UVA CS 6316: Machine Learning

Lecture 5 Extra: Nonparametric Regression Models

Dr. Yanjun Qi

University of Virginia

Department of Computer Science

Where are we ? ➔ Five major sections of this course

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

Regression (supervised)

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

variations of $\underset{\theta}{\mathrm{argmin}}\, L(\theta)$

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

variations of $f(x)$

variations of $L(\theta)$

Today Extra ➔ Nonparametric Regression (supervised)

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

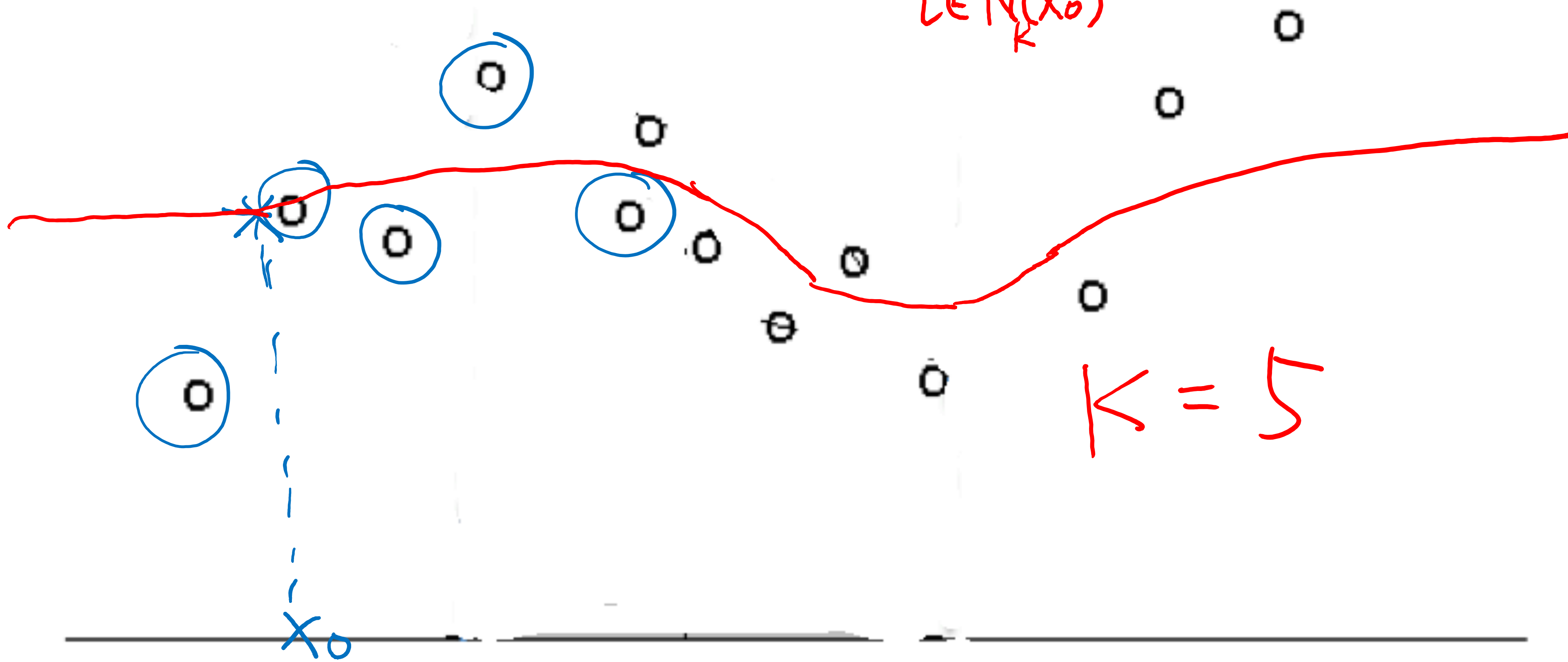
- ❑ ~~Supervised regression models~~
  - ❑ ~~Linear regression (LR)~~
  - ❑ ~~LR with non-linear basis functions~~
  - ❑ kNN based LR
  - ❑ Locally weighted LR
  - ❑ LR with Regularizations

