Introduction to Machine Learning

Brown University CSCI 1950-F, Spring 2012
Prof. Erik Sudderth

Lecture 16:
Kernels & Perceptrons
Gaussian Process Regression & Classification

Many figures courtesy Kevin Murphy's textbook,
Machine Learning: A Probabilistic Perspective

Mercer Kernel Functions

$\mathcal{X} \longrightarrow$ arbitrary input space (vectors, functions, strings, graphs, …)

- A *kernel function* maps pairs of inputs to real numbers:

$$k : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R} \qquad k(x_i, x_j) = k(x_j, x_i)$$

Intuition: Larger values indicate inputs are "more similar"

- A kernel function is *positive semidefinite* if and only if for *any* $n \geq 1$, and *any* $x = \{x_1, x_2, \ldots, x_n\}$, the *Gram matrix* is positive semidefinite:

$$K \in \mathbb{R}^{n \times n} \qquad K_{ij} = k(x_i, x_j)$$

- Mercer's Theorem: Assuming certain technical conditions, every positive definite kernel function can be represented as

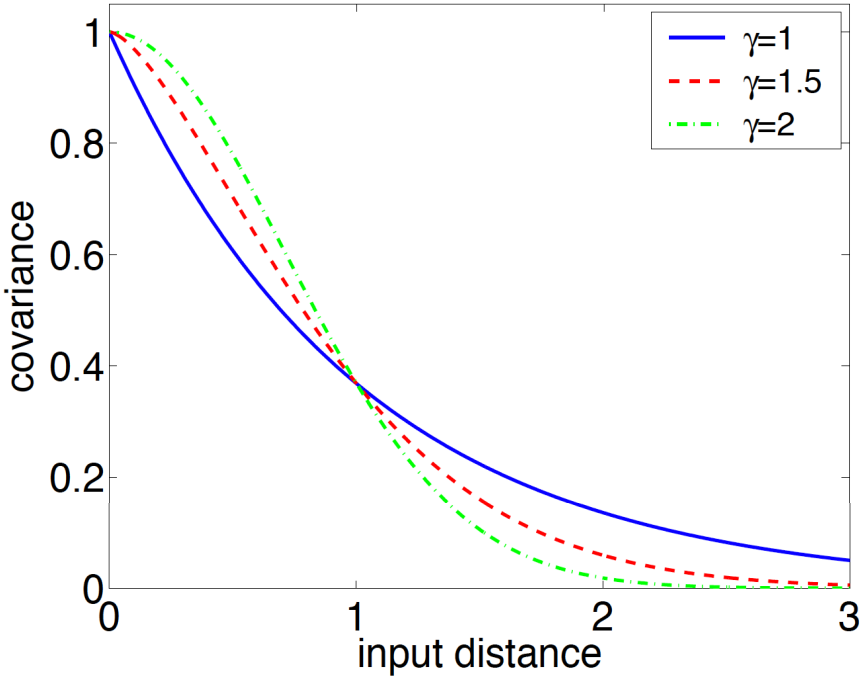
$$k(x_i, x_j) = \sum_{\ell=1}^{d} \phi_\ell(x_i)\phi_\ell(x_j)$$

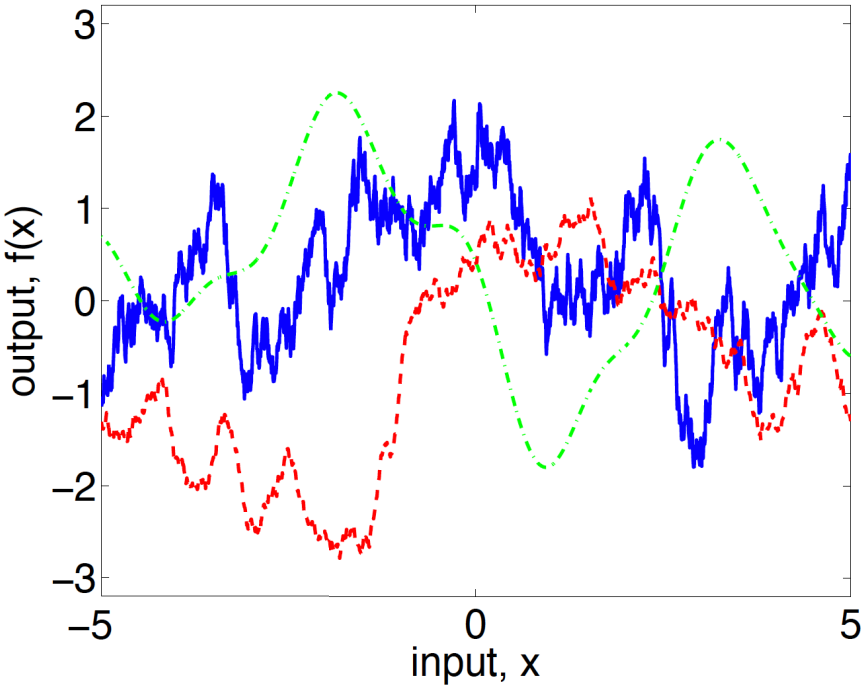
for some feature mapping ϕ (but may need $d \rightarrow \infty$)

Exponential Kernels

$\mathcal{X} \rightarrow$ real vectors of some fixed dimension

$$k(x_i, x_j) = \exp\left\{-\left(\frac{|x_i - x_j|}{\sigma}\right)^{\gamma}\right\} \qquad 0 < \gamma \leq 2$$





We can construct a covariance matrix by evaluating kernel at any set of inputs, and then sample from the zero-mean Gaussian distribution with that covariance. This is a Gaussian process.

