Lecture 4

- Homework
  - Hw 1 and 2 will be reoped after class for every body. New deadline 4/20
  - Hw 3 and 4 online (Nima is lead)
  - Python.

- Pod-cast lecture on-line

- Final projects
  - Nima will register groups next week. Email/tell Nima.
  - Give proposal in week 5
  - See last years topic on webpage. Choose your own or proposed topics/Kaggle

- Linear regression 3.0-3.3+ SVD

- Next lectures:
  - I posted a rough plan.
  - It is flexible though so please come with suggestions

Projects

- **3-4** person groups
- Deliverables: Poster & Report & main code (plus proposal, midterm slide)
- Topics your own or chose form suggested topics
- **Week 3 groups** due to TA Nima (if you don't have a group, ask in week 2 and we can help).
- **Week 5** proposal due. TAs and Peter can approve.
- Proposal: One page: Title, A large paragraph, data, weblinks, references.
- Something physical
- **Week ~7** Midterm slides? Likely presented to a subgroup of class.
- **Week 10/11 (likely 5pm 6 June** Jacobs Hall lobby**)** final poster session?
- Report due Saturday 16 June.

Mark's Probability and Data homework

Hi

Bayes Rule

$$P(A|B) = \frac{P(B|A)\,P(A)}{P(B)}$$

General Terminology

$P(A|B)$, $P(B|A)$ ← conditional

likelihood

Bayesian Terminology

$P(A|B)$ 'posterior probability'

$P(A)$ 'prior probability'

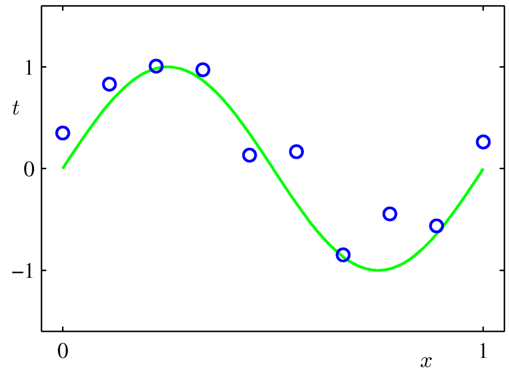
$P(B)$ 'evidence'

Linear regression: Linear Basis Function Models (1)

Generally

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

- where $\phi_j(x)$ are known as *basis functions*.
- Typically, $\phi_0(x) = 1$, so that w_0 acts as a bias.
- Simplest case is linear basis functions: $\phi_d(x) = x_d$.



$$y(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \ldots + w_M x^M = \sum_{j=0}^{M} w_j x^j$$

http://playground.tensorflow.org/

Maximum Likelihood and Least Squares (3)

Computing the gradient and setting it to zero yields

$$\nabla_{\mathbf{w}} \ln p(\mathbf{t}|\mathbf{w}, \beta) = \beta \sum_{n=1}^{N} \left\{ t_n - \mathbf{w}^{\mathrm{T}} \boldsymbol{\phi}(\mathbf{x}_n) \right\} \boldsymbol{\phi}(\mathbf{x}_n)^{\mathrm{T}} = \mathbf{0}.$$

Solving for w,

$$\mathbf{w}_{\mathrm{ML}} = \overbrace{\left(\mathbf{\Phi}^{\mathrm{T}} \mathbf{\Phi} \right)^{-1} \mathbf{\Phi}^{\mathrm{T}}}^{\text{The Moore-Penrose pseudo-inverse, } \mathbf{\Phi}^{\dagger}.} \mathbf{t}$$

where

$$\mathbf{\Phi} = \begin{array}{cc} & M \\ N & \begin{pmatrix} \phi_0(\mathbf{x}_1) & \phi_1(\mathbf{x}_1) & \cdots & \phi_{M-1}(\mathbf{x}_1) \\ \phi_0(\mathbf{x}_2) & \phi_1(\mathbf{x}_2) & \cdots & \phi_{M-1}(\mathbf{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_0(\mathbf{x}_N) & \phi_1(\mathbf{x}_N) & \cdots & \phi_{M-1}(\mathbf{x}_N) \end{pmatrix} \end{array} = \begin{pmatrix} -\phi_1^{\mathrm{T}}- \\ \phi_2^{\mathrm{T}} \\ -\phi_3^{\mathrm{T}}- \\ -\phi_n^{\mathrm{T}}- \end{pmatrix}$$

Least mean squares: An alternative approach for big datasets

$$\mathbf{w}^{\tau+1} = \mathbf{w}^{\tau} - \eta \boxed{\nabla E_{n(\tau)}} \quad ✓$$

weights after seeing training case tau+1

learning rate ✓

squared error derivatives w.r.t. the weights on the training case at time tau.

This is **"on-line" learning**. It is efficient if the dataset is redundant and simple to implement.

