# TU-Delft Deep Learning course 2018-2019

03.backprop

20 Feb 2019

Delft University of Technology

Lecturer: Jan van Gemert
Several slides credit to Roger Grosse

# Learning goals

After this lecture you can:

- Understand what the goal of backpropagation is
- Explain the forward/backward pass
- Apply the calculus chain rule to compute derivatives
- Apply the backpropagation algorithm to efficiently compute derivatives
- Understand the modularity of network nodes

Book chapters: 4.3, 6.5

# Training a network

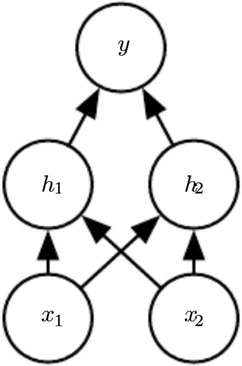
Chapter 4.3

Training set:

| Data | | Label |
|---|---|---|
| 0 | 1 | 1 |
| 1 | 1 | 0 |
| 0 | 0 | 0 |
| 1 | 0 | 1 |

While not converged:

1. Present a training sample
2. Compare the results
3. Update the weights



# Training a network

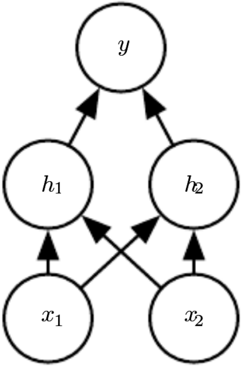
Chapter 4.3

Training set:

| Data | | Label |
|---|---|---|
| 0 | 1 | 1 |
| 1 | 1 | 0 |
| 0 | 0 | 0 |
| 1 | 0 | 1 |

While not converged:

1. Present a training sample → Forward pass
2. Compare the results
3. Update the weights



# Training a network

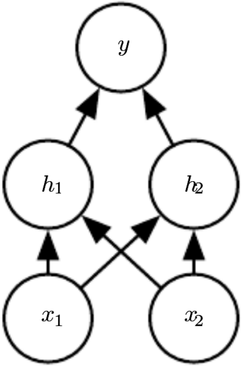
Chapter 4.3

Training set:

| Data | | Label |
|---|---|---|
| 0 | 1 | 1 |
| 1 | 1 | 0 |
| 0 | 0 | 0 |
| 1 | 0 | 1 |

While not converged:

1. Present a training sample $\rightarrow$ Forward pass
2. Compare the results $\rightarrow$ Loss
3. Update the weights

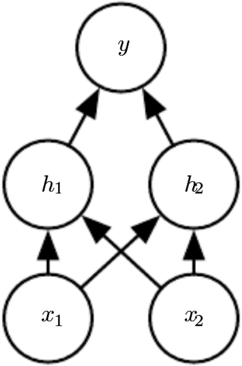


# Training a network
Chapter 4.3

Training set:

| Data | | Label |
|---|---|---|
| 0 | 1 | 1 |
| 1 | 1 | 0 |
| 0 | 0 | 0 |
| 1 | 0 | 1 |

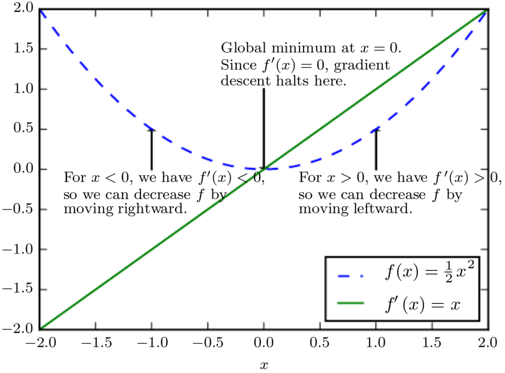
While not converged:
1. Present a training sample → Forward pass
2. Compare the results → Loss
3. Update the weights → Backward pass



# Backward pass: Backpropagation

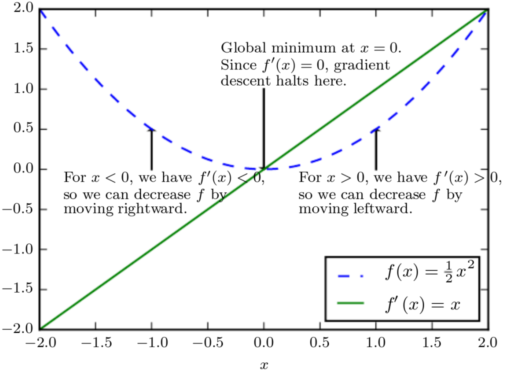
- Backpropagation: Efficient algorithm to compute gradients
- Used to train a huge majority of deep nets
- Engine that powers "End-to-End" optimization and representation learning
- Its "just" a clever application of the chain rule of calculus

