Instructor: Ryan Lowe (ryan.lowe@cs.mcgill.ca)

Slides mostly by: Joelle Pineau (jpineau@cs.mcgill.ca)

Class web page: www.cs.mcgill.ca/~hvanho2/comp551

Unless otherwise noted, all material posted for this course are copyright of the instructor, and cannot be reused or reposted without the instructor's written permission.
Announcements

• No class next week
  – Easter Monday
  – Midterm on Wednesday

• Project 5: problem statement released. Three tracks
  – Improve baselines
  – Model ablation
  – Reproducibility challenge
What is unsupervised learning?

- Given only input data: \( D = \langle x_i \rangle, \ i=1:n \), find some patterns or regularity in the data.

- Typically use **generative approaches**: model the available data.

- Different classes of problems:
  1. Clustering
  2. Anomaly detection
  3. Dimensionality reduction
  4. Autoregression
A simple clustering example

• A fruit merchant approaches you, with a set of apples to classify according to their variety.
  – Tells you there are five varieties of apples in the dataset.
  – Tells you the weight and colour of each apple in the dataset.

• Can you label each apple with the correct variety?
  – What would you need to know / assume?

\[ \text{Data} = \langle x_1, \_ \rangle, \langle x_2, \_ \rangle, \ldots, \langle x_n, \_ \rangle \]
A simple clustering example

• You know there are 5 varieties.

• Assume each variety generates apples according to a (variety-specific) 2-D Gaussian distribution.
A simple clustering example

• You know there are 5 varieties.

• Assume each variety generates apples according to a (variety-specific) 2-D Gaussian distribution.

• If you know $\mu_i$, $\sigma_i^2$ for each class, it’s easy to classify the apples.

• If you know the class of each apple, it’s easy to estimate $\mu_i$, $\sigma_i^2$. 
A simple clustering example

- You know there are 5 varieties.

- Assume each variety generates apples according to a (variety-specific) 2-D Gaussian distribution.

- If you know \( \mu_i, \sigma_i^2 \) for each class, it’s easy to classify the apples.

- If you know the class of each apple, it’s easy to estimate \( \mu_i, \sigma_i^2 \).

What if we know neither?
A simple algorithm: K-means clustering

• **Objective:** Cluster \( n \) instances into \( K \) distinct classes.

• **Preliminaries:**
  - **Step 1:** Pick the desired number of clusters, \( K \).
  - **Step 2:** Assume a parametric distribution for each class (e.g. Normal).
  - **Step 3:** Randomly estimate the parameters of the \( K \) distributions.
A simple algorithm: K-means clustering

• **Objective:** Cluster $n$ instances into $K$ distinct classes.

• **Preliminaries:**
  – **Step 1:** Pick the desired number of clusters, $K$.
  – **Step 2:** Assume a parametric distribution for each class (e.g. Normal).
  – **Step 3:** Randomly estimate the parameters of the $K$ distributions.

• **Iterate, until convergence:**
  – **Step 4:** Assign instances to the most likely classes based on the current parametric distributions.
  – **Step 5:** Estimate the parametric distribution of each class based on the latest assignment.
This data could easily be modeled by Gaussians.

1. Ask user how many clusters.

*Image courtesy of Andrew Moore, Carnegie Mellon U.*
K-means algorithm

This data could easily be modeled by Gaussians.

1. Ask user how many clusters.
2. Randomly guess k centers:
   \[ \{ \mu_1, \ldots, \mu_k \} \] (assume \( \sigma^2 \) is known).
K-means algorithm

This data could easily be modeled by Gaussians.

1. Ask user how many clusters.
2. Randomly guess k centers:
   \[ \{ \mu_1, \ldots, \mu_k \} \] (assume \( \sigma^2 \) is known).
3. Assign each data point to the center.

*Image courtesy of Andrew Moore, Carnegie Mellon U.*
K-means algorithm

This data could easily be modeled by Gaussians.

1. Ask user how many clusters.
2. Randomly guess k centers:
   \( \{ \mu_1, \ldots, \mu_k \} \) (assume \( \sigma^2 \) is known).
3. Assign each data point to the center.
4. Each center finds the centroid of the points it owns.

*Image courtesy of Andrew Moore, Carnegie Mellon U.*
K-means algorithm

This data could easily be modeled by Gaussians.

1. Ask user how many clusters.
2. Randomly guess k centers:
   \[ \{ \mu_1, \ldots, \mu_k \} \text{ (assume } \sigma^2 \text{ is known)} \]
3. Assign each data point to the closest center.
4. Each center finds the centroid of the points it owns… and jumps there.