K-Nearest Neighbor

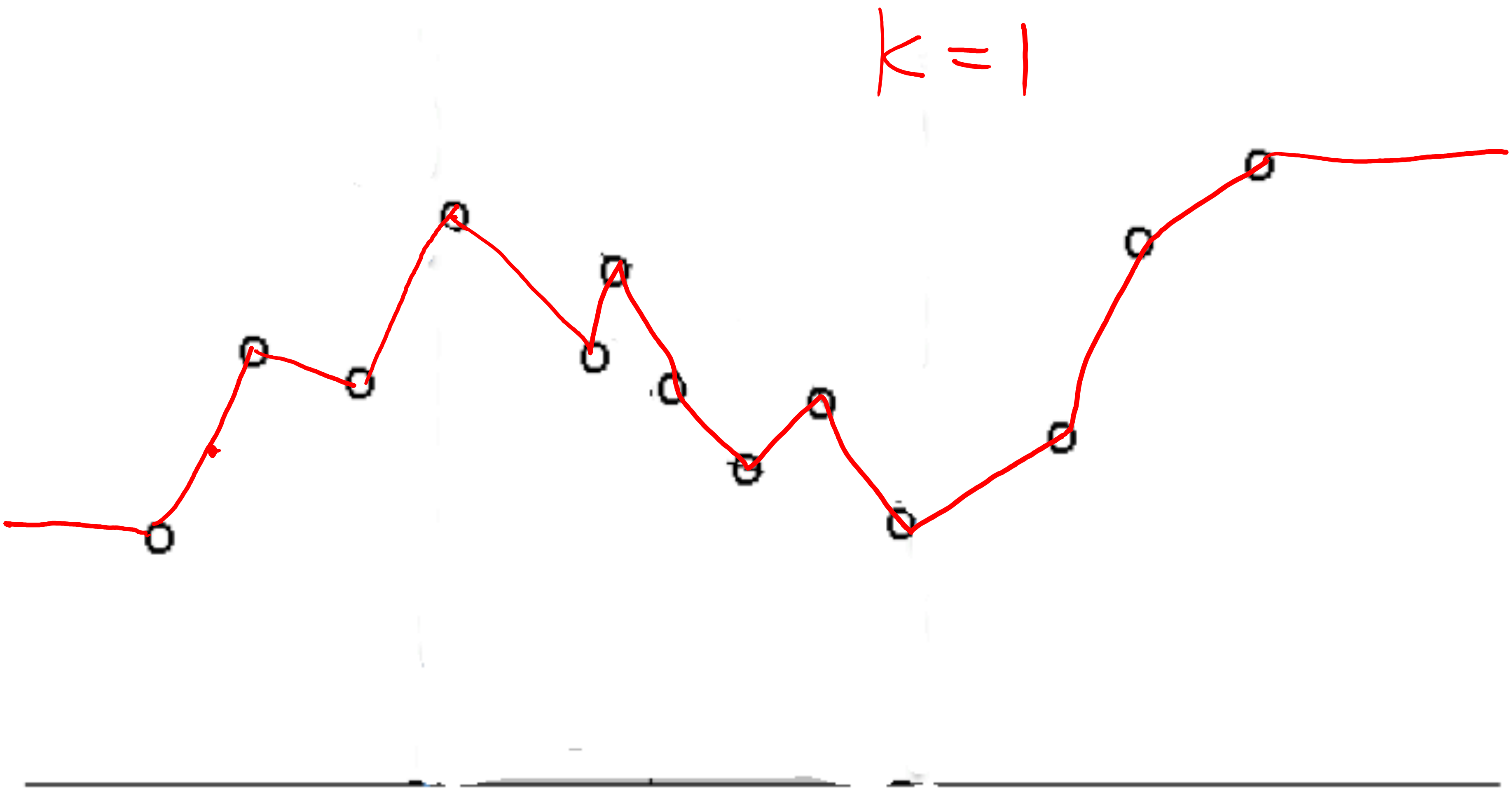
- Features
  - All instances correspond to points in an p-dimensional Euclidean space
  - Regression is delayed till a new instance arrives
  - Regression is done by comparing feature vectors of the different points
  - Target function may be discrete or real-valued
    - When target is continuous, the prediction is the mean value of the k nearest training examples

