Explaining Deep Learning Models for Natural Language Processing

A Tutorial at ICON 2018

Kevin Patel & Himanshu Singh

December 15, 2018
Outline

1. Basics of Neural Network
   - Perceptron
   - Feed Forward Neural Network
   - Gradient Descent
   - Vanishing Gradient Problem

2. Essence of Deep Learning
   - Overview
   - Word Embeddings
   - Convolutional Neural Networks
   - Recurrent Neural Networks
Outline

3 Model Explainability
- Criteria for ML Systems Evaluation
- The Mythos of Model Interpretability
- Open the Black Box
- Interpretable Machine Learning
- Towards Data Science
- Shape of Data

4 Interpretable WE

5 Methods and Techniques for Interpretability
- Locally Interpretable Model-Agnostic Explanations (LIME)
- Layerwise Relevance Propagation (LRP)
- Integrated Gradients (IG)
Part I

By
Kevin
NLP through the Ages
1950 -- ~1990s → Write rules
1950 -- ~1990s → Write rules

1990s -- ~2000s → Corpus based statistics
NLP through the Ages

1950 -- ~1990s → Write rules
1990s -- ~2000s → Corpus based statistics
2000s -- ~2014 → Supervised Machine Learning
NLP through the Ages

1950 -- ~1990s → Write rules
1990s -- ~2000s → Corpus based statistics
2000s -- ~2014 → Supervised Machine Learning
2014 -- today → Deep Learning
NLP through the Ages

1950 -- ~1990s  →  Write rules  ←  Transparent

1990s -- ~2000s  →  Corpus based statistics

2000s -- ~2014  →  Supervised Machine Learning

2014 -- today  →  Deep Learning
NLP through the Ages

1950 -- ~1990s → Write rules ← Transparent

1990s -- ~2000s → Corpus based statistics

2000s -- ~2014 → Supervised Machine Learning

2014 -- today → Deep Learning ← Black Box
This is your machine learning system?

Yup! You pour the data into this big pile of linear algebra, then collect the answers on the other side.

What if the answers are wrong?

Just stir the pile until they start looking right.

https://xkcd.com/1838/
NLP through the Ages

1950 -- ~1990s → Write rules ← Transparent
1990s -- ~2000s → Corpus based statistics
2000s -- ~2014 → Supervised Machine Learning
2014 -- today → Deep Learning ← Black Box

What is desired today?
NLP through the Ages

1950 -- ~1990s → Write rules  ← Transparent
1990s -- ~2000s → Corpus based statistics
2000s -- ~2014 → Supervised Machine Learning
2014 -- today → Deep Learning  ← Black Box

What is desired today? → Write rules aided by ML/DL
Knowing the Right Question

- In case of model explainability, different people asking different questions
  - *Why did the model give this decision for this input? What is the underlying cause?*
  - *What is the relation between a particular feature and the model’s decision making?*
  - *Given a wrong prediction for a particular input, how to update the model to fix it?*
  - etc.
People still struggling with some questions from deep learning

- Why do we optimize on mean squared error or cross entropy? Why do we not directly optimize on accuracy?
- Why did we ditch perceptrons and move on to more complicated neurons?
Overall Game Plan

- Provide a brief overview of deep learning
  - Answering questions commonly asked by practitioners who interact with us
  - Should reduce unexpected behaviors observed by practitioners from their neural networks
- Provide a glimpse into the vast literature on model explainability
  - Highlight some techniques that are applicable in NLP
  - Demonstrate some of these techniques
Neural Network
Question 1

How to emulate basic decision making of the human brain?
Perceptron

- A simple artificial neuron (Rosenblatt, 1958)
- Input: one or more binary values \( x_i \)
- Output: single binary value \( y \)
- Output computed using weighted sum of inputs and a threshold
- Giving different weights to different features while making a decision
y = \begin{cases} 
1, & \text{if } \sum w_i x_i > \text{threshold} \\
0, & \text{otherwise}
\end{cases}
Some Conventions

- Inputs also treated as neurons (no input, output is the actual value of the feature)
- Rewrite $\sum w_i x_i$ as $w.x$
- Move threshold to the other side of the equation; Call it bias $b = -\text{threshold}$

$$y = \begin{cases} 
1, & \text{if } w.x + b > 0 \\
0, & \text{otherwise}
\end{cases}$$
Bias

- An indication of how easy it is for a neuron to fire
- The higher the value of bias, the easier it is for the neuron to fire
- Consider it as a prior inclination towards some decision
  - The higher your initial inclination, the smaller the amount of extra push needed to finally make some decision
Perceptron Example

Should I go to lab?

- $x_1$: My guide is here
- $x_2$: Collaborators are in lab
- $x_3$: The buses are running
- $x_4$: Tasty tiffin in the mess
- $x_0$: Bias: My inclination towards going to the lab no matter what
Perceptron Example

Should I go to lab?

- $x_1$: My guide is here
- $x_2$: Collaborators are in lab
- $x_3$: The buses are running
- $x_4$: Tasty tiffin in the mess
- $x_0$: Bias: My inclination towards going to the lab no matter what

What if $x_0 = -3$?
Perceptron Example

Should I go to lab?

- $x_1$: My guide is here
- $x_2$: Collaborators are in lab
- $x_3$: The buses are running
- $x_4$: Tasty tiffin in the mess
- $x_0$: Bias: My inclination towards going to the lab no matter what

What if $x_0 = -3$?
What if $x_0 = 1$?
Perceptron’s Interpretability

- Weight of different features indicate their corresponding importance
  - Directly indicating the working of the model
- Applicable to other linear models too
Question 2
How to fix those weights?
Learning

- Simple weight update rule to learn parameters of a single perceptron
- Perceptron Convergence Theorem guarantees that learning will converge to a correct solution in case of linearly separable data.
- However, learning is difficult in case of network of perceptrons
  - Ideally, a learning process involves changing one of the input parameters by a small value, hoping that it will change the output by a small value.
  - Here, a small change in parameters of a single perceptron ⇒ flipped output ⇒ change behavior of entire network
  - Need some machinery such that gradual change in parameters lead to gradual change in output
If \( y \) is a function of \( x \), then change in \( y \) i.e. \( \Delta y \) is related to change in \( x \) i.e. \( \Delta x \) as follows (Linear Approximation):

\[
\Delta y \approx \frac{dy}{dx} \Delta x
\]

**Example**

\[
f(x) = x^2
\]

\[
f'(x) = 2x
\]

\[
f(4.01) \approx f(4) + f'(4)(4.01 - 4)
\]

\[
= 16 + 2 \times 4 \times 0.01
\]

\[
= 16.08
\]
For a fixed datapoint with two features \((x_1, x_2)\), the change in output of the perceptron depends on the corresponding changes in weights \(w_1\) and \(w_2\) and the bias \(b\).

Thus, change in \(y\) - \(\Delta y\) is

\[
\Delta y \approx \frac{\partial y}{\partial w_1} \Delta w_1 + \frac{\partial y}{\partial w_2} \Delta w_2 + \frac{\partial y}{\partial b} \Delta b
\]
Question 3

Why did we move on to other neurons? Why can’t we stick to perceptrons?
Learning (contd.)

- For a fixed datapoint with two features \((x_1, x_2)\), the change in output of the perceptron depends on the corresponding changes in weights \(w_1\) and \(w_2\) and the bias \(b\).

- Thus, change in \(y - \Delta y\) is

\[
\Delta y \approx \frac{\partial y}{\partial w_1} \Delta w_1 + \frac{\partial y}{\partial w_2} \Delta w_2 + \frac{\partial y}{\partial b} \Delta b
\]

- Partial derivative **ill-defined** in case of perceptron, which creates hurdle for learning
  - This is why perceptrons were ditched!
Sigmoid Neurons

- Another simple artificial neuron (McCulloch and Pitts, 1943)
- Input: one or more real values $x_i$
- Output: single real value $y$
- Output computed by applying sigmoid function $\sigma$ on the weighted sum of inputs and bias

$$y = \sigma(w.x + b) = \frac{1}{1 + e^{-(w.x + b)}}$$

- Decision making using sigmoid:
  - Given real valued output, use threshold
  - If $y > 0.5$, output 1, else 0
- Partial derivative $\frac{\partial y}{\partial x} = \sigma(1 - \sigma)$
Sigmoid vs. Perceptron

Step (Perceptron)
- Sigmoid is continuous and differentiable over its domain
- Learning is possible via small changes in parameters and using linear approximation
Activation Functions

Linear

Sigmoid

Tanh

ReLU
Notations

- $x$: Input
- $w_{ij}^l$: weight from $j^{th}$ neuron in $(l-1)^{th}$ layer to $i^{th}$ neuron in $l^{th}$ layer
- $b_j^l$: bias of the $j^{th}$ neuron in the $l^{th}$ layer
- $z_j^l$: $w^l a^{l-1} + b_j^l$
- $a_j^l$: $f(z_j^l)$
For MNIST Digit recognition

