Deep Learning for NLP - Focus on Medical Applications

Multiclass, Metrics, Optimizers & Tips

In Today's Episode

- What to do when we are stuck with the Accuracy
- Multiclass classification
- Accuracy vs Precision vs Recall vs F_1
- Unbalanced datasets
- Different optimizers
    - Adam, SGD, GD
- Courses
- Papers (Twitter)

No Improvements

How long to tune the parameters

- Accuracy does not improve?
    - Don’t over do it
    - Change the number of layers
    - Change the number of neurons
    - Dropout / Regularization
    - Understand the problem
        - Check the output and find where is it making mistakes

How long to tune the parameters

- Convert continuous input variables to categorical
- You should choose the number of categories depending on your problem
- Don't do this too often and only if you are sure that it makes sense

Age - [0, 100]   e.g.   33.4

Create the categories, here I've chosen **3**

C_1 - [0:18]
C_2 - [18:65]
c_3 - [65:100]

One hot encode them:

C_1 - [1, 0, 0]
C_2 - [0, 1, 0]
C_3 - [0, 0, 1]

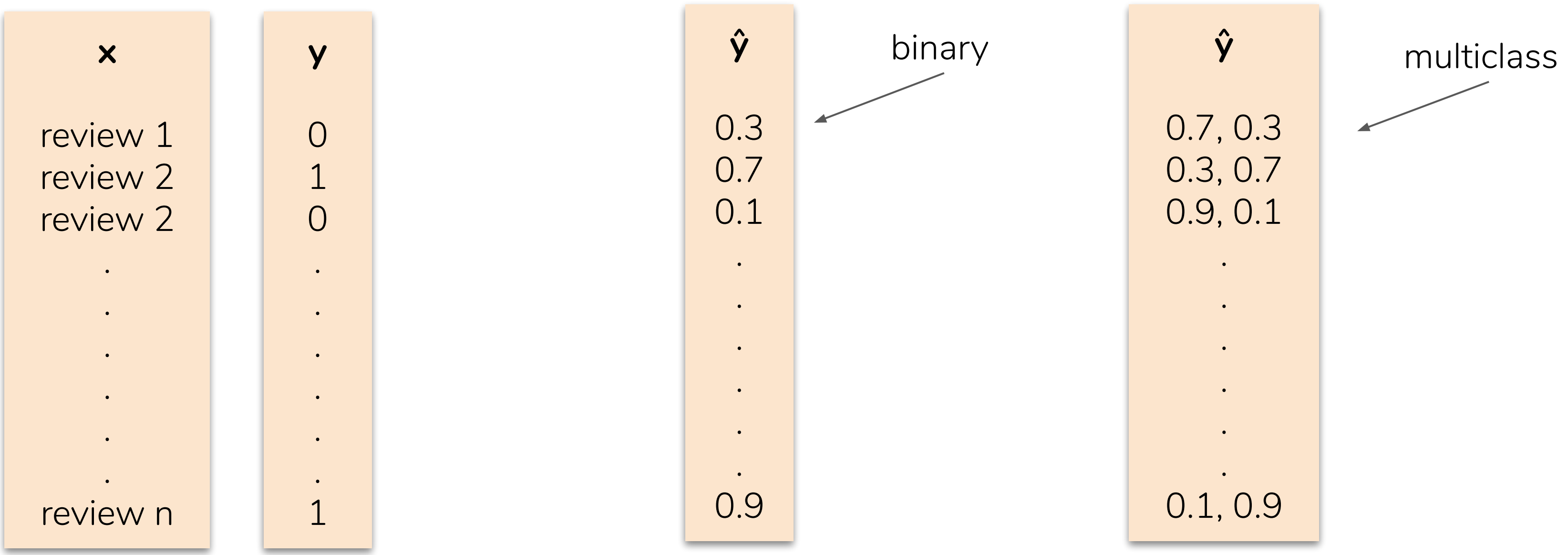
- No need to learn a continuous variable
- Better generalization for smaller datasets
- Faster convergence
- A smaller network is needed

- A variable is categorical only if it is one hot encoded. Not if it's value is e.g. [1, 2, 3]

Multiclass

Multiclass Classification

- Binary - only one class
  - Something either belongs or doesn't to a class
- Multiclass
  - Multiple output neurons



Multiclass Classification

- Binary - only one class
  - Something either belongs or doesn't to a class
- Multiclass
  - Multiple output neurons

binary

$\hat{y}$

0.3
0.7
0.1
.
.
.
.
.
.
0.9

```
if yh > 0.5:
    output = 1
else:
    output = 0
```

multiclass

$\hat{y}$

0.7, 0.3
0.3, 0.7
0.9, 0.1
.
.
.
.
.
.
0.1, 0.9

output = ***index of max in row***

Metrics

Metrics - Accuracy

- Number of correct predictions divided by number of total predictions

$$\texttt{accuracy}(y, \hat{y}) = \frac{1}{n_\text{samples}} \sum_{i=0}^{n_\text{samples}-1} 1(\hat{y}_i = y_i)$$

Metrics

- For anything but a simple balanced dataset, accuracy is not the best score to calculate

y	$\hat{y}$
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
1	0
1	1

$$\text{Accuracy} = n_{correct} \;/\; n_{all} = 0.9$$

The dataset is not balanced, meaning, not enough examples to learn the class “1” so the model predicts almost everything as 0.

