neural networks for natural language processing (nn4nlp)

Chris Kedzie
kedzie@cs.columbia.edu

October 31, 2017

who am i?

Chris Kedzie
kedzie@cs.columbia.edu
Ph.D. Candidate
(Advisor: Kathy McKeown)
Interested in text summarization, compression, and generation.

Lesson Plan

linear models

multi-layer perceptron

optimization

feed-forward language model

Lesson Plan

linear models

multi-layer perceptron

optimization

feed-forward language model

Linear Models

Whenever we want to classifiy a document, a tweet, etc., we typically train a discriminative model $p(Y|X;W)$.

Predictive models usually built around a linear decision function:

$$\sum_{i=1}^{d} W_{y,i} \cdot \phi_i(x,y) > \sum_{i=1}^{d} W_{y',i} \cdot \phi_i(x,y') \quad \forall y' \neq y$$

Linear Models

Whenever we want to classifiy a document, a tweet, etc., we typically train a discriminative model $p(Y|X;W)$.

Predictive models usually built around a linear decision function:

$$\sum_{i=1}^{d} W_{y,i} \cdot \phi_i(x,y) > \sum_{i=1}^{d} W_{y',i} \cdot \phi_i(x,y') \quad \forall y' \neq y$$

- $W \in \mathbb{R}^{|\mathcal{Y}| \times d} \triangleq$ a matrix of weights for each feature function and class label $y \in \mathcal{Y}$

Linear Models

Whenever we want to classifiy a document, a tweet, etc., we typically train a discriminative model $p(Y|X;W)$.

Predictive models usually built around a linear decision function:

$$\sum_{i=1}^{d} W_{y,i} \cdot \phi_i(x,y) > \sum_{i=1}^{d} W_{y',i} \cdot \phi_i(x,y') \quad \forall y' \neq y$$

- $W \in \mathbb{R}^{|\mathcal{Y}| \times d} \triangleq$ a matrix of weights for each feature function and class label $y \in \mathcal{Y}$
- Feature templates of the form $\phi_i : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}$

Example: Classifiying Political Tweets w/ Logistic Regression

Example: Classifiying Political Tweets w/ Logistic Regression

$$\phi(x, y) = \mathbb{1}\left\{\text{ngram}(affordable,\ care,\ act) \in x \wedge y = democrat)\right\}$$

Example: Classifiying Political Tweets w/ Logistic Regression

$$\phi(x, y) = \mathbb{1}\left\{\text{ngram}(affordable,\ care,\ act) \in x \wedge y = democrat)\right\}$$

$$\phi(x, y) = \mathbb{1}\left\{\text{ngram}(obamacare) \in x \wedge y = republican)\right\}$$

Example: Classifiying Political Tweets w/ Logistic Regression

$$\phi(x, y) = \mathbb{1}\left\{\text{ngram}(affordable,\ care,\ act) \in x \wedge y = democrat)\right\}$$

$$\phi(x, y) = \mathbb{1}\left\{\text{ngram}(obamacare) \in x \wedge y = republican)\right\}$$

$$\phi(x, y) = \mathbb{1}\left\{\text{ngram}(ACA) \in x \wedge y = democrat\right\}$$

$$\vdots$$

Example: Classifiying Political Tweets w/ Logistic Regression

$$\phi(x, y) = \mathbb{1}\{\text{ngram}(affordable,\ care,\ act) \in x \wedge y = democrat)\}$$

$$\phi(x, y) = \mathbb{1}\{\text{ngram}(obamacare) \in x \wedge y = republican)\}$$

$$\phi(x, y) = \mathbb{1}\{\text{ngram}(ACA) \in x \wedge y = democrat\}$$

$$\vdots$$

$$p(Y = democrat|X = x) = \frac{\exp\left(\sum_i W_{dem,i} \cdot \phi_i(x, dem)\right)}{\sum_{y \in \{dem, rep\}} \exp\left(\sum_i W_{y,i} \cdot \phi_i(x, y)\right)}$$

Example: Classifiying Political Tweets w/ Logistic Regression

- Designing features can be challenging in certain domains. E.g. speech recognition or image classification.

Example: Classifiying Political Tweets w/ Logistic Regression

- Designing features can be challenging in certain domains. E.g. speech recognition or image classification.
- Ineffecient parameter sharing!

Learning about

$$\phi(x, y) = \mathbb{1}\left\{\text{ngram}(affordable,\ care,\ act) \in x \wedge y = democrat)\right\}$$

doesn't tell us anything about

$$\phi(x, y) = \mathbb{1}\left\{\text{ngram}(ACA) \in x \wedge y = democrat\right\}$$

even though they may occur in similar contexts.

By comparison, neural network models will allow us to efficiently share parameters and learn useful representations.

They also have their own particular shortcomings as well!

Lesson Plan

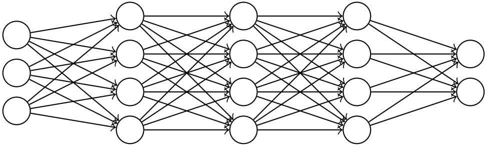
linear models

multi-layer perceptron

optimization

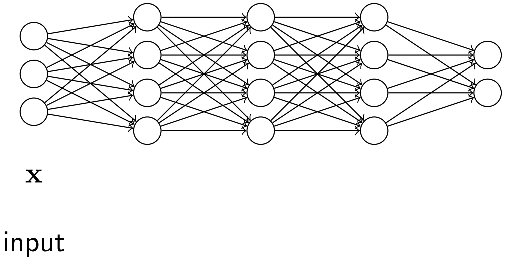
feed-forward language model

Feed-forward Neural Network



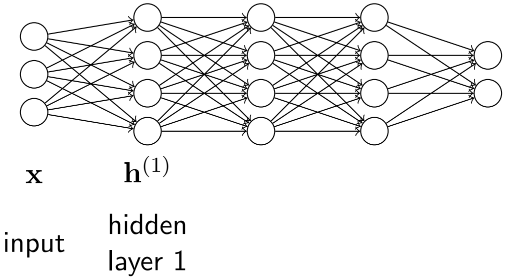
Feed-forward Neural Network

- Input is introduced to the first layer neurons.



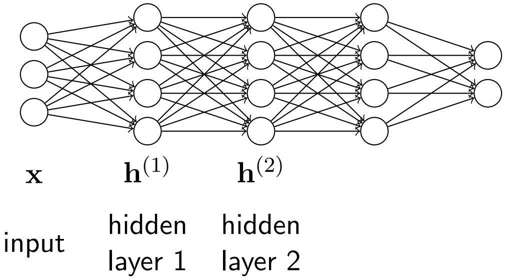
Feed-forward Neural Network

- Input is introduced to the first layer neurons.
- Each successive layer activates the next layer,



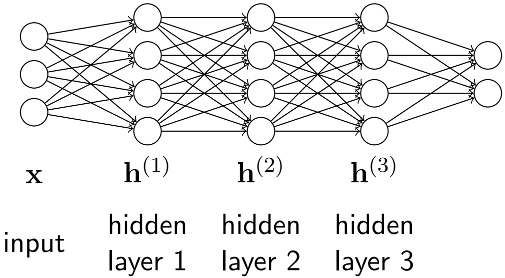
Feed-forward Neural Network

- Input is introduced to the first layer neurons.
- Each successive layer activates the next layer,



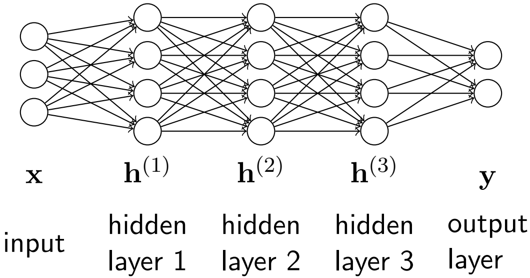
Feed-forward Neural Network

- Input is introduced to the first layer neurons.
- Each successive layer activates the next layer,



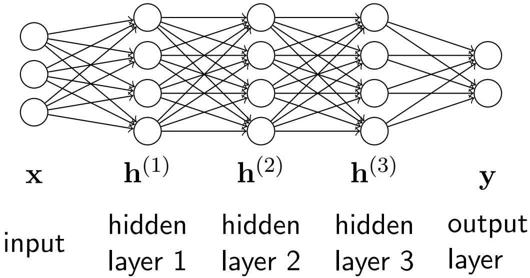
Feed-forward Neural Network

- Input is introduced to the first layer neurons.
- Each successive layer activates the next layer,
- finally, producing activations at the output neurons.



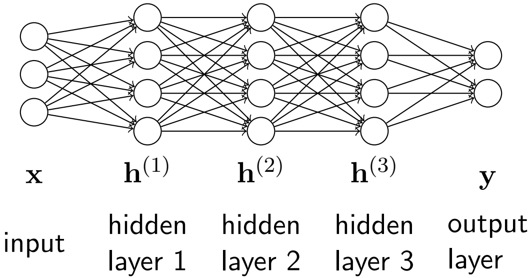
Feed-forward Neural Network

- Input is introduced to the first layer neurons.
- Each successive layer activates the next layer,
- finally, producing activations at the output neurons.
- Fully connected: each neuron in layer i connects to every neuron in layer $i + 1$.



