Convolutional Neural Network and Convex optimization

Si Chen and Yufei Wang
Department of Electrical and Computer Engineering
University of California San Diego
{sic046, yuw176}@ucsd.edu

Abstract

Latent Dirichlet allocation(LDA) is a generative topic model to find latent topics in a text corpus. It can be trained via collapsed Gibbs sampling. In this project, we train LDA models on two datasets, Classic400 and BBCSport dataset. We discuss possible ways to evaluate goodness-of-fit and to detect overfitting problem of LDA model, and we use these criteria to choose proper hyperparameters, observe convergence, and evaluate the models, the criteria we use include perplexity, VI-distance, visualization of clustering results, and highest-probability words.

1 Introduction

Deep learning

Convex optimization

SVM-loss

2 Sub-model Convolutional Network

2.1 Theoretical basis: Convolutional neural network

Convolutional neural networks(CNN) are a special kind of deep neural networks. It exploits local correlation by enforcing a local connectivity pattern between neurons of adjacent layers. For a certain hidden layer m, the hidden units in it are connected to a local subset of units in the $(m-1)$th layer. Additionally, each sparse filter h_i is replicated across the entire visual field. The replicated units share the same parametrization, i.e. the same weight vector and same bias. The layer is called feature map.

Mathematically, a feature map h^k is obtained by convolving the input with a linear filter, adding a bias term and then applying a non-linear function, it can be shown as follow:

$$h^k_{ij} = \mathrm{f}((W^k * x)_{ij} + b_k) \tag{1}$$

where W^k and b_k are weight and bias of kth feature map, and $\mathrm{f}(\cdot)$ is the nolinearity. In our experiments, Rectified Linear Units(ReLU) nonlinearity is used, which has been shown to be more efficient than conventional function $\tanh(\cdot)$.[1] ReLU nonlinearity is as follow:

$$f(x) = \max(0, x) \tag{2}$$

Another important type of layers is pooling. It is a form of non-linear down-sampling. There are several types of pooling, two common types of which are max-pooling and average-pooling. They partition the input image into a set of non-overlapping or overlapping rectangles and outputs the

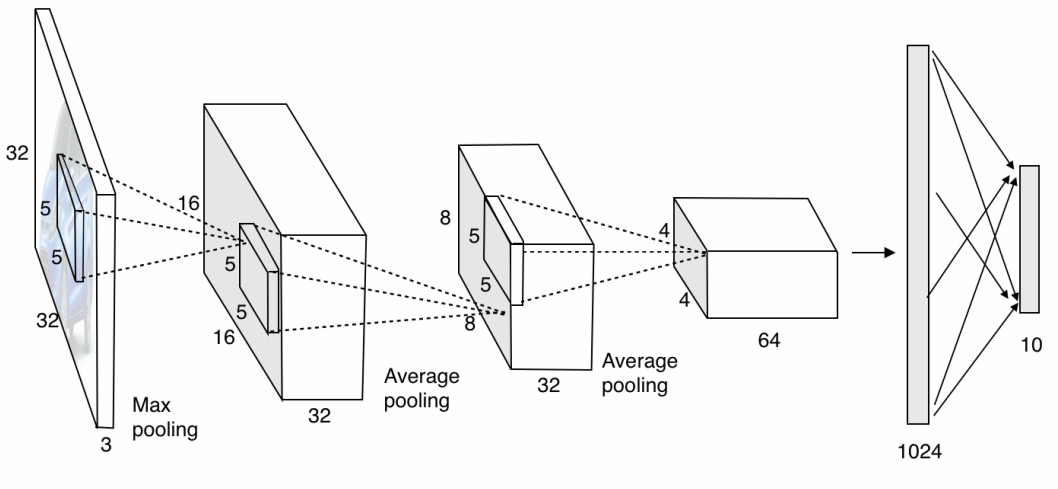


Figure 1: The architecture of our CNN.

maximum/average value for each such sub-region. By pooling, the model can reduce the computational complexity for upper layers, and can provide a form of translation invariance.

Typically, the last layer of a CNN is a logistic regression layer. Each unit of the output reflects a class membership probability:

$$P(Y = i|x, W, b) = softmax_i(Wx + b) = \frac{e^{W_i x + b_i}}{\sum_j e^{W_j x + b_j}} \tag{3}$$

The parameters of the network are trained using back propagation[2]. The loss function used for training is the negative-log likelihood of the training dataset D under the model:

$$L = \sum_{i=0}^{|D|} \log(P(Y = y^{(i)}|x^{(i)}, W, b)) \tag{4}$$

Finally, the prediction of the model is done by taking the argmax of the vector of $P(Y = i|x, W, b)$:

$$y_{pred} = \text{argmax}_i P(Y = i|x, W, b) \tag{5}$$

2.2 Overall architecture

The overall architecture of our CNN is shown in Figure 1. There are three convolutional layers and pooling layers alternatively. Overlapping pooling is performed. Each pooling layer consists of a grid of pooling units spaced $s = 2$ pixels apart, each summarizing a neighborhood of size 3×3 centered at the location of the pooling unit.

