Machine Learning

4771

Instructors:
Adrian Weller and Ilia Vovsha
Lecture 16

• Update on course so far: instructors, TAs, midterm, HW3
• Review Clustering, K-Means (15:13-21)
• Mixture Models and Hidden Variables
• Expectation Maximization for Gaussian Mixtures
• Entropy, KL Divergence, Jensen’s Inequality
Example: Vector Quantization

- Use K-means for lossy data compression

- Each pixel is a point in 3D (R,G,B) space. Instead need only store ?
Mixtures for Flexibility

- With mixtures (e.g. mixtures of Gaussians) we can handle complicated distributions (e.g. multi-bump, nonlinear).

  subpopulations:  
  G1 = compact car  
  G2 = mid-size car  
  G3 = cadillac

- In fact, if we have enough Gaussians (maybe infinite) we can approximate any distribution...
Mixtures as Hidden Variables

• Consider a dataset with K subpopulations but don’t know which subpopulation each point belongs to

  e.g. consider height of adult people, we see K=2 subpopulations: males & females

  e.g. looking at weight and height of people we see K=2 subpopulations: males & females

• Because of the ‘hidden’ variable (z can be 1 or 2), these distributions are not Gaussians but Mixture of Gaussians

\[
p(\mathbf{x}) = \sum_z p(\mathbf{x}, z) = \sum_z p(z) p(\mathbf{x} \mid z) = \sum_z \pi_z N(\mathbf{x} \mid \mu_z, \Sigma_z)
= \sum_{k=1}^{K} \pi_k \frac{1}{(2\pi)^{D/2} \sqrt{\Sigma_k}} \exp \left( -\frac{1}{2} (\mathbf{x} - \mu_k)^T \Sigma_k^{-1} (\mathbf{x} - \mu_k) \right)
\]
Hidden / Unlabeled = Clustering

• Recall classification problem:
  maximize the log-likelihood:

\[
l(\pi, \mu, \Sigma) = \sum_{n=1}^{N} \log p(\vec{x}_n, z_n | \pi, \mu, \Sigma)
\]

\[
= \sum_{n=1}^{N} \log \pi_k N(\vec{x}_n | \mu_k, \Sigma_k)
\]

• If we don’t know the class, marginalize over hidden variable maximizing the log-likelihood with unlabeled data:

\[
l = \sum_{n=1}^{N} \log p(\vec{x}_n | \pi, \mu, \Sigma) = \sum_{n=1}^{N} \log \sum_{z=1}^{K} p(\vec{x}_n, z | \pi, \mu, \Sigma)
\]

\[
= \sum_{n=1}^{N} \log \left( \pi_1 N(\vec{x}_n | \mu_1, \Sigma_1) + \ldots + \pi_K N(\vec{x}_n | \mu_K, \Sigma_K) \right)
\]

• Instead of classification, we now have a clustering problem
Mixture of Gaussians

- Represent each hidden $z$ integer (1 to $K$) coded as a hidden binary indicator vector $z$
  
  \[ \tilde{z} \in \mathcal{B}^K, \sum_{k=1}^K \tilde{z}(k) = 1 \text{ or } \tilde{z} \in \{\bar{\delta}_1, \ldots, \bar{\delta}_K\} \text{ where } \bar{\delta}_k(k) = 1 \]

- Each likelihood requires summing over all possible $z$
  
  \[ p(\tilde{x} | \theta) = \sum_z p(\tilde{z} | \theta) p(\tilde{x} | \tilde{z}, \theta) = \sum_{k=1}^K p(\tilde{z} = \bar{\delta}_k | \theta) p(\tilde{x} | \tilde{z} = \bar{\delta}_k, \theta) \]

  - mixing proportions (prior) = \[ \pi_k = p(\tilde{z} = \bar{\delta}_k | \theta) \]
  
  - mixture components (likelihood) = \[ p(\tilde{x} | \tilde{z} = \bar{\delta}_k, \theta) \]
  
  - posteriors (responsibilities) = \[ \tau_{n,k} = p(\tilde{z} = \bar{\delta}_k | \tilde{x}_n, \theta) = \frac{p(\tilde{x}_n | \tilde{z} = \bar{\delta}_k, \theta) p(\tilde{z} = \bar{\delta}_k | \theta)}{p(\tilde{x}_n | \theta)} \]

  log likelihood = \[ \sum_{n=1}^N \log p(\tilde{x}_n | \pi, \mu, \Sigma) = \sum_{n=1}^N \log \sum_{k=1}^K \pi_k N(\tilde{x}_n | \bar{\mu}_k, \Sigma_k) \]

- Can’t easily take derivatives of log-likelihood and set to 0.
- Not nice, seems to need gradient ascent...
- Or, can we do something else?
K-Means Clustering

- An old “heuristic” clustering algorithm
- Gobble up data with a divide & conquer scheme
- Assume each point $x$ has a discrete multinomial vector $z$
- Chicken and Egg problem:
  If know classes, we can get model (max likelihood!)
  If know the model, we can predict the classes (classifier!)

