CS 224S / LINGUIST 285
Spoken Language Processing

Andrew Maas
Stanford University
Spring 2017

Lecture 6: Course Project Introduction and Deep Learning Preliminaries

Outline for Today

- Course projects
  - What makes for a successful project
  - Leveraging existing tools
  - Project archetypes and considerations
  - Discussion
- Deep learning preliminaries

Silence models for HMM-GMM

- SIL is a phoneme to a recognizer
- Always inserted at start and end of utterance
- Corrupting silence with bad forced alignments can break recognizer training (silence eats everything)
- The sound of silence
  - Turns out to be difficult to model!
  - Silence GMM models must capture lots of noise artifacts, breathing, laughing (depending on data transcription standards)
  - Microphones in the wild with background noise make SIL/non-speech even more difficult
- Special models for silence transition since we often stay there a long time

Course project goals

- A substantial piece of work related to topics specific to this course
- A successful project
  - Results in most of a conference paper submission if academically oriented
  - A portfolio item / work sample for job interviews related to ML, NLP, or SLP
- Reflects deeper understanding of SLP technology than simply applying existing API's for ASR, voice commands, etc.
- No midterm or final exam to allow more focus on projects

A successful project

- Course-relevant topic. Proposed experiments or system address a challenging, unsolved SLP problem
- Proposes and executes a sensible approach informed by previous related work
- Performs error analysis to understand what aspects of the system are good/bad
- Adapts system or introduces new hypotheses/components based on initial error analysis
- Goes beyond simply combining existing components / tools to solve a standard problem

Complexity and focus

- SLP systems are some of the most complex in AI
- Example: A simple voice command system contains:
  - Speech recognizer (Language model, pronunciation lexicon, acoustic model, decoder, lots of training options)
  - Intent/command slot filling (some combination of lexicon, rules, and ML to handle variation)
- Get a complete baseline system working by milestone
- Focus on a subset of all areas to make a bigger contribution there. APIs/tools are a great choice for areas not directly relevant to your focus

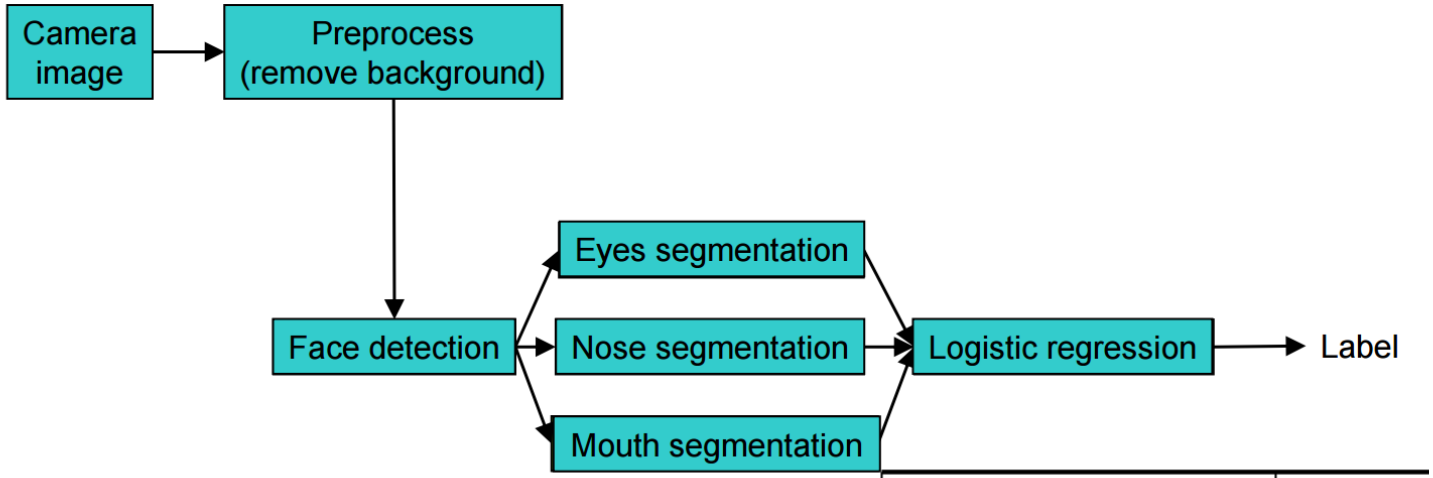
Balancing scale and depth

- Working on “real” scale datasets/problems is a plus…
- But don’t let scale distract from getting to the meat of your technical contribution
- Example:
  - Comparing some neural architectures for end-to-end speech recognition
  - Case 1: Use WSJ. Medium sized corpus, read speech. SOTA error rates ~3%
  - Case 2: Use Switchboard: Large, conversational corpus. SOTA error rates ~15%
- Case 2 stronger overall *if you run the same experiments / error analysis. Don’t let scale prevent “thoughtful loops”*

Thoughtful loops

- A single loop:
  - Try something reasonable
  - Perform relatively detailed error analysis using what we know from the course
  - Propose a modification / new experiment based on what you find
  - Try it!
  - Repeat above
- A successful project does this at least once
- Scale introduces risk of overly slow loops
- Ablative analysis or oracle experiments are a great way to guide what system component to work on

Oracle experiments



How much error is attributable to each of the components?

Plug in ground-truth for each component, and see how accuracy changes.

Conclusion: Most room for improvement in face detection and eyes segmentation.

Component	Accuracy
Overall system	85%
Preprocess (remove background)	85.1%
Face detection	91%
Eyes segmentation	95%
Nose segmentation	96%
Mouth segmentation	97%
Logistic regression	100%

Slide from Andrew Ng's CS229 lecture on applying ML
http://cs229.stanford.edu/materials/ML-advice.pdf

Ablation experiments

Error analysis tries to explain the difference between current performance and perfect performance.

Ablative analysis tries to explain the difference between some baseline (much poorer) performance and current performance.

E.g., Suppose that you've build a good anti-spam classifier by adding lots of clever features to logistic regression:

- Spelling correction.
- Sender host features.
- Email header features.
- Email text parser features.
- Javascript parser.
- Features from embedded images.

Question: How much did each of these components really help?

Slide from Andrew Ng's CS229 lecture on applying ML
http://cs229.stanford.edu/materials/ML-advice.pdf

Ablation experiments

Simple logistic regression without any clever features get 94% performance.

Just what accounts for your improvement from 94 to 99.9%?

Ablative analysis: Remove components from your system one at a time, to see how it breaks.

Component	Accuracy	
Overall system	99.9%	
Spelling correction	99.0	
Sender host features	98.9%	
Email header features	98.9%	
Email text parser features	95%	
Javascript parser	94.5%	
Features from images	94.0%	[baseline]

Conclusion: The email text parser features account for most of the improvement.

