UVA CS 4501 - 001 / 6501 – 007 Introduction to Machine Learning and Data Mining

Lecture 17: Review / Bias-Variance Tradeoff

Yanjun Qi / Jane, , PhD

University of Virginia
Department of
Computer Science

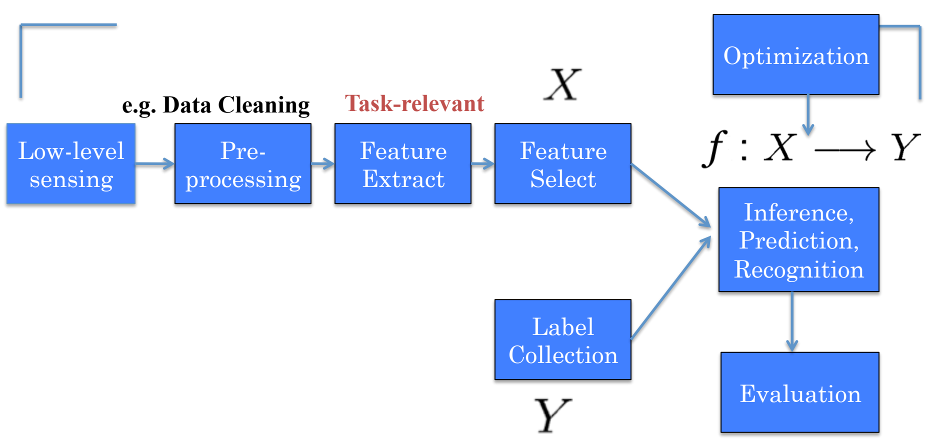
Where are we ? ➔ Five major sections of this course

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

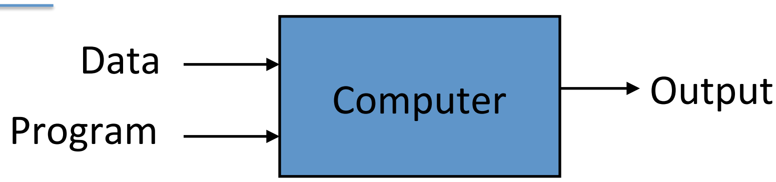
Today

- ❑ Review of basic pipeline
- ❑ Review of regression models
  - Linear regression (LR)
  - LR with non-linear basis functions
  - Locally weighted LR
  - LR with Regularizations
- ❑ Review of classification models
  - Support Vector Machine
  - Bayes Classifier
  - Logistic Regression
  - K-nearest Neighbor
- ❑ Model Selection

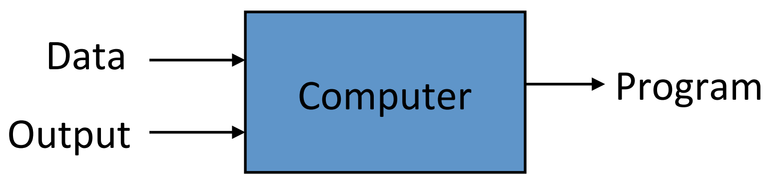
A Typical Machine Learning Pipeline



Traditional Programming



Machine Learning



e.g. SUPERVISED LEARNING

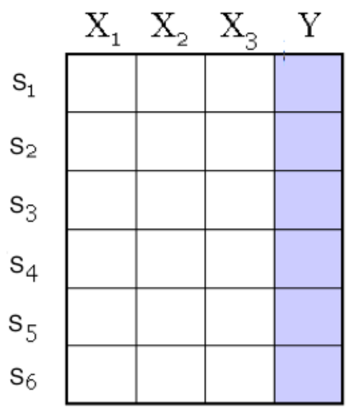
$$f : X \longrightarrow Y$$

- 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

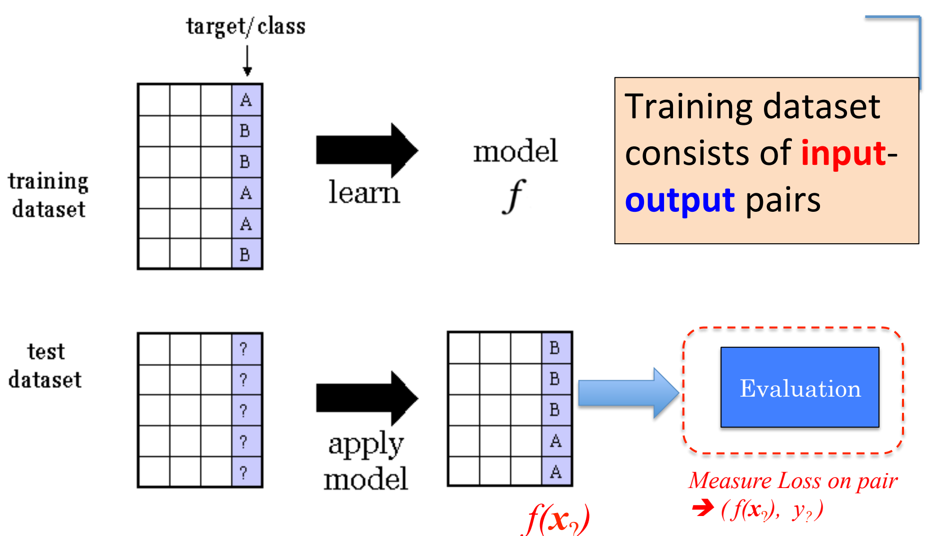
A Dataset



$$f : X \longrightarrow Y$$

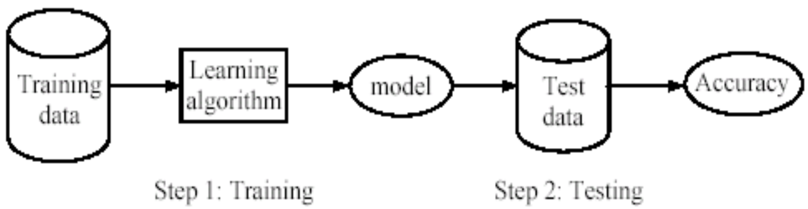
- **Data**/*points/instances/examples/samples/records*: [rows]
- **Features**/*attributes/dimensions/independent variables/covariates/ predictors/regressors*: [columns, except the last]
- **Target**/*outcome/response/label/dependent variable*: special column to be predicted [last column]

SUPERVISED LEARNING



Evaluation Choice-I:

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



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

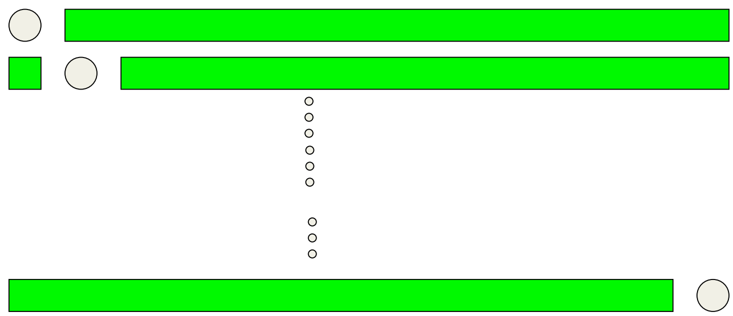
Evaluation Choice-II:

e.g. 10 fold Cross Validation

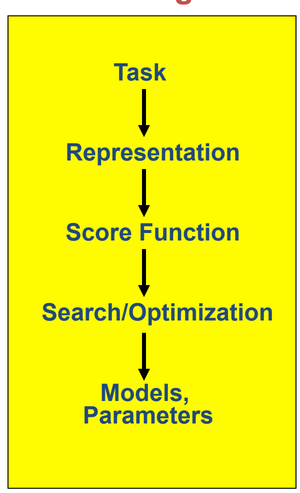
- Divide data into 10 equal pieces
- 9 pieces as training set, the rest 1 as test set
- Collect the scores from the diagonal

