CPSC 540: Machine Learning
Mixture Models, Expectation Maximization

Mark Schmidt
University of British Columbia
Winter 2017
Admin

- **Assignment 2:**
  - Due tonight.
  - 1 late day to hand it in Wednesday.
  - 2 late days to hand it in next Wednesday.

- **Class cancelled Wednesday:**
  - So you can go to the TensorFlow lecture at the same time (check website for location).

- **Assignment 3:**
  - Out later this week.
Last Time: Density Estimation

Last time we started discussing unsupervised task of density estimation.

- Given data $X$, estimate probability density $p(\hat{x}^i)$.

$$X = \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 1 & 0 & 1 \\
1 & 0 & 1 & 1 \\
\end{bmatrix}$$

$$\hat{X} = \begin{bmatrix}
\end{bmatrix}$$

- What is the probability of having $\hat{x}^i = [1 \ 1 \ 0 \ 0]$ in test data?

- A “master” ML problem that lets you solve many other problems...
Supervised Learning with Density Estimation

- Density estimation can be used for **supervised learning**:
  - 340 discussed **generative models** that model joint probability of $x^i$ and $y^i$,
    \[
P(y^i|x^i) \propto p(x^i, y^i) = p(x^i|y^i)p(y^i).
    \]

- Estimating $p(x^i, y^i)$ is a density estimation problem.
  - **Naive Bayes** models $p(x^i|y^i)$ as product of independent distributions.
  - **Linear discriminant analysis (LDA)** models $p(x^i|y^i)$ as a multivariate Gaussian.

- Generative models have been unpopular for a while, but are coming back:
  - Naive Bayes regression is being used for CRISPR gene editing.
  - Generative adversarial networks and variational autoencoders (deep learning).
  - We believe that most human learning is unsupervised.
The multivariate normal distribution models PDF of vector $x$ as

$$p(x|\mu, \Sigma) = \frac{1}{(2\pi)^{d/2} |\Sigma|^{1/2}} \exp \left( -\frac{1}{2} (x - \mu)^T \Sigma^{-1} (x - \mu) \right)$$

where $\mu \in \mathbb{R}^d$ and $\Sigma \in \mathbb{R}^{d \times d}$ and $\Sigma \succ 0$.

Closed-form MLE:

$$\mu = \frac{1}{n} \sum_{i=1}^{n} x^i, \quad \Sigma = \frac{1}{n} \sum_{i=1}^{N} \underbrace{(x^i - \mu)(x^i - \mu)^T}_{d \times d}.$$

Closed under several operations: products of PDFs, marginalization, conditioning.

Light-tailed: assumes all data is close to mean.

- Not robust to outliers or data far away from mean.
Outline

1. Mixture Models
2. Learning with Hidden Values
3. Expectation Maximization
1 Gaussian for Multi-Modal Data

- Major drawback of Gaussian is that it’s uni-modal.
- It gives a terrible fit to data like this:

- If Gaussians are all we know, how can we fit this data?
2 Gaussians for Multi-Modal Data

- We can fit this data by using **two Gaussians**

- Instead of assuming data comes from one Gaussian, we assume:
  - Half the time it comes from Gaussian 1.
  - Half the time it comes from Gaussian 2.
Our probability density in the previous example is given by

\[ p(x \mid \mu_1, \mu_2, \Sigma_1, \Sigma_2) = \frac{1}{2} p(x \mid \mu_1, \Sigma_1) + \frac{1}{2} p(x \mid \mu_2, \Sigma_2), \]

where \( p(x \mid \mu_c, \Sigma_c) \) is the PDF of a Gaussian.

If data comes from one Gaussian more often than the other, we could use

\[ p(x \mid \mu_1, \mu_2, \Sigma_1, \Sigma_2, \pi_1, \pi_2) = \pi_1 p(x \mid \mu_1, \Sigma_1) + \pi_2 p(x \mid \mu_2, \Sigma_2), \]

where \( \pi_1 + \pi_2 = 1 \) and both are non-negative.
Mixture of Gaussians

- If instead of 2 Gaussians we need $k$ Gaussians, our PDF would be

$$p(x \mid \mu, \Sigma, \pi) = \sum_{c=1}^{k} \pi_c p(x \mid \mu_c, \Sigma_c),$$

where $(\mu_c, \Sigma_c)$ are the parameters mixture/cluster $c$.

- To make the $\pi_c$ probabilities we need that $\pi_c \geq 0$ and $\sum_{c=1}^{k} \pi_c = 1$.

