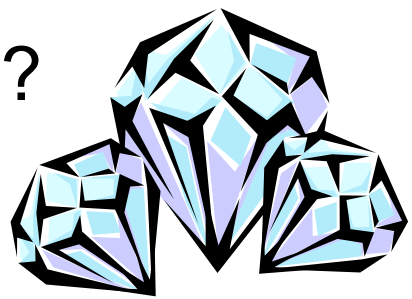
ECE7252
Statistical Learning for Signal Processing

Lecture 25: Summary

Chin-Hui Lee
School of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, GA 30332, USA
chl@ece.gatech.edu

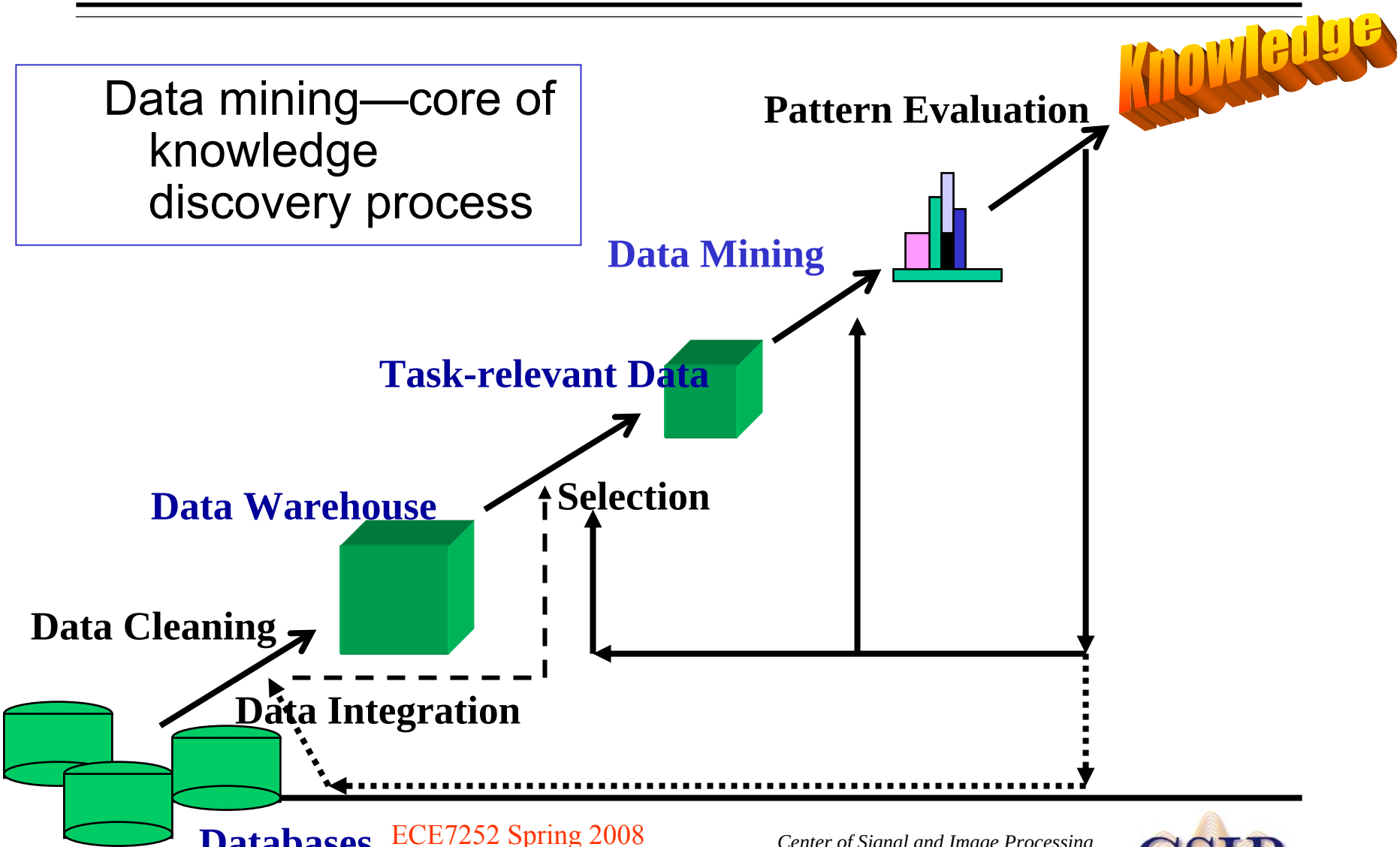
Data Mining and Machine Learning

- Data mining (knowledge discovery from data)
  - Extraction of interesting (non-trivial, implicit, previously unknown and potentially useful) patterns or knowledge from huge amount of data
- Alternative names
  - Knowledge discovery (mining) in databases (KDD), knowledge extraction, data/pattern analysis, data archeology, data dredging, information harvesting, business intelligence, etc.
- Watch out: Is everything "data mining"?
  - Simple search and query processing
  - (Deductive) expert systems

