2
Análise Probabilística de Semântica Latente

A *Análise Probabilística de Semântica Latente* tem por objetivo o aprendizado de um modelo estatístico de misturas gerado através de uma ou mais variáveis não observáveis. Um exemplo deste tipo de modelo é apresentado na figura 2.1. Neste um conjunto de exemplos é gerado através de duas distribuições normais pelo processo de seleção primeiramente de uma das duas distribuições, e, em seguida, escolha de um exemplo coerente com esta. Uma vez gerados os dados, a distribuição com que foram escolhidas as normais é uma variável não observável, sendo necessários métodos de aprendizado para que esta variável seja examinada. Com este objetivo o *PLSA* maximiza a função de verossimilhança de forma a identificar as distribuições de probabilidade das variáveis ocultas, utilizando para isto o algoritmo *Expectation Maximization* (Dempster et al., 1977). Devido ao fato do *PLSA* ser baseado no *EM* é necessário compreendê-lo para estudarmos o *PLSA*, por este motivo explicaremos o funcionamento do *EM* a seguir.

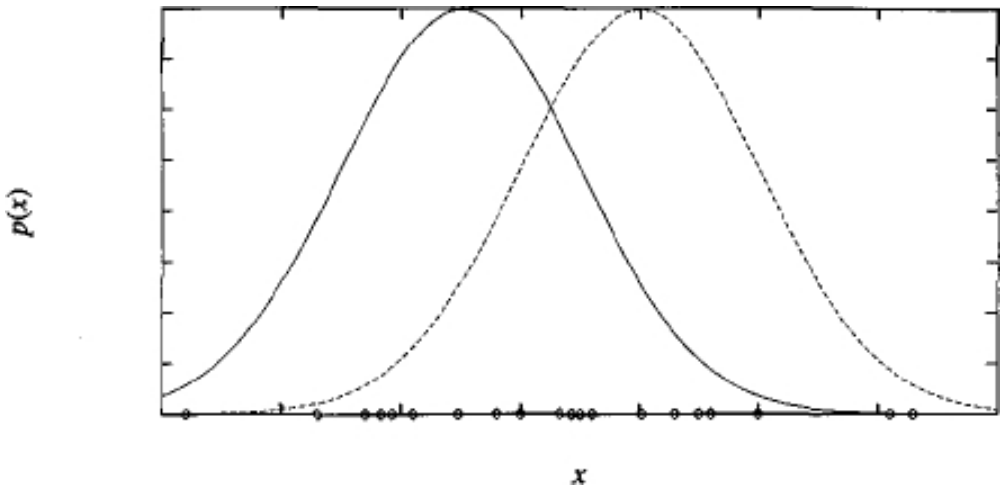

Figura 2.1 – Modelo de misturas gerado por duas distribuições normais (Mitchell, 1997)

2.1. Expectation Maximization

Em aprendizado de máquina estamos constantemente interessados em descobrir a melhor hipótese de um espaço de hipóteses *H*, dado conjunto de dados observados *D*. Uma forma de atingir este objetivo é procurar pela *hipótese mais provável*, contudo para isto é necessário calcular a probabilidade das hipóteses específicas $P(h|D)$. Uma forma simples de identificar este valor é utilizar a *Teorema de Bayes* como mostrado na equação 2.1, pois este calcula a probabilidade a posteriori $P(h|D)$ através das probabilidades a priori das hipóteses $P(h)$, juntamente com $P(D)$ e $P(D|h)$, que representam respectivamente a probabilidade a priori conjunto de dados *D* ser observado e a chance do mesmo ser observado dada a hipótese *h*.

$$P(h \mid D) = \frac{P(D \mid h)P(h)}{P(D)} \tag{2.1}$$

Desta forma são identificadas as probabilidades das hipóteses dado o conjunto de treino *D*, porém gostaríamos de encontrar a hipótese mais provável $h \in H$, ou seja, a hipótese com probabilidade *máxima a posteriori (MAP)*. Esta pode ser obtida através do Teorema de Bayes como é mostrado na equação 2.2.

$$\begin{aligned} h_{MAP} &\equiv \arg\max_{h \in H} P(h \mid D) \\ &= \arg\max_{h \in H} \frac{P(D \mid h)P(h)}{P(D)} \\ &= \arg\max_{h \in H} P(D \mid h)P(h) \end{aligned} \tag{2.2}$$

Considerando que todas as hipóteses *h* pertencentes a *H* e têm a mesma probabilidade, a hipótese mais provável pode ser encontrada através da simplificação da equação 2.2 removendo o termo $P(h)$. Com isto a expressão depende apenas do termo $P(D|h)$ conhecido também como verossimilhança, desta forma qualquer hipótese que maximize a $P(D|h)$ é chamada de hipótese de *máxima verossimilhança* (equação 2.3).

