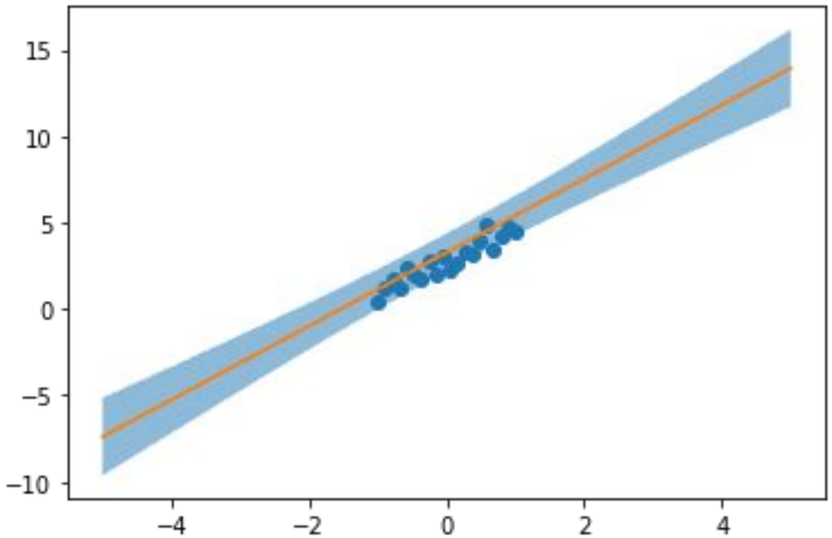
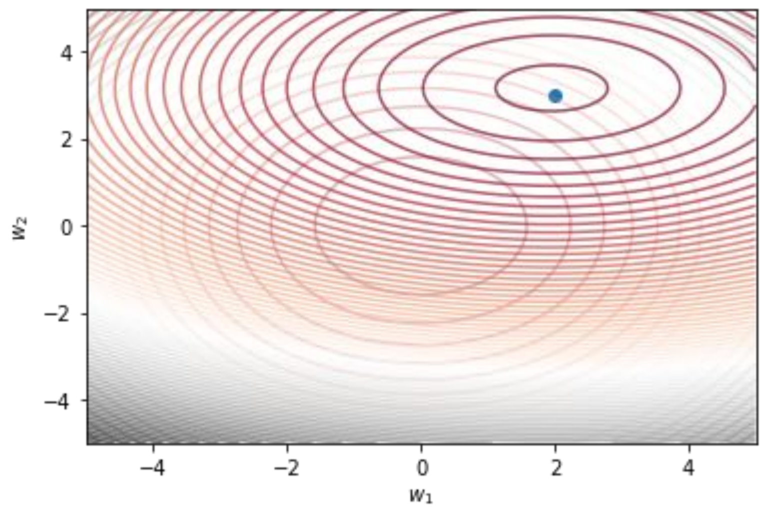
Bayesian Inference

Prof. Dr. Nico Serra - University of Zurich





Introduction

Joint distribution

$$p(x, y, z)$$

Marginal distribution

$$p(x, y) = \int p(x, y, z)\, dz$$

Conditional distribution

$$p(x|y) = \frac{p(x, y)}{p(y)} = \frac{\int p(x, y, z)\, dz}{\int p(x, y, z)\, dxdz}$$

Chain product rule

$$p(x, y, z) = p(x|y, z)\, p(y|z)\, p(z)$$

Bayes Theorem

Posterior

Likelihood

prior

$$p(\,theory|data) = \frac{p(\,data|theory)\,p(\,theory)}{p(\,data)}$$

Evidence

Bayes Theorem

Posterior

Likelihood

prior

$$p(\, theory | data) = \frac{p(\, data | theory)\, p(\, theory)}{p(\, data)}$$

Evidence

Model parameters give data:

$$p(\,\vartheta | X) = \frac{p(\,X|\theta)\,p(\,\theta)}{p(\,X)} = \frac{p(\,X|\theta)\,p(\,\theta)}{\int p(\,X|\theta)\,p(\,\theta)\,d\theta}$$

$X = Data$

$\theta = Model\ parameters$

Bayes Theorem

Posterior Likelihood prior

$$p(\,theory|data\,) = \frac{p(\,data|theory\,)\,p(\,theory\,)}{p(\,data\,)}$$

Evidence

Model parameters give data:

$$p(\,\vartheta|X) = \frac{p(\,X|\theta)\,p(\,\theta)}{p(\,X)} = \frac{p(\,X|\theta)\,p(\,\theta)}{\int p(\,X|\theta)\,p(\,\theta)\,d\theta}$$

$X = Data$

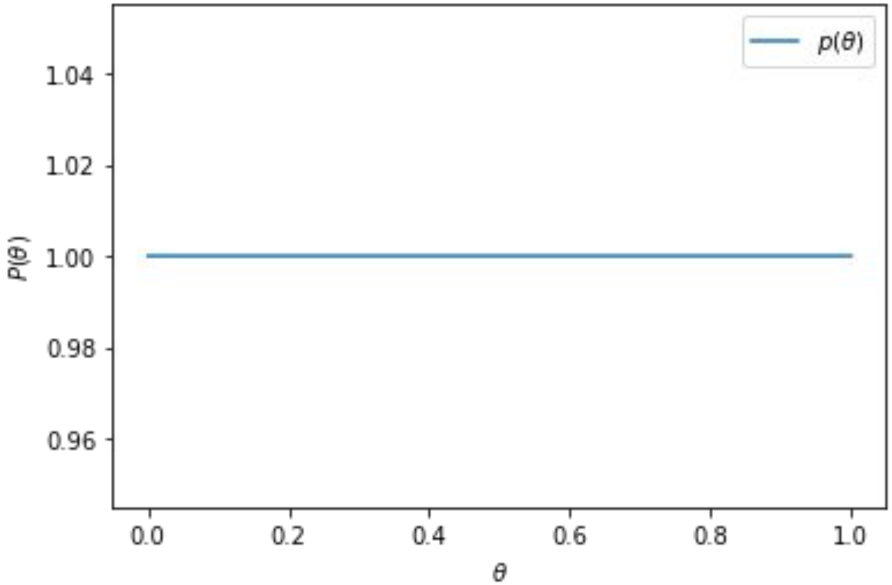
$\theta = Model\ parameters$

Problems:

- We need to start from a prior
- We need to calculate the Evidence, which often is an intractable integral

Bayes Theorem - Example

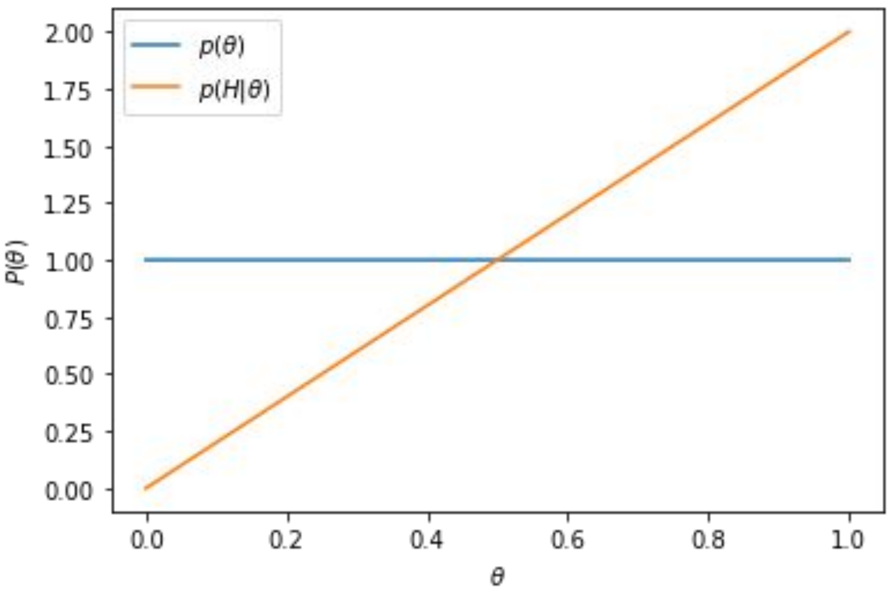
We are tossing a coin, but we do not know if it is a fair coin, we want to establish the head probability (for a fair coin of course θ=0.5).



$$p(\vartheta|X) = \frac{p(X|\theta)\,p(\theta)}{p(X)} = \frac{p(X|\theta)\,p(\theta)}{\int p(X|\theta)\,p(\theta)\,d\theta}$$

