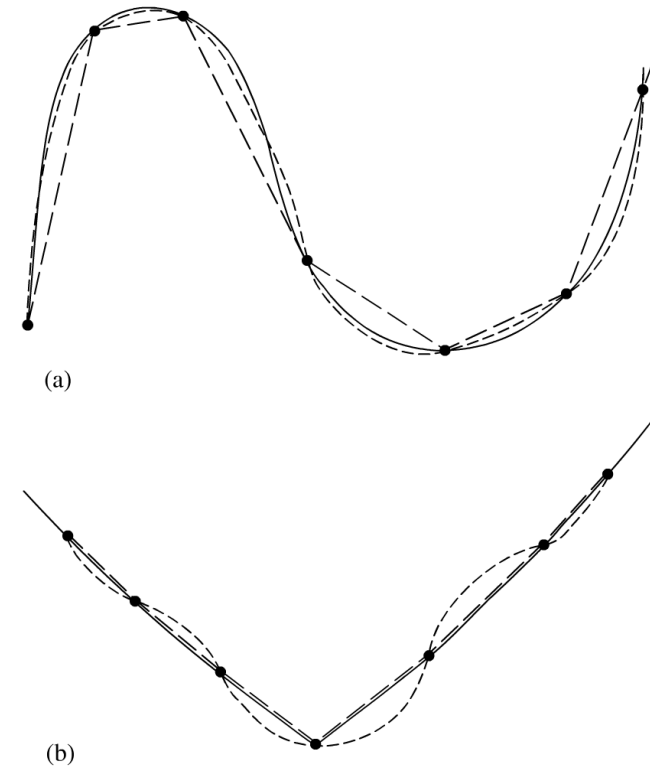


LECTURE 11: From Interpolations to Regressions to Gaussian Processes

- So far we were mostly doing linear or nonlinear regression of data points with simple small basis (for example, linear function $y = ax + b$)
- The basis can be arbitrarily large and can be defined in many different ways: we do not care about values of $a, b \dots$ but about predictions for y
- This is the task of machine learning
- Interpolation can be viewed as regression without noise (but still sparse sampling of data)
- There are many different regressions

Interpolation

- Goal: we have function $y = f(x)$ defined at points $y_i = f(x_i)$, and we want to know its value at any x . For now we will assume solution goes through x_i 's
- Why? Perhaps it is very expensive to evaluate it everywhere. Or perhaps we really do not know it.
- Local interpolation: use a few nearby points
- Example: polynomial interpolation
 $f(x) = \sum_{i=1}^N a_i x^i$
- How do we choose N ? Higher N better for smooth functions and worse for sharp kinks



Differentiability

- Lagrange formula for polynomial interpolation:

$$\begin{aligned} P(x) = & \frac{(x - x_1)(x - x_2)\dots(x - x_{M-1})}{(x_0 - x_1)(x_0 - x_2)\dots(x_0 - x_{M-1})} y_0 \\ & + \frac{(x - x_0)(x - x_2)\dots(x - x_{M-1})}{(x_1 - x_0)(x_1 - x_2)\dots(x_1 - x_{M-1})} y_1 + \dots \\ & + \frac{(x - x_0)(x - x_1)\dots(x - x_{M-2})}{(x_{M-1} - x_0)(x_{M-1} - x_1)\dots(x_{M-1} - x_{M-2})} y_{M-1} \end{aligned}$$

- Polynomial interpolation does not guarantee that derivatives are differentiable everywhere.
- Stiffer solution are splines, where we enforce derivatives to be continuous
- Most popular is cubic spline where 1st derivatives are smooth and 2nd derivatives are continuous

Cubic Spline

- Start with piecewise linear interpolation

$$y = Ay_j + By_{j+1} \quad A \equiv \frac{x_{j+1} - x}{x_{j+1} - x_j} \quad B \equiv 1 - A = \frac{x - x_j}{x_{j+1} - x_j}$$

- This will not have 2nd derivative continuous: it is zero inside intervals and infinite at x_i
- But we can add another interpolation of 2nd derivatives y'' . If we also arrange it to be 0 at x_i then we do not spoil linear interpolation above.
- This has a unique solution:

$$y = Ay_j + By_{j+1} + Cy_j'' + Dy_{j+1}''$$

$$C \equiv \frac{1}{6}(A^3 - A)(x_{j+1} - x_j)^2 \quad D \equiv \frac{1}{6}(B^3 - B)(x_{j+1} - x_j)^2$$

Cubic Spline

- So far we assumed we know y'' , but we do not
- We can determine it by requiring continuity of 1st derivatives on both sides of x_i :

$$\frac{dy}{dx} = \frac{y_{j+1} - y_j}{x_{j+1} - x_j} - \frac{3A^2 - 1}{6}(x_{j+1} - x_j)y_j'' + \frac{3B^2 - 1}{6}(x_{j+1} - x_j)y_{j+1}''$$

$$\frac{x_j - x_{j-1}}{6}y_{j-1}'' + \frac{x_{j+1} - x_{j-1}}{3}y_j'' + \frac{x_{j+1} - x_j}{6}y_{j+1}'' = \frac{y_{j+1} - y_j}{x_{j+1} - x_j} - \frac{y_j - y_{j-1}}{x_j - x_{j-1}}$$

- We get $N-2$ equations for N unknown y_i'' , tridiagonal system, $O(N)$
- Natural cubic spline: set $y_0'' = 0$ and $y_N'' = 0$

