Neural Network Implementation I

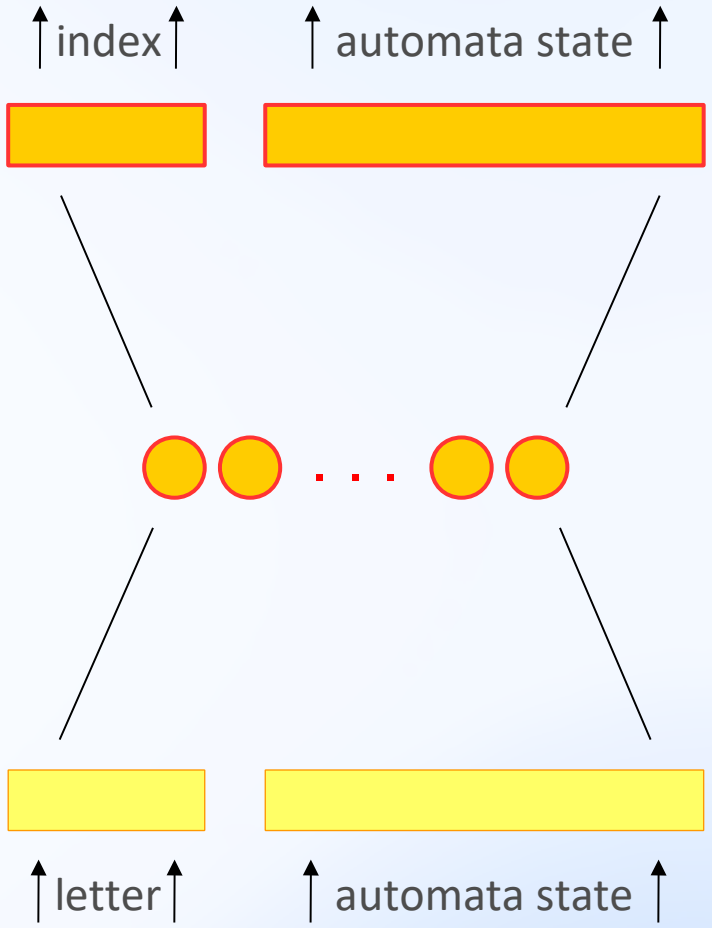
Task 2 - Words

A second debate was cancelled after Trump contracted coronavirus and spent three days in hospital.

- Detection of violent or danger text (e-mails, web)
- Keywords recognition for lexical preprocessing

Task 2 - Words

First approach
Letter by letter



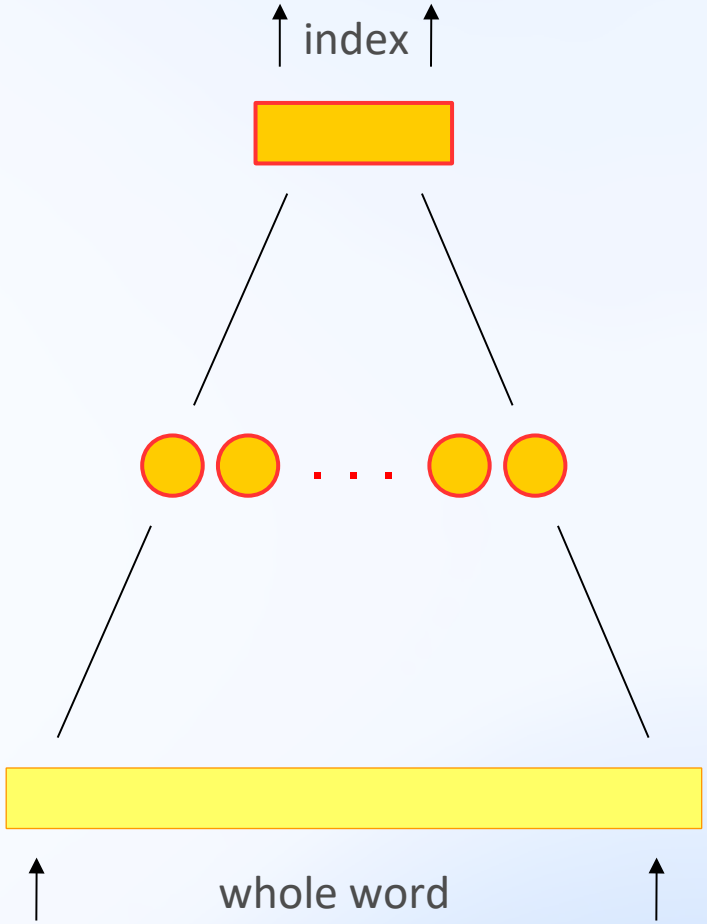
- Input text is presented to the network as letter by letter
- One input vector is composed from one letter plus state of the finite automata that accepts the words
- Output vector contains both word identification (index) and next automata internal state

