Zusammenfassung Script Neural Nets

Table of Contents

Chapter 1: Introduction and motivation

Definitions

- *embodiment:* the brain is always embedded into a physical system that interacts with the real world
- *emergence*: how the global behavior of the brain-body system emerges from the activity of many individual units
- *graceful degradation:* a property of natural systems that moden computer lack to a large extent unless it is explicitily provided for. The term is used to designate systems that still operate – at least partially – if certain parts malfunction or if the situation changes in unexpected way. -> (a) noise and fault tolerant (b) able to generalize
- *paradox of the expert:*
  computer: umso mehr Daten vorhanden sind, umso länger geht eine Suche.
  brain: je mehr man weiss, umso schneller kann etwas bestimmtes gefunden werden.
- *connectism*: particular type of computational model consisting of many, relatively simple, interconnected units working in parallel.
- *emergent properties:* behaviors a neural network exhibits that were not programmed into the system (e.g. NETTalk)

Differences between computers and brains

Brains	**Computers**
Massively parallel	Largely serial
Learning system	Limited learning
Plasticity	No plasticity
Embodied	Not embodied (only in a trivial sense)
Robustness (fault & noise tolerance)	No fault tolerance
Content-adressable memory / assocative memory	RAM (Adressability)
Focus of attention	-
No such distinction	Seperation program/data
Inseperable	Seperation hardware/software
No central clock	Central clock
Analog	Digital
Consciousness	?
Ability of generalization	-
Tire (recover)	Works indefinetly
Relativly slow (~ms)	Fast (~10^ -9 s cycle times)
10^11 neurons, 10^14 synapses	
Forgetting	No forgetting

Terms

neuron: node, neuron, model neuron

dendrites: connection with propagation node
cell body: activation function / transfer function
spike (axon): output of node
synapses: connection weights / strenghts

Abstractions

- some kind of a clock which synchronizes all the activity in the network
- inputs to an neur are simply summed to yield a level of activation, whereas to model a real biological system one would have to take the precise arrival times of the incoming signals into account or one would have to assume a statistical distribution.
- the spikes are not modeled individually but only their average firing rate
- in natural brains there are different types of neurons
- the amount of time it takes for a signal to travel along the axon is neglected

NETTalk

- After some learning, the network starts to behave as if it had learned the rules of English pronunciation, even though there were no rules in the network.

Rosenblatt's perceptron

- clear similartities to what is called a perceptron in today's neural network literature, the feedback connections between the response cells and the association are normally missing.

General Applications

- 4 basic orientations
  - cognitive science/artificial intelligence
  - neurobiological modeling (develop model of biological neurons)
  - scientific modeling
  - computer science and its application (views NN as an interesting class of algorithms that has properties) -> Optimization, Control, Signal processing, Pattern recognition, Stock market prediction, classification problems

Chapter 2: Basic concepts

4 or 5 basics

- the characteristics of the node
  - normally the output is taken to be the level of activation
  - most widely used activation functions: linear function, linear threshold, sigmoid or logistic function
- the connectivity
  - how the individual nodes are connected to one another
  - connections in both directions can be used to implement some kind of memory
  - networks having connections in both directions are also called recurrent networks.
  - Nodes that have similair characteristics and are connected to other nodes in similar ways are

    sometimes called layer
  - for larger networks, the graph notation gets cumbersome and its better to use matrices -> connectivity matrix
  - positive connections are exciatory, negative ones inhibitory
  - w(ij) is the connection between node j and node i
  - the values of the input nodes are often called input vector
  - if the connectivity matrix has all zeros in the diagonal and above the diagonal, we have a feed-forward network
  - networks in which all nodes are connected to each other in both directions are called Hopfield nets
- the propagation rule
  - the propagation rule determines how activation is propagated through the network
  - it is so common that it is often not even mentioned
  - at each time step the activation is propagated through the net according to the propagation rule
- the learning rules
  - learning works through modification of the weight matrix
  - 3 types of learning
    - supervised learning
      - the correct output must be known -> the network must find the mapping
      - the larger the error the more the wights are changed -> error-correcting-learning
      - examples: perceptron learning rule, delta rule, backpropagation
    - reinforcement learning
      - blame assignement: the problem of attributing the error to the right cause
      - a particular behavior is to be reinforced
    - unsupervised learning
      - examples: hebbian learning & Kohonen networks
      - if two nodes are active simultaneously the connection between them is strenghtened
      - advantage: it requires only local information
- embedding the network in the physical system