# Updating the weights: Gradient descent




1-d gradient descent

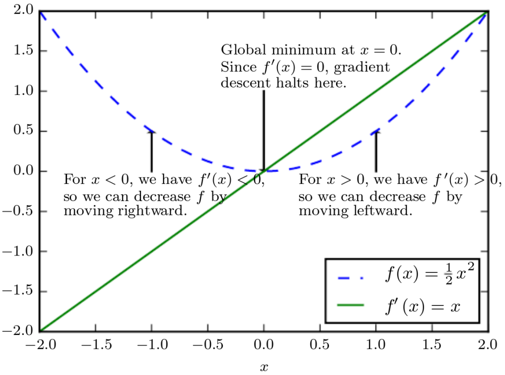
# Updating the weights: Gradient descent




1-d gradient descent

- Q: How does "1-d" change for a multi-layer network?

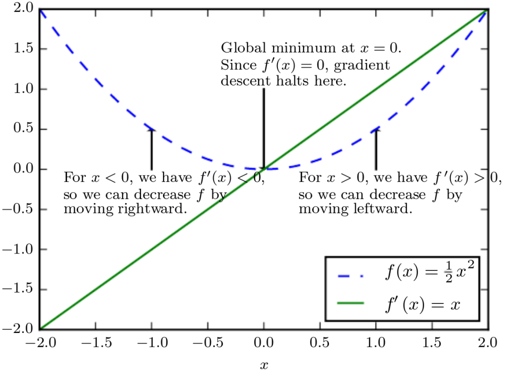
# Updating the weights: Gradient descent




1-d gradient descent

- Q: How does "1-d" change for a multi-layer network?
- A: One coordinate for each weight/bias in *all* the layers

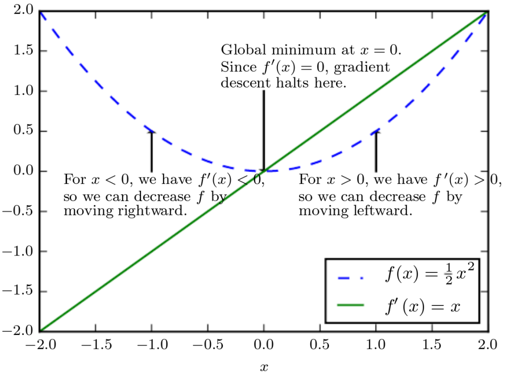
# Updating the weights: Gradient descent




1-d gradient descent

- Q: How does "1-d" change for a multi-layer network?
- A: One coordinate for each weight/bias in *all* the layers
- Compute cost gradient $\frac{d\mathcal{E}}{d\mathbf{w}}$: vector of partial derivatives
- Q: How to get cost gradient from training samples?

# Updating the weights: Gradient descent




1-d gradient descent

- Q: How does "1-d" change for a multi-layer network?
- A: One coordinate for each weight/bias in *all* the layers
- Compute cost gradient $\frac{d\mathcal{E}}{d\mathbf{w}}$: vector of partial derivatives
- Q: How to get cost gradient from training samples?
- A: Average $\frac{d\mathcal{L}}{d\mathbf{w}}$ for loss $\mathcal{L}$ over all training samples

# Chain rule to compute derivatives

- Deep net is a function chain $f(x) = f^{(3)}(f^{(2)}(f^{(1)}(x)))$

  Q: How to compute $f(g(x))'$ ?

# Chain rule to compute derivatives

- Deep net is a function chain $f(x) = f^{(3)}(f^{(2)}(f^{(1)}(x)))$

  Q: How to compute $f(g(x))'$ ?
  A: $f(g(x))' = f'(g(x))g'(x)$

# Chain rule to compute derivatives

- Deep net is a function chain $f(x) = f^{(3)}(f^{(2)}(f^{(1)}(x)))$

  Q: How to compute $f(g(x))'$ ?
  A: $f(g(x))' = f'(g(x))g'(x)$
- Let: $y = g(x)$, and $z = f(g(x)) = f(y)$.
- Other notation:

$$\frac{dz}{dx} = \frac{dz}{dy}\frac{dy}{dx}$$

- Multi-variate chain rule, let $f = (x(t), y(t))$, then

# Chain rule to compute derivatives

- Deep net is a function chain $f(x) = f^{(3)}(f^{(2)}(f^{(1)}(x)))$

  Q: How to compute $f(g(x))'$ ?
  A: $f(g(x))' = f'(g(x))g'(x)$
- Let: $y = g(x)$, and $z = f(g(x)) = f(y)$.
- Other notation:
$$\frac{dz}{dx} = \frac{dz}{dy}\frac{dy}{dx}$$
- Multi-variate chain rule, let $f = (x(t), y(t))$, then
$$\frac{d}{dt}f(x(t), y(t)) = \frac{\partial f}{\partial x}\frac{dx}{dt} + \frac{\partial f}{\partial y}\frac{dy}{dt}$$

# Chain rule to compute derivatives

- Deep net is a function chain $f(x) = f^{(3)}(f^{(2)}(f^{(1)}(x)))$

  Q: How to compute $f(g(x))'$ ?
  A: $f(g(x))' = f'(g(x))g'(x)$
- Let: $y = g(x)$, and $z = f(g(x)) = f(y)$.
- Other notation:
$$\frac{dz}{dx} = \frac{dz}{dy}\frac{dy}{dx}$$
- Multi-variate chain rule, let $f = (x(t), y(t))$, then
$$\frac{d}{dt}f(x(t), y(t)) = \frac{\partial f}{\partial x}\frac{dx}{dt} + \frac{\partial f}{\partial y}\frac{dy}{dt}$$
- Interpretation: How much does output change if the input slightly changes

# Assignment: Compute loss $\mathcal{L}$ derivatives

Model: $\mathcal{L} = \frac{1}{2}\left(\sigma(\text{wx+b})\text{-t}\right)^2$, where $t =$ true label

Q: What to compute?

# Assignment: Compute loss $\mathcal{L}$ derivatives

Model: $\mathcal{L} = \frac{1}{2}\left(\sigma(\mathsf{wx}+\mathsf{b})\text{-}\mathsf{t}\right)^2$, where $t =$ true label

Q: What to compute?

A: Derivative to $w$

$$\frac{\partial}{\partial w}\mathcal{L} = \frac{\partial}{\partial w}\left[\frac{1}{2}\left(\sigma(\mathsf{wx}+\mathsf{b})\text{-}\mathsf{t}\right)^2\right]$$

A: Derivative to $b$

$$\frac{\partial}{\partial b}\mathcal{L} = \frac{\partial}{\partial b}\left[\frac{1}{2}\left(\sigma(\mathsf{wx}+\mathsf{b})\text{-}\mathsf{t}\right)^2\right] =$$

# Assignment: Compute loss $\mathcal{L}$ derivatives

Model: $\mathcal{L} = \frac{1}{2}\left(\sigma(\mathsf{wx+b})\text{-}\mathsf{t}\right)^2$, where $t =$ true label

Q: What to compute?

A: Derivative to $w$

$$\frac{\partial}{\partial w}\mathcal{L} = \frac{\partial}{\partial w}\left[\frac{1}{2}\left(\sigma(\mathsf{wx+b})\text{-}\mathsf{t}\right)^2\right]$$

$$= (\sigma(\mathsf{wx+b})\text{-}\mathsf{t})\frac{\partial}{\partial w}(\sigma(\mathsf{wx+b})\text{-}\mathsf{t})$$

$$= (\sigma(\mathsf{wx+b})\text{-}\mathsf{t})\sigma'(\mathsf{wx+b})\frac{\partial}{\partial w}(\mathsf{wx+b})$$

$$= (\sigma(\mathsf{wx+b})\text{-}\mathsf{t})\sigma'(\mathsf{wx+b})x$$

A: Derivative to $b$

$$\frac{\partial}{\partial b}\mathcal{L} = \frac{\partial}{\partial b}\left[\frac{1}{2}\left(\sigma(\mathsf{wx+b})\text{-}\mathsf{t}\right)^2\right] =$$

$$(\sigma(\mathsf{wx+b})\text{-}\mathsf{t})\frac{\partial}{\partial b}(\sigma(\mathsf{wx+b})\text{-}\mathsf{t}) =$$

$$(\sigma(\mathsf{wx+b})\text{-}\mathsf{t})\sigma'(\mathsf{wx+b})\frac{\partial}{\partial b}(\mathsf{wx+b}) =$$

$$(\sigma(\mathsf{wx+b})\text{-}\mathsf{t})\sigma'(\mathsf{wx+b})$$

# Assignment: Compute loss $\mathcal{L}$ derivatives

Model: $\mathcal{L} = \frac{1}{2}\left(\sigma(\text{wx+b)-t}\right)^2$, where $t =$ true label

Q: What to compute?