- Pros:
- Small reading window on text (one letter)
- Tiny input and output vectors

- Cons:
- Preceding construction of the automata
- Adding a new word results in starting from scratch
- A new automata changes not only the training set but also influences the number of neurons in input and output layer

Task 2 - Words

Second approach
Word by word



- Input text is presented to the network as a whole word
- One input vector is composed from all letters of one word
- Output vector contains word identification (index) only

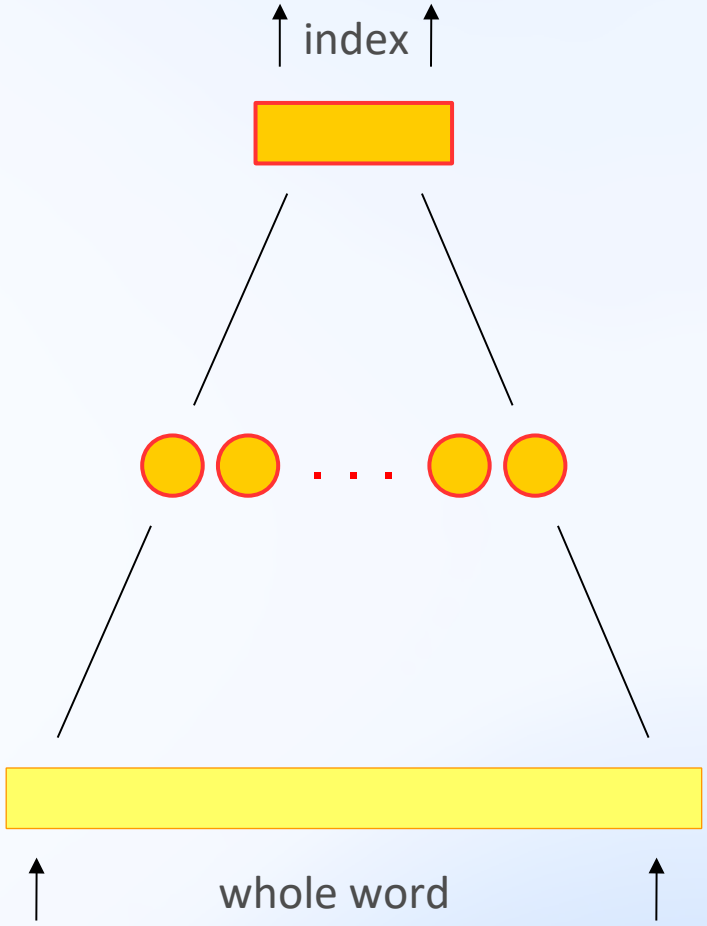
- Pros:
- No finite automata needed, no internal network state
- Possibility of adding new words (online learning)

- Cons:
- Wide reading window covering maximum length of the word
- Existence of false attractors calls for extensive training

- Showing unknown word may result in unexpected results depending on how the network can successfully predict (generalize)

