Priors on Functions

Mark van der Wilk

Department of Computing
Imperial College London

@markvanderwilk
m.vdwilk@imperial.ac.uk

January 29, 2021

Part II: Gaussian Processes

Now that we know how to specify and manipulate probabilistic models, let's look at a real one:

Gaussian Processes

Gaussian process: A probability distribution on functions, which is useful in regression tasks.

Part II: Gaussian Processes

Now that we know how to specify and manipulate probabilistic models, let's look at a real one:

Gaussian Processes

Gaussian process: A probability distribution on functions, which is useful in regression tasks.

GPs are considered the "gold standard" for uncertainty-aware regression. Practically, they excel in tasks that are

Part II: Gaussian Processes

Now that we know how to specify and manipulate probabilistic models, let's look at a real one:

Gaussian Processes

Gaussian process: A probability distribution on functions, which is useful in regression tasks.

GPs are considered the "gold standard" for uncertainty-aware regression. Practically, they excel in tasks that are

- are low dimensional (e.g. tens of dimensions rather than 100s),

Part II: Gaussian Processes

Now that we know how to specify and manipulate probabilistic models, let's look at a real one:

Gaussian Processes

Gaussian process: A probability distribution on functions, which is useful in regression tasks.

GPs are considered the "gold standard" for uncertainty-aware regression. Practically, they excel in tasks that are

- are low dimensional (e.g. tens of dimensions rather than 100s),
- have little data (or data is expensive to obtain),

Part II: Gaussian Processes

Now that we know how to specify and manipulate probabilistic models, let's look at a real one:

Gaussian Processes

Gaussian process: A probability distribution on functions, which is useful in regression tasks.

GPs are considered the "gold standard" for uncertainty-aware regression. Practically, they excel in tasks that are

- are low dimensional (e.g. tens of dimensions rather than 100s),
- have little data (or data is expensive to obtain),
- are noisy (random fluctuations that obscure the signal),

Part II: Gaussian Processes

Now that we know how to specify and manipulate probabilistic models, let's look at a real one:

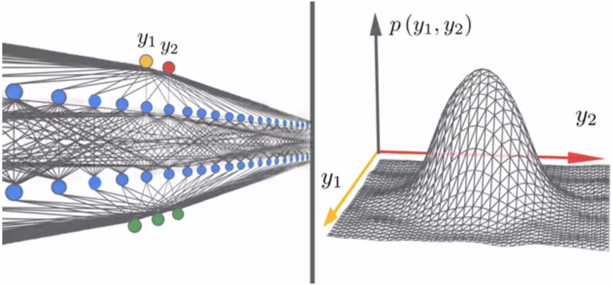
Gaussian Processes

Gaussian process: A probability distribution on functions, which is useful in regression tasks.

GPs are considered the "gold standard" for uncertainty-aware regression. Practically, they excel in tasks that are

- are low dimensional (e.g. tens of dimensions rather than 100s),
- have little data (or data is expensive to obtain),
- are noisy (random fluctuations that obscure the signal),
- require uncertainty estimates.

Application Areas

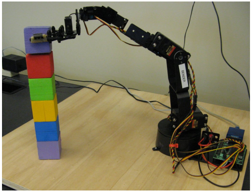


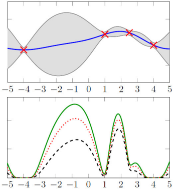
From: Fast and Easy Infinitely Wide Networks with Neural Tangents

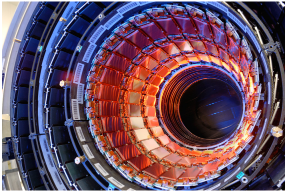
Applied widely in

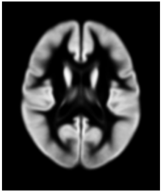
- statistics (epidemiology, mineral deposits, gene expression, ...)
- ML (behaviour of dynamical systems, analysis of DNNs)

Application Areas









- Reinforcement learning and robotics
- Bayesian optimization (experimental design)
- Geostatistics
- Sensor networks
- Time-series modeling and forecasting
- High-energy physics
- Medical applications

Part II: Teaching Aims

- Concrete illustration of how to specify Bayesian model

Part II: Teaching Aims

- Concrete illustration of how to specify Bayesian model
- Show features of Bayesian inference

Part II: Teaching Aims

- Concrete illustration of how to specify Bayesian model
- Show features of Bayesian inference
  - Uncertainty

Part II: Teaching Aims

- Concrete illustration of how to specify Bayesian model
- Show features of Bayesian inference
  - Uncertainty
  - Automatic (less trial-and-error)

Part II: Teaching Aims

- Concrete illustration of how to specify Bayesian model
- Show features of Bayesian inference
  - Uncertainty
  - Automatic (less trial-and-error)
- Show influence of the prior

Part II: Gaussian Processes

- Gaussian processes are nothing else than a **different representation** of Bayesian Linear Regression.

Part II: Gaussian Processes

- Gaussian processes are nothing else than a **different representation** of Bayesian Linear Regression.
- I like to say that they are a different representation of a neural network layer.

Part II: Gaussian Processes

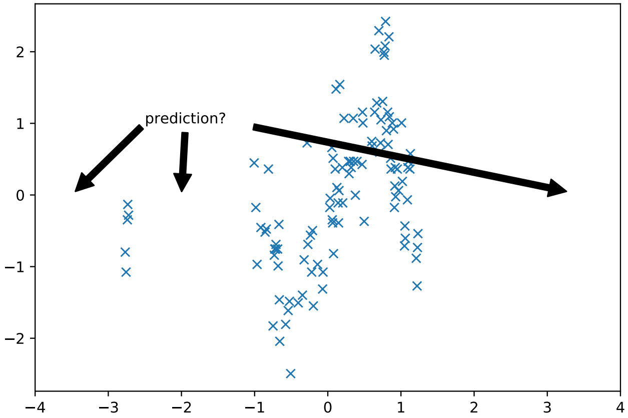
- Gaussian processes are nothing else than a **different representation** of Bayesian Linear Regression.
- I like to say that they are a different representation of a neural network layer.