Bayes Theorem - Example

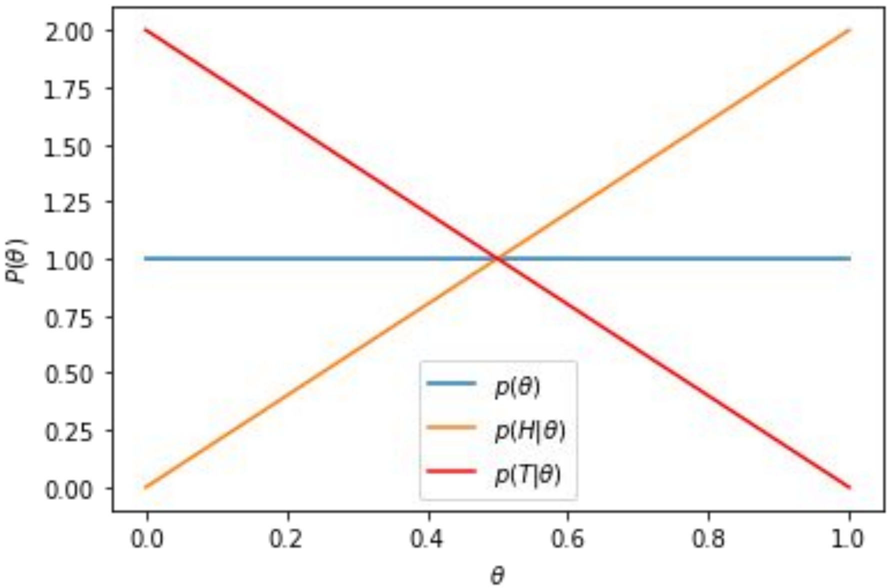
We are tossing a coin, but we do not know if it is a fair coin, we want to establish the head probability (for a fair coin of course θ=0.5).



$$p(\vartheta|X) = \frac{p(X|\theta)\,p(\theta)}{p(X)} = \frac{p(X|\theta)\,p(\theta)}{\int p(X|\theta)\,p(\theta)\,d\theta}$$

Bayes Theorem - Example

We are tossing a coin, but we do not know if it is a fair coin, we want to establish the head probability (for a fair coin of course θ=0.5).



$$p(\vartheta|X) = \frac{p(X|\theta)\,p(\theta)}{p(X)} = \frac{p(X|\theta)\,p(\theta)}{\int p(X|\theta)\,p(\theta)\,d\theta}$$

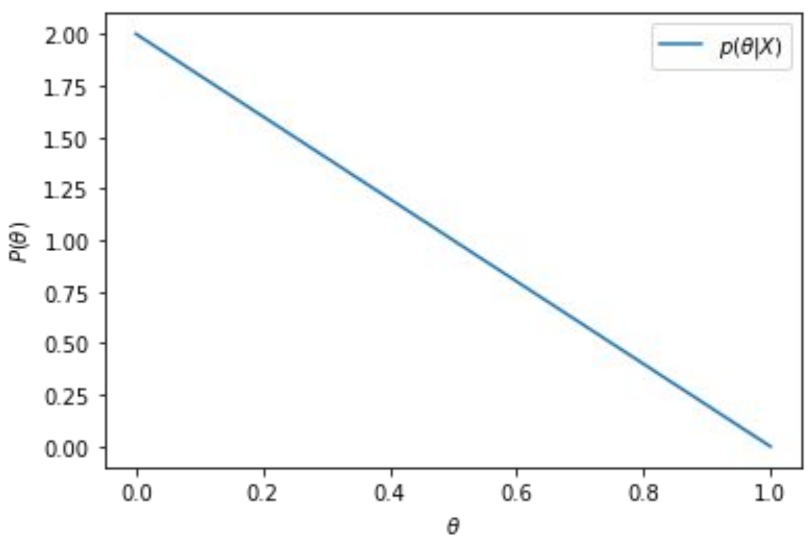
Bayes Theorem - Example

We are tossing a coin, but we do not know if it is a fair coin, we want to establish the head probability (for a fair coin of course θ=0.5).

$$p(\vartheta|X) = \frac{p(X|\theta)\,p(\theta)}{p(X)} = \frac{p(X|\theta)\,p(\theta)}{\int p(X|\theta)\,p(\theta)\,d\theta}$$

We start tossing the coin and find:

[0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, …]



MISIS National University of Science and Technology

University of Zurich[UZH]

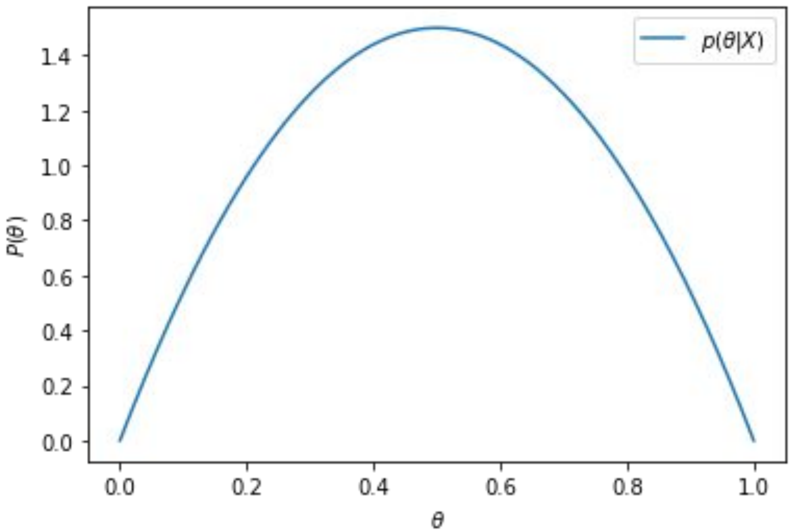
Bayes Theorem - Example

We are tossing a coin, but we do not know if it is a fair coin, we want to establish the head probability (for a fair coin of course θ=0.5).

$$p(\vartheta|X) = \frac{p(X|\theta)\,p(\theta)}{p(X)} = \frac{p(X|\theta)\,p(\theta)}{\int p(X|\theta)\,p(\theta)\,d\theta}$$

We start tossing the coin and find:

[0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, …]



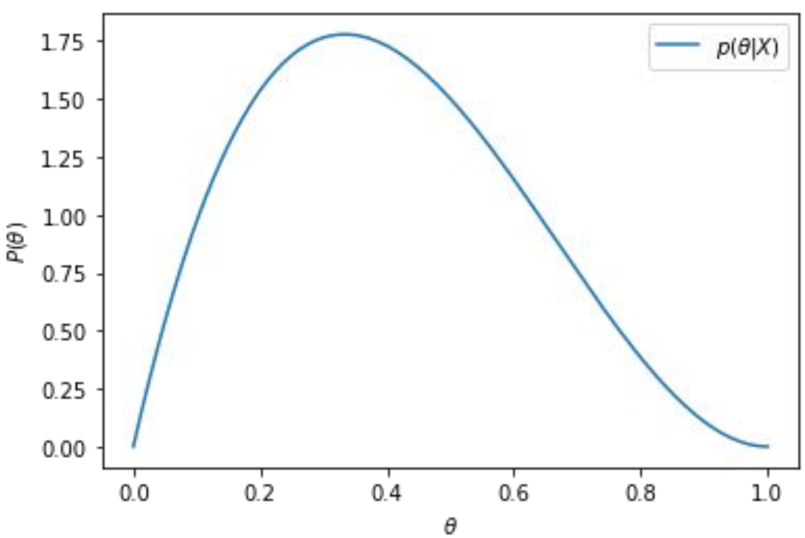
Bayes Theorem - Example

We are tossing a coin, but we do not know if it is a fair coin, we want to establish the head probability (for a fair coin of course θ=0.5).

$$p(\vartheta|X) = \frac{p(X|\theta)\,p(\theta)}{p(X)} = \frac{p(X|\theta)\,p(\theta)}{\int p(X|\theta)\,p(\theta)\,d\theta}$$

We start tossing the coin and find:

[0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, …]



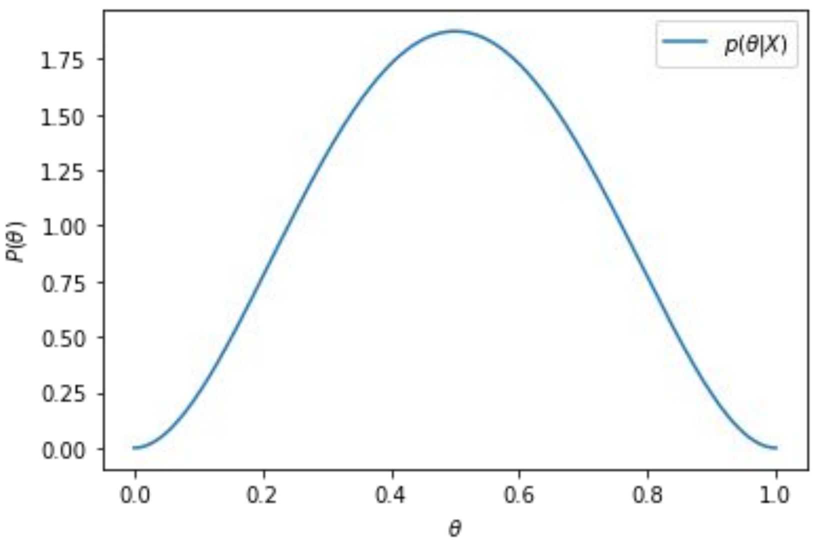
Bayes Theorem - Example

We are tossing a coin, but we do not know if it is a fair coin, we want to establish the head probability (for a fair coin of course θ=0.5).