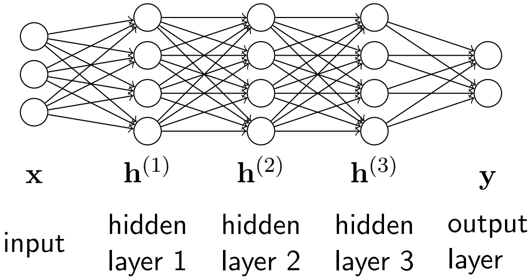
Feed-forward Neural Network

- Input is introduced to the first layer neurons.
- Each successive layer activates the next layer,
- finally, producing activations at the output neurons.
- Fully connected: each neuron in layer i connects to every neuron in layer $i+1$.
- No feedback/cycles (network is a directed acyclic graph).

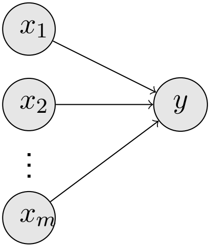


Feed-forward Neural Network

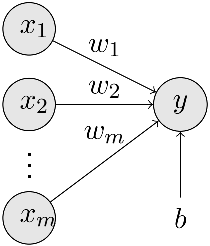
- Input is introduced to the first layer neurons.
- Each successive layer activates the next layer,
- finally, producing activations at the output neurons.
- Fully connected: each neuron in layer i connects to every neuron in layer $i+1$.
- No feedback/cycles (network is a directed acyclic graph).
- Not a generative model of the input (discriminative).



Single Layer Perceptron
(m input neurons, 1 output neuron)



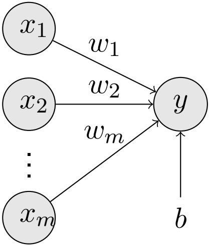
Single Layer Perceptron (m input neurons, 1 output neuron)



activation $y = f\left(\underbrace{\sum_{i=1}^{m} w_i \cdot x_i + b}_{\text{preactivation}}\right)$

- w_i indicates the strength of the connection between the input activation x_i and the output activation y.

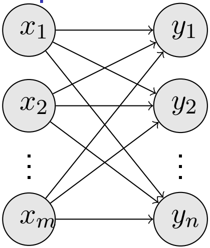
Single Layer Perceptron
(m input neurons, 1 output neuron)



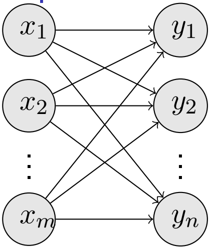
$$\text{activation } y = f\left(\underbrace{\sum_{i=1}^{m} w_i \cdot x_i + b}_{\text{preactivation}}\right)$$

- w_i indicates the strength of the connection between the input activation x_i and the output activation y.
- $f : \mathbb{R} \rightarrow \mathbb{R}$ is a nonlinear function.
  Typically, tanh, relu, sigmoid, or softmax.

Single Layer Perceptron (m input neurons, n output neuron)

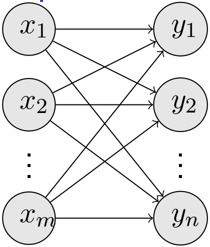


Single Layer Perceptron (m input neurons, n output neuron)



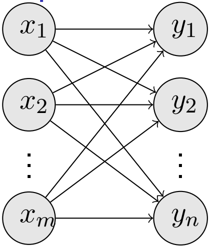
- $y_i = f\left(\sum_{j=1}^{m} w_{i,j} \cdot x_j + b_i\right)$

Single Layer Perceptron
(m input neurons, n output neuron)



- $y_i = f\left(\sum_{j=1}^{m} w_{i,j} \cdot x_j + b_i\right)$
- Equivalently, $y = f(W^\intercal x + b)$
  where $W \in \mathbb{R}^{m \times n}$, $x \in \mathbb{R}^m$, and $b \in \mathbb{R}^n$

Single Layer Perceptron
(m input neurons, n output neuron)



- $y_i = f\left(\sum_{j=1}^{m} w_{i,j} \cdot x_j + b_i\right)$
- Equivalently, $y = f(W^\intercal x + b)$
  where $W \in \mathbb{R}^{m \times n}$, $x \in \mathbb{R}^m$, and $b \in \mathbb{R}^n$
- f is applied elementwise to a vector $v \in \mathbb{R}^n$:

$$f(v) = \begin{bmatrix} f(v_1) & f(v_2) & \dots & f(v_n) \end{bmatrix}$$
$$f(v)_i = f(v_i)$$

Limitations of a single layer perceptron

- Can only learn functions where input is linearly separable.

Limitations of a single layer perceptron

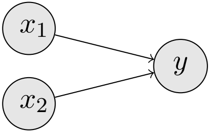
- Can only learn functions where input is linearly separable.
- E.g. Can't learn xor function.

Limitations of a single layer perceptron

- Can only learn functions where input is linearly separable.
- E.g. Can't learn xor function.
- We could design a different kernel/feature representation.

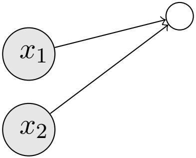
Limitations of a single layer perceptron

- Can only learn functions where input is linearly separable.
- E.g. Can't learn xor function.
- We could design a different kernel/feature representation.
- Or simply add more layers...



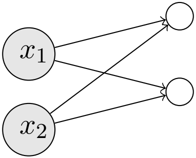
Limitations of a single layer perceptron

- Can only learn functions where input is linearly separable.
- E.g. Can't learn xor function.
- We could design a different kernel/feature representation.
- Or simply add more layers...



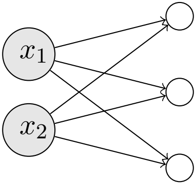
Limitations of a single layer perceptron

- Can only learn functions where input is linearly separable.
- E.g. Can't learn xor function.
- We could design a different kernel/feature representation.
- Or simply add more layers...



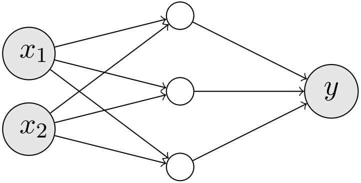
Limitations of a single layer perceptron

- Can only learn functions where input is linearly separable.
- E.g. Can't learn xor function.
- We could design a different kernel/feature representation.
- Or simply add more layers...

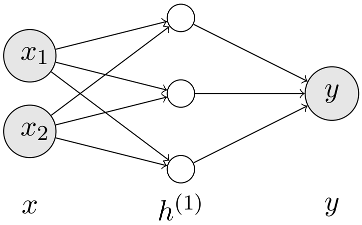


Limitations of a single layer perceptron

- Can only learn functions where input is linearly separable.
- E.g. Can't learn xor function.
- We could design a different kernel/feature representation.
- Or simply add more layers...

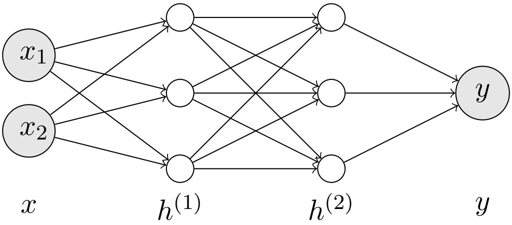


Multi-Layer Perceptron



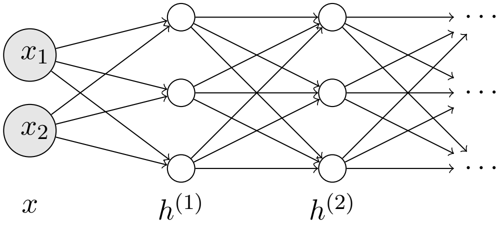
$$
\begin{aligned}
h^{(1)} &= f\left(W^{(1)} \cdot x + b^{(1)}\right) \\
y &= f\left(W^{(2)} \cdot h^{(1)} + b^{(2)}\right)
\end{aligned}
$$

Multi-Layer Perceptron



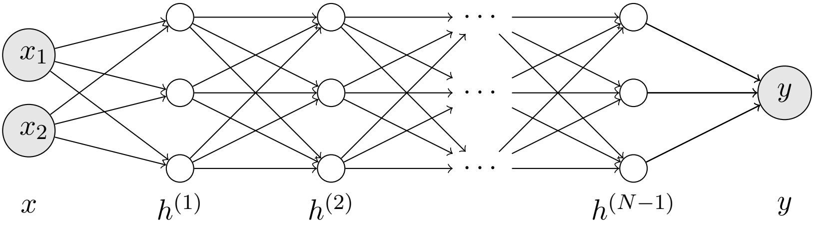
$$
\begin{aligned}
h^{(1)} &= f\left(W^{(1)} \cdot x + b^{(1)}\right) \\
h^{(2)} &= f\left(W^{(2)} \cdot h^{(1)} + b^{(2)}\right) \\
y &= f\left(W^{(3)} \cdot h^{(2)} + b^{(3)}\right)
\end{aligned}
$$

Multi-Layer Perceptron



$$
\begin{aligned}
h^{(1)} &= f\left(W^{(1)} \cdot x + b^{(1)}\right) \\
h^{(2)} &= f\left(W^{(2)} \cdot h^{(1)} + b^{(2)}\right) \\
\vdots \quad &\vdots \quad \vdots
\end{aligned}
$$

Multi-Layer Perceptron



$$
\begin{aligned}
h^{(1)} &= f\left(W^{(1)} \cdot x + b^{(1)}\right) \\
h^{(2)} &= f\left(W^{(2)} \cdot h^{(1)} + b^{(2)}\right) \\
\vdots \quad & \vdots \quad \vdots \\
y &= f\left(W^{(N)} \cdot h^{(N-1)} + b^{(N)}\right)
\end{aligned}
$$

Activation Functions

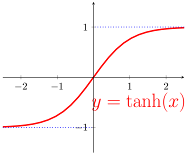
- tanh
- ReLU
- sigmoid
- softmax

There are many variants/alternative functions with different properties.