- It is called **stochastic gradient descent** if the training cases are picked randomly.
- Care must be taken with the learning rate to prevent divergent oscillations. Rate must decrease with tau to get a good fit.

$$\frac{\partial E_n}{\partial w} = \cancel{\sum_{n}^{N}} \phi_n (t_n - w^T \phi_n)$$

Regularized least squares

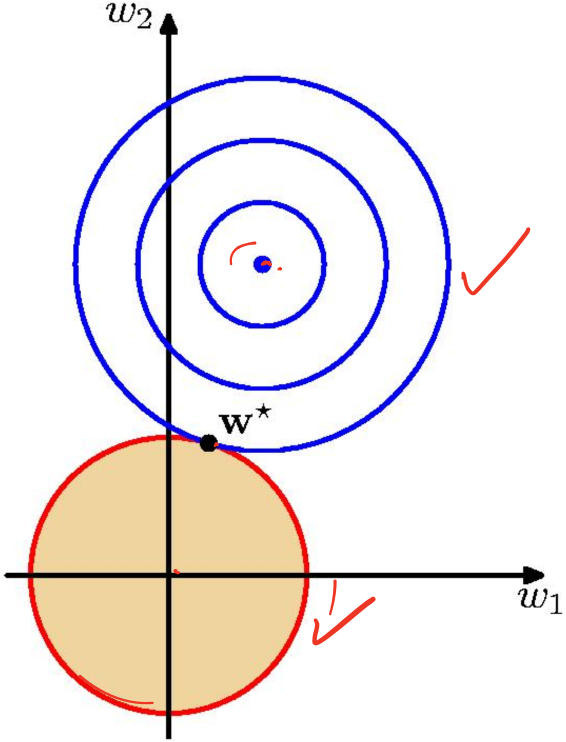
$$\widetilde{E}(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{y(\mathbf{x}_n, \mathbf{w}) - t_n\}^2 + \frac{\lambda}{2}\|\mathbf{w}\|^2$$

The squared weights penalty is mathematically compatible with the squared error function, giving a closed form for the optimal weights:

$$\mathbf{w}^* = (\lambda\,\mathbf{I} + \mathbf{X}^T\mathbf{X})^{-1}\,\mathbf{X}^T\,\mathbf{t}$$

identity matrix

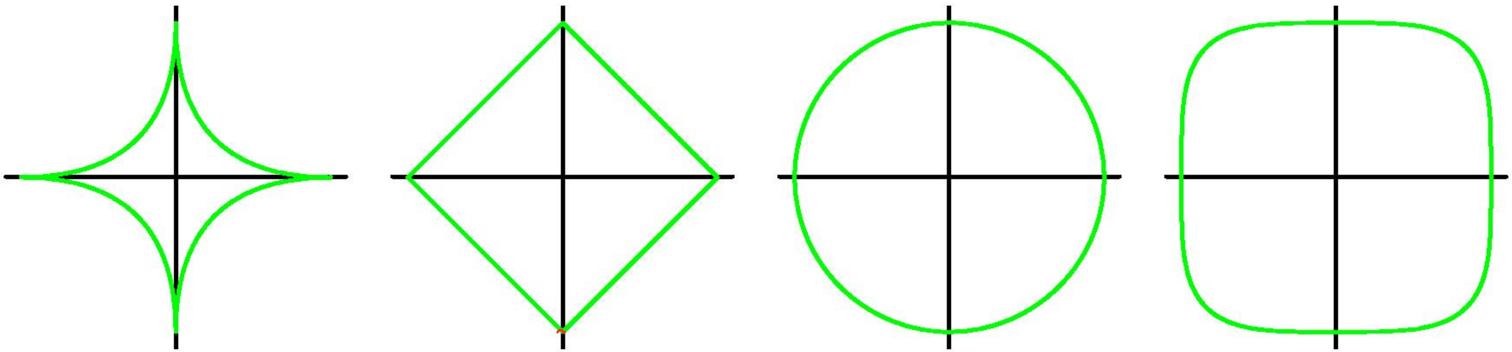
A picture of the effect of the regularizer



- The overall cost function is the sum of two parabolic bowls.
- The sum is also a parabolic bowl.
- The combined minimum lies on the line between the minimum of the squared error and the origin.
- The L2 regularizer just **shrinks** the weights.

Other regularizers $|w|^q$

- We do not need to use the squared error, provided we are willing to do more computation.
- Other powers of the weights can be used.



$q = 0.5$ $q = 1$ $q = 2$ $q = 4$

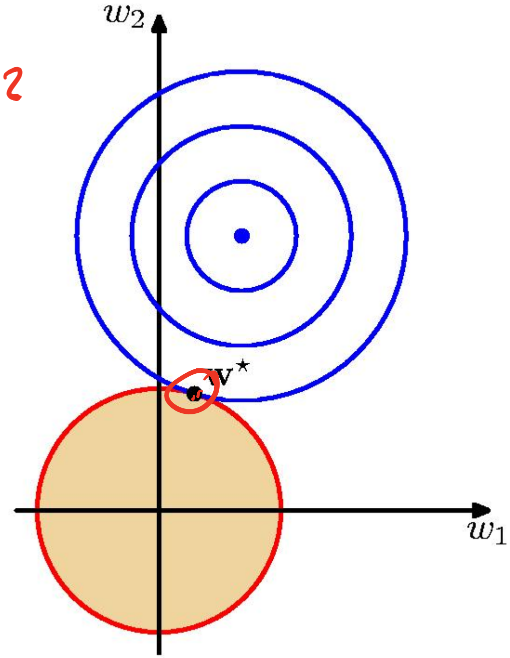
The lasso: penalizing the absolute values of the weights