Slide from Andrew Ng's CS229 lecture on applying ML
http://cs229.stanford.edu/materials/ML-advice.pdf

Pitfalls in project planning

- Data!
  - What dataset will you use for your task?
  - If you need to collect data, why? Understand that a project with a lot of required data collection creates high risk of not being able to execute enough loops
  - Do you really need to collect data? Really?
- Overly complex baseline system
- Relying on external tools to the point that connecting them becomes the entire effort and makes innovation hard
- Off-topic. Could this be a CS 229 project instead?

Deliverables

- All projects
  - Proposal: What task, dataset, evaluation metrics and approach outline?
  - Milestone: Have you gotten your data and built a baseline for your task?
  - Final paper: Methods, results, related work, conclusions. Should read like aconference paper
- Audio/Visual material
  - Include links to audio samples for TTS. Screen capture videos for dialog interactions (spoken dialog especially)
  - Much easier to understand your contribution this way than leave us to guess. Even if it doesn't quite work.
  - Available on laptop at poster session (live demo!)

Leveraging existing tools

- Free to use any tool, but realize using the Google speech API does not constitute ‘building a recognizer’
- Ensure the tool does not prevent trying the algorithmic modifications of interest (e.g. can’t do acoustic model research on speech API’s)
- Projects that combine existing tools in a straightforward way should be avoided
- Conversely, almost every project can *and should* use some form of tool:
  - Tensorflow, speech API, language model toolkit, Kaldi, etc.
- Use tools to focus on your project hypotheses

Error analysis with tools

- Project writeup / presentation should be able to explain:
  - What goal does this tool achieve for our system?
  - Is the tool a source of errors? (e.g. oracle error rate for a speech API)
  - How could this tool be modified / replaced to improve the system? (maybe it is perfect and that's okay)
- As with any component, important to isolate sources of errors
- Work with tools in a way that reflects your deeper understanding of what they do internally (e.g. n-best lists)

Sample of tools and APIs

- Speech APIs: Google, IBM, Microsoft all have options
  - Varying levels of customization and conveying n-best
- Speech synthesis APIs: same as speech + Festival
- Slack or Facebook for text dialog interfaces
  - Slack allows downloading of historical data which could help train systems
  - Howdy.ai / botkit for integration
- Intent recognition APIs
  - Wit.ai, API.ai. Amazon Alexa

Sample project archetypes

Speech recognition research

- Benchmark corpus (WSJ, Switchboard, noisy ASR on CHIME)
- Baseline system in Kaldi. State of the art known
- Template very amenable to publication in speech or machine learning conferences
- Can be very difficult to improve on state of the art. The best systems have a lot of heuristics that might not be in Kaldi
- Systems can be cumbersome to train
- Lots of algorithmic variations to try
- Successful projects do not need to improve on best existing results

Speech synthesis

- Blizzard challenge provides training data and systems for comparison
- Evaluation is difficult. No single metric
- Matching state of the art can be very tedious signal processing
- Open realm of experiments to try, especially working to be expressive or improve prosody
- Relatively large systems without the convenience of a tool like Kaldi

Extracting affect from speech

- Beyond transcription, understanding emotion, accent, or mental state (intoxication, depression, Parkinson's etc.)
- Very dataset dependent. How will you access labeled data to train a system?
- Can't be just a classifier. Need to use insights from this course or combine with speech recognition
- Should be spoken rather than just written text

Dialog systems

- Build a dialog system for a task that interests you (bartender, medical guidance, chess)
- *Must be multi-turn*. Not just voice commands or single slot intent recognizers
- Evaluation is difficult, likely will have to collect any training data yourself
- Don’t over-invest in knowledge engineering
- Lots of room to be creative and design interactions to hide system limitations
- More difficult to publish smaller scale systems, but make for great demos / portfolio items

Deep learning approaches

- Active area of research for every area of SLP
- Beware:
  - Do you have enough training data compared to the most similar paper to your approach?
  - Do you have enough compute power?
  - How long will a single model take to train? Think about your time to complete one ‘loop’
- Ensure you are doing SLP experiments not just tuning neural nets for a dataset
- Hot area for academic publications at the moment

Summary

- Have fun
- Build something you're proud of
- Project ideas posted to Piazza by Friday and more through next week

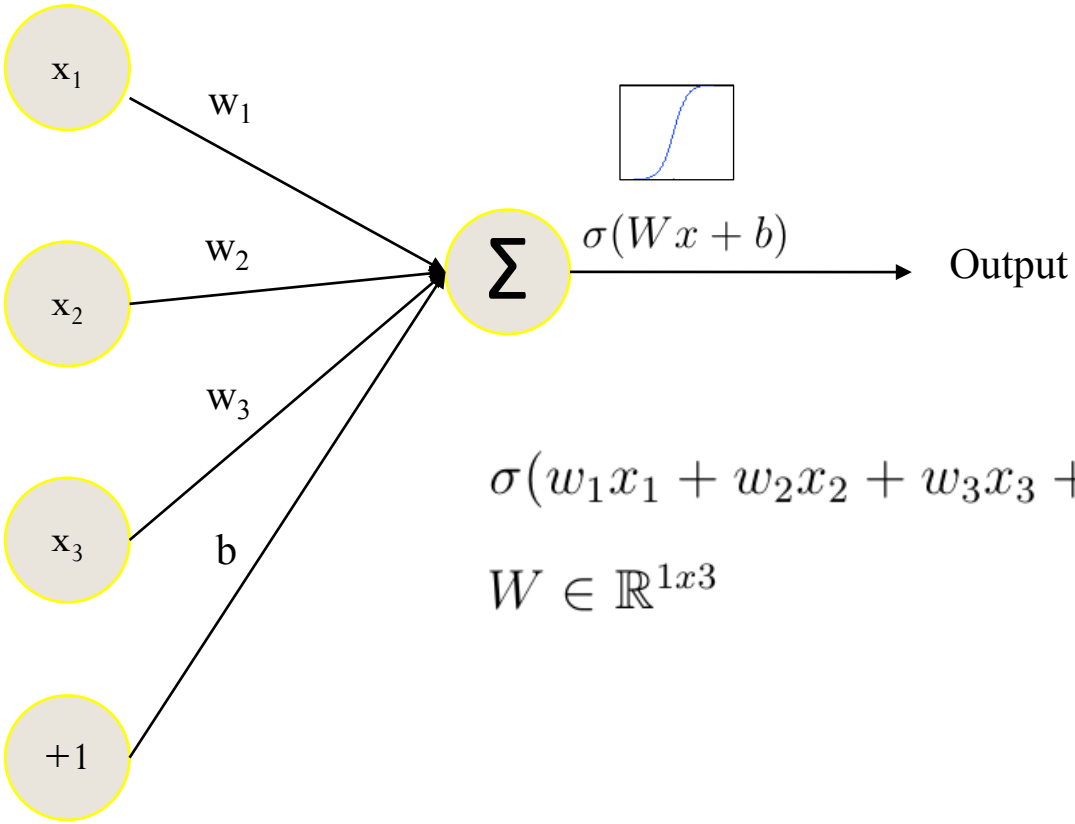
Discussion/Questions

Outline for Today

- Course projects
  - What makes for a successful project
  - Leveraging existing tools
  - Project archetypes and considerations
  - Discussion
- **Deep learning preliminaries**

