Neural Networks: Backpropagation

Machine Learning
Fall 2023

The slides are partly from Vivek Srikumar
Neural Networks

• What is a neural network?
• Predicting with a neural network
• Training neural networks
• Practical concerns
This lecture

• What is a neural network?

• Predicting with a neural network

• **Training neural networks**
  – Backpropagation

• Practical concerns
Training a neural network

• **Given**
  – A network architecture (layout of neurons, their connectivity and activations)
  – A dataset of labeled examples
    • \( S = \{(x_i, y_i)\} \)

• **The goal**: Learn the weights of the neural network

• **Remember**: For a fixed architecture, a neural network is a function parameterized by its weights
  – **Prediction**: \( y = NN(x, w) \)
Recall: Learning as loss minimization

We have a classifier $NN$ that is completely defined by its weights. Learn the weights by minimizing a loss $L$

$$\min_w \sum_i L(NN(x_i, w), y_i)$$

Perhaps with a regularizer
Recall: Learning as loss minimization

We have a classifier $NN$ that is completely defined by its weights. Learn the weights by minimizing a loss $L$

$$\min_w \sum_i L(\text{NN}(x_i, w), y_i)$$

Perhaps with a regularizer

So far, we saw that this strategy worked for:

1. Support Vector Machines
2. Perceptron
3. LMS regression

Each minimizes a different loss function

All of these are linear models

Same idea for non-linear models too!
Back to our running example

Given an input $x$, how is the output predicted

\[
\begin{align*}
    y &= w^o_{01} + w^o_{11} z_1 + w^o_{21} z_2 \\
    z_2 &= \sigma(w^h_{02} + w^h_{12} x_1 + w^h_{22} x_2) \\
    z_1 &= \sigma(w^h_{01} + w^h_{11} x_1 + w^h_{21} x_2)
\end{align*}
\]
Back to our running example

Given an input $x$, how is the output predicted

output $y = w_{01}^o + w_{11}^o z_1 + w_{21}^o z_2$

$z_2 = \sigma(w_{02}^h + w_{12}^h x_1 + w_{22}^h x_2)$

$z_1 = \sigma(w_{01}^h + w_{11}^h x_1 + w_{21}^h x_2)$

Suppose the true label for this example is a number $y^*$
Back to our running example

Given an input $x$, how is the output predicted

$$y = w^o_{01} + w^o_{11}z_1 + w^o_{21}z_2$$

$$z_2 = \sigma(w^h_{02} + w^h_{12}x_1 + w^h_{22}x_2)$$

$$z_1 = \sigma(w^h_{01} + w^h_{11}x_1 + w^h_{21}x_2)$$

Suppose the true label for this example is a number $y^*$

We can write the *square loss* for this example as:

$$L = \frac{1}{2} (y - y^*)^2$$
Learning as loss minimization

We have a classifier $NN$ that is completely defined by its weights
Learn the weights by minimizing a loss $L$

$$
\min_w \sum_i L(NN(x_i, w), y_i)
$$

Perhaps with a regularizer

How do we solve the optimization problem?
Stochastic gradient descent

Given a training set \( S = \{(x_i, y_i)\}, x \in \mathbb{R}^d \)

1. Initialize parameters \( w \)
2. For epoch = 1 ... T:
   a. Shuffle the training set
   b. For each training example \((x_i, y_i) \in S\):
      • Treat this example as the entire dataset
      • Compute the gradient of the loss \( \nabla L(\text{NN}(x_i, w), y_i) \)
      • Update: \( w \leftarrow w - \gamma \nabla L(\text{NN}(x_i, w), y_i) \)
3. Return \( w \)
Stochastic gradient descent

Given a training set $S = \{(x_i, y_i)\}$, $x \in \mathbb{R}^d$

1. Initialize parameters $w$
2. For epoch = 1 ... $T$:
   1. Shuffle the training set
3. Return $w$
Stochastic gradient descent