K=5-Nearest Neighbor (1D input)

$$\hat{y} = \frac{1}{k} \sum_{i \in N_k(x_0)} y_i$$



K=1-Nearest Neighbor (1D input)



K-Nearest Neighbor

Task	Regression/ classification
↓	↓
Representation	Local Smoothness
↓	↓
Score Function	NA
↓	↓
Search/Optimization	NA
↓	↓
Models, Parameters	Training Samples

Variants: Distance-Weighted k-Nearest Neighbor Algorithm

- Assign weights to the neighbors based on their "distance" from the query point
  - Weight "may" be inverse square of the distances
- All training points may influence a particular instance
  - E.g., Shepard's method/ Modified Shepard, … by Geospatial Analysis

e.g. $\hat{y} = \frac{1}{K} \sum_{i \in N_k(x_0)} W_i y_i$

$W_i \rightarrow$ RBF

$K_\lambda(x_i, x_0)$

Instance-based Regression vs. Linear Regression

- Linear Regression Learning
  - Explicit description of target function on the whole training set

- Instance-based Learning
  - Learning=storing all training instances
  - Referred to as “Lazy” learning

Today Extra ➔ Nonparametric 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~~
  - ❑ kNN based LR
  - ❑ Locally weighted LR
  - ❑ LR with Regularizations

Locally weighted regression

- *aka* locally weighted regression, local linear regression, LOESS, …
  - A combination of kNN and Linear regression

