CPSC 340: Machine Learning and Data Mining

Regularization

Admin

- Assignment 2:
  - 1 late day to hand it in today, 2 for tomorrow, 3 for Wednesday.
- Assignment 3 is out.
  - Due next Friday (right before the break)
- Tutorials this week: assignment 3 practice
- Real-time feedback system is back up

Last Time: Normal Equations and Change of Basis

- Last time we derived normal equations:

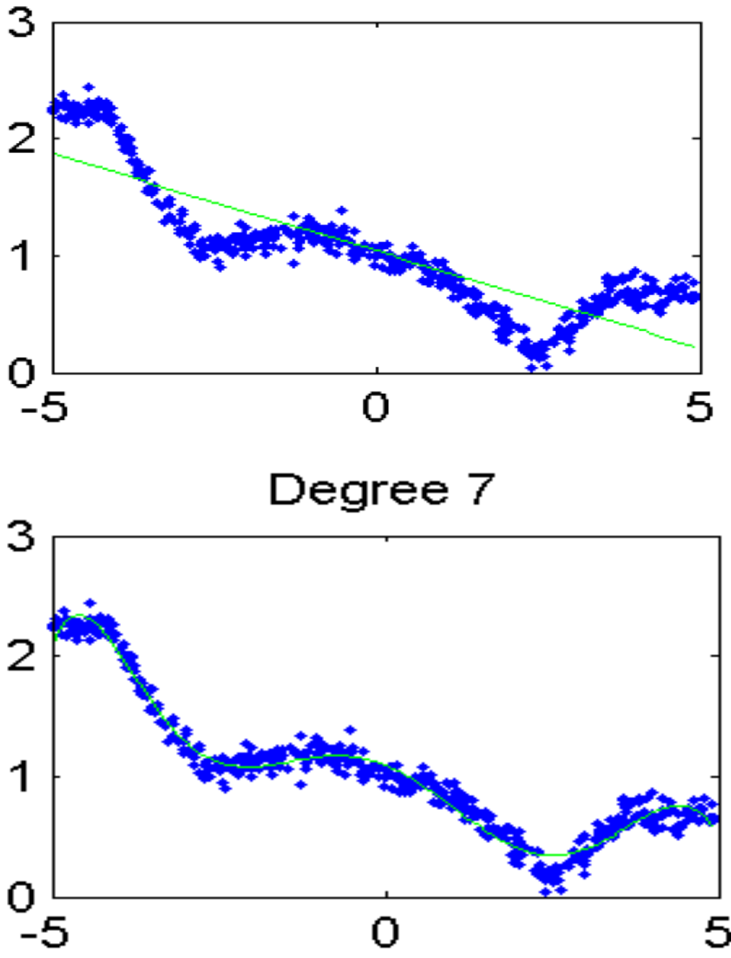
$$X^T X w = X^T y$$

  - Solutions 'w' minimize squared error in linear model.
- We also discussed change of basis:
  - E.g., polynomial basis:

Replace $X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$ with $Z = \begin{bmatrix} 1 & x_1 & (x_1)^2 & \cdots & (x_1)^p \\ 1 & x_2 & (x_2)^2 & \cdots & (x_2)^p \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & (x_n)^2 & \cdots & (x_n)^p \end{bmatrix}$

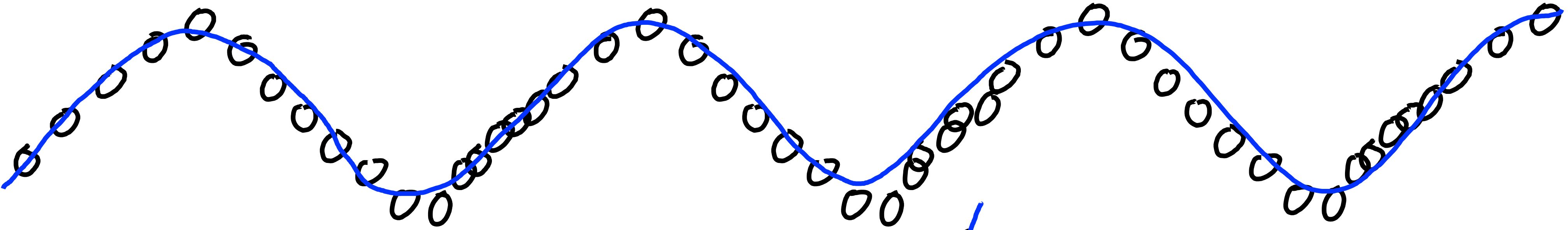
  - Let's you fit non-linear models with linear regression.