2.3 Sub-model combination

Let's focus on the penultimate layer in our CNN architecture (Figure 1). We refer to the already trained CNN model in Figure 1 as the original model. There are totally 1024 units in the penultimate layer, each of which votes for the class membership probabilities $P(Y = i|x, W, b)$, and the importance of each unit's vote is weighted by W_{ij}. The 1024-unit vector of the penultimate layer is denoted as h_{orig}.

Now we define a sub-model m_i as randomly setting units as zero in the penultimate layer with chance of 50%, and the remaining units are multiplied by 2:

$$P(h^j_{m_i} = 2h^j_{orig}) = P(h^j_{m_i} = 0) = \frac{1}{2}. \tag{6}$$

where h_{m_i} is the 1024-unit vector of the sub-model m_i, and $h^j_{(\cdot)}$ is the jth element of the vector.

Therefore, the original model can be viewed as a combination of all the possible sub-models:

$$h_{orig} = \lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} h_{m_i} = E\left[h_{m_i}\right] \tag{7}$$

For each sub-model, the class membership probability output is denoted as P_{m_i}. Rather than combining the penultimate layers h_{m_i} of the sub-models, we explore the weighted combination of output layer P_{m_i} of sub-models:

$$P_{comb} = \sum_{i=1}^{n} l_i P_{m_i} \tag{8}$$

where l_i is the weight of each model m_i, and P_{comb} is the class membership probabilities of the weighted combined model.

Given a set of sub-models $m_i, (i = 1, 2, ..., n)$ extracted from the original model m_{orig}, the optimal weight l_i can be obtained by the following optimization problem:

$$\text{minimize } \sum_{i=1}^{N} \left\| l^t \cdot P_i - y_i \right\|_2^2 \text{ subject to } l \geq 0$$

2.4 Dropout and sub-models

3 Multiclass

4 Experiments

4.1 Sub-model convolutional network

LDA is a mixture model. It assumes that each document contains various topics, and words in the document are generated from those topics. All documents contain a particular set of topics, but the proportion of each topic in each document is different.

The generative process of the LDA model can be described as follows:

Given: Dirichlet distribution with parameter vector α of length K

Given: Dirichlet distribution with parameter vector β of length V

for topic number 1 to topic number K

draw a word distribution, i.e. a multinomial with parameter vector ϕ_k

according to β. $\phi \sim Dirichlet(\beta)$

for document number 1 to topic number M

draw a topic distribution, i.e. a multinomial with parameter vector θ

according to α. $\theta \sim Dirichlet(\alpha)$

for each word in the document

draw a topic z according to θ. $z \sim Multinomial(\theta)$

draw a word w according to ϕ_z. $w \sim Multinomial(\phi_z)$

Not that V is the cardinality of the vocabulary. The words of a document is observed, but the topic assignment is latent.

4.2 Training via collapsed Gibbs sampling

Training data in this project is the words in all documents. The goal of training is to infer the multinomial parameters θ for each document, and ϕ_k for each topic.

We use collapsed Gibbs sampling for learning. First it infers the hidden value z_{nm} for each word occurrence in each document: $p(\vec{z}|\vec{w})$. Note z_{mn} is the mixture indicator that chooses the topic for nth word in document m. $\vec{w}$ is the vector of words making up the entire corpus, and $\vec{z}$ is the corresponding topics.

The idea of Gibbs sampling is assuming that we know the value of $\vec{z}$ for every word occurrence in the corpus except occurrence number i, then we draw a z_i value for i according to its distribution. Then we assume that this value is known to be true value, and draw a z_j for another word, and so on. Eventually this process will converge to a correct distribution $p(\vec{z}|\vec{w})$. Therefore, we need the conditional distribution for a word token with index $i = (m, n)$:

$$p(z_i = k|\vec{z}_{-i}, \vec{w}) \propto \frac{n^{(t)}_{k,-i} + \beta}{\sum_{t=1}^{V}(n^{(t)}_{k,-i} + \beta)}(n^{(k)}_{m,-i} + \alpha) \tag{9}$$

where z_i is the topic association of ith word. $\vec{z}_{-i}$ is $\vec{z}$ with ith topic removed. $n^{(t)}_k$ is the number of times that word t has been observed with topic k. $n^{(k)}_m$ refers to the number of times that topic k occurs with a word of document m. $n^{(\cdot)}_{\cdot,-1}$ indicates that the word i is excluded from the corresponding document or topic.

Finally, we can obtain the multinomial parameters θ and ϕ:

$$\phi_{k,t} = \frac{n^{(t)}_k + \beta}{\sum_{t=1}^{V}(n^{(t)}_k + \beta)} \tag{10}$$

$$\theta_{m,k} = \frac{n^{(k)}_m + \alpha}{\sum_{k=1}^{K}(n^{(k)}_m + \alpha)} \tag{11}$$

where $\phi_{k,t}$ is the multinomial parameter for word t drawn from topic k. $\theta_{m,k}$ is the parameter of topic k drawn from document m.

4.3 Goodness-of-fit of the model

There are two typical ways to evaluate LDA model.

4.3.1 Likelihood of held-out data

One way is to calculate the probability of held-out documents that are not used for training. The likelihood $p(\vec{w}|\alpha, \beta)$ is intractable, and there are several ways to estimate it.