String Kernels

$\mathcal{X} \longrightarrow$ strings of characters from some finite alphabet, of size *A*

Amino Acids $\{A, R, N, D, C, E, Q, G, H, I, L, K, M, F, P, S, T, W, Y, V\}$

x IPTSALVKETLALLSTHRTLLIANETLRIPVPVHKNHQLCTEEIFQGIGTLESQTVQGGTV
ERLFKNLSLIKKYIDGQKKKCGEERRRVNQFLDYLQEFLGVMNTEWI

x' PHRRDLCSRSIWLARKIRSDLTALTESYVKHQGLWSELTEAERLQENLQAYRTFHVLLA
RLLEDQQVHFTPTEGDFHQAIHTLLLQVAAFAYQIEELMILLEYKIPRNEADGMLFEKK
LWGLKVLQELSQWTVRSIHDLRFISSHQTGIP

- Feature vector: Count of number of times that *every substring*, of every possible length, occurs within string

$$D = A + A^2 + A^3 + A^4 + \cdots$$

- Using *suffix trees*, the kernel can be evaluated in time *linear* in the length of the input strings

Kernelizing Learning Algorithms

- Start with any learning algorithm based on features $\phi(x)$
  (Don't worry that computing features might be expensive or impossible.)
- Manipulate steps in algorithm so that it depends not directly on features, but only their inner products: $k(x_i, x_j) = \phi(x_i)^T \phi(x_j)$
- Write code that only uses calls to kernel function
- Basic identity: Squared distance between feature vectors

$$||\phi(x_i) - \phi(x_j)||_2^2 = k(x_i, x_i) + k(x_j, x_j) - 2k(x_i, x_j)$$

- Feature-based *nearest neighbor classification*
- Feature-based *clustering algorithms* (later)
- Feature-based *nearest centroid classification*:

$$\hat{y}_{\text{test}} = \arg\min_c ||\phi(x_{\text{test}}) - \mu_c||_2$$

$$\mu_c = \frac{1}{N_c} \sum_{i|y_i=c} \phi(x_i)$$

mean of the N_c training examples of class c

Perceptron MARK 1 Computer

Frank Rosenblatt, late 1950s

Decision Rule: $\hat{y}_i = \mathbb{I}(\theta^T \phi(x_i) > 0)$

Learning Rule:

$$\text{If } \hat{y}_k = y_k, \theta_{k+1} = \theta_k$$

$$\text{If } \hat{y}_k \neq y_k, \theta_{k+1} = \theta_k + \tilde{y}_k \phi(x_k)$$

$$\tilde{y}_k = 2y_k - 1 \in \{+1, -1\}$$

Kernelized Perceptron Algorithm

Decision Rule: $\hat{y}_{\text{test}} = \mathbb{I}(\theta^T \phi(x_{\text{test}}) > 0)$

Learning Rule:

$$\text{If } \hat{y}_k = y_k, \theta_{k+1} = \theta_k$$

$$\text{If } \hat{y}_k \neq y_k, \theta_{k+1} = \theta_k + \tilde{y}_k \phi(x_k)$$

$$\tilde{y}_k = 2y_k - 1 \in \{+1, -1\}$$

Representation:
D feature weights

Problem: May be intractable to compute/store $\phi(x_k), \theta_k$

Initialize with $\theta_0 = 0$. By induction, for all *k*

$$\theta_k = \sum_{i=1}^{N} s_{ik} \phi(x_i) \qquad \text{for some integers } s_{ik}$$

Representation:
N training example weights

Decision Rule: $\hat{y}_{\text{test}} = \mathbb{I}\left(\sum_{i=1}^{N} \hat{s}_i k(x_{\text{test}}, x_i) > 0 \right)$

Learning Rule:

$$\text{If } \hat{y}_k = y_k, s_{k,k+1} = s_{k,k}$$

$$\text{If } \hat{y}_k \neq y_k, s_{k,k+1} = s_{k,k} + \tilde{y}_k$$

Gaussian Processes

- Linear regression models predict outputs by a linear function of fixed, usually non-linear features:

$$f(x) = w^T \phi(x) \qquad \phi(x) \in \mathbb{R}^{m \times 1}$$

- Consider Gaussian prior on weight vector for regularization:

$$p(w) = \mathcal{N}(w \mid 0, \alpha^{-1} I_m) \qquad w \in \mathbb{R}^{m \times 1}$$

- What is the joint distribution of the predictions for any inputs?

$$x = \{x_1, x_2, \ldots, x_n\} \qquad f = [f(x_1), \ldots, f(x_n)]^T = \Phi w$$

$$p(f) = \mathcal{N}(f \mid 0, \alpha^{-1} \Phi \Phi^T) = \mathcal{N}(f \mid 0, K)$$

$$K_{ij} = \alpha^{-1} \phi(x_i)^T \phi(x_j)$$

- This is a *Gaussian process*: Not a single Gaussian distribution, but a family of Gaussian distributions, one for each *n* and *x*

Gaussian Process Regression

$$x = \{x_1, x_2, \ldots, x_n\} \qquad f = [f(x_1), \ldots, f(x_n)]^T = \Phi w$$

$$p(y_i \mid f_i) = \mathcal{N}(y_i \mid f_i, \beta^{-1})$$

noisy observation of unobserved function

- Feature-based regression estimates *m*-dim. feature vector
- GP regression estimates *n*-dim. function at training data:

$$p(f) = \mathcal{N}(f \mid 0, K) \qquad K_{ij} = \alpha^{-1}\phi(x_i)^T\phi(x_j)$$