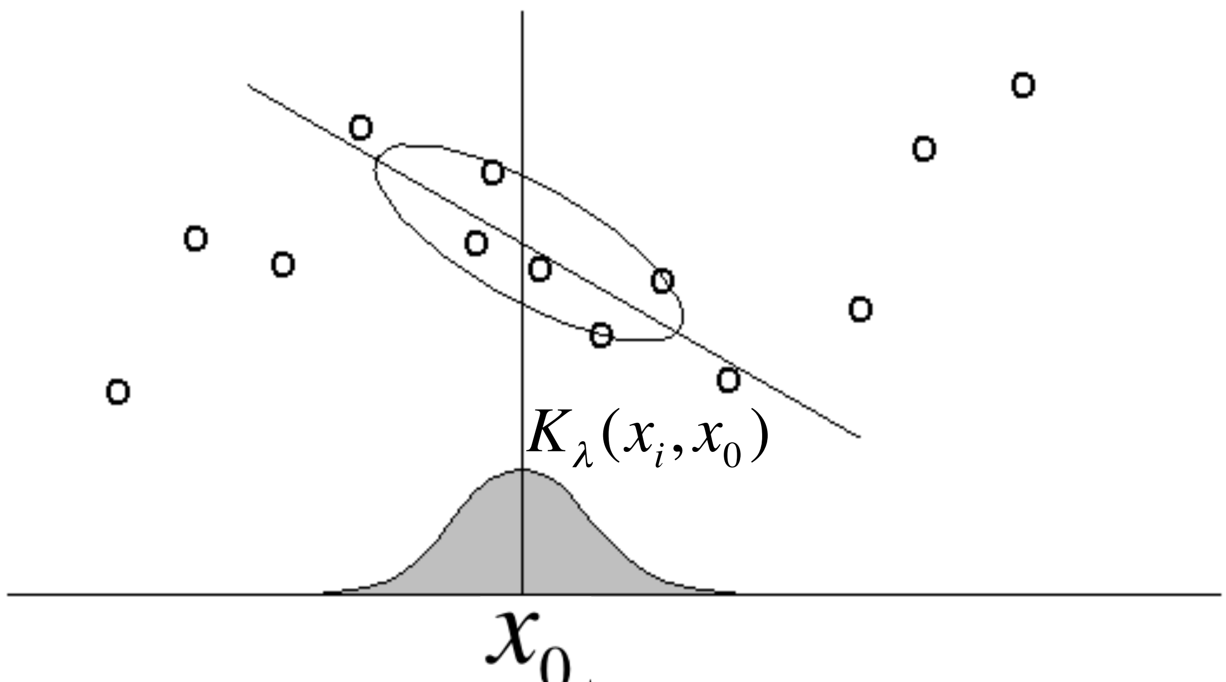


Figure 2: In locally weighted regression, points are weighted by proximity to the current **x** in question using a kernel. A regression is then computed using the weighted points.