Task 2 - Words

Output layer coding

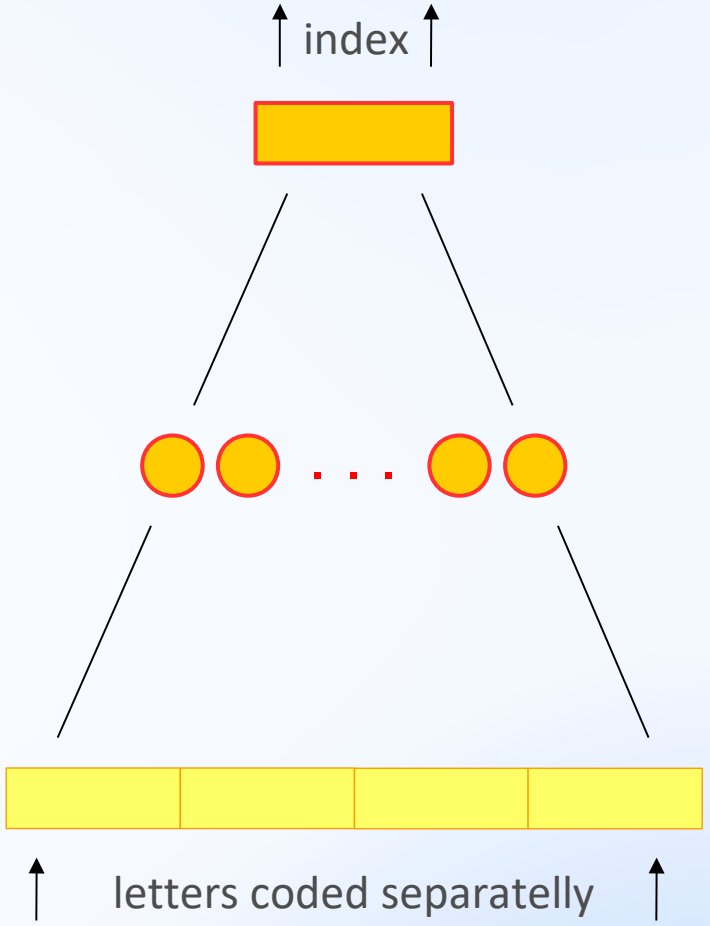


- Often a binary coding is used
- Advantage: it uses just few neurons – ***log(n)*** for ***n*** output cases
- Disadvantage is that you cannot recognize a failure or wrong output
- Also a difference (error) on the bit of higher order is much more significant than expected, i.e. getting 1101 instead od 0101 is worse than getting 0100 – the difference is higher

- When classification is in place unary coding is prefered regardless it consumes more neurons – ***n*** for ***n*** classes
- Higher reliability - you can easily identify one wrong output bit (none or few ones on the output indicates a failure)
- In our case any output neuron can fire for just one keyword
- No activated output neuron means no recognition

Task 2 - Words

Input layer coding



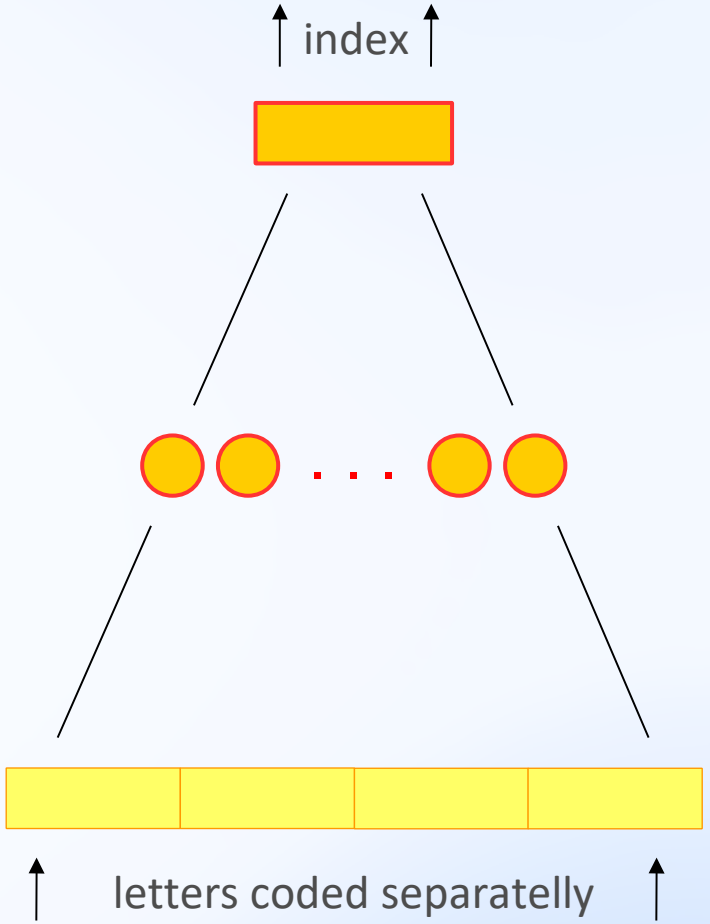
- Whole word presented on the input layer
- Each letter separatelly to its specific set of neurons

- The letter can be coded as an ASCII code 0 to 255 with only one input neuron taking this value
- It reduces the number of input neurons but usually calls for at least one more hidden layer

- The ASCII value can be binary coded as binary inputs to 8 neurons per letter
- Network usually performs better on binary data as it uses less hidden layers (one or two) what speeds up the training

Task 2 - Words

Hidden layer



- We can start with one hidden layer and if unlucky we can use two of them
- Also the binomic formula for the network throughtput takes place

- Imagine 6 keywords with the max. length of 10 characters
- We can simplify the task by padding the words with a space to a fixed length
- Words can use characters A to Z plus the space letter, i.e. 5 bits (neurons) for each letter

