UVA CS 6316 – Fall 2015 Graduate: Machine Learning

Lecture 5: Non-Linear Regression Models

Dr. Yanjun Qi

University of Virginia

Department of
Computer Science

Where we are ? ➔ Five major sections of this course

- ❑ Regression (supervised)
- ❑ Classification (supervised)
- ❑ Unsupervised models
- ❑ Learning theory
- ❑ Graphical models

Today ➔ Regression (supervised)

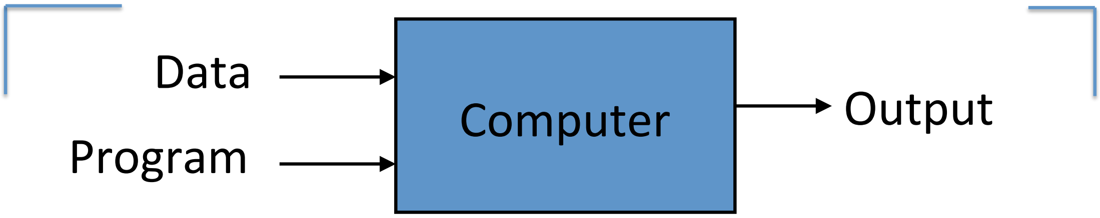
- ❑ Four ways to train / perform optimization for linear regression models
  - ❑ Normal Equation
  - ❑ Gradient Descent (GD)
  - ❑ Stochastic GD
  - ❑ Newton's method

- ❑ Supervised regression models
  - ❑ Linear regression (LR)
  - ❑ LR with non-linear basis functions
  - ❑ Locally weighted LR
  - ❑ LR with Regularizations

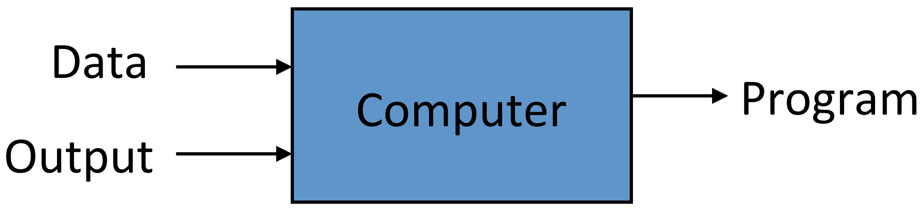
Today

- ❑ Machine Learning Method in a nutshell
- ❑ Regression Models Beyond Linear
  - LR with non-linear basis functions
  - Locally weighted linear regression
  - Regression trees and Multilinear Interpolation (later)