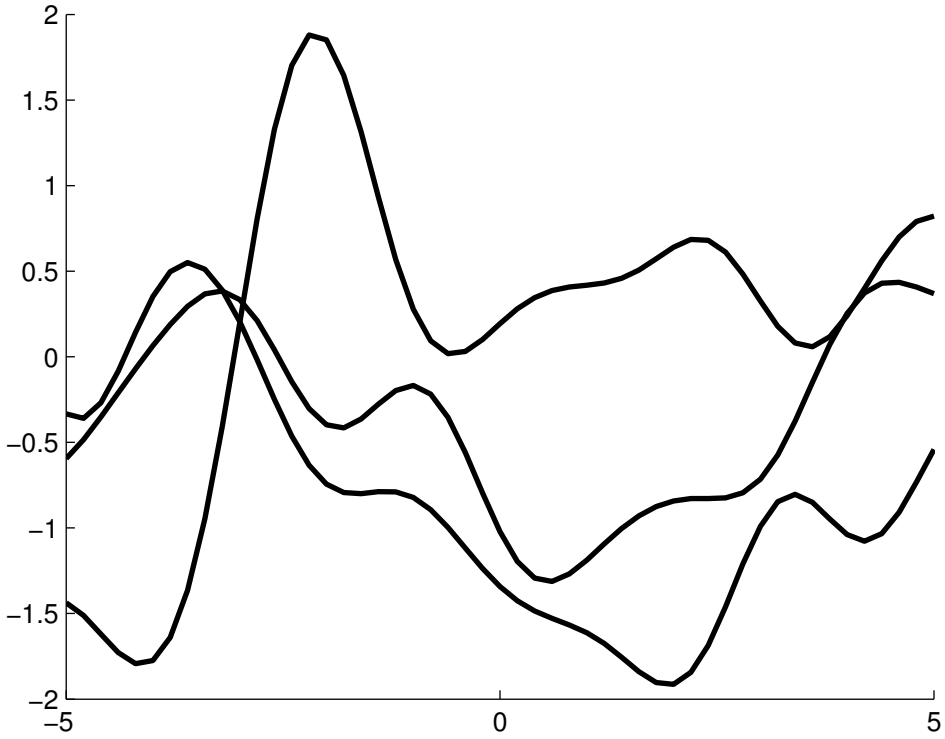
$$p(y) = \mathcal{N}(y \mid 0, C) \qquad C = K + \beta^{-1}I_n$$

- To make a prediction for a test point, we don't need to know the underlying weight vector, only the distribution

$$p(y_{n+1} \mid x_{n+1}, x, y) = \mathcal{N}(y_{n+1} \mid m(x_{n+1}), \sigma^2(x_{n+1}))$$

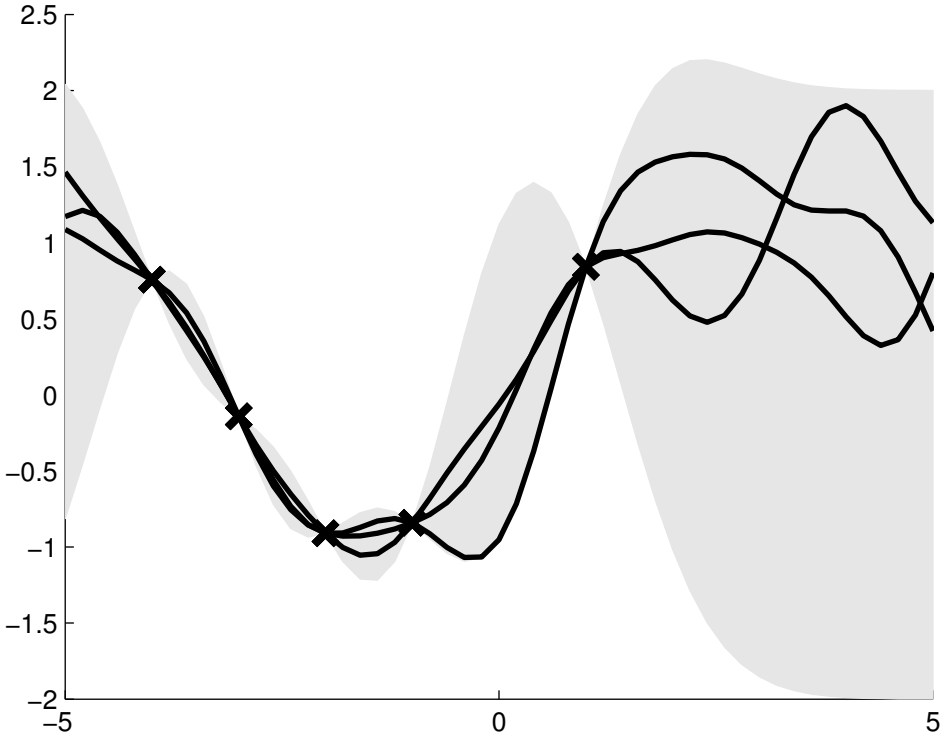
- Mean and covariance computed by applying standard formulas for Gaussian conditionals to covariance matrix *C*

