Music and Machine Learning (IFT6080 Winter 08)
Prof. Douglas Eck, Université de Montréal

These slides follow closely the (English) course textbook
Pattern Recognition and Machine Learning
by Christopher Bishop

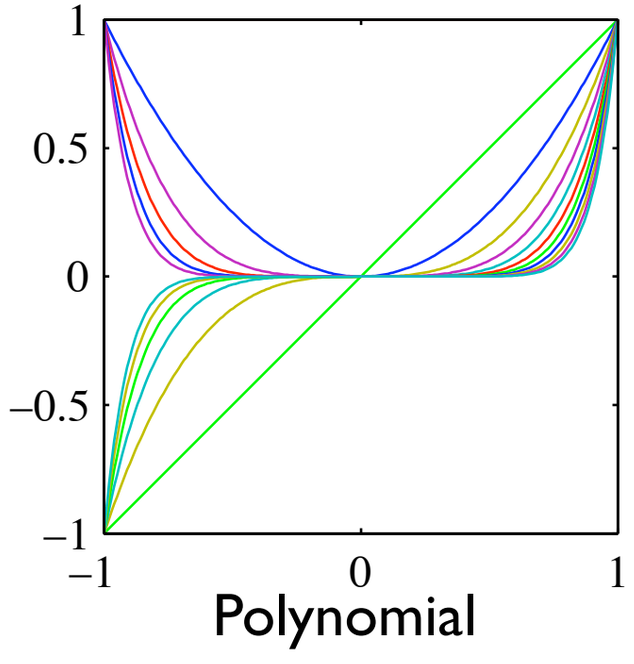
Linear Basis Function Models

- Linear regression extended to consider fixed basis functions:

$$y(\mathbf{x}, \mathbf{w}) = \sum_{j=0}^{M-1} w_j \phi_j(\mathbf{x}) = \mathbf{w}^{\mathrm{T}} \boldsymbol{\phi}(\mathbf{x})$$

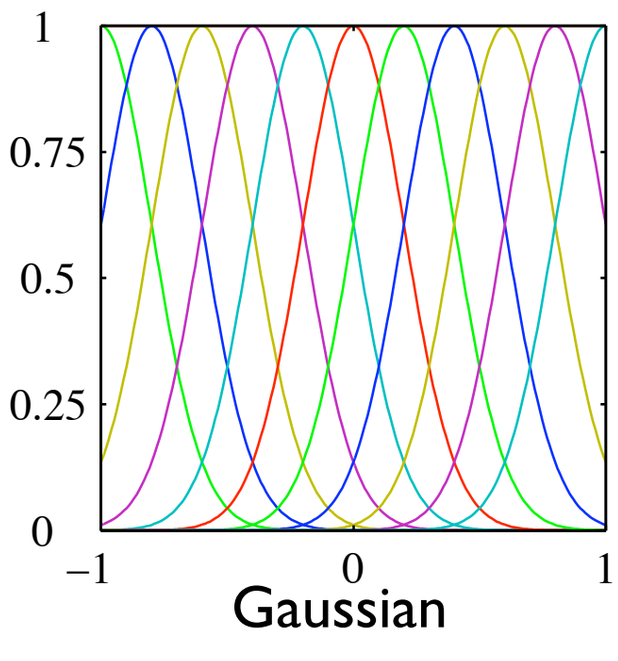
where $\mathbf{w} = (w + 0, ..., w_{M-1})^{\mathrm{T}}$ and $\boldsymbol{\phi} = (\phi_0, ..., \phi_{M-1})^{\mathrm{T}}$

- Possible basis functions include polynomials, Fourier, wavelet, ...



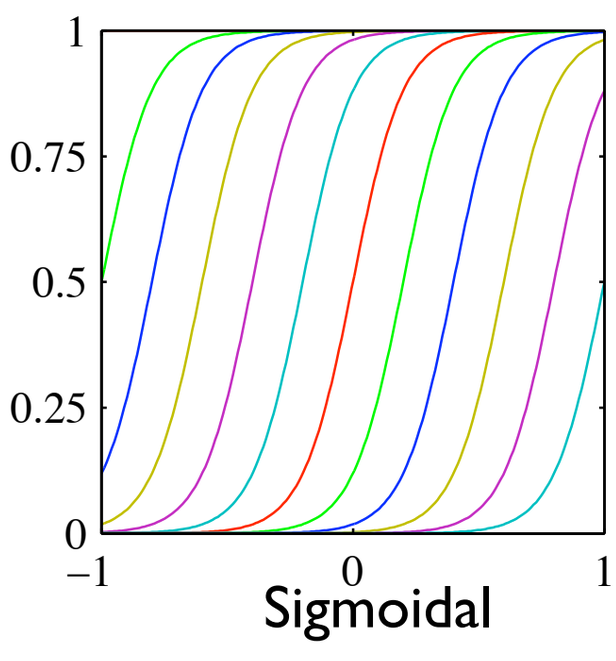
Polynomial

$$\phi_j(x) = x^j$$



Gaussian

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



Sigmoidal

$$\phi_j(x) = \sigma\left(\frac{x = \mu_j}{s}\right)$$

$$\text{where } \sigma(a) = \frac{1}{1 + \exp(-a)}$$

Maximum likelihood and least squares

- Presume target *t* is generated via deterministic function plus gaussian noise ϵ having precision β

$$t = y(\mathbf{x}, \mathbf{w}) + \epsilon$$

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

- With Gaussian conditional distribution conditional mean is:

$$\mathbb{E}[t|\mathbf{x}] = \int tp(t|\mathbf{x})\mathrm{d}t = y(\mathbf{x}, \mathbf{w})$$

- With a set of input points $\mathbf{X} = \{\mathbf{x}_1, ..., \mathbf{x}_N\}$ independently drawn:

$$p(\mathbf{t}|\mathbf{X}, \mathbf{w}, \beta) = \prod_{n=1}^{N} \mathcal{N}(t_n|\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n), \beta^{-1})$$

Maximum likelihood and least squares

- Log likelihood:

$$\begin{aligned}\ln p(\mathbf{t}|\mathbf{w},\beta) &= \sum_{n=1}^{N} \ln \mathcal{N}(t_n|\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n),\beta^{-1}) \\ &= \frac{N}{2}\ln\beta - \frac{N}{2}\ln(2\pi) - \beta E_D(\mathbf{w})\end{aligned}$$

$$\text{where } E_D(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\left\{t_n - \mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n)\right\}^2$$

- Gradient:

$$\nabla \ln p(\mathbf{t}|\mathbf{w},\beta) = \sum_{n=1}^{N}\left\{t_n - \mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n)\right\}\boldsymbol{\phi}(\mathbf{x}_n)^{\mathrm{T}}$$

Maximum likelihood and least squares

- Set gradient to 0:

$$0 = \sum_{n=1}^{N} t_n \boldsymbol{\phi}(\mathbf{x}_n)^{\mathrm{T}} - \mathbf{w}^{\mathrm{T}} \left(\sum_{n=1}^{N} \boldsymbol{\phi}(\mathbf{x}_n) \boldsymbol{\phi}(\mathbf{x}_n)^{\mathrm{T}} \right)$$

- Solve for our weights yields *normal equations* for least squares:

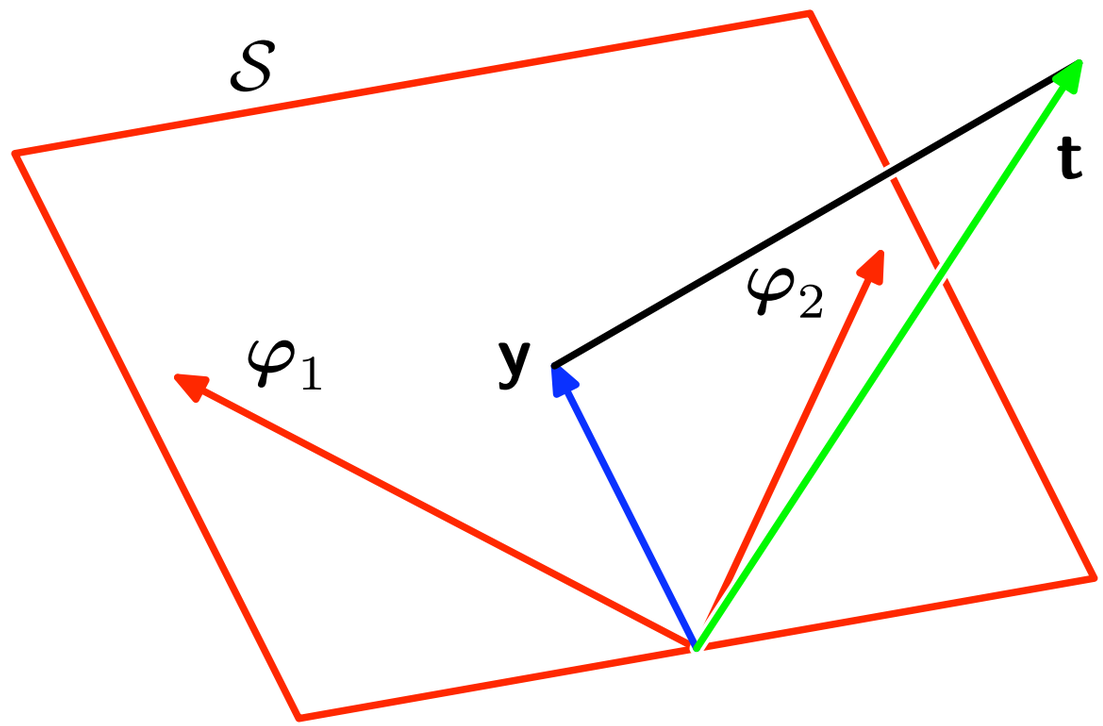
$$\mathbf{w}_{\mathrm{ML}} = \left(\boldsymbol{\Phi}^{\mathrm{T}} \boldsymbol{\Phi} \right)^{-1} \boldsymbol{\Phi}^{\mathrm{T}} \mathbf{t}$$

- where Φ is the design matrix

$$\boldsymbol{\Phi} = \begin{pmatrix} \phi_0(\mathbf{x}_1) & \phi_1(\mathbf{x}_1) & \cdots & \phi_{M-1}(\mathbf{x}_1) \\ \phi_0(\mathbf{x}_2) & \phi_1(\mathbf{x}_2) & \cdots & \phi_{M-1}(\mathbf{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_0(\mathbf{x}_N) & \phi_1(\mathbf{x}_N) & \cdots & \phi_{M-1}(\mathbf{x}_N) \end{pmatrix}$$

- and where $\boldsymbol{\Phi}^{\dagger} = \left(\boldsymbol{\Phi}^{\mathrm{T}} \boldsymbol{\Phi}^{-1} \right) \boldsymbol{\Phi}^{\mathrm{T}}$ is the pseudo-inverse of Φ

Geometry of least squares



- Least-squares regression is obtained by finding the orthogonal projection of the data vector **t** onto the subspace spanned by the basis functions.
- Intuition: Sum of squares error is 1/2 squared Euclidean distance between **y** and **t**. Thus least-squares solution would move **t** as close as possible to **y** in the subspace *S*.

Online learning

- For large datasets may need to learn sequentially on sequences of smaller datasets, summing error

$$E = \sum_n E_n$$

- Sequential gradient descent (also called *stochastic gradient descent*:

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \nabla E_n$$

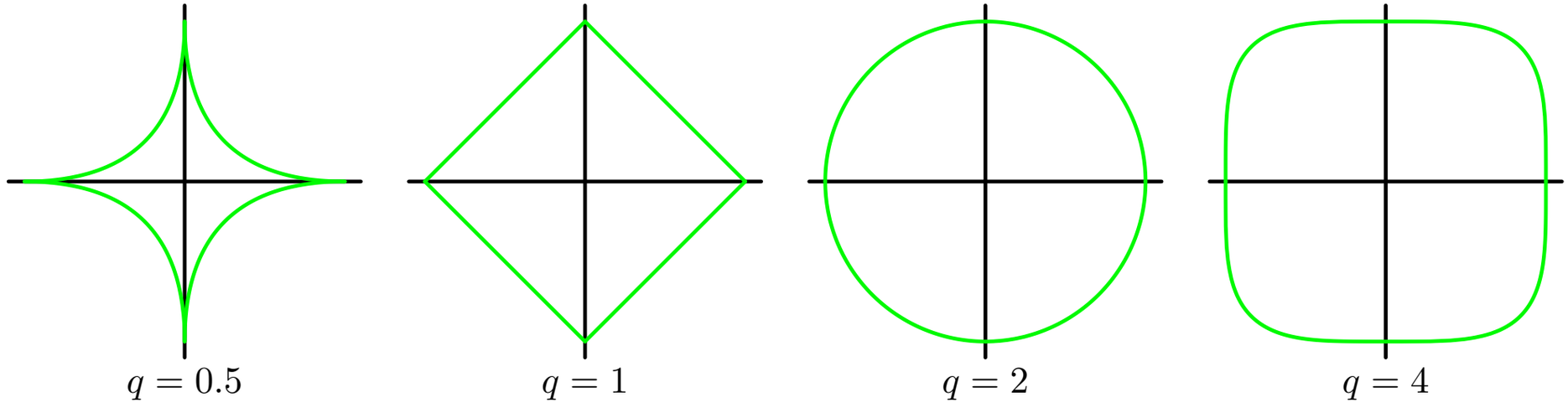
where τ is the iteration number and η is the learning rate

- For sum-of-squares we get Least Mean Squares (LMS):

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} + \eta(t_n - \mathbf{w}^{(\tau)\mathrm{T}}\boldsymbol{\phi}_n)\boldsymbol{\phi}_n$$

- Learning rate must be chosen carefully

Regularized least squares



- Regularize magnitude of weights:

$$\frac{1}{2}\sum_{n=1}^{N}\left\{t_n - \mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n)\right\}^2 + \frac{\lambda}{2}\mathbf{w}^{\mathrm{T}}\mathbf{w}$$

