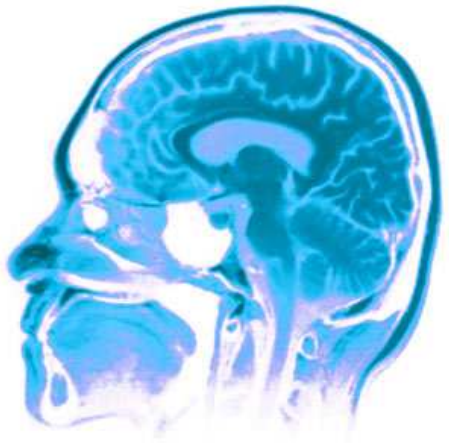
CPSC340



Nonlinear prediction and generalization

Nando de Freitas
October, 2012
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:

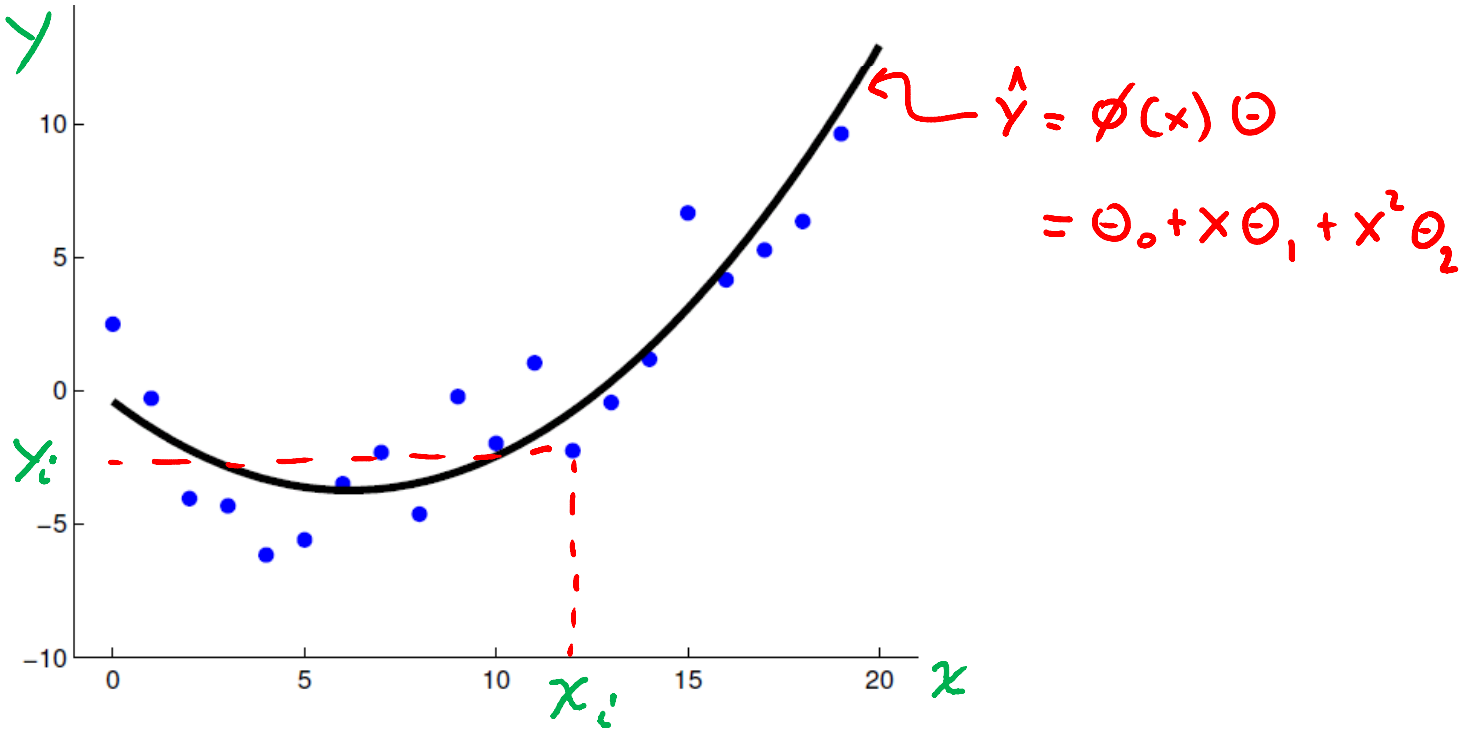
- ❑ Learn **polynomial regression**.
- ❑ Understand that, if the bases are given, the problem of learning the parameters is still linear.
- ❑ Understand the effect of the regularizer on function **smoothness**.
- ❑ Learn about **RBFs** and **kernel regression**.
- ❑ Learn to choose the regularization coefficient by **cross-validation.**
- ❑ Learn about training error and validation error.
- ❑ Understand the effects of the number of data and the number of basis functions on **generalization**.