A: Derivative to $w$

$$\frac{\partial}{\partial w}\mathcal{L} = \frac{\partial}{\partial w}\left[\frac{1}{2}\left(\sigma(\text{wx+b)-t}\right)^2\right]$$

$$= (\sigma(\text{wx+b)-t})\frac{\partial}{\partial w}(\sigma(\text{wx+b)-t})$$

$$= (\sigma(\text{wx+b)-t})\sigma'(\text{wx+b})\frac{\partial}{\partial w}(\text{wx+b})$$

$$= (\sigma(\text{wx+b)-t})\sigma'(\text{wx+b})x$$

A: Derivative to $b$

$$\frac{\partial}{\partial b}\mathcal{L} = \frac{\partial}{\partial b}\left[\frac{1}{2}\left(\sigma(\text{wx+b)-t}\right)^2\right] =$$

$$(\sigma(\text{wx+b)-t})\frac{\partial}{\partial b}(\sigma(\text{wx+b)-t}) =$$

$$(\sigma(\text{wx+b)-t})\sigma'(\text{wx+b})\frac{\partial}{\partial b}(\text{wx+b}) =$$

$$(\sigma(\text{wx+b)-t})\sigma'(\text{wx+b})$$

Q: What are the disadvantages of this approach?

# More efficient computation of loss $\mathcal{L}$ derivatives

Model: $\mathcal{L} = \frac{1}{2}\left(\sigma(\mathsf{wx}+\mathsf{b})\text{-}\mathsf{t}\right)^2$

Decompose in: $z = wx + b, \quad y = \sigma(z), \quad \mathcal{L} = \frac{1}{2}(y - t)^2$

Q: Rewrite using chain rule in terms of $\frac{dz}{dx} = \frac{dz}{dy}\frac{dy}{dx}$?

# More efficient computation of loss $\mathcal{L}$ derivatives

Model: $\mathcal{L} = \frac{1}{2}\left(\sigma(\text{wx+b)-t}\right)^2$
Decompose in: $z = wx + b, \quad y = \sigma(z), \quad \mathcal{L} = \frac{1}{2}(y - t)^2$

Q: Rewrite using chain rule in terms of $\frac{dz}{dx} = \frac{dz}{dy}\frac{dy}{dx}$?

A: Derivative to $w$

$$\frac{d\mathcal{L}}{dw} = \frac{dL}{dy}\frac{dy}{dz}\frac{dz}{dw}$$

A: Derivative to $b$

$$\frac{d\mathcal{L}}{db} = \frac{dL}{dy}\frac{dy}{dz}\frac{dz}{db}$$

# More efficient computation of loss $\mathcal{L}$ derivatives

$$\text{Model: } \mathcal{L} = \frac{1}{2}\left(\sigma(\text{wx+b})\text{-t}\right)^2$$
$$\text{Decompose in: } z = wx + b, \quad y = \sigma(z), \quad \mathcal{L} = \frac{1}{2}(y - t)^2$$

Q: Rewrite using chain rule in terms of $\frac{dz}{dx} = \frac{dz}{dy}\frac{dy}{dx}$?

A: Derivative to $w$

$$\frac{d\mathcal{L}}{dw} = \frac{dL}{dy}\frac{dy}{dz}\frac{dz}{dw}$$

A: Derivative to $b$

$$\frac{d\mathcal{L}}{db} = \frac{dL}{dy}\frac{dy}{dz}\frac{dz}{db}$$

Q: Nr unique terms to compute both derivatives?

# More efficient computation of loss $\mathcal{L}$ derivatives

Model: $\mathcal{L} = \frac{1}{2}\left(\sigma(\text{wx+b})\text{-t}\right)^2$

Decompose in: $z = wx + b, \quad y = \sigma(z), \quad \mathcal{L} = \frac{1}{2}(y-t)^2$

Q: Rewrite using chain rule in terms of $\frac{dz}{dx} = \frac{dz}{dy}\frac{dy}{dx}$?

