Analyzing Software using Deep Learning

Introduction (Part 3)

Prof. Dr. Michael Pradel
Software Lab, University of Stuttgart
Summer 2020
Overview

- **Motivation**
  - What the course is about
  - Why it is interesting
  - How it can help you

- **Organization**
  - Lectures and final exam
  - Course project

- **Basics**
  - Program analysis
  - Deep learning
Program Representations

Many ways to represent (parts of) a program

- Sequence of characters
- Sequence of tokens
- Abstract syntax tree
- Control flow graph
- Data dependence graph
- Call graph
- etc.
Program Representations

Many ways to represent (parts of) a program

- Sequence of characters
- Sequence of tokens
- Abstract syntax tree
- Control flow graph
- Data dependence graph
- Call graph
- etc.
Tokens

Tokenizer (or lexer)

- Part of compiler
- Splits sequence of characters into subsequences called tokens

E.g., for Java, six kinds of tokens:

- Identifiers, e.g., MyClass
- Keywords, e.g., if
- Separators, e.g., . or {
- Operators, e.g., * or ++
- Literals, e.g., 23 or "hi"
- Comments, e.g., /* bla */
Token: Example

```java
if (flag == true) {
    name = "Joe";
}
```

- Keyword
- Separators
- Identifier
- Operators
- Literals
Abstract Syntax Tree

- **Tree** representation of source code
- ”Abstract” because some details of syntax omitted
  - E.g., `{` in Java

- **Nodes**: Construct in source code
- **Edges**: Parent-child relationship

- Check out **Esprima** for obtaining ASTs of Javascript:
  [http://esprima.org/demo](http://esprima.org/demo)
Abstract Syntax Tree: Example

Example: javascript

var x = 6 * y;
Control Flow Graph

- Models flow of control through a program

Graph \((N, E)\) with

- Nodes \(N\): Basic blocks = Sequence of operations executed together
- Edges \(E\): Possible transfers of control

- Typically on the method-level
Control Flow Graph: Example

```javascript
if (c) {
    x = 5
} else {
    x = 7
}
console.log(x)
```
Data Dependence Graph

- Models flow of data from "definition" to "use"
- Graph \((N, E)\) with
  - Nodes \(N\): Operations that define and/or use data
  - Edges \(E\): Possible definition-use relationships
    - Edge \(e = (n_1, n_2)\) means \(n_2\) may use data defined at \(n_1\)
Data Dependence Graph: Example

\[ x = 3 \]
\[ y = 5 \]
\[ \text{if } (x \geq 1) \]
\[ y = x \]
\[ z = x + y \]
Deep Learning: Example

Example: Handwriting recognition

- Goal: Recognize digits 0..9
- Easy for a human but challenging for a computer
- Idea: Learn from a large number of training examples
- Deep learning: > 99% accuracy

Following slides based on Chapter 1 of neuralnetworksanddeeplearning.com
Network of neurons

input layer

e.g., pixels of an image

hidden layer

network

output layer

e.g., whether it's the digit 3
Perceptrons

1) Most basic kind of neurons
2) Binary inputs
3) Binary output

\[
\text{output} = \begin{cases} 
0 & \text{if } \sum_j w_j \cdot x_j \leq \text{threshold} \\
1 & \text{if } \sum_j w_j \cdot x_j > \text{threshold} 
\end{cases}
\]

\[
= \begin{cases} 
0 & \text{if } w \cdot x + b \leq 0 \\
1 & \text{if } w \cdot x + b > 0 
\end{cases}
\]

\[x_1 \xrightarrow{w_4} o \xrightarrow{w_5} \text{output} \]
\[x_2 \xrightarrow{w_6} \]
\[x_3 \xrightarrow{w_7} \]

w.. weights
b.. bias
Example

\(x_1 = \text{Weather is good}\)
\(x_2 = \text{Friends go}\)
\(x_3 = \text{Like cheese}\)

\(w_1 = 5\)
\(w_2 = 3\)
\(w_3 = 1\)

bias = -7

Output = Go to cheese festival

Assume: \(x_1 = 1, x_2 = 1, x_3 = 0\)

\(w \cdot x = 5 \cdot 1 + 3 \cdot 1 + 0 \cdot 1 = 8\)