$$h_{ML} = \arg\max_{h \in H} P(D \mid h) \tag{2.3}$$

De forma análoga podemos encontra a hipótese de máxima verossimilhança para domínios em que as probabilidades sejam representadas como variáveis contínuas utilizando a função de *densidade* de probabilidade *p* ao invés da probabilidade discreta *P* (equação 2.4). Além desta variação uma forma muito utilizada para maximizar a verossimilhança é a maximização de seu logaritmo (equação 2.5), pois como o log de uma função de densidade de probabilidade é monotônica, maximizá-la também maximiza seu argumento.

$$h_{ML} = \arg\max_{h \in H} p(D \mid h) \tag{2.4}$$

$$h_{ML} = \arg\max_{h \in H} \ln p(D \mid h) \tag{2.5}$$

Tendo visto alguns conceitos podemos analisar o funcionamento do algoritmo *Expectation Maximization*. Este tem como objetivo estimar um conjunto de parâmetros Θ que definem a distribuição de probabilidades de um conjunto de dados não observáveis *Z*. Um exemplo seria estimar as médias de uma mistura de duas normais $\theta = \langle \mu_1, \mu_2 \rangle$ como foi apresentado anteriormente na figura 2.1. Considere a seguinte notação, *X* representa o conjunto de dados observáveis, ou seja, os valores gerados pela mistura e *Z* o conjunto de dados não observáveis, neste caso as escolhas tomadas entre uma das duas normais em cada dado selecionado. Considere também $Y = X \cup Z$ como sendo o conjunto de dados completo, a hipótese *h* como sendo um valor atribuído ao conjunto de parâmetros Θ e *h'* a hipótese revisada por uma iteração do algoritmo *EM*.

O algoritmo *Expectation Maximization* estima os parâmetros Θ através da busca pela hipótese *h'* que maximiza a expressão 2.6. Nesta $P(Y \mid h')$ representa a verossimilhança dos dados completos *Y* à hipótese revisada *h'*, e, por facilidade, é maximizado o logaritmo desta função, já que a maximização do logaritmo de uma distribuição de probabilidades também maximiza a própria função. Além disto, é necessária a utilização de uma esperança, pois como *Z* é gerado pela distribuição de probabilidades de sua mistura esta é uma variável aleatória, tornando sua união com os dados observados ($Y = X \cup Z$) também uma variável aleatória.

$$E[\ln P(Y \mid h')] \tag{2.6}$$

Para encontrar a hipótese que maximiza a expressão 2.6 é necessário estimar a distribuição de probabilidades de *Y,* porém, esta é dependente do parâmetro Θ que não é conhecido. Para resolver tal conflito o *EM* utiliza a hipótese atual *h* como valor do parâmetro Θ*,* de forma a calcular a expressão 2.6 em função de *h'* assumindo a hipótese *h* e também os dados observáveis *X*. Esta transição é representada pela função $Q(h' \mid h)$ mostrada na equação 2.7.

$$Q(h' \mid h) = E[\ln p(Y \mid h') \mid h, X] \tag{2.7}$$

Assim o algoritmo *EM* na sua forma geral repete dois passos até convergir: o passo de Estimativa (*E*), onde a função $Q(h' \mid h)$ é calculada utilizando a hipótese *h* e os dados observados *X*, estimando desta maneira a distribuição de probabilidade de *Y* (expressão 2.8), e o passo maximização (*M*), que substitui a hipótese *h* pela hipótese *h'* que maximiza a função $Q(h' \mid h)$, este passo é representado pela expressão 2.9. Através deste processo quando a função $Q(h' \mid h)$ é contínua o algoritmo converge para um ponto estacionário da função de verossimilhança *P*(*Y*|*h'*), porém a convergência do algoritmo só garante a obtenção de máximos locais. A prova de convergência do algoritmo *EM* não será mostrada neste trabalho podendo ser encontrada em (Dempster, 1977).

$$Q(h' \mid h) \leftarrow E[\ln p(Y \mid h') \mid h, X] \tag{2.8}$$

$$h \leftarrow \arg\max_{h'} Q(h' \mid h) \tag{2.9}$$

2.2. Gaussian Probabilistic Latent Semantic Analysis

Nesta seção será apresentada uma das variantes do *PLSA,* o *Gaussian Probabilistic Latent Semantic Analysis,* que é uma extensão do *PLSA* original para suportar variáveis contínuas. Porém, para o entendimento deste modelo, será necessário estudar alguns conceitos que veremos a seguir.