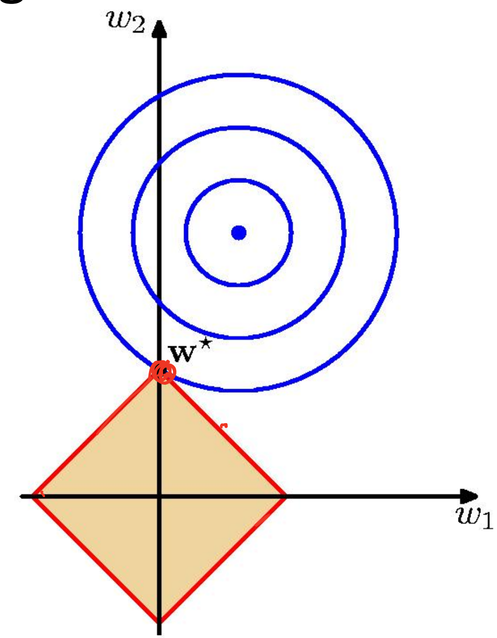
$$\widetilde{E}(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{y(\mathbf{x}_n, \mathbf{w}) - t_n\}^2 \; + \; \lambda \sum_i |\, \mathbf{w}_i \,|$$

- Finding the minimum requires quadratic programming but its still unique because the cost function is convex (a bowl plus an inverted pyramid)
- As lambda increases, many weights go to exactly zero.
  - This is great for interpretation, and it is also prevents overfitting.

Geometrical view of the lasso compared with a penalty on the squared weights

ℓ_2





Notice **w1=0** at the optimum

Minimizing the absolute error

$$\min_{over\ \mathbf{w}} \sum_n | t_n - \mathbf{w}^T \mathbf{x}_n |$$

- This minimization involves solving a linear programming problem.
- It corresponds to maximum likelihood estimation if the output noise is modeled by a Laplacian instead of a Gaussian.

$e^{-\frac{1}{2}(t_n - y_n)^2}$

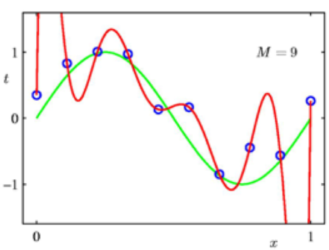
$$p(t_n \mid y_n) = a\, e^{-a\,|t_n - y_n|}$$

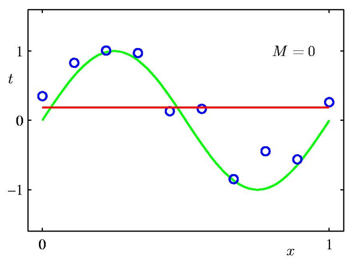
$$-\log p(t_n \mid y_n) = -a\,| t_n - y_n | + const$$

The bias-variance trade-off
(a figment of the frequentists lack of imagination?)

- Imagine a training set drawn at random from a whole set of training sets.
- The squared loss can be decomposed into a
  - Bias = systematic error in the model's estimates
  - Variance = noise in the estimates cause by sampling noise in the training set.
- There is also additional loss due to noisy target values.
  - We eliminate this extra, irreducible loss from the math by using the average target values (i.e. the unknown, noise-free values)

9 Order Polynomial





The bias-variance decomposition

model estimate for testcase n trained on dataset D

average target value for test case n

“Bias” term is the squared error of the average, over training datasets D, of the estimates.

Bias: average between prediction and desired.

$$\left\langle \left\{ y(\mathbf{x}_n; D) - \bar{t}_n \right\}^2 \right\rangle_D = \boxed{\left\{ \left\langle y(\mathbf{x}_n; D) \right\rangle_D - \bar{t}_n \right\}^2} + \boxed{\left\langle \left\{ y(\mathbf{x}_n; D) - < y(\mathbf{x}_n; D) >_D \right\}^2 \right\rangle_D}$$

<. > means expectation over D

$y(x_n; D) - \langle y \rangle_D + \langle y \rangle_D - t$

“Variance” term: variance over training datasets D, of the model estimate.