Locally weighted regression

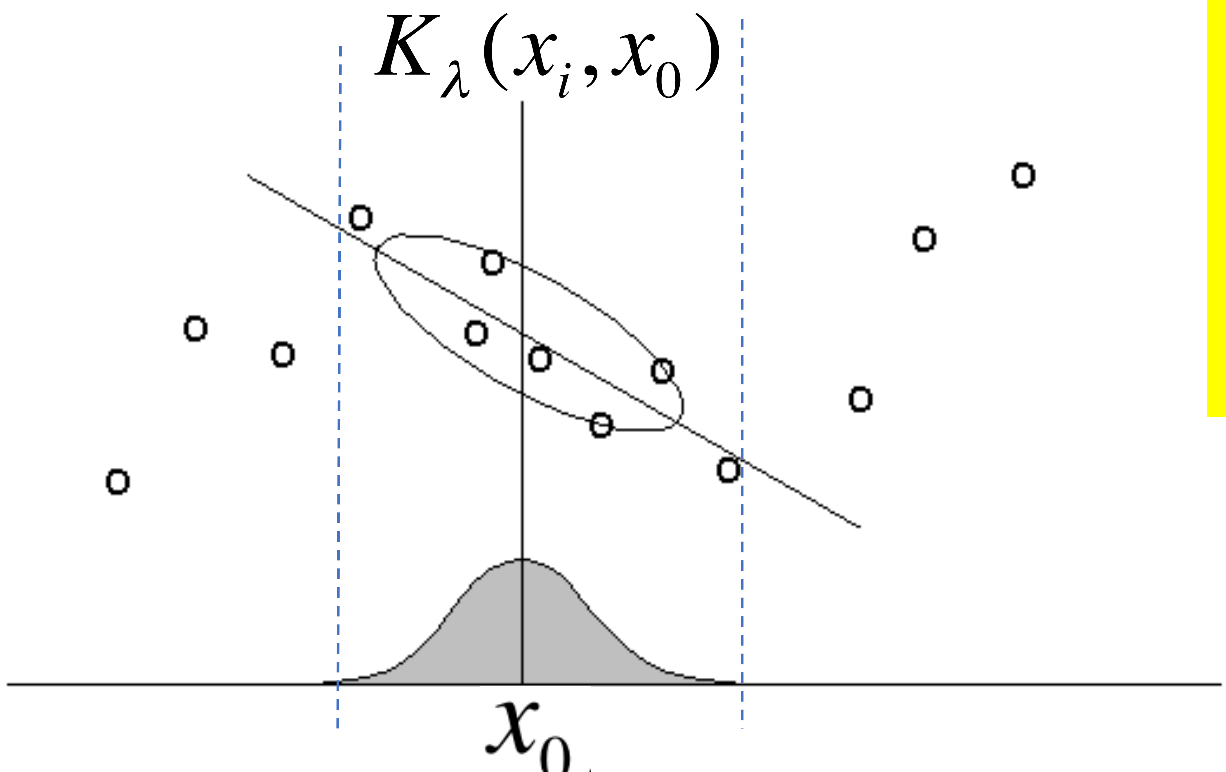
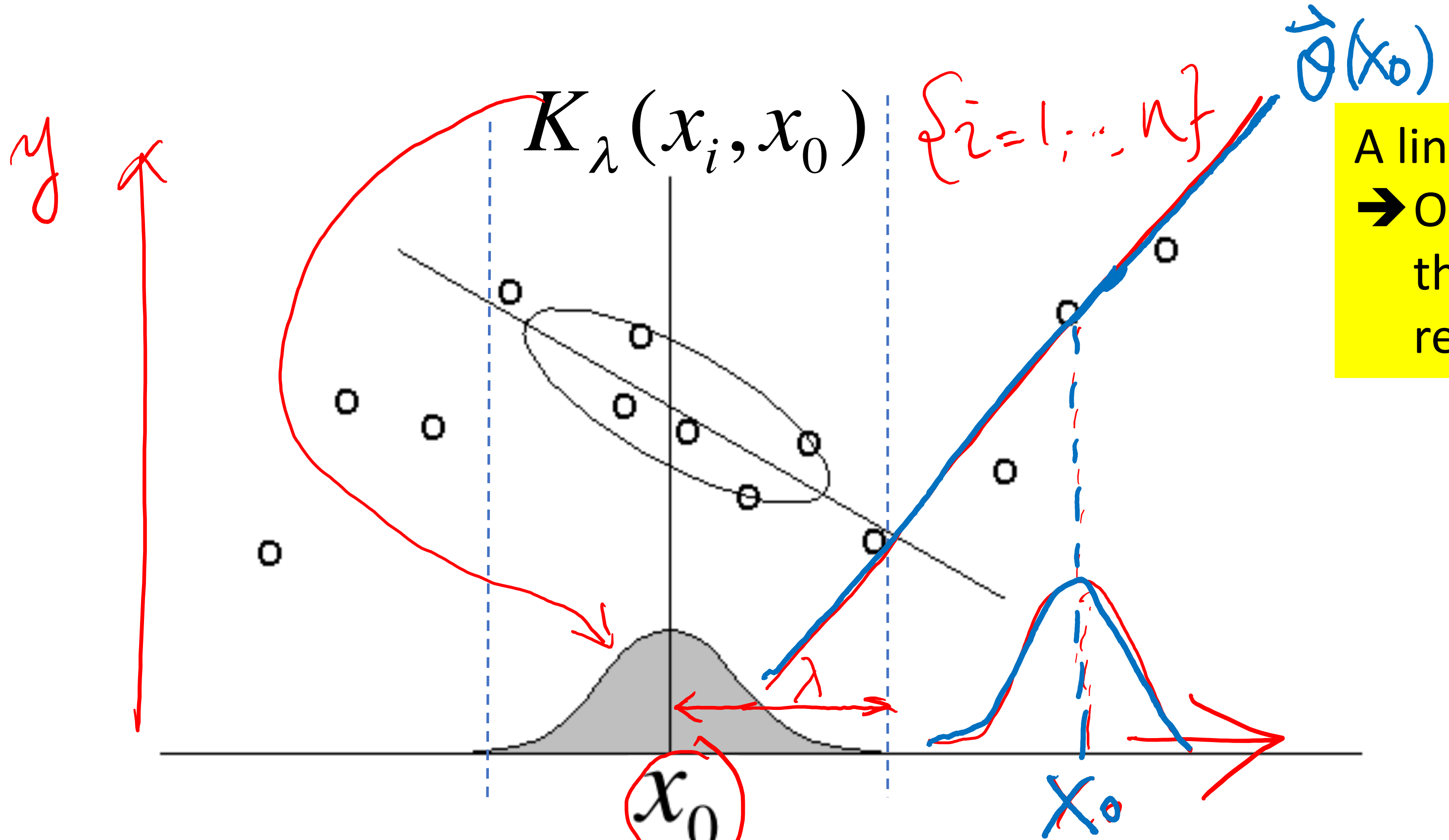


Figure 2: In locally weighted regression, points are weighted by proximity to the current **x** in question using a kernel. A regression is then computed using the weighted points.