1D Gaussian Process Regression



Samples from Prior

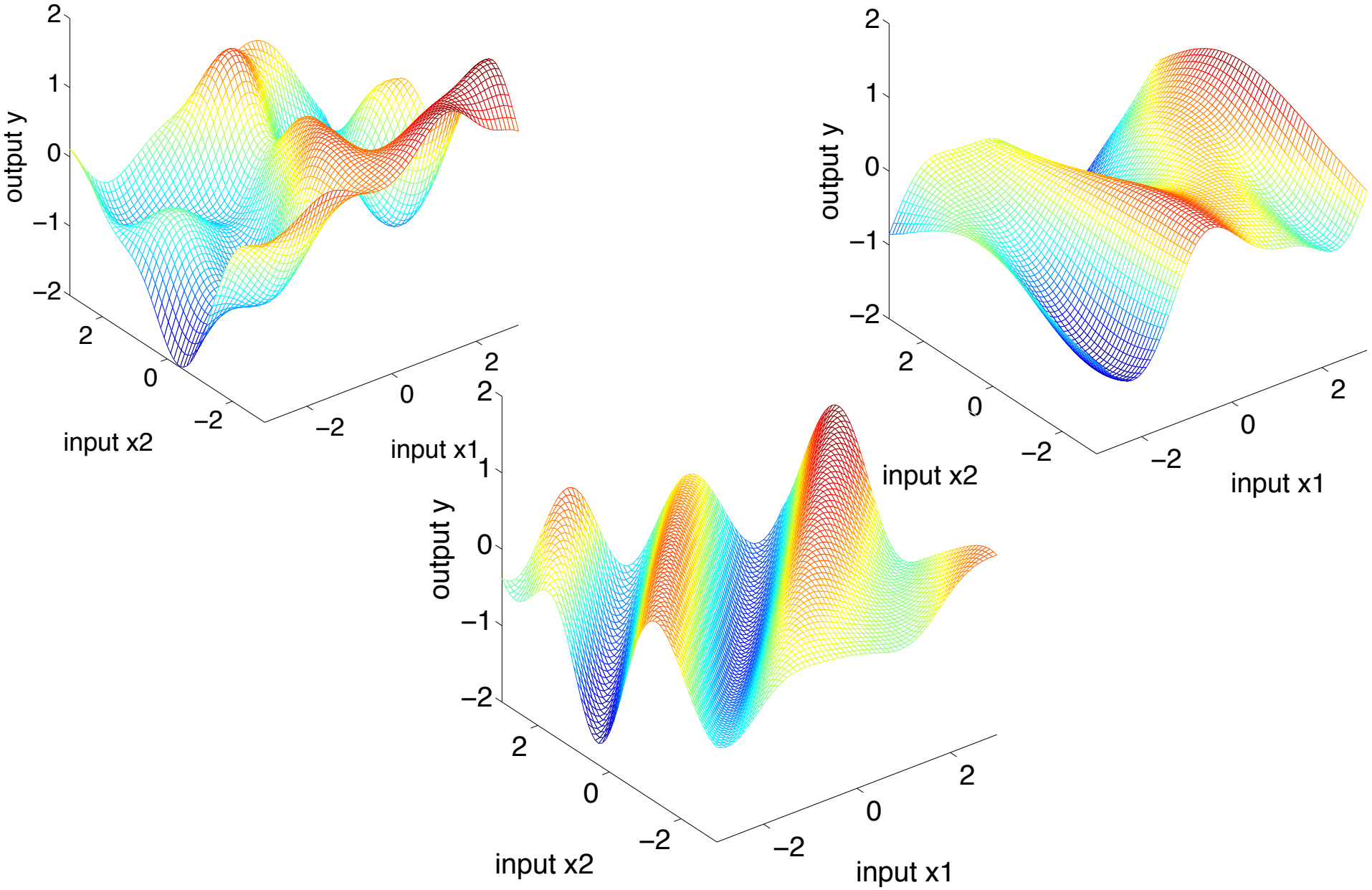
$$\kappa(x, x') = \sigma_f^2 \exp(-\frac{1}{2\ell^2}(x - x')^2)$$



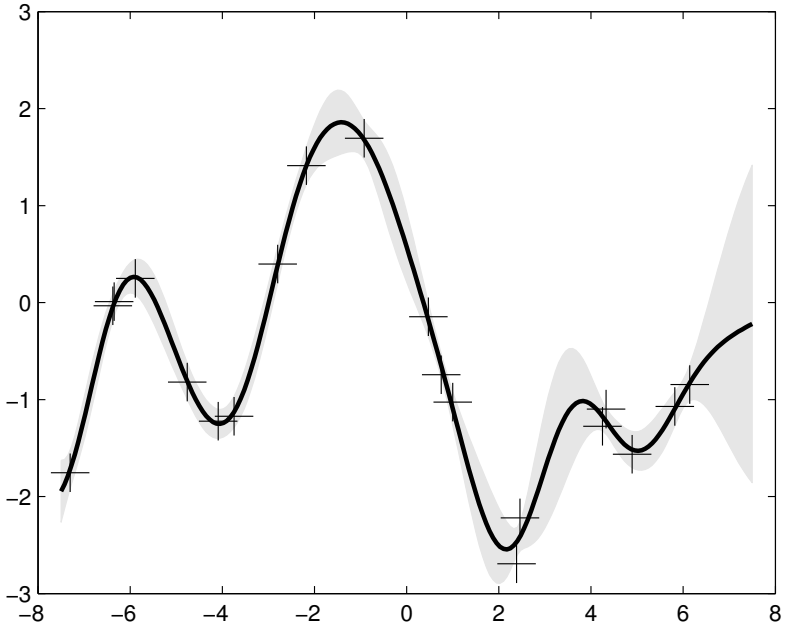
Posterior Given 5 Noise-Free Observations

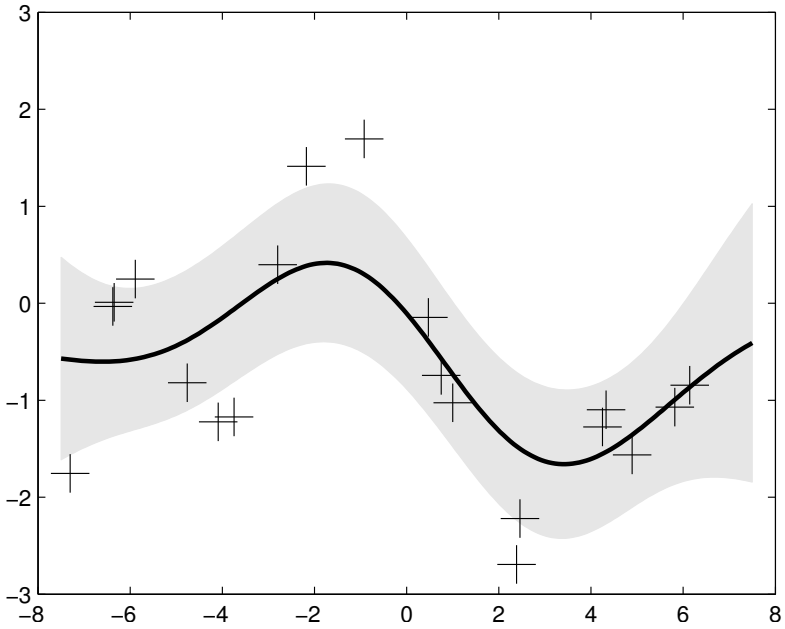
Squared exponential kernel or radial basis function (RBF) kernel