Going nonlinear via basis functions

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

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

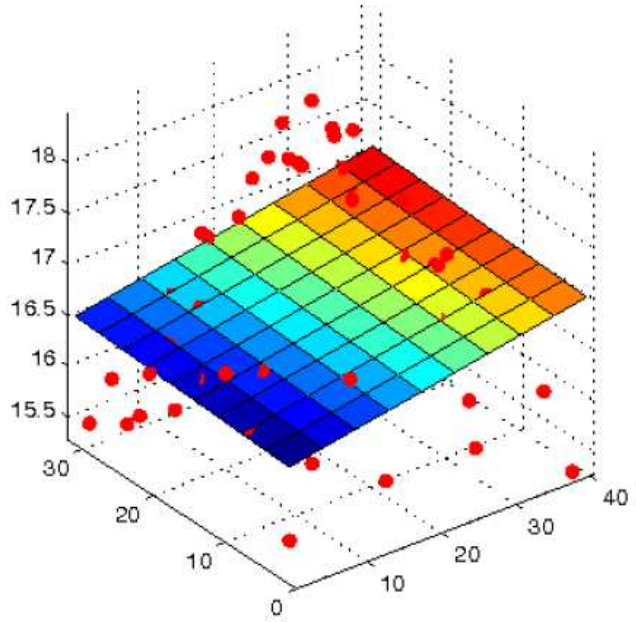
For example, $\boldsymbol{\phi}(x) = [1, x, x^2]$

