CSC2541 Lecture 2
Bayesian Occam's Razor and Gaussian Processes

Roger Grosse

Adminis-Trivia

- Did everyone get my e-mail last week?
  - If not, let me know.
  - You can find the announcement on Blackboard.
- Sign up on Piazza.
- Is everyone signed up for a presentation slot?
- Form project groups of 3–5. If you don't know people, try posting to Piazza.

Advice on Readings

- 4–6 readings per week, many are fairly mathematical
- They get lighter later in the term.
- Don't worry about learning every detail. Try to understand the main ideas so you know when you should refer to them.
    - What problem are they trying to solve? What is their contribution?
    - How does it relate to the other papers?
    - What evidence do they present? Is it convincing?

Advice on Readings

- 4–6 readings per week, many are fairly mathematical
- They get lighter later in the term.
- Don't worry about learning every detail. Try to understand the main ideas so you know when you should refer to them.
  - What problem are they trying to solve? What is their contribution?
  - How does it relate to the other papers?
  - What evidence do they present? Is it convincing?
- Reading mathematical material
  - You'll get to use software packages, so no need to go through line-by-line.
  - What assumptions are they making, and how are those used?
  - What is the main insight?
  - Formulas: if you change one variable, how do other things vary?
  - What guarantees do they obtain? How do those relate to the other algorithms we cover?
- Don't let it become a chore. I chose readings where you still get something from them even if you don't absorb every detail.

This Lecture

- Linear regression and smoothing splines
- Bayesian linear regression
- "Bayesian Occam's Razor"
- Gaussian processes
- We'll put off the Automatic Statistician for later

Function Approximation

- Many machine learning tasks can be viewed as function approximation, e.g.
  - object recognition (image $\rightarrow$ category)
  - speech recognition (waveform $\rightarrow$ text)
  - machine translation (French $\rightarrow$ English)
  - generative modeling (noise $\rightarrow$ image)
  - reinforcement learning (state $\rightarrow$ value, or state $\rightarrow$ action)
- In the last few years, neural nets have revolutionized all of these domains, since they're really good function approximators
- Much of this class will focus on being Bayesian about function approximation.

Review: Linear Regression

- Probably the simplest function approximator is linear regression. This is a useful starting point since we can solve and analyze it analytically.
- Given a training set of inputs and targets $\{(\mathbf{x}^{(i)}, t^{(i)})\}_{i=1}^N$
- Linear model:

$$y = \mathbf{w}^\top \mathbf{x} + b$$

- Squared error loss:

$$\mathcal{L}(y, t) = \frac{1}{2}(t - y)^2$$

- Solution 1: solve analytically by setting gradient to 0

$$\mathbf{w} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{t}$$

- Solution 2: solve approximately using gradient descent

$$\mathbf{w} \leftarrow \mathbf{w} - \alpha \mathbf{X}^\top (\mathbf{y} - \mathbf{t})$$

Nonlinear Regression: Basis Functions

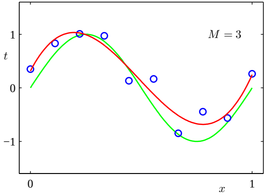
- We can model a function as linear in a set of basis functions (i.e. feature mapping):

$$y = \mathbf{w}^\top \phi(x)$$

- E.g., we can fit a degree-k polynomial using the mapping

$$\phi(\mathbf{x}) = (1, x, x^2, \ldots, x^k).$$

- Exactly the same algorithms/formulas as ordinary linear regression: just pretend $\phi(x)$ are the inputs!
- Best-fitting cubic polynomial:



— Bishop, Pattern Recognition and Machine Learning

- Before 2012, feature engineering was the hardest part of building many AI systems. Now it's done automatically with neural nets.

Nonlinear Regression: Smoothing Splines

- An alternative approach to nonlinear regression: fit an arbitrary function, but encourage it to be smooth.
- This is called a smoothing spline.

$$\mathcal{E}(f, \lambda) = \underbrace{\sum_{i=1}^{N} (t^{(i)} - f(x^{(i)}))^2}_{\text{mean squared error}} + \lambda \underbrace{\int (f''(z))^2 \, \mathrm{d}z}_{\text{regularizer}}$$

- What happens for $\lambda = 0$? $\lambda = \infty$?

Nonlinear Regression: Smoothing Splines

- An alternative approach to nonlinear regression: fit an arbitrary function, but encourage it to be smooth.
- This is called a smoothing spline.

$$\mathcal{E}(f, \lambda) = \underbrace{\sum_{i=1}^{N} (t^{(i)} - f(x^{(i)}))^2}_{\text{mean squared error}} + \lambda \underbrace{\int (f''(z))^2 \, \mathrm{d}z}_{\text{regularizer}}$$

- What happens for $\lambda = 0$? $\lambda = \infty$?
- Even though f is unconstrained, it turns out the optimal f can be expressed as a linear combination of (data-dependent) basis functions
  - I.e., algorithmically, it's just linear regression! (minus some numerical issues that we'll ignore)

Nonlinear Regression: Smoothing Splines

- Mathematically, we express f as a linear combination of basis functions:

$$f(x) = \sum_i w_i \phi_i(x) \qquad \mathbf{y} = f(\mathbf{x}) = \mathbf{\Phi w}$$

- Squared error term (just like in linear regression):

$$\|\mathbf{t} - \mathbf{\Phi w}\|^2$$

- Regularizer:

$$\begin{aligned}
\int (f''(z))^2 \, \mathrm{d}z &= \int \left(\sum_i w_i \phi_i(z) \right)^2 \, \mathrm{d}z \\
&= \int \sum_i \sum_j w_i w_j \, \phi_i''(z) \, \phi_j''(z) \, \mathrm{d}z \\
&= \sum_i \sum_j w_i w_j \underbrace{\int \phi_i''(z) \, \phi_j''(z) \, \mathrm{d}z}_{= \Omega_{ij}} \\
&= \mathbf{w}^\top \mathbf{\Omega w}
\end{aligned}$$

