

## 5.1 Neural Networks

### *Machine Learning 1: Foundations*

Marius Kloft (TUK)

# Recap

Previous lectures: **linear** and/or **kernel** learning methods.

- ▶ learn a linear classifier  $f(\mathbf{x}) = \text{sign}(\mathbf{w}^\top \mathbf{x} + b)$ .
- ▶ learn a kernel classifier  $f(\mathbf{x}) = \text{sign}(\sum_{i=1}^n \alpha_i k(\mathbf{x}_i, \mathbf{x}) + b)$ .

# Recap

Previous lectures: **linear** and/or **kernel** learning methods.

- ▶ learn a linear classifier  $f(\mathbf{x}) = \text{sign}(\mathbf{w}^\top \mathbf{x} + b)$ .
- ▶ learn a kernel classifier  $f(\mathbf{x}) = \text{sign}(\sum_{i=1}^n \alpha_i k(\mathbf{x}_i, \mathbf{x}) + b)$ .

Quiz: What are the advantages of these classifiers?  
What can be improved?

Neural networks can **learn**  
a feature representation.

# Contents of this Class

## Neural Networks

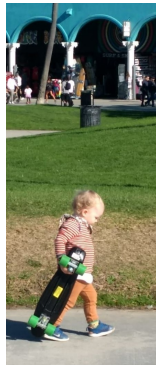
- 1 The Core Idea in a Nutshell
- 2 Natural Neural Networks
- 3 Artificial Neural Networks
- 4 Convolutional Neural Networks

- 1 The Core Idea in a Nutshell
- 2 Natural Neural Networks
- 3 Artificial Neural Networks
- 4 Convolutional Neural Networks

# Example: State of the Art in Image Classification

Before 2012: two steps

- 1 design some hand-crafted features
- 2 train an SVM on these features



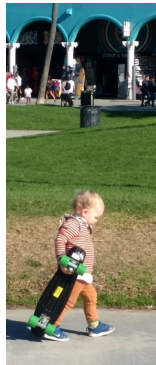
# Example: State of the Art in Image Classification

Before 2012: two steps

- 1 design some hand-crafted features
- 2 train an SVM on these features

After 2012: train a deep convolutional neural network

- ▶ learns the **features** and the classifier together in one go
  - ▶ this means the classifier—not a human expert—guides the design of the features





# Problem of Learning Good Feature Representations

# Problem of Learning Good Feature Representations

- Say we are given images  $\mathbf{x}_1, \dots, \mathbf{x}_n$



# Problem of Learning Good Feature Representations

- ▶ Say we are given images  $\mathbf{x}_1, \dots, \mathbf{x}_n$
- ▶ Want to learn a map  $\phi$  that assigns any image  $\mathbf{x}$  with a vector representation  $\phi(\mathbf{x})$



$$\phi \rightarrow \begin{pmatrix} 0.3 \\ \vdots \\ 1.4 \end{pmatrix}$$

# Problem of Learning Good Feature Representations

- ▶ Say we are given images  $\mathbf{x}_1, \dots, \mathbf{x}_n$
- ▶ Want to learn a map  $\phi$  that assigns any image  $\mathbf{x}$  with a vector representation  $\phi(\mathbf{x})$



$$\phi \rightarrow \begin{pmatrix} 0.3 \\ \vdots \\ 1.4 \end{pmatrix}$$

How can we learn a good feature representation  $\phi$ ?

# Core Idea to Learn a Good Feature Representation $\phi$

# Core Idea to Learn a Good Feature Representation $\phi$

Let the learning machine figure it out!

# Core Idea to Learn a Good Feature Representation $\phi$

Let the learning machine figure it out!

Recall logistic regression:

$$\min_{b \in \mathbb{R}, \mathbf{w} \in \mathbb{R}^d} \quad \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^n \ln \left( 1 + \exp \left( -y_i (\mathbf{w}^\top \mathbf{x}_i + b) \right) \right)$$

# Core Idea to Learn a Good Feature Representation $\phi$

Let the learning machine figure it out!

Recall logistic regression:

$$\min_{b \in \mathbb{R}, \mathbf{w} \in \mathbb{R}^d} \quad \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^n \ln \left( 1 + \exp \left( -y_i (\mathbf{w}^T \phi(\mathbf{x}_i) + b) \right) \right)$$

- Want to learn  $\phi$



# Core Idea to Learn a Good Feature Representation $\phi$

Let the learning machine figure it out!