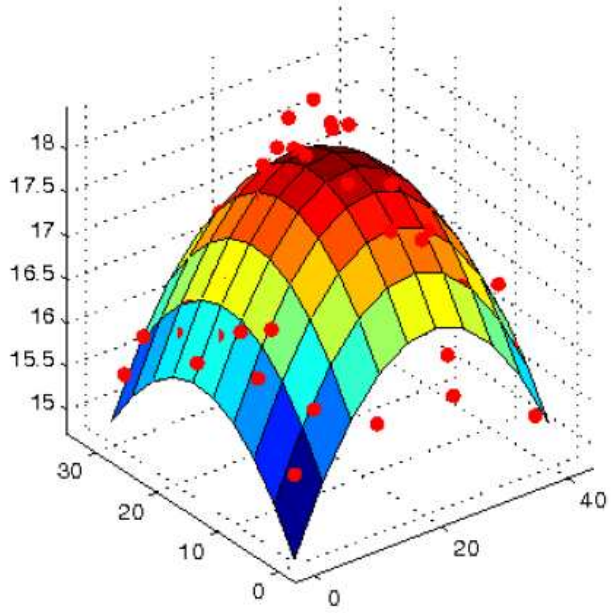


Going nonlinear via basis functions

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

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



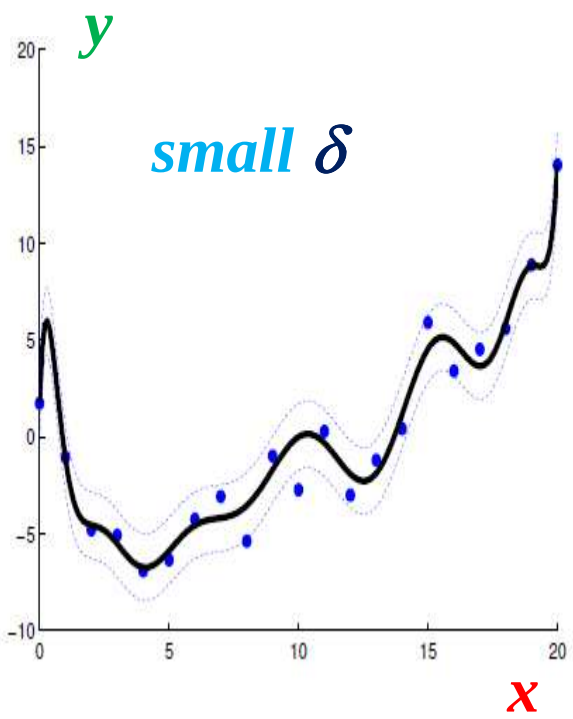


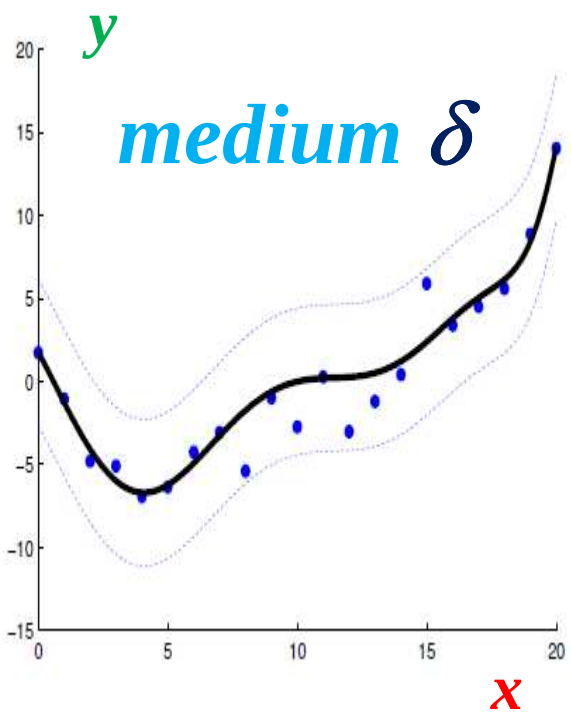
***Example**: Ridge regression with a polynomial of degree 14*

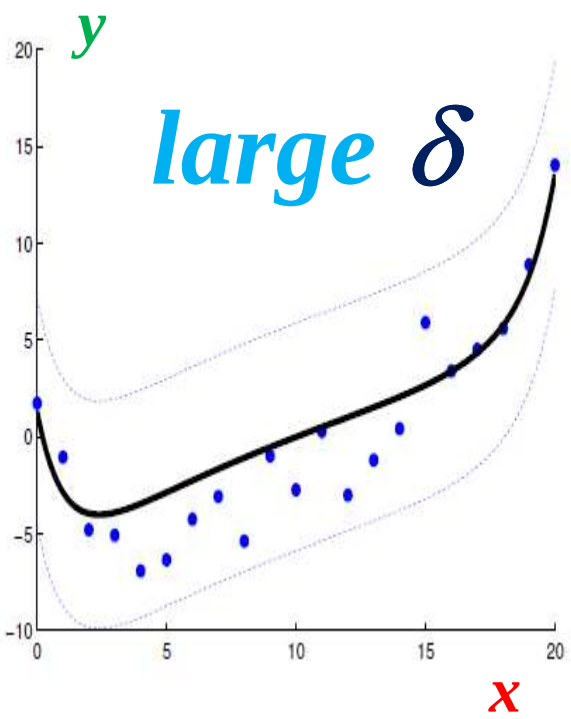
$$\hat{y}(x_i) = 1\ \theta_0 + x_i\ \theta_1 + x_i^2\ \theta_2 + \ldots + x_i^{13}\ \theta_{13} + x_i^{14}\ \theta_{14}$$

