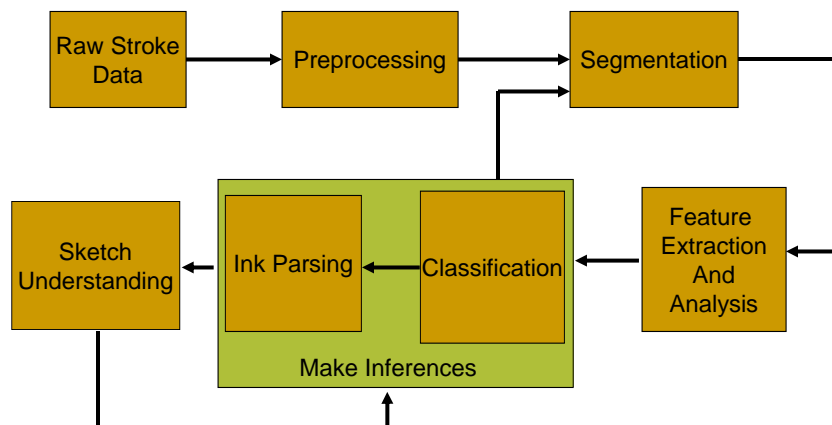


Symbol Recognition in Sketch-Based Interfaces

Lecture #8: Symbol Recognition
Joseph J. LaViola Jr.
Fall 2012

Recall Pen-Based Interface Dataflow



Symbol Recognition

- Want to recognize handwritten symbols
 - characters
 - shapes
 - gestures
- Use machine learning approach
- Which algorithm?
 - depends on number of symbols in alphabet
 - complexity (i.e., similarity of symbols)
 - distribution assumptions

Recognition Algorithms

- Many different approaches
- Machine learning techniques (classification)
 - linear classifiers
 - k-means classifiers
 - neural networks
 - Hidden Markov Models
 - template matching
 - support vector machines
 - AdaBoost
- Curve matching
 - elastic matching
- Primitive decomposition

Rubine's Gesture Recognition Algorithm (Rubine 1991)

- Simple linear classifier
- Utilizes rejection metrics
- Assumes normality for features
- Simple to implement
- Does not need a lot of training samples

Recall Rubine's Feature Set

- Cosine and sine of initial angle
- Length and angle of bounding box diagonal
- Distance between first and last point
- Cosine and sine of angle between first and last point
- Total gesture length
- Total angle traversed
- Sum of absolute value of the angle at each point
- Sum of squared values of the angle at each point
- Maximum speed
- Stroke duration

Rubine Classifier

$$v_{\hat{c}} = w_{\hat{c}0} + \sum_{i=1}^F w_{\hat{c}i} f_i \quad 0 \leq c < C$$

where F is the number of features,
 $w_{\hat{c}}$ is the weights, and the classification
of symbol g is the c that maximizes $v_{\hat{c}}$

- Evaluate each gesture $0 \leq c < C$.
- $v_{\hat{c}}$ = value = goodness of fit for that gesture c .

Rubine Classifier Training

- Collect E samples for each symbol class
- Calculate feature vector for each sample for each class
 - $f_{\hat{c}ei}$ = the feature value of the i^{th} feature for the e^{th} sample of the c^{th} symbol
- For each symbol calculate the mean value for each feature

$$\bar{f}_{\hat{c}i} = \frac{1}{E_{\hat{c}}} \sum_{e=0}^{E_{\hat{c}}-1} f_{\hat{c}ei} \quad \text{where } 0 \leq e < E_{\hat{c}}$$

and $E_{\hat{c}}$ is the number of training samples per class

Rubine Classifier – Computing Weights

- We first need the covariance matrix of each class c

$$\Sigma_{\hat{c}ij} = \frac{1}{E_{\hat{c}} - 1} \sum_{e=0}^{E_{\hat{c}}-1} (f_{\hat{c}ei} - \overline{f_{\hat{c}i}})(f_{\hat{c}ej} - \overline{f_{\hat{c}j}})$$

Rubine Classifier – Computing Weights (2)

- Using the covariance matrices from each class, find the common covariance matrix
 - numerator = non-normalize total covariance
 - denominator = normalization factor = total number of examples – total number of shapes

$$\Sigma_{ij} = \frac{\sum_{c=0}^{C-1} \Sigma_{\hat{c}ij}}{-C + \sum_{c=0}^{C-1} E_{\hat{c}}}$$

Rubine Classifier – Computing Weights (3)