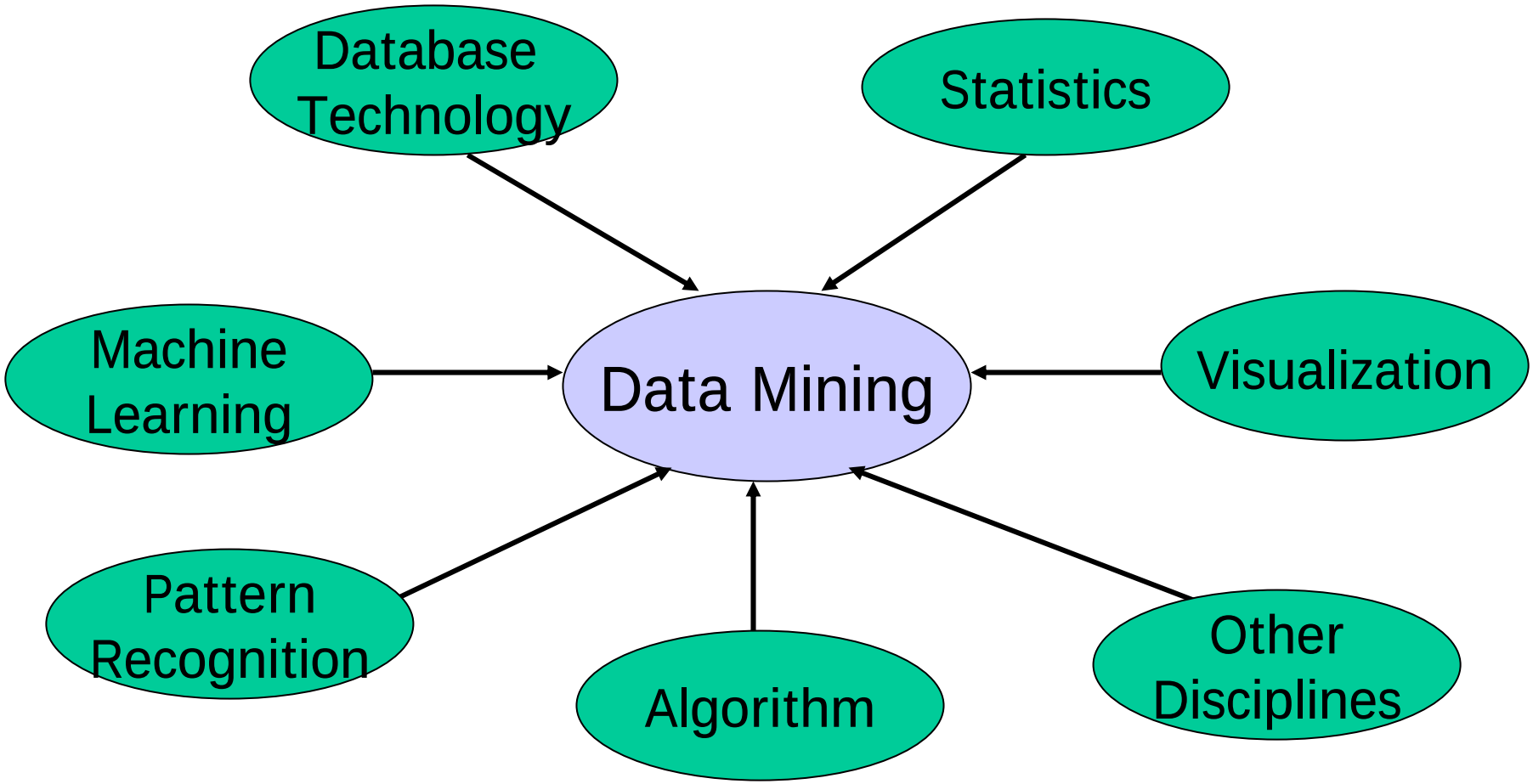


Knowledge Discovery (KDD) Process

Data mining—core of knowledge discovery process



Data Mining: Confluence of Multiple Disciplines



Regression

- Prediction of quantitative output given one or more inputs: given examples $\{(\boldsymbol{x}, y), \ldots\}$, predict $y=f(\boldsymbol{x})$, $\boldsymbol{x}$ can be a vector
- Sample algorithms:
  - Linear and nonlinear regression
  - Artificial neural networks
  - Support vector machines
  - Auto-regression of time series (e.g. speech)
  - Decision trees

Classification

- Prediction of qualitative output given one or more inputs: given samples *{(**x**,C), …}*, predict *C=f(**x**)*, ***x*** can be a vector, and *C* is a class label": email spam: "is this a junk mail?"
- Sample algorithms:
  - Support vector machines (SVM)
  - Artificial neural networks (ANN)
  - Bayesian networks (BN)
  - K-nearest neighbor (KNN)
  - Linear discriminant function (LDF)
  - Hidden Markov model (HMM)
  - Decision trees (DT)

Prediction by Learning from Data

Assume that we have a data set

x_{11}	x_{21}	.	.	.	x_{p1}	y_1
x_{12}	x_{22}	.	.	.	x_{p2}	y_2
.	.				.	.
.	.				.	.
.	.				.	.
x_{1n}	x_{2n}	.	.	.	x_{pn}	y_n

which shows the outcome (response) y **for a set of investigated objects with features** $x_1, \ldots, x_p$

Prediction by learning from data implies that

$$\hat{y} = \hat{f}(x_1, x_2, \ldots, x_p)$$

that can be used to foresee the outcome for new objects (with known or observed features)

Supervised & Unsupervised Learning

- Supervised learning
  - A teacher provides a category label or cost for each pattern in the training set (i.e., ground truth based on experts' knowledge)
  - Can we trust such ground truth? Inconsistency?
- Unsupervised learning
  - The system forms clusters or "natural groupings" of the input patterns
- Semi-supervised learning
  - Use both labeled and un-labeled patterns to reduce the labeling cost

Algorithms for Classification and Regression

- Linear perceptron, least squares
- LDA/FDA (linear/Fisher discriminant analysis): simple linear cuts, kernel non-linear generalizations
- SVM (support vector machine): optimal, wide margin linear cuts, kernel non-linear generalizations
- Decision trees: logical rules
- KNN (*k*-nearest N\neighbors): simple non-parametric
- Artificial neural networks (ANN): general non-linear models, adaptation ability, “artificial brain”

Three Phases in Supervised Learning

- Collecting labeled training data
  - Examples with label assignment
  - Labels depend on application needs
- Learning classification models
  - Appropriate model and representation of the problem need to be selected in terms of attributes, distance (similarity) measure and classifier type
  - Adaptive parameters in the model need to optimized to provide correct classification of training examples (e.g. minimizing the number of misclassified training vectors)
- Validation
  - Cross-validation, independent control sets and other measure of "real" accuracy and generalization should be used to assess the success of the model *(finding trade off between accuracy and generalization is not trivial)*

Similarity Measures

Consider a set of N objects represented as vectors in a certain p-dimansional feature space (e.g. measurments on p attributes, such as expression levels for p genes) $\boldsymbol{x}_i=[x_{i1}, x_{i2}, \ldots, x_{im}, \ldots, x_{ip}]^T$. For quantitative atributes (variables), a commonly used measure of dissimilarity is the *squared distance*:

$$d_2(\boldsymbol{x}_i, \boldsymbol{x}_j)=\sum_m (x_{im}-x_{jm})^2 \quad .$$

In case of gene expression data, the correlation coefficient is often used to measure similarity between expression profiles across genes or samples:

$$cc(\boldsymbol{x}_i, \boldsymbol{x}_j)=\frac{\sum_m (x_{im}-\langle x_i\rangle)(x_{jm}-\langle x_j\rangle)}{\sqrt{\sum_m (x_{im}-\langle x_i\rangle)^2 \sum_m (x_{jm}-\langle x_j\rangle)^2}} \quad ,$$

where $\langle x_i\rangle=\sum_m x_{im}/p$.

Performance Analysis

- Need mechanisms and metrics to compare the effectiveness of various algorithms
- False Rejection: *the fraction of false positives*
- False Alarm: *the fraction of false negatives*
- Recall: *the fraction of relevant cases retrieved*
- Precision: *the fraction of retrieved cases that were relevant*
- Error Rate: *the fraction of wrong classification*
- F-Value*: combination of recall and precision*
- Usually, increased performance in one metric results in decreased performance in the other

Learning and Data Selection

- Divide the data into two sets
  - Training: use the training data to fit all potential models
  - Test (Evaluation): use the test data to evaluate and validate the trained models
- Data-rich environment (even better): divide the data into three sets
  - Training: use the training data to fit all potential models
  - Validation: use the validation data to select model
  - Test: use the test data to obtain an unbiased estimate of the selected models