The probability of the held-out documents $\vec{w}$ given training data $\vec{w'}$ can be written as

$$p(\vec{w}|\vec{w}') = \int \mathrm{d}\phi \mathrm{d}\alpha\, p(\vec{w}|\phi, \alpha)p(\phi, \alpha|\vec{w}') \tag{12}$$

[3] approximates this integral by evaluating at a point estimate, and the problem becomes how to evaluate $p(\vec{w}|\phi, \alpha)$. [3] summaries several estimating methods: importance sampling methods, harmonic mean method, and annealed importance sampling. [3] also proposes two alternative methods: a Chib-style estimator and a "left-to-right" evaluation algorithm.

For example, the frequently used method, Harmonic mean method, estimates the probability as follows

$$p(\vec{w}|\phi, \alpha) \simeq \frac{1}{\frac{1}{s}\sum_s \frac{1}{p(\vec{w}|\vec{z}^{(s)}, \phi)}} \tag{13}$$

where $\vec{z}^{(s)}$ is drawn from $p(\vec{z}|\vec{w})$.

[**?**] approximates the held-out likelihood with empirical likelihood(EL). It produces a large set of pseudo documents with parameter α and β, and the pseudo documents are then used to train a tractable model. The true likelihood of the test set is then estimated as its likelihood under the tractable model.

Perplexity is another way to calculate the likelihood. It is defined as the reciprocal geometric mean of the token likelihoods in the test corpus given the model:

$$p(\vec{W}|M) = \exp - \frac{\sum_{m=1}^{M} \log p(\tilde{\vec{w}}_{\tilde{m}}|M)}{\sum_{m=1}^{M} N_m} \tag{14}$$

where M is the trained model, and $\vec{w}_{\tilde{m}}$ is the word vector in document $\tilde{m}$. Lower values of perplexity indicate lower misrepresentation of the words of the test documents by the trained topics.

The log-likelihood can be expressed as:

$$\log p(\tilde{\vec{w}}_{\tilde{m}}|M) = \sum_{t=1}^{V} n_{\tilde{m}}^{(t)} \log(\sum_{k=1}^{K} \phi_{k,l} \cdot \theta_{\tilde{m},k}) \tag{15}$$

where $n_{\tilde{m}}^{(t)}$ is the number of times word t occurs in document $\tilde{m}$.

4.3.2 Evaluation of clustering result

Another way to evaluate the model is to measure its performance on some secondary tasks, for example, document classification or clustering. The LDA model provides a soft clustering of the documents. The evaluation of clustering quality can be done in many ways:

1. We can simply check if the most frequent words generated from each topic are semantically related.
2. We can also check if intra-class document similarity is higher than inter-class document similarity.
3. When we have the label of each document, and the number of topics K equals the number of given labels, we can convert the soft clustering result into a hard result, and calculate the categorization accuracy. Converting soft clustering to a hard one is simply done by assigning topic with highest probability to each document.
4. We can also directly compare the soft clustering result with the priori categorization([4]). Variation of Information distance (VI-distance) can be used to compare the two clusterings. It assumes two distributions over class for each document: $p(c = j|d_m)$ and $p(z = k|d_m)$, where $j \in [1, J]$ and $k \in [1, K]$ are the class labels or topics of the two distribution. Note that K doesn't need to be the same with J. d_m is mth document. The class probabilities are obtained by averaging over the corpus:$p(c = j) = \frac{1}{M} \sum_m p(c = j|d_m)$ and $p(z = k) = \frac{1}{M} \sum_m p(z = k|d_m)$. The joint probability of co-occurring pairs is $p(c = j, z = k) = \frac{1}{M} \sum_m p(c = j|d_m)p(z = k|d_m)$. Then the VI-distance measure is defined as follow:

$$D_{VI}(C, Z) = H(C) + H(Z) - 2I(C, Z) \tag{16}$$

with

$$I(C, Z) = \sum_{j=1}^{J} \sum_{k=1}^{K} p(c = j, z = k)[\log_2 p(c = j, z = k) - \log_2 p(c = j)p(z = k)] \tag{17}$$

$$H(C) = -\sum_{j=1}^{J} p(c = j) \log_2 p(c = j) \tag{18}$$

$$H(Z) = -\sum_{k=1}^{K} p(z = k) \log_2 p(z = k) \tag{19}$$

VI-distance is always nonnegative, and smaller VI-distance indicates the two clustering are more similar.

4.4 Overfitting monitoring

In Section 4.3.1, we introduce several ways to calculate the likelihood of held-out data. In addition to using likelihood or perplexity to evaluate the goodness-of-fit, we can also use it to monitor overfitting. By calculating likelihood/perplexity of the training data, and comparing it with likelihood/perplexity of test data, we can get the idea whether overfitting occurs. When no overfitting occurs, the difference between two types of likelihood should remain low.

5 Implementation of algorithm

Algorithms are realized in Python.

Data The vocabulary of words is very large. However, in one document, only a small part of the vocabulary occurs, and there is no need to traverse every word in vocabulary for each document. Therefore, word counts of all documents are stored in `scipy.sparse.csc_matrix`, a compressed sparse column matrix. Non-zero values can be efficiently located, so the training process can be speeded up.

Initialization The hyperparameters α, β, and K are predefined. The initial topics z_{mn} associating with words w_{mn} are random number ranging from 1 to K. For words in all documents, the counts $n_k^{(t)}$ and $n_m^{(k)}$ are calculated.