Given a training set \( S = \{(x_i, y_i)\}, x \in \mathbb{R}^d \)

1. Initialize parameters \( w \)

2. For epoch = 1 ... T:
   1. Shuffle the training set
   2. For each training example \((x_i, y_i)\) \(\in S\):

3. Return \( w \)
Stochastic gradient descent

Given a training set \( S = \{(x_i, y_i)\}, x \in \mathbb{R}^d \)

1. Initialize parameters \( w \)

2. For epoch = 1 ... T:
   1. Shuffle the training set
   2. For each training example \((x_i, y_i)\) \(\in S\):
      • Treat this example as the entire dataset
      Compute the gradient of the loss \( \nabla L(\text{NN}(x_i, w), y_i) \)

3. Return \( w \)
Stochastic gradient descent

Given a training set $S = \{(x_i, y_i)\}, \mathbf{x} \in \mathbb{R}^d$

1. Initialize parameters $w$

2. For epoch $= 1 \ldots T$:
   
   1. Shuffle the training set
   
   2. For each training example $(x_i, y_i) \in S$:
      
      • Treat this example as the entire dataset
      
      Compute the gradient of the loss $\nabla L(\text{NN}(x_i, w), y_i)$

      • Update: $w \leftarrow w - \gamma_t \nabla L(\text{NN}(x_i, w), y_i)$$

3. Return $w$
Stochastic gradient descent

Given a training set \( S = \{(x_i, y_i)\}, \ x \in \mathbb{R}^d \)

1. Initialize parameters \( w \)
2. For epoch = 1 ... T:
   1. Shuffle the training set
   2. For each training example \((x_i, y_i) \in S\):
      • Treat this example as the entire dataset
      • Compute the gradient of the loss \( \nabla L(NN(x_i, w), y_i) \)
      • Update: \( w \leftarrow w - \gamma_t \nabla L(NN(x_i, w), y_i) \)

3. Return \( w \)
Stochastic gradient descent

Given a training set $S = \{(x_i, y_i)\}$, $x \in \mathbb{R}^d$

1. Initialize parameters $w$

2. For epoch = 1 ... $T$:
   1. Shuffle the training set
   2. For each training example $(x_i, y_i) \in S$:
      • Treat this example as the entire dataset
        Compute the gradient of the loss $\nabla L(NN(x_i, w), y_i)$
      • Update: $w \leftarrow w - \gamma_t \nabla L(NN(x_i, w), y_i))$

3. Return $w$

The objective is not convex.
Initialization can be important

$\gamma_t$: learning rate, many tweaks possible
Stochastic gradient descent

Given a training set \( S = \{ (x_i, y_i) \}, \ x \in \mathbb{R}^d \)

1. Initialize parameters \( w \)
2. For epoch = 1 ... T:
   1. Shuffle the training set
   2. For each training example \((x_i, y_i) \in S:\n       \quad \text{• Treat this example as the entire dataset}
       \quad \text{Compute the gradient of the loss} \ \nabla L(NN(x_i, w), y_i)
       \quad \text{• Update:} \ w \leftarrow w - \gamma_t \nabla L(NN(x_i, w), y_i))

3. Return \( w \)

Have we solved everything?

\[ \min_w \sum_i L(NN(x_i, w), y_i) \]

The objective is not convex.
Initialization can be important
\( \gamma_t \): learning rate, many tweaks possible
The derivative of the loss function? 

\[ \nabla L(\mathcal{NN}(x_i, w), y_i) \]

If the neural network is a differentiable function, we can find the gradient

- Or maybe its sub-gradient
- This is decided by the activation functions and the loss function

It was easy for SVMs and LMS regression

- Only one layer
If the neural network is a differentiable function, we can find the gradient
   – Or maybe its sub-gradient
   – This is decided by the activation functions and the loss function

It was easy for SVMs and LMS regression
   – Only one layer

But how do we find the sub-gradient of a more complex function?
   – Eg: A recent paper used a ~250 layer neural network for image classification!
The derivative of the loss function? \( \nabla L(\text{NN}(x_i, w), y_i) \)

If the neural network is a differentiable function, we can find the gradient
- Or maybe its sub-gradient
- This is decided by the activation functions and the loss function

It was easy for SVMs and LMS regression
- Only one layer

But how do we find the sub-gradient of a more complex function?
- Eg: A recent paper used a ~150 layer neural network for image classification!