- Input Layer $28 \times 28 = 784$ neurons
- Output Layer - 10? 4?
Question 4

Why 1-hot for output? Why can’t we use binary representation for class labels?
One-hot output vs. Binary encoded output

- Given 10 digits, we fixed 10 neurons in output layer
  - Why not 4 neurons, and generate binary representation of digits?
- The task is to observe features and learn to decide whether it is a particular digit
- Observing visual features and trying to predict, say, most significant bit, will be hard
  - Almost no correlation there
Feed Forward Computation

Given input $x$

- First layer
  - $z^1 = w^1 \cdot x + b^1$
  - $a^1 = \sigma(z^1)$

- Intermediate layers
  - $z^l = w^l \cdot a^{l-1} + b^l$
  - $a^l = \sigma(z^l)$

- $a^L$ is the output, where $L$ is the last layer

Note that output contains real numbers (due to $\sigma$ function)
Question 5

Why do we need separate loss functions? Why can’t we directly optimize on accuracy measure?
Loss functions

- Consider a network with parameter setting P1

<table>
<thead>
<tr>
<th>True</th>
<th>Predicted</th>
<th>Correct</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 1 0</td>
<td>0.3 0.4 0.3</td>
<td>yes</td>
</tr>
<tr>
<td>1 0 0</td>
<td>0.1 0.2 0.7</td>
<td>no</td>
</tr>
<tr>
<td>0 0 1</td>
<td>0.3 0.3 0.4</td>
<td>yes</td>
</tr>
</tbody>
</table>

- Number of correctly classified examples $= \frac{2}{3}$
- Classification error $= 1 - \frac{2}{3} = \frac{1}{3}$

- Consider same network with parameter setting P2

<table>
<thead>
<tr>
<th>True</th>
<th>Predicted</th>
<th>Correct</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 1 0</td>
<td>0.1 0.7 0.2</td>
<td>yes</td>
</tr>
<tr>
<td>1 0 0</td>
<td>0.3 0.4 0.3</td>
<td>no</td>
</tr>
<tr>
<td>0 0 1</td>
<td>0.1 0.2 0.7</td>
<td>yes</td>
</tr>
</tbody>
</table>

- Classification error still the same
- Need a smooth function of weights and biases
Loss functions (contd.)

- Mean Squared Error: \( \text{MSE} = \frac{1}{M} \sum (y_i - t_i)^2 \)

<table>
<thead>
<tr>
<th>True</th>
<th>Predicted</th>
<th>Correct</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1 0 0</td>
<td>0.3 0.4 0.3</td>
</tr>
<tr>
<td>1</td>
<td>0 0 0</td>
<td>0.1 0.2 0.7</td>
</tr>
<tr>
<td>0</td>
<td>0 1 0</td>
<td>0.3 0.3 0.4</td>
</tr>
</tbody>
</table>

- Mean Squared Error \( = (0.54 + 0.54 + 1.34)/3 = 0.81 \)

<table>
<thead>
<tr>
<th>True</th>
<th>Predicted</th>
<th>Correct</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1 0 0</td>
<td>0.1 0.7 0.2</td>
</tr>
<tr>
<td>1</td>
<td>0 0 0</td>
<td>0.3 0.4 0.3</td>
</tr>
<tr>
<td>0</td>
<td>0 1 0</td>
<td>0.1 0.2 0.7</td>
</tr>
</tbody>
</table>

- Mean Squared Error \( = (0.14 + 0.14 + 0.74)/3 = 0.34 \)

- Indicates that second parameter setting is better
Loss functions (contd.)

- **Mean Cross Entropy**
  
  \[ MCE = \frac{1}{M} \sum (-t_i \log y_i - (1 - t_i) \log(1 - y_i)) \]

<table>
<thead>
<tr>
<th>True</th>
<th>Predicted</th>
<th>Correct</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 1 0</td>
<td>0.3 0.4 0.3</td>
<td>yes</td>
</tr>
<tr>
<td>1 0 0</td>
<td>0.1 0.2 0.7</td>
<td>no</td>
</tr>
<tr>
<td>0 0 1</td>
<td>0.3 0.3 0.4</td>
<td>yes</td>
</tr>
</tbody>
</table>

- **Mean Cross Entropy**
  
  \[ MCE = -(\ln(0.4) + \ln(0.4) + \ln(0.1))/3 = 1.38 \]

<table>
<thead>
<tr>
<th>True</th>
<th>Predicted</th>
<th>Correct</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 1 0</td>
<td>0.1 0.7 0.2</td>
<td>yes</td>
</tr>
<tr>
<td>1 0 0</td>
<td>0.3 0.4 0.3</td>
<td>no</td>
</tr>
<tr>
<td>0 0 1</td>
<td>0.1 0.2 0.7</td>
<td>yes</td>
</tr>
</tbody>
</table>

- **Mean Cross Entropy**
  
  \[ MCE = -(\ln(0.7) + \ln(0.7) + \ln(0.3))/3 = 0.64 \]

- Indicates that second parameter setting is better
Minimizing Loss

- Consider a function $C$ that depends on some parameter $x$ as shown below:

![Graph of function C]

- How to find the value of $x$ for which $C$ is minimum?
- Idea: Choose a random value for $x$, place an imaginary ball there. It will eventually lead to a valley
- **Gradient Descent**!
Question 6

- Gradient Descent: \( x_{t+1} = x_t - \eta \frac{dC}{dx} \)
- Why is the second term subtracted? Why exactly that term?
Gradient Descent

- Recall $\Delta C \approx \frac{dC}{dx} \cdot \Delta x$
- We want to change $x$ such that $C$ is reduced i.e. $\Delta C$ has to be always negative
- What if we choose $\Delta x = -\eta \frac{dC}{dx}$?

\[
\Delta C \approx \frac{dC}{dx} \cdot \Delta x = \frac{dC}{dx} \cdot (-\eta \frac{dC}{dx}) = -\eta \cdot \left(\frac{dC}{dx}\right)^2 \leq 0
\]

- Gradient Descent: $x_{t+1} = x_t - \eta \frac{dC}{dx}$
Gradient Descent

![Graph showing Gradient Descent](image_url)
Training Neural Network Using Gradient Descent

- Weight of the neural networks → Input variables
- Loss function → Output variables
- Use Gradient descent
- How to compute gradient of loss function with respect to weights near the input layer?
Back Propagation

- Effective use shown in (Williams et al., 1986)
- Every neuron taking part in the decision
- Every neuron shares the blame for error
- Decision made by a neuron dependent on its weights and biases
- Thus error is caused due to these weights and biases
- Need to change weights and biases such that overall error is reduced
  - Can use gradient descent here
  - For a weight $w^k_{ij}$, the weight update will be

$$w^k_{ij} \leftarrow w^k_{ij} - \eta \frac{\partial C}{\partial w^k_{ij}}$$
Back Propagation (contd.)

Will use calculus chain rule to obtain the partial derivatives

1. First find error for each neuron on the last layer
   \[ \delta^L = \nabla_a C \odot \sigma'(z^L) \]

2. Then find error for each neuron on the interior neurons
   \[ \delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l) \]

3. Update weights and biases using following gradients:
   \[ \frac{\partial C}{\partial b^l_j} = \delta^l_j \]
   \[ \frac{\partial C}{\partial w^l_{jk}} = a^{l-1}_k \delta^l_j \]
Question 7

What exactly is the Vanishing Gradient problem?
Vanishing Gradient Problem

- Observed by Hochreiter (1991)
- Important point: the $\sigma'(z^l)$ term in the previous steps
- Derivative of the activation function
- Will be multiplied at each layer during back propagation
- Example: 3 layer network

\[
\begin{align*}
\delta^4 &= A.\sigma'() \\
\delta^3 &= X.\delta^4.\sigma'() = X.A.\sigma'().\sigma'() \\
\delta^2 &= Y.\delta^3.\sigma'() = Y.X.A.\sigma'().\sigma'().\sigma'() \\
\delta^1 &= Z.\delta^2.\sigma'() = Z.Y.X.A.\sigma'().\sigma'().\sigma'().\sigma'()
\end{align*}
\]
Vanishing Gradient Problem (contd.)

Sigmoid

- Maximum value of sigmoid’s derivative $= 0.25$
- $0.25^n \approx 0$ as $n \rightarrow \infty$
- Gradient tends to 0 i.e. vanishes

Derivative of Sigmoid
TensorFlow Playground Demo for Vanishing Gradient

Tinker With a **Neural Network** Right Here in Your Browser.
Don’t Worry, You Can’t Break It. We Promise.

**DATA**
Which dataset do you want to use?

**FEATURES**
Which properties do you want to feed in?

**OUTPUT**
Test loss 0.501
Training loss 0.500

Colors show data, neuron and weight values.