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

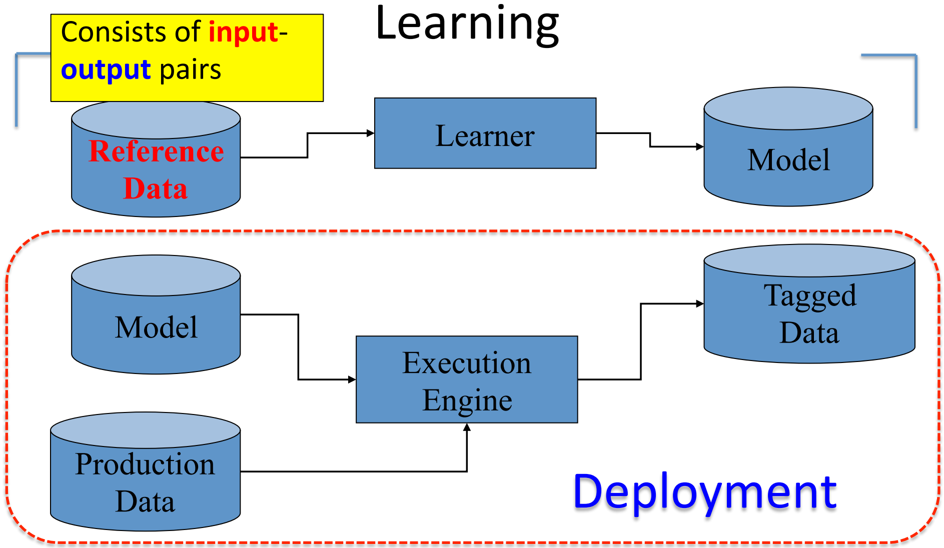
e.g. Leave-one-out (n-fold cross validation)



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
- ❑ Review of classification models
  - Support Vector Machine
  - Bayes Classifier
  - Logistic Regression
  - K-nearest Neighbor
- ❑ Model Selection

(1) Multivariate Linear Regression

Task	→	Regression
Representation	→	Y = Weighted linear sum of X's
Score Function	→	Least-squares
Search/Optimization	→	Linear algebra
Models, Parameters	→	Regression coefficients

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

(1) Linear Regression (LR)

$$f : X \longrightarrow Y$$

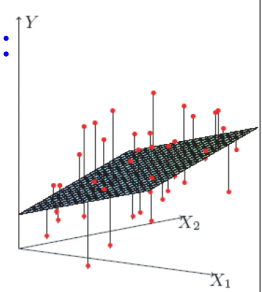
→ e.g. Linear Regression Models

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

→ To minimize the cost function:

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

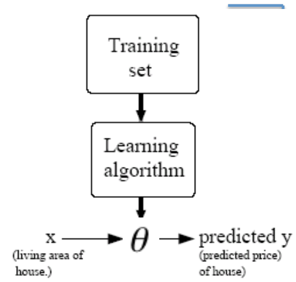
→ θ



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:

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

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

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

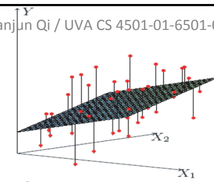
$$\Rightarrow \quad X^T X\theta = X^T\vec{y}$$

The normal equations

$$\Downarrow$$

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

Probabilistic Interpretation of Linear Regression



- 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:

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

- By independence (among samples) assumption:

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

Probabilistic Interpretation of Linear Regression (cont.)

- Hence the log-likelihood is:

$$l(\theta) = n \log \frac{1}{\sqrt{2\pi}\sigma} - \frac{1}{\sigma^2} \frac{1}{2} \sum_{i=1}^{n} (y_i - \theta^T \mathbf{x}_i)^2$$

- Do you recognize the last term?

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

- Thus under independence assumption, residual means square is equivalent to MLE of ϑ !

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

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

$$\theta^{t+1} = \theta^t + \alpha \sum_{i=1}^{n} (y_n - \mathbf{x}_n^T \theta^t) \mathbf{x}_n$$

– 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:

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

  - This is known as the Least-Mean-Square update rule, or the Widrow-Hoff learning rule
  - This is actually a "**stochastic**", "**coordinate**" descent algorithm
  - This can be used as a **on-line** algorithm

Method IV: Newton's method for optimization

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

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

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

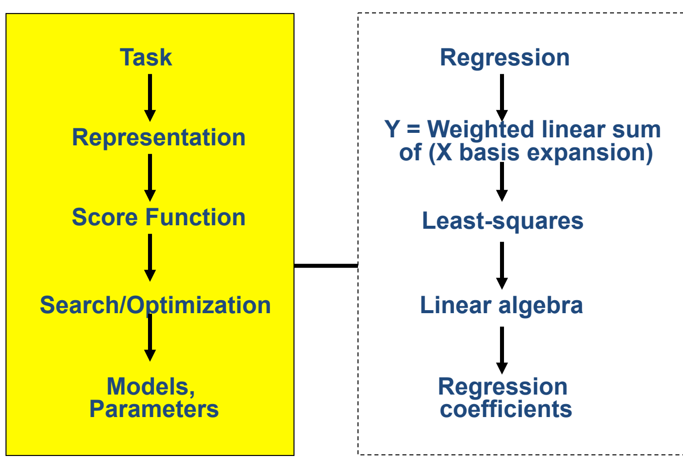
$$= \theta^t - (X^T X)^{-1}[X^T X \theta^t - X^T \vec{y}]$$

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

WHY ??? Normal Eq?

Newton's method for Linear Regression

(2) Multivariate Linear Regression with basis Expansion



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

(2) LR with polynomial basis functions

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

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

where the $\phi_j(x)$ are fixed basis functions (also define $\phi_0(x) = 1$)

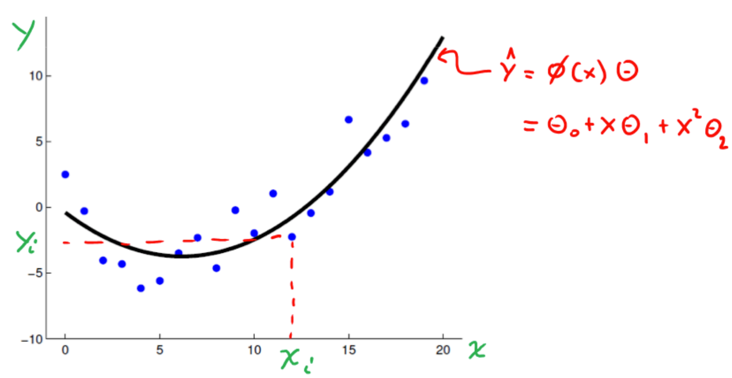
- E.g.: polynomial regression:

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

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

e.g. **polynomial regression**

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



Dr. Nando de Freitas's tutorial slide

(2) LR with radial-basis functions

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

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

where the $\phi_j(x)$ are fixed basis functions (also define $\phi_0(x) = 1$)

- E.g.: LR with RBF regression:

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

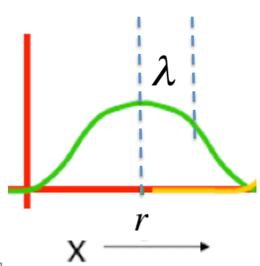
$$\theta^* = \left(\varphi^T \varphi\right)^{-1} \varphi^T \vec{y}$$

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

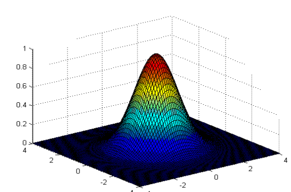
Gaussian RBF →

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

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



p=1

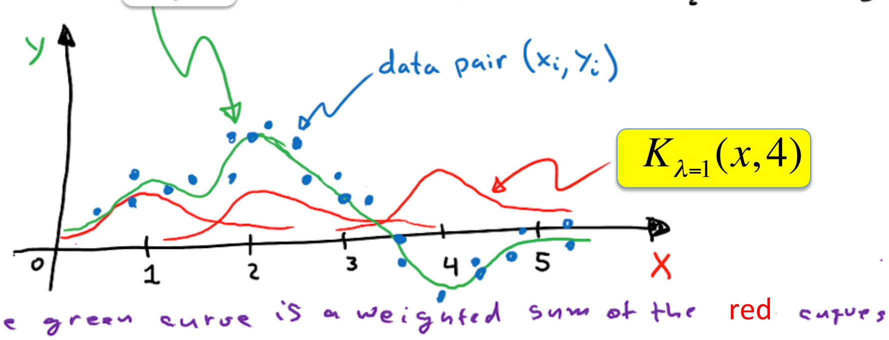


P=2

(2) Linear regression with RBF basis functions (predefined centres)

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

$$\hat{y} = \theta_0 + e^{-\|x-1\|^2}\theta_1 + e^{-\|x-2\|^2}\theta_2 + e^{-\|x-4\|^2}\theta_3$$



