COSC6328 3.0
Speech & Language Processing

No.5

Pattern Classification (III) & Pattern Verification & WFST

Prof. Hui Jiang
Department of Computer Science
York University

Model Parameter Estimation

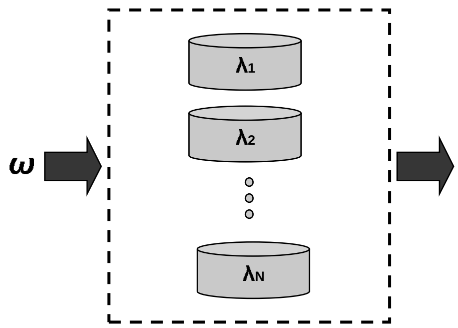
- Maximum Likelihood (ML) Estimation:
  - ML method: most popular model estimation
  - EM (Expected-Maximization) algorithm
  - Examples:
    - Univariate Gaussian distribution
    - Multivariate Gaussian distribution
    - Multinomial distribution
    - Gaussian Mixture model
    - Markov chain model: n-gram for language modeling
    - Hidden Markov Model (HMM)
- Discriminative Training — alternative model estimation method
  - Maximum Mutual Information (MMI)
  - Minimum Classification Error (MCE)
- 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 $(X_1, \omega_1), (X_2, \omega_2), \ldots, (X_T, \omega_T)$ 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, ω_1), (X_2, ω_2), …, (X_T, ω_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, ω_t)>0, incorrect classification for X_t ➔ 1 error*
if *d(X_t, ω_t)<=0, correct classification for X_t ➔ 0 error*

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

- **Approximate *d(X_t, ω_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

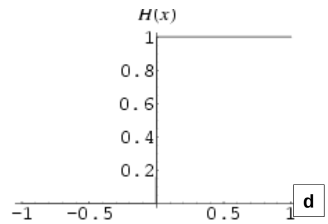
$$d(X_t,\omega_t) \approx -\ln[\,p(\omega_t)\,p(X_t \mid \lambda_{\omega_t})] + \ln\left[\frac{1}{N-1}\sum_{\omega_{t'} \neq \omega_t} \exp[\eta \cdot \ln(\,p(\omega_{t'})\,p(X_t \mid \lambda_{\omega_{t'}}))]\right]^{1/\eta}$$

where η>1.

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

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

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

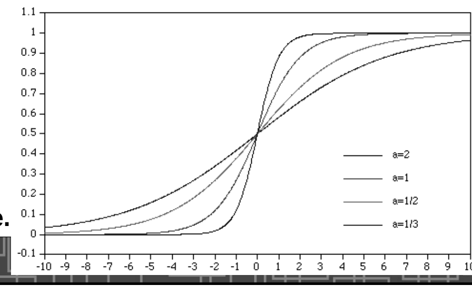


- **Step function is not differentiable, approximated by a sigmoid function ➔ smoothed total errors in training set.**

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

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.**

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

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?

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

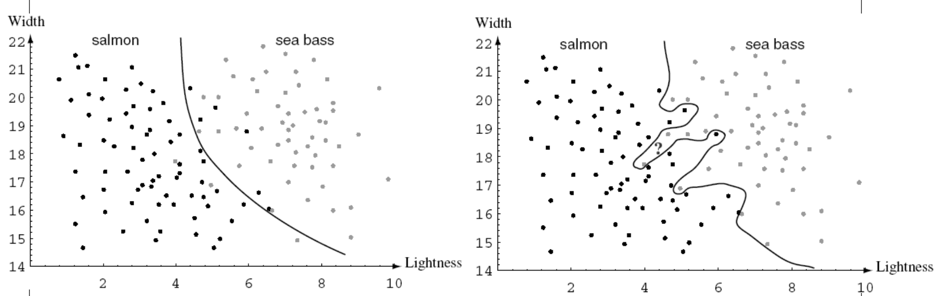
$$= \sum_{t=1}^{T} \frac{\partial l(d)}{\partial d} \cdot \frac{\partial d(X_t, \omega_t)}{\partial \lambda_i}$$

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

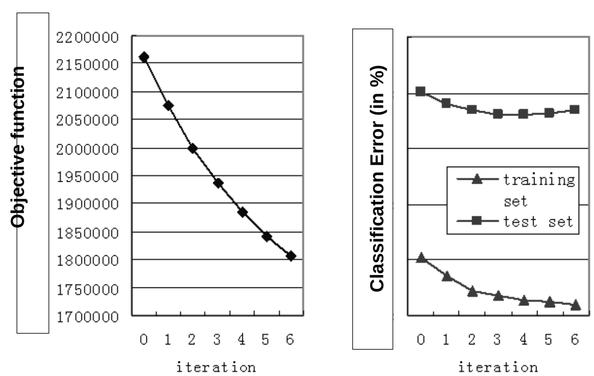
- The key issue in MCE/GPD is how to set a proper step size experimentally.