Nonlinear Regression: Smoothing Splines

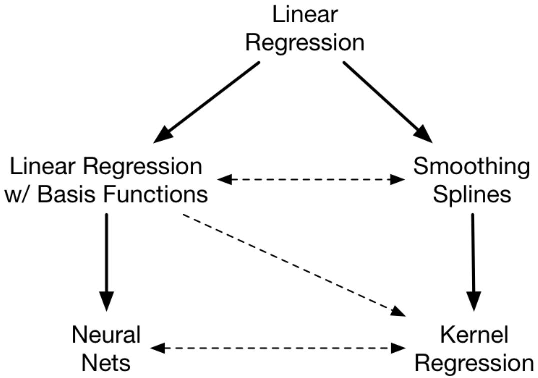
- Full cost function:

$$\mathcal{E}(\mathbf{w}, \lambda) = \|\mathbf{t} - \boldsymbol{\Phi}\mathbf{w}\|^2 + \lambda \mathbf{w}^\top \boldsymbol{\Omega} \mathbf{w}$$

- Optimal solution (derived by setting gradient to zero):

$$\mathbf{w} = (\boldsymbol{\Phi}^\top \boldsymbol{\Phi} + \lambda \boldsymbol{\Omega})^{-1} \boldsymbol{\Phi}^\top \mathbf{t}$$

Foreshadowing



Linear Regression as Maximum Likelihood

- We can give linear regression a probabilistic interpretation by assuming a Gaussian noise model:

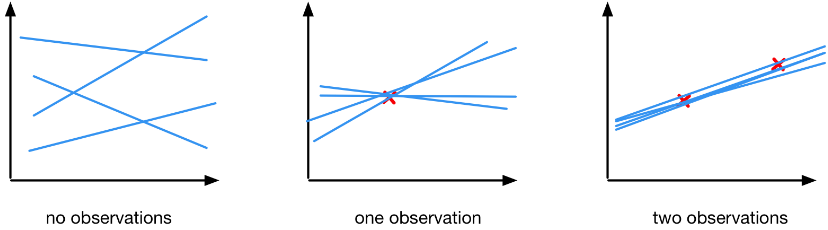
$$t \mid \mathbf{x} \sim \mathcal{N}(\mathbf{w}^\top \mathbf{x} + b,\ \sigma^2)$$

- Linear regression is just maximum likelihood under this model:

$$\begin{aligned}
\frac{1}{N} \sum_{i=1}^{N} \log p(t^{(i)} \mid \mathbf{x}^{(i)}; \mathbf{w}, b) &= \frac{1}{N} \sum_{i=1}^{N} \log \mathcal{N}(t^{(i)}; \mathbf{w}^\top \mathbf{x} + b, \sigma^2) \\
&= \frac{1}{N} \sum_{i=1}^{N} \log \left[\frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(t^{(i)} - \mathbf{w}^\top \mathbf{x} - b)^2}{2\sigma^2} \right) \right] \\
&= \mathrm{const} - \frac{1}{2N\sigma^2} \sum_{i=1}^{N} (t^{(i)} - \mathbf{w}^\top \mathbf{x} - b)^2
\end{aligned}$$

Bayesian Linear Regression

- Bayesian linear regression considers various plausible explanations for how the data were generated.
- It makes predictions using all possible regression weights, weighted by their posterior probability.



Bayesian Linear Regression

- Leave out the bias for simplicity
- **Prior distribution:** a broad, spherical (multivariate) Gaussian centered at zero:

$$\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \nu^2 \mathbf{I})$$

- **Likelihood:** same as in the maximum likelihood formulation:

$$t \,|\, \mathbf{x}, \mathbf{w} \sim \mathcal{N}(\mathbf{w}^\top \mathbf{x},\ \sigma^2)$$

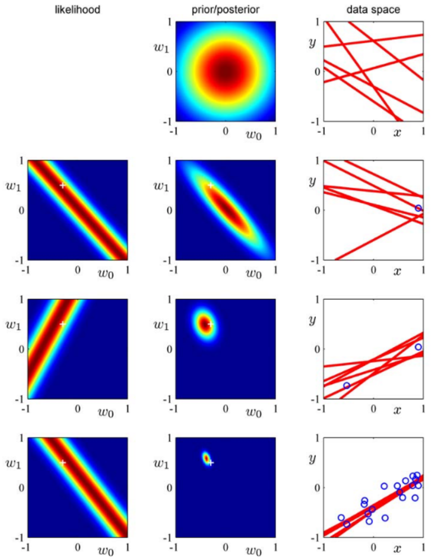
- **Posterior:**

$$\begin{aligned} \mathbf{w} \,|\, \mathcal{D} &\sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \\ \boldsymbol{\mu} &= \sigma^{-2} \boldsymbol{\Sigma} \mathbf{X}^\top \mathbf{t} \\ \boldsymbol{\Sigma}^{-1} &= \nu^{-2} \mathbf{I} + \sigma^{-2} \mathbf{X}^\top \mathbf{X} \end{aligned}$$

- Compare with linear regression formula:

$$\mathbf{w} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{t}$$

Bayesian Linear Regression

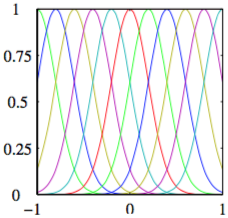


— Bishop, Pattern Recognition and Machine Learning

Bayesian Linear Regression

- We can turn this into nonlinear regression using basis functions.
- E.g., Gaussian basis functions

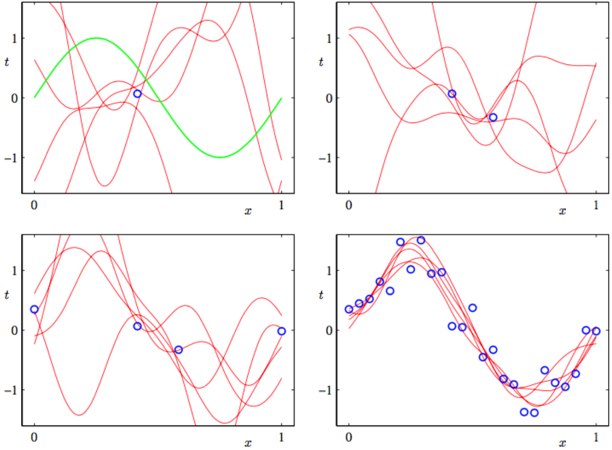
$$\phi_j(x) = \exp\left(-\frac{(x-\mu_j)^2}{2s^2}\right)$$



— Bishop, Pattern Recognition and Machine Learning

Bayesian Linear Regression

Functions sampled from the posterior:



— Bishop, Pattern Recognition and Machine Learning

Bayesian Linear Regression

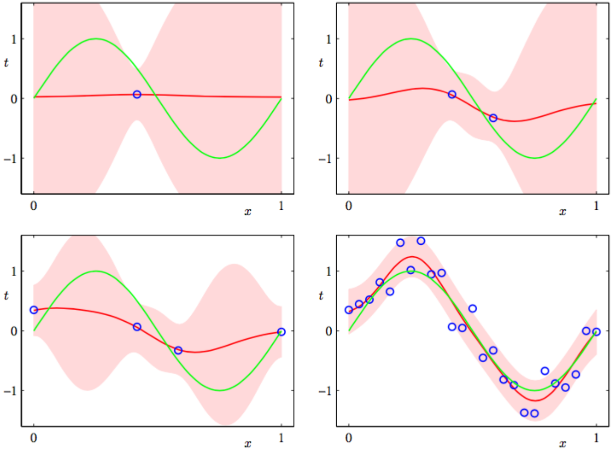
- Posterior predictive distribution:

$$\begin{aligned} p(t \mid \mathbf{x}, \mathcal{D}) &= \int p(t \mid x, \mathbf{w}) p(\mathbf{w} \mid \mathcal{D}) \, \mathrm{d}\mathbf{w} \\ &= \mathcal{N}(t \mid \boldsymbol{\mu}^\top \mathbf{x}, \sigma^2_{\mathrm{pred}}(x)) \\ \sigma^2_{\mathrm{pred}}(x) &= \sigma^2 + \mathbf{x}^\top \boldsymbol{\Sigma} \mathbf{x}, \end{aligned}$$

where $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ are the posterior mean and covariance of $\boldsymbol{\Sigma}$.

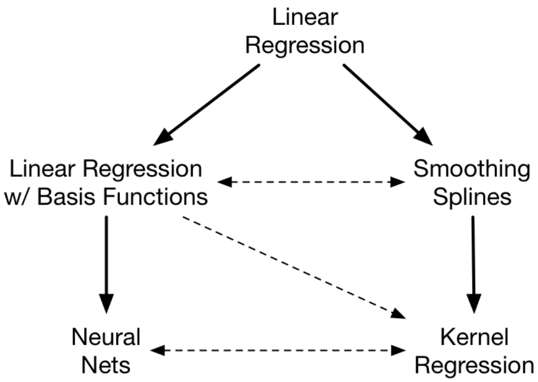
Bayesian Linear Regression

Posterior predictive distribution:

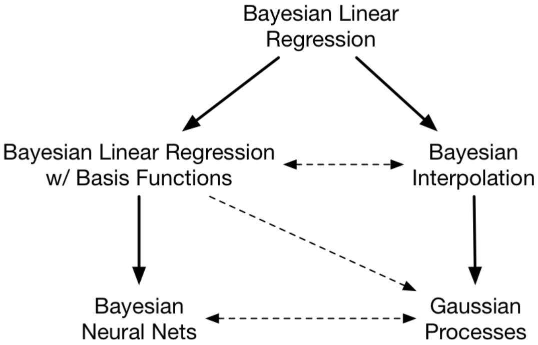


— Bishop, Pattern Recognition and Machine Learning

Foreshadowing

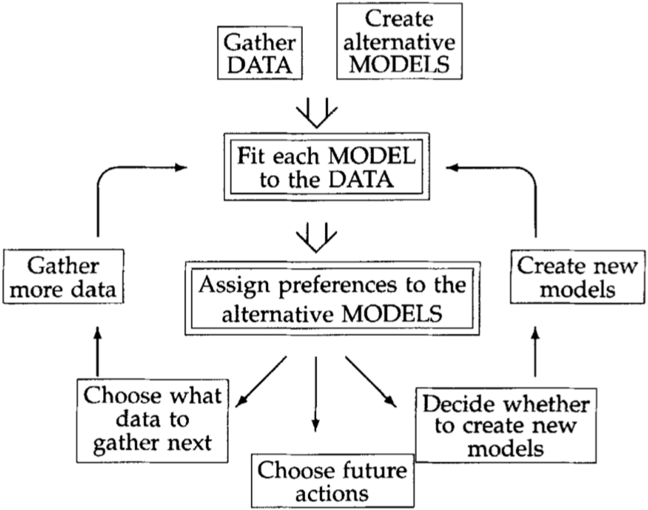


Foreshadowing



Occam's Razor

- Data modeling process according to MacKay:



Occam's Razor

- Occam's Razor: "Entities should not be multiplied beyond necessity."
  - Named after the 14th century British theologian William of Occam
- Huge number of attempts to formalize mathematically
  - See Domingos, 1999, "The role of Occam's Razor in knowledge discovery" for a skeptical overview.
    `https://homes.cs.washington.edu/~pedrod/papers/dmkd99.pdf`
- Common misinterpretation: your prior should favor simple explanations

Occam's Razor

- Suppose you have a finite set of models, or hypotheses $\{\mathcal{H}_i\}_{i=1}^{M}$ (e.g. polynomials of different degrees)
- Posterior inference over models (Bayes' Rule):

$$p(\mathcal{H}_i \,|\, \mathcal{D}) \propto \underbrace{p(\mathcal{H}_i)}_{\text{prior}} \underbrace{p(\mathcal{D} \,|\, \mathcal{H}_i)}_{\text{evidence}}$$

- Which of these terms do you think is more important?

Occam's Razor

- Suppose you have a finite set of models, or hypotheses $\{\mathcal{H}_i\}_{i=1}^M$ (e.g. polynomials of different degrees)
- Posterior inference over models (Bayes' Rule):

$$p(\mathcal{H}_i \,|\, \mathcal{D}) \propto \underbrace{p(\mathcal{H}_i)}_{\text{prior}} \underbrace{p(\mathcal{D} \,|\, \mathcal{H}_i)}_{\text{evidence}}$$

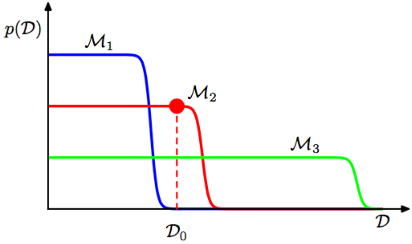
- Which of these terms do you think is more important?
- The evidence is also called marginal likelihood since it requires marginalizing out the parameters:

$$p(\mathcal{D} \,|\, \mathcal{H}_i) = \int p(\mathbf{w} \,|\, \mathcal{H}_i) \, p(\mathcal{D} \,|\, \mathbf{w}, \mathcal{H}_i) \, \mathrm{d}\mathbf{w}$$

- If we're comparing a handful of hypotheses, $p(\mathcal{H}_i)$ isn't very important, so we can compare them based on marginal likelihood.

