Using spaCy and Keras to build a neural network classifier

Intro to NLP

Early draft tutorial

Tong Liu, with Cecilia O. Alm

For more, see Ch. 8 in J&M 3rd edition

Using word vectors in NLP with deep learning tasks

- Learn multiple layers of feature representations mapping to output
  - Raw inputs such as words, characters, embeddings (e.g., of words)

- Why the visibility of deep learning?
  - Has outperformed other ML techniques in benchmarks
    - Computer vision
    - Speech recognition

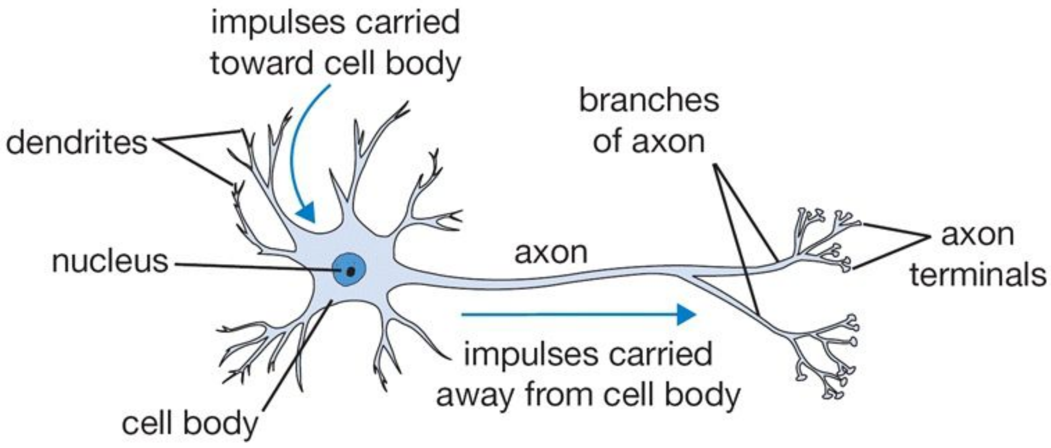
- Deep Neural Networks (DNNs) are facilitated by
  - Massive data availability (texts, images, videos, etc.)
  - Hardware acceleration (CPU, GPU)
  - New algorithms, heuristics improving on fundamental NN concepts

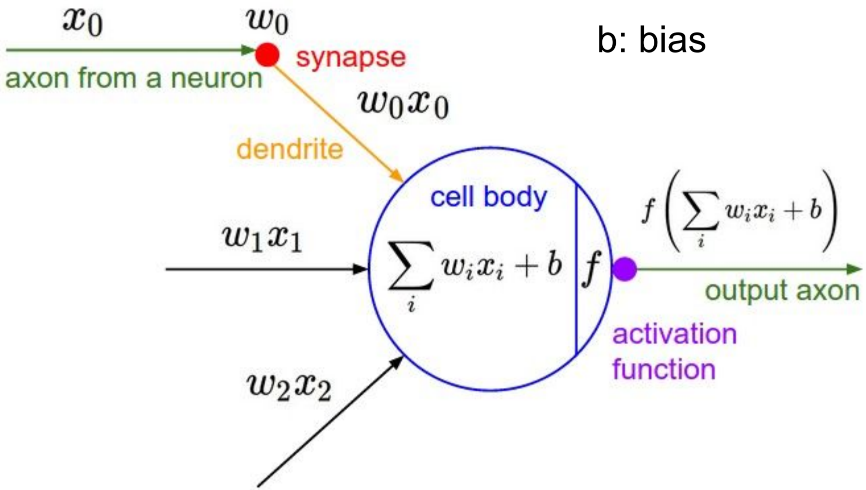
Deep learning for NLP

- Use representation learning and deep learning (DL) to solve NLP problems
- Representation learning: techniques to automatically discover the representations from raw data.
  - Such as word2vec, GloVe
- NLP has achieved improvements with DL in recent years:
  - Problems in morphology [1], syntax [2], semantics [3]
  - Applications like machine translation [4], question answering [5], sentiment analysis [6]