[Show test data] [Discretize output]
Deep Learning
Question 8

What is Deep Learning?
Deep Learning

- Set of techniques and architectures that tackles such learning problems and helps to reach optimal parameters faster
- Various methods:
  - Start at near optimal values of parameters so smaller updates due to vanishing gradients is not much of a problem
  - Use better activation functions which can avoid such problems
  - Use better optimizers than standard gradient descent
  - Use novel architectures
  - Use better data representations
Tackling Vanishing Gradients via Greedy Unsupervised Pretraining

Proposed by Bengio et al. (2007)
Tackling Vanishing Gradients via Greedy Unsupervised Pretraining

Proposed by Bengio et al. (2007)
Tackling Vanishing Gradients via Greedy Unsupervised Pretraining

Proposed by Bengio et al. (2007)
Tackling Vanishing Gradients via Greedy Unsupervised Pretraining

Proposed by Bengio et al. (2007)
Tackling Vanishing Gradients via Greedy Unsupervised Pretraining

Proposed by Bengio et al. (2007)
Tackling Vanishing Gradients via Novel Activation Functions

- Rectified Linear Unit (Nair and Hinton, 2010): Derivative = 1 when non-zero, else 0
- Product of derivatives does not vanish
- But once a ReLU gets to 0, it is difficult to get it to one again (Dead ReLU problem)
  - Addressed by better variants such as Leaky ReLU (Maas et al., 2013), Parametric ReLU (He et al., 2015), etc.
Types of Gradient Descent

Based on the amount of data used for training

- **Batch GD**: all training data per update, slow, not applicable in online setting, but guaranteed to converge to global minimum for convex and local minimum for non-convex.

- **Stochastic GD**: one training datapoint per update, fluctuates a lot, allows to jump to new and potentially better local minima, this complicates convergence, has been shown that by decreasing learning rate almost certainly converges to global in convex and local in non-convex.

- **Mini-batch GD**: batch of \( n \) datapoints per update, best of both worlds - relatively stable convergence and can use matrix operations for batches.
Tackling Learning Difficulties via Optimizers

- SGD mainly used for a long time
- Converges slowly
- Can get stuck in local minima
SGD + Momentum

- Developed by Qian (1999)
Nesterov Accelerated Gradient

Developed by Nesterov (1983)
Other Optimizers

- **AdaGrad**: Adapts the learning rate to the parameters, performing larger updates for infrequent and smaller updates for frequent parameters *Duchi et al. (2011)*
- **AdaDelta**: Does not need an initial learning rate *Zeiler (2012)*
- **RMSProp**: Good with recurrent networks; Unpublished method from Hinton’s Coursera Lectures
- **Adam**: Benefits of RMSProp and AdaDelta mixed with momentum tricks *Kingma and Ba (2014)*
Novel architectures made to specific problems

Example:
- Derivative of activation function in LSTM is identity function is 1. Gradient does not vanish
- Effective weight depends on forget gate activation, whose derivative is never $> 1.0$. So Gradient does not explode
In simple terms, Machine Learning comprises of

- Representing data in some numeric form
- Learning some function on that representation
In simple terms, Machine Learning comprises of:
- Representing data in some numeric form
- Learning some function on that representation
In simple terms, Machine Learning comprises of:
- Representing data in some numeric form
- Learning some function on that representation
In simple terms, Machine Learning comprises of:
- Representing data in some numeric form
- Learning some function on that representation
In simple terms, Machine Learning comprises of:
- Representing data in some numeric form
- Learning some function on that representation
In simple terms, Machine Learning comprises of:
- Representing data in some numeric form
- Learning some function on that representation
**Gist of Machine Learning**

- In simple terms, Machine Learning comprises of:
  - Representing data in some numeric form
  - Learning some function on that representation

![Diagram showing sentiment analysis]

- How to place words to learn, say, Binary Sentiment Classification?
  - Good: Positive
  - Awesome: Positive
  - Bad: Negative

Kevin and Himanshu
Representations for Learning Algorithms

- Detect whether the following image is dog or not?
  ![Dog Image]

  - Basic idea: feed raw pixels as input vector
  - Works well:
    - Inherent structure in the image

- Detect whether a word is a dog or not?
  Labrador

  - Nothing in spelling of *labrador* that can connect it to *dog*
  - Need a representation of *labrador* which indicates that it is a dog
Local Representations

- Information about a particular item located solely in the corresponding representational element (dimension)
- Effectively one unit is turned on in a network, all the others are off
- No sharing between represented data
- Each feature is independent
- No generalization on the basis of similarity between features
Distributed Representations

- Information about a particular item distributed among a set of (not necessarily) mutually exclusive representational elements (dimensions)
  - One item spread over multiple dimensions
  - One dimension contributing to multiple items
- A new input is processed similar to samples in training data which were similar
  - Better generalization
## Distributed Representations: Example

<table>
<thead>
<tr>
<th>Number</th>
<th>Local Representation</th>
<th>Distributed Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1 0 0 0 0 0 0 0</td>
<td>0 0 0</td>
</tr>
<tr>
<td>1</td>
<td>0 1 0 0 0 0 0 0</td>
<td>0 0 1</td>
</tr>
<tr>
<td>2</td>
<td>0 0 1 0 0 0 0 0</td>
<td>0 1 0</td>
</tr>
<tr>
<td>3</td>
<td>0 0 0 1 0 0 0 0</td>
<td>0 1 1</td>
</tr>
<tr>
<td>4</td>
<td>0 0 0 0 1 0 0 0</td>
<td>1 0 0</td>
</tr>
<tr>
<td>5</td>
<td>0 0 0 0 0 1 0 0</td>
<td>1 0 1</td>
</tr>
<tr>
<td>6</td>
<td>0 0 0 0 0 0 1 0</td>
<td>1 1 0</td>
</tr>
<tr>
<td>7</td>
<td>0 0 0 0 0 0 0 1</td>
<td>1 1 1</td>
</tr>
</tbody>
</table>
Question 9

So what is a word embedding?
Word Embeddings: Intuition

- Word Embeddings: distributed vector representations of words such that the similarity among vectors correlate with semantic similarity among the corresponding words.
  
  Given that $\text{sim}(\text{dog, cat})$ is more than $\text{sim}(\text{dog, furniture})$, $\cos(\text{dog, cat})$ is greater than $\cos(\text{dog, furniture})$.

- Such similarity information uncovered from context.
Word Embeddings: Intuition

- Word Embeddings: distributed vector representations of words such that the similarity among vectors correlate with semantic similarity among the corresponding words

  Given that \( \text{sim}(\text{dog}, \text{cat}) \) is more than \( \text{sim}(\text{dog}, \text{furniture}) \),
  \( \text{cos}(\text{dog}, \text{cat}) \) is greater than \( \text{cos}(\text{dog}, \text{furniture}) \)

- Such similarity information uncovered from context

- Consider the following sentences:
  - I like sweet food .
  - You like spicy food .
  - They like \( \text{xyzabc} \) food .

- What is \( \text{xyzabc} \) ?

- Meaning of words can be inferred from their neighbors (context) and words that share neighbors

  - Neighbors of \( \text{xyzabc} \): \{like, food\}
  - Words that share neighbors of \( \text{xyzabc} \): \{sweet, spicy\}
Question 10

So do word embeddings capture meaning?
Modelling Meaning via Word Embeddings

- Geometric metaphor of meaning (Sahlgren, 2006):
  - Meanings are locations in semantic space, and semantic similarity is proximity between the locations.
Modelling Meaning via Word Embeddings

- Geometric metaphor of meaning (Sahlgren, 2006):
  - Meanings are locations in semantic space, and semantic similarity is proximity between the locations.

- Distributional Hypothesis (Harris, 1970)
  - Words with similar distributional properties have similar meanings
Modelling Meaning via Word Embeddings

- Geometric metaphor of meaning (Sahlgren, 2006):
  - Meanings are locations in semantic space, and semantic similarity is proximity between the locations.

- Distributional Hypothesis (Harris, 1970)
  - Words with similar distributional properties have similar meanings
  - Only differences in meaning can be modelled
Modelling Meaning via Word Embeddings

- Geometric metaphor of meaning (Sahlgren, 2006):
  - Meanings are locations in semantic space, and semantic similarity is proximity between the locations.

- Distributional Hypothesis (Harris, 1970)
  - Words with similar distributional properties have similar meanings
  - Only differences in meaning can be modelled
Modelling Meaning via Word Embeddings

- Geometric metaphor of meaning (Sahlgren, 2006):
  - Meanings are locations in semantic space, and semantic similarity is proximity between the locations.