Occam's Razor

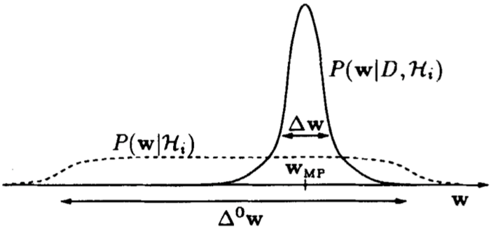
- Suppose M_1, M_2, and M_3 denote a linear, quadratic, and cubic model.
- M_3 is capable of explaning more datasets than M_1.
- But its distribution over $\mathcal{D}$ must integrate to 1, so it must assign lower probability to ones it can explain.



— Bishop, Pattern Recognition and Machine Learning

Occam's Razor

- How does the evidence (or marginal likelihood) penalize complex models?



- Approximating the integral:

$$p(\mathcal{D}\,|\,\mathcal{H}_i) = \int p(\mathcal{D}\,|\,\mathbf{w},\mathcal{H}_i)\,p(\mathbf{w}\,|\,\mathcal{H}_i)$$

$$\simeq \underbrace{p(\mathcal{D}\,|\,\mathbf{w}_{\mathrm{MAP}},\mathcal{H}_i)}_{\text{best-fit likelihood}}\;\underbrace{p(\mathbf{w}_{\mathrm{MAP}}\,|\,\mathcal{H}_i)\,\Delta\mathbf{w}}_{\text{Occam factor}}$$

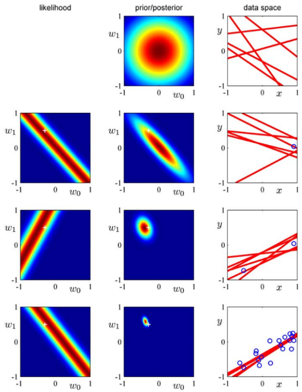
Occam's Razor

- Multivariate case:

$$p(\mathcal{D} \mid \mathcal{H}_i) \simeq \underbrace{p(\mathcal{D} \mid \mathbf{w}_{\mathrm{MAP}}, \mathcal{H}_i)}_{\text{best-fit likelihood}} \underbrace{p(\mathbf{w}_{\mathrm{MAP}} \mid \mathcal{H}_i) \, |\mathbf{A}|^{-1/2}}_{\text{Occam factor}},$$

where $\mathbf{A} = \nabla^2_{\mathbf{w}} \log p(\mathcal{D} \mid \mathbf{w}, \mathcal{H}_i)$

- The determinant appears because we're taking the volume.
- The more parameters in the model, the higher dimensional the parameter space, and the faster the volume decays.



— Bishop, Pattern Recognition and Machine Learning

Occam's Razor

- Analyzing the asymptotic behavior:

$$
\begin{aligned}
\mathbf{A} &= \nabla^2_{\mathbf{w}} \log p(\mathcal{D} \,|\, \mathbf{w}, \mathcal{H}_i) \\
&= \sum_{j=1}^{N} \underbrace{\nabla^2_{\mathbf{w}} \log p(y_i \,|\, \mathbf{x}_i, \mathbf{w}, \mathcal{H}_i)}_{\triangleq A_i} \\
&\approx N \, \mathbb{E}[A_i] \\
\text{log Occam factor} &= \log p(\mathbf{w}_{\mathrm{MAP}} \,|\, \mathcal{H}_i) + \log |\mathbf{A}|^{-1/2} \\
&\approx \log p(\mathbf{w}_{\mathrm{MAP}} \,|\, \mathcal{H}_i) + \log |N \, \mathbb{E}[A_i]|^{-1/2} \\
&= \log p(\mathbf{w}_{\mathrm{MAP}} \,|\, \mathcal{H}_i) - \frac{1}{2} \log |\mathbb{E}[A_i]| - \frac{D \log N}{2} \\
&= \text{const} - \frac{D \log N}{2}
\end{aligned}
$$

- Bayesian Information Criterion (BIC): penalize the complexity of your model by $\frac{1}{2} D \log N$.

Occam's Razor

- Summary

$$p(\mathcal{H}_i \,|\, \mathcal{D}) \propto p(\mathcal{H}_i)\, p(\mathcal{D} \,|\, \mathcal{H}_i)$$
$$p(\mathcal{D} \,|\, \mathcal{H}_i) \simeq p(\mathcal{D} \,|\, \mathbf{w}_{\mathrm{MAP}}, \mathcal{H}_i)\, p(\mathbf{w}_{\mathrm{MAP}} \,|\, \mathcal{H}_i)\, |\mathbf{A}|^{-1/2}$$

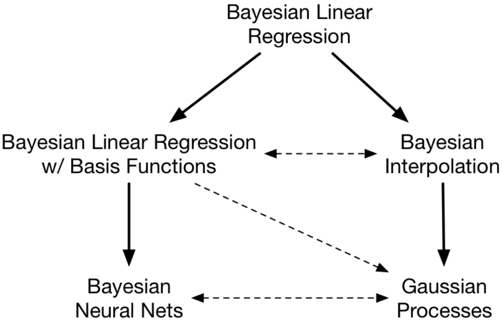
  Asymptotically, with lots of data, this behaves like

$$\log p(\mathcal{D} \,|\, \mathcal{H}_i) = \log p(\mathcal{D} \,|\, \mathbf{w}_{\mathrm{MAP}}, \mathcal{H}_i) - \frac{1}{2} D \log N.$$

- Occam's Razor is about integration, not priors (over hypotheses).

Bayesian Interpolation

- So all we need to do is count parameters? Not so fast!
- Let's consider the Bayesian analogue of smoothing splines, which MacKay refers to as Bayesian interpolation.



Bayesian Interpolation

- Recall the smoothing spline objective. How many parameters?

$$\mathcal{E}(f, \lambda) = \underbrace{\sum_{i=1}^{N} (t^{(i)} - f(x^{(i)}))^2}_{\text{mean squared error}} + \lambda \underbrace{\int (f''(z))^2 \, \mathrm{d}z}_{\text{regularizer}}$$

Bayesian Interpolation

- Recall the smoothing spline objective. How many parameters?

$$\mathcal{E}(f,\lambda) = \underbrace{\sum_{i=1}^{N} (t^{(i)} - f(x^{(i)}))^2}_{\text{mean squared error}} + \lambda \underbrace{\int (f''(z))^2 \, \mathrm{d}z}_{\text{regularizer}}$$

- Recall we can convert it to basis function regression with one basis function per training example.
  - So we have N parameters, and hence a log Occam factor $\approx \frac{1}{2} N \log N$?
  - You would never prefer this over a constant function!
  - Fortunately, this is not what happens.
- For computational convenience, we could choose some other set of basis functions (e.g. polynomials).