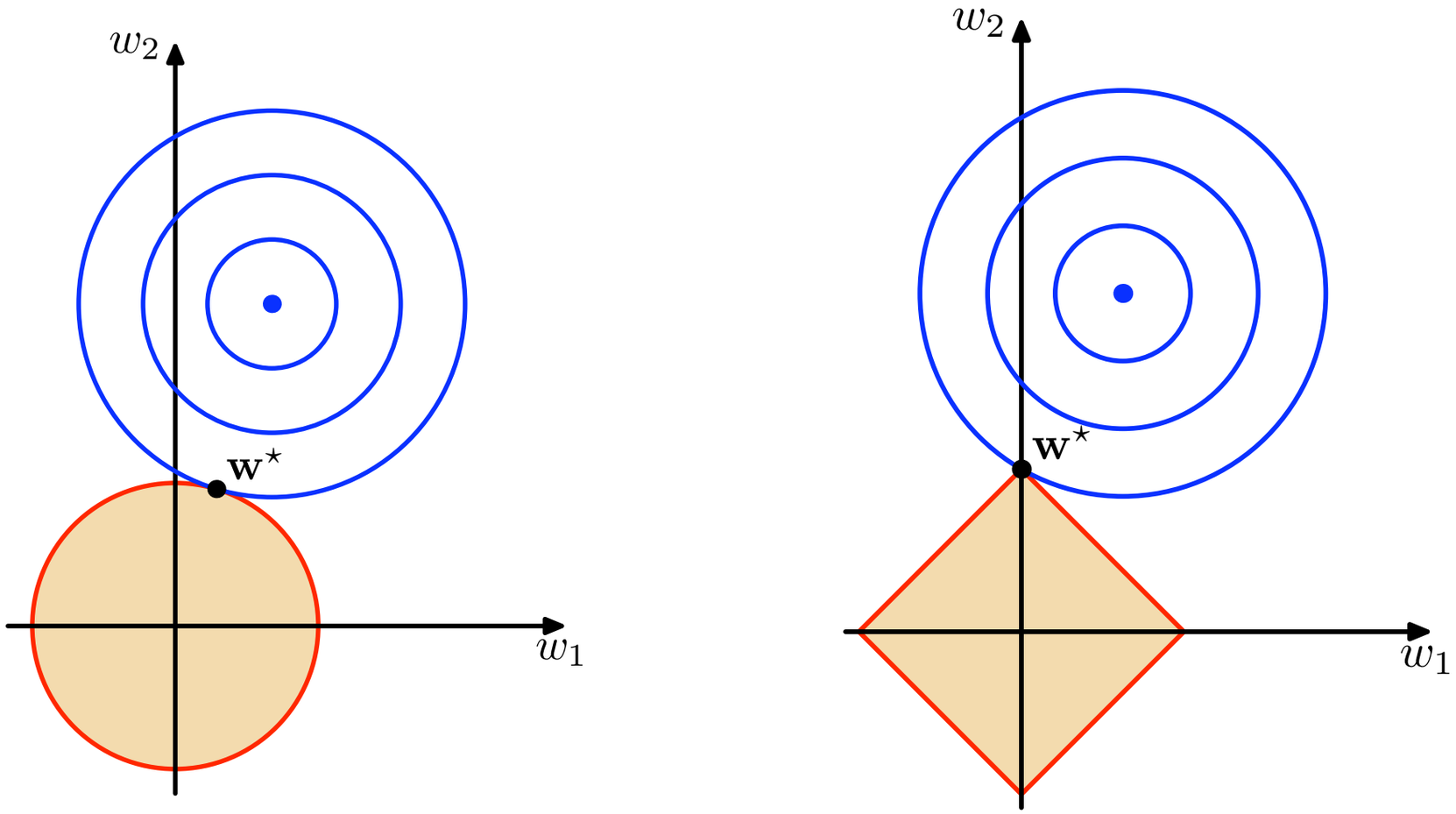
- Gradient with respect to 0 yields extension of least squares:

$$\mathbf{w} = \left(\lambda\mathbf{I} + \boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}\right)^{-1}\boldsymbol{\Phi}^{\mathrm{T}}\mathbf{t} \qquad \mathbf{w}_{\mathrm{ML}} = \left(\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}\right)^{-1}\boldsymbol{\Phi}^{\mathrm{T}}\mathbf{t}$$

- More general regularizer; when q=1 we have "lasso" regularizer which selects for sparse models:

$$\frac{1}{2}\sum_{n=1}^{N}\left\{t_n - \mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n)\right\}^2 + \frac{\lambda}{2}\sum_{j=1}^{M}|w_j|^q$$

Visualization of regularized least squares



Plot of contours of unregularized error function along with constraint region on the quadratic regularizer (left, q=2) versus lasso regularizer (right, q=1). For lasso, a sparse solution is generated with $w_1=0$

Bias-Variance decomposition

- How to best set the λ parameter for regularization?
- Conditional expectation:

$$h(\mathbf{x}) = \mathbb{E}[t|\mathbf{x}] = \int t p(t|\mathbf{x}) \mathrm{d}t$$

- Expected squared loss written with noise as second term:

$$\mathbb{E}[L] = \int \{y(\mathbf{x}) - h(\mathbf{x})\}^2 p(\mathbf{x})\, \mathrm{d}\mathbf{x} + \int \{h(\mathbf{x}) - t\}^2 p(\mathbf{x}, t)\, \mathrm{d}\mathbf{x}\, \mathrm{d}t$$

- We will minimize the first term. But we cannot hope to ever know the perfect regression function *h*(x)
- In a Bayesian model uncertainty is expressed as posterior over **w**
- In frequentist treatment we make a point-estimate of **w**. Assess confidence by making predictions over subsets of data and taking mean performance.

Bias-Variance decomposition

- Take integrand of first term using some subset of data *D*.
  $\{y(\mathbf{x}, \mathcal{D}) - h(\mathbf{x})\}^2$ which varies with data, thus take its mean

- Add and subtract expected value for the data

$$\{y(\mathbf{x}; \mathcal{D}) - \mathbb{E}_{\mathcal{D}}[y(\mathbf{x}; \mathcal{D})] + \mathbb{E}_{\mathcal{D}}[y(\mathbf{x}; \mathcal{D})] - h(\mathbf{x})\}^2$$

$$= \{y(\mathbf{x}; \mathcal{D}) - \mathbb{E}_{\mathcal{D}}[y(\mathbf{x}; \mathcal{D})]\}^2 + \{\mathbb{E}_{\mathcal{D}}[y(\mathbf{x}; \mathcal{D})] - h(\mathbf{x})\}^2$$
$$+2\{y(\mathbf{x}; \mathcal{D}) - \mathbb{E}_{\mathcal{D}}[y(\mathbf{x}; \mathcal{D})]\}\{\mathbb{E}_{\mathcal{D}}[y(\mathbf{x}; \mathcal{D})] - h(\mathbf{x})\}$$

- Take expectation with respect to *D*; final term vanishes

$$\mathbb{E}_{\mathcal{D}}\left[\{y(\mathbf{x}; \mathcal{D}) - h(\mathbf{x})\}^2\right]$$
$$= \{\mathbb{E}_{\mathcal{D}}[y(\mathbf{x}; \mathcal{D})] - h(\mathbf{x})\}^2 + \mathbb{E}_{\mathcal{D}}\left[\{y(\mathbf{x}; \mathcal{D}) - \mathbb{E}_{\mathcal{D}}[y(\mathbf{x}; \mathcal{D})]\}^2\right]$$

(bias)^2 variance

- First term is bias : extent to which average prediction differs from desired regression function

- Second term is variance: extent to which individual solutions vary around the average. Thus measures sensitivity to data.