This representation improves on Bayesian Linear Regression by:

- making it easier to specify sensible prior distributions (remember, our inferences are only as sensible as our prior assumptions!),
- providing better uncertainty estimates by allowing an infinite number of basis functions.

Regression

Curve fitting in 1D. Inputs $\in \mathbb{R}$, outputs $\in \mathbb{R}$:



Goal: Find $f : \mathbb{R}^D \rightarrow \mathbb{R}$ from example pairs $\{\mathbf{x}_n, y_n\}_{n=1}^N$.

Maximum Likelihood Polynomial Regression

Approach:
Find $f(\cdot)$ that at last goes as close as possible to the training data.

Maximum Likelihood Polynomial Regression

Approach:
Find $f(\cdot)$ that at last goes as close as possible to the training data.

1. Parameterise set of functions

$$f(\mathbf{x}) = \boldsymbol{\phi}(\mathbf{x})^\mathsf{T}\boldsymbol{\theta} \tag{1}$$

$$\text{e.g. } f(x) = \sum_{d=0}^{D} x^d \theta_d \tag{2}$$

Maximum Likelihood Polynomial Regression

Approach:
Find $f(\cdot)$ that at last goes as close as possible to the training data.

1. Parameterise set of functions

$$f(\mathbf{x}) = \boldsymbol{\phi}(\mathbf{x})^\top \boldsymbol{\theta} \tag{1}$$

$$\text{e.g. } f(x) = \sum_{d=0}^{D} x^d \theta_d \tag{2}$$

2. Maximise likelihood:

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \log p(\mathbf{y}|\boldsymbol{\theta}) = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \sum_{n=1}^{N} \log p(y_n|\boldsymbol{\theta}) \tag{3}$$

Maximum Likelihood Polynomial Regression

Approach:
Find $f(\cdot)$ that at last goes as close as possible to the training data.

1. Parameterise set of functions

$$f(\mathbf{x}) = \boldsymbol{\phi}(\mathbf{x})^\top \boldsymbol{\theta} \tag{1}$$

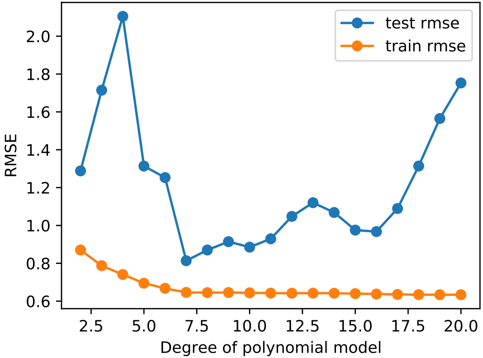
$$\text{e.g. } f(x) = \sum_{d=0}^{D} x^d \theta_d \tag{2}$$

2. Maximise likelihood:

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \log p(\mathbf{y}|\boldsymbol{\theta}) = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \sum_{n=1}^{N} \log p(y_n|\boldsymbol{\theta}) \tag{3}$$

Warning: can overfit.

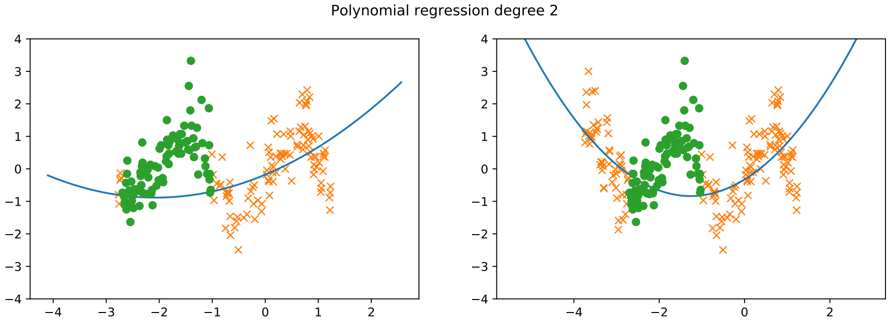
Reminder: Overfitting



As the degree of the polynomial gets higher:

- training error goes down, as we only get more flexibility to fit the training data,
- test error goes up, as we fit to irregularities in the training data rather than trend.

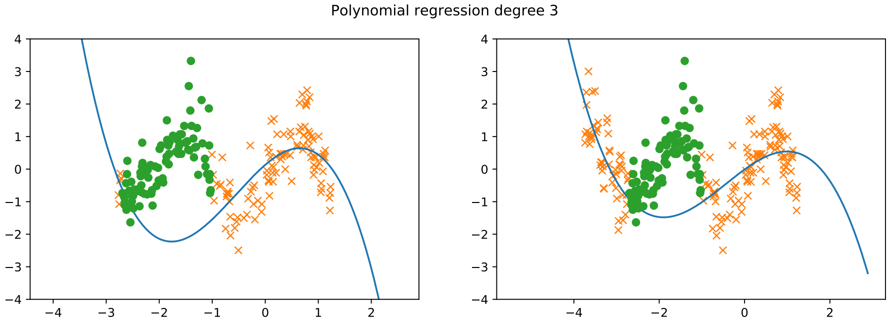
Maximum Likelihood Polynomial Regression



What degree of polynomial should I use? There is no good answer:

- Extrapolation to left always too extreme and wrong!
- Lower degree polynomials have less extreme interpolations, but less flexibility to fit additional data.
- Higher degree polynomials interpolate in much more extreme ways, but can fit the additional data...

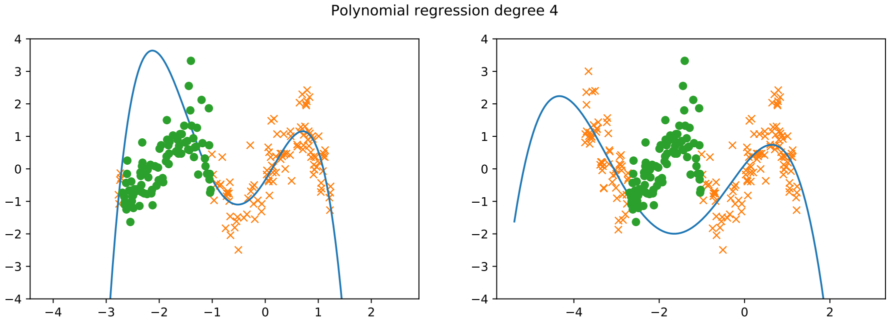
Maximum Likelihood Polynomial Regression



What degree of polynomial should I use? There is no good answer:

- Extrapolation to left always too extreme and wrong!
- Lower degree polynomials have less extreme interpolations, but less flexibility to fit additional data.
- Higher degree polynomials interpolate in much more extreme ways, but can fit the additional data...