Dr. Nando de Freitas's tutorial slide

$$K_{\lambda=1}(x,1) = e^{-\|x-1\|^2}$$

$$K_{\lambda=1}(x,2) = e^{-\|x-2\|^2}$$

$$K_{\lambda=1}(x,4) = e^{-\|x-4\|^2}$$

$$\phi(x) := \left[1, e^{-\|x-1\|^2}, e^{-\|x-2\|^2}, e^{-\|x-4\|^2}\right]$$

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

$$\hat{y} = \varphi(x)\theta$$

$$= \theta_0 + \theta_1 \exp\left(-(x-1)^2\right) + \theta_2 \exp\left(-(x-2)^2\right) + \theta_3 \exp\left(-(x-4)^2\right)$$

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

(3) Locally Weighted / Kernel Regression

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

Regression → Y = Weighted linear sum of X' s → 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, …

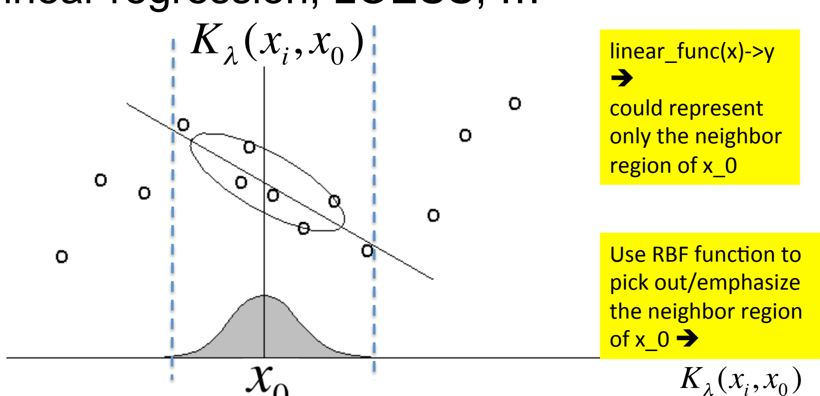
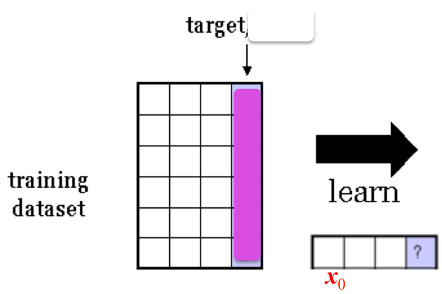


Figure 2: In locally weighted regression, points are weighted by proximity to the current x in question using a kernel. A regression is then computed using the weighted points.

LEARNING of Locally weighted linear regression



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

Task	Regression
↓	↓
Representation	Y = Weighted linear sum of X's
↓	↓
Score Function	Least-squares
↓	↓
Search/Optimization	Linear algebra + Regularization
↓	↓
Models, Parameters	Regression coefficients

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

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

35/54

Error on data + Regularization

(4) 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

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

to minimize $J(\beta)$, 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 classification models
  - Support Vector Machine
  - Bayes Classifier
  - Logistic Regression
  - K-nearest Neighbor
- ❑ Model Selection / Bias Variance Tradeoff

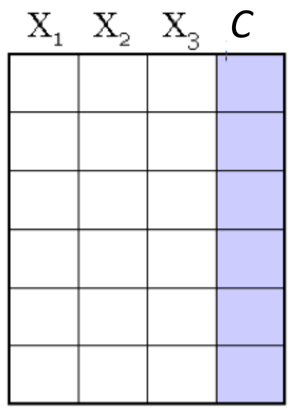
Where are we ? ➔ Three major sections for classification

- We can divide the large variety of classification approaches into roughly three major types

1. Discriminative
   - directly estimate a decision rule/boundary
   - e.g., logistic regression, support vector machine, decisionTree

2. Generative:
   - build a generative statistical model
   - e.g., naïve bayes classifier, Bayesian networks

3. Instance based classifiers
   - Use observation directly (no models)
   - e.g. K nearest neighbors



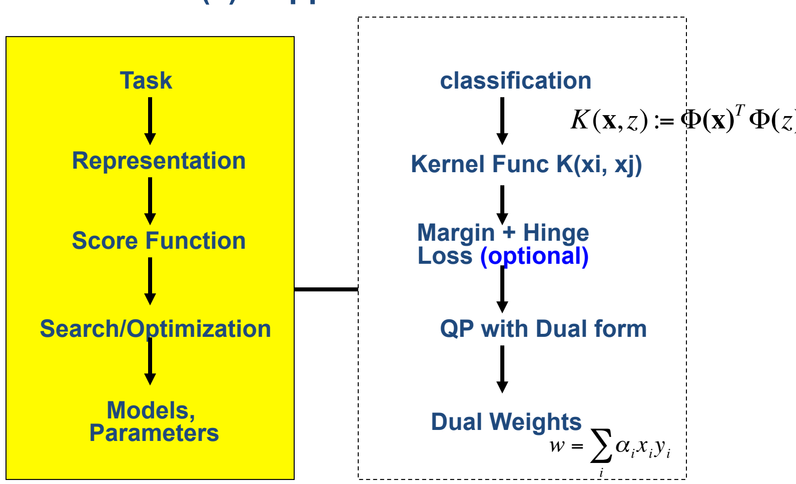
A Dataset for classification

$f : X \longrightarrow C$

Output as Discrete Class Label
$C_1, C_2, \ldots, C_L$

- **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]

(1) Support Vector Machine

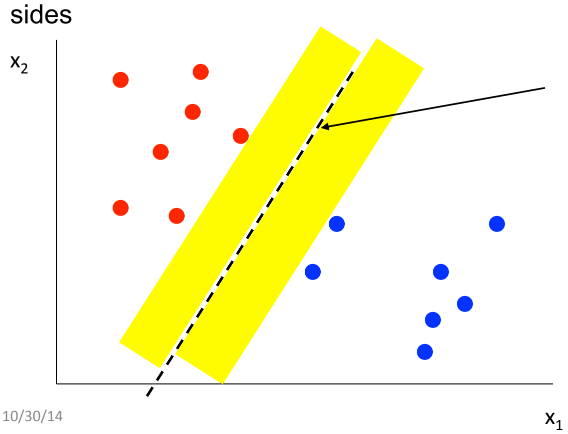


$$\underset{\mathbf{w},b}{\operatorname{argmin}} \sum_{i=1}^{p} w_i^2 + C\sum_{i=1}^{n} \varepsilon_i$$

$$\text{subject to } \forall \mathbf{x}_i \in Dtrain : y_i(\mathbf{x}_i \cdot \mathbf{w} + b) \geq 1 - \varepsilon_i$$

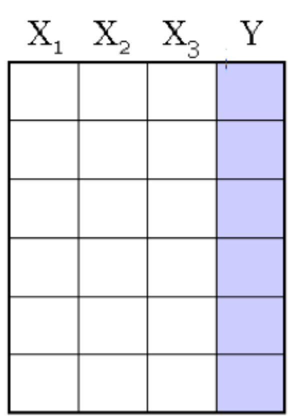
(1) SVM as Max margin classifiers

- Instead of fitting all points, focus on boundary points
- Learn a boundary that leads to the largest margin from points on both sides



Why?