We need an efficient algorithm: Backpropagation
Where are we

If we have a neural network (structure, activations and weights), we can make a prediction for an input. If we had the true label of the input, then we can define the loss for that example. If we can take the derivative of the loss with respect to each of the weights, we can take a gradient step in SGD.
If we have a neural network (structure, activations and weights), we can make a prediction for an input.
If we have a neural network (structure, activations and weights), we can make a prediction for an input

If we had the true label of the input, then we can define the loss for that example
If we have a neural network (structure, activations and weights), we can make a prediction for an input.

If we had the true label of the input, then we can define the loss for that example.

If we can take the derivative of the loss with respect to each of the weights, we can take a gradient step in SGD.
If we have a neural network (structure, activations and weights), we can make a prediction for an input.

If we had the true label of the input, then we can define the loss for that example.

If we can take the derivative of the loss with respect to each of the weights, we can take a gradient step in SGD.

Questions?
Reminder: Chain rule for derivatives

- If $z$ is a function of $y$ and $y$ is a function of $x$
  - Then $z$ is a function of $x$, as well
- Question: how to find $\frac{\partial z}{\partial x}$

\[
\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y} \frac{\partial y}{\partial x}
\]
Reminder: Chain rule for derivatives

- If $z = a$ function of $y_1$ and $y_2$, and the $y_i$'s are functions of $x$
  - Then $z$ is a function of $x$, as well
- Question: how to find $\frac{\partial z}{\partial x}$

$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y_1} \frac{\partial y_1}{\partial x} + \frac{\partial z}{\partial y_2} \frac{\partial y_2}{\partial x}$$
Reminder: Chain rule for derivatives

- If $z$ is a functions of $y_i$'s, and the $y_i$'s are functions of $x$
  - Then $z$ is a function of $x$, as well
- Question: how to find $\frac{\partial z}{\partial x}$

$$\frac{\partial z}{\partial x} = \sum_{i=1}^{n} \frac{\partial z}{\partial y_i} \frac{\partial y_i}{\partial x}$$
Backpropagation

\[ L = \frac{1}{2} (y - y^*)^2 \]

output \( y = w_0^o + w_1^o z_1 + w_2^o z_2 \)

\[ z_2 = \sigma(w_0^h + w_1^h x_1 + w_2^h x_2) \]

\[ z_1 = \sigma(w_0^h + w_1^h x_1 + w_2^h x_2) \]
We want to compute $L = \frac{1}{2} \left( y - y^* \right)^2$

output $y = w_{01}^o + w_{11}^o z_1 + w_{21}^o z_2$

$z_2 = \sigma(w_{02}^h + w_{12}^h x_1 + w_{22}^h x_2)$

$z_1 = \sigma(w_{01}^h + w_{11}^h x_1 + w_{21}^h x_2)$

We want to compute $\frac{\partial L}{\partial w_{ij}^o}$ and $\frac{\partial L}{\partial w_{ij}^h}$
Backpropagation

Applying the chain rule to compute the gradient (And remembering partial computations along the way to speed up things)

\[ L = \frac{1}{2} (y - y^*)^2 \]