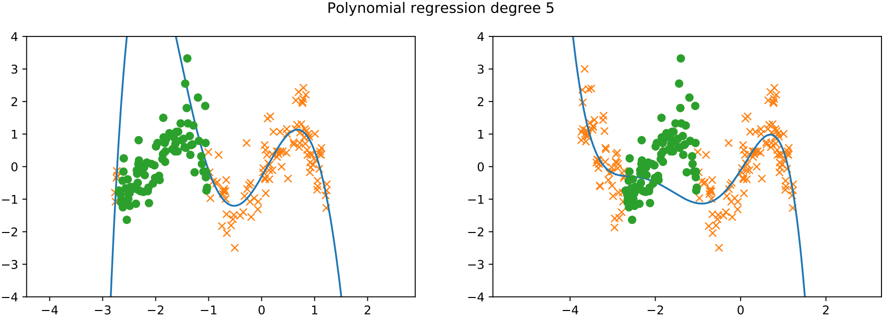
Maximum Likelihood Polynomial Regression



What degree of polynomial should I use? There is no good answer:

- Extrapolation to left always too extreme and wrong!
- Lower degree polynomials have less extreme interpolations, but less flexibility to fit additional data.
- Higher degree polynomials interpolate in much more extreme ways, but can fit the additional data...

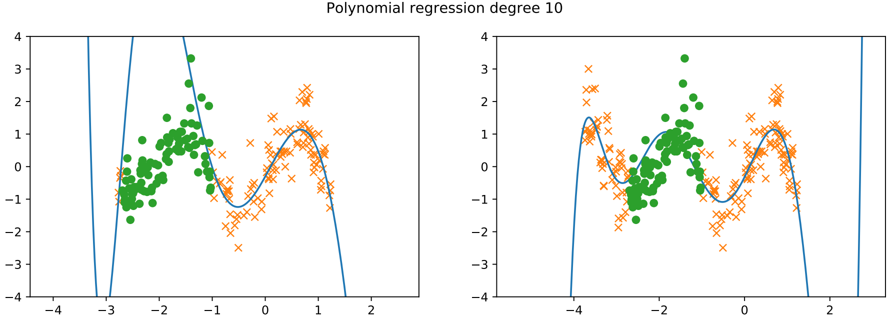
Maximum Likelihood Polynomial Regression



What degree of polynomial should I use? There is no good answer:

- Extrapolation to left always too extreme and wrong!
- Lower degree polynomials have less extreme interpolations, but less flexibility to fit additional data.
- Higher degree polynomials interpolate in much more extreme ways, but can fit the additional data...

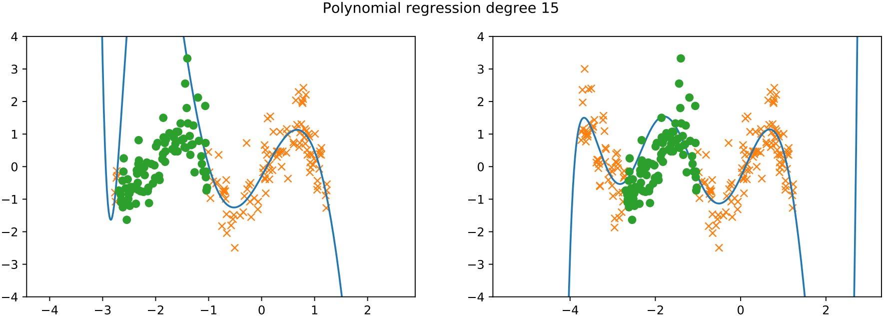
Maximum Likelihood Polynomial Regression



What degree of polynomial should I use? There is no good answer:

- Extrapolation to left always too extreme and wrong!
- Lower degree polynomials have less extreme interpolations, but less flexibility to fit additional data.
- Higher degree polynomials interpolate in much more extreme ways, but can fit the additional data...

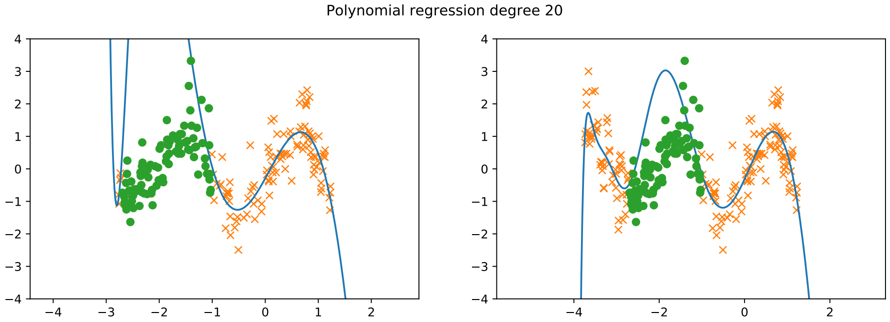
Maximum Likelihood Polynomial Regression



What degree of polynomial should I use? There is no good answer:

- Extrapolation to left always too extreme and wrong!
- Lower degree polynomials have less extreme interpolations, but less flexibility to fit additional data.
- Higher degree polynomials interpolate in much more extreme ways, but can fit the additional data...

Maximum Likelihood Polynomial Regression



What degree of polynomial should I use? There is no good answer:

- Extrapolation to left always too extreme and wrong!
- Lower degree polynomials have less extreme interpolations, but less flexibility to fit additional data.
- Higher degree polynomials interpolate in much more extreme ways, but can fit the additional data...

Maximum Likelihood Polynomial Regression

Two problems:

- Extrapolation always too extreme and wrong.
- No single model order worked well for both sets of training data.

