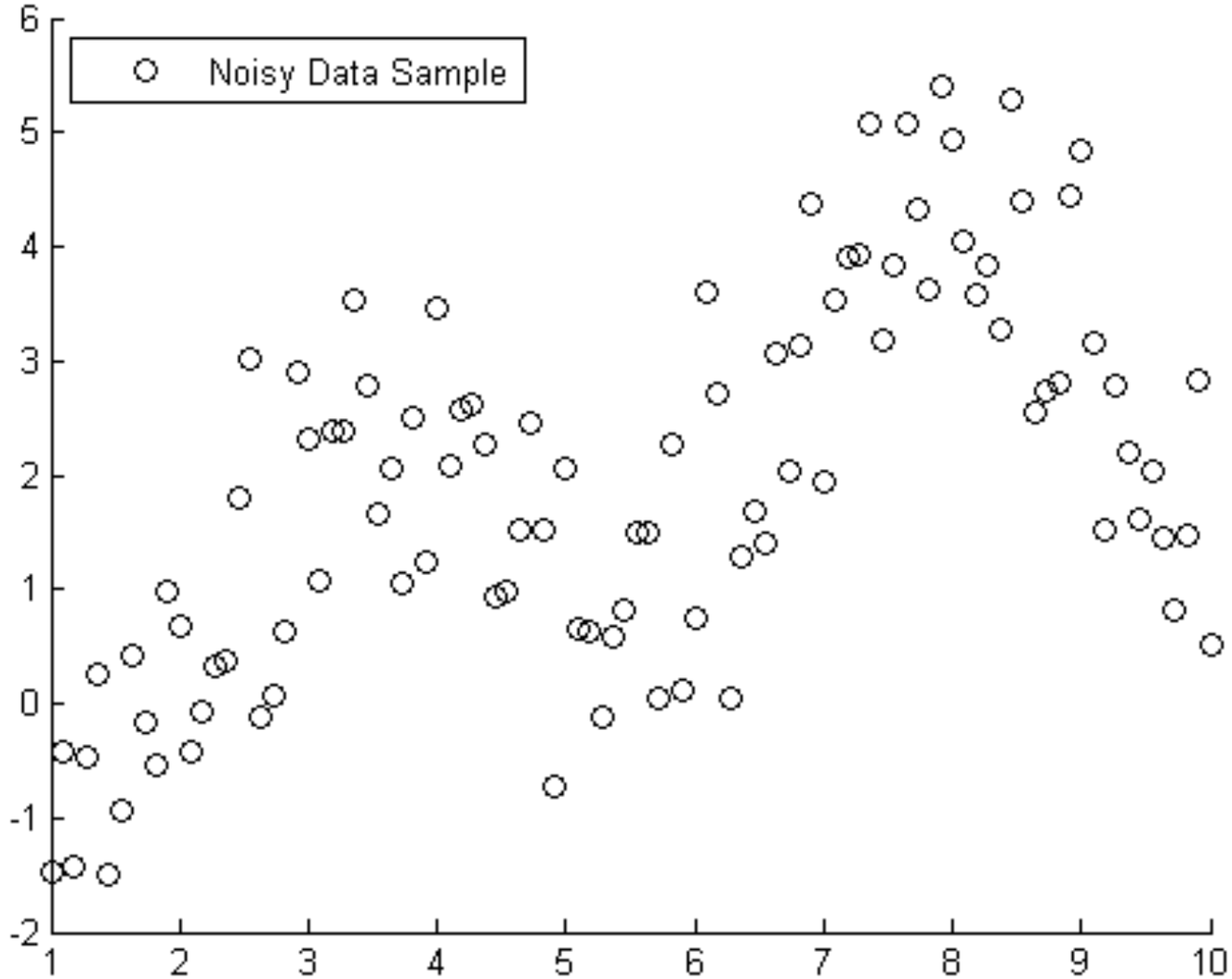
Nonparametric Learning Part I

STAT 37710 / CMSC 35300

Rebecca Willett and Yuxin Chen

Example: nonlinear regression



How should we adapt model complexity to growing data set size?

