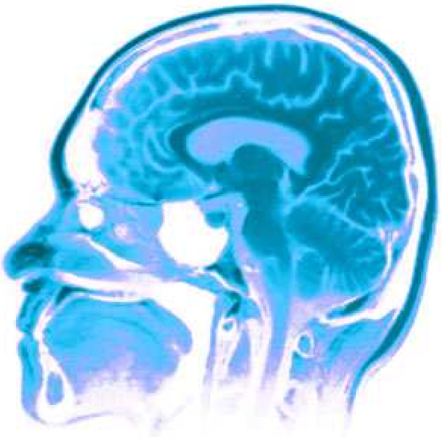
CPSC540



Regularization, nonlinear prediction and generalization

Nando de Freitas
Janurary, 2013
University of British Columbia

Outline of the lecture

This lecture will teach you how to fit nonlinear functions by using bases functions and how to control model complexity. The goal is for you to:

❑ Learn how to derive **ridge regression**.
❑ Understand the trade-off of fitting the data and **regularizing** it.
❑ Learn **polynomial regression**.
❑ Understand that, if basis functions are given, the problem of learning the parameters is still linear.
❑Learn **cross-validation.**
❑ Understand the effects of the number of data and the number of basis functions on **generalization**.

Regularization

All the answers so far are of the form

$$\underset{d\times 1}{\widehat{\boldsymbol{\theta}}} = (\underset{d\times n}{\mathbf{X}^T}\underset{n\times d}{\mathbf{X}})^{-1}\underset{d\times n}{\mathbf{X}^T}\underset{n\times 1}{\mathbf{y}}$$

They require the inversion of $\mathbf{X}^T\mathbf{X}$. This can lead to problems if the system of equations is poorly conditioned. A solution is to add a small element to the diagonal:

$$\widehat{\boldsymbol{\theta}}_R = (\mathbf{X}^T\mathbf{X} + \underline{\delta^2 I_d})^{-1}\mathbf{X}^T\mathbf{y}$$

This is the ridge regression estimate. It is the solution to the following **regularised quadratic cost function**

$$J(\boldsymbol{\theta}) = (\mathbf{y} - \mathbf{X}\boldsymbol{\theta})^T(\mathbf{y} - \mathbf{X}\boldsymbol{\theta}) + \underbrace{\overset{\text{scalar}}{\delta^2}\boldsymbol{\theta}^T\boldsymbol{\theta}}_{\substack{\text{penalty}\\ \text{regularizer}}}$$

Derivation

$$J(\theta) = (Y - X\theta)^T (Y - X\theta) + \delta^2 \theta^T \theta$$

$$\frac{\partial J(\theta)}{\partial \theta} = 2X^T X\theta - 2X^T y + 2\delta^2 I\theta$$

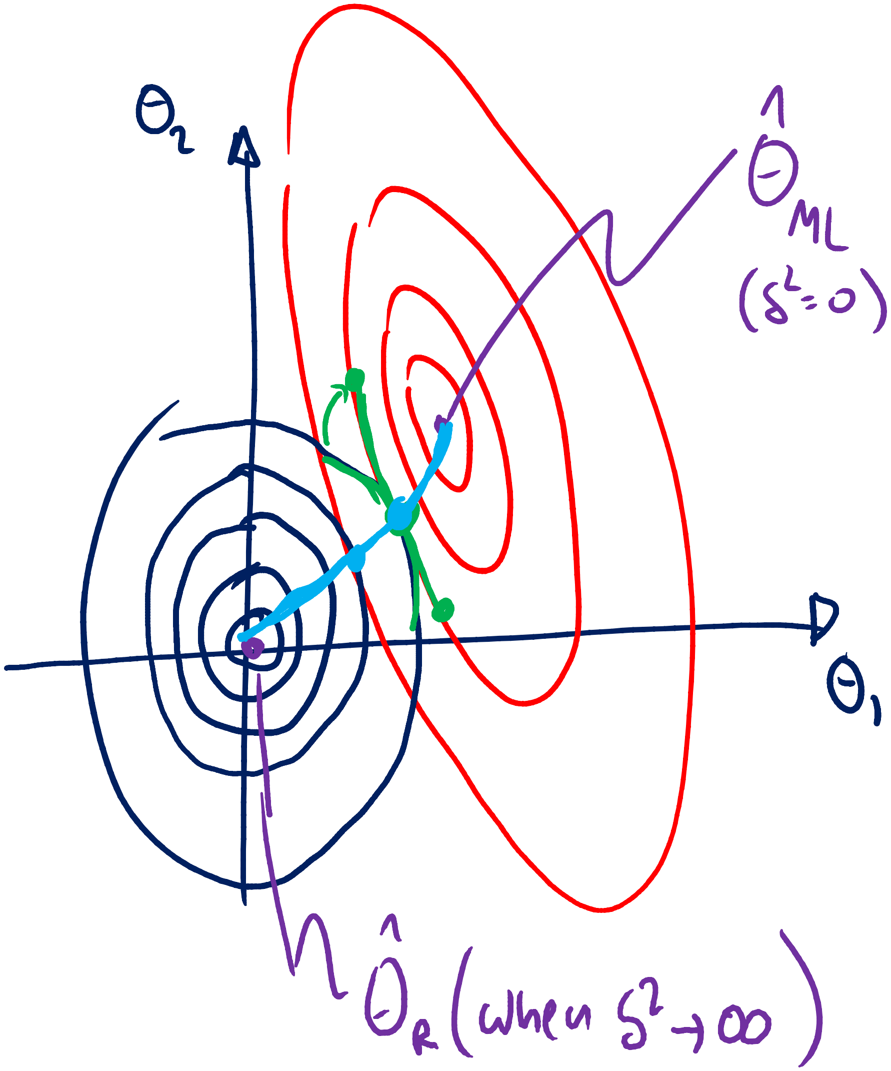
equating to zero

$$(X^T X + \delta^2 I)\theta = X^T y$$

Ridge regression as constrained optimization

$$J(\boldsymbol{\theta}) = \underbrace{\overset{\text{ellipses}}{(\mathbf{y}-\mathbf{X}\boldsymbol{\theta})^T(\mathbf{y}-\mathbf{X}\boldsymbol{\theta})}}_{} + \delta^2 \underline{\cancel{\boldsymbol{\theta}}^T\boldsymbol{\theta}}$$

$$\min_{\boldsymbol{\theta}\,:\,\boxed{\boldsymbol{\theta}^T\boldsymbol{\theta} \le t(\delta)}} \{\underline{(\mathbf{y}-\mathbf{X}\boldsymbol{\theta})^T(\mathbf{y}-\mathbf{X}\boldsymbol{\theta})}\}$$



