CSE6328 3.0
Speech & Language Processing

YORK UNIVERSITÉ UNIVERSITY redefine THE POSSIBLE.

No.5

Pattern Classification (III) & Pattern Verification

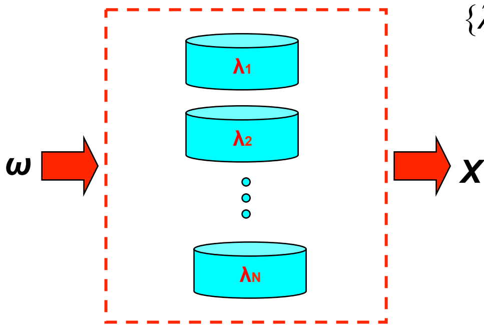
Prof. Hui Jiang
Department of Computer Science and Engineering
York University

Model Parameter Estimation

- Maximum Likelihood (ML) Estimation:
  - ML method: most popular model estimation
  - EM (Expected-Maximization) algorithm
  - Examples:
    - Univariate Gaussian distribution
    - Multivariate Gaussian distribution
    - Multinomial distribution
    - Gaussian Mixture model
    - Markov chain model: n-gram for language modeling
    - Hidden Markov Model (HMM)
- Discriminative Training — alternative model estimation method
  - Maximum Mutual Information (MMI)
  - Minimum Classification Error (MCE)
  - Large Margin Estiamtion (LME)
- Bayesian Model Estimation: Bayesian theory
- MDI (Minimum Discrimination Information)

Discriminative Training(I): Maximum Mutual Information Estimation (1)

- **The model is viewed as a noisy data generation channel**
  class id *ω* ➔ observation feature *X*.
- **Determine model parameters to maximize mutual information between *ω* and *X*. (close relation between *ω* and *X*)**



noisy data generation channel

$$\{\lambda_1 \cdots \lambda_N\}_{MMI} = \arg\max_{\lambda_1 \cdots \lambda_N} \; I(\omega, X)$$

$$I(\omega, X) = \sum_{\omega} \sum_{X} p(\omega, X) \log_2 \frac{p(\omega, X)}{p(\omega) p(X)}$$

$$= \sum_{\omega} \sum_{X} p(\omega, X) \log_2 \frac{p(X \mid \omega)}{p(X)}$$

$$= \sum_{\omega} \sum_{X} p(\omega, X) \log_2 \frac{p(X \mid \omega)}{\sum_{\omega} p(X \mid \omega)}$$

$$= \sum_{\omega} \sum_{X} p(\omega, X) \log_2 \frac{p(X \mid \lambda_\omega)}{\sum_{\omega} p(X \mid \lambda_\omega)}$$

Discriminative Training(I): Maximum Mutual Information Estimation (2)

- **Difficulty: joint distribution *p(ω,X)* is unknown.**
- **Solution: collect a representative training set *(X_1, ω_1), (X_2, ω_2), …, (X_T, ω_T)* to approximate the joint distribution.**

$$\{\lambda_1 \cdots \lambda_N\}_{MMI} = \arg\max_{\lambda_1 \cdots \lambda_N} \; I(\omega, X)$$

$$= \arg\max_{\lambda_1 \cdots \lambda_N} \sum_{\omega} \sum_{X} p(\omega, X) \log_2 \frac{p(X \mid \lambda_\omega)}{\sum_{\omega} p(X \mid \lambda_\omega)}$$

$$\approx \arg\max_{\lambda_1 \cdots \lambda_N} \sum_{t=1}^{T} \log_2 \frac{p(X_t \mid \lambda_{\omega_t})}{\sum_{\omega} p(X_t \mid \lambda_{\omega_t})}$$

- **Optimization:**
  - **Iterative gradient-ascent method**
  - **Growth-transformation method**

Discriminative Training(II): Minimum Classification Error Estimation (1)

