Unsupervised Clustering and Latent Dirichlet Allocation

Mark Gales

Lent 2011

Machine Learning for Language Processing: Lecture 8

MPhil in Advanced Computer Science
Introduction

• So far described a number of models for word sequences
  – most common are based on $N$-grams and mixtures of $N$-grams

• In this lecture we will examine:
  – the application of $N$-grams (and extensions) to topic clustering;
  – an alternative generative model latent Dirichlet allocation

• The last slides will not be covered in the lectures - briefly mention
  – what happens as the number of clusters tends to infinity
  – infinite Gaussian mixture models
  – Dirichlet processes
Unsupervised Document Clustering

- Use a topic-dependent $N$-gram language model to perform clustering
  - word sequence $w = \{w_1, \ldots, w_N\}$
  - start, $w_0$, and end $w_{N+1}$ symbols added
  - $z$ indicator variable over topics $s_1, \ldots, s_K$
  - plate repeated for every document

- Training data fully observed (supervised training) standard $N$-gram training
  - **BUT** interested in **unsupervised clustering** - indicator variable $z$ unobserved

- Likelihood of one document with word sequence $w$ can be written as

$$P(w) = \sum_{k=1}^{K} P(s_k)P(w|s_k) = \sum_{k=1}^{K} P(s_k) \prod_{i=1}^{N+1} P(w_i|w_{i-1}, s_k)$$
Unsupervised Clustering

- The likelihood has been written as marginalising over the latent variable
  - standard mixture model - use EM BUT interested in clustering documents
- Rather than using the “soft” assignment in EM, use a hard assignment

\[ z_r^{[l]} = \arg\max_{s_k} \left\{ P(s_k | \lambda^{[l]}) P(w^{(r)} | s_k, \lambda^{[l]}) \right\} \]

  - compare to EM where at iteration \( l \) compute \( P(s_k | w, \lambda^{[l]}) \)
  - allows documents to be clustered together (unique label for each document)

For parameters of component \( s_k \):

\[ \lambda^{[l+1]}_k = \arg\max_{\lambda} \left\{ \prod_{r: z_r^{[l]} = s_k} P(w^{(r)} | \lambda) \right\} \]

- Iterative procedure (similar to Viterbi training) - example of K-means clustering
  - can initialise model parameters by using \( K \) randomly selected examples
Language Model Components

- For simplicity only consider a unigram language model - for BNs below
  - inner plate repeated for each word (start/end symbols ignored as unigram)
  - outer plate for each document

\[
\sum_{k=1}^{K} P(s_k) \prod_{i=1}^{N} P(w_i|s_k) \prod_{i=1}^{N} \left( \sum_{k=1}^{K} P(s_k)P(w_i|s_k) \right)
\]

- Interesting to contrast two forms of latent variable model
  - (left) indicator variable $z$ over space of language models
  - (right) indicator variable $z$ over space of language model predictions

- Possible to combine latent variable models (a hierarchical model)
Bayesian Approaches

• Consider a generative model for class $\omega_j$ (supervised training)
  – **training data**: $D = \{x_1 \ldots, x_n\}$
  – parametric form of distribution (the model), $M$, is known (and fixed) with (unknown) parameters $\theta$

• Rather than estimating the parameters of the model, $\hat{\theta}$, use a distribution
  – from training data obtain the **posterior distribution over model parameters**
    $$p(\theta|D, M) = \frac{p(D|\theta, M)p(\theta|M)}{p(D|M)}$$
    Note MAP $\hat{\theta} = \arg\max_\theta \{p(\theta|D, M)\}$
  – $p(\theta|M)$ is the **prior distribution** over the model parameters

• Likelihood of an observation $x$ then computed as
  $$p(x|D, M) = \int p(x|\theta, M)p(\theta|D, M)d\theta$$
  Note MAP $p(x|D, M) \approx p(x|\hat{\theta}, M)$
Distribution of the Mean Estimate

