IN5550: Neural Methods in Natural Language Processing
Sub-lecture 2.4
From linear models to neural networks

Andrey Kutuzov

University of Oslo

31 January 2023

Contents

1 Advantages and limitations of linear models

2 Going deeply non-linear: multi-layered perceptrons

3 What's next?

Advantages and limitations of linear models

- Linear classifiers are efficient and effective.

Advantages and limitations of linear models

- Linear classifiers are efficient and effective.
- They are interpretable to a degree: you can find the most important features by looking at the weights

Advantages and limitations of linear models

- Linear classifiers are efficient and effective.
- They are interpretable to a degree: you can find the most important features by looking at the weights
- Can be used on their own (often enough in practice)...
- ...or as building blocks for non-linear neural classifiers.

Advantages and limitations of linear models

- Linear classifiers are efficient and effective.
- They are interpretable to a degree: you can find the most important features by looking at the weights
- Can be used on their own (often enough in practice)...
- ...or as building blocks for non-linear neural classifiers.

Unfortunately, linear models can represent only linear relations in the data

Advantages and limitations of linear models

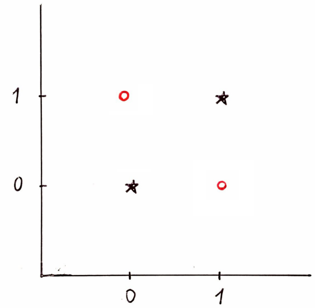
- Are there non-linear functions that linear models can't deal with?

Advantages and limitations of linear models

- Are there non-linear functions that linear models can't deal with?
- Yes, there are.

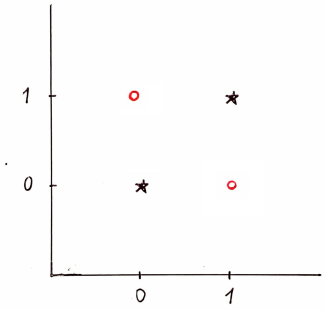
Advantages and limitations of linear models

- Are there non-linear functions that linear models can't deal with?
- Yes, there are.
- One example is the XOR ('excluding OR') function:



Advantages and limitations of linear models

- Are there non-linear functions that linear models can't deal with?
- Yes, there are.
- One example is the XOR ('excluding OR') function:



It is clearly not linearly separable.

Possible solutions

- We can transform the input so that it becomes linearly separable.

Advantages and limitations of linear models

Possible solutions

- We can transform the input so that it becomes linearly separable.
- Linear transformations will not be able to do this.

Possible solutions

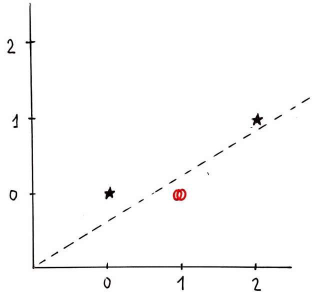
- We can transform the input so that it becomes linearly separable.
- Linear transformations will not be able to do this.
- We need non-linear transformations.

Advantages and limitations of linear models

Possible solutions

- We can transform the input so that it becomes linearly separable.
- Linear transformations will not be able to do this.
- We need non-linear transformations.

For example, $\phi(x_1, x_2) = [x_1 + x_2, x_1 \times x_2]$ maps the instances to another representation and makes the XOR problem linearly separable:



Advantages and limitations of linear models

- But how to find the transformation function ϕ suitable for the task at hand?

Advantages and limitations of linear models

- But how to find the transformation function ϕ suitable for the task at hand?
- Often, this implies mapping instances to a higher-dimensional space, making it even more difficult to choose ϕ manually.

Advantages and limitations of linear models

- But how to find the transformation function ϕ suitable for the task at hand?
- Often, this implies mapping instances to a higher-dimensional space, making it even more difficult to choose ϕ manually.
- Support Vector Machines (SVM) classifiers handle this to some extent...
  [Cortes and Vapnik, 1995]

Advantages and limitations of linear models

- But how to find the transformation function ϕ suitable for the task at hand?
- Often, this implies mapping instances to a higher-dimensional space, making it even more difficult to choose ϕ manually.
- Support Vector Machines (SVM) classifiers handle this to some extent...
  [Cortes and Vapnik, 1995]
- ...but they scale linearly in time on the size of the training data (slow!).

Advantages and limitations of linear models

Training mapping functions

- Idea: leave it for the algorithm to train a suitable representation mapping function!

Advantages and limitations of linear models

Training mapping functions

- Idea: leave it for the algorithm to train a suitable representation mapping function!

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

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

- ...where g is a non-linear activation function, and $\boldsymbol{W}', \boldsymbol{b}'$ are its trainable parameters.