in 2D

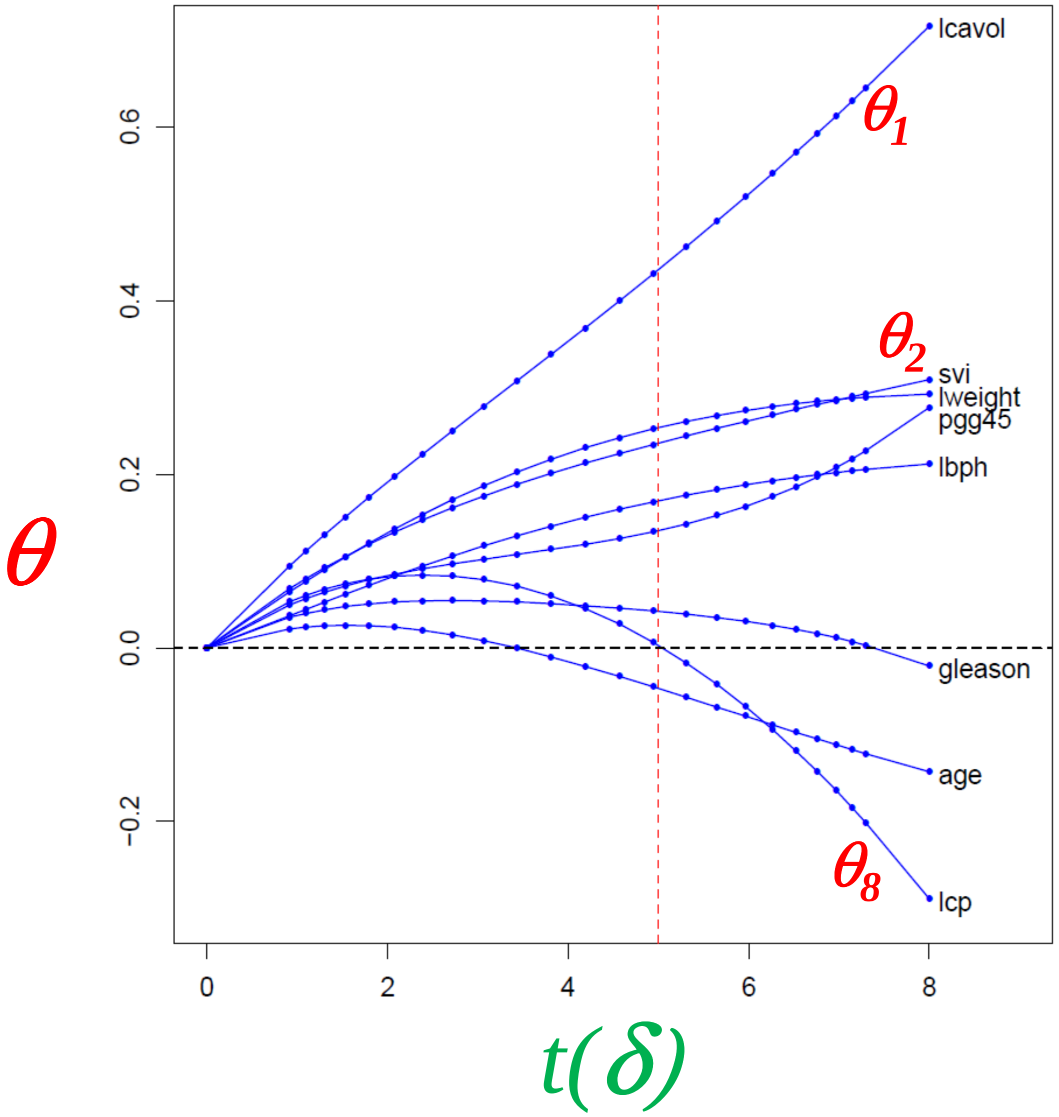
$\underline{\theta} = (\theta_1, \theta_2)$

$\underline{\theta}^T\underline{\theta} = \theta_1^2 + \theta_2^2$

$\theta_1^2 + \theta_2^2 \le t^2$

Regularization paths

As δ increases, $t(\delta)$ decreases and each θ_i goes to zero.



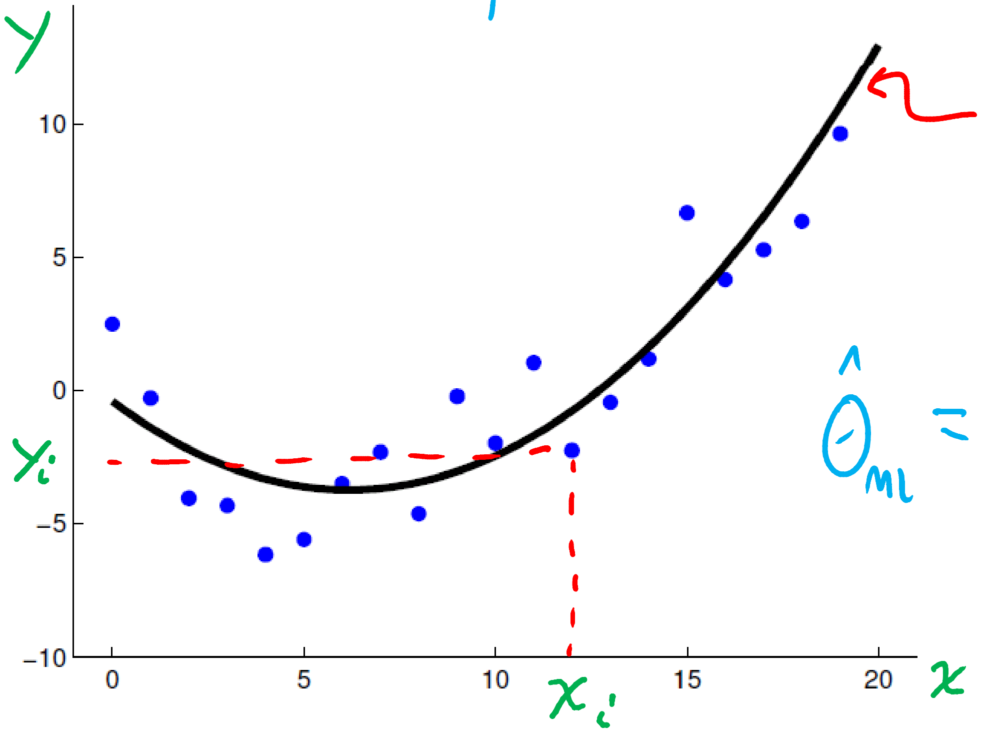
[Hastie, Tibshirani & Friedman book]

Going nonlinear via basis functions

We introduce basis functions $\phi(\cdot)$ to deal with nonlinearity:

$$y(\mathbf{x}) = \underbrace{\phi(\mathbf{x})}_{X}\boldsymbol{\theta} + \epsilon$$

For example, $\phi(x) = [1, x, x^2]$



$$\hat{y} = \phi(x)\,\theta$$

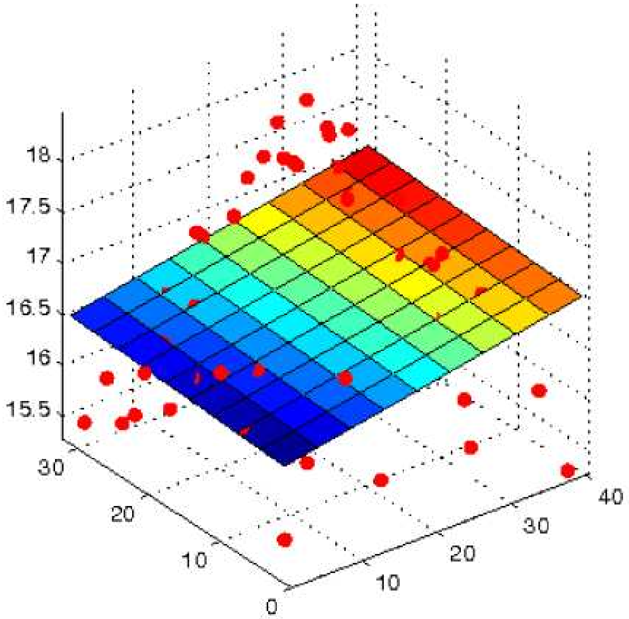
$$= \theta_0 + x\theta_1 + x^2\theta_2$$