1. Luong, Thang, Richard Socher, and Christopher D. Manning. "Better word representations with recursive neural networks for morphology." CoNLL. 2013.
2. Socher, Richard, et al. "Parsing natural scenes and natural language with recursive neural networks." *Proceedings of the 28th international conference on machine learning (ICML-11)*. 2011.
3. Bowman, Samuel R., Christopher Potts, and Christopher D. Manning. "Recursive neural networks can learn logical semantics." arXiv preprint arXiv:1406.1827 (2014).
4. Mikolov, Tomas, Quoc V. Le, and Ilya Sutskever. "Exploiting similarities among languages for machine translation." arXiv preprint arXiv:1309.4168 (2013).
5. Iyyer, Mohit, et al. "A Neural Network for Factoid Question Answering over Paragraphs." EMNLP. 2014.
6. Socher, Richard, et al. "Recursive deep models for semantic compositionality over a sentiment treebank." Proceedings of the conference on empirical methods in natural language processing (EMNLP). Vol. 1631. 2013.

Inspirations for neural network models from biology





b: bias

A coarse biological brain neuron system (left) and its approximate mathematical computation form (right)

Recommended readings from Neuroscience:
https://neurophysics.ucsd.edu/courses/physics_171/annurev.neuro.28.061604.135703.pdf
http://www.sciencedirect.com/science/article/pii/S0959438814000130

Figures from https://cs231n.github.io/neural-networks-1/

Feed-Forward Neural Networks

Basic features:

- A multilayer network without cycles between connected neural units
  - In contrast, recurrent NN have cycles
- Outputs from units in one layer are passed to units in the next layer
- Nothing flows back to the previous layers

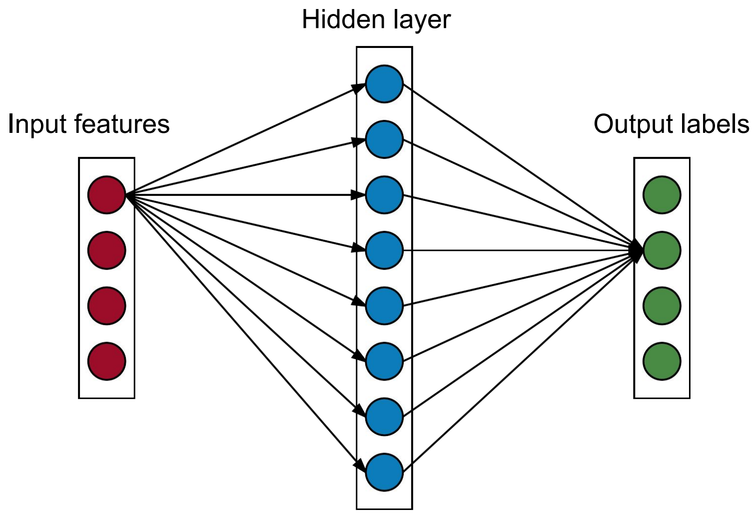


Figure inspired by and modified from http://www.kdnuggets.com/2016/01/seven-steps-deep-learning.html

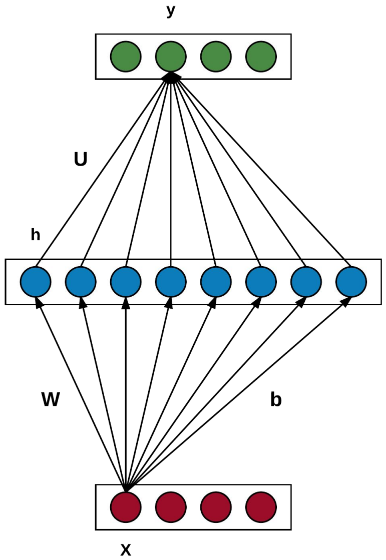
Multilayer perceptrons (MLPs)

- A feedforward fully-connected NN
- Consists of multiple directed layers of units
  - Layers connected with certain weights (and bias)
- Except for the input nodes, each unit represents a nonlinear activation function

$$h = f(Wx + b)$$

$$z = Uh$$

$$y = softmax(z)$$



x	Input units
h	Hidden units
y	Output units (probability distribution)
f	Non-linear function
W, U	Weight matrices
b	Bias vector
softmax()	A vector normalization function