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**

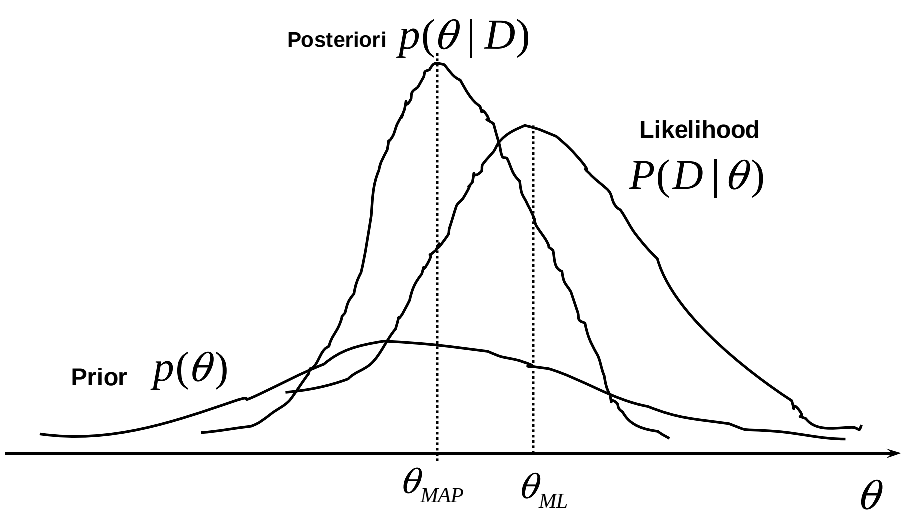
Bayesian Theory

- 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

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

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

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

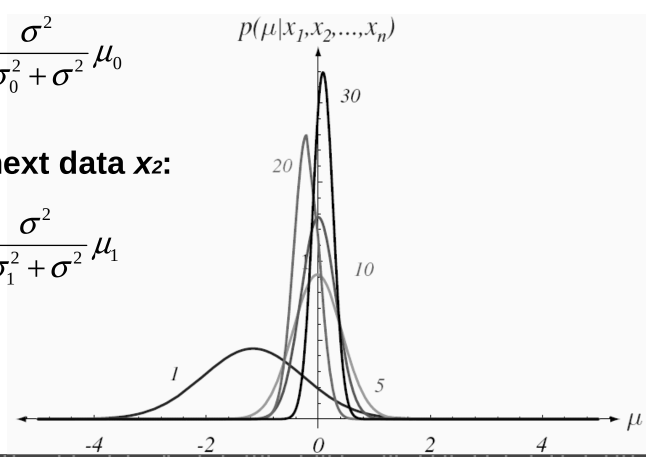
The sequential MAP Estimate of Gaussian

- 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:

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



Pattern classification based on Discriminant Functions (I)

- Instead of designing a classifier based on probability distribution of data, we can build an ad-hoc classifier based on some discriminant functions to model class boundary info directly.
- Classifier based on discriminant functions:
  - For *N* classes, we define a set of discriminant functions $g_i(X)$ $(i=1,2,\ldots,N)$, one for each class.
  - For an unknown pattern with feature vector *Y*, the classifier makes the decision as

$$\omega_Y = \arg\max_i g_i(Y)$$

  - Each discriminant function $g_i(X)$ has a pre-defined function form and a set of unknown parameters θ_i, rewrite it as $g_i(X\,;\,\theta_i)$.
  - Similarly θ_i $(i=1,2,\ldots,N)$ need to be estimated from some training data.

Pattern classification based on Discriminant Functions (II)

- Some common forms for discriminant funtions:
  - Linear discriminant function:

$$g(X) = w^t \cdot X + w_0$$

  - Quadratic discrimiant function: (2^{nd} order)
  - Polynomial discriminant function: (N-th order)
  - *Neural network*: (arbitrary nonlinear functions)
  - Optimal MAP classifier is a special case when choosing discriminant functions as class posterior probabilities.

- Unknown parameters of discriminant functions are estimated by some gradient descent method to minimize an objective function, such as empirical classification errors in training set, etc.

Pattern Verification

- For an unknown pattern/object P, we can observe/measure some features X of the pattern P.
- Based on the features X, we need to answer a binary question (Yes/No) regarding P.
- Example of pattern verification: speaker id verification
  - A user claims its id as *abc*;
  - System prompts and records some voice *X* from the user.
  - Based on the voice *X*, system makes a decision whether the user is *abc* or not. (voiceprints for security)
- Pattern verification can be viewed as a 2-class classification problem; but better not to do so.
- A proper view is to cast it as a *statistical hypothesis testing* problem.

Statistical Hypothesis Testing(I)

- In statistics, we normally need test a hypothesis based on some observation data. The problem is formulated as a test between two complementary hypotheses:
  - *H_0: null hypothesis*
  - *H_1: alternative hypothesis*
- Example: Given $X_1, X_2, \ldots, X_n$ is a random sample from a Gaussian distribution $N(\mu, \sigma^2)$, where variance σ^2 is known. We need to verify whether its mean is a given value or not. Thus we do hypothesis testing between:
  - $H_0 : \mu = \mu_0$ against
    $H_1 : \mu \neq \mu_0$
- In Hypothesis testing, we have two types of errors:
  - Type I: false rejection error; falsely reject H_0 when H_0 is true.
  - Type II: false alarm error; falsely accept H_0 when H_1 is true.