Neural Network Basics: Single Unit

Logistic regression as a "neuron"



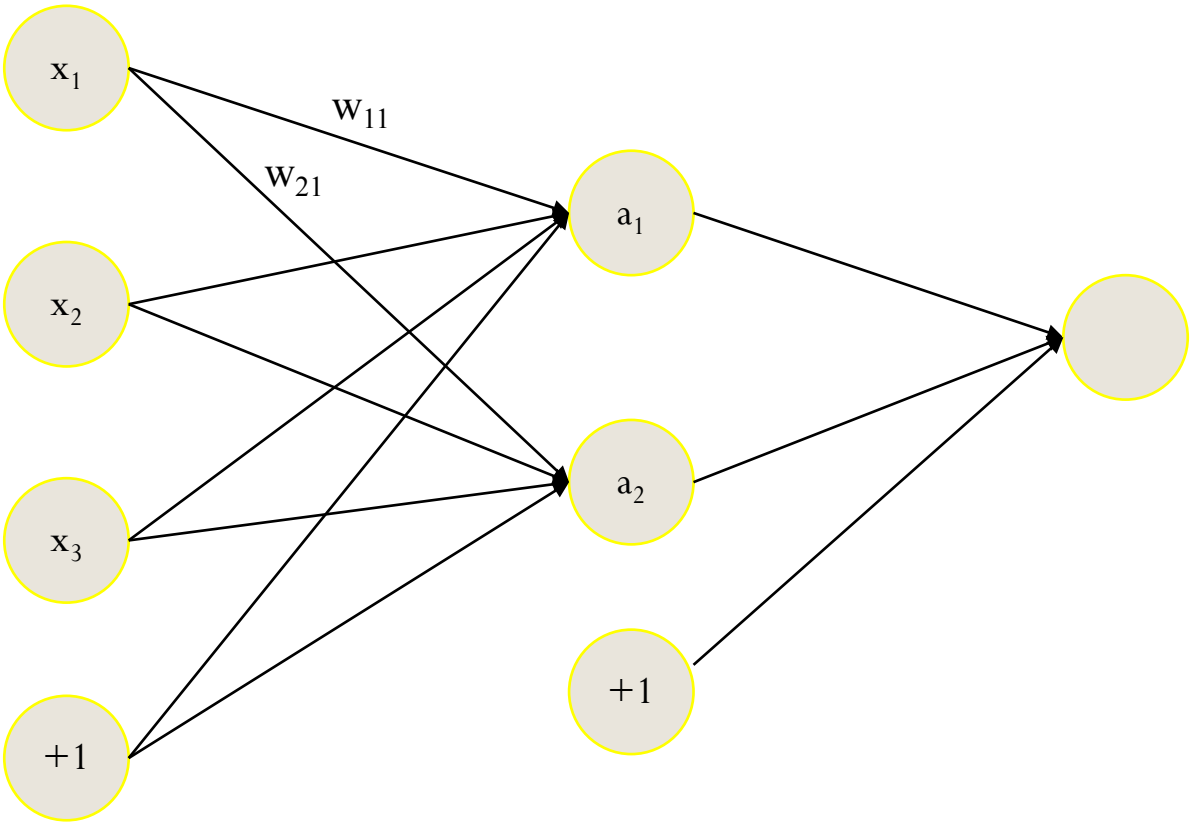
$$\sigma(w_1x_1 + w_2x_2 + w_3x_3 + b) = \sigma(Wx + b)$$

$$W \in \mathbb{R}^{1x3}$$

Slides from Awni Hannun (CS221 Autumn 2013)

Single Hidden Layer Neural Network

Stack many logistic units to create a Neural Network



Slides from Awni Hannun (CS221 Autumn 2013)

Notation

$l = 1, ..., L$ - l-th layer

$W^{(l)} \in \mathbb{R}^{mxn}$ - weights for layer l

$b^{(l)} \in \mathbb{R}^{m}$ - bias for layer l

$\sigma(z)$ - activation function (for the folllowing σ is the sigmoid function)

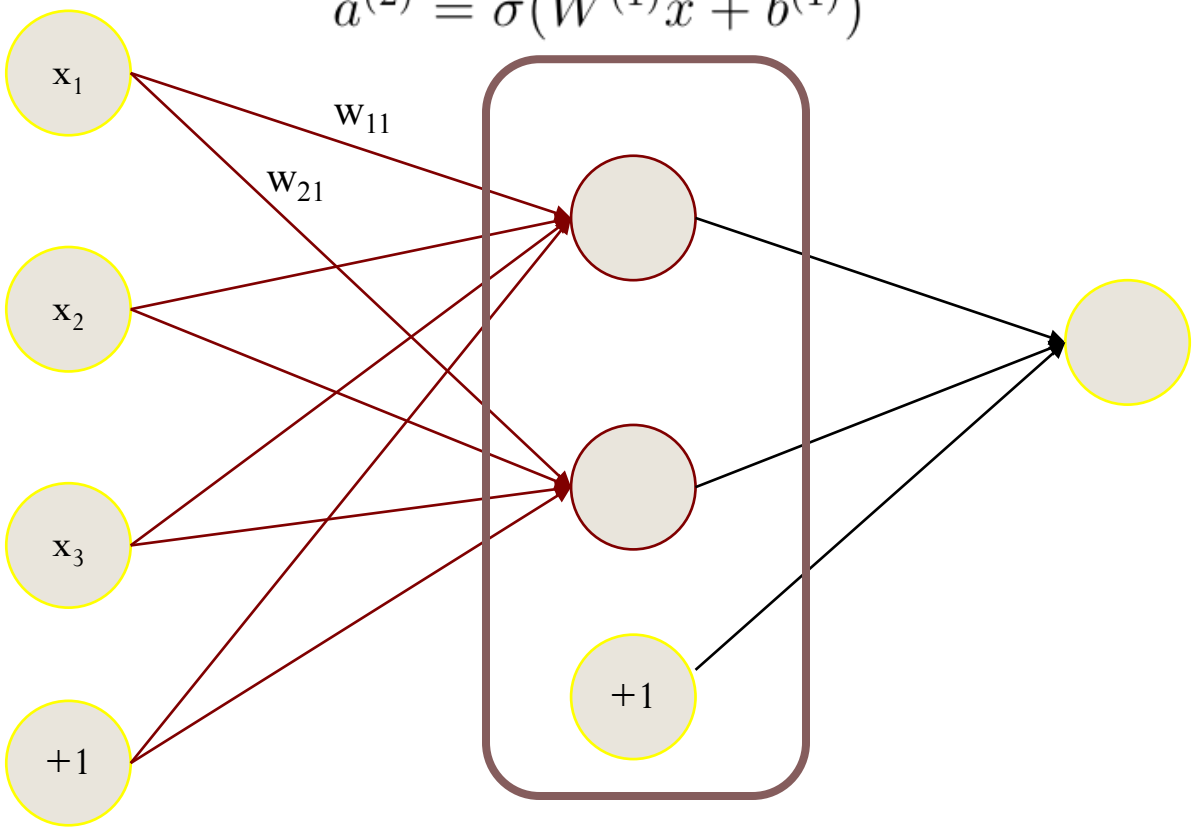
$z^{(l+1)} = W^{(l)}a^{(l)} + b^{(l)}$ - input to $(l+1)$-st layer

