Lecture 3

NN Classification Notation:

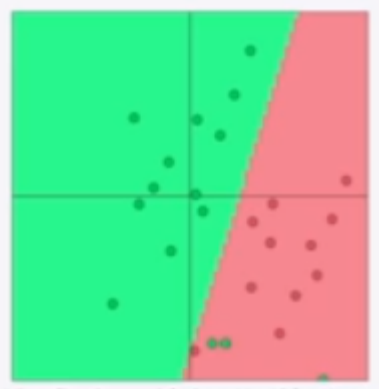
2. Classification setup and notation

- Generally we have a training dataset consisting of samples

$$\{x_i, y_i\}_{i=1}^{N}$$

- x_i are inputs, e.g. words (indices or vectors!), sentences, documents, etc.
  - Dimension d
- y_i are labels (one of C classes) we try to predict, for example:
  - classes: sentiment, named entities, buy/sell decision
  - other words
  - later: multi-word sequences

Example of a learned line:



X_i : tells its position
Y_i: tells if it is green or red

Negative log Probability with Softmax function:

In logistic regression with the above example, we have the X_i which we multiply with a learned Weights and put the results into a sigmoid function and if the probability is less than 0.5 we classify it as case 1 else case 2. Similarly we extend this to N cases with N-1 weights.

On the other hand the softmax probability function has weights for every case separately. For example, in the above red-green example with two cases, in softmax probability method we don't calculate one value and classify the instance as one of the cases depending on if the calculated number is closer to zero or one, instead we calculate two probabilities with two weights each corresponding to one of the cases and classify it as the case with a higher probability.

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

Where the given probability is the probability for the case y and C is the total number of cases.

Note: here W_y is a **row** the weight matrix w corresponding to a single case.

We take negative log of this probability to minimize because

- Negative so that we can minimize instead of maximize
- Log just to squish the probability around.

Note: **NLL** : negative log likelihood.

Cross entropy model:

Cross Entropy loss:

Let the true probability be p
Let the probability calculated by our model be q.

We use a function H called cross-Entropy function to calculate the error in calculated probabilities across all the classes for a case.

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

Notice how when both p and q are either together zero or one, H function disappears. Hence H function beautifully calculates the difference between p and q.

Notice:

- Assuming a ground truth (or true or gold or target) probability distribution that is 1 at the right class and 0 everywhere else: $p = [0,\ldots,0,1,0,\ldots 0]$ then:
- **Because of one-hot p, the only term left is the negative log probability of the true class**

Therefore:

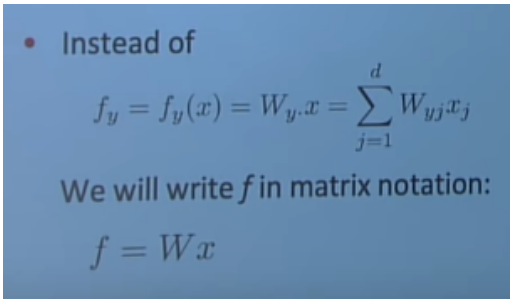
$$H(p,q) = -\log(q_i)$$

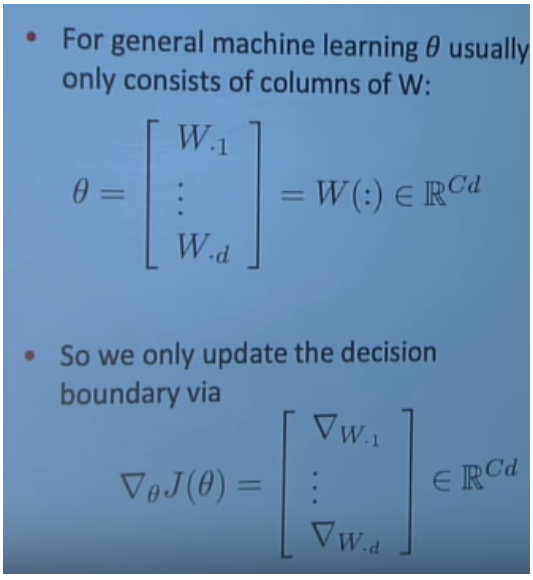
Where i is the true case / the only case where p is 1.
The above explanation is for a single point, if we have a full dataset,