Bayesian Interpolation

- To define a Bayesian analogue of smoothing splines, let's convert it to a Bayesian basis function regression problem.
- The likelihood is easy:

$$p(\mathcal{D} \mid \mathbf{w}) = \prod_{i=1}^{N} \mathcal{N}(y_i \mid \mathbf{w}^\top \phi(x_i), \sigma^2)$$

- We'd like a prior which favors smoother functions:

$$\begin{aligned} p(\mathbf{w}) &\propto \exp\left(-\frac{\lambda}{2} \int (f''(z))^2 \, \mathrm{d}z\right) \\ &= \exp\left(-\frac{\lambda}{2} \mathbf{w}^\top \boldsymbol{\Omega} \mathbf{w}\right). \end{aligned}$$

Note: this is a zero-mean Gaussian.

Bayesian Interpolation

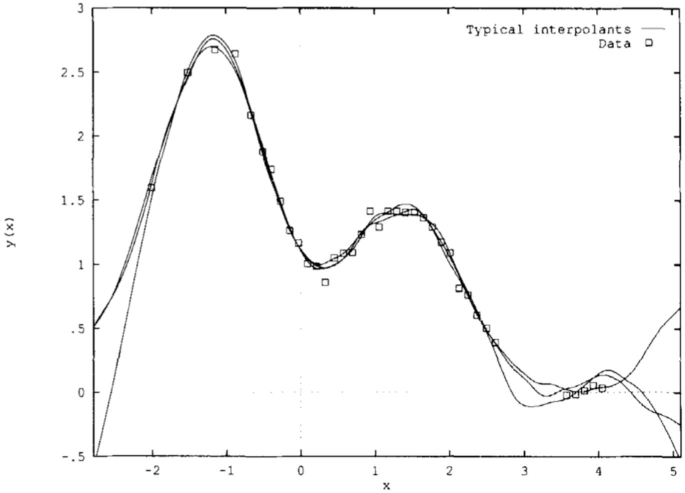
- Posterior distribution and posterior predictive distribution (special case of Bayesian linear regression)

$$\begin{aligned}
\mathbf{w} \,|\, \mathcal{D} &\sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \\
\boldsymbol{\mu} &= \sigma^{-2} \boldsymbol{\Sigma} \mathbf{X}^\top \mathbf{t} \\
\boldsymbol{\Sigma}^{-1} &= \lambda \boldsymbol{\Omega} + \sigma^{-2} \mathbf{X}^\top \mathbf{X} \\
p(t \,|\, \mathbf{x}, \mathcal{D}) &= \sigma^2 + \mathbf{x}^\top \boldsymbol{\Sigma} \mathbf{x}
\end{aligned}$$

- Optimize the hyperparameters σ and λ by maximizing the evidence (marginal likelihood).
    - This is known as the evidence approximation, or type 2 maximum likelihood.

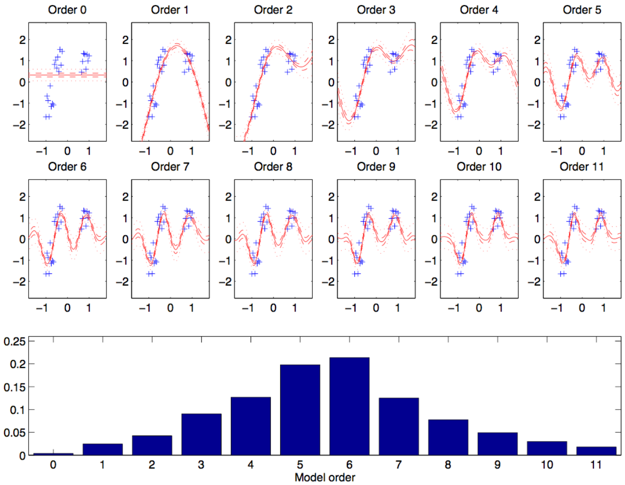
Bayesian Interpolation

- This makes reasonable predictions:



Bayesian Interpolation

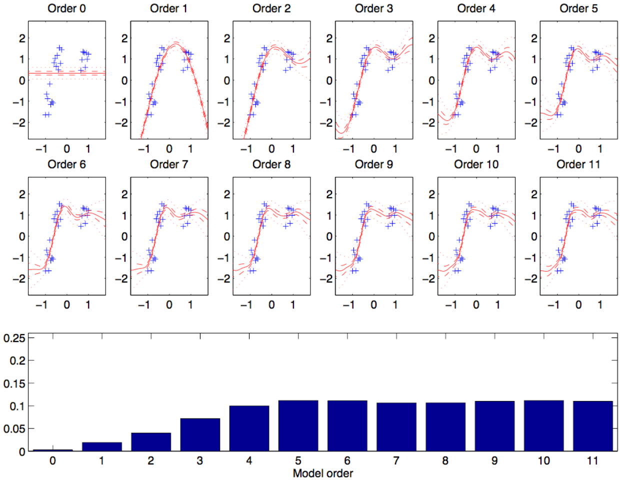
Behavior w/ spherical prior as we add more basis functions:



— Rasmussen and Ghahramani, "Occam's Razor"

Bayesian Interpolation

Behavior w/ smoothness prior as we add more basis functions:



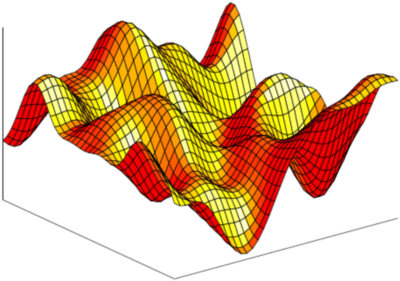
— Rasmussen and Ghahramani, "Occam's Razor"

Towards Gaussian Processes

- Splines stop getting more complex as you add more basis functions.
- Bayesian Occam's Razor penalizes the complexity of the distribution over functions, not the number of parameters.
- Maybe we can fit infinitely many parameters!
- Rasmussen and Ghahramani (2001): in the infinite limit, the distribution over functions approaches a Gaussian process.

Towards Gaussian Processes

- Gaussian Processes are distributions over functions.
- They're actually a simpler and more intuitive way to think about regression, once you're used to them.



