IN5550: Neural Methods in Natural Language Processing
Sub-lecture 3.2
Basic deep learning

Andrey Kutuzov

University of Oslo

7 February 2023

Contents

Enters deep learning

What exactly is ‘deep’ in deep neural networks?

Enters deep learning

What exactly is 'deep' in deep neural networks?

- Input data goes through successive transformations ϕ at each layer.

Enters deep learning

What exactly is 'deep' in deep neural networks?

- Input data goes through successive transformations ϕ at each layer.
- The transformations themselves are still linear...

Enters deep learning

What exactly is 'deep' in deep neural networks?

- Input data goes through successive transformations ϕ at each layer.
- The transformations themselves are still linear...
- ...but followed with some non-linear operation g at each layer.

Enters deep learning

What exactly is ‘deep’ in deep neural networks?

- Input data goes through successive transformations ϕ at each layer.
- The transformations themselves are still linear...
- ...but followed with some non-linear operation g at each layer.
- At the last layer, the prediction $\hat{\boldsymbol{y}}$ is produced.

Enters deep learning

What exactly is 'deep' in deep neural networks?

- Input data goes through successive transformations ϕ at each layer.
- The transformations themselves are still linear...
- ...but followed with some non-linear operation g at each layer.
- At the last layer, the prediction $\hat{\boldsymbol{y}}$ is produced.
- The whole system is trained simultaneously.

$$\phi(\boldsymbol{x}) = g(\boldsymbol{x} \cdot \boldsymbol{W}') \tag{1}$$

$$\hat{\boldsymbol{y}} = \phi(\boldsymbol{x}) \cdot \boldsymbol{W} \tag{2}$$

Enters deep learning

Feed-forward neural networks

- 'Feed-forward neural network' is a more precise name for a multi-layer perceptron with non-linearities.

Enters deep learning

Feed-forward neural networks

- 'Feed-forward neural network' is a more precise name for a multi-layer perceptron with non-linearities.
- A network in which the neurons/units are connected without cycles.

Enters deep learning

Feed-forward neural networks

- 'Feed-forward neural network' is a more precise name for a multi-layer perceptron with non-linearities.
- A network in which the neurons/units are connected without cycles.
- Outputs from neurons in each layer are passed to neurons in the next layer, multiplied by the correspondent weights.

Enters deep learning

Feed-forward neural networks

- 'Feed-forward neural network' is a more precise name for a multi-layer perceptron with non-linearities.
- A network in which the neurons/units are connected without cycles.
- Outputs from neurons in each layer are passed to neurons in the next layer, multiplied by the correspondent weights.
- A non-linear function is applied element-wise after each layer.

Enters deep learning

Feed-forward neural networks

- 'Feed-forward neural network' is a more precise name for a multi-layer perceptron with non-linearities.
- A network in which the neurons/units are connected without cycles.
- Outputs from neurons in each layer are passed to neurons in the next layer, multiplied by the correspondent weights.
- A non-linear function is applied element-wise after each layer.
- No outputs are passed back to previous layers.

Feed-forward neural networks

- 'Feed-forward neural network' is a more precise name for a multi-layer perceptron with non-linearities.
- A network in which the neurons/units are connected without cycles.
- Outputs from neurons in each layer are passed to neurons in the next layer, multiplied by the correspondent weights.
- A non-linear function is applied element-wise after each layer.
- No outputs are passed back to previous layers.

1. The first layer contains input units,

Enters deep learning

Feed-forward neural networks

- 'Feed-forward neural network' is a more precise name for a multi-layer perceptron with non-linearities.
- A network in which the neurons/units are connected without cycles.
- Outputs from neurons in each layer are passed to neurons in the next layer, multiplied by the correspondent weights.
- A non-linear function is applied element-wise after each layer.
- No outputs are passed back to previous layers.

