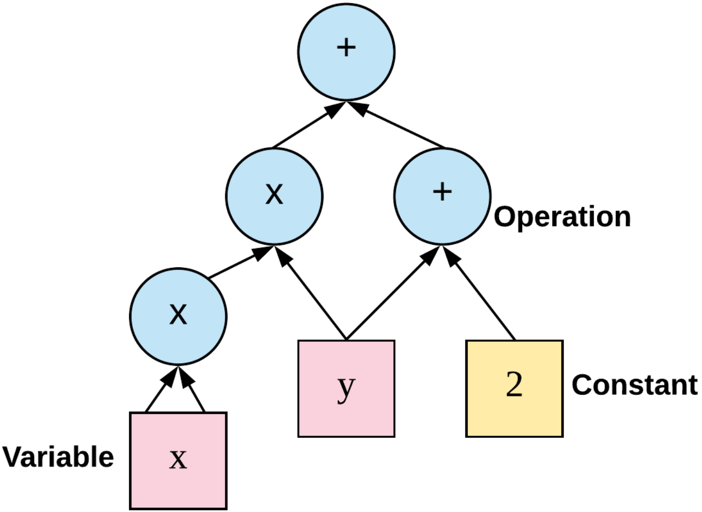
Deep Learning

Vazgen Mikayelyan

YSU, Krisp

September 30, 2020

Computational Graphs



- Nodes are operators (ops), variables and constants.

Computational Graphs

- Nodes are operators (ops), variables and constants.
- Edges are tensors
  - 0-d is a scalar
  - 1-d is a vector
  - 2-d is a matrix
  - Etc.

Computational Graphs

- Nodes are operators (ops), variables and constants.
- Edges are tensors
    - 0-d is a scalar
    - 1-d is a vector
    - 2-d is a matrix
    - Etc.
- Constants are fixed value tensors - not trainable.

Computational Graphs

- Nodes are operators (ops), variables and constants.
- Edges are tensors
  - 0-d is a scalar
  - 1-d is a vector
  - 2-d is a matrix
  - Etc.
- Constants are fixed value tensors - not trainable.
- Variables are tensors initialized in a session - trainable / not trainable.

Computational Graphs

- Nodes are operators (ops), variables and constants.
- Edges are tensors
    - 0-d is a scalar
    - 1-d is a vector
    - 2-d is a matrix
    - Etc.
- Constants are fixed value tensors - not trainable.
- Variables are tensors initialized in a session - trainable / not trainable.
- Placeholders are tensors of values that are unknown during the graph construction, but passed as input during a session.

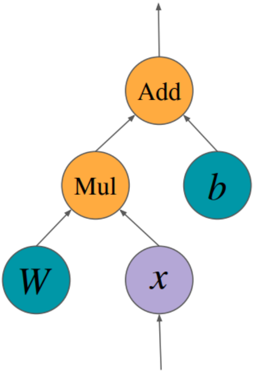
Computational Graphs

- Nodes are operators (ops), variables and constants.
- Edges are tensors
  - 0-d is a scalar
  - 1-d is a vector
  - 2-d is a matrix
  - Etc.
- Constants are fixed value tensors - not trainable.
- Variables are tensors initialized in a session - trainable / not trainable.
- Placeholders are tensors of values that are unknown during the graph construction, but passed as input during a session.
- Ops are functions on tensors.

Computational Graphs

$$\hat{y} = Wx + b$$

variable ops placeholder variable



Sessions

- Session is the runtime environment of a graph, where operations are executed and tensors are evaluated.

Sessions

- Session is the runtime environment of a graph, where operations are executed and tensors are evaluated.

```
>>> import tensorflow as tf
>>> a = tf.constant(1)
>>> b = tf.constant(2)
>>> op1 = tf.add(a, b)
>>> print(a)
Tensor("Const:0", shape=(), dtype=int32)
>>> print(b)
Tensor("Const_1:0", shape=(), dtype=int32)
>>> print(op1)
Tensor("Add:0", shape=(), dtype=int32)
```

- Session is the runtime environment of a graph, where operations are executed and tensors are evaluated.

```
>>> import tensorflow as tf
>>> a = tf.constant(1)
>>> b = tf.constant(2)
>>> op1 = tf.add(a, b)
>>> print(a)
Tensor("Const:0", shape=(), dtype=int32)
>>> print(b)
Tensor("Const_1:0", shape=(), dtype=int32)
>>> print(op1)
Tensor("Add:0", shape=(), dtype=int32)
```

```
>>> import tensorflow as tf
>>> a = tf.constant(1)
>>> b = tf.constant(2)
>>> c = tf.add(a, b)
>>> with tf.Session() as sess:
...     print(sess.run(a))
...     print(sess.run(b))
...     print(sess.run(c))
...
1
2
3
```

- Session is the runtime environment of a graph, where operations are executed and tensors are evaluated.

```
>>> import tensorflow as tf
>>> a = tf.constant(1)
>>> b = tf.constant(2)
>>> op1 = tf.add(a, b)
>>> print(a)
Tensor("Const:0", shape=(), dtype=int32)
>>> print(b)
Tensor("Const_1:0", shape=(), dtype=int32)
>>> print(op1)
Tensor("Add:0", shape=(), dtype=int32)
```

```
>>> import tensorflow as tf
>>> a = tf.constant(1)
>>> b = tf.constant(2)
>>> c = tf.add(a, b)
>>> with tf.Session() as sess:
...     print(sess.run(a))
...     print(sess.run(b))
...     print(sess.run(c))
...
1
2
3
```

- a.eval() is equivalent to session.run(a), but in general, "eval" is limited to executions of a single op and ops that returns a value.

