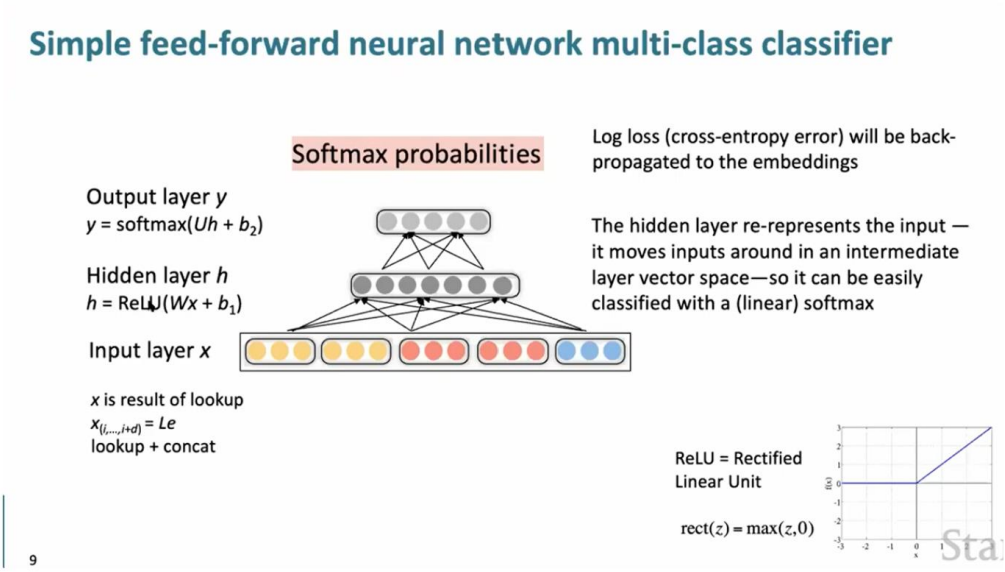
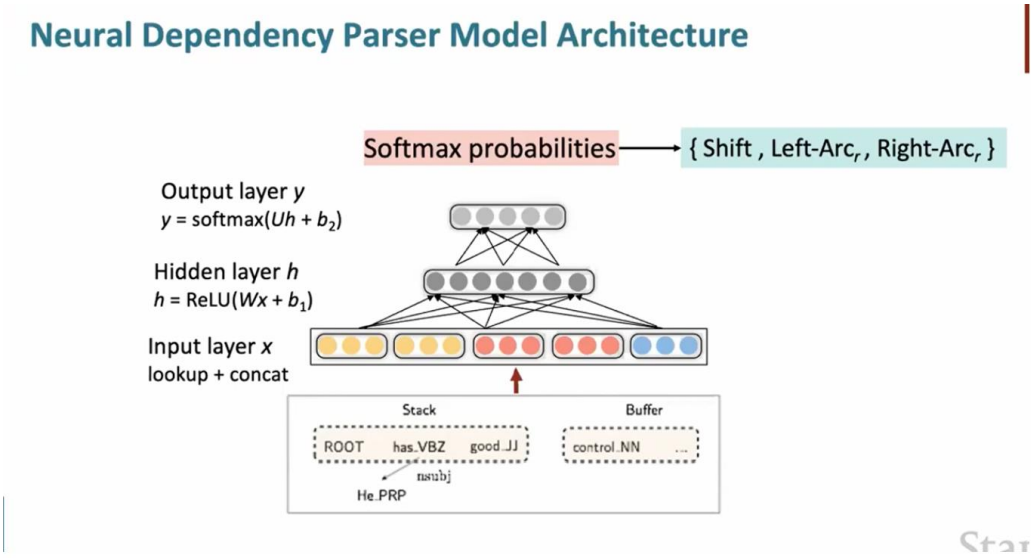
1. Neural dependency parsing

   Problem 1. sparse 2. incomplete 3. expensive computation





softmax layer

-hidden layer를 거친 feature vecto를 linear projection 후 softmax function 적용

-shift, left arc, right arc 중 가장 확률값이 높은 경우의 수를 output로 산출

unlabeled attachment score(UAS): Arc 방향 예측

labeled attachment score(LAS): Arc 방향, label 예측

Dependency Parsing Method

1) Ablation Studies

- Cube function이 다른 activation function보다 높은 성능을 보임
- pre-trained word vector를 사용하는 것이 random initialization보다 더 높은 성능을 보임
- word, POS, label를 모두 활용하는 것이 가장 높은 성능 보임

2) POS and Label Embedding

- 학습이 진행되면서 Random initialization된 POS tag와 arc-label vector가 의미적 유사성 내포
- t-SNE를 통해 2차원 공간상에 표현했을 때 유사한 요소들이 서로 가까이 위치함

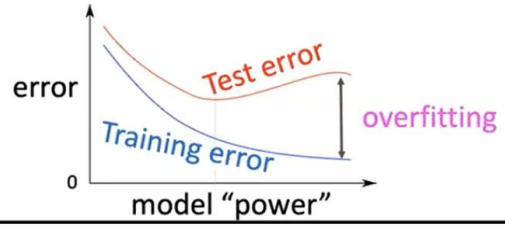
2. A bit more about neural network

We have models with many parameters! Regularization!