Rational Function Expansion

- Spline is mostly used for interpolation
- Polynomials can be used for extrapolation outside the interval (x_0, x_N) , but the polynomial with the largest power dominates and results in divergence
- Rational functions can be better for extrapolation: if we know the function goes to 0 we choose $n > m$

$$R_{i(i+1)\dots(i+m)} = \frac{P_\mu(x)}{Q_\nu(x)} = \frac{p_0 + p_1x + \dots + p_\mu x^\mu}{q_0 + q_1x + \dots + q_\nu x^\nu}$$

- Rational functions are better for interpolation if the functions has poles (in real or complex plane)
- Rational functions can also be used for analytic work (Pade approximation)

Interpolation on a grid in higher dimension

- Simplest 2d example: bilinear interpolation

$$t \equiv (x_1 - x_{1i}) / (x_{1(i+1)} - x_{1i})$$
$$u \equiv (x_2 - x_{2j}) / (x_{2(j+1)} - x_{2j})$$

(so that t and u each lie between 0 and 1) and

$$y(x_1, x_2) = (1 - t)(1 - u)y_0 + t(1 - u)y_1 + t uy_2 + (1 - t)uy_3$$

- Higher order accuracy: polynomials (biquadratic...)
- Higher order smoothness: bicubic spline

From Spline to B-spline to Gaussian

- We can write basis functions for (cubic) splines (B for basis), so that the solution is a linear combination of them. For cubic B-splines on uniform points

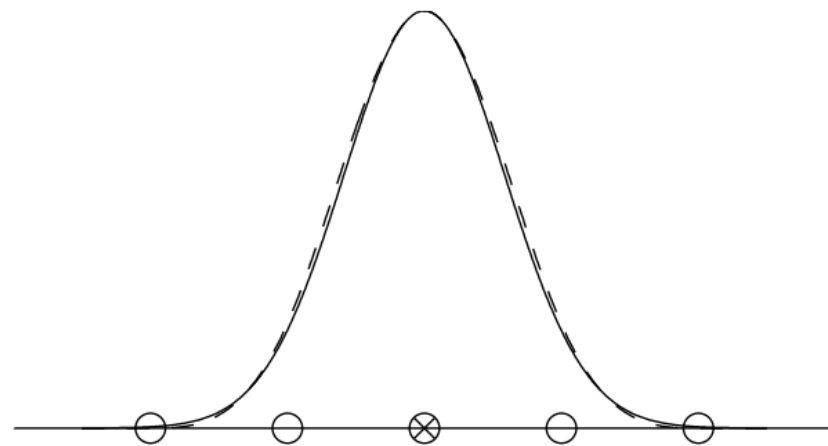
$$\mu(x) = \sum_{h=1}^H \beta_h b_h(x)$$

$$b_h(x) = \begin{cases} \frac{1}{6}u^3 & \text{for } x \in (x_h, x_{h+1}), \quad u = (x - x_h)/\delta \\ \frac{1}{6}(1 + 3u + 3u^2 - 3u^3) & \text{for } x \in (x_{h+1}, x_{h+2}), \quad u = (x - x_{h+1})/\delta \\ \frac{1}{6}(4 - 6u^2 + 3u^3) & \text{for } x \in (x_{h+2}, x_{h+3}), \quad u = (x - x_{h+2})/\delta \\ \frac{1}{6}(1 - 3u + 3u^2 - u^3) & \text{for } x \in (x_{h+3}, x_{h+4}), \quad u = (x - x_{h+3})/\delta \\ 0 & \text{otherwise.} \end{cases}$$

- Very close to gaussian

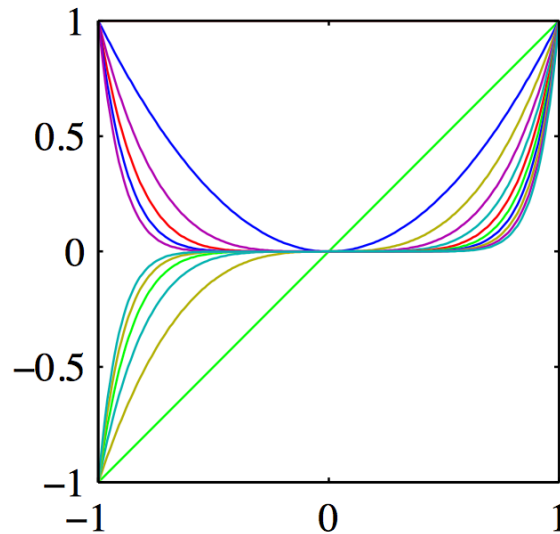
$$b_h(x) = \exp\left(-\frac{|x - x_h|^2}{l^2}\right)$$