Gibbs sampling The counts $n_k^{(t)}$ and $n_m^{(k)}$ are stored in matrices. Instead of looping over K to calculate the conditional probability $p(z_i = k|\vec{z}_{-i}, \vec{w})$ for each topic k, we use matrix operation to calculate unnormalized conditional probability of all topics. This is faster than simply looping over K topics for each word.

Visualization of clustering result To evaluate the clustering result in an intuitive manner, we draw a mapping from a document to a 3D point according to each document's multinomial parameter θ. When there are 3 topics or less, we can simply assign each topic to one axis. In this way, each coordinate corresponds to one topic distribution (θ). When there are more than 3 topics, we use principle component analysis (PCA) to reduce the dimension. Three principle components are extracted, and then each documents can be mapped to a 3D point. We use points with different colors and shapes to represent different classes of documents. Therefore we can observe classification result as well as clustering result in the 3D image.

6 Design of experiments

6.1 Datasets

We use two datasets for experiments. The first is Classic400 dataset [1]. It consists of 400 documents, with a vocabulary of 6205 words. There are 3 categories of the documents. The second dataset we use is the BBC Sports dataset [2]. It consists of 737 documents with a vocabulary of 4613 words. There are 5 classes of the documents. For both datasets, we randomly choose10% from each category as test data, and 90% as training data.

6.2 Choice of hyperparameters

We use perplexity and VI-distance as the criteria to choose the three hyperparameters: α, β, and K. Best choices of α and K have strong correlation, as is shown in [**?**], which chooses hyperparameters as $\alpha = 50/K$, $\beta = 0.1$. For both datasets, we have the true label of every document. Therefore, an intuitive guess of K would be the number of classes. Grid search for all three hyperparameters requires too much work, so we first fix K as the number of classes, and apply grid search of α and β. After deciding best α and β for fixed K, we then search for best K with fixed β and changing α, under the assumption that $\alpha \propto \frac{1}{K}$.

6.3 Evaluating results

We use several ways to evaluate the models:

1. We look at the 10 highest-probability words for each topic, to see if topics we learn are meaningful.

[1] http://cseweb.ucsd.edu/users/elkan/151/classic400.mat
[2] http://mlg.ucd.ie/datasets/bbc.html

Parameter(α, β)	Perplexity (training)	Perplexity (test)	VI-distance (training)
(0.01, 0.1)	1376.024120	2500.267304	0.796688
(0.01,1)	1803.455444	2303.252994	1.381389
(0.01,2)	2088.023795	2506.341132	0.909851
(0.1,0.1)	1451.690149	2646.197142	1.387363
(0.1,1)	1753.193891	2256.134722	0.801898
(0.1,2)	2071.336636	2462.702027	0.704664
(1,0.1)	1484.825815	2536.642742	1.567750
(1,1)	1802.399166	2310.176229	1.260563
(1,2)	2128.084925	2492.720476	1.362268

Table 1: Perplexity and VI-distance of Classic400 dataset, with $K = 3$ and varying α, β.

2. We calculate perplexity of training data and perplexity of test data for every 5 epochs, to check convergence and check overfitting.
3. We calculate VI-distance of training data to monitor clustering result.
4. We visualize the clustering result in 3 dimensional space.
5. We also compare the running time of each Gibbs sampling epoch of the two models.

7 Experimental results

7.1 Classic400 dataset

7.1.1 Best hyperparameters

By fixing K at 3, we use grid search for pairs (α, β). We choose from $\alpha = \{0.01, 0.1, 1\} = \{0.03/K, 0.3/K, 3/K\}$, and $\beta = \{0.1, 1, 2\}$. Figure 2 shows the visualizing result of the 9 pairs of parameters, and Table 1 shows the corresponding perplexity and VI-distance.

From Figure 2, we can see the role of α. For smaller α, documents in each cluster are more scattered; for larger α, different clusters are more separated, and the documents are prone to locate in three corners of the triangle. This is also illustrated in Table 1, where VI-distance increases with α. This observation is intuitive: α is the pseudo count of topic per document. Larger α will bring an averaging effect of different topics.

From Table 1, we can observe the role of β. For smaller β, perplexity of training data is smaller. However, perplexity of test data doesn't change much with β. β is the pseudo count of word per document. Larger β brings a larger averaging effect of all the words, therefore making the model not so fit for training data.

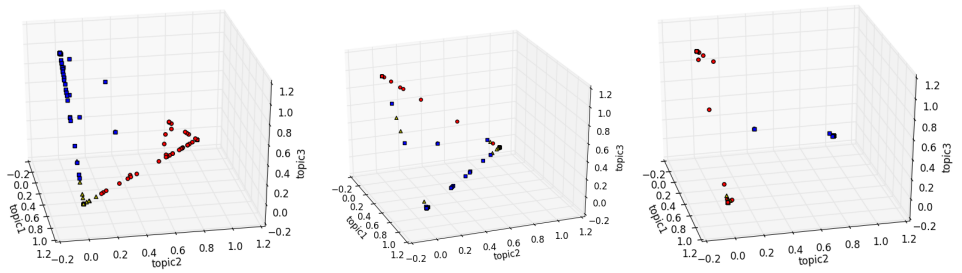
We choose $(\alpha = 0.3/K, \beta = 1)$ as the best parameter pairs, to achieve low perplexity and VI-distance.