- This is called a mixture of Gaussians model.
  - We can use it to model complicated densities with Gaussians (like RBFs).
Mixture of Gaussians

- Gaussian vs. **Mixture of 4 Gaussians** for 2D multi-modal data:
Mixture of Gaussians

- **Gaussian vs. Mixture of 5 Gaussians** for 2D multi-modal data:
Mixture of Gaussians

How a mixture of Gaussian “generates” data:

1. Sample cluster $c$ based on prior probabilities $\pi_c$ (categorical distribution).
2. Sample example $x$ based on mean $\mu_c$ and covariance $\Sigma_c$.

We usually fit these models with expectation maximization (EM):

- EM is a general method for fitting models with hidden variables.
- For mixture of Gaussians: we treat cluster $c$ as a hidden variable.
Last Time: Independent vs. General Discrete Distributions

- We also considered density estimation with discrete variables,

\[ X = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}, \]

and considered two extreme approaches:

- **Product of independent Bernoullis:**
  
  \[ p(x|\theta) = \prod_{j=1}^{d} p(x_j|\theta_j). \]

  Easy to fit but strong **independence assumption:**
  
  - Knowing \( x_j \) tells you nothing about \( x_k \).

- **General discrete distribution:**
  
  \[ p(x|\theta) = \theta_x. \]

  No assumptions but **hard to fit:**
  
  - Parameter vector \( \theta_x \) for each possible \( x \).
Independent vs. General Discrete Distributions on Digits

- Consider handwritten images of digits:

\[ x^i = \text{vec} \begin{pmatrix} \vdots \\ -5 \\ -10 \\ -15 \\ -20 \\ -25 \\ \vdots \end{pmatrix} \]

so each row of \( X \) contains all pixels from one image of a 0, 1, 2, \ldots, 9.

- Previously we had labels and wanted to recognize that this is a 4.

- In density estimation we want \textit{probability distribution} over images of digits.

- Given an image, what is the probability that it's a digit?

- Sampling from the density should generate images of digits.
Independent vs. General Discrete Distributions on Digits

- We can visualize probabilities in independent Bernoulli model as an image:

![Image of independent Bernoulli model]

- Samples generated from independent Bernoulli model:

![Images of samples generated from independent Bernoulli model]

- This is clearly a terrible model: misses dependencies between pixels.
Independent vs. General Discrete Distributions on Digits

- Here is a sample from the MLE with a general discrete distribution:

![Sample Image](image1)

- Here is an image with a probability of 0:

![Image with Probability 0](image2)

- This model *memorized training images* and doesn’t generalize.
  - MLE puts probability at least $\frac{1}{n}$ on training images, and 0 on non-training images.

- A model lying between these extremes is the *mixture of Bernoullis*.
Mixture Models

Consider a coin flipping scenario where we have two coins:
- Coin 1 has $\theta_1 = 0.5$ (fair) and coin 2 has $\theta_2 = 1$ (biased).

Half the time we flip coin 1, and otherwise we flip coin 2:

$$p(x = 1|\theta_1, \theta_2) = \pi_1 p(x = 1|\theta_1) + \pi_2 p(x = 1|\theta_2)$$

$$= \frac{1}{2} \theta_1 + \frac{1}{2} \theta_2.$$

This mixture model is not very interesting:
- It's equivalent to flipping one coin with $\theta = 0.75$.

But this gets more interesting with multiple variables...
Mixture of Independent Bernoullis

Consider a mixture of independent Bernoullis:

\[
p(x \mid \theta_1, \theta_2) = \frac{1}{2} \prod_{j=1}^{d} p(x_j \mid \theta_{1j}) + \frac{1}{2} \prod_{j=1}^{d} p(x_j \mid \theta_{2j}).
\]

Conceptually, we now have two sets of coins:
- Half the time we throw the first set, half the time we throw the second set.

With \(d = 4\) we could have \(\theta_1 = [0 \ 0.7 \ 1 \ 1]\) and \(\theta_2 = [1 \ 0.7 \ 0.8 \ 0]\).

Have we gained anything?
- In the mixture of Bernoullis the variables are not independent:
  - In this example knowing \(x_1 = 1\) gives you the cluster, which gives you \(x_4 = 0\).
  - So we have dependencies: \(p(x_4 = 1 \mid x_1 = 1) \neq p(x_4 = 1)\).
Mixture of Independent Bernoullis

- General mixture of independent Bernoullis:
  \[
  p(x|\Theta) = \sum_{c=1}^{k} \pi_c p(x|\theta_c),
  \]
  where \( \Theta \) contains all the model parameters.

- Mixture of Bernoullis can model dependencies between variables
  - Individual Bernoullis act like clusters of the binary data.
  - Knowing cluster of one variable gives information about other variables.