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

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







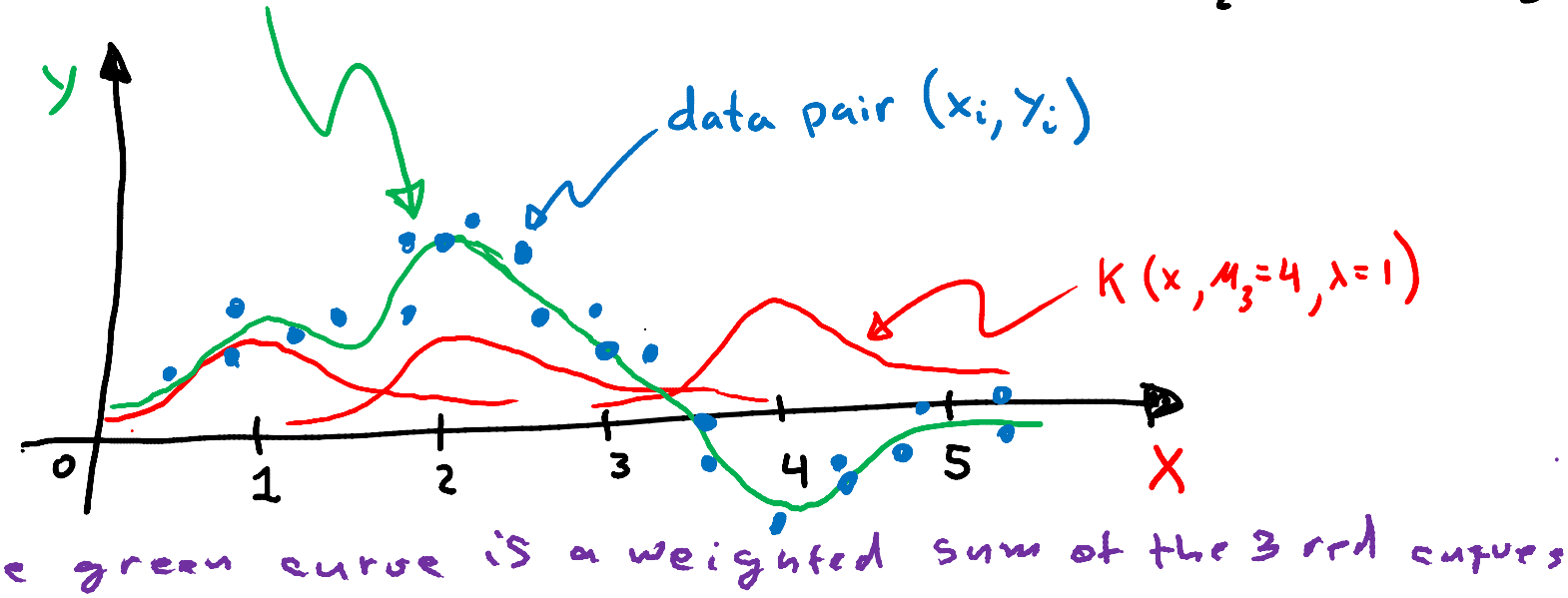
Kernel regression and RBFs

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

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

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

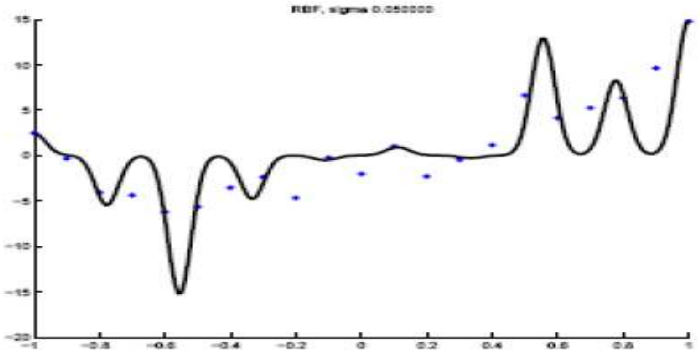
Example 1: $\hat{Y}(x) = e^{-\|x-1\|^2}\theta_1 + e^{-\|x-2\|^2}\theta_2 + e^{-\|x-4\|^2}\theta_3$