- Input layer then has 10*5 = 50 neurons
- Output layer has 6 neurons
- The hidden layer can have some 20-30 neurons
- Binomic formula cannot be used since there are no two adjacent layers with decreasing number of neurons

Task 2 - Words

Common requirements

```
Training progress
Epoch  1    Global error  12.25845
...
Epoch 40    Global error   0.04578

Testing
Input     Output    Response              Error Accuracy Reliability
hello     1 0 0 0   0.92 0.00 0.03 0.08   0.014   100%   100%
begin     0 1 0 0   0.12 0.78 0.95 0.17   0.994    75%   100%
today     0 0 1 0   0.28 0.31 0.58 0.08   0.357   100%    50%
after     0 0 0 1   0.12 0.18 0.14 0.76   0.124   100%   100%
```

Sample output

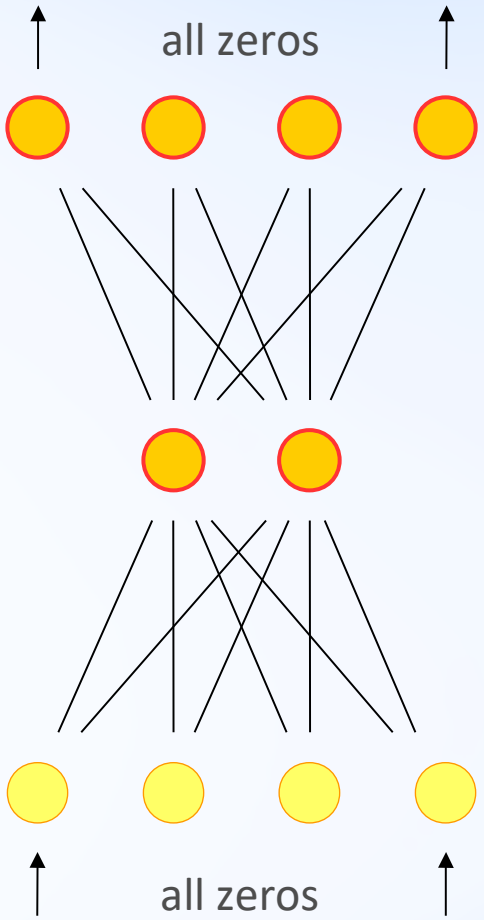
- Send your program sources too

- Send a short description of how you code the data and how you set up the network. For example:
- any input letter is coded in x bits with A as xxx and Z as xxx, space as xxx
- output is coded as follows: xxx
- network has x input neurons, x output neurons, 1 or 2 hidden layers with x-y neurons
- list of training set - if data stored in file send it

- Send both outputs and the intermediate progress of error values in a human readable format

- Run the testing automatically over all the data, not by entering them by hand

Task 2 - Words



- Always prefer to code insignificant data with zeros
- Applies for both input and output data
- Logic: if you do not feed the neuron it should not be excited

- Input data in our case: padding space should be coded as a zero vector

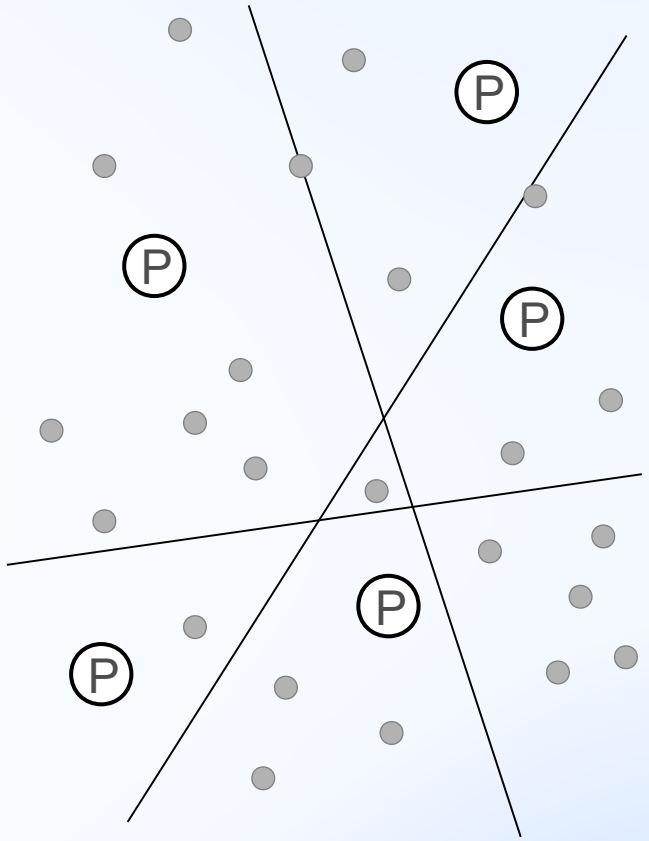
- Output data in our case:
- any recognized pattern fires a neuron and
- no excited output neurons mean that no patterns were detected

