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

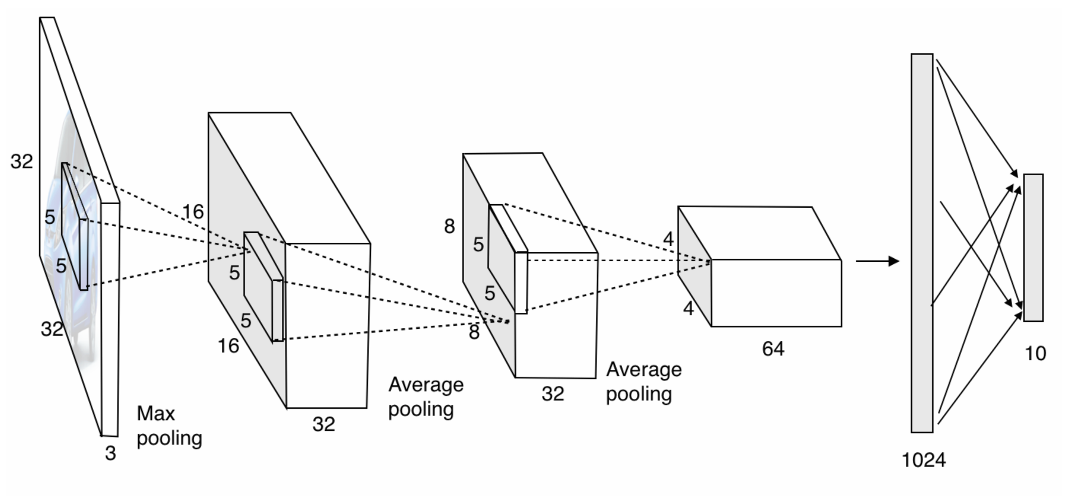


Figure 1: The architecture of our CNN.

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. The output of the network is a vector of class membership probability with 10 units, corresponding to 10 classes in our CIFAR-10 dataset[1].

2.3 Dropout and sub-model combination

Dropout is a recently-introduced technique to reduce overfitting [1]. It consists of setting to zero the output of each hidden neuron with probability of 0.5 in training procedure. So every time a training example is presented, the neural network samples a different architecture, which we denote here as a sub-model m_k. All these sub-models share weights. At test time, we use all neurons but multiply their outputs by 0.5, which is an approximation to taking the geometric mean of the predictive

[1]http://www.cs.toronto.edu/ kriz/cifar.html

distributions $P_{m_k} = [P_{m_k}(Y = 1), P_{m_k}(Y = 2), ..., P_{m_k}(Y = d)]^t$ produced by the sub-models (where d is the number of classes):

$$\begin{aligned} (P_{m_1}(Y = i) \cdot P_{m_2}(Y = i) \cdot \cdots \cdot P_{m_n}(Y = i))^{\frac{1}{n}} &= \frac{e^{(b_i + W_i \frac{1}{n} \sum_{k=1}^{n} h_{m_k})}}{(\sum_{j=1}^{d} e^{(nb_j + W_j \sum_{k=1}^{n} h_{m_k})})^{\frac{1}{n}}} \\ &= \frac{e^{(b_i + W_i \cdot h_{comb})}}{(\sum_{j=1}^{d} e^{(nb_j + W_j \sum_{k=1}^{n} h_{m_k})})^{\frac{1}{n}}} \end{aligned} \tag{6}$$

$$P_{comb}(Y = i) = \frac{e^{b_i + W_i \cdot h_{comb}}}{\sum_{j=1}^{d} e^{b_j + W_j \cdot h_{comb}}} \tag{7}$$

where h_{m_k} is the neurons of the penultimate layer of sub-model m_k, and h_{comb} is the that of the combined model by using all the neurons but multiply them by 0.5. $P_{comb}(\cdot)$ is the class membership probabilities of the combined model.

Apparently, the predictive distribution of the combined model is only a biased approximation of taking the mean of the predictive distribution of all sub-models.