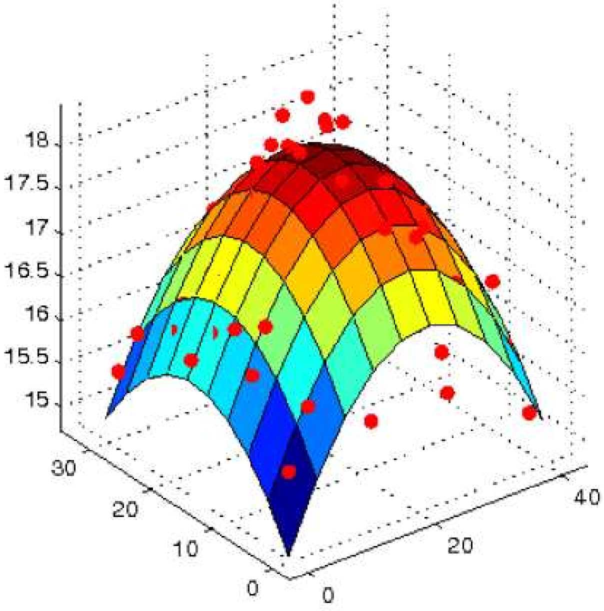
$$\hat{\theta}_{ML} = \left[\phi(x)^T\phi(x)\right]^{-1}\phi(x)^T y$$

Going nonlinear via basis functions

$$y(\mathbf{x}) = \boldsymbol{\phi}(\mathbf{x})\boldsymbol{\theta} + \epsilon$$

$$\boldsymbol{\phi}(\mathbf{x}) = \underbrace{[1, x_1, x_2]} \qquad \boldsymbol{\phi}(\mathbf{x}) = \underbrace{[1, x_1, x_2, \overset{x_1 x_2}{x_1^2}, x_2^2]}$$



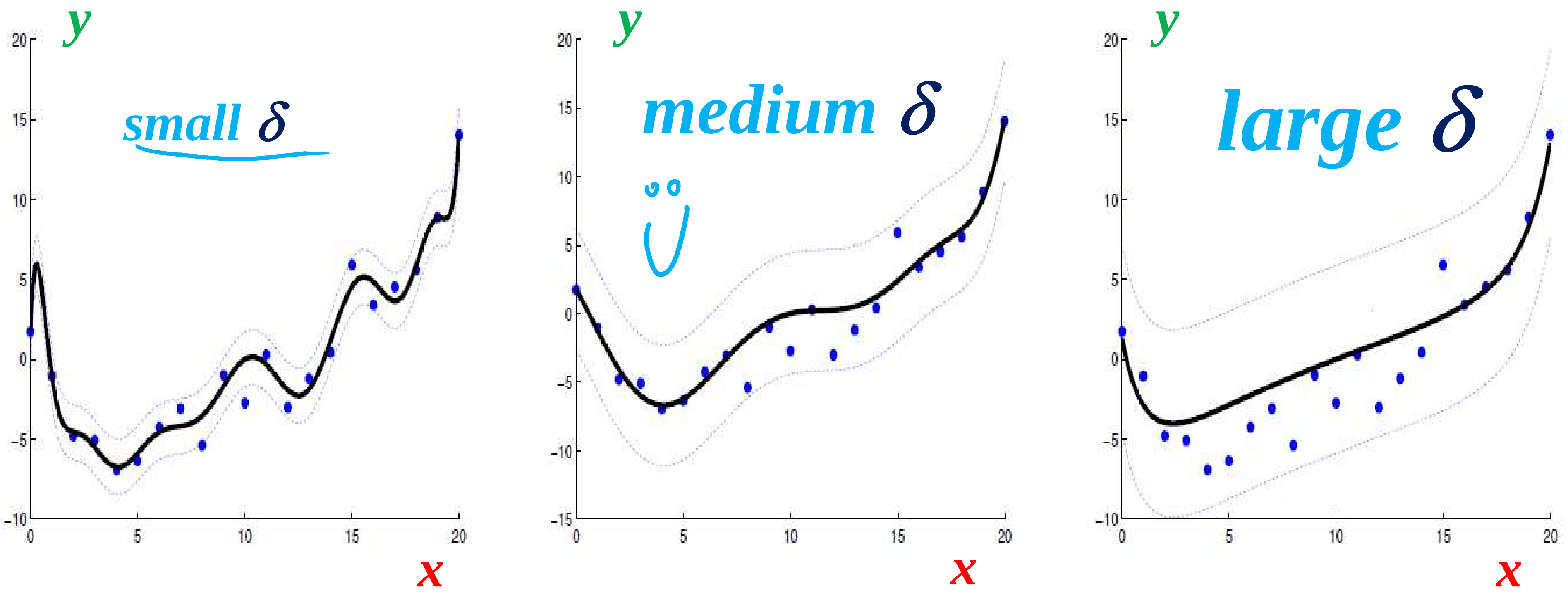


Example: Ridge regression with a polynomial of degree 14

$$\hat{y}(x_i) = 1\,\theta_0 + x_i\,\theta_1 + x_i^2\,\cancelto{0}{\theta_2} + \ldots + x_i^{13}\,\theta_{13} + x_i^{14}\,\theta_{14}$$

$$\underline{\Phi} = [\,1\;\; x_i\;\; x_i^2\;\; \ldots\;\; x_i^{13}\;\; x_i^{14}\,]\;\; x_i^{15}\; \ldots$$

$$J(\theta) = (\,y - \Phi\,\theta\,)^T(\,y - \Phi\,\theta\,) + \underbrace{\delta^2\,\underline{\theta}^T\theta}$$



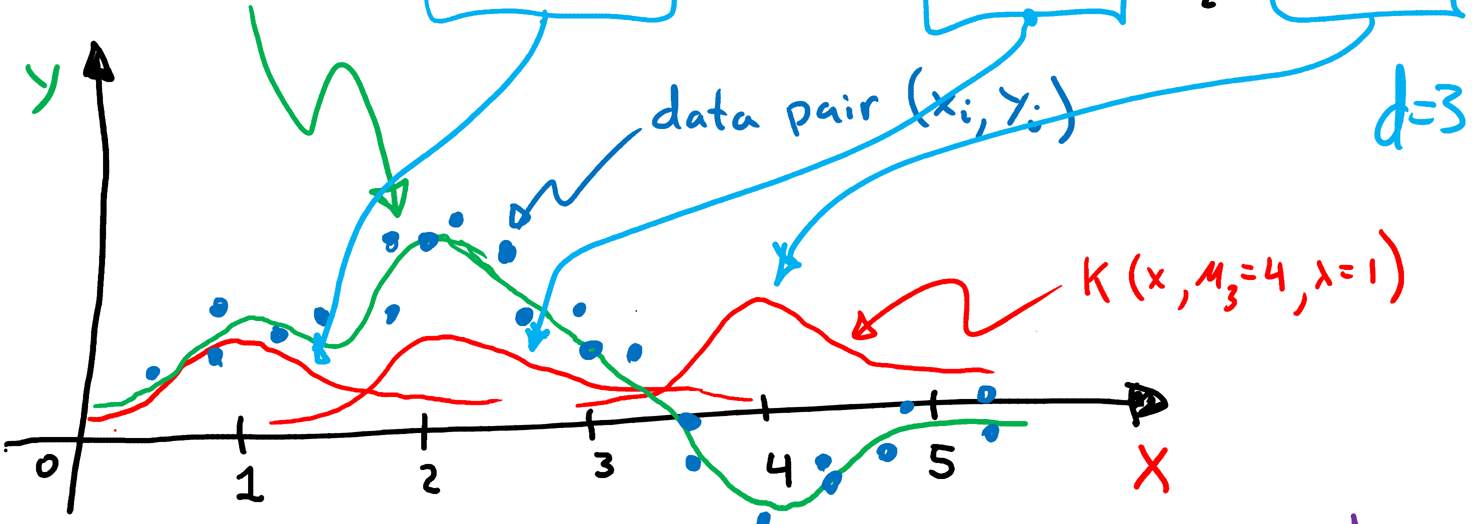
Kernel regression and RBFs

We can use kernels or radial basis functions (RBFs) as features:

$$\phi(\mathbf{x}) = [\kappa(\mathbf{x},\boldsymbol{\mu}_1,\lambda),\ldots,\kappa(\mathbf{x},\boldsymbol{\mu}_d,\lambda)], \quad e.g.\ \ \kappa(\mathbf{x},\boldsymbol{\mu}_i,\lambda) = e^{(-\frac{1}{\lambda}\|\mathbf{x}-\boldsymbol{\mu}_i\|^2)}$$

$$\hat{y}(x_i) = \phi(x_i)\,\theta = 1\theta_0 + k(x_i,\mu_1,\lambda)\,\theta_1 + \ldots + k(x_i,\mu_d,\lambda)\,\theta_d$$