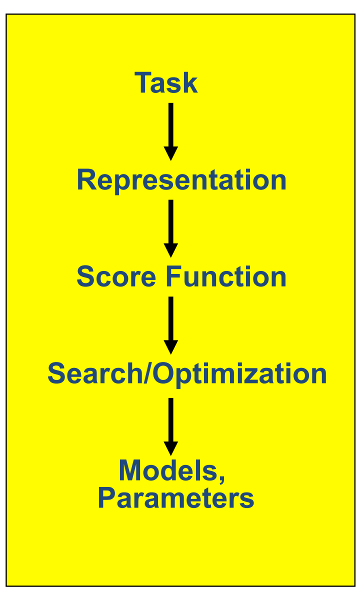
Traditional Programming



Machine Learning



Machine Learning in a Nutshell

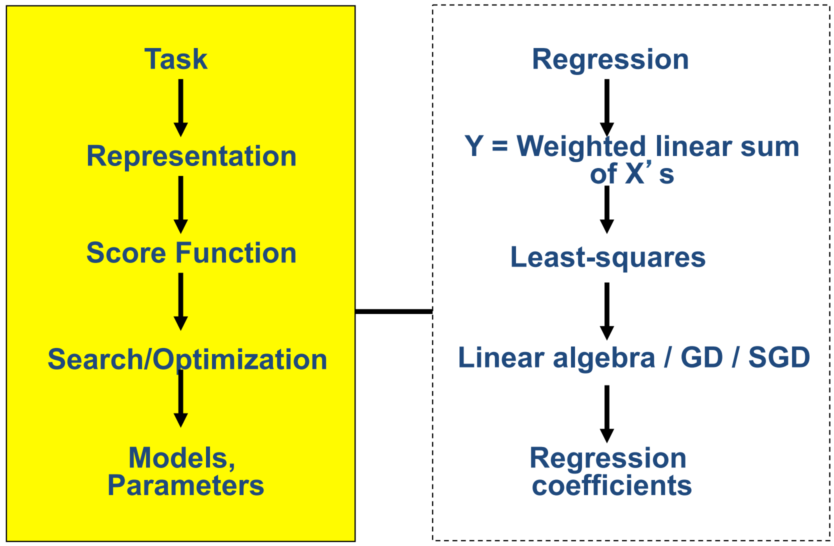


ML grew out of work in AI

Optimize a performance criterion using example data or past experience,

Aiming to generalize to unseen data

(1) Multivariate Linear Regression



$$\hat{y} = f(x) = \theta_0 + \theta_1 x^1 + \theta_2 x^2$$

$$= \theta^T \vec{x} = \vec{x}^T \theta$$

Today

- ❑ Machine Learning Method in a nutshell
- ❑ Regression Models Beyond Linear
  - LR with non-linear basis functions
  - Locally weighted linear regression
  - Regression trees and Multilinear Interpolation (later)
- ❑ Linear Regression Model with Regularizations
  - ❑ Ridge Regression
  - ❑ Lasso Regression

LR with non-linear basis functions

- LR does not mean we can only deal with linear relationships

$$\theta^T \vec{X} \Longrightarrow y=\theta_0+\sum_{j=1}^{m}\theta_j\varphi_j(x)=\theta^T\varphi(x)$$

- We are free to design (non-linear) features (e.g., basis function derived) under LR

where the $\varphi_j(x)$ are fixed basis functions (also define $\varphi_0(x)=1$).

- E.g.: polynomial regression:

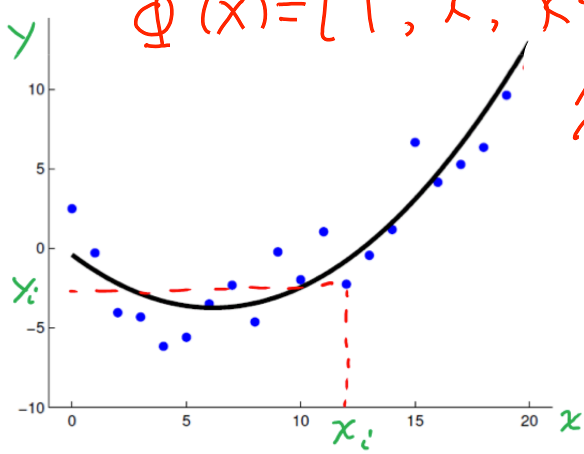
$$[1, x]^T \Rightarrow \varphi(x):=\left[1,x,x^2,x^3\right]^T$$

e.g. (1) **polynomial regression**

For example,

(x_i, y_i) training points $i=1,\dots,n$

$\Phi(x)=[1, x, x^2]^T$



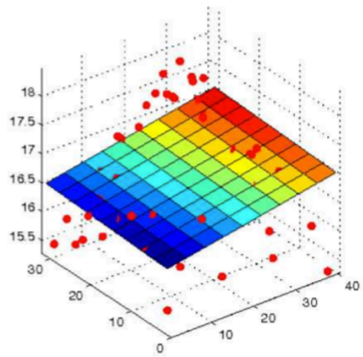
$$\hat{y}=\theta^T\Phi(x) = \theta_0 + \theta_1 x + \theta_2 x^2$$

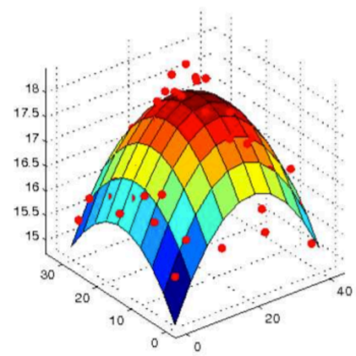
$$\theta^* = \left(\varphi^T\varphi\right)^{-1}\varphi^T\vec{y}$$