- Distributional Hypothesis (Harris, 1970)
  - Words with similar distributional properties have similar meanings
  - Only differences in meaning can be modelled
Entire Vector vs. Individual dimensions

- Only proximity in the entire space is represented
- Individual dimensions do not mean anything
  - Those embedding models where individual dimensions do have some meaning, are known as *interpretable models*
Co-occurrence matrix (Rubenstein and Goodenough, 1965)
- A mechanism to capture distributional properties
- Rows of co-occurrence matrix can be directly considered as word vectors

Neural Word Embeddings
- Vector representations learnt using neural networks - Bengio et al. (2003); Collobert and Weston (2008); Mikolov et al. (2013)
Skip Gram

- Proposed by Mikolov et al. (2013)
- Predict Context given word
Given a sequence of training words $w_1, w_2, \ldots, w_T$, maximize

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

where

$$
p(w_O | w_I) = \frac{\exp(u_{w_O}^T v_{w_I})}{\sum_{w=1}^{W} \exp(u_w^T v_{w_I})}
$$
Visualizing Skip Gram Training

Skip Gram Visualization

Training Data:
- drink apple juice
- drink mango juice
- drink orange juice
- eat apple pie
- eat mango fruit

Neurons:
- apple
- drink
- eat
- fruit
- juice
- mango
- orange
- pie

Vector Maps:
- Input Vectors
- Output Vectors

Credits: https://ronxin.github.io/wevi/
Kevin and Himanshu
Explaining DL Models for NLP
Part II

By
Himanshu
Convolutions

http://deeplearning.stanford.edu/wiki/index.php/Feature_extraction_using_convolution
Convolutions

Convolution

```
http://deeplearning.stanford.edu/wiki/index.php/Feature_extraction_using_convolution
```
Convolution

http://deeplearning.stanford.edu/wiki/index.php/Feature_extraction_using_convolution
Convolutions

http://deeplearning.stanford.edu/wiki/index.php/Feature_extraction_using_convolution
Convolution

http://deeplearning.stanford.edu/wiki/index.php/Feature_extraction_using_convolution
Convolution

HTTP://DEEPLearning.STANford.EDU/WIKI/INDEX.php/FEATURE_EXTRACTION_USING_CONVOLUTION
Convolution

http://deeplearning.stanford.edu/wiki/index.php/Feature_extraction_using_convolution
Convolution

http://deeplearning.stanford.edu/wiki/index.php/Feature_extraction_using_convolution
Convolution

http://deeplearning.stanford.edu/wiki/index.php/Feature_extraction_using_convolution
Why Convolution?

Averaging each pixel with its neighboring values blurs an image:

![Taj Mahal](https://docs.gimp.org/en/plug-in-convmatrix.html)

Taking the difference between a pixel and its neighbors detects edges:

![Edge Detection](https://docs.gimp.org/en/plug-in-convmatrix.html)
What is Convolutional Neural Network?

**CNN**

It is several layers of convolutions with nonlinear activation functions like ReLU or tanh applied to the results

- During the training phase, a CNN automatically learns the values of its filters based on the task you want to perform.
- **Location Invariance**: Let’s say you want to classify whether or not there’s an elephant in an image. Because you are sliding your filters over the whole image you don’t really care where the elephant occurs.
- **Compositionality**: Each filter composes a local patch of lower-level features into higher-level representation.
What has CNN for NLP?

Problems with CNN

- **Location Invariance**: You probably do care a lot where in the sentence a word appears unlike images.

- **Local Compositionality**: Pixels close to each other are likely to be semantically related (part of the same object), but the same isn’t always true for words. In many languages, parts of phrases could be separated by several other words.

- **Compositional aspect is intuitive in Computer Vision** i.e. edges form shapes and shapes form form objects. Clearly, words compose in some ways, like an adjective modifying a noun, but how exactly this works what higher level representations actually “mean” isn’t as obvious as in the Computer Vision case.
Why not a traditional neural network for sequential task?

Problems:
- Inputs and outputs can be of different lengths in different examples
- Traditional NN doesn’t share features learned across different positions of text

**Recurrent Neural Network**

RNN solves above two problems along with the problems posed by CNNs.
An Unrolled RNN

NOTE: Hidden state ($h_t$) tells us summary of the sequence till time $t$

Forward pass

$$h_t = \tanh(W h_{t-1} + U x_t + b_h)$$

$$z_t = \text{softmax}(V h_t + b_z)$$
Backpropagation in RNN

Notation: \( E(x, y) = - \sum_t y_t \log z_t \)
\( E \): above objective function (i.e. sum of errors at all time stamps)
\( E(t) \): to indicate the output at time \( t \)

We have \( h_t = \tanh(Wh_{t-1} + Ux_t + b_h) \)
\( z_t = \text{softmax}(Vh_t + b_z) \)

Gradient of \( E \) w.r.t \( V \)
let \( \alpha_t = Vh_t + b_z \) then
\[
\frac{\partial E}{\partial V} = \sum_t \frac{\partial E}{\partial \alpha_t} \frac{\partial \alpha_t}{\partial V}
\]

\[
\frac{\partial E}{\partial \alpha_t} \text{ is derivative of softmax function w.r.t it’s input } \alpha_t
\]
\[
\frac{\partial E}{\partial \alpha_t} = z_t - y_t \text{ and } \frac{\partial \alpha_t}{\partial V} = h_t
\]
Backpropagation in RNN

We have

\[ h_t = \tanh(W h_{t-1} + U x_t + b_h) \]
\[ z_t = \text{softmax}(V h_t + b_z) \]
\[ E = -\sum_t y_t \log z_t \]

**Gradient of E w.r.t W**

\[
\frac{\partial E(t)}{\partial W} = \frac{\partial E(t)}{\partial z_t} \frac{\partial z_t}{\partial W} = \frac{\partial E(t)}{\partial z_t} \frac{\partial z_t}{\partial h_t} \frac{\partial h_t}{\partial W}
\]

from forward pass equations, \( h_t \) partially depends on \( h_{t-1} \)

\[
\frac{\partial E(t)}{\partial W} = \frac{\partial E(t)}{\partial z_t} \frac{\partial z_t}{\partial h_t} \frac{\partial h_t}{\partial h_{t-1}} \frac{\partial h_{t-1}}{\partial W}
\]

Keep substituting \( h_{t-1} \) in \( h_t \), we see \( h_t \) depends on \( h_{t-2}, h_{t-3} \) ...

\[
\frac{\partial E(t)}{\partial W} = \sum_{k=1}^t \frac{\partial E(t)}{\partial z_t} \frac{\partial z_t}{\partial h_t} \frac{\partial h_t}{\partial h_k} \frac{\partial h_k}{\partial W}, \text{ and}
\]
\[
\frac{\partial E}{\partial W} = \sum_t \sum_{k=1}^t \frac{\partial E(t)}{\partial z_t} \frac{\partial z_t}{\partial h_t} \frac{\partial h_t}{\partial h_k} \frac{\partial h_k}{\partial W}
\]
**Backpropagation in RNN**

We have

\[ h_t = \tanh(W h_{t-1} + U x_t + b_h) \]
\[ z_t = \text{softmax}(V h_t + b_z) \]
\[ E = - \sum_t y_t \log z_t \]

**Gradient of \( E \) w.r.t \( U \)**

We can’t consider \( h_{t-1} \) as constant when taking partial derivative of \( h_t \) w.r.t \( U \) because \( h_{t-1} \) depends on \( U \) i.e.

\[ h_{t-1} = \tanh(W h_{t-2} + U x_{t-1} + b_h) \]

Again, we get a similar form

\[
\frac{\partial E}{\partial U} = \sum_t \sum_{k=1}^t \frac{\partial E(t)}{\partial z_t} \frac{\partial z_t}{\partial h_t} \frac{\partial h_t}{\partial h_k} \frac{\partial h_k}{\partial U}
\]
Problem with RNN

Look closely to these equations:
\[
\frac{\partial E}{\partial W} = \sum_t \sum_{k=1}^{t} \frac{\partial E(t)}{\partial z_t} \frac{\partial z_t}{\partial h_t} \frac{\partial h_t}{\partial h_k} \frac{\partial h_k}{\partial W}
\]
\[
\frac{\partial E}{\partial U} = \sum_t \sum_{k=1}^{t} \frac{\partial E(t)}{\partial z_t} \frac{\partial z_t}{\partial h_t} \frac{\partial h_t}{\partial h_k} \frac{\partial h_k}{\partial U}
\]

We find out that \( \frac{\partial h_t}{\partial h_k} \) is again a chain rule.
\[
\frac{\partial h_t}{\partial h_k} = \frac{\partial h_t}{\partial h_{k-1}} \frac{\partial h_{k-1}}{\partial h_{k-2}} \cdots \frac{\partial h_{k+1}}{\partial h_k}
\]

- If sequence length is large then there will be more number of terms in the product which will result in vanishing gradient problem or exploding gradient problem depending on whether each individual value is less/greater than 1.
- LSTM solves this problem to a large extent.
Long Short Term Memory (LSTM) Network

Forward Pass

\[
f_t = \sigma(W_f[h_{t-1}; x_t] + b_f)
\]
\[
i_t = \sigma(W_i[h_{t-1}; x_t] + b_i)
\]
\[
a_t = \tanh(W_a[h_{t-1}; x_t] + b_a)
\]
\[
C_t = f_t \times C_{t-1} + i_t \times a_t
\]
\[
o_t = \sigma(W_o[h_{t-1}; x_t] + b_o)
\]
\[
h_t = o_t \times \tanh(C_t)
\]
Vanishing Gradient Problem Addressed

It runs straight down the entire chain, with only some minor linear interactions. It’s very easy for information to just flow along it unchanged. (Mathematical proof on later slides)
Gates in LSTM

**Forget Gate**
Decides what information should be thrown away from the cell state

\[ f_t = \sigma(W_f[h_{t-1}; x_t] + b_f) \]
Gates in LSTM

Input Gate

σ layer decides which values to update and $a_t$ is a vector of new candidate values

$$i_t = \sigma(W_i[h_{t-1}; x_t] + b_i)$$
Gates in LSTM

**Updating Memory Cell**

Multiply the old state by $f_t$, forgetting the things we decided to forget earlier.