Linear regression: least squares

$$\hat{w} = \arg\min_{w} \sum_{i=1}^{n} \left(w^\top x_i - y_i \right)^2$$

The above equatian in matrix form:

$$\hat{w} = \arg\min_{w} \left(w^\top X - y \right)^\top \left(w^\top X - y \right)$$

Closed-form solution $\hat{w}$ via basic matrix algebra

$$\hat{w} = \left(X^\top X \right)^{-1} X^\top y$$

Ridge regression: regularized least squares

$$\hat{w} = \arg\min_{w} \sum_{i=1}^{n} \left(w^\top x_i - y_i \right)^2 + \lambda \left\| w \right\|_2^2$$

Can find analytical solution $\hat{w}$:

$$\hat{w} = \left(X^\top X + \lambda I \right)^{-1} X^\top y$$

- Encourage small weights via penalty functions
  - reduces sensitivity to noise and help prevent overfitting
  - higher bias, lower variance

Nonlinear regression

In some cases, the application under consideration might naturally lend itself to a specific nonlinear model.

Examples of nonlinear regression:

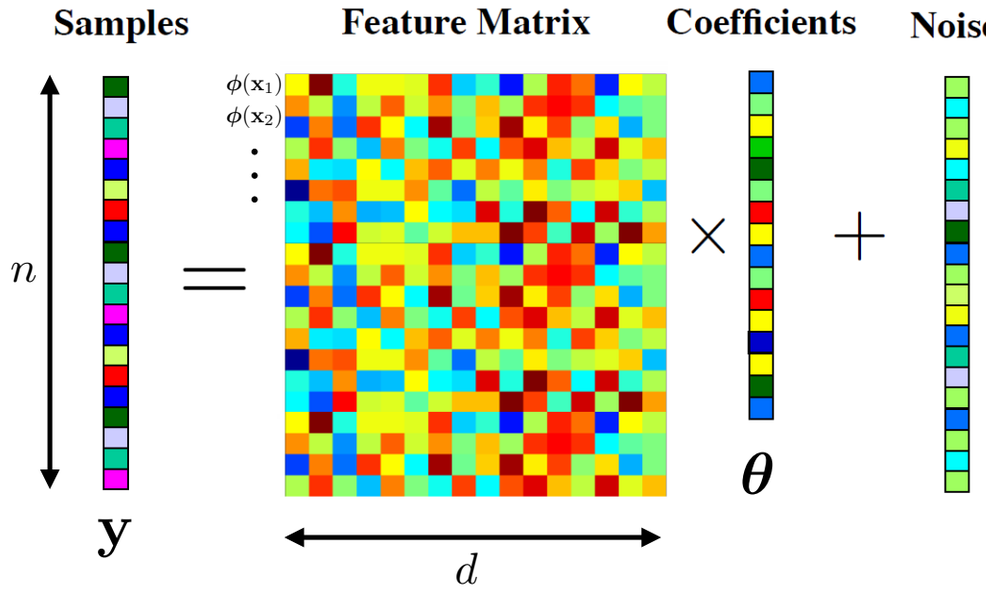
- Logistic regression (e.g., classification)
- Generalized linear models $y = f(w^\top x + b)$, f nonlinear
- Multi-layer perceptron/ neural networks
- ...

Choosing a "good" nonlinear model can be difficult

Linear regression with nonlinear basis functions

We can fit nonlinear functions via linear regression, using nonlinear features of our data (basis functions).

- Choose a feature space $\phi(x) = (\phi_1(x), \ldots, \phi_d(x))^\top$
- Assume (approximately) linear relation between $\phi(x)$ and y: $y = \theta^\top \phi(x) + z$ where z is possible noise

