Administrivia

How to ask questions outside lecture:

- Use Canvas! Read Canvas first to see if your question has already been answered!
- If you want confidentiality, or aren't yet registered, then use the instructors' mailing list (but note that TAs will all see it)

Try not to sweat quiz grades. We'll drop (at least) your lowest, and in the final calculation, most of the credit is for the attempt.

Natural Language Processing (CSE 447/547M): Neural Language Models (Wrap-Up) and Text Classification

Noah Smith

© 2019

University of Washington
`nasmith@cs.washington.edu`

January 23, 2019

Quick Review

A language model is a probability distribution over $\mathcal{V}^\dagger$.

Typically p decomposes into probabilities $p(x_i \mid \boldsymbol{h}_i)$.

- n-gram: $\boldsymbol{h}_i$ is $(\mathrm{n}-1)$ previous symbols; three ways to assign probabilities:
    - directly estimate each one: count and normalize (with smoothing)
    - share using *features* in a log-linear model
    - share using a neural network
- non-n-gram: recurrent neural network

Comparison: Classical Probabilistic vs. Connectionist Modeling

	Classical Probabilistic	**Connectionist**
What do we engineer?	features, assumptions	architectures
Theory?	as N gets large	not really
Interpretation of parameters?	often easy	usually hard

Parting Shots about Language Models

Parting Shots about Language Models

- I said very little about *estimating* the parameters.

Parting Shots about Language Models

- I said very little about *estimating* the parameters.
- For NNs, it's still essentially maximum likelihood estimation; this is equivalent to minimizing "log loss" and "cross-entropy loss."

Parting Shots about Language Models

- I said very little about *estimating* the parameters.
- For NNs, it's still essentially maximum likelihood estimation; this is equivalent to minimizing "log loss" and "cross-entropy loss."
  - At present, it's almost always stochastic gradient descent with heavy use of the chain rule from calculus ("backpropagation").

Parting Shots about Language Models

- I said very little about *estimating* the parameters.
- For NNs, it's still essentially maximum likelihood estimation; this is equivalent to minimizing "log loss" and "cross-entropy loss."
    - At present, it's almost always stochastic gradient descent with heavy use of the chain rule from calculus ("backpropagation").
    - Regularization and step size methods are important!

Parting Shots about Language Models

- I said very little about *estimating* the parameters.
- For NNs, it's still essentially maximum likelihood estimation; this is equivalent to minimizing "log loss" and "cross-entropy loss."
  - At present, it's almost always stochastic gradient descent with heavy use of the chain rule from calculus ("backpropagation").
  - Regularization and step size methods are important!
  - New libraries to help you are coming out all the time.

Parting Shots about Language Models

- I said very little about *estimating* the parameters.
- For NNs, it's still essentially maximum likelihood estimation; this is equivalent to minimizing "log loss" and "cross-entropy loss."
  - At present, it's almost always stochastic gradient descent with heavy use of the chain rule from calculus ("backpropagation").
  - Regularization and step size methods are important!
  - New libraries to help you are coming out all the time.
  - Many of them use GPUs to speed things up.

Parting Shots about Language Models

- I said very little about *estimating* the parameters.
- For NNs, it's still essentially maximum likelihood estimation; this is equivalent to minimizing "log loss" and "cross-entropy loss."
  - At present, it's almost always stochastic gradient descent with heavy use of the chain rule from calculus ("backpropagation").
  - Regularization and step size methods are important!
  - New libraries to help you are coming out all the time.
  - Many of them use GPUs to speed things up.
- This progression is worth reflecting on:

	history:	represented as:
before 1996	$(n-1)$-gram	discrete
1996–2003		feature vector
2003–2010		embedded vector
since 2010	unrestricted	embedded

Quick Aside

Important Idea: Words as Vectors

The idea of "embedding" words in $\mathbb{R}^d$ is much older than neural language models.
ou should think of this as a *generalization* of the discrete view of $\mathcal{V}$.

