

Deep Learning - Lernen von vielschichtigen neuronalen Netzen

KJARTAN FERSTL

DIPLOMARBEIT

eingereicht am

Fachhochschul-Masterstudiengang

INFORMATION ENGINEERING UND -MANAGEMENT

in Hagenberg

im Juni 2014

Erklärung

Hiermit erkläre ich an Eides statt, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt und die aus anderen Quellen entnommenen Stellen als solche gekennzeichnet habe.

Hagenberg, am 11. Juni 2014

Kjartan Ferstl

Inhaltsverzeichnis

Erklärung	i
Vorwort	iv
Kurzfassung	v
Abstract	vi
1 Einleitung	1
1.1 Motivation	1
1.2 Ziele und Aufgaben	2
1.3 Abgrenzung der Arbeit	2
2 Grundlegendes	3
2.1 Definition	3
2.2 Entstehung	3
2.3 Hürden	4
2.4 Neuronale netze	4
2.5 Überwacht / Unüberwacht	5
3 Algorithmen	7
3.1 Backpropagation	7
3.2 Resctricted Boltzmann Maschines	8
3.2.1 Grundgedanke	8
3.2.2 Abkürzung	10
3.2.3 Begründung	10

Inhaltsverzeichnis	iii
3.3 Eignung	10
3.4 Autoencoders	10
3.5 Convolutional Neural Networks	11
3.6 betrachtung als restricted bolzman maschine	12
4 Anwendungen	13
4.1 Bilderkennung	13
4.1.1 Google Projekt	13
4.2 Spracherkennung	14
4.3 Fazit aus Anwendungen	14
5 Zusammenfassung	15
5.1 Ergebnisse	15
5.2 Allgemeines Resümee	15
5.3 Persönliches Resümee	15

Vorwort

Kurzfassung

An dieser Stelle steht eine Zusammenfassung der Arbeit, Umfang max. 1 Seite. Im Unterschied zu anderen Kapiteln ist die Kurzfassung (und das Abstract) üblicherweise nicht in Abschnitte und Unterabschnitte gegliedert. Auch Fußnoten sind hier falsch am Platz.

Abstract

This should be a 1-page (maximum) summary of your work in English.

Kapitel 1

Einleitung

1.1 Motivation

Komplexe neuronale Netze können bei einigen Problemstellungen, besonders in der Bild- und Spracherkennung ein sehr mächtiges Mittel zur Problemlösung sein. Das Lernen solcher Netze bringt Schwierigkeiten mit sich die in den vergangenen Jahrzehnten schlecht oder gar nicht gelöst werden konnten.

Aufgrund der Weiterentwicklung von PC-Hardware und der Verwendung der Grafik-Recheneinheit (GPU), sind seit den 00er-Jahren bereits bemerkenswerte Ergebnisse möglich. Momentan arbeiten sogar Chiphersteller an preiswerten, dedizierten Chips mit trainierbaren neuronalen Netzen. Qualcomm wird den ersten kommerziellen Chip mit einem integrierten neuronalen Netz noch dieses Jahr (2014) veröffentlichen. Neuronale Netze sind immer nur so gut, wie sie trainiert werden, eines der wesentlichsten Themen im weiteren Fortschritt von neuronalen Netzen sind daher die Algorithmen zum Trainieren.

Aus der eingeleiteten Motivation ergeben sich daher folgende Kernaufgaben für die Seminararbeit:

Google und Facebook werben Deep Learning Spezialisten an

<http://deeplearning.net/tag/deep-learning/>

<http://www.wired.com/2013/12/facebook-yann-lecun/>

<http://www.wired.com/2014/01/geoffrey-hinton-deep-learning>

querbeet information und übersicht <http://deeplearning.net/reading-list/>

- Einführung in die Problematik von Deep Learning mit kurzem Blick

auf die bisherige Geschichte

- Analyse der Hürden im Deep Learning
- Analyse aktueller Deep Learning-Algorithmen

1.2 Ziele und Aufgaben

Folgende zentrale Fragestellungen sollen in der Seminararbeit beantwortet werden:

- Was ist Deep Learning?
- Welche Mustertypen können erkannt werden und welche Anwendungen sind möglich
- Wie sind die gelernten Modelle konkret definiert?
- Welche Verfahren werden verwendet um vielschichtige Modelle zu lernen?
- Was ist aktuell mit Deep Learning in Verbindung mit neuronalen Netzen bei der Mustererkennung in Bildern möglich?

