Plan

Cool stuff

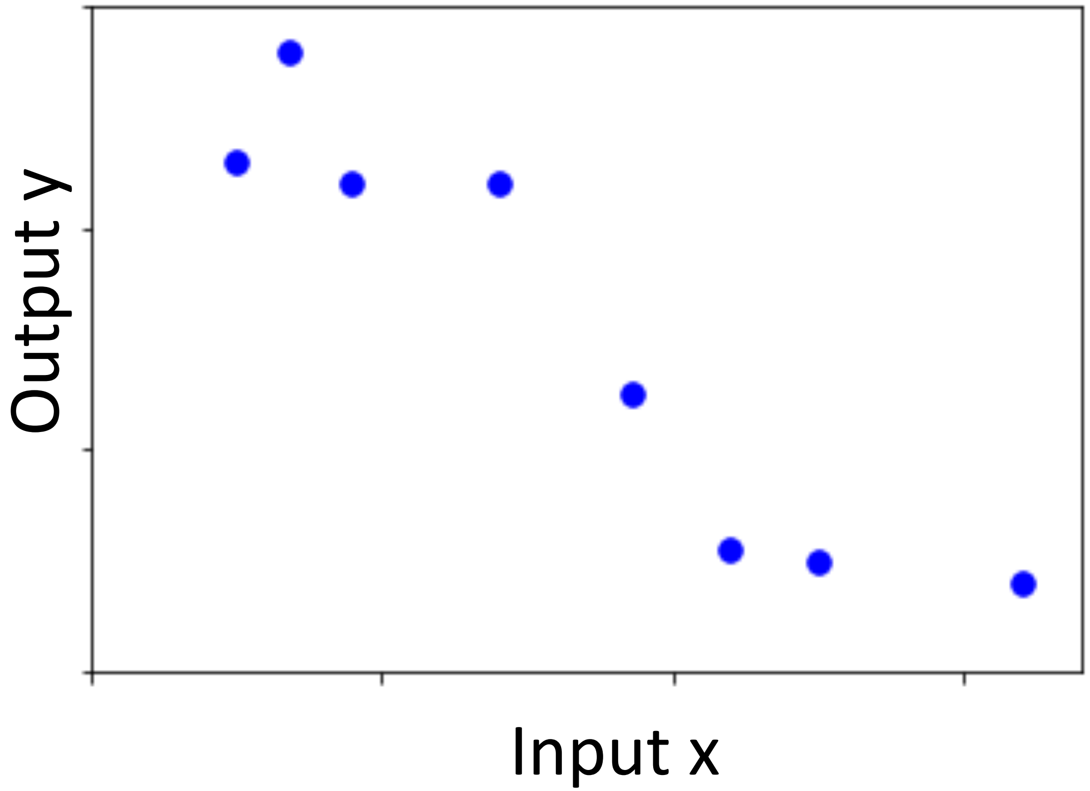
- Expectation-Maximization algorithm
  - Gaussian mixture models for clustering
- Kernels
  - Linear regression
  - Support vector machines
- Duality
  - Support vector machines

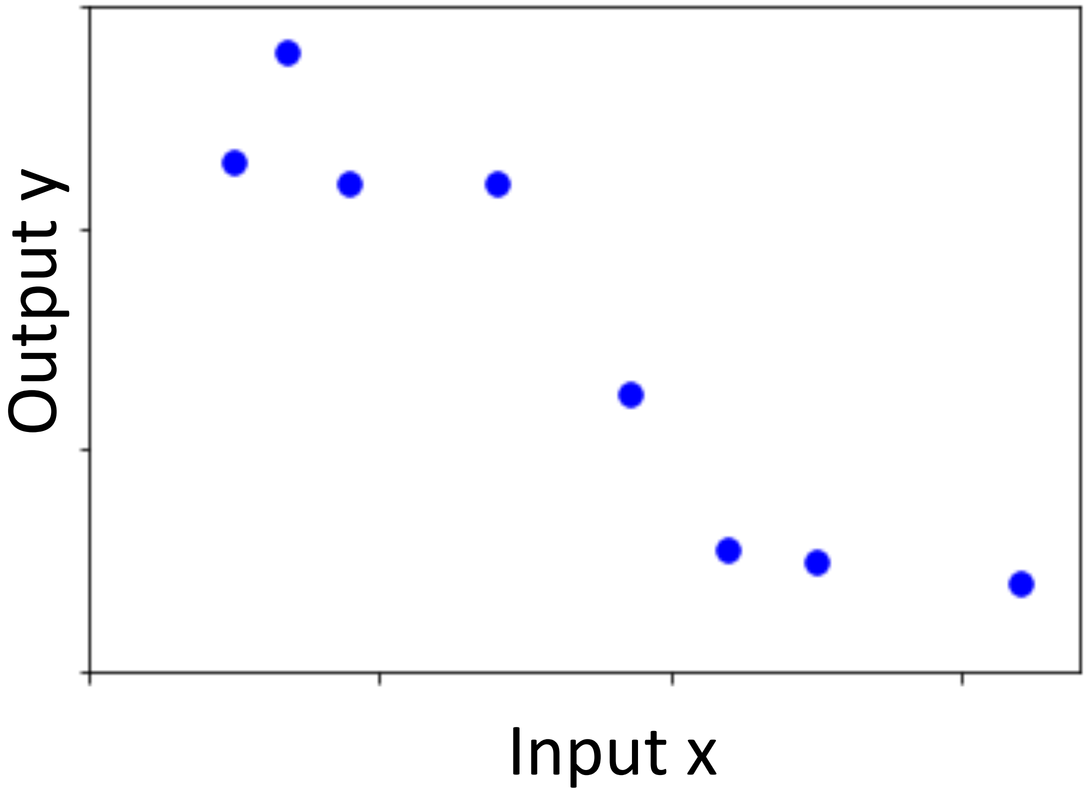
10-315
Introduction to ML

Nonparametric Regression and Kernels

Instructor: Pat Virtue

Parametric vs Nonparametric Regression





Parametric vs Nonparametric

Two different definitions

Statistics

A nonparametric model does not follow a specific distribution (thus doesn’t have parameters that define that distribution)

Machine learning

The number of parameters in a nonparametric model scales with the number of training data points

Parametric vs Nonparametric

Which models are nonparametric?

Statistics

does not follow a
specific distribution

Machine learning

number of parameters
scales with training

Linear regression

Logistic regression

Neural nets

Naïve Bayes

Discriminant analysis

K-nearest neighbor

Decision trees

Poll 1

Are decision trees parametric or non-parametric?