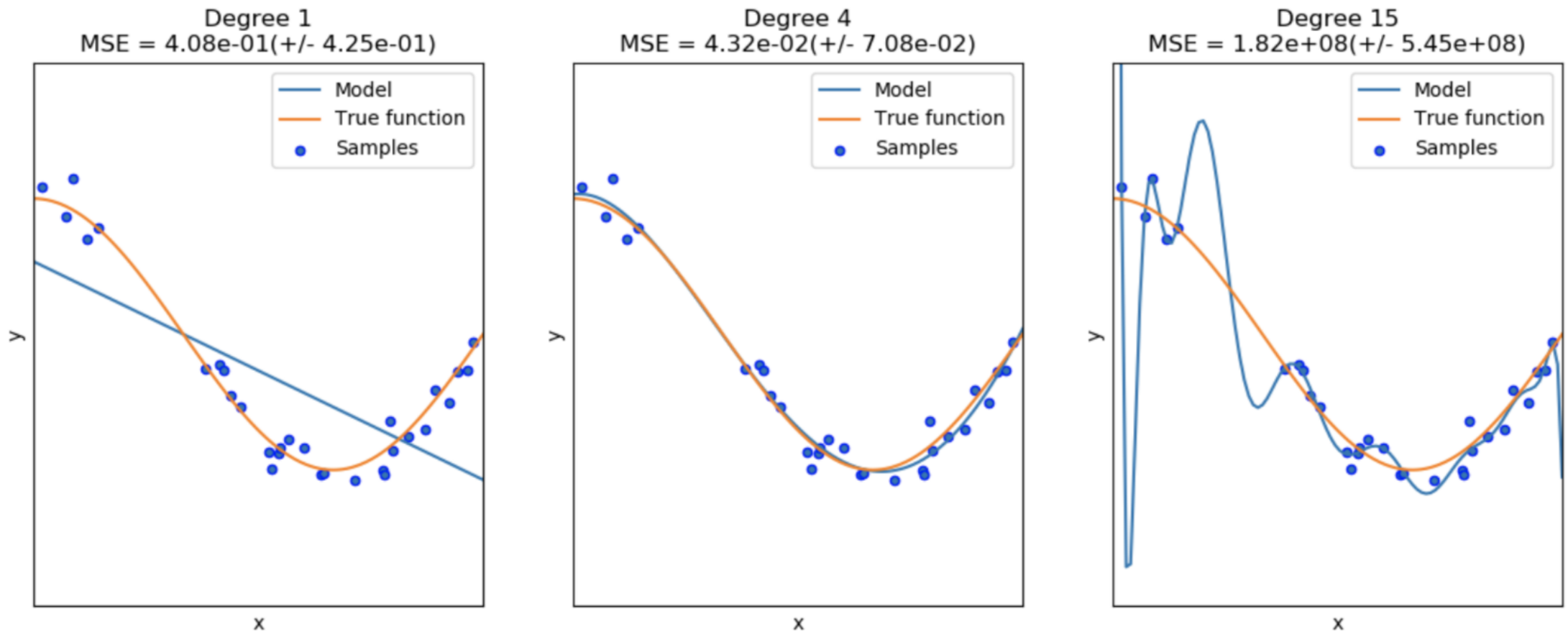


Courtesy [J. Scarlett, 2018]

Example: Polynomial regression for $x \in \mathbb{R} : \phi(x) = (1, x^2, \ldots x^p)^\top$

Designing "good" features can be difficult

Example: fitting polynomials



scikit-learn.com

Polynomials in high dimensions

- Suppose we wish to use polynomial features, but our input is higher-dimensional
- Can still use monomial features
- Example: Monomials in 2 variables, degree $= 2$

- Need $O(d^p)$ dimensions to represent (multivariate) polynomials of degree p on d features
- Example: $d = 1000, p = 2$; Need $\approx$1M dimensions
- In the following, we can see how we can efficiently implicitly operate in such high-dimensional feature spaces (i.e., without ever explicitly computing the transformation)

The "Kernel Trick"

Can we avoid using an explicit feature map?

- Express problem s.t. it only depends on inner products
- Replace inner products by kernels

$$\phi(x_i)^\top \phi(x_j) \to k(x_i, x_j)$$

Key idea: There are clever choices of $\phi(\cdot)$ ensuring that we can efficiently compute $\phi(x_i)^\top \phi(x_j)$ without ever explicitly mapping to the feature space

This "trick" is very widely applicable!

- Ridge regression (this lecture, non-parametrics part 1)
- Gaussian process regression (non-parametrics part 2)
- Support vector machine (in "dual" form) (week 9)
- nearest-neighbor methods
- ...