- Gaussian is infinitely differentiable (i.e. smoother) but not sparse

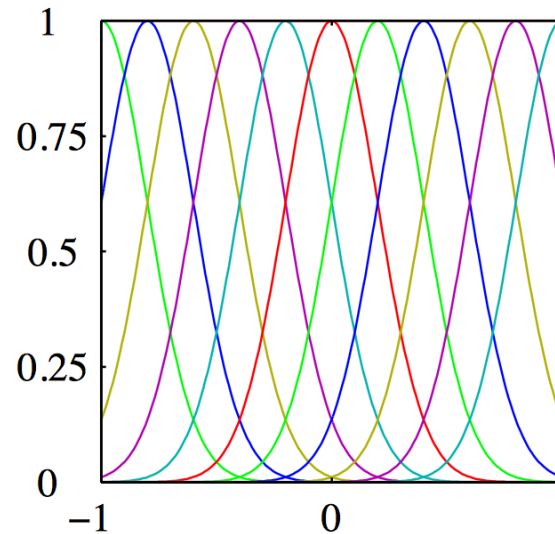


Example of Basis Functions

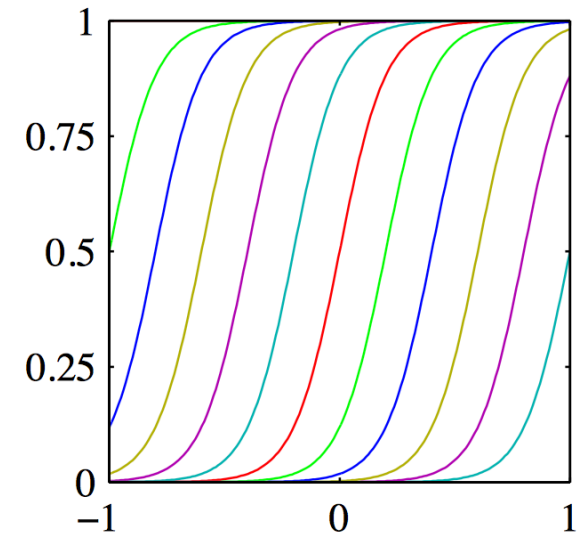
- Polynomial



- Spline/Gaussian



- Sigmoid



- Sigmoid (used for classification)

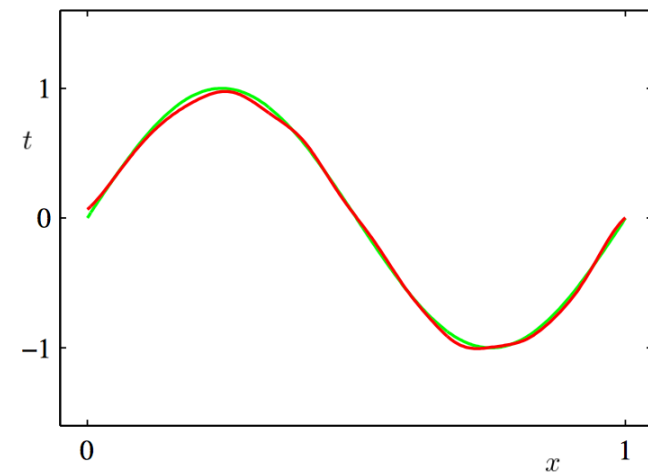
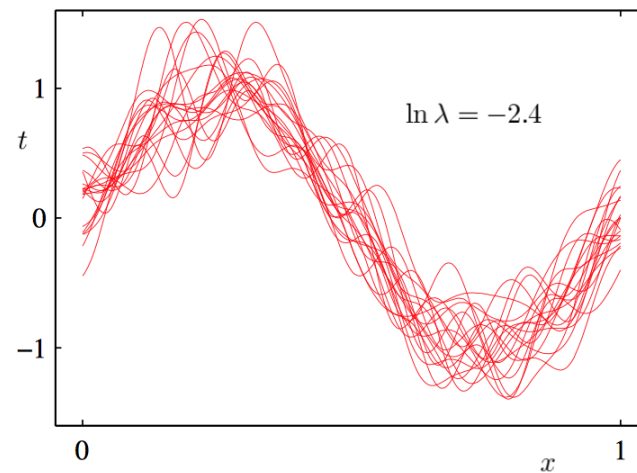
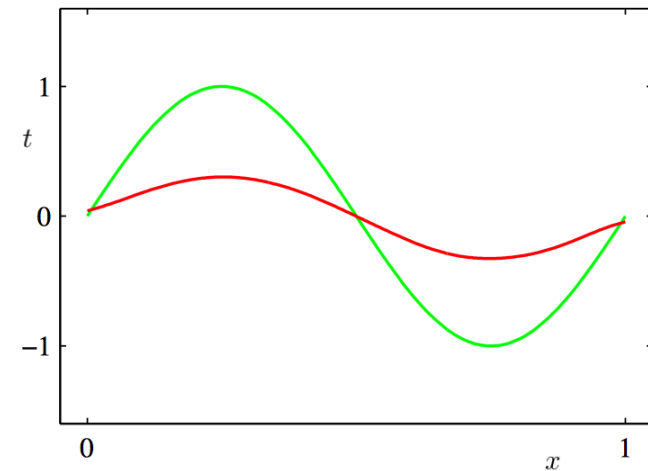
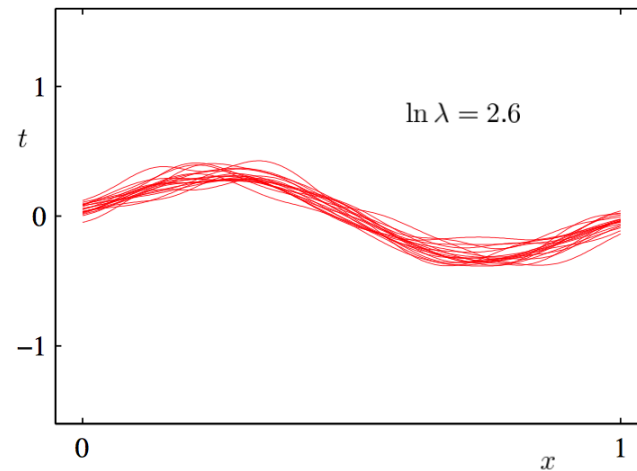
$$\phi_j(x) = \sigma\left(\frac{x - \mu_j}{s}\right) \text{ with } \sigma(a) = \frac{1}{1 + e^{-a}}$$