$a^{(l+1)} = \sigma(z^{(l+1)})$ - activation of $(l+1)$-st layer, let $a^{(1)} = x$

Slides from Awni Hannun (CS221 Autumn 2013)

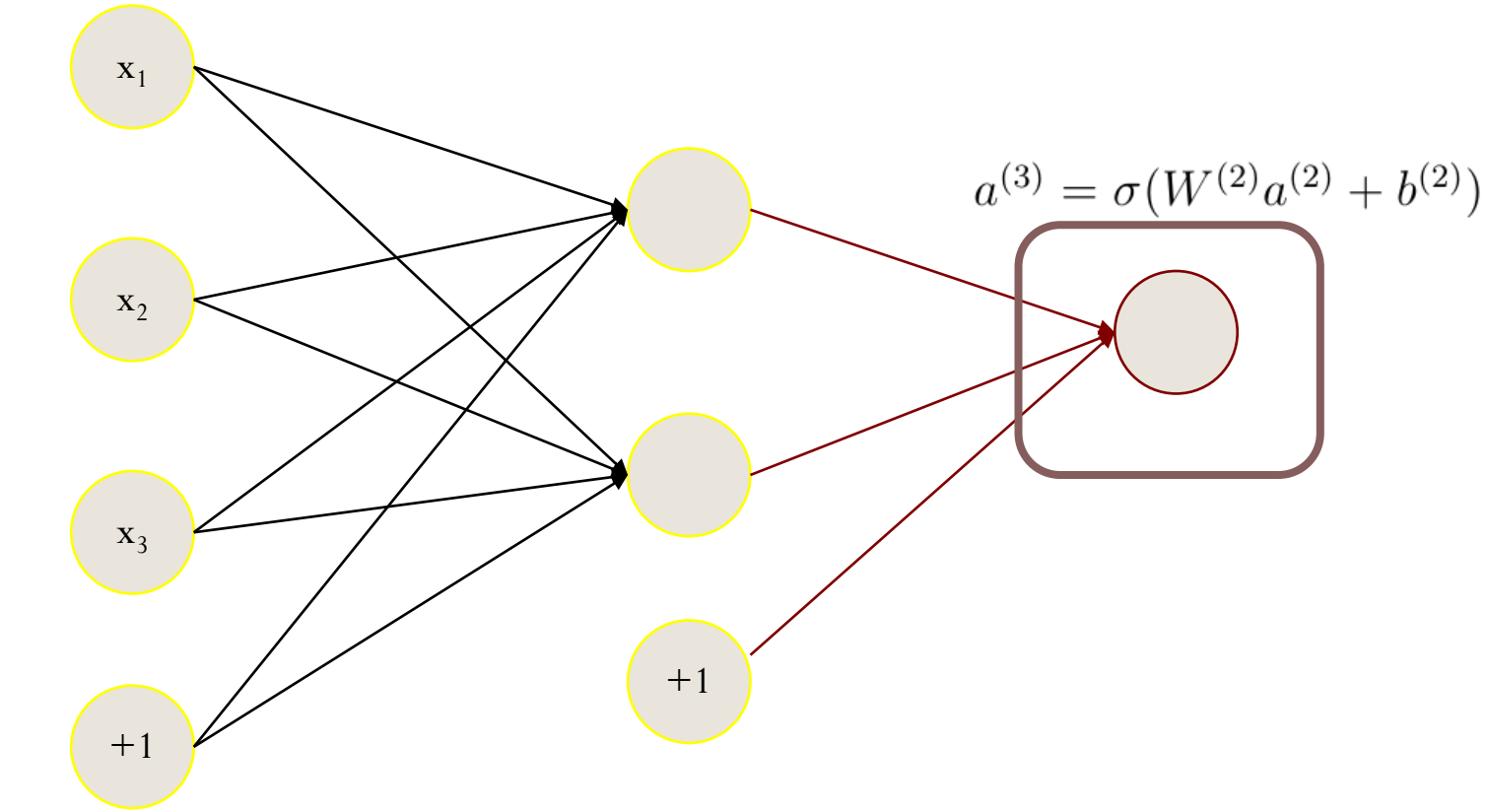
Forward Propagation

$$a^{(2)} = \sigma(W^{(1)}x + b^{(1)})$$



Slides from Awni Hannun (CS221 Autumn 2013)