- Consider Bayesian estimation of the mean $\mu$ of a Gaussian distribution

- Posterior $p(\mu|\mathcal{D}, \mathcal{M})$ variation (from DHS)
  - Gaussian distributed - $\mu \sim \mathcal{N}(\hat{\mu}, \hat{\Sigma})$
  - prior $\mathcal{N}(0, \Sigma_p)$

\[
\hat{\mu} = \left( n\Sigma^{-1} + \Sigma_p^{-1} \right)^{-1} \left( \Sigma^{-1} \sum_{i=1}^{n} x_i \right)
\]

\[
\hat{\Sigma} = \left( n\Sigma^{-1} + \Sigma_p^{-1} \right)^{-1}
\]

- Shape of posterior distribution changes as $n$ increases
  - the posterior becomes more sharply peaked (reduced variance)
  - MAP estimate (the mode of the distribution) moves towards ML estimate
Latent Dirichlet Allocation

- Interested in applying Bayesian approaches to language processing
  - consider a mixture-of-unigrams language model

\[
P(w) = \prod_{i=1}^{N} \sum_{k=1}^{K} P(s_k) P(w_i | s_k)
\]

where \( P(s_k) \) is estimated from training data

- alternatively consider a Bayesian version over the topic priors

\[
P(w | \alpha) = \int p(\theta | \alpha) \left( \prod_{i=1}^{N} \sum_{k=1}^{K} P(s_k | \theta) P(w_i | s_k) \right) d\theta
\]

where \( p(\theta | \alpha) \) obtained from the training data

**What form of distribution/latent variable model to use?**

MPhil in Advanced Computer Science
(Reminder) Multinomial Distribution

- **Multinomial** distribution: \( x_i \in \{0, \ldots, n\} \)

\[
P(x|\theta) = \frac{n!}{\prod_{i=1}^{d} x_i!} \prod_{i=1}^{d} \theta_i^{x_i}, \quad n = \sum_{i=1}^{d} x_i, \quad \sum_{i=1}^{d} \theta_i = 1, \quad \theta_i \geq 0
\]

- When \( n = 1 \) the multinomial distribution simplifies to

\[
P(x|\theta) = \prod_{i=1}^{d} \theta_i^{x_i}, \quad \sum_{i=1}^{d} \theta_i = 1, \quad \theta_i \geq 0
\]

- a unigram language model with 1-of-V coding (\( d = V \) the vocabulary size)
- \( x_i \) indicates word \( i \) of the vocabulary observed, \( x_i = \begin{cases} 1, & \text{word } i \text{ observed} \\ 0, & \text{otherwise} \end{cases} \)
- \( \theta_i = P(w_i) \) the probability that word \( i \) is seen
(More) Probability Distributions

- **Dirichlet** (continuous) distribution with parameters $\alpha$

  $$p(x|\alpha) = \frac{\Gamma(\sum_{i=1}^{d} \alpha_i)}{\prod_{i=1}^{d} \Gamma(\alpha_i)} \prod_{i=1}^{d} x_i^{\alpha_i - 1}; \quad \text{for "observations": } \sum_{i=1}^{d} x_i = 1, \quad x_i \geq 0$$

  - $\Gamma()$ is the **Gamma distribution**
  - Conjugate prior to the multinomial distribution
    (form of posterior $p(\theta|D, M)$ is the same as the prior $p(\theta|M)$)

- **Poisson** (discrete) distribution with parameter $\xi$

  $$P(x|\xi) = \frac{\xi^x \exp(-\xi)}{x!}$$

  - probability of the number of events in a specific interval
  - here used for number of words in a document
Dirichlet Distribution Example

- Note: $x + y + z = 1$
- Vector: $(\alpha_1, \alpha_2, \alpha_3)$
Bayesian Network for Latent Dirichlet Allocation (LDA) is shown above

- **explicitly** includes dependence on model parameters \( \lambda = \{\alpha, \beta\} \)

\[
P(w|\alpha, \beta) = \int p(\theta|\alpha) \left( \prod_{i=1}^{N} \sum_{k=1}^{K} P(s_k|\theta) P(w_i|s_k, \beta) \right) d\theta
\]