From Interpolation to Regression

- One can view function interpolation as regression in the limit of zero noise, but still sparse sampling
- Sparse sampling will induce an error, as will noise
- Both can use the same basis expansion $\phi(x)$
- For regression with noise or sparse sampling we add a regularization term (Tikhonov/ridge/L2, Lasso/L1...), this can prevent overfitting
- $l = \sum_{i=1}^N [\sum_{j=1}^M a_j \phi(x_i) - y_i]^2 + \lambda \sum_{j=1}^M a_j^2$
- The question is how to choose the regularization parameter λ

High λ : low scatter, high bias,
Low λ : high scatter, low bias

Example:
 $N = M = 25$,
Gaussian
basis



Bayesian Regression

$$t = \underbrace{y(\mathbf{x}, \mathbf{w})}_{\text{deterministic}} + \underbrace{\epsilon}_{\text{Gaussian noise}}$$

- In the Bayesian context we perform regression of coefficients a_j assigning them some prior distribution, such as a gaussian with some precision α . If we also have noise precision β then $\ln p$ is

$$\ln p(\mathbf{w}|\mathbf{t}) = -\frac{\beta}{2} \sum_{n=1}^N (t_n - \mathbf{w}^T \phi(\mathbf{x}_n))^2 - \frac{\alpha}{2} \mathbf{w}^T \mathbf{w} + \text{const}$$

- So regularizing parameter is $\lambda = \alpha/\beta$

Linear Algebra Solution

- More general prior $p(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mathbf{m}_0, \mathbf{S}_0)$

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

- Posterior
$$\left| \begin{array}{lcl} \mathbf{m}_N & = & \mathbf{S}_N(\mathbf{S}_0^{-1}\mathbf{m}_0 + \beta\Phi^T\mathbf{t}) \\ \mathbf{S}_N^{-1} & = & \mathbf{S}_0^{-1} + \beta\Phi^T\Phi \end{array} \right.$$

- We want to predict at arbitrary t :

$$p(t|\mathbf{t}, \alpha, \beta) = \int p(t|\mathbf{w}, \beta)p(\mathbf{w}|\mathbf{t}, \alpha, \beta)d\mathbf{w}$$

Prediction for t

$$p(t|\mathbf{t}, \alpha, \beta) = \int p(t|\mathbf{w}, \beta) p(\mathbf{w}|\mathbf{t}, \alpha, \beta) d\mathbf{w}$$

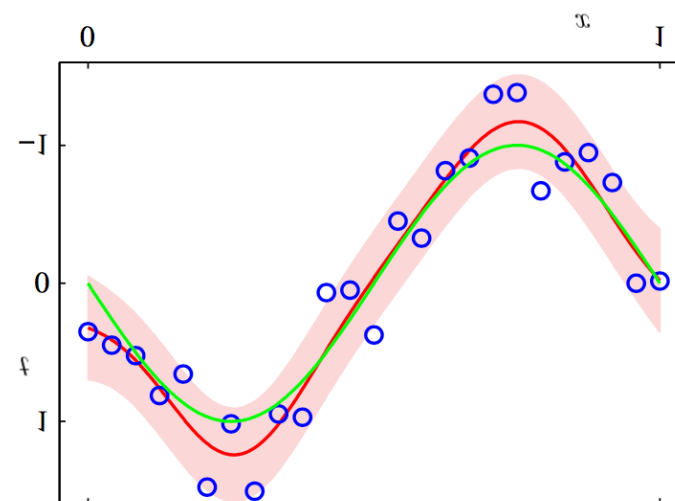
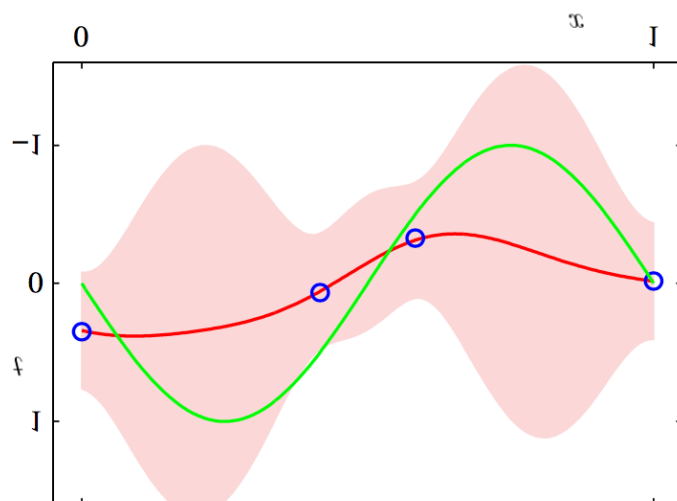
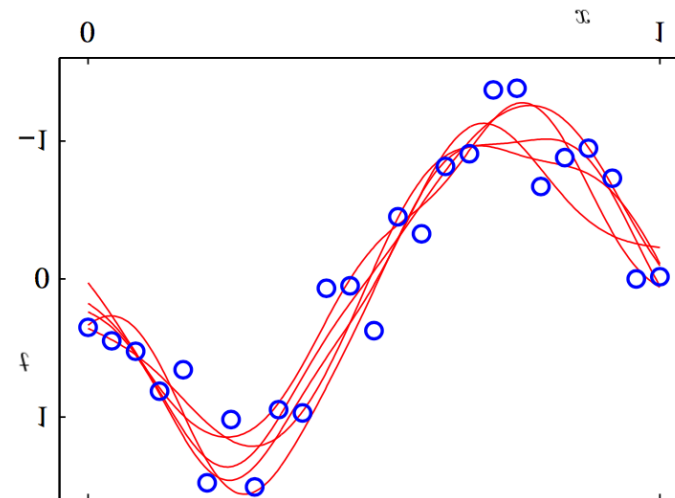
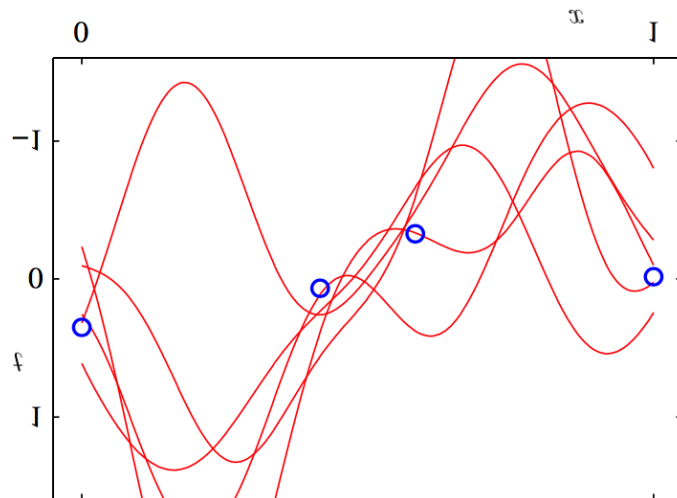
$$p(t|\mathbf{w}, \beta) = \mathcal{N}(t|y(\mathbf{x}, \mathbf{w}), \beta^{-1})$$