Then, we fix $\beta = 1$, $\alpha = 0.3/K$, and search for best K. We check perplexity, VI-distance and the mapping image for $K = \{2, 3, 4, 5\}$. Figure 3 shows visualization of clustering results with different K. For different K, location of points forms different 3D shapes: for $K = (2, 3, 4, 5)$, the shapes are line, triangle, tetrahedron, tetrahedron with an extra point in the middle, respectively. Points reside in vertex of those shapes. Table 2 shows the evaluation of goodness-of-fit. $K = 3$ achieves best perplexity on test data and VI-distance. Training perplexity decreases with larger K, because when we have larger topic number we have more parameters to fit the training data, however, bigger K causes overfitting: Test perplexity increases when K is big, this is because when K becomes too large, the number of parameters are too large for the training data. Therefore, we choose $K = 3$ as the best parameter.

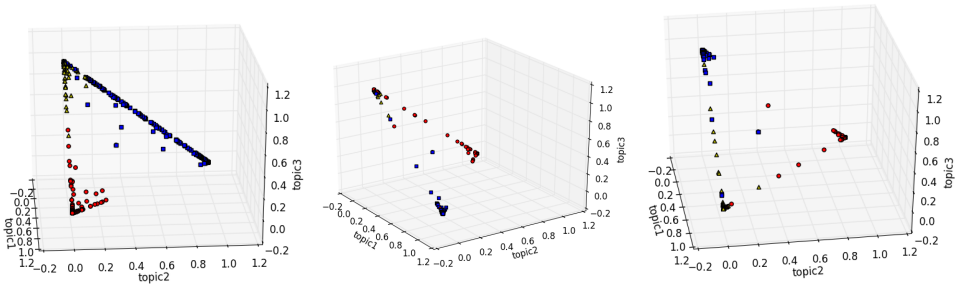
7.1.2 Evaluation results

We have decided the parameters to be $K = 3, \alpha = 0.3/K, \beta = 1$. Then, we evaluate the model.

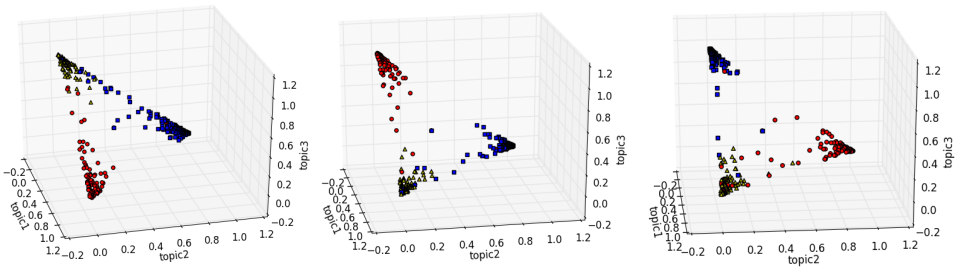
Figure 4 shows training and test perplexity of the model with number of epochs. Perplexity on test data is higher than that on training data, and the dropping rate of test perplexity is slightly



(a) $\alpha = 0.01, \beta = 0.1$ (b) $\alpha = 0.01, \beta = 1$ (c) $\alpha = 0.01, \beta = 2$



(d) $\alpha = 0.1, \beta = 0.1$ (e) $\alpha = 0.1, \beta = 1$ (f) $\alpha = 0.1, \beta = 2$

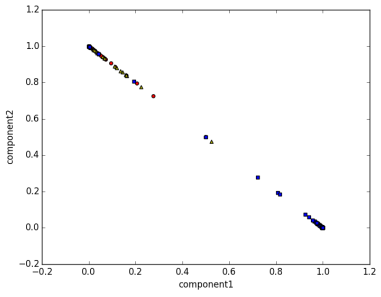


(g) $\alpha = 1, \beta = 0.1$ (h) $\alpha = 1, \beta = 1$ (i) $\alpha = 1, \beta = 2$

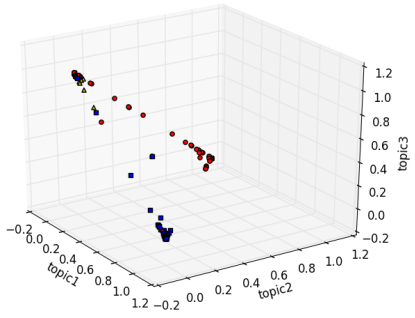
Figure 2: Plots of Classic400 dataset with $K = 3$ and varying α and β.

