Topics in AI (CPSC 532S): Multimodal Learning with Vision, Language and Sound
Course Logistics

- Update on **course registrations** — 39 students registered!

  11 moved from waitlist, 15 still on the waitlist

- [Piazza](piazza.com/ubc.ca/winterter12022/cpsc532s) — 29 students signed up so far

- Assignment 0 is out (for practice only, no credit)

- Assignment 1 will be out later today (due in 1 week)

- My and TA of **finance** hours will be posted by today (mine are 12:30-1:30 pm today)
Course Logistics

- Update on course registrations — 39 students registered!
  11 moved from waitlist, 15 still on the waitlist

- Piazza — 29 students signed up so far
  - piazza.com/ubc.ca/winterter12022/cpsc532s
  - Access code: McGurk
- Update on **course registrations** — 39 students registered!
  
11 moved from waitlist, 15 still on the waitlist

- **Piazza** — 29 students signed up so far
  
  - piazza.com/ubc.ca/winterter12022/cpsc532s

  - Access code: **McGurk**

- **Assignment 0** is out (for practice only, no credit)

- **Assignment 1** will be out later today (due in 1 week)
Course Logistics

- Update on course registrations — 39 students registered!

  11 moved from waitlist, 15 still on the waitlist

- Piazza — 29 students signed up so far
  - piazza.com/ubc.ca/winterter12022/cpsc532s
  - Access code: McGurk

- Assignment 0 is out (for practice only, no credit)

- Assignment 1 will be out later today, at least in part (due in 1 week)

- Mine and TA office hours will be posted today (mine are 12:30-1:30 pm)
Topics in AI (CPSC 532S): Multimodal Learning with Vision, Language and Sound

Lecture 1: Introduction
Grading Criteria

- **Assignments** (programming) — 40% (total)
- Research papers — 20%
- **Project** — 40%
Grading Criteria

- **Assignments** (programming) — 40% (total)
- **Research papers** — 20%
- **Project** — 40%

**NO LATE SUBMISSIONS** — If you don’t complete the assignment, hand in what you have
Assignments (5 assignments and 40% of grade total)

- Assignment 0: Introduction to PyTorch (0%)
- Assignment 1: Neural Network Introduction (5%) — 🐍 python™

Assignments all use Python Jupiter Notebooks, use Canvas to hand everything in. Assignments always due at 11:59pm PST on due date.
Assignments (5 assignments and 40% of grade total)

- Assignment 0: Introduction to PyTorch (0%)
- Assignment 1: Neural Network Introduction (5%) — 🐍 python™
- Assignment 2: Convolutional Neural Networks (5%) — PyTorch

Assignments all use Python Jupiter Notebooks, use Canvas to hand everything in. Assignments always due at 11:59pm PST on due date.
Assignments (5 assignments and 40% of grade total)

• Assignment 0: **Introduction to PyTorch** (0%)

• Assignment 1: **Neural Network Introduction** (5%) — 🐍 *Python™*

• Assignment 2: **Convolutional Neural Networks** (5%) — PyTorch

• Assignment 3: RNN **Language Modeling and Translation** (10%) — PyTorch

Assignments all use **Python Jupiter Notebooks**, use **Canvas** to hand everything in. Assignments always due at **11:59pm PST** on due date.
Assignments (5 assignments and 40% of grade total)

- Assignment 0: **Introduction to PyTorch** (0%)
- Assignment 1: **Neural Network Introduction** (5%) — 🐍 python
- Assignment 2: **Convolutional Neural Networks** (5%) — PyTorch
- Assignment 3: **RNN Language Modeling and Translation** (10%) — PyTorch
- Assignment 4: **Neural Model for Image Captioning / Retrieval** (10%) — PyTorch

Assignments all use **Python Jupiter Notebooks**, use **Canvas** to hand everything in. Assignments always due at **11:59pm PST** on due date.
Assignments (5 assignments and 40% of grade total)

- Assignment 0: **Introduction to PyTorch** (0%)
- Assignment 1: **Neural Network Introduction** (5%) — ![Python](python.png)
- Assignment 2: **Convolutional Neural Networks** (5%) — ![PyTorch](pytorch.png)
- Assignment 3: RNN **Language Modeling and Translation** (10%) — ![PyTorch](pytorch.png)
- Assignment 4: Neural Model for **Image Captioning / Retrieval** (10%) — ![PyTorch](pytorch.png)
- Assignment 5: Advanced Architectures **Graph NN and GANs** (10%) — ![PyTorch](pytorch.png)

Assignments all use Python Jupiter Notebooks, use Canvas to hand everything in. Assignments always due at 11:59pm PST on due date.
I reserve the right to change release and due dates for the assignments to accommodate constraints of the course, do not take the dates on web-page as “set in stone”.

Assignments (5 assignments and 40% of grade total)
Research Papers (reviews and presentation, 20% of grade total)

Presentation - 10%

• You will need to present 1 paper individually or as a group (group size will be determined by # of people in class)

• Pick a paper from the syllabus individually (we will have process to pick #1, #2, #3 choices)

• Will need to prepare slides and meet with me or TA for feedback

• It is your responsibility to schedule these meetings

• I will ask you to record these presentation and we will make these available
Research Papers (reviews and presentation, 20% of grade total)

Presentation - 10%

- You will need to **present 1 paper** individually or as a group (group size will be determined by # of people in class)
- Pick a paper from the syllabus individually (we will have process to pick #1, #2, #3 choices)
- Will need to prepare slides and **meet with me or TA** for feedback
- It is your responsibility to schedule these meetings
- I will ask you to **record** these presentation and we will make these available

Reading Reviews - 10%

- Individually, one for most lectures after the first half of semester
- Due 11:59pm a day before class where reading assigned, submitted via Canvas
Good **Presentation**

- You are effectively taking on responsibility for being an instructor for part of the class (*take it seriously*)

- What makes a **good presentation**?
  - High-level overview of the problem and motivation
  - Clear statement of the problem
  - Overview of the technical details of the method, including necessary background
  - Relationship of the approach and method to others discussed in class
  - Discussion of strengths and weaknesses of the approach
  - Discussion of strengths and weaknesses of the evaluation
  - Discussion of potential extensions (published or potential)
Reading **Reviews**

- Designed to make sure you read the material and have thought about it prior to class (to stimulate discussion)

  - Short summary of the paper (3-4 sentences)
  - Main contributions (2-3 bullet points)
  - Positive / negative points (2-3 bullet points each)
  - What did you not understand (was unclear) about the paper (2-3 bullet points)
Final **Project** (40% of grade total)

- Group project (groups of 3 are encouraged, but fewer maybe possible)
- Groups are self-formed, you will not be assigned to a group
- You need to come up with a project proposal and then work on the project as a group (each person in the group gets the same grade for the project)
- Project needs to be **research** oriented (not simply implementing an existing paper); you can use code of existing paper as a starting point though

Project proposal + class presentation: 15%
Project + final presentation (during finals week): 25%
Sample **Project Ideas**

- Translate an image into a cartoon or Picasso drawing better than existing approaches (e.g., experiment with loss functions, architectures)
- Generating video clips by retrieving images relevant to lyrics of songs
- Generating an image based on the sounds or linguistic description
- Compare different feature representation and role of visual attention in visual question answering
- Storyboarding movie scripts
- Grounding a language/sound in an image

... there are endless possibilities ... think creatively and have fun!
Topics in AI (CPSC 532S):
Multimodal Learning with Vision, Language and Sound

Lecture 2: Introduction to Deep Learning
Introduction to **Deep Learning**

There is a **lot packed** into today’s lecture (excerpts from a few lectures of CS231n)

Covering: foundations and most important aspects of DNNs

**Not-covering:** neuroscience background of deep learning, optimization (CPSC 340 & CPSC 540), and not a lot of theoretical underpinning

if you want more details, check out CS231n lectures on-line
## Linear regression (review)

### Inputs (features)

<table>
<thead>
<tr>
<th></th>
<th>$x_1^{(1)}$</th>
<th>$x_2^{(1)}$</th>
<th>$x_3^{(1)}$</th>
<th>$x_4^{(1)}$</th>
<th>$x_5^{(1)}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>production costs</td>
<td>$x_1^{(2)}$</td>
<td>$x_2^{(2)}$</td>
<td>$x_3^{(2)}$</td>
<td>$x_4^{(2)}$</td>
<td>$x_5^{(2)}$</td>
</tr>
<tr>
<td>promotional costs</td>
<td>$x_1^{(3)}$</td>
<td>$x_2^{(3)}$</td>
<td>$x_3^{(3)}$</td>
<td>$x_4^{(3)}$</td>
<td>$x_5^{(3)}$</td>
</tr>
<tr>
<td>genre of the movie</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>box office first week</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>total book sales</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Outputs

<table>
<thead>
<tr>
<th></th>
<th>$y_1^{(1)}$</th>
<th>$y_2^{(1)}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>total revenue USA</td>
<td>$y_1^{(2)}$</td>
<td>$y_2^{(2)}$</td>
</tr>
<tr>
<td>total revenue USA</td>
<td>$y_1^{(3)}$</td>
<td>$y_2^{(3)}$</td>
</tr>
</tbody>
</table>

*slide adopted from V. Ordonex*
Linear *regression* (review)

<table>
<thead>
<tr>
<th>Inputs (features)</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>production costs</td>
<td>$y^{(1)}_1$</td>
</tr>
<tr>
<td>promotional costs</td>
<td>$y^{(1)}_2$</td>
</tr>
<tr>
<td>genre of the movie</td>
<td>$y^{(2)}_1$</td>
</tr>
<tr>
<td>box office first week</td>
<td>$y^{(2)}_2$</td>
</tr>
<tr>
<td>total book sales</td>
<td>$y^{(3)}_1$</td>
</tr>
<tr>
<td>total revenue USA</td>
<td>$y^{(3)}_2$</td>
</tr>
<tr>
<td>total revenue international</td>
<td>$y^{(4)}_1$</td>
</tr>
<tr>
<td>total book sales</td>
<td>$y^{(4)}_2$</td>
</tr>
<tr>
<td>total revenue USA</td>
<td>$y^{(5)}_1$</td>
</tr>
<tr>
<td>total revenue international</td>
<td>$y^{(5)}_2$</td>
</tr>
</tbody>
</table>

*slide adopted from V. Ordonex*
**Linear regression (review)**

Inputs (features):

<table>
<thead>
<tr>
<th></th>
<th>production costs</th>
<th>promotional costs</th>
<th>genre of the movie</th>
<th>box office first week</th>
<th>total book sales</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$x_1^{(1)}$</td>
<td>$x_2^{(1)}$</td>
<td>$x_3^{(1)}$</td>
<td>$x_4^{(1)}$</td>
<td>$x_5^{(1)}$</td>
</tr>
<tr>
<td></td>
<td>$x_1^{(2)}$</td>
<td>$x_2^{(2)}$</td>
<td>$x_3^{(2)}$</td>
<td>$x_4^{(2)}$</td>
<td>$x_5^{(2)}$</td>
</tr>
<tr>
<td></td>
<td>$x_1^{(3)}$</td>
<td>$x_2^{(3)}$</td>
<td>$x_3^{(3)}$</td>
<td>$x_4^{(3)}$</td>
<td>$x_5^{(3)}$</td>
</tr>
<tr>
<td></td>
<td>$x_1^{(4)}$</td>
<td>$x_2^{(4)}$</td>
<td>$x_3^{(4)}$</td>
<td>$x_4^{(4)}$</td>
<td>$x_5^{(4)}$</td>
</tr>
<tr>
<td></td>
<td>$x_1^{(5)}$</td>
<td>$x_2^{(5)}$</td>
<td>$x_3^{(5)}$</td>
<td>$x_4^{(5)}$</td>
<td>$x_5^{(5)}$</td>
</tr>
</tbody>
</table>

Outputs:

<table>
<thead>
<tr>
<th></th>
<th>total revenue USA</th>
<th>total revenue international</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$y_1^{(1)}$</td>
<td>$y_2^{(1)}$</td>
</tr>
<tr>
<td></td>
<td>$y_1^{(2)}$</td>
<td>$y_2^{(2)}$</td>
</tr>
<tr>
<td></td>
<td>$y_1^{(3)}$</td>
<td>$y_2^{(3)}$</td>
</tr>
</tbody>
</table>

$$\hat{y}_j = \sum_i w_{ji} x_i + b_j$$

*slide adopted from V. Ordonex*
Linear regression (review)

\[ \hat{y}_j = \sum_i w_{ji} x_i + b_j \]

each output is a linear combination of inputs plus bias, easier to write in matrix form:

\[ \hat{y} = W^T x + b \]

*slide adopted from V. Ordonex*
Linear regression (review)

\[ \hat{y}_j = \sum_i w_{ji} x_i + b_j \]

each output is a linear combination of inputs plus bias, easier to write in **matrix form**:

\[ \hat{y} = W^T x + b \]

Key to accurate prediction is **learning parameters** to minimize discrepancy with historical data

\[ D_{train} = \{(x^{(d)}, y^{(d)})\} \]

*slide adopted from V. Ordonex*
Linear regression (review)

\[ \hat{y}_j = \sum_i w_{ji} x_i + b_j \]

each output is a linear combination of inputs plus bias, easier to write in matrix form:

\[ \hat{y} = W^T x + b \]

Key to accurate prediction is learning parameters to minimize discrepancy with historical data

\[ D_{train} = \{(x^{(d)}, y^{(d)})\} \]

\[ \mathcal{L}(W, b) = \sum_{d=1}^{\left| D_{train}\right|} l(\hat{y}^{(d)}, y^{(d)}) \]

\[ W^*, b^* = \arg \min \mathcal{L}(W, b) \]

*slide adopted from V. Ordonex
Linear regression (review)

\[ \hat{y}_j = \sum_i w_{ji} x_i + b_j \]

each output is a linear combination of inputs plus bias, easier to write in matrix form:

\[ \hat{y} = W^T x + b \]

Key to accurate prediction is learning parameters to minimize discrepancy with historical data

\[ D_{train} = \{(x^{(d)}, y^{(d)})\} \]

\[ \mathcal{L}(W, b) = \sum_{d=1}^{D_{train}} ||\hat{y}^{(d)} - y^{(d)}||^2 \]

\[ W^*, b^* = \arg \min \mathcal{L}(W, b) \]

*slide adopted from V. Ordonex
Linear **regression** (review) — Learning \( w \) Least Squares

\[
\mathcal{L}(W, b) = \sum_{d=1}^{|D_{\text{train}}|} \left\| W^T x^{(d)} + b - y^{(d)} \right\|^2
\]

\[
W^*, b^* = \arg \min \mathcal{L}(W, b)
\]

Solution:

*slide adopted from V. Ordonex*
Linear regression (review) — Learning /w Least Squares

\[ L(W, b) = \sum_{d=1}^{|D_{\text{train}}|} \left| \left| W^T x^{(d)} + b - y^{(d)} \right| \right|^2 \]

\[ W^*, b^* = \arg \min L(W, b) \]

Solution:

\[ \frac{\partial L(W, b)}{\partial w_{ji}} = \frac{\partial}{\partial w_{ji}} \sum_{d=1}^{|D_{\text{train}}|} \left| \left| W^T x^{(d)} + b - y^{(d)} \right| \right|^2 \]

*slide adopted from V. Ordonex*
Linear regression (review) — Learning /w Least Squares

\[ L(W, b) = \sum_{d=1}^{\left| D_{\text{train}} \right|} \left\| W^T x^{(d)} + b - y^{(d)} \right\|^2 \]

\[ W^*, b^* = \arg \min L(W, b) \]

Solution:

\[ \frac{\partial L(W, b)}{\partial w_{ji}} = \frac{\partial}{\partial w_{ji}} \left( \sum_{d=1}^{\left| D_{\text{train}} \right|} \left\| W^T x^{(d)} + b - y^{(d)} \right\|^2 \right) \]

\[ \frac{\partial L(W, b)}{\partial w_{ji}} = \frac{\partial}{\partial w_{ji}} \sum_{d=1}^{\left| D_{\text{train}} \right|} \left\| W^T x^{(d)} + b - y^{(d)} \right\|^2 = 0 \]

*slide adopted from V. Ordonex*
Linear regression (review) — Learning \( \text{w} \) Least Squares

\[
L(W, b) = \sum_{d=1}^{|D_{\text{train}}|} \left\| W^T x^{(d)} + b - y^{(d)} \right\|^2
\]

\[W^*, b^* = \arg \min L(W, b)\]

Solution:

\[
\frac{\partial L(W, b)}{\partial w_{ji}} = \frac{\partial}{\partial w_{ji}} \sum_{d=1}^{|D_{\text{train}}|} \left\| W^T x^{(d)} + b - y^{(d)} \right\|^2
\]

\[
\frac{\partial L(W, b)}{\partial w_{ji}} = \frac{\partial}{\partial w_{ji}} \sum_{d=1}^{|D_{\text{train}}|} \left\| W^T x^{(d)} + b - y^{(d)} \right\|^2 = 0
\]

after some operations \( W^* = (X^TX)^{-1}X^TY \)

*slide adopted from V. Ordonex
One-layer Neural Network

Input Layer

Output Layer

Weighted Sum

Activation Function

\[ a(x) = x \]

Linear Activation

\[ a(x) = x \]
One-layer Neural Network

Input Layer

Multi-layer Perceptron Layer (MLP) / Fully Connected (FC) Layer

Weighted Sum → Activation Function

\[ y_1 = a(\sum x_i w_i + b) \]
\[ y_2 = a(\sum x_i w_i + b) \]

\( W_o, b_o \)

Fully Connected (FC) Layer = Activation Function (Linear Layer)
One-layer **Neural Network**

Input Layer

Multi-layer Perceptron Layer (MLP) / Fully Connected (FC) Layer
Multi-layer Neural Network

Input Layer

$\mathbf{x}_1$

$\mathbf{x}_2$

$\mathbf{x}_3$

$\mathbf{x}_4$

$\mathbf{x}_5$

1st Hidden Layer

$\mathbf{W}_{h1}, \mathbf{b}_{h1}$

2nd Hidden Layer

$\mathbf{W}_{h2}, \mathbf{b}_{h2}$

Output Layer

$\mathbf{y}_1$

$\mathbf{y}_2$

$\mathbf{W}_{o}, \mathbf{b}_{o}$
**Question:** What is a Neural Network?

