Neural Network and Backpropagation Questions

Daeyoung Kim, Xintian Han

CDS, NYU

April 21, 2021

[JK] Annotated

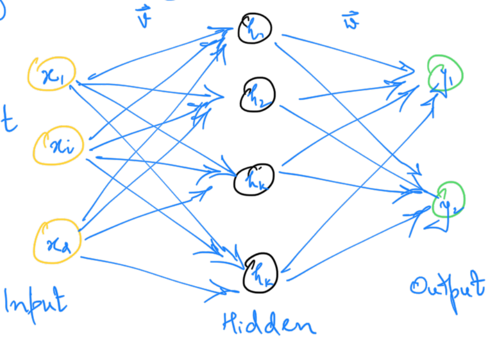
Recap: Neural Network and Backpropagation

Gradient Descent

- Feature / representation learning
- Initial idea from neuroscience
- Single hidden layer is sufficient to approximate any fn

- Special case of automatic differentiation
- Algorithm to computed gradient

partial derivatives + chain rule



$$h_k = \sigma(v_k^T x)$$

$$f(x) = \sum_{k=1}^{K} w_k h_k(x)$$

$$= \sum_{k=1}^{K} w_k \sigma(v_k^T x)$$

Suppose we have a neural network with one hidden layer.

$$f(x) = w_0 + \sum_i w_i h_i(x); \quad h_i(x) = g(b_i + v_i x),$$

where activation function g is defined as

$$\underbrace{g(z)}_{\text{step fn}} = \begin{cases} 1 & \text{if } z \geqslant 0 \\ 0 & \text{if } z < 0 \end{cases}$$

Which of the following functions can be exactly represented by this neural network?

- polynomials of degree one: $l(x) = ax + b$
- hinge loss: $l(x) = max(1 - x, 0)$
- polynomials of degree two: $l(x) = ax^2 + bx + c$
- piecewise constant functions

[1] From CMU

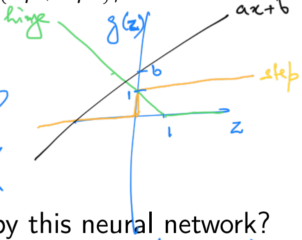
[Solution] Question 1: Step Activation Function

Suppose we have a neural network with one hidden layer.

$$f(x) = w_0 + \sum_i w_i h_i(x); \quad h_i(x) = g(b_i + v_i x),$$

where activation function g is defined as

$$g(z) = \begin{cases} 1 & \text{if } z \geqslant 0 \\ 0 & \text{if } z < 0 \end{cases}$$



Which of the following functions can be exactly represented by this neural network?

- polynomials of degree one: $l(x) = ax + b$ **No** — $f(x) = w_0 + \sum_i w_i \mathbb{1}(h_i(x) \geq 0)$
  If g can be identity function, then the answer is **Yes** — $f(x) = w_0 + wvx + wb$
  $\Rightarrow a = wv, \; b = w_0 + wb$
- hinge loss: $l(x) = max(1 - x, 0)$ **No**
- polynomials of degree two: $l(x) = ax^2 + bx + c$ **No**
- piecewise constant functions **Yes**
  $(-c) \cdot g(x - b) + (c) \cdot g(x - a)$ can represent $l(x) = c, a \leqslant x < b$.

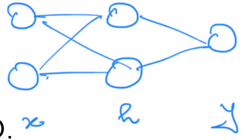
[Solution] Question 1: Step Activation Function

BLANK

Question 2: Power of ReLU [2]

Rectified Linear Unit

Consider the following small NN:



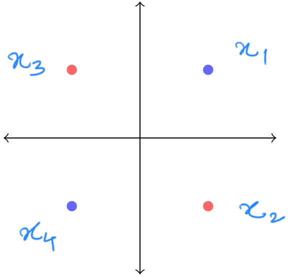
$$w_2^\top \textbf{ReLU}\left(W_1 x + b_1\right) + b_2$$

where the data is 2D, W_1 is 2 by 2, b_1 is 2D, w_2 is 2D and b_2 is 1D.

2x1 2x1

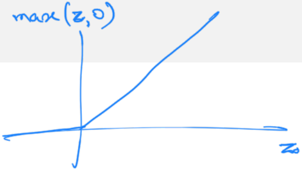
$$x_1 = (1,1)\ \ y_1 = 1; \quad x_2 = (1,-1)\ \ y_2 = -1; \quad x_3 = (-1,1)\ \ y_3 = -1; \quad x_4 = (-1,-1)\ \ y_4 = 1$$

Find b_1, b_2, W_1, w_2 to solve the problem. (Separate points from class $y = 1$ and $y = -1$.)



