UVA CS 4501: Machine Learning

Lecture 5: Non-Linear Regression Models

Dr. Yanjun Qi

University of Virginia
Department of Computer Science

2/6/18
Where are we?

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

- Regression Models Beyond Linear
  - LR with non-linear basis functions
    - Instance-based Regression: K-Nearest Neighbors
    - Locally weighted linear regression
    - Regression trees and Multilinear Interpolation (later)
LR with non-linear basis functions

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

\[ \hat{y} = \theta^T x \]

\[ \hat{y} = \theta_0 + \sum_{j=1}^{m} \theta_j \varphi_j(x) = \theta^T \varphi(x) \]
LR with non-linear basis functions

• We are free to design basis functions (e.g., non-linear features):

  Here \( \varphi_j(x) \) are fixed basis functions (also define \( \varphi_0(x) = 1 \))

• E.g.: polynomial regression:

\[
\varphi(x) := \begin{bmatrix} 1, x, x^2 \end{bmatrix}^T
\]
e.g. (1) polynomial regression

\[ \hat{y} = \theta^T \mathbf{x} \]

\[ \hat{y} = \theta^T \varphi(\mathbf{x}) \]

\[ \theta^* = (X^T X)^{-1} X^T \hat{y} \]

\[ \theta^* = (\varphi^T \varphi)^{-1} \varphi^T \hat{y} \]

\[ \varphi(x) := \begin{bmatrix} 1, x, x^2 \end{bmatrix}^T \]
e.g. (1) polynomial regression

\[ \hat{y} = \theta^T x \]

\[ \hat{y} = \theta^T \varphi(x) \]

\[ \varphi(x) = [1, x_1, x_2, x_1^2, x_2^2] \]

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
  \[ \varphi_j(x) = x^{j-1} \]
  \[ \begin{bmatrix} 1, x, x^2, x^3, \ldots, x^d \end{bmatrix} \]

  – 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
Many Possible Basis functions

**RBF**

**Sigmoid**

<table>
<thead>
<tr>
<th>Even</th>
<th>Odd</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f(x) = x^6$</td>
<td>$h(x) = x^7$</td>
</tr>
<tr>
<td>$g(x) = x^4$</td>
<td>$g(x) = x^5$</td>
</tr>
<tr>
<td>$h(x) = x^2$</td>
<td>$f(x) = x^3$</td>
</tr>
</tbody>
</table>
e.g. (2) LR with radial-basis functions

- E.g.: LR with RBF regression:

\[ \hat{y} = \theta_0 + \sum_{j=1}^{m} \theta_j \phi_j(x) = \phi(x)^T \theta \]

\[ \phi(x) := \begin{bmatrix} 1, K_{\lambda_1}(x,r_1), K_{\lambda_2}(x,r_2), K_{\lambda_3}(x,r_3), K_{\lambda_4}(x,r_4) \end{bmatrix}^T \]

\[ \theta^* = \left( \phi^T \phi \right)^{-1} \phi^T \hat{y} \]
RBF = radial-basis function: a function which depends only on the radial distance from a centre point

Gaussian RBF ➔

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

as distance from the center \( r \) increases, the output of the RBF decreases

1D case

2D case
\[ K_\lambda(x,r) = \exp \left( -\frac{(x-r)^2}{2\lambda^2} \right) \]

<table>
<thead>
<tr>
<th>( x = )</th>
<th>( K_\lambda(x,r) = )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( r )</td>
<td>( 1 )</td>
</tr>
<tr>
<td>( r + \lambda )</td>
<td>( 0.6065307 )</td>
</tr>
<tr>
<td>( r + 2\lambda )</td>
<td>( 0.1353353 )</td>
</tr>
<tr>
<td>( r + 3\lambda )</td>
<td>( 0.0001234098 )</td>
</tr>
</tbody>
</table>
e.g. another Linear regression with 1D RBF basis functions (assuming 3 predefined centres and width)

\[\varphi(x) := \begin{bmatrix} 1, K_{\lambda_1}(x, r_1), K_{\lambda_2}(x, r_2), K_{\lambda_3}(x, r_3) \end{bmatrix}^T\]

\[\theta^* = \left(\varphi^T \varphi\right)^{-1} \varphi^T \bar{y}\]
e.g. a LR with 1D RBFs (3 predefined centres and width)

• 1D RBF

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

• After fit:

\[ y^{\text{est}} = 2\phi_1(x) + 0.05\phi_2(x) + 0.5\phi_3(x) \]
e.g. Even more possible Basis Func?
Two main issues:

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

• How to choose the model order,
  – E.g. what polynomial degree for polynomial regression
  – E.g., where to put the centers for the RBF kernels? How wide?
e.g. 2D Good and Bad RBF Basis