Advantages and limitations of linear models

Training mapping functions

- Idea: leave it for the algorithm to train a suitable representation mapping function!

$$\phi(\boldsymbol{x}) = g(\boldsymbol{x}\boldsymbol{W}' + \boldsymbol{b}') \tag{1}$$
$$\hat{\boldsymbol{y}} = \phi(\boldsymbol{x})\boldsymbol{W} + \boldsymbol{b} \tag{2}$$

- ...where g is a non-linear activation function, and $\boldsymbol{W}', \boldsymbol{b}'$ are its trainable parameters.
- The equation above defines a simple multi-layer perceptron (MLP): our first neural model.

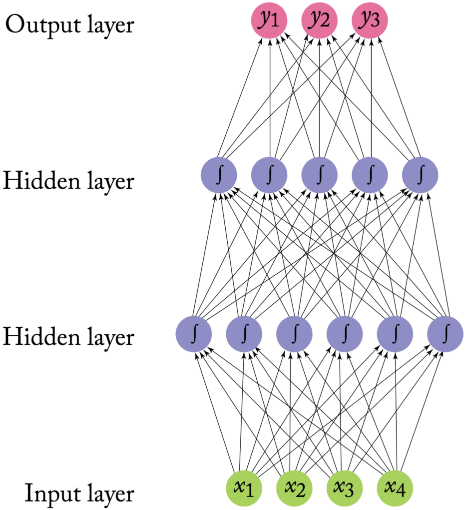
Contents

1 Advantages and limitations of linear models

2 Going deeply non-linear: multi-layered perceptrons

3 What's next?

Going deeply non-linear: multi-layered perceptrons



Perceptron with 2 hidden layers

Going deeply non-linear: multi-layered perceptrons

The nature of perceptrons

- Input data goes through successive transformations at each layer.

Going deeply non-linear: multi-layered perceptrons

The nature of perceptrons

- Input data goes through successive transformations at each layer.
- The transformations are linear, but followed with a non-linear activation at each hidden layer.

The nature of perceptrons

- Input data goes through successive transformations at each layer.
- The transformations are linear, but followed with a non-linear activation at each hidden layer.
- At the last layer, the prediction $\hat{y}$ is produced.

Going deeply non-linear: multi-layered perceptrons

The nature of perceptrons

- Input data goes through successive transformations at each layer.
- The transformations are linear, but followed with a non-linear activation at each hidden layer.
- At the last layer, the prediction $\hat{y}$ is produced.
- Representation functions and the linear classifiers are trained simultaneously.

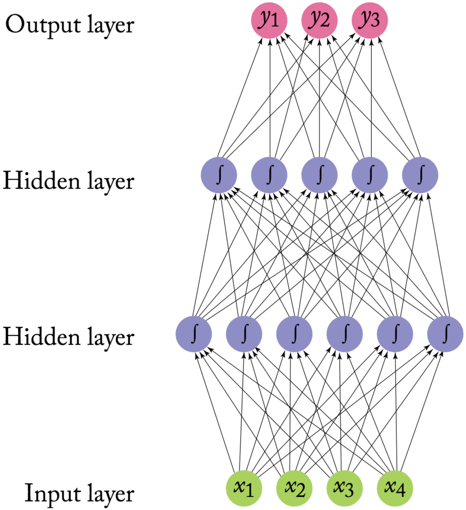
Going deeply non-linear: multi-layered perceptrons

The nature of perceptrons

- Input data goes through successive transformations at each layer.
- The transformations are linear, but followed with a non-linear activation at each hidden layer.
- At the last layer, the prediction $\hat{y}$ is produced.
- Representation functions and the linear classifiers are trained simultaneously.

Important: **neural networks with hidden layers can theoretically approximate any function** [Leshno et al., 1993].

Going deeply non-linear: multi-layered perceptrons



A simple feed-forward neural network. More next week!

Contents

1 Advantages and limitations of linear models

2 Going deeply non-linear: multi-layered perceptrons

3 What's next?

What's next?

Q&A session, January 31

- Ask questions about the pre-recorded lectures published on January 27
- Group work and discussions
- ...

What's next?

Q&A session, January 31

- Ask questions about the pre-recorded lectures published on January 27
- Group work and discussions
- ...

Next week: Training Deep Neural Networks

- More on multi-layer perceptrons and feed-forward neural networks.
- Are they really like brain?
- Common activation functions.
- Regularizing neural networks with dropout.
- Computation graphs.

What's next?

Obligatory assignment

- The first obligatory assignment will be out on January 31!
- Due February 15.
- Look it up on the course page.

What's next?

Obligatory assignment

- The first obligatory assignment will be out on January 31!
- Due February 15.
- Look it up on the course page.

Group session February 1st

- Hands-on: PyTorch basics
- Hands-on: Data loading and linear model building with PyTorch.
- Make sure you have access to *Fox*!

What's next?

Obligatory assignment

- The first obligatory assignment will be out on January 31!
- Due February 15.
- Look it up on the course page.

Group session February 1st

- Hands-on: PyTorch basics
- Hands-on: Data loading and linear model building with PyTorch.
- Make sure you have access to *Fox*!

References I

Cortes, C. and Vapnik, V. (1995).
Support-vector networks.
In Machine Learning, pages 273–297.

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.