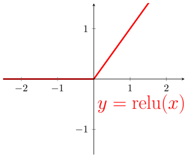
Must be continuous and differentiable (almost everywhere)

Activation Functions (hidden layers)

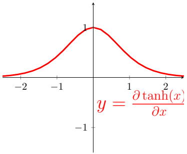
$$\tanh(x) = \frac{1-\exp(-2x)}{1+\exp(-2x)}$$



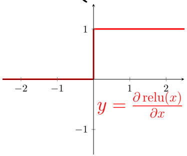
$$\text{relu}(x) = \max(0, x)$$



$$\frac{\partial \tanh(x)}{\partial x} = 1 - \tanh^2(x)$$



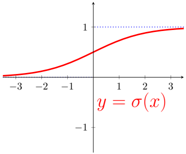
$$\frac{\partial\, \text{relu}(x)}{\partial x} = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases}$$



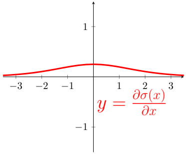
Activation Functions (hidden layers/output layers)

Logistic Sigmoid

$\sigma(x) = \frac{1}{1+\exp(-x)}$



$\frac{\partial \sigma(x)}{\partial x} = \sigma(x) \cdot (1 - \sigma(x))$



Softmax

$\sigma(x)_i = \frac{\exp(x_i)}{\sum_{i'=1}^{d} \exp(x_{i'})}$ where $x \in \mathbb{R}^d$

$$\frac{\partial \sigma(x)_i}{\partial x_j} = \begin{cases} \sigma(x)_i \cdot (1 - \sigma(x)_i) & \text{if } i = j \\ -\sigma(x)_i \cdot \sigma(x)_j & \text{if } i \neq j \end{cases}$$

Activation Functions (output layers)

- Output layer typically a sigmoid or softmax
- $a = W^{(N)} \cdot h^{(N-1)} + b^{(N)}$
- sigmoid:

$$p(Y = 1 | X = x; \theta) = \sigma(a) = \frac{1}{1 + \exp(-a)}$$

$$p(Y = 0 | X = x; \theta) = 1 - p(Y = 1 | X = x; \theta)$$

- softmax:

$$p(Y = i | X = x; \theta) = \sigma(a)_i = \frac{\exp(a_i)}{\sum_{i'} \exp(a_{i'})}$$

Lesson Plan

linear models

multi-layer perceptron

optimization

feed-forward language model

Loss Functions/Objective Functions

Cross Entropy

- Given a training dataset $\mathcal{D} = (x^{(i)}, y^{(i)})|_{i=1}^{N}$
- Multi-Class Cross Entropy loss:

$$\mathcal{L}(\theta) = -\frac{1}{N}\sum_{i} \ln p(y^{(i)}|x^{(i)};\theta)$$

- Also referred to as the negative log likelihood.

Optimization

Learning of the network parameters θ is done by minimizing the loss function with respect to θ.

$$\min_{\theta} \mathcal{L}(\theta)$$

Typically, this is done by performing some variant of **stochastic gradient descent** (SGD).

Algorithm 1 Stochastic Gradient Descent

```
Randomly initialize θ.
for EPOCH = 1 to MAXEPOCHS do
    Shuffle dataset D = (x^(i), y^(i))|_{i=1}^N
    for i = 1 to N do
        θ ← θ − η ∂L_i(θ)/∂θ
    end for
end for
```

Optimization

Learning of the network parameters θ is done by minimizing the loss function with respect to θ.

$$\min_\theta \mathcal{L}(\theta)$$

Typically, this is done by performing some variant of **stochastic gradient descent** (SGD).

Algorithm 2 Stochastic Gradient Descent

1: Randomly initialize θ.
2: **for** $\text{EPOCH} = 1$ to $\textsc{MaxEpochs}$ **do**
3: Shuffle dataset $\mathcal{D} = (x^{(i)}, y^{(i)})|_{i=1}^{N}$
4: **for** $i = 1$ to N **do**
5: $\theta \leftarrow \theta - \eta \frac{\partial \mathcal{L}_i(\theta)}{\partial \theta}$
6: **end for**
7: **end for**

η is the learning rate, typically a small value e.g. 10^{-3}

Backpropagation

To perform SGD, we need to efficiently compute $\frac{\partial \mathcal{L}_i(\theta)}{\partial \theta}$.

- Forward pass — compute the $\mathcal{L}_i(\theta)$ (e.g. the probability of y_i) given input x_i with the current θ.
  (Store intermediate outputs for backward pass)
- Backward pass — propagate the gradient of the loss backwards through the network, collecting the parameter gradients $\nabla\theta$

Chain Rule of Calculus

We want to compute the derivative of nested function $f(g(x))$ with respect to x.

By the chain rule:

$$\frac{df(g(x))}{dx} = \frac{df(g(x))}{dg(x)} \cdot \frac{dg(x)}{dx}$$

Chain Rule of Calculus

We want to compute the derivative of nested function $f(g(x))$ with respect to x.

By the chain rule:

$$\frac{df(g(x))}{dx} = \frac{df(g(x))}{dg(x)} \cdot \frac{dg(x)}{dx}$$

A concrete example:

$$f(z) = \ln z \qquad \frac{df(z)}{dz} = \frac{1}{z}$$
$$g(x) = 2x \qquad \frac{dg(x)}{dx} = 2$$

Chain Rule of Calculus

We want to compute the derivative of nested function $f(g(x))$ with respect to x.

By the chain rule:

$$\frac{df(g(x))}{dx} = \frac{df(g(x))}{dg(x)} \cdot \frac{dg(x)}{dx}$$

A concrete example:

$$f(z) = \ln z \qquad \frac{df(z)}{dz} = \frac{1}{z}$$
$$g(x) = 2x \qquad \frac{dg(x)}{dx} = 2$$

$$\frac{df(g(x))}{dx} = \frac{df(g(x))}{dg(x)} \cdot \frac{dg(x)}{dx}$$

Chain Rule of Calculus

We want to compute the derivative of nested function $f(g(x))$ with respect to x.

By the chain rule:

$$\frac{df(g(x))}{dx} = \frac{df(g(x))}{dg(x)} \cdot \frac{dg(x)}{dx}$$

A concrete example:

$$f(z) = \ln z \qquad \frac{df(z)}{dz} = \frac{1}{z}$$
$$g(x) = 2x \qquad \frac{dg(x)}{dx} = 2$$

$$\begin{aligned}\frac{df(g(x))}{dx} &= \frac{df(g(x))}{dg(x)} \cdot \frac{dg(x)}{dx} \\ &= \frac{1}{2x} \cdot 2\end{aligned}$$

Chain Rule of Calculus

We want to compute the derivative of nested function $f(g(x))$ with respect to x.

By the chain rule:

$$\frac{df(g(x))}{dx} = \frac{df(g(x))}{dg(x)} \cdot \frac{dg(x)}{dx}$$

A concrete example:

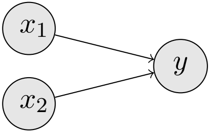
$$f(z) = \ln z \qquad \frac{df(z)}{dz} = \frac{1}{z}$$
$$g(x) = 2x \qquad \frac{dg(x)}{dx} = 2$$

$$\begin{aligned}\frac{df(g(x))}{dx} &= \frac{df(g(x))}{dg(x)} \cdot \frac{dg(x)}{dx} \\ &= \frac{1}{2x} \cdot 2 = \frac{1}{x}\end{aligned}$$

Backpropagation (Forward Pass)

Simple, 1-layer sigmoid network

- $a = w_1 \cdot x_1 + w_2 \cdot x_2 + b$
- $o = p(Y = 1|x; w_1, w_2, b) = \sigma(a)$
- $\mathcal{D} = \{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)})\}$
  where $x^{(1)}, x^{(2)} \in \mathbb{R}^2$ and $y^{(1)}, y^{(2)} \in \{0, 1\}$
- $y^{(1)} = 1$, $y^{(2)} = 0$



Backpropagation (Forward Pass)

- $a = w_1 \cdot x_1 + w_2 \cdot x_2 + b$
- $o = p(Y = 1|x; w_1, w_2, b) = \sigma(a)$
- $\mathcal{D} = \{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)})\}$
  where $x^{(1)}, x^{(2)} \in \mathbb{R}^2$ and $y^{(1)}, y^{(2)} \in \{0, 1\}$
- $y^{(1)} = 1$, $y^{(2)} = 0$