Metrics - Precision

- Measures the ability of the classifier not to label something as positive that originally is negative

y	ŷ
0	1
0	0
0	0
0	0
0	0
0	0
0	0
0	0
1	0
1	0
1	1

$$\text{precision} = \frac{tp}{tp + fp},$$

tp = 1
fp = 1

precision = 0.5

Metrics - Recall

- Measures the ability of the classifier to find positive examples

y	ŷ
0	1
0	0
0	0
0	0
0	0
0	0
0	0
1	0
1	0
1	1

$$\text{recall} = \frac{tp}{tp + fn}$$

tp = 1
fn = 2

recall = 0.333

Metrics - F1

- Weighted harmonic mean of Precision and Recall

y	ŷ
0	1
0	0
0	0
0	0
0	0
0	0
0	0
1	0
1	0
1	1

$$F_\beta = (1 + \beta^2)\frac{\text{precision} \times \text{recall}}{\beta^2\text{precision} + \text{recall}}$$

p= 0.5
r = 0.333

recall = 0.1665 / 0.833 = 0.2

Unbalanced datasets

- Weighted loss function
  - Each class has a weight
  - The more examples a class has the smaller its weight
  - The loss is higher if the weight for that class is higher

Input Dataset

- Two classes
- size = 3500 examples
- 3000 examples of C_1
- 500 examples of C_2

Weights:

$w_1 = (1 - 3000 / 3500) = 0.143$

$w_2 = (1 - 500 / 3500) = 0.857$

Unbalanced datasets

- Batch sampling
  - Given N - classes
  - During training for each batch we choose the same/similar amount of examples from each class
  - We do not loop over the dataset as before

Input Dataset

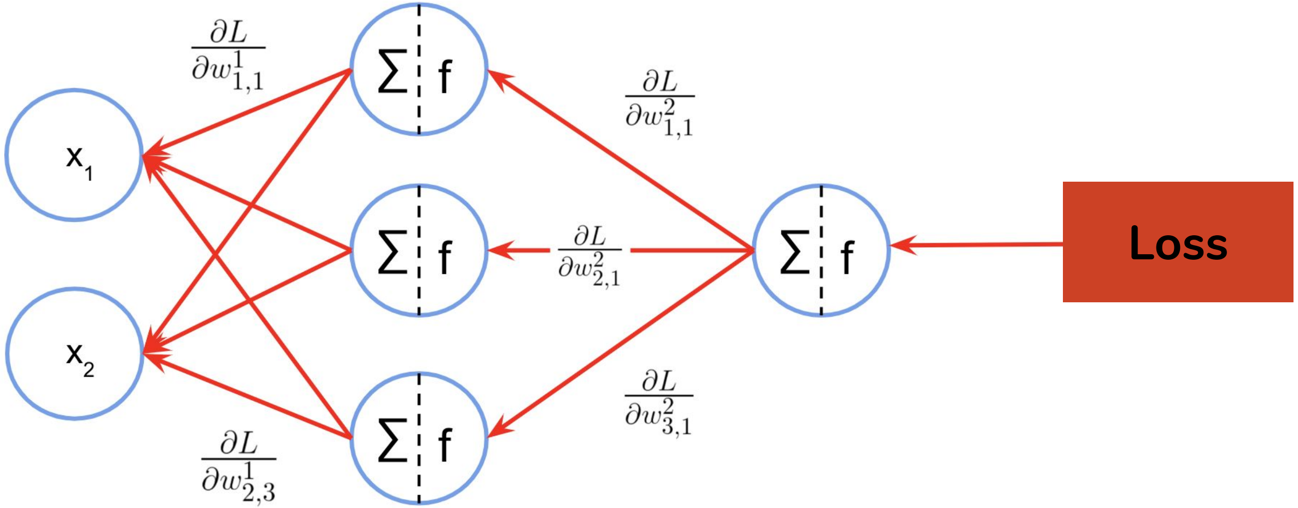
- Two classes
- size = 3500 examples
- 3000 examples of C_1
- 500 examples of C_2