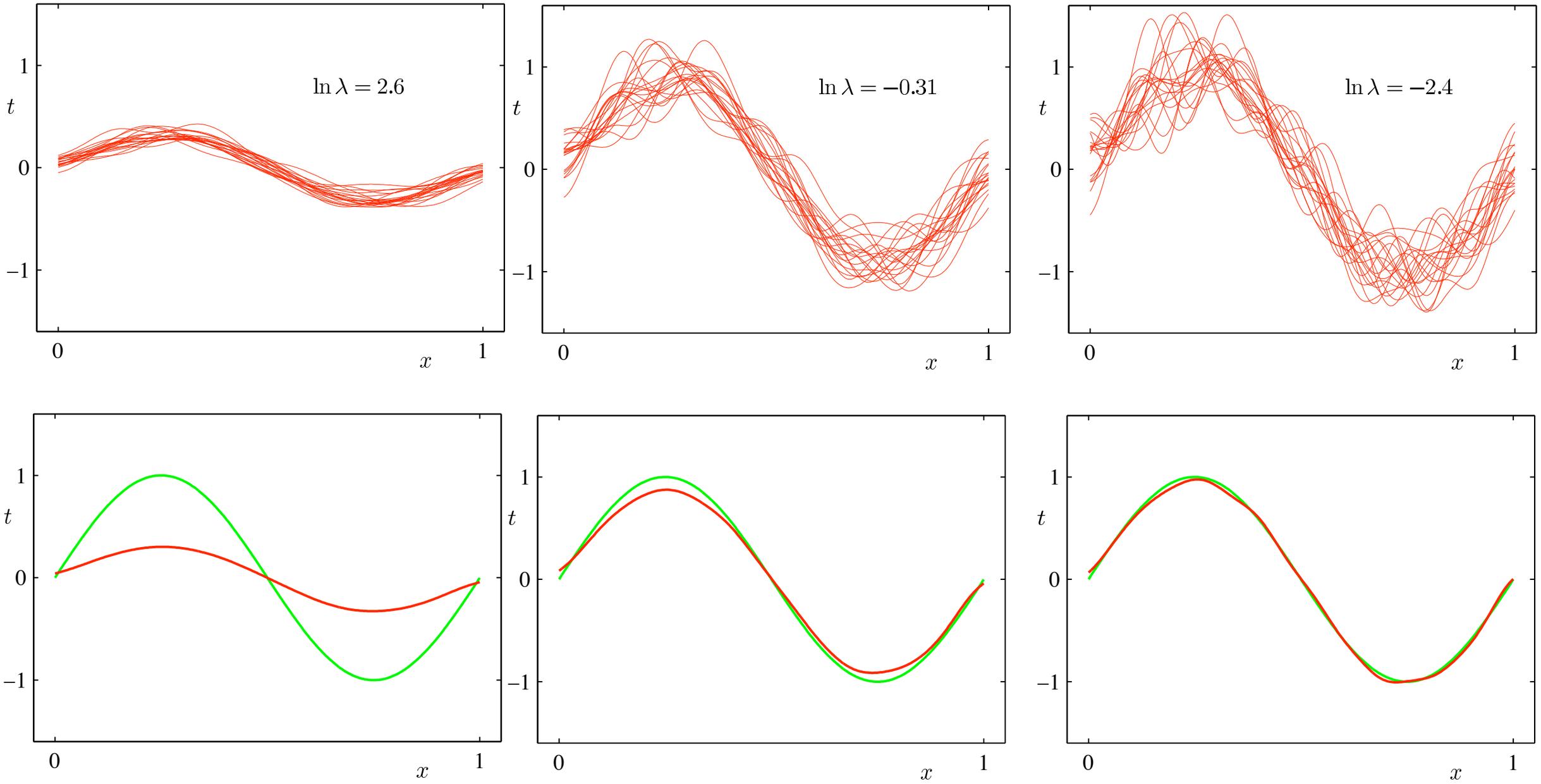
Bias-Variance decomposition

- expected loss = (bias)^2 + variance + noise where:

$$
\begin{aligned}
(\text{bias})^2 &= \int \{\mathbb{E}_{\mathcal{D}}[y(\mathbf{x};\mathcal{D})] - h(\mathbf{x})\}^2 p(\mathbf{x})\,\mathrm{d}\mathbf{x} \\
\text{variance} &= \int \mathbb{E}_{\mathcal{D}}\left[\{y(\mathbf{x};\mathcal{D}) - \mathbb{E}_{\mathcal{D}}[y(\mathbf{x};\mathcal{D})]\}^2\right] p(\mathbf{x})\,\mathrm{d}\mathbf{x} \\
\text{noise} &= \int \{h(\mathbf{x}) - t\}^2 p(\mathbf{x},\mathbf{t})\,\mathrm{d}\mathbf{x}\,\mathrm{d}\mathbf{t}
\end{aligned}
$$

- Very flexible models have low bias and high variance
- Relatively rigid models have high bias and low variance
- Optimal model balances the two

Bias variance example



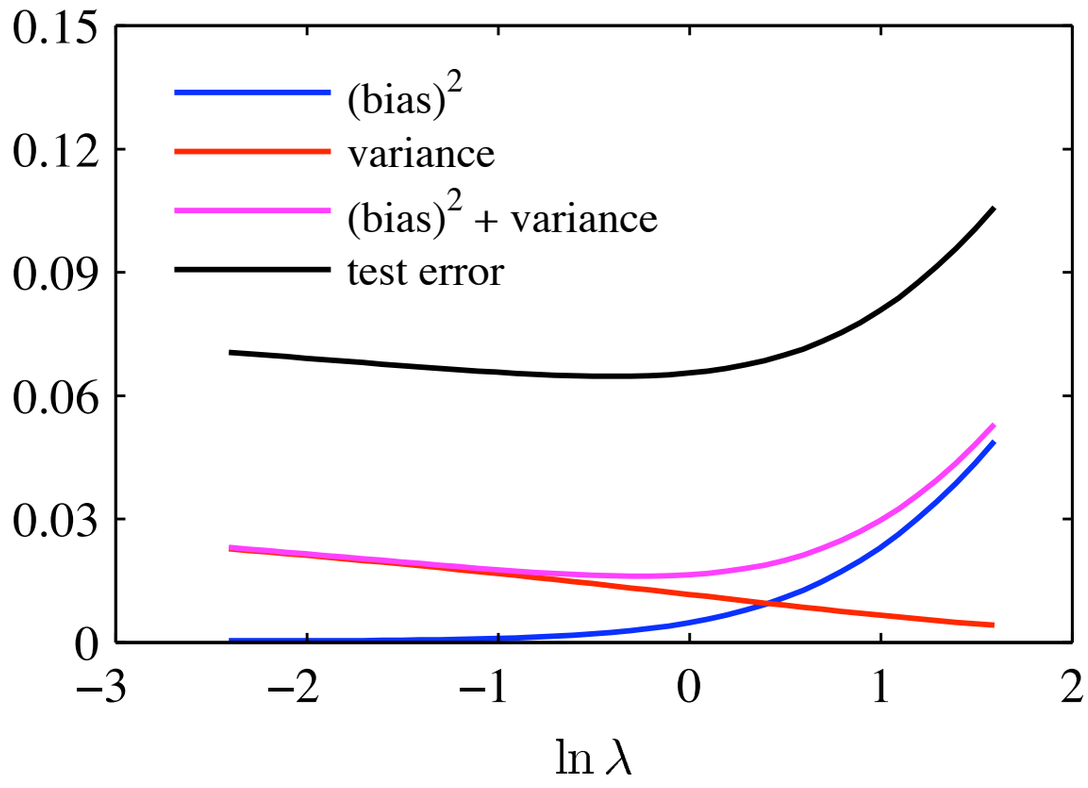
100 datasets each with 25 data points. Fit with 25 Gaussian basis functions.
Regularization parameter λ is varied. Top are individual fits.
Bottom is average fit along with generating sine function in green.

