# CPSC 440: Machine Learning

Recurrent Neural Networks

Winter 2022

# Last Time: What are we Learning?

- Modern ML is amazing:
  - Unprecedented performance on difficult problems.
  - Good enough to be used in many products.
  - Deep models seem to learn increasingly-complicated features.
- Modern ML is awful:
  - Easily-fooled by out-of-distribution or adversarial examples.
  - Confuses correlation and causation.
  - Can propagate and even enhance harmful biases.
  - Does not work well for some problems (social prediction).
- For some applications current ML methods should not be used.

# Some Issues with Algorithms for Social Prediction

- Does fighting over-fitting give bad predictions on sub-groups?
  - If you have 99% "Group A" in your dataset,
    model can do well on average by only focusing on Group A.
    - Treat the other 1% as outliers.
  - Does "not trying to overfit" mean we perform badly on some groups?
  - Can we discover what groups exist in our dataset?

- What if all institutions use the same algorithm?
  - You apply for jobs everywhere, and are always rejected by the algorithm?
    - Even though you may be arbitrarily-close to the decision threshold.

- Fixing the various societal problems with using ML algorithms:
  - Hot research topic at the moment (good thesis or course project topic).
  - We do not currently have nice "solutions" for these issues.
    - Try to think of potential confounding factors, and consider whether ML is not appropriate.

# Energy Costs

- Current methods require:
  - A lot of data.
  - A lot of time to train.
  - Many training runs to do hyper-parameter optimization.

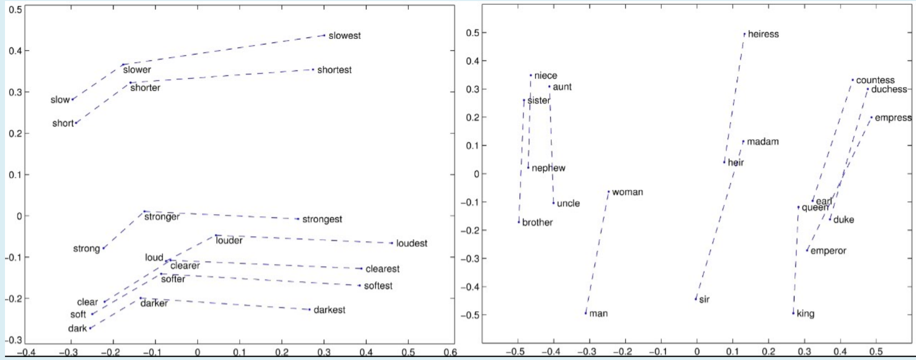
- Recent paper regarding recent deep language models:
  - Entire training procedure emits 5 times more $CO_2$ than lifetime emission of a car, including making the car.

# Next Topic: Recurrent Neural Networks

# Review: Word Representations

- How do we represent words with features?

- Lexical features:
  - Represent words using a "1 of k" encoding.
    - Where 'k' is the number of words in training data.
      - Or "words that appear at least 5 times in the training data".
      - Set all these features to 0 for other words.