We want to compute \( \frac{\partial L}{\partial w_{ij}^o} \) and \( \frac{\partial L}{\partial w_{ij}^h} \)

\[ \text{output } y = w_{o1}^o + w_{11}^o z_1 + w_{21}^o z_2 \]

\[ z_2 = \sigma(w_{02}^h + w_{12}^h x_1 + w_{22}^h x_2) \]

\[ z_1 = \sigma(w_{01}^h + w_{11}^h x_1 + w_{21}^h x_2) \]

\[ \text{output } y = w_{01}^o + w_{11}^o z_1 + w_{21}^o z_2 \]
Backpropagation example

Output layer

\[ L = \frac{1}{2} (y - y^*)^2 \]

output \[ y = w_{01}^o + w_{11}^o z_1 + w_{21}^o z_2 \]

\[ \frac{\partial L}{\partial w_{01}^o} \]
Output layer

\[ L = \frac{1}{2} (y - y^*)^2 \]
output \( y = w^o_{01} + w^o_{11}z_1 + w^o_{21}z_2 \)

\[ \frac{\partial L}{\partial w^o_{01}} = \frac{\partial L}{\partial y} \frac{\partial y}{\partial w^o_{01}} \]
Backpropagation example

Output layer

\[ L = \frac{1}{2} (y - y^*)^2 \]

output \[ y = w_{01}^o + w_{11}^o z_1 + w_{21}^o z_2 \]

\[ \frac{\partial L}{\partial w_{01}^o} = \frac{\partial L}{\partial y} \frac{\partial y}{\partial w_{01}^o} \]

\[ \frac{\partial L}{\partial y} = y - y^* \]
Backpropagation example

Output layer

\[ L = \frac{1}{2} (y - y^*)^2 \]

output \[ y = w_{01}^o + w_{11}^o z_1 + w_{21}^o z_2 \]

\[ \frac{\partial L}{\partial w_{01}^o} = \frac{\partial L}{\partial y} \frac{\partial y}{\partial w_{01}^o} \]

\[ \frac{\partial L}{\partial y} = y - y^* \]

\[ \frac{\partial y}{\partial w_{01}^o} = 1 \]
Backpropagation example

Output layer

\[ L = \frac{1}{2} (y - y^*)^2 \]

output \quad y = w^{o}_{01} + w^{o}_{11}z_{1} + w^{o}_{21}z_{2}

\[
\frac{\partial L}{\partial w^{o}_{11}}
\]
Backpropagation example

Output layer

\[ L = \frac{1}{2} (y - y^*)^2 \]
output \[ y = w^{o}_0 + w^{o}_{11}z_1 + w^{o}_{21}z_2 \]

\[ \frac{\partial L}{\partial w^{o}_{11}} = \frac{\partial L}{\partial y} \frac{\partial y}{\partial w^{o}_{11}} \]
Backpropagation example

Output layer

\[ L = \frac{1}{2} (y - y^*)^2 \]

output \[ y = w_{01}^o + w_{11}^o z_1 + w_{21}^o z_2 \]

\[ \frac{\partial L}{\partial w_{11}^o} = \frac{\partial L}{\partial y} \frac{\partial y}{\partial w_{11}^o} \]

\[ \frac{\partial L}{\partial y} = y - y^* \]
Backpropagation example

\[
L = \frac{1}{2} (y - y^*)^2
\]

output \quad \begin{align*}
y &= w_0^o + w_{11}^o z_1 + w_{21}^o z_2 \\

\frac{\partial L}{\partial w_{11}^o} &= \frac{\partial L}{\partial y} \frac{\partial y}{\partial w_{11}^o} \\
\frac{\partial L}{\partial y} &= y - y^* \\
\frac{\partial y}{\partial w_{11}^o} &= z_1
\end{align*}
\]
We have already computed this partial derivative for the previous case

Cache to speed up!
Hidden layer derivatives

\[ L = \frac{1}{2} (y - y^*)^2 \]

output \( y = w^0_{01} + w^0_{11} z_1 + w^0_{21} z_2 \)

\[ z_2 = \sigma(w^h_{02} + w^h_{12} x_1 + w^h_{22} x_2) \]

\[ z_1 = \sigma(w^h_{01} + w^h_{11} x_1 + w^h_{21} x_2) \]
Hidden layer derivatives