batch_size = 100

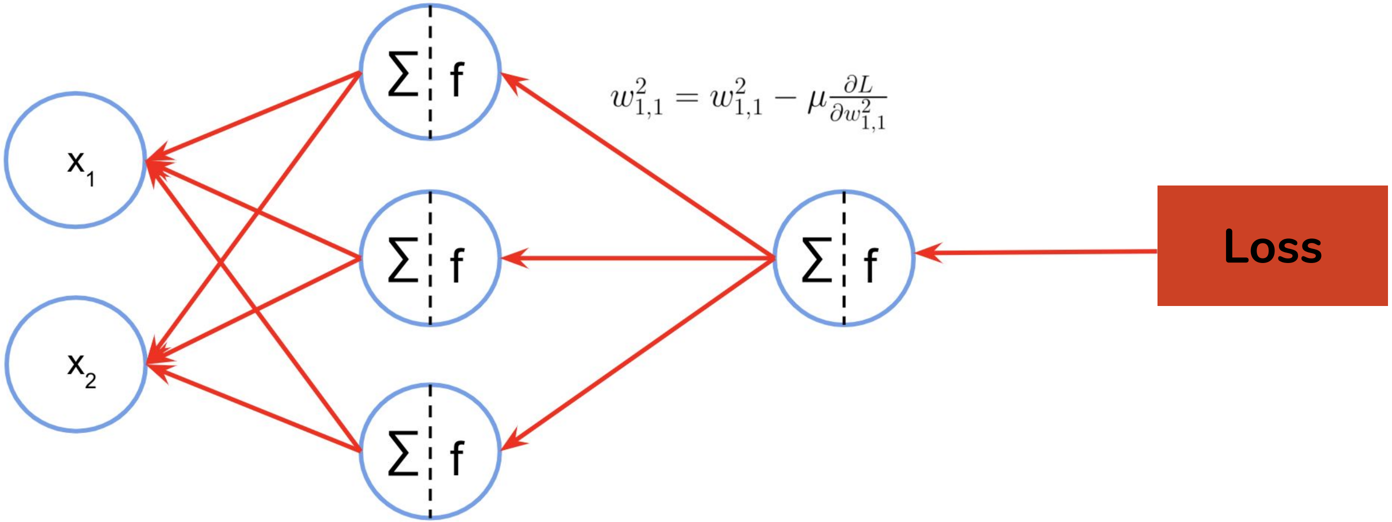
batch = [50 from C_1 **and** 50 from C_2]

Optimizers

Optimizers - Backprop



Optimizers - Stohastic Gradient Descent (SGD)



Why

- Why not always use Gradient Descent
  - Some networks/datasets work better with different optimizers
  - More advanced optimizers usually train the network faster
- Some reasons:
  - Sparse input dataset - usually adaptive learning rates perform better
  - Each weight requires a different learning rate
  - Speed

- In a lot of cases SGD works perfectly it needs some time, but it will get there

Optimizers

- Stochastic Gradient Descent (SGD)
  - For each input example update the weights
- Mini-batch gradient descent
  - Update the weights once per mini-batch
- Gradient Descent
  - Update the weights only one per whole dataset
- Adam
  - Used instead of SGD
  - Individual adaptive learning rates

$$\nu_t = \beta_1 * \nu_{t-1} - (1 - \beta_1) * g_t$$

$$s_t = \beta_2 * s_{t-1} - (1 - \beta_2) * g_t^2$$

$$\Delta\omega_t = -\eta \frac{\nu_t}{\sqrt{s_t + \epsilon}} * g_t$$

$$\omega_{t+1} = \omega_t + \Delta\omega_t$$

η : *Initial Learning rate*

g_t : *Gradient at time t along* ω^j

ν_t : *Exponential Average of gradients along* ω_j

s_t : *Exponential Average of squares of gradients along* ω_j

β_1, β_2 : *Hyperparameters*

Adam: https://arxiv.org/abs/1412.6980

Courses

- If Neural Networks are not so clear (only the first two)
  - Deep Learning Course
- If Python is a problem
  - Go to Aurlies course
- If Machine Learning is a problem
  - Machine Learning with Python

Don't over do it

Papers and News

- How to find papers
  a. Easiest is to go on Twitter
  b. If interested I'll send a list of people to follow in this field
- Read Papers
  a. Kaggle Reading Group on YouTube
     - One of the easiest ways
     - Very slow and sometimes goes through the implementation also
     - Could be too slow for more advanced people

Finally

- This marks the end of our introduction into Neural Networks
- We now can
    - Clean and Tokenize text using SpaCy
    - Build neural networks with as many layers as needed
    - Use different activation functions
    - Use GPUs
    - Regularize - what to do when we are overfitting
    - Use different optimizers
    - How to classify text into as many categories as we want
    - How to Interpret simple networks

Next Time (in two weeks)

- Recurrent neural networks (RNNs)
  - Basic principle
  - When and Why to use it
  - Advantages
  - Drawbacks
  - Language Modeling
  - LSTMs