Ziel dieser Arbeit ist es, dem interessierten Leser einen Einstieg in Themen deep learning, mit besonderem Fokus auf neuronale Netze zu vermitteln. Aufbauend auf diesen Erkenntnissen soll gezeigt werden, welche wesentlichen Algorithmen bis heute entwickelt wurden und welche technischen Problemstellungen mit diesen Algorithmen gelöst werden können oder sogar bereits heute damit gelöst werden. Ein wesentlicher Fokus soll dabei auf die Mustererkennung in Bildern gerichtet werden.

1.3 Abgrenzung der Arbeit

write

Inhalt...

Achtung, im Rahmen der Verwendung von dinnat und natbib für das Literaturverzeichnis kann nicht `\cite` verwendet werden, sondern es muss `\citep` verwendet werden!

Beispiel: `[?]`, `[?]`, `[?]`, `[?]`, `[?]`, `[?]`.

Kapitel 2

Grundlegendes

2.1 Definition

eine teil aus dem Bereich der Maschine Learning-Algorithmen zum errechnen vieler Repräsentationen und Features durch die Berechnung nicht-linearer Informationen supervised und unsupervised feature extracaction

Deep Learning kommt aus dem Maschine Learning und befasst sich mit dem maschinellen lernen von vielschichtigen netzten, wie zum Beispiel vielschichtigen neuronalen Netzen oder Deep Belive Netzten.

2.2 Entstehung

Der Backpropagation Algorithmus wurde in den 1970ern und 1980ern mehrmals erfunden?! (Werbos, Amari?, Parker, LeCun, Rumelhart, ..)

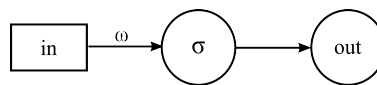
Backpropagation sah vielversprechend aus, aber in den 1990ern hat er an Bedeutung verlohren da die vielen hidden layer nicht trainiert werden konnten (rechenpower außer bei time-delay und convolutional nets) + hat nicht für netzwerke mit feedback funktioniert

negativ: benötigt gelabelte daten, ungelabelt stehen viel viel viel mehr zur verfügung negativ: zeit für lernen skaliert schlecht negativ: langsam bei netzwerken mit mehreren hidden layern negativ: bleibt oft in lokalen minima hängen, besonders bei vielschichtigen netzten!

frage ob deep-nets wirklich besser als wenigschichtige Netzte sind

supervised gegen unsupervised: ist unsupervised lerning nur ein behelf weil zu wenig gelabelte daten vorhanden sind oder ist eigentlich jedes lernen unsupervised und es scheitert nur an der Rechenleistung?

wie viel wissen sollten wir vorab in die daten stecken (kantenerkennung etc.),

**Abbildung 2.1:** Neuron

ist das netzt mit entsprechender Rechenpower nicht immer besser?

backpropagation-limitierungen mit notwendigkeit für gelabelte daten aufheben: mit unsupervised-learning beheben, gewichte anpassen wie in backpropagation gut, aber versuchen das eingangsbild aus den parametern wiederherzustellen, damit lernt das netz automatisch in unterster schicht wie bilder aussehen bevor es bilder identifiziert

backpropagation invented several times in the 1970 and 1980

heute besonders durch die vermehrte Speicherung von unkategorisierten Daten durch Unternehmen wie Google, Facebook und co. vermehrt unsupervised algorithmen und massiv viel rechenpower vorab rechenpower und datenmänge gewinnt gegen coolen algorithmus