Bias variance example

$$\text{average} = \overline{y}(x) = \frac{1}{L}\sum_{l=1}^{L} y^{(l)}(x)$$

$$(\text{bias})^2 = \frac{1}{N}\sum_{n=1}^{N}\{\overline{y}(x_n) - h(x_n)\}^2$$

$$\text{variance} = \frac{1}{N}\sum_{n=1}^{N}\frac{1}{L}\sum_{l=1}^{L}\left\{y^{(l)}(x_n) - \overline{y}(x_n)\right\}^2$$



Plot of squared bias and variance together with their sum. The minimum is at λ=-0.31 which is close to the value yielding minimum test error

Bayesian Linear Regression

- Bias-variance decomposition requires splitting data. Inefficient.
- Avoids overfitting of maximum likelihood
- Leads to automatic way of determining model complexity
- Now we look quickly at Bayesian approach. Will return to it later in semester.

- Define prior over weights using zero-mean Gaussian prior:

$$p(\mathbf{w}|\alpha) = \mathcal{N}(\mathbf{w}|\mathbf{0}, \alpha^{-1}\mathbf{I})$$

- Log of posterior is sum of log likelihood and log of prior:

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

with quadratic regularization term $\lambda = \alpha/\beta$ in least squares sense

Sequential Bayesian Learning

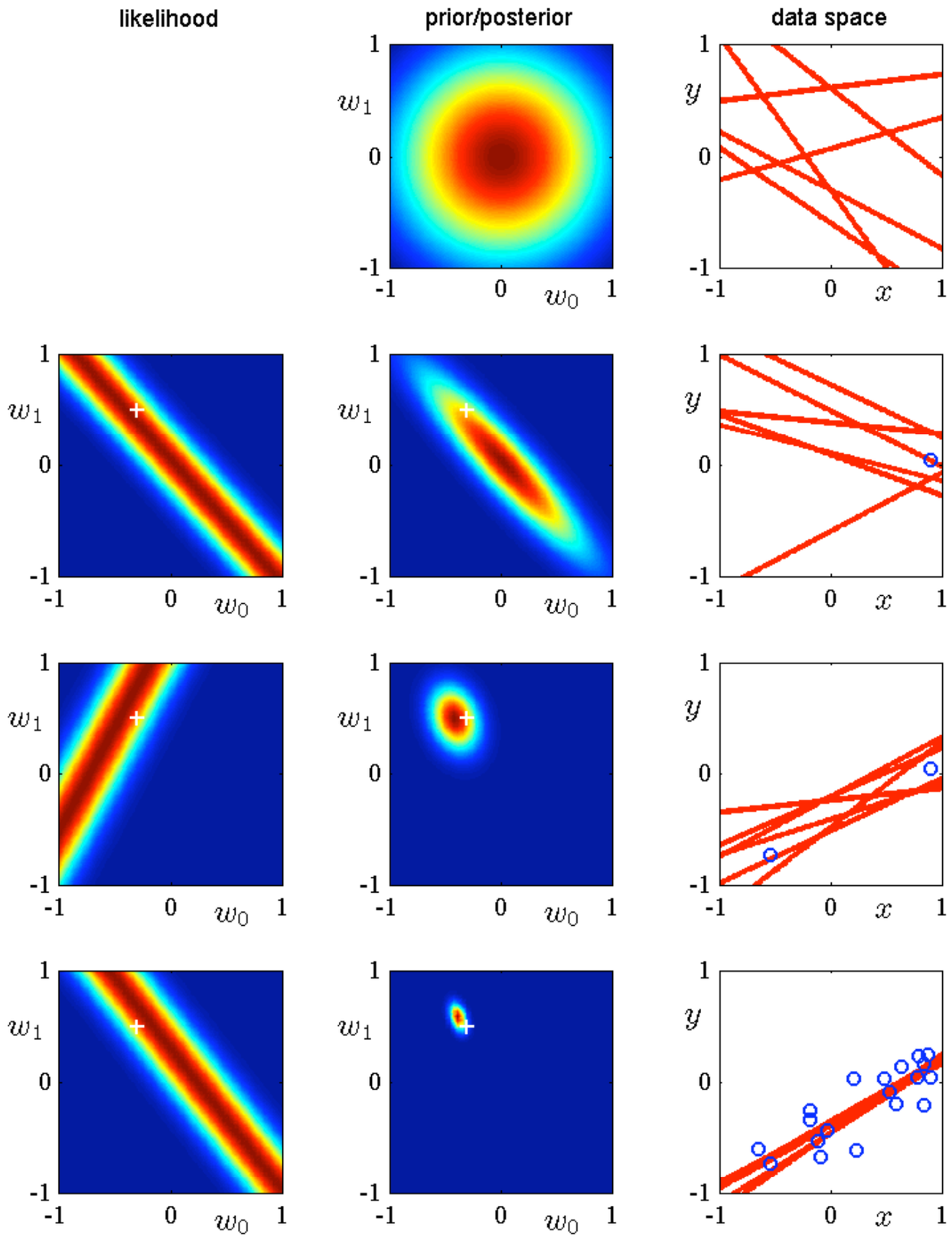
- Consider simple input variable *x*, single target *t* and a linear model of form $y(x, \mathbf{w}) = w_0 + w_1 x$
- Just two weights, can plot prior and posteriors
- Generate synthetic data using $f(x_n, \mathbf{a}) = -0.3 + 0.5 x_n + \epsilon$
- Goal is to recover **a** = {-0.3, 0.5} from data
- Basic algorithm:
  - observe point (*x*,*t*) from dataset
  - calculate likelihood p(*t*|*x*,**w**) based on estimate of noise precision β
  - multiply likelihood by previous prior over **w** to yield new posterior

Sequential Bayesian Learning

Basic algorithm:

- Observe point (*x*,*t*) from dataset
- Calculate likelihood p(*t*|*x*,**w**) based on estimate of noise variance β
- Multiply likelihood by previous prior over **w** to yield new posterior
- Observe another point ...

Samples from posterior are shown on right



Predictive distribution

- We are generally not interested in priors over **w** but rather for predicting new values *t* from x.
- Evaluate predictive distribution

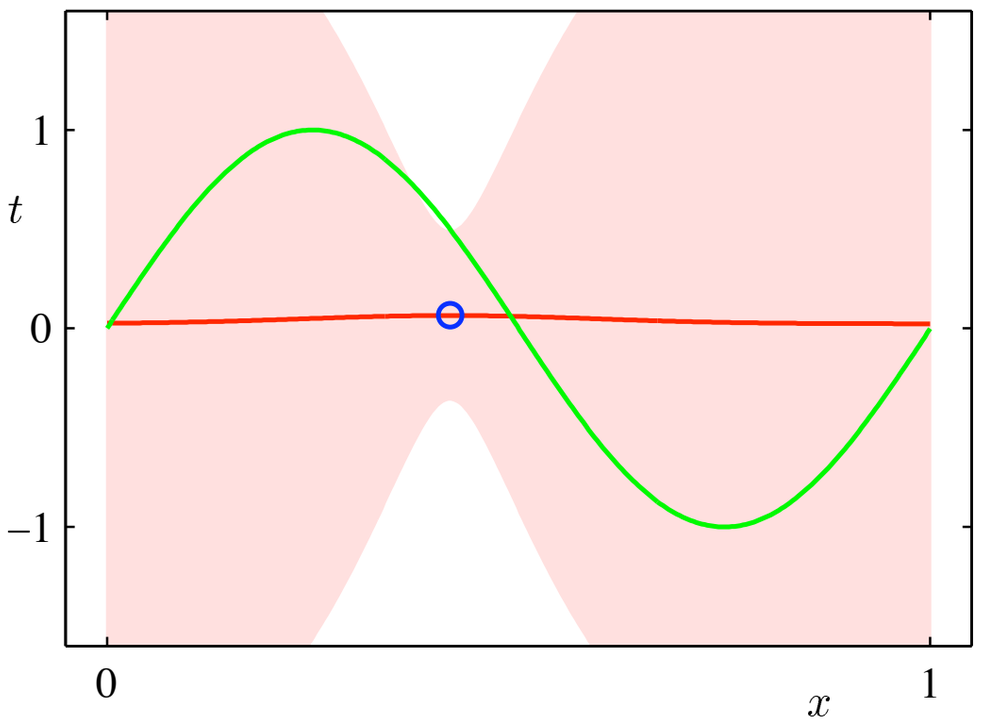
$$p(t|\mathbf{t}, \alpha, \beta) = \int p(t|\mathbf{w}, \beta) p(\mathbf{w}|\mathbf{t}, \alpha, \beta) \, \mathrm{d}\mathbf{w}$$