Statistical Hypothesis Testing(II)

- **In essence, a hypothesis test will partition the observation space into two disjoined parts, *C* and *U*. When an observation *X* lies in the region *C*, we reject *H0*; when *X* in *U*, we accept *H0*. *C* is called critical region (or rejection region).**

- **So type I error probability (also called significant level) of a test:**

$$\alpha = \Pr(E_1) = \Pr(X \in C \mid H_0)$$

- **Type II error probability of a test:**

$$\beta = \Pr(E_2) = \Pr(X \in U \mid H_1) = 1 - \Pr(X \in C \mid H_1) = 1 - \gamma$$

where $\gamma = \Pr(X \in C \mid H_1)$ **is defined as the *power* of the test.**

- **At the significant level α, *the most powerful test* is defined as the one which maximizes the power *γ* (in turn minimizes Type II error β).**

Statistical Hypothesis Testing(III)

- **A hypothesis can be *simple* or *composite*:**
  - **Simple hypothesis: completely specifies the distribution, e.g.**

$$H_0 : \theta = \theta_0$$

  - **Composite hypothesis: involves a region or interval, e.g.**

$$H_1 : \theta \neq \theta_0 \quad \text{or} \quad H_1 : \theta > \theta_0$$

Statistical Hypothesis Testing(IV)

- ***Neyman Pearson Theorem***:
  - For a simple H_0 and simple H_1, if the distributions under both H_0 and H_1 are known, i.e., $f_0(X|\theta_0)$ and $f_1(X|\theta_1)$. Given any i.i.d. observation data $D=\{X_1,\ldots,X_T\}$, for any significance level α, the most powerful test is formulated as:

If $LR = \frac{\prod_{t=1}^{T} f_0(X_t|\theta_0)}{\prod_{t=1}^{T} f_1(X_t|\theta_1)} > \tau$, accept H_0; otherwise reject H_0.

The threshold τ is adjusted to make the significance of the test to be α. If the both pdf's have the same form, the only difference is parameters, The ratio is also called likelihood ratio (LR).

Statistical Hypothesis Testing(V)

- The Neyman Pearson Theorem provides a method of constructing the most powerful tests for simple hypotheses when the distribution of the observation is known.
- How about if the hypothesis is composite
- Likelihood Ratio Test (LRT): assume the distributions are known except some parameters,

If $T = \frac{\max_{\theta \in H_0} f_{H_0}(X|\theta)}{\max_{\theta \in H_1 \cup H_0} f_{H_1}(X|\theta)} > \tau$, accept H_0; otherwise reject H_0.

  - LRT is not always uniformly most powerful but has some desirable properties.
  - Distribution of T is complicated, $p(T)$; only computable asymptotically.
  - Widely used for many practical applications.

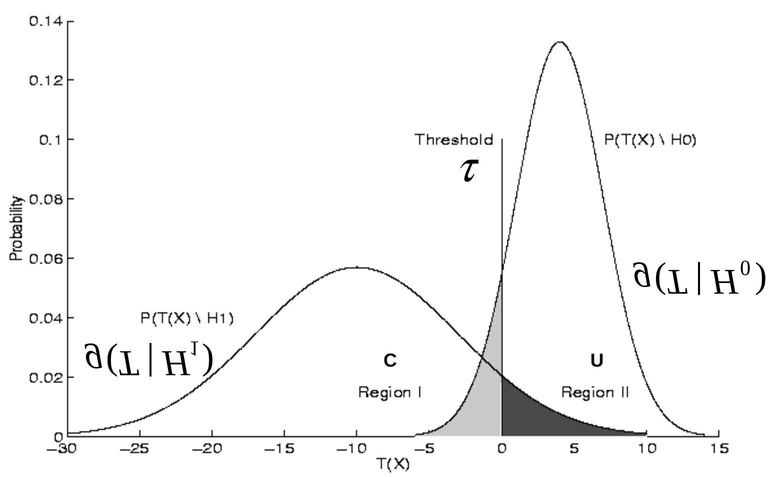
Pattern Verification as Statistical Hypothesis Testing

- **Based on the question to be answered, design two complementary hypotheses,**
  - **The *null* hypothesis H_0: corresponds to YES of the answer.**
  - **The *alternative* hypothesis H_1: corresponds to NO.**
- **The feature distribution under either H_0 or H_1 is unknown.**
- **Training: apply the same idea of data modeling:**
  - **Choose proper statistical model for either H_0 or H_1.**
  - **The model parameters are estimated from some training samples collected from H_0 or H_1.**
- **Decision: use likelihood ratio test (LRT) to make decision**

If $T = \frac{f_0(X \mid \hat{\theta}_o)}{f_1(X \mid \hat{\theta}_1)} > \tau$ **, answer YES; otherwise NO.**

where $f_0(.)$ is the model chosen for H_0, $f_1(.)$ for H_1. $\hat{\theta}_o$ and $\hat{\theta}_1$ **are parameters estimated from data.**