1. The first layer contains input units,
2. the last layer contains output units,

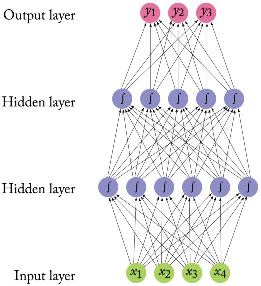
Enters deep learning

Feed-forward neural networks

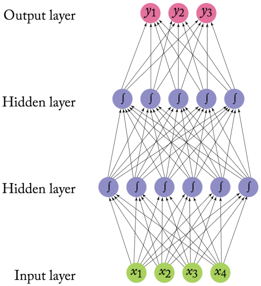
- 'Feed-forward neural network' is a more precise name for a multi-layer perceptron with non-linearities.
- A network in which the neurons/units are connected without cycles.
- Outputs from neurons in each layer are passed to neurons in the next layer, multiplied by the correspondent weights.
- A non-linear function is applied element-wise after each layer.
- No outputs are passed back to previous layers.

1. The first layer contains input units,
2. the last layer contains output units,
3. all layers in between (hidden layers) contain hidden units, which form hidden representations of the input data.

Enters deep learning

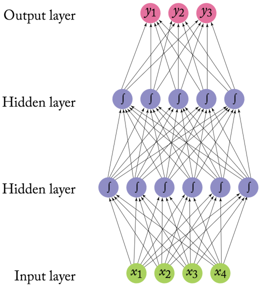


Enters deep learning



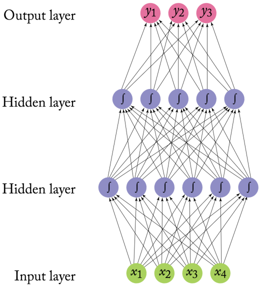
- Feed-forward network with 2 hidden layers;

Enters deep learning



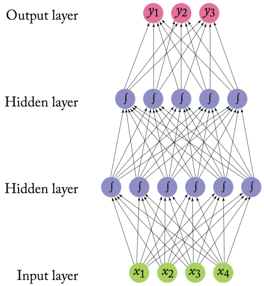
- Feed-forward network with 2 hidden layers;
- Before the transition to the l^{n+1}, the output vector of l^n goes through a non-linear function (instead of simple weighted sum).

Enters deep learning



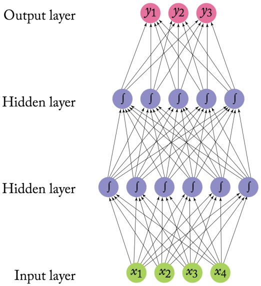
- Feed-forward network with 2 hidden layers;
- Before the transition to the l^{n+1}, the output vector of l^n goes through a non-linear function (instead of simple weighted sum).
- Fully-connected: transitions between layers are linear transformations:

Enters deep learning



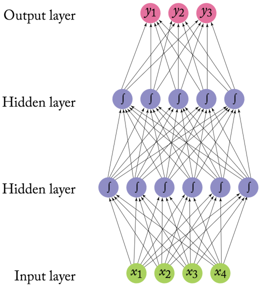
- Feed-forward network with 2 hidden layers;
- Before the transition to the l^{n+1}, the output vector of l^n goes through a non-linear function (instead of simple weighted sum).
- Fully-connected: transitions between layers are linear transformations:
  - each neuron in the layer l^n is connected to each neuron in the layer l^{n+1}

Enters deep learning



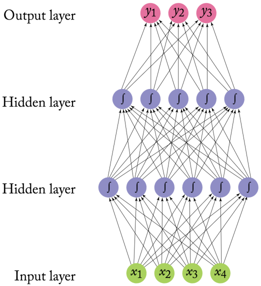
- Feed-forward network with 2 hidden layers;
- Before the transition to the l^{n+1}, the output vector of l^n goes through a non-linear function (instead of simple weighted sum).
- Fully-connected: transitions between layers are linear transformations:
  - each neuron in the layer l^n is connected to each neuron in the layer l^{n+1}
  - ... mathematically, a vector-matrix multiplication $\boldsymbol{x^n} \cdot \boldsymbol{W^n} = \boldsymbol{x^{n+1}}$

Enters deep learning



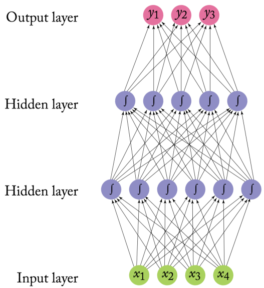
- Feed-forward network with 2 hidden layers;
- Before the transition to the l^{n+1}, the output vector of l^n goes through a non-linear function (instead of simple weighted sum).
- Fully-connected: transitions between layers are linear transformations:
  - each neuron in the layer l^n is connected to each neuron in the layer l^{n+1}
  - ... mathematically, a vector-matrix multiplication $\boldsymbol{x^n} \cdot \boldsymbol{W^n} = \boldsymbol{x^{n+1}}$
- NB: not all neural architectures are fully connected (more on that in the next lectures).