$$H(p,q) = (-\Sigma \log(qi))/N$$:summed over the entire dataset,

With substitution of softmax probability q_i formula,

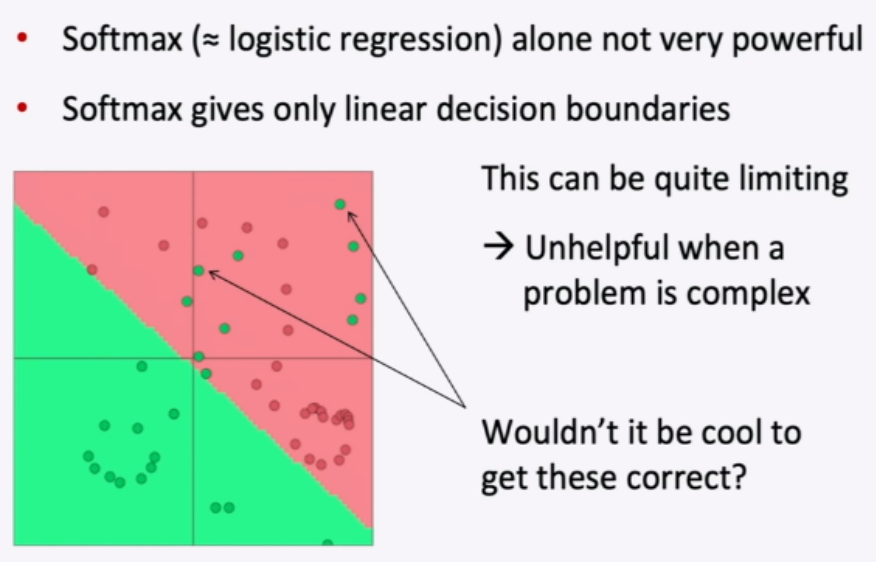
Matrix optimization:

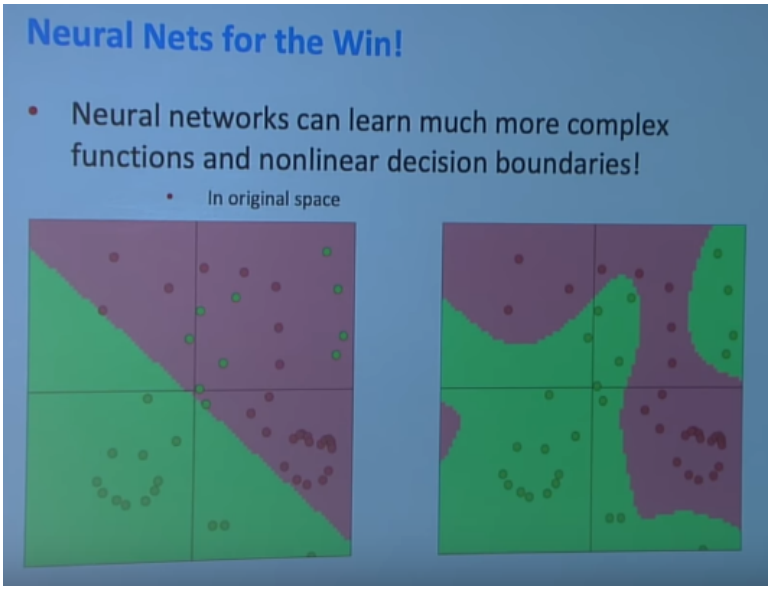




Here each weight w_i is a row vector consisting of number of elements corresponding to the number of elements in the input x.
Now do normal / stochastic gradient decs etc.

Why do we need neural network classifiers ?





Why do we need Non-Linearity functions ?

Non-Linearity functions are functions like sigmoid function that are applied to z_i to make them a_i . We call the sigmoid squishification function an activation function because in analogy it does the work an activation potential does in a neuron (The all or null effect).

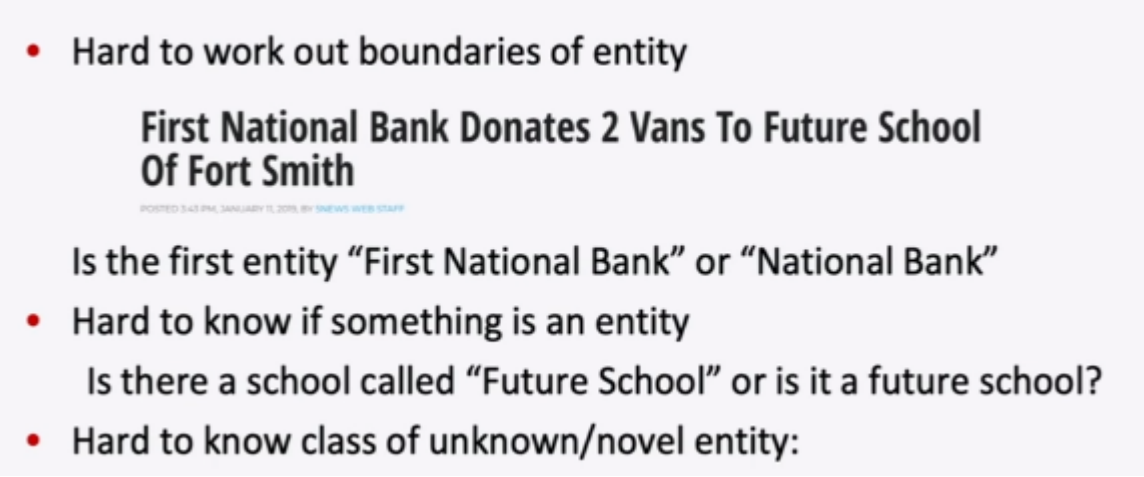
They are needed because without them these multilayer neural networks will basically still just produce linear functions as extra layers can just be compiled down to single linear function. $w_1w_2x = Wx$.