**Answer:** Complex mapping from an input (vector) to an output (vector)
Question: What is a Neural Network?
Answer: Complex mapping from an input (vector) to an output (vector)

Question: What class of functions should be considered for this mapping?
Answer: Compositions of simpler functions (a.k.a. layers)? We will talk more about what specific functions next …

* slide from Marc’Aurelio Renzato
Neural Network Intuition

**Question:** What is a Neural Network?
**Answer:** Complex mapping from an input (vector) to an output (vector)

**Question:** What class of functions should be considered for this mapping?
**Answer:** Compositions of simpler functions (a.k.a. layers)? We will talk more about what specific functions next …

**Question:** What does a hidden unit do?
**Answer:** It can be thought of as classifier or a feature.

* slide from Marc’Aurelio Renzato
Neural Network **Intuition**

**Question:** What is a Neural Network?

**Answer:** Complex mapping from an input (vector) to an output (vector).

**Question:** What class of functions should be considered for this mapping?

**Answer:** Compositions of simpler functions (a.k.a. layers)? We will talk more about what specific functions next ...

**Question:** What does a hidden unit do?

**Answer:** It can be thought of as classifier or a feature.

**Question:** Why have many layers?

**Answer:**
1) More layers = more complex functional mapping
2) More efficient due to distributed representation

---

<table>
<thead>
<tr>
<th>Inputs (features)</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>production costs</td>
<td>y₁(1)</td>
</tr>
<tr>
<td>promotional costs</td>
<td>y₂(1)</td>
</tr>
<tr>
<td>genre of the movie</td>
<td>y₁(2)</td>
</tr>
<tr>
<td>box office first week</td>
<td>y₂(2)</td>
</tr>
<tr>
<td>total book sales</td>
<td>y₁(3)</td>
</tr>
<tr>
<td>total revenue USA</td>
<td>y₂(3)</td>
</tr>
<tr>
<td>total revenue international</td>
<td>y₁(3)</td>
</tr>
</tbody>
</table>

| Training Set | | Outputs |
|--------------| | ---------|
| x₁(1) x₂(1) x₃(1) x₄(1) x₅(1) | y₁(1) y₂(1) |
| x₁(2) x₂(2) x₃(2) x₄(2) x₅(2) | y₁(2) y₂(2) |
| x₁(3) x₂(3) x₃(3) x₄(3) x₅(3) | y₁(3) y₂(3) |

1 1 0 0 0

* slide from Marc’Aurelio Renzato

---

e.g., hidden unit = production cost + promotion cost

e.g., p(film over budget) = sigmoid (hidden unit)
Neural Network Intuition

**Question:** What is a Neural Network?
**Answer:** Complex mapping from an input (vector) to an output (vector)

**Question:** What class of functions should be considered for this mapping?
**Answer:** Compositions of simpler functions (a.k.a. layers)? We will talk more about what specific functions next …

**Question:** What does a hidden unit do?
**Answer:** It can be thought of as classifier or a feature.

**Question:** Why have many layers?
**Answer:** 1) More layers = more complex functional mapping
    2) More efficient due to distributed representation

* slide from Marc’Aurelio Renzato
Multi-layer Neural Network

Input Layer

$X_1$
$X_2$
$X_3$
$X_4$
$X_5$

1st Hidden Layer

$W_{h1}, b_{h1}$

2nd Hidden Layer

$W_{h2}, b_{h2}$

Output Layer

$y_1$
$y_2$

Recall: $a(x) = x$

Linear Activation

$y_1$
$y_2$
Multi-layer Neural Network

Recall: \( a(x) = x \)

Why?

Input Layer

Output Layer

1st Hidden Layer

2nd Hidden Layer

Linear Activation
Multi-layer Neural Network

Input Layer

[Diagram of a neural network showing layers and connections]

Recall: $a(x) = x$

Why?

$W_0 (W_{h2} (W_{h1}x + b_{h1}) + b_{h2}) + b_o =$

$[W_0 W_{h1} W_{h2}] x + [W_0 W_{h1} b_{h1} + W_0 b_{h2} + b_o]$

$W'$

$b'$

Linear Activation
Multi-layer Neural Network

Why?

$$W_o (W_{h2} (W_{h1}x + b_{h1}) + b_{h2}) + b_o = \frac{[W_o W_{h1} W_{h2}] x + [W_o W_{h1} b_{h1} + W_o b_{h2} + b_o]}{W'} + b'$$

Recall: $a(x) = x$ => entire neural network is linear, which is not expressive
One-layer Neural Network

Input Layer

Output Layer

Weighted Sum

Activation Function

$\sum \, a(\cdot) \rightarrow y_1$

$\sum \, a(\cdot) \rightarrow y_2$

$W_o, b_o$

Linear Activation

$a(x) = x$
One-layer **Neural Network**

Input Layer

```
\begin{align*}
\sum \quad & a(\cdot) \\
\sum \quad & a(\cdot) \\
\end{align*}
```

Output Layer

```
\begin{align*}
a(x) &= \text{sigmoid}(x) = \frac{1}{1 + e^{-x}} \\
\end{align*}
```

Sigmoid Activation
Light Theory: Neural Network as Universal Approximator

Neural network can arbitrarily approximate any \textbf{continuous} function for every value of possible inputs

*slide adopted from http://neuralnetworksanddeeplearning.com/chap4.html*
Neural network can arbitrarily approximate any continuous function for every value of possible inputs. The guarantee is that by using enough hidden neurons we can always find a neural network whose output $g(x)$ satisfies $|g(x) - f(x)| < \epsilon$ for an arbitrarily small $\epsilon$.
Light Theory: Neural Network as Universal Approximator

**Let's start with a simple network**: one hidden layer with two hidden neurons and a single output layer with one neuron (with sigmoid activations)

*slide adopted from http://neuralnetworksanddeeplearning.com/chap4.html*
Light Theory: Neural Network as Universal Approximator

Let's start with a simple network: one hidden layer with two hidden neurons and a single output layer with one neuron (with sigmoid activations).

Let’s look at output of this (hidden) neuron as a function of parameters (weight, bias).

Let's start with a simple network: one hidden layer with two hidden neurons and a single output layer with one neuron (with sigmoid activations)

Let's look at output of this (hidden) neuron as a function of parameters (weight, bias)

*slide adopted from http://neuralnetworksanddeeplearning.com/chap4.html*
Light Theory: Neural Network as Universal Approximator

Let’s start with a simple network: one hidden layer with two hidden neurons and a single output layer with one neuron (with sigmoid activations)

Let’s look at output of this (hidden) neuron as a function of parameters (weight, bias)

Light Theory: Neural Network as Universal Approximator

By dialing up the weight (e.g. $w = 999$) we can actually create a “step” function.

Light Theory: Neural Network as Universal Approximator

By dialing up the weight (e.g. $w = 999$) we can actually create a “step” function.

It is easier to work with sums of step functions, so we can assume that every neuron outputs a step function.

By dialing up the weight (e.g. $w = 999$) we can actually create a “step” function.

It is easier to work with sums of step functions, so we can assume that every neuron outputs a step function.

**Location of the step?**

*slide adopted from http://neuralnetworksanddeeplearning.com/chap4.html*
By dialing up the weight (e.g. $w = 999$) we can actually create a “step” function.

It is easier to work with sums of step functions, so we can assume that every neuron outputs a step function.

Location of the step?

$$s = -\frac{b}{w}$$
By dialing up the weight (e.g. $w = 999$) we can actually create a “step” function.

It is easier to work with sums of step functions, so we can assume that every neuron outputs a step function.

Location of the step?

$$s = -\frac{b}{w}$$

*slide adopted from http://neuralnetworksanddeeplearning.com/chap4.html*
The output neuron is a **weighted combination of step functions** (assuming bias for that layer is 0)
Light Theory: Neural Network as Universal Approximator

The output neuron is a **weighted combination of step functions** (assuming bias for that layer is 0)

The output neuron is a **weighted combination of step functions** (assuming bias for that layer is 0).

Light Theory: Neural Network as Universal Approximator

*slide adopted from http://neuralnetworksanddeeplearning.com/chap4.html*
Light Theory: Neural Network as Universal Approximator

Riemann sum approximation

Light Theory: Neural Network as Universal Approximator

Riemann sum approximation
Light Theory: Neural Network as Universal Approximator

Conditions needed for proof to hold: Activation function needs to be well defined

\[
\lim_{{x \to \infty}} a(x) = A
\]

\[
\lim_{{x \to -\infty}} a(x) = B
\]

\[
A \neq B
\]

*slide adopted from http://neuralnetworksanddeeplearning.com/chap4.html*
**Light Theory**: Neural Network as Universal Approximator

**Conditions needed for proof to hold**: Activation function needs to be well defined

\[
\lim_{{x \to \infty}} a(x) = A
\]

\[
\lim_{{x \to -\infty}} a(x) = B
\]

\[A \neq B\]

**Note**: This gives us another way to provably say that linear activation function cannot produce a neural network which is an universal approximator.

**Light Theory:** Neural Network as Universal Approximator

**Universal Approximation Theorem:** Single hidden layer can approximate any continuous function with compact support to arbitrary accuracy, when the width goes to infinity.  

[ Hornik et al., 1989 ]
**Light Theory**: Neural Network as Universal Approximator

**Universal Approximation Theorem**: Single hidden layer can approximate any continuous function with compact support to arbitrary accuracy, when the width goes to infinity.

[ Hornik et al., 1989 ]

**Universal Approximation Theorem (revised)**: A network of infinite depth with a hidden layer of size $d + 1$ neurons, where $d$ is the dimension of the input space, can approximate any continuous function.

[ Lu et al., NIPS 2017 ]
**Universal Approximation Theorem**: Single hidden layer can approximate any continuous function with compact support to arbitrary accuracy, when the width goes to infinity.  

[ Hornik et al., 1989 ]

**Universal Approximation Theorem (revised)**: A network of infinite depth with a hidden layer of size $d + 1$ neurons, where $d$ is the dimension of the input space, can approximate any continuous function.  

[ Lu et al., NIPS 2017 ]

**Universal Approximation Theorem (further revised)**: ResNet with a single hidden unit and infinite depth can approximate any continuous function.  

[ Lin and Jegelka, NIPS 2018 ]
**Light Theory**: Neural Network as Universal Approximator

Universal Approximation Theorem (further revised): ResNet with a single hidden unit and infinite depth can approximate any continuous function.

[ Lin and Jegelka, NIPS 2018 ]
Practical Observations

Neural Network represents a function using a piece-wise linear approximation.
Practical Observations

Neural Network represents a function using a **piece-wise linear** approximation

**Expressivity** (theoretic quality) of NN = the number of piece-wise linear regions
- Number of regions is a polynomial function of units per layer (breadth of NN)
- Number of regions is an exponential function of layers (depth of NN)

Note: in recent literature the # of parameters have been used as a proxy for expressiveness of NN, this is not a great practice, because it ignores topology.
Practical Observations

Neural Network represents a function using a piece-wise linear approximation

**Expressivity** (theoretic quality) of NN = the number of piece-wise linear regions
- Number of regions is a polynomial function of units per layer (breadth of NN)
- Number of regions is an exponential function of layers (depth of NN)

**Effectiveness** (practical quality) of the NN is also a function of optimization
- Deep networks are generally harder to optimize
Practical Observations

Neural Network represents a function using a **piece-wise linear** approximation

**Expressivity** (theoretic quality) of NN = the number of piece-wise linear regions
- Number of regions is a polynomial function of units per layer (breadth of NN)
- Number of regions is an exponential function of layers (depth of NN)

**Effectiveness** (practical quality) of the NN is also a function of optimization
- Deep networks are generally harder to optimize

**Note:** in recent literature the # of parameters have been used as a proxy for expressiveness of NN, this is not a great practice, because it ignores topology.
One-layer Neural Network

Input Layer

Output Layer

Weighted Sum

Activation Function

\[ a(x) = \text{sigmoid}(x) = \frac{1}{1 + e^{-x}} \]

Sigmoid Activation
Learning Parameters of One-layer Neural Network

\[ L(W, b) = \sum_{d=1}^{|D_{train}|} \left( \text{sigmoid} \left( W^T x^{(d)} + b \right) - y^{(d)} \right)^2 \]

\[ W^*, b^* = \arg \min L(W, b) \]
Learning Parameters of One-layer Neural Network

\[ \mathcal{L}(W, b) = \sum_{d=1}^{D_{\text{train}}} \left( \text{sigmoid} \left( W^T x^{(d)} + b \right) - y^{(d)} \right)^2 \]

\[ W^*, b^* = \arg \min \mathcal{L}(W, b) \]

Solution:

\[ \frac{\partial \mathcal{L}(W, b)}{\partial w_{ji}} = \frac{\partial}{\partial w_{ji}} \sum_{d=1}^{D_{\text{train}}} \left( \text{sigmoid} \left( W^T x^{(d)} + b \right) - y^{(d)} \right)^2 \]

\[ \frac{\partial \mathcal{L}(W, b)}{\partial w_{ji}} = \frac{\partial}{\partial w_{ji}} \sum_{d=1}^{D_{\text{train}}} \left( \text{sigmoid} \left( W^T x^{(d)} + b \right) - y^{(d)} \right)^2 = 0 \]

*slide adopted from V. Ordonex*
Learning Parameters of One-layer Neural Network

\[ \mathcal{L}(W, b) = \sum_{d=1}^{D_{\text{train}}} \left( \text{sigmoid} \left( W^T x^{(d)} + b \right) - y^{(d)} \right)^2 \]

\[ W^*, b^* = \arg \min \mathcal{L}(W, b) \]

Solution:

\[ \frac{\partial \mathcal{L}(W, b)}{\partial w_{ji}} = \frac{\partial}{\partial w_{ji}} \sum_{d=1}^{D_{\text{train}}} \left( \text{sigmoid} \left( W^T x^{(d)} + b \right) - y^{(d)} \right)^2 \]

\[ \frac{\partial \mathcal{L}(W, b)}{\partial w_{ji}} = \frac{\partial}{\partial w_{ji}} \sum_{d=1}^{D_{\text{train}}} \left( \text{sigmoid} \left( W^T x^{(d)} + b \right) - y^{(d)} \right)^2 = 0 \]

