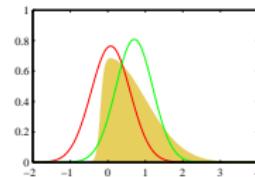
The approximation q in red
The target p distribution in green

Common approximation properties

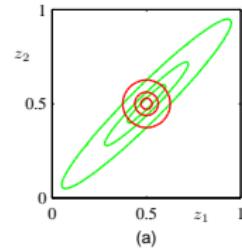
The *variational approximation* q for target distribution $p \approx q$ is defined as $q_* = \arg \min_{q \in \mathcal{Q}} \mathbb{D}[q||p]$

$$\text{KL}[q||p] = \int q(z) \log \left[\frac{q(z)}{p(z)} \right] dz \quad \text{KL}[p||q] = \int p(z) \log \left[\frac{p(z)}{q(z)} \right] dz$$

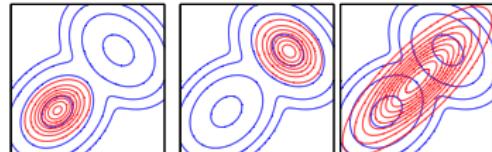
1. The variational approximation (green) often covers to posterior mass of target (yellow) better than the Laplace approximation (red)



2. For the combination of *factorized variational families and KL* $[q||p]$, the variational approximation (red) often *underestimates* the variance of the target distribution (green)



3. Approximations (red) based on $\text{KL}[q||p]$ are often said to be *mode-seeking*, whereas $\text{KL}[p||q]$ are often said to be *mass-covering*



The α -divergence

- The Kullback-Leibler divergence is not the only possible choice for the divergence

$$\text{KL}[q||p] = \int q(z) \log \frac{q(z)}{p(z)} dz$$

- The α -divergence is defined as

$$\mathbb{D}_\alpha[p||q] = \frac{1}{\alpha(1-\alpha)} \int \alpha p(z) + (1-\alpha)q(z) - p(z)^\alpha q(z)^{1-\alpha} dz$$

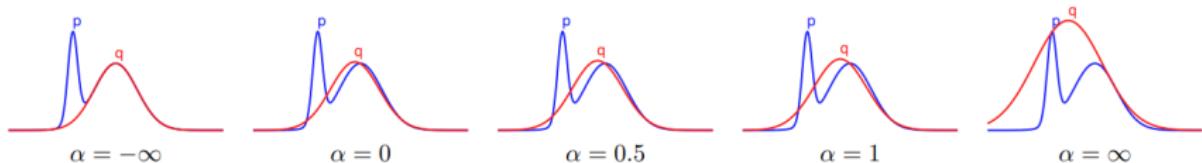
- Some interesting *special cases*

$$\lim_{\alpha \rightarrow 0} \mathbb{D}_\alpha[p||q] = \text{KL}[q||p]$$

$$\lim_{\alpha \rightarrow 1} \mathbb{D}_\alpha[p||q] = \text{KL}[p||q]$$

$$\mathbb{D}_{\frac{1}{2}}[p||q] = 2 \int \left(\sqrt{p(z)} - \sqrt{q(z)} \right) dz \quad (\text{Hellinger distance})$$

- α determines the properties of the resulting approximation $q^* = \arg \min_q \mathbb{D}_\alpha[p||q]$



Variational inference and divergences: KL-divergences in other contexts (detour)

De-tour: KL-divergences and maximum likelihood

- Suppose we have a data set $\mathcal{D} = \{\mathbf{x}_i\}_{i=1}^N$, where $\mathbf{x}_i \sim p_D$. Assume we want to fit some parametric distribution $p(\mathbf{x}|\theta)$ with parameters θ to the data \mathcal{D} .

$$\begin{aligned}\text{KL}[p_D(\mathbf{x})||p(\mathbf{x}|\theta)] &= \mathbb{E}_{p_D(\mathbf{x})} \left[\log \frac{p_D(\mathbf{x})}{p(\mathbf{x}|\theta)} \right] \\ &= \mathbb{E}_{p_D(\mathbf{x})} [\log p_D(\mathbf{x})] + \mathbb{E}_{p_D(\mathbf{x})} \left[\log \frac{1}{p(\mathbf{x}|\theta)} \right] \\ &= \mathbb{E}_{p_D(\mathbf{x})} \left[\log \frac{1}{p(\mathbf{x}|\theta)} \right] + K \\ &= -\mathbb{E}_{p_D(\mathbf{x})} [\log p(\mathbf{x}|\theta)] + K \\ &= -\frac{1}{N} \sum_{n=1}^N \log p(\mathbf{x}_n|\theta) + K\end{aligned}$$

- Therefore,

$$\hat{\theta}^* = \arg \min_{\theta} \text{KL}[p_D(\mathbf{x})||p(\mathbf{x}|\theta)] \approx \arg \max_{\theta} \sum_{n=1}^N \log p(\mathbf{x}_n|\theta) \equiv \hat{\theta}_{\text{MLE}}$$

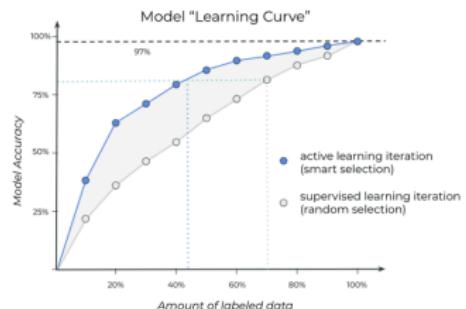
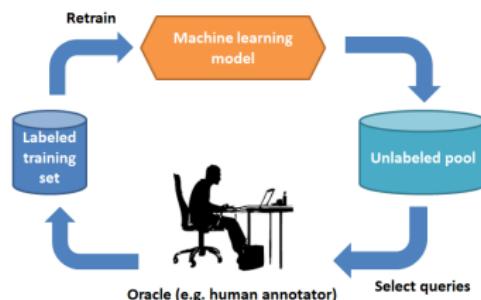
- Asymptotically ($N \rightarrow \infty$), *maximum likelihood learning* is equivalent to minimizing the *KL divergence between the true data distribution $p_D(\mathbf{x})$ and your model $p(\mathbf{x}|\theta)$* .

De-tour: Information gain

- Consider some Bayesian model with prior $p(\mathbf{w})$ and posterior $p(\mathbf{w}|\mathcal{D})$
- The KL divergence between the posterior and the prior is also sometimes called *information gain*

$$\text{KL}[p(\mathbf{w}|\mathcal{D})||p(\mathbf{w})]$$

- ... and quantifies how much *information* we gain by moving from the prior to the posterior
- Often used in *active learning* and *optimal experiment design* to choose the most informative subset of data for more data-efficient learning.



Li et al, 2018: Reversed Active Learning based Atrous DenseNet for Pathological Image Classification

<https://www.kdnuggets.com/2018/10/introduction-active-learning.html>

Black-box variational inference

Free-form and fixed-form variational inference

- In *free-form* VI, we use a *factorized approximation* for approximating the target distribution $p \equiv p(\mathbf{w}|\mathbf{y})$

$$q(\mathbf{w}) = \prod_{j=1}^J q(\mathbf{w}_j), \quad \text{where } \mathbf{w} = [\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_J]$$

- In *fixed-form* VI we choose a specific family distributions, e.g.

$$q(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mathbf{m}, \mathbf{V})$$

- ... and then evaluate and optimize the ELBO wrt. $q(\mathbf{w})$

$$\mathcal{L}[q] = \mathbb{E}_q[\log p(\mathbf{y}, \mathbf{w})] - \mathbb{E}_q[\log q(\mathbf{w})]$$

- | | |
|---|--|
| + Optimal functional form given assumptions
(CAVI) | - Requires model-specific derivations |
| + Fast optimization | - Required integrals may be intractable |
| | - Optimal forms may not be "known" distributions |

Black-box variational inference

- How can we avoid the need for model-specific derivations? How can we avoid the restrictions on the choice of models we can approximate?
- Again, we use *fixed-form* families for continuous parameters, e.g. *mean-field* or *full-rank* Gaussians

$$q_{\text{MF}}(\mathbf{w}) = \prod_{i=1}^D \mathcal{N}(w_i | m_i, v_i) \quad q_{\text{FR}}(\mathbf{w}) = \mathcal{N}(\mathbf{w} | \mathbf{m}, \mathbf{V})$$

- Optimize ELBO wrt. *variational parameters* $\lambda = \{\mathbf{m}, \mathbf{V}\}$

$$\lambda^* = \arg \max_{\lambda} \mathcal{L}[q_{\lambda}] = \arg \max_{\lambda} \{ \mathbb{E}_{q_{\lambda}} [\log p(\mathbf{y}, \mathbf{w})] - \mathbb{E}_{q_{\lambda}} [\log q_{\lambda}(\mathbf{w})] \}$$

- We want to find an "automatic" method for
 1. evaluating $\mathcal{L}[q_{\lambda}]$
 2. computing gradients of $\mathcal{L}[q_{\lambda}]$
- ... allowing us to easily prototype, implement and test different models

Black-box variational inference: The evidence lower bound and entropy

The evidence lower bound and entropy

- The evidence lower bound $\mathcal{L}[q]$ is defined as

$$\mathcal{L}[q] = \mathbb{E}_q [\log p(\mathbf{y}, \mathbf{w})] - \mathbb{E}_q [\log q(\mathbf{w})] = \mathbb{E}_q [\log p(\mathbf{y}, \mathbf{w})] + \mathcal{H}[q]$$

- Second term is called the (differential) *entropy* of q , i.e. $\mathcal{H}[q] \equiv -\mathbb{E}_q[\log q(\mathbf{w})]$
- For "named" distributions, like Gaussians, the entropy is often easy to calculate or can be looked up.
- If $q(w_i) = \mathcal{N}(w_i | m_i, v_i)$, then

$$\mathcal{H}[q(w_i)] \equiv -\mathbb{E}_{q(w_i)}[\log q(w_i)] = \frac{1}{2} \log(2\pi e v_i)$$

- The entropy term and its gradient can often be computed analytically
- Therefore, we will focus on dealing with the expected log joint term: $\mathbb{E}_q [\log p(\mathbf{y}, \mathbf{w})]$

Quiz

Quiz time!

Spend 4 minutes answering the following questions in the quiz

DTU Learn - Quiz - Lecture12: Variational families

Black-box variational inference: Dealing with expectations

How to deal with the expected log joint

- We can approximate the first term using Monte Carlo samples:

$$\mathbb{E}_q [\log p(\mathbf{y}, \mathbf{w})] \approx \frac{1}{S} \sum_{s=1}^S \log p(\mathbf{y}, \mathbf{w}^{(s)}),$$

where

$$\mathbf{w}^{(s)} \sim q_{\lambda}(\mathbf{w}) \quad \text{for } s = 1, \dots, S$$

- Example: if $q_{\lambda}(\mathbf{w}) = \mathcal{N}(\mathbf{w} | \mathbf{m}, \mathbf{V})$, we sample $\mathbf{w}^{(s)} \sim \mathcal{N}(\mathbf{w} | \mathbf{m}, \mathbf{V})$ and plug them into the sum above
- Easy to implement and fits the bill
 1. All we need is to be able to evaluate the log joint $p(\mathbf{y}, \mathbf{w})$
 2. Not restricted by existence of closed-form analytical results for expectations
 3. No model-specific derivations
- How about the gradients?

$$\nabla_{\lambda} \mathbb{E}_{q_{\lambda}} [\log p(\mathbf{y}, \mathbf{w})]$$

Black-box variational inference: Stochastic gradients

How to compute the gradients?