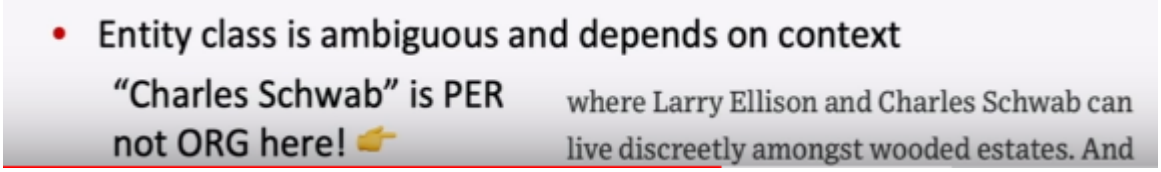
Named Entity Recognition:

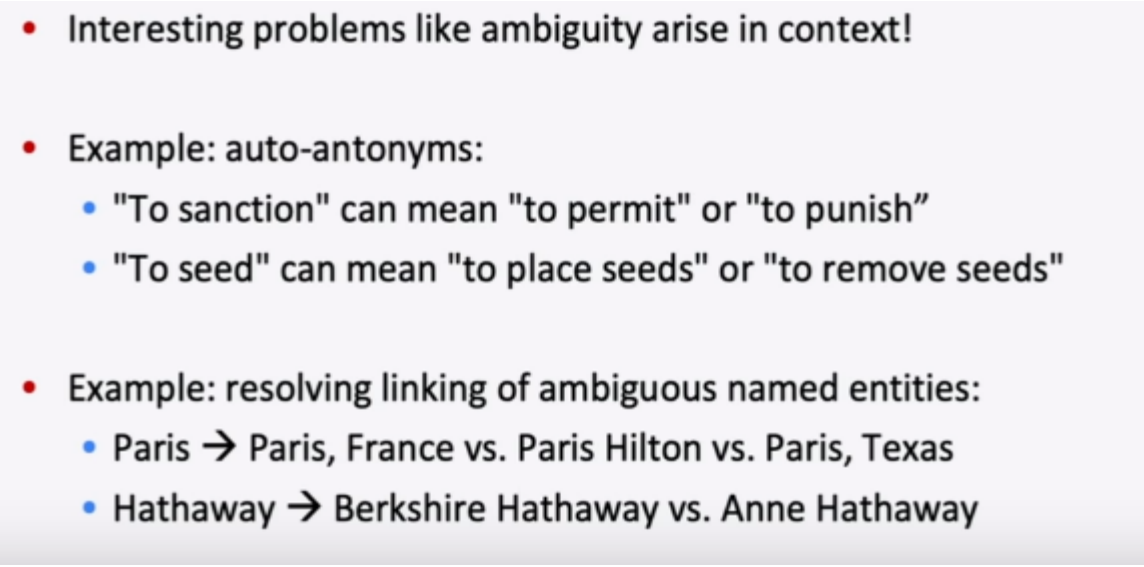
As the name suggests it is used to find named entities like name of organisations , people, countries etc.it has many uses like keeping track of how often a company is mentioned, or in answering questions.

Common way of doing it is using a classification nn.

Tricky cases:



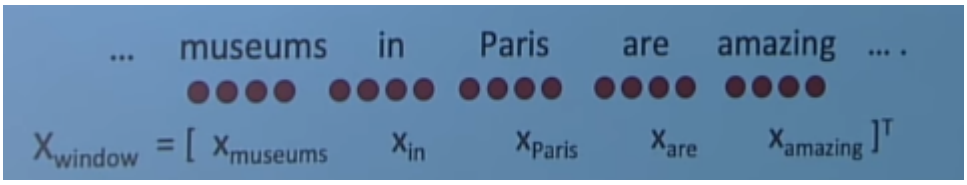




Solution:

The solution to the dependency on context for the class of a named entity is passing nearby word's word vector along with word vector in question as arguments to the neural network.

Example, if we want to classify paris in a particular context, lets just pass 2 words b4 and after it to the neural net.



This X_{window} is the input to the NN.

Important points:

1. In deep learning in NLP we not only learn the weights but we can also learn the word vectors together.