- Intuitive, 'makes sense'
- Some theoretical support
- Works well in practice



A Dataset for binary classification

$$f : X \longrightarrow Y$$

Output as Binary Class Label: 1 or -1

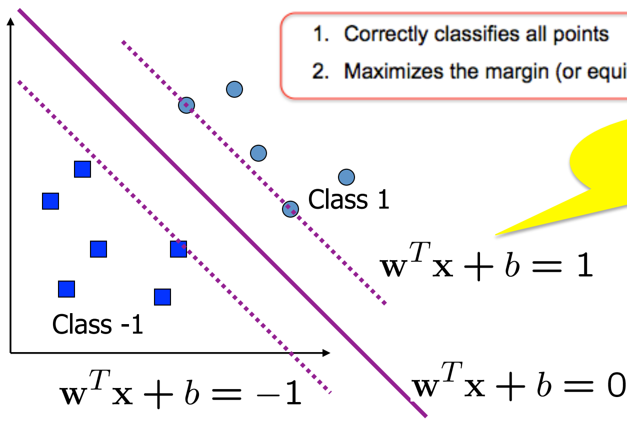
- **Data**/*points/instances/examples/samples/records*: [rows]
- **Features**/*attributes/dimensions/independent variables/covariates/predictors/regressors*: [columns, except the last]
- **Target**/*outcome/response/label/dependent variable*: special column to be predicted [last column]

When linearly Separable Case

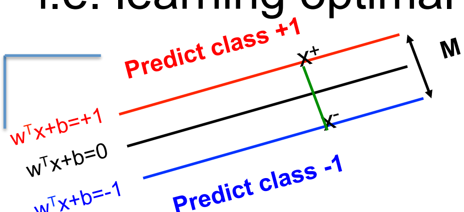
- The decision boundary should be as far away from the data of both classes as possible

1. Correctly classifies all points
2. Maximizes the margin (or equivalently minimizes $w^T w$)

W is a p-dim vector; b is a scalar



Optimization Step i.e. learning optimal parameter for SVM



$M = \frac{2}{\sqrt{w^T w}}$

1. Correctly classifies all points
2. Maximizes the margin (or equivalently minimizes $w^T w$)

Min $(w^T w)/2$

subject to the following constraints:

For all x in class + 1

$w^Tx+b \geq 1$

For all x in class - 1

$w^Tx+b \leq -1$

A total of n constraints if we have n input samples

$$\underset{\mathbf{w},b}{\operatorname{argmin}} \sum_{i=1}^{p} w_i^2$$

$$\text{subject to } \forall \mathbf{x}_i \in Dtrain : y_i(\mathbf{x}_i \cdot \mathbf{w} + b) \geq 1$$

SVM as a QP problem

Dual formulation

Two optimization problems: For the separable and non separable cases

Min $(w^Tw)/2$

For all x in class + 1

$w^Tx+b \geq 1$

For all x in class - 1

$w^Tx+b \leq -1$

$\min_w \frac{w^T w}{2} + \sum_{i=1}^{n} C\varepsilon_i$ → Hinge loss (optional)

For all x_i in class + 1

$w^Tx+b \geq 1- \varepsilon_i$

For all x_i in class - 1

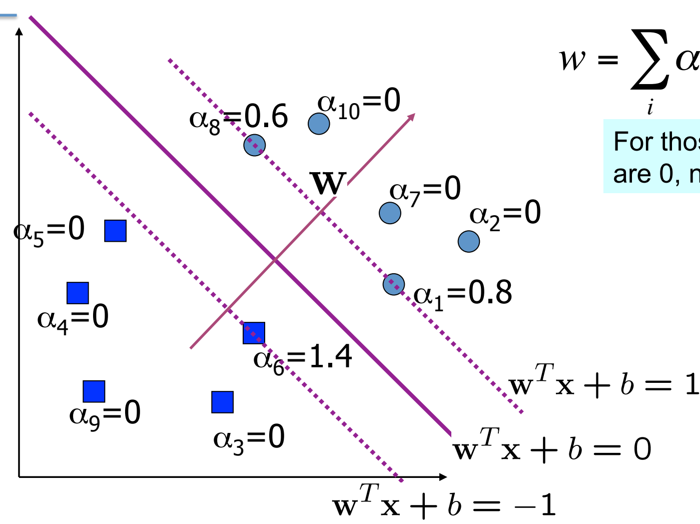
$w^Tx+b \leq -1+ \varepsilon_i$

For all i

$\varepsilon_i \geq 0$

- Instead of solving these QPs directly we will solve a dual formulation of the SVM optimization problem
- The main reason for switching to this type of representation is that it would allow us to use a neat trick that will make our lives easier (and the run time faster)

A Geometrical Interpretation



$$w = \sum_i \alpha_i x_i y_i$$

For those α_i that are 0, no influence

The kernel trick

How many operations do we need for the dot product?

$$K(\mathbf{x},z) := \Phi(\mathbf{x})^T \Phi(z)$$

$$\Phi(x)^T \Phi(z) = \sum_i 2x_i z_i + \sum_i x_i^2 z_i^2 + \sum_i \sum_{j=i+1} 2x_i x_j z_i z_j + 1$$

m   m   m(m-1)/2   **=~ m²**

However, we can obtain dramatic savings by noting that

$$\Phi(x)^T \Phi(z) = (x^T z + 1)^2 = (x.z+1)^2 = (x.z)^2 + 2(x.z) + 1$$

$$= (\sum_i x_i z_i)^2 + \sum_i 2x_i z_i + 1$$

$$= \sum_i 2x_i z_i + \sum_i x_i^2 z_i^2 + \sum_i \sum_{j=i+1} 2x_i x_j z_i z_j + 1$$

We only need m operations!

So, if we define the kernel function as follows, there is no need to carry out ϕ(.) explicitly

$$K(\mathbf{x},z) = (x^T z + 1)^2$$

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 classification models
  - Support Vector Machine
  - Bayes Classifier
  - Logistic Regression
  - K-nearest Neighbor
- ❑ Model Selection

$$\operatorname*{argmax}_k P(C\_k \mid X) = \operatorname*{argmax}_k P(X,C) = \operatorname*{argmax}_k P(X \mid C)P(C)$$

(2) Bayes Classifier

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

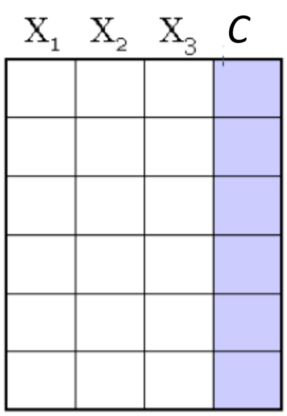
classification → Prob. models p(X|C) → EPE, with Log likelihood(optional) → Many options → Prob. Models' Parameter

$P(X_1, \cdots, X_p \mid C)$

Gaussian Naive: $\hat{P}(X_j \mid C = c_k) = \frac{1}{\sqrt{2\pi}\sigma_{jk}} \exp\left(-\frac{(X_j - \mu_{jk})^2}{2\sigma_{jk}^2}\right)$

Bernoulli Naive: $p(W_i = true \mid c_k) = p_{i,k}$

Multinomial: $P(W_1 = n_1, \ldots, W_v = n_v \mid c_k) = \frac{N!}{n_{1k}! n_{2k}! \ldots n_{vk}!} \theta_{1k}^{n_{1k}} \theta_{2k}^{n_{2k}} \ldots \theta_{vk}^{n_{vk}}$



A Dataset for classification

$f : X \longrightarrow C$

Output as Discrete Class Label $C_1, C_2, \ldots, C_L$

Generative: $\operatorname*{argmax}_C P(C \mid X) = \operatorname*{argmax}_C P(X,C) = \operatorname*{argmax}_C P(X \mid C)P(C)$

- **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]

(2) Bayes classifier

- Treat each attribute and class label as random variables.
- Given a sample **x** with attributes ($x_1, x_2, \ldots, x_p$):
  - Goal is to predict class *C.*
  - Specifically, we want to find the value of C_i that maximizes $p(C_i \mid x_1, x_2, \ldots, x_p)$.
- Bayes classification

$$P(C \mid \mathbf{X}) \propto P(\mathbf{X} \mid C)P(C) = P(X_1, \cdots, X_p \mid C)P(C)$$

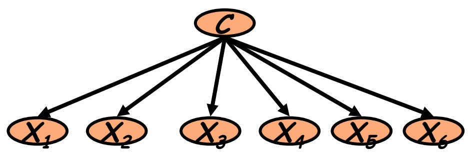
Difficulty: learning the joint probability $P(X_1, \cdots, X_p \mid C)$

(2.1) Naïve Bayes Classifier

Difficulty: learning the joint probability $P(X_1, \cdots, X_p \mid C)$

- Naïve Bayes classification
  - Assumption that all input attributes are conditionally independent!

$$\begin{aligned} P(X_1, X_2, \cdots, X_p \mid C) &= P(X_1 \mid X_2, \cdots, X_p, C)P(X_2, \cdots, X_p \mid C) \\ &= P(X_1 \mid C)P(X_2, \cdots, X_p \mid C) \\ &= P(X_1 \mid C)P(X_2 \mid C) \cdots P(X_p \mid C) \end{aligned}$$



Adapt from Prof. Ke Chen NB slides

(2.2) Multinomial Naïve Bayes as Stochastic Language Models

Model C1	
0.2	the
0.01	boy
0.0001	said
0.0001	likes
0.0001	black
0.0005	dog
0.01	garden

