IIIS 2018 Spring: ATCS - Selected Topics in Optimization
Lecture date: May 7, 2018
Instructor: Jian Li **Scribe: Dingli Yu**

1 Attention

Continue from the last lecture, we introduce some important implementation of attentions. This section is based on Prof. Li's lecture slides Attentions II [3].

Instead of using RNN to do sequence to sequence learning, Gehring et al. [1] propose an architecture based on CNN and attention model. Figure 1 gives the structure of their model. They use multi-step attention in their framework, which allows the decoder immediate access to the attention history of the $k-1$ previous time steps (k is the number of layers). Under their model, it is easier to relate words by corresponding distance.

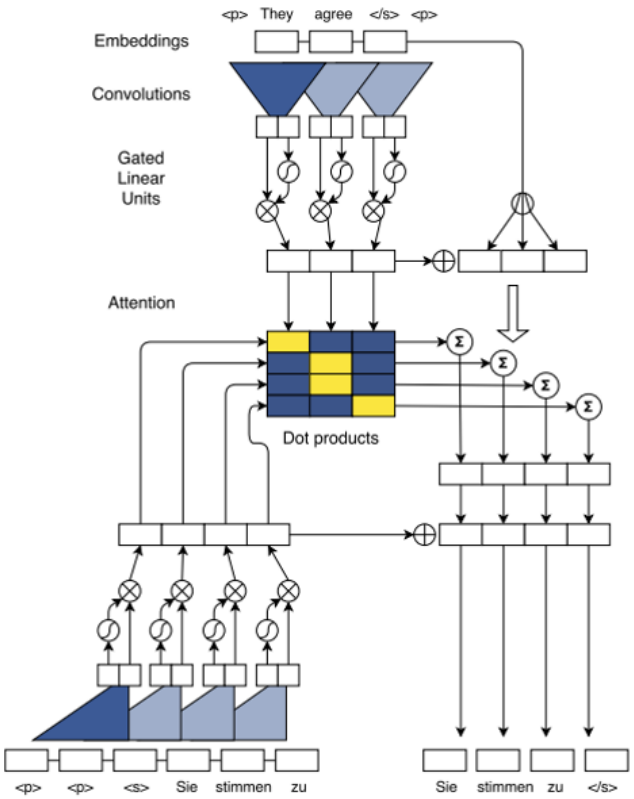

Figure 1: Structure of convolutional seq2seq.

Jumping out of sequence to sequence model, Vaswani et al.'s paper "attention is all you need" [7] uses a multi-head self-attention mechanism to solve problems in NLP. The key value attention

is used, that is,

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V,$$

where Q is a set of queries with dimension d_k, K is a set of keys with dimension d_k as well, and V is a set of values with dimension d_v. (The dot product of Q and K gives how close the queries and keys are.) Besides of that, they also use multi-head attention, which is defined as

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O,$$

where

$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V).$$

Their network architecture, Transformer, is shown in Figure 2. In the Encoder side(left hand side of the figure), they use the same Q, K, V as the input of multi-head attention. This self-attention strategy helps coreference resolution. For example, comparing the sentence "the animal didn't cross the street because it was too tired" with "the animal didn't cross the street because it was too wide", "it" refers to different things ("animal" or "street") based on the last words ("tired" or "wide"). It is worth noting that the Add and Norm layer is very important for network training and convergence. It basically implements skip connection and layer normalization. On the other hand, when the network decodes, it uses Q and K from the encoder network, and uses masked multi-head attention for the first attention layer. Since one should not know any future information in decoder phase, they set all unknown queries and keys to $-\infty$ in masked multi-head attention.

2 Normalization

As we mentioned in last section, normalization helps neural network to converge. Particularly, CNN starts to converge after Batch Normalization (BN). This section is based on Prof. Li's lecture slides Deep Learning, Normalization [4].

BN normalizes each mini-batch of data. Essentially, it transforms $x^{(k)}$ to

$$\hat{x}^{(k)} = \frac{x^{(k)} - E[x^{(k)}]}{\sqrt{\text{Var}(x^{(k)})}}.$$

One may ask why BN useful? In statistical learning, we assume the training distribution should be the same as the test distribution. However, if we regards each layer in networks as a learning problem, then the training distribution may not be the same as testing distribution, because any changes in previous layers cause and amplify the change of input distribution. People refer this phenomenon as internal covariance shift, and believe that BN can reduce this effect[2].

Since BN usually requires larger mini-batch size, it is not easy to be applied to RNN. It may be possible, but no conclusion has been made until now.