$$y_i = w^T z_i = w_0 + w_1 x_i + w_2 x_i^2 + w_3 x_i^3 + \cdots + w_p x_i^p$$



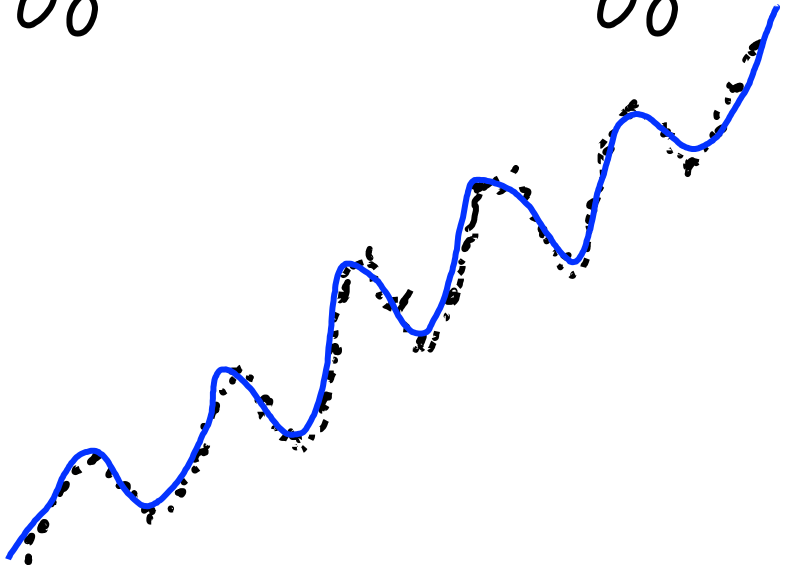
Parametric vs. Non-Parametric Bases

- Polynomials are not the only possible bases:
  - Exponentials, logarithms, trigonometric functions, etc.
  - The right basis will vastly improve performance.



For periodic data we might use

$$Z = \begin{bmatrix} \sin(x_1) \\ \sin(x_2) \\ \vdots \\ \sin(x_n) \end{bmatrix}$$



You can have different types of bases

$$Z = \begin{array}{c} \begin{bmatrix} x_1 & \sin(6x_1) \\ x_2 & \sin(6x_2) \\ \vdots & \vdots \\ x_n & \sin(6x_n) \end{bmatrix} \\ \begin{array}{cc} \underbrace{\quad}_{\text{linear}} & \underbrace{\qquad\quad}_{\text{periodic}} \end{array} \end{array}$$

Parametric vs. Non-Parametric Bases

- Polynomials are not the only possible bases:
  - Exponentials, logarithms, trigonometric functions, etc.
  - The right basis will vastly improve performance.
  - But the right basis may not be obvious.
- What happens if we use the wrong basis?
  - As ‘n’ increases, we can fit ‘w’ more accurately.
  - But eventually more data doesn’t help if basis isn’t “flexible” enough.
- Alternative is non-parametric bases:
  - Size of basis (number of features) grows with ‘n’.
  - Model gets more complicated as you get more data.
  - You can model very complicated functions where you don’t know the right basis.

Non-Parametric Basis: RBFs

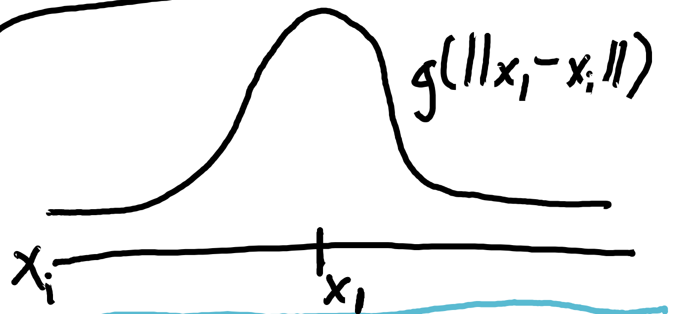
- Radial basis functions (RBFs):
  - Non-parametric bases that depend on distances to training points.

Replace $X = \underbrace{\begin{bmatrix} \\ \\ \\ \end{bmatrix}}_{d} \Big\}n$ by $Z = \underbrace{\begin{bmatrix} g(\|x_1 - x_1\|) & g(\|x_1 - x_2\|) & \cdots & g(\|x_1 - x_n\|) \\ g(\|x_2 - x_1\|) & g(\|x_2 - x_2\|) & \cdots & g(\|x_2 - x_n\|) \\ \vdots & & \ddots & \vdots \\ g(\|x_n - x_1\|) & g(\|x_n - x_2\|) & \cdots & g(\|x_n - x_n\|) \end{bmatrix}}_{n} \Big\}n$

  - Most common 'g' is Gaussian RBF:

$$g(\alpha) = \exp\left(-\frac{\alpha^2}{2\sigma^2}\right)$$

Parameter



    - Variance σ^2 controls influence of nearby points.
    - This affects fundamental trade-off (set it using a validation set).

Do we need $\sigma\sqrt{2\pi}$?
–No because $w^T x_i = (\frac{1}{\beta} w)^T (\beta x_i)$

Non-Parametric Basis: RBFs

- Radial basis functions (RBFs):
  - Non-parametric bases that depend on distances to training points.

