Gen AI demystified

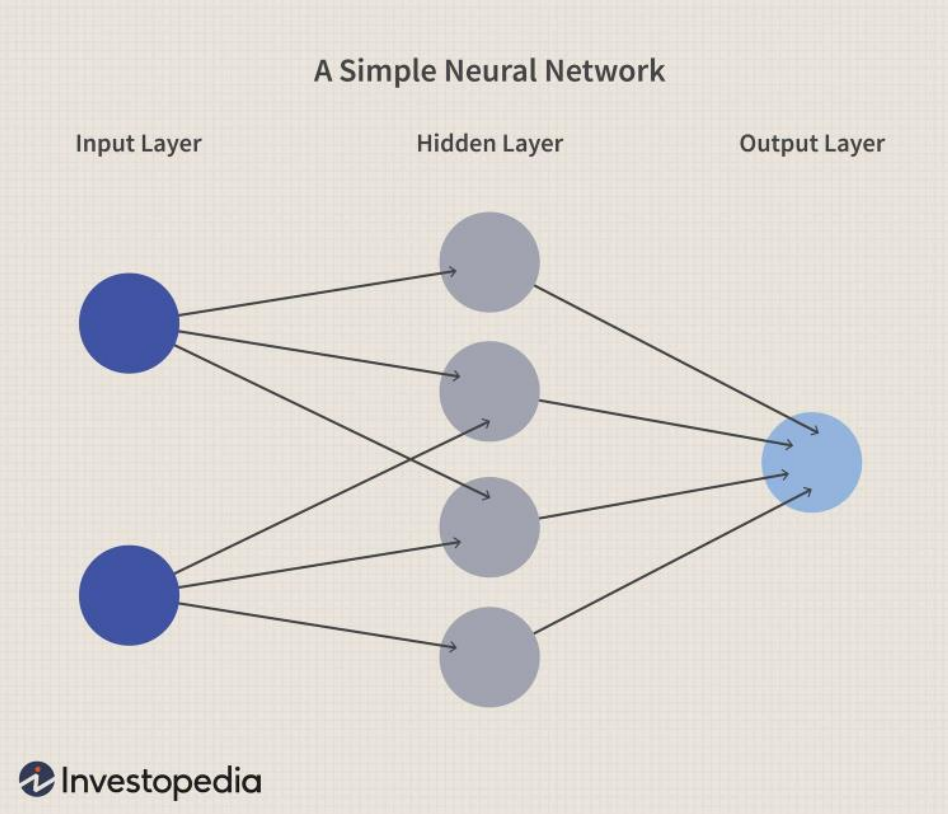
It is not magic just a lot of matrix multiplication.
Quick workshop showing of a very basic model that can name your future baby.

Agenda

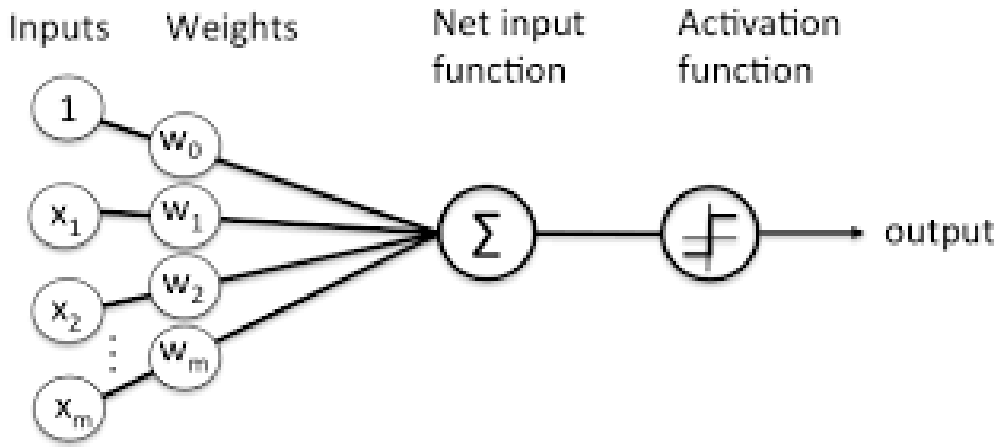
1. Theory
2. Bigram model
3. Multilayer perceptron
4. Transformers as in GPT (generative pre-trained transformer)

