Lecture 15

Expectation Maximization (EM), Unsupervised Learning and Clustering

Tuesday, October 19, 1999

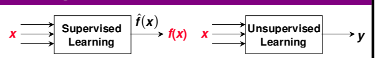
William H. Hsu
Department of Computing and Information Sciences, KSU
http://www.cis.ksu.edu/~bhsu

Readings:
Section 6.12, Mitchell
Section 3.2.4, Shavlik and Dietterich (Rumelhart and Zipser)
Section 3.2.5, Shavlik and Dietterich (Kohonen)

Lecture Outline

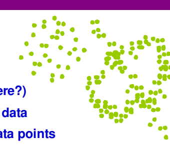
- Readings: 6.12, Mitchell; Rumelhart and Zipser
- Suggested Reading: Kohonen
- This Week's Review: "The Future of Time Series", Gershenfeld and Weigend
- Unsupervised Learning and Clustering
  - Definitions and framework
  - Constructive induction
    - Feature construction
    - Cluster definition
  - EM, *AutoClass*, Principal Components Analysis, Self-Organizing Maps
- Expectation-Maximization (EM) Algorithm
  - More on EM and Bayesian Learning
  - EM and unsupervised learning
- Next Lecture: Time Series Learning
  - Intro to time series learning, characterization; stochastic processes
  - Read Chapter 16, Russell and Norvig (decisions and utility)

Unsupervised Learning: Objectives



- Unsupervised Learning
  - Given: data set *D*
    - Vectors of attribute values (x_1, x_2, …, x_n)
    - No distinction between input attributes and output attributes (class label)
  - Return: (synthetic) descriptor *y* of each *x*
    - Clustering: *grouping points* (*x*) into inherent regions of mutual similarity
    - Vector quantization: *discretizing continuous space* with best labels
    - Dimensionality reduction: *projecting many attributes* down to a few
    - Feature extraction: *constructing (few) new attributes* from (many) old ones
- Intuitive Idea
  - Want to map independent variables (*x*) to dependent variables (*y* = *f*(*x*))
  - *Don't always know what "dependent variables" (y) are*
  - Need to discover *y* based on numerical criterion (e.g., distance metric)

Clustering



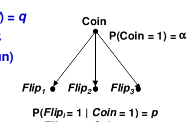
- A Mode of Unsupervised Learning
  - Given: a collection of data points
  - Goal: *discover structure* in the data
    - *Organize data into sensible groups* (how many here?)
    - Criteria: convenient and valid organization of the data
    - NB: not necessarily rules for classifying future data points
  - Cluster analysis: study of algorithms, methods for discovering this structure
    - *Representing* structure: organizing data into clusters (cluster formation)
    - *Describing* structure: cluster boundaries, centers (cluster segmentation)
    - *Defining* structure: assigning meaningful names to clusters (cluster labeling)
- Cluster: Informal and Formal Definitions
  - Set whose entities are alike and are different from entities in *other* clusters
  - Aggregation of points in the instance space such that distance between any two points in the cluster is less than the distance between any point in the cluster and any point not in it

Quick Review: Bayesian Learning and EM

- Problem Definition
  - Given: data (*n*-tuples) with missing values, *aka* partially observable (PO) data
  - Want to fill in *?* with expected value
- Solution Approaches
  - Expected = distribution over possible values
  - Use "best guess" Bayesian model (e.g., BBN) to estimate distribution
  - Expectation-Maximization (EM) algorithm can be used here
- Intuitive Idea
  - Want to find h_{ML} in PO case (*D* ≡ *unobserved variables* ° *observed variables*)
  - Estimation step: calculate *E*[*unobserved variables* | *h*], assuming current *h*
  - Maximization step: update w_{ijk} to maximize *E*[lg *P*(*D* | *h*)], *D* ≡ *all variables*

$$h_{ML} = \arg\max_{h \in H} \frac{\#\,\text{data cases with } \vec{n}, \vec{e}}{\#\,\text{data cases with } \vec{e}} = \arg\max_{h \in H} \frac{\sum_j I_{\vec{N}=\vec{n}, \vec{E}=\vec{e}}\left(\vec{X}_j\right)}{\sum_j I_{\vec{E}=\vec{e}}\left(\vec{X}_j\right)}$$

