096222: Language, Computation and Cognition
Homework Assignment 3
due 16 May 2023

2 May 2023

The Colab notebook for this homework is available here.

1 Logistic Regression (30 points)

In this problem we look at the English dative alternation. You will work with a dataset of annotated examples of the dative alternation from spontaneous speech in conversation, reported in Bresnan et al. (2007) and made available through the `languageR` package (Baayen, 2007). The Colab notebook for this problem automatically downloads this dataset as `dative.csv`.

The dative alternation involves ditransitive verbs,

(1) a. Kim mailed $\overbrace{\text{me}}^{\text{Recipient}}$ $\overbrace{\text{a gift}}^{\text{Theme}}$. [D(ouble) O(bject)]
    b. Kim mailed $\underbrace{\text{a gift}}_{\text{Theme}}$ to $\underbrace{\text{me}}_{\text{Recipient}}$. [P(repositional) D(ative)]

and the theme and recipient arguments are, respectively, what gets acted upon (usually transferred in physical location or possession), and the recipient or destination of the action. One qualitative intuition often reported about the dative alternation is that cases where the recipient argument is a large phrase (as measured, e.g., in number of words) are awkward in the Double Object construction, such as the below example:

(2) ?Kim mailed everyone who had attended the party yesterday a gift.

Alternatively, some researchers have proposed that what is more crucial is whether the recipient and theme arguments are pronouns. In this problem, we test these ideas by building simple logistic regression models of the dative alternation to look at the predictive effects of the length and pronominality of the recipient and theme arguments. We will use the `pandas` and `statmodels` Python packages for this.

In general, in studying the factors influencing speaker preference in the dative alternation, we will be interested in estimating the following probabilistic model:

$$P(\text{Construction} = \text{Double Object}|\text{Subject}, \text{Verb}, \text{Recipient}, \text{Theme})$$

where any of a number of features of the subject, verb, recipient, and theme might influence the speaker or writer's choice of linguistic construction. For purposes of this problem, however, we will dramatically simplify. First, we will ignore the subject and verb altogether, simplifying our problem to:[1]

$$P(\text{Construction} = \text{Double Object}|\text{Recipient}, \text{Theme})$$

Also, we'll only use a few features of the recipient and theme in our predictive model. Recall that logistic regression is characterized by the following equations:

$$\eta = \sum_i \beta_i X_i \qquad \text{(linear predictor)}$$

$$P(\text{success}) = \frac{e^\eta}{1 + e^\eta} \qquad \text{(logistic transform of linear predictor)}$$

For the dative alternation, the two possible outcomes are the double object construction (**DO**) and prepositional dative construction (**PD**). We arbitrarily choose DO as the outcome corresponding to "success".

Unlike the case of binomial ordering choice, there is a systematic difference between the possible outcomes, that holds across all instances of the dative alternation: it is always the same two constructions that are being chosen between. To capture the possibility of an overall preference for one construction or the other, we add what is called an "intercept" or "bias" term to the equation determining the linear predictor. This is often expressed in the statistics literature as (assuming M predictors):

$$\eta = \alpha + \sum_{i=1}^{M} \beta_i X_i \qquad \text{(linear predictor)},$$

but it can equivalently be expressed by defining a "dummy" predictor X_0 whose value is always 1, and writing the linear predictor as:

$$\eta = \sum_{i=0}^{M} \beta_i X_i \qquad \text{(linear predictor)},$$

[1] This is an oversimplification, actually: in particular, the identity of the verb has a *lot* of predictive information. This information is most effectively brought into our model by introducing a hierarchical component, endowing verbs with idiosyncratic preferences for which construction they prefer and by how much. Bresnan et al. (2007) and Morgan and Levy (2015) are good references for seeing how to include such a hierarchical (sometimes called "mixed-effects") component for the dative alternation and for binomials, respectively.

a formulation more common in the machine learning literature. (Note that it would be inappropriate to include an intercept in the model for binomial ordering preferences, because there is no intrinsic difference between “success” and “failure” outcomes that is consistently defined across different specific cases of binomial ordering choice.)

We can evaluate the quality of a logistic regression model in a couple of ways. One is predicting the CLASS of the outcome: here, DO or PD? We say that a logistic regression model predicts “success” for a datum if it assigns $P(\text{success}) > 0.5$ for that datum, otherwise “failure”. A second is the LOG-LIKELIHOOD of the dataset—the summed log-probabilities of all the observations in the dataset under the fitted model.