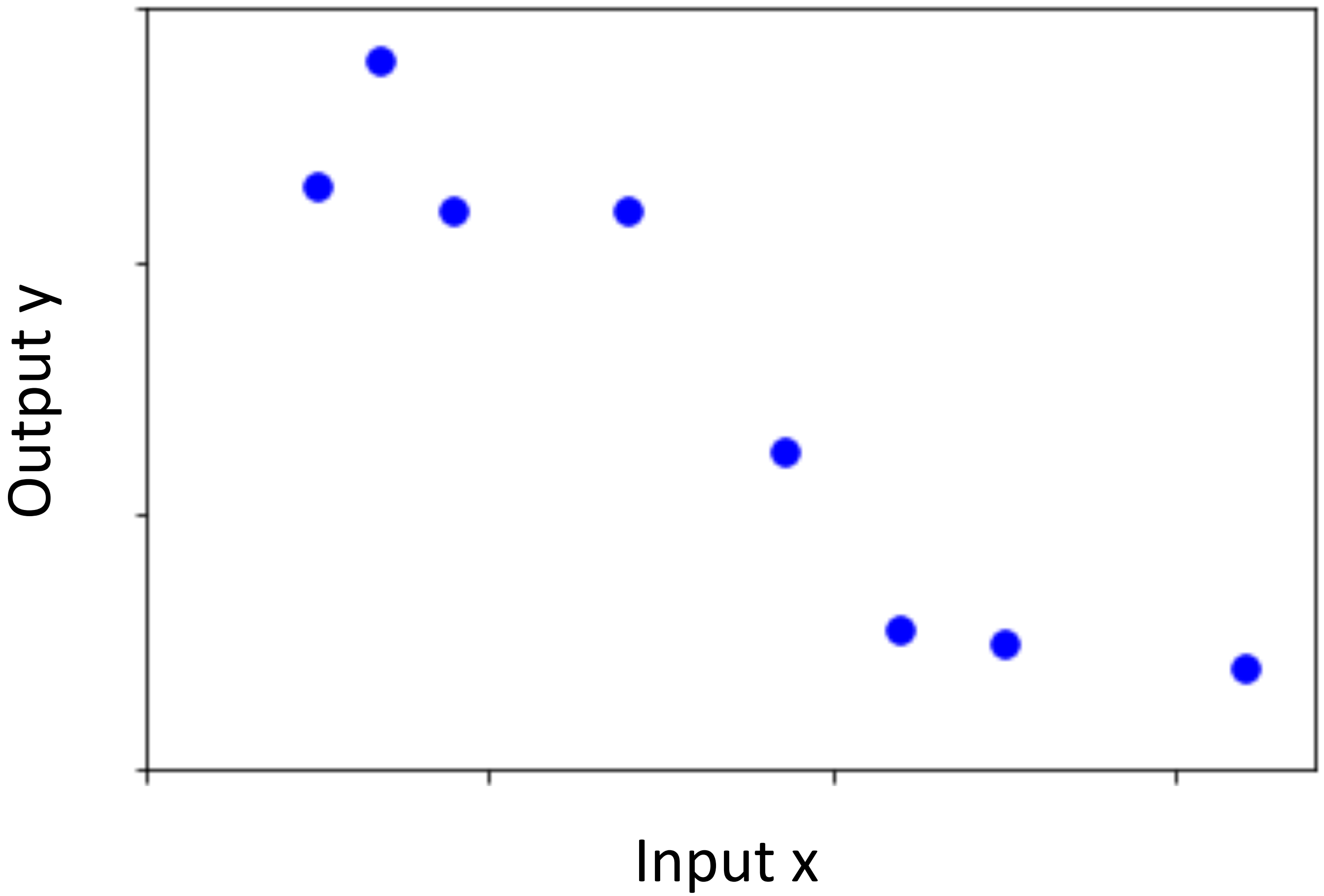
A.

B.

C.