Besides BN, people also do layer normalization, which is a summation over (C, H, W) in CNN. Moreover, we can take any summation in any groups, which is called group normalization.

3 Transfer Learning and Multi-task Learning

This section is based on Prof. Li's lecture slides Transfer Learning, Multi-task Learning [5].

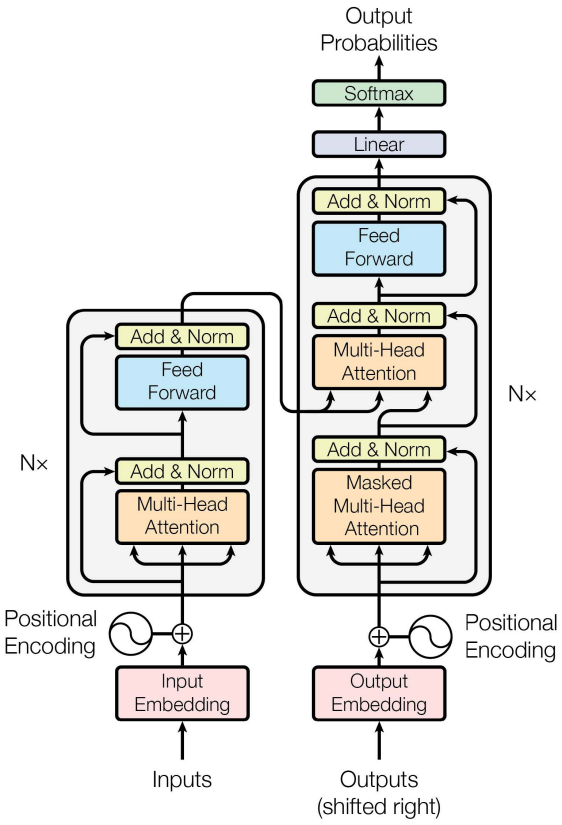

Figure 2: Transformer.

In a standard machine learning framework, data must be properly chosen. But what should we do if we want to apply a classifier to a new or shifting domain? It would be too expensive to retrain the model. People are trying to use existing classifier as a starting point to make a shortcut, which is called transfer learning.

There are two types of differences between the two problem. One is domain differences, that is, the data set may be different or of different distribution. Another is task differences, that is, the labeling set may be different or of different distribution. For different data distribution, Adaboost naturally fits transfer learning since it reduces effect of "bad" instances and encourages effect of "good" instances in each iteration.

Multi-task learning uses similar idea as transfer learning. When learning a hard task with insufficient data, we can use some auxiliary task to help training. Auxiliary task are typically homogeneous tasks, and can provide additional supervised information and additional structural knowledge if we train these two tasks jointly.

For example, in travel time estimation, we are given path, start time, driver, and we need to predict the travel time. However, the given data is actually a sequence of GPS points. Therefore, in stead of only estimating the entire path, we can estimating some local paths as an auxiliary task. In this way, the number of training samples are increased, and the structure is better understood.

4 Generalization

Along the path to understand machine learning, two major issues remain unknown — generalization and optimization, especially in deep learning. For optimization, researchers wonder how to calculate the optimal parameters. As far as we know, computing global minimum is impossible in neural networks, which leaves us a new problem of finding local minimum in the non-convex setting. One may think gradient descent is enough for finding local minimal. However, points with zero gradient are not necessarily local minimums, but can also be saddle points.

In this lecture, we mainly discuss generalization, which refers to the problem of avoiding "overfitting". We hope the model we get from training can reach similar performance in other data samples. Therefore, in machine learning, especially in training deep neural networks, we want to find local minimum that (i) has similar loss as global minimum and (ii) generalize well.

4.1 Notations

Consider a machine learning problem with data set D, for a *training sample* $S = \{z_1, \ldots, z_n\}$ where each z_i is independent and identically draw from D, we define the *empirical risk* or the *empirical loss* of a model with parameter w to be

$$R_S[w] \triangleq \frac{1}{n} \sum_{i=1}^{n} f(w, z_i).$$

Empirical loss is what we often optimize in training, but what we really care about is the *population loss*,

$$R[w] = \mathop{\mathbb{E}}_{z \sim D}[f(w, z)].$$

Here we take expectation on $z \sim D$, which is often called the *test sample*.

Typically, after running a randomized training algorithm A, the empirical loss would be smaller than the population loss, and we define the *generalization gap* or the *generalization error* to be the expected difference, i.e.,

$$\epsilon_{\mathsf{gen}} = \left| \mathop{\mathbb{E}}_{S,A}[R[A(S)] - R_S[A(S)]] \right|,$$

where the expectation is taken on $S \overset{\text{i.i.d.}}{\sim} D$ and randomness of algorithm A, while $A(S)$ is the model learned by A on S.