Maximum Likelihood Polynomial Regression

Two problems:

- Extrapolation always too extreme and wrong.
- No single model order worked well for both sets of training data.

There are lots of problems. Is it only overfitting? That's definitely one of the problems...

Maximum Likelihood Polynomial Regression

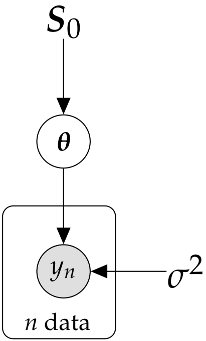
Two problems:

- Extrapolation always too extreme and wrong.
- No single model order worked well for both sets of training data.

There are lots of problems. Is it only overfitting? That's definitely one of the problems...

People say that Bayesian inference helps against overfitting!
Let's try that.

Bayesian Linear Regression: Model



Prior $\quad p(\boldsymbol{\theta}) = \mathcal{N}\big(\boldsymbol{m}_0,\, \boldsymbol{S}_0\big)$

Likelihood $\quad p(y_n|\boldsymbol{x}_n, \boldsymbol{\theta}) = \mathcal{N}\big(y_n \,|\, \boldsymbol{\phi}^\top(\boldsymbol{x}_n)\boldsymbol{\theta},\, \sigma^2\big)$

$$\implies y_n = \boldsymbol{\phi}^\top(\boldsymbol{x}_n)\boldsymbol{\theta} + \epsilon_n, \quad \epsilon_n \overset{\text{iid}}{\sim} \mathcal{N}\big(0,\, \sigma^2\big)$$

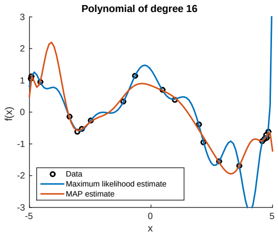
- Parameter $\boldsymbol{\theta}$ becomes a latent (random) variable
- Distribution $p(\boldsymbol{\theta})$ induces a distribution over plausible functions
- Choose a conjugate Gaussian prior
  - Closed-form computations
  - Gaussian posterior

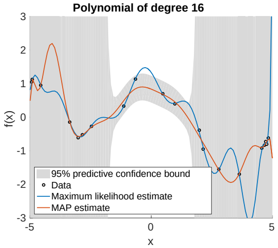
Bayesian Linear Regression: Procedure

- For predictions we need:

$$p(y_*|x_*, \mathbf{y}, \boldsymbol{X}) = \int p(y_*|x_*, \boldsymbol{\theta})p(\boldsymbol{\theta}|\mathbf{y}, \boldsymbol{X})\mathrm{d}\boldsymbol{\theta}$$

Where $\boldsymbol{X} \in \mathbb{R}^{N\times D}$ are the stacked inputs, and $\mathbf{y} \in \mathbb{R}^{N}$ are the stacked outputs.





Bayesian Linear Regression: Procedure

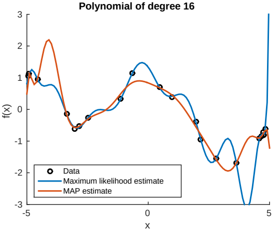
- For predictions we need:

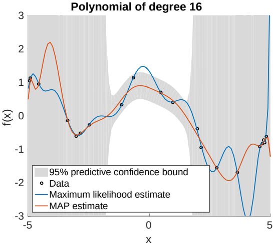
$$p(y_*|x_*, \mathbf{y}, \boldsymbol{X}) = \int p(y_*|x_*, \boldsymbol{\theta})p(\boldsymbol{\theta}|\mathbf{y}, \boldsymbol{X})\mathrm{d}\boldsymbol{\theta}$$

Where $\boldsymbol{X} \in \mathbb{R}^{N\times D}$ are the stacked inputs, and $\mathbf{y} \in \mathbb{R}^N$ are the stacked outputs.

- Find posterior; use to find predictive.

$$p(\boldsymbol{\theta}|\mathbf{y}, \boldsymbol{X}) = \frac{\prod_{n=1}^N p(y_n|\boldsymbol{\theta}, \mathbf{x}_n)p(\boldsymbol{\theta})}{p(\mathbf{y}|\boldsymbol{X})}$$





Bayesian Linear Regression: Procedure

- For predictions we need:

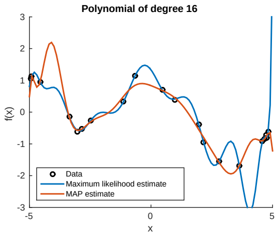
$$p(y_*|x_*, \mathbf{y}, \boldsymbol{X}) = \int p(y_*|x_*, \boldsymbol{\theta}) p(\boldsymbol{\theta}|\mathbf{y}, \boldsymbol{X}) \mathrm{d}\boldsymbol{\theta}$$

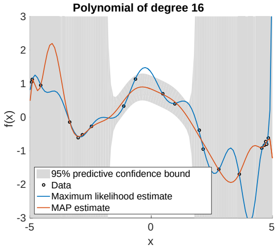
  Where $\boldsymbol{X} \in \mathbb{R}^{N \times D}$ are the stacked inputs, and $\mathbf{y} \in \mathbb{R}^N$ are the stacked outputs.

- Find posterior; use to find predictive.

$$p(\boldsymbol{\theta}|\mathbf{y}, \boldsymbol{X}) = \frac{\prod_{n=1}^{N} p(y_n|\boldsymbol{\theta}, \mathbf{x}_n) p(\boldsymbol{\theta})}{p(\mathbf{y}|\boldsymbol{X})}$$

- Prior and posterior induce distribution over functions.





Bayesian Linear Regression: Procedure

- For predictions we need:

$$p(y_*|x_*, \mathbf{y}, \boldsymbol{X}) = \int p(y_*|x_*, \boldsymbol{\theta})p(\boldsymbol{\theta}|\mathbf{y}, \boldsymbol{X})\mathrm{d}\boldsymbol{\theta}$$

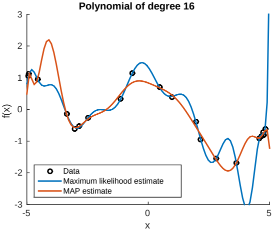
  Where $\boldsymbol{X} \in \mathbb{R}^{N\times D}$ are the stacked inputs, and $\mathbf{y} \in \mathbb{R}^N$ are the stacked outputs.