- In a N-class pattern classification problem, given a set of training data, $D=\{ (X_1, \omega_1), (X_2, \omega_2), \dots, (X_T, \omega_T)\}$, estimate model parameters for all class to minimize total classification errors in *D.*
  - *MCE: minimize empirical classification errors*
- Objective function ➔ total classification errors in *D*
  - For each training data, (X_t, ω_t), define misclassification measure:

$$d(X_t,\omega_t) = -p(\omega_t)p(X_t \mid \lambda_{\omega_t}) + \max_{\omega_{t'} \neq \omega_t} p(\omega_{t'})p(X_t \mid \lambda_{\omega_{t'}})$$

or

$$d(X_t,\omega_t) = -\ln[p(\omega_t)p(X_t \mid \lambda_{\omega_t})] + \max_{\omega_{t'} \neq \omega_t} \ln[p(\omega_{t'})p(X_t \mid \lambda_{\omega_{t'}})]$$

if $d(X_t, \omega_t)>0$, *incorrect classification for* X_t ➔ *1 error*

if $d(X_t, \omega_t)<=0$, *correct classification for* X_t ➔ *0 error*

Discriminative Training(II): Minimum Classification Error Estimation (2)

- Soft-max: approximate $d(X_t, \omega_t)$ by a differentiable function:

$$d(X_t,\omega_t) \approx -p(\omega_t)p(X_t \mid \lambda_{\omega_t}) + \ln\left[\frac{1}{N-1}\sum_{\omega_{t'} \neq \omega_t} \exp[\eta \cdot p(\omega_{t'})p(X_t \mid \lambda_{\omega_{t'}})]\right]^{1/\eta}$$

or

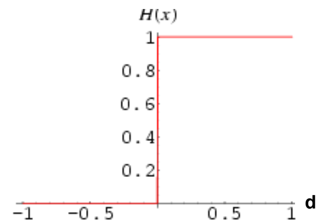
$$d(X_t,\omega_t) \approx -\ln[p(\omega_t)p(X_t \mid \lambda_{\omega_t})] + \ln\left[\frac{1}{N-1}\sum_{\omega_{t'} \neq \omega_t} \exp[\eta \cdot \ln(p(\omega_{t'})p(X_t \mid \lambda_{\omega_{t'}}))]\right]^{1/\eta}$$

where $\eta>1$.

Discriminative Training(II): Minimum Classification Error Estimation (3)

- Error count for one data, *(X_t, ω_t), is $H(d(X_t, \omega_t))$, where H(.) is step function.*
- Total errors in training set:

$$Q(\Lambda) = \sum_{t=1}^{T} H(d(X_t, \omega_t))$$

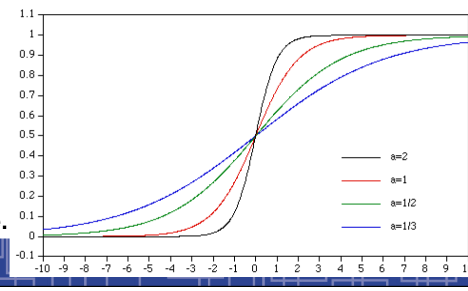


- Step function is not differentiable, approximated by a sigmoid function ➔ smoothed total errors in training set.

$$Q(\Lambda) \approx Q'(\Lambda) = \sum_{t=1}^{T} l(d(X_t, \omega_t))$$

where $l(d) = \dfrac{1}{1 + e^{-a \cdot d}}$

a>0 is a parameter to control its shape.



Discriminative Training(II): Minimum Classification Error Estimation (3)

- MCE estimation of model parameters for all classes:

$$\{\lambda_1 \cdots \lambda_N\}_{MCE} = \arg\min_{\lambda_1 \cdots \lambda_N} \; Q'(\lambda_1 \cdots \lambda_N)$$

- Optimization: no simple solution is available
  - Iterative gradient descent method.
  - GPD (generalized probabilistic descent) method.

$$\lambda_i^{(n+1)} = \lambda_i^{(n)} - \varepsilon \cdot \frac{\partial}{\partial \lambda_i} Q'(\lambda_1 \cdots \lambda_N) \,|_{\lambda_i = \lambda_i^{(n)}}$$