A: Derivative to $w$

$$\frac{d\mathcal{L}}{dw} = \frac{dL}{dy}\frac{dy}{dz}\frac{dz}{dw}$$

A: Derivative to $b$

$$\frac{d\mathcal{L}}{db} = \frac{dL}{dy}\frac{dy}{dz}\frac{dz}{db}$$

Q: Nr unique terms to compute both derivatives? A: 4 (efficient re-use)

# More efficient computation of loss $\mathcal{L}$ derivatives

Model: $\mathcal{L} = \frac{1}{2}\left(\sigma(\text{wx+b})\text{-t}\right)^2$

Decompose in: $z = wx + b, \quad y = \sigma(z), \quad \mathcal{L} = \frac{1}{2}(y - t)^2$

Q: Rewrite using chain rule in terms of $\frac{dz}{dx} = \frac{dz}{dy}\frac{dy}{dx}$?

A: Derivative to $w$

$$\frac{d\mathcal{L}}{dw} = \frac{dL}{dy}\frac{dy}{dz}\frac{dz}{dw}$$

A: Derivative to $b$

$$\frac{d\mathcal{L}}{db} = \frac{dL}{dy}\frac{dy}{dz}\frac{dz}{db}$$

Q: Nr unique terms to compute both derivatives? A: 4 (efficient re-use)

$$(1) : \frac{d\mathcal{L}}{dy} = \qquad (2) : \frac{d\mathcal{L}}{dz} = \qquad (3) : \frac{d\mathcal{L}}{dw} = \qquad (4) : \frac{d\mathcal{L}}{db} =$$

# More efficient computation of loss $\mathcal{L}$ derivatives

Model: $\mathcal{L} = \frac{1}{2}\left(\sigma(\text{wx+b})\text{-t}\right)^2$

Decompose in: $z = wx + b, \quad y = \sigma(z), \quad \mathcal{L} = \frac{1}{2}(y - t)^2$

Q: Rewrite using chain rule in terms of $\frac{dz}{dx} = \frac{dz}{dy}\frac{dy}{dx}$?

A: Derivative to $w$

$$\frac{d\mathcal{L}}{dw} = \frac{dL}{dy}\frac{dy}{dz}\frac{dz}{dw}$$

A: Derivative to $b$

$$\frac{d\mathcal{L}}{db} = \frac{dL}{dy}\frac{dy}{dz}\frac{dz}{db}$$

Q: Nr unique terms to compute both derivatives? A: 4 (efficient re-use)

$$(1) : \frac{d\mathcal{L}}{dy} = y - t, \quad (2) : \frac{d\mathcal{L}}{dz} = \frac{d\mathcal{L}}{dy}\sigma'(z), \quad (3) : \frac{d\mathcal{L}}{dw} = \frac{d\mathcal{L}}{dz}x, \quad (4) : \frac{d\mathcal{L}}{db} = \frac{d\mathcal{L}}{dz}.$$

# More efficient computation of loss $\mathcal{L}$ derivatives

Model: $\mathcal{L} = \frac{1}{2}\left(\sigma(\text{wx+b})\text{-t}\right)^2$
Decompose in: $z = wx + b, \quad y = \sigma(z), \quad \mathcal{L} = \frac{1}{2}(y - t)^2$

Q: Rewrite using chain rule in terms of $\frac{dz}{dx} = \frac{dz}{dy}\frac{dy}{dx}$?

A: Derivative to $w$

$$\frac{d\mathcal{L}}{dw} = \frac{dL}{dy}\frac{dy}{dz}\frac{dz}{dw}$$

A: Derivative to $b$

$$\frac{d\mathcal{L}}{db} = \frac{dL}{dy}\frac{dy}{dz}\frac{dz}{db}$$

Q: Nr unique terms to compute both derivatives? A: 4 (efficient re-use)

$$(1): \frac{d\mathcal{L}}{dy} = y - t, \quad (2): \frac{d\mathcal{L}}{dz} = \frac{d\mathcal{L}}{dy}\sigma'(z), \quad (3): \frac{d\mathcal{L}}{dw} = \frac{d\mathcal{L}}{dz}x, \quad (4): \frac{d\mathcal{L}}{db} = \frac{d\mathcal{L}}{dz}.$$

Q: Dependency order of computation?

# More efficient computation of loss $\mathcal{L}$ derivatives

Model: $\mathcal{L} = \frac{1}{2}\left(\sigma(\text{wx+b})\text{-t}\right)^2$
Decompose in: $z = wx + b, \quad y = \sigma(z), \quad \mathcal{L} = \frac{1}{2}(y-t)^2$

Q: Rewrite using chain rule in terms of $\frac{dz}{dx} = \frac{dz}{dy}\frac{dy}{dx}$?