Theory

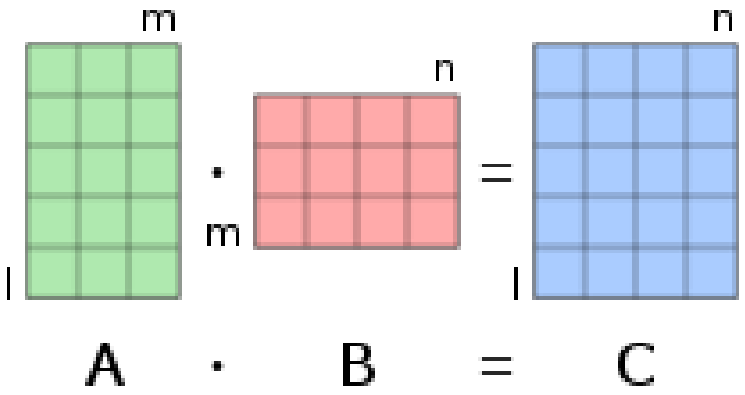
Neural network

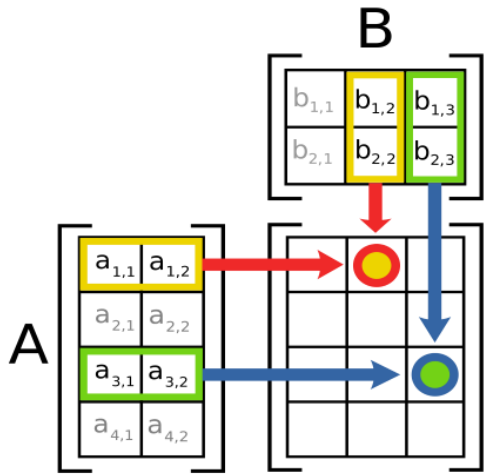


One of the neurons



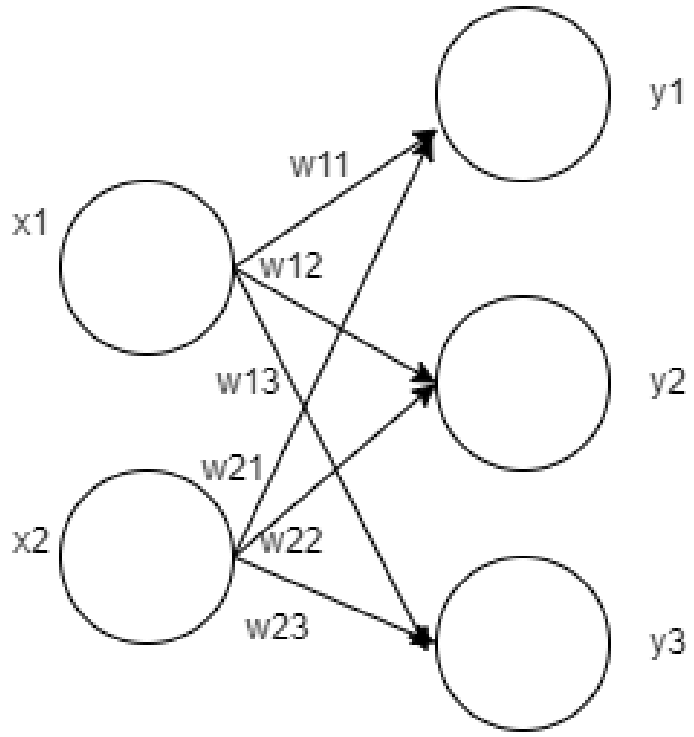
Matrix multiplication





For matrix multiplication, the number of columns in the first matrix must be equal to the number of rows in the second matrix. The result matrix has the number of rows of the first and the number of columns of the second matrix.