Regularization parameter affects the bias and variance terms

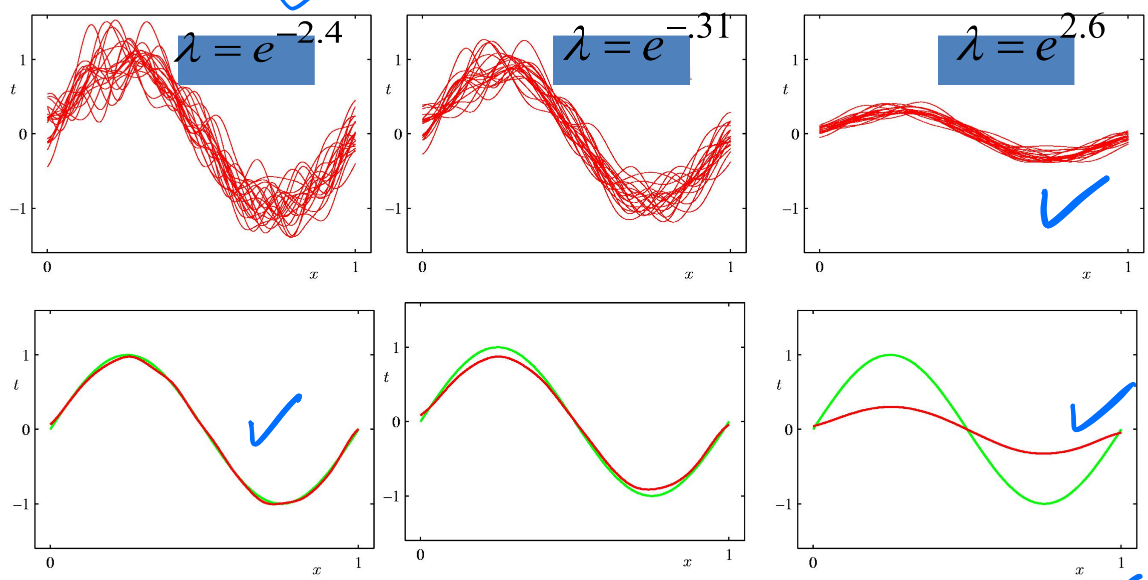
high variance

low variance

20 realizations

True model

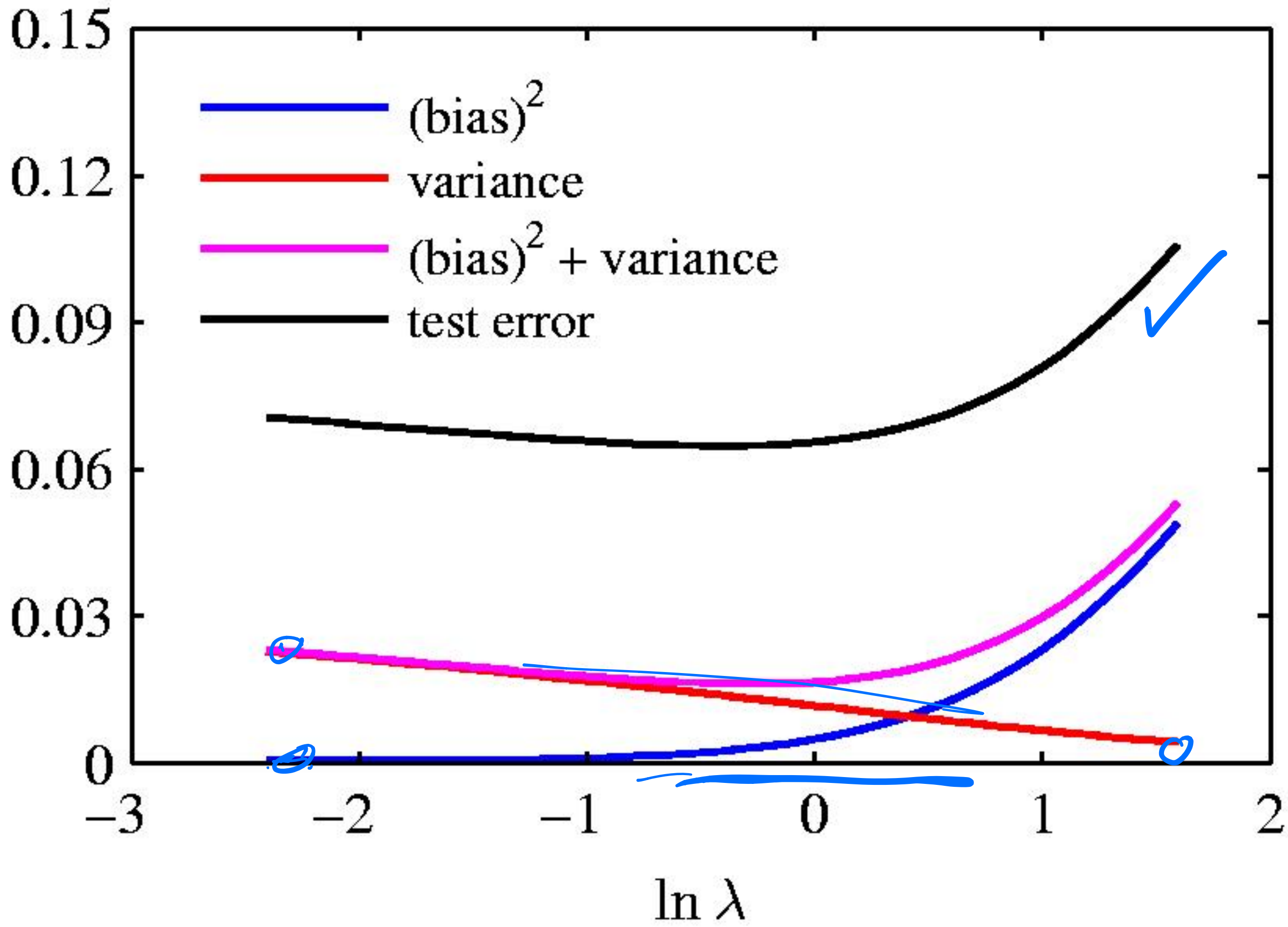
average



high bias

low bias

An example of the bias-variance trade-off



Beating the bias-variance trade-off

- Reduce the variance term by averaging lots of models trained on different datasets.
  - Seems silly. For lots of different datasets it is better to combine them into one big training set.
    - More training data has much less variance.
- Weird idea: We can create different datasets by bootstrap sampling of our single training dataset.
  - This is called "bagging" and it works surprisingly well.
- If we have enough computation its better doing it Bayesian:
  - Combine the predictions of many models using the posterior probability of each parameter vector as the combination weight.