Enters deep learning



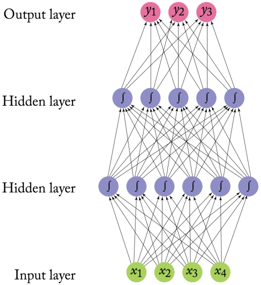
- Feed-forward network with 2 hidden layers;
- Before the transition to the l^{n+1}, the output vector of l^n goes through a non-linear function (instead of simple weighted sum).
- Fully-connected: transitions between layers are linear transformations:
  - each neuron in the layer l^n is connected to each neuron in the layer l^{n+1}
  - ... mathematically, a vector-matrix multiplication $\boldsymbol{x^n} \cdot \boldsymbol{W^n} = \boldsymbol{x^{n+1}}$
- NB: not all neural architectures are fully connected (more on that in the next lectures).
- Question: **total number of trainable weights in this network?**

Enters deep learning



- Bias is added trivially to each layer (often except the last one).

Enters deep learning

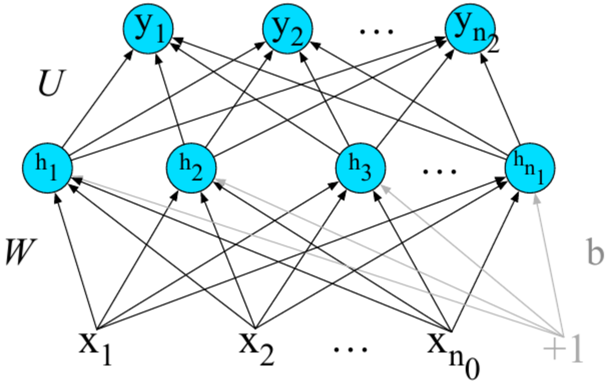


- Bias is added trivially to each layer (often except the last one).
- Then, such a neural network is described with:

$$\hat{\boldsymbol{y}} = g^2(g^1(\boldsymbol{x} \cdot \boldsymbol{W^1} + \boldsymbol{b^1})) \cdot \boldsymbol{W^2} + \boldsymbol{b^2}) \cdot \boldsymbol{W^3} \tag{3}$$

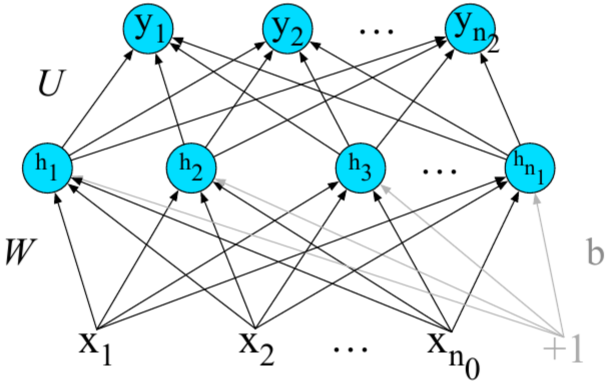
- $\boldsymbol{x} \in \mathbb{R}^4, \boldsymbol{W^1} \in \mathbb{R}^{4\times 6}, \boldsymbol{b^1} \in \mathbb{R}^6, \boldsymbol{W^2} \in \mathbb{R}^{6\times 5}, \boldsymbol{b^1} \in \mathbb{R}^5, \boldsymbol{W^3} \in \mathbb{R}^{5\times 3}$
- g^1, g^2 are non-linear activation functions.