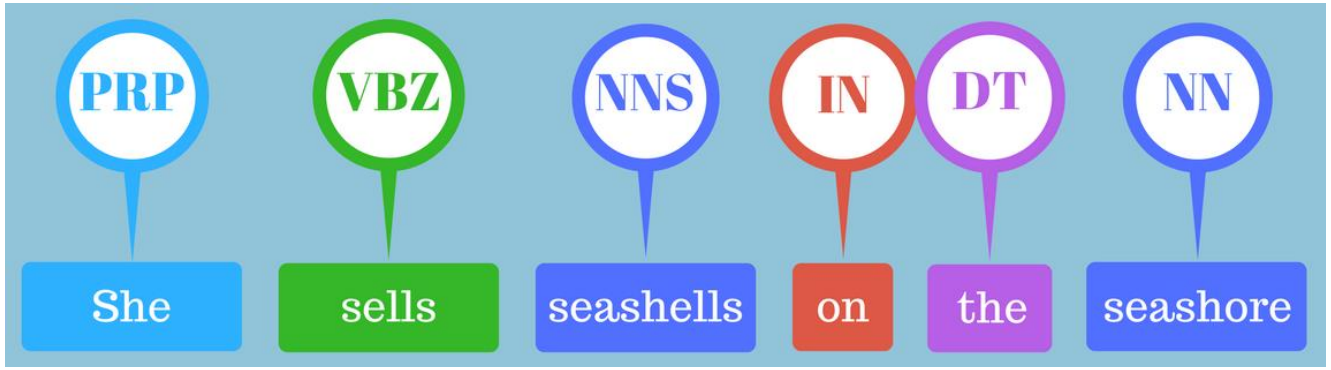



- Latent-factor models like word2vec or GloVe:
  - Unsupervised learning of a set of continuous features for each word.
    - Distances in this space may approximate semantic meaning.
    - May do sensible things for words not seen during training.

http://sebastianruder.com/secret-word2vec

# Motivation: Part of Speech (POS) Tagging

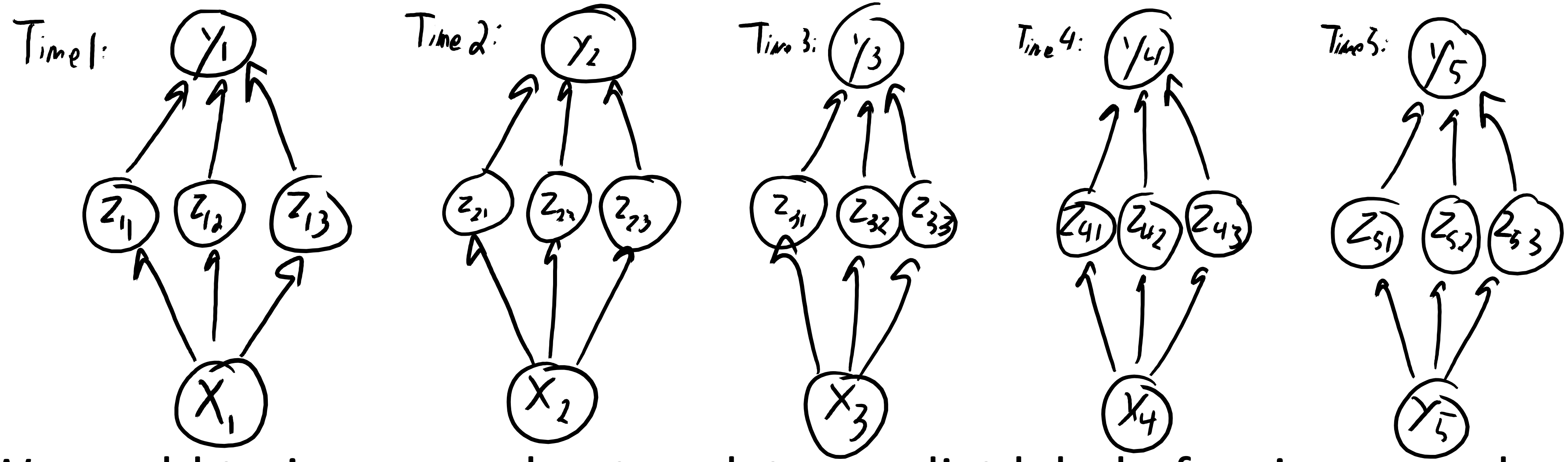
- Consider predicting part of speech for each word in a sentence:




- Input is a sequence of words.
  - Could be represented as “1 of k” or using continuous vectors like word2vec.
- Output is a categorical label for each word.
  - In English there are more than 40 categories.
    - And there are some dependencies in labels (like “only 1 verb in the sentence”).
- General problem: sequence labeling.
  - Biological sequences, various language tasks, sound processing.