- \( z \) is an indicator variable for one of the \( K \) **topics**: \( \{s_1, \ldots, s_K\} \)
- **inner plate** is repeated for \( N \) words, **outer plate** is repeated for \( R \) documents

**Bayesian approach** learn posterior distribution of the component priors, \( \theta \),

- Dirichlet distribution \( p(\theta|\alpha) = p(\theta|D, M) \), and noting \( P(s_k|\theta) = \theta_k \)
LDA Generative Process

- LDA assumes the following generative process for the words $w$ is a document
  1. Choose length of document - $N \sim \text{Poisson}(\xi)$
  2. Choose parameters of multinomial - $\theta \sim \text{Dir}(\alpha)$
  3. For each of the $N$ words $w_n$:
     (a) Choose topic: $z_n \sim \text{Multinomial}(\theta)$
     (b) Choose word: $w_n$ from multinomial probability conditioned on topic $z_n$ with parameters $\beta$

- The parameters that need to be estimated for LDA
  - $\alpha = \{\alpha_1, \ldots, \alpha_K\}$: $K$ parameters
    the prior distribution over the multinomial parameters
  - $\beta = \{\beta_{11}, \beta_{1V}, \ldots, \beta_{K1}, \ldots, \beta_{KV}\}$: $KV$ parameters
    Note $\beta_{ki} \geq 0$, $\sum_{i=1}^{V} \beta_{ki} = 1 \ \forall k, i$ - this is the equivalent of topic-unigrams
LDA Parameter Estimation

- Given corpus of documents $\{w^{(1)}, \ldots, w^{(R)}\}$ need to estimate $\alpha, \beta$

$$L(\alpha, \beta) = \sum_{r=1}^{R} \log \left( P(w^{(r)}|\alpha, \beta) \right)$$

- Unfortunately likelihood calculation is intractable need to compute

$$P(w|\alpha, \beta) = \frac{\Gamma(\sum_{k=1}^{K} \alpha_k)}{\prod_{k=1}^{K} \Gamma(\alpha_k)} \int \left( \prod_{k=1}^{K} \theta_k^{\alpha_k - 1} \right) \left( \prod_{i=1}^{N} \sum_{k=1}^{K} \theta_k \prod_{j=1}^{V} (\beta_{kj})^{I(w_i,j)} \right) d\theta$$

- word indicator: $I(w_i, j) = \begin{cases} 1, & w_i = \text{word } j \text{ in the vocabulary} \\ 0, & \text{otherwise} \end{cases}$

- $P(s_k|\theta) = \theta_k$ and $P(w_i|s_k, \beta) = \beta_{ki}$

- Not possible to use EM: require $p(\theta, z|w, \alpha, \beta) = \frac{p(\theta, z, w|\alpha, \beta)}{P(w|\alpha, \beta)}$
Variational EM (Reference)

Latent Dirichlet Allocation Variational Approximation

- LDA can be estimated using variational EM with the mean-field approximation
  - use a variational approximation \( q(\theta, z|\gamma, \phi) \) - see diagram on right

\[
q(\theta, z|\gamma, \phi) = q(\theta|\gamma) \prod_{i=1}^{N} q(z_i|\phi_i)
\]

- parameters - minimise KL-divergence: 
  \[
  \text{KL}(q()||p()) = \int p(x) \log \left( \frac{q()}{p()} \right) dx
  \]

\[
\{ \gamma^{[l]}, \phi^{[l]} \} = \arg\min_{\gamma, \phi} \left\{ \text{KL}(q(\theta, z|\gamma, \phi)||p(\theta, z|w, \alpha^{[l]}, \beta^{[l]}) ) \right\}
\]
LDA and Topic Mixture of Unigrams

- **Latent Dirichlet allocation** - parameters $K(1 + V)$ - continuous mixture

\[
P(w|\alpha, \beta) = \frac{\Gamma(\sum_{k=1}^{K} \alpha_k)}{\prod_{k=1}^{K} \Gamma(\alpha_k)} \int \left( \prod_{k=1}^{K} \theta_k^{\alpha_k-1} \right) \left( \prod_{i=1}^{N} \sum_{k=1}^{K} \theta_k \prod_{j=1}^{V} (\beta_{kj})^I(w_{i,j}) \right) d\theta
\]