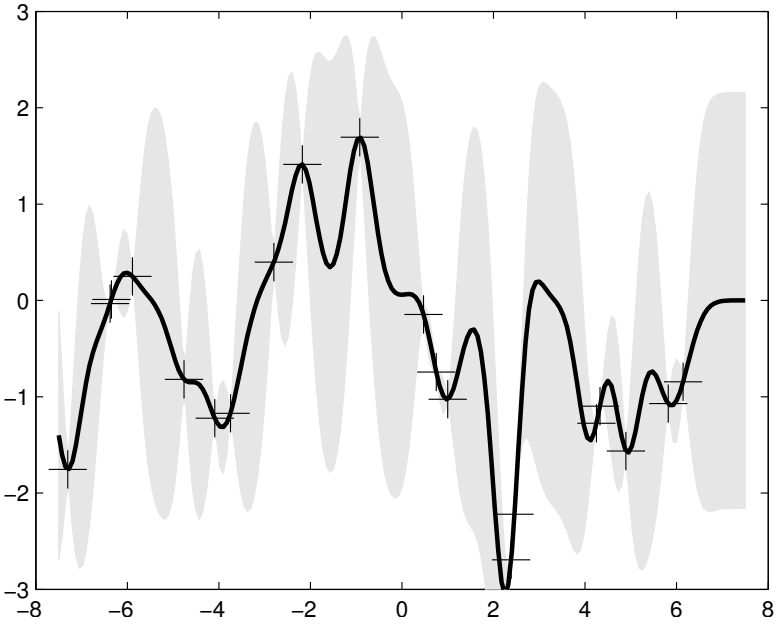
2D Gaussian Processes



Gaussian Process Hyperparameters





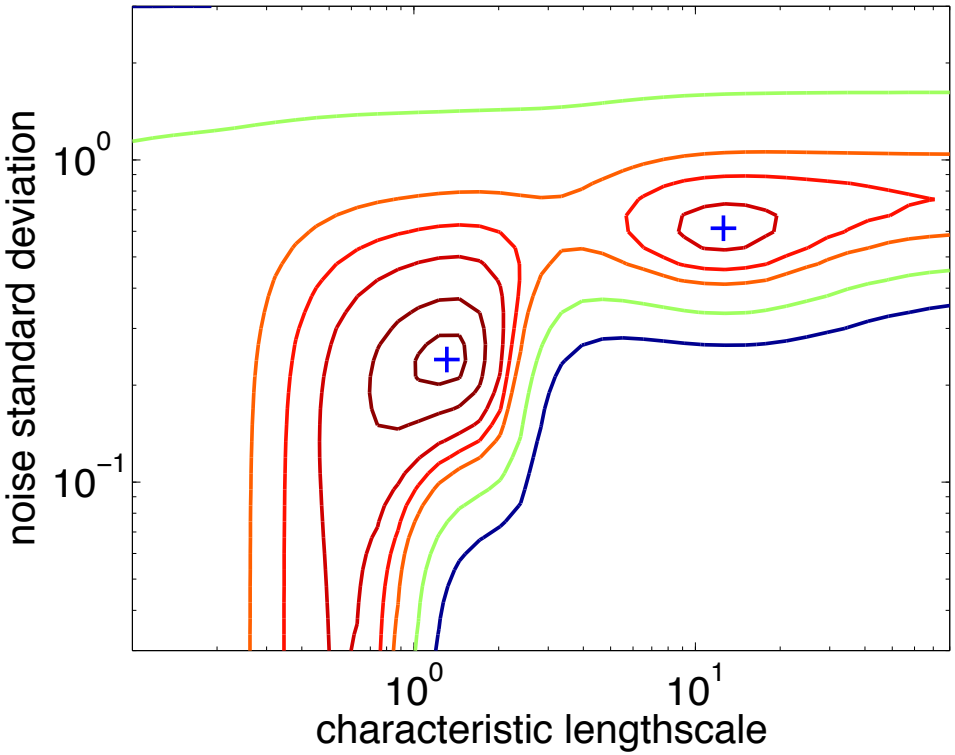


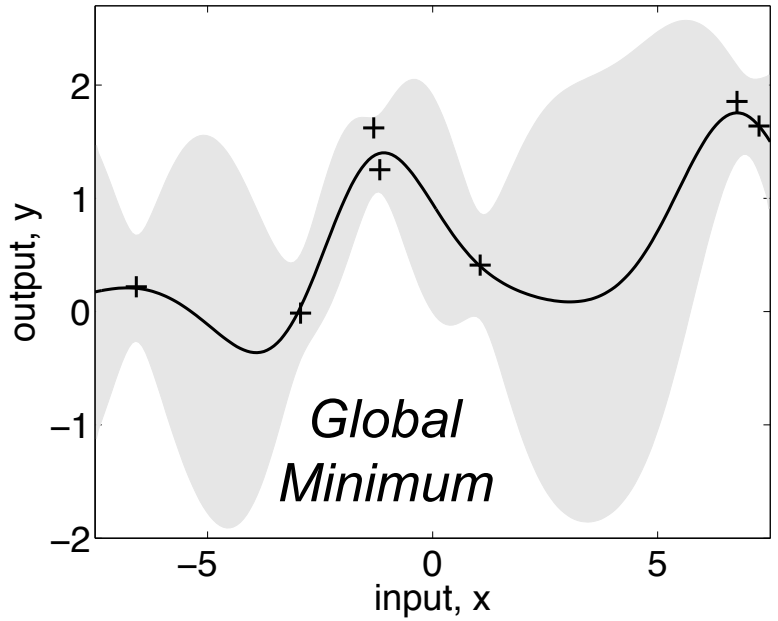
$$\kappa(x, x') = \sigma_f^2 \exp(-\frac{1}{2\ell^2}(x - x')^2)$$