Problem: No closed form solution \( \frac{\partial \mathcal{L}(W, b)}{\partial w_{ji}} = 0 \)

*slide adopted from V. Ordonex*
Gradient Descent (review)

\[ L(W, b) = \sum_{d=1}^{\text{\mid}D_{\text{train}}\text{\mid}} \left( \text{sigmoid} \left( W^T x^{(d)} + b \right) - y^{(d)} \right)^2 \]
Gradient Descent (review)

\[ \mathcal{L}(W, b) = \sum_{d=1}^{|D_{\text{train}}|} \left( \text{sigmoid} \left( W^T x^{(d)} + b \right) - y^{(d)} \right)^2 \]

1. Start from random value of \( W_0, b_0 \)

*slide adopted from V. Ordonex*
Gradient Descent (review)

\[ \mathcal{L}(W, b) = \sum_{d=1}^{\mid D_{\text{train}} \mid} \left( \text{sigmoid} \left( W^T x^{(d)} + b \right) - y^{(d)} \right)^2 \]

1. Start from random value of \( W_0, b_0 \)

*slide adopted from V. Ordonex*
Gradient Descent (review)

\[ \mathcal{L}(W, b) = \sum_{d=1}^{D_{\text{train}}} \left( \text{sigmoid} \left( W^T x^{(d)} + b \right) - y^{(d)} \right)^2 \]

1. Start from random value of \( W_0, b_0 \)

For \( k = 0 \) to max number of iterations

2. Compute gradient of the loss with respect to previous (initial) parameters:

\[ \nabla \mathcal{L}(W, b) \bigg|_{W=W_k, b=b_k} \]

*slide adopted from V. Ordonex*
Gradient Descent (review)

\[ L(W, b) = \sum_{d=1}^{|D_{\text{train}}|} \left( \text{sigmoid} \left( W^T x^{(d)} + b \right) - y^{(d)} \right)^2 \]

1. Start from random value of \( W_0, b_0 \)

For \( k = 0 \) to max number of iterations

2. Compute gradient of the loss with respect to previous (initial) parameters:

\[ \nabla L(W, b) \big|_{W=W_k, b=b_k} \]

*slide adopted from V. Ordonex*
Gradient Descent (review)

\[ \mathcal{L}(W, b) = \sum_{d=1}^{|D_{\text{train}}|} \left( \text{sigmoid} \left( W^T x^{(d)} + b \right) - y^{(d)} \right)^2 \]

1. Start from random value of \( W_0, b_0 \)

For \( k = 0 \) to max number of iterations

2. Compute gradient of the loss with respect to previous (initial) parameters:

\[ \nabla \mathcal{L}(W, b)|_{W=W_k, b=b_k} \]

3. Re-estimate the parameters

\[ W_{k+1} = W_k - \lambda \frac{\partial \mathcal{L}(W, b)}{\partial W}|_{W=W_k, b=b_k} \]

\[ b_{k+1} = b_k - \lambda \frac{\partial \mathcal{L}(W, b)}{\partial b}|_{W=W_k, b=b_k} \]

*slide adopted from V. Ordonex*
Gradient Descent (review)

\[ \mathcal{L}(W, b) = \sum_{d=1}^{\left| D_{\text{train}} \right|} \left( \text{sigmoid} \left( W^T x^{(d)} + b \right) - y^{(d)} \right)^2 \]

1. Start from random value of \( W_0, b_0 \)

For \( k = 0 \) to max number of iterations

2. Compute gradient of the loss with respect to previous (initial) parameters:

\[ \left. \nabla \mathcal{L}(W, b) \right|_{W=W_k, b=b_k} \]

3. Re-estimate the parameters

\[ W_{k+1} = W_k - \lambda \frac{\partial \mathcal{L}(W, b)}{\partial W} \bigg|_{W=W_k, b=b_k} \]
\[ b_{k+1} = b_k - \lambda \frac{\partial \mathcal{L}(W, b)}{\partial b} \bigg|_{W=W_k, b=b_k} \]

*slide adopted from V. Ordonex*
Gradient Descent (review)

\[ \mathcal{L}(W, b) = \frac{1}{|D_{\text{train}}|} \sum_{d=1}^{|D_{\text{train}}|} \left( \text{sigmoid} \left( W^T x^{(d)} + b \right) - y^{(d)} \right)^2 \]

1. Start from random value of \( W_0, b_0 \)

For \( k = 0 \) to max number of iterations

2. Compute gradient of the loss with respect to previous (initial) parameters:

\[ \nabla \mathcal{L}(W, b) \bigg|_{W=W_k, b=b_k} \]

3. Re-estimate the parameters

\[
\begin{align*}
W_{k+1} &= W_k - \lambda \left. \frac{\partial \mathcal{L}(W, b)}{\partial W} \right|_{W=W_k, b=b_k} \\
b_{k+1} &= b_k - \lambda \left. \frac{\partial \mathcal{L}(W, b)}{\partial b} \right|_{W=W_k, b=b_k}
\end{align*}
\]

*slide adopted from V. Ordonex*
Gradient Descent (review)

\[ \mathcal{L}(W, b) = \sum_{d=1}^{|D_{\text{train}}|} \left( \text{sigmoid} \left( W^T x^{(d)} + b \right) - y^{(d)} \right)^2 \]

1. Start from random value of \( W_0, b_0 \)

For \( k = 0 \) to max number of iterations

2. Compute gradient of the loss with respect to previous (initial) parameters:

\[ \nabla \mathcal{L}(W, b)|_{W=W_k, b=b_k} \]

3. Re-estimate the parameters

\[ W_{k+1} = W_k - \lambda \frac{\partial \mathcal{L}(W, b)}{\partial W}|_{W=W_k, b=b_k} \]

\[ b_{k+1} = b_k - \lambda \frac{\partial \mathcal{L}(W, b)}{\partial b}|_{W=W_k, b=b_k} \]

\( \lambda \) - is the learning rate

*slide adopted from V. Ordonex
Stochastic Gradient Descent (review)

\[ \frac{\partial L(W, b)}{\partial w_{ji}} = \frac{\partial}{\partial w_{ji}} \sum_{d=1}^{|D_{\text{train}}|} \left( \text{sigmoid} \left( W^T x^{(d)} + b \right) - y^{(d)} \right)^2 \]
Stochastic Gradient Descent (review)

\[
\frac{\partial L(W, b)}{\partial w_{ji}} = \frac{\partial}{\partial w_{ji}} \sum_{d=1}^{D_{\text{train}}} \left( \text{sigmoid} \left( W^T x^{(d)} + b \right) - y^{(d)} \right)^2
\]

**Problem:** For large datasets computing sum is expensive
Stochastic Gradient Descent (review)

Problem: For large datasets computing sum is expensive

Solution: Compute approximate gradient with mini-batches of much smaller size (as little as 1-example sometimes)
Stochastic Gradient Descent (review)

\[
\frac{\partial \mathcal{L}(W, b)}{\partial w_{ji}} = \sum_{d=1}^{\lvert D_{\text{train}} \rvert} \left( \text{sigmoid} \left( W^T x^{(d)} + b \right) - y^{(d)} \right)^2
\]

**Problem:** For large datasets computing sum is expensive

**Solution:** Compute approximate gradient with mini-batches of much smaller size (as little as 1-example sometimes)

**Problem:** How do we compute the actual gradient?
Numerical Differentiation

We can approximate the gradient numerically, using:

$$\frac{\partial f(x)}{\partial x_i} \approx \lim_{h \to 0} \frac{f(x + h1_i) - f(x)}{h}$$

1_i - Vector of all zeros, except for one 1 in i-th location

*slide adopted from T. Chen, H. Shen, A. Krishnamurthy CSE 599G1 lecture at UWashington
We can approximate the gradient numerically, using:

\[
\frac{\partial f(x)}{\partial x_i} \approx \lim_{h \to 0} \frac{f(x + h1_i) - f(x)}{h}
\]

Even better, we can use central differencing:

\[
\frac{\partial f(x)}{\partial x_i} \approx \lim_{h \to 0} \frac{f(x + h1_i) - f(x - h1_i)}{2h}
\]

\(1_i\) - Vector of all zeros, except for one 1 in i-th location

*slide adopted from T. Chen, H. Shen, A. Krishnamurthy CSE 599G1 lecture at UWashington
Numerical Differentiation

We can approximate the gradient numerically, using:

\[
\frac{\partial f(x)}{\partial x_i} \approx \lim_{h \to 0} \frac{f(x + h \mathbf{1}_i) - f(x)}{h}
\]

Even better, we can use central differencing:

\[
\frac{\partial f(x)}{\partial x_i} \approx \lim_{h \to 0} \frac{f(x + h \mathbf{1}_i) - f(x - h \mathbf{1}_i)}{2h}
\]

However, both of these suffer from rounding errors and are not good enough for learning (they are very good tools for checking the correctness of implementation though, e.g., use \( h = 0.000001 \)).

*slide adopted from T. Chen, H. Shen, A. Krishnamurthy CSE 599G1 lecture at UWashington
Numerical Differentiation

We can approximate the gradient numerically, using:

\[
\frac{\partial L(W, b)}{\partial w_{ij}} \approx \lim_{h \to 0} \frac{L(W + h1_{ij}, b) - L(W, b)}{h}
\]

\[
\frac{\partial L(W, b)}{\partial b_{j}} \approx \lim_{h \to 0} \frac{L(W, b + h1_{j}) - L(W, b)}{h}
\]

Even better, we can use central differencing:

\[
\frac{\partial L(W, b)}{\partial w_{ij}} \approx \lim_{h \to 0} \frac{L(W + h1_{ij}, b) - L(W, b)}{2h}
\]

\[
\frac{\partial L(W, b)}{\partial b_{j}} \approx \lim_{h \to 0} \frac{L(W, b + h1_{j}) - L(W, b)}{2h}
\]

However, both of these suffer from rounding errors and are not good enough for learning (they are very good tools for checking the correctness of implementation though, e.g., use \(h = 0.000001\)).
Symbolic Differentiation

Input function is represented as **computational graph** (a symbolic tree)

\[ y = f(x_1, x_2) = \ln(x_1) + x_1 x_2 - \sin(x_2) \]

Implements differentiation rules for composite functions:

- **Sum Rule**
  \[
  \frac{d}{dx} (f(x) + g(x)) = \frac{df(x)}{dx} + \frac{dg(x)}{dx}
  \]

- **Product Rule**
  \[
  \frac{d}{dx} (f(x) \cdot g(x)) = \frac{df(x)}{dx} g(x) + f(x) \frac{dg(x)}{dx}
  \]

- **Chain Rule**
  \[
  \frac{d}{dx} (f(g(x))) = \frac{df(g(x))}{dg(x)} \cdot \frac{dg(x)}{dx}
  \]

*slide adopted from T. Chen, H. Shen, A. Krishnamurthy CSE 599G1 lecture at UWashing
Symbolic Differentiation

Input function is represented as **computational graph** (a symbolic tree)

![Computational Graph](image)

Implements differentiation rules for composite functions:

- **Sum Rule**
  \[
  \frac{d}{dx}(f(x) + g(x)) = \frac{d}{dx}f(x) + \frac{d}{dx}g(x)
  \]

- **Product Rule**
  \[
  \frac{d}{dx}(f(x) \cdot g(x)) = \frac{d}{dx}f(x) \cdot g(x) + f(x) \frac{d}{dx}g(x)
  \]

- **Chain Rule**
  \[
  \frac{d}{dx}(f(g(x))) = \frac{d}{dx}f(g(x)) \cdot \frac{d}{dx}g(x)
  \]

**Problem:** For complex functions, expressions can be exponentially large; also difficult to deal with piece-wise functions (creates many symbolic cases)

*slide adopted from T. Chen, H. Shen, A. Krishnamurthy CSE 599G1 lecture at UWashington*
Automatic Differentiation (AutoDiff)  

\[ y = f(x_1, x_2) = \ln(x_1) + x_1 x_2 - \sin(x_2) \]

**Intuition:** Interleave symbolic differentiation and simplification

**Key Idea:** apply symbolic differentiation at the elementary operation level, evaluate and keep intermediate results

*slide adopted from T. Chen, H. Shen, A. Krishnamurthy CSE 599G1 lecture at UWashington*
**Automatic Differentiation (AutoDiff)**

\[ y = f(x_1, x_2) = \ln(x_1) + x_1 x_2 - \sin(x_2) \]

**Intuition:** Interleave symbolic differentiation and simplification

**Key Idea:** apply symbolic differentiation at the elementary operation level, evaluate and keep intermediate results

Success of **deep learning** owes A LOT to success of AutoDiff algorithms (also to advances in parallel architectures, and large datasets, …)

*slide adopted from T. Chen, H. Shen, A. Krishnamurthy CSE 599G1 lecture at UWashington*
Automatic Differentiation (AutoDiff) 

\[ y = f(x_1, x_2) = \ln(x_1) + x_1 x_2 - \sin(x_2) \]

Each node is an input, intermediate, or output variable

Computational graph (a DAG) with variable ordering from topological sort.

*slide adopted from T. Chen, H. Shen, A. Krishnamurthy CSE 599G1 lecture at UWashington
Automatic Differentiation (AutoDiff)  

Each node is an input, intermediate, or output variable

Computational graph (a DAG) with variable ordering from topological sort.

\[
y = f(x_1, x_2) = \ln(x_1) + x_1 x_2 - \sin(x_2)
\]

\[
\begin{align*}
  v_0 &= x_1 \\
  v_1 &= x_2 \\
  v_2 &= \ln(v_0) \\
  v_3 &= v_0 \cdot v_1 \\
  v_4 &= \sin(v_1) \\
  v_5 &= v_2 + v_3 \\
  v_6 &= v_5 - v_4 \\
  y &= v_6
\end{align*}
\]

*slide adopted from T. Chen, H. Shen, A. Krishnamurthy CSE 599G1 lecture at UWashington
Automatic Differentiation (AutoDiff)

Each node is an input, intermediate, or output variable

Computational graph (a DAG) with variable ordering from topological sort.

\[
y = f(x_1, x_2) = \ln(x_1) + x_1 x_2 - \sin(x_2)
\]

Lets see how we can evaluate a function using computational graph (DNN inferences)

Computational graph is governed by these equations

\[
\begin{align*}
v_0 &= x_1 \\
v_1 &= x_2 \\
v_2 &= \ln(v_0) \\
v_3 &= v_0 \cdot v_1 \\
v_4 &= \sin(v_1) \\
v_5 &= v_2 + v_3 \\
v_6 &= v_5 - v_4 \\
y &= v_6
\end{align*}
\]

*slide adopted from T. Chen, H. Shen, A. Krishnamurthy CSE 599G1 lecture at UWashington
Automatic Differentiation (AutoDiff)

Each node is an input, intermediate, or output variable.

Computational graph (a DAG) with variable ordering from topological sort.

\[
y = f(x_1, x_2) = \ln(x_1) + x_1 x_2 - \sin(x_2)
\]

Let's see how we can evaluate a function using computational graph (DNN inferences)

Forward Evaluation Trace:

<table>
<thead>
<tr>
<th>(f(2, 5))</th>
</tr>
</thead>
<tbody>
<tr>
<td>(v_0 = x_1)</td>
</tr>
<tr>
<td>(v_1 = x_2)</td>
</tr>
<tr>
<td>(v_2 = \ln(v_0))</td>
</tr>
<tr>
<td>(v_3 = v_0 \cdot v_1)</td>
</tr>
<tr>
<td>(v_4 = \sin(v_1))</td>
</tr>
<tr>
<td>(v_5 = v_2 + v_3)</td>
</tr>
<tr>
<td>(v_6 = v_5 - v_4)</td>
</tr>
<tr>
<td>(y = v_6)</td>
</tr>
</tbody>
</table>
Automatic Differentiation (AutoDiff)

Each node is an input, intermediate, or output variable.

Computational graph (a DAG) with variable ordering from topological sort.

\[ y = f(x_1, x_2) = \ln(x_1) + x_1 x_2 - \sin(x_2) \]

Let's see how we can evaluate a function using computational graph (DNN inferences).