Dr. Nando de Freitas's tutorial slide

e.g. (1) **polynomial regression**

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

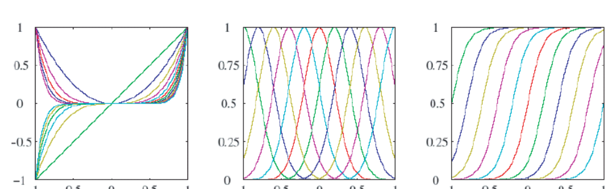




KEY: if the bases are given, the problem of learning the parameters is still linear.

Many Possible Basis functions

- There are many basis functions, e.g.:
  - Polynomial $\varphi_j(x) = x^{j-1}$
  - Radial basis functions $\phi_j(x) = \exp\left(-\frac{(x-\mu_j)^2}{2s^2}\right)$
  - Sigmoidal $\phi_j(x) = \sigma\left(\frac{x-\mu_j}{s}\right)$
  - Splines,
  - Fourier,
  - Wavelets, etc



e.g. (2) LR with radial-basis functions

- E.g.: LR with RBF regression:

$$\hat{y} = \theta_0 + \sum_{j=1}^{m} \theta_j \varphi_j(x) = \varphi(x)^T \theta$$

$$\varphi(x) := \left[1, K_{\lambda_1}(x, r_1), K_{\lambda 2}(x, r_2), K_{\lambda_3}(x, r_3), K_{\lambda 4}(x, r_4)\right]^T$$

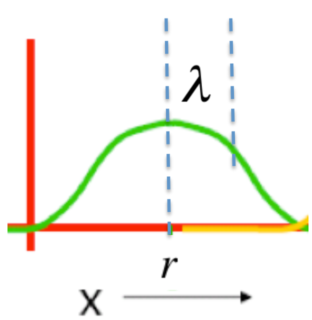
$$\Rightarrow \quad \theta^* = \left(\varphi^T \varphi\right)^{-1} \varphi^T \vec{y}$$

RBF = radial-basis function: a function which depends only on the radial distance from a centre point

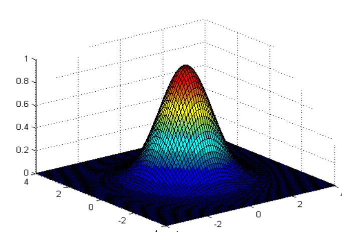
Gaussian RBF →

$$K_\lambda(\underline{x}, r) = \exp\left(-\frac{(\underline{x} - r)^2}{2\lambda^2}\right)$$

as distance from the centre r increases, the output of the RBF decreases

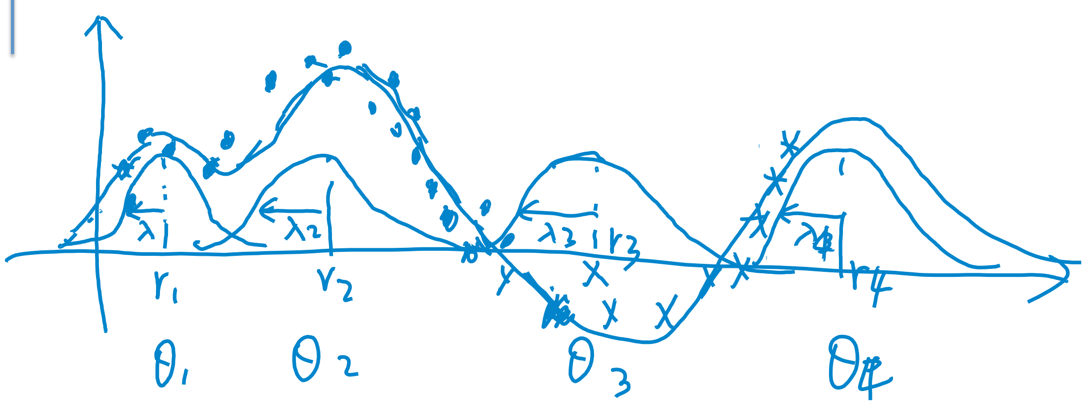


1D case



2D case

e.g. another Linear regression with 1D RBF basis functions (4 predefined centres and width)