Example 1: $\hat{Y}(x) = \underbrace{e^{-\|x-1\|^2}}\,\theta_1 + \underbrace{e^{-\|x-\overset{\mu_2=2}{2}\|^2}}\,\theta_2 + \underbrace{e^{-\|x-\overset{\mu_3=4}{4}\|^2}}\,\theta_3$



The green curve is a weighted sum of the 3 red curves

$$\phi(x_i) = [1 \quad \kappa(x_i, \mu_1, \lambda) \quad \kappa(x_i, \mu_2, \lambda) \quad \kappa(x_i, \mu_3, \lambda)]$$

$\phi(x_i)$ is a vector with 4 entries. There are 3 bases.

The corresponding vector of parameters is $\underline{\theta} = [\theta_0 \ \ \theta_1 \ \ \theta_2 \ \ \theta_3]^T$

$$\hat{y}_i = \phi(x_i)\,\underline{\theta}$$

If we have $i = 1, \ldots, N$ data, let

$$Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} \qquad \Phi = \begin{bmatrix} \phi(x_1) \\ \phi(x_2) \\ \vdots \\ \phi(x_N) \end{bmatrix}$$

Then

$$\hat{Y} = \Phi \Theta$$

and

$$\hat{\Theta}_{LS} = \left(\Phi^T \Phi\right)^{-1} \Phi^T Y$$

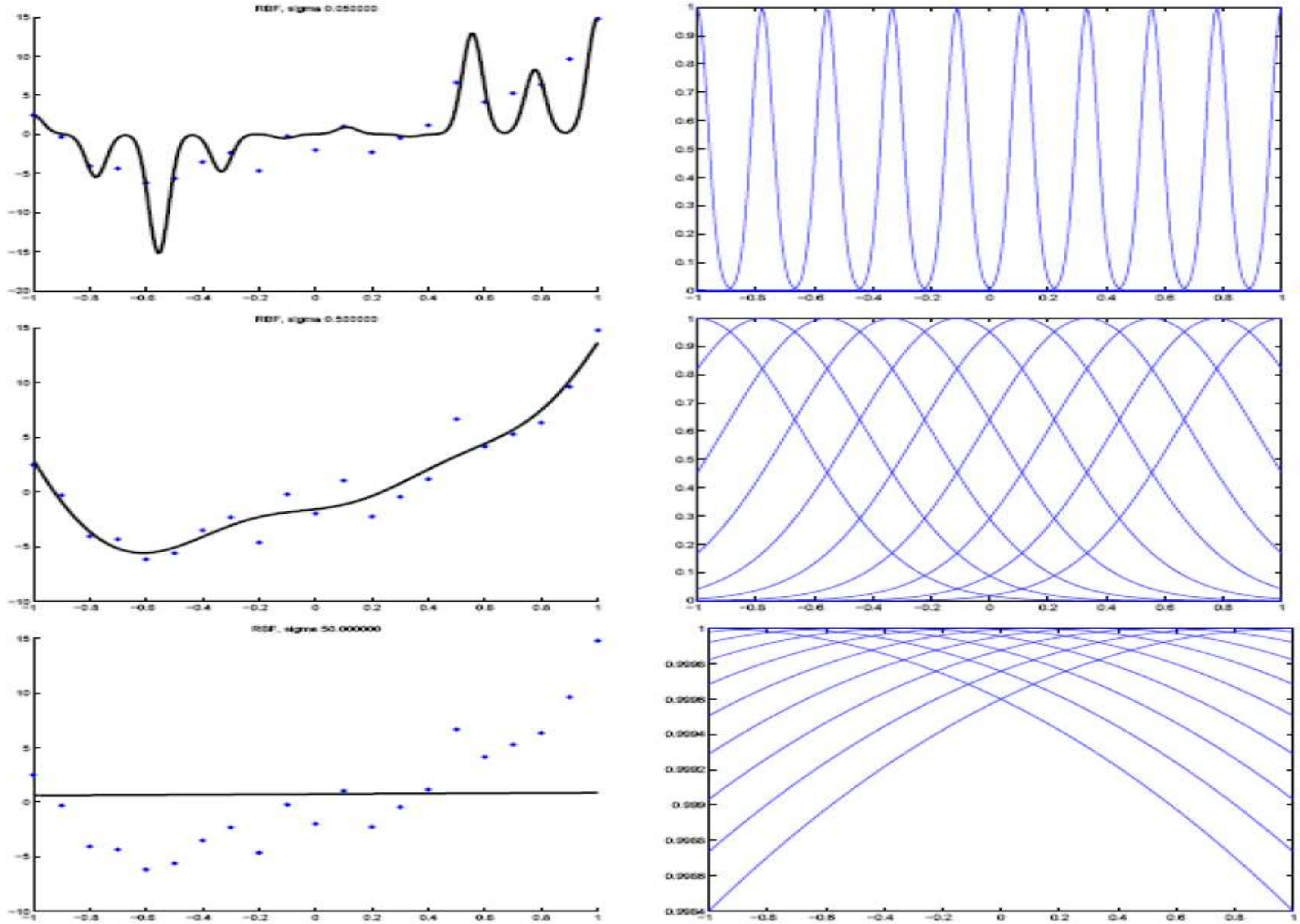
or

$$\hat{\Theta}_{ridge} = \left(\Phi^T \Phi + \delta^2 I\right)^{-1} \Phi^T Y$$

Hence, this is still linear regression, with X replaced by Φ.

We can choose the locations μ of the ***basis functions*** *to be the inputs.*
That is, $\mu_i = \mathbf{x}_i$. These basis functions are the known as ***kernels***.
The choice of width λ is tricky, as illustrated below.

kernels



Too small λ

Right λ

Too large λ

The big question is how do we choose the regularization coefficient, the width of the kernels or the polynomial order?