Model C2	
0.2	the
0.0001	boy
0.03	said
0.02	likes
0.1	black
0.01	dog
0.0001	garden

the	boy	likes	black	dog
0.2	0.01	0.0001	0.0001	0.0005
0.2	0.0001	0.02	0.1	0.01

P(s|C2) P(C2) > P(s|C1) P(C1)

(2.3) Gaussian Naïve Bayes Classifier

- Continuous-valued Input Attributes
  - Conditional probability modeled with the normal distribution

$$\hat{P}(X_j \mid C = c_i) = \frac{1}{\sqrt{2\pi}\sigma_{ji}} \exp\left(-\frac{(X_j - \mu_{ji})^2}{2\sigma_{ji}^2} \right)$$

μ_{ji} : mean (avearage) of attribute values X_j of examples for which $C = c_i$

σ_{ji} : standard deviation of attribute values X_j of examples for which $C = c_i$

  - Learning Phase: for $\mathbf{X} = (X_1, \cdots, X_p)$, $C = c_1, \cdots, c_L$
    Output: $p \times L$ normal distributions and $P(C = c_i)$ $i = 1, \cdots, L$
  - Test Phase: for $\mathbf{X}' = (X'_1, \cdots, X'_p)$
    - Calculate conditional probabilities with all the normal distributions
    - Apply the MAP rule to make a decision

Naïve Gaussian means ?

Not Naïve

$$P(X_1, X_2, \cdots, X_p \mid C) =$$

$$\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{D/2}} \frac{1}{|\boldsymbol{\Sigma}|^{1/2}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right\}$$

Naïve

$$P(X_1, X_2, \cdots, X_p \mid C = c_k) = P(X_1 \mid c_k) P(X_2 \mid c_k) \cdots P(X_p \mid c_k)$$

$$= \prod_j \frac{1}{\sqrt{2\pi}\sigma_{j,k}} \exp\left(-\frac{(X_j - \mu_{j,k})^2}{2\sigma_{j,k}^2}\right)$$

Diagonal Matrix $\Sigma\_c_k = \Lambda\_c_k$

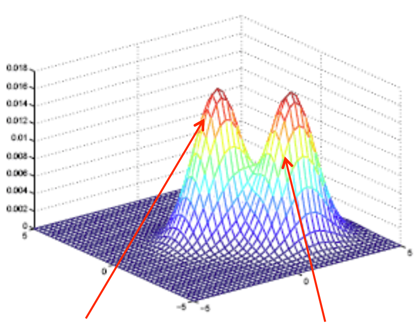
Each class' covariance matrix is diagonal

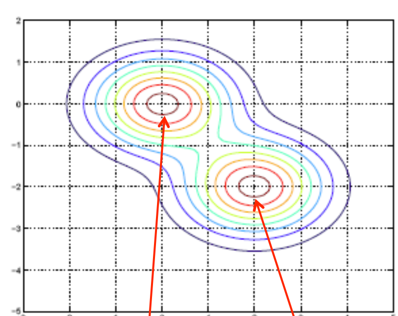
(2.4) LDA (Linear Discriminant Analysis)

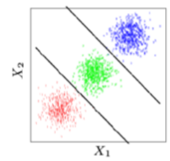
Linear Discriminant Analysis : $\Sigma_k = \Sigma, \ \forall k$

Each class' covariance matrix is the same

The Gaussian Distribution are shifted versions of each other







Optimal Classification

$$\underset{k}{\operatorname{argmax}} P(C\_k \mid X) = \underset{k}{\operatorname{argmax}} P(X, C) = \underset{k}{\operatorname{argmax}} P(X \mid C) P(C)$$

$$= \arg\max_k \Big[-\log((2\pi)^{p/2}|\Sigma|^{1/2}) - \frac{1}{2}(x-\mu_k)^T\Sigma^{-1}(x-\mu_k) + \log(\pi_k) \Big]$$

$$= \arg\max_k \Big[-\frac{1}{2}(x-\mu_k)^T\Sigma^{-1}(x-\mu_k) + \log(\pi_k) \Big]$$

Linear Discriminant Function for LDA

- Note

$$-\frac{1}{2}(x-\mu_k)^T\Sigma^{-1}(x-\mu_k) = x^T\Sigma^{-1}\mu_k - \frac{1}{2}\mu_k^T\Sigma^{-1}\mu_k - \frac{1}{2}x^T\Sigma^{-1}x$$

Define Linear Discriminant Function

$$\delta(x) = -\frac{1}{2}(x-\mu_k)^T \sum^{-1} (x-\mu_k) + log\pi_k$$

➔ The Decision Boundary Between class k and l, $\{x : \delta_k(x) = \delta_l(x)\}$, is a linear line/plane

$$\log\frac{P(C\_k \mid X)}{P(C\_l \mid X)} = \log\frac{P(X \mid C\_k)}{P(X \mid C\_l)} + \log\frac{P(C\_k)}{P(C\_l)}$$

$$= \log\frac{\pi_k}{\pi_\ell} - \frac{1}{2}(\mu_k+\mu_\ell)^T\Sigma^{-1}(\mu_k-\mu_\ell) + x^T\Sigma^{-1}(\mu_k-\mu_\ell) \quad (4.9)$$

Equals to zero

Boundary points X : when P(c_k|X) == P(c_l|X), the left linear equation ==0, a linear line / plane

(2.5) QDA (Quadratic Discriminant Analysis)

- Estimate the covariance matrix Σ_k separately for each class k, $k = 1, 2, ..., K$.
- *Quadratic discriminant function*:

$$\delta_k(x) = -\frac{1}{2}\log|\Sigma_k| - \frac{1}{2}(x-\mu_k)^T\Sigma_k^{-1}(x-\mu_k) + \log\pi_k .$$

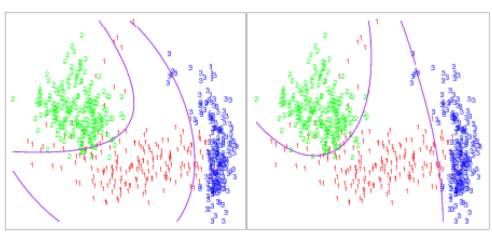
- Classification rule:

$$\hat{G}(x) = \arg\max_k \delta_k(x) .$$

- Decision boundaries are quadratic equations in x.
- QDA fits the data better than LDA, but has more parameters to estimate.

(2.6) LDA on Expanded Basis

- Expand input space to include X_1X_2, X_1^2, and X_2^2.
- Input is five dimensional: $X = (X_1, X_2, X_1X_2, X_1^2, X_2^2)$.



LDA with quadratic basis Versus QDA

Figure 4.6: *Two methods for fitting quadratic boundaries. The left plot shows the quadratic decision boundaries for the data in Figure 4.1 (obtained using LDA in the five-dimensional space $x_1, x_2, x_{12}, x_1^2, x_2^2$). The right plot shows the quadratic decision boundaries found by QDA. The differences are small, as is usually the case.*

(2.7) Regularized Discriminant Analysis

- A compromise between LDA and QDA.
- Shrink the separate covariances of QDA toward a common covariance as in LDA.
- Regularized covariance matrices:

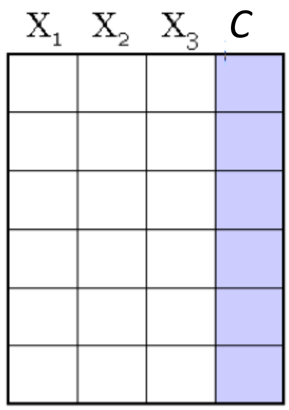
$$\hat{\Sigma}_k(\alpha) = \alpha\hat{\Sigma}_k + (1-\alpha)\hat{\Sigma} .$$

- The quadratic discriminant function $\delta_k(x)$ is defined using the shrunken covariance matrices $\hat{\Sigma}_k(\alpha)$.
- The parameter α controls the complexity of the model.