The MCE/GPD Method

- Find initial model parameters, e.g., ML estimates
- Calculate gradient of the objective function
- Calculate the value of the gradient based on the current model parameters
- Update model parameters

$$\lambda_i^{(n+1)} = \lambda_i^{(n)} - \varepsilon \cdot \frac{\partial}{\partial \lambda_i} Q'(\lambda_1 \cdots \lambda_N)|_{\lambda_i = \lambda_i^{(n)}}$$

- Iterate until convergence

How to calculate gradient?

$$\frac{\partial}{\partial \lambda_i} Q'(\lambda_1 \cdots \lambda_N) = \sum_{t=1}^{T} \frac{\partial}{\partial \lambda_i} l\big[d(X_t, \omega_t)\big]$$

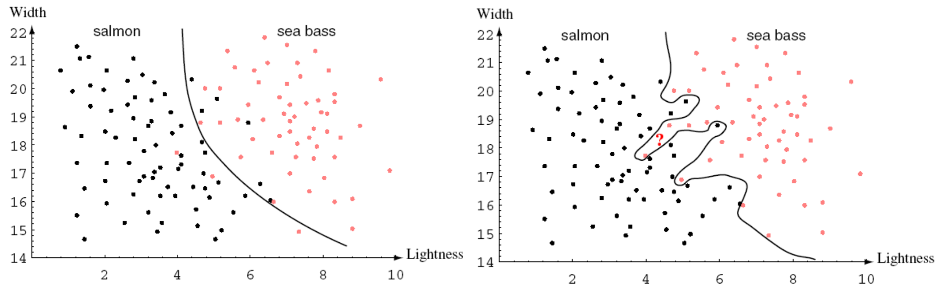
$$= \sum_{t=1}^{T} \frac{\partial l(d)}{\partial d} \cdot \frac{\partial d(X_t, \omega_t)}{\partial \lambda_i}$$

$$= \sum_{t=1}^{T} a \cdot l(d) \cdot [1 - l(d)] \cdot \frac{\partial d(X_t, \omega_t)}{\partial \lambda_i}$$

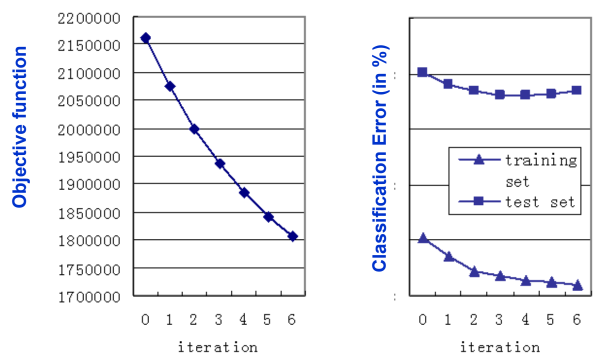
- The key issue in MCE/GPD is how to set a proper step size experimentally.

Overtraining (Overfitting)

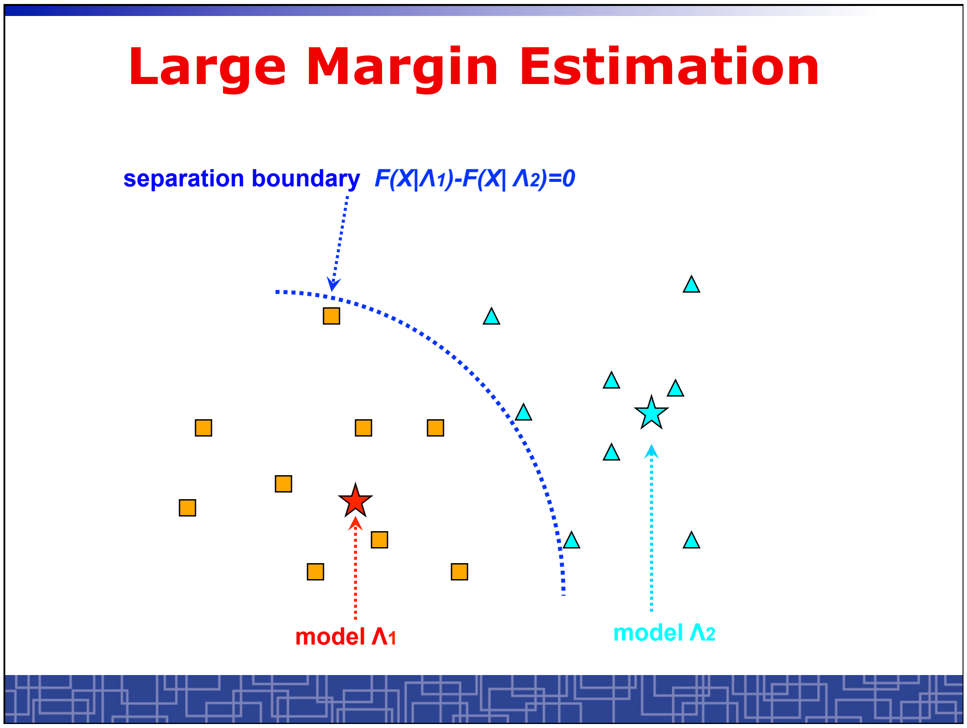
- Low classification error rate in training set does not always lead to a low error rate in a new test set due to overtraining.