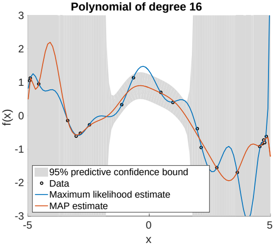
- Find posterior; use to find predictive.

$$p(\boldsymbol{\theta}|\mathbf{y}, \boldsymbol{X}) = \frac{\prod_{n=1}^{N} p(y_n|\boldsymbol{\theta}, \mathbf{x}_n)p(\boldsymbol{\theta})}{p(\mathbf{y}|\boldsymbol{X})}$$

- Prior and posterior induce distribution over functions.

Exercise: Derive everything above carefully from the model definition. Try w/ graphical model. Find the pred. dist. in terms of the post mean and var. (discuss on Weds).





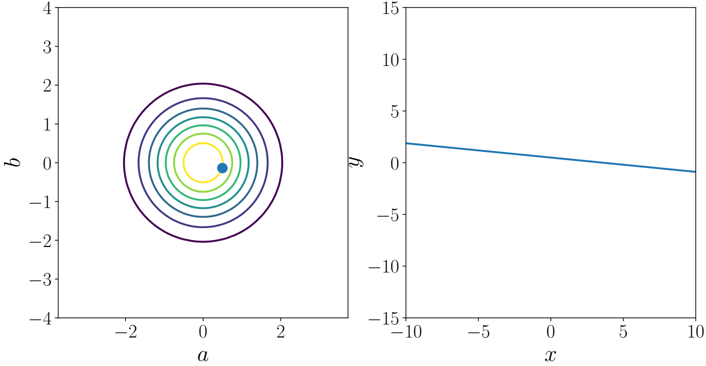
Sampling from the Prior over Functions

Consider a linear regression setting

$$y = f(x) + \epsilon = a + bx + \epsilon\,, \quad \epsilon \sim \mathcal{N}\left(0,\, \sigma_n^2\right)$$

$$p(a, b) = \mathcal{N}\left(\mathbf{0},\, \boldsymbol{I}\right)$$

$$f_i(x) = a_i + b_i x, \quad [a_i, b_i] \sim p(a, b)$$



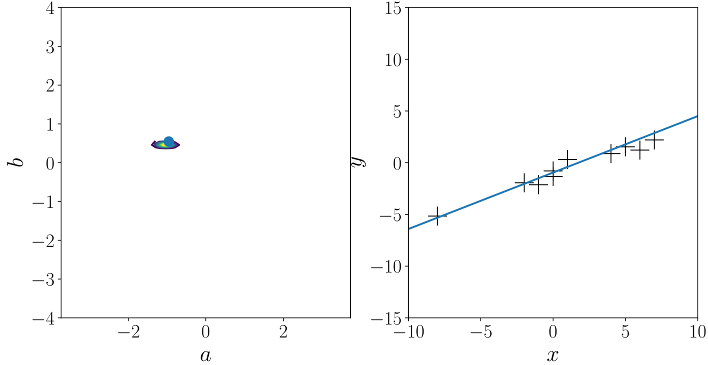
Sampling from the Posterior over Functions

Consider a linear regression setting

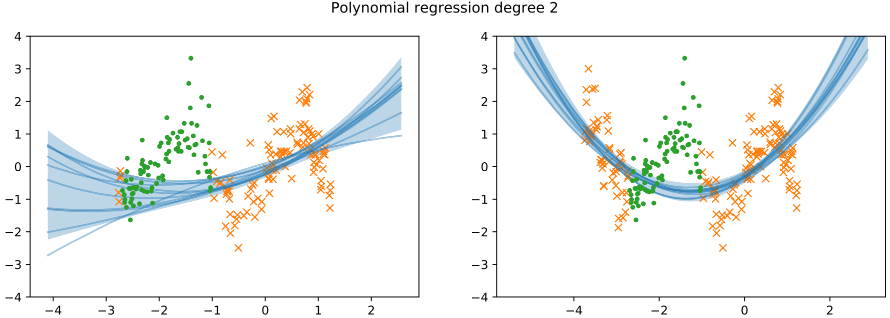
$$y = f(x) + \epsilon = a + bx + \epsilon\,, \quad \epsilon \sim \mathcal{N}\left(0, \sigma_n^2\right)$$

$$[a_i, b_i] \sim p(a, b|\boldsymbol{X}, \boldsymbol{y})$$

$$f_i = a_i + b_i x$$

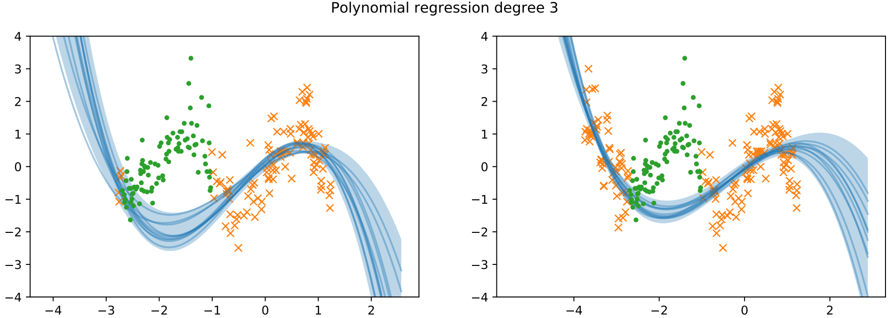


Bayesian Polynomial Regression



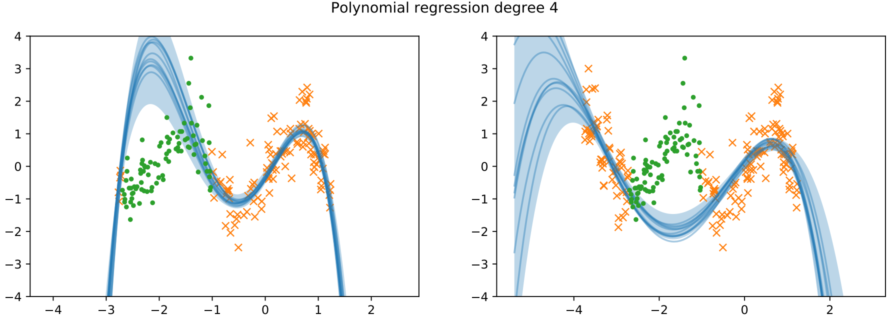
- Improvement: We see large uncertainty when extrapolating.
- Extrapolations are still huge...
- Still not particularly great behaviour in the interpolation.
- Somehow it seems that the model is... struggling.