Forward Pass

Backpropagation (Forward Pass)

- $a = w_1 \cdot x_1 + w_2 \cdot x_2 + b$
- $o = p(Y = 1|x; w_1, w_2, b) = \sigma(a)$
- $\mathcal{D} = \{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)})\}$
  where $x^{(1)}, x^{(2)} \in \mathbb{R}^2$ and $y^{(1)}, y^{(2)} \in \{0, 1\}$
- $y^{(1)} = 1,\ y^{(2)} = 0$

Forward Pass

1. $a^{(1)} = w_1 \cdot x_1^{(1)} + w_2 \cdot x_2^{(1)} + b$

Backpropagation (Forward Pass)

- $a = w_1 \cdot x_1 + w_2 \cdot x_2 + b$
- $o = p(Y = 1|x; w_1, w_2, b) = \sigma(a)$
- $\mathcal{D} = \{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)})\}$
  where $x^{(1)}, x^{(2)} \in \mathbb{R}^2$ and $y^{(1)}, y^{(2)} \in \{0, 1\}$
- $y^{(1)} = 1$, $y^{(2)} = 0$

Forward Pass

1. $a^{(1)} = w_1 \cdot x_1^{(1)} + w_2 \cdot x_2^{(1)} + b$
2. $p(Y = 1|x^{(1)}) = \sigma(a^{(1)})$

Backpropagation (Forward Pass)

- $a = w_1 \cdot x_1 + w_2 \cdot x_2 + b$
- $o = p(Y = 1|x; w_1, w_2, b) = \sigma(a)$
- $\mathcal{D} = \{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)})\}$
  where $x^{(1)}, x^{(2)} \in \mathbb{R}^2$ and $y^{(1)}, y^{(2)} \in \{0, 1\}$
- $y^{(1)} = 1,\ y^{(2)} = 0$

Forward Pass

1. $a^{(1)} = w_1 \cdot x_1^{(1)} + w_2 \cdot x_2^{(1)} + b$
2. $p(Y = 1|x^{(1)}) = \sigma(a^{(1)})$
3. $p(y^{(1)}|x^{(1)}) = p(Y = 1|x^{(1)})$

Backpropagation (Forward Pass)

- $a = w_1 \cdot x_1 + w_2 \cdot x_2 + b$
- $o = p(Y = 1|x; w_1, w_2, b) = \sigma(a)$
- $\mathcal{D} = \{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)})\}$
  where $x^{(1)}, x^{(2)} \in \mathbb{R}^2$ and $y^{(1)}, y^{(2)} \in \{0, 1\}$
- $y^{(1)} = 1,\ y^{(2)} = 0$

Forward Pass

1. $a^{(1)} = w_1 \cdot x_1^{(1)} + w_2 \cdot x_2^{(1)} + b$
2. $p(Y = 1|x^{(1)}) = \sigma(a^{(1)})$
3. $p(y^{(1)}|x^{(1)}) = p(Y = 1|x^{(1)})$
4. $a^{(2)} = w_1 \cdot x_1^{(2)} + w_2 \cdot x_2^{(2)} + b$

Backpropagation (Forward Pass)

- $a = w_1 \cdot x_1 + w_2 \cdot x_2 + b$
- $o = p(Y = 1|x; w_1, w_2, b) = \sigma(a)$
- $\mathcal{D} = \{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)})\}$
  where $x^{(1)}, x^{(2)} \in \mathbb{R}^2$ and $y^{(1)}, y^{(2)} \in \{0, 1\}$
- $y^{(1)} = 1,\ y^{(2)} = 0$

Forward Pass

1. $a^{(1)} = w_1 \cdot x_1^{(1)} + w_2 \cdot x_2^{(1)} + b$
2. $p(Y = 1|x^{(1)}) = \sigma(a^{(1)})$
3. $p(y^{(1)}|x^{(1)}) = p(Y = 1|x^{(1)})$
4. $a^{(2)} = w_1 \cdot x_1^{(2)} + w_2 \cdot x_2^{(2)} + b$
5. $p(Y = 1|x^{(2)}) = \sigma(a^{(2)})$

Backpropagation (Forward Pass)

- $a = w_1 \cdot x_1 + w_2 \cdot x_2 + b$
- $o = p(Y = 1|x; w_1, w_2, b) = \sigma(a)$
- $\mathcal{D} = \{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)})\}$
  where $x^{(1)}, x^{(2)} \in \mathbb{R}^2$ and $y^{(1)}, y^{(2)} \in \{0, 1\}$
- $y^{(1)} = 1$, $y^{(2)} = 0$

Forward Pass

1. $a^{(1)} = w_1 \cdot x_1^{(1)} + w_2 \cdot x_2^{(1)} + b$
2. $p(Y = 1|x^{(1)}) = \sigma(a^{(1)})$
3. $p(y^{(1)}|x^{(1)}) = p(Y = 1|x^{(1)})$
4. $a^{(2)} = w_1 \cdot x_1^{(2)} + w_2 \cdot x_2^{(2)} + b$
5. $p(Y = 1|x^{(2)}) = \sigma(a^{(2)})$
6. $p(y^{(2)}|x^{(2)}) = 1 - p(Y = 1|x^{(1)})$

Backpropagation (Forward Pass)

- $a = w_1 \cdot x_1 + w_2 \cdot x_2 + b$
- $o = p(Y = 1|x; w_1, w_2, b) = \sigma(a)$
- $\mathcal{D} = \{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)})\}$
  where $x^{(1)}, x^{(2)} \in \mathbb{R}^2$ and $y^{(1)}, y^{(2)} \in \{0, 1\}$
- $y^{(1)} = 1$, $y^{(2)} = 0$

Forward Pass

1. $a^{(1)} = w_1 \cdot x_1^{(1)} + w_2 \cdot x_2^{(1)} + b$
2. $p(Y = 1|x^{(1)}) = \sigma(a^{(1)})$
3. $p(y^{(1)}|x^{(1)}) = p(Y = 1|x^{(1)})$
4. $a^{(2)} = w_1 \cdot x_1^{(2)} + w_2 \cdot x_2^{(2)} + b$
5. $p(Y = 1|x^{(2)}) = \sigma(a^{(2)})$
6. $p(y^{(2)}|x^{(2)}) = 1 - p(Y = 1|x^{(1)})$
7. $\mathcal{L} = -\frac{1}{2}\left[\ln p(y^{(1)}|x^{(1)}) + \ln p(y^{(2)}|x^{(2)})\right]$

Backpropagation (Backward Pass)

- $a = w_1 \cdot x_1 + w_2 \cdot x_2 + b$
- $o = p(Y = 1|x; w_1, w_2, b) = \sigma(a)$

$$\frac{\partial \mathcal{L}}{\partial w_1} = -\frac{1}{2}\left[\frac{\partial \ln p(y^{(1)}|x^{(1)})}{\partial w_1} + \frac{\partial \ln p(y^{(2)}|x^{(2)})}{\partial w_1}\right]$$

Backpropagation (Backward Pass)

- $a = w_1 \cdot x_1 + w_2 \cdot x_2 + b$
- $o = p(Y = 1|x; w_1, w_2, b) = \sigma(a)$

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial w_1} &= -\frac{1}{2}\left[\frac{\partial \ln p(y^{(1)}|x^{(1)})}{\partial w_1} + \frac{\partial \ln p(y^{(2)}|x^{(2)})}{\partial w_1}\right] \\ &= -\frac{1}{2}\left[\frac{\partial \ln p(y^{(1)}|x^{(1)})}{\partial p(y^{(1)}|x^{(1)})} \cdot \frac{\partial p(y^{(1)}|x^{(1)})}{\partial w_1} + \frac{\partial \ln p(y^{(2)}|x^{(2)})}{\partial p(y^{(2)}|x^{(2)})} \cdot \frac{\partial p(y^{(2)}|x^{(2)})}{\partial w_1}\right]\end{aligned}$$

Backpropagation (Backward Pass)

- $a = w_1 \cdot x_1 + w_2 \cdot x_2 + b$
- $o = p(Y = 1|x; w_1, w_2, b) = \sigma(a)$

$$
\begin{aligned}
\frac{\partial \mathcal{L}}{\partial w_1} &= -\frac{1}{2}\left[\frac{\partial \ln p(y^{(1)}|x^{(1)})}{\partial w_1} + \frac{\partial \ln p(y^{(2)}|x^{(2)})}{\partial w_1}\right] \\
&= -\frac{1}{2}\left[\frac{\partial \ln p(y^{(1)}|x^{(1)})}{\partial p(y^{(1)}|x^{(1)})} \cdot \frac{\partial p(y^{(1)}|x^{(1)})}{\partial w_1} + \frac{\partial \ln p(y^{(2)}|x^{(2)})}{\partial p(y^{(2)}|x^{(2)})} \cdot \frac{\partial p(y^{(2)}|x^{(2)})}{\partial w_1}\right] \\
&= -\frac{1}{2}\left[\begin{array}{c}
\frac{\partial \ln p(y^{(1)}|x^{(1)})}{\partial p(y^{(1)}|x^{(1)})} \cdot \frac{\partial p(y^{(1)}|x^{(1)})}{\partial a^{(1)}} \cdot \frac{\partial a^{(1)}}{\partial w_1} \\
+ \\
\frac{\partial \ln p(y^{(2)}|x^{(2)})}{\partial p(y^{(2)}|x^{(2)})} \cdot \frac{\partial p(y^{(2)}|x^{(2)})}{\partial a^{(2)}} \cdot \frac{\partial a^{(2)}}{\partial w_1}
\end{array}\right]
\end{aligned}
$$

