UVA CS 6316 – Fall 2015 Graduate: Machine Learning

Lecture 7: Review of Regression

Dr. Yanjun Qi

University of Virginia

Department of Computer Science

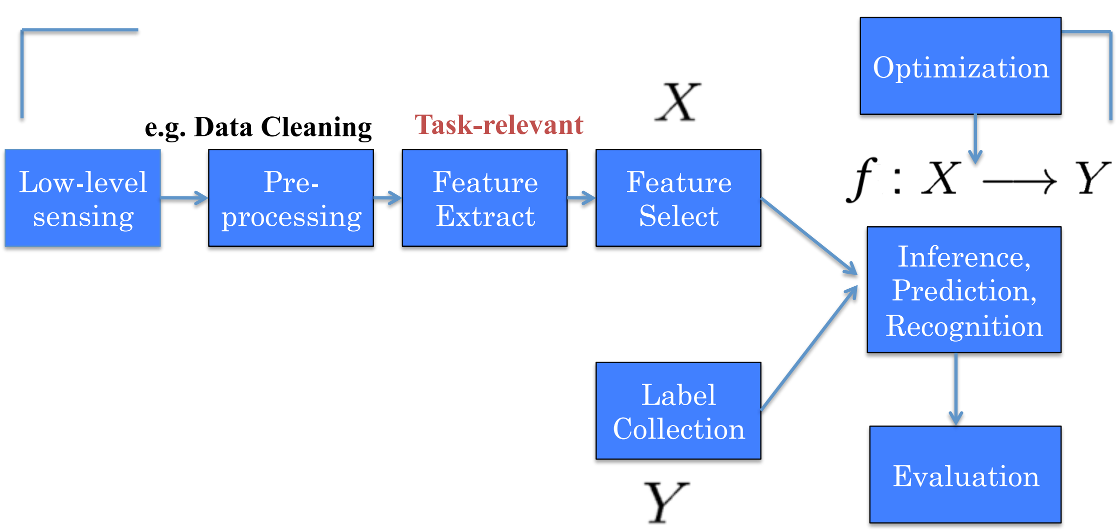
Where are we ? ➔ Five major sections of this course

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

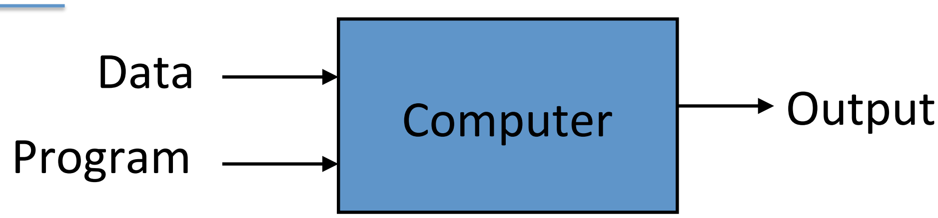
Today

- ❑ Review of basic pipeline
- ❑ Review of regression models
  - Linear regression (LR)
  - LR with non-linear basis functions
  - Locally weighted LR
  - LR with Regularizations
- ❑ Model Selection