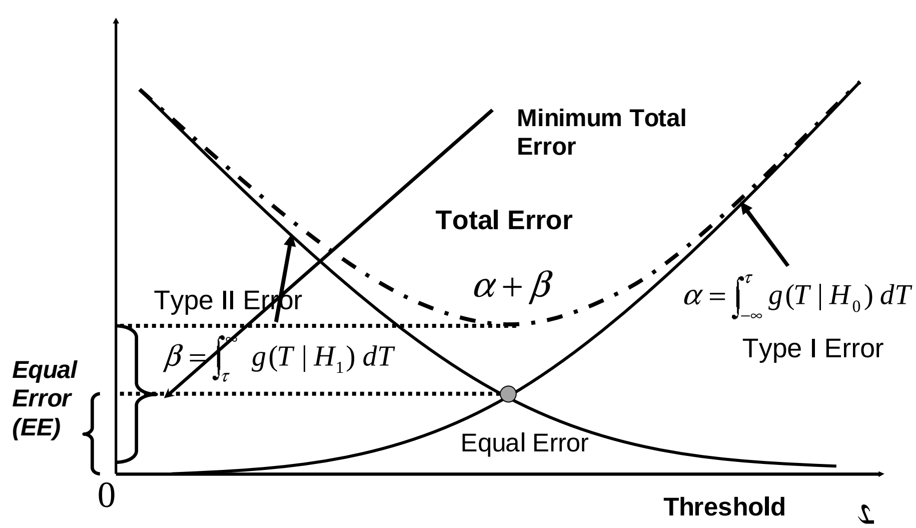
Distributions of LR



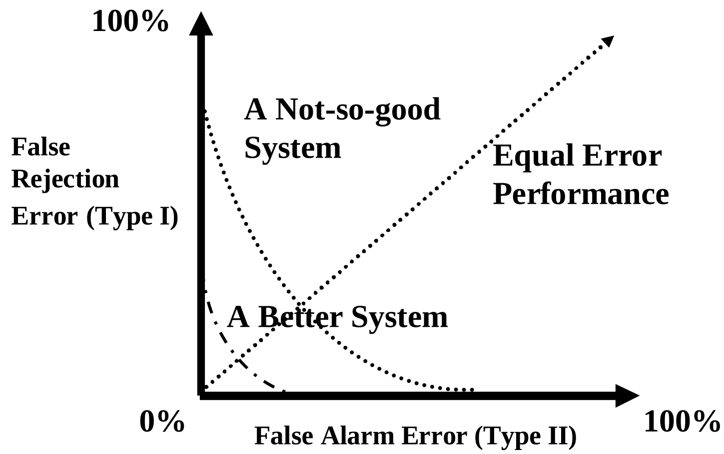
Pattern Verification

- More generally, *T* can be any test statistics from observation data.
  - LRT is a special case for *T.*
- Given a test statistic *T*, we can't minimize both type I error and type II error at the same time.
- Improve verification by choosing different test statistics
  - Distributions of *T*: less overlap → better separation → better verification accuracy (smaller type I and type II errors)
- The key in designing a pattern verification is to find a test statistics *T* and its corresponding parameters so that the overlap between the two distributions is minimized.
- What does it mean by a better verification accuracy?
  - Type I error (false rejection error)
  - Type II error (false alarm error)

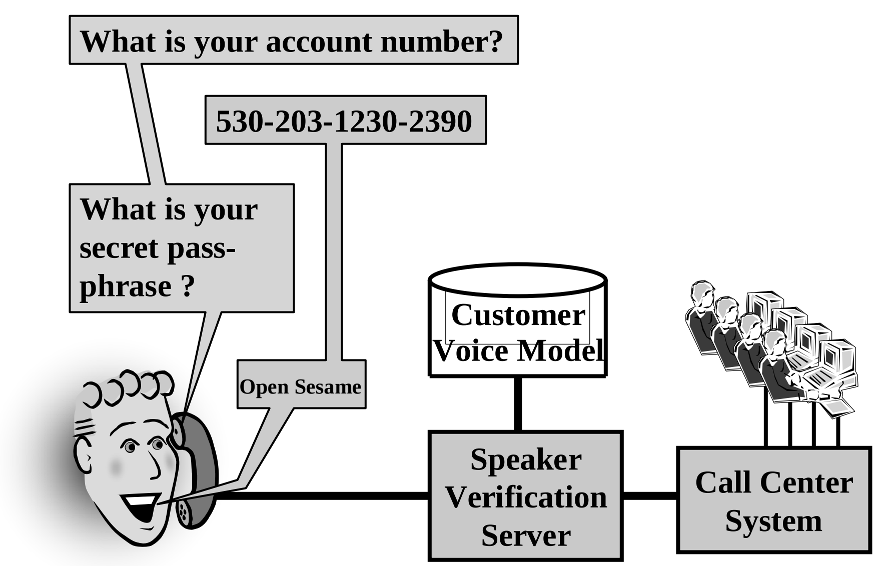
Evaluating Verification (I)



Evaluating Verification (II): ROC curve (Receiver Operating characteristic)



Speaker Verification (SV)



Example(I): Speaker Verification(1)

- **Speaker verification: verify user ID based on the voice. The user first claims a user ID, the system records some voice sample from the user and try to answer YES/NO to the question "Is the person the claimed user or not?".**
- **Speaker verification: if a person claims to be the user A,**
  - **Observation: a segment of voice ➔ feature vectors *X***
  - ***H_0: X is from the claimed user A.***
  - ***H_1: X is NOT from the claimed user A.***
- **Data modeling: commonly use GMM for both H_0 and H_1.**
  - **Mixture number depends on the amount of available data, usually from *16* to *256*.**
  - **For simplicity or estimation reliability, each Gaussian mixand is assumed to be diagonal.**
  - **For each known user *a* registered in the system, we must estimate two GMM's Λ_a and $\overline{\Lambda}_a$ for its H_0 and H_1.**

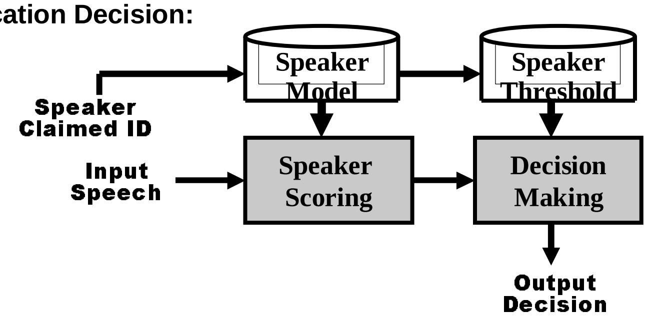
Example(I): Speaker Verification(2)

- **Model estimation:**
  - **For Λ_a in H_0: collect some training samples from the known user and train it based on ML criterion.**