$$p(\vartheta|X) = \frac{p(X|\theta)\,p(\theta)}{p(X)} = \frac{p(X|\theta)\,p(\theta)}{\int p(X|\theta)\,p(\theta)\,d\theta}$$

We start tossing the coin and find:

[0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, …]



Bayes Theorem - Example

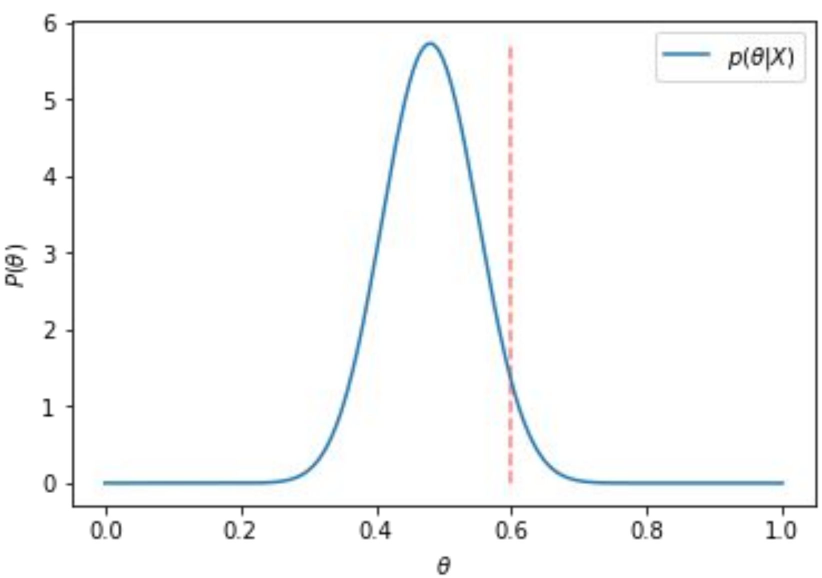
We are tossing a coin, but we do not know if it is a fair coin, we want to establish the head probability (for a fair coin of course θ=0.5).

$$p(\vartheta|X) = \frac{p(X|\theta)\,p(\theta)}{p(X)} = \frac{p(X|\theta)\,p(\theta)}{\int p(X|\theta)\,p(\theta)\,d\theta}$$

We start tossing the coin and find:

[0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, …]

After 50 iterations



Bayes Theorem - Example

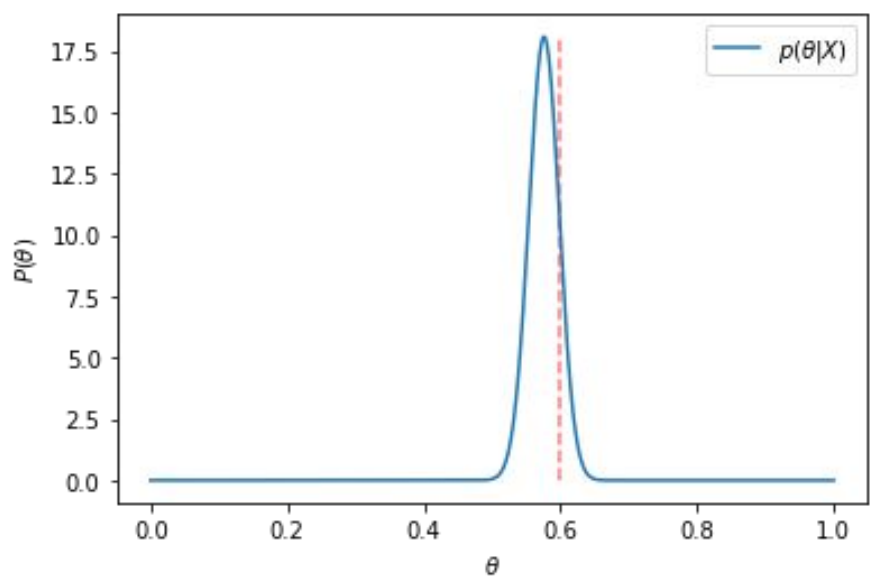
We are tossing a coin, but we do not know if it is a fair coin, we want to establish the head probability (for a fair coin of course θ=0.5).

$$p(\vartheta|X) = \frac{p(X|\theta)\,p(\theta)}{p(X)} = \frac{p(X|\theta)\,p(\theta)}{\int p(X|\theta)\,p(\theta)\,d\theta}$$

We start tossing the coin and find:

[0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, …]

After 500 iterations



Bayes Theorem - Example

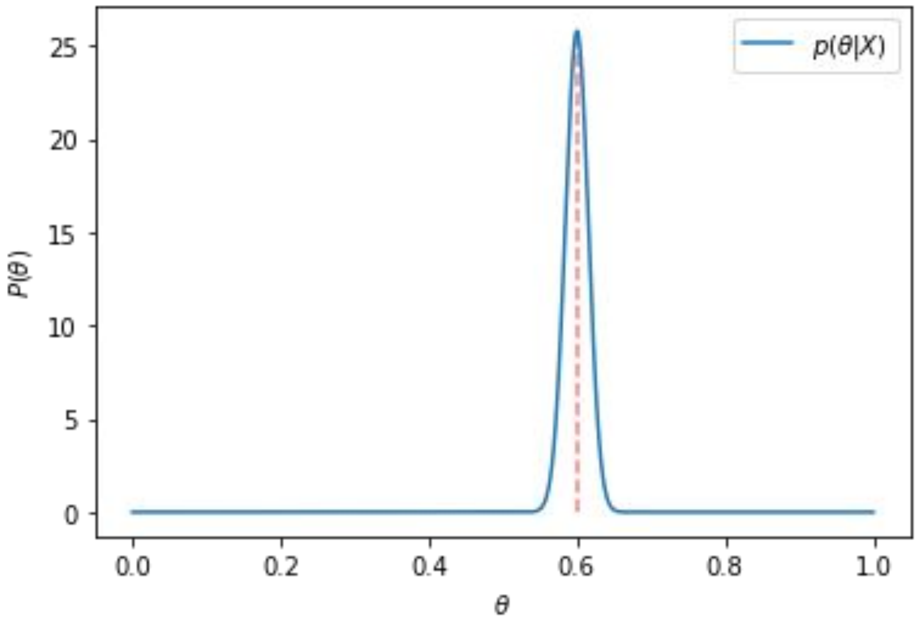
We are tossing a coin, but we do not know if it is a fair coin, we want to establish the head probability (for a fair coin of course θ=0.5).

$$p(\vartheta|X) = \frac{p(X|\theta)\,p(\theta)}{p(X)} = \frac{p(X|\theta)\,p(\theta)}{\int p(X|\theta)\,p(\theta)\,d\theta}$$

We start tossing the coin and find:

[0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, …]

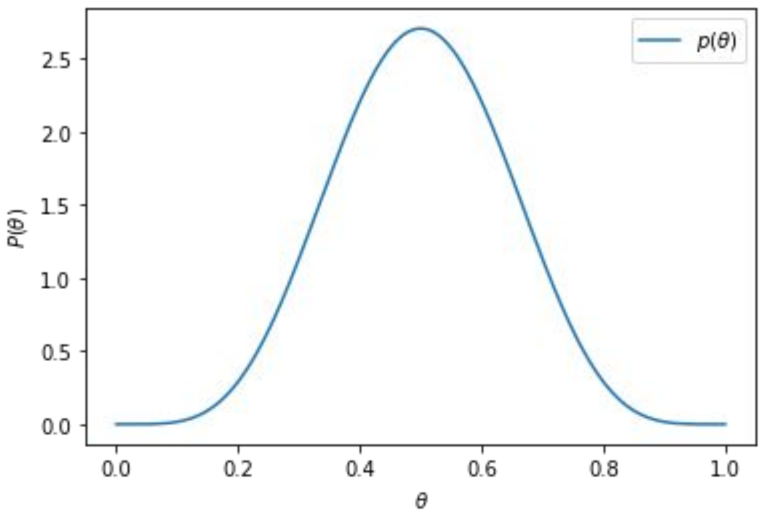
After 1000 iterations



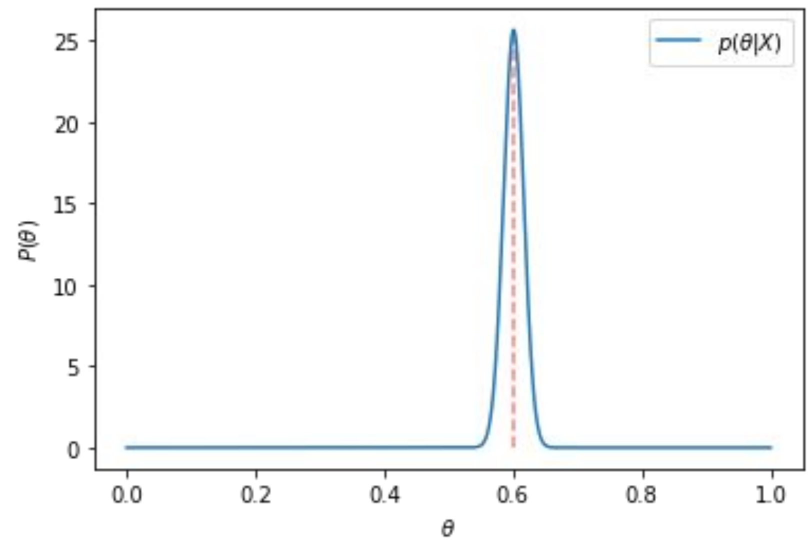
Bayes Theorem - Prior

- Some people see the fact you have to start with a prior as a problem, but if you have enough data the prior does not matter so much