Bayesian Linear Regression (1)

Define a conjugate prior over w

$$p(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mathbf{m}_0, \mathbf{S}_0).$$

- Combining this with the likelihood function and using results for multiplying Gaussians, gives the posterior

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

$$= p(t|w)\, p(w)$$

$$\mathbf{m}_N = \mathbf{S}_N \left(\mathbf{S}_0^{-1}\mathbf{m}_0 + \beta\mathbf{\Phi}^{\mathrm{T}}\mathbf{t}\right)$$

$$\mathbf{S}_N^{-1} = \mathbf{S}_0^{-1} + \beta\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi}.$$

- A common simpler prior

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

- Which gives

$$\mathbf{m}_N = \beta\mathbf{S}_N\mathbf{\Phi}^{\mathrm{T}}\mathbf{t}$$

$$\mathbf{S}_N^{-1} = \alpha\mathbf{I} + \beta\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi}.$$

From lecture 3:

Bayes for linear model

$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{n}$ $\quad$ $\boldsymbol{n} \sim \mathrm{N}(\boldsymbol{0}, \boldsymbol{C_n})$ $\quad$ $\mathbf{y} \sim \mathrm{N}(\boldsymbol{A}\boldsymbol{x}, \boldsymbol{C_n})$ $\quad$ prior: $\boldsymbol{x} \sim \mathrm{N}(\boldsymbol{0}, \boldsymbol{C_x})$

$p(\boldsymbol{x}|\boldsymbol{y}) \sim p(\boldsymbol{y}|\boldsymbol{x})p(\boldsymbol{x}) \sim N\left(\boldsymbol{x_p}, \boldsymbol{C}_p\right)$ $\quad$ mean $\quad$ $\boldsymbol{x_p} = \boldsymbol{C}_p \boldsymbol{A}^T \boldsymbol{C}_n^{-1} \boldsymbol{y}$

Covariance $\quad$ $\boldsymbol{C}_p^{-1} = \boldsymbol{A}^T \boldsymbol{C}_n^{-1} \boldsymbol{A} + \boldsymbol{C}_x^{-1}$

$$\sim e^{-\frac{1}{2}(x-x_p)^T C_p^{-1}(x-x_p)} \leftarrow$$

$$= e^{-\frac{1}{2}(y-Ax)^T C_n^{-1}(y-Ax)}\, e^{-\frac{1}{2}x^T C_x^{-1} x}$$

$$= e^{-\frac{1}{2}\underbrace{(x^T A^T C_n^{-1} A x + x^T C_x^{-1} x)}_{x^T C_p^{-1} x^T}}\, e^{-\frac{1}{2}\underbrace{x^T A^T C_n^{-1} y}_{x^T C_p^{-1} x_p}}$$

$$C_p^{-1} = A^T C_n^{-1} A + C_x^{-1}$$

$$x_p = C_p A^T C_n^{-1} y$$

Interpretation of solution

M features

$$\Phi = \begin{bmatrix} -\phi_1^T- \\ -\phi_2^T- \\ \vdots \\ -\phi_N^T- \end{bmatrix} \quad N \text{ obs}$$

$$\mathbf{m}_N = \beta \mathbf{S}_N \mathbf{\Phi}^{\mathrm{T}} \mathbf{t}$$

$$\mathbf{S}_N^{-1} = \alpha \mathbf{I} + \beta \mathbf{\Phi}^{\mathrm{T}} \mathbf{\Phi}. = \alpha I + \beta \sum_{n}^{N} \phi_n \phi_n^T$$

Draw it

Sequential, **conjugate prior**

$$p(w|t_0) = p(w)$$

$$p(w|t_1) = p(t_1|w)\, p(w|t_0)$$

$$p(w|t_2) = p(t_2|w)\, p(w|t_1)$$

$$S_1^{-1} = \alpha I + \beta \phi_1 \phi_1^T$$

$$S_2^{-1} = \alpha I + \beta(\phi_1 \phi_1^T + \phi_2 \phi_2^T)$$

$$p(\boldsymbol{x}|\boldsymbol{y}) \sim p(\boldsymbol{y}|\boldsymbol{x})\, p(\boldsymbol{x}) \sim \mathrm{N}(\boldsymbol{A}\boldsymbol{x}, \boldsymbol{C}_n)\ \mathrm{N}(\boldsymbol{0}, \boldsymbol{C}_x) \sim N(\boldsymbol{x}_p, \boldsymbol{C}_p)$$