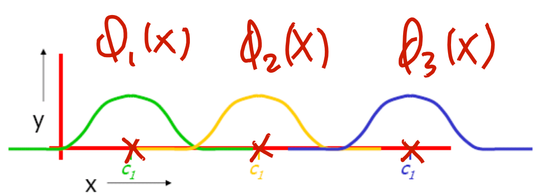


$$\varphi(x) := \left[1, K_{\lambda_1}(x, r_1), K_{\lambda 2}(x, r_2), K_{\lambda_3}(x, r_3), K_{\lambda 4}(x, r_4)\right]^T$$

$$\theta^* = \left(\varphi^T \varphi\right)^{-1} \varphi^T \vec{y}$$

e.g. a LR with 1D RBFs (3 predefined centres and width)

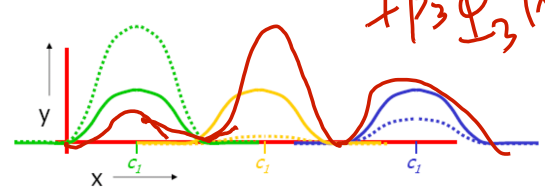
- 1D RBF



$y^{est} = \beta_1 \phi_1(x) + \beta_2 \phi_2(x) + \beta_3 \phi_3(x)$

$$\hat{y} = \beta_1 \Phi_1(x) + \beta_2 \Phi_2(x) + \beta_3 \Phi_3(x) + \beta_0$$

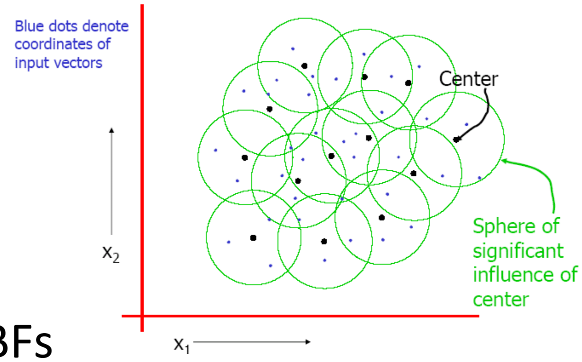
- After fit:



$y^{est} = 2\phi_1(x) + 0.05\phi_2(x) + 0.5\phi_3(x)$

e.g. 2D Good and Bad RBFs

- A good 2D RBF



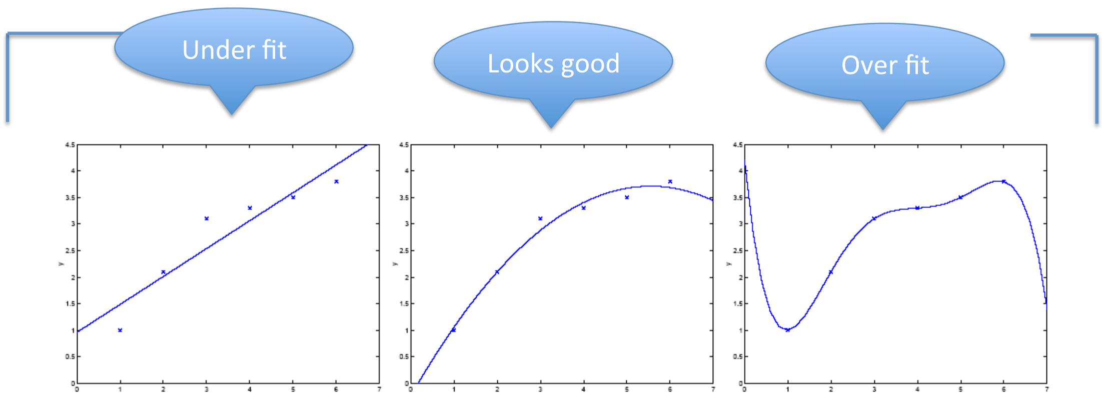
- Two bad 2D RBFs

Two main issues:

- Learn the parameter \theta
  - Almost the same as LR, just ➔ X to $\varphi(x)$
  - Linear combination of basis functions (that can be non-linear)

- How to choose the model order, e.g. polynomial degree for polynomial regression

Issue: Overfitting and underfitting

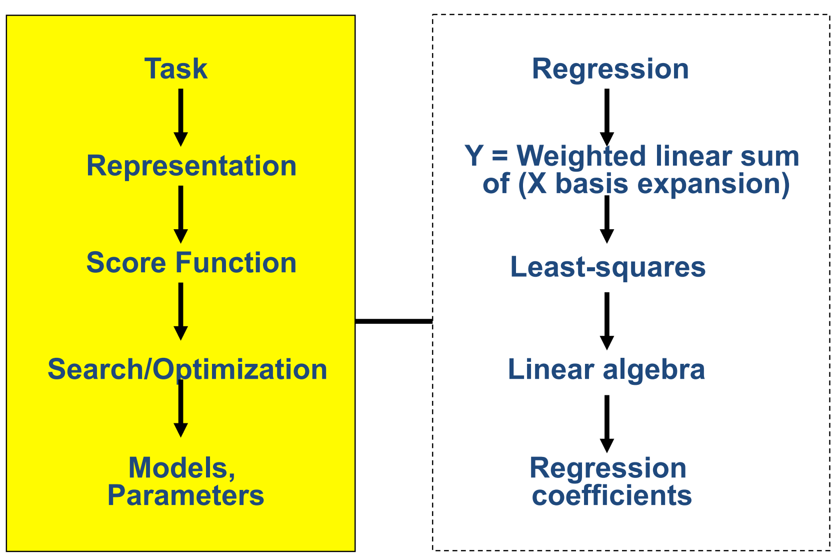


$y=\theta_0+\theta_1 x$ $\quad$ $y=\theta_0+\theta_1 x+\theta_2 x^2$ $\quad$ $y=\sum_{j=0}^{5}\theta_j x^j$

Generalisation: learn function / hypothesis from past data in order to "explain", "predict", "model" or "control" new data examples

K-fold Cross Validation !!!!

(2) Multivariate Linear Regression with basis Expansion



$$\hat{y}=\theta_0+\sum_{j=1}^{m}\theta_j\varphi_j(x)=\varphi(x)^T\theta$$

Today

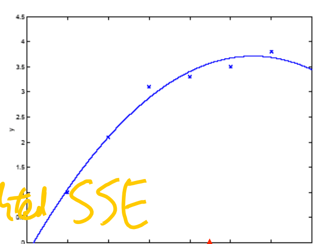
- ❑ Machine Learning Method in a nutshell
- ❑ Regression Models Beyond Linear
  - – LR with non-linear basis functions
  - – Locally weighted linear regression
  - – Regression trees and Multilinear Interpolation (later)

Locally weighted linear regression

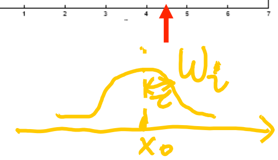
- The algorithm:

  Instead of minimizing $J(\theta) = \frac{1}{2}\sum_{i=1}^{n}(\mathbf{x}_i^T\theta - y_i)^2$ SSE

  now we fit θ to minimize $J(\theta) = \frac{1}{2}\sum_{i=1}^{n} w_i (\mathbf{x}_i^T\theta - y_i)^2$ Weighted SSE



  Where do w_i's come from? $w_i = K(\mathbf{x}_i, \mathbf{x}_0) = \exp\left(-\frac{(\mathbf{x}_i - \mathbf{x}_0)^2}{2\tau^2}\right)$