$$p(\mathbf{w}|\mathbf{t}, \alpha, \beta) = \mathcal{N}(\mathbf{w}|\mathbf{m}_N, \mathbf{S}_N)$$

$$p(t|\mathbf{x}, \mathbf{t}, \alpha, \beta) = \mathcal{N}(t|\mathbf{m}_N^T \Phi(\mathbf{x}), \sigma_N^2(\mathbf{x}))$$

$$\sigma_N^2(\mathbf{x}) = \underbrace{\beta^{-1}}_{\text{noise in data}} + \underbrace{\Phi(\mathbf{x})^T \mathbf{S}_N \Phi(\mathbf{x})}_{\text{uncertainty in } \mathbf{w}}$$

Examples



Kernel Picture

- Kernels work as closeness measure, giving more weight to nearby points. Assuming $m_0 = 0$

$y(\mathbf{x}, \mathbf{m}_N)$ rewrites as $\sum_{n=1}^N k(\mathbf{x}, \mathbf{x}_n) \mathbf{t}_n$ where

$$k(\mathbf{x}, \mathbf{x}') = \beta \Phi(\mathbf{x})^T \mathbf{S}_N \Phi(\mathbf{x}')$$

Basis functions \longleftrightarrow kernel duality

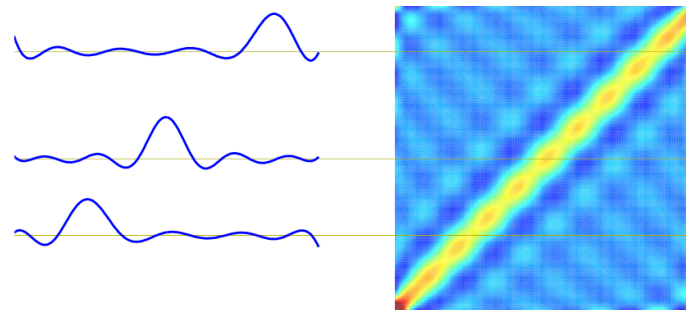
With $\Psi(\mathbf{x}) = \beta^{-1/2} \mathbf{S}_N^{1/2} \Phi(\mathbf{x})$, $k(\mathbf{x}, \mathbf{x}') = \Psi(\mathbf{x})^T \Psi(\mathbf{x}')$

The kernel sums to one (over the training set)

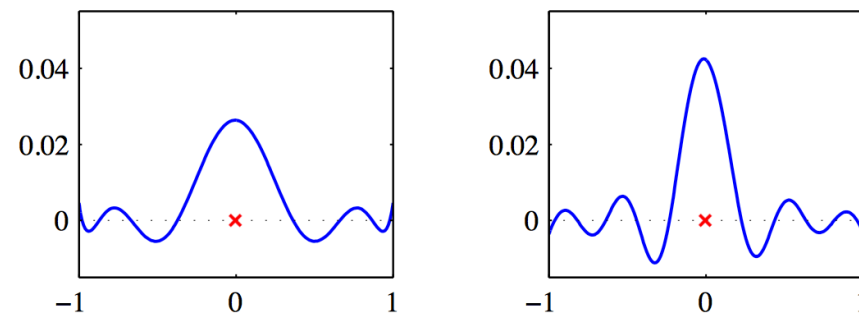
$$\text{cov}(y(\mathbf{x}), y(\mathbf{x}')) = \beta^{-1} k(\mathbf{x}, \mathbf{x}')$$

Kernel Examples

Kernel from Gaussian basis functions



Kernels at $\mathbf{x} = 0$ for kernels corresponding (left) to the polynomial basis functions and (right) to the sigmoidal basis functions.