Image courtesy of Andrew Moore, Carnegie Mellon U.
K-means algorithm starts
K-means algorithm continues (2)

Image courtesy of Andrew Moore, Carnegie Mellon U.
K-means algorithm continues (3)
K-means algorithm continues (4)

Image courtesy of Andrew Moore, Carnegie Mellon U.
K-means algorithm continues (5)

Image courtesy of Andrew Moore, Carnegie Mellon U.
K-means algorithm continues (6)

Image courtesy of Andrew Moore, Carnegie Mellon U.
K-means algorithm continues (7)
K-means algorithm continues (8)

Image courtesy of Andrew Moore, Carnegie Mellon U.
K-means algorithm continues (9)

Image courtesy of Andrew Moore, Carnegie Mellon U.
K-means algorithm terminates
A simple algorithm: K-means clustering

- **Objective**: Cluster $n$ instances into $K$ distinct classes.

- **Preliminaries**:
  - **Step 1**: Pick the desired number of clusters, $K$.
  - **Step 2**: Assume a parametric distribution for each class (e.g. Normal).
  - **Step 3**: Randomly estimate the parameters of the $K$ distributions.

- **Iterate, until convergence**:
  - **Step 4**: Assign instances to the most likely classes based on the current parametric distributions. **Hard assignment**
  - **Step 5**: Estimate the parametric distribution of each class based on the latest assignment. **Maximization step**
Properties of K-means

- Optimality?
Properties of K-means

- **Optimality?**
  - Converges to a local optimum.
  - Can use random re-starts to get better local optimum.
  - Alternately, can choose your initial centers carefully:
    - Place $\mu_1$ on top of a randomly chosen datapoint.
    - Place $\mu_2$ on top of datapoint that is furthest from $\mu_1$.
    - Place $\mu_3$ on top of datapoint that is furthest from both $\mu_1$ and $\mu_2$. 
Properties of K-means

- **Optimality?**
  - Converges to a local optimum.
  - Can use random re-starts to get better local optimum.
  - Alternately, can choose your initial centers carefully:
    - Place $\mu_1$ on top of a randomly chosen datapoint.
    - Place $\mu_2$ on top of datapoint that is furthest from $\mu_1$.
    - Place $\mu_3$ on top of datapoint that is furthest from both $\mu_1$ and $\mu_2$.

- **Complexity?**
Properties of K-means

- **Optimality?**
  - Converges to a local optimum.
  - Can use random re-starts to get better local optimum.
  - Alternately, can choose your initial centers carefully:
    - Place $\mu_1$ on top of a randomly chosen datapoint.
    - Place $\mu_2$ on top of datapoint that is furthest from $\mu_1$.
    - Place $\mu_3$ on top of datapoint that is furthest from both $\mu_1$ and $\mu_2$.

- **Complexity?** $O(knm)$ where $k = \#\text{centers}$
  
  $n = \#\text{datapoints}$
  
  $m = \text{dimensionality of data}$
K-means: pros and cons

• Good:
  – We realize that maximizing parameters is easy once we have assignments

• Bad:
  – What about points that are about equally far to two clusters?
  – We can only update the mean (not variance)
  – We have to assume equal variance between clusters
Beyond K-means

• How do we fit data where variance is unknown or non-identical between clusters?

Copyright C.M. Bishop, PRML
Gaussian Mixture Model

- **Idea**: Fit data with a combination of Gaussian distributions.

- Like a ‘soft’ version of K-means

- What defines a set of Gaussians?
**Gaussian Mixture Model**

- **Idea**: Fit data with a combination of Gaussian distributions.

- Write \( p(x) \) as a linear combination of Gaussians:
  \[
  p(x) = \sum_{k=1}^{K} p(z_k) \ p(x \mid z_k)
  \]
  where \( p(z_k) \) is the probability of the \( k^{th} \) mixture component
  and \( p(x \mid z_k) = N(x \mid \mu_k, \sigma_k^2) \) is the prob. of \( x \) for the \( k^{th} \) mixture component.

- Determining \( p(z \mid x) \) is easy once we know parameters \( p(z_k), \mu_k, \sigma_k^2 \),
  (Bayes rule)
Gaussian **Mixture** Model

- **Maximum likelihood** often gives good parameter estimate:

\[
p(X|\theta) = \sum_Z p(X,Z|\theta)
\]

- **Why is it hard here?**
Expectation Maximization (more generally)

- Iterative method for learning the maximum likelihood estimate of a probabilistic model, when the model contains unobservable variables.
Expectation Maximization (more generally)

• Iterative method for learning the maximum likelihood estimate of a probabilistic model, when the model contains unobservable variables.

• Main idea:
  – If we knew all variables (e.g. cluster assignments) we could easily maximize the likelihood
  – With unobserved variables, we “fantasize” how the data should look based on the current parameter setting. I.e. Compute Expected sufficient statistics.
  – Then we Maximize parameter setting, based on these statistics.
Expectation Maximization (more generally)

• Start with some initial parameter setting.