Why matrix multiplication is useful

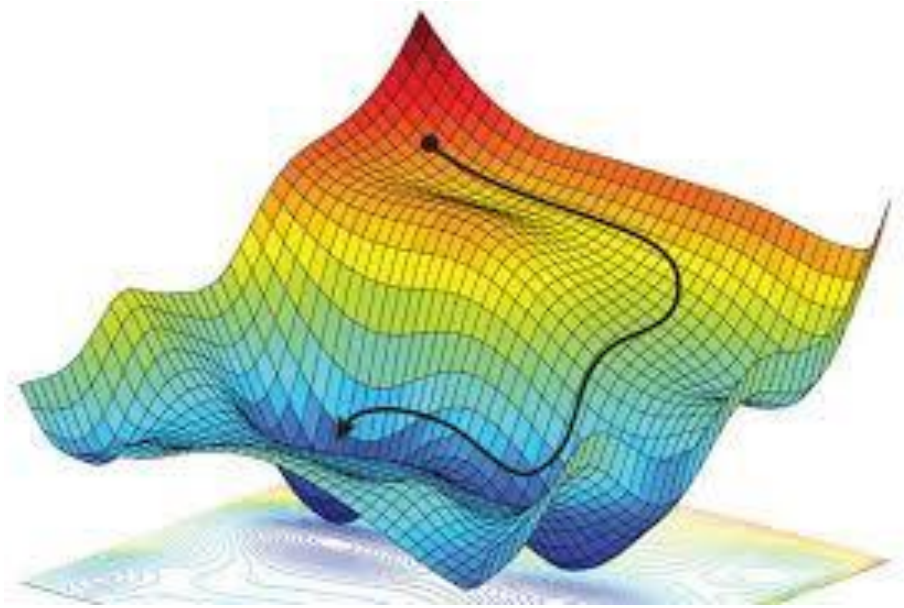


Y1=x1w11+x2w21
Y2=x1w12+x2w22
Y3=x1w13+x2w23

X=[x1,x2]
W=[w11,w12,w13
;

w21,w22,w23]
Y=X*W

Gradient decent



Bigram model

Simple model that only looks at the letter before to predict the next one.

Multilayer perceptron (MLP)

MLP collab link

Multilayer perceptron structure

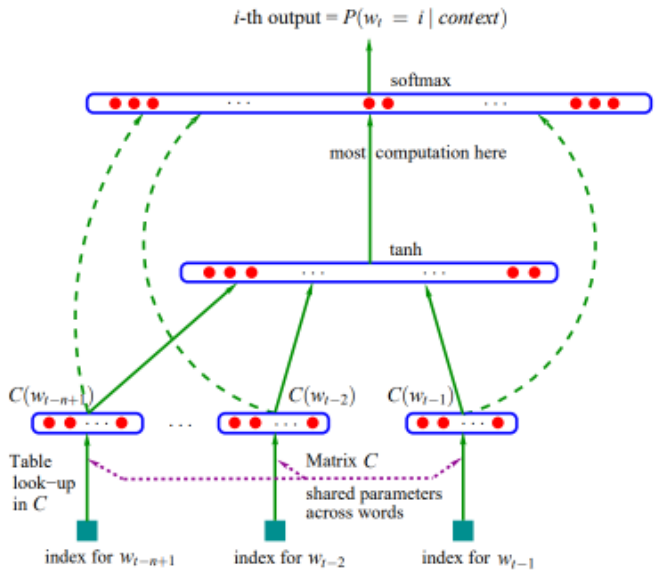


Figure 1: Neural architecture: $f(i, w_{t-1}, \cdots, w_{t-n+1}) = g(i, C(w_{t-1}), \cdots, C(w_{t-n+1}))$ where g is the neural network and $C(i)$ is the i-th word feature vector.

Algorithm for our MLP

- Associate with each token(*for us a character*) in the vocabulary a distributed word feature vector*(in simple words each integer that represents our letter is assigned a vector of length EMBEDING_DIMENSION)*
- Express the joint probability function of token sequences in terms of the feature vectors of these token in the name
- Learn simultaneously the word feature vectors and the parameters of that probability function.

Hyperparameters

- Hyperparameters in machine learning are parameters that are set prior to the training process.
- For our example we will use:
  BLOCK_SIZE = 3

  BATCH_SIZE = 32

  EMBEDING_DIMENSION=10

  HIDDEN_LAYER_SIZE=200

Building data sets

- Important to remember to split the data so we use most of the data for learning. However, some of the data needs to be `hidden` from the model while learning. So, it can be used for verification.
- *build_dataset* in the code

Define the matrixes for the neural net

```
g = torch.Generator().manual_seed(2147483647) # for reproducibility
C = torch.randn((VOCAB_SIZE, EMBEDING_DIMENSION), generator=g)
W1 = torch.randn((EMBEDING_DIMENSION*BLOCK_SIZE, HIDDEN_LAYER_SIZE), generator=g)
b1 = torch.randn(HIDDEN_LAYER_SIZE, generator=g)
W2 = torch.randn((HIDDEN_LAYER_SIZE, VOCAB_SIZE), generator=g)
b2 = torch.randn(VOCAB_SIZE, generator=g)
parameters = [C, W1, b1, W2, b2]
```