New Prior



Almost identical posterior after 1000 iterations



- When your sample is not so large the prior matters more, but it also acts as a regularizer and allows you to encode your prior knowledge of the parameters

Exercise

Implement the coin toss experiment in python which allows you to calculate the posterior distribution given a prior and a set of observations.

HINT: Use numpy to generate random data and scipy.integrate.simps to normalize the distributions and get the posterior

NB: This problem can be fully solved analytically, however, there are so few problems that can be solved analytically that we will ignore analytical solutions in this course. So in general we will treat the simple problems you can solve analytically with the same techniques you can use in general for real life more complex problems.

Machine Learning implications

We have a ML architecture (e.g. a neural network) that depends on some parameters (θ) and we observe some data X (=x_1,x_2,x_3 ,...)

Traditional framework

Traditional Machine Learning optimization consists of finding the point estimate of the parameters (θ) that maximises the likelihood

$$\theta_{ML} = arg\ max\ p(X|\theta) = arg\ max \prod_i p\left(x_i|\theta\right) = arg\ max \sum_i \log p\left(x_i|\theta\right)$$

Bayesian Machine Learning

Finding the posterior distribution given the data

$$p(\vartheta|X) = \frac{p(X|\theta)\,p(\theta)}{p(X)} = \frac{p(X|\theta)\,p(\theta)}{\int p(X|\theta)\,p(\theta)\,d\theta}$$

Training Bayesian ML

Let's for now concentrate on supervised learning, we are interested in the quantity

$$p\left(y_{test} | x_{test}, x_{train}, y_{train} \right)$$

Training Bayesian ML

Let’s for now concentrate on supervised learning, we are interested in the quantity

$$p\left(y_{test} | x_{test}, x_{train}, y_{train}\right) = \int p\left(y_{test} | x_{test}, \theta\right) p\left(\theta | x_{train}, y_{train}\right) d\theta$$

Once I have built my ML model (or any model) this part I have

This part comes from the training, it is the posterior of the parameters given the training set

Training Bayesian ML

Let's for now concentrate on supervised learning, we are interested in the quantity

$$p(y_{test} | x_{test}, x_{train}, y_{train}) = \int p(y_{test} | x_{test}, \theta)\, p(\theta | x_{train}, y_{train})\, d\theta$$

Once I have built my ML model (or any model) this part I have

This part comes from the training, it is the posterior of the parameters given the training set

$$p(\theta | x_{train}, y_{train}) = \frac{p(x_{train}, y_{train} | \theta)\, p(\theta)}{\int p(x_{train}, y_{train} | \theta)\, p(\theta)\, d\theta} = \frac{p(y_{train} | x_{train}, \theta)\, p(\theta)}{\int p(y_{train} | x_{train}, \theta)\, p(\theta)\, d\theta}$$

MISIS National University of Science and Technology

University of Zurich[UZH]

Training Bayesian ML

Testing

$$p\left(y_{test} | x_{test}, x_{train}, y_{train}\right) = \int p\left(y_{test} | x_{test}, \theta\right) p\left(\theta | x_{train}, y_{train}\right) d\theta$$

Training

$$p\left(\theta | x_{train}, y_{train}\right) = \frac{p\left(x_{train}, y_{train} | \theta\right) p(\theta)}{\int p\left(x_{train}, y_{train} | \theta\right) p(\theta)\, d\theta} = \frac{p\left(y_{train} | x_{train}, \theta\right) p(\theta)}{\int p\left(y_{train} | x_{train}, \theta\right) p(\theta)\, d\theta}$$

- Exact bayesian inference implies solving the two formulas above

Training Bayesian ML

Testing

$$p(y_{test}|x_{test}, x_{train}, y_{train}) = \boxed{\int p(y_{test}|x_{test}, \theta)\, p(\theta|x_{train}, y_{train})\, d\theta}$$

Training

$$p(\theta|x_{train}, y_{train}) = \frac{p(x_{train}, y_{train}|\theta)\, p(\theta)}{\int p(x_{train}, y_{train}|\theta)\, p(\theta)\, d\theta} = \frac{p(y_{train}|x_{train}, \theta)\, p(\theta)}{\boxed{\int p(y_{train}|x_{train}, \theta)\, p(\theta)\, d\theta}}$$

- Exact bayesian inference implies solving the two formulas above
- One problem with “exact” bayesian inference is that the two integrals highlighted can be hard to solve or even intractable
- For this reason in most real situations (apart for known special cases) approximate methods are used

Conjugate Distributions

Exact solution can be applied when the prior and the likelihood are conjugate distributions

$$p(\theta) \in A(\theta) \qquad\qquad p(Y|X,\theta) \in B(\theta)$$

- The prior and the likelihood belong in general to two different classes of distributions
- If the product of prior and likelihood (i.e. the posterior) also belong to the same class as the prior then the prior and the likelihood are conjugate

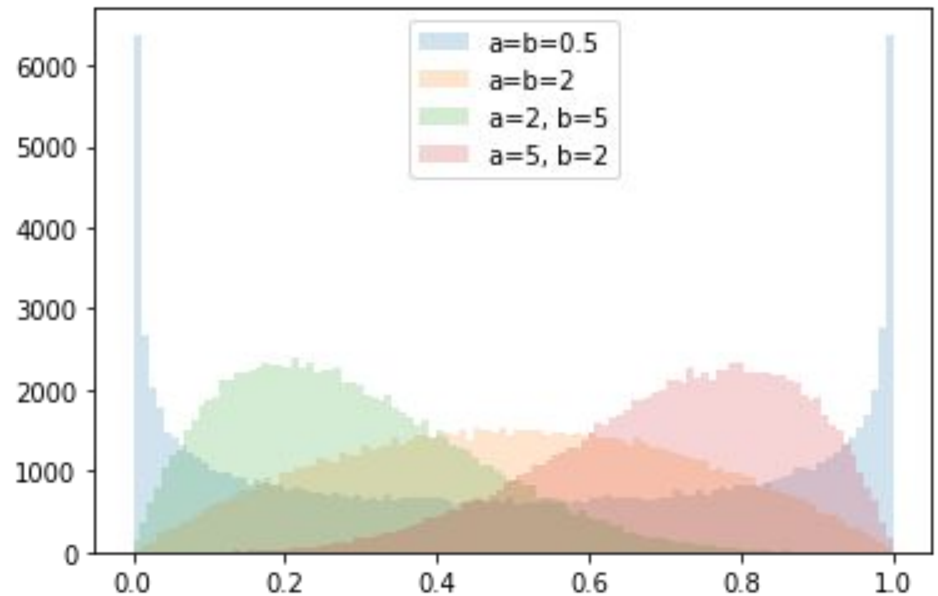
$$p(\theta|X,Y) \in A(\theta)$$

Conjugate Distributions

In the coin example before the probability to get head =1 or tail =0 is described by the Bernoulli distribution