- First, we can use *Leibniz'* rule to change order of the gradient and the integral

$$\begin{aligned}\nabla_{\lambda} \mathbb{E}_{q_{\lambda}} [\log p(\mathbf{y}, \mathbf{w})] &\equiv \nabla_{\lambda} \int q_{\lambda}(\mathbf{w}) \log p(\mathbf{y}, \mathbf{w}) d\mathbf{w} \\ &= \int \nabla_{\lambda} q_{\lambda}(\mathbf{w}) \log p(\mathbf{y}, \mathbf{w}) d\mathbf{w}\end{aligned}$$

- We can't estimate the gradients directly using MC sampling. Why not?
- In order to estimate a quantity using MC, we need to able to express it as an expectation. For example,

$$\mathbb{E}_q [\log p(\mathbf{y}, \mathbf{w})] = \int q(\mathbf{w}) \log p(\mathbf{y}, \mathbf{w}) d\mathbf{w} \approx \frac{1}{S} \sum_{s=1}^S \log p(\mathbf{y}, \mathbf{w}^{(s)}),$$

- To generate samples, we need a distribution, but $\nabla_{\lambda} q_{\lambda}(\mathbf{w})$ is not a distribution

The score function estimator

- **Log-derivative trick:** Using the chain-rule on the $\log q_\lambda(\mathbf{w})$ yields

$$\nabla_\lambda \log q_\lambda(\mathbf{w}) = \frac{1}{q_\lambda(\mathbf{w})} \nabla_\lambda q_\lambda(\mathbf{w})$$

- We can use this to re-write the gradient

$$\begin{aligned}\nabla_\lambda \mathbb{E}_{q_\lambda} [\log p(\mathbf{y}, \mathbf{w})] &\equiv \int \nabla_\lambda q_\lambda(\mathbf{w}) \log p(\mathbf{y}, \mathbf{w}) d\mathbf{w} \\&= \int \frac{q_\lambda(\mathbf{w})}{q_\lambda(\mathbf{w})} \nabla_\lambda q_\lambda(\mathbf{w}) \log p(\mathbf{y}, \mathbf{w}) d\mathbf{w} \\&= \int q_\lambda(\mathbf{w}) \frac{1}{q_\lambda(\mathbf{w})} \nabla_\lambda q_\lambda(\mathbf{w}) \log p(\mathbf{y}, \mathbf{w}) d\mathbf{w} \\&= \int q_\lambda(\mathbf{w}) \nabla_\lambda \log q_\lambda(\mathbf{w}) \log p(\mathbf{y}, \mathbf{w}) d\mathbf{w} \quad (\text{log-derivative trick}) \\&= \mathbb{E}_{q_\lambda} [\nabla_\lambda \log q_\lambda(\mathbf{w}) \log p(\mathbf{y}, \mathbf{w})] \quad (\text{Def. of expectation}) \\&\approx \frac{1}{S} \sum_{s=1}^S \nabla_\lambda \log q_\lambda(\mathbf{w}^s) \log p(\mathbf{y}, \mathbf{w}^s) \quad \text{where} \quad \mathbf{w}^s \sim q_\lambda\end{aligned}$$

- This is called the *score function gradient estimator*

The score function gradient estimator

- The score function gradient estimator is defined as

$$\nabla_{\lambda} \mathbb{E}_{q_{\lambda}} [\log p(\mathbf{y}, \mathbf{w})] \approx \frac{1}{S} \sum_{s=1}^S \nabla_{\lambda} \log q_{\lambda}(\mathbf{w}^s) \log p(\mathbf{y}, \mathbf{w}^s) \quad \text{where} \quad \mathbf{w}^s \sim q_{\lambda}(\mathbf{w})$$

- $\nabla_{\lambda} \log q_{\lambda}$ can easily be calculated by hand or using automatic differentiation (e.g. Pytorch, Tensorflow, Autograd etc.)
- Only need to be able to evaluate $\log p(\mathbf{y}, \mathbf{w})$. No need for model-specific derivations
- General properties
 1. Very general and unbiased
 2. Applies to both continuous and discrete parameters
 3. Often high variance
- Big picture: approximating posterior distributions of general models
 1. We have chosen a variational family
 2. We aim to maximize the ELBO using gradient-based methods
 3. Estimate gradients using the score function estimator without the need for model-specific derivations
 4. Drawback: stochastic gradients

The BBVI algorithm

Putting everything together

Given a model and variational family (e.g. mean-field Gaussians)

$$p(\mathbf{y}, \mathbf{w}) = p(\mathbf{y}|\mathbf{w})p(\mathbf{w}), \quad q(\mathbf{w}) = \prod_{i=1}^D \mathcal{N}(w_i|m_i, v_i)$$

Black-box variational inference w. score function estimator

- Initialize variational parameters $\lambda_1 = \{\mathbf{m}_1, \log \mathbf{v}_1\}$
- For iteration $t = 1, \dots, T$ (or until convergence)
 1. Generate samples $\mathbf{w}^{(s)} \sim q(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mathbf{m}_t, \mathbf{v}_t)$ for $s = 1, \dots, S$
 2. Estimate ELBO

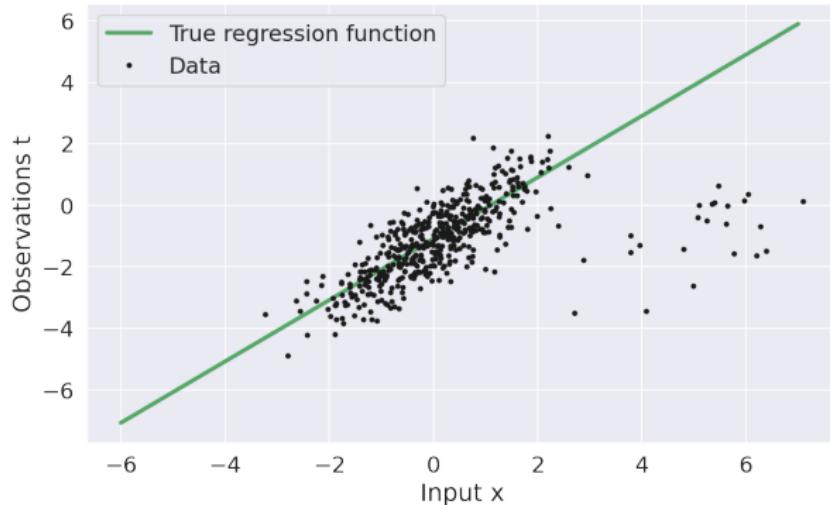
$$\hat{\mathcal{L}}[q] = \frac{1}{S} \sum_{s=1}^S \log p(\mathbf{y}, \mathbf{w}^{(s)}) + \mathcal{H}[q]$$

- 3. Estimate gradient

$$\nabla_{\lambda} \hat{\mathcal{L}}[q] \approx \frac{1}{S} \sum_{s=1}^S \nabla_{\lambda} \log q_{\lambda}(\mathbf{w}^s) \log p(\mathbf{y}, \mathbf{w}^s) + \nabla_{\lambda} \mathcal{H}[q]$$

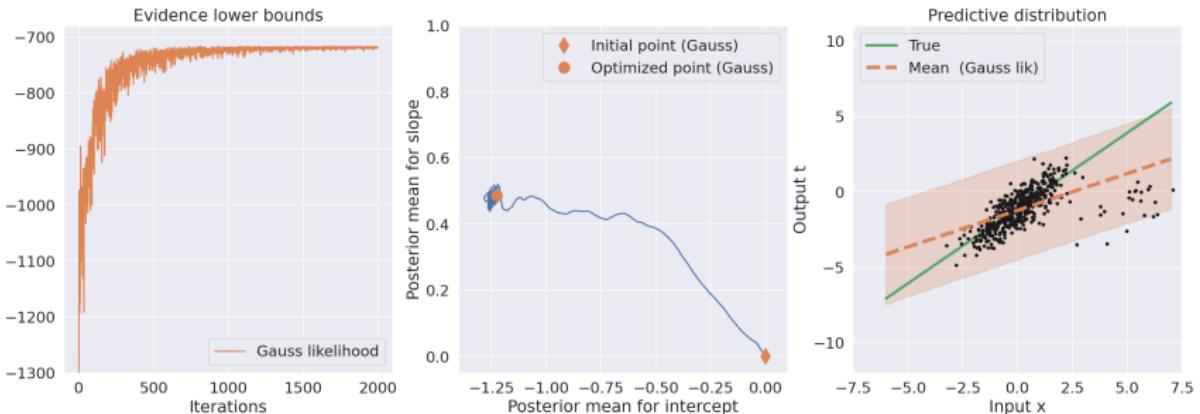
- 4. Update variational parameters to get $\lambda_{t+1} = \{\mathbf{m}_{t+1}, \log \mathbf{v}_{t+1}\}$ using gradient estimate
- Predictions & model checking: Evaluate training error, validation errors etc
- Go back and refine model if needed

Example: Robust regression I



$$p(\mathbf{y}, \mathbf{w}) = \prod_{n=1}^N \mathcal{N}(y_n | \mathbf{w}^T \mathbf{x}_i, \sigma^2) \mathcal{N}(\mathbf{w} | \mathbf{0}, \kappa^2 \mathbf{I})$$

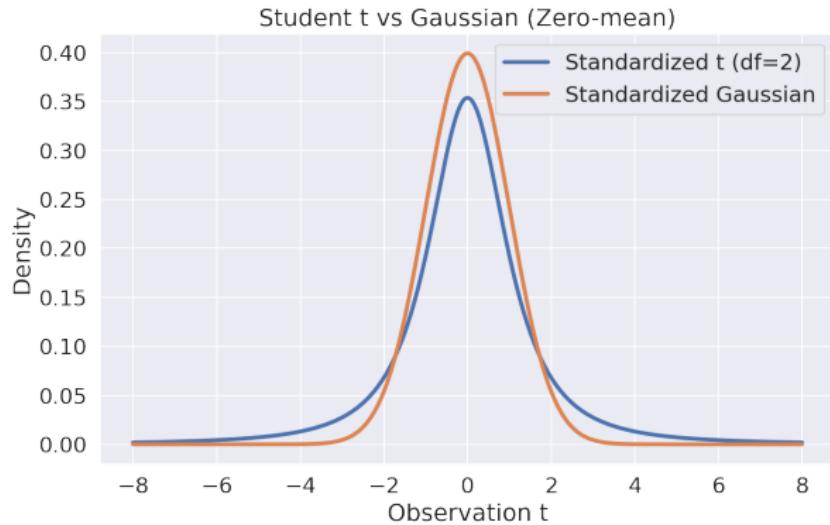
Example: Robust regression II



$$p(\mathbf{y}, \mathbf{w}) = \prod_{n=1}^N \mathcal{N}(y_n | \mathbf{w}^T \mathbf{x}_i, \sigma^2) \mathcal{N}(\mathbf{w} | \mathbf{0}, \kappa^2 \mathbf{I})$$

- Running BBVI with $S = 5$ samples for 2000 iterations

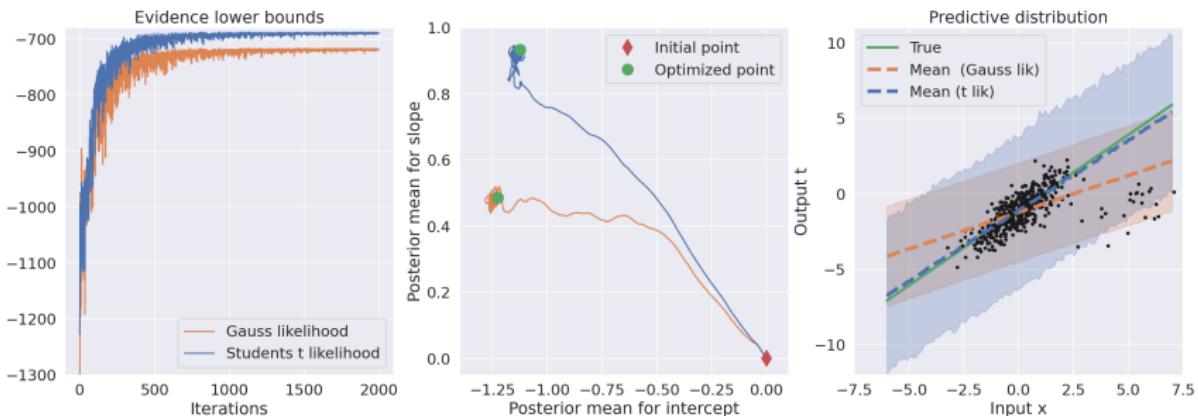
Example: Robust regression III



$$p_{\text{gauss}}(\mathbf{y}, \mathbf{w}) = \prod_{n=1}^N \mathcal{N}(y_n | \mathbf{w}^T \mathbf{x}_i, \sigma^2) \mathcal{N}(\mathbf{w} | \mathbf{0}, \kappa^2 \mathbf{I})$$

$$p_{\text{student-t}}(\mathbf{y}, \mathbf{w}) = \prod_{n=1}^N t_2(y_n | \mathbf{w}^T \mathbf{x}_i, \sigma^2) \mathcal{N}(\mathbf{w} | \mathbf{0}, \kappa^2 \mathbf{I})$$