- **Topic mixture of unigrams** - parameters $M + K(M + V)$ - discrete mixture

\[
P(w|\alpha, \beta, \theta) = \sum_{m=1}^{M} \alpha_m \left( \prod_{i=1}^{N} \sum_{k=1}^{K} \theta_{mk} \prod_{j=1}^{V} (\beta_{kj})^I(w_{i,j}) \right)
\]
Properties of LDA

• LDA is a **generative model** of a document
  
  – **compact** model of the data
  – infinite component priors represented by $K$-parameter distribution $p(\theta|\alpha)$
  – can be combined with standard language model smoothing for $\beta$

• Consider using LDA as a generative model for classification for
  
  – for each class $\omega_j$ estimate $\{\alpha^{(j)}, \beta^{(j)}\}$ using all documents from class $\omega_j$
  – estimate the prior for each class $P(\omega_j)$
  – perform classification for sequence $w$ based on

\[
\hat{\omega} = \arg\max_{\omega_j} \left\{ P(\omega_j)P(w|\alpha^{(j)}, \beta^{(j)}) \right\}
\]

• LDA has also been used for a range of language processing applications
How Many Topics?

• So far not consider the number of topics, $K$, for LDA
  – how about using a Bayesian approach

$$P(w|\alpha^{(1)}, \ldots, \alpha^{(\infty)}) =$$

$$\sum_{K=1}^{\infty} P(K) \int p(\theta^{(K)}|\alpha^{(K)}) \left( \prod_{i=1}^{N} \sum_{k=1}^{K} P(s_k|\theta^{(K)}) P(w_i|s_k, \beta) \right) d\theta^{(K)}$$

  – each of the priors of infinite mixture models has a Dirichlet distribution

• There’s a infinite number of components
  – unfortunately an infinite number of parameters $\alpha^{(1)}, \ldots, \alpha^{(\infty)}, \beta$ to train

  Can we keep the infinite model, but make it tractable?

• Non-parametric Bayesian approaches: (hierarchical) Dirichlet Processes
Gaussian Mixture Models

• Consider simpler (illustrative) example - the Infinite Gaussian Mixture Model

• Standard form of $M$-component Gaussian Mixture Model (GMM) is

$$p(x|\theta, \beta) = \sum_{m=1}^{M} P(c_m|\theta)p(x|c_m, \beta) = \sum_{m=1}^{M} P(c_m|\theta)N(x; \mu_m, \Sigma_m)$$

**Interested in what happens as $M \to \infty$?**

• Must use Bayesian approaches as the number of parameters infinite
  
  – what sort of prior distributions to use?

• Introduce prior distributions $\{\alpha_0, \beta\}$
  
  – $\alpha_0$ - prior parameter for the Dirichlet distribution
  – $\beta$ - prior distribution for Gaussian components
Infinite Gaussian Mixture Models

- From the Bayesian network above

\[
p(x_1, \ldots, x_N | \alpha_0, G_0) = \int \int p(\theta | \alpha_0) p(\beta | G_0) \prod_{i=1}^{N} \sum_{m=1}^{M} P(c_m | \theta) p(x_i | c_m, \beta) d\theta d\beta
\]

where: \( \theta | \alpha_0 \sim \text{Dirichlet} \left( \frac{\alpha_0}{M}, \ldots, \frac{\alpha_0}{M} \right) \); \( \beta_m \sim G_0 \); \( c_m | \theta \sim \text{Multinomial}(\theta) \)

- Estimate the hyper-parameters from training data, \( \{x_1, \ldots, x_N\} \) - maximise

\[
\mathcal{L}(\alpha_0, G_0) = \log \left( p(x_1, \ldots, x_N | \alpha_0, G_0) \right)
\]
Sample-Based Approximations

• Simple approach to approximate integrals is to use

\[ \int f(\mathbf{x})p(\mathbf{x}|\theta) d\mathbf{x} \approx \frac{1}{N} \sum_{i=1}^{N} f(\mathbf{x}^{(i)}); \quad \mathbf{x}^{(i)} \sim p(\theta) \]