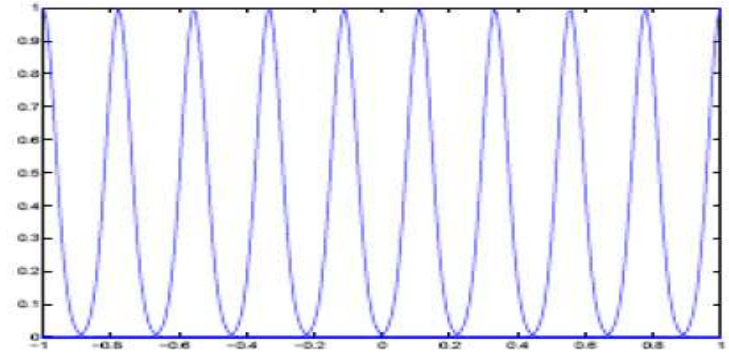


The green curve is a weighted sum of the 3 rbf curves

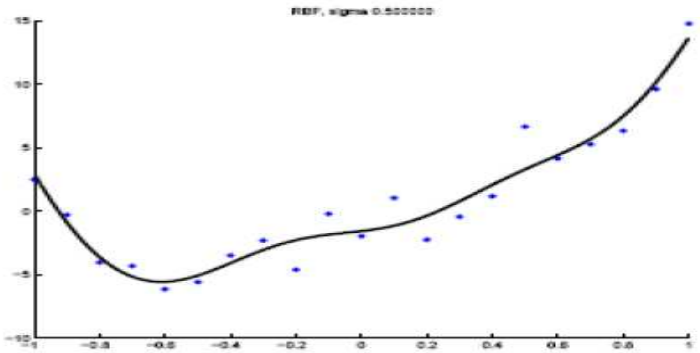
We can choose the locations μ *of the* ***basis functions*** *to be the inputs. That is,* $\mu_i = x_i$ *. These basis functions are the known as* ***kernels****. The choice of width* λ *is tricky, as illustrated below.*