Bayesian Polynomial Regression



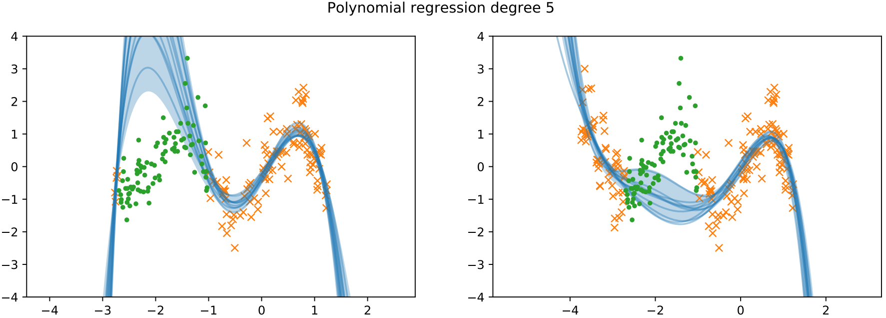
- Improvement: We see large uncertainty when extrapolating.
- Extrapolations are still huge...
- Still not particularly great behaviour in the interpolation.
- Somehow it seems that the model is... struggling.

Bayesian Polynomial Regression



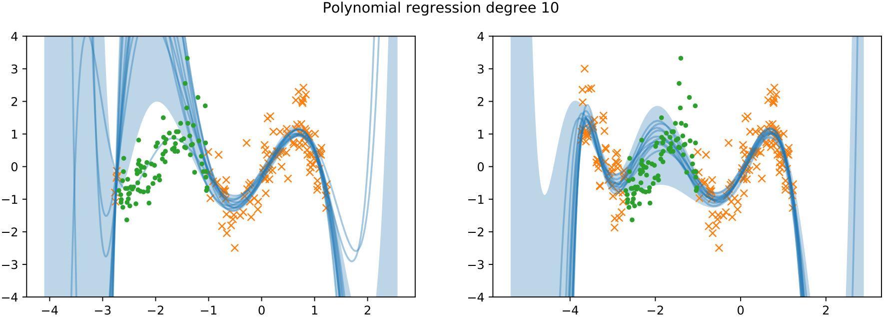
- Improvement: We see large uncertainty when extrapolating.
- Extrapolations are still huge...
- Still not particularly great behaviour in the interpolation.
- Somehow it seems that the model is... struggling.

Bayesian Polynomial Regression



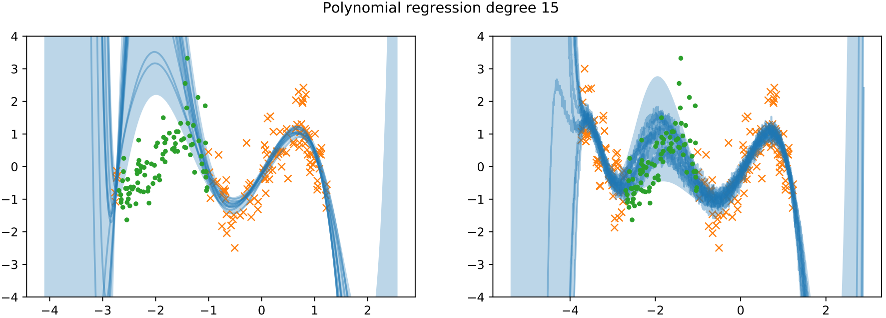
- Improvement: We see large uncertainty when extrapolating.
- Extrapolations are still huge...
- Still not particularly great behaviour in the interpolation.
- Somehow it seems that the model is... struggling.

Bayesian Polynomial Regression



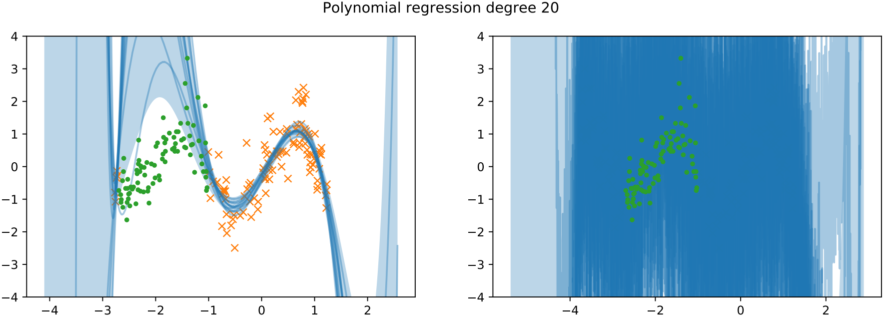
- Improvement: We see large uncertainty when extrapolating.
- Extrapolations are still huge...
- Still not particularly great behaviour in the interpolation.
- Somehow it seems that the model is... struggling.

Bayesian Polynomial Regression



- Improvement: We see large uncertainty when extrapolating.
- Extrapolations are still huge...
- Still not particularly great behaviour in the interpolation.
- Somehow it seems that the model is... struggling.

Bayesian Polynomial Regression



- Improvement: We see large uncertainty when extrapolating.
- Extrapolations are still huge...
- Still not particularly great behaviour in the interpolation.
- Somehow it seems that the model is... struggling.