- as \( N \to \infty \) the approximation will become an equality
- \( N \) needs to increase as dimension \( \mathbf{x} \) increases - need to sample the space

marginalising is simply sampling

• If a sample can’t be directly generated from the multivariate distribution \( p(\theta) \)
  - Gibbs sampling from conditional distributions can be used
  - assume that we have samples \( x_1^{(i)}, \ldots, x_{k-1}^{(i)}, x_{k+1}^{(i)}, x_d^{(i)} \) generate \( x_k^{(i)} \)
  - sample from

\[ p(x_k|x_1, \ldots, x_{k-1}, x_{k+1}, x_d, \theta) \]

- assumes that possible to sample from the conditional
Gaussian Mixture Model Sampling

\[ p(x|\theta, \beta) = \sum_{m=1}^{M} P(c_m|\theta)p(x|c_m, \beta) \]
\[ = \sum_{m=1}^{M} P(c_m|\theta)p(x|\beta_m) \]

- Sampling approach from distribution comprises
  1. Generate component indicator \( z_n \sim \text{Multinomial}(\theta) \)
  2. Generate observation: \( x_n \sim \mathcal{N}(\beta_{z_n}) \)

- Simple to train using EM (see lecture 5)
  - non-Bayesian - point estimates of the model parameters \( \{\theta, \beta\} \)
  - number of components \( M \) fixed
IGMM Sampling Procedure

How to generate samples from infinite components?

• **Gibb’s Sampling** process to generate \( \{x_1, \ldots, x_N\} \) for \( N \) samples

1. Generate component indicator \( z_n | z_{-n} \) \((z_{-n} = \{z_1, \ldots, z_{n-1}\})\)

\[
P(z_n = c_j | z_{-n}, \alpha_0) = \begin{cases} 
\frac{\sum_{i=1}^{n-1} 1(z_i, c_j)}{n-1+\alpha_0} & c_j \text{ represented} \\
\frac{\alpha_0}{n-1+\alpha_0} & c_j \text{ unrepresented}
\end{cases}
\]

2. If component indicted by \( z_n \) is unrepresented: \( \beta_{z_n} \sim G_0 \)

3. Generate observation: \( x_n \sim N(\beta_{z_n}) \)

• At most \( N \) of the infinite possible samples represented
**IGMM Hyper-Parameter Training**

- Using Gibb's sampling to training hyper-parameters of $G_0$
  - sampling process to generate $\{z^{(l)}, \beta^{(l)}\}$ for these $N$ samples, $\{x_1, \ldots, x_N\}$

1. Generate component indicators $z^{(l)}|z_{-n}, \beta^{(l-1)}, x_n$ (dropped dependence)

   \[
   P(z^{(l)}_n = c_j|\alpha_0^{(l-1)}, G_0^{(l-1)}) \propto \begin{cases} 
   \frac{\sum_{i=1}^{n-1} \mathbf{1}(z^{(l)}_i, c_j)}{n-1+\alpha_0^{(l-1)}} p(x_n|\beta_j^{(l-1)}) & \text{c}_j \text{ represented} \\
   \frac{\alpha_0^{(l-1)}}{n-1+\alpha_0^{(l-1)}} \int p(x_n|\beta)p(\beta|G_0^{(l-1)})d\beta & \text{c}_j \text{ unrepresented}
   \end{cases}
   \]

2. Foreach represented component $c_j, j \in \{1, \ldots, k_{\text{rep}}\}$
   - sample component mean and variance: $\beta_j^{(l)} = \{\mu_j^{(l)}, \Sigma_j^{(l)}\} \sim G_0^{(l-1)}$

3. Update hyper-parameters $\{\alpha_0^{(l)}, G_0^{(l)}\}$ using component values $\beta_1^{(l)}, \ldots, \beta_{k_{\text{rep}}}^{(l)}$
   (a) increment the counter $l = l + 1$
IGMM Classification