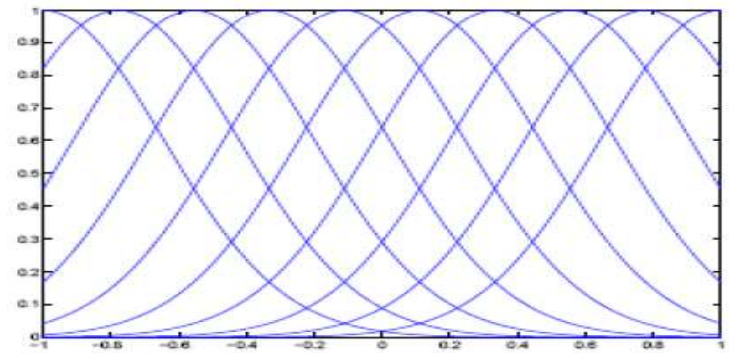
kernels



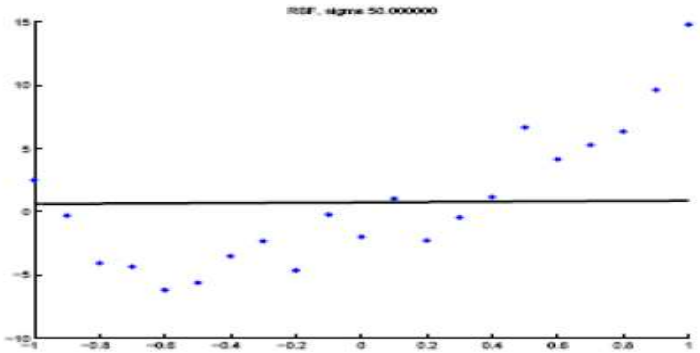


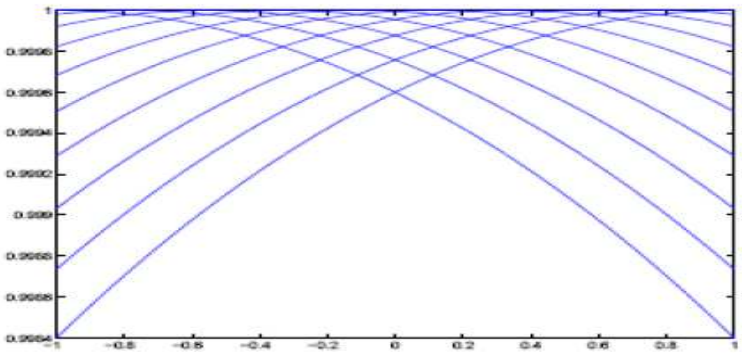
Too small λ





Right λ





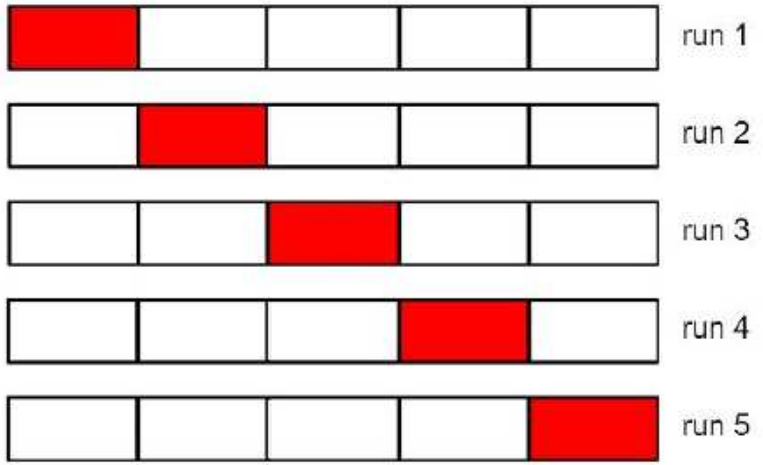
Too large λ

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

Solution: **cross-validation**

λ	S^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 (worst case)	min-max	avg
0	0.1	100	2	100		51
24	1	10	11	11	Choose this	10.5
3	10	1	19	19		10
8	50	20	0	20		10
0	100	100	1000	1000		big

K-fold crossvalidation

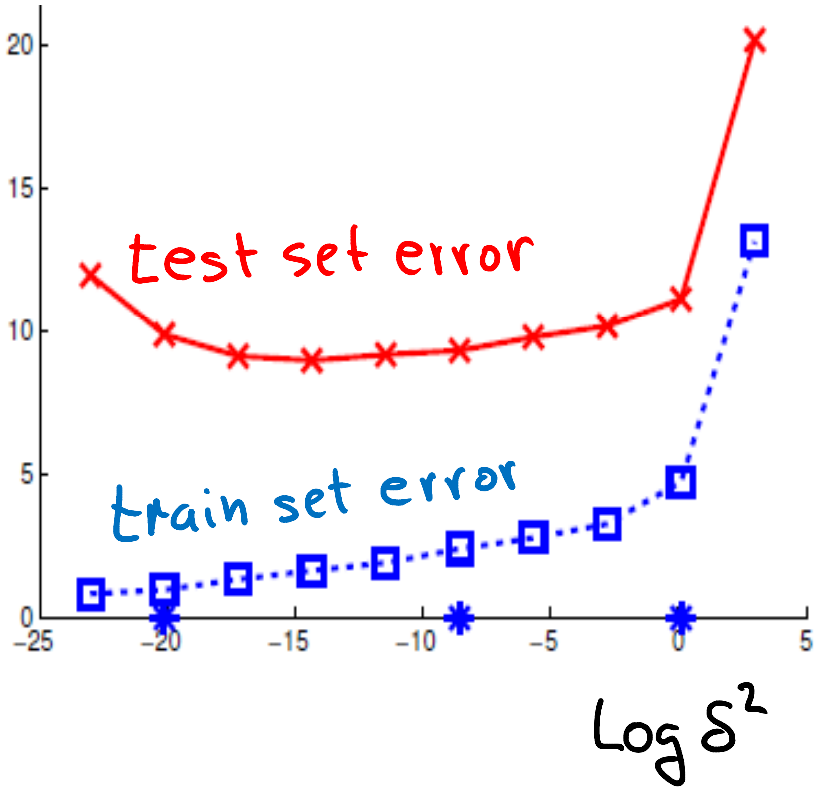


The idea is simple: we split the training data into K **folds**; then, for each fold $k \in \{1, \ldots, K\}$, we train on all the folds but the k'th, and test on the k'th, in a round-robin fashion.