Model Selection & Generalization

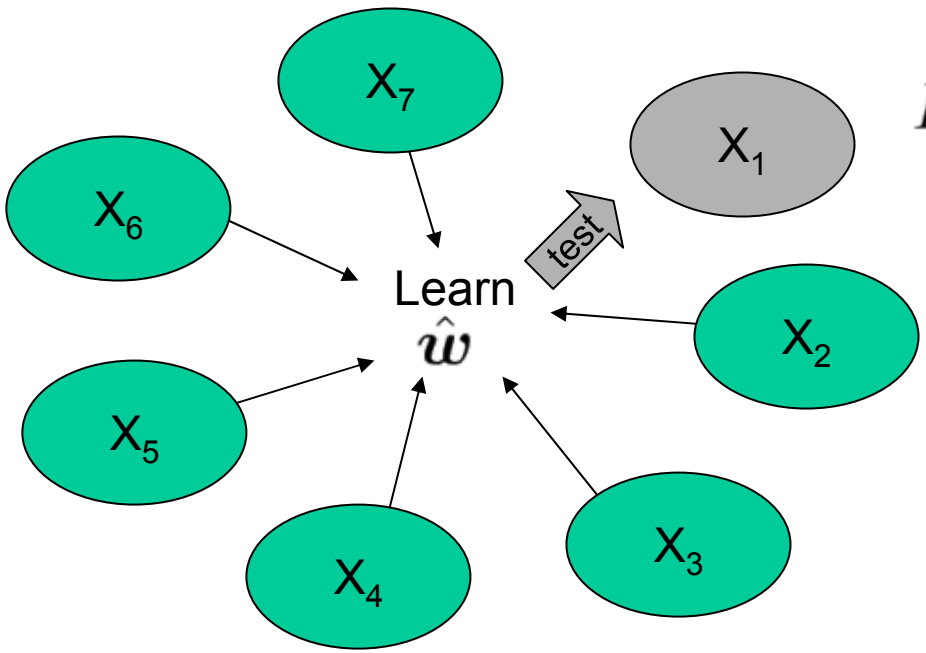
- Learning is an ill-posed problem; data is not sufficient to find a unique solution
- The need for inductive bias: assumptions about the hypothesized model, $\mathcal{H}$, from true C
- Generalization: How well a model performs on new data?
- Overfitting: $\mathcal{H}$ more complex than true C or f
- Underfitting: $\mathcal{H}$ less complex than true C or f

Cross-Validation

- To estimate generalization error, we need data unseen during training. We split the data as
  - Training set (50%)
  - Validation set (25%)
  - Test (publication) set (25%)
- Resampling when there is few data

K-fold Cross Validation

- A technique for estimating test error
- Uses *all* of the data to validate
- Divide data into K groups $\{X_1, X_2, \ldots, X_K\}$.
- Use each group as a validation set, then average all validation errors



$$L_1 = \sum_{(\boldsymbol{x},y)\in X_1} (y - \hat{\boldsymbol{w}}^\top \boldsymbol{x})$$

$$CV(s) = \frac{1}{K}\sum_{i=1}^{K} L_i$$

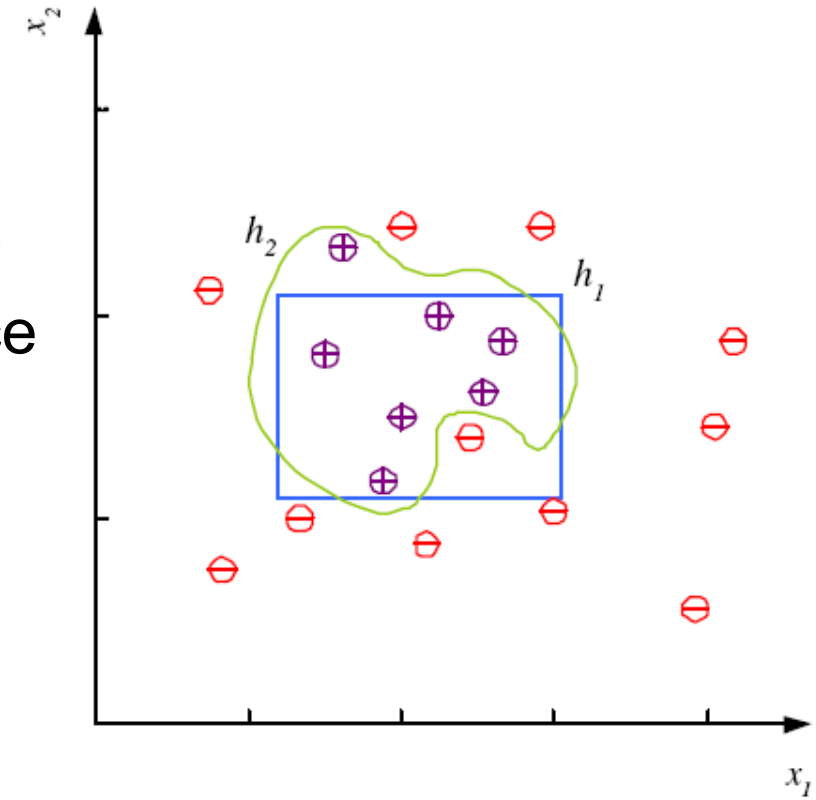
Triple Trade-Off

- There is a trade-off between three factors (Dietterich, 2003):
  - Complexity of $\mathcal{H}$: $c(\mathcal{H})$,
  - Training set size: N,
  - Generalization error, E, on new data
- As $N\uparrow$, $E\downarrow$
- As $c(\mathcal{H})\uparrow$, first $E\downarrow$ and then $E\uparrow$