Quando representamos um problema através de probabilidades é necessária a utilização de variáveis aleatórias. Estas podem ser entendidas como elementos do mundo cujo status é desconhecido, tendo assim um domínio de valores com que ela pode assumir com uma determinada probabilidade, por exemplo, uma variável aleatória que determina se irá chover hoje pode assumir os valores {*verdadeiro*, *falso*} com 70% de probabilidade de ser verdadeira e 30% de ser falsa. Quando desejamos representar todas as variáveis aleatórias dentro de um determinado domínio, relacionando suas probabilidades, geralmente utilizamos tabelas de distribuição conjunta também conhecida como tabela de conjunção de probabilidades. Porém, o calculo das probabilidades de uma tabela de conjunção de probabilidade pode ser inviável para problemas grandes.

Uma *Rede Bayesiana* é uma representação compacta da tabela de conjunção de probabilidades através de tabelas de probabilidades condicionais. Tal modelo é apresentado com um grafo orientado acíclico (*DAG*), onde os nós representam variáveis aleatórias e as arestas as influencias dos nós sobre as variáveis. Se houver uma aresta do nó *X* até o nó *Y*, *X* será denominado *pai* de *Y*. Cada nó X_i possui uma distribuição de probabilidade condicional $P(X_i \mid Pais(X_i))$ que quantifica o efeito dos pais sobre o nó X_i. Tal representação só é possível se assumirmos a hipótese de independência condicional (expressão 2.10), que define que duas variáveis aleatórias *X* e *Y* são independentes dada uma terceira variável aleatória *Z*. Assumindo esta hipótese conseguimos representar a tabela de conjunção de probabilidades através do produto de suas tabelas de probabilidade condicional, como é mostrado na expressão 2.11.

$$P(X,Y \mid Z) = P(X \mid Z)P(Y \mid Z) \tag{2.10}$$

$$P(X_1,...,X_n) = \prod_i P(X_i \mid pais(X_i)) \tag{2.11}$$

Existem diversos modelos de rede bayesianas voltados para sistemas de recomendação, dentre estes se encontra o *gPLSA* (*Gaussian Probabilistic Latent Semantic Analysis*), mostrado na figura 2.2. Como dito anteriormente, este modelo assume a existência de comunidades de usuários (*Z*) onde cada comunidade atribui notas aos itens segundo o comportamento de uma gaussiana. Neste *U* representa os usuários, *Y* os itens e *V* o valor das avaliações, sendo que estes elementos se relacionam através do comportamento das notas dadas pelas comunidades de usuários aos itens, representados pelos dois parâmetros das gaussianas, a média (*Mzy*) e a variância (*Vzy*). Tais parâmetros não são variáveis aleatórias sendo, portanto, são apresentadas no diagrama como uma elipse com uma borda dupla.

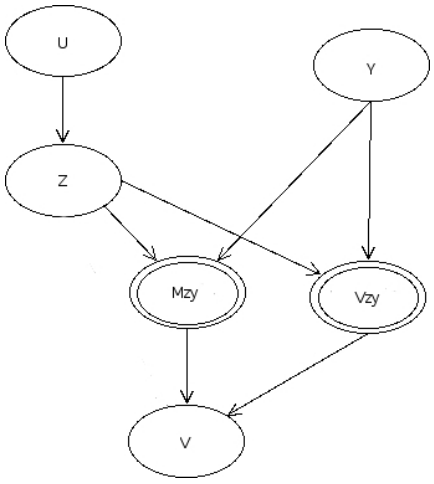

Figura 2.2 – Modelo do *gPLSA*

O modelo do *gPLSA* é expressado analiticamente através da equação 2.12, onde a função *f* representa a probabilidade de ocorrer o valor *v* dados os parâmetros da gaussiana (equação 2.13). Porém, para que a recomendação de novos itens seja possível, é necessário aprender o conjunto de parâmetros $\theta = (A, M, \Sigma)$ através dos dados observados $T = \{(u_i, y_i, v_i)\}_{i=1}^{n}$, onde $A = p(z \mid u)$, $M = \mu_{ZY}$ e $\Sigma = \sigma_{ZY}$. Com o intuito de aprender tais parâmetros é utilizado o algoritmo *EM*.

$$p(u, v, y, z) = p(u)\,p(y)\,p(z \mid u)\,f(v \mid \mu_{ZY}, \sigma_{ZY}) \tag{2.12}$$

$$f(v \mid \mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}(v-\mu)^2} \tag{2.13}$$