Example: Robust regression IV



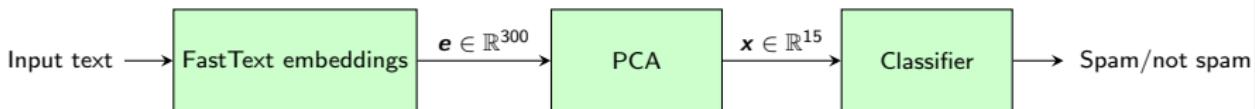
$$p_{\text{gauss}}(\mathbf{y}, \mathbf{w}) = \prod_{n=1}^N \mathcal{N}(y_n | \mathbf{w}^T \mathbf{x}_i, \sigma^2) \mathcal{N}(\mathbf{w} | \mathbf{0}, \kappa^2 \mathbf{I})$$

$$p_{\text{student-t}}(\mathbf{y}, \mathbf{w}) = \prod_{n=1}^N t_2(y_n | \mathbf{w}^T \mathbf{x}_i, \sigma^2) \mathcal{N}(\mathbf{w} | \mathbf{0}, \kappa^2 \mathbf{I})$$

- Running BBVI with $S = 5$ samples for 2000 iterations

Text classification: Spam detection

- In the exercise notebook, we will build a simple spam detector based on an SMS dataset
- Text examples
 1. "urgent! your mobile number has been awarded with a £2000 prize guaranteed. call 09061790121 from land line. claim 3030. valid 12hrs only 150ppm"
 2. "do we have any spare power supplies"
 3. "merry christmas to u too annie!"
- Transfer learning: Embed texts using FastText embedding (www.fasttext.cc)



- Which likelihood to use to gain robustness to outliers?

$$p(\mathbf{y}, \mathbf{w}) = \prod_{n=1}^N p(y_n | \mathbf{x}_n, \mathbf{w}) \mathcal{N}(\mathbf{w} | \mathbf{0}, \alpha^{-1} \mathbf{I})$$

Black-box variational inference: A few words on stochastic optimization

Simple example: Optimizing $J(\lambda)$

- Suppose $q(\mathbf{w}) = \mathcal{N}(\mathbf{w} | \boldsymbol{\lambda}, \mathbf{I})$ and $\log p(\mathbf{y}, \mathbf{w}) = -\mathbf{w}^T \mathbf{w}$

$$J(\boldsymbol{\lambda}) = \mathbb{E}_{q_{\boldsymbol{\lambda}}} [\log p(\mathbf{y}, \mathbf{w})] = \mathbb{E}_{q_{\boldsymbol{\lambda}}} [-\mathbf{w}^T \mathbf{w}]$$

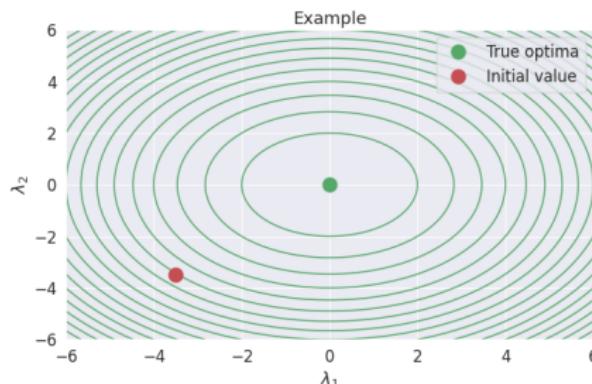
- We optimize J using gradient ascent with step-size $\rho = 0.1$

$$\boldsymbol{\lambda}^{t+1} = \boldsymbol{\lambda}^t + \rho \hat{\mathbf{g}}(\boldsymbol{\lambda}^t)$$

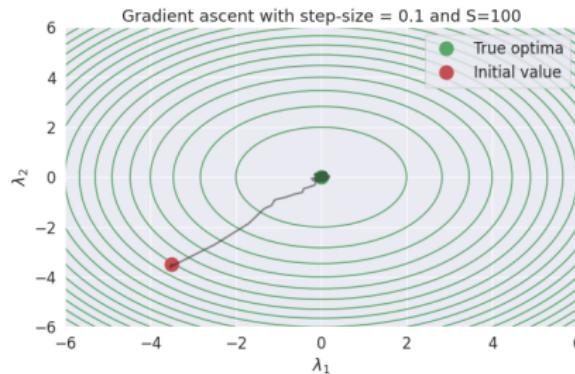
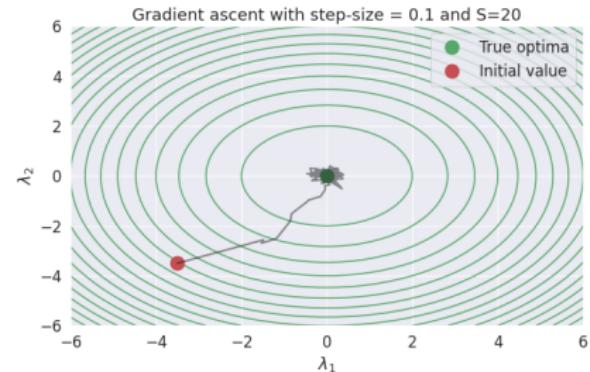
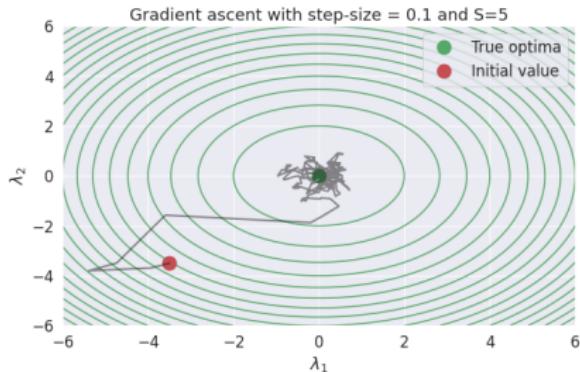
- Using the score function gradient estimator

$$\hat{\mathbf{g}} = \frac{1}{S} \sum_{s=1}^S \nabla_{\boldsymbol{\lambda}} \log q_{\boldsymbol{\lambda}}(\mathbf{w}_s) \log p(\mathbf{y}, \mathbf{w}_s) = \frac{1}{S} \sum_{s=1}^S 2(\mathbf{w}_s - \boldsymbol{\lambda}) \mathbf{w}_s^T \mathbf{w}_s$$

where $\mathbf{w}_s \sim q_{\boldsymbol{\lambda}}(\mathbf{w})$



Simple example: Optimizing $J(\lambda)$ II



- Gradient ascent does not converge with constant step size
- The number of samples S trades off gradient variance vs computational cost

Stochastic gradient descent (SGD)

- The gradient ascent update with *constant step-size* η

$$\lambda^{t+1} = \lambda^t + \eta \hat{g}(\lambda^t)$$

- In *stochastic gradient ascent* the step-size is decreased in each iterations

$$\lambda^{t+1} = \lambda^t + \rho_t \eta \hat{g}(\lambda^t)$$

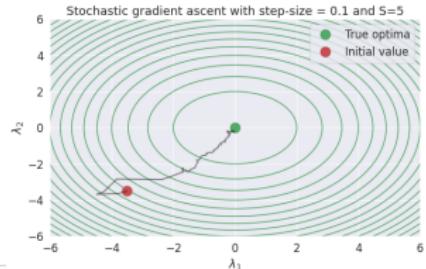
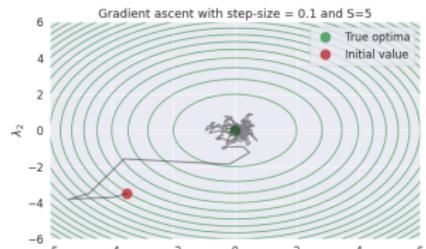
- The *Robbins-Monro conditions* guarantees convergence in probability if the gradient estimator is unbiased (and under regularity conditions)

$$\sum_{t=1}^{\infty} \rho_t = \infty \quad \text{and} \quad \sum_{t=1}^{\infty} \rho_t^2 < \infty$$

- Example: $\tau > 0$ and $\kappa \in (0.5, 1]$

$$\rho_t = (t + \tau)^{-\kappa}$$

Robbins & Monro (1951): A stochastic approximation method



A zoo of stochastic optimizers

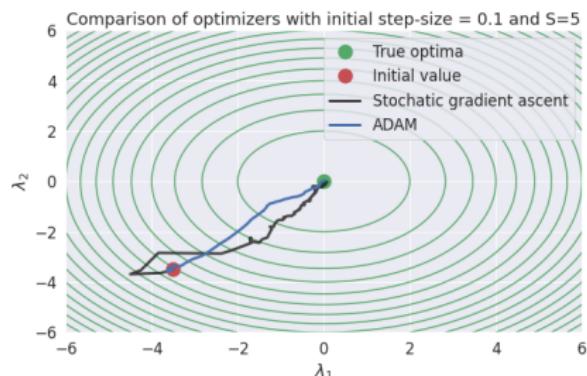
- Many recent methods for stochastic optimization:
Adam, RMSProp, AdaGrad, AdaDelta etc
- Several key ideas for faster and more robust optimization
- **Momentum:** use gradient from previous iteration
(rolling ball analogy)

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$$
$$\lambda_{t+1} = \lambda_t + \eta m_t$$

- **Adaptive learning rates:** Adaptive learning rate based on magnitude of gradient

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$$
$$\lambda_{t+1} = \lambda_t + \eta \frac{m_t}{\sqrt{v_t} + \epsilon}$$

- **Individual learning:** Each parameter is allowed to have its own adaptive learning rate
- The ADAM optimizer combines all these ideas (along with a bias-correction) and is often a good first choice



Quiz time

Quiz time!

DTU Learn - Quiz - Lecture 12 - BBVI quiz

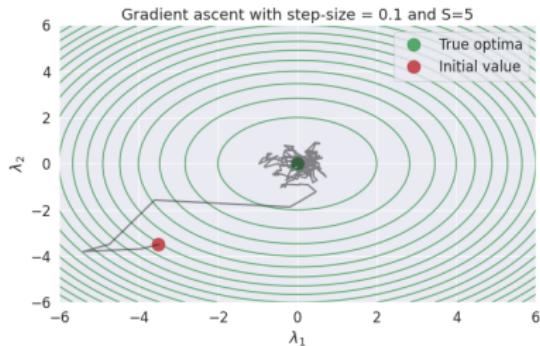
The re-parametrization trick: gradient estimator with lower variance

- Gradient estimators with lower variance leads to faster optimization, because *higher variance requires smaller step-sizes*. Often we can do better than the score function gradient
- If $q(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mathbf{m}, \mathbf{V})$ is Gaussian and if $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, then $\mathbf{w} = \mathbf{m} + \mathbf{L}\epsilon \sim \mathcal{N}(\mathbf{w}|\mathbf{m}, \mathbf{V})$, where \mathbf{L} is the Cholesky factorization (matrix square root) of $\mathbf{V} = \mathbf{L}\mathbf{L}^T$.
- We *re-parametrize* \mathbf{w} as $\mathbf{w}_\epsilon = g(\boldsymbol{\lambda}, \epsilon) = \mathbf{m} + \mathbf{L}\epsilon$

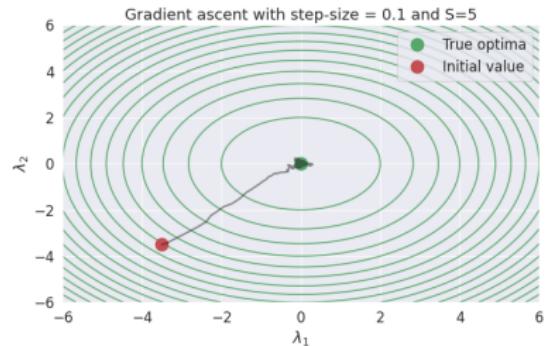
$$\begin{aligned}\nabla_{\boldsymbol{\lambda}} \mathbb{E}_{q_{\boldsymbol{\lambda}}(\mathbf{w})} [\log p(\mathbf{y}, \mathbf{w})] &= \nabla_{\boldsymbol{\lambda}} \mathbb{E}_{q(\epsilon)} [\log p(\mathbf{y}, \mathbf{w}_\epsilon)] && \text{(Re-parametrization)} \\ &= \nabla_{\boldsymbol{\lambda}} \int q(\epsilon) \log p(\mathbf{y}, \mathbf{w}_\epsilon) d\epsilon && \text{(Def. of expectation)} \\ &= \int q(\epsilon) \nabla_{\boldsymbol{\lambda}} \log p(\mathbf{y}, \mathbf{w}_\epsilon) d\epsilon && \text{(Leibniz' rule)} \\ &= \mathbb{E}_{q(\epsilon)} [\nabla_{\boldsymbol{\lambda}} \log p(\mathbf{y}, \mathbf{w}_\epsilon)] && \text{(Def. of expectation)} \\ &\approx \frac{1}{S} \sum_{s=1}^S \nabla_{\boldsymbol{\lambda}} \log p(\mathbf{y}, \mathbf{w}_{\epsilon_s}) && \text{where } \epsilon_s \sim q(\epsilon)\end{aligned}$$

- The *re-parametrized gradient estimator* requires the transformation g and $p(\mathbf{y}, \mathbf{w})$ to be differentiable, which rules out discrete variables.