Replace $X = \underbrace{\begin{bmatrix} \quad \end{bmatrix}}_{d} \Big\}n$ by $Z = \underbrace{\begin{bmatrix} g(\|x_1 - x_1\|) & g(\|x_1 - x_2\|) & \cdots & g(\|x_1 - x_n\|) \\ g(\|x_2 - x_1\|) & g(\|x_2 - x_2\|) & \cdots & g(\|x_2 - x_n\|) \\ \vdots & & \ddots & \vdots \\ g(\|x_n - x_1\|) & g(\|x_n - x_2\|) & \cdots & g(\|x_n - x_n\|) \end{bmatrix}}_{n} \Big\}n$

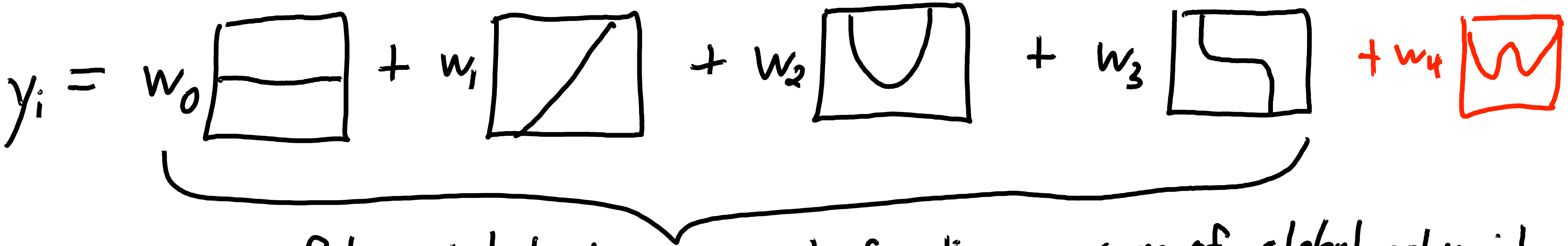
To make predictions on $\hat{X} = \underbrace{\begin{bmatrix} \quad \end{bmatrix}}_{d} \Big\}t$ use $\hat{Z} = \underbrace{\begin{bmatrix} \quad g(\|\hat{x}_i - x_j\|) \quad \end{bmatrix}}_{n} \Big\}t$

I.e., $y_i = w^T \hat{z}_i$ or $y = \hat{Z}w$

Number of "features" is number of training examples.

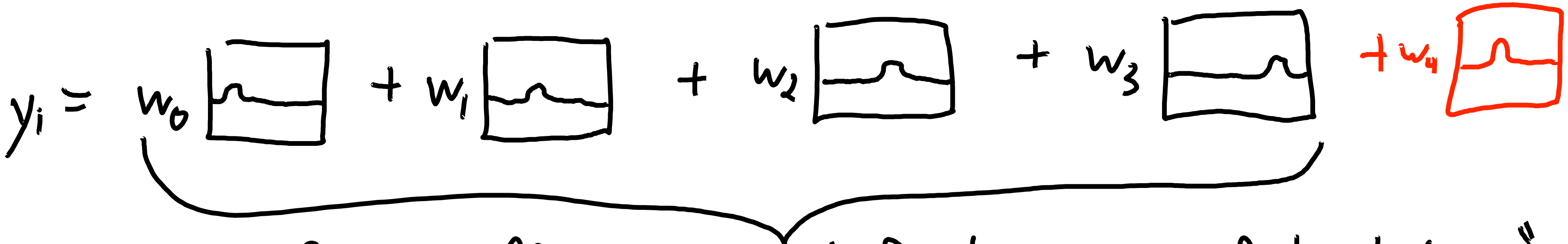
Non-Parametric Basis: RBFs

Cubic basis:



Polynomial basis represents function as sum of <u>global</u> polynomials.