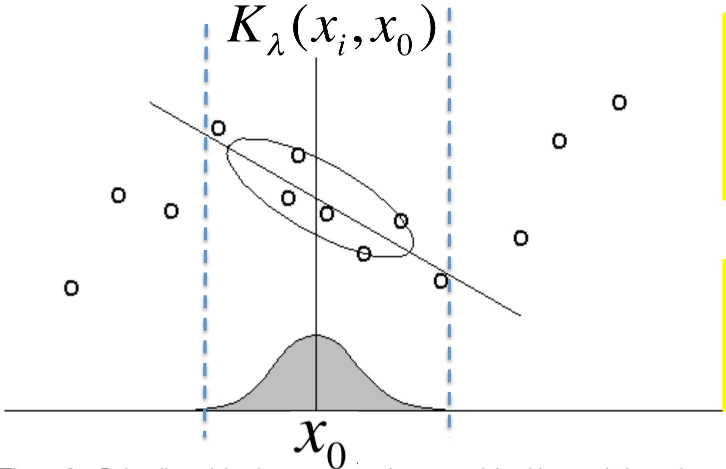


  - where **x_0** is the query point for which we'd like to know its corresponding **y**

→ Essentially we put higher weights on (those errors from) training examples that are close to the query point x_0 (than those that are further away from the query point)

Locally weighted regression

- *aka* locally weighted regression, locally linear regression, LOESS, …



linear_func(x)->y
➔
could represent only the neighbor region of x_0

Use RBF function to pick out/emphasize the neighbor region of x_0 ➔ $K_\lambda(x_i, x_0)$

Figure 2: In locally weighted regression, points are weighted by proximity to the current x in question using a kernel. A regression is then computed using the weighted points.

Locally weighted linear regression

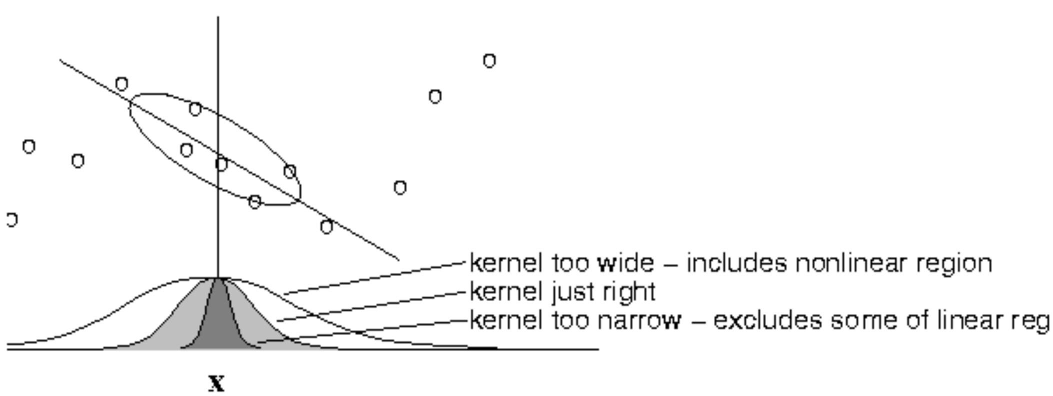


Figure 3: The estimator variance is minimized when the kernel includes as many training points as can be accommodated by the model. Here the linear LOESS model is shown. Too large a kernel includes points that degrade the fit; too small a kernel neglects points that increase confidence in the fit.

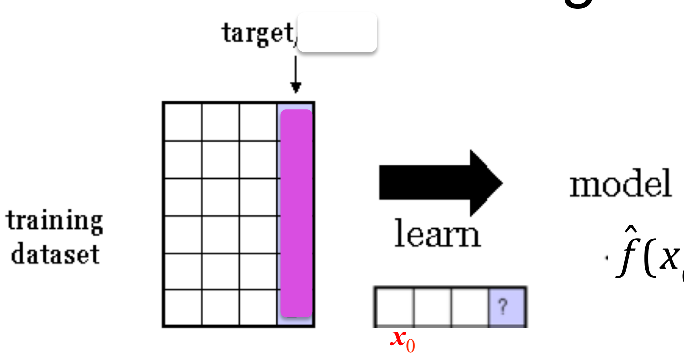
Locally weighted linear regression

- Separate weighted least squares **at each target point x_0:**

$$\min_{\alpha(x_0),\beta(x_0)} \sum_{i=1}^{N} K_\lambda(x_i,x_0)[y_i-\alpha(x_0)-\beta(x_0)x_i]^2$$