Forward Propagation

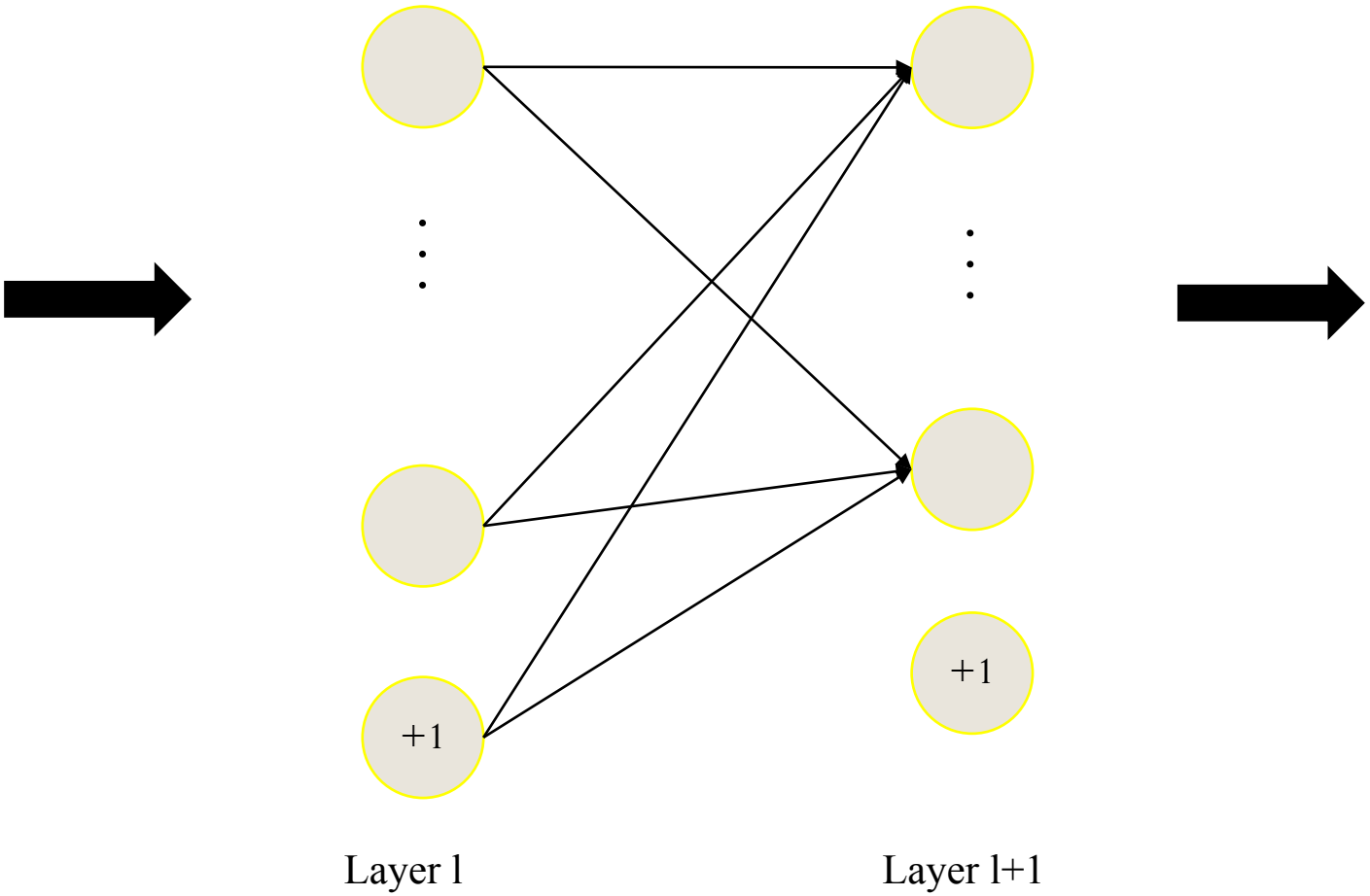


Slides from Awni Hannun (CS221 Autumn 2013)

Forward Propagation with Many Hidden Layers

$$z^{(l+1)} = W^{(l)} a^{(l)} + b^{(l)}$$

$$a^{(l+1)} = \sigma(z^{(l+1)})$$



Slides from Awni Hannun (CS221 Autumn 2013)

Forward Propagation as a Single Function

$$f(x) = \sigma\left(W^{(2)}\sigma\left(W^{(1)}x + b^{(1)}\right) + b^{(2)}\right)$$ ction of the input

- But what about multi-class outputs?
  - Replace output unit for your needs
  - "Softmax" output unit instead of sigmoid

$$p(y = k; z) = \frac{\exp(w_k^T z)}{\sum_{j=1}^{K} \exp(w_j^T z)}$$

Slides from Awni Hannun (CS221 Autumn 2013)

Objective Function for Learning

- Supervised learning, minimize our classification errors
- Standard choice: Cross entropy loss function
  - Straightforward extension of logistic loss for binary

$$Loss(x, y; W, b) = -\sum_{k=1}^{K} (y = k) \log f(x)_k$$

- This is a *frame-wise* loss. We use a label for each frame from a forced alignment
- Other loss functions possible. Can get deeper integration with the HMM or word error rate

The Learning Problem

- Find the optimal network weights

$$argmax_{W,b} \sum_{(x,y)\in D} Loss(x, y; W, b)$$

- How do we do this in practice?
  - Non-convex
  - Gradient-based optimization
  - Simplest is stochastic gradient descent (SGD)
  - Many choices exist. Area of active research

Computing Gradients: Backpropagation

Backpropagation

Algorithm to compute the derivative of the loss function with respect to the parameters of the network

$$\nabla_W \text{Loss}(x, y; W, b)$$

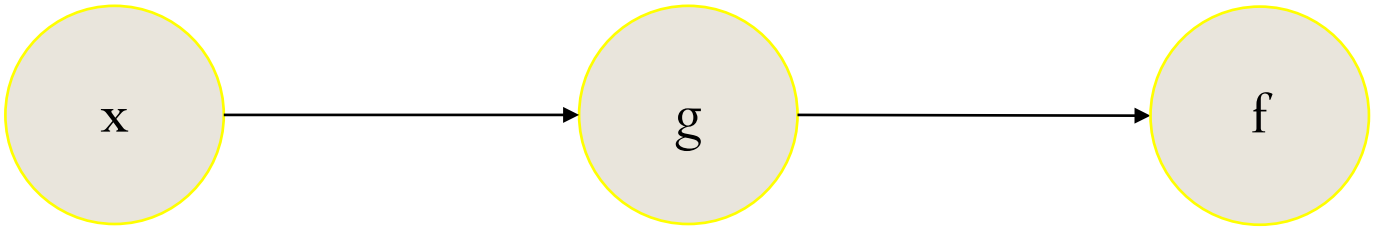
$$\nabla_b \text{Loss}(x, y; W, b)$$

Slides from Awni Hannun (CS221 Autumn 2013)

Chain Rule

Recall our NN as a single function: $f(x) = \sigma\left(W^{(2)}\sigma\left(W^{(1)}x + b^{(1)}\right) + b^{(2)}\right)$