Recall logistic regression:

$$\min_{b \in \mathbb{R}, \mathbf{w} \in \mathbb{R}^d, \phi} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^n \ln \left( 1 + \exp \left( -y_i (\mathbf{w}^\top \phi(\mathbf{x}_i) + b) \right) \right)$$

- ▶ Want to learn  $\phi$
- ▶ Idea: Optimize also over  $\phi$ !

# Core Idea to Learn a Good Feature Representation $\phi$

Let the learning machine figure it out!

Recall logistic regression:

$$\min_{b \in \mathbb{R}, \mathbf{w} \in \mathbb{R}^d, \phi} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^n \ln \left( 1 + \exp \left( -y_i (\mathbf{w}^\top \phi(\mathbf{x}_i) + b) \right) \right)$$

- ▶ Want to learn  $\phi$
- ▶ Idea: Optimize also over  $\phi$ !
- ▶ Problem: the search space of all mappings  $\phi$  is too large... which restrictions to make?

# Core Idea to Learn a Good Feature Representation $\phi$

Let the learning machine figure it out!

Recall logistic regression:

$$\min_{b \in \mathbb{R}, \mathbf{w} \in \mathbb{R}^d, \phi} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^n \ln \left( 1 + \exp \left( -y_i (\mathbf{w}^\top \phi(\mathbf{x}_i) + b) \right) \right)$$

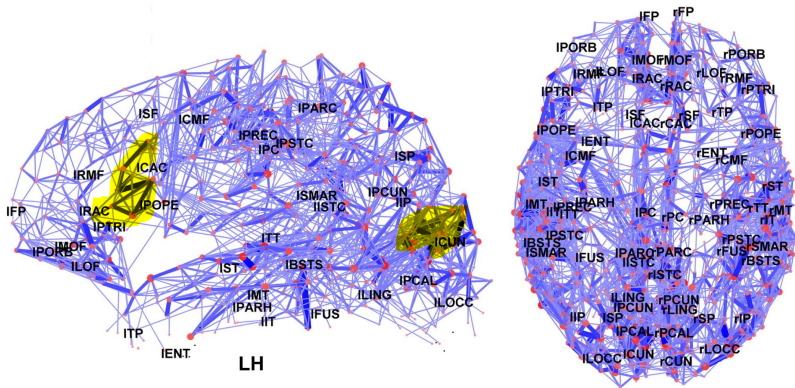
- ▶ Want to learn  $\phi$
- ▶ Idea: Optimize also over  $\phi$ !
- ▶ Problem: the search space of all mappings  $\phi$  is too large... which restrictions to make?

Idea: our brain also performs classification, for which it learns good feature representations ... **how does it do that?**

- 1 The Core Idea in a Nutshell
- 2 Natural Neural Networks**
- 3 Artificial Neural Networks
- 4 Convolutional Neural Networks

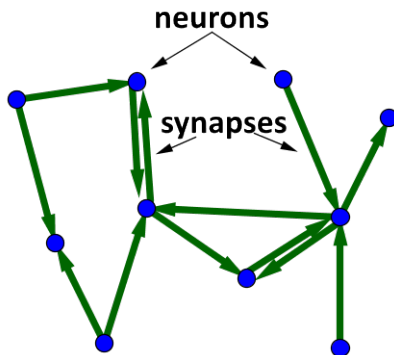
# How Does the Brain Work?

# How Does the Brain Work?



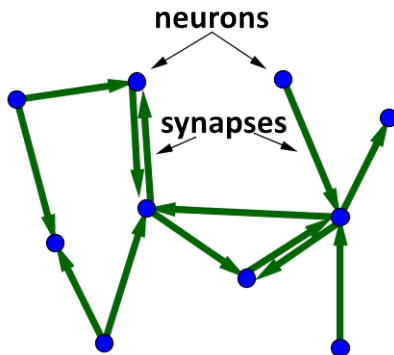
Think of it as a graph!