EM Algorithm: Example [1]



- Experiment
  - Two coins: *P*(*Head on Coin 1*) = *p*, *P*(*Head on Coin 2*) = *q*
    - Experimenter first selects a coin: *P*(*Coin* = 1) = α
    - Chosen coin tossed 3 times (per experimental run)
  - Observe: *D* = {(1 H H T), (1 H T T), (2 T H T)}
  - Want to predict: α, *p*, *q*
  - *How to model the problem?*
    - Simple Bayesian network
    - Now, can find most likely values of parameters α, *p*, *q* given data *D*
- Parameter Estimation
  - Fully observable case: easy to estimate *p*, *q*, and α
    - Suppose *k* heads are observed out of *n* coin flips
    - Maximum likelihood estimate v_{ML} for $Flip_i$: *p* = *k*/*n*
  - Partially observable case
    - Don't know which coin the experimenter chose
    - Observe: *D* = {(H H T), (H T T), (T H T)} ≡ {(? H H T), (? H T T), (? T H T)}

EM Algorithm: Example [2]

- **Problem**
  - When we knew *Coin* = 1 or *Coin* = 2, there was no problem
  - *No known analytical solution* to the partially observable problem
    - i.e., not known how to compute estimates of p, q, and α to get v_{ML}
    - Moreover, not known what the computational complexity is
- **Solution Approach: Iterative Parameter Estimation**
  - Given: a *guess* of $P(Coin = 1 \mid x)$, $P(Coin = 2 \mid x)$
  - Generate "fictional data points", weighted according to this probability
    - $P(Coin = 1 \mid x) = P(x \mid Coin = 1)\ P(Coin = 1) / P(x)$ based on our guess of α, p, q
    - Expectation step (the "E" in EM)
  - Now, can find most likely values of parameters α, p, q given "fictional" data
    - Use gradient descent to update our guess of α, p, q
    - Maximization step (the "M" in EM)
  - Repeat until termination condition met (e.g., stopping criterion on validation set)
- **EM Converges to Local Maxima of the Likelihood Function $P(D \mid \Theta)$**

EM Algorithm: Example [3]

- **Expectation Step**
  - Suppose we observed m actual experiments, each n coin flips long
    - Each experiment corresponds to one choice of coin ($\sim\alpha$)
    - Let h denote the number of heads in experiment x_i (a single data point)
  - Q: How did we simulate the "fictional" data points, $E[\sum \log P(x \mid \hat{\alpha}, \hat{p}, \hat{q})]$?
  - A: By estimating (for $1 \le i \le m$, i.e., the real data points)

$$P(Coin = 1 / \vec{x}_i) = \frac{P(\vec{x}_i / Coin = 1) \cdot P(Coin = 1)}{P(\vec{x}_i)}$$

$$\approx \frac{\hat{\alpha} \cdot \hat{p}^h (1 - \hat{p})^{n-h}}{\hat{\alpha} \cdot \hat{p}^h (1 - \hat{p})^{n-h} + (1 - \hat{\alpha}) \cdot \hat{q}^h (1 - \hat{q})^{n-h}}$$

- **Maximization Step**
  - Q: What are we updating? What objective function are we maximizing?
  - A: We are updating $\hat{\alpha}, \hat{p}, \hat{q}$ to maximize $\frac{\partial E}{\partial \hat{\alpha}}, \frac{\partial E}{\partial \hat{p}}, \frac{\partial E}{\partial \hat{q}}$ where $E = \mathrm{E}\left[\sum_{i=1}^{m} \log P(\vec{x}_i / \hat{\alpha}, \hat{p}, \hat{q})\right]$

$$\hat{\alpha} = \frac{\sum P(Coin = 1 / \vec{x}_i)}{m}, \hat{p} = \frac{\sum \frac{h_i}{n} P(Coin = 1 / \vec{x}_i)}{\sum P(Coin = 1 / \vec{x}_i)}, \hat{q} = \frac{\sum \frac{h_i}{n} [1 - P(Coin = 1 / \vec{x}_i)]}{\sum [1 - P(Coin = 1 / \vec{x}_i)]}$$