[2] From Harvard

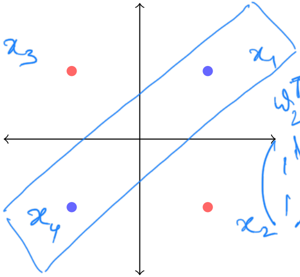
[Solution] Question 2: Power of ReLU



$$\begin{cases} z & z > 0 \\ 0 & z \le 0 \end{cases}$$

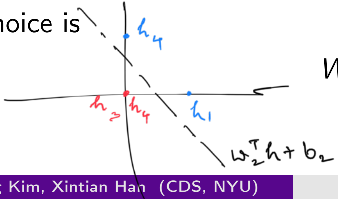
			h	$f(x)$
x_1	1	1	$[2\ \ 0]$	1
x_2	1	-1	$[0\ \ 0]$	-1
x_3	-1	1	$[0\ \ 0]$	-1
x_4	-1	-1	$[0\ \ 2]$	1

$$w_2^\top \textbf{ReLU}\,(W_1 x + b_1) + b_2$$



$$\underbrace{\begin{pmatrix} 1 \\ 1 \end{pmatrix}}_{w_2^T} \left[\underbrace{\begin{pmatrix} 1 & 1 \\ -1 & -1 \end{pmatrix}}_{W_1} \underbrace{\begin{pmatrix} \pm 1 \\ \pm 1 \end{pmatrix}}_{x} + \underbrace{\begin{pmatrix} 0 \\ 0 \end{pmatrix}}_{b_1} \right]^+ + \underbrace{(-1)}_{b_2}$$

One choice is



$$W_1 = \begin{pmatrix} 1 & 1 \\ -1 & -1 \end{pmatrix}, b_1 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

$$w_2 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, b_2 = -1$$

[Solution] Question 2: Power of ReLU

Idea: Map points into different space after performing various operations so as to make them separable / be able to classify

e.g. linear transformations / matrix multiplications

Combination

$W = U \Sigma V^T$

Singular Value Decomposition

- Rotation: orthonormal matrix
- Scaling: diagonal matrix
- Reflection: matrix determinant -ve

+

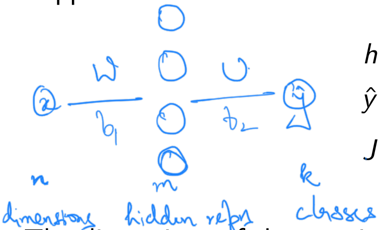
- Shearing
- Projection
- Squeeze

Affine - translation $(+b)$, non-linear transformations

Reference: https://atcold.github.io/pytorch-Deep-Learning/en/week01/01-3/

Question 3: Backpropagation [3]

Suppose we have a one hidden layer network and computation is:



non-linear affine

$$h = \text{RELU}(Wx + b1)$$

$$\hat{y} = \text{softmax}(Uh + b_2)$$

2-layer nw

$$J = \text{Cross entropy}(y, \hat{y}) = -\sum_i y_i \log \hat{y}_i$$

The dimensions of the matrices are:

$$W \in \mathbb{R}^{m \times n} \quad x \in \mathbb{R}^n \quad b_1 \in \mathbb{R}^m \quad U \in \mathbb{R}^{k \times m} \quad b_2 \in \mathbb{R}^k$$

Use backpropagation to calculate these four gradients

$$\frac{\partial J}{\partial b_2} \quad \frac{\partial J}{\partial U} \quad \frac{\partial J}{\partial b_1} \quad \frac{\partial J}{\partial W}$$

[3] From Stanford

[Solution] Question 3: Backpropagation

$$\hat{y}_i = \text{softmax}(z_2^{(i)}) = \frac{e^{z_2^{(i)}}}{\sum e^{z_2^{(i)}}}$$

$\hat{y} = \text{softmax}(z_2)$

$$z_2 = Uh + b2 \quad \delta_1 = \frac{\partial J}{\partial z_2} = \hat{y} - y$$

$$\delta_1 = \frac{\partial J}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z_2}$$