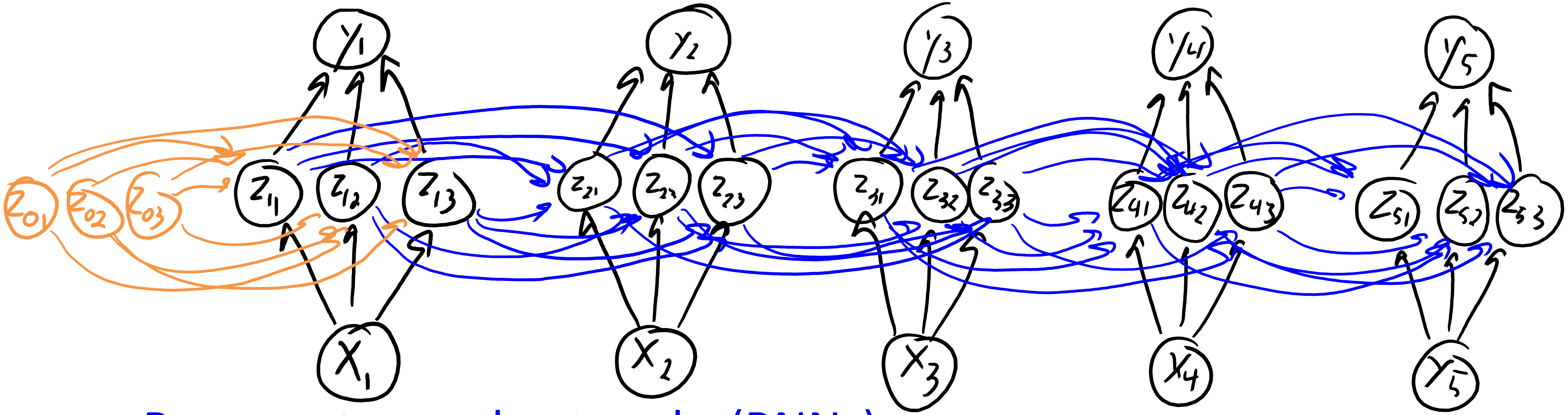
https://medium.com/analytics-vidhya/pos-tagging-using-conditional-random-fields-92077e5eaa31