Parameter(K)	Perplexity (training)	Perplexity (test)	VI-distance(training)
2	1931.919028	2415.384221	0.864250
3	1753.193891	2256.134722	0.801898
4	1718.810470	2283.469816	1.339418
5	1704.840676	2364.641784	1.864304

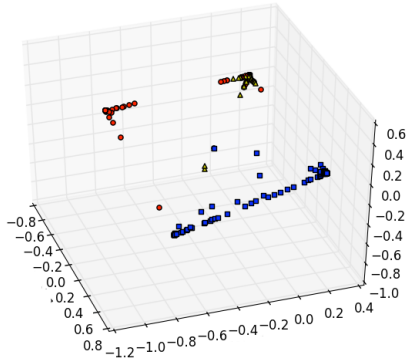
Table 2: Perplexity and VI-distance of Classic400 dataset, with $\beta = 1, \alpha = 0.3/K$ and varying K



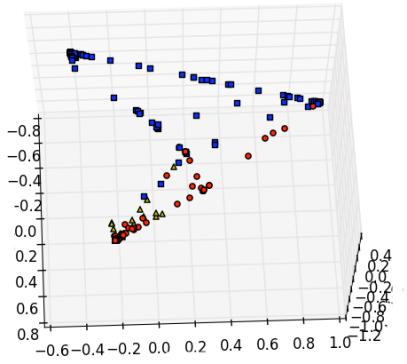
(a) $K = 2$, $\alpha = 0.15$



(b) $K = 3$, $\alpha = 0.1$



(c) $K = 4$, $\alpha = 0.075$



(d) $K = 5$, $\alpha = 0.06$

Figure 3: Plots of Classic400 dataset with $\beta = 1, \alpha = 0.3/K$ and varying K.

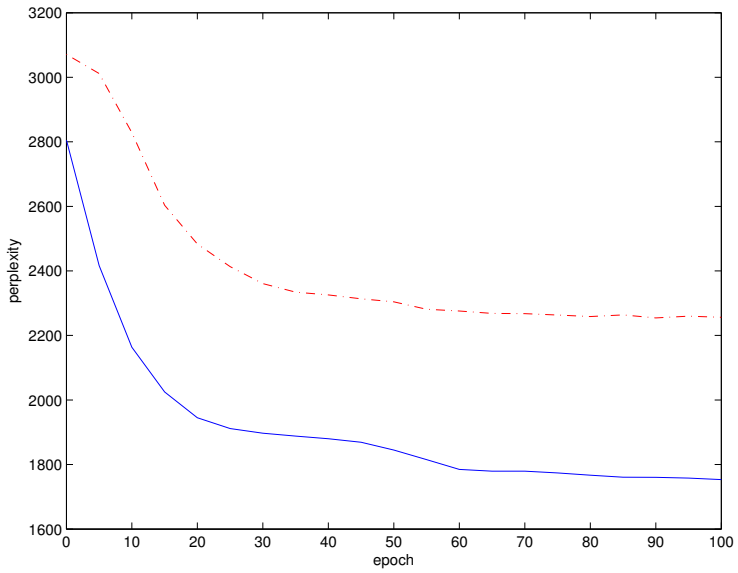


Figure 4: Training and test perplexity of LDA model for Classic400. Blue hard line: training data; Red dash-dot line: test data.

Topic	Ten most frequent words
1: 'Medical'	patients fatty acids nickel ventricular aortic cases left glucose septal
2:'Scientific methods'	system scientific retrieval research language science methods systems subject journals
3:'aero-physics'	boundary layer wing supersonic velocity mach wings ratio jet plate

Table 3: Top 10 frequent words for each topic of LDA model for Classic400.

lower. However, the trend of the two perplexities are similar. Therefore, we could infer there may exist slight overfitting, this may be caused by lack of data (we only have 360 training documents). According to the figure, the model converges after 60 epochs.

Table 3 shows 10 words that most frequently occur in each topic. In each topic, words are semantically related, and they are related to the true topic assigned by human. This means the model we train has a good clustering for the topics, with semantic meaning.

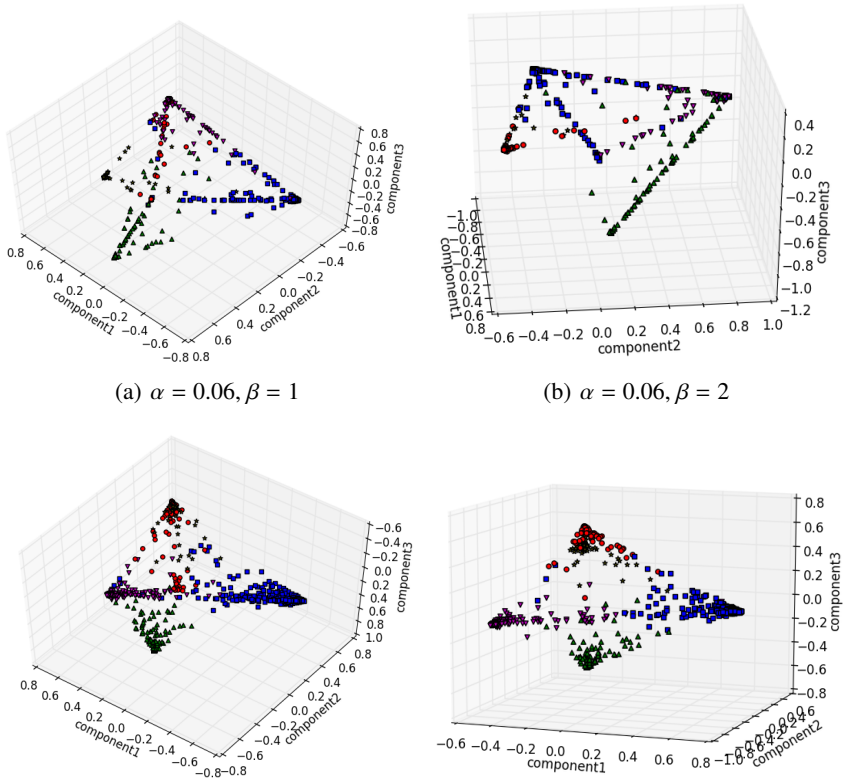
We calculate the average running time for every epoch of Gibbs sampling. The average running time of one epoch is 3.3649 seconds, which is shown in Table 7.