EM for Unsupervised Learning

- **Unsupervised Learning Problem**
  - Objective: estimate a probability distribution with unobserved variables
  - Use EM to estimate mixture policy (more on this later; see 6.12, Mitchell)
- **Pattern Recognition Examples**
  - Human-computer intelligent interaction (HCII)
    - Detecting facial features in emotion recognition
    - Gesture recognition in virtual environments
  - Computational medicine [Frey, 1998]
    - Determining morphology (shapes) of bacteria, viruses in microscopy
    - Identifying cell structures (e.g., nucleus) and shapes in microscopy
  - Other image processing
  - Many other examples (audio, speech, signal processing; motor control; etc.)
- **Inference Examples**
  - Plan recognition: mapping from (observed) actions to agent's (hidden) plans
  - Hidden changes in context: e.g., aviation; computer security; MUDs

Unsupervised Learning: *AutoClass* [1]

- **Bayesian Unsupervised Learning**
  - Given: $D = \{(x_1, x_2, \ldots, x_n)\}$ (vectors of indistingushed attribute values)
  - Return: set of class labels that has maximum *a posteriori* (MAP) probability
- **Intuitive Idea**
  - Bayesian learning: $h_{MAP} = \arg\max_{h \in H} P(h \mid D) = \arg\max_{h \in H} P(D \mid h) P(h)$
  - MDL/BIC (Occam's Razor): priors $P(h)$ express "cost of coding" each model h
  - *AutoClass*
    - Define mutually exclusive, exhaustive clusters (class labels) $y_1, y_2, \ldots, y_J$
    - Suppose: each y_j ($1 \le j \le J$) contributes to x
    - Suppose also: y_j's contribution ~ *known pdf*, e.g., Mixture of Gaussians (MoG)
    - Conjugate priors: priors on y of same form as priors on x
- **When to Use for Clustering**
  - Believe (or can *assume*): clusters generated by known pdf
  - Believe (or can *assume*): clusters combined using finite mixture (later)

Unsupervised Learning: *AutoClass* [2]

- ***AutoClass* Algorithm [Cheeseman *et al*, 1988]**
  - Based on maximizing $P(x \mid \Theta_j, y_j, J)$
    - Θ_j: class (cluster) parameters (e.g., mean and variance)
    - y_j: synthetic classes (can estimate marginal $P(y_j)$ any time)
  - Apply Bayes's Theorem, use numerical BOC estimation techniques (cf. Gibbs)
  - Search objectives
    - Find best J (*ideally*: integrate out Θ_j, y_j; *really*: start with big J, decrease)
    - Find Θ_j, y_j: use MAP estimation, then "integrate in the neighborhood" of y_{MAP}
- **EM: Find MAP Estimate for $P(x \mid \Theta_j, y_j, J)$ by Iterative Refinement**
- **Advantages over Symbolic (Non-Numerical) Methods**
  - Returns probability distribution over class membership
    - More robust than "best" y_j
    - Compare: fuzzy set membership (similar but probabilistically motivated)
  - Can deal with continuous as well as discrete data

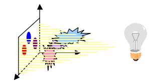
Unsupervised Learning: *AutoClass* [3]