Simple example revisited



Score function gradient estimator



Re-parametrized gradient estimator

- Comparison of the score function gradient estimator and the re-parametrized gradient estimator
- Using gradient ascent with fixed step-size for the purpose of illustration
- Score function is more general, but suffers from larger variance
- The re-parametrized gradient is less general and applies only to continuous variables, but has much lower variance

Reparametrized gradients

- Many variational families can be re-parametrized

Target	$p(z; \theta)$	Base $p(\epsilon)$	One-liner $g(\epsilon; \theta)$
Exponential	$\exp(-x); x > 0$	$\epsilon \sim [0; 1]$	$\ln(1/\epsilon)$
Cauchy	$\frac{1}{\pi(1+x^2)}$	$\epsilon \sim [0; 1]$	$\tan(\pi\epsilon)$
Laplace	$\mathcal{L}(0; 1) = \exp(- x)$	$\epsilon \sim [0; 1]$	$\ln(\frac{\epsilon_1}{\epsilon_2})$
Laplace	$\mathcal{L}(\mu; b)$	$\epsilon \sim [0; 1]$	$\mu - b sgn(\epsilon) \ln(1 - 2 \epsilon)$
Std Gaussian	$\mathcal{N}(0; 1)$	$\epsilon \sim [0; 1]$	$\sqrt{\ln(\frac{1}{\epsilon_1})} \cos(2\pi\epsilon_2)$
Gaussian	$\mathcal{N}(\mu; RR^\top)$	$\epsilon \sim \mathcal{N}(0; 1)$	$\mu + R\epsilon$
Rademacher	$Rad(\frac{1}{2})$	$\epsilon \sim Bern(\frac{1}{2})$	$2\epsilon - 1$
Log-Normal	$\ln \mathcal{N}(\mu; \sigma)$	$\epsilon \sim \mathcal{N}(\mu; \sigma^2)$	$\exp(\epsilon)$
Inv Gamma	$i\mathcal{G}(k; \theta)$	$\epsilon \sim \mathcal{G}(k; \theta^{-1})$	$\frac{1}{\epsilon}$

Black-box variational inference: Mini-batching for large scale inference

Mini-batching for large scale inference

- Assuming a model of the form

$$p(\mathbf{y}, \mathbf{w}) = \prod_{n=1}^N p(y_n | \mathbf{w}) p(\mathbf{w})$$

- Let $\mathbf{w}^s \sim q_\lambda$ be samples from the variational approximation, then

$$\begin{aligned}\mathcal{L}[q] &\approx \frac{1}{S} \sum_{s=1}^S [\log p(\mathbf{y}, \mathbf{w}^s)] + \mathcal{H}[q] \\ &= \frac{1}{S} \sum_{s=1}^S \left[\sum_{n=1}^N \log p(y_n, \mathbf{w}^s) + \log p(\mathbf{w}^s) \right] + \mathcal{H}[q]\end{aligned}$$

- Evaluating the ELBO and its gradients becomes slow for large N

- Mini-batching:* Let \mathcal{M} be a random sample of the data points of size M , then the following is an unbiased estimator of the ELBO

$$\mathcal{L}[q] \approx \frac{1}{S} \sum_{s=1}^S \left[\frac{N}{M} \sum_{n \in \mathcal{M}} \log p(y_n, \mathbf{w}^s) + \log p(\mathbf{w}^s) \right] + \mathcal{H}[q]$$

- We can use the same ideas to speed up the calculations of the gradients, but the price is increased variance

Summary

Putting everything together

Given a model and variational family (e.g. mean-field Gaussians)

$$p(\mathbf{y}, \mathbf{w}) = p(\mathbf{y}|\mathbf{w})p(\mathbf{w}), \quad q(\mathbf{w}) = \prod_{i=1}^D \mathcal{N}(w_i|m_i, v_i)$$

Black-box variational inference w. the re-parametrized gradient

- Initialize variational parameters $\lambda_1 = \{\mathbf{m}_1, \log \mathbf{v}_1\}$
- For iteration $t = 1, \dots, T$ (or until convergence)
 1. Generate samples $\epsilon_s \sim \mathcal{N}(\epsilon|0, I)$ for $s = 1, \dots, S$
 2. Estimate ELBO (use mini-batching if necessary)

$$\hat{\mathcal{L}}[q] = \frac{1}{S} \sum_{s=1}^S \log p(\mathbf{y}, \mathbf{w}_{\epsilon_s}) + \mathcal{H}[q], \quad \text{where } \mathbf{w}_{\epsilon_s} = \mathbf{m}_t + \mathbf{L}_t \epsilon_s$$

- 3. Estimate gradient (use mini-batching if necessary)

$$\nabla_{\lambda} \hat{\mathcal{L}}[q] \approx \frac{1}{S} \sum_{s=1}^S \nabla_{\lambda} \log p(\mathbf{y}, \mathbf{w}_{\epsilon_s}) + \nabla_{\lambda} \mathcal{H}[q]$$

- 4. Update variational parameters to get $\lambda_{t+1} = \{\mathbf{m}_{t+1}, \log \mathbf{v}_{t+1}\}$ using gradient estimate
- Predictions & model checking: Evaluate training error, validation errors etc
- Go back and refine model if needed

02477 – Bayesian Machine Learning: Lecture 13

Michael Riis Andersen

Technical University of Denmark,
DTU Compute, Department of Applied Math and Computer Science

Outline

1 Wrapping up VI

- Variational autoencoders
- Summary

2 Bayesian Neural Networks

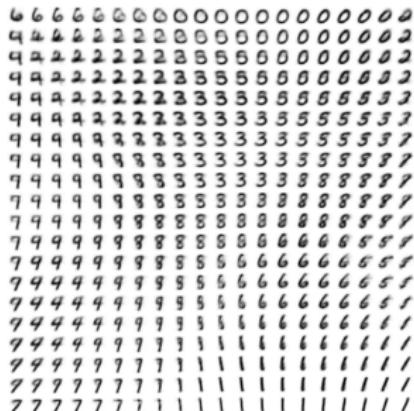
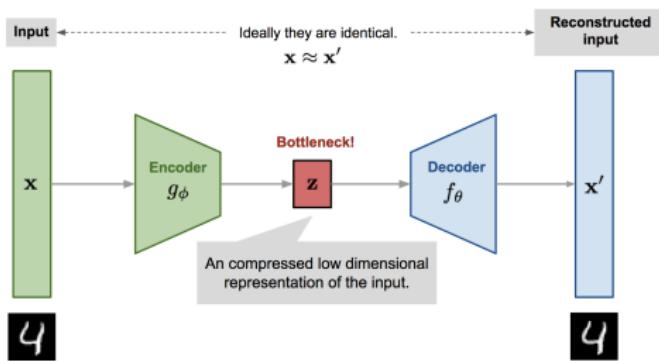
- Motivation and challenges
- Approximating the posterior for neural networks
- Monte Carlo Dropout
- Mean-field
- Laplace approximations
- Deep Ensembles

Wrapping up VI

Wrapping up VI: Variational autoencoders

Variational autoencoders (VAEs)

How does VAEs related to all this?



Variational auto-encoders

- Unsupervised learning
- Finds low-dimensional latent representation $z \in \mathbb{R}^K$ for data $x \in \mathbb{R}^D$ (encoder)
- Generative model: Can sample z and transform back to data-space (decoder)

<https://lilianweng.github.io/posts/2018-08-12-vae/>

Kingma & Welling, 2014: Auto-encoding variational Bayes

Factor models: PCA and pPCA

- Principal component analysis

$$\mathbf{x}_n \approx \mathbf{W}\mathbf{z}_n$$

- Probabilistic PCA (PPCA)

$$\mathbf{x}_n | \mathbf{z}_n \sim \mathcal{N}(\mathbf{W}\mathbf{z}_n, \sigma^2 \mathbf{I})$$

$$\mathbf{z}_n \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

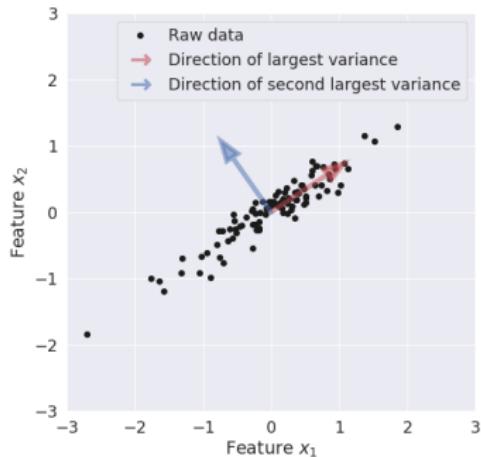
- Properties of PPCA

1. Linear Gaussian model
2. Conjugate model: easy inference
3. Generative model: can sample new data points
4. Can handle missing data
5. Enables probabilistic reasoning

- Replacing the linear model with a neural network $f_\theta(\mathbf{z}) = \text{NN}_\theta(\mathbf{z})$

$$\mathbf{x}_n | \mathbf{z}_n \sim \mathcal{N}(f_\theta(\mathbf{z}_n), \sigma^2 \mathbf{I})$$

$$\mathbf{z}_n \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$



Amortized variational inference

- Non-linear factor model

$$\begin{aligned}\mathbf{x}_n | \mathbf{z}_n &\sim \mathcal{N}(f_{\theta}(\mathbf{z}_n), \sigma^2 \mathbf{I}) \\ \mathbf{z}_n &\sim \mathcal{N}(\mathbf{0}, \mathbf{I})\end{aligned}$$

- No longer conjugate: variational inference to the rescue

$$q(\mathbf{Z}) = \prod_{n=1}^N q(\mathbf{z}_n) \quad q(\mathbf{z}_n) = \prod_{k=1}^K \mathcal{N}(z_{nk} | m_{nk}, v_{nk})$$

- *Amortized inference*: Avoiding the number of variational parameters growing with the number of data points

$$q(\mathbf{z}_n) = \prod_{k=1}^K \mathcal{N}(z_{nk} | m(\mathbf{x}_n), v(\mathbf{x}_n))$$

where m and v controlled by another neural network g_{ϕ} with two outputs

$$\begin{aligned}m(\mathbf{x}_n) &= g_{\phi}^1(\mathbf{x}_n) \\ \ln v(\mathbf{x}_n) &= g_{\phi}^2(\mathbf{x}_n)\end{aligned}$$

Variational autoencoders

How does VAEs relates to BBVI?