One Solution: **cross-validation**

① Given training data (X_{train}, y_{train}), and some δ^2 guess, compute $\hat{\Theta}$

② $\hat{y}_{train} = X_{train}\hat{\Theta}$ (compute training set predictions)

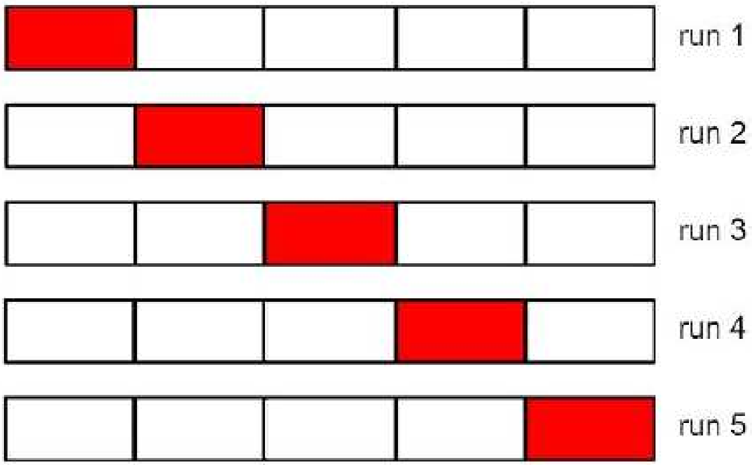
③ $\hat{y}_{test} = X_{test}\hat{\Theta}$

train | test

validation

	δ^2	Train error $\sum_{i \in train}(y_i-\hat{y}_i)^2$	Test error $\sum_{i \in test}(y_i-\hat{y}_i)^2$	max	min_max	avg
0	0.1	100	2	100		51
35	1	10	11	11	X	10.5
3	10	1	19	19		10 } X
21	50	20	0	20		10
0	100	100	1000	1000		550

K-fold crossvalidation

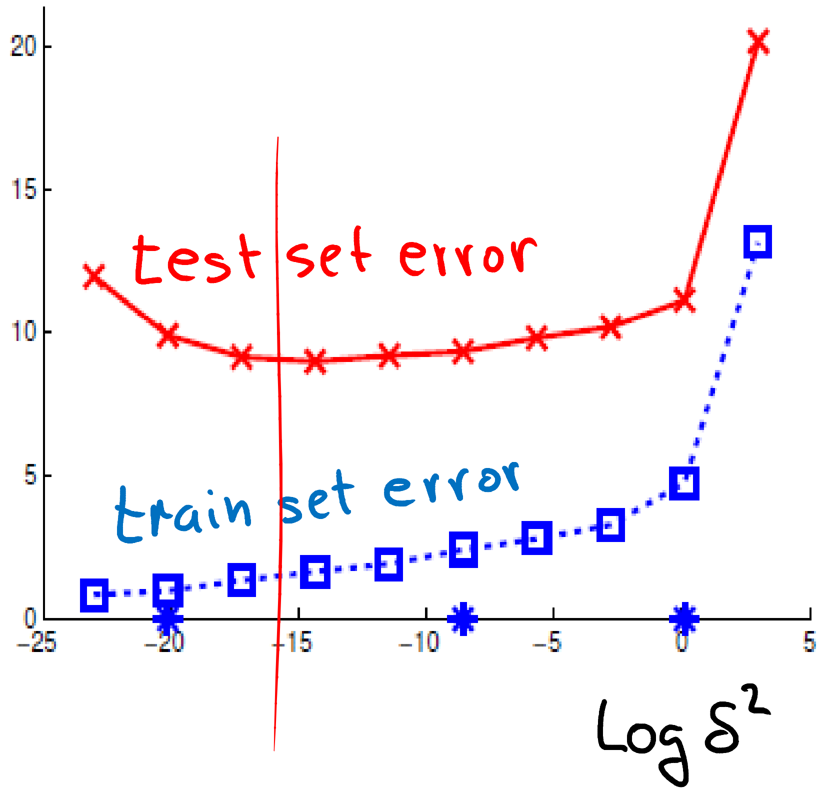


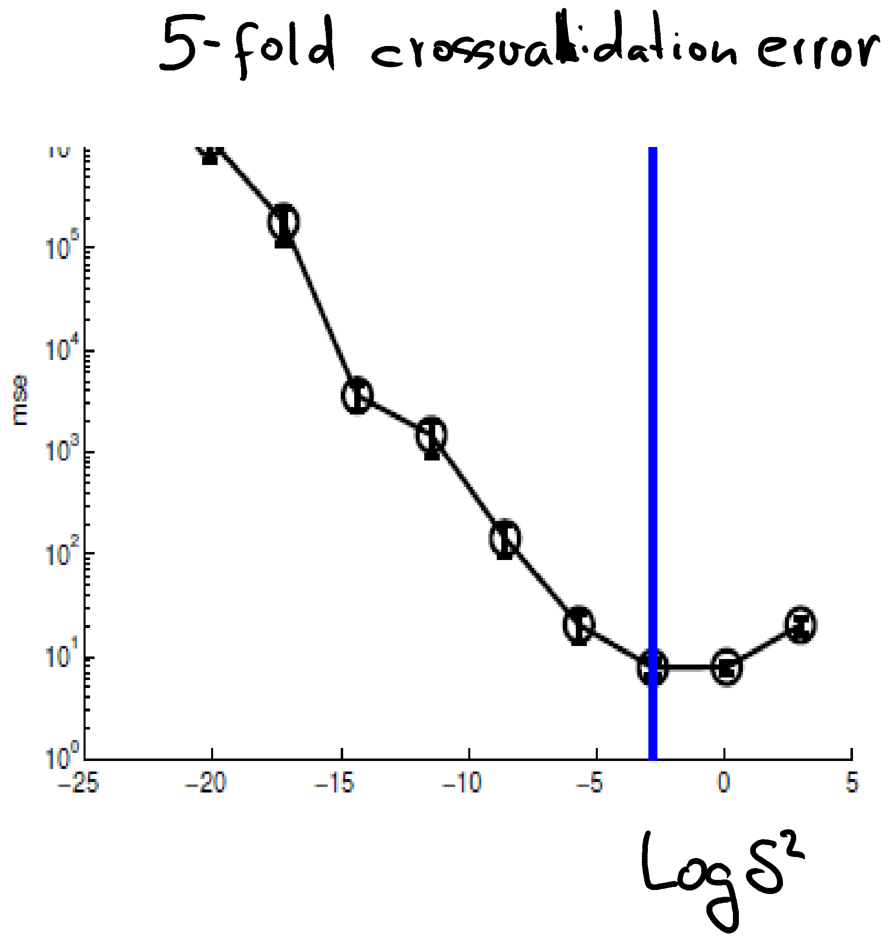
The idea is simple: we split the training data into K **folds**; then, for each fold $k \in \{1, \ldots, K\}$, we train on all the folds but the k'th, and test on the k'th, in a round-robin fashion.

It is common to use $K = 5$; this is called 5-fold CV.

If we set $K = N$, then we get a method called **leave-one out cross validation**, or **LOOCV**, since in fold i, we train on all the data cases except for i, and then test on i.