— GPML

Towards Gaussian Processes

- A Bayesian linear regression model defines a distribution over functions:
$$f(\mathbf{x}) = \mathbf{w}^\top \phi(\mathbf{x})$$
Here, $\mathbf{w}$ is sampled from the prior $\mathcal{N}(\boldsymbol{\mu}_\mathbf{w}, \boldsymbol{\Sigma}_\mathbf{w})$.
- Let $\mathbf{f} = (f_1, \ldots, f_N)$ denote the vector of function values at $(\mathbf{x}_1, \ldots, \mathbf{x}_N)$.
- The distribution of $\mathbf{f}$ is a Gaussian with
$$\mathbb{E}[f_i] = \boldsymbol{\mu}_\mathbf{w}^\top \phi(\mathbf{x})$$
$$\mathrm{Cov}(f_i, f_j) = \phi(\mathbf{x}_i)^\top \boldsymbol{\Sigma}_\mathbf{w} \phi(\mathbf{x}_j)$$
- In vectorized form, $\mathbf{f} \sim \mathcal{N}(\boldsymbol{\mu}_\mathbf{f}, \boldsymbol{\Sigma}_\mathbf{f})$ with
$$\boldsymbol{\mu}_\mathbf{f} = \mathbb{E}[\mathbf{f}] = \boldsymbol{\Phi}\boldsymbol{\mu}_\mathbf{w}$$
$$\boldsymbol{\Sigma}_\mathbf{f} = \mathrm{Cov}(\mathbf{f}) = \boldsymbol{\Phi}\boldsymbol{\Sigma}_\mathbf{w}\boldsymbol{\Phi}^\top$$

Towards Gaussian Processes

- Recall that in Bayesian linear regression, we assume noisy Gaussian observations of the underlying function.

$$y_i \sim \mathcal{N}(f_i, \sigma^2) = \mathcal{N}(\mathbf{w}^\top \phi(\mathbf{x}_i), \sigma^2).$$

- The observations $\mathbf{y}$ are jointly Gaussian, just like $\mathbf{f}$.

$$\mathbb{E}[y_i] = \mathbb{E}[f(\mathbf{x}_i)]$$
$$\mathrm{Cov}(y_i, y_j) = \begin{cases} \mathrm{Var}(f(\mathbf{x}_i)) + \sigma^2 & \text{if } i = j \\ \mathrm{Cov}(f(\mathbf{x}_i), f(\mathbf{x}_j)) & \text{if } i \neq j \end{cases}$$

- In vectorized form, $\mathbf{y} \sim \mathcal{N}(\boldsymbol{\mu}_\mathbf{y}, \boldsymbol{\Sigma}_\mathbf{y})$, with

$$\boldsymbol{\mu}_\mathbf{y} = \boldsymbol{\mu}_\mathbf{f}$$
$$\boldsymbol{\Sigma}_\mathbf{y} = \boldsymbol{\Sigma}_\mathbf{f} + \sigma^2 \mathbf{I}$$

Towards Gaussian Processes

- Bayesian linear regression is just computing the conditional distribution in a multivariate Gaussian!
- Let $\mathbf{y}$ and $\mathbf{y}'$ denote the observables at the training and test data.
- They are jointly Gaussian:

$$\begin{pmatrix} \mathbf{y} \\ \mathbf{y}' \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} \boldsymbol{\mu}_{\mathbf{y}} \\ \boldsymbol{\mu}_{\mathbf{y}'} \end{pmatrix}, \begin{pmatrix} \boldsymbol{\Sigma}_{\mathbf{y}\mathbf{y}} & \boldsymbol{\Sigma}_{\mathbf{y}\mathbf{y}'} \\ \boldsymbol{\Sigma}_{\mathbf{y}'\mathbf{y}} & \boldsymbol{\Sigma}_{\mathbf{y}'\mathbf{y}'} \end{pmatrix} \right).$$

- The predictive distribution is a special case of the conditioning formula for a multivariate Gaussian:

$$\begin{aligned} \mathbf{y}' \mid \mathbf{y} &\sim \mathcal{N}(\boldsymbol{\mu}_{\mathbf{y}'|\mathbf{y}}, \boldsymbol{\Sigma}_{\mathbf{y}'|\mathbf{y}}) \\ \boldsymbol{\mu}_{\mathbf{y}'|\mathbf{y}} &= \boldsymbol{\mu}_{\mathbf{y}'} + \boldsymbol{\Sigma}_{\mathbf{y}'\mathbf{y}} \boldsymbol{\Sigma}_{\mathbf{y}\mathbf{y}}^{-1} (\mathbf{y} - \boldsymbol{\mu}_{\mathbf{y}}) \\ \boldsymbol{\Sigma}_{\mathbf{y}'|\mathbf{y}} &= \boldsymbol{\Sigma}_{\mathbf{y}'\mathbf{y}'} - \boldsymbol{\Sigma}_{\mathbf{y}'\mathbf{y}} \boldsymbol{\Sigma}_{\mathbf{y}\mathbf{y}}^{-1} \boldsymbol{\Sigma}_{\mathbf{y}\mathbf{y}'} \end{aligned}$$

- We're implicitly marginalizing out $\mathbf{w}$!

Towards Gaussian Processes

- The marginal likelihood is just the PDF of a multivariate Gaussian:

$$
\begin{aligned}
p(\mathbf{y} \mid \mathbf{X}) &= \mathcal{N}(\mathbf{y}; \boldsymbol{\mu}_{\mathbf{y}}, \boldsymbol{\Sigma}_{\mathbf{y}}) \\
&= \frac{1}{(2\pi)^{d/2} |\boldsymbol{\Sigma}_{\mathbf{y}}|^{1/2}} \exp\left(-\frac{1}{2} (\mathbf{y} - \boldsymbol{\mu}_{\mathbf{y}})^\top \boldsymbol{\Sigma}_{\mathbf{y}}^{-1} (\mathbf{y} - \boldsymbol{\mu}_{\mathbf{y}}) \right)
\end{aligned}
$$