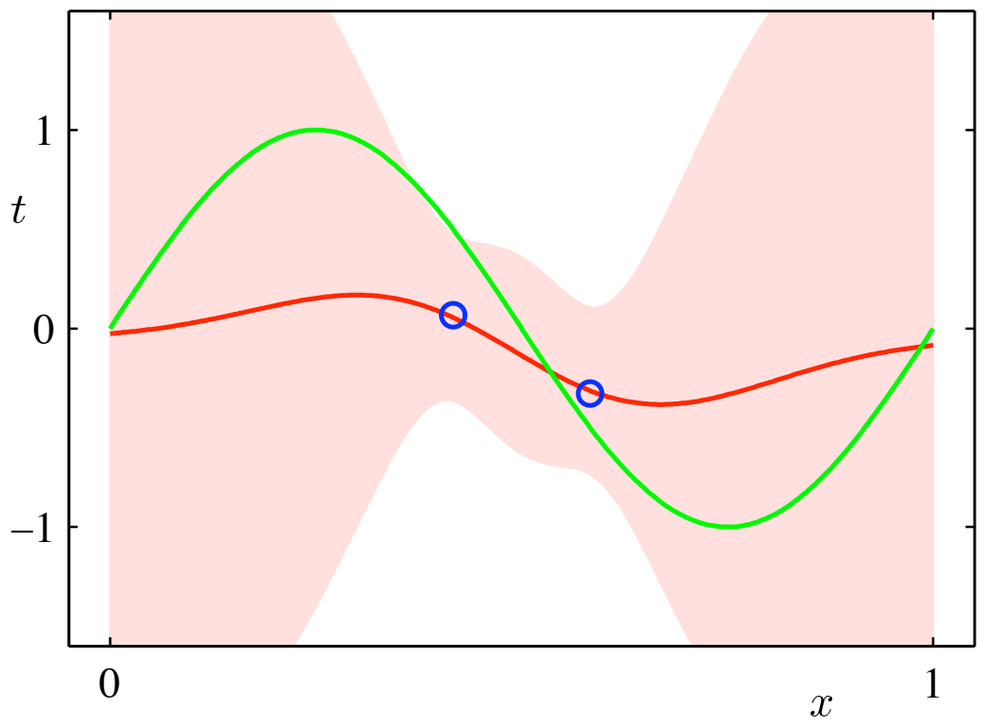
- This is convolution of conditional distribution of target with posterior **w**. For our problem (2 Gaussians) results in:

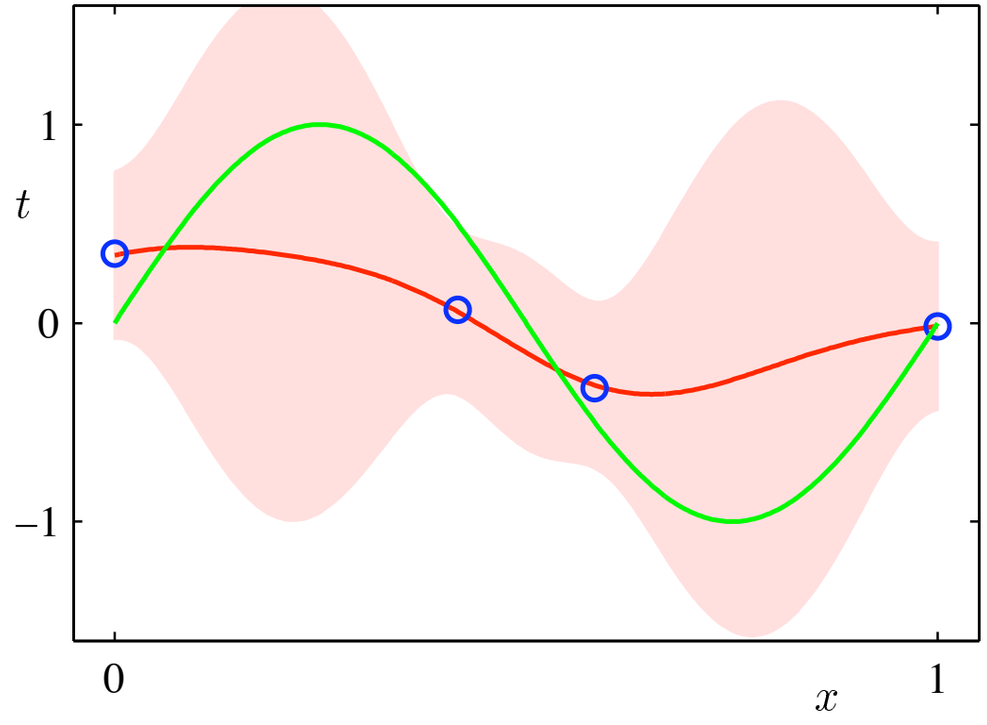
$$p(t|\mathbf{x}, \mathbf{t}, \alpha, \beta) = \mathcal{N}(t|\mathbf{m}_N^T \boldsymbol{\phi}(\mathbf{x}), \sigma_N^2(\mathbf{x}))$$

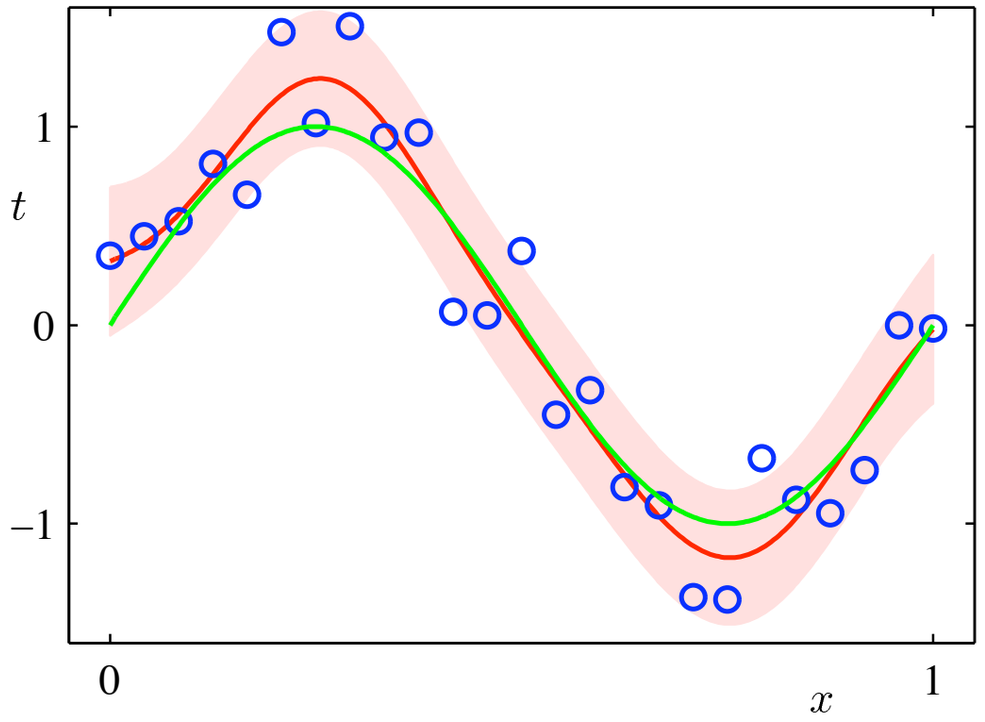
$$\sigma_N^2(\mathbf{x}) = \frac{1}{\beta} + \boldsymbol{\phi}(\mathbf{x})^T \mathbf{S}_N \boldsymbol{\phi}(\mathbf{x})$$