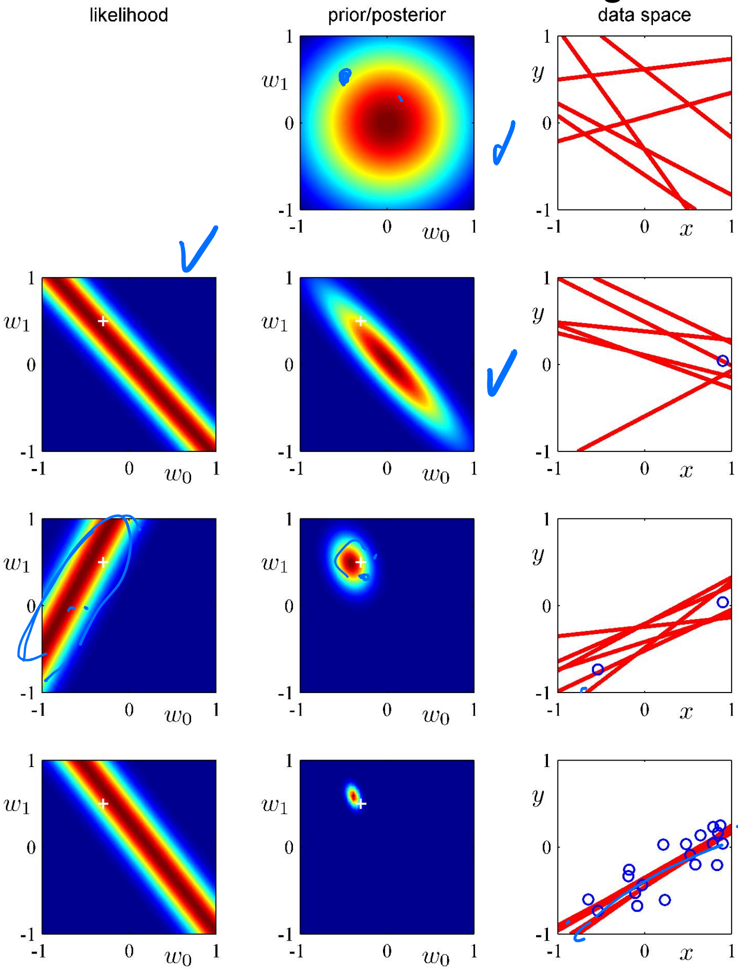
Covariance $\boldsymbol{C}_p^{-1} = \boldsymbol{A}^T \boldsymbol{C}_n^{-1} \boldsymbol{A} + \boldsymbol{C}_x^{-1}$

$$S_2^{-1} = \phi_2 (\beta I) \phi_2^T + S_1^{-1} = \alpha I + \beta(\phi_1 \phi_1^T + \phi_2 \phi_2^T)$$

Likelihood, prior/posterior Bishop Fig 3.7

$$y = w_0 + w_1 x + N(0,0.2)$$

Data generated with. w_0=-0.3, w_1=0.5

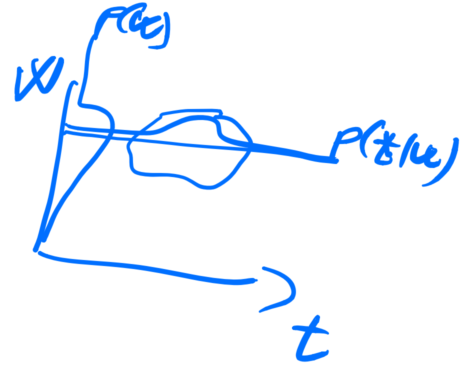


With no data we sample lines from the prior.

With 20 data points, the prior has little effect

Predictive distributions

$p(t, w | T) = p(t | w, T)\, p(w | T)$



$p(t | T)$

- marginal

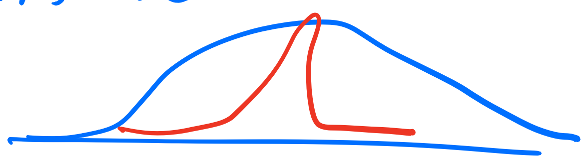
$p(t | T) = \int p(t, w)\, dw$

$= \int p(t | w, T)\, p(w | T)\, dw$

$p(t | T, \beta, \alpha) = \int p(t | w, \beta)\, p(w | T, \alpha, \beta)\, dw$

- Prior predictive

$\underline{p(t | \hat{w}_{ML}, \beta)}$



to optimstic

Predictive Distribution

Predict t for new values of x by integrating over w (Giving the marginal distribution of t):