Towards Gaussian Processes

- To summarize:

$$\begin{aligned}
\boldsymbol{\mu}_{\mathbf{f}} &= \boldsymbol{\Phi}\boldsymbol{\mu}_{\mathbf{w}} \\
\boldsymbol{\Sigma}_{\mathbf{f}} &= \boldsymbol{\Phi}\boldsymbol{\Sigma}_{\mathbf{w}}\boldsymbol{\Phi}^{\top} \\
\boldsymbol{\mu}_{\mathbf{y}} &= \boldsymbol{\mu}_{\mathbf{f}} \\
\boldsymbol{\Sigma}_{\mathbf{y}} &= \boldsymbol{\Sigma}_{\mathbf{f}} + \sigma^2\mathbf{I} \\
\boldsymbol{\mu}_{\mathbf{y}'|\mathbf{y}} &= \boldsymbol{\mu}_{\mathbf{y}'} + \boldsymbol{\Sigma}_{\mathbf{y}'\mathbf{y}}\boldsymbol{\Sigma}_{\mathbf{y}\mathbf{y}}^{-1}(\mathbf{y} - \boldsymbol{\mu}_{\mathbf{y}}) \\
\boldsymbol{\Sigma}_{\mathbf{y}'|\mathbf{y}} &= \boldsymbol{\Sigma}_{\mathbf{y}'\mathbf{y}'} - \boldsymbol{\Sigma}_{\mathbf{y}'\mathbf{y}}\boldsymbol{\Sigma}_{\mathbf{y}\mathbf{y}}^{-1}\boldsymbol{\Sigma}_{\mathbf{y}\mathbf{y}'} \\
p(\mathbf{y} \,|\, \mathbf{X}) &= \mathcal{N}(\mathbf{y}; \boldsymbol{\mu}_{\mathbf{y}}, \boldsymbol{\Sigma}_{\mathbf{y}})
\end{aligned}$$

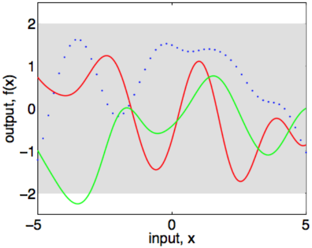
- After defining $\boldsymbol{\mu}_{\mathbf{f}}$ and $\boldsymbol{\Sigma}_{\mathbf{f}}$, we can forget about **w** and **x**!
- What if we just let $\boldsymbol{\mu}_{\mathbf{f}}$ and $\boldsymbol{\Sigma}_{\mathbf{f}}$ be anything?

Gaussian Processes

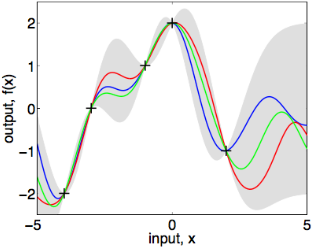
- When I say let $\boldsymbol{\mu}_\mathbf{f}$ and $\boldsymbol{\Sigma}_\mathbf{f}$ be anything, I mean let them have an arbitrary functional dependence on the inputs.
- We need to specify
  - a mean function $\mathbb{E}[f(\mathbf{x}_i)] = \mu(\mathbf{x}_i)$
  - a covariance function called a kernel function: $\mathrm{Cov}(f(\mathbf{x}_i), f(\mathbf{x}_j)) = k(\mathbf{x}_i, \mathbf{x}_j)$
- Let $\mathbf{K}_\mathbf{X}$ denote the kernel matrix for points $\mathbf{X}$. This is a matrix whose (i, j) entry is $k(\mathbf{x}_i, \mathbf{x}_j)$.
- We require that $\mathbf{K}_\mathbf{X}$ be positive semidefinite for *any* $\mathbf{X}$. Other than that, μ and k can be arbitrary.

Gaussian Processes

- We've just defined a distribution over *function values* at an arbitrary finite set of points.
- This can be extended to a distribution over *functions* using a kind of black magic called the Kolmogorov Extension Theorem. This distribution over functions is called a Gaussian process (GP).
- We only ever need to compute with distributions over function values. The formulas from a few slides ago are all you need to do regression with GPs.
- But distributions over functions are conceptually cleaner.



(a), prior



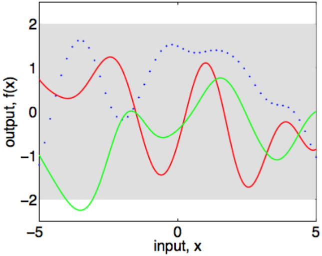
(b), posterior

GP Kernels

- One way to define a kernel function is to give a set of basis functions and put a Gaussian prior on $\mathbf{w}$.
- But we have lots of other options. Here's a useful one, called the squared-exp, or Gaussian, or radial basis function (RBF) kernel:

$$k_{\mathrm{SE}}(\mathbf{x}_i, \mathbf{x}_j) = \sigma^2 \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\ell^2}\right)$$

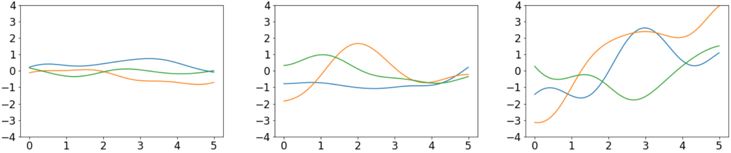
- More accurately, this is a kernel family with hyperparameters σ and ℓ.
- It gives a distribution over smooth functions:



GP Kernels

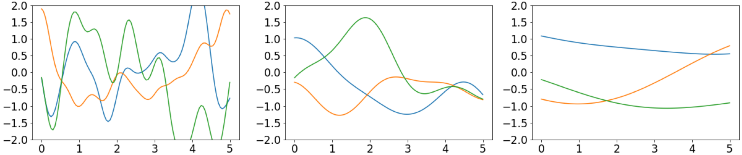
$$k_{\mathrm{SE}}(x_i, x_j) = \sigma^2 \exp\left(-\frac{(x_i - x_j)^2}{2\ell^2}\right)$$

- The hyperparameters determine key properties of the function.
- Varying the output variance σ^2:



$\sigma^2 = 0.3$ $\sigma^2 = 1$ $\sigma^2 = 3$

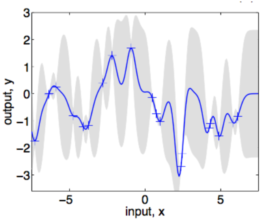
- Varying the lengthscale ℓ:



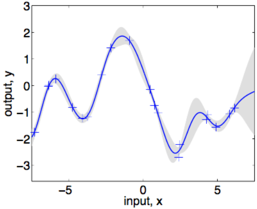
$\ell = 0.3$ $\ell = 1$ $\ell = 3$

GP Kernels

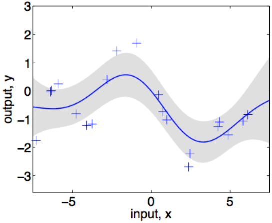
- The choice of hyperparameters heavily influences the predictions:



(b), $\ell = 0.3$



(a), $\ell = 1$



(c), $\ell = 3$

- In practice, it's very important to tune the hyperparameters (e.g. by maximizing the marginal likelihood).