Tasks:

1. Define and implement an 80/20 train/test random split of the `dative` dataset.
2. Fit a logistic regression model to the training set that uses *only* recipient pronominality and an intercept term. What is its classification accuracy on the held-out test dataset? How about its log-likelihood?
3. Add theme pronominality as a predictor to the model and see whether that improves the model’s predictive power as assessed by held-out classification accuracy and log-likelihood.
4. Determine whether additionally adding theme and recipient length (in number of words) to the model further improves fit. Try both raw length or log-transformed length. Which gives better performance?
5. Look at the β coefficients of a fitted model with all four predictors and interpret them theoretically (don’t worry about interpreting the intercept α, whose value will depend on the numeric coding scheme used for the predictors). Are there any general linguistic principles manifested in the values of all four predictors? Do you see any ways to simplify the model (reduce the number of predictor weights that have to be learned) based on general linguistic principles, without sacrificing much predictive accuracy? This may involve creating a new set of predictors that are a function of the four predictors you’ve been working with up until now.

2 Distance, similarity, and analogies in word embeddings (30 points)

This problem set involves manipulating and using **word embeddings**: representations of the semantics of words as high-dimensional vectors. We will be using off-the-shelf semantic vectors derived in previous work (Pennington et al., 2014), called GloVe vectors. You should use one of the data zip files from https://nlp.stanford.edu/projects/glove/. We recommend `glove.6B.zip`, but you may use any of the vector files provided on the website. Note that working with larger vector files will make processing times in your code slower. Unzipping the file `glove.6B.zip`, you will see a number of `.txt` files. For the exercises below, we recommend using the 300-dimensional vectors in the file `glove.6B.300d.txt`. (Note that all words the GloVe word vector files are converted to lower-case, so you will have to do the same in this exercise.)

The Colab notebook contains Python code for downloading the GloVe vectors and reading them into a dictionary data structure. We call the resulting dictionary `e` (for embedding), so calling `e['car']` returns an array representing the semantics of the word *car*, and so on.

1. One of the main functions of semantic vectors is to represent similarity relations among words. For example, *frog* and *toad* are very similar in meaning, while *frog* and *yesterday* are very dissimilar.

   Write a function to compute the **cosine similarity** between two vectors. Cosine similarity is a score between -1 and 1 indicating similarity, where 1 is maximal similarity and -1 is minimal similarity. Cosine similarity between two vectors $\mathbf{A}$ and $\mathbf{B}$ is defined as:

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A}\cdot\mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum\limits_{i=1}^{n} A_i B_i}{\sqrt{\sum\limits_{i=1}^{n} A_i^2}\sqrt{\sum\limits_{i=1}^{n} B_i^2}}.$$

   Hint: You will probably find it faster and more convenient to use the function `numpy.dot` from the `numpy` package rather than manually implementing all the summations above!

   (a) Verify that your implementation of cosine similarity is correct by checking that it is **symmetrical**: it should be the case that `similarity(x,y) == similarity(y,x)` for all `x` and `y`. Demonstrate that this is the case with a few examples.

   (b) As sanity checks, verify that the following similarity relations are true in the GloVe vectors given your implementation of cosine similarity. Report the similarity relations for these examples.

      i. *car* is closer to *truck* than to *person*

      ii. *Mars* is closer to *Venus* than to *goes*

      iii. *warm* is closer to *cool* than to *yesterday*

      iv. *red* is closer to *blue* than to *fast*

      v. Come up with two more examples that demonstrate correct similarity relations.

      vi. Come up with two examples where cosine similarity in the semantic vectors does not align with your intuitions about word similarity.

   (c) For the examples where cosine similarity does not match your intuitions, what do you think went wrong?

   (d) **Bonus 3 points**: Try a different distance metric, such as Euclidean distance. Does it result in qualitatively different patterns on your test suite?

2. Write a function to perform the **analogy task**: Given words w_1, w_2, and w_3, find a word x such that $w_1 : w_2 :: w_3 : x$. For example, for the analogy problem *France*:*Paris* :: *England*:x, the answer should be *London*. To solve analogies using semantic vectors, letting $e(w)$ indicate the embedding for a word w, calculate a vector $y = e(w_2) - e(w_1) + e(w_3)$ and find the word whose vector is closest to y.

   (a) Explain why the analogy-solving method described above makes sense.