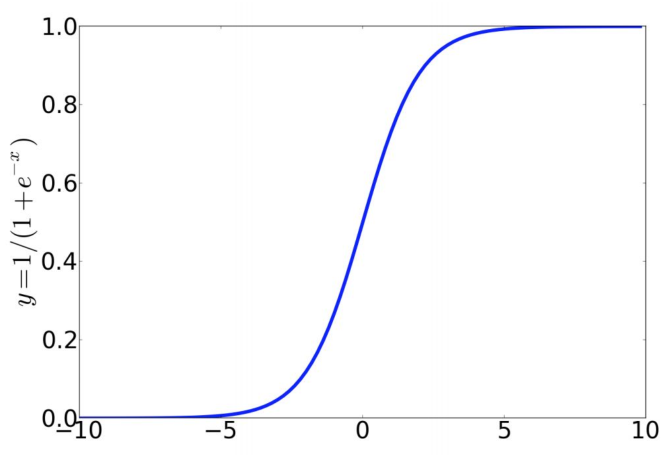
Figure inspired by and modified from kdnuggets.com/2016/01/seven-steps-deep-learning.html

Non-linear functions

sigmoid

Computation: $\sigma(z) = \frac{1}{1+e^{-z}}$

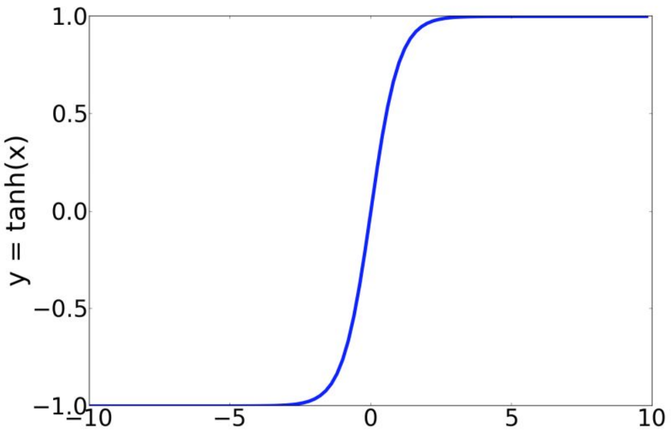
[0, 1]



tanh

Computation: $y = \frac{e^z - e^{-z}}{e^z + e^{-z}}$

[-1, +1]

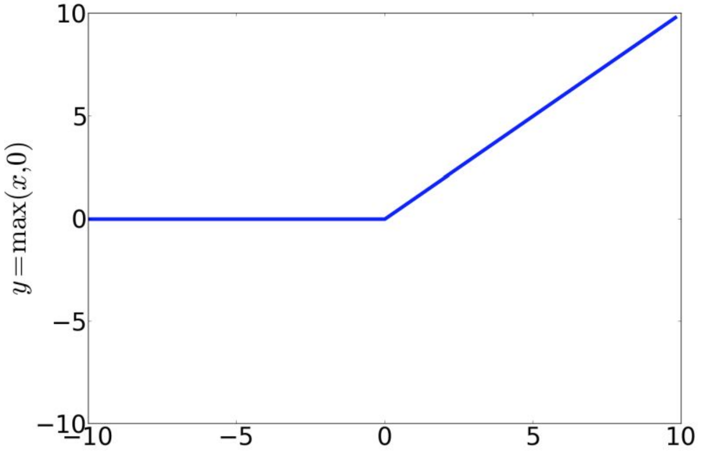


Figures from Ch. 8 in J&M 3rd edition

Non-linear functions

ReLU (Rectified Linear Unit)

Computation: $y = max(0, x)$



Softmax

Computation: $softmax(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}}$

$1 \leq i \leq D$

1. A D-dimensional vector z of real values
2. *Interesting property:* The calculated probabilities are in the range of [0,1].
3. All probabilities are summed up to 1.