- Probabilistic model (*decoder*)

$$\mathbf{x}_n | \mathbf{z}_n \sim \mathcal{N}(f_\theta(\mathbf{z}_n), \sigma^2 \mathbf{I})$$

$$\mathbf{z}_n \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

- Variational approximation (*encoder*)

$$q(\mathbf{z}_n) = \prod_{k=1}^K \mathcal{N}(z_{nk} | m(\mathbf{x}_n), v(\mathbf{x}_n))$$

$$m(\mathbf{x}_n) = g_\phi^1(\mathbf{x}_n)$$

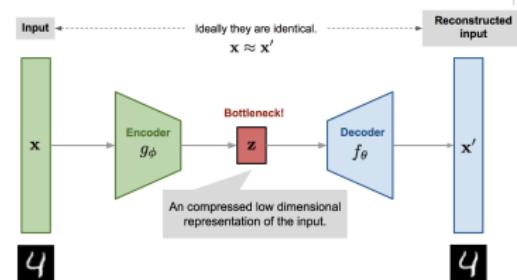
$$\ln v(\mathbf{x}_n) = g_\phi^2(\mathbf{x}_n)$$

- Fit model by optimizing the ELBO wrt. ϕ, θ (NN parameters)

$$\arg \max_{\theta, \phi} \mathcal{L}[q]$$

- Reparametrization trick to estimate gradients

- VAE = Non-linear factor model + variational inference



Credit: <https://lilianweng.github.io/posts/2018-08-12-vae/>

Wrapping up VI: Summary

Wrap-up of variational inference I

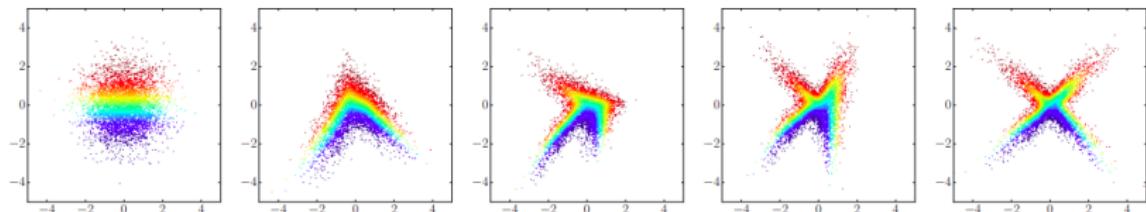
Variational inference

1. Flexible tool for approximating posterior for many models (from linear model to neural networks)
2. Better control of accuracy vs speed trade-off compared to Laplace approx. and MCMC
3. No strong theoretical guarantees as in MCMC
4. Can be scaled to huge datasets using mini batching
5. BBVI is the basis for variational inference in Tensorflow, Pytorch, Pyro, Stan etc.

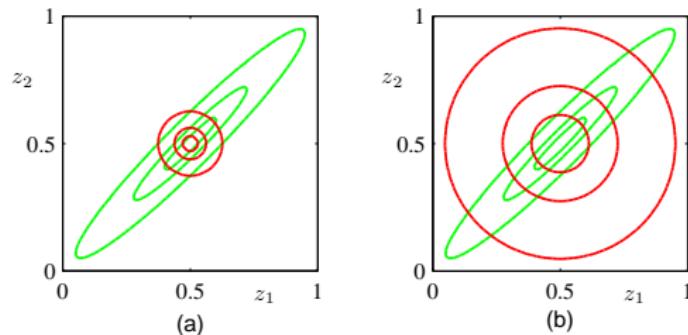
Active research area: how to improve VI?

Improving variational inference is a very active area of research. Examples

Constructing complex distributions using normalizing flows (Papamakarios et al 2021)



Investigating other divergences ($\text{KL}[q||p]$, $\text{KL}[p||q]$, α -divergences, χ^2 -divergences, f -divergences)



Which variational families are most suitable for high-dimensional posterior of NNs?

Bayesian Neural Networks

Neural Network notation

- Sequence of linear (affine) and non-linear mappings from input \mathbf{x} to output y
- Two-layer neural network (NN) with single output

$$\mathbf{z}_1 = h(\mathbf{W}_0 \mathbf{x} + \mathbf{b}_0)$$

$$\mathbf{z}_2 = h(\mathbf{W}_1 \mathbf{z}_1 + \mathbf{b}_1)$$

$$f = \mathbf{W}_2 \mathbf{z}_2 + \mathbf{b}_2$$

- All parameters of a network with L hidden layers

$$\mathbf{w} = \{\mathbf{W}_0, \mathbf{b}_0, \mathbf{W}_1, \mathbf{b}_1, \dots, \mathbf{W}_L, \mathbf{b}_L\}$$

- From input to output

$$f_{\mathbf{w}}(\mathbf{x}) = \mathbf{W}_2 h(\mathbf{W}_1 h(\mathbf{W}_1 \mathbf{x} + \mathbf{b}_0) + \mathbf{b}_1) + \mathbf{b}_2$$

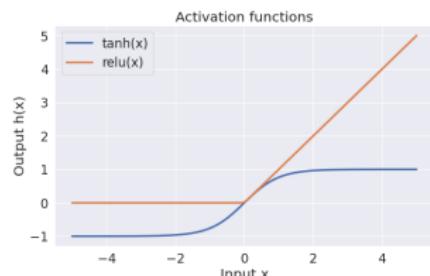
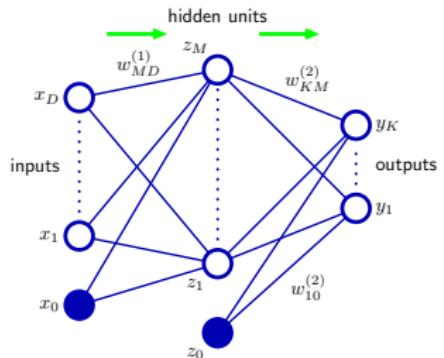
- NNs in probabilistic modelling

$$p(y_n | \mathbf{w}) = \mathcal{N}(y_n | f_{\mathbf{w}}(\mathbf{x}), \sigma^2)$$

$$p(y_n | \mathbf{w}) = \text{Ber}(y_n | \sigma(f_{\mathbf{w}}(\mathbf{x})))$$

- Completing the model with a prior over the weights, e.g.

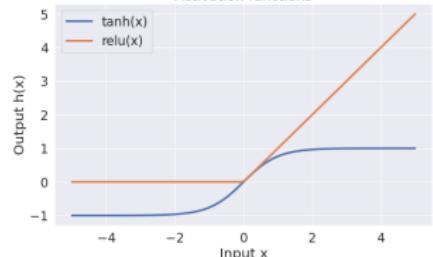
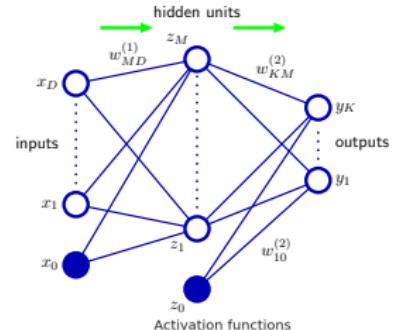
$$p(\mathbf{w}) = \mathcal{N}(\mathbf{w} | \mathbf{0}, \alpha^{-1} \mathbf{I})$$



Bayesian Neural Networks: Motivation and challenges

Bayesian Neural Networks: Motivation

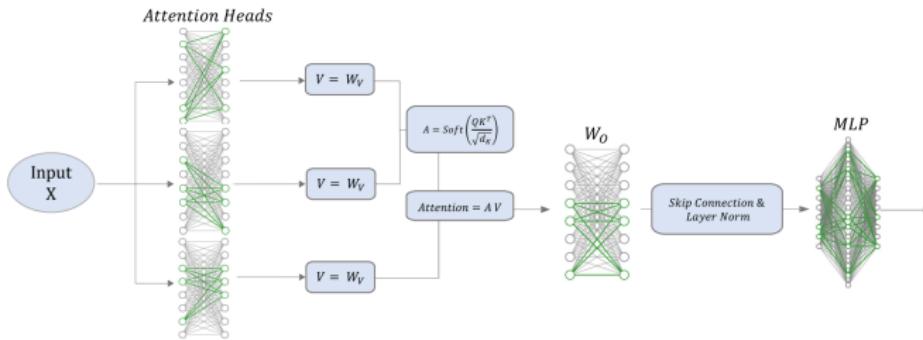
- Bayesian deep learning is a very active research area
- Why Bayesian neural networks?
 1. Predictive performance
 2. Epistemic uncertainty quantification
 3. Better calibration
 4. Sequential updating
 5. Less prone to overfitting
 6. Side step pathologies with maximum likelihood learning
 7. Active learning & data efficiency



$$\begin{aligned} z_1 &= h(\mathbf{W}_0 \mathbf{x} + \mathbf{b}_0) \\ z_2 &= h(\mathbf{W}_1 z_1 + \mathbf{b}_1) \\ f &= \mathbf{W}_2 z_2 + \mathbf{b}_2 \end{aligned}$$

Bayesian Neural Networks: Motivation

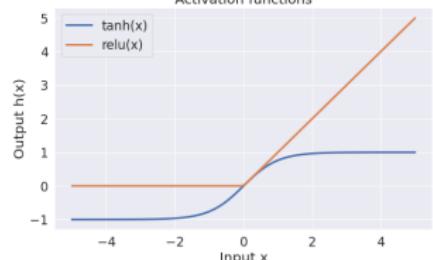
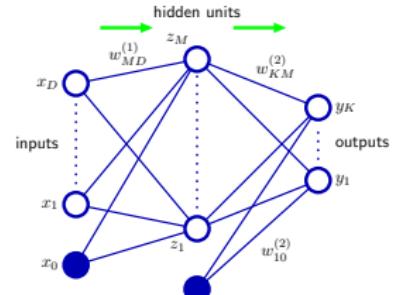
- On-going research: Can Bayesian methods to improve Transformers for NLP tasks?
- General Language Understanding Evaluation (GLUE) tasks: Stanford Sentiment Treebank (SST-2), Microsoft Research Paraphrase Corpus (MRPC), Recognizing Textual Entailment (RTE)



Methods	SST-2		MRPC		RTE	
	NLL↓	ECE↓	NLL↓	ECE↓	NLL↓	ECE↓
MAP	0.74 ± 0.10	0.11 ± 0.01	0.57 ± 0.34	0.14 ± 0.04	1.71 ± 0.23	0.33 ± 0.05
Temp. Scaled	0.36 ± 0.02	0.06 ± 0.00	0.44 ± 0.01	0.05 ± 0.02	0.66 ± 0.01	0.13 ± 0.04
S-KFAC	0.28 ± 0.00	0.04 ± 0.01	0.37 ± 0.02	0.04 ± 0.00	0.65 ± 0.01	0.05 ± 0.01
Last Layer	0.33 ± 0.02	0.06 ± 0.00	0.42 ± 0.01	0.07 ± 0.01	0.67 ± 0.01	0.08 ± 0.03
Fully Stoch. (LA)	0.27 ± 0.01	0.03 ± 0.00	0.39 ± 0.01	0.06 ± 0.00	0.66 ± 0.01	0.06 ± 0.01
Fully Stoch. (SWAG)	0.27 ± 0.06	0.03 ± 0.03	0.62 ± 0.11	0.07 ± 0.02	0.69 ± 0.00	0.05 ± 0.02