Para aplicarmos o EM no modelo do PLSA é necessário obter uma equação para função $Q(h' \mid h)$ e em seguida encontrar os parâmetros que a maximizam. Aplicando a equação 2.6 ao contexto do PLSA encontramos com a equação 2.14, onde o conjunto completo de dados $Y = (u, y, v, z)$.

$$Q(\theta' \mid \theta) = E[\ln \Pi_{i=1}^{n} p(u_i, y_i, v_i, z_i \mid \theta') \mid \theta, T] \tag{2.14}$$

Substituindo a equação 2.12 na 2.14 e efetuando o devido algebrismo obtemos:

$$\begin{aligned} Q(\theta' \mid \theta) &= \sum_{i=1}^{n} [\ln(p(u_i)) + \ln(p(y_i))] \\ &+ \sum_{i=1}^{n} \sum_{Z} p(z \mid u_i, y_i, v_i, \theta') \ln(p(z \mid u)) \\ &- \sum_{i=1}^{n} \sum_{Z} p(z \mid u_i, y_i, v_i, \theta') [\tfrac{1}{2} \ln(2\pi) + \ln(\sigma_{ZY})] \\ &- \sum_{i=1}^{n} \sum_{Z} p(z \mid u_i, y_i, v_i, \theta') \left[\frac{1}{2} \left(\frac{v_i - \mu_{ZY}}{\sigma_{ZY}} \right) \right] \end{aligned} \tag{2.15}$$

Nesta através da definição de probabilidade condicional temos:

$$\begin{aligned} p(z \mid u_i, y_i, v_i, \theta') &= \frac{p(u_i, y_i, v_i, z \mid \theta')}{p(u_i, y_i, v_i \mid \theta')} \\ p(z \mid u_i, y_i, v_i, \theta') &= \frac{p(u_i) p(y_i) p(z' \mid u_i) f(v_i \mid \mu'_{ZY}, \sigma'_{ZY})}{\sum_{Z} p(u_i) p(y_i) p(z' \mid u_i) f(v_i \mid \mu'_{ZY}, \sigma'_{ZY})} \\ p(z \mid u_i, y_i, v_i, \theta') &= \frac{p(z' \mid u_i) f(v_i \mid \mu'_{ZY}, \sigma'_{ZY})}{\sum_{Z} p(z' \mid u_i) f(v_i \mid \mu'_{ZY}, \sigma'_{ZY})} \end{aligned} \tag{2.16}$$

Com isto conseguimos uma fórmula para calcular a equação $Q(\theta' \mid \theta)$, porém ainda precisamos encontrar a hipótese que maximiza esta. Para isto é utilizado o lagrangeano sobre esta função com as restrições da equação 2.17, resultando na equação 2.18:

$$\sum_U p(z \mid u) = 1$$
$$\sum_Y p(y) = 1 \qquad (2.17)$$
$$\sum_Y p(u) = 1$$

$$\begin{aligned}
&\Lambda(\theta, \lambda) = \textstyle\sum_{i=1}^{n} [\ln(p(u_i)) + \ln(p(y_i))] \\
&+ \textstyle\sum_{i=1}^{n} \sum_Z p(z \mid u_i, y_i, v_i, \theta') \ln(p(z \mid u)) \\
&- \textstyle\sum_{i=1}^{n} \sum_Z p(z \mid u_i, y_i, v_i, \theta')[\tfrac{1}{2} \ln(2\pi) + \ln(\sigma_{ZY})] \\
&- \textstyle\sum_{i=1}^{n} \sum_Z p(z \mid u_i, y_i, v_i, \theta') \left[\frac{1}{2} \left(\frac{v_i - \mu_{ZY}}{\sigma_{ZY}} \right) \right] \\
&+ \lambda_1 \left(\textstyle\sum_U p(u) - 1 \right) + \lambda_2 \left(\textstyle\sum_Y p(y) - 1 \right) + \textstyle\sum_u \rho_u \left(\textstyle\sum_Z p(z \mid u) - 1 \right)
\end{aligned} \qquad (2.18)$$

Aplicando as derivadas parciais sobre os parâmetros $P(u \mid z)$, μ_{ZY} e σ_{ZY} e isolando os termos encontramos as equações 2.19, 2.20 e 2.21.

$$p(z \mid u) = \frac{\sum_U p(z \mid u_i, y_i, v_i, \theta')}{\sum_U \sum_Z p(z \mid u_i, y_i, v_i, \theta')} \qquad (2.19)$$

$$\mu_{ZY} = \frac{\sum_{yi=y} v_i p(z \mid u_i, y_i, v_i, \theta')}{\sum_{yi=y} p(z \mid u_i, y_i, v_i, \theta')} \qquad (2.20)$$

$$\sigma_{ZY} = \frac{\sum_{yi=y} p(z \mid u_i, y_i, v_i, \theta')(v_i - \mu_{ZY})}{\sum_{yi=y} p(z \mid u_i, y_i, v_i, \theta')} \qquad (2.21)$$