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

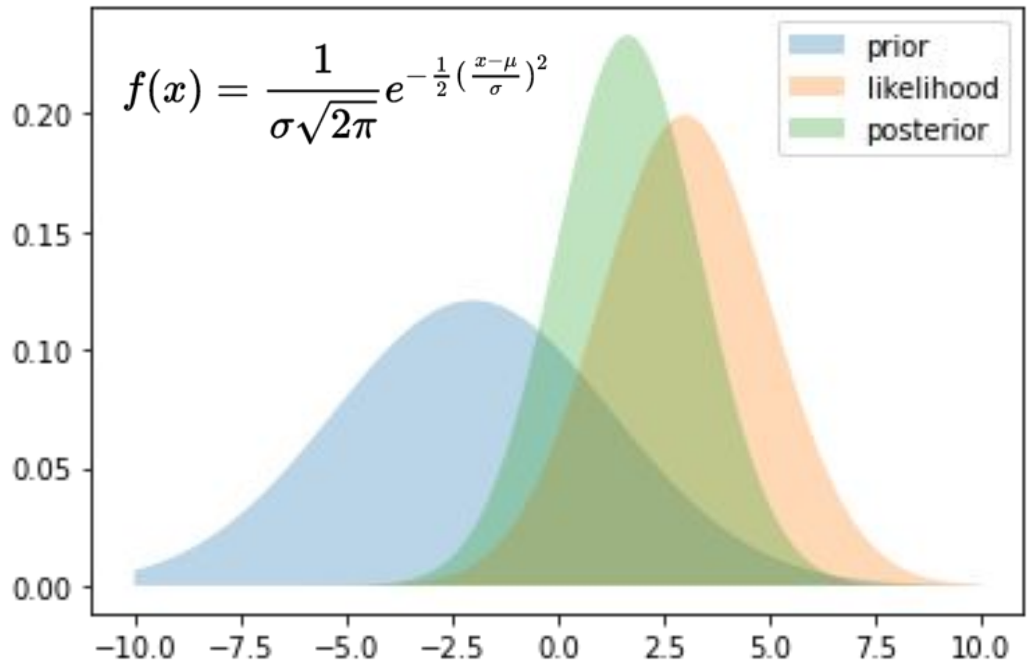
The conjugate distrinution of the Bermoulli is the Beta function, so if the prior is a beta function we know the posterior is also a Bernoulli distribution



Conjugate Distributions

If the likelihood is Gaussian, than the prior has to be Gaussian, since the product of two gaussians distributions is a gaussian distribution

$$\frac{1}{\sqrt{(2\pi)^k \Sigma}} e^{\left\{-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\right\}}$$

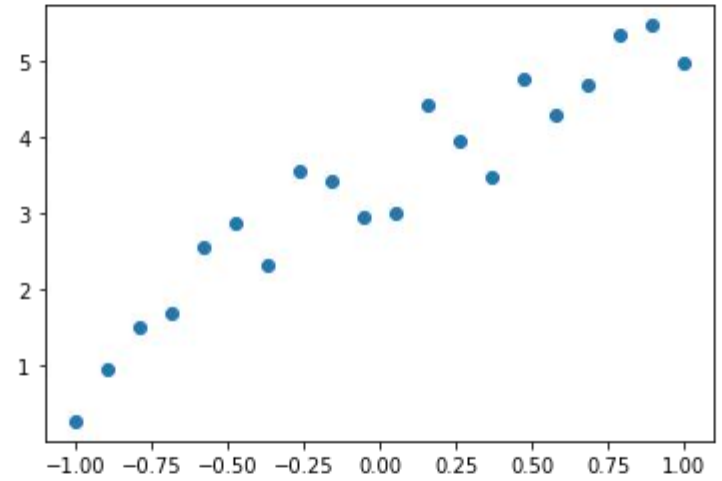


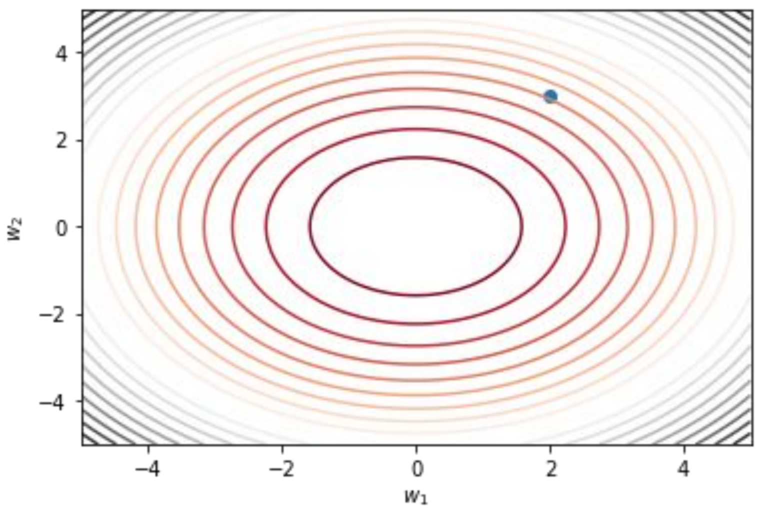
Bayesian Linear Regression

Let's consider a standard linear regression

$$y = w_1 x + w_2 + \varepsilon$$

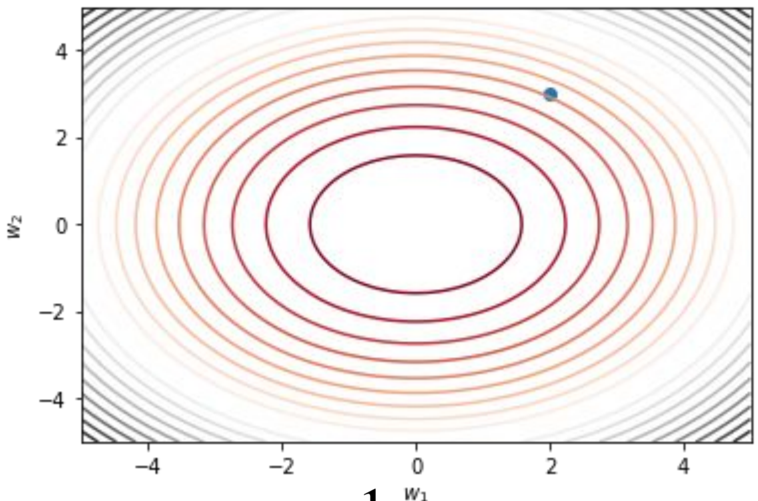
The error is gaussian, so let's assume a gaussian prior

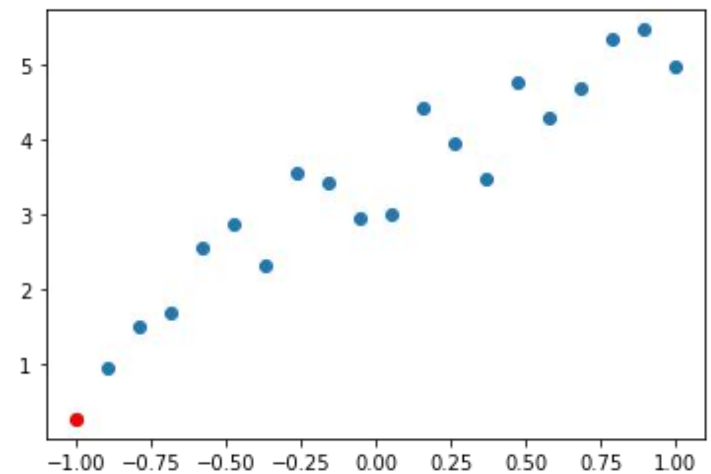




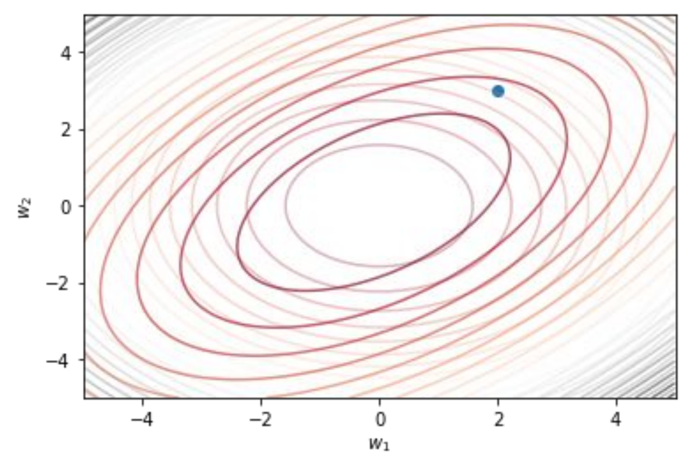
$$p(\theta | X, Y) = \frac{1}{Z} p(Y | X, \theta)\, p(\theta)$$