- A full loss function includes regularization over all parameters θ, e.g., L2 regularization:

$$J(\theta) = \frac{1}{N}\sum_{i=1}^{N} -\log\left(\frac{e^{f_{y_i}}}{\sum_{c=1}^{C} e^{f_c}}\right) + \lambda\sum_{k}\theta_k^2$$

- Classic view: Regularization works to prevent overfitting when we have a lot of features (or later a very powerful/deep model, etc.)



17

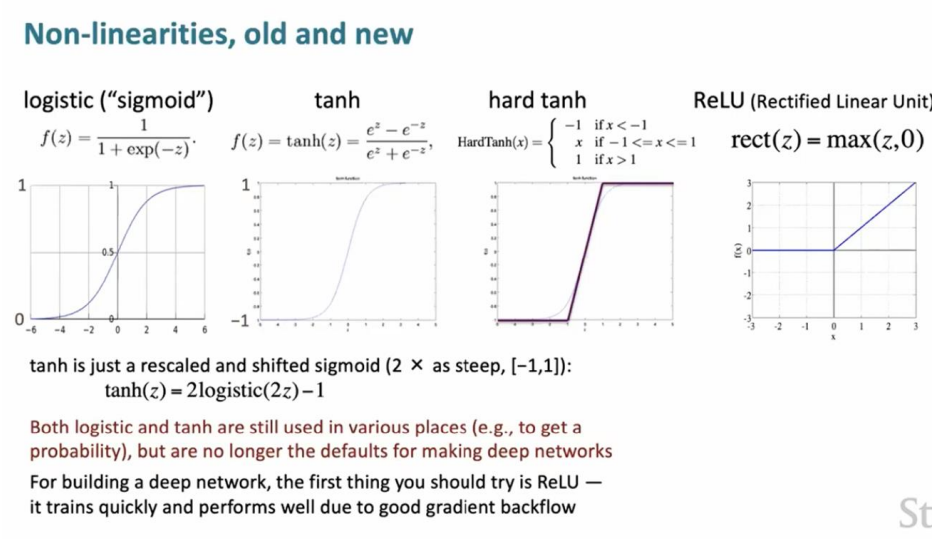
"Vectorization"

- E.g., looping over word vectors versus concatenating them all into one large matrix and then multiplying the softmax weights with that matrix:

```
from numpy import random
N = 500 # number of windows to classify
d = 300 # dimensionality of each window
C = 5 # number of classes
W = random.rand(C,d)
wordvectors_list = [random.rand(d,1) for i in range(N)]
wordvectors_one_matrix = random.rand(d,N)

%timeit [W.dot(wordvectors_list[i]) for i in range(N)]
%timeit W.dot(wordvectors_one_matrix)
```

- 1000 loops, best of 3: **639 µs** per loop
  10000 loops, best of 3: **53.8 µs** per loop ← Now using a single a C x N matrix
- Matrices are awesome!!! Always try to use vectors and matrices rather than for loops!
- The speed gain goes from 1 to 2 orders of magnitude with GPUs!



Parameter Initialization

-가중치가 0인 경우 hidden layer biases를 0으로 초기화하고 biases를 최적 값으로 출력

3. Language modeling + RNNs

-A new NLP task : Language Modeling

Language Modeling: 문장에 확률을 할당하여 다음에 나타날 단어를 예측하는 과정

ex) 기계번역, 음성인식, 자동완성과 같은 기능에 이용된다.

n-gram Language models : 이전에 등장한 n-1개의 단어들을 이용하여 다음 단어 예측

문제점

1) 희소성: 데이터에 단어가 등장하지 않을 경우 확률 계산이 불가

2) 저장공간: corpus가 커질수록 모델의 크기도 증가

3)일관성: 문맥의 충분한 반영 부족

-A new family of neural networks: Recurrent Neural Networks(RNNs)

기존 신경망 구조보다 입력값이 독립적임. 이전의 계산 결과에 영향을 받는다.

장점

1. 입력 제한이 없음
2. 길이가 긴 timestep t에 대해 처리 가능
3. 입력에 따라 모델의 크기 변화 없음
4. 매 timestep t에 동일한 가중치 적용하여 대칭.

단점

1. 다음단계의 진행을 위해 이전 단계의 계산이 완료되어야 해서 계산이 병렬적으로 진행되지 않음.