Cross-entropy ⊗ Softmax

$$\frac{\partial J}{\partial z_2} \cdot \frac{\partial z_2}{\partial b_2} \longrightarrow \frac{\partial J}{\partial b_2} = \delta_1$$

$$\frac{\partial J}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z_2} \cdot \frac{\partial z_2}{\partial U} \longrightarrow \frac{\partial J}{\partial U} = \delta_1 h^T$$

$$\frac{\partial J}{\partial h} = U^T \delta_1$$

$h = \text{ReLU}(z_1)$

$$z_1 = Wx + b_1 \quad \delta_2 = \frac{\partial J}{\partial z_1} = \underbrace{U^T \delta_1}_{\frac{\partial J}{\partial h}} \circ \underbrace{1\{h > 0\}}_{\frac{\partial h}{\partial z_1}}$$

$$\underbrace{\underbrace{\frac{\partial J}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z_2}}_{\delta_1} \cdot \underbrace{\frac{\partial z_2}{\partial h}}_{U^T}}_{} \cdot \frac{\partial h}{\partial z_1}$$

$$\frac{\partial J}{\partial z_1} \cdot \frac{\partial z_1}{\partial b_1} \longrightarrow \frac{\partial J}{\partial b_1} = \delta_2$$

$$\frac{\partial J}{\partial W} = \delta_2 x^T$$

[Solution] Question 3: Backpropagation

Softmax

$$\frac{\partial \hat{y}_j}{\partial z_2^{(i)}} = \frac{\partial}{\partial z_2^{(i)}} \left[\frac{e^{z_2^{(j)}}}{\sum_i e^{z_2^{(i)}}} \right]$$

Quotient rule

for $f(x) = \frac{g(x)}{h(x)}$

$\to f'(x) = \frac{g'(x)h(x) - h'(x)g(x)}{[h(x)]^2}$

if $i = j$

$$= \frac{e^{z_2(j)}\Sigma - e^{z_2(j)} e^{z_2(j)}}{\Sigma^2}$$

$$= \frac{e^{z_2(j)}}{\Sigma} - \left(\frac{e^{z_2^{(j)}}}{\Sigma} \right)^2$$

$$\frac{\partial \hat{y}_j}{\partial z_2^{(i)}} = \hat{y}_i - \hat{y}_i \hat{y}_i = \hat{y}_i (1 - \hat{y}_i)$$

if $i \neq j$

$$= \frac{0\,\Sigma - e^{z_2^{(i)}} e^{z_2^{(j)}}}{\Sigma^2} = -\frac{e^{z_2^{(i)}} e^{z_2^{(j)}}}{\Sigma^2} = -\hat{y}_i \hat{y}_j$$

Coding Exercise

Cross-Entropy

$$\frac{\partial J}{\partial \hat{y}_i} = \frac{\partial}{\partial \hat{y}_i}\left[-\sum_j y_j \log \hat{y}_j\right]$$

$$= -\frac{y_i}{\hat{y}_i}$$

- Computation graph hands-on

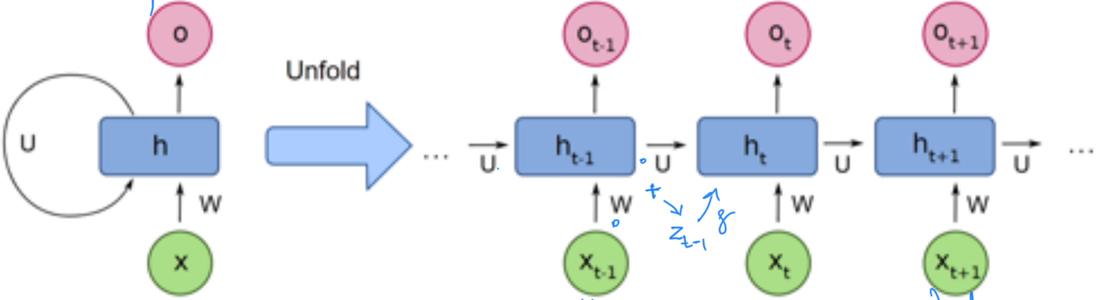
$$\frac{\partial J}{\partial z_2^{(i)}} = \frac{\partial J}{\partial \hat{y}_i} \cdot \frac{\partial \hat{y}_i}{\partial z_2^{(i)}} = \overbrace{-\frac{y_i}{\hat{y}_i}\hat{y}_i(1-\hat{y}_i)}^{i=j} - \overbrace{\sum_{j\neq i} \frac{y_j}{\hat{y}_j}(-\hat{y}_j\hat{y}_i)}^{i\neq j}$$

$$= -y_i + y_i\hat{y}_i - \sum_{j\neq i} y_j(-\hat{y}_i)$$

$$= \hat{y}_i\underbrace{\left(y_i + \sum_{j\neq i} y_j\right)}_{\sum y_i = 1} - y_i = \hat{y}_i - y_i$$