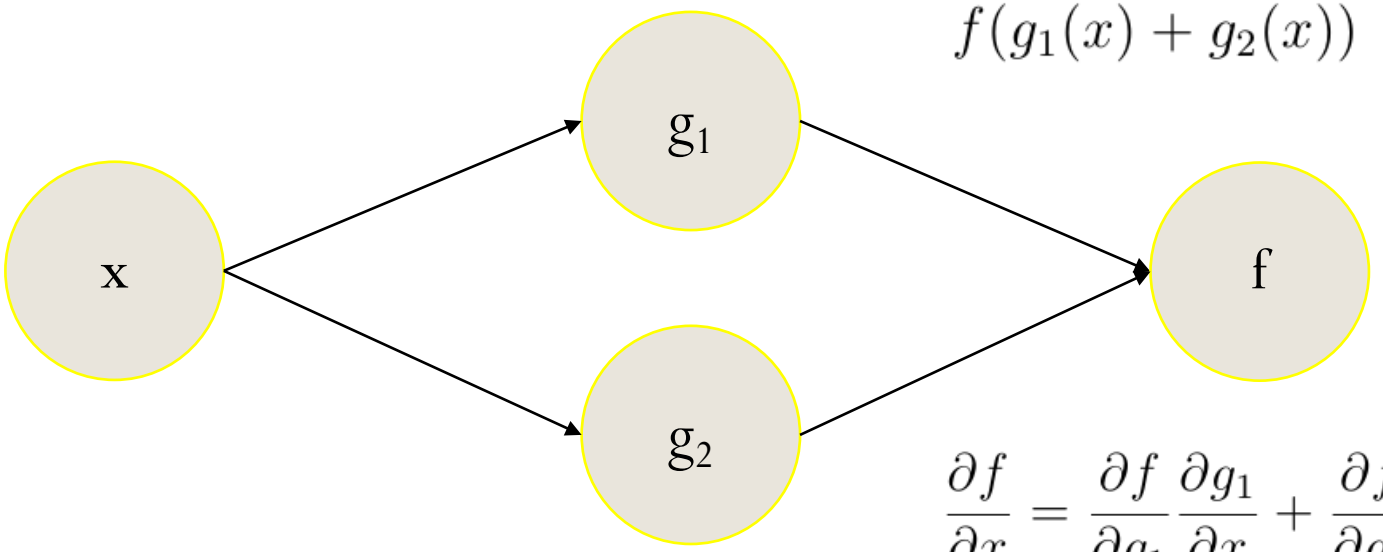
$$(f \circ g)(x) = f(g(x))$$



$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial g}\frac{\partial g}{\partial x}$$

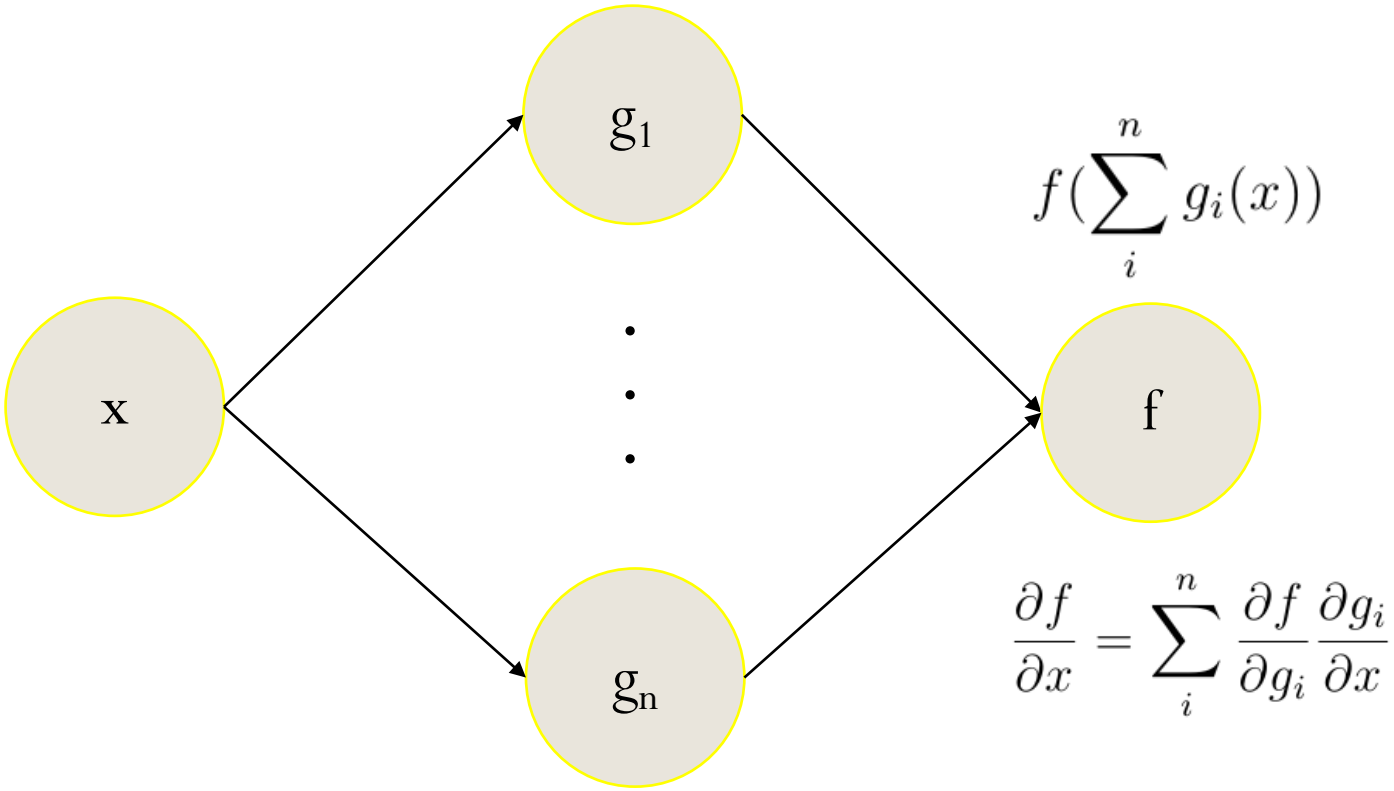
Slides from Awni Hannun (CS221 Autumn 2013)

Chain Rule



Slides from Awni Hannun (CS221 Autumn 2013)

Chain Rule

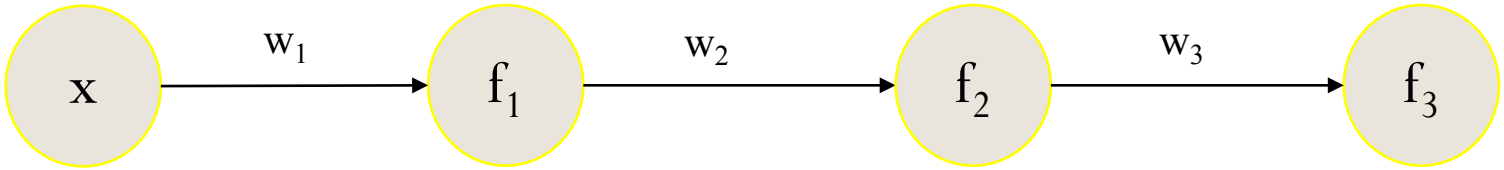


Slides from Awni Hannun (CS221 Autumn 2013)