- Accuracy can be smaller than reliability

- Global summed error and mean (average) square error differ only in multiplication constant
- Positions of local minima are the same

Task 2 - Words

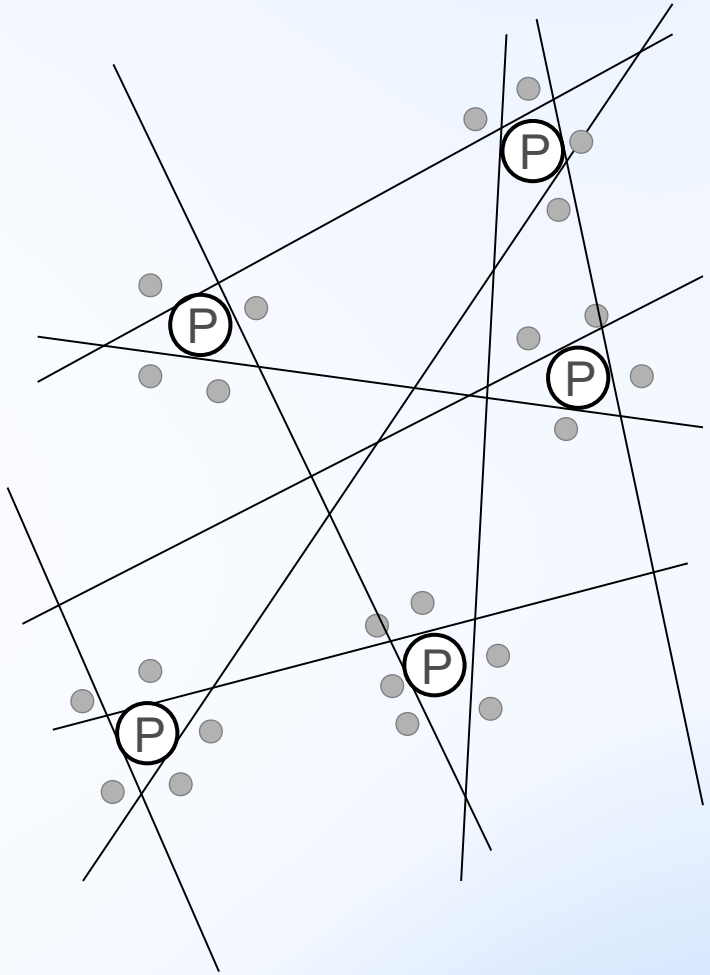
Introducing negative data



- Limited number of positive data trained with 100% accuracy (with no error)
- Huge number (almost unlimited) of negative words (patterns) that must be rejected by the network but cannot be trained as a whole
- You can add many of them to the training set by Monte Carlo method
- Cons:
- How to choose a significant set of negative words – still some of them will not pass through the training process and may generate false acceptance
- Training works as a minimization of the (mean) square error:
- If the number of negative words is much larger than of positive ones then the positive set becomes ignored and the network learns only how to reject all the data (zero weight vector produces the smallest error)
- Solution: multiple instances of positive words with some probability ratio p = 0.4-0.6 of positive to negative words
- Random (shuffled) access to the training data
- New negative data each epoch or even each iteration
- New solution: Choose one of the positive words with a probablity p, otherwise generate a negative word randomly

Task 2 - Words

Separating negative data



- Negative words can be generated totally in random, like xzubefg
- Or coming from the close neighborhood of the positive words
- It forces the network to settle the hyperplanes in very close distance of the positive words and results in inhibiting the false attractors
- To make tight boundaries more hyperplanes are necessary, i.e. more neurons
- Conversely we need a high degree of generalization, keeping the binomic formula as low as possible, i.e. reducing the number of neurons
- Keep the number of hidden neurons as low as possible or add another layer:
- First hidden layer makes hyperplanes
- Second hidden layer makes intersections of the hyperspaces
- Last (output) layer makes the unifications of the convex areas

- How close the negative words and positive one are to be?
- Not too much since it causes troubles in positioning the hyperplanes
- Do not generate negative word by just changing one or two bits
- Rather change one or two characters or add one character to the end of the positive word