**Forward Evaluation** Trace:

<table>
<thead>
<tr>
<th>( v_0 )</th>
<th>( v_1 )</th>
<th>( v_2 )</th>
<th>( v_3 )</th>
<th>( v_4 )</th>
<th>( v_5 )</th>
<th>( v_6 )</th>
<th>( y )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x_1 )</td>
<td>( x_2 )</td>
<td>( \ln(v_0) )</td>
<td>( v_0 \cdot v_1 )</td>
<td>( \sin(v_1) )</td>
<td>( v_2 + v_3 )</td>
<td>( v_5 - v_4 )</td>
<td>( y = v_6 )</td>
</tr>
</tbody>
</table>

\( f(2, 5) \):

\[ y = f(2, 5) = \ln(2) + 2 \cdot 5 - \sin(5) \]
Automatic Differentiation (AutoDiff)

Each node is an input, intermediate, or output variable

Computational graph (a DAG) with variable ordering from topological sort.

Let's see how we can evaluate a function using computational graph (DNN inferences)

**Forward Evaluation** Trace:

<table>
<thead>
<tr>
<th></th>
<th>$f(2, 5)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$v_0 = x_1$</td>
<td>2</td>
</tr>
<tr>
<td>$v_1 = x_2$</td>
<td>5</td>
</tr>
<tr>
<td>$v_2 = \ln(v_0)$</td>
<td></td>
</tr>
<tr>
<td>$v_3 = v_0 \cdot v_1$</td>
<td></td>
</tr>
<tr>
<td>$v_4 = \sin(v_1)$</td>
<td></td>
</tr>
<tr>
<td>$v_5 = v_2 + v_3$</td>
<td></td>
</tr>
<tr>
<td>$v_6 = v_5 - v_4$</td>
<td></td>
</tr>
<tr>
<td>$y = v_6$</td>
<td></td>
</tr>
</tbody>
</table>

$y = f(x_1, x_2) = \ln(x_1) + x_1 x_2 - \sin(x_2)$
Automatic Differentiation (AutoDiff)  

\[ y = f(x_1, x_2) = \ln(x_1) + x_1x_2 - \sin(x_2) \]

Each **node** is an input, intermediate, or output variable.

**Computational graph** (a DAG) with variable ordering from topological sort.

Let's see how we can **evaluate a function** using computational graph (DNN inferences).

Forward Evaluation Trace:

| \( v_0 = x_1 \) | 2  |
| \( v_1 = x_2 \) | 5  |
| \( v_2 = \ln(v_0) \) | \( \ln(2) = 0.693 \) |
| \( v_3 = v_0 \cdot v_1 \) |
| \( v_4 = \sin(v_1) \) |
| \( v_5 = v_2 + v_3 \) |
| \( v_6 = v_5 - v_4 \) |
| \( y = v_6 \) |
Automatically Differentiation (AutoDiff)

Each node is an input, intermediate, or output variable

Computational graph (a DAG) with variable ordering from topological sort.

\[
y = f(x_1, x_2) = \ln(x_1) + x_1x_2 - \sin(x_2)
\]

Let's see how we can evaluate a function using computational graph (DNN inferences)

**Forward Evaluation** Trace:

\[
\begin{align*}
v_0 &= x_1 & 2 \\
v_1 &= x_2 & 5 \\
v_2 &= \ln(v_0) & \ln(2) = 0.693 \\
v_3 &= v_0 \cdot v_1 & 2 \times 5 = 10 \\
v_4 &= \sin(v_1) & \sin(5) = 0.959 \\
v_5 &= v_2 + v_3 & 0.693 + 10 = 10.693 \\
v_6 &= v_5 - v_4 & 10.693 + 0.959 = 11.652 \\
y &= v_6 & 11.652
\end{align*}
\]
**Automatic Differentiation (AutoDiff)**

Each node is an input, intermediate, or output variable.

**Computational graph** (a DAG) with variable ordering from topological sort.

\[
y = f(x_1, x_2) = \ln(x_1) + x_1 x_2 - \sin(x_2)
\]

Let's see how we can **evaluate a function** using computational graph (DNN inferences)

**Forward Evaluation** Trace:

<table>
<thead>
<tr>
<th></th>
<th>( f(2, 5) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( v_0 = x_1 )</td>
<td>2</td>
</tr>
<tr>
<td>( v_1 = x_2 )</td>
<td>5</td>
</tr>
<tr>
<td>( v_2 = \ln(v_0) )</td>
<td>( \ln(2) = 0.693 )</td>
</tr>
<tr>
<td>( v_3 = v_0 \cdot v_1 )</td>
<td>( 2 \times 5 = 10 )</td>
</tr>
<tr>
<td>( v_4 = \sin(v_1) )</td>
<td>( \sin(5) = 0.959 )</td>
</tr>
<tr>
<td>( v_5 = v_2 + v_3 )</td>
<td>( 0.693 + 10 = 10.693 )</td>
</tr>
<tr>
<td>( v_6 = v_5 - v_4 )</td>
<td>( 10.693 + 0.959 = 11.652 )</td>
</tr>
<tr>
<td>( y = v_6 )</td>
<td>11.652</td>
</tr>
</tbody>
</table>
Automatic Differentiation (AutoDiff)

\[ y = f(x_1, x_2) = \ln(x_1) + x_1 x_2 - \sin(x_2) \]

**Forward Evaluation** Trace:

<table>
<thead>
<tr>
<th>( v_0 = x_1 )</th>
<th>( f(2, 5) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( v_1 = x_2 )</td>
<td>2</td>
</tr>
<tr>
<td>( v_2 = \ln(v_0) )</td>
<td>5</td>
</tr>
<tr>
<td>( v_3 = v_0 \cdot v_1 )</td>
<td>( \ln(2) = 0.693 )</td>
</tr>
<tr>
<td>( v_4 = \sin(v_1) )</td>
<td>( 2 \times 5 = 10 )</td>
</tr>
<tr>
<td>( v_5 = v_2 + v_3 )</td>
<td>( \sin(5) = 0.959 )</td>
</tr>
<tr>
<td>( v_6 = v_5 - v_4 )</td>
<td>( 0.693 + 10 = 10.693 )</td>
</tr>
<tr>
<td>( y = v_6 )</td>
<td>( 10.693 + 0.959 = 11.652 )</td>
</tr>
</tbody>
</table>
AutoDiff - Forward Mode

\[ y = f(x_1, x_2) = \ln(x_1) + x_1 x_2 - \sin(x_2) \]

\[
\begin{align*}
\frac{\partial f(x_1, x_2)}{\partial x_1} \bigg|_{(x_1=2, x_2=5)}
\end{align*}
\]

Let's see how we can evaluate a derivative using computational graph (DNN learning)

We will do this with forward mode first, by introducing a derivative of each variable node with respect to the input variable.
AutoDiff - **Forward Mode**

\[ y = f(x_1, x_2) = \ln(x_1) + x_1x_2 - \sin(x_2) \]

**Forward Evaluation** Trace:

<table>
<thead>
<tr>
<th>( v_0 = x_1 )</th>
<th>( f(2, 5) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( v_1 = x_2 )</td>
<td>2</td>
</tr>
<tr>
<td>( v_2 = \ln(v_0) )</td>
<td>( \ln(2) = 0.693 )</td>
</tr>
<tr>
<td>( v_3 = v_0 \cdot v_1 )</td>
<td>2 \times 5 = 10</td>
</tr>
<tr>
<td>( v_4 = \sin(v_1) )</td>
<td>( \sin(5) = 0.959 )</td>
</tr>
<tr>
<td>( v_5 = v_2 + v_3 )</td>
<td>0.693 + 10 = 10.693</td>
</tr>
<tr>
<td>( v_6 = v_5 - v_4 )</td>
<td>10.693 + 0.959 = 11.652</td>
</tr>
<tr>
<td>( y = v_6 )</td>
<td>11.652</td>
</tr>
</tbody>
</table>

**Forward Derivative** Trace:

\[
\frac{\partial f(x_1, x_2)}{\partial x_1} \bigg|_{(x_1=2, x_2=5)}
\]
AutoDiff - **Forward Mode**

\[ y = f(x_1, x_2) = \ln(x_1) + x_1 x_2 - \sin(x_2) \]

**Forward Evaluation Trace:**

| \( f(2, 5) \) |  
|---|---|---|---|---|---|---|---|
| \( v_0 = x_1 \) | 2 |
| \( v_1 = x_2 \) | 5 |
| \( v_2 = \ln(v_0) \) | \( \ln(2) = 0.693 \) |
| \( v_3 = v_0 \cdot v_1 \) | \( 2 \times 5 = 10 \) |
| \( v_4 = \sin(v_1) \) | \( \sin(5) = 0.959 \) |
| \( v_5 = v_2 + v_3 \) | \( 0.693 + 10 = 10.693 \) |
| \( v_6 = v_5 - v_4 \) | \( 10.693 + 0.959 = 11.652 \) |
| \( y = v_6 \) | 11.652 |

**Forward Derivative Trace:**

\[
\frac{\partial f(x_1, x_2)}{\partial x_1} \bigg|_{(x_1=2, x_2=5)}
\]
AutoDiff - **Forward Mode**

\[ y = f(x_1, x_2) = \ln(x_1) + x_1 x_2 - \sin(x_2) \]

**Forward Evaluation** Trace:

<table>
<thead>
<tr>
<th>( v_0 = x_1 )</th>
<th>( f(2, 5) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( v_1 = x_2 )</td>
<td>2</td>
</tr>
<tr>
<td>( v_2 = \ln(v_0) )</td>
<td>( \ln(2) = 0.693 )</td>
</tr>
<tr>
<td>( v_3 = v_0 \cdot v_1 )</td>
<td>2 x 5 = 10</td>
</tr>
<tr>
<td>( v_4 = \sin(v_1) )</td>
<td>( \sin(5) = 0.959 )</td>
</tr>
<tr>
<td>( v_5 = v_2 + v_3 )</td>
<td>0.693 + 10 = 10.693</td>
</tr>
<tr>
<td>( v_6 = v_5 - v_4 )</td>
<td>10.693 + 0.959 = 11.652</td>
</tr>
<tr>
<td>( y = v_6 )</td>
<td>11.652</td>
</tr>
</tbody>
</table>

**Forward Derivative** Trace:

\[ \frac{\partial f(x_1, x_2)}{\partial x_1} \bigg|_{(x_1=2,x_2=5)} = 1 \]
AutoDiff - **Forward Mode**

\[
y = f(x_1, x_2) = \ln(x_1) + x_1 x_2 - \sin(x_2)
\]

**Forward Evaluation Trace:**

<table>
<thead>
<tr>
<th>( v_0 = x_1 )</th>
<th>( f(2, 5) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( v_1 = x_2 )</td>
<td>2</td>
</tr>
<tr>
<td>( v_2 = \ln(v_0) )</td>
<td>5</td>
</tr>
<tr>
<td>( v_3 = v_0 \cdot v_1 )</td>
<td>( \ln(2) = 0.693 )</td>
</tr>
<tr>
<td>( v_4 = \sin(v_1) )</td>
<td>2 \times 5 = 10</td>
</tr>
<tr>
<td>( v_5 = v_2 + v_3 )</td>
<td>( 0.693 + 10 = 10.693 )</td>
</tr>
<tr>
<td>( v_6 = v_5 - v_4 )</td>
<td>10.693 + 0.959 = 11.652</td>
</tr>
<tr>
<td>( y = v_6 )</td>
<td>11.652</td>
</tr>
</tbody>
</table>

**Forward Derivative Trace:**

\[
\frac{\partial f(x_1, x_2)}{\partial x_1} \bigg|_{(x_1=2, x_2=5)} = 1
\]
$$\begin{align*}
y &= f(x_1, x_2) = \ln(x_1) + x_1 x_2 - \sin(x_2) \\
\end{align*}$$
AutoDiff - Forward Mode

\[ y = f(x_1, x_2) = \ln(x_1) + x_1 x_2 - \sin(x_2) \]

Forward Derivative Trace:

<table>
<thead>
<tr>
<th>( \frac{\partial f(x_1, x_2)}{\partial x_1} )</th>
<th>(( x_1 = 2 ), ( x_2 = 5 ))</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \frac{\partial v_0}{\partial x_1} )</td>
<td>1</td>
</tr>
<tr>
<td>( \frac{\partial v_1}{\partial x_1} )</td>
<td>0</td>
</tr>
<tr>
<td>( \frac{\partial v_2}{\partial x_1} )</td>
<td></td>
</tr>
<tr>
<td>( \frac{\partial v_3}{\partial x_1} )</td>
<td></td>
</tr>
<tr>
<td>( \frac{\partial v_4}{\partial x_1} )</td>
<td></td>
</tr>
<tr>
<td>( \frac{\partial v_5}{\partial x_1} )</td>
<td></td>
</tr>
<tr>
<td>( \frac{\partial v_6}{\partial x_1} )</td>
<td></td>
</tr>
</tbody>
</table>

Trace:

| \( v_0 = x_1 \) | 2  |
| \( v_1 = x_2 \) | 5  |
| \( v_2 = \ln(v_0) \) | \( \ln(2) = 0.693 \) |
| \( v_3 = v_0 \cdot v_1 \) | \( 2 \times 5 = 10 \) |
| \( v_4 = \sin(v_1) \) | \( \sin(5) = 0.959 \) |
| \( v_5 = v_2 + v_3 \) | \( 0.693 + 10 = 10.693 \) |
| \( v_6 = v_5 - v_4 \) | \( 10.693 + 0.959 = 11.652 \) |
| \( y = v_6 \) | \( 11.652 \) |
\[
y = f(x_1, x_2) = \ln(x_1) + x_1 x_2 - \sin(x_2)
\]

**Forward Derivative**

Trace:

\[
\begin{align*}
\frac{\partial f(x_1, x_2)}{\partial x_1} & \bigg|_{(x_1=2, x_2=5)} \\
\frac{\partial v_0}{\partial x_1} & = 1 \\
\frac{\partial v_1}{\partial x_1} & = 0 \\
\frac{\partial v_2}{\partial x_1} & \\
\frac{\partial v_3}{\partial x_1} & \\
\frac{\partial v_4}{\partial x_1} & \\
\frac{\partial v_5}{\partial x_1} & \\
\frac{\partial v_6}{\partial x_1} & \\
y & = 11.652
\end{align*}
\]

**Chain Rule**

**Forward Evaluation**

Trace:

\[
\begin{align*}
f(2, 5) & \\
v_0 & = x_1 \quad \text{2} \\
v_1 & = x_2 \quad \text{5} \\
v_2 & = \ln(v_0) \quad \ln(2) = 0.693 \\
v_3 & = v_0 \cdot v_1 \quad 2 \times 5 = 10 \\
v_4 & = \sin(v_1) \quad \sin(5) = 0.959 \\
v_5 & = v_2 + v_3 \quad 0.693 + 10 = 10.693 \\
v_6 & = v_5 - v_4 \quad 10.693 + 0.959 = 11.652 \\
y & = v_6 \quad 11.652
\end{align*}
\]

**AutoDiff - Forward Mode**
AutoDiff - **Forward Mode**

\[ y = f(x_1, x_2) = \ln(x_1) + x_1 x_2 - \sin(x_2) \]

**Forward Evaluation** Trace:

<table>
<thead>
<tr>
<th>( v_0 )</th>
<th>( v_1 )</th>
<th>( v_2 )</th>
<th>( v_3 )</th>
<th>( v_4 )</th>
<th>( v_5 )</th>
<th>( v_6 )</th>
<th>( y )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x_1 )</td>
<td>( x_2 )</td>
<td>( \ln(v_0) )</td>
<td>( v_0 \cdot v_1 )</td>
<td>( \sin(v_1) )</td>
<td>( v_2 + v_3 )</td>
<td>( v_5 - v_4 )</td>
<td>( v_6 )</td>
</tr>
<tr>
<td>( 2 )</td>
<td>( 5 )</td>
<td>( \ln(2) = 0.693 )</td>
<td>( 2 \times 5 = 10 )</td>
<td>( \sin(5) = 0.959 )</td>
<td>( 0.693 + 10 = 10.693 )</td>
<td>( 10.693 + 0.959 = 11.652 )</td>
<td>( 11.652 )</td>
</tr>
</tbody>
</table>

\[ \frac{\partial v_0}{\partial x_1} = 1 \]
\[ \frac{\partial v_1}{\partial x_1} = 0 \]
\[ \frac{\partial v_2}{\partial x_1} = \frac{1}{v_0} \frac{\partial v_0}{\partial x_1} \]