Example: Ridge regression with polynomial of degree 14



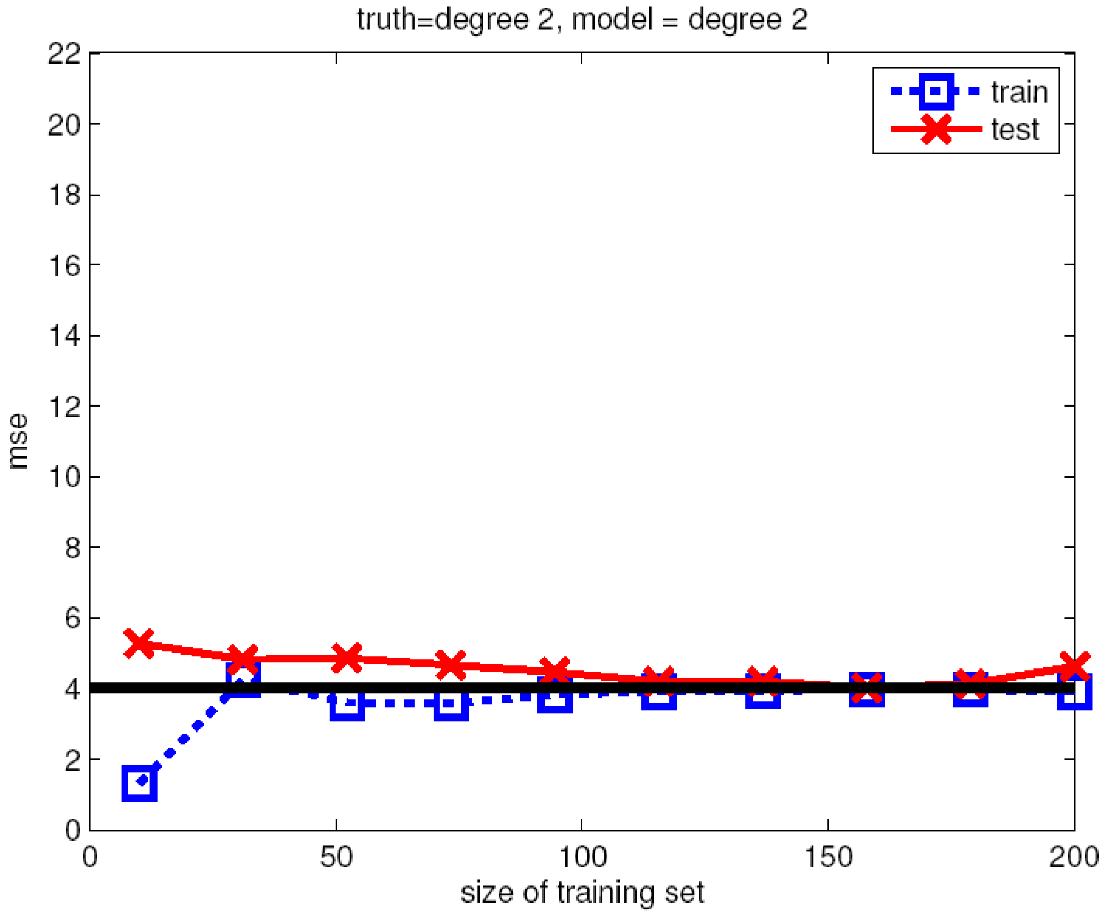


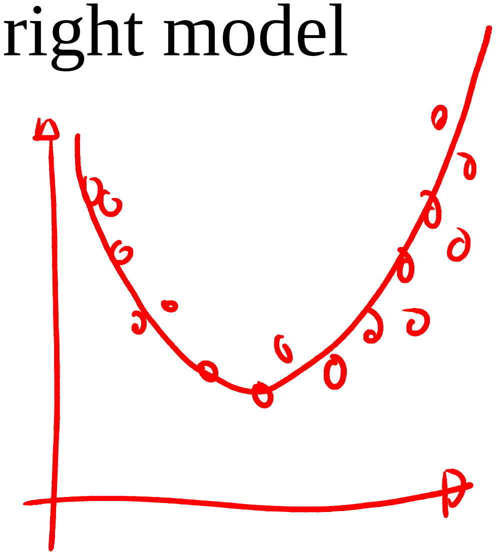
The larger δ, the larger the train set error. However the test set error improves. For good generalization we want to do well in all test sets.

Effect of data when we have the right model

$$y_i = \theta_0 + x_i\,\theta_1 + x_i^2\,\theta_2 + \mathcal{N}(0\,,\sigma^2)$$

$$\hat{y}_i = \hat{\Theta}_0 + x_i\,\hat{\theta}_1 + x_i^2\,\hat{\Theta}_2 \quad (\text{model})$$