Then we add $i_t \ast a_t$. This is the new candidate values, scaled by how much we decided to update each state value.

$$C_t = f_t \ast C_{t-1} + i_t \ast a_t$$
Gates in LSTM

**Output Gate**

Output will be based on our cell state, but will be a filtered version. Cell state is put through tanh to push the output between -1 and 1.

\[
\begin{align*}
o_t &= \sigma(W_o[h_{t-1}; x_t] + b_o) \\
h_t &= o_t \cdot \tanh(C_t)
\end{align*}
\]
Backpropagation in LSTM

Error propagation

Error propagation happens through $C_t$ and $h_t$
Backpropagation in LSTM

Error propagation

Error propagation through $h_t$

\[ \Delta C_t = \frac{\partial E}{\partial C_t} \]

\[ \Delta o_t = \frac{\partial E}{\partial o_t} \]

\[ \Delta h_t = \frac{\partial E}{\partial h_t} \]
Backpropagation in LSTM

\[
\begin{align*}
o_t &= \sigma(W_o[h_{t-1}; x_t] + b_o) \\
h_t &= o_t \ast \text{tanh}(C_t)
\end{align*}
\]
Backpropagation in LSTM

\[ o_t = \sigma(W_o[h_{t-1}; x_t] + b_o) \]

\[ h_t = o_t \times \tanh(C_t) \]
Backpropagation in LSTM

Error propagation

Error propagation through $C_t$

\[ \Delta C_{t-1} = \frac{\partial E}{\partial C_{t-1}} \]

\[ \Delta C_t = \frac{\partial E}{\partial C_t} \]

\[ \Delta f_t = \frac{\partial E}{\partial f_t} \]

\[ \Delta i_t = \frac{\partial E}{\partial i_t} \]

\[ \Delta a_t = \frac{\partial E}{\partial a_t} \]

\[ \Delta o_t = \frac{\partial E}{\partial o_t} \]
Backpropagation in LSTM

\[ C_t = f_t \cdot C_{t-1} + i_t \cdot a_t \]
### Backpropagation in LSTM

**NOTE**: This $\Delta C_t$ will be used at $(t - 1)^{th}$ timestamp for further error propagation. If $f$ is close to 1 then gradient from $t^{th}$ timestamp is propagated perfectly to $(t - 1)^{th}$ timestamp.

\[
\Delta C_{t-1} = \Delta C_t \cdot o_f_i \\
C_t = f_t \cdot C_{t-1} + i_t \cdot a_t
\]
Backpropagation in LSTM

\[ \Delta C_{t-1} = \Delta C_t \circ f_t \]

\[ \Delta f_t = \Delta C_t \circ C_{t-1} \]

\[ C_t = f_t \ast C_{t-1} + i_t \ast a_t \]
Backpropagation in LSTM

\[ C_t = f_t \ast C_{t-1} + i_t \ast a_t \]
Backpropagation in LSTM

\[ \Delta C_{t-1} = \Delta C_t \circ f_t, \]

\[ \Delta f_t = \Delta C_t \circ C_{t-1}, \]

\[ \Delta i_t = \Delta C_t \circ a_t, \]

\[ \Delta a_t = \Delta C_t \circ i_t, \]

\[ C_t = f_t \times C_{t-1} + i_t \times a_t \]

Kevin and Himanshu

Explaining DL Models for NLP

88
Backpropagation in LSTM

Combined Error

Error propagation from $C_t$ and $h_t$ both

$$\Delta W_f = \frac{\partial E}{\partial W_f}$$
$$\Delta W_i = \frac{\partial E}{\partial W_i}$$
$$\Delta W_o = \frac{\partial E}{\partial W_o}$$

Kevin and Himanshu

Explaining DL Models for NLP
Backpropagation in LSTM

\[ \Delta W_f = \frac{\partial E}{\partial f_i} \frac{\partial f_i}{\partial W_f} \]

\[ = \Delta f_i \sigma f_i (1 - f_i) \sigma(h_{t-1} ; x_t) \quad (f_i = \sigma(W_f[h_{t-1} ; x_t])) \]
Backpropagation in LSTM

\[
\Delta W_i = \frac{\partial E}{\partial i_t} \frac{\partial i_t}{\partial W_i} = \Delta i_t \sigma (1 - i_t)[h_{t-1} : x_t]
\]

\( (i_t = \sigma(W_i[h_{t-1} : x_t])) \)
### Backpropagation in LSTM

\[
\Delta W_a = \frac{\partial E}{\partial a_t} \cdot \frac{\partial a_t}{\partial W_a} = \Delta a_t \cdot \sigma (1 - a_t^2) [h_{t-1} ; x_t]
\]

\[
\sigma_t = \tanh(W_a [h_{t-1} ; x_t])
\]
Backpropagation in LSTM

\[ C_{t-1} \]

\[ C_t \]

\[ f_t \]

\[ i_t \]

\[ a_t \]

\[ \text{tanh} \]

\[ o_t \]

\[ h_{t-1} \]

\[ x_t \]

\[ \sigma \]

\[ W_f \]

\[ \Delta W_f \]

\[ W_i \]

\[ \Delta W_i \]

\[ W_a \]

\[ \Delta W_a \]

\[ W_o \]

\[ \Delta W_o = \frac{\partial E}{\partial W_o} \]

\[ \Delta O_t = \frac{\partial E}{\partial O_i} \frac{\partial O_i}{\partial W_o} \]

\[ \Delta O_t = \Delta O_i o O_t o (1 - O_i)[h_{t-1} ; x_t] \]

\[ (O_t = \sigma(W_o[h_{t-1} ; x_t])) \]
Backpropagation in LSTM

\[
\Delta h_{t-1} = \frac{\partial E}{\partial h_{t-1}} = \frac{\partial E}{\partial f_t} \frac{\partial f_t}{\partial h_{t-1}} + \frac{\partial E}{\partial i_t} \frac{\partial i_t}{\partial h_{t-1}} + \frac{\partial E}{\partial a_t} \frac{\partial a_t}{\partial h_{t-1}} + \frac{\partial E}{\partial o_t} \frac{\partial o_t}{\partial h_{t-1}}
\]
Backpropagation in LSTM

NOTE: $\Delta h_{t-1}$ calculated here will be used by previous timestamp for further back propagation.

\[ \Delta h_{t-1} = \frac{\partial E}{\partial h_{t-1}} = \frac{\partial E}{\partial f_t} \frac{\partial f_t}{\partial h_{t-1}} + \frac{\partial E}{\partial i_t} \frac{\partial i_t}{\partial h_{t-1}} + \frac{\partial E}{\partial a_t} \frac{\partial a_t}{\partial h_{t-1}} + \frac{\partial E}{\partial O_t} \frac{\partial O_t}{\partial h_{t-1}} \]

\[ \Delta h_{t-1} = \Delta f_t \sigma f_t o (1 - f_t) W_{h_t} + \Delta i_t \sigma i_t o (1 - i_t) W_{h_i} + \Delta a_t \sigma (1 - a_t^2) W_{a_o} + \Delta O_t \sigma O_t o (1 - O_t) W_{o_o} \]

\[ f_t = \sigma([w_f; w_{f_t}] h_{t-1} ; x_t) \]
We have calculated $\Delta W_f, \Delta W_i, \Delta W_a$ and $\Delta W_o$. Next step is to do gradient descent:

$W^* = W^* - \alpha \Delta W^*$ where $* \in f, i, a, o$
Part III

By
Kevin
Model Explainability
Knowing the Right Question

In case of model explainability, different people asking different questions

- Why did the model give this decision for this input? What is the underlying cause?
- What is the relation between a particular feature and the model’s decision making?
- Given a wrong prediction for a particular input, how to update the model to fix it?
- etc.
Lipton (2016)

Motives for interpretability and technical descriptions of interpretable models are *diverse* and *occasionally discordant*, suggesting that interpretability refers to *more than one concept*.