- So how can we perform classification - need the class-likelihood (prior simple)
  - consider observation $\mathbf{x}$ given training data for class $\omega_j$: $\mathcal{D} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$

$$p(\mathbf{x}|\mathcal{D}, \alpha_0, G_0) = \frac{p(\mathbf{x}, \mathcal{D}|\alpha_0, G_0)}{p(\mathcal{D}|\alpha_0, G_0)} = \frac{p(\mathbf{x}, \mathbf{x}_1, \ldots, \mathbf{x}_N|\alpha_0, G_0)}{p(\mathbf{x}_1, \ldots, \mathbf{x}_N|\alpha_0, G_0)}$$

  - clearly a non-parametric model - explicit dependence on training observations

- Use a sample-based approximations for numerator/denominator thus

$$p(\mathbf{x}_1, \ldots, \mathbf{x}_N|\alpha_0, G_0) \approx \frac{1}{L} \sum_{l=1}^{L} \prod_{i=1}^{N} p(\mathbf{x}_i|z^{(l)}, \beta^{(l)})$$

  - follow hyper-parameter training without update to hyper-parameters
  - similar for $p(\mathbf{x}, \mathbf{x}_1, \ldots, \mathbf{x}_N|\alpha_0, G_0)$
Dirichlet Processes

• Dirichlet Processes are a generalisation of the Dirichlet distribution
  – both can be viewed as distributions over distributions
  – **BUT** Dirichlet processes act over infinite components

• Model has the form
  \[ G \sim DP(\alpha_0, G_0); \]
  – \( G_0 \) is the **base measure** (distribution)
  – \( \alpha_0 \) is the **concentration parameter**

• If the measure is parametrised with \( \theta \)
  – each **draw** of \( G \) from \( G_0 \) yields \( \theta_k \sim G_0 \)
  – \( \delta_{\theta_k} \) indicates a \( \delta \) function at the parameters for draw \( k \), \( \theta_k \)
  – **Reminder:**
    \[ \int f(x|\theta)\delta_{\theta_k} d\theta = f(x|\theta_k) \]
The likelihood of the word sequence \( w = \{w_1, \ldots, w_N\} \) can be expressed as

\[
P(w|\alpha_0, G_0) = \int P(G|\alpha_0, G_0) \int P(\beta) \int p(\theta|G) P(w|\theta, G, \beta) d\theta d\beta dG
\]

- \( G \) is distributed according to the Dirichlet Process \( \text{DP}(\alpha_0, G_0) \)
- if \( K \) is the number of components associated with the \( G \)

\[
P(w|\theta, G, \beta) = \prod_{i=1}^{N} \sum_{k=1}^{K} P(s_k|\theta) P(w_i|s_k, \beta)
\]

- **BUT** can’t share cluster parameters \((\beta)\) across different draws
  - no relationship between clusters ... **hierarchical Dirichlet priors**
Dirichlet Processes Generative Process

- Can’t directly sample from Dirichlet process - use Gibb’s sampling
  - behaviour of $\theta_n$ given previous $n - 1$ draw $\theta_1, \ldots, \theta_{n-1}$

\[
\theta_n | \theta_1, \ldots, \theta_{n-1}, \alpha_0, G_0 \sim \frac{\alpha_0}{n - 1 + \alpha_0} G_0 + \sum_{i=1}^{n-1} \frac{1}{n - 1 + \alpha_0} \delta \theta_i
\]

- this is the equivalent of the generative process where

\[
\theta_n = \begin{cases} 
\theta_i & \text{with probability } \frac{1}{n - 1 + \alpha_0} \text{ for } 1 \leq i \leq (n - 1) \\
\theta & \text{with probability } \frac{\alpha_0}{n - 1 + \alpha_0} 
\end{cases}
\]

- A draw from a Dirichlet process (stick-breaking representation)

\[
G = \sum_{k=1}^{\infty} \pi_k \delta \theta_k; \quad \theta_k \sim G_0; \quad \psi_k \sim \text{Beta}(1, \alpha_0); \quad \pi_k = \psi_k \prod_{i=1}^{k-1} (1 - \psi_i)
\]

- Google Chinese Restaurant Process for a simple example