Bayesian Linear Regression

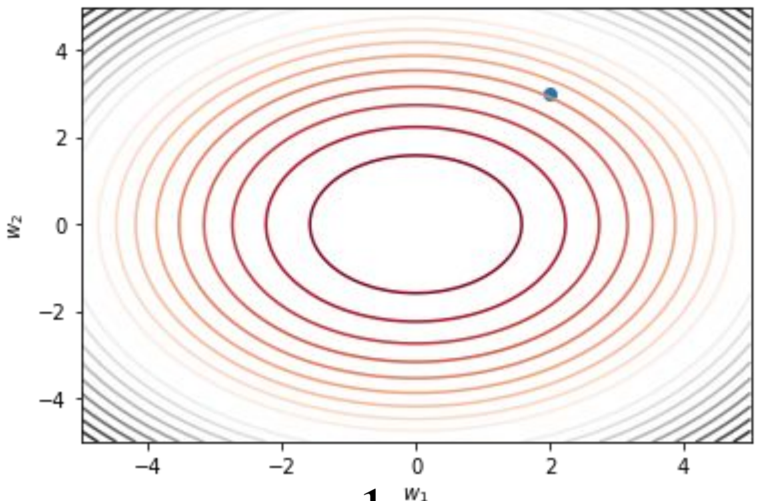


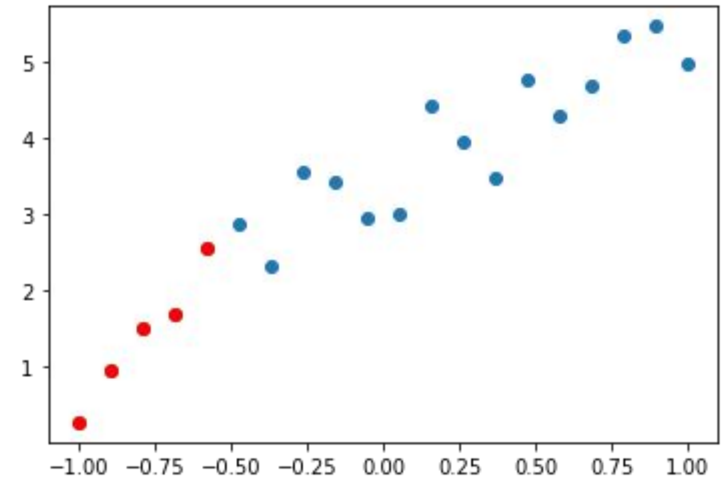


$$p(\theta|X, Y) = \frac{1}{Z} p(Y|X, \theta)\, p(\theta)$$

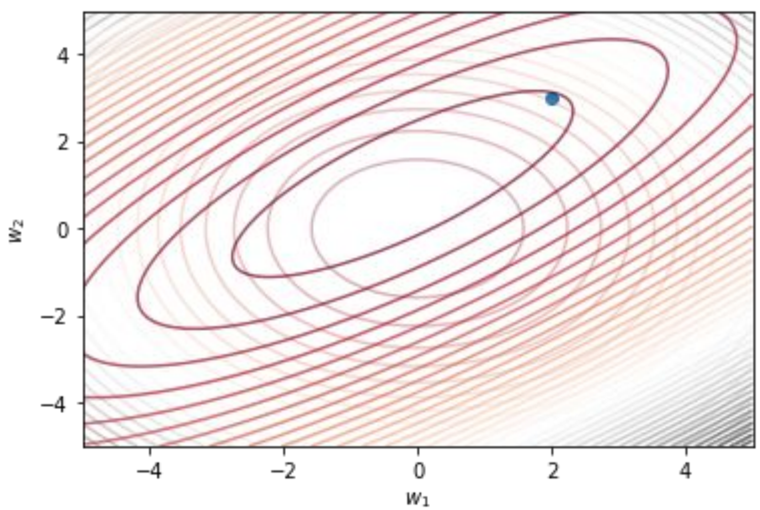


Bayesian Linear Regression

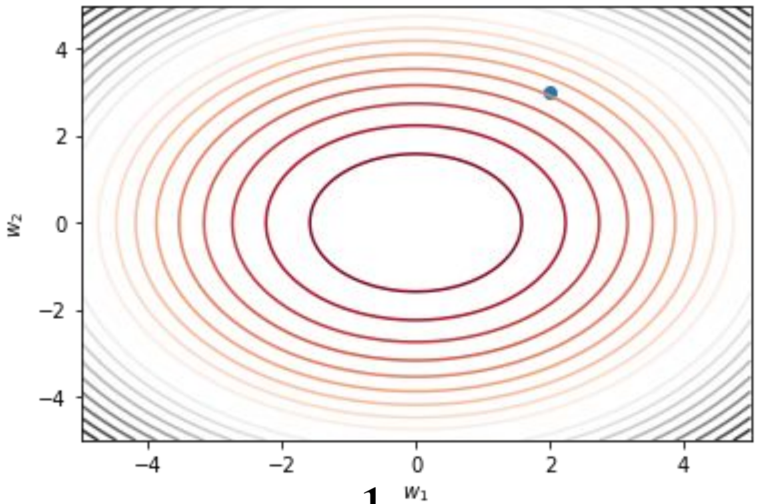


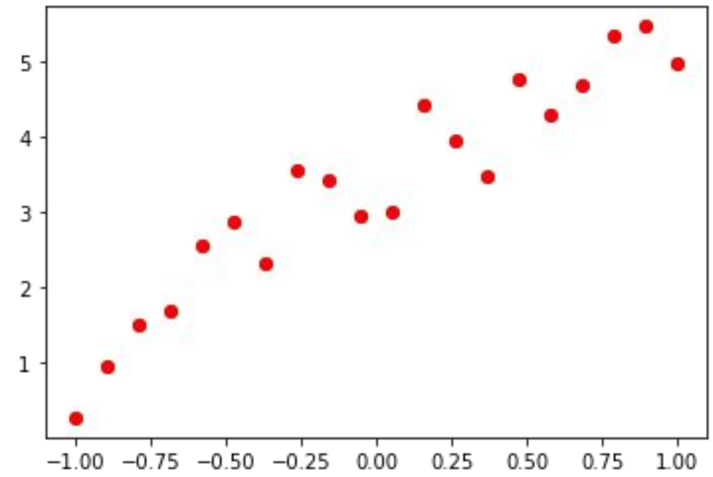


$$p(\theta|X,Y) = \frac{1}{Z} p(Y|X,\theta)\, p(\theta)$$

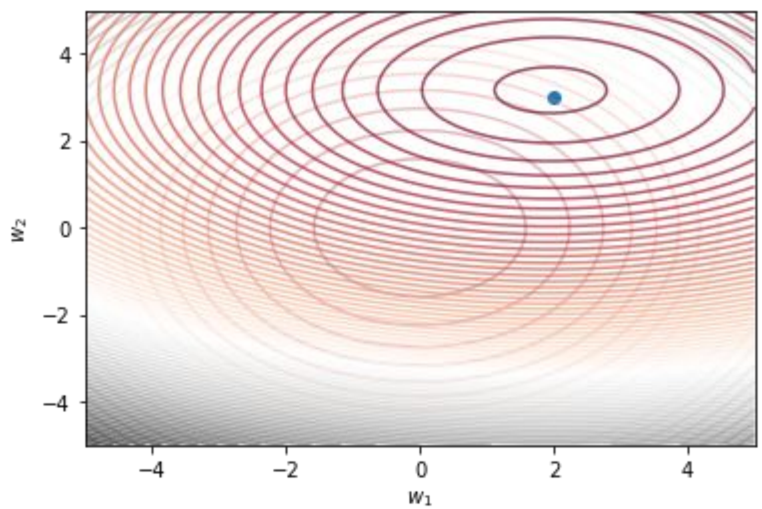


Bayesian Linear Regression



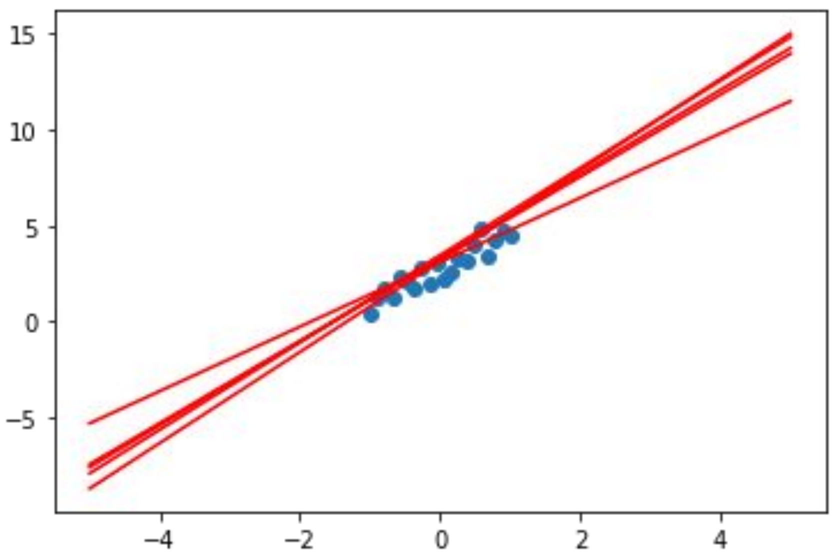


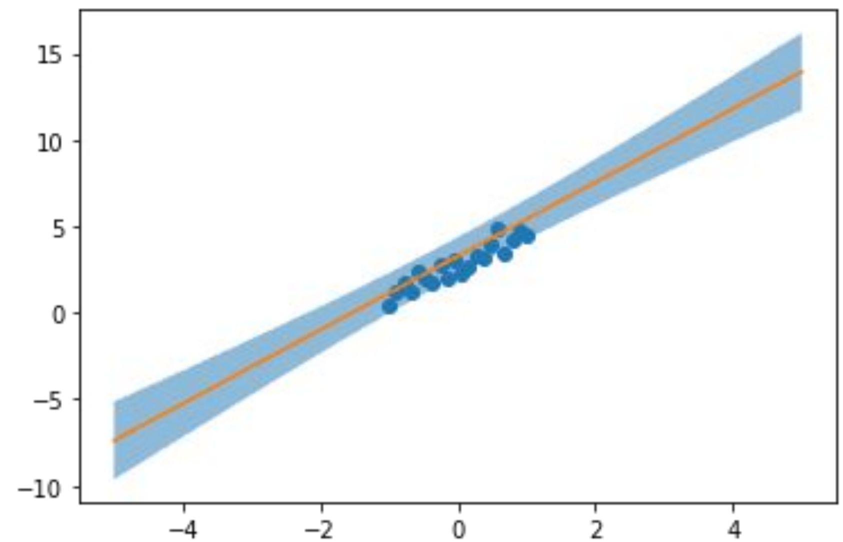
$$p(\theta|X,Y) = \frac{1}{Z} p(Y|X,\theta)\, p(\theta)$$