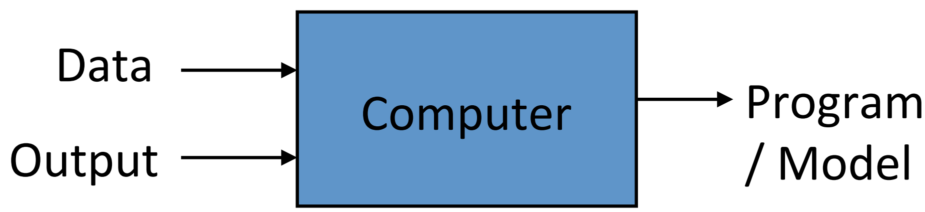
A Typical Machine Learning Pipeline



Traditional Programming



Machine Learning



e.g. SUPERVISED LEARNING

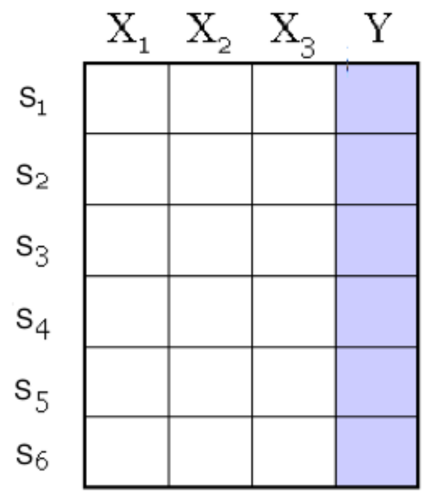
$$\boxed{f : X \longrightarrow Y}$$

- Find function to map input space X to output space Y

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

KEY

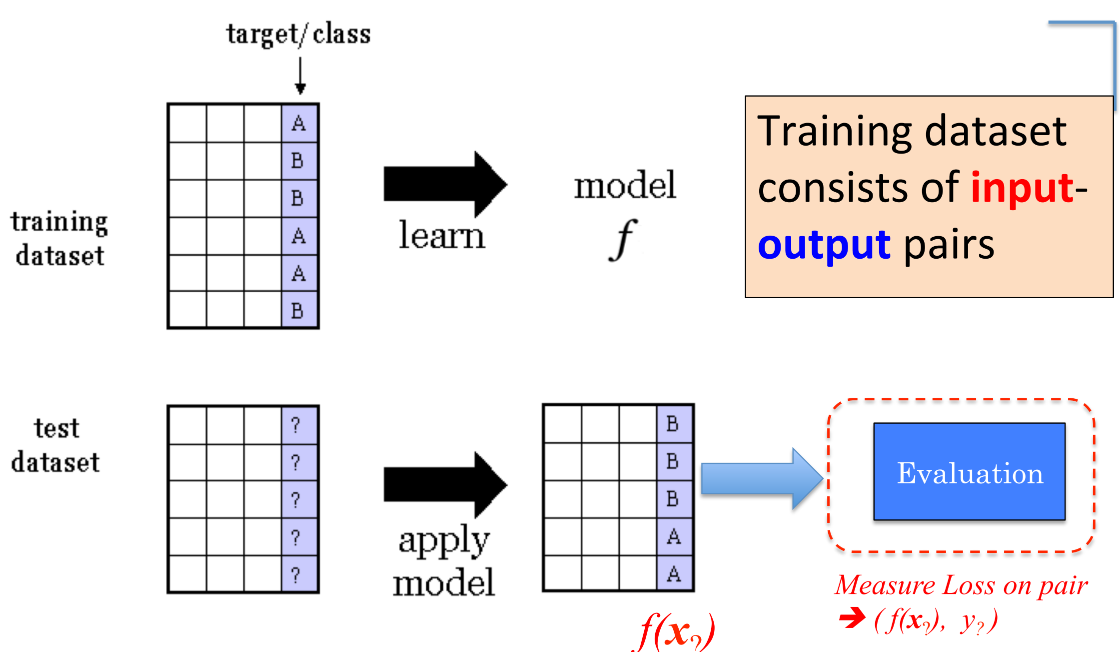
A Dataset



$$f : X \longrightarrow Y$$

- **Data**/*points/instances/examples/samples/records*: [rows]
- **Features**/*attributes/dimensions/independent variables/covariates/predictors/regressors*: [columns, except the last]
- **Target**/*outcome/response/label/dependent variable*: special column to be predicted [last column]

SUPERVISED LEARNING



Evaluation Metric
e.g. for linear regression models

training dataset

$$\mathbf{X}_{train} = \begin{bmatrix} -- & \mathbf{x}_1^T & -- \\ -- & \mathbf{x}_2^T & -- \\ \vdots & \vdots & \vdots \\ -- & \mathbf{x}_n^T & -- \end{bmatrix} \qquad \vec{y}_{train} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

test dataset

$$\mathbf{X}_{test} = \begin{bmatrix} -- & \mathbf{x}_{n+1}^T & -- \\ -- & \mathbf{x}_{n+2}^T & -- \\ \vdots & \vdots & \vdots \\ -- & \mathbf{x}_{n+m}^T & -- \end{bmatrix} \qquad \vec{y}_{test} = \begin{bmatrix} y_{n+1} \\ y_{n+2} \\ \vdots \\ y_{n+m} \end{bmatrix}$$

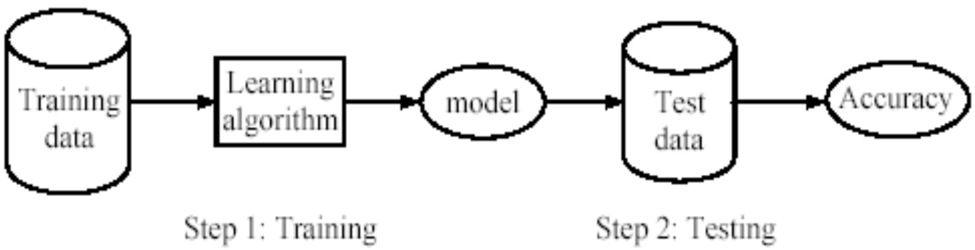
Evaluation Metric
e.g. for linear regression models

- Testing MSE (mean squared error) to report:

$$J_{test} = \frac{1}{m} \sum_{i=n+1}^{n+m} (\mathbf{x}_i^T \theta^* - y_i)^2$$

Evaluation Choice-I:

- ✓ Training (Learning): Learn a model using the training data
- ✓ Testing: Test the model using unseen test data to assess the model accuracy



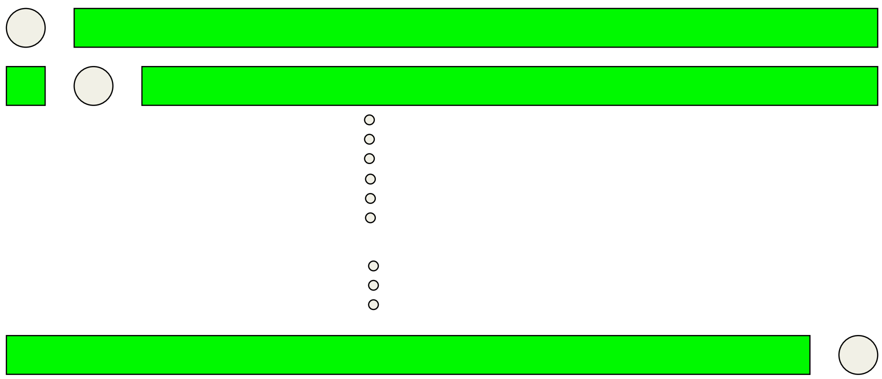
$$Accuracy = \frac{\text{Number of correct classifications}}{\text{Total number of test cases}},$$

