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

- Low-level sensing
- Pre-processing
- Feature Extract
- Feature Select
- Inference, Prediction, Recognition
- Label Collection
- Optimization
- Evaluation

\[ f : X \rightarrow Y \]
Tradional Programming

Data → Computer → Output

Program → Computer

Machine Learning

Data → Computer → Program / Model

Output → Program / Model

9/21/15

Dr. Yanjun Qi / UVA CS 6316 / f15

DR.

Tradional Programming

Data → Computer → Output

Program → Computer

Machine Learning

Data → Computer → Program / Model

Output → Program / Model

e.g. SUPERVISED LEARNING

\[ f : X \longrightarrow Y \]

- 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
• **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**

Training dataset consists of **input-output** pairs

Measure Loss on pair 

\( f(x), y) \)
Evaluation Metric

e.g. for linear regression models

\[
X_{\text{train}} = \begin{bmatrix}
- & x_1^T & - \\
- & x_2^T & - \\
\vdots & \vdots & \vdots \\
- & x_n^T & -
\end{bmatrix}, \quad \tilde{y}_{\text{train}} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}
\]

\[
X_{\text{test}} = \begin{bmatrix}
- & x_{n+1}^T & - \\
- & x_{n+2}^T & - \\
\vdots & \vdots & \vdots \\
- & x_{n+m}^T & -
\end{bmatrix}, \quad \tilde{y}_{\text{test}} = \begin{bmatrix} y_{n+1} \\ y_{n+2} \\ \vdots \\ y_{n+m} \end{bmatrix}
\]

• Testing MSE (mean squared error) to report:

\[
J_{\text{test}} = \frac{1}{m} \sum_{i=n+1}^{n+m} \left( x_i^T \theta^* - y_i \right)^2
\]
**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

---

### Step 1: Training

### Step 2: Testing

\[
\text{Accuracy} = \frac{\text{Number of correct classifications}}{\text{Total number of test cases}}
\]

---

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

Machine Learning in a Nutshell

- Task
- Representation
- Score Function
- Search/Optimization
- Models, Parameters
An Operational Model of Machine Learning

- Consists of input-output pairs
- Reference Data
- Learner
- Model
- Model
- Execution Engine
- Tagged Data
- Deployment
- Production Data

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)

\[ \hat{y} = f(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 \]

\( \rightarrow \) e.g. Linear Regression Models

\( \rightarrow \) To minimize the “least square” cost function:

\[ J(\theta) = \frac{1}{2} \sum_{i=1}^{n} (\hat{y}_i(\bar{x}_i) - y_i)^2 \]
Linear regression (1D example)

\[ y = \theta_0 + \theta_1 x_1 \]

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

• We can represent the whole Training set:

\[
X = \begin{bmatrix}
- & x_1^T & - & \\
- & x_2^T & - & \\
\vdots & \vdots & \vdots \\
- & x_n^T & - & \\
\end{bmatrix}
\begin{bmatrix}
x_1^0 & x_1^1 & \cdots & x_1^{p-1} \\
x_2^0 & x_2^1 & \cdots & x_2^{p-1} \\
\vdots & \vdots & \vdots & \vdots \\
x_n^0 & x_n^1 & \cdots & x_n^{p-1} \\
\end{bmatrix}
\]

\[
Y = \begin{bmatrix}
y_1 \\
y_2 \\
\vdots \\
y_n \\
\end{bmatrix}
\]

• Predicted output for each training sample:

\[
\begin{bmatrix}
f(x_1^T) \\
f(x_2^T) \\
\vdots \\
f(x_n^T) \\
\end{bmatrix} = \begin{bmatrix}
x_1^T \theta \\
x_2^T \theta \\
\vdots \\
x_n^T \theta \\
\end{bmatrix} = X^T \theta
\]
Method I: normal equations

- Write the cost function in matrix form:

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

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

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

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

\[
\Rightarrow \quad X^T X \theta = X^T \bar{y}
\]

The normal equations

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

Method II: LR with batch Steepest descent / Gradient descent

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

For the t-th epoch

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

\[
\theta_{j+1}^t = \theta_j^t + \alpha \sum_{i=1}^{n} (y_i - \bar{x}_i^T \theta^t) x_i^j
\]

- This is as a batch gradient descent algorithm
Method III: LR with Stochastic GD

- From the batch steepest descent rule:
  \[ \theta_j^{t+1} = \theta_j^t + \alpha \sum_{i=1}^n (y_i - \bar{x}_i^T \theta^t) x_i^j \]

- For a single training point, we have:
  \[ \theta^{t+1} = \theta^t + \alpha (y_i - \bar{x}_i^T \theta^t) \bar{x}_i \]

- 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
  \[ \theta_{k+1} = \theta_k - H_K^{-1} g_k \]

- Updating parameter with

  \[ \Rightarrow \quad \theta^{t+1} = \theta^t - H^{-1} \nabla f(\theta) \]
  \[ = \theta^t - (X^T X)^{-1} \left[ X^T (\theta^t - X^T y) \right] \]
  \[ = (X^T X)^{-1} X^T y \]

Newton’s method for Linear Regression
(2) Multivariate Linear Regression with basis Expansion

Task
- Representation
- Score Function
- Search/Optimization

Regression
- Y = Weighted linear sum of (X basis expansion)
- Least-squares
- Linear algebra
- 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:

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

\[ \theta^* = (\varphi^T \varphi)^{-1} \varphi^T \hat{y} \]
**e.g. polynomial regression**

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

\[
\hat{y} = \phi(x) \Theta = \phi_0 + x \phi_1 + x^2 \phi_2
\]

**LR with radial-basis functions**

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

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

- E.g.: LR with RBF regression:

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

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

\[
\theta^* = (\varphi^T \varphi)^{-1} \varphi^T \hat{y}
\]
(3) Locally Weighted / Kernel Regression

Task
- Representation
- Score Function
- Search/Optimization

Regression
- $Y = \text{Weighted linear sum of } X'$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, ...

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

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

- 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

\[
\min J(\beta) = \sum_{i=1}^{n} (Y_i - \hat{Y})^2 + \lambda \sum_{j=1}^{p} \beta_j^2
\]
(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:
  \[ \min J(\beta) = \sum (Y - \hat{Y})^2 \]

• LASSO estimation:
  \[ \min J(\beta) = \sum_{i=1}^{n} (Y - \hat{Y})^2 + \lambda \sum_{j=1}^{p} |\beta_j| \]

• Ridge regression estimation:
  \[ \min J(\beta) = \sum_{i=1}^{n} (Y - \hat{Y})^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \]

Error on data + Regularization

---


dr. Yanjun Qi / UVA CS 6316 / f15

LR with Regularizations / Ridge Estimator

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

\[ \hat{\beta}^* = \left( X^T X + \lambda I \right)^{-1} X^T \hat{y} \]

• 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

$$y = \theta_0 + \theta_1 x$$

$$y = \theta_0 + \theta_1 x + \theta_2 x^2$$

$$y = \sum_{j=0}^{5} \theta_j x^j$$

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

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 Fig 3.8),

Ridge Regression

when varying $\lambda$, how $\theta_j$ varies.

Choose $\lambda$ that generalizes well!

Regularization path of a Lasso Estimator

Choose $\lambda$ that generalizes well!
References

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