    (how to do ML estimation for GMM?)
  - **How about $\overline{\Lambda}_a$ in H_1 ?**
    - **Anti-speaker model: Train it based on training data collected for all other known users (except *a*). (ML estimation)**
    - **Training it based on training data from some "cohort" speakers who are confusing with the current speaker *a.* (how to choose cohort speaker?)**
    - **For simplicity, use the same background model $\overline{\Lambda}$ for all known users in the system. $\overline{\Lambda}$ is trained based on all users' training data.**

Example(I): Speaker Verification(3)

- **Verification Decision:**



  - **A new user claim id as *A*, based on the recorded voice feature Y:**

If $T = \frac{p(Y \mid H_0)}{p(Y \mid H_1)} = \frac{p(Y \mid \Lambda_A)}{p(Y \mid \overline{\Lambda}_A)} > \tau$ **, accept the user as A; otherwise, reject the user.**

The decision threshold τ is determined empirically in practice.

Example(II): reject outliers in pattern classification

- **How to reject outliers (belonging to none of known classes) in pattern classification ?**
  - **In speech recognition, how to detect unknown words, called out-of vocabulary (OOV) words used by users??**
- **Solution 1: treat outliers as another class → (N+1)-class patterns**
- **Solution 2:**
  - **Stage 1: do N-class pattern classification, find the best match, say class k;**
  - **Stage 2: verify the decision made in stage 1.**
  - **Stage 2 is a pattern verification problem:**
    - **H_0: the pattern *X* really comes from class *k***
    - **H_1: the pattern *X* does NOT come from class *k***

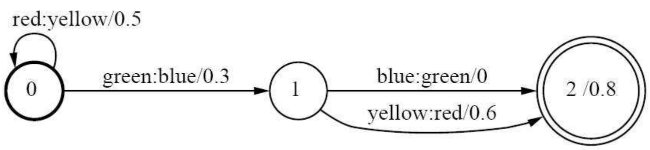
$$\Lambda = \frac{\Pr(X \mid H_0)}{\Pr(X \mid H_1)} > \zeta \text{ accept the decision; otherwise reject}$$

Weighted Finite State Transducer (WFST)

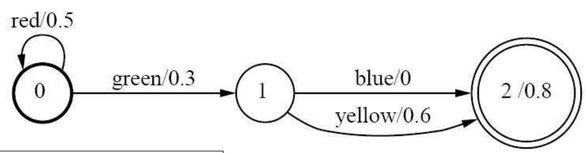
- Efficient algorithms for various operations.
- Weights
  - Handle uncertainty in text, handwritten text, speech, image, biological sequences.
- Applications:
  - Text: pattern-matching, indexation, compression.
  - Speech: speech recognition, speech synthesis.
  - Image: image compression, filters.

Weighted Finite State Transducer (WFST)

- Transducers:

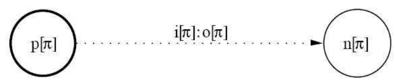


- Automata/Acceptors



WFST Definition (I)

- A path *π*: a sequence of transitions.
  - Original and destination states
  - Input and output labels



- A semiring ≡ a ring without negation
  - Number set K.
  - Sum $\oplus$ and Product $\otimes$.
- Semiring examples:
  - Probability semiring: *R, +, X.*
  - Tropical semiring: *R, min*, +.

WFST Definition (II)

- General Definitions
  - Alphabets: input *Σ*, output *Δ*
  - States: *Q*, initial *I*, final *F*.
  - Transitions: E → Q * (*Σ U ϵ) * (Δ U ϵ) * K * Q*
  - Initial/Final weights: *λ = I → K, ρ = F → K*
- WFST *T = (Σ, Q, I, F, E, λ, ρ):*

$$[T](x, y) = \bigoplus_{\pi \in P(I,x,y,F)} \lambda(p[\pi]) \otimes w[\pi] \otimes \rho(n[\pi])$$

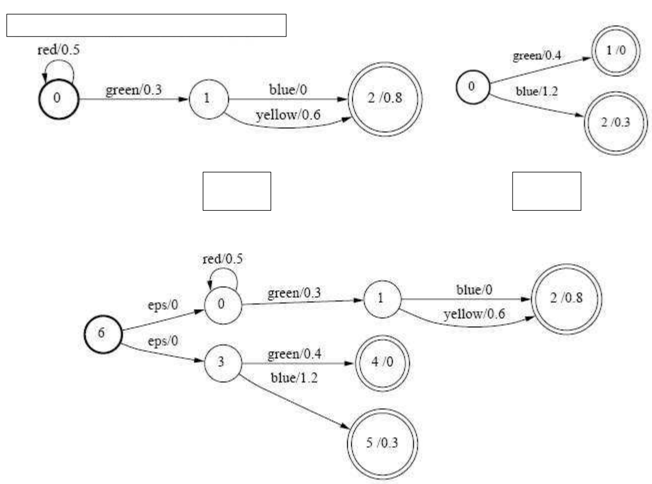
for all $x \in \Sigma^*$ and $y \in \Delta^*$.