Interpretability is not a monolithic concept, but in fact reflects several distinct ideas.
Current Situation

- Audience from academia and industry
- Audience interested in both applying interpretability (end users) and researching (researchers) on interpretability
- Plethora of articles/papers/techniques on model explainability/interpretability
  - End users confused among the options
  - Researchers frustrated while pitching their papers
    That includes me !!!
Game Plan

1. Stick to a popular paper to establish some definitions
2. Clarify basic definitions
   - End users will know the precise terms while defining their requirements and searching for solutions
   - Researchers will know how to pitch their work so that it is more acceptable to the community
3. Discuss and demonstrate a couple of existing techniques
   - End users can apply them
4. Discuss boundaries of existing state of model interpretability
   - Researchers can push those boundaries
Different Definitions

- Big Picture Criteria by Michie (1988)
- Survey papers: Lipton (2016); Guidotti et al. (2018), etc.
- Blogs: Towards Data Science, Shape of Data
Michie (1988) proposed 3 criteria to evaluate machine learning systems

- **Weak criterion**: a system improves its performance on unseen data based on learning from a sample of data.
- **Strong criterion**: weak plus the system is able to communicate the learned hypotheses in explicit symbolic form.
- **Ultra-strong criterion**: strong plus the user should be able to comprehend the system’s output and its possible consequences.
Explanation, Interpretation, Comprehension?

- Standard machine learning only addresses the weak criterion
  - Learning can be performed with high predictive accuracy
- We are operating near the level of strong criteria
Question 11

What is Interpretability?
Definition of Interpretability

- Many papers claim some models are more interpretable than the others
  - But they do not precisely define interpretability
- Motives and technical descriptions of interpretability are diverse and occasionally discordant
- Not a monolithic concept - means different things for different people
- Papers use words like **interpretable, explainable, intelligible, transparent and understandable** interchangeably
- Common thread - interpretability is something other than performance

---

Lipton (2016)
Question 12
What do people want from machine learning models?
Expectation from Machine Learning models

- We want good models
  - Goodness measured by metrics
- But, we also want interpretable models
- So we want something that cannot be obtained by improving performance on metrics
  - Metrics/loss functions fundamentally mismatched from real life objectives
- Contribution of paper: refine discourse on interpretability by introducing specific terms for both objectives and methods
Question 13

What can an interpretable model do?
Increase Trust

- Does the model know when it’s uncertain?
- Does the model make the same mistakes as a human would in its stead?
- Are we comfortable replacing a human with the model?
Causality

- What can the model tell us about the natural world?
- Generally, supervised models are trained to make predictions, but are used to make decisions and take actions accordingly
  - They mainly learn correlation, and not cause
Transferability

- Machine learning model trained in a controlled setting
- Will it perform in a similar fashion when deployed?
- In other words, has the model truly learnt to detect underlying phenomenon or is it mimicing the artifacts of training data?
  - Need sanity checks
Informativeness

- A model may be trained to make a decision
- But it could also be used to aid a person in making a decision
- Can it provide useful information of this kind?
Question 14

What properties of a model can confer it interpretability?
Two main categories of proposed solutions:
- Transparency: answers how does the model work
- Explainability: deals with model’s ability to provide post-hoc explanation
Simulatability

- One way to define transparency
- A model is transparent if a person can step through the algorithm in reasonable time
Let’s simulate a decision tree. Follow the right path and do the action

--

Credits: Been Kim’s talk *Introduction to Interpretable Machine Learning* CVPR 2018
Is this transparent and simulatable?

# people < 100

Weather = Rainy

Left

Weather = Sunny

Right

Stomp

Clap!
Is this transparent and simulatable?

Weather = Sunny

Time = morning
- Left
- Right

Time = afternoon
- Stomp
- Clap!
Is this transparent and simulatable?
Is this transparent and simulatable?

- What if we had a larger tree?
- Easily possible if, say one tries to learn decision trees on word embeddings
- Can we truly explain the overall logic of the system?
- After going through the tree with multiple datapoints, can we figure out which features were important?
Can the whole be understood by the sum of its parts?

Figuring out working of a model in terms of its individual components

Example: Nodes of a decision tree, Weights of a linear model
A weaker notion with the only expectation being that we understand the behavior of the algorithm

Example: Convergence of convex optimizations
This mainly entails extracting some explanation from the model in terms of input values.

Tries to uncover the relation between a particular prediction and the corresponding input features.

Thereby can be used to answer questions such as *what if I change this input feature?*
Verbal Explanations

- Generate natural language explanations
- Closer to human generated explanations
- Train another model (separately, or may be jointly) to generate explanations
- Joint image recognition and caption generation - here captions can be considered as interpretations of object predictions
Most of the time, uncovering the relationship between the input and the output is impossible over the entire domain of the image.

- Idea - use local explanations

- Example: Locally linear estimators
Case Based Explanations

- Explain the current prediction in terms of other cases where similar predictions were made
- Analogy - k-nearest neighbors
Discussion Points

- Linear models not strictly more interpretable than deep learning
- Claims about interpretability must be qualified
- Transparency may be at odds with the goals of AI
- Post-hoc interpretations may potentially mislead
Different Formulations for Model Explanation

Guidotti et al. (2018) provides the following formulations for model explanations:

- Black Box Model Explanation
- Black Box Outcome Explanation
- Black Box Inspection
- Transparent Box Design
Black Box Model Explanation

Training Data: D=(X,Y) → Black Box Learning → Black Box Predictor: B → Black Box Model Explanation → Explainable Predictor: E
Black Box Outcome Explanation

Training Data: $D=(X,Y)$

Black Box Learning

Black Box Predictor: $B$

Black Box Outcome Explanation

Test Input: $X'$

Explanation for $X'$
Black Box Inspection

Training Data: D=(X,Y) → Black Box Learning → Black Box Predictor: B → Black Box Inspection → Behavior of B

Test Input: X'

Iris Datapoint
Transparent Box Design

- **Training Data**: $D=(X,Y)$
- **Transparent Box Predictor**: $T$
- **Test Input**: $X'$

**Iris Datapoint**
- Petal width $> 1$ and Sepal width $< 1$
- **Explanation for $X'$**: Setosa
Interpretability Methods and their Outcomes

- Feature summary statistic: a summary statistic of how each feature affects the model predictions e.g. feature importance measures or statistics about the interaction strength between features
- Feature summary visualization: Some feature summaries can only be visualized and not meaningfully be placed in a table
- Model internals: Structure of the model used for explanation e.g. the learned weights in linear models, visualization of feature detectors that are learned in convolutional neural networks, etc.

Interpretability Methods and their Outcomes (contd.)

- Data point: methods that return data points (can be existing or newly created) to make a model interpretable e.g. To explain the prediction of a data point, find a similar data point by changing some of the features for which the predicted outcome changes in a relevant way (like a flip in the predicted class), or the identification of prototypes of predicted classes. Requirement - datapoints themselves should be interpretable.

- Intrinsically interpretable model: approximate black box (either globally or locally) with an interpretable model.
What Everyone Needs to Know about Interpretability in Machine Learning?

- Machine learning systems make predictions based on a set of input features (i.e. a bunch of numbers)
- Machine learning discovers correlations in data (but does not understand causality)
- Some models are special, but interpretability is not the norm
- The real question is: how was the system created?

Credits: Towards Data Science
Kevin and Himanshu
Explaining DL Models for NLP
Some models are easy for humans to interpret, but this is the exception more than the rule.
- Not all models can be represented in a way that is easy for humans to understand, at least not without some loss of fidelity.

More important question than how a model works, is why we ended up with that particular model.
- Ultimately depends on the training data that was used, how that data was represented, and the modeling decisions that were made.

When applying machine learning in social domain, especially important to think about the training data being used and any biases present in it.

Finally, remember that the vast majority of supervised machine learning models work by discovering **correlations** in the data.
- This should not be interpreted as imply any kind of causal connection between inputs and outputs.
Interacting with Models

- Differentiates between mental model and algorithmic model
  - Mental model: our internal decision making
  - Algorithmic model: an external model learned using machine learning
- Mental models can adjust according to current contextual factors etc.
- Algorithmic models restricted to the original feature set on which they were learned
- Interpretable models and methods attempting to break the barrier between mental models and algorithmic models

Credits: Shape of Data
Goals of Interpretability

If *I have an interpretable model, what should it allow me to do?*

1. Identify and mitigate biases
2. Account for context
3. Extract knowledge
4. Generalize
If I have an interpretable model, what should it allow me to do?

1. Identify and mitigate biases
   - All models are biased
   - Cannot eliminate biases completely
   - Reduce them
   - Identifying biases in mental models difficult, in black box models even harder
   - Can use self-reflection in case of mental models
   - Algorithmic models can learn on larger data, but cannot self-reflect
   - Goal - Right type of interpretability can allow practitioner to apply self-reflection to algorithmic models
Goals of Interpretability (contd)

If I have an interpretable model, what should it allow me to do?