K-means Algorithm:

0) Input dataset $\{\tilde{x}_1, \ldots, \tilde{x}_N\}$
1) Randomly initialize means $\bar{\mu}_1, \ldots, \bar{\mu}_K$
2) Find closest mean for each point $z_n(k) = \begin{cases} 1 & \text{if } k = \arg\min_j \|\tilde{x}_n - \bar{\mu}_j\|^2 \\ 0 & \text{otherwise} \end{cases}$
3) Update means $\bar{\mu}_k = \frac{\sum_{n=1}^{N} \tilde{x}_n z_n(k)}{\sum_{n=1}^{N} z_n(k)}$
4) If any $z$ has changed go to 2

TIP: In practice, for EM approaches, sometimes easier to initialize $z$ then start with an update to means.
K-Means Clustering

• Geometric, each point goes to closest Gaussian
• Recompute the means by their assigned points
• Essentially minimizing the following cost function:

$$\min_{\mu} \min_{z} J(\vec{\mu}_1, \ldots, \vec{\mu}_K, \vec{z}_1, \ldots, \vec{z}_N) = \sum_{n=1}^{N} \sum_{k=1}^{K} \vec{z}_n(k) \| \vec{x}_n - \vec{\mu}_k \|^2$$

$$\vec{z}_n(k) = \begin{cases} 
1 & \text{if } k = \arg \min_j \| \vec{x}_n - \vec{\mu}_j \| \\
0 & \text{otherwise}
\end{cases}$$

$$\vec{\mu}_k = \frac{\sum_{n=1}^{N} \vec{x}_n \vec{z}_n(k)}{\sum_{n=1}^{N} \vec{z}_n(k)}$$

• Guaranteed to improve per iteration and converge
• Like Coordinate Descent (lock one var, maximize the other)
• A.k.a. Axis-Parallel Optimization or Alternating Minimization

Local Minima
Example: K-means using Old Faithful data set

X marks show $\mu_k$ locations
(a) Initialization
(b) First E step
(c) First M step
... to convergence
Expectation-Maximization (EM)

• EM allows a soft/fuzzy version of K-Means (winner-takes-all, closest Gaussian Mean completely wins datapoint)

\[
\hat{z}_n(k) = \begin{cases} 
1 & \text{if } k = \arg \min_j \| \overrightarrow{x}_n - \overrightarrow{\mu}_j \|^2 = \arg \max_j N(\overrightarrow{x}_n | \overrightarrow{\mu}_j, I) = \arg \max_j p(\overrightarrow{x}_n | \overrightarrow{\mu}_j) \\
0 & \text{otherwise}
\end{cases}
\]

• Instead, consider soft percentage assignment of datapoint

assign \( \propto \pi_k \frac{1}{(2\pi)^{D/2}} \exp \left( -\frac{1}{2} \| \overrightarrow{x}_n - \overrightarrow{\mu}_k \|^2 \right) \)

• EM is ‘less greedy’ than K-Means uses \( \tau_{n,k} = p(\hat{z} = \delta_k | \overrightarrow{x}_n, \theta) \) as shared responsibility for \( \overrightarrow{x}_n \)

• Update for the means are then ‘weighted’ by responsibilities

\[
\overrightarrow{\mu}_k = \frac{\sum_{n=1}^{N} \tau_{n,k} \overrightarrow{x}_n}{\sum_{n=1}^{N} \tau_{n,k}}
\]
Example: EM Mixture of Gaussians using Old Faithful data set

Initialization with same $\mu_k$
Showing 1 stdev contours
(b) First E step
(c) First M step, after L=1 complete cycle
... subsequent cycles

Note:
Typically longer convergence time, and more overhead per cycle than K-means.

How might we initialize?
Expectation-Maximization

• EM uses expected value of $\tilde{z}_n(k)$ rather than max

$$\tau_{n,k} = E\left\{ \tilde{z}_n(k) \mid \tilde{x}_n \right\} = p\left( \tilde{z}_n = \delta_k \mid \tilde{x}_n, \theta \right)$$

• EM updates covariances, mixing proportions AND means...

• The algorithm for Gaussian mixtures:

**EXPECTATION:**

$$\tau_{n,k}^{(t)} = \frac{\pi_k^{(t)} N\left( \tilde{x}_n \mid \tilde{\mu}_k^{(t)}, \Sigma_k^{(t)} \right)}{\sum_j \pi_j^{(t)} N\left( \tilde{x}_n \mid \tilde{\mu}_j^{(t)}, \Sigma_j^{(t)} \right)}$$

**MAXIMIZATION:**

$$\tilde{\mu}_k^{(t+1)} = \frac{\sum_n \tau_{n,k}^{(t)} \tilde{x}_n}{\sum_n \tau_{n,k}^{(t)}}$$

$$\pi_k^{(t+1)} = \frac{\sum_n \tau_{n,k}^{(t)}}{N}$$

$$\Sigma_k^{(t+1)} = \frac{\sum_n \tau_{n,k}^{(t)} \left( \tilde{x}_n - \tilde{\mu}_k^{(t+1)} \right) \left( \tilde{x}_n - \tilde{\mu}_k^{(t+1)} \right)^T}{\sum_n \tau_{n,k}^{(t)}}$$

• Neat demo... [http://www.cs.cmu.edu/~alad/em/](http://www.cs.cmu.edu/~alad/em/)

• Makes intuitive sense, but what can we prove?
Expectation-Maximization

• EM uses expected value of $\tilde{z}_n(k)$ rather than max
  $$\tau_{n,k} = E\left\{ \tilde{z}_n(k) \mid \tilde{x}_n \right\} = p\left( \tilde{z}_n = \delta_k \mid \tilde{x}_n, \theta \right)$$

• EM updates covariances, mixing proportions AND means...