Backpropagation (Backward Pass)

- $a = w_1 \cdot x_1 + w_2 \cdot x_2 + b$
- $o = p(Y = 1|x; w_1, w_2, b) = \sigma(a)$

$$
\begin{aligned}
\frac{\partial \mathcal{L}}{\partial w_1} &= -\frac{1}{2}\left[\frac{\partial \ln p(y^{(1)}|x^{(1)})}{\partial w_1} + \frac{\partial \ln p(y^{(2)}|x^{(2)})}{\partial w_1}\right] \\
&= -\frac{1}{2}\left[\frac{\partial \ln p(y^{(1)}|x^{(1)})}{\partial p(y^{(1)}|x^{(1)})} \cdot \frac{\partial p(y^{(1)}|x^{(1)})}{\partial w_1} + \frac{\partial \ln p(y^{(2)}|x^{(2)})}{\partial p(y^{(2)}|x^{(2)})} \cdot \frac{\partial p(y^{(2)}|x^{(2)})}{\partial w_1}\right] \\
&= -\frac{1}{2}\left[\begin{array}{c} \frac{\partial \ln p(y^{(1)}|x^{(1)})}{\partial p(y^{(1)}|x^{(1)})} \cdot \frac{\partial p(y^{(1)}|x^{(1)})}{\partial a^{(1)}} \cdot \frac{\partial a^{(1)}}{\partial w_1} \\ + \\ \frac{\partial \ln p(y^{(2)}|x^{(2)})}{\partial p(y^{(2)}|x^{(2)})} \cdot \frac{\partial p(y^{(2)}|x^{(2)})}{\partial a^{(2)}} \cdot \frac{\partial a^{(2)}}{\partial w_1} \end{array}\right]
\end{aligned}
$$

Backpropagation (Backward Pass)

- $a = w_1 \cdot x_1 + w_2 \cdot x_2 + b$
- $o = p(Y = 1|x; w_1, w_2, b) = \sigma(a)$

$$
\begin{aligned}
\frac{\partial \mathcal{L}}{\partial w_1} &= -\frac{1}{2}\left[\frac{\partial \ln p(y^{(1)}|x^{(1)})}{\partial w_1} + \frac{\partial \ln p(y^{(2)}|x^{(2)})}{\partial w_1}\right] \\
&= -\frac{1}{2}\left[\frac{\partial \ln p(y^{(1)}|x^{(1)})}{\partial p(y^{(1)}|x^{(1)})} \cdot \frac{\partial p(y^{(1)}|x^{(1)})}{\partial w_1} + \frac{\partial \ln p(y^{(2)}|x^{(2)})}{\partial p(y^{(2)}|x^{(2)})} \cdot \frac{\partial p(y^{(2)}|x^{(2)})}{\partial w_1}\right] \\
&= -\frac{1}{2}\left[\begin{array}{c} \frac{\partial \ln p(y^{(1)}|x^{(1)})}{\partial p(y^{(1)}|x^{(1)})} \cdot \frac{\partial p(y^{(1)}|x^{(1)})}{\partial a^{(1)}} \cdot x_1 \\ + \\ \frac{\partial \ln p(y^{(2)}|x^{(2)})}{\partial p(y^{(2)}|x^{(2)})} \cdot \frac{\partial p(y^{(2)}|x^{(2)})}{\partial a^{(2)}} \cdot x_1 \end{array}\right]
\end{aligned}
$$

Backpropagation (Backward Pass)

- $a = w_1 \cdot x_1 + w_2 \cdot x_2 + b$
- $o = p(Y = 1|x; w_1, w_2, b) = \sigma(a)$

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial w_1} &= -\frac{1}{2}\left[\frac{\partial \ln p(y^{(1)}|x^{(1)})}{\partial w_1} + \frac{\partial \ln p(y^{(2)}|x^{(2)})}{\partial w_1}\right] \\
&= -\frac{1}{2}\left[\frac{\partial \ln p(y^{(1)}|x^{(1)})}{\partial p(y^{(1)}|x^{(1)})} \cdot \frac{\partial p(y^{(1)}|x^{(1)})}{\partial w_1} + \frac{\partial \ln p(y^{(2)}|x^{(2)})}{\partial p(y^{(2)}|x^{(2)})} \cdot \frac{\partial p(y^{(2)}|x^{(2)})}{\partial w_1}\right] \\
&= -\frac{1}{2}\left[\begin{array}{c} \frac{\partial \ln p(y^{(1)}|x^{(1)})}{\partial p(y^{(1)}|x^{(1)})} \cdot \frac{\partial p(y^{(1)}|x^{(1)})}{\partial a^{(1)}} \cdot x_1 \\ + \\ \frac{\partial \ln p(y^{(2)}|x^{(2)})}{\partial p(y^{(2)}|x^{(2)})} \cdot \frac{\partial p(y^{(2)}|x^{(2)})}{\partial a^{(2)}} \cdot x_1 \end{array}\right]
\end{aligned}$$

Backpropagation (Backward Pass)

- $a = w_1 \cdot x_1 + w_2 \cdot x_2 + b$
- $o = p(Y = 1|x; w_1, w_2, b) = \sigma(a)$

$$\frac{\partial \mathcal{L}}{\partial w_1} = -\frac{1}{2}\left[\frac{\partial \ln p(y^{(1)}|x^{(1)})}{\partial w_1} + \frac{\partial \ln p(y^{(2)}|x^{(2)})}{\partial w_1}\right]$$

$$= -\frac{1}{2}\left[\frac{\partial \ln p(y^{(1)}|x^{(1)})}{\partial p(y^{(1)}|x^{(1)})} \cdot \frac{\partial p(y^{(1)}|x^{(1)})}{\partial w_1} + \frac{\partial \ln p(y^{(2)}|x^{(2)})}{\partial p(y^{(2)}|x^{(2)})} \cdot \frac{\partial p(y^{(2)}|x^{(2)})}{\partial w_1}\right]$$

$$= -\frac{1}{2}\left[\begin{array}{c} \frac{\partial \ln p(y^{(1)}|x^{(1)})}{\partial p(y^{(1)}|x^{(1)})} \cdot \sigma(a^{(1)})(1 - \sigma(a^{(1)})) \cdot x_1 \\ + \\ \frac{\partial \ln p(y^{(2)}|x^{(2)})}{\partial p(y^{(2)}|x^{(2)})} \cdot -\sigma(a^{(2)})(1 - \sigma(a^{(2)})) \cdot x_1 \end{array}\right]$$

Backpropagation (Backward Pass)

- $a = w_1 \cdot x_1 + w_2 \cdot x_2 + b$
- $o = p(Y = 1|x; w_1, w_2, b) = \sigma(a)$

$$
\begin{aligned}
\frac{\partial \mathcal{L}}{\partial w_1} &= -\frac{1}{2}\left[\frac{\partial \ln p(y^{(1)}|x^{(1)})}{\partial w_1} + \frac{\partial \ln p(y^{(2)}|x^{(2)})}{\partial w_1}\right] \\
&= -\frac{1}{2}\left[\frac{\partial \ln p(y^{(1)}|x^{(1)})}{\partial p(y^{(1)}|x^{(1)})} \cdot \frac{\partial p(y^{(1)}|x^{(1)})}{\partial w_1} + \frac{\partial \ln p(y^{(2)}|x^{(2)})}{\partial p(y^{(2)}|x^{(2)})} \cdot \frac{\partial p(y^{(2)}|x^{(2)})}{\partial w_1}\right] \\
&= -\frac{1}{2}\left[\begin{array}{c} \frac{\partial \ln p(y^{(1)}|x^{(1)})}{\partial p(y^{(1)}|x^{(1)})} \cdot \sigma(a^{(1)})(1-\sigma(a^{(1)})) \cdot x_1 \\ + \\ \frac{\partial \ln p(y^{(2)}|x^{(2)})}{\partial p(y^{(2)}|x^{(2)})} \cdot -\sigma(a^{(2)})(1-\sigma(a^{(2)})) \cdot x_1 \end{array}\right]
\end{aligned}
$$

Backpropagation (Backward Pass)