\(\text{Output} = \begin{cases} 0 & \text{if } 8 - 7 \leq 0 \\ 1 & \text{if } 8 - 7 > 0 \end{cases} \rightarrow \text{Go to festival}\)
Computing Logical Functions

NAND gate

\[ x_1 \quad w_1 = -2 \quad \text{bias} = 3 \]

\[ x_2 \quad w_2 = -2 \]

output

<table>
<thead>
<tr>
<th>( x_1 )</th>
<th>( x_2 )</th>
<th>output</th>
<th>because ( 0 + 3 &gt; 0 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>
Universal Computation

- Networks of **NAND perceptrons** can simulate every circuit containing only NAND gates

- Can express **arbitrary computations**!
Example: Adding Two Bits

NAND gate:

\[ x_1, x_2 \rightarrow \text{sum: } x_1 \oplus x_2, \text{ carry bit: } x_1x_2 \]

Network of perceptrons:

\[ x_1, x_2 \rightarrow \text{sum: } x_1 \oplus x_2, \text{ carry bit: } x_1x_2 \]
Challenge: Set Weights and Biases

- More complex networks can perform arbitrary computations

- How to decide on the weights and biases?

- Option 1: Hand-tune them
  → Infeasible for complex networks

- Option 2: Learn them
  → Key idea behind machine learning with neural networks
Making learning possible

\[ w + \Delta w \]

\[ \text{output} + \Delta \text{output} \]

Want: Small change of weights & biases cause small change of output

Problem: Perceptron doesn't provide this property

\[ \text{output} = \text{step} (w \cdot x + b) \]
Sigmoid neuron

\[
\begin{align*}
\text{output} &= \sigma (w \cdot x + b) \\
\text{sigmoid function: } \sigma(z) &= \frac{1}{1+e^{-z}} = \frac{1}{1 + \exp(-z)} = \frac{1}{1 + \exp(-\sum w_j x_j + b)}
\end{align*}
\]

→ Enables learning: Small change causes small change
Activation Functions

- step function
- sigmoid function
- logistic function
- identity function
- rectified linear unit
Learning: Cost Function

Cost function: feedback on how good the output is for given input

Example:

\[ C = \frac{1}{n} \sum_{x} \| y(x) - a \|^2 \]

nb. of training inputs

If digit is known is 6, want output:
\[ y(x) = (0, 0, 0, 0, 0, 0, 1, 0, 0, 0) \]

Actual output may be:
\[ a = (0, 0, 0, 0.2, 0, 0, 0.7, 0.7, 0, 0) \]

quadratic cost fct.
or
mean squared error.
Quiz: Cost Function

- Recognition of hand-written digits
- Only digits 0, 1, and 2
- Training examples:

<table>
<thead>
<tr>
<th>Example</th>
<th>Actual</th>
<th>Desired</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(0, 1, 0)ᵀ</td>
<td>(0.5, 0.5, 0)ᵀ</td>
</tr>
<tr>
<td>2</td>
<td>(1, 0, 0)ᵀ</td>
<td>(1, 0, 0)ᵀ</td>
</tr>
</tbody>
</table>

- What is the value of the cost function?
\[ C = \frac{1}{n} \cdot \sum_{x} \| y(x) - a \|^{2} \]

\[ = \frac{1}{2} \cdot \left( \| -0.5, 0.5, 0 \|^{2} + \| (0, 0, 0) \|^{2} \right) \]

\[ = \frac{1}{2} \cdot (0.5 + 0) = 0.25 \]
Goal: Minimize Cost Function

- **Goal of learning:** Find weights and biases that **minimize the cost function**

- **Approach:** **Gradient descent**
  - Compute **gradient** of $C$: Vector of partial derivatives
  - "Move" closer toward minimum step-by-step
  - **Learning rate** determines step size
Training Examples

- Effort of computing gradient depends on number of examples
- **Stochastic gradient descent**
  - Use small sample of all examples
  - Compute estimate of true gradient
- **Epochs and mini-batches**
  - Split training examples into $k$ mini-batches
  - Train network with each mini-batch
  - Epoch: Each mini-batch used exactly once