Chapter 3: Simple perceptrons and adalines

The perceptron

- proposed by Rosenblatt (1958)
- learning ability of perceptrons is limited, if:
  - it contains only image points within a limited radius
  - it depends on a maximum of n image points

    - it contains random selection of the image points
    - => then i cannot learn the correct classification of sets of points for topological predicates such as circle, convex figure or connected figure
- the algorithm with the perceptron learning rule terminates after a finite number of iterations with constant increment, if there is a a configuration of weight for which the classification is correct. To reach this configuration, the classes have to be linearly separable
- linear separability: a plane can be found in the e-space separating the patterns for which the desired value is +1 and for which the desired value is 0.

Adalines

- there is a better learning rule which takes the size of the error into account -> delta rule (also called adaline rule)
- blame assingment: input large -> large contribution to error
- optimization techniques can be applied
- the delta rule realizes a gradient descent procedure in the error function
- existence of a solution (is there a set of weight so that all patterns can be learned?)
    - linear units
        - input pattern must be linearly independent
        - linear independence implies linear separability, but the reverse is not true
        - if the number of vectors is larger than the dimension, they are always linearly dependent
    - non linear units
        - the input pattern must be linearly independet
        - there may be local minima in the error function
- offline: the order in which the patterns appear is irrelevant
- online: the order in which the patterns are presented to the network matters
- cycle: 1 pattern presentation, propagate activation through network, change weights
- epoch: 1 „round“ of cycles through all the patterns

Chapter 4: Multilayer perceptrons and backpropagation

- MLP have one or several hidden layers of nodes
- the limits of simple perceptrons do not apply to MLPs
- a network with just one hidden layer can represent any Boolean function

Back-propagation algorithm

- thresholds gave been omitted. They can be taken care of by adding an extra input unit, connecting to all the nodes in the network and clamping it's value to (-1) -> the weights from this unit represent the threshold of each unit
- operations:
    1. Initialize weights to small random numbers

2. choose a pattern from the training set and apply it to the input layer
3. propagated the activation through the network
4. compute the deltas for the output layer
5. compute the deltas for the preceding layers by successively propagating the errors backwards
6. update all connections
7. go back to step 2 and repeat for the next pattern

- in the offline version the weights are changed only once all the patterns have been processed
- properties:
  - learning, not programming
  - generalization: it implies, that not every potential situation has to be predefined in the system
  - noise and fault tolerance -> resault of the massive parallelism
  - re-learning: fast re-learning -> if the network is distorted to a particular performance level it re-learns faster than a new network starting at the same performance level
  - emergent properties
  - universality: with a two-layer feed-forward network every set of discrete function values can be represented, or in a particular interval, a continous function can be approximated to any degree of accuracy
- problems:
  - there are performance problems
  - there are doubts whether supervised learning schemes like backpropagation have a biologival or psyhcological reality
  - users don't like black-boxes: there are no explicit rules that would give us an idea of what the network actually knows an does
- performance:
  - questions to ask
    - when has something been learned?
    - when is the network good?
    - how long does learning take?
  - Learning criterion
    - in general we need a global error measure E and we define a critical error E(0). Then the condition for learning is $E < E(0)$. Secondary we need a maximum deviation F(0)
    - when we have binary ouput units we can define that an example has been leaerned if the correct output has been produced by the network. If we can achieve that for all the patterns, we haven an error of 0.
- convergence (minimizes error function)
  - can be visualized using error surfaces -> system moves on the error surface to a local minimum
  - the error function depends not only on the data to be learned but also on the activation

functions

  - on of the problem with all gradient descent algorithms is that the may get stuck in local minima (this seems no to be the case with many dimensions) -> a random variable is added to the weights
  - convergence rates with back-propagation are typically slow -> many possible improvements (momentum, quickprop, adaptive parameters)
- architecture
  - influences on the performance
    - number of layers
    - number of nodes in hidden layer (if to small the network will not be able to learn the training examples. If too large, generalization will not be good)
    - connectivity: a priori information about the problem may be included here
  - the error on the test set is called the generalization error
  - if there are to many free parameters, the network will start overfitting the data set which leads to sub-optimum generalization
  - good strategy to avoid overfitting is to add noise to the data. The state space is better explored then

Cross Validation (CV)

1) one subset N(j) is removed
2) The network is trained on the remaining 9/10 of the data set until the error is minimized
3) the network ist tested on the data set N(j)