De forma resumida o algoritmo *gPLSA* calcula iterativamente a formula 2.6 (Passo E) tomando como hipótese os parâmetros calculados na iteração anterior, e, utiliza o valor gerado para calcular as equações 2.19, 2.20 e 2.21 (Passo M), repetindo o processo até convergir. Com o modelo treinado construímos um preditor simples através da combinação convexa das médias do item para cada grupo ponderado pela probabilidade do usuário pertencer a cada um deles, como é mostrado na equação 2.22. Podemos observar que a complexidade computacional deste preditor depende somente da quantidade de comunidades de usuários (*Z*),

onde tal parâmetro é fixado para que possa ser efetuado o treinamento, como conseqüência disto, temos a predição em tempo constante.

$$V(u, y) = \sum_Z p(z \mid u)\mu_{YZ} \tag{2.22}$$

O fato de ser possível construir um preditor de complexidade computacional constante através dos parâmetros aprendidos pelo PLSA consiste em um ponto forte para sua utilização em sistemas de recomendação, permitindo sua escalabilidade no nível de predição. Porém, existem alguns contratempos na sua fase de treinamento, sendo seus maiores obstáculos a possibilidade do algoritmo atingir máximos locais e sua complexidade de treinamento de ordem quadrática $O(IxZ)$, onde I representa o número de exemplos e Z o número de grupos, onde tal complexidade é necessária para o cálculo da equação 2.16. Mostraremos a seguir algumas estratégias para contornar estes problemas.

A obtenção de máximos locais no treinamento do PLSA está ligada diretamente ao ponto de partida do algoritmo, ou seja, as probabilidades e parâmetros que são passados para o algoritmo em sua inicialização. Para mostrar este funcionamento considere o seguinte exemplo numérico formado pelos exemplos da tabela 2.1, onde u representa o usuário, y o item e v uma nota atribuída pelo usuário ao item. Neste exemplo forçamos que os usuários 0 e 1 se comportassem de maneira similar, atribuindo notas baixas para os itens 0 e 3 e nota alta para o item 1, já os usuários 2 e 3 se comportam de maneira inversa atribuindo notas altas para 0 e 3 e baixa para 1, este comportamento é representado pelas cores dos asteriscos nas tabelas 2.2 e 2.3. Desta forma, esperamos que algoritmo agrupe os itens (0,1) e (2,3) assim atingindo o máximo global.

Este comportamento se realiza quando inicializamos o algoritmo de forma a indicarmos o caminho para o máximo global. Na tabela 2.2 mostramos um exemplo onde os valores das probabilidades $p(z|u)$ a priori dão indícios de agrupamento dos itens (0,1) e (2,3) e os finais confirmam o agrupamento real. Já na tabela 2.3 mostramos um exemplo de convergência para um máximo local, sendo que nesta os valores da inicialização do algoritmo não dão evidências fortes do agrupamento esperado. Como conseqüência disto é observado que o

agrupamento final gera um erro quadrático superior ao do encontrado na tabela 2.2. Os dados das tabelas 2.2 e 2.3 foram obtidos execução do algoritmo através da implementação que será apresentada no capítulo 3, sendo que em ambos os casos o algoritmo convergiu em menos de 10 iterações.

Tabela 2.1 – Conjunto de Treino

u	y	v
0	0	2
1	0	3
2	0	8
3	0	9
0	1	8
1	1	7
3	1	2
0	2	4
3	2	5
0	3	2
2	3	8

Tabela 2.2 – Convergência do *gPLSA* para o máximo global

		Y				P(z|u) Inicial		P(z|u) Final	
		0	1	2	3	Z1	Z2	Z1	Z2
U	0	*	*	*	*	0,55	0,45	1	0
	1	*	*			0,55	0,45	1	0
	2	*			*	0,45	0,55	0	1
	3	*	*	*		0,45	0,55	0	1
Media ZY	Z1	2,5	7,5	4,0	2,0	RMSE = 0.369			
	Z2	8.5	2,0	5,0	8,0				
Variância ZY	Z1	0,25	0,25	0	0				
	Z2	0,25	0	0	0				

Uma das abordagens para tratar o problema dos máximos locais no *PLSA* é a utilização de outros algoritmos sobre os dados de treino, com o intuito de produzir um modelo que sirva para inicializar o *PLSA*. Com isto, conseguimos orientar melhor o algoritmo de forma a aumentar a chance de atingirmos um máximo global, ou, atingirmos um máximo local melhor do que é gerado por uma

inicialização aleatória. Por exemplo, um algoritmo de *clustering* atinge este objetivo, pois este agrupa usuários de forma a indicar uma chance maior de um usuário pertencer a determinado grupo, fornecendo assim, as probabilidades iniciais do *PLSA* e conseqüentemente uma direção de convergência para o algoritmo.

