Unsupervised learning from data:
Clustering algorithms and applications

Greg Hamerly
ghamerly@cs.ucsd.edu

Computer Science Department
University of California, San Diego

June 7, 2002
My story

I got my BS in computer science from Cal Poly in 1999. I love to program, and wanted more science. Enter graduate school.

At first I thought Artificial Intelligence was hokey, but it turns out to be a very interesting cross-section of

- real-world problems
- understanding “learning” and “intelligence”
- math, statistics, computer science, other fields
- programming, simulation, experimentation

So now I work in the AI lab at UCSD! I’m in my third year, and am enjoying it.
Outline

1. Machine learning introduction
2. Project 1: Clustering for hard drive failure prediction
3. Project 2: Finding better-quality clusterings
Machine learning is a field related to AI which aims to “learn” concepts from data. Steps of machine learning:

1. **Pose a problem**, such as “How can we discriminate between fraudulent and valid credit card transactions?”

2. **Gather data** related to the problem at hand, such as time/amount/location of credit card purchase.

3. **Pick a model** to represent the concept, such as a Hidden Markov Model, or a Neural Network.

4. **Choose a learning algorithm** to learn the parameters of the model from the data.

5. **Apply the model** by using it to make predictions on new data, or by inferring some knowledge from the model (e.g. “typically, a large credit card purchase immediately after a small purchase is fraudulent”).
Clustering (unsupervised learning)

Clustering is the task of learning self-similar groups in data. We can cluster anything on which we can measure “distance” between two points.

Example: Given a dataset of locations of stars and planets, we could discover clusters of related bodies (e.g. galaxies) based on their distances from one another.

A typical model for clustering is to have small “density estimators” and learn their locations.
Clustering example

On the left, the colors are the labels of the data. On the right is the unlabeled dataset. Clustering is inferring the left graph from the right one.
Real applications of data clustering

Clustering is not always an end, but often a mean:

- image segmentation into objects
- data mining and knowledge discovery
- probability density estimation
- unsupervised classification (e.g. documents on the web)
- information retrieval
Project 1: Clustering for hard drive failure prediction
Project 1: Clustering for HD failure prediction

The Center for Magnetic Recording Research (CMRR) at UCSD is interested in predicting hard drive failure.

**Goal:** learn a model of normally-operating drives using clustering to predict will-fail/will-not-fail on other drives.

**Pros:**

- Preventative maintenance: customers can back up their data and return the drive.
- Drive manufacturers can see how drives fail.

**Cons:**

- Falsely predicting failure looks bad to the consumer, and costs the manufacturer money.
Learning from “SMART” data

We have data about hard drive behavior, such as:

- RSE  read soft errors
- SKE  seek errors
- GDC  grown defects
- POH  power-on hours

There are a total of 9 such attributes.

This comes from an industry “standard” called SMART (Self-Monitoring And Reporting Technology, introduced by Compaq in 1992).
Our dataset

We have a dataset representing 1,934 Quantum Eclipse hard disk drives analyzed by drive engineers.

Of these 1,934, only 9 were actual failed drives, though 16 were labeled as failures by the returner. The rest were still operational.

From each drive we have time-stamped snapshots of the SMART data leading up to the end of its life. The total number of snapshots is 94,022. This is the data we want to model.
Our model: NBEM

NB stands for naïve Bayes, an intuitive statistical model that is basically a histogram of the data. It is “naïve” because each dimension is considered conditionally independent.

Each NB model gives a probability $p(x|M)$, where $x$ is a datapoint and $M$ is the model.

$$p(x|M) = \prod_{j=1}^{d} p(x_j|M)$$

$$p(x_j|M) = \frac{\text{count}(x_j, M)}{\text{count}(M)}$$

We must discretize the data into “bins” before we can compute a histogram. In our case, we create 5 bins: the value zero, and 4 quartiles of the data.
An ensemble of NB models

The whole model is a linear combination of NB models:

\[ p(x) = \sum_{i}^{k} p(i)p(x|i) \]

where \( p(i) \) is a prior probability of model \( i \), and \( p(x|i) \) is the probability of \( x \) according to model \( i \).