Kernel Regression

- So far the kernel was defined in terms of a finite number of basis functions $\Phi(x)$
- We can eliminate the concept of the basis functions and work simply with kernels
- Nadaraya-Watson regression
$$\widehat{m}_h(x) = \frac{\sum_{i=1}^n K_h(x - x_i) y_i}{\sum_{i=1}^n K_h(x - x_i)}$$
- We want to estimate $y(x) = h(x)$: we use points close to x weighted by some function of the distance $K_h(x - x_i)$: as the distance increases the weights drop off
- Kernel K_h does not have to be a covariance function, but if it is then this becomes a gaussian process

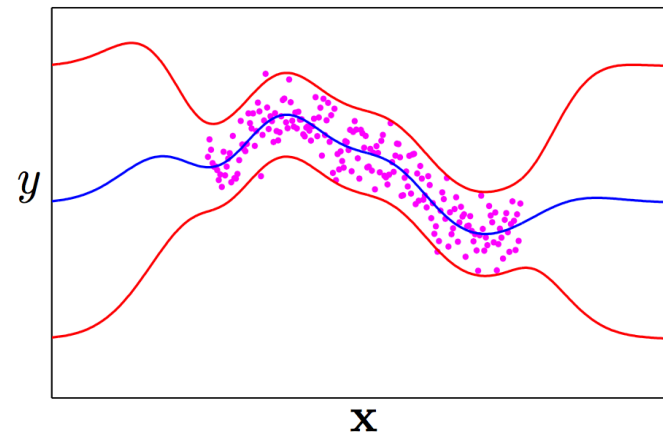
Gaussian Process

- We interpret the kernel as $K_{ij} = \text{Cov}(y(x_i), y(x_j))$ and define $y(x)$ as a random variable with a multi-variate gaussian distribution $N(\mu, K)$

- 2-variables

$$\mu = \begin{bmatrix} \mu(x) \\ \mu(x') \end{bmatrix} \quad \Sigma = \begin{bmatrix} K(x, x) & K(x, x') \\ K(x', x) & K(x', x') \end{bmatrix}$$

- We then use posterior prediction for y : we get both the mean m and variance $K^{1/2}$
- We can interpret GP as regression on an infinite basis



Gaussian Process

- Example covariance function K for translational and rotational invariant case

$$K(x_i, x_j) = v_0 \exp \left\{ - \left(\frac{|x_i - x_j|}{r} \right)^\alpha \right\} + v_1 + v_2 \delta_{ij}$$

- Kernel parameters can be learned from data using optimization

v_0	signal variance
v_1	variance of bias
v_2	noise variance
r	lengthscale
α	roughness

GP for Regression

- Model $\mathcal{D} = \{(\mathbf{x}_i, y_i)_{i=1}^n\} = (\mathbf{X}, \mathbf{y})$

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

$$f \sim \text{GP}(\cdot|0, K)$$

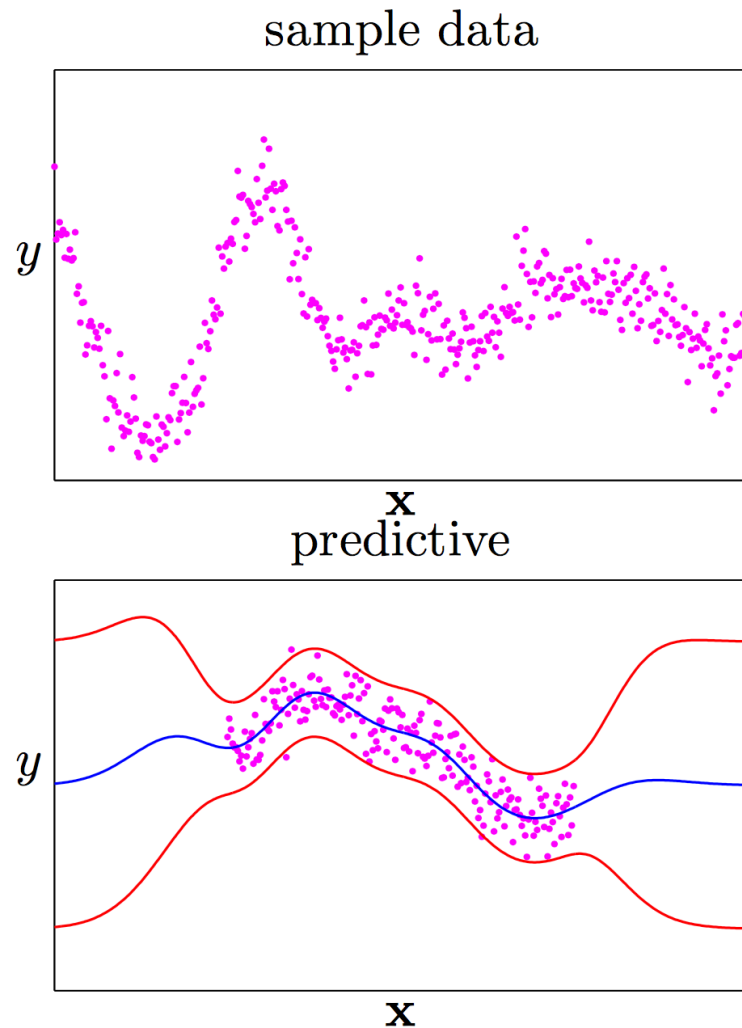
$$\epsilon_i \sim \text{N}(\cdot|0, \sigma^2)$$