Bayesian Linear Regression

We have now a posterior for the weights w1 and w2 and I can generate from the weight space and I get several curves in the coordinate space



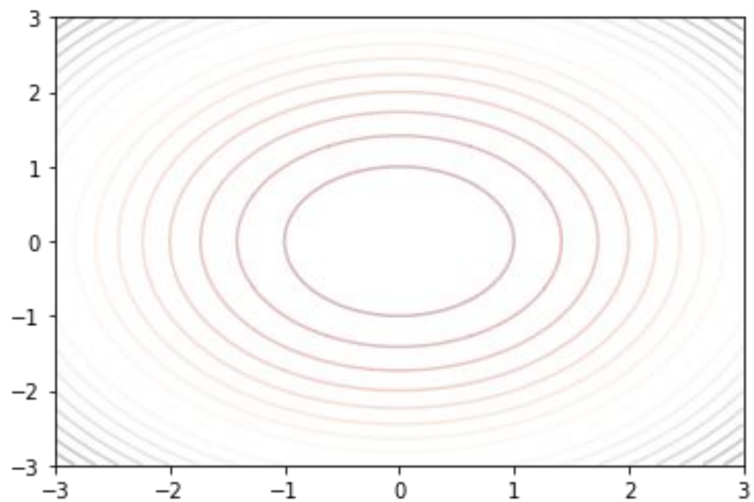


Markov Chain Monte Carlo (MCMC)

- Since the integral which gives the normalization might be intractable, numerical methods are fundamental

$$p\left(\theta | X_{train}, Y_{train}\right) = \frac{p\left(Y | X_{train}, \theta\right) p(\theta)}{\int p\left(Y | X_{train}, \theta\right) p(\theta)\, d\theta}$$

- MCMC is a random walk which aims to sampling from un unnormalized distribution
- It can be shown that after enough steps the sample does not depend on the initial value

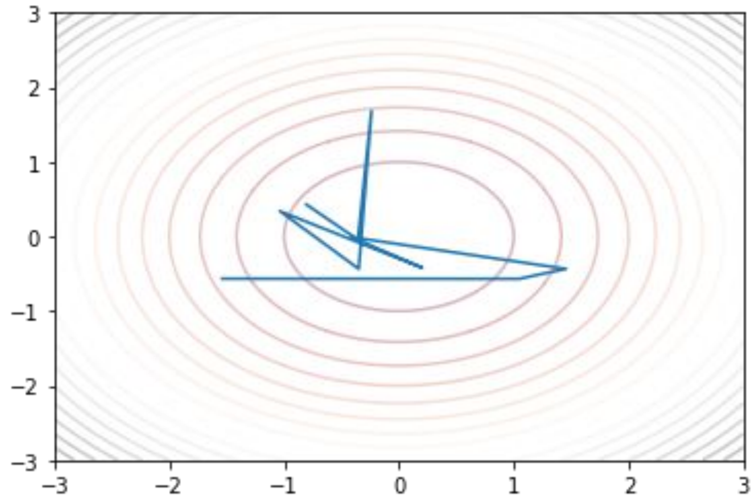


Markov Chain Monte Carlo (MCMC)

- Since the integral which gives the normalization might be intractable, numerical methods are fundamental

$$p\left(\theta | X_{train}, Y_{train}\right) = \frac{p\left(Y | X_{train}, \theta\right) p(\theta)}{\int p\left(Y | X_{train}, \theta\right) p(\theta)\, d\theta}$$

- MCMC is a random walk which aims to sampling from un unnormalized distribution
- It can be shown that after enough steps the sample does not depend on the initial value

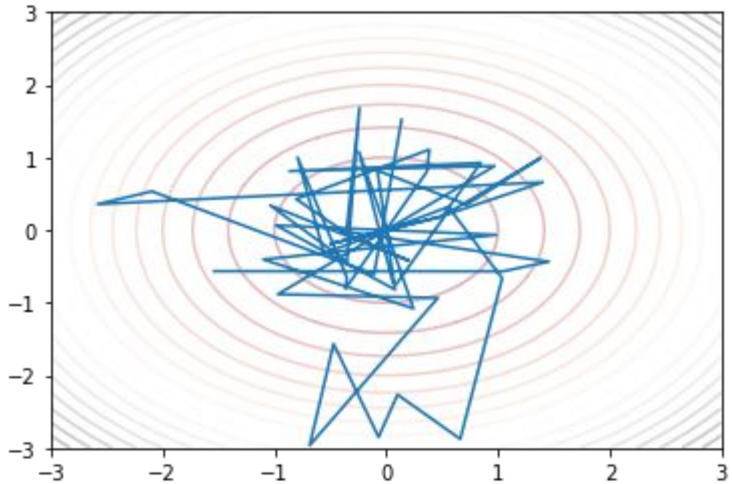


Markov Chain Monte Carlo (MCMC)

- Since the integral which gives the normalization might be intractable, numerical methods are fundamental

$$p\left(\theta|X_{train}, Y_{train}\right) = \frac{p\left(Y|X_{train}, \theta\right)p(\theta)}{\int p\left(Y|X_{train}, \theta\right)p(\theta)\, d\theta}$$

- MCMC is a random walk which aims to sampling from un unnormalized distribution
- It can be shown that after enough steps the sample does not depend on the initial value

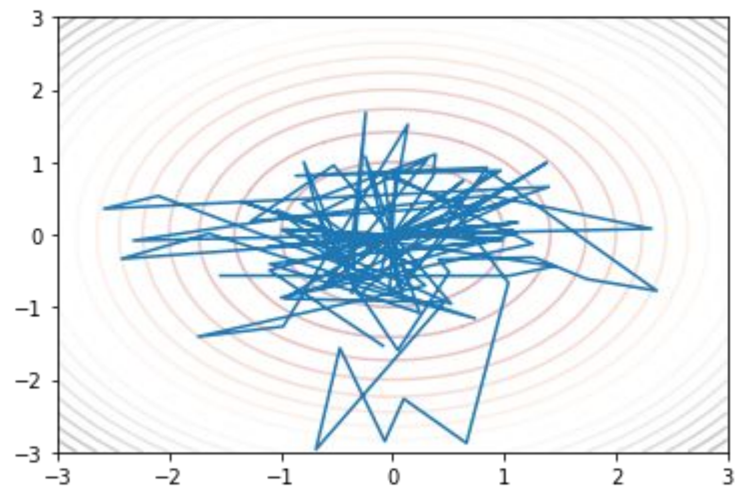


Markov Chain Monte Carlo (MCMC)

- Since the integral which gives the normalization might be intractable, numerical methods are fundamental

$$p\left(\theta|X_{train}, Y_{train}\right) = \frac{p\left(Y|X_{train}, \theta\right) p(\theta)}{\int p\left(Y|X_{train}, \theta\right) p(\theta)\, d\theta}$$

- MCMC is a random walk which aims to sampling from un unnormalized distribution
- It can be shown that after enough steps the sample does not depend on the initial value

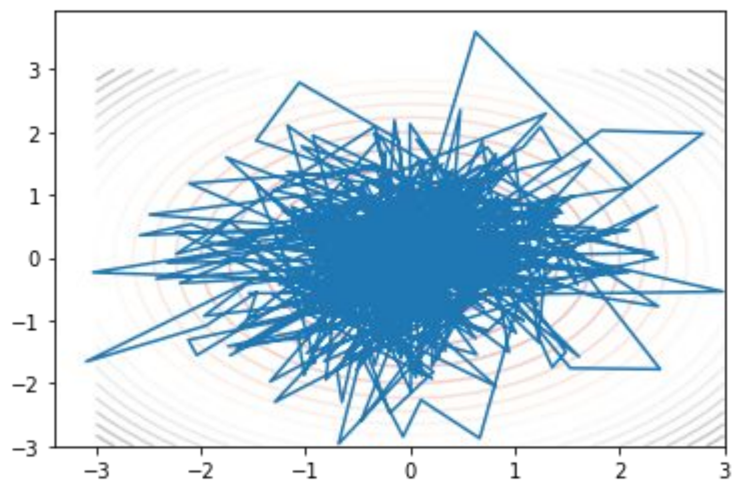


Markov Chain Monte Carlo (MCMC)