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 classification models
  - Support Vector Machine
  - Bayes Classifier
  - Logistic Regression
  - K-nearest Neighbor
- ❑ Model Selection

A Dataset for classification



$$f : X \longrightarrow C$$

Output as Discrete Class Label $C_1, C_2, \ldots, C_L$

Discriminative $P(C \mid \mathbf{X}) \quad C = c_1, \cdots, c_L$

- **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]

(3) Logistic Regression

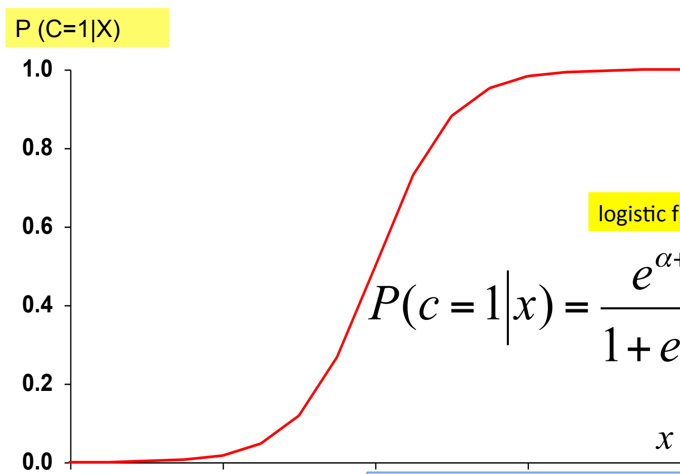
Task	classification
Representation	Log-odds(Y) = linear function of X's
Score Function	EPE, with conditional Log-likelihood
Search/Optimization	Iterative (Newton) method
Models, Parameters	Logistic weights

$$P(c = 1 \mid x) = \frac{e^{\alpha + \beta x}}{1 + e^{\alpha + \beta x}}$$

Discriminative

Logistic regression models for binary target variable coded 0/1.

e.g. Probability of disease



logistic function

$$P(c=1|x) = \frac{e^{\alpha+\beta x}}{1+e^{\alpha+\beta x}}$$

Logit function

Decision Boundary ➔ equals to zero

$$\ln\left[\frac{P(c=1|x)}{P(c=0|x)}\right] = \ln\left[\frac{P(c=1|x)}{1-P(c=1|x)}\right] = \alpha + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_p x_p$$

MLE for Logistic Regression Training

Let's fit the logistic regression model for K=2, i.e., number of classes is 2

Training set: (x_i, y_i), i=1,…,N

For Bernoulli distribution

$$p(y|x)^y(1-p)^{1-y}$$

(conditional) Log-likelihood:

$$l(\beta) = \sum_{i=1}^{N}\{\log \Pr(Y=y_i|X=x_i)\}$$

$$= \sum_{i=1}^{N} y_i \log(\Pr(Y=1|X=x_i)) + (1-y_i)\log(\Pr(Y=0|X=x_i))$$

$$= \sum_{i=1}^{N}(y_i \log \frac{\exp(\beta^T x_i)}{1+\exp(\beta^T x_i)}) + (1-y_i)\log\frac{1}{1+\exp(\beta^T x_i)})$$

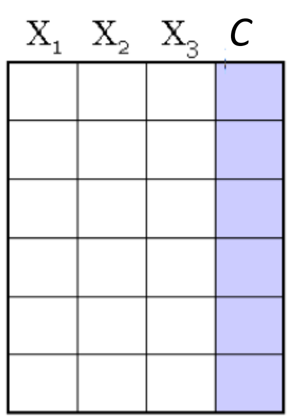
$$= \sum_{i=1}^{N}(y_i \beta^T x_i - \log(1+\exp(\beta^T x_i)))$$

x_i are $(p+1)$-dimensional input vector with leading entry 1
β is a $(p+1)$-dimensional vector
y_i = 1 if C_i =1; y_i = 0 if C_i =0

 We want to maximize the log-likelihood in order to estimate β

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 classification models
  - Support Vector Machine
  - Bayes Classifier
  - Logistic Regression
  - K-nearest Neighbor
- ❑ Model Selection



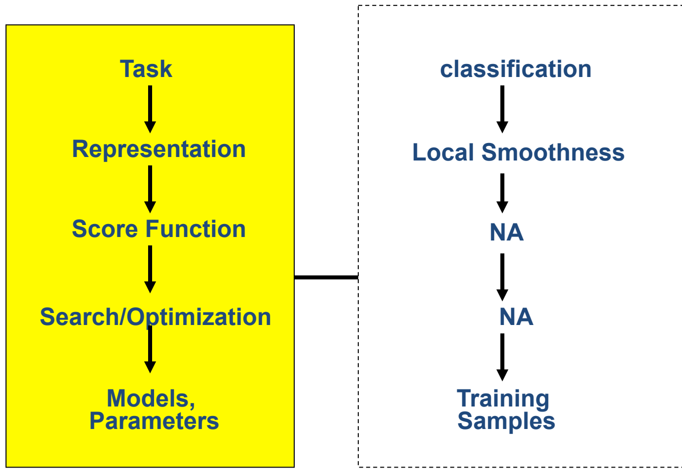
A Dataset for classification

$$f : X \longrightarrow C$$

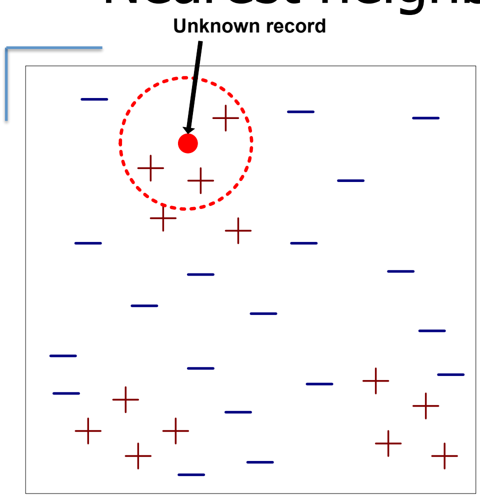
Output as Discrete Class Label $C_1, C_2, \ldots, C_L$

- **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]

(4) K-Nearest Neighbor



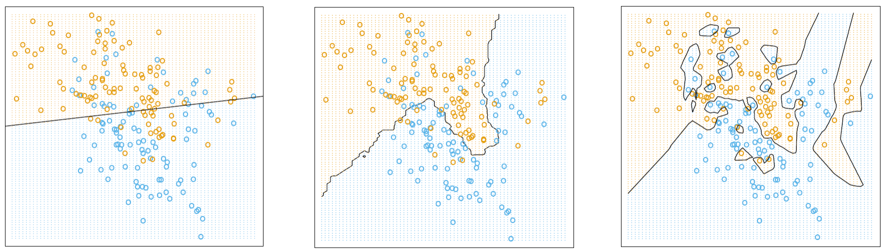
Nearest neighbor classifiers



To classify unknown sample:

1. Compute distance to other training records
2. Identify *k* nearest neighbors
3. Use class labels of nearest neighbors to determine the class label of unknown record (e.g., by taking majority vote)

Decision boundaries in global vs. local models



linear regression

- global
- stable
- can be inaccurate

15-nearest neighbor

1-nearest neighbor

- local
- accurate
- unstable

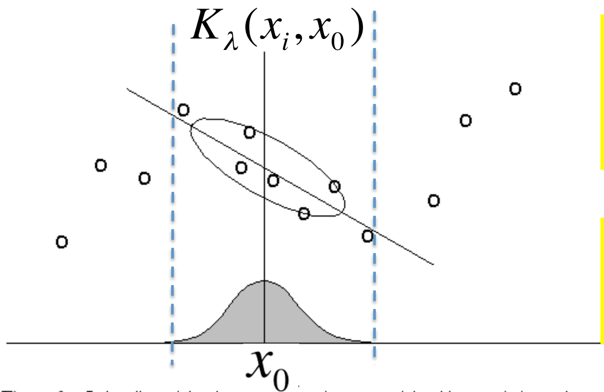
What ultimately matters: ***GENERALIZATION***

Nearest neighbor classification

- *k*-Nearest neighbor classifier is a lazy learner
  - Does not build model explicitly.
  - Unlike eager learners such as decision tree induction and rule-based systems.
  - Classifying unknown samples is relatively expensive.
- *k*-Nearest neighbor classifier is a local model, vs. global model of linear classifiers.

Vs. Locally weighted regression

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



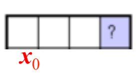
linear_func(x)->y ➔ could represent only the neighbor region of x_0

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.