- $a = w_1 \cdot x_1 + w_2 \cdot x_2 + b$
- $o = p(Y = 1|x; w_1, w_2, b) = \sigma(a)$

$$\frac{\partial \mathcal{L}}{\partial w_1} = -\frac{1}{2}\left[\frac{\partial \ln p(y^{(1)}|x^{(1)})}{\partial w_1} + \frac{\partial \ln p(y^{(2)}|x^{(2)})}{\partial w_1}\right]$$

$$= -\frac{1}{2}\left[\frac{\partial \ln p(y^{(1)}|x^{(1)})}{\partial p(y^{(1)}|x^{(1)})} \cdot \frac{\partial p(y^{(1)}|x^{(1)})}{\partial w_1} + \frac{\partial \ln p(y^{(2)}|x^{(2)})}{\partial p(y^{(2)}|x^{(2)})} \cdot \frac{\partial p(y^{(2)}|x^{(2)})}{\partial w_1}\right]$$

$$= -\frac{1}{2}\left[\begin{array}{c} \frac{1}{\sigma(a^{(1)})} \cdot \sigma(a^{(1)})(1-\sigma(a^{(1)})) \cdot x_1 \\ + \\ \frac{1}{1-\sigma(a^{(2)})} \cdot -\sigma(a^{(2)})(1-\sigma(a^{(2)})) \cdot x_1 \end{array}\right]$$

Backpropagation (Backward Pass)

- $a = w_1 \cdot x_1 + w_2 \cdot x_2 + b$
- $o = p(Y = 1|x; w_1, w_2, b) = \sigma(a)$

$$
\begin{aligned}
\frac{\partial \mathcal{L}}{\partial w_1} &= -\frac{1}{2}\left[\frac{\partial \ln p(y^{(1)}|x^{(1)})}{\partial w_1} + \frac{\partial \ln p(y^{(2)}|x^{(2)})}{\partial w_1}\right] \\
&= -\frac{1}{2}\left[\frac{\partial \ln p(y^{(1)}|x^{(1)})}{\partial p(y^{(1)}|x^{(1)})} \cdot \frac{\partial p(y^{(1)}|x^{(1)})}{\partial w_1} + \frac{\partial \ln p(y^{(2)}|x^{(2)})}{\partial p(y^{(2)}|x^{(2)})} \cdot \frac{\partial p(y^{(2)}|x^{(2)})}{\partial w_1}\right] \\
&= -\frac{1}{2}\left[\begin{array}{c} \frac{1}{\sigma(a^{(1)})} \cdot \sigma(a^{(1)})(1-\sigma(a^{(1)})) \cdot x_1 \\ + \\ \frac{1}{1-\sigma(a^{(2)})} \cdot -\sigma(a^{(2)})(1-\sigma(a^{(2)})) \cdot x_1 \end{array}\right] \\
&= -\frac{1}{2}\left[(1-\sigma(a^{(1)})) \cdot x_1 - \sigma(a^{(2)}) \cdot x_1\right]
\end{aligned}
$$

Backpropagation (Backward Pass)

- $a = w_1 \cdot x_1 + w_2 \cdot x_2 + b$
- $o = p(Y = 1|x; w_1, w_2, b) = \sigma(a)$

$$\frac{\partial \mathcal{L}}{\partial w_1} = -\frac{1}{2}\left[(1 - \sigma(a^{(1)})) \cdot x_1 - \sigma(a^{(2)}) \cdot x_1\right]$$

Backpropagation (Backward Pass)

- $a = w_1 \cdot x_1 + w_2 \cdot x_2 + b$
- $o = p(Y = 1|x; w_1, w_2, b) = \sigma(a)$

$$\frac{\partial \mathcal{L}}{\partial w_1} = -\frac{1}{2}\left[(1 - \sigma(a^{(1)})) \cdot x_1 - \sigma(a^{(2)}) \cdot x_1\right]$$

$w_1 \leftarrow w_1 - \eta \frac{\partial \mathcal{L}}{\partial w_1}$

Backpropagation (Backward Pass)

- $a = w_1 \cdot x_1 + w_2 \cdot x_2 + b$
- $o = p(Y = 1|x; w_1, w_2, b) = \sigma(a)$

$$\frac{\partial \mathcal{L}}{\partial w_1} = -\frac{1}{2}\left[(1 - \sigma(a^{(1)})) \cdot x_1 - \sigma(a^{(2)}) \cdot x_1\right]$$

$w_1 \leftarrow w_1 - \eta \frac{\partial \mathcal{L}}{\partial w_1}$

Repeat for w_2 and b

Optimization tricks

- When implementing a model, try to fit to 100% accuracy on 1 or 2 data points.
- Decrease the learning rate with each epoch, or when the loss stops decreasing on validation data.
- Find a good initial learning rate before adjusting other hyperparameters.
- Train with dropout. (Great for image classification; YMMV for NLP)
- Even better use a different optimizer:
  - SGD with momentum or Nesterov accelerated gradient
  - rmsprop
  - adagrad
  - adadelta
  - adam

All of these take the parameter gradient as input.
For a good overview of these methods, see
`http://ruder.io/optimizing-gradient-descent/`

Lesson Plan

linear models

multi-layer perceptron

optimization

feed-forward language model

Language Modeling and you

A *language model* assigns a probability to an arbitrary sequence of word tokens.

Often used in speech recognition and machine translation.

Typically, lm's make a low-order Markov assumption.

$$p(the,\ werewolf,\ howled,\ at,\ the,\ moon) =$$

$$\begin{aligned} & p(the|\square,\ \square,\ \square) \\ \times\ & p(werewolf|\square,\ \square,\ the) \\ \times\ & p(howled|\square,\ the,\ werewolf) \\ \times\ & p(at|the,\ werewolf,\ howled) \\ \times\ & p(the|werewolf,\ howled,\ at) \\ \times\ & p(moon|howled,\ at,\ the) \end{aligned}$$

Language Modeling and you

Traditionally, the design of *ngram language models* focused on estimating terms like $p(moon|howled, \; at, \; the)$ by:

- counting occurrence of ngrams
  $(howled, \; at, \; the, \; moon)$,
  $(at, \; the, \; moon)$,
  $(the, \; moon)$,
  $(moon)$
- interpolating lower order models built on these counts

Unfortunately, these counts are sparse (especially beyond trigrams)

Observing $(barked, \; at, \; the, \; moon)$ doesn't tell us much about $(howled, \; at, \; the, \; moon)$

A Feedforward Language Model

A Neural Probabilistic Language Model (Bengio et al., 2003)

The main ideas (copied verbatim from the paper):

1. associate with each word in the vocabulary a distributed *word feature vector* (a real-valued vector in $\mathbb{R}^m$),
2. express the joint *probability function* of word sequences in terms of the feature vectors of these words in the sequence, and
3. learn simultaneously the *word feature vectors* and the parameters of that *probability function*.

loss $\mathcal{L} = -\ln o_y$

Softmax Layer (output)

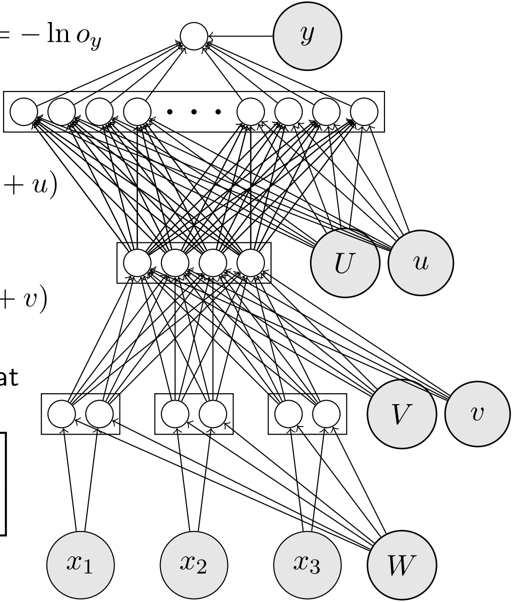
$$o = \text{softmax}(Uh^{(2)} + u)$$

Hidden Layer

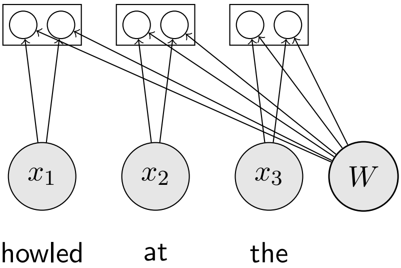
$$h^{(2)} = \text{relu}(Vh^{(1)} + v)$$

Lookup + Concat Layer

$$h^{(1)} = \begin{bmatrix} Wx_1 \\ Wx_2 \\ Wx_3 \end{bmatrix}$$



Lookup and Concat Layer (View 1)



$$\begin{bmatrix} 0 \\ 1 \\ 0 \\ \vdots \end{bmatrix} \quad \begin{bmatrix} 0 \\ 0 \\ 1 \\ \vdots \end{bmatrix} \quad \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \end{bmatrix}$$

Each word is encoded as a one-hot vector $x_i \in \mathbb{R}^V$.