- Since the integral which gives the normalization might be intractable, numerical methods are fundamental

$$p(\theta|X_{train}, Y_{train}) = \frac{p(Y|X_{train}, \theta)\, p(\theta)}{\int p(Y|X_{train}, \theta)\, p(\theta)\, d\theta}$$

- MCMC is a random walk which aims to sampling from un unnormalized distribution
- It can be shown that after enough steps the sample does not depend on the initial value

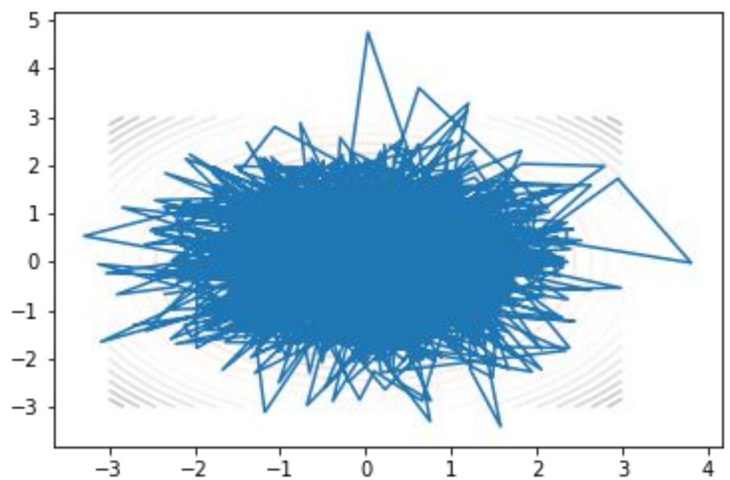


Markov Chain Monte Carlo (MCMC)

- Since the integral which gives the normalization might be intractable, numerical methods are fundamental

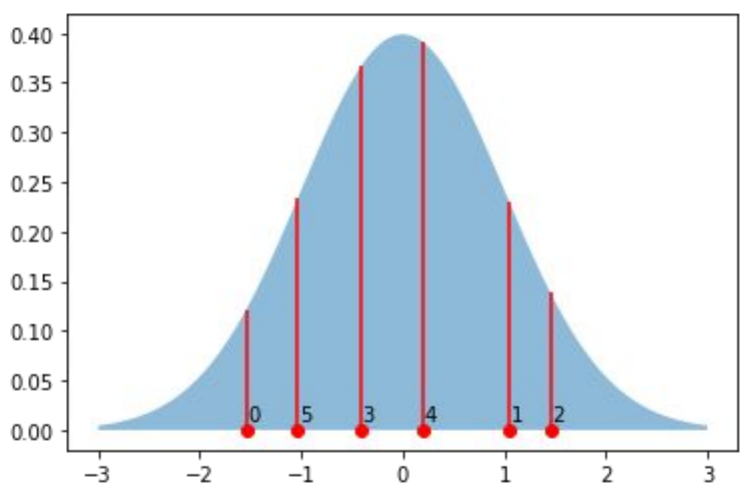
$$p\left(\theta | X_{train}, Y_{train} \right) = \frac{p\left(Y | X_{train}, \theta \right) p(\theta)}{\int p\left(Y | X_{train}, \theta \right) p(\theta)\, d\theta}$$

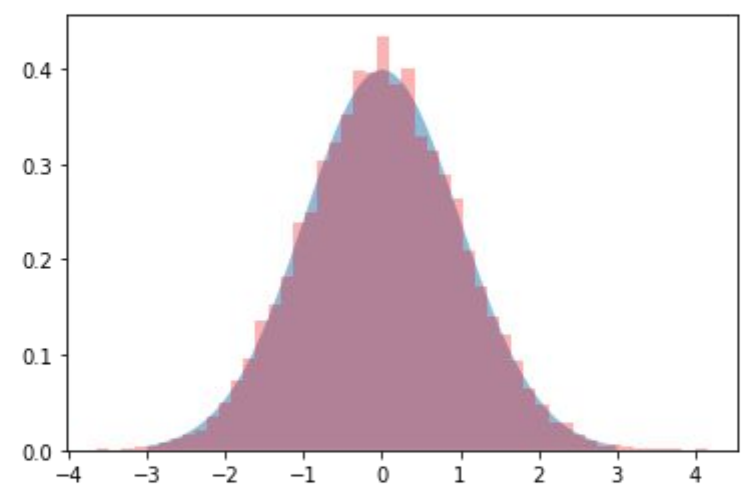
- MCMC is a random walk which aims to sampling from un unnormalized distribution
- It can be shown that after enough steps the sample does not depend on the initial value



Markov Chain Monte Carlo (MCMC)

- The random walk Metropolis-Hastings algorithm consists of:
    - Current state
    - Proposal is a gaussian function with a given sigma (hyper parameter)
    - Probability Accept = Prob. Proposal / Prob. Current





- There are many other more sophisticated and efficient methods to generate numbers , e.g. the No U-Turn Sample

Variational Inference

- The problem with MCMC techniques is that (while lots of progresses have been made) they in general need large samples and you quickly run into the course of dimensionality
- This is particularly problematic for Machine Learning when the space has typically very large dimensionality
- Variational Inference: approximate the posterior with a parametric distribution

$$p(\theta|x) \approx q(\theta) \in Q$$

- Need to have a good parametrization of the posterior
- Fast and scalable
- Our objective is to minimize the KL divergence

$$min_{q(\theta) \in Q} \quad KL(q(\theta) || p(\theta|x))$$

- For optimizing the previous formula we need to know the posterior, which is what we are looking for

MISIS National University of Science and Technology

University of Zurich UZH

Variational Inference

- It can be shown that

$$\log p(x) = \mathcal{L}(q(\theta)) + KL(q(\theta)||p(\theta|x))$$

Not dependent on q(θ)

Evidence Lower Bound (ELBO)

What we want to minimize

- To minimize the quantity we care about we can just maximize the ELBO

$$\mathcal{L}(q(\theta)) = E_{q(\theta)} \log p(x|\theta) - KL(q(\theta)||p(\theta))$$

Data part

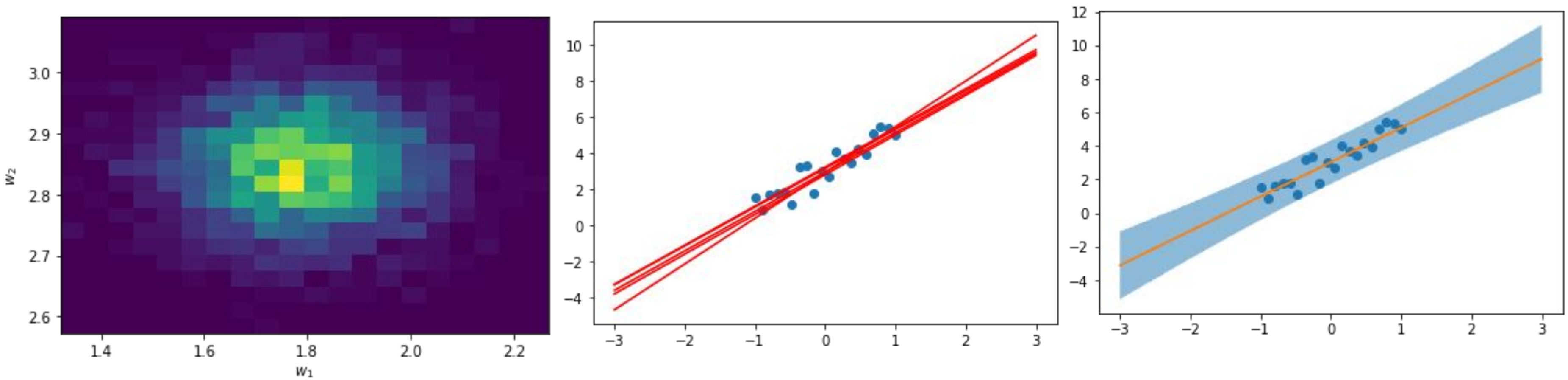
Modeling part

Variational Inference Example

- Let's consider the previous example of bayesian linear regression and instead of using MCMC to calculate the posterior we model w1
- We are modeling the fit with the formula

$$y = w_1 x + w_2 + \varepsilon$$

- We model our posterior as multivariate gaussian distribution



Probabilistic Programming Language

- There are several Probabilistic Programming Languages (PPL) that can be used to do bayesian inference, here you have a few useful packages in python

- Numpyro based on jax
- Pyro based on Pytorch

- PyMC3
- Tensorflow Probability