- With \( k \) large enough, mixtures are sufficient to model any discrete distribution.
  - Possibly with \( k << 2^d \).
Mixture of Independent Bernoullis

- Plotting parameters $\theta_c$ with 10 mixtures trained on MNIST: digits.
  (hand-written images of the the numbers 0 through 9)

0.12 0.14 0.12 0.06 0.13

1 9 9 0 3

0.07 0.05 0.15 0.07 0.09

2 5 1 0 6

- Remember this is unsupervised: it hasn’t been told there are ten digits.
  - Density estimation tries to figure out how the world works.
Mixture of Independent Bernoullis

- Plotting parameters $\theta_c$ with 10 mixtures trained on MNIST: digits.
  (hand-written images of the numbers 0 through 9)

- You could use this model to “fill in” missing parts of an image:
  - By finding likely cluster/mixture, you find likely values for the missing parts.

Outline

1. Mixture Models
2. Learning with Hidden Values
3. Expectation Maximization
Learning with Hidden Values

- We often want to learn with unobserved/missing/hidden/latent values.
- For example, we could have a dataset like this:

\[
X = \begin{bmatrix}
N & 33 & 5 \\
F & 10 & 1 \\
F & ? & 2 \\
M & 22 & 0 \\
\end{bmatrix},
\]

\[
y = \begin{bmatrix}
-1 \\
+1 \\
-1 \\
? \\
\end{bmatrix}.
\]

- Missing values are very common in real datasets.
- An important issue to consider: why is data missing?
We’ll focus on data that is missing at random (MAR):
- The assumption that \( ? \) is missing does not depend on the missing value.

This definition doesn’t agree with intuitive notion of “random”:
- A variable that is always missing would be “missing at random”.
- The intuitive/stronger version is missing completely at random (MCAR).

Examples of MCAR and MAR for digit data:
- Missing random pixels/labels: MCAR.
- Hide the top half of every digit: MAR.
- Hide the labels of all the “2” examples: not MAR.

We’ll consider MAR, because otherwise you need to model why data is missing.
Consider a dataset with MAR values:

\[
X = \begin{bmatrix}
N & 33 & 5 \\
F & 10 & 1 \\
F & ? & 2 \\
M & 22 & 0
\end{bmatrix},
\quad y = \begin{bmatrix}
-1 \\
+1 \\
-1 \\
? 
\end{bmatrix}.
\]

**Imputation** method is one of the first things we might try:

1. **Initialization**: find parameters of a density model (often using “complete” examples).
2. **Imputation**: replace each ? with the most likely value.
3. **Estimation**: fit model with these imputed values.

You could also alternate between imputation and estimation.
Semi-Supervised Learning

- Important special case of MAR is semi-supervised learning.

\[ X = \begin{bmatrix} \vdots \end{bmatrix}, \quad y = \begin{bmatrix} \vdots \end{bmatrix}, \]
\[ \tilde{X} = \begin{bmatrix} \vdots \end{bmatrix}, \quad \tilde{y} = \begin{bmatrix} \vdots \end{bmatrix}. \]

- Motivation for training on labeled data \((X, y)\) and unlabeled data \(\tilde{X}\):
  - Getting labeled data is usually expensive, but unlabeled data is usually cheap.
Semi-Supervised Learning

- Important special case of MAR is **semi-supervised learning**.

\[
X = \begin{bmatrix} \end{bmatrix}, \quad y = \begin{bmatrix} \end{bmatrix},
\]

\[
\tilde{X} = \begin{bmatrix} \end{bmatrix}, \quad \tilde{y} = \begin{bmatrix} ? \end{bmatrix},
\]

- Imputation approach is called **self-taught learning**:
  - Alternate between guessing \( \hat{y} \) and fitting the model with these values.
To fit mixture models we often introduce $n$ MAR variables $z^i$.

Why???

Consider mixture of Gaussians, and let $z^i$ be the cluster number of example $i$:

- Given the $z^i$ it’s easy to optimize the means and variances $(\mu_c, \Sigma_c)$:
  - Fit a Gaussian to examples in cluster $i$.

- Given the $(\mu_c, \Sigma_c)$ it’s easy to optimize the clusters $z^i$:
  - Find the cluster with highest $p(x^i | \mu_c, \Sigma_c)$.

If all clusters have the same covariance $\Sigma_c = \Sigma$, this is k-means clustering.

- $\mu_c$ is the mean of cluster $c$ and $z^i$ is the nearest mean.
Outline

1. Mixture Models
2. Learning with Hidden Values
3. Expectation Maximization
Drawbacks of Imputation Approach

- The imputation approach to MAR variables is simple:
  - Use density estimator to "fill in" the missing values.
  - Now fit the "complete data" using a standard method.

- But "hard" assignments of missing values lead to propagation of errors.
  - What if cluster is ambiguous in k-means clustering?
  - What if label is ambiguous in "self-taught" learning?