Evaluation Choice-II:
e.g. 10 fold Cross Validation

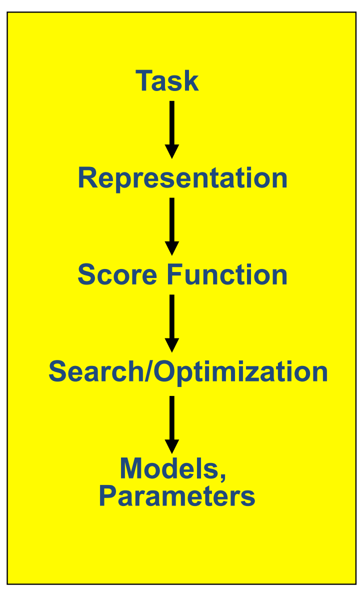
- Divide data into 10 equal pieces
- 9 pieces as training set, the rest 1 as test set
- Collect the scores from the diagonal

model	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10
1	train	train	train	train	train	train	train	train	train	test
2	train	train	train	train	train	train	train	train	test	train
3	train	train	train	train	train	train	train	test	train	train
4	train	train	train	train	train	train	test	train	train	train
5	train	train	train	train	train	test	train	train	train	train
6	train	train	train	train	test	train	train	train	train	train
7	train	train	train	test	train	train	train	train	train	train
8	train	train	test	train	train	train	train	train	train	train
9	train	test	train	train	train	train	train	train	train	train
10	test	train	train	train	train	train	train	train	train	train

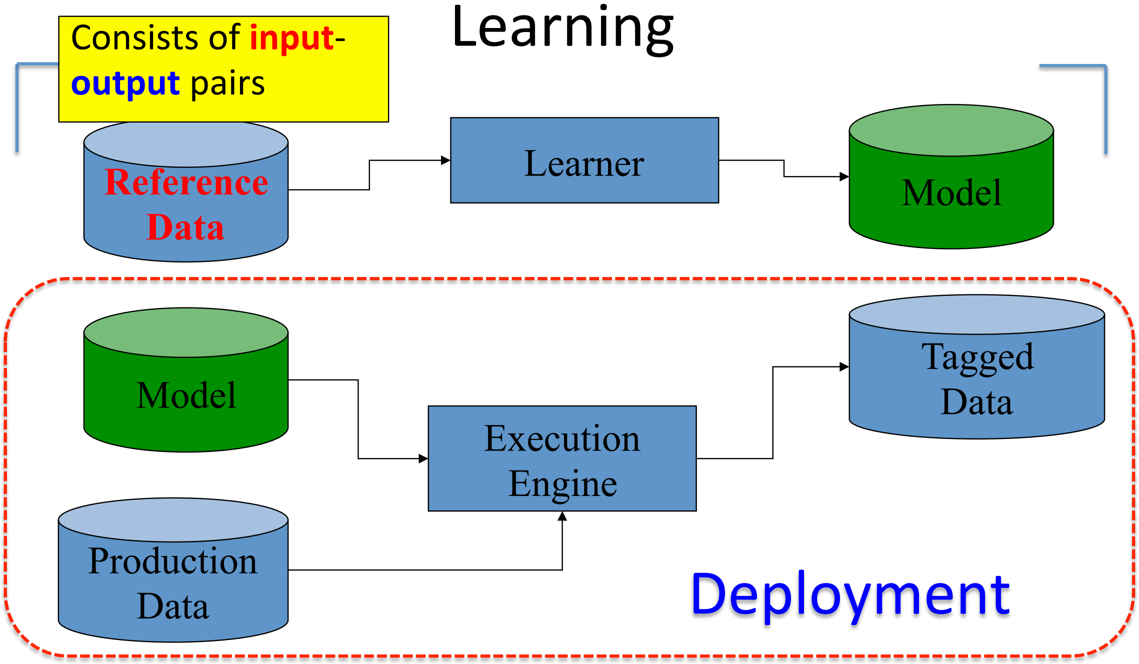
e.g. Leave-one-out (n-fold cross validation)