In our experiments, the architecture is shown in Figure 1, dropout is performed in the output of penultimate layer, and $d = 10$ is the number of classes of our dataset.

Here, we want to explore a better option of combining the sub-models. Rather than giving an approximation, we want to actually taking the linear combination of the predictive distribution of n sub-models:

$$P_{l.comb}(Y = i) = \sum_{k=1}^{n} l_k \times P_{m_k}(Y = i) \tag{8}$$

where l_k is the weight of the distribution of kth model. The best weight $l = [l_1, l_2, ..., l_n]^t$ can be obtained solving following optimization problem:

minimize $\sum_{i=1}^{N} ||P_i \cdot l - y_i||_2^2$,

subject to $l \geq 0$.

with the variable l, where N is the number of all the training examples, y_i is the 10×1 binary column vector indicating the true label of ith data point, and $P_i = [P_{m_1}, P_{m_2}, ..., P_{m_n}]$ is a $10 \times n$ matrix. By minimizing the sum of squared l2 norm of difference between predicted distribution and true label, we find the non-negative weight vector l.

The objective function can be simplified as following:

$$\begin{aligned} \sum_{i=1}^{N} ||P_i \cdot l - y_i||_2^2 &= \sum_{i=1}^{N} (P_i \cdot l - y_i)^t (P_i \cdot l - y_i) \\ &= (P \cdot l - y)^t (P \cdot l - y) \\ &= l^t P^t P l - 2y^t P l + y^t y \end{aligned} \tag{9}$$

where $y = \left[y_1^t, y_2^t, ...y_N^t\right]^t$ is the concatenation of label indicator vectors of all data points, which is a $10N \times 1$ vector. $P = \left[P_1^t, P_2^t, ..., P_n^t\right]^t$ is the concatenation of probability distributions of all data points, which is a $10N \times n$ matrix. Then the optimization problem can be written as follow:

minimize $l^t P^t P l - 2y^t P l + y^t y$,

subject to $l \geq 0$.

with the variable l. This is a Quadratic Optimization Problem (QP), apparently a convex optimization problem, and is easy to solve.

2.4 Sub-model combination in non-dropout CNN

Now let's move back to CNN without dropout training. The same idea of sub-model combination can also be utilized to improve the performance of CNN when the network is trained without dropout.

The sub-models can be obtained by similar fashion with the dropout sub-models: given the already trained CNN model without dropout, randomly set penultimate units to zero with probability of 50%, and multiply the remaining units by 2:

$$\Pr(h^j_{m_i} = 2h^j_{orig}) = \Pr(h^j_{m_i} = 0) = \frac{1}{2}. \tag{10}$$

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

Then, we can obtain the predictive distribution of each sub-model P_{m_k}, and the new model takes linear combination of the predictive distribution of n sub-models, which is the same as the dropout condition:

$$P_{l.comb}(Y = i) = \sum_{k=1}^{n} l_k \times P_{m_k}(Y = i) \tag{11}$$

Therefore, the optimization problem is very similar with dropout condition, and the only difference is the multiplication by 2 when obtaining the penultimate layer of sub-models.

3 Multiclass

4 Experiments

4.1 Sub-model convolutional network

We use Theano[3] to train and test the CNN model, and cvx toolbox[2] in matlab to solve the convex optimization problem finding best weights. The training and test data are CIFAR-10 dataset.

We separately train the model with and without dropout, and the the result on the test set is shown in 1. Accuracy of dropout network is about 2% higher than nondropout network, which shows the advantage of dropout method.

