A fast and simple algorithm for training neural probabilistic language models

Andriy Mnih & Yee Whye Teh
Gatsby Computational Neuroscience Unit
University College London

Overview

- In spite of their superior performance, neural probabilistic language models (NPLMs) are far less widely used than n-gram models due to their notoriously long training times.
- We introduce a simple training algorithm for NPLMs based on noise-contrastive estimation, with **time complexity independent of the vocabulary size**.
  - Over an order of magnitude faster than maximum-likelihood estimation.
  - The resulting models perform just as well.
- We demonstrate the algorithm's scalability by training several large neural language models on the **MSR Sentence Completion Challenge** dataset, achieving **state-of-the-art** results.

Statistical language modelling

- Goal: Model the joint distribution of words in a sentence.
- Applications: speech recognition, machine translation, information retrieval.
- **Markov assumption**:
  - The distribution of the next word depends only on k words that immediately precede it.
  - Though clearly false, the assumption makes the task much more tractable without making it trivial.

n-gram models

- Task: predict the **next word** w_n from $n-1$ preceding words $h = w_1, ..., w_{n-1}$ (called the **context**).
- n-gram models are conditional probability tables for $P(w_n|h)$.
  - Estimated by smoothing word n-tuple counts.
  - Most widely used statistical language models due to their simplicity and good performance.
- Cannot take advantage of similarity between words / contexts.
- Curse of dimensionality:
  - The number of model parameters is exponential in the context size.
  - Cannot take advantage of large context sizes.

Neural probabilistic language models

- Neural probabilistic language models use **distributed representations** of words to deal with the curse of dimensionality.
  - Words are represented with real-valued feature vectors learned from data.
  - A neural network maps contexts (sequences of word feature vectors) to next word distributions.
  - Word feature vectors and neural net parameters are learned jointly.
- NPLMs generalize well because smooth functions map nearby inputs to nearby outputs.
- Similar representations are learned for words with similar usage patterns.
- Main drawback: very long training times.

Training neural language models

- A NPLM quantifies the compatibility between a context h and a candidate next word w using a scoring function $s_\theta(w, h)$.
- The distribution for the next word is defined in terms of scores:

$$P_\theta^h(w) = \frac{1}{Z_\theta(h)} \exp(s_\theta(w, h)),$$

where $Z_\theta(h) = \sum_{w'} \exp(s_\theta(w', h))$.

Maximum-likelihood estimation

- The gradient of the log-likelihood is

$$\frac{\partial}{\partial\theta} \log P_\theta^h(w) = \frac{\partial}{\partial\theta} s_\theta(w, h) - \frac{\partial}{\partial\theta} \log Z_\theta(h)$$
$$= \frac{\partial}{\partial\theta} s_\theta(w, h) - \sum_{w'} P_\theta^h(w') \frac{\partial}{\partial\theta} s_\theta(w', h).$$

- Computing $\frac{\partial}{\partial\theta} \log Z_\theta(h)$ is expensive – **the time complexity is linear in the vocabulary size**.
- Can approximate $\frac{\partial}{\partial\theta} \log Z_\theta(h)$ using importance sampling (Bengio and Senécal, 2003):
  - Sample words from a proposal distribution and reweight the gradients.
  - **Stability issues**: need either a lot of samples or an adaptive proposal distribution.

Noise-contrastive estimation

- Idea: **Fit a density model by learning to discriminate between samples from the data distribution and samples from a known noise distribution** (Gutmann and Hyvärinen, 2010).
- If noise samples are k times more frequent than data samples, the posterior probability that a sample came from the data distribution is

$$P^h(D = 1|w) = \frac{P_d^h(w)}{P_d^h(w) + kP_n(w)}.$$

- To fit a model $P_\theta^h(w)$ to the data, use $P_\theta^h(w)$ in place of $P_d^h(w)$ and maximize $J^h(\theta) =$

$$E_{P_d^h}\left[\log \frac{P_\theta^h(w)}{P_\theta^h(w) + kP_n(w)}\right] + kE_{P_n}\left[\log \frac{kP_n(w)}{P_\theta^h(w) + kP_n(w)}\right].$$

- **NCE allows working with unnormalized distributions** $P_{\theta^0}^{h0}(w)$.
  - Set $P_\theta^h(w) = P_{\theta^0}^{h0}(w)/Z^h$ and **learn** Z^h.
  - θ^0 are the parameters of the unnormalized distribution and $\theta = \{\theta^0, \log Z^h\}$.
- The gradient of the objective for context h is

$$\frac{\partial}{\partial\theta} J^h(\theta) = E_{P_d^h}\left[\frac{kP_n(w)}{P_\theta^h(w) + kP_n(w)} \frac{\partial}{\partial\theta} \log P_\theta^h(w)\right] -$$
$$kE_{P_n}\left[\frac{P_\theta^h(w)}{P_\theta^h(w) + kP_n(w)} \frac{\partial}{\partial\theta} \log P_\theta^h(w)\right].$$

- **Much easier to estimate than the importance sampling gradient** because the weights on $\frac{\partial}{\partial\theta} \log P_\theta^h(w)$ are always between 0 and 1.
  - Can use far fewer noise samples as a result.
- The global NCE objective is a sum of the per-context objectives weighted by the empirical context probabilities $P(h)$:

$$J(\theta) = \sum_h P(h) J^h(\theta).$$

Speedup over MLE

The NCE parameter update is $\frac{cd+V}{cd+k}$ **times faster than the ML update**.

- Here c is the context size, d is the feature vector dimensionality, V is the vocabulary size, and k is the number of noise samples.

Penn Treebank results

Data: news stories from Wall Street Journal
- Training/validation/test set: 930K/74K/82K words
- Vocabulary: 10K words

Training alg.	Num. of samples	Training time (h)	PPL w. noise unigram	PPL w. noise uniform
ML		21	**163.5**	163.5
NCE	1	1.5	192.5	291.0
NCE	5	1.5	172.6	233.7
NCE	25	1.5	**163.1**	195.1
NCE	100	1.5	**159.1**	173.2

Sentence completion results

Task: given a sentence with a missing word find the correct completion from a list of candidate words.
- Training set: 522 19th-century novels (48M words)
- Test set: 1,040 sentences from five Sherlock Holmes novels
- Five candidate completions per sentence.

Method	Context size	Latent dim	Test PPL	Percent correct
Chance	0			20.0
3-gram	2		130.8	36.0
5-gram	4		121.5	38.7
6-gram	5		121.7	38.4
LSA	sentence	300		49
RNN	sentence	?	?	45
LBL	2	100	145.5	41.5
LBL	3	100	135.6	45.1
LBL	5	100	129.8	49.3
LBL	10	100	124.0	50.0
LBL	10	200	117.7	52.8
LBL	10	300	116.4	**54.7**
LBL	10×2	100	38.6	44.5

Conclusions

Noise-contrastive estimation provides a fast and simple way of training neural language models:
- Over an order of magnitude faster than maximum-likelihood estimation.
- Models trained using NCE with 25 noise samples per datapoint perform as well as the ML-trained ones.