CSC2515 FALL 2008 INTRODUCTION TO MACHINE LEARNING

APPLICATIONS OF MACHINE LEARNING TO LANGUAGE MODELING

Andriy Mnih

Statistical language modelling

- Goal: Model the joint distribution of words in a sentence.
- Such a model can be used to
 - predict the next word given several preceding ones
 - arrange bags of words into sentences
 - assign probabilities to documents
- Applications: speech recognition, machine translation, information retrieval.
- Most statistical language models are based on the Markov assumption:
 - The distribution of the next word depends on only *n* words that immediately precede it.
 - This assumption is clearly wrong but useful it makes the task much more tractable.

n-gram models

- *n*-gram models are simply conditional probability tables for $P(w_n|w_{1:n-1})$.
 - estimated by counting *n*-tuples of words and normalizing
 - smoothing the estimates is essential for good performance
 - many different smoothing methods exist
- *n*-gram models are the most widely used statistical language models due to their simplicity and excellent performance.
- Curse of dimensionality: number of model parameters is exponential in *n*.

Training *n***-gram models**

- Let #*s* be the number of times a sequence of words *s* occurs in the training set.
- Then we can estimate a trigram model as follows:

$$P(w_3|w_1, w_2) = \frac{\#w_1w_2w_3}{\#w_1w_2}$$

- Problem: if $w_3w_2w_1$ does occur in the training set, it is assigned zero probability.
- That's bad the model does not generalize to new word triples!
- One solution: smooth the trigram estimates by interpolating them with the bigram estimates

$$P(w_3|w_1, w_2) = \lambda \times \frac{\#w_1w_2w_3}{\#w_1w_2} + (1-\lambda) \times \frac{\#w_2w_3}{\#w_2}$$

• Can also smooth with the unigram estimates and the uniform distribution.

Why *n***-gram models are hopeless for large** *n*

- *n*-gram models don't take advantage of the fact that some words are used in similar ways.
- Suppose you know that words *snow* and *rain* are used in similar ways, as are *Monday* and *Tuesday*.
- If you are told that the following sentence is probable:
 - -*It's going to rain on Monday.*
- Then you can infer that the following sentence is also probable:
 - -*It's going to snow on Tuesday.*
- *n*-gram models cannot generalize this way because all words are treated as arbitrary symbols, with each word being equally (dis)similar to all others.
- Using distributed representations for words allows similarity between words to be captured.

Distributed representations

- Estimation of high-dimensional discrete distributions from data is hard.
 - the number of parameters is exponential
 - no a priori smoothness constraint on parameters / probabilities
- Estimation of distributions over continuous spaces is easier due to automatic smoothing.
- Idea: map discrete inputs to continuous vectors and learn a smooth function that maps them to probability distributions.
- Used for language modelling with neural nets and Bayes nets.

Word representations embedded in 2D (I)

•part •hours •hours miles •^{tv}television days montheweek more trade •years sen •much •radio economic •prime_juit_verture •presspolitical economic •pressident •he father never •evidence •chairmawile of service day •evidence •chairmawife •reports •leader •head director •who chief •life national northern anti police million billion authorities nato former bospiantepublican officialicate •an •u.n •israeli •iserb •iussian . verv •groups •second five two •groups •second •six, eigh •members •first seven •power •town id •six eight sevenne #n •four •#\$n too • •his name •pay •support and •high •time •'s •federal •armyeconorficcord •force governmention half •#n •just ⁺•nearlv •one •court •hospital tital electionampaign which soldiers of a sol •a_topmeonlyat least about many •something •at most several trial placews workey tims trial voie cancicate and the superior of the superio a_few • office Pales tion • jobs •anothe •association of the second se another •) •owline of the second s local daily
 foreign
 food ioregi i iood i john iplanscompany/rjöjskur
 courtigerene white_freeseatkighting ied interference
 city state healthtupeblic interference
 city state healthtupeblic interference
 courty security scatterenerse interference
 courtigerence
 senate house back
 home ۰to congrest tax...
 boshigamerica u.s. beyar.
 israel information
 tax administration washingther
 tax administration washingther
 tax administration texasil formet york
 browstexander washingther
 tax administration texasil formet york
 browstexander washingther
 tay administration texasil formet york
 tay administration texasil formet york even •has •had •would might could didn't •mutere •#na •have ,•homes •Show along Within • area • law • system • mrs •mustolesnitgoing_to •should •don't •outside through •program •programs •david Ike
 Across
 Such_astween
 Such_astween
 Such_astween
 Such_astween
 Calleghange
 Calleghange
 Sooff herethere
 around •near •across ۰r such the including at calleer including at calleer inst for without keegeven use around with getput grough in held state •ifwhether •against for •whybecause •believe •believe •thicksays •thicky said •then •think* •still already •magle • are • were • is • has been • was.• would be • was.• being • become •reported •saying ouncedast ;re m •do •did •report •newspapehimself •really report •SO toldasked •saying •released killed e •however arrested • late •out •wanted •does •times ever •such ۰. may •tried •ago , be trying accused been down based édhenday jedhenday sunday , march april •y