Machine Learning in a Nutshell



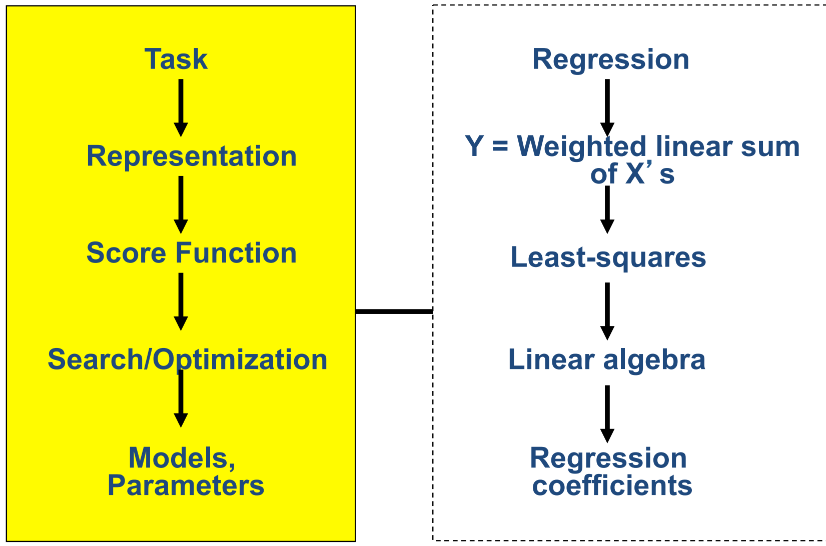
An Operational Model of Machine Learning



Today

- ❑ Review of basic pipeline
- ❑ Review of regression models
  - – Linear regression (LR)
  - – LR with non-linear basis functions
  - – Locally weighted LR
  - – LR with Regularizations
- ❑ Model Selection

(1) Multivariate Linear Regression



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

(1) Linear Regression (LR)

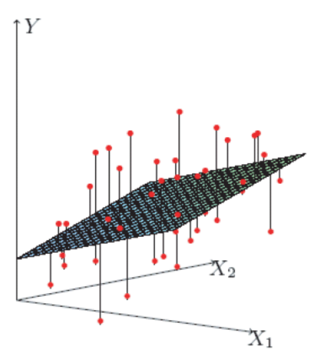
$$\boxed{f} : X \longrightarrow Y$$

➔ e.g. Linear Regression Models

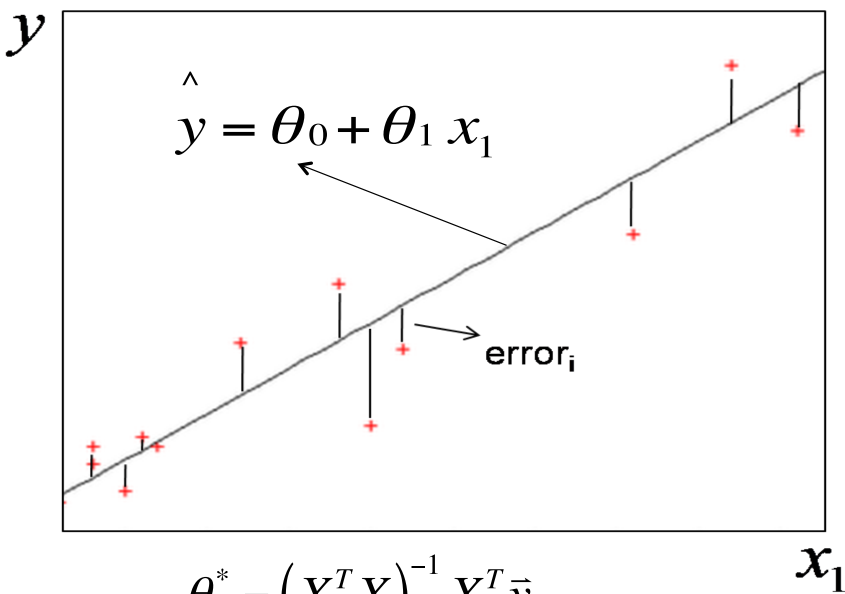
$$\hat{y} = f(x) = \theta_0 + \theta_1 x^1 + \theta_2 x^2$$

➔ To minimize the "least square" cost function:

$$J(\theta) = \frac{1}{2} \sum_{i=1}^{n} (\hat{y}_i(\vec{x}_i) - y_i)^2 \rightarrow \theta$$



Linear regression (1D example)

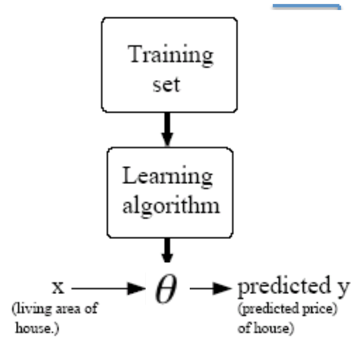


$$\theta^* = \left(X^T X\right)^{-1} X^T \vec{y}$$

Our goal:

- We can represent the whole Training set:

$$\mathbf{X} = \begin{bmatrix} -- & \mathbf{x}_1^T & -- \\ -- & \mathbf{x}_2^T & -- \\ \vdots & \vdots & \vdots \\ -- & \mathbf{x}_n^T & -- \end{bmatrix} = \begin{bmatrix} x_1^0 & x_1^1 & \dots & x_1^{p-1} \\ x_2^0 & x_2^1 & \dots & x_2^{p-1} \\ \vdots & \vdots & \vdots & \vdots \\ x_n^0 & x_n^1 & \dots & x_n^{p-1} \end{bmatrix}$$



$$\mathbf{Y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

- Predicted output for each training sample:

$$\begin{bmatrix} f(\mathbf{x}_1^T) \\ f(\mathbf{x}_2^T) \\ \vdots \\ f(\mathbf{x}_n^T) \end{bmatrix} = \begin{bmatrix} \mathbf{x}_1^T\theta \\ \mathbf{x}_2^T\theta \\ \vdots \\ \mathbf{x}_n^T\theta \end{bmatrix} = X\theta$$