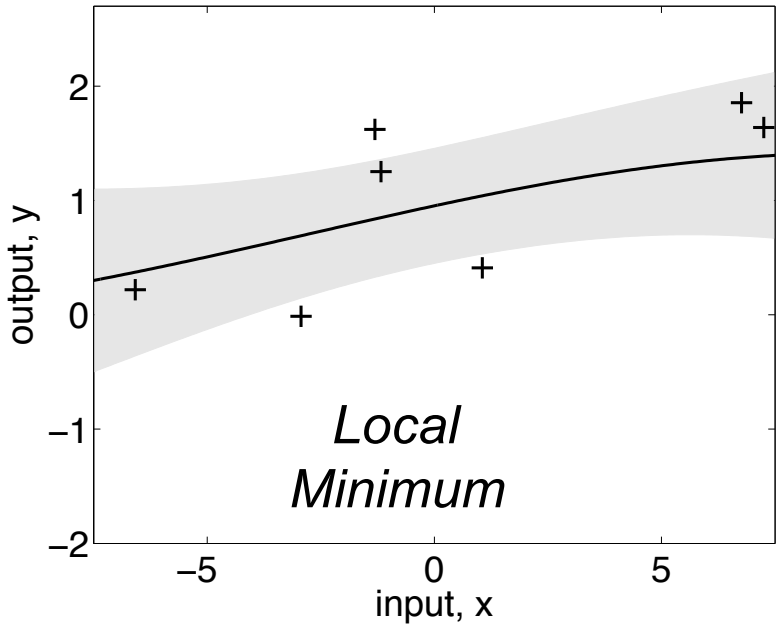
How should we fit to data?

- Cross-validation
- Maximize marginal likelihood (empirical Bayes, tractable for GP regression)