Sessions

- Session is the runtime environment of a graph, where operations are executed and tensors are evaluated.

```
>>> import tensorflow as tf
>>> a = tf.constant(1)
>>> b = tf.constant(2)
>>> op1 = tf.add(a, b)
>>> print(a)
Tensor("Const:0", shape=(), dtype=int32)
>>> print(b)
Tensor("Const_1:0", shape=(), dtype=int32)
>>> print(op1)
Tensor("Add:0", shape=(), dtype=int32)
```

```
>>> import tensorflow as tf
>>> a = tf.constant(1)
>>> b = tf.constant(2)
>>> c = tf.add(a, b)
>>> with tf.Session() as sess:
...     print(sess.run(a))
...     print(sess.run(b))
...     print(sess.run(c))
...
1
2
3
```

- a.eval() is equivalent to session.run(a), but in general, "eval" is limited to executions of a single op and ops that returns a value.
- Upon op execution, only the subgraph (required for calculating its value) is evaluated

Structure of Training Code

1. Assembling the graph
    - Create placeholders.

Structure of Training Code

1. Assembling the graph
   - Create placeholders.
   - Create variables / neural network.

Structure of Training Code

1. Assembling the graph
   - Create placeholders.
   - Create variables / neural network.
   - Define a loss function.

Structure of Training Code

1. Assembling the graph
    - Create placeholders.
    - Create variables / neural network.
    - Define a loss function.
    - Define an optimizer.

Structure of Training Code

1. Assembling the graph
   - Create placeholders.
   - Create variables / neural network.
   - Define a loss function.
   - Define an optimizer.
2. Training in a session
   - Start a session.

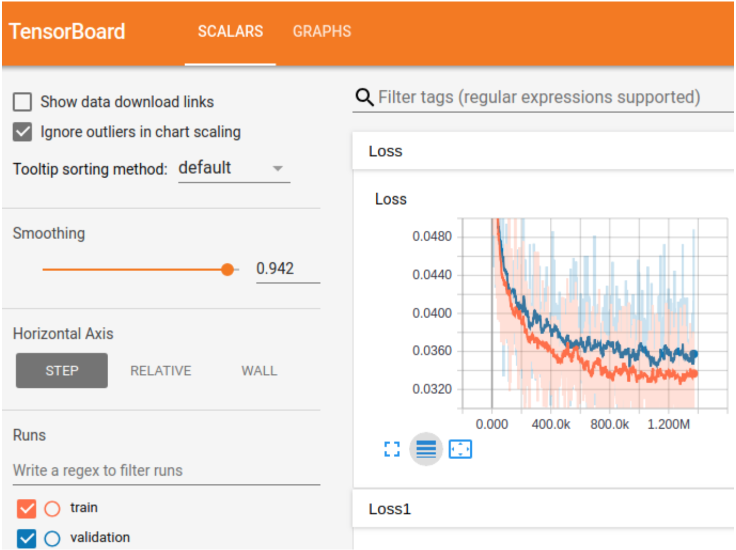
Structure of Training Code

1. Assembling the graph
   - Create placeholders.
   - Create variables / neural network.
   - Define a loss function.
   - Define an optimizer.
2. Training in a session
   - Start a session.
   - Initialize variables / restore from checkpoint.

Structure of Training Code

1. Assembling the graph
    - Create placeholders.
    - Create variables / neural network.
    - Define a loss function.
    - Define an optimizer.
2. Training in a session
    - Start a session.
    - Initialize variables / restore from checkpoint.
    - Run the optimizer over batches.

Tensorboard



Outline

1 Back-Propagation

Question: How to calculate the derivative of the function $\sin x^2$?

Back-Propagation

Question: How to calculate the derivative of the function $\sin x^2$?

Theorem 1

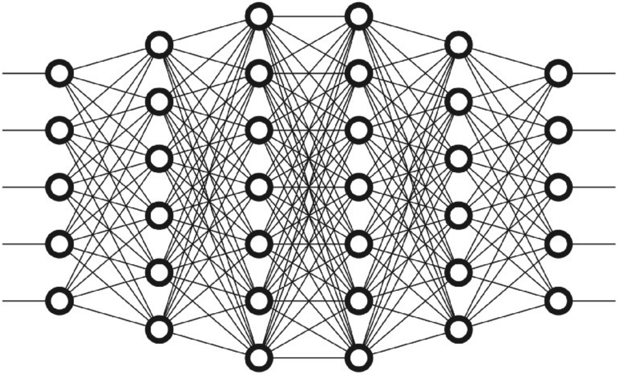
Given n functions $f_1, \dots, f_n$ with the composite function