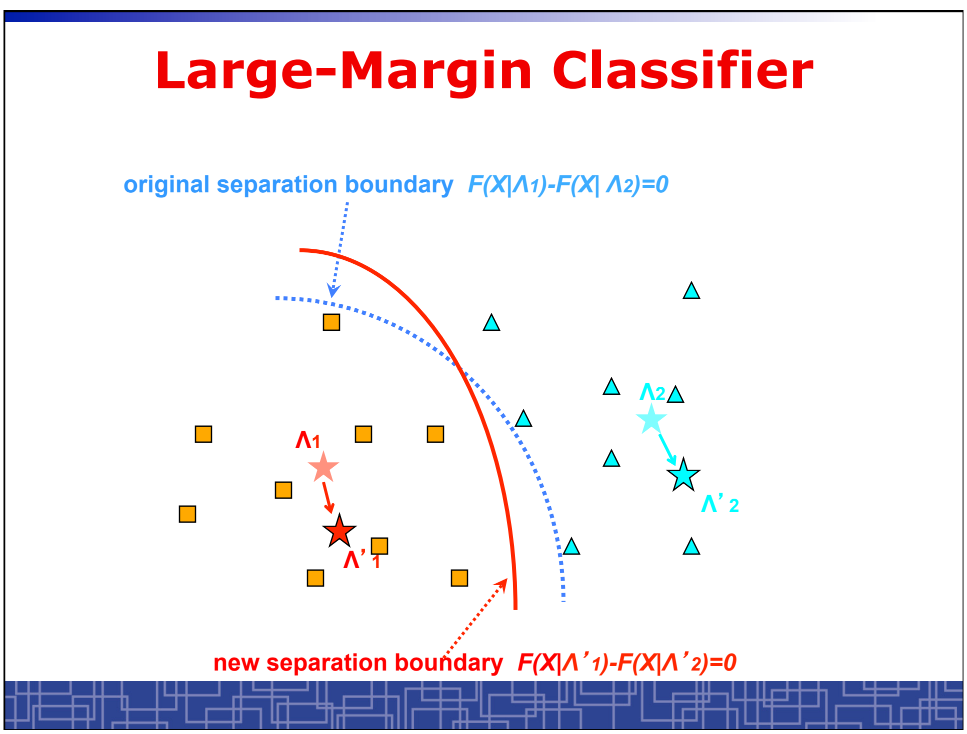


Measuring Performance of MCE



- When to converge: monitor three quantities in the MCE/GPD
  - The objective function
  - Error rate in training set
  - Error rate in test set


Large Margin Estimation
separation boundary F(X|Λ1)-F(X| Λ2)=0
model Λ1
model Λ2


Large-Margin Classifier
original separation boundary F(X|Λ1)-F(X| Λ2)=0
Λ1
Λ'1
Λ2
Λ'2
new separation boundary F(X|Λ'1)-F(X|Λ'2)=0

How to define separation *margin?* (1)