• The algorithm for Gaussian mixtures:

  **EXPECTATION:**
  $$\tau_{n,k}^{(t)} = \frac{\pi_k^{(t)} N\left( \tilde{x}_n \mid \tilde{\mu}_k^{(t)}, \Sigma_k^{(t)} \right)}{\sum_j \pi_j^{(t)} N\left( \tilde{x}_n \mid \tilde{\mu}_j^{(t)}, \Sigma_j^{(t)} \right)}$$

We'd like the true probability $p(z \mid x, \theta)$

Instead we use an approximation $q_t(z) = p(z \mid x, \theta_t)$

Need a way to think about the difference between $p$ and $q_t$

• Neat demo... [http://www.cs.cmu.edu/~alad/em/](http://www.cs.cmu.edu/~alad/em/)

• Makes intuitive sense, but what can we prove?
Entropy & KL Divergence

• Step back, reconsider Entropy, introduced in 9:16
• We’ll extend the idea to Relative Entropy of 2 variables
• We’ll also need Jensen’s Inequality, see 11:7

• Let \( h(x) \) be the information content of an event \( x \)
• We’d like: if \( x \) and \( y \) unrelated, then \( h(x,y) = h(x) + h(y) \)
• Two unrelated events \( \sim \) independent, so \( p(x,y) = p(x)p(y) \)
• Leads to

\[
h(x) = \log \frac{1}{p(x)} = -\log p(x) \geq 0 \quad \text{units? base 2 \textit{bits} base e \textit{nats}}
\]

• For a discrete random variable \( X \), its entropy is the average information content

\[
H(X) = \mathbb{E}_{p(x)}[h(x)] = \sum_x p(x) \log \frac{1}{p(x)} = -\sum_x p(x) \log p(x)
\]
Entropy Properties

• Loosely speaking, $H(X)$ is an $\varepsilon$-achievable lower bound on the average code rate (Shannon noiseless coding theorem)

• Example:
  • Variable $X$ has 8 states, all equally likely
  • What’s $H(X)$ in bits?

• Example: (Cover & Thomas, 1991)
  • Variable $Y$ has 8 states, probabilities

$$H(X) = \frac{1}{2} \log_2 2 + \frac{1}{4} \log_2 4 + \ldots$$

$$= \frac{1}{2} \cdot 1 + \frac{1}{4} \cdot 2 + \frac{1}{8} \cdot 3 + \frac{1}{16} \cdot 4 + \frac{4}{64} \cdot 6 = 2$$

Possible code 0 10 110 1110 \{111100,111101,111110,111111\}
Example: letters in English

The entropy of a randomly selected letter in an English document is about 4.11 bits (Mackay 2.4)

Compare $\log_2 27 \approx 4.75$
Entropy Properties

For discrete $X$

- $H(X) \geq 0$
- $H(X) = 0$ iff exists some value $y$ s.t. $X = y$ a.s
- If $X$ takes finite $n$ possible values, then $H(X) \leq \log n$ with equality iff $X$ is uniformly distributed (maximum entropy)

For continuous $X$, define differential entropy

$$H(X) = - \int p(x) \log p(x) \, dx$$

- Note now $H(X)$ need not be positive (e.g. consider $U[0,a]$)
- For given mean and variance, distribution with maximum entropy is a Gaussian
KL Divergence

• Suppose have a probability distribution $p(x)$
• We’ll approximate it with some distribution $q(x)$
• Consider coding scheme using $q(x)$: information content based on $q(x)$ but average over the true distribution $p(x)$
• Hence minimum average additional information required to specify $x$ is

$$-\int p(x) \log q(x) \, dx - (\int p(x) \log p(x) \, dx) = \int p(x) \log \frac{p(x)}{q(x)} \, dx$$

$$=: KL(p \parallel q)$$

• Kullback-Leibler divergence or relative entropy between distributions $p(x)$ and $q(x)$, continuous or discrete
• Not symmetric but provides a notion of distance
Key result: KL Divergence ≥ 0

- Recall Jensen’s Inequality
  - For convex $f$, $\mathbb{E}[f(x)] \geq f(\mathbb{E}[x])$

- Apply to KL divergence:
  \[
  KL(p \parallel q) := \int p(x) \cdot -\log \frac{q(x)}{p(x)} \, dx \\
  \geq -\log \int p(x) \frac{q(x)}{p(x)} \, dx = 0
  \]
  -log is strictly convex

- Holds for discrete or continuous variables
- Equality iff $q(x) = p(x)$ almost everywhere