Predictive distribution



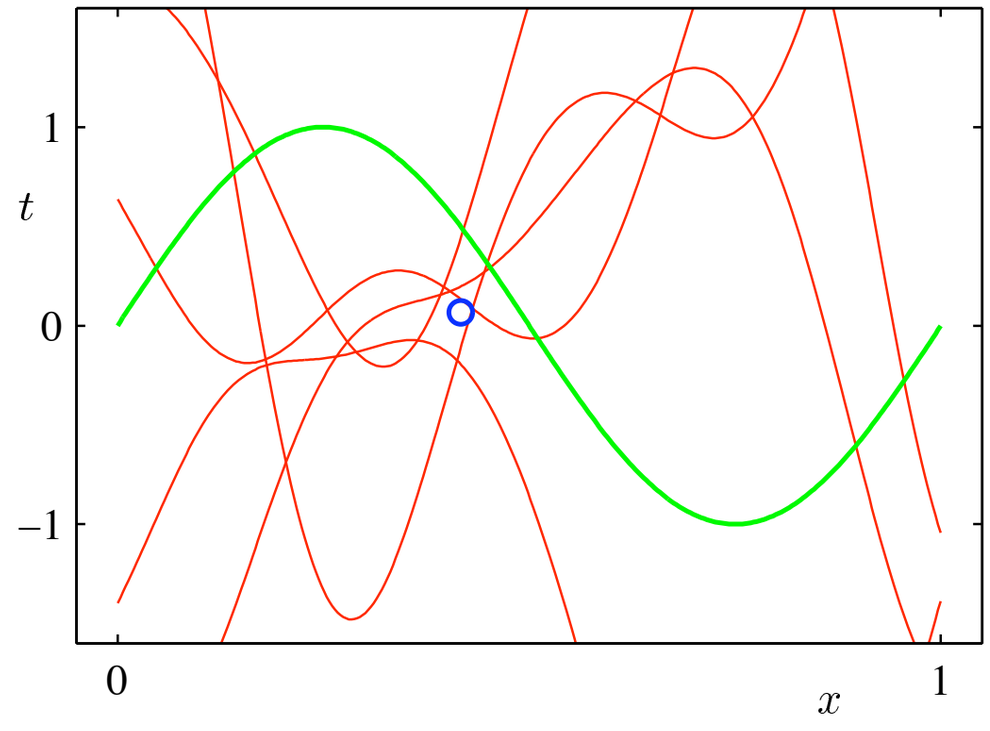


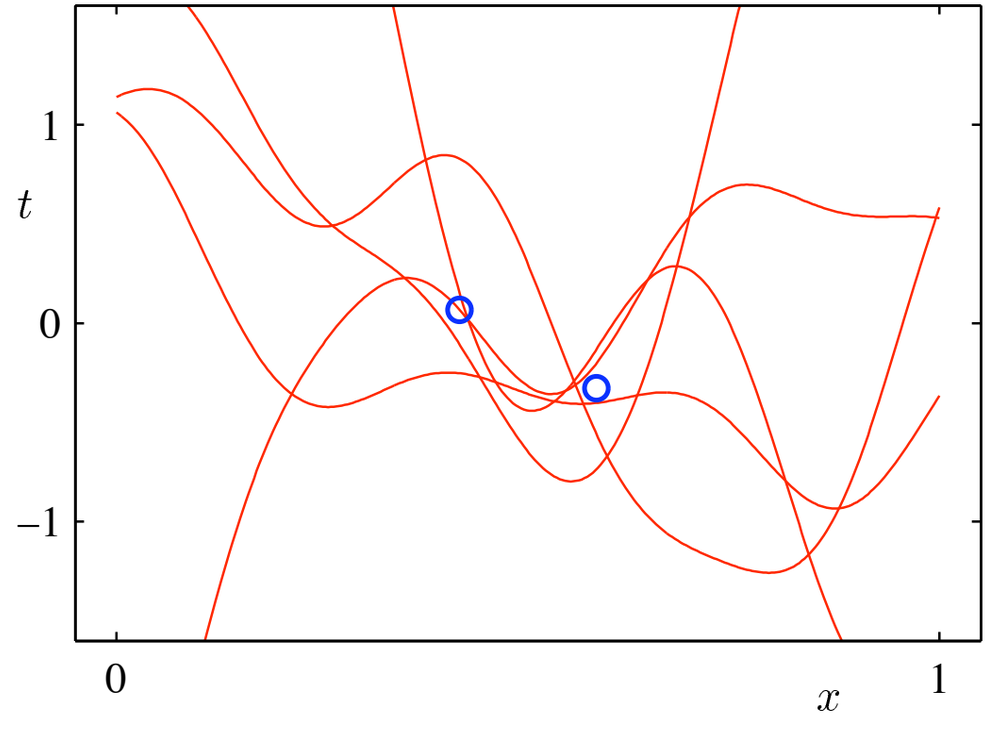


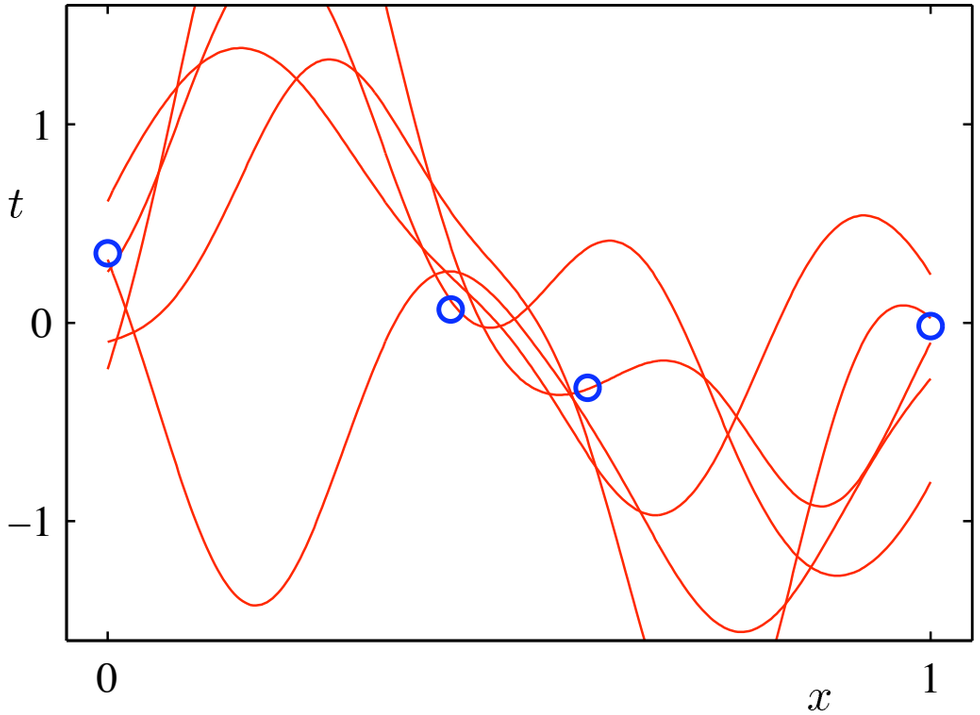


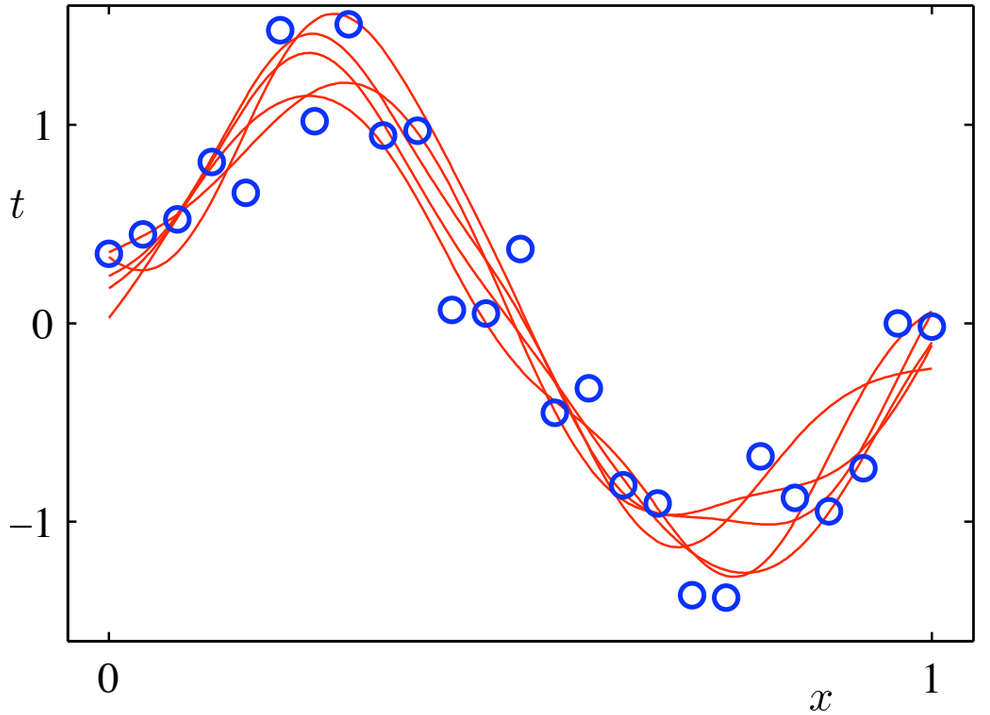
Predictive distributions for 9 Gaussian basis functions fitting f(x)=sin(2Πx)+ϵ in green. Red curve is mean of predictive distributions. Red shaded regions are 1 std dev. away from mean.

Predictive distribution









Plots of the functions y(x,**w**) using samples from the posterior distributions over **w** corresponding to the previous plots.

Equivalent kernel

- Posterior means can be interpreted as kernels; sets stage for kernel methods including Gaussian processes.

- Predictive mean can be written as:

$$y(\mathbf{x}, \mathbf{m}_N) = \mathbf{m}_N^{\mathrm{T}} \boldsymbol{\phi}(\mathbf{x}) = \beta \boldsymbol{\phi}(\mathbf{x})^{\mathrm{T}} \mathbf{S}_N \boldsymbol{\Phi}^{\mathrm{T}} \mathbf{t} = \sum_{n=1}^{N} \beta \boldsymbol{\phi}(\mathbf{x})^{\mathrm{T}} \mathbf{S}_N \boldsymbol{\phi}(\mathbf{x}) t_n$$