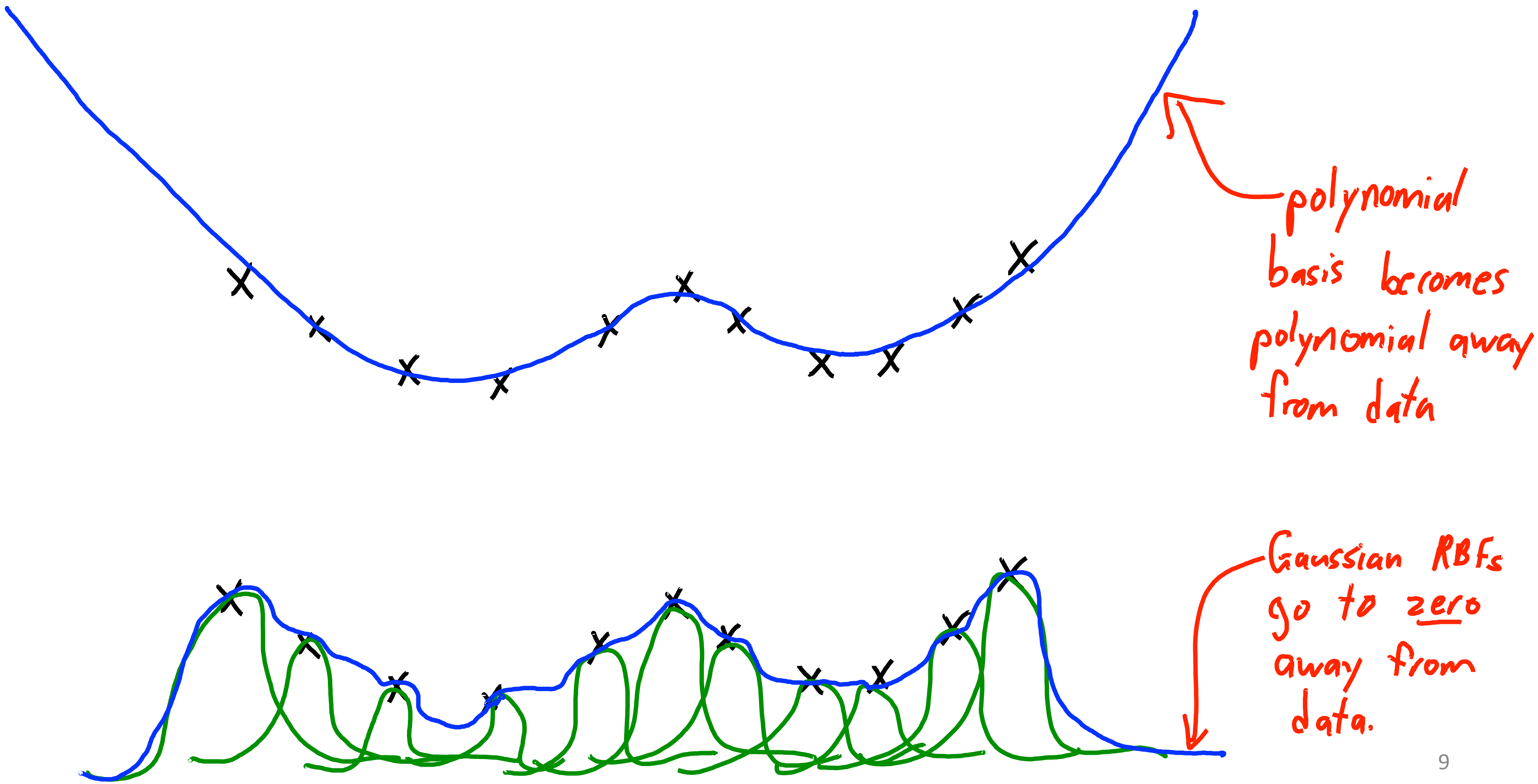
Gaussian RBFs:



Gaussian RBFs represent function as sum of <u>local</u> "bumps"

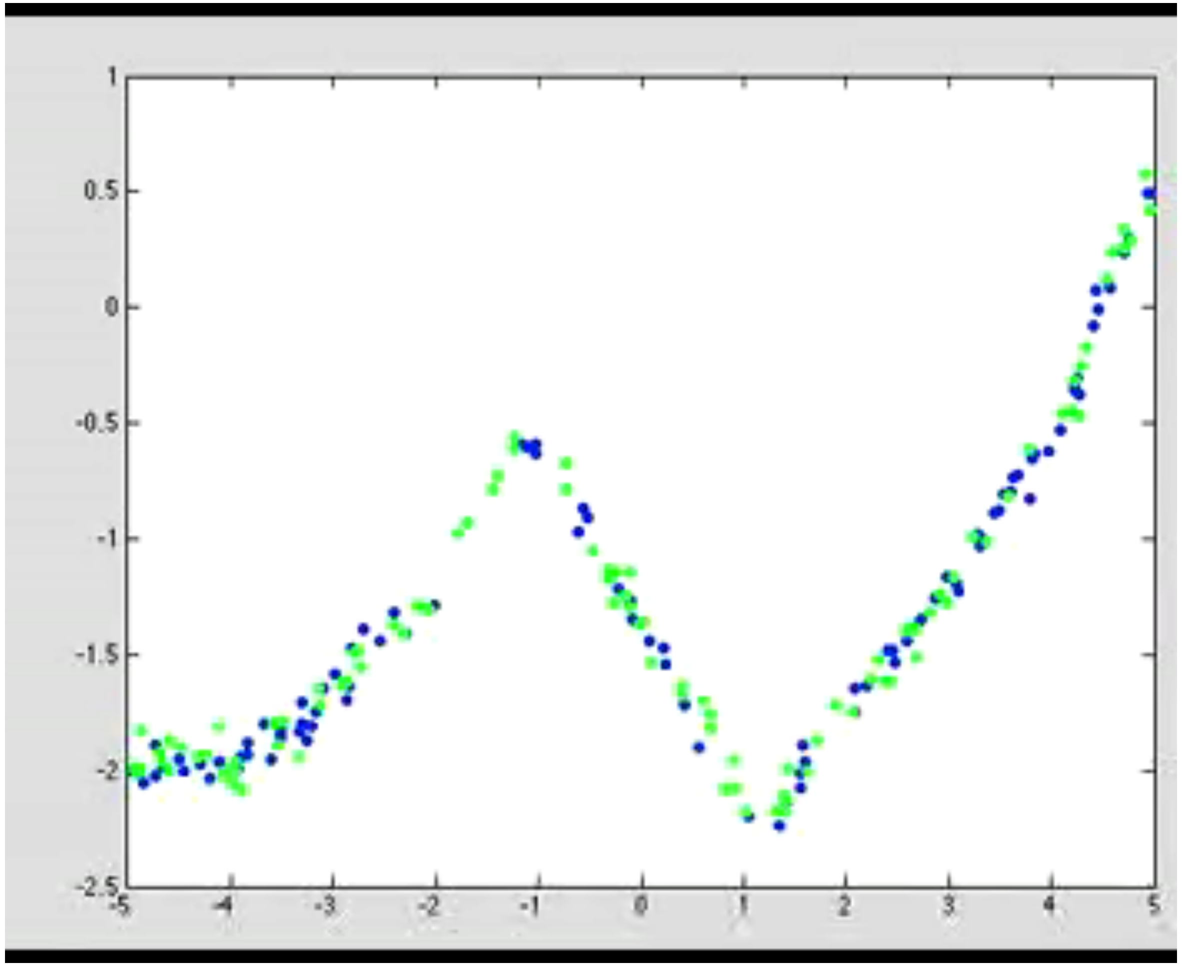
- Gaussian RBFs are universal approximators (compact subets of $\mathbb{R}^d$)
  - Can approximate any continuous function to arbitrary precision.
  - Achieve irreducible error as 'n' goes to infinity.