   (b) Write a function to calculate y. The output should be a semantic vector.

   (c) Write a function to find the nearest words to y in terms of cosine similarity, and output the top 5.

   (d) Report the top 5 results for the following analogies, and describe whether you think they are sensible and why:

      i. France : Paris :: England : x

      ii. man : woman :: king : x

      iii. tall : taller :: warm : x

      iv. tall : short :: warm : x

      v. Come up with 4 more analogies, 2 of which work in your opinion, and 2 of which do not work.

   (e) Did you notice any patterns or generalizations while exploring possible analogies? For the ones that went wrong, why do you think they went wrong?

3 Using semantic vectors to decode brain activation (40 points)

Pereira et al. (2018) **decode** the imaging data to determine which words and pictures the subjects were looking at, with above-chance accuracy. In this section you will be replicating a simple version of their model, given their data.

Subjects were exposed to 180 ”concepts” (words presented along with pictures). For each concept, levels of brain activation (BOLD response) were recorded at $\sim 150,000$ 3D coordinates (called **voxels**) inside the brain. We are providing a dataset with the imaging data for one subject from this experiment. We will refer to the number of voxels in the dataset as V.

In the file `learn_decoder.py`, we provide functions which read the provided semantic vector and imaging data files, and a function `learn_decoder` which takes a data matrix (a $180 \times V$ matrix of V voxel activations for each of the 180 concepts) and a matrix of semantic vectors (a 180x300 matrix mapping each concept to 300 semantic dimensions) and outputs a matrix M, which is a **decoder** that can decode imaging data into a 300 -dimensional semantic vector. More precisely, **the dot product of a data vector (a V-dimensional vector representing the imaging data for a concept) and M outputs a 300-dimensional**

semantic vector, which is the model's best guess as to the concept that the subject was being exposed to.

The decoding process is evaluated in the following way. A decoder is trained on imaging data from 170 concepts (the training set), and then used to decode the imaging data from a held-out set of 10 concepts (the test set). The decoded vectors are evaluated according to the "average rank" metric: - Average rank: Given a decoded vector $\hat{v}$ for a concept whose true semantic vector is v, rank all the semantic vectors in order of their closeness (cosine similarity in our case) to $\hat{v}$. Now calculate the position of v in this ranking: for example, if the vector for v is the 10th closest vector to $\hat{v}$, then the rank is 10. Then we can get the average ranking for all the decoded vectors. The average ranking is an accuracy score where the optimal score would be 1 and the worst possible score would be 180. If the decoder is outputting random noise, then the resulting average rank should be 90.

The authors perform this process for multiple training and test sets using **k-fold cross-validation**. k-fold cross-validation splits a dataset into k different training and test sets. Suppose $k = 18$. Then the first training set-test set pair would take the first 10 concepts as the test set, and use the rest as the training set. The second training set would take the second 10 concepts (concepts 11-20) as the test set, and use the rest as the training set, and so on. Thus the procedure produces 18 different accuracy scores, one for each fold of the cross-validation.

Your job is to write code to split the data into training and test sets according to 18-fold cross-validation, train the decoder using the provided code, use the decoder to decode semantic vectors, and evaluate the accuracy of these decoded vectors using the average rank method for each fold.

Your writeup should include answers to the following questions:

1. What are the accuracy scores? Show a plot of accuracy scores for each of the 18 folds.

2. Which concepts can be decoded with more or less success?

3. Are the results satisfactory, in your opinion? Why or why not?

References

Baayen, R. H. (2007). The languager package. *Available on-line at URL: http://cran. r-project. org/doc/packages/languageR. pdf.*

Bresnan, J., Cueni, A., Nikitina, T., & Baayen, R. H. (2007). Predicting the dative alternation. In *Cognitive foundations of interpretation* (pp. 69–94). KNAW.

Morgan, E., & Levy, R. (2015). Modeling idiosyncratic preferences: How generative knowledge and expression frequency jointly determine language structure. *CogSci*.

Pennington, J., Socher, R., & Manning, C. D. (2014). Glove: Global vectors for word representation. *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 1532–1543.

Pereira, F., Lou, B., Pritchett, B., Ritter, S., Gershman, S. J., Kanwisher, N., Botvinick, M., & Fedorenko, E. (2018). Toward a universal decoder of linguistic meaning from brain activation. *Nature communications*, *9*(1), 1–13.