Word embeddings $W \in \mathbb{R}^{m \times V}$

Lookup and Concat Layer (View 1)

$$\begin{bmatrix} W_{1,1} & W_{1,2} & \dots & W_{1,V} \\ W_{2,1} & W_{2,2} & \dots & W_{2,V} \\ \vdots & \vdots & \ddots & \vdots \\ W_{m-1,1} & W_{m-1,2} & \dots & W_{m-1,V} \\ W_{m,1} & W_{m,2} & \dots & W_{m,V} \end{bmatrix} \times \begin{bmatrix} 0 \\ 1 \\ \vdots \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} W_{1,2} \\ W_{2,2} \\ \vdots \\ W_{m-1,2} \\ W_{m,2} \end{bmatrix}$$

$$W \cdot x_1 = W_{:,2}$$

Lookup and Concat Layer (View 1)

$$\begin{bmatrix} W_{1,1} & W_{1,2} & \cdots & W_{1,V} \\ W_{2,1} & W_{2,2} & \cdots & W_{2,V} \\ \vdots & \vdots & \ddots & \vdots \\ W_{m-1,1} & W_{m-1,2} & \cdots & W_{m-1,V} \\ W_{m,1} & W_{m,2} & \cdots & W_{m,V} \end{bmatrix} \times \begin{bmatrix} 0 \\ 1 \\ \vdots \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} W_{1,2} \\ W_{2,2} \\ \vdots \\ W_{m-1,2} \\ W_{m,2} \end{bmatrix}$$

$$W \cdot x_1 = W_{:,2}$$

Lookup and Concat Layer (View 1)

Each individual embedding is then concatenated into a larger single vector.

$$h^{(1)} = \begin{bmatrix} W \cdot x_1 \\ W \cdot x_2 \\ W \cdot x_3 \end{bmatrix}$$

Lookup and Concat Layer (View 1)

Each individual embedding is then concatenated into a larger single vector.

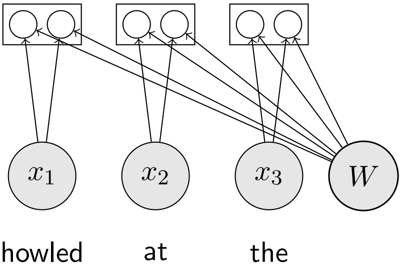
$$h^{(1)} = \begin{bmatrix} W \cdot x_1 \\ W \cdot x_2 \\ W \cdot x_3 \end{bmatrix} = \begin{bmatrix} W_{:,2} \\ W_{:,3} \\ W_{:,1} \end{bmatrix}$$

Lookup and Concat Layer (View 1)

Each individual embedding is then concatenated into a larger single vector.

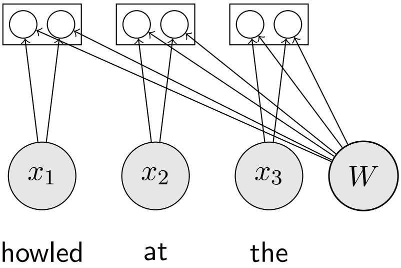
$$h^{(1)} = \begin{bmatrix} W \cdot x_1 \\ W \cdot x_2 \\ W \cdot x_3 \end{bmatrix} = \begin{bmatrix} W_{:,2} \\ W_{:,3} \\ W_{:,1} \end{bmatrix} = \begin{bmatrix} W_{1,2} \\ \vdots \\ W_{m,2} \\ W_{1,3} \\ \vdots \\ W_{m,3} \\ W_{1,1} \\ \vdots \\ W_{m,1} \end{bmatrix}$$

Lookup and Concat Layer (View 2)



$x_1 = \text{index}(\textit{howled}) = 2$
$x_2 = \text{index}(\textit{at}) = 3$
$x_3 = \text{index}(\textit{the}) = 1$

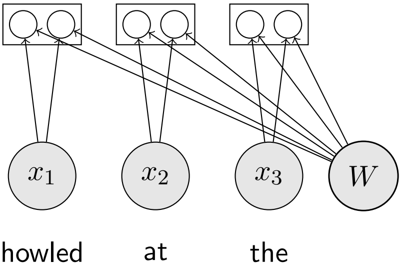
Lookup and Concat Layer (View 2)



$$x_1 = \text{index}(howled) = 2$$
$$x_2 = \text{index}(at) = 3$$
$$x_3 = \text{index}(the) = 1$$

$$\text{lookup}(W, i) = W_{:,i}$$

Lookup and Concat Layer (View 2)



$$x_1 = \text{index}(howled) = 2$$
$$x_2 = \text{index}(at) = 3$$
$$x_3 = \text{index}(the) = 1$$

$$\text{lookup}(W, i) = W_{:,i}$$

$$h^{(1)} = \begin{bmatrix} \text{lookup}(W, x_1) \\ \text{lookup}(W, x_2) \\ \text{lookup}(W, x_3) \end{bmatrix} = \begin{bmatrix} W_{:,2} \\ W_{:,3} \\ W_{:,1} \end{bmatrix}$$

loss $\mathcal{L} = -\ln o_y$

Softmax Layer (output)

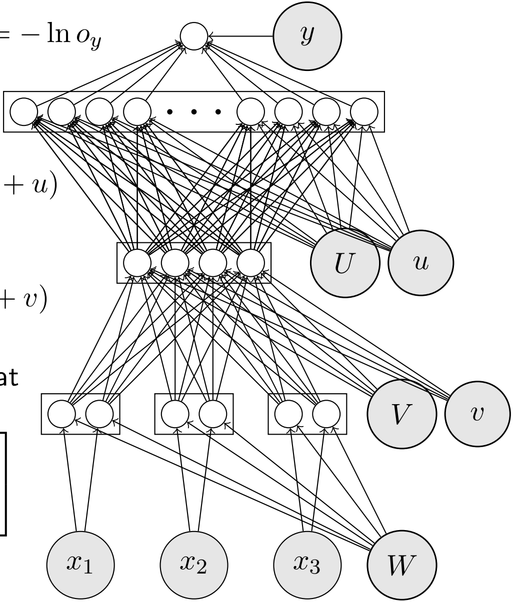
$o = \text{softmax}(Uh^{(2)} + u)$

Hidden Layer

$h^{(2)} = \text{relu}(Vh^{(1)} + v)$

Lookup + Concat Layer

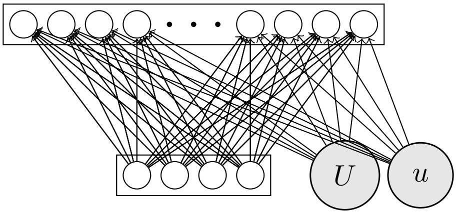
$$h^{(1)} = \begin{bmatrix} Wx_1 \\ Wx_2 \\ Wx_3 \end{bmatrix}$$



Softmax Layer

Softmax Layer
(output)
$o = \text{softmax}(Uh^{(2)} + u)$

Hidden Layer $h^{(2)}$



$h^{(2)} \in \mathbb{R}^d$, encoding of input word prefix into a vector space

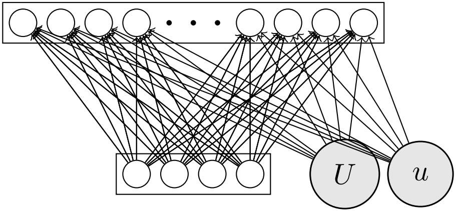
The ouput layer $o \in (0,1)^V$ contains one neuron (unit) for every word in the vocabulary.

o_i represents the probability of the i-th word in the vocabulary occurring after the word prefix represented by (x_1, x_2, x_3).

Softmax Layer

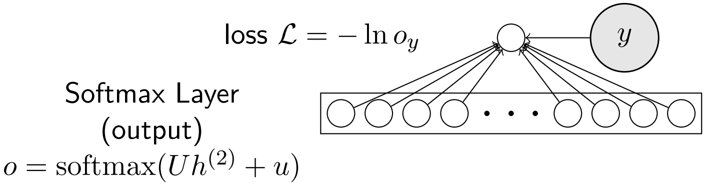
Softmax Layer (output)
$o = \text{softmax}(Uh^{(2)} + u)$

Hidden Layer $h^{(2)}$



$U \in \mathbb{R}^{V \times d}$ is also a matrix of word embeddings.

Loss Layer

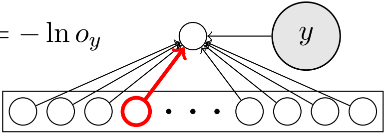


To compute the negative log likelihood (cross entropy loss), simply pick out the y-th element of o and take the negative log.

Loss Layer

loss $\mathcal{L} = -\ln o_y$

Softmax Layer
(output)
$o = \text{softmax}(Uh^{(2)} + u)$



To compute the negative log likelihood (cross entropy loss), simply pick out the y-th element of o and take the negative log.