Enters deep learning



Feed-forward network with one hidden layer and bias
(from Jurafsky and Martin, 2023)

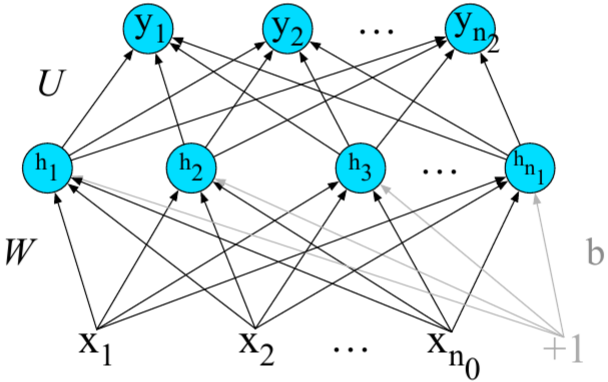
Enters deep learning



Feed-forward network with one hidden layer and bias
(from Jurafsky and Martin, 2023)

Thus, what is 'deep'?

Enters deep learning



Feed-forward network with one hidden layer and bias
(from Jurafsky and Martin, 2023)

Thus, what is 'deep'? Multiple layers and non-linearities between them.

Enters deep learning

Important: **feed forward neural networks with multiple layers and non-linear transformations can theoretically approximate any function** (given enough layers and neurons).

[Leshno et al., 1993].

Enters deep learning

Unfortunately, that gives no guarantees to real word problems:

Enters deep learning

Unfortunately, that gives no guarantees to real word problems:

- Is such a network learnable?

Enters deep learning

Unfortunately, that gives no guarantees to real word problems:

- Is such a network learnable?
- Can we approximate the true function given only limited amounts of training data?

Enters deep learning

Unfortunately, that gives no guarantees to real word problems:

- Is such a network learnable?
- Can we approximate the true function given only limited amounts of training data?
- Anyway, MLPs are still good baselines for many tasks.

Network parameters

- Like in linear classifiers, weights and biases are parameters of the network;
- a.k.a. θ;

Enters deep learning

Network parameters

- Like in linear classifiers, weights and biases are parameters of the network;
- a.k.a. θ;
- the aim of the training is to learn optimal values for θ.

Enters deep learning

Network parameters

- Like in linear classifiers, weights and biases are parameters of the network;
- a.k.a. θ;
- the aim of the training is to learn optimal values for θ.
- Unlike in linear classifiers, the error/loss function is usually not convex...

Enters deep learning

Network parameters

- Like in linear classifiers, weights and biases are parameters of the network;
- a.k.a. θ;
- the aim of the training is to learn optimal values for θ.
- Unlike in linear classifiers, the error/loss function is usually not convex...
- ...but gradient-based optimizers still work well in practice.

Network parameters

- Like in linear classifiers, weights and biases are parameters of the network;
- a.k.a. θ;
- the aim of the training is to learn optimal values for θ.
- Unlike in linear classifiers, the error/loss function is usually not convex...
- ...but gradient-based optimizers still work well in practice.
- NB: one can balance between the number of layers and their sizes (number of neurons).

Enters deep learning

Network parameters

- Like in linear classifiers, weights and biases are parameters of the network;
- a.k.a. θ;
- the aim of the training is to learn optimal values for θ.
- Unlike in linear classifiers, the error/loss function is usually not convex...
- ...but gradient-based optimizers still work well in practice.
- NB: one can balance between the number of layers and their sizes (number of neurons).
- Different layers learn different hidden representations
  - eliminates the need to manually design feature combinations:

Enters deep learning

Network parameters

- Like in linear classifiers, weights and biases are parameters of the network;
- a.k.a. θ;
- the aim of the training is to learn optimal values for θ.
- Unlike in linear classifiers, the error/loss function is usually not convex...
- ...but gradient-based optimizers still work well in practice.
- NB: one can balance between the number of layers and their sizes (number of neurons).
- Different layers learn different hidden representations
  - eliminates the need to manually design feature combinations:
  - the network is able to learn that '*not*' and '*good*' co-occurrence in a text is a powerful feature in itself...