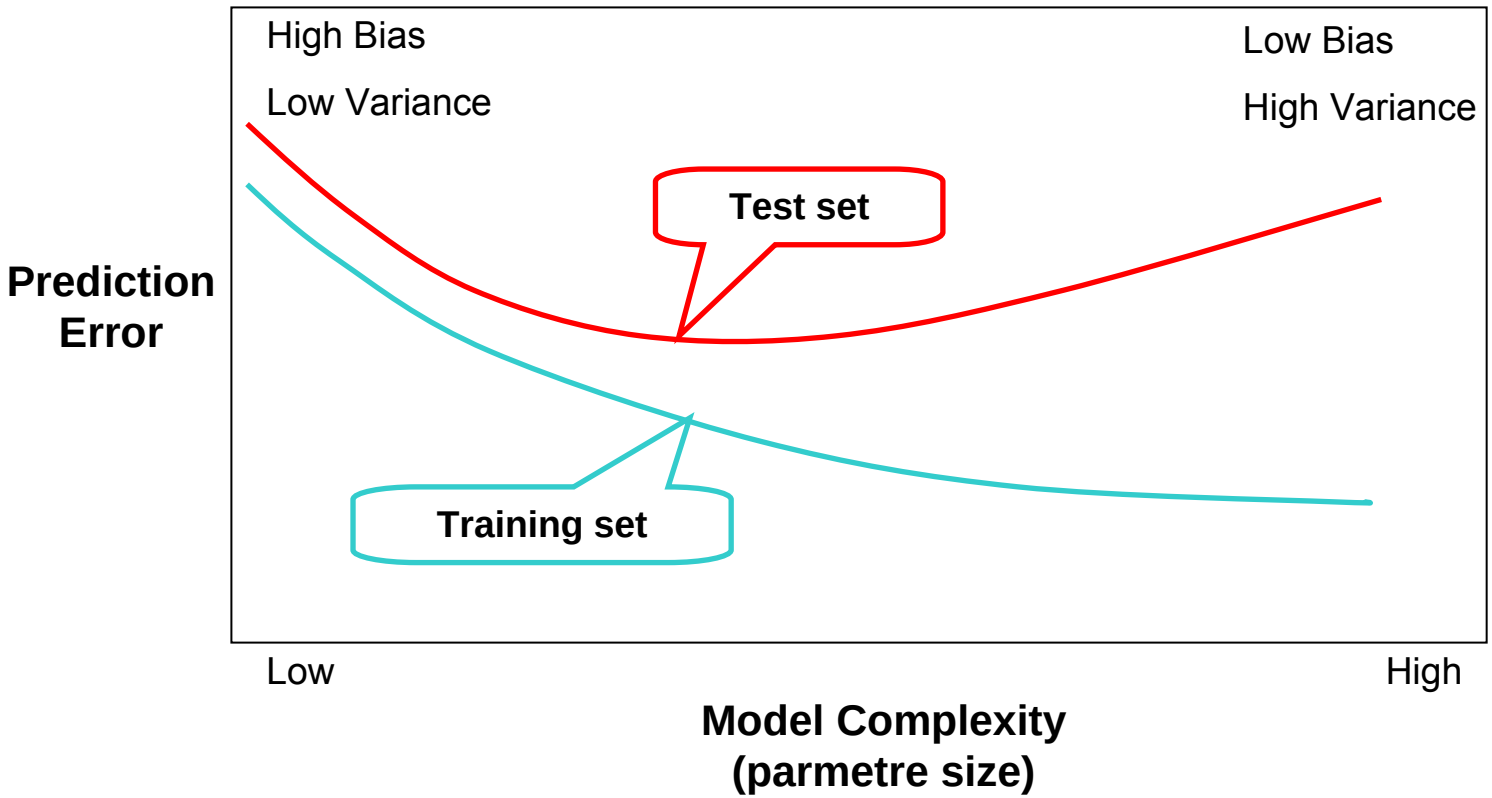
Noise and Model Complexity

Use the less complex model because

- Simpler to use :(lower computational complexity)
- Easier to train (lower space complexity)
- Easier to explain (more interpretable)
- Generalizes better (lower variance - Occam's razor)



Bias, Variance and Generalization



Software Packages

- HTK: speech modeling kits (for hidden Markov model)
- GMTK: graphical model took kit (for speech/language)
- LIBSVM: http://www.csie.ntu.edu.tw/~cjlin/libsvm/
- NETLAB: http://www.ncrg.aston.ac.uk/netlab/
- CMU AI Repository
  - http://www.cs.cmu.edu/afs/cs/project/ai-repository/ai/areas/learning/systems/0.html
- JMLR machine learning open source software
  - http://jmlr.csail.mit.edu/mloss/
- Weka: data mining tool in Java
  - http://www.cs.waikato.ac.nz/ml/weka/
- R: http://www.r-project.org/
  - A free alternative to S-Plus developed at Bell Labs
  - If you know C, you will be right at home with R

Machine Learning Dataset Links

- UCI machine learning repository
  - http://archive.ics.uci.edu/ml/
- Open Directory Project:
  - http://www.dmoz.org/Computers/Artificial_Intelligence/Machine_Learning/Datasets/
- Datasets for knowledge discovery
  - http://www.kdnuggets.com/datasets/
- Machine learning & data mining: face, objects, etc.
  - http://cervisia.org/machine_learning_data.php
- BBC datasets: news and sports
  - http://mlg.ucd.ie/content/view/21/

An Objective of Regression

- Goal : find a "good" model $f(X)$ to predict a variable Y from a set of predictors, $X_1, X_2 \ldots X_p$
  - Quality measure: expected prediction error = $E((Y-f(X))^2)$
  - Solution: take a statistic, $f(x) = E(Y|X=x)$, but … what is $E(Y|X=x)$?
  - Decomposition of the expected prediction error EPE

If $Y = E(Y|X) + e$ with $Var(\varepsilon) = \sigma^2$ and $\hat{f}(X)$ is an estimator of the model

$$EPE = E_{Y|X=x}((y-\hat{f}(x))^2)$$

$$= E((y-E(Y|X=x))^2) + E((\hat{f}(x)-f(x))^2) + (f(x)-E(Y|X=x))^2$$

$$= \sigma^2 + Var(\hat{f}(x)) + Biais^2$$

Mean Square Error

Least Square Estimation Solutions

- Least square is the most popular method to estimate linear models by minimizing "residual" sum of squares:

$$\arg\min_{\beta} RSS(\beta) = \arg\min_{\beta} \sum_{i=1}^{N} (y_i - f(x_i))^2$$

- Normal equation: $X'(Y - X\hat{\beta}) = 0$
- Solution: $\hat{\beta} = (X'X)^{-1}X'Y$ (if psuedoinverse $(X'X)^{-1}$ exists)
- Predicted values: $\hat{y} = X\hat{\beta} = X(X'X)^{-1}X'Y = HY$
  - The predictions are an orthogonal projections of *y* in the subspace of R^n defined by the columns of the *X* matrix. *H* is the projection matrix. (refer to your textbook on linear algebra)
- The method can also be generalized to multiple *Y*'s
  - If the set of *k variables, Y_i*'s, are independent, the multivariate solution is identical to doing *k* separate classical regressions

Shrinkage Methods: Ridge Regression

- Shrink coefficients by imposing a penalty to their size

$$\hat{\beta}^{ridge} = min_{\beta}\left\{\sum_{i=1}^{N}(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda\sum_{j=1}^{p}\beta_j^2\right\},\ \lambda > 0$$

This is equivalent to $\hat{\beta}^{ridge} = min_{\beta}\left\{\sum_{i=1}^{N}(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2\right\}$ **subject to** $\sum_{j=1}^{p}\beta_j^2 \le s$

- Solutions not equivalent under scaling of inputs, standardize before solving. with centered $x_i - \overline{x}$
- Intercept with penalization and estimated by $\hat{\beta}_0 = \overline{y}$

$$RSS(\lambda) = (y - X\beta)^t(y - X\beta) + \lambda\beta\beta^t \qquad \hat{\beta}^{ridge} = (X^tX + \lambda I)^{-1}X^ty$$

If inputs are orthogonal $\hat{\beta}^{ridge} = \gamma\hat{\beta}^{LS},\ 0 \le \gamma \le 1$ is a function of λ

Autocorrelation Method: Normal Equations

- since $R_n(k)$ is even, then

$$\phi_n(i,k) = R_n(|i-k|),\ 1 \le i \le p,\ 0 \le k \le p$$

- thus the basic equation becomes

$$\sum_{k=1}^{p} \alpha_k \phi_n(i-k) = \phi_n(i,0), \quad 1 \le i \le p$$

$$\sum_{k=1}^{p} \alpha_k R_n(|i-k|) = R_n(i), \quad 1 \le i \le p$$

- with the minimum mean-squared prediction error of the form

$$E_n = \phi_n(0,0) - \sum_{k=1}^{p} \alpha_k \phi_n(0,k)$$

$$= R_n(0) - \sum_{k=1}^{p} \alpha_k R_n(k)$$

Durbin's Algorithm (Levinson Recursion)