$$\hat{f}(x_0)=\hat{\alpha}(x_0)+\hat{\beta}(x_0)x_0$$

LEARNING of Locally weighted linear regression



$$\hat{f}(x_0)=\hat{\alpha}(x_0)+\hat{\beta}(x_0)x_0$$

➔ Separate weighted least squares **at each target point x_0**

Locally weighted linear regression

e.g. when for only one feature variable

- Separate weighted least squares **at each target point x_0:**

$$\min_{\alpha(x_0),\beta(x_0)} \sum_{i=1}^{N} K_\lambda(x_0,x_i)[y_i-\alpha(x_0)-\beta(x_0)x_i]^2$$

$$\hat{f}(x_0)=\hat{\alpha}(x_0)+\hat{\beta}(x_0)x_0$$

- $b(x)^T=(1,x)$; B: Nx2 regression matrix with i-th row $b(x)^T$;

$$W_{N\times N}(x_0)=diag\left(K_\lambda(x_0,x_i)\right), i=1,\ldots,N$$

LWR

$$\hat{f}(x_0)=b(x_0)^T(B^TW(x_0)B)^{-1}B^TW(x_0)y \Rightarrow \theta^*(x_0)$$

versus

LR $\hat{f}(x_q)=(x_q)^T\theta^*=(x_q)^T\left(X^TX\right)^{-1}X^T\vec{y}$

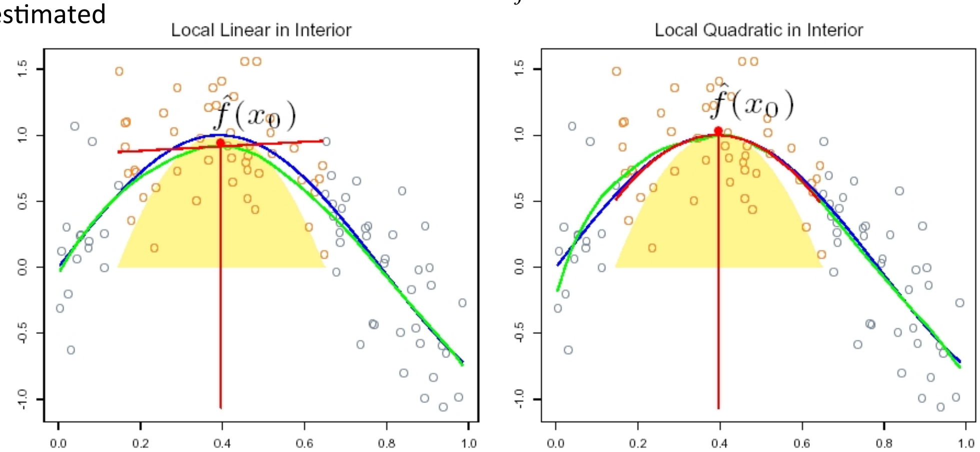
More ➔ Local Weighted Polynomial Regression

- Local polynomial fits of any degree d

$$\min_{\alpha(x_0),\beta_j(x_0),j=1,\ldots,d} \sum_{i=1}^{N} K_\lambda(x_0,x_i)\left[y_i-\alpha(x_0)-\sum_{j=1}^{d}\beta_j(x_0)x_i^j\right]^2$$