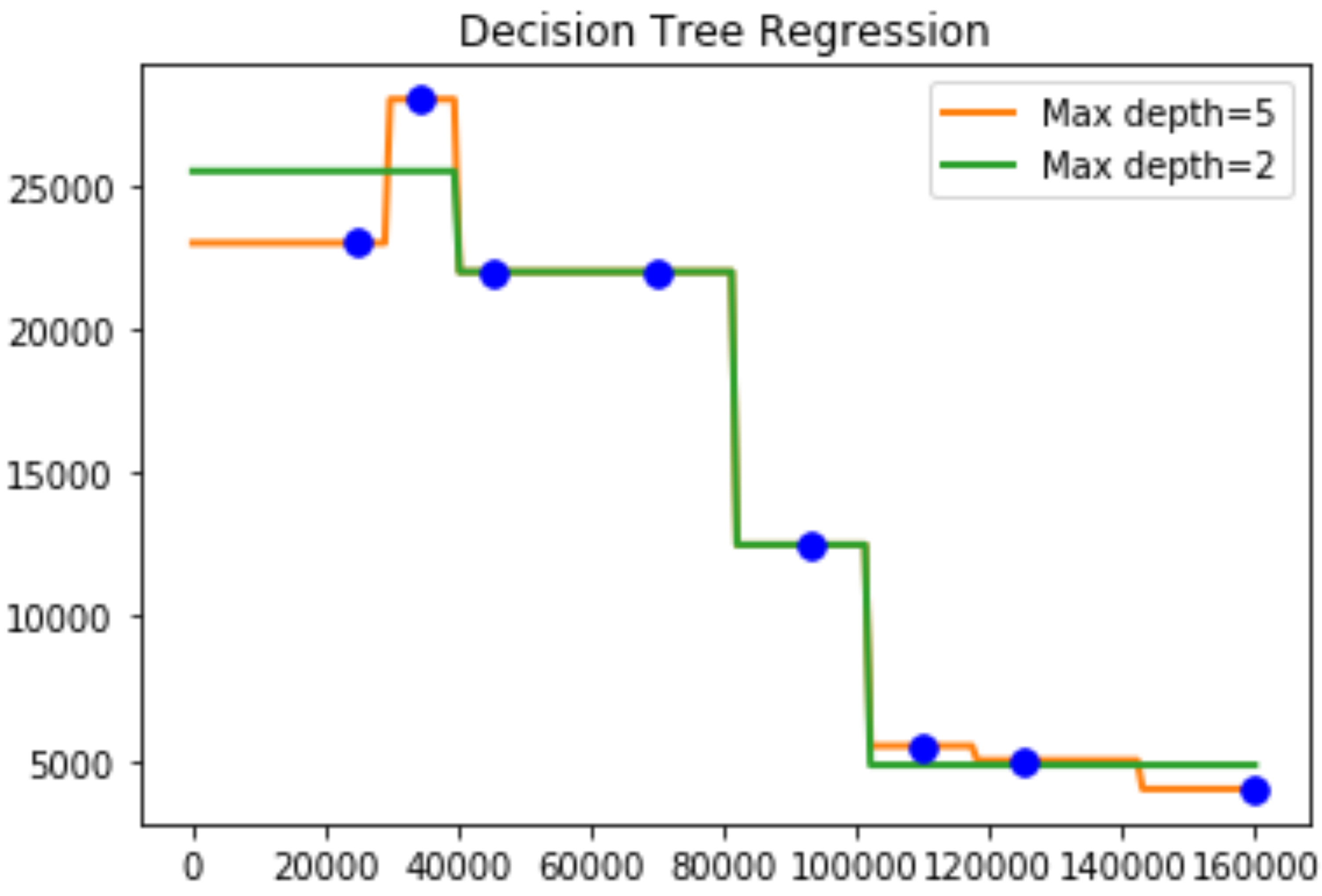
Nonparametric Regression

Decision Trees



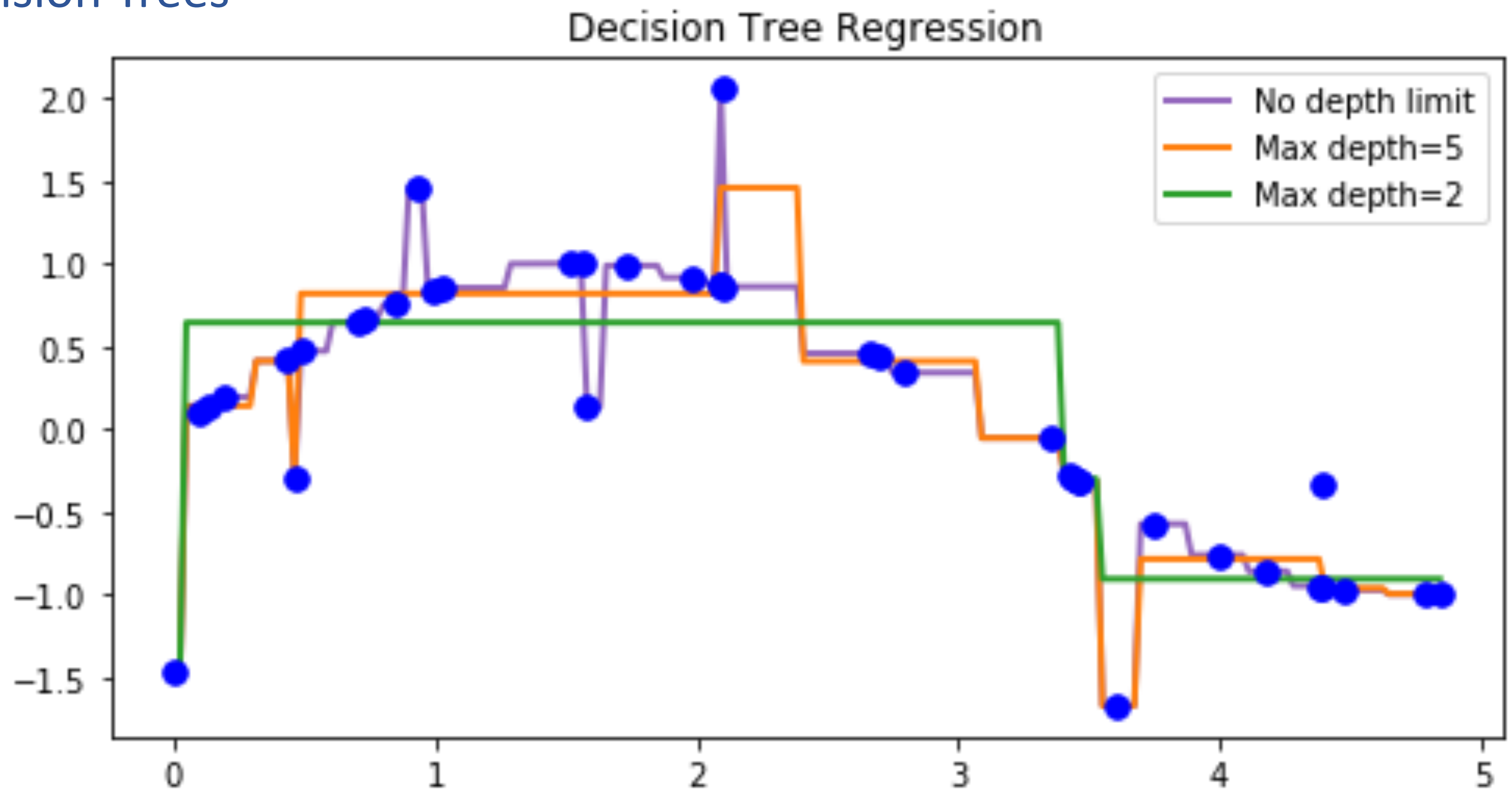
Nonparametric Regression

Decision Trees



Nonparametric Regression

Decision Trees



Poll 1

Are decision trees parametric or non-parametric?

A.

B.

C.

Poll 1

Are decision trees parametric or non-parametric?

It depends :)

- If no limits on depth or reuse of attributes, then non-parametric
  - Model complexity will grow with data
- If pruned/limited to fix size
  - Parametric
- If attributes only used once
  - Parametric; model complexity is limited by number of features

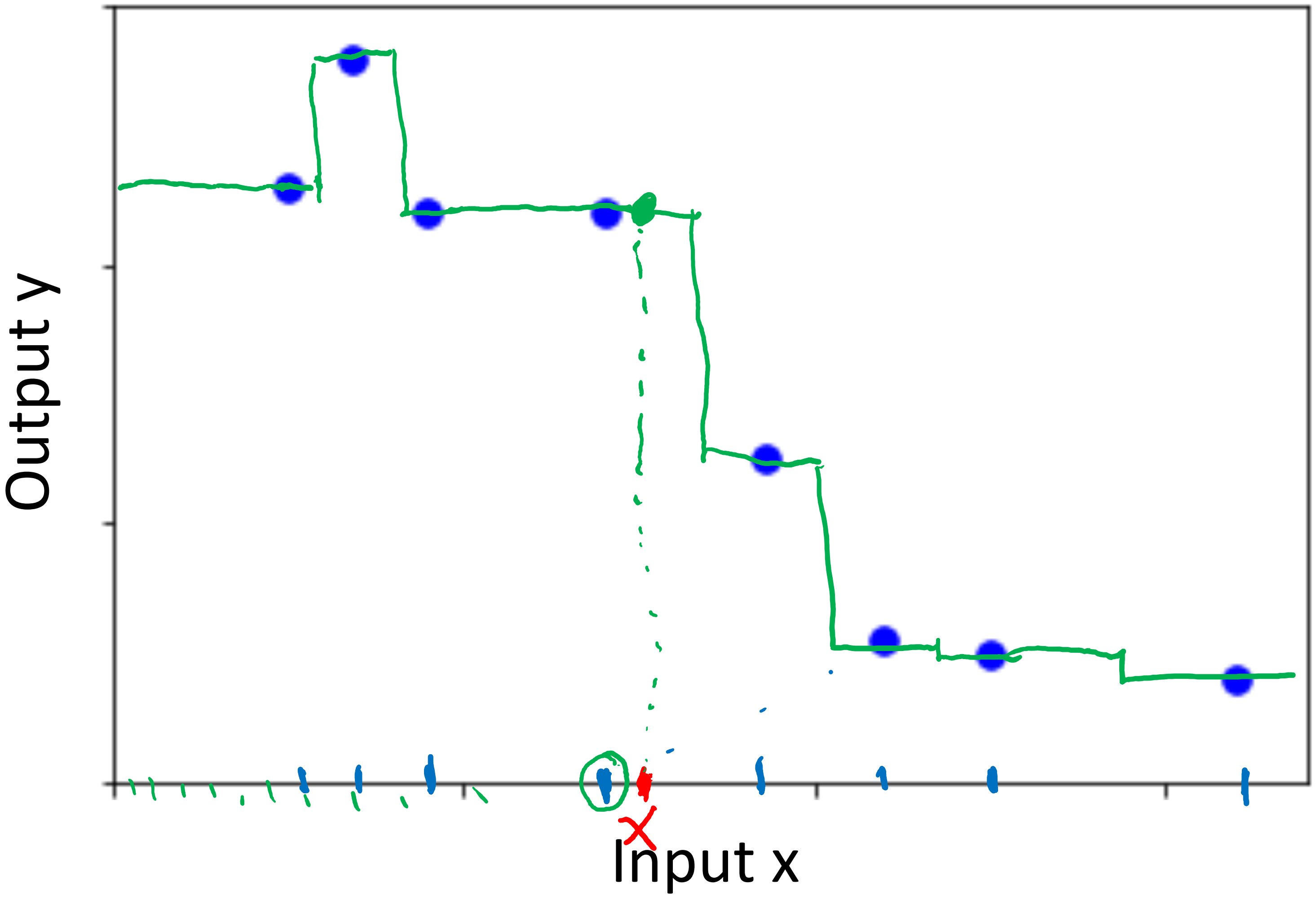
Trade-offs

- Non-parametric methods have very powerful representation capabilities
- But
  - Easily overfit
  - Can take up memory proportional to training size too