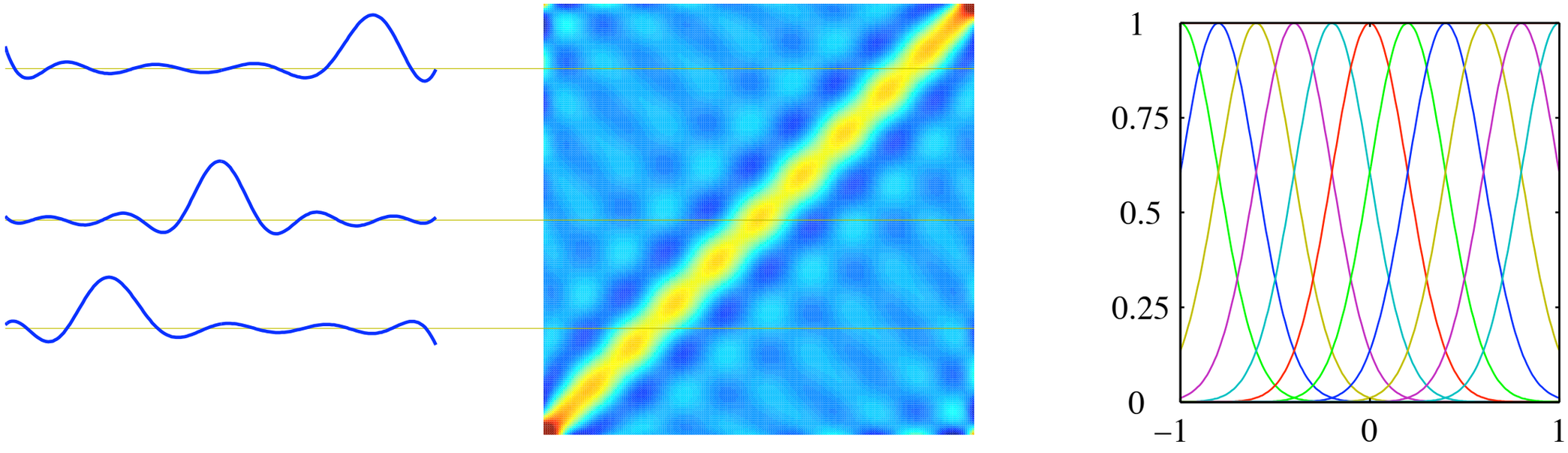
- We can also rewrite this as a kernel function:

$$y(\mathbf{x}, \mathbf{m}_N) = \sum_{n=1}^{N} k(\mathbf{x}, rbx_n) t_n$$

where the function $k(\mathbf{x}, \mathbf{x}') = \beta \boldsymbol{\phi}(\mathbf{x})^{\mathrm{T}} \mathbf{S}_N \boldsymbol{\phi}(\mathbf{x}')$ is known as the *smoother matrix* or *equivalent kernel*

- Regression functions which predict using linear combinations of target values are known as *linear smoothers*

Equivalent kernel



Equivalent kernel (left, middle) for Gaussian basis function (right)

- Above $k(x,x')$ is plotted as a function of x. Note that it is localized around x.
- Mean of predictive distribution at x given by $y(x,\mathbf{m}_N)$ is obtained using a weighted combination where points close to x are given higher weight.
- Idea of using a localized kernel in place of a set of basis functions leads to *Gaussian processes* (to be covered later).