# Individual-Word Neural Network Classifier



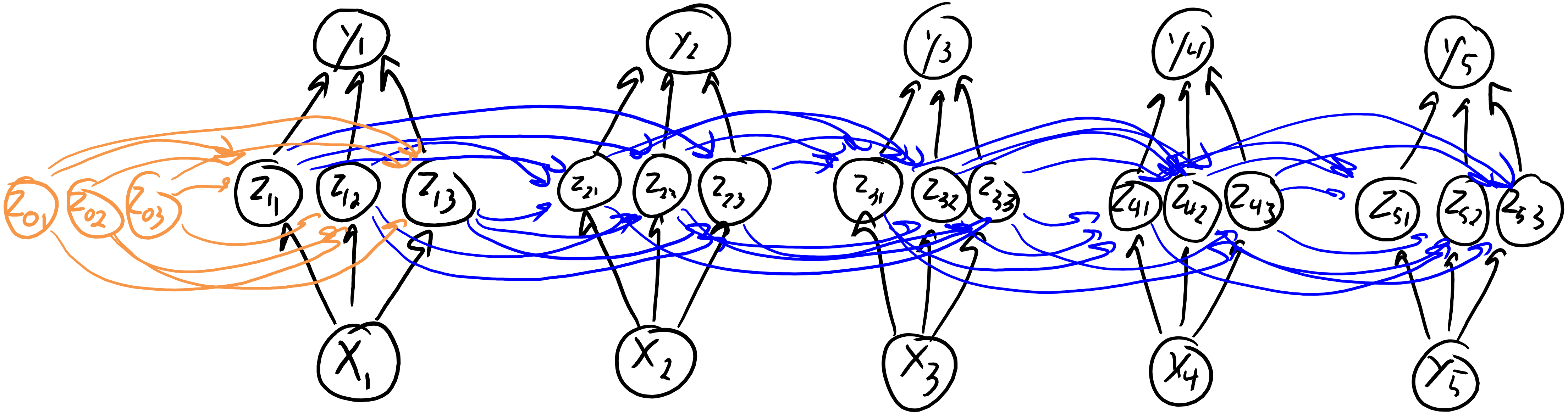

- We could train a neural network to predict label of a given word.
  - Above we have 1 input feature for each time.
    - But each time might have multiple features (if we use something like word2vec).
  - We are also not showing the non-linear transform or bias variables.
- But this type of model would not capture dependencies.
  - Information from earlier in sentence does influence prediction.
    - The word “desert” could be a noun or a verb depending on context.

# Recurrent Neural Network for Sequence Labeling




- Recurrent neural networks (RNNs):
  - Add connections between adjacent different times to model dependencies.
  - Add an initial hidden state.
  - Use the same parameters across time.
- Repeating parameters in different places is called parameter tieing.
  - We previously saw convolutions, which use parameter tieing across space.
  - By tieing parameters across time, RNNs can label sequences of different lengths.

# Recurrent Neural Network for Sequence Labeling




$$\hat{y}_t = \underbrace{V}_{\text{We have a matrix 'V' because we are doing multi-class}} h(z_t)$$

$$z_t = W x_t + \underbrace{U}_{\text{weights on temporal connections}} \underbrace{h(z_{t-1})}_{\text{hidden units at previous time}}$$

Parameters: $W$, $V$, $U$ (and possibly $z_0$)

Use $\hat{y}_t$ vector in softmax at each time

(Notice that we use the same "tied" matrices $\{W, V, U\}$ for all times '$t$'.)

# Recurrent Neural Network Inference

$$\underbrace{\hat{y}_t}_{k\times 1} = \underbrace{V}_{k\times m} h(\underbrace{z_t}_{m\times 1}) \qquad \underbrace{z_t}_{m\times 1} = \underbrace{W}_{m\times d}\underbrace{x_t}_{d\times 1} + \underbrace{U}_{m\times m} h(\underbrace{z_{t-1}}_{m\times 1})$$

- Assume we have:
  - ‘k’ different classes that each $\hat{y}_t$ can take.
  - ‘m’ hidden units at each time.
  - ‘T’ times (length of sequene).
- Cost to compute all $\hat{y}_t$ if each time has ‘m’ units and we have ‘T’ times:
  - We need to do an O(md) operations ‘T’ times to compute $Wx_t$ for all ‘t’.
  - We need to do an O(km) operation ‘T’ times to compute $\hat{y}_t$ for all ‘t’.
  - We need to do a $O(m^2)$ operation ‘T’ times to compute each $z_t$.
  - Total cost: $O(tmd + tkm + tm^2)$.
- For the likelihood, we could use an independent softmax for each time.
  - $p(y_{1:T} \mid x_{1:T}, W, V, U) = p(y_1 \mid x_1, W, V, U)p(y_2 \mid x_{1:2}, W, V, U)\cdots p(y_T \mid x_{1:T}, W, V, U)$.
    - Where each p(yt | x1:T, W, V, U) is given by softmax over $\hat{y}_t$ values.
    - Conditioned on features and parameters, this assumes a “product of categoricals” model.

# RNN Learning

- The objective function we use to train RNNs is the NLL:

$$f(W,V,U) = -\sum_{i=1}^{n}\sum_{t=1}^{T} \log p(y_t^i \mid x_{1:T}^i, W, V, U)$$

  - In the above I assume all sequences have the same length 'T'.
    - But in practice you will often have sequences of different lengths.
- Computing gradient called "backpropagation through time" (BTT).
  - Equations are the same as usual backpropagation/chain-rule.
    - If you do it by hand, make sure to add all terms for tied parameters.
  - Automatic differentiation is commonly used.
- Usually trained with SGD.
  - Sample an example 'i' on each iteration, do BTT, update all parameters.
  - which has usual challenges.