# Brain As Graph



- ▶ The nodes are called **neurons**
- ▶ The edges are called **synapses**

# Brain As Graph



- ▶ The nodes are called **neurons**
- ▶ The edges are called **synapses**

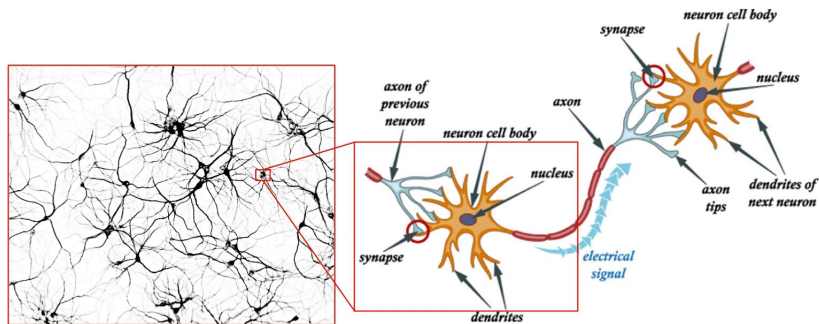
Another word for graph is **network** and our network consists of neurons. Therefore we have a **neural network**.



# Spikes and Potential

Whenever there is some action ongoing in our brain  
(e.g., we spotted a yummy box of chocolate)

- ▶ short pulses of electrical current (**Spikes**)  
shoot through our brain



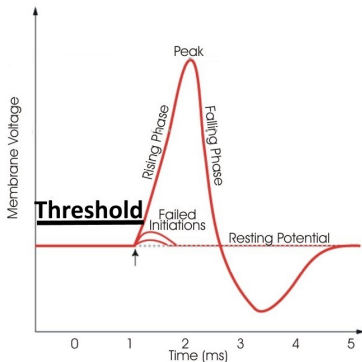
- ▶ Each neuron  $i$  receives spikes from a other neurons  $j$ 
  - ▶ the stronger the synapse  $W_{ij}$ , the higher the total current (called **potential**  $u_i$ ) that arrives at neuron  $i$

# Potential

Only if a neuron's potential  $u$  exceeds a threshold,

► (we say the neuron is **activated**)

it fires an impulse  $v$  (spike) to its neighbors in the neural network

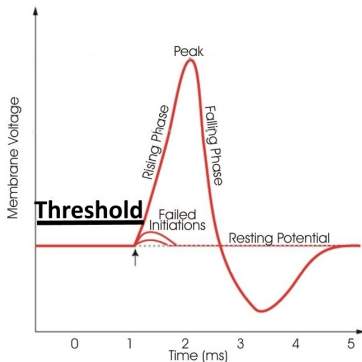


# Potential

Only if a neuron's potential  $u$  exceeds a threshold,

► (we say the neuron is **activated**)

it fires an impulse  $v$  (spike) to its neighbors in the neural network



Impulse can propagate from neuron to neuron  
through the entire brain

- 1 The Core Idea in a Nutshell
- 2 Natural Neural Networks
- 3 Artificial Neural Networks**
- 4 Convolutional Neural Networks

# Artificial Neurons

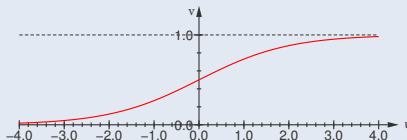
A classic neuron model is by McCulloch and Pitts (1950s):

## McCulloch and Pitts model

Denote by  $u$  the neuron's potential and by  $v$  the emitted spike ("activation"). Then:

$$v = \sigma(u)$$

where  $\sigma$  is the **sigmoid** function:  $\sigma(u) = \frac{1}{1+e^{-u}}$



# Artificial Neurons

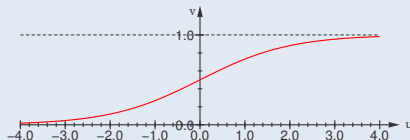
A classic neuron model is by McCulloch and Pitts (1950s):

## McCulloch and Pitts model

Denote by  $u$  the neuron's potential and by  $v$  the emitted spike ("activation"). Then:

$$v = \sigma(u)$$

where  $\sigma$  is the **sigmoid** function:  $\sigma(u) = \frac{1}{1+e^{-u}}$



ReLU :



Instead of the sigmoid function, today's ANNs usually use the **ReLU** activation function:  $\sigma(u) := \max(0, u)$