Method I: normal equations

- Write the cost function in matrix form:

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

$$\mathbf{X} = \begin{bmatrix} -- & \mathbf{x}_1^T & -- \\ -- & \mathbf{x}_2^T & -- \\ \vdots & \vdots & \vdots \\ -- & \mathbf{x}_n^T & -- \end{bmatrix} \qquad \vec{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

To minimize $J(\theta)$, take derivative and set to zero:

$$\Rightarrow \boxed{X^T X\theta = X^T\vec{y}}$$

The normal equations

$$\Downarrow$$

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

Method II: LR with batch Steepest descent / Gradient descent

For the t-th epoch

$$\theta_t = \theta_{t-1} - \alpha\nabla J(\theta_{t-1})$$

$$\nabla_\theta J = \left[\frac{\partial}{\partial\theta_1}J, \ldots, \frac{\partial}{\partial\theta_k}J\right]^T = -\sum_{i=1}^{n}(y_i - \mathbf{x}_i^T\theta)\mathbf{x}_i$$

$$\theta_j^{t+1} = \theta_j^{t} + \alpha\sum_{i=1}^{n}(y_i - \vec{\mathbf{x}}_i^T\theta^t)x_i^j$$

– This is as a **batch** gradient descent algorithm

Method III: LR with Stochastic GD →

- From the batch steepest descent rule:

$$\theta_j^{t+1} = \theta_j^{t} + \alpha \sum_{i=1}^{n} (y_i - \vec{\mathbf{x}}_i^T \theta^t) x_i^j$$

- For a single training point, we have:

$$\Rightarrow \quad \theta^{t+1} = \theta^{t} + \alpha (y_i - \vec{\mathbf{x}}_i^T \theta^t) \vec{\mathbf{x}}_i$$

  - a "**stochastic**", "**coordinate**" descent algorithm
  - This can be used as an **on-line** algorithm

Method IV: Newton's method for optimization

- The most basic second-order optimization algorithm
- Updating parameter with

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \mathbf{H}_K^{-1} \mathbf{g}_k$$

$$\Rightarrow \quad \theta^{t+1} = \theta^t - H^{-1} \nabla f(\theta)$$

$$= \theta^t - (X^T X)^{-1} [X^T X \theta^t - X^T \vec{y}]$$

WHY ??? Normal Eq?

$$= (X^T X)^{-1} X^T \vec{y}$$

Newton's method for Linear Regression

(2) Multivariate Linear Regression with basis Expansion

Task	Regression
↓	↓
Representation	Y = Weighted linear sum of (X basis expansion)
↓	↓
Score Function	Least-squares
↓	↓
Search/Optimization	Linear algebra
↓	↓
Models, Parameters	Regression coefficients

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

(2) LR with polynomial basis functions

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

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

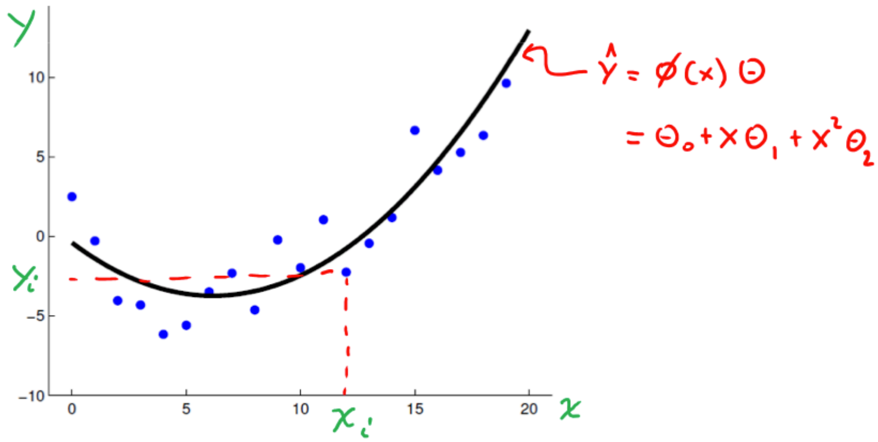
- E.g.: polynomial regression:

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

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

e.g. **polynomial regression**

For example, $\phi(x) = [1, x, x^2]$



Dr. Nando de Freitas's tutorial slide

LR with radial-basis functions

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

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

- E.g.: LR with RBF regression: $K_\lambda(\underline{x}, r) = \exp\left(-\frac{(\underline{x}-r)^2}{2\lambda^2}\right)$

$$\varphi(x) := \left[1, K_{\lambda=1}(x,1), K_{\lambda=1}(x,2), K_{\lambda=1}(x,4)\right]$$