# RNN Learning – Extra Challenges

- Unfortunately, training RNNs presents some extra challenges:
  - Computing gradient requires a lot of memory for long sequences.
    - There are a lot intermediate calculations.
    - Make sure AD package handles matrix multiplication.
  - Parameter tieing often leads to vanishing/exploding gradient problems.
    - Consider a linear RNN and just consider the temporal 'U' updates:
      - $z^L = U*U*U*\cdots*U*z_0 = U^L z_0$.
      - For typical $z_0$, the quantity $z_L$ either diverges exponentially or converges to zero exponentially.
        - » If largest singular value of 'U' is > 1, $||z_L||$ increases exponentially with 'L'.
        - » If largest singular value of 'U' is < 1, $||z_L||$ converges to zero exponentially with 'L'.
  - Usual SGD methods tend not to work well.
    - Often need to use optimizers like Adam or use gradient clipping:
      - If norm of gradient is larger than some threshold, "shrink" norm to threshold:

$$\text{if } \|g\| > u$$

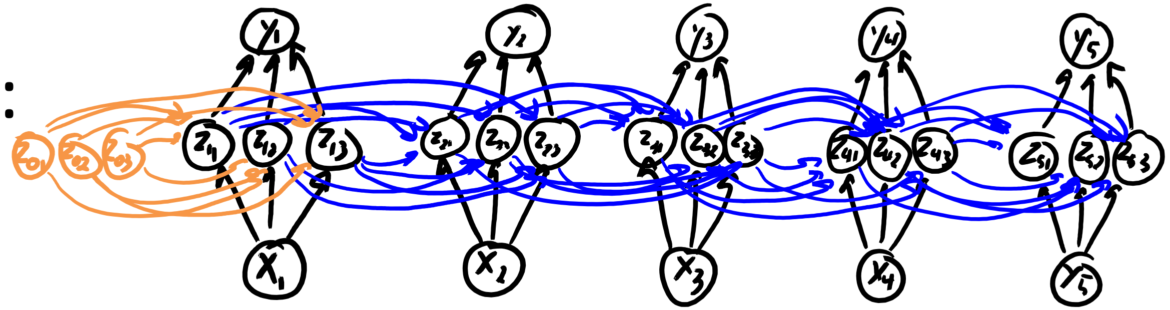
$$g \leftarrow \frac{gu}{\|g\|}$$

    - People are trying to explore initialization/keeping 'U' orthogonal.
      - So that all singular values are 1 (some positive and negative results on this).

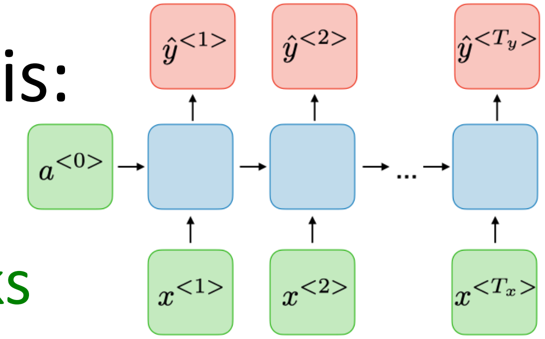
https://towardsdatascience.com/neural-network-optimization-7ca72d4db3e0

# Deep RNNs

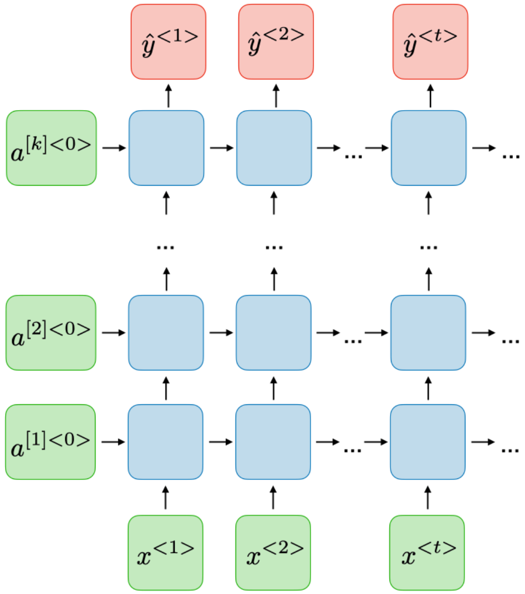
- Instead of drawing this:



- We often use diagrams like this:
  – Up to some notation changes.
  – We connect everything in blocks connected by arrows.