$E^{0} = R[0]$

for $i = 1, 2, \ldots, p$

$$k_i = \left(R[i] - \sum_{j=1}^{i-1} \alpha_j^{(i-1)} R[i-j] \right) / E^{(i-1)}$$

$$\alpha_i^{(i)} = k_i$$

if $i > 1$, then for $j = 1, 2, \ldots, i-1$

$$\alpha_j^{(i)} = \alpha_j^{(i-1)} - k_i \alpha_{i-j}^{(i-1)}$$

end

$$E^{(i)} = (1 - k_i^2) E^{(i-1)}$$

end

$$\alpha_j = \alpha_j^{(p)} \qquad j = 1, 2, \ldots, p$$

The k_i are called PARCOR coefficients

$$\Rightarrow A^{(i)}(z) = A^{(i-1)}(z) - k_i z^{-i} A^{(i-1)}(z^{-1})$$

The $E^{(i)}$ are the prediction error for an i^{th}-order predictor

Linear Regression with Indicator Matrix

Indicator : $Y_k = 1 \text{ if } G = k$ $\quad k = 1, \ldots, K$

$Y_k = 0 \text{ if } G \neq k$

Indicator response matrix: $\underset{NxK}{Y}$

Example : 2 groups (K=2) and 5 observations (N=5)

$$Y = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 1 & 0 \end{bmatrix}$$

observations 1 and 5 in group 1

observations 2, 3 and 4 in group 2

⇒ Two-class classification !!

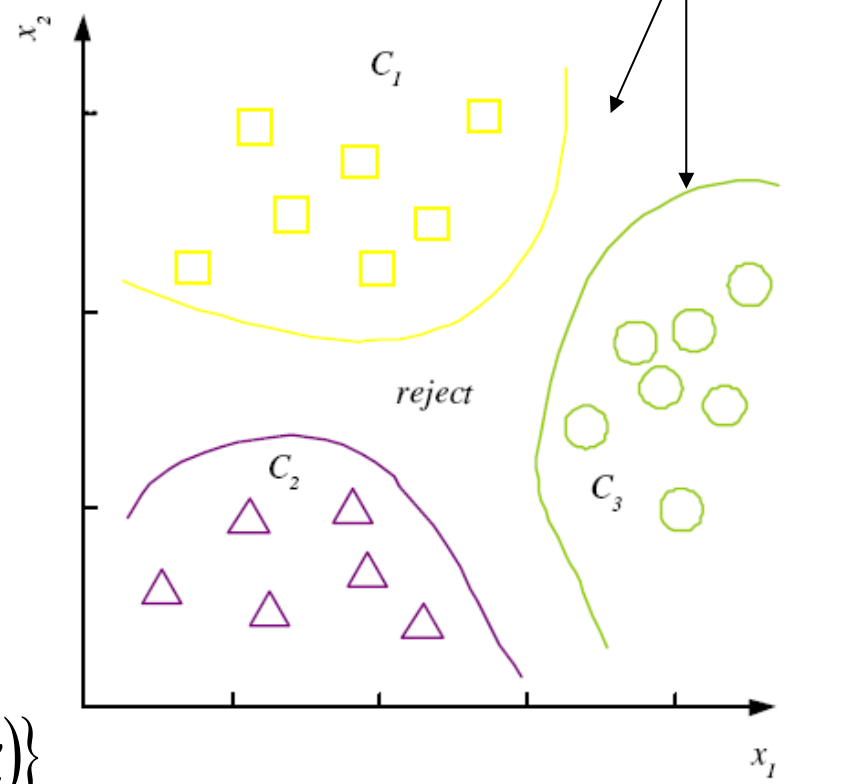
Defining Discriminant Functions

Choose C_i if $g_i(\mathbf{x}) = \max_k g_k(\mathbf{x})$

$g_i(\boldsymbol{x}), i = 1,\ldots,K$

A few examples of discriminant functions:

$$g_i(\mathbf{x}) = \begin{cases} -R(\alpha_i \mid \mathbf{x}) \\ P(C_i \mid \mathbf{x}) \\ p(\mathbf{x} \mid C_i)P(C_i) \end{cases}$$



K decision regions $\mathcal{R}_1,\ldots,\mathcal{R}_K$

$$\mathcal{R}_i = \{\boldsymbol{x} \mid g_i(\boldsymbol{x}) = \max_k g_k(\boldsymbol{x})\}$$

Linear Discriminant Analysis

$$P(G = k \mid X = x) = \frac{f_k(x)\pi_k}{\sum_{l=1}^{K} f_l(x)\pi_l}$$

density of X in class $G=k$

prior probability

$f_k(x)$ Gaussian and the class have a common covariance matrix

$\Longrightarrow$ log-ratio : $\log \frac{P(G = k \mid X = x)}{P(G = l \mid X = x)}$ is linear in x

$\Longrightarrow$ decision boundaries are linear

Discriminant function: $\delta_k(x) = x^T \Sigma^{-1} \mu_k - \frac{1}{2} \mu_k^T \Sigma^{-1} \mu_k + \log \pi_k$

Classification : $\arg\max_k \delta_k(x)$

Logistic Regression

Model specified by (*K-1*) log-odds or logit transformations:

$$\log \frac{P(G=1 \mid X=x)}{P(G=K \mid X=x)} = \beta_{10} + \beta_1^T x$$

$$\log \frac{P(G=2 \mid X=x)}{P(G=K \mid X=x)} = \beta_{20} + \beta_2^T x$$

...

$$\log \frac{P(G=K-1 \mid X=x)}{P(G=K \mid X=x)} = \beta_{(K-1)0} + \beta_{K-1}^T x$$

$$P(G=k \mid X=x) = \frac{\exp(\beta_{k0} + \beta_k^T x)}{1 + \sum_{l=1}^{K-1} \exp(\beta_{l0} + \beta_l^T x)}, \quad k = 1, \ldots, K-1$$

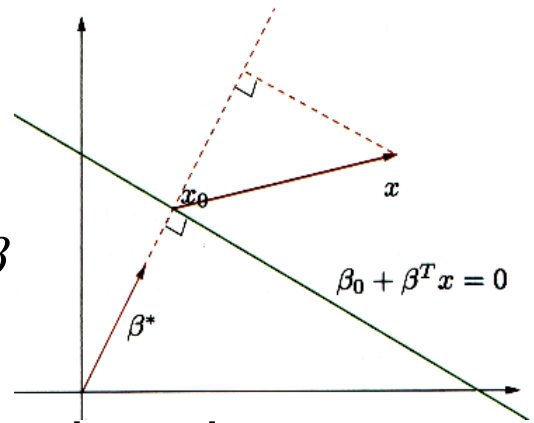
$$P(G=K \mid X=x) = \frac{1}{1 + \sum_{l=1}^{K-1} \exp(\beta_{l0} + \beta_l^T x)}$$

Separating Hyperplanes

Perceptron = classifiers such as: $\left\{x : \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 = 0\right\}$

hyperplane or affine set L: defined by the equation

$$f(x) = \beta_0 + \beta^T x = 0 \quad \text{(= a line)}$$



Properties :

- vector normal to the surface *L*: $\beta^* = \beta / \|\beta$
- for any point x_0 in *L*, $\beta^T x_0 = -\beta_0$
- the signed distance of any point *x* to *L* is given by:

$$\beta^{*T}(x - x_0) = \frac{1}{\|\beta\|}(\beta_0 + \beta^T x) = \frac{1}{\|f'(x)\|} f(x)$$

Rosenblatt's perceptron learning algorithm.