# Artificial Neural Networks (ANNs)

Let us propagate the activation from neuron to neuron:

- ▶ The  $j$ th neuron

- ▶ is connected to other neurons with connection strength  $w_{ij}$
- ▶ thus it has potential

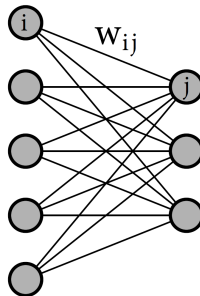
$$u_j = \sum_i w_{ij} v_i$$

- ▶ and activation

$$v_j = \sigma(u_j) = \sigma\left(\sum_i w_{ij} v_i\right)$$

- ▶ All neurons together have

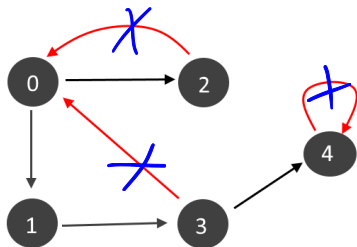
- ▶ potential  $\mathbf{u} = W^T \mathbf{v}$
- ▶ and activation  $\mathbf{v} = \sigma(\mathbf{u}) = \sigma(W^T \mathbf{v})$ .



Problem:  $\mathbf{v}$  is both on the left and right hand side of the equation (recursion)  $\Rightarrow$  mathematical nightmare! :)

Notation: for a vector  $\mathbf{u} = (u_1, \dots, u_d)^T$ , we define  $\sigma(\mathbf{u}) := (\sigma(u_1), \dots, \sigma(u_d))^T$ .  
The potential is also called pre-activation.

# We Therefore Want to Avoid Cycles



Cycles  
0->1->3->0  
0->2->0  
4->4

---

Although more difficult to deal with, there exist also cyclic ANNs, for instance, infinite impulse recurrent networks.



# Feed-forward ANNs

The resulting architecture is called **feed-forward** ANN or multi-layer perceptron (MLP).

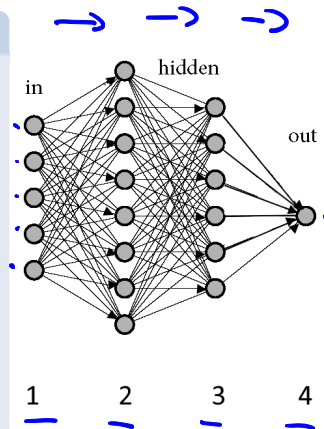
## Example

Shown to the right is a network with

- ▶ an input layer (5 nodes)
- ▶ two hidden layers (8 and 6 neurons, respectively)
- ▶ an output layer (one node)

We index the four layers as follows:

- ▶ 1: input
- ▶ 2,3: hidden layers
- ▶ 4: output layer



Recall :

$$\mathbf{v} = \sigma(W^T \mathbf{v}),$$

where:

- ▶  $\mathbf{v}$  is the vector of activations of all neurons in the network
- ▶  $W = (w_{ij})$  are the weights of the connections of the neurons.

Recall :

$$\mathbf{v} = \sigma(W^T \mathbf{v}),$$

where:

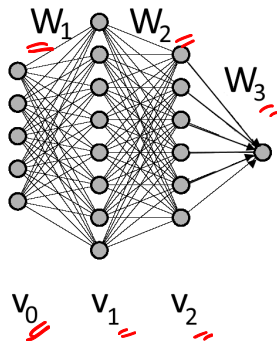
- ▶  $\mathbf{v}$  is the vector of activations of all neurons in the network
- ▶  $W = (w_{ij})$  are the weights of the connections of the neurons.

For a feed-forward ANN this simplifies:

$$\mathbf{v}_l = \sigma(\underbrace{W_l^T \mathbf{v}_{l-1}}_{=: \mathbf{u}_l}),$$

where:

- ▶  $\mathbf{v}_l$  is the vector of activations of the neurons in the  $l$ th layer
- ▶  $W_l = (w_{lij})$  are the strengths of the connections between neurons  $i$  in the  $l$ th layer and neurons  $j$  in the  $(l + 1)$ th layer.