A: Derivative to $w$

$$\frac{d\mathcal{L}}{dw} = \frac{dL}{dy}\frac{dy}{dz}\frac{dz}{dw}$$

A: Derivative to $b$

$$\frac{d\mathcal{L}}{db} = \frac{dL}{dy}\frac{dy}{dz}\frac{dz}{db}$$

Q: Nr unique terms to compute both derivatives? A: 4 (efficient re-use)

$$(1): \frac{d\mathcal{L}}{dy} = y - t, \quad (2): \frac{d\mathcal{L}}{dz} = \frac{d\mathcal{L}}{dy}\sigma'(z), \quad (3): \frac{d\mathcal{L}}{dw} = \frac{d\mathcal{L}}{dz}x, \quad (4): \frac{d\mathcal{L}}{db} = \frac{d\mathcal{L}}{dz}.$$

Q: Dependency order of computation? A: Starts backwards.

# Re-using pre-computed values

Expressions to evaluate

$$\frac{df}{dt} = \frac{\partial f}{\partial x}\frac{dx}{dt} + \frac{\partial f}{\partial y}\frac{dy}{dt}$$

Values already computed

[1] Bar notation credits to Roger Grosse

# Re-using pre-computed values

Expressions to evaluate

$$\frac{df}{dt} = \frac{\partial f}{\partial x}\frac{dx}{dt} + \frac{\partial f}{\partial y}\frac{dy}{dt}$$

Values already computed

Bar notation:

- Backprop is an algorithm to compute values
- Bar notation[1] for computed values: $\overline{y} = \frac{d\mathcal{E}}{dy}$
- Less cluttered and emphasizes value re-use

[1] Bar notation credits to Roger Grosse

# Re-using pre-computed values

Expressions to evaluate

$$\frac{df}{dt} = \frac{\partial f}{\partial x}\frac{dx}{dt} + \frac{\partial f}{\partial y}\frac{dy}{dt}$$

Values already computed

Bar notation:

- Backprop is an algorithm to compute values
- Bar notation[1] for computed values: $\overline{y} = \frac{d\mathcal{E}}{dy}$
- Less cluttered and emphasizes value re-use

Example: $z = wx + b, \quad y = \sigma(z), \quad \mathcal{L} = \frac{1}{2}(y-t)^2$

[1] Bar notation credits to Roger Grosse

# Re-using pre-computed values

Expressions to evaluate

$$\frac{df}{dt} = \frac{\partial f}{\partial x}\frac{dx}{dt} + \frac{\partial f}{\partial y}\frac{dy}{dt}$$

Values already computed

Bar notation:

- Backprop is an algorithm to compute values
- Bar notation[1] for computed values: $\overline{y} = \frac{d\mathcal{E}}{dy}$
- Less cluttered and emphasizes value re-use

Example: $z = wx + b, \quad y = \sigma(z), \quad \mathcal{L} = \frac{1}{2}(y-t)^2$

Instead of:

$$(1): \frac{d\mathcal{L}}{dy} = y - t, \quad (2): \frac{d\mathcal{L}}{dz} = \frac{d\mathcal{L}}{dy}\sigma'(z), \quad (3): \frac{d\mathcal{L}}{dw} = \frac{d\mathcal{L}}{dz}x, \quad (4): \frac{d\mathcal{L}}{db} = \frac{d\mathcal{L}}{dz}.$$

[1] Bar notation credits to Roger Grosse

# Re-using pre-computed values

Expressions to evaluate

$$\frac{df}{dt} = \frac{\partial f}{\partial x}\frac{dx}{dt} + \frac{\partial f}{\partial y}\frac{dy}{dt}$$

Values already computed

Bar notation:

- Backprop is an algorithm to compute values
- Bar notation[1] for computed values: $\overline{y} = \frac{d\mathcal{E}}{dy}$
- Less cluttered and emphasizes value re-use

Example: $z = wx + b, \quad y = \sigma(z), \quad \mathcal{L} = \frac{1}{2}(y-t)^2$

Instead of:

$$(1): \frac{d\mathcal{L}}{dy} = y - t, \quad (2): \frac{d\mathcal{L}}{dz} = \frac{d\mathcal{L}}{dy}\sigma'(z), \quad (3): \frac{d\mathcal{L}}{dw} = \frac{d\mathcal{L}}{dz}x, \quad (4): \frac{d\mathcal{L}}{db} = \frac{d\mathcal{L}}{dz}.$$