Nonparametric Regression

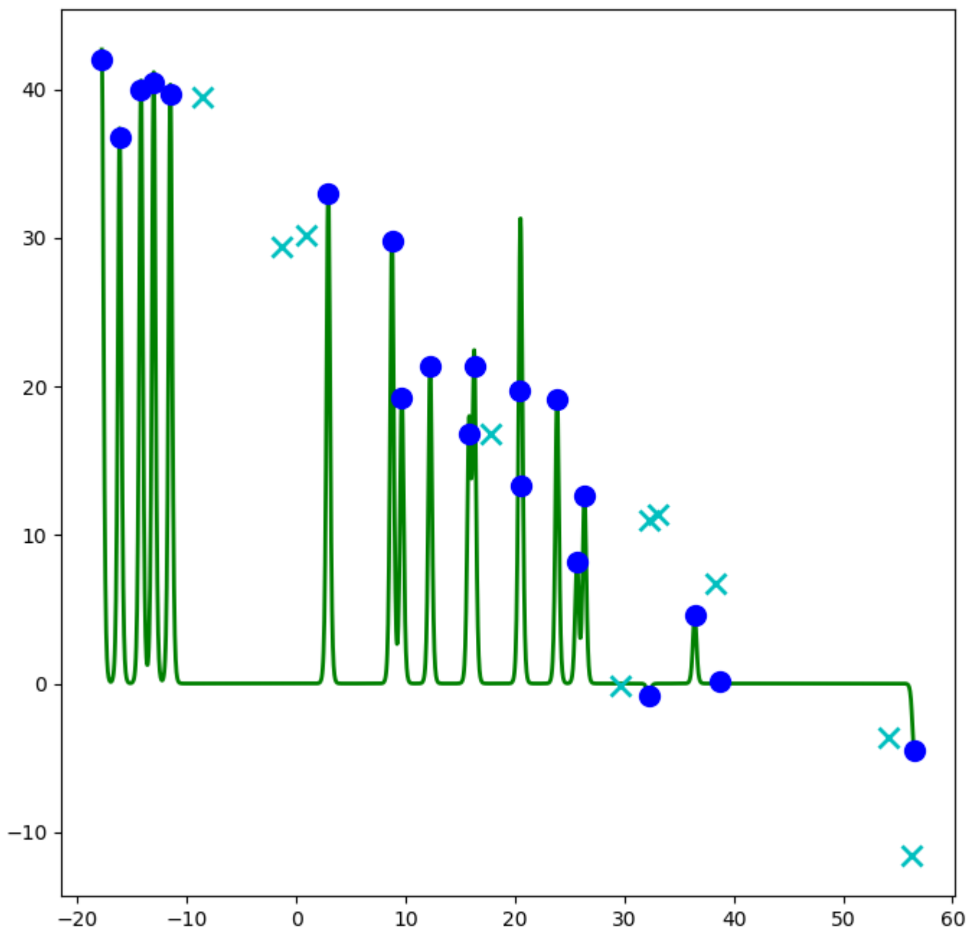
$\hat{y} = h(x)$

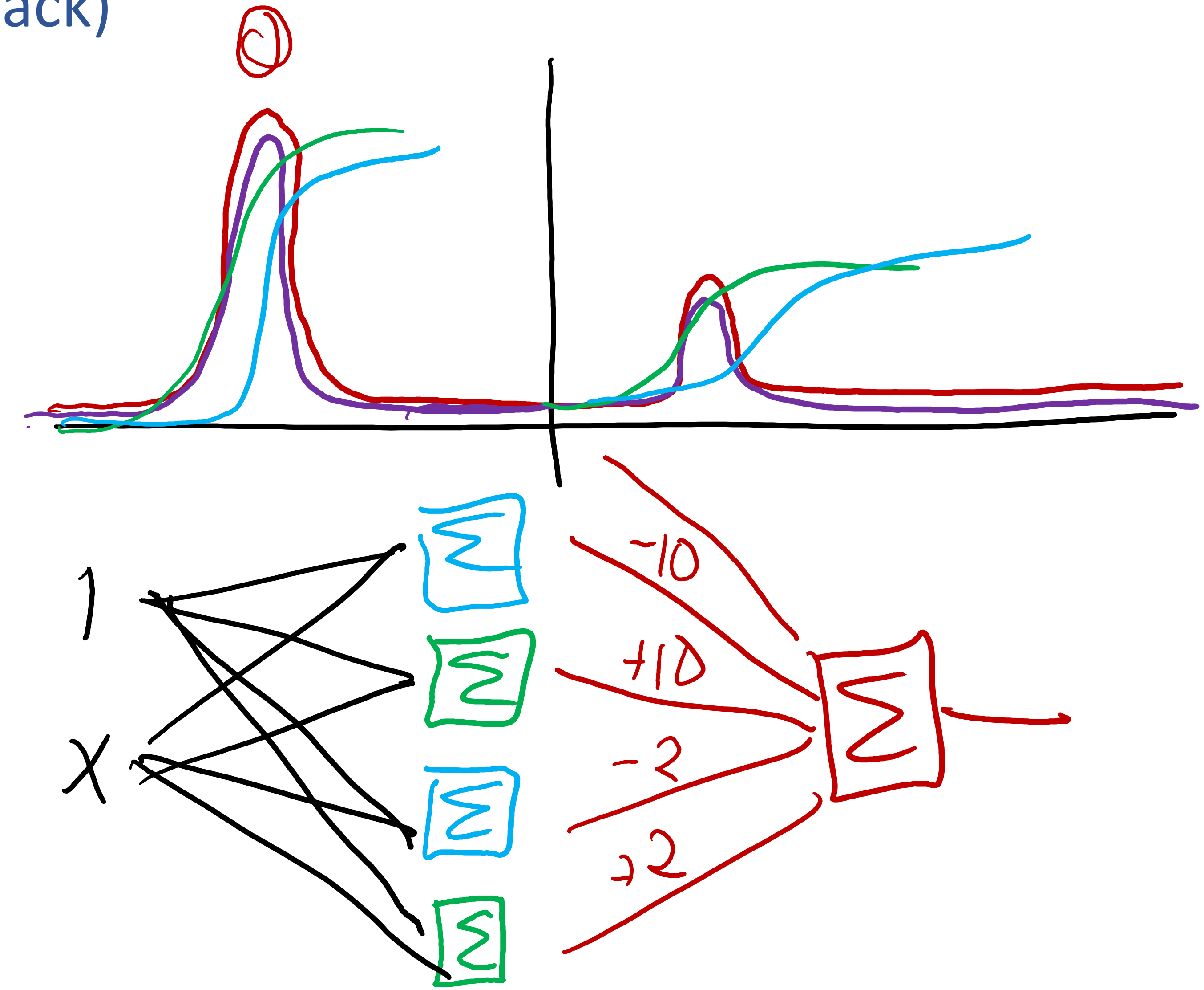
Nearest Neighbor



Nonparametric Regression

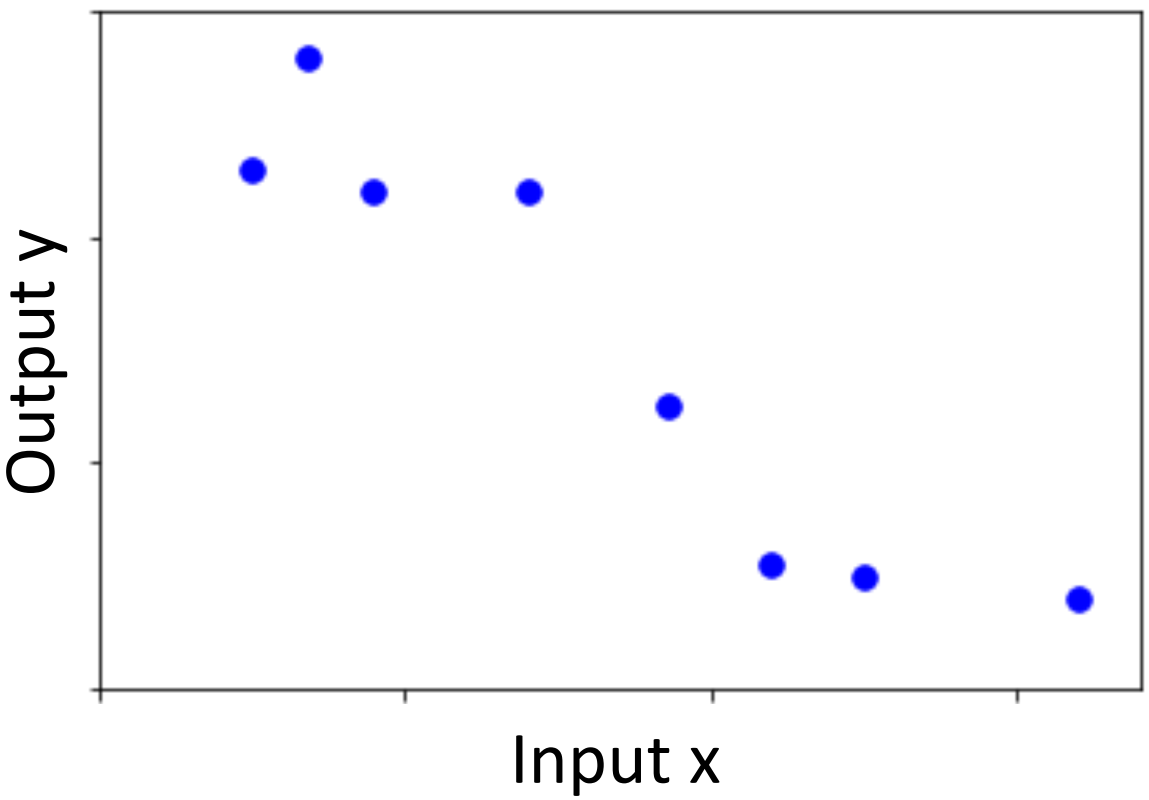
Neural Networks (nonparametric hack)