Backpropagation example

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

output \( y = w^0_{01} + w^0_{11}z_1 + w^0_{21}z_2 \)

\( z_2 = \sigma(w^h_{02} + w^h_{12}x_1 + w^h_{22}x_2) \)

\( z_1 = \sigma(w^h_{01} + w^h_{11}x_1 + w^h_{21}x_2) \)

We want \( \frac{\partial L}{\partial w^h_{22}} \)
Hidden layer derivatives

Backpropagation example

\[ L = \frac{1}{2} (y - y^*)^2 \]

\[ \frac{\partial L}{\partial w_{22}^h} = \frac{\partial L}{\partial y} \frac{\partial y}{\partial w_{22}^h} \]
Hidden layer

\[ y = w_{01}^o + w_{11}^o z_1 + w_{21}^o z_2 \]

\[ \frac{\partial L}{\partial w_{22}^h} = \frac{\partial L}{\partial y} \frac{\partial y}{\partial w_{22}^h} = \frac{\partial L}{\partial y} \left( \frac{\partial y}{\partial z_0} \frac{\partial z_0}{\partial w_{22}^h} + \frac{\partial y}{\partial z_1} \frac{\partial z_1}{\partial w_{22}^h} + \frac{\partial y}{\partial z_2} \frac{\partial z_2}{\partial w_{22}^h} \right) \]
Backpropagation example

**Hidden layer**

\[ y = w^o_{01} + w^o_{11}z_1 + w^o_{21}z_2 \]

\[ \frac{\partial L}{\partial w^h_{22}} = \frac{\partial L}{\partial y} \frac{\partial y}{\partial w^h_{22}} \]

\[ = \frac{\partial L}{\partial y} \left( \frac{\partial y}{\partial z_0} \frac{\partial z_0}{\partial w^h_{22}} + \frac{\partial y}{\partial z_1} \frac{\partial z_1}{\partial w^h_{22}} + \frac{\partial y}{\partial z_2} \frac{\partial z_2}{\partial w^h_{22}} \right) \]

**Question:**

\[ \frac{\partial z_0}{\partial w^h_{22}} = ? \quad \frac{\partial z_1}{\partial w^h_{22}} = ? \]
Backpropagation example

**Hidden layer**

\[
y = w_{01}^o + w_{11}^o z_1 + w_{21}^o z_2
\]

\[
\frac{\partial L}{\partial w_{22}^h} = \frac{\partial L}{\partial y} \frac{\partial y}{\partial w_{22}^h} = \frac{\partial L}{\partial y} \left( \frac{\partial y}{\partial z_0} \frac{\partial z_0}{\partial w_{22}^h} + \frac{\partial y}{\partial z_1} \frac{\partial z_1}{\partial w_{22}^h} + \frac{\partial y}{\partial z_2} \frac{\partial z_2}{\partial w_{22}^h} \right)
\]

**Question:**

\[
\frac{\partial z_0}{\partial w_{22}^h} = ? \quad \frac{\partial z_1}{\partial w_{22}^h} = ?
\]

The answer is 0, because \(z_0\) and \(z_1\) do not depend on \(w_{22}^h\)
Backpropagation example

Hidden layer

\[ y = w_{01}^o + w_{11}^o z_1 + w_{21}^o z_2 \]

\[
\frac{\partial L}{\partial w_{22}^h} = \frac{\partial L}{\partial y} \frac{\partial y}{\partial w_{22}^h} \\
= \frac{\partial L}{\partial y} \left( \frac{\partial y}{\partial z_0} \frac{\partial z_0}{\partial w_{22}^h} + \frac{\partial y}{\partial z_1} \frac{\partial z_1}{\partial w_{22}^h} + \frac{\partial y}{\partial z_2} \frac{\partial z_2}{\partial w_{22}^h} \right)
\]