Interpolation vs. Extrapolation



Non-Parametric Basis: RBFs

- Least squares with Gaussian RBFs for different σ values:



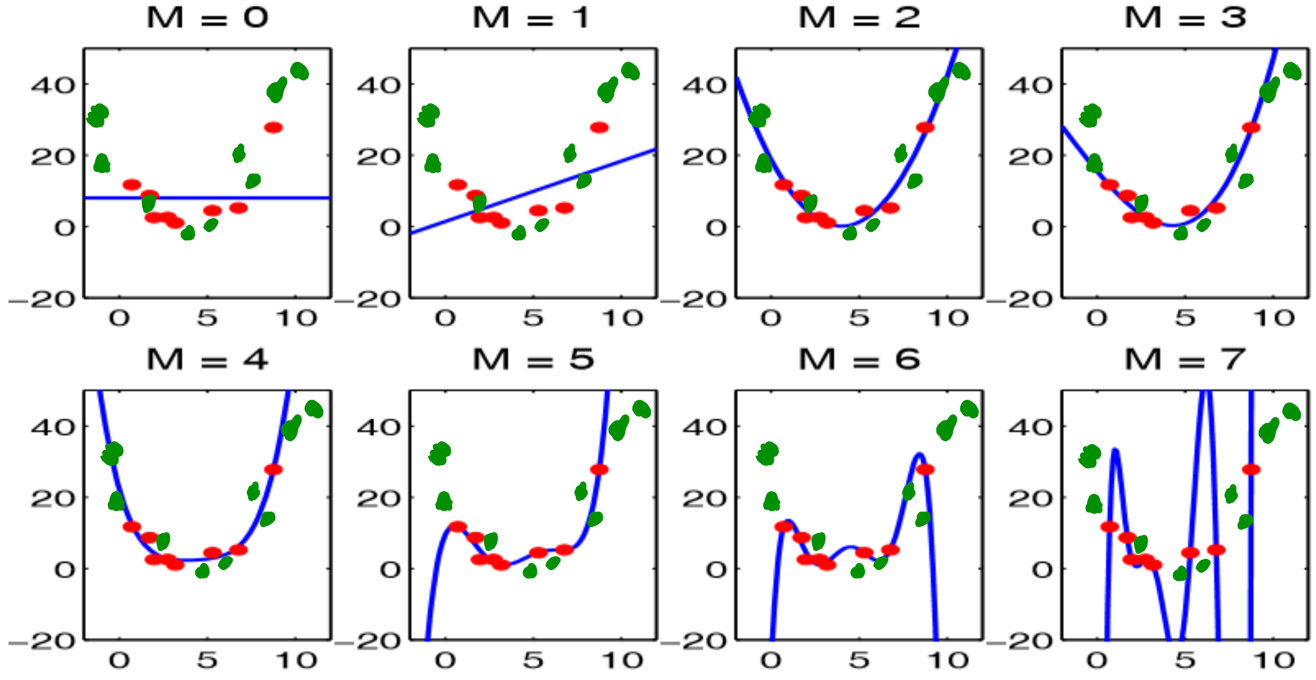
Could add <u>bias</u> and linear basis:

$$Z = \left[\begin{array}{c|c|c} \underbrace{\begin{matrix}1\\1\\1\\\vdots\\1\end{matrix}}_{1} & \underbrace{\begin{matrix}\text{—}\, x_1 \,\text{—}\\ \text{—}\, x_2 \,\text{—}\\ \text{—}\, x_3 \,\text{—}\\ \vdots \\ \text{—}\, x_n \,\text{—}\end{matrix}}_{d} & \underbrace{\begin{matrix} g(\|x_1 - x_1\|) & \cdots & g(\|x_1 - x_n\|) \\ \vdots & \ddots & \vdots \\ g(\|x_n - x_1\|) & \cdots & g(\|x_n - x_n\|)\end{matrix}}_{n} \end{array}\right]$$

This reverts to <u>linear regression</u> instead of 0 away from data.

Last Time: Polynomial Degree and Training vs. Testing

- As the polynomial degree increases, the training error goes down.
- But training error becomes worse approximation test error.



- Same effect as we decrease variance in Gaussian RBF.
- But what if we need a complicated model?

Controlling Complexity

- Usually "true" mapping from x_i to y_i is complex.
  - Might need high-degree polynomial or small σ^2 in RBFs.
- But complex models can overfit.
- So what do we do???

- There are many possible answers:
  - Model averaging: average over multiple models to decrease variance.
  - Regularization: add a penalty on the complexity of the model.

L2-Regularization

"lambda"

- Standard regularization strategy is L2-regularization:

$$f(w) = \frac{1}{2}\|Xw - y\|^2 + \frac{\lambda}{2}\|w\|^2 \quad \text{or} \quad f(w) = \frac{1}{2}\sum_{i=1}^{n}(w^Tx_i - y_i)^2 + \frac{\lambda}{2}\sum_{j=1}^{d} w_j^2$$

- Intuition: large w_j tend to lead to overfitting (cancel each other).
- So minimize squared error plus penalty on L2-norm of 'w'.
  - This objective balances getting low error vs. having small slope 'w'.
    - Training error will increase *because you're no longer minimizing it*.
    - But reduces overfitting.
  - Regularization parameter $\lambda > 0$ controls "strength" of regularization.
    - Large λ puts large penalty on slope.
    - There is such a thing as "too much" regularization: $\lambda \rightarrow \infty$ will cause w=0

L2-Regularization

- Standard regularization strategy is L2-regularization:

$$f(w) = \frac{1}{2}\|Xw - y\|^2 + \frac{\lambda}{2}\|w\|^2 \quad \text{or} \quad f(w) = \frac{1}{2}\sum_{i=1}^{n}(w^T x_i - y_i)^2 + \frac{\lambda}{2}\sum_{j=1}^{d} w_j^2$$

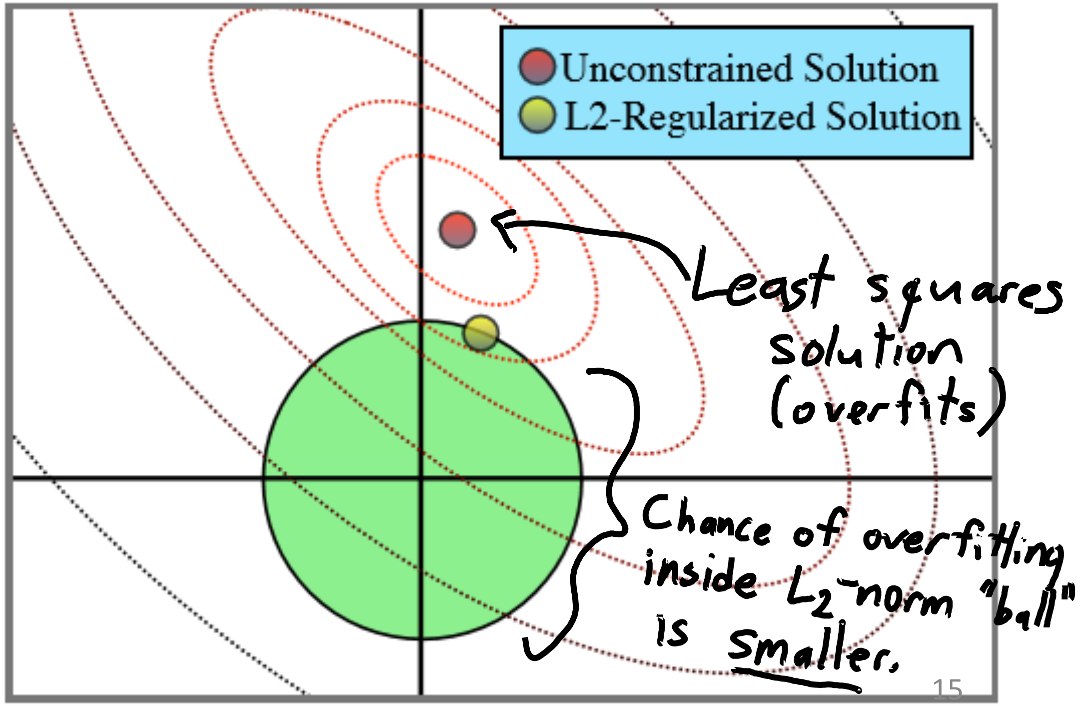
- In terms of fundamental trade-off:
  - Regularization increases training error.
  - Regularization makes training error better approximation of test error.
- How should you choose λ?
  - Theory: as ‘n’ grows λ should be in the range O(1) to O($n^{1/2}$).
  - Practice: optimize validation set or cross-validation error.
    - This almost always decreases the test error.