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

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

training data

precision of prior

precision of output noise

$$\mathbf{m}_N = \beta \mathbf{S}_N \boldsymbol{\Phi}^{\mathrm{T}} \mathbf{t}$$

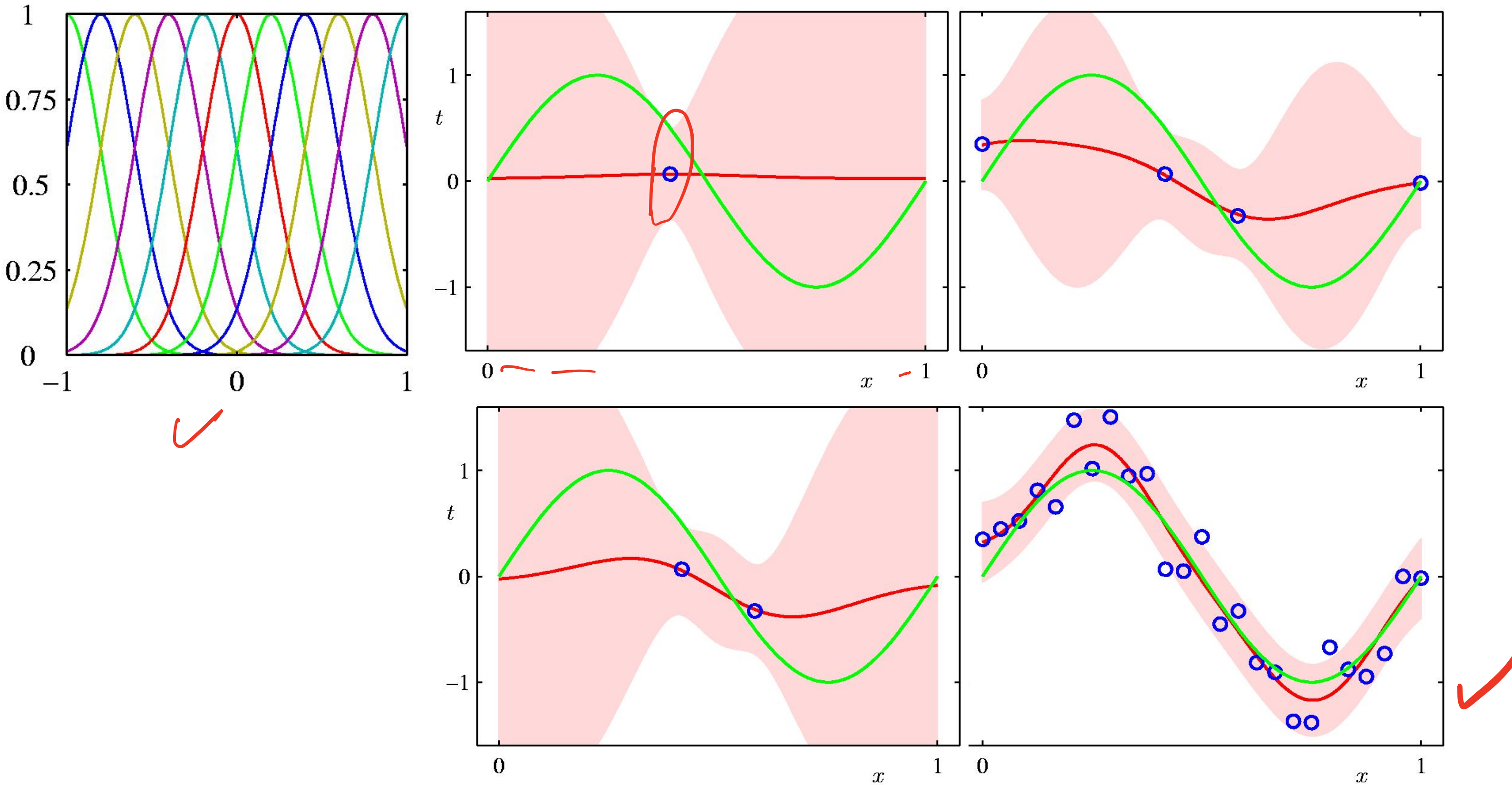
$$\mathbf{S}_N^{-1} = \alpha \mathbf{I} + \beta \boldsymbol{\Phi}^{\mathrm{T}} \boldsymbol{\Phi}.$$

- where

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

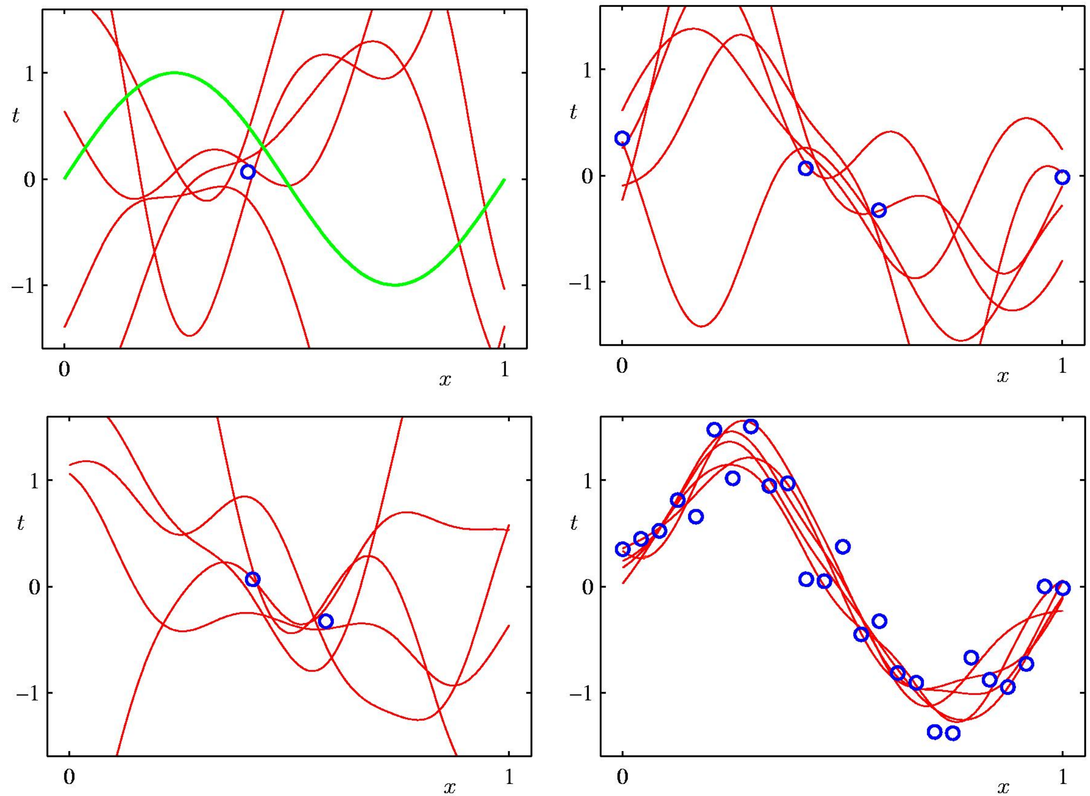
0

Predictive distribution for noisy sinusoidal data modeled by linear combining 9 radial basis functions.