Hyperparameter Marginal Likelihoods



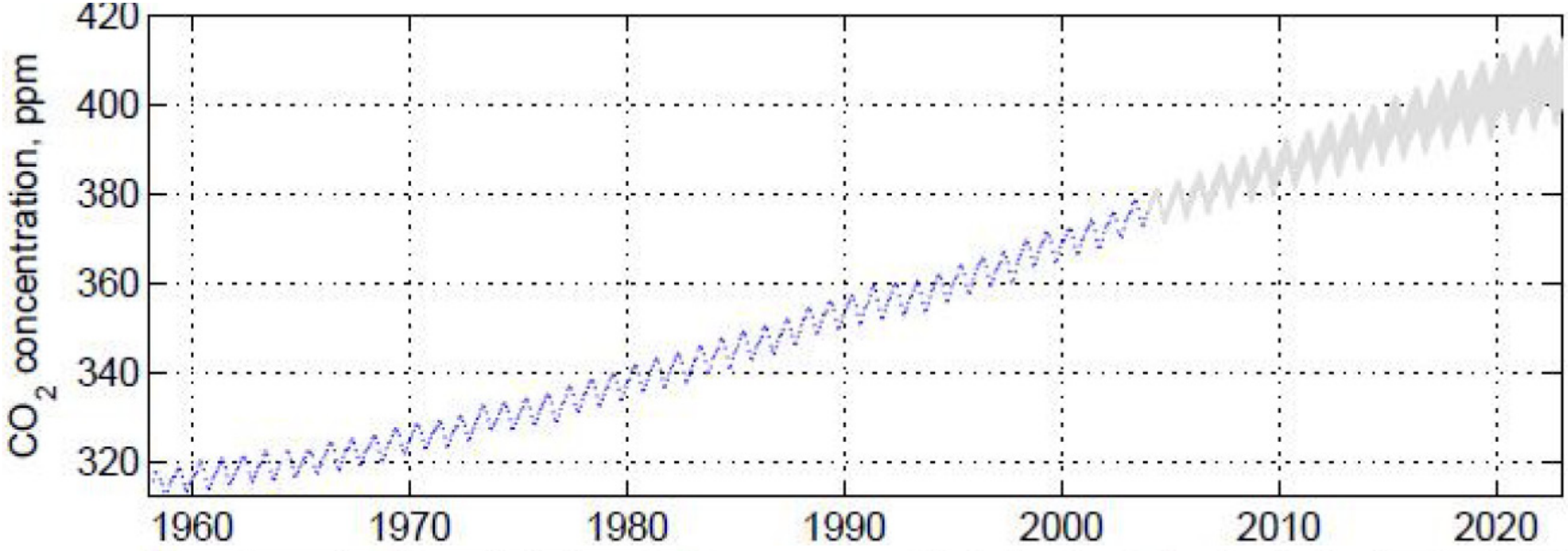


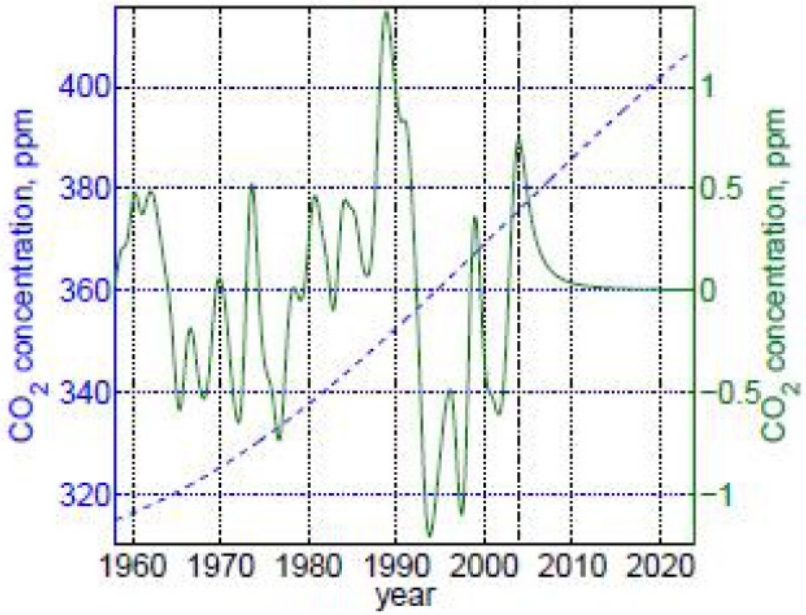


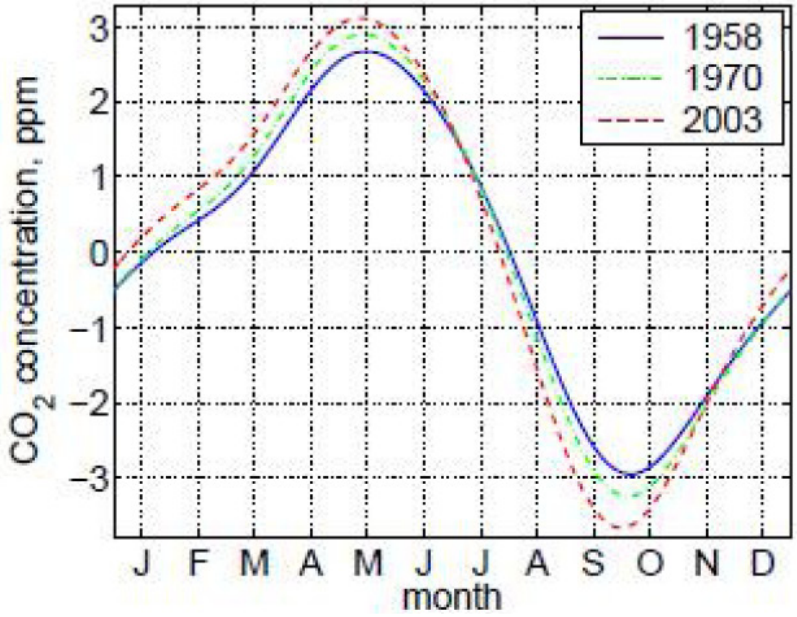
$$p(\mathbf{y}|\mathbf{X}) = \int p(\mathbf{y}|\mathbf{f},\mathbf{X})p(\mathbf{f}|\mathbf{X})d\mathbf{f}$$

$$\log p(\mathbf{y}|\mathbf{X}) = -\frac{1}{2}\mathbf{y}\mathbf{K}_y^{-1}\mathbf{y} - \frac{1}{2}\log|\mathbf{K}_y| - \frac{N}{2}\ln(2\pi)$$

Example: CO_2 Concentration Over Time







Mauna Loa Observatory in Hawaii, analyzed by Rasmussen & Williams 2006

Mixing Kernels for CO_2 GP Regression

Smooth global trend

$$\kappa_1(x, x') = \theta_1^2 \exp\left(-\frac{(x - x')^2}{2\theta_2^2}\right)$$

Seasonal periodicity

$$\kappa_2(x, x') = \theta_3^2 \exp\left(-\frac{(x - x')^2}{2\theta_4^2} - \frac{2\sin^2(\pi(x - x'))}{\theta_5^2}\right)$$

Medium term irregularities

$$\kappa_3(x, x') = \theta_6^2 \left(1 + \frac{(x - x')^2}{2\theta_8\theta_7^2}\right)^{-\theta_8}$$

Correlated Observation Noise

$$\kappa_4(x_p, x_q) = \theta_9^2 \exp\left(-\frac{(x_p - x_q)^2}{2\theta_{10}^2}\right) + \theta_{11}^2\delta_{pq}$$

Generalized Linear Models

- Recall *parametric* generalized linear models (GLMs):

$$p(y_i \mid x_i, w) = \exp\{y_i f_i - A(f_i)\}$$

any exponential family distribution

$$f_i = w^T \phi(x_i)$$

$$p(w) = \mathcal{N}(w \mid 0, \alpha^{-1} I_m) \qquad w \in \mathbb{R}^{m \times 1}$$

- Gaussian processes lead to *nonparametric* GLMs:

$$p(y_i \mid x_i, f_i) = \exp\{y_i f_i - A(f_i)\}$$

any exponential family distribution

$$p(f) = \mathcal{N}(f \mid 0, K) \qquad K_{ij} = k(x_i, x_j)$$

- The Mercer kernel function corresponds to some set of underlying features, but we need not know or compute them
- The model is “nonparametric” because the number of underlying features, and hence parameters, can be infinite