7.2 BBCSport dataset

7.2.1 Best hyperparameters

First we fix $K = 5$, which is the number of labels given. With the experience in Classic400 dataset, this time we do the grid search for $\alpha = \{0.3/K, 3/K\}, \beta = \{1, 2\}$. Figure 5 shows the visualization result of four models with each parameter pair. Table 4 shows the perplexity and VI-distance of the corresponding models.

From Figure 5, we can see that when α increases, the documents are more scattered. Note that there are 5 topics, and PCA cannot separate them entirely on the 3D image. From Table 4, we observe that larger β leads to larger perplexity on both training and test data. VI-distance is larger with larger α. These trends are similar with what we observe at Classic400 dataset. For best perplexity and VI-distance, we choose the pair $(\alpha = 0.3/K, \beta = 1)$. Then, we fix $(\alpha = 0.3/K, \beta = 1)$ and change K,



(a) $\alpha = 0.06, \beta = 1$ (b) $\alpha = 0.06, \beta = 2$

(c) $\alpha = 0.6, \beta = 1$ (d) $\alpha = 0.6, \beta = 2$

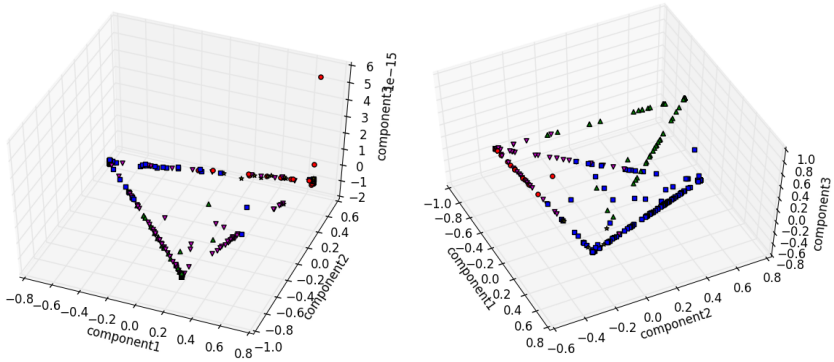
Figure 5: Plots of BBCSports dataset with $K = 5$ and varying β and α.

Parameters (α, β)	Perplexity(training)	Perplexity(test)	VI-distance
(0.06,1)	1194.730739	1413.515161	1.836619
(0.06,2)	1278.352876	1499.750074	1.735083
(0.6,1)	1207.601372	1434.633843	2.127757
(0.6,2)	1282.181586	1491.711059	2.013811

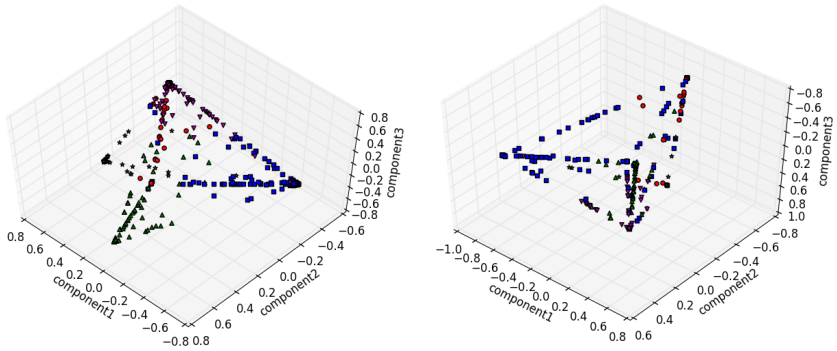
Table 4: Perplexity and VI-distance of BBCSport dataset, with $K = 5$ and varying α, β.

Parameter (K)	Perplexity(training)	Perplexity(test)	VI-distance
3	1347.412626	1610.204673	1.691689
4	1272.958416	1498.080381	1.801365
5	1163.215412	1395.538324	1.671155
6	1155.549721	1402.170199	2.183145

Table 5: Perplexity and VI-distance of BBCSport dataset, with $\beta = 1, \alpha = 0.3/K$ and varying K



(a) $K = 3, \alpha = 0.1$ (b) $K = 4, \alpha = 0.075$



(c) $K = 5, \alpha = 0.06$ (d) $K = 6, \alpha = 0.05$

Figure 6: Plots of BBCSport dataset with $\beta = 1, \alpha = 0.3/K$ and varying K.

with $K = \{3, 4, 5, 6\}$. From Figure 6 and Table 6, we choose $K = 5$, with lowest perplexity in both training and test data, and low VI-distance.

Therefore, the hyperparameters we choose are $K = 5, \alpha = 0.3/K, \beta = 1$.