Tabela 2.3 – Convergência do *gPLSA* para o máximo local

		Y				P(z|u) Inicial		P(z|u) Final	
		0	1	2	3	Z1	Z2	Z1	Z2
U	0	*	*	*	*	0,58	0,42	0,75	0,25
	1	*	*			0,57	0,43	1	0
	2	*			*	0,74	0,26	0,5	0,5
	3	*	*	*		0,37	0,63	0	1
Media ZY	Z1	2,5	7,5	4,0	8,0	RMSE = 1.996			
	Z2	0,85	2	5,0	2,0				
Variância ZY	Z1	0,25	0,25	0	0				
	Z2	0,25	0	0	0				

2.3. Escalabilidade

Outra questão a ser considerada para a utilização do *PLSA* em sistemas de recomendação é a sua escalabilidade. Devido a complexidade de tempo de $O(IxZ)$ o algoritmo torna dispendiosa sua utilização para conjuntos de dados grandes. Uma das soluções mais comuns para abordar este problema é a distribuição do algoritmo, porém não encontramos referências na literatura atual sobre a distribuição do *PLSA*. Devido a isto iremos aprofundar o assunto mostrando as maiores dificuldades para sua distribuição.

A solução ideal para o problema da distribuição do *PLSA* é a distribuição dos dados e do processamento tornando possível aos *peers* (nós ou máquinas na rede) trabalharem de forma independente. Uma abordagem para atingir este objetivo é a distribuição dos exemplos entre os *peers*, porém ao distribuir os exemplos precisamos distribuir também o calculo das equações do *PLSA*. Como já vimos os parâmetros $p(z|u)$, μ_{ZY} e σ_{ZY} são representados por tabelas das dimensões ZxU, ZxY e ZxY respectivamente, alem dos valores esperados (equação 2.16) que são representados por uma tabela IxZ. Para calcular $p(z|u)$ é necessário somar valores esperados para cada item y de cada usuário u, e, normalizar os valores efetuando a soma na dimensão z. Com isto uma abordagem para separação dos cálculos é mover todos os exemplos que envolvam um conjunto de usuários U' para um determinado *peer*, permitindo o cálculo de $p(z|u)$ localmente.

Já o calculo das médias e variâncias (μ_{ZY} e σ_{ZY}) são feitas através da soma sobre os valores esperados de todos os usuários u que possuem exemplos relacionados a cada item y. Assim, para que possa ser feito o cálculo de μ_{ZY} e σ_{ZY} localmente é necessário que todos os exemplos que contenham determinado item y estejam presentes no *peer* onde será feito o seu cálculo. Neste momento é evidente que para efetuar o cálculo local das tabelas envolvidas no *PLSA* é necessária uma distribuição dos dados onde se um usuário for associado a um *peer* todos os seus exemplos devem estar contidos no mesmo *peer*, da mesma maneira, se um item estiver presente em um *peer* todos os seus exemplos também deverão estar presentes no mesmo *peer*.

Um procedimento para separar os dados desta forma é selecionar um determinado usuário e atribuí-lo a um *peer*, em seguida trazer para este *peer* todos

os exemplos deste usuário. Após isto, buscar todos os exemplos dos itens associados aos exemplos do usuário selecionado, e em seguida trazer todos os exemplos associados aos usuários destes exemplos, repetindo este procedimento até não serem mais encontrados exemplos associados aos selecionados. Se após isto ainda sobrarem exemplos, será iniciado o procedimento novamente para outro *peer*. Considerando os exemplos como um grafo, onde os nós são representados pelos usuários e itens e existe uma aresta entre eles sempre que existir um exemplo os relacionando, o procedimento acima descrito encontra os componentes conexos em um grafo bipartido.

Apesar do procedimento de busca dos componentes conexos em um grafo ser relativamente simples, nem sempre os dados são favoráveis para esta separação. Constatamos este fato a partir de um experimento, onde foram extraídos os componentes conexos formados a partir de uma base de anúncios na web (descreveremos melhor esta base no capítulo 4). Neste experimento contamos o tamanho de cada componente conexo através do número de arestas, ou exemplos (tabela 2.5). Como se pode observar existe um componente conexo que agrupa cerca de 92% dos dados, tornando inviável a separação dos dados para a distribuição do *PLSA*.