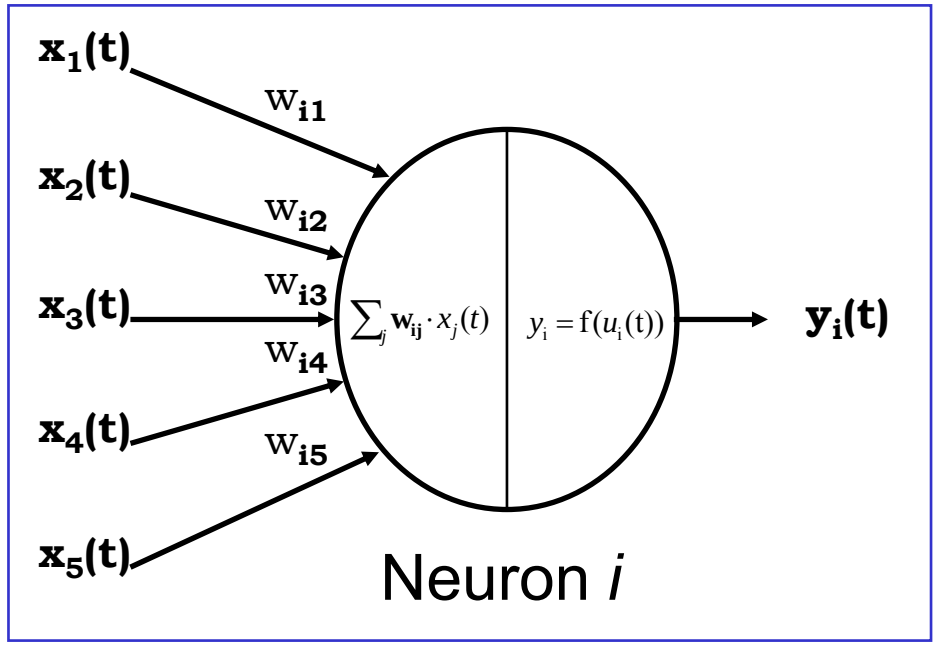
The Artificial Neuron

Stimulus

$$u_i(t) = \sum_j w_{ij} \cdot x_j(t)$$

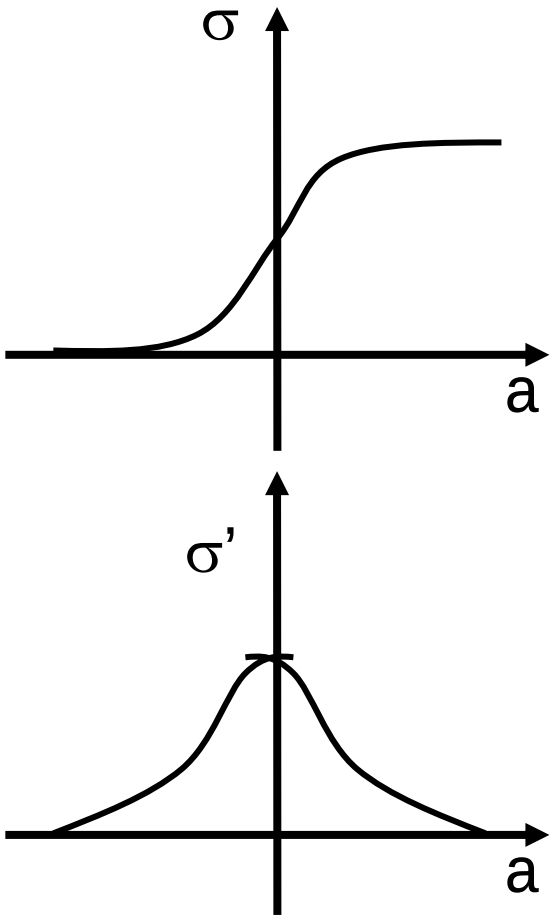
Response

$$y_i(t) = f\left(u_{rest} + u_i(t)\right)$$



u_{rest} = resting potential
$x_j(t)$ = output of neuron j at time t
w_{ij} = connection strength between neuron i and neuron j
$u(t)$ = total stimulus at time t

Gradient Descent for Sigmoid Output



Sigmoid: $E^p[w_1,\ldots,w_n] = \frac{1}{2}(t^p-y^p)^2$

t^p: Desired output

$$\partial E^p/\partial w_i = \partial/\partial w_i \, \frac{1}{2}(t^p-y^p)^2$$
$$= \partial/\partial w_i \, \frac{1}{2}(t^p- \sigma(\Sigma_i w_i x_i^p))^2$$
$$= (t^p-y^p)\,\sigma'(\Sigma_i w_i x_i^p)\,(-x_i^p)$$

for $y=\sigma(a) = 1/(1+e^{-a})$

$\sigma'(a)= e^{-a}/(1+e^{-a})^2=\sigma(a)\,(1-\sigma(a))$

$w'_i= w_i + \Delta w_i = w_i + \alpha\, y(1-y)(t^p-y^p)\, x_i^p$

Backpropagation Algorithm

Initialize each w_i to some small random value

Until the termination condition is met, Do

For each training example $<(x_1,\ldots x_n),t>$ Do

Input the instance $(x_1,\ldots,x_n)$ to the network and compute the network outputs y_k

For each output unit k

$\delta_k=y_k(1-y_k)(t_k-y_k)$

For each hidden unit h

$\delta_h=y_h(1-y_h)\ \Sigma_k\ w_{h,k}\ \delta_k$

For each network weight $w_{i,j}$ Do

$w_{i,j}=w_{i,j}+\Delta w_{i,j}$ where

$\Delta w_{i,j}=\eta\ \delta_j\ x_{i,j}$

Linear SVM: the Separable Case

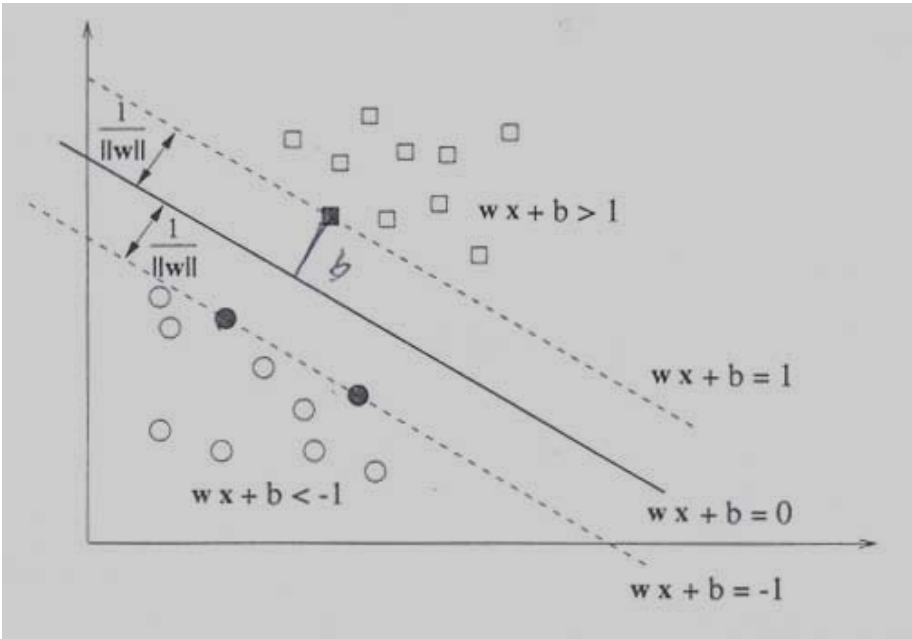
Problem 1: Minimize $\frac{1}{2}\|w\|^2$

subject to $z_k(w^t x_k + w_0) \geq 1, \quad k = 1, 2, .., n$

$$z_k(\mathbf{w}^t\mathbf{x}_k + w_0) > 1, \ for \ k = 1, 2, ..., n$$

maximize margin:

$$\frac{2}{\|w\|}$$



quadratic programming problem !