7.2.2 Evaluation results

Figure 7 shows training and test perplexity of the model as number of epochs increases. Perplexity on test data is still higher than that on training data, and the dropping rate of test perplexity is

Topic	Ten most frequent words
1: Rugby	england wale game ireland rugbi against nation plai six player
2: Athletics	olymp world athlet race year indoor athen test champion win
3: Cricket	test cricket plai england first seri south match run australia
4: Football	game player club plai chelsea arsen unit leagu goal footbal
5: Tennis	plai open win match first set year final game roddick

Table 6: Top 10 frequent words for each topic of LDA model for BBCSport.

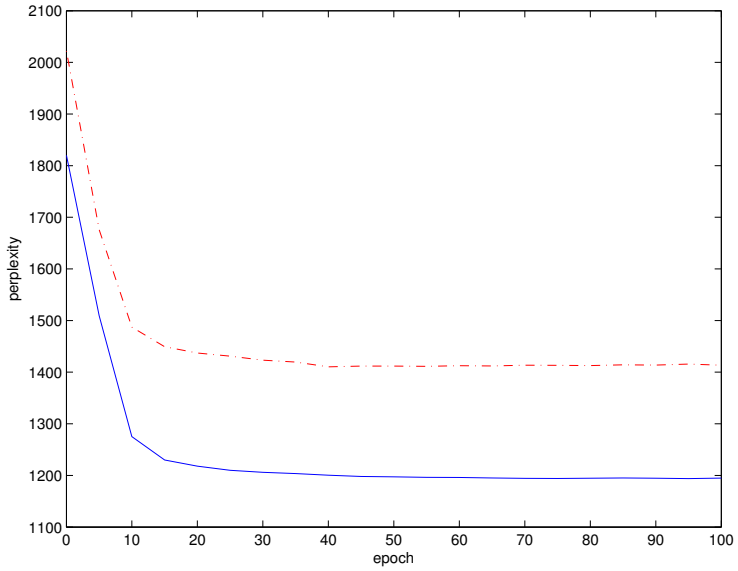


Figure 7: Training and test perplexity of LDA model for BBCSport. Blue hard line: training data; Red dash-dot line: test data.

slightly lower. Compared with Figure 7, we can see that model of BBCSport has less overfitting than Classic400's model. The reason may be there being more training data in this BBCSport dataset: BBCSport training set has 109579 words in total, while Classic400 training set has only 27415 words. Although BBCSport dataset needs to train more parameters, it still has larger dataset-parameter ratio. From the figure, we can also observe rate of convergence of the model. The model converges after 40 epochs.

Table 6 shows ten words that most frequently occur in each topic. The words in the dataset are pre-processed before training: Stemming is performed. This avoids a word of different tense or a word in plural form being regarded as a different word. Since we only care about the topic related to the words, eliminating the impact of tense or plural form is sensible. Note that $K = 5$ is the true label number, therefore we can assign each topic to a label, judging by the semantic meaning of the top 10 words. Some words are very closely related to the label, for example, "chelsea", "arsen" (which stands for arsenal) are strongly related to football. There are other words that are shared by some topics, for example, "year", "win" and "plai" are shared by two or three topics. This is because the five topics are all about sports. They are semantically close to each other, and share some words that associate with all kinds of sports.

From one training process, we calculate the average running time for every epoch of Gibbs sampling. The average running time of one epoch is 12.3923 seconds, as is shown in Table 7. The normal time to perform one epoch of Gibbs sampling is $O(NK)$, where N is the total number of words in all documents. However, as is described in Section 5, we can make the time much less relevant to K by using matrix operation, making the time comparable to $O(N)$ when K is not too large. This is also illustrated in Table 7: For BBCDataset, increasing K from 5 to 20 doesn't affect running time much. When K becomes 1000, there is a noticeable time increase. Therefore, one-epoch running time for BBCSport dataset is approximately 4 times of Classic400 dataset, which is proportional to their word number ratio.

8 Conclusions

In this project, we realize the LDA model by Gibbs sampling, and evaluate it on two datasets. We use multiple types of evaluation criteria to check performance of the model. From the experiment results, we find that hyperparameters have different roles in the model: increasing α makes document

Datasets	Number of topics: K	Total number of words:N	Running time (s)
Classic400	3	27415	3.3649
BBCSport	5	109579	12.3923
BBCSport	6	109579	12.3834
BBCSport	10	109579	12.1788
BBCSport	20	109579	11.9438
BBCSport	1000	109579	18.0922

Table 7: Comparison of running time of two datasets.

clusters more scattered; smaller β results in smaller perplexity when no overfitting occurs; and larger topic number K makes training perplexity smaller but may cause overfitting. The perplexity results of two datasets suggest probable overfitting, which is caused by insufficient data. However, we can observe that latter dataset with larger data gets less overfitting. One problem of this implementation of LDA model is that it is not well scalable. Training time for one epoch is proportional to scale of dataset. Therefore training will become very slow with a large dataset.

References

[1] Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems. (2012)

[2] Y. LeCun, L. Bottou, G.O., Muller, K.: Efficient backprop. Neural Networks: Tricks of the trade (1998)

[3] Wallach, H.M., Murray, I., Salakhutdinov, R., Mimno, D.: Evaluation methods for topic models. In: Proceedings of the 26th Annual International Conference on Machine Learning, ACM (2009) 1105–1112

[4] Heinrich, G.: Parameter estimation for text analysis. Technical report (2004)