- **AutoClass Resources**
  - Beginning tutorial (*AutoClass II*): Cheeseman *et al*, 4.2.2 Buchanan and Wilkins
  - Project page: http://ic-www.arc.nasa.gov/ic/projects/bayes-group/autoclass/
- **Applications**
  - Knowledge discovery in databases (KDD) and data mining
    - Infrared astronomical satellite (IRAS): *spectral atlas* (*sky survey*)
    - Molecular biology: pre-clustering DNA acceptor, donor sites (mouse, human)
    - *LandSat* data from Kansas (30 km^2 region, 1024 x 1024 pixels, 7 channels)
    - Positive findings: see book chapter by Cheeseman and Stutz, online
  - Other typical applications: see KD Nuggets (http://www.kdnuggets.com)
- **Implementations**
  - Obtaining source code from project page
    - *AutoClass III*: Lisp implementation [Cheeseman, Stutz, Taylor, 1992]
    - *AutoClass C*: C implementation [Cheeseman, Stutz, Taylor, 1998]
  - These and others at: http://www.recursive-partitioning.com/cluster.html

Unsupervised Learning: Competitive Learning for Feature Discovery

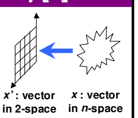
- Intuitive Idea: Competitive Mechanisms for Unsupervised Learning
  - Global organization from local, competitive weight update
    - Basic principle expressed by Von der Malsburg
    - Guiding examples from (neuro)biology: lateral inhibition
  - Previous work: Hebb, 1949; Rosenblatt, 1959; Von der Malsburg, 1973; Fukushima, 1975; Grossberg, 1976; Kohonen, 1982
- A Procedural Framework for Unsupervised Connectionist Learning
  - Start with identical ("neural") processing units, with random initial parameters
  - Set limit on "activation strength" of each unit
  - Allow units to compete for right to respond to a set of inputs
- Feature Discovery
  - Identifying (or *constructing*) new features relevant to supervised learning
  - Examples: finding distinguishable letter characteristics in handwriten character recognition (HCR), optical character recognition (OCR)
  - Competitive learning: transform X into X'; train units in X' closest to x

Unsupervised Learning: Kohonen's Self-Organizing Map (SOM) [1]



- Another Clustering Algorithm
  - *aka* Self-Organizing Feature Map (SOFM)
  - Given: vectors of attribute values (x_1, x_2, …, x_n)
  - Returns: vectors of attribute values (x_1', x_2', …, x_k')
    - Typically, $n >> k$ (n is high, k = 1, 2, or 3; hence "dimensionality reducing")
    - Output: vectors x', the projections of input points x; also get $P(x_j' \mid x_i)$
    - Mapping from x to x' is topology preserving
- Topology Preserving Networks
  - Intuitive idea: similar input vectors will map to similar clusters
  - Recall: informal definition of cluster (isolated set of mutually similar entities)
  - Restatement: "clusters of X (high-D) will still be clusters of X' (low-D)"
- Representation of Node Clusters
  - Group of neighboring artificial neural network units (neighborhood of nodes)
  - SOMs: combine ideas of topology-preserving networks, unsupervised learning
- Implementation: http://www.cis.hut.fi/nnrc/ and *MATLAB* NN Toolkit

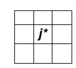
Unsupervised Learning: Kohonen's Self-Organizing Map (SOM) [2]



- Kohonen Network (SOM) for Clustering
  - Training algorithm: unnormalized competitive learning
  - Map is organized as a grid (shown here in 2D)
    - Each node (grid element) has a weight vector w_j
    - Dimension of w_j is n (same as input vector)
    - Number of trainable parameters (weights): $m \cdot m \cdot n$ for an m-by-m SOM
    - 1999 state-of-the-art: typical small SOMs 5-20, "industrial strength" > 20
  - Output found by selecting j^* whose w_j has minimum Euclidean distance from x
    - *Only one active node, aka* Winner-Take-All (WTA): winning node j^*
    - i.e., $j^* = \arg\min_j \| w_j - x \|_2$
- Update Rule
  - Same as competitive learning algorithm, with one modification
  - Neighborhood function associated with j^* spreads the w_j around

$$\vec{w}_j(t+1) = \begin{cases} \vec{w}_j(t) + r(t)\, h_{j,j^*}\left(\vec{x} - \vec{w}_j(t)\right) & \text{if } j \in \textit{Neighborhood}(j^*) \\ \vec{w}_j(t) & \text{otherwise} \end{cases}$$