$$\hat{f}(x_0)=\hat{\alpha}(x_0)+\sum_{j=1}^{d}\hat{\beta}_j(x_0)x_0^j$$

Blue: true
Green: estimated



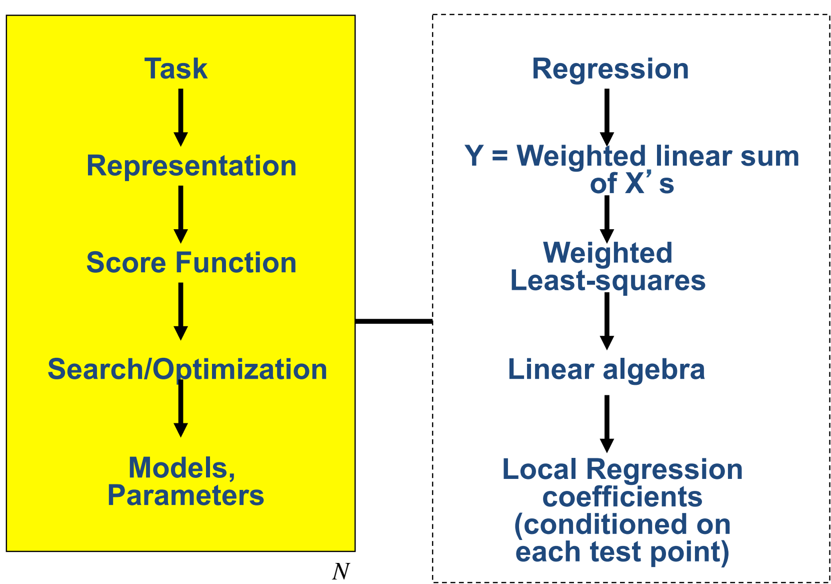
Parametric vs. non-parametric

- Locally weighted linear regression is a **non-parametric** algorithm.
- The (unweighted) linear regression algorithm that we saw earlier is known as a **parametric** learning algorithm
  - because it has a fixed, finite number of parameters (the θ), which are fit to the data;

    $p \times 1$
  - Once we've fit the *theta* and stored them away, we no longer need to keep the training data around to make future predictions.
  - In contrast, to make predictions using locally weighted linear regression, we need to keep the entire training set around.
- The term "non-parametric" (roughly) refers to the fact that the amount of knowledge we need to keep, in order to represent the hypothesis grows with linearly the size of the training set.

$n >> p$

(3) Locally Weighted / Kernel Linear Regression



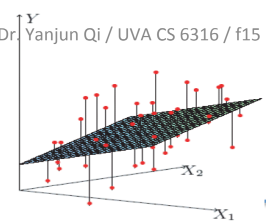
$$\min_{\alpha(x_0),\beta(x_0)} \sum_{i=1}^{N} K_\lambda(x_i, x_0)[y_i - \alpha(x_0) - \beta(x_0)x_i]^2$$

$$\hat{f}(x_0) = \hat{\alpha}(x_0) + \hat{\beta}(x_0)x_0$$

Today Recap

- ❑ Machine Learning Method in a nutshell
- ❑ Regression Models Beyond Linear
  - – LR with non-linear basis functions
  - – Locally weighted linear regression
  - – Regression trees and Multilinear Interpolation (later)

Probabilistic Interpretation of Linear Regression (LATER)



- Let us assume that the target variable and the inputs are related by the equation:

$$y_i = \theta^T \mathbf{x}_i + \varepsilon_i$$

where ε is an error term of unmodeled effects or random noise

- Now assume that ε follows a Gaussian $N(0,\sigma)$, then we have:

$$p(y_i \mid x_i;\theta) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y_i - \theta^T \mathbf{x}_i)^2}{2\sigma^2} \right)$$

Many more variations of LR from this perspective, e.g. binomial / poisson (LATER)

- By independence (among samples) assumption:

$$L(\theta) = \prod_{i=1}^{n} p(y_i \mid x_i;\theta) = \left(\frac{1}{\sqrt{2\pi}\sigma} \right)^n \exp\left(-\frac{\sum_{i=1}^{n} (y_i - \theta^T \mathbf{x}_i)^2}{2\sigma^2} \right)$$

References

- Big thanks to Prof. Eric Xing @ CMU for allowing me to reuse some of his slides
- ❑ Prof. Nando de Freitas's tutorial slide