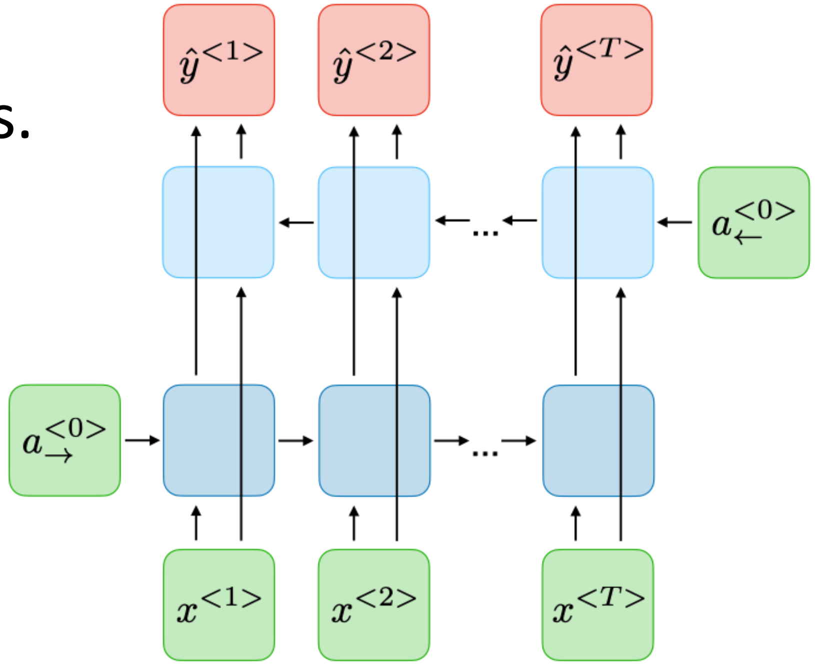

- Deep RNNs add multiple hidden layers at each time:




https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks

# Bi-Directional RNNs

- Sometimes later information later changes meaning:
  - "I've had a perfectly wonderful evening, but this wasn't it."
    - "Paraprosdokian".
- Bi-directional RNNs have hidden layers running in both directions:
  - Use different parameters for the forward and backward directions.



# Next Topic: Sequence to Sequence RNNs

# Motivating Problem: Machine Translation

- Consider the problem of machine translation:
  - Input is text from one language.
  - Output is text from another language with the same meaning.