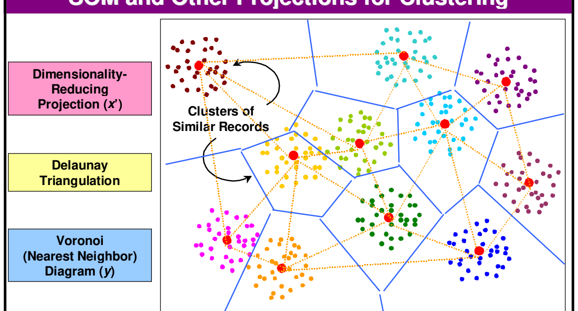
Unsupervised Learning: Kohonen's Self-Organizing Map (SOM) [3]



Neighborhood of 1

- Traditional Competitive Learning
  - Only train j^*
  - Corresponds to neighborhood of 0
- Neighborhood Function h_{j,j^*}
  - For 2D Kohonen SOMs, h is typically a square or hexagonal region
    - j^*, the winner, is at the center of *Neighborhood* (j^*)
    - $h_{j^*,j^*} \equiv 1$
  - Nodes in *Neighborhood* (j) updated whenever j wins, i.e., $j^* = j$
  - Strength of information fed back to w_j is inversely proportional to its distance from the j^* for each x
  - Often use exponential or Gaussian (normal) distribution on neighborhood to decay weight delta as distance from j^* increases
- Annealing of Training Parameters
  - Neighborhood must shrink to 0 to achieve convergence
  - r (learning rate) must also decrease monotonically

Unsupervised Learning: SOM and Other Projections for Clustering



Cluster Formation and Segmentation Algorithm (Sketch)

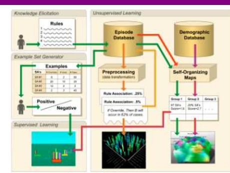
Unsupervised Learning: Other Algorithms (PCA, Factor Analysis)

- Intuitive Idea
  - Q: *Why are dimensionality-reducing transforms good for supervised learning?*
  - A: There may be many attributes with undesirable properties, e.g.,
    - Irrelevance: x_i has little discriminatory power over $c(x) = y_i$
    - Sparseness of information: "feature of interest" spread out over many x_i's (e.g., text document categorization, where x_i is a word position)
    - We want to increase the "information density" by "squeezing X down"
- Principal Components Analysis (PCA)
  - Combining redundant variables into a single variable (*aka* component, or factor)
  - Example: ratings (e.g., Nielsen) and polls (e.g., Gallup); responses to certain questions may be correlated (e.g., "like fishing?" "time spent boating")
- Factor Analysis (FA)
  - General term for a class of algorithms that includes PCA
  - Tutorial: http://www.statsoft.com/textbook/stfacan.html

Clustering Methods: Design Choices

- **Intuition**
  - Functional (declarative) definition: easy ("We recognize a cluster when we see it")
  - Operational (procedural, constructive) definition: much harder to give
  - Possible reason: clustering of objects into groups has taxonomic semantics (e.g., shape, size, time, resolution, etc.)
- **Possible Assumptions**
  - Data generated by a particular probabilistic model
  - No statistical assumptions
- **Design Choices**
  - Distance (similarity) measure: standard metrics, transformation-invariant metrics
    - L_1 (Manhattan): $\sum |x_i - y_i|$, L_2 (Euclidean): $\sqrt{\sum (x_i - y_i)^2}$, L_∞ (Sup): $\max |x_i - y_i|$
    - Symmetry: Mahalanobis distance
    - Shift, scale invariance: covariance matrix
  - Transformations (e.g., covariance diagonalization: rotate axes to get rotational invariance, cf. PCA, FA)

Clustering: Applications



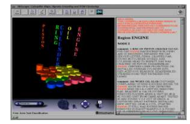
NCSA *D2K* 1.0 - http://www.ncsa.uiuc.edu/STI/ALG/

Transactional Database Mining

Data from T. Mitchell's web site:
http://www.cs.cmu.edu/~tom/faces.html

http://www.cnl.salk.edu/~wiskott/Bibliographies/FaceFeatureFinding.html

Facial Feature Extraction



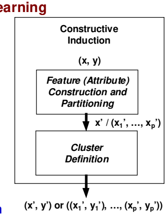
Confidential and proprietary to Caterpillar; may only be used with prior written consent from Caterpillar.