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

(3) Locally Weighted / Kernel Regression

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

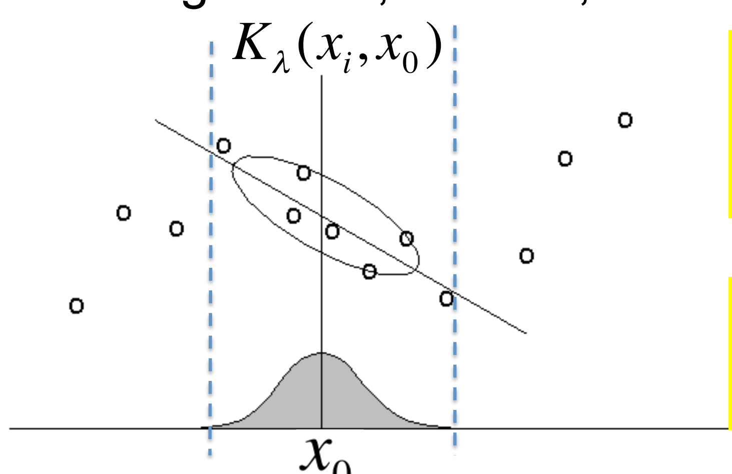
Regression → Y = Weighted linear sum of X' s → Weighted Least-squares → Linear algebra → Local Regression coefficients (conditioned on each test point)

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

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

(3) Locally weighted regression

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



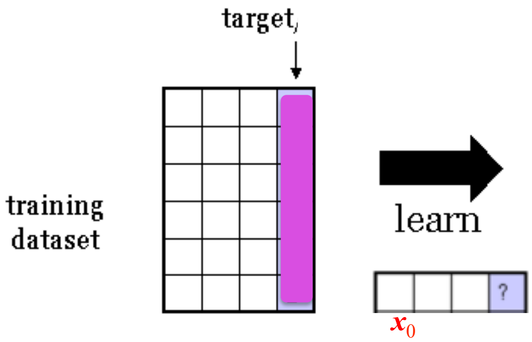
linear_func(x)->y
➔
could represent only the neighbor region of x_0

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.

LEARNING of Locally weighted linear regression



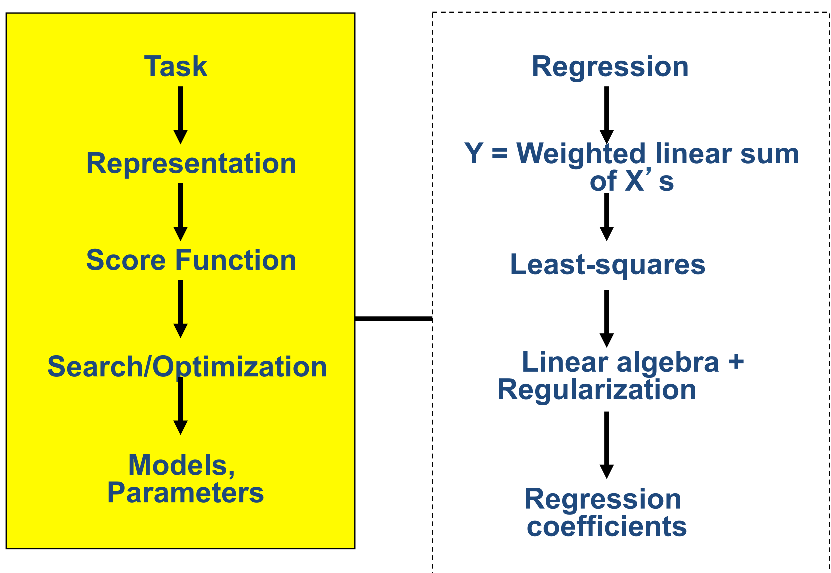
model

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

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

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

(4) Regularized multivariate linear regression



$$\min J(\beta) = \sum_{i=1}^{n}\left(Y - \hat{Y}\right)^2 + \lambda \sum_{j=1}^{p} \beta_j^2$$

(4) LR with Regularizations / Regularized multivariate linear regression

- Basic model $\hat{Y} = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \cdots + \hat{\beta}_p x_p$
- LR estimation: $\min J(\beta) = \sum \left(Y - \hat{Y}\right)^2$
- LASSO estimation: $\min J(\beta) = \sum_{i=1}^{n} \left(Y - \hat{Y}\right)^2 + \lambda \sum_{j=1}^{p} |\beta_j|$
- Ridge regression estimation: $\min J(\beta) = \sum_{i=1}^{n} \left(Y - \hat{Y}\right)^2 + \lambda \sum_{j=1}^{p} \beta_j^2$

33/54

Error on data + Regularization

LR with Regularizations / Ridge Estimator

$$\hat{Y} = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \cdots + \hat{\beta}_p x_p$$

$$\beta^* = \left(X^T X + \lambda I\right)^{-1} X^T \vec{y}$$

- The ridge estimator is solution from RSS (regularized sum of square errors)