E.g. $y = 4$, $-\ln o_y = -\ln o_4$

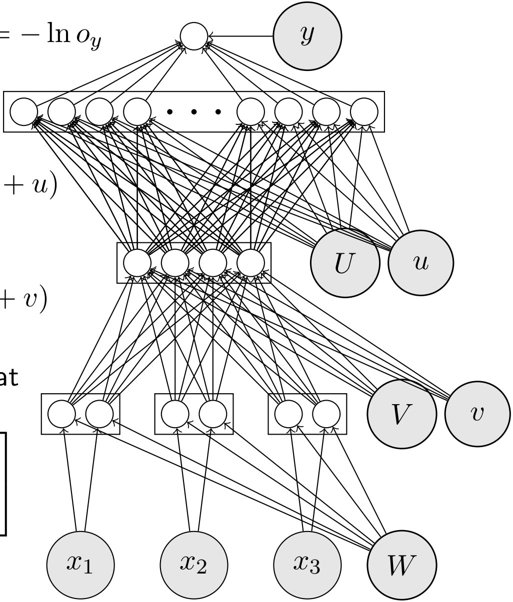
loss $\mathcal{L} = -\ln o_y$

Softmax Layer
(output)
$o = \text{softmax}(Uh^{(2)} + u)$

Hidden Layer
$h^{(2)} = \text{relu}(Vh^{(1)} + v)$

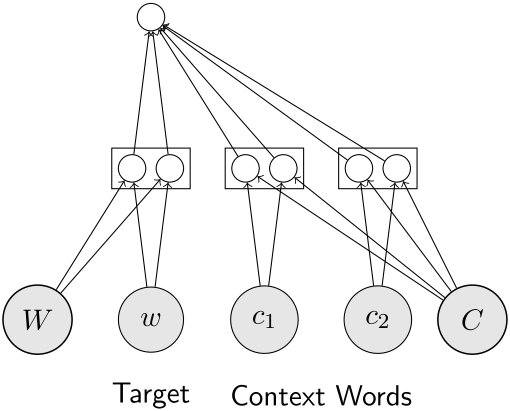
Lookup + Concat
Layer

$$h^{(1)} = \begin{bmatrix} Wx_1 \\ Wx_2 \\ Wx_3 \end{bmatrix}$$

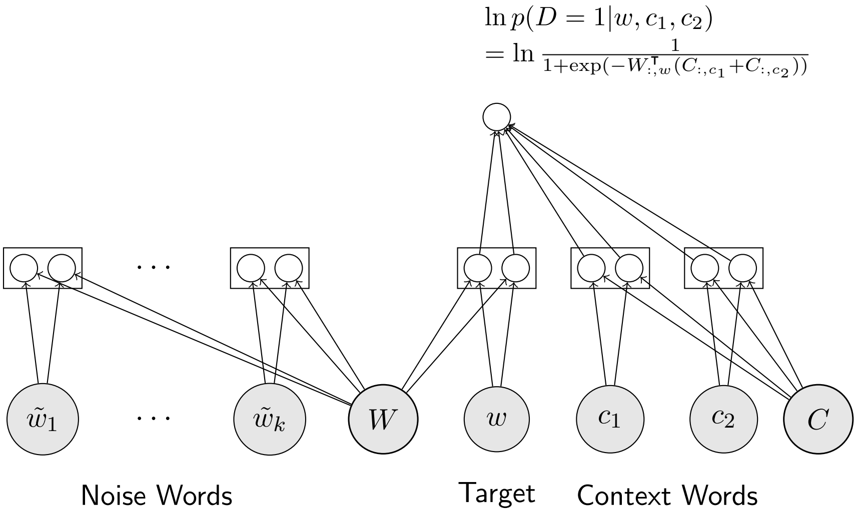


CBOW

$$\ln p(D = 1|w, c_1, c_2)$$
$$= \ln \frac{1}{1+\exp(-W_{:,w}^\intercal (C_{:,c_1} + C_{:,c_2}))}$$



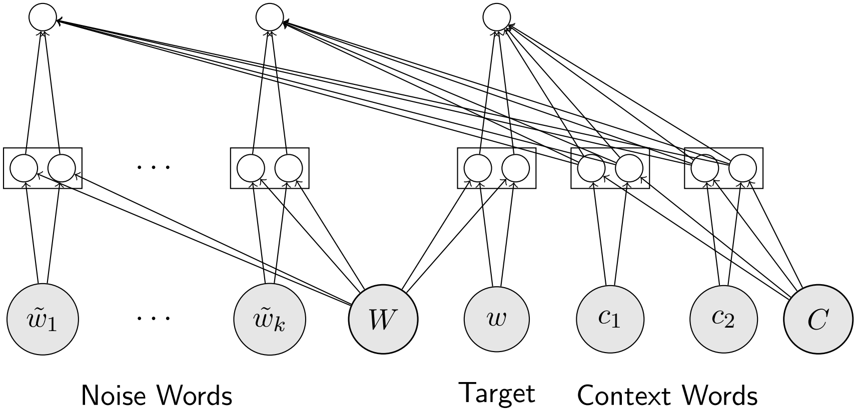
CBOW


$\ln p(D = 1|w, c_1, c_2)$
$= \ln \frac{1}{1+\exp(-W_{:,w}^{\intercal}(C_{:,c_1}+C_{:,c_2}))}$
. . .
$\tilde{w}_1$
. . .
$\tilde{w}_k$
W
w
c_1
c_2
C
Noise Words
Target
Context Words

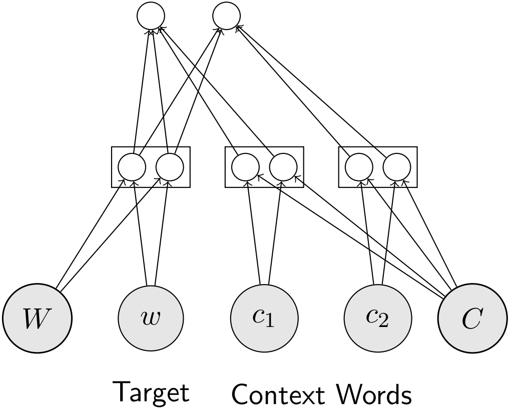
CBOW

$$\sum_{i=1}^{k} \ln p(D=0|\tilde{w}_i, c_1, c_2) = \sum_{i=1}^{k} \ln \left(1 - \frac{1}{1+\exp(-W_{:,\tilde{w}_i}^{\intercal}(C_{:,c_1}+C_{:,c_2}))}\right)$$

$$\ln p(D=1|w, c_1, c_2) = \ln \frac{1}{1+\exp(-W_{:,w}^{\intercal}(C_{:,c_1}+C_{:,c_2}))}$$

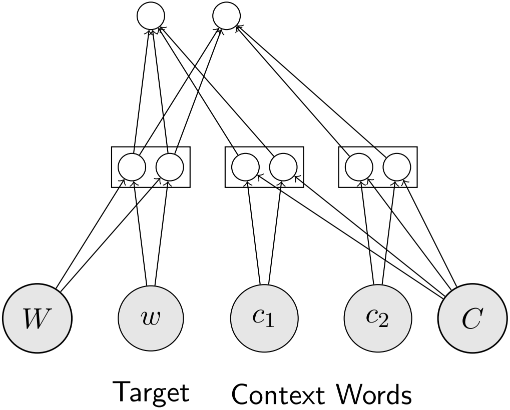


Skip-Gram



Skip-Gram

$$\ln p(D=1|w,c_1,c_2)$$
$$= \ln p(D=1|w,c_1) + \ln p(D=1|w,c_2)$$
$$= \sum_{i=1}^{2} \ln \frac{1}{1+\exp(-W_{:,w}^{\intercal} C_{:,c_i})}$$

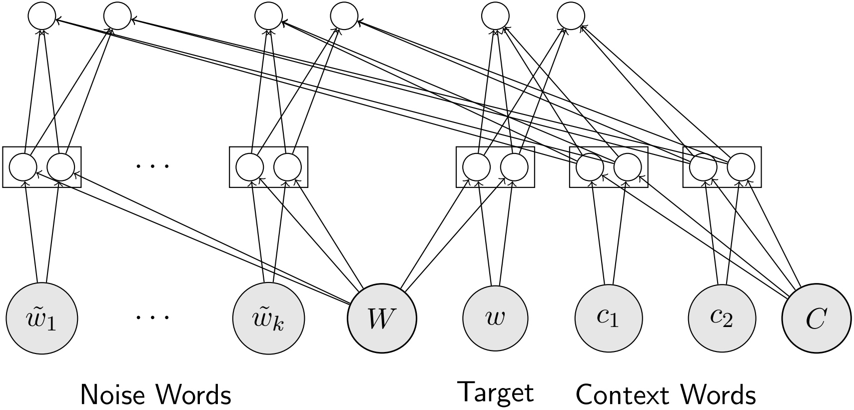


Skip-Gram

$\ln p(D = 1|w, c_1, c_2)$

$= \ln p(D = 1|w, c_1) + \ln p(D = 1|w, c_2)$

$= \sum_{i=1}^{2} \ln \frac{1}{1+\exp(-W_{:,w}^{\intercal} C_{:,c_i})}$



Skip-Gram

$$\sum_{i=1}^{k} \ln p(D=0|\tilde{w}_i, c_1, c_2)$$
$$= \sum_{i=1}^{k} \ln p(D=0|\tilde{w}_i, c_1) + \ln p(D=0|\tilde{w}_i, c_2)$$
$$= \sum_{i=1}^{k} \sum_{j=1}^{2} \ln \left(1 - \frac{1}{1+\exp(-W_{:,\tilde{w}_i}^{\mathsf{T}} C_{:,c_j})}\right)$$

$$\ln p(D=1|w, c_1, c_2)$$
$$= \ln p(D=1|w, c_1) + \ln p(D=1|w, c_2)$$
$$= \sum_{i=1}^{2} \ln \frac{1}{1+\exp(-W_{:,w}^{\mathsf{T}} C_{:,c_i})}$$