Linear SVM: Dual Formulation

- Use Lagrange optimization:

$$L(\mathbf{w}, w_0, \lambda) = \frac{1}{2} \| \mathbf{w} \|^2 - \sum_{k=1}^{n} \lambda_k [z_k (\mathbf{w}^t \mathbf{x}_k + w_0) - 1], \quad \lambda_k \geq 0$$

- Easier to solve the "dual" problem:

$$\textbf{Problem 2: Maximize} \sum_{k=1}^{n} \lambda_k - \frac{1}{2} \sum_{k,j}^{n} \lambda_k \lambda_j z_k z_j \mathbf{x}_j^t \mathbf{x}_k$$

$$\text{subject to} \sum_{k=1}^{n} z_k \lambda_k = 0, \quad \lambda_k \geq 0, k = 1, 2, . . , n$$

- Decision function

$$f(x) = \text{sign}(\Sigma_i \, \alpha_i \, y_i \langle x, x_i \rangle + b)$$

Linear SVM: Solution

- The solution is given by:

$$w = \sum_{k=1}^{n} z_k \lambda_k \mathbf{x}_k$$

$$w_0 = z_k - w^t \mathbf{x}_k$$

only support vectors contribute to the solution!!

$$g(x) = w^T \bullet x + w_0 = \sum_k z_k \lambda_k (x^T \bullet x_k) + w_0$$

- It can be shown that if x_k is <u>not</u> a support vector, then $\lambda_k = 0$

Linear SVM: the Non-Separable Case

- Allow misclassifications (i.e., soft margin classifier) by introducing error variables ψ_k :

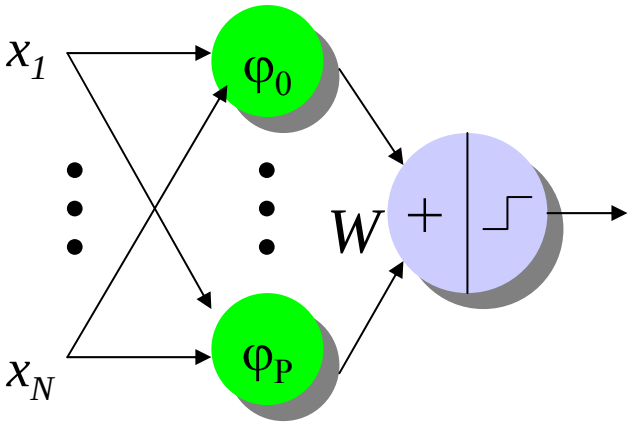
$$z_k(w^t \mathbf{x}_k + w_0) \geq 1 - \psi_k, \quad k = 1, 2, ..., n$$

Problem 3: Minimize $\frac{1}{2}\|w\|^2 + c\sum_{k=1}^{n}\psi_k$

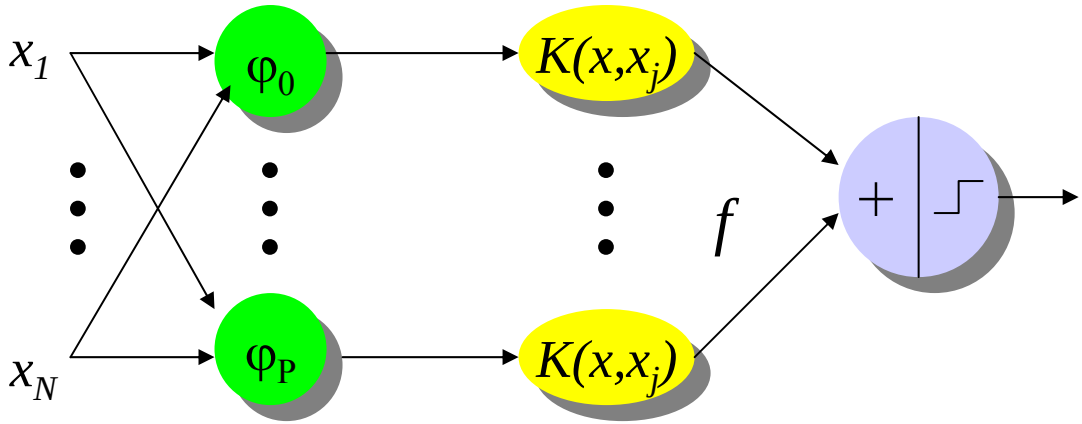
subject to $z_k(w^t x_k + w_0) \geq 1 - \psi_k, \quad k = 1, 2, .., n$

- The result is a hyperplane that minimizes the sum of errors ψ_k while maximizing the margin for the correctly classified data.

SVM Using Nonlinear Kernels



Nonlinear transform



Kernel evaluation

Using kernel, low dimensional feature vectors will be mapped to high dimensional (may be infinite dim) kernel feature space where the data are likely to be linearly separable

The Kernel Trick

- Compute dot products using a kernel function:
  - "Given an algorithm which is formulated in terms of a positive definite kernel *K1*, one can construct an alternative algorithm by replacing *K1* with another positive definite kernel *K2*"

$$K(\mathbf{x}, \mathbf{x}_k) = \Phi(\mathbf{x}) . \Phi(\mathbf{x}_k)$$

- Advantages of using a kernel
  - No need to know $\Phi()$!!
  - The discriminant is given by:

$$g(\mathbf{x}) = \sum_{k=1}^{n} z_k \lambda_k K(\mathbf{x}, \mathbf{x}_k) + w_0$$

Interpolation Problem Formulation

Radial Basis function for interpolation:

Given $\{\boldsymbol{x}_i;\ 1 \le i \le K\}$ and $\{d_i;\ 1 \le i \le K\}$, find a function $F(\boldsymbol{x})$ that satisfies the interpolation condition:

$$F(\boldsymbol{x}_i) = d_i \qquad 1 \le i \le K \quad (1)$$

One possible choice of $F(\boldsymbol{x})$ is a radial basis function of the following form:

$$F(x) = \sum_{i=1}^{K} w_i \varphi(\| x - x_i \|) \quad (2)$$

where $\{\boldsymbol{x}_i;\ 1 \le i \le K\}$ are the centers of the radial basis functions.