Important Idea: Words as Vectors

The idea of "embedding" words in $\mathbb{R}^d$ is much older than neural language models. You should think of this as a *generalization* of the discrete view of $\mathcal{V}$.

- Why?

Important Idea: Words as Vectors

The idea of "embedding" words in $\mathbb{R}^d$ is much older than neural language models. You should think of this as a *generalization* of the discrete view of $\mathcal{V}$.

- Why?
- Deerwester et al. (1990) explored dimensionality reduction techniques for information retrieval-style querying of text collections.

Important Idea: Words as Vectors

The idea of "embedding" words in $\mathbb{R}^d$ is much older than neural language models. You should think of this as a *generalization* of the discrete view of $\mathcal{V}$.

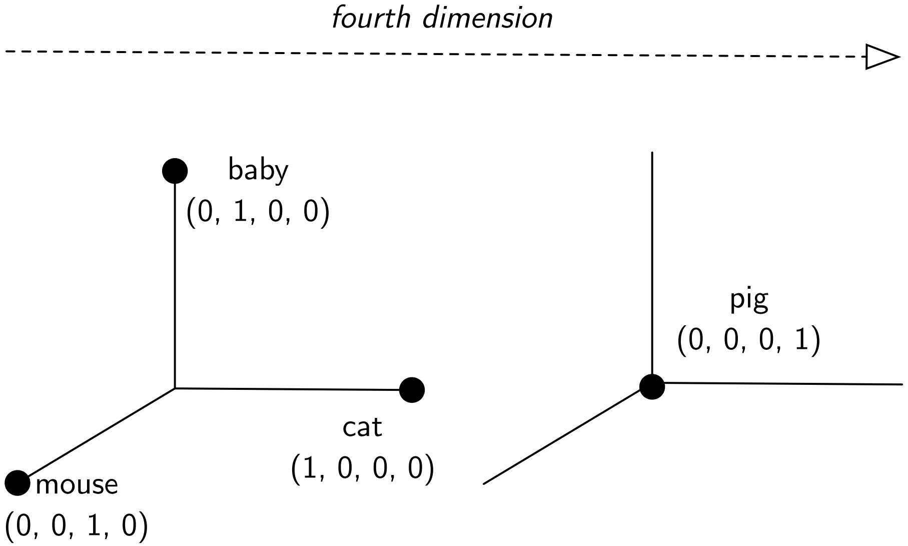
- Why?
- Deerwester et al. (1990) explored dimensionality reduction techniques for information retrieval-style querying of text collections.
- Considerable ongoing research on learning word representations to capture linguistic *similarity* (Turney and Pantel, 2010); this is known as **vector space semantics**.
    - Why "semantics"?

Important Idea: Words as Vectors

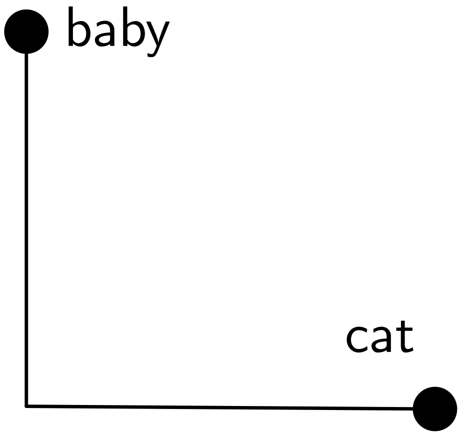
The idea of "embedding" words in $\mathbb{R}^d$ is much older than neural language models. You should think of this as a *generalization* of the discrete view of $\mathcal{V}$.

- Why?
- Deerwester et al. (1990) explored dimensionality reduction techniques for information retrieval-style querying of text collections.
- Considerable ongoing research on learning word representations to capture linguistic *similarity* (Turney and Pantel, 2010); this is known as **vector space semantics**.
  - Why "semantics"?
- Something like this also turns up in traditional linguistic theories, e.g., marking nouns as "animate" or not.