2 Account for context
   - Algorithmic models cannot account for all the factors that will affect the decision
   - Goal - Interpretable model helps understand the included factors so that we can adjust prediction on additional factors
If I have an interpretable model, what should it allow me to do?

- Extract knowledge
  - Mental models - relatively simple causal relationships, augmented by flexible, subconscious intuition
  - Algorithmic models - probability distributions that measure correlations between rigidly defined values
  - Mental models can’t prevent themselves from trying to extract rules from some patterns
  - These patterns may not be real - especially if dataset is biased
  - Interpretable model should help identify if the patterns are really there or artifacts of a biased dataset
  - Goal - ability to combine strong pattern recognition and simplification skills of the human mind with the algorithmic model’s ability to learn from massive amounts of data to obtain knowledge
  - Extracted knowledge need not be only causality rules
Goals of Interpretability (contd.)

If I have an interpretable model, what should it allow me to do?

4 Generalize

- Algorithmic models trained for narrowly defined problems
- Mental models trained on variety of input and applied to vaguely defined problems that they were not trained for
- Same thing may work for some algorithmic models in some scenarios
- Goal: An interpretable model should help you determine if and how it can be generalized
Properties of Interpretability

Properties of explanations about individual predictions, without necessarily describing the process for calculating them

1. Concise
2. Faithful
3. Complete
4. Comparable
5. Global
6. Consistent
7. Engaging
Properties of explanations about individual predictions, without necessarily describing the process for calculating them

- **Concise**
  - An explanation should not overwhelm the user
  - Reason we use ML is to delegate the complexity which our mental models cannot handle
  - An explanation should minimize the cognitive load required of the user
Properties of explanations about individual predictions, without necessarily describing the process for calculating them

2 Faithful

- Explanation should accurately describe the way the model made the prediction
- Global surrogate models are not faithful
Properties of explanations about individual predictions, without necessarily describing the process for calculating them

3 Complete

- An explanation is complete if it explains all the factors and elements that went into a prediction
- Trade-off between concise and complete
Properties of explanations about individual predictions, without necessarily describing the process for calculating them

Comparable

- An explanation should help you compare different models to each other by examining how they handle individual examples
- Handled well by model-agnostic techniques, not so well by model-dependent techniques
Properties of explanations about individual predictions, without necessarily describing the process for calculating them

5 Global

- The explanation should indicate how each individual prediction fits into the overall structure of the model
- Understanding how the overall model works
- The global explanation only has to indicate enough of the model’s structure to provide reasonable context for each individual prediction
Properties of explanations about individual predictions, without necessarily describing the process for calculating them

Consistent

- An explanation algorithm is consistent if each successive explanation helps the user to better understand later predictions.
- The user should never perceive a contradiction between different explanations.
- Example: if a feature increases a risk prediction for one data point but decreases it for another data point, the explanations should include enough information for the user to understand why.
Properties of explanations about individual predictions, without necessarily describing the process for calculating them

- **Engaging**
  - An explanation should encourage a user to pay attention to the important details
  - Users who pay more attention to explanations become familiar with model faster and ultimately make better decisions
  - Seems like a User Experience (UX) issue
Interpretable WE
Black Box Nature of Word Embeddings

Classical data from machine learning (Iris dataset)

<table>
<thead>
<tr>
<th>flower id</th>
<th>sepal length (cm)</th>
<th>sepal width (cm)</th>
<th>petal length (cm)</th>
<th>petal width (cm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>5.1</td>
<td>3.5</td>
<td>1.4</td>
<td>0.2</td>
</tr>
<tr>
<td>1</td>
<td>4.9</td>
<td>3.0</td>
<td>1.4</td>
<td>0.2</td>
</tr>
<tr>
<td>2</td>
<td>4.7</td>
<td>3.2</td>
<td>1.3</td>
<td>0.2</td>
</tr>
<tr>
<td>3</td>
<td>4.6</td>
<td>3.1</td>
<td>1.5</td>
<td>0.2</td>
</tr>
<tr>
<td>4</td>
<td>5.0</td>
<td>3.6</td>
<td>1.4</td>
<td>0.2</td>
</tr>
</tbody>
</table>
Black Box Nature of Word Embeddings

- Classical data from machine learning (Iris dataset)

<table>
<thead>
<tr>
<th>flower id</th>
<th>sepal length (cm)</th>
<th>sepal width (cm)</th>
<th>petal length (cm)</th>
<th>petal width (cm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>5.1</td>
<td>3.5</td>
<td>1.4</td>
<td>0.2</td>
</tr>
<tr>
<td>1</td>
<td>4.9</td>
<td>3.0</td>
<td>1.4</td>
<td>0.2</td>
</tr>
<tr>
<td>2</td>
<td>4.7</td>
<td>3.2</td>
<td>1.3</td>
<td>0.2</td>
</tr>
<tr>
<td>3</td>
<td>4.6</td>
<td>3.1</td>
<td>1.5</td>
<td>0.2</td>
</tr>
<tr>
<td>4</td>
<td>5.0</td>
<td>3.6</td>
<td>1.4</td>
<td>0.2</td>
</tr>
</tbody>
</table>

- Word embeddings (Ordinary)

<table>
<thead>
<tr>
<th>word</th>
<th>dim 1</th>
<th>dim 2</th>
<th>dim 3</th>
<th>...</th>
<th>dim 98</th>
<th>dim 99</th>
<th>dim 100</th>
</tr>
</thead>
<tbody>
<tr>
<td>apple</td>
<td>0.15</td>
<td>0.29</td>
<td>-0.05</td>
<td>...</td>
<td>0.58</td>
<td>-0.30</td>
<td>-0.11</td>
</tr>
<tr>
<td>cat</td>
<td>-0.10</td>
<td>0.18</td>
<td>-0.41</td>
<td>...</td>
<td>0.07</td>
<td>-0.26</td>
<td>-0.16</td>
</tr>
<tr>
<td>chair</td>
<td>-0.17</td>
<td>0.16</td>
<td>-0.42</td>
<td>...</td>
<td>-0.06</td>
<td>-0.61</td>
<td>-0.22</td>
</tr>
<tr>
<td>dog</td>
<td>-0.12</td>
<td>0.10</td>
<td>-0.56</td>
<td>...</td>
<td>-0.01</td>
<td>-0.06</td>
<td>-0.46</td>
</tr>
<tr>
<td>furniture</td>
<td>0.21</td>
<td>-0.09</td>
<td>-0.53</td>
<td>...</td>
<td>0.08</td>
<td>-0.27</td>
<td>-0.34</td>
</tr>
<tr>
<td>orange</td>
<td>0.22</td>
<td>-0.28</td>
<td>-0.31</td>
<td>...</td>
<td>0.69</td>
<td>-0.08</td>
<td>0.65</td>
</tr>
</tbody>
</table>

- Individual dimensions do not capture any specific concept
### Interpretable Word Embeddings

<table>
<thead>
<tr>
<th>Word</th>
<th>Dim 28</th>
<th>Word</th>
<th>Dim 131</th>
<th>Word</th>
<th>Dim 272</th>
</tr>
</thead>
<tbody>
<tr>
<td>strawberries</td>
<td>0.961</td>
<td>elegantly</td>
<td>0.904</td>
<td>thigh</td>
<td>0.875</td>
</tr>
<tr>
<td>peaches</td>
<td>0.956</td>
<td>attractively</td>
<td>0.874</td>
<td>knee</td>
<td>0.872</td>
</tr>
<tr>
<td>oranges</td>
<td>0.954</td>
<td>beautifully</td>
<td>0.870</td>
<td>shoulder</td>
<td>0.866</td>
</tr>
<tr>
<td>pears</td>
<td>0.949</td>
<td>brilliantly</td>
<td>0.860</td>
<td>elbow</td>
<td>0.857</td>
</tr>
<tr>
<td>apples</td>
<td>0.949</td>
<td>exquisitely</td>
<td>0.859</td>
<td>wrist</td>
<td>0.853</td>
</tr>
<tr>
<td>blueberries</td>
<td>0.946</td>
<td>delicately</td>
<td>0.858</td>
<td>ankle</td>
<td>0.852</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>america</td>
<td>0.000</td>
<td>compiling</td>
<td>0.000</td>
<td>meticulously</td>
<td>0.000</td>
</tr>
<tr>
<td>shear</td>
<td>0.000</td>
<td>apples</td>
<td>0.000</td>
<td>umbrella</td>
<td>0.000</td>
</tr>
<tr>
<td>nuanced</td>
<td>0.000</td>
<td>propagate</td>
<td>0.000</td>
<td>amusing</td>
<td>0.000</td>
</tr>
<tr>
<td>koreans</td>
<td>0.000</td>
<td>autoantibody</td>
<td>0.000</td>
<td>plead</td>
<td>0.000</td>
</tr>
<tr>
<td>fy</td>
<td>0.000</td>
<td>niagara</td>
<td>0.000</td>
<td>cautionary</td>
<td>0.000</td>
</tr>
<tr>
<td>bandit</td>
<td>0.000</td>
<td>bandit</td>
<td>0.000</td>
<td>bandit</td>
<td>0.000</td>
</tr>
</tbody>
</table>

Examples from **NNSE** embeddings ([Murphy et al., 2012](#)). One can easily infer that the three dimensions correspond to the latent concepts *fruits*, *an adverb of manner*, and *part of a body*
Methods
Given the question *How many feet are there in a fathom?*

What is the expected type of answer from the set
- \{ABBREVIATION, DESCRIPTION, ENTITY, HUMAN, LOCATION, NUMBER\}? 