Network parameters

- Like in linear classifiers, weights and biases are parameters of the network;
- a.k.a. θ;
- the aim of the training is to learn optimal values for θ.
- Unlike in linear classifiers, the error/loss function is usually not convex...
- ...but gradient-based optimizers still work well in practice.
- NB: one can balance between the number of layers and their sizes (number of neurons).
- Different layers learn different hidden representations
  - eliminates the need to manually design feature combinations:
  - the network is able to learn that '*not*' and '*good*' co-occurrence in a text is a powerful feature in itself...
  - ...and reflect this in its hidden representations.

Enters deep learning

There is a cool interactive visualization from Andrei Karpathy (Stanford) which can give you some clue about how different variables in deep networks interact and influence learned decision boundaries.

Let's look at it to develop intuitions.

`https://cs.stanford.edu/people/karpathy/convnetjs/demo/classify2d.html`

Play with the demo (will discuss at the Q&A / group session as well)

1. Use a 2 layer linear classifier for the spiral data (change '*tanh*' to '*linear*' for all layers). What happens?

Enters deep learning

Play with the demo (will discuss at the Q&A / group session as well)

1. Use a 2 layer linear classifier for the spiral data (change '*tanh*' to '*linear*' for all layers). What happens?
2. Add more neurons ('*num_neurons*'=20)

Enters deep learning

Play with the demo (will discuss at the Q&A / group session as well)

1. Use a 2 layer linear classifier for the spiral data (change '*tanh*' to '*linear*' for all layers). What happens?
2. Add more neurons ('*num_neurons*'=20)
3. Add more layers (copy and paste them). What changes?

Enters deep learning

Play with the demo (will discuss at the Q&A / group session as well)

1. Use a 2 layer linear classifier for the spiral data (change '*tanh*' to '*linear*' for all layers). What happens?
2. Add more neurons ('*num_neurons*'=20)
3. Add more layers (copy and paste them). What changes?
4. Then try a 2 layer non-linear classifier (change to '*tanh*'). Any difference?

Enters deep learning

Play with the demo (will discuss at the Q&A / group session as well)

1. Use a 2 layer linear classifier for the spiral data (change '*tanh*' to '*linear*' for all layers). What happens?
2. Add more neurons ('*num_neurons*'=20)
3. Add more layers (copy and paste them). What changes?
4. Then try a 2 layer non-linear classifier (change to '*tanh*'). Any difference?
5. Add more non-linear layers. What happens?

Enters deep learning

Play with the demo (will discuss at the Q&A / group session as well)

1. Use a 2 layer linear classifier for the spiral data (change '*tanh*' to '*linear*' for all layers). What happens?
2. Add more neurons ('*num_neurons*'=20)
3. Add more layers (copy and paste them). What changes?
4. Then try a 2 layer non-linear classifier (change to '*tanh*'). Any difference?
5. Add more non-linear layers. What happens?
6. Add more neurons. Any changes?

Enters deep learning

Play with the demo (will discuss at the Q&A / group session as well)

1. Use a 2 layer linear classifier for the spiral data (change '*tanh*' to '*linear*' for all layers). What happens?
2. Add more neurons ('*num_neurons*'=20)
3. Add more layers (copy and paste them). What changes?
4. Then try a 2 layer non-linear classifier (change to '*tanh*'). Any difference?
5. Add more non-linear layers. What happens?
6. Add more neurons. Any changes?
7. Try using different non-linearities ('*sigmoid*', '*tanh*', '*relu*'). What differences do you notice?

Enters deep learning

Play with the demo (will discuss at the Q&A / group session as well)

1. Use a 2 layer linear classifier for the spiral data (change '*tanh*' to '*linear*' for all layers). What happens?
2. Add more neurons ('*num_neurons*'=20)
3. Add more layers (copy and paste them). What changes?
4. Then try a 2 layer non-linear classifier (change to '*tanh*'). Any difference?
5. Add more non-linear layers. What happens?
6. Add more neurons. Any changes?
7. Try using different non-linearities ('*sigmoid*', '*tanh*', '*relu*'). What differences do you notice?

The next sub-lecture 3.3: practicalities of training deep neural nets.

References I

Leshno, M., Lin, V. Y., Pinkus, A., and Schocken, S. (1993).
Multilayer feedforward networks with a nonpolynomial activation function can approximate any function.
Neural Networks, 6(6):861–867.