Apesar desta limitação, existem formas de distribuir a computação do *PLSA* repetindo dados nos *peers*, para tal é necessária uma análise mais completa dos dados do problema para que não seja criado um *overhead* de comunicação entre os *peers.*

Uma solução para a escalabilidade do *PLSA* foi apresentada através de uma patente intitulada *Scalable Probabilistic Latent Semantic Analysis* ou *sPLSA* da empresa Microsoft (Lin et al., 2007). Esta tem como objetivo diminuir o número de cálculos efetuados pelo *PLSA* através da eliminação de casos cuja probabilidade é baixa. Para isto é utilizada uma estratégia em dois passos, o primeiro identifica os casos de baixa probabilidade, e um segundo efetua os cálculos do *PLSA* eliminando os casos identificados no primeiro passo. Vejamos alguns detalhes sobre o *sPLSA*.

Considere o modelo do *PLSA* original (Hoffman, 1999) apresentado na figura 1.1 (b) no contexto de indexação de documentos, onde temos um conjunto de documentos (*D*), palavras (*W*) e grupos (*Z*). Este modelo é estendido através da adição de duas novas variáveis, o grupo de documentos D' e de palavras W',

que são calculados separadamente através de um algoritmo de agrupamento como o k-means (Steinhaus, 1956). Com isto o modelo do *sPLSA* é representado pelo da figura 2.3.

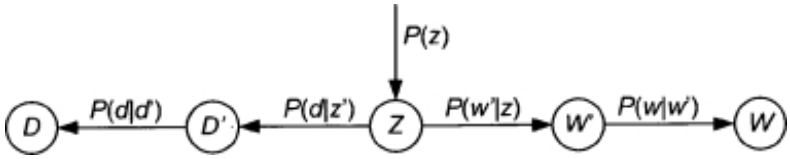

Figura 2.3 – Modelo do *sPLSA*

Tabela 2.4 – Separação dos componentes conexos do corpus de anúncios na web

Tamanho do componente	Número de comp. Com este tamanho	Número de exemplos	Perc. Do Total
1	5421	5421	3,0%
2	1419	2838	1,6%
3	462	1386	0,8%
4	238	952	0,5%
5	128	640	0,4%
6	80	480	0,3%
7	40	280	0,2%
8	30	240	0,1%
9	31	279	0,2%
10	18	180	0,1%
11	15	165	0,1%
12	4	48	0,0%
13	4	52	0,0%
14	4	56	0,0%
15	4	60	0,0%
16	2	32	0,0%
17	4	68	0,0%
18	1	18	0,0%
19	4	76	0,0%
20	1	20	0,0%
21	3	63	0,0%
22	1	22	0,0%
24	1	24	0,0%
25	1	25	0,0%
36	1	36	0,0%
39	1	39	0,0%
168590	1	168590	92,6%
Total:	**7919**	**182090**	**100%**

Com estes grupos calculados é aplicado o *PLSA* sobre o modelo utilizando somente os dados dos grupos (variáveis *D'*, *W'* e *Z*), aprendendo assim os parâmetros $P(z)$, $P(w' \mid z)$ e $P(d' \mid z)$. Dados estes parâmetros é possível extrapolar o modelo de grupos para o modelo original, utilizando as fórmulas 2.23 e 2.24. Nestas $P(d \mid d')$ representa à probabilidade de um documento *d* ocorrer em um determinado grupo *d'*, e, analogamente $P(w \mid w')$ a probabilidade de uma palavra *w* ocorrer dado um grupo de palavras *w',* sendo que tais valores podem ser obtidos através da computação da distância do elemento até o centróide do *cluster* em questão.

$$P(d \mid z) = \sum_{D'} P(d \mid d')P(d' \mid z) \tag{2.23}$$

$$P(w \mid z) = \sum_{W'} P(w \mid w')P(w' \mid z) \tag{2.24}$$

Após calcular as probabilidades dos elementos dados seus grupos são eliminados os casos que possuem probabilidade pequena. Esta eliminação interfere muito pouco no resultado final do algoritmo devido a observação do decaimento exponencial das probabilidades envolvidas no *PLSA*. Considere $Z_d^{(k)}$ como sendo *k-ésima* mais provável classe latente para um documento específico *d*. Temos então que $P(z_d^{(k)} \mid d)$ obedece aproximadamente o decaimento exponencial da fórmula 2.25 (figura 2.4), onde $\lambda(d) > 0$ e $A(d) = e^{\lambda(d)} P(z_d^{(1)} \mid d)$.