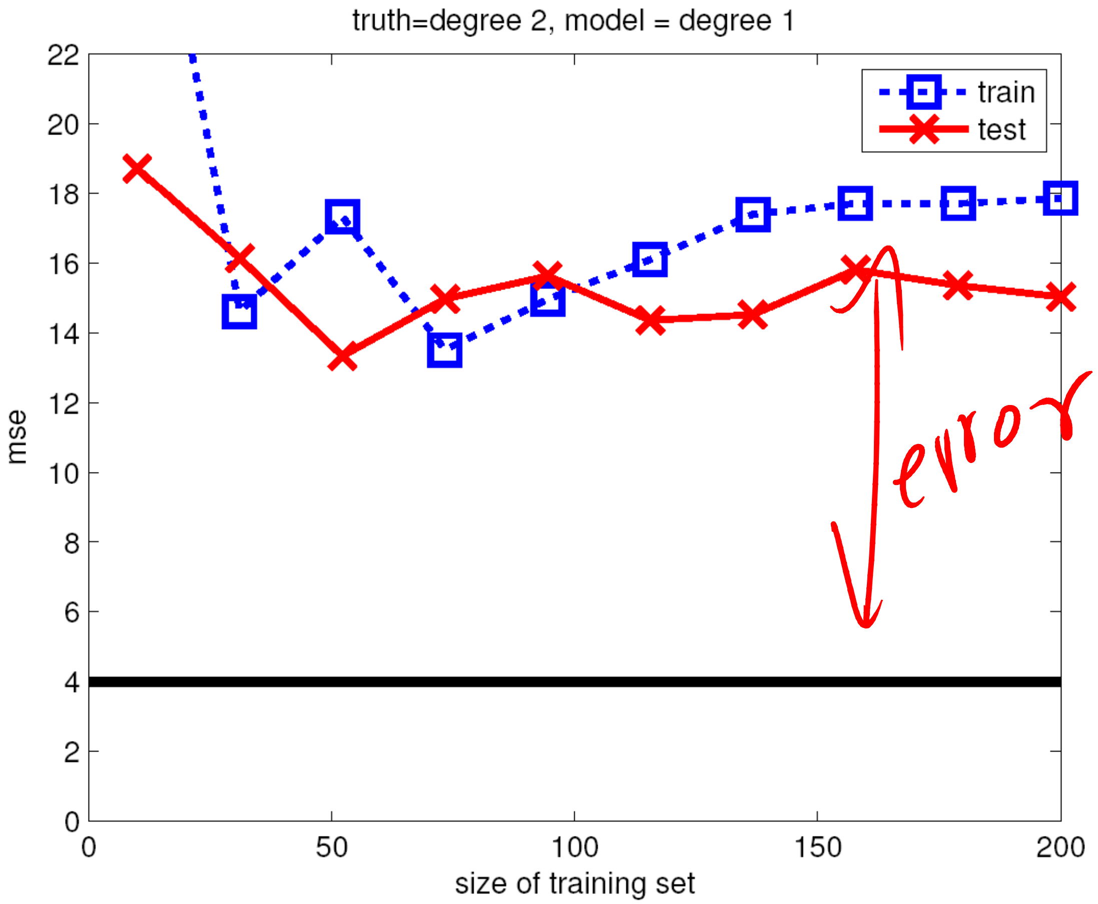


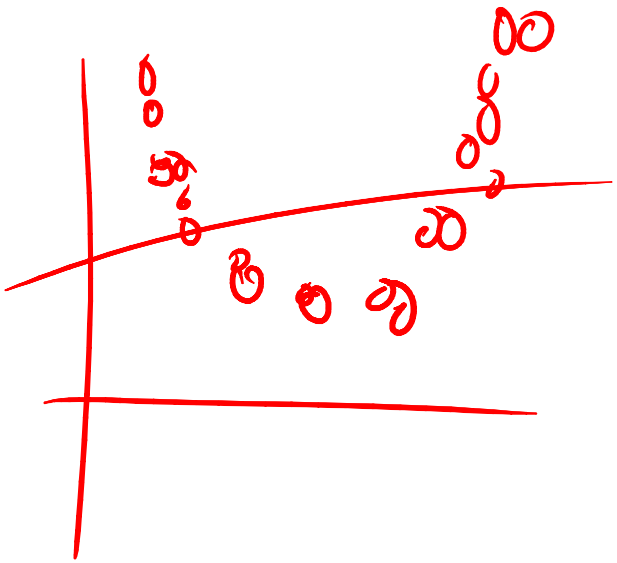


Effect of data when the model is too simple

$$y_i = \theta_0 + x_i\,\theta_1 + x_i^2\,\theta_2 + \mathcal{N}(0, \sigma^2)$$

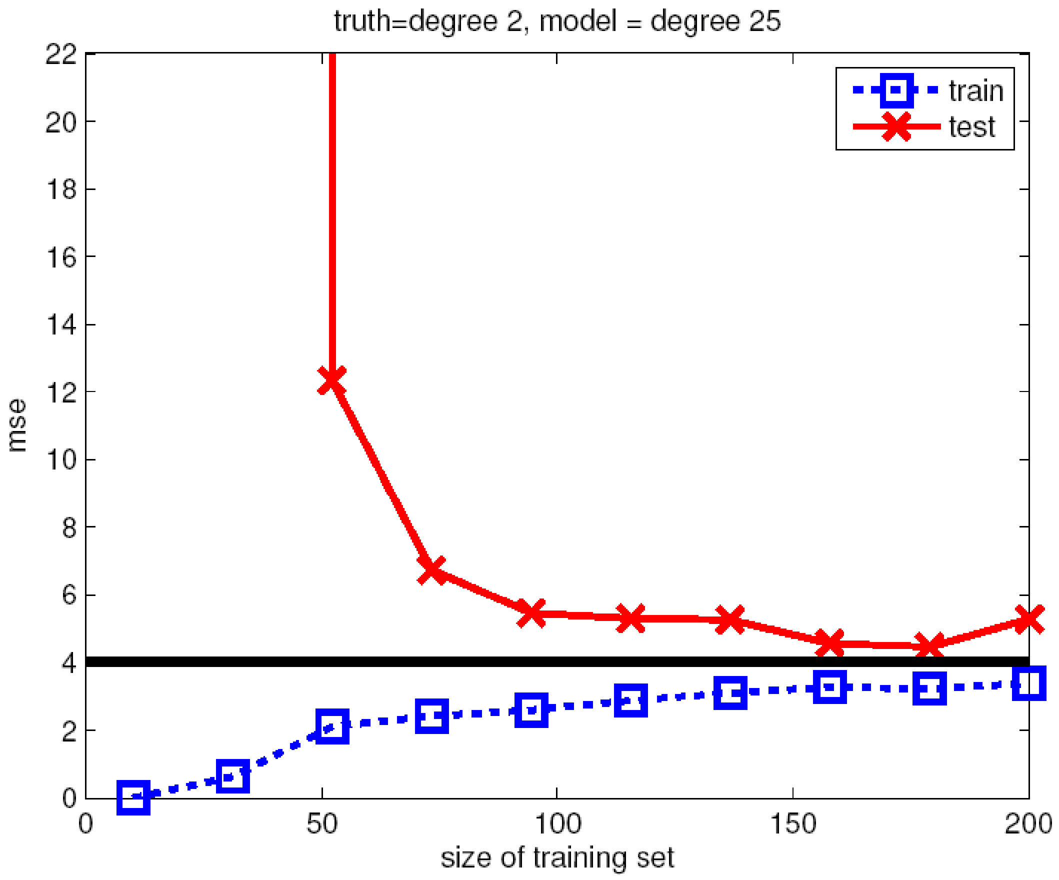
$$\hat{Y}_i = \hat{\theta}_0 + x_i\,\hat{\theta}_1$$

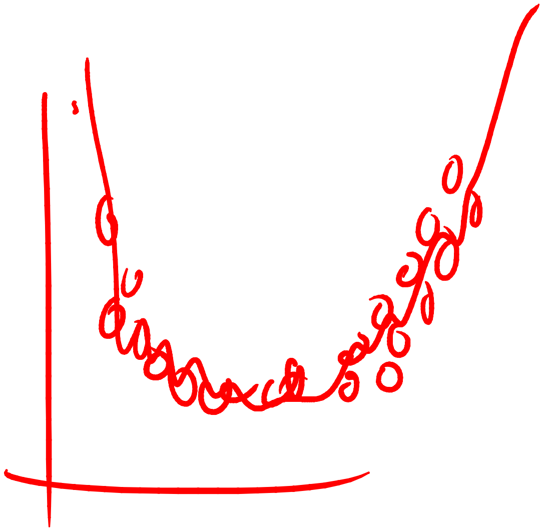


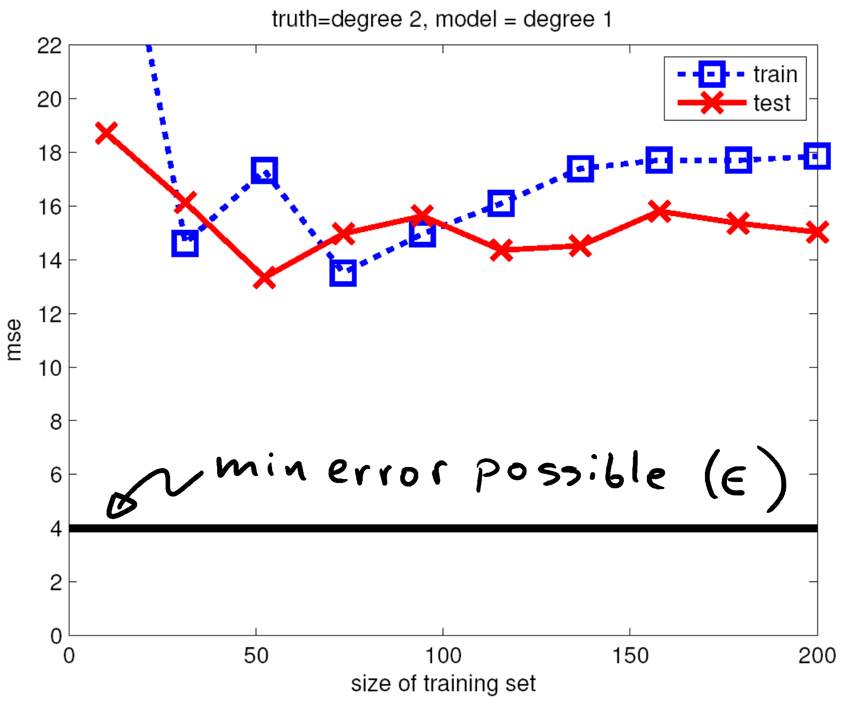


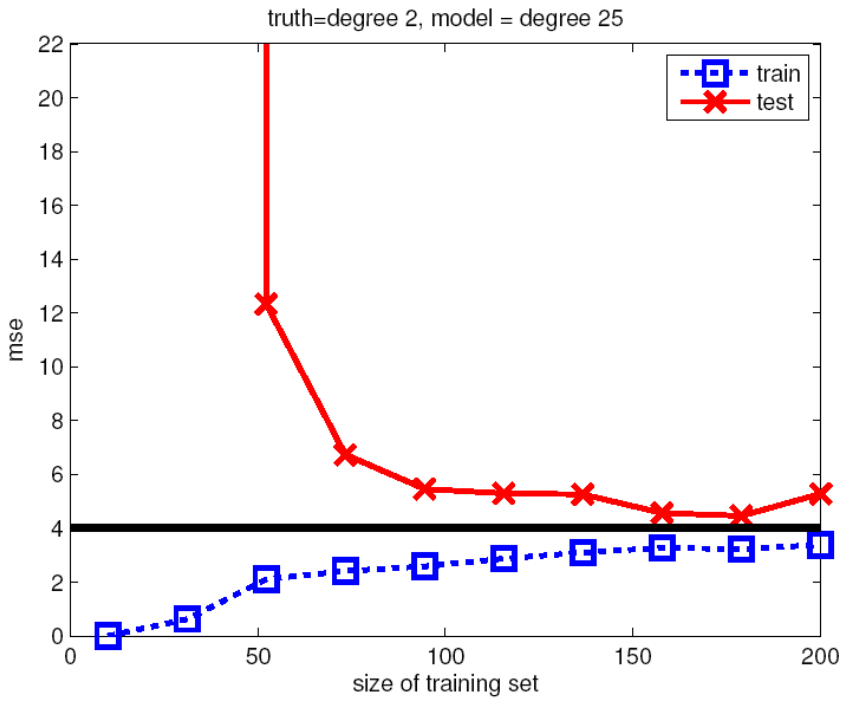
Effect of data when the model is very complex