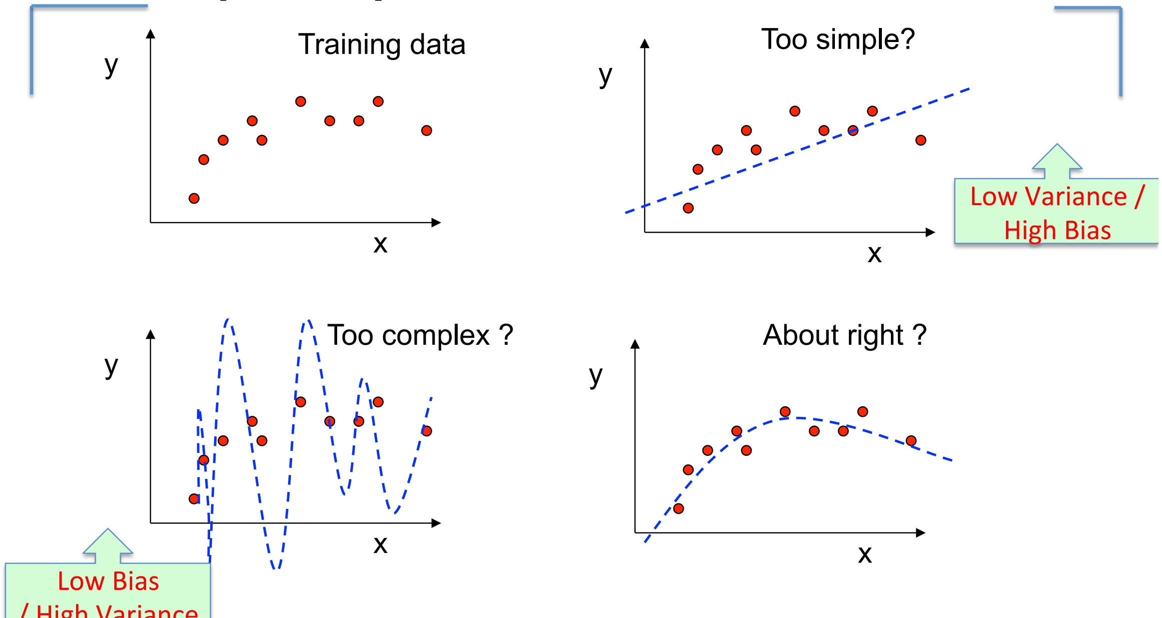
$$\hat{\beta}^{ridge} = \arg\min J(\beta) = \arg\min (y - X\beta)^T (y - X\beta) + \lambda \beta^T \beta$$

to minimize, take derivative and set to zero

Today

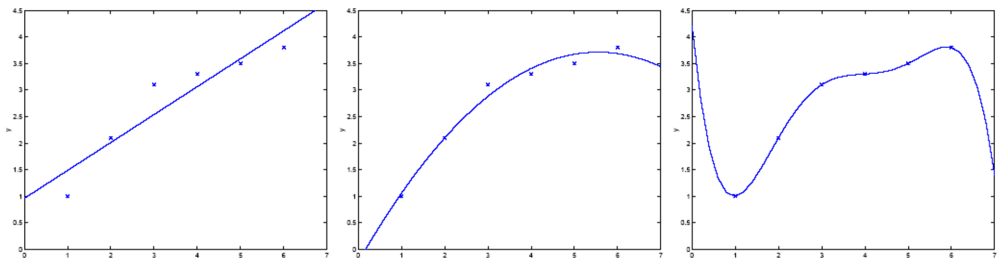
- ❑ Review of basic pipeline
- ❑ Review of regression models
  - – Linear regression (LR)
  - – LR with non-linear basis functions
  - – Locally weighted LR
  - – LR with Regularizations
- ❑ Review of Model Selection

Regression: Complexity versus Goodness of Fit



What ultimately matters: ***GENERALIZATION***

Which function *f to choose*? Many possible choices , e.g. LR with polynomial basis functions