Question:

\[ \frac{\partial z_0}{\partial w_{22}^h} = ? \quad \frac{\partial z_1}{\partial w_{22}^h} = ? \]

The answer is 0, because \( z_0 \) and \( z_1 \) do not depend on \( w_{22}^h \)
Backpropagation example

Hidden layer

\[ y = w_0^o + w_{11}^o z_1 + w_{21}^o z_2 \]

\[
\frac{\partial L}{\partial w_{22}^h} = \frac{\partial L}{\partial y} \frac{\partial y}{\partial w_{22}^h} = \frac{\partial L}{\partial y} \frac{\partial y}{\partial z_2} \frac{\partial z_2}{\partial w_{22}^h}
\]
Backpropagation example

Hidden layer

\[ y = w_{01}^o + w_{11}^o z_1 + w_{21}^o z_2 \]

\[
\frac{\partial L}{\partial w_{22}^h} = \frac{\partial L}{\partial y} \frac{\partial y}{\partial w_{22}^h} = \frac{\partial L}{\partial y} \frac{\partial y}{\partial z_2} \frac{\partial z_2}{\partial w_{22}^h}
\]

Each of these partial derivatives is easy

\[
\frac{\partial L}{\partial y} = y - y^*
\]
Hidden layer

Backpropagation example

\[ y = w_{01}^o + w_{11}^o z_1 + w_{21}^o z_2 \]

Each of these partial derivatives is easy

\[ \frac{\partial L}{\partial y} = y - y^* \]

\[ \frac{\partial y}{\partial z_2} = w_{21}^o \]
Hidden layer

Backpropagation example

\[ z_2 = \sigma(w_{02}^h + w_{12}^h x_1 + w_{22}^h x_2) \]

\[
\frac{\partial L}{\partial w_{22}^h} = \frac{\partial L}{\partial y} \frac{\partial y}{\partial w_{22}^h} \\
= \frac{\partial L}{\partial y} \frac{\partial y}{\partial z_2} \frac{\partial z_2}{\partial w_{22}^h}
\]

How about \[ \frac{\partial z_2}{\partial w_{22}^h} \]?
Hidden layer

\[
\frac{\partial L}{\partial w^h_{22}} = \frac{\partial L}{\partial y} \frac{\partial y}{\partial w^h_{22}} = \frac{\partial L}{\partial y} \frac{\partial y}{\partial z_2} \frac{\partial z_2}{\partial w^h_{22}}
\]

How about \( \frac{\partial z_2}{\partial w^h_{22}} \)?

\[
\frac{\partial z_2}{\partial w^h_{22}} = \frac{\partial \sigma}{\partial s} \frac{\partial s}{\partial w^h_{22}}
\]

\[
z_2 = \sigma(w^h_{02} + w^h_{12}x_1 + w^h_{22}x_2)
\]

Call this \( s \)
Backpropagation example

Hidden layer

\[
\frac{\partial L}{\partial w^h_{22}} = \frac{\partial L}{\partial y} \frac{\partial y}{\partial w^h_{22}}
\]

How about \( \frac{\partial z_2}{\partial w^h_{22}} \)?

\[
\frac{\partial z_2}{\partial w^h_{22}} = \frac{\partial \sigma}{\partial s} \frac{\partial s}{\partial w^h_{22}}
\]

\[
\frac{\partial s}{\partial w^h_{22}} = x_2
\]
Backpropagation example

Hidden layer

\[ z_2 = \sigma(w_{02}^h + w_{12}^h x_1 + w_{22}^h x_2) \]

\[
\frac{\partial L}{\partial w_{22}^h} = \frac{\partial L}{\partial y} \frac{\partial y}{\partial w_{22}^h} = \frac{\partial L}{\partial y} \frac{\partial y}{\partial z_2} \frac{\partial z_2}{\partial w_{22}^h}
\]

\[
\frac{\partial z_2}{\partial w_{22}^h} = \frac{\partial \sigma}{\partial s} x_2
\]

\[
\frac{\partial \sigma}{\partial s} = ?
\]

\[
\sigma(s) = \frac{1}{1+\exp(-s)}
\]

\[
\sigma(s) = \max(0, s)
\]

