Deep Learning Introduction and Natural Language Processing Applications

GMU CSI 899

Jim Simpson, PhD
Jim.Simpson@Cynnovative.com

9/18/2017
Agenda

• Fundamentals
  • Linear and Logistic Regression
  • Logistic Regression to Neural Networks
  • Neural Networks to Deep Learning
  • Representation Learning with Deep Neural Networks

• Natural Language Processing Applications
  • Word Embeddings/Vectors
    • Word2Vec
  • Language Models
    • Long-Short-Term-Memory Recurrent Neural Networks

• Additional Reading
Deep Learning Models

- Are Neural Networks with more than one hidden layer

Neural Networks

- Are two-dimensional arrays of Logistic Regressors loosely inspired by how neurons are connected in the mammalian brain

Deep Learning vs Traditional Machine Learning

- Deep Learning can learn complex non-linear relationships in the data
- Can do this without explicit manual feature engineering
- Adapts to all types of data (even unstructured – images and natural language)
Regression Analysis Overview

- **Linear Regression**
  - Dependent Variable (Predictions): Continuous
  - Simple Case: Equation of a line
    \[ y = \beta_0 + \beta_1 x \]
  - Multiple Linear Regression:
    \[ Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 \]

- **Logistic Regression**
  - Dependent Variable (Predictions): Categorical
  - Simple Case: Sigmoid function
    \[ y = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x)}} \]
  - Multiple Logistic Regression:
    \[ \text{logit}(Y) = \ln \left( \frac{Y}{1 - Y} \right) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 \]
Two input dimensions are combined linearly to form single dimension output.

\[ Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 \]
Logistic Regression to Neural Networks

- Add extra steps between input and output

- With multiple dimensions
Neural Networks with Hidden Units

- Add non-linearity through layering activation functions

![Diagram of neural network with hidden units](image)

- Advantages
  - Adding these Hidden Units allows us to capture complex interactions between the variables, whereas we previously treated them as linearly independent
  - The non-linearity on the Hidden Units results in a warping of the feature space that is hard to visualize but really beneficial
  - Being able to choose the number of Hidden Units allows us to change the dimensionality of the problem, potentially making classification far easier in a higher-dimensional space

\[ f(x) = \tanh(x) \]
Deep Learning uses Neural Networks with multiple hidden layers.

- Number of neurons per layer and number of layers become hyper-parameters.

Input Dimension e.g. Number of pixels in image

Output Classes e.g. Numbers 0-9 in Digits Recognition

Input Dimension

Output Classes
Logistic Regression without Feature Engineering

Logistic Regression without manual feature engineering is NOT able to separate blue dots from orange dots

http://playground.tensorflow.org/#activation=relu&regularization=L2&batchSize=20&dataset=circle&regDataset=reg-plane&learningRate=0.1&regularizationRate=0.001&noise=0&networkShape=&seed=0.27923&showTestData=false&discretize=false&percTrainData=80&x=true&y=true&xTimesY=false&xSquared=false&ySquared=false&cosX=false&sinX=false&cosY=false&sinY=false&collectStats=false&problem=classification&initZero=false&hideText=false
Learning Non-linear Decision Boundaries

Logistic Regression with Manual Feature Engineering

Adding additional hand derived features allows logistic regression to separate blue dots from orange dots

http://playground.tensorflow.org/#activation=relu&regularization=L2&batchSize=20&dataset=circle&regDataset=reg-plane&learningRate=0.1&regularizationRate=0.001&noise=0&networkShape=&seed=0.27923&showTestData=false&discretize=false&percTrainData=80&x=true&y=true&xTimesY=false&xSquared=true&ySquared=true&cosX=false&sinX=false&cosY=false&sinY=false&collectStats=false&problem=classification&initZero=false&hideText=false
Neural Network without Manual Feature Engineering

A very simple neural network can separate the two without any manual feature engineering.
Learning Non-linear Decision Boundaries

Deep Neural Network

http://playground.tensorflow.org/#activation=relu&regularization=L2&batchSize=20&dataset=spiral&regDataset=reg-plane&learnRate=0.03&regularizationRate=0.001&noise=0&networkShape=8,8,6&seed=0.99514&showTestData=false&discretize=false&percTrainData=80&x=true&y=true&xTimesY=false&xSquared=false&ySquared=false&cosX=false&sinX=false&cosY=false&sinY=false&collectStats=false&problem=classification&initZero=false&hideText=false
Natural Language Processing (NLP) Tasks and Recurrent Neural Networks