$$f = f_1 \circ (f_2 \circ \cdots (f_{n-1} \circ f_n)),$$

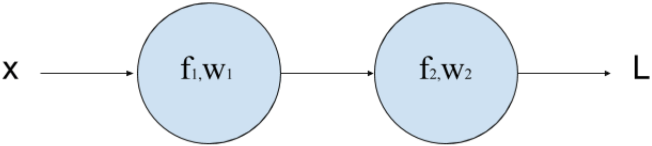
if each function f_i is differentiable at its immediate input, then the composite function is also differentiable by the repeated application of Chain Rule, where the derivative is

$$\frac{df}{dx} = \frac{df_1}{df_2} \frac{df_2}{df_3} \cdots \frac{df_n}{dx}.$$

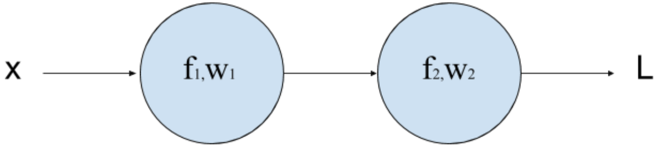
Back-Propagation



Back-Propagation

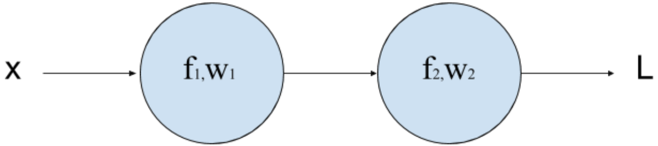


Back-Propagation



In this case we have the following function

$$L(w_1, w_2) = \left(f_2\left(w_2 f_1\left(w_1 x\right)\right) - y\right)^2$$

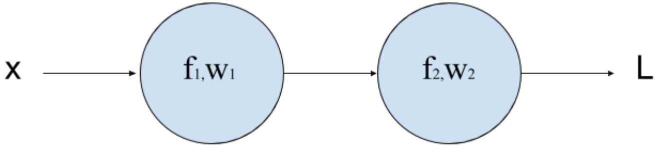


In this case we have the following function

$$L\left(w_1, w_2\right) = \left(f_2\left(w_2 f_1\left(w_1 x\right)\right) - y\right)^2$$

We have to calculate the derivatives $\dfrac{\partial L}{\partial w_1}$ and $\dfrac{\partial L}{\partial w_2}$:

Back-Propagation

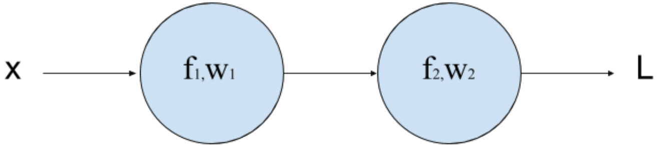


In this case we have the following function

$$L(w_1, w_2) = (f_2(w_2 f_1(w_1 x)) - y)^2$$

We have to calculate the derivatives $\frac{\partial L}{\partial w_1}$ and $\frac{\partial L}{\partial w_2}$:

$$\frac{\partial L}{\partial w_2} = \frac{\partial L}{\partial f_2} \frac{\partial f_2}{\partial (w_2 f_1)} \frac{\partial (w_2 f_1)}{\partial w_2},$$



In this case we have the following function

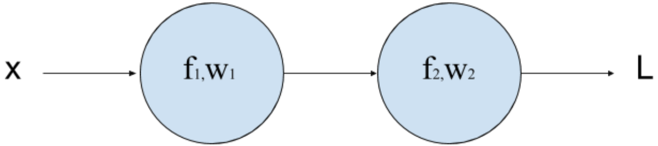
$$L\left(w_1, w_2\right) = \left(f_2\left(w_2 f_1\left(w_1 x\right)\right) - y\right)^2$$

We have to calculate the derivatives $\frac{\partial L}{\partial w_1}$ and $\frac{\partial L}{\partial w_2}$:

$$\frac{\partial L}{\partial w_2} = \frac{\partial L}{\partial f_2} \frac{\partial f_2}{\partial\left(w_2 f_1\right)} \frac{\partial\left(w_2 f_1\right)}{\partial w_2},$$

$$\frac{\partial L}{\partial w_1} =$$

Back-Propagation



In this case we have the following function

$$L(w_1, w_2) = (f_2(w_2 f_1(w_1 x)) - y)^2$$

We have to calculate the derivatives $\dfrac{\partial L}{\partial w_1}$ and $\dfrac{\partial L}{\partial w_2}$:

$$\frac{\partial L}{\partial w_2} = \frac{\partial L}{\partial f_2} \frac{\partial f_2}{\partial (w_2 f_1)} \frac{\partial (w_2 f_1)}{\partial w_2},$$

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial f_2} \frac{\partial f_2}{\partial (w_2 f_1)} \frac{\partial (w_2 f_1)}{\partial f_1} \frac{\partial (f_1)}{\partial (w_1 x)} \frac{\partial (w_1 x)}{\partial w_1}.$$