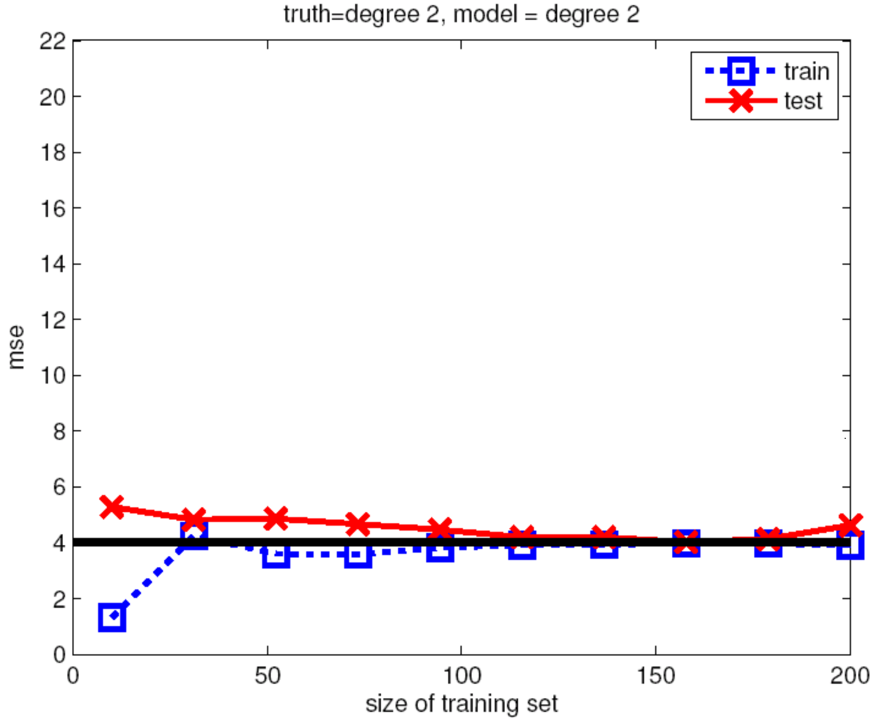
$$y_i = \theta_0 + x_i\,\theta_1 + x_i^2\,\theta_2 + \mathcal{N}(0, \sigma^2)$$





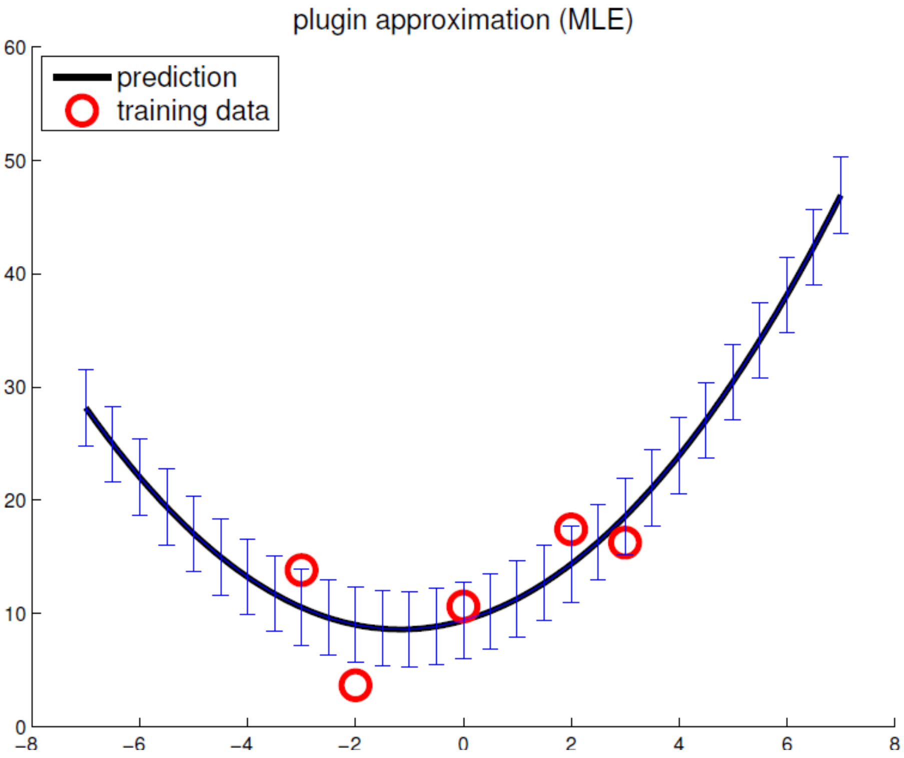


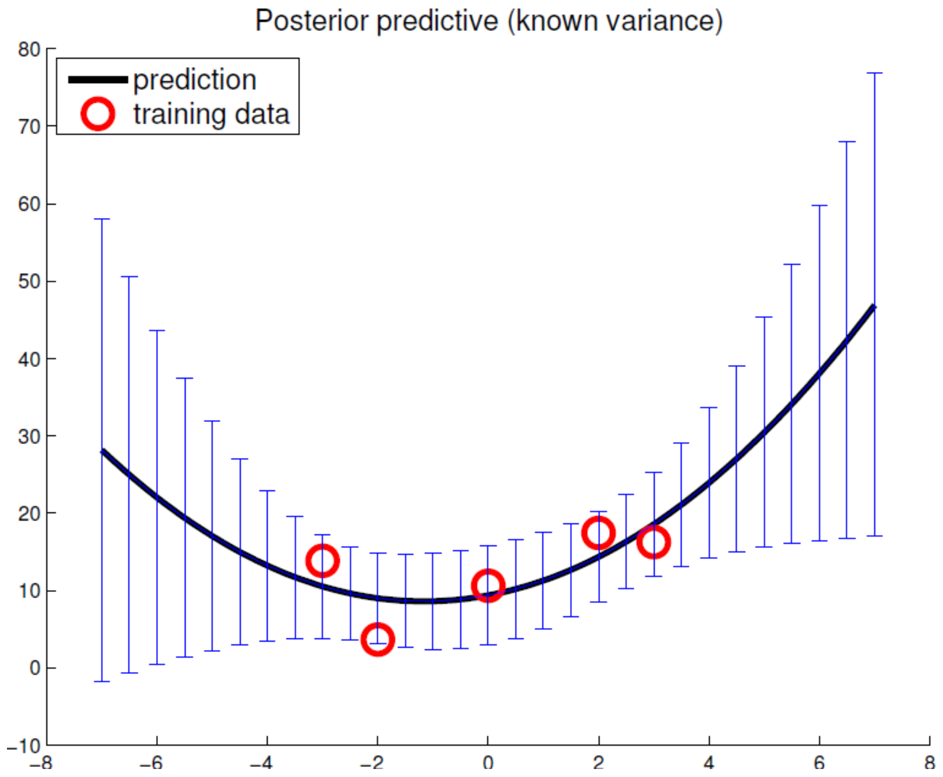




More data improves results, but only if the model has the right complexity.

Confidence in the predictions


plugin approximation (MLE)


Posterior predictive (known variance)

Next lecture

In the next lecture, we introduce Bayesian inference, and show how it can provide us with an alternative way of learning a model from data.