- We can marginalize over f

$$p(y_*|\mathbf{x}_*, \mathcal{D}) = \int p(y_*|\mathbf{x}_*, f, \mathcal{D}) p(f|\mathcal{D}) df$$

GP Regression

Gaussian observation noise: $y_n = f_n + \epsilon_n$, where $\epsilon_n \sim \mathcal{N}(0, \sigma^2)$



marginal likelihood

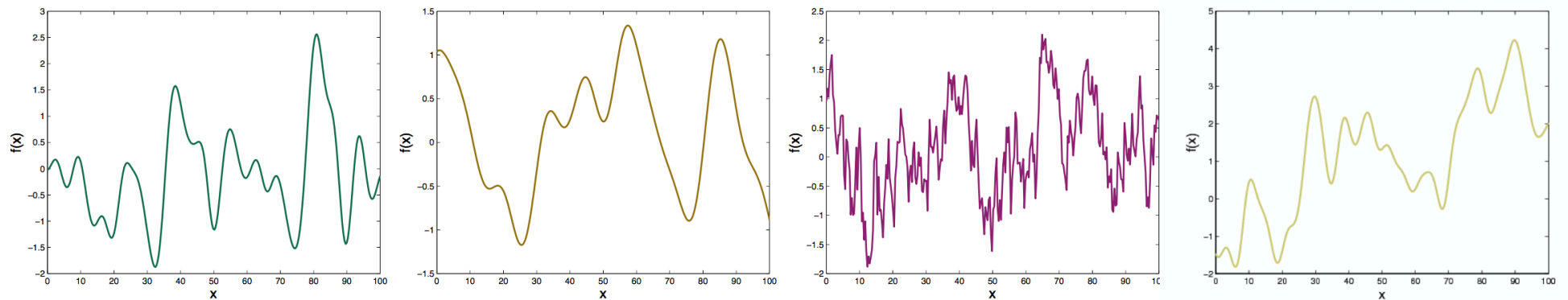
$$p(\mathbf{y}|\mathbf{X}) = \mathcal{N}(\mathbf{0}, \mathbf{K}_N + \sigma^2 \mathbf{I})$$

predictive distribution

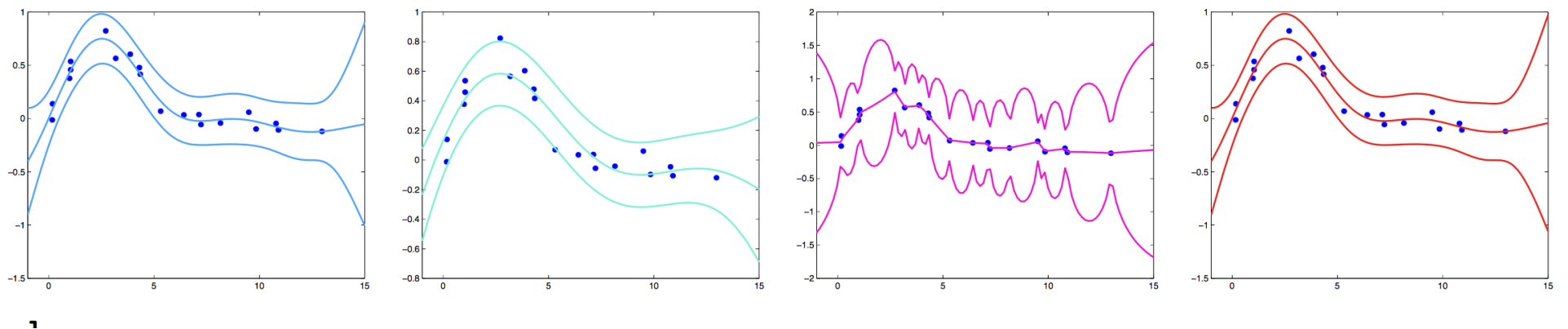
$$p(y_*|\mathbf{x}_*, \mathbf{X}, \mathbf{y}) = \mathcal{N}(\mu_*, \sigma_*^2)$$
$$\mu_* = \mathbf{K}_{*N}(\mathbf{K}_N + \sigma^2 \mathbf{I})^{-1} \mathbf{y}$$
$$\sigma_*^2 = K_{**} - \mathbf{K}_{*N}(\mathbf{K}_N + \sigma^2 \mathbf{I})^{-1} \mathbf{K}_{N*} + \sigma^2$$

Predictions for different kernels

A sample from the prior for each covariance function:



Corresponding predictions, mean with two standard deviations:



Learning the Kernel

$$K_{\boldsymbol{\theta}}(\mathbf{x}_i, \mathbf{x}_j) = v_0 \exp \left\{ - \sum_{d=1}^D \left(\frac{|x_i^{(d)} - x_j^{(d)}|}{r_d} \right)^\alpha \right\} + v_1$$