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

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

Choose *f that generalizes well !*

Which kernel width to choose ? e.g. locally weighted LR

Choose *kernel width that generalizes well !*

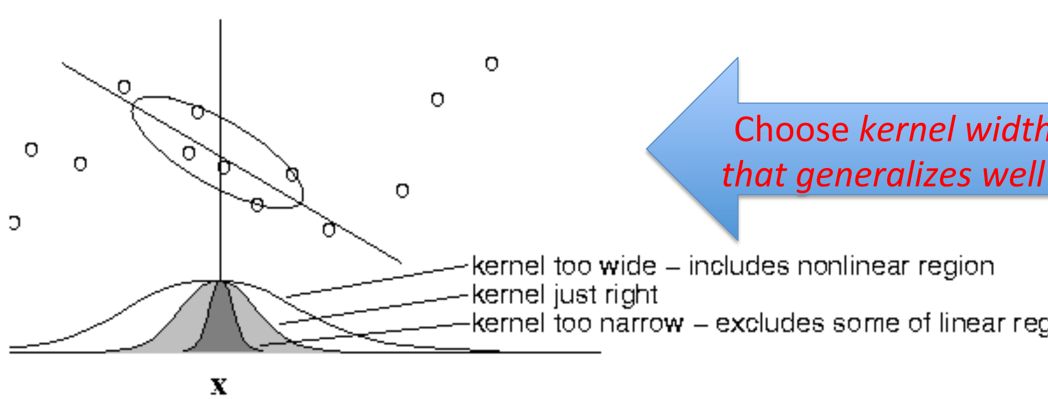
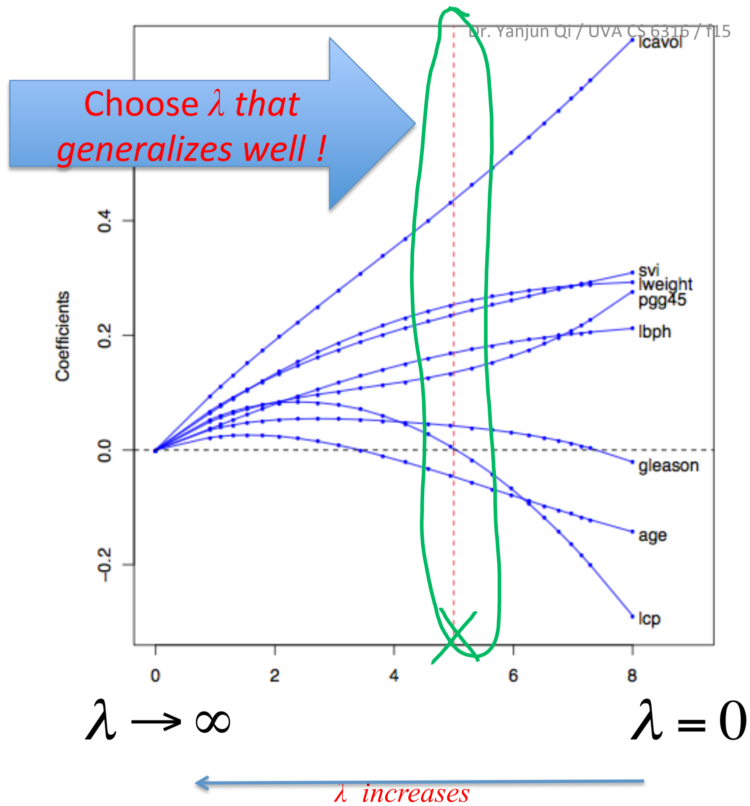


Figure 3: The estimator variance is minimized when the kernel includes as many training points as can be accommodated by the model. Here the linear LOESS model is shown. Too large a kernel includes points that degrade the fit; too small a kernel neglects points that increase confidence in the fit.

an example (ESL Fig3.8),

Ridge Regression

when varying λ, how θ_j varies.



Regularization path of a Lasso Estimator

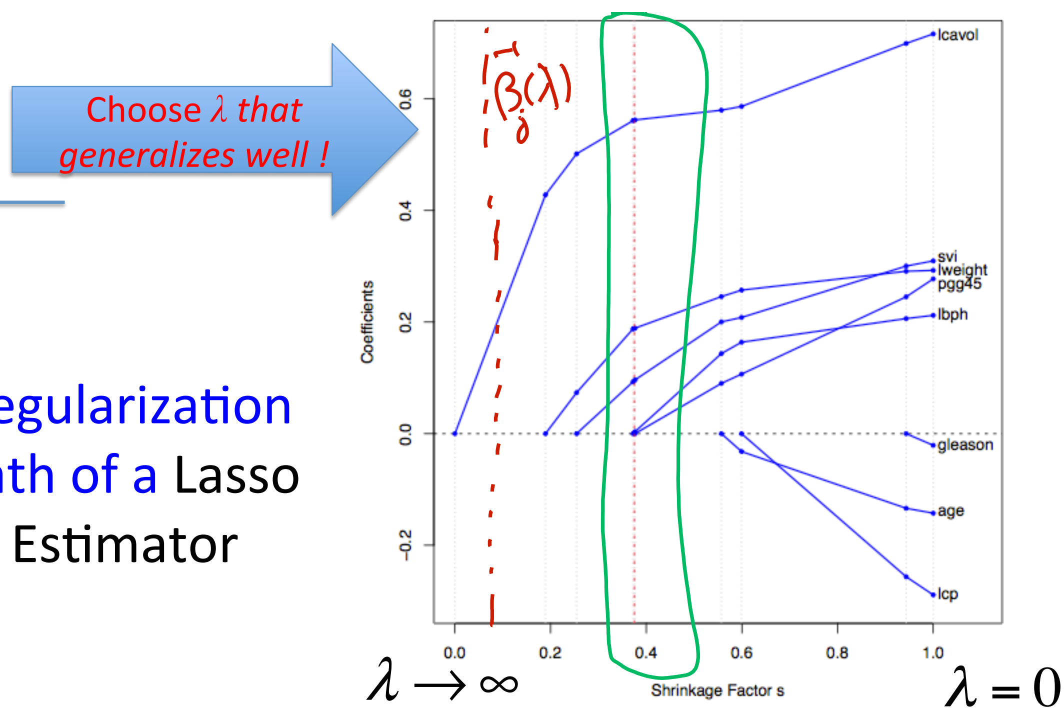


FIGURE 3.10. *Profiles of lasso coefficients, as the tuning parameter t is varied. Coefficients are plotted versus $s = t/\sum_1^p |\hat{\beta}_j|$. A vertical line is drawn at $s = 0.36$, the value chosen by cross-validation. Compare Figure 3.8 on page 65; the lasso profiles hit zero, while those for ridge do not. The profiles are piece-wise linear, and so are computed only at the points displayed; see Section 3.4.4 for details.*

References

- ❑ Big thanks to Prof. Eric Xing @ CMU for allowing me to reuse some of his slides
- ❑ Prof. Alexander Gray's slides