Figures from Ch. 8 in J&M 3rd edition

Summary overview of deep learning alternatives

Architectures	Conventional (Sequence or Non-Recursive)	Recursive	Combination of Conventional and Recursive
Basic types	Recurrent Neural Network (RNN) Long Short-Term Memory (LSTM) Convolutional Neural Network (CNN)	Recursive Autoencoders (RAE) Recursive NN (RsNN)	Tree-LSTM
Variants	Bi-RNN, Bi-LSTM, CNN-multichannel, CNN-non-static, DCNN	MV-RsNN, RsNTN	DTree-LSTM, CTree-LSTM, Deep RsNN

from Rojas-Barahona (2016, p. 708)

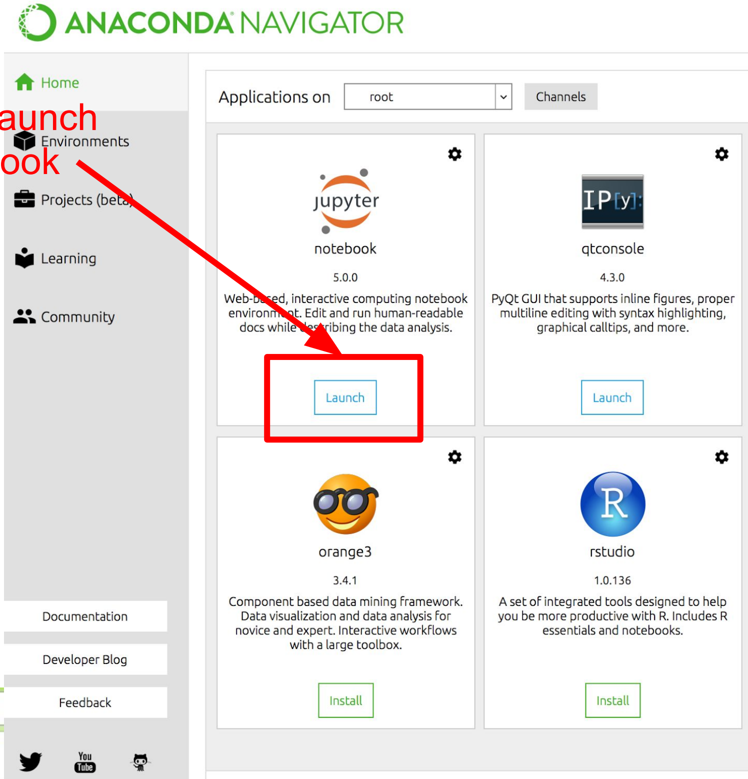
Activity: Building a DL classifier

- **Keras** (https://keras.io/): A high-level NN library for DL in Python
- Lets you use TensorFlow, Theano and CNTK low-level libraries
  - TensorFlow is the proposed default installation (compatible with the other backends depending on your application)
- Lets you use out-of-the-box implementations of common NN architectures
  - Convolutional neural networks (CNN)
  - Recurrent neural networks (RNN)
  - Their combinations

Let's get started!

1. Download and expand the folder NeuralNetworkLab.zip
2. Launch “Jupyter Notebook” in Anaconda. It will start the notebook server and open a page in the web browser
3. In the Files tab, navigate to the folder and open “polarity_mlp_glove.ipynb” by clicking on it.

2. Click here to launch Jupyter Notebook



3. Navigate to your folder and open the notebook

jupyter polarity_mlp_glove Last Checkpoint: an hour ago (autosaved)

Logout

File Edit View Insert Cell Kernel Help

Trusted | Python 3

Run Code

Click **Run** to execute one code block each time

In [1]:
```python
import os
# http://www.cs.cornell.edu/people/pabo/movie-review-data/
# polarity dataset v2.0
# 1000 positive and 1000 negative movie reviews
TEXT_DATA_DIR = 'review_polarity/txt_sentoken/'
```