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

Locally weighted regression

$$\hat{f}(x_0) = \widehat{\theta_0}(x_0) + \widehat{\theta_1}(x_0)x_0$$



A linear_func(x)->y
➔ Only to represent the neighbor region of 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

Instead of minimizing

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

SSE

$\downarrow$

now we fit to minimize

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

WSSE

$$w_i=K_\lambda(\mathbf{x}_i,\mathbf{x}_0)=\exp\left(-\frac{(\mathbf{x}_i-\mathbf{x}_0)^2}{2\lambda^2}\right)$$

where $\mathbf{x\_0}$ is the query point for which we'd like to know its corresponding $\mathbf{y}$

Locally weighted linear regression

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

w_i comes from:

$$w_i = K_\lambda(\mathbf{x}_i, \mathbf{x}_0) = \exp\left(-\frac{(\mathbf{x}_i - \mathbf{x}_0)^2}{2\lambda^2}\right)$$

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

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

Locally weighted linear regression

- The width of RBF matters !

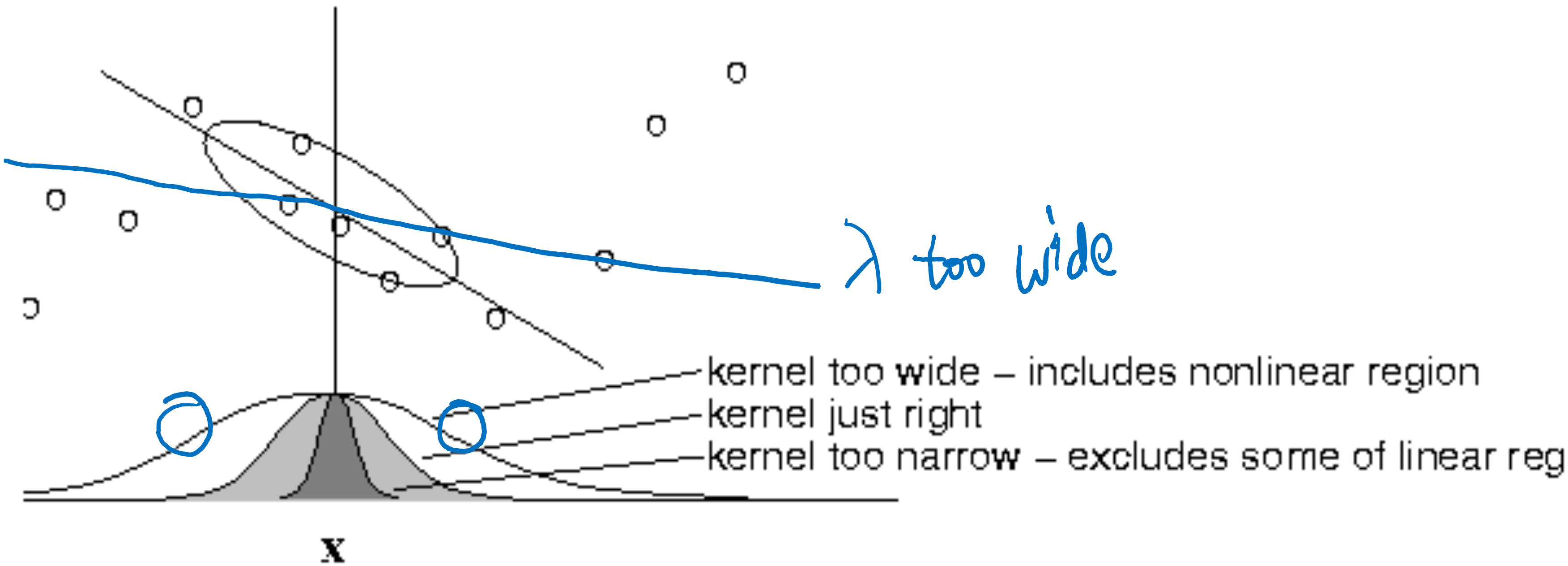
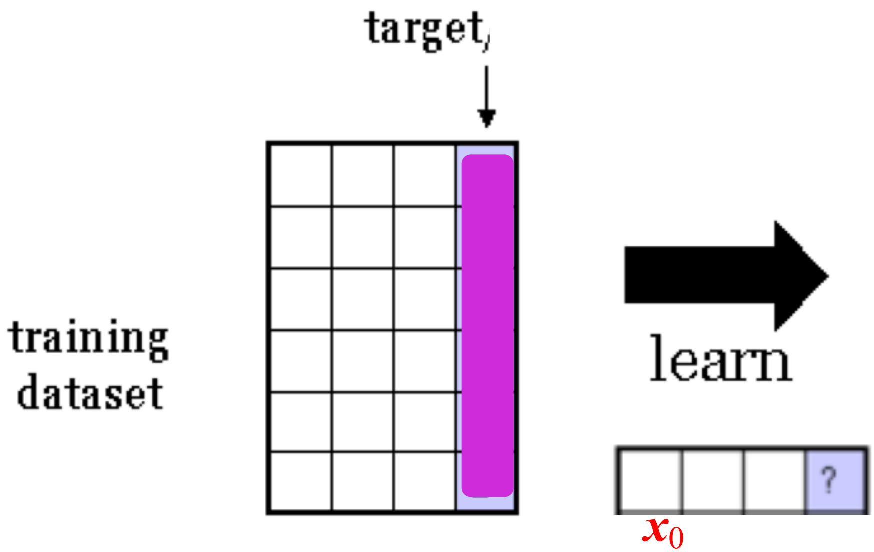