Gaussian Mixture Model (GMM)

- Gaussian mixture model (GMM)
  - Multivariate density: $p(X) = \sum_{k=1}^{K} \omega_k \cdot N(X \mid \mu_k, \Sigma_k)$
  - GMM is a mixture of single Gaussian distribution (each one is called mixand) which have different means and variances
  - ω_k is called mixture weight, prior probability of each mixand. They satisfy $\sum_{k=1}^{K} \omega_k = 1$
- GMM is widely used for speaker recognition, audio classification, audio segmentation, etc.

Splines

- **Def:** A piecewise polynomial is called *order-M spline* if it has continuous derivatives up to order *M-1* at the knots
- Alternative: An order-*M* spline is a function which can be represented by basis functions (*K* = # of knots)

$$h_j(X) = X^{j-1},\ j = 1 \ldots M$$

$$h_{M+l}(X) = (X - \xi_l)_+^{M-1},\ l = 1 \ldots K$$

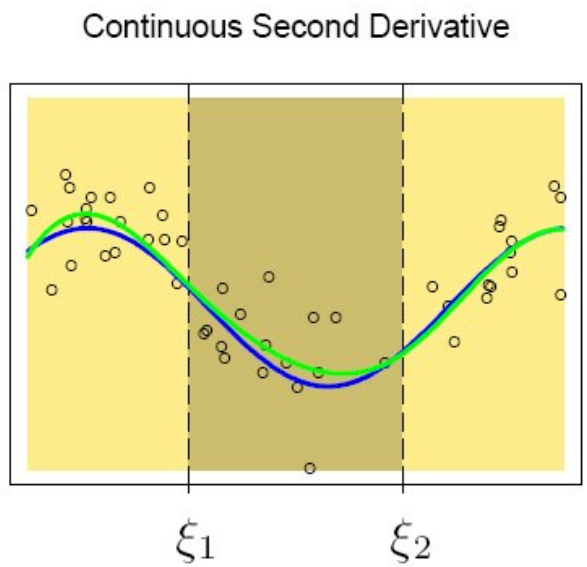
- Theorem. The definitions above are equivalent
- **Def:** Order-4 spline is called *cubic spline (look at basis and compare the number of free parameters)*
- Note. Cubic splines: knot-discontinuity is not visible

Cubic Spline

- Fit cubic polynomial in each region: 3 types of constraints:
  1. Zero order continuity (i.e., continuity of the curve) at knot points
  2. First order continuity (i.e., has first derivative) at knot points
  3. Second order continuity (i.e., has second derivative) at knot points
- The model fits a smooth curve to the data

Basis functions:

$$h_1(X) = 1$$
$$h_2(X) = X$$
$$h_3(X) = X^2$$
$$h_4(X) = X^3$$
$$h_5(X) = (X - \xi_1)_+^3$$
$$h_6(X) = (X - \xi_2)_+^3$$



Natural Cubic Splines

- In order to reduce the variance of cubic spline near the boundary knots, natural cubic spline places some rigidity in the model – the function is linear beyond the two boundary knots (2nd and 3rd derivatives are zero)
- This frees up some degrees of freedom in the cubic spline (i.e., it is not as wild near the boundaries now)
- There are *K* basis functions for *K* knots:

$$N_1(X) = 1$$

$$N_2(X) = X$$

$$N_{k+2}(X) = d_k(x) - d_{K-1}(X)$$

$$d_k(x) = \frac{(X - \xi_k)_+^3 - (X - \xi_K)_+^3}{\xi_K - \xi_k}$$

Decision Tree Notation (Cont.)

- Define the data that gets to a node μ recursively

$$\Lambda_{\mu 1} = \{x_a \in \Lambda_\mu \ s.t. T_\mu(x_a) = T\}.$$
$$\Lambda_{\mu 2} = \{x_a \in \Lambda_\mu \ s.t. T_\mu(x_a) = F\}.$$

- The root node contains all data $\Lambda_{\mu_0} = \Lambda$

- Define Impurity (entropy)

$$I(\mu) = -\sum_j P_\mu(\omega_j) \log_2 P_\mu(\omega_j).$$
$$P_\mu(\omega_j) = \sum_{a \in \Lambda_\mu} \delta_{\omega_a, \omega_j} / |\Lambda_\mu|$$

Learning Decision Tree

- Iterative Design Principle: For all leaf nodes, calculate the ***maximal decrease in impurity*** (by searching over all tests). Expand the leaf node with maximal decrease and add its child nodes to the tree
- Decrease in impurity:

$$\Delta I(\mu) = I(\mu) - I(\mu_1, \mu_2).$$

$$I(\mu_1, \mu_2) = \frac{|\Lambda_{\mu 1}|}{|\Lambda_\mu} I(\mu_1) + \frac{|\Lambda_{\mu 2}|}{|\Lambda_\mu} I(\mu_2)$$

- The lower the impurity, the easier to classify

Learning Decision Tree (Cont.)

- Greedy Strategy:
  - Start at root node μ_0. select the test $T^*_{\mu_0}$ that has maximum $\Delta I(\mu_0)$,
  - For each child node, μ_a select the test that has maximal $\Delta I(\mu_a)$.
  - Repeat until each node is a pure leaf nod $I(\mu) = 0$
  - Classify leaf nodes by majority vote
- Note: learning algorithm is $O(|\Phi||\Lambda|\{\log|\Lambda|\}^2)$.
- Run time is $O(\log|\Lambda|)$. (Test rules & data points)

Summary: Clustering Problem

- Given a set of vectors $\{x_n; 1 \le n \le N\}$, find a set of K clustering centers $\{\boldsymbol{w}(k); 1 \le k \le K\}$ such that each x_n is assigned to a cluster, say, $\mathbf{w}(k^*)$, according to a distance (distortion, similarity) measure $d(x_n, \mathbf{w}(k))$ to minimize average distortion

$$D = \frac{1}{N} \sum_{k=1}^{K} \sum_{n=1}^{N} I(x_n, k) d(x_n, w(k))$$

- $\boldsymbol{I}(x_n,k) = 1$ if x is assigned to cluster k with cluster center $w(k)$; and = 0 otherwise -- indicator function
- Key: reasonable distortion (or similarity) measure

Summary: *K*-means (EM) Algorithm

Initialization: Initial cluster center $\boldsymbol{w}(k)$; $1 \le k \le K$, $D(-1) = 0$, $\boldsymbol{I}(x_n,k) = 0$, $1 \le n \le N$, $1 \le k \le K$;

Repeat

(A) Assign cluster membership (Expectation step)

Evaluate $d(x_n, \boldsymbol{w}(k))$; $1 \le n \le N$, $1 \le k \le K$

$\boldsymbol{I}(x_n,k) = 1$ if $d(x_n, \boldsymbol{w}(k)) < d(x_n, \boldsymbol{w}(m))$, $m \ne k$; $1 \le n \le N$

$= 0$; otherwise

(B) Evaluate distortion D: $D(iter) = \sum_{n=1}^{N} I(x_n,k) d(x_n, w(k)) \quad 1 \le k \le K$

(C) Update code words with new assignment (Maximization)

$$w(k) = \sum_{n=1}^{N} I(x_n,k) x_n, \; N_k = \sum_{n=1}^{N} I(x_n,k), 1 \le k \le K$$

(D) Check for convergence

if $1 - D(\text{Iter}-1)/D(\text{Iter}) < e$, then convergence = TRUE,