2.3 Hürden

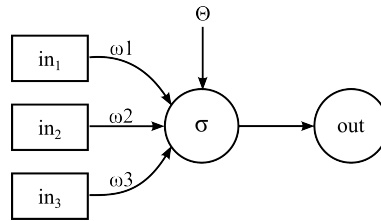
rotation, mutation der Natur

beschreibt dass es früher probleme gab, linzer in beteiligung:
<http://www.kurzweilai.net/how-bio-inspired-deep-learning-keeps-winning-competitions>

2.4 Neuronale netze

Neuronale Netze sind Strukturen aus der Technik, die dem Nervensystem von Lebewesen ähneln. Es sind Modelle, die Eingangsdaten über Neuronen gewichtet kombinieren und daraus einen Ausgang errechnen. Neuronale Netze sind in der Technik zur Vereinfachung meist in mehreren Schichten aufgebaut.

Abbildung 2.1 zeigt ein einfaches Neuron mit einem Eingang. Ein Eingang wird mit dem Gewicht multipliziert um dann anhand der Übertragungsfunktion den Ausgang zu berechnen. Als Übertragungsfunktion dient meist die Sigmoid-Funktion, welche sich einfach differenzieren lässt und sich daher besonders gut eignet. Der Ausgang *out* dieses Neurons ergibt sich somit aus der dem Eingang *in* multipliziert mit dem Gewicht ω in der Sigmoid-Funktion

**Abbildung 2.2:** Neuron mit mehreren Eingängen**Abbildung 2.3:** Layer von Neuronen

σ .

$$out = \sigma(\omega * in)$$

Ein Neuron hat in der Regel, wie in Abbildung 2.2 zu sehen, mehrere Eingänge. Außerdem wird noch ein Bias-Wert θ für die Sigmoid-Funktion hinzugefügt. Der Ausgang dieses Neurons kann somit wie folgt berechnet werden:

$$out = \sigma(\omega_1 * in_1 + \omega_2 * in_2 + \omega_3 * in_3 + \theta)$$

Um aus den einzelnen Neuronen ein Netzwerk zu machen, werden, wie in Abbildung 2.3 Layer gebildet. Ein Layer ist eine Menge an Neuronen, die untereinander nicht direkt verknüpft sind. Layer werden sequenziell hintereinander gehängt, wobei jedes Neuron eines Layers als Eingang für jedes Neuron des jeweils nächsten Layers dient. Ein solches Netzwerk ist auch in Abbildung 2.4 zu sehen. Dieses Netzwerk hat einen Layer für die Eingabedaten, einen für die Ausgabedaten und zwei innere Layer. Die inneren Layer werden als von außen Unsichtbar betrachtet und daher auch als Hidden-Layer bezeichnet.

2.5 Überwacht / Unüberwacht

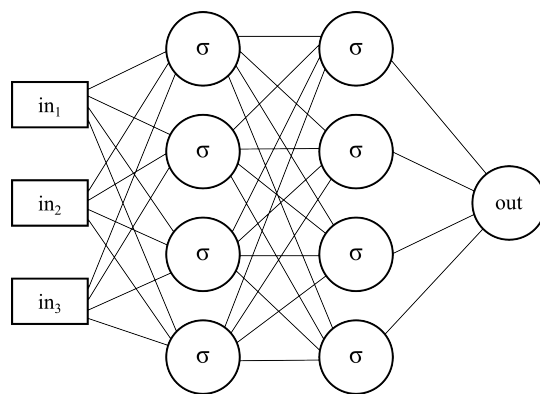


Abbildung 2.4: Ein Neuron

Kapitel 3

Algorithmen

Fokus: viele Daten, parallele Verarbeitung, viel Rechenpower

3.1 Backpropagation

Backpropagation ist ein überwachter Algorithmus zum Trainieren von vereinfachten neuronalen Netzen bei denen sich der Ausgang auf einfachem Weg berechnen werden kann, so genannten Feedforward-Perzeptrons, zu sehen in Abbildung ???. Überwacht bedeutet, dass das Netzwerk beim lernen anhand von kategorisierten Eingabedaten bestimmte Ausgabedaten zu errechnen. Der Fehler der Berechnung des Ausgangs wird beim Backpropagation-Algorithmus in die Adaptierung der Gewichte rückgeführt. Der Trainingsprozess beginnt mit zufälligen Gewichten und arbeitet in den folgenden Schritten:

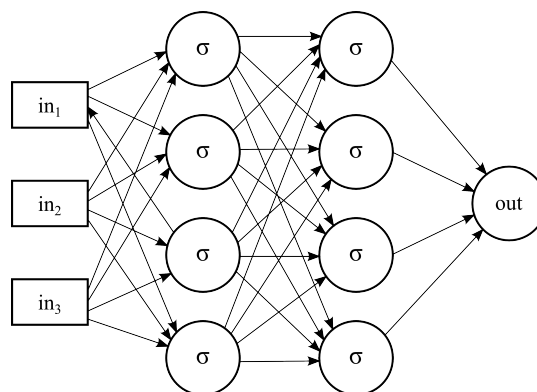


Abbildung 3.1: Ein Feedforward-Perzeptron

1. Ausgang des Netzes für bestimmte Eingabedaten berechnen
2. Ausgang mit dem Sollwert vergleichen und daraus den Fehler errechnen

$$error = \frac{1}{2} \sum_{k \in K} (out_k - target_k)^2$$

target ... Fehler, der später Einfluss in die Gewichte nimmt
out ... Wert den der Ausgang erreicht hat
target ... Wert den der Ausgang erreichen sollte
K ... Anzahl der Neuronen in der Ausgangsschicht

3. Gewichte je nach Größe des Fehlers anpassen

Diese Schritte werden wiederholt, bis die Gewichte so angepasst wurden, das Eingabedaten möglichst gut klassifiziert werden können. Die Anpassung der Gewichte errechnet sich im wesentlichen anhand der Gradienten der Fehler und Gewichte. Zudem nimmt Schicht, vor der sich das jeweilige Gewicht befindet, Einfluss.

Wie der Name des Algorithmus bereits verrät, wird der Fehler dabei im wesentlichen über eine mathematische Funktion in die Gewichte zurückgeführt. Diese Rückführung bringt zwei wesentliche Probleme mit sich:

- Lokale Minima
- Überanpassung

Wird der Fehler geringer, so wird auch die Anpassung geringer. Befindet man sich auf dem Weg in ein lokales Optimum, so kann dieses sehr weit weg von dem globalen Optimum sein. Ist das Optimum entsprechend groß, so ist die Wahrscheinlichkeit, dass der Algorithmus in diesem Optimum hängen bleibt groß. Abhilfe können hier Verfahren wie der Bergsteigeralgorithmus oder Zufallsstreuungen in den Werten schaffen.

(sigmoid training function als übertragungsfunktion zwischen neuronenvendet, e function, einfach zu differenzieren) -> output = sigmoid(gewicht*input+theta)
theta .. bias

viele inputs in ein neuron

3.2 Restricted Boltzmann Machines

Uneingeschränkte Neuronale-Netze sind schwierig und aufwendig zu trainieren. Restricted-Boltzmann-Maschinen, im weiteren auch als RBMs bezeichnet, sind eingeschränkte neuronale Netzwerke. Wie in Abbildung 3.1 dargestellt, existieren hierbei nur Verbindungen zwischen aneinander liegenden