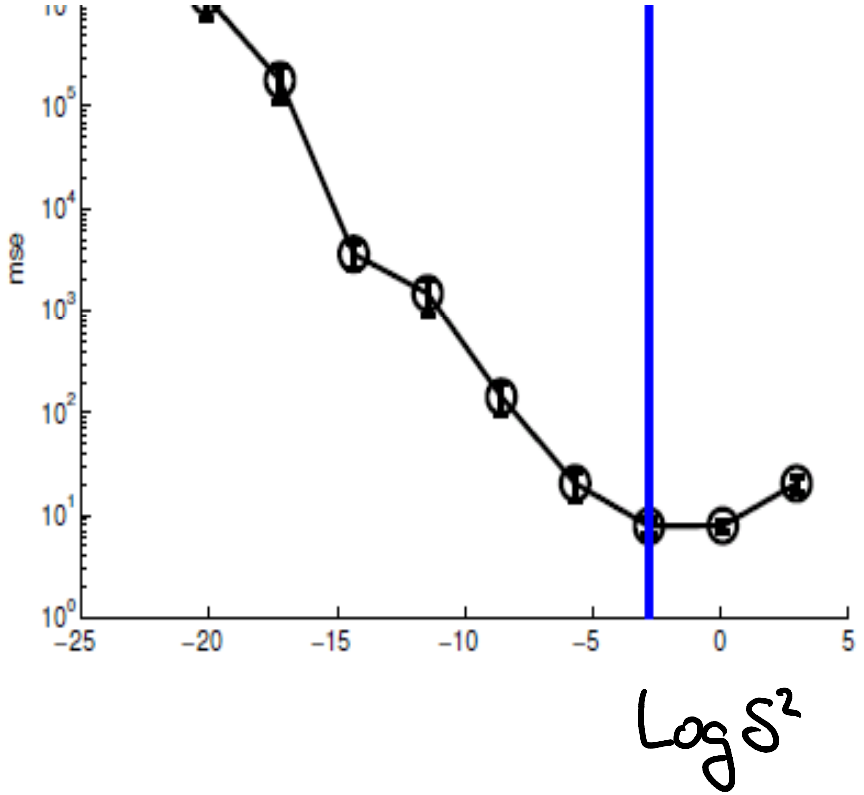
It is common to use $K = 5$; this is called 5-fold CV.

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.

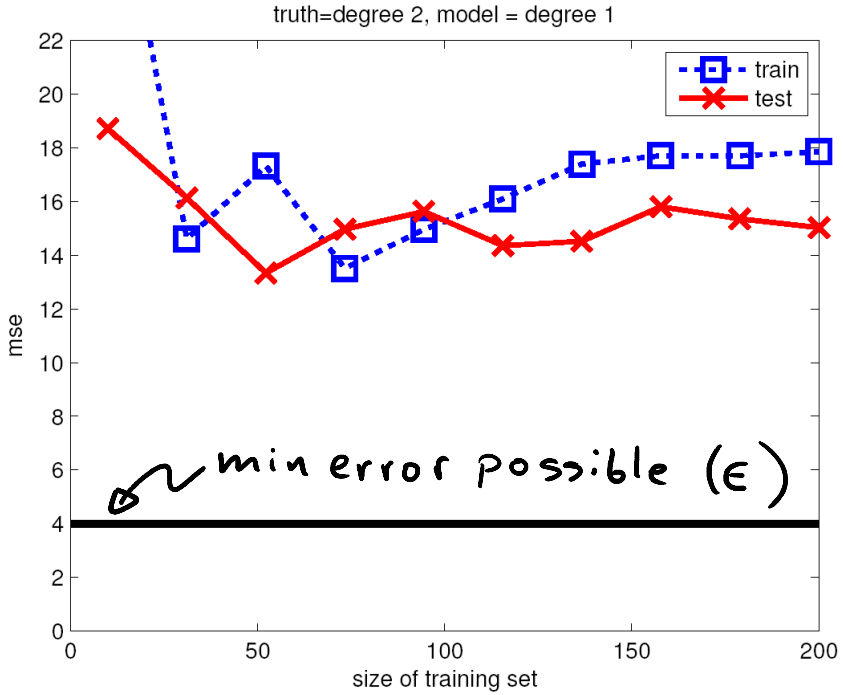
Example: Ridge regression with polynomial of degree 14