Influence of the Prior on the Posterior

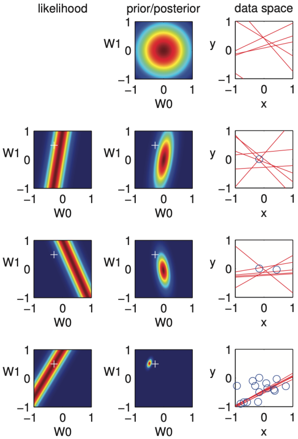
Why does the model make such extreme predictions?

Remember: Our inferences depend on our assumptions.
Incorrect assumptions lead to incorrect conclusions, even if our reasoning was correct!

$$p(\boldsymbol{\theta}|y_{1:n+1}) = \frac{p(y_{n+1}|\boldsymbol{\theta})p(\boldsymbol{\theta}|y_{1:n})}{Z} \qquad (4)$$

Exercise: Holds when $y_i \perp\!\!\!\perp y_j \,|\, \boldsymbol{\theta}, i \neq j$. Prove this. Find Z.

Each term in the likelihood "cuts away" spread from the prior.

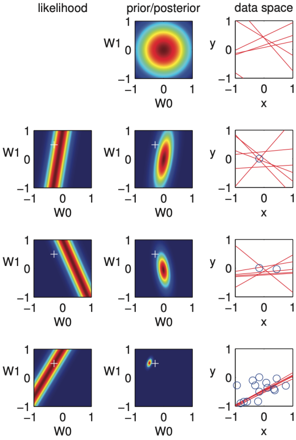


(Murphy (old), figure 7.11)

Influence of the Prior on the Posterior

Two ways that a model can be **misspecified** through the prior are:

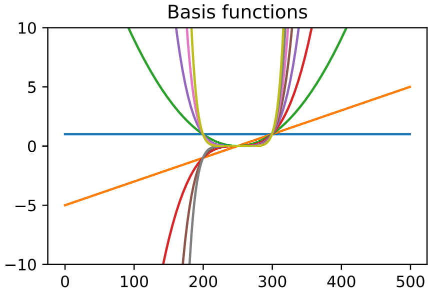
1. The prior does not give probability to good solutions.
   A posterior will never place probability on a set that has no probability under the prior.
2. The prior does places too much probability on bad solutions.
   It takes too long for the likelihood to cut out all the bad solutions.

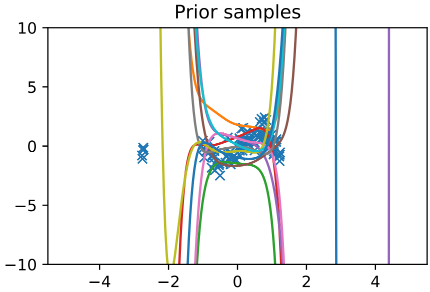


(Murphy (old), figure 7.11)

Investigating the Prior

Let's visualise our prior:

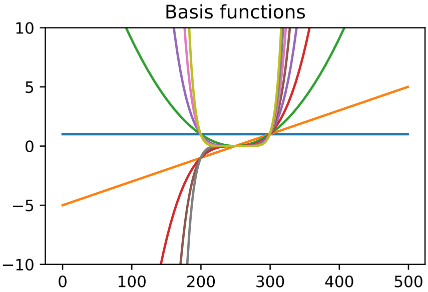


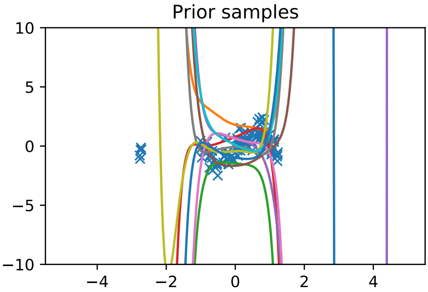


- Far too much probability on functions that increase rapidly.
- Not enough flexibility for variation in the data range.

Investigating the Prior

Let's visualise our prior:





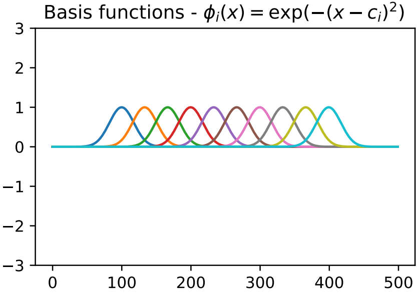
- Far too much probability on functions that increase rapidly.
- Not enough flexibility for variation in the data range.

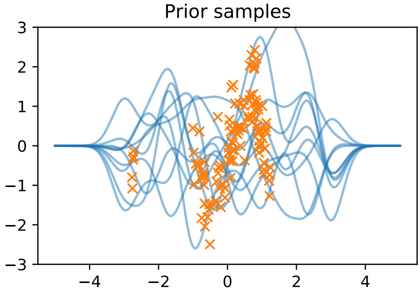
High-order polynomials **can** represent all (reasonable) functions, but our prior doesn't place the mass in the right place!

Squared Exponential Basis Functions

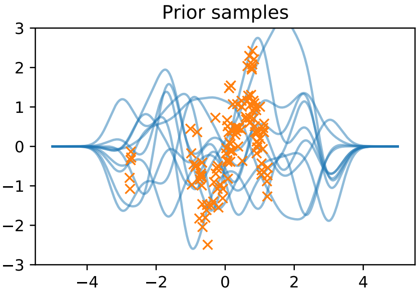
To fix the behaviour of our model, we make the prior more sensible by choosing different basis functions.

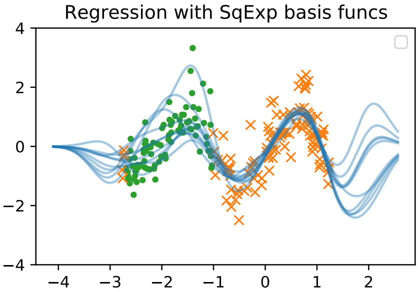
- We prevent wild extrapolations by choosing basis functions which are bounded in output value.
- We prevent sensitivity on distant values by choosing basis functions with a bounded input range where they have effect. (Not the only way to get this behaviour.)