Nonparametric Regression

Kernel Regression

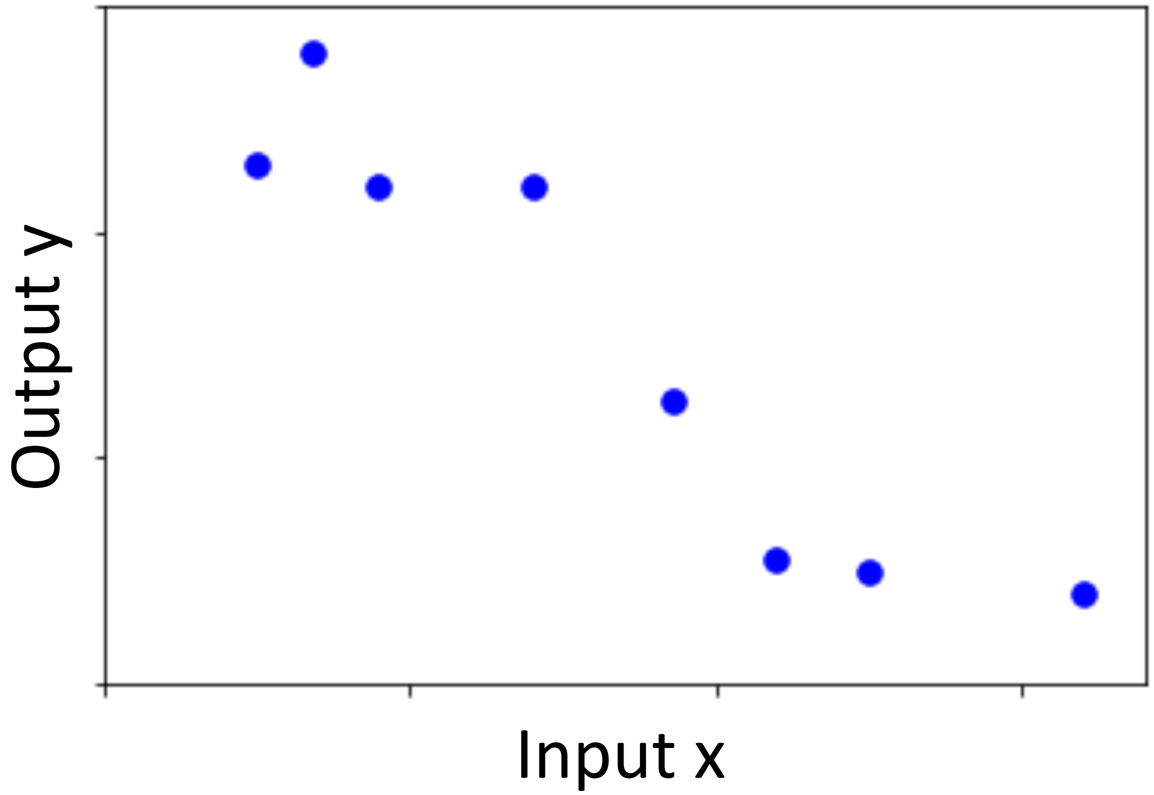


Nonparametric Regression

Kernel Regression: 2D input

Nonparametric Regression

Kernel Regression



RBF Kernel function

$$k(x, x') = e^{\frac{-\|x-x'\|_2^2}{2\sigma^2}}$$

$$= e^{-\gamma\|x-x'\|_2^2}$$

Poll 2

As x and x' get closer the RBF function:

A. Increases
B. Decreases
C. Stays the same

RBF Kernel function

$$k(x, x') = e^{\frac{-\|x-x'\|_2^2}{2\sigma^2}}$$

$$= e^{-\gamma\|x-x'\|_2^2}$$

Poll 3

As γ increases the RBF function:

A. Gets wider
B. Gets narrower
C. Stays the same

RBF Kernel function

$$k(x, x') = e^{\frac{-\|x-x'\|_2^2}{2\sigma^2}}$$

$$= e^{-\gamma\|x-x'\|_2^2}$$

Poll 4

As γ increases the max height of the RBF:

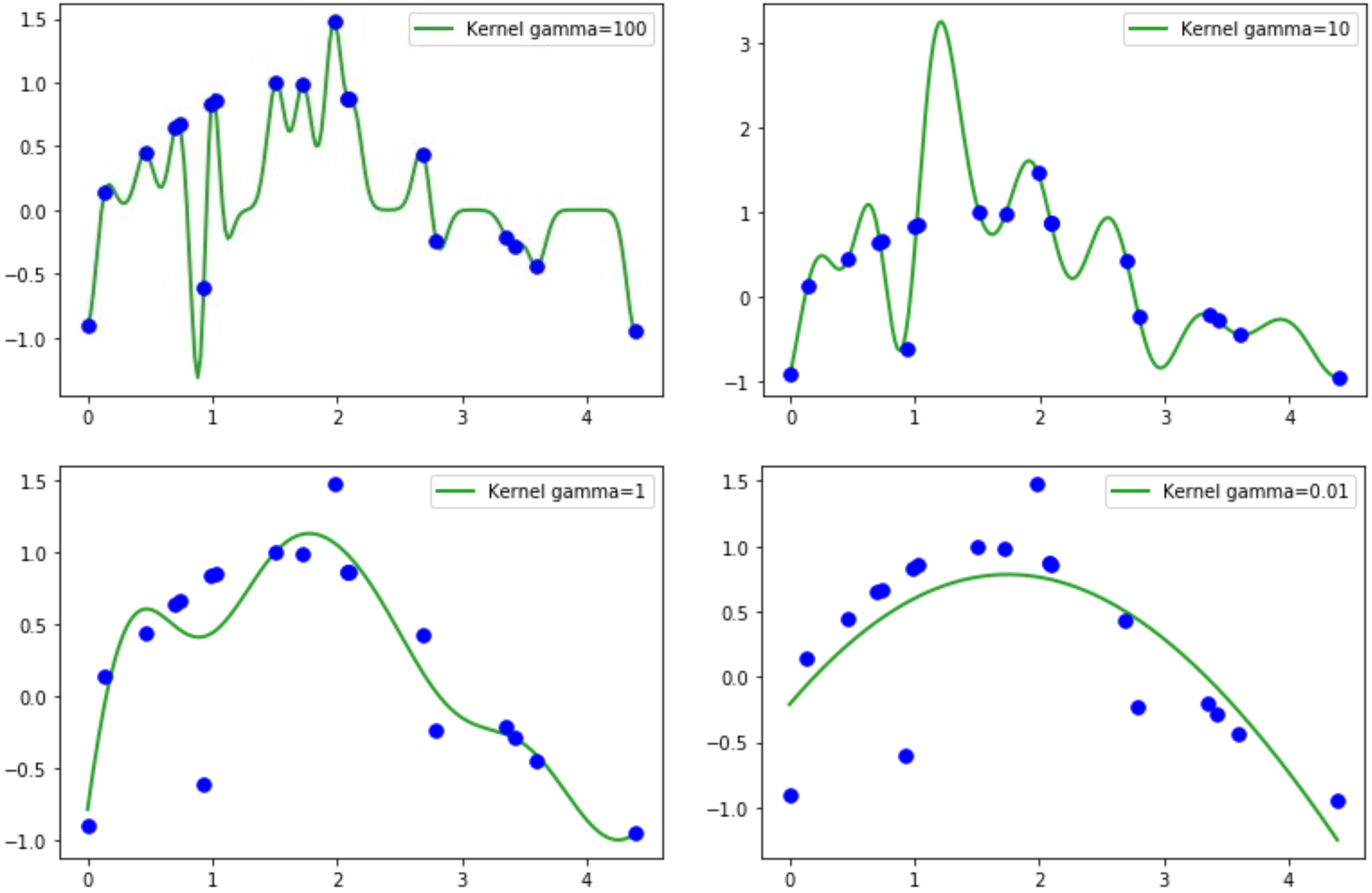
A. Increases
B. Decreases
C. Stays the same