WFST Operations

- Sum
- Product
- Closure
- Reversal
- Composition
- Determinization
- Weight pushing
- Minimization

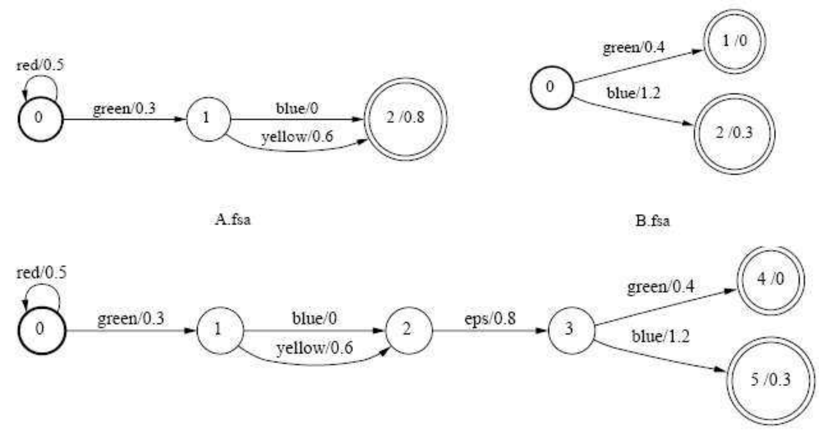
WFST Sum

- **Sum:** $[T_1 \oplus T_2](x,y) = [T_1](x,y) \oplus [T_2](x,y)$



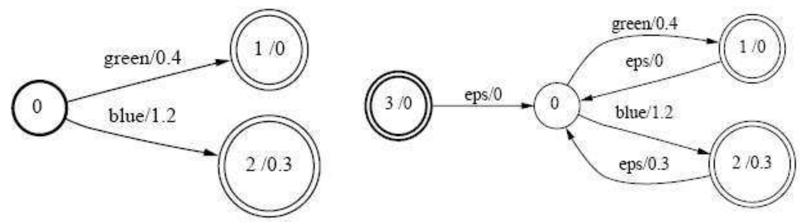
WFST Product

- **Product:** $[T_1 \otimes T_2](x,y) = \bigoplus_{x=x_1x_2,\, y=y_1y_2} [T_1](x_1,y_1) \otimes [T_2](x_2,y_2)$



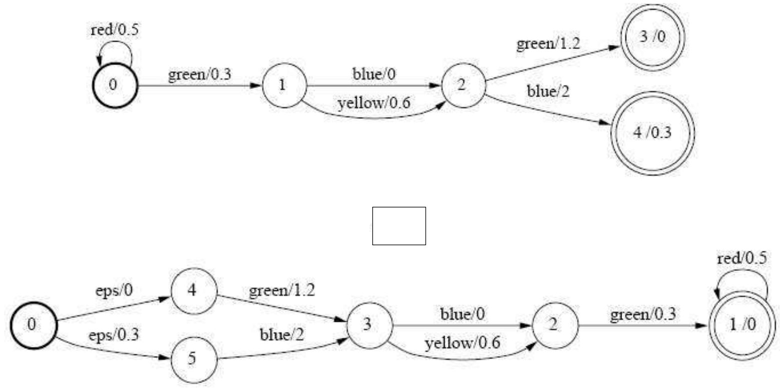
WFST Closure

- **Closure:** $[T^*](x,y) = \bigoplus_{n=0}^{\infty} [T]^n (x,y)$



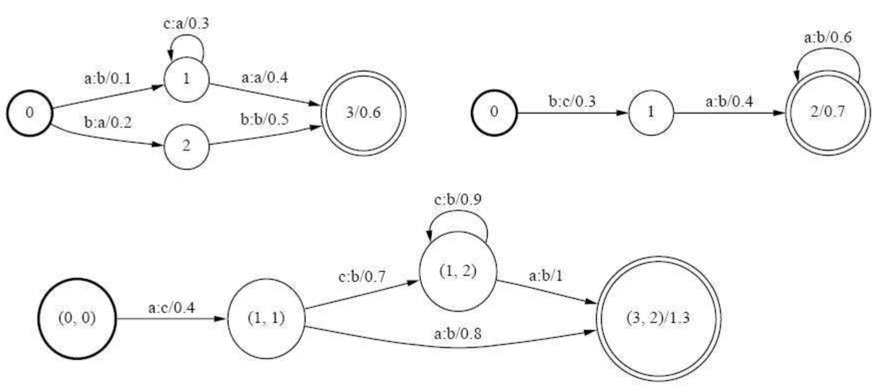
WFST Reversal

- **Reversal:** $\left[\tilde{T}\right](x,y)=\left[T\right](\tilde{x},\tilde{y})$



WFST Composition

- **Composition:** $\left[T_1 \circ T_2\right](x,y)=\bigoplus_z \left[T_1\right](x,z)\otimes\left[T_2\right](z,y)$



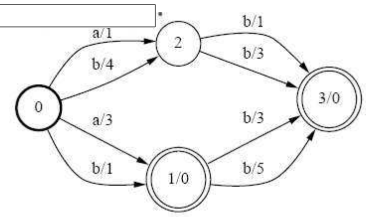
WFST Composition Algorithm

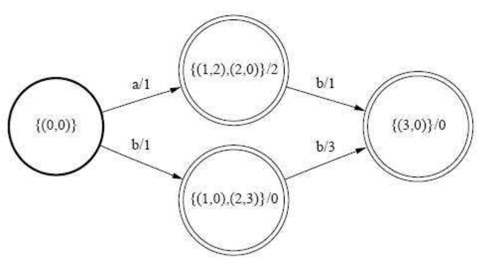
WEIGHTED-COMPOSITION(T_1, T_2)