- In 2-class separable problem:
  - For a data token, x_1, of class Λ_1

$$d(x_1) = F(x_1|\Lambda_1) - F(x_1|\Lambda_2) \quad > 0$$

  - For a data token, x_2, of class Λ_2

$$d(x_2) = F(x_2|\Lambda_2) - F(x_2|\Lambda_1) \quad > 0$$

How to define separation *margin?* (2)

- Extend to multiple-class problem:
  - N classes $\Lambda_1, \Lambda_2, \ldots, \Lambda_N$,
  - For a data token, x_i, of class Λ_i

$$d(x_i) = F(x_i|\Lambda_i) - \max_{j \neq i} F(x_i|\Lambda_j) = \min_{j \neq i} \left[F(x_i|\Lambda_i) - F(x_i|\Lambda_j) \right]$$

Large Margin Estimation

- An *N*-class problem: each class is represented by one model

$$\mathbf{\Lambda} = \{\Lambda_1, \Lambda_2, \cdots, \Lambda_N\}$$

- Given a training set *D*, define a subset, called *support token set S*, based on initial model as:

$$S = \{X_i \mid X_i \in D \text{ and } 0 \le d(X_i) \le \varepsilon\}$$

- Large-Margin Estimation (LME):

$$\hat{\mathbf{\Lambda}} = \arg\max_{\mathbf{\Lambda}} \min_{X_i \in S} d(X_i) \quad (\text{subject to all } d(X_i) > 0)$$

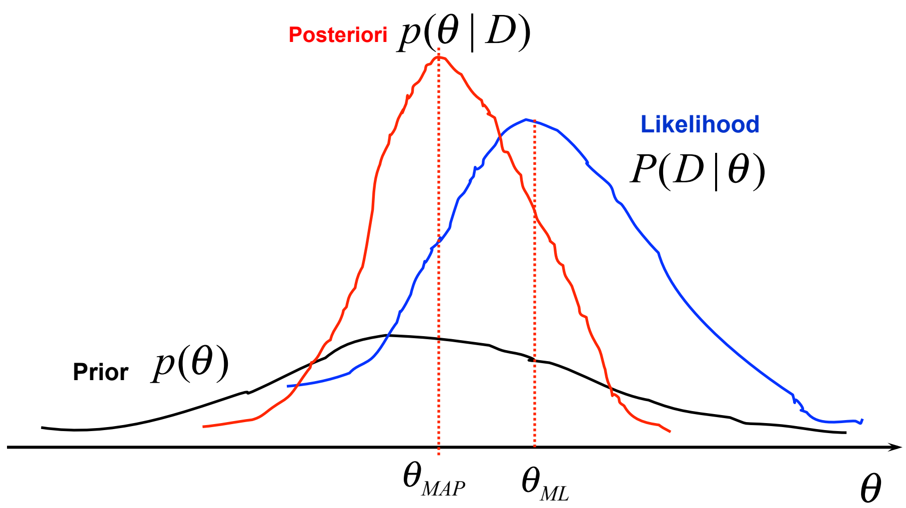
Bayesian Theory

- Bayesian methods view model parameters as random variables having some known prior distribution. (Prior specification)
  - Specify prior distribution of model parameters θ as p(θ).
- Training data *D* allow us to convert the prior distribution into a posteriori distribution. (Bayesian learning)

$$p(\theta \mid D) = \frac{p(\theta) \cdot p(D \mid \theta)}{p(D)} \propto p(\theta) \cdot p(D \mid \theta)$$

- We infer or decide everything solely based on the posteriori distribution. (Bayesian inference)
  - Model estimation: the MAP (maximum a posteriori) estimation
  - Pattern Classification: Bayesian classification
  - Sequential (on-line, incremental) learning
  - Others: prediction, model selection, etc.

Bayesian Learning



The MAP estimation of model parameters

- Do a point estimate about θ based on the posteriori distribution

$$\theta_{MAP} = \arg\max_{\theta} \ p(\theta | D) = \arg\max_{\theta} \ p(\theta) \cdot p(D | \theta)$$

- Then θ_{MAP} is treated as estimate of model parameters (just like ML estimate). Sometimes need the EM algorithm to derive it.

- MAP estimation optimally combine prior knowledge with new information provided by data.

