**CS224n: NLP with Deep Learning** **Winter 2019**

# Lecture 3

*Course Coordinator: Prof. Chris Manning* *Scribes: Akash Gupta*

**Classification Intuition -**

- Training data: $\{x_i, y_i\}_{i=1}^{N}$
- Traditional ML/Stats approach: assume $x_i$ are fixed, train (i.e., set) softmax/logistic regression weights $W \epsilon \mathbb{R}^{Cxd}$ to determine a decision boundary (hyperplane).
- For each $x$, predict (C are the classes):

$$p(y|x) = \frac{exp(W_y.x)}{\sum_{c=1}^{C}(W_c.x)}$$

**Cross-entropy loss**:

$$H(p,q) = -\sum_{c=1}^{C} p(c)log(c)$$

$->$ Assume ground truth is in one hot. The only term left is the -ve log likelihood probability of the true class.

$->$ For a full dataset,

$$J(\theta) = \frac{1}{N}\sum_{i=1}^{N} -log(\frac{\exp f_{y_i}}{\sum_{c=1}^{C}\exp f_c})$$

$->$ Ground truth is not necessarily in one-hot as there can be cases when humans are not sure of the class. This refers to semi-supervised learning and cross-entropy is more commonly used there. So the ground truth will have smth like - [...., 0.5, ...0.5...]

**Neural Networks**:

- Allows to learn non-linear decision boundaries.
- Commonly in NLP, we learn both, $W$ and $x$ i.e. both conventional params and represetations.
- Word vectors re-represent one-hot vectors.
- A NN = running several logistic regressions at the same time.
- Non-linearities are needed as without them you are just doing linear transforms.

**Named Entity Recognition (NER)**:

- Find names of things from text and classify them.

- Predict entities by classifying words in context and then extracting entities as word subsequences

**Why NER is hard?** - Because (1) hard to work out boundaries of entities (2) Hard to know if smth is entity (3) Hard to know class of novel/unknown entity (4) Entity class is ambiguous and depends on context.

**Window classification**:

- Idea: classify a word in its context window of neighboring words.
- Simple way to do this is to take average of word vectors in a window. But problem is you will lose position inforamtion
- Try softmax classifier to classify a center word by taking concatenation of word vectors surrounding it in a window.

**Binary classification for NER location**: E.g. Make a classifier that gives high score if location named entity in middle and low score for otherwise.