We have to train the model, to learn the correct values for \( p(x_j|i) \) and \( p(i) \).
Training using Expectation-Maximization

EM stands for Expectation-Maximization. The basic idea is:

1. **Initialize** each NB model randomly (e.g. set $p(x_j|i)$ and $p(i)$ randomly).

2. **Expectation**: compute $p(i|x)$ for each $x$ and $i$ using Bayes rule.

3. **Maximization**: re-compute $p(x_j|i)$ and $p(i)$ based on $p(i|x)$.

...repeat steps 2 and 3 until nothing changes.
Predicting failure/testing the model

We will predict failure of a drive if $p(x) < t$, where $t$ is a chosen threshold.

We try many different values of $t$.

We test our model by training on 90% of the good drives and testing on 10% of the good drives and the bad drives. We do this 10 times to obtain averages (cross-validation).

Performance measured by true positive and false positive rates.

- TPR = percent of drives correctly predicted to fail
- FPR = percent of drives incorrectly predicted to fail
Results

Testing with $k = 2$ NB models...
Conclusions

We were able to perform better than the best current industry estimate using a simple histogram-based model.

More information is not always better! We did better with 3 attributes than with 9 attributes.
Project 2: Finding better-quality clusterings
Project 2: Finding better-quality clusterings

A very popular clustering algorithm, \( k \)-means, finds groups in metric data.

Introduced by MacQueen in 1967, it is fast and simple to implement.

Its model assumption is that a single point (a “center”) represents data that is closer to it than to any other center. It is not a statistical model.
Training $k$-means

The goal of $k$-means is to minimize the cluster distortion:

$$KM(X, C) = \sum_{i=1}^{n} \min_{j \in \{1..k\}} ||x_i - c_j||^2$$

It is similar to the EM algorithm we saw earlier:

1. **Initialize** the centers randomly.

2. **Assign each data point** to its closest center.

3. **Move each center** to the mean of the points it “owns”.

Problem: it often finds poor solutions, because of the random initialization and “hard membership” of datapoints in centers.
A good run of $k$-means
A bad run of $k$-means
Other $k$-means variants

There are other models similar to $k$-means but which minimize different objective functions.

- **Gaussian EM** uses EM training on Gaussian centers, and has soft membership of datapoints in centers.
- **Fuzzy $k$-means** is $k$-means with soft membership.
- **$k$-harmonic means** has soft membership and a changing data weighting function.

Training these algorithms is similar to $k$-means.

**Question:** which algorithms find the “best-quality” clusterings?
$k$-harmonic means (KHM)

This new algorithm by Bin Zhang at HP Labs uses a novel objective function based on the harmonic mean.

$$KHM(X, C) = \sum_{i=1}^{n} \frac{k}{\sum_{j=1}^{k} \frac{1}{||x_i - c_j||^p}}$$

This measures quality in terms of the harmonic mean of the distance between each point and every center. Thus, it has soft membership.
Harmonic mean versus minimum

The harmonic mean is similar to minimum, but it is smooth and differentiable.

This makes it similar to $k$-means, but the harmonic mean takes contributions from every center, not just the closest.
Changing weight function

The KHM algorithm has a novel weighting function which comes from its objective function. It gives more weight to points that are far away from every center.
Tests

We created two new algorithms, H1 and H2, which are combinations of KM and KHM.

- H1 has hard membership and changing weights.
- H2 has soft membership but static weights.

We tried each algorithm (KM, KHM, GEM, FKM, H1, H2) on synthetic 2/4/6-dimensional datasets with 50 true clusters and 2500 total points. On 100 datasets we ran each algorithm from the same initializations.
Results for 2-d data using different initializations

Left: Forgy initialization (choose points from the dataset randomly)

Right: Random partition initialization (assign points to centers, then compute one update step)
Conclusions

There is a strong need for good-quality clustering algorithms.

The two most common algorithms, KM and GEM, get stuck at bad solutions, especially on bad initializations.

KHM with its soft membership and changing weights finds better solutions.

The soft weights benefitted $k$-means the most, but changing weights were also helpful.
Image segmentation example

Original “hand” image

Convert this image into data points based on color (L,U,V) and position (X,Y).
Image segmentation example

with $k = 5$...

Initialization Random partition Forgy
$k$-means

$k$-harmonic means

KHM finds the same, better segmentation for both initializations.