<table>
<thead>
<tr>
<th>Original Words</th>
<th>How</th>
<th>many</th>
<th>feet</th>
<th>are</th>
<th>there</th>
<th>in</th>
<th>a</th>
<th>fathom</th>
<th>?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Relevance Scores</td>
<td>1.4</td>
<td>1.4</td>
<td>1.9</td>
<td>0.33</td>
<td>0.22</td>
<td>-0.08</td>
<td>0.33</td>
<td>0.24</td>
<td>0</td>
</tr>
</tbody>
</table>

Example of relevance scores attributed to different words for question label classification.
LIME: Overview

- A model-agnostic technique proposed by Ribeiro et al. (2016)
  - Can be used to explain any models like Random Forests, SVMs, Neural Nets, etc.
- Learns the behavior by perturbing the input and observing the corresponding change in prediction.
- Perturbation at the input level enables practitioner to make sensible changes to the input.
  - for example, he/she can change words and not worry about what is happening after the embedding layer.
LIME: Overview

- Generates data points in the vicinity of current input, learns a linear classifier using this data, returns **weights of linear model** as **relevance scores** over input features.
LIME: Working

Image whose label has to be explained
LIME: Working (contd.)

Original Image
P(tree frog) = 0.54

Explaning the previous image
Using LIME

- Can be directly installed via `pip`
- Needs a model that can output probability distributions over classes
  - Easily usable with scikit-learn as well as modern neural network libraries
**LRP: Overview**

- A model-dependent technique proposed by Bach et al. (2015)
- Explains individual predictions by redistributing the final prediction output back in the network

Assigns relevance scores to each input variable
  - Input neurons that contribute the most to higher-layer get max relevance
LRP: Overview

Input → Forward Pass → Output

Heatmap → Relevance Propagation → Output
Output is considered as relevance of final layer: \( R_7^{(3)} = f(x) \)

Relevance conserved at each layer:
\[
R_4^{(2)} + R_5^{(2)} + R_6^{(2)} = R_1^{(1)} + R_2^{(1)} + R_3^{(1)}
\]

Relevance for layer \( l \) comes from the layer \( l + 1 \) via messages
**LRP: Working**

Two main equations

\[
R_i^{(l)} = \sum_{k: \text{is input for neuron } k} R_{i \leftarrow k}^{(l,l+1)} \tag{3}
\]

\[
R_{i \leftarrow k}^{(l,l+1)} = R_k^{(l+1)} \frac{a_i w_{ik}}{\sum_h a_h w_{hk}} \tag{4}
\]

Example:

\[
R_3^{(1)} = R_{3 \leftarrow 5}^{(1,2)} + R_{3 \leftarrow 6}^{(1,2)} \tag{5}
\]

\[
R_{3 \leftarrow 5}^{(1,2)} = R_5^{(2)} \frac{a_3 w_{35}}{\sum_h a_h w_{h5}} \tag{6}
\]
Model-dependence implies extra instrumentation needed in network code by practitioners.

Library specific packages released by developers.

However, it allows one to not only investigate relevance at input layers, but also at different intermediate layers.
Using LRP (contd.)

A Sample CNN Architecture
Using LRP (contd.)

1-gram Relevance Flow
Using LRP (contd.)

**2-gram Relevance Flow**
Using LRP (contd.)

Convolutions:
- Convolution: $C_1$  
  Filter Size: 1 x embed_dim
- Convolution: $C_2$  
  Filter Size: 2 x embed_dim
- Convolution: $C_3$  
  Filter Size: 3 x embed_dim
- Convolution: $C_4$  
  Filter Size: 4 x embed_dim

Max Pools:
- Max Pool: $P_1$  
  Size: 2x1
- Max Pool: $P_2$  
  Size: 2x1
- Max Pool: $P_3$  
  Size: 2x1
- Max Pool: $P_4$  
  Size: 2x1

ReLUs:
- ReLU: $Rel_1$
- ReLU: $Rel_2$
- ReLU: $Rel_3$
- ReLU: $Rel_4$

Reshapes:
- Reshape: $R_1$
- Reshape: $R_2$
- Reshape: $R_3$
- Reshape: $R_4$

Concat

Linear with ReLU activation

Linear with Softmax activation

3-gram Relevance Flow

how many feet are there in a fathom?

PAD

Embedding LookUp

how many feet are there in a fathom?
Using LRP (contd.)

4-gram Relevance Flow
LRP Usage for Question Classification

True Label: Desc, Pred Label: Desc
LRP Usage for Question Classification

True Label: Human, Pred Label: Human
LRP Usage for Sentiment Classification

True Label: Negative, Pred Label: Negative
LRP Usage for Sentiment Classification

Model 1 - True Label: Positive, Pred Label: Positive
LRP Usage for Sentiment Classification

Model 1 - True Label: Positive, Pred Label: Positive

Model 2 - True Label: Positive, Pred Label: Negative
LRP Usage for Sentiment Classification

Model 1 - True Label: Positive, Pred Label: Positive

Model 2 - True Label: Positive, Pred Label: Negative

Model 1 is better!
IG: Overview

- A model agnostic technique by Sundararajan et al. (2017)
- Generates explanation by comparing the current input against a baseline.
  - By accumulating gradients for each point on a path between current input and baseline
- They propose two axioms for interpretability: Sensitivity and Implementation Invariance
- Their proposed technique satisfies both of these axioms
Sensitivity

- A method satisfies Sensitivity axiom if for every input and baseline that differ in exactly one feature and having different predictions, the differing feature is given a non-zero attribution.
- If the function (implemented by our model) does not depend (mathematically) on some variable in the input, then the attribution to that variable should be always zero.
IG: Axioms (contd.)

- Implementation Invariance:
  - Two networks are functionally equivalent if their outputs are equal for all inputs, despite having very different structure.
  - A method satisfies Implementation Invariance when the attributions are always identical for two functionally equivalent networks.
IG: Working

- Generates explanation by comparing the current input against a baseline.
  - By accumulating gradients for each point on a path between current input and baseline
- Specifically it,
  - considers all points on a path between the current input and a baseline
  - computes the prediction at each point
  - computes the gradients for each point with respect to the baseline
  - returns the weighted sum of these gradients as relevance assignment over the input space.

\[
e_i = (x_i - x'_i) \times \sum_{k=1}^{m} \frac{\partial F(x' + \frac{k}{m}(x - x'))}{\partial x_i} \times \frac{1}{m}
\]  

(7)
No officially available code

However, the logic is simple and can be implemented easily in modern libraries

Also, logic depends only on input and output, so one need not do any network code instrumentation (unlike LRP)

This also prevents one from investigating different layers of the network

Model-agnosticness is a double edged sword!
List of Demos

- **LIME**
  - Explaining Document Classification by Random Forest
  - Explaining Document Classification by Neural Network

- **LRP**
  - Explaining Sentiment Classification by CNN
  - Explaining Addition of 2 numbers in a sequence by RNN
  - Explaining Part of Speech Tagging by LSTM

- **IG**
  - Explaining CIFAR image classification by CNN
  - Explaining Dummy Sentiment Classification by FFN
Conclusion

- Deep Learning definitely helpful in many problems
  - But one needs to tread carefully
- Model Explainability an ill defined problem
  - Different formulations exist out there
  - May choose a formulation that works for us and proceed
- Discussed and demonstrated a set of techniques - LIME, LRP and IG
  - Use them!
References I


Web References

- colah.github.io/posts/2015-08-Understanding-LSTMs
- www.wildml.com/2015/11/
  understanding-convolutional-neural-networks-for-nlp
- www.youtube.com/watch?v=KGOBB3wUbdc
- https://arxiv.org/abs/1510.03820
- https://arxiv.org/abs/1610.02583
- www.wildml.com/2015/10/
  recurrent-neural-networks-tutorial-part-3-backpropagation
We have strived to ensure that all references are properly included and all credits are properly given where due. However, in case we have missed some, please write to us at kevin.patel@cse.iitb.ac.in and we will add those in the online copy.