Word representations embedded in 2D (II)



Distributed / neural language models

- A number of neural probabilistic language models based on distributed representations have been proposed.
- Common approach:
 - Represent each word with a real-valued feature vector
 - Represent the context by the sequence of the context word feature vectors
 - Train a neural network to output the distribution for the next word from the context representation.
 - Learn word feature vectors jointly with other neural net parameters
- Neural language models can outperform *n*-gram language models, especially when little training data is available.
- Main drawback: very long training and testing times.

Neural Probabilistic Language Model (Bengio et al., 2000)

- The original and still the most popular neural language model.
- A lookup table is used to map context words to feature vectors.
- Architecture: 1-hidden layer neural net
 - Input: sequence of the context word feature vectors.
 - -Output: distribution over the next word (softmax over words).
- Outperforms *n*-gram models on small (~ 1M words) datasets.
- For better results, predictions of a NPLM are interpolated with those of an *n*-gram model.

Neural Probabilistic Language Model



Log-bilinear model (Mnih & Hinton, 2007)

- The LBL model is similar to the NPLM, but is simpler and slightly faster.
 - Does not have non-linearities.
- Given the context $w_{1:n-1}$, the LBL model predicts the representation for the next word w_n by linearly combining the representations for the context words:

$$\hat{r} = \sum_{i=1}^{n-1} C_i r_{w_i}$$

• Then the distribution for the next word is computed based on the similarity between the predicted representation and the representations of all words in the vocabulary:

$$P(w_n = w | w_{1:n-1}) = \frac{\exp(\hat{r}^T r_w)}{\sum_j \exp(\hat{r}^T r_j)}.$$

Structuring the vocabulary

- Computing the probability of the given word being the next word requires considering all *N* words in the vocabulary.
 - -Need to normalize over all words because the space of words is unstructured.
- Idea (due to Bengio): Organize words in the vocabulary into a (somewhat balanced) binary tree and exploit its structure to speed up normalization.
 - Construct a binary tree over words
 * words are associated with leaf nodes
 * one word per leaf
 - Predicting the next word: replace one *N*-way decision by a sequence of O(log N) two-way decision.
 * Can achieve exponential speedup!

Tree-based factorization



- To define a distribution over leaf nodes:
 - -Specify the probability of taking the left branch at each non-leaf node.
 - Then the probability of a leaf node is simply the probability of the sequence of left/right decisions that lead from the root node to the leaf node.