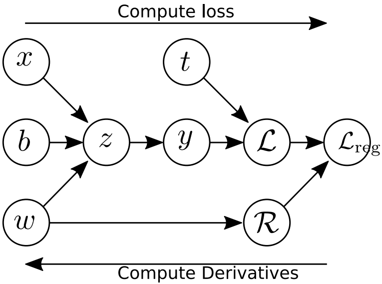
Write:

$$(1): \overline{y} = y - t, \qquad (2): \overline{z} = \overline{y}\sigma'(z), \qquad (3): \overline{w} = \overline{z}x, \qquad (4): \overline{b} = \overline{z}.$$

[1] Bar notation credits to Roger Grosse

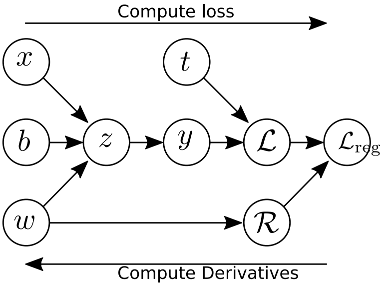
# Computational graph

Book: Section 6.5.1



# Computational graph

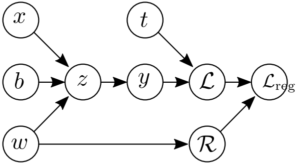
Book: Section 6.5.1




- A node is a variable: scalar, vector, matrix, tensor, other node, etc.
- An operation: simple function of 2 variables (often omitted)
- Operation have single output (possibly multiple entries)
- $x \to y$: y is computed by applying operation to x.

# Backpropagation algorithm

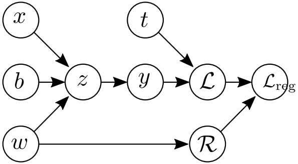
Book: Algorithms 6.1 and 6.2




Topological ordering:
Linear ordering of vertices so for every directed edge uv, vertex u comes before v in the ordering.

# Backpropagation algorithm

Book: Algorithms 6.1 and 6.2

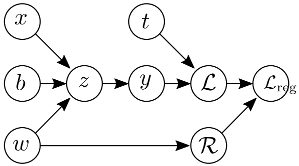



Topological ordering:
Linear ordering of vertices so for every directed edge uv, vertex u comes before v in the ordering.

Q: Topological ordering of graph?

# Backpropagation algorithm

Book: Algorithms 6.1 and 6.2

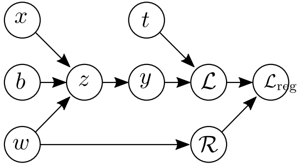



Topological ordering:
Linear ordering of vertices so for every directed edge uv, vertex u comes before v in the ordering.

Q: Topological ordering of graph?
A: $(w, b, x, z, y, t, \mathcal{L}, \mathcal{R}, \mathcal{L}_{\textsf{reg}})$

# Backpropagation algorithm

Book: Algorithms 6.1 and 6.2




Topological ordering:
Linear ordering of vertices so for every directed edge uv, vertex u comes before v in the ordering.

Q: Topological ordering of graph?
A: $(w, b, x, z, y, t, \mathcal{L}, \mathcal{R}, \mathcal{L}_{\text{reg}})$

Backpropagation algorithm to compute gradients of node $n_N$:

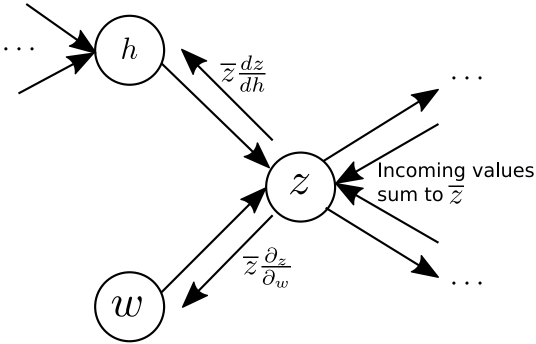
1. Create topological ordering of graph

Forward pass $= \{$
2. For $i = 1, 2, \ldots N$ :
   Evaluate $n_i$ using its function $f^{(i)}(n_i)$

Backward pass $= \{$
3. $\overline{n_N} = 1$ (gradient of a node wrt itself is 1; $\frac{df}{df} = 1$)
4. For $i = N-1, \ldots 1$ :
   $\overline{n_i} = \sum_{n_j \in \text{Children}(n_i)} \overline{n_j} \frac{\partial_{n_j}}{\partial_{n_i}}$

