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

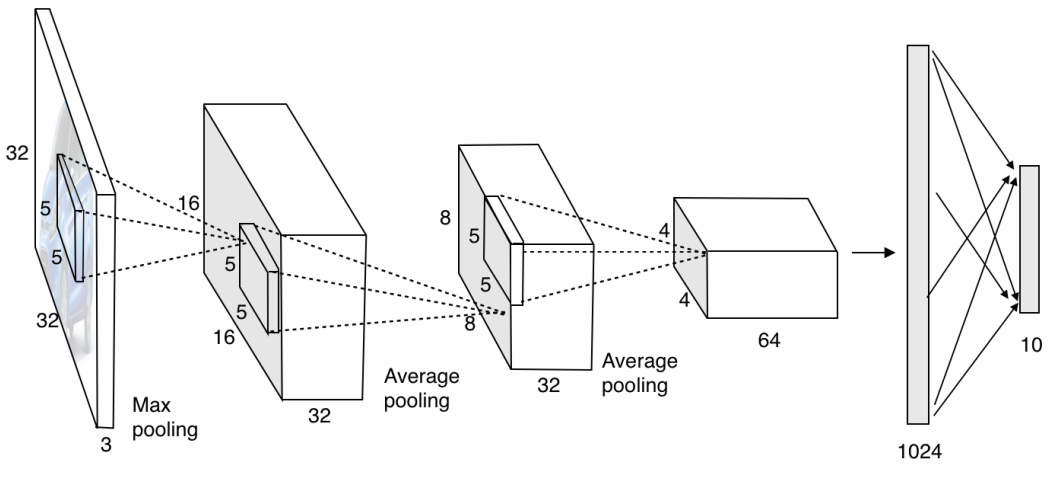

Figure 1: The architecture of our CNN.

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:

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

2.4 Dropout and sub-models

3 Convex Loss functions

Deep neural network is well-known for its highly non-convexity. But we can still borrow some ideas from the convex optimization literature, where people have found many useful convex loss functions for classification tasks. In this section, we will talk about three different convex loss functions, and discuss how to use them in multiclass classification problem within the deep learning architecture.

3.1 Cross Entropy

Cross entropy (CE) is the most standard loss function for neural networks, which is defined by:

$$H(p, q) = -\sum_{x} p(x) log q(x), \tag{9}$$

where p and q are ground-truth posterior probability and estimated posterior probability respectively. CE can be directly used for multiclass classfication. This has been discussed in Section 2.1.

3.2 Hinge Loss

Hinge loss is widely used in Support Vector Machines (SVMs). Given training data and its corresponding labels (x_n, y_n), $n = 1, ..., N$, $x_n \in R^D$, $t_n \in -1, +1$, for a binary classification problem, SVMs learning includes the following convex optimization:

$$\begin{array}{lll} \min\limits_{w,\psi} & \frac{1}{2} w^T w + C \sum_{n=1}^{N} \psi_n & \\ \text{s.t.} & w^T x_n t_n & \geq \quad 1 - \psi_n \forall n \\ & \psi_n & \geq 0 \quad \forall n \end{array} \tag{10}$$

Hinge loss, while doesn't have a clear probability interpretation as CE, can be regarded as a maximum margin object function. And it can be directly used in the deep learning architecture by deriving the corresponding unconstraint optimization problem:

$$\min_{w} \quad \frac{1}{2} w^T w + C \sum_{n=1}^{N} max(1 - w^T x_n t_n, 0) \tag{11}$$

The second term of the object function is the Hinge loss. For the first term, there is a specific term called Weight Decay in the deep learning literature(which can also be seen as L2 regularization). The derivative w.r.t x is:

$$\frac{\partial l(x)}{\partial x_n} = -C t_n (\mathbb{I}(1 > w^T x_n t_n)), \tag{12}$$

which is fast to compute.

Another similar loss function is the squared hinge loss:

$$\sum_{n=1}^{N} max(1 - w^T x_n t_n, 0)^2, \tag{13}$$

which is also differentiable and easy to compute. It imposes a bigger loss for points which violate the margin.

To predict the class label, for a binary classification problem, it's simply:

$$\begin{cases} +1, if \quad w^T x > 0 \\ -1, \quad otherwise \end{cases} \tag{14}$$

3.3 Multiclass Deep Neural Network with Hinge Loss

Many methods have been proposed to generaliza SVM to deal with multiclass problem. Here we talk about two most popular ways: One-Versus-All method and One-Versus-One method.

3.3.1 One-Versus-All

This is the earliest method for SVM multiclass classification. The basic idea is to train K SVM models where K is the number of classes. The ith SVM uses the data with label i as the positive samples and the data with other labels as the negative samples. Interestingly, this idea can be easily implemented in deep learning architecture. We can have K output units on top of the penultimate layer with the hinge loss. Then each output unit corresponds to one of these One-Versus-All SVMs. We can also regard this as replacing the softmax and cross-entropy layer with the hinge loss layer. This kind of method works well for problem with a small number of classes. When the class number grows larger, the imbalance between positive and negative samples for each SVM becomes intolerable.

3.3.2 One-Versus-One

The One-Versus-One method is to construct $K(K-1)/2$ classifiers. Each is responsible for one of the $K(K-1)/2$ class pairs. While testing, we simply let all the binary classifiers vote for the most probable one, which is also called the "Max Wins" strategy.To incorporate this method to deep learning architecture, we can first have $K(K-1)/2$ output units, each of which corresponds to one classifier. All of the units use hinge loss. For each training data, only $K-1$ of these units are activated. The other units are temperorily cut off. This is illustrated in Fig. The good property here is that for each SVM, the positive and negative training samples are balanced, no matter how many classes there are. However, The number of SVMs needed are quadratic in K, which makes it not scalable.

3.4 Ranking Loss

Another way to directly model the classification problem is the pairwise-ranking loss. Since we just want to pick the one with the highest linear output at test time. This lead to the following minimization problem:

$$Loss = \sum_{n=1}^{N} \sum_{i \in C_n^*} max(0, 1 - f_j(x_n + f_i(x_n))), \tag{15}$$

where j is the label of the nth data and C_n^* indicates the set of all the other labels. This loss can be directly generalized to multilabel predition problem and is widely used in many recommandation systems.

4 Experiments

4.1 Sub-model convolutional network

4.2 Convolutional network with different convex loss functions

We show the experiment results for using different convex loss functions, including softmax and cross-entropy(SCE), One-Versus-One with hinge loss(1v1HINGE), One-Versus-All with hinge loss(1vaHINGE) and rank loss(RANK). This set of experiments are implemented using Caffe. And the code is released on Si's Github.

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)