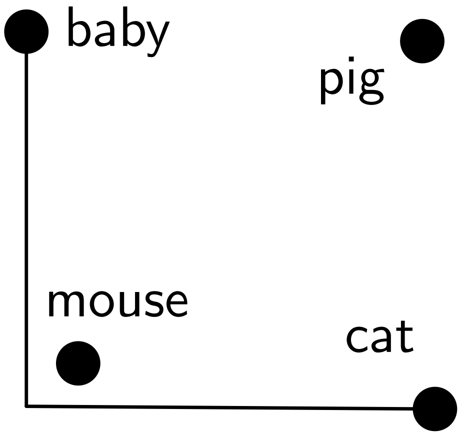
Words as Vectors: Example



Words as Vectors: Example



Words as Vectors: Example



Next Major Topic: Text Classification

Text Classification

Input: a piece of text $\boldsymbol{x} \in \mathcal{V}^{\dagger}$, usually a document (r.v. $\boldsymbol{X}$) Output: a label from a finite set $\mathcal{L}$ (r.v. L)

Standard line of attack:

1. Human experts label some data.
2. Feed the data to a supervised machine learning algorithm that constructs an automatic classifier $\text{classify} : \mathcal{V}^{\dagger} \rightarrow \mathcal{L}$
3. Apply classify to as much data as you want!

Note: we assume the texts are segmented already, even the new ones.

Text Classification: Examples

- Library-like subjects (e.g., the Dewey decimal system)
- News stories: politics vs. sports vs. business vs. technology ...
- Reviews of films, restaurants, products: postive vs. negative
- Author attributes: identity, political stance, gender, age, ...
- Email, arXiv submissions, etc.: spam vs. not
- What is the reading level of a piece of text?
- How influential will a scientific paper be?
- Will a piece of proposed legislation pass?

Closely related: relevance to a query.

Evaluation

Accuracy:

$$\begin{aligned}
\mathrm{A}(\text{classify}) &= p(\text{classify}(\boldsymbol{X}) = L) \\
&= \sum_{\boldsymbol{x} \in \mathcal{V}^\dagger, \ell \in \mathcal{L}} p(\boldsymbol{X} = \boldsymbol{x}, L = \ell) \cdot \begin{cases} 1 & \text{if } \text{classify}(\boldsymbol{x}) = \ell \\ 0 & \text{otherwise} \end{cases} \\
&= \sum_{\boldsymbol{x} \in \mathcal{V}^\dagger, \ell \in \mathcal{L}} p(\boldsymbol{X} = \boldsymbol{x}, L = \ell) \cdot \mathbf{1}\left\{\text{classify}(\boldsymbol{x}) = \ell\right\}
\end{aligned}$$

where p is the *true* distribution over data. Error is $1 - \mathrm{A}$.

This is *estimated* using a test dataset $\langle \bar{\boldsymbol{x}}_1, \bar{\ell}_1 \rangle, \ldots \langle \bar{\boldsymbol{x}}_m, \bar{\ell}_m \rangle$:

$$\hat{\mathrm{A}}(\text{classify}) = \frac{1}{m} \sum_{i=1}^{m} \mathbf{1}\left\{\text{classify}(\bar{\boldsymbol{x}}_i) = \bar{\ell}_i\right\}$$

Issues with Test-Set Accuracy

Issues with Test-Set Accuracy

- Class imbalance: if $p(L = \text{not spam}) = 0.99$, then you can get $\hat{\mathrm{A}} \approx 0.99$ by always guessing “not spam.”

Issues with Test-Set Accuracy

- Class imbalance: if $p(L = \text{not spam}) = 0.99$, then you can get $\hat{\text{A}} \approx 0.99$ by always guessing "not spam."
- Relative importance of classes or cost of error types.