Backpropagation

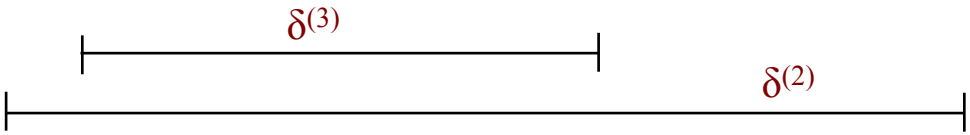
Idea: apply chain rule recursively

$$f(x) = f_3(w_3 f_2(w_2 f_1(w_1 x)))$$



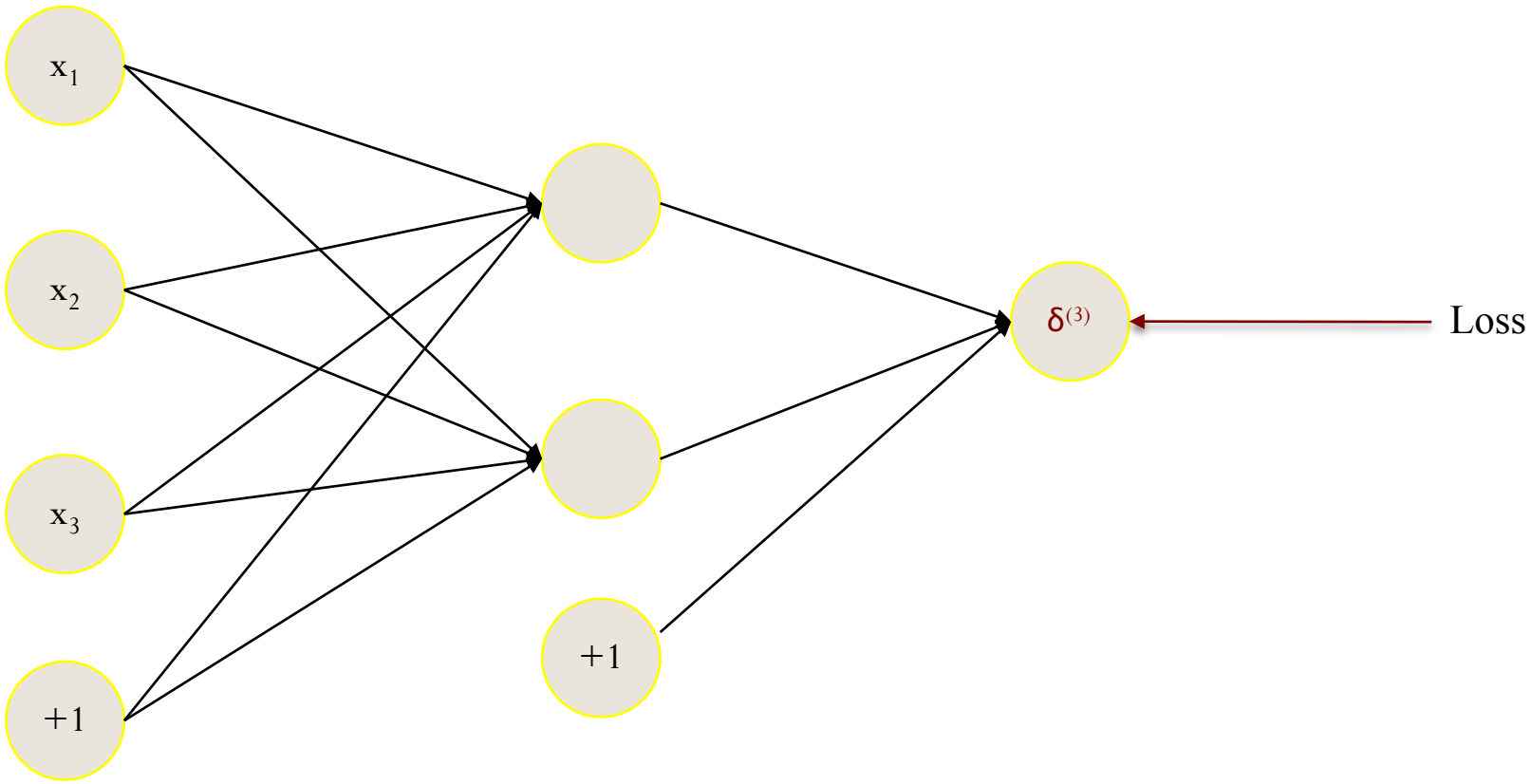
$$\frac{df}{dx} = f_3'(w_3 f_2(w_2 f_1(w_1 x)))\frac{d}{dx}(w_3 f_2(w_2 f_1(w_1 x))$$

$$\frac{df}{dx} = w_3 f_3'(w_3 f_2(w_2 f_1(w_1 x)))f_2'(w_2 f_1(w_1 x))\frac{d}{dx}(w_2 f_1(w_1 x))$$



Slides from Awni Hannun (CS221 Autumn 2013)

Backpropagation



Slides from Awni Hannun (CS221 Autumn 2013)