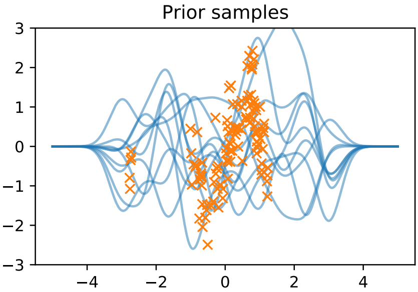
Squared Exponential Basis Functions

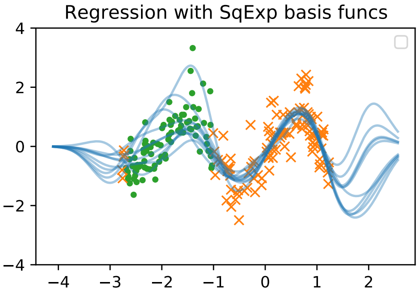




- Likelihood “cuts away” prior samples that don’t fit the data.
- Prior is matched better: more sensible posterior

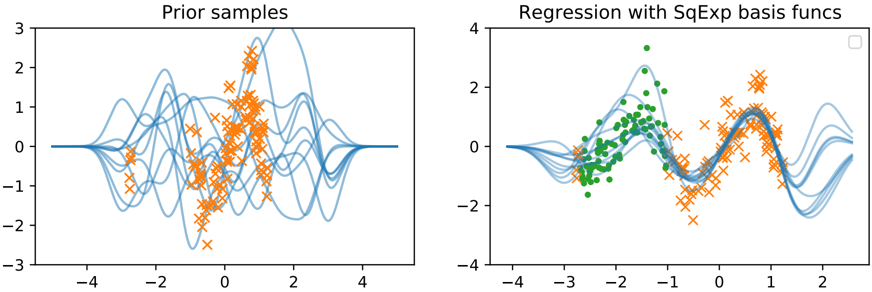
Squared Exponential Basis Functions





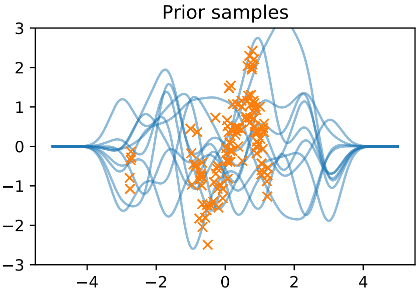
- Likelihood “cuts away” prior samples that don’t fit the data.
- Prior is matched better: more sensible posterior
- Interpolation is much better, but what about extrapolation?

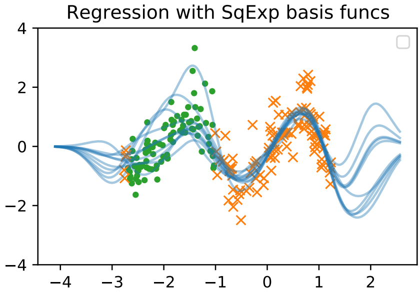
Squared Exponential Basis Functions



- Likelihood “cuts away” prior samples that don’t fit the data.
- Prior is matched better: more sensible posterior
- Interpolation is much better, but what about extrapolation?
- Prior is *super certain* that nothing can happen outside $[-4, 3]$!

Squared Exponential Basis Functions





- Likelihood "cuts away" prior samples that don't fit the data.
- Prior is matched better: more sensible posterior
- Interpolation is much better, but what about extrapolation?
- Prior is *super certain* that nothing can happen outside $[-4, 3]$!
- Not realistic. Can we not just put basis functions everywhere?

Summary

We saw:

- How assumptions in the prior influence the posterior.
- How poor assumptions can lead to poor behaviour (i.e. Bayes doesn't solve everything!).
- A way to specify a better prior on functions.
- That perhaps we need many, many (infinite) basis functions.

Code for plots:

`https://github.com/markvdw/inference-plots/blob/main/priors-o`

References I

[1] G. Bertone, M. P. Deisenroth, J. S. Kim, S. Liem, R. R. de Austri, and M. Welling. Accelerating the BSM Interpretation of LHC Data with Machine Learning. arXiv preprint arXiv:1611.02704, 2016.

[2] N. A. C. Cressie. *Statistics for Spatial Data*. Wiley-Interscience, 1993.

[3] M. Cutler and J. P. How. Efficient Reinforcement Learning for Robots using Informative Simulated Priors. In *Proceedings of the International Conference on Robotics and Automation*, 2015.

[4] M. P. Deisenroth, C. E. Rasmussen, and D. Fox. Learning to Control a Low-Cost Manipulator using Data-Efficient Reinforcement Learning. In *Proceedings of Robotics: Science and Systems*, 2011.

[5] M. P. Deisenroth, C. E. Rasmussen, and J. Peters. Gaussian Process Dynamic Programming. *Neurocomputing*, 72(7–9):1508–1524, Mar. 2009.

[6] M. P. Deisenroth, R. Turner, M. Huber, U. D. Hanebeck, and C. E. Rasmussen. Robust Filtering and Smoothing with Gaussian Processes. *IEEE Transactions on Automatic Control*, 57(7):1865–1871, 2012.

[7] R. Frigola, F. Lindsten, T. B. Schön, and C. E. Rasmussen. Bayesian Inference and Learning in Gaussian Process State-Space Models with Particle MCMC. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, pages 3156–3164. Curran Associates, Inc., 2013.

[8] A. Krause, A. Singh, and C. Guestrin. Near-Optimal Sensor Placements in Gaussian Processes: Theory, Efficient Algorithms and Empirical Studies. *Journal of Machine Learning Research*, 9:235–284, Feb. 2008.

[9] M. A. Osborne, S. J. Roberts, A. Rogers, S. D. Ramchurn, and N. R. Jennings. Towards Real-Time Information Processing of Sensor Network Data Using Computationally Efficient Multi-output Gaussian Processes. In *Proceedings of the International Conference on Information Processing in Sensor Networks*, pages 109–120. IEEE Computer Society, 2008.

[10] S. Roberts, M. A. Osborne, M. Ebden, S. Reece, N. Gibson, and S. Aigrain. Gaussian Processes for Time Series Modelling. *Philosophical Transactions of the Royal Society (Part A)*, 371(1984), Feb. 2013.