- Ideally, we should use probabilities of different assignments ("soft" assignments):
  - If the MAR values are obvious, this will act like the imputation approach.
  - For ambiguous examples, takes into account probability of different assignments.
Expectation Maximization Notation

- **Expectation maximization (EM)** is an optimization algorithm for MAR values:
  - Applies to problems that are easy to solve with "complete" data (i.e., you knew $H$).
  - Allows probabilistic or "soft" assignments to MAR values.

- EM is the most cited paper in statistics?

- EM notation: we use $O$ as observed variables and $H$ as hidden variables.
  - Mixture models: observe data $O = \{X\}$ but don’t observe clusters $H = \{z_i\}_{i=1}^n$.
  - Semi-supervised learning: observe $O = \{X, y, \hat{X}\}$ but don’t observe $H = \{\hat{y}\}$.

- When we choose one $H$ by imputation it’s called “hard” EM.

- We use $\Theta$ as parameters we want to optimize.
Complete Data and Marginal Likelihoods

- Assume observing $H$ makes “complete” likelihood $p(O, H|\Theta)$ “nice”.
  - It has a closed-form MLE for Bernoulli or Gaussians, NLL convex for logistic, etc.

- From marginalization rule, likelihood of $O$ in terms of “complete” likelihood is

$$p(O|\Theta) = \sum_{H_1} \sum_{H_2} \cdots \sum_{H_m} p(O, H|\Theta) = \sum_H p(O, H|\Theta).$$

where we sum (or integrate) over all $H$ (the “marginal likelihood” of $O$).

- The marginal log-likelihood thus has the form

$$-\log p(O|\Theta) = -\log \left( \sum_H p(O, H|\Theta) \right),$$

which has a sum inside the log.

  - This does not preserve convexity: minimizing it is usually NP-hard.
Expectation Maximization Bound

- To compute $\Theta^{t+1}$, the approximation used by EM and hard-EM is
  \[
  - \log p(O|\Theta) = - \log \left( \sum_H p(O, H|\Theta) \right) \approx - \sum_H \alpha^t_H \log p(O, H|\Theta),
  \]
  where $\alpha^t_H$ is a weight for the assignment $H$ to the hidden variables.

- In hard-EM we set $\alpha^t_H = 1$ for the most likely $H$ given $\Theta^t$ (all other $\alpha^t_H = 0$).

- In soft-EM we set $\alpha^t_H = p(H|O, \Theta^t)$, weighting $H$ by probability given $\Theta^t$.

- We’ll show the EM approximation minimizes an upper bound,
  \[
  - \log p(O|\Theta) \leq - \sum_H p(H|O, \Theta^t) \log p(O, H|\Theta) + \text{const.},
  \]
  \[Q(\Theta|\Theta^t)\]
**Expectation Maximization as Bound Optimization**

- **Expectation maximization** is a bound-optimization method:
  - At each iteration we optimize a bound on the function.

- In gradient descent, our bound came from Lipschitz-continuity of the gradient.
- In EM, our **bound comes from expectation** over hidden variables (non-quadratic).
Expectation Maximization (EM)

- So EM starts with $\Theta^0$ and sets $\Theta^{t+1}$ to maximize $Q(\Theta|\Theta^t)$.

- This is typically written as two steps:
  1. **E-step**: Define expectation of complete log-likelihood given $\Theta^t$,  
     \[
     Q(\Theta|\Theta^t) = \sum_H p(H|O, \Theta^t) \log p(O, H|\Theta)
     \]
     fixed weight nice term
     
     which is a weighted version of the “nice” $\log p(O, H)$ values.
  2. **M-step**: Maximize this expectation,  
     \[
     \Theta^{t+1} = \arg\max_{\Theta} Q(\Theta|\Theta^t).
     \]
Convergence Properties of Expectation Maximization

- We’ll show that

\[
\log p(O|\Theta^{t+1}) - \log p(O|\Theta^t) \geq Q(\Theta^{t+1}|\Theta^t) - Q(\Theta^t|\Theta^t),
\]

that guaranteed progress is at least as large as difference in \( Q \).

- Does this imply convergence?
  - Yes, if likelihood is bounded above.

- Does this imply convergence to a stationary point?
  - No, although many papers say that it does.
    - Could have maximum of 3 and objective values of 1, 1 + 1/2, 1 + 1/2 + 1/4, …

- Almost nothing is known about rate of convergence.
Summary

- **Mixture models** write probability as convex combination of probabilities.
  - Model dependencies between variables even if components are independent.
  - Probability of belonging to mixtures is a soft-clustering of examples.

- **Missing at random**: fact that variable is missing does not depend on its value.

- **Semi-supervised learning**: learning with labeled and unlabeled data.

- **Expectation maximization**:
  - Optimization with MAR variables, when knowing MAR variables make problem easy.

- Next time: what “parts” make up your personality? (Beyond PCA)