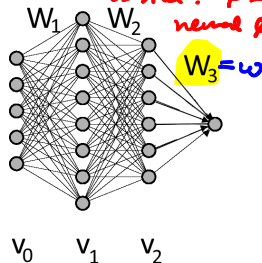


The computation in a feed-forward ANN can be thus be summarized as a nested function:

General:  $f(x) = \omega^T \phi(x)$   
 lin:  $\phi = \text{id}$   
 kernel:  $\phi = \phi_{\text{ker}}$   
 neural:  $\phi = \phi_{\text{w}}$

$$f(\mathbf{x}) := \underbrace{\mathbf{w}^T \phi_{\mathbf{w}}(\mathbf{x})}_{=\mathbf{u}_{L+1}}$$

$$\underbrace{\phi_{\mathbf{w}}(\mathbf{x})}_{=\mathbf{u}_{L+1}} := \sigma \left( \underbrace{W_L^T \sigma \left( \dots \sigma \left( \underbrace{W_1^T \underbrace{\mathbf{x}}_{=\mathbf{v}_0}}_{=\mathbf{u}_1} \right) \dots \right)}_{=\mathbf{v}_1} \right)_{=\mathbf{v}_L}$$



Remarks:

- ▶ We use a data point  $\mathbf{v}_0 := \mathbf{x}$  as the network's initial activation.
- ▶  $L$  denotes the number of hidden layers (example:  $L = 2$ ).
- ▶ For a network with one output node,  $\mathbf{w} := W_{L+1}$  is a vector.

# Aim

In order to use  $f$  for prediction, we need to:

- ▶ find weights  $\mathbf{w}, W_1, \dots, W_L$
- ▶ such that  $f(\mathbf{x}_i) \approx y_i$  on the training points.

# Aim

In order to use  $f$  for prediction, we need to:

- ▶ find weights  $\mathbf{w}, W_1, \dots, W_L$
- ▶ such that  $f(\mathbf{x}_i) \approx y_i$  on the training points.

How?

# Neural Network Learning

We wrap a learning machine around the network and let it figure out good network weights!

# Neural Network Learning

We wrap a learning machine around the network and let it figure out good network weights!

We do so as follows:



# Neural Network Learning

We wrap a learning machine around the network and let it figure out good network weights!

We do so as follows:

- ▶ As learning machine we take logistic regression (LR)

# Neural Network Learning

We wrap a learning machine around the network and let it figure out good network weights!

We do so as follows:

- ▶ As learning machine we take logistic regression (LR)
- ▶ In LR, we replace all occurrences of the inputs  $\mathbf{x}_i$  by  $\phi_W(\mathbf{x}_i)$

# Neural Network Learning

We wrap a learning machine around the network and let it figure out good network weights!

We do so as follows:

- ▶ As learning machine we take logistic regression (LR)
- ▶ In LR, we replace all occurrences of the inputs  $\mathbf{x}_i$  by  $\phi_W(\mathbf{x}_i)$
- ▶ We optimize over the weights  $W_1, \dots, W_L$  of the network!  
*and  $w$*

# Thus We Obtain:

## Feed-forward ANN

$$\min_{\mathbf{w}, W} \frac{1}{2} \|\mathbf{w}\|^2 + \frac{1}{2} \sum_{l=1}^L \|W_l\|_{\text{Fro}}^2 + C \sum_{i=1}^n \log(1 + \exp(-y_i \mathbf{w}^\top \phi_W(\mathbf{x}_i)))$$

where

- ▶  $W := (W_1, \dots, W_L)$
- ▶  $\phi_W(\mathbf{x}_i) := \sigma(W_L^\top \sigma(\dots \sigma(W_1^\top \mathbf{x}_i) \dots))$

---

Given a matrix  $M = (m_{ij}) \in \mathbb{R}^{m \times n}$ , the Frobenius norm is defined as  $\|M\|_{\text{Fro}}^2 = \sum_{ij} m_{ij}^2$

# Conclusion

## Artificial neural networks (ANN)

- ▶ motivated by how the brain works
- ▶ consisting of neurons organized in multiple layers with (feed-forward) connections
- ▶ learning feature representation (network weights) and classifier at the same time

Next video: ANNs on images (CNNs)

# References I



W. S. McCulloch and W. Pitts, A logical calculus of the ideas immanent in nervous activity, *The bulletin of mathematical biophysics*, vol. 5, no. 4, pp. 115–133, 1943.