RBF Kernel function

$$k(x, x') = e^{\frac{-\|x-x'\|_2^2}{2\sigma^2}}$$

$$= e^{-\gamma\|x-x'\|_2^2}$$

Kernel Regression



Kernel Regression

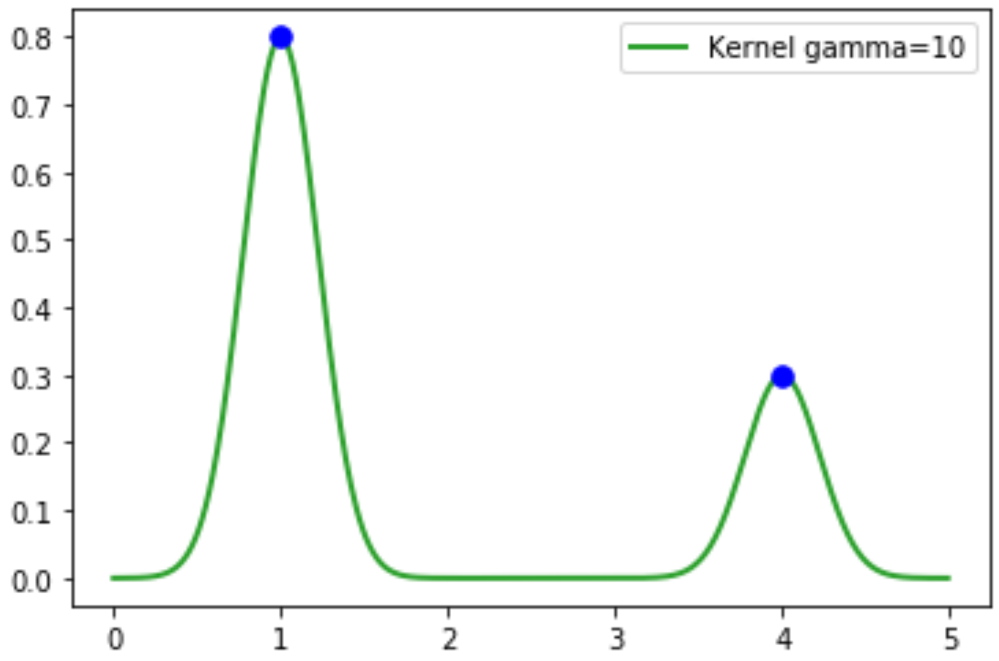
RBF kernel and corresponding hypothesis function

Distance kernel (Gaussian / Radial Basis Function)

- Close to point should be that point
- Far should be zero
- Mini Gaussian window

$$k(x, x') = e^{\frac{-\|x-x'\|_2^2}{2\sigma^2}} = e^{-\gamma\|x-x'\|_2^2}$$

- We control the variance

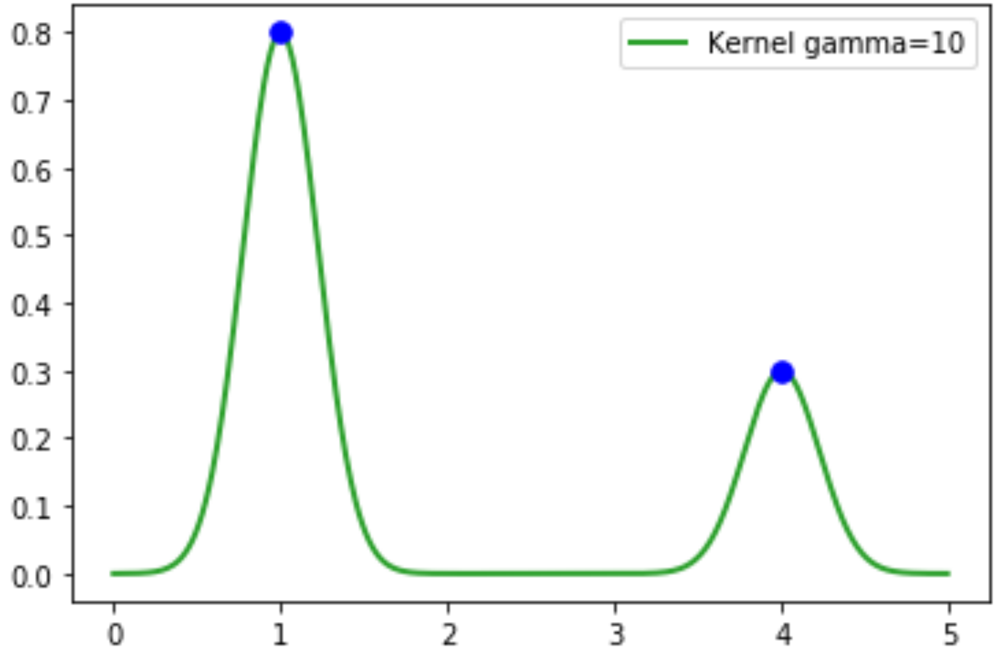


Kernel Regression

RBF kernel and corresponding hypothesis function

Prediction?

- Weighted sum of these little windows
  - $\hat{y} = h(x) = \sum_i \alpha_i k\left(x, x^{(i)}\right)$
  - What should α_i be?
    - $\alpha_i = y_i, \alpha = y$?
    - Need to account for points that are close together

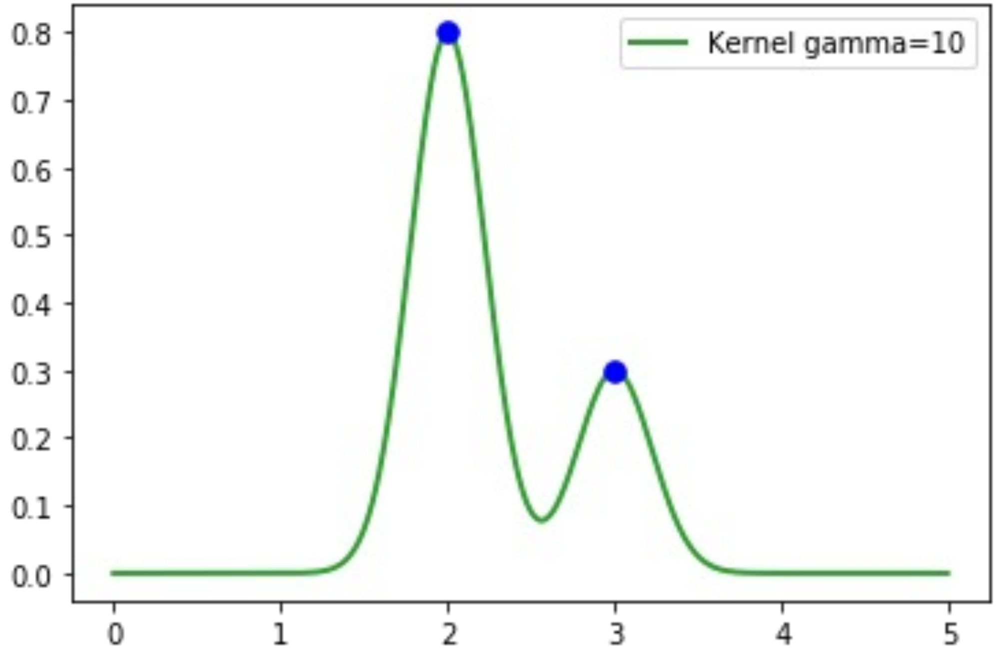


Kernel Regression

RBF kernel and corresponding hypothesis function

Prediction?