$$\boldsymbol{\theta} = (v_0, v_1, r_1, \dots, r_d, \alpha)$$

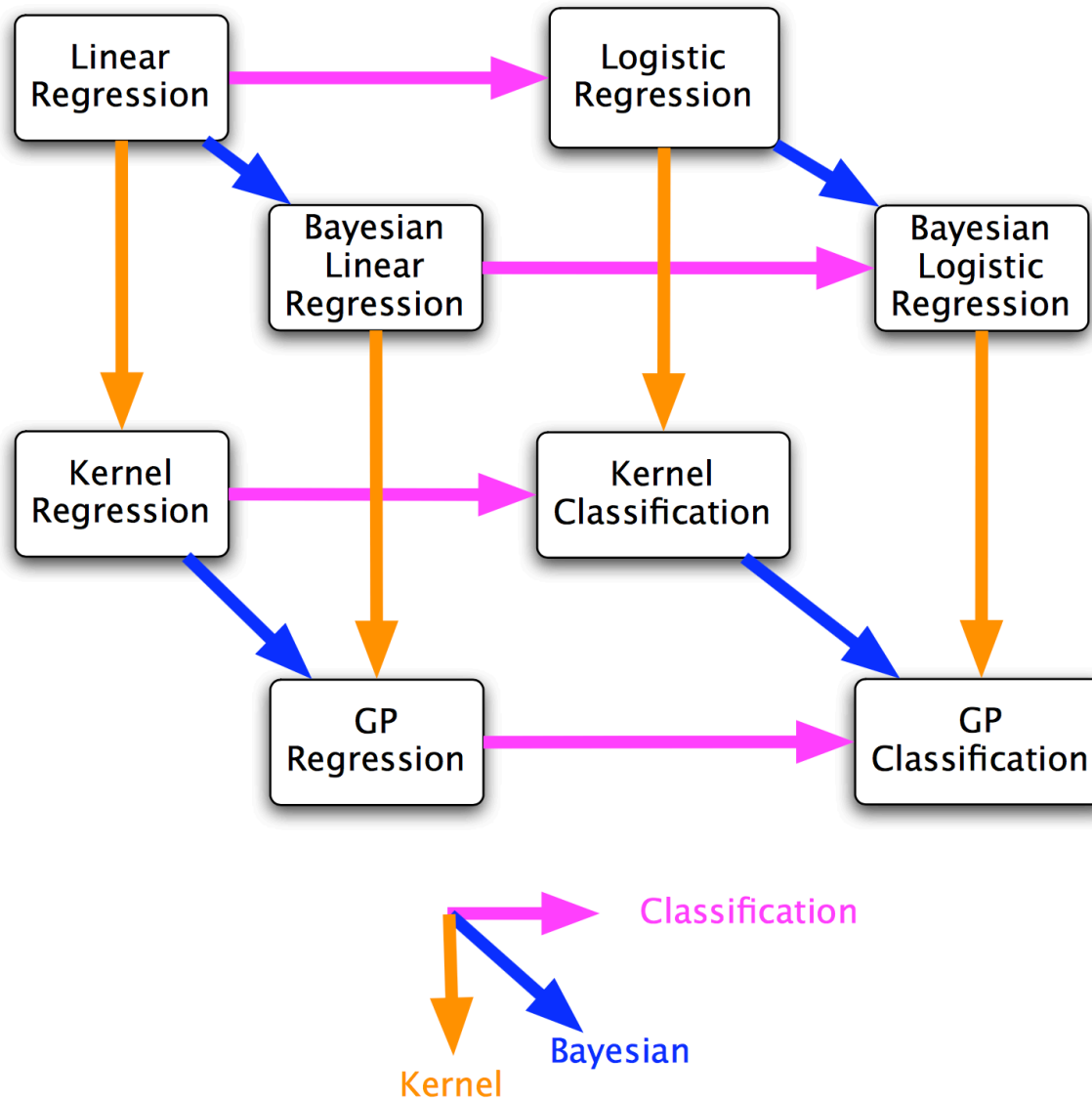
$$\ln p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}) = -\frac{1}{2} \ln \det(\mathbf{K}_{\boldsymbol{\theta}} + \sigma^2 \mathbf{I}) - \frac{1}{2} \mathbf{y}^\top (\mathbf{K}_{\boldsymbol{\theta}} + \sigma^2 \mathbf{I})^{-1} \mathbf{y} + \text{const}$$

- This can be viewed as a likelihood of θ given the data (x,y)
- We can use NL optimization to determine MLE θ

From Linear Regression to GP

- We started with a linear regression with inputs x_i and noisy y_i :
 $y_i = a + bx_i + \varepsilon_i$
- We generalized this to a general basis with M basis functions
 $y_i = \sum_{j=1}^M a_j \phi(x_i) + \varepsilon_i$
- Next we performed Bayesian regression by adding a gaussian prior on coefficients $a_j = N(0, \lambda_j)$
- This is equivalent to adding L_2 norm and minimize
 $l = \sum_{i=1}^N [\sum_{j=1}^M a_j \phi(x_i) - y_i]^2 + \sum_{j=1}^M \lambda_j a_j^2$
- Next we marginalize over a_j : we are left with $E(y_i) = 0$ and
 $K_{ij} = \text{Cov}(y(x_i), y(x_j)) = \sum_{j=1}^M \lambda_j \phi(x_i) \phi(x_j)$
- This is a gaussian process with a finite number of basis functions. Many GP kernels correspond to infinite number of basis functions.

Connections between regression/ classification models



Summary

- Interpolations: polynomial, spline, rational...
- They are just a subset of regression models, and one can connect them to regularized regression, which can be connected to Bayesian regression, which can be connected to kernel regression
- This can also be connected to classification, next lecture

Literature

- *Pattern Recognition and Machine Learning*, C. Bishop, Chapter 3
- *Bayesian Data Analysis*, Gelman et al. , Chapter 20-21
- <http://www.gaussianprocess.org>
- <https://arxiv.org/pdf/1505.02965.pdf>
- <http://mlss2011.comp.nus.edu.sg/uploads/Site/lect1gp.pdf>