- MAP estimation is used in speech recognition to adapt speech models to a particular speaker to cope with various accents
  - From a generic speaker-independent speech model ➔ prior
  - Collect a small set of data from a particular speaker
  - The MAP estimate give a speaker-adaptive model which suit better to this particular speaker.

Bayesian Classification

- Assume we have *N* classes, ω_i $(i=1,2,\ldots,N)$, each class has a class-conditional pdf $p(X|\omega_i,\theta_i)$ with parameters θ_i.
- The prior knowledge about θi is included in a prior $p(\theta_i)$.
- For each class ω_i, we have a training data set D_i.
- Problem: classify an unknown data *Y* into one of the classes.
- The Bayesian classification is done as:

$$\omega_Y = \arg\max_i p(Y \mid D_i) = \arg\max_i \int p(Y \mid \omega_i, \theta_i) \cdot p(\theta_i \mid D_i)\, \mathrm{d}\theta_i$$

where

$$p(\theta_i \mid D_i) = \frac{p(\theta_i) \cdot p(D_i \mid \omega_i, \theta_i)}{p(D_i)} \propto p(\theta_i) \cdot p(D_i \mid \omega_i, \theta_i)$$

Recursive Bayes Learning (Sequential Bayesian Learning)

- Bayesian theory provides a framework for *on-line learning* (a.k.a. *incremental learning*, *adaptive learning*).
- When we observe training data one by one, we can dynamically adjust the model to learn incrementally from data.
- Assume we observe training data set $D=\{X_1,X_2,\ldots,X_n\}$ one by one,

$$p(\theta) \xrightarrow{X_1} p(\theta \mid X_1) \xrightarrow{X_2} p(\theta \mid X_1, X_2) \cdots\cdots p(\theta \mid D^{(n)})$$

Learning Rule: $posteriori \propto prior \times likelihood$

Knowledge about Model at this stage | Knowledge about Model at this stage | Knowledge about Model at this stage | Knowledge about Model at this stage

How to specify priors

- Noninformative priors
  - In case we don't have enough prior knowledge, just use a flat prior at the beginning.
- *Conjugate priors*: for computation convenience
  - For some models, if their probability functions are a reproducing density, we can choose the prior as a special form (called *conjugate prior*), so that after Bayesian leaning the posterior will have the exact same function form as the prior except the all parameters are updated.
  - Not every model has conjugate prior.

Conjugate Prior

- For a univariate Gaussian model with only unknown mean:

$$p(x|\omega_i) = N(x|\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}}\exp[-\frac{(x-\mu)^2}{2\sigma^2}]$$

- If we choose the prior as a Gaussian distribution (Gaussian's conjugate prior is Gaussian)

$$p(\mu) = N(\mu|\mu_0,\sigma_0^2) = \frac{1}{\sqrt{2\pi\sigma_0^2}}\exp[-\frac{(\mu-\mu_0)^2}{2\sigma_0^2}]$$

- After observing a new data x_1, the posterior will still be Gaussian:

$$p(\mu|x_1) = N(\mu|\mu_1,\sigma_1^2) = \frac{1}{\sqrt{2\pi\sigma_1^2}}\exp[-\frac{(\mu-\mu_1)^2}{2\sigma_1^2}]$$

where $$\mu_1 = \frac{\sigma_0^2}{\sigma_0^2+\sigma^2}x_1 + \frac{\sigma^2}{\sigma_0^2+\sigma^2}\mu_0$$

$$\sigma_1^2 = \frac{\sigma_0^2\sigma^2}{\sigma_0^2+\sigma^2}$$

The sequential MAP Estimate of Gaussian

- For univariate Gaussian with unknown mean, the MAP estimate of its mean after observing x_1:

$$\mu_1 = \frac{\sigma_0^2}{\sigma_0^2 + \sigma^2} x_1 + \frac{\sigma^2}{\sigma_0^2 + \sigma^2} \mu_0$$

- After observing next data x_2:

$$\mu_2 = \frac{\sigma_1^2}{\sigma_1^2 + \sigma^2} x_2 + \frac{\sigma^2}{\sigma_1^2 + \sigma^2} \mu_1$$