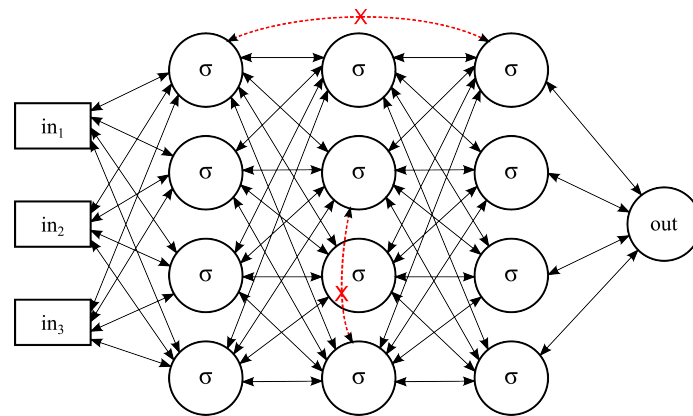


Abbildung 3.2: Eine Restricted-Boltzmann-Maschine

Schichten. Das Modell erlaubt keine Verbindungen von Neuronen zu sich selbst und alle Verbindungen müssen gleichermaßen in beide Richtungen gehen. So lassen sich unter Festlegung der Werte einer Schicht direkt auf die nächste versteckte Schicht weiter gerechnet werden. RBMs arbeiten in der Regel mit bool'schen Werten, lassen sich durch Erweiterung aber auch mit anderen Zahlenräumen verwenden.

RBMs wurden bereits 1986 von Paul Smolensky unter dem Namen Harmonium erfunden, fanden jedoch erst 1998, rund 10 Jahre später, durch die Entwicklung von effizienten Lernalgorithmen durch Geoffrey Hinton Anwendung.

ref

Zum trainieren von RBMs existieren verschiedene Algorithmen, die meist auf dem Prinzip beruhen, die Gewichte so anzupassen, das das hin und her Rechnen zwischen zwei Schichten wieder die Ausgangsdaten ergibt. Im folgenden wird der Algorithmus von Geoffrey Hinton, der zugleich als der erste effizienten Lernalgorithmus für RBMs gesehen wird, erklärt.

reference

3.2.1 Grundgedanke

Sind die Werte einer Schicht fixiert, so kann einfach auf die nächste versteckte Schicht weiter gerechnet. Zu beginn werden wird ein Trainingsdatenset an die Eingänge angelegt und die folgenden Operationen, wie auch in Abbildung 3.2 zu sehen, wiederholt ausgeführt:

1. Wahrscheinlichkeit für die versteckten Neuronen berechnen

$$p(h_j = 1) = \frac{1}{1 + e^{-(b_j + \sum_i (v_i * w_{ij}))}}$$

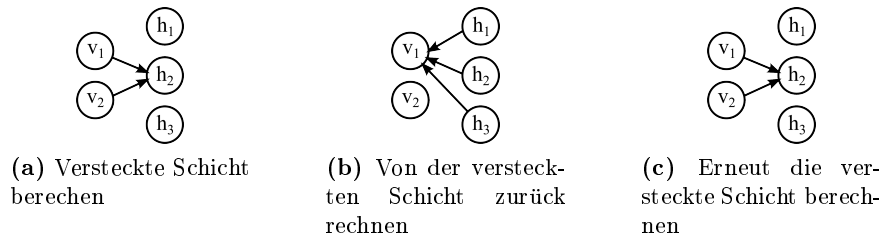


Abbildung 3.3: Drei Berechnungsschritte zum Lernen von RBMs.

2. Werte für die versteckte Schicht aus dem Mittelwert der Wahrscheinlichkeiten berechnen
3. Gradientenmatrix über das dyadische Produkt errechnen

$$\langle v_i h_j \rangle = v * y^T$$

4. Ausgehend von den errechneten versteckten Neuronen, zurück auf die sichtbare Schicht rechnen

Wiederholt man die in der Liste angeführten Schritte sehr oft, so pendeln sich für die Neuronen Werte ein, die im wesentlichen vom Modell und den den Wahrscheinlichkeiten abhängen, jedoch sehr wenig mit den Eingangsdaten zu tun haben. Anhand der Gradientenmatrix des letzten Durchlaufes lässt sich jedoch eine Distanz zum gewünschten Modell herausfinden und so können die Gewichte wie folgt angepasst werden:

$$\Delta \omega_{ij} = \varepsilon (\langle v_i h_j \rangle^0 - \langle v_i h_j \rangle^\infty)$$

Diese Art der Berechnung lieferte sehr gute Modelle. Durch die vielen Iterationen benötigt der Algorithmus sehr viel Rechenzeit und ist daher nicht praxistauglich. Zudem ist es schwierig festzustellen, wie viele Iterationen bis zum Einpendeln notwendig sind.