**Forward Derivative** Trace:

\[ \frac{\partial f(x_1, x_2)}{\partial x_1} \bigg|_{(x_1=2, x_2=5)} \]

**Chain Rule**
AutoDiff - **Forward Mode**

\[ y = f(x_1, x_2) = \ln(x_1) + x_1 x_2 - \sin(x_2) \]

**Forward Evaluation** Trace:

| \( v_0 = x_1 \) | 2 |
| \( v_1 = x_2 \) | 5 |
| \( v_2 = \ln(v_0) \) | \( \ln(2) = 0.693 \) |
| \( v_3 = v_0 \cdot v_1 \) | 2 \times 5 = 10 |
| \( v_4 = \sin(v_1) \) | \( \sin(5) = 0.959 \) |
| \( v_5 = v_2 + v_3 \) | 0.693 + 10 = 10.693 |
| \( v_6 = v_5 - v_4 \) | 10.693 + 0.959 = 11.652 |
| \( y = v_6 \) | 11.652 |

**Forward Derivative** Trace:

\[
\frac{\partial f(x_1, x_2)}{\partial x_1} \bigg|_{(x_1=2, x_2=5)} = 1 \\
\frac{\partial v_0}{\partial x_1} = 0 \\
\frac{\partial v_2}{\partial x_1} = \frac{1}{v_0} \frac{\partial v_0}{\partial x_1} = \frac{1}{2} \times 1 = 0.5
\]

**Chain Rule**
AutoDiff - Forward Mode

Forward Evaluation Trace:

<table>
<thead>
<tr>
<th>$v_0 = x_1$</th>
<th>$f(2, 5)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$v_1 = x_2$</td>
<td>2</td>
</tr>
<tr>
<td>$v_2 = \ln(v_0)$</td>
<td>5</td>
</tr>
<tr>
<td>$v_3 = v_0 \cdot v_1$</td>
<td>$\ln(2) = 0.693$</td>
</tr>
<tr>
<td>$v_4 = \sin(v_1)$</td>
<td>$2 \times 5 = 10$</td>
</tr>
<tr>
<td>$v_5 = v_2 + v_3$</td>
<td>$0.693 + 10 = 10.693$</td>
</tr>
<tr>
<td>$v_6 = v_5 - v_4$</td>
<td>$10.693 + 0.959 = 11.652$</td>
</tr>
<tr>
<td>$y = v_6$</td>
<td>11.652</td>
</tr>
</tbody>
</table>

$y = f(x_1, x_2) = \ln(x_1) + x_1 x_2 - \sin(x_2)$

Forward Derivative Trace:

<table>
<thead>
<tr>
<th>$\frac{\partial f(x_1, x_2)}{\partial x_1}$</th>
<th>$(x_1=2, x_2=5)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\frac{\partial v_0}{\partial x_1}$</td>
<td>1</td>
</tr>
<tr>
<td>$\frac{\partial v_1}{\partial x_1}$</td>
<td>0</td>
</tr>
<tr>
<td>$\frac{\partial v_2}{\partial x_1}$</td>
<td>$\frac{1}{v_0} \cdot \frac{\partial v_0}{\partial x_1}$</td>
</tr>
<tr>
<td>$\frac{\partial v_3}{\partial x_1}$</td>
<td>$1/2 \times 1 = 0.5$</td>
</tr>
</tbody>
</table>
AutoDiff - Forward Mode

\[ y = f(x_1, x_2) = \ln(x_1) + x_1x_2 - \sin(x_2) \]

**Forward Derivative** Trace:

<table>
<thead>
<tr>
<th>( \frac{\partial f(x_1, x_2)}{\partial x_1} )</th>
<th>( (x_1=2, x_2=5) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \frac{\partial v_0}{\partial x_1} )</td>
<td>1</td>
</tr>
<tr>
<td>( \frac{\partial v_1}{\partial x_1} )</td>
<td>0</td>
</tr>
<tr>
<td>( \frac{\partial v_2}{\partial x_1} )</td>
<td>( \frac{1}{v_0} \frac{\partial v_0}{\partial x_1} )</td>
</tr>
<tr>
<td>( \frac{\partial v_3}{\partial x_1} )</td>
<td>( \frac{1}{v_0} \frac{\partial v_0}{\partial x_1} )</td>
</tr>
<tr>
<td>( \frac{\partial v_4}{\partial x_1} )</td>
<td>( \frac{1}{v_0} \frac{\partial v_0}{\partial x_1} )</td>
</tr>
</tbody>
</table>

---

**Product Rule**

| \( v_0 = x_1 \) | 2 |
| \( v_1 = x_2 \) | 5 |
| \( v_2 = \ln(v_0) \) | \( \ln(2) = 0.693 \) |
| \( v_3 = v_0 \cdot v_1 \) | 2 \times 5 = 10 |
| \( v_4 = \sin(v_1) \) | \( \sin(5) = 0.959 \) |
| \( v_5 = v_2 + v_3 \) | 0.693 + 10 = 10.693 |
| \( v_6 = v_5 - v_4 \) | 10.693 + 0.959 = 11.652 |
| \( y = v_6 \) | 11.652 |
AutoDiff - **Forward Mode**

\[ y = f(x_1, x_2) = \ln(x_1) + x_1 x_2 - \sin(x_2) \]

**Forward Derivative Trace:**

\[
\begin{align*}
\frac{\partial f(x_1, x_2)}{\partial x_1} &\bigg|_{(x_1=2, x_2=5)} \\
\frac{\partial v_0}{\partial x_1} & = 1 \\
\frac{\partial v_1}{\partial x_1} & = 0 \\
\frac{\partial v_2}{\partial x_1} & = \frac{1}{v_0} \frac{\partial v_0}{\partial x_1} \\
\frac{\partial v_3}{\partial x_1} & = \frac{\partial v_0}{\partial x_1} \cdot v_1 + v_0 \cdot \frac{\partial v_1}{\partial x_1} \\
\end{align*}
\]

**Product Rule**

\[
\begin{align*}
v_0 &= x_1 \\
v_1 &= x_2 \\
v_2 &= \ln(v_0) \quad \ln(2) = 0.693 \\
v_3 &= v_0 \cdot v_1 \quad 2 \times 5 = 10 \\
v_4 &= \sin(v_1) \quad \sin(5) = 0.959 \\
v_5 &= v_2 + v_3 \quad 0.693 + 10 = 10.693 \\
v_6 &= v_5 - v_4 \quad 10.693 + 0.959 = 11.652 \\
y &= v_6 \quad 11.652
\end{align*}
\]
AutoDiff - Forward Mode

Forward Evaluation Trace:

\[
\begin{align*}
  v_0 &= x_1 \\
  v_1 &= x_2 \\
  v_2 &= \ln(v_0) \\
  v_3 &= v_0 \cdot v_1 \\
  v_4 &= \sin(v_1) \\
  v_5 &= v_2 + v_3 \\
  v_6 &= v_5 - v_4 \\
  y &= v_6
\end{align*}
\]

\[
y = f(x_1, x_2) = \ln(x_1) + x_1 x_2 - \sin(x_2)
\]

Forward Derivative Trace:

\[
\begin{align*}
  \frac{\partial v_0}{\partial x_1} &= 1 \\
  \frac{\partial v_1}{\partial x_1} &= 0 \\
  \frac{\partial v_2}{\partial x_1} &= \frac{1}{v_0} \frac{\partial v_0}{\partial x_1} \\
  \frac{\partial v_3}{\partial x_1} &= \frac{\partial v_0}{\partial x_1} \cdot v_1 + v_0 \cdot \frac{\partial v_1}{\partial x_1} \\
  \frac{\partial f(x_1, x_2)}{\partial x_1} &= 1 \cdot 5 + 2 \cdot 0 = 5
\end{align*}
\]

Product Rule
AutoDiff - **Forward Mode**

\[ y = f(x_1, x_2) = \ln(x_1) + x_1 x_2 - \sin(x_2) \]

**Forward Derivative** Trace:

<table>
<thead>
<tr>
<th>( \frac{\partial f(x_1, x_2)}{\partial x_1} )</th>
<th>( (x_1=2, x_2=5) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \frac{\partial v_0}{\partial x_1} )</td>
<td>1</td>
</tr>
<tr>
<td>( \frac{\partial v_1}{\partial x_1} )</td>
<td>0</td>
</tr>
<tr>
<td>( \frac{\partial v_2}{\partial x_1} )</td>
<td>( \frac{1}{v_0} ) \cdot \frac{\partial v_0}{\partial x_1} ) = 0.5</td>
</tr>
<tr>
<td>( \frac{\partial v_3}{\partial x_1} )</td>
<td>( v_1 ) \cdot \frac{\partial v_1}{\partial x_1} \cdot \frac{\partial v_1}{\partial x_1} ) = 5</td>
</tr>
<tr>
<td>( \frac{\partial v_4}{\partial x_1} )</td>
<td>( \frac{\partial v_1}{\partial x_1} ) \cdot \cos(v_1) ) = 0</td>
</tr>
<tr>
<td>( \frac{\partial v_5}{\partial x_1} )</td>
<td>( \frac{\partial v_2}{\partial x_1} + \frac{\partial v_3}{\partial x_1} ) = 5.5</td>
</tr>
<tr>
<td>( \frac{\partial v_6}{\partial x_1} )</td>
<td>( \frac{\partial v_5}{\partial x_1} - \frac{\partial v_4}{\partial x_1} ) = 5.5</td>
</tr>
<tr>
<td>( \frac{\partial y}{\partial x_1} )</td>
<td>( \frac{\partial v_6}{\partial x_1} ) = 5.5</td>
</tr>
</tbody>
</table>

**Forward Evaluation** Trace:

| \( v_0 \) = x_1 | 2 |
| \( v_1 \) = x_2 | 5 |
| \( v_2 = \ln(v_0) \) | \( \ln(2) = 0.693 \) |
| \( v_3 = v_0 \cdot v_1 \) | \( 2 \times 5 = 10 \) |
| \( v_4 = \sin(v_1) \) | \( \sin(5) = 0.959 \) |
| \( v_5 = v_2 + v_3 \) | \( 0.693 + 10 = 10.693 \) |
| \( v_6 = v_5 - v_4 \) | \( 10.693 + 0.959 = 11.652 \) |
We now have:

\[
\left. \frac{\partial f(x_1, x_2)}{\partial x_1} \right|_{(x_1=2, x_2=5)} = 5.5
\]
AutoDiff - Forward Mode

\[ y = f(x_1, x_2) = \ln(x_1) + x_1 x_2 - \sin(x_2) \]

**Forward Derivative**

Trace:

<table>
<thead>
<tr>
<th>( \frac{\partial f(x_1, x_2)}{\partial x_1} )</th>
<th>( (x_1=2, x_2=5) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \frac{\partial v_0}{\partial x_1} )</td>
<td>1</td>
</tr>
<tr>
<td>( \frac{\partial v_1}{\partial x_1} )</td>
<td>0</td>
</tr>
<tr>
<td>( \frac{\partial v_2}{\partial x_1} )</td>
<td>( \frac{1}{v_0} \cdot \frac{\partial v_0}{\partial x_1} )</td>
</tr>
<tr>
<td>( \frac{\partial v_3}{\partial x_1} )</td>
<td>( v_0 \cdot \frac{\partial v_0}{\partial x_1} + v_0 \cdot \frac{\partial v_1}{\partial x_1} )</td>
</tr>
<tr>
<td>( \frac{\partial v_4}{\partial x_1} )</td>
<td>( \frac{\partial v_1}{\partial x_1} \cdot \cos(v_1) )</td>
</tr>
<tr>
<td>( \frac{\partial v_5}{\partial x_1} )</td>
<td>( \frac{\partial v_2}{\partial x_1} + \frac{\partial v_3}{\partial x_1} )</td>
</tr>
<tr>
<td>( \frac{\partial v_6}{\partial x_1} )</td>
<td>( \frac{\partial v_5}{\partial x_1} - \frac{\partial v_4}{\partial x_1} )</td>
</tr>
<tr>
<td>( \frac{\partial y}{\partial x_1} )</td>
<td>( \frac{\partial v_6}{\partial x_1} )</td>
</tr>
</tbody>
</table>

We now have:

\[ \frac{\partial f(x_1, x_2)}{\partial x_1} \bigg|_{(x_1=2, x_2=5)} = 5.5 \]

Still need:

\[ \frac{\partial f(x_1, x_2)}{\partial x_2} \bigg|_{(x_1=2, x_2=5)} \]
AutoDiff - **Forward Mode**

**Forward mode** needs $m$ forward passes to get a full Jacobian (all gradients of output with respect to each input), where $m$ is the number of inputs.

$$y = f(x) : \mathbb{R}^m \rightarrow \mathbb{R}^n$$

*slide adopted from T. Chen, H. Shen, A. Krishnamurthy CSE 599G1 lecture at UWashington*
AutoDiff - **Forward Mode**

**Forward mode** needs $m$ forward passes to get a full Jacobian (all gradients of output with respect to each input), where $m$ is the number of inputs:

$$y = f(x) : \mathbb{R}^m \rightarrow \mathbb{R}^n$$

**Problem:** DNN typically has large number of inputs:
- image as an input, plus all the weights and biases of layers = millions of inputs!

and very few outputs (many DNNs have $n = 1$)

*slide adopted from T. Chen, H. Shen, A. Krishnamurthy CSE 599G1 lecture at UWashington*
AutoDiff - **Forward Mode**

**Forward mode** needs \( m \) forward passes to get a full Jacobian (all gradients of output with respect to each input), where \( m \) is the number of inputs.

\[
y = f(x) : \mathbb{R}^m \rightarrow \mathbb{R}^n
\]

**Problem:** DNN typically has large number of inputs:
- image as an input, plus all the weights and biases of layers = millions of inputs!
- and very few outputs (many DNNs have \( n = 1 \))

Why?

*slide adopted from T. Chen, H. Shen, A. Krishnamurthy CSE 599G1 lecture at UWashington*
Forward mode needs $m$ forward passes to get a full Jacobian (all gradients of output with respect to each input), where $m$ is the number of inputs.

$$y = f(x) : \mathbb{R}^m \rightarrow \mathbb{R}^n$$

**Problem:** DNN typically has a large number of inputs:
- Image as an input, plus all the weights and biases of layers = millions of inputs!
- And very few outputs (many DNNs have $n = 1$)

Automatic differentiation in **reverse mode** computes all gradients in $n$ backwards passes (so for most DNNs in a single back pass — **back propagation**)

*slide adopted from T. Chen, H. Shen, A. Krishnamurthy CSE 599G1 lecture at UWashington*
AutoDiff - **Reverse Mode**

**Forward Evaluation** Trace:

<table>
<thead>
<tr>
<th>$v_i$</th>
<th>$f(2, 5)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$v_0 = x_1$</td>
<td>2</td>
</tr>
<tr>
<td>$v_1 = x_2$</td>
<td>5</td>
</tr>
<tr>
<td>$v_2 = \ln(v_0)$</td>
<td>$\ln(2) = 0.693$</td>
</tr>
<tr>
<td>$v_3 = v_0 \cdot v_1$</td>
<td>$2 \times 5 = 10$</td>
</tr>
<tr>
<td>$v_4 = \sin(v_1)$</td>
<td>$\sin(5) = 0.959$</td>
</tr>
<tr>
<td>$v_5 = v_2 + v_3$</td>
<td>$0.693 + 10 = 10.693$</td>
</tr>
<tr>
<td>$v_6 = v_5 - v_4$</td>
<td>$10.693 + 0.959 = 11.652$</td>
</tr>
<tr>
<td>$y = v_6$</td>
<td>11.652</td>
</tr>
</tbody>
</table>

Traverse the original graph in the reverse topological order and for each node in the original graph introduce an **adjoint node**, which computes derivative of the output with respect to the local node (using Chain rule):

$$
\bar{v}_i = \frac{\partial y_j}{\partial v_i} = \sum_{k \in \text{pa}(i)} \frac{\partial v_k}{\partial v_i} \frac{\partial y_j}{\partial v_k} = \sum_{k \in \text{pa}(i)} \frac{\partial v_k}{\partial v_i} \bar{v}_k
$$

"local" derivative
AutoDiff - Reverse Mode

Forward Evaluation Trace:

<table>
<thead>
<tr>
<th>( v_0 = x_1 )</th>
<th>( f(2, 5) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( v_1 = x_2 )</td>
<td>5</td>
</tr>
<tr>
<td>( v_2 = \ln(v_0) )</td>
<td>( \ln(2) = 0.693 )</td>
</tr>
<tr>
<td>( v_3 = v_0 \cdot v_1 )</td>
<td>2 x 5 = 10</td>
</tr>
<tr>
<td>( v_4 = \sin(v_1) )</td>
<td>( \sin(5) = 0.959 )</td>
</tr>
<tr>
<td>( v_5 = v_2 + v_3 )</td>
<td>0.693 + 10 = 10.693</td>
</tr>
<tr>
<td>( v_6 = v_5 - v_4 )</td>
<td>10.693 + 0.959 = 11.652</td>
</tr>
<tr>
<td>( y = v_6 )</td>
<td>11.652</td>
</tr>
</tbody>
</table>

Backwards Derivative Trace:

\[
\bar{v}_6 = \frac{\partial y}{\partial v_6}
\]
AutoDiff - Reverse Mode

Forward Evaluation Trace:

<table>
<thead>
<tr>
<th></th>
<th>$f(2, 5)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$v_0$</td>
<td>$x_1$</td>
</tr>
<tr>
<td>$v_1$</td>
<td>$x_2$</td>
</tr>
<tr>
<td>$v_2$</td>
<td>$\ln(v_0)$</td>
</tr>
<tr>
<td>$v_3$</td>
<td>$v_0 \cdot v_1$</td>
</tr>
<tr>
<td>$v_4$</td>
<td>$\sin(v_1)$</td>
</tr>
<tr>
<td>$v_5$</td>
<td>$v_2 + v_3$</td>
</tr>
<tr>
<td>$v_6$</td>
<td>$v_5 - v_4$</td>
</tr>
<tr>
<td>$y$</td>
<td>$v_6$</td>
</tr>
</tbody>
</table>

Backwards Derivative Trace:

$$\bar{v}_6 = \frac{\partial y}{\partial v_6}$$
AutoDiff - Reverse Mode

Forward Evaluation Trace:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Expression</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$v_0$</td>
<td>$x_1$</td>
<td>2</td>
</tr>
<tr>
<td>$v_1$</td>
<td>$x_2$</td>
<td>5</td>
</tr>
<tr>
<td>$v_2$</td>
<td>$\ln(v_0)$</td>
<td>$\ln(2) = 0.693$</td>
</tr>
<tr>
<td>$v_3$</td>
<td>$v_0 \cdot v_1$</td>
<td>2 x 5 = 10</td>
</tr>
<tr>
<td>$v_4$</td>
<td>$\sin(v_1)$</td>
<td>$\sin(5) = 0.959$</td>
</tr>
<tr>
<td>$v_5$</td>
<td>$v_2 + v_3$</td>
<td>0.693 + 10 = 10.693</td>
</tr>
<tr>
<td>$v_6$</td>
<td>$v_5 - v_4$</td>
<td>10.693 + 0.959 = 11.652</td>
</tr>
<tr>
<td>$y$</td>
<td>$v_6$</td>
<td>11.652</td>
</tr>
</tbody>
</table>

Backwards Derivative Trace:

$$\bar{v}_6 = \frac{\partial y}{\partial v_6}$$
AutoDiff - Reverse Mode

Forward Evaluation Trace:

<table>
<thead>
<tr>
<th>$v_0 = x_1$</th>
<th>$f(2, 5)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$v_1 = x_2$</td>
<td>2</td>
</tr>
<tr>
<td>$v_2 = \ln(v_0)$</td>
<td>5</td>
</tr>
<tr>
<td>$v_3 = v_0 \cdot v_1$</td>
<td>$\ln(2) = 0.693$</td>
</tr>
<tr>
<td>$v_4 = \sin(v_1)$</td>
<td>$2 \times 5 = 10$</td>
</tr>
<tr>
<td>$v_5 = v_2 + v_3$</td>
<td>$\sin(5) = 0.959$</td>
</tr>
<tr>
<td>$v_6 = v_5 - v_4$</td>
<td>$0.693 + 10 = 10.693$</td>
</tr>
<tr>
<td>$y = v_6$</td>
<td>$10.693 + 0.959 = 11.652$</td>
</tr>
</tbody>
</table>

Backwards Derivative Trace:

$\bar{v}_5 = \frac{\bar{v}_6}{\partial v_5}$

$\bar{v}_6 = \frac{\partial y}{\partial v_6}$
AutoDiff - Reverse Mode

Forward Evaluation Trace:

<table>
<thead>
<tr>
<th>$v_0 = x_1$</th>
<th>$v_1 = x_2$</th>
<th>$v_2 = \ln(v_0)$</th>
<th>$v_3 = v_0 \cdot v_1$</th>
<th>$v_4 = \sin(v_1)$</th>
<th>$v_5 = v_2 + v_3$</th>
<th>$v_6 = v_5 - v_4$</th>
<th>$y = v_6$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$v_0 = x_1$</td>
<td>$v_1 = x_2$</td>
<td>$v_2 = \ln(v_0)$</td>
<td>$v_3 = v_0 \cdot v_1$</td>
<td>$v_4 = \sin(v_1)$</td>
<td>$v_5 = v_2 + v_3$</td>
<td>$v_6 = v_5 - v_4$</td>
<td>$y = v_6$</td>
</tr>
</tbody>
</table>

$f(2, 5)$

- $v_0 = x_1$
- $v_1 = x_2$
- $v_2 = \ln(v_0)$
- $v_3 = v_0 \cdot v_1$
- $v_4 = \sin(v_1)$
- $v_5 = v_2 + v_3$
- $v_6 = v_5 - v_4$
- $y = v_6$

$f(2, 5) = 11.652$

$$f(2, 5) = \ln(2) \cdot \sin(5)$$

- $\ln(2) = 0.693$
- $\sin(5) = 0.959$

$$f(2, 5) = 0.693 \cdot 0.959 = 0.669$$

$$f(2, 5) = 11.652$$

Backwards Derivative Trace:

1. $\bar{v}_5 = \bar{v}_6 \frac{\partial v_6}{\partial v_5}$
2. $\bar{v}_6 = \frac{\partial y}{\partial v_6}$
AutoDiff - Reverse Mode

Forward Evaluation Trace:

<table>
<thead>
<tr>
<th>( v_0 )</th>
<th>( v_1 )</th>
<th>( v_2 )</th>
<th>( v_3 )</th>
<th>( v_4 )</th>
<th>( v_5 )</th>
<th>( v_6 )</th>
<th>( y )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x_1 )</td>
<td>( x_2 )</td>
<td>( \ln(v_0) )</td>
<td>( v_0 \cdot v_1 )</td>
<td>( \sin(v_1) )</td>
<td>( v_2 + v_3 )</td>
<td>( v_5 - v_4 )</td>
<td>( y )</td>
</tr>
<tr>
<td>( 2 )</td>
<td>( 5 )</td>
<td>( \ln(2) = 0.693 )</td>
<td>( 2 \times 5 = 10 )</td>
<td>( \sin(5) = 0.959 )</td>
<td>( 0.693 + 10 = 10.693 )</td>
<td>( 10.693 + 0.959 = 11.652 )</td>
<td>( 11.652 )</td>
</tr>
</tbody>
</table>

Backwards Derivative Trace:

\[
\frac{\partial v_5}{\partial v_6} = v_6\cdot 1
\]

\[
\frac{\partial v_6}{\partial v_6} = y
\]
AutoDiff - Reverse Mode

Forward Evaluation Trace:

| $v_0 = x_1$ | $v_0 = x_1$ |
| $v_1 = x_2$ | $v_1 = x_2$ |
| $v_2 = \ln(v_0)$ | $v_2 = \ln(v_0)$ |
| $v_3 = v_0 \cdot v_1$ | $v_3 = v_0 \cdot v_1$ |
| $v_4 = \sin(v_1)$ | $v_4 = \sin(v_1)$ |
| $v_5 = v_2 + v_3$ | $v_5 = v_2 + v_3$ |
| $v_6 = v_5 - v_4$ | $v_6 = v_5 - v_4$ |
| $y = v_6$ | $y = v_6$ |

<table>
<thead>
<tr>
<th>$f(2, 5)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>0.693 + 10 = 10.693</td>
</tr>
<tr>
<td>10.693 + 0.959 = 11.652</td>
</tr>
<tr>
<td>11.652</td>
</tr>
</tbody>
</table>

Backwards Derivative Trace:

$\overline{v}_5 = \overline{v}_6 \frac{\partial v_6}{\partial v_5} = \overline{v}_6 \cdot 1$

$\overline{v}_6 = \frac{\partial y}{\partial v_6}$

$1 \times 1 = 1$
AutoDiff - **Reverse Mode**

\[ x_1 \rightarrow v_0 \rightarrow v_2 \rightarrow v_5 \]
\[ x_2 \rightarrow v_1 \rightarrow v_4 \rightarrow v_6 \rightarrow y \]

**Forward Evaluation** Trace:

<table>
<thead>
<tr>
<th>( v_0 )</th>
<th>( v_1 )</th>
<th>( v_2 )</th>
<th>( v_3 )</th>
<th>( v_4 )</th>
<th>( v_5 )</th>
<th>( v_6 )</th>
<th>( y )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x_1 )</td>
<td>( x_2 )</td>
<td>( \ln(v_0) )</td>
<td>( v_0 \cdot v_1 )</td>
<td>( \sin(v_1) )</td>
<td>( v_2 + v_3 )</td>
<td>( v_5 - v_4 )</td>
<td>( v_6 )</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>( \ln(2) = 0.693 )</td>
<td>2 ( \times 5 = 10 )</td>
<td>( \sin(5) = 0.959 )</td>
<td>( 0.693 + 10 = 10.693 )</td>
<td>( 10.693 + 0.959 = 11.652 )</td>
<td>( 11.652 )</td>
</tr>
</tbody>
</table>

**Backwards Derivative** Trace:

\[ \bar{v}_4 = \bar{v}_6 \frac{\partial v_6}{\partial v_4} \]
\[ \bar{v}_5 = \bar{v}_6 \frac{\partial v_6}{\partial v_5} = \bar{v}_6 \cdot 1 \]
\[ \bar{v}_6 = \frac{\partial y}{\partial v_6} \]

\[ 1 \times 1 = 1 \]

\[ 1 \]
AutoDiff - **Reverse Mode**

\[ \begin{align*}
  v_0 &= x_1 \\
  v_1 &= x_2 \\
  v_2 &= \ln(v_0) \\
  v_3 &= v_0 \cdot v_1 \\
  v_4 &= \sin(v_1) \\
  v_5 &= v_2 + v_3 \\
  v_6 &= v_5 - v_4 \\
  y &= v_6
\end{align*} \]

**Forward Evaluation** Trace:

<table>
<thead>
<tr>
<th>( v_0 )</th>
<th>( \ln(v_0) )</th>
<th>( v_0 \cdot v_1 )</th>
<th>( \sin(v_1) )</th>
<th>( v_2 + v_3 )</th>
<th>( v_5 - v_4 )</th>
<th>( v_6 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x_1 )</td>
<td>( \ln(2) )</td>
<td>( 2 \times 5 = 10 )</td>
<td>( \sin(5) )</td>
<td>( 0.693 + 10 = 10.693 )</td>
<td>( 10.693 + 0.959 = 11.652 )</td>
<td>( 11.652 )</td>
</tr>
</tbody>
</table>

\[ f(2, 5) = 11.652 \]

**Backwards Derivative** Trace:

\[ \begin{align*}
  \frac{\partial f}{\partial v_6} &= 1 \\
  \frac{\partial f}{\partial v_5} &= \frac{\partial v_6}{\partial v_5} = v_6 \cdot 1 \\
  \frac{\partial f}{\partial v_4} &= \frac{\partial v_6}{\partial v_4} \\
  \frac{\partial f}{\partial v_3} &= \frac{\partial v_5}{\partial v_3} \\
  \frac{\partial f}{\partial v_2} &= \frac{\partial v_5}{\partial v_2} \\
  \frac{\partial f}{\partial v_1} &= \frac{\partial v_4}{\partial v_4} \\
  \frac{\partial f}{\partial v_0} &= \frac{\partial v_2}{\partial v_0} \\
  \frac{\partial f}{\partial x_1} &= \frac{\partial v_0}{\partial x_1} \\
  \frac{\partial f}{\partial x_2} &= \frac{\partial v_1}{\partial x_2}
\end{align*} \]
AutoDiff - Reverse Mode

Forward Evaluation Trace:

<table>
<thead>
<tr>
<th>v_0 = x_1</th>
<th>f(2, 5)</th>
</tr>
</thead>
<tbody>
<tr>
<td>v_1 = x_2</td>
<td></td>
</tr>
<tr>
<td>v_2 = ln(v_0)</td>
<td>2</td>
</tr>
<tr>
<td>v_3 = v_0 \cdot v_1</td>
<td>ln(2) = 0.693</td>
</tr>
<tr>
<td>v_4 = sin(v_1)</td>
<td>2 \times 5 = 10</td>
</tr>
<tr>
<td>v_5 = v_2 + v_3</td>
<td>sin(5) = 0.959</td>
</tr>
<tr>
<td>v_6 = v_5 - v_4</td>
<td>0.693 + 10 = 10.693</td>
</tr>
<tr>
<td>y = v_6</td>
<td>10.693 + 0.959 = 11.652</td>
</tr>
</tbody>
</table>

Backwards Derivative Trace:

\[ \bar{v}_4 = \bar{v}_6 \frac{\partial v_6}{\partial v_4} = \bar{v}_6 \cdot (-1) \]
\[ \bar{v}_5 = \bar{v}_6 \frac{\partial v_6}{\partial v_5} = \bar{v}_6 \cdot 1 \]
\[ \bar{v}_6 = \frac{\partial y}{\partial v_6} = 1 \]
AutoDiff - Reverse Mode

Forward Evaluation Trace:

<table>
<thead>
<tr>
<th>v_0 = x_1</th>
<th>f(2, 5)</th>
</tr>
</thead>
<tbody>
<tr>
<td>v_1 = x_2</td>
<td>2</td>
</tr>
<tr>
<td>v_2 = ln(v_0)</td>
<td>ln(2) = 0.693</td>
</tr>
<tr>
<td>v_3 = v_0 \cdot v_1</td>
<td>2 \times 5 = 10</td>
</tr>
<tr>
<td>v_4 = sin(v_1)</td>
<td>sin(5) = 0.959</td>
</tr>
<tr>
<td>v_5 = v_2 + v_3</td>
<td>0.693 + 10 = 10.693</td>
</tr>
<tr>
<td>v_6 = v_5 - v_4</td>
<td>10.693 + 0.959 = 11.652</td>
</tr>
<tr>
<td>y = v_6</td>
<td>11.652</td>
</tr>
</tbody>
</table>

Backwards Derivative Trace:

\[
\begin{align*}
\bar{v}_4 &= \bar{v}_6 \frac{\partial v_6}{\partial v_4} = \bar{v}_6 \cdot (-1) \\
\bar{v}_5 &= \bar{v}_6 \frac{\partial v_6}{\partial v_5} = \bar{v}_6 \cdot 1 \\
\bar{v}_6 &= \frac{\partial y}{\partial v_6} \quad 1x-1 = -1 \\
\end{align*}
\]
## AutoDiff - Reverse Mode

**Forward Evaluation** Trace:

<table>
<thead>
<tr>
<th></th>
<th>$f(2,5)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$v_0 = x_1$</td>
<td>2</td>
</tr>
<tr>
<td>$v_1 = x_2$</td>
<td>5</td>
</tr>
<tr>
<td>$v_2 = \ln(v_0)$</td>
<td>$\ln(2) = 0.693$</td>
</tr>
<tr>
<td>$v_3 = v_0 \cdot v_1$</td>
<td>2 x 5 = 10</td>
</tr>
<tr>
<td>$v_4 = \sin(v_1)$</td>
<td>$\sin(5) = 0.959$</td>
</tr>
<tr>
<td>$v_5 = v_2 + v_3$</td>
<td>0.693 + 10 = 10.693</td>
</tr>
<tr>
<td>$v_6 = v_5 - v_4$</td>
<td>10.693 + 0.959 = 11.652</td>
</tr>
<tr>
<td>$y = v_6$</td>
<td>11.652</td>
</tr>
</tbody>
</table>