Example: ridge regression

$$\hat{w} = \arg\min_{w} \sum_{i=1}^{n} \left(w^\top x_i - y_i \right)^2 + \lambda \left\| w \right\|_2^2$$

$$\Rightarrow \hat{w} = \left(X^\top X + \lambda I \right)^{-1} X^\top y$$

Fundamental insight: $\hat{w}$ lives in the span of the data

$$\hat{w} = \sum_{i=1}^{n} \alpha_i x_i$$

- Handwavy proof: one can construct such a representation via (stochastic) gradient descent
- More abstract proof: follows from the "representer theorem" (not discussed here)

Reformulating ridge regression

Consider $k(x_i, x_j) = \langle \phi(x_i), \phi(x_i) \rangle$, and for now, let $\phi(x) = x$.

$$\hat{w} = \arg\min_{w} \sum_{i=1}^{n} (\underbrace{w^\top x_i}_{(a)} - y_i)^2 + \lambda \underbrace{\|w\|_2^2}_{(b)} \qquad \hat{w} = \sum_{i=1}^{n} \alpha_i x_i \qquad (1)$$

$$(a) = \sum_{j=1}^{n} \alpha_j (\underbrace{x_j^\top x_i}_{k(x_i, x_j)}) = \alpha^\top k_i$$

$$(b) = \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j (x_j^\top x_i)$$

$$K = \begin{pmatrix} k(x_1, x_1) & \dots & k(x_1, x_n) \\ \vdots & & \vdots \\ k(x_n, x_1) & \dots & k(x_n, x_n) \end{pmatrix} \quad \begin{matrix} k_1 & & k_n \end{matrix}$$

Therefore, Eq. (1) amounts to solving

$$\hat{\alpha} = \arg\min_{\alpha_{1:n}} \underbrace{\sum_{i=1}^{n} \left(\sum_{j=1}^{n} \alpha_j k(x_i, x_j) - y_i \right)^2}_{\|\alpha^\top K - y\|_2^2} + \lambda \underbrace{\sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j k(x_i, x_j)}_{\alpha^\top K \alpha}$$

"Kernelized" ridge regression

Learning: solve least squares problem

$$\hat{\alpha} = \arg\min_{\alpha} \left\| \alpha^\top K - y \right\|_2^2 + \lambda \alpha^\top K \alpha$$

Closed-form solution: $\hat{\alpha} = (K + \lambda I)^{-1} y$

Prediction: For data point x predict response y as

$$\hat{y} = \sum_{i=1}^{n} \hat{\alpha}_i k(x_i, x)$$

Example: Linear kernel: $k(x_i, x_j) = x_i^\top x_j$

$$\hat{y} = \sum_{i=1}^{n} \hat{\alpha}_i k(x_i, x) = \underbrace{\left(\sum_{i=1}^{n} \hat{\alpha}_i x_i^\top \right)}_{\hat{w}^\top} x = \hat{w}^\top x$$

How to choose kernel functions?

Can we use any function $k(\cdot, \cdot)$?

Definition: kernel functions

Let $\mathcal{X}$ be the data space. A kenerl function is a function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ satisfying

- symmetry: $k(x_i, x_j) = k(x_j, x_i)$ for any $x_i, x_j \in \mathcal{X}$.
- positive semi-definiteness: for any n, any set $S = \{x_1, \ldots, x_n\} \subset \mathcal{X}$, the kernel (Gram) matrix must be positive semi-definite.