In [2]:
```python
def prepare_texts_labels(TEXT_DATA_DIR):
    texts = []
    labels = []
    # iterate through directories to read each file
    for name in sorted(os.listdir(TEXT_DATA_DIR)):
        if not name.startswith('.'):
            path = os.path.join(TEXT_DATA_DIR, name)
            if os.path.isdir(path):
                for fname in sorted(os.listdir(path)):
                    fpath = os.path.join(path, fname)
                    with open(fpath, 'r') as myfile:
                        doc = myfile.read().replace('\n', '')
                        # store each review and label respectively
                        texts.append(doc)
                        if name == 'pos':
                            labels.append(+1)
                        else:
                            labels.append(0)
    return texts, labels
```

Remember: multiple prior code blocks need to be executed again if there is a parameter/function change above

variables from prior code blocks remain accessible in the following blocks

In [3]:
```python
# Check the numbers of texts and labels
texts, labels = prepare_texts_labels(TEXT_DATA_DIR)
print(len(texts), len(labels))
```

```
2000 2000
```

In [4]:
```python
# Check the average/maximum/minimum length of reviews in characters
total_avg = sum( map(len, texts) ) / len(texts)
print(total_avg)
print(len(max(texts, key=len)))
print(len(min(texts, key=len)))
```

Task - Classify movie reviews into positive/negative categories

This is an overview of the process you will complete over the rest of the tutorial:

1. Convert the dataset into a list of review instances and a parallel list of their polarity labels
   - polarity dataset v2.0 with 1000 positive and 1000 negative movie reviews (http://www.cs.cornell.edu/people/pabo/movie-review-data/)
2. Split randomized data into 90% training and 10% validation parts
3. Load embedding vectors as input features - using pre-trained GloVe vectors from spaCy
4. Load this embedding matrix into a Keras sequential model
5. Train a binary classifier using a 2-layer NN (extra doc: https://keras.io/getting-started/sequential-model-guide/#training)
6. Evaluate it against the held-out (validation) data
7. Run 10-fold cross validation if time allows

Task adapted from https://blog.keras.io/using-pre-trained-word-embeddings-in-a-keras-model.html

Preparing the training/test data

[1] Set the text data directory variable

[2] Iterate over the folders to read each file, store the text sample and the class label in two lists, respectively

[3] Check the numbers of texts and labels

[4] Check the average/maximum/minimum length of reviews in characters

[5, 6] Load pre-trained vectors from spaCy

[7] Check the average/maximum/minimum length of reviews in tokens

[8] Split data (and labels) into training and testing subsets
- 90% training and 10% testing
- Check the number of pos/neg samples in each subset (should be equal)

Features and labels

- X

  [9] Set a maximum sequence length to use in features
  - Use the average number of tokens

  [10] Define a function to convert words in text into a matrix of vectors as features
  - Use a simple concatenate representation in this exercise

  [11, 12] Extract features for training and testing data (slow -- just wait)

- Y

  [13] Converts class labels to numpy arrays

Model training

[14, 15] Create a *Sequential* model with a linear stack of layers

[16] Add a hidden layer

[17] Add an output layer

[18] Configure the learning process before model training

- Optimizer, loss function, classification metrics

[19] Train the model, iterating on the data in batches of 128 samples, 5 epochs

- Epoch: number of forward and backward passes of *all* training samples
- Batch size: number of samples in one forward and backward pass

Model evaluation

[20] Print a summary of your model

[21] Evaluate the model against the 10% validation data in terms of the loss value & metrics values

- Results may vary, because data are split randomly in cell [8]

[22*] Instantiate a Stratified 10-Folds cross-validator

* if time allows

[23*] Repeat the above process [7-18] for 10 times to get average performance metrics

[24] Delete session from keras backend to free sources

References and further readings

1. https://elitedatascience.com/keras-tutorial-deep-learning-in-python
2. https://blog.keras.io/category/tutorials.html
3. http://machinelearningmastery.com/tutorial-first-neural-network-python-keras/
4. Stanford CS224d: Deep Learning for Natural Language Processing
   http://cs224d.stanford.edu/index.html
5. Stanford CS231n Convolutional Neural Networks for Visual Recognition
   https://cs231n.github.io/convolutional-networks/
6. https://keras.io/