3.2.2 Abkürzung

Der oben genannte Algorithmus lässt sich in seiner Komplexität erheblich reduzieren, in dem die versteckte Schicht lediglich zwei mal berechnet wird. Geoffrey Hinton hat mit der Vereinfachung gezeigt, dass es möglich ist, bereits mit dem Vergleich der ersten und zweiten Gradientenmatrix möglich ist gute Ergebnisse zu erzielen. Diese Methode wird *Contrastive Divergence* genannt.

Die Regel zur Anpassung der Gewichte lautet dann:

$$\Delta \omega_{ij} = \varepsilon (\langle v_i h_j \rangle^0 - \langle v_i h_j \rangle^1)$$

3.2.3 Begründung

Der Grundgedanke von *Contrastive Divergence* ist, dass das Modell mit Zufallsgewichten weg von den Eingabedaten, hin zu Daten die ihm besser gefallen, wandert. Wenn man erkennt wo hin das Modell die Daten ändert, kann man die Gewichte so adaptieren, dass dem Modell die Eingabedaten am besten gefallen.

3.3 Eignung

Das Modell besitzt nicht genügend Neuronen um alle Eingabedaten zu speichern und muss daher, um die Eingabedaten tatsächlich reproduzieren zu können, etwas aus den Eingabedaten lernen. Es eignet sich daher, um aussagekräftige Merkmale aus unkategorisierten Daten zu extrahieren.

3.4 Autoencoders

Autoencoders: Neuronales netz, mit backpropagation trainiert, das zumindest in einer Schicht weniger Neuronen als Eingangsparameter besitzt (bottleneck). Es wird so trainiert, dass der Ausgang dem Eingang entspricht. Steckt man vorne ein Bild hinein und das Netzwerk produziert am Ausgang das gleiche Bild (mit allen Pixeln), so hat es offensichtlich etwas aus dem Bild gelernt und nicht nur alle Pixel durchgetunnelt (da Anzahl der Neuronen kleiner als Anzahl der Pixel). Es entsteht ein "Hash" für die Bilder - der wieder zurückgewandelt werden kann. Würde man es schaffen, beliebige Eingangsdaten sicher wiederherzustellen, so könnte man davon sprechen, dass das Netz das dahinter liegende System (die Natur) erlernt hat. Im weiteren könnte man so zum Beispiel von Bildern, Videos oder Audiospuren lediglich die Ergebnisse des Netztes speichern und die Eingangsdaten jederzeit wiederherstellen. So ließen sich diverse Daten unter Umständen erheblich komprimieren.

Füttert man solche Algorithmen mit Bildern und visualisiert die entstandenen Gewichte, so erkennt man, dass solche Algorithmen in erster Instanz meist Kanten und Farbintensitäten in Bildern finden. In der zweiten Schicht findet man einfache Kombinationen dieser Elemente und ab der dritten Schicht werden meist schon sehr brauchbare Neuronen zur vollständigen Objekterkennung ausgebildet.

Blockdiagramm mit mehreren Stufen (siehe Bilderkennung), zum Beispiel Matchen+Normalisieren, Matchen+Normalisieren, . . .

3.5 Convolutional Neural Networks

bild quelle: <http://www.image-net.org/challenges/LSVRC/2012/supervision.pdf>

inhalt quelle: wiki, <http://googlesystem.blogspot.co.at/2013/06/how-googles-image-recognition-works.html>

erläuterung für bilder: ein neuronales netzwerk, bestehend aus lauter kleineren gruppen an neuronen und vielen schichten. die kleinen Gruppen betrachten Ausschnitte des Bildes und überlappen in ihrer Anordnung. Durch den Blick auf den Ausschnitt eines Bildes wird das dahinter liegende Netzwerk in Teile geteilt und somit die Anzahl der Parameter und somit die Größe und benötigte Trainings und Rechenzeit stark verringert. Durch das Überlappen der einzelnen Bereiche wird berücksichtigt, dass ein späteres High-Level Feature nicht genau in einen Bereich trifft. Das Überlappen gibt es auch zwischen den einzelnen Layern. Convolutional networks können auch Layer enthalten, die mehrere Ausgänge von neuronnen kombinieren, so genannte pooling layer.