[Optional] Recurrent Neural Networks

- For processing variable length i/p, e.g. text - sentences of diff len
- Applies recursive fn to each element, e.g. words of sentence



He runs fast

Source: https://medium.com/deeplearningbrasilia/deep-learning-recurrent-neural-networks-f9482a24d010

feature + positional / temporal

- Also helps carry over information from prior time steps/position through representations learnt

[Optional]: Backpropagation in RNN

+ BPTT : through time

Suppose we have a recurrent neural network (RNN). The recursive function is:

$$\boldsymbol{z}_{t-1} = \boldsymbol{W}\boldsymbol{x}_{t-1} + \boldsymbol{U}\boldsymbol{h}_{t-1},$$
$$\boldsymbol{h}_t = g(\boldsymbol{z}_{t-1}),$$

where $\boldsymbol{h}_t$ is the hidden state and $\boldsymbol{x}_t$ is the input at time step t. $\boldsymbol{W}$ and $\boldsymbol{U}$ are the weighted matrix. g is an element-wise activation function. And $\boldsymbol{h}_0$ is a given fixed initial hidden state.

- Assume loss function $\mathcal{L}$ is a function of $\boldsymbol{h}_T$. Given $\partial\mathcal{L}/\partial\boldsymbol{h}_T$, calculate $\partial\mathcal{L}/\partial\boldsymbol{U}$ and $\partial\mathcal{L}/\partial\boldsymbol{W}$.
- Suppose g' is always greater than λ and the smallest singular value of U is larger than $1/\lambda$. What will happen to the gradient $\partial\mathcal{L}/\partial\boldsymbol{U}$ and $\partial\mathcal{L}/\partial\boldsymbol{W}$?
- Suppose g' is always smaller than λ and the largest singular value of U is smaller than $1/\lambda$. What will happen to the gradient $\partial\mathcal{L}/\partial\boldsymbol{U}$ and $\partial\mathcal{L}/\partial\boldsymbol{W}$?

[Solution] [Optional]: Backpropagation in RNN

$$\frac{\partial L}{\partial U} = \sum_t \frac{\partial L}{\partial U}; \qquad \frac{\partial L^{[T]}}{\partial U} = \frac{\partial L}{\partial h_T} \cdot \frac{\partial h_T}{\partial z_{T-1}} \cdot \frac{\partial}{\partial U}\left(W x_{T-1} + U h_{T-1}\right)$$

$$= \frac{\partial L}{\partial h_T} \cdot \underbrace{\frac{\partial h_T}{\partial z_{T-1}}}_{g'(z_{T-1})} h_{T-1}^T + \frac{\partial L}{\partial h_T} \cdot \frac{\partial h_T}{\partial z_{T-1}} \cdot U^T \cdot \boxed{\frac{\partial h_{T-1}}{\partial U}}$$

$$+ \ldots$$

[Solution] [Optional]: Backpropagation in RNN

- $$\frac{\partial \mathcal{L}}{\partial U} = \sum_{t=1}^{T} \left(\Pi_{k=t-1}^{T-1}(\boldsymbol{U}^T D_k)\right) \frac{\partial \mathcal{L}}{\partial \boldsymbol{h}_T} \boldsymbol{h}_{t-1}^T$$
  $$\frac{\partial \mathcal{L}}{\partial W} = \sum_{t=1}^{T} \left(\Pi_{k=t-1}^{T-1}(\boldsymbol{U}^T D_k)\right) \frac{\partial \mathcal{L}}{\partial \boldsymbol{h}_T} \boldsymbol{x}_{t-1}^T$$
  $D_k = \text{diag}(g'(\boldsymbol{z}_k))$ is the Jacobian matrix of the element-wise activation function.
- The smallest singular value of the $\boldsymbol{U}^T D_{k-1}$ will be greater than one. So the smallest singular value of the gradient $\frac{\partial h_s}{\partial h_t}$ will be larger than a^{s-t} for some $a > 1$. So the gradient is going to be exponentially large. This is called exploding gradient.
- The largest singular value of the $\boldsymbol{U}^T D_{k-1}$ will be smaller than one. So the largest singular value of the gradient $\frac{\partial h_s}{\partial h_t}$ will be smaller than a^{s-t} for some $a < 1$. So the gradient is going to be exponentially small. This is called vanishing gradient.