Approaches to tree construction

- The approach of Morin and Bengio:
 - Start with the WordNet IS-A hierarchy (which is a DAG)
 - Manually select one parent node per word
 - Use clustering to make the resulting tree binary
 - Use the NPLM model for making the left/right decisions
- Drawbacks: tree construction uses expert knowledge; the resulting model does not work as well as its non-hierarchical counterpart.
- An alternative (Mnih & Hinton, 2008):
 - Construct the word tree from data alone (no experts needed)
 - Allow each word to occur more than once in the tree
 - Use the simplified log-bilinear language model for making the left/right decisions

Hierarchical log-bilinear model (Mnih & Hinton, 2008)

- Let *d* be the binary string / code that encodes the sequence of left-right decisions in the tree that lead to word *w*.
- Each non-leaf node in the tree is given a feature vector that captures the difference between the words in its left and right subtrees.
- The probability of taking the left branch at a particular node is given by

$$P(d_i = 1 | q_i, w_{1:n-1}) = \sigma(\hat{r}^T q_i),$$

where \hat{r} is computed as in the LBL model and q_i is the feature vector for the node.

• Then the probability of word *w* being the next word is simply the probability of *d* under the binary decision model:

$$P(w_n = w | w_{1:n-1}) = \prod_i P(d_i | q_i, w_{1:n-1}).$$

Data-driven tree construction

- We would like to cluster words based on the distribution of contexts in which they occur.
- This distribution is hard to estimate and work with due to the high dimensionality of the space of contexts (the same sparsity problem *n*-gram models suffer from).
- To avoid this problem, we represent contexts using distributed representations and cluster words based on their *expected* context representation.
- To construct a word tree:
 - 1. Train a model using a random (balanced) tree over words.
 - 2. Compute the expected predicted representation over all occurrences of the given word.
 - 3. Perform hierarchical clustering on these expected representations.

Hierarchical clustering

- We "cluster" the feature vectors using top-down hierarchical clustering.
- At each step, we fit a mixture of two Gaussians with spherical covariances using EM to the current group of word representations.
- Once the mixture has been fit, we assign the words to the two components based on the mixture component responsibilities.
- We considered several splitting rules:
 - BALANCED: Sort the responsibilities and make the split to ensure a balanced tree.
 - ADAPTIVE: Assign the word to the component with the greater responsibility.
 - ADAPTIVE(ϵ): Assign the word to a component if its responsibility for the word is at least 0.5- ϵ .

Dataset and evaluation

- We compared the models on the APNews dataset:
 - A collection of Associated Press news stories (16 million words)
 - Training/validation/test split: 14M/1M/1M words
- Preprocessing (Bengio):
 - convert all words to lower case
 - map all rare words and proper nouns to special symbols
 - Result: just under 18000 unique words.
- Models were compared based on the perplexity they assigned to the test set.
- Perplexity is the geometric average of $\frac{1}{P(w_n|w_{1:n-1})}$.

Random vs. non-random trees

The effect of the feature dimensionality and the tree-building algorithm on the test set perplexity of the model.

Feature	Perplexity using	Perplexity using	Reduction
dimensionality	a RANDOM tree	a BALANCED tree	in perplexity
25	191.6	162.4	29.2
50	166.4	141.7	24.7
75	156.4	134.8	21.6
100	151.2	131.3	19.9

Perplexity on the test set:

Model	Tree generating	Perplexity	Minutes
type	algorithm		per epoch
HLBL	RANDOM	151.2	4
HLBL	BALANCED	131.3	4
HLBL	ADAPTIVE	127.0	4
HLBL	ADAPTIVE(0.25)	124.4	6
HLBL	ADAPTIVE(0.4)	123.3	7
HLBL	ADAPTIVE(0.4) \times 2	115.7	16
HLBL	ADAPTIVE(0.4) \times 4	112.1	32
LBL	—	117.0	6420
KN3		129.8	_
KN5	—	123.2	_

- LBL and HLBL used 100D feature vectors and a context size of 5.
- KN*n* is a Kneser-Ney *n*-gram model.

Observations

- Hierarchical distributed language models can outperform non-hierarchical models when they use sufficiently well-constructed trees over words.
 - Expert knowledge is not needed for building good trees.
 - Allowing words to occur more than once in a tree is essential for good performance.
- Even when very large trees are used, the hierarchical LBL model is more than two orders of magnitude faster than the LBL model.

The END