$Q \leftarrow I_1 \times I_2$
$S \leftarrow I_1 \times I_2$
while $S \neq \emptyset$ **do**
    $(q_1, q_2) \leftarrow$ HEAD(S)
    DEQUEUE(S)
    if $(q_1, q_2) \in I_1 \times I_2$ **then**
        $I \leftarrow I \cup \{(q_1, q_2)\}$
        $\lambda(q_1, q_2) \leftarrow \lambda_1(q_1) \otimes \lambda_2(q_2)$
    if $(q_1, q_2) \in F_1 \times F_2$ **then**
        $F \leftarrow F \cup \{(q_1, q_2)\}$
        $\rho(q_1, q_2) \leftarrow \rho_1(q_1) \otimes \rho_2(q_2)$
    for each $(e_1, e_2) \in E[q_1] \times E[q_2]$ such that $o[e_1] = i[e_2]$ **do**
        if $(n[e_1], n[e_2]) \notin Q$ **then**
            $Q \leftarrow Q \cup \{(n[e_1], n[e_2])\}$
            ENQUEUE$(S, (n[e_1], n[e_2]))$
        $E \leftarrow E \cup \{((q_1, q_2), i[e_1], o[e_2], w[e_1] \otimes w[e_2], (n[e_1], n[e_2]))\}$
return T

WFST Determinization

- Deterministic WFST: no common input label for all outgoing transitions from any state.
- Determinimization: determinizable WFST → deterministic W.





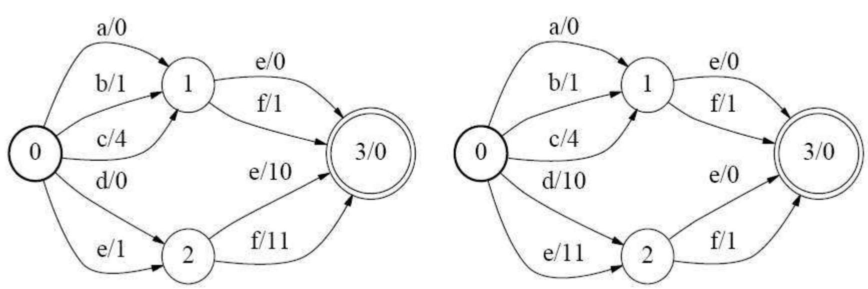
WFST Determinization Algorithm

WEIGHTED-DETERMINIZATION(A)

$i' \leftarrow \{(i, \lambda(i)) : i \in I\}$
$\lambda'(i') \leftarrow \overline{1}$
$S \leftarrow \{i'\}$
while $S \neq \emptyset$ **do**
    $p' \leftarrow$ HEAD(S)
    DEQUEUE(S)
    for each $x \in i[E[Q[p']]]$ **do**
        $w' \leftarrow \bigoplus \{v \otimes w : (p, v) \in p', (p, x, w, q) \in E\}$
        $q' \leftarrow \{(q, \bigoplus \{w'^{-1} \otimes (v \otimes w) : (p, v) \in p', (p, x, w, q) \in E\}) :$
            $q = n[e], i[e] = x, e \in E[Q[p']]\}$
        $E' \leftarrow E' \cup \{(p', x, w', q')\}$
        if $q' \notin Q'$ **then**
            $Q' \leftarrow Q' \cup \{q'\}$
            if $Q[q'] \cap F \neq \emptyset$ **then**
                $F' \leftarrow F' \cup \{q'\}$
                $\rho'(q') \leftarrow \bigoplus \{v \otimes \rho(q) : (q, v) \in q', q \in F\}$
            ENQUEUE(S, q')
return T'

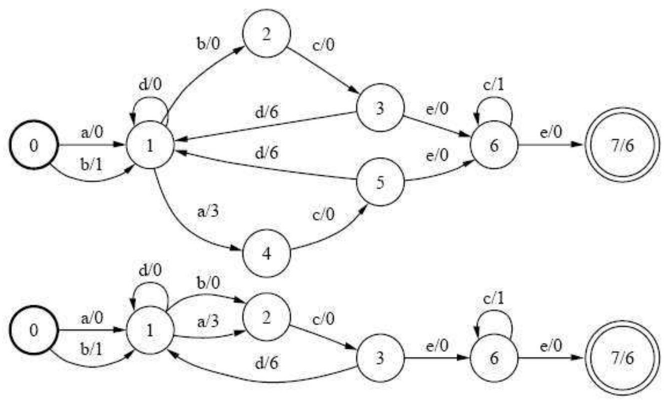
WFST Weights Pushing

- **Weight pushing: re-distribute all weights along paths.**



WFST Minimization

- Minimize number of states and transitions of a deterministic WFST.



WFST Applications

- String search/match
- String conversion/ language normalization
- Representing Language models and probabilistic grammar
- Sentence generation