......
Let’s review (sub-)derivative (whit board)
Backpropagation example

Hidden layer

\[ z_2 = \sigma(w_{02}^h + w_{12}^h x_1 + w_{22}^h x_2) \]

![Diagram of neural network with expressions for backpropagation]

\[
\frac{\partial L}{\partial w_{22}^h} = \frac{\partial L}{\partial y} \frac{\partial y}{\partial w_{22}^h} = \frac{\partial L}{\partial y} \frac{\partial y}{\partial z_2} \frac{\partial z_2}{\partial w_{22}^h}
\]

Take \[ \sigma(s) = \frac{1}{1 + \exp(-s)} \] as an example

\[
\frac{\partial \sigma}{\partial s} = \sigma(s)(1 - \sigma(s))
\]

\[
\frac{\partial z_2}{\partial w_{22}^h} = \frac{\partial \sigma}{\partial s} x_2 = \sigma(s)(1 - \sigma(s)) x_2
\]

The diagram shows the flow of information through the hidden layer, with each node and connection labeled with weights or variables. The backpropagation equations are shown alongside the diagram to illustrate how the gradient is calculated.
Hidden layer

Backpropagation example

\[ z_2 = \sigma(w_{02}^h + w_{12}^h x_1 + w_{22}^h x_2) \]

\[
\begin{align*}
\frac{\partial L}{\partial w_{22}^h} &= \frac{\partial L}{\partial y} \frac{\partial y}{\partial w_{22}^h} = \frac{\partial L}{\partial y} \frac{\partial y}{\partial z_2} \frac{\partial z_2}{\partial w_{22}^h} \\
\frac{\partial z_2}{\partial w_{22}^h} &= \frac{\partial \sigma}{\partial s} x_2 = \sigma(s)(1 - \sigma(s))x_2 \\
\frac{\partial L}{\partial y} &= y - y^* \\
\frac{\partial y}{\partial z_2} &= w_{21}^o \\
\frac{\partial L}{\partial w_{22}^h} &= (y - y^*) w_{21}^o \sigma(s)(1 - \sigma(s))x_2
\end{align*}
\]
Backpropagation example

Hidden layer

\[
\frac{\partial L}{\partial w_{22}^h} = \frac{\partial L}{\partial y} \frac{\partial y}{\partial z_2} \frac{\partial z_2}{\partial w_{22}^h}
\]

\[
= (y - y^*) w_{21}^o \sigma(s)(1 - \sigma(s))x_2
\]

\[
z_2 = \sigma(w_{02}^h + w_{12}^h x_1 + w_{22}^h x_2)
\]
Backpropagation example

Hidden layer

\[ z_2 = \sigma(w_{02}^h + w_{12}^h x_1 + w_{22}^h x_2) \]

\[
\frac{\partial L}{\partial w_{22}^h} = \frac{\partial L}{\partial y} \frac{\partial y}{\partial z_2} \frac{\partial z_2}{\partial w_{22}^h}
\]

\[ = (y - y^*) \, w_{21}^o \, \sigma(s)(1 - \sigma(s))x_2 \]

In general, the for any \( \frac{\partial L}{\partial w_{mn}^h} \)

You draw all \textbf{paths} from \( n \) to \( y \) (i.e., the root),

\[ \frac{\partial L}{\partial w_{mn}^h} \]

Only related to \( \partial L \) over the nodes on these \textbf{paths}
Backpropagation example