Figure 3: The estimator variance is minimized when the kernel includes as many training points as can be accommodated by the model. Here the linear LOESS model is shown. Too large a kernel includes points that degrade the fit; too small a kernel neglects points that increase confidence in the fit.

LEARNING of Locally weighted linear regression



model

$$\hat{f}(x_0) = \widehat{\theta_0}(x_0) + \widehat{\theta_1}(x_0)x_0$$

- Separate weighted least squares training and inference **at each target point x_0**

Locally weighted linear regression

- ➔Separate weighted least square error minimization **at each target point x_0:**

$$\theta^*(\mathbf{x}_0) = \arg\min \frac{1}{2}\sum_{i=1}^{n} w_i (\mathbf{x}_i^T \theta(\mathbf{x}_0) - y_i)^2$$

$$= \arg\min \frac{1}{2}\sum_{i=1}^{n} K_\lambda(x_i, x_0)(\mathbf{x}_i^T \theta(\mathbf{x}_0) - y_i)^2$$

$$\hat{f}(x_0) = \mathbf{x}_0^T \theta^*(\mathbf{x}_0)$$

e.g. $\hat{f}(x_0) =$

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

Extra: Solution of Locally weighted linear/NonLinearBasis regression

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

Locally weighted LR : $(X^T W_{x_0} X)^{-1} X^T W_{x_0} \hat{y}$

LWR $\theta^*(\mathbf{x}_0) = (B^T W(x_0) B)^{-1} B^T W(x_0) y$

Locally weighted e.g. Polynomial Regression $X \to B$

versus

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

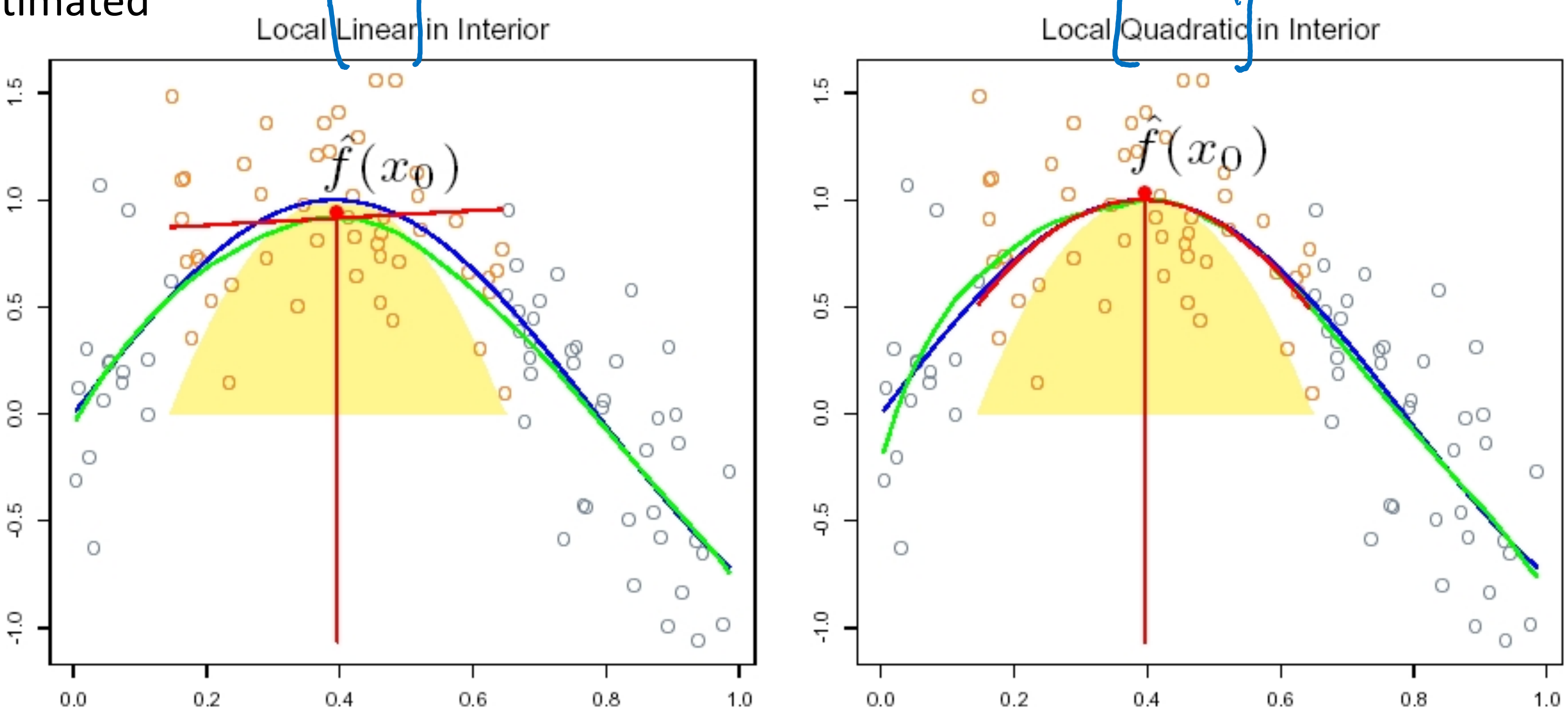
More ➔ Local Weighted Polynomial Regression

- ~~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



Extra: Parametric vs. non-parametric

- Locally weighted linear regression is a **non-parametric** algorithm.

$f(x_i) = x_i^T \theta^*$

- 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;
  - 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.

$f(x_?) = x_?^T \theta^*(x_?)$

(3) Locally Weighted / Kernel Linear Regression

Task	Regression
Representation	Y = Weighted linear sum of X’ s
Score Function	Weighted SSE
Search/Optimization	Linear algebra
Models, Parameters	Local Regression coefficients (conditioned on each test point)

$$\boxed{\theta^*(\mathbf{x}_0) = (B^T W(x_0) B)^{-1} B^T W(x_0) y}$$

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

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

References

- Big thanks to Prof. Eric Xing @ CMU for allowing me to reuse some of his slides

❑ Prof. Nando de Freitas’s tutorial slide

❑ Prof. Andrew Moore’s slides @ CMU