$$P(z_d^{(k)} \mid d) = A(d)e^{-\lambda(d)k} \tag{2.25}$$

Este decaimento exponencial é observado também nas probabilidades $P(z_w^{(k)} \mid w)$ e $P(z_{d,w}^{(k)} \mid d, w)$, sendo que quando fixamos $A(d)$ e $\lambda(d)$ obtemos $P(w|d)$ aproximadamente através da consideração somente das *K* mais prováveis classes latentes para *d* e *w,* onde *K* é uma constante obtida através da multiplicação da constante $0 < \beta < 1$ pelo número de classes latentes |Z|, como é mostrado na fórmula (2.26). Assim o *sPLSA* gera um modelo aproximado do *PLSA* com menor número de classes latentes para cada exemplo, obtendo uma melhor performance computacional. Um exemplo de modelo gerado utilizando o *sPLSA* para a indexação de documentos é apresentado na figura 2.4, onde, no

modelo original do *PLSA* seriam consideradas todas as arestas entre *w, z* e *z, d,* e utilizando o *sPLSA* as arestas menos prováveis entre *w, z* e *z, d* são eliminadas.

$$P(w \mid d) = \sum_{1 \le k \le |Z|} P(w \mid z_d^{(k)}) P(z_d^{(k)} \mid d)$$

$$= \sum_{1 \le k \le |Z|} P(w \mid z_d^{(k)}) e^{-\lambda k}$$

$$\approx \sum_{1 \le k \le \beta |Z|} P(w \mid z_d^{(k)}) e^{-\lambda k}$$

$$\approx \sum_{1 \le k \le \beta |Z|} P(w \mid z_d^{(k)}) P(z_d^{(k)} \mid d) \qquad (2.26)$$

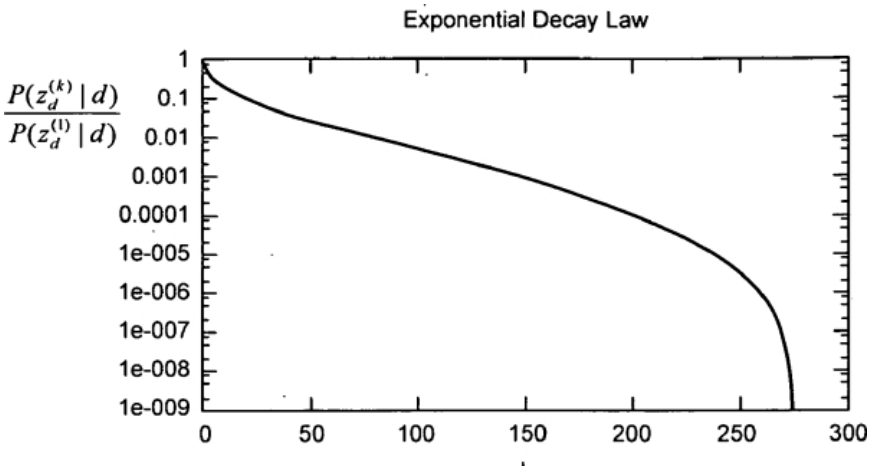

Figura 2.4 – Decaimento exponencial das probabilidades condicionais no *PLSA*

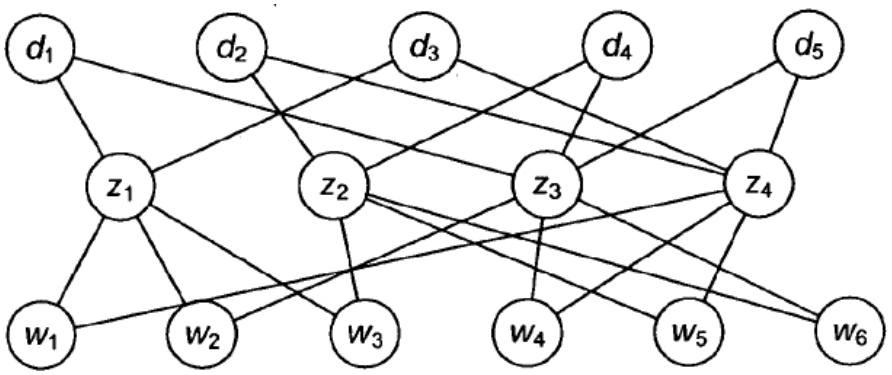

Figura 2.4 – Efeito do *sPLSA* sobre o modelo de indexação de documentos – eliminação de algumas arestas