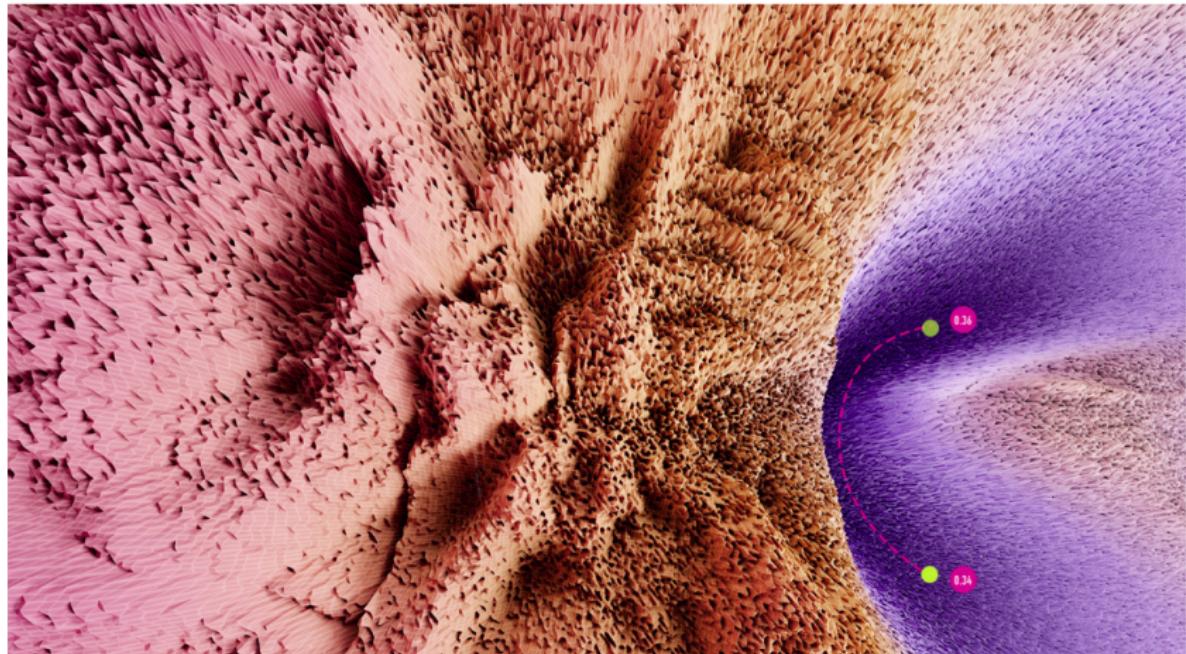
Bayesian Neural Networks: Challenges

- Posterior geometries of NNs are complicated!
 1. NNs are highly non-linear
 2. Posteriors are highly multi-modal
 3. NNs have weight-space symmetries
 4. Often underdetermined by the data
- Bayesian inference in neural networks is generally a really *difficult open problem*



$$\begin{aligned} z_1 &= h(\mathbf{W}_0 \mathbf{x} + \mathbf{b}_0) \\ z_2 &= h(\mathbf{W}_1 z_1 + \mathbf{b}_1) \\ f &= \mathbf{W}_2 z_2 + \mathbf{b}_2 \end{aligned}$$

Posterior geometries of NNs are complicated!



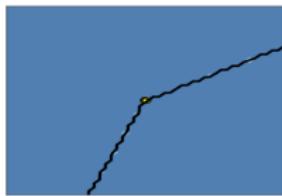
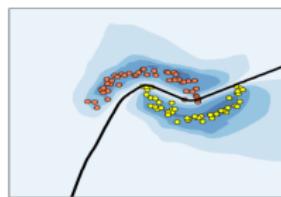
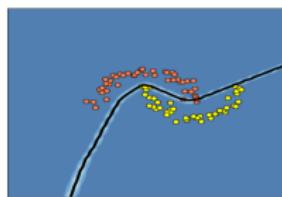
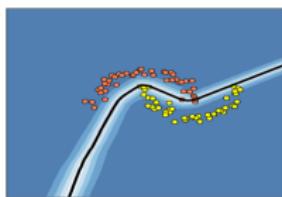
Visualization of loss surface for ResNet20

https://izmailovpavel.github.io/curves_blogpost/ (image credit)

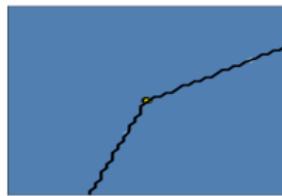
Uncertainty and overconfident neural networks

- If it is so difficult, why do we bother?
- Classical neural networks are very often *overconfident*
- ReLU networks can be made *arbitrary confident* far away from training data, e.g.

$$\lim_{\delta \rightarrow \infty} \text{softmax}(y(\delta \mathbf{x}))_i = 1 \quad \text{for some class } i$$



(a) MAP



(b) Temp. scaling



(c) Bayesian (last-layer)

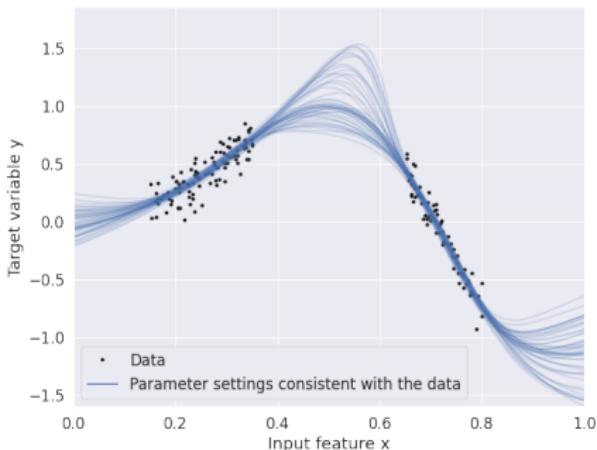
Model uncertainty due to finite data

- There exists many different parameter settings (for the same network), which lead to low training losses, but to very different predictions
- In classical training, we pick one set of weights (and one set of predictions)

$$\mathbf{w}_{\text{MAP}} = \arg \max_{\mathbf{w}} p(\mathbf{y}|\mathbf{w})p(\mathbf{w})$$

- In the Bayesian world, we want to compute a weighted averaged over all of them

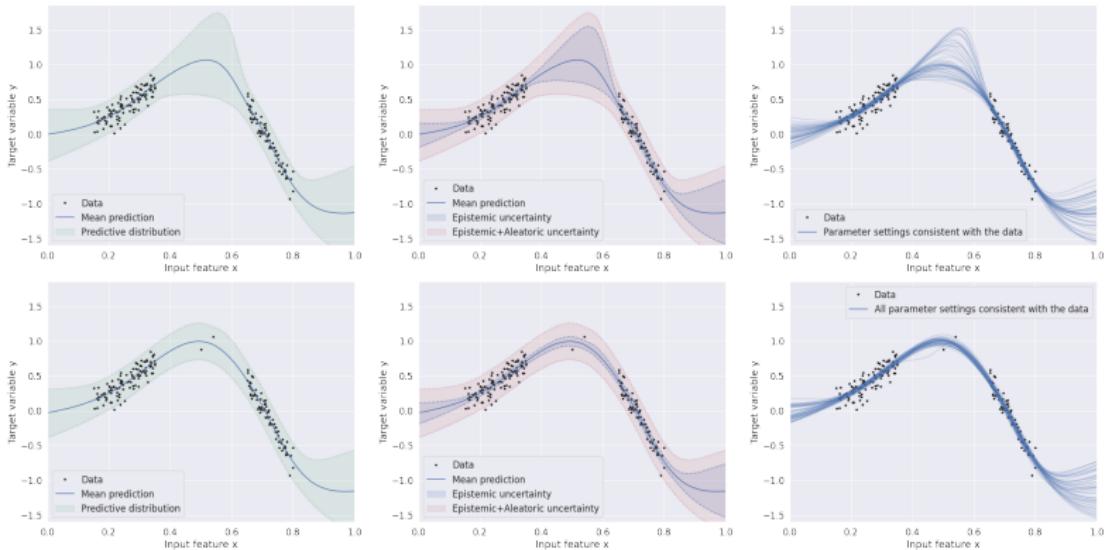
$$p(y^*|x^*) = \int p(y^*|x^*, \mathbf{w})p(\mathbf{w}|y)d\mathbf{w}$$



Uncertainty quantification

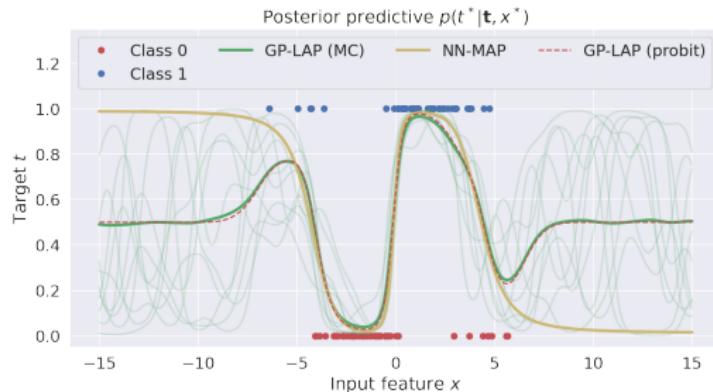
Two sources of uncertainty

1. *Epistemic uncertainty* is due to lack of knowledge (e.g. often due to a limited data set). Also sometimes called the *reducible* uncertainty.
2. *Aleatoric uncertainty* refers to the inherent randomness (e.g. measurement noise). Also sometimes called the *irreducible* uncertainty.



- Classical training only captures aleatoric uncertainty, *not epistemic uncertainty*

DTU Learn quiz: Epistemic and aleatoric uncertainty for classification



DTU Learn - Quiz - Lecture 13: Aleatoric and epistemic uncertainty

Priors on neural network parameters

- Recall our study of linear models

$$y(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + \epsilon$$

- We imposed Gaussian priors on each $w_i \sim \mathcal{N}(w_i | 0, \kappa^2)$
- We defined the prior in the *weight-space*, but it's easy to understand the effect in *function-space*
- *Gaussian processes*: a prior directly on function-space – often easy to interpret parameters
- It is common to impose Gaussian priors on the parameters of a NN, but it is *hard to interpret* what it means in function space.
- Let's generate some samples from the prior $\mathbf{w} \sim \mathcal{N}(0, \kappa^2 \mathbf{I})$ for $\mathbf{W} = \{\mathbf{W}_0, \mathbf{b}_0, \mathbf{W}_1, \mathbf{b}_1, \mathbf{W}_2, \mathbf{b}_2\}$

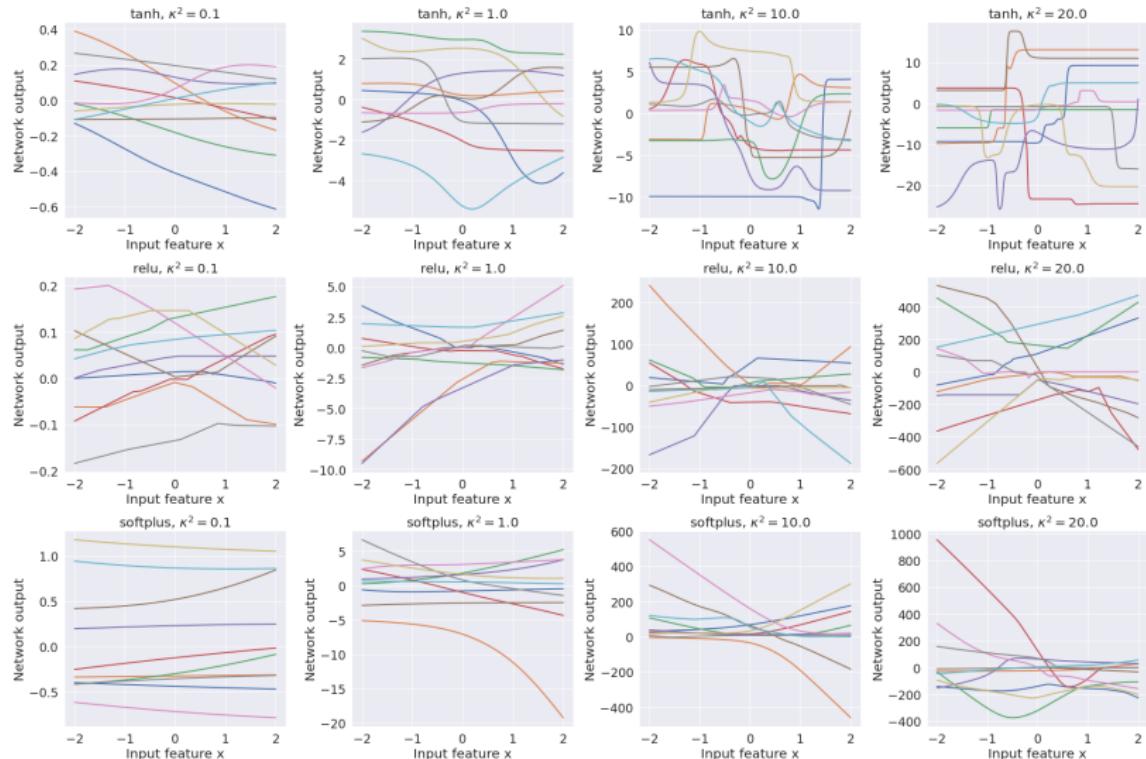
$$\mathbf{z}_1 = h(\mathbf{W}_0 \mathbf{x} + \mathbf{b}_0)$$

$$z_2 = h(\mathbf{W}_1 \mathbf{z}_1 + \mathbf{b}_1)$$

$$y = \mathbf{W}_2 \mathbf{z}_2 + \mathbf{b}_2$$

Prior samples from Bayesian neural network

Gaussian priors on all parameters $\mathbf{w} \sim \mathcal{N}(0, \kappa^2 \mathbf{I})$, 2 layers with 5 units each