Information Retrieval: Text Document Categorization

ThemeScapes - http://www.cartia.com   NCSA *D2K* 2.0 - http://www.ncsa.uiuc.edu/STI/ALG/

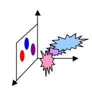
Unsupervised Learning and Constructive Induction

- **Unsupervised Learning in Support of Supervised Learning**
  - Given: $D \equiv$ labeled vectors (x, y)
  - Return: $D' \equiv$ transformed training examples (x', y')
  - Solution approach: constructive induction
    - Feature "construction": generic term
    - Cluster definition
- **Feature Construction: Front End**
  - Synthesizing new attributes
    - Logical: $x_1 \vee \neg x_2$, arithmetic: $x_1 + x_5 / x_2$
    - Other synthetic attributes: $f(x_1, x_2, \ldots, x_n)$, etc.
  - Dimensionality-reducing projection, feature extraction
  - Subset selection: finding relevant attributes for a given target y
  - Partitioning: finding relevant attributes for given targets $y_1, y_2, \ldots, y_p$
- **Cluster Definition: Back End**
  - Form, segment, and label clusters to get intermediate targets y'
  - Change of representation: find an (x', y') that is good for learning target y



Clustering: Relation to Constructive Induction

- **Clustering versus Cluster Definition**
  - Clustering: 3-step process
  - Cluster definition: "back end" for feature construction



- **Clustering: 3-Step Process**
  - Form
    - $(x_1', \ldots, x_k')$ in terms of $(x_1, \ldots, x_n)$
    - *NB*: *typically* part of construction step, *sometimes* integrates both
  - Segment
    - $(y_1', \ldots, y_J')$ in terms of $(x_1', \ldots, x_k')$
    - *NB*: number of clusters J not necessarily same as number of dimensions k
  - Label
    - Assign names (discrete/symbolic labels $(v_1', \ldots, v_J')$) to $(y_1', \ldots, y_J')$
    - Important in document categorization (e.g., clustering text for info retrieval)
- **Hierarchical Clustering: Applying Clustering Recursively**

Terminology

- **Expectation-Maximization (EM) Algorithm**
  - Iterative refinement: repeat until convergence to a locally optimal label
  - Expectation step: estimate parameters with which to simulate data
  - Maximization step: use simulated ("fictitious") data to update parameters
- **Unsupervised Learning and Clustering**
  - Constructive induction: using unsupervised learning *for* supervised learning
    - Feature construction: "front end" - *construct* new x values
    - Cluster definition: "back end" - use these to reformulate y
  - Clustering problems: formation, segmentation, labeling
  - Key criterion: distance metric (points closer intra-cluster than inter-cluster)
  - Algorithms
    - *AutoClass*: Bayesian clustering
    - Principal Components Analysis (PCA), factor analysis (FA)
    - Self-Organizing Maps (SOM): topology preserving transform (dimensionality reduction) for competitive unsupervised learning

Summary Points

- **Expectation-Maximization (EM) Algorithm**
- **Unsupervised Learning and Clustering**
  - Types of unsupervised learning
    - Clustering, vector quantization
    - Feature extraction (typically, dimensionality reduction)
  - Constructive induction: unsupervised learning *in support of* supervised learning
    - Feature construction (*aka* feature extraction)
    - Cluster definition
  - Algorithms
    - EM: mixture parameter estimation (e.g., for *AutoClass*)
    - *AutoClass*: Bayesian clustering
    - Principal Components Analysis (PCA), factor analysis (FA)
    - Self-Organizing Maps (SOM): projection of data; competitive algorithm
  - Clustering problems: formation, segmentation, labeling
- **Next Lecture: Time Series Learning and Characterization**