Gaussian Process Classification

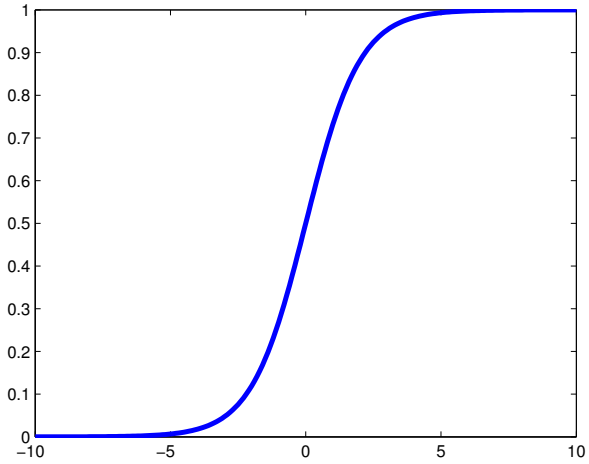
$$y_i \in \{0, 1\}$$

$$p(y_i \mid x_i, f_i) = \exp\{y_i f_i - A(f_i)\}$$

Bernoulli distribution

$$p(f) = \mathcal{N}(f \mid 0, K) \qquad K_{ij} = k(x_i, x_j)$$

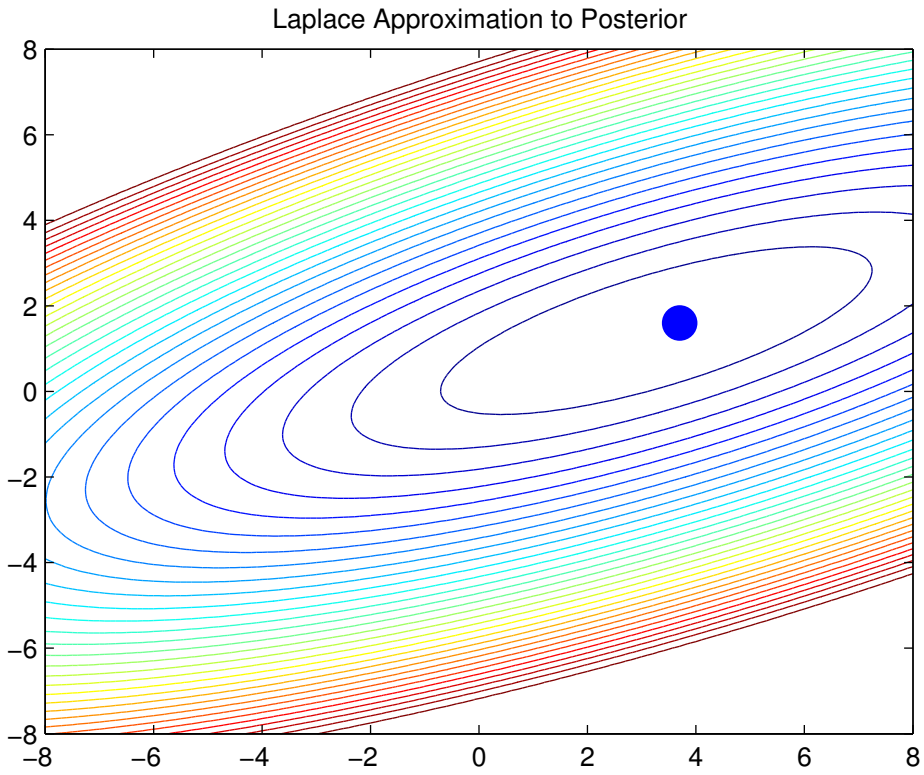
$$p(y_i \mid x_i, f_i) = \mathrm{Ber}(y_i \mid \mathrm{sigm}(f_i))$$

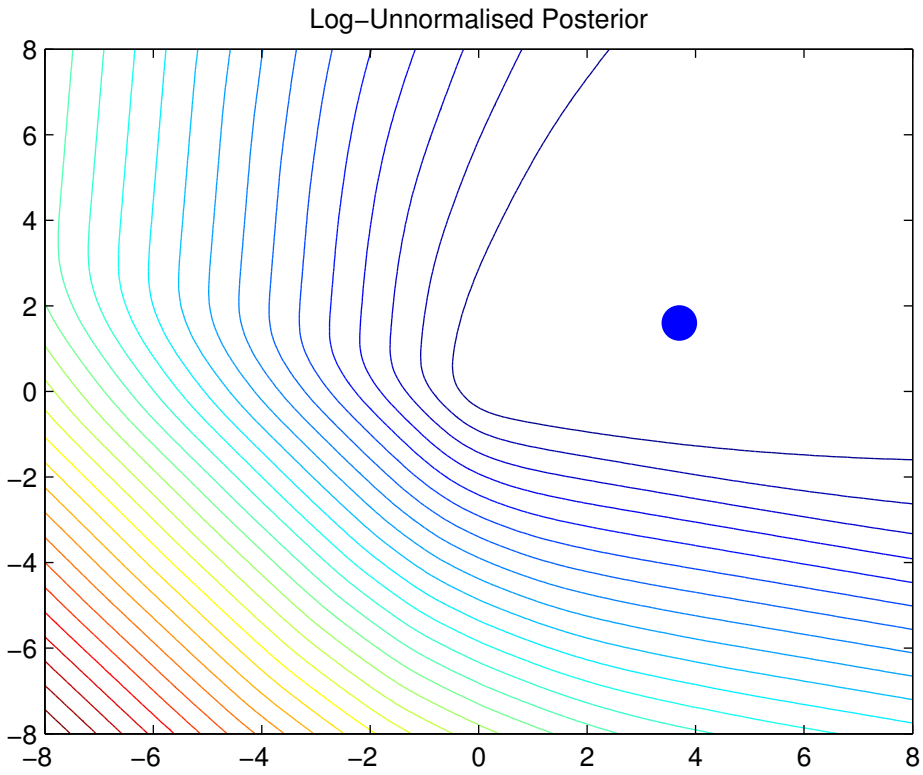


$$\mathrm{sigm}(\eta) := \frac{1}{1 + \exp(-\eta)} = \frac{e^{\eta}}{e^{\eta} + 1}$$

- Equivalent to logistic regression, but uses kernels rather than features
- Gaussian prior on weights replaced by Gaussian prior on training log-odds
- As in logistic regression, cannot exactly average of parameters to compute test data predictions
- Use Gaussian approximations instead

Laplace Approximations





Laplace (Gaussian) Approximation *Log Posterior Distribution*

- Logistic regression approximates M-dim. distribution of weights *w*
- GP classification approximates N-dim. distribution of training log-odds *f*
- Both require similar gradient descent algorithms

Kernels or Features?

$N \longrightarrow$ number of training examples

$M \longrightarrow$ number of features

$L \longrightarrow$ cost of kernel function evaluation, at worst $\mathcal{O}(M)$

$\Phi \longrightarrow$ *NxM* matrix evaluating each feature for all training data

- Feature-based linear regression: $\mathcal{O}(NM^2 + M^3)$
- Kernel-based GP regression: $\mathcal{O}(LN^2 + N^3)$
- Roughly, the difference corresponds to using either

$$(\Phi^T \Phi)^{-1} \qquad (\Phi \Phi^T)^{-1}$$

- Relative costs of logistic regression and GP classification are similar, per iteration of optimization-based learning
- What if *N* and *M* are both large???

Approximate!!! Endless options, none perfect…