- CV is a quantitative measure for generalization
- if the network is large, then we have to use more training data to prevent overfitting

Optimal Brain damage

- uses information theoretic ideas to make an optimal selection of unnecessary weights which are removed

Cascade-Correlation

- supervised learning algorithm that builds its multilayer structure during learning
  1. starts with 0 hidden nodes. Train with the entire training set
  2. if the error no longer gets smaller we add a new hidden unit to the net
  3. the new unit is trained, its input weights are frozen and all output weights are once again trained. Repeat until the error is acceptably small
  4. if a hidden unit correlates positively with the error at a given unit, it will develop a negative connection weight to that unit, attempting to cancel some of the error
- adavantages:
  - the network architecture does not have to be designed beforehand
  - cascade-correlation is fast because each unit „sees“ a fixed problem and can move

decisively to solve that problem

  - cascade-correlation can build deep nets (for higher-order feature detectors)
  - incremental learning is possible
  - there is no nedd to propagate error signals backwards through the network connections

Modeling procedure

1) Can the problem be turned into a classification problem?
2) Does the problem requires generalization?
3) Is the mapping from input to ouput unknown?
4) Determine training and test set. Can the data be easily acquired?
5) Encoding at input. Is it straightforward? What kind of processing is required?
6) Encoding at ouput. Can it easily be mapped onto required solution?
7) Determine network architecture
8) determine performance measure

Applications

- research laboratories
- real-world applications
  - recognition of handwritten characters
  - mutual adaption between a prosthetic hand and patient
  - ALVINN (autonomous land vehicle)
    - classifying camera images
    - traditional methods seem to work better
  - Stock market prediction
    - combination of methods is used

Example: NETTalk

- translates english text into speech
- uses a multilayer feedforward backpropagation network model
- after presentation of many thousands patterns, the weights converge (= the network picks up the correct pronunciation)
- very robust: superimposing random distortions on the weights, removing certain connections in the architecture, and errors in the encoding do not significantly influence the network's behavior
- it can generalize: pronounce words correctly it has no encountered before
- emergent properties: consonant-vowel distinction
- not used in practice beache the enormous amount of structure in language is hard to extract with a monolythic neural network -> combinations of methods are use

Chapter 5: Recurrent networks

= networks with loops -> internal dynamics

- retain information about the past in terms of activation levels -> activation is preseverd for a certain amount of time -> it is passed back through the recurrent connections
- associative memories:
  - content-adressable memories
  - realisable with Hopfield nets
  - are capable of reconstructing patterns even if only a small portion of the entire pattern is available

Hopfield nets

- associative memory: store a set of patterns in such a way that when presented with a new pattern, the network responds by producing whichever one of the stored patterns most closley resembles the original pattern
- nodes are binary thresholds
- the weights are symmetric and the nodes are not connected to themselves
- propagation rule
  - the updating can be carried out essentially in two ways:
    - synchronously
      - requires a central clock
      - can lead to oscillations
    - asynchronously
      - is preferred because it is more natural and leads to more stable behavior
      - two ways
        - one unit is selected randomly for updating
        - each unit updates itself with a certain propability
- An associative memory with binary units and asynchronous updating, using Hebb's rule is called Hopfield model
- if the crosstalk term is zero, we have stability.
- Storage capacity
  - the number of symbols that can be stored in n bits is 2^n
  - in a Hopfield network it is on the order of 0.15 * N
- Boltzman machines:
  - can be seen as extension of Hopfield nets
  - a distinction is made between hidden an visible units, similarly to MLPs

energy function

- is the enable for making a connection between the physical world an dthe on of information processing or signal processing in neural networks

- it always decrease (or remains constant) as the system envolves according to the dynamical rule
- storing as many patterns as possible correspond to local minima of the energy function

Attractors

- three types of attractors
  - chaotic attractors: trajectories which wander in an uncorrelated way int the space of network states
  - limit cycles: trajectories which lead, rapidly, to small cycles of states
  - fixed points: trajectories which lead the network to dwll for an appreciable period on a single state
- attractors are a possibility to store an input pattern
- reserved state (or inversed attractor): an instance of the spurious state
- spurious states: attractors in the network that do not correspond to stored pattern -> can lead to errors of recognition or recall
  - the situation is no so bad because the spurious attractors are typically at higher levels in the landscape
  - with the addition of a certain noise level the spurious states can be destabilized while stored memories remain good attractors

other recurrent network models

- recurrent backpropagation
  - if we were interested in behavior int the real world -> sequence of pattern is necessary
  - hopfield nets haven been extended to include the possibility for storing and retrieving sequences
  - context layer is added
- continous time recurrent neural networks (CTRNN)
  - biological brains are not clocked -> they change continiously
  - forms a directed cycle