- NLP Applications
  - Sentiment Analysis
  - Machine Translation
  - Question Answering
  - Dialogue Agents
  - Language Generation

- Common Across all Applications
  - Recurrent Neural Networks (RNNs)
  - Word Embeddings/Vectors

Recommended Resource: Stanford CS224d/n: Natural Language Processing with Deep Learning:
http://web.stanford.edu/class/cs224n/
Problem: consider the sentence

“I made her duck”

Approach: Distributional Hypothesis

“You shall know a word by the company it keeps” – J. R. Firth

Solution: Word Embeddings/Vectors

https://www.tensorflow.org/tutorials/word2vec
Given a corpus with these three sentences

- I like deep learning.
- I like NLP.
- I enjoy flying.

**Co-Occurrence Matrix**

<table>
<thead>
<tr>
<th>counts</th>
<th>I</th>
<th>like</th>
<th>enjoy</th>
<th>deep</th>
<th>learning</th>
<th>NLP</th>
<th>flying</th>
<th>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>like</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>enjoy</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>deep</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>learning</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>NLP</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>flying</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>.</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

**Singular Value Decomposition**

- Problems:
  - Computation scales quadratically for n x m matrix: O(mn²)
  - Hard to add new words or documents
• Instead of capturing co-occurrence counts directly
• Predict surrounding words of every word
• In a window of length c of every word
• Objective function: Maximize the log probability of any context word given current center word:

\[ J(\theta) = \frac{1}{T} \sum_{t=1}^{T} \sum_{-c \leq j \leq c, j \neq 0} \log p(w_{t+j} | w_t) \]

• Simplest first formulation for conditional probability:

\[ p(w_O | w_I) = \frac{\exp \left( v'_w^T v_{w_I} \right)}{\sum_{w=1}^{W} \exp \left( v'_w^T v_{w_I} \right)} \]
Word2Vec: Skip-Gram with Negative Sampling

- Word2Vec embeds each word into a low-dimensional vector space using:
  - Skip-Gram: Train for center word $w_I$ at time $t$ in a local context window of length $c$
  - Negative Sampling: Clever way to frame the problem as a supervised classification problem

Maximize probability of: **a true pair**
   (the center word and word in its context window)

$$J(\theta) = \log \sigma(v'_wO^T v_{w_I}) + \sum_{i=1}^{k} \mathbb{E}_{w_i \sim P_n(w)} \left[ \log \sigma(-v'_w_i^T v_{w_I}) \right]$$

This simple logistic regression problem moves the vectors for the true pair closer

Minimize probability of: **a couple of random pairs**
   (the center word and a random word outside context window)

This simple logistic regression problem moves the vectors for the random pairs apart
Reduced Dimensional (300-dim to 2-d) Word Vectors Trained on English Wikipedia

Relationships

- niece
- aunt
- sister
- nephew
- uncle
- woman
- brother
- man

Superlatives

- slowest
- slower
- shorter
- short

Named Entities

- Chrysler
- United
- Exxon
- Wal-Mart
- IBM
- Citigroup
- Marchionne
- Smisek
- Tillerson
- McMillon
- Corbat
- Rometty
Language Model using Word Vectors

- A language model:
  - Assigns probabilities to sentences (sequence of $m$ words)
  - By predicting next word $w_i$, in a sentence given history of $i - 1$ previous words

$$P(w_1, \ldots, w_m) = \prod_{i=1}^{m} P(w_i \mid w_1, \ldots, w_{i-1})$$

- It is a classification problem where the target class at each iteration is $w_i$
- The model is trained to predict a probability distribution over the vocabulary
- The loss or error is the distance between the prediction and the target
Language Model using Neural Networks

- Trained using a Recurrent Neural Networks (RNNs):
  - Neural networks with feedback loops, allowing information to persist
  - Natural architecture for working with sequences

- With Long-Short-Term Memories (LSTMs):
  - RNNs with more complex units
  - To capture both long-term and short-term dependencies
Single Cell Visualization of Language Model trained on Linux Source Code