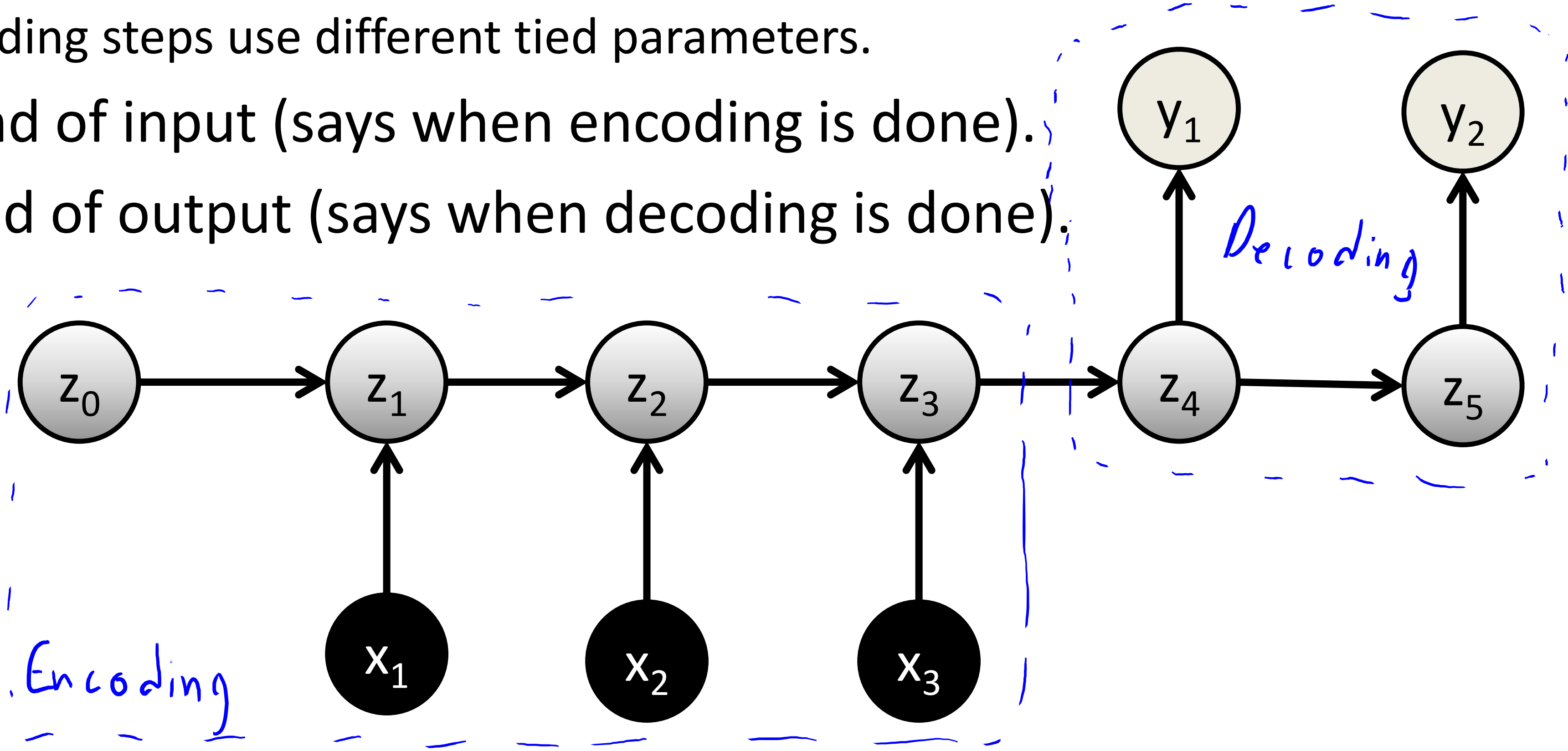
This course is intended as a second or third university-level course on machine learning, a field that focuses on using automated data analysis for tasks like pattern recognition and prediction.

Ce cours est conçu comme un cours de deuxième ou troisième niveau universitaire sur l'apprentissage automatique, un domaine qui se concentre sur l'utilisation de l'analyse de données automatisée pour des tâches telles que la reconnaissance de formes et la prédiction.

- A key difference with pixel labeling:
  - Input and output sequences may have different lengths.
    - We do not just "find the French word corresponding to the English word".
  - We may not know the output length.

# Sequence-to-Sequence RNNs

- Sequence-to-sequence RNNs encode and decode sequences:
  - Each encoding step has one word as input and no output.
  - Each decoding step outputs one word and has no input.
    - Encoding and decoding steps use different tied parameters.
  - Special "BOS" at end of input (says when encoding is done).
  - Speical "EOS" at end of output (says when decoding is done).



# Summary

- Recurrent neural networks (RNNs):
  - Neural networks for sequence prediction.
  - Have connections between hidden units at adjacent times.
  - Use parameter tieing across time.
    - Allows sequences of different lengths.
    - Leads to vanishing and exploding gradients.
- Sequence-to-Sequence RNNs:
  - Encoding phase takes in one input at a time until we reach “BOS”.
  - Decoding phase outputs one output at a time until we output “EOS”.
  - Allows input and output sequences whose lengths differ.

- Next time: generating poetry.