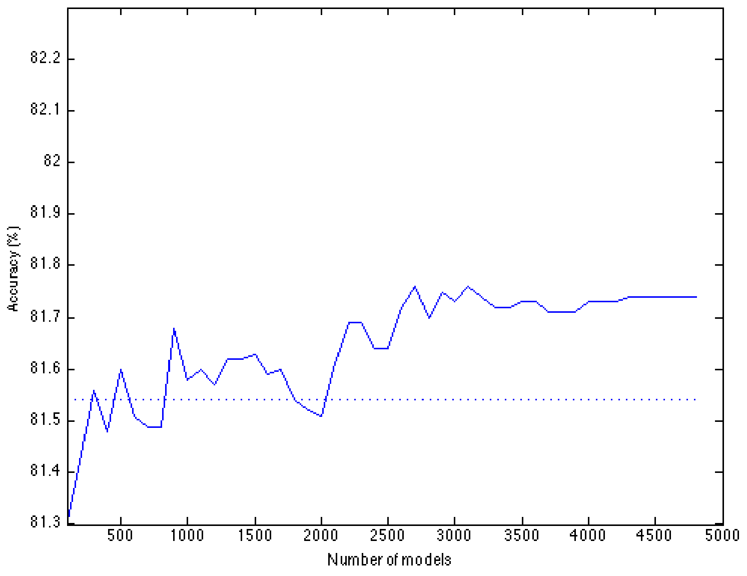
Then, we randomly extract 4800 sub-models from dropout/non-dropout network respectively, find the optimal weight l by solving the convex optimization problem using different number of sub-models. The plot of the accuracy on test set versus number of sub-models is shown in Figure 2. The dotted line in both figures are the accuracy of the original model, which is the approximation of geometric mean of all possible sub-models.

Clearly, the accuracy tend to increase with the number of sub-models used. This is intuitive: the sub-models are chosen randomly, the more sub-models we use, the more probable we have useful sub-models and better combinations. As is shown in Figure 2(a), for the model trained by dropout, the improvement is limited: approximately 0.2% improvement is achieved. However, for model trained without dropout, the improvement by weighted combination of sub-models is much more significant: the accuracy of original model is 79.52%, and the weighed combination of sub-models is always better than the original model, even when there are only 100 sub-models. The improvement is up to ???? when the number of sub-models reaches to ???.

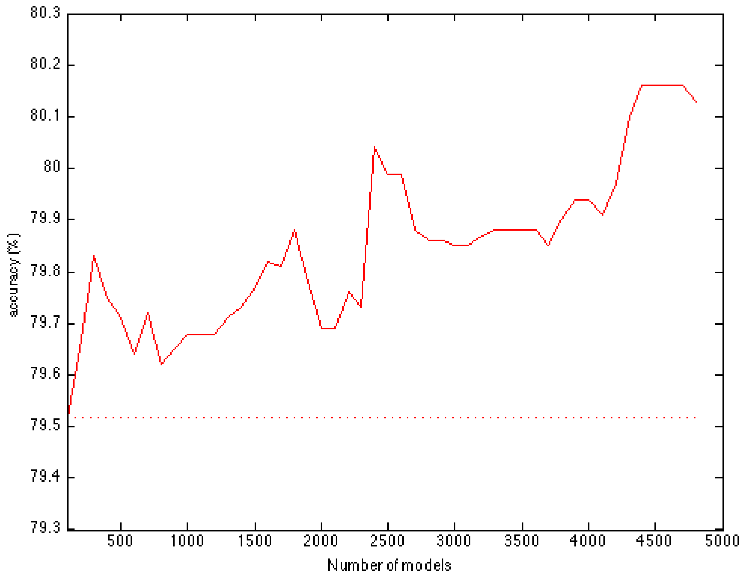
4.2 The role of dropout

The reason of the lower improvement for dropout network is explained in this section. Dropout training trains different sub-model at a time, but the weights are shared by all the sub-models. This procedure forces the network to learn more robust features that are useful for many different random sub-models, and after training, the sub-models in the dropout network can give similar results for

[2]http://cvxr.com



(a) Accuracy-number of sub-models for dropout network



(b) Accuracy-number of sub-models for nondropout network

Figure 2: Accuracy-number of sub-models.