Cell that turns on inside comments and quotes:
```c
/* duplicate LSM field information. The lsm_rule is opaque, so
 * re-initialized. */
static inline int audit_dupe_lsm_field(struct audit_field *df,
                                 struct audit_field *sf)
{
    int ret = 0;
    char *lsm_str;
    /* our own copy of lsm_str */
    lsm_str = kstrdup(sf->lsm_str, GFP_KERNEL);
    if (unlikely(!lsm_str))
        return -ENOMEM;
    df->lsm_str = lsm_str;
    /* our own (refreshed) copy of lsm_rule */
    ret = security_audit_rule_init(df->type, df->op, df->lsm_str,
                                 (void **)&df->lsm_rule);
    /* Keep currently invalid fields around in case they
     * become valid after a policy reload. */
    if (ret == -EINVAL) {
        pr_warn("audit rule for LSM %s is invalid\n",
                df->lsm_str);
        ret = 0;
    }
    return ret;
}
```

Cell that is sensitive to the depth of an expression:
```c
#ifdef CONFIG_AUDITSYSCALL
static inline int audit_match_class_bits(int class, u32 *mask)
{
    int i;
    if (classes[class]) {
        for (i = 0; i < AUDIT_BITMASK_SIZE; i++)
            if (mask[i] & classes[class][i])
                return 1;
    }
    return 0;
}
```

http://karpathy.github.io/2015/05/21/rnn-effectiveness/
Cell that might be helpful in predicting a new line. Note that it only turns on for some ‘\n’:
char *audit_unpack_string(void **bufp, size_t *remain, size_t len)
{
    char *str;
    if (!bufp || (len == 0) || (len > *remain))
        return ERR_PTR(-EINVAL);
    /* Of the currently implemented string fields, PATH_MAX defines the longest valid length. */
    if (len > PATH_MAX)
        return ERR_PTR(-ENAMETOOLONG);
    str = kmalloc(len + 1, GFP_KERNEL);
    if (unlikely(!str))
        return ERR_PTR(-ENOMEM);
    memcpy(str, *bufp, len);
    str[len] = 0;
    *bufp += len;
    *remain -= len;
    return str;
}
Additional Reading

• Papers
    • https://arxiv.org/abs/1606.06737v2

• Blog Posts
  • Andrej Karpathy: The Unreasonable Effectiveness of Recurrent Neural Networks
    • http://karpathy.github.io/2015/05/21/rnn-effectiveness/
  • Chris Olah: Understanding LSTM Networks
    • http://colah.github.io/posts/2015-08-Understanding-LSTMs/
  • Chris Olah: Attention and Augmented Recurrent Neural Networks
    • https://distill.pub/2016/augmented-rnns/

• Code!
  • Keras: https://github.com/fchollet/keras-resources
  • TensorFlow: https://www.tensorflow.org/tutorials/
Research Ideas

• Uncertainty of Predictions in Recurrent Neural Networks
    • [http://mlg.eng.cam.ac.uk/yarin/blog_2248.html](http://mlg.eng.cam.ac.uk/yarin/blog_2248.html)
  • Tom Wiecki: Bayesian Deep Learning
    • [http://twiecki.github.io/blog/2016/06/01/bayesian-deep-learning/](http://twiecki.github.io/blog/2016/06/01/bayesian-deep-learning/)
  • Uber Engineering: Application Motivation

• Distributed Deep Learning of Recurrent Neural Networks
  • Scaling Out using Spark and Scaling Up using TensorFlow/Keras
    • [https://github.com/databricks/tensorframes](https://github.com/databricks/tensorframes)
    • [https://github.com/yahoo/TensorFlowOnSpark](https://github.com/yahoo/TensorFlowOnSpark)
    • [https://github.com/cerndb/dist-keras](https://github.com/cerndb/dist-keras)

Computer Vision Tasks and Convolutional Neural Networks

- Computer Vision Applications
  - Image Classification
  - Object Detection
  - Semantic Segmentation
  - Image Captioning
  - Style Transfer
  - Image Generation

- Common Across All Applications
  - Convolutional Neural Networks

Recommended Resource: Stanford CS231n: Convolutional Neural Networks for Visual Recognition:
http://cs231n.stanford.edu

9/18/17
Convolutional Neural Networks

Deep Learning learns layers of features
Convolutional Neural Networks