**Backwards Derivative** Trace:

\[
\begin{align*}
\bar{v}_3 &= \bar{v}_5 \frac{\partial v_5}{\partial v_3} \\
\bar{v}_4 &= \bar{v}_6 \frac{\partial v_6}{\partial v_4} = \bar{v}_6 \cdot (-1) \\
\bar{v}_5 &= \bar{v}_6 \frac{\partial v_6}{\partial v_5} = \bar{v}_6 \cdot 1 \\
\bar{v}_6 &= \frac{\partial y}{\partial v_6} 
\end{align*}
\]

1x1 = 1  
1x-1 = -1
AutoDiff - Reverse Mode

Forward Evaluation Trace:

| v0 = x1 | 2 |
| v1 = x2 | 5 |
| v2 = ln(v0) | ln(2) = 0.693 |
| v3 = v0 · v1 | 2 x 5 = 10 |
| v4 = sin(v1) | sin(5) = 0.959 |
| v5 = v2 + v3 | 0.693 + 10 = 10.693 |
| v6 = v5 - v4 | 10.693 + 0.959 = 11.652 |
| y = v6 | 11.652 |

Backwards Derivative Trace:

\[ \bar{v}_3 = \bar{v}_5 \frac{\partial v_5}{\partial v_3} \]
\[ \bar{v}_4 = \bar{v}_6 \frac{\partial v_6}{\partial v_4} = \bar{v}_6 \cdot (-1) \]
\[ 1x-1 = -1 \]
\[ \bar{v}_5 = \bar{v}_6 \frac{\partial v_6}{\partial v_5} = \bar{v}_6 \cdot 1 \]
\[ 1x1 = 1 \]
\[ \bar{v}_6 = \frac{\partial y}{\partial v_6} \]
\[ 1 \]
**AutoDiff - Reverse Mode**

**Forward Evaluation** Trace:

<table>
<thead>
<tr>
<th></th>
<th>( f(2, 5) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( v_0 = x_1 )</td>
<td>2</td>
</tr>
<tr>
<td>( v_1 = x_2 )</td>
<td>5</td>
</tr>
<tr>
<td>( v_2 = \ln(v_0) )</td>
<td>( \ln(2) = 0.693 )</td>
</tr>
<tr>
<td>( v_3 = v_0 \cdot v_1 )</td>
<td>2 \times 5 = 10</td>
</tr>
<tr>
<td>( v_4 = \sin(v_1) )</td>
<td>( \sin(5) = 0.959 )</td>
</tr>
<tr>
<td>( v_5 = v_2 + v_3 )</td>
<td>0.693 + 10 = 10.693</td>
</tr>
<tr>
<td>( v_6 = v_5 - v_4 )</td>
<td>10.693 + 0.959 = 11.652</td>
</tr>
<tr>
<td>( y = v_6 )</td>
<td>11.652</td>
</tr>
</tbody>
</table>

**Backwards Derivative** Trace:

\[
\begin{align*}
\bar{v}_3 &= \bar{v}_5 \frac{\partial v_5}{\partial v_3} = \bar{v}_5 \cdot 1 \\
\bar{v}_4 &= \bar{v}_6 \frac{\partial v_6}{\partial v_4} = \bar{v}_6 \cdot 0 \\
\bar{v}_5 &= \bar{v}_6 \frac{\partial v_6}{\partial v_5} = \bar{v}_6 \cdot 1 \\
\bar{v}_6 &= \frac{\partial y}{\partial v_6} \\
\end{align*}
\]
AutoDiff - **Reverse Mode**

**Forward Evaluation** Trace:

<table>
<thead>
<tr>
<th>$v_0 = x_1$</th>
<th>$f(2, 5)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$v_1 = x_2$</td>
<td>2</td>
</tr>
<tr>
<td>$v_2 = \ln(v_0)$</td>
<td>5</td>
</tr>
<tr>
<td>$v_3 = v_0 \cdot v_1$</td>
<td>ln(2) = 0.693</td>
</tr>
<tr>
<td>$v_4 = \sin(v_1)$</td>
<td>2 x 5 = 10</td>
</tr>
<tr>
<td>$v_5 = v_2 + v_3$</td>
<td>sin(5) = 0.959</td>
</tr>
<tr>
<td>$v_6 = v_5 - v_4$</td>
<td>0.693 + 10 = 10.693</td>
</tr>
<tr>
<td>$y = v_6$</td>
<td>10.693 + 0.959 = 11.652</td>
</tr>
</tbody>
</table>

**Backwards Derivative** Trace:

$\bar{v}_3 = \bar{v}_5 \frac{\partial v_5}{\partial v_3} = \bar{v}_5 \cdot 1$

$\bar{v}_4 = \bar{v}_6 \frac{\partial v_6}{\partial v_4} = \bar{v}_6 \cdot (-1)$

$\bar{v}_5 = \bar{v}_6 \frac{\partial v_6}{\partial v_5} = \bar{v}_6 \cdot 1$

$\bar{v}_6 = \frac{\partial y}{\partial v_6}$

$1 \times 1 = 1$

$1 \times -1 = -1$

$1 \times 1 = 1$

$1$
**AutoDiff - Reverse Mode**

**Forward Evaluation**

<table>
<thead>
<tr>
<th>Trace</th>
<th>$f(2, 5)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$v_0 = x_1$</td>
<td>2</td>
</tr>
<tr>
<td>$v_1 = x_2$</td>
<td>5</td>
</tr>
<tr>
<td>$v_2 = \ln(v_0)$</td>
<td>$\ln(2) = 0.693$</td>
</tr>
<tr>
<td>$v_3 = v_0 \cdot v_1$</td>
<td>$2 \times 5 = 10$</td>
</tr>
<tr>
<td>$v_4 = \sin(v_1)$</td>
<td>$\sin(5) = 0.959$</td>
</tr>
<tr>
<td>$v_5 = v_2 + v_3$</td>
<td>$0.693 + 10 = 10.693$</td>
</tr>
<tr>
<td>$v_6 = v_5 - v_4$</td>
<td>$10.693 + 0.959 = 11.652$</td>
</tr>
<tr>
<td>$y = v_6$</td>
<td>11.652</td>
</tr>
</tbody>
</table>

**Backwards Derivative**

Trace:

\[
\begin{align*}
\bar{v}_2 &= \frac{\partial v_5}{\partial v_2} \\
\bar{v}_3 &= \frac{\partial v_5}{\partial v_3} = v_5 \cdot (1) \\
\bar{v}_4 &= \frac{\partial v_6}{\partial v_4} = \bar{v}_6 \cdot (-1) \\
\bar{v}_5 &= \frac{\partial v_6}{\partial v_5} = \bar{v}_6 \cdot 1 \\
\bar{v}_6 &= \frac{\partial y}{\partial v_6}
\end{align*}
\]

- $1 \times 1 = 1$
- $1 \times -1 = -1$
- $1 \times 1 = 1$
- $1$
### Forward Evaluation

<table>
<thead>
<tr>
<th></th>
<th>( f(2, 5) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( v_0 = x_1 )</td>
<td>2</td>
</tr>
<tr>
<td>( v_1 = x_2 )</td>
<td>5</td>
</tr>
<tr>
<td>( v_2 = \ln(v_0) )</td>
<td>( \ln(2) = 0.693 )</td>
</tr>
<tr>
<td>( v_3 = v_0 \cdot v_1 )</td>
<td>( 2 \times 5 = 10 )</td>
</tr>
<tr>
<td>( v_4 = \sin(v_1) )</td>
<td>( \sin(5) = 0.959 )</td>
</tr>
<tr>
<td>( v_5 = v_2 + v_3 )</td>
<td>0.693 + 10 = 10.693</td>
</tr>
<tr>
<td>( v_6 = v_5 - v_4 )</td>
<td>10.693 + 0.959 = 11.652</td>
</tr>
<tr>
<td>( y = v_6 )</td>
<td>11.652</td>
</tr>
</tbody>
</table>

### Backwards Derivative

\[ v_2 = \frac{\partial v_5}{\partial v_2} \]
\[ v_3 = \frac{\partial v_5}{\partial v_3} = v_5 \cdot (1) \]
\[ v_4 = \frac{\partial v_6}{\partial v_4} = \bar{v}_6 \cdot (-1) \]
\[ v_5 = \frac{\partial y}{\partial v_5} = \frac{\partial y}{\partial v_6} = \bar{v}_6 \cdot 1 \]
\[ \bar{v}_6 = \frac{\partial y}{\partial v_6} \]
AutoDiff - Reverse Mode

Forward Evaluation Trace:

<table>
<thead>
<tr>
<th></th>
<th>( f(2, 5) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( v_0 = x_1 )</td>
<td>2</td>
</tr>
<tr>
<td>( v_1 = x_2 )</td>
<td>5</td>
</tr>
<tr>
<td>( v_2 = \ln(v_0) )</td>
<td>( \ln(2) = 0.693 )</td>
</tr>
<tr>
<td>( v_3 = v_0 \cdot v_1 )</td>
<td>2 \times 5 = 10</td>
</tr>
<tr>
<td>( v_4 = \sin(v_1) )</td>
<td>( \sin(5) = 0.959 )</td>
</tr>
<tr>
<td>( v_5 = v_2 + v_3 )</td>
<td>0.693 + 10 = 10.693</td>
</tr>
<tr>
<td>( v_6 = v_5 - v_4 )</td>
<td>10.693 + 0.959 = 11.652</td>
</tr>
<tr>
<td>( y = v_6 )</td>
<td>11.652</td>
</tr>
</tbody>
</table>

Backwards Derivative Trace:

\[
\bar{v}_2 = \bar{v}_5 \frac{\partial v_5}{\partial v_2} = \bar{v}_5 \cdot (1)
\]

\[
\bar{v}_3 = \bar{v}_5 \frac{\partial v_5}{\partial v_3} = \bar{v}_5 \cdot (1)
\]

\[
\bar{v}_4 = \bar{v}_6 \frac{\partial v_6}{\partial v_4} = \bar{v}_6 \cdot (-1)
\]

\[
\bar{v}_5 = \bar{v}_6 \frac{\partial v_6}{\partial v_5} = \bar{v}_6 \cdot 1
\]

\[
\bar{v}_6 = \frac{\partial y}{\partial v_6}
\]

\( 1 \times -1 = -1 \)

\( 1 \times 1 = 1 \)

\( 1 \)

\( 1 \times 1 = 1 \)
AutoDiff - Reverse Mode

Forward Evaluation Trace:

<table>
<thead>
<tr>
<th></th>
<th>$f(2, 5)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$v_0 = x_1$</td>
<td>2</td>
</tr>
<tr>
<td>$v_1 = x_2$</td>
<td>5</td>
</tr>
<tr>
<td>$v_2 = \ln(v_0)$</td>
<td>$\ln(2) = 0.693$</td>
</tr>
<tr>
<td>$v_3 = v_0 \cdot v_1$</td>
<td>2 x 5 = 10</td>
</tr>
<tr>
<td>$v_4 = \sin(v_1)$</td>
<td>$\sin(5) = 0.959$</td>
</tr>
<tr>
<td>$v_5 = v_2 + v_3$</td>
<td>0.693 + 10 = 10.693</td>
</tr>
<tr>
<td>$v_6 = v_5 - v_4$</td>
<td>10.693 + 0.959 = 11.652</td>
</tr>
<tr>
<td>$y = v_6$</td>
<td>11.652</td>
</tr>
</tbody>
</table>

Backwards Derivative Trace:

$\frac{\partial v_2}{\partial v_5} = v_5 \cdot (1)$

$\frac{\partial v_3}{\partial v_5} = v_5 \cdot (1)$

$\frac{\partial v_4}{\partial v_5} = v_6 \cdot (-1)$

$\frac{\partial v_5}{\partial v_6} = v_6 \cdot 1$

$\frac{\partial y}{\partial v_6}$

$1x1 = 1$

$1x1 = 1$

$1x-1 = -1$

$1x1 = 1$

$1$
AutoDiff - Reverse Mode

Forward Evaluation Trace:

<table>
<thead>
<tr>
<th>( v_0 = x_1 )</th>
<th>( f(2, 5) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( v_1 = x_2 )</td>
<td>2</td>
</tr>
<tr>
<td>( v_2 = \ln(v_0) )</td>
<td>5</td>
</tr>
<tr>
<td>( v_3 = v_0 \cdot v_1 )</td>
<td>( \ln(2) = 0.693 )</td>
</tr>
<tr>
<td>( v_4 = \sin(v_1) )</td>
<td>2 \times 5 = 10</td>
</tr>
<tr>
<td>( v_5 = v_2 + v_3 )</td>
<td>0.693 + 10 = 10.693</td>
</tr>
<tr>
<td>( v_6 = v_5 - v_4 )</td>
<td>10.693 + 0.959 = 11.652</td>
</tr>
<tr>
<td>( y = v_6 )</td>
<td>11.652</td>
</tr>
</tbody>
</table>

Backwards Derivative Trace:

\[ \tilde{v}_1 : \]
\[ \tilde{v}_2 = \frac{\partial v_5}{\partial v_2} = \tilde{v}_5 \cdot (1) \]
\[ 1 \times 1 = 1 \]

\[ \tilde{v}_3 = \frac{\partial v_5}{\partial v_3} = \tilde{v}_5 \cdot (1) \]
\[ 1 \times 1 = 1 \]

\[ \tilde{v}_4 = \frac{\partial v_6}{\partial v_4} = \tilde{v}_6 \cdot (-1) \]
\[ 1 \times -1 = -1 \]

\[ \tilde{v}_5 = \frac{\partial v_6}{\partial v_5} = \tilde{v}_6 \cdot 1 \]
\[ 1 \times 1 = 1 \]

\[ \tilde{v}_6 = \frac{\partial y}{\partial v_6} \]
\[ 1 \]
**AutoDiff - Reverse Mode**

**Forward Evaluation** Trace:

<table>
<thead>
<tr>
<th>$v_0 = x_1$</th>
<th>$f(2, 5)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$v_1 = x_2$</td>
<td>2</td>
</tr>
<tr>
<td>$v_2 = \ln(v_0)$</td>
<td>5</td>
</tr>
<tr>
<td>$v_3 = v_0 \cdot v_1$</td>
<td>$\ln(2) = 0.693$</td>
</tr>
<tr>
<td>$v_4 = \sin(v_1)$</td>
<td>$2 \cdot 5 = 10$</td>
</tr>
<tr>
<td>$v_5 = v_2 + v_3$</td>
<td>$\sin(5) = 0.959$</td>
</tr>
<tr>
<td>$v_6 = v_5 - v_4$</td>
<td>$0.693 + 10 = 10.693$</td>
</tr>
<tr>
<td>$y = v_6$</td>
<td>$10.693 + 0.959 = 11.652$</td>
</tr>
</tbody>
</table>

**Backwards Derivative** Trace:

| $\bar{v}_1 = \bar{v}_3 \frac{\partial v_3}{\partial v_1} + \bar{v}_4 \frac{\partial v_4}{\partial v_1}$ | $1 \times 1 = 1$ |
| $\bar{v}_2 = \bar{v}_5 \frac{\partial v_5}{\partial v_2}$ | $1 \times 1 = 1$ |
| $\bar{v}_3 = \bar{v}_5 \frac{\partial v_5}{\partial v_3}$ | $1 \times 1 = 1$ |
| $\bar{v}_4 = \bar{v}_6 \frac{\partial v_6}{\partial v_4}$ | $1 \times -1 = -1$ |
| $\bar{v}_5 = \bar{v}_6 \frac{\partial v_6}{\partial v_5}$ | $1 \times 1 = 1$ |
| $\bar{v}_6 = \frac{\partial y}{\partial v_6}$ | $1$ |
AutoDiff - Reverse Mode

Forward Evaluation Trace:

<table>
<thead>
<tr>
<th>$v_0 = x_1$</th>
<th>$v_1 = x_2$</th>
<th>$v_2 = \ln(v_0)$</th>
<th>$v_3 = v_0 \cdot v_1$</th>
<th>$v_4 = \sin(v_1)$</th>
<th>$v_5 = v_2 + v_3$</th>
<th>$v_6 = v_5 - v_4$</th>
<th>$y = v_6$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$</td>
<td>$x_2$</td>
<td>$\ln(2.5) = 0.693$</td>
<td>$2 \times 5 = 10$</td>
<td>$\sin(5) = 0.959$</td>
<td>$0.693 + 10 = 10.693$</td>
<td>$10.693 + 0.959 = 11.652$</td>
<td>$11.652$</td>
</tr>
</tbody>
</table>