Kernel composition rules

Suppose we have two kernels

$$k_1 = \mathcal{X} \times \mathcal{X} \to \mathbb{R} \qquad\qquad k_2 = \mathcal{X} \times \mathcal{X} \to \mathbb{R}$$

Then the following functions are valid kernels

$k(x_i, x_j) = k_1(x_i, x_j) + k_2(x_i, x_j)$

$k(x_i, x_j) = c \cdot k_1(x_i, x_j)$ for $c > 0$

$k(x_i, x_j) = k_1(x_i, x_j) \cdot k_2(x_i, x_j)$

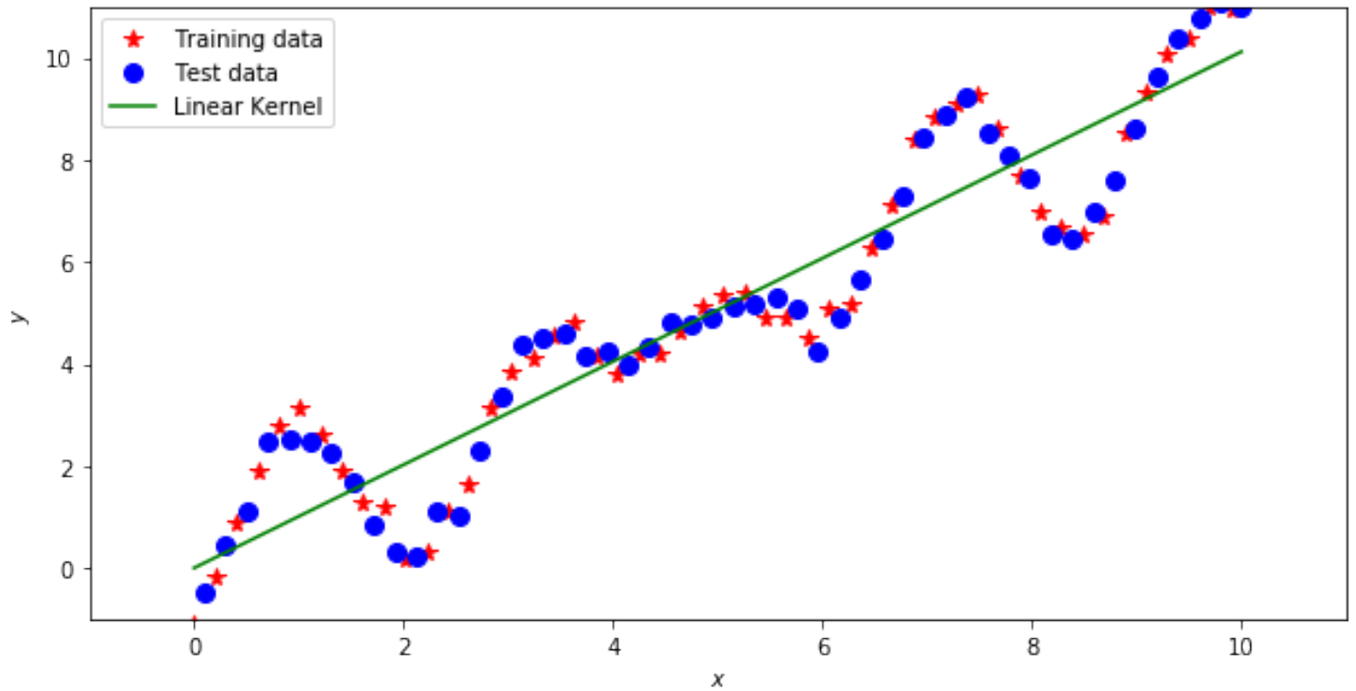
$k(x_i, x_j) = x_i^\top A x_j$ where A is positive semi-definite