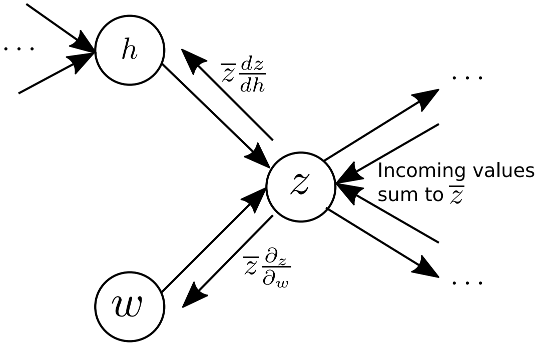
Note on step 4: node only gets gradients wrt children nodes

# Backprop as message passing



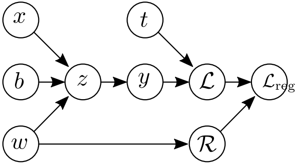

- Each node aggregates the error signal from its children
- Each node passes messages to its parents
- Q: Benefit from this node-centered point of view?

# Backprop as message passing




- Each node aggregates the error signal from its children
- Each node passes messages to its parents
- Q: Benefit from this node-centered point of view?
- A: Modularity: Only needs to compute derivatives wrt its arguments.

# Example: Univariate logistic least squares regression




Q: Do the backward pass:

Forward pass:

$$z = wx + b$$

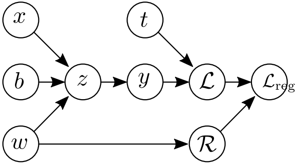
$$y = \sigma(z)$$

$$\mathcal{L} = \frac{1}{2}(y - t)^2$$

$$\mathcal{R} = \frac{1}{2}w^2$$

$$\mathcal{L}_{reg} = \mathcal{L} + \lambda\mathcal{R}$$

# Example: Univariate logistic least squares regression



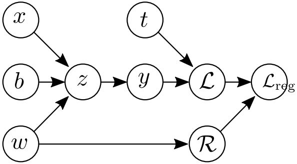

Forward pass:

$$
\begin{aligned}
z &= wx + b \\
y &= \sigma(z) \\
\mathcal{L} &= \frac{1}{2}(y - t)^2 \\
\mathcal{R} &= \frac{1}{2}w^2 \\
\mathcal{L}_{reg} &= \mathcal{L} + \lambda\mathcal{R}
\end{aligned}
$$

Q: Do the backward pass:

$$
\begin{aligned}
\overline{\mathcal{L}_{reg}} &= \\
\overline{\mathcal{R}} &= \\
\overline{\mathcal{L}} &= \\
\overline{y} &= \\
\overline{z} &= \\
\overline{w} &= \\
\overline{b} &=
\end{aligned}
$$

# Example: Univariate logistic least squares regression




Forward pass:

$$
\begin{aligned}
z &= wx + b \\
y &= \sigma(z) \\
\mathcal{L} &= \frac{1}{2}(y-t)^2 \\
\mathcal{R} &= \frac{1}{2}w^2 \\
\mathcal{L}_{reg} &= \mathcal{L} + \lambda\mathcal{R}
\end{aligned}
$$

Q: Do the backward pass:

$$
\begin{aligned}
\overline{\mathcal{L}_{reg}} &= 1 \\
\overline{\mathcal{R}} &= \overline{\mathcal{L}_{reg}}\frac{d\mathcal{L}_{reg}}{d\mathcal{R}} = \overline{\mathcal{L}_{reg}}\lambda \\
\overline{\mathcal{L}} &= \overline{\mathcal{L}_{reg}}\frac{d\mathcal{L}_{reg}}{d\mathcal{L}} = \overline{\mathcal{L}_{reg}} \\
\overline{y} &= \overline{\mathcal{L}}\frac{d\mathcal{L}}{dy} = \overline{\mathcal{L}}(y-t) \\
\overline{z} &= \overline{y}\frac{dy}{dz} = \overline{y}\sigma'(z) \\
\overline{w} &= \overline{z}\frac{\partial z}{\partial w} + \overline{\mathcal{R}}\frac{\partial \mathcal{R}}{\partial w} = \overline{z}x\overline{\mathcal{R}}w \\
\overline{b} &= \overline{z}\frac{\partial z}{\partial b} = \overline{z}
\end{aligned}
$$

# Learning goals

- Understand how backpropagation powers deep learning
- Explain about the forward/backward pass
- Apply the calculus chain rule to compute derivatives
- Apply the backpropagation algorithm to efficiently compute derivatives
- Understand the modularity of network nodes