L2-Regularization

- Standard regularization strategy is L2-regularization:

$$f(w) = \frac{1}{2}\|Xw - y\|^2 + \frac{\lambda}{2}\|w\|^2 \quad \text{or} \quad f(w) = \frac{1}{2}\sum_{i=1}^{n}(w^Tx_i - y_i)^2 + \frac{\lambda}{2}\sum_{j=1}^{d} w_j^2$$

- Equivalent to minimizing squared error with L2-norm constraint:
- Connection to Occam's razor
  - Small values in 'w' are "simpler" models
  - L2-regularization favors small 'w'
- Regularization is a way of incorporating prior knowledge



Why use L2-Regularization?

- Almost always decreases test error
- Intuition: try to make the objective function reflect test error
  - The original objective function was just *training error*
  - Create an optimization problem that you actually want to solve
- But here are 6 more reasons (for linear regression):
  1. Solution ‘w’ is unique.
  2. $X^T X$ does not need to be invertible.
  3. Less sensitive to changes in X or y (related to uniqueness).
  4. Makes algorithms for computing ‘w’ converge faster.
  5. Stein’s paradox: if $d \geq 3$, ‘shrinking’ moves us closer to ‘true’ w on average.
     - In other words, it’s a good idea even if the prior knowledge is wrong (!!) – see bonus slides
  6. Worst case: just set λ small and get the same performance.

RBFs, Regularization, and Validation

- A model that is hard to beat:
  - RBF basis with L2-regularization and cross-validation to choose σ and λ.
  - Flexible non-parametric basis, magic of regularization, and tuning for test error!

Example:

Find regularized value of 'w' for particular λ and σ by

minimizing $f(w) = \frac{1}{2}\|Zw - y\|^2 + \frac{\lambda}{2}\|w\|^2$

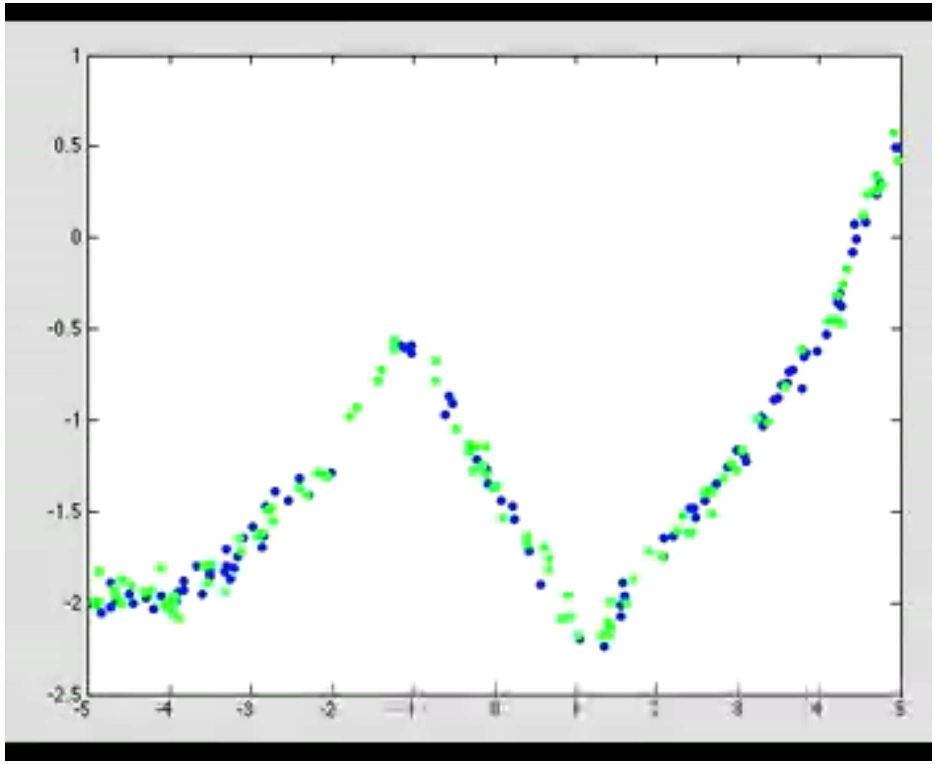
RBF basis with variance σ

And choose λ and σ to minimize

$\frac{1}{2}\|\hat{Z}\hat{w} - \hat{y}\|^2$

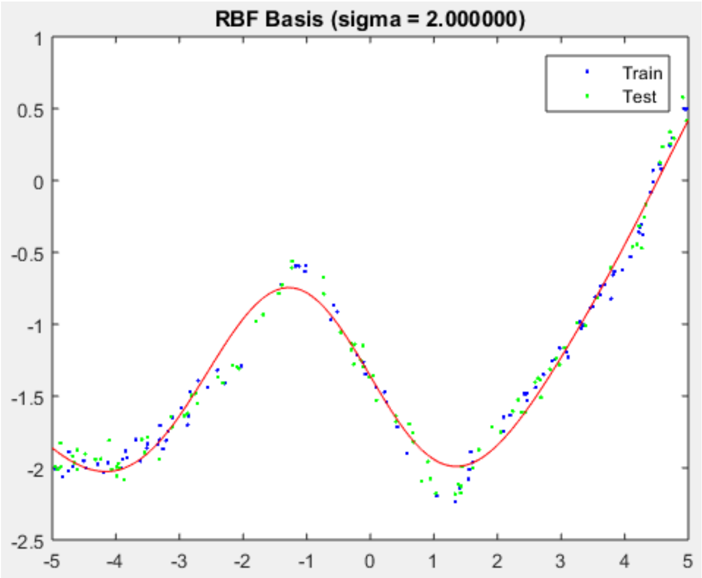
Validation set

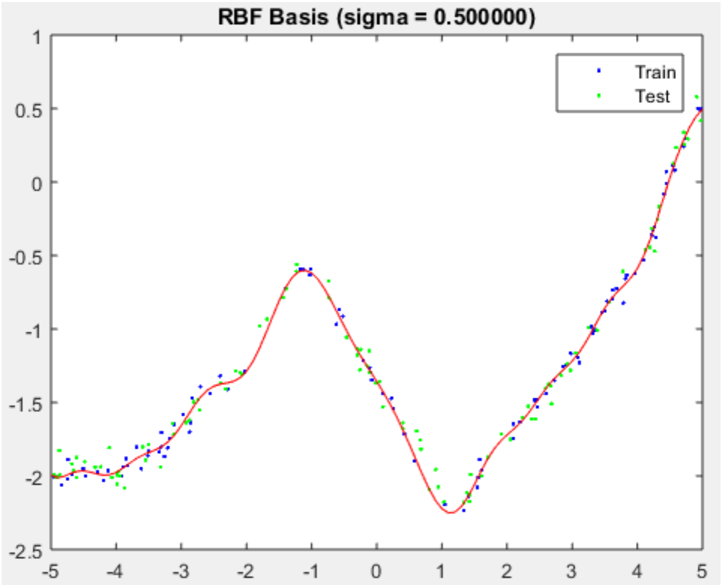
Regularized value of w

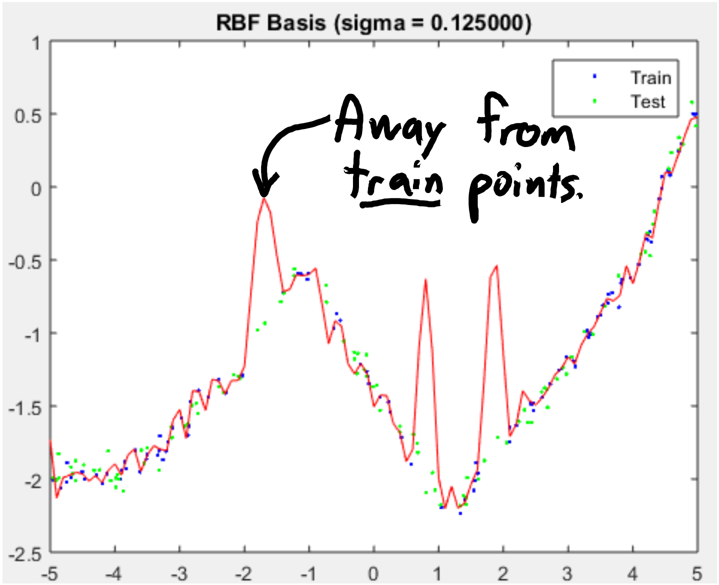


RBFs, Regularization, and Validation

- A model that is hard to beat:
  - RBF basis with L2-regularization and cross-validation to choose σ and λ.
  - Flexible non-parametric basis, magic of regularization, and tuning for test error!







  - Can add bias or linear/poly basis to do better away from data.
  - Expensive at test time: need distance to all training examples.

Summary

- Radial basis functions:
  - Non-parametric bases that can model any function.
- Regularization:
  - Adding a penalty on model complexity.
  - Improves test error because it is magic.
- L2-regularization: penalty on L2-norm of regression weights 'w'.

- Next time:
  - Going downhill…