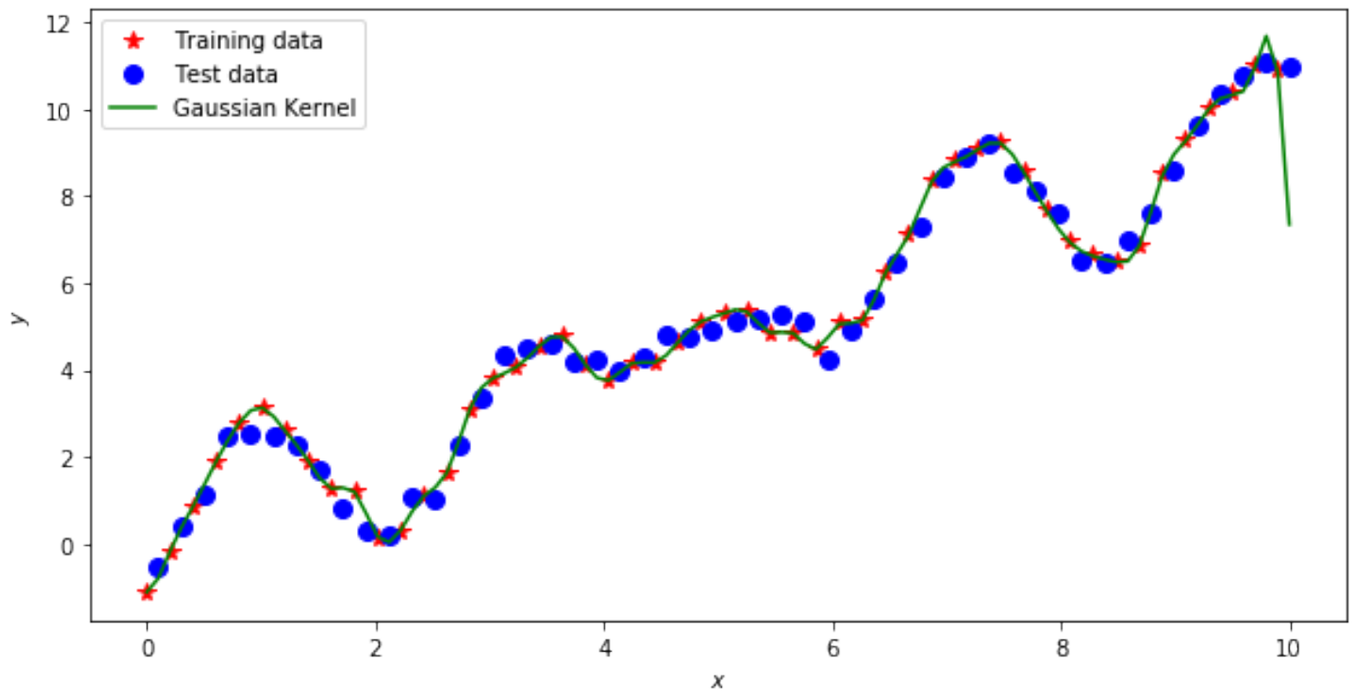
Ch 6.2: Bishop

Examples of kernels

- linear kernel: $k(x_i, x_j) = x_i^\top x_j$
- polynomial kernel: $k(x_i, x_j) = \left(x_i^\top x_j + 1\right)^d$
- Gaussian (squared exp.) kernel: $k(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|_2^2}{2h^2}\right)$
- Laplacian kernel $k(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|_1}{h}\right)$

Demonstration: Kernel Ridge Regression (KR)





https://las.inf.ethz.ch/courses/introml-f18/

Interpretation of the prediction rule

Prediction rule:

$$\hat{y}(x) = k_x \left(K + \lambda I\right)^{-1} y$$

where $k_x = (k(x_1, x), \ldots, k(x_n, x))^\top$, and $y = (y_1, \ldots, y_n)^\top$.

- $k(x_i, x_j)$ measures similarity
- For a new point x, the estimate $\hat{y}$ is a weighted sum of the $\{y_i\}_{i=1}^n$ in the data set, with higher weights when x_i is more similar to x