Executed at 2024.04.22 16:09:52 in 80ms

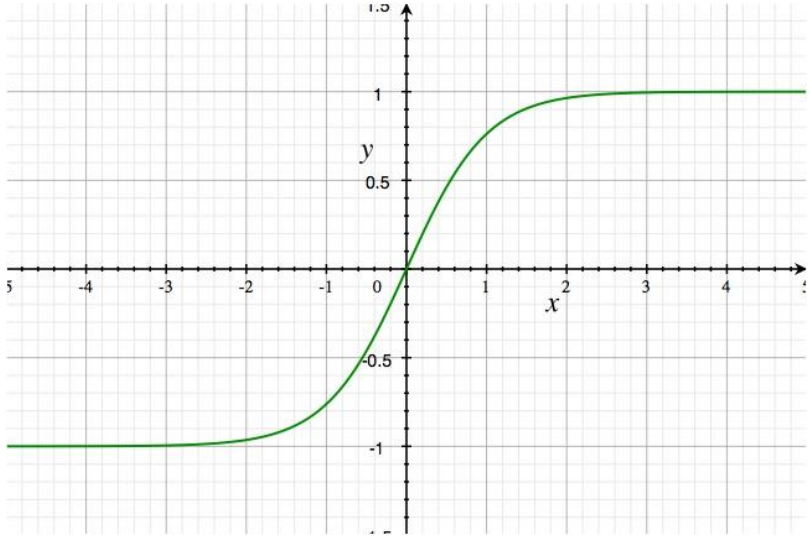
Forward pass (inference)

```
# forward pass
emb = C[Xtr[ix]] # (32, 3, 10)
h = torch.tanh(emb.view(-1, EMBEDING_DIMENSION*BLOCK_SIZE) @ W1 + b1) # (32, 100)
logits = h @ W2 + b2 # (32, 34)
loss = F.cross_entropy(logits, Ytr[ix])
#print(loss.item())
```

Nonlinearity

- Nonlinear activation functions introduce nonlinearity into the network, allowing it to model and learn complex relationships in the data

Tanh:



Pytorch magic

- Indexing in Pytorch:
  You can do C[1] you get row 1
  You can do C[1,2] you get row 1 and 2
  Finally, you can also do C[X] where X is a multidimensional and you get a vector for each element of X
- .view() you can read more here: http://blog.ezyang.com/2019/05/pytorch-internals/
  But the gist of it is matrixes in Pytorch are always one-dimensional vectors in memory and we can view them in any form we want
- Cross entropy:

```
counts = logits.exp()
prob = counts / counts.sum(1, keepdims=True)
loss = -prob[torch.arange(32), Y].log().mean()
```

GPT

Generative pre-trained transformer

GPT architecture

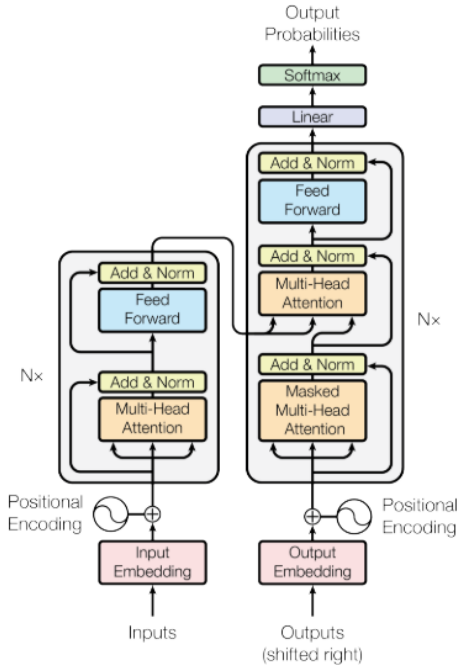


Figure 1: The Transformer - model architecture.

Data for the model

- smallShakespeare.txt concatenated works of Shakespeare
- https://wolnelektury.pl/api/ polish books
- Then specifically refined on “Pan Tadeusz”

Some numbers

- Our model has 10 million parameters
- GPT XL has 1.5 billion
- Estimates put GPT 4 at 1.76 trillion

Reference list

- Andrej Karpathy series on neural networks: https://www.youtube.com/playlist?list=PLAqhIrjkxbuWI23v9cThsA9GvCAUhRvKZ
- Data:https://dane.gov.pl/pl/dataset/1667,lista-imion-wystepujacych-w-rejestrze-pesel-osoby-zyjace
- Base neural network info: https://www.youtube.com/playlist?list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi
- MLP:https://www.jmlr.org/papers/volume3/bengio03a/bengio03a.pdf
- Attention Is All You Need: https://arxiv.org/pdf/1706.03762.pdf
- GPT 2 paper: https://d4mucfpksywv.cloudfront.net/better-language-models/language-models.pdf
- Code and presentation on GitHub: https://github.com/bodzio50318/genAiWorkshop