- Weighted sum of these little windows
  - $\hat{y} = h(x) = \sum_i \alpha_i k\left(x, x^{(i)}\right)$
  - What should α_i be?
    - $\alpha_i = y_i, \alpha = y$?
    - Need to account for points that are close together

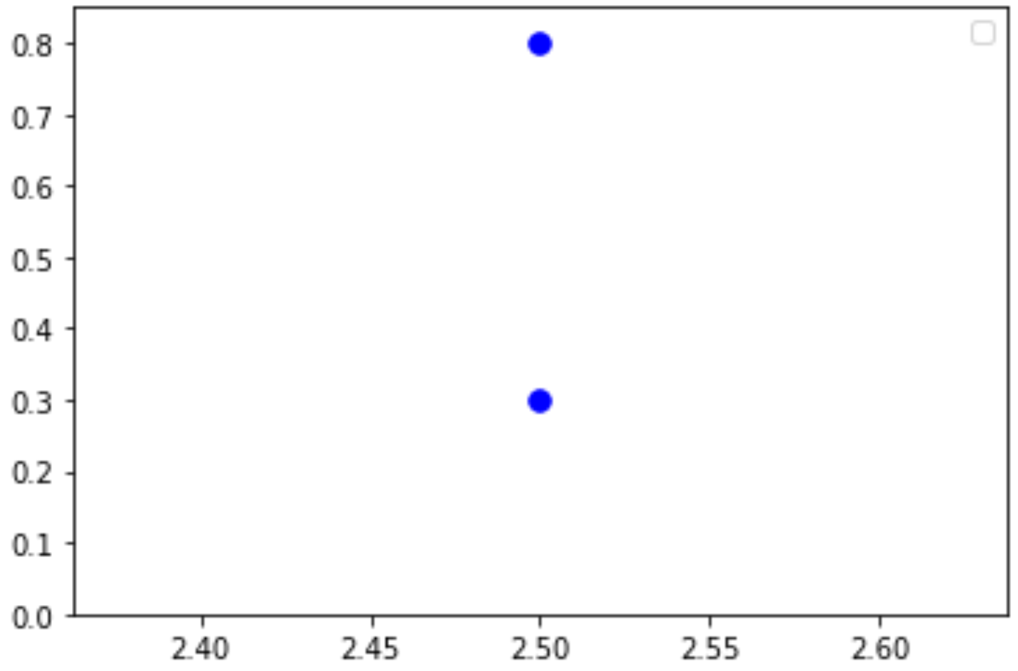


Kernel Regression

RBF kernel and corresponding hypothesis function

Prediction?

- Weighted sum of these little windows
  - $\hat{y} = h(x) = \sum_i \alpha_i k\left(x, x^{(i)}\right)$
  - What should α_i be?
    - $\alpha_i = y_i, \alpha = y$?
    - Need to account for points that are close together

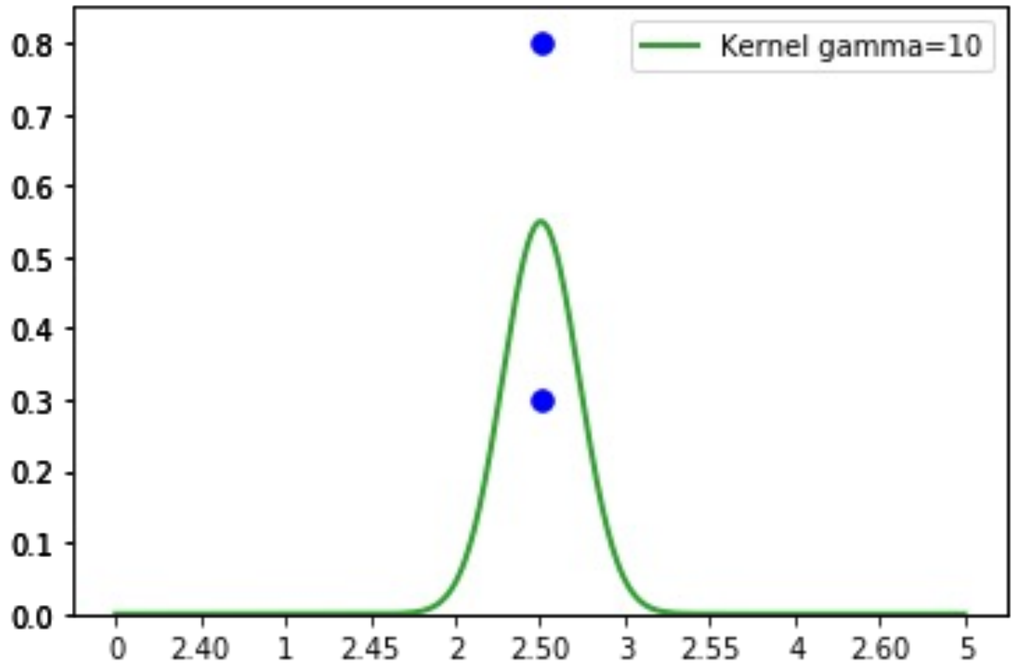


Kernel Regression

RBF kernel and corresponding hypothesis function

Prediction?

- Weighted sum of these little windows
  - $\hat{y} = h(x) = \sum_i \alpha_i k\left(x, x^{(i)}\right)$
  - What should α_i be?
    - Need to account for points that are close together



Kernel Regression

RBF kernel and corresponding hypothesis function

Prediction?

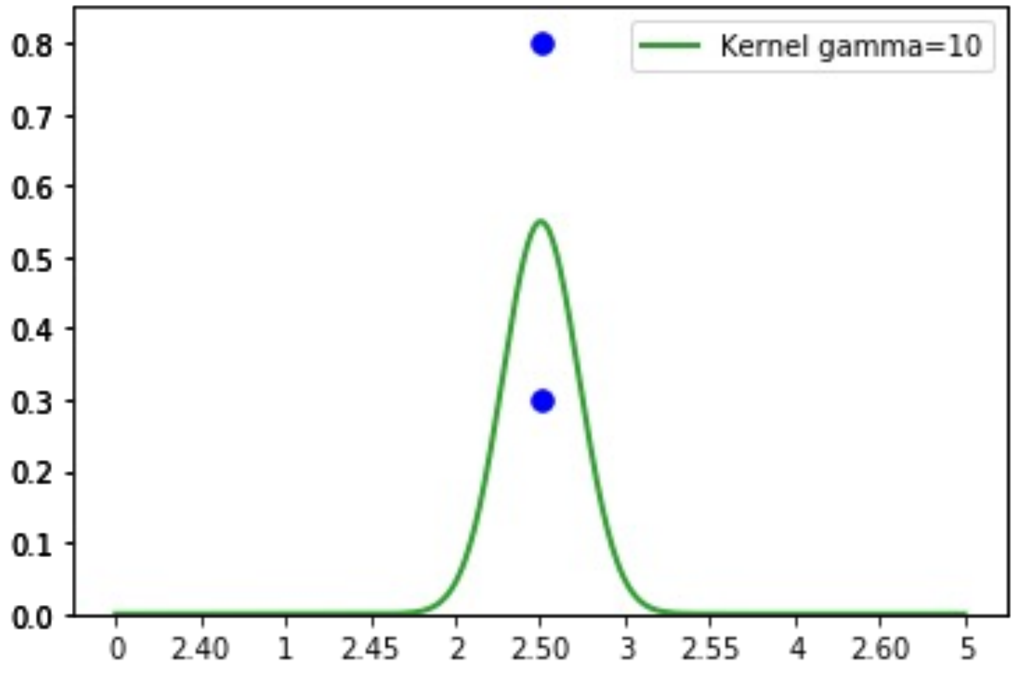
- Weighted sum of these little windows
  - $\hat{y} = h(x) = \sum_i \alpha_i k\left(x, x^{(i)}\right)$
  - What should α_i be?
    - Need to account for points that are close together

$$\alpha = (K)^{-1} y$$

$$\alpha = (K + \lambda I)^{-1} y$$

where $K_{ij} = k\left(x^{(i)}, x^{(j)}\right)$

and λ is small to help inversion



Kernelized Linear Regression

Reminder: Polynomial Linear Regression

Polynomial feature function

Reminder: Polynomial Linear Regression

Polynomial feature function

Least squares formulation

Least squares solution

Reminder: Polynomial Linear Regression

Polynomial feature function

- $x \rightarrow \phi(x) = [1, x, x^2, x^3]^T$
- $X \rightarrow \Phi$

Least squares formulation

- $\min_w \|y - \Phi w\|_2^2$

Least squares solution

- $w = (\Phi^T \Phi)^{-1} \Phi^T y$

Plus L2 regularization

- $\min_w \|y - \Phi w\|_2^2 + \lambda \|w\|_2^2$
- $w = (\Phi^T \Phi + \lambda \mathrm{I})^{-1} \Phi^T y$

