Deep Learning Basics
Lecture 2: backpropagation

Princeton University COS 495
Instructor: Yingyu Liang
How to train the dragon?
How to get the expected output

Loss of the system

\[ l(x; \theta) = l(f_\theta, x, y) \]
How to get the expected output

Find direction $d$ so that:

$\text{Loss } l(x; \theta + d) \approx 0$
How to get the expected output

How to find $d$: $l(x; \theta + \epsilon v) \approx l(x; \theta) + \nabla l(x; \theta) * \epsilon v$ for small scalar $\epsilon$

Loss $l(x; \theta + d) \approx 0$
How to get the expected output

Move $\theta$ along $-\nabla l(x; \theta)$ for a small amount

Loss $l(x; \theta + d)$
Gradient

- Gradient of the loss is simple
  - E.g., \( l(f_\theta, x, y) = (f_\theta(x) - y)^2 / 2 \)
  - \( \frac{\partial l}{\partial \theta} = (f_\theta(x) - y) \frac{\partial f}{\partial \theta} \)
- Key part: gradient of the hypothesis
Open the box: real circuit
Single neuron

Function: $f = x_1 - x_2$
Single neuron

Function: \( f = x_1 - x_2 \)

Gradient: \( \frac{\partial f}{\partial x_1} = 1, \frac{\partial f}{\partial x_2} = -1 \)
Two neurons

Function: \( f = x_1 - x_2 = x_1 - (x_3 + x_4) \)
Two neurons

Function: \( f = x_1 - x_2 = x_1 - (x_3 + x_4) \)

Gradient: \( \frac{\partial x_2}{\partial x_3} = 1, \frac{\partial x_2}{\partial x_4} = 1 \). What about \( \frac{\partial f}{\partial x_3} \)?
Two neurons

Function: \( f = x_1 - x_2 = x_1 - (x_3 + x_4) \)

Gradient: \( \frac{\partial f}{\partial x_3} = \frac{\partial f}{\partial x_2} \frac{\partial x_2}{\partial x_3} = -1 \)
Multiple input

Function: $f = x_1 - x_2 = x_1 - (x_3 + x_5 + x_4)$

Gradient: $\frac{\partial x_2}{\partial x_5} = 1$
Function: \( f = x_1 - x_2 = x_1 - (x_3 + x_5 + x_4) \)

Gradient: \( \frac{\partial f}{\partial x_5} = \frac{\partial f}{\partial x_5} \frac{\partial x_5}{\partial x_3} = -1 \)
Weights on the edges

Function: $f = x_1 - x_2 = x_1 - (w_3 x_3 + w_4 x_4)$
Weights on the edges

Function: \( f = x_1 - x_2 = x_1 - (w_3 x_3 + w_4 x_4) \)
Weights on the edges

Function: \( f = x_1 - x_2 = x_1 - (w_3 x_3 + w_4 x_4) \)

Gradient: \( \frac{\partial f}{\partial w_3} = \frac{\partial f}{\partial x_2} \frac{\partial x_2}{\partial w_3} = -1 \times x_3 = -x_3 \)
Function: \( f = x_1 - x_2 = x_1 - \sigma(w_3x_3 + w_4x_4) \)
Activation

Function: \( f = x_1 - x_2 = x_1 - \sigma(w_3 x_3 + w_4 x_4) \)
Function: \( f = x_1 - x_2 = x_1 - \sigma(w_3 x_3 + w_4 x_4) \)

Gradient: \( \frac{\partial f}{\partial w_3} = \frac{\partial f}{\partial x_2} \frac{\partial x_2}{\partial \text{net}_2} \frac{\partial \text{net}_2}{\partial w_3} = -1 \times \sigma' \times x_3 = -\sigma' x_3 \)
Activation

Function: \( f = x_1 - x_2 = x_1 - \sigma(w_3 x_3 + w_4 x_4) \)

Gradient:
\[
\frac{\partial f}{\partial w_3} = \frac{\partial f}{\partial x_2} \frac{\partial x_2}{\partial \text{net}_2} \frac{\partial \text{net}_2}{\partial w_3} = -1 \times \sigma' \times x_3 = -\sigma' x_3
\]
Multiple paths

$$f = x_1 - x_2 = (x_1 + x_5) - \sigma(w_3 x_3 + w_4 x_4)$$
Multiple paths

Function: \( f = x_1 - x_2 = (x_1 + x_5) - \sigma(w_3 x_3 + w_4 x_4) \)
Multiple paths

Function: \( f = x_1 - x_2 = (x_3 + x_5) - \sigma(w_3 x_3 + w_4 x_4) \)

Gradient:
\[
\frac{\partial f}{\partial x_3} = \frac{\partial f}{\partial x_2} \frac{\partial net_2}{\partial x_3} + \frac{\partial f}{\partial x_1} \frac{\partial x_1}{\partial x_3} = -1 \times \sigma' \times w_3 + 1 \times 1 = -\sigma' w_3 + 1
\]
Summary

• Forward to compute $f$
• Backward to compute the gradients
Gradient descent

• Minimize loss $\hat{L}(\theta)$, where the hypothesis is parametrized by $\theta$

• Gradient descent
  • Initialize $\theta_0$
  • $\theta_{t+1} = \theta_t - \eta_t \nabla \hat{L}(\theta_t)$
Stochastic gradient descent (SGD)

• Suppose data points arrive one by one

\[ \hat{L}(\theta) = \frac{1}{n} \sum_{t=1}^{n} l(\theta, x_t, y_t), \] but we only know \( l(\theta, x_t, y_t) \) at time \( t \)

• Idea: simply do what you can based on local information
  • Initialize \( \theta_0 \)
  • \( \theta_{t+1} = \theta_t - \eta_t \nabla l(\theta_t, x_t, y_t) \)
Mini-batch

• Instead of one data point, work with a small batch of $b$ points
  \[(x_{tb+1}, y_{tb+1}), \ldots, (x_{tb+b}, y_{tb+b})\]

• Update rule
  \[
  \theta_{t+1} = \theta_t - \eta_t \nabla \left( \frac{1}{b} \sum_{1 \leq i \leq b} l(\theta_t, x_{tb+i}, y_{tb+i}) \right)
  \]

• Typical batch size: $b = 128$