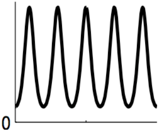
GP Kernels

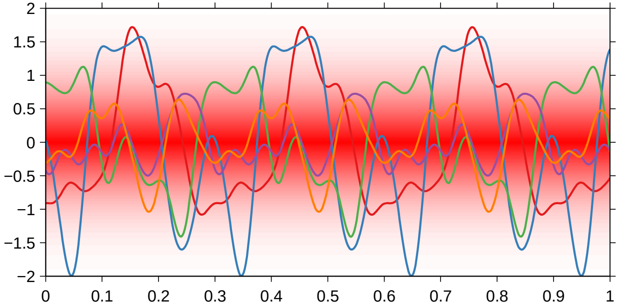
$$k_{\mathrm{SE}}(x_i, x_j) = \sigma^2 \exp\left(-\frac{(x_i - x_j)^2}{2\ell^2}\right)$$

- The squared-exp kernel is stationary because it only depends on $x_i - x_j$. Most kernels we use in practice are stationary.
- We can visualize the function $k(0, x)$:

GP Kernels

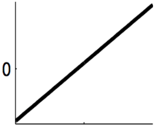
- The periodic kernel encodes for a probability distribution over periodic functions

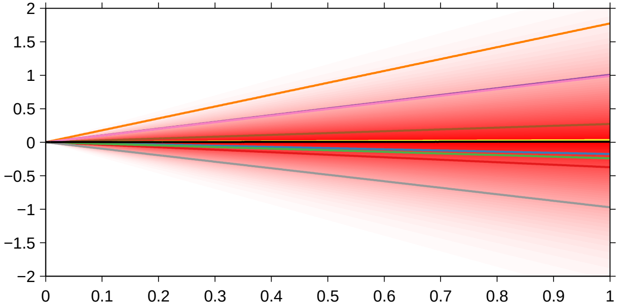




GP Kernels

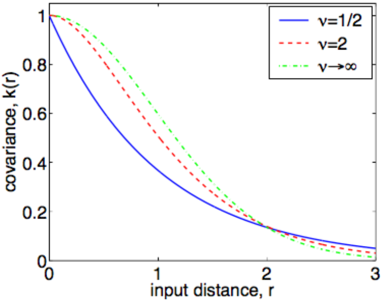
- The linear kernel results in a probability distribution over linear functions



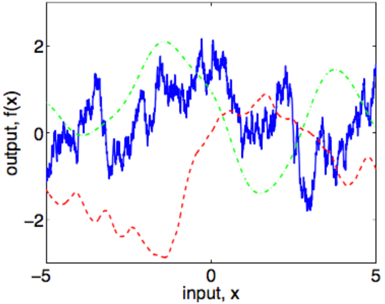


GP Kernels

- The Matern kernel is similar to the squared-exp kernel, but less smooth.
- See Chapter 4 of GPML for an explanation (advanced).
- Imagine trying to get this behavior by designing basis functions!



(a)



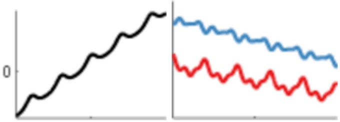
(b)

GP Kernels

- We get exponentially more flexibility by combining kernels.
- The sum of two kernels is a kernel.
  - This is because valid covariance matrices (i.e. PSD matrices) are closed under addition.
- The sum of two kernels corresponds to the sum of functions.

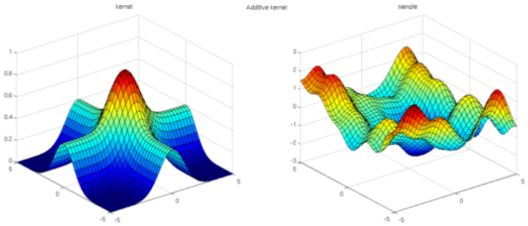
Linear + Periodic

e.g. seasonal pattern w/ trend



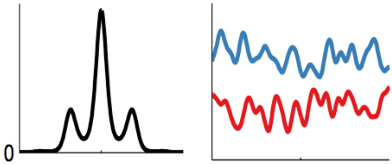
Additive kernel

$$k(x, y, x', y') = k_1(x, x') + k_2(y, y')$$



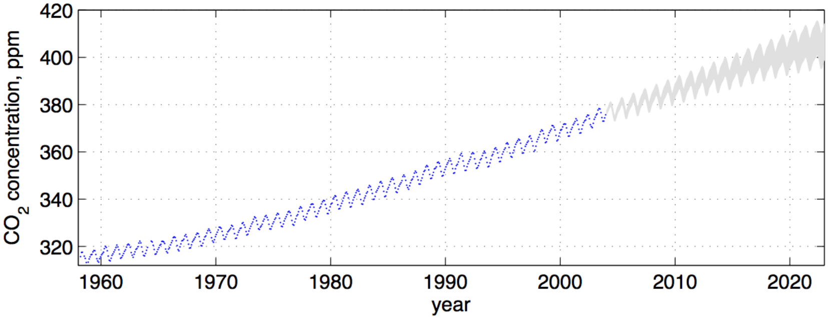
GP Kernels

- A kernel is like a similarity function on the input space. The sum of two kernels is like the OR of their similarity.
- Amazingly, the product of two kernels is a kernel. (Follows from the Schur Product Theorem.)
- The product of two kernels is like the AND of their similarity functions.
- Example: the product of a squared-exp kernel (spatial similarity) and a periodic kernel (similar location within cycle) gives a locally periodic function.



GP Kernels

- Modeling CO2 concentrations:
  trend + (changing) seasonal pattern + short-term variability + noise
- Encoding the structure allows sensible extrapolation.



Summary

- Bayesian linear regression lets us determine uncertainty in our predictions.
- We can make it nonlinear by using fixed basis functions.
- Bayesian Occam's Razor is a sophisticated way of penalizing the complexity of a distribution over functions.
- Gaussian processes are an elegant framework for doing Bayesian inference directly over functions.
- The choice of kernels gives us much more control over what sort of functions our prior would allow or favor.
- Next time: Bayesian neural nets, a different way of making Bayesian linear regression more powerful.