Vs. Locally weighted regression

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



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

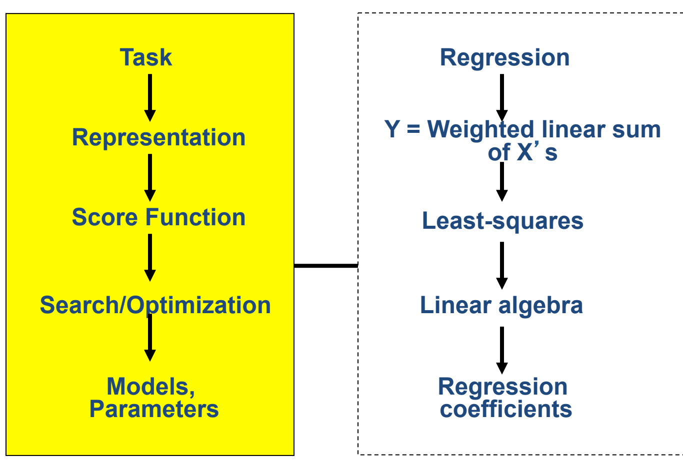
$$\hat{f}(x_0) = \hat{\alpha}(x_0) + \hat{\beta}(x_0)x_0$$

$$K_\tau(\mathbf{x}_i, \mathbf{x}0) = \exp\left(-\frac{(\mathbf{x}_i - \mathbf{x}0)^2}{2\tau^2}\right)$$

Today

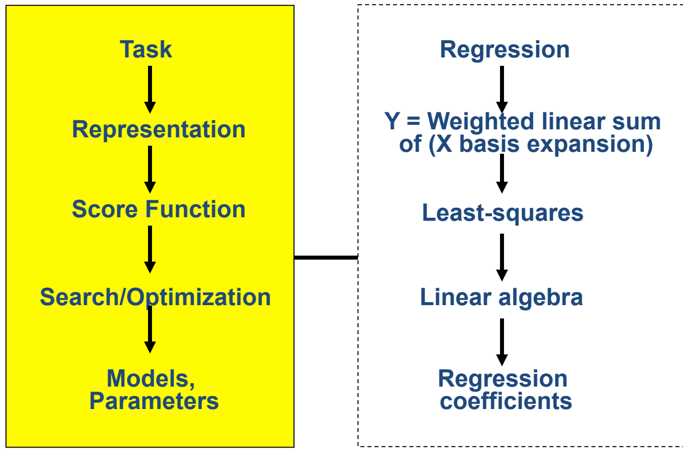
- ❑ Review of basic pipeline
- ❑ Review of regression models
  - Linear regression (LR)
  - LR with non-linear basis functions
  - Locally weighted LR
  - LR with Regularizations
- ❑ Review of classification models
  - Support Vector Machine
  - Bayes Classifier
  - Logistic Regression
  - K-nearest Neighbor
- ❑ Model Selection / Bias Variance Tradeoff

(1) Multivariate Linear Regression



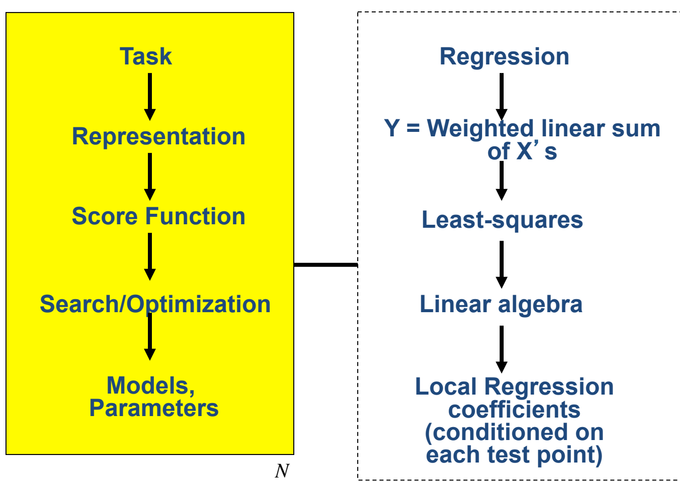
$$\hat{y} = f(x) = \theta_0 + \theta_1 x^1 + \theta_2 x^2$$

(2) Multivariate Linear Regression with basis Expansion



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

(3) Locally Weighted / Kernel Regression



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

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

(4) Regularized multivariate linear regression

Task	Regression
↓	↓
Representation	Y = Weighted linear sum of X's
↓	↓
Score Function	Least-squares
↓	↓
Search/Optimization	Linear algebra + Regularization
↓	↓
Models, Parameters	Regression coefficients

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

(1) Support Vector Machine

Task	classification
↓	↓
Representation	Kernel Func K(xi, xj)
↓	↓
Score Function	Margin + Hinge Loss (optional)
↓	↓
Search/Optimization	QP with Dual form
↓	↓
Models, Parameters	Dual Weights

$$K(\mathbf{x}, z) := \Phi(\mathbf{x})^T \Phi(z)$$

$$w = \sum_i \alpha_i x_i y_i$$

$$\underset{\mathbf{w},b}{\operatorname{argmin}} \sum_{i=1}^{p} w_i^2 + C\sum_{i=1}^{n} \varepsilon_i$$

subject to $\forall \mathbf{x}_i \in Dtrain : y_i\left(\mathbf{x}_i \cdot \mathbf{w} + b\right) \geq 1 - \varepsilon_i$

$$\underset{k}{\text{argmax}}\, P(C\_k \mid X) = \underset{k}{\text{argmax}}\, P(X,C) = \underset{k}{\text{argmax}}\, P(X \mid C)P(C)$$

Yanjun Qi / UVA CS 4501-01-6501-07

(2) Bayes Classifier

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

classification → Prob. models p(X|C) → EPE, with Log likelihood(optional) → Many options → Prob. Models' Parameter

$$P(X_1, \cdots, X_p \mid C)$$

Gaussian Naive

$$\hat{P}(X_j \mid C = c_k) = \frac{1}{\sqrt{2\pi}\sigma_{jk}} \exp\left(-\frac{(X_j - \mu_{jk})^2}{2\sigma_{jk}^2} \right)$$

Bernoulli Naive

$$p(W_i = true \mid c_k) = p_{i,k}$$

Multinomial

$$P(W_1 = n_1, \ldots, W_v = n_v \mid c_k) = \frac{N!}{n_{1k}! n_{2k}! \ldots n_{vk}!} \theta_{1k}^{n_{1k}} \theta_{2k}^{n_{2k}} \ldots \theta_{vk}^{n_{vk}}$$

10/30/14 81

Yanjun Qi / UVA CS 4501-01-6501-07

(3) Logistic Regression

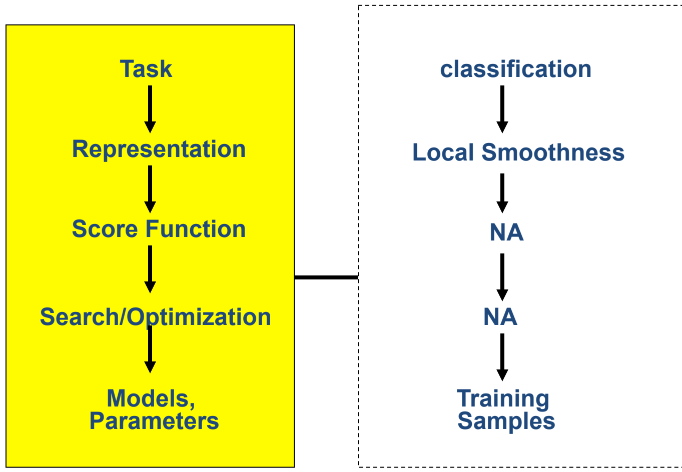
Task → Representation → Score Function → Search/Optimization → Models, Parameters

classification → Log-odds(Y) = linear function of X's → EPE, with conditional Log-likelihood → Iterative (Newton) method → Logistic weights

$$P(c = 1 | x) = \frac{e^{\alpha + \beta x}}{1 + e^{\alpha + \beta x}}$$

10/30/14 82

(4) K-Nearest Neighbor



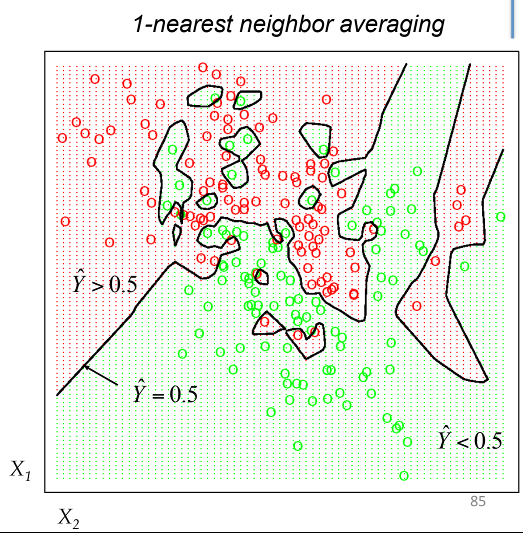
Today

- ❑ 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 classification models
  - Support Vector Machine
  - Bayes Classifier
  - Logistic Regression
  - K-nearest Neighbor
- ❑ Model Selection / Bias Variance Tradeoff

e.g. Training Error from KNN, Lesson Learned

- When k = 1,
- No misclassifications (on training): Overtraining

- Minimizing training error is not always good (e.g., 1-NN)