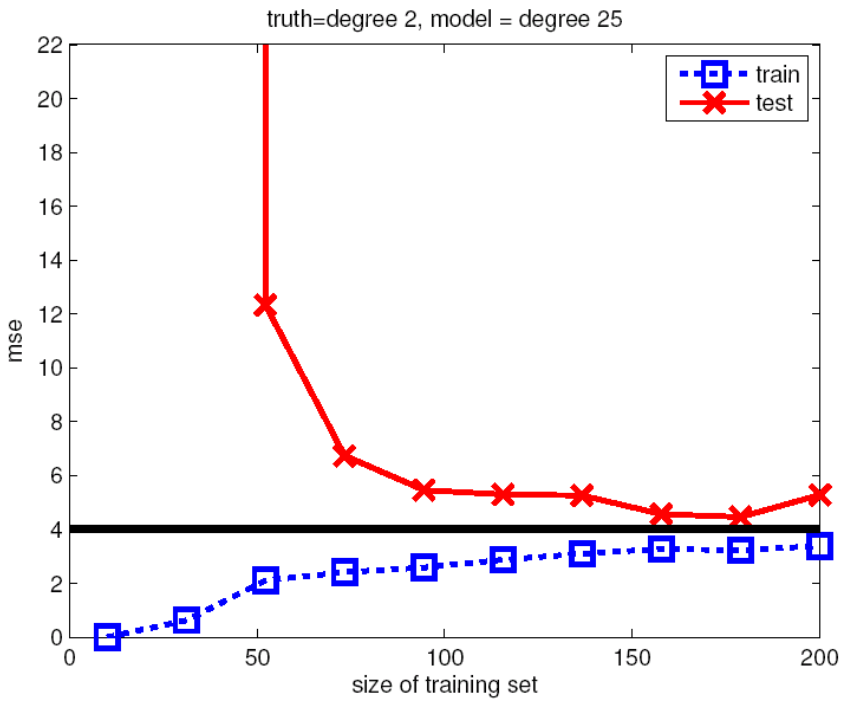


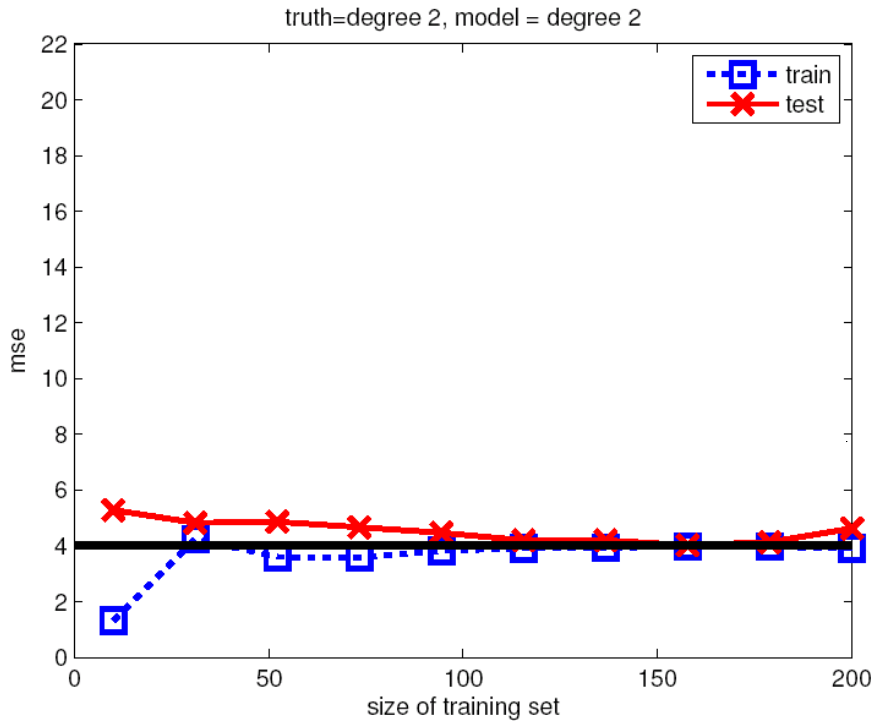
5-fold crossvalidation error



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.







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

Next lecture

In the next lecture, we study the problem of feature selection and introduce a new form of regularizer: the L_1 norm.

This type of regularization is at the heart of a recent revolution in data acquisition known as compressed sensing.