A way to see the covariance of predictions for different values of x

We sample models at random from the posterior and *show the mean* of each model's predictions



Equivalent Kernel BISHOP 3.3.3

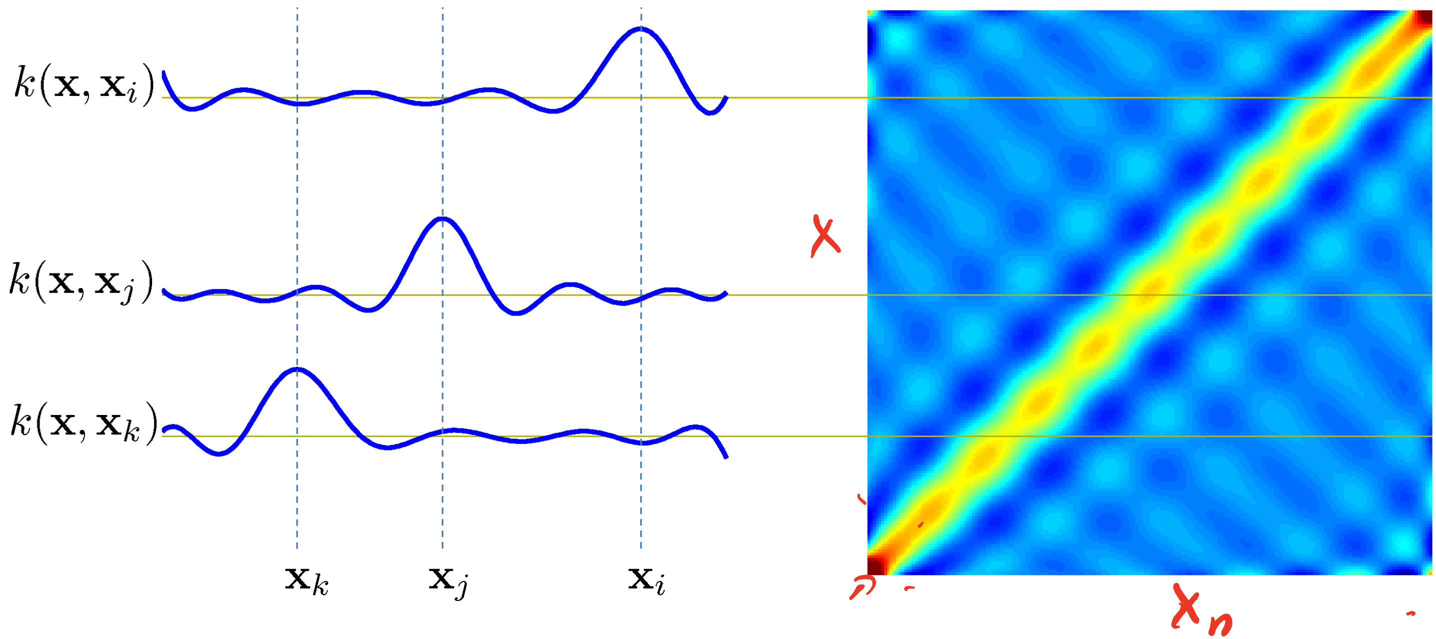
The predictive mean can be written

$$
\begin{aligned}
y(\mathbf{x}, \mathbf{m}_N) &= \mathbf{m}_N^{\mathrm{T}} \boldsymbol{\phi}(\mathbf{x}) = \beta \boldsymbol{\phi}(\mathbf{x})^{\mathrm{T}} \mathbf{S}_N \boldsymbol{\Phi}^{\mathrm{T}} \mathbf{t} \\
&= \sum_{n=1}^{N} \beta \boldsymbol{\phi}(\mathbf{x})^{\mathrm{T}} \mathbf{S}_N \boldsymbol{\phi}(\mathbf{x}_n) t_n \\
&= \sum_{n=1}^{N} k(\mathbf{x}, \mathbf{x}_n) t_n.
\end{aligned}
$$

Equivalent kernel or *smoother matrix*.

This is a weighted sum of the training data target values, t_n.

Equivalent Kernel (2)



Weight of t_n depends on distance between x and x_n; nearby x_n carry more weight.

Equivalent Kernel (4)

- The kernel as a covariance function: consider

$$\begin{aligned} \text{cov}[y(\mathbf{x}), y(\mathbf{x}')] &= \text{cov}[\boldsymbol{\phi}(\mathbf{x})^{\mathrm{T}}\mathbf{w}, \mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}')] \\ &= \boldsymbol{\phi}(\mathbf{x})^{\mathrm{T}}\mathbf{S}_N\boldsymbol{\phi}(\mathbf{x}') = \beta^{-1}k(\mathbf{x}, \mathbf{x}'). \end{aligned}$$

- We can avoid the use of basis functions and define the kernel function directly, leading to *Gaussian Processes* (Chapter 6).
- No need to determine weights.

- Like all kernel functions, the equivalent kernel can be expressed as an inner product:

$$k(\mathbf{x}, \mathbf{z}) = \boldsymbol{\psi}(\mathbf{x})^{\mathrm{T}}\boldsymbol{\psi}(\mathbf{z})$$

$$\boldsymbol{\psi}(\mathbf{x}) = \beta^{1/2}\mathbf{S}_N^{1/2}\boldsymbol{\phi}(\mathbf{x})$$