$f(2, 5) = 2$ 

Backwards Derivative Trace:

$\bar{v}_1 = \bar{v}_3 \frac{\partial v_3}{\partial v_1} + \bar{v}_4 \frac{\partial v_4}{\partial v_1}$

$\bar{v}_2 = \bar{v}_5 \frac{\partial v_5}{\partial v_2} = \bar{v}_5 \cdot (1)$

$\bar{v}_3 = \bar{v}_5 \frac{\partial v_5}{\partial v_3} = \bar{v}_5 \cdot (1)$

$\bar{v}_4 = \bar{v}_6 \frac{\partial v_6}{\partial v_4} = \bar{v}_6 \cdot (-1)$

$\bar{v}_5 = \bar{v}_6 \frac{\partial v_6}{\partial v_5} = \bar{v}_6 \cdot (1)$

$\bar{v}_6 = \frac{\partial y}{\partial v_6}$
AutoDiff - Reverse Mode

**Forward Evaluation** Trace:

<table>
<thead>
<tr>
<th></th>
<th>$f(2, 5)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$v_0 = x_1$</td>
<td>2</td>
</tr>
<tr>
<td>$v_1 = x_2$</td>
<td>5</td>
</tr>
<tr>
<td>$v_2 = \ln(v_0)$</td>
<td>$\ln(2) = 0.693$</td>
</tr>
<tr>
<td>$v_3 = v_0 \cdot v_1$</td>
<td>2 x 5 = 10</td>
</tr>
<tr>
<td>$v_4 = \sin(v_1)$</td>
<td>$\sin(5) = 0.959$</td>
</tr>
<tr>
<td>$v_5 = v_2 + v_3$</td>
<td>0.693 + 10 = 10.693</td>
</tr>
<tr>
<td>$v_6 = v_5 - v_4$</td>
<td>10.693 + 0.959 = 11.652</td>
</tr>
<tr>
<td>$y = v_6$</td>
<td>11.652</td>
</tr>
</tbody>
</table>

**Backwards Derivative** Trace:

\[
\bar{y} = \bar{v}_6 \frac{\partial y}{\partial v_6} = \bar{v}_6 \cdot 1
\]

\[
\bar{v}_5 = \bar{v}_6 \frac{\partial v_6}{\partial v_5} = \bar{v}_6 \cdot (-1)
\]

\[
\bar{v}_4 = \bar{v}_6 \frac{\partial v_6}{\partial v_4} = \bar{v}_6 \cdot (1)
\]

\[
\bar{v}_3 = \bar{v}_5 \frac{\partial v_5}{\partial v_3} = \bar{v}_5 \cdot (1)
\]

\[
\bar{v}_2 = \bar{v}_5 \frac{\partial v_5}{\partial v_2} = \bar{v}_5 \cdot (1)
\]

\[
\bar{v}_1 = \bar{v}_3 \frac{\partial v_3}{\partial v_1} + \bar{v}_4 \frac{\partial v_4}{\partial v_1} = \bar{v}_3 v_0 + \bar{v}_4 \cos(v_1)
\]

\[
x_1 \rightarrow v_0 \rightarrow v_2 \rightarrow v_5
\]

\[
x_2 \rightarrow v_1 \rightarrow v_4 \rightarrow -v_6 \rightarrow y
\]
**AutoDiff - Reverse Mode**

**Forward Evaluation Trace:**

\[
\begin{array}{|c|c|}
\hline
v_0 &= x_1 & 2 \\
v_1 &= x_2 & 5 \\
v_2 &= \ln(v_0) & \ln(2) = 0.693 \\
v_3 &= v_0 \cdot v_1 & 2 \times 5 = 10 \\
v_4 &= \sin(v_1) & \sin(5) = 0.959 \\
v_5 &= v_2 + v_3 & 0.693 + 10 = 10.693 \\
v_6 &= v_5 - v_4 & 10.693 + 0.959 = 11.652 \\
y &= v_6 & 11.652 \\
\hline
\end{array}
\]

**Backwards Derivative Trace:**

\[
\begin{align*}
\bar{v}_1 &= \bar{v}_3 \frac{\partial v_3}{\partial v_1} + \bar{v}_4 \frac{\partial v_4}{\partial v_1} = \bar{v}_3 v_0 + \bar{v}_4 \cos(v_1) & 1.716 \\
\bar{v}_2 &= \bar{v}_5 \frac{\partial v_5}{\partial v_2} = \bar{v}_5 \cdot (1) & 1 \times 1 = 1 \\
\bar{v}_3 &= \bar{v}_5 \frac{\partial v_5}{\partial v_3} = \bar{v}_5 \cdot (1) & 1 \times 1 = 1 \\
\bar{v}_4 &= \bar{v}_6 \frac{\partial v_6}{\partial v_4} = \bar{v}_6 \cdot (-1) & 1 \times -1 = -1 \\
\bar{v}_5 &= \bar{v}_6 \frac{\partial v_6}{\partial v_5} = \bar{v}_6 \cdot 1 & 1 \times 1 = 1 \\
\bar{v}_6 &= \frac{\partial y}{\partial v_6} & 1
\end{align*}
\]
AutoDiff - Reverse Mode

Forward Evaluation Trace:

<table>
<thead>
<tr>
<th>$v_0 = x_1$</th>
<th>$f(2, 5)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$v_1 = x_2$</td>
<td>2</td>
</tr>
<tr>
<td>$v_2 = \ln(v_0)$</td>
<td>$\ln(2) = 0.693$</td>
</tr>
<tr>
<td>$v_3 = v_0 \cdot v_1$</td>
<td>2 x 5 = 10</td>
</tr>
<tr>
<td>$v_4 = \sin(v_1)$</td>
<td>$\sin(5) = 0.959$</td>
</tr>
<tr>
<td>$v_5 = v_2 + v_3$</td>
<td>$0.693 + 10 = 10.693$</td>
</tr>
<tr>
<td>$v_6 = v_5 - v_4$</td>
<td>$10.693 + 0.959 = 11.652$</td>
</tr>
<tr>
<td>$y = v_6$</td>
<td>11.652</td>
</tr>
</tbody>
</table>

Backwards Derivative Trace:

$$
\begin{align*}
\bar{v}_0 &= \bar{v}_3 \frac{\partial v_3}{\partial v_0} + \bar{v}_2 \frac{\partial v_2}{\partial v_0} = \bar{v}_3 v_1 + \bar{v}_2 \frac{1}{v_0} \\
\bar{v}_1 &= \bar{v}_3 \frac{\partial v_3}{\partial v_1} + \bar{v}_4 \frac{\partial v_4}{\partial v_1} = \bar{v}_3 v_0 + \bar{v}_4 \cos(v_1) \\
\bar{v}_2 &= \bar{v}_5 \frac{\partial v_5}{\partial v_2} = \bar{v}_5 \cdot (1) \\
\bar{v}_3 &= \bar{v}_5 \frac{\partial v_5}{\partial v_3} = \bar{v}_5 \cdot (1) \\
\bar{v}_4 &= \bar{v}_6 \frac{\partial v_6}{\partial v_4} = \bar{v}_6 \cdot (-1) \\
\bar{v}_5 &= \bar{v}_6 \frac{\partial v_6}{\partial v_5} = \bar{v}_6 \cdot 1 \\
\bar{v}_6 &= \frac{\partial y}{\partial v_6} = 1
\end{align*}
$$
AutoDiff - Reverse Mode

Forward Evaluation Trace:

<table>
<thead>
<tr>
<th></th>
<th>$f(2,5)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$v_0 = x_1$</td>
<td>2</td>
</tr>
<tr>
<td>$v_1 = x_2$</td>
<td>5</td>
</tr>
<tr>
<td>$v_2 = \ln(v_0)$</td>
<td>$\ln(2) = 0.693$</td>
</tr>
<tr>
<td>$v_3 = v_0 \cdot v_1$</td>
<td>2 x 5 = 10</td>
</tr>
<tr>
<td>$v_4 = \sin(v_1)$</td>
<td>$\sin(5) = 0.959$</td>
</tr>
<tr>
<td>$v_5 = v_2 + v_3$</td>
<td>0.693 + 10 = 10.693</td>
</tr>
<tr>
<td>$v_6 = v_5 - v_4$</td>
<td>10.693 + 0.959 = 11.652</td>
</tr>
<tr>
<td>$y = v_6$</td>
<td>11.652</td>
</tr>
</tbody>
</table>

Backwards Derivative Trace:

$\frac{\partial y}{\partial v_6} = \frac{\partial y}{\partial v_5} \frac{\partial v_5}{\partial v_6}$

\[\begin{align*}
\bar{v}_0 &= \bar{v}_3 \frac{\partial v_3}{\partial v_0} + \bar{v}_2 \frac{\partial v_2}{\partial v_0} = \bar{v}_3 v_1 + \bar{v}_2 \frac{1}{v_0} \\
\bar{v}_1 &= \bar{v}_3 \frac{\partial v_3}{\partial v_1} + \bar{v}_4 \frac{\partial v_4}{\partial v_1} = \bar{v}_3 v_0 + \bar{v}_4 \cos(v_1) \\
\bar{v}_2 &= \bar{v}_5 \frac{\partial v_5}{\partial v_2} = \bar{v}_5 \cdot (1) \\
\bar{v}_3 &= \bar{v}_5 \frac{\partial v_5}{\partial v_3} = \bar{v}_5 \cdot (1) \\
\bar{v}_4 &= \bar{v}_6 \frac{\partial v_6}{\partial v_4} = \bar{v}_6 \cdot (-1) \\
\bar{v}_5 &= \bar{v}_6 \frac{\partial v_6}{\partial v_5} = \bar{v}_6 \cdot 1 \\
\bar{v}_6 &= \frac{\partial y}{\partial v_6} \\
\end{align*}\]
Automatic Differentiation (AutoDiff)

AutoDiff can be done at various **granularities**

**Elementary function** granularity:

\[
y = f(x_1, x_2) = \ln(x_1) + x_1x_2 - \sin(x_2)
\]

**Complex function** granularity:
Backpropagation Practical Issues

Easier to deal with in vector form
$y = f(W, b, x) = \text{sigmoid}(W \cdot x + b)$
Backpropagation Practical Issues

\[ y = f(W, b, x) = \text{sigmoid}(W \cdot x + b) \]

\[ \frac{\partial L}{\partial x} = \frac{\partial y}{\partial x} \frac{\partial L}{\partial y} \]

\[ \frac{\partial L}{\partial W} = \frac{\partial y}{\partial W} \frac{\partial L}{\partial y} \]

\[ \frac{\partial L}{\partial b} = \frac{\partial y}{\partial b} \frac{\partial L}{\partial y} \]
Jacobian of Sigmoid layer

Element-wise sigmoid layer:

\[ x, y \in \mathbb{R}^{2048} \]
Jacobian of Sigmoid layer

Element-wise sigmoid layer:

$$x, y \in \mathbb{R}^{2048}$$

What is the dimension of **Jacobian**?
Jacobian of Sigmoid layer

Element-wise sigmoid layer:

$x, y \in \mathbb{R}^{2048}$

What is the dimension of Jacobian?

What does it look like?
**Jacobian** of Sigmoid layer

Element-wise sigmoid layer:

\[
\begin{array}{c}
\text{x} \\
\text{sigmoid} \\
\text{y}
\end{array}
\]

What is the dimension of **Jacobian**?

What does it look like?

If we are working with a mini batch of 100 inputs-output pairs, technically Jacobian is a matrix 204,800 x 204,800
Backpropagation: Common questions

**Question:** Does BackProp only work for certain layers?

**Answer:** No, for any differentiable functions

**Question:** What is computational cost of BackProp?

**Answer:** On average about twice the forward pass

**Question:** Is BackProp a dual of forward propagation?

**Answer:** Yes

* Adopted from slides by Marc’Aurelio Ranzato*
**Backpropagation: Common questions**

**Question:** Does BackProp only work for certain layers?

**Answer:** No, for any differentiable functions

**Question:** What is computational cost of BackProp?

**Answer:** On average about twice the forward pass

**Question:** Is BackProp a dual of forward propagation?

**Answer:** Yes

* Adopted from slides by Marc'Aurelio Ranzato
Activation Function: Sigmoid

\[ a(x) = \text{sigmoid}(x) = \frac{1}{1 + e^{-x}} \]
Computational Graph: 1-layer network

\[ y_i \]

\[ x_i \]

\[ b \]

\[ W \]

\[ o = W \cdot x + b \]

\[ a = \text{sigmoid}(o) \]

\[ l = \text{MSE}_{loss}(\hat{y}, y) \]
**Activation Function:** Sigmoid

\[ a(x) = \text{sigmoid}(x) = \frac{1}{1 + e^{-x}} \]
Activation Function: Sigmoid

**Pros:**
- Squishes everything in the range [0,1]
- Can be interpreted as “probability”
- Has well defined gradient everywhere

**Cons:**
- Saturated neurons “kill” the gradients
- Non-zero centered
- Could be expensive to compute

\[
a(x) = \text{sigmoid}(x) = \frac{1}{1 + e^{-x}}
\]
Activation Function: Sigmoid

\[ a(x) = \text{sigmoid}(x) = \frac{1}{1 + e^{-x}} \]

Cons:
- Saturated neurons "kill" the gradients
- Non-zero centered
- Could be expensive to compute

* slide adopted from Li, Karpathy, Johnson’s CS231n at Stanford
Activation Function: Sigmoid

\[ a = \text{sigmoid}(x) = \frac{1}{1 + e^{-x}} \]

Cons:
- Saturated neurons "kill" the gradients
- Non-zero centered
- Could be expensive to compute

*S slide adopted from Li, Karpathy, Johnson’s CS231n at Stanford*
Activation Function: Sigmoid

\[ a = \operatorname{sigmoid}(x) = \frac{1}{1 + e^{-x}} \]

Cons:
- Saturated neurons “kill” the gradients
- Non-zero centered
- Could be expensive to compute

* slide adopted from Li, Karpathy, Johnson’s CS231n at Stanford
**Activation Function:** Sigmoid

\[ a = \text{sigmoid}(x) = \frac{1}{1 + e^{-x}} \]

**Cons:**
- Saturated neurons “kill” the gradients
- Non-zero centered
- Could be expensive to compute

* slide adopted from Li, Karpathy, Johnson’s CS231n at Stanford
Activation Function: Sigmoid

Cons:
- Saturated neurons “kill” the gradients
- Non-zero centered
- Could be expensive to compute

\[ a = \text{sigmoid}(x) = \frac{1}{1 + e^{-x}} \]

\[ \frac{\partial L}{\partial x} = \frac{\partial}{\partial x} \frac{\partial \text{sigmoid}(x)}{\partial a} \quad \frac{\partial L}{\partial a} \]

* slide adopted from Li, Karpathy, Johnson’s CS231n at Stanford
Activation Function: Sigmoid

\[ a = \text{sigmoid}(x) = \frac{1}{1 + e^{-x}} \]

Cons:
- Saturated neurons "kill" the gradients
- Non-zero centered
- Could be expensive to compute

*S* slide adopted from Li, Karpathy, Johnson’s CS231n at Stanford
Activation Function: Tanh

Pros:
- Squishes everything in the range \([-1, 1]\)
- Centered around zero
- Has well defined gradient everywhere

Cons:
- Saturated neurons “kill” the gradients

\[ a(x) = \tanh(x) = 2 \cdot \text{sigmoid}(2x) - 1 \]

\[ a(x) = \tanh(x) = \frac{2}{1 + e^{-2x}} - 1 \]

* slide adopted from Li, Karpathy, Johnson’s CS231n at Stanford
Activation Function: Rectified Linear Unit (ReLU)

Pros:
- Does not saturate (for $x > 0$)
- Computationally very efficient
- Converges faster in practice (e.g. 6 times faster)

Cons:
- Not zero centered

\[
a(x) = \max(0, x)
\]
\[
a'(x) = \begin{cases} 
1 & \text{if } x \geq 0 \\
0 & \text{if } x < 0
\end{cases}
\]