Bayesian Neural Networks: Approximating the posterior for neural networks

Approximating the posterior for neural networks

- All the standard tools have been used
 1. Laplace approximations
 2. Variational inference (BBVI)
 3. MCMC (Hamiltonian Monte Carlo)
- MCMC is the golden standard, but does not scale to interesting datasets
- Modern networks can have huge parameter spaces
 1. Image: ResNet-50 contains ≈ 23 million parameters
 2. Text: GPT-3 has 175 billion parameters
- Even the simplest mean-field approximation would contain twice as many variational parameters
- Many approximate solutions have been proposed
 1. Monte Carlo Drop-out (McD)
 2. Last-layer Laplace approximations (LLLA)
 3. Deep ensembles
 4. Stochastic Weight Averaging Gaussian (SWAG)
 5. Last-layer Laplace + Variational inference with normalizing flows
 6. ...
- Large emphasis on *post-hoc* Bayesian approximations with little computational overhead *given a pretrained model w_{MAP}*

Approximating the predictive distribution I

- Overall goal is to get the *posterior predictive distribution* by averaging wrt. the posterior distribution

$$p(y^* | \mathbf{y}) = \int p(y^* | \mathbf{x}^*, \mathbf{w}) p(\mathbf{w} | \mathbf{y}) d\mathbf{w} \approx \int p(y^* | \mathbf{x}^*, \mathbf{w}) q(\mathbf{w}) d\mathbf{w}$$

- Given $q(\mathbf{w})$, the *posterior predictive* can be estimated using *Monte Carlo* sampling

$$p(y^* | \mathbf{y}) \approx \int p(y^* | \mathbf{x}^*, \mathbf{w}) q(\mathbf{w}) d\mathbf{w} \approx \frac{1}{S} \sum_{i=1}^S p(y^* | \mathbf{x}^*, \mathbf{w}^{(i)}) \quad \text{for } \mathbf{w}^{(i)} \sim q(\mathbf{w})$$

- There are two common overall strategies to obtain “posterior samples” $\mathbf{w}^{(i)}$

1. We do a proper posterior approximation $q(\mathbf{w})$ followed by sampling $\mathbf{w}^{(i)} \sim q(\mathbf{w})$

For instance, variational inference, Laplace approximations, ...

2. We collect a set of samples $\{\mathbf{w}^{(i)}\}_{i=1}^S$ directly without having an actual distribution $q(\mathbf{w})$

For instance, SGD as approximate inference, SWAG, deep ensembles, MAP-inference, ...

Bayesian Neural Networks: Monte Carlo Dropout

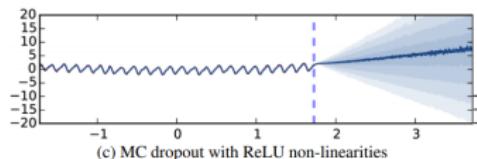
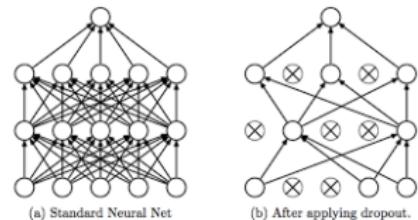
Monte Carlo Dropout I

- *Dropout is a regularization technique for training NNs:*

During training, replace input for each layer with a product of a random variable, e.g. $\mathbf{W}_0 \mathbf{x}$ is replaced with

$$\mathbf{W}_0(\mathbf{x} \circ \mathbf{s}), \quad \text{where } \mathbf{s} \sim p(\mathbf{s})$$

- For example, $\mathbf{s} \sim \text{Ber}(p)$, where $1 - p$ is the drop out rate *ignores* a random subset of the input in each iteration
- Dropout as regularization can be shown to be equivalent to a special case of ridge regression (ℓ_2 -penalty)
- *Monte Carlo Dropout:* Perform drop-out at test time and average over results
- Very popular, very easy to implement, but not really justified in anyway



(c) MC dropout with ReLU non-linearities

Monte Carlo Dropout II

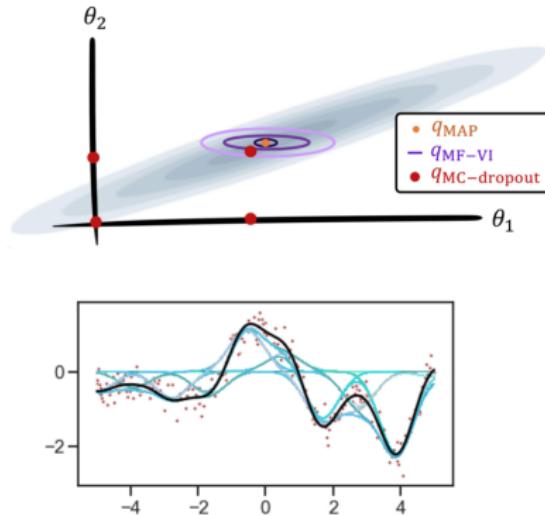


Fig. 3. MC-dropout posterior on 1-layer RBF regression. Black line: true model. Brown dots: observations. Blue lines are not samples from the approximate posterior, but the set of 1024 delta-Dirac functions forming the multimodal MC-dropout posterior. Evidently the location of the modes is unrelated to the model uncertainty and is instead simply an artefact of the MC-dropout approximation.

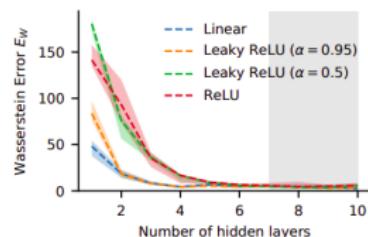
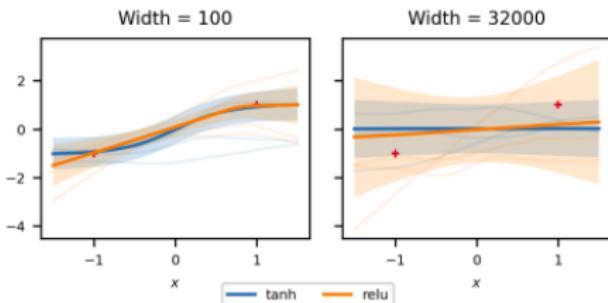
- Epistemic uncertainty does not decrease as size of dataset increases
- Yet, there are many papers claiming to benefit from MC Dropout.

Bayesian Neural Networks: Mean-field

Mean-field for BNNs I

What about mean-field for approximating $q(\mathbf{w})$?

- Mean-field approximations can be tricky for BNNs: $q(\mathbf{w}) = \prod_i \mathcal{N}(w_i | m_i, v_i)$
- Helps to initialize m_i using $(\mathbf{w}_{MAP})_i$ and initailize v_i to small values, i.e. $v_i = 10^{-6}$
- New research and theory on MFVI for BNNs show
 1. For a BNN with a *single* hidden layer: if we make the network wider and wider, eventually the posterior approximation will converge to the *prior distribution*. That is, the model will eventually *completely ignore the data*.
 2. As networks becomes deeper and deeper, the posterior distribution will become *easier* to approximate with a mean-field approximation



Coker et al., 2021: Wide Mean-Field Variational Bayesian Neural Networks Ignore the Data

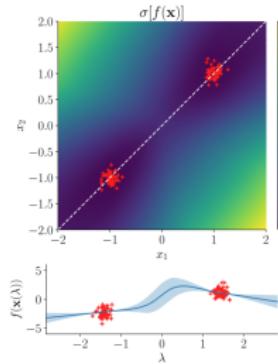
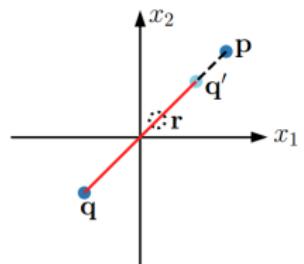
Farquhar et al., 2020: Liberty or Depth: Deep Bayesian Neural Nets Do Not Need Complex Weight Posterior Approximations

Mean-field for BNNs II

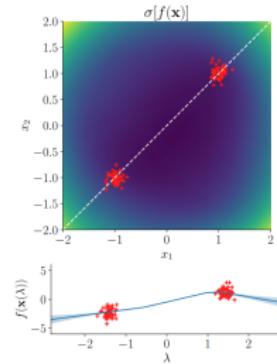
- For single-layer networks: if $\mathbf{r} = (1 - \lambda)\mathbf{p} + \lambda\mathbf{q}$ for $\lambda \in [0, 1]$, it has been proven that

$$\mathbb{V}[y(\mathbf{r})] \leq \mathbb{V}[y(\mathbf{p})] + \mathbb{V}[y(\mathbf{q})]$$

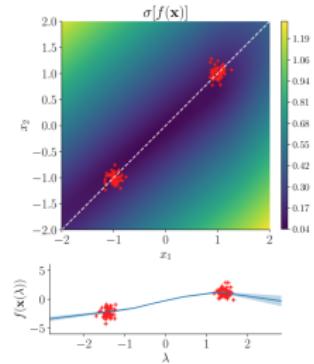
- Note: Figure from paper represents NN with f rather than y



(b) HMC



(c) MFVI



(d) MCDO

Bayesian Neural Networks: Laplace approximations

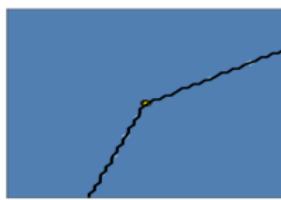
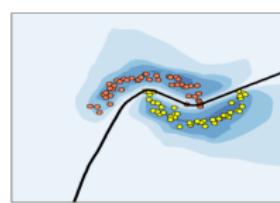
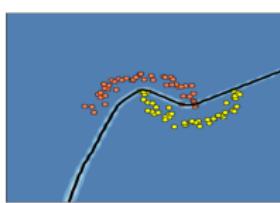
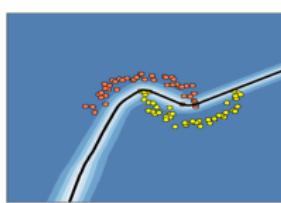
Laplace approximations for BNNs

- Recall the Laplace approximation

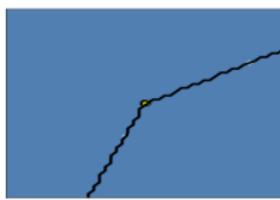
$$p(\mathbf{w}|\mathbf{y}) \approx q(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mathbf{w}_{\text{MAP}}, -\mathbf{A}^{-1})$$

where

$$\mathbf{A} = -\nabla \nabla \ln p(\mathbf{y}|\mathbf{w})p(\mathbf{w})|_{\mathbf{w}=\mathbf{w}_{\text{MAP}}}$$



(a) MAP



(b) Temp. scaling



(c) Bayesian (last-layer)

Laplace approximations for BNNs

- Recall the Laplace approximation

$$p(\mathbf{w}|\mathbf{y}) \approx q(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mathbf{w}_{\text{MAP}}, -\mathbf{A}^{-1})$$

where

$$\mathbf{A} = -\nabla \nabla \ln p(\mathbf{y}|\mathbf{w})p(\mathbf{w})|_{\mathbf{w}=\mathbf{w}_{\text{MAP}}}$$

- Many nice properties

1. Can be applied post-hoc given a pretrained model \mathbf{w}_{MAP}
2. For many likelihoods (e.g. binary classification), it can be shown to give the same *decisions*, but are often better calibrated
3. Simple, intuitive and relatively easy to implement

- But what is the size of the Hessian for a neural network with 10 million parameters?

$$\mathbf{A} \in \mathbb{R}^{10^7 \times 10^7}$$

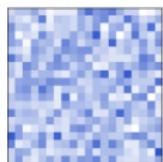
- It would require approximately 728 TB to store in memory (64 bit precision)