- Using the common covariance matrix and the mean feature vectors from each class, we can compute the weights

$$w_{\hat{c}j} = \sum_{i=1}^F (\Sigma^{-1})_{ij} \overline{f_{\hat{c}i}}, \quad 1 \leq j \leq F$$

$$w_{\hat{c}0} = -\frac{1}{2} \sum_{i=1}^F w_{\hat{c}i} \overline{f_{\hat{c}i}}$$

Rubine Classifier – Rejection Measures

- Linear classifier always will classify a symbol as one of the C classes
 - want to try to reject outliers and ambiguous symbols
 - two approaches
 - probabilistic
 - distance measure

Rubine Classifier – Probabilistic Rejection Measure

- Given a symbol g with feature vector \mathbf{f} classified as class i ($v_i > v_j, \forall j \neq i$)

$$\tilde{P}(i | g) = \frac{1}{\sum_{j=0}^{c-1} e^{(v_j - v_i)}}$$

Reject symbols with $\tilde{P}(i | g) < 0.95$

Rubine Classifier – Rejection based on Distance

- Mahalanobis distance – the number of standard deviations a symbol g is away from the mean of its chosen class i

$$\delta^2 = \sum_{j=1}^F \sum_{k=1}^F (\Sigma^{-1})_{jk} (f_j - \bar{f}_{ij})(f_k - \bar{f}_{ik})$$

Rejecting symbols for which $\delta^2 > \frac{1}{2} F^2$

- May need to be careful not to reject too many good symbols (a simple alternate list to correct mistakes will be helpful)

AdaBoost (Schapire 1997)

- Not really a classification algorithm – more like a framework
- Can use many different classification algorithms within AdaBoost framework
- Works with series of weak (base) classifiers
 - Want to increase the importance of incorrectly classified examples
 - series of weak hypotheses and weights form a strong hypothesis
 - need to ensure weak learners output either 1 or -1
- Many different variants (M1, M2, etc...)

AdaBoost Algorithm

Given $(x_1, y_1), \dots, (x_m, y_m)$ where $x_i \in X, y_i \in Y = \{-1, +1\}$

Initialize $D_1(i) = 1/m$

For $t = 1 \dots T$

- Train weak learner using distribution on D_t
- Get weak hypothesis $h_t: X \rightarrow \{-1, +1\}$ with error

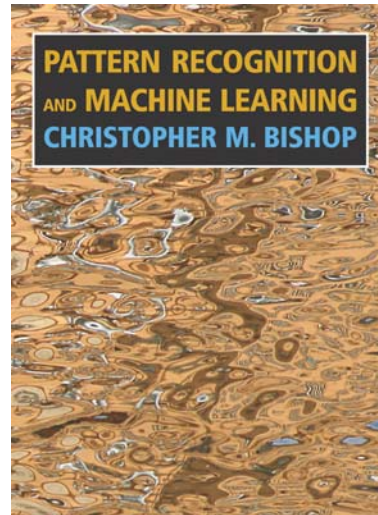
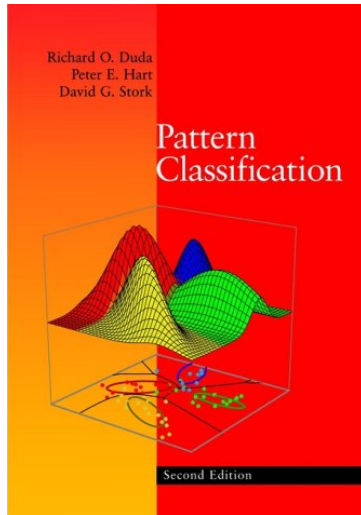
$$\varepsilon_t = \Pr_{i \sim D_t}[h_t(x_i) \neq y_i] = \sum_{i: h_t(x_i) \neq y_i} D_t(i)$$

- Compute $\alpha_t = \frac{1}{2} \ln \left(\frac{1 - \varepsilon_t}{\varepsilon_t} \right)$

- Update $D_{t+1}(i) = \frac{D_t(i) e^{-\alpha_t y_i h_t(x_i)}}{Z_t}$

Final hypothesis is $H(x) = \text{sign} \left(\sum_{t=1}^T \alpha_t h_t(x) \right)$

More Information on Machine Learning



Fall 2012

CAP 6105 – Pen-Based User Interfaces

©Joseph J. LaViola Jr.

Next Class

- Features, Features and more Features.

Fall 2011

CAP 6105 – Pen-Based User Interfaces

©Joseph J. LaViola Jr.