From the definition, one can see that as long as $R_S[A(S)]$ is small and ϵ_{gen} is small, $R[A(S)]$ should be small.

4.2 Algorithmic stability

Many students may have learned Vapnik uniform convergence bound (a.k.a. VC theory) as one of the most important generalization bounds. Unfortunately, it does not work for deep learning. For VC and other complexity theory, the size of the training set should be the same order of the number of parameters, while for deep learning, the number of parameters are often much much larger than the size of training data.

Here, we offer a highly different approach to guarantee generalization — via algorithmic stability. We will define uniform stability of an algorithm below, and then prove the uniform stability of an

algorithm implies its generalization gap. In the next lecture, we will prove the uniform stability of SGD, and thus automatically gives its generalization bound. Other optimization algorithms may not have this properties and fail to generalize well. For example, Adam has been proved failing to converge in some circumstances by one of three ICLR'18 best papers [6].

Definition 1 (Uniform stability) *A random algorithm A is ϵ-uniform stable if and only if for all S and S' (such that S and S' differ in only one sample),*

$$\sup_z \mathop{\mathbb{E}}_A[|f(A(S), z) - f(A(S'), z)|] \leq \epsilon.$$

The definition basically says that if an algorithm has good uniform stability, then the loss for any sample z should not change too much if we change only one training sample.

Theorem 2 *Suppose an algorithm A is ϵ-uniform stable, then*

$$\epsilon_{\mathsf{gen}} = \left| \mathop{\mathbb{E}}_{S,A}[R[A(S)] - R_S[A(S)]] \right| \leq \epsilon.$$

Proof: Let $S = \{z_1, \ldots, z_n\}$, $S' = \{z'_1, \ldots, z'_n\}$, and $z_i, z'_i \overset{\text{i.i.d.}}{\sim} D$ for all $i \in \{1, \ldots, n\}$.

Let $S^{(i)} = \{z_1, \ldots, z_{i-1}, z'_i, z_{i+1}, \ldots, z_n\}$, then

$$\begin{aligned}
\mathop{\mathbb{E}}_{S \overset{\text{i.i.d.}}{\sim} D} \mathop{\mathbb{E}}_A[R_S(A(S))] &= \mathop{\mathbb{E}}_S \mathop{\mathbb{E}}_A \left[\frac{1}{n} \sum_{i=1}^n f(A(S), z_i) \right] \\
&= \frac{1}{n} \sum_{i=1}^n \mathop{\mathbb{E}}_S \mathop{\mathbb{E}}_A [f(A(S), z_i)] \\
&= \frac{1}{n} \sum_{i=1}^n \mathop{\mathbb{E}}_S \mathop{\mathbb{E}}_{z'_i \sim D} \mathop{\mathbb{E}}_A \left[f\left(A\left(S^{(i)} \right), z'_i \right) \right] \\
&= \mathop{\mathbb{E}}_S \mathop{\mathbb{E}}_{S'} \mathop{\mathbb{E}}_A \left[\frac{1}{n} \sum_{i=1}^n f\left(A\left(S^{(i)} \right), z'_i \right) \right] \\
&\leq \mathop{\mathbb{E}}_S \mathop{\mathbb{E}}_{S'} \mathop{\mathbb{E}}_A \left[\frac{1}{n} \sum_{i=1}^n f(A(S), z'_i) \right] + \epsilon \\
&= \mathop{\mathbb{E}}_S \mathop{\mathbb{E}}_A \left(\mathop{\mathbb{E}}_{z \sim D} [f(A(S), z)] \right) + \epsilon \\
&= \mathop{\mathbb{E}}_{S,A} [R(A(S))] + \epsilon,
\end{aligned}$$

where the inequality is by the uniform stability. □

References

[1] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. Convolutional sequence to sequence learning.CoRR, abs/1705.03122, 2017.

[2] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network trainingby reducing internal covariate shift.arXiv preprint arXiv:1502.03167, 2015.

[3] Jian Li. Attentions II. ATCS: Learning and Optimization - theory and practice, 2018 Spring.

[4] Jian Li. Deep Learning, Normalization. ATCS: Learning and Optimization - theory and practice, 2018 Spring.

[5] Jian Li. Transfer Learning, Multi-task Learning. ATCS: Learning and Optimization - theory and practice, 2018 Spring.

[6] Sashank J. Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of adam and beyond. International Conference on Learning Representations, 2018.

[7] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. InAdvances in Neural Information Processing Systems, pages 6000–6010, 2017.