Network	Accuracy on test set (%)
Dropout	81.540
Non-dropout	79.517

Table 1: Accuracy of networks

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

all the data. This is the eventual goal of the dropout training. In other words, the sub-models in the dropout network are similar to each other, therefore whatever combination of them would not achieve much improvement. On the other hand, the network trained without dropout doesn't have the invariant property, thus the sub-models are quite different to each other, leading to greater improvement by combining them.

Figure 3 illustrates this effect of dropout training clearly. accuracies achieved by 4800 sub-models for nondropout/ dropout network are shown in the two histograms. Clearly, the sub-models of dropout network are much more concentrated than those of the nondropout network. This shows the larger variance of the sub-models in nondropout network.

5 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 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.

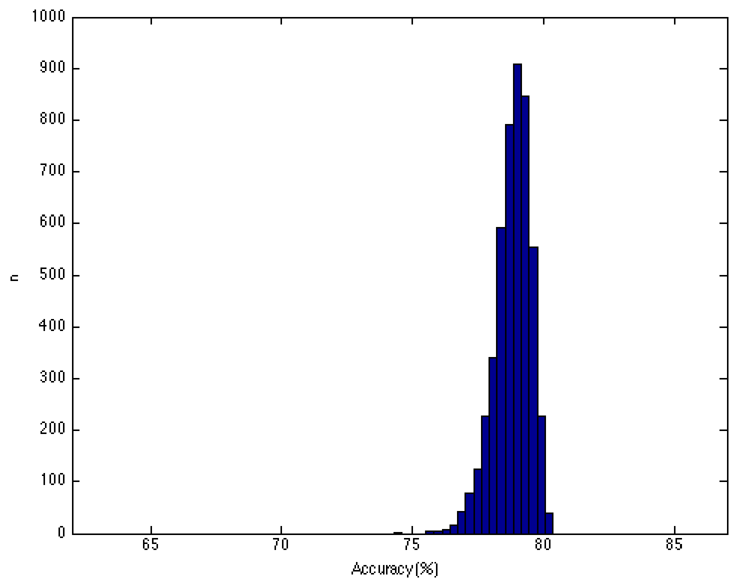
References

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

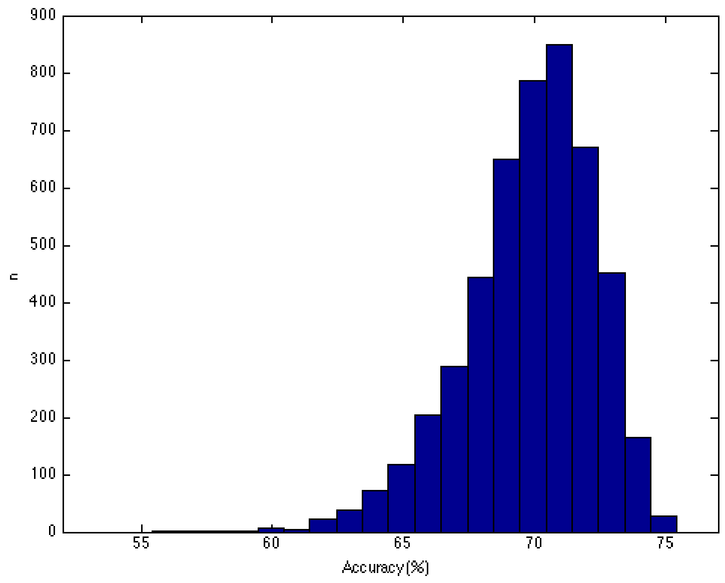
[2] Y. LeCun, L. Bottou, G.O., Muller, K.: Efficient backprop. Neural Networks: Tricks of the trade (1998)

[3] Bergstra, J., Breuleux, O., Bastien, F., Lamblin, P., Pascanu, R., Desjardins, G., Turian, J., Warde-Farley, D., Bengio, Y.: Theano: a CPU and GPU math expression compiler. In: Proceedings of the Python for Scientific Computing Conference (SciPy). (2010)

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



(a) Histogram of accuracies of sub-models of dropout network



(b) Histogram of accuracies of sub-models of nondropout network

Figure 3: Histograms of sub-models.