Chapter 6: Non-supervised networks

- in all networks so far, physical arrangement did no matter -> no metrik was defined
- in non-supervised networks the training sets are no longer pairs of input-desired output, but simply patterns
- clusters the pattern in certain ways

Competitive learning

- winner-take-all
  - the output nodes in this network compete for the activation
  - once they have an advantage they inhibt the other nodes in the layer through the negative connections and activate themselves through positives connections

  - after a certain time only one node int the output layer will be active and all the others inactive -> category is given
  - similar inputs leads to the same categorization
  - the winner is normally the unit with the largest input (the winner is the unit for which the normalized weight vector is most similar to the input vector)
- standard competitive learning rule
  1) start with small random values for the wight
  2) a set of input patterns is applied to the input layer in random order
  3) for each input find the winner in the output layer
  4) update the weights for the winning unit only. The weight vector is moved closer to the input vector. This in turn has the effect that next time, if a similar stimulus is presented, the node has an increased change of being activated
- vector quantization
  - categorize a given set or a distribution of input vectors into M classes and represent any vectory by the class into which it falls
  - if we transmit or store only the index of the class, rather than the input vector itself, a lot of storage space and transmission capacity can be saved. -> first a codebook has to be agreed on
  - the classes are represented by M prototype vectors and we find the class of a given input by finding the nearest prototype vector using Euclidean metric
  - Problem:
    - dead units: those units with weights vectors that are too far away form any of the input vectors that they never become active
    - Alternatives:
      - Adding noise with a wide distribution to the input vectors
      - initialize the weights with the inputs
      - update the weights in such a way that also the losers have a chance to leran, but with a lower learning rate -> leaky learning
      - bias subtraction: increase the threshold for those units that win often and lower the threshiold for those that have not been successful
      - not only the weights of the winner should be updated but also the weights belognging to a neighbour
    - Generally: reduce the learning rate successively
- Voronoi tessalations
  - divide space into polyhedral regions according to which of the prototype vectros is closest
  - the boundaries are perpendicular to the lines joining pairs of neighbouring prototype vectors

Feature mapping

- Topographic: neighbouring sensors are mapped onto nearby neural representations
- a larger number of neurons is allocated to regions on the body with high density of sensors
- redundancy principle: there should be partially overlapping functionality in sensory and motor

systems -> if some neurons are destroyed, adjacent ones can partially take over -> topographic maps increase the adaptivity of an organism

- use normal competitive learning with a slightly difference
  - mexikan hat: the connections near node j are exicatory (positive), the ones a bit further away inhibitory (negative), and the ones yet further away, zero
  - the nodes are on a grid and thus there is a notion of distance which can in turn be used to define a topology
  - through learning, neighbouring units learn similar reactions to input patterns. Nodes that are further away become sensitive do different patterns
  - Problems:
    - computationally intensive: many connections, a lot of iterations required before a stable state is reached
    - ==> new alternativ: Kohonen suggest a short-cut that has precisely the same effect
- Kohonen's Algorithm
  - neighbouring function which is 1 at the winning node and drops off with increasing distance
  - we start with a large region and a large learning rate and they are successively reduced
  - typical neighbouring function: Gaussian
  - advantage:
    - we do not have to deal with the complex network dynamics but the results are the same
    - Kohonen networks perform a dimensionality reduction (reducing a high-dimensional space to just a few dimension while preserving the topology)

Extended feature maps – robot control

- Kohonen maps are also useful for controlling systems that have to interact with the real world (e.g. robots)
- a extended version of Kohonen maps can decided what action should the system take given particular input data
- supervised Kohonen version: instead of providing the exact values for the output as in the supervised algorithm it is possible to provide only a global evaluation function, a reinforcement value

Hebbian learning

- there is no teacher in Hebbian learning but it does not have the competitive character of feature mapping
- with hebbian learning, the network learns, over time, how „typical“ a certain input vector is for the distribution P(vector)
  - the higher the probability the larger the output on average
- frequent patterns have a bigger influence on the weights than infrequent ones.
- To ensure that the weights are normalized we use Oja's rule
- active forgetting: forgetting is required because otherwise the weights would become too large over time. Forgetting only takes place when something is learned.