Issues with Test-Set Accuracy

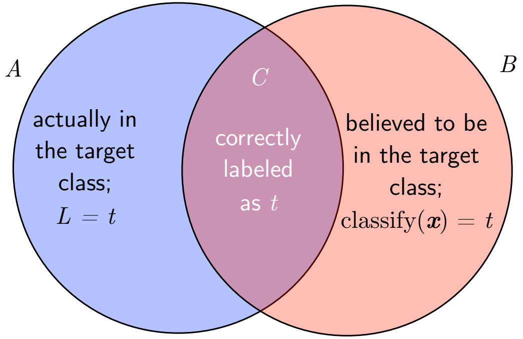
- Class imbalance: if $p(L = \text{not spam}) = 0.99$, then you can get $\hat{\mathrm{A}} \approx 0.99$ by always guessing "not spam."
- Relative importance of classes or cost of error types.
- Variance due to the test data.

Evaluation in the Two-Class Case

Suppose we have two classes, and one of them, $t \in \mathcal{L}$ is a "target."

- E.g., given a query, find relevant documents.

Precision and **recall** encode the goals of returning a "pure" set of targeted instances and capturing *all* of them.



A
actually in the target class; $L = t$
C
correctly labeled as t
B
believed to be in the target class; $\text{classify}(\boldsymbol{x}) = t$

$$\hat{\mathrm{P}}(\text{classify}) = \frac{|C|}{|B|} = \frac{|A \cap B|}{|B|}$$

$$\hat{\mathrm{R}}(\text{classify}) = \frac{|C|}{|A|} = \frac{|A \cap B|}{|A|}$$

$$\hat{F}_1(\text{classify}) = 2 \cdot \frac{\hat{\mathrm{P}} \cdot \hat{\mathrm{R}}}{\hat{\mathrm{P}} + \hat{\mathrm{R}}}$$

Another View: Contingency Table

	$L = t$	$L \neq t$	
$\text{classify}(\boldsymbol{X}) = t$	C (true positives)	$B \setminus C$ (false positives)	B
$\text{classify}(\boldsymbol{X}) \neq t$	$A \setminus C$ (false negatives)	(true negatives)	
	A		

Evaluation with > 2 Classes

Macroaveraged precision and recall: let each class be the target and report the average $\hat{\mathrm{P}}$ and $\hat{\mathrm{R}}$ across all classes.

Microaveraged precision and recall: pool all one-vs.-rest decisions into a single contingency table, calculate $\hat{\mathrm{P}}$ and $\hat{\mathrm{R}}$ from that.

Cross-Validation

Remember that $\hat{\mathrm{A}}$, $\hat{\mathrm{P}}$, $\hat{\mathrm{R}}$, and $\hat{F}_1$ are all *estimates* of the classifier's quality under the true data distribution.

- Estimates are noisy!

K-fold cross-validation:

- Partition the training set into K non-overlapping "folds" $\boldsymbol{x}^1, \ldots, \boldsymbol{x}^K$.
- For $i \in \{1, \ldots, K\}$:
  - Train on $\boldsymbol{x}_{1:n} \setminus \boldsymbol{x}^i$, using $\boldsymbol{x}^i$ as development data.
  - Estimate quality on the ith development set: $\hat{\mathrm{A}}^i$
- Report the average:

$$\hat{\mathrm{A}} = \frac{1}{K} \sum_{i=1}^{K} \hat{\mathrm{A}}^i$$

and perhaps also the standard error.

Statistical Significance

Suppose we have two classifiers, $\mathrm{classify}_1$ and $\mathrm{classify}_2$.

Statistical Significance

Suppose we have two classifiers, $\mathrm{classify}_1$ and $\mathrm{classify}_2$.

Is $\mathrm{classify}_1$ better? The "null hypothesis," denoted H_0, is that it isn't. But if $\hat{\mathrm{A}}_1 \gg \hat{\mathrm{A}}_2$, we are tempted to believe otherwise.

Statistical Significance

Suppose we have two classifiers, classify_1 and classify_2.