• A good 2D RBF

• Two bad 2D RBFs
Issue: Overfitting and underfitting

Under fit
Looks good
Over fit

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

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

- Regression Models Beyond Linear
  - LR with non-linear basis functions
  - Instance-based Regression: K-Nearest Neighbors
  - Locally weighted linear regression
  - Regression trees and Multilinear Interpolation (later)
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
$\hat{y} = \frac{1}{k} \sum_{i \in N_k(x_0)} y_i$
K=1-Nearest Neighbor (1D input)
K-Nearest Neighbor

Task
- Representation
- Score Function
- Search/Optimization
- Models, Parameters

Regression/classification
- Local Smoothness
- NA
- Training Samples

Dr. Yanjun Qi / UVA CS 6316 / f16

2/6/18
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

\[ \hat{y} = \frac{1}{K} \sum_{i \in N_k(x_0)} W_i y_i \]
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

- Regression Models Beyond Linear
  - LR with non-linear basis functions
  - Instance-based Regression: K-Nearest Neighbors
  - Locally weighted linear regression
  - Regression trees and Multilinear Interpolation (later)
Locally weighted regression

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

\[ K_\lambda(x_i, x_0) \]

**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 regression

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 regression

\[ f(x_0) = \hat{\theta}_0(x_0) + \hat{\theta}_1(x_0)x_0 \]

A linear function (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

\[ J(\theta) = \frac{1}{2} \sum_{i=1}^{n} (x_i^T \theta - y_i)^2 \]

now we fit to minimize

\[ J(\theta) = \frac{1}{2} \sum_{i=1}^{n} w_i (x_i^T \theta - y_i)^2 \]

where \( x_0 \) is the query point for which we'd like to know its corresponding \( y \)

\[ w_i = K_\lambda (x_i, x_0) = \exp \left( -\frac{(x_i - x_0)^2}{2\lambda^2} \right) \]
Locally weighted linear regression

We fit $\theta$ to minimize

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

$w_i$ comes from:

$$w_i = K_\lambda(x_i, x_0) = \exp\left(-\frac{(x_i - x_0)^2}{2\lambda^2}\right)$$

- $x_0$ 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

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

$$\hat{f}(x_0) = \hat{\theta}_0(x_0) + \hat{\theta}_1(x_0)x_0$$
Locally weighted linear regression

Separate weighted least square error minimization at each target point $x_0$:

$$\theta^*(x_0) = \arg\min \frac{1}{2} \sum_{i=1}^{n} w_i (x_i^T \theta(x_0) - y_i)^2$$

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

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

e.g. $\hat{f}(x_0) = \hat{\alpha}(x_0) + \hat{\beta}(x_0)x_0$
Extra: Solution of Locally weighted linear/NonLinearBasis regression

\[ W_{N \times N}(x_0) = \text{diag}(K_\lambda(x_0, x_i)), i = 1, \ldots, N \]

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

LWR \[ \theta^*(x_0) = (B^T W(x_0) B)^{-1} B^T W(x_0) y \]

Locally weighted e.g. Polynomial Regression \( X \rightarrow B \)

LR \[ \theta^* = (X^T X)^{-1} X^T \bar{y} \]
More → Local Weighted Polynomial Regression

- Local polynomial fits of any degree $d$

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

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

Blue: true
Green: estimated

Local Linear in Interior
Local Quadratic in Interior
Extra: Parametric vs. non-parametric

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

\[
\hat{f}(x_i) = x_i^T \theta^*
\]

• 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 \( \theta \)), 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.

\[
\hat{f}(x_i) = x_i^T \theta^*(x_i)
\]
(3) Locally Weighted / Kernel Linear Regression

- **Task**
  - Representation
  - Score Function
  - Search/Optimization
  - Models, Parameters

- **Regression**
  - Y = Weighted linear sum of X’s
  - Weighted SSE
  - Linear algebra
  - Local Regression coefficients (conditioned on each test point)

\[
\theta^*(x_0) = (B^T W(x_0) B)^{-1} B^T W(x_0) y
\]

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

\[
\hat{f}(x_0) = \hat{\alpha}(x_0) + \hat{\beta}(x_0)x_0
\]
Today Recap

- Regression Models Beyond Linear
  - LR with non-linear basis functions
  - Instance-based Regression: K-Nearest Neighbors
  - Locally weighted linear regression
  - Regression trees and Multilinear Interpolation (later)
References

• Big thanks to Prof. Eric Xing @ CMU for allowing me to reuse some of his slides
  ❏ Prof. Nando de Freitas’s tutorial slide