Neural network with regression loss

Minimize

$$\sum_i ||\hat{\mathbf{y}} - \mathbf{y}||^2$$

Output Layer

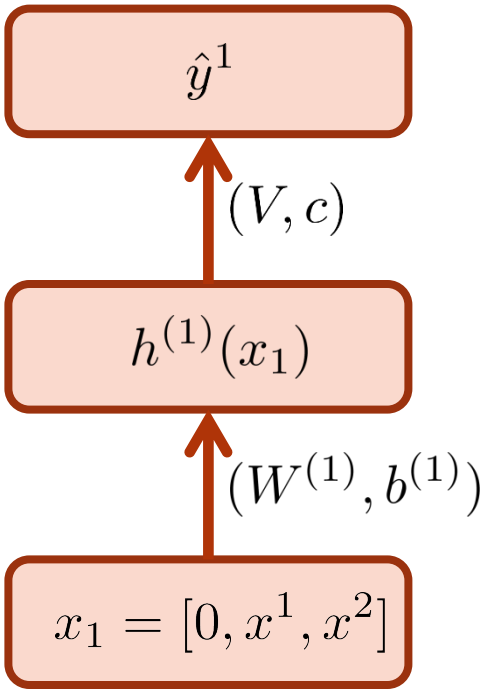
$$\hat{y}^1 = V h^{(1)}(x_1) + c$$

Hidden Layer

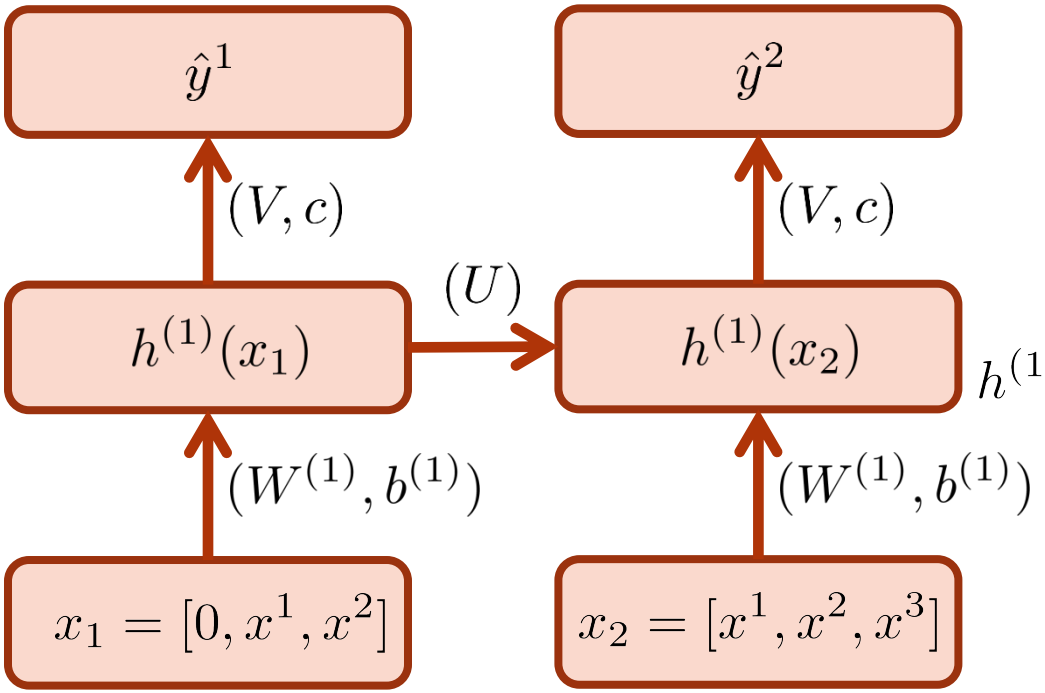
$$h^{(1)}(x_1) = \sigma(W^{(1)} x_1 + b^{(1)})$$

Noisy Input

$$\mathbf{X}$$



Recurrent Network



Output Layer

$$\hat{y}^2 = Vh^{(1)}(x_2) + c$$

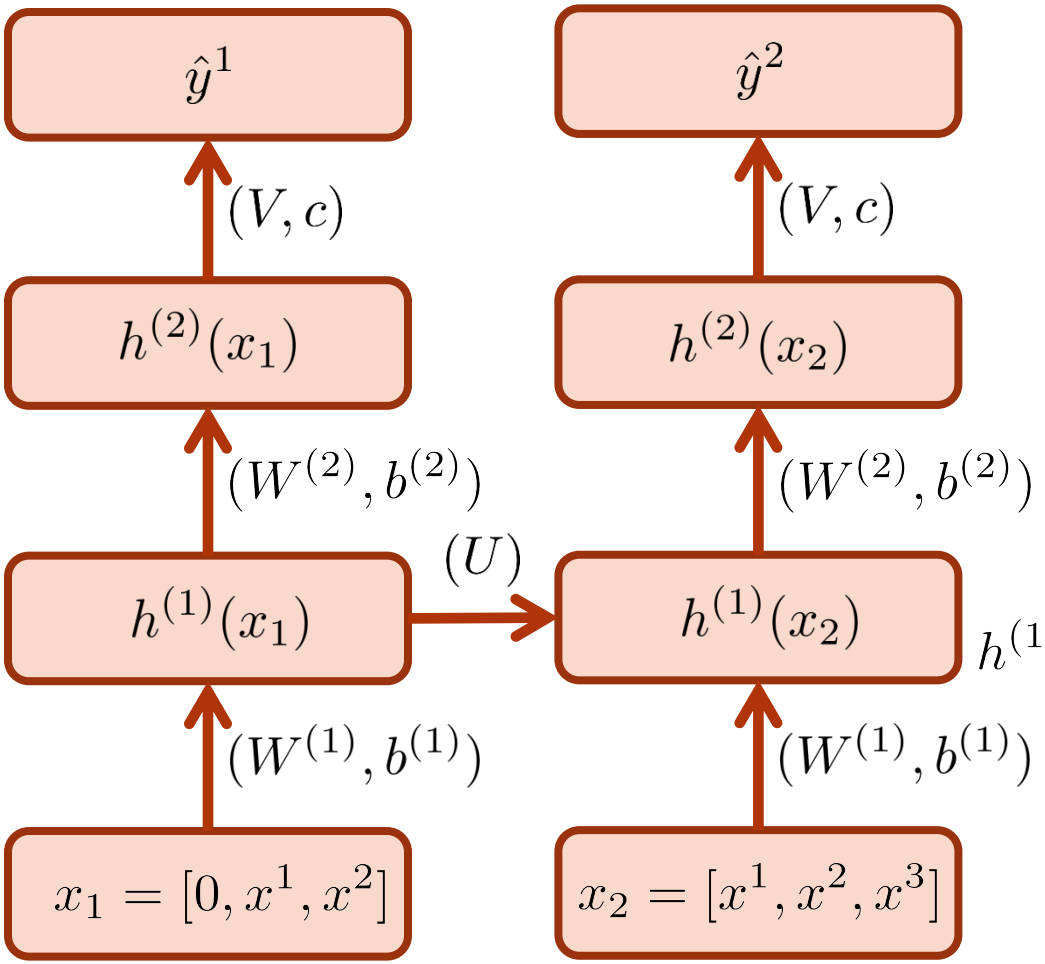
Hidden Layer

$$h^{(1)}(x_2) = \sigma(W^{(1)}x_2 + b^{(1)} + Uh^{(1)}(x_1))$$

Noisy Input

$\mathbf{X}$

Deep Recurrent Network



Output Layer

$$\hat{y}^2 = Vh^{(1)}(x_2) + c$$

Hidden Layer

$$h^{(2)}(x_2) = \sigma(W^{(2)}h^{(1)}(x_2) + b^{(2)})$$

Hidden Layer

$$h^{(1)}(x_2) = \sigma(W^{(1)}x_2 + b^{(1)} + Uh^{(1)}(x_1))$$

Noisy Input

$\mathbf{X}$

Compute graphs