Statistical Decision Theory

- Random input vector: X
- Random output variable: Y
- Joint distribution: $\Pr(X, Y)$
- Loss function $L(Y, f(X))$

- Expected prediction error (EPE):
- $$EPE(f) = E(L(Y, f(X))) = \int L(y, f(x)) \Pr(dx, dy)$$
  $$\text{e.g.} = \int (y - f(x))^2 \Pr(dx, dy)$$

Consider population distribution

e.g. Squared error loss (also called L2 loss)

Expected prediction error (EPE)

$$\mathrm{EPE}(f) = \mathrm{E}(L(Y, f(X))) = \int L(y, f(x)) \Pr(dx, dy)$$

Consider sample population distribution

- For L2 loss: e.g. $= \int (y - f(x))^2 \Pr(dx, dy)$

under L2 loss, best estimator for EPE (Theoretically) is :

Conditional mean $f(x) = \mathrm{E}(Y \mid X = x)$

e.g. KNN

NN methods are the direct implementation (approximation)

- For 0-1 loss: $L(k, \ell) = 1 - \delta_{kl}$

$\hat{G}(X) = C_k$ if

$\Pr(C_k \mid X = x) = \max_{g \in C} \Pr(g \mid X = x)$

Bayes Classifier

Decomposition of EPE

$Y = f(X) + \epsilon, \ \epsilon \sim (0, \sigma^2)$

– When additive error model:

– Notations

- Output random variable: Y
- Prediction function: f
- Prediction estimator: $\hat{f}$

$$\begin{aligned} EPE(x_0) &= E[(Y - \hat{f})^2 | X = x_0] \\ &= E[((Y - f) + (f - \hat{f}))^2 | X = x_0] \\ &= E[\underbrace{(Y - f)}_{\epsilon}{}^2 | X = x_0] + \underbrace{E[(f - \hat{f})^2 | X = x_0]}_{MSE} \\ &= \sigma^2 + Var(\hat{f}) + \mathrm{Bias}^2(\hat{f}) \end{aligned}$$

MSE component of f-hat in estimating f

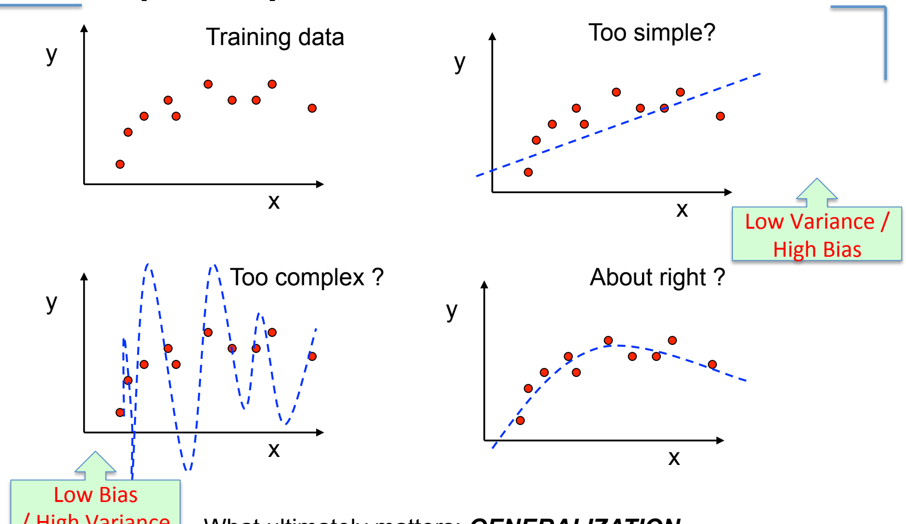
Irreducible / Bayes error

BIAS AND VARIANCE TRADE-OFF

- θ : true value (normally unknown)
- $\hat{\theta}$: estimator
- $\bar{\theta} := E[\hat{\theta}]$ (mean, i.e. expectation of the estimator)

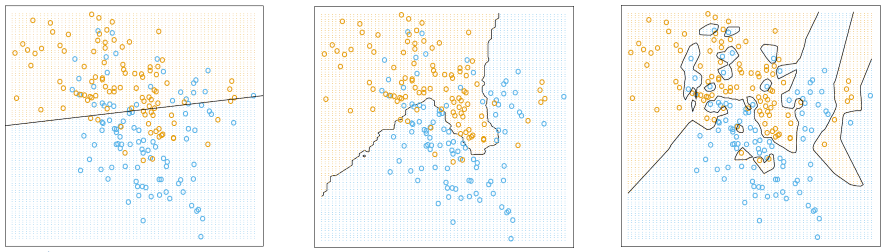
- Bias $E[(\bar{\theta} - \theta)^2]$
  - measures accuracy or quality of the estimator
  - low bias implies on average we will accurately estimate true parameter or func from training data
- Variance $E[(\hat{\theta} - \bar{\theta})^2]$
  - Measures precision or specificity of the estimator
  - Low variance implies the estimator does not change much as the training set varies

Regression: Complexity versus Goodness of Fit



What ultimately matters: ***GENERALIZATION***

Classification,
Decision boundaries in global vs. local models



Low Variance / High Bias

linear regression
- global
- stable
- can be inaccurate

15-nearest neighbor

Low Variance / High Bias

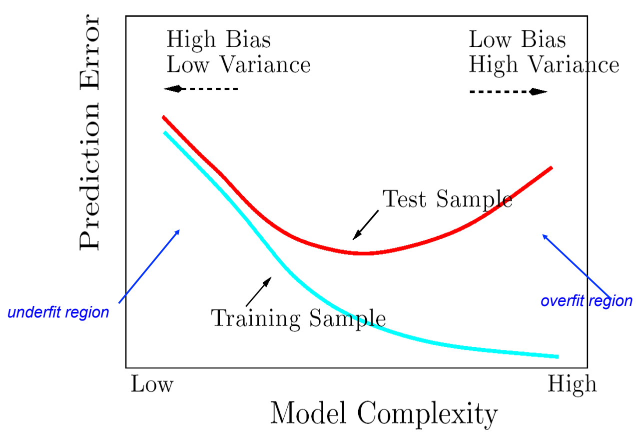
KNN
- local
- accurate
- unstable

1-nearest neighbor

Low Bias / High Variance

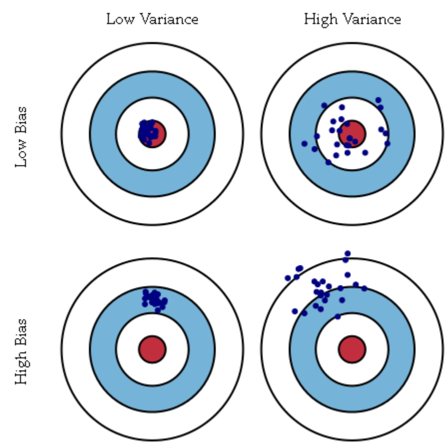
What ultimately matters: ***GENERALIZATION***

Bias-Variance Tradeoff / Model Selection



Model "bias" & Model "variance"

- Middle RED:
  - TRUE function
- Error due to bias:
  - How far off in general from the middle red
- Error due to variance:
  - How wildly the blue points spread



References

- ❑ Prof. Tan, Steinbach, Kumar's "Introduction to Data Mining" slide
- ❑ Prof. Andrew Moore's slides
- ❑ Prof. Eric Xing's slides
- ❑ Hastie, Trevor, et al. *The elements of statistical learning*. Vol. 2. No. 1. New York: Springer, 2009.

Midterm

- Open Note / Open Book
- No laptop / No Cell phone / No internet access

- Easier than sample questions in HW4