**Hidden layer**

\[
\frac{\partial L}{\partial w^h_{01}} = \frac{\partial L}{\partial y} \frac{\partial y}{\partial z_1} \frac{\partial z_1}{\partial w^h_{01}}
\]

In general, the for any

\[
\frac{\partial L}{\partial w^h_{mn}}
\]

You draw all **paths** from \( n \) to \( y \) (i.e., the root),

\[
\frac{\partial L}{\partial w^h_{mn}} \quad \text{Only related to derivatives over the nodes on these paths}
\]
In general

To compute each \( \frac{\partial L}{\partial w_{mn}^h} \)

For convenience, we generalize the notation here:

- \( h \) is the layer number,
- \( w_{mn}^h \) connects the \( m \)-th node in layer \( h - 1 \) (i.e., \( z_{m}^{h-1} \)) to the \( n \)-th node in layer \( h \) (i.e., \( z_{n}^{h} \)).

Note: the input layer is Layer 0,
If the output layer is Layer \( t \)
Then
\[
z_1^t = y
\]
In general

To compute each $\frac{\partial L}{\partial w_{mn}^h}$

1. Initialize $\frac{\partial L}{\partial w_{mn}^h} \leftarrow 0$

1. Find all the paths from $z_n^h$ to the output node $y$

2. For each path $s$

   2.1 For each node $z$ in $s$

   - Compute the partial derivative of $z$’s parent over $z$ if the node is the output node $y$, then compute $\frac{\partial L}{\partial y}$
   - Multiply all the partial derivatives along the path $s$ to obtain $g_s$
   - Add to the derivative: $\frac{\partial L}{\partial w_{mn}^h} \leftarrow \frac{\partial L}{\partial w_{mn}^h} + g_s \frac{\partial z_n^h}{\partial w_{mn}^h}$

For convenience, we generalize the notation
$h$ is the layer number, $w_{mn}^h$ connects $m$-th node in layer $h-1$ (i.e., $z_m^{h-1}$) to the $n$-th node in layer $h$ (i.e., $z_n^h$)
A more general case (see white board)
Backpropagation example

Hidden layer

\[
\frac{\partial L}{\partial w_{01}^h} = \frac{\partial L}{\partial y} \frac{\partial y}{\partial z_1} \frac{\partial z_1}{\partial w_{01}^h}
\]

More important: We have already computed many of these partial derivatives because we are proceeding from top to bottom (i.e. backwards)

Only related to derivatives over the nodes on these paths
The Backpropagation Algorithm

The same algorithm works for multiple layers

Repeated application of the chain rule for partial derivatives

- First perform forward pass from inputs to the output
- Compute loss

- From the loss, proceed backwards to compute partial derivatives over the nodes and weights using the chain rule
- Cache partial derivatives as you compute them
  - Will be used for lower layers
Look at the last example again (see white board)
Mechanizing learning

• Backpropagation gives you the gradient that will be used for gradient descent
  – SGD gives us a generic learning algorithm
  – Backpropagation is a generic method for computing partial derivatives

• A recursive algorithm that proceeds from the top of the network to the bottom

• Modern neural network libraries implement automatic differentiation using backpropagation
  – Allows easy exploration of network architectures
  – Don’t have to keep deriving the gradients by hand each time
Stochastic gradient descent

Given a training set \( S = \{ (x_i, y_i) \}, \ x \in \mathbb{R}^d \)

1. Initialize parameters \( w \)
2. For epoch = 1 ... T:
   1. Shuffle the training set
   2. For each training example \( (x_i, y_i) \in S \):
      • Treat this example as the entire dataset
      • Compute the gradient of the loss \( \nabla L(\text{NN}(x_i, w), y_i) \) using backpropagation
      • Update: \( w \leftarrow w - \gamma_t \nabla L(\text{NN}(x_i, w), y_i) \)

3. Return \( w \)

The objective is not convex. Initialization can be important.