Parametric vs nonparametric learning

Parametric models have finite set of parameters

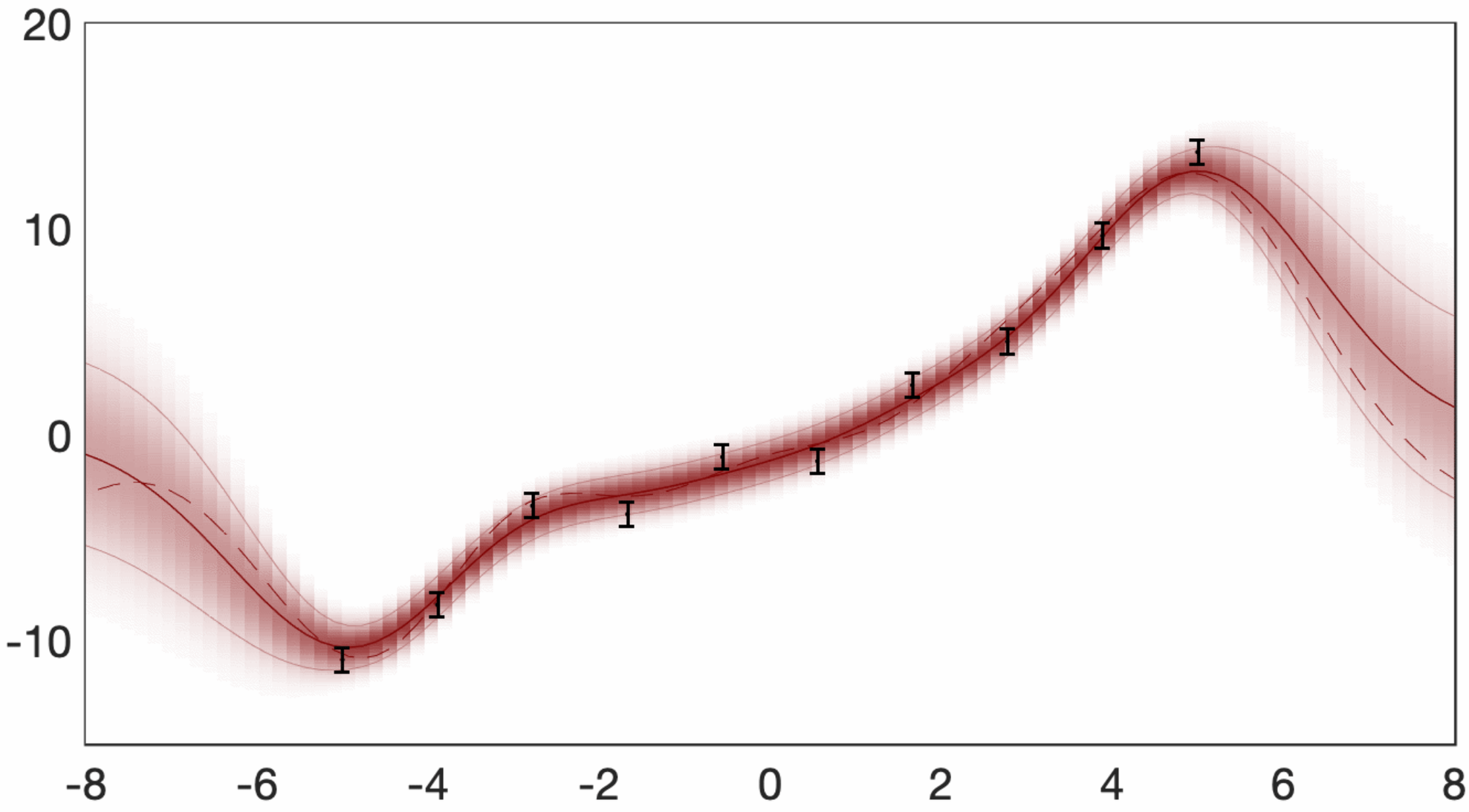
- Example: Linear regression, linear perceptron, neural networks (finite width)

Nonparametric models grow in complexity with size of the data

- Avoid introducing an explicit input-output relationship and its associated parameters (e.g., θ)
- Instead, construct model via interpolation of available samples
- Potentially much more expressive
- Example: "kernelized" linear regression

Kernels provide a principled way of deriving nonparametric models from parametric ones

Outlook: “kernel trick” + Bayesian learning



Summary

Kernel method

- Learning algorithms depending only on $\langle x_i, x_j \rangle$ can be kernelized
- Kernel allows us to implicitly work in complicated feature spaces

Kernel ridge regression

- Ridge regression depends on the data only through inner product
- Can be solved analytically or as an optimization problem

Next video: quantifying uncertainty in nonparametric learning

- Bayesian ridge reg. + kernel trick = Gaussian process regression

Reading materials

- Ch 6: C.M. Bishop, Pattern Recognition and Machine Learning, Springer, 2006
- M. Jordan: Kernel methods: an overview, 2007 `https://people.eecs.berkeley.edu/~jordan/kernels/0521813972c02_p25-46.pdf`