Dieser Typ von Netzwerk ist bereits 1980 , wurde in den folgenden Jahrzehnten optimiert und erlebte einen Aufschwung durch eine Implementierung auf einer GPU . Nach weiteren Optimierungen in den vergangenen Jahren wurde der Algorithmus unter anderem für ein Projekt von Google eingesetzt, das nicht zuletzt das so genannte face-neuron ausprägte.

convolution: operation that expresses the amount of overlap with another function image preprocessing mit gabor filters, erkennt kanten in alle richtungen

super
video, 10
minuten,
erklärt
das
ganze
thema:

https://www.youtube.com

bild

paper
Kunihiko
fukushi-
ma

Dan Ci-
resan

google
paper
finden

3.6 betrachtung als restricted bolzman maschine

Kapitel 4

Anwendungen

4.1 Bilderkennung

4.1.1 Google Projekt

google autoencoder, <https://www.youtube.com/watch?v=g4ZmJJWR34Q> 23:54

36:00 L2 polling, ignoriert dass ein neuron invertiert ausgelöst wird, zählt alles + auch wenn negativ, da die welt/bilder auch immer verdreht und manipuliert daher kommt, sprache muss akzente ignorieren, gesichts-erkennung die haarfarbet etc. auf der richtigen ebene an Features ist das daher ein sehr nützlicher faktor

local constrast normalization, hilft auch bilder invariationen zu ignorieren

google hat mit obrigen in den letzten jahren viel in der Bildersuche und youtube erreicht - was genau?, unsupervised learning mit einer unmänge an daten aus youtube videos

44:46+ bild: stages, drei stages mit jeweils in reihe: filtering, l2 polling, lcn (normalization step) - am ende haben einige neuronen muster erkannt, eines feuert bei gesichtern, eines bei katzen

49:02 number of parameters: 1 billion ... 200x200px bilder, 18x18 filters, 8 filters per location, l2 polling and lcn over 5x5 neighborhoods - wie viele neuronen gesamt?

wegen rechenpower nur mit sehr wenig pixel (200x200) gerechnet, unsere augen sehen $n \times n$ pixel, noch einiges möglich

the input image that maximises the neuron to fire: 53:00, das neuron war auf der 3. ebene

neuron auf 1. ebene sind edges, 2. ebene kombinationen von edges, 3. ebene muster wie das gesicht

IMAGENET, Datenbank mit gelabelten bildern an der sich viele Bench-
marken

erkennt kanten da farben von der beleuchtung abhängen und somit varriieren
lernen mittels back propagation

4.2 Spracherkennung

Deep Learning wird auf dem Gebiet immer stärker und hat die Methode des
Gaussian Mixture Models (GMM-Methoden) abgelöst

google now, apply siri, microsoft cortana

mit deep-nets große fortschritte

4.3 Fazit aus Anwendungen

Daten Daten Daten

Rechenpower Rechenpower Rechenpower

ergibt: Die großen gewinnen immer

Bessere Algorithmen schaffen dann am meisten wenn sie die Hardware
besser nutzen können (parallelisierung, datencenter) damit gewinnen wie-
der die großen

Kagel .. has data for machine learning algorithms, man löst erkenntnispro-
bleme für Firmen, deren ziel nicht immer ganz ein guter ist.

Was in anderen Bereichen passiert und passieren könnte

Dimensionen in der Zukunft, wo könnte es hin gehen

Traffic sign recognition (99,46prozent, besser als menschen! gewinner <http://www.kurzweilai.net/how-bio-inspired-deep-learning-keeps-winning-competitions>)

Kapitel 5

Zusammenfassung

5.1 Ergebnisse

5.2 Allgemeines Resümee

5.3 Persönliches Resümee