Example I: keyword detection

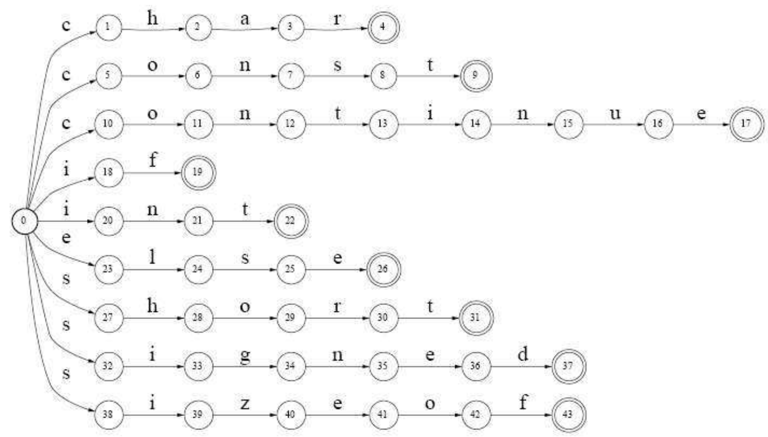
- **C identifiers:**
  $\{char, const, continue, if, int, else, short, signed, sizeof\}$
- **Brute-force search:**

```
if(strcmp(token, "char") == 0) return 1;
if(strcmp(token, "const") == 0) return 1;
if(strcmp(token, "oontinue") == 0) return 1;
if(strcmp(token, "if") == 0) return 1;
if(strcmp(token, "int") == 0) return 1;
if(strcmp(token, "else") == 0) return 1;
if(strcmp(token, "short") == 0) return 1;
if(strcmp(token, "signed") == 0) return 1;
if(strcmp(token, "sizeof") == 0) return 1;
else return 0;
```

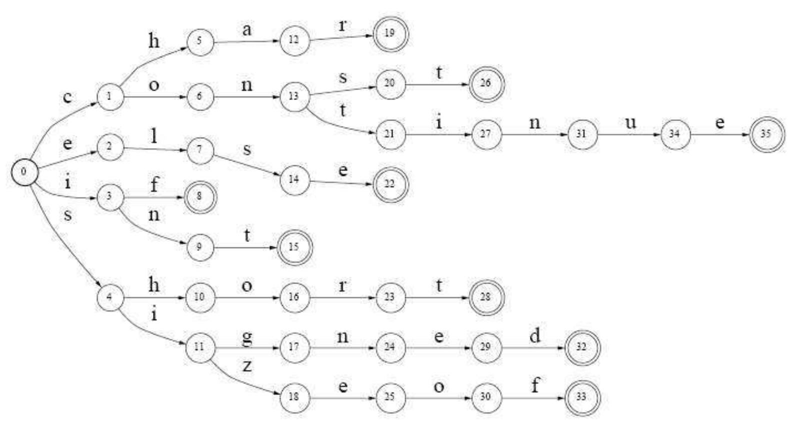
Example I: keyword detection: tabular search

```
#define NKEYS 9
char *keywrds[NKEYS] = {   "char",
                            "const",
                            "continue",
                            "else",
                            "if",
                            "int",
                            "short",
                            "signed",
                            "sizeof" };

int keycmp(const void *x, const void *y)
{ return strcmp(x, *(char **)y);}

bsearch(token, keywrds, NKEYS, sizeof(char *), keycmp);
```

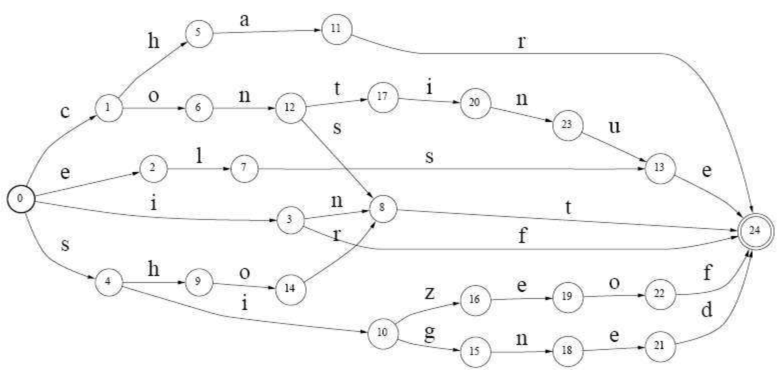
Example I: keyword detection: Automata Search



Example I: keyword detection: Deterministic Search



Example I: keyword detection: Minimal Deterministic Search



Example II: Context-dependent Phones

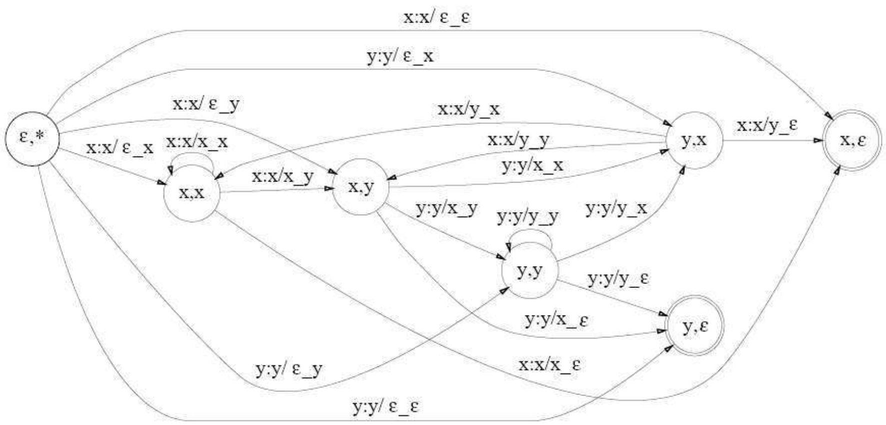
- *Monophone vs. Triphone*
- *Sentence: How do they turn out later ?*
- *Monophones: h aw d uh dh eh t er n aw t l ai t er*
- Triphones:

  <s>-h+aw h-aw+d aw-d+uh d-uh+dh uh-dh+eh …

- WFST: mapping context-independent monophones to context-dependent triphones

Example II: Context-dependent Phones

- A simple example with only two symbols x,y:



Example III: Representing Language model

- Representing language models as WFST
- Representing HMMs as WFST
- Representing overall grammar as WFST
- Come back later …