Approximating the Hessian

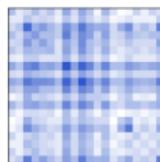
- The (negative) Hessian is given by

$$\mathbf{A} = -\nabla \nabla \ln p(\mathbf{y}|\mathbf{w}) p(\mathbf{w}) \Big|_{\mathbf{w}=\mathbf{w}_{\text{MAP}}} = -\underbrace{\nabla \nabla \ln p(\mathbf{y}|\mathbf{w}) \Big|_{\mathbf{w}=\mathbf{w}_{\text{MAP}}}}_{\text{Hessian of log lik.}} - \underbrace{\nabla \nabla \ln p(\mathbf{w}) \Big|_{\mathbf{w}=\mathbf{w}_{\text{MAP}}}}_{\text{Hessian of log prior}}$$

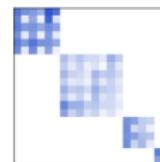
- For a Gaussian prior, the Hessian is easy: $\frac{\alpha}{2} \mathbf{I}$
- Several approximation proposed for the Hessian of the log likelihood
 - Full (no approximation)
 - Low rank
 - Kronecker-factored approximate curvature (KFAC)
 - Diagonal



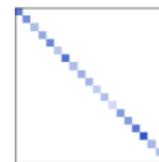
(a) Full



(b) LRank



(c) KFAC

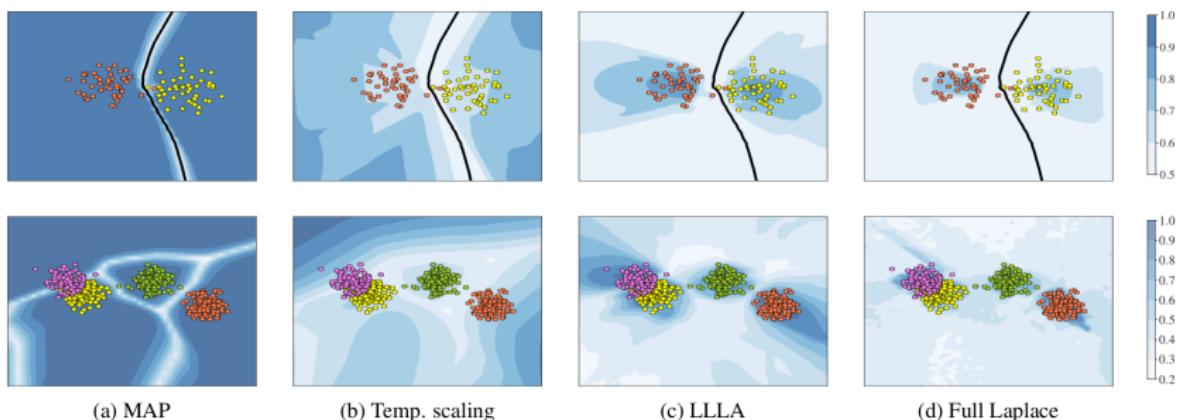


(d) Diag.

- Hessians are not guaranteed to be positive definite, so sometimes the generalized Gauss-Newton matrix is used instead

Last-layer Laplace approximations (LLL) I

- Laplace approximations for the last layer only is often a very good alternative
- ResNet18 to be fine-tuned to CIFAR10 (image dataset with 10 classes)
 1. Full network: contains $\approx 11 \cdot 10^6$ parameters in total
 2. Last fully connected layer: contains $\approx 5 \cdot 10^3$ parameters
- Often works almost as well as for the full Laplace approximation, but much faster



Kristiadi et al., 2020: Being Bayesian, Even Just a Bit, Fixes Overconfidence in ReLU Networks (image credit)

Last-layer Laplace approximations (LLL) II

- Consider our simple network with two hidden layers

$$\mathbf{z}_1 = h(\mathbf{W}_0 \mathbf{x} + \mathbf{b}_0)$$

$$\mathbf{z}_2 = h(\mathbf{W}_1 \mathbf{z}_1 + \mathbf{b}_1)$$

$$f = \mathbf{W}_2 \mathbf{z}_2 + \mathbf{b}_2$$

■ Recipe for fitting a LLLA

Step 1: Compute \mathbf{w}_{MAP} for full network (or use pretrained model)

Step 2: Estimate (full) Hessian for parameters of the last layer $\mathbf{w}_L = \{\mathbf{W}_2, \mathbf{b}_2\}$

Step 3: Construct approximation $q_{LLLA}(\mathbf{w}_L) = \mathcal{N}(\mathbf{w}_L | \mathbf{w}_L, \mathbf{A}_L^{-1})$

■ Recipe for making predictions with LLLA for new point \mathbf{x}^*

Step 1: Generate samples $\mathbf{w}_L^{(i)} \sim q_{LLLA}(\mathbf{w}_L)$ for $i = 1, \dots, S$

Step 2: First feed input \mathbf{x}^* through first layers of network to get \mathbf{z}_2^*

Step 3: Compute $f^{(i)} = f(\mathbf{z}_2^*)$ for each posterior sample of the weights $\mathbf{w}_L^{(i)} = \{\mathbf{W}_2^{(i)}, \mathbf{b}_2^{(i)}\}$

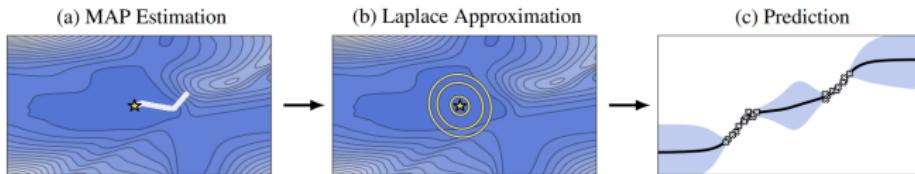
Step 4: Use $\left\{f^{(i)}\right\}_{i=1}^S$ as samples to estimate the posterior predictive

Last-layer Laplace approximations (LLL) III

- *Laplace Redux*: Very convenient and robust package for Laplace approximations in Pytorch

<https://github.com/AlexImmer/Laplace>

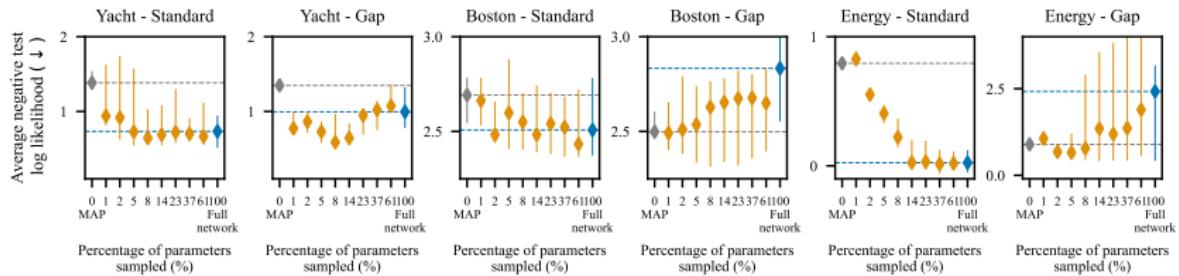
- *Plug and play* for many common Pytorch models
- Very good starting point for a more *Bayesian approach* to deep learning *in practice*
- Supports both *full Laplace* (with various approximations) and *last-layer Laplace*



Daxberger et al, 2022: Laplace Redux – Effortless Bayesian Deep Learning (image credit)

Is a stochastic last layer really the best?

- Recent work has shown that networks with a small amount of (input-independent) stochasticity *before* the last layer are *universal conditional distribution approximators*.
- This is verified experimentally: partially stochastic networks consistently either match or outperform their fully-stochastic variants.
- However, it is unclear if the sampling for the fully stochastic networks has adequately captured the posterior.

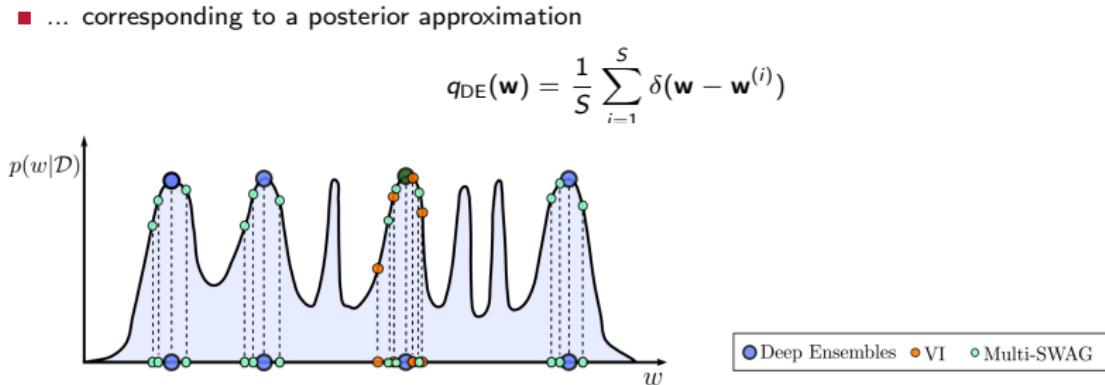


Bayesian Neural Networks: Deep Ensembles

Deep Ensembles I

- *Deep ensembling* is often an easy way to improve generalization and calibration
- Super simple idea
 1. re-train your model S times from different initial parameter values (e.g. different seeds)
 2. compute predictions for the S different model parameters and average the results
- More formally, deep ensembles approximates the predictive posterior as follows
 - ... corresponding to a posterior approximation

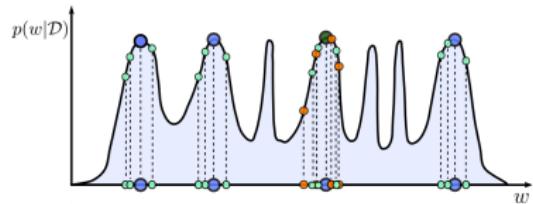
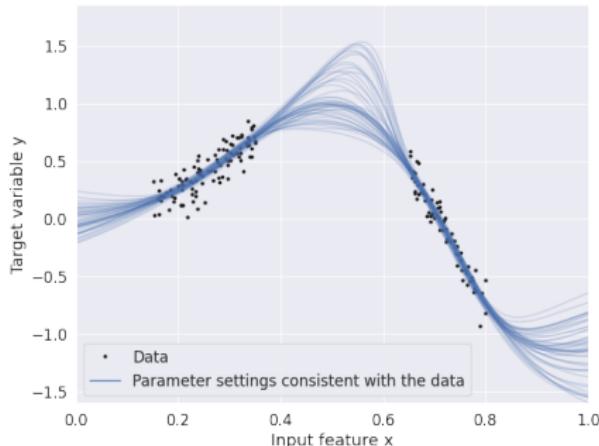
$$p(y^* | \mathbf{y}, \mathbf{x}^*) \approx \int p(y^* | \mathbf{w}, \mathbf{x}^*) q_{\text{DE}}(\mathbf{w}) d\mathbf{w} = \frac{1}{S} \sum_{i=1}^S p(y^* | \mathbf{w}^{(i)}, \mathbf{x}^*).$$



Lakshminarayanan et al, 2017: Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles

Wilson & Izmailov, 2021: Bayesian Deep Learning and a Probabilistic Perspective of Generalization (image credit)

Deep Ensembles II



Pros

1. Simple and very easy to implement
2. Training each model can easily be parallelized
3. Often works very well in practice

Cons

1. Requires training S copies of the model
2. Predictions become more expensive (S forward passes needed)
3. All models in the ensemble are given the same importance

Lakshminarayanan et al, 2017: Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles

Wilson & Izmailov, 2021: Bayesian Deep Learning and a Probabilistic Perspective of Generalization (image credit)

Summary for BNNs

- Motivation

1. Classical neural networks are often overconfident
2. We want both epistemic and aleatoric uncertainty

- Very challenging to do Bayesian inference for NNs

1. Complicated posterior geometries
2. Very high-dimensional parameter spaces
3. Hard to assign informative priors
4. Must scale to large datasets

- We discussed several options for approximate inference

1. MC Dropout
2. Mean-field
3. Laplace approximations
4. Deep ensembles

The end

- Thank you for participating in the course and for being patient with all the small issues: Typos in materials etc.
- Thank you for taking the time to provide valuable feedback and improving the course and the material
- Remember: Q&A session on Wednesday the 14th of May from 10-12 in building 321, room 033