• **Repeat** (as long as desired):
  
  – **Expectation (E) step**: Complete the data by assigning “values” to the missing items.
  
  – **Maximization (M) step**: Compute the maximum likelihood parameter setting based on the completed data.

Once the data is completed (E-step), computing the log-likelihood and new parameters (M-step) is easy! This is what we did for K-means.
EM for clustering

- **Objective**: Cluster \( n \) instances into \( K \) distinct classes.

- **Preliminaries**:
  - **Step 1**: Pick the desired number of clusters, \( K \).
  - **Step 2**: Assume a parametric distribution for each class (e.g. Normal).
  - **Step 3**: Randomly estimate the parameters of the \( K \) distributions.

- **Iterate, until convergence**:
  - **Step 4**: Assign responsibility for instances to classes based on the current parametric distributions. **Soft assignment**
  - **Step 5**: Estimate the parametric distribution of each class based on the latest assignment. **Maximization step**
EM for clustering
Expectation Maximization: Properties

- Likelihood function is guaranteed to improve (or stay the same) with each iteration.
- Convergence to a local optimum of the likelihood function.
- Re-starts with different initial parameters are often necessary.
- K-means can be seen as a specific case of EM (where variance is fixed to a value that decreases to 0)

EM is very useful in practice!
Anomaly detection

http://www.anomalydetectionresearch.com
Anomaly detection

- K-means (and other discriminative approaches) tend to be ineffective when one class is much more rare than the other.
Anomaly detection

- K-means (and other discriminative approaches) tend to be ineffective when one class is much more rare than the other.

- A simple **generative** approach:
  - Fit a model, $p(x)$ using the input data.
  - Set a decision threshold $\varepsilon$ and predict $Y = \{1 \text{ if } p(x) > \varepsilon, \ 0 \text{ otherwise}\}$.
  - Use a validation set to measure performance (can use cross-validation to set $\varepsilon$).
Anomaly detection vs Supervised learning

**Anomaly detection**
- Small number of positive examples (e.g. <10).
- Large number of negative examples (e.g. >100).

**Supervised learning**
- (Usually) similar number of positive and negative examples

Anomaly detection vs Supervised learning

**Anomaly detection**

- Small number of positive examples (e.g. <10).
- Large number of negative examples (e.g. >100).
- Many different “types” of anomalies, so don’t want to fit a model for the positive class.

**Supervised learning**

- (Usually) similar number of positive and negative examples
- More homogeneity within classes, or enough data to sufficiently characterize each class.

A simple example

Does the distribution of nominal data look familiar?

A simple example

Another GMM!

A simple example

- GMM can be fit again with EM
- Note that before we were mainly interested in the cluster assignments (which items go together)
- Here we are interested in the final density

Another GMM!
Dimensionality reduction

• Given points in an $m$-dimensional space (for large $m$), project to a low dimensional space while preserving trends in the data.

• Principal Components Analysis
Autoregressive models for time series

• The problem:
  – Given a time series:  \( X = \{x_1, x_2, \ldots, x_T\} \)
  – Predict \( x_t \) from \( x_{1:t-1} \).
Autoregressive models for time series

• The problem:
  – Given a time series: \( X = \{x_1, x_2, \ldots, x_T\} \)
  – Predict \( x_t \) from \( x_{1:t-1} \).

• A simple autoregressive (AR) model:
  \[
  X_t = w_0 + \varepsilon + \sum_{i=1:p} w_i x_{t-i} + \varepsilon_t
  \]
  where \( w_i \) are the parameters and \( \varepsilon_t \) is white noise.
Autoregressive models for time series

• The problem:
  – Given a time series: $X = \{x_1, x_2, \ldots, x_T\}$
  – Predict $x_t$ from $x_{1:t-1}$.

• A simple autoregressive (AR) model:
  $$X_t = w_0 + \varepsilon + \sum_{i=1:p} w_i x_{t-i} + \varepsilon_t$$
  where $w_i$ are the parameters and $\varepsilon_t$ is white noise.

• Can also use more complicated models (e.g. neural networks) for this
WaveNet

- Uses a 1-D CNN to predict next audio sample
- Can add supervised component: condition on text to be generated
WaveNet

- Audio samples: https://deepmind.com/blog/wavenet-generative-model-raw-audio/
What you should know

• The general form of the unsupervised learning problem.

• Basic functioning and properties of useful algorithms:
  – K-means / Gaussian mixture models
  – Expectation-maximization

• Characteristics of common problems:
  – clustering, anomaly detection, dimensionality reduction, autoregression, autoencoding