Is classify_1 better? The "null hypothesis," denoted H_0, is that it isn't. But if $\hat{\text{A}}_1 \gg \hat{\text{A}}_2$, we are tempted to believe otherwise.

How much larger must $\hat{\text{A}}_1$ be than $\hat{\text{A}}_2$ to *reject* H_0?

Statistical Significance

Suppose we have two classifiers, classify_1 and classify_2.

Is classify_1 better? The "null hypothesis," denoted H_0, is that it isn't. But if $\hat{\mathrm{A}}_1 \gg \hat{\mathrm{A}}_2$, we are tempted to believe otherwise.

How much larger must $\hat{\mathrm{A}}_1$ be than $\hat{\mathrm{A}}_2$ to *reject* H_0?

Frequentist view: how (im)probable is the observed difference, given $H_0 = \text{true}$?

Statistical Significance

Suppose we have two classifiers, $\mathrm{classify}_1$ and $\mathrm{classify}_2$.

Is $\mathrm{classify}_1$ better? The "null hypothesis," denoted H_0, is that it isn't. But if $\hat{\mathrm{A}}_1 \gg \hat{\mathrm{A}}_2$, we are tempted to believe otherwise.

How much larger must $\hat{\mathrm{A}}_1$ be than $\hat{\mathrm{A}}_2$ to *reject* H_0?

Frequentist view: how (im)probable is the observed difference, given $H_0 =$ true?

Caution: statistical significance is neither necessary nor sufficient for research significance or practical usefulness!

A Hypothesis Test for Text Classifiers

McNemar (1947)

1. The null hypothesis: $\mathrm{A}_1 = \mathrm{A}_2$
2. Pick significance level α, an "acceptably" high probability of incorrectly rejecting H_0.
3. Calculate the test statistic, k (explained in the next slide).
4. Calculate the probability of a *more extreme* value of k, assuming H_0 is true; this is the p-value.
5. Reject the null hypothesis if the p-value is less than α.

The p-value is $p(\text{this observation} \mid H_0 \text{ is true})$, not the other way around!

McNemar's Test: Details

Assumptions: independent (test) samples and binary measurements. Count test set error patterns:

	classify_1 is incorrect	classify_1 is correct	
classify_2 is incorrect	c_{00}	c_{10}	
classify_2 is correct	c_{01}	c_{11}	$m \cdot \hat{\mathrm{A}}_2$
		$m \cdot \hat{\mathrm{A}}_1$	

If $\mathrm{A}_1 = \mathrm{A}_2$, then c_{01} and c_{10} are each distributed according to $\mathrm{Binomial}(c_{01} + c_{10}, \frac{1}{2})$.

$$\text{test statistic } k = \min\{c_{01}, c_{10}\}$$

$$p\text{-value} = \frac{1}{2^{c_{01}+c_{10}-1}} \sum_{j=0}^{k} \binom{c_{01}+c_{10}}{j}$$

Other Tests

Different tests make different assumptions.

Sometimes we calculate an interval that would be “unsurprising” under H_0 and test whether a test statistic falls in that interval (e.g., t-test and Wald test).

In many cases, there is no closed form for estimating p-values, so we use random approximations (e.g., permutation test and paired bootstrap test).

If you do lots of tests, you need to correct for that!

Read lots more in Smith (2011), appendix B.

References I

Scott C. Deerwester, Susan T. Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6): 391–407, 1990.

Quinn McNemar. Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika*, 12(2):153–157, 1947.

Noah A. Smith. *Linguistic Structure Prediction*. Synthesis Lectures on Human Language Technologies. Morgan and Claypool, 2011. URL `http://www.morganclaypool.com/doi/pdf/10.2200/S00361ED1V01Y201105HLT013.pdf`.

Peter D. Turney and Patrick Pantel. From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research*, 37(1):141–188, 2010. URL `https://www.jair.org/media/2934/live-2934-4846-jair.pdf`.