Can rewrite as

- $w = \Phi^T (\Phi \Phi^T + \lambda \mathrm{I})^{-1} y$

Kernelized Linear Regression

L2 regularized linear regression (with feature function)

- $\min_{w} \|y - \Phi w\|_2^2 + \lambda \|w\|_2^2$
- $w = (\Phi^T \Phi + \lambda \mathrm{I})^{-1} \Phi^T y$

Can rewrite as

- $w = \Phi^T (\Phi \Phi^T + \lambda \mathrm{I})^{-1} y$

Let

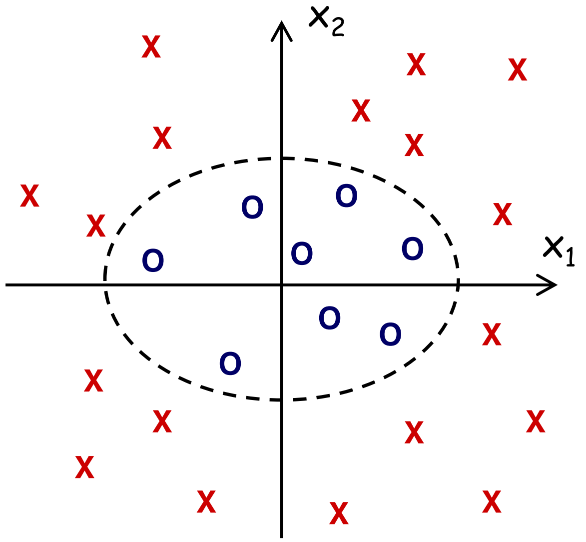
$\boldsymbol{\alpha} = (\Phi \Phi^T + \lambda \mathrm{I})^{-1} \mathbf{y}$

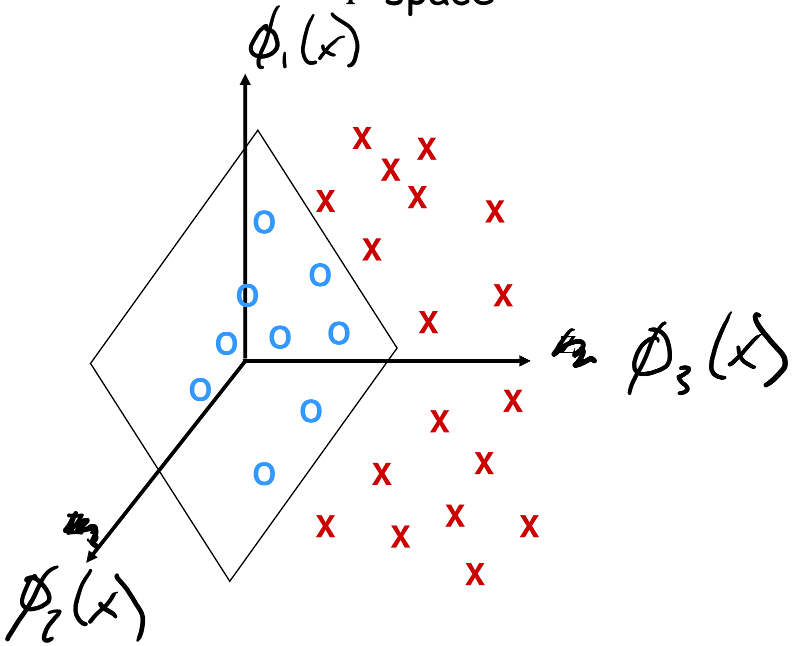
Prediction

- $\hat{y} = h(x) = \mathrm{w}^{\mathrm{T}} \mathrm{x}$

Example: Polynomial Kernel

https://www.youtube.com/watch?v=3liCbRZPrZA





Slide credit: CMU MLD Nina Balcan

Kernels: Motivation

Motivation #1: Inefficient Features

- Non-linearly separable data requires **high dimensional** representation
- Might be **prohibitively expensive** to compute or store

Motivation #2: Memory-based Methods

- k-Nearest Neighbors (KNN) for facial recognition allows a **distance metric** between images -- no need to worry about linearity restriction at all

Kernel Methods

Key idea:

1. Rewrite the algorithm so that we only work with **dot products** $x^T z$ of feature vectors
2. Replace the **dot products** $x^T z$ with a **kernel function** $k(x, z)$

The kernel $k(x,z)$ can be **any** legal definition of a dot product:

$$k(x, z) = \phi(x)^T \phi(z) \text{ for any function } \phi: \mathcal{X} \rightarrow \mathbf{R}^D$$

So we only compute the ϕ dot product **implicitly**

This **"kernel trick"** can be applied to many algorithms:

- classification: perceptron, SVM, …
- regression: ridge regression, …
- clustering: k-means, …

Slide credit: CMU MLD Matt Gormley

Kernel Methods

Q: These are just non-linear features, right?

A: Yes, but…

Q: Can't we just compute the feature transformation φ explicitly?

A: That depends...

Q: So, why all the hype about the kernel trick?

A: Because the **explicit features** might either be **prohibitively expensive** to compute or **infinite length** vectors

Slide credit: CMU MLD Matt Gormley

Example: Polynomial Kernel

For n=2, d=2, the kernel $K(x, z) = (x \cdot z)^d$ corresponds to

$$\phi: R^2 \rightarrow R^3, (x_1, x_2) \rightarrow \Phi(x) = (x_1^2, x_2^2, \sqrt{2} x_1 x_2)$$

$$\phi(x) \cdot \phi(z) = \left(x_1^2, x_2^2, \sqrt{2} x_1 x_2\right) \cdot (z_1^2, z_2^2, \sqrt{2} z_1 z_2)$$

$$= (x_1 z_1 + x_2 z_2)^2 = (x \cdot z)^2 = K(x, z)$$

Slide credit: CMU MLD Nina Balcan

Kernel Examples

Side Note: The feature space might not be unique!

Explicit representation #1:

$$\phi: R^2 \rightarrow R^3, (x_1, x_2) \rightarrow \Phi(x) = (x_1^2, x_2^2, \sqrt{2}x_1x_2)$$

$$\phi(x) \cdot \phi(z) = \left(x_1^2, x_2^2, \sqrt{2}x_1x_2\right) \cdot (z_1^2, z_2^2, \sqrt{2}z_1z_2)$$

$$= (x_1z_1 + x_2z_2)^2 = (x \cdot z)^2 = K(x, z)$$

Explicit representation #2:

$$\phi: R^2 \rightarrow R^4, (x_1, x_2) \rightarrow \Phi(x) = (x_1^2, x_2^2, x_1x_2, x_2x_1)$$

$$\phi(x) \cdot \phi(z) = (x_1^2, x_2^2, x_1x_2, x_2x_1) \cdot (z_1^2, z_2^2, z_1z_2, z_2z_1)$$

$$= (x \cdot z)^2 = K(x, z)$$

These two different feature representations correspond to the same kernel function!

Slide credit: CMU MLD Nina Balcan

Kernel Examples

Name	Kernel Function (implicit dot product)	Feature Space (explicit dot product)
Linear	$K(\mathbf{x}, \mathbf{z}) = \mathbf{x}^T\mathbf{z}$	Same as original input space
Polynomial (v1)	$K(\mathbf{x}, \mathbf{z}) = (\mathbf{x}^T\mathbf{z})^d$	All polynomials **of** degree d
Polynomial (v2)	$K(\mathbf{x}, \mathbf{z}) = (\mathbf{x}^T\mathbf{z} + 1)^d$	All polynomials **up to** degree d
Gaussian (RBF)	$K(\mathbf{x}, \mathbf{z}) = \exp(-\frac{\lvert\lvert \mathbf{x} - \mathbf{z} \rvert\rvert_2^2}{2\sigma^2})$	Infinite dimensional space
Hyperbolic Tangent (Sigmoid) Kernel	$K(\mathbf{x}, \mathbf{z}) = \tanh(\alpha\mathbf{x